CRANFIELD UNIVERSITY
LIBRARY

CRANFIELD UNIVERSITY

GOH KIAH MOK

A SYSTEM DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS

SCHOOL OF APPLIED SCIENCES

EngD THESIS

ProQuest Number: 10832196

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.



ProQuest 10832196

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by Cranfield University.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

CRANFIELD UNIVERSITY

SCHOOL OF APPLIED SCIENCES

EngD THESIS

ACADEMIC YEAR 2005 – 2009

GOH KIAH MOK

A SYSTEM DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS

SUPERVISORS: DR. B. TJAHJONO AND PROF. T.S. BAINES

JULY 2009

This thesis is submitted in partial fulfilment of the requirements
for the Degree of Engineering Doctorate

© Cranfield University 2009. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright holder.

ABSTRACT

In recent years, Singapore's manufacturing sector has contributed more than a quarter of the total Gross Domestic Product (GDP) and has established global leadership positions in several manufacturing areas such as electronics, Information Technology (IT) and industrial automation. The Singapore Economic Review Committee (ERC) recommendation states that "software and embedded systems that drive products are one of the most important technologies for the manufacturing sector. "

With the increasing adoption of automated and intelligent products, embedded systems have emerged as a crucial technology for Singapore. However, the development of embedded applications is not a trivial undertaking as it can usually involve multi-discipline parties and different application platforms. Most embedded application developments use either vendor specific or desktop-based methodologies. Vendor specific methodologies constrain the company to rely on the specific vendor's solutions, whereas desktop-based methodologies are not well suited to embedded application development. Therefore, this research aims to develop a standard-based system development methodology for embedded applications.

The research programme comprises 5 stages. The first stage reviews the existing system development methodologies for embedded applications. The next stage formulates the proposed conceptual methodology followed by the development of the proof-of-concept tool to demonstrate the merits of the proposed approach. The methodology is then tested and evaluated respectively by using industrial experiments and feedback from a workshop. The final stage refines the methodology based on the feedback and presents the final system development methodology. The research has provided a sound foundation which future research in methodology for embedded applications to develop further.

PUBLICATIONS

Conference papers

1. K. M. Goh, B. Tjahjono, T. Baines, S. Subramaniam. (2006). A Review of Manufacturing Prognostic Research, 4th IEEE International Conference on Industrial Informatics 2006 (INDIN 06), 16-Aug-2006 to 18-Aug-2006, Singapore, pp. 417-422.
2. K. M. Goh, B. Tjahjono and A. J. R. Aendenroomer. (2007). A Rapid Configurable Embedded Development Methodology, IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 25-28 Sep 2007, Patras - Greece, vol. 1, pp. 135-140.
3. K.M. Goh, B. Tjahjono, T. Baines, J. Ang, (2008). Literature Review of Research in Manufacturing Prognostics, Cranfield Multi-Strand Conference: Creating Wealth Through research and Innovation (CMC 2008), Cranfield University 6-7 May 2008.
4. K.M. Goh, B. Tjahjono, A, Manaf, (2008). An IEC 61499-based Run-to-run Controller for Chemical Mechanical Planarisation Process, 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 08), Hamburg, Germany, September 15-18, 2008, pp. 25-28.
5. K.M. Goh, B. Tjahjono, J. Joseph, (2008). Automation Objects for Product Service System Enabler Device, 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 08), Hamburg, Germany, September 15- 18, 2008, pp. 50-53.
6. K.M. Goh, B. Tjahjono, W. Ding, (2009). Iterative Knowledge Based Code Generator for IEC61499 Function Block, TENCON 2009, Singapore, 23-26 Nov 2009.

Journal papers

7. K. M. Goh, B. Tjahjono, T. Baines, (2009). Towards a Rapid Configurable Embedded Development for Manufacturing Prognostics: A Review and Proposed Framework, International Journal of Condition Monitoring and Diagnostic Engineering Management (COMADEM), Invited journal paper, September 2008. Submitted.
8. K. M. Goh, B. Tjahjono, W. Ding, (2009). Iterative EBNF Model for Embedded Applications, IEEE Transactions on Industrial Informatics, Special Section "Model-based Approaches for Embedded Systems", Submitted, Expected publication date August/November 2010.

Technical report

9. K. M. Goh, B. Tjahjono, T. Baines, (2009). Design of a new IEC 61499-based embedded system development framework, SIMTech Quarterly Technical Report, Singapore Institute of Manufacturing Technology, July 2009.

Book Chapter

10. K. M. Goh, B. Tjahjono (2009). IEC 61499-Based System Development Methodology for Reconfigurable Embedded Application, "Reconfigurable Embedded Control Systems: From Modelling To Implementation", IGI Global, Invited book chapter submitted, Scheduled for release in 2011.

ACKNOWLEDGEMENTS

The author would like to thank and acknowledge the following groups of people for their help and assistance through the first year of this EngD research project and for their help in the production of this report:

Cranfield University:

Dr. Benny Tjahjono, Prof. Tim Baines, academic supervisors, Mr. Rob Lambert, MBA supervisor and Dr. Rick Greenough, chairman of internal review panel for their invaluable advice, guidance, direction and support throughout the year.

SIMTech:

Dr. Fong Aik Meng, my industrial supervisor for his encouragement, support, advice and sharing of the potential applications.

Mr. Wong Ming Mao and Dr. Zhang Jing Bing, our group managers for their support, advice, and direction.

Dr. Roland Lim, Dr Anton Aendenroomer, Associate Professor Ling Keck Voon and Embedded Controller Technology team members for various brainstorming, discussion and sharing of the research trend, gaps and needs.

Family:

My parents, my wife Janet, my son Ryan, my siblings and their family for their encouragement and support. Special memory and thanks to both my father and my second sister who continuously gave me support and encouragement even before the last minute of their life.

CONTENTS

ABSTRACT I

PUBLICATIONS II

ACKNOWLEDGEMENTS IV

CONTENTS V

LIST OF FIGURES IX

LIST OF TABLES XII

GLOSSARY OF TERMS XIII

CHAPTER 1 INTRODUCTION 1

1.1 RESEARCH BACKGROUND 2
1.2 OVERVIEW OF THE RESEARCH AIMS, OBJECTIVES AND PROGRAMME 2
1.3 OVERVIEW OF RESEARCH CONTRIBUTIONS 3
1.4 STRUCTURE OF THE THESIS 3

CHAPTER 2 INDUSTRIAL CONTEXT 4

2.1 MANUFACTURING WITHIN SINGAPORE'S ECONOMY 5
2.2 CHALLENGES FACING THE ELECTRONICS CLUSTER IN SINGAPORE 6
2.3 OVERVIEW OF THE ELECTRONICS CLUSTER IN SINGAPORE 8
2.4 GOVERNMENT INITIATIVES 9
2.5 THE NEED FOR A DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS 11
2.6 CHAPTER SUMMARY 12

CHAPTER 3 LITERATURE REVIEW: EMBEDDED APPLICATION DEVELOPMENT 14

3.1 SEARCH STRATEGY 15
3.2 RESULTS AND ANALYSIS 16
3.3 KEY FINDINGS AND OUTSTANDING ISSUES 21
3.4 DEFINITION OF A SYSTEM DEVELOPMENT METHODOLOGY 21
3.5 DEFINITION OF AN EMBEDDED SYSTEM 21
3.6 COMPLEXITY AND RAPID GROWTH OF EMBEDDED APPLICATION 23
3.7 MERGING OF HARDWARE AND SOFTWARE 23
3.8 SOFTWARE COMPONENTS, RE-USABILITY, DESIGN PATTERN AND CODE GENERATION 24
3.9 RESEARCH FOCUS ON STANDARDS LIKE SYSTEMC, UML, FUNCTION BLOCK 26
3.10 EVALUATION CRITERIA 28
3.11 THE NEED FOR AN EMBEDDED APPLICATION SYSTEM DEVELOPMENT METHODOLOGY 30
3.12 CHAPTER SUMMARY 32

CHAPTER 4 RESEARCH PROGRAMME 33

4.1 INDUSTRIAL PROBLEMS AND THE RESEARCH CONTEXT 34

4.2 RESEARCH AIM AND OBJECTIVES 35
4.3 RESEARCH PROCESS 35
4.3.1 Stage 1: Review of the existing system development methodologies 37
4.3.2 Stage 2: Formulate the conceptual methodology 38
4.3.3 Stage 3: Develop of the proof-of-concept tool 38
4.3.4 Stage 4: Evaluate the methodology 39
4.3.5 Stage 5: Refine final methodology 39
4.4 CHAPTER SUMMARY 42

CHAPTER 5 REVIEW OF THE EXISTING SYSTEM METHODOLOGIES 43

5.1 RESEARCH METHOD 44
5.2 EMBEDDED SYSTEM HARDWARE PLATFORMS AND APPLICATION DEVELOPMENT PROCESSES 45
5.2.1 Microprocessor 45
5.2.2 FPGA 45
5.2.3 ASIC 46
5.2.4 FPGA and ASIC development process 46
5.2.5 Microprocessor and FPGA development process 47
5.3 REVIEW OF SYSTEM DEVELOPMENT METHODOLOGIES 49
5.3.1 UML-based system development methodologies 49
5.3.2 IEC 61499-based system development methodologies 56
5.3.3 Other methodologies 60
5.4 ANALYSIS AND DISCUSSION 60
5.4.1 Strengths of current research 61
5.4.2 Weaknesses of current research 62
5.4.3 Opportunities from current research 64
5.5 CHAPTER SUMMARY 64

CHAPTER 6 FORMULATION OF THE CONCEPTUAL METHODOLOGY 69

6.1 METHOD 70
6.2 DETERMINATION OF THE DEVELOPMENT STEPS 71
6.2.1 Common development steps 71
6.2.2 Other development steps 71
6.3 CONCEPTUAL METHODOLOGY 75
6.3.1 Requirements definition 76
6.3.2 Detail design 76
6.3.3 Design repository 76
6.3.4 Design representation 77
6.3.5 Code generation 77
6.3.6 Implementation 77
6.3.7 Target application 77
6.4 CHAPTER SUMMARY 78

CHAPTER 7 DEVELOPMENT OF A PROOF-OF-CONCEPT TOOL 80

7.1 METHOD 81
7.2 SYSTEM ARCHITECTURE DESIGN 82
7.3 EXAMINATION AND DEVELOPMENT OF THE PROOF-OF-CONCEPT TOOL 85
7.3.1 IEC 61499 Function Blocks: 85
7.3.2 eXtended Markup Language (XML): 91
7.3.3 eXtended Style Sheet Language (XSL): 91
7.3.4 XML parser 93
7.3.5 SystemC 94
7.3.6 SystemC to Verilog 95
7.4 VERIFICATION OF THE TOOLS 96
7.4.1 Step A: Requirements definition 96

7.4.2 *Step B: Detail design* ... 96
7.4.3 *Step C: Design repository* ... 98
7.4.4 *Step D: Design representation* ... 98
7.4.5 *Step E: Code generation* ... 98
7.4.6 *Step F: Implementation* ... 99
7.4.7 *Step G: Target application* ... 100
7.5 DISCUSSION ... 101
7.6 CHAPTER SUMMARY ... 102

CHAPTER 8 EVALUATION OF THE METHODOLOGY ... 104

8.1 METHOD ... 105
8.2 SELECTION OF COMPANIES ... 107
8.2.1 *Understanding the manufacturing process* ... 107
8.2.2 *Selection criteria* ... 108
8.2.3 *Selection of industrial companies for workshop* ... 109
8.3 INDUSTRIAL EXPERIMENTS ... 109
8.3.1 *Experiment 1: Polishing time estimator* ... 110
8.3.2 *Experiment 2: Fast Fourier Transforms(FFT) with compression* ... 124
8.3.3 *Experiment 3: PID controller* ... 134
8.3.4 *Summary of the evaluation of the experiments* ... 146
8.4 WORKSHOP AND FEEDBACK ... 149
8.4.1 *Feedback questionnaire* ... 149
8.4.2 *Analysis of the feedback results* ... 153
8.5 ANALYSIS AND DISCUSSION OF RESULTS ... 157
8.5.1 *Strengths of the methodology* ... 157
8.5.2 *Weaknesses of the methodology* ... 159
8.5.3 *Summary of opportunities for further refinement* ... 159
8.6 CHAPTER SUMMARY ... 160

CHAPTER 9 REFINEMENT OF THE FINAL METHODOLOGY ... 161

9.1 METHOD ... 162
9.2 IMPROVING THE CODE GENERATION ... 163
9.2.1 *XML tree* ... 163
9.2.2 *Extended Backus–Naur Form* ... 164
9.2.3 *Code generation* ... 165
9.2.4 *Example of converting XML to SystemC* ... 168
9.3 THE FINAL SYSTEM DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS ... 169
9.3.1 *The input* ... 170
9.3.2 *The process* ... 170
9.3.3 *The technique* ... 170
9.3.4 *The output* ... 171
9.4 DESIGN THE GUIDE FOR THE METHODOLOGY ... 173
9.4.1 *Step A: Requirements definition* ... 173
9.4.2 *Step B: Detail design* ... 176
9.4.3 *Step C: Design repository* ... 181
9.4.4 *Step D: Code generation* ... 183
9.4.5 *Step E: Implementation for FPGA-based solution* ... 186
9.4.6 *Step E: Implementation for microprocessor-based solution* ... 188
9.5 CHAPTER SUMMARY ... 190

CHAPTER 10 CONCLUSIONS ... 191

10.1 OVERVIEW OF RESEARCH THE AIM, OBJECTIVE AND PROGRAMME ... 192
10.2 RESEARCH FINDINGS ... 193
10.3 RESEARCH CONTRIBUTIONS ... 194

10.4 LIMITATIONS OF THE RESEARCH 197
10.4.1 Limitations of the research programme 197
10.4.2 Limitations of research findings 198
10.4.3 Sample size in the workshop 198
10.5 DIRECTIONS FOR FUTURE RESEARCH 199
10.5.1 Opportunities for enhancing the proposed system development methodology for embedded applications 199
10.5.2 Opportunities for further testing 199
10.6 CONCLUDING REMARKS 200

REFERENCES 201

APPENDIX A: INTRODUCTION TO FUNCTION BLOCK PROGRAMMING 215

APPENDIX B: PROOF-OF-CONCEPT TOOL INSTALLATION INSTRUCTIONS 226

APPENDIX C: EBNF CODE GENERATOR SETUP 238

APPENDIX D: FEEDBACK FORMS FROM POTENTIAL USERS 244

LIST OF FIGURES

FIGURE 2-1: SINGAPORE 2006 GDP [EDB, 2008] 5
FIGURE 2-2: KEY BUSINESS CHALLENGES, 2005 [WDA, 2006] 6
FIGURE 2-3: AVERAGE TRAINING COST (IN SGD) PER TRAINEE BY INDUSTRY, 2003 & 2004 [WDA, 2006] 7
FIGURE 2-4: THE INTRICATE RELATIONSHIP BETWEEN MICROELECTRONICS AND EMBEDDED SYSTEMS IN SINGAPORE [LIM, 2007] 8
FIGURE 4-1: RESEARCH PROCESS FOR SYSTEMS DEVELOPMENT [ADAPTED FROM NUNAMAKER ET AL, 1991] 36
FIGURE 4-2: RESEARCH PROGRAMME 40
FIGURE 4-3: DECOMPOSITION OF STAGE 3 AND STAGE 4 41
FIGURE 5-1: UML TO SYSTEM C [TAN ET AL, 2004] 49
FIGURE 5-2: UML TO HDL [COYLE & THORNTON, 2005] 51
FIGURE 5-3: UML TO SYSTEMC [BOUDOUR & KIMOUR, 2006] 52
FIGURE 5-4: MDA-BASED SOC DESIGN FLOW [WANG ET AL, 2006] 53
FIGURE 5-5: UML-BASED MULTIPROCESSOR SOC SYSTEM DEVELOPMENT METHODOLOGY [KANGAS ET AL, 2006] 54
FIGURE 5-6: THE STATECHARTS DIAGRAM [MURA ET AL, 2007] 55
FIGURE 5-7: THE MODEL-VIEW-CONTROLLER [BRENNAN ET AL, 2004] 57
FIGURE 5-8: SAMPLE ARCHITECTURE OF AUTOMATION COMPONENTS [CENGIC ET AL, 2006] 58
FIGURE 5-9: PROPOSED MODEL-BASED DESIGN METHODOLOGY [MARIAN & GUO, 2006] 59
FIGURE 5-10: PROPOSED SEMANTIC WEB-BASED METHODOLOGY [THAMBOULIDIS ET AL, 2007] 60
FIGURE 6-1: PROPOSED SYSTEM DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS 75
FIGURE 7-1: STAGE 3 - DESIGN OF THE SYSTEM ARCHITECTURE AND DEVELOPMENT OF THE PROOF-OF-CONCEPT TOOL 82
FIGURE 7-2: THE PROPOSED SYSTEM ARCHITECTURE 84
FIGURE 7-3: EXECUTION MODEL [FISCHER & BOUCHER, 2008] 86
FIGURE 7-4: BASIC FUNCTION BLOCK TYPE 87
FIGURE 7-5: EXECUTION CONTROL CHART 88
FIGURE 7-6: COMPOSITE FUNCTION BLOCK INTERFACE 88
FIGURE 7-7: COMPOSITE FUNCTION BLOCK IMPLEMENTATION (GRAPHICAL CONSTRUCTION) 89
FIGURE 7-8: TEXTUAL DESCRIPTION 89
FIGURE 7-9: A SERVICE INTERFACE FUNCTION BLOCK 91
FIGURE 7-10: XSL FOR DIFFERENT LANGUAGE FORMATS 92
FIGURE 7-11: XML DOCUMENT FOR OR OPERATION 92
FIGURE 7-12: XSL WHICH CONVERTS XML TO C 93
FIGURE 7-13: LANGUAGE COMPARISON AND GENERATION 95
FIGURE 7-14: MODELLING THE SWITCH AND LIGHT SYSTEM 97
FIGURE 7-15: SIMULATED DIGITAL WAVEFORM 100
FIGURE 7-16: TESTING SWITCH AND LIGHT SYSTEM 100
FIGURE 7-17: SWITCH AND LIGHT SYSTEM IMPLEMENTED ON AN FPGA 101
FIGURE 8-1: STAGE 4 – EVALUATION AND TESTING 106
FIGURE 8-2: SEMICONDUCTOR PRODUCTION OVERVIEW 107
FIGURE 8-3: CMP OPERATION (SEQUENCE OF STEPS) 111
FIGURE 8-4: PHYSICAL SYSTEM FOR THE POLISHING TIME ESTIMATOR 114
FIGURE 8-5: SYSTEM SKETCH FOR THE POLISHING TIME ESTIMATOR 115
FIGURE 8-6: THE REMOVAL RATE CONTROL LOOP FOR PREVIOUS LOTS 117
FIGURE 8-7: FUNCTION BLOCK IMPLEMENTATION – REMOVAL RATE 117
FIGURE 8-8: POLISH TIME CALCULATION (USING THE PREVIOUSLY CALCULATED REMOVAL RATE) 119
FIGURE 8-9: FUNCTION BLOCK IMPLEMENTATION – POLISH TIME 121
FIGURE 8-10: REMOVAL RATE CALCULATION MODULE 121
FIGURE 8-11: POLISH TIME CALCULATION MODULE 122
FIGURE 8-12: ACTUAL VS. CALCULATED POLISHING TIME WITH A BIAS OF 0.5 SEC. 123
FIGURE 8-13: PROPOSED DEVICE MODEL 126
FIGURE 8-14: ENCAPSULATE JAVA ALGORITHMS IN FUNCTION BLOCK 129
FIGURE 8-15: FFT WITH COMPRESSION AND RECONSTRUCTION 131
FIGURE 8-16: OUTPUT VALUES OF FFT 132

FIGURE 8-17: COMPRESSION AND RECONSTRUCTION 132
FIGURE 8-18: GRAPHICAL REPRESENTATION OF THE CTRL FUNCTION BLOCK 140
FIGURE 8-19: GRAPHICAL REPRESENTATION OF THE TANK FUNCTION BLOCK 141
FIGURE 8-20: FUNCTION BLOCK REPRESENTATION OF PID CONTROLLER 143
FIGURE 8-21: SIMULATION OF PID_TANK SYSTEM 144
FIGURE 8-22: MAIN INDUSTRIAL SECTORS 153
FIGURE 8-23: YEARS OF EXPERIENCE IN SOFTWARE DEVELOPMENT OF PARTICIPANTS. 154
FIGURE 9-1: STAGE 5 – REFINED FINAL METHODOLOGY 163
FIGURE 9-2: XML TREE CONSTRUCTION 164
FIGURE 9-3: INTERACTIVE CODE GENERATOR 165
FIGURE 9-4: SIMILAR ATTRIBUTE IDENTIFICATION 166
FIGURE 9-5: SIMILAR NODE SEARCHING 167
FIGURE 9-6: DATA TYPE MATCHING 168
FIGURE 9-7: INTERACTIVE CODE GENERATOR GUI 169
FIGURE 9-8: FINAL SYSTEM DEVELOPMENT METHODOLOGY 169
FIGURE 9-9: CONTENT FOR THE SYSTEM DEVELOPMENT METHODOLOGY FOR EMBEDDED APPLICATIONS. 172
FIGURE 9-10: TOP-DOWN SKETCH 174
FIGURE 9-11: SYSTEM MODELS 177
FIGURE 9-12: DESCRIPTION OF THE PHYSICAL DEVICE 178
FIGURE 9-13: DESCRIPTION OF THE SYSTEM 179
FIGURE 9-14: DESCRIPTION OF THE IMPLEMENTATION 180
FIGURE 9-15: NEW SYSTEM ARCHITECTURE 184
FIGURE 9-16: EBNF PARSER 185
FIGURE 9-17: THE PROCESS FOR FPGA CODE GENERATION 187
FIGURE 9-18: THE PROCESS FOR MICROPROCESSOR CODE GENERATION 189
FIGURE A-1: AN INTRODUCTION TO THE BASIC COMMANDS IN FBDK. 217
FIGURE A-2: SIDE WINDOWS EXPLANATION 217
FIGURE A-3: EXPLANATION OF THE DEVICE DISPLAY DURING RUN ON THE MAIN WINDOW 218
FIGURE A-4: SIMPLE LIGHT SWITCH EXAMPLE 219
FIGURE A-5: CONVERSION FROM THE ALGORITHM FLOWCHART TO THE LOGIC GATE DIAGRAM 219
FIGURE A-6: ILLUSTRATED STEPS OF NEW FILE CREATION 220
FIGURE A-7: SAVING AND LOADING FILES IN FBDK 220
FIGURE A-8: DEVICE EDITING - ILLUSTRATED INSTRUCTIONS 221
FIGURE A-9: RESOURCE EDITING IN FBDK 222
FIGURE A-10: ADDITION OF FUNCTION BLOCKS INTO THE SYSTEM 222
FIGURE A-11: EVENTS LINE CONNECTIONS IN FBDK 223
FIGURE A-12: DATA FLOW CONNECTIONS IN FBDK 223
FIGURE A-13. ASSIGNING DATA INPUTS 224
FIGURE A-14: MAPPING FROM THE FUNCTION BLOCK TO THE RESOURCE 224
FIGURE A-15: PROGRAM RUN RESULTS 225
FIGURE B-1: LAUNCHING THE SWITCH AND LIGHT SYSTEM 227
FIGURE B-2: BATCH FILE FOR PARSER 228
FIGURE B-3: EXECUTION OF BATCH FILE FOR XML TO SYSTEMC CONVERSION 229
FIGURE B-4: ADDING SYSTEMC FILES TO A VC++ PROJECT 230
FIGURE B-5: ADDING THE SYSTEMC HEADER FILE DIRECTORY 231
FIGURE B-6: ADDING SYSTEMC LIBRARY MODULES AND DIRECTORIES 232
FIGURE B-7: DIGITAL WAVEFORM OF THE SIMULATED RESULT OF SWITCH AND LIGHT SYSTEM 233
FIGURE B-8: CYGIN SETUP 235
FIGURE B-9: SUCCESSFUL COMPILATION OF THE SC2V SOURCE CODE 236
FIGURE B-10: INCORRECT CODE IN SC2V.SH 236
FIGURE B-11: CORRECT CODES IN SC2V.SH 236
FIGURE B-12: TRANSLATING FROM SYSTEMC TO VERILOG 237
FIGURE C-1: SELECT THE MODULE NAME 240
FIGURE C-2: CHANGE MODULE NAME TO "SYSTEMC" 241
FIGURE C-3: DEFINE THE MODULE-NAME AND INSTANT-NAME 241
FIGURE C-4: DEFINE THE PORT-NAME AND DATA-TYPE 242
FIGURE C-5: DEFINE PORT-TYPE 242

FIGURE C-6: GENERATE THE SYSTEMC CODE 243

LIST OF TABLES

TABLE 3-1: LITERATURE KEYWORD SEARCH RESULTS 16
TABLE 3-2: KEY PAPERS IDENTIFIED 18
TABLE 3-3: COMPARISON OF EVALUATION CRITERIA 29
TABLE 5-1: COMPARISON OF MICROPROCESSORS, FPGA AND ASIC DEVELOPMENT PROCESSES [ADAPTED FROM KLINGMAN, 2006; ASLAM, 2007] 48
TABLE 5-2: COMPARISON OF UML-BASED METHODOLOGIES 65
TABLE 5-3: COMPARISON OF IEC 61499-BASED METHODOLOGIES 67
TABLE 6-1: MAPPING SELECTED METHODOLOGIES AGAINST COMMON STEPS 73
TABLE 8-1: POLISHING TIME ESTIMATOR USER REQUIREMENTS 112
TABLE 8-2: FFT USER REQUIREMENTS TABLE 126
TABLE 8-3: PID CONTROLLER USER REQUIREMENTS 136
TABLE 8-4: EVALUATION OF THE EXPERIMENTAL APPLICATIONS 148
TABLE 8-5: NUMBER OF RESPONSES 149
TABLE 8-6: SUMMARY OF THE FEEDBACK SCORE 155
TABLE 9-1: USER REQUIREMENTS TABLE 173
TABLE 9-2: USER REQUIREMENTS TABLE 175

GLOSSARY OF TERMS

A*STAR	Agency of Science, Technology and Research
AAGR	Average Annual Growth Rate
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
CORFU	Common Object-Oriented Real-Time Methodology for Unified Development
EDB	Economic Development Board
EDIF	Electronic Data Interchange Format
ERC	Economic Review Committee
ESS	Engineering Support System
FBDK	Function Block Development Kit
FFT	Fast Fourier Transforms
FPGA	Field Programmable Gate Arrays
GDP	Gross Domestic Product
GUI	Graphical User Interface
HDL	Hardware Description Language
HTML	Hypertext Mark-up Language
ICT	Information and Communication Technology
IEC	International Electro Technical Commission
JDOM	Java-based Document Object Model
LEX	Lexical Analyser Generator
MNC	Multinational Corporation
MTI	Ministry of Trade and Industry
OOONEIDA	Open Object-Oriented knowledge Economy in Intelligent Industrial Automation
PLC	Programmable Logic Controller
PID	Proportional, Integral and Derivative

GLOSSARY OF TERMS (CONTINUED)

PV	Process Value
RTL	Register Transfer Logic
RTTI	Run-Time Type Information
SAX	Simple API for XML
SIMTech	Singapore Institute of Manufacturing Technology
SV	Set-point Value
UML	Unified Modelling Language
VCD	Value Change Dump
VHDL	Very High Speed Integrated Circuit Hardware Description Language
W3C	World Wide Web Consortium
PATH	XML Path Language
XMI	XML Metadata Interchange
XML	extensible Mark-up Language
XSL	extensible Style Sheet Language
XSL-FO	XSL Formatting Objects

CHAPTER 1 INTRODUCTION

This chapter provides an introduction to the research described in this thesis. The chapter is organised as follows: Section 1.1 presents the background context of the research, followed by an overview of the research aim, objectives and programme in Section 1.2. Section 1.3 briefly summarises the primary thesis contributions. Finally, the organisation of the thesis as well as a short description of each chapter is presented in Section 1.4.

1.1 Research background

The manufacturing sector in Singapore contributes more than one quarter of the country's Gross Domestic Product (GDP) and is a prominent sector driving economic development of Singapore. With the ever-increasing competition from China, India and other South East Asian countries, the Singapore manufacturing sector is facing its toughest challenge in recent history. Consequently, these challenges force the sector to continually reduce cost, improve resource utilisation and develop innovative solutions to maximise Singapore's competitiveness.

With the increasing trend towards industrial automation and intelligent products, embedded systems are becoming a crucial component of advanced technology, complex equipment, products and automation. However, the development of embedded applications is a complex process, as engineers have to deal with multiple platforms, different development environments and programming languages. Currently Singapore is lagging behind many foreign counterparts in the development of embedded applications. Nevertheless, Singapore aims to leverage on its current world-class IT and electronics industries to capture the huge global embedded software market.

This research is sponsored by the Singapore Institute of Manufacturing Technology (SIMTech). SIMTech is one of the fourteen research institutes under the Agency for Science, Technology and Research (A*STAR), the Singapore government agency supporting and promoting research and development for the Singapore industry.

1.2 Overview of the research aims, objectives and programme

The previous section outlines the motivation and rationale driving this research to investigate a system development methodology for embedded applications. The aim of this research is therefore to propose an improved methodology to support embedded applications development. The objectives involve reviewing the existing system development methodologies, formulating the conceptual methodology, developing the proof-of-concept tool, testing and evaluating the methodology and finally refining the proposed methodology.

In order to achieve the research aim and objectives, a structured research programme is developed, outlining the project details, phases, relevant objectives, tasks required and the deliverables.

1.3 Overview of research contributions

The main contributions of this thesis include the formulation of the new international standard-based system development methodology which seeks to guide future research and implementation direction, a proof-of-concept tool for embedded applications that allow researchers with the necessary test beds for experimentation and future improvement.

1.4 Structure of the thesis

The thesis is organised into the following chapters:

Chapter 2 Defines the industrial context of the research in Singapore, identifies the fields of research and outstanding issues that should be addresses.

Chapter 3 Reviews the relevant literature related to embedded software applications development. The current state of research and challenges with existing approaches are discussed, further motivating the need for the newly proposed approach.

Chapter 4 Develops the research aim, research objectives and a structured research programme. The plans and details of individual work packages are also identified.

Chapter 5 Reviews of the existing system development methodologies for embedded applications and analyses the strengths, weaknesses and opportunities of each methodology

Chapter 6 Presents and discusses the formulation of the conceptual system development methodology for embedded applications.

Chapter 7 Develops the proof-of-concept tool, components and code generator in order to demonstrate the merits of the proposed approach and satisfy the various requirements for the system development methodology for embedded applications.

Chapter 8 Evaluates the methodology with a number of experimental applications and analyse the potential improvement to the methodology from the workshop carried out.

Chapter 9 Refines and documents the final system development methodology for embedded applications

Chapter 10 Concludes the thesis and identifies future research opportunities.

CHAPTER 2 INDUSTRIAL CONTEXT

The objective of this chapter is to understand the needs of the manufacturing industry in Singapore, while also identifying the main challenges faced. Chapter 1 presented an overview of the thesis including its research aim, objectives and research programme, with a summary of research contributions also being provided.

This chapter is organised as follows: Section 2.1 emphasises the importance of the manufacturing industry for Singapore's economy, highlighting the key pillar represented by this sector in terms of Singapore's overall economic power. Section 2.2 surveys the important business challenges for the electronics cluster in order to identify important issues that need to be addressed. Section 2.3 gives an overview of the overall market in this cluster. Section 2.4 presents the Singapore government's initiatives to serve, support, promote and inspire growth of the cluster. The need for a development methodology for embedded applications is highlighted in Section 2.5, with Section 2.6 then providing a summary of the chapter.

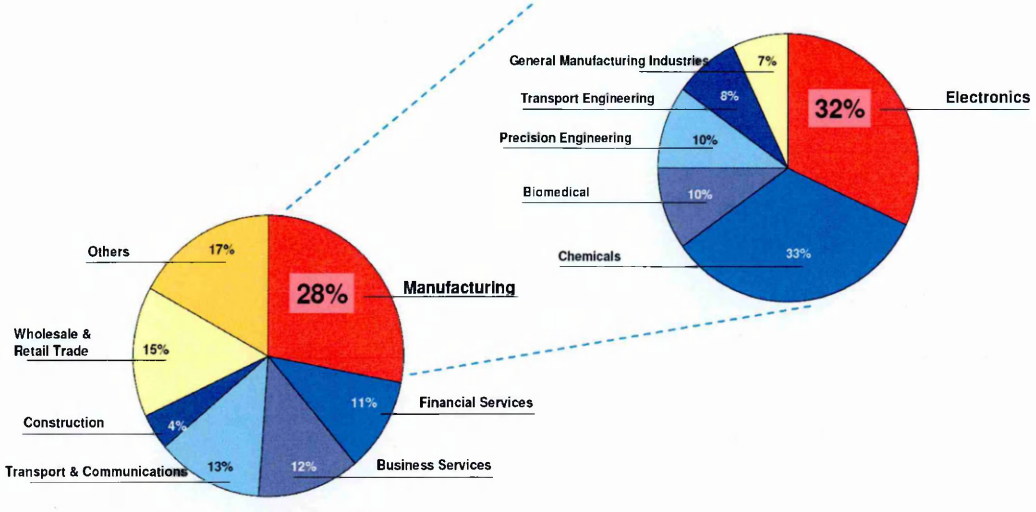


Figure 2-1: Singapore 2006 GDP [EDB, 2008]

2.1 Manufacturing within Singapore's economy

The manufacturing sector, which contributed 28% of Singapore's GDP in 2006, is the most important contributor to Singapore economy. The Singapore's Economic Development Board (EDB) seeks to double the manufacturing output of Singapore, while maintaining its one-quarter share of GDP by 2018 [EDB, 2008]. In manufacturing, Singapore has established global leadership positions in several areas including electronics and electrical products, repairs, maintenance, industrial automation and instrumentation equipment. As shown in Figure 2-1, manufacturing is the largest single contributor to Singapore's GDP and the electronics cluster is one of the key pillars of the Singapore

manufacturing sector. The EDB has estimated that the output from the electronics cluster will grow from S$73 billion in 2006 to S$117 billion in 2018.

The electronics cluster is one of the principal pillars for manufacturing in Singapore. Over the past 15 years, Singapore has transpired as a main regional electronics manufacturing hub, contributing over 4% of worldwide electronics manufacturing and ranking amongst the top 5 countries in the world in terms of electronics manufacturing output [Wong, 2002]. The electronics cluster, including semiconductor production, computer peripherals, consumer electronics, information and communication technology (ICT), contributed about one-third of the overall GDP growth over the period 2000-06. The largest segment within the domestic electronics industry is semiconductor production [MAS, 2007]. There are about 14 semiconductor wafer manufacturing companies in Singapore, 20 assembly and test companies and 40 integrated circuit design companies. In fact, Singapore is home to the world's top 3 in wafer production, the world's top 3 in assembly and test operation and 4 of the world's top 10 fabless design companies [Lee, 2008]. Manufacturing is therefore likely to remain one of the key drivers of Singapore's economy. The primary business challenges facing the electronics cluster are discussed in the next section.

2.2 Challenges facing the electronics cluster in Singapore

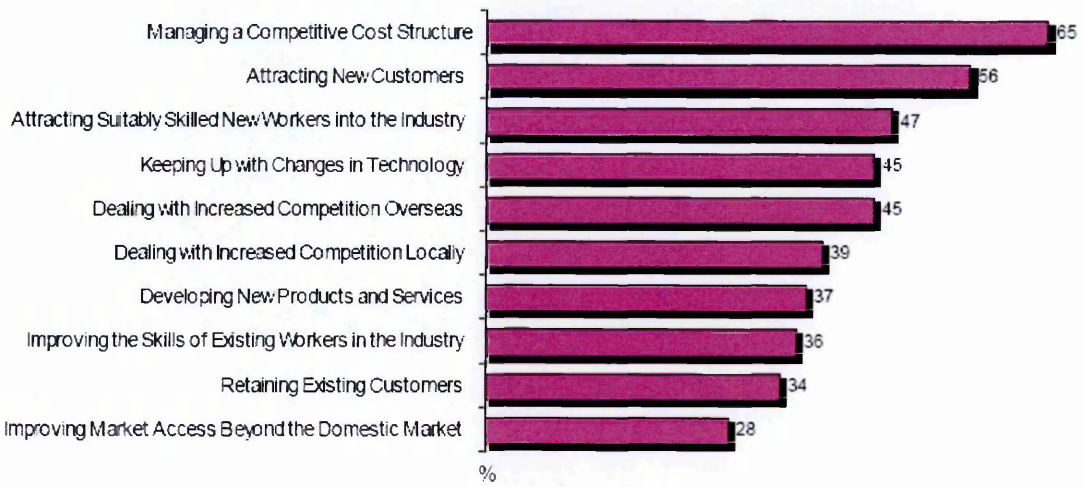


Figure 2-2: Key business challenges, 2005 [WDA, 2006]

A survey conducted by Singapore's Workforce Development Agency (WDA) between 2005 and 2006 provided valuable information regarding the business challenges, needs, current staff skill levels and training cost allocation for various segments within the electronics cluster [WDA, 2006]. As shown in Figure 2-2, the key business challenge of the electronics cluster is managing the competitive cost structure, which includes improving productivity and reducing the cost of components, engineering and development. The next most

critical challenge is to attract new customers by developing novel and innovative products, services, promotions and tapping potential new markets. Attracting skilled employees and keeping up with changes in technology is another important challenge due to the rapid changes of new technologies and the lack of suitably skilled resources. The challenges for engineers are to create solutions at a faster rate, with ever smaller and lower power consuming new products in electronics manufacturing [Lee, 2008].

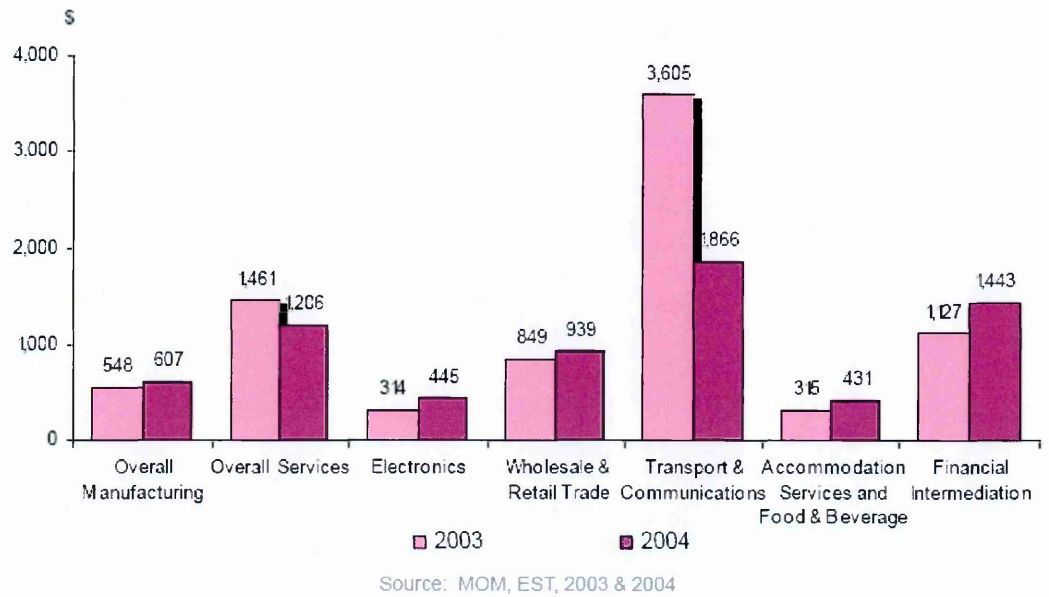


Figure 2-3: Average training cost (in SGD) per trainee by industry, 2003 & 2004 [WDA, 2006]

Inspite of the widely acknowledged importance of the electronics cluster, its average investment in employee training is second lowest as compared to other clusters. From 2003 to 2004, it has had a small increase from S$314 to S$445 per trainee, as shown in Figure 2-3. This lack of investment may affect the competitiveness of the companies in this electronics cluster. Limited emphasis on training may be due to an increasing cost, the rapid pace of innovation and the introduction of new technology. The survey also reveals that the electronics companies need to improve their job-specific skills and productivity. The most crucial job-specific training is related to electronics and embedded system technology.

Unfortunately, the electronics cluster is affected by rapid technological advances which have grown faster than most other industries over the past several years. The rising cost of hiring experienced staff, ever-increasing cost of training, the rapid pace of innovation and the rapid introduction of new technology poses more challenges to the industry. The key issue is how to retain knowledge and reduce the learning curve for various new technologies.

Some of the companies run into trouble because they cannot keep up with the rapid technological developments. Such uncertainties can be expected to continue [BLS, 2007].

2.3 Overview of the electronics cluster in Singapore

The electronics cluster in Singapore is mainly dominated by multinational corporations (MNCs), which include those related to the manufacture of semiconductors, telecommunications, consumer electronics, automations, computer peripherals, industrial and precision equipment, medical technology and automotive electronics. Integrated circuits, televisions, computers, pagers, printers and many other products are now manufactured in Singapore for the global markets [Tilley & Williams, 1997]. As shown in Figure 2-4, most of the companies in the electronics cluster have an intricate integration of microelectronic and embedded systems [Lim, 2007].

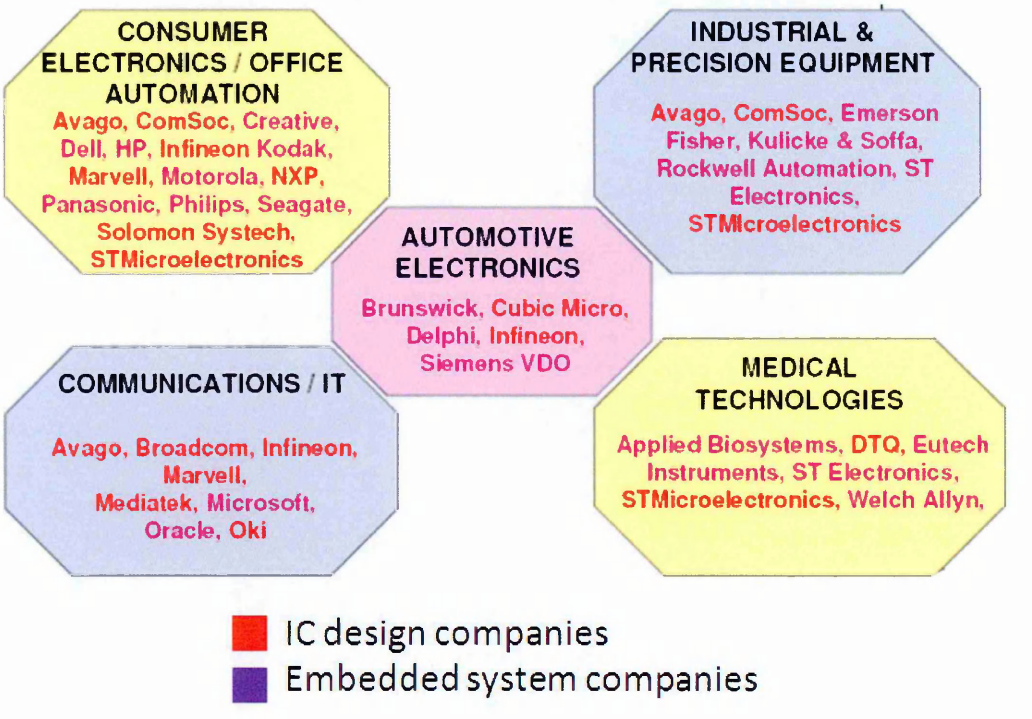


Figure 2-4: The intricate relationship between microelectronics and embedded systems in Singapore [Lim, 2007]

Mr. Heng Chee How, Minister of State for Trade & Industry of Singapore, explained that embedded systems play a very important role in the electronics clusters and the expected significant contribution to the Singapore economy. Electronic companies are important to Singapore's manufacturing output and employment. Electronic product and manufacturing relies heavily on embedded

systems and embedded applications. It is one of the fastest growing clusters in Singapore [Heng, 2005].

The global embedded software market is vast and growing rapidly. The world market size was US$45.9 billion in 2004 and the estimated Average Annual Growth Rate (AAGR) between 2004 and 2009 is 14%, growing to US$88 billion [BCC, 2000]. Singapore is aiming at this large embedded software market with plans by the EDB to position Singapore as a global centre of excellence for embedded software technology. Mr. Ko Kheng Hwa, Managing Director of EDB, proposed that Singapore could capitalise on its strong competency such as its existing world-class information technologies and electronics industries, to develop embedded software expertise. He reiterated that embedded software has the important capabilities of supporting Singapore R&D and manufacturing in various industrial clusters. He further outlined that embedded systems represent the convergence of both hardware and software for manufacturing and services. Developing embedded application software requires a new skill-set in comparison to conventional software and the entry barrier for companies venturing into design and development of innovative embedded software is generally quite high. With improvements of embedded software development environments, Singapore can leverage on the current strengths of its world class IT and electronics industries to provide world-class embedded system designs and development services, as well as industry resources in a convenient and cost-effective manner [Ko, 2005].

2.4 Government initiatives

Huff [1995] argues that Singapore's model of economic development is one that depends on services and manufacturing exports with extensive government intervention and planning. In a small country like Singapore, government initiatives are essential to the successful expansion of the manufacturing sector. Understanding Singapore's government plans and initiatives will help to focus the direction of research and investment. Tilley & Williams [1997] further explained that concerted attempts have been made by the government to entice foreign companies to re-locate their R&D centres to Singapore. Future government-led developments in include the support of local capabilities in embedded systems. The success of the Singapore manufacturing sector is also because of a strong focus on science and technology initiatives. A ten-year master plan consisting of various technology initiatives in embedded software and other focus areas, have immensely benefited the electronics cluster.

Mr. Lee Hsien Loong, the Prime Minister of Singapore, formed the Economic Review Committee (ERC) in Dec 2001 to study economic growth and provide recommendation for future directions. The manufacturing subcommittee of the ERC has invested substantial effort formulating strategies to further uphold Singapore's competitive position in manufacturing and to ensure that the

resources invested will be transformed into sustainable economic growth in the future. The ERC's Manufacturing subcommittee has recommended that Singapore should focus its R&D on "embedded software that makes hardware devices more intelligent with improved functionalities" [Splunter, 2002]. The report further explained that software and embedded systems are one of the key technologies for the manufacturing sector.

In response to the ERC recommendation, the Ministry of Trade and Industry (MTI) published its Science and Technology plan [S&T Plan, 2006], which is the third Singapore research and development (R&D) plan. One area selected by the committee and addressed in this thesis, is embedded and hybrid system research. The Science and Technology plan is an important plan that encourages researchers from various research institutes, universities and companies in Singapore to focus on similar, complementary directions and visions. Singapore is a small country and must focus its limited resources to develop peaks of excellence. R&D funding is then directed towards the selected areas where there is most potential for scientific breakthroughs, which may yield economic benefit for Singapore.

Currently, Singapore's government is actively promoting and developing the embedded software industry. The development thrust focuses on four main areas [EDB, 2005]:

1. *Technology Development:* Singapore will strengthen its embedded software technology by encouraging local R&D activities. The Agency for Science, Technology & Research (A*STAR) has started the "Embedded & Hybrid System Phase I and Phase II" research programme and has sponsored more than 19 research projects in various research institutes, including SIMTech and the Universities. The EDB also has various programmes to promote more R&D in embedded software among companies in Singapore.

2. *Enterprise Development:* Singapore will attract and nurture both local and foreign embedded software development enterprises. Moving forward, Singapore also plans to build on its existing base of more than one hundred companies who develop embedded software application. The Start-up Enterprise Development Scheme (SEEDS) funding programme will support more start-up companies.

3. *Human Capital Development:* Singapore will expand the existing small pool of one thousand embedded system engineers by strengthening its education programmes. The National University of Singapore (NUS) has started an embedded system undergraduate degree. A Masters degree in embedded systems has also been formed in collaboration with the Design Technology Institute (DTI). Nanyang Technological University

(NTU) also started a part-time Masters degree on embedded technology in 2003.

4. *Partnership Development:* Singapore is encouraging cross-industry partnerships to address the complexities of embedded systems and its related technology. The EDB has started the "Embedded in Singapore" platform in 2002 with activities and internal portals to strengthen the community, explore new opportunities and generate new technology partnerships so that embedded system technologies developed in Singapore can capture international attention. More than thirty embedded technology companies have already joined this platform.

2.5 The need for a development methodology for embedded applications

In the electronics industry, companies need to continually enhance their capabilities and delivery lead-times in system integration, automation, equipment, process, quality control and traceability in order to create new value and sustain growth [Yeo, 2008]. The next generation of factory automation will require full-scale, industry-ready paradigms, methodologies and advanced proof-of-concepts [Bouyssounouse & Sifakis, 2005].

For Singapore's electronics industry to maintain competitiveness in the global market, companies must address serious challenges such as reducing embedded application engineering costs, as well as enhancing its functionality and agility to cater for solutions that are more complex. Traditional embedded systems such as programmable logic controllers (PLC) and microprocessors will not be able to handle the design requirements, whereas PC-based solutions are likely to be too bulky for some industrial environments.

There are many different types of embedded systems available in the market, ranging from the single 8-bit microcontroller and microprocessors to multi-core processors. There is also a new family of embedded systems with reconfigurable hardware processors such as Field Programmable Gate Arrays (FPGA) and reconfigurable multi-core processors. The reconfigurable hardware processors use embedded software to reconfigure the gates in the processor. This has provided many opportunities for companies to explore complex solutions either on a single embedded system or on multiple networked embedded systems. Embedded system development has therefore evolved into large-scale, distributed environments, which requires new engineering methodologies and management. Nowadays, embedded system development projects usually involve many engineers, outsourcing partners, hardware partners, third party software components, open-source systems and software. Most embedded system development methodologies are either vendor specific or desktop-based methodologies. Vendor specific methodologies will render the company reliant on the specific vendor's solution, whereas desktop-based

systems are not well suited for embedded system solutions. As a short-term solution to challenges faced by embedded system development projects, companies generally hire more engineers and allocate more resources, which increase the overall cost and complexity of project management. Such action may further delay delivering embedded software applications or deliver applications with bugs, which compromise the quality [Underseth, 2007].

Moreover, embedded systems generally require different hardware platforms, programming languages and expertise. It is therefore difficult for companies to maintain a large pool of engineers with adequate expertise across all disciplines. It is also difficult for existing software or hardware engineers to adapt their skills to new embedded hardware platforms and programming languages. Most companies in Singapore select one or two embedded hardware platforms and languages along with a very dedicated engineer to handle the specific platform, which limits the companies' competitiveness. Other engineers who are not trained in the same platform may not be able to contribute to the detailed design and development. Increasingly, such challenges cannot be effectively met by simply importing embedded software expertise from overseas and increasing the training of engineers.

The growing complexity of embedded application development and the business challenges faced by the electronics cluster reinforce the need for a system development methodology with an automated application generator to reduce the dependency on human coding for the various embedded platforms. The objective is to have engineers and integrators create reusable components, which can be shared and automated throughout the development cycle. Therefore, the challenge of this research can be summarised by the question:

"Is there a development methodology for embedded applications, which can retain knowledge and reduce the learning curve for various new embedded application technologies to address some of the existing business challenges?"

Should such a development methodology exist, it would be capable of addressing some of the industrial problems previously discussed. If the methodology were not available, then the research problem would focus on the development of a methodology based on new and existing knowledge. The next step is therefore to understand the current state of research and the challenges related to embedded application development.

2.6 Chapter summary

This chapter highlighted the importance of the manufacturing sector and the electronic cluster to the economy of Singapore. It presented the main business challenges facing the electronics cluster and highlighted that the limited emphasis on training may be due to increasing cost, rapid pace of innovation

and new technology introduction. The role of embedded systems in the electronics manufacturing cluster, the estimated market size and the long-term initiatives from the Singapore government have clearly illustrated the importance of embedded technology. Companies need increasingly responsive to customer demands and technology complexity, as they have become vital for companies to have a shorter time to market. With limited training resources, rapid technology advancements and difficulties in finding suitably skilled staff, companies urgently need help to address these challenges. Based on this proposition, it is therefore necessary to understand the current state of research and the challenges related to embedded application development.

CHAPTER 3 LITERATURE REVIEW: EMBEDDED APPLICATION DEVELOPMENT

The motivation of this research is driven by the industrial challenges outlined in the previous chapter. This chapter presents an overview of embedded application development and the current state of research. The purpose of this chapter is to define the key terms, findings, outstanding issues and to address the following important questions:

a) What is a system development methodology?

b) What is an embedded system?

c) What is the current research trend for embedded application development?

d) What are the primary research issues associated with embedded application development?

The chapter is structured as follows: Section 3.1 illustrates the research plan using a systematic search strategy. Section 3.2 discusses the review and selection process of relevant papers. Findings from the literature review are presented in Section 3.3 along with the identified outstanding issues. Finally, a summary of the chapter is provided in Section 3.4.

3.1 Search strategy

A review of relevant literature was carried out to establish the key definitions, findings and issues regarding embedded application design and system development methodologies. The scope of the literature review is therefore limited to papers dealing with embedded applications and development methodologies. The search strategy begins by identifying the database source through keyword searches and gradually narrowing down the selection to those covering journals, conference papers and books from the research community. An extensive database covering manufacturing, computing, electronic, engineering along with library catalogue systems was initially targeted. These database sources allow access to large collections of research publications. The search narrowed down to a few key databases including Association of Computing Machinery (ACM) Digital Library, IEEE Xplore, Science Direct (Elsevier), ISI Web of Knowledge (ISI) and Inspec on Engineering Village (EI).

A range of keyword combinations including "embedded system", "embedded application", "embedded software", design, development, platform or methodology were used. As shown in Table 3-1, it is impractical to review to extensive list of relevant literature. In order to restrict the search to more recent publications, the timeframe was set from 2000 to 2008. However, the search of literature remained broad due to the citations in the surveyed recent literature. Some citations in key publications were also selected as part of the review.

Some of the publications are duplications from different databases. For example, ISI and EI reference many papers from other databases such as ACM, IEEE Xplore and Elsevier. Therefore, the list of the papers needed to be further checked for duplication, relevance of the title, keywords listed and their abstracts before selecting the final list of articles for review.

3.2 Results and analysis

The search strategy began by identifying relevant databases through different search strings as shown in Table 3-1.

Table 3-1: Literature keyword search results

Keywords searched	ACM	IEEE	EI	ISI	Elsevier	Total
S1:"Embedded system" + Development + Methodology	614	50	258	59	15	996
S2: "Embedded system" + Development + Platform	696	86	381	79	10	1252
S3: “Embedded system” + Development + Methodology	649	197	269	2	2	1119
S4: "Embedded software" + Development + Methodology	649	24	12	30	649	1364
S5: "Embedded software" + Development + Platform	681	52	250	49	3	1035
S6: "Embedded software" + Development + Methodology	618	52	250	49	3	972
S7: "Embedded Application" + Development + Methodology	98	4	30	5	2	139
S8: "Embedded Application" + Development + Platform	111	1	36	8	4	160
S9: "Embedded Application" + Development + Methodology	79	3	25	3	1	111

Table 3-1: Literature keyword search results (continued)

Keywords searched	ACM	IEEE	EI	ISI	Elsevier	Total
S10: "Embedded system" + Design + Methodology	9743	411	0	6	1997	12157
S11: "Embedded system" + Design + Platform	7431	547	0	3	55	8036
S12: "Embedded system" + Design + Methodology	9498	716	0	2	97	10313
S13: "Embedded software" + Design + Methodology	7,949	40	0	0	36	8025
S14: "Embedded software" + Design + Platform	6579	90	0	2	30	6701
S15: "Embedded software" + Design + Methodology	7295	113	0	0	32	7440
S16: "Embedded Application" + Design + Methodology	9273	30	0	0	48	9351
S17: "Embedded Application" + Design + Platform	7187	34	0	0	35	7256
S18: "Embedded Application" + Design + Methodology	8914	51	0	0	51	9016

The initial selection, after a filtering process, comprised approximately 150 papers from various international journals and conferences. Each of the papers was carefully reviewed to identify the key findings and outstanding issues. Once the list of relevant papers is identified, the first step of the review is to summarise the key findings related to the pertinent questions outlined in the introduction. This allows for a clear understanding of the current research direction. Each paper was analysed in detail to highlight the key findings and identify the outstanding issues. Finally, the findings and issues are summarised to form the research aim, objectives and programmes. Table 3-2 shows a list of 26 key papers selected, with publication dates ranging from 1997 to 2008.

Once the list of relevant papers is identified, the first step of the review is to summarise the key findings related to the pertinent questions outlined in the introduction. This allows for a clear understanding of the current research direction. Each paper was analysed in detail to highlight the key findings and

identify the remaining issues. Finally, the findings and issues are summarised to form the research aim, objectives and programmes.

Table 3-2: Key papers identified

Author	Title	Source
Paulin et al, 1997	Embedded Software in Real-Time Signal Processing Systems: Application and Architecture Trends	Proceedings of the IEEE
Sangiovanni-Vincentelli & Martin, 2001	Platform-Based Design and Software Design Methodology for Embedded Systems	IEEE Design & Test of Computers
Martin, 2002	UML for Embedded Systems Specification and Design: Motivation and Overview	Proceedings of the Design, Automation and Test in Europe Conference
Paulin & Santana, 2002	FlexWare: A Retargetable Embedded-Software Development Environment	IEEE Design & Test of Computers
Yen et al, 2002	A Component-based Approach for Embedded Software Development	IEEE Symposium on Object-Oriented Real-Time Distributed Computing
Alur et al, 2003	Generating Embedded Software from Hierarchical Hybrid Models	Proceedings of the ACM
Damaševičius et al, 2003	Application of Design Patterns for Hardware Design	Proceedings of the 40th Conference on Design Automation
Graaf et al, 2003	Embedded Software Engineering: The State of the Practice	IEEE Computer Society IEEE Software
Karsai et al, 2003	Model-Integrated Development of Embedded Software	Proceedings of The IEEE
Brisolara et al, 2004	Evaluating High-level Models for Real Time Embedded System Design	Design Methods and Applications for Distributed Embedded Systems Conference 2004

Table 3-2: Key papers identified (continued)

Author	Title	Source
Brunel et al, 2004	SoftContract: an Assertion-Based Software Development Process that Enables Design-by-Contract	Proceedings of the Design, Automation and Test in Europe Conference
Mustapic et al, 2004	Real World Influences on Software Architecture - Interviews with Industrial System Experts	Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture
Posadas et al, 2004	Single Source Design Environment for Embedded Systems Based on SystemC	Design Automation for Embedded Systems
Baleani et al, 2005	Correct-by-Construction Transformations across Design Environments for Model-Based Embedded Software Development	Proceedings of the Design, Automation and Test in Europe Conference
Graaf et al, 2005	Evaluating an Embedded Software Reference Architecture — Industrial Experience Report	Proceedings of the 9th European Conference on Software Maintenance and Reengineering
Luan et al, 2005	A Heterogeneous Evolutional Architecture for Embedded Software	Proceedings of the 5th International Conference on Computer and Information Technology
Mattos et al, 2005	Making Object Oriented Efficient for Embedded System Applications	Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design
Hänninen et al, 2006	Present and Future Requirements in Developing Industrial Embedded Real-Time Systems - Interviews with Designers in the Vehicle Domain	Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems

Table 3-2: Key papers identified (continued)

Author	Title	Source
Ibrahim et al, 2006	Embedded Systems Development: Quest for Productivity and Reliability	Proceedings of the 5th International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems
Jeon et al, 2006	Using Operational Architecture to Model Embedded Software	Proceedings of the 13th Asia Pacific Software Engineering Conference
Ma et al, 2006	QoS Analysis for Component-based Embedded Software: Model and Methodology	The Journal of Systems and Software
Oliveira et al, 2006	Early Embedded Software Design Space Exploration Using UML-based Estimation	Proceedings of the Seventeenth International Workshop on Rapid System Prototyping
Zhou et al, 2006	A Software Enhancement System for Embedded Software Development	Proceedings of the 9th International Symposium on Object and Component-Oriented Real-Time Distributed Computing
Balasubramanian et al, 2007	A Platform-independent Component Modelling Language for Distributed Real-time and Embedded System	Journal of Computer and System Science
M. Wenger et al, 2008	Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector State-of-the-Art and Technology Review Report	The MEDEIA Consortium, 2008

3.3 Key findings and outstanding issues

The literature review provides a better understanding of current research trends, establishes the key findings and highlights outstanding research challenges. This section discusses each of these in detail.

3.4 Definition of a system development methodology

Since the research topic of this thesis requires the proposal of a new system development methodology, it is necessary to discuss the meaning of this term. This section presents the definition of a system development methodology that is used throughout the thesis.

There have been many definitions of a system. Hall [1962] explains, "System Engineering is probably not amenable to a clear, sharp, one sentence definition". There are several definitions found in the literature. Hall defines a system as "a set of objects with relationships between the objects and between their attributes". Rechtin [1991] offers a deeper definition of a system as "a collection of things working together to produce something greater...each system is inherently a part of a still larger system". Rechtin & Maier [1997] suggest "a system is a collection of different things which together produce results unachievable by themselves alone". Ellis & Ludwig [1962], Chase [1974] and Minds [1995] define a system as essentially "anything that performs work on an input in order to generate an output".

Blanchard & Fabrycky [1981] defines system development as "a sequential and iterative methodology to reach cost-effective solutions". They continue by stating that system development "is directly concerned with the transition from requirements identification to a fully defined system configuration ready for production and ultimate customer use". On the other hand, Chase [1974] defines system development as "the irreducible gross functional steps which must be followed". Halblander [2004] defines a methodology as "a set of guidelines for use in a larger (less specific) context than would be considered for a method or process. A methodology provides flexibility in the form of general characteristics and guidelines which allow the engineer to choose whatever method or technique is best suited for a particular situation".

Finding 1: For the purpose of this research, the term "System development methodology" is defined as "A set of sequential guidelines for the development of an embedded application that performs work on input(s) in order to generate output(s)".

3.5 Definition of an embedded system

The term "embedded system" is commonly used by both researchers and industrial practitioners, however a consensus regarding its definition is generally lacking as it is a complete list of their characteristics and properties [Graaf et al

2002; Bertolott, 2006]. It is therefore difficult to have a clear definition of embedded system due to the rapid and frequent changes in hardware and software [Noergaard, 2005; Bertolott, 2006].

Generally, most authors agree that an embedded system is:

- a mixed hardware and software system [Barr, 1990; Kumar et al, 1996; Lopez et al, 1998; Calyez, 1993; Kuvaja et al, 1999; Graaf et al, 2002; Bjärnemyr & Dahlberg, 2004; Noergaard, 2005];
- dedicated to a specific application [Barr, 1990; Kumar et al, 1996; Lopez et al, 1998; Calyez, 1993; Paulin et al, 1997; Kuvaja et al, 1999; Wolf, 2000; Graaf et al, 2002; Bjärnemyr & Dahlberg, 2004; Noergaard, 2005; Bertolott, 2006];
- to control, monitor or assist the operation of equipment, machinery or plants, 'embedded' signifying that they are an integrated part of that equipment [Kornecki, 1999; Kippenberger, 1999; Bjärnemyr & Dahlberg,, 2004; Bertolott, 2006]; and
- not seen or programmable by the user as a computer. It is an execute-only target environment [Kornecki, 1999; Bertolott, 2006].

The following are some of the additional characteristics or properties of embedded systems highlighted by various authors:

- It has real-time behaviour [Paulin et al, 1997].
- Correct design is essential due to the potential impact on the surrounding environment or person using the equipment [Paulin et al, 1997].
- It can perform very limited functions and specific control capability [Pressman, 1997].
- It includes one or more microprocessors [Bjärnemyr & Dahlberg, 2004].
- The principal role is not only the conversion of data but also to interact with the physical world [Luan et al, 2005].
- It is a computer system with higher quality and reliability requirements than other types of computer systems [Noergaard, 2005].

Finding 2: There is no general consensus on the definition of embedded system due to the constant change and advancement of the technology. Some of the generally agreed characteristics or properties of embedded systems are:

- a mixed hardware and software system
- a special purpose and dedicated functional system

- a device that controls, monitors or assists the operation of equipment, machinery or plants
- the operation is hidden from the user's point of view.

3.6 Complexity and rapid growth of embedded application

Paulin et al [1997] predicted that embedded systems would continue to grow in terms of both market share and complexity. He argued that the rapidly increasing effort associated with embedded application development goes beyond hardware-oriented development. Embedded systems are constantly increasing in complexity required to fuel the recent explosion in demand. Due to the growing sophistication and complexity of modern applications, embedded applications are becoming more complex and difficult to develop [Yen et al, 2002; Graaf et al, 2003; Jeon et al, 2006; Ibrahim et al, 2006, Lau & Pritchard, 2006]. This requires embedded application development engineers to constantly learn different tools, making it more difficult to deliver high-quality applications on time.

As the complexity, number and diversity of applications increase, more and more companies are encountering challenges in achieving innovative embedded application within a short timeframe [Graaf et al, 2003]. There is no single dominant operating system, which results in an added problem concerning the selection of a suitable embedded system. Unlike PC-based systems, an operating system for embedded system needs to be adapted for specific architecture and hardware specification. Embedded application developers require precise skill sets and domain knowledge regarding the specific hardware and software system under development. Many companies may not have such a broadly skilled workforce readily available [Paulin & Santana, 2002].

