ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

A new approach for 8 bit core processor & its IP design for small scale systems

Pulkit Trivedi, Deepak Asati

***Abstract*— As we know a microprocessor is a general purpose IC which follow the instructions given to it, and the instructions set for the microprocessor designed such a way that it and handle any type of computations. Different type of architectures are available in the market like CISC, RISC, ARM etc. all of them have their own different approaches to perform computations. Our concept for this paper comes after mulling over all of these and we are proposing a new microprocessor architecture which will have CISC type instruction set (large instruction set good for multiple applications) and RISC type feature for executing every instruction in one cycle. To achieve these we just exclude few instructions (total five) of CISC. Though still we have covered all small scale and medium scale applications with our proposed architecture on cost of nothing and all sophisticated scale application on cost of extra few nanoseconds.**

***Index Terms*— CISC, microcode, opcode, RISC, slice registers, specifier.**

I. INTRODUCTION

From the architecture point of view, the microprocessor chips can be classified into two categories: Complex Instruction Set Computers (CISC) and Reduce Instruction Set Computers (RISC). In either case, the objective is to improve system performance. The debates between these two architectures made this research area very interesting, challenging, and sometimes confusing. CISC (Complex Instruction Set Computer) computers are based on a complex instruction set in which instructions are executed by microcode. Microcode allows developers to change hardware designs and still maintain backward compatibility with instructions for earlier computers by changing only the microcode, thus make a complex instruction set possible and flexible. Although CISC designs allow a lot of hardware flexibility, the supporting of microcode slows microprocessor performance because of the number of operations that must be performed to execute each CISC instruction. A CISC instruction set typically includes many instructions with different sizes and execution cycles, which makes CISC instructions harder to pipeline.[1] From the 60's CISC microprocessors became prevalent, each successive processor having more and more complicated hardware and more and more complex instruction sets. This trend started from Intel 80486, Pentium MMX to Pentium III. RISC (Reduced Instruction Set Computer) chips evolved around the mid-1970 as a reaction at CISC chips. In 70's, John Cocke at IBM's T.J Watson Research Center provided the fundamental concepts of RISC, the idea came from the IBM 801 minicomputer built in 1971 which is used as a fast controller in a very large telephone switching system. This chip contained many traits a later RISC chip should have few instructions, fix-sized instructions in a fixed format, execution on a single cycle of a processor and a Load / Store architecture. These ideas were further refined and articulated by a group at University Of California Berkeley led by David Patterson, who coined the term "RISC"[6]. They realized that RISC promised higher performance, less cost and faster design time. The simple load/store computers such as MIPS 2 are commonly called RISC architectures. David A. Patterson was the finder of the term RISC, after that John L. Hennessy invented the MIPS architecture to represent RISC.

Manuscript received June, 2012.

***Pulkit Trivedi**, VLSI & embedded system, RGPV/Gyan Ganga Institute of Tehnology & Sciences,. Jabalpur, India ,+919893735768*

***Deepak Asati**, Electronics & Communication Dept., Gyan Ganga Institute of Technology & Science ,Jabalpur, India, 8817792685.*

II. CONVENTIONAL RISC & CISC

When the term RISC was introduced a second term was created, Complex Instruction Set Computing, or CISC, which was basically a label applied to the existing popular computer architectures such as the IBM S/370, DEC VAX, Intel x86, and Motorola 680×0. Compared to the remarkably similar ISAs of the self-proclaimed RISC architectures, the CISC group was quite diverse in nature. Some were organized around large general purpose register files while others had just a few special purpose registers and were oriented to processing data in memory. In general, the CISC architectures were the product of decades of evolutionary progress towards ever more complex instruction sets and addressing modes brought about by the enabling technology of microcode control logic, and driven by the pervasive thought that computer design should close the "semantic gap" with high level programming languages to make programming simpler and more efficient.

In some ways CISC was a natural outgrowth of the economic reality of computer technology up until the late 1970′s. Main memory was slow and expensive, while read only memory for microcode was relatively cheap and many times faster.[1][2] The instructions in the so-called CISC ISAs tend to vary considerably in length and be tightly and sequentially encoded (i.e. the instruction decoder had to look in one field to tell if a second optional field or extension was present, which in turn would dictate where a third field might be located in the instruction stream, and so on). For example,

a VAX-11 instruction varied in length from 1 to 37 bytes. The opcode byte would define the number of operand specifiers (up to 6) and each had to be decoded in sequence because there could be 8, 16, or 32 bit long displacement or immediate values associated with each specifier. This elegant scheme is a delight for VAX assembly language programmers, because they could use any meaningful combination of addressing modes for most instructions without worrying if instruction X supported addressing mode Y. However, it would become a major hurdle to the construction of high performance VAX implementations within a decade after its introduction. Other CISC architectures, like x86, had a simpler and less orthogonal set of addressing modes but still included features that contributed to slow, sequential instruction decode. For example, an x86 instruction opcode could be preceded by an optional instruction prefix byte, an optional address size prefix byte, an optional operand size prefix byte, and an optional segment override prefix byte[3]. Not only are these variable length schemes complex and slow, but are also susceptible to design errors in processor control logic. For example, the recent "FOOF" bug in Intel Pentium II processors was a security hole related to the "$F0_{16}$" lock instruction prefix byte wherein a rogue user mode program could lock up a multi-user system or server.

