Reconfigurable Processor Core Design for Network-on-a-Chip

Shih-Lun Chen, Jer-Min Jou, Chien-Ming Sun, Yuan-Chin Wu, Haoi Yang, Hong-Yi Su
Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R. O. C.
jou@j92a21.ee.ncku.edu.tw

Abstract-*Today, the cost of mask in SoC is increasing rapidly. Designing a complex system-on-a-chip (SoC) confronts many challenges. Networks-on-a-chip (NoC) is a new architectural template, which helps to meet many of these challenges and enables fast time to market for new designs. How to transfer high-speed and macro data for computing and reusing macro transistors proves to be more and more important in the area of IC design. The reconfigurable processor turns into a research focus by many SoC researchers in the world.*

This paper offers a new powerful, flexible, and reusable reconfigurable processor for NoC, which will process no matter general or special purpose applications with high performance. It could handle software well like a superscalar CPU and process some special applications with macro data efficiently like an ASIC. Our results show that the reconfigurable processor design greatly improves the performance than traditional processor design.

Keywords: Reconfigurable, NoC

1. Introduction

Recently, a single chip may contain up to one billion transistors; with such massive resources, SoC designers can implement much complex hardware on a chip. Those designers now face many unusual challenges, such as performance and power, reusability, adaptability, and scalability.

An efficient solution to these problems is to treat SoCs as *micronetworks*, or *Networks on Chips* (NoCs) where the interconnections are designed using an adaptation of the protocol stack [1]. Networks have regular structure, so the design of global wires could be fully optimized and as a result their properties are more predictable. Regularity enables design modularity, which provides a standard interface for easier component reuse and better interoperability.

The reconfigurable computing research and design are development items nowadays. The reconfiguration provides a lot of advantages in design for designers [2]. No matter in reducing design cost, shorting design time, diminishing the difficulties and improving the integrating of IP components, the reconfiguration plays an important role.

The MATRIX (Multiple Alu architecture with Reconfigurable Interconnect eXperiment) [3] is a multi-granular array of 8-bits BFUs (Basic Functional Units) with procedurally programmable instruction memory and a controller which can generate local control signals from ALU output by a pattern matcher, a reduction network or 0 half NOR PLA.

The CHESS Array [4] hexagonal array features a chessboard-like floorplan with interleaved rows of alternating ALU / switchbox sequence. Embedded RAM areas support high memory requirements. Switchbox can be converted to 16 word by 4 bits RAMs if needed.

TRIPS [5] contains mechanisms that enable the processing cores and the on-chip memory system to be configured and combined in different modes for instruction, data, or thread-level parallelism. To adapt to small and large-grain concurrency, thTRIPS architecture contains four out-of-order, 16-wide-issue Grid Processor cores, which can be partitioned when easily extractable fine-grained parallelism exists.

This paper will introduce how to design a reconfigurable processor for NoC. Section 2 introduces the architectures of NoC and reconfigurable circuits. Section 3 presents the reconfigurable processor architecture and describes details of three kinds of components, communication components, basic processor components and reconfigurable components of it. Section 4 describes the scheduling of reconfigurable processor. Section 5 presents how to hierarchically design a hardware and how to map application data flow to hardware hierarchically. Section 6 shows the simulation results and components costs of processor with reconfigurable design or not, which would verify that reconfigurable processor is an advantaged design.

2. NoC System Architectures

The topology of the network in NoC is 2D mesh topology. It is easily mapped to 2D layout and expanded. Taking figure 1 for an example, this is a 16-node 2D mesh network and the pair of (x, y) represents the address of nodes in the network.

The NoC design is the combination of dynamic compositions of heterogeneous IP blocks, which require that the on-chip interconnect network is scalable, programmable, and reusable. There are four main components types in the NoC:

(1) Reconfigurable communication components: contain interconnect network, routers (communication controllers), and network interface.
(2) Reconfigurable computation components: they could be reconfigurable processors, DSPs, or application-specific logics, i.e. FPGAs, or ASICs.
(3) Distributed global/local memory components are used to store shared or unshared data.
(4) Application interface (AI): contains computational instructions and communicational primitives.

The reconfigurable processor is the most important component of the NoC. It's a reconfigurable platform and enables NoC could be hierarchically mapped application data flow. The NoC is also a reconfigurable platform for hierarchical data flow mapping.

