Comparative Study of RISC Architectures

Chhayadevi M Bhamare , Meghanand A. Bhamare

EEE & I Department BITS-Pilani Goa Campus Goa,India
Email: cmb@bits-goa.ac.in, mab@bits-goa.ac.in

***Abstract* - RISC or Reduced Instruction Set Computer is a design architecture that focuses on simplification of the instruction set to achieve the goal of a simplified architecture implementation. It is characterized by certain distinguishing features such as single cycle execution time, simplified instruction set and load/store architecture. However, new generation RISC processors are modifying some of these features, while adding more attributes based on their areas of application. The current paper reviews some of the RISC architectures seen over the years and analyzes their salient features.**

***Index terms* – RISC, Superscalar , MIPS**

I INTRODUCTION

RISC(Reduced Instruction Set Computer) was first conceptualized at the university of California, Berkely with the aim of developing a single-chip computer with a simplified instruction set, as opposed to the CISC (Complex Instruction Set Computer) architecture that was prevalent at the time. The motivation behind the development of RISC was to avoid consequences that accompanied the complex architecture of CISC, namely increased design time, increased design errors and inconsistent implementations. Power consideration and the limited number of transistors that could be accommodated on a single chip were further limitations to the development of complex architecture. The hypothesis, therefore, behind RISC was to reduce the complexity of the instruction set, thereby simplifying the architecture and providing for better utilization of scarce chip resources.

There is no precise definition of what constitutes a RISc design. However, certain characteristics are always associated with RISC processors. While initial RISC processors had fewer instructions compared to their CISC counterparts, new generation RISC processors have hundreds of instructions, some of them as complex as CISC processors. These can consequently be viewed as sorts of ' Hybrid' RISC design. However, there are certain principles that most RISC design follows. These include :

1. Simple operations: the objective is to design simple instructions which can b executed in a single machine cycle, thus simplifying processor design.
2. Register-to-Register operations: only load and store operations access memory. Operands are always present in registers.
3. Simple addressing modes: this follows fast addresss computation for operands. In RISC, most instructions use register based addressing.
4. Large register set: since register-to-register operations are used, a large number of registers have to be provoded. These also provide the advantage of minimizing overheds associated with procedure calls and returns
5. Fixed-length, simple instruction format: variable length instructions cause implementation and execution inefficiencies. Fixed length instructions allow efficient decoding and scheduling.

This paper is organized as follows: section II presents early RISC architecturesthat puioneered the RISC revolution. Section III discusses superscalar architectures which is in advance form of RISC architectures. Section IV lists special RISC architectures which are specific to a particular application.

II. EARLY RISC ARCHITECTURES

A. RISC I

RISC 1 [1] is one of the erliest RISxc processors. The most distinguished feature of the RISC architecture is Register Windowing Feature. This mainly facilitates decrease in the call and the branch overhead as the registers need not be backed up on a stack, only the register window needs to be moved between the calls. This feature also increases the sped of recursive algorithms. Delayed branch feature prevents the pipeline hazards by inserting NOPs after conditional pipeline.. this architecture with only 32 instructions gave superior performance as compared to the existing CISC architecture like VAX. The RISC 1 relied highly on the Register Windowing Feature, but this feature was later discarded in advanced RISC architecture.

B. MIPS

Contemporory to the RISC is design of MIPS[2] architecture (Microprocessor Without Interlock Pipeline Stages). It was developed on similar lines of RISC I but had many additional features like Software Interlocking. Here, the pipeline stall overhead is handled by the software. Various tasks can be reorganized at compile time itself. It has the obvious disadvantage of increased code size, but the main focus of faster execution of instructions is achieved. However, the increased code size demands frequent memory access and features like off-chip MMU, which introduces memory access overhead. MIPS is a word addressed Harvard architecture with the core incorporating carry-look-ahead ALU, multiply, divide and barrel shifter with byte insert/extract capabilities. Code compression involves compressing a compressing a load/store instruction with less than 32 bits into an ALU instruction. This feature is especially noteworthy and is followed in many advanced RISC and DSP processors. However, it has no condition codes and no floating point implementation. Although, the RISC 1 and MIPS have a basic common philosophy, the main difference between the two lies in the complexity of the compiler and the user instruction set in case of the MIPS architecture.

