

Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array

Frank Bouwens¹, Mladen Berekovic¹, Andreas Kanstein², and Georgi Gaydadjiev³

¹ IMEC vzw, DESICS

Kapeldreef 75, B-3001 Leuven, Belgium

{frank.bouwens, mladen.berekovic}@imec.be

² Freescale Semiconducteurs, Wireless & Mobile Systems Group

Kapeldreef 75, B-3001 Leuven, Belgium

a.kanstein@freescale.com

³ Delft University of Technology, Computer Engineering

Mekelweg 4, 2628 CD, Delft, The Netherlands

g.n.gaydadjiev@its.tudelft.nl

Abstract. Reconfigurable computational architectures are envisioned to deliver power efficient, high performance, flexible platforms for embedded systems design. The coarse-grained reconfigurable architecture ADRES (Architecture for Dynamically Reconfigurable Embedded Systems) and its compiler offer a tool flow to design sparsely interconnected 2D array processors with an arbitrary number of functional units, register files and interconnection topologies. This article presents an architectural exploration methodology and its results for the first implementation of the ADRES architecture on a 90nm standard-cell technology. We analyze performance, energy and power trade-offs for two typical kernels from the multimedia and wireless domains: IDCT and FFT. Architecture instances of different sizes and interconnect structures are evaluated with respect to their power versus performance trade-offs. An optimized architecture is derived. A detailed power breakdown for the individual components of the selected architecture is presented.

1 Introduction

The requirements for higher performance and lower power consumption are becoming more stringent for newest generation mobile devices. Novel chip architectures should be able to execute multiple performance demanding applications while maintaining low power consumption, small area, non-recurring engineering costs and short time-to-market. IMEC develops a coarse-grained reconfigurable array (CGRA) called *Architecture for Dynamically Reconfigurable Embedded Systems* (ADRES) [1]. ADRES is a power efficient, but still flexible platform targeting 40MOPS/mW with 90nm technology. ADRES provides the designer with the tools to design an application specific processor instance based on an architecture template. This enables the designer to combine an arbitrary number of functional units, interconnects and register files.

The main contributions of the paper are:

- A toolflow for energy, power and performance explorations of ADRES based CGRA architectures;

- A novel methodology for power analysis that replaces RTL simulation with instruction set simulation to obtain activity stimuli;
- Analysis of performance, power and energy tradeoffs for different array sizes and interconnect topologies;
- Derivation of an optimized architecture for key multimedia and wireless kernels and the power breakdown of its components.

This paper is organized as follows. Section 2 briefly introduces the ADRES architectural template. In Section 3 the toolflow and the developed power simulation methodology are presented. Section 4 contains the results of the architectural exploration experiments, the selection of the power/performance optimized instance and the detailed power breakdown of its components. Finally, the conclusions are presented in Section 5.

2 ADRES Template

The ADRES architecture is a tightly-coupled architecture between two views: VLIW and CGA. Figure 1 shows an example of a 4x4 ADRES instance.

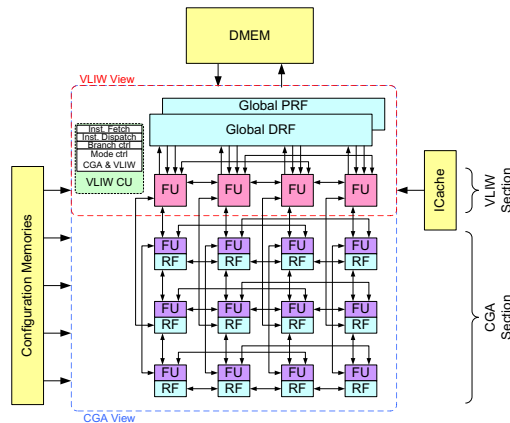


Fig. 1. ADRES Instance example

The VLIW control unit (CU) starts and stops the CGA loops when initiated by an CGA instruction in VLIW mode that works as a function call. It also accesses the instruction cache for the next instruction for the VLIW section and calculates the next Program Counter (PC).