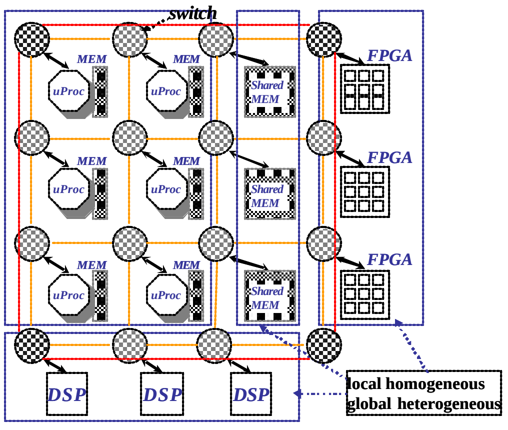


Figure 1 Architecture of NoC.

2.2 Reconfigurable Architectures

The reconfigurable architectures could be classified into fine-grained circuits and coarse-grained circuits [6]. Fine-grained reconfigurable circuits consist of an array of CLBs (Configurable Logic Blocks) with a path width of 1 bit, which are embedded in a reconfigurable interconnect fabrics.

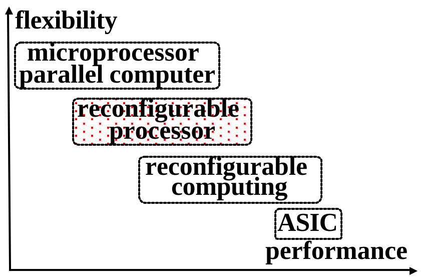


Figure 2 Reconfigurable processor orientation.

Coarse-grained reconfigurable circuits consist of an array of CFBs (Configurable Functional blocks), also called rDPU (reconfigurable Datapath Unit).

The coarse-grained architectures support in arithmetic layer, data-path in character layer and the powerful and area efficient in routing switches. The most advantage in the coarse-grained architecture is reducing the configuration memories and configuration time.

Figure 2 shows that the reconfigurable processor is bridging the gap between reconfigurable computing and microprocessors parallel computer. The key point to choice which architecture to design is dependent on what kind of purpose the users would need. As shown in the figure, the reconfigurable processor has more flexibility than reconfigurable computing and ASIC, and more performance than microprocessor parallel computer.

3. Reconfigurable Processor Architecture

This section presents instruction set and three kinds of components which are communication, basic processor, and reconfigurable components in reconfigurable processor. Figure 3 shows entire reconfigurable architecture example, we can easily find that there are sixteen function units in this reconfigurable processor design.

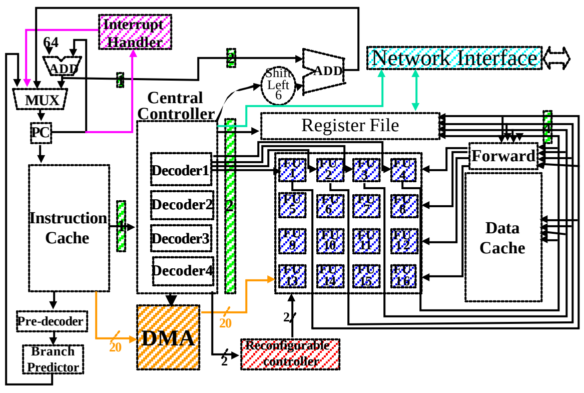


Figure 3 Reconfigurable processor architecture.

3.1 Instruction Set

After comparing with CISC and RISC instruction set, the very long instruction word (VLIW) [7] is the best and most suitable for the reconfigurable CPU design. As figure 4 shown, there are four 32-bit slots in one instruction and the length of each instruction is 128-bit.

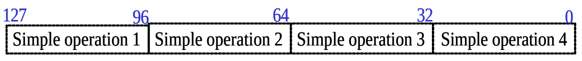


Figure 4 VLIW instruction format in the reconfigurable processor design.

There are five types of slots in reconfigurable processor. Register, immediate, jump and reconfigurable type slots are used for calculation and much like MIPS instructions with a modify bit, “r” for separating the slot types are reconfigurable or not. The final type is communication type slot that are used for communications.

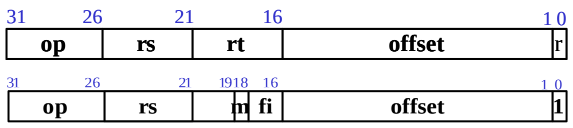


Figure 5 Reconfigurable type slots.

