Transactions Briefs

Interconnect Exploration for Energy Versus Performance Tradeoffs for Coarse Grained Reconfigurable Architectures

Andy Lambrechts, Praveen Raghavan, Murali Jayapala, Bingfeng Mei, Francky Catthoor, and Diederik Verkest

***Abstract*—Modern portable embedded devices require processors that can provide sufficient *performance* for demanding multimedia and wireless applications. At the same time they have to be *flexible* to support a wide range of products and extremely *energy efficient* to provide a long battery life. *Coarse Grained Reconfigurable Architectures* (CGRAs) potentially meet these constraints by providing a mix of flexible computational resources and large amounts of programmable interconnect. The *vast design space* of CGRAs complicates the development of optimized processors. Most effort has been spent on improving the performance. However, the energy cost of the programmable *interconnect* is becoming more expensive and this cost can no longer be neglected. In this work we present an energy- and performance-aware exploration for the interconnect of a CGRA and show that important tradeoffs can be made for those metrics. This will enable designers to develop more efficient architectures, tuned to a targeted application domain.**

***Index Terms*—Energy-aware design, interconnect-aware design, low power, processor architecture.**

I. Introduction

Modern consumers carry a mobile phone, GPS, PDA, digital camera, laptop, MP3 player, etc. To be able to combine all these into one, without compromising on the functionality, flexible platforms are needed that can provide a high performance with extremely high energy efficiencies. Coarse grained reconfigurable architectures (CGRAs) can provide this flexibility, while boosting performance compared to embedded VLIW processors. CGRAs consist of: 1) functional units (FUs), that operate on a word level (ALU, MUL, MAC, etc.); 2) distributed register files for intermediate data storage; 3) programmable interconnect; and 4) configuration memories. A higher parallelism and the usage of local memories lead to a high performance, while still keeping the energy consumption under control. A programmable interconnection topology provides the flexibility to map different applications efficiently to one processor. Proposed CGRAs differ in the number or organization of the resources described above.

Designing the optimal CGRA architecture involves many tradeoffs. While a large amount of research has been done to improve the performance, very little has been done to systematically explore the energy efficiency versus performance tradeoff for these processors.

Manuscript received June 11, 2007; revised October 25, 2006. First published November 18, 2008; current version published December 17, 2008. This paper is extended work of "Energy-Aware Interconnect-Exploration of Coarse Grained Reconfigurable Processors," presented at the Workshop on Application Specific Processors, September 22, 2005, NY. This work was supported in part by IWT Flanders.

The authors are with IMEC vzw and KULeuven, Leuven 3001, Belgium (e-mail: lambreca@imec.be; ragha@imec.be).

M. Jayapala and B. Mei are with IMEC vzw, Belgium, D. Verkest is with IMEC vzw, KULeuven and VUB, Belgium.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2008.2002993

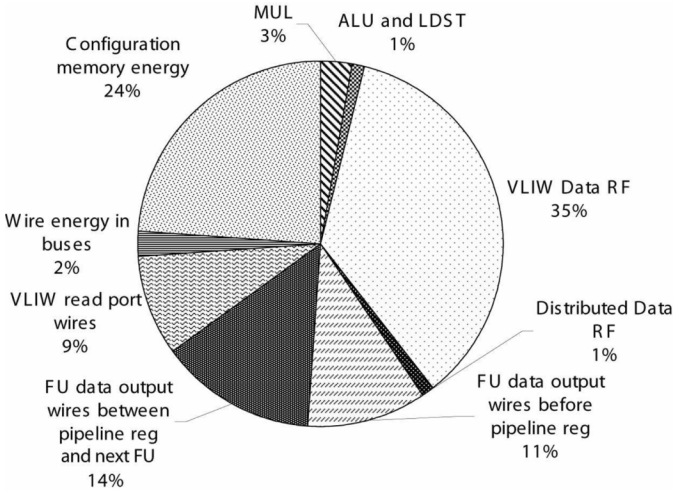

Fig. 1. Detailed energy breakdown of $4 \times 4\_b\_neg\_nh\_rf$, an instance of the ADRES CGRA, for a UMC 130-nm standard cell technology.

