

Rapid Energy Estimation of Computations on FPGA based Soft Processors

Jingzhao Ou and Viktor K. Prasanna

Department of Electrical Engineering, University of Southern California

Los Angeles, California, 90089-2560 USA

Email: {ouj, prasanna}@usc.edu

Abstract—FPGA based soft processors are an attractive option for implementing embedded applications. As energy efficiency has become a key performance metric, techniques that can quickly and accurately obtain the energy performance of these soft processors are needed. While low-level simulation based on traditional FPGA design flow is too time consuming for obtaining such energy performance, we propose a methodology based on instruction level energy profiling. We first analyze the energy dissipation of various instructions. An energy estimator is built using this information. To illustrate the effectiveness of our approach, the energy performance of several FFT and matrix multiplication software programs running on a state-of-the-art soft processor is evaluated using the estimator. Compared with the results obtained through low-level simulation, an average estimation error of 5.9% is observed in our experiments.

I. INTRODUCTION

There is a strong trend towards integrating FPGA with various heterogeneous hardware components, such as RISC processors, embedded multipliers, memory blocks (e.g. BRAMs in Xilinx FPGAs), etc. Such integration leads to a reconfigurable System-on-Chip (SoC) platform, an attractive option for implementing many embedded systems. Soft processors are RISC processors realized using the configurable resources available on FPGA devices. Examples of such processors include MicroBlaze and PicoBlaze from Xilinx [16], and Nios from Altera [1]. The use of such soft processors offers new design trade-offs by time sharing the limited hardware resources available on the target device. The design of a GSM down converter using MicroBlaze is shown in [8]. This design requires reduced amount of hardware resources and is able to achieve considerable energy reduction (by fitting the design into a smaller device) compared with a traditional FPGA implementation.

Energy efficiency is a key performance metric in the design of embedded systems. In order to exploit the advantages offered by soft processors, techniques that can quickly and accurately obtain the energy dissipation of the software programs executing on them are needed. While there are several energy estimation tools for ASIC style processors (see Section II), to the best of our knowledge, no prior work has addressed rapid energy estimation for FPGA based soft processors.

Since soft processors are implemented using FPGA resources, energy estimation of the software programs running on them can be obtained by following the traditional FPGA design flow. For example, for the MicroBlaze soft processor configured on Xilinx FPGAs, the user can perform register transfer/gate level timing simulation using ModelSim [7]. Then, XPower [16] can be used to analyze the simulation record files (.vcd files, which are generated by ModelSim and record the switching activities of all the logics and wires on the FPGA device) and calculate the energy dissipation. While such low level simulation can obtain accurate energy estimates, they are extremely time consuming. For a complex number 8-point

floating-point FFT software program (see Section IV-B), the actual execution time of the software program on a MicroBlaze soft processor operating at 50 MHz is ~ 2.66 ms. Low level simulation using ModelSim 5.7 on a Windows machine with dual Intel Xeon 800 MHz CPUs and 2 Gbyte memory takes around ~ 6 hours and generates an ~ 8.5 Gbyte simulation record file. Using XPower to analyze the generated simulation record file and calculate the energy dissipation requires an additional ~ 3 hours. The time and storage requirements of energy estimation through such low level simulation can prove to be overwhelming for many software programs.

In this paper, we propose a methodology that can rapidly estimate the energy dissipation of software programs running on FPGA based soft processors. Our approach can obtain the energy dissipation of the FFT program discussed above within minutes, a significant speed-up compared with the traditional approach based on register-transfer level simulation. Besides, for the four FFT and matrix multiplication software programs considered in Section IV, our approach leads errors in the range of 3.2% to 7.8% and 5.9% on average.

The paper is organized as follows. Section II discusses related work. Section III presents our methodology for rapid energy estimation of computation on MicroBlaze, a state-of-the-art soft processor configured on Xilinx FPGAs. The energy profiling of the various instructions of the MicroBlaze soft processor as well as the energy estimation of several FFT and matrix multiplication software programs running on the processor are shown in Section IV. We conclude in Section V.