Finding 3: It is evident that embedded systems will continue to grow in demand, market share and complexity. Embedded applications are becoming more complex and difficult to develop because of the growing sophistication of application, hardware and software environments. There is no single dominant operating system, software architecture and hardware specification for embedded systems. There is also an increasing tendency of shifting functionality and complexity away from hardware into software. Embedded application developers require familiarity with specific programming languages, as well as a broad range of skills and domain knowledge which companies may not readily have access to.

3.7 Merging of hardware and software

There is an increasing tendency of shifting functionality and complexity away from hardware into software [Paulin et al, 1997; Mustapic et al, 2004; Hänninen

et al, 2006; Ibrahim et al, 2006; Lau & Pritchard, 2006]. A survey of some electronics manufacturing companies in Europe suggested that embedded systems are moving from "hardware first" to "software first" approaches. Increasingly, more intelligence is encapsulated in embedded system software [Mustapic et al, 2004]. In another separate survey of requirements for developing embedded systems, Hänninen et al [2006] found that more hardware solutions are being replaced by software solutions, resulting in added complexity of embedded systems.

The trend for shifting functionality and complexity into software is mainly due to economic reasons. Manufacturers need to reduce the cost of products but simultaneously seek configurable components. Shifting functionality to software can reduce the hardware. In addition, it is easier to modify software rather than hardware. Furthermore, using software for the configuration of hardware is more cost effective. However, these added functionalities increase complexity and when coupled with a lack of system design tools, greatly complicates the process of embedded system development [Ibrahim et al, 2006].

The increasing range of features, complexity and high performance requirements of embedded systems require configurable hardware to enhance computations that were originally performed by microprocessors [Lau & Pritchard, 2006]. The job of embedded application developers is further complicated by shifting functionality and complexity away from hardware into software [Lau & Pritchard, 2006; Ibrahim et al, 2006].

Finding 4: Increasingly, more functions that are typically executed hardware are being moved to software because manufacturers seek to reduce product cost while maintaining configurability. By encapsulating intelligence and hardware functions into software, the product cost can be reduced and configurability improved. On the other hand, the complexity of the developing the embedded application will increase.

3.8 Software components, re-usability, design pattern and code generation

Techniques for reusable application development, component-based methods, design pattern methods and code generation are favoured by most companies because they can significantly reduce the development time and cost [Yen et al, 2002; Paulin & Santana, 2002; Damaševičius et al, 2003; Graaf et al, 2003; Mustapic et al, 2004; Baleani et al, 2005; Ibrahim et al, 2006; Hänninen et al, 2006; Ma et al, 2006; Balasubramanian et al, 2007]. An industrial survey by Graaf et al [2002] and Mustapic et al [2004] found that reusability is often considered one of the most important advantages of embedded systems. The reuse of subsystems or components in a new system design was considered as an important cost saving practice. As the cost of developing embedded

application software increases, it becomes increasingly important to reuse software components to minimise investment requirements.

However, most companies practice reusability in an ad-hoc manner. Project team members simply copy project documents, requirements and coding from similar projects previously developed. Most of the components need to comply with legacy systems, as most projects do not start from scratch. This presents a challenge for any new introduction of technology, in that it should be at least compatible with the legacy technology. Improving reusability could also be done at the system level using design pattern oriented modelling and components [Damaševičius et al, 2003; Rinc'on et al, 2005].

Design pattern is the encapsulation of common solutions to recurring design problems at the system level. Design pattern can increase reusability, quality and productivity in embedded application design. The benefits of using design patterns include abstract and implementation independence, ease of communication between difference design teams and the reuse of existing testing methodologies to ensure better application quality [Damaševičius et al, 2003].

Software components are designed as reusable or standard-based components. Standard-based component models are increasingly replacing the use of commercial-off-the-shelf (COTS) proprietary components for embedded application development, especially for large-scale embedded applications. Companies are looking for standard-based component development environments believing that they can facilitate development which improves the embedded application software quality since components are often used in multiple applications and are subject to extensive testing. A component-based development approach can significantly reduce software development time and cost [Ma et al, 2006; Hänninen et al, 2006; Balasubramanian et al, 2007].

To enhance the productivity of embedded applications, software development has shifted away from programming to integration components and code generation. A useful methodology must involve automatic or semi-automatic code generation with support by software tools. Code generation from standard-based component tools can improve the embedded application automation and productivity as well as accelerate reusability, sharing and transferability [Damaševičius et al, 2003; Baleani et al, 2005; Ma et al, 2006].

The benefits of reusable standard-based components, design pattern and code generation approaches in embedded application development include:

- *Timely functional products:* The reuse of components, design patterns and code generation can help to cope with the limited availability resources (time, human or financial) constraints and result in improved productivity [Ibrahim et al, 2006; Damaševičius et al, 2003; Balasubramanian et al, 2007].

- *Reliability:* The reuse of well-tested and designed components and design patterns can dramatically improve the reliability of the final system [Ibrahim et al, 2006; Hänninen et al, 2006].
- *Improve understanding through abstraction:* The reuse through components and design patterns helps developers abstract detailed implementations of components from the system. This can help in system level understanding, communication, simplify development and further improve productivity [Ibrahim et al, 2006; Damaševičius et al, 2003].
- *Incremental development:* Components can be formed from others of minimum functionality which can later be enhanced and made more complex. This can help to improve the time to market [Ibrahim et al, 2006].
- *Maintainability:* An individual component can be enhanced and easily customised to meet new requirements. It is more manageable to alter the functionality of individual components rather than changing the full system [Paulin & Santana, 2002; Ibrahim et al, 2006].
- *Minimise risk:* Components and code generators help to make the system easier to modify and replace. They also reduce the time, cost and risk of developing the embedded application [Damaševičius et al, 2003; Ibrahim et al, 2006].

Finding 5: One way to develop embedded applications is by using design patterns, standard-based component development and code generation techniques. This can reduce time and cost while improving software reusability and quality. New embedded applications can be easily assembled from reusable components.

3.9 Research focus on standards like SystemC, UML, Function Block

The literature has identified numerous research activities in the area of embedded systems, with the main research direction being focusing on standard-based approaches such as Unified Modelling Language (UML), Function Block and SystemC. Other research efforts are quite fragmented across areas ranging from new compiler, CASE tools, expert system and Petri-net. Standard-based approaches are detailed next.

UML Standard: UML is a well-accepted modelling standard for desktop PC-based software engineering. Many researchers focus on high level abstraction subsets such as Model Driven Architecture (MDA) development [Ibrahim et al, 2006; Oliveira et al, 2006; Balasubramanian et al, 2007]. For example, Platform-based design (PBD) uses MDA development methods with UML to

promote the reuse of pre-designed components with rich libraries of components that allows developers to assemble a system [Oliveira et al, 2006]. Another example, using UML-based model-driven development (MDD) model to develop and apply domain-specify modelling languages (DSMLS) to Distributed Real-time and Embedded (DRE) systems [Balasubramanian et al, 2007]. Some researchers use model-based design methodologies to solve verification-related problems [Baleani et al, 2005]. Many researchers [Tan et al, 2004; Coyle & Thornton, 2005; Wang et al, 2006; Nguyen et al, 2007] extended the UML standard by using eXtensible Markup Language (XML) as an intermediate language to convert a UML design to either SystemC or hardware description language (HDL) for a single processor environment. In addition, Kangas et al, [2006] further explored UML as a modelling method for multi-processor and FPGA environments. The high adoption of XML shows that the research is moving towards open standards. UML is commonly used for the modelling and design of desktop PC-based software systems as it is a collection of diagrams that are used for software design specifications. UML is therefore generally more suitable for software engineers. However, basic UML lacks representation for gates, I/Os, timing as well as distribution and interconnection through multiple controllers. Consequently, new diagrams and representations are needed to enhance the UML standards for Field Programmable Gate Array (FPGA)-based embedded systems.

Function Block Standard: The Function Block modelling approach has been used by automation, signal processing, control engineering communities and is widely accepted in industrial design [Brisolara et al, 2004]. Open Object-Oriented kNowledge Economy in Intelligent inDustrial Automation (OOONEIDA or O3NEIDA) introduced various researches to reconfigurable industrial control and automation objects [Vyatkin et al, 2005a]. It strongly supports the International Electrotechnical Commission (IEC) 61499 Function Block standard. Woll [2007] claimed that IEC 61499 is 'simple, cost effective and repeatable', to meet the industry demand of mass customisations and 24/7 operation. Thamboulidis [2005] developed a Common Object-Oriented Real-Time Methodology for Unified Development (CORFU) Engineering Support System (ESS) to translate UML to IEC 61499, but it does not deal with FPGA-based embedded systems. There are many research focusing on microprocessor-based development environment, the semantics, modelling and verification using IEC 61499 [Khalgui et al, 2004; Thamboulidis, 2006; Dubinin & Vyatkin, 2006a; Dubinin et al, 2006b]. In addition, Zoitl et al, [2007] summarised that most of the current research is on design and execution environments, the definition of semantics, application scenarios and coding tools. Most IEC 61499 tools support programmable logic controller (PLC), with limited microprocessor based solutions, but do not address other types of embedded controllers that are using FPGA or a hybrid of FPGA and microprocessor.

SystemC Standard: The IEEE 1666-2005 SystemC standard is gaining increasing popularity as a specification language for embedded systems. SystemC is a subset of the C++ language which can support the complete specification of a system and model the platform. SystemC can be a single source code for system-level specification and profiling, hardware software co-simulation and embedded software generation [Posadas et al, 2004; Tan et al, 2004; Coyle & Thornton, 2005; Wang et al, 2006; Nguyen et al, 2007].

Other research: StateCharts and Petri Nets have also been explored as code generators by a few researchers [Mura et al, 2007; Gomes et al, 2004; Nunes et al, 2007; Hagge & Wagner, 2005]. Gomes et al [2004] and Nunes et al [2007] use Petri Nets as modelling diagrams to capture the design and to convert XML-based intermediate languages to C and potentially Very High speed integrated circuit hardware Description Languages (VHDL). Their research interest is to provide a system level design using open standard, XML-like tools for embedded systems. Other research efforts include creating a new compiler that can use standard American National Standards Institute (ANSI) C Language to generate FPGA [Lau & Pritchard, 2006] and new Computer-Aided Software Engineering (CASE) tools. These are aimed at assisting embedded application development [Luan et al, 2005] and creating expert systems to support semi-automated source code optimisation and reconfiguration [Zhou et al, 2006].

The aforementioned research has explored XML as an intermediate language for code generation. However, this functionality is still lacking for IEC 61499 based development environments for FPGA-based embedded systems. Future research should therefore be focussed on IEC 61499-based embedded system development environments for both FPGAs and microprocessors. This would enable the environment to be more accessible to industrial engineers and hence increase the adoption of embedded systems in manufacturing industries.

Finding 6: Current embedded application development research mainly focuses on standard-based related work such as UML, IEC 61499 Function Block and IEEE 1666 SystemC. Other research efforts appear fragmented.

3.10 Evaluation criteria

In order to evaluate various system development methodologies, evaluation criteria is needed to provide a reference for comparison. In the literature, there are a number of evaluation criteria that need to be reviewed. The evaluation criteria highlighted by various papers are listed below and summarised in Table 3-3..

Table 3-3: Comparison of evaluation criteria

	Authors			
Evaluation Criteria	Brisolara et al [2004]	Mustapic et al [2004]	Graaf et al [2005]	Ibrahim et al [2006]
Testability	✓		✓	✓
Portability	✓	✓	✓	✓
Component modelling		✓		✓
Interoperability		✓	✓	✓
Understandability	✓	✓		✓
Reusability		✓	✓	✓

- *Testability:* The methodology should offer simulation before implementation. It should evaluate whether the specification can translated into an implementation code that is compatible with the rest of the system, for example, code generation [Brisolara et al, 2004; Graaf et al, 2005; Ibrahim et al, 2006].
- *Portability:* The methodology should be independent from the system implementation [Brisolara et al, 2004; Mustapic et al, 2004; Graaf et al, 2005; Ibrahim et al, 2006].
- *Component Modelling:* The methodology should support component modelling [Mustapic et al, 2004; Ibrahim et al, 2006].
- *Interoperability:* The methodology should be able to integrate with different components or legacy applications [Mustapic et al, 2004; Graaf et al, 2005; Ibrahim et al, 2006].

- *Understandability:* The methodology should be easy to understand by various stakeholders [Mustapic et al, 2004; Ibrahim et al, 2006].
- *Reusability:* The methodology should support the reuse of components or source codes [Mustapic et al, 2004; Graaf et al, 2005; Ibrahim et al, 2006].

Finding 7: The evaluation criteria for selecting a system development methodology for embedded applications should include testability, portability, component modelling, interoperability, understandability and reusability.

3.11 The need for an embedded application system development methodology

The literature review identifies that industry practitioners are not satisfied with existing system development methodologies for embedded applications [Mattos et al, 2005; Ibrahim et al, 2006; Hänninen et al, 2006]. The embedded hardware capabilities have improved leading to customers demanding more elaborate products with increasing software complexity. There is an increasing demand for faster and simpler development methodologies, which has fuelled research interest. The current software engineering and system development methodologies for desktop environments are no longer adequate for embedded applications. The methodologies, techniques and tools for embedded application development are different from desktop PC-based application development and are in fact lagging behind their desktop counterparts by many years. To improve productivity, quality and shorten delivery time of the embedded application, companies must apply appropriate software engineering technologies [Paulin et al, 1997; Sangiovanni-Vincentelli & Martin, 2001; Paulin & Santana, 2002; Graaf et al, 2003; Luan et al, 2005; Mattos et al, 2005; Ibrahim et al, 2006; Hänninen et al, 2006].

The organisational factors and architecture of embedded systems are relatively multi-disciplinary. The development of embedded system applications requires highly skilled embedded software engineers. Furthermore, the design of embedded systems requires multidisciplinary engineers as subsystems are implemented by different disciplines. The development process, it also involves external and internal stakeholders from other disciplines who are not software practitioners. Stakeholders must be able to participate and contribute to embedded application development without necessarily having an in-depth knowledge of the specific embedded application programming languages. A good design and system development methodology will aid communication between different development teams, both at the horizontal and vertical levels. Vertical levels of communication occur between stakeholders who are responsible for different levels of the systems, subsystems and components. On the other hand, horizontal communication occurs between stakeholders who are responsible for the development at the same level. Many of the embedded

system engineers mostly come from either software, mechanical or electronic backgrounds. The tools must to be easy enough for mechanical and electronic engineers to use, yet also suitable for software engineers. In some companies, the embedded application development engineers are not involved in the system level design because they do not have the system view expertise for a given application. On the other hand, mechanical and electronic engineers might not understand the embedded application software development. Even embedded application developers cannot communicate with another team of embedded application developers if both are using different tools and platforms.

Hardware properties also narrow application development flexibility and testing. These problems become so serious that some companies are forced to employ engineers who know multiple disciplines. Another approach is to have a development environment that multidisciplinary engineers could understand and to which they could participate at different levels of design and development. Each level will decompose the multidisciplinary systems into single-disciplinary subsystems or components. The system development methodology needs to consider stakeholders and should integrate more accepted and easy to understand technologies to shield the complexity. Communication, observation and discovery among different stakeholders will need to be created to develop a better embedded software application [Yen et al, 2002; Paulin & Santana, 2002; Damaševičius et al, 2003; Graaf et al, 2003; Mustapic et al, 2004].

A new system development methodology for embedded applications should be like the PC-based application development methodology which uses international standards. This will free up the embedded application developers from the selection of processor architectures since embedded application development standards can provide them with more hardware architecture freedom. In this way, embedded application developers are not restricted by hardware or legacy development tools and only need to define the functions of the application using the standards. With a new methodology, embedded applications can be developed faster to meet the market needs [Paulin et al, 1997; Sangiovanni-Vincentelli & Martin, 2001].

The increase in the range of features and complexity of embedded systems requires more time for development and integration. Increasing demand in performance requires configurable embedded processor like FPGAs to accelerate computations that were originally performed by microprocessors. FPGAs are becoming increasingly used as an appropriate solution to meet the challenges faced by embedded system designers. A system development methodology should cover both the microprocessor-based and embedded FPGA-based hardware system and embedded software development environments. There is a clear need for new methodology, which support embedded applications on both FPGAs and microprocessors. Research trends

are evolving towards a standard-based methodology exploiting component reusability [Posadas et al, 2004, Lau & Pritchard, 2006].

Industries require system development methodologies for embedded applications to help in integration and possibly, to provide domain specific components that support code generators. Component-based development environments can facilitate faster development and improve software quality because components are often used in multiple applications and are thus subjected to extensive testing. The benefits of using components include abstract and implementation-independence, ease of communication between different design teams and the use of existing testing methodologies to ensure better application quality. A better representation of components can help application developers to assemble components into a system. With code generation tools, productivity embedded application automation can be improved as well as accelerating reusability, sharing and transferability. Component-based solutions can retain the knowledge of engineers which is important given the general high turnover in the field [Yen et al, 2002; Damaševičius et al, 2003; Mustapic et al, 2004; Ma et al, 2006; Hänninen et al, 2006].

Issue: A new system development methodology for embedded applications is required. The methodology must be easy to understand by various stakeholders, as well as support existing and new processor platforms.

3.12 Chapter summary

This chapter has captured the characteristics of embedded systems, highlighting the various findings and outstanding research issue. Numerous findings have been uncovered. The current needs of system development for embedded applications are still not resolved. In the next chapter, a research programme is developed based on the industrial problems described previously in Chapter 2 and the research issues identified in Chapter 3.

CHAPTER 4 RESEARCH PROGRAMME

The previous chapter presented various research findings related to embedded application development and identified the outstanding issues, emphasising that a new system development methodology for embedded applications is required. This chapter discusses the overall research aim and objectives. It also discusses the research process adopted. The chapter is organised as follows: Section 4.1 gives an overview of industrial problems and the context of the research, with Section 4.2 presenting the research aim and objectives. In order to achieve the research aim and objectives, a detailed research process is discussed in Section 4.3.

4.1 Industrial problems and the research context

As highlighted in Chapter 2, the manufacturing sector is the most important contributor to Singapore's economy, with the electronics cluster being one of the most important manufacturing pillars. The key business challenges from the electronics cluster are to reduce engineering and development cost, keep up with the rapid changes and introduction of new technologies and the lack of a suitably skilled workforce. Embedded systems play a crucial role in the electronics industry and contribute extensively to Singapore's economy. The ERC's manufacturing subcommittee has recommended that Singapore should focus its R&D on "embedded software that makes equipment and products intelligent with improved functionalities". Various government initiatives including R&D programmes, enterprise, human capital and partnership development aim to position Singapore as a global centre of excellence for embedded software technology. The growing complexity of embedded application development and the numerous business challenges faced by electronics cluster fuel the need for an efficient development methodology for embedded systems.

A literature review of embedded application development related research presented in Chapter 3 discussed various findings and research issues. As the result of constant changes and advancement of technology, there is no general consensus on the definition of an embedded system. As such, the general agreed characteristics or properties of an embedded system were established. The complexity and rapid growth of embedded applications have forced developers to focus on specific skills and domain knowledge for limited embedded system solutions. With the transition of hardware functionality to software, the complexity of embedded applications development further increases. Most researchers and practitioners find that standard-based components and code generation can help to reduce time and cost, while improving software reusability and quality. Most of the research focuses on standards such as SystemC, UML and Function Block but most practitioners find that UML is difficult to use, communicate and understand. Most practitioners employed free-form box-line diagrams to communicate. In addition, the common evaluation criteria of selecting system development

methodology for embedded applications were established. Finally, needs were outlined for a new system development methodology for embedded applications which supports both microprocessor and FPGA platforms with components and code generation features.

4.2 Research aim and objectives

The industrial problem and research findings have confirmed the need for a new system development methodology for embedded applications. Therefore, the aim of this research is to

"Develop a standard-based system development methodology for embedded applications"

In order to realise the research aim for the research project, the following research objectives have been set to:

1. Review of the existing system development methodologies,
2. Formulate the conceptual methodology,
3. Develop the proof-of-concept tool,
4. Evaluate the methodology,
5. Refine final methodology.

The next section proposes the research programme and methods to execute the research.

4.3 Research process

This research has been made possible through sponsorship from Singapore Institute of Manufacturing Technology (SIMTech) under the Agency for Science, Technology & Research (A*STAR) in Singapore. SIMTech specialises in manufacturing related research for Singapore-based industries.

System development requires formulating a conceptual methodology, building a proof-of-concept system and tools [Morton, 1984; Nunamaker & Chen, 1989; Gregg et al, 2001] whereas conducting laboratory experiments [Basili et al, 1986; Nunamaker & Chen, 1989; Gregg et al, 2001], can be used as a research domain. The research process for a system development methodology which will be used in this thesis is shown in Figure 4-1.

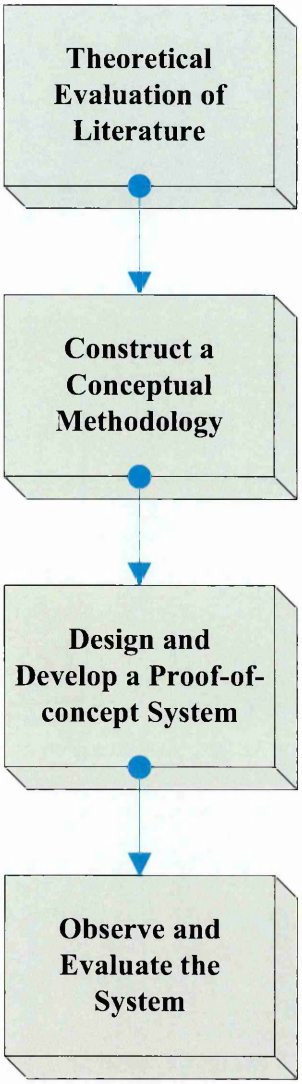


Figure 4-1: Research process for systems development [Adapted from Nunamaker et al, 1991]

In order to achieve the research aim and objectives, a structured research process similar to Nunamaker et al [1991] has been developed. The research process is briefly outlined in five stages.

Stage 1: Review of the existing system development methodologies. The review and analysis of current system development methodology for embedded applications from the literature is an essential foundation to develop the newly proposed approach.

Stage 2: Formulate the conceptual methodology. The conceptual system development methodology presents the preferred approach to system development for embedded applications and is proposed in this stage.

Stage 3: Develop the proof-of-concept tool. The proposed conceptual methodology from stage 2 needs to be supported by proof-of-concept tool. This stage will select suitable tools that already exist and will develop other tools that are not readily available.

Stage 4: Evaluate the methodology. This stage evaluates the tools using a number of experiments and collects feedback from potential users or participants.

Stage 5: Refine final methodology. The last stage collates the feedback from the evaluation carried out in stage 4 and further refines the proposed methodology. An illustrated documentation for the final system development methodology is also presented.

The following sections will describe the required stages, the associated objectives and processes necessary to realise each objective. Each stage will be expanded into detailed research activities, which will be presented as separate chapters in the remainder of the thesis.

4.3.1 Stage 1: *Review* of the existing system development methodologies

This stage aims to study the relevant body of knowledge for approaches and ideas in order to understand and review of the existing system development methodologies for embedded applications. According to Baines [1995], there are basically three approaches to develop a new solution. The first approach is to ignore the existing knowledge which allows for the uninfluenced development of a new methodology. This approach will require a considerable amount of effort and may ultimately develop a methodology that already exists. The second approach is to develop a methodology based on existing knowledge from the literature. However, the weaknesses in the existing literature may mislead research efforts and deliver a sub-optimal solution. Both the approaches require testing to be carried out to gain confidence that the methodology is suitable to be used in system development for embedded applications in order to avoid being criticised as an unsupported solution. The third approach is a combination of the first and second approaches which critically assesses the capabilities and effectiveness of existing methodologies and identifies the shortcomings. The shortcomings are then addressed by a new methodology.

The third approach provides an incremental advance in existing methodologies with the addition of new concepts and is the most appropriate for this stage of research. It builds on previous literature and avoids developing something that is already available. Therefore, stage 1 of the research is to review of existing system development methodologies for embedded applications. This is followed by a detail literature review focused on the system development methodologies using international standards such as SystemC, UML and

Function Block with special emphasis on free-form or box-line type system approaches. The analysis of current system development methodologies for embedded applications is carried out by looking at their comparative strengths, weaknesses and opportunities of improvement based on the common evaluation criteria mentioned in Section 3.2.8. From the analysis, it is essential to capture the capabilities and effectiveness of current system development methodologies and to identify the shortcoming. These shortcomings are then addressed by the newly proposed methodology.

4.3.2 Stage 2: Formulate the conceptual methodology

The second objective of the research is to formulate the conceptual system development methodology for embedded applications. The state-of-the-art review of stage 1 provides the foundation and exposes the weaknesses of various system development methodologies. Once the weaknesses are identified, the challenge is then to design a new conceptual methodology that can take the best from the existing research to meet the industrial needs. The conceptual methodology will be generic and will not be constrained by existing available tools so that it can serve as a guide for future research directions. The methodology consists of steps that represent the process of system development for embedded applications and the interrelationships amongst them.

4.3.3 Stage 3: Develop of the proof-of-concept tool

The third objective of the research is to select and develop the proof-of-concept tool for system development methodology. According to Nunamaker et al [1991], a system development methodology cannot be verified mathematically and as a result must be tested empirically through observation and experimentation, which requires proof-of-concept tool to demonstrate the validity of the approach. Once the proof-of-concept tool has been built, researchers can use them to build experimental applications to study its performance and the phenomena related to its use to gain more insight.

The conceptual methodology from stage 2 provides a preferred approach for the selection and development of the proof-of-concept tool. It puts the key software componentsinto their correct perspective, specifies their functionality and defines the structure relationships and interactions among the software components. This stage will convert the conceptual methodology into system architecture, identify the key software components and adopt some features of existing software components into the proof-of-concept tool to support the methodology. Those software components which are unavailable will be developed to bridge the gaps. In order to verify the proof-of-concept tool, an experimental application is developed. The system architecture and proof-of-concept tool are then refined to produce the final proof-of-concept tool. Building the proof-of-concept tool will allow further understanding of the concept,

methodology and design as well as gain insight into the problems and complexities of the tools. In addition to the methodology, the proof-of-concept tool will be built for experimental applications for observation and study in the next stage.

4.3.4 Stage 4: Evaluate the methodology

The task in stage 4 is to evaluate and test the proposed methodology using proof-of-concept tool developed in stage 3. The embedded system design requires multidisciplinary expertise as the subsystems are implemented by different disciplines. Feedback from multidisciplinary manufacturing and automation engineers is important to understand their expectations. This stage will select the industrial participants and build experimental applications to test the methodology using proof-of-concept tool. The evaluation will comprise both industrial experiments and feedback from a workshop. The experimental approach is likely to provide the most trustworthy results. The workshop feedback solicits an assessment from a number of different people. However, in this case, participation in the workshop feedback is at the discretion of the attendees.

4.3.5 Stage 5: Refine final methodology

This is the final stage of the research programme. This stage deals with the collation of the lessons learned from stages 1 to 4, identifies any shortcoming with the proposed methodology and suggests refinements which could then contribute to the potential future research direction. The fully documented refined final system development methodology together with the proof-of-concept tool is the outcome of the research programme. The methodology is presented in the form of a guide to aid engineers, practitioners and researchers in the system development of embedded applications. The proof-of-concept tool in the form of computerised software is provided to aid in its illustration and demonstration. The overall research programme is illustrated graphically in Figure 4-2. Both stage 3 and stage 4 are further decomposed into multiple sub-stages in Figure 4-3.

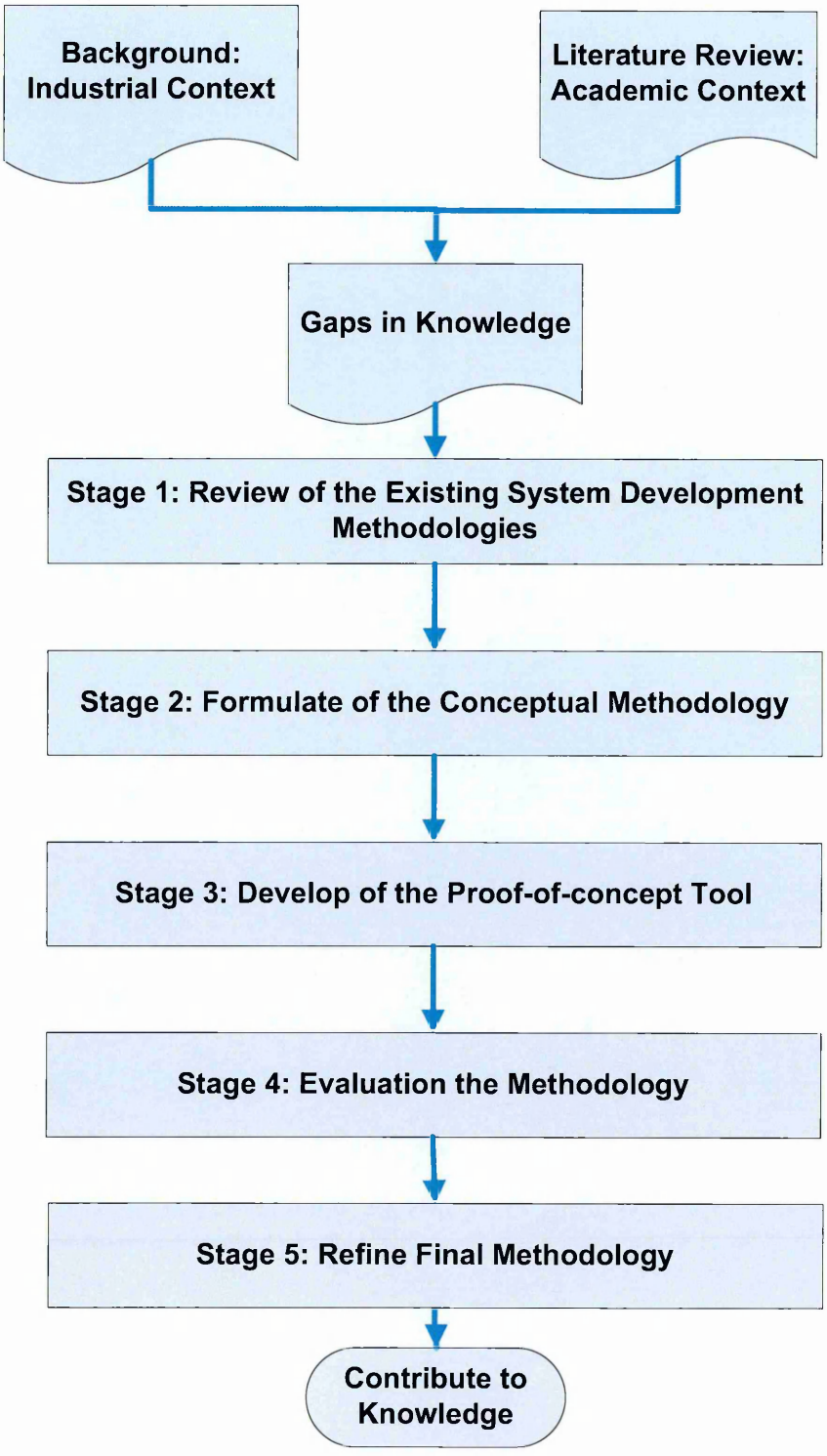


Figure 4-2: Research programme

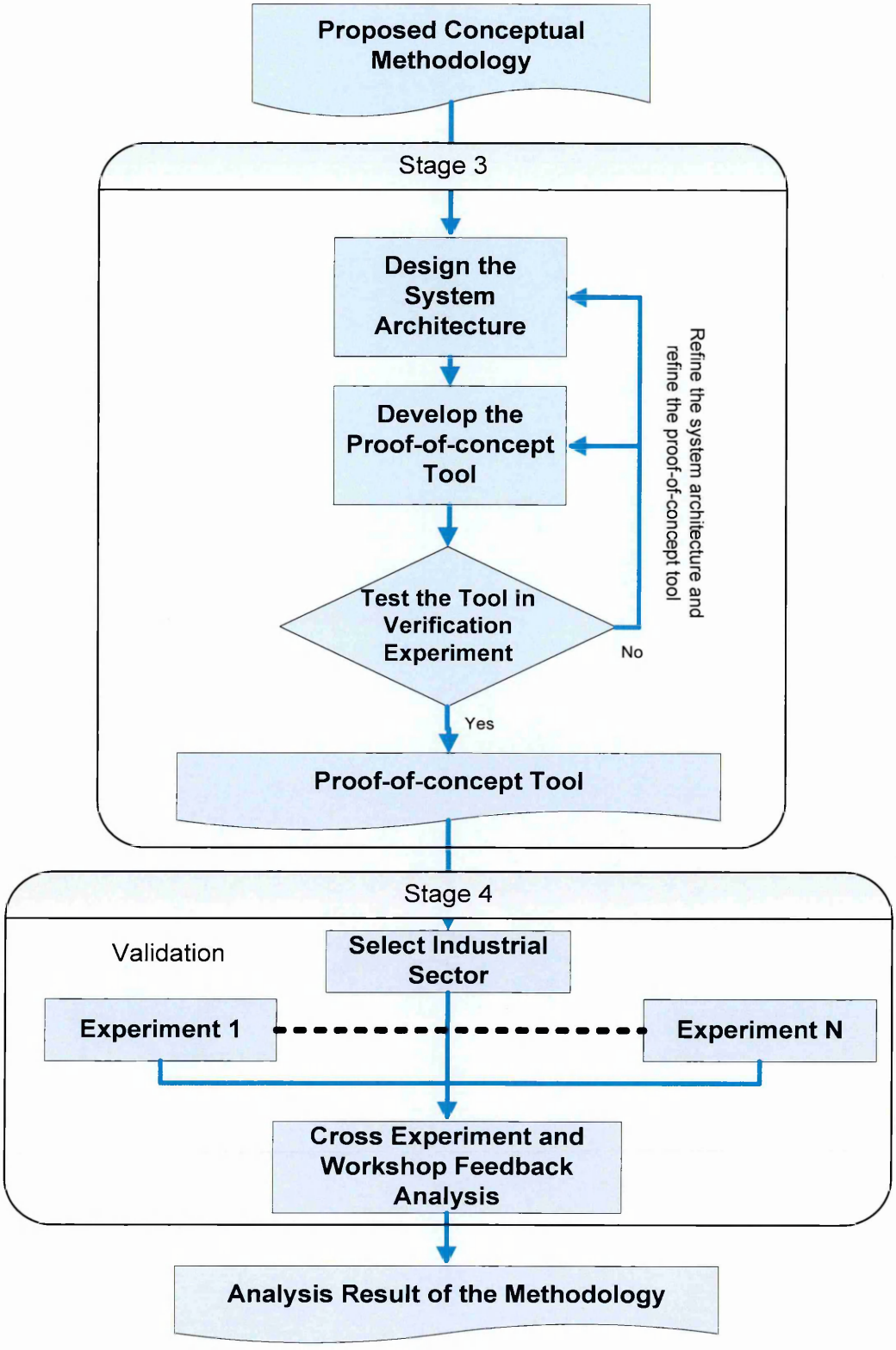


Figure 4-3: Decomposition of stage 3 and stage 4

4.4 Chapter summary

This chapter has defined the research aim and objective based on the industrial problems identified in Chapter 2 and the outstanding issues identified from previously published work detailed in Chapter 3. A five-four stage research programme is defined based on the research objectives. The research programme starts by reviewing of the existing system development methodologies from the academic literature to form the essential foundation for a new methodology. This leads to the formulation of a conceptual methodology that will guide future research direction and implementation. The proof-of-concept tool will provide a test bed for experiments and future improvement. The evaluation stage will use experimental and feedback approaches to validate the proof-of-concept tool for the system development methodology. Finally, analysis of the experiments and feedback from potential users will lead to the refinement and illustration of the final system development methodology.

CHAPTER 5 REVIEW OF THE EXISTING SYSTEM METHODOLOGIES

This chapter presents the first stage of the research which focuses on reviewing of the existing system development methodologies for embedded applications. The purpose of this chapter is to review and analyse the current state of research in system development methodologies for embedded applications, using the common evaluation criteria outlined previously in Section 3.2.8. The observed gaps and limitations of existing methodologies will be addressed through the development of additional proof-of-concept tool that form the new methodology.

This chapter is organised into 5 sections. Section 5.1 discusses the research method for this chapter. Section 5.2 presents the embedded system hardware platforms and the application development processes. Section 5.3 reviews system development related research while Section 5.4 analyses and discusses the strengths, weaknesses of the current research and opportunities for future enhancement.

5.1 Research method

This chapter addresses objective 1 of the research highlighted in Section 4.2, which is to review the existing system development methodologies and identify the gaps in these methodologies. As mentioned in Section 4.3.1, the theoretical evaluation focuses on the incremental changes in present methodologies and tools available for SystemC, UML and IEC 61499 Function Block with a special emphasis on the free-form or box-line type of Function Block. The preference from practitioners on software components, re-usability, design pattern, code generation and evaluation criteria will be used for the selection. As such, the following questions need to be answered.

1. What are the main embedded hardware platforms and what are the limitations of these embedded application development processes?
2. What is the related research relevant to the system development methodologies for embedded applications?
3. What are the development steps, strengths and weaknesses of current research and the opportunities for future work?

The method to address these questions includes:

1. Understand the main groups of embedded hardware platforms, as well as application development processes and identify their limitations.
2. Current system development research primarily focuses on standard-based related work such as UML, IEC 61499 and Function Block. The review of the existing system development methodologies will focus on UML and Function Block. However, the search of literature will be

widened to cover the main embedded hardware platforms highlighted in Step 1.

3. Use the common evaluation criteria from Section 3.2.8 to evaluate the existing system development methodologies highlighted in Step 2. Extract the development steps, analyse the strengths, weaknesses of the current methodologies and highlight opportunities for future research.

5.2 Embedded system hardware platforms and application development processes

This section describes the main embedded system hardware platforms and identifies the main limitations in the application development process to address the first issue mentioned in Section 5.1.

An embedded system is a special-purpose and dedicated functional hardware and software system. There are three main hardware platform groups to implement the embedded system. The first group consists of the Central Processing Unit (CPU)-based hardware such as microprocessors, microcontrollers and digital signal processors. The second group consists of reconfigurable gates chip such as Complex Programmable Logic Devices (CPLDs) and FPGAs while the third group is called Application Specific Integrated Circuits (ASIC). [Calyez & Pasquier, 1998; Talavera et al, 2004; Zurawski, 2006; Salewski & Taylor, 2007]

The following sections discuss the comparison of microprocessor, FPGA and ASIC. Their detail attributes are discussed below.

5.2.1 Microprocessor

A Microprocessor is a programmable digital electronic component that incorporates the functions of a CPU on a single semiconductor integrated circuit (IC). Microprocessors typically serve as the CPU in a computer system, embedded system, or handheld device [Simsie, 2003; Zurawski, 2006]. The microprocessor provides a general purpose control system that can be adapted to a wide selection of applications with little circuit modification.

5.2.2 FPGA

An FPGA is a semiconductor chip containing programmable logic gates and programmable interconnects [Simsie, 2003]. More complex combinational functions and duplicated functionality of basic logic gates can be created by configuring the programmable logic gates. Since control system requirements change from time to time FPGAs can be reconfigured without significant hardware rework. FPGA can execute parallel operations so that multiple logics can be executed concurrently to speed up the operation. With the increasing density and complexity of the gates, FPGAs are becoming more attractive to

embedded system platforms. Some FPGA developers are providing low-cost Intellectual Property (IP) cores or building blocks that allow faster time-to-market and greater design flexibility. FPGA costs lower than ASIC for moderate volume applications and more flexible than the other alternate approaches [Altera, 2003; Zurawski, 2006]. Traditionally, embedded systems have only one data bus and one address bus. In contrast, FPGAs have more data bus and address bus, which allows for simultaneous transactions. Therefore, systems will have higher throughput using FPGAs than microprocessor-based systems. An FPGA is an attractive solution for quick prototyping, reconfigurable application and quick time to market.

5.2.3 ASIC

An ASIC is a customised chip for a particular application. The advantage of ASICs is that they are designed and optimised for specific applications from the performance point of view. The manufacturing process for an ASIC can be optimised to exclude unnecessary circuitry due to the fixed functionality which thus reduces unit cost. The ASIC design cycle takes a long time and upgrading or designing corrections take a similarly large amount of time and cost [Altera, 2003; Zurawski, 2006]. Therefore, the investment risk for an ASIC is higher than that for FPGAs. Moreover, Non-Recurring Engineering (NRE) and development costs are higher due to the complexity of ASIC design. [Zurawski, 2006; Aslam, 2007]. To overcome this, FPGA offer a practical and competitive alternative to ASIC solutions.

Hence considering the implementation of the embedded system, FPGAs superior attributes than those of ASICs from the point of view of development cost, flexibility, prototype, time-to-market and design cycle time. When FPGAs are compared to microprocessors, a microprocessor platform is more suitable for general purpose control system and applications. Therefore, both FPGA and microprocessor-based platforms are chosen in this thesis and used in the experimental verification of the proposed development methodology for embedded control systems.

The application development process consists of broader and more comprehensive activities such as system level design, high and low level languages, compilation, placement, routing, configuration and debugging.

5.2.4 FPGA and ASIC development process

The ASIC development flow involves a wide variety of complex tasks such as placement and physical optimisation, clock tree synthesis, signal integrity and routing. When compared to FPGA design flow, ASIC design flow is much more complex and needs many different Electronic Design Automation (EDA) software tools to implement the complex tasks [Altera, 2003]. FPGA development can be accomplished with a single software tool. Table 5-1 shows

a comparison of Microprocessor, FPGA and ASIC design flows. It is noticeable that a common system level development methodology for Microprocessor, FPGA and ASIC-based embedded applications is lacking. Both FPGA and ASIC design flows require the definition of the design specifications such as input/output standards, memory requirements and the selection of the hardware platform. Different FPGA chip vendors support different input/output standards and different numbers of input/output pins. Design development of FPGA and ASIC uses HDL such as Verilog or VHDL for Register Transfer Logic (RTL) coding, specification of the external and internal memory. Functional simulation and synthesis are similar in both FPGA and ASIC design flows. ASIC place-and-route processes are more complex and differ somewhat from those for FPGAs.

5.2.5 Microprocessor and FPGA development process

Compared to FPGA and ASIC, microprocessor development is similar to PC-based applications which produce object files and usually invoke the linker after compilation. This provides the linking of object files and library files to form an executable file. The executable file provides instructions to control the sequence of logic in the microprocessor. UML is commonly used as a system level design tool for software engineers for PC-based applications. Operational functions are described in class diagrams, use case diagrams, which UML software tools translate to high level programming languages such as C++ or Java. For each compilation in microprocessor development, an object file is produced. After compiling, the linker is usually invoked to link the object files and the library files into an executable file or a Library file. The compiled microprocessor program controls the sequence of logical gates, which is used to write data into data buses, latches and registers and across channels.

For FPGA and ASIC programming, HDL is compiled into RTL. The compilation process for both FPGA and ASIC is similar but is differs from that of microprocessors. The synthesis process is required to generate files in Electronic Design Interchange Format (EDIF) that can be converted to "Bit" files which then configure gates, fill registers and memory on an FPGA and ASIC [Lee, 2002].

The synthesis step defines gate patterns according to the logic of the program while the microprocessor compiler produces bits to control fixed-gate patterns. One of the advantages of FPGAs is that the system can be designed in textual or graphical format and then synthesised into logic gates for execution. After synthesising the program, EDIF files are placed and routed into the FPGA. For FPGA development, bit files are used to configure gates in FPGA. For microprocessor development, object files are loaded onto Read Only Memory (ROM).

As shown in Table 5-1, a common universal system level development methodology or standard for microprocessor, FPGA and ASIC design is lacking. Without this, engineers need to learn many different proprietary development tools and computer languages which increases the engineering and development cost. The company and its engineers require constant training in order to keep up with the rapid changes in technology. Furthermore, most engineers focus on single chip level design and do not consider distributed solutions, because the designers tend to have very specific knowledge of low level gates, I/Os and electronics circuits rather than system level applications.

Table 5-1: Comparison of microprocessors, FPGA and ASIC development processes [adapted from Klingman, 2006; Aslam, 2007]

Development Process	Embedded System Hardware Platforms		
	Microprocessor	FPGA	ASIC
System level development methodologies	UML or IEC 61499 Lack of common standard	Lack of common standard	Lack of common standard
High level languages	↓ C++, C, Java etc	↓ SystemC, Handel C	↓ SystemC, Handel C
Low level languages	↓ Assembly language	↓ Verilog, VHDL	↓ Verilog, VHDL
Compilation	↓ Compile program into object code	↓ Compile program into RTL	↓ Compile program into RTL
	↓ Link objects into executable file	↓ Synthesise program into EDIF timing and area optimisation	↓ Synthesise program into EDIF timing & power analysis
Placement & routing	↓	↓	↓ Test synthesis (scan, insertion, synthesis)
	↓	↓ Placement and route the logic components and interconnection	↓ Placement and physical optimisation
	↓	↓	↓ Clock tree synthesis
	↓	↓	↓ Routing
Load & configuration	↓ Load program into ROM	↓ Load configuration file (.bit) to configure the FPGA	↓ Signal integrity
Debug & test	↓ Debug software	↓ Debug software	↓ Test the chip

Research into embedded application development methodologies is one of the interesting areas of research in recent years. A detailed review of UML and Function Block-based methodologies for embedded application development is presented in the following section.

5.3 Review of system development methodologies

5.3.1 UML-based system development methodologies

UML is a well accepted modelling standard for desktop PC-based software engineering. There are different research approaches using UML as a system development methodology for both FPGA and microprocessor-based embedded applications. The following section discusses the research using UML-based system development methodologies for embedded applications.

1. UML to Synthesisable SystemC

An UML-based methodology was proposed by Tan et al [2004] using Rational Rose RT tools to generate SystemC code as shown in Figure 5-2.

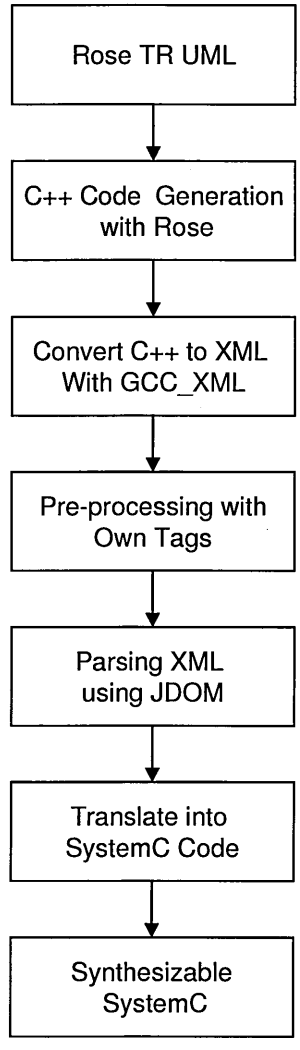


Figure 5-1: UML to System C [Tan et al, 2004]

In his research, Rational Rose RT which can support the development of UML compatible system descriptions is chosen. In addition, Rational Rose RT can generate C++ code structure from UML's class diagrams, state diagrams and structure diagrams. After translation from UML to C++ codes using Rational Rose RT, GCC_XML [GCC, 2007] is used to convert C++ into eXtensible Markup Language (XML) documents. XML documents are required to modify with pre-defined tag to convert to SystemC code using Java-based document object model (JDOM) [Curtin, 2007]. For suitably restricted designs, the generated SystemC code is synthesisable in that it is accepted by the Synopsis CoCentric compiler [Synopsis, 2008].

In his research, it is evident that there were many steps necessary to convert UML to SystemC. If UML can be converted directly to SystemC directly without using C++ code, this methodology will be better. If the researchers wish to provide both C++ code for software specifications and SystemC code for hardware specifications, UML to C++ and UML to SystemC could be an option that is selectable by users. For example, after converting from UML to XML, that XML can be used in either C++ code generation or SystemC code generation as required. Tan's methodology is proprietary and only uses Rational Rose RT, without this tool, the methodology does not work. The attributes generated can only be accessed and modified by Rational Rose RT. User-defined C++ code and functions can only be invoked in the 'Action' section of a state transition in state diagrams, which are not truly object oriented.

2. UML to HDL

Coyle & Thornton [2005] focus on the use of Model Driven Architectures (MDA) as a methodology to define high-level model-based system descriptions. This methodology, from MDA to HDL hardware descriptions, is an effort to use a model-based architecture to further optimise and automate the embedded systems development. MDA is an extension of the standard UML which is used to configure the subsystems with system functionality so that subsystems activation is correlated with scenarios of usage described in the UML's use cases. As shown in Figure 5-2, UML system models are used to define system requirements. System functional requirements are described as classes, use cases and state diagrams in the UML model. Non-functional requirements consist of system quality attributes such as performance and timing requirements, defined by UML profiles which can be added to UML models.

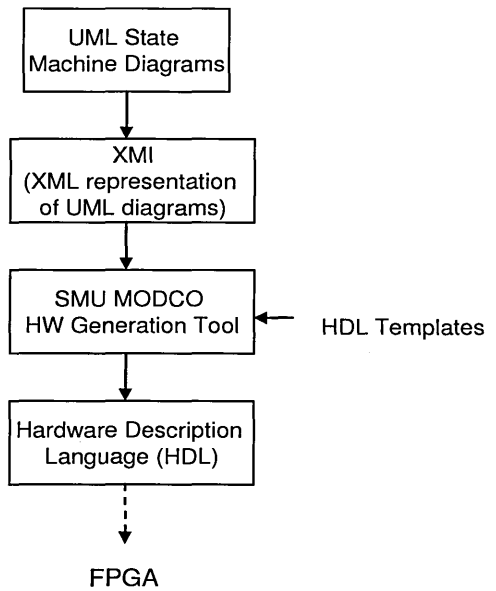


Figure 5-2: UML to HDL [Coyle & Thornton, 2005]

UML diagrams are converted to XML documents using XML Metadata Interchange (XMI) which is a standard XML-based intermediate form. The initial drive for XMI was to enable UML diagrams to be imported and exported across different UML tools. Those XML documents are used to generate HDL codes by using an XML parsing utility called MODCO tool and HDL templates [Coyle & Thornton, 2005]. MODCO is a transformation tool which takes a UML state diagram as input and generates HDL output suitable for use in FPGA circuit design. This approach tried to bridge the gap between hardware and software design with broad applicability for embedded systems development. UML provides benefits from the object-oriented point of view such as modularity, re-usability, concurrency etc. UML is normally used for desktop PC-based software applications and is not suitable for embedded systems development.

3. UML State machines to SystemC

Boudour & Kimour [2006] used UML state machines and object diagrams for system development. The objective of their research was to facilitate the building of reliable software tools by assembling and integrating components while validating and evaluating functional and non-functional requirements. Figure 5-3 shows the transformation process from UML to SystemC. Here, state machines and object diagrams are defined by the developer to generate the corresponding XMI file to build dependency state matrix. SystemC code is generated using the state machine and a specific SystemC template with appropriate mapping rules.

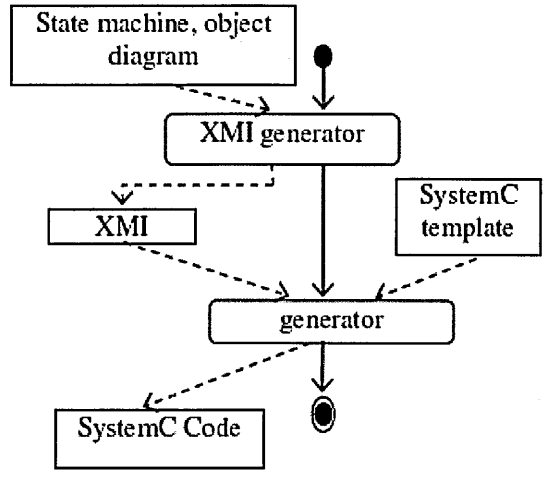


Figure 5-3: UML to SystemC [Boudour & Kimour, 2006]

Boudour & Kimour [2006] did not provide in depth technical detail on how the system processes from UML to SystemC code. Generating XMI from UML using an XMI generator might be similar to the approach of Coyle & Thornton [2005]. Unfortunately both did not provide the details and the tools are not available for evaluation. Furthermore, both methodologies focus on single embedded systems and do not consider distributed industrial applications. Both methodologies cover only conversion of a UML-based design to a SystemC and did not cater for other embedded system languages.

4. MDA-BASED System on Chip (SoC)

Wang et al [2006] proposed an MDA-based approach which combines the capability of UML with SystemC by extending UML to express SystemC language. This approach provides semiautomatic conversion from UML to SystemC code.

Figure 5-4 shows the flow of the MDA approach. UML is used to capture the specification but not the implementation detail. The UML will generate the SystemC code skeleton but detailed code development is done manually in SystemC level.

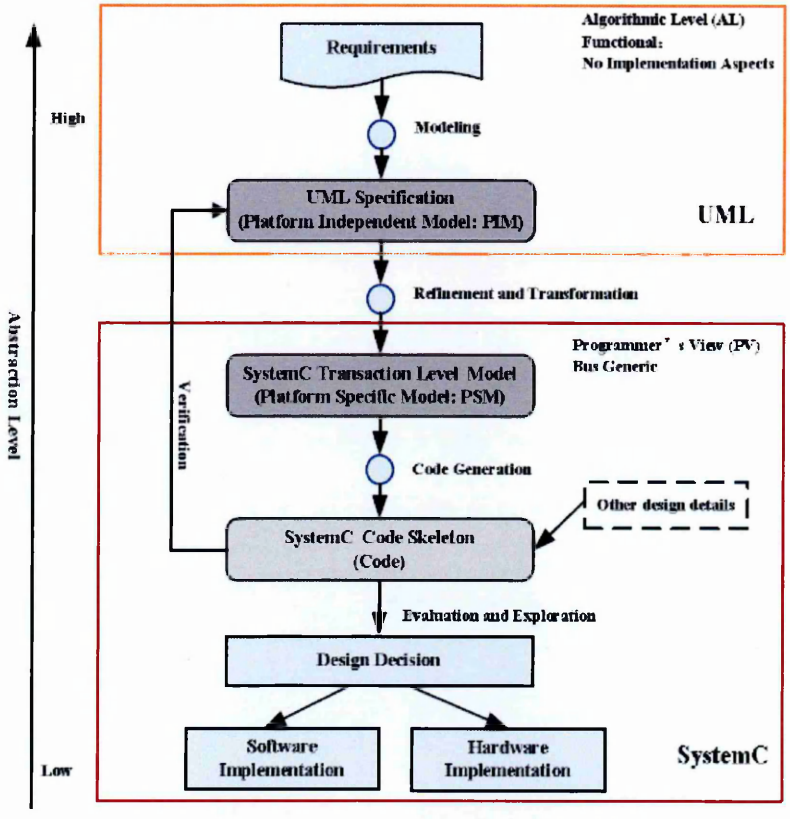


Figure 5-4: MDA-based SoC design flow [Wang et al, 2006]

5. UML-Based Multiprocessor SoC Design Flow

Kangas et al [2006] presented a UML-based system development methodology for multiprocessor SoC which covers the system modelling to the FPGA prototype as shown in Figure 5-5. UML is used for architecture modelling, application modelling and model mapping for FPGAs. The proposed flow includes the automatic code generation, simulation and verification for FPGA. However, they did not provide tools for evaluation or testing.

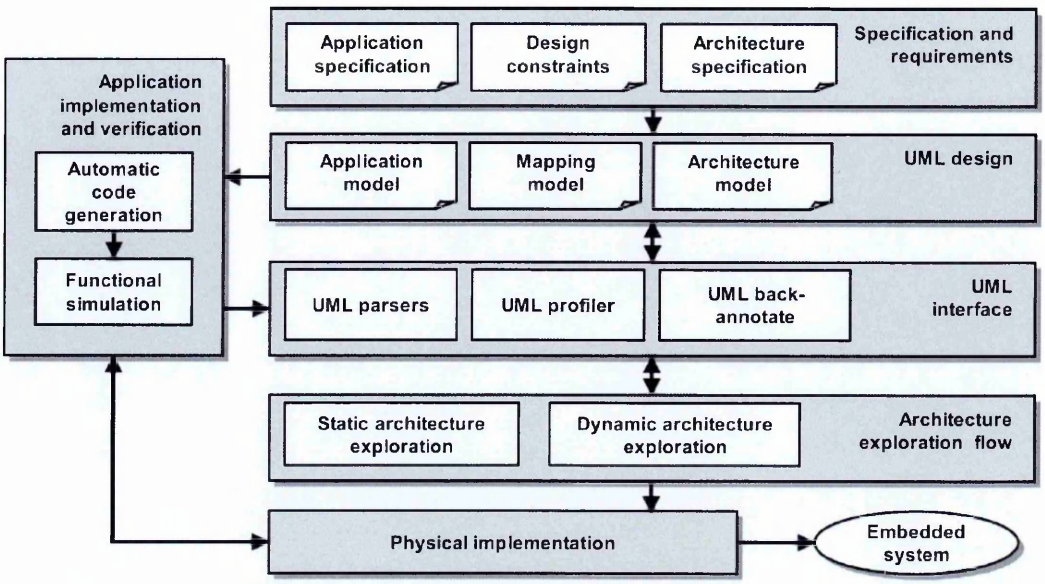


Figure 5-5: UML-based multiprocessor SoC system development methodology [Kangas et al, 2006]

6. StateCharts to SystemC

Mura et al [2007] presented a methodology that converted specifications written in StateCharts to SystemC as shown in Figure 5-7. The approach is relatively simple and reusable. The system can be modelled at different levels of abstraction. The StateCharts are converted to SystemC using XMI as an intermediate representation. Similar to most other research in UML, the tools are not available for evaluation or testing.

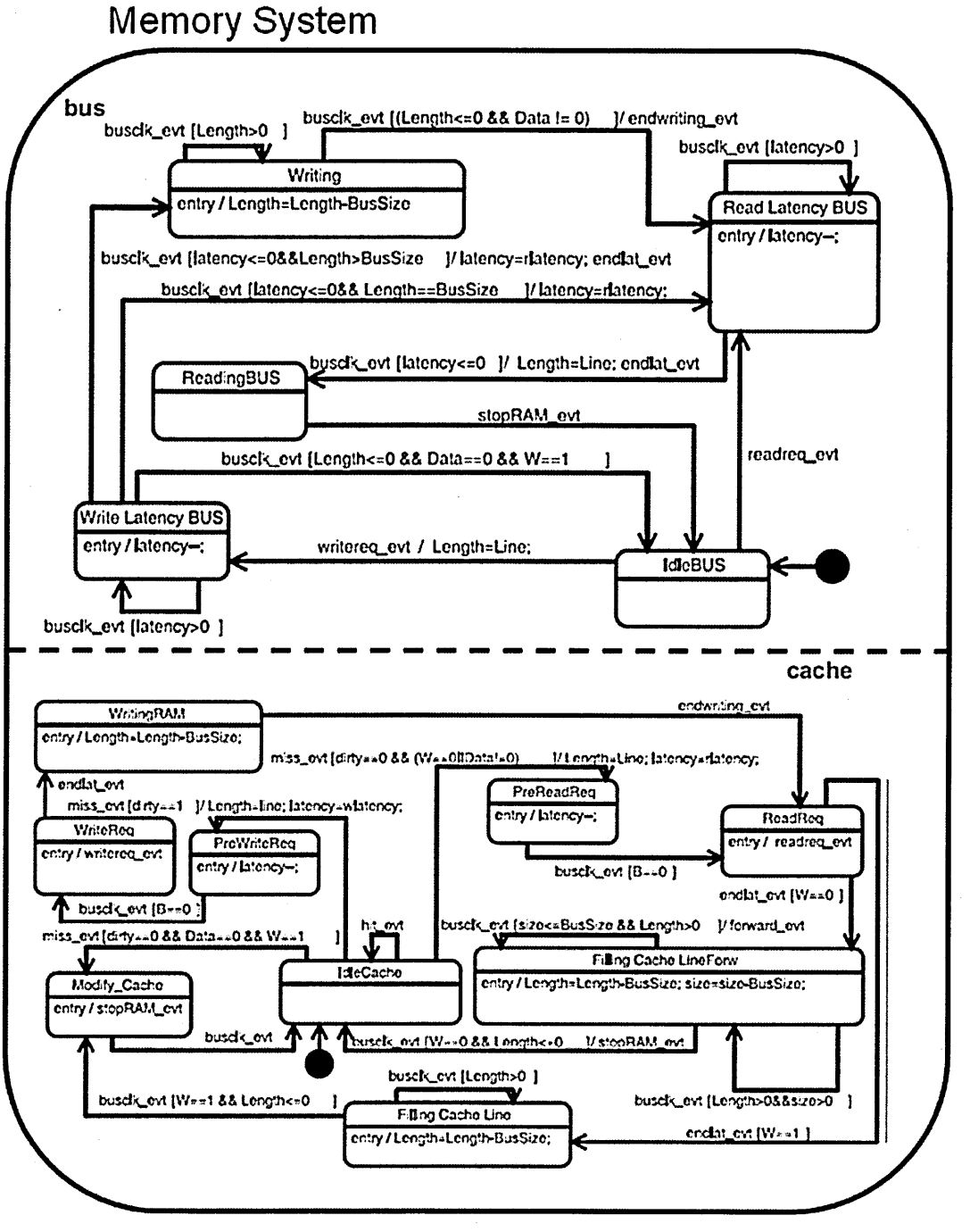


Figure 5-6: The StateCharts diagram [Mura et al, 2007]

The research highlighted above is more focused on using UML or a subset of UML such as MDA or StateCharts to generate SystemC codes which can then be converted HDL for FPGA. Most of the research caters for system level design for only FPGA and bridging software engineering methodologies to

electronic gates design. Detailed research and software tools were not made available for further investigation. The next section presents IEC 61499-based system development related research

5.3.2 IEC 61499-based system development methodologies

The International Electro-Technical Commission (IEC) published the IEC 61499 Function Block standard in 2005 to standardise the application of software modules in distributed industrial control systems [IEC, 2005]. Various researchers have explored IEC 61499 as a viable system model to support configurable, portable and interoperable distributed automation components [Khalgui et al, 2004; Auinger et al, 2005; Vyatkin et al, 2005a; Thamboulidis, 2006; Dubinin & Vyatkin , 2006a; Dubinin et al, 2006b; Valentini et al, 2006]. The following section discusses the prominent research using IEC 61499 as a system development methodology for embedded applications.

1. The Model-view-controller (MVC)

Brennan et al [2004] propose software design patterns to assist in the design and development of distributed embedded control software. As shown in Figure 5-7, the model-view-controller (MVC) divides an interactive Java-based embedded application into three components using an IEC 61499 Function Block:

i. Human Machine Interface (HMI) view that is used to display information to users

ii. Controller view that is used to handle user input and

iii. Model view that contains the core functionality of the application and the data.

The development tool is available for evaluation but the research mainly focuses on design patterns and Java-based embedded devices. The research does not cover other embedded platforms and languages.

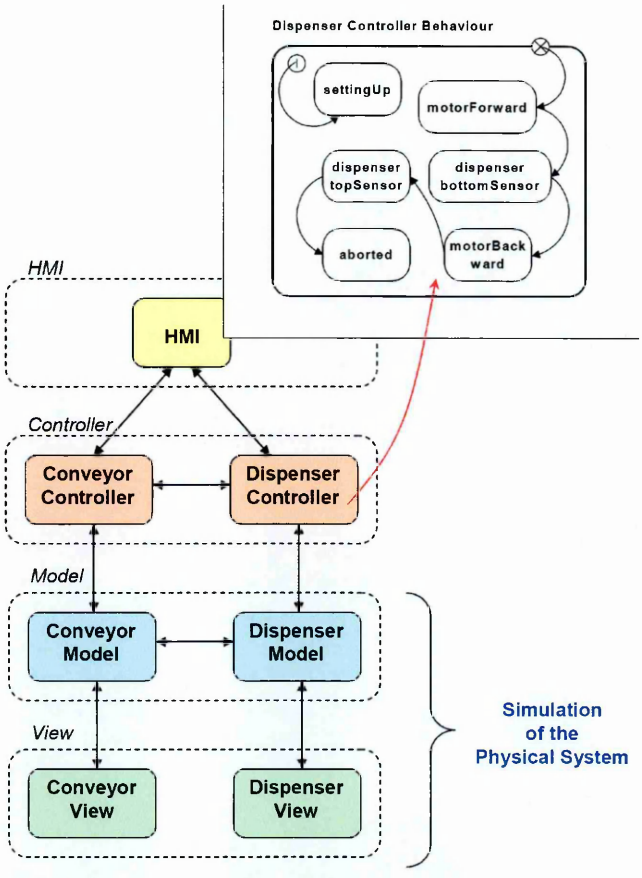


Figure 5-7: The model-view-controller [Brennan et al, 2004]

2. Automation component

Cengic et al [2006] propose a methodology using IEC 61499 automation component-based for distributed embedded control software. The new automation components can be hierarchically embedded into other components to produce new components or applications. As shown in Figure 5-8, there are 4 layers that include the:

i. Model subcomponent layer which provides basic functions of an automation component. The component might be a unit of the controlled system (e.g. embedded controller), functional (e.g. transportation operation), or structural (e.g. motor).

ii. Controller subcomponent layer, which provides functions for the automation component. It represents functions ranging from polling sensors' status to complex control algorithms.

iii. Interface subcomponent layer, which represents a method of interaction with the automation components. It gives access to other components that need to interact in the automation component hierarchy.

iv. Service subcomponent layer, which provides functions for monitoring and manipulating the automation component's status and behaviour using graphical user interface or some other kind of man-machine interaction interface.

