Architecture Enhancements for the ADRES Coarse-Grained Reconfigurable Array

Frank Bouwens[1,2], Mladen Berekovic[1,2], Bjorn De Sutter[1], and Georgi Gaydadjiev[2]

[1] IMEC vzw, DESICS
Kapeldreef 75, B-3001 Leuven, Belgium
{bouwens,berekovic,desutter}@imec.be
[2] Delft University of Technology, Computer Engineering
Mekelweg 4, 2628 CD, Delft, The Netherlands
g.n.gaydadjiev@its.tudelft.nl

Abstract. Reconfigurable architectures provide power efficiency, flexibility and high performance for next generation embedded multimedia devices. ADRES, the IMEC Coarse-Grained Reconfigurable Array architecture and its compiler DRESC enable the design of reconfigurable 2D array processors with arbitrary functional units, register file organizations and interconnection topologies. This creates an enormous design space making it difficult to find optimized architectures. Therefore, architectural explorations aiming at energy and performance trade-offs become a major effort. In this paper we investigate the influence of register file partitions, register file sizes and the interconnection topology of ADRES. We analyze power, performance and energy delay trade-offs using IDCT and FFT as benchmarks while targeting 90nm technology. We also explore quantitatively the influences of several hierarchical optimizations for power by applying specific hardware techniques, i.e. clock gating and operand isolation. As a result, we propose an enhanced architecture instantiation that improves performance by 60 - 70% and reduces energy by 50%.

1 Introduction

Power and performance requirements for next generation multi-media mobile devices are becoming more relevant. The search for high performance, low power solutions focuses on novel architectures that provide multi-program execution with minimum non-recurring engineering costs and short time-to-market. IMEC's coarse-grained reconfigurable architecture (CGRA) called *Architecture for Dynamically Reconfigurable Embedded Systems* (ADRES) [1] is expected to deliver superior energy eficiency of 60MOPS/mW based on 90nm technology.

Several CGRAs are proposed in the past and applied in a variety of fields. The KressArray [2] has a flexible architecture and is ideal for pure dataflow organizations. SiliconHive [3] provides an automated flow to create reconfigurable architectures. Their architectures can switch between *standard DSP mode* and *pure dataflow*. The DSP mode fetches several instructions in parallel, which requires a wide program memory. In pure dataflow mode these instructions are executed in a single cycle. PACT XPP [4]

P. Stenström et al. (Eds.): HiPEAC 2008, LNCS 4917, pp. 66–81, 2008.

is a commercial reconfigurable architecture designed for multi-media and telecommunication applications. The architecture is fixed to 64 ALUs, which means the kernel mapping is constrained by these ALUs. MorphoSys [5] is a typical CGRA consisting of 8x8 basic units split-up into 4 tiles of 4x4 reconfigurable cells. Each cell in a tile is connected to all cells in the same row and column. There are also connections between the different tiles. The array speeds up the kernel, while a TinyRISC is utilized for the control section of the code. A more extensive overview of CGRAs is provided by Hartenstein in [6].

The ADRES template enables the designer to configure the architecture based on a variable number of functional units, register files and interconnections allowing advanced power and performance optimizations. Finding the optimal architecture for the customizable processor is not trivial task as there is a huge space of possible design points. Architectural explorations are mandatory to empirically find the architecture that best balances power, performance and cost characteristics.

Previous architectural explorations of ADRES were performed [7], [8] to find an optimal interconnection scheme for a good performance and power ratio. The architecture template obtained in [8] will function as the starting point of this work. This base template consists of a 4x4 array of FUs and local data register files. All these components are vertically, horizontally and diagonally interconnected.

The architecture in [8] showed the importance of not only interconnections, but also the register files of the coarse-grained array (CGA) of ADRES as these have a significant influence on power and performance. Kwok et al. [9] performed initial analysis of architectural explorations for the register file (RF) sizes and under utilization of the CGA, which motivated the drastic RF size reduction. As the DRESC CGA compiler and ADRES architectural template evolved the CGA utilization improved considerably making Kwok's results outdated for the latest compiler version (DRESC2.x).

This paper elaborates on the results of [8] on the interconnect and component level optimizations as the data sharing among functional units and register files can be improved significantly. We show the relevance of explorations to derive an efficient design for two widely used wireless and multi-media kernels (FFT and IDCT). We also show the fallacy that a system with distributed, fully interconnected register files is the best in terms of energy-delay. Our ADRES architectural proposal is modified by decreasing the sizes of the local register files.

The main contributions of this paper are:

- Careful architectural exploration of the register file distribution, interconnection topology and register file sizes for the CGRA;
- Empirical study of power, performance and energy-delay of all proposed intermediate and the final architectures;
- Quantitative evaluation of specific optimizations such as clock gating, operand isolation and pipelining;
- Determination of an energy optimized architecture for IDCT and FFT;
- Array size modifications of the proposed architecture for energy-delay analysis.

This paper is organized as follows. Section 2 briefly describes the ADRES architecture and the programming model. Section 3 presents the utilized tool flow during the explorations. Three different optimizations are described and benchmarked in Section 4 to

select the final architecture based on power and energy-delay results. Section 5 presents the intermediate and final implemented results of the created architecture instances. The conclusions section finalizes this paper.