II. RELATED WORK

Previous research has focused on estimating the energy dissipation of a few popular commercial and academic ASIC processors. Sinha *et al.* [12] propose *JouleTrack*, a web based tool for software energy profiling. After analyzing the energy dissipation of the instruction set of StrongARM SA-1100 and Hitachi SH-4 processors, they conclude that “the common overheads (such as cache, decode logic, etc.) in these processors are large and overshadow any instruction specific variations. Therefore, estimating software energy dissipation with an elaborate instruction trace and inter-instruction is overkill.” In contrast, we show in Section IV-A that there are significant variations in the energy dissipation of the instructions of soft processors. Thus, the first order and second order models proposed in [12] cannot guarantee accurate energy estimates for soft processors. Other energy estimation frameworks for processors are proposed in [2] and [17]. They use much more detailed energy models to capture the various sources of energy dissipation on processors, such as the switching capacitance table used in [17]. As soft processors are configured on FPGA and mostly consist of LUTs (Look-up Tables) with similar architectures, using these detailed energy models would turn out to be a *overkill*. Our approach is similar to the instruction level energy profiling framework proposed in [6]. The inter-influence of different components of the

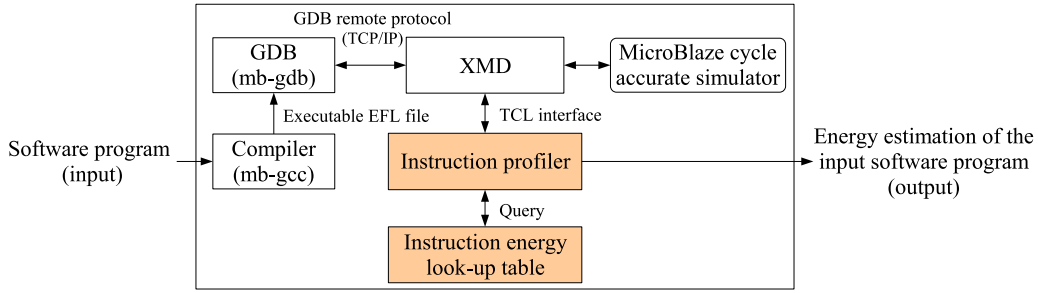


Fig. 1. Design flow of the proposed methodology

processor is ignored in our approach as FPGAs consist of similar basic units (e.g. slices in Xilinx FPGAs). By utilizing low-level simulation tools, we are able to obtain a more detailed analysis of the energy performance of the instructions.

For special purpose architectures implemented on FPGAs, we propose a domain specific approach in [3]. It has been used to estimate energy dissipation of various designs [4].

Performance of soft processors has also been addressed recently. Shannon *et al.* [13] introduce *SnoopP*, which is a non-intrusive, real-time timing profiling tool for MicroBlaze. However, energy performance of soft processors is not addressed in their work and is the focus of this paper.

III. METHODOLOGY

A. Design Flow

The design flow of the proposed methodology is shown in Figure 1. While our technique can be applied to other soft processors, Xilinx MicroBlaze is used to demonstrate the design process since it is widely available. We use the *gcc* compiler for MicroBlaze, the GDB (GNU debugger) and XMD debugger, and the MicroBlaze cycle accurate simulator provided by Xilinx EDK (Embedded Development Kit) [16]. Our contributions are the creation of the instruction energy look-up table that stores the energy dissipation of each instruction in the instruction set, as well as the instruction profiler that integrates all these tools and outputs the energy dissipation of the input software program. Both of the two contributions are highlighted as shaded boxes in Figure 1. Besides, as required by the MicroBlaze cycle accurate simulator, the configurations of the soft processor, its memory systems and other peripherals are pre-determined and are shown in Figure 2. Since MicroBlaze supports split bus transactions, the instructions and the data of the user software programs are stored in the dual port BRAMs and are made available to the MicroBlaze processor through two separate LMB interface controllers. The MicroBlaze processor and the two LMB interface controllers are required to operate at the same frequency. Under this setting, a fixed latency of one clock cycle is guaranteed for the soft processor to access data through these two controllers.

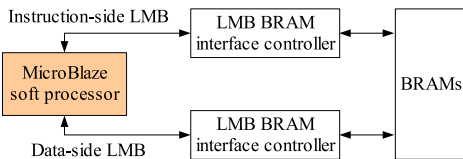


Fig. 2. Configuration of the MicroBlaze processor system

The GNU compiler for MicroBlaze, *mb-gcc*, is used to compile the software program and generate an ELF (Executable and Linking Format) file, which can be downloaded to and

executed on the MicroBlaze processor. This ELF file is then provided to the GNU debugger for MicroBlaze, *mb-gdb*.

The *Xilinx microprocessor debugger* (XMD) is a tool from Xilinx [16] for debugging programs and verifying systems using the PowerPC (Virtex-II Pro) or MicroBlaze microprocessors. It contains a GDB interface. Through this interface, XMD can communicate with *mb-gdb* using TCP/IP protocol and get access to the executable ELF file that resides in it. XMD also integrates a built-in cycle accurate simulator that can simulate the execution of instructions within the MicroBlaze processor. The simulator assumes that all the instructions and program data are fetched through two separate local memory buses (LMBs). Currently, simulation of the MicroBlaze processor with other configuration, e.g. different bus protocols and other peripherals, are not supported by this simulator.

