VLSI DESIGN OF REDUCED INSTRUCTION SET COMPUTER PROCESSOR CORE USING VHDL

M.YUGANDHAR & N.SURESH BABU

Department of Electronics and Communication Engineering, Chirala Engineering College, Chirala, India

ABSTRACT

This paper describes a 32-bit RISC microprocessor core that has been designed for embedded and portable applications. This RISC embodies 15 basic instructions involving Arithmetic, Logical, Data Transfer and control instructions. To implement these instructions the design incorporates various design blocks like Control Unit (CU), Arithmetic and Logic Unit (ALU), Accumulator, Program Counter (PC), Instruction Register (IR), Memory and additional logic. The RISC is designed to enhance processor performance by keeping the following goals in mind. The RISC uses simple constructs and has small instruction set. It is basically designed in order to achieve faster executions. Control logic design is very much simplified. RISC can execute each instruction within one clock cycle.

KEYWORDS: Reduced Instruction Set Computer, ALU, Accumulator, Memory

INTRODUCTION

This RISC processor is designed using pipelined architecture; through this we can improve the speed of the operation. In this we are using 5-stage pipelining. The 5 stages are Fetch, Decode, Execute, Memory and Write Back. The reason for wide use is that they are small; therefore, they do not take up much die area and are cost effective to fabricate.

This processor will follow the RISC architecture because it supports a predefined set of instructions. In this all the instructions have same length. RISC processors, first developed in the eighties. Embedded processors demand instruction set suitable for the specific applications, various and fast interrupt handling, consumption and so on. For this properties RISC is more efficient in comparison with micro programmed than CISC.

INSTRUCTION SET

The instruction set comprises 8 basic instruction formats.

Two of these make use of the on-chip arithmetic logic unit, barrel shifter and multiplier to perform high-speed operations. Three types among these instructions control the transfers of data between main memory and the register block. One is optimized for flexibility of addressing, another for rapid context switching and the third for swapping data.

Three instruction formats control the flow and privilege level of execution. The most significant feature of instruction set is that all of instruction may be executed conditionally according to the condition flags. A condition field within all instructions is compared with the condition flag in program status register. And the result of the comparison determines whether the instruction can be executed or not. This reduces the need for forward branches and allows very dense in-line code (without branches) to be written. In addition, shift operation is performed in serial with ALU in all data path cycles. This enables data processing instructions to execute various arithmetic operations using shift operation in one clock cycle. Block data transfer instructions move all 16 available register to/from memory. These instructions require maximum 18 clock cycles to complete and are used for fast context switching. The data swap instruction is used to swap a byte or a word quantity between a register and external memory.
This class of instruction is particularly useful for implementing software semaphores. Also, there are multiply, multiply and add instruction.

The most significant feature of instruction set is that all of instruction may be executed conditionally according to the condition flags. A condition field within all instructions is compared with the condition flag in program status register. And the result of the comparison determines whether the instruction can be executed or not. This reduces the need for forward branches and allows very dense in-line code to be written. In addition, shift operation is performed in serial with ALU in all data path cycles. This enables data processing instructions to execute various arithmetic operations using shift operation in one clock cycle. Block data transfer instructions move all 16 available register to/from memory. The multiplication instruction uses Booth’s algorithm to perform integer multiplication. They may be useful for digital signal processing.

ARCHITECTURE

To implement these instructions the design incorporates various design blocks like Control Unit (CU), Arithmetic and Logic Unit (ALU), Accumulator, Program Counter (PC), Instruction Register (IR), Memory and additional logic. It has 32-bit data bus and address bus. The CPU can directly manipulate two data types: bytes and words. A word is aligned on 4-byte boundaries, where the two least significant address bits me both zero. All instructions occupy exactly single word, and intend data operations are performed only on 32-bit base. Byte data types may be loaded from and stored to memory and the data is extended to 32-bit length before it will be stored to register. CPU operating modes are supported and each mode can be changed by software, and external interrupts or exception handling routine. One of these modes is for reducing the overhead of context switching in interrupt handling. In this architecture five stages pipelining composed of fetch, decode, execute, Write Back and Memory is employed and each stage is operated on a non-overlapping clocking.

There are internal 32-bit data paths, each of which is associated with one of the register block ports. Memory interface has been designed to allow the performance potential to be realized without incurring high costs in the memory system. Speed critical signals are pipelined to allow system control functions to be implemented in standard logic, and these control signals facilitate the fast access mode supported by industry standard DRAM. When the memory system cannot respond in the normal cycle time, one of the clock phases is stretched. Fig 2 is a block diagram of this RISC processor.
The architecture consists of five stage pipelining. Instruction fetch, Instruction decode, Execute, Memory and Write Back. Also, we added a Data Forward and Hazard detection unit to maintain proper data flow through the pipeline stages.