The VLIW section in the example has an issue width of four instructions, while the CGA section has an issue width of 4 by 3 (12) instructions. The FUs in the VLIW view communicate through a multi-port global Data Register File (DRF), while those in the CGA use:

- global DRF;
- local register files (LRF);
- dedicated interconnects between the FUs.

Example of the data path in ADRES is depicted in Figure 2. All FUs have 1 destination and 3 source ports at most. There are local Data and Predicate Register Files (PRF) used for storage of variables. The ADRES template is so flexible that it allows all, operation without local RFs, single local RF per FU or shared register files among several FUs [8].

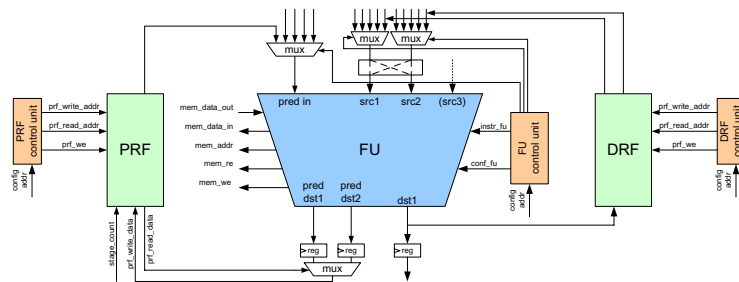


Fig. 2. ADRES Data path example

3 Tool Flow

Figure 3 depicts the proposed tool flow for architectural exploration. It consists of basically three parts: 1) Compilation and Assembly providing binary files and a compiled instruction level simulator, 2) Synthesis providing gate level netlist and physical characteristics needed for power calculation and 3) Simulation to obtain power and performance figures.

The *Compile and Assemble* part transforms the ANSI-C application code into an optimized binary file. The code is first processed by the IMPACT [3] frontend that performs various ILP optimization and transforms it into an intermediate representation called Lcode. The DRESC2.0 compiler in Figure 3 reads the Lcode, performs ILP scheduling, register allocation and modulo scheduling for CGA mode and generates the optimized DRE files. These files are used to create a high level simulator which provides basic performance parameters such as instructions per cycle (IPC). In addition, the Assembler tool creates the binary files needed for the cycle true Esterel and ModelSim simulations.

The ADRES instance XML description is transformed into VHDL files and is synthesized in the *Synthesize* part. Synopsys *Physical Compiler v2004.12-SP1* [5] is used to create the netlist for 90nm CMOS library, from which area, capacitance and resistor values are extracted to be used multiple times.

In the *Simulate* part three different simulators are used. The compiled ISA simulator (marked as A in Figure 3) provides the performance numbers. ModelSim RTL simulator is used to simulate the generated RTL VHDL files at highest level of hardware accuracy