Using a designer-friendly architecture exploration framework that enables quick evaluation of different architecture variants, we report both detailed energy and performance estimations for a set of representative application benchmarks. The obtained performance and energy efficiency depend heavily on the provided interconnection network. Changes in the interconnect topology change how efficiently other resources can be utilized, the number of intermediate variables that have to be stored and therefore also the total number of instructions, register file accesses, configuration memory accesses, etc. At the same time, it has been shown that interconnect is getting more expensive for smaller technologies and therefore aggravating this problem as technology scales [4]. Given the importance of the interconnect it is crucial to explicitly evaluate design choices for interconnect related parameters.

Fig. 1 shows the energy breakdown of a CGRA instance with a very flexible interconnect topology (explained in detail in Section V) as is typically the result of a performance driven optimization. For a 32-tap complex FIR Filter this breakdown shows that for a 130-nm technology as much as 30% of the total processor energy is spent in the interconnect. In scaled technologies (45 nm) this interconnect energy can go up to 50%–60% (extrapolation based on ITRS roadmap for interconnect capacitance scaling). To perform an energy-aware architecture design, *it is therefore crucial to analyze the energy consumption in the interconnect*.

The schedule that can be obtained depends heavily on the flexibility of the interconnect. A limited interconnect will reduce the schedulability of applications on the CGRA processor. Since the amount of interconnect that connects different FUs inside a VLIW or RISC processor is small, the energy consumption of this communication can be neglected during the design of these processors in a first approximation. For CGRA designs this is not true, because of the large amount of communication resources between different FUs and registers files. As a result, the design of the interconnect has a *significant effect on both energy efficiency and performance* and has to be taken into account during architecture exploration.

In this paper, we vary the interconnect topology and use a designer-friendly framework to generate detailed energy and performance estimates. Our closed loop simulation and estimation flow supports the exploration of interconnect topologies in combination with other parameters, e.g., the size of the array, or size of distributed register files, in order to consider the interaction between these parameters. However, this interaction and the exploration of the other parameters are outside the scope of this paper. We do include the direct effect of the change in interconnection on both energy and performance, and the indirect effect on energy consumption as a result of changes in the schedule.

This paper is organized as follows. Section II discusses related work, while Section III briefly introduces the CGRA template and the compilation and simulation tool-chain. Section IV presents our extended simulation framework and the used energy models. Section V discusses the performed interconnect topology architecture exploration. Section VI presents detailed exploration results and an analysis of the experimental data. Finally, Section VII concludes this paper.

II. Related Work

Recently, a large number of CGRA-based processors have been proposed: ADRES [9], Montium [11], Morphosys [18], Pleiades [16], RaPiD [5], SiliconHive [14], TRIPS [3], etc. Some architectures like [3] are using coarser granularity processing elements and are targeted toward high performance systems. A growing number of CGRAs [9], [11], [14] are explicitly targeted at low power embedded systems.

Since most CGRAs have a relatively fixed architecture or no retargetable compiler, little work has been done in architectural exploration in this domain. Others do have a flexible template, but due to the large number of architectural parameters, the exploration space is very large and no systematic exploration of the complete space has been published. However, Wilton *et al.* [6] explored the various configurations and sizes of the register files, but they looked only at performance. Bansal *et al.* [1] investigated the impact of different network topologies for mesh-based CGRAs, but their template is too restricted and they also looked only at performance. Others, like Silicon Hive [14], could support architecture exploration, but have not integrated fast and interconnect aware energy estimation into their framework, which is one of our main contributions of this paper.

III. CGRA Template Architecture Description

CGRAs offer a large architectural space of exploration, changing any of four categories of parameters: computational resources, data storage, interconnection resources and configuration storage. However no systematic energy-aware and interconnect-aware architecture exploration is available.

The presented energy- and interconnect-aware architecture exploration is built on the ADRES [9] framework. ADRES is a flexible template that includes a tightly coupled VLIW processor and a CGRA. Many other proposed CGRA architectures can be represented by this parameterizable template. The ADRES template consists of a number of FUs (ADD, MUL, etc.) that operate on data words, in the form of an array. The array can be configured with different array sizes (e.g., 4×4, 4×6, 8×8) and different types of FUs (homogeneous or heterogeneous). The template also supports distributed storage of data (registers and register files) and of configuration memories that can be shared by more than one FU. Examples of ADRES instances are shown in Fig. 3. For more details on the ADRES architecture template the reader is referred to [9].

The ADRES architecture template is developed in conjunction with a retargetable simulator and compiler, called DRESC [8]. The ADRES-DRESC framework allows a designer to vary many of the critical design parameters of a CGRA and quickly evaluate the effect on performance.