Figure 5 shows the reconfigurable type slots. The above one could be immediate type or running reconfigurable slot depend on the last bit “r” flag. If r flag equals zero, this slot would be immediate type slot. If it equals one, this slot is reconfigurable running slot. So, reconfigurable processor uses the one bit “r” flag to determine this slot is normal operation or reconfigurable operation. The below one is reconfigurable download slot. It would provide the information about start address, frame memory number and download amount. The DMA will use such information to execute the action of download reconfigurable operations.

3.2 Communication Components

The main communication component in reconfigurable processor is network interface. It manages the data transfer in NoC. Figure 6 shows a data transfer example from tile A to tile B. If the reconfigurable processor tile A wants to send a data to other tile B, maybe a global memory or other reconfigurable processor, the reconfigurable processor would decode the communication instruction and then send the data address and destination to network interface.

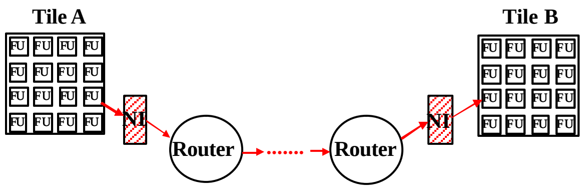


Figure 6 Data transfer from tile A to tile B.

Then the network interface will make packages and then send to routers by some signals handshaking. After the router receiving the packages, it would send the packages to other routers by routing algorithm. Finally the router will send the packages to the network interface of tile B, the network interface would compose the packages to data. The receiving data would be stored in the receiving buffer of the network interface and waiting the receiving instruction to load the data to register file or store to data caches of tile B. After such sending and receiving actions, accomplish a communication action.

3.3 Basic Processor Components

Combining the program counter datapath and arithmetic datapath, we could get a basic processor. It contains instruction cache for instructions reading, data cache for data reading and writing, register file for temporary storing data for computing, decoder for decoding instructions to produce control signals, ALU for executing arithmetic operation and addresses generation for load and store, and interrupt handler for handling I/O interrupt.

The reconfigurable processor is a five-stage pipeline design. The first stage is fetch stage which uses the address stored in PC to read instructions from instruction cache. The second stage is decode stage which decodes instructions to produce control signals and reads the data from register file. The third state is execution stage which executes the arithmetic operations and calculates addresses for load and store. The fourth stage is memory stage which loads data from data cache or stores data into it. The fifth stage is write back stage which reads data from pipeline register between memory and write back stage and determines to write it into the register file or not

The forward circuit is designed for handling the data hazard which caused when a data needed to operate in this clock, but it must write back register file several clocks later. The forwarding action is getting the data early from the internal resources before it write to register file in fifth stage. The branch predictor is designed for handling the control hazard which caused by branch instructions. The reconfigurable processor uses Hysteresis counter two-bit self predictor. If the prediction is correct, the pipeline will continues execute. Contrarily, if the prediction is incorrect, the instructions that were fetched and decoded must be discarded and the instructions that were executed, loaded or stored in memory and written back would continues execute.

3.4 Reconfigurable Components

The special components designs in reconfigurable processor are reconfigurable controller, DMA and reconfigurable design in function unit and combines with some registers, circuits, wires and memories for reconfiguration.

The reconfigurable controller controls which frame needed in this reconfigurable computing in all function units. As FIGIGRE 5 shown, the reconfigurable controller gets the running-times value from central controller for decision and then sending control signals to function units.

The principle of the DMA design of reconfigurable CPU is modicum like common DMA. As shown in figure 7, the data could transfer directly and rapidly from instruction cache to frames (in M1 or M2) for reconfiguration, this action called download reconfiguration. This download action could execute in the same time with running other instructions.

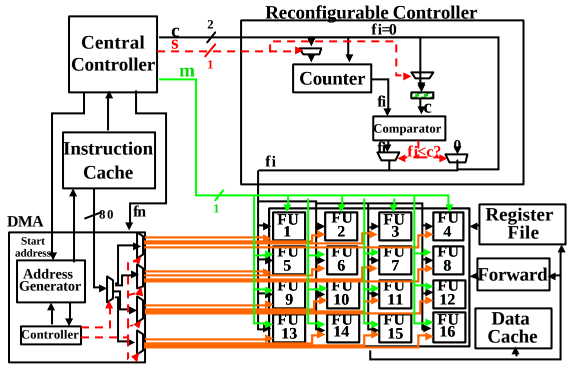


Figure 7 Reconfigurable controller and DMA architectures.