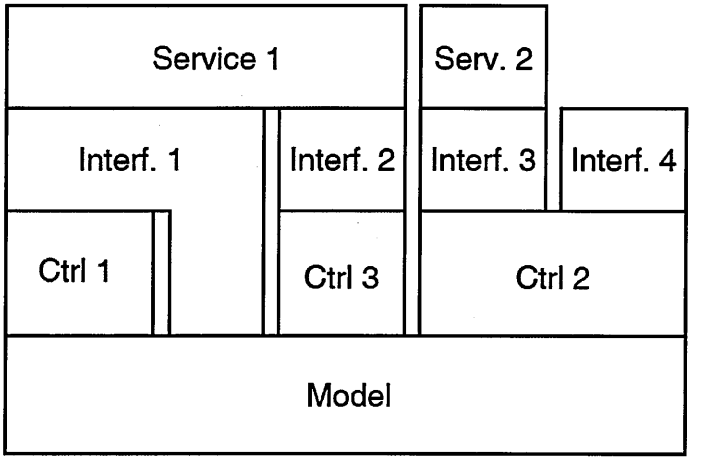


Figure 5-8: Sample architecture of automation components [Cengic et al, 2006]

The methodology is inspired by the VHDL component concept for FPGA but the research focuses mainly on Java-based embedded environments. The tools used in the methodology were not discussed.

3. visualSTATE

visualSTATE is a model-based system development methodology for embedded applications proposed by Marian & Guo [2006]. As shown in Figure 5-10, the model-based design uses IEC 61499, prefabricated components and code generators together with visualSTATE which is based on StateCharts and dataflow diagrams. IEC 61499 was used in both the design and validation space. They proposed visualSTATE as a paradigm for system development that includes explicit tools and operational descriptions based on formal methods. This technique captures executable model descriptions. The basic paradigm shift is that the software engineers become model builders, instead of programmers. The advantages of the methodology include a well structured design, flexibility and complexity, scalability and consistency between the design, testing and the implementation. Unfortunately, the research discussion mainly addresses the StateCharts to Function Block design whereas no discussion on the actual implementation of the methodology is provided.

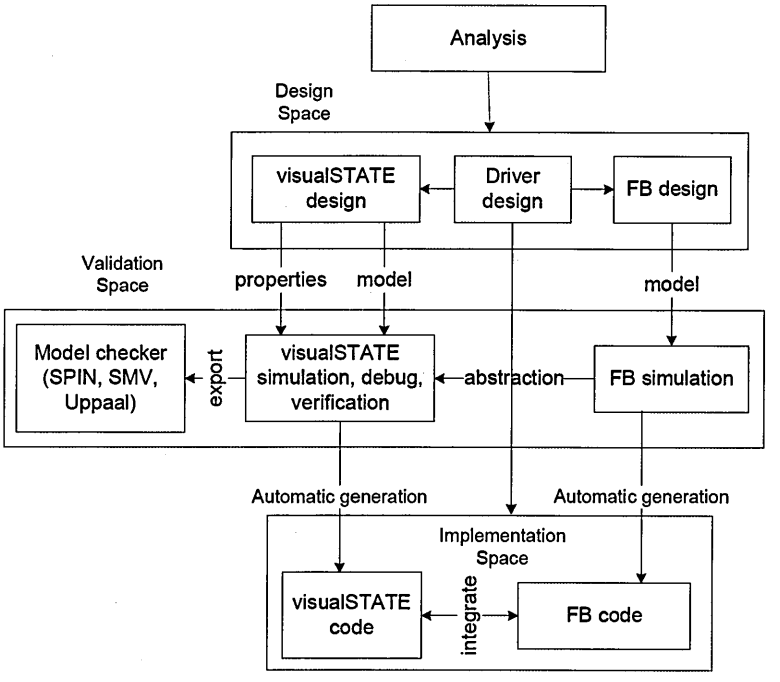


Figure 5-9: Proposed model-based design methodology [Marian & Guo, 2006]

4. Embedded Systems' Engineering Support Environment (eSESE)

Thamboulidis et al [2007] proposed a service-oriented computing with semantic web integrated system development methodology for embedded applications. It is an embedded system development process where the required features are defined by users as web-services and will be published. The developer will construct the project specific integrated development environment and construct the embedded solution by selecting from existing IEC 61499 device models. Figure 5-10 shows three layers of services namely the application layer, service layer and knowledge layer. The application layer allows vendors and developers to publish, search and configure their available services. The service layer represents the services that are available while the knowledge layer has a collection of various IEC 61499 and device ontologies for selection. The methodology is only at the proposal stage, it does not discuss details of the creation of the embedded solution process.

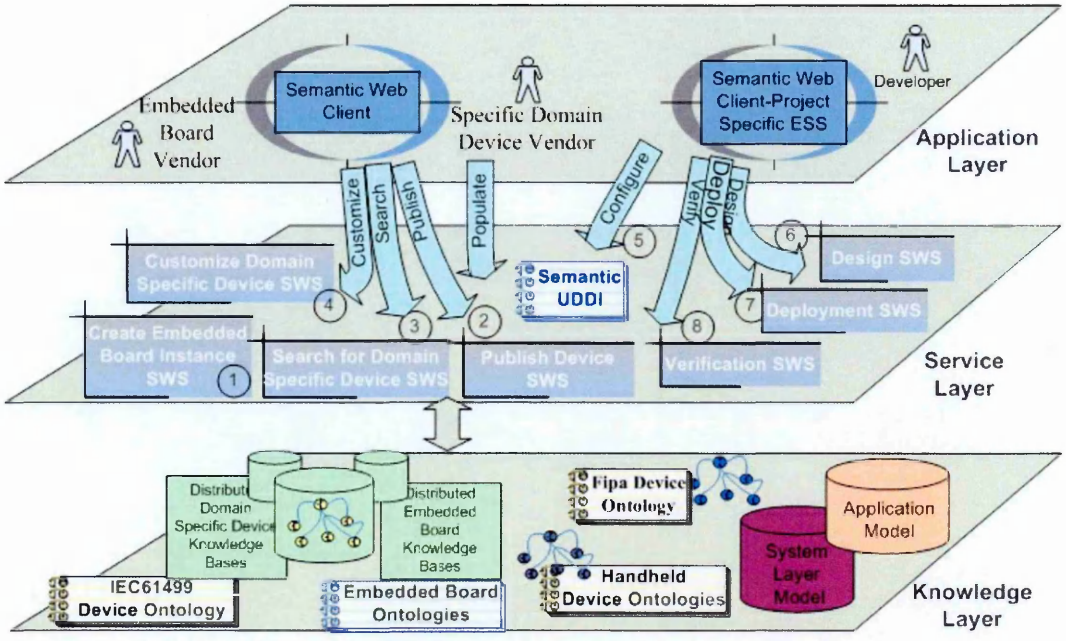


Figure 5-10: Proposed semantic web-based methodology [Thamboulidis et al, 2007]

5.3.3 Other methodologies

Petri Nets have also been explored as a system development methodology and code generator by many researchers including Gomes et al [2004], Hagge & Wagner [2005] and Nunes et al [2007] use Petri Nets as modelling diagrams to capture the design and to convert an XML-based intermediate language to C and potentially to VHDL. The research interest is to provide a system development methodology using open standard, XML-alike tools for embedded systems.

5.4 Analysis and discussion

This section presents the analysis of the system development methodologies for embedded applications based on the evaluation criteria from Section 3.2.8. The goal is to evaluate the existing system development methodologies, extract the development steps, analyse the strengths and weaknesses of the current methodologies and identify opportunities for future research. The survey of the UML-based system development methodologies is summarised in Table 5-2 and that for the IEC 61499-based methodologies is summarised in Table 5-3. The first 4 columns present the methodologies, processes, techniques and the graphical representation. The remaining columns show the evaluation criteria highlighted in Section 3.2.8. The subsequent columns show the common development stages that were discussed in some methodologies. The last column shows the generated computer languages and indicates if the

methodologies have been tested with some embedded hardware. There are 10 system development methodologies for embedded application which were identified as research relevant to the work of this thesis. Analysis of the strengths, weaknesses and opportunities analysis of the methodologies are presented below.

5.4.1 Strengths of current research

1. UML-based methodologies

UML is commonly used as an objected oriented modelling of software system designs for desktop-based computer applications as it is a collection of various diagrams that are mainly used for software design specification.

As shown in Table 5-2, the development methodology from Tan et al [2004] generates C++ using Rational Rose RT version of UML tools. The system will be converted the C++ to XML using GCC_XML tool. The result will convert into SystemC which is subset of C++. SystemC and could be used in microprocessor or FPGA-based platform Fowler's [2004].

UML to HDL methodology is the simplest amongst the reviewed approaches. He developed a proprietary tool to convert from UML to HDL [Coyle & Thornton, 2005]. The methodology uses MDA and state diagrams. MDA is an extension of the standard UML.

The rest of the UML-based system development methodologies are similar to the first two methodologies, i.e. Tan et al [2004]; Coyle & Thornton [2005]. The commonly used tools are MDA, state diagrams for detailed representation, XMI as an intermediate representation and SystemC as the generated source code. One of the strengths of the research is the use of intermediate representations of XMI and SystemC which are both international standards. An international standard approach is important to ensure interoperability and should be considered in this research.

2. IEC 61499-based methodologies

The survey of the IEC 61499-based system development methodologies for embedded application is summarised in Table 5-3. The Model-View-Controller (MVC) Methodology from Brennan et al [2004] uses IEC 61499-based design patterns to represent HMI, controller and model views for the embedded application. Cengic et al [2006] propose IEC 61499-based automation components to represent model subcomponent layer, controller subcomponent layer and interface subcomponent layer. visualSTATE (Marian & Guo [2006]) is an embedded development tool using both IEC 61499 and StateCharts. eSESE [Thamboulidis et al, 2007] used a web-based service-oriented methodology together with IEC 61499 for embedded application development.

The strength of IEC 61499-based methodology research is mainly on industrially oriented applications especially for manufacturing and automation. The tools could apply to distributed control and event-based control. Some researchers [Marian & Guo, 2006; Thamboulidis et al, 2007] are exploring links between UML and IEC 61499. The main target users are automation engineers, manufacturing engineers and system integrators. Software engineers can provide components with algorithms and logic which are called automation objects. Zoitl et al [2007] summarised that most of the current IEC 61499 research focuses on system design, development and industrial execution methodologies, definition of semantics and application scenarios. The international standard intermediate representation XML is specified in the IEC 61499 standard which opens opportunities for future expansion of the tools [IEC, 2005].

In addition, there are a few IEC 61499 development tools available for evaluation and testing. For example, freeware Function Block Development Kit (FBDK) [Holobloc, 2008] is commonly used by researchers, a commercial tool from ISaGRAF [Weehuizen et al, 2007] is used for industrial deployment and open source tools like FBench [FBench, 2008] and 4DIAC [4DIAC, 2008] are available for future research and enhancement. Thamboulidis [2005] developed a Common Object-Oriented Real-Time Methodology for Unified Development (CORFU) Engineering Support System (ESS) to translate UML to IEC 61499 and generate Java code for embedded controllers, which is also available for evaluation. All this effort provides the open platform for future research and development. Some of the researchers have deployed the IEC 61499 methodologies in Java-based embedded hardware boards [Brennan et al, 2004, Marian & Guo, 2006, Thamboulidis et al, 2007].

Woll [2007] claimed that IEC 61499 is 'simple, cost effective and repeatable' to meet the industry demand of mass customisation and 24/7 operation. It is a standard specially targeted for automation, system integration and manufacture engineering.

5.4.2 Weaknesses of current research

1. UML-based methodologies

The development methodology from Tan et al [2004] required many steps which require converting from UML to C++ to XML, C++ to XML and XML to SystemC. Since SystemC is a subset of C++, it is not clear why C++ source codes have to be converted to the intermediate XML before they could be converted to SystemC. Moreover, there was no discussion on the testability or interoperability of the tools.

Tan et al [2004] used Rose RT as the modelling tool which is the proprietary commercial version of real time UML tools from Rational. There is no

discussion of the interoperability of the components from other non Rose RT tools.

Most of the other methodologies do not provide detail discussion of the implementation, tools or source code for future research and development.

As shown in Table 5-2, most UML related methodologies do not discuss the application development stages, embedded hardware platform and other computer languages like Java, C and C++. This might be due to the fact that UML provides many different tools and diagrams that can help in various stages of development. Although most of the methodologies generate SystemC solutions which can cater for both microprocessor-based and FPGA-based platform, none of the papers elaborates on the interoperability of the methodologies. Both Kangas et al [2006] and Mura et al [2007] have used simulation but most other research papers do not discuss the testability of the methodologies in industrial applications. Most of the above mentioned UML-based system development tools or source code are not available to the public at this point.

UML was originally designed for desktop PC-based software development. For this reason, an UML methodology is more suited to software engineers than to automation engineers, industrial engineers and non-software experts. UML lacks gate level representation, hardware representation, hardware/software co-design and distributed embedded control applications.

In the industrial survey by Graaf et al [2003], it was found that UML is not common practice adopted by most companies. Projects that employed diagrams mostly use free-form and box-line diagrams. Graaf et al [2003] argued that UML is difficult to use and required specialised skills. Those projects which did use UML found that it was difficult to communicate as most stakeholders did not completely understand the UML diagrams. In fact, some of the engineers just used UML simply as drawing tools.

Dubinin & Vyatkin [2005] highlighted various UML problems including weak orientation of control engineers as advanced software engineering methods, difficulties in implementation of event flows and generation of a fully executable control.

2. IEC 61499-based methodologies

In contrast to UML-based approaches, IEC 61499-based system development methodologies focus mainly on the distributed applications and industrial examples, with Java language being the only generated source code. Most of the generated Java code required a Java virtual machine to be run. There are practically no detail explanations or examples of how IEC 61499 can link to external hardware through an interface card. IEC 61499 tools for FPGA-based embedded systems and other microprocessor languages like C, C++, SystemC

are still lacking. Future research should therefore be focusing on IEC 61499-based system development methodologies for both FPGA and other microprocessor languages with interfaces to the hardware platform. In addition, publicly available tools like FBDK, CORFU and FBench can generate partial Java codes but the tools neither generate C or C++ codes nor address FPGA-based embedded platforms. The only commercial product that supports IEC 61499 is ISaGRAF [Weehuizen et al, 2007]. However, ISaGRAF is a proprietary solution that does not use XML, does not interoperate with other IEC 61499 tools like FBDK, FBench, CORFU and it does not support FPGA-based environment as well.

5.4.3 Opportunities from current research

The weaknesses of the system development methodologies for embedded applications provide the potential opportunities for future research. The first opportunity is to consolidate and simplify the flow of the system development methodology for embedded applications with a conceptual methodology so that it is easier to understand and follow. With the conceptual methodology, researchers could explore existing components and tools as well as develop the missing components as proof-of-concept tool. With the proof-of-concept tool, it is possible to test and validate the research through experiments and feedback.

In addition, organisations such as Open Object-Oriented kNowledge Economy in Intelligent inDustrial Automation (OOONEIDA or O3NEIDA) initiated research on reconfigurable industrial control and automation objects for manufacturing and automation [Vyatkin et al, 2005a]. OOONEIDA strongly supports the IEC 61499 Function Block for automation and manufacturing industries will increase the usage of the standard in both research and industries.

IEC 61499 is a suitable standard for manufacturing and automation related industries. There is a research opportunity to extend the IEC 61499 features to address the interface to hardware, the needs for the FPGA-based embedded systems and various other microprocessor languages like C and C++. This effort would make the system development methodology for embedded applications more easily accessible to industrial engineers and therefore increase the adoption of embedded systems in manufacturing industries.

5.5 Chapter summary

This chapter has presented a brief overview of embedded platforms which include microprocessor, FPGA, ASIC highlighting the differences and gaps in the system development processes for embedded applications. The chapter reviewed various research on system development methodologies for embedded applications with a focus on the UML and IEC 61499 methodologies. The last section analysed the strengths, weaknesses of current research and highlighted opportunities for future research.

Table 5-2: Comparison of UML-based methodologies

Methodology	Process	Technique	Graphical representation	Evaluation criteria						Development steps					Computer languages					Embedded hardware
				Testability	Portability	Component modelling	Interoperability	Understandability	Reusability	Requirement step	Structuring step	Repository step	Intermediate representation	Code generation	Java	C	C++	SystemC	HDL	
Rose RT to SystemC [Tan et al, 2004]	1. Rose RT UML 2. C++ Code Generator 3. Convert C++ to XML 4. Convert XML to SystemC	Rose RT UML, code generator and XML	RT UML		SystemC	✓		✓	✓	Rose RT	Rose RT		XML	✓			Rose RT	✓		
UML to HDL [Coyle & Thornton, 2005]	1. UML State Machine Diagrams 2. XMI (XML representation of UML 3. HDL Generation Tools	UML state machine diagram, MODCO HDL generation tool with XMI	UML State diagram		HDL	✓		✓	✓		✓		XMI	✓					✓	
UML state machines [Boudour & Kimour, 2006]	1. UML State Machine Diagrams 2. XMI (XML representation of UML 3. SystemC Template 4. Generate SystemC Code	UML State machine diagram, XMI and code generator	UML State diagram		SystemC	✓		✓	✓		✓		XMI	✓				✓		

Table 5-2: Comparison of UML-based methodologies (continued)

Methodology	Process	Technique	Graphical representation	Evaluation criteria						Development steps					Computer languages					Embedded hardware
				Testability	Portability	Component modelling	Interoperability	Understandability	Reusability	Requirement step	Structuring step	Repository step	Intermediate representation	Code generation	Java	C	C++	SystemC	HDL	
MDA-BASED SOC [Wang et al, 2006]	1. UML with MDA 2. Convert to SystemC 3. separate into hardware & software implementation	MDA, SystemC	UML with MDA		SystemC	✓		✓	✓	✓	✓			✓				✓		
UML-Based Multi-processor SoC Design Flow [Kangas et al, 2006]	1. System Model and Analysis using UML. 2. FPGA Code generation using XML. Code simulation	UML for modelling, XML	UML	Simulation	SystemC, FPGA	✓		✓	✓	✓	✓		XML	✓				✓		
StateCharts to SystemC [Mura et al, 2007]	1. System Model by StateCharts 2. Extract inform to XMI 3. Convert to intermediate representation 4. simulate in SystemC	StateChart, XMI and SystemC	UML State diagram	Simulation	SystemC	✓		✓	✓		✓		XMI	✓				✓		

Table 5-3: Comparison of IEC 61499-based methodologies

Methodology	Process	Technique	Graphical representation	Evaluation criteria						Development steps					Computer languages					Embedded hardware
				Testability	Portability	Component modelling	Interoperability	Understandability	Reusability	Requirement step	Structuring step	Repository step	Intermediate representation	Code generation	Java	C	C++	SystemC	HDL	
Model-View-Controller (MVC) Methodology [Brennan et al, 2004]	1. a Model that contains the core functionality of the application and the data, 2. a View that is used to display information to users and 3. a Controller that is used to handle user input.	Divide an interactive application into model, view and controller components	Function Block	With both hardware and software	Function block	✓		✓	✓	✓	✓				✓					✓
Automation components [Cengic et al, 2006]	The methodology proposes new software components, called automation components that can be hierarchically embedded to produce new components. Automation components are also combined to produce hierarchical component-based applications.	Use IEC 61499 to define the model, controller, interface and service subcomponen t and link them into system	Function Block	An example control of a fixture, used to fixate parts that are welded to the floor of a car in a manufacturing cell		✓		✓	✓		✓	✓			✓					

Table 5-3: Comparison of IEC 61499-based methodologies (continued)

Methodology	Process	Technique	Graphical representation	Evaluation criteria						Development steps					Computer languages					Embedded hardware
				Testability	Portability	Component modelling	Interoperability	Understandability	Reusability	Requirement definition	Detail design	Design repository	Design representation	Code generation	Java	C	C++	SystemC	HDL	
visualSTATE [Marian & Guo, 2006]	1. Analysis 2. Design space 3. Validation space 4. Automatic generation	StateCharts like visualSTATE with Function Block coordinator	StateCharts and Function Block	Gripper system consists of a robotic arm with shear force sensors, microcontroller	Function block, automatic code generator	✓		✓	✓	✓	✓			✓	✓					✓
Embedded Systems' Engineering Support Environment (eSESE) [Thamboulidis et al, 2007]	1. Application layer 2. Service layer 3. Knowledge layer	using ontology-based device modelling with UML	UML class diagram, Function Block	Proof-of-concept demo		✓		✓	✓	✓	✓	✓			✓					✓

CHAPTER 6 FORMULATION OF THE CONCEPTUAL METHODOLOGY

The previous chapter highlighted the strengths and weaknesses of the development methodologies for embedded systems available in the literature and identified avenues for future research. Based on that, this chapter proposes a new system development methodology. The chapter is organised as follows. Section 6.1 outlines the research method adopted in this chapter, while Section 6.2 discusses the strategy to determine the common steps of the methodology. Section 6.3 describes the content of the proposed methodology followed by a step-by-step description of the methodology and a discussion on the issues involved in each step.

6.1 Method

This chapter addresses objective 2 of the research outlined previously in Section 4.2, which was to formulate the conceptual system development methodology for embedded applications. It is important to formulate a conceptual methodology because it provides a generic guideline comprising easy to understand systematic steps with a pragmatic guide for future research directions and implementation, which is not constrained by currently available tools [Nunamaker & Chen, 1989].

The method adopted to form of the conceptual methodology proposed in this chapter. Consequently, the following steps are adopted.

1. *Determine the development steps:* This involves the extraction of the key steps from existing methodologies. Six UML-based methodologies and four IEC 61499-based methodologies are chosen for analysis. The key steps from the existing methodologies are extracted into development steps for the conceptual methodology. Identifying and extracting the key steps from existing methodologies provides a solid foundation for the proposed new conceptual methodology and is a far more efficient approach than developing a completely new list of unknown steps. Baines [1995] suggested an incremental change in existing methodologies with the addition of new features is the most appropriate strategy for this stage of research. The key steps in UML-based methodologies include the requirement gathering, structuring and code generation, while steps originating from the IEC 61499 methodologies are the requirement gathering, repository, structuring and intermediate representation steps.

2. *Formulate the conceptual methodology:* Having established the structure and content, this step formulates the conceptual methodology by describing the combination of common development steps and contents for each step. During the process of formulating the new conceptual development methodology, emphasis needs to be put on the representation style of the steps, as they are required to be simple and

suitably representative for the majority of engineers. Some of the future research issues will also be highlighted.

6.2 Determination of the development steps

This section establishes the development steps of the conceptual system development methodology for embedded applications. The development steps are based on the identification of the five development steps highlighted in the review of existing system development methodologies described previously in Sections 5.3.1 and 5.3.2. This section therefore illustrates how the steps are mapped with existing methodologies to form the new approach.

The process of mapping the steps of the selected methodologies to the new development steps of the conceptual system development methodology for embedded applications is illustrated in Table 6.1. The table demonstrates the generic structure of the proposed approach. The top columns A to G of the table display the development steps addressed by the selected UML and IEC 61499 methodologies from the literature. The subsequent rows outline the existing methodology at each development step as well as the tools used. The last two rows describe the names of the selected development steps, their mapping interpretation and rationale.

6.2.1 Common development steps

As shown in Table 6.1, the unshaded columns A, B and D are the common development steps that are used by the existing methodologies. The requirements definition step in column A appears in most existing methodologies which is the first step to capture the requirements from users. The detail design step in column B outlines the requirement to structure the components into architecture. The design representation step in column D appears in most UML-based methodologies but not in IEC 61499-based methodologies. This is because XML is already defined in IEC 61499 as the design representation format. Some of the methodologies also include style sheets as part of the design representation.

6.2.2 Other development steps

As shown in Table 6.1, the shaded columns C, E, F and G are other development steps proposed for the new conceptual methodologies. The design repository step in column C was highlight by Cengic et al [2006] and Thamboulidis et al [2007]. This step is important as most systems do not start totally from scratch. The design repository step assumes that the reusable components can be stored and retrieved for future used. The code generation step in column E was mainly highlighted in UML-based methodologies but not IEC 61499-based methodologies. This step is necessary for generating code for various embedded applications. Most existing methodologies do not discuss

the implementation and the target application in much detail. For the target language and hardware platform, it is highly likely that proprietary tools, like Java compiler, SystemC compiler, are used. As such, it is necessary to have both implementation and target application steps for the new methodologies.

In summary, the table defines the grouping of the new development steps proposed in this thesis on the basis of the common steps extracted from selected methodologies and other development steps that are necessary for embedded applications. Further details of the potential proof-of-concept tool are discussed later in Chapter 7. The aim of the next section is to present the conceptual system development methodology for embedded applications.

Table 6-1: Mapping selected methodologies against common steps

Methodology	Development steps						
	A	B	C	D	E	F	G
Rose RT to SystemC [Tan et al, 2004]	UML Rose RT	UML Rose RT		XML	UML Rose RT to C++ then SystemC	Rose RT and proprietary tools for SystemC	
UML to HDL [Coyle & Thornton, 2005]		State Machine		XMI	UML State Machine Diagrams to HDL language	Proprietary tools for HDL	
UML state machines [Boudour & Kimour, 2006]		State Machine		XMI	UML State Machine Diagrams to SystemC	Proprietary tools for SystemC	
MDA-Based SOC [Wang et al, 2006]	UML-based requirement modelling	Design decision for hardware or software implementation			UML to SystemC	Proprietary tools for SystemC	
UML-Based Multiprocessor SoC Design Flow [Kangas et al, 2006]	UML-based requirement modelling	UML-based design flow		XML	UML to SystemC	Proprietary tools for SystemC	
StateChart to SystemC [Mura et al, 2007]		State Charts		XMI	State Charts to SystemC	Proprietary tools for SystemC	
Model-View-Controller (MVC) Framework [Brennan et al, 2004]	Model-View-Controller-based Function Block	Model-View-Controller-based Function Block				Java Compiler	Microprocessor
Automation components [Cengic et al, 2006]		Function block	Automation components			Java Compiler	
visualSTATE [Marian & Guo, 2006]	StateChart-based requirement modelling	visualSTATE design space			visualSTATE or FB code	Java Compiler	Microprocessor

Table 6-1: Mapping selected methodologies against common steps (continued)

Methodology	Development steps						
	A	B	C	D	E	F	G
Embedded Systems' Engineering Support Environment (eSESE) [Thamboulidis et al, 2007]	UML or IEC 61499-based requirement modelling	CORFU Tools	Application and system repository			Java Compiler	Microprocessor
Steps for conceptual methodology	Requirements definition	Detail design	Design repository	Design representation	Code generation	Implementation	Target application
Interpretation and rationale for mapping to system development methodology for embedded applications steps	The first step is to capture requirements that lay the foundation of the embedded application.	The detail design step outlines the requirements to structure the components into architecture.	The repository step assumes that the reusable components can be stored or retrieved.	Most of the design representation is using standard like XML. Some paper has mentioned style sheet but did not go into detail how they define the style sheet for target language.	Code generation step is used in some methodologies but the detail is not available.	Most of the papers does not discuss about the implementation detail. From the target language and hardware platform, it can be certain that proprietary tools are used.	The target application step assumes that the selection of the platform will also define the style of source code and compilation tools required.

6.3 Conceptual methodology

Figure 6-1 shows the block representation of the conceptual system development methodology for embedded applications which represents the preferred approach to an idea or thought for future research and development.

The following sections describe in more detail the individual steps of the methodology and the associated relevant future research issues, which is beyond the scope of this thesis but presented to ensure its thoroughness.

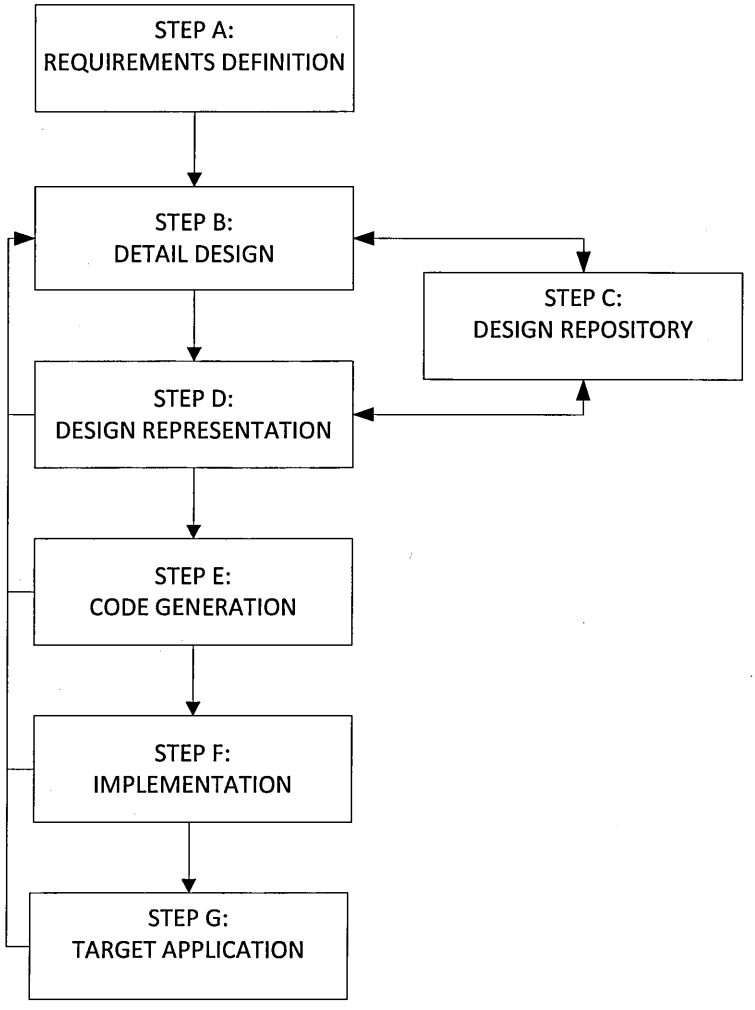


Figure 6-1: Proposed system development methodology for embedded applications

6.3.1 Requirements definition

As shown in Figure 6-1 step A, the requirements gathering step captures the user and system requirements. There are some desktop PC-based system development methodologies and tools available that can be used in this step. UML is one of the standards for capturing users' requirements, which is more often used for desktop PC software development rather than embedded application development [Graaf, et al 2003]. IEC 61499 is 'simple, cost effective, repeatable' and well suited to meet industry demands [Woll, 2007] that are explored in this research for embedded applications. The potential areas of future research in this step could address 'how to make the methodologies interoperable?', 'how to enhance the methodologies for both the hardware and software solutions?' and 'how to rapidly translate user requirements into application software?'.

6.3.2 Detail design

The detail design step aims to adapt the technical and application requirements into documents that represent mixture of software components, hardware components, reusable components structure and the system architecture. As shown in Figure 6-1 step B, the detail design step requires documents that represent the system and components. The components could be hardware-based, software-based or a combination of both, with each component having its own specific functions interface, control and attributes. These components then form the building blocks for the application. The potential future research issues that should be addressed include 'how to decide which components should be software-based, hardware-based or a hybrid of hardware and software?' and 'how components could be rapidly created and reused' and. 'how to determine the most suitable architecture and structure?'.

6.3.3 Design repository

This is a collection of style sheets, software templates, reference designs and implementations, either from previously developed systems or from other sources. As shown in Figure 6-1 step C, the functionality of the design repository step needs to be encapsulated into reusable blocks. Style sheets define the format of the target source code, e.g. VHDL, Java, C, C++, or HTML, languages and allow potential code generators to produce the source codes. The reference designs and template can be configured to the new applications, which accelerate the software development process. Interesting research issues at this step may include 'how to decide the most suitable reference design or templates to use', 'how to define the style sheets?' and 'how to make the style sheet interoperable with others open standards like XML and XSL?'.

6.3.4 Design representation

In this step, system designers are required to decide the best suited development approach, be it software-based, hardware-based or a hybrid of both hardware and software, for each part of the components. As shown in Figure 6-1 step D, the detail is saved into open standard representation like XML format. The style sheet will save the required target application language format. Both the design representation and style sheet can save into the design repository for future use. Other challenges that need further research are 'how to select the correct style sheet?', 'how to generate style sheet for various languages?' and 'how to represent the detail design in the open standard formation?'.

6.3.5 Code generation

In this step, both the design representation and style sheet will be input to a code generator. As shown in Figure 6-1 step E, the application code generator will be based on the requirements specification and style sheet from previous steps to generate the source code of high level languages for the target application. As shown in Table 5-1, different embedded system hardware platforms require different high level languages. One of the potential future research directions is to provide real-time reconfiguration of resource-constrained embedded systems. Other challenges deserving of further attention include 'how to build a real time partition of application space within the resource-constraint environment?', 'how to reconfigure the solution at run-time?' and 'how to split complex algorithms into parallel algorithms in order to speed up the process and reduce resource requirements' and 'how to measure rapid development?'.

6.3.6 Implementation

In step F, engineers are required to decide the best suited development tools and approach, be it software-based, hardware-based or a hybrid of both hardware and software, for each component. As shown in Table 5-1 different embedded system hardware platforms use different high level languages. Each embedded system hardware platform has its own proprietary compiler and synthesis tools. Other challenges that require further research are 'Is there an open standard compiler for multiple embedded hardware platforms?', 'how can the generated source code be optimised for a specific platform?' and 'how can source code be converted back into detail design?'.

6.3.7 Target application

The last step, as shown in Figure 6-1 step G, is the target application. The potential target application could be one that runs on an FPGA-based, microprocessor-based or a hybrid of microprocessor and FPGA-based

embedded platform. Different platforms will have different tools and associated research challenges.

1. Reconfigurable hardware-based platform

   One potential target application could be a reconfigurable FPGA-based platform. In this step, the use of an FPGA, i.e. a reconfigurable processor that has numerous logic gates, is proposed. The FPGA gates can be configured into different circuits with parallel processing in order to accelerate the computation of complex algorithms. Over recent years, the cost associated with FPGAs has significantly reduced, coupled with an increased suitability for computationally intensive processing. Areas of further research may be identified as 'how a complex algorithm could rapidly be converted into a hardware-based parallel solution?', 'how to make rapid development methodologies for both FPGA and microprocessors-based embedded solutions interoperable?' and 'how real time reconfiguration can be done on an FPGA?'.

2. Microprocessor-based platform

   This solution is comparable to a PC-based solution except it is in a small embedded system with a microprocessor. It requires an operating system and a software application. The system could also have small real time databases. The shortcoming of this solution is that the processing speed depends on the speed of the microprocessor, which could be slower than that for PCs. Therefore, the potential research issues which should be addressed are 'how the solution could be reconfigured at run-time without shutting down the entire system?', 'how to rapidly convert complex algorithms?' and 'how an operating system could be reconfigured based on user requirements?'.

Target applications of embedded systems could be in the form of standalone or handheld devices, or perhaps added onto new or existing equipment. An embedded system can have various sensors, input, output and algorithms to capture and process raw signal data, while also communicating with other systems. Interesting areas of research can address 'how embedded solutions can be synchronised and made to communicate with other enterprise systems?' and 'how to rapidly convert a standalone application into a distributed embedded application?'.

6.4 Chapter summary

This chapter has presented a conceptual system development methodology for embedded applications which included microprocessor and FPGA-based systems using simple graphical representation. Potential future research challenges were highlighted at each step. The new proposed conceptual methodology provides a clearer direction and approach for future research.

The next chapter discusses the design and development of the proof-of-concept tool for the proposed embedded application development.

CHAPTER 7 DEVELOPMENT OF A PROOF-OF-CONCEPT TOOL

This chapter presents the third stage of the research which focuses on designing the system architecture and developing the proof-of-concept software tool for the methodology. The architecture was uniquely designed for extendibility and modularity while maintaining the structure and behaviour of the system. This stage also selects software components from existing sources and develops other unavailable software components in order to form the proof-of-concept tool that support the system development methodology.

This chapter is organised into five sections. Section 7.1 discusses the research method for this chapter, with Section 7.2 presenting the system architecture design for the proposed methodology. Section 7.3 introduces the software components and technologies adopted, with Section 7.4 presenting the implementation of the proof-of-concept tool. Finally, the conclusions of the chapter are presented in Section 7.5.

7.1 Method

This aim of this chapter is to fulfil Objective 3 of the research outlined previously in Section 4.2 and depicted again below in Figure 7-1, which is to develop the proof-of-concept tool for the system development methodology for embedded applications. To perform this task, the following questions need to be addressed:

1. How are the conceptual methodology steps mapped to the system architecture?
2. What software components and technologies are available to support the system architecture?
3. How is the proof-of-concept tool tested?
4. How are the system architecture and/or the proof-of-concept tool refined?

The methods required to address these questions include:

1. Proposal of the system architecture and mapping of the steps from the conceptual methodology to the system architecture
2. Selection of the software components and technologies available to build the proof-of-concept tool. Those components that are unavailable will be developed in order to bridge the gaps
3. Choice of the experiment to verify the proof-of-concept embedded application development environment
4. Redefinition of the system architecture and/or the proof-of-concept tool

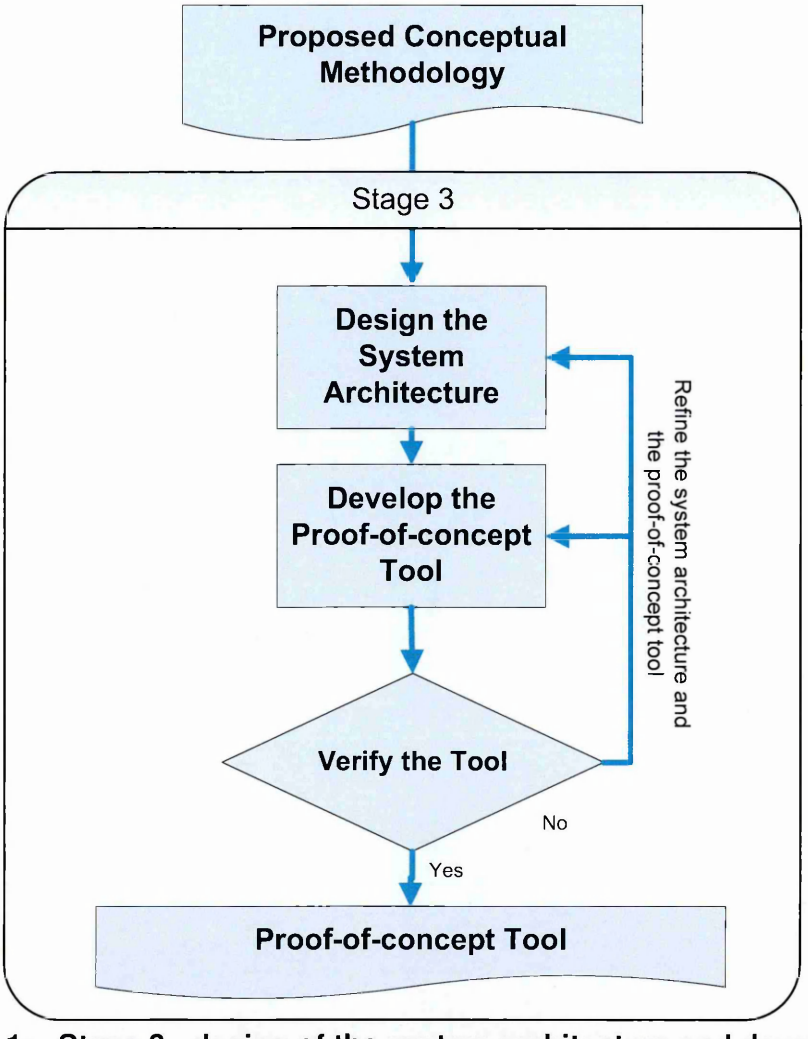


Figure 7-1: Stage 3 - design of the system architecture and development of the proof-of-concept tool

7.2 System architecture design

Figure 7-2 presents the system architecture design based on the conceptual methodology shown in Figure 6-1. The left column shows the steps from the conceptual methodologies and the right columns show the system architecture. The rectangular box in the architecture represents the process. The document is represented as a rectangle with a wavy base.

The requirements definition step aims to capture user requirements and system requirements through interview, discussion or brainstorming sessions with users. The output of this step is the user requirements and system requirements documentation.

The detail design step aims to adapt the user and system requirements into IEC 61499 Function Blocks. With the aid of the documentation, engineers can

structure the requirements into various Function Blocks using an IEC 61499 editor. Some of the design components can be retrieved from design repositories and newly developed design components can be stored in the repository for future use. The output of this step is detail function block design documentation in IEC 61499 format.

The design repository step is a collection of XSL style sheets definition, application templates, reference designs, existing Function Block components and external libraries from either newly built components, previously developed systems or other sources. The repository could be a repository database or file directory.

The design representation step aims to represent Function Blocks in an open standard format like XML. The output from the IEC 61499 editor is in XML format.

The code generation step aims to generate application source code from the XML representation of the detail design. The open standard XSL style sheet defines the target language format. Through the use of an XSL parser or code generator, Function Block designs can be translated into the target computer language according to the XSL style sheet definition. The output of this step is a high level computer language source code, such as open stand SystemC, Java or C++, for the target application platform.

The implementation step aims to use a proprietary application development tools to compile the source code into an executable application for the target platform. There are various commercial and readily available open source tools can convert high level SystemC code to Verilog for FPGAs.

The target application step could be running on FPGA-based embedded application platforms and microprocessor-based embedded application platforms, depending on the style sheet selected.

To verify the proof-of-concept tool, a simple light switch experiment is used, which generates Java code that can be run on PC or microprocessor-based embedded systems and SystemC code that can be converted for FPGA-based embedded systems. The following sections discuss the software components and the associated technology for the proof-of-concept tool in more detail.

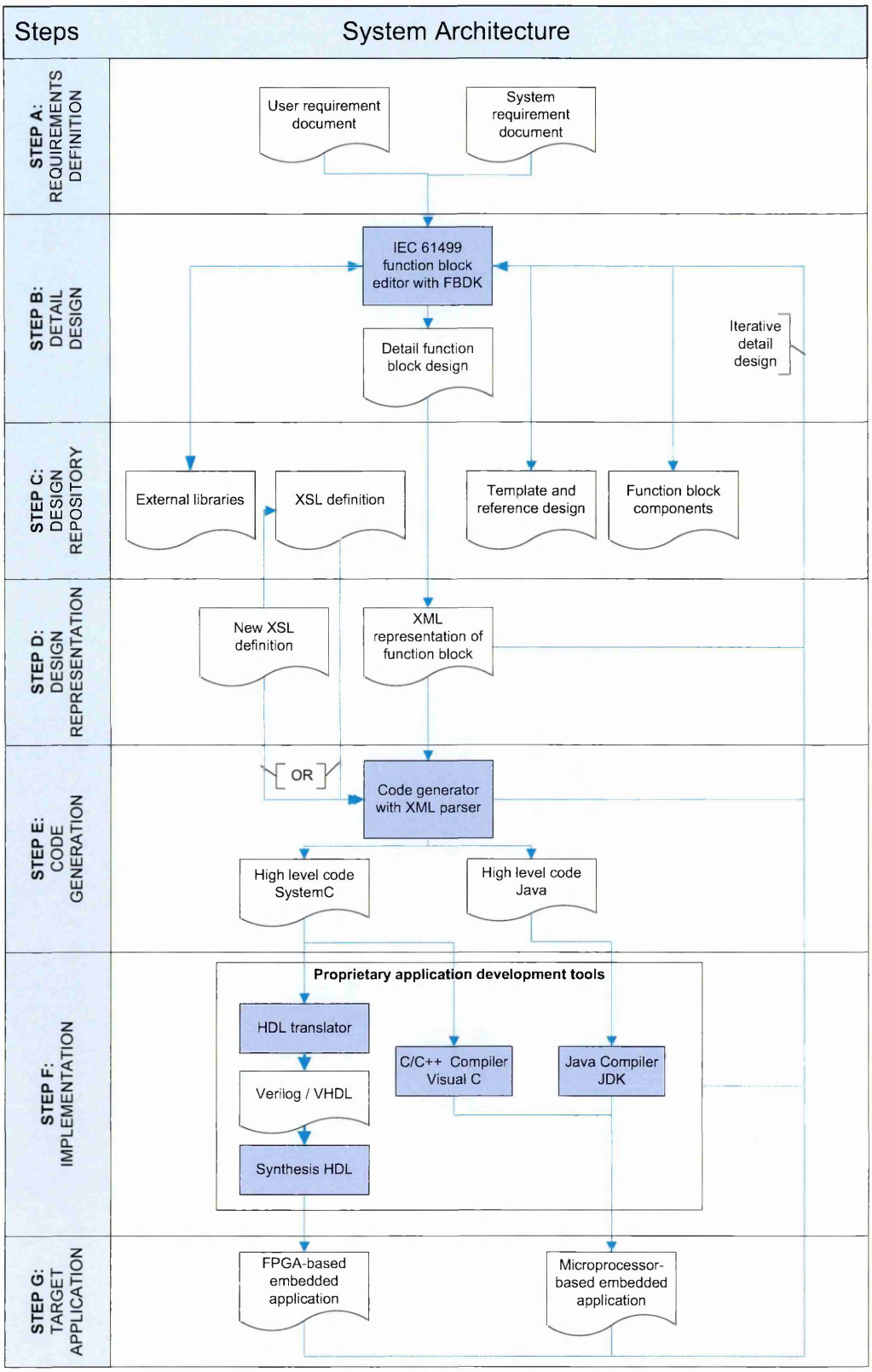


Figure 7-2: The proposed system architecture

7.3 Examination and development of the proof-of-concept tool

This section examines existing standard-based technologies, techniques and develops the proof-of-concept tool. The technologies identified include open standard IEC 61499 Function Blocks, intermediate language XML, style sheet XSL, XML parser and SystemC.

7.3.1 IEC 61499 Function Blocks:

International Electro Technical Commission (IEC) has defined various international standards for industrial automation. The IEC 61499 Function Block is an open standard component-based Function Block model that helps both the design functionality and integration capabilities of the system. Function blocks share many features with Unified Modelling Language (UML) objects, such as containing instances of other Function Blocks and using input or output variables and events. Data can be synchronised with events in Function Blocks, they can be used for capturing both current and future system requirements as is the case for the requirement step, with further uses as building blocks for applications in the detail design step. Industrial system specialists, who lack software training but are familiar with connecting physical devices together, can use the proven encapsulated Function Blocks and link them together in the most intuitive manner to quickly develop the system solution. Function Blocks can model a broad variety of solutions ranging from logic gates to large scale distributed systems. They are the functional units of software applications, with internal algorithms and data, which allow functions to operate on different platforms so that distributed control systems can be constructed. Basic Function Blocks can be created like software libraries which can be connected to larger Function Blocks for more complex problems. Function Blocks utilise an Execution Control Chart (ECC) state machine to trigger the execution of an algorithm. As shown in Figure 7-3, the algorithms inside the Function Block are hidden internally and are scheduled according to the ECC state machine. The ECC defines the Function Block behaviour according to input events, operations on internal variable values, input and output values. After successful algorithm execution, the execution control will generate output events accordingly.

The IEC 61499 standard defines various types of Function Blocks. A Function Block is a software module that can be stored in a library and repeatedly used for various differing applications. Every instance of a Function Block has a unique instance name and creates its own set of variables. The definition of the interface is performed first, follow by the definition of the block's functionality. The types of Function Block available include basic, composite and service interface. Their differences relate to the way their internal structure and functionality is described, although their interfaces are built using a common set of rules.

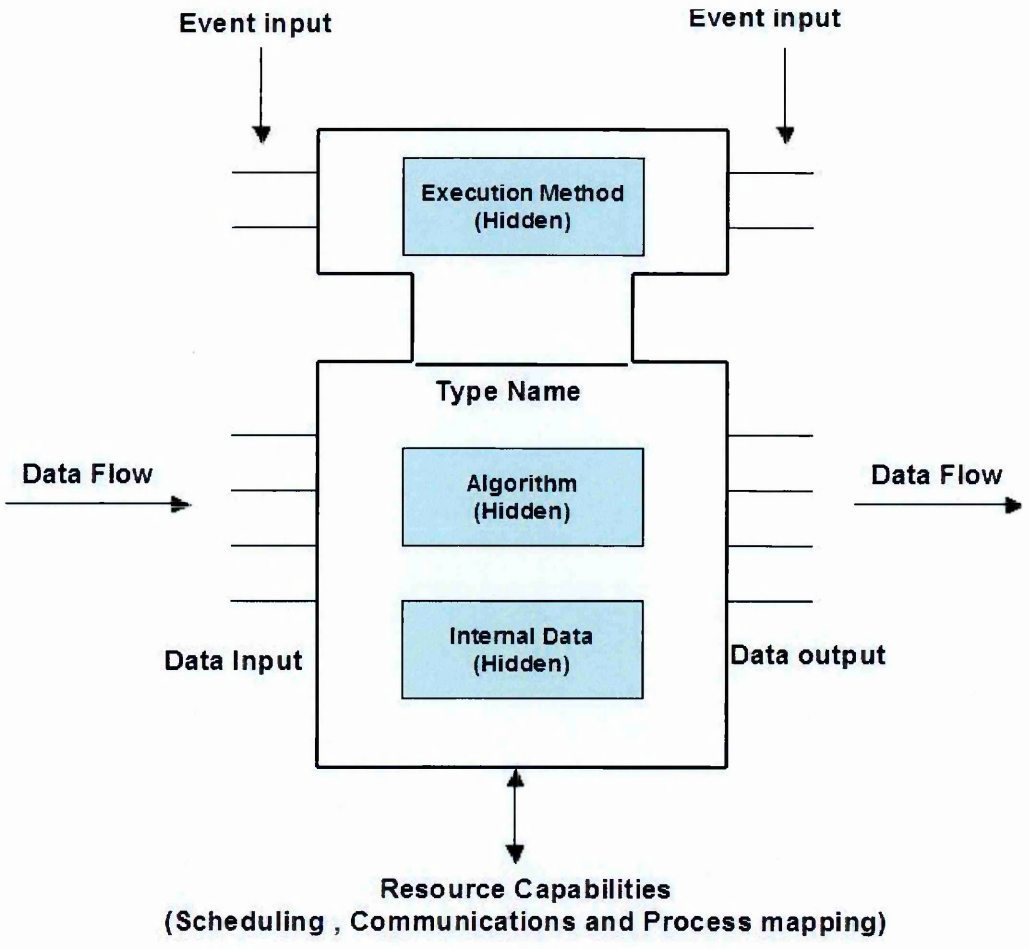


Figure 7-3: Execution model [Fischer & Boucher, 2008]

As mentioned in Section 5.4.1 sub-section 2, there is a selection of IEC 61499 development tools available for evaluation and testing. Freeware Function Block Development Kit (FBDK) [Holobloc, 2008] is the first IEC 61499 editor commonly used by researchers. The commercial tool from ISaGRAF [Weehuizen et al, 2007] is used for industrial deployment but does not fully meet the IEC 61499 standard as it does not save their files in XML and cannot interoperate with other IEC 61499 tools. Open source tools like FBench [FBench, 2008] and 4DIAC [4DIAC, 2008] are available for future research and enhancement. However, it has numerous bugs and is not very stable. Among IEC-compliant tools, FBDK [Holobloc, 2008] is the most stable freeware tool to test the methodology. FBDK can be used to generate XML codes and Graphical User Interfaces (GUIs). Through the use of an FBDK editor, the control system can be configured by adding properties such as Function Block name, identification, version and type definition.

1. BASIC FUNCTION BLOCKS

A basic Function Block is a platform-independent abstraction of a software component. The standard implies separation of the functions from the execution control and implementation through algorithms, as shown in Figure 7-4. The execution of the Function Blocks is event-driven, perhaps from other Function Blocks, meaning that the algorithms are executed only if an input event activates the block. They may encapsulate interfaces to the environment i.e. controlled process, communication networks, or hardware of a particular computational device. The execution function of a basic Function Block is defined in the form of a state machine that is available for documentation and specification purposes even if the source code of the algorithms remains hidden to the user.

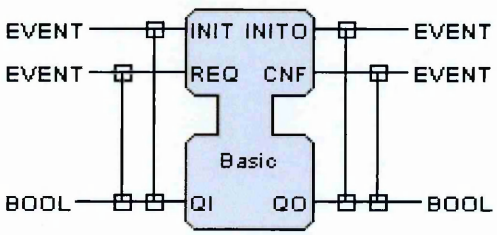


Figure 7-4: Basic Function Block type

The internal data of a Function Block can be used only by the internal algorithms and is inaccessible by other Function Blocks. Furthermore, they can be designed without any knowledge of the particular hardware on which the algorithms will later be executed.

The execution control method refers to the algorithm that is called after a certain input event. It is specified by using the execution control chart (ECC), with the basic block only having a single ECC, as shown in Figure 7-5.

An ECC is a finite state machine with a defined initial state. It consists of states with associated actions and of state transactions indicated by arrows in Figure 7-5. The actions contain algorithms to invoke and output events which are to be issued upon the completion of the algorithm. The execution of the block is initiated by the event input REQ (Request). The Function Blocks in the algorithm REQ are executed sequentially in the order of their occurrence in the textual representation of the algorithm. The initial state of ECC is START. The INIT (initialisation) event will transfer the EC (Execution Control) state to the INIT state and then the INIT algorithm will be executed. Upon completion of the algorithm, the event output INITO (initialisation output) will be generated. Condition 1 then resets the INIT state back to the START state.

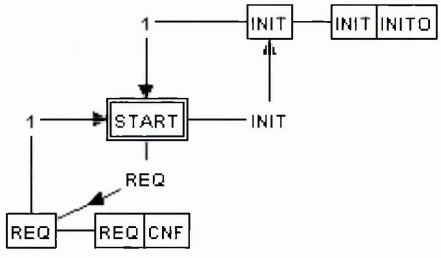


Figure 7-5: Execution control chart

2. COMPOSITE FUNCTION BLOCKS

A composite Function Block is a container that integrates other Function Blocks. The container as such does not perform any specific action except for setting input and output variables for the activities of its components. The network can include basic, service interface and composite Function Block types, allowing hierarchically organised applications to be built. A composite Function Block does not have an ECC and interface variables. Its functionality depends on the state of the basic Function Blocks as well as their events and data interconnections. Figure 7-6 and Figure 7-7 show the interface and an implementation example of the composite Function Block type respectively. In Figure 7-7, the start and stop switches of lamp Function Blocks are used to emulate inputs and outputs, with the control logic being represented by FB_AND and FB_NOT.

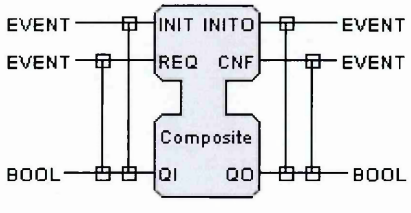


Figure 7-6: Composite Function Block interface

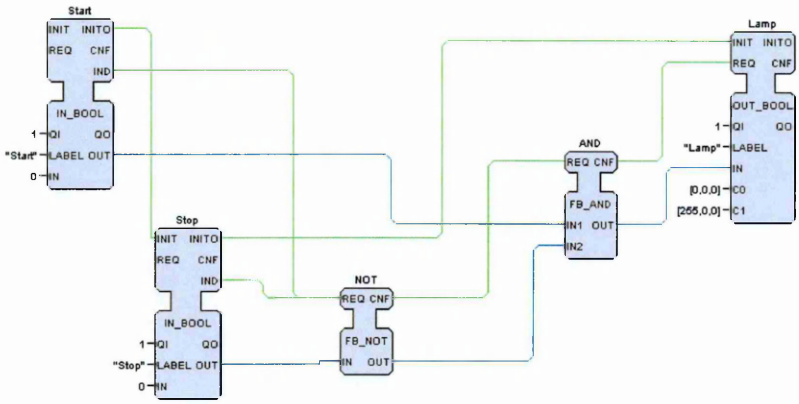


Figure 7-7: Composite Function Block implementation (graphical construction)

a. Connections in composite Function Blocks

Members of the Function Block network within a composite Function Block can be linked to the inputs and outputs of the "container block" through event and data connections. Typically, sets of rules and conventions exist for managing event and data connections.

b. Textual representation of composite Function Blocks

By using either textual or graphical declarations, it is possible to define instances i.e. copies of basic Function Blocks, created from the basic Function Block definition and copies of the composite Function Blocks, created from the composite Function Block definition. The behaviour of a composite Function Block type is determined by its type definition. However, it has its own storage for the associated input, output and events variables as well as for storing the state of the ECC. For example, a composite Function Block for one NOT and AND Function Block can be represented by the textual description as shown in Figure 7-8 below.

```
1.    FUNCTION_BLOCK ON_OFF_OC (* Composite Function Block Type *)
2.    EVENT_INPUT
3.    REQ WITH IN1O,IN2C; (* Normal Execution Request *)
4.    END_EVENT
5.    EVENT_OUTPUT
6.    CNF WITH OUT; (* Execution Confirmation *)
7.    END_EVENT
8.    VAR_INPUT
9.    IN1O : BOOL;
10.   IN2C : BOOL;
11.   END_VAR
12.   VAR_OUTPUT
13.   OUT : BOOL;
14.   END_VAR
```

Figure 7-8: Textual description

```
15.     FBS
16.     NOT : FB_NOT;
17.     AND : FB_AND;
18.     Start : IN_BOOL(
19.     QI := 1,
20.     LABEL := "Start",
21.     IN := 0);
22.     Stop : IN_BOOL(
23.     QI := 1,
24.     LABEL := "Stop",
25.     IN := 0);
26.     Lamp : OUT_BOOL(
27.     QI := 1,
28.     LABEL := "Lamp",
29.     C0 := [0,0,0],
30.     C1 := [255,0,0]);
31.     END_FBS
32.     EVENT_CONNECTIONS
33.     NOT.CNF TO AND.REQ ;
34.     Start.INITO TO Stop.INIT ;
35.     Start.IND TO NOT.REQ ;
36.     Stop.INITO TO Lamp.INIT ;
37.     Stop.IND TO NOT.REQ ;
38.     AND.CNF TO Lamp.REQ ;
39.     END_CONNECTIONS
40.     DATA_CONNECTIONS
41.     NOT.OUT TO AND.IN2 ;
42.     Start.OUT TO AND.IN1 ;
43.     Stop.OUT TO NOT.IN ;
44.     AND.OUT TO Lamp.IN ;
45.     END_CONNECTIONS
46.     END_FUNCTION_BLOCK
```

Figure 7-8: Textual description (continued)

3. SERVICE INTERFACE FUNCTION BLOCKS

As shown in Figure 7-9, Service Interface Function Blocks (SIFB) implements the interface between an application and the specific functionality that is jointly provided by the hardware and system software of embedded devices. While the code of a SIFB can be internally hidden, it can still provide the capability of specifying and documenting its functionality in a graphical form of service sequences, thus defining the interaction between an application and an external resource or library. The IEC 61499 standard defines labels for input and output parameters, such as INIT represents initialisation, INITO represents confirmation of the initialisation and QI represents the input qualifier.

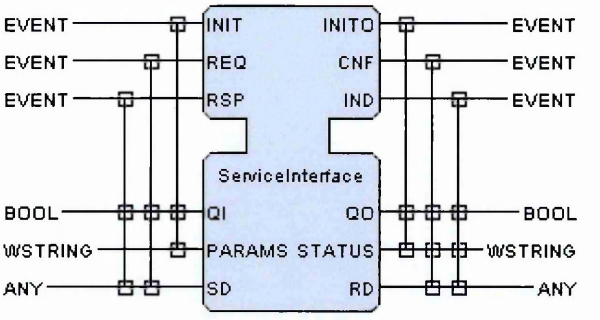


Figure 7-9: A Service Interface Function Block

7.3.2 eXtended Markup Language (XML):

XML is a flexible, open and simple text format originally designed for web publishing. XML is standardised by the World Wide Web Consortium (W3C) and was jointly developed by IBM, Microsoft and Sun Micro Systems. In this research, XML is used as the intermediate meta language to define the Function Block. XML is a widely accepted universal industrial data format standard for exchanging data. The XML format has flexible features which improve the ease of custom code development and enhance the use of code generators in application development through the use of a corresponding parser to convert the XML format into other textual formats.

7.3.3 eXtended Style Sheet Language (XSL):

XSL is the customisable interpreter for XML documents which describes the styles and formatting for XML documents as well as defining the display of the XML documents. By applying different XSLs to the same data, the layout of the formatted documents can be custom designed for a given purpose. XSL contains three components specifically for XML documents such as XML Path Language (XPath), XSL Transformation (XSLT) and XSL Formatting Objects (XSL-FO). XSLT includes template rules for transforming the XML documents. XPath is used to navigate and XSL-FO is used to format XML documents respectively. XSL formats the style of XML documents, which can then be used as a template code generator for a variety of languages, as shown in Figure 7-10.

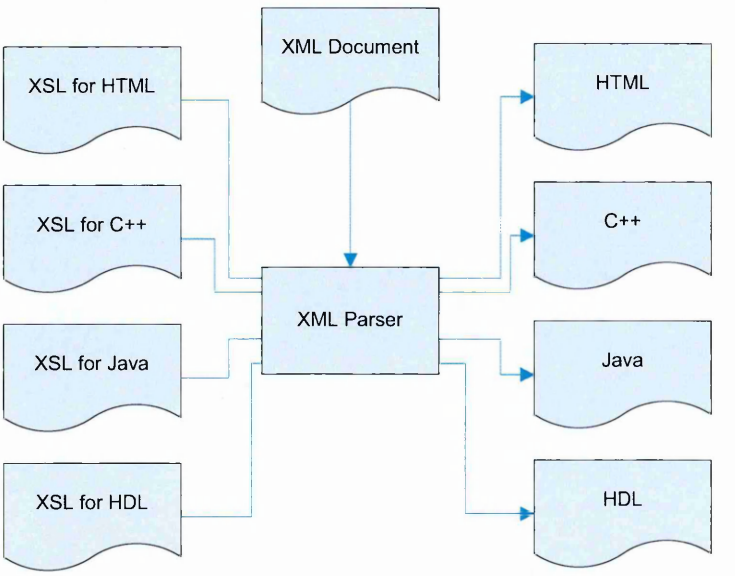


Figure 7-10: XSL for different language formats

XSL documents start with <?xml version="1.0" encoding=" "?> syntax which defines the version number. The next line, <xsl:stylesheet> indicates that this document is an XSL style sheet format. The 'match' attribute is used to define a template and to associate that template with an XML element. The value of the match attribute is defined by an XPath expression, for example: <xsl:template match="c:/data/device1">. The template is then tested against every node in the XML document. If a match is found then the content of the template is generated as output. Once matched, the child nodes will not be considered unless an explicit instruction is provided. The <xsl:value-of> element can be used to extract the value of an XML element and that value can be added to the output stream of the transformation.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet
type="text/xsl" href="transform.xsl"?>
<fb>
<port>
          <in name="porta"></in>
          <in name="portb"></in>
          <out name="portab"></out>
</port>
<algorithm
            expression="porta|portb"
            resultto="portab">
</algorithm>
</fb>
```

Figure 7-11: XML document for OR operation

Since XSL is used to manipulate the XML documents where information is stored, conditional tests against the expression can be done in XSL documents. There are several XML elements available to do the conditional tests. The element <xsl:if> is one of the elements that allows this. In order to use <xsl:if>, test conditions must be provided as the attribute of the <xsl:if> element. If the test condition is true then the next element inside the conditional test element will be executed. Multiple conditional tests can be done with <xsl:when> and <xsl:otherwise> elements together with the <xsl:choose> element. The <xsl:for-each> element is used to select every XML element between user defined tags as shown in Figure 7-12, which depicts the XSL template that helps the XML parser generate the C codes.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" />
<xsl:template match="fb">
void main()
{
     <xsl:for-each select="port/in">
     int <xsl:value-of select="@name"/>;
     </xsl:for-each>
     <xsl:for-each select="port/out">
     int <xsl:value-of select="@name"/>;
     </xsl:for-each>
     <xsl:value-of
select="algorithm/@resultto"/>=<xsl:value-of
select="algorithm/ @expression"/>;
}
</xsl:template>
</xsl:stylesheet>
```

Figure 7-12: XSL which converts XML to C

7.3.4 XML parser

An XML parser or code generator, available in numerous forms, can be used in conjunction with XML and XSL to generate target text file based on the XSL defined format. In this thesis, XSL defines the target source code formation for embedded applications and an XML parser converts Function Block detail representation in XML format to C++, Verilog, SystemC and Java languages based on XSL format definition. The parser loads the XML document into the computer's memory, with the most simple being the plain string parser which treats the XML document as a string content and returns element values by sensing the presence of < and /> within the string. Following the document being loaded into memory, the data can be manipulated using the Document Object Model (DOM). DOM is a hierarchical method of representing the elements in an XML document and then accessing them in a tree-like manner.

