Power-Efficient Reconfiguration Control in Coarse-Grained Dynamically Reconfigurable Architectures*

Dmitrij Kissler, Andreas Strawetz, Frank Hannig, and Jürgen Teich

Hardware/Software Co-Design
Department of Computer Science
University of Erlangen-Nuremberg, Germany

Abstract. Coarse-grained reconfigurable architectures deliver high performance and energy efficiency for computationally intensive applications like mobile multimedia and wireless communication. This paper deals with the aspect of power-efficient dynamic reconfiguration control techniques in such architectures. Proper clock domain partitioning with custom clock gating combined with automatic clock gating resulted in a 35% total power reduction. This is more than a threefold as compared to the single clock gating techniques applied separately. The corresponding case study application with 0.064 mW/MHz and 124 MOPS/mW power efficiency outperforms the major coarse-grained and general purpose embedded processor architectures by a factor of 1.7 to 28.

1 Introduction

New application areas for embedded systems e.g. mobile multimedia processing, wireless communication, and medical image processing with computational complexity up to several hundreds giga operations per second pose a challenge for modern embedded hardware architectures. Especially for the area of mobile and ubiquitous computing implementations on standard general purpose processors or multicore architectures are not suitable due to still low energy and power efficiency with tens of milliwatts per megahertz combined with a high prime cost. Residing architecturally between the general purpose multicore systems on the one side and ASIC implementation on the other, coarse-grained reconfigurable architectures (CGRAs) tend to inherit the domain specific flexibility from MPSoCs as well as the area and power efficiency from an ASIC implementation. For an overview over the existing CGRAs refer e.g. to [1]. Since up to now increased computational power of several orders of magnitude faces roughly a fourfold battery capacity increase, the flexibility and power/area efficiency are the key features for mobile, battery-powered devices like cell phones, digital still cameras, and other handheld electronics. This paper deals with the aspect of

* The authors wish to thank the Cadence Academic Network and Cadence Design Systems for tool support and assistance.

L. Svensson and J. Monteiro (Eds.): PATMOS 2008, LNCS 5349, pp. 307–317, 2009.

power-efficient dynamic reconfiguration control techniques in CGRAs. To substantially reduce power consumption, we propose to partition clock domains with custom dual clock gating scheme and to combine this with automatic clock gating of the whole design.

2 Related Work

Although there are many academical and commercial coarse-grained reconfigurable architectures, see e.g. overview given in [1], the aspect of applying specially suited power reduction techniques on such architectures has not been covered in depth, so far. Often single figures and breakdowns of power consumption are given, like in [2] or [3], but no attempts were made to exploit architecture-specific features to reach a better power and energy efficiency, besides the configuration cache optimization given in [1]. In this work we combine the custom hierarchical and fine-grained automatic clock gating technique, see Section 3.2, and obtain a considerable increase in power efficiency for our case study CGRA. Although such a combination of clock gating at different levels is already known and widely used in commercial RISC processor cores like e.g. [4], still no detailed experimental results and figures are available. Clock gating is a simple and effective method for decreasing the dynamic power consumption, see e.g. [5] and [6]. On the register transfer level clock gating disables the clock of a component whenever its output values are not used. Clock gating should reduce the power by decreasing the switching activity in the flip-flops, gates in the fan-out of the flip-flops and the clock tree [7]. However, if not used carefully, automatic clock gating can also introduce a significant amount of redundant switching power in the inserted clock gating cells, and the corresponding clock tree buffers. This important fact is also verified in the case study in Section 5.

3 Power Reduction in Coarse-Grained Reconfigurable Architectures

A common architectural structure for reconfiguration management can easily be observed for many coarse-grained reconfigurable architectures. Usually, there is a global or distributed memory where different algorithm configurations are kept. Furthermore, some kind of configuration loading modules are needed to implement certain configuration protocols. They can be as simple as a pair of registers with some multiplexers or quite complex with a set of hierarchical state machines. Since each reconfigurable architecture has to be programmed or configured before performing any computation, two work phases can clearly be distinguished: the *(re-)configuration phase*, and the subsequent *functional processing phase*. A common property for all kind of reconfiguration logic (coarse-grained or even fine-grained like for instance FPGA) is that most of the time it is in the idle state. Even if the reconfiguration is used on a e.g. frame basis for video decoding, the portion of time for reconfiguration will roughly amount to a few percents of the frame processing time. This also holds true for any data flow dominant

applications for example from the area of digital audio and video processing. Regarding the power consumption aspect this observation gives us a hint that it will surely be worth considering to apply power reduction techniques especially for the reconfiguration logic in such coarse-grained reconfigurable architectures.