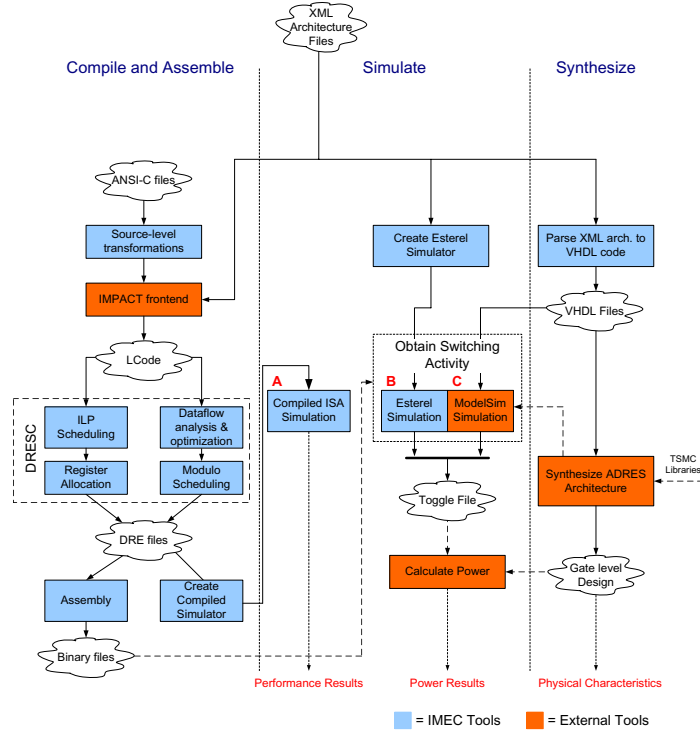


Fig. 3. Global Tool Flow overview

and to obtain the switching activity figures needed for RTL power calculations. The Esterel simulator [4] is based on the Esterel synchronous language dedicated to control-dominated reactive systems and models the entire ADRES instance as a state machine. The advantage of the Esterel simulator is the reduction in time to simulate a benchmark compared to ModelSim RTL simulations, as it is 8 - 12 times faster reducing overall time by a factor of 3 - 6.5 depending on the application. In addition, it is the behavior-level reference model for the core architecture available much earlier than the verified RTL.

Facilitated by its language, the Esterel simulator has the same structure as the actual implementation. Thus by enhancing it with bit toggle counting functions written in C, we are able of capturing the signal activity statistics of almost all relevant connections. This switching activity is what an RTL HDL simulator also generates for the power analysis tool. The match between the generated data is established by using the same signals as defined in the XML architecture description file.

The switching activities obtained after simulations are annotated on the gate level design created in the synthesize part. The toggling file and the gate level design are used by the *PrimePower v2004.12-SP1* of Synopsys [5] to estimate power.

For evaluation of the accuracy of the Esterel cycle accurate simulation versus the ModelSim VHDL simulation we used the IDCT kernel from MPEG. The IDCT benchmark is compiled for the VLIW only and for VLIW/CGA mode. The results are

Table 1. Differences between Esterel and ModelSim for ADRESv0

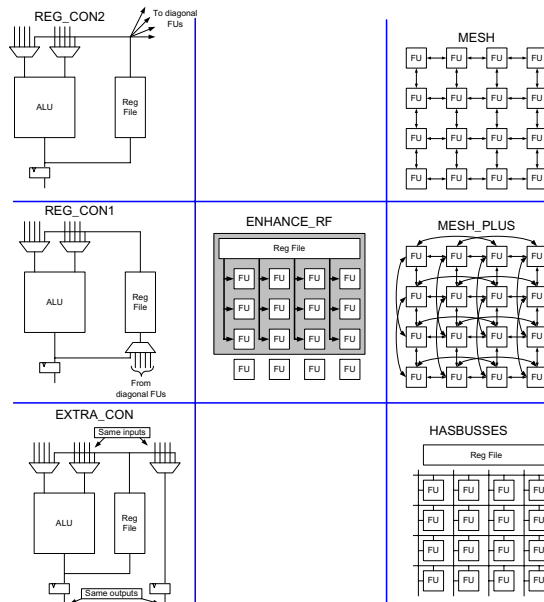
Simulator	IDCT	
	VLIW only (mW)	CGA (mW)
ModelSim	46.5	59.16
Esterel	57.75	65.65
Difference	24.2%	10.9%

depicted in the left column and in the right column of Table 1. The simulations are based on an non-pipelined ADRESv0 with a frequency of 100MHz.

From the results in Table 1 we can conclude that the proposed power simulation methodology with Esterel simulation works better in CGA mode that in VLIW mode. This is due to the fact that the CGA structure is captured by the simulator with more detail in the XML architecture file, while the VLIW part assumes a couple of implicit signals. Therefore the structure of the Esterel simulator and of the VHDL code match better for the CGA, resulting in more accurate signal switching statistics.

