A METHODOLOGY FOR PRECISE COMPARISONS OF PROCESSOR CORE ARCHITECTURES FOR HOMOGENEOUS MANY-CORE DSP PLATFORMS

B. Rousseau, Ph. Manet, I. Loisel, J.-D. Legat

Université catholique de Louvain (UCL)
Laboratoire de microélectronique (DICE)
Place du Levant, 3
B-1348, Louvain-la-Neuve, Belgium

H. Vandierendonck

Ghent University
Dept. ELIS/HiPEAC
St.-Pietersnieuwstraat, 41
B-9000 Gent, Belgium

ABSTRACT

The power efficiency of an HMCP heavily depends on the architecture of its processor cores. It is thus very important to choose it carefully. When comparing processing architectures for their use in a many-core platform, one must evaluate its IPC, but also its power and area. Precise power and area evaluations can only be done with real implementations. However, comparing processor implementations is a difficult task since the implementation specificities introduce interferences on the performances. This paper proposes a methodology that allows to realize precise comparisons of performance for different processor architectures. Using this methodology, it is possible to choose the best architecture for an HMCP targeting DSP applications. The methodology is based on the use of a common architectural template to build the cores, and on the application of specific optimizations when relevant. In order to validate the methodology, three RISC cores are implemented: a single-issue core, and two VLIW processors with respectively 3 and 5 issues. The implemented cores are precisely compared on a set of DSP kernels.

Index Terms— homogeneous many-core, signal processing, processor architecture, power efficiency

1. INTRODUCTION

Homogeneous many-core platforms (HMCP) are used for DSP applications. At present, those platforms use up to several hundred processor cores [1, 2]. Those cores are typically RISC architectures, having single or multiple issues like VLIW processors. Thanks to their very high parallelism, they can reach very high throughputs. They also have a very high programmability level, and a good compilation support [3] compared to heterogeneous platforms like SIMD accelerators [4]. HMCPs targeting DSP applications must have a very high power efficiency since DSP applications have a very limited power budget. The architecture of the cores composing the platform has a strong influence on the platform efficiency, it should thus be chosen carefully.

On many-core platforms, to get more performances, one can use more cores. However, adding more cores increases the platform power and area, and the amount of increase depends on the power and area of the cores. Different cores will lead to different platform configurations and performances. For instance, using simple cores will provide a low IPC, but their low area and power consumption allow to put many of them on an HMCP with a given power and area budget. On the contrary, using more complex cores will provide better IPC, but will also require more area and power [5]. In this case, less cores can be used with the same budget. As those examples illustrate, there is a strong interaction between the performances of an HMCP and the IPC, power and area of its cores.

In order to choose the best architecture for the cores of an HMCP, besides the IPC, it is also required to compare the power and area of the candidates. To evaluate the IPC of a core, one can use a simulator, but to evaluate the power and area, it is necessary to use real implementations, like an IP or even a chip, to get precise results. However, when comparing different processor architectures by using specific implementations, those differ on many aspects: ISA, technology, process flavor, hardware optimizations or compilation optimizations. Each of those aspects has an influence on the core performances. In order to isolate the impact of the core architecture on the platform performances, it is necessary to reduce the interferences introduced by a specific architecture implementation.

To enable precise and fair performance comparisons at the architectural level, this work proposes a methodology that strongly reduces the variations introduced by the specific implementations of the cores. The methodology is based on the use of a common architectural template to build implementations of the compared cores, and on the application of specific optimizations on them when relevant. Using a template guarantees uniform implementations between the different architectures and provides shared generic implementations for the functionalities of a core. However, those generic implementations could be a disadvantage for some specific architectures

compared to others. For instance, a VLIW processor with many issues has a huge RF, which is a disadvantage compared to a single-issue RISC processor using a generic register file implementation [6]. For most of those drawbacks, numerous contributions have already been proposed to mitigate their bottlenecks. In order to realize a fair comparison, it is therefore necessary to implement them. The proposed methodology suggests therefore to use a common architectural template together with specific optimizations when they are relevant, depending on their impacts on the overall performances that are speed, power and area.

