



Ain Shams University – Faculty of Engineering
Computer and Systems Engineering Department

C-based high-level synthesis using SystemC

A Thesis

Submitted in partial fulfillment of the requirements for the degree of

Master of Computer Engineering

Submitted By

Samer I. Mohamed Aly

B.Sc. Electrical Engineering
(Computer & Systems Engineering)
Ain Shams University, 1998

Supervised By

Prof Dr. Ashraf El-Farghaly Salem
Dr. Ayman Mohamed Wahba

Cairo June 2005

APPROVAL SHEET

Name Samer I. Mohamed Aly
Thesis C-based high level synthesis using SystemC
Degree Master of Computer Engineering

Examiners Committee

Name, Title and Affiliation	Signature
Prof. Dr. Abd El-Monem Abd El-Zaher Wahdan Faculty of Engineering Ain-Shams Univ.	
Prof Dr. Gamal Darwish Faculty of computer science Cairo Univ.	
Prof Dr. Ashraf El-Farghaly Salem Faculty of Engineering Ain-Shams Univ.	
Dr. Ayman Mohamed Wahba Faculty of Engineering Ain-Shams Univ.	

Date: / / 2005

STATEMENT

This dissertation is submitted to Ain Shams University for the degree of Master of Science in Computer Engineering.

The work included in this thesis was carried out by the author at the Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University.

No part of this thesis has been submitted for a degree or qualification at other university or institution.

Name Samer I. Mohamed Aly

Signature

Date / / 2005

ABSTRACT

C-BASED HIGH LEVEL SYNTHESIS USING SYSTEMC

Samer I. Mohamed Aly

Master of Science in Computer Engineering

Ain Shams University, 2005

Silicon technology now allows us to build chips consisting of tens of millions of transistors. This technology promises new levels of system integration onto a single chip but also presents significant challenges to the chip designers who see current design tools and methodologies as inadequate for developing million gates ASICs from scratch. There is considerable pressure to keep design team size and design schedules constant even as design complexities grow.

Design reuse is the use of pre-designed and pre-verified cores and taken to be the most promising opportunity to bridge the gap between the available gate count and designer productivity.

In this thesis, we focus on using SystemC, a new modeling methodology based on C++, to describe an effective methodology for creating reusable designs to be used in a system-on-chip (SOC) and shows how through adapting linting rules and guidelines, the developer will be able to create reusable designs efficient in terms of quality and with higher performance.

For this thesis, we design a complete synthesis flow; start by parsing the input SystemC, estimate the logic and area consumed by design, map the RTL logic to Xilinx technology and write the corresponding EDIF gate netlist to be taken as an input for Xilinx Place and Route tool. Also the system gets benefit from the existing VHDL-based tool through transferring the input synthesizable SystemC models into corresponding synthesizable VHDL models.

Keywords: SoC, SystemC, Design reuse, ASIC.

ACKNOWLEDGMENT

First of all I want to thank God for everything and one of these things is supporting me to share in publishing this thesis as a small contribution for our nation development and progress.

Second, I wish to give my deep appreciation to my teacher, instructor and advisor Prof. Dr Ashraf Salem who stands beside me from the beginning through technical support based on his familiarity with ideas and needs of the thesis, Finally I wish to thank him for his continuous effort and time to produce this thesis in an efficient and professional way.

Third, I wish to thank Dr. Ayman Wahba whose support with whatever technical information is an added value and contribute to publish this research in the best way.

Forth, I want to thank my company, my managers and colleagues there, for their support and for using company resources, tools and valuable time.

Finally, I'd like to thank my family who deserve more than thanks for their support and encouragement all over my life.

TABLE OF CONTENTS

Approval Sheet	1
Statement	2
Abstract.....	3
Acknowledgment	4
Table of Contents.....	5
List of Figures	9
List of Tables.....	12
Glossary.....	13
Chapter 1	15
Introduction	15
1.1 SystemC Modeling Methodology	15
1.2 Linting methodology	15
1.3 High level synthesis methodology....	16
1.4 Thesis summary.....	16
1.5 Organization of the thesis....	17
Chapter 2	18
SystemC modeling Methodology	18
2.1 Introduction.....	18
2.2 System constructs description	19
2.2 .1 Modules	19
2.2.2 Ports.....	20
2.2.3 Signals.....	22
2.2.4 Reading and wiring ports and signals.....	23
2.2.5 Processes.....	24
2.2.6 Reading and wiring processes.....	26
2.2.7 Types of processes	26
2.2.8 Data types	28
2.3 SystemC synthesizable subset	28
2.3.1 Nonsynthesizable SystemC constructs	28
2.3.2 Nonsynthesizable data types.....	33
2.3.3 Recommended data types for synthesis.....	33
2.4 SystemC advantages	34
Chapter 3	35
SystemC linting Methodology	35
3.1 Overview of coding guidelines.....	35
3.2 Lexical guidelines	36
3.2.1 Rules for naming conventions.....	36
3.2.2 Include header in source files	37