3.1 Component-Level Power Minimization

Regarding the relative power consumption in a CGRA we can identify the sequential logic modules (memory, register files, single configuration control registers and state machines) as components with the highest fraction of static and also total power consumption, as well as area contribution. Compare e.g. the data for the ADRES 4x4 design [3] accelerating a video processing application, with the relative total power and area contribution of the configuration memory of 37%. Regarding the fact that in CGRAs these components are active only a small percentage of the computation time, they are the ideal objects for module–level power reduction techniques, like for example clock and power gating.

Meanwhile, clock-gating technology in EDA tools has evolved to a state, where it is completely automated and does not break the design flow methodology. These tools automatically find flip-flops that have an enable control and which are implemented with a feedback from output to input and a multiplexer that selects whether the previous output is to be kept or a new input is to be accepted. Thereupon, they exchange these to a clock gate cell and a standard flip-flop. No manual *enable* signals are needed since this control logic is already in the design.

Nevertheless, as already mentioned, one should bear in mind some subtle issues about automatic clock gating, like it will be shown in Section 5. Furthermore, automatic clock gating can only be applied to flip-flops and not to complete macro blocks, like on-chip cache and other types of memory, which normally contribute the most to the power and area consumption. To use the advantages of clock gating also for macro blocks in the design, the custom application knowledge has to be exploited, and the logic has to be partitioned into different clock domains. Customized clock gating statements are therefore inserted for example in the HDL source code which are then translated by the synthesis tool to the clock gating cells of the underlying library. In case of CGRAs it does not even introduce any additional control signals, since such signals already naturally exist to separate the reconfiguration and functional processing phases.

3.2 Dual Hybrid Clock Gating

Regarding the application of clock gating in coarse-grained reconfigurable systems, an important observation is that the configuration and functional processing phase normally do not overlap. Therefore, depending on the underlying hardware architecture, the whole design can readily be partitioned into at least two separate clock domains. We propose to use a dual clocking scheme, like it is schematically depicted in Fig. 1. The idea is to have a separate clock (*config_clk*) for logic performing configuration and another clock (*fu_clk*) for functional modules. Although the use of different clock trees in a design is not new, to our best knowledge it has not been applied to coarse-grained reconfigurable architectures so far, to separate

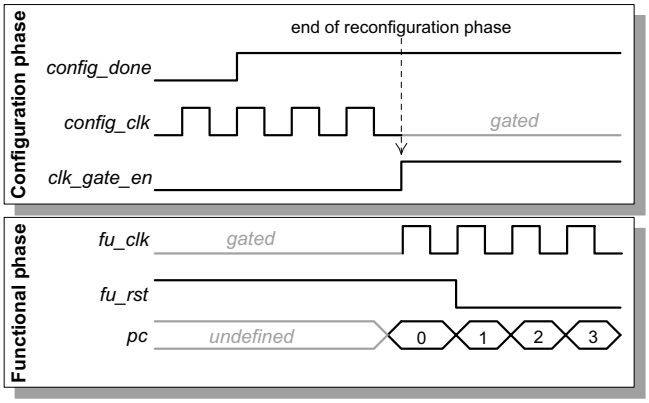


Fig. 1. Dual clock gating scheme with the corresponding functional and configuration phases

the reconfiguration and processing phases physically and gain a substantial reduction in both power and energy. After a global reset and during the configuration phase, *config_clk* is relinquished and the *fu_clk* is gated. After the configuration is done, a *clk_gate_en* signal is asserted gating the *config_clk* and relinquishing the *fu_clk* signal. To ensure the correct behaviour of the functional logic, it is important to assert existing synchronous reset signals for some clock cycles of *fu_clk*. The whole control can be accomplished by the reconfiguration logic itself, so that the control overhead is kept at a minimum.