XMD has a programmable TCL (Tool Command Language) [14] interface. By communicating with XMD through the TCL interface, the instruction profiler obtains the numbers and the types of instructions executed on the MicroBlaze processor. Then, the profiler queries the instruction energy look-up table to find out the energy dissipation of each instruction executed. By summing up all the energy values, the energy dissipation of the input software program is obtained.

B. Generation of Instruction Energy Look-up Table

The flow for generating the instruction energy look-up table is shown in Figure 4. We created sample software programs that represent each instruction in the MicroBlaze instruction set by embedding assembly code into the sample C programs. For the embedded assembly code, we let the instruction of interest repeatedly executed for a certain amount of time with more than 100 different sets of input data and under various execution contexts. The MicroBlaze processor has a *PC_EX* bus which displays the value of the program counter and a *VALID_INSTR* signal which is high when the value on the *PC_EX* bus is valid. By performing behavioral simulation and checking the status of *PC_EX* and *VALID_INSTR*, we can determine the intervals over which the instructions of interest are executed. Then, ModelSim records the gate-level switching activities of the device during these intervals and generates simulation record files (.vcd files). In the last step, XPower is used to analyze the design files (.ncd files) and the simulation record files. Finally, the energy dissipation of the instructions is obtained from the XPower output.

Note that the MicroBlaze soft processor is shipped as a register-transfer level IP core. Its time and energy performance are independent of its placement on the FPGA device. Moreover, the instructions are stored in precompiled memory components. The locations of the memory blocks have negligible impact on the energy dissipation for accessing the instructions and data stored in the BRAMs. Therefore, if the MicroBlaze

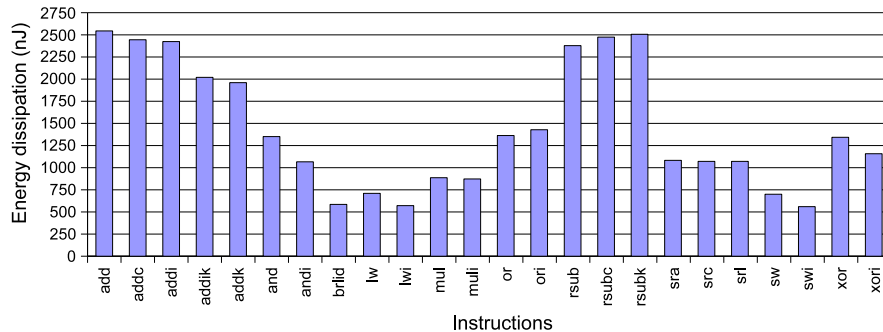


Fig. 3. Energy profiling of the MicroBlaze instruction set

system is configured as in Figure 2, the instruction energy look-up table can be used for energy estimation, regardless of the actual placement of the system. Thus, there is no need to incur the time-consuming low-level simulation every time the table is used for estimation in a design.

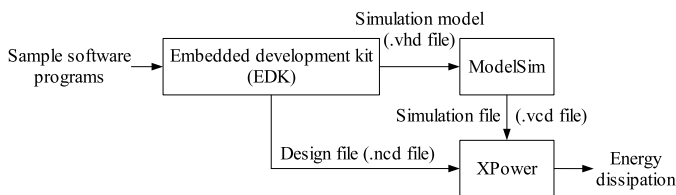


Fig. 4. Flow of instruction energy profiling

IV. EXPERIMENTAL RESULTS

The MicroBlaze based system shown in Figure 2 is configured on a Xilinx Virtex-II Pro xc2vp7ff672-6 device, which integrates dedicated 18bit \times 18bit multipliers and BRAMs. We configured MicroBlaze to use three dedicated multipliers for the multiplication instructions (e.g. *mul* and *muli*) and use the BRAMs to store the user programs. According to the requirement of the cycle-accurate simulator, the operating frequencies of the MicroBlaze processor and the two LMB interface controllers are set at 50 MHz.

For the experiments discussed in the paper, we use EDK 6.1.02 for describing the hardware designs and compiling the software programs, ISE 5.2.03 [16] for synthesis and implementation, and ModelSim 5.7 [7] for simulation. The functional correctness of the design was verified on an ML300 Virtex-II Pro prototyping board [16].