4 Architecture Explorations

In this section we propose a variety of architectural options to create an ADRES instance. ADRES explorations were performed in [8] producing scheduling and area results only as [9] provides energy results of a 32-tap Complex FIR filter based on abstract energy models. Different architectures are composed from the architectural options and


Fig. 4. Interconnection Options for Architectural Experiments

are evaluated for power, energy and performance. Finally, an optimized architecture is derived and a detailed power distribution for it is provided. This selection process utilizes two key benchmark kernels from multimedia and wireless application domains: IDCT and FFT. For the IDCT 396 8x8 block calculations are performed. For the FFT a single 1024 points FFT is performed. More benchmarks for evaluation were not available at the moment of writing.

4.1 Architecture Exploration Options

Fourteen different architectures are constructed from 7 different interconnection options as depicted in Figure 4. The architectures are described in the XML architecture file, which will be used in the tool flow described earlier. Exhaustive search of all configurations is impossible due to the large search space and required time.

The simplest interconnection option is *mesh*. It creates connections between destinations and sources of adjacent FUs in horizontal and vertical directions. The *mesh_plus* interconnection is an extension of mesh with additional connections that routes over the neighboring FUs. The *reg_con1* and *reg_con2* options create diagonal connections between neighboring FUs and RFs. This creates additional routing and the possibility of data sharing among FUs connecting directly to the RFs. The difference is that *reg_con1* receives data from its neighbors while *reg_con2* sends data to them. The *extra_con* option offers an extra FU bypass to enable parallel processing and routing. The *enhance_rf* option has shared read and write data ports from the global DRF to the vertically connected FUs as proposed by Kwok et al. [2]. This is beneficial for communication as the global DRF is used as communication medium in the array, however it increases power consumption due to the frequent accesses to the global DRF. Splitting up the power-hungry DRF into smaller, local DRFs was one of the main features of power reduction [6]. The option *has_busses* determines if predicate and data busses of 1 and 32 bits wide respectively are implemented in the design. By combining various interconnection options 14 different architectures are created as noted in Table 2.

Table 2. Architectural Exploration Selection Options

Architecture	mesh	mesh plus	reg con1	reg con2	extra con	enhance RF	has busses
mesh	X						
mesh_plus	X	X					
xtra_con	X	X			X		
reg_con1	X	X	X				
reg_con2	X	X		X			
reg_con_all	X	X	X	X			
enh_rf	X	X				X	
busses	X	X					X
arch_1	X	X		X	X	X	
arch_2	X	X	X		X	X	
arch_3	X	X		X		X	
arch_4	X	X	X			X	
all	X	X	X	X	X	X	X
ref	X	X	X		X	X	X

The physical properties of RFs and FUs are the same for all architectures. The local and global RFs have 16 and 64 words, respectively. The data bus width between DRFs, external memories and FUs is 32-bits. Each FU is capable of regular additions and multiplications of which only the FUs in the VLIW section are capable of load/store operations. The store capabilities of the FUs requires 12 read and 4 write ports for the global DRF. The global PRF has 4 read and write ports. The local DRFs have 2 read and 1 write ports and the local PRFs have 1 read and write port. The configuration memories (CM) are of fixed 128 words depth to store the benchmark configurations for the CGA. The CMs' bus sizes are variable and each FU, RF and multiplexor has its dedicated memory. We use standard 90nm general purpose libraries for both the standard cells and the memories.

The modeled non-pipelined architectures assume zero cache miss data memories and zero miss instruction cache. The external busses, the data memory and the I-cache are not considered in our power estimation study. However, the configuration memory, which serves as an instruction memory for the array mode, is considered in the CM module.

4.2 Instance Selection

The FFT and IDCT benchmarks are used to select the appropriate instance based on power, energy and performance trade-offs. As a comparison base a non-pipelined scalar RISC architecture is modeled as an array of size 1x1 (*np_1x1_reg*).