As opposed to fine-grained automatic clock-gating, this approach is more advantageous regarding the power and energy reduction, since gating the clock trees at the highest hierarchy level prevents a much larger number of clock tree buffers and hard macro blocks from unnecessary toggling.

However, in the following we exemplarily show, that the highest power and energy reduction can be achieved by combining the *custom clock gating* of the reconfiguration logic and additional *automatic clock gating* of the whole design to a *hybrid* clock gating scheme, as discussed in the following sections. To verify feasibility, this method was applied to a coarse-grained reconfigurable architecture and led to more than a threefold power reduction as compared to the single automatic or full custom clock gating techniques, applied on their own.

4 Architecture Description

Our main example of a CGRA consists of arrays of weakly programmable processor elements (WPPEs) each having a VLIW (*Very Long Instruction Word*) structure, see Fig. 2. These PEs have the common property that only a limited amount of instruction memory in each processor element is available and the control overhead is kept as small as possible for a given application domain of algorithms. For example, there is no support for interrupts and exceptions. The instruction

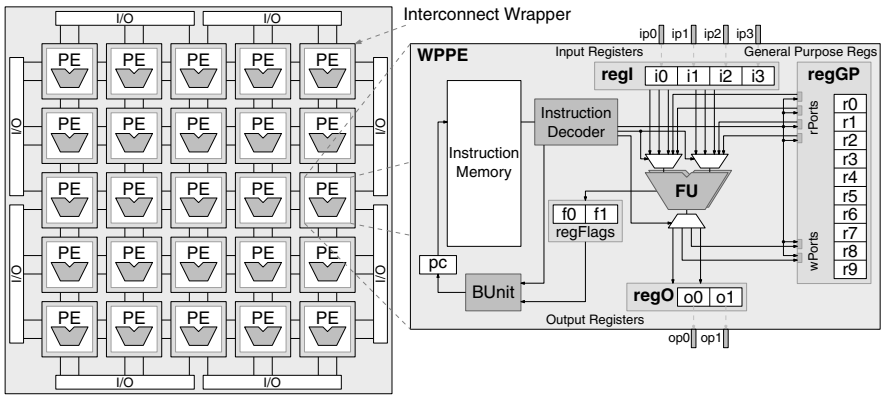


Fig. 2. Example of a weakly programmable processor array (WPPA) with parameterizable processing elements (WPPEs)

memory contains very large instruction word programs. Every single PE can be parameterized at synthesis-time with respect to the number and types of functional units, like adders/subtractors, multipliers, shifters, and modules for logical operations, see Fig. 2. Special functional units such as multiply-accumulate (MAC), barrel shifters, etc. can be specified and added to the architecture template. A parameterizable register file for data is provided to store intermediate computational results. For control signals, a similar parameterizable control register file is used. Furthermore, FIFO buffers (*First In First Out*) at the inputs of each PE are used to store incoming data.

The instruction set of a single processor element is minimized also according to domain-specific computational needs. Since our architecture primarily addresses embedded systems-on-a-chip for computation-intensive algorithms, the instruction set needs to be optimized individually for each application.

In parallel hardware architectures, interprocessor communication plays a very important role. Therefore, the concept of an interconnect module enclosing each processor element is used. It is also used to describe and parameterize capabilities of switching networks. Different user-specified and standard topologies between the single PEs in the array, e.g. torus, 4D hypercube, and others can be implemented and optimized statically but also changed dynamically, see e.g. [8] and [9].

