Operand-Value-Based Modeling of Dynamic Energy Consumption of Soft Processors in FPGA

Zaid Al-Khatib(✉) and Samar Abdi

Electrical and Computer Engineering, Concordia University, Montreal, Canada
z_alk@live.concordia.ca, samar@ece.concordia.ca

Abstract. This paper presents a novel method for estimating the dynamic energy consumption of soft processors in FPGA, using an operand-value-based model at the instruction level. Our energy model contains three components: the instruction base energy, the maximum variation in the instruction energy due to input data, and the impact of one's density of the operand values during software execution. Using multiple benchmarks, we demonstrate that our model has only 4.7% average error and 12% worst case error compared to the reference post-place-and-route simulations, and is more than twice as accurate as existing instruction-level models.

Keywords: Energy modeling · Soft processors · Power estimation

1 Introduction

Processor core energy consumption is a first order metric for FPGA-based embedded system design, due to its direct impact on battery life. Early and accurate modeling of soft-processor performance and energy consumption, for a given application, is needed to perform early design space exploration with reasonable confidence. A fine-grained and rapid estimation tool is also important for reconfiguration and customization of system architecture as well as optimization of software implementation for energy consumption.

Modeling the dynamic energy consumption of a soft processor remains a major challenge. Power measurement techniques typically used with ASIC processors, as in the work presented by Bazzaz *et al.* [1] cannot be adopted to measure the power dissipated in the FPGA resources implementing the soft processor core, independent of the other FPGA components. Hence, low level post-place-and-route models and simulations are preferred as they enable capturing the switching activity of the final FPGA logic resources [2]. However these are incredibly slow, thereby making the design space exploration and SW optimization process impractical. Alternatively, energy models based on processor power states are used for quick estimates but they can be very inaccurate. Instruction-level models promise higher accuracy than mode-based models, however, it is very difficult to accurately characterize the energy consumption of individual instructions in soft processors implementations. In fact, conventional methods used in previous work to isolate the energy consumed by an instruction examine its execution under very unique states that do not accurately

K. Sano et al. (Eds.): ARC 2015, LNCS 9040, pp. 65–76, 2015.
DOI: 10.1007/978-3-319-16214-0_6

reflect the energy it would consume when it is executed as part of a normal application run. Our goal is to build a concise instruction-level model that can provide fast and accurate estimates of dynamic energy consumption for soft processors in FPGA. Using a heuristic approach, our model accounts for both the inter-instruction effects and the operand values of the instructions for estimating the energy consumption.

Previous work suggest insignificant impact of operand values on processor power dissipation in ASIC [3]. However, we observed that the operand values greatly impact the energy consumed by soft processor cores in FPGA. This is due to the fact that Soft-processor data-path units are implemented on several Configurable Logic Blocks (CLBs) and DSPs. This requires the operand values propagate through much longer routes between these block than in ASIC, giving them higher charge capacitance. These signals are also frequently reset to Zero by instructions like the NOP and small operand operations [4]. As such, there's an increased probability of signals switching after the execution of instructions operating with values containing many ones. The higher density of ones in an instructions operands, the larger the probability of switching, hence higher power dissipation and energy consumption by the processor core. A profiling and estimation tool is required to calculate this operand value metric and to apply the energy models to large applications.

The novel contributions of this paper are as follows:

1. We designed a novel energy data model for a soft-processor FPGA implementation that can be generated heuristically without analysing the processor architecture and pipeline, using sets of applications described in Section 3.
2. We designed an estimation tool that analyses C code at the machine instruction level, applies the energy data model and annotation techniques to estimate the energy of each instruction executed in a given run, as described in Section 4.