2 ADRES Base Architecture and Programming Model

The ADRES architecture based on its template [1] is a tightly-coupled architecture that can operate in either VLIW or CGA mode. Figure 1 shows the selected 4x4 ADRES base architecture of [8] including data and instruction memories. When the architecture is operating in VLIW mode the performance is improved due to instruction level parallelism. In CGA mode performance is improved by parallelizing loops on the array (loop level parallelism). ADRES based systems are programmed in ANSI-C language. Any existing ANSI-C program can be modified to suit the ADRESS CGA by modifying the if-conversions and removing all nested loops as described in [1]. No additional instructions in the source code are needed for the compiler to map the loops to the CGA. Code that could not be mapped on the array is executed on the VLIW relying only on instruction level parallelism.

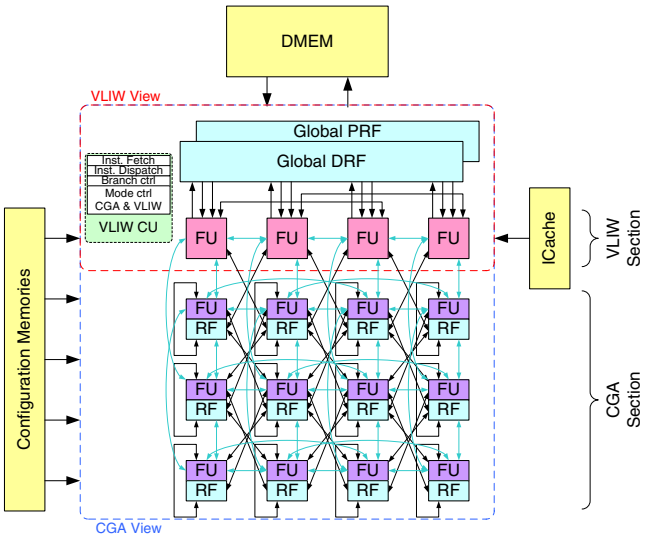


Fig. 1. ADRES Instance example

The VLIW control unit (CU) controls the program counter (PC) and is responsible for fetching instructions from the instruction cache. The switch between VLIW and CGA modes is directed by the same VLIW CU by fetching a CGA instruction in VLIW mode. The configuration memories (CM) are addressed by a control unit and provide instructions and routing data for the entire array during CGA operation.

Communication between the two modes goes via the multi-port global Data Register File (DRF) or data memory (DMEM). The VLIW functional units (FU) communicate

through the global DRF with eachother. The CGA FUs communicate through the global DRF, local data and predicate register files (PRF) and the dedicated interconnections. The predicate register files or busses handle branches, loop prologue and epilogue for proper control flow.

The VLIW section of the base architecture has a four instructions issue width, while the CGA section has an issue width of 4 by 3 (12) instructions. The template consists of mesh, mesh plus and diagonal interconnections [8] between FUs and local DRFs. This resulted in good routing capabilities in the array, but can be improved as researched in this paper.

3 Tool Flow

The tool flow used in this architecture exploration study is the same as used in [8]. It provides the necessary results in terms of performance, power and energy usage. A simplified representation of this flow is depicted in Figure 2.

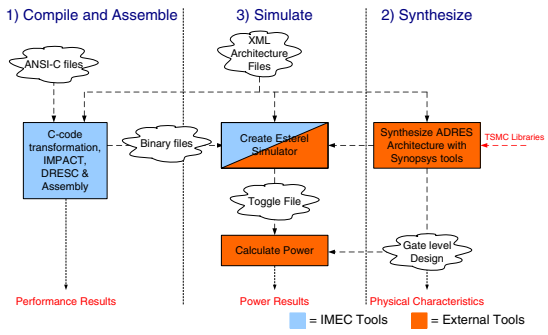


Fig. 2. Simple representation of Tool Flow

All three steps in the figure (Compile and Assemble, Synthesize and Simulate) use the same ADRES XML architecture file as input. The first step, *Compile and Assemble*, maps ANSI-C code on either the CGA or the VLIW architecture views. This step generates the program binary files needed for the HDL simulation stage, but also provides the number of instructions and cycles. The latter are used to calculate performance using high-level simulations of the applications on a given architectural instance.

The second step, *Synthesize*, translates the XML file into a top-level VHDL file and synthesizes the architecture using 90nm TSMC libraries. Physical characteristics are obtained from the gate-level architecture with 90nm, regular-Vt (1V, 25°C) general purpose libraries. The final architecture is also placed and routed to obtain the circuit layout.

The third step, *Simulation*, utilizes either the enhanced Esterel [10] or the ModelSim v6.0a simulator for HDL verification and analysis. The Esterel simulator provides faster results compared to ModelSim without significant loss of accuracy [8]. Annotating the captured switching activity of the HDL simulations onto the gate-level design results in power figures.

4 Architectural Explorations

The architecture explorations in the following sections start from the base architecture presented in Figure 1 that was analyzed for energy efficiency in [8]. The power distribution of the base architecture for IDCT are depicted in Figure 3. The components with the highest consumption (primary targets for improvement) are the configuration memories (CMs: 37.22%), the FUs (19.94%) and the DRFs (14.80%) of the CGA.