4.1 Reconfiguration Process and Management

A group of VLIW programs for a certain algorithm and the corresponding interconnect scheme (i.e. values of configuration registers, see also [8]) are grouped together to build one so called *setup type*. The global configuration or setup memory (*cfg_mem*) contains several processor array setups for algorithms which can alternatively be processed by the array. The overall architecture for reconfiguration management in the processor array can be seen in Fig. 3. Besides the binary code, each setup contains metadata describing e.g. the processor array tile

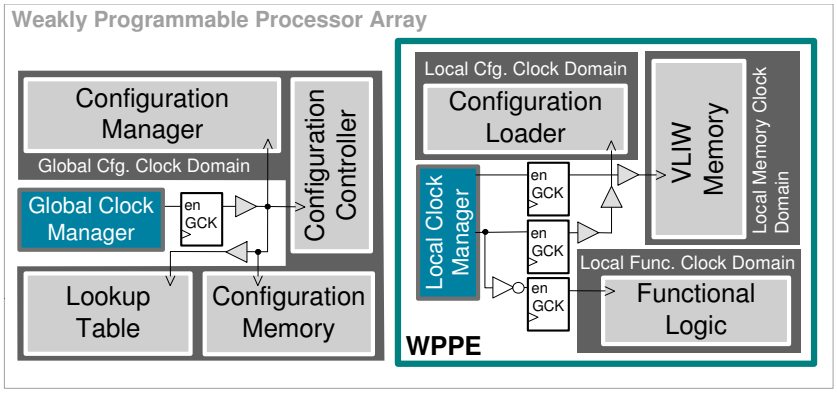


Fig. 3. Reconfiguration management and clock domain partitioning

size and position for current algorithm in form of one or many multicast signatures [8], position of the corresponding binary code in the configuration memory, etc. These metadata are processed by the global configuration manager (*cfg_mgr*) module during an initialization process after a global reset. Among other things, the auxiliary lookup table (*lu_tab*) is filled with different setup numbers and the corresponding configuration memory addresses of the binary code. This is necessary to allow reallocation of binary code in the configuration memory. Thereafter, the configuration manager runs a simple protocol with the global configuration controller (*cfg_ctrl*) providing it with algorithm specific configuration data. Processing elements which should be configured with a new algorithm are selected with the help of one or many algorithm specific multicast signatures. Finally, the local configuration loader modules (*ldr_x_y*) in each selected PE are loading the binary code fetched by the global configuration controller from the configuration memory into their local VLIW instruction memories over a global configuration bus.

4.2 Logic Partitioning for Clock Gating

To apply our proposed hybrid dual clock gating method, the whole array design was partitioned into the following clock domains: the *global configuration clock domain* on the array level, and three *local clock domains* in each PE, see Fig. 3. According to this scheme, on the array level, configuration manager, configuration controller, auxiliary lookup table, and the global configuration memory can all be gated by a single latch-based clock gating cell (GCK). The information about the current configuration status from configuration manager as well as the internal or external configuration requests are used by the global clock manager module to switch this clock domain on or off. On the processing element level, in addition to a local configuration clock domain, a functional clock domain is also needed. Since the configuration and processing phases in CGRAs normally do not overlap, these two clock domains are inversely related to each other and can be controlled by a single *clk_gate_en* signal from Fig. 1 and its inversion,

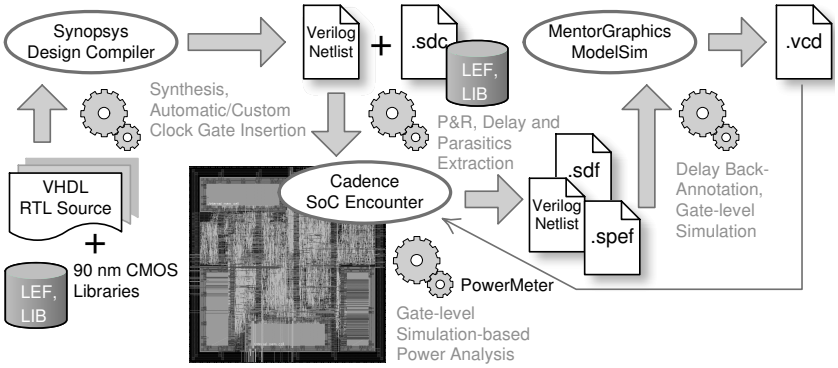


Fig. 4. Case study tool flow overview for power analysis

respectively. A third local clock domain is introduced for the instruction memory, which can be separated from the clock just after it has been configured and before the functional processing phase starts. This intermediate state in single processing elements can last considerably long, if a sufficiently large processor array, e.g. of size 10×10, is being configured with many different setups.

