In: IEEE International Symposium on Circuits and Systems, ISCAS 2009, Taipei, Taiwan, pp.2213-2216, May, 2009

Design of Interleaved Multithreading for Network Processors on Chip

Henrique C. Freitas[†], Felipe L. Madruga, Marco A. Z. Alves, Philippe O. A. Navaux
Graduate Program in Computer Science, Informatics Institute
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS, Brazil
{hcfreitas, flmadruga, mazalves, navaux}@inf.ufrgs.br

***Abstract*—Thread level parallelism and multi-core processors are current alternatives to increase performance of general-purpose applications. In the same way, Networks-on-Chip (NoCs) are the main alternatives for supporting packet throughput for the next generations of many-core processors. NPoC (Network Processor on Chip) is a proposal to increase the performance of programmable NoC routers and multi-cluster NoC architectures using Interleaved Multithreading (IMT) technique. Therefore, the main goal of this paper is to present the design impact of interleaved multithreading for Network Processors on Chip focusing on area and performance feasibility. Results show that NPoC-based router has an acceptable and similar area relative to a conventional NoC, and higher performance up to 7.1% than the same NPoC version without IMT.**

I. INTRODUCTION

In accordance with high-performance computing, the next generation of general-purpose processors focuses on many processing cores [1] and Thread Level Parallelism (TLP) [2]. Due to the native parallel infrastructure, a large number of cores can increase the data throughput running different threads from different applications. Using TLP and ILP (Instruction Level Parallelism) [2] a processing core can explore parallelism through two levels. However, superscalar pipelines increase occupied area considerably, and for several workloads, many-core processors using scalar pipelines and TLP can obtain a better performance and area occupation.

Figure 1 shows examples of hardware multithreading in order to support TLP technique. The main TLP class is called Explicit Multithreading [2] and it is divided into the following classes: IMT (Interleaved Multithreading), BMT (Blocked Multithreading) and SMT (Simultaneous Multithreading). These three classes are presented in Figure 1 mapped onto scalar or superscalar pipelines and Chip Multiprocessor (CMP). It is important to notice that, through superscalar pipeline and SMT technique, the hardware is capable of executing threads simultaneously, but the cost is very high and there are several problems related to shared resources. For this reason, through interleaving instructions (IMT) or blocks (BMT) from different threads, the hardware achieves an optimized utilization and a good data throughput. Moreover, if the workload has a low instruction parallelism, the cost of a superscalar pipeline is very high. Therefore, IMT supports high performance and low cost based on the following characteristics: fine-grain multithreading parallelism, low pipeline stalls, high thread throughput, and low area impact.

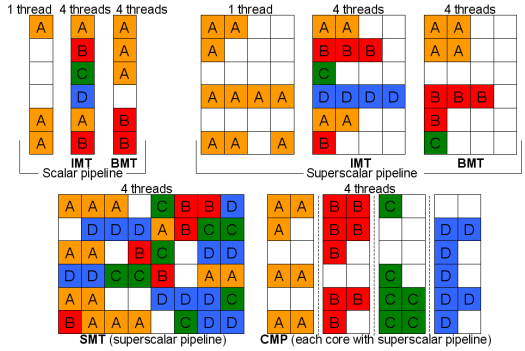

Figure 1. Examples of Hardware Multithreading

Due to the large number of cores in the next generation of many-core processors, Networks-on-Chips (NoCs) [3] are alternatives to reduce the wire influence. For instance, large shared buses or crossbar switches increase the access latency due to the high wire resistance, besides problems related to wire routing. Our previous work [4] describes a NoC for multi-cluster chips in a context of many-core processors. A Multi-Cluster NoC can reduce the latency and packet contention related to network traffic exploring the locality of cores in the same cluster to achieve a high speedup. In the example presented in Figure 2, different domains of applications run onto different or shared clusters supporting distributed processing and increasing application throughput and performance.

However, different from conventional NoCs that have dedicated routers, the Multi-Cluster NoC has programmable routers [5] in order to add flexibility and performance,

This work is supported by CNPq (National Council for Scientific and Technological Development).