The methodology is validated by the implementation of three complete processor cores that can be used in an HMCP: a single-issue scalar RISC core, and two VLIW processors with respectively 3 and 5 issues. Their implementations have been realized on the basis of a common architectural template, and each core has received specific optimizations. They have been implemented using a standard cell library of a low power SVT CMOS 65nm technology from STMicroelectronics. Their performances have been evaluated using 6 DSP kernels coming from multimedia and communication applications that are representative of the application domain. Additionally to the validation of the methodology, this paper also gives precise results for the comparison of the three cores.

This paper is organized as follows: the next section presents existing many-core architectures and several works in the domain. The proposed methodology is described in the section 3. It discusses the criterions allowing to build comparable processor cores and realize a fair comparison of their performances. Section 4 and 5 describes the concept of architectural templates and discuss the need to apply specific optimizations on the compared cores. Three cores are implemented to validate the methodology. Their compositions and implementations are presented in the sections 6 and 7. Section 8 presents results validating the methodology. The optimizations applied to the compared cores are described, and the results illustrating their impact on the core performances are discussed. Finally, section 9 presents and compares the performances of the implemented cores. The last section concludes the paper.

2. RELATED WORK

There are numerous existing HMCPs, some of which are also called massively parallel processor arrays (MPPA). However, there is no work that tries to justify which processor core architecture is the best for those platforms. PicoArray from picoChip [1] uses more than 300 3-issue VLIW cores with a 16-bit datapath. The Tile64 platform from Tilera, based on the RAW research platform [7], has 64 32-bit 3-issue VLIW cores. Their larger platform, the Tile-Gx, uses 100 cores. Ambric platforms [2] have 336 32-bit single-issue RISC DSP cores. Among research platforms, AsAP2 is also an HMCP, with 167 single-issue cores [8]. The many-core WPPA plat-

form [9] has a configurable number of small VLIW cores.

Several works propose design space exploration frameworks for multi-core platforms. Those frameworks help to quickly identify several platform configurations which are potential solutions for a group of applications [10, 11]. They are generally based on fast simulators and performance models of processor architectures. This approach speeds up the exploration but reduces the precision of the results. The calibration of those models is performed with a limited set of real implementations, like provided in this work.

Other works propose architectural description languages (ADL) that allow the high-level description of a processor architecture. On the basis of this description, the associated tools can automatically generate a simulator, a toolchain and RTL code [12, 13, 14]. The use of ADLs allows to easily evaluate the performances of several architectural solutions, by describing the different architectures and simulating the applications using the generated tools. Nevertheless, the performances obtained with the automatic optimizations performed by the ADL tools are limited [15], which is a disadvantage for some architectures, like VLIWs for example.

3. METHODOLOGY FOR ARCHITECTURAL COMPARISON

Precise processor comparisons are performed using implementations, by comparing chips, IPs, or results from datasheets. Those implementations have different ISAs, technology nodes, process flavors, hardware optimizations or software compilation optimizations. Those implementation specificities introduce interferences on the performances of compared cores. These interferences are caused by different sources of variations, they are listed in Table 1. Those variations make it very difficult to evaluate the influence of a single aspect, like the core architecture, on the performances of the compared implementations. To build processor cores comparable at the architectural level, those variations must be removed.

When comparing processor architectures, it is important to make sure that the architecture comparisons are fair. To evaluate them correctly, each core must be able to fully exploit the benefits of their architecture. It is thus important to guarantee that the processor implementations and the code they execute are fair regarding the evaluations of architectural features.

In order to build comparable processors and to realize fair comparisons, the methodology proposed in this work consists in complying with the following criterions:

1. **the cores are implemented in the same technology:** it allows the cores to benefit from the same timing and power performances and to operate in the same conditions.

Table 1. List of variations introducing interferences between processor implementations

Source of variations	Causes
Physical implementation	Technology, process flavor, development flow, supply voltage.
Microarchitecture	Core internal organization, function implementations, available functional units, ISA, memory blocks.
Software	Code scheduling, benchmarks.