Simple API for XML (SAX) is an alternative to DOM. SAX opens innumerable possible uses for the XML document. Since each element, or node occurrence, is parsed and triggered as an event in the application, it can be interpreted in an application specific method thus providing a special representation. Some popular SAX and DOM parsers are Megginson (a Java-based SAX parser) [Megginson, 2008], Aelfred (a Java-based SAX parser) [Aelfred, 2008] and Microsoft's DOM parser [Microsoft, 2008]. Megginson, an industry standard API, is an open source application, which is used to access the XML document in Java. The latest release supports SAX version 2.0 and is available for various programming languages. Aelfred is a lightweight XML parser for Java and is known as a variation of Megginson's SAX parser. Microsoft XML parser is developed by Microsoft Corporation and is available for any programming language which supports the Microsoft ActiveX technologies.

7.3.5 SystemC

SystemC is an open standard system modelling language that supports hardware/software co-design and co-simulation using C++. It is used in both electronic and non-electronic systems designs. It can provide higher levels of abstraction for system components and exploit the advantages of C++ class library constructs through their timing, concurrency and reactive behaviours. Figure 7-13 shows how the IEC 61499 Function Block can be used as the system level design tool, as it generates different languages for various embedded environments. SystemC is used in this proposed approach instead of Verilog because SystemC allows for designs to be expressed and verified at high levels of abstraction and implemented in the lower level hardware. This approach helps system designers to design and simulate both the hardware and software behaviours of the system.

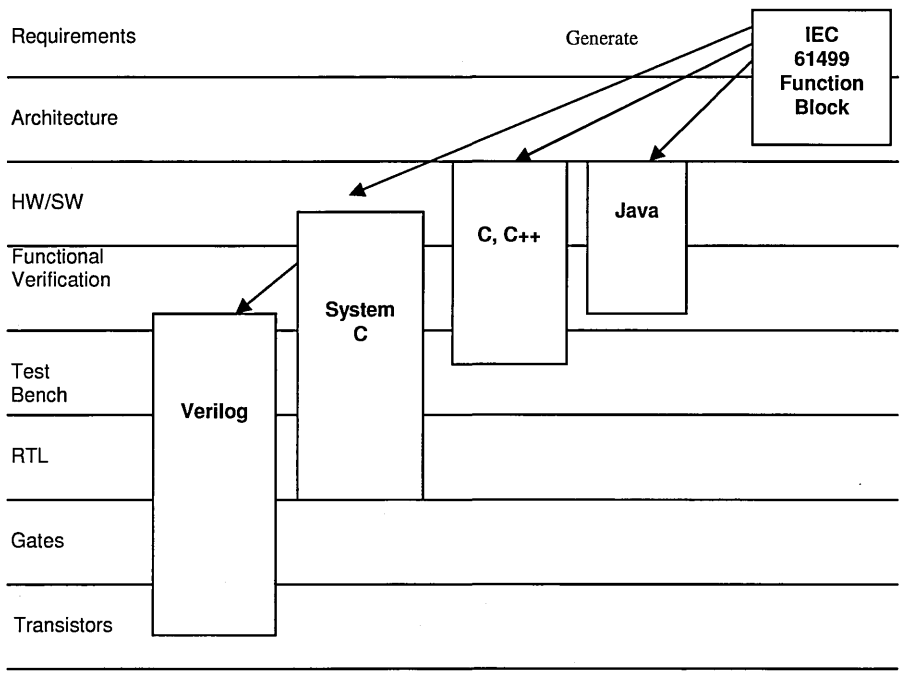


Figure 7-13: Language comparison and generation

7.3.6 SystemC to Verilog

SystemC is a hardware modelling language based on the C++ language. It provides the APIs and the simulation kernel to design and test electronic systems with high levels of abstraction. However, SystemC is a hardware modelling language and the tools to convert directly from SystemC into Electronic Data Interchange Format (EDIF) for FPGAs are still lacking. It is therefore necessary to initially convert SystemC into an intermediate hardware description language (HDL) such as Verilog, since numerous tools are available to convert Verilog into EDIF.

The SC2V [SC2V, 2008] converter is an open source software tool which translates the SystemC source codes into Verilog-equivalent ones, based on the Lexical Analyser Generator (LEX) and Yet Another Compiler-Compiler (YACC). LEX is a lexical processor for character input streams whereas YACC uses the code generated by the LEX and constructs the parse tree. SC2V then checks whether the input program is syntactically correct or not. Both LEX and YACC are the fundamental code generator components. There are some expensive commercial tools available to convert SystemC to Verilog such as Forte Cynthesiser [Forte, 2008], Synopsis CoCentric System Studio [Synopsis, 2008] and Agility Compiler [Agility, 2008].

7.4 Verification of the tools

This section presents an experiment to verify the proof-of-concept tool of the IEC 61499-based implementation, which will generate Java based solutions, SystemC code and subsequently convert that code into Verilog for an FPGA. For the purpose of testing the tools, the switch and light system [Fischer & Boucher, 2008] is used to model and simulate the IEC 61499 Function Blocks. An introduction to Function Block editor FBDK is provided in Appendix A and the proof-of-concept tool installation instructions as given in Appendix B.

7.4.1 Step A: Requirements definition

In this verification experiment, the switch and light system to test the fill proof-of-concept tool. The logic of the switch and light system is as follows:

$$Light1 = Switch1$$

$$Light2 = Switch1\ AND\ \overline{Switch2}$$

Switch 1 and Switch 2 are the inputs and Light 1 and Light 2 are the outputs. In this system, Light 1 and Light 2 will be ON when Switch 1 is turned ON while Switch 2 is OFF. Only Light 1 will be ON when both Switch 1 and Switch 2 are turned ON. Since switch 1 is the main switch, if Switch 1 is turned OFF, both Light 1 and Light 2 will be OFF.

7.4.2 Step B: Detail design

The system is modelled using a network of basic Function Blocks. The switch and light control system will be controlled by NOT and AND Function Blocks which are the main Function Blocks of the Function Block network. As depicted in Figure 7-13, the model has seven Function Blocks: START, (2 Switches) IN_BOOL, (2 Lights) OUT_BOOL, NOT and AND.

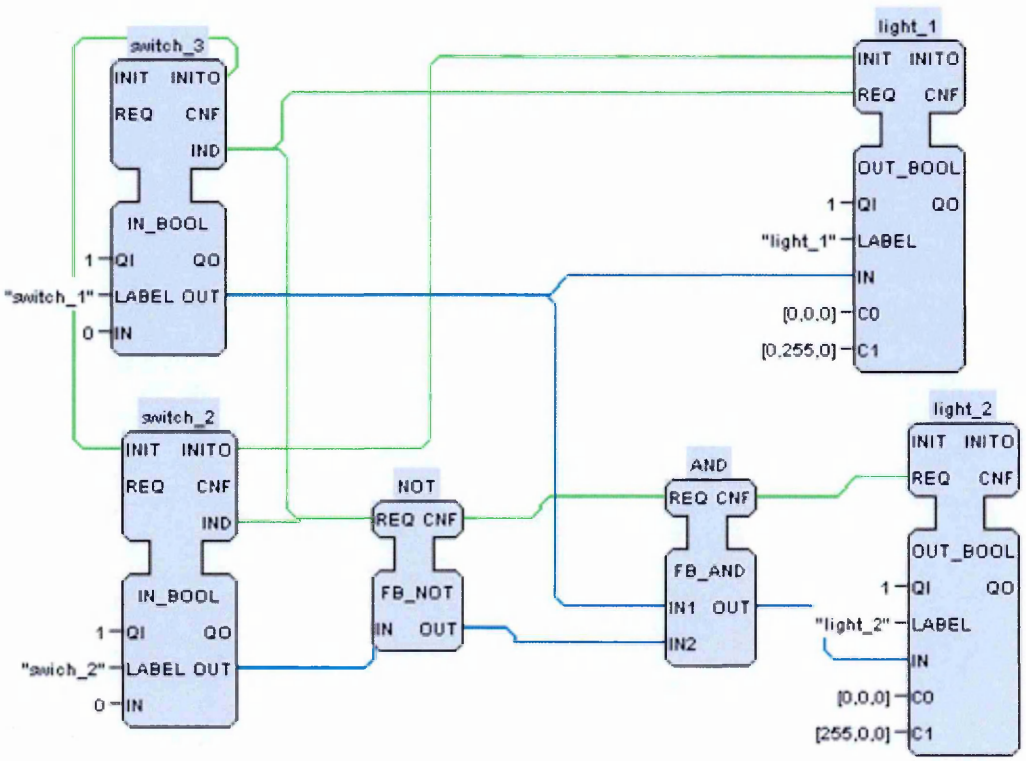


Figure 7-14: Modelling the switch and light system

1. START BLOCK:

   The START block, shown in Figure 7-14, is used to control the starting and stopping of the Function Blocks. The START block has three start events namely STOP, COLD and WARM. The COLD event starts the system and the WARM event restarts the system after it has already been started. The STOP event ends the system. Figure 7-14 illustrates the time sequence diagram for the COLD, WARM and STOP events of the START Function Block respectively.

2. SWITCH BLOCK:

   The SWITCH block shown in Figure 7-14 serves as the Boolean input Function Block, which may take on one of two Boolean states - ON or OFF. The Boolean input 0 or 1 is set by the value of the IN input where 0 corresponds to switch being OFF and 1 meaning the switch is ON. For this system, the initial state of the input is 0. As can be seen, there are two input events and three output events in the IN_BOOL block. In the switch and light system, two inputs INIT, REQ and two outputs INITO, IND are used. The INIT event initialises the input variable used by the Function Blocks while the INITO event generates output on successful initialisation. The REQ event reads the value from the IND

event. There are two switches in this system - Switch 1 and Switch 2, both having similar behaviours but potentially differing system-dependant event and data connections.

3. LIGHT BLOCK:

   The LIGHT block serves as an output Function Block, with OUT_BOOL being a composite Function Block. It is composed of two basic Function Blocks. This Function Block utilises an instance of the FB_SEL_COLOR and the OUT_COLOR to display a coloured circle with a label. According to the REQ event input, the colour is updated to the value specified by the C0 or C1 input depending on whether the value of the IN input is FALSE or TRUE, respectively.

4. FB_NOT FUNCTION BLOCK:

   The FB_NOT Function Block simply operates as the NOT logic operator that is TRUE if its operand is FALSE and is FALSE if its operand is TRUE. There is a single input, either true or false, into IN, while the OUT output will generate a single output as described.

5. FB_AND BLOCK:

   The FB_AND block operates as an AND logic that outputs a value of TRUE if both of its operands are TRUE and otherwise outputs FALSE. A single output, true or false, will be generated according to the input values of IN1 and IN2.

7.4.3 Step C: Design repository

Boolean logic from the repository, such as the FRAME_DEVICE, AND and NOT operators Function Block, is used to implement this system.

7.4.4 Step D: Design representation

The design of the switch and light control system is stored in a .sys file type and each Function Block is saved in .fbt file type. Both .sys and .fbt file types are in XML format.

7.4.5 Step E: Code generation

Most embedded application programmers are required to be fluent in numerous computer languages such as Java, C and C++ which typically requires extensive training. Since many embedded systems have product specific programming languages, code generation can provide a more convenient solution for engineers who are not familiar in the relevant programming language. Function Blocks can be used to 'draw' and 'link' the application, while the code generator will then generate the user selected source codes according

to the embedded system language requirements. Choosing the correct tool to perform the code generation is crucial to the completion of the task.

XML is the metafile for web and data representation with XSL being the style sheet that defines the resulting format of the XML. Software developers can create an XSL style sheet to define the required language template. As shown previously in Figure 7-10, an XML parser (e.g. Microsoft's MSXSL) can process the XML source and convert it to the target language based on the XSL. In this experiment, the Function Block design is converted into 2 different computer languages, namely Java and SystemC. Java is for microprocessors-based embedded systems and SystemC is for FPGA-based embedded systems.

7.4.6 Step F: Implementation

1. IEC 61499 Function Block

The FRAME_DEVICE basic Function Block type from the repository is used for the switch and light system with the associated bounds and grid parameters. Using a Java compiler, the Function Block can be compile into a Java application for a microprocessor based embedded system. The next step requires the mapping of the Function Blocks to the correct resources in the device. Resources and devices may reside in one or more systems.

2. SystemC with Visual C++

Using Microsoft VC++ or other compilers, the generated SystemC code can be compiled into an executable file. The compiler needs to link the SystemC header file (SystemC.h) and the library file (SystemC.lib) with the Run-Time Type Information (RTTI) enabled. RTTI provides the run time type information of the dynamic variables, which are necessary for systems that adopt dynamic variables with multiple inheritances and dynamic type casting as well as for the template classes. SystemC library files are heavily dependent on the RTTI. The executable file will show output in the console but will not be accurate as a real system due to the existence of an inherent signal delay in embedded systems. To solve this problem, tf=sc_create_vcd_trace_file ("sl_trace") needs to be added into the SystemC test bench to generate the Value Change Dump (VCD) files. These allow the output to be viewed in a waveform viewer (e.g. SystemC Win) for visual checking, as shown in Figure 7-15. A VCD file is an ASCII file which consists of header information, variable definitions and the value changes for all variables specified in the task calls. VCD is a standard format that is compatible with many waveform viewers and eliminates the need to know the internal logic of the system.

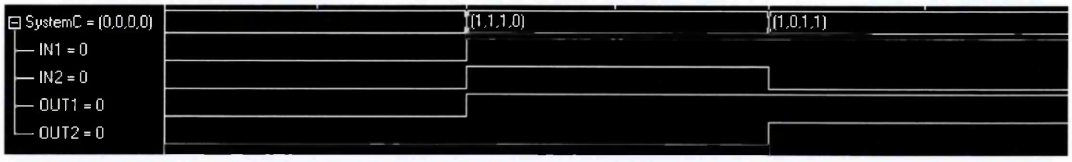


Figure 7-15: Simulated digital waveform

3. SystemC to Verilog

SystemC is a hardware modelling language and needs to be converted into an intermediate hardware description language (HDL) such as Verilog. The SC2V converter is used to translate the SystemC source code into Verilog. Sub-Section 7 in Appendix B shows the detail steps of how to use SC2V for conversion of SystemC to Verilog.

7.4.7 Step G: Target application

1. Microprocessor-based embedded application

The final Java application provides a graphical view of the system that could be run on a microprocessor-based embedded system. In this case, the experiment is running on a PC 104 embedded system as shown in Figure 7-16.

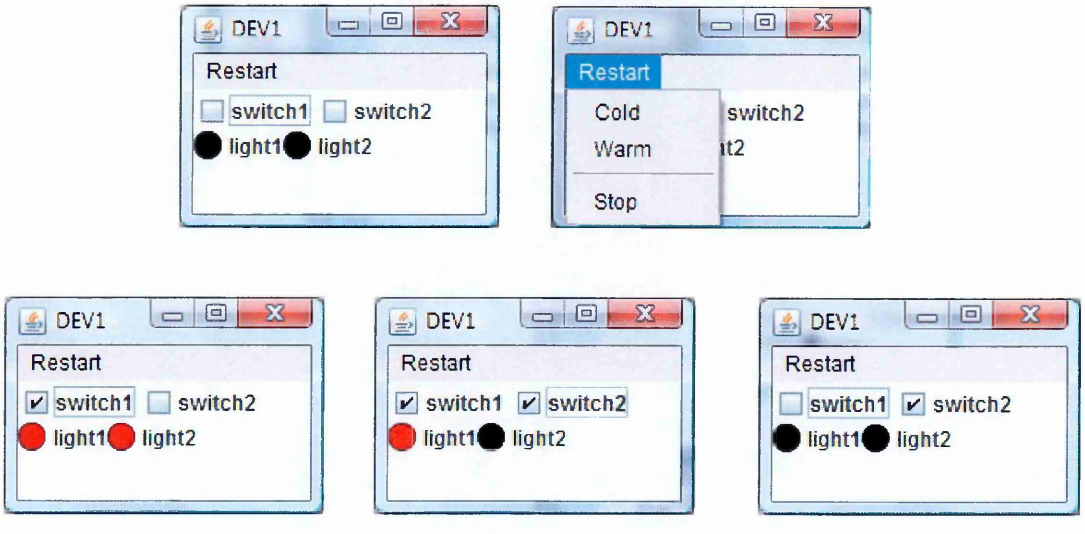


Figure 7-16: Testing switch and light system

2. FPGA-based embedded application

Once the Verilog file is generated, it can be readily used for configuration of a FPGA-based embedded system using the development environment from the FPGA vendors, e.g. Xilinx's ISE. The configuration process including placement, routing and BIT file configuration for the FPGA chip. Figure 7-17 shows the experimental result on a Xilinx's FPGA board.

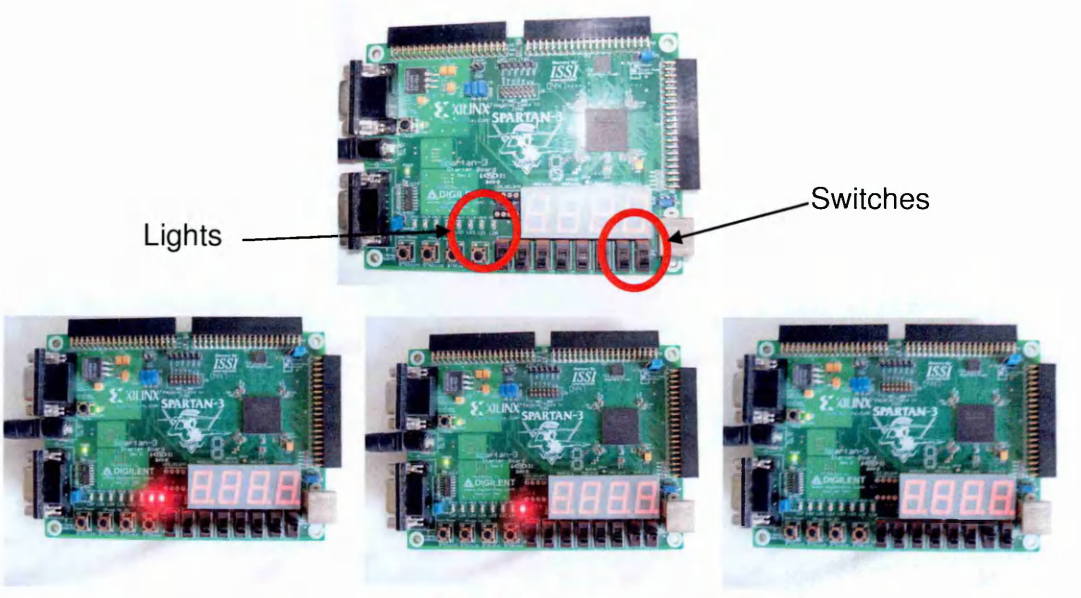


Figure 7-17: Switch and light system implemented on an FPGA

7.5 Discussion

The common evaluation criteria of a methodology, highlighted previously in Chapter 3, include testability, portability, component modelling, interoperability, understanding and reusability. The result of the evaluation is as follows.

- Testability: The verification experiment indicated that the proof-of-concept tool could generate a Java-based GUI and digital waveform simulations which allow developers and users to conduct different tests prior to implementation. This shows that the proof-of-concept tool for the methodology is readily testable through simulation.
- Portability: The simple light and switch experimental application can generate both Java and SystemC code that is independent of the implementation and can be deployed on both microprocessor-based and FPGA-based embedded devices.
- Component Modelling: The experimental applications use components-based Function Blocks. Some of the basic Function Blocks like FRAME_DEVICE, AND and NOT gate are used to construct the experimental application. Each Function Block can be interconnected with bigger composite Function Blocks or with different applications.

- Interoperability: The verification experiment demonstrated a light switch application that can interoperate in a microprocessor-based as well as an FPGA-based environment.
- Understanding: All experiments were built using graphical Function Blocks with simple lines representing connections. This is the simplest form of design and development and can be readily understood by most engineers. As highlighted by Graaf et al [2003], projects that employ diagrams, typically adopt free-form and box-line diagrams. Function Blocks resemble the easily understandable box-line diagrams. A half day workshop was conducted for industrial users to collect feedback on ease of understanding. A detailed discussion on the feedback is presented later in Section 8.7.
- Reusability: Reusability is clearly demonstrated in the experimental application through the reuse of the FRAME_DEVICE, AND and NOT gate basic function blocks to construct light switch. The XSL style sheet for this experiment can also be reused in other applications.

The proof-of-concept tool demonstrates the concept of the methodology with a verification experiment. The experiment generated Java code through standard Function Block editor and SystemC code through an XML parser. The final application can run on both microprocessor and FPGA-based embedded platforms. The verification experiment is executed in an environment where the researcher has full control over the requirements. The main purpose is to test the proof-of-concept tool and demonstrate how they function. The only drawback is that the XSL style sheet requires a lot of effort to create. The process of creating a style sheet requires an engineer with a strong background in software engineering especially in computer language structure. Once the style sheet is created, it can be reused in other applications. For the proof-of-concept tool, only the Gate level style sheet is created to prove the concept of the methodology. To create a full industrial strength style sheet for multiple languages would require an extensive engineering effort, rending it infeasible for the scope of work in the thesis.

7.6 Chapter summary

In this chapter, the system architecture of the proof-of-concept tool has been discussed. The methodology explores the use of the IEC 61499 standard for non-software specialists together with XML and XSL to carry out code generation. The methodology also includes an XSL-based style sheet definition for different computer languages, a code generator that generates different source codes and a SystemC to Verilog code converter for FPGA-based embedded systems. The methodology provides a flexible embedded software

development environment that could be used together with new or existing tools within the tool chain.

A simple test application of the switch and light system has been modelled in Function Blocks using the FBDK tool, which supports XML. In this case, the various language formats are defined by the XSL style sheet to generate source codes for both microprocessor-based (e.g. C++, C, Java) and FPGA-based (e.g. SystemC) systems. In order to implement the control system on an FPGA, the switch and light system must be in HDL format. This is because it is the only format capable of describing the circuit's operation, design and organisation, allowing for the testing and verification of the operation by means of simulation. Among the many available hardware description languages, Verilog is chosen since there are numerous available free and commercial tools that can readily synthesise and route to an FPGA. The Function Block can be seen to be more user-friendly for non-software specialists to design embedded systems, compared to learning different programming languages and manually coding the systems.

CHAPTER 8 EVALUATION OF THE METHODOLOGY

This chapter presents the fourth stage of the research highlighted previously in Chapter 4, which focuses on the evaluation and validation of the methodology through three industrial experiments and a workshop. This chapter is organised into 5 sections. Section 8.1 discusses the research method for this chapter. Section 8.2 gives an overview of the selected semiconductor manufacturing process, which is one of most widespread electronic industries in Singapore. To validate the proposed methodology, Section 8.3 presents 3 industrial experiments. The first experiment involves a methodology implementation for a polishing time estimator in the semiconductor manufacturing process. The second industrial experiment applies the proposed methodology to remote sensing, through the use of digital signal processing with compression. The development methods and experimental results are discussed in their respective sections. The third experiment presents the application of the methodology to a PID design with analysis and validation. Section 8.4 then presents the analysis result of the workshop. Section 8.5 analyses the strengths, weaknesses of the methodology and identifies opportunities for further refinement.

8.1 Method

The objective of this stage is to evaluate the effectiveness of the methodology with industrial participation. The research strategy is to observe the application of the methodology using the proof-of-concept tool in practice in order to evaluate whether it is practical, useful and relevant to the industrial participants. The evaluation will comprise industrial experiments and workshop feedback from participants.

According to Nunamaker et al [1991], a system development methodology cannot be verified mathematically and therefore must tested empirically through observation and experiments. The main advantage of experimentation and observation is the direct approach in which enables the researcher to collect firsthand data and interact with the experiment participants. Robson [2004] highlighted that experiments and observation prevent contamination of information between the researcher and the object of research.

In order to collect feedback from a wider range of industrial participants on the usefulness of the methodology, a workshop with hands-on demonstration is conducted. The workshop feedback is an additional data collection instrument used in this phase.

In this case, industrial experiments feedback is from observation and informal interview with participants whereas workshop feedback is at the discretion of the workshop attendees.

As shown in Figure 8-1, there are 4 steps comprising this fourth stage of the research which include:

1. To understand electronics related manufacturing process, select a suitable industrial sector and select suitable companies for both the experiments and workshop.

2. To understand the selected industrial related processes and validate the methodology using the proof-of-concept tool from stage 3 through industrial experimentation.

3. To conduct a workshop for a wider range of potential industrial users and capture their feedback through questionnaires for analysis and refinement.

4. To analyse cross experiments and workshop feedback from potential users to understand the strengths and weaknesses of the methodology and to identify opportunities for further refinement.

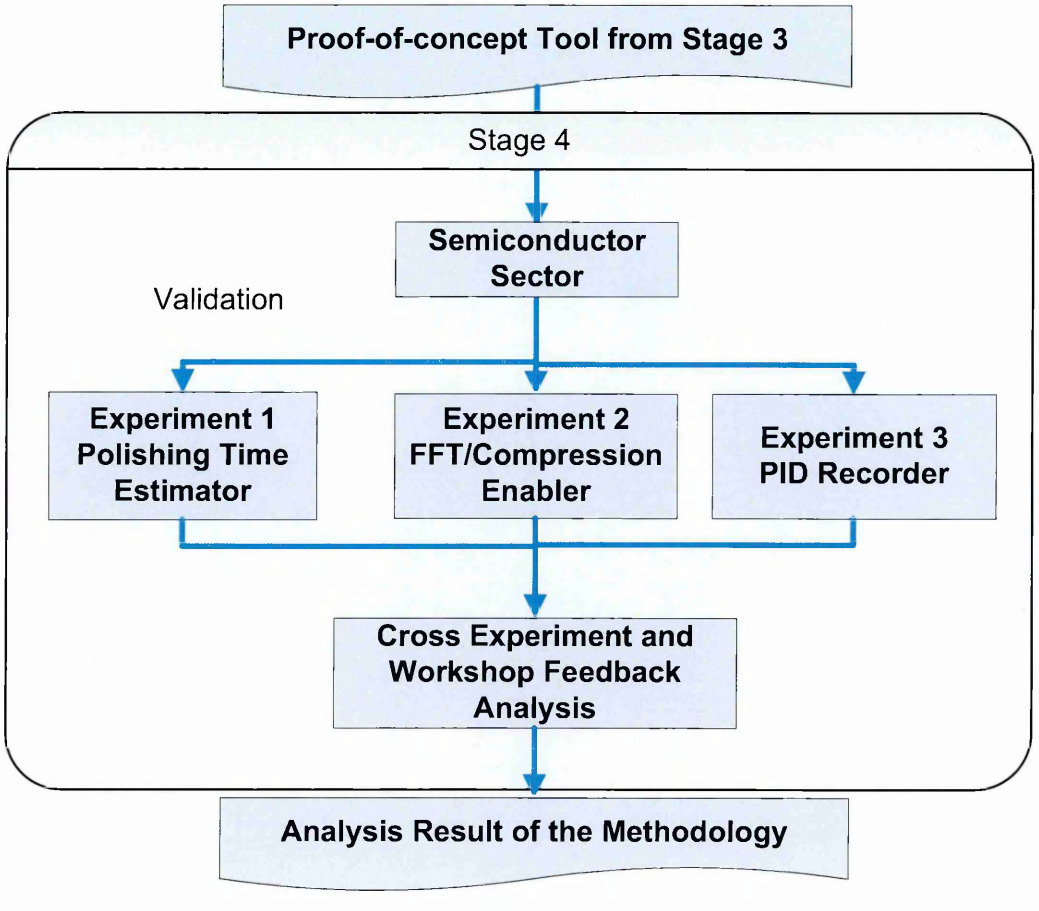


Figure 8-1: Stage 4 – Evaluation of the Methodology

8.2 Selection of companies

This aim understand electronics the general manufacturing process and to select suitable industrial companies for both the experiments and workshop.

8.2.1 Understanding the manufacturing process

An electronics cluster includes semiconductor production, computer peripherals, infocomms and consumer electronics. The largest segment within the Singaporean domestic electronics industry is semiconductor production [MAS, 2007]. As such, the industrial experiments will be selected from the semiconductor sector.

As shown in Figure 8-2, semiconductor production facilities can be divided widely into wafer fabrication, front-end and back-end semiconductor fabrication, assembly, mark and test. A brief description of production facilities and their functions are provided next.

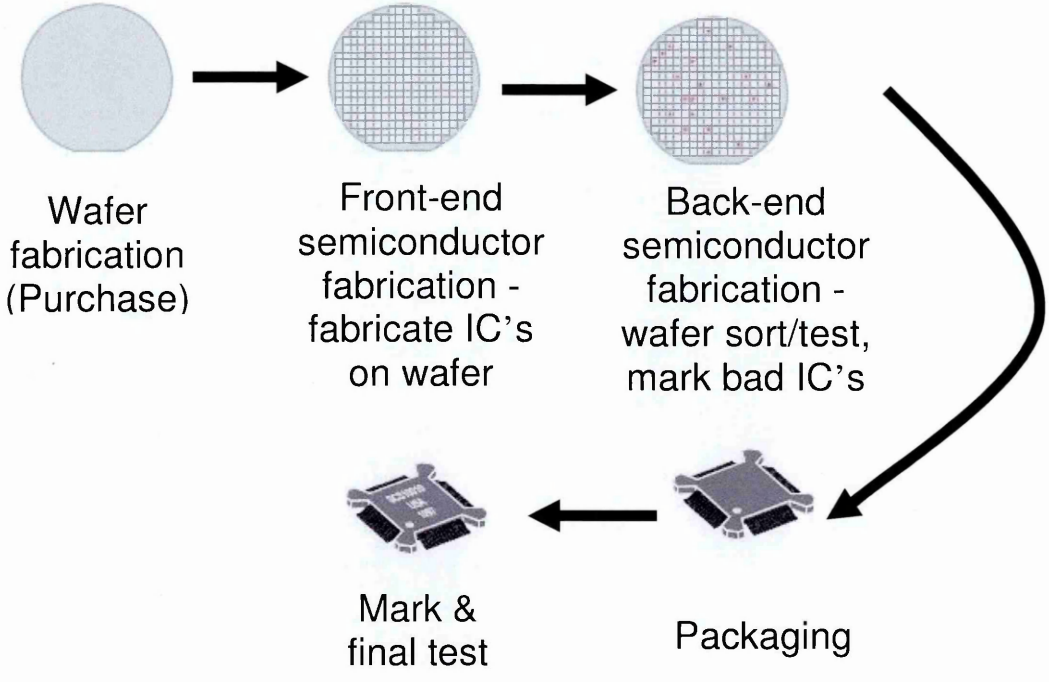


Figure 8-2: Semiconductor production overview

1. Wafer fabrication – An electronic grade wafer is manufactured from pure silica. A silicon ingot is pulled from a furnace and is sliced subsequently into fine thin silicon disks. Each silicon disk is then polished according to the specifications of the front-end semiconductor fabrication process. Wafer fabrication produces raw wafers of various sizes and electronic grades, depending on the process requirements determined by the components to be built on the wafer.

2. Front-end semiconductor fabrication – During the design at thousands of circuits on the wafer, the electronic grade silicon is subjected to numerous electronic processes. The processes could require up to 700 individual steps, including processing and measurement steps. Each product manufactured during fabrication goes through a specific routine. Once the wafer is complete, it is shipped to a back-end semiconductor fabrication facility.
3. Back-end semiconductor fabrication – Back-end semiconductor fabrication measures and evaluates the quality of each of the circuits (called a Die) on the wafer. A considered 'bad' Die will be marked and after the measurement process, be cut out from the wafer. The remaining 'good' Dies are then placed in a plastic encapsulation with legs attached to it for connecting to external circuits.
4. Packaging – The process involves the manufactured integrated chip being put on to a module. The assembly process will assemble the Die into a package, a process which includes Die bonding, wire-bonding and moulding.
5. Mark and test – The final process tests and evaluates the functionalities of each package and marks the package with the final product information.

8.2.2 Selection criteria

To guide the selection of industrial companies for the experiments, the following criteria have been set:

1. The companies should be in the related or selected industrial sector
2. The companies should provide a engineer to evaluate the methodology
3. The companies should have no methodology in place and should be exploring a new approach.

Once the criteria are determined, the next step is to find companies' contact information from SIMTech's contact list. Eight companies were invited because they are in the semiconductor sector. Three companies responded positively to the invitation. The remainder were not selected because they could not commit resources for the evaluation. The next step was follow up those companies with formal meetings, presentations and discussion on the project scope, time-scale and roles.

For confidentiality reasons and to allow a greater freedom to discuss the results of the evaluation, the 3 companies are referred to as CMP-Co, FFT-Co, PID-Co hereafter. A short description of the companies is given in the introduction section of the experiments.

8.2.3 Selection of industrial companies for workshop

In order to collect feedback from a wider range of potential industrial users, a workshop with hands-on demonstrations was conducted. The workshop invitations were sent to 50 contacts within the electronics cluster. The next step involved the preparation of hands-on demonstration slides, setup and questionnaires. A total of 35 participants signed up but only 28 participants joined the workshop. There were 19 participants from 11 companies dealing with electronics consumer product manufacturing, semiconductor chip manufacturing and engineering services provision. The other 9 participants were from research Institutes and Universities.

The next section describes the 3 industrial related experiments that validate the proposed methodology. The sections provide a background of the companies, their requirements, the processes, experimental implementations and results.

8.3 Industrial experiments

The industrial experiments were conducted at the participating companies' site. The industrial participants are external to the researcher's workplace but they were available for close interaction and discussion. Furthermore, the researcher was not directly involved in executing the experiments.

The researcher, however,

- Devised the assessment criteria with the industrial participants.
- Planned and assisted the participants with regards to the general procedures for running the experiments, which included:
    - Organisation of training and hands-on tutorials for industrial participants;
    - Provision of the proof-of-concept tool to the industrial participants;
    - Identification of an appropriate experimental design and application scenario;
    - Identification of the tasks to be performed by the industrial participants;
    - Running the required experimental trials according to the application scenario;
    - Consolidation of the feedback and observation for each experiments.

8.3.1 Experiment 1: Polishing time estimator

1. Background of the company

The CMP-Co is a multinational manufacturing company based in USA and with a regional headquarters based in Singapore. The company is one of the world leaders in PC manufacturing, semiconductor manufacturing, system integration and specialist solution provision for semiconductor manufacturing.

The automation division of the CMP-Co provides system integration services for both internal and external customers. Over the last few years, CMP-Co have experienced problems in hiring local Singaporean engineers and have thus sourced suitable candidates from overseas like India and China to support automation and system integration. Unfortunately, the job requires multi-disciplinary expertise including manufacturing, mechanical, electronics and software. Due to a high turnover rate and short time to deliver solution, CMP-Co finds that is difficult to retain the knowledge of engineers, hire the right engineers and train the engineer with the right skill.

The semiconductor production of CMP-Co occurs in an expensive clean room environment. The majority of companies require embedded systems as opposed to PCs to enhance their existing systems and control equipment. This is due to embedded systems being smaller in size and using fanles solid-state components, which do not generate dust. Due to rapid changes in semiconductor technology, the existing production equipment and systems always needs to be retrofitted and enhanced using embedded applications. CMP-Co aims to find a methodology that could be easily understood and used by non-programmers, to help in system integration for semiconductor manufacturing.

CMP-Co assigned a mechanical engineer to explore the proposed methodology for 3 months from October 2007 to December 2007. The engineer is trained in mechanical engineering and is not familiar with software programming. He provided the domain expertise, desensitised shop floor data and assistance in the development of a simple proof-of-concept embedded application using the methodology. To develop the polishing time estimator, the methodology was carried out in full and an understanding of the operation of each step within the methodology was obtained. As the methodology and proof-of-concept tool are designed to be self explanatory, it was found necessary to provide a brief overview so that the participating engineer understood the implementation details. The brief overview included some basic material, such as a tutorial on the Function Block editor as shown in Appendix A and proof-of-concept tool installation & instructions provided in Appendix B.

2. Step A: Requirements definition

To evaluate the methodology, the wafer production chemical mechanical planarisation (CMP) process was selected because CMP-Co had some problems in controlling the process yield. The CMP process is used in semiconductor production to planarise wafers which is the last step of the front-end semiconductor production process as shown in Figure 8-2. CMP is the process of flattening a wafer for providing a better depth of focus for upcoming layers. Due to the non-linear characteristics of the components used, CMP is a highly variable process. This leads to the need for constant re-adjustment of the control parameters so that the planarisation process could yield an on-target performance. The operation consists of three steps, pre-thickness measurement, the planarisation process and post-thickness measurement, as shown in Figure 8-3.

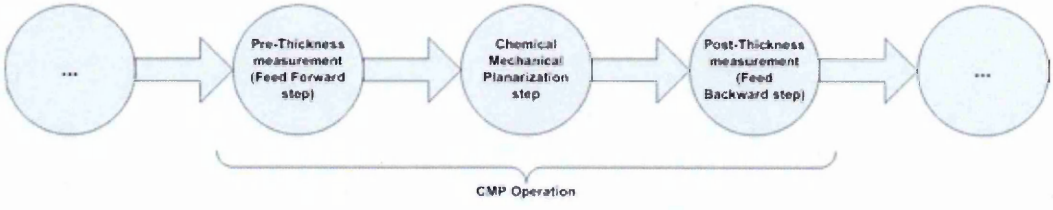


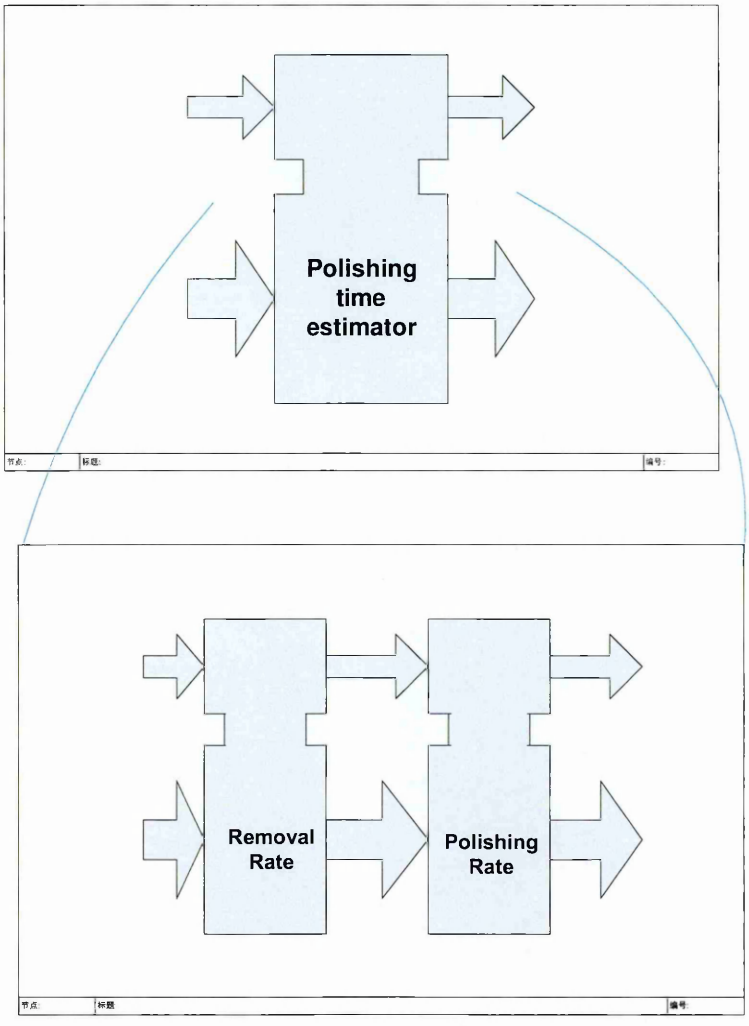
Figure 8-3: CMP operation (sequence of steps)

Through the use a pre/post measurements and the planarisation steps a feedback control loop is established. In this process, a polishing time estimator is used to estimate both the removal rate and polishing time. Figure 8-1 summaries the requirements definition step for the polishing time estimator.

Table 8-1: Polishing time estimator user requirements

a) Requirements gathering	
Goals	Polishing time estimation for the CMP operation.
Functions	**Removal rate calculation:** Estimate the rate at which a polisher is able to remove film or materials from a wafer. **Polishing Time estimator:** Estimate the duration for which a wafer is to be polished on the platen. Polish time is specified as a recipe parameter for each of the wafers.
Constraints	• Data must be accurate up to 4 decimals places. • Must be reusable functions. • Allow a built in bias (correction) number which will be defined by CMP-Co.
Resources	• Will provide desensitised actual polish time from history to verify and quantify the error in the estimation results. • Engineer from company. • Max 3 months with 30% of time to participate.
b) Analyse requirements	
Listing the functions	Removal rate calculation. Polishing time estimator.
Relations	Removal rate calculation result will be the input for the polishing time estimator.

Table 8-1: Polishing time estimator user requirements (continued)

c) Top-down sketch	
	
d) Review requirements	
Changes	No.
e) Hardware determination	
Hardware	PC 104-based fanles embedded hardware.

The engineer found that using a form was simpler and clearer. It allows non-engineers to participate and guide them to capture the correct and necessary information for discussion. The engineer highlighted that the current format

should have some key fields and CMP-Co may add more fields in the future to capture requirements.

3. Step B: Detail design

In the requirement definition step, the relationship between each function and the constraints are defined. The user requirements of the final application are sketched and the functions are linked into the structure of the application. This procedure is next described in detail.

Physical System

The requirement definition step identified that two processes, the removal rate calculation and polish time calculation, are necessary to construct the polishing time estimator. Each of these tasks corresponds to a set of functions which could be represented by Function Blocks. The physical system, as shown in Figure 8-5, gives a graphical representation of the final Polish Time Estimator module. The Polish Time Estimator begins with pre-polish thickness measurement data. The removal rate, calculated from previous cycles of the polishing operation, is used to compute the polish time for the current lot. The "Target Thickness" in Figure 8-4 represents the desired resulting thickness from the polishing process. The Polish Time Estimator module calculates the required polishing time for each lot, based on the previously calculated removal rate, pre-film and target thicknesses.

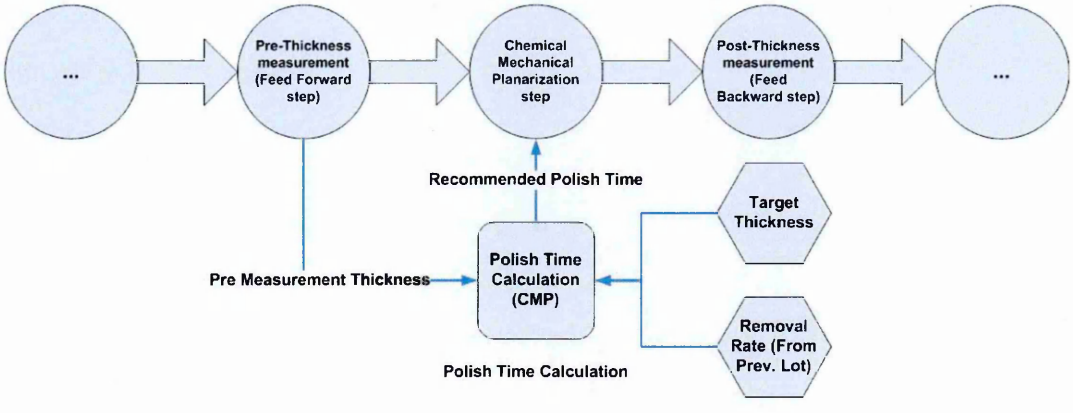


Figure 8-4: Physical system for the Polishing Time Estimator

System Sketch

Both the removal rate calculation and polish time calculation is provided by CMP-Co's engineer. Both an IEC 61499-based removal rate calculation and a polish time calculation module were developed for the Polishing Time Estimator, with module results being recorded for analysis. The polish time

result will be plotted against the actual data captured from the manufacturing line of the company. The system sketch is shown in Figure 8-6.

The removal rate and polish time calculation applications are implemented using Function Blocks. The development of the Function Blocks is an exact representation of Function Block programming.

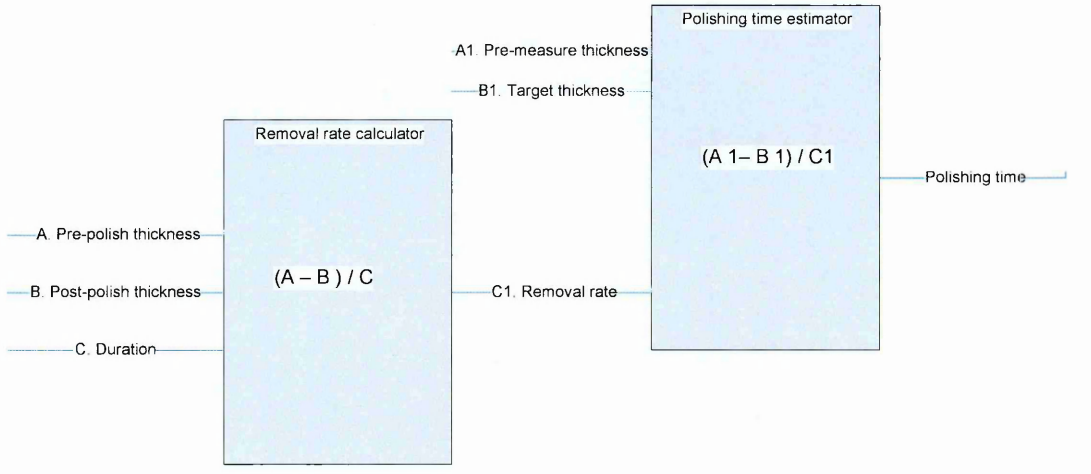


Figure 8-5: System sketch for the polishing time estimator

In this stage, the engineer found that it is easy to understand as it is manually draws the inputs of A, B and C into the first removal rate calculator box which produce an output C1. Inputs of A1, B1 and C1 will be the input to the polishing time estimator box which produces the estimated polishing time.

4. Step C: Design repository

There are some Function Blocks like DIV, enter text, text label are available from the IEC 61499 editor. These can be reused in both the removal rate calculation and polish time calculation.

5. Step D: Design representation

The detail design is saved into a .fbt file which is in XML formation.

6. Step E: Code generation step

As the CMP-Co wished to run the application in a PC-104 embedded system, they accepted the use of Java-based code for future development.

7. Step F: Implementation

This section details the implementation of the estimator. The implement details separately described the development methods for both the removal rate calculation and the polish time calculation.

Removal rate calculation

The removal rate is the rate at which a polisher is able to remove film or materials from a wafer. It is a function of the components used in the polishing process, which usually include the slurry used and polishing pads. Apart from the components themselves, the condition of the polishing components also has an impact on removal rate. Removal rates can be calculated using following equation:

$$R = \frac{T_2 - T_1}{D}$$

where

R = Removal rate (0.1 nanometre /min)

T_1 = Post polish thickness: Wafer thickness after the polishing step is complete. Wafer thickness is measured at the polishing step itself using an inline or post-polishing metrology step.

T_2 = Pre polish thickness: Wafer thickness before the polishing step is done. This thickness is measured at the pre-polish thickness step.

D = Duration of polish process: This is the time required or duration for which a wafer is polished on the polishing tool. This is recorded by the main polishing step and does not include the wafer handling and transfer time.

CMP advanced process control is achieved through a two step calculation. Firstly, the removal rate is calculated each time a full set of data (thickness before the planarisation process, the polishing time and the post-planarisation thickness) is available. The schematic of the removal rate calculation is shown below in Figure 8-6.

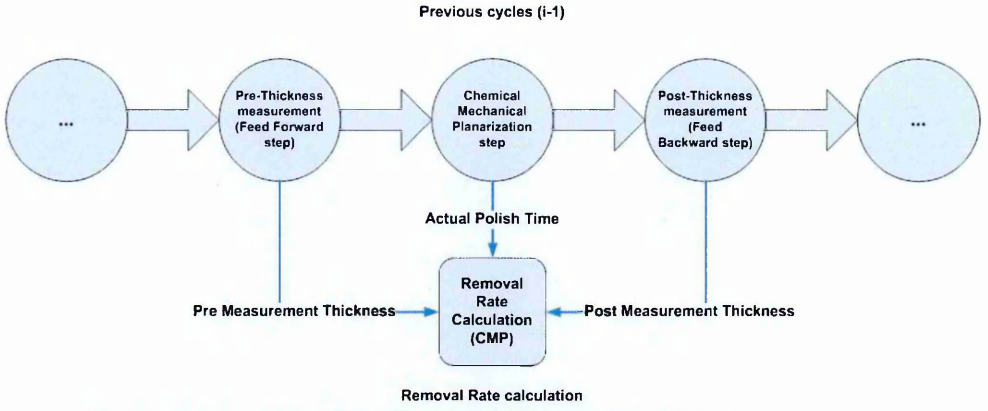


Figure 8-6: The removal rate control loop for previous lots

The module is created using the system development methodology proof-of-concept tool. Each block in the module is a Function Block which is then interconnected to form a network of Function Blocks. This network is a single standalone application, which will provide the required functionality for the removal rate calculation.

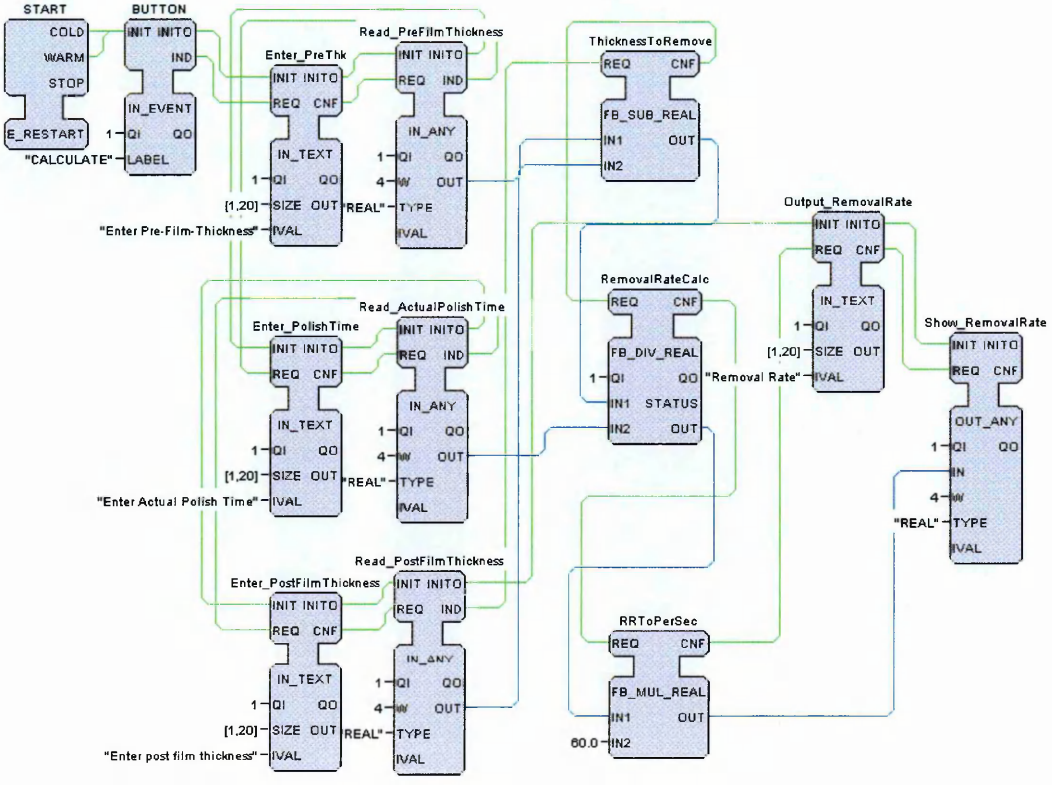


Figure 8-7: Function block implementation – removal rate

As shown in Figure 8-7, the START component is used to initialise the Function Block. It generates various events, which can be used for initialisation

purposes. In this application, COLD and WARM events are mapped to the INIT event of the BUTTON Function Block. Once the button is initialised, it sets the INITO output event on the button, which then activates the Enter_PreThick (IN_TEXT) Function Block.

Enter_PreThick is used only for displaying the text – "Enter Pre-Film-Thickness" as an indication to the user to enter the pre-film thickness in the right text field. Similarly, Enter_PolishTime, Enter_PostFilmThickness and Out_RemovalRate are used to display "Enter Actual Polish Time", "Enter post film thickness" and "Removal Rate" respectively. An IN_ANY Function Block displays an empty text field next to the labels (Enter_PreThick, Enter_PolishTime, Enter_PostFilmThickness and RemovalRate). Read_PreFilmThickess, Read_ActualPolishTime and Read_PostFilmThickness components use IN_ANY Function Block. The function of the IN_ANY Function Block is to allow the user enter the necessary parameters for the calculation.

ThicknessToRemove uses FB_SUB_REAL Function Block which is used to subtract two REAL values. ThicknessToRemove calculates the difference of pre-film thickness and post-film thickness. It gives the total film thickness to be removed. After calculation, it set the value at its output and then raises a CNF event.

RemovalRateCalc component adopts a FB_DIV_REAL Function Block which used to divide two REAL numbers. The result from the ThicknessToRemove component is divided by actual polish time from the Read_ActualPolishTime component to give the actual removal rate in minutes. Multiplying by 60 then gives the removal rate in seconds. The calculated value is displayed using the Show_RemovalRate component, which is an OUT_ANY component.

Polish time calculation

The polish time is the duration for which a wafer is to be polished on the platen. The thickness achieved as a result of the planarisation process is a function of the polishing time. The polish time is provided as an input parameter for each of the wafers on the platen. It is a common practice to introduce a bias to the polish time correction based on practical observations to compensate for assumptions or unconsidered aspects of the process. The polish bias or correction factor provides an offset to the polish time such that the target film thickness can be obtained. Incorporating this polish bias, the polish time equation is shown below.

$$P = \frac{T_2 - T_1}{R} + B$$

where

P = Polish time

T_1 = Post polish thickness

T_2 = Pre polish thickness

R = Removal Rate

B = Bias or correction factor: It is a common practice to introduce a bias factor to polish time correction based on the observation in real life operation.

The polish time is calculated using a set of feed forward (film thickness of wafers before planarisation), feedback (film thickness after the planarisation process) and process data (polish time used for the planarisation process). The following section describes how the polish time is calculated using the feed forward, feedback and process data. The schematic of the polish time calculation is shown below in Figure 8-8.

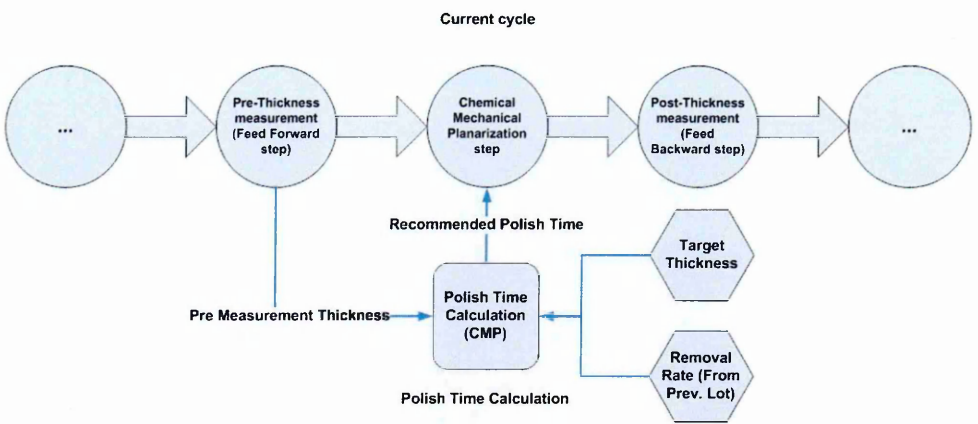


Figure 8-8: Polish time calculation (using the previously calculated removal rate)

The application is created using an IEC 61499-based editor. Like the previous application, each block in the module is a Function Block which is then interconnected to form a network of Function Blocks. This network is a single standalone module which will provide the required functionality for the Polish Time Calculation module.

As shown in Figure 8-9, the BUTTON provides an event driven mechanism for initiating calculations. When the BUTTON is pressed, it generates an IND out event which then invokes the IN_TEXT (Enter_PreThick) component which further drives the IN_ANY (Read_PreFilmThickness) component. The IND event from Read_PreFilmThickness drives the Enter_TargetThickness component which then calls Read_TargetThickness. The IND event from the

Read_TargetThickness Function Block then drives both the Enter_RemovalRate and Read_RemovalRate components.

The ThicknessToRemove Function Block takes its input from the Read_PreFilmThickness and Read_TargetThickness Function Blocks. The Function Block ThicknessToRemove is driven by an IND event from the Read_RemovalRate Function Block. The PolishTime Function Block takes its input from the ThicknessToRemove and Read_TargetThickness blocks. The PolishTime Function Block is driven by a CNF event from the ThicknessToRemove Function Block. The PolishTime is converted into seconds by multiplying the result from the ThicknessToRemove block by 60. The time in seconds is then offset by 0.5 seconds using the AddBias Function Block.

The result from AddBias is displayed on the window using the ShowPolishTime Function Block. The Enter_PolishTime Function Block is an IN_TEXT Function Block which is only provided to display the text – "Polish Time" – an indication to the user that the subsequent visual component represents an output from the diagram. Function Blocks such as the Enter_PreThick, Enter_TargetThickness, Enter_RemovalRate and Enter_PolishTime Function Blocks are only used to display labels on the output window are considered as non-functional (or implicit requirements) blocks. Thus, without these functions, the application will be considered incomplete. Figure 8-9 shows the Function Block implementation of the polish time calculation application.

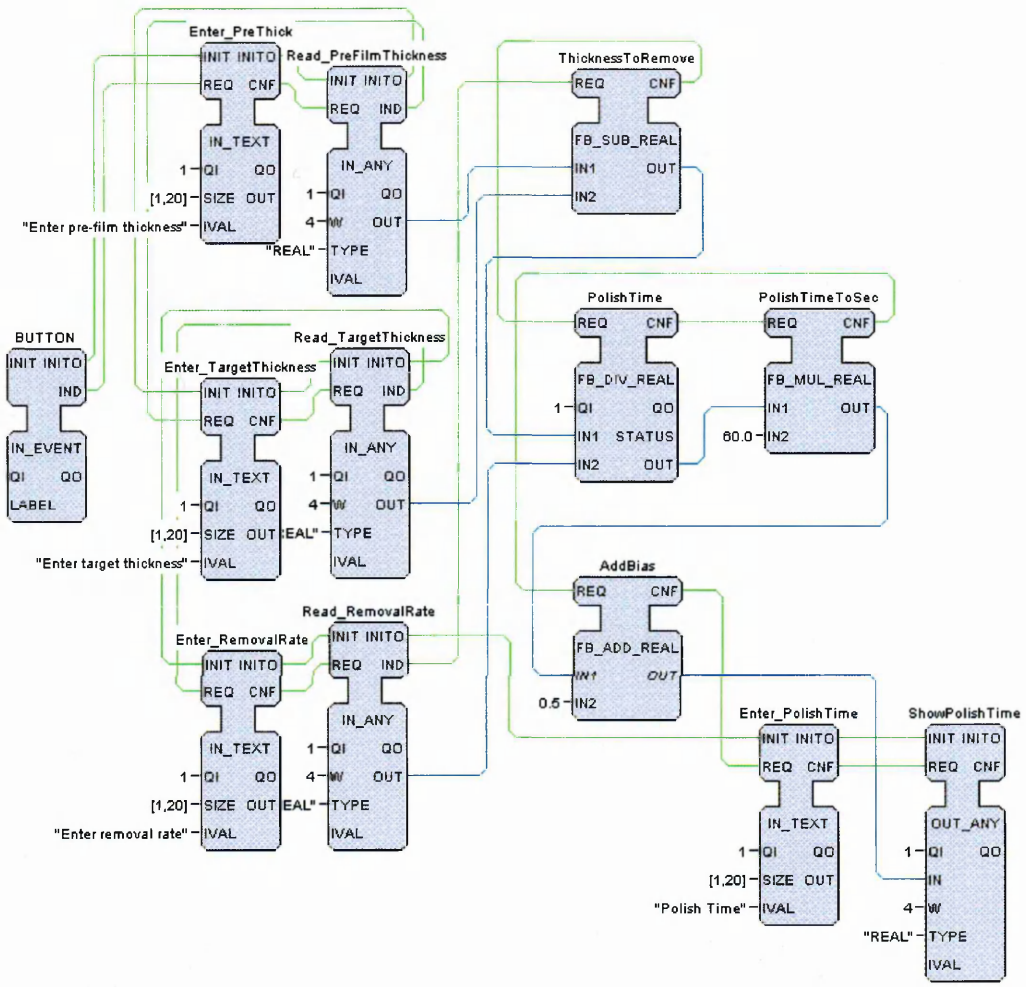


Figure 8-9: Function block implementation – polish time

8. Step G: Target application

This subsection details the numerical results and analysis for the removal rate and polish time calculations discussed previously.

Removal rate calculation

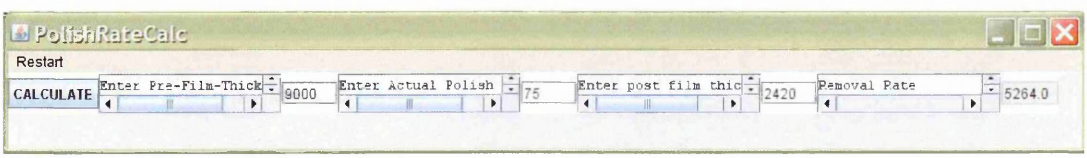


Figure 8-10: Removal rate calculation module

Figure 8-10 shows the Function Block application which is converted into a Java application. The window shows text boxes asking the user to enter data, with a text field provided to input the data. The “Removal Rate” field displays

the result from the calculation. Data from pre-measurement thickness, actual polishing time and post-film thickness is used to calculate the removal rate.

The calculator is repeatedly used by the engineer and compared with against the actual data from the production. The feed forward pre-film thickness, post polish film thickness and actual polish times are passed to the module in order to produce the estimated removal rate.

Polish time calculation

When the polish time calculation Function Block application is executed, the integrated environment displays the application window shown in Figure 8-11. The pre-film thickness, target thickness and removal rate values are entered into their respective fields and the "BUTTON" is pressed. The button press causes a sequence of algorithm executions in each of the Function Blocks producing the result in the "Polish Time" output field.

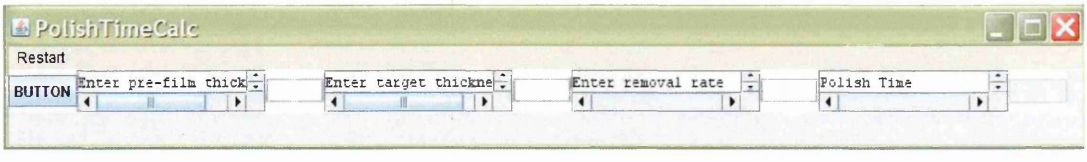


Figure 8-11: Polish time calculation module

The removal rate from the previous lot is used to calculate the polish time for each lot. In the case of no previous polish time being available, a default removal rate will be assumed.

CMP-Co has used this polish time calculation function for the comparison with their actual polish time estimation. The graph depicted in Figure 8-12 demonstrates the result of the polish time estimation. The measured value for polishing time is captured from the actual lot process, while the calculated value is obtained from the polish time calculation application. The calculated polishing time chart is within the same range (difference between maximum and minimum polishing times) as that of the measured values of polishing time.

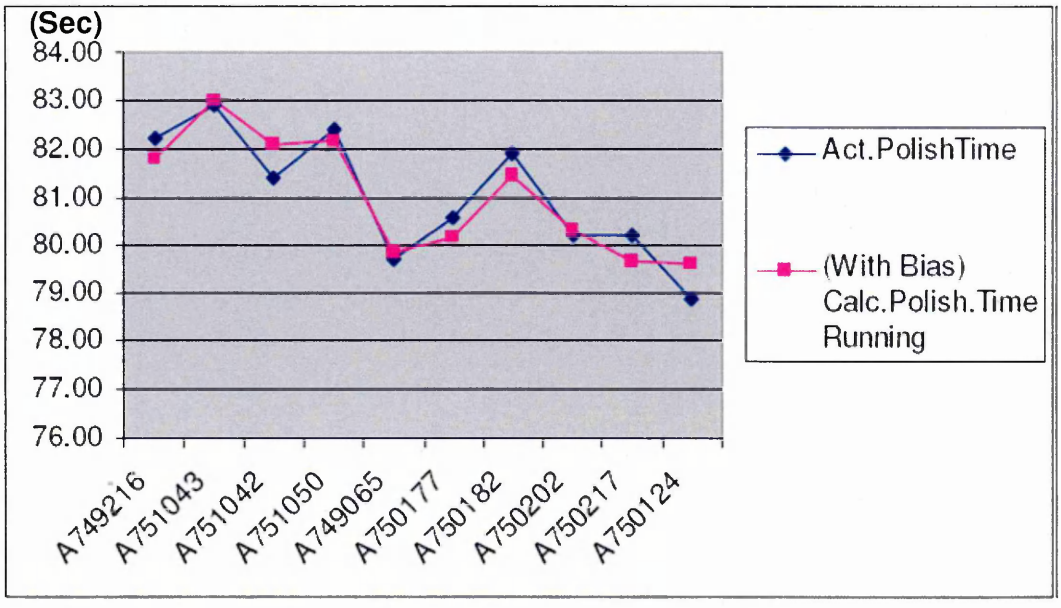


Figure 8-12: Actual vs. calculated polishing time with a bias of 0.5 sec.