2 Related Work

There are several modeling techniques to estimate energy consumption of soft processors. They can be categorized based on the model's abstraction level. The most accurate models simulate the processor at the gate level using post-place and route simulations, tracking the switching rate all internal signals. Examples of tools, based on such models, include Xilinx's XPower Analyzer XPA [1], [5] and Synopsys's Power Compiler [6]. Accurate, low level simulation models suffer from very slow simulation speeds. In fact, in order to estimate the energy consumed executing the Dhrystone benchmark [7] on a Microblaze processor core [8] using XPA took over 2 days of simulation time. This simulation was done on a quad-core i7 PC with 16 GB RAM.

Higher level modeling of processor energy and power is typically done by characterizing the workload of the processor. These can be divided in three broad groups based on the number of contributing factors in each model. We will refer to these groups as first, second and third order models. They are identified as follows:

- First order models are state-based, in which the energy consumption is derived from the state of the processor. An average dynamic power value is assigned to each power state. The dynamic energy is then estimated by multiplying the weighted average power by the total execution time. Jouletrack implements a very simple first order model [3]. System level estimation tools like Softwatt [9], Wattch [10], and other state-based estimation techniques [11] use first order processor models because of their simplicity. Energy aware scheduling techniques using first order models have also been proposed utilizing such models [12]. However, first order models do not reflect the processor energy savings from software optimizations independently of performance. For example, re-ordering assembly instructions to increase the number of repeating instructions reduces energy consumption. This effect cannot be observed by first order models.

- Second order models assign an estimated average energy value for each instruction in the processor's instruction set. The energy required to execute a program is then estimated as the sum of the energy values assigned to all the executed instructions. Many completed works use this technique such as [3], [13], [14]. More work however is required to prove the accuracy of these models to estimate large applications running on soft processors. In fact, we observed very negligible accuracy improvement over the first order models.
- Third order processor models incorporate inter-instruction energy effects. When analyzing the energy required for completing an instruction, the neighboring instructions are also accounted for. The reasoning is that neighboring instructions indicate the state of the processor before and after executing an instruction. Several processors, including an Intel 486DX2 processor [1] and a Fujitsu DSP [15], have been modeled using this approach. VLIW processors have also been modeled using third order models [16]. Third order models have been applied to estimate system-level energy consumption [17]. However, third order models do not take the effects of input data for the application into account, which can lead to significant errors.

The techniques proposed in these works were adopted to generate three models of an implementation of the Microblaze processor implemented on a Xilinx Virtex5 FPGA. The Microblaze instruction set architecture is similar to the RISC-based DLX architecture [18]. We used these models to evaluate the applicability of these techniques on a modern soft processor. The results presented in this paper demonstrate significant errors in excess of 100%. In the quest of identifying the causes of these inaccurate results, we derived results showing significant impact of operand value on energy consumption. The following section describes the proposed model, generated using a novel instruction characterization technique.

3 Operand-Value-Based Processor Energy Model

This section describes the proposed methodology for creating an Operand-Value-Based Model (OVBM) for a processor implementation consisting of:

- The base energy cost for each instruction, with zero operands,
- The maximum energy variance, due to operand values, for each instruction, and
- Linear parameters, slope and intercept (m and b) modeling the correlation between the one's density and the energy consumed in the processor data-path signals.

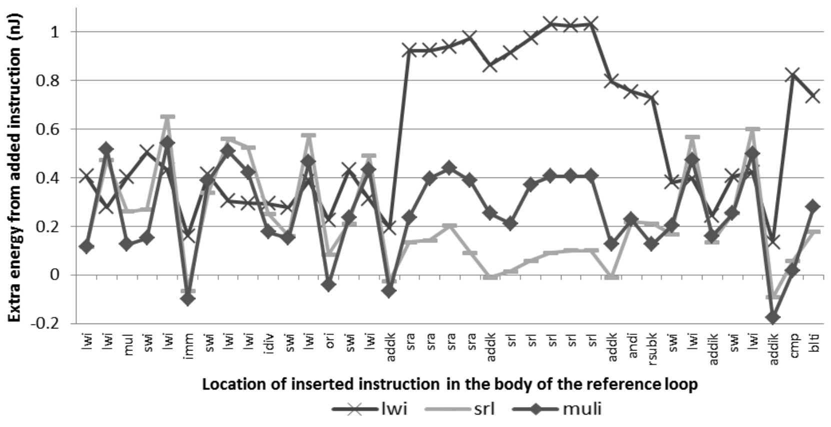

Fig. 1. Location Based Energy Profile of three instructions (Load Word, Shift Right Logic, and Multiply)

A single OVBM is sufficient to estimate the processor energy cost of any application running on a given implementation of the processor. A separate model is required for alternative design configurations and optimizations of the processor.