III. SUPERSCALAR ARCHITECTURE

A. Ultra SPARC

Ultra SPARC[3] (Scalable Processor Architecture) is a superscalar RISC architecture that implements the 64-bit SPARC V9 architecture. It is a four way superscalar design with four instructions dispatched together, consisting of combination of two integers, two floating point, one load/store and one branch. The data path width is 64-bit/128-bit. Condition code for both 32- bit/64-bit data type can be set. 64-bit integer multiply and divide capabilities are present. It possess a Visual Instruction Set(VIS) which makes it suitable for graphics applications. It has 128-bit pipelined and split transaction bus. All floating-point and graphics operations, except divide and square root, are pipelined, and have one to three cycle latencies. It has nine stage pipeline. Branching capabilities include software-settable branch prediction and branch on register values. The processor has a Prefetch/Dispatch unit which fetches four instructions per clock, expands them to 76-bits and stores them into a twelve level deep instruction buffer. The Integer Execution unit is responsible for integer arithmetic and pipeline control. Floating point/graphic unit is present, which consists of floating point adder, multuiplier and divider. The floating point multiplier is fully pipelined and performs single and doubkle precision FP multiplications in thjree cycles. Eight –window register file with seven read ports and three write ports is present. Two ALU's and dedicated hardware for address calculation helps to increase processing speed. It also utilizes Register window Mabnagment. Exceptions are handled through a nested five level trap, with each level storing the necessary processor sattes for every trap. A special processor state called RED-state (Reset, Error and debug state) is entered upon resets or faults that trap into the fifth level. The Load/Store unit executes instructions, transferring data between memory and register files. It consists of 16KB direct mapped Data cache, nine level load Buffer organized as a circular queue and an eight-level Store Buffer. The Memory Management Unit consists of two independent MMU's: one for instruction and other for the data. Each MMU contains a fully associative, 64- entry translation Look-aside Buffer(TLB) with one virtual to physical address translation pr cycle. External cache ranging from 512 to 4MB is supported and is used to service misses from the instruction and Data cache. It is physically addressed and physically tagged. There are two data buffers between the processor and external memory to electrically isolate them. The arbitration is distributed, round robin with hysteresis to reduce the latency for the last master. A special parking mode for the uni-processor is also available. Interrupts are sent or received as write packets. This eliminates centralized interrupt controllers and avoids the normal polling overhead that accompanies interrupt handling.

B. RNIW

RNIW [4] (RISC plus Normal Instruction Word) is a novel architecture proposed for DSP. RNIW is a superscalar architecture with 32 general purpose registers, sevnty three 32-bit instructions and 32-bit data width. Modular and saturation arithmetic is supported. It has a 32-bit linear address space, eight instruction formats, three addressing modes and almost zero cycled branch instructions. The RNIW allowsSIMD and MIMD possible in the normal instruction word byLoad FIFO's and concurrent instructions of 32 bit with 4 sub fields with one opcode per field indicating one registers group operation. However load and store instructions cannot b executed parallel. It consists of three logical units viz. Integer unit 1(IU 1), Integer unit 2(IU 2) and Load-store unit. IU 1 consists of a five port register file with 16 registers, an address increment logic unit and a CALU unit(central ALU consisting of 1 multiplier, 1 ALU, 1 barrel shifter). The IU1 does the overall operation of the processor by executing control instructions and computing memory addresses and dispatches instructions to the load-store unit. The branch unit prefetches packet of four instructions and dispatches the instructions to the functionbal units in parallel or sequential manner depending on whether the instructions are dependent or not. Thus a maximum rate of four instructions per cycle can be achieved. The IU2 consists of four register groups, four CALU units and a crossbar interconnection network. The load-store unit consists of eight load FIFO buffers. Throuput and speed is increased due to pipelining of the register groups as loadstore overhead is decreased. At the same time, less hardware is required. The load FIFO's increase data throughput which enables multimedia applications possible. MPEG decompression is one multimedia application featured by this application. The crossbar swithching connects one CALU to another CALU during dynamic configuration instead of the load FIFOs. So, data actually flows from one CALU to another. Subword mode of concurrency is also supported. If any overflows occur during this operation, it is compensated by Wrap around arithmetic and saturation arithmetic. Special instructions for dynamic instruction format and concurrent logical and concurrent logical and arithmetic instructions format is designed. In contrast to the superscalar architecture, RNIW has separate control and calculation instructions.