9. Evaluation of the experiment

The engineer indicated that he is fully confident to develop the module after a short one hour discussion and briefing. The advantage of using this approach is that the engineer would be able to learn the methodology step-by-step along with the proof-of-concept tool. In addition, from a developer's point of view, it is easier to develop and test the IEC 61499-based module, which can be easily reused in other part of the automation project.

The common evaluation criteria of a methodology which are used for discussion after the experiment. The result of the evaluation is as follows:

- Testability: The engineer from CMP-Co indicated that he could conduct various tests with the estimator within their company. The Engineer found that the proof-of-concept tool could generate Java-based GUI simulations which allow developers and users to conduct different tests before implementation.
- Portability: In this experiment, only Java code was used as CMP-Co was more interested in the methodology and result instead of the programming language. The experiment executed the Java application on both PC and PC104 embedded systems without problem. This is due to the nature of Java, which is run on top of a virtual machine that is available from various operating systems.

- Component Modelling: The experimental applications used components-based Function Blocks. Some of the basic Function Blocks like FRAME_DEVICE, math and data entry were used to construct the experimental application. Each Function Block can be interconnected into bigger composite Function Block or with different applications. The removal rate calculation Function Block was used in the final polish time estimation.
- Interoperability: In this experiment, CMP-Co did not allow any access to the actual database or system. They preferred to develop the final application within the company for reasons of confidentiality. As such, it was not possible to test interoperability was not able to test in this experiment.
- Understanding: The experiments were built using graphical Function Blocks with simple lines representing connections. The engineer reported that the box and line were easy to use and could be readily understood by most engineers. The initial difficulty in using the tools was resolved during face-to-face demonstration and discussion.
- Reusability: Reusability was clearly demonstrated in the experimental application as the calculation for removal rate is reused in the polish time estimation. According to the engineer, he will use the block for the actual application.

8.3.2 Experiment 2: Fast Fourier Transforms(FFT) with compression

1. Background of the company

FFT-Co is a Singapore-based front-end semiconductor manufacturer. The company has a Computer Integrated Manufacturing (CIM) division to implement automation and integration projects for internal customer. The company has a team of about 20 engineers with various backgrounds. Most of the engineers have mechanical and production backgrounds because most of them are posted from the production department to the CIM department. A lot of them learn how to write software using database related tools for information processing or SCADA related applications for supervisory control and monitoring. Most of the engineers are unfamiliar with embedded systems and always have to sub-contract the embedded application development to external vendors. The problem they encounter is that most vendors' solutions are not flexible and lack continuity. They also have found difficulty in retaining the knowledge of existing engineers as well as hiring new engineers both locally and from overseas who have multi-disciplinary expertise.

The front-end wafer manufacturing process occurs within a clean room environment which required fanles embedded systems and embedded applications. FFT-Co wanted to use the methodology for wireless sensing devices with Fast Fourier Transforms (FFT) to collect and monitor wafer process data and clean room data at the sensor source. The company wanted to use the results for various monitoring of the wafer manufacturing process.

The company's Computer Integrated Manufacturing (CIM) division provided the requirements and resources for this experiment by assigning an existing contract engineer who is capable of implementing the FFT algorithm. The engineer is familiar with Java language and is trained in Computer Integrated Manufacturing with a Mechatronics Engineering background. The contract engineer is only available for 2 months from mid-December 2007 to mid-February 2008. Since the company has mixed background employees, they wanted to explore a new methodology to capture knowledge and encapsulate some of their in-house developed algorithms in a Function Block in order to simplify their usage and future implementation. The engineer provided the domain expertise and assistance to develop an experimental proof-of-concept for the methodology. The company aims to find a methodology that could be easily understood by its non-software trained engineers.

2. Step A: Requirements definition

FFT-Co proposed FFT signal compression device model designed to perform a remote monitoring task as shown in Figure 8-13. The signals acquired from the sensor need to be digitised for further processing and analysis. Once the analogue-to-digital conversion (ADC) is done, the signal is subject to an FFT process, which gives the frequency domain representation of the acquired signal. The frequency domain information can be analysed to provide information about the environmental conditions in the clean room. Once the required processing has been carried out, the frequency information to stored for transmission.

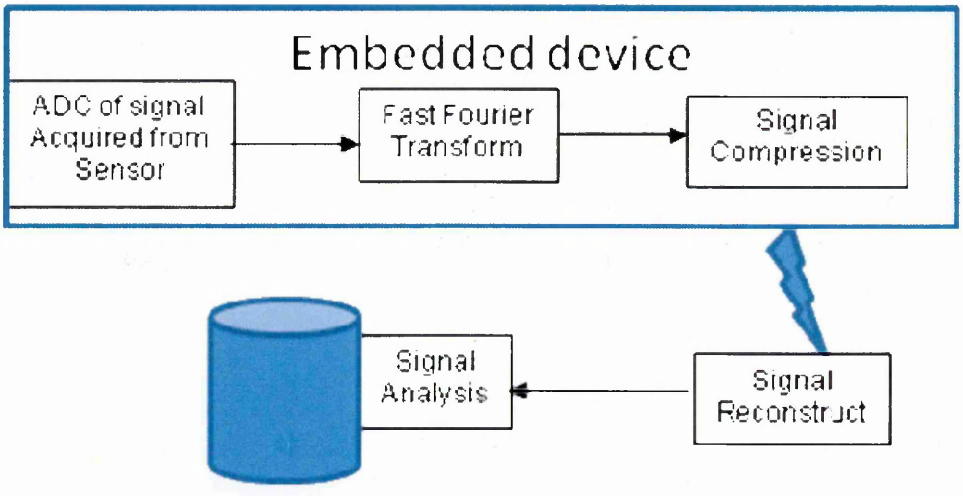


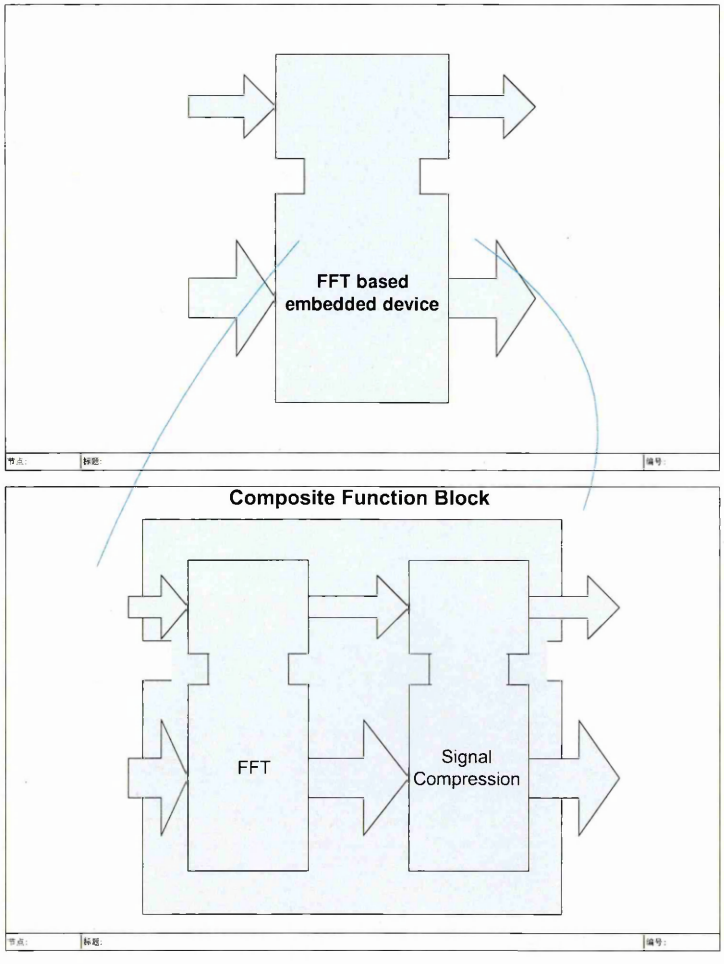
Figure 8-13: Proposed device model

It is possible that these frequencies be stored in a more efficient manner that requires less storage space through the use of compression algorithms. A potential 50% reduction in storage space can be realised by performing this operation. The compressed signal can then be transmitted to company data analysis station where they can study the frequencies and develop some technology to combat defects highlighted by the data. Thus, the transmission efficiency will be vastly increased, since only 50% of the data is transmitted instead of the entire the signal. Table 8-2 summaries the requirement step for the FFT embedded device.

Table 8-2: FFT user requirements table

a) Requirements gathering	
Goals	To explore the use of a wireless sensing device with Fast Fourier Transforms (FFT) to process clean room environmental data at the sensor source.
Functions	**FFT:** FFT algorithm block which gives the frequency domain representation of the acquired signal. **Compression:** able to effectively compress a given signal into half of its size.
Constraints	• Data will be accurate up to 4 decimals place. • Must adopt reusable functions. • Can encapsulate existing algorithms. • Need to compare result with Matlab and compression.
Resources	• Company will provide both the FFT and compression algorithms and domain expertise. • Engineer from company. • Max 2 months with 40% of time to participate.
b) Analyse requirements	
Listing the functions	FFT and compression Function Blocks.
Relations	FFT and compression will be standalone Function Blocks.

Table 8-2: FFT user requirements table (continued)

c) Top-down sketch	
	
d) **Review requirements**	
Changes	No.
e) **Hardware determination**	
Hardware	Wireless sensor node.

The engineer commented that the form is simple and could capture key items of the requirement. He mentioned that each company would modify the form to include more information so that it could serve as part of the project documents.

3. Step B: Detail design

This subsection provides the detail design steps and development methods for the FFT and compression operations.

Fast Fourier Transforms (FFT)

The algorithms for computing the FFT and signal compression were initially developed in the Java programming language by CMP-Co's engineer. In order to validate the FFT source code, a sine wave input and square wave input were fed into the Java-based FFT program. The results obtained were compared with the output from similar MatLab software. The Java code was then encapsulated into a Function Block. The entire module can be built as a composite Function Block which is built on a network of interconnected basic Function Blocks. Each of these basic Function Blocks were built and tested independently. At later stages, the required basic Function Blocks were interconnected with each other as an automated object that performs the required application.

The basic Function Blocks were tested independently by feeding in a set of values and the comparing output against a set of manually calculated values. If an error was encountered for a particular basic Function Block, then debugging was performed by building the basic Function Block again and testing the Function Block after each level of modification. This procedure helped overcome some underlying bugs in the XML code. Once the basic Function Blocks were completed, the composite Function Blocks were made by interconnecting the required basic Function Blocks. The input side of the interface of the composite Function Block was connected to the first step of the basic Function Block network and the output side of the interface of the composite Function Block was connected to the last step of the basic Function Block network.

Signal compression

In order to check the integrity of the compression algorithm, the signal compression algorithm provided by the FFT-Co was fed with three different sets of values comprising a sine wave, square wave and random samples. The result after compression and reconstruction shows that the signal compressor is able to effectively compress a given signal into half its size. This can be verified by comparing the input and reconstructed values, with an almost exact match being obtained. There were small discrepancies which due to the fact that only the trend or the average values are stored or transmitted.

4. Step C: Design repository

In this case, FFT-Co has their own algorithms written in Java. The key operation from the Java function can be reuses and encapsulated into a basic Function Block. As shown in Figure 8-14, the function block has encapsulated Java-based algorithm. These Function Blocks can be reused for other applications. The results of FFT composite Function Block, Compression Function Block and Reconstruct Function Block are saved in the design repository for future reuse.

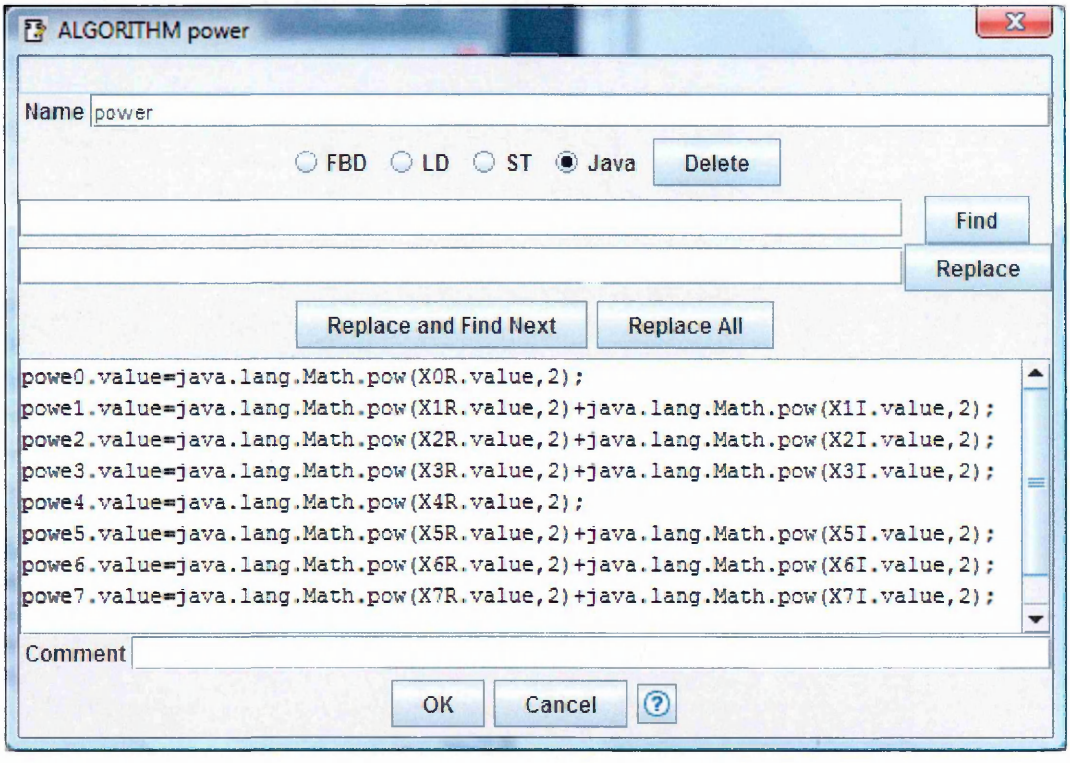


Figure 8-14: Encapsulate Java algorithms in Function Block

FFT-Co is very interested in the design repository and the feature of encapsulating Java algorithms into function blocks. The participating engineer was very impressed and planned to introduce the methodology and proof-of-concept tool to his director and engineers.

5. Step D: Design representation

The detail design is saved into a .fbt file which is in XML formation.

6. Step E: Code generation step

As FFT-Co wanted to run the application in wireless sensor nodes which have Java support, they accepted the use of Java-based code for future enhancement.

7. Step F: Implementation

Once the developed codes were validated, all the above algorithms were modelled by IEC 61499 Function Blocks. The algorithms were further split into separate functions that perform different but independent operations. Each of these Function Blocks was modelled by separate basic Function Blocks.

The interface for the composite Function Block implementation of the FFT is given in Figure 8-15. The interface of the block has the input event REQ, which triggers the operations and is connected to the REQ event of Stage 1. The input values of the interface block are connected to the input values of Stage 1. Once the REQ event of the interface block is triggered, the signal values are fed into the input values of Stage 1. Following the completion of Stage 1 operations, the CNF event will trigger the REQ of Stage 2 and the output values from Stage 1 will be fed as the input values to Stage 2. This process is repeated through subsequent blocks until the final Function Block of the network which computes the frequency spectrum of the signal. Upon executing the algorithm of the final Function Block, its CNF will trigger the output values as ready for further analysis.

The basic Function Block consists of the first step of the FFT computation which the engineer called the 'butterfly computation' (Stage 1), the second step of the 'butterfly computation' (Stage 2), the third step of the 'butterfly computation' (Stage 3), the power series computation (Power), the frequency domain representation (FFT), the signal compression (Compress) and finally the reconstruction of the compressed signal (Reconstruct).

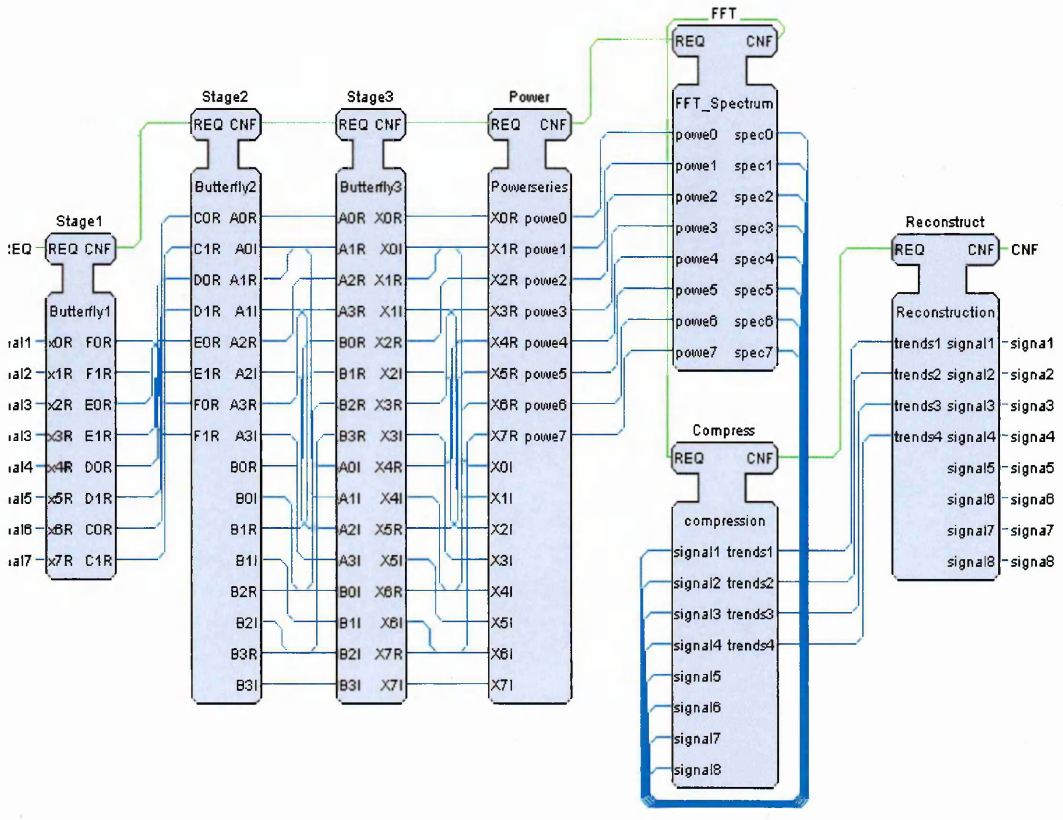


Figure 8-15: FFT with compression and reconstruction

8. Step G: Target application

The target application is to create FFT and compression Function Blocks for Java-based wireless sensor node.

Each of the basic Function Blocks were fed with test values and the generated outputs were then compared against the manually computed theoretical outputs. The output values from each of the basic Function Blocks were seen to be exactly the same as the manually calculated ones, thus the computational operations performed by the Function Blocks were confirmed accurate. Once the basic Function Blocks were built and tested, they were built into composite Function Blocks by interconnecting the required basic Function Blocks. Three sets of composite Function Blocks were modelled. One set of composite Function Blocks was for performing FFT computation, while the other was for performing signal compression and reconstruction. As shown in Figure 8-16, the input of the Sine wave will produce the FFT curve. The FFT result was verified with a Matlab result and is identical. Figure 8-17 shows the FFT-Co's compression curve which is only 50% of the original FFT bandwidth. The result of the reconstruction curve is identical to the FFT result.

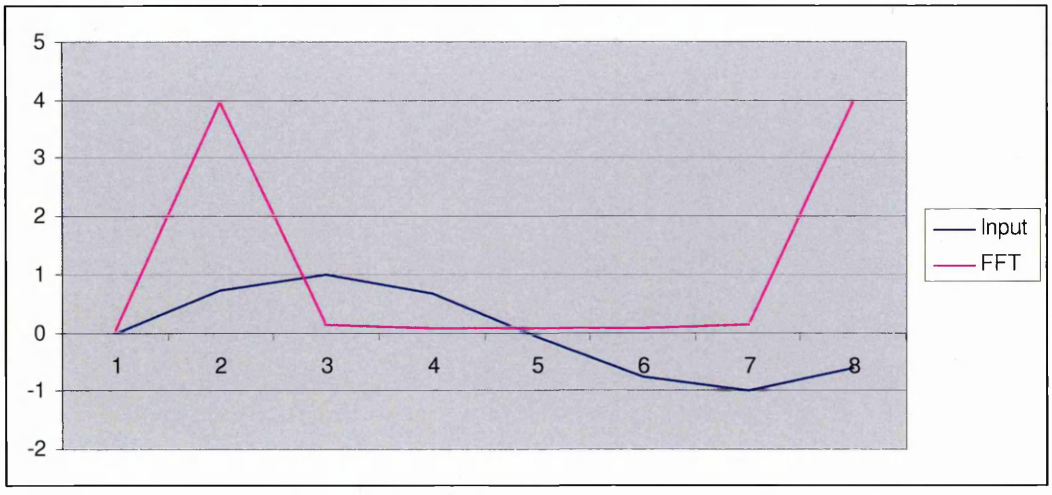


Figure 8-16: Output values of FFT

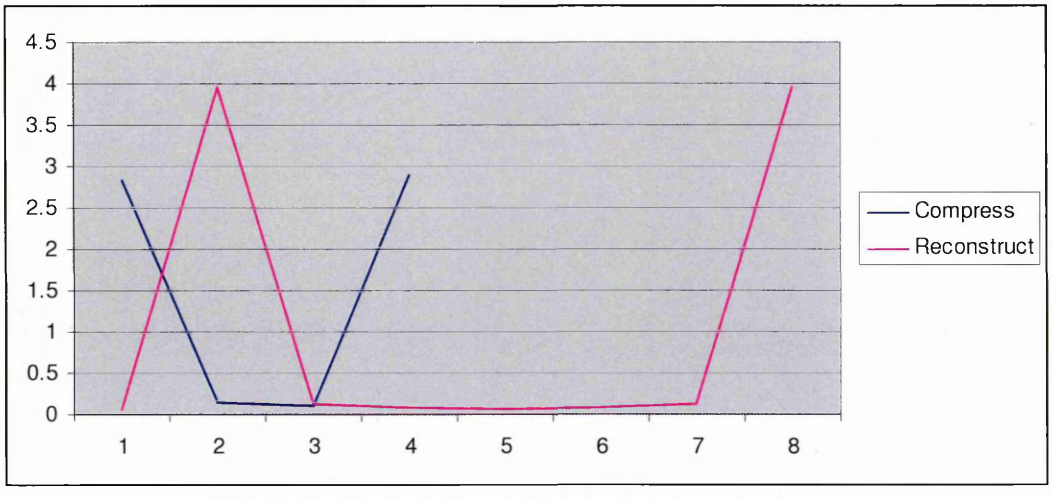


Figure 8-17: Compression and reconstruction

The advantage of using this proposed approach is that an experience engineer is able to encapsulate complex algorithms into a composite Function Block for future reuse. The user of the Function Block does not need to be software trained and does not need to understand the detailed implementation of the algorithm. In addition, from the developer's point of view, it is easier to develop and test the Function Block-based module. Furthermore, the Function Blocks created can be readily reused by other non-software trained engineers in other parts of the automation project.

9. Evaluation of the experiment

The engineer from FFT-Co was very impressed with the result. He was able to encapsulate complex algorithms into basic and composite Function Blocks for future reuse. The result was also checked with Matlab to verify the accuracy.

The common evaluation criteria of a methodology are used for discussion after the experiment. The result of the evaluation is as follows.

- Testability: The engineer from FFT-Co indicated that he was comfortable to encapsulate algorithms into Function Blocks for future use. The proof-of-concept tool provides a simple GUI interface for the engineer to check the result of each step with Matlab tools so that errors can be resolved at an early stage.

- Portability: In this experiment, only Java code is used because of Java is one of the accepted FFT-Co company standards. An FPGA-based concept was introduced to the company but due to the lack of FPGA-based wireless sensor nodes, the company did not want to dedicate time to exploring it. They expressed interest in the FPGA-based solution when there is FPGA-based wireless sensor node hardware to explore.

- Component Modelling: The experimental applications encapsulated Java-based algorithms into both basic and composite Function Blocks. The Function Blocks can be easily assembled into different applications. In this case, FFT, Compression and Reconstruction Function Blocks will be reused in potential applications in FFT-Co.

- Interoperability: In this experiment, FFT-Co has successfully encapsulated Java-based algorithms into Function Blocks. It proved that the tools are able to interoperate with other languages like Java. Another internal testing from FFT-Co also showed that the tools could interoperate with other libraries or device drivers using Service Interface Function Block (SIFB) which was mentioned in Section 7.3.1.

- Understanding: The FFT-Co's engineer has a Java programming background. He preferred to encapsulate Java code in the Function Block. He found that the proof-of-concept tool is easy to understand and he thought this would benefit FFT-Co especially getting other non-software trained engineers involved in some of the application development. He even suggested that the methodology should also be used in higher level systems such as databases or other manufacturing process controls. There were some initial difficulties in debugging due to the fact that the proof-of-concept tool was not an industrial strength tool and thus a lot of debugging features were lacking. Debugging block by

block with other tools like Matlab or manual calculation can easily solve this limitation.

- Reusability: Reusability is clearly demonstrated in the experimental application as the FFT, Compression and Reconstruction Function Blocks were going to be reused by FFT-Co in various monitoring applications. According to the engineer, FFT-Co will deploy the approach in the CMP process to measure the thickness. It seems that both CMP-Co and FFT-Co have identified the same CMP area in front-end Semiconductor process that needs improvement.

8.3.3 Experiment 3: PID controller

1. Background of the company

The PID-Co is a small local engineering service company that provides retrofit and enhancement solutions for front-end semiconductor manufacturing. PID-Co does not have any manufacturing facilities but they are very familiar with a few local semiconductor manufacturers.

PID-Co seeks to create a small device to provide Proportional Integral Derivative (PID) control and data recording enhancements for existing wafer processing equipment. The existing wafer processing equipment currently adopts an alternative controller; however, the results observed were not as good as expected. To enhance performance, the company desires to retrofit the existing controller with another add-on embedded system consisting of PID control and data recorder features. An engineer, trained in mechanical engineering with PID control knowledge but lacking in embedded software development expertise, was assigned by the PID-Co to evaluate the methodology. An assessment was thus carried out to evaluate the usefulness of the proposed methodology to address this problem. The engineer worked on the evaluation for 2 months from March 2008 until April 2008. For simplicity, the experiment concentrated on the modelling of a chemical tank level controller and data recorder using IEC 61499 Function Blocks. The aim is to maintain the level in the tank at the desired level using a PID controller.

2. Step A: Requirements definition

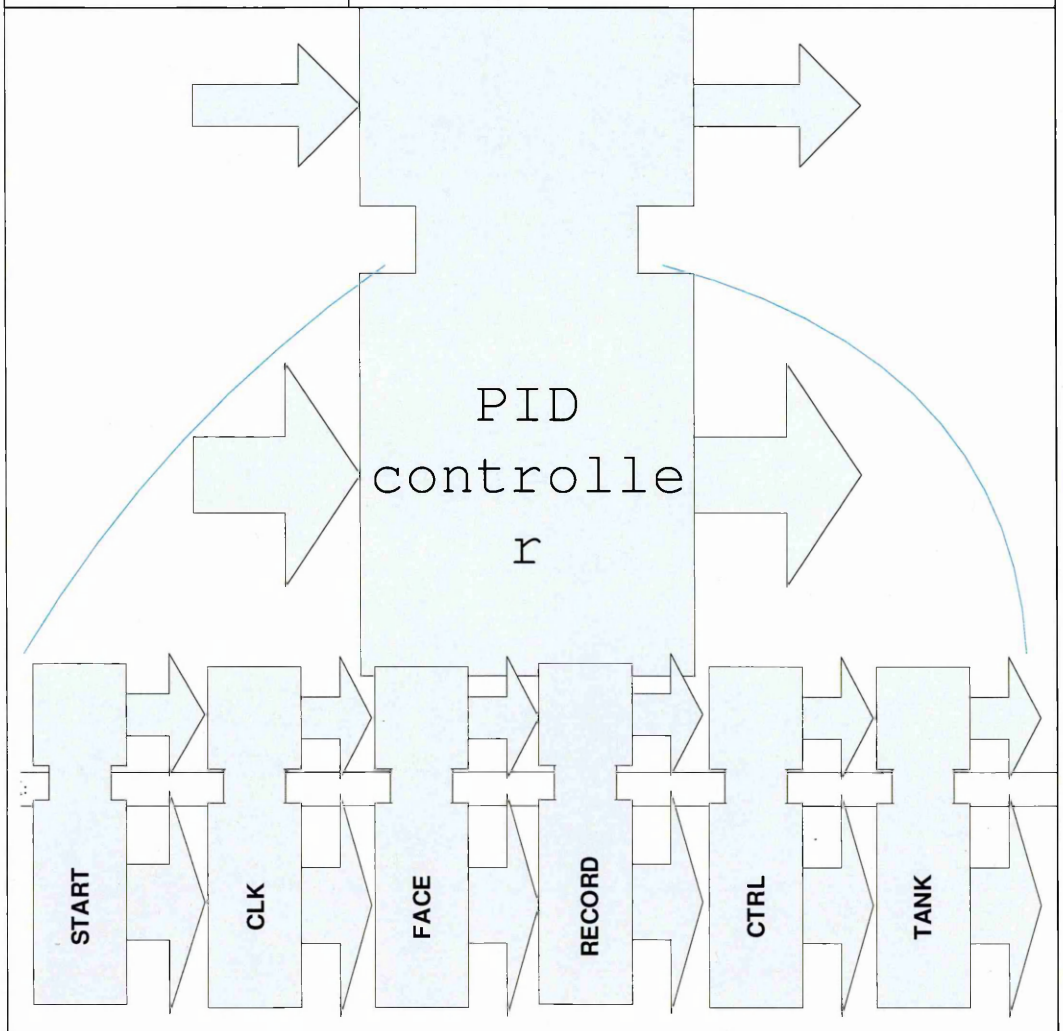
Since the wafer process may be subjected to disturbances caused by external factors or by unpredictable changes, a PID controller is used. The PID controller uses feedback signals to calculate the error between the current process values (PV) and the required process value or set point (SP). The integral and derivative values of the error are then used to generate the new output value to drive the process. This output value is used to drive an actuator in order to bring the process value closer to the set point. The PID controller

has a number of parameters that must be 'tuned' to match the process under control. This is required to ensure that the control action is responsive, smooth and with minimum overshoot. The requirement steps for the PID controller experiment are summarised in Table 8-3.

Table 8-3: PID controller user requirements

a) Requirements gathering	
Goals	To provide PID (Proportional Integral Derivative) control and data recording enhancement to existing wafer processing equipment.
Functions	**Start/Stop:** provide cold start, warm start or stop function for the PID Controller. **Clock:** generate an event at regular time intervals. Interface: to display result. **PID Control:** to perform necessary calculations required for executing a PID algorithm. **Tank:** graphical representation of the tank.
Constraints	• Data must be accurate up to 4 decimals place. • Must use reusable functions. • Must have an option of simple graphical. representation or text based results.
Resources	• The company will provide the knowledge of PID algorithms. • Engineer from company. • Max 2 months with 30% of time to participate.
b) Analyse requirements	
Listing the functions	The START block to represent start and stop, the CLK block to represent clock, the FACE block to represent the user interface, the CTRL block to represent PID control and TANK represent the level in the tank.
Relations	The START block will trigger the CLK block to generate the clock signal and run the FACE block. The FACE block will activate the CTRL block which runs both the CTRL and the TANK block.

Table 8-3: PID controller user requirements (continued)

c) Top-down sketch	
	
d) **Review requirements**	
Changes	No.
e) **Hardware determination**	
Hardware	PC 104-based embedded hardware.

The engineer was unfamiliar with detail documents as small companies normally just get a simple requirement description from customers and try to get on with the project. After explaining to them the importance of capturing requirements in documents for both internal and external customers, they agreed. They highlighted that requirement documents can be used to verify the scope of the project with customers to avoid misunderstanding. It can also used as a reference for internal technical teams to provide support and follow-up. The engineer will try to adopt the requirement document table in their company needs.

3. Step B: Detail design

The system is modelled using a network of simple and composite Function Blocks. The PID control will be executed by a PID control algorithm, which will be the main Function Block of the Function Block network. The system consists of six major Function Blocks.

- START
- CLK
- FACE
- CTRL
- RECORDER
- TANK

The above mentioned simple and composite Function Blocks need to be connected in a proper sequence and their interconnections should make the system work in the desired way. A detailed analysis of each block, the algorithm associated with each block and the Execution Control Chart of Simple Function Blocks are illustrated as follows:

START:

This is an instance of an 'E_RESTART' Block and is used to initialise the chain of Function Blocks. The 'E_Restart' Event Function Block provides single events when the STOP, COLD and WARM resources start and when the resource is stopped by an external agent. In any particular system, it is always the first Function Block in the chain of Function Blocks.

CLK:

The CLK Function Block is an instance of an 'E_CYCLE' Function Block and is used to generate events at regular time intervals. The event stream starts when an event arrives at the input of START and stops with the input of STOP. The

duration between each output event is set by DT (Duration Time). The value in DT is the sampling time to execute the chain of Function Blocks regularly at each time interval. Thus, after every time interval defined in DT, an output event EO is generated until it receives the event input STOP.

FACE:

The FACE Function Block is an instance of a 'FACEPLATE' block to display parameters by means of a bar graph. The width and height of the bar graph is determined by the input values W and H respectively. It has two input events 'INIT' and 'REQ' and two output events 'INITO' and 'CNF'. The 'INIT' event initialises the input variables used by the block and on successful initialisation, it generates the output event 'INITO'. The 'REQ' event is used to read in the latest value at the 'SPI' input, 'PV' and 'XOUT'. On successful execution, it generates output event 'CNF'. By this stage, the output variable SP should have the value of the input variable SPI. The FACE Function Block provides the bar graph display of the SPI, PV, XOUT variables respectively.

RECORDER:

The 'RECORDER' Function Block is a Simple Function Block with two input events 'INIT' and 'REQ'. The INIT event initialises the input variables like max, min scale values and pen colour. The 'REQ' event is used to read in the latest values of the input variables connected to its three pens which draw the graph.

CTRL:

The CTRL Function Block is an instance of the 'FB_PID' Function Block and it performs the necessary calculations required for executing a PID algorithm. It is the central and core part of the system since this is where the bulk of the algorithm execution takes place. FB_PID Function Block is a Composite Function Block and is composed of four Simple Function Blocks namely PRE, ITERM, DTERM and POST. It has an input event 'REQ' and is used to read in the values of the AUTO, PV, SP, KP, TR, TD and CYCLE input variables. At the arrival of each input event 'REQ', the values of input variables are passed to its internal Simple Function Blocks, invoking their respective algorithms to carry out necessary calculations. On successful execution, an output event 'CNF' with output variable XOUT is generated. The FB_PID block has two modes, 'AUTO' and 'MANUAL', set respectively by 'true' and 'false' inputs to AUTO. In the AUTO mode, the PID algorithm is used to update the output 'XOUT'. In the MANUAL mode, the PID is not active. In this mode, the output XOUT is set directly by a manually supplied value given by the XO input.

A graphical representation of the CTRL Function Block is depicted below in Figure 8-18.

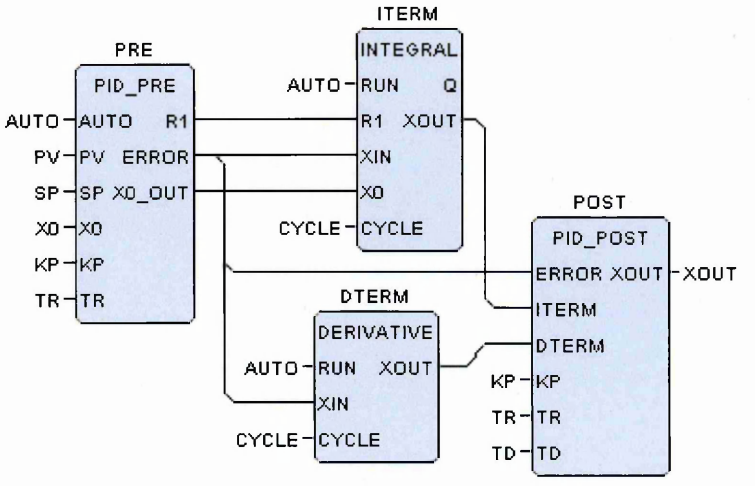


Figure 8-18: Graphical representation of the CTRL function block

The Simple Function Blocks are specified below:

- The 'PRE' Function Block is an instance of the PID_PRE Function Block. On each arrival of the 'REQ' input event, an algorithm is executed to calculate the error between set-point (SP) and process value (PV) and an output value ERROR is generated. This output ERROR is forwarded to the ITERM and DTERM blocks for evaluation of integral and derivative terms respectively.
- The 'ITERM' Function Block is an instance of an 'INTEGRAL' Function Block. On each arrival of input event 'REQ' to the FB_PID Function Block, it runs an algorithm to calculate the integral value by taking the 'ERROR' value from the 'PRE' block at its input XIN. An output as XOUT is outputted to the 'POST' block for further calculation.
- The 'DTERM' Function Block is an instance of the 'DERIVATIVE' Function Block. On the arrival of each 'REQ' input event to the FB_PID Function Block, it runs an algorithm to calculate the derivative value by taking the 'ERROR' value from the 'PRE' block at its XIN input. In AUTO mode (i.e. RUN) it calculates the derivative term as defined in the algorithm below, whereas in MANUAL mode, it gives out a zero value. The output is provided at XOUT and is sent to the 'POST' block for further processing.
- The 'POST' Function Block is an instance of the 'PID_POST' Function Block. It takes the value XOUT from the ITERM and DTERM Function Blocks at its iterm and dterm inputs respectively. At the arrival of the 'REQ' input event to the FB_PID Function Block, this block executes an algorithm to calculate the final PID algorithm output XOUT.

TANK:

The TANK Function Block is an instance of 'TANK_MV' Function Block and it performs the necessary operations required by the tank model. The actual Tank model has been simulated through Function Blocks. The TANK_MV Function Block is a Composite Function Block and is composed of three simple MDL, CNVRT and VIEW Function Blocks. It has two input events 'INIT' and 'REQ'. The 'INIT' event initialises the LVLI, TF, DT, FGD, BKGD, W and H input variables namely and on successful initialisation, the output event 'INITO' is generated. The 'REQ' event is used to read the latest value of the 'FEED' and 'DRAW' input variables. At the arrival of each 'REQ' input event, the values of the input variables are passed to its internal Simple Function Blocks and the respective algorithms are invoked to carry out the necessary calculations. On successful execution, an output event 'CNF' with the output variable 'LEVEL', is generated. A graphical representation of the three component Function Blocks that make up the TANK Composite Function Block is depicted below in Figure 8-20.

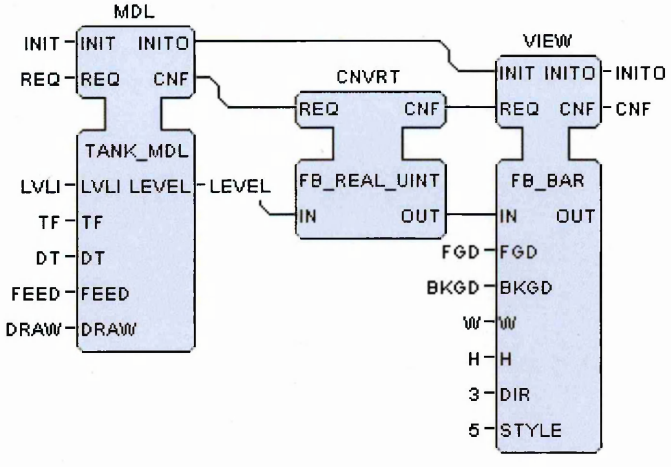


Figure 8-19: Graphical representation of the TANK Function Block

- The 'MDL' Function Block is an instance of the 'TANK_MDL' Function Block. On the arrival of the 'INIT' and 'REQ' input events, the respective algorithms are run to calculate the output.
- The 'CNVRT' Function Block is an instance of the 'FB_REAL_UINT' Function Block. On the arrival of an 'REQ' input event, it converts the real value at its 'IN' input to an unsigned integer value giving out an output event 'CNF with the output variable 'OUT'.
- The 'VIEW' Function Block is an instance of an 'FB_BAR' Function Block. On the arrival of input event 'INIT', it reads the values of W, H, BKGD, FGD and DIR and on each arrival of input event 'REQ'; it reads the value of the

'IN' input variable and generates an output event 'CNF' with the output variable 'OUT'. The VIEW Function Block shows a graphical display of the Tank model. The width, height, background colour and foreground colour can be changed by changing parameters of W, H, BKGD and FGD respectively.

4. Step C: Design repository

Blocks such as Integral, FB_Bar, Derivative are available from the IEC 61499 editor. The end result of the PID Function Block, Recorder Function Block can be reused in other applications.

5. Step D: Design representation

The detail design is saved into a .fbt file which is in XML formation.

6. Step E: Code generation step

As the PID-Co wanted to run the application in a PC-104 embedded system, they accepted the use of Java-based code which will use the standard Java-based generator and compiler.

7. Step F: Implementation

The previous section presented an overview of the functionality and properties associated with each Function Block used to model the Tank Level Controller System. The development methods are discussed here such as to how to interconnect the various Function Blocks used in the system and the execution sequence which makes the system operate in its desired manner. Since each Function Blocks may or may not require initial data to begin their execution, it is important to initialise the entire chain of Function Blocks before the normal operation proceeds. Therefore, to start the system, the chain of blocks must be initialised with their respective initial data. This is importance for the smooth operation of the system and is performed by the Function Block-'START', which generates the cold event upon system execution. The cold and warm events of the START Block are linked to the INIT event of the 'FACE' Block. On completion of its initialisation, its INITO output event is passed to the INIT event of the 'RECORDER', upon whose subsequent completion, passes the output event INITO to the INIT input event of the 'TANK' Function Block. After the 'TANK' Function Block is initialised, its INITO output event is linked to the 'CLK' Block, which generates EO event signals every 250ms (parameter provided to the DT input variable). This in turn is connected to the 'REQ' of the 'FACE' Block to smoothly carry out the desired operations of the system.

A graphical representation of the entire modelled system is depicted below in Figure 8-20.

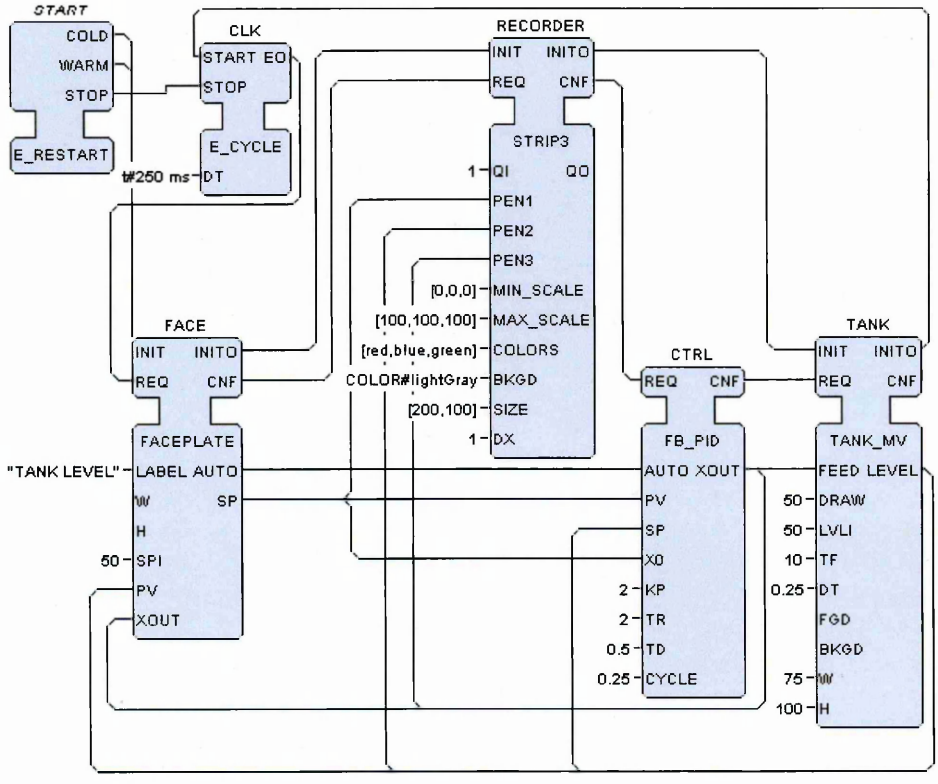


Figure 8-20: Function block representation of PID controller

8. Step G: Target application

The 'FACE' block displays a 3-bar graph indication of the Set-point, Level and PID output value. The set point can be adjusted by moving the up-down slider of the set-point bar graph. Initially a value of 50 is assigned to the set point, which is then transmitted to the SP output variable and passed to SP input variable of the 'CTRL' Function Block. The CNF output event is connected to the REQ input event of the 'RECORDER' Block. Upon the arrival of each REQ input event, the values of the parameters connected to its pens are updated. The set-point value, Level and PID outputs are connected to pen1, pen2 and pen3 respectively.

When the set point is altered, a difference in the set-point (SP) and process value (PV) (i.e. ERROR) is generated. Based on this error value, the output value of the PID controller is automatically adjusted with the aim to drive the error signal to zero. On the arrival of the REQ input event, algorithms are invoked in the various Simple Function Blocks connected to the 'CTRL' Composite Function Block. In the 'CTRL' Block, the ERROR is calculated in the 'PRE' block and is forwarded to the 'ITERM' and 'DTERM' Function Blocks to

calculate integral and derivative values respectively. Both these values are then forwarded to 'POST' to calculate final PID output i.e. XOUT.

On successful execution of the 'CTRL' block, a CNF event output is generated along with the output variable XOUT, which are connected to the REQ and FEED inputs of the 'TANK' Block respectively. On the arrival of the REQ input event, the input variables are updated and the internal algorithms of its Simple Function Blocks are invoked. The DRAW and SP parameters are initially set identical. However if SP is changed, which in turn changes XOUT of 'CTRL', a difference between FEED and DRAW is generated. This difference is evaluated in the 'MDL' Simple Function Block which passes its output to the 'CNVRT' Function Block to convert the output into integer form. This output is then fed to the 'VIEW' Function Block to get a graphical display of output. This completes one cycle of execution of the Function Blocks used to model the Tank Level Controller. The output of the modelled PID_TANK System is shown below in Figure 8-21.

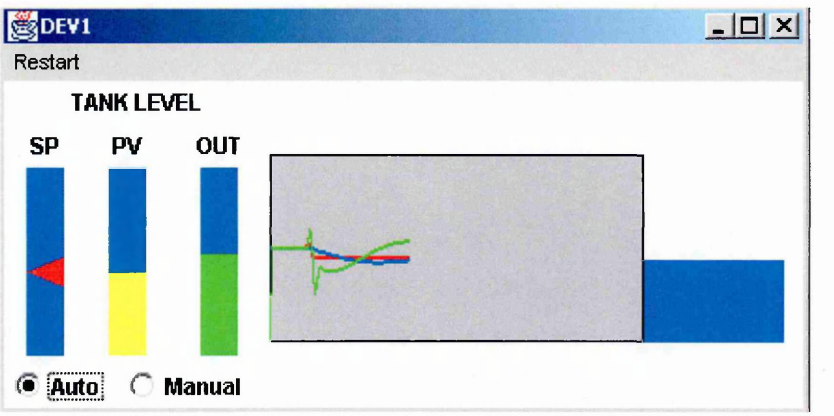


Figure 8-21: Simulation of PID_TANK system

9. Evaluation of the experiment

The PID-Co's engineer was having some difficulty especially with the graphical user interface (GUI). After searching from the repository, he found some GUI examples showing Function Blocks that can be reused. He simply assembled those GUI related Function Blocks with the PID algorithm to form into a more user friendly application.

The application needs to be connected to some external Input and Output (I/O) ports. The researcher had to demonstrate how to build new function blocks that link to external device drivers through the Service Interface Function Block (SIFB) in order to enable the engineer to test some simple I/O with sensors and actuators with the PC104 embedded system.

The engineer indicated that he was fully confident to develop a full application after the demonstration of the external I/O Service Interface Function Block and the GUI Function Blocks.

The common evaluation criteria of a methodology are used for discussion after the experiment. The result of the evaluation is as follows:

- Testability: The engineer from PID-Co indicated that he could conduct various tests with simple external I/O within the company. The engineer found that the proof-of-concept tool generated the Java-based GUI could be easily used for testing especially when the Function Blocks were build and could be easily used by other engineers to readily build and test other application.
- Portability: In this experiment only Java code was used as PID-Co. The experiment executed has run the Java application on both PC and PC104 embedded systems without problem.
- Component Modelling: The experimental applications used components-based Function Blocks. Some of the basic Function Blocks like Integral, FB_Bar and Derivative were used to construct the experimental application. The Service Interface Function Blocks to external I/O ports can also be reused for other applications.
- Interoperability: In this experiment PID-Co needed to use external I/O ports with service interface Function Blocks which linked to external device drivers written in other languages without source code. The experiment showed that the proof-of-concept tool could interoperate with external I/O port device drivers.
- Understanding: The engineer expressed that it was difficult to create service interfaces and GUI Function Blocks. Once those blocks were made available and demonstrated, it is easy to understand and use them. The experiments were built using the proof-of-concept tool with simple boxes and lines that are easily understood by most engineers. The difficulty in using GUI and service interface function block was resolved during face-to-face demonstration and discussion.
- Reusability: Reusability was clearly demonstrated in the experimental application as some of the blocks from the repository could be reused. The engineer highlighted the PID controller can be easily changed to other controllers using other

algorithms. The PID controller can run as a standalone application or as a component of another system.

8.3.4 Summary of the evaluation of the experiments

The evaluation of each experiment involved observation and informal discussion with industrial participants. A summary of the evaluation of the verification experiment and 3 industrial validation experiments are shown in Table 8-4. As before, the evaluation is based on the common evaluation criteria.

- Testability: All the four experimental applications indicate that the methodology could generate PC-based simulations and applications which allowed engineers to conduct different tests. This shows that the new methodology is readily testable through simulation and application to a practical industrial problem.
- Portability: All four experimental applications could generate code that was independent of the system implementation and could be deployed on both PC and embedded devices. All the 3 industrial experiments only used the standard code generator for Java because they found that writing XSL style Sheets was very difficult. They observed that it would be beneficial to have some simple and automated means of defining the style sheet for different languages.
- Component Modelling: All four experimental applications used component based Function Blocks. Each Function Block can be interconnected into bigger composite Function Block for different applications.
- Interoperability: The first verification experiment demonstrated a light switch application that can run in a PC based as well as an FPGA based environment. The first and third experiment used a purely IEC 61499 Function Block development environment. They could generate Java codes that can be easily integrate with other legacy components or applications. The second application demonstrated legacy algorithms that were previously built in Java. The algorithms can be encapsulated into IEC 61499 Function Blocks. This demonstrates the interoperability merits of the methodology.
- Understanding: All experiments were built using graphical Function Blocks with simple lines representing connections. This is the simplest form of design and development and can be readily understood by most engineers. As highlighted by

Graaf et al [2003], projects that employ diagrams, usually adopt mostly free-form and box-line diagrams. Function Blocks resemble the easily understandable box-line diagrams. All 3 engineers who participated in the industrial experiments suggested that a guide would greatly help understanding of the methodology. A half day workshop was conducted for potential industrial users to collect more feedback on their ease of understanding. A detailed discussion on the feedback is presented in section 8.7.

- Reusability: Reusability was clearly demonstrated in all the experimental applications. All the experiments used Function Blocks components which could be easily reused. The first experiment reused AND and NOT gate Function Blocks to construct light switch. The second reused Removal Time Calculation Function Blocks in the Polish Time Estimator embedded application for CMP. The third experiment reused a Java based FFT algorithm which was built into a Function Block. The FFT Function Block could be reused in any wireless signal analysis application. The last experiment created a reusable PID composition block. Each solution can run either as a standalone application or as a component of another system.

Table 8-4: Evaluation of the experimental applications

Experiment	Semiconductor Process	Components	Evaluation criteria						Development steps					Computer languages					Embedded hardware
			Testability	Portability	Component modelling	Interoperability	Understandability	Reusability	Requirement step	Repository step	Structuring step	code generation	Intermediate representation	Java	C	C++	SystemC	HDL	
Verification																			
Light and Switch		AND, NOT, OR gates components	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Validation																			
1) Polish Time Estimator	Chemical Mechanical Planarisation (CMP) Process	Remove Rate Calculation & Polish Time Calculation Components	✓		✓		✓	✓	✓	✓	✓		✓	✓					✓
2) FFT with Compression Device	CMP Polishing Head Monitoring	FFT (External Algorithm) & Compression, Reconstruction Components	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓					✓
3) PID Controller	Chemical Cleaning mixture controller	Recorder, Tank, Process value (PV) and Set Point Value Components	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓

8.4 Workshop and feedback

In order to get feedback from wider range of industrial participants on the proposed methodology, a half-day workshop with hands-on demonstration was conducted on 2nd July 2008 in SIMTech, which was attended by 28 participants. As shown in Table 8-5, 19 participants were from 11 companies such as electronics consumer product manufacturing, semiconductor chip manufacturing and engineering services provision. The remaining participants were from university, polytechnic and research institutes. The purpose of the workshop is to educate participants on the details of the proposed system development methodology for embedded applications and allow participants to have hands-on experience using the light switch verification example with the methodology. The workshop allows for direct communication with participants and in-depth understanding their feedback and concerns can be achieved. At the end of the workshop, a feedback form was provided with 13 participants returning completed forms.

Table 8-5: Number of responses

Groups	No. of participants signup	No. of participants joint	No. of feedback form received
Industries	26	19	10
University, Polytechnic and Research Institutes	9	9	3
Total	35	28	13
Percentage of Response			46%

8.4.1 Feedback questionnaire

The feedback questions are comprised of 3 different types of questions. Question statements with multiple choice answers, several statements based on a specific focus with different response ratings and a personal comments section. The rating feedback questions have rating ranges from 1 to 5, with strongly disagree rated as 1, disagree as 2, somewhat agree as 3, agree as 4 and strongly agree as 5.

1. Industrial sector and type of business

The aim of the first two questions is to understand the industrial and primary business activities of the participants.

(1) **Please specify the industrial sector to which your company belongs to:**

☐ Aerospace
☐ Biomedical
☐ Electronics
☐ Engineering Services
☐ Food & Healthcare Products
☐ Marine & Offshore
☐ Precision Engineering
☐ Semiconductor
☐ Transport & Logistics
☐ Others (Please specify)

_____

(2) **Which category best describes your company's business (you may tick more than one)?**

☐ Component and Sub-Assembly
☐ Contract Manufacturing
☐ Logistics Providers
☐ Maintenance, Repair and Overhaul
☐ Material and Technology Suppliers
☐ OEM
☐ Product and Service Providers
☐ Product Manufacturing
☐ Solution Providers
☐ Tool & Die
☐ Others (Please specify)

_____

2. Participants' background and experience

The purpose of questions 3 and 4 is to gather information regarding the participants' background and experience in order to segregate the users into different specialised groups.

(3) **How long did you practice software development on a professional level?**

---

(4) **What is your area of specialty for work / study?**

☐ Mechanical Engineering
☐ Electrical Engineering
☐ Computer Engineering
☐ Computer Science
☐ Business / Marketing / Arts
☐ Others (Please specify)

3. Ease of Use

Question 5 is based on an ease-of-use evaluation for the IEC 61499 Function Block and the system development methodology as a whole.

(5) **Ease of use**

	Rank levels
IEC 61499 Function Block and the development methodology is easy to use	1 2 3 4 5
IEC 61499 Function Block and the development methodology is easy to understand	1 2 3 4 5

4. Usefulness

Question 6 assesses the evaluation of the usefulness of the IEC 61499 Function Block and the system development methodology.

(6) **Usefulness**

	Rank levels
The IEC 61499 Function Block and the development methodology may help me in my current or future course of work	1 2 3 4 5
The IEC 61499 Function Block and the development methodology are versatile for my current or future needs.	1 2 3 4 5
I will recommend the IEC 61499 Function Block and the development methodology to	1 2 3 4 5

my organisation, colleagues and friends	
I will use the IEC 61499 Function Block and the development methodology if I need to do programming.	1 2 3 4 5
It is easy to implement changes or perform modifications using the IEC 61499 Function Block	1 2 3 4 5
There are a lot of potential uses for the IEC 61499 Function Block and development methodology	1 2 3 4 5
The IEC 61499 Function Block and development methodology will help to simplify application development in my organisation.	1 2 3 4 5
The IEC 61499 Function Block and development methodology will allow more stakeholders to participate in application or system design.	1 2 3 4 5

5. Interest in the event and follow-up

Question 7 to 9 focus on evaluating participants' interest in the workshop, to receive further news, discussion and the benefits of the event.

(7) **Are you interested in receiving news, events and training relating to IEC 61499 Technologies?**

☐ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?**

☐ Yes ☐ No

(9) **Did you find this event beneficial to your company?**

☐ Yes ☐ No

6. Other comments

The final section of the feedback form provides space for participants to give any personal comments related to the workshop.

8.4.2 Analysis of the feedback results

The response is measured based on the feedback forms received. As shown previously in Table 8-5, the percentage of response was 46%, since some of the participants had to leave early and did not complete the form. The following section shows the details of the feedback received.

1. Industrial sector and company business

Figure 8-22 shows that 46% of the participants are from the electronics sector, 31% from engineering services who serve many sectors including the electronics sector.

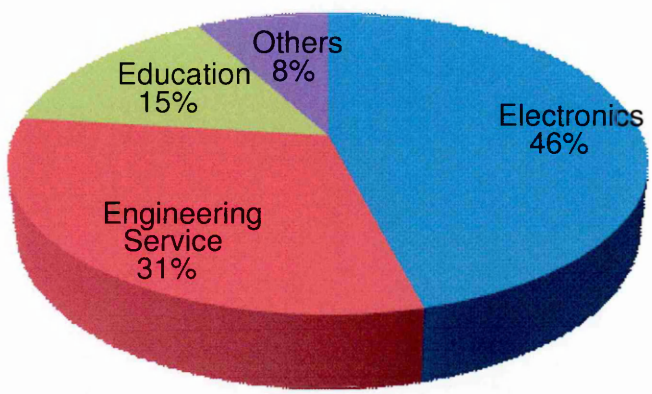


Figure 8-22: Main industrial sectors

2. Participant profiles

Question 3 of the feedback form focussed on the number of year's experience participants have in software development. As shown in Figure 8-23, 8 participants (61%) had no software development experience whatsoever. 23% had 1 to 5 years of software development experience and only one person had more than 25 years experience.

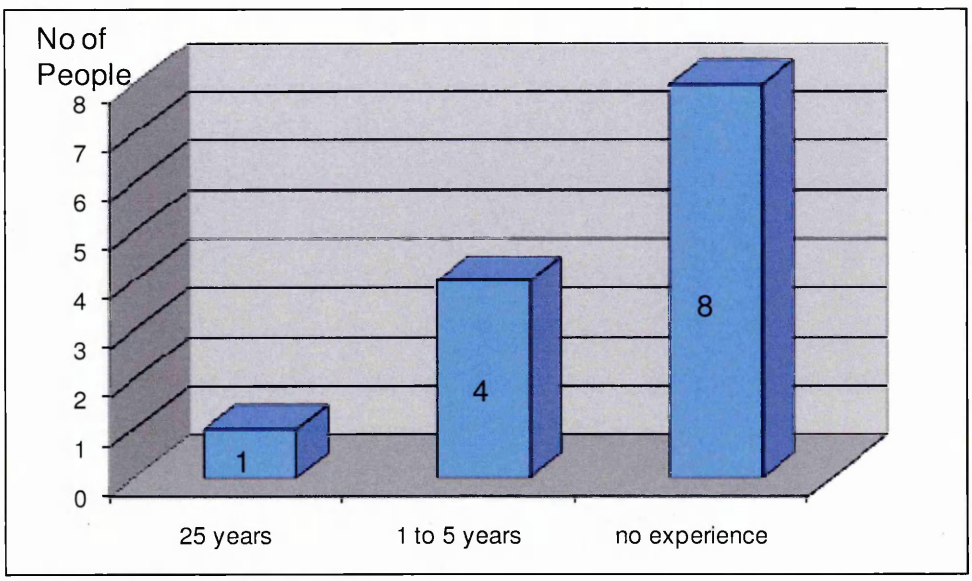


Figure 8-23: Years of experience in software development of participants.

Table 8-6: Summary of the feedback score

Participant	1	2	3	4	5	6	7	8	9	10	11	12	13	Average (Ave.) Score	Ave. Score for 0 year s/w experience	Ave. Score for 1-5 years of s/w experience	Ave. Score for 25 years of s/w experience
Question 3 Years of S/W experience	25	3	5	2	1	0	0	0	0	0	0	0	0				
Section 5																	
5.1 Easy to use	3	3	3	4	5	5	3	4	5	4	4	4	4	3.9	4.1	3.8	3
5.2 Easy to understand	3	2	4	4	5	5	2	4	4	4	5	4	4	3.8	4.0	3.8	3
Average score for Section 5 *Average percentage for Section 5*														*3.9* *78%*	*4.1* *81%*	*3.8* *75%*	*3.0* *60%*
Section 6																	
6.1 Help in the course of work	4	2	3		5	5	3	3	5	4	5	3	3	3.8	3.9	3.3	4
6.2 Versatile	4	2	4	5	5	5	4	3	4	4	5	4	4	4.1	4.1	4.0	4
6.3 Recommend to others	4	3	3	5	5	5	3	3	5	3	5	3	4	3.9	3.9	4.0	4
6.4 Will use the tools	3	2	4		4	5	3	3	5	3	5	3	4	3.7	3.9	3.3	3

Table 8-6: Summary of the feedback score (continued)

Participant	1	2	3	4	5	6	7	8	9	10	11	12	13	Average (Ave.) Score	Ave. Score for 0 year s/w experience	Ave. Score for 1-5 years of s/w experience	Ave. Score for 25 years of s/w experience
6.5 Easy to implement changes or modify	4	2	3	5	5	5	4	3	5	4	4	3	4	3.9	4.0	3.8	4
6.6 Potential for the methodology	4	2	4	5	5	5	3	3	5	3	5	4	3	3.9	3.9	4.0	4
6.6 Potential for the methodology	4	2	4	5	5	5	3	3	5	3	5	4	3	3.9	3.9	4.0	4
6.7 Simplify application development	4	2	4	5	5	5	3	3	5	4	5	4	3	4.0	4.0	4.0	4
6.8 allow more stakeholders to participate	4	2	3	5	5	5	3	3	4	3	5	3	3	3.7	3.6	3.8	4
Average score for Section 6														*3.9*	*3.9*	*3.8*	*3.9*
Average percentage for Section 6														*77%*	*78%*	*75%*	*78%*

3. Methodology usability

The usability criterion was intended to prove how easily the methodology could be applied and followed. As shown in Table 8-6 question 5.1, the participants' perceptions of the usability of the methodology were good with an average score of 3.9 or 78%. For the participant who has 25 years of software development experience, a lower score of 3 or 60% was given. He agreed that the methodology is easy to use. Those participants who did not have any software experience found that the methodology was easy to use and gave an average score of 4.1 or 82%. As shown in Table 8-6 question 5.2, participants' agreed that the methodology is easy to understand and rated it 3.8 or 76% on average. Those participants who have no software experience could easily understand the methodology and gave average score of 4 or 80%.

4. Usefulness of applying the methodology

The usefulness criterion was intended to prove that the methodology could provide a useful output to meet expectations. From Table 8-6, all measures of usefulness of the methodology in Section 6 highlight that the methodology was perceived to be good by all participants. The overall feedback from the post workshop feedback on the usefulness of the methodology was 77%. For the eight questions asked, participants with no software development experience and the participant with 25 years software development experience scored 78%, participants that had 1 to 5 years software development experience rated it at 75%. Many of the participants have indicated that the overall workshop and methodology has been very successful and well worth attending.

8.5 Analysis and discussion of results

The objective of this Section is to use the results from testing of the methodology to analyse its strengths and weaknesses and to identify opportunities for further refinement.

8.5.1 Strengths of the methodology

From the results of the verification and 3 industrial validation application experiments, numerous strengths in the proof-of-concept tool of the methodology could be observed. First, as a whole, the methodology is systematic and generic, comprising of seven steps that are unconstrained by existing available tools. The proof-of-concept tool demonstrated the openness of using some of the potential international standard available and the implementations of the proof-of-concept tool are not restricted to a single source of technology.

The sequence of the steps is consistent and makes logical sense. The methodology is well suited to non-software specialists and many of the

participants have indicated that the overall process has been successful and the workshop well worth attending. It simplifies the system development for embedded application and allows users with or without software development experience to participate in the system development process. The result also provides a new opportunity to have reusable designs and components, as well as reducing the manpower requirement for highly specialised staff. Some participants have indicated that they would even explore the development of their own tools using the methodology as a long-term plan of the organisation. The proof-of-concept tool provides a window to enable non-software engineers to participate in the design and development of the embedded application.