A. Energy Profiling of Instructions

Figure 3 shows the energy profiling of various MicroBlaze instructions obtained using the methodology described in Section III-B. We consider the energy dissipated by the complete system shown in Figure 2, which includes the MicroBlaze processor, two memory controllers and the BRAMs. Instructions for addition and subtraction have the highest energy dissipation while memory access instructions (both load and store) and branch instructions have the lowest energy dissipation. A variance of 486% in energy dissipation is observed for the entire instruction set, which is much higher than the 38% variance for both StrongARM and Hitachi SH-4 processors reported in [12]. These data justify our motivation for taking an approach based on instruction-level energy profiling to build the energy estimator for soft processors, rather than using the first order and the second order models proposed in [12].

Note that the addition and subtraction instructions (e.g. *add*, *addi*, *sub*, *rsub*, *rsubk*, etc.) dissipate much more energy than the multiplication instructions. This is because in our configuration of the soft processor, addition and subtraction are implemented using “pure” FPGA resources (e.g. look-up tables) while multiplication instructions are realized using the embedded multipliers available on Xilinx Virtex series FPGAs. Our work in [4] shows that the use of these embedded multipliers (instead of slice-based ones) can substantially reduce the energy dissipation of the designs.

We have also analyzed the impact of different addressing modes on the energy dissipation of the instructions. This is shown in Figure 5. Compared with instruction decoding and the actual computation, different addressing modes contribute little to the overall energy dissipation of the instructions and thus their effect can be ignored to speed up the energy estimation process.

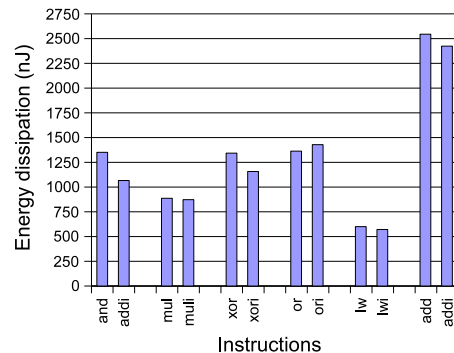


Fig. 5. Impact of different instruction addressing modes

B. Energy Estimation of Example Software Programs

In order to demonstrate the effectiveness of our approach, we analyze the energy performance of two FFT software programs and two matrix multiplication software programs running on the MicroBlaze processor system configured as shown in Figure 2. Details of these four software programs can be found in [11].

FFT and matrix multiplication are widely used in embedded signal processing systems, such as software defined radio [9]. For floating-point computation, there are efforts in supporting basic operations on FPGAs, e.g. addition/subtraction, multiplication, division, etc. [5] [10]. These implementations require a significant amount of FPGA resources and involve considerable efforts to realize floating-point FFT and matrix multiplication on FPGAs. However, floating-point FFT and

TABLE I
ENERGY DISSIPATION OF THE FFT AND MATRIX MULTIPLICATION SOFTWARE PROGRAMS

Program	Number of cycles	Measured energy	Estimated energy	Error	Comment
FFT	97766	212.45	197.35	7.1%	8-point, complex floating-point data
	16447	33.19	32.13	3.2%	8-point, complex integer data
Matrix Multiplication	9877	21.44	20.50	5.5%	3×3 matrix, real floating-point data
	9344	16.33	15.05	7.8%	3×3 matrix, real integer data

matrix multiplication can be easily implemented on soft processors through software emulation.

By time sharing the resources, designs using soft processors usually demand a smaller amount of resources than the corresponding designs with parallel architectures. Thus, they can be fit into smaller devices. The complete MicroBlaze system occupies 576 (11%) of 4928 available slices, 3 (6.8%) of 44 available dedicated multipliers on the target device. The floating-point FFT program occupies 23992 bits while the floating-point matrix multiplication program occupies 3976 bits. All four software programs can be fit into 2 (6.8%) of 44 available 18-Kbit BRAMs. While parallel architectures reduce the latency for floating-point matrix multiplication, they require 33 to 58 times more slices (19045 and 33589 slices) than the MicroBlaze design [18]. As quiescent power accounts for increasingly more percentage of the overall power consumption on modern FPGAs [15], choosing a smaller device can effectively reduce the quiescent energy.

The energy dissipation of the FFT programs and the matrix multiplication programs estimated using our technique is shown in Table I. All these estimates were obtained within few minutes. We compare the estimated data against the measured data obtained through low-level simulation. An average estimation error of 5.9% are observed.

We further analyze the variation of the power consumption of the soft processor during the execution of the floating-point FFT program. First, we use ModelSim to generate a simulation record file every 13.3293 μ sec (1/200 of the total simulation time). Then, XPower is used to measure the average power consumption of each period represented by these simulation record files. The results are shown in Figure 6. The regions between the lines represent the energy dissipation of the MicroBlaze processor, the instruction access cost and the program data access cost (both through the LMB bus).