C. *IBM second generation RISC*

This is numerically intensive design[5] with separate instruction data caches to aoid the critically important cache conflicts. Instruction cache is two way set associative 8 kb cache with a line size of 64 bytes and translation- lookaside buffers. Four instructions are fetched concurrently of which two are dispatched to the Instruction cache unit(branch and condition register instructions), and two to the fixed and floating point unit. Data cache unit is four-way set associative 64kB size, and divided into four units of 16K each. The line size is 128 bytes. In case of cache miss, cache reload buffers are present so that CPU doesnot have to wait for the entire cache line to be brought from the main memory. ' Dirty' cache lines are stored in store back buffers rather than the main memory, so that new lines can be simultaneously brought from the main memory to cache, instead of having to wait for a write-back sequence. The Fixed point unit (FXU) performs all fixed point instructions and floating pointload/store instructions. It contains general purpose registers and the ALU. Register tagging is implemented, which allows data cache access operations and independent register-register instructions to occur simultaneously. The Floating Point unit is tightly coupled to the CPU and has the same priority as the fixed point unit. It has a full 64-bit double precision datapath. It provides the key feature of a single cycle execution of multiply/add operation. It uses register remapping to allow floating point loads to be executed independently from floating point arithmetic operation. Similarly, a pending store queue allows the FXU to perform floating point stores independently from floating point arithmetic. A storage Control Unit is present that controls communication between CPU, main memory and I/O. it also controls the interface between d- Cache andsystem memory, and oversees DMA operations. The key objective of achieving CPI close to one is possible by making the peak instruction execution rate greater than one instruction per cycle. This requires independent functional units executing instructions concurrently. This design therefore implements separate branch, fixed point and floating point units and can execute four instructions per cycle. Pipeline delay is a significant overhead in RISC architectures. This is avoided by performing a significantly far look-ahead branching, while also utilizing the features of four word interface to the I- cache and a large number of instructions buffers. The floating point unit can execute parallel with the fixed point unit, thereby improving floating point performance. MAC instructions can be executed with the same dely as multiply instruction. Another feature different from other RISC architectures is precise interrupts. Here when an instruction causes an interrupt, the pipeline is stopped before the next instruction can affect the machine state. Therefore, return from an interrupt can be resumed at the interrupting instruction. Instructions for string operations and load/store update are present, which are typically not seen in other RISC architectures.

IV. OVERVUIEW OF SPECIAL RISC ARCHITECTURES

A. *YD- RISC*

Dual core processors with a RISC and DSP cores on the same die are very popular in the embedded arena. However the main drawback in such architectures is the performance cost. YD-RISC [6] attempts to combine both the cores as parallel execution units on the same die wherein the functional units and the pipeline both operate in parallel. YD-RISC is a 32-bit RISC with DSP functionality designed through logic synthesis. It has a high clock rate with five stage pipelined architecture. The instructions are divided in to load/store, ALU and DSP pipelines. The DSP can execute one 32 –bit multiplication in one cycle. Also, the area occupied by the 32-bit multiplier is half that of a 16-bit multiplier. Operand examination scheme reduces the overhead on the cycles of a 32-bit multiplier. To overcome the drawback due to the absence of data and memory cache, a 4-K SRAM and a circular buffer with a head and tail pointer are implemented. The loop control of this buffer reduces the bus overhead of the CPU increasing the performance. The architecture has variable instruction size of 16, 32 and 48 –bit length which improves the code density. It has a fixed point DSP with provision of 32 bit multiplication and circular addressing mode. But, it does not have separate data and memory cache or separate data and memory bus. Code density is achieved by variable length instructions. The architecture has support for low power modes by using the clock disable circuitry and disable signals to the various functional units. Only timer, interrupt, refresh and reset logic is provided with a clock in this mode. A comparative performance of YD RISC with different DSP procesors shown in TABLE I clearly reveal the superior performance of YDRISC as compared to the other DSP cores. In addition, this is possible at a reduced cost and power consumption by the use of low power modes of operation. The architecture has twice the performance in the DSP and in integer applications when compared with GMS30C32132 (A 32 bit RISC with DSP functionality) architecture. Also, the clock frequency that can be used for giving the same performance is comparatively large.

TABLE I

FFT PERFORMANCE COMPARISONS OF YD-RISC WITH DIFFERENT PROCESSORS.