Each of the steps has their own unique strengths such as:

- The requirement definition step captures the application and technical requirements from users. The user can use the IEC 61499 Function Block to capture the flow of information, control requirements and logic functions. The Function Block resembles the box and line diagrams which can be easily understood by all levels of users.
- The detail design step converts the technical and application requirements into various software or hardware components and produces a system level architecture. The IEC 61499 Function Block tools allow the developer to draw the blocks and easily organise the events and data links. Previously built components can also be used. Each component will have its own algorithm, internal data, execution method, event and data flow controls. The user can also structure the application using various Function Blocks, with the methodology also having the flexibility to use existing algorithms or components.
- The design repository step provides the repository of Function Blocks components, external libraries, XSL definition, template and reference design for reuse. In the currently used proof-of-concept tool, a simple file directory is used to simplify the user interface. In practical implementation of the tool, the company can link to other databases or component repositories.
- The design representation step uses open standard XML method to represent the IEC 61499 Function Block design. XML is an open, simple and flexible text format that can be used with other tools. This is an open opportunity for future enhancement and extension of the tools. XSL is used to define the style and format of the target language.
- The Code generation step uses XML representation of the detail design and XSL to represent the formation of the target language. Since both XML and XSL are international standards, the engineer could choose

difference parsers to generate the target source code and would not be restricted to company specific software tools.

- The implementation step uses proprietary tools like computer language compilers and translators to convert the source code into the target applications. XSL style sheets to generate SystemC source which could be easily converted to VHDL for FPGA. This step was demonstrated using simple gates and logic for the light switch application. The user can simulate the system using an IEC 61499 proof-of-concept tool and view the results on the PC prior to downloading to an embedded hardware platform.
- The target application step shows that the IEC 61499 design can be used in various embedded applications on either FPGA-based or microprocessor-based hardware platforms.

8.5.2 Weaknesses of the methodology

This section discusses some of its weaknesses. During the evaluation of the proof-of-concept tool and feedback form analysis, various weaknesses are identifiable, especially during the code generation step using XSL as the style sheet. Most participants commented that these steps should be automated or semi-automated, especially regarding the task of developing the XSL style sheet, as this was very tedious, time consuming and required the assistance of an experienced software developer. Furthermore, some participants suggested that a guide with more detail content to the methodology would be useful in helping their embedded system development. Some of the participants suggested that the implementation and target application steps are closely related and could be combined into a single unified step. They also suggest that design representation could be defined in the detail design step.

It was also noted that some participants wanted to further develop the proof-of-concept tool to a fully functional suite, as the current proof-of-concept tool provides only a proof-of-concept demonstration of the proposed methodology. With the experimental examples shown in this chapter, participants can readily extend or enhance the existing proof-of-concept tool.

8.5.3 Summary of opportunities for further refinement

The opportunities to improve the methodology are identified from analysis of the strengths and weaknesses outlined in the previous sub-sections. The opportunities to improve the methodology fall into 2 categories, namely:

Participants suggested simplifying the creation of the style sheet using XSL because it is very tedious, time consuming and required experience software developer to assist. This is necessary and it can be simplified by using the

computer language grammar definition, which is widely available with interactive parsers to generate the target languages, as a style sheet.

As a whole, the methodology and proof-of-concept tool can be summarised into a guide. The guide should be independent of the implementation of the software but with structure and content. The methodology guide can be supplemented with step-by-step application examples. This is a necessary improvement to better aid potential users in their understanding of the methodology and use of the proof-of-concept tool.

8.6 Chapter summary

This chapter has addressed the fourth objective of the research, which was to evaluate the development methodology for embedded systems using a proof-of-concept tool with experimental applications from relevant industries. The experiments helped illustrate the beneficial attributes of the proposed methodology, particularly with regards to its ease of use through the IEC 61499 Functional Blocks and versatility when compared to previous approaches. The workshop and feedback helped to gain an in-depth understanding of how well the methodology was accepted by participants and to determine whether the methodology is sufficiently robust. The results and feedback obtained have indicated a number of important improvements which would benefit the proposed methodology, which form the basis of the work in the following chapter.

CHAPTER 9 REFINEMENT OF THE FINAL METHODOLOGY

In the previous chapter, three industrial experiments were carried out to validate the applicability of the proposed system development methodology for embedded applications using the proof-of-concept tool. As a result of the experiments, workshop and constructive feedback, a number of areas of potential improvement emerged. This chapter addresses stage 5 of the research, which is to further refine, deploy and present the final methodology. The chapter begins by stating its objective and method in Section 9.1, followed by a presentation on code generation refinements in Section 9.2. An overview and description of final methodology is presented in Section 9.3, with chapter conclusions provided in Section 9.4.

9.1 Method

The objective of stage 5 of the research is to refine the final system development methodology for embedded applications. This is the final stage of the research, which comprised a logical sequence of the following stages:

1. Review of the existing system development methodologies (Chapter 5)
2. Formulate the proposed conceptual methodology (Chapter 6)
3. Develop the proof-of-concept tool (Chapter 7)
4. Evaluate the methodology (Chapter 8)

The purpose of this stage is to collate all the refinement suggestions obtained at stage 4 in order to fine tune the final system development methodology for embedded applications. From the 3 validation experiments and workshop described in the previous chapter, further refinements have been suggested by users which need to be incorporated before the system development methodology for embedded applications can be finalised. As shown in Figure 9-1, these refinements can be summarised as:

1. Improve the code generation using a computer language grammar definition that is widely available and create a parser to generate the target language;
2. Develop a simple guide by combining some steps together, defining the generic content for each step and providing simple procedures in an easy to understand flow.

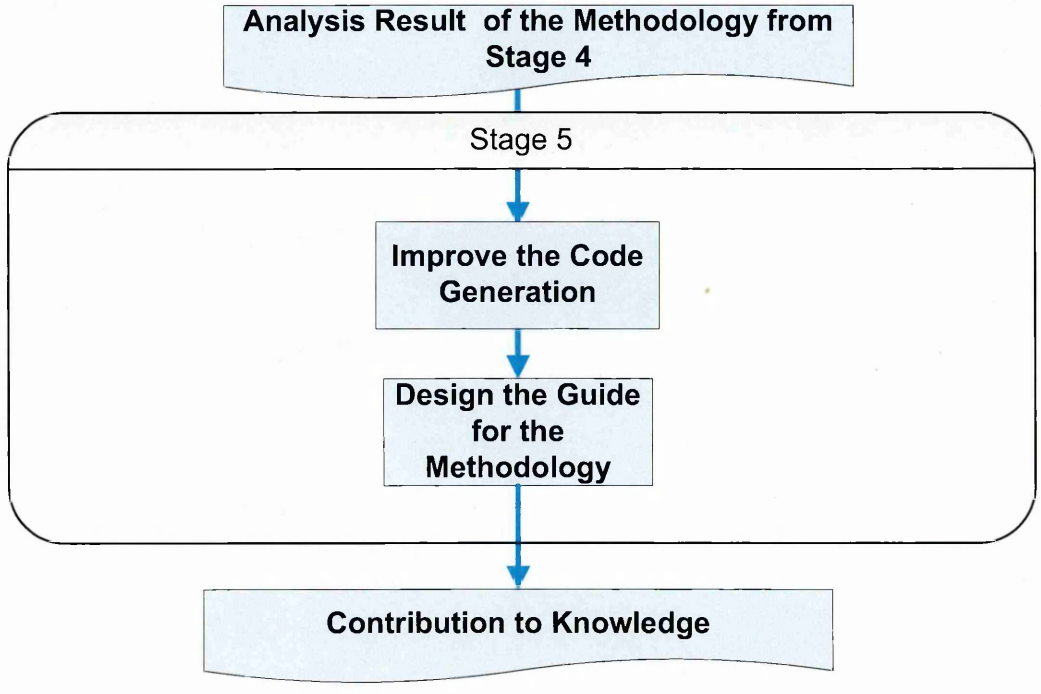


Figure 9-1: Stage 5 – Refined final methodology

Having outlined the primary refinements needed, the following sections respectively describe improvements to the code generation and the creation of a simplified guide for the final system development methodology for embedded applications.

9.2 Improving the code generation

This section sets out to refine the code generation based on the feedback from the validation experiments and workshop detailed in the previous chapter. In stage 4, it was mentioned that using style sheets with XSL is very tedious, time consuming and requires the assistance of experienced software developers. As highlighted in the feedback, it is necessary to simplify this process and this section thus proposes an interactive parser using the computer language grammar definition as the style sheet.

9.2.1 XML tree

In IEC 61499, there are six types of Function Block types, namely FB, Device, Resource, System, Data and Adapter Types [IEC, 2005] corresponding to six types of generated XML documents. In other words, the style sheet for each Function Block type can be predefined. For each style sheet, when compared with the customised XML document from IEC 61499, differences are notable in the node's attributes. In the following portion of XML code, additional nodes

named "VarDeclaration" with attributes such as "name" and "type" inserted without influencing the target codes' generation.

```
<InterfaceList>
<InputVars>
<VarDeclaration Name="IN1" Type="BOOL" />
<VarDeclaration Name="IN2" Type="BOOL" />
</InputVars>
<OutputVars>
<VarDeclaration Name="OUT" Type="BOOL" Comment="Result" />
</OutputVars>
</InterfaceList>
```

The node's attribute is a leaf node for further data extraction. The XML tree's leaf node, depicted in Figure 9-2, is the attribute node.

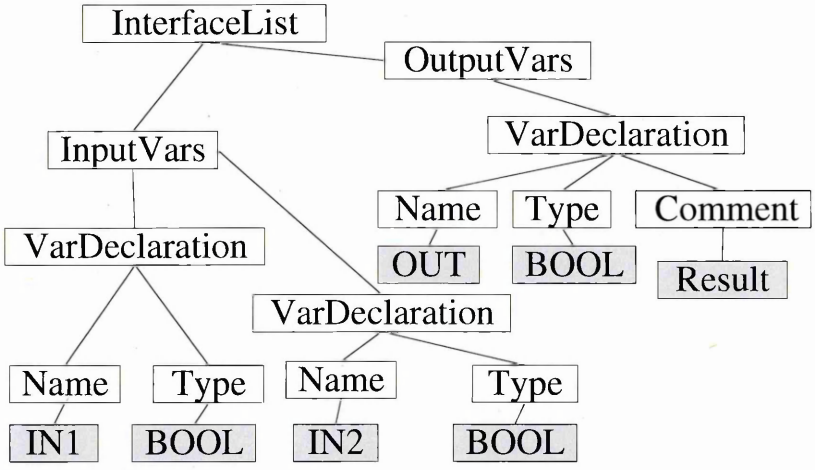


Figure 9-2: XML tree construction

9.2.2 Extended Backus–Naur Form

The Extended Backus–Naur Form (EBNF) is used as the grammar definition for programming languages. In EBNF, the syntax expression is highly convenient. It consists of four operators, "|" which represents "alternatives", "[ ]" represents "option", "{ }" represent "repetition" and "( )" represent "grouping" [ISO/IEC, 1996]. Each language's EBNF has a start symbol which is the root node of the EBNF Tree. For each of the EBNF rules, the Left Hand Side (LHS) is a tree node and the Right Hand Side (RHS) are the children of this node. For example:

```
decl-begin ::= 'SC_MODULE' '(' module-name ')'
```

where the LHS, "decl-begin", is a tree node and the RHS, "'SC_MODULE', '(', 'module-name' and ')'", are its children. The majority of computer languages have their grammar definitions represented in EBNF. EBNF simplifies the definition of style sheets through more standard definitions. Using an EBNF definition for the target computer language as a style sheet subsequently removes the necessity for XSL. Furthermore, EBNF, like XML, can be represented as a hierarchical tree. Users simply need to match the XML tree to the target language EBNF tree to create a new style sheet for code generation. The style sheet can be saved for future use. The EBNF Tree contains three kinds of leaf nodes: the language's identifier node, the language's keyword node and the EBNF keyword node. The leaf node, which is also called the attribute node, in an XML tree corresponds to the language's identifier node. An improved interactive graphical user interface (GUI) code generator is developed, which simplifies the creation of the style sheet by dragging the XML definition to match the target language EBNF. Figure 9-3 shows the interactive code generator GUI. The detail guide of the code generator can be found in Appendix C.

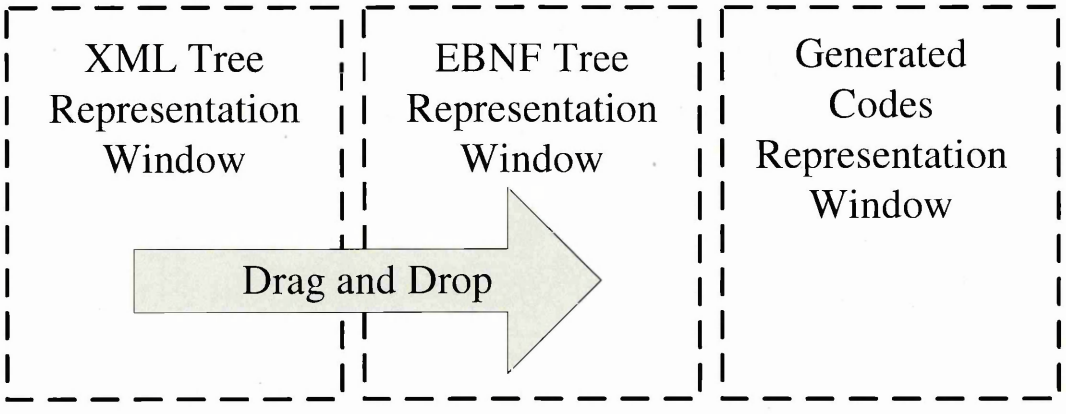


Figure 9-3: Interactive code generator

9.2.3 Code generation

The main problem with EBNF code generation is the need for repetition. Repetition has the EBNF syntax format - {a}, a{, a} and {a, }a respectively. The repeat times can be found by identifying how many similar nodes exist for a specified EBNF node. When several similar nodes are found in an XML tree, the EBNF node that needs to be first associated is always that which is outside the curly brackets. Take a{, a} for example, assuming there is no similar XML node to be found, there is only one node to match to the corresponding EBNF node. If the first "a" is not assigned to the XML node, the result becomes ", a" while the expected result is "a".

Searching for other nodes are similar to a specified XML node. The code generator identifies whether repeat EBNF nodes need to be added which will only occurs on the tree level that contains "{" and "}" symbols. The value of this node's corresponding EBNF node will be replaced, but the remaining nodes' corresponding to newly added EBNF nodes will not. This is because the EBNF node itself does not know where its newly added EBNF nodes are located. Only the nodes on the level where "{ }" resides know this information. One solution is to append a reference for this EBNF node, listing the location of other "similar" nodes.

1. Similar attribute identification

When an attribute is assigned to a corresponding EBNF node, similar attributes can be identified by searching the attribute node's parent node. Figure 9-4 below illustrates the procedure.

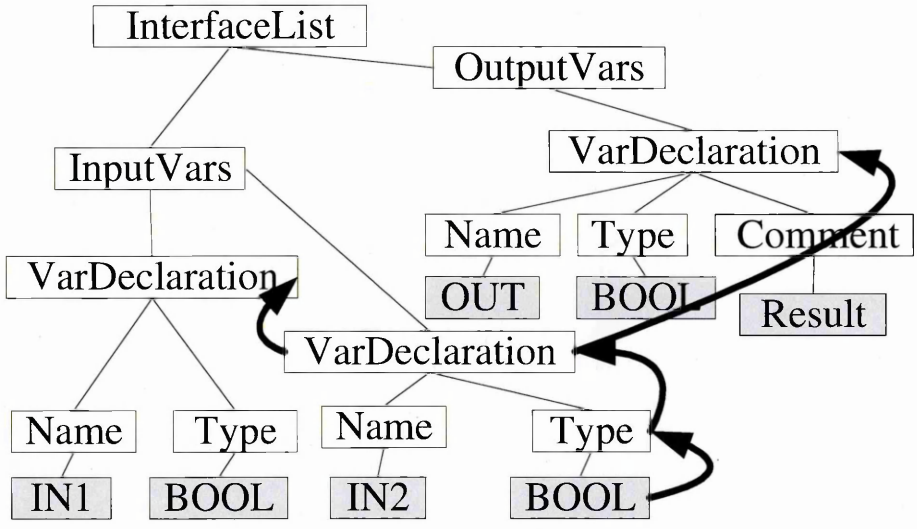


Figure 9-4: Similar attribute identification

If there is a specified node called "VarDeclaration", then any node named "VarDeclaration" will be regarded as being a similar node.

2. Searching for similar nodes

Complex EBNF rules cannot be handled automatically in the initial stage. Take the previous piece of XML code, for example, the "IN1" and "IN2" are the input variables and "OUT" is the output variable. Different languages handle inputs and outputs differently. In SystemC language, sensitive list would be initialised as "sensitive <<IN1 <<IN2" while an Interactive Code Generator may generate "sensitive <<IN1 <<IN2 <<OUT". The problem lies in identifying the similarities

between these three nodes. On the "VarDeclaration" level, these nodes seem to be the same, however on examining their parent nodes, differences become evident, as shown in Figure 9-5. Proceeding up to the next parent node, it can be seen that both nodes converge to "InterfaceList". In this case, "IN1" and "IN2" are the most similar nodes and will be handled in a similar manner, while "OUT" contains some differences and may be treated through different means. The algorithm to determine the nodes' similarity can be described as follows:

Step 1: Find those nodes whose parent's parent node has the same name. Put them in a group and set their similarity index to 0.

Step 2: Look up one more parent node for each node, taking each node itself as the reference. Compare this node individually with each of the other nodes. Those nodes that have parents with the same name will keep their original similarity index. Those whose parents have different names will increment their similarity index by 1.

Repeat step 2 until all nodes converge to the same parent.

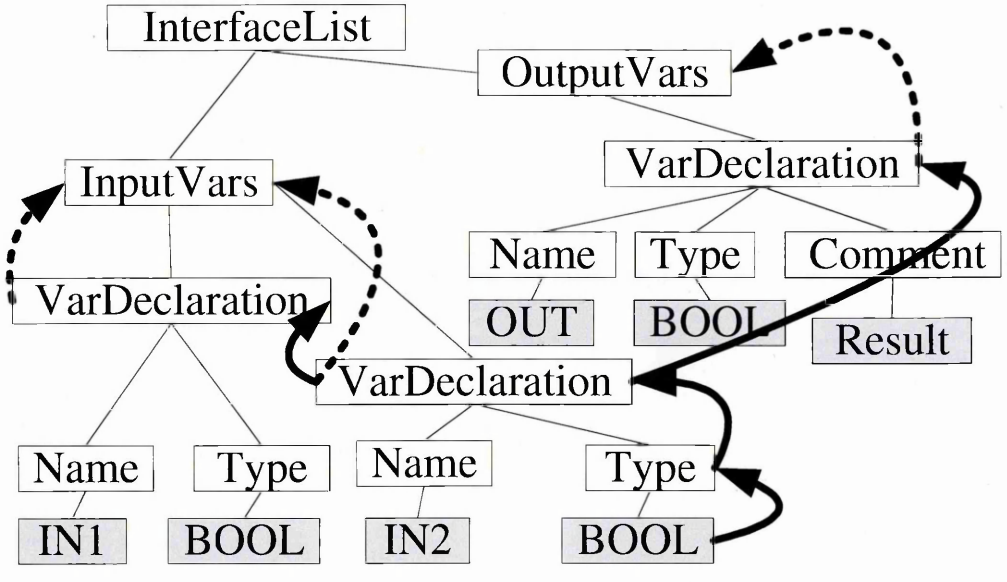


Figure 9-5: Similar node searching

3. Relating the similarity index to EBNF

The similarity index says how "far" to look up the tree structure in order to determine which node should be considered as a similar node. Nodes deemed similar will then be treated in the same manner. When a new XML template is first loaded, the parser will start from the attribute node's parent to determine similarity. If so required, the user could interactively configure the EBNF tree using the GUI code generator. The corresponding relationship between the

XML and EBNF nodes will then be stored, allowing for the desired codes to be generated accordingly.

4. Data type matching

The standard data types defined in IEC 61499 include Integer, Floating, Boolean, String, Date and Time types. For each type, several sub-types may exist, e.g., Integer type contains "BYTE", "WORD", "INT" and so on. If required, the user can also define various derived data types, with data storage being in an XML format [Lindlar & Zimmermann, 2008]. For the Interactive Code Generator, every language's data type follows the data type classification in IEC 61499. The methodology in Figure 9-6 shows how data types' auto-matching works. After dragging and dropping the XML node "BOOL" into the EBNF node "Boolean", the Code Generator will record the relationship between the two roots. The EBNF node and all its sibling nodes are thus data type definitions in this language and correspond to those defined in IEC 61499.

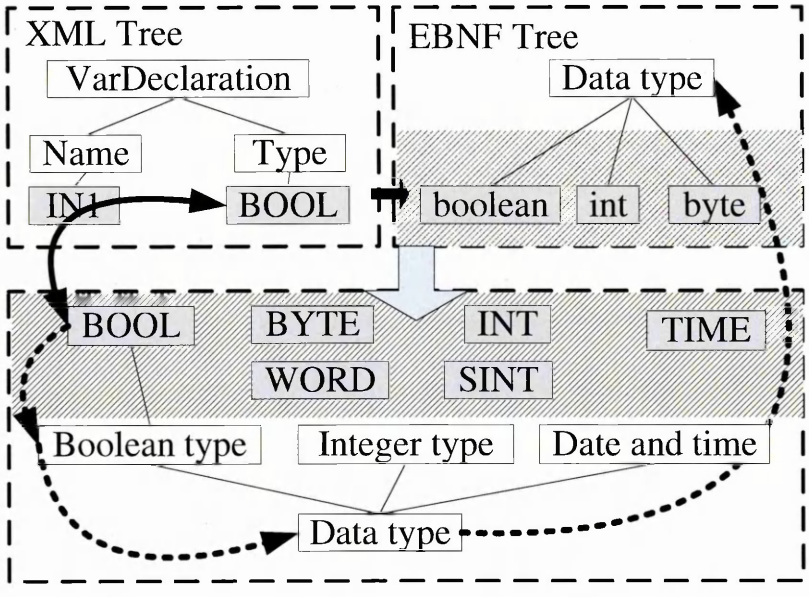


Figure 9-6: Data type matching

9.2.4 Example of converting XML to SystemC

In this section, an XML document generated from an IEC 61499 proof-of-concept tool is studied. This example shows the code generation from a Basic Function Block represented by the first column in XML tree, using the style sheet from second column and output SystemC codes in the third column. The GUI in Figure 9-7 below illustrates the procedure. The detail steps of how to setup the EBNF code generator is in Appendix C.

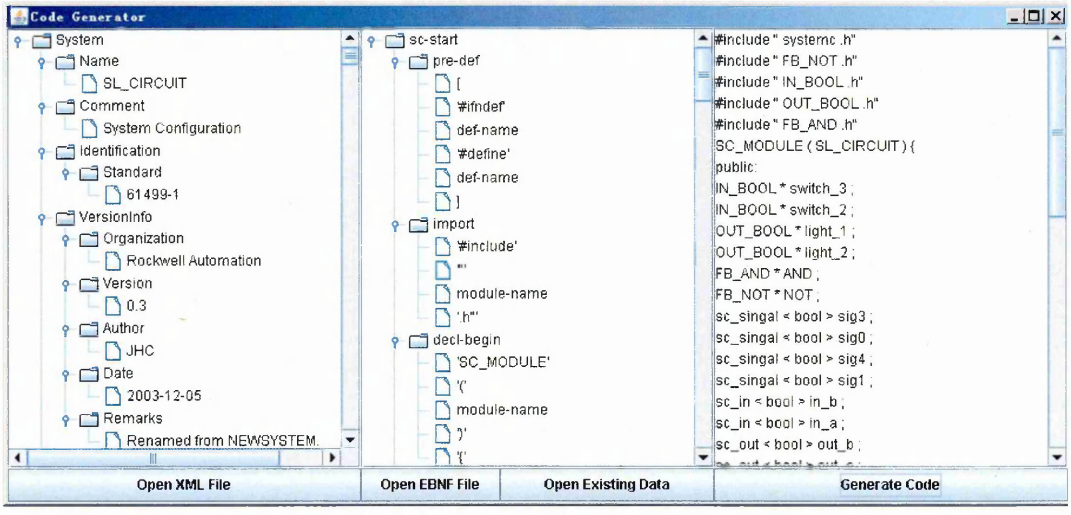


Figure 9-7: Interactive code generator GUI

In summary, the interactive code generator uses the EBNF as a generic means of generating various computer languages' code from XML documents. The core idea is to minimise the human-computer interaction while maximising generated code accuracy. The next section describes the second refinement, which is to provide a detailed guide to the final system development methodology for embedded applications.

9.3 The final system development methodology for embedded applications

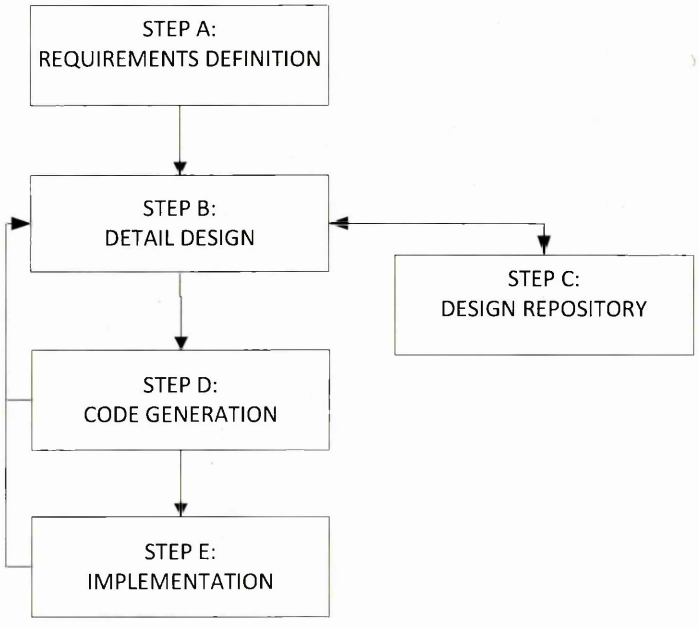


Figure 9-8: Final system development methodology

Figure 9-8 shows the final system development methodology for embedded applications. As suggested by workshop and experiments participants, the design representation will be part of the detail design step because design representation will be based on open standards like XML and could be specified in the detail design step. The target application will be part of the implementation step because the target application is the end result of the implementation which could be combined together. This will simplify the final system development methodology to 5 steps instead of 7. The description of the final system development methodology for embedded applications is provided in the form of guide to illustrate the overall structure, the content of the individual steps as well as descriptions of their usage.

The content of the system development methodology for embedded applications includes the input, process, techniques and output. To enable the development of the embedded application, developers need to systematically following the content of the methodology which consists of input, process, technique, output together with tools to construct the system that solve the problem [Denning et al, 1988; Nunamaker et al, 1991; Gregg et al, 2001]. Figure 9-9 shows an overview of the content for the final system development methodology. It consists of:

9.3.1 The input

The Input could comprise documents, diagrams, interviews, discussions or the outputs from software tools. In the requirements definition steps, the purpose of the input is to collect system requirements through interviews and discussions with users or customers. For all subsequent steps, the input is similar to the output of the previous step.

9.3.2 The process

The information from the input must be processed in order to get the desired output. Most of the processing steps involve manual decisions to organise and analyse the input information to develop it into a usable output. Some of the steps like the design repository step, code generation step and target application step require specific software tools to produce an output. While some of the necessary software tools may be readily available, those being unavailable require further investigation and development as a proof-of-concept tool.

9.3.3 The technique

A technique is a practical method applied to some particular task. Some of the techniques require software tools, for example the code generation step and implementation step needs software tools like parsers and compilers. The techniques for other steps may only require user interviews, brainstorming,

technical discussions or drawings, which could be carried out with or without the use of software tools.

9.3.4 The output

The output could be in the form of documents, drawings, software files, source code or executable files. In the requirements step, the output is the documentation of the requirement and top-down system specifications. On the other hand, the output of the code generation step is source code that is required by the specific target application platform development software tool to be compiled into an executable file.

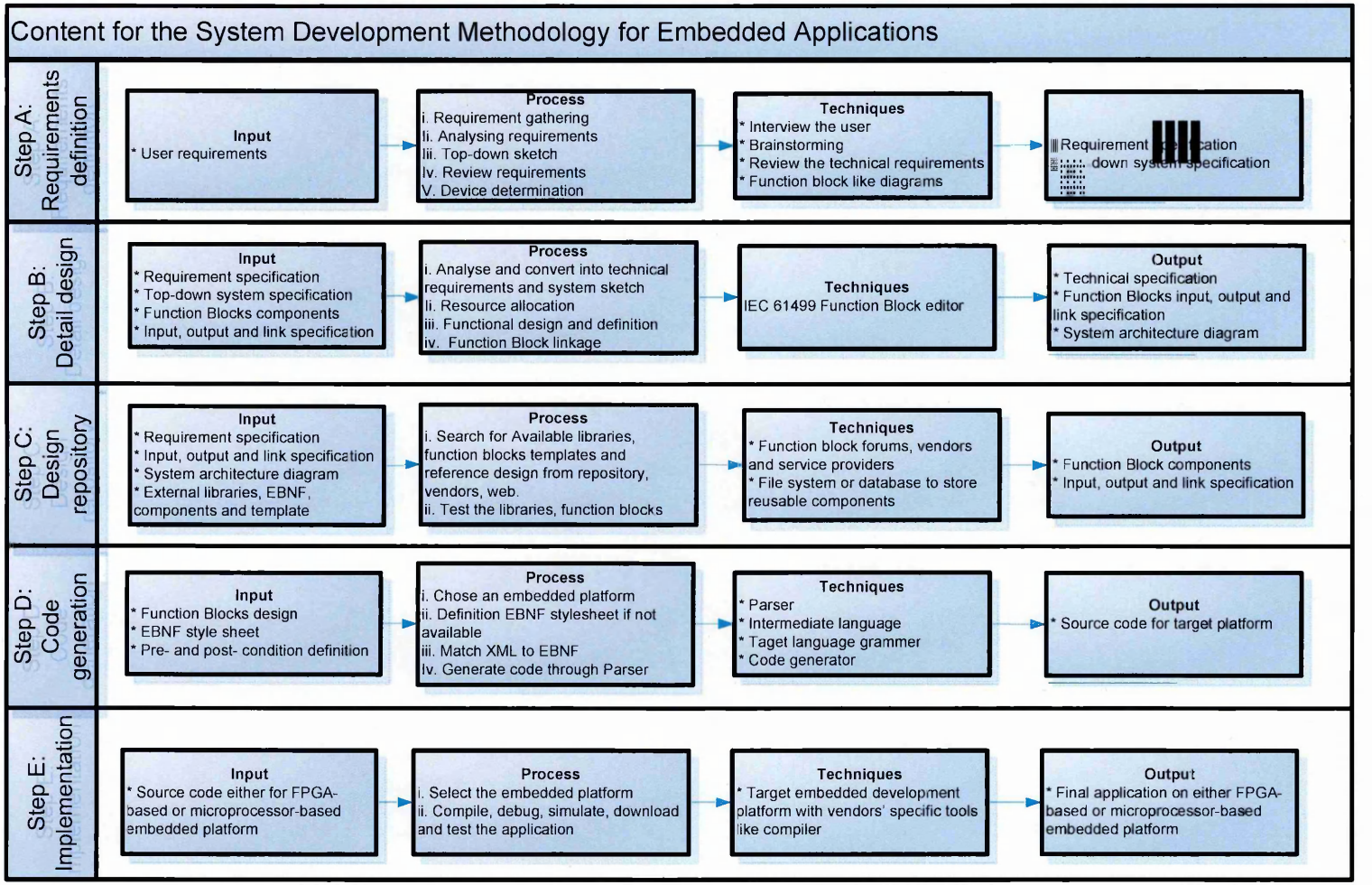


Figure 9-9: Content for the system development methodology for embedded applications.

9.4 Design the guide for the methodology

9.4.1 Step A: Requirements definition

The objective of this step is to obtain the requirements from users and customers.

1. Input

The input is mainly users' or customers' requirements through discussions and interviews.

2. Process

a) Requirements gathering: The first task in the step is to communicate with customers and/or users to determine what their needs. In the requirement gathering process, three main types of requirements can be gathered from the users. The three key questions to be asked are shown in Table 9-1 below:

Table 9-1: User requirements table

What are the goals or functions? List as many as possible.	
What are the constraints? The details of time, precision and so on.	
What are the resources? The tools, references, people and so on.	

b) Analysing requirements: The next task is to determine whether the requirements are unclear, incomplete, ambiguous or contradictory and to then resolve and clarify any potential issues with the users. Once these issues are clarified, we could,

i. List all the functions

ii. List all the relations between each function.

c) Top-Down sketch: The next task is to draw a topology of the requirements through IEC 61499 Function Block representation and give a top-down description of the requirements. As introduced previously in

the Section 7.3.1, the Function Block is a function unit, which can represent a system and/or a sub system. It can be used to capture user requirements by sketching the main system as well as detailing its inputs, outputs and internal sub-systems. A top-down Function Block sketch can provide a detailed system overview, from a general topography, down to each individual sub-system as shown in Figure 9-10.

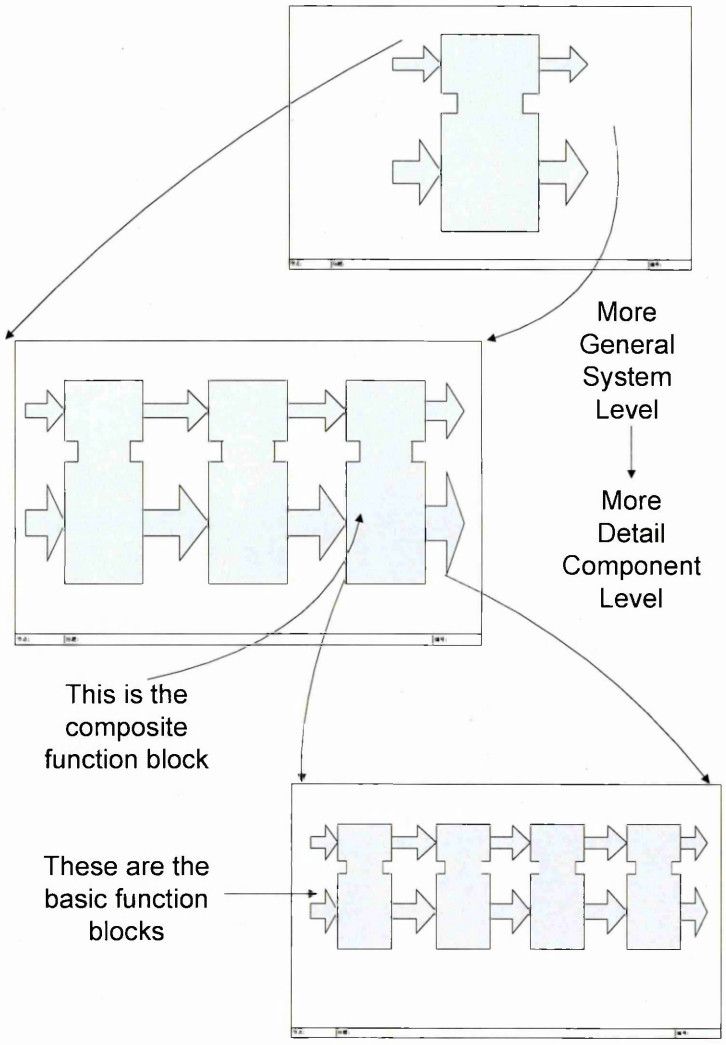


Figure 9-10: Top-down sketch

d) Review requirements: Typically, user requirements cannot be fully defined at the beginning of the project. Some requirements may alter over the course of the development, perhaps by internal or external

influences. It is essential to review the requirements with users and make the necessary adjustments before moving onto next task.

e) Hardware determination: The final task for this requirements step is to decide what type of embedded system is best suited, according to user needs, the system functions, constraints and resource requirements.

Table 9-2 shows the user requirements table that assist in the process of requirement definition.

Table 9-2: User requirements table

a) Requirements gathering	
Goals	
Functions	
Constraints	
Resources	
b) Analyse requirements	
Listing all the functions	
List all the relations between each function	
c) Top-down sketch	
d) Review requirements	
Changes	
e) Hardware determination	

3. Tools and techniques

The IEC 61499 Function block is a simple box and line diagram which can capture the event/data input, process and event/data output for the systems

and sub-systems. Other techniques include user interviews, brainstorming and frequent requirement reviews.

4. Output

The output of the requirements step is requirement specification document and a top-down system specification.

9.4.2 Step B: Detail design

The main aim of this step is to convert the user requirements into system and technical requirements, which consist of various software components, hardware components, reusable components structure and system level architecture. Components could be software-based components, hardware-based components or a combination of both. Each component has its own interface, functions, control and attributes. These comprise the building blocks for the whole application.

1. Input

The output from the previous requirements step acts as the input to the structuring step. It includes the requirements specification documentation, the top-down sketch from Step A and Function Blocks components, input, output, link specification from Step C.

2. Process

During the requirement definition step, the relations between each function and the devices are defined. Given the user requirements, the final application and the physical device can be sketched into the structure of the final system according to their defined relations. The procedure is described as follows:

- System sketching:
    i. Draw a graphical representation of the physical device, system and implementation diagrams as shown in Figure 9-11.
    ii. Summarise the user requirements for each step as shown in Figure 9-12, Figure 9-13 and Figure 9-14.
    iii. Implement the system sketch using IEC 61499 Function Blocks.

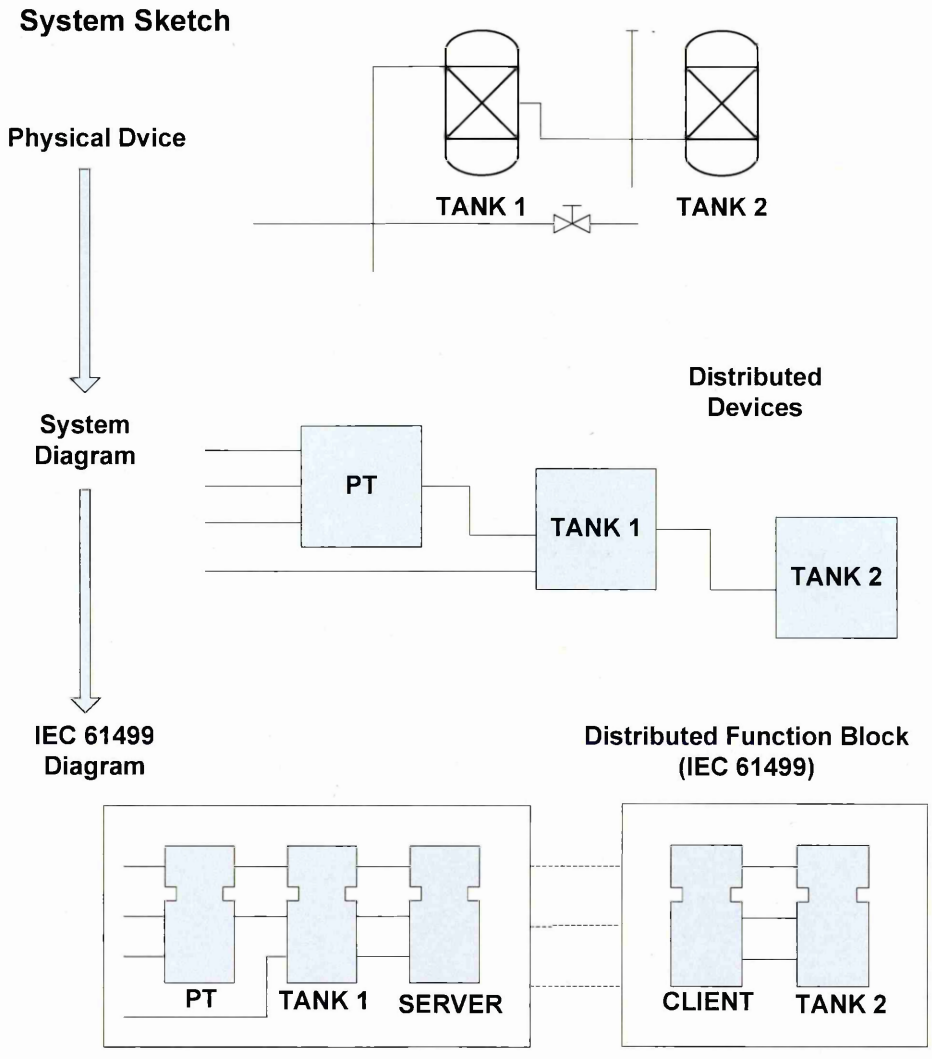


Figure 9-11: System models

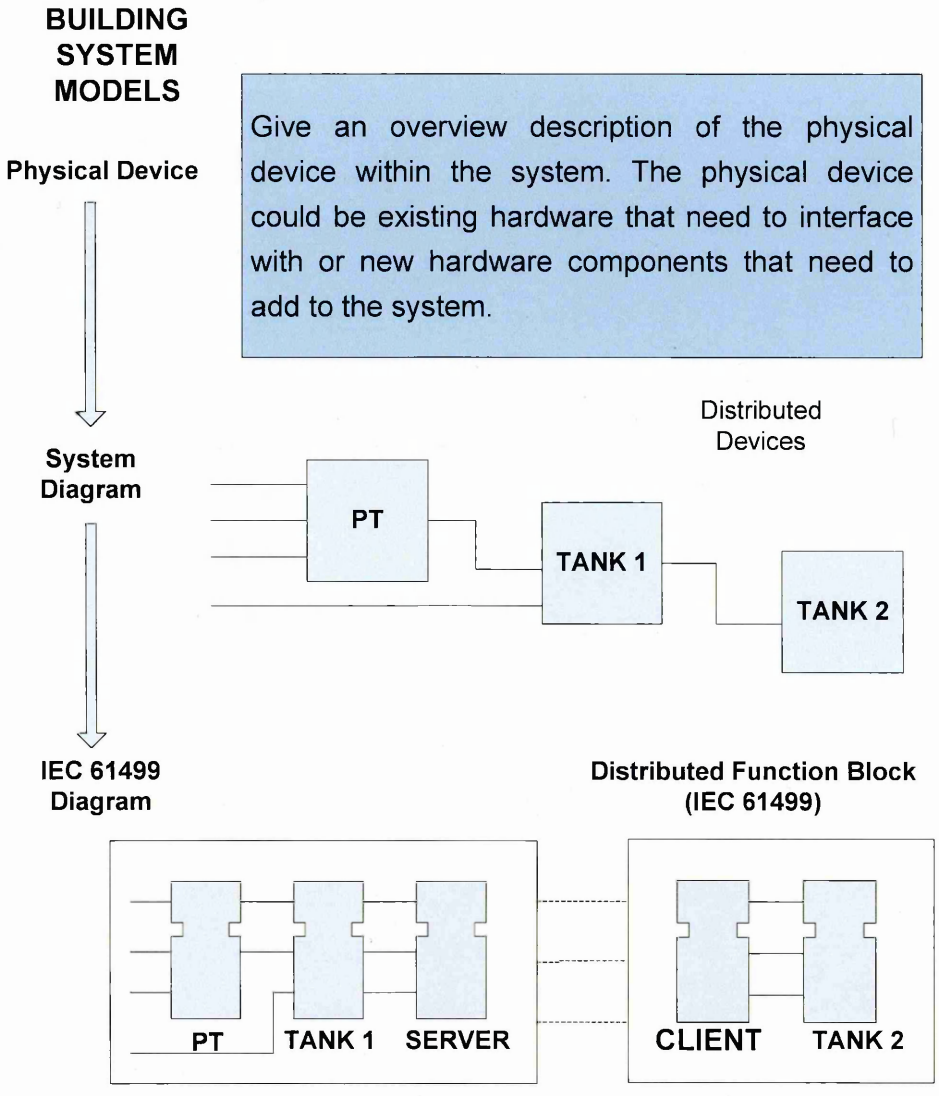


Figure 9-12: Description of the physical device

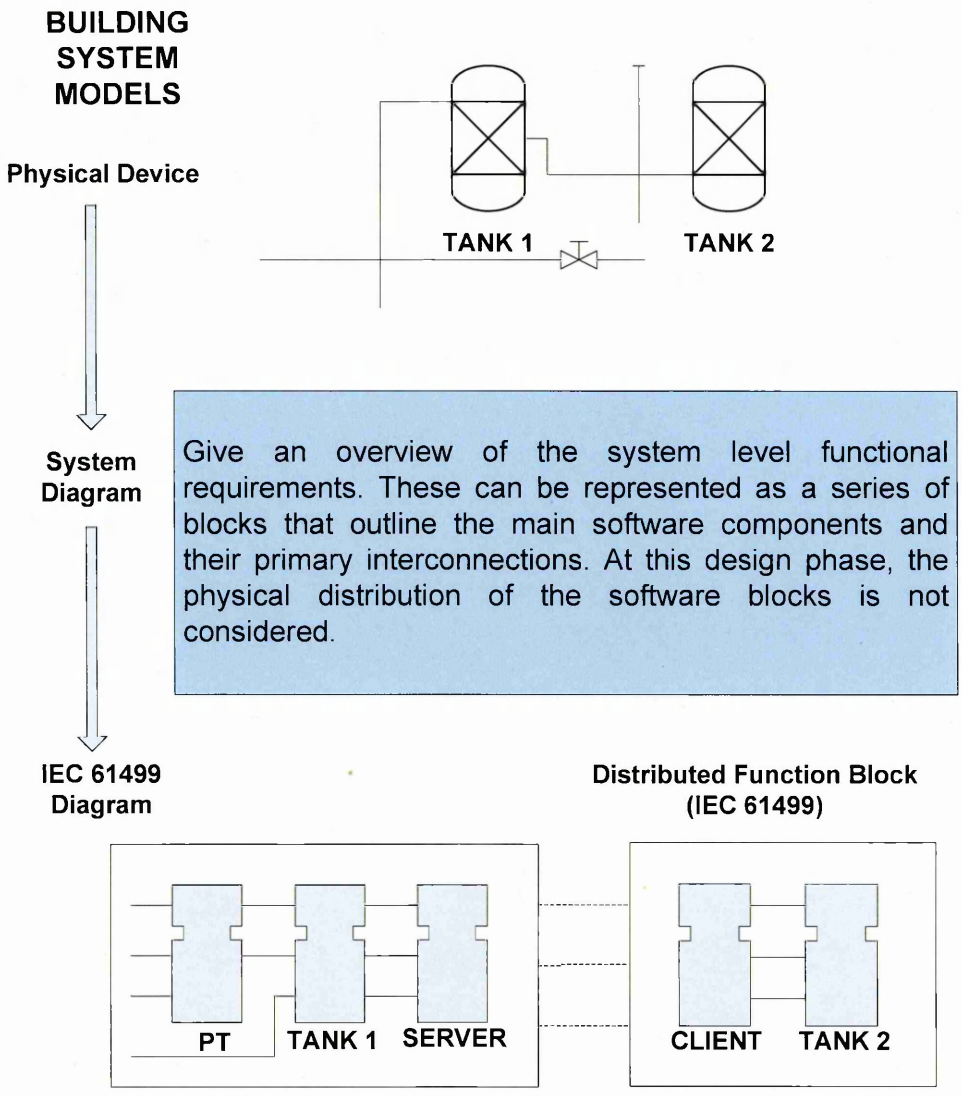


Figure 9-13: Description of the system

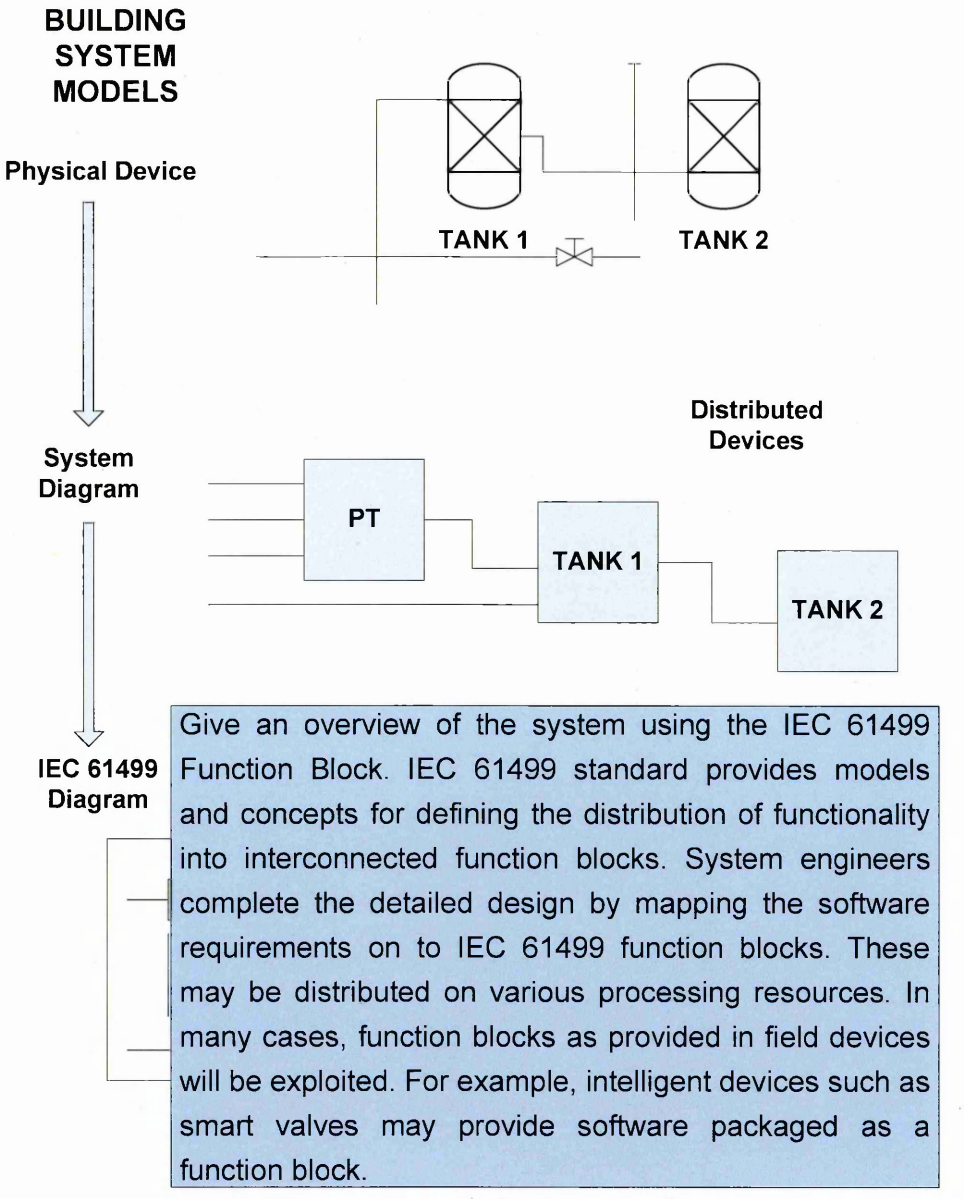


Figure 9-14: Description of the implementation

- Resource allocation: According to the concept of a resource model of IEC 61499, resources are allocated in the devices as a functional unit. In this step, devices are to be assigned into the resource unit. The assignment of the resource model should be agreed from the previous requirement step.
- Function design and definition: Define the functional unit with basic and composite Function Blocks. Define external or distributed resources with the Service Interface Function Block.

- Function block linkage: Link the events and data to Function Blocks inputs. The output of the Function Blocks should also trigger and provide inputs to other Function Blocks until the final objective is achieved.

3. Tools and techniques

The physical device overview can be created with any drawing tools. Both the system diagram and Function Block diagrams can be formed using the IEC 61499 Function Block tools. The description can be created from the comments generated by the IEC Function Block tools.

4. Output

The output of this step is the Function Block design, function input, output and link specifications, system architecture diagrams and technical specifications.

9.4.3 Step C: Design repository

The objective of this step is to encapsulate functionality into reusable blocks through the use of a repository of implementations, software templates, reference design and style sheets, either from previous projects or external sources. EBNF-based Style sheets define the style of the target application languages which could be in C++, VHDL, Java, C or HTML languages, Code generator performs conversion from the XML-based Function block representation into the target embedded application source code. The templates and reference designs can be reconfigured to other applications, thus increasing the speed of the embedded application development cycle.

1. Input

The input for this step encompasses the outputs from step A, B and D, which include the user requirements, technical specification documentation, the input, output, link specifications for the Function Blocks, EBNF Style sheet, external libraries, design template as well as the system architecture design document.

2. Process

There are two main types of templates, the Function Block template and the non-Function Block template. Since both types of templates will be used in different sections of the methodology, the following will give a summary of both templates.

a) Templates and reference design for Function Blocks.

   i. Available template: There are various types of Function Blocks such as the basic, composite and service interface. Project design or reference design of an individual module may also be carried

out through the use of Function Blocks. The source of the templates can be various vendors who sell templates, open source repositories, web searches or from previous projects.

ii. Modifying the Function Block: If the existing Function Block does not perform as required, it can be modified according to the IEC 61499 standard.

iii. Saving in the repository: New and modified Function Blocks should be stored in a repository for repeated future use.

b) Templates and reference design for non- Function Blocks.

i. Available template: Some templates may not be from a Function Block source. For example, the EBNF grammar definition is a template that can be modified to convert Function Blocks into other programming languages. In addition, other source codes and program libraries can be encapsulated into Function Blocks by either using a Service Interface Function Block (SIFB) to interface to a library or to include the source code into the basic Function Block. The source for such templates is similar to that for Function Blocks, including existing or previous projects, from vendors or from open source repositories.

ii. Modifying the Function Block: Like before, in some cases, the capabilities or performance of the existing function unit is not as was anticipated. Therefore, the algorithm must be modified and encapsulated into a Function Block according to the IEC 61499 standard. For the EBNF template, pre-definition of the target language platform may be necessary. For example, FPGA-based code will need to pre-define the input and output pins of the target chip in order to capture input and send results to the output.

iii. Saving in the repository: New and modified templates should be stored in a repository for repeated future use.

3. Tools and techniques

Reusable components can be found from some of the forums, service providers, previous projects or purchase from other vendors. The design repository can use simple file system or database system to store the information. For file system, difference reusable components can stored in different sub-directory. For database, need to create different user interface to store, change, describe the components.

4. Output

The output of this step is the Function Block components, external libraries, EBNF style sheet, template, reference design and documents.

9.4.4 Step D: Code generation

The objective of this step is to define the EBNF-based style sheet and to generate the target language source code. In this step, system designers are required to choose which part of the components should be using FPGA, microprocessor or a hybrid of FPGA and Microprocessor approaches. The code generator will be based on the EBNF-based style sheet and requirement specifications.

1. Input

At this step, the input will be the Function Block design, selected EBNF Style sheet for the target application language, pre- and post-condition definitions of the target application platform.

2. Process

The flow chart of Figure 9-15 presents the new system architecture for embedded applications tools system architecture. Steps A, B and C use IEC 61499 Function Block tools which will store the design in XML format. The system developer will then select and modify the EBNF style sheet and define the pre- and post-conditions within the style sheet for the target application platform. The parser will be based on the XML definition and style sheet and will generate the target source code. The objective of step E (described in the next section) is to prepare and synthesise the source code into an FPGA-based hardware embedded system. The objective of step F (described later) is to generate source code that is suited to a microprocessor-based embedded system. The following presents further details of the process.

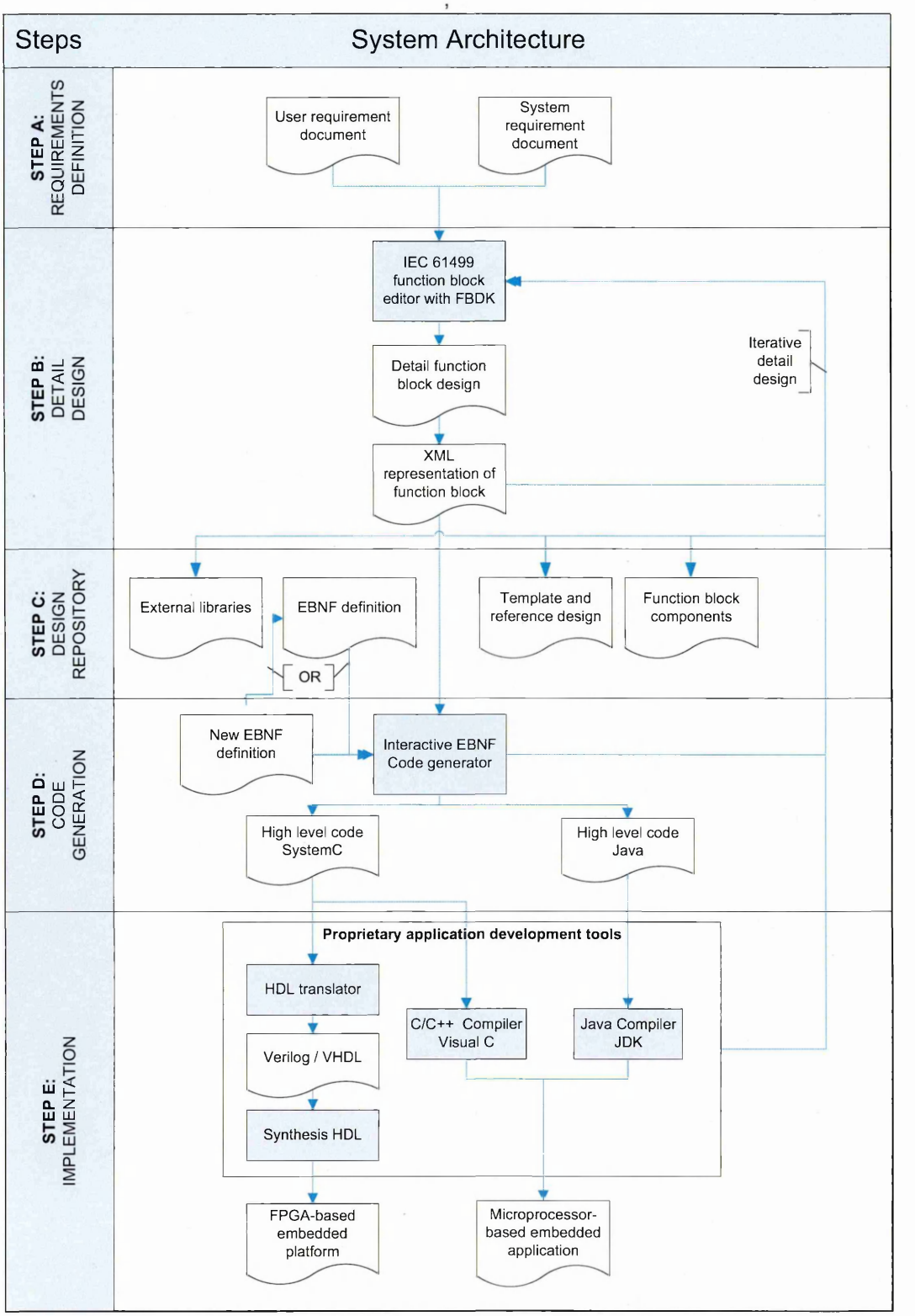


Figure 9-15: New system architecture

- Choose an embedded application platform: The selection of the embedded application platform will affect the selection of style sheets and the compilation process. It could be an FPGA-based reconfigurable hardware, microprocessor-based or a hybrid of both. The FPGA-based hardware uses VHDL, Verilog and SystemC languages, requiring pre- and post-condition definitions of the input and output pins. Microprocessors-based embedded platforms commonly use computer languages like Java, C, C++ or some proprietary languages.
- Search for the EBNF definition of the target language: There are numerous EBNF definitions available from various sources such as compiler reference manuals, computer language reference manuals, the web, compiler vendors etc.
- Match the XML to the target EBNF: The output of the IEC 61499 Function Block is XML. The developer needs to pre-define some of the input and output pins for an FPGA-based solution. As mentioned in Section 9.2.3, the Interactive Code Generator will then matches similar attributes and nodes and update it to the EBNF.

3. Tools and techniques

Generate C++ or Java source code through a Parser: As shown Figure 9-16 below, this step can be addressed by the Interactive Code Generator, which takes the XML and EBNF files as inputs and generates the target language source code.

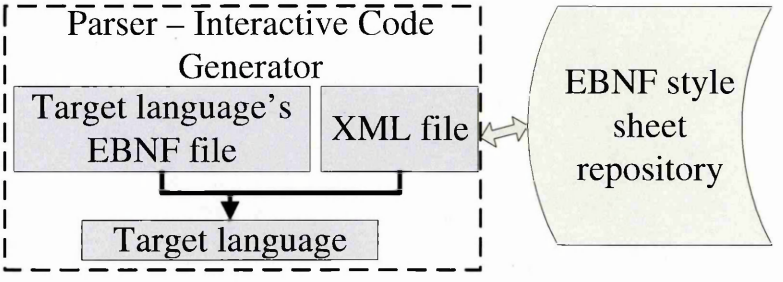


Figure 9-16: EBNF parser

4. Output

The output of this process is the generated target language source code. The new EBNF style sheet will be saved in the repository for future use.

9.4.5 Step E: Implementation for FPGA-based solution

The objective of this step is to implement source code generated by the code generator step for the FPGA-based reconfigurable hardware solution and use a specific FPGA compiler for debugging, simulation and synthesis.

1. Input

The input comprises the application source code generated by the code generator step.

2. Process

Figure 9-17 shows the process of generating FPGA code from XML. The system developer will either use an existing EBNF style sheet from the repository or develop a new EBNF one. The style sheet will include the pre- and post-conditions related to the FPGA hardware chip. For example, the allocation of the input and output pins of the FPGA chip should be pre-define manually in the EBNF style sheet. The parser will use the style sheet with the XML file to generate SystemC or other related source code for the FPGA. The follow section presents the details of the process.

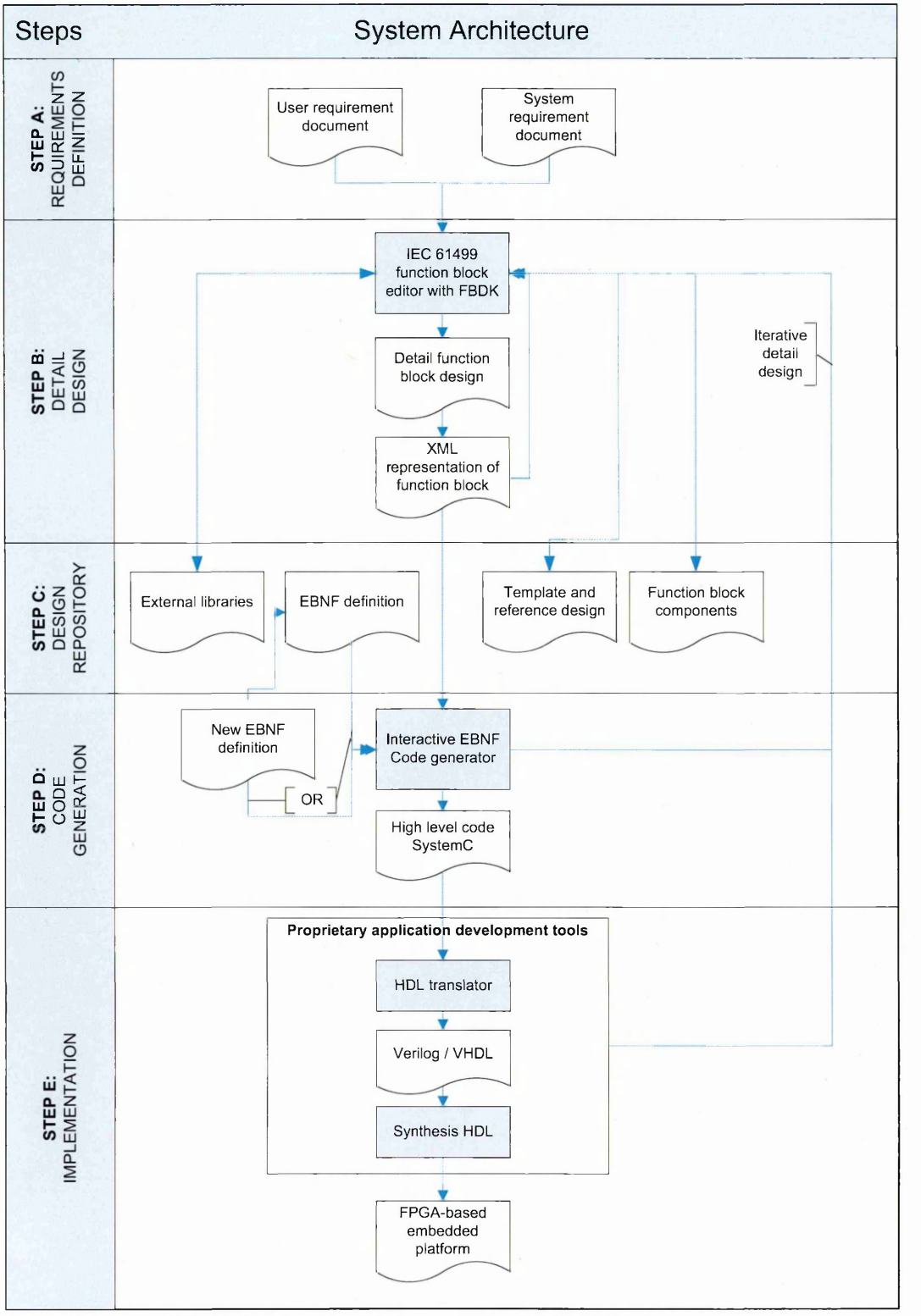


Figure 9-17: The process for FPGA code generation

- Compile the source code: The source code from step D needs to be compiled by a specific FPGA compiler. The compiler converts the source code into a bin file which can be used for synthesis by the FPGA gate.
- Debugging and simulation: The majority of FPGA compilers have debugging and simulation features. This step reduces errors and eliminates bugs from the code before it is used to configure the FPGA chip.
- Synthesis: FPGA compilers also generally have synthesis features, which use the bin file to configure the gates on the FPGA chip.
- Implement on hardware: Once the FPGA is synthesised with the required configuration, it can be placed in the PCB hardware for further testing or deployment.