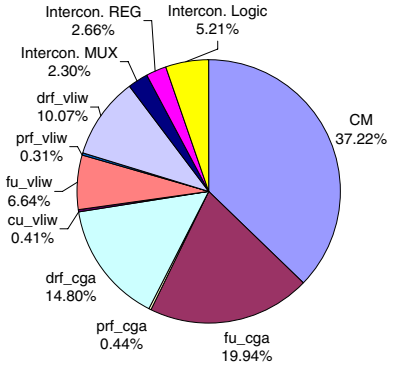


Fig. 3. Power distribution of our base architecture for IDCT: 80.45mW

We optimize these three components using the following methods:

CM: Create an array with less configuration bits by reducing the number of architecture components or by using simpler interconnections;
FU_CGA: Improve the FU design from non-pipelined to pipelined (VHDL modifications) and optimize the routing of the CGA array (XML architecture description update);
DRF_CGA: Reduce the register file sizes, apply clock gating and use register file sharing.

Sharing data between the FUs and the local DRFs and PRFs is important for the power consumption and performance of the architecture as these influence the CMs, FUs and the local DRFs in power and performance. Therefore we will focus the explorations on routing and the register files. We only utilize IDCT and FFT kernels for architecture explorations due to the fact that simulating complete applications such as MPEG2 would result in prohibitively long simulation times. We will perform the following experiments for the explorations:

Local DRF distribution: Determine the influences of the RFs in the array by exploring the distribution of the local data register files;
Interconnection Topology: Determine the influence of additional interconnections. More interconnections improve routing, but increases the power consumption and vice-versa;

Register File Size Reduction: Determine what is the minimum size of the local DRFs and PRFs. This results in local register file size optimally fitting the array increasing the performance vs. power ratio.

Our exploration and comparison starts from the result architecture obtained in [8] and shown in Figure 1. All explored architectures have a CGA dimension of 4x4, 32-bit data bus and are non-pipelined. Pipelining of the FUs is of little relevance for our study as the performance vs. power ratio remains constant. However, pipelining will be applied to the selected architecture in Section 5 and analyzed with different array sizes. Furthermore, the VLIW DRF and PRF have 64 words of which 14 are rotating for the DRF and 32 for the PRF. The local DRFs and PRFs have 16 words created as rotating register files. The configuration memories have 128 words and vary in width depending on the CGA organization.

4.1 Distributing Local Data Register Files

This section investigates the influence of the local DRFs and PRFs on power and performance for the array. This study is based on a fixed mesh plus interconnection topology between the FUs with vertical and horizontal connections [8]. Among the FUs we explore variable register file distributions proposed in [11] which are also depicted in Figure 4. There are eight FUs with multiplication and four FUs with memory LD/ST capabilities.

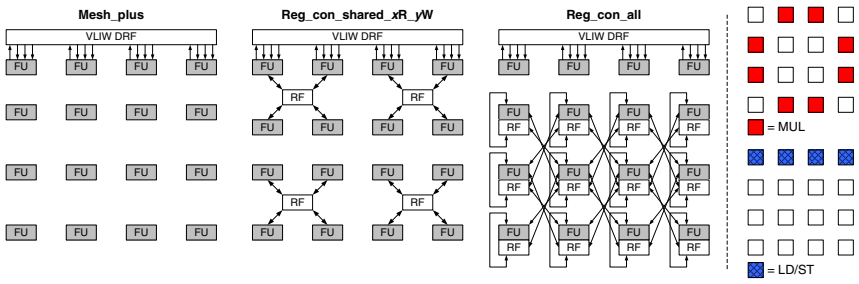


Fig. 4. Distribution of the Local DRFs

The three different architectures have a local PRF for each FU that can be replaced by predicated busses (noted with suffix *_pd* in Table 1). The capability of storing and later processing is not possible with the busses that is a potential performance bottleneck.

The simplest architecture *mesh_plus* does not have local DRFs and completely rely on the available data busses for all data transfers. Only the first row of FUs is connected to the global DRF. The architecture *reg_con_shared_xR_yW* shares its inputs and outputs of the RFs to decrease the area of the RFs and share data more efficiently. For the shared RF architecture we investigate the influence of the number of ports for the local RFs. More precisely we simulated instances with 2, 4 and 8 read ports. The most complex architecture *reg_con_all* has a DRF for each FU in the array and is similar to the one

Table 1. Distributed DRFs Names

Original	Renamed	Original	Renamed
4x4_mesh_plus	arch_1	4x4_mesh_plus_pred_bus	arch_1_pb
4x4_reg_con_shared_2R_1W	arch_2	4x4_reg_con_shared_2R_1W_pred_bus	arch_2_pb
4x4_reg_con_shared_4R_2W	arch_3	4x4_reg_con_shared_4R_2W_pred_bus	arch_3_pb
4x4_reg_con_shared_8R_4W	arch_4	4x4_reg_con_shared_8R_4W_pred_bus	arch_4_pb
4x4_reg_con_all	arch_5	4x4_reg_con_all_pred_bus	arch_5_pb