Figure 5 presents the leakage power results of the architectures. The *4x4_all* has high leakage due to the all interconnect options in Table 2. The Synopsys synthesis tool is not always consistent as the leakage power of *4x4_ref* is significantly reduced compared to *4x4_arch_1*. Figures 6 and 7 present the total power results for the two benchmarks. The order of components in the legends are the same in the charts of which PRF_CGA, PRF_VLIW and VLIW_CU are negligible. The following components are separately measured and divided in CGA and VLIW parts. There is a data register file DRF, predicated register file PRF, functional units FU and configuration memory CM.

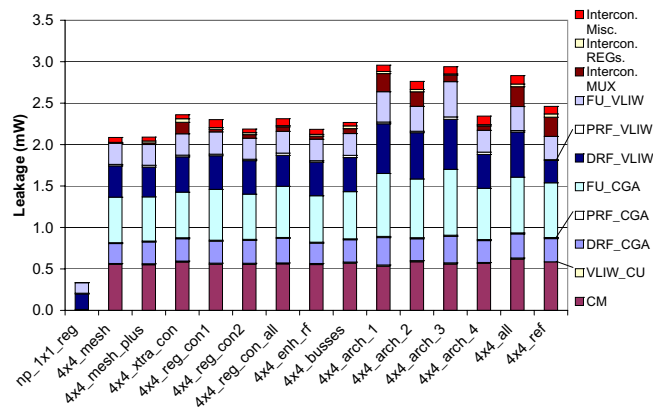


Fig. 5. Leakage Power

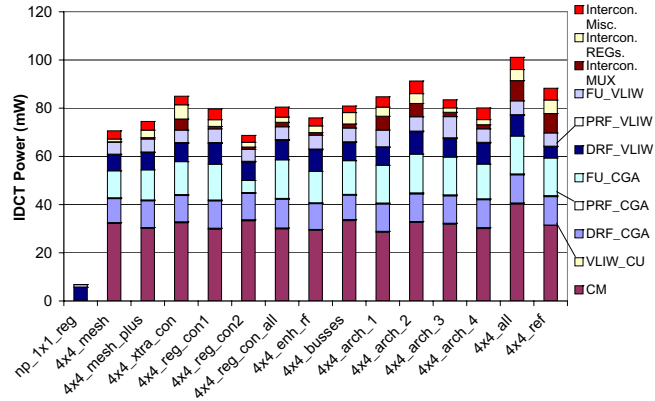


Fig. 6. IDCT Power @ 100MHz

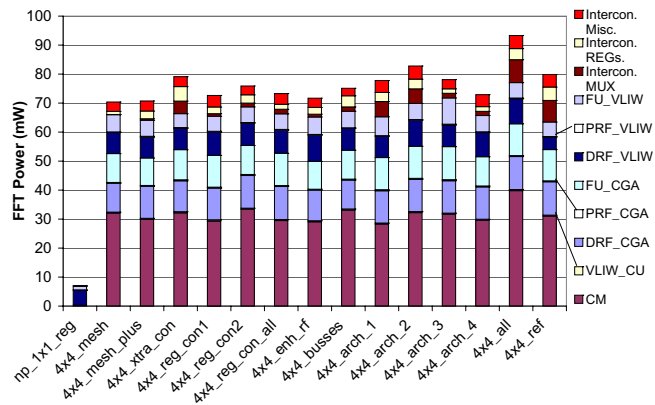


Fig. 7. FFT Power @ 100MHz

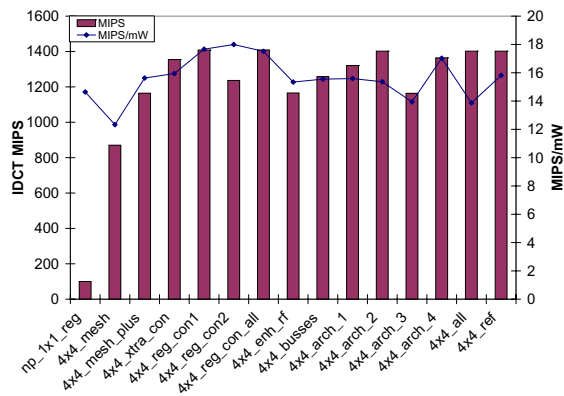


Fig. 8. IDCT Performance @ 100MHz

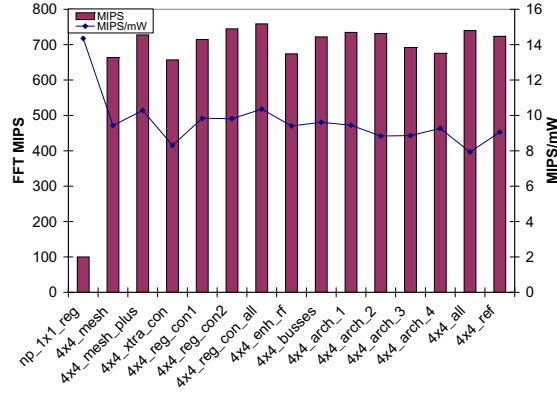


Fig. 9. FFT Performance @ 100MHz

The register files (PRF and DRF) in VLIW mode are global while in CGA mode the sum of all local register files is considered. The *4x4_all* option consumes highest total power compared to the other options mainly due to the large amount of interconnections requiring larger CMs.

