DESIGN OF ASYNCHRONOUS PROCESSOR

PUAH WEI BOO

FK 2001 60

DESIGN OF ASYNCHRONOUS PROCESSOR

By

PUAH WEI BOO

Thesis Submitted in Fulfilment of the Requirement for the Degree of Master of Science in the Faculty of Engineering Universiti Putra Malaysia

July 2001

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Master of Science

DESIGN OF ASYNCHRONOUS PROCESSOR

By

PUAH WEI BOO

July 2001

Chairman : Dr. Bambang Sunaryo Suparjo

Faculty : Engineering

There has been a resurgence of interest in asynchronous design recently. The renewed interest in asynchronous design results from its potential to address the problem faced by the synchronous design methodology. In asynchronous methodology, there is no global clock controlling the synchronization of a circuit; instead, the data communication between each functional unit is completed through local request-acknowledge handshake protocol.

The growth in demand of high performance portable systems has accelerated asynchronous logic design technique which can offers better performance and lower power consumption especially in the development of the asynchronous processor for mobile and portable application.

In this thesis, the design and verification of an 8-bit asynchronous pipelined processor is presented. The developed asynchronous processor is based on Harvard architecture and uses Reduced Instruction Set Computer (RISC) instruction set architecture. 24 instructions are supported by the processor including register, memory, branch and jump operations. The processor has three-stage pipelining i.e. fetch, decode and execution pipeline. Micropipelines framework with 2-phase signalling protocol and bundled-data approach is employed in designing complex and powerful asynchronous control circuits for the processor. Very High Speed Integrated Circuit Hardware Description Language (VHDL) is used to design and construct all parts of the asynchronous processor. Simulation, synthesis and verification of the processor are carried out using MAX+PLUS II software.

The simulation results have demonstrated that the developed 8-bit asynchronous RISC processor is working correctly using current Field Programmable Gate Array (FPGA) technology. This processor employed 903 logic cells and has 6144 memory bits for instruction and data memory. Each of the processor subsystem can operates at different cycle time, thus enable an asynchronous processor achieving 11.95MHz average speed performance.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia
sebagai memenuhi keperluan untuk ijazah Master Sains

REKABENTUK PEMPROSES ASINKRONI

Oleh

PUAH WEI BOO

Julai 2001

Pengerusi	**: Dr. Bambang Sunaryo Suparjo**
Fakulti	**: Kejuruteraan**

Baru-baru ini terdapat kebangkitan minat dalam dunia rekabentuk asinkroni. Pembaharuan minat ini adalah disebabkan oleh potensinya dalam mengatasi kepincangan yang dihadapi oleh perkaedahan rekabentuk segerak. Dalam perkaedahan asinkroni, tiada terdapat pemasa yang global bagi mengawal penyelarasan sesuatu litar; sebaliknya, komunikasi data antara unit kefungsian dilakukan melalui protokol perjabatan tangan permintaan-pemberitahuan tempatan.

Perkembangan pesat dalam permintaan sistem mudah-alih yang berkeupayaan tinggi telah menyemarakkan lagi teknik rekabentuk logik asinkroni di mana ia boleh menawarkan keupayaan yang lebik baik dan menjimatkan penggunaan tenaga terutamanya dalam pembangunan pemproses asinkroni untuk applikasi mudah-alih.

Tesis ini membentangkan rekabentuk dan penentusahan pemproses saluran asinkroni 8-bit. Pemproses asinkroni yang dibina adalah berdasarkan senibena Harvard dan menggunakan set arahan RISC. Pemproses ini menyokong 24 pelaksanaan arahan yang merangkumi operasi pendaftaran, ingatan, cabangan dan lompatan. Terdapat tiga saluran perhubungan dalam pemproses ini iaitu, saluran pengambilan, penyahkod dan perlaksanaan. Rangka "Micropipelines" dengan protokol isyarat dua-fasa dan pendekatan ikatan-data digunakan dalam merekabentuk litar kawalan asinkroni yang rumit dan berkeupayaan tinggi untuk pemproses. Semua bahagian pemproses asinkroni direkabentuk dan dibina menggunakan bahasa pengaturcaraan VHDL. Simulasi, sintesis dan penentusahan pemproses ini dijalankan dengan bantuan perisian MAX+PLUS II.