Figure 8 shows the architecture of the most important arithmetic component function unit. As the figure shown, the function unit connects with the decoder, reconfigurable controller, DMA, register file, forward, and data cache. Different from traditional general purpose processor, there is a special memory called frame memory, consisted of four parts including operation, source one, source two and destination register file addresses, of the function units in reconfigurable processor.

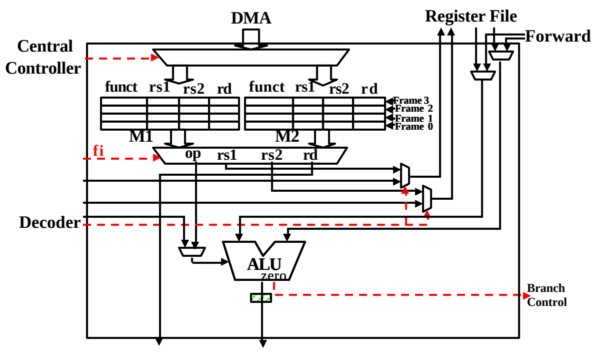


Figure 8 Function unit architecture.

The frame memories are stored the reconfigurable operations in frame 0 to frame 3 in M1 or M2 by DMA. The M1 and M2 are four-frame memories, which are designed for storing reconfigurable operations for reconfigurable computations. When the function unit runs M1 or other non-reconfigurable instruction, the DMA would download reconfigurable operations from instruction
cache in the same time.

The function unit additional combined an address generator for the reconfigurable computation because when it runs reconfigurable computation, the ALU would get the operation and source addresses from frame memory and execute them. At the same time, the processor could load data from data cache to register file, the load address is produced by the additional adder.

4. Reconfigurable Processor Scheduling

The reconfigurable processor is a dynamic run-time reconfigurable design. It could handle software well like a superscalar CPU and process some special applications with macro data efficiently like an ASIC. Figure 9 shows its running schedule. In the start it runs normal operation 1 instructions like a superscalar CPU, but at the same time the reconfigurable operations are downloaded from instruction cache to the frame memory M1 of each function unit by DMA. As normal operation 1

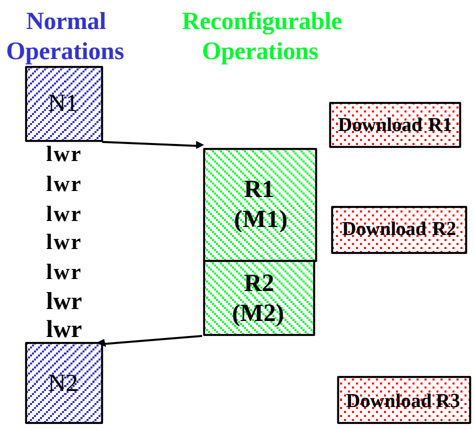
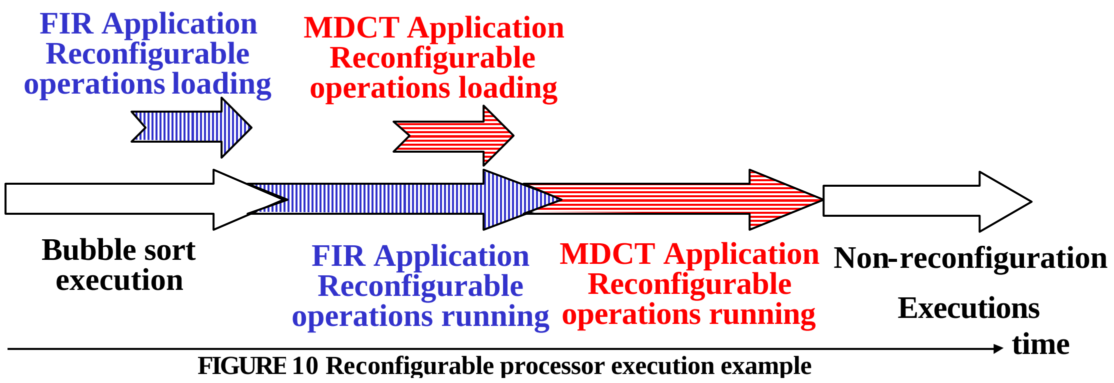


Figure 9 Reconfigurable processor scheduling.



