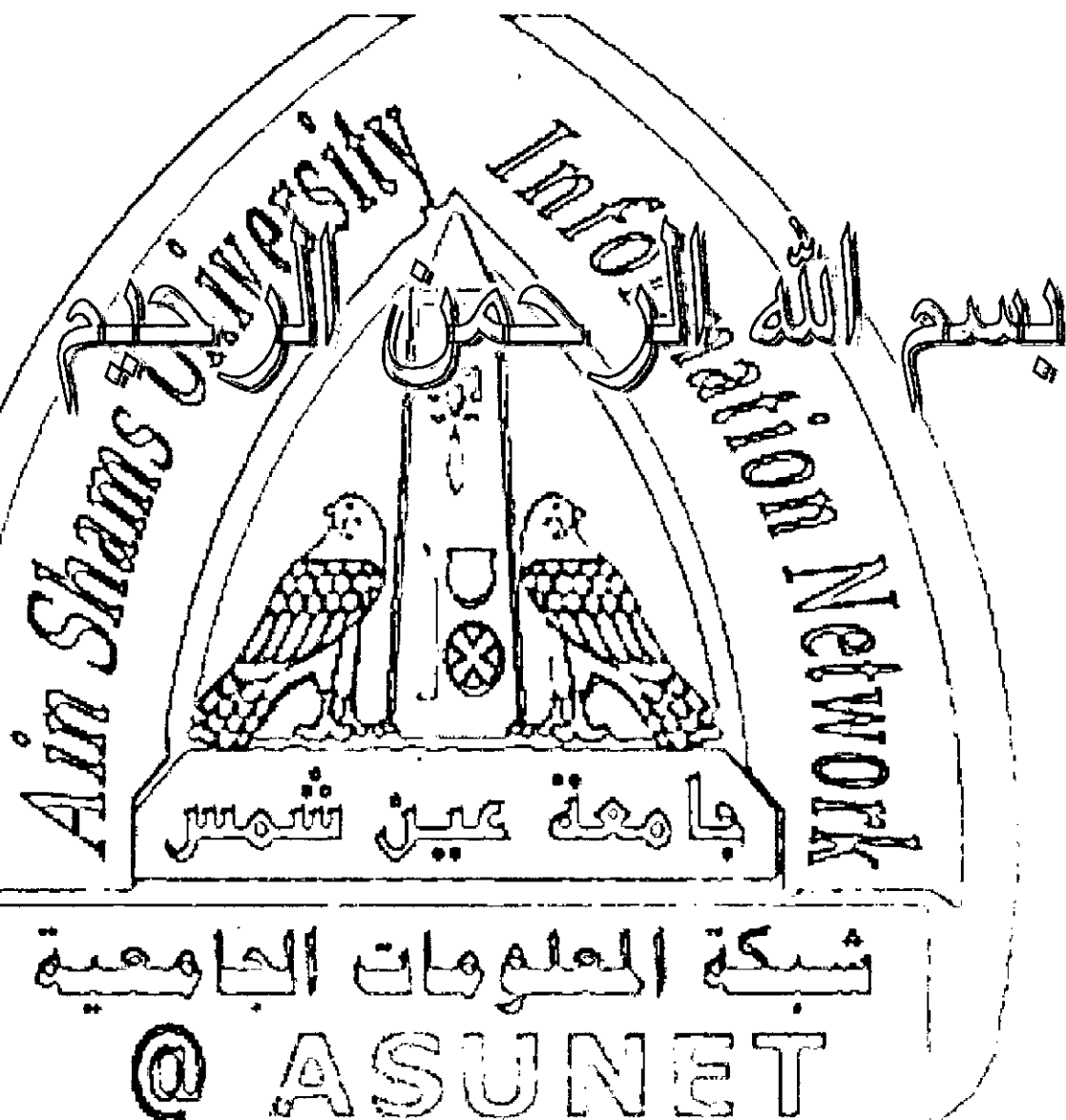




شبكة المعلومات الجامعية





شبكة المعلومات الجامعية التوثيق الالكتروني والميكروفيلم



شبكة المعلومات الجامعية

جامعة عين شمس

التوثيق الالكتروني والميكروفيلم

قسم

نقسم بالله العظيم أن المادة التي تم توثيقها وتسجيلها
على هذه الأفلام قد أعدت دون أية تغيرات



يجب أن

تحفظ هذه الأفلام بعيدا عن الغبار

في درجة حرارة من ١٥-٢٥ مئوية ورطوبة نسبية من ٢٠-٤٠%

To be Kept away from Dust in Dry Cool place of
15-25- c and relative humidity 20-40%

بالرسالة صفحات لح



شبكة المعلومات الجامعية
@ ASUNET

بعض الوثائق
شبكة جامعة
Ain Shams University
International Network
جامعة عين شمس

شبكة المعلومات الجامعية
@ ASUNET

621,381

SPEEDING UP SYSTEMC SIMULATIONS USING PROCESS SPLITTING

By

Youssef Nagy Naguib

B.Sc. in Electronics and Communications Engineering – Cairo University

A Thesis Submitted to the

Faculty of Engineering at Cairo University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Electronics and Electrical Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

