American Journal of Applied Sciences 5 (11): 1528-1534, 2008
ISSN 1546-9239
© 2008 Science Publications

Implementing a Large Data Bus VLIW Microprocessor

[1]Weng Fook Lee and [2]Ali Yeon Md Shakaff
[1]Emerald Systems Design Center, Kompleks Sri Sg Nibong, Bayan Lepas, 11900, Penang, Malaysia
[2]School of Computer and Communication Engineering, University Malaysia Perlis,
Arau, 02600 Perlis, Malaysia

Abstract: Microprocessors have grown tremendously in its computing and data crunching capability since the early days of the invention of a microprocessor. Today, most microprocessors in the market are at 32 bits, while the latest microprocessors from IBM, Intel and AMD are at 64 bits. To further grow the computational capability of a microprocessor, there are two possible paths. One method is to increase the data bus size of the microprocessor to 128/256/512 bits. The larger the data bus size, the more data can be crunched at any one time. The second method is to implement multiple microprocessor core in a single microprocessor unit. For example, Intel's Pentium 4 Dual Core and AMD's Athlon Dual Core both have two microprocessor core within a single microprocessor unit. Latest from Intel and AMD are quad core microprocessors with four microprocessor core within a single microprocessor unit. Both methods have its advantages and disadvantages. Both methods yields different design issues and have different engineering limitations. This research looks into the possibility of implementing a large data bus size VLIW microprocessor core of 256 bits on the data bus. VLIW is chosen as opposed to CISC and RISC due to its ease of scalability.

Keywords: Large data bus size microprocessor, multicore

INTRODUCTION

Micro-processors and micro-controllers are widely used in the world today. It is used in everyday electronic systems, be it a system used in the industries or a system used by consumers. Complex electronic systems such as ATM machine, POS systems, financial systems, transaction systems, control systems, database systems all uses some form of micro-controller or micro-processor as the core of their system. Consumer electronic systems such as home security systems, credit cards, microwave ovens, cars, cell phones, PDA, refrigerators and other daily appliances have within the core of the system either a micro-controller or micro-processor.

What is a micro-controller and micro-processor? If they are such a big part of our daily life, what exactly is their function?

Micro-processors and micro-controllers are very similar in nature. In fact, from a top level perspective, a micro-processor is the core of a micro-controller. A micro-controller basically consists of a micro-processor as its CPU (central processing unit) with peripheral logic surrounding the micro-processor core. As such it can be viewed that a micro-processor is the building block for a micro-controller.

A microprocessor's capability to crunch data is dependent on its bus width. A larger data bus width allows the microprocessor to crunch more data at any one time. For example, the crunching capability of a 32 bit microprocessor is at a comparable doubling factor of a 16 bit microprocessor. Therefore having a microprocessor with larger data bus size allows for more data crunching capability. However there is a drawback to using larger data bus size. The larger the data bus size, the greater amount of logic is required and the larger the die size. Most microprocessors in the market today such as Intel's Xeon and EMT64 microprocessor, AMD's Athlon 64 and Opteron microprocessor, IBM's PowerPC microprocessor are 64 bit microprocessors. They are able to crunch data at 64 bits at a time.

Moving forward, in order to have a microprocessor to have more data crunching capability, there are two methods of progress:

- increase the data bus size from 64 bits to 128/256/512 bits and beyond
- increase the amount of microprocessor core in a single microprocessor

Corresponding Author: Weng Fook Lee, Emerald Systems Design Center, 737-1-10 Kompleks Sri Sg Nibong, Jalan Sultan Azlan Shah, 11900 Bayan Lepas, Penang, Malaysia Tel: +604-6461113 Fax: +604-6448181,

Method (1) increases the data bus width to accommodate for more data crunching capability, while method (2) uses multiple microprocessor core in a single microprocessor to allow for multiple activities. Each method has its advantages and disadvantages.

INCREASING DATA BUS SIZE TO IMPROVE DATA CRUNCHING POWER OF MICROPROCESSORS

One obvious method to increase the data crunching capability of microprocessors is by increasing its data bus size. This allows the microprocessor to crunch more data at any one time[1,2,3,4,5,6,20,21,22].