FIR Application Reconfigurable operations loading
MDCT Application Reconfigurable operations loading
Bubble sort execution
FIR Application Reconfigurable operations running
MDCT Application Reconfigurable operations running
Non-reconfiguration Executions
time

FIGURE 10 Reconfigurable processor execution example

finished the reconfigurable operation 1, an special application like hardware, had already downloaded to frame memories of all function units by DMA. So, the reconfigurable processor could execute reconfigurable operation 1 right away.

During executing reconfigurable operation 1, the processor not only could load data that would be used later by additional adder of function units but also could download reconfigurable operation 2 from instruction cache to the other frame memory M2 of each function unit by DMA at the same time. In the same way, the reconfigurable operation 2 executes closely following reconfigurable operation 1. After finishing reconfigurable operation 2, the processor could execute normal operation 2 or other special application like hardware. Figure 10 shows the timing flow of a reconfigurable processor running example. Firstly, the processor runs normal operation bubble sort. At the same time, the FIR application reconfigurable operations are downloaded into M1.

After finishing the non-reconfigurable bubble sort execution, the processor runs FIR application reconfigurable operations continuously. At the same time, the MDCT application reconfigurable operations are downloaded to M2. After finishing the FIR executions, the processor runs MDCT application reconfigurable operations continuously. The processor runs the non-reconfigurable executions again following with the end of MDCT reconfigurable application reconfigurable operations finished.

In order to estimate the performance of the reconfigurable processor, we define SR (Speedup Ratio) for analyzing our design [8]. SR is computed as follows:

NoCP time =Normal operation time + Reconfigurable operation time.
Normal operation time = SW_only_ time – SW_loop_time
Reconfigurable time = SW_loop_time/RS

$$S=\frac{\text{Normal\_only\_time}}{\text{Normal time} + \text{Reconfigurable time}}$$

$$=\frac{\text{Normal\_only\_time}}{\text{Normal\_only\_time}-\text{SW loop time} + \dfrac{\text{SW\_loop\_time}}{\text{RS}}}$$

$$=\frac{1}{1-\dfrac{\text{SW\_loop\_time}}{\text{RS}}\left(1-\dfrac{1}{\text{RS}}\right)}$$

Where Normal_only_time is the time from normal operation only execution and the SW_loop_time is the time taken on a CPU by the loop that will be mapped to hardware. *RS* (Reconfigurable speedup) denotes the speedup expected on the loop by mapping it to reconfigurable computation.

Using bubble sort, FIR and MDCT for example, the SR value is 2.36. So, the reconfigurable processor design greatly improves the performance than traditional processors by saving loading and download time.

6. Simulation Results

This section will introduce the results of analyzing several design alternatives, including 1-function-unit, 2-function-units, 4-function-units, 9-function-units and 16-function-units of the processors with reconfigurable design and non-reconfigurable design by C language.

In order to evaluate the design effectiveness of different architecture designs, a general application example FIR filter and MDCT are used. Figure 13 shows the simulation results of FIR and Figure 14 shows the simulation results of MDCT, Notice the unit of y coordinate axis, it represents the hardware could produce how many results each clock. We could easily find that the reconfigurable design is greatly promoting the performance in FIR and MDCT application than general processors.

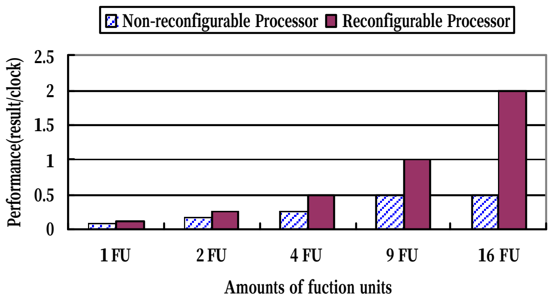


Figure 13 Compare of performance between architectures with and without reconfigurable design in FIR example.

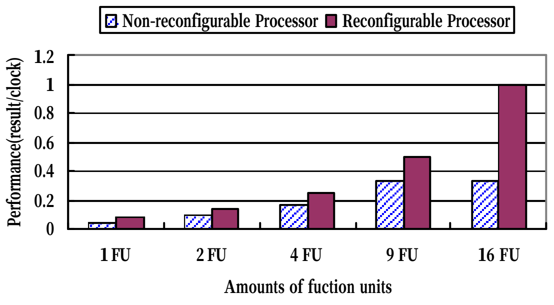


Figure 14 Compare of performance between architectures with and without reconfigurable design in MDCT example.