2007

CPVVP

SPEEDING UP SYSTEMC SIMULATIONS USING PROCESS SPLITTING

By

Youssef Nagy Naguib

B.Sc. in Electronics and Communications Engineering – Cairo University

A Thesis Submitted to the

Faculty of Engineering at Cairo University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Electronics and Electrical Communications Engineering

Supervised by

Dr. Serag El Din Habib

Professor, Faculty of Engineering, Cairo University

Dr. Rafik Safwat Guindi

Associate Professor, Faculty of Engineering, Cairo University



Faculty of Engineering, Cairo University

Giza, Egypt

2007

SPEEDING UP SYSTEMC SIMULATIONS USING PROCESS SPLITTING

By

Youssef Nagy Naguib

B.Sc. in Electronics and Communications Engineering – Cairo University

A Thesis Submitted to the

Faculty of Engineering at Cairo University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Electronics and Electrical Communications Engineering

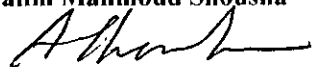
Approved by the examining committee


Dr. Ashraf Mohamed El Farghaly

Ain Shams University, member.

Dr. Abd El Halim Mahmoud Shousha

Cairo University, member.

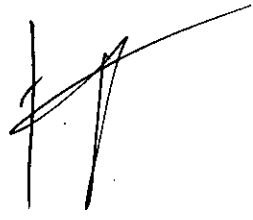

Dr. Serag El Din Habib

Cairo University, thesis advisor

Faculty of Engineering, Cairo University

Giza, Egypt

2007



Abstract

This theses presents a new approach that can be used to speed up SystemC simulations by automatically optimizing the model for simulation. The work addresses the inefficiency of the standard SystemC scheduler that may lead, in some situations, to unnecessary wake-up calls, as well as unnecessary code execution.

Previous solutions presented for this issue recommend remedies by applying changes to the SystemC simulation engine and provide extra non-standard SystemC descriptions to the model. The main drawback of this approach is that the simulation model cannot be simulated on any SystemC engine, and changes to the SystemC standard simulation engine is not a preferred solution. Moreover the previously presented solution does not solve the problem of unnecessary code execution.

The method presented in this work analyzes the SystemC model to automatically extract signal dependencies based on a set of rules. The rules are based on the concepts of software slicing. This information is then used to split large processes into smaller ones. The code of the original process is divided among the new smaller processes. By applying process splitting on a SystemC model, only the code that is necessary to evaluate the output signal is put in the new small processes, and is executed only when evaluating output signals is necessary.

Process splitting is performed by a tool – SplitPro – which generates an optimized code that can be run on any standard SystemC engine. The SplitPro tool analyzes the SystemC modules in the simulation model and generates the dependency tree of each output port. The dependency tree of each output port determines the input-output relationships and the necessary code for evaluating each output. It then checks if it is possible to split the code inside one process among a number of smaller processes without changing the behavior of the system. This way, whenever a process is called, it will only execute the necessary code to evaluate the required output.

The SplitPro tool was used to analyze the simulation models of an Alpha super scalar processor and a 4-port switch. The results of these simulations are analyzed and presented in the theses.

Acknowledgements

*I have the pleasure and honor to express great gratitude to my supervisors
Dr. Serag El Din Habib and Dr. Rafik Safwat Guindi
for encouragement and unfailing guidance which
enabled achieving the research goals.*

Grateful thanks are also due to Eng. Ahmed Sabry for his help and advice.

The role of Eng. Mina Aziz and Eng. John Selim is also gratefully acknowledged.

*Last, but not least, a special expression of gratitude and appreciation goes to
my family for their patience and generous support.*

Table of Contents

Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	viii
List of Tables.....	x
Chapter 1 Introduction.....	1
Chapter 2 Simulation of Digital Systems	5
2.1 Analog Simulation.....	5
2.2 Digital Simulation	6
2.2.1 Event Simulation versus Cycle Simulation	8
2.2.2 Compiled and Interpreted Simulators.....	8
2.3 Simulation Levels and Methodologies	9
2.3.1 Design Level.....	9
2.3.2 System Level Simulation Methodologies.....	11
Chapter 3 The SystemC Framework	13
3.1 Language Constructs	13
3.1.1 Structure and Connectivity	13
3.1.2 Functionality and Concurrency	15
3.2 Simulation	15
3.3 Synthesis.....	17
3.4 Extending the SystemC Framework.....	18
3.4.1 Improving Simulation Speed.....	18
3.4.2 Mixed Signal Simulation.....	19
3.4.3 Synthesizing SystemC.....	20
Chapter 4 The SystemC Scheduler.....	22
4.1 Elaboration and Simulation	22
4.2 The Scheduler.....	22
4.2.1 Scheduler Sets	23
4.2.2 Scheduler phases	24
4.3 The Inefficiency in the SystemC Scheduler	26
4.4 The FastSysC Engine	27
Chapter 5 Process Splitting	29
5.1 Overview of Program Slicing.....	29

