FPGA Based Implementation of 16 bit RISC Microcontroller

Nilam Patel
Dept. Of E&TC, D.N.Patel College Of Engineering, Shahada, Nandurbar India.

Prof. J.H.Patil
Dept. Of E&TC, D.N.Patel College Of Engineering, Shahada, Nandurbar India.

ABSTRACT

This project describes the design and implementation of some of the internal hardware components of a microcontroller. The subsystems designed are the fundamental hardware components necessary to create the 16-bit timer's input capture and output compare modes, the index register, and other systems. The subsystems designed were the clock controller, the 16-bit timer, the register controller, the register, and the comparator. The project engineers implemented their system using Xilinx to test their logic and VLSI to construct the gates. VHDL code was written in order to implement the project onto an FPGA board.

INTRODUCTION

When the controller design becomes more complex in CISC and the performance was also not up to expectations, people started looking at some other alternatives. It had been found that when a processor talks to the memory the speed gets killed. So the one improvement on CPI was to make the instruction set very simple. Simple in not the way it works but the way it looks. That's why there are very few instructions in any typical RISC architecture where processor asks data from memory probably not other than Load and Store. At the end the pipelining added a new dimension in the speed just with the help of some Addition registers, which increases throughput by reducing CPI. Hence the instruction can be executed effectively in one clock cycle[1].

A common misunderstanding of the phrase "Reduced Instruction Set Computer" is the mistaken idea that instructions are simply eliminated, resulting in a smaller set of instructions. In fact, over the years, RISC instruction sets have grown in size and today many of them have a larger set of instructions than many CISC CPUs. The term “Reduced” in that phrase was intended to describe the fact that the amount of work any single instruction accomplishes is reduced at most a single data memory cycle compared to the “complex instructions” of CISC CPUs that may require a number of data memory cycles inorder to execute a single instruction[2]. Most microprocessors in today's market are based on either RISC or CISC architectures. Research has shown that RISC architecture greatly boosts computer speed by using simplified machine instructions for frequently used functions. The following features typically found in RISC-based systems.

1) Pre-fetching: The process of fetching next instruction or instructions into an event queue before the current instruction is complete is called pre-fetching.
2) Pipelining: Pipelining allows issuing an instruction prior to the completion of the currently executing one.
3) Superscalar operation: Superscalar operation refers to a processor that can issue more than one instruction simultaneously.

ARCHITECTURE

The objective of the project is to design a 16-bit RISC processor which utilizes a minimum functional units. The architecture of proposed 16-bit Processor is shown in Fig.1. The processor incorporates 16-bit ALU capable of performing 11 arithmetic and logical operations, 16-bit program counter, 24-bit Instruction register, 16-bit general purpose registers, 3-bit flag register to indicate carry, zero and parity.

The processor has four states idle, fetch, decode and execute. The control unit provides necessary signal interaction to perform expected function in all the states.

The main objective of this project is to design a RISC microcontroller using VHDL and implement it in an FPGA. The microcontroller instruction set and features are based on Atmel AVR AT90S1200 RISC microcontroller.

Fig. 1. Architecture Overview

shows the top-level block diagram of the design, the bus structure has been simplified, but every block represents a module to be designed. At first glance, there are 11 modules in the top-level, with the 3 ports sharing the same module. These 11 modules are to be design separately using the top-down design approach. Some modules like the instruction register and status register are easy to design, but modules like ALU and the control unit require a lot of understanding. The overall dataflow and bus structure between all the modules must be understand before designing the modules individually.

There are basically two kinds of buses, direct bus and common bus. Direct bus connects two modules directly and is used specifically by the connected modules. There are many direct buses, such as the connection between program counter and program ROM, between program ROM and IR, between register file and ALU, etc. No control signals are required for direct buses.

The data bus is the only common bus in this design. The data bus provides connection between the general purpose register file, ALU, status register, SRAM and all the I/O features. The register file can only receive data from the data bus. All others modules can receive and send data to the data bus. Since there are so many possible data flows, control signals are required to control the correct flow direction. Only one source to the data bus is allowed at a time. If not, logic contentions will happen and the value of the data bus will be invalid. Tri-state bus is used to implement the common data bus. The impedance is so high that it can be seen as unconnected to the bus system. If the ALU is
the datasource, the data bus will be flooded with the result of the ALU and is available to all the connected modules. Control logic will generate an enable signal for the real destination to receive the data. The system can be divided into 3 units, the fetch unit, execute unit and I/O unit. Fetch unit is in charge of fetching the next instruction and the execute unit is in charge of executing the current instruction. I/O unit provides a connection with the outside world. The fetch unit and execute unit form the CPU of the microcontroller. The first module of the fetch unit is the program counter (PC). The PC contains the address of the next instruction to be executed. It points to the program ROM to locate the instruction. The instruction from the ROM is then latched into the instruction register (IR). The control unit takes the content of the IR and decodes it. It then asserts appropriate control signals to execute the instruction. All modules are connected with direct buses.