5 Case Study

As an experimental setup for analysis of power savings, two basis algorithms from digital signal processing were chosen and implemented with the help of dynamic reconfiguration on the same WPPA hardware platform, namely an FIR filter and an edge detection (ED) algorithm. ED is a very important task in image processing for both color and grey-scale images. Feature extraction techniques like for instance Hough transform or image segmentation all require edge information as input. The data path should use 16-bit fixed point arithmetics in all cases. A processor array of size 2×2 was instantiated from a template written in VHDL with each processing element parameterized according to Table 1(a). The tool flow used for power analysis is depicted in Fig. 4. At the front end the design was synthesized with the help of Synopsys *Design Compiler* [10] (ver. Y-2006.06) and a 90 nm, 1.0 V supply voltage standard cell library for a 200 MHz maximum target frequency (corresponds to 1024 × 768 XGA resolution, @30 fps, greyscale). Two other possible operational modes are the 125 MHz mode for an SVGA resolution, @30 fps, greyscale and the 75 MHz mode for a VGA resolution, @30 fps, greyscale. For the two lower frequency operational modes no voltage scaling was performed. After place and route using Cadence *SoC Encounter* [11] (ver. 5.20), the gate-level netlist with back-annotated delays was simulated with real image data in *ModelSim* [12] (ver. 5.8e). For accurate power analysis, particulary for the clock tree, the simulator-generated value change dump file (.vcd) was read back and processed by the *SoC Encounter v.7.1* power estimator which is based on the Cadence *PowerMeter* engine, a sign-off power analysis tool. To compare the power consumption of designs with and without different clock gating

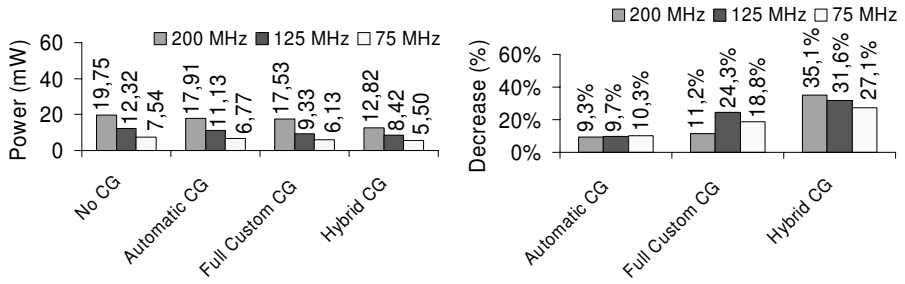
Table 1. Excerpt from the static parameter values of a single WPPE and array (a), and decrease (+) and increase (−) of switching power for different clock gating schemes and 200 MHz target frequency (b)

(a) Static case study parameters for PEs and the array

Parameter	Value
Adders/Element	1
Multipliers/Element	1
Data path width [bit]	16
FU instr. width [bit]	16
Regs. number/Element	5
Instr. memory/Element	16x64
array size	2x2
Config. bus width [bit]	32
Config. memory (ROM)	512x32

(b) Decrease (+) and increase (−) of switching power in reconfiguration modules

Module	Automatic	Full Custom	Hybrid
cfg_mgr	−5.67%	30.71%	1.47%
lu_tab	−272.59%	2.60%	−104.14%
cfg_mem	7.71%	43.06%	39.48%
cfg_ctrl	−70.95%	10.14%	−33.74%
ldr_1_1	−1855.65%	−29.43%	−153.71%
ldr_1_2	−1450.94%	20.75%	−151.19%
ldr_2_1	−1359.73%	−4.87%	−295.47%
ldr_2_2	−1476.15%	11.26%	−196.95%
rc logic	−180.60%	−34.07%	−12.22%



(a) Total power consumption (b) Relative power reduction

Fig. 5. Total power consumption in a 2x2 WPPA for different clock gating (CG) strategies and target frequencies (a), and decrease in percent compared to a not gated design (b).