The advantages of this method:

- The large data bus size microprocessor can process large amounts of data at any one time. This will make the microprocessor ideally suited for applications that require data crunching
- Although power consumption increases when the data bus size is doubled from 64 to 128, the increase in power consumption is lower compared to having two 64 bit microprocessor core in a single microprocessor unit. Thermal management is more manageable on the 128 bit microprocessor as compared to dual 64 bit microprocessor core, as the larger bit size microprocessor has lower logic density
- The larger bit size microprocessor can still fully utilize the memory bandwidth since effectively there is still one microprocessor addressing one memory bus. This is a clear advantage over multicore microprocessor whereby the memory bandwidth is reduced to 1/n (n is the amount of microprocessor core in the microprocessor unit)[7,8,9]

This method however creates several disadvantages:

- Increasing the data bus size of a microprocessor to large bus size of 256/512 provide a design challenge especially in terms of engineering resource (design schedule and engineering man-power)
- When a microprocessor is increased from 64 bits to 128/256/512 bits data bus size, the amount of die area increases tremendously This increase is due to:
  a) paths and logic will have to be duplicated to cater to the larger data bus size. This would lead to more logic and larger die size
  b) having a 128/256/512 bit data bus means that the amount of metal routing on silicon for that bus will have to be increased to 128/256/512 as well. This will increase the physical layout area of the design and results in larger die size
- Increasing the data bus size from 64 bits to 128/256/512 means that the ALU (Arithmetic Logic Unit)[10,20,21,22] within the microprocessor will also need to increase its computation capability to accommodate 128/256/512 bits. This increases the ALU logic and thus increases the die size.
- When a microprocessor is increased in bus width, the die size increases, the amount of logic required for the design increases and power consumption increases
- As the die size of the microprocessor increases, the probability of having defects per wafer also increases[11]. This means that yield will drop therefore further increasing the cost of the microprocessor
- Software and OS support is required for a system to take advantage of the additional data bus size. Most software and OS today only supports 32 bits and 64 bits operations. To fully utilize the crunching capability of 256/512 bits, the software and OS must be updated to address the larger data bus size

MULTIPLE MICROPROCESSOR CORE TO IMPROVE DATA CRUNCHING POWER OF MICROPROCESSORS

Apart from increasing the data bus size of the microprocessor, one other method to improve the data crunching power of a microprocessor is to put multiple microprocessor core within one microprocessor unit[12,13,14,20,21,22]. This means that a microprocessor unit may have two or more microprocessor core within that one package. The multiple core may be on one silicon or on separate silicon within the package. This method allows the two separate microprocessor core to crunch data separately and therefore increasing the crunching capability of the system.

The advantages of this method:

- Multiple core on the same silicon have smaller travel time allowing cache coherency logic circuits to operate at higher frequency as compared to travel time to external of the chip package[15]
- Multicore microprocessors shares level 2 cache among the cores to allow faster access to data for each core from the same cache[16]

- Design risks are lower as the multicore microprocessor unit is basically using multiple core which is proven to work on a single microprocessor unit. This allows for lower design risk compared to designing a large super size microprocessor
- Having two microprocessor core in a single microprocessor unit reduces the motherboard area compared to having two microprocessor unit on the motherboard. This helps to reduce cost on the PCB of the motherboard

The disadvantages of this method:

- Putting two or more microprocessor core in a microprocessor unit strains the thermal management of the microprocessor package as heat generated from both core will increase the temperature of the package significantly
- Yield during production will drop due to the difficult integration of the multiple core[11]
- Operating systems utilizing this multi core microprocessor must be able to optimally support the additional core resource. This means that the software will have to be able to perform multiprocessing efficiently
- If the multicore microprocessor shares one same memory bus, the memory bandwidth is reduced to 1/n, whereby n is the amount of microprocessor core in the microprocessor unit[7,8,9]
- This research looks into the possibility of implementing a large data bus size VLIW microprocessor core of 256 bits on the data bus

USING VLIW MICROPROCESSOR FOR LARGE DATA BUS SIZE IMPLEMENTATION

The first microprocessor was developed by Intel Corp in 1971. It was called 4004[23]. It was a simple IC chip, by today's standards of microprocessor design, but back in 1971, the 4004 was a revolutionary break-through from Intel Corp. The original design of the 4004 was meant for a calculator called Busicom[23]. Intel Corp however saw the potential of the 4004 and used it for applications other than just for a calculator.

From the humble beginnings of the first microprocessor in the world, it has grown by leaps and bounds and today's computational capability of microprocessors far surpass that of the original design.