2. **they are implemented using the same development flow:** the designs must be synthesized, placed and routed with the same tools and the same constraints. Thanks to this, they take benefit of the same automatic optimizations.
3. **they use the same memories:** using the same memory blocks give them the same performances. To do so, memory netlists can be generated with the same memory compiler.
4. **they use the same ISA:** the ISA has an influence on the complexity of the decoding circuits, on kernel sizes, and on the instruction memory access count. The compared processors have access to the same DSP instructions to optimize kernel execution.
5. **they use a maximum of resources defined with the same code:** identical functional blocks must be defined with the same HDL code or the same placed and routed netlists (e.g.: ALUs, instruction decoder, memory ports). This gives them the same performances.
6. **the executed code is compiled by hand:** it allows to take the most out each architecture instance, which allows in turn to evaluate their specific performances. Compilation by hand also prevents the code to depend on some specific compiler optimizations.
7. **they are well balanced:** for instance, the set of functional units of the cores must be chosen in order to maximize both ILP and resource usage. It allows a specific architecture to provide a representative amount of parallelism.
8. **specific optimizations are applied on the core when relevant:** it allows to ensure that no architecture implementations suffer from detrimental overheads which introduce biases in the evaluation of its performance. This aspect is discussed in section 5

Criteria 1-3 allow to remove the physical variations in the implementations. Criteria 4 & 5 allow to remove the vari-

Table 2. List of common and specific microarchitecture characteristics between several N-issue RISC processors

Common	Specific
Pipeline stage count & function	Execution issue count
Instruction fetcher	RF size
Instruction decoder	RF port count
Register file	Bypass input count
Bypass network	ALU count
Functional units	Memory port count

ations of the microarchitecture. To comply to those latter criteria, this paper proposes to implement the processor by using common architectural templates. Those architectural templates are explained in the next section. Criteria 6-8 allows to preserve the specificities of the architectures without bias. It also provides fair comparisons between the implementations. Criterion 6 also allows to remove software variations.

4. ARCHITECTURE TEMPLATE

The microarchitecture of a core defines its internal organization and implementation. It has a strong impact on its performances. Some elements and characteristics of the microarchitecture can be common to different architectures, some others, on the contrary, are specific. When comparing architectures, it is very important to identify common microarchitectural characteristics and impose a comparable implementation between them. This approach allows to bring the variations in microarchitectures strictly to the specific differences of an architecture. To illustrate this, Table 2 identifies some common and specific elements in the microarchitectures of a family of N-issue RISC processors. This family of processor is the one compared in this work.

In order to build comparable implementations that reduces variations of the microarchitecture, this paper proposes to use a common architectural template to build the compared architectures. Using a template guarantees to get uniform implementations for the common functionalities between the different architectures and provides shared generic implementations for the functionalities of a core. It defines a common organization for all the implementations and their evolutions with the issue count increases.

For instance, it defines:

- pipeline stage count and composition;
- the functional units available for all the compared architectures;
- the presence of specific functionalities, like the bypass network or the interlock signals preventing write hazards;

Fig. 1. Architectural template for 1-issue to N-issue RISC cores

- the evolution of the different modules when the issue count increases (e.g.: the read/write port count of the register files, the number of bypass network inputs).

Figure 1 illustrates the concept of architectural template for a family of N-issue RISC processors. In this template, multiple-issue cores correspond to VLIW processors. This template is the one used for the architectures compared in this paper.

5. MICROARCHITECTURE OPTIMIZATIONS

When the number of issue of a processor is increased, the complexity or the timing of some functional modules do not scale well. It is notably the case of the following elements:

- **the register file** presents a significant overhead when the read/write port count increases;
- **the bypass network:** the data source selector of the bypass network, as well as their control circuits do not scale well with the increase in inputs. It causes additional delays which worsen the timing of the functional units.
- **interlock signals:** those signals require long control lines across the pipeline stages. They can introduce additional delays.

For most of those drawbacks, numerous contributions have already been proposed to mitigate their bottlenecks. In order to realize a fair comparison, it is therefore necessary to implement them.

The development tools can also introduce suboptimal implementations. Some automatic optimizations that are realized by the synthesis tools do not benefit to every functional modules. For instance, automatic clock gating can cause the insertion of too many clock gating cells. This can cause an

overhead in area and a degradation of the timing in some circuits. Those degradations can eventually cause an increase in power consumption. To fix those overheads, some optimizations need to be realized manually.

To enable fair comparisons, it is thus necessary to realize specific optimizations that allows to reduce or remove the overheads caused by the scaling of the issue count and the tools. Consequently, the methodology proposed in this work suggests to identify the functional modules that can benefit from those specific optimizations, and apply them when they are relevant, depending on their impact on the performances and the required precision. In the same way, automatic optimizations realized by the tools must be monitored, and be replaced by more efficient manual optimizations when necessary.

