Parallelism Level Impact on Energy Consumption in Reconfigurable Devices

Robin BONAMY, Daniel CHILLET, Olivier SENTIEYS
IRISA - CNRS UMR 6074
INRIA Rennes Bretagne Atlantique
{robin.bonamy, daniel.chillet, olivier.sentieys}@irisa.fr

Sebastien BILAVARN
LEAT - CNRS UMR 6071
University of Nice Sophia Antipolis
sebastien.bilavarn@unice.fr

ABSTRACT

Nowadays, System-on-Chip architectures are composed of several execution resources which support complex applications. As it shares silicon area and limits the cost of the global circuit, the embedding of a reconfigurable resource in these SoC provides flexibility to the hardware. In this case, several implementations of the same algorithm, offering different characteristics, can be considered in order to optimize performances. In general, the tasks mapped on reconfigurable resources are algorithms that can be defined through several levels of parallelism. Clearly, parallelism directly affects the area and the execution time, this paper shows that the energy consumption is not constant, and decreases when the parallelism grows up.

1. INTRODUCTION

With the emergence of FPGAs, it is now easy and relatively cheap to implement a configurable device in a System-on-Chip. Critical tasks can be efficiently implemented to offload the processor within this type of resource [1].

If several hardware tasks can be executed sequentially in order to reduce the necessary silicon area, this type of execution needs specific management to ensure a correct function [2]. This management is generally done by an operating system [3], which has to decide, on-the-fly, the spatio-temporal scheduling of tasks.

A possible way of optimizing the global execution of an application would be to define several implementations for each hardware task. To offer a good trade-off between performances and silicon cost, several parallel hardware implementations can be developed for each task. From these different implementations, the operating system can select the best configuration of each task in order to ensure performances and to limit energy consumption. Several works have been done on code transformations for the reduction of power, energy, area or execution time [4].

This work focuses on energy consumption with various parallelism level implementations of tasks. Using a parallel description of tasks generally does not change the number of computations. Thus, with this transformation, the energy consumption of a specific task execution should stay constant. A hardware task is complex and is not only composed of algorithm computations but also of control instructions which contribute to area and power consumption [5]. Therefore, we assume that the parallelism level modifies energy consumption. This paper presents several results showing that using parallelism is a good way to reduce energy consumption for hardware accelerated tasks.

The results are obtained by measurement on a real FPGA circuit and for three different algorithms and present the trend between the power consumption and the acceleration.

This paper is organized as follows. Section 2 presents the methodology used to perform the measurements while section 3 presents the measures and models. Then we conclude this paper on how our measurements can help the designer during the first design steps.

2. MEASUREMENT PROCEDURE

Since the energy reduction is obtained by reducing the execution time, we assume that the energy versus execution time trend is linear and it may follow

$$E = \alpha + \beta t \tag{1}$$

with α representing the energy needed to perform the computation part of the algorithm and β the energy per time unit varying with the parallelism level.

To confirm this assumption, a measurement campaign of the power consumption of the core of an FPGA device is realized. The Xilinx ML550 board is used for the experiments. This board includes a Virtex-5 XC5VLX50T FPGA and provides power supplies which can be used to monitor the current consumed by the FPGA core.

The easiest way to exploit the parallelism in an algorithm is to perform loop unrolling through high level synthesis [6][7]. C code is used for the description of the applications and *Mentor CatapultC* high-level synthesis tool is chosen to generate the blocks with different loop unrolling.

As first benchmark, we use the matrix multiplication, which is a highly parallelizable function. Because of the three nested loops, this application is representative in terms of loop level parallelism.

3. POWER AND TIMING MODEL

Most of the combinations of the loop unrolling (LU) capacities are generated and tested. The best energy versus execution time values are reported in Table 1. An unrolling index of eight does not correspond to a block eight times faster than a sequential version, essentially because of data fetching.

Fig. 1 presents the energy consumed by the hardware block versus the execution time. This figure shows that applying parallelization using loop unrolling trans-

Table 1: Execution time for different loop unrolling indexes

Loop unrolling versions	1,1,1	1,1,8	1,1,16	1,8,32	4,4,32
Execution Time (ms)	1.04	0.51	0.47	0.49	0.39
Area (slices)	112	136	153	555	661
Energy (mJ)	61.99	27.34	27.55	27.48	31.58

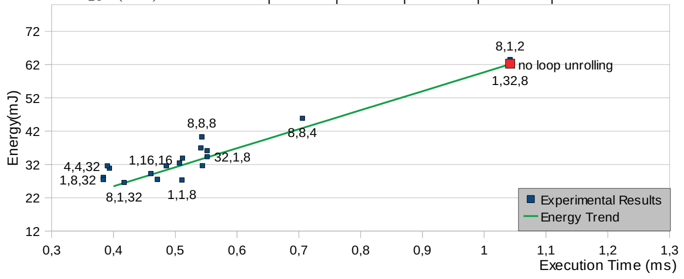

Figure 1: Energy of the matrix multiplication hardware accelerated block versus the execution time. Matrix size is $M = N = 32$. LUIs are separated by a comma to identify the index for each loop: LUI_1, LUI_2, LUI_3. The equation of the model is Eq. (2).

formations does not imply a constant energy but varies linearly with the execution time.