We measured the performance in MIPS using the number of executed instructions obtained from the compiled instruction set simulator divided by the total number of execution cycles. The resulting performance charts are depicted in Figures 8 and 9. The *np_1x1_reg* shows a low MIPS result, but yet gives a high MIPS/mW. The performance charts also show that the fully interconnected architecture *4x4_all* is not an efficient option. The bypass feature in *4x4_xtra_con* has some advantage for IDCT only, but certainly not for FFT. This is a typical application that accesses DRFs quite often, which is shown in Figure 9 by *4x4_enh_rf* and *4x4_reg_con_all* having a higher MIPS/mW factor compared to *4x4_xtra_con*. The presence of busses in the design does not give much advantages, since it requires additional multiplexors, larger CMs and data networks, that all increase power.

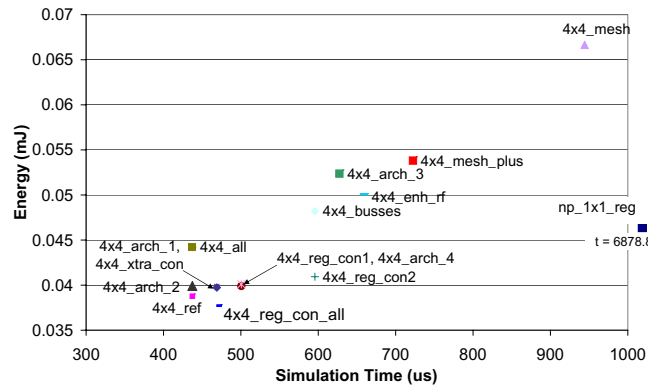


Fig. 10. IDCT Energy-Delay @ 100MHz

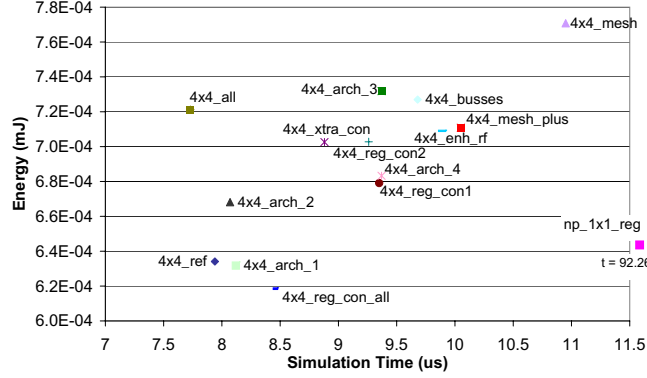


Fig. 11. FFT Energy-Delay @ 100MHz

The power and performance charts are a good indication for selection of the appropriate architecture, however the energy vs runtime shown in Figures 10 and 11 is of more significance.