methods, the above mentioned tool flow was applied to four designs: without any clock gating, with automatically inserted clock gating cells, with custom inserted clock gating cells according to Fig. 3, and finally with combination of the latter two. The testbench contained a configuration phase of the array for the ED algorithm of 4 μs with a subsequent image processing phase of 16 μs. The results can be seen in Fig. 5. The power reduction of automatic and full custom scheme amounts to 9% and 11% for the 200 MHz clock frequency, respectively. Whereas more than a threefold power reduction of 35% is achieved by combining the automatic and full custom clock gating at 200 MHz. While for the automatic clock gating the power reduction hardly varies with clock frequency, it ranges from 11% to 24% and 27% to 35% for the full custom and hybrid cases, respectively. In case of the hybrid scheme the power savings slightly increase with increasing frequency, since more dynamic power can be saved. For all three frequencies the

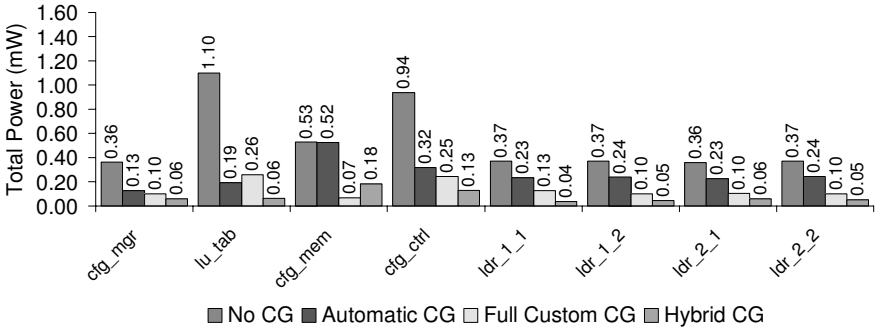


Fig. 6. Total power consumption in reconfiguration modules for different clock gating (CG) strategies (200 MHz target frequency)

Table 2. Statistics for different clock gating strategies

	Area statistics		Register statistics		
Design	Gate area	Overhead	CG elements	Reg. total	Gated reg.
Not gated	0.233 mm^2	0%	0	2767	0%
Automatic	0.225 mm^2	−3.56%	181	2770	79.89%
Full custom	0.239 mm^2	2.85%	9	2788	92.75%
Hybrid	0.226 mm^2	−2.84%	195	2788	99.00%

hybrid approach outperforms the rest. The absolute values for power in different reconfiguration modules are shown in Fig. 6. According to these figures, hybrid clock gating achieves 80% to 90% total power reduction for the reconfiguration logic at 200 MHz. For hybrid clock gating, 99% of registers in the design are gated, which is more than for the automatic (79.89%) and full custom (92.75%) case, see Table 2. Also to note, a small area decrease of approx. 3% for automatic and hybrid gating, like it is usually reported in this case.

So what is the reason for the poor power reduction of automatic clock gating? The answer is shown in Table 1(b).Contrary to the intuitive expectation, automatic clock gate insertion increases the switching activity and switching power of some logic modules up to 18-fold (or 1855%). Automatic clock gating increases the switching power of the reconfiguration logic by 180%, whereas the full custom and hybrid clock gating only by 34% and 12%, respectively. In this case, fine-grained glock gating alone substantially worsen the switching power dissipation. Since the power consumption in memory modules and sequential cells is mostly dominated by internal power and not the switching power, automatic clock gating still achieves a power reduction of 9%, but further improvements are nullified by the switching power increase. In some CAD synthesis tools a *hierarchical* option for automatic clock gating can be selected, meaning that the top design and all subdesigns are processed in one step to identify common enable conditions and insert CG cells on higher hierarchy levels. This option was also tested separately, but gave no remarkable improvements.