To illustrate the large contrast between the instruction encoding formats used by CISC and RISC processors, the instruction formats for the Intel x86 and Compaq Alpha processor architectures are shown in Figure 1. In the case of x86 there is a lot of sequential decoding that has to be accomplished (although modern x86 processors often predecode x86 instructions while loading them into the instruction cache, and store instruction hints and boundary information as 2 or 3 extra bits per instruction byte). For the Alpha (and virtually every other classic RISC design) the instruction length is fixed at 32-bits and the major fields appear in the same locations in all the formats. It is standard practice in RISC processors to fetch operand data from registers (or bypass paths) even as the instruction opcode field is decoded.

When RISC processors first appeared on the scene most CISC processors were microcoded monsters with relatively little instruction execution pipelining. Processors like the VAX, the 68020, and the Intel i386 for the most part processed only one instruction at a time and took, on the average, five to ten clock cycles to execute each one. The first RISC processors were fully pipelined, typically with five stages, and averaged between 1.3 and 2.0 clock cycles to execute an instruction.[2][3] RISC-based microprocessors typically were more compact and had fewer transistors (no microcode) than their CISC counterparts, and could execute at higher clock rates. Although programs compiled for RISC architectures often needed to execute more native instructions to accomplish the same work, because of the large disparity in CPI (clocks per instruction), and higher clock rates, the RISC processors offered two to three times more performance. In Table 1 is a case study comparison of an x86 and RISC processor of the early RISC era (1987).

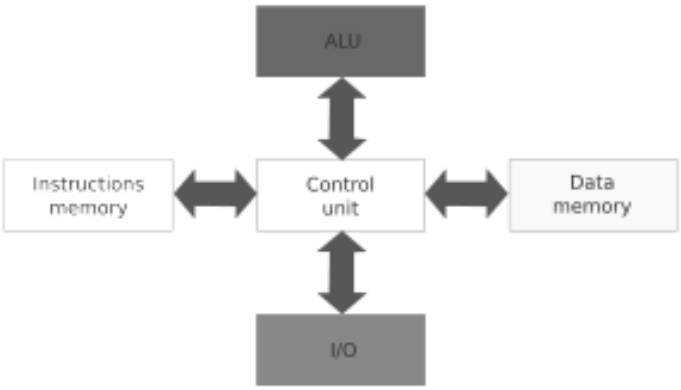

Figure 1 Conventional design

III. PROPOSED ARCHITECTURE

Our concept for this paper comes after mulling over all of these and we are proposing a new microprocessor architecture which will have CISC type instruction set (large instruction set good for multiple applications) and RISC type feature for executing every instruction in one cycle. To achieve these we just exclude few instructions (total five) of CISC. Though still we have covered all small scale and medium scale applications with our proposed architecture on cost of nothing and all sophisticated scale application on cost of extra few nanoseconds.

The proposed design is shown in Figure 2 which comprises of the mixture of both RISC type and CISC type architechture. It employs the double Harvard architechture which further subdivides the program memory into instruction memory, immediate value and immediate address

In order to satisfy the functionality requirements of electronic products from the user, the architecture of electronic product is getting more and more complicated. Therefore, there are many different kinds of successful design methodologies to eliminate the design complexity of system-on-chip and application-specific processors. By combining the former successful researches and our implementation experience, a compromising design methodology is proposed as Figure 1. Generally, the architecture is a top-down design except the gray functional block. These functional blocks represent the hardware implementation. We adapt the bottom-up design here to expect we can make a better use of the pre-designed basic hardware components, for example, the half/full adders, flipflops, or comparators and increase the reuse ratio of the hardware resource and make our proposed design methodology more efficient. We propose a new application specific RISC processor architecture to overcome the performance bottleneck of traditional RISC processor.

.