[†] On leave from Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Belo Horizonte, MG, Brazil, cota@pucminas.br.

adapting topologies dynamically, according to specific communication patterns. Each router presented in Figure 2 has a processor called NPoC (Network Processor on Chip). Figure 3 shows the programmable router architecture that also consists of input buffers and a Reconfigurable Crossbar Switch (RCS).

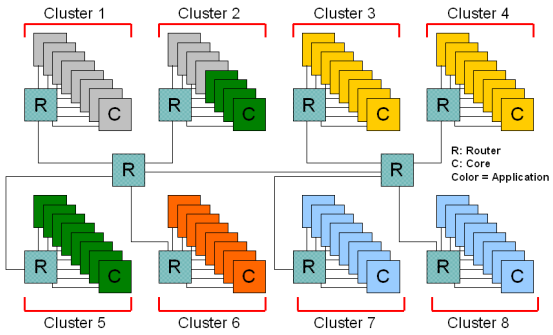

Figure 2. Multi-Cluster Architecture

The main problem of Network Processors is related to packet and program throughput. Therefore, it is important to achieve network performance executing different types of programs. According to this problem, the goal of this paper is to present the design impact of interleaved multithreading for network processors on chip. Our results show that IMT technique on NPoCs increases the area due to the number of register banks. On the other hand, the program throughput increases considerably, reducing the processing time up to 7.1%.

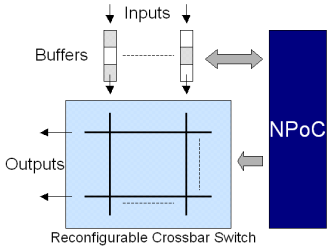

Figure 3. Programmable NoC Router

The state-of-the-art presents multithreading design on Network Processors [6], but for NoC router designs there is no related work, since there are no programmable routers. However, IMT technique has increased the performance of commercial Network Processors focusing on packet throughput. Our previous work [5] shows the design of a programmable router without IMT, but considering the importance of this technique, it is necessary a design impact evaluation. Therefore, the main contribution of this paper is related to NPoC feasibility results focusing on advantages of Multi-Cluster NoC architectures.

Next section shows the architecture and design decisions of Network Processor on Chip with IMT technique.

II. NPoC Architecture

NPoC was designed to increase flexibility and performance of NoC router architectures. Figure 4 shows the main details of the first scalar architecture [7] without IMT technique. NPoC has a pipeline based on five stages, general-purpose and network instructions. The fourth stage responsible for accessing memory in common architectures is also responsible for accessing input buffers and the reconfigurable crossbar switch (RCS) in order to perform scheduling and topology reconfiguration.

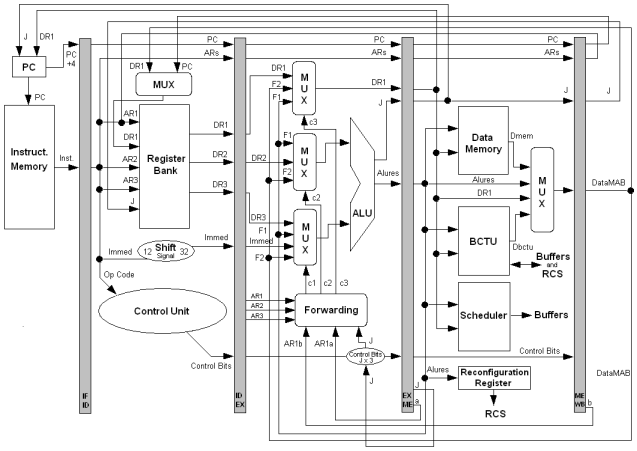

Figure 4. NPoC Architecture [7]

Network instructions are used to handle the traffic and communication from processing cores or clusters of cores regarding physical topology shown in Figure 2. These instructions, through fourth pipeline stage, access the crossbar switch to implement topologies in accordance with communication patterns based on traffic monitoring [4]. NPoC and RCS work together to perform the packet processing according to fast-path and slow-path approaches. Packets that do not need processing are forwarded through RCS without NPoC influence (fast-path). On the other hand, packets that need processing demand NPoC operations (slow-path).