Keputusan simulasi telah menunjukkan pemproses asinkroni 8-bit yang dibina dapat berfungsi dengan baik menggunakkan teknologi FPGA yang terkini. Pemproses ini menggunakan sebanyak 903 sel logik dan 6144 bit ingatan untuk ingatan arahan dan data. Keupayaan setiap subsistem dalam pemproses beroperasi pada kitaran masa yang berlainan membolehkannya mencapai kelajuan purata 11.95MHz .

ACKNOWLEDGEMENTS

A sincere appreciation is delivered to my project supervisors, Dr. Bambang Sunaryo Suparjo, Mr. Rahman Wagiran and Dr. Roslina Sidek for their invaluable guidance, constructive suggestions and encouragement throughout the duration of this project.

I also wish to extend my deepest personal thanks to my dearly coursemates to whom I owe my sincere appreciation. They are G. H. Tan, C.L Lee, Lini Lee and Philip Tan, which have indeed made my project more interesting and meaningful.

Lastly, I would like to express my sincere appreciation to my family for their undying love, patience and supports, which have enable me to complete the project successfully.

I certify that an Examination Committee met on 12th July 2001 to conduct the final examination of Puah Wei Boo on his Master of Science thesis entitled "Design of Asynchronous Processor" in accordance with Universiti Pertanian Malaysia (Higher Degree) Act 1980 and Universiti Pertanian Malaysia (Higher Degree) Regulation 1981. The committee recommends that the candidate be awarded the relevant degree. Members of the Examination Committees are as follows:

Norman Mariun, Ph.D.
Associate Professor,
Faculty of Engineering,
Universiti Putra Malaysia
(Chairman)

Bambang Sunaryo Suparjo, Ph.D.
Faculty of Engineering,
Universiti Putra Malaysia
(Member)

Roslina Sidek, Ph.D.
Faculty of Engineering,
Universiti Putra Malaysia
(Member)

Rahman Wagiran, M.Sc.
Faculty of Engineering,
Universiti Putra Malaysia
(Member)

[illegible]

MOHD. GHAZALI MOHAYIDIN, Ph.D.
Professor/Deputy Dean of Graduate School,
Universiti Putra Malaysia.

Date: 16 JUL 2001

This thesis submitted to the Senate of Universiti Putra Malaysia has been accepted as fulfilment of the requirement for the degree of Master of Science.

AINI IDERIS, Ph.D.
Professor,
Dean of Graduate School
Universiti Putra Malaysia

Date:

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UPM or other institutions.

[illegible]

(PUAH WEI BOO)

Date: 14 · 7 · 2001

TABLE OF CONTENTS

		Page
ABSTRACT		ii
ABSTRAK		iv
ACKNOWLEDGEMENTS		vi
APPROVAL SHEETS		vii
DECLARATION FORM		ix
LIST OF TABLES		xiii
LIST OF FIGURES		xiv
LIST OF ABBREVIATIONS		xix
CHAPTER		
1	**INTRODUCTION**	1
	Research Objective	3
	Thesis Organization	3
2	**ASYNCHRONOUS LOGIC**	5
	Synchronous versus Asynchronous Logic	5
	Advantages of Asynchronous Logic	6
	Delay Modelling	9
	Handshake Protocols	10
	Two-Phase Signalling	11
	Four-Phase Signalling	13
	Choice of Signalling Protocol	14
	Asynchronous Data Encoding	15
	Micropipelines	17
	Event Control Modules	18
	An Event-Controlled Storage Element	21
	Event Register	22
	Micropipelines Stage	26
3	**COMPUTER ARCHITECTURE**	28
	Von Neumann Architecture	28
	Harvard Architecture	29
	CISC Architecture	30
	RISC Architecture	33
	Benefits of RISC Design	35
	Pipelining	37
	Synchronous and Asynchronous Pipelines	40
	Asynchronous Processors	41