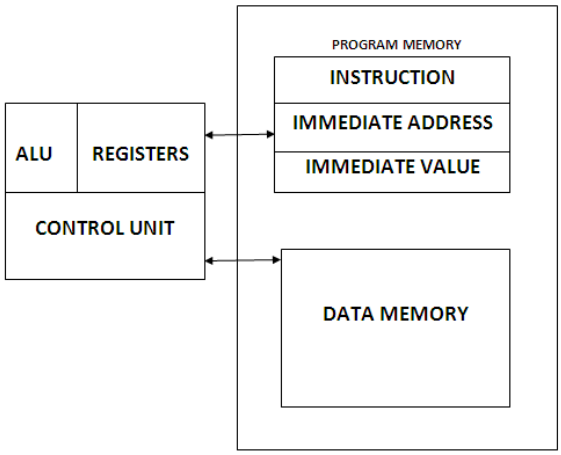

Figure 2 New proposed design

There is still considerable controversy among experts about which architecture is better. Some say that RISC is cheaper and faster and therefor the architecture of the future. Others note that by making the hardware simpler, RISC puts a greater burden on the software. Software needs to become more complex. Software developers need to write more lines for the same tasks. Therefore they argue that RISC is not the architecture of the future, since conventional CISC chips are becoming faster and cheaper anyway. RISC has now existed more than 10 years and hasn't been able to kick CISC out of the market. If we forget about the embedded market and mainly look at the market for PC's, workstations and servers I guess a least 75% of the processors are based on the CISC architecture. Most of them the x86 standard (Intel, AMD, etc.), but even in the mainframe territory CISC is dominant via the IBM/390 chip. RISC and CISC architectures are becoming more and more alike. Many of today's RISC chips support just as many instructions as yesterday's CISC chips. The PowerPC 601, for example, supports *more* instructions than the Pentium. Yet the 601 is considered a RISC chip, while the Pentium is definitely CISC. Furthermore today's CISC chips use many techniques formerly associated with RISC chips.

An important factor is also that the x86 standard, as used by for instance Intel and AMD, is based on CISC architecture. X86 is thé standard for home based PC's. Windows 95 and 98 won't run at any other platform. Therefore companies like AMD an Intel will not abandoning the x86 market just overnight even if RISC was more powerful. Changing their chips in such a way that on the outside they stay compatible with the CISC x86 standard, but use a RISC architecture inside is difficult and gives all kinds of overhead which could undo all the possible gains. Nevertheless Intel and AMD are doing this more or less with their current CPU's. Most acceleration mechanisms available to RISC CPUs are now available to the x86 CPU's as well. Since in the x86 the competition is killing, prices are low, even lower than for most RISC CPU's. Although RISC prices are dropping also a, for instance, SUN UltraSPARC is still more expensive than an equal performing PII workstation is. Equal that is in terms of integer performance. In the floating point-area RISC still holds the crown. However CISC's 7th generation x86 chips like the K7 will catch up with that. The one exception to this might be the Alpha EV-6. Those machines are overall about twice as fast as the fastest x86 CPU available. However this Alpha chip costs about €20000, not something you're willing to pay for a home PC. Maybe interesting to mention is that it's no coincidence that AMD's K7 is developed in co-operation with Alpha and is for al large part based on the same Alpha EV-6 technology.

The biggest threat for CISC and RISC might not be eachother, but a new technology called EPIC. EPIC stands for **E**xplicitly **P**arallel **I**nstruction **C**omputing. Like the word parallel already says EPIC can do many instruction executions in parallel to one another.

EPIC is a created by Intel and is in a way a combination of both CISC and RISC. This will in theory allow the processing of Windows-based as well as UNIX-based applications by the same CPU.

It will not be until 2000 before we can see an EPIC chip. Intel is working on it under code-name *Merced*. Microsoft is already developing their Win64 standard for it. Like the name says, Merced will be a 64-bit chip.

If Intel's EPIC architecture is successful, it might be the biggest thread for RISC. All of the big CPU manufactures but Sun and Motorola are now selling x86-based products, and some are just waiting for Merced to come out (HP, SGI). Because of the x86 market it is not likely that CISC will die soon, but RISC may.

The new architecture also uses the pipelining operation of the processors hence leading to use the clock cyle efficiently.