In order to optimize NPoC architecture increasing packet and program throughput, IMT-based hardware was designed on the original scalar pipeline to support four threads. The main IMT characteristics are active thread contexts and reduction of memory latency effects from context switching. In common architectures without multithreading support, there is a high latency to switch thread context between register bank and memory. In multithreaded architectures, thread contexts are supported by a set of register banks and program counters. For instance, four supported threads need four register banks and four program counters. This is the basic proposal to modify the first NPoC design.

Figure 5 shows this main modification in order to reduce memory access latency to perform the context switching. In this case, there is no latency overhead since all contexts are present in register banks. In each cycle the processor uses a register bank from a different thread, without accessing the memory.

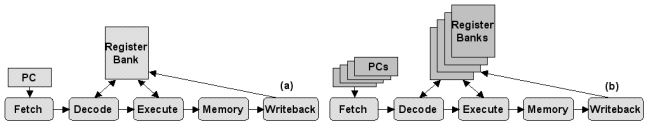

Figure 5. Scalar Pipeline Stages: (a) without IMT, (b) with IMT

Figure 6 shows the interleaved instruction stream based on NPoC scalar pipeline. It is important to notice that for each stage there is a different instruction from a different thread. This behavior is responsible for reducing data and control dependencies and, consequently, the hardware complexity to solve these problems. Other advantage is related to high-latency events. In this case, if thread 3 (T_3) is waiting data from memory or network (high-latency example) other thread is fetched (for instance T_2) reducing the number of stalls and penalties on performance. Consequently, this behavior increases the thread throughput improving the NoC router performance.

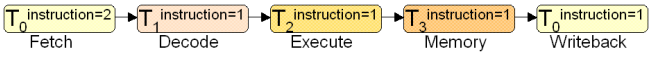

Figure 6. Interleaved Instruction Stream

According to fast-path approach, NPoC is not intrusive during a packet transmission. However, NPoC decisions based on network algorithms can delay or accelerate the initial communication. The evaluation of network algorithms is not the goal of this paper, but the hardware multithreading based on IMT can influence on the performance of multiple threads. If a thread is responsible for handling the traffic and for implementing topologies onto crossbar switch, so, it can increase the initial latency of a communication. The main reason is illustrated in Figure 6, since there are other three threads running and sharing the same scalar pipeline. Therefore, in terms of performance, there is a speedup of throughput, but maybe (it depends on scheduling rules), an increase of execution time for a single thread.

However, the results of this paper show that the initial latency on packet forwarding through interleaved multithreading is very low. Moreover, IMT technique has advantages, since it is possible to execute other threads to perform or solve other problems related to the network traffic reducing switching overhead. Next section presents the evaluation methodology, the low impact on area and the high performance.

III. Evaluation Results

A. Evaluation Methodology

An NPoC version described on VHDL (VHSIC Hardware Description Language) was used to evaluate the area impact. This version is based on open-source code called Plasma [8]. NPoC version has the following features: five pipeline stages, and reduced number of instructions and registers. This reduction is important since there are NoC router constraints and this new design has to be similar to original design [7].

The area impact was evaluated through Xilinx ISE tools mapping on a 2vp100-6ff1704 device from the Virtex II Pro family. FPGA (Field Programmable Gate Array) occupation results were used to compare prototype versions without and with IMT. In this way, the first NPoC architecture used to design the programmable router [5] was modified to support two and four threads. Hence, it is important to check the feasibility results from our previous work and from a conventional NoC architecture called SoCIN (System-on-Chip Interconnection Network) [9] based on mesh and torus topologies.

The performance evaluation is based on the following four programs: i) Two particular and specific NPoC programs for monitoring the communication pattern called ComPat and for implementing demanded topologies called TopPat. ii) A conventional general-purpose algorithm to calculate factorial. iii) From MiBench [10], Dijkstra is a typical algorithm to calculate the shortest path between every pair of NoC routers. It is important to notice, that the performance evaluation does not focus on these algorithms, but on IMT support and its advantages and impact running these algorithms.