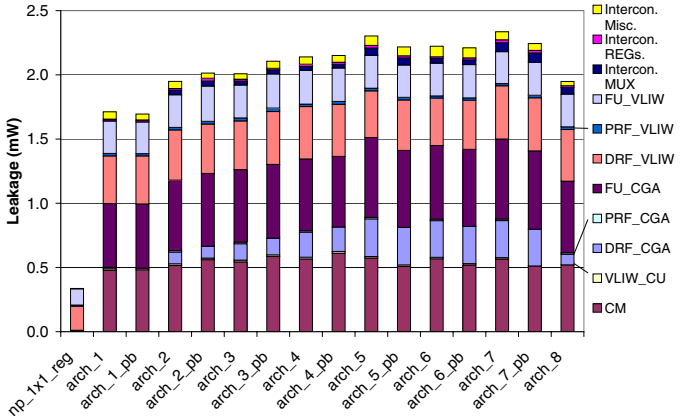


Fig. 5. Leakage Power

depicted in Figure 1. The created architectures are noted in Table 1 and the names are abbreviated for ease of explanation.

All results presented here and in Section 4.2 are placed together in Figures 5 and 7 - 10. The leakage results are depicted in Figure 5 and the power results at 100MHz of IDCT and FFT are presented in Figures 7 and 8, respectively. The energy-delay results are depicted in Figures 9 and 10. We will discuss the results presented here first.

Removing the local DRFs in the first two architectures (*arch_1* and *arch_1_pb*) result in a low energy consumption, but decreased performance as depicted in Figures 9 and 10. This indicates the DRFs are beneficial during CGA operation. Replacing the PRFs with a bus increases energy consumption by (FFT: 10 - 46%, IDCT: 5 - 37%) except for *arch_5*. This experiment shows that storing predicate results of previous operations in local PRFs is essential for overall power consumption in cases with few local DRFs or no DRF at all.

The experiment in this section showed that register file sharing creates additional routing for the DRESC compiler that improves scheduling density of the array. Interconnection complexity is reduced requiring less configuration bits, hence decreasing the size of the configuration memories. Replacing four RFs with only one reduced leakage and IDCT/FFT power consumption of the local DRFs. Local RFs with 1 write and 2 read ports in *arch_2* outperforms in terms of power and energy-delay the fully distributed RFs architecture *arch_5* for both, FFT and IDCT, kernels.

4.2 Interconnection Topology

This section explores the impact of additional interconnections on power and performance. This enhancement has a fixed *reg_con_all* RF distribution similar to the base architecture in [8]. The three different interconnection topologies are depicted in Figure 6. The resources and LD/ST units are distributed in the same way as in Figure 4.

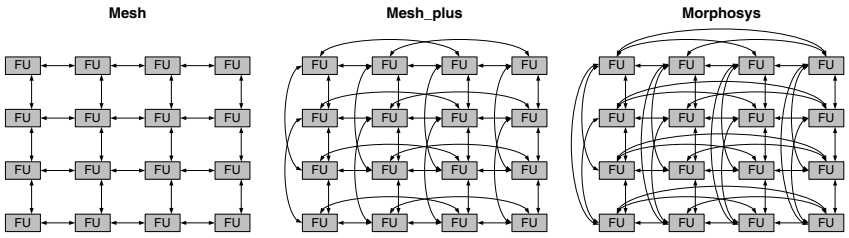


Fig. 6. Interconnection Topologies

We apply the same methodology as in Section 4.1 by replacing the local PRFs with busses. The different architectures explored are listed in Table 2 and their results are merged with the results of Section 4.1 and depicted in Figures 5 to 10. Combining the Mesh_plus architecture with the reg_con_all base architecture in Figure 6 would result in *arch_5* in Table 1, hence it is omitted in Table 2. The Morphosys interconnection is based on the Morphosys architecture [5] that fully interconnects the array in both row and column directions. The final *arch_8* architecture will be explained at the end of this section.

By considering only the architectures noted in Table 2 we notice that additional interconnections in the base architecture are beneficial for energy-delay [9]. This is especially noticeable with the Morphosys architecture (*arch_7*). Figures 7 and 8 show that replacing the local PRFs by predicate busses decreases the overall power and energy consumption. Although the architecture *arch_5* is better than *arch_7_pb* in both power and energy, the additional Morphosys connection would be useful for larger arrays e.g. 8x8 and more. Therefore, we selected the *arch_7_pb* as the best fit for IDCT and FFT.