3.2.3 Use comments	38
3.2.4 Keep commands in separate lines.....	38
3.2.5 Line length.....	38
3.2.6 Indentation	39
3.2.7 Avoid SystemC reserved words	40
3.2.8 Port ordering.....	40
3.2.9 Port maps	41
3.2.10 SystemC module and methods implementation	41
3.2.11 Use functions.....	41
3.2.12 Use loops and arrays	42
3.2.13 Avoid use hard-coded numeric values.....	42
3.3 Coding guidelines for synthesis	43
3.3.1 Avoid latches.....	43
3.3.2 Avoid combinational feedback.....	45
3.3.3 Specify complete sensitivity list	45
3.3.4 Case statement versus if-else statement.....	46
3.4 Partitioning for synthesis.....	48
3.4.1 Register all outputs.....	48
3.5 Practical implementation	49
3.5.1 ScLint tool features	49
3.5.2 ScLint data structures	49
3.5.3 ScLint parsing rules.....	52
3.5.4 ScLint linting algorithms	54
3.5.5 ScLint error/warning messages	55
3.5.6 ScLint linting results	55
Chapter 4	63
SystemC Synthesis Methodology	63
4.1 Introduction.....	63
4.2 Behavioral synthesis process.....	65
4.2.1 CDFG generation	65
4.2.2 Resource allocation.....	66
4.2.3 Scheduling	67
4.2.4 Register allocation	69
4.2.5 Register sharing.....	69
4.2.6 Library binding	69
4.2.6.1 Definition.....	69
4.2.6.2 Goal of library binding.....	69
4.2.6.3 Importance of library binding.....	69
4.2.6.4 Practical approaches to library binding	70
4.2.6.5 Library binding formulation.....	70
4.2.6.5.1 Terminology.....	70
4.2.6.5.2 Conditions for network covering problem.....	71

4.2.6.5.3 Algorithms for covering problem.....	71
4.2.6.6 Structural approach	71
4.2.6.6.1 Covering algorithm based on structural approach	71
4.2.6.6.2 Tee-based matching problem	74
4.2.6.6.3 Optimal tree covering	75
4.2.6.6.4 Disadvantages of structural matching.....	76
4.2.6.7 Boolean approach.....	76
4.2.6.7.1 Covering algorithm based on Boolean approach	76
4.2.6.7.2 Conclusions.....	77
4.2.6.8 Field Programmable Gate Array (FPGA) mapping.....	78
4.2.6.8.1 Introductions	85
4.2.6.8.2 Binding for different FPGA architectures	86
4.2.6.9 Rule based library binding.....	87
4.2.6.9.1 Introductions	87
4.2.6.9.2 Comparison of algorithmic and rule-based binding.....	87
4.2.7 Data path and state machine extraction.....	87
4.2.8 Net listing.....	88
4.3 Area estimation techniques	91
4.3.1 basic concepts.....	91
4.3.2 area estimation algorithm	92
4.3.2.1 CLBs count.....	92
4.3.2.2 Using Boolean equations	94
4.3.2.3 Using SOP form	94
4.3.2.4 Using RTL description	95
4.4 Practical implementation	97
4.4.1 ScMapper tool features	97
4.4.2 Functionality.....	98
4.4.3 ScMapper data structures.....	98
4.4.4 SystemC conversion into VHDL.....	103
4.4.4.1 ScMapper parsing rules.....	103
4.4.4.2 ScMapper conversion algorithm	103
4.4.4.3 ScMapper SystemC-to-VHDL conversion results	103
4.4.5 SystemC area and logic estimation.....	113
4.4.5.1 ScMapper logic estimation algorithm	113
4.4.5.2 ScMapper area estimation algorithm.....	114
4.4.6 SystemC mapper.....	116
4.4.6.1 ScMapper mapper limitations.....	116
4.4.6.2 ScMapper mapper algorithm	116
4.4.6.3 ScMapper mapper advantages.....	117
4.4.6.4 EDIF netlist writer.....	118
Chapter 5	126
Conclusions & Future Work	126

5.1 Conclusions.....	126
5.2 Future Work	127
References	128
Appendix A:	133
A.1 ScLint BNF rules	133
Appendix B:	138
B.1 ScLint 8-bit adder Code Listing	138