6 Conclusions and Future Work

To sum up, we can look at the achieved power efficiency of the proposed WPPA architecture and compare it with some well-established commercial and academical embedded processor architectures, see Table 3. With a power efficiency of 0.064 mW/MHz in a case study image processing algorithm (1024×768 XGA resolution, @30 fps, greyscale) our array outperforms the TMS320C6454 DSP from Texas Instruments (core power) by a factor of 28 and the low-power featured ARM946E-S embedded processor (with cache) by a factor of 1.7 in average. With performance-power efficiency of up to 124 MOPS/mW for a case study FIR and edge detection implementation, it definitely provides an ASIC-like performance, simultaneously offering domain-specific flexibility and reconfiguration features. Our future work will include the study of efficient power-aware design space exploration and compilation techniques for CGRAs and WPPA in particular.

Table 3. Power efficiency overview of some known commercial and academical programmable embedded architectures (90 nm CMOS technology, or scaled to 90 nm*)

Name	Architecture	MHz	mW/MHz	Power Efficiency, Norm.
WPPA	WPPA 2x2	200	0.064	28
ARM	ARM946E-S [13]	230	0.11	16.3
ARC International	ARC EP20 [14]	155	0.11	16.3
Recore Systems	Montium TP [2]	140*	0.17*	10.5*
Silicon Hive	HiveFlex CSP2000 [15]	200	0.24	7.5
Renesas	SH-X2 (LP) [6]	266	0.3	6.0
ADRES	ADRES 4x4 [3]	100	0.7	2.6
Texas Instruments	TMS320C6454 [16]	720	1.79	1.0

References

1. Kim, Y., Park, I., Choi, K., Paek, Y.: Power-conscious Configuration Cache Structure and Code Mapping for Coarse-grained Reconfigurable Architecture. In: Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED), pp. 310–315. ACM, New York (2006)
2. Smit, L., Rauwerda, G., Molderink, A., Wolkotte, P., Smit, G.: Implementation of a 2-D 8x8 IDCT on the Reconfigurable Montium Core. In: Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL), Amsterdam, Netherlands, pp. 562–566. IEEE, Los Alamitos (2007)
3. 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.) ARCS 2007. LNCS, vol. 4419, pp. 1–13. Springer, Heidelberg (2007)
4. Greenhalgh, P.: Power Management Techniques for Soft IP. In: Online Proceedings of the Synopsys Users Group European Conference, p. 12 (2004)

5. Patel, R., Rajgopal, S., Singh, D., Baez, F., Mehta, G., Tiwari, V.: Reducing Power in High-Performance Microprocessors. DAC 00, 732–737 (1998)
6. Yamada, T., Abe, M., Nitta, Y., Ogura, K., Kusaoke, M., Ishikawa, M., Ozawa, M., Takada, K., Arakawa, F., Nishii, O., Hattori, T.: Reducing Consuming Clock Power Optimization of a 90 nm Embedded Processor Core. IEICE Transactions 89-C(3), 287–294 (2006)
7. Pedram, M., Rabaey, J.: Power Aware Design Methodologies. Kluwer Academic Publishers, Norwell (2002)
8. Kissler, D., Hannig, F., Kupriyanov, A., Teich, J.: A Highly Parameterizable Parallel Processor Array Architecture. In: Proceedings of the IEEE International Conference on Field Programmable Technology (FPT), Bangkok, Thailand, pp. 105–112. IEEE, Los Alamitos (2006)
9. Kupriyanov, A., Hannig, F., Kissler, D., Teich, J., Lallet, J., Sentieys, O., Pillement, S.: Modeling of interconnection networks in massively parallel processor architectures. In: Lukowicz, P., Thiele, L., Tröster, G. (eds.) ARCS 2007. LNCS, vol. 4415, pp. 268–282. Springer, Heidelberg (2007)
10. Synopsys, Inc.: RTL Synthesis. Online (2008)
11. Cadence Design Systems, Inc.: SoC Encounter. Online (2008)
12. Mentor Graphics: ModelSim 5.8e. Online (2008)
13. Advanced RISC Machines: ARM946E-S. Online (2008)
14. ARC International: ARC EP20 Core. Online (2008)
15. Silicon Hive: HiveFlex CSP2000 Series, Communication Signal Processor. Online (2008)
16. Texas Instrumets: Data sheet: TMS320C6454 Fixed-Point Digital Signal Processor (Rev. D). Online (2008)