When the results of Sections 4.1 and 4.2 are combined, the *arch_8* architecture with shared DRFs, Morphosys connections and local PRFs is created as depicted in Figure 11. The predicate busses are omitted as they showed not to improve power and performance of the architecture with shared register files. Figure 9 and 10 show *arch_2* is energy and delay appropriately, however it lacks the Morphosys interconnection that

Table 2. Interconnection Topologies Names

Original	Renamed	Original	Renamed
4x4_reg_con_all_mesh	arch_6	4x4_reg_con_all_mesh_pred_bus	arch_6_pb
4x4_reg_con_all_morphosys	arch_7	4x4_reg_con_all_morphosys_pred_bus	arch_7_pb
4x4_reg_con_shared_2R_1W_morphosys	arch_8		

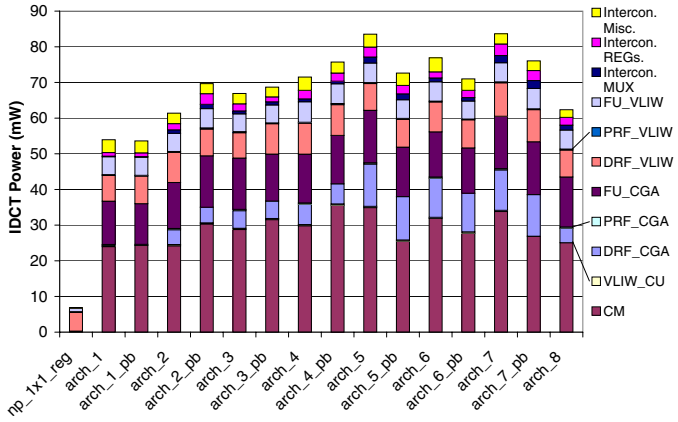


Fig. 7. IDCT Power @ 100MHz

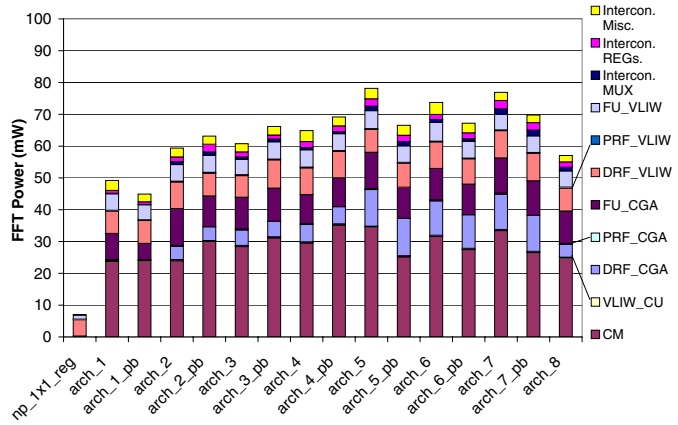


Fig. 8. FFT Power @ 100MHz

is beneficial for larger arrays. Table 3 notes the improvements of *arch_8* over the base architecture in Figure 1. The results clearly show an improvement in performance (MIPS/mW) of 14 - 16%. Power consumption was decreased by 22%, however, energy improved only by 1.6% as additional execution cycles are required for the applications due to the reduced local DRFs and PRFs sizes. Minimizing the number of local DRFs has a beneficial effect on area resulting in 14.4% reduce. We select the *arch_8* architecture for further optimizations and explorations.

4.3 Register File Size Modification

In this section we evaluate the selected architecture *arch_8* of Section 4.2 with variable local DRF and PRF sizes to improve power while maintaining performance of the architecture for kernels considered here. The global DRF and PRF can not modified and

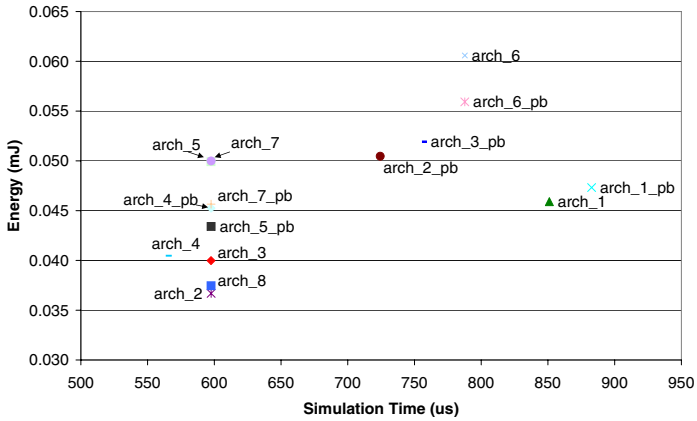


Fig. 9. IDCT Energy-Delay @ 100MHz

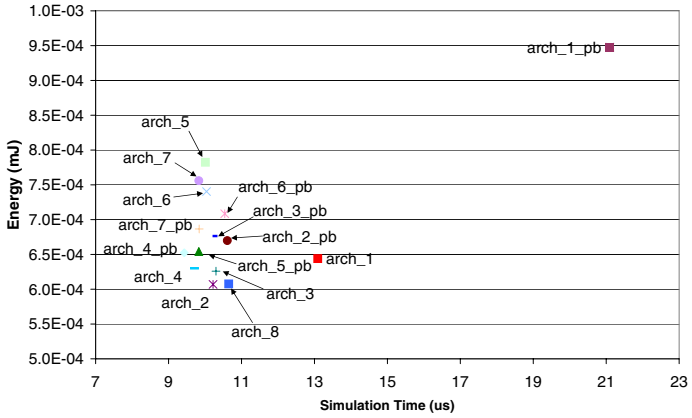


Fig. 10. FFT Energy-Delay @ 100MHz

Table 3. Base vs. arch_8 for IDCT & FFT @ 100MHz

Benchmark	MIPS/mW	mW/MHz	Power (mW)	Energy (uJ)	Area mm^2
IDCT					
base	17.51	0.81	80.45	37.72	1.59
arch_8	20.00	0.63	62.68	37.46	1.36
Improvement	14.22%	22%	22%	0.6%	14.4%
FFT					
base	9.40	0.72	73.28	0.62	
arch_8	10.95	0.57	57.05	0.61	
Improvement	16.5%	20.8%	22.1%	1.6%	

fixed at 64 words. The global DRF has 8 read and 4 write ports, which are 32-bits wide. The global PRF has 4 read and write ports each 1-bit wide.