6. DESIGNED PROCESSOR CORES

Three processor cores have been implemented by following the methodology proposed in this work. Their implementations have been realized using the template presented in Figure 1. Each core uses an identical standard DSP instruction set. The first architecture is a scalar single-issue RISC processor, called *DSP1*. The two other implemented architectures are VLIW processors, with respectively 3 and 5 issues, called *VLIW3* and *VLIW5*. These cores cover the range of candidate architectures for HMCPs.

The three architectures use the same datapath components:

1. **ALU_{INT}:** 32-bit integer computation unit. This unit performs basic arithmetic operations (e.g.: addition, subtractions), logical operations and comparisons. It supports bit-level operations, like byte swapping or bit rotations. Those operations are performed in one cycle.
2. **ALU_{SIMD}:** SIMD operation unit. It performs 2×16 -bit and 4×8 -bit operations like absolute differences, scalar products, etc. Those operations are performed in one or two cycles.
3. **MAC:** multiplication-accumulation unit, which can perform a double 16-bit multiplication in two cycles, and a multiplication-accumulation in three cycles. This unit allows to implement efficiently filtering operations that are numerous in telecommunication applications.

The composition of the three processors, with the description of their units, the size of their register files, their bypass networks and their memory ports is summarized in Table 3. The set of functional units selected for each core instance has been chosen to balance them correctly, as explained in section 3. The number of memory ports and the selected ALUs allow to maximize the use of the available resources while also maximizing the ILP.

Table 3. Composition of the implemented processor cores

<i>DSP1</i>	<i>VLIW3</i>	<i>VLIW5</i>
1 issue	3 issues	5 issues
32×32-bit reg.	64×32-bit reg.	64×32-bit reg.
1×ALU _{INT}	3×ALU _{INT}	5×ALU _{INT}
1×ALU _{SIMD}	2×ALU _{SIMD}	3×ALU _{SIMD}
1×MAC	1×MAC	2×MAC
1 memory port	2 memory ports	2 memory ports
Simple bypass	7-input bypass	11-input bypass

Table 4. Comparison of dynamic and leakage power for the three implemented cores on the 802.11a benchmark.

	<i>DSP1</i>	<i>VLIW3</i>	<i>VLIW5</i>
Dynamic power at 100MHz	2.69e-3	8.72e-3	15.40e-3
Leakage power	6.58e-6	7.51e-6	20.94e-6

The three processor cores are accompanied by instruction memories and scratchpads for their data. The *DSP1* processor has a 4KB instruction memory, and a 8KB scratchpad for its data. The VLIW processors have larger instruction memories since their codes are bigger due to the use of unrolling and software pipelining techniques. Those memories have been precisely dimensioned following their code growths. The growth factors have been estimated by comparing the size of the benchmark codes of the VLIW cores with the codes of the *DSP1* core. For the *VLIW3*, this growth factor is 2×, and 2.7× for the *VLIW5*. Both VLIW processors have the same total amount of data memory as the *DSP1*, distributed in two 4KB scratchpads. The cores have all the same amount of memory for their data since the size of those memories is dictated by the size of the data elements processed by the algorithms. They cannot be modified for the different architectures.

7. PROCESSOR CORE IMPLEMENTATIONS

The three processors have been coded using verilog HDL, then synthesized, placed and routed using digital standard cell libraries of a low power SVT CMOS 65nm technology from STMicroelectronics. The memory blocks have been obtained using memory compilers from this technology. This process flavor has a high threshold voltage, it allows to strongly reduce the leakage of the transistors. Because of this, the power consumption of the cores is dominated by their dynamic power, there is no bias from the leakage power. Table 4 compares the dynamic and leakage power of the three implemented cores. In those designs, the leakage power is two orders of magnitude lower than the dynamic power. However, the high threshold voltage also limits the transistor speed, which limits the working frequency of the designs realized in this technology.

Figures 2 and 3 respectively show the evolution of the

Fig. 2. Areas of the processor cores for each frequency constraint applied during place and route. White markers represent the selected netlists for each architecture used in the following experiments. **Fig. 3.** Energies of the processor cores on the 802.11a benchmark for each frequency constraint applied during place and route. White markers represent the selected netlists for each architecture used in the following experiments.

netlist areas and consumed energies for the three implemented processors with respect to the frequency constraint imposed to the synthesis and physical implementation tools. The evaluated energy consumption corresponds to the execution of a benchmark performing the modulation of a frame of the 802.11a wireless telecommunication standard [16].