Looking at the history of microprocessor design, there are three distinct types of microprocessor that have played an important role in the growth and development of the microprocessor; CISC, RISC and VLIW[26,27]. CISC (Complex Instruction Set Computing) is based on the concept of allowing programmers to use as little instructions as possible to write programs for a microprocessor. CISC consists of many instructions, ranging from simple basic microprocessor instructions to complex instructions. The idea was to use as few instructions as possible to write a program for a microprocessor. CISC microprocessors were largely popular until John Cocke from IBM Research[24,25] brought forward the idea that most CISC microprocessors utilizes very little of the complex instructions while the simple basic microprocessor instructions were utilized heavily. From this idea, the RISC microprocessor was developed.

Apart from the CISC and RISC microprocessors, there is a different generation of microprocessor based on a concept called VLIW (Very Long Instruction Word). VLIW microprocessors make use of ILP (Instruction Level Parallelism - the ability to execute multiple instructions in parallel). VLIW microprocessors are not the only type of microprocessors that takes advantage of executing multiple instructions in parallel. Superscalar superpipeline microprocessors are also able to achieve this parallelism.

For this research, VLIW is chosen for building the large data bus size microprocessor as opposed to CISC or RISC because of its hardware simplicity to allow for up scaling of data bus size.

IMPLEMENTATION OF LARGE DATA BUS SIZE VLIW MICROPROCESSOR ON FPGA

A 64 bit custom instruction set VLIW microprocessor core is implemented on Altera's Stratix II FPGA (EP2S180F1508I4) using the operations of arithmetic, logic, load, read and compare[28,29,30,31,32], creating a minimal customized instruction set of 16 instructions:

1. nop
2. add
3. sub
4. mul
5. load
6. move
7. read
8. compare
9. xor
10. nand

11. nor
12. not
13. shift left
14. shift right
15. barrel shift left
16. barrel shift right

Altera's EP2S180F1508I4 FPGA is used as it needs to have adequate elements and enough usable IO pins for implementation of the VLIW microprocessor core. The implemented microprocessor core is a 4 stage pipeline, 3 parallel pipes superscalar VLIW microprocessor[19,26,27]. The microprocessor core is designed with a shared register file with 16 registers accessible by all 3 pipes. Each register's width is the same as that of the data bus.

Figure 1 shows the top level block diagram of the microprocessor core. Figure 2 shows the system level implementation utilizing the microprocessor core implemented on FPGA. The cache, prefetch, branch prediction is implemented external to the microprocessor core on the system level as the FPGA (implemented microprocessor core) has limited resource. Figure 3 shows the micro-architecture of the 4 stages (fetch, decode, execute and writeback and shared register file).

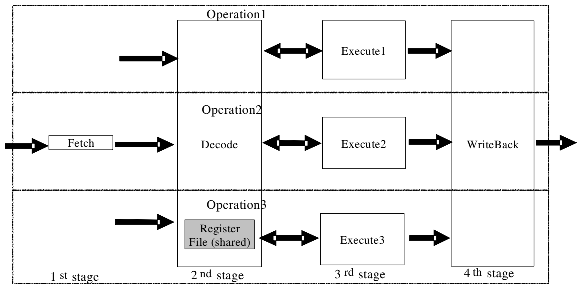

Fig. 1: Diagram showing top level block diagram of the microprocessor core

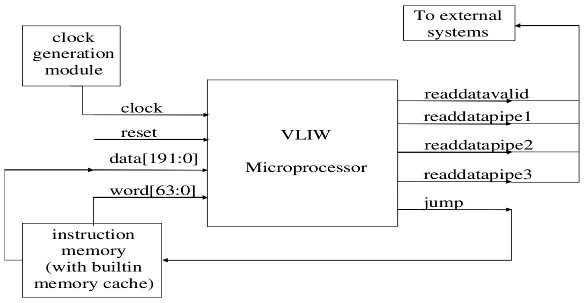

Fig. 2: Diagram showing system level implementation utilizing microprocessor core on FPGA