The results in Table 4 show that 4 registers provide the least amount of instructions and cycles with an optimal instructions per cycle (IPC) for an 4x4 array. The registers that are not used will be clock gated reducing the negative impact on power. Interesting to note is the decrease of IPC with increasing RF sizes. This was unexpected as IPC usually saturates with increasing RF sizes. Nevertheless, our tests showed that the scheduler of DRESC2.x improved over time [9] as the IPC increased with better usage of the local DRFs, but the number of utilized registers is still relatively low.

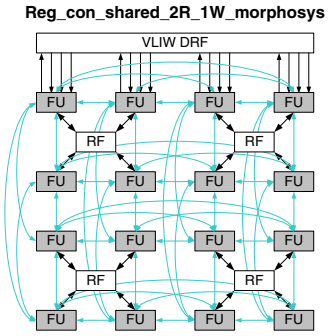


Fig. 11. Proposed ADRES architecture with shared RFs

Table 4. Reducing Register File Size arch_8

Application	Local Register File Size 2	4	8	16
IDCT				
Instructions	974632	923940	923980	923960
Cycles	62924	59755	59762	59757
IPC	9.69	10.21	10.21	10.21
FFT				
Instructions	11364	10532	11040	11060
Cycles	1087	1035	1063	1065
IPC	2.48	2.73	2.57	2.58

5 Final Results

The optimizations in Sections 4.1 till 4.3 led to *arch_8* architecture with shared local DRFs, Morphosys interconnections and reduced RF sizes. The architecture discussed is a non-pipelined version with no further optimizations of the architecture and data path. We apply three additional optimizations for the architecture and data path in this section: clock gating, operand isolation and pipelining. *Clock gating* targets the register files by reducing their switching activity. This feature is implemented automatically by Synopsys *Power Compiler*. Empirical results show a power reduction of the register file between 50 - 80%. A pipelined version of the same architecture shows 20 - 25% power improvement in overall. *Operand Isolation* targets the data path of a FU reducing switching activity of unused components. It is implemented manually as the automated version built in the design tools used only reduced the power by 1%. Our manual implementation using OR-based isolation [12] reduced power by 30% for a single FU and 30 - 40% for the overall pipelined system. *Pipelining* increases performance significantly by creating shorter critical paths and provides higher throughput. Pipelining, which is implemented by hand, has a disadvantage that power increases linearly when increasing the frequency unless clock gating and operand isolation is used. These optimizations are most efficient with multi-cycle architectures and very suitable for pipelined architectures.

Table 5. Comparing base (100MHz) with final instance (312MHz)

	Total Power (mW)	Total Energy (uJ)	MIPS	MIPS/mW	mW/MHz
FFT					
Base	73.28	0.619	759	10.35	0.7328
Final	67.29	0.307	1190	17.68	0.2153
Improve	8.17%	50.4%	56.78%	70.82%	70.62%
IDCT					
Base	80.45	37.72	1409	17.51	0.8045
Final	81.99	19.14	2318	28.27	0.2624
Improve	-1.91%	49.25%	64.51%	61.45%	67.38%

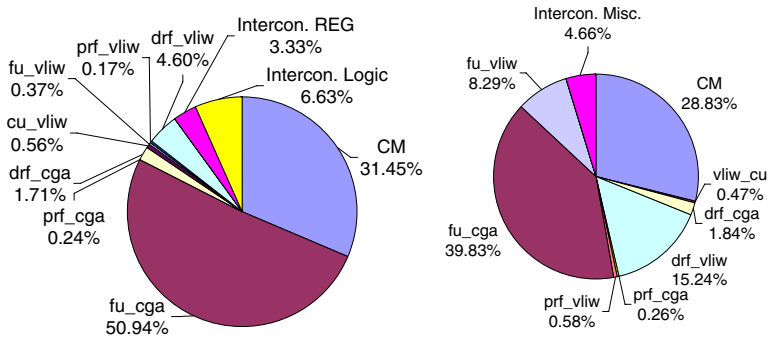
5.1 Putting It All Together

Combining the *arch_8* architecture with the aforementioned optimizations results in a low power, high performance ADRES instance: *4x4_arch_8_4L_final*. A comparison between the proposed architecture with the base architecture (shown in Figure 1) is provided in Table 5. The results indicate a moderate improvement in power of 8%, but with a higher performance of 56 - 65% due to the pipelining and routings features. This results in lower energy dissipation of the architecture by 50%. The area of the proposed architecture was improved from 1.59mm^2 (544k gates) to 1.08mm^2 (370k gates), which is equivalent to a 32% improvement.