There is a strong degradation of the area and energies around 400 and 500MHz constraints. Those degradations are due to the oversizing of the circuits in order to reach the imposed constraints. The netlists that are retained for the rest of this work are highlighted in the figures with white markers. They correspond to configurations allowing to reach the highest frequencies without having strong interferences from the constraints on the area and energy consumption. For the *DSP1*, the retained netlist is generated for a 500MHz constraint. For the *VLIW3* and *VLIW5* processors, the generated netlists are generated for a 400MHz constraint. The *DSP1* processor can thus work at a frequency which is 25% higher than the VLIW processor frequencies. The simplicity of its circuits, mainly the register file and the bypass network, allows it to reach better timing performances.

Table 5. Performance results on kernel benchmarks for the *DSP1*, *VLIW3*, and *VLIW5* platforms.

Name	<i>DSP1</i>			<i>VLIW3</i>					<i>VLIW5</i>				
	Cycles	IPC	Energy [J]	Cycles	Speedup	IPC	Energy [J]	Δ_E [%]	Cycles	Speedup	IPC	Energy [J]	Δ_E [%]
fir32	9543	0.99	8.18e-7	3398	2.81	2.69	8.39e-7	3%	2249	4.24	4.30	8.94e-7	9%
fft64	5430	0.93	3.99e-7	2133	2.55	2.58	4.45e-7	12%	1438	3.78	4.22	5.49e-7	37%
d8psk	13130	0.93	1.08e-6	5403	2.43	2.44	1.23e-6	14%	3597	3.65	3.82	1.35e-6	25%
802.11a	27760	0.95	2.27e-6	10766	2.58	2.55	2.48e-6	9%	7148	3.88	4.06	2.75e-6	21%
sad	346	1.00	2.79e-8	120	2.88	2.88	2.54e-8	-9%	76	4.55	4.59	2.68e-8	-4%
dct	770	0.93	5.87e-8	370	2.08	1.89	7.50e-8	28%	193	3.99	3.66	7.76e-8	32%
Mean	-	0.96	-	-	2.55	2.51	-	9%	-	4.02	4.11	-	20%

Fig. 4. Dynamic power breakdown for the three implemented cores. The “OTHER” category corresponds to top-level circuits, its power is dominated by the clock tree.

8. METHODOLOGY VALIDATION

The performances of the three processors have been evaluated on a set of 6 DSP benchmarks. Those benchmarks are kernels from telecommunication and image processing applications. They represent the most important workload of those applications. As explained in the section 3, the benchmarks have been optimized by hand in order to maximize the exploitation of the parallelism available in the cores, and the SIMD and DSP instructions. The reachable parallelism is then limited by the available resources as well as the dependencies between instructions. The performance results obtained for the three architectures are summarized in Table 5. One can see that the three cores reach very high IPCs on the evaluated benchmarks compared with their own issue width. This shows that following the proposed methodology allows to build well-balanced processor cores.

The power consumptions of the cores have been evaluated by extracting switching activities from post-layout netlist simulations on the benchmarks. Figure 4 shows the breakdown of the dynamic power for the modules composing the three cores. The results show that most of the power is consumed in the execution stage, this behavior confirms that the cores are well-balanced since most of the energy is actually consumed to perform useful work. An important part of the power is also consumed by the register files.

Several specific optimizations have been realized on the three implemented cores in order to reduce their disadvantages, as suggested by the methodology. Clock gating has

Fig. 5. Normalized power consumption reduction in the *VLIW5* processor due to architectural optimizations. Power figures are normalized to the power of the module in the baseline configuration.

been carefully applied on all designs with *OR* gates. Bypass controls of the VLIW processors are precomputed during the decode stage in order to reduce the critical paths of the input source selection circuits. Several optimizations have been applied on the register files in order to reduce their power consumption [17, 18]. First, registers have been partitioned in several groups and data gating cells have been placed on write port data signals for each partition. This technique allows to reduce the fanout of these signals. Second, unnecessary read and write operations in the register file are masked. The unnecessary read operations correspond to values that are provided by the bypass network to the datapaths, and unnecessary write operations correspond to register values that are replaced by new ones before they are read with the data produced in other pipeline stages.