3. Tools and techniques

The tools are mainly HDL translator, synthesiser, place and rounting software from various FPGA-based vendors.

4. Output

This step creates the FPGA chip that is configured with the application.

9.4.6 Step E: Implementation for microprocessor-based solution

The objective of this step is to implement the source code generated from the code generator step for a microprocessor-based solution, to use a specific microprocessor compiler for debugging, testing and to download the application to the microprocessor.

1. Input

As with the microprocessor-based solution, the input will be the generated application language such as C, C++ or Java source code.

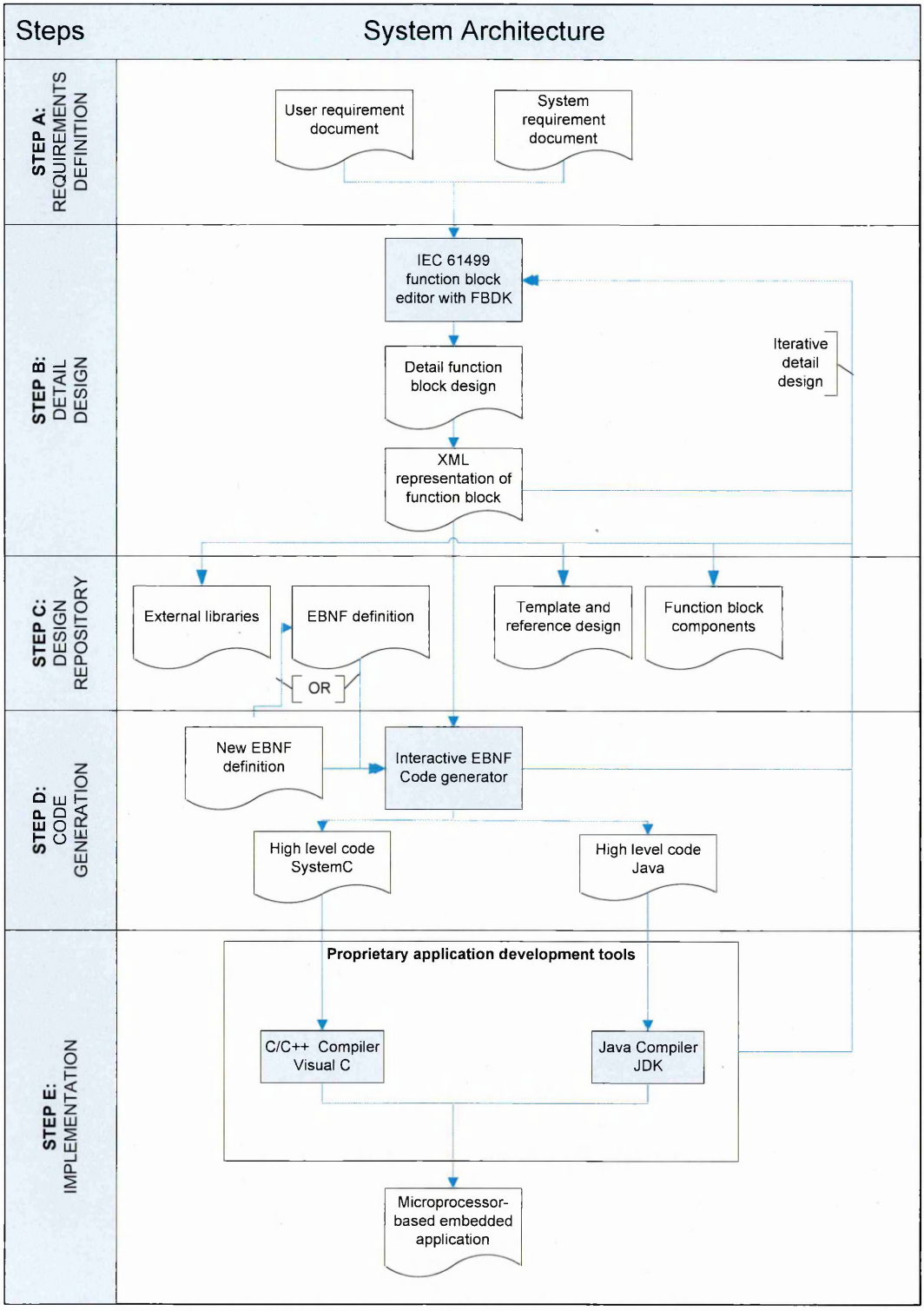


Figure 9-18: The process for microprocessor code generation

2. Process
   - Compile the program: The source code from step D must be compiled by the specific microprocessor compiler. The compiler will convert the source code into object code so that it can be downloaded to the microprocessor-based embedded system for testing.
   - Debugging and simulation: This is similar to PC-based applications in that the debugging and simulation features of the compiler can be used. Hardware-based debuggers or software-based microprocessor emulators can also be used for testing prior to downloading to the actual board. Most IEC 61499 tools have built in simulation packages for a PC environment.
   - Implement on hardware: After compilation, debugging and testing, the executable code will be downloaded to the microcontroller board. Different microcontroller chipsets have different tools and interface connectors for downloading executable code to the board. Some of the commonly used interfaces are JTAG, RS232 and network ports.

3. Tools and techniques

The tools are mainly C, C++ or Java compilers from various microprocessor vendors.

4. Output

This step creates the executable code for the microprocessor-based embedded application.

9.5 Chapter summary

This chapter has explained the improvements in the proposed methodology based on the refinements and feedback obtained from users at the workshop. Specifically, the refinements resulted in code generation improvements using EBNF and the creation of a guide for the final system development methodology for embedded applications. An overview of the methodology was presented, including a detailed description of each individual step. The methodology is open, descriptive and focuses on how to carry out the various steps of the methodology effectively from start to end. The next chapter concludes the research, summarising the contributions of the thesis and recommends interesting avenues of further research.

CHAPTER 10 CONCLUSIONS

This research envisaged the creation a new system development methodology for embedded applications. This final chapter summarises the overall thesis, highlighting the primary contributions and concludes the research. The chapter is organised as follows: Section 10.1 presents the research aims, objectives and programme, the research findings are given in Section 10.2 while the primary contributions of the thesis are provided in Section 10.3. In addition, the limitations of the proposed methodology are discussed in Section 10.4, with future research directions suggested in Section 10.4. Finally, the concluding remarks are given in Section 10.5.

10.1 Overview of research the aim, objective and programme

This section provides an overview of the research aim, objectives and programme. The research aim of this work was as follows:

"To develop a standard-based system development methodology for embedded applications"

The research aim was realised by sequentially executing the following five research objectives:

1. Review of the existing system development methodologies,
2. Formulate the conceptual methodology,
3. Select and develop of the proof-of-concept tool,
4. Evaluate and test the methodology,
5. Refine final methodology.

To achieve these objectives, a five stage structured research programme, developed in Section 4.3, was executed and has resulted in the formation of a system development methodology to support embedded applications.

- Stage 1: Existing system development methodologies related to embedded systems were reviewed from supporting literature to understand the strengths and weaknesses of each approach and identify area of further research (Chapter 5).
- Stage 2: The conceptual methodology was formulated based on the findings from stage 1 (Chapter 6).
- Stage 3: Based on the conceptual methodology, the proof-of-concept tool was developed to demonstrate the validity of the methodology through a validation experiment (Chapter 7).
- Stage 4: The methodology was further evaluated through three real-world experiments with industrial participants and users' feedback was obtained

from a workshop to identify particular characteristics for final refinement and improvement (Chapter 8).

- Stage 5: Results from the feedback were collated and used to refine the methodology. Thorough documentation and a guide for the final system development methodology for embedded applications were developed with a proof-of-concept tool (Chapter 9).

This thesis executed the aforementioned stages, encountering numerous challenges and new findings. These are examined in the next section.

10.2 Research findings

The following section summarises the findings from the research, together with general issues observed from the experiments.

1. A review of existing system development methodologies for embedded applications in chapter 3 indicated a lack of solutions for both FPGA-based and microprocessor-based embedded platforms. After the completion of this thesis, another literature search was done on 10 Sept 2009 based on the methodology developed. There is no research progress from other researchers in the areas of system development methodology for embedded application. The following issues are drawn:
    - The existing IEC 61499 based system development methodologies for embedded applications is lacking support to FPGA-based embedded platforms
    - The existing UML-based system development methodologies for embedded applications is lacking support to microprocessor-based embedded platforms
    - There is lack of detail given to code generation and implementation steps
    - There is a lack of interactive style sheet generators for various computer languages.
2. The research found that semiconductor manufacturing companies are very protective of their information. In the research validation process, three semiconductor manufacturing companies were willing to participate in the study; however, they were unwilling to provide detailed and accurate information about their system. This might be due to strong competition between semiconductor manufacturing sectors. All three companies desired to test the tools with very specific and simple experiments without disclosing any real production information. All the information provided has been simplified and desensitised by the companies. Once the companies tested

the key technology, they use their internal team to develop the full solutions without disclosing any further information.

3. The research found that the usefulness and usability of the methodology were promising. In Chapter 8, the perceptions of the workshop participants' with regards to the usability of the methodology were good, with an average score of 3.9 or 78%. The feedback on the usefulness of the methodology was 77%. Many of the participants indicated that the overall workshop and methodology was very successful and well worth participating.

4. The research found that a unified design for many embedded platforms is possible. The research has addressed some of the problems faced by industries, by providing a step-by-step methodology to capture the requirements, the structure, the design, the definition of style sheets and code generation for embedded applications. The methodology was well accepted by both software and non-software industrial participants during the workshop and industrial experiments. With this methodology, a unified design for many platforms is potentially realisable. Such a design could address some of the business challenges in keeping up with rapid technological advancements, reducing the cost of training, retaining knowledge and encouraging contribution of various engineers. The methodology proposed in this research addressed questions such as how to realise a common methodology which is acceptable by various engineers with reusable components.

10.3 Research contributions

The research presented in this thesis has provided numbers of contributions to the field of system development methodology for embedded applications. These contributions are next described.

Contribution 1: New international standard-based methodology

The research demonstrated that, although there is plentiful research in system development methodologies for embedded applications, there is little detailed research about how to exploit international standards to resolve some of the associated challenges. The purpose of the methodology and proof-of-concept tool developed in this thesis is to aid practitioners in simplifying and supporting embedded application development. This incorporates a series of international standards such as Function Blocks (IEC 61499), XML (ISO/IEC 8825), EBNF (ISO/IEC 14977) and SystemC (IEEE 1666-2005) standards. Their associated software techniques also provided a common approach to system development methodology for both microprocessor and FPGA-based embedded application platforms. The proof-of-concept tool was validated in industrial-based experiments (Section 8.3) and user feedback was obtained from a hands-on workshop. The findings from the experiments and feedback were as follows:

1) The overall methodology is based on IEC 61499 Function Blocks, can be easily understood by various stakeholders and is easy to use. It can also be used as tool for capturing user requirements.

2) The intermediate representation of the system design is through XML, which can be easily converted to different computer languages using an XSL parser.

3) In the creation of style sheets, improvements were introduced through the drag and drop EBNF tools to automate and simplify the process.

4) New techniques have been introduced to allow companies to consider leveraging the methodology for both existing and new embedded application development.

5) The final methodology was developed in the form of a guide with a computerised proof-of-concept software tool.

On this basis, the structured and procedural final system development methodology with its proof-of-concept tool created for embedded applications forms the primary research contribution of this thesis.

Contribution 2: Novel approach of extending IEC 61499 to both system design and embedded code generation

IEC 61499 was originally used for distributed PLC-based control whose the Function Blocks resemble the simple blocks and lines diagrams. In this research, IEC 61499 Function Blocks was used for high level user requirements capturing, system design and innovatively linking to the embedded code generation using XSL and EBNF (Section 7.3). The same methodology can be further extended to generate other applications and could simplify software development.

Contribution 3: Interactive EBNF parser with knowledge retention

EBNF grammar is normally used in creating token trees for compiler. In this research, EBNF was used for both the token tree of XML representation and target language generation. The interactive drag and drop feature allows for the interactive mapping of the original Function Block representation to the target language representation. The result of the conversion knowledge can be retained, reused and extended for future use.

Contribution 4: Conceptual methodology

The conceptual methodology comprises easy to understand systematic steps for future research directions, implementation and is not constrained by currently available tools.

The feedback from industrial participants showed that the conceptual methodology provided a clear and simplified view of the methodology. They could readily link the conceptual methodology to the logic and process flow. As for researchers and system developers, their interest is generally in the detail and they may have different preferences regarding implementation of the tools. Overall, the researchers and system developers find that a conceptual methodology gives them a clearer direction.

Contribution 5: In-depth knowledge of embedded application development practices

In this research, relevant literature on embedded application development was thoroughly reviewed. Numerous key findings were uncovered including the lack of a general consensus on the definition of embedded systems, their rapid growth and complexity, the merging of hardware and software in embedded systems. Furthermore, the reusability of software components and design patterns are preferred, embedded system research direction is fragmented; UML and Petri-net are not adopted as the common practice in most companies and in common evaluation criteria. The literature also identifies that industry practitioners are not satisfied with the existing system development methodologies for embedded applications. It was observed that current software engineering and system development methodologies for embedded applications are no longer adequate. The methodologies, techniques and tools are very different from those used in desktop environments and are in fact lagging behind their desktop counterparts [Paulin et al, 1997; Sangiovanni-Vincentelli & Martin, 2001; Paulin & Santana, 2002; Graaf et al, 2003; Luan et al, 2005; Mattos et al, 2005; Ibrahim et al, 2006; Hänninen et al, 2006].

Contribution 6: The state-of-the-art review

One of these contributions is a rigorous analysis of the state-of-the-art UML-based (Section 5.3), IEC 61499-based (Section 5.4) and other system development methodology research for embedded application development. This revealed that despite the considerable amount of work conducted, none of the methodologies uniformly addresses all criteria which are considered important by the collective body of knowledge (Section 3.2.8). The strengths, weaknesses and opportunities for further research were highlighted.

In summary, these findings provide a valuable foundation regarding the state-of-the-art in embedded application development practices. It also uncovers the criteria for a new system development methodology with code generators that could be readily understood by various stakeholders. These contributions further illustrate the impact of the research described in this thesis. The next section discusses the limitations of both the research content produced and the research process followed in this thesis.

10.4 Limitations of the research

Although the research has achieved the provision of system development methodology for embedded applications, there are a number of limitations worth mentioning. The nature of the design and implementation of the research programme gives rise to some limitation that could affect the findings of this research. The section begins by describing the limitations of the research programme, followed by problems within the participating companies and constrains with the small sample size of workshop. Finally, limitations of the methodology and the research findings are presented.

10.4.1 Limitations of the research programme

The research programme adopted in this thesis constructed a conceptual methodology, developed a proof-of-concept tool then observed and evaluated the system. It is believed that this programme is a novel approach to system development methodology design. To the author's knowledge, this form of research programme is newly developed for embedded applications research and therefore a number of limitations exist.

Firstly, the scope of the research carried out was very broad as it required knowledge in a number of subject areas including business, engineering and computer science. Significant effort was spent examining traditional literature to understand manufacturing industries in Singapore, analysing direction from both the Singaporean government and sponsor organisation (SIMTech), understanding the business challenges of the electronics sector, reviewing the embedded application development process, identifying areas of research, selecting the components, developing the proof-of-concept tool and testing through industrial related experiments. The software tool needed to be comprised of simple boxes and lines such that various stakeholders can understand. In addition, it was necessary to prepare a hands-on workshop to get feedback from industrial participants and prepare a simple guide to disseminate the methodology. All these requirements imposed a substantial workload over a relatively short time. The shortcomings were that, because of the breadth of the work and the time required designing, developing, verifying and validating the methodology and proof-of-concept tool, certain issues could not be researched and developed in depth. As a result, some of the findings are experimental and the results can illustrate only at proof-of-concept level. A major difficulty has been to achieve sufficient coverage of business, engineering and research issues but at the same time place sufficient focus on the completion of the research within the required timescale.

Secondly, the research programme adopted began with a review of the literature to examine the existing system development methodologies, which was then used as the foundation for designing the proposed conceptual methodology. However, the literature only shows the high-level design of

system methodologies and software tools to evaluate the methods are lacking. Therefore, there could be bias towards one methodology or misunderstanding of the existing methodologies. Nonetheless, through the design and development of the proof-of-concept tool, subsequent validation with industrial experiments and feedback from industries proved that the methodology is feasible, usable and useful. Thus, it is verified that the proposed system development methodology can be used as the foundation for future research.

10.4.2 Limitations of research findings

During the evaluation of the proof-of-concept tool, it was difficult to achieve full control of the industrial environment because the experiments required domain knowledge of the participating companies. For example in experiment 1, due to organisational changes, there was an unforeseen transfer of the engineer who helped in the polishing time estimator. Fortunately, the engineer still supported the experiment by long distance teleconference and emails. The other 2 companies that helped with experiment 2 and 3 did not want to disclose much industrial information and the assigned engineers from the companies were not available at times due to other requirements, which resulted in unforeseeable circumstances. To overcome these difficulties, it was required work with the engineers to test the proof-of-concept tool with certain algorithms, to simplify the setting and thus not aiming for the full application. The engineers studied the proposed tools and then confidentially refine them within their companies. The electronic cluster is also expansive and the experiment selected was based on companies' interests. Therefore, the trials did not cover the full range of requirements from the same cluster. However, the diversity of the verification and validation experiments was arranged to cover evaluation criteria and has provided useful verification for this research.

10.4.3 Sample size in the workshop

The level of generalisation assumed from the workshop can be said to be limited due to the lack of a large sample of participants. Ideally, feedback from a greater number of attendees or workshops should have been obtained. However, within the given time frame and limited available embedded hardware for hand-on session, this was not possible. The workshop was broadcast to SIMTech's email contacts in specific sectors meaning that detailed conclusions about any one particular industry could not be made. However, the diversity of the organisations and participants' experience provided very useful feedbacks and results.

This section has highlighted some of the research method limitations related to the programme and observations. Having recognised the limitations of this research, the findings observed are still valid. Within the context of the study, the methodology supports the activity of system development for embedded applications. Some of the limitations of the research programme provide

opportunities for future research. The next section considers such possible avenues of future research work.

10.5 Directions for future research

In the course of addressing the research aim and objectives, other gaps arise which expose areas that require future investigation. As discussed in the previous section, this section identifies the direction that future work should take to support the progress of the research in this area. It starts by outlining opportunities for enhancing the system development methodology for embedded applications, then direction for further testing and finally gives concluding remarks.

10.5.1 Opportunities for enhancing the proposed system development methodology for embedded applications

The system development methodology for embedded applications developed by this research has proved usable but is still far from a fully functional tool and would benefit from further refinements.

- The proposed system development methodology for embedded applications provides a proof-of-concept tool and guide and is far from a fully functional completed tool. Further developments are necessary to enhance the collection of style sheets as well as integrate some of the tools into a fully functional environment.
- The current proof-of-concept tool can be enhanced through code optimisation and software verification so that the generated code can be optimised to the target embedded platform. This allows for the performance objectives to be met and guarantees that the performance of the generate code is as expected in the design and simulation.
- There are still research issues in the reversal of source code into a system design. This could be appealing to a company that already has systems developed in other languages and wish to reverse the source code into the system design and then deploy it in other platforms with other languages.

10.5.2 Opportunities for further testing

The assessment of the system development methodology for embedded applications formed a major segment of the research. The study suggests however that the methodology should be further evaluated with more embedded platforms. This would extend the generalisation of the methodology and tools. During the research, only four experiments were carried out. Only limited evaluation was possible within the timeframe to understand the usefulness of the methodology. Further evaluation and testing of the

methodology would provide a deeper understanding and may lead to further refinements.

10.6 Concluding remarks

This chapter has summarised the main research issues, the research aims and discussed major contributions. The limitations of the research have been identified and numerous recommendations for future research were suggested. The research has made novel contributions to system development methodologies for embedded application. It is hoped that the main contributions of this thesis will have similar impact in future industrial practices.

REFERENCES

1. 4DIAC, (2009). 4DIAC – Framework for Distributed Industrial Automation and Control, available at: www.fordiac.org. Date accessed 4 November 2009.

2. Aelfred, (2009). Aelfred Parser, available at: saxon.sourceforge.net/aelfred.html. Date accessed 4 November 2009.

3. Agility, (2009). Agility Compiler, available at: http://agilityds.com/. Date accessed 4 November 2009.

4. Altera, (2003). Standard Cell ASIC to FPGA Design Methodology and Guidelines, July 2003, ver 1.0. available at: www.altera.com/literature/an/an311.pdf. Date accessed 4 November 2009.

5. Alur R., Ivancic F., Kim J., Lee I., Sokolsky O., (2003). Generating Embedded Software from Hierarchical Hybrid Models, Special Issue: Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool support for embedded systems - LCTES'03, San Diego, California, USA, pp. 171 – 182.

6. Arisholm E., Sjoberg, (2004). Evaluating the Effect of a Delegated versus Centralised Control Style on the Maintainability of Object-oriented Software, IEEE Transactions on Software Engineering, Vol: 30, Issue: 8, pp. 521- 534.

7. Aslam N., (2009). A Low-cost Solution for FPGA-based PCI Express Implementation, Altera Corp, available at: http://www.design-reuse.com/articles/13258/a-low-cost-solution-for-fpga-based-pci-express-implementation.html. Date accessed 4 November 2009.

8. Auinger F., Brennan R., Christensen J., Martinez Lastra J. L., Vyatkin V., (2005). Requirements and Solutions to Software Encapsulation and Engineering in Next Generation Manufacturing Systems: OOONEIDA Approach, International Journal Computer Integrated Manufacturing, Vol. 18, pp. 572–85.

9. Baines T., (1995). Modelling in the Evaluation of a Manufacturing Strategy. Ph.D. thesis, Cranfield University.

10. Balasubramanian K., Balasubramanian J., Parsons J., Gokhale A., Schmidt D., (2007). A Platform-Independent Component Modeling Language for Distributed Real-time and Embedded Systems, Journal of Computer and System Sciences, Vol. 73 , pp.171–185.

11. Baleani M., Ferrari A., Mangeruca L., Sangiovanni-Vincentelli A.L., Freund U., Schlenker E., Wolff H.J., (2005). Correct-by-Construction Transformations across Design Environments for Model-Based Embedded Software Development, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition - DATE'05, Vol. 2, pp. 1044 - 1049,

12. Barji A., Hagge N., Wagner B., (2006). Comparative Study of Using CNet, IEC 61499, and StateCharts for Behavioural Models of Real-time Control Applications, IEEE Conference on Emerging Technologies and Factory Automation, 2006 - ETFA '06, pp. 750-757.

13. Barr M., (1990). Programming Embedded Systems in C and C++, O'Reilly, first edition.

14. Basili V. R., Selby R. W., Hutchens D. H., (1986). Experimentation in Software Engineering, IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, pp. 733-743.

15. BCC (2000). Future of Embedded Systems Technology, The BCC Research Group.

16. Bertolott I. C., (2006). Real-Time Embedded Operating Systems: Standards and Perspectives, Embedded Systems Handbook, pp. 11:1-33, CRC Press.

17. Bichler M., Lin K. J., (2006). Service-Oriented Computing, IEEE Computer Vol. 39, Issue. 3, pp. 99 – 101.

18. Bjärnemyr D., Dahlberg J., (2004). The Strategic Role of Remote Embedded Systems, Master Thesis, Handelshögskolan Vid Goteborgs Universitet.

19. Blanchard B. S. and Fabrycky W. J., (1981). Systems Engineering and Analysis, Englewood Cliffs, NJ: Prentice-Hall, Inc.

20. BLS, (2007). Computer and Electronic Product Manufacturing, The 2008-09 Career Guide to Industries, Bureau of Labor Statistics (BLS), U.S. Department of Labor.

21. Bouyssounouse B., Sifakis J., (2005). Embedded Systems Design, Published by Springer.

22. Boudour R., Kimour M. T., (2006). From Design Specification to SystemC, Department of Computer Science, University of Annaba, Journal of Computer Science 2. pp. 201 – 204.

23. Brennan R., Wang J., Norrie D., (2004). A Pattern-based Approach to Distributed Control System Design, The Inaugural CDEN Design Conference, McGill University and École Polytechnique, in Montréal,.

24. Brennan R., (2007). Toward Real-Time Distributed Intelligent Control: A Survey of Research Themes and Applications, IEEE Transactions on System, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 37, No. 5, pp. 744-765.

25. Brisolara L., Becker L. B., Carro L., Wagner F. R., Pereira C. E., (2004). Evaluating High-level Models for Real Time Embedded System design, Design Methods and Applications for Distributed Embedded Systems Conference 2004.

26. Brunel J.Y., Di Natale M., Ferrari A., Giusto P., Lavagno L., (2004). SoftContract: an Assertion-Based Software Development Process that Enables Design-by-Contract, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition -DATE'04, Vol.1, pp. 358- 363..

27. Calyez J. P., (1993). Embedded Real-time Systems: A Specification and Design Methodology, Wiley Series in Software Engineering Practice, John Wiley & Sons.

28. Calyez J. P., Pasquier O., (1998). Performance Monitoring and Assessment of Embedded HW/SW Systems, Design Automation for Embedded Systems, Vol. 3, No. 1 , pp. 5-22. Springer Netherlands.

29. Cengic G., Ljungkrantz O., Åkesson K., (2006). A Framework for Component-based Distributed Control Software Development Using IEC 61499, Proceedings of the 2006 IEEE International Conference on Emerging Technology and Factory Automation, Prague, Czech Republic.

30. Chase W.P., (1974). Management of System Engineering. John Wiley and Sons.

31. Coyle F. P., Thornton M. A., (2005). From UML to HDL: a Model Driven Architectural Approach to Hardware-Software Co-Design, Information Systems: New Generations Conference - ISNG, pp. 88-93.

32. Curtin R., (1990). ASIC design methods using VHDL, Euro ASIC '90, pp. 176-179

33. Damaševičius R., Majauskas G., Štuikys V., (2003). Application of Design Patterns for Hardware Design, Annual ACM IEEE Design Automation Conference, Proceedings of the 40th conference on Design automation - DAC 2003, pp. 48 – 53, .

34. Denning, P.J., Comer, D.E., Gries, D., Mulder, M.C., Tucker, A., Turner, A.J., and Young, P.R. (1988). Draft Report of the ACM Task Force on the Core of Computer Science, DRAFT 3.

35. Dubinin V., Vyatkin V., (2005). Engineering of Validatable Automation Systems Based on an Extension of UML Combined With Function Blocks of IEC 61499, Proceedings of the 2005 IEEE, International Conference on Robotics and Automation, Barcelona, Spain, pp. 3996-4001..

36. Dubinin V., Vyatkin V., (2006a). Towards a Formal Semantic Model of IEC 61499 Function Blocks, in 4th IEEE International Conference on Industrial Informatics (INDIN), Singapore. pp. 6-11.

37. Dubinin V., Vyatkin V., Hanisch H.M., (2006b). Modelling and Verification of IEC 61499 Applications using Prolog, 11th IEEE Conference on Emerging Technologies and Factory Automation, Prague. pp. 774-781.

38. EDB, (2005). Singapore Comes of Age, Economic Development Board (EDB) of Singapore, 01 Aug 2005.

39. EDB, (2008). Why Singapore – Manufacturing Excellent, Economic Development Board (EDB) of Singapore.

40. Ellis D.O.,Ludwig, F.J., (1962). Systems Philosophy, Prentice-Hall..

41. FBench, (2009). FBench project – Open Tool for IEC 61499 Function block Engineering, available at: www.ece.auckland.ac.nz/~vyatkin/fbench/. Date accessed 4 November 2009.

42. Fischer J., Boucher T. O., (2008). Workbook for Designing Distributed Control Applications using Rockwell Automation's HOLOBLOC Prototyping Software, Working paper, pp. 5-17.

43. Forte, (2009). Forte Cynthesizer, available at: www.forteds.com/products/cynthesizer.asp. Date accessed 4 November 2009.

44.

45. Fowler M., (2004). UML distilled: A Brief Guide to the Standard Object, Addison-Wesley.

46. GCC, (2009). GCC-XML, available at: www.gccxml.org/HTML/Index.html. Date accessed 4 November 2009.

47. GentleWare, (2009). Poseidon for UML, available at: www.gentleware.com/products.html. Date accessed 4 November 2009.

48. Goh K. M., Tjahjono B., Baines T., Subramaniam S.. (2006). A Review of Manufacturing Prognostic Research, 4th IEEE International Conference on Industrial Informatics 2006 - INDIN 06, Singapore, pp. 417-422.

49. Goh K. M., Tjahjono B., Aendenroomer A. J. R.. (2007). A Rapid Configurable Embedded Development Methodology, IEEE International Conference on Emerging Technologies and Factory Automation - ETFA, Patras - Greece, Vol. 1, pp. 135-140.

50. Goh K.M., Tjahjono B., Baines T., Ang J., (2008). Literature Review of Research in Manufacturing Prognostics, Cranfield Multi-Strand Conference: Creating Wealth Through research and Innovation - CMC 2008,

51. Goh K.M., Tjahjono B., Manaf A,, (2008). An IEC 61499-based Run-to-run Controller for Chemical Mechanical Planarisation Process, 13th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 08, Hamburg, Germany,

52. Goh K.M., Tjahjono B., Joseph J., (2008). Automation Objects for Product Service System Enabler Device, 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 08), Hamburg, Germany, pp. 50-53.

53. Goh K.M., Tjahjono B., Ding W., (2009). Iterative Knowledge Based Code Generator for IEC61499 Function Block, TENCON 2009.

54. Gomes L., Barros J. P., Costa A., (2004). Structuring Mechanisms in Petri Net Models: From specification to FPGA based implementations. Design of embedded control systems, Kluwer.

55. Graaf B., Lormans M., Toetenel H., (2002). Software Technologies for Embedded System: An Industry Inventory, Proceedings of the 4th International Conference on Product Focused Software Process Improvement - PROFES 2002, Rovanieni, Springer. pp. 453-455.

56. Graaf B., Lormans M., Toetenel H., (2003). Embedded Software Engineering: The State of the Practice, IEEE Computer Society IEEE Software, Vol. 20, Issue. 6, pp. 61- 69.

57. Graaf B., van Dijk H., van Deursen A., (2005). Evaluating an Embedded Software Reference Architecture — Industrial Experience Report, SMR 005. Ninth European Conference on Software Maintenance and Reengineering, pp. 354 – 363.

58. Gregg D. G., Kulkarni U. R. and Vinzé A. S., (2001). Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems, Information Systems Frontiers, Vol. 3, No. 2.

59. Hagge N., Wagner B., (2005). A New Function Block Modelling Language Based on Petri Nets for Automatic Code Generation, IEEE Transactions on Industrial Informatics, Vol. 1, No. 4, pp. 226-237.

60. Halblander M. F., (2004). Requirements Engineering for Software Systems, Master Thesis, Northeastern Illinois University.

61. Hall A.D., (1962). A Methodology for Systems Engineering, New York, Van Nostrand.

62. Hänninen K., Mäki-Turja J., Nolin M., (2006). Present and Future Requirements in Developing Industrial Embedded Real-Time Systems - Interviews with Designers in the Vehicle Domain, Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems - ECBS'06, pp. 139 – 150.

63. Harel D., (1987). StateCharts: A Visual Formalism for Complex Systems, Science of Computer Programming, Vol. 8, No. 3, pp. 231-274.

64. Heng C. H., (2005). Launch of the Embedded Systems Community, Singapore Ministry of Trade and Industry and National Development.

65. Heverhagen T., Tracht R., Hirschfeld R., (2003). A Profile for Integrating Function Blocks into the Unified Modelling Language, Software Tools for Technology Transfer Talk at Specification and Validation of UML Models for Real Time Embedded Systems - SVERTS'03.

66. Holobloc, (2008). Holobloc FBDK, available at: www.holobloc.com, Date accessed 4 November 2009.

67. Huff W. G., (1995). What is the Singapore Model of Economic Development?, University of Glasgow and Australian National University, Cambridge Journal of Economics.

68. Ibrahim A.E., Zhao L., Kinghorn, J., (2006). Embedded Systems Development: Quest for Productivity and Reliability, Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, pp.10-20.

69. IEC, (2005). International Electrotechnical Commission, Function blocks—Part 1: Architecture, International Electrotechnical Commission & Part 2: Software Tools Requirements, Geneva, Switzerland, Tech. Rep. IEC 61499-1 & 2.

70. ISO/IEC, (1996). International Standard Organisation (ISO) / International Electrotechnical Commission (IEC) standard 14977, International Electrotechnical Commission, Geneva, Switzerland, ISO/IEC 14977.

71. Jeon S., Hong J., Bae D., (2006). Using Operational Architecture to Model Embedded Software, 13th Asia Pacific Software Engineering Conference - APSEC'06, pp. 157-164.

72. Kangas T., Kukkala P., Orsila H., Salminen E., Hännikäinen M., Hämäläinen T. D., Riihimäki J. and Kuusilinna K., (2006). UML-based Multiprocessor SoC Design Framework, ACM Transactions on Embedded Computing Systems, May 1, 2006, Vol. 5, Issue 2, pp. 281-320.

73. Karsai G., Sztipanovits J., Ledeczi A., Bapty T., (2003). Model-Integrated Development of Embedded Software, Proceedings of the IEEE, Vol. 91, No. 1.

74. Kelly K., Kratky J., Wells K., (2003). Model Driven Compound Document Development, Xtech05 Conference Processing, January 4, 2003, available at: www.idealliance.org/proceedings/xtech05/papers/04-01-03/. Date accessed 4 November 2009.

75. Khalgui M., Rebeuf X., Simonot-Lion F., (2004). A Behavior Model for IEC 61499 Function Blocks, Third Workshop on Modelling of Objects, Components and Agents 2004 - MOCA'04, pp. 71-87.

76. Kippenberger T., (1999). Wrestling with Embedded Systems, The Antidote, Vol. 4, No. 2, pp. 20-23.

77. Kitchenham B., (1996). Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods, ACM Sigsoft, Software Engineering Notes, Vol. 21, No. 1. pp. 11-15.

78. Klingman Ed., (2006). FPGA Programming Step by Step, available at: www.embedded.com/showArticle.jhtml?articleID=18201956. Date accessed 4 November 2009.

79. Ko K. H., (2005). Singapore Comes of Age, Economic Development Board (EDB) of Singapore, August 2005.

80. Kornecki A. J., (1999). Teaching Real-Time Concepts to Software Engineers, Department of Computing and Mathematics, Embry Riddle Aeronautical University. available at: faculty.erau.edu/korn/presentations/rt_berlin/sld006.htm. Date accessed 4 November 2009.

81. Kumar S., Aylor J. H., Johnson B. W., et al, (1996). The Codesign of Embedded Systems: A Unified Hardware/Software Representation. Kluwer Academic Publishers.

82. Kuvaja P., Maansaari J., Seppanen V., et al, (1999). Specific Requirements for Assessing Embedded Product Development., International Conference on product Focused Software Process Improvement, pp. 68-85, VTT Electronics and University of Oulu, Finland.

83. Lau D. J., Pritchard O., (2006). Rapid System-on-programmable-chip Development and Hardware Acceleration of ANSI C Functions, International Conference on Field Programmable Logic and Applications, FPL '06, pp. 1-6.

84. Lee J. M., (2002). Verilog Quickstart: A practical Guide to Simulation and Synthesis in Verilog, Third Edition, Kluwer Academic Publishers.

85. Lee C. S., (2008). Singapore's Hottest Engineering Sectors, Job Center Community, available at: community.jobscentral.com.sg/node/256. Date accessed 4 November 2009.

86. Luan X., Ying J., Wu M., (2005). A Heterogeneous Evolutional Architecture for Embedded Software, Proceedings of the 2005. The Fifth International Conference on Computer and Information Technology -CIT'05, pp. 901-905.

87. Lewis R., (2001). Modelling Control Systems Using IEC 61499, The Institution of Electrical Engineers, London, ISBN 0-85296-796-9.

88. Lindlar F., Zimmermann A., (2008). A Code Generation Tool for Embedded Automotive Systems Based on Finite State Machines, Industrial Informatics, pp. 1539 – 1544.

89. Lim S., (2007). Microelectronic and Embedded System in Singapore, Microelectronics & Embedded Systems Network in Singapore, The Institution of Engineering and Technology.

90. Lopez J. C., Hermida R., Geisselhardt W., (1998). Embedded Systems Design and Test, Kluwer Academic Publishers.

91. Lu S., Halang W., Zhang L., (2005). A Component-based UML Profile to Model Embedded Real-Time Systems Designed by the MDA Approach, Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications - RTCSA'05.

92. Ma H., Yen I., Zhou J., Cooper K., (2006). QoS analysis for component-based embedded software: Model and methodology, The Journal of Systems and Software, pp. 859–870.

93. Martin G., (2002). UML for Embedded Systems Specification and Design: Motivation and Overview, Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition - DATE.02, pp. 773- 775.

94. Marian N., Guo Y., (2006). Model-Based Design of Embedded Software, Research and Education in Mechatronics - REM 2006, Stockholm, Sweden.

95. MAS, (2007). Macroeconomic Review, Economic Policy Department, Monetary Authority of Singapore (MAS), Vol. 6, Issue 2.

96. Mattos J., Specht E., Neves B., Carro L., (2005). Making Object-oriented Efficient for Embedded System Applications, SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design.

97. Matusiak R., (2001). Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences with the TMS320 DSP Platform, Application Report - SPRA291.

98. Megginson, (2008). Megginson Parser, available at: www.megginson.com. Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences with the TMS320 DSP Platform

99. Microsoft, (2008). Microsoft's DOM Parser, available at: msdn.microsoft.com/xml/. Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences with the TMS320 DSP Platform

100. Minds, Kevin S., (1995). System Engineering The People System, The Journal of International Council Systems Engineering - INCOSE.

101. MOF, (2002). MetaObjectFacility (MOF) Specification Version 1.4, Object Management Group, Inc. (OMG), April 2002, www.omg.org/mof/. Date accessed 4 November 2009.

102. Morton M. S., (1984). The State of Art of Research, in McFarland F. W. (ed.), The Information Systems Research Challenge, Harvard Business School Press, pp. 12-41.

103. Mura M., Paolieri M., Negri L., Sami M. G., (2007). StateCharts to SystemC: A High Level Hardware Simulation Approach, Proceedings of the 17th Great lakes symposium on VLSI, Stresa-Lago Maggiore, Italy, pp. 505 – 508.

104. Mustapic G., Wall A., Norström C., Crnkovic I., Sandström K., Fröberg J., Andersson J., (2004). Real World Influences on Software Architecture - Interviews with Industrial System Experts, Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture - WICSA'04.

105. Nguyen K.D., Thiagarajan P.S., Wong W.F., (2007). A UML-based Design Methodology for Time-triggered Applications. 28th IEEE Real-Time Systems Symposium (RTSS 07). Tucson, Arizona, U.S.A..

106. Noergaard T., (2005). Embedded System Architecture – A Comprehensive Guide for Engineers and programmers, Elsevier.

107. Nunamaker J. F., Chen M., (1989). Software productivity: A Methodology of Study and Approach to Reusable Components, The 22nd International Conference on System Sciences, Hawaii.

108. Nunamaker, J. F., Chen, M., Purdin, T. D. M., (1991). Systems Development in Information Systems Research. Journal of Management Information Systems. Vol. 7, No. 3, pp. 89-106.

109. Nunes R., Gomes L., Barros, (2007). A Graphical Editor for the Input-Output Place-Transition Petri Net Class, 12th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA07, pp. 788-791.

110. Oliveira M, F. S. O., Lisane B. B., Luigi Carro L., Wagner F. R., (2006). Early Embedded Software Design Space Exploration Using UML-based Estimation, Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping - RSP'06, pp. 24-32.

111. Paulin P. G., Liem C., Cornero M., Fran ¸ Cois N.¸ Cabal, Goossens G., (1997). Embedded Software in Real-Time Signal Processing Systems: Application and Architecture Trends, Proceedings of the IEEE, Vol. 85, No. 3, Pp. 417-435.

112. Paulin P. G., Santana M., (2002). FlexWare: A Retargetable Embedded-Software Development Environment, Design & Test of Computers, IEEE, Vol. 19, Issue: 4, pp. 59-69.

113. Pereira C., Carro L., (2007). Distributed Real-time Embedded Systems: Recent Advances, Future Trends and Their Impact on Manufacturing Plant Control, Annual Reviews in Control 31, pp. 81–92, Elsevier.

114. Posadas H., Herrera F., FernÁndez V., SÁnchez P., Villar E., (2004). Single Source Design Environment for Embedded Systems Based on

SystemC, Design Automation for Embedded Systems, pp. 293–312, Springer.

115. Pressman R S., (1997). Software Engineering a Practitioner's Approach. McGraw-Hill.

116. Rechtin E., (1991). Systems Architecting: Creating and Building Complex Systems, Prentice-Hall, 1991

117. Rechtin E., Maier M., (1997). The Art of System Architecting, CRC Press, 1997

118. Relax, (2003). RELAX NG home page, www.relaxing.org. Date accessed 4 November 2009.

119. Rinćon F., Moya F., Barba J., López J., (2005). Model Reuse through Hardware Design Patterns, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition - DATE'05.

120. Robson C., (2004). Real World Research. 2 edn. Blackwell Publishing, Oxford.

121. Rose, (2007). Rational Rose RT, available at: www.ghs.com/partners/rational/rose-rt.pdf. Date accessed 4 November 2009.

122. S&T Plan, (2006). Science and Technology Plan 2010, Ministry of Trade and Industry of Singapore, ISBN: 981-05-5335-8.

123. Salewski F., Taylor A., (2007). Fault Handling in FPGAs and Microcontrollers in Safety-Critical Embedded Applications: A Comparative Survey, 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools - DSD 2007, pp. 124-131.

124. Sangiovanni-Vincentelli A., Martin G., (2001). Platform-Based Design and Software Design Methodology for Embedded Systems, Design & Test of Computers, IEEE, Vol. 18, Issue. 6, pp. 23-33.

125. SC2V, (2008). SystemC to Verilog Converter, available at: www.opencores.org/projects.cgi/web/sc2v. Date accessed 4 November 2009.

126. Schematron, (2006). Information technology — Document Schema Definition Languages (DSDL) —Part 3: Rule-based validation — Schematron, International Standard ISO/IEC 19757-3, First edition.

127. Simsie L., (2003). Using Programmable Logic for Embedded Systems, Altera Corporation.

128. Splunter J., (2002). The Pursuit of Competitive Advantage Value Manufacturing in Singapore, Sub-committee on Manufacturing, Economic Review Committee Report, Ministry of Trade and Industry of Singapore.

129. Synopsis, (2008). Synopsis CoCentric SystemC Compiler, available at: www.synopsys.com/products/cocentric_systemC/cocentric_systemC_ds.html. Date accessed 4 November 2009.

130. Tan W.H., Thiagarajan P.S., Wong W.F., Zhu Y., Pilakkat S.K., (2004). Synthesisable SystemC Code from UML Models, Presented at International Workshop on UML for SoC Design (USOC 2004). Sponsored by Design Automation Conference.

131. Talavera G., Nollet V., Mignolet J-Y, Verkest D., Vernalde S., Lauwereins R., Carrabina J., (2004). Hardware-Software Debugging Techniques for Reconfigurable Systems-on-Chip, IEEE International Conference on Industrial Technology - ICIT, Yasmine-Hammamet, Tunisia.

132. Thamboulidis K., (2005). Model-integrated Mechatronics - Toward a New Paradigm in the Development of Manufacturing Systems, IEEE Transactions on Industrial Informatics, Vol. 1 Issues 1, pp. 54- 61.

133. Thamboulidis K., (2006). IEC 61499 in Factory Automation, Advances in Computer, Information and Systems Sciences and Engineering, Springer Netherlands, pp. 115-124.

134. Thamboulidis K., Doukas G., Koumoutsos G., (2007). Device Modelling for a Flexible Embedded Systems Development Process, Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing -ISORC'07.

135. Tilley K.J., Williams D.J., (1997). An Assessment of the Electronics Industry in Southeast Asia, IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C, Vol. 20, Issue 4, pp. 230 – 234.

136. Underseth M., (2007). Addressing the Hidden Embedded Software Crisis in the Industry, EETimes Asia.

137. W3C, (2001). Semantic Web, available at: www.w3.org/2001/sw/. Date accessed 4 November 2009.

138. Wang Y., Zhou X. G., Zhou B., Liang L., Peng C. L., (2006). A MDA based SoC Modeling Approach using UML and SystemC, Proceedings of The Sixth IEEE International Conference on Computer and Information Technology - CIT'06.

139. WDA, (2006). Electronics Skills Report 2005/2006, Singapore Workforce Development Agency.

140. Weehuizen F., Brown A., Sunder C., Hummer O., (2007). Implementing IEC 61499 Communication with the CIP Protocol, Emerging Technologies & Factory Automation, 2007. ETFA. IEEE Conference on Publication, pp. 498-501.

141. M. Wenger et al, (2008), Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector State-of-the-Art and Technology Review Report, The MEDEIA Consortium, http://www.medeia.eu/fileadmin/Deliverables/DL-MEDEIA_D21-StateOfTheArtAndTechnologyReviewReport_V005_2008-07-10.pdf. Date accessed 20 November 2009.

142. Wolf W., (2000). What and Why About Architecture for Embedded. Workshop on Computer Architecture Education - WCAE 2000.

143. Woll D., (2007). Setting the Stage for the Next Generation of Automation Control System Software: A Discussion of IEC 61499, ARC Insights, Insight 2007-13M.

144. Wong P. K., (2002). Globalisation of US, European and Japanese Production Networks and the Growth of Singapore's Electronics Industry, International Journal of Networking and Virtual Organisations, Vol. 1, No. 1, pp. 55-81.

145. Wurmus H., (2002). CNet – Komponentenbasierter Entwurf verteilter Steuerungssysteme mit Petri-Netzen, Institute for Systems Engineering, Hannover, Germany.

146. Valentini A., Martel A., Ferroni B., Martinez Lastra J., Brennan R. W., Christensen J. H., Vyatkin V., Brown G., (2006). O3NEIDAactivity report, O3NEIDA. Ottawa, ON, Canada, Tech. Rep. 04-2006.

147. Vyatkin V., Christensen J., Lastra J., (2005a). OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Industrial Automation, IEEE Transactions on Industrial Informatics, Vol. 1, Issues 1, pp. 4-17.

148. Vyatkin V., Karras S., Pfeiffer T., (2005b). Architecture for Automation System Development Based on IEC61499 Standard, 2005 3rd IEEE International Conference on Industrial Informatics - INDIN 2005.

149. Yang P., (1997). New chemical Mechanical Planarisation (CMP) end Point Detection Apparatus, United States Patent 5667424.

150. Yen I.L., Goluguri J., Bastani F., Khan L., (2002). A Component-based Approach for Embedded Software Development, Proceedings of the

Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing - ISORC.02.

151. Yeo P., (2008). New SPRING Programme to Strengthen Semiconductor Capabilities, SPRING Singapore, www. spring.gov.sg. Date accessed 4 November 2009.

152. Zhou J., Cooper K., Yen I., Linn J., Paul R., (2006). A Software Enhancement System for Embedded Software Development, Proceedings of the Ninth IEEE International Symposium on Object and Component-oriented Real-time Distributed Computing - ISORC 2006.

153. Zoitl A., Strasser T., Hall K., Staron R., Sünder C., Favre-Bulle B., (2007). The Past, Present and Future of IEC 61499, Lecture Notes in Computer Science: Holonic and Multi-Agent Systems for Manufacturing, Vol. 4659/2007, Springer Berlin, pp. 1-14.

154. Zelkowitz M.V., Wallace D.R., (1998). Experimental Models for Validating Technology, Computer, Vol. 31, Issue 5, pp. 23 – 31.

155. Zurawski R., (2006). Embedded Systems Handbook Published 2006,CRC Press.

APPENDIX A: INTRODUCTION TO FUNCTION BLOCK PROGRAMMING

This document was prepared for the purpose of introducing the IEC 61499 function block programming tool to beginners. The presentation introduces the software interface as well as detailed steps involved in the creation of a simple program, based on a sample problem. To simplify the presentation, the program requires only basic functionality so as not to overburden first-time users.

1. Introduction

An introduction to function block programming is created for the purpose of providing a guide for first-time and novice users.

2. Objective

To provide a clear overview and functionality guide for first time users of function blocks with the Function Block Development Kit (FBDK). The guide should be comprehensible to users with a little programming background or experience

3. Scope

i. To provide a short introduction on the FBDK interface, including the basic commands and controls.

ii. To fully illustrate and present the detailed steps required to solve a sample problem using FBDK.

4. FBDK Interface and commands

The interface for the main commands has numerous similarities with a typical windows application interface. The interface for FBDK allows commands to be selected and picked through the use of a mouse or by hotkeys. Figure A-1 below shows the descriptions of the various basic functions for FBDK.

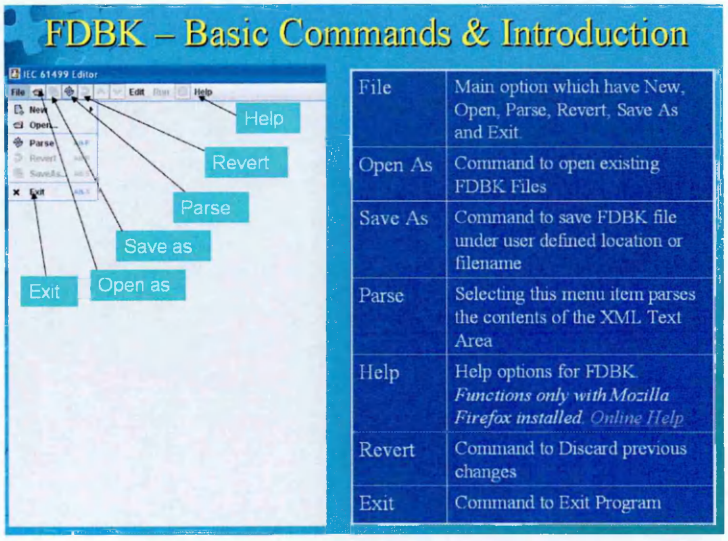


Figure A-1: An introduction to the basic commands in FBDK.

Other than the basic commands of creating a new file, opening or saving an existing file, it has revert functions that allows the undoing of recent modifications. An internet browser is required to access FBDK's web-based help function, similar to the help provided by, for instance, Microsoft Office.

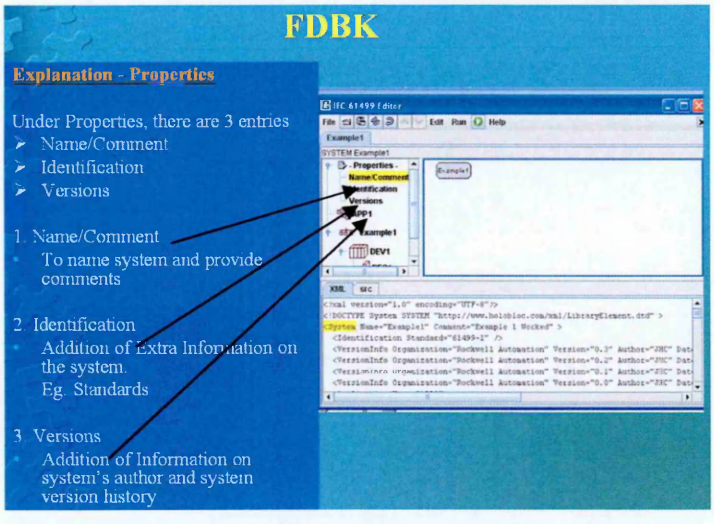


Figure A-2: Side windows explanation

Figure A-2 shows the descriptions and explanations of the various attributes within the side window. The attributes describe the system by its name, functional comments, extra information that uniquely identifies the system from others as well as information on the system version history and its author.

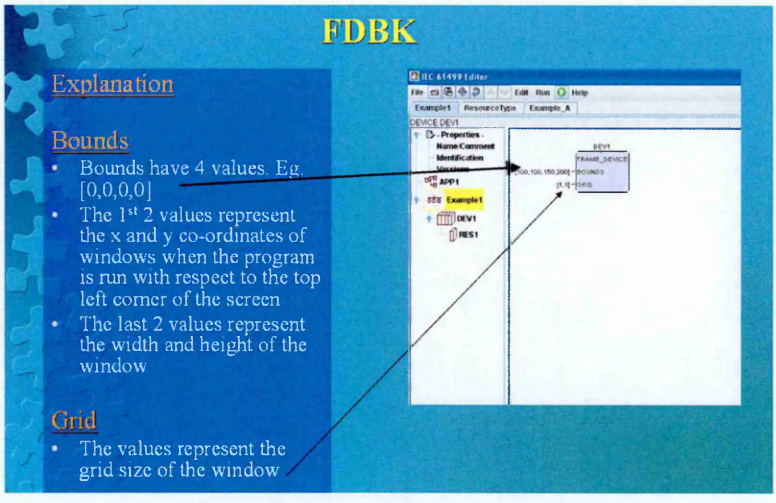


Figure A-3: Explanation of the device display during run on the main window

The attributes as shown in Figure A-3 describe the display panel of the device during the execution of the program. The bounded values state the relative values of the display screen for the first two values relative to the top left corner of the window screen with respect to the top left corner of the application window display. The grid sets the grid size of the window during the execution of the program.

5. Simple light switch example

A problem is proposed within the presentation slide of Figure A-4, with the following intentions and purposes:

i. To give the user a hands-on experience in problem solving using FBDK

ii. To guide engineering users new to programming with detailed steps in order for them to experience the programming experience within FDBK.

iii. To provide engineering users an opportunity to experience the ease of use in Function Block Programming with FDBK

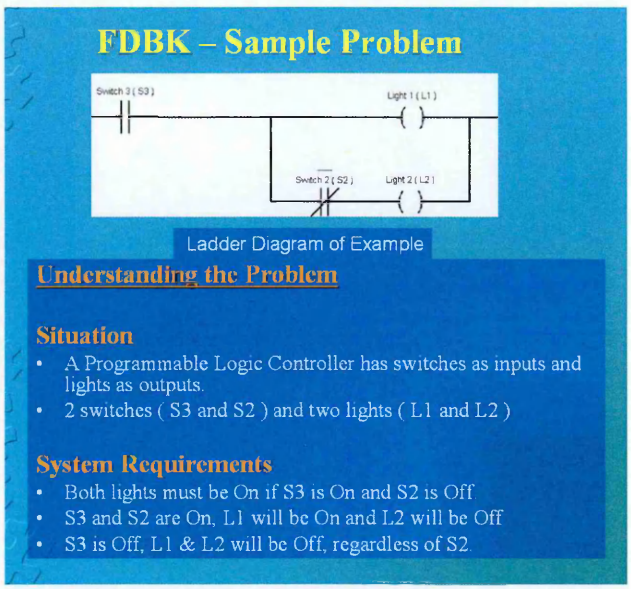


Figure A-4: Simple light switch example

Figure A-4 shows the details of the sample problem. The system requirements are then translated into a ladder diagram. The problem is initially translated into ladder diagram since it is a format which is generally familiar to mechanical and automation engineers who are likely to lack a thorough background in software programming.

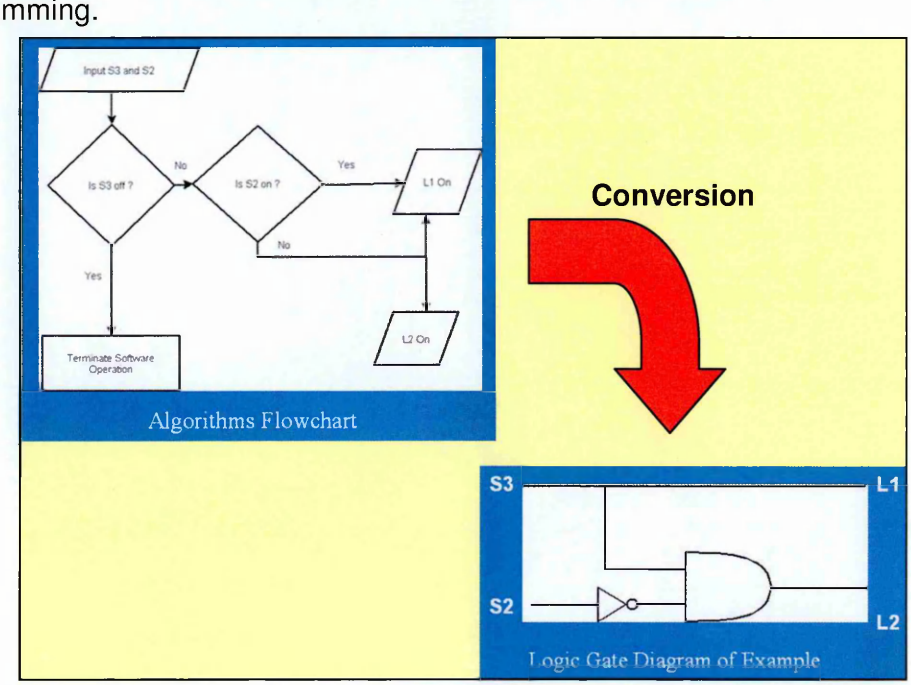


Figure A-5: Conversion from the algorithm flowchart to the logic gate diagram

Based on the ladder diagram or problem specifications, the algorithm flowchart is next generated, as shown in Figure A-5. It is then converted to a Logic Gate Diagram for programming within FBDK, utilising the logic gate modules of the software.

6. Step-by-step implementation

The first step is the creation of a new blank file. The illustrated steps can be seen within the presentation slides of the figures. Figure A-6 firstly depicts the overall system details.

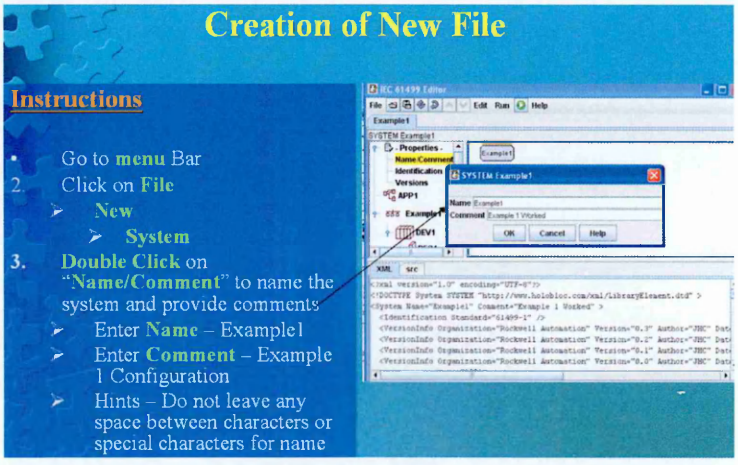


Figure A-6: Illustrated steps of new file creation

An important step is to learn how to save and load a file in FBDK, as shown in Figure A-7.

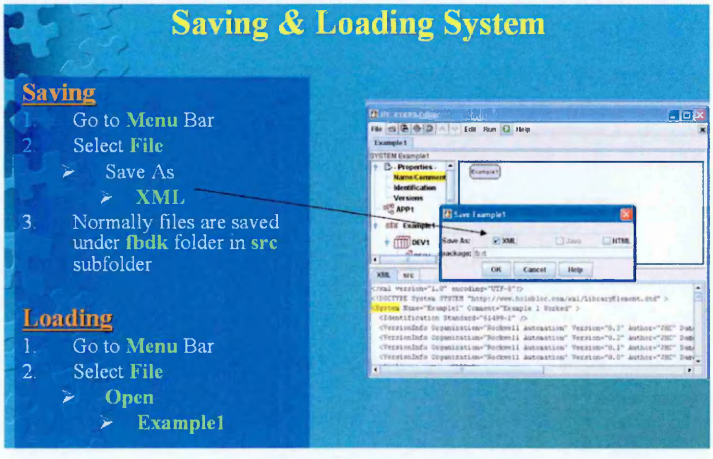


Figure A-7: Saving and loading files in FBDK

The next step involves inputting device attributes such as the device type, bounds and grid. The attributes will determine the device display during the running of the program. As explained earlier, the bound and grid size can be specified here, as shown in Figure A-8.

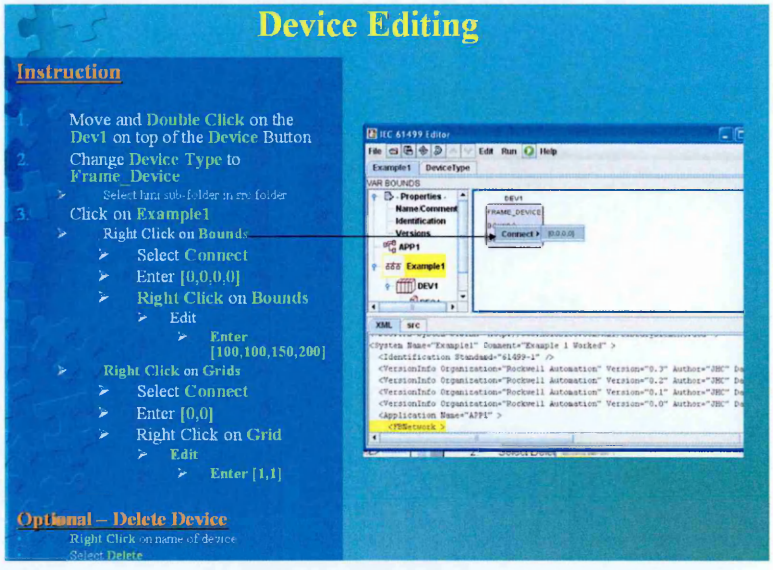


Figure A-8: Device editing - illustrated instructions

Devices are made up of resources that contain the function blocks required for the program to operate. For the simple proposed problem, only 1 resource is necessary, while the other resources, which are created by default, are deleted. The instructions for managing panel resources and deleting the unwanted resources are shown in Figure A-9.

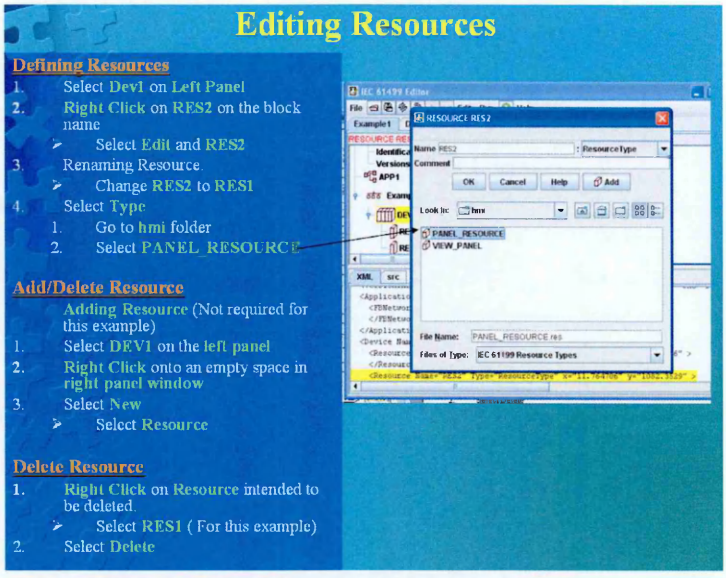


Figure A-9: Resource editing in FBDK

Function Blocks are added to the system based on the earlier logic gate diagram. The various logic gates or Boolean function block are preloaded within the software itself. New Function Blocks, including composite function blocks, can be added into the library for further use. In this example, 2 inputs function blocks, 2 output function blocks, 1 NOT gate function block and 1 AND gate function block are added into the system, as shown in Figure A-10.

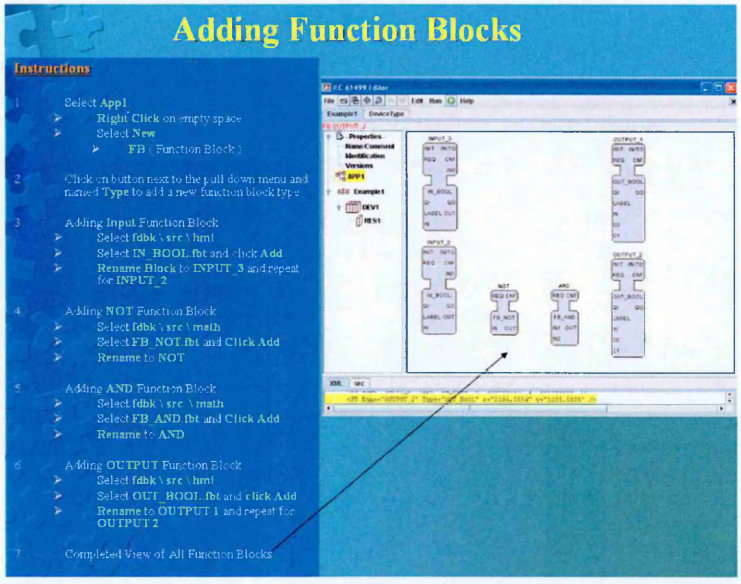


Figure A-10: Addition of function blocks into the system

The next step is to link the event and data flows for the various function blocks. In this illustration, shown in Figure A-11, the event flows are green lines which

can be connected to function blocks by holding the Alt key, then by clicking the mouse at the source and destination points. Input and output function blocks have additional events called INIT and INITO which are the initialisation events into and out of the function blocks respectively.