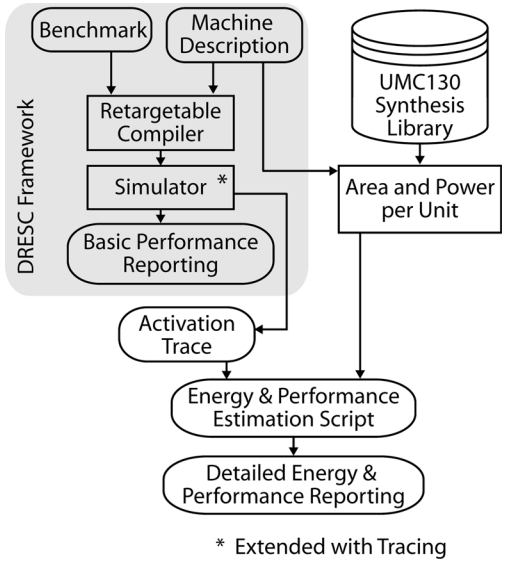

Fig. 2. Extended energy-aware architecture exploration framework.

IV. Power Modeling and Estimation

To enable a meaningful architecture exploration for a CGRA, the energy consumption of all components has to be taken into account. A design targeted at boosting performance only, will lead to an over-design and a suboptimal point on the energy versus performance tradeoff curve.

To support detailed energy estimations, the original ADRES-DRESC framework was extended with additional components to generate detailed performance results and to include a detailed energy breakdown (see Fig. 2). The components of the ADRES template were synthesized for a UMC 130-nm standard cell library, back-annotated with switching activity and energy consumption per activation and area were computed. The area estimates for the components are used later for interconnect length estimation.

The energy consumption of register files is calculated using the EMPIRE model for register files [15], which gives accurate energy estimate for various instances and is calibrated using the same technology libraries and estimation flow. The configuration memories are modeled using a 130 nm, 1.2 V SRAM memory macro taken from [2].

Accurate area estimations of different components are used to construct a high level floor-plan of the ADRES architecture and wire lengths of each individual wire are computed in an automated way. The wire length model for buses and point-to-point interconnections are taken from the same technology to ensure consistency. To get fast, but fairly accurate estimations on the interconnect energy, without going to a full placement and routing for each architecture instance, interconnect lengths between the different functional units and connections to register files are computed as the Manhattan distance between these architectural components. Note that this leads to an optimistic estimate for interconnect length, leading to a lower bound on the interconnect energy estimation.

The energy estimation is completed by extending the simulator to keep track of the activation of every component of the architecture. A detailed activation trace is processed and the total energy consumption is computed, per component or per category (e.g., distributed register files or only a certain type of wires) and detailed reports are generated.

V. Energy-Aware Architecture Exploration

Different types of interconnections can be added to the ADRES architecture, both regular and irregular, as the template allows a designer

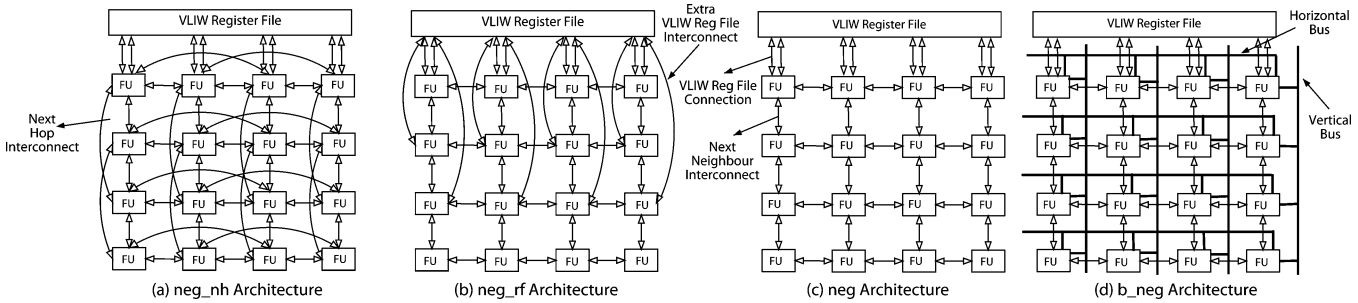

Fig. 3. Examples of different interconnection topologies for the ADRES CGRA: (a) nearest neighbor and next hop connections; (b) nearest neighbor connections and extra connections to the central register file; (c) nearest neighbor and next hop connections only; (d) nearest neighbor connections and horizontal and vertical buses. (a) neg_nh architecture; (b) neg_nf architecture; (c) neg architecture; (d) b_neg architecture.