3.1 Base Energy Cost of Instructions

The minimum dynamic energy required by the processor to complete an instruction is defined as the base energy cost of the instruction. It is determined by the type and operation of the instruction as well as the state of the processor's internal signals prior to executing the instruction. This state dependency is referred to as the inter-instruction energy effect. Normally, we expect two consecutive instructions of the same type to consume less energy than two of different types.

Most instruction energy estimation techniques suggested in the literature rely on measuring or estimating the average power consumed by the processor while

Table 1. Base energies and maximum energy variations for Microblaze in Nanojoules (nJ)

Inst.	Base energy after instruction from class:			Max. instr. Energy Variance
	Arithmetic & Logic	*Memory*	*Shift*	
add	0.1147	0.4882	0.1608	1.0034
rsubk	0.3461	1.0352	0.7762	0.7872
mul	0.1233	0.4819	0.4019	0.9795
idiv	0.1850	0.5401	0.4419	0.7602
and	0.0892	0.5306	0.4213	0.6977
xori	0.3257	0.6345	0.5921	0.6977
cmp	0.1821	0.7108	0.5727	1.0456
nop	0.1343	0.4808	0.1959	0
lwi	0.7680	0.3536	0.9858	0.5310
swi	0.8159	0.4108	0.9761	0.2208
srl	0.1628	0.5550	0.1124	1.0782
sra	0.1571	0.5836	0.1899	1.0373

repeatedly executing an instruction or a pair of instructions in either an infinite loop or in very long programs of the repeating instructions [1], [3], [13, 14, 15, 16]. This, however, accounts for a very special and limited execution case of the instruction, not representative of its expected executions in real applications. The inaccuracy of such models is clearly demonstrated in the Section 5.

In order to accurately characterize the base energy required by an instruction while also accounting for inter-instruction effect, we designed a reference application that executes a single basic block of approximately 100 diverse instructions operating on changing values in an infinite loop. This is done to represent a general and diverse environment in which we examine the energy characteristics of the instruction. A set of benchmarking applications is created for each instruction. In each application, an instruction is inserted between a different pair of instructions in the reference application. To ensure a minimum energy consumption, we set the operands values of this instruction to zero. Hence, we create as many applications per instruction as there are instructions in the reference application. The low level model of the processor is then used to evaluate the increase of the energy consumed as a result of the inserted instruction. Presented graphically, we plot the energy of the instruction when it was inserted after the instruction of reference application as shown on the horizontal axis as in Figure 1. We refer to each plot as the *Location-Based Energy Profiles* (LBEP) of the instruction.

We derived LBEPs for all Microblaze instructions, three of which are shown in Figure 1. We observed a strong similarity in the patterns of instructions that utilize the same processor data-path units. The energy consumed by each instruction is minimum when it is inserted after an instruction of the same type, and varies when inserted after other types of instructions. This implies that the Microblaze instruction set can be grouped based on the data-path units they utilize. These groups are: memory access, shift operation, and arithmetic and logic operations. Figure 1 illustrates the LBEP of three instructions: memory load (lwi), logical right shift (srl) and integer multiplication (muli). The X-axis represents the sequence of instructions of the reference application. The Y-axis is the increase in energy consumption due to the insertion of the instruction before the reference application instruction on the X-axis. The LBEPs for the arithmetic and logic instructions are similar to that exhibited by (muli). Similarly, the LBEPs of the shift and memory instructions