In order to validate those optimizations, their impact has been evaluated by measuring the power consumption reduction of different optimized circuits. Figure 5 illustrates those reductions for three circuits of the *VLIW5* core, and on the whole core. In the register files, the cumulated techniques allow to reduce the energy consumption by $3\times$ compared to a naive implementation without aggressive optimizations and using automatic clock gating cell insertion. Bypass control signal precomputation allows to reduce significantly the power consumption of this stage, and of the execution (EXE) stage. This is explained by the improvements of the timing of the functional units, which allows the synthesis tools to use smaller logic gates consuming less power. After opti-

Fig. 6. Processor and memories mean powers (left) and mean power per issue (right) for each architecture.

mizations, the power consumption of the bypass stage and the execution stage is reduced by respectively 60% and 20%. These power reductions validate the need to operate optimizations manually in order to realize a fair comparison between different architectures.

9. CORE COMPARISONS

Using the results of the processing for the execution of the benchmarks, presented in Table 5, it is finally possible to compare the performances of the implemented architectures. The results show that the *VLIW3* and *VLIW5* processors can provide mean speedups of respectively 2.55 and 4.02 compared to the *DSP1* processor.

Figure 6 shows the breakdown of the total mean power consumption of the three cores and their memories. Being the simplest core, the *DSP1* dissipates the less total power. However, Figure 6 also illustrates the mean power consumption normalized by their execution issue count. One can see that the normalized power is roughly identical between all implemented cores, and that the *DSP1* is actually the core consuming the more power per issue, which do not take the IPC into account.

The energy consumed by the three cores on the benchmarks is presented in Table 5. For most benchmarks, the *DSP1* consumes the less energy. The only exception is for the *sad* benchmark where the very high speedup on the two other cores allow to compensate their higher total power. The *VLIW3* and *VLIW5* cores have mean energy consumptions which are respectively 9% and 20% higher than *DSP1* energy consumption. The loss of power efficiency of the VLIW processors is explained by the higher complexity of some of their circuits like the bypass and the register file, for which each access consumes more power when the number of inputs is higher. Moreover, in those architectures, the code is filled with more NOP instructions, which also causes a power overhead. Those results indicate that using more complex cores introduces an overhead in energy consumption. However, even if the *DSP1* has a better power efficiency, the *VLIW5* core provides a 4× speedup for an overhead of only 20%.

Figure 7 compares the areas of the processors and their memories. The total areas of the *VLIW3* and *VLIW5* pro-

Fig. 7. Processor and memories areas (left) and areas normalized per issue (right) for each architecture.

cessors with their memories are respectively $1.7\times$ and $2.4\times$ larger than the area of the *DSP1* core with its memories. Figure 7 also shows the total areas divided by the number of issues. The *DSP1* has a higher area per issue ratio since it must have all datapath units but only one instruction can be executed per cycle. Core with several issues thus make a better use of their area.

10. CONCLUSION

This paper proposes a methodology that allows to realize precise comparisons of performance for different processor architectures. Using this methodology, it is possible to choose the best architecture for an HMCP targeting DSP applications. It allows to precisely compare their IPC, power and area, which in turn enables to evaluate the performances of HMCP based on them. The methodology is based on the use of a common architectural template, together with the application of specific optimizations when relevant for the core performances. A validation of the methodology is performed through the implementation of three RISC cores: single-issue RISC core, and two VLIW processors with 3 and 5 issues. The cores are implemented in low power SVT 65nm from STMicroelectronics. This technology has a high threshold voltage, which keeps leakages at a very low level. Their performances are evaluated on 6 DSP kernels. Results shows that the methodology allows to build well-balanced cores and that optimizations can significantly impact performances. Therefore, it confirms that those optimizations are required to realize fair comparisons. Finally, comparisons are made between the three implemented cores. The results show that simpler cores have better power efficiency, but worse area efficiency. However, cores with more issues can provide high speedups with a limited power overhead: for instance, the *VLIW5* core provides a 4× speedup for an overhead of only 20% in energy. Future works will compare performances of HMCPs based on the cores implemented with the proposed methodology. Different technology and process corner will also be evaluated.