**A. Instruction Fetch:** This stage consists of Program counter, Instruction Memory, and the Branch Decode unit.

**Program Counter:** The program counter (PC) contains the address of the instruction that will be fetched from the Instruction memory during the next clock cycle. Normally the PC is incremented by one during each clock cycle unless a branch instruction is executed. When a branch instruction is encountered, the PC is incremented/decremented by the amount indicated by the branch offset. The PC Write input of the PC serves as an enable signal. When PC Write signal is high, the contents of the PC are incremented during the next clock cycle, and when it is low, the contents of the PC remain unchanged.

**Instruction Memory:** The Instruction Memory contains the instructions that are executed by the processor. The input to this unit is a 32-bit address from the program counter and the output is a 32-bit instruction word.

**Branch Decide Unit:** The Branch Decide Unit is responsible for determining whether a branch is to take place or not based on the 2-bit Branch signal from the control unit and the Zero flag from the Arithmetic and Logic Unit (ALU). The output of this unit is a 1-bit value which is high when a branch is to take place, and otherwise it is low. This output controls a multiplexer which in turn controls whether the PC gets incremented by one or by the amount indicated by the branch offset.

**B. Instruction Decode:** This stage consists of the Control Unit, Register File.

**Control Unit:** The control unit generates all the control signals needed to control the coordination among the entire component of the processor. The input to this unit is the 4-bit opcode field of the instruction word. This unit generates signals that control all the read and write operation of the register file, the Data memory. It is also responsible for generating signals that decide when to use the multiplier and when to use the ALU, and it also generates appropriate branch flags that are used by the Branch Decide unit.

**C. Execute:** This stage consists of the Branch Adder, Arithmetic Logic Unit (ALU), and the Control Unit.

Let us consider an instance when some information is stored in the memory. Now when the system is switched on, CPU is initialized. In order to fetch an instruction, as a result the program goes to the location in the memory that is pointed out by the program counter. After some instance, the instruction from the memory is put on the data bus. This cycle is called the instruction fetch cycle. The instruction is now available at the data bus. At next instance; the instruction is loaded into the instruction register. This is called the instruction load.

In this cycle the 4 msb’s of the instruction are separated and put in the opcode register and are loaded to control unit as well as ALU. The rest of the bits are sent out as Irout. The outputs of the instruction register and the program counter are connected to a mux. During the negative edge of the fetch signal, the output of the instruction register is selected, while the output from the program counter is selected during the positive edge of fetch cycle.

Now when the fetch signal goes low the mux selects the output from the instruction register and it points to the location of the operand. Now the operand present in the location is placed on the data bus. After an instruction is fetched the program counter is incremented. It points to the next location. Now the operand is available at the ALU. The operand is taken in by the ALU and operates on it. Now the result is available at acc1 at positive edge of execlk. During the negative edge of execlk, the result at the Acc1 register is placed on the data bus, which is sent and loaded into
the accumulator for any further operations. If the data has to be stored into the memory, then during this clock cycle, Rd and Wr has to be 0 & 1 respectively. As a result the accumulator is connected to the memory and the value in the accumulator is sent back to a location in the memory through a module named Buffer. A characteristic of RISC processor is their ability to execute one instruction per clock cycle.

RESULTS & CONCLUSIONS

The simulation and implementation results are shown below Fig 4 & 5.

Stages in the design and implementation of 32 bit RISC processor are:

1. An architecture for implementing simple 32-bit RISC processor is to be taken.
2. All individual modules in the above 32-bit RISC processor architecture are to be designed and simulated using Hardware Description Language (HDL).
3. Synthesize those individual modules to extract Gate level net list.
4. Finally it should be simulated and synthesized in Modelsim and Xilinx.

This code was compiled and then simulation was carried out. Analysis and Synthesis reports of various modules were observed. The results obtained were found to agree with the expected results. Finally the individual modules were integrated.
This top order module was compiled and then simulated. RTL views of the individual modules were also observed after simulation and synthesis as shown in Fig 5 & 6.

The further scope of this project is that the design can be verified for timing functionality and can be implemented into FPGA chips.

Additional features like interrupt processing, instructions employing conditional branches, more registers can be included in the RISC processor design aspects. The CMOS logic for the above design can be implemented, synthesized and simulated using VLSI software tools.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.
REFERENCES


3. Alex van Someren, Carol Atack, "The ARM RISC Chip, A Programmers Guide", 1993


AUTHORS PROFILE

Macharla Yugandhar is pursuing M.Tech in VLSI &ES at Chirala Engineering College, Chirala.

Prof. N. Suresh Babu is Vice-Principal & HOD of ECE Dept in CEC, Chirala. He got his M.Tech in Microwave Engineering from Birla Institute of Technology, Ranchi. He has 13 years of teaching Experience and 2 Years of Industrial Experience in various organisations.