A Verilog-Based Design of 16–Bit RISC Processor

Miss Divya, Miss Deepty

***Abstract*— RISC is a design philosophy to reduce the complexity of instruction set that in turn reduces the amount of space, cycle time, cost and other parameters taken into account during the implementation of the design. Microcontrollers and microprocessors are finding their way into almost every field in today's world, incorporating an element of "smartness" into conventional devices. Energy efficient, space efficient and optimized microcontrollers are the need of the day. Our paper proposes a new Instruction Set that is a subset of the MIPS architecture. It derives the advantages of MIPS like simplicity and speed. Besides, since it is a smartly optimized subset of MIPS, it is a smaller version consisting of the most commonly required instructions.**

***Index Terms*— RISC, control unit, processor, ISA, MIPS, Processor design, RISC.**

I. INTRODUCTION

Now days, Computers are mainstream in quotidian activities. RISC Processor is a CPU design strategy that uses simplified instructions for higher performance with faster execution of instruction. It also reduces the delay in execution. It uses general instructions rather than specialized instructions. They are less costly to design, test and manufacture. This has helped in implementation of RISC in technological field. Its range of application includes signal processing, convolution application, supercomputers such as K computers and wider base for smart phones.

MIPS is a reduced instructions set computer (RISC) architecture. It is one of the first RISC Instruction set architectures. MIPS is an acronym for "Microprocessor without interlocked pipeline stages". It was developed by a team led by John Hennessey at Stanford University. MIPS implementations are primarily used in embedded systems such as Windows CE devices, routers, residential gateways, and video game consoles such as the Sony PlayStation 2 and PlayStation Portable.

II. MIPS 16 INSTRUCTION SET DESCRIPTION

A. Motivation: Small scale applications do not require that much of computing power. This paper proposes a reduced version of MIPS instruction set for such small scale applications. This ISA will be called MIPS 16. The main aim of this ISA is to reduce the transistor count of a MIPS processing unit by scaling down the bus and register width and providing less but enough number of instructions for small scale applications. The implementation of such an instruction set would take up less real estate on the chip (or FPGA) and will allow more peripherals to be fabricated on a single chip making it ideal for a System-On-Chip (SOC) implementation of an application. It will also be beneficial in embedded system design where a custom processor core implementation is required with tight instruction requirements so that it takes less space on a FPGA.

B. Instruction Set Specification [3] MIPS instructions have fixed width. The original MIPS 32 ISA has 32 bits wide instructions. Each instruction in MIPS16 is 16 bits wide. Further, MIPS16 has 8 internal registers as opposed to the 32 registers of MIPS32. As the name suggests, data bus is 16 bits wide and address bus is preferably 16 bits wide too. I/O support is memory mapped. Memory is accessed by LOAD and STORE instructions. The instructions follow an <operand register, register, register> format. The Instructions can be divided into 4 groups:

a. **Arithmetic**: Basic computational instructions add and subtract.

b. **Logical**: Operations like AND, OR, EXOR

c. **Data Transfer**: Load and Store operations

d. **Branch and control**: Jump, Call, Return, etc.

The ISA supports direct and immediate addressing modes.

C. Instruction Word Format

A MIPS16 instruction is 16 bits wide. Since MIPS uses a Register-Register type of instruction a general instruction specifies two source registers and a destination registers. The format of such an instruction will be ADD R_{s1}, R_{s2}, R_d R_{s1} = First source operand register R_{s2} = Second Source operand register R_d = Destination register The instruction word has a **5 bit op-code** specifying the operation to be performed. Number of operands may be variable e.g. ADD requires three operands while NOT requires only two. Format of a three operand instruction word is shown in Table

I TABLE I
THREE OPERAND INSTRUCTION

Op-Code	Rs 1	Rs 2	Rd	Reserved
5-bit	3-bit	3-bit	3-bit	2-bit

In case of ALU instructions, the 2 reserved bits act as function bits where they are used to distinguish between versions of a common instruction. For instance, the instructions ADD and ADC have the same opcode but different function bits. This results in a simpler control logic as the reserved bits are decoded directly by the ALU control logic. In case of lesser operands appropriate operand is given a constant value. E.g. NOT instruction requires only one source and one destination operand. Therefore, Rs2 field will be made "000" as shown in Table II. Likewise a POP instruction will require only destination and hence both the source operands will be constant and only destination needs to be provided as shown in Table III.

Miss Divya, Amity School of Engineering and Technology, Amity University Noida,7417946506.

Miss Deepty, Amity School of Engineering and Technology, Amity University Noida, 8533903260.