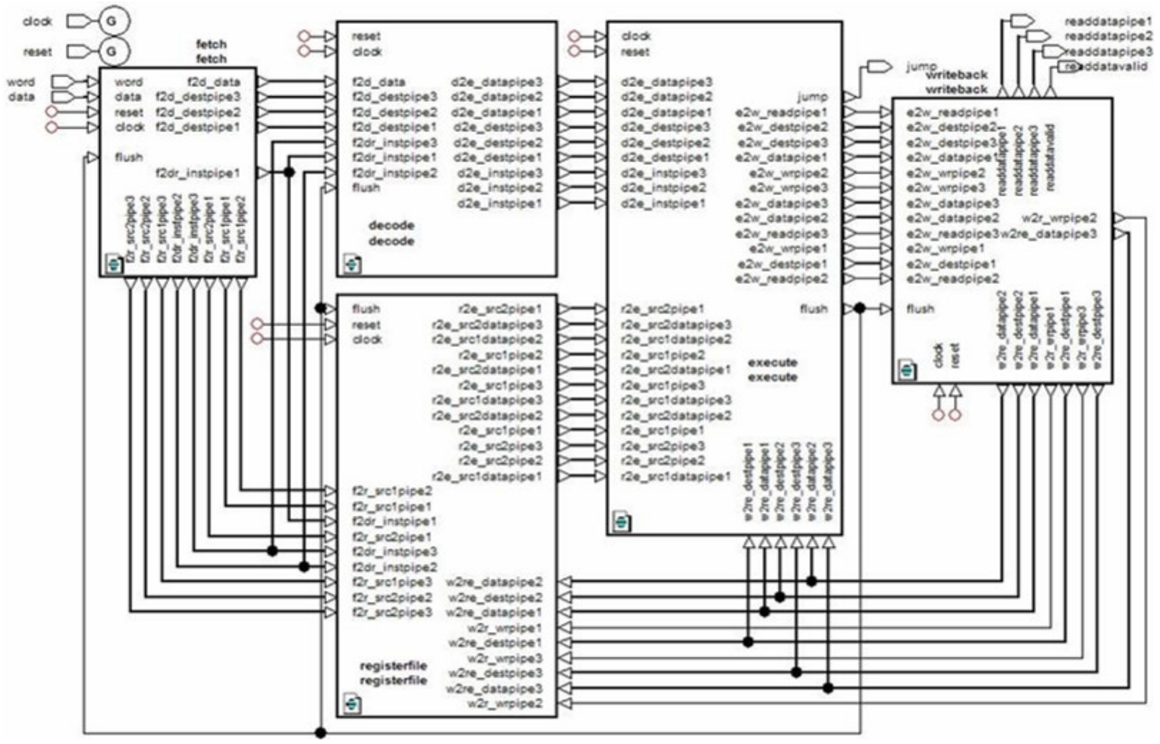

Fig. 3: Diagram showing micro-architecture of the 4 stages of the microprocessor core

The execute stage of all 3 parallel pipes are also designed with register bypass mechanism to cater for all cases of instruction dependency[21]. For an n (n = 3 to 6) pipe superscalar pipeline microprocessor, the register bypass mechanism must cater for a total of y number of conditions that require register bypassing[21].

Intra-pipe register bypass conditions = n (n+2)
Inter-pipe register bypass conditions = n^4+n^2-2n
Total conditions = y = n^4+2n^2 = 81 + 15 = 99

Register bypass logic is implemented for all 15 conditions of intra pipe and 84 conditions of inter pipe bypass, resulting in a total of 99 bypass conditions.

IMPLEMENTATION RESULTS OF LARGE DATA BUS SIZE VLIW MICROPROCESSOR CORE ON FPGA

Each pipe stage for all 3 parallel pipes in the microprocessor core are designed in verilog RTL and synthesized onto Altera's EP2S180F1508I4 FPGA using Altera's Quartus II full version 6.0 (synthesis, fitting, P and R) while verilog RTL simulation is done using Mentor Graphics's Modelsim version 6.1.

The verilog RTL is written using only combinational logic and sequential logic and does not use any IP components or library components from Altera's MegaIPCore Library. For the microprocessor core in 64 bit data bus, the critical path delay, FPGA cell element usage and power consumption is analyzed.

The microprocessor core on the FPGA is then expanded to larger data bus size of 96/128/160/192/224/256 bits and the same data is collected for each implementation. When the microprocessor core's data bus size is expanded from 64 bits to larger data bus size, the internal core logic as well as the system level logic is expanded to accommodate the microprocessor core's larger data bus size. From the results obtained, the normalized power-delay product for each data bus size implementation is calculated and plotted as shown in Fig. 4. Power-delay product is commonly not a priority factor considered in FPGA design as FPGA draws more power compared to ASIC[17,18,33,34,35,36]. However for this research, the normalized power-delay product is important as it provides an indication reference on the power-delay product increase per the increase in bus size implementation. This indication reference can serve as a good reference point when migrating the large data bus size implementation into ASIC. The FPGA cell element usage is plotted and shown in Fig. 5.