4	**VHDL**	48
	Advantages of VHDL	49
	Design Entity	51
	VHDL Description Style	53
	Behavioral Modelling	53
	Dataflow (Register Transfer Level) Modelling	55
	Structural (Gate Level) Modelling	55
	VHDL Design Flow	58
5	**METHODOLOGY**	62
	MAX+PLUS II Development Tools	62
	Processor Architecture Design	66
	Processor Specification	66
	Instruction Set Design	69
	Processor Datapath Design	75
	Processor Controller Design	83
	Functional Blocks Design	88
	Logic Synthesis	90
	Placement and Routing	92
	Simulation and Verification	94
	Chip Prototype	96
6	**MICROPIPELINES BASIC BUILDING BLOCKS DESIGN**	97
	FPGA Event Control Logic Modules Design	97
	Exclusive OR Gate (XOR)	98
	Muller-C Element	99
	TOGGLE Module	101
	SELECT Module	103
	CALL Module	104
	ARBITER Module	106
	FPGA Event Register	108
7	**IMPLEMENTATION OF AN 8-BIT ASYNCHRONOUS RISC PROCESSOR**	111
	Instruction Fetch Unit	111
	Instruction Fetch Registers	113
	Asynchronous Multiplexers	116
	Asynchronous Incrementer Unit	118
	Fetch Event Controllers	119
	Asynchronous Instruction Memory	121
	Instruction Decode Unit	123
	Instruction Decoder	125
	Asynchronous Processor Controller	126
	Select_JAL Circuit	127
	Asynchronous Register File	128
	Asynchronous Sign-Extender Unit	132
	Asynchronous Multiplexer	133

Asynchronous Branch Unit 134
Decode Event Controller 136
Bypass Branch and Jump Control Circuit 139
Instruction Execution Unit 140
Asynchronous ALU 140
Status Registers 144
Asynchronous Multiplexers 145
Execution Event Controller 145
Asynchronous Data Memory 148
Completed 8-bit Asynchronous RISC Processor 150

8 **RESULT AND DISCUSSION** 152
8-bit Asynchronous RISC Processor Verification 152
Instruction Fetch Unit 154
Instruction Decode Unit 161
Instruction Execution Unit 167
Processor Performance 171
Area 171
Speed 174
Bundled Data Path Element 175
Hazards 176

9 **CONCLUSION AND FURTHER DEVELOPMENT** 178

REFERENCES 180

APPENDICES

A Micropipelines Event Control Logics VHDL Model 183
B Asynchronous Processor Datapath Circuits VHDL Model 203

BIODATA OF THE AUTHOR 230

LIST OF TABLES

Table		Page
2.1	Comparison between Single-rail and Dual-rail Data Encoding	16
3.1	Asynchronous Microprocessors	45
5.1	8-bit Asynchronous RISC Processor Specification	67
5.2	Register File	68
5.3	Status Registers	69
5.4	Instruction Set Summary	74
5.5	Control Signals Function	84
5.6	8-bit Asynchronous RISC Processor Controller Truth Table	85
6.1	XOR Gate Truth Table	98
6.2	Event Registers	110
7.1	Fetch Unit Registers	114
7.2	4-to-1 Multiplexer Output	117
7.3	Instruction Operands Event Control Signals	139
7.4	ALU Operations	142
7.5	ALU Input Enabled Events	146
8.1	Summarize Results of Instruction Fetch Unit	160
8.2	Event Control Logics Area Consumption	172
8.3	Event Registers Area Consumption	172
8.4	Comparison of Normal and Bundled Datapath Area Consumption	172
8.5	Asynchronous Control and Datapath Circuits Area Consumption	173
8.6	Processor Subsystems Area Consumption	173
8.7	Average Speed of the 8-bit Asynchronous RISC Processor	175