4 ***NOT Rs, Rd***

TABLE II
TWO OPERAND INSTRUCTION

Op-Code	Rs 1	Rs 2	Rd	Reserved
5-bit	3-bit	000	3-bit	2-bit

POP Rd

TABLE III
ONE OPERAND INSTRUCTION

Op-Code	Rs 1	Rs 2	Rd	Reserved
5-bit	3 bit	000	3-bit	2-bit

The unused fields in an instruction are also used to provide immediate input. The size of the immediate field depends on the number of operands instruction uses. *ADDI Rs, Rd, #10* Rs = Source Register Rd = Destination register #10 = Immediate value (0-31 in decimal)

TABLE IV
IMMEDIATE INSTRUCTION

Op-Code	Rs 1	Rs 2	Immidiate
5-bit	7:5 bits of imm value	3-bit	4:0 bits of imm value

Jump instructions have two modes viz..PC relative and absolute modes. In PC relative mode. The lower 5 bits of the instruction are used to specify a 5 bit signed value as shown in Table IV. This value is added/subtracted to the PC to get the jump address. The PC relative mode is used for conditional jump instructions. The absolute mode is used for unconditional jumps and jump-&-link instructions. In these instructions the all bits other than the opcode are used to specify a 11 bit signed PC offset value. The instruction set is so designed so as to simplify the instruction decoding logic and the control logic.

TABLE VI
JUMP (ABSOLUTE MODE) INSTRUCTION

Op-Code	**Immediate**
5-bit	11 it signed PC Relative offset

D. Comparison between MIPS-16 and MIPS-32 [7]

MIPS-16 can be considered to be a derivative of MIPS-32 instruction set. But the philosophies behind their design are different. MIPS-16 provides more flexibility in terms of optimizing the design by keeping only the required instructions. MIPS-16 is designed for small scale applications while MIPS-32 is a high performance 32-bit architecture which can handle large data and perform fast calculations by employing multiple pipelines and multiple registers at the cost of larger chip area and complicated logic design. Some key differences have been highlighted in TABLE VII

TABLE VII
COMPARISON OF MIPS-16 AND MIPS-32 ISA

Serial No.	**MIPS-16 Instruction set**	**MIPS-32 Instruction set**
1.	Instruction word length is 16-bit	Instruction word length is 32-bit
2.	Supports only 8 general purpose registers	Supports only 8 general purpose registers
3.	Register is 16 bits wide	Register is 16 bits wide
4.	Program counter should be incremented by 2 after every instruction (for non-branching instructions)	Program counter should be incremented by 4 after every instruction (for non-branching instructions)
5.	ALU is simpler. It does not support operations with bulky logic like Multiplication and Division	ALU is complicated. It supports complicated operations like Multiplication and Division.
6.	Floating point instructions are not included in MIPS-16	Floating Point instructions are included and are called SIMD instructions
7.	Pipelining is not essential and depends on the application.	Pipelining is a key feature of a MIPS-32 based processor.
8.	Transistor count and chip area is less	Transistor count and chip area is more

E. List of Instructions

As the op-code has a 5 bit length there are 32 possible distinct instructions. If the reserved bits at the end of the instruction are utilized for grouping 2 or more similar instructions more op-codes can be incorporated in the instruction set e.g. ADD and ADC can be grouped together as they perform similar function with the difference being inclusion of carry into the

sum. A complete list of 37 instructions has been provided in Table VIII with a short description of each instruction.

F. Implementation Strategies

The implementation strategies that can be employed will depend on the application. Some of the design considerations are listed below:

a. Single-Cycle or Multi-cycle implementation: [3], [4], [5] Implementation can use a single cycle or a multi cycle control system for its datapath. A single cycle control system performs all the elementary data path operations in a single cycle. This generally requires dedicated hardware for every phase of instruction fetching, decoding and execution. It is faster at the cost of larger chip area. A multi cycle implementation divides the execution of an instruction into well-defined time states. The execution happens in a timely order and might require different number of time states for different instruction. The main advantage is the hardware can be shared for similar elementary functions in different time states. Multi cycle should be preferred for applications with smaller chip area requirements.

b. Pipelining Requirement:[6] Pipelining provides performance enhancement by concurrent execution of more than one pair able instructions. The design involves use of multiple data paths and a logic to check for pairability and hazard removal that occurs due to concurrent execution. This significantly complicates the design and takes a larger chip area. But the performance improvement would be tremendous.

III. CONCLUSION

MIPS-16 is thus a low-cost, compact and hence in effect a low power RISC instruction set architecture as compared to the MIPS-32 architecture. Its compact size and flexibility makes it ideal for an optimized implementation of an embedded system. It provides all the basic instruction and functionality for a small scale embedded system not involving heavy arithmetic calculations. MIPS-16 can be implemented on FPGA by an appropriate strategy as per the application's requirement. Single cycle design should be used for better performance while multi cycle design should be preferred for compactness. Pipelining can further improve the system performance.

REFERENCES

[1] http://en.wikipedia.org/wiki/MIPS_architecture.
[2] http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm
[3] David A. Patterson and John L. Hennessy, Computer Organization and Design, 3rd ed.
[4] Lecture notes by Howard Huang, University of Illinois at Urbana-Champaign. [Online]. Available: http://www.howardhuang.us/teaching/cs232/11-Single-cycle-MIPS-processor.pdf
[5] Lecture notes by Howard Huang, University of Illinois at Urbana-Champaign. [Online]. Available: http://www.howardhuang.us/teaching/cs232/12-Multicycle-datapath.pdf
[6] Lecture notes by Howard Huang, University of Illinois at Urbana-Champaign. [Online]. Available: http://www.howardhuang.us/teaching/cs232/15-Pipelining.pdf
[7] MIPS Official Website. Available: http://www.mips.com/products/architectures/mips32/