Table 1 shows the execution time of three applications bubble sort, FIR, and MDCT. The bubble sort application depends on too many branch

Time(clock) / Application	Download Time	Normal Time	Reconfigurable Time	Total Time	Normal Operation Only Time
Bubble Sort	0	1010	0	1010	1010
FIR	16	10	5000	5010	20010
MDCT	16	10	20000	20010	40010

Table 1 Execution time of bubble sort, FIR, and MDCT applications

instructions. So its not suitable for reconfigurable computing. The FIR and MDCT applications are suitable for reconfigurable computing. Both of them cost 16 clocks time for download, but it could be saved by executing other instructions and downloading at the same time. To observe the simulation results, the applications using both of normal operation and reconfigurable operation for executing are greatly improving the performance that only using normal operations.

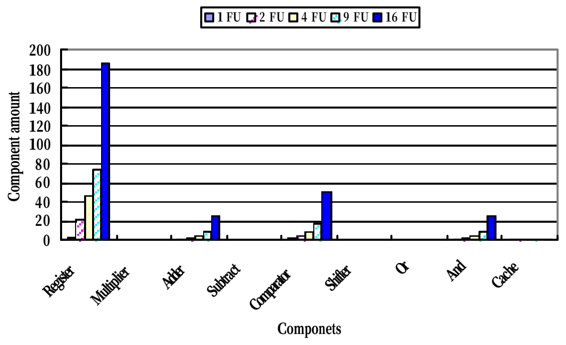


Figure 15 Components increasing amounts of each kind of processor architectures adding reconfigurable design.

Figure 15 shows the components increasing amounts of each kind of processor architectures from the architectures without reconfigurable design to with reconfigurable design. We could easily find that the multipliers, Subtracts, shifters and caches are zero growing-up in component amounts. The increasing components percentages of the and gates are from 3.13% to 7.69%, the comparators are from 4.44% to 21.46%, the adders are from 5.46% to 34.25% and the registers are from 4.92% to 40.74%.

After analyzing the results, the performances are increasing from 62% to 300% and the hardware components costs are all less than 50%. We could get the conclusion that the processor with reconfigurable design is more efficient than traditional processors.

7. Conclusion

We proposed a coarse-grained dynamic reconfiguration processor. It could handle software well like a superscalar CPU and process some special applications with macro data efficiently like an ASIC. After simulation, we verify that Reconfigurable processor is more efficient in special applications and has more flexibilities than traditional general purpose processors We hope one day, the reconfigurable processor could combine with the network interface, interconnection network and routers well, and then complete the NoC design. Finally we look forward to accomplish NoC mesh for entire hierarchical dataflow mapping into a chip.

References

[1] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, A. Sangiovanni-Vincentelli, "Addressing the System-on-a-Chip interconnect Woes through Communication-Based Design," Design Automation Conference, pp. 667-672, 2001.

[2] Jou, Jer Min: Reconfigurable SoC Architectures, The 14th VLSI Design/CAD Symposium, pp.217-200, 2003.

[3] Hauck, S.; Hosler, M.M.; Fry, T.W: High performance carry chains for FPGA's Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , Volume: 8 , Issue: 2 , April 2000.

[4] A. Marshall et al.: A Reconfigurable Arithmetic Array for Multimedia Applications; Proc. ACM/SIGDA FPGA'99, Monterey, Feb. 21-23, 1999.

[5] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D.C. Burger, S.W. Keckler, and C.R."Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture, Moore, 30th Annual International Symposium on Computer Architecture (ISCA), June 2003.

[6] Hartenstein, R.;" A decade of reconfigurable computing: a visionary retrospective", Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001.

[7] http//www.semiconductors.philips.com/acrobat/other/vliw-wp.pdf.

[8] Dinesh C. Suresh, Riverside Walid A. Najjar, Riverside Frank Vahid. "Profiling Tools for Hardware/Software Partitioning of Embedded Applications". Proc. of the 2003 ACM SIGPLAND Conf. on Languages, Compiler and Tools for Embedded Systems, Sam Diego, CA Juen 2003.