5.2 Final Architecture Power Decomposition

The final *4x4_arch_8_4L_final* architecture is placed and routed using Cadence SOC Encounter v4.2. The power and area of the proposed architecture layout are decomposed in Figures 12(a) and 12(b), respectively. These figures are of the ADRES core architecture excluding data and instruction memories. Due to the fact that the final architecture is pipelined the clock tree contribution (4.67mW) is included in these figures. The data memory and the instruction cache were not included in the synthesis for which no power estimations are made. The multiplexors in the CGA datapath were removed during synthesis by the synthesis tool as this was beneficial for performance.

Comparing Figure 3 with Figure 12(a) we notice that the shared local DRFs combined with clock gating results in lower power consumption. The configuration memories still require a vast amount of power and area, but have decreased in size as well. Further optimizations of the configuration memories require advance power management e.g. power gating, which was not applied in the final architecture. Interesting to note is the relatively higher power consumption of the CGA FUs compared to Figure 3. This is caused by the higher utilization of the array compared to the base architecture consuming more power, but providing higher performance. This increases power efficiency as noticeable in Table 3. The 16 CGA FUs and the CMs require 68.66% of all the area as depicted in Figure 12(b). The largest single component is the global DRF (noted as drf_vliw) with 8 read and 4 write ports.



(a) IDCT Power: 81.99mW (b) Area: 1.08mm^2

Fig. 12. Results of 4x4_arch_8_4L_final @ 312MHz

5.3 Energy-Delay Architectures Analysis of Different Array Sizes

The proposed architecture was created as a 4x4 array, however, different array sizes e.g. 2x2 and 8x8 are of interest to determine the most efficient architecture dimensions using the same interconnection network. The same sizes for the global (64 registers) and local RFs (4 registers) are maintained and the FUs are pipelined. Changing the array dimension implies different routing capabilities. For example, a 2x2 array has less routing overhead and requires reduces the possibilities for the compiler to map loops efficiently on the array. This requires deeper and larger configuration memories to map a loop on the array and increases power consumption. An 8x8 array improves the possibilities for the compiler to map loops on the array and reduces the sizes of the CMs.

We compare the pipelined architectures with non-pipelined key architectures mentioned in this paper. All key architectures in this paper are noted in Table 6 including their frequencies of which the energy-delays are depicted in Figures 13 and 14. The first three architectures are non-pipelined as the last three are pipelined. The *4x4_arch_8_16L* architecture has 16 register words in the local DRFs and PRFs. The architectures with *4L* in their name have 4 register words in the local DRFs and PRFs as explained in Section 4.3.

Table 6 shows that modifying the size of an ADRES instance creates different critical paths by increasing frequency of a 2x2 and decreasing for an 8x8 instance. The energy-delay charts in Figure 13 and 14 show that the proposed pipelined 4x4 architecture is superior to all other architectures. The 8x8 instance has the same performance as the 4x4 architecture for the IDCT code, however, due to its larger size and power consumption the energy consumption is also higher. For the FFT code the 8x8 architecture is

Table 6. Key Architectures

Non-pipelined Architecture	Freq (MHz)	Pipelined Architecture	Freq (MHz)
base	100	4x4_arch_8_4L_final	312
4x4_arch_8_16L (16L DRFs)	100	2x2_arch_8_4L_final	322
4x4_arch_8_4L (4L DRFs)	100	8x8_arch_8_4L_final	294

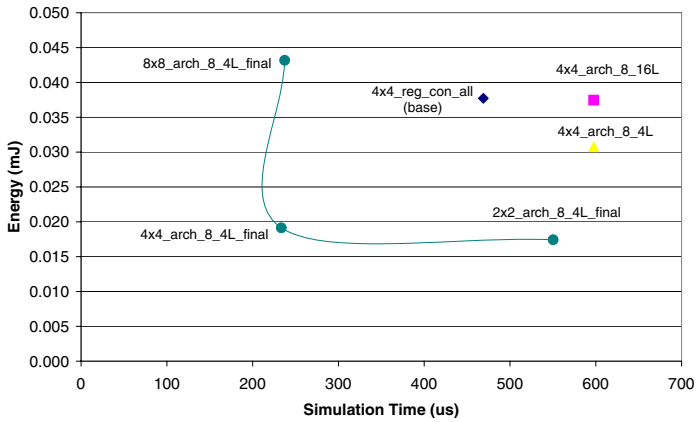


Fig. 13. Energy-Delay IDCT Results of Key Architectures

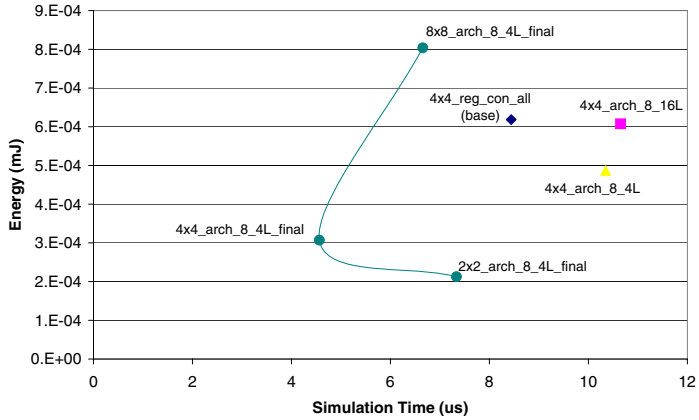


Fig. 14. Energy-Delay FFT Results of Key Architectures

minimally faster than the 2x2. This is because the scheduler failed to map one key loop of the FFT code on the array that is reducing performance considerably.

