VLIW Implementation of a Multiprocessor Reconfigurable Architecture

Arnaldo Azevedo[1], Rodrigo Soares[2], Ivan Saraiva Silva[2], Flavio Wagner[1], Sérgio Bampi[1]

[1]*Universidade Federal do Rio Grande do Sul*
Instituto de Informática

[2]*Universidade Federal do Rio Grande do Norte*
Departamento de Informática e Matemática Aplicada

apafilho@inf.ufrgs.br; rodrigo@lcc.ufrn.br; ivan@dimap.ufrn.br; {Flavio, bampi}@inf.ufrgs.br

Abstract

The X4CP32 is an architecture that combines both Parallel and reconfigurable paradigms. It consists of grid of Reconfigurable and Programming Unit (RPU), responsible for all the processing and program flow. This paper presents architectural modification in order to maximize the computational use of the Cells in a RPU. A change to a very large instruction word (VLIW) philosophy in the RPU was implemented to reach the objective. This changes raise the Instructions per Cycle of the RPU from 0.5 to 1 with no area overhead and no influence in clock frequency.

1. Introduction

The Reconfigurable Architectures' [1, 2, 3] importance is increasing, as it's a good tradeoff between ASICs performance and microprocessor's flexibility. ASICs are usually very efficient on performing the tasks it was developed to, but are incapable to perform any other application. In the other hand, a microprocessor can perform virtually any task, but that usually implies in a low performance. Between these two extremes are the Reconfigurable Architectures (RA), which try both to combine ASIC's and microprocessor's advantages and overcome their weaknesses.

The X4CP32 [4, 5, 6] is a reconfigurable microprocessor that also uses the Parallel paradigm, to obtain programmability uncommon to other RAs. The parallel processing, along with the flexibility to execute the most important part of a code systolically, brings new horizons to the current system designing.

In order to accelerate the data processing by the processor, this paper explores the design space of the Reconfigurable and Programming Unit (RPU). To utilize as much parallelism as possible, with the present components, a very large instruction word (VLIW) philosophy was implemented in the RPU.

The next section presents the architectural modifications. Section 3 brings the results of the implementation. The paper ends presenting the conclusions, in Section 4, and the references in the Section 5.

2. X4CP32 VLIW

The X4CP32 [4, 5, 6] architecture doesn't achieve its full potential. The RPU, when working in Programming Execution Mode, leaves for the top left Cell the responsibility to distribute the instructions to others Cells in the same RPU. This task takes operational cycles from the Cell, which can't process data while doing the distribution. Other weak point is that distributing an instruction to another Cell takes 2 clock cycles. Since the Cells take 4 cycles to execute an instruction, except multiplication, that takes 6 cycles, the top left Cell can only send instructions to other 2 Cells, to keep them totally busy. This means that from the 4 Cells available to compute, only 2, or 3 when all of them are multiplying, can really be used to process data at the same time. To solve this problem we propose a VLIW implementation of the RPU.

VLIW describes an instruction-set philosophy in which the compiler packs a number of simple, non-interdependent operations into the same instruction word. When fetched from cache or memory into the processor, these words are easily broken up and the operations dispatched to independent execution units.

In the original design of RPU, each instruction has the information of in which Cell it will be executed. The top left Cell, where the Instruction-Memory lays,

verifies this field and sends the instruction to the destined Cell, or executes it if the instruction is destined to itself.

This process doesn't exist in the VLIW implementation. The control, in the VLIW version, fetches the whole instruction word and verifies if the Cells are ready to receive their instruction. If so the top left Cell starts to execute its part of instruction while sends the rest to the others Cells in RPU. If not, the execution stalls and waits until all Cells are ready. So the longest Cell instruction, in a VLIW instruction word, defines the time taken to execute it.

The size of the VLIW word is 128 bits long. This is the sum of the four 32-bit instructions of each Cell. The VLIW word is formed by 4 Cell instructions. Each group of 32 bits is destined to a different Cell. As each Cell has an independent instruction, and keeps its total functionality, a reduction of the VLIW word wasn't possible.

The branch logic is placed in the top left Cell, where is stored the Cell Level program. The correspondent field for this Cell in the VLIW word is the first one. The other instructions in the same VLIW word are executed in parallel with the branch evaluation. In others words, the instructions, in a same VLIW word, are executed independently of the result of the jump instruction in the top left Cell.

To implement the VLIW structure just small changes were necessary. Once the Cells organization are very close to a VLIW fashion, changes were done in the top left Cell control logic and in its instruction memory. The main difference is that now the control waits until all Cells in the RPU are prepared to receive to send the instructions.

No changes were implemented in the others Cells of the RPU. The original mechanism of receiving and synchronization of instructions suffered no changes. The original organization of the RPU is very favorable to a VLIW implementation, so this is a natural evolution.

3. Results

Table I shows the synthesis results for conventional and VLIW architectures. The VHDL was synthesized with Quartus II 4.0 from Altera. The results presented are the same of the conventional implementation, except for the memory bits that have increased in 24,576bits. Results refer a 1k positions G-MEM rather the 64k of the architecture specification.

In the Programming Execution Mode the maximum Instruction per Cycle (IPC) of the RPU raises from 0,5 to 1, since the four Cells can now process data in parallel, avoiding the instruction distribution overhead.

Table I - Synthesis Results

Device	EP2A15B724C7
Logic Elements	13,227
Total memory bits	73,728
Clock	53.7 Mhz

This means an increasing factor of 2 with no area overhead and no influence in the clock frequency. This change has no effect in the Reconfigurable Execution Mode.

4. Conclusions and future works

This paper presented the modification of the RPU of the XP4C32 processor to a very large instruction word, VLIW, philosophy. The paper demonstrated the methodology applied to increase the IPC of the RPU using its already described structure.

The next step is the development of an assembler that supports the VLIW implementation. Other work to be done is an implementation of a pipeline in the data path of the Cells.

5. References

[1] R. Harteinstein, "Reconfigurable Computing: the Roadmap to a New Business Model – and its Impact on SoC Design", SBCCI Tutorial – 2001.

[2] Mirsky, E., DeHon, A., "Matrix: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources", IEEE 1996.

[3] E. Waingold, M. Taylor, D. Srikishna, V.Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, A/ Agarwal, "Baring It All to Software:Raw Machines", September -1997.

[4] Azevedo, A.; Soares, R.; SIlva, I. S.; A new hybrid parallel/reconfigurable architecture: The X4CP32. Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on, Sept. 8-11, 2003, Pp: 225-230.

[5] Soares, R.; Azevedo, A.; Silva, I.S.; "X4CP32: A New Parallel/Reconfigurable General-Purpose Processor"; 15th Symposium on Computer Architecture and High Performance, 2003.

[6] Soares, R.; Azevedo, A.; Silva, I.S.; "X4CP32: A Coarse Grain General Purpose Reconfigurable Microprocessor"; Parallel and Distributed Processing Symposium, 2003. Proceedings. International, April 22-26, 2003 P. 171–178.