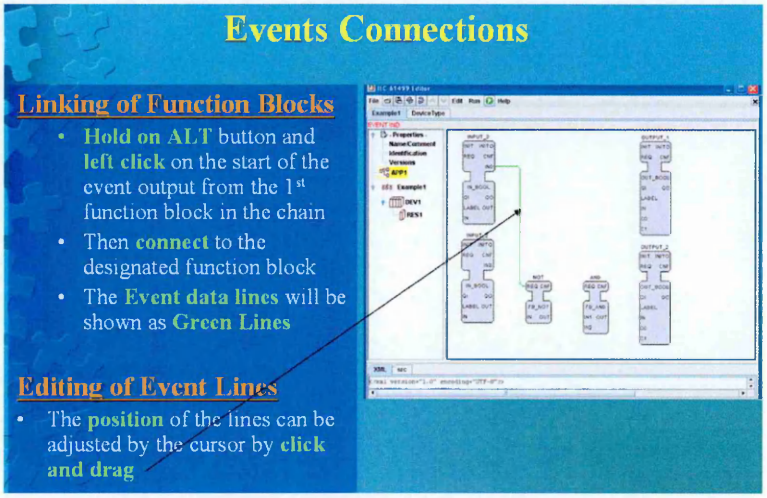


Figure A-11: Events line connections in FBDK

Data flows are only read when an event flow enters the function block. The data is processed by the function block algorithm and the processed data then exits the block. The data connection lines are connected using the same method as for the event flow connections. Figure A-12 shows the illustrated steps in connecting data lines in FBDK.

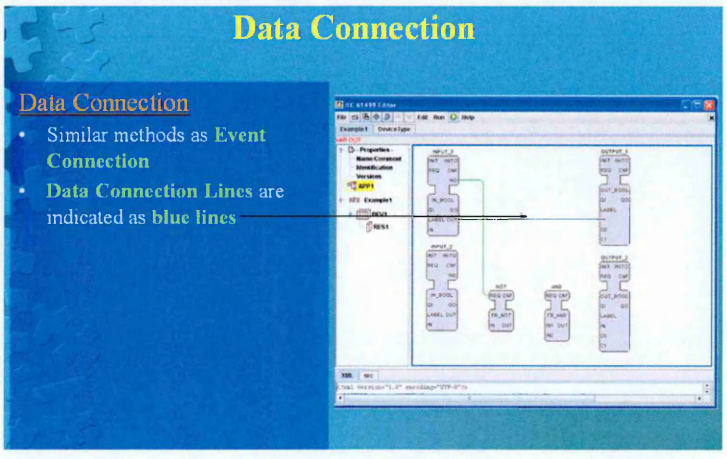


Figure A-12: Data flow connections in FBDK

The various inputs are specified within this step, they include the initial data input flow into the various function blocks as well as each data input flow label

and its description. The labels of the indicator lights represent its description labels. These steps and their illustration are shown in Figure A-13.

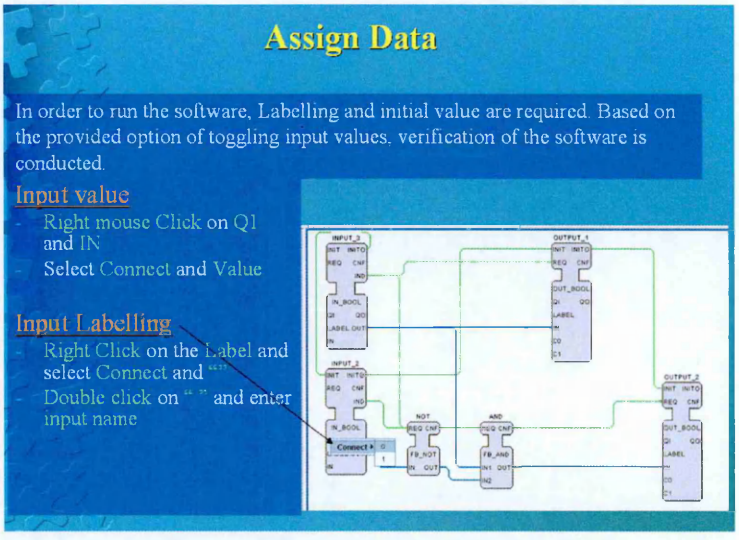


Figure A-13. Assigning data inputs

In a practical scenario, following successful simulating and debugging, the function block has to be downloaded to the embedded platform. To analyse the program within the software, the function blocks can be mapped to outputs 1 and 2 as shown in Figure A-14. A Start function block is then connected to the INIT port as the first input function block for the system. As this is the final step before running the program, error checks must be first carried out.

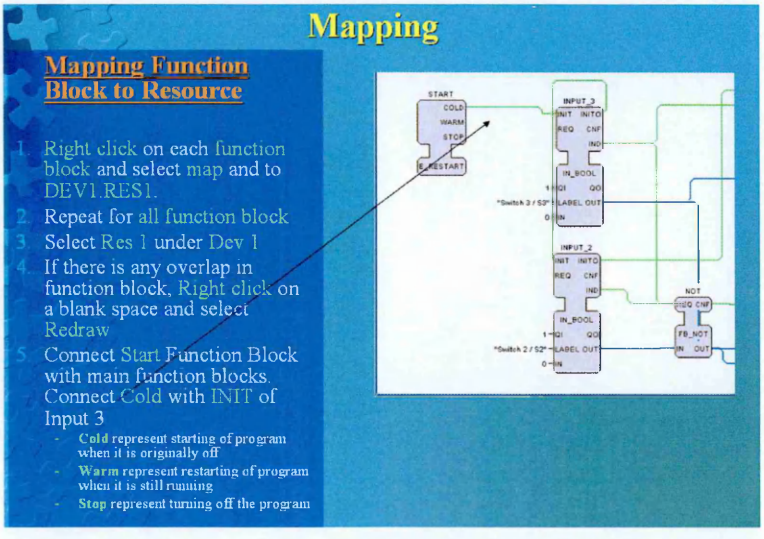


Figure A-14: Mapping from the function block to the resource

7. Program Run

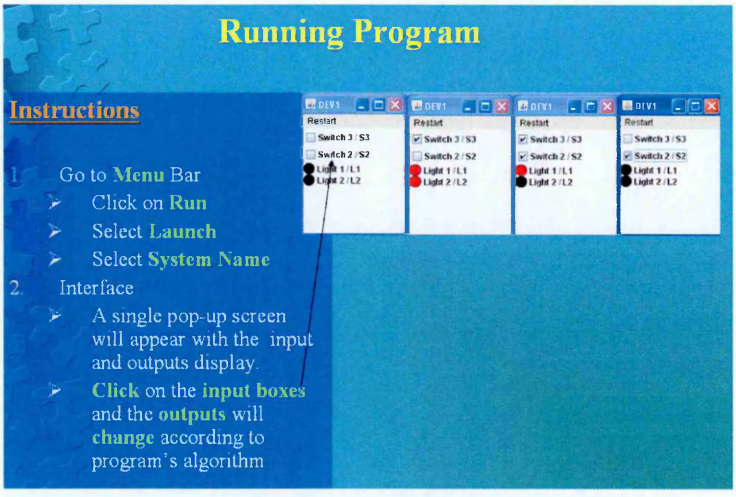


Figure A-15: Program run results

As shown in the Figure A-15, there are 4 different permutations of the 2 inputs. For illustration purposes, the figure shows the 4 different inputs in 4 separate windows. Within each window on the right hand side of the figure, the separate output results are shown. By clicking on the box beside the inputs, a 0 or 1 input can be toggled to show the different influences. After checking and confirming the correct outputs from the various input selections, the program is completed.

APPENDIX B: PROOF-OF-CONCEPT TOOL INSTALLATION INSTRUCTIONS

1. Introduction

This document is a step-by-step system developer's installation guide for the proof-of-concept tool, again within the context of the simple light switch example. The light switch example show the process of code generation using function blocks, converting to SystemC and Verilog. The document assumes readers have hands-on experience with IEC 61499 function block programming and C compilers.

2. Using the Function Block Development Kit (FBDK)

Appendix A showed how a Function block application can be created using FBDK. To run a function block system, click on Run > Launch > the system name, as shown in Figure B-1.

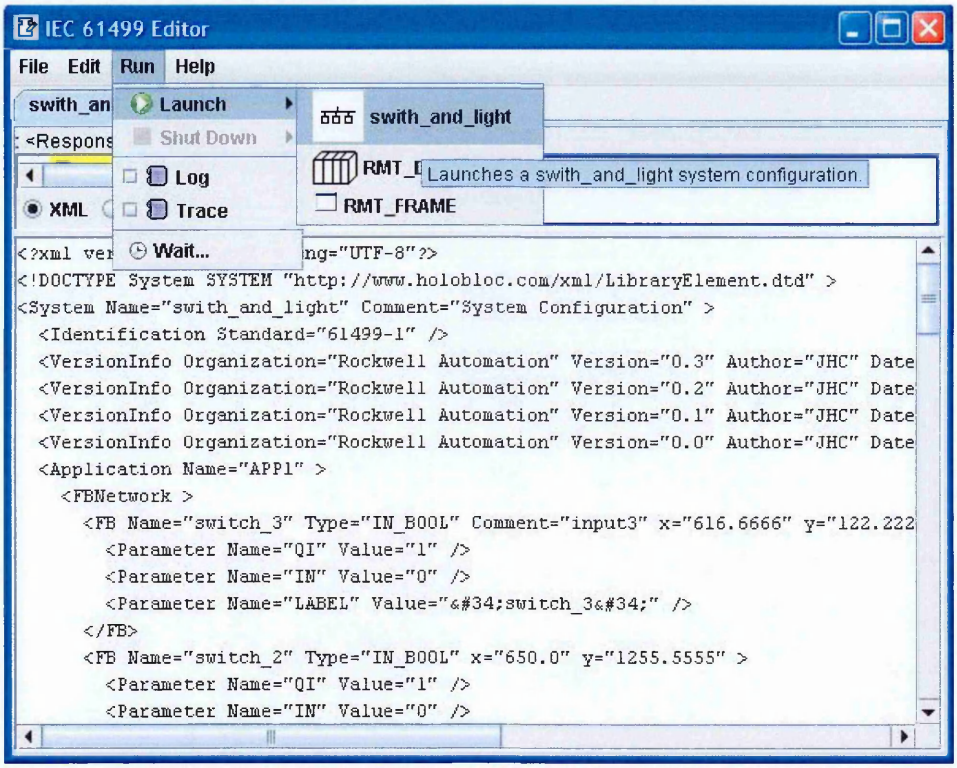


Figure B-1: Launching the switch and light system

The simple light switch system created from Appendix A consists of the two Boolean switch inputs which appear as checkboxes switch 1 and switch 2. The two lights are simulated by Boolean outputs light1 and light 2 as shown in Figure A-15. In the next section, the simple light switch block program will be converted to SystemC using XML and XSL for FPGA-based embedded platform.

3. XML to SystemC using MSXSL

The IEC 61499 function block tool saves all function blocks in XML format with the file extension '.fbt'. The following steps are required to generate the SystemC code from XML using the Microsoft XSL (MSXSL) parser.

a) Place all the XML files and XSL files together with the MSXSL.EXE parser (or any XSL parser) in one folder.

b) Create the batch file and save it as .bat file format, for example parse.bat. The contents of the batch file are as shown in Figure B-2. All the function blocks are saved with the same files extension '.fbt'. The Function block files (.fbt) comprise XML documents such as FB_AND.fbt, FB_NOT.fbt, IN_BOOL.fbt, OUT_BOOL.fbt. SL_CIRCUIT.SYS. XSL files (.xsl) are the corresponding style sheet for the XML. MSXSL will convert XML files according to XSL's style sheet definition into header files in the '.h' format and C++ source code in the '.cpp' format. For SystemC, header and system files are required for each function block. FB_AND.h, FB_NOT.h, IN_BOOL.h, OUT_BOOL.h and SL_CIRCUIT.h represent the header files, with FB_AND.cpp, FB_NOT.cpp, IN_BOOL.cpp, OUT_BOOL.cpp and SL_CIRCUIT.cpp being the system files in SystemC language format.

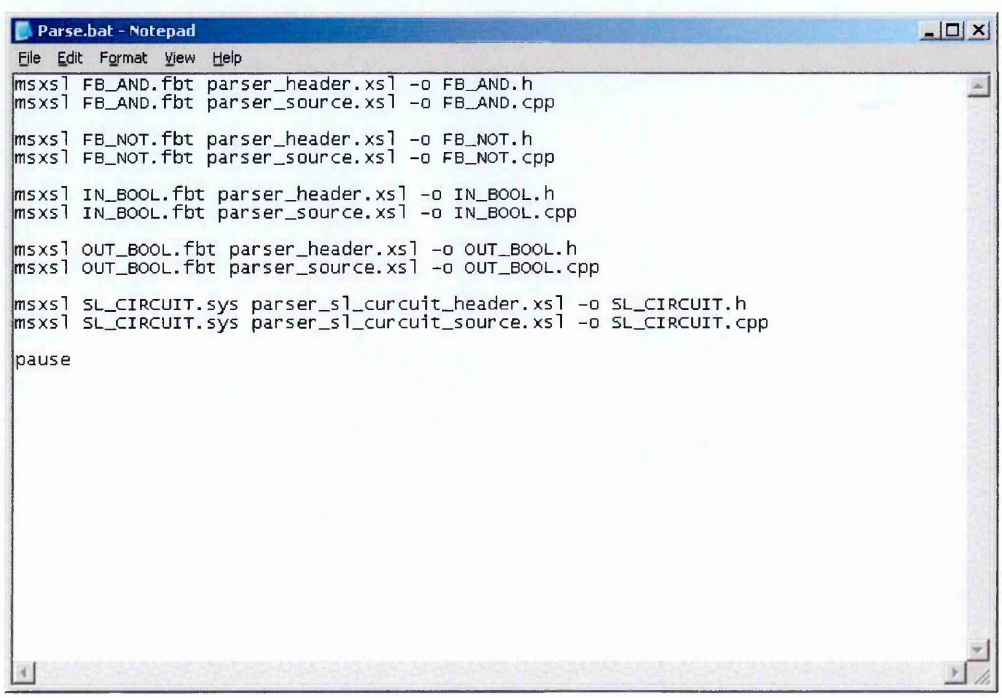


Figure B-2: Batch file for parser

c) Run the batch file 'parse.bat' to execute the script. Figure B-3 shows the batch file execution in the command prompt. The last command line 'pause' is to enable the computer to pause the currently running batch

file until the user presses any key. The corresponding *.h and *.cpp files will be generated in the same directory as the other function block files.

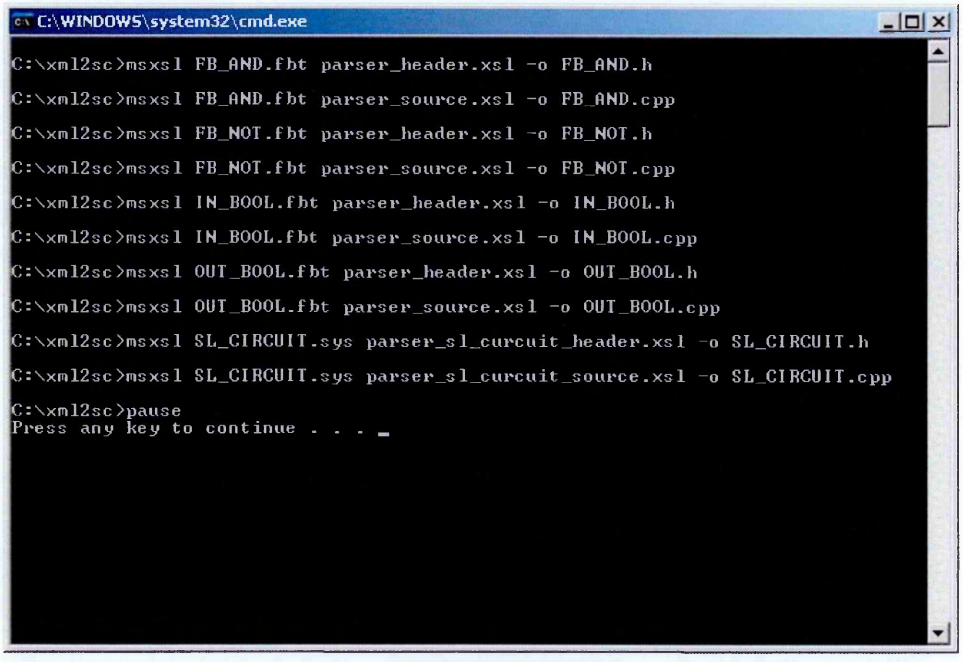


Figure B-3: Execution of batch file for XML to SystemC conversion

4. **Changing the text encoding scheme**

When SystemC files are generated, they need to change the text encoding format into ANSI because the generated SystemC files inherently contain some unnecessary non-ASCII characters, which may cause errors when compiled with Microsoft Visual C++ compilers (VC++). In order to remove the unnecessary ASCII characters, open each of SystemC file in NOTEPAD (or any text format editor), select Format -> Encode in ANSI from the menu and then save all the files.

5. Executable file for the switch and light system

The header and system files can be compiled into an *.exe file using VC++. The following are the sequential steps required to execute the program.

Setup a new VC++ project

a) Open VC++ 6.0 (or higher), Click File->New or alternatively press Ctrl+N to create a new VC++ project.

b) When 'New' dialog appears, go to Projects tab and select Win32 Console Application project.

c) Key-in the Project name and new Project location in the Location text

box.

d) Click OK to continue to next step.

e) The next step is to choose the empty project option and click Finish.

f) A workspace window will appear and then click file view.

g) Right click on the <Project Name> files SL_CIRCUIT files and select Add Files to Project option as shown in Figure B-4 Add the generated SystemC files and click OK. Now all SystemC header and system files are added to the VC++ project.

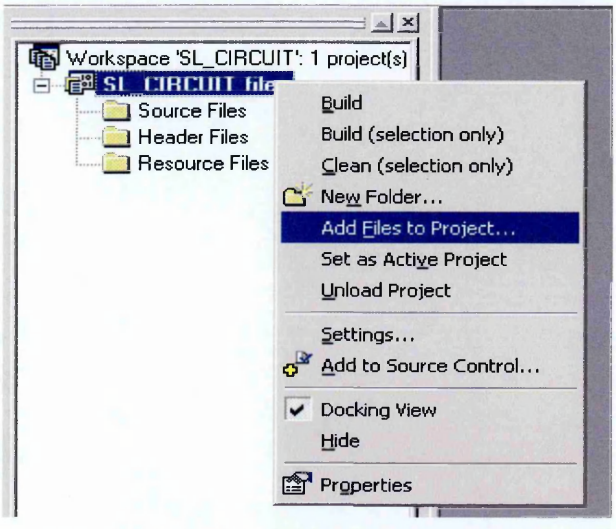


Figure B-4: Adding SystemC files to a VC++ project

Compile and execute the program

In order to compile the SystemC source code into an executable file, the VC++ compilers need to link to the SystemC 2.1 header file. Go to the Project -> Settings menu in VC++, or alternatively press Alt+F7. Select the C/C++ tab and then the Pre-processor option in Category to specify the SystemC 2.1 header file location in the Additional include directories option, as shown in Figure B-5. The SystemC header file (systemc.h) can be found in the systemc2.1 folder\src sub-directories. Example: C:\systemc-2.1.v1\src. Select the C++ Language option in Category and select the Enable Run-Time Type Information (RTTI) check box to avoid run time errors. RTTI provides the run time type information of the dynamic variables for systems which are using dynamic variables with multiple inheritances, dynamic type casting as well as template classes. Most of the SystemC library files are heavily dependent on the Run Time Type Information.

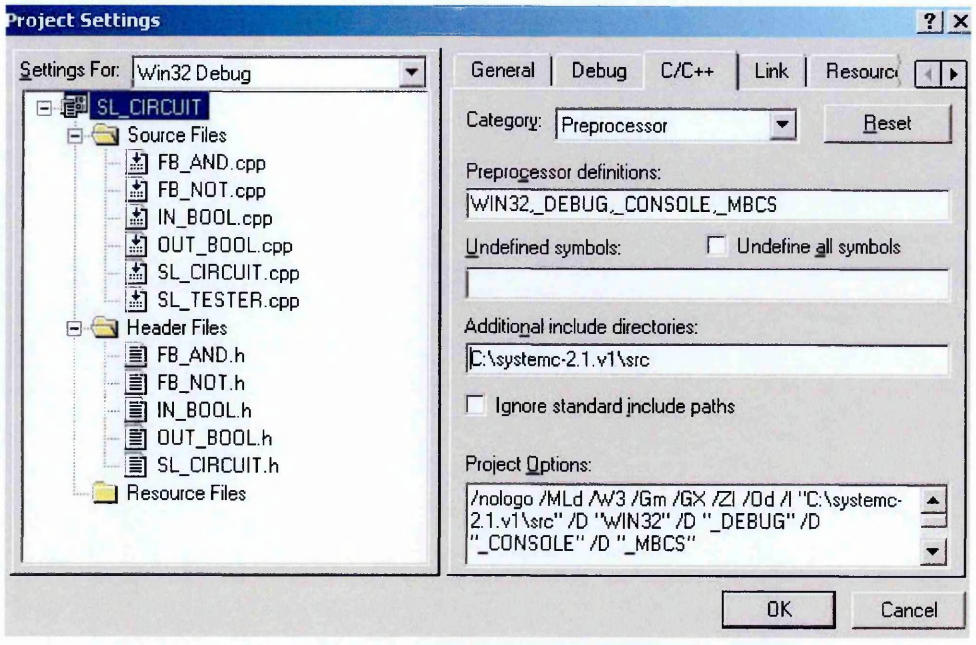


Figure B-5: Adding the SystemC header file directory

The next step is to specify the SystemC library file name and file location at the Link tab. Go to the Link tab and select the input option in Category. Key-in SystemC.lib in the Object/Library modules option tab. Then specify the systemc.lib file location in the Additional library path, as shown in Figure B-6. Normally systemc.lib files can be found under the SystemC 2.1 folder\msvc60\SystemC\Release directory. If the Release folder does not exist, compile the SystemC source code in release mode. To do this, open the SystemC source project file (SystemC.dsw) located in SystemC 2.1 Folder\msvc60\SystemC\ and then go to the Build-> Set Active Project Configuration menu and select Win32 Release. Press F7 to compile the SystemC source code.

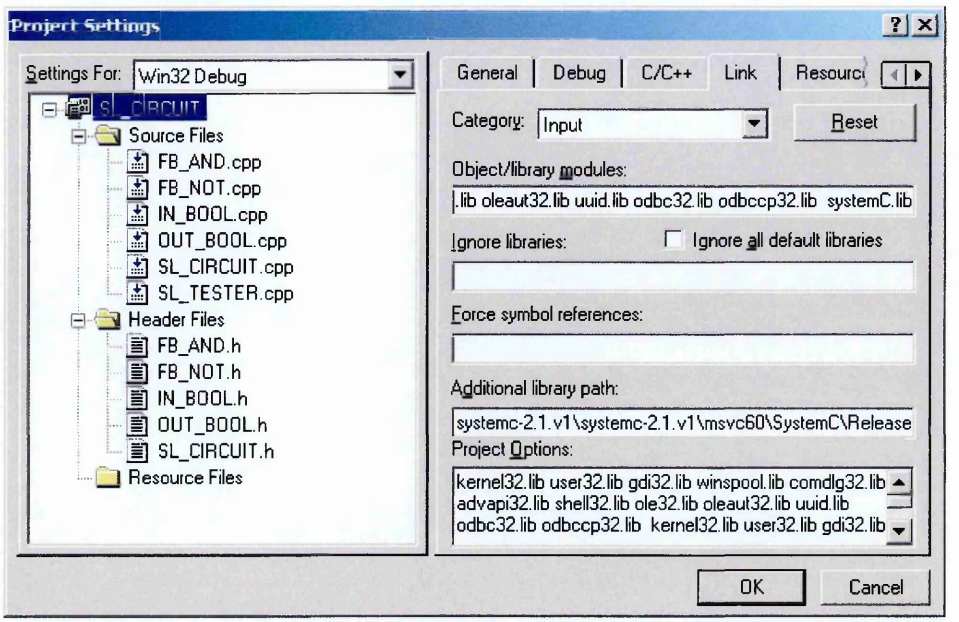


Figure B-6: Adding SystemC library modules and directories

The final step is to compile and execute the project as an '.exe' file. Select Build from the Visual C++ menu and choose Build <Project Name>, for example Build SL_CIRCUIT.EXE, or alternatively press F7 to compile the project. To execute the program, select Build from the menu and choose Execute <Project Name>, for example Execute SL_CIRCUIT.EXE, alternatively presses the Ctrl+F5 key to execute the project.

6. View the Simulated Results though the Wave Form Viewer

A Waveform viewer can be used to view the actual results of the SystemC file. A Value Change Dump (VCD) file can be easily created by the adding tf=sc_create_vcd_trace_file("sl_trace") command to the SystemC file. VCD files automate the annotation of signals and reduce the need to know the internal logic of the system. A VCD file is a STANDARD format and is compatible with various waveform viewers. A VCD file is an ASCII file which consists of header information, variable definitions and the value changes for all variables specified in the task calls.

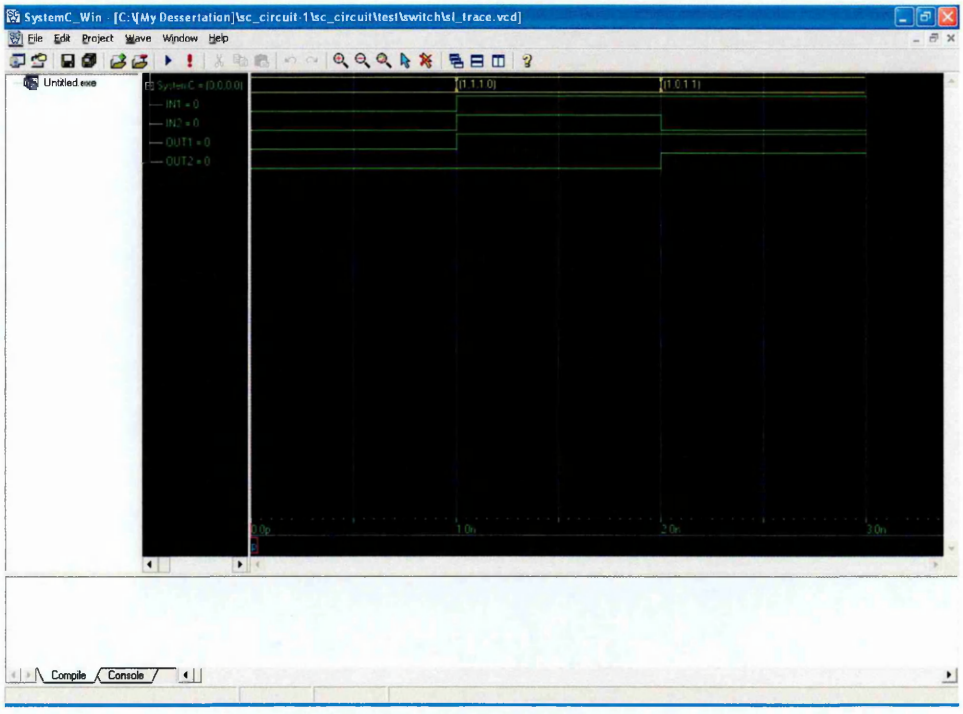


Figure B-7: Digital waveform of the simulated result of switch and light system

There are many waveform viewers which are compatible with the VCD format. In this guide, SystemC_Win software is used. Open the SystemC_Win software and go to File menu and then select open vcd files. Choose the vcd file from the directory. Press the zoom full menu from the system menu. The results can be seen in digital waves as shown in Figure B-7.

There are two horizontal main screens in SystemC_Win. The first main screen is to show the VCD file data and the second main screen is to show the status message as well as the diagnostic messages of the application. There are three vertical sub-screens in first main screen. The first sub screen is to show the meta-data of the SystemC application such as executable file name. The second sub screen is to show the in port and out ports. In the Switch and Light example, there are two input ports IN1 and IN2 and two output ports OUT1 and OUT2. The values of the input and output ports initially zero, as shown in Figure B-8. A zero means that the port is not active or has no data, while a one means that the port is active. The third sub screen is to show the value changes of the in and out ports in graphical format. The green colour line represents each in and out port and the yellow colour line represents the value of each port per simulation cycle. It is possible to see step-by-step value changes by clicking on the particular line and checking the value of port in second sub screen. After modelling and simulating the system in SystemC language, a Verilog code generator is required to produce the necessary code for an FPGA

implementation. The next section discusses how Verilog code can be auto-generated.

7. Using the SC2V_PRE_PARSER

Copy the SystemC files into the SC2V_pre_parser folder. In this folder, open and edit the "run.bat" file to input the SystemC file names, which are to be modified, for example sc2v_pre_parser.vbs FB_OR. This command line helps to execute the sc2v_pre_parser.vbs with the parameters from the FB_OR SystemC file. After editing the 'run.bat' file, run the batch file and the parser will generate relevant SystemC files which are ready to be used in the SC2V software tool.

Setting up CYGWIN

Although SC2V software is designed to run on a Linux environment, it can also be used in Microsoft Window environment by using Cygwin. Cygwin is a collection of free software tools developed by Cygnus Solutions to provide virtual LINUX or UNIX systems in various versions of the Microsoft Windows environment. In order to install the Cygwin on the Window Environment, an Internet connection must be available with a minimum of 100 MB of free disk space. Install Cygwin directly from the Internet, define the root folder location and leave the rest of the options at their default values. The next step is to define the location of the installation files and make sure that the location is different from the location of the root folder. The installation program will automatically download and show a download site list to choose the preferred download site. Some recommended download sites are http://mirror.averse.net, ftp://ftp.u-aizu.ac.jp and ftp://ftp.nctu.edu.tw. Then choose the software packages that need to be installed for Cygwin setup, as shown in Figure B-8.

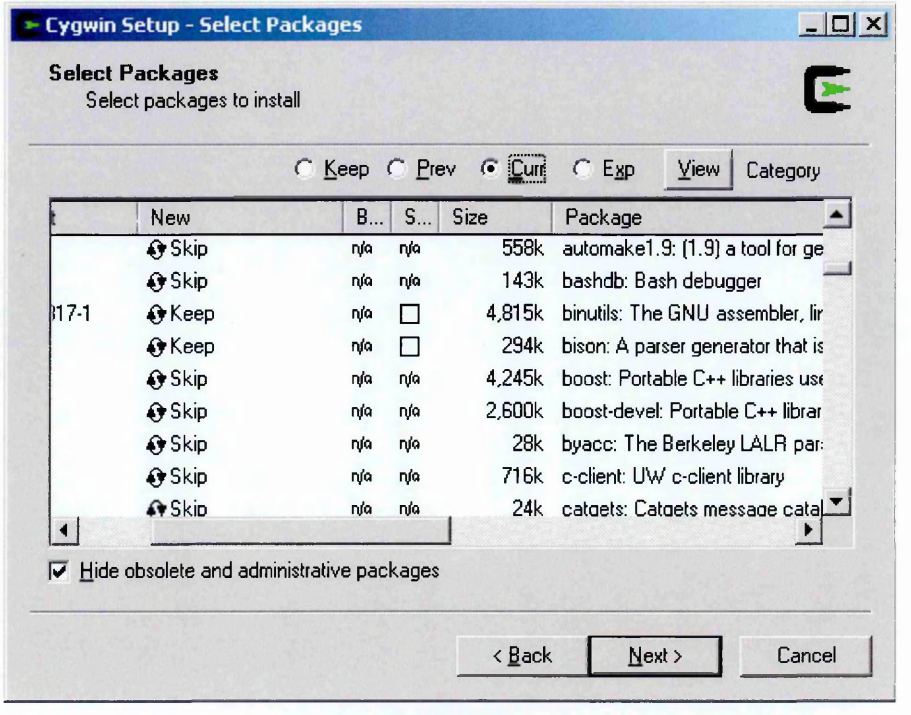


Figure B-8: Cygin Setup

For choosing software packages, some recommended software packages are "bison", "flex", "gcc C compiler", "gcc C++ compiler" and the "make" utility. Bison is a variation of YACC (Yet Another Compiler Compiler) and flex is a variation of LEX (LEXical analysers). Both YACC and LEX are important components of the compiler. The C++ compiler is chosen because SC2V source code must be compiled before it can be used. "GCC" is the open source version of C++ compiler, which is widely used in both Linux and Microsoft Window environments, the "Make" utility is also known as "GNU-Make". It is a utility to build C++ files into an executable format. SC2V source codes contain a "make" file which is a set of instructions for the "make utility" to compile the SC2V source code. After choosing the software packages, the Cygwin setup program will automatically download the selected packages and install them on computer. When the above steps are completed, Cygwin should be successfully to provide a virtual Linux environment on Microsoft Window platform allowing for SC2V to be used in Windows.

8. Using SC2V for SystemC to VERILOG conversion

Once Cygwin is successfully installed, SC2V needs to be compiled. Copy **SC2V folder** to the **<Cygwin Folder>\home\administrator** folder, then run the **Cygwin Bash Shell** program to create the command prompt. Ensure that the SC2V folder is under the root folder of the administrator through the use of the **dir** command. The next step is to change the current directory to SC2V by

using the change directory command **cd**. When the current directory is the **administrator** ~/**SC2V** folder, type **make** in the command prompt to compile the SC2V source code. The step-by-step compilation will be shown in the command prompt, as depicted in Figure B-9. After compiling the SC2V program, open the **<Cygwin sFolder>\home\Administrator\sc2v\bin** folder and copy the **sc2v.sh**, **sc2v_step1.exe**, **sc2v_step2.exe** and **sc2v_step3**.exe files into the **<Cygwin Folder>\bin** folder. The **<Cygwin Folder>\bin** is a global folder and any application from that folder is accessible from any part of the system without changing the directory.

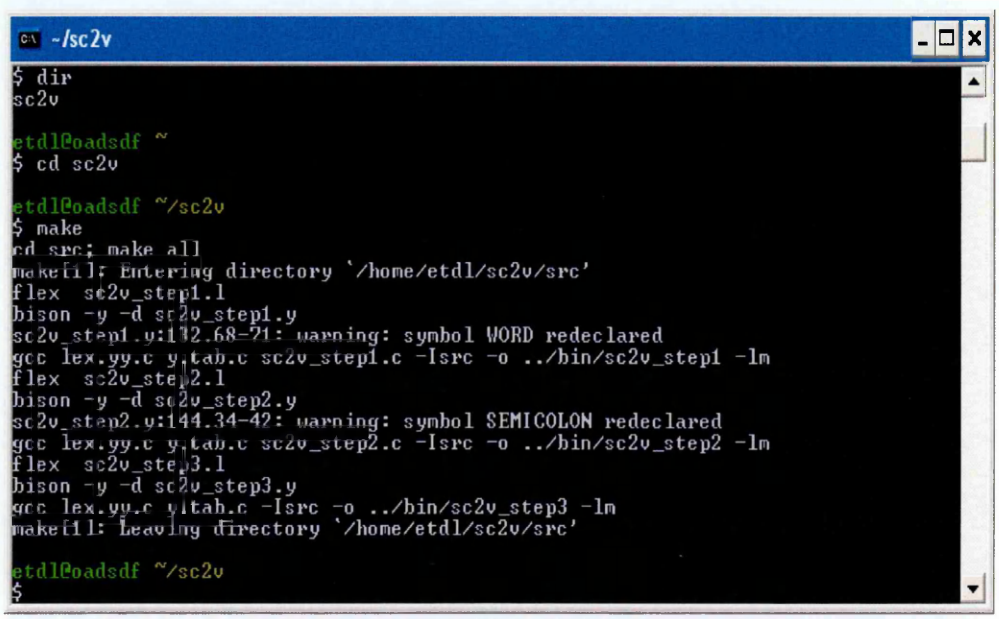


Figure B-9: Successful compilation of the SC2V source code

In this research, the **sc2v.exe** files and **sc2v.sh** files are placed together in the **<Cygwin Folder>\bin** folder, it is thus necessary to correct the command in the **sc2v.sh** file. In order to correct **sc2v.sh**, open the file from **<Cygwin Folder>\bin,** delete the code shown in Figure B-10 and replaced it by the new code shown in Figure B-11.

../bin/sc2v_step1 < $1.cpp
../bin/sc2v_step2 < $1.h > $1.sc2v
../bin/sc2v_step3 < $1.sc2v > $1.v
\rm *.sc2v

Figure B-10: Incorrect code in SC2V.SH

sc2v_step1 < $1.cpp
sc2v_step2 < $1.h > $1.sc2v
sc2v_step3 < $1.sc2v > $1.v

Figure B-11: Correct codes in SC2V.SH

Now SC2V is ready to be used. Run the **Cygwin Bash Shall** program and go to the SystemC file directory using the **cd** (change directory) command. It is easier to access if the SystemC source files are placed under the **cgwin/home/administrator/** directory. Note that all file names should be absent of spaces. Type <**sc2v.sh filename**>, for example **sc2v.sh FB_AND**, to convert the SystemC source code to Verilog codes. There is no need to consider the file extension because SC2V requires that .h and .cpp files must exist for each program module. Figure B-12 depicts the step-by-step process of converting SystemC to Verilog by using a SC2V parser.

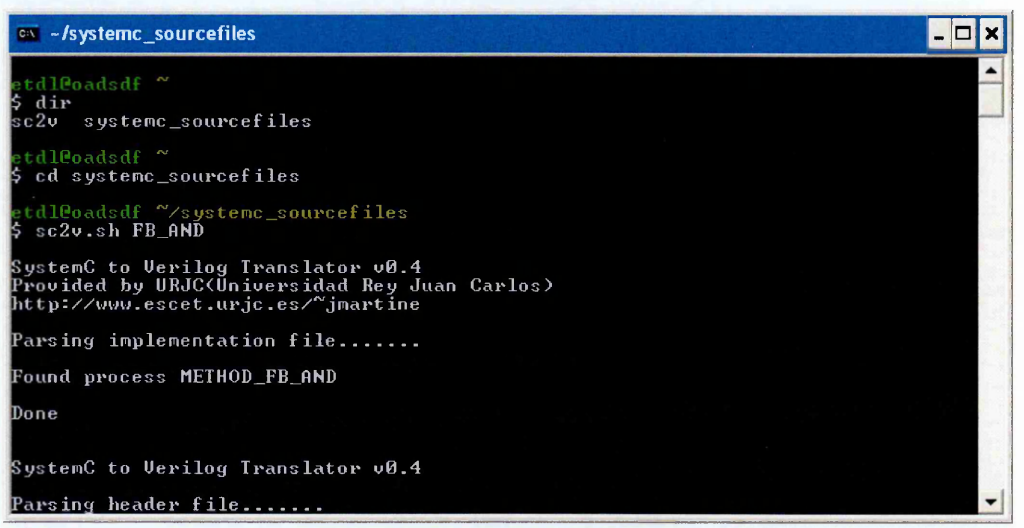


Figure B-12: Translating from SystemC to Verilog

All SystemC files need to be parsed one by one and converted to Verilog. The corresponding Verilog files will be generated in the same folder as the SystemC source code whenever each SystemC file translation is finished. The Verilog complier and simulation tools are then used to test the generated Verilog code.

APPENDIX C: EBNF CODE GENERATOR SETUP

This document is a step-by-step system developer's guide for the proof-of-concept tool - Extended Backus–Naur Form (EBNF) code generator. The EBNF code generator is an improved interactive graphical user interface (GUI) code generator which simplifies the creation of the style sheets by dragging the XML definition to match the target language EBNF.

The document will use the XML created by the light switch example from Appendix A to generate SystemC source code. The document assumes that readers understand EBNF and IEC 61499 function blocks.

1. Before running this software, download and install JDK 6 or higher.
2. Create an XML file (Here, this file is "SC_CIRCUIT.xml"). It is proposed that this file name be the same as your FBDK auto-generated file name.

   At the beginning, attach the following code to the file

```
<?xml version="1.0" encoding="UTF-8"?>
<FB Name="IN_BOOL" Language="SystemC">
  <DataType>
    <ReplacementRule Source="BOOL" Destination="bool"
    />
    <ReplacementRule  Source="INT"  Destination="int"
    />
  </DataType>
  <Body>
  </Body>
</FB>
```

   The tag "ReplacementRule" should be created by the user. This format can help store the corresponding data types of the source language (FBDK) and the target language.

3. As the Function Block Development Kit (FBDK) auto-generated XML file cannot tell the location of input output ports, the user should manually assign them. In the example below, tag "<Parameter Name="IN" Value="0" />" should be the input port. The user then needs to change this tag's name into "Input". (for output ports, change it into "Output").

```
<FB Name="switch_2" Type="IN_BOOL" x="650.0"
y="1255.5555">
        <Parameter Name="QI" Value="1" />
        <Parameter Name="IN" Value="0" />
        <Parameter Name="LABEL"
        Value="&amp;quot;swich_2&amp;quot;" />
        <Connection Id="sig3" Name="OUT1" Type="bool"
        Dir="Source" />
</FB>
```

Then, it becomes:

```
<FB Name="switch_2" Type="IN_BOOL" x="650.0"
y="1255.5555">
        <Parameter Name="QI" Value="1" />
         <Input Name="IN" Value="0" Id="in_a" Type =
         "bool" />
        <Parameter Name="LABEL"
        Value="&amp;quot;swich_2&amp;quot;" />
        <Connection Id="sig3" Name="OUT1" Type="bool"
        Dir="Source" />
</FB>
```

Attribute "Id" is used to mark this port. "Type" is this port's data type.

4. All the programs are in the folder "Temp". The EBNF code generator is compiled into an executable file called "CodeGenerator.jar". Double click "CodeGenerator.jar" to run this software.

5. As shown in Figure C-1, click on the button "Open XML File" to choose the source file which is auto-generated from FBDK. (For example, here this file is "SC_CIRCUIT.sys").

6. Click on the button "Open Existing Data" and choose the file just created.

7. Double click "Open EBNF File" to choose the specified target language's EBNF. (For example, the EBNF for SystemC is "SystemC.txt").

8. As shown in Figure C-2, double click "module-name" as "module-name's value will be assigned by the user". Then input the value in this text area. In this example, we change the module name to "SystemC" as shown in Figure C-2. After changing the module name press <Enter>.

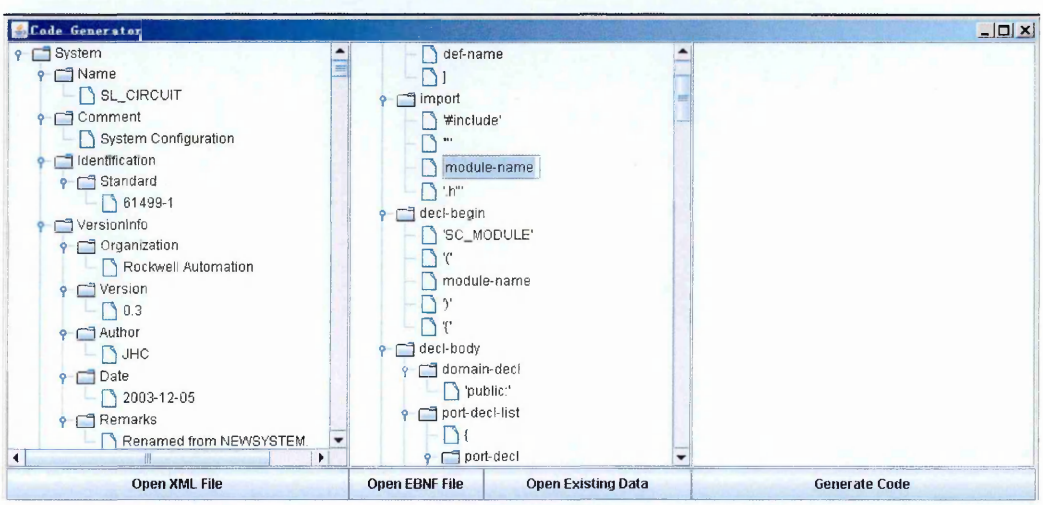


Figure C-1: Select the module name

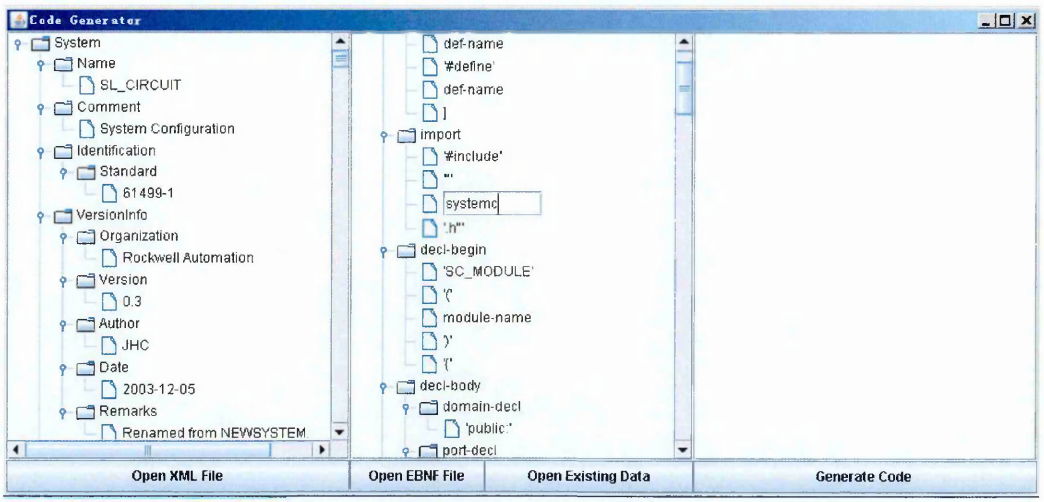


Figure C-2: Change module name to "SystemC"

9. As shown in Figure C-3, drag the "Name" definition from the XML window to "instant-name" field in the EBNF window. This is followed by dragging the "Type" definition to the "module-name" definition. This is to define the relationship of the XML file within the EBNF rules.

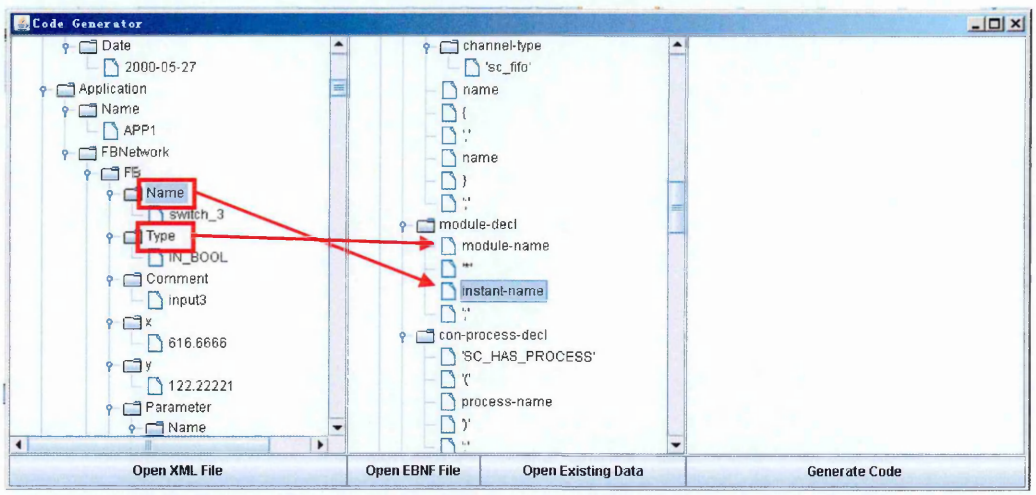


Figure C-3: Define the module-name and instant-name

10. As shown in Figure C-4, to define the port-name and data-type, simply drag from the "Id" and "Type" fields in the XML window to the "port-name" and "data-type" in the EBNF window respectively.

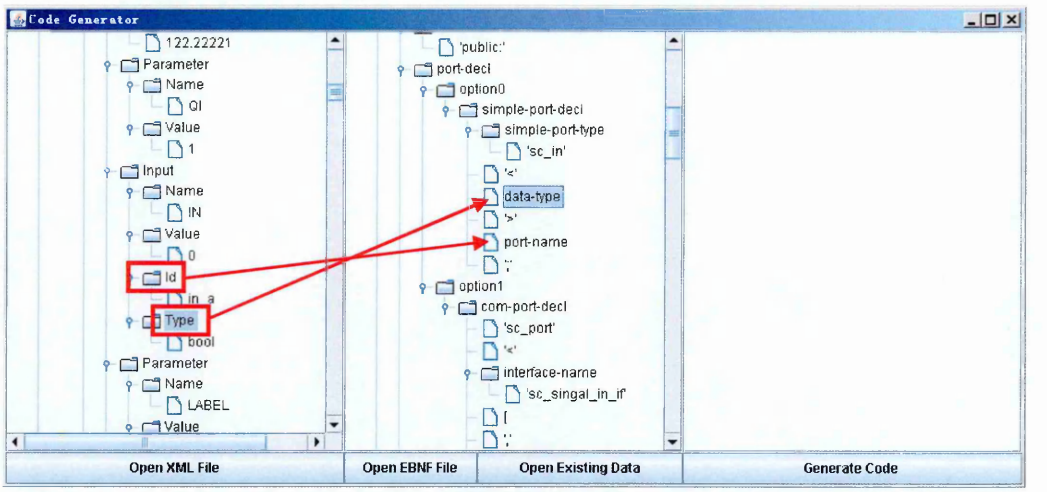


Figure C-4: Define the port-name and data-type

11. As shown in Figure C-5, to define the port-type double click the "sc_in" field. The system will gather the various possible port-types from the EBNF definition and present it as a drop down combo box. The user just needs to select the desired type. In this example, "sc_out" will be used to define the port-type as an output port.

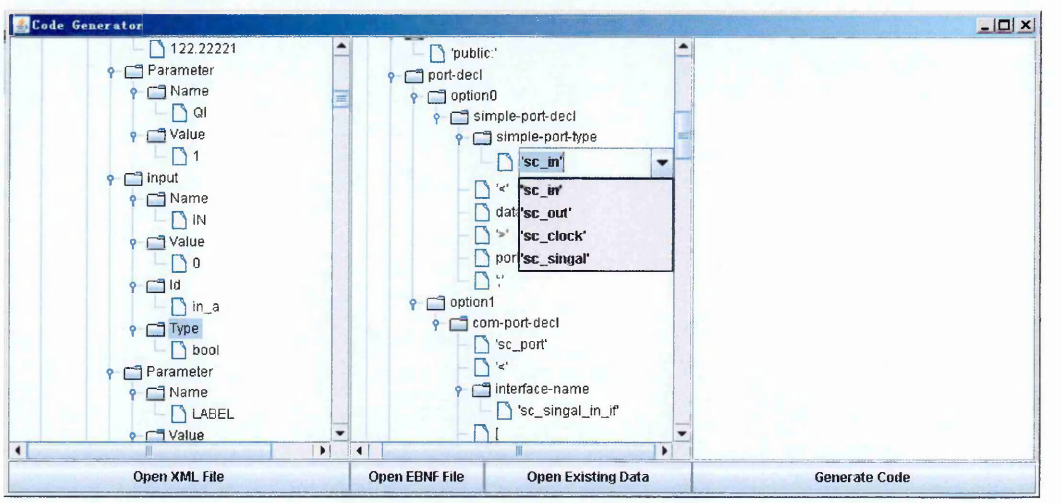


Figure C-5: Define port-type

12. Repeat the above step for user to define the functions and data types.

13. As shown in Figure C-6, when all the user defined fields are linked, the EBNF is ready to be used. The user may click the "Generate Code" button to generate the target source code. In this example, the target source code is SystemC.

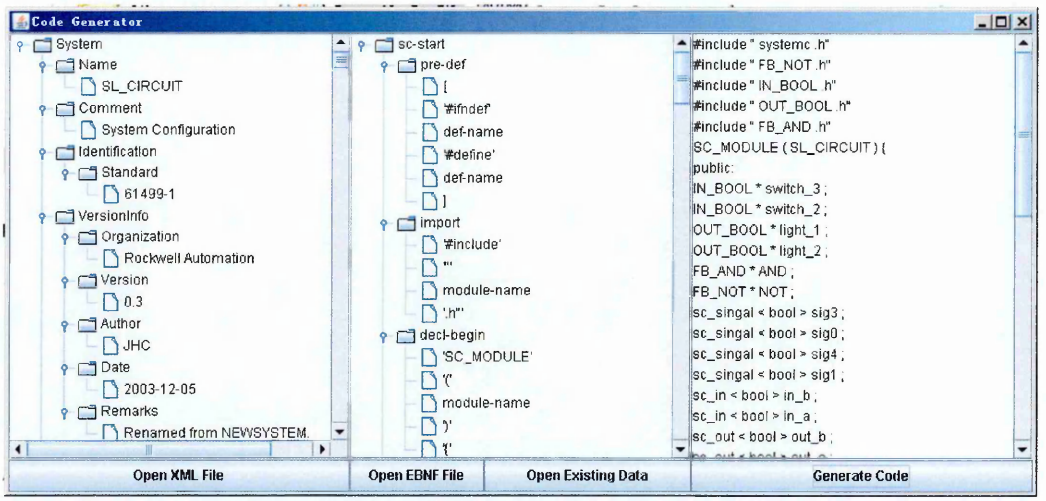


Figure C-6: Generate the SystemC code

Once the EBNF is modified with the user defined fields, the same EBNF can be used for other function block files. The user is only required to modify the EBNF when there are new user defined fields. The same EBNF will be stored in a repository and can be reused repeatedly.

APPENDIX D: FEEDBACK FORMS FROM POTENTIAL USERS

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☑ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☑ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

_____

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 4 (5)
IEC 61499 function block and the development framework is easy to understand	1 2 3 4 (5)

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 (5)
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 (5)
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 (5)
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 (5)
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 (5)
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 (5)
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 (5)
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 (5)

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☑ Yes ☐ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☑ Others (Please specify) MANUFACTURING.

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☑ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

5 years.

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 (3) 4 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 (4) 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 (3) 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 (4) 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 (3) 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 (4) 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 (3) 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 (4) 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 (4) 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 (3) 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☐ Yes ☐ No Probably.

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

2

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☑ Others (Please specify) all of the above.

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☑ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

2yrs

(4) What is your area of specialty for work / study?

- ☑ Mechanical Engineering
- ☑ Electrical Engineering
- ☑ Computer Engineering
- ☐ Computer Science
- ☑ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) Ease of use

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 (4) 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 (4) 5

(6) Usefulness

	Rank levels	
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 5	NA
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 (5)	
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 (5)	
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 5	NA
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 (5)	
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 (5)	
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 (5)	
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 (5)	

(7) Are you interested to receive news, events and training relating to IEC 61499 Technologies? ☑ Yes ☐ No

(8) Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies? ☐ Yes ☑ No

(9) Do you find this event beneficial to your company? ☑ Yes ☐ No

(10) Other Comments: Our product from Mathworks : Matlab & Simulink does provide similar features that may be in tune to IEC61499 Stateflow and Simevents. Such standard requires knowledge of flow charting concept thus prefer IEC61499 at system level where codes can be also generated automatically.

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

3

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☑ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☑ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☑ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

_____

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☑ Computer Engineering
- ☐ Computer Science
- ☑ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 (3) 4 5
IEC 61499 function block and the development framework is easy to understand	1 (2) 3 4 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 (3) 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 (4) 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 (3) 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 (3) 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 (4) 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 (3) 4 5
IEC 61499 function block and development framework will help to simplify application development	1 2 (3) 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 (3) 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☐ Yes ☑ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☑ Biomedical
- ☑ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☑ Others (Please specify) R University

(3) How long did you practice software development on a professional level?

1

(4) What is your area of specialty for work / study?

- ☑ Mechanical Engineering
- ☑ Electrical Engineering
- ☑ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 4 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 4 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☑ Yes ☐ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** Interesting area. Intends to work on it.

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☑ Others (Please specify) Electro platy.

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☑ Others (Please specify) Electro platy.

(3) How long did you practice software development on a professional level?

_____

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 (4) 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 (4) 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 (3) 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 (3) 4 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 (3) 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 (3) 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 (3) 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 (3) 4 5
IEC 61499 function block and development framework will help to simplify application development	1 2 (3) 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 (3) 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☐ Yes ☑ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☐ Yes ☑ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☑ Others (Please specify) Education

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☑ Others (Please specify) Education

(3) How long did you practice software development on a professional level?

25 Yrs.

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☐ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☑ Others (Please specify) Automation

SIMTech A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) Ease of use

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 (3) 4 5
IEC 61499 function block and the development framework is easy to understand	1 2 (3) 4 5

(6) Usefulness

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 (4) 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 (4) 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 (4) 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 (3) 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 (4) 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 (4) 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 (4) 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 (4) 5

(7) Are you interested to receive news, events and training relating to IEC 61499 Technologies? ☑ Yes ☐ No

(8) Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies? ☐ Yes ☑ No

(9) Do you find this event beneficial to your company? ☑ Yes ☐ No

(10) Other Comments: _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☑ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☑ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

3 yr

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) Ease of use

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 (3) 4 5
IEC 61499 function block and the development framework is easy to understand	1 (2) 3 4 5

(6) Usefulness

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 (2) 3 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 (2) 3 4 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 (3) 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 (2) 3 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 (2) 3 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 (2) 3 4 5
IEC 61499 function block and development framework will help to simplify application development	1 (2) 3 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 (2) 3 4 5

(7) Are you interested to receive news, events and training relating to IEC 61499 Technologies? ☐ Yes ☐ No

(8) Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies? ☐ Yes ☐ No

(9) Do you find this event beneficial to your company? ☐ Yes ☐ No

(10) Other Comments: _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) STUDENT

(2) Which category best describes your company's business (you may tick more than one)?

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☐ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

_____

(4) What is your area of specialty for work / study?

- ☐ Mechanical Engineering
- ☐ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) MECHATRONICS ENGINEERING.

SIMTech A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) Ease of use

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 4 (5)
IEC 61499 function block and the development framework is easy to understand	1 2 3 (4) 5

(6) Usefulness

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 (5)
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 (4) 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 (5)
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 (5)
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 (5)
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 (5)
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 (5)
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 (4) 5

(7) Are you interested to receive news, events and training relating to IEC 61499 Technologies? ☑ Yes ☐ No

(8) Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies? ☑ Yes ☐ No

(9) Do you find this event beneficial to your company? ☐ Yes ☐ No

(10) Other Comments: _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The NThe Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) **Please specify the industrial sector to which your company belongs to:**

- ☐ Aerospace
- ☐ Biomedical
- ☐ Electronics
- ☑ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☑ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) **Which category best describes your company's business (you may tick more than one)?**

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☑ Product and Service Providers
- ☐ Product Manufacturing
- ☑ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) **How long did you practice software development on a professional level?**

_____

(4) **What is your area of specialty for work / study?**

- ☐ Mechanical Engineering
- ☑ Electrical Engineering
- ☑ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) _____

SIMTech A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 4 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 4 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☐ Yes ☑ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) **Please specify the industrial sector to which your company belongs to:**

- ☐ Aerospace
- ☐ Biomedical
- ☑ Electronics
- ☐ Engineering Services
- ☐ Food & Healthcare Products
- ☐ Marine & Offshore
- ☐ Precision Engineering
- ☐ Semiconductor
- ☐ Transport & Logistics
- ☐ Others (Please specify) _____

(2) **Which category best describes your company's business (you may tick more than one)?**

- ☐ Component and Sub-Assembly
- ☐ Contract Manufacturing
- ☐ Logistics Providers
- ☐ Maintenance, Repair and Overhaul
- ☐ Material and Technology Suppliers
- ☐ OEM
- ☑ Product and Service Providers
- ☐ Product Manufacturing
- ☐ Solution Providers
- ☐ Tool & Die
- ☐ Others (Please specify) _____

(3) **How long did you practice software development on a professional level?**

N.A.

(4) **What is your area of specialty for work / study?**

- ☐ Mechanical Engineering
- ☐ Electrical Engineering
- ☐ Computer Engineering
- ☐ Computer Science
- ☐ Business / Marketing / Arts
- ☐ Others (Please specify) Industrial Automation

SIMTech
A*STAR
Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 (4) 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 4 (5)

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 (5)
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 (5)
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 (5)
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 (5)
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 (4) 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 (5)
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 (5)
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 (5)

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☑ Yes ☐ No

(9) **Do you find this event beneficial to your company?** ☑ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) Please specify the industrial sector to which your company belongs to:

☐ Aerospace
☐ Biomedical
☐ Electronics
☐ Engineering Services
☐ Food & Healthcare Products
☐ Marine & Offshore
☐ Precision Engineering
☐ Semiconductor
☐ Transport & Logistics
☐ Others (Please specify) _____

(2) Which category best describes your company's business (you may tick more than one)?

☐ Component and Sub-Assembly
☐ Contract Manufacturing
☐ Logistics Providers
☐ Maintenance, Repair and Overhaul
☐ Material and Technology Suppliers
☐ OEM
☐ Product and Service Providers
☐ Product Manufacturing
☐ Solution Providers
☐ Tool & Die
☐ Others (Please specify) _____

(3) How long did you practice software development on a professional level?

_____

(4) What is your area of specialty for work / study?

☐ Mechanical Engineering
☐ Electrical Engineering
☐ Computer Engineering
☐ Computer Science
☐ Business / Marketing / Arts
☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) Ease of use

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 (4) 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 (4) 5

(6) Usefulness

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 (3) 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 (4) 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 (3) 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 (3) 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 (3) 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 (4) 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 (4) 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 (3) 4 5

(7) Are you interested to receive news, events and training relating to IEC 61499 Technologies? ☑ Yes ☐ No

(8) Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies? ☐ Yes ☑ No

(9) Do you find this event beneficial to your company? ☑ Yes ☐ No

(10) Other Comments: Thank you

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The N The Next Generation of Automation Control Standard IEC 61499 Function Block Seminar Feedback Form

Thank you for attending the seminar. We appreciate your feedback in sharing your suggestions and interests. Thank you for your time in completing the feedback form.

About You: (or Attach Name Card Here)

Name: _____
Email: _____
Designation: _____
Department/Section: _____
Company Name: _____
Contact No. _____
Company Address: _____

(1) **Please specify the industrial sector to which your company belongs to:**

☐ Aerospace
☐ Biomedical
☐ Electronics
☑ Engineering Services
☐ Food & Healthcare Products
☐ Marine & Offshore
☐ Precision Engineering
☑ Semiconductor
☐ Transport & Logistics
☐ Others (Please specify) _____

(2) **Which category best describes your company's business (you may tick more than one)?**

☐ Component and Sub-Assembly
☐ Contract Manufacturing
☐ Logistics Providers
☐ Maintenance, Repair and Overhaul
☐ Material and Technology Suppliers
☐ OEM
☑ Product and Service Providers
☐ Product Manufacturing
☑ Solution Providers
☐ Tool & Die
☐ Others (Please specify) _____

(3) **How long did you practice software development on a professional level?**

_____

(4) **What is your area of specialty for work / study?**

☐ Mechanical Engineering
☑ Electrical Engineering
☑ Computer Engineering
☐ Computer Science
☐ Business / Marketing / Arts
☐ Others (Please specify) _____

SIMTech
A*STAR

Fax (65) 6791 6377 to: Mr. Goh Kiah Mok
Singapore Institute of Manufacturing Technology (SIMTech)

The purpose of these two lists with level of ranking are to receive the feedback for the usefulness of the IEC 61499 Function Block Standard and development framework such as development tools, methodologies and concept. (Please circle the numbers between 1-5, 1 being Strongly Disagree & 5 being Strongly Agree)

(5) **Ease of use**

	Rank levels
IEC 61499 function block and the development framework is easy to use	1 2 3 4 5
IEC 61499 function block and the development framework is easy to understand	1 2 3 4 5

(6) **Usefulness**

	Rank levels
IEC 61499 function block and the development framework may helps me in my current or future course of work	1 2 3 4 5
IEC 61499 function block and the development framework is versatile for current or future needs	1 2 3 4 5
I will recommend IEC 61499 function block and the development framework to my organisation, colleagues and friends	1 2 3 4 5
I will use the IEC 61499 function block and the development framework if I need to do programming	1 2 3 4 5
It is easy to implement changes or modify using IEC 61499 function block	1 2 3 4 5
There are a lot of potential for IEC 61499 function block and development framework	1 2 3 4 5
IEC 61499 function block and development framework will help to simplify application development	1 2 3 4 5
IEC 61499 function block and development framework will allow more stakeholders to participate in design of the application or system	1 2 3 4 5

(7) **Are you interested to receive news, events and training relating to IEC 61499 Technologies?** ☑ Yes ☐ No

(8) **Would you like us to follow up with a visit to your company to explain more details about IEC 61499 Technologies?** ☑ Yes ☐ No

(9) **Do you find this event beneficial to your company?** ☐ Yes ☐ No

(10) **Other Comments:** _____

Thank you for your invaluable feedback. Please return the completed feedback form directly to:

Mr. Goh Kiah Mok kmgoh@SIMTech.a-star.edu.sg or Fax to (65) 6791-6377