The key conclusions that can be drawn from our study are:

- The CGA is highly dependent on the local DRFs. Sharing DRFs among FUs improves routing for the DRESC compiler increasing scheduling density of the array;
- Replacing PRFs with busses is only beneficial if there are sufficient local DRFs;
- The optimal local DRF and PRF sizes of the proposed architecture is 4 words without influencing performance. The DRESC compiler can be enhanced for the array by improving the local DRF utilizations for loops;
- An array dimension of 4x4 balances energy vs. performance best for the FFT and IDCT kernels.

6 Conclusions

This paper delivers for the first time systematic explorations on the ADRES coarse-grained reconfigurable array template. An existing ADRES instance was optimized in terms of power, performance and energy. Several architectures were explored by focussing on the register file distributions and different interconnect topologies. The proposed architecture is evaluated for reduction of register file sizes.

The distribution of local data register files provided optimal results when the DRFs with 2 read and 1 write port are shared among 4 diagonally, neighboring functional units. This created additional routing capabilities for the DRESC scheduler and improved data sharing and scheduling density of the array. Replacing predicate register files with busses diminished power and energy-delay results. The results of the interconnection topology exploration showed that a fully interconnected array of the FUs and RFs in both row and column direction was optimal. Applying predicate busses improved power and energy consumption when there were sufficient local DRFs available. The final proposed ADRES instance consists of local PRFS and shared local DRFs, the Morphosys interconnection scheme and optimizations like clock gating, operand isolation and pipelining. Comparing 2x2, 4x4 and 8x8 instances based on the energy vs. delay shows that the 4x4 instance performed optimal as the instruction scheduling density is highest. The DRESC compiler has significant room for improvement for larger arrays.

In conclusion we show that ADRES offers an attractive path for low power scaling of e.g. VLIW DSP cores. The proposed ADRES architecture shows good performance efficiency of 25MIPS/mW and power efficiency 0.24mW/MHz at 312MHz. This improves the performance (MIPS/mW) by 60 - 70% and energy by 50% The energy consumption is 0.307uJ - 19.14uJ and the performance is 1190 - 2318 MIPS for FFT and IDCT, respectively. The area utilization is 1.08mm^2 for the ADRES core studied here targeting 90nm TSMC libraries. Comparing the ADRES base architecture with the proposed ADRES architecture the performance (MIPS/mW) improved by 60 - 70%, energy by 50% and area by 32%.

References

1. Mei, B., Vernalde, S., Verkest, D., Man, H.D., Lauwereins, R.: ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix. In: IMEC 2003, Kapeldreef 75, B-3001, Leuven, Belgium (DATE 2004)
2. KressArray, `http://kressarray.de`
3. SiliconHive, `http://www.silicon-hive.com`
4. PACT XPP Technologies, `http://www.pactxpp.com`
5. Singh, H., Lee, M.-H., Lu, G., Kurdahi, F.J., Bagherzadeh, N.: MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications. In: University of California (US) and Federal University of Rio de Janeiro (Brazil), pp. 465–481. IEEE Transactions on Computers, Los Alamitos (2000)
6. Hartenstein, R.: A Decade of Reconfigurable Computing: A Visionary Retrospective, CS Dept (Informatik), University of Kaiserlautern, Germany, March 2001, Design, Automation and Test in Europe, 2001. Conference and Exhibition pp. 642–649 (2001)

7. Lambrechts, A., Raghavan, P., Jayapala, M.: Energy-Aware Interconnect-Exploration of Coarse Grained Reconfigurable Processors. In: WASP. 4th Workshop on Application Specific Processors (September 2005)
8. Bouwens, F., Berekovic, M., Kanstein, A., Gaydadjiev, G.: Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array. In: Diniz, P.C., Marques, E., Bertels, K., Fernandes, M.M., Cardoso, J.M.P. (eds.) ARC 2007. LNCS, vol. 4419, pp. 1–13. Springer, Heidelberg (2007)
9. Kwok, Z., Wilton, S.J.E.: Register File Architecture Optimization in a Coarse-Grained Reconfigurable Architecture. In: FCCM 2005. Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, vol. 00, University of British Columbia (2005)
10. http://www-sop.inria.fr/esterel.org/
11. Mei, B., Lambrechts, A., Mignolet, J.-Y., Verkerst, D., Lauwereins, R.: Architecture Exploration for a Reconfigurable Architecture Template. In: IEEE Design & Test of Computers, pp. 90–101. IMEC and Katholieke Universiteit Leuven (March 2005)
12. Münch, M., Wurth, B., Mehra, R., Sproch, J., Wehn, N.: Automating RT-Level Operand Isolation to Minimize Power Consumption in Datapaths. In: Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000, pp. 624–631 (2000)