LIST OF FIGURES

Figure		Page
2.1	Synchronous Circuit	5
2.2	Asynchronous Circuit	6
2.3	General Request-Acknowledge Communication	11
2.4	Data-validity Scheme for Two-phase Signalling Protocol	12
2.5	Data-validity Schemes for Four-phase Signalling Protocol	13
2.6	Two-phase Bundled Data Protocol	18
2.7	Bundled Data Interface Constraints	18
2.8	Event Logic Library	19
2.9	Capture-Pass Latch	22
2.10	Event Register	23
2.11	Normally Closed Event Register	25
2.12	FIFO Buffer (Micropipeline without Processing)	26
2.13	Micropipeline with Processing Logic	27
3.1	Von Neumann Architecture	29
3.2	Harvard Architecture	30
3.3	Three-stage Pipeline	38
3.4	Non-pipelined Processor Execution	39
3.5	Pipelined Processor Execution	39
3.6	Three Stages Asynchronous Pipeline Timing Diagram	40
4.1	Design Entity for 2-input AND Gate	52
4.2	Behavioral Description of Full Adder	54

4.3	Dataflow Description of Full Adder	55
4.4	Full Adder Schematic	57
4.5	Structural Description of Full Adder	57
4.6	VHDL Design Flow	61
5.1	MAX+PLUS II Development System	63
5.2	8-bit Asynchronous RISC Processor Development Design Flow	65
5.3	8-bit Asynchronous RISC Processor Block Diagram	67
5.4	Instruction Set Formats	70
5.5	Register Operation Datapath	75
5.6	Immediate Operation Datapath	76
5.7	Load Operation Datapath	77
5.8	Store Operation Datapath	77
5.9	Branch Operation Datapath	78
5.10	Jump Operation Datapath	79
5.11	Jump and Link Operation Datapath	79
5.12	Jump Register Operation Datapath	80
5.13	Jump and Link Register Operation Datapath	81
5.14	8-bit Asynchronous RISC Processor Datapath Organization	82
5.15	8-bit Asynchronous RISC Processor Controller	83
5.16	Text Editor (VHDL)	88
5.17	Functional Simulation	89
5.18	Graphical Editor	90
5.19	Global Project Logic Synthesis	91
5.20	Define Synthesis Style	92

5.21	FPGA Device Assignment	93
5.22	Floorplan Editor	94
5.23	Timing Simulation	95
5.24	Delay Matrix	95
5.25	MAX+PLUS II Programmer	96
6.1	XOR Gate	98
6.2	Timing Simulation for XOR Gate	99
6.3	Muller-C Element Schematic	100
6.4	Timing Simulation for Muller-C Element	100
6.5	TOGGLE Module	101
6.6	Timing Simulation for TOGGLE Module	102
6.7	SELECT Module	103
6.8	Timing Simulation for SELECT Module	104
6.9	CALL Module	105
6.10	Timing Simulation for CALL Module	106
6.11	ARBITER Module	107
6.12	8-bit Normally Closed Event Register	108
6.13	Timing Simulation for 8-bit Event Register	110
7.1	Instruction Fetch Unit	112
7.2	IFSELECT Circuit	115
7.3	6-input Muller-C Element	115
7.4	VHDL Code for 8-bit 2-to-1 Multiplexer	116
7.5	Asynchronous 2-to-1 Multiplexer	117
7.6	Asynchronous 4-to-1 Multiplexer	118