5.1.1 HDL Slicing	31
5.1.2 SystemC Slicing	31
5.2 Split Processes	31
5.3 Splitting Inside the Same Dependency Tree	35
5.4 Limitations of Process Splitting	36
5.5 Parallel Simulation of SystemC	38
Chapter 6 Signal Dependency Extraction	40
6.1 Overview	40
6.2 Dependency Types	41
6.3 Building Dependency Trees from Process Code	42
6.4 Data Structure of Dependency Trees	45
Chapter 7 SplitPro	48
7.1 Analyzing C++ Code	48
7.1.1 Parsing Phase	49
7.1.2 Abstract Syntax Tree Generation	49
7.2 Parsing SystemC Modules	51
7.3 Example	53
7.4 Implementation	55
7.4.1 The GP Engine	55
7.4.2 Vector class	56
7.4.3 SplitPro Classes	57
Chapter 8 Practical Results	60
8.1 Alpha Superscalar Processor	60
8.1.1 Simulation Results	64
8.2 4-Port Switch	69
Chapter 9 Conclusion and Possible Future Work	73
Appendix A Code Samples	75

List of Figures

Figure 3.1 SystemC modules connected using ports and signals, taken from [12].	14
Figure 3.2 SystemC modules connected using a channel and interfaces, taken from [12].	15
Figure 3.3 Overview of SystemC simulation methodology, taken from [12].	16
Figure 3.4 Synthesis flow in the SystemC framework, taken from [12].	17
Figure 3.5 Nested function calls in the SystemC framework, taken from [1].	19
Figure 3.6 ODETTE synthesizer, taken from [14].	21
Figure 4.1 SystemC scheduler, taken from [2].	25
Figure 4.2 A system with two codependent processes.	26
Figure 5.1 The system of Figure 4.2 after applying process splitting.	32
Figure 5.2 Dependency graph of processes.	34
Figure 5.3 Dependency tree of an output signal with one of the input nodes parent to another input node.	36
Figure 5.4 Process P which writes to m outputs and have $n+1$ inputs.	37
Figure 5.5 Process P_x sensitive to I_x and C .	37
Figure 6.1 Dependency tree.	43
Figure 6.2 SystemC code of a module to generate a dependency tree.	44
Figure 6.3 The function <i>ExternalControl</i> rewritten in a different form.	45
Figure 6.4 Flow chart for dependency tree generation.	47
Figure 7.1 C++ code before parsing.	49
Figure 7.2 Stream of tokens generated after parsing.	49
Figure 7.3 Example C code to generate AST.	50
Figure 7.4 AST of the code in Figure 7.3.	51
Figure 7.5 Process P_{xy} which evaluates both X and Y .	52
Figure 7.6 Process P_{xy} split to processes P_{xy_x} which evaluates X and P_{xy_y} which evaluates Y .	52
Figure 7.7 Process P_{xy} which evaluates X and Y , input to SplitPro tool.	53
Figure 7.8 Process P_{xy} split to processes P_{xy_X} which evaluates X and P_{xy_Y} which evaluates Y , generated by SplitPro tool.	54
Figure 8.1 Code to read time stamp counter.	61
Figure 8.2 GTKWave displaying waveforms of internal signals of a SystemC module.	63
Figure 8.3 The simulation time of DRAM process that can be split and provide speedup gain after splitting.	64
Figure 8.4 The DRAM process before and after splitting.	65

Figure 8.5 The simulation time of InstructionBroadcaster process that can be split and does not provide speedup after splitting.	66
Figure 8.6 The InstructionBroadcaster process before and after splitting.	67
Figure 8.7 The simulation time of the FunctionalUnit process that can not be split.	68
Figure 8.8 The FunctionalUnit process without splitting.	68
Figure 8.9 The normalized simulation time for all processes that can be split and provide speedup gain.	69
Figure 8.10 The simulation time of the 4-port switch executable.	70
Figure 8.11 The core process of the 4-port switch before and after splitting.	71

List of Tables

Table 2.1 The simulation slowdown assuming the processor is simulating a model of itself, taken from [27].	10
Table 4.1 Sets of the SystemC scheduler.	23
Table 4.2 Phases of the SystemC scheduler.	24
Table 4.3 Delta cycles details of the system in Figure 4.2 running on the reference simulator.	27
Table 4.4 Delta cycles details of the system in Figure 4.2 running on the FastSysC engine.	28
Table 5.1 Delta cycles details of the system in Figure 4.2 running on the reference simulator	32
Table 5.2 Delta cycles details of the system of Figure 5.1 running on a signal-dependency aware simulator.	34
Table 6.1 Sets used in building dependency trees of a SystemC module.	45
Table 8.1 Alpha processor SystemC modules and there functionality.	61
Table 8.2 Simulation times in milliseconds and average gain in simulation speed.	72