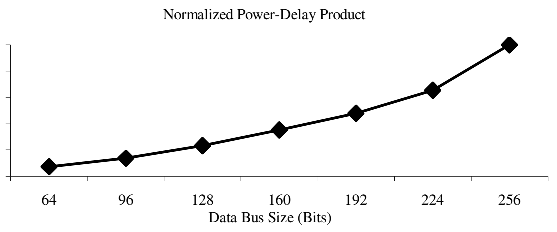

Fig. 4: Diagram showing normalized power-delay product for different data bus size

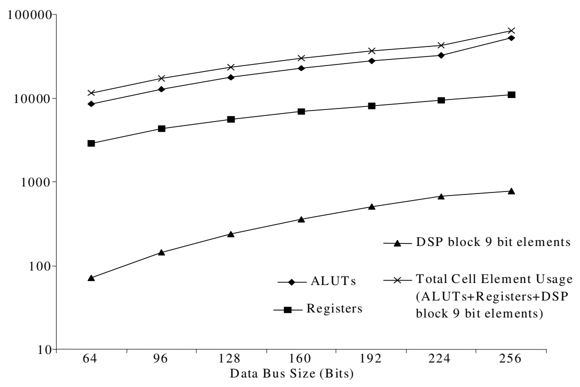

Fig. 5: Diagram showing FPGA cell element usage for different data bus size

Referring to Figure 4, the power-delay product increases by 3x when the data bus size is increased from 64 to 128 bits. When the data bus size increased from 64 to 256 bits, the power-delay product increases by 13x. This shows a steep increase in power-delay product when the data bus size is increased fourfold to 256 bits.

As for the FPGA cell element usage (Fig. 5), ALUTs increases by 1x when data bus size increased from 64 to 128 bits. It however increases by 5x when the data bus size is increased from 64 to 256 bits. The sequential elements (registers) increases by 1x when data bus size increased from 64 to 128 bits. It increases by 3x when the data bus size is increased from 64 to 256 bits. The DSP 9 bit block element increases by more than 2x when data bus size increases from 64 to 128 bits. It however increases by almost 10× when the data bus size increases from 64 to 256 bits. The steep increase in DSP 9 bit block element is due to larger ALU that requires computation of larger chunks of data as the data bus size is increased. For the total FPGA cell elements usage, it increases by 1x when data bus size increase from 64 to 128 bits. However when data bus size increase from 64 to 256 bits, the total FPGA cell elements usage increases by 4.6x.

From the results of Figure 4 and Figure 5, increasing the data bus size comes at a certain cost. To double the data bus size from 64 bits to 128 bits results in an increase of 3x in power-delay product and an increase of 1x in total FPGA cell elements usage. However when the data bus size is quadrupled to 256 bits, there is an increase of 13x in power-delay product and an increase of 4.6x in total FPGA cell elements usage.

CONCLUSION

From the results of the implemented microprocessor core on FPGA as shown in Figure 4 and Figure 5, increasing the data bus size comes at a certain cost. To quadruple the data bus size from 64 bits to 256 bits, there is a increase of 13x in power-delay product and an increase of 4.6x in total FPGA cell elements usage.

Moving forward, more work is needed to expand the VLIW microprocessor's VLIW instruction set to encompass a larger set of operations, A software assembler and compiler must also be created to allow assembly language to be compiled for the VLIW microprocessing environment. This will create a full system level implementation for the large data bus size VLIW microprocessor.

Once the infrastructure to build a system level implementation for a large data bus size microprocessor is achieved, software applications can be modified or rewritten for the large data bus size environment. By running real time software applications on a large data bus size microprocessing environment, more data can be collected on real time applications that can be used to benchmark the performance/cost/power advantages/disadvantages of a large data bus size VLIW microprocessor in comparison to that of multicore microprocessing environment.

ACKNOWLEDGEMENTS

Special thanks to Bernard Lee, CEO of Emerald Systems and Azrul Halim, Director of Design Engineering, Emerald Systems for funding this project.