7.7	Structural VHDL Description of 8-bit Incrementer	118
7.8	Asynchronous Incrementer Unit	119
7.9	FETCHCTRL Event Control Circuit	120
7.10	VHDL Code for Instruction Memory	121
7.11	Asynchronous Instruction Memory	122
7.12	Instruction Decode Unit	124
7.13	VHDL Code for Instruction Decoder	125
7.14	Dataflow VHDL Code for Processor Controller	126
7.15	Asynchronous Processor Controller	127
7.16	Select_JAL Circuit	128
7.17	Asynchronous Register File	130
7.18	3-to-8 Asynchronous Decoder	131
7.19	Asynchronous Register File Request and Acknowledge Control Circuit	131
7.20	VHDL Code for Sign-extender Unit	132
7.21	Asynchronous Sign-extender Unit	133
7.22	Asynchronous Branch Unit	134
7.23	VHDL Code for 8-bit Comparator	135
7.24	VHDL Code for Conditional Branch Circuit	135
7.25	VHDL Code for 8-bit Adder	136
7.26	Decode Event Controller	138
7.27	Bypass Branch and Jump Event Control Circuit	139
7.28	Instruction Execution Unit	141
7.29	Asynchronous ALU	142

7.30	Structural VHDL Description of ALU	143
7.31	ALU Event Controller	146
7.32	LPM Data Memory VHDL Code	149
7.33	Asynchronous Data Memory	150
7.34	Completed 8-bit Asynchronous RISC Processor	151
8.1	Instruction Memory Contents (Instrmem.mif)	153
8.2	Data Memory Contents (datamem.mif)	154
8.2(b)	Fetch Unit Asynchronous Control Sequence	155
8.3	Instruction Fetch Unit (Register Operations)	156
8.4	Instruction Fetch Unit (Immediate, Load and Store Operations)	157
8.5	Instruction Fetch Unit (Branch and Jump Operations)	158
8.6	Instruction Decode Unit (Store Data Into Register File)	163
8.7	Instruction Decode Unit (Register Operations)	164
8.8	Instruction Decode Unit (Immediate, Load and Store Operations)	165
8.9	Instruction Decode Unit (Branch and Jump Operations)	166
8.10	Instruction Execution Unit (Register Operations)	168
8.11	Instruction Execution Unit (Immediate, Load and Store Operations)	169
8.12	Store Operation Verification	170

LIST OF ABBREVIATIONS

ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
C	Carry
CAD	Computer Aided Design
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DMEM	Data Memory
EDA	Electronic Design Automation
EDIF	Electronic Design Interchange Format
EXE	Execution Pipeline
FIFO	First In First Out Buffer
FPGA	Field Programmable Gate Arrays
HDL	Hardware Description Language
HIGH	Logic '1'
ID	Instruction Decode Pipeline
IEEE	Institute of Electrical and Electronics Engineers
IF	Instruction Fetch Pipeline
IMEM	Instruction Memory
imm	Immediate operand
IR	Instruction Register
LOW	Logic '0'

LPM	Library of Parameterized Modules
LSB	Less Significant Bit
MSB	Most Significant Bit
N	Negative
OPCODE	Operation Code
PC	Program Counter
PDAs	Personal Digital Assistants
RAM	Random Access Memory
rd	Destination Register Address
RISC	Reduced Instruction Set Computer
rs	First Register Source Address
rt	Second Register Source Address
RTL	Register Transfer Level
SNF	Timing Simulator Nelist File
SOC	System-On-Chip
SOF	SRAM Object File
SRAM	Static Random Access Memory
V	Overflow
VHDL	Very-High-Speed-Integrated-Circuit Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VITAL	VHDL Initiative Towards ASIC Libraries
VLSI	Very Large Scale Integration
WYSIWYG	What You See Is What You Get
Z	Zero

CHAPTER 1

INTRODUCTION

Portable and mobile computing has become a new trend in today's electronic industry. With the increasing demand in high-performance portable systems such as notebooks, laptops, Personal Digital Assistants (PDAs) and mobile phones, power consumption has suddenly become a hot issue and increasingly important factor in digital system design.

One of the most critical penetrations of these battery-powered portable equipments is the operating periods of the battery. Improving the energy density of the battery or reducing the power consumption of the system can be done in order to increase and extend the battery life. Since the progress rate in battery technology is slow, the alternative is to employ low power circuits design techniques to achieve longer battery life.