to add any point to point connection. In this work, four different types of connections that are commonly found in state of the art architectures are combined in order to vary the flexibility of the resulting interconnect topology from very flexible to restricted. We focus on fairly regular types, because this tends to lead to more compiler-friendly architectures and keeps the problem tractable. Fig. 3 shows four different architectures, providing four different possible instances of the interconnect topology.

The following naming conventions are used to denote the different configurations of the CGRA architecture template parameters that are used:

1) *size of the array*: 4×4 or 8×8 FUs;
2) *buses* (horizontal and vertical): *b*: With buses, connecting all FUs in the same row/column;
3) *interconnect to nearest neighbors—_neg*: connection with nearest neighbors (from the output of an FU to the input of all next neighbors, in both horizontal and vertical direction, but no diagonal connections);
4) *interconnect to next hop neighbors—_nh*: connection with neighboring FUs one hop away, both in horizontal and vertical directions, but no diagonal connections;
5) *interconnect to VLIW register file—_rf*: extra connections from the CGRA to VLIW register file, to facilitate distribution of live-in/live-out variables, to the second and third rows of FUs;

e.g., 4×4_b_*rf*: the interconnect of this 4×4 array instance consists of both vertical and horizontal buses and extra connections that connect the second and third row of the array to the VLIW register file.

Horizontal and vertical buses [see Fig. 3(d)] provide extra flexibility to the compiler, by enabling communication over a longer distance. The interconnect between an FU and its nearest neighbors [see Fig. 3(c)] provides the basic connectivity between all FUs in the form of a basic mesh topology. Adding extra connections to the next hop neighbors [see Fig. 3(a)] can improve the mapping efficiency, but adds an extra cost to every communication, as a larger tree of interconnect has to be driven. Additional connections to FUs that are further away are not considered here, as they would add an even higher energy cost and become even more expensive than using the bus, which is preferred for long distance communications. As the bus is shared between all FUs in a row or column, and can only be used by one of them at a given cycle, long distance communications are intentionally a scarce resource. Adding extra connections to the central register file [see Fig. 3(b)] allows the compiler to distribute loaded data easily over the array, which leads to a better utilization. Adding these connections leads to a larger fan-out for the ports of the central register file, translating to a higher energy cost for each access.

For a fixed size of the architecture, e.g., 8×8 FUs, these 4 template parameters lead to 16 possible different architecture instances. Excluding the architectures without any interconnect, the one with only the extra connections to the central register file, we end up with 14 valid possibilities. We also exclude all architectures that do not have nearest neighbor connections, but do have next hop connections. These two are equivalent from a compiler point of view, but next hop connections are more expensive in the physical implementation. To study the effect of the variation of the interconnect topology on performance and energy efficiency, the remaining ten architecture instances are evaluated in Section VI.

These interconnection architecture parameters have been chosen to be representative for a large sub-set of CGRAs. KressArray, Matrix, RAW, Garp, Remarc, MorphoSys and DreAM all consist of 2-D arrays of computational elements and use nearest neighbor, next hop and/or full length or segmented buses and therefore the analysis presented here is directly applicable to these architectures. Montium and RaPiD, use a 1-D organization of FUs and storage elements, connected with reduced crossbars or segmented buses. These architectures are a sub-class of the generic CGRA template, and the proposed interconnect aware architecture exploration method can still be applied. The Montium FUs internally consist of five smaller FUs, to which also the analysis directly applies. Architectures from e.g., SiliconHive are more heterogeneous, both in computational resources and in interconnection topology. In this case more effort has to be spent on a high level layout, in order to get reasonable wire length estimates, but the exploration method can still be applied. However, as all these architectures feature a comparable amount of computational resources and interconnections, it can be expected that interconnect will also be a major energy consumer and interconnect energy-aware exploration is essential.

VI. Experimental Section

Interconnect has a significant impact on the overall processor core performance and energy consumption: a more rich and flexible interconnect which directly increases the net energy consumption; and an indirect impact through the effect on the compiler viz. a higher flexibility can often lead to a better mapping and a corresponding higher performance. To quantify this impact, ten different processor instances, featuring the interconnect parameters that have been presented in Section V are used.