The *np_1x1_reg* architecture does not have the highest energy consumption, however it is about 8 times slower than the *4x4_mesh* architecture. Of all considered 4x4 arrays the *4x4_mesh* has the highest energy consumption for both benchmarks. The *4x4_reg_con_all* has the lowest energy consumption for both benchmarks, but it is not the fastest option, which is *4x4_all*. The *4x4_ref* architecture is a good alternative between energy and performance results.

We select the architecture with the lowest energy consumption and reasonable performance, which can be seen from the energy-delay charts is *4x4_reg_con_all*. The *4x4_reg_con_all* interconnection architecture consists of a *mesh* and *mesh plus* interconnection topology and diagonal connections via multiplexers between FUs and local and global RFs as depicted in Figure 12.

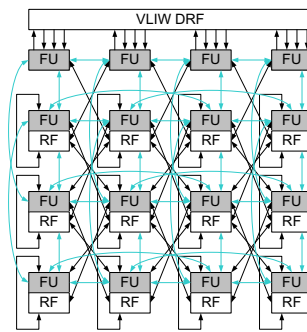


Fig. 12. Proposed ADRES Instance

4.3 Power Breakdown

The power and performance figures of the proposed ADRES instance are summarized in Table 3. The detailed power and area breakdown for IDCT are depicted in Figure 13. The components with *_vliw* postfix are located in the VLIW section and are operational in both VLIW and CGA mode, while those with the *_cga* postfix are located in the CGA section.

Table 3. Characteristics of 4x4_reg_con_all

	Total		MIPS	MIPS/mW	mW/MHz
	Power (mW)	Energy (uJ)			
FFT	73.28	0.619	759	10.35	0.7328
IDCT	80.45	37.72	1409	17.51	0.8045

The power chart in Figure 13(a) shows that the CMs, FUs and DRFs consume most of the power. Interesting to note is the low power consumption of less than 10% of the interconnections between modules as opposed to 10 - 30% that were reported for regular DSP design [7]. The VLIW control unit (cu_vliw) and PRFs consume the least amount of power as these are the smallest components.

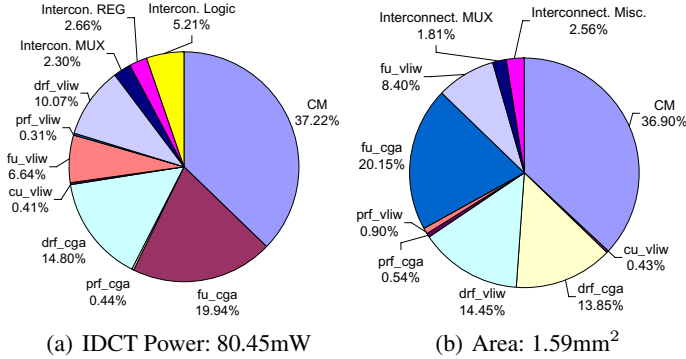


Fig. 13. Power and Area distribution 4x4_reg_con_all @ 100MHz

The area chart in Figure 13(b) shows that the CMs, FUs and DRFs require most area. The CMs have a fixed depth of 128 words during the architectural explorations.