LIST OF FIGURES

<i>Chapter-Number</i>		<i>Page</i>
2-1	SystemC modeling methodology	18
2-2	Module configuration block.....	18
2-3	Module syntax declaration.....	19
2-4	Module ports.....	20
2-5	Ports syntax declaration.....	20
2-6	Module signals.....	21
2-7	Signals syntax declaration.....	22
2-8	Reading and writting ports and signals declaration.....	22
2-9	Process syntax declaration.....	24
2-10	Combinational process syntax declarations.....	24
2-11	Sequential process syntax declarations.....	25
2-12	Method process syntax declarations.....	26
3-1	Example for naming conventions.....	37
3-2	Example for the header in the SystemC code.....	38
3-3	Example For line length and indentation though process body...	39
3-4	Example for port ordering and using comments.....	41
3-5	Example for using functions within SystemC code.....	41
3-6	Example for using for loops in the within functions.....	42
3-7	Example for hard-coded numeric values.....	42
3-8	Example shows how designer can avoid inferring latches.....	44
3-9	Example shows how sensitivity list handled through process...	45
3-10	Example shows how sensitivity list handled through sequential process.....	46
3-11	Example Shows if-else versus case statements.....	47
3-12	Good Example: All output signals are registered.....	48
3-13	Data type data structure.....	49
3-14	Port type data structure	50
3-15	Signal type data structure	51
3-16	Sensitivity list type data structure	51
3-17	Process type data structure	51
3-18	Vector port type data structure	52
3-19	Example 1 Linting	56
3-20	Example 2 Linting	57
3-21	Example 3 Linting	58
3-22	Example 4 Linting	59
3-23	Example 5 Linting	59

3-24	Example 6 Linting	60
3-25	Example 7 Linting	61
3-26	Example 8 Linting	62
4-1	Behavioral synthesis process.....	65
4-2	Binding of operations to components.....	67
4-3	NAND network.....	72
4-4	NAND2 network.....	73
4-5	Trivial covering.....	73
4-6	Better covering.....	74
4-7	Virtex architecture.....	79
4-8	Virtex Input Output Block IOB.....	80
4-9	2-Slice Virtex CLB.....	81
4-10	CLB sliced diagram for Virtex family..	83
4-11	Virtex Local Routing.....	84
4-12	4-bit counter VHDL source code.....	89
4-13	Leonardo Spectrum gate level RTL.....	89
4-14	4-bit counter mapping targeting Virtex family.....	90
4-15	Data type data structure	98
4-16	Port type data structure	99
4-17	Signal type data structure	99
4-18	Sensitivity list type data structure	100
4-19	Process type data structure	100
4-20	Vector port type data structure	101
4-21	LUT type data structure	101
4-22	Target type data structure	102
4-23	Net type data structure	102
4-24	Example 1 SystemC.....	104
4-25	Example 1 VHDL.....	104
4-26	Example 2 SystemC.....	105
4-27	Example 2 VHDL.....	106
4-28	Example 3 SystemC.....	107
4-29	Example 3 VHDL.....	108
4-30	Example 4 SystemC.....	108
4-31	Example 4 VHDL.....	109
4-32	Example 5 SystemC.....	110
4-33	Example 5 VHDL.....	111
4-34	Logic estimator input.....	113
4-35	Logic estimator output.....	113
4-36	Area estimator input.....	116
4-37	Area estimator output.....	116
4-38	EDIF header from ScMapper.....	118
4-39	Primitives in the EDIF.....	118

4-40	SystemC mapper example.....	121
4-41	EDIF output from ScMapper.....	125

LIST OF TABLES

<i>Chapter-Number</i>		<i>Page</i>
2-1	Non-synthesiable SystemC subsets.....	32
2-2	Synthesizable SystemC data types.....	34

GLOSSARY¹

ASIC. Application Specific Integrated Circuit. An IC created for a dedicated purpose.

Design reuse. The ability to reuse portions of existing circuitry in new designs. When design cores are sold to a customer for reuse, it is sometimes referred to as intellectual property (IP).

HDL: Hardware Design Language. A method of writing a textual description of a circuit's behavior that becomes the specification for that circuit or module. HDL can be simulated and subsequently synthesized into gates. The most common HDLs are Verilog and VHDL.

Hierarchical design. The creation of a higher level of abstraction, often represented by a black box that can be defined later (top down design) or first (bottom up design).

IP (Intellectual Property). A reference to the marketing of pre-designed electronic cores for use in a customer's own designs. Cores can either be soft, such as HDL descriptions not yet synthesized, or hard already deployed in silicon or as gate level descriptions. Soft cores have the advantage that they are tool and foundry independent.

Logic simulator. A software application that simulates the operation of digital circuitry in order to perfect its operation before it's actually produced or prototyped.

RTL. Register Transfer Level. A style of HDL that describes circuit behavior in a structured fashion that more closely resembles logic elements. Most synthesis tools require HDL to be in RTL format before they can process it.

System on Chip (SoC). Advances in silicon processing technology enable integration of ever more complex systems on a single chip. These so-called Systems on Chip contain dedicated hardware components, programmable processors, memories, requiring not only the design of digital hardware but also the design of embedded software.

¹ This Glossary focuses mainly on EDA terminology presented in chapter 1.