A. Benchmarks and Base Architecture

The experiments have been performed on a set of representative benchmarks from the wireless communication and multimedia domains—multiple-input–multiple-output (MIMO): MIMO channel estimation kernel with 52 pilots; Viterbi: a 189-state Soft Viterbi (SOVA); AVC_*motion*: in-house optimized version of the AVC motion estimation; AVC_*interpolate*: unoptimized version of the AVC half-pixel interpolation filter.

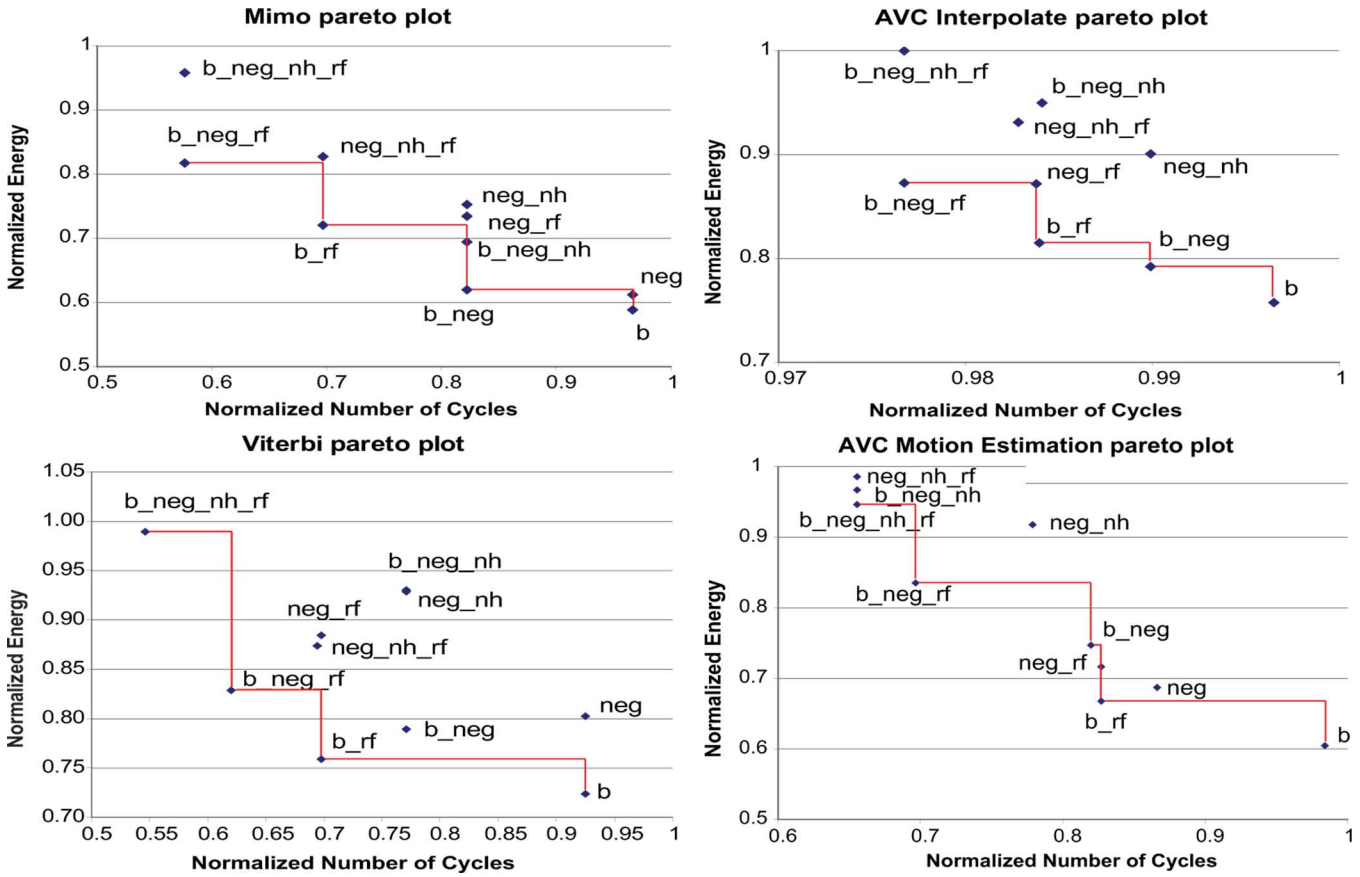

Fig. 4. Pareto plot of interconnect architecture exploration for MIMO, Viterbi, AVC Interpolate, and AVC motion estimation benchmarks.