The power measurements of ADRES were based on the core architecture only. To get a power prediction of the complete ADRES processor including instruction cache and data memories we simulated the activities of these blocks. For the IDCT the data memories were activated for 100% of the cycles and the instruction cache for 4.3% respectively. According to the data sheets of SRAM memory models this results in a power consumption of 0.03mW/MHz for the data memories and 0.05 mW/MHz for

the instruction cache. The power consumption of the not activated memory modules is negligible and assumed as zero. The distribution of all the components in the processor for the IDCT benchmark is shown in Figure 14. It shows that in addition to the FUs,

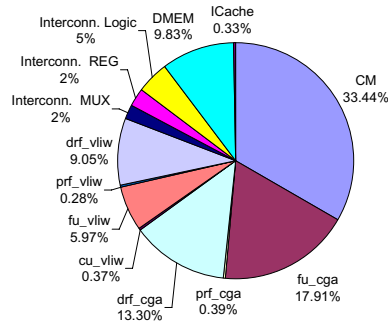


Fig. 14. Overview of ADRES Processor Power Consumption with IDCT

DRFs and CMs, the data memory (DMEM) consumes significant amount of power. The I-cache power consumption is negligible.

The experiments were obtained utilizing the non-pipelined version ADRESv0, however, a pipelined version ADRESv1 is under development at the moment. Preliminary results are encouraging reaching the performance target of 40MOPS/mW. The reconfigurable architecture Morphosys [10] is comparable to ADRES, but does not provide power results of IDCT and FFT. No other comparable architectures are known at the moment of writing.

5 Conclusions

In this paper we explored various architectures for the ADRES coarse-grained reconfigurable array and selected the appropriate one based on power, energy and performance trade-offs. An especially defined methodology and tool flow 1) maps the FFT and IDCT benchmarks on the array using the DRES compiler for high performance, low energy execution, 2) synthesizes each architecture into a front-end design and 3) simulates with either the compiled ISA simulator, ModelSimv6.0a or the Esterel simulator for performance and power evaluations. Power is calculated by annotating the switching activity after Esterel or RTL simulation onto the gate-level design. Our three-fold simulation approach permits to obtain results quickly, which is important for architecture exploration.

Fourteen different ADRES instances were created based on 7 different architectural features and evaluated with the available tool flow. The obtained power, energy and performance charts show that a combination of mesh and mesh plus interconnection topologies with diagonal connections between functional units and local data register files results in good performance of 10.35 - 17.51 MIPS/mW and power of 73.28 - 80.45mW with the least amount of energy 0.619 - 37.72uJ for FFT and IDCT, respectively.

References

1. Bingfeng Mei, Serge Vernalde, Diederik Verkest, Hugo De Man and Rudy Lauwereins: *ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix*, IMEC, 2003, Kapeldreef 75, B-3001, Leuven, Belgium, DATE 2004
2. Zion Kwok and Steven J. E. Wilton: *Register File Architecture Optimization in a Coarse-Grained Reconfigurable Architecture*, University of British Columbia, April 2005, 35–44, Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05) - Volume 00
3. The IMPACT Group. <http://www.crhc.uiuc.edu/Impact/>
4. <http://www-sop.inria.fr/esterel-org/>
5. <http://www.synopsys.com/>
6. Bingfeng Mei: *A Coarse-Grained Reconfigurable Architecture Template and its Compilation Techniques*, Katholieke Universiteit Leuven, Januari 2005, ISBN: 90-5682-578-X
7. Renu Mehra and Jan Rabaey: *Behavioral Level Power Estimation and Exploration*, April 1994, Proc. First International Workshop on Low Power Design, University of California at Berkeley
8. Bingfeng Mei, Andy Lambrechts, Jean-Yves Mignolet, Diederik Verkest and Rudy Lauwereins: *Architecture Exploration for a Reconfigurable Architecture Template*, March 2005, IEEE Design & Test of Computers, IMEC and Katholieke Universiteit Leuven
9. Andy Lambrechts, Praveen Raghavan and Murali Jayapala: *Energy-Aware Interconnect-Exploration of Coarse Grained Reconfigurable Processors*, 4th Workshop on Application Specific Processors (WASP), September 2005
10. Hartej Singh, Ming-Hau Lee, Guangming Lu, Fadi J. Kurdahi and Nader Bagherzadeh: *MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications*, University of California (US) and Federal University of Rio de Janeiro (Brazil), May 2000, 465 – 481, IEEE Transactions on Computers