11. ACKNOWLEDGMENT

Bertrand Rousseau holds a F.R.S.-FNRS fellowship (Belgian Fund for Scientific Research). Philippe Manet and Igor Loisel are funded by the Walloon region of Belgium.

12. REFERENCES

- [1] R. Baines and D. Pulley, "The picoArray and reconfigurable baseband processing for wireless basestations," *Software Defined Radio*, 2004.
- [2] M. Butts, AM Jones, and P. Wasson, "A structural object programming model, architecture, chip and tools for reconfigurable computing," in *Field-Programmable Custom Computing Machines*, 2007. *FCCM 2007. 15th Annual IEEE Symposium on*, 2007, pp. 55–64.
- [3] A. Duller, D. Towner, G. Panesar, A. Gray, and W. Robbins, "picoArray technology: the tool's story," in *Design, Automation and Test in Europe*, 2005. *Proceedings*, March 2005, pp. 106–111 Vol. 3.
- [4] K. van Berkel, P. Meuwissen, N. Engin, and S Balakrishnan, "CVP: A programmable co vector processor for 3G mobile baseband processing," in *Proceedings of World Wireless Congress*, 2003.
- [5] MJ Flynn and P. Hung, "Microprocessor design issues: thoughts on the road ahead," *IEEE Micro*, vol. 25, no. 3, pp. 16–31, 2005.
- [6] A. Terechko, M. Garg, H. Corporaal, P. Res, and N. Eindhoven, "Evaluation of speed and area of clustered VLIW processors," in *VLSI Design, 2005. 18th International Conference on*, 2005, pp. 557–563.
- [7] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrati, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpfen, M. Frank, S. Amarasinghe, and A. Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25–35, Mar/Apr 2002.
- [8] D. N. Truong, W. H. Cheng, T. Mohsenin, Z. Yu, A. T. Jacobson, G. Landge, M. J. Meeuwsen, A. T. Tran, Z. Xiao, E. W. Work, J. W. Webb, P. Mejia, and B. M. Baas, "A 167-processor computational platform in 65 nm cmos," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.
- [9] D. Kissler, F. Hannig, A. Kupriyanov, and J. Teich, "A dynamically reconfigurable weakly programmable processor array architecture template," in *International Workshop on Reconfigurable Communication Centric System-on-Chips (ReCoSoC)*, 2006, pp. 31–37.
- [10] A. Richard, A. Vander Biest, A. Bartzas, A. Papanikolaou, D. Soudris, D. Milojevic, and F. Robert, "A Multi-Criteria Estimation Tool for System-on-Chip," .
- [11] G. Palermo, C. Silvano, and V. Zaccaria, "An efficient design space exploration methodology for multiprocessor soc architectures based on response surface methods," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2008. *SAMOS 2008. International Conference on*, 2008, pp. 150–157.
- [12] G. Goossens, D. Lanneer, W. Geurts, and J. Van Praet, "Design of ASIPs in multi-processor SoCs using the Chess/Checkers retargetable tool suite," in *System-on-Chip, 2006. International Symposium on*, 2006, pp. 1–4.
- [13] "Caware/lisatek," .
- [14] A. Halambi, P. Grun, V. Ganesh, A. Khare, N. Dutt, and A. Nicolau, "EXPRESSION: A language for architecture exploration through compiler/simulator retargetability," in *Design, Automation, and Test in Europe*. Springer, 1999, pp. 31–45.
- [15] O. Schliebusch, A. Chattopadhyay, R. Leupers, G. Ascheid, H. Meyr, M. Steinert, G. Braun, and A. Nohl, "RTL processor synthesis for architecture exploration and implementation," in *Proceedings of the conference on Design, automation and test in Europe-Volume 3*. IEEE Computer Society, 2004, p. 30156.
- [16] "802.11a-1999 high-speed physical layer in the 5 ghz band," Tech. Rep., February 1999.
- [17] H. Takamura, K. Inoue, and V. Moshnyaga, "Register File Energy Reduction by Operand Data Reuse," *Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*, pp. 289–306.
- [18] M. Müller, S. Simon, H. Gryska, A. Wortmann, and S. Buch, "Low power synthesizable register files for processor and IP cores," *Integration, the VLSI Journal*, vol. 39, no. 2, pp. 131–155, 2006.