All experiments are performed for an 8×8 ADRES array of which only the interconnection topology is varied (in our experience, the interconnect study of arrays of similar sizes, e.g., 4×4 to 12×12, leads to the same general conclusions). All FUs are of the arithmetic logic unit (ALU)-type, except the third and sixth columns, which are multipliers. The base architecture uses configuration memories of 64 entries deep and local register files that can store 16 words, while the central VLIW register file can store 64 words. The exploration of other array parameters is outside of the scope of this paper, and for more information the reader is referred to [9], [12].

Fig. 4 shows the simulation results for all four benchmarks. The X-axis shows the normalized number of cycles that is needed to complete the benchmark (inverse of performance), while the Y-axis shows the normalized energy consumption (of the complete architecture) for that specific task, and this over the ten architecture instances that are evaluated. A line connects all instances that are Pareto optimal. Only these are interesting for a design that considers the energy-performance tradeoff. By moving on a line from left to right, top to bottom, we move to more energy efficient architectures, but loose performance.

B. Results and Analysis

The experimental results show that a variation in interconnection topology has a significant effect on the energy consumption of the complete processor in all four cases, and ranging from 12% to 35% between the worst and best Pareto-optimal architectures for a certain benchmark. Compared to non-Pareto-optimal design choices, this difference can be even bigger. For performance, three out of four benchmarks also show a significant effect, from 35% to 40% between the best and worst Pareto-optimal variant. Looking at all four experiments, we can conclude that there is only a sub-set of interconnect architecture options (discussed in the following) that are often Pareto-optimal. There is a clear trend for both energy consumption and performance.

For the AVC interpolate filter in Fig. 4, unoptimized code was taken. In this case the compiler achieves an average instructions per cycle (IPC) count of only 3.5, meaning only about 3 FUs out of the 64 are used on average (5%), while the IPC of the array for the other benchmarks is between 30 and 40, which corresponds to a utilization of over 50%. Manual optimizations can improve this performance drastically and boost parallelism. In this case, we have chosen not to do this to show that even without any effect on performance, the interconnect can have a big effect on the energy consumption. The performance variation between the best and the worst architecture shown in Fig. 4 is only 2% for this benchmark, while the difference in energy consumption is still 23% when compared to the architecture with the most rich interconnect (b_neg_nh_rf).

1) b Architecture: The architecture that provides only horizontal and vertical buses for data interconnection (b) is both cheapest in terms of energy consumption and worst for performance (e.g., see Fig. 4, for all benchmarks). Because the buses are shared by a complete row or a complete column of FUs, interconnection resources are scarce if only buses are provided and performance is hurt badly.

2) neg Architecture: The architecture that provides only nearest neighbor connections (neg) is still restrictive and only performs better for the AVC motion estimation. For the unoptimized AVC interpolation filter however, the compiler was not able to map the benchmark on this architecture (there is no neg point in Fig. 4 for AVC interpolate).

This architecture is not very efficient for energy, and is Pareto-optimal in none of the benchmarks.

3) b_neg Architecture: Adding more connections slowly improves the performance, adding to the energy cost. b_neg is Pareto-optimal for three out of four benchmarks, giving a significant performance improvement of 15% in all cases (except AVC interpolate, where the range in performance is small overall). The energy cost that has to be paid for this is about 4% to 15% (e.g., moving from *b* to b_neg in Fig. 4 Viterbi).

4) _nh Architectures: (*neg_nh*, *b_neg_nh*, *neg_nh_rf*, *b_neg_nh_rf*): Adding next hop connections, although sometimes good for performance, almost never leads to a Pareto-optimal point, except for the "fully connected" b_neg_nh_rf in for AVC motion estimation and Viterbi. This architecture actually has the best performance for all of the benchmarks, but is only Pareto-optimal for two of them. In two of the four cases the extra interconnection freedom does not result in any performance improvement.