Our previous work [4] shows that input buffers are responsible for higher dynamic power consumption than instructions running on NPoC. The static power is practically the same for different NPoC sizes and dynamic power is not influenced by instructions. Hence, this paper references the work [4] to understanding the impact on power consumption for NPoC-based routers prototyped on FPGAs.

B. Area Impact

Reduction of area has been very important during the NPoC design. However, the IMT technique increases the number of components used for program counters and register banks. Hence, the increase of area must achieve an acceptable and similar result according to conventional NoCs.

Table I shows the results of NPoC with IMT based on two threads (2-IMT) and four threads (4-IMT) comparing with NPoC version without IMT. The number of Flip-Flops (FFs) increased up to 55% and 195% for 2-IMT and 4-IMT respectively. The number of Look-up-Tables (LUTs) increased up to 6.3% and 54.2% for 2-IMT and 4-IMT respectively. The NPoC-based router has the same behavior, but with lower rates comparing with router version without IMT. The number of Flip-Flops (FFs) increased up to 10.5% and 36.6% for 2-IMT and 4-IMT respectively. The number of Look-up-Tables (LUTs) increased up to 11.5% and 34.4% for 2-IMT and 4-IMT respectively.

TABLE I. FPGA Occupation

Prototypes		*Without IMT*	*With 2-IMT*	*With 4-IMT*
NPoC	FFs	733	1136	2167
	4-LUTs	2174	2311	3357
Router 8x8 crossbar	FFs	3935	4354	5375
	4-LUTs	4555	5081	6121
Conventional NoC		*2x4 mesh*	*2x4 torus*	
SoCIN	FFs	3425	3948	
	4-LUTs	5464	6486	

However, the main result is the comparison with NPoC-based router for a cluster of eight cores and SoCIN (conventional NoC) that supports the same number of cores. The first comparison shows 2-IMT feasibility. Even though the higher number of Flip-Flops up to 27% and 10%, the number of LUTs is lower than respective SoCIN versions (mesh and torus) up to 7% and 22%. The 4-IMT feasibility is checked by following results: the number of Flip-Flops increased up to 57% and 36% comparing with SoCIN versions (mesh and torus), and the number of LUTs increased up to 12% comparing with mesh version and reduced up to 5.7% comparing with torus version. In terms of area, a programmable router composed of IMT-based NPoC is feasible due to the similar number of components shown by conventional NoC.

C. *Performance Impact*

Table II shows the performance results based on the following metrics: CPI (Cycles per Instruction), ACPI (Absolute CPI: focus on one thread), RCPI (Relative CPI: influence from other threads), Executed Cycles and Processing Time. Instruction streams (threads) from each program were evaluated in order to check the IMT impact. Due to the pipelined architecture, the average CPI is 1.10 for executed threads without IMT support. ComPat and TopPat have low number of executed cycles, since the complexity to handle the input buffers and crossbar switch is very low. Factorial thread represents a typical general-purpose algorithm supported by general-purpose instructions. Dijkstra is more complex and the processing time is higher than other threads. Its complexity is typical for network programs that manage whole topologies. With IMT support the RCPI increases due to the instruction switching while the ACPI has an ideal behavior (one instruction per cycle without stalls). In addition, it is important to notice two behaviors: i) the overhead on packet forwarding is low and ii) the thread throughput increases a lot.

TABLE II. PERFORMANCE RESULTS (50 MHZ)

Programs	*Without IMT*			*With 4-IMT*			
	CPI	Cycles	Time	RCPI	ACPI	Cycles	Time
ComPat	1.03	27	0.54µs	4	1	104	2.08µs
TopPat	1.20	6	0.12µs	4	1	20	0.40µs
Factorial	1.18	259	5.18µs	4	1	876	17.50µs
Dijkstra	1.00	59355000	1.18710s	4	1	237412024	4.74824s
Dijkstra*				1	1	59356000	1.18712s

Through the programmable router (focus on fast-path), the total transmission time to send one thousand 4096-bit packets (broadcast pattern) is 2.56ms (50 MHz). ComPat program is responsible for identifying the final transmission and a new pattern. Its impact on transmission time without or with IMT is unexpressive, as follows: 0.021% and 0.081% respectively. Considering that all threads are re-executed after their last instruction, the influence of each one increases the processing time for a single thread as shown in Table II (with IMT). However, scheduling rules allow instruction fetch for the same thread, since other threads are finished. Dijkstra* that has the large number of executed cycles illustrates this behavior. Its number of cycles increases due to the influence of other threads, but this influence is very small, since these threads are finished. Therefore, the processing time and RCPI are practically the same without IMT support. Moreover, the throughput increases; all threads can run before the end of other threads improving the router global performance regarding Figure 7 illustrates. This figure does not have a proportional scale for a better visualization, but it has the correct behavior and total processing time.

