

Multithreaded Extension to Multicenter VLIW Processors for Embedded Applications

Domenico Barretta, William Fornaciari, Mariagiovanna Sami
Politecnico di Milano, Dipartimento di Elettronica e Informazione

Daniele Bagni
STMicroelectronics

Abstract

Instruction Level Parallelism (ILP) extraction for multicenter VLIW processors is a very hard task. In this paper, we propose a retargetable architecture that can exploit ILP and thread level parallelism jointly, thus allowing an easier parallelism extraction and improving the performance with respect to traditional multicenter VLIW processors.

1. Introduction

In current Very long Instruction Word (VLIW) architectures, exploiting high amounts of latent Instruction Level Parallelism (ILP) may be a very difficult task [4], because the degree of ILP is mainly limited by the number of functional units that can work simultaneously when they are connected to a single register file.

Multicenter VLIW architectures tackle this problem by introducing more than one register file and clustering the functional units based on the register files they are connected to. This approach allows higher ILP levels than the base VLIW architectures. However, since each functional unit can access just its own register file, communication may have a significant negative impact on the overall performance when data are moved from a register file to another.

Furthermore, with few exceptions, complex applications do not exhibit a constantly predominant type of parallelism through the set of procedures of which they are composed and massive ILP can be found only in some segments of the application, leading to low usage of VLIW CPU resource. This fact limits the efficiency of highly parallel architectures when executing such an application and is the reason for the interest in flexible architectures that can exploit different types of parallelism in order to increase the overall performance. Some examples of this trend are constituted by [1, 3, 6, 7]. All the cited examples are based on super-scalar architectures.

Other architectures that exploit both ILP and thread level parallelism are dynamically scheduled architectures. In our

case we must confront with the problem of introducing thread level parallelism in a “pure” VLIW processor, that is therefore totally static.

2. The proposed architecture

The goal of our work is introducing thread level parallelism in a multicenter VLIW processor by allowing each cluster to execute a different thread. Since this approach is very scalable, we will consider for simplicity a processor that is composed of two clusters.

The main idea is to design a processor that can switch at run time between two different computation models. In *ILP mode* all clusters execute one single thread using long instructions (*bundles*) composed of up to $2M$ possible operations (where M is the issue width). In *MT mode* each cluster executes a different thread using bundles composed of up to M operations.

Instruction issue unit (IIU) and instruction cache are strictly correlated, therefore they must be designed jointly so that both ILP and MT fetch behaviors may be implemented.

Consider a single-ported, n -way set associative (where n is the number of clusters) multibank instruction cache composed by $2M$ banks. In ILP mode we can fetch $2M$ operations from the only program counter in use. In MT mode, at each cycle we fetch $2M$ instructions - starting from the address pointed by the program counter of a thread - and store the instructions in a buffer. At the following cycle, we fetch from the other program counter. This solution allows the issue, at each cycle, of M instructions per thread, while reading from the cache $2M$ consecutive instructions, as described in Fig. 1. The control signal named *TS* (Thread Selection) is applied either to select the program counter to access the instruction cache or to enable the input for the correct instruction buffer. The addressing logic between the instruction buffer and the instruction issue unit is used to select the correct M operations in the buffer to be issued.

In the original architecture we have just one branch unit that computes the program counter. Since the proposed architecture may execute instructions from more than

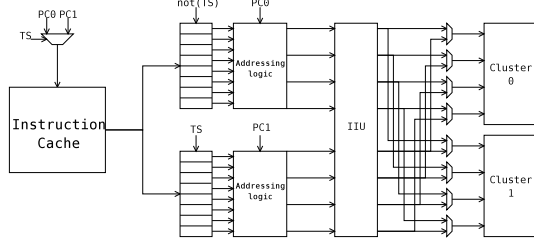


Figure 1. The proposed instruction issue mechanism

one control flow, we need a branch unit for each program counter we introduce. We need therefore a branch unit per cluster. This allows a simple way for each cluster to update its program counter based on the subroutine in execution.

In ILP mode, the buffers are bypassed, and the *TS* control signal is forced to always be zero, so that the program counter from cluster 0 is always sent to the instruction cache.

Other parts of the architecture have to be replicated, such as memory protection units, control registers, and so on. Furthermore, memory protection units must be modified to tackle the problem of possible concurrent memory operations accessing the same address.

Decision about which parts of the code should be run in ILP mode and which parts should be run in MT mode must be taken by the programmer. Source code is by default compiled for ILP mode and the programmer must explicitly call a function to create threads to be run in MT mode. The call is very similar to the traditional “create_thread” call, with the exception that one thread per cluster must be created. The compiler automatically inserts mode switch instructions and schedules the code for the correct execution mode.

3. Experimental results

In order to evaluate the performance improvements that can be obtained by our method, we have done several experiments by using as driving application an industrial benchmark hereinafter called *jpeg2ppm*. Two application kernels were also used: *Aes* encryption/decryption and *Matrix Product*.

Comparisons are provided with a reference multicluster VLIW architecture named ST200 [5]. The simulator for the proposed architecture was developed by modifying a cycle accurate Instruction Set Simulator (ISS), jointly developed by *STMicroelectronics* and *Università di Genova* [2] for micro architecture modeling of media processors and embedded systems. The performance obtained on a two clustered modified architecture was compared with the perfor-

mance of a two clustered ST200 that exploits only ILP. The percentage improvement in Table 1 is computed as $\frac{Cycles_{New} - Cycles_{Old}}{Cycles_{Old}} \cdot 100$. Max IPC indicates the theoretical parallelism extracted by the compiler, whereas actual parallelism indicates the average number of instructions executed in a clock cycle, including stall cycles.

	Aes	Jpeg2ppm	Matrix
RISC ops	1.24%	2.04%	8.73%
Active cycles	-45.72%	-19.16%	-49.24%
Inactive cycles	-13.99%	-2.19%	-39.46%
Total cycles	-33.74%	-11.88%	-46.87%
Max IPC	86.51%	26.21%	114.27%
Actual IPC	52.77%	15.79%	104.63%

Table 1. Performance Comparison

4. Concluding remarks

In this paper, we proposed a multicluster VLIW architecture that can switch at run-time between two different computation modes, ILP and MT, in order to better exploit the types of parallelism present in each part of an application.

A very important consideration is scalability: ILP exposure does not increase dramatically when the number of clusters of a VLIW processor grows beyond two. The performance of the proposed architecture on MT portions of code is, instead, expected to scale well with the number of clusters.

References

- [1] <http://developer.intel.com/technology/hyperthread/>.
- [2] I. Barbieri, M. Bariani, and M. Raggio. A VLIW architecture simulator innovative approach for HW-SW co-design. In *Proceedings of ICME 2000*, 2000.
- [3] R. Espasa and M. Valero. Simultaneous multithreaded vector architecture: Merging ILP and DLP for high performance. *Proceedings of the 4th international conference on high performance computing*, pages 350–357, 1997.
- [4] P. Faraboschi, J. A. Fisher, and C. Young. Instruction scheduling for instruction level parallel processors. In *Proceedings of the IEEE*, volume 89, pages 1638–1659, November 2001.
- [5] P. Faraboschi and F. Homewood. ST200: A VLIW architecture for media-oriented applications. *Microprocessor Forum 2000*. San Jose, CA, 2000.
- [6] C. Molina, A. Gonzalez, and J. Tubella. Trace level speculative multithreaded architecture. In *Proceedings of the 2002 IEEE International Conference on Computer Design*, pages 402–407, 2002.
- [7] J. Tsai et al. The superthreaded processor architecture. *IEEE Transaction on computers*, pages 881–902, Sept. 1999.