Most of the portable systems powered by batteries are performing tasks requiring increasing computational performance. For instance, PDAs demand a high-performance processor for functions such as handwriting and speech recognition. To complete this complex computational task, a higher clock speed is needed. However, increasing the clock rate will cause greater power consumption since clock distribution is responsible for large amounts of power usage. Power is dissipated when the transistors in the circuit change state on every clock cycle even if there is no useful work done in the circuit.

As VLSI feature size shrink to the 0.18-micron level and below, there has been a tendency to incorporate more functional units onto a single silicon die. Synchronous design methodology has reached its limitation where the clock distribution to all parts of the chip becomes increasingly difficult. Most of the design effort has been spent on designing the clock distribution scheme to overcome problem such as clock skew. Clock skew is a serious problem for the fastest CPU; witness DEC's Alpha, where more than a quarter of the silicon area is devoted to clock driver circuit (Pountain, D., 1993). Since the clock distribution is not scalable, the clock distribution network must be carefully redesigned each time.

This has lead to resurgence of interest in asynchronous logic design technique. Asynchronous circuits do not require global clocking; instead they rely on internally generated timing and use a communication protocol to allow data passing with other stages. They are also known as self-timed circuits because of the internally generated timing feature. Since there is no global clock in asynchronous circuits, clock skew problem can be avoided, silicon area can be saved and the most important is the power dissipation can be reduced.

Many academic research groups have been established to exploit the benefits of the asynchronous circuits by designing asynchronous processors. The most successful project is AMULET processor, a first commercial available asynchronous processor developed at the University of Manchester. This processor is mainly developed using full custom design, which needs a lot of time and effort.

Meanwhile, Field Programmable Gate Arrays (FPGA) has been used extensively in digital system design recently. The programmable chips serve for both quick prototyping and rapid time-to-market solutions in many applications. System design development using FPGA chips has promised a shorten time period compared to conventional Application Specific Integrated Circuit (ASIC) design cycle. Besides that, FPGA chips are getting cheaper with higher density gates. With the rapid development of FPGA technology beyond million gates, designing an asynchronous processor on FPGA becomes feasible.

Research Objective

The main objective of the work described in this thesis is to design and implement an 8-bit asynchronous Reduced Instruction Set Computer (RISC) pipelined processor using ALTERA FPGA chip. This processor employs two-phase transition signaling and bundled data approach which functions within the Micropipelines methodology. This work is carried out using Very High Speed Hardware Description Language (VHDL) for simulation and verification with the help of MAX+PLUS-II software.

Thesis Organization

The thesis is divided into 9 chapters. This chapter has presented an introduction to the growing concern of power consumption in digital system design, clock distribution problem in synchronous design and asynchronous design as a solution to these problems.

Chapter 2 to 4 gives a fundamental and essential background for the asynchronous processor design. Chapter 2 introduces the basic concepts of asynchronous logic. Micropipelines, a modular asynchronous design methodology is discussed in detail. Chapter 3 reviews the computer architecture tradeoff. Von Neumann, Harvard, CISC and RISC architectures are presented. In the last section of the chapter, recent built asynchronous processors are briefly described. Fundamental of VHDL language and the VHDL design flow are given in Chapter 4.

The scope of the project will be captured in four chapters, which is from chapter 5 to chapter 8. Chapter 5 discussed the methodology and the step taken using MAX+PLUS II development tools to accomplish this project. The design of basic building blocks for Micropipelines framework is discussed in Chapter 6. Chapter 7 detailed the 8-bit asynchronous RISC processor architecture. Each of the processor unit is presented. Chapter 8 discussed the result obtained from the simulation and significant findings during the development of the asynchronous RISC processor.

Finally, Chapter 9 summarized the conclusions drawn from the project presented in the thesis. Further work and possibilities are also introduced.

The Appendices contain the complete VHDL model of the 8-bit asynchronous RISC processor including Micropipelines event logic library and processor functional blocks.