Due to the IMT, register banks support more than one context (4 threads), and for this reason, there is no penalty to switch context (42 cycles). Figure 7 shows that the total processing time is lower than a network processor without IMT. For a large thread running, such as Dijkstra, this impact is practically inexistent. However, if thread 3 (T3) is another stream from a Factorial program, the impact on total processing time is high. Hence, considering that NPoC runs small threads more than large threads, the use of IMT reduce the total processing time, regarding our results, up to 7.1%. Moreover, scheduling rules, and cache misses can increase the thread switching on NPoC without IMT. In these conditions, the speedup with IMT increases considerably.

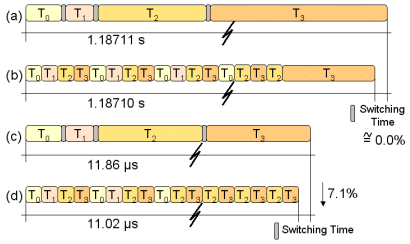

Figure 7. Processing Time: (a) without IMT (large thread), (b) with IMT (large thread), (c) without IMT (small thread), (d) with IMT (small thread)

IV. CONCLUSIONS

Many processing cores demand NoC architectures and new design approaches to support an intensive communication. Through interleaved multithreading design on a Network Processor on Chip, this paper shows the importance on increase of flexibility, throughput and performance of programmable NoC routers. The impact on area shows the feasibility, due to the similar number of components shown by conventional NoC. Focusing on performance, the throughput increases since there are more active threads, resulting in a reduction of processing time up to 7.1%, due to the reduction of context switching effects (data from register bank to memory and vice-versa).

Future work will focus on thread mapping on the proposed multi-cluster NoC and its management rules for NPoC.

REFERENCES

[1] K. Olukotun, and L. Hammond, "The Future of Microprocessors", ACM Queue, Vol. 3, Issue 7, pp. 26-29, September 2005.

[2] T. Ungerer, B. Robic and J. Silc, "A Survey of Processors with Explicit Multithreading", ACM Computing Surveys, Volume 35, Issue 1, pp. 29-63, March 2003.

[3] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip", ACM Computing Surveys, Vol. 38, No 1, pp. 1-51, March 2006.

[4] H. C. Freitas, T. G. S. Santos and P. O. A. Navaux, "NoC Architecture Design for Multi-Cluster Chips", IEEE International Conference on Field Programmable Logic and Applications, Heidelberg, Germany, pp. 53-58, 2008.

[5] H. C. Freitas., T. G. S. Santos and P. O. A. Navaux, "Design of Programmable NoC Router Architecture on FPGA for Multi-Cluster NoCs", IET Electronics Letters, Vol. 44, Issue 16, pp. 969-971, 31st July 2008.

[6] Comer, D. E., "Network Systems Design Using Network Processors", Prentice Hall, 2003.

[7] H. C. Freitas and P. O. A. Navaux, "The First NPoC Design", Workshop de Processamento Paralelo e Distribuído, Porto Alegre, RS, Brazil, 4p., 2008.

[8] S. Rhoads, "Plasma Processor", http://www. opencores.org/, 11/21/2007.

[9] C. A. Zeferino and A. A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip", IEEE Symposium on Integrated Circuits and Systems Design, São Paulo, SP, Brazil, pp. 169-174, 2003.

[10] M. R. Guthaus, et al., "MiBench: A free, commercially representative embedded benchmark suite", IEEE 4th Annual Workshop on Workload Characterization, Austin, TX, USA, pp. 3-14 , 2001.