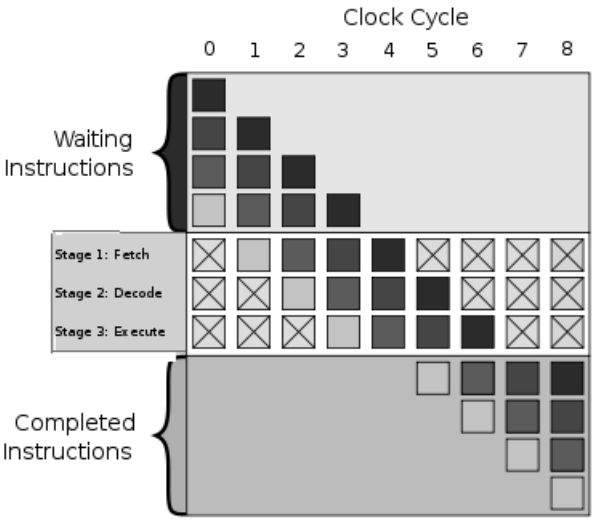

Figure 3 Pipelining

IV. PROBLEM & REMEDY

Since we have apllied the pipelining architecture this will lead to a limitation which causes the JUMP instruction to fail. In order to remove this we will fetch and decode the JUMP instruction at the same cycle which help us in utilising the machine cycle efficiently.

V. RESULT & SIMULATION

The Proposed core architecture has been designed and simulated with the help of Xillinx ISE software. The result is as shown below.

Logic Utilization	Used	Ref1
No. of slice registers	98	110
No. of slice LUT's	37	42
No. of fully used bit slices	33	36
No. of bonded IOB's	4	6
No. of block RAM/FIFO	1	2
No. of BUFG/BUFGCTRLs	3	5

Device Utilization Summ	
Logic Utilization	**Used**
Number of Slice Registers	98
Number of Slice LUTs	37
Number of fully used Bit Slices	33
Number of bonded IOBs	4
Number of Block RAM/FIFO	1
Number of BUFG/BUFGCTRLs	3

Figure3 Simulation Result for proposed design

VI. CONCLUSION

A new trend of CISC and RISC architectures is addressed. This paper gives a new trend of fast processor design with less no. of splices used. Some of previous works was highlighted, and a new technology is presented. For the best performance and scalability processor, the following are important factors: (a) fast cache-to-cache communication, (2) large L2 or shared capacity, (3) fast L2 access delay, and (4) fair resource (cache) sharing.

REFERENCES

[1] Yanfen Chen*,Wuchen Wu,Ligang Hou,Jie Hu, "Design and Implementation of 8-bit RISC MCU," *IEEE* 978-1-4244-6736-5/$20.0©2010 IEEE.
[2] Tomas Balderas-Contreras, Rene Cumplido *, Claudia Feregrino-Uribe "On the design and implementation of a RISC processor extension for the KASUMI encryption algorithm . Sciencedirect, Computer and electrical engineering 34 (2008)
[3] D. Seal, ARMArchitecture Reference Manual. 2nd Edition, Addison-Wesley 200 1.
[4] M. K. Jain, M. Balakrishnan, and A. Kumar, "ASIP Design Methodologies: Survey and Issues," Proceeding of Fourteenth International Conference on VLSIDesign, pp. 76-8 1, Jan. 2001.
[5] K. Wakabayashi, and T. Okamoto, "C-Based SoC Design Flow and EDA Tools: An ASIC and System Vendor Perspective," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 19, No. 12, pp. 1507-1522, Dec. 2000.
[6] M. Gschwind, "Instruction set selection for ASIP design," Proceeding of the Seventh of International Workshop on Hardware/Software Codesign, pp. 7-1 1, May 1999.
[7] I. H. Jeng, F. Lai, and Y. D. Tseng, "FACE: Fine-tuned Architecture Codesign Environment for ASIP Development," Design Automation for Embedded Systems, Kluwer Academic Publishers, Vol. 4, No. 4, pp. 329-25 1, Oct. 1999.
[8] J. Park, K. Muhammad, and K. Roy, "High-performance FIR Filter Design Based on Sharing Multiplication," IEEE Transactions on Very Large Scale Integration System, Vol. 11, No. 2, pp. 245-253, Apr. 2003.
[9] P. Bougas, P. Kalivas, A. Tsirikos, and K. Z. Pekmestzi, "Pipelined Array-Based FIR Filter Folding," IEEE Transactions on Circuit and System, Vol. 52, No. 1, pp. 108-118, Jan. 2005.
[10] C. C. Lee, and S. P. Cheng, "Application-Specific RISC Architecture Enhancements: Circular Buffering of Registers for Efficient Filtering," Proceedings of 2004 International SOC Design Conference, Seoul Korea, Oct. 2004.
[11] V. G. Oklobdzija, "An Algorithmic and Novel Design Of a Leading Zero Detector Circuit: Comparison with Logic Synthesis," IEEE Transactions on Very Large Scale Integration Systems, Vol. 2, No. 1, pp. 124-128, Mar. 1994.

Pulkit Trivedi currently pursuing M.tech. in VLSI design and embedded systems form Gyan Ganga Institute of Technology & science.
Deepak Asati currently working as an Asst. Professor at Gyan Ganga Institute of Technology & Science.