The scatter plot of energy versus execution time shown on Fig. 1 clearly shows a linear trend. This trend is explained as follows: two different sources of power consumption can be isolated, the first one is due to the operators performing the computation and a second part is due to the control of loops and data fetching. The total energy consumed by the computation of the algorithm is roughly the same independently of the parallelism level. Conversely, the required power consumption for the control part is almost constant, so its energy is time dependent. Thus, we propose a model to follow the trend of energy versus execution time using a linear fit. With the sequential version (version (1,1,1) of the table 1) for reference, we can establish a trend of energy versus execution time achieved through loop unrolling for other algorithms. Normalizing time and energy, the equation is the following

$$E \quad = \quad \alpha \times E_0 + \beta \times \frac{E_0}{t_0} \times t \qquad (2)$$

where E is the total energy in mJ and t the execution time in ms, t_0 is the execution time of the algorithm measured without loop unrolling, E_0 the energy consumed by this sequential implementation, $\alpha = \frac{2.64}{62}$ and $\beta = \frac{59}{62}$.

The trend of energy consumption versus execution time is an affine function where the coefficients α and β are dependent of the application performance. These coefficients are evaluated using measurement of the sequential version of an algorithm.

This trend is validated on two other algorithms, motion estimation and Deblocking Filter for video coding applications, measurement are reported in Table 2. We notice that hardware implementation is 28 times faster and is 8.91 times more energy efficient than software execution and that parallelized versions show energy gain of 2.26 versus sequential one for matrix multiplication.

4. CONCLUSION

Basing ourselves on high level synthesis with loop unrolling, this paper shows that the energy is not constant when executing different parallelized solutions and that the trend is linear, depending of execution time. The results show that the energy decreases when the parallelism level increases. To improve energy efficiency, the designer should expand loops and use available space of the system.

Model accuracy can be improved by decomposing tasks in different parts involving control, data processing, memory accesses and parallelization capabilities. This future work will be based on our high-level compiler infrastructure, GeCoS [10], which offers a complete compilation framework for source-to-source transformations.

The main goal of this work is to procure a set of area, time and energy performance values of given tasks to find a good implementation in a reconfigurable device. This work will be extended to include the partial dynamic reconfiguration to reach defined goals: energy savings and increasing performance in system-on-chip.

This work is funded by the French National Research Agency in the scope of the Open PEOPLE project [11].

Table 2: Comparison of execution time and energy between three different implementations of each algorithm.

	Matrix mult.		Full Search		Deblock. filter	
	Time	Energy	Time	Energy	Time	Energy
Soft (ms, mJ)	10.75	244.79	0.4786	18.2	0.5742	26.09
HardS (ms, mJ)	1.04	61.99	0.0369	2.01	0.0529	3.68
HardP (ms, mJ)	0.38	27.48	0.0246	1.05	0.0417	2.74
Soft/HardP **Ratio**	**28.29**	**8.91**	**19.37**	**17.33**	**13.77**	**9.52**
HardS/HardP **Ratio**	**2.74**	**2.26**	**1.50**	**1.91**	**1.26**	**1.34**

Soft represents an execution on the microblaze core,
HardS represents a sequential implementation of the hardware task,
HardP represents the best paralleled solution in terms of time.

5. REFERENCES

[1] Altera Corporation, AN 531: Reducing Power with Hardware Accelerators, 2008

[2] Pao-Ann Hsiung, Exploiting Hardware and Software Low power Techniques for Energy Efficient Co-Scheduling in Dynamically Reconfigurable Systems, Field Programmable Logic and Applications, FPL, Amsterdam, August 2007

[3] Kwok T.T.-O. et al., Practical design of a computation and energy efficient hardware task scheduler in embedded reconfigurable computing systems, Parallel and Distributed Processing Symposium. IPDPS, Rhodes Island, April 2006

[4] Q. Liu et al., Combining Optimizations in Automated Low Power Design in Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10, Dresden, Germany, March 2010

[5] K. Srikanth et al., The Impact of Loop Unrolling on Controller Delay in High Level Synthesis, Design, Automation & Test in Europe Conference & Exhibition, DATE '07, Nice, France, April 2007

[6] K.K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley, pp. 119–147, 1999

[7] B. Buyukkurt et al., Impact of Loop Unrolling on Area, Throughput and Clock Frequency in ROCCC: C to VHDL Compiler for FPGAs, In International Workshop on Applied Reconfigurable Computing, Delft, The Netherlands, March 2006

[8] Optimizing Impulse C Code for Performance, Application Note IATAPP-102, `http://www.impulseaccelerated.com/AppNotes/`

[9] Mentor, Catapult C Synthesis, `http://www.mentor.com/products/esl/high_level_synthesis/catapult_synthesis/`

[10] GeCoS - Generic Compiler Suite, `http://gecos.gforge.inria.fr/`

[11] Open-PEOPLE - Open-Power and Energy Optimization PLatform and Estimator, `http://www.open-people.fr/`