MICRO-SEQUENCER BASED CONTROL UNIT DESIGN FOR A CENTRAL PROCESSING UNIT

TAN CHANG HAI

A project report submitted in partial fulfillment of the
requirement for the award of the degree of
Master of Engineering (Computer & Microelectronic Systems)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

APRIL 2007

DEDICATION

To my beloved wife, parents and family members

ACKNOLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers and academicians. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my thesis supervisor, Professor Dr. Mohamed Khalil Hani, for encouragement, guidance and friendships. I am also very thankful to my friends and family members for their great support, advices and motivation. Without their continued support and interest, this thesis would not have been as presented here.

ABSTRACT

Central Processing Unit (CPU) is a processing unit that controls the computer operations. The current in house CPU design was not complete therefore the purpose of this research was to enhance the current CPU design in such a way that it can handle hardware interrupt operation, stack operations and subroutine call. Register transfer logic (RTL) level design methodology namely top level RTL architecture, RTL control algorithm, data path unit design, RTL control sequence table, micro-sequencer control unit design, integration of control unit and data path unit, and the functional simulation for the design verification are included in this research.

ABSTRAK

Unit pusat pemprosesan (CPU) merupakan sebuah mesin yang berfungsi untuk menjana fungsi komputer. Buat masa kini, rekaan CPU masih belum sempurna. Malah tujuan utama penyelidikan ini adalah meningkatkan prestasi CPU dari segi operasi seperti gangguan, timbunan dan panggilan para fungsi. Rekaan metalogy logic pendaftaran berpindah (RTL) terdiri daripada rekaan RTL, kawalan algoritma RTL, rekaan unit data, rekaan unit micro-peringkat kawalan dan fungsi penyerupaan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALU	-	Arithmetic Logic Unit
CPU	-	Central Processing Unit
CU	-	Control Unit
FPGA	-	Field-Programmable Gate Array
ISA	-	Instruction Set Architecture
JMAP	-	Jump Map
PC	-	Program Counter
RAM	-	Random Access Memory
ROM	-	Read Only Memory
RTL	-	Register Transfer Logic
VHDL	-	Very High Speed Integrated Circuit Hardware Description Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This project is to describe a Central Processing Unit (CPU) can be implemented with a micro-programmed control unit. This chapter will cover, general CPU organization, objective of this project, scope of the project and the design methodology that are use in this project.

1.1 General CPU Organization

In general, a CPU controls the computer operations. It fetches instructions from memory, supplying the address and the control signals needed by the memory to access its data. The CPU decodes the instruction and controls the execution procedure. It performs some operations internally and supplies the address, data and control signals needed by memory and input/output (I/O) devices to execute the instruction.

Generally, a CPU has three sections as show in *Figure 1.1*. The register section includes a set of registers and bus or other communication mechanism. The registers in a processor's instructions set architecture are found in this section of the

CPU. The system address and data buses interact with this section of the CPU. The CPU includes registers to latch the address being accessed in memory and a temporary storage register.

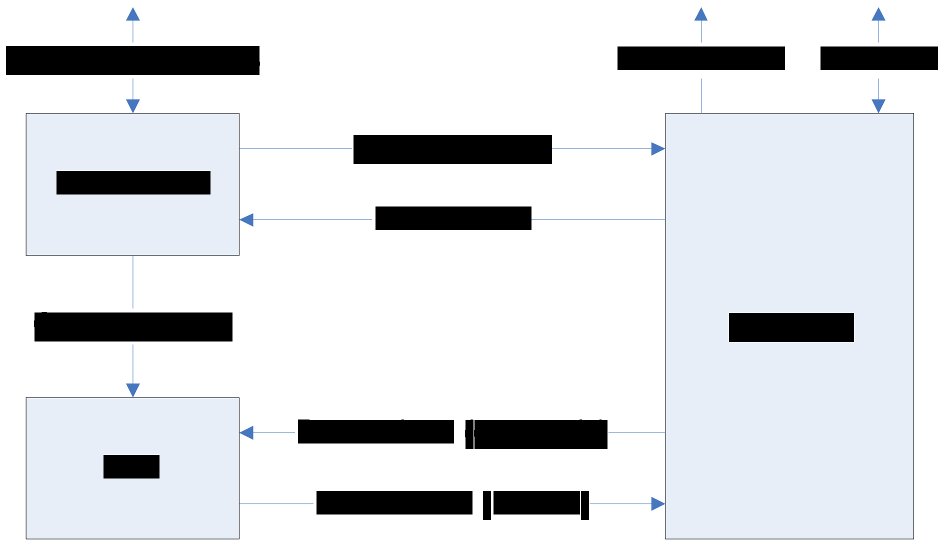

Figure 1.1: CPU internal organization

During the fetch portion of the instruction cycle, the processor first outputs the address of the instruction onto the address bus. The processor has a register called program counter to keeps the address of the next instruction to be fetched in this register. Before the CPU outputs the address onto the system address bus, it retrieves the address from the program counter register. At the end of the instruction fetch, the CPU reads the instruction code from the system data bus. It stores this value in an internal register called instruction register.

The Arithmetic Logic Unit (ALU) performs most of the arithmetic and logical operations such as adding, ORing values, XORing values and so on. It receives its operand from register section of the CPU and stores its results back to the register section.

The control unit controls the CPU, it generates the internal control signals that cause registers to load data, increment or clear their contents and output their contents as well as cause the ALU to perform the correct function. These signals are shown as control signals as shown in *Figure 1.1*. The control unit receives some data values from the register unit which it uses to generate the control signals. This data includes the instruction code and the values of some flag registers. A microprocessor typically performs a sequence of operations to fetch, decode and execute an instruction. By asserting these internal and external control signals in the proper sequence, the control unit causes the CPU and the rest of the computer to perform the operations needed to correctly process the instructions.

1.2 Objectives

The main objective of this project is to enhance and upgrade the in house CPU design. The work can be divided into sub-objectives

i. To enhance the in house developed CPU with additional subroutine instructions, hardware interrupts and stacks operations.
ii. To upgrade the hard-wired control unit to micro-sequencer based control unit.

1.3 Scope Of Work

The scope of work can be divided into the following:

i. This project is not to design a CPU but enhance the in house CPU design
ii. New instruction sets will be added to CPU.my to handle hardware interrupt, subroutine call and stack operations.
iii. Hardwired control unit will be upgrade to micro-sequencer base control unit.

1.4 Design Methodology

In this project, a CPU with the scope and objective that mention in previous section will be develop by applying hierarchical modular design concept. VHDLmg from UTM will be used as design entry tool to model the CPU at RTL level. Altera Quartus II design software tool will be used to synthesize the HDL for equivalent logic and targeted implementation platform at FPGA development board. The design will be verified with software functional waveform simulation. A set of testing program will be use to test all the instruction execution.