For the above floating-point FFT software program, the MicroBlaze processor dissipates around 3 times as much energy as that for accessing the instructions and data stored in the BRAMs. Most importantly, this result shows significant fluctuations in the power consumption of the entire system. This is consistent with the large variations in the energy profiling of the MicroBlaze instruction set shown in Figure 3. As different instructions are executed in each sampling period, the large differences in energy dissipation among the instructions would result in significant variation in the average power consumption in these sampling periods.

V. CONCLUSION

A methodology for rapid energy estimation of software programs running on soft processors was proposed. We built an energy estimator for a popular soft processor using the proposed methodology. Using four integer and floating-point software programs, we demonstrated that our approach can rapidly and accurately obtain the energy dissipation of computations on soft processors.

A major limitation of our methodology is that we relied on the built-in cycle-accurate simulator provided by XMD

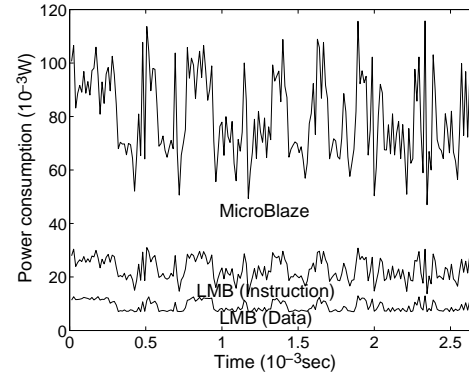


Fig. 6. Power consumption of the FFT program

to obtain the instruction profiling information of the software programs. Since this built-in simulator can only be applied to the configuration of the system as shown in Figure 2, we are unable to capture the energy dissipation of other bus protocols (e.g. the three types of IBM CoreConnect buses). Also, the impact of other peripherals and user IP cores attached to the soft processors on energy performance is not addressed by our approach. We are working on replacing the built-in simulator with the hardware assisted profiling framework proposed in [13] in order to overcome this limitation. Besides, we are also extending our work to provide confidence interval information of the energy estimates.

REFERENCES

- [1] Altera, Inc., <http://www.altera.com>.
- [2] D. Brooks, V. Tiwari, M. Martonosi, "Watch: A Framework for Architectural-Level Power Analysis and Optimizations," *International Symposium on Computer Architecture (ISCA)*, 2000.
- [3] S. Choi, J.-W. Jang, S. Mohanty and V. K. Prasanna, "Domain-Specific Modeling for Rapid Energy Estimation of Reconfigurable Architectures," *Journal of Supercomputing*, Kluwer, 2004.
- [4] S. Choi, R. Scrofano, V. K. Prasanna, and J.-W. Jang, "Energy-Efficient Signal Processing Using FPGAs," *ACM FPGA*, 2003.
- [5] G. Govindu, L. Zhuo, S. Choi, V. K. Prasanna, "Analysis of High-Performance Floating-Point Arithmetic on FPGAs," *RAW*, 2004.
- [6] H. Mehta, R. M. Owens, M. J. Irwin, "Instruction Level Power Profiling," *ICASSP*, 1996.
- [7] Mentor Graphics, Inc., www.mentor.com.
- [8] "MicroBlaze Application Notes," *Programmable World*, 2003.
- [9] J. Mitola, "The Software Radio Architecture," *IEEE Comm. Mag.*, 1995.
- [10] Nallatech, Inc., <http://www.nallatech.com>.
- [11] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, "Numerical Recipes in C: The Art of Scientific Computing (Second Edition)," *Cambridge University Press*, 2002.
- [12] A. Sinha and A. Chandrakasan, "JouleTrack: A Web Based Tool For Software Energy Profiling," *Design Automation Conf. (DAC)*, 2001.
- [13] L. Shannon, P. Chow, "Using Reconfigurability to Achieve Real-Time Profiling for Hardware/Software Codesign," *ACM FPGA*, 2004.
- [14] TCL Developer Exchange, <http://www.scriptics.com>
- [15] T. Tuan and B. Lai, "Leakage Power Analysis of A 90nm FPGA," *IEEE Custom Integrated Circuits Conference (CICC)*, 2003.
- [16] Xilinx, Inc., <http://www.xilinx.com>.
- [17] W. Ye, N. V. Krishnan, M. Kandemir, and M. J. Irwin, "The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool," *Design Automation Conference (DAC)*, 2000.
- [18] L. Zhuo and V. K. Prasanna, "Scalable and Modular Algorithms for Floating-Point Matrix Multiplication on FPGA," *IPDPS*, 2004.