REFERENCES

1. IBM Inc. (2006, Nov 21). IBM WebSphere Application Server 64-bit Performance Demystified White Paper
2. Vladimir Romanchenko. (2006, June 27). Evolution of the multi-core processor architecture Intel Core: Conroe, Kentsfield. Digital Daily.
3. Microsoft Inc. (2006). Benefits of Microsoft Windows 64 Edn. White Paper.
4. White Paper: Intel's multi-core processors move forward. (2006, Jan). Computer Power User. 6 (1).
5. Advanced Micro Devices, Inc. (2005, June 14). 86-64TM Technology White Paper.
6. Advanced Micro Devices, Inc. (2003, April 20). 64-bit Technology:Driving The Digital Media Revolution White Paper.
7. Richard Goering, 2007. Speaker cites multicore benchmarking challenges. EE Times.
8. Bader, D.A., Varun Kanade and Kamesh Madduri, 2007. SWARM: A Parallel Programming Framework for Multicore Processors. IEEE 1-4244-0910-1/07
9. Alan Zeichick, 2005. Driving in the Fast Lane: What Multi-Core Computing Means for Programmers. AMD Developer Central.
10. Godfrey P. D'Souza, 1998. Arithmetic logic unit with improved critical path performance. US Patent #5764550
11. Christopher W. Hampson, 1997. Redundancy and high-volume manufacturing methods. Int. Technol. J.
12. Benson Inkley and Scott Tetrick, 2006. Intel Multi-core Architecture and Implementations. Intel IDF.
13. Bob Valentine, 2006. Inside the Intel Core (TM) Microarchitecture. Intel IDF.
14. Advanced Micro Devices Inc., 2005. Multi-core Processors-The Next Evolution In Computing. 33211A.
15. Advanced Micro Devices Inc. AMD Developer Central, 2006. AMD's Multiple Threads, Multiple Cores, Multiple Gains. Peter Aitken.
16. Ramanathan, R.M., 2006. Intel® Multi-Core Processors Making the Move to Quad-Core and Beyond-Intel. White Paper. Technology@Intel Magazine
17. Weng Fook Lee, 2003. Verilog Coding For Logic Synthesis. John Wiley. pp: 3-14.
18. Weng Fook Lee, 2000. VHDL Coding and Logic Synthesis With Synopsys. Academic Press Publication. pp: 1-23
19. Weng Fook Lee, 2007. VLIW Microprocessor Hardware Design for ASIC and FPGA. McGraw Hill. pp: 1-23.
20. Weng Fook Lee, Ali Yeon Md Shakaff. implementation results on increasing data bus size on a 4 stage pipe in a pipeline superscalar microprocessor core implemented on FPGA. Proceedings of 10th EuroMicro Conference on Digital System Design, WIP, SEA-Publications SEA-SR-16, July 2007-11-23. ISBN 978-3-902457-16-5.

21. Weng Fook Lee, Azrul Halim, Nor Hisham, Yap Vooi Voon, Lo Hai Hiung, Patrick Sebastian. Implementation results on register bypass conditions of an n-parallel pipes superscalar pipeline microprocessor core on FPGA. Proceedings of 10th EuroMicro Conference on Digital System Design, WIP, SEA-Publications SEA-SR-16, July 2007-11-23. ISBN 978-3-902457-16-5.
22. Weng Fook Lee, Ali Yeon Md Shakaff, 2007. Implementation of 128/256 Bit Data Bus Microprocessor Core on FPGA. Journal of Programmable Devices, Circuits and Systems. International Congress of Global Sci. and Technol., 7 (1): 7-13.
23. Intel Corp. Intel's First Microprocessor-the Intel® 4004.
24. John Cocke, V. Markstein. IBM Corp. The Evolution of RISC Technology at IBM.
25. MIT Inventor of the Week Archive. John Cocke. Reduced Instruction Set Computing. Aug 1999.
26. John L Hennessy and David A. Patterson. (2003). Computer Architecture: A Quantative Approach. Morgan Kaufmann Publication.
27. John L Hennessy and David A. Patterson, 2003. Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann Publication.
28. Weng Fook Lee, 2000. VHDL Coding and Logic Synthesis With Synopsys. Academic Press Publication.
29. IBM Corp. 2000. PowerPCTM Microprocessor Family: The Programming Environments for 32-Bit Microprocessors
30. Intel Corp, 1999. Intel Architecture Software Developer's Manual.
31. Advanced Micro Devices Corp, 1997. Am186 and Am188 Family Instruction Set Manual.
32. Intel Corp, 1994. i960® Jx Microprocessor Instruction Set and Register Quick Reference.
33. Emerald Systems Design Center, 2007. IC Design Synthesis Using RTL Training Course.
34. Emerald Systems Design Center, 2007. IC Design Using Verilog HDL Training Course.
35. Emerald Systems Design Center, 2007. Intermediate Verilog Coding for IC Design Training Course.
36. Emerald Systems Design Center, 2007. FPGA Implementation and Verification Training Course.