The execute unit in charge of executing most instructions. Normally, to execute an instruction, 2 operands are output from the register file to the ALU. The ALU then performs the operation and send the result to the data bus. Contents of the data bus (result) is then stored back to the register file. The ALU also evaluates the status register flags and sends them directly to the status register (SR). The whole execution process is done in a single cycle. The ALU performs many operations - include passing the content of a general register to the data bus. SR also has a direct bus connection to the control unit required for branch evaluation. The register file is addressed directly by some bits in IR.

**INSTRUCTION SET**

The operation of the CPU is determined by the instruction it executes, referred to as machine instructions or computer instructions. The collection of different instructions that the CPU can execute is referred to as the CPU’s instruction set. Since the instruction set defines the datapath and everything else in a processor, it is necessary to study it first.

Table 1 shows the instruction set summary of the designed microcontroller, while the instruction set summary of the original AT90S1200 is shown. There are 92 instructions grouped into 4 categories: arithmetic and logic instructions, branch instructions, data transfer instructions and the bit and bit-test instructions. The instruction set defines the datapath and everything else in a processor, it is necessary to study it first.

### Addressing Modes

There are 7 addressing modes in the microcontroller. Rd and Rr are devoted to the destination register and source register.

1. **Direct Single Register Addressing** The operand is in Rd.
2. **Direct Double Register Addressing** The operands are in Rd and Rr. Result is stored back to Rd.
3. **I/O Direct Addressing** First operand is one of the I/O registers. The address is contained in 6 bits of the instruction word. The second operand is either Rd or Rr. Used by IN and OUT instructions to read from or write to the I/O registers.
4. **Data Indirect Addressing** Operand address is the contents of the Z-register. Used when accessing the SRAM with LD and ST instructions.
5. **Data Indirect Addressing with Pre-Decrement** Z-pointer is decremented by 1 before the operation. Operand address is the-

---

**FINITE STATE MACHINE STATES**

Fig 2 shows the state diagram of the finite state machine (FSM). The 8 states are EXE (execute), SLEEP, BRANCH1, BRANCH2, SBICS (skip if bit in I/O clear/set), CBISBI (clear/set bit in I/O), ST and LD.

**SIMULATION RESULTS**

The 3 states are SLEEP, BRANCH1, BRANCH2. Fig 2 shows the state diagram of the finite state machine (FSM). The instruction set of the microcontroller is based on the AT90S1200 instruction set. In this way, the design can use the same assembler and simulator provided by Atmel since the microcontroller is an AT90S1200 compatible microcontroller. The first module of the fetch unit is the program counter (PC). The PC contains the address of the next instruction to be executed. It points to the program ROM to locate the instruction. The instruction from the ROM is then latched into the instruction register (IR). The control unit takes the content of the IR and decodes it. It then asserts the appropriate control signals to execute the instruction. All modules are connected with direct buses.

The execute unit in charge of executing most instructions. Normally, to execute an instruction, 2 operands are output from the register file to the ALU. The ALU then performs the operation and send the result to the data bus. Contents of the data bus (result) is then stored back to the register file. The ALU also evaluates the status register flags and sends them directly to the status register (SR). The whole execution process is done in a single cycle. The ALU performs many operations - include passing the content of a general register to the data bus. SR also has a direct bus connection to the control unit required for branch evaluation. The register file is addressed directly by some bits in IR.

Table 1 shows the instruction set summary of the designed microcontroller, while the instruction set summary of the original AT90S1200 is shown. There are 92 instructions grouped into 4 categories: arithmetic and logic instructions, branch instructions, data transfer instructions and the bit and bit-test instructions. The instruction set defines the datapath and everything else in a processor, it is necessary to study it first.

### Addressing Modes

There are 7 addressing modes in the microcontroller. Rd and Rr are devoted to the destination register and source register.

1. **Direct Single Register Addressing** The operand is in Rd.
2. **Direct Double Register Addressing** The operands are in Rd and Rr. Result is stored back to Rd.
3. **I/O Direct Addressing** First operand is one of the I/O registers. The address is contained in 6 bits of the instruction word. The second operand is either Rd or Rr. Used by IN and OUT instructions to read from or write to the I/O registers.
4. **Data Indirect Addressing** Operand address is the contents of the Z-register. Used when accessing the SRAM with LD and ST instructions.
5. **Data Indirect Addressing with Pre-Decrement** Z-pointer is decremented by 1 before the operation. Operand address is the-

---

**SIMULATION RESULTS**

The 3 states are SLEEP, BRANCH1, BRANCH2. Fig 2 shows the state diagram of the finite state machine (FSM). The instruction set of the microcontroller is based on the AT90S1200 instruction set. In this way, the design can use the same assembler and simulator provided by Atmel since the microcontroller is an AT90S1200 compatible microcontroller. The first module of the fetch unit is the program counter (PC). The PC contains the address of the next instruction to be executed. It points to the program ROM to locate the instruction. The instruction from the ROM is then latched into the instruction register (IR). The control unit takes the content of the IR and decodes it. It then asserts the appropriate control signals to execute the instruction. All modules are connected with direct buses.