DSP processors	Relative performance
YD-RISC	1
LGSGMS 30C32132	0.54
TI TMS320C50	0.31
ADSP2175	0.68
Motorola 56002	0.78
AT & T 1617	0.30
NEC u P77016	0.74

B. COOLRISC

CoolRISC [7] is a Harvard architecture designed for low power and low-voltage embedded applications. Conventional architectures like 68HCxx, PIC and 8051 cannot achieve a single CPI (Cycles Per Instruction). To achieve good MIPS, they need to be operated at high frequencies which inturn increase the power consumption. RISC processors, on the other hand, achieve one CPI, but cannot be used in 8 bit low cost applications due to software incompatibility, increase in code size and program memory and hardware complexity. Hence memory-memory architecture cannot achieve single CPI, whereas for the required frequencies of operation(1 to 10MHz), pure RISC architecture with its deep pipelining becomes too complex to manage. The trade-off is to go in for register-Memory architectures for 8 bit microcontrollers. Low power design requires the use of a lower clock frequency to achieve 1 MIPS. But at such low frequencies, deep pipelining is not required. Power consumption reduction of microcontroller-based applications can be achieved by designing low poer software that reduces the number of executed instructions as well as by selecting microcontroller architecture that minimizes the number of clocks xecuted per instructions. CoolRISC uses a 31 instructions Register-memory architecture i.e. one operand of an arithmetic instruction can be fetched from the memory and the other one in the register bank. It has a three stage pipeline to minimize the amount of hardware and power consumption. Branch instruction executed in only one clock cycle prevents branch delay. CoolRISC can achieve a strict CPI=1. Hardware stach for CALL instruction reduces the overhead of saving return address when compared to RISC. Low power operation is achieved through gated clock technique and by using a programmable internal clock. The MIPS/W figure can be improved by using low power design techniques like gated clock, along with multithread processing (processing several independent tasks parallel) and use of Thumb architecture. This figure can further be improved if power supply can be reduced without affecting speed.

C. RDA

RDA [8] (RISC for Data Acquisition) is a data-capturing device. It has a specialized instruction set to allow a flexible trigger event detection and storage memory control. A complete set of trigger conditions are implemented on the RDA based on the existing DASs (Data Acquisition Systems) RDA has a 16 –bit data path with 4 K data samples and a maximum of 10 Msmples/s which is an optimal point between cost and performance. A binary counter and static memories are used for circular buffering which enabled pre-trigger and post- trigger operation simultaneously. The architecture has a word recognizer feature wherein the user enters a word to the word recognizer whose output indicates whether a match occurred or not. Special instructions exist for the control and data capture of dat e.g. sample and stop instructions. Instructions described in the high level language and are compild into machine instructions by host before loading into the RDA. RDA can execute around 4096 trigger instructions. RDA executes 1 instruction per cycle with no pipeline implementation, since there is no time to flush the pipeline.

V. CONCLUSION

It can be concluded that the RISC philosophy, in general has proved superior to the CISC . Various architectures have been proposed each having its own pro’s and cons. the suitability of a particular architecture depends on the applications at hand. .pipelining, efficient cache implementation, power design consideration and speculative branching are some of the features that are used to improve the performance of the RISC architectures.

REFERENCES

[1] D.A. Patterson and C.H. Sequin, “RISC I: A Reduced Instruction Set VLSI Computer,”

[2] J.Henessey et al “ MIPS :A VLSI Architecture,” Stanford University, 1981.

[3] D. Greenley et al., “Ultra SPARC: the next generation superscalar 64 bit SPARC,” Compcon,

pp.442, 40th IEEE computer society International Conference (COMPCON'95), 1995

[4] H. Qing and H.C. " RNIW: a novel general purpose DSP architecture," Acoustics, Speech, and signal processing, 1996. ICASSWP-96

[5] H.B. Bakoglu et al., "IBM Second Generation RISC Machine Organization," *Computer Design: VLSI in Computers and Processors, 1989. ICCD'89. Proceedings., 1989 IEEE International Conference on*, 2-4 Oct. 1989, pp.138 - 142.

[6] B.I. Moon et al., "A 32-bit RISC Microprocessor with DSP Functionality," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E84-A, No.5, pp.1339-1347, January 2001.

[7] C. Piguet et al., "Low-Power Design of 8-bit Embedded CoolRISC Microcontroller Cores," *IEEE Journal Of Solid-State Circuits*, Vol. 32, No. 7, July 1997

[8] D.L. Gribble., "A RISC Architecture for High-Speed Data Acquisition," *Instrumentation and Measurement, IEEE Transactions on*, Vol. 43, Issue 3, June 1994, pp. 457 - 462