5) _rf Architectures: (b_rf, neg_rf, b_neg_rf, neg_nh_rf, b_neg_nh_rf): All architectures score better for performance when extra connectivity to the central register file (indicated as _rf) are added. These connections allow the compiler to distribute the incoming and outgoing data more efficiently over the array, and this leads to a better utilization and a higher performance. The improvement in performance for equal architectures, where the rf connection is added, can be over 25% in some cases (e.g., from b to b_rf for MIMO). However, adding these long wires to the cost of all register file read/writes adds between 5% to 20% to the energy cost, depending on the architecture (e.g., 13% for the same point).

For the given set of benchmarks and the explored architectures, we can conclude that the following architectures are Pareto-optimal, and can be used in the energy efficiency versus performance tradeoff: *b*, b_neg, b_rf, b_neg_rf and b_neg_nh_rf.

VII. Conclusion

In this paper, we have introduced an interconnect-aware exploration framework that allows a fast simulation based evaluation of different CGRA architecture instances, both from an energy and performance perspective. We have used this framework to present a study of different interconnect topologies in the context of the ADRES CGRA template, which supports a broad range of CGRAs styles. The results show an energy versus performance tradeoff of up to 30% for both criteria and we were able to identify a subset of architectures that consistently perform better (are Pareto-optimal) for the presented representative benchmarks from both the wireless communication and multimedia application domains.

References

[1] N. Bansal, S. Gupta, N. Dutt, A. Nicolau, and R. Gupta, "Network topology exploration of mesh-based coarse-grain reconfigurable architectures," in *DATE*, 2004, p. 10474.

[2] L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, and R. Zafalon, "A power modeling and estimation framework for vliw-based embedded system," *ST J. Syst. Res.*, vol. 3, no. 1, pp. 110–118, Apr. 2002.

[3] D. Burger, S. W. Keckler, K. S. McKinley, M. Dahlin, L. K. John, C. Lin, C. R. Moore, J. Burrill, R. G. McDonald, and W. Yoder, "Scaling to the end of silicon with edge architectures.," *IEEE Comput.*, vol. 37, no. 7, pp. 44–55, Jul. 2004.

[4] H. De Man, "Ambient intelligence: Giga-scale dreams and nano-scale realities.," in *Proc. ISSCC*, Feb. 2005, pp. 29–35.

[5] C. Ebeling, D. C. Cronquist, and P. Franklin, "Rapid—reconfigurable pipelined datapath.," in *Proc. FPL*, 1996, pp. 126–135.

[6] Z. Kwok and S. J. E. Wilton, "Register file architecture optimization in a coarse-grained reconfigurable architecture.," in *Proc. FCCM*, 2005, pp. 35–44.

[7] G. Lu, H. Singh, M.-H. Lee, N. Bagherzadeh, F. J. Kurdahi, and E. M. Chaves Filho, "The MorphoSys parallel reconfigurable system.," in *Proc. Euro-Par*, 1999.

[8] B. Mei, , S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "DRESC: A retargetable compiler for coarse-grained reconfigurable architectures.," in *Proc. FPL*, 2002, pp. 166–173.

[9] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix.," in *Proc. FPL*, 2003, pp. 61–70.

[10] T. Miyamori and K. Olukotun, "REMARC: Reconfigurable multimedia array coprocessor.," in *Proc. FPGA*, 1998, p. 261.

[11] Montium TP Processor, "Montium tile processor reference manual," 2005 [Online]. Available: http://www.recoresystems.com

[12] D. Novo, B. Bougard, P. Raghavan, H. Souk, and L. Van der Perre, "Energy-performance exploration of a CGA-based SDR processor," presented at the SDR Forum, Orlando, FL, 2006.

[13] PACT XPP Technologies, Los Gatos, CA, "PACT XPP Technologies homepage," 2003 [Online]. Available: http://www.pactcorp.com

[14] Philips Research, Eindhoven, The Netherlands, "Philips SiliconHive Avispa accelerator," [Online]. Available: http://www.siliconhive.com

[15] P. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, and D. Verkest, "EMPIRE: Empirical power/area/timing models for register files," *Int. J. Embed. Syst.*, 2007, to be published.

[16] J. M. Rabaey, "Reconfigurable computing: The solution to low power programmable DSP," in *Proc. ICASSP*, 1997, pp. 275–278.

[17] Sandbridge Technologies, Tarrytown, NY, "The Sandblaster Architecture," [Online]. Available: http://www.sandbridgetech.com

[18] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications.," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 465–481, May 2000.