The execute unit in charge of executing most instructions. Normally, to execute an instruction, 2 operands are output from the register file to the ALU. The ALU then performs the operation and send the result to the data bus. Contents of the data bus (result) is then stored back to the register file. The ALU also evaluates the status register flags and sends them directly to the status register (SR). The whole execution process is done in a single cycle. The ALU performs many operations - include passing the content of a general register to the data bus. SR also has a direct bus connection to the control unit required for branch evaluation. The register file is addressed directly by some bits in IR.

### Addressing Modes

There are 7 addressing modes in the microcontroller. Rd and Rr are devoted to the destination register and source register.

1. **Direct Single Register Addressing** The operand is in Rd.
2. **Direct Double Register Addressing** The operands are in Rd and Rr. Result is stored back to Rd.
3. **I/O Direct Addressing** First operand is one of the I/O registers. The address is contained in 6 bits of the instruction word. The second operand is either Rd or Rr. Used by IN and OUT instructions to read from or write to the I/O registers.
4. **Data Indirect Addressing** Operand address is the contents of the Z-register. Used when accessing the SRAM with LD and ST instructions.
5. **Data Indirect Addressing with Pre-Decrement** Z-pointer is decremented by 1 before the operation. Operand address is the-

---

**Fig. 2. State Diagram**

The state diagram shows the state flow but does not clearly show the inputs. The inputs to the FSM are the 46 output lines of the instruction decoder, timer IRQ, external IRQ, skip request and branch request. Branch request is generated by the branch evaluation unit when the condition of the conditional branch instruction is fulfilled.

**SIMULATION RESULTS**

The 3 states are SLEEP, BRANCH1, BRANCH2. Fig 2 shows the state diagram of the finite state machine (FSM). The instruction set of the microcontroller is based on the AT90S1200 instruction set. In this way, the design can use the same assembler and simulator provided by Atmel since the microcontroller is an AT90S1200 compatible microcontroller. The first module of the fetch unit is the program counter (PC). The PC contains the address of the next instruction to be executed. It points to the program ROM to locate the instruction. The instruction from the ROM is then latched into the instruction register (IR). The control unit takes the content of the IR and decodes it. It then asserts the appropriate control signals to execute the instruction. All modules are connected with direct buses.

The execute unit in charge of executing most instructions. Normally, to execute an instruction, 2 operands are output from the register file to the ALU. The ALU then performs the operation and send the result to the data bus. Contents of the data bus (result) is then stored back to the register file. The ALU also evaluates the status register flags and sends them directly to the status register (SR). The whole execution process is done in a single cycle. The ALU performs many operations - include passing the content of a general register to the data bus. SR also has a direct bus connection to the control unit required for branch evaluation. The register file is addressed directly by some bits in IR.

**Addressing Modes**

There are 7 addressing modes in the microcontroller. Rd and Rr are devoted to the destination register and source register.

1. **Direct Single Register Addressing** The operand is in Rd.
2. **Direct Double Register Addressing** The operands are in Rd and Rr. Result is stored back to Rd.
3. **I/O Direct Addressing** First operand is one of the I/O registers. The address is contained in 6 bits of the instruction word. The second operand is either Rd or Rr. Used by IN and OUT instructions to read from or write to the I/O registers.
4. **Data Indirect Addressing** Operand address is the contents of the Z-register. Used when accessing the SRAM with LD and ST instructions.
5. **Data Indirect Addressing with Pre-Decrement** Z-pointer is decremented by 1 before the operation. Operand address is the-

---

**Fig. 3 MVI**

The instruction set defines the datapath and everything else in a processor, it is necessary to study it first.
CONCLUSION AND FUTURE WORK

As a conclusion, this project has been completed successfully fulfilling all the objectives and scopes specified. The author has used his extra time to optimize the speed of the design until 12 MHz. The data RAM that is not specified in the scope of the project has also been included. Hardware stack is enlarged to 4-level instead of 3 and atotal of 24 I/O lines are available.

The design can be improved in number of ways. To achieve a more sophisticated design more features can be added to the current design. The number of instructionsthat the processor supports can be increased. Pipelining can be added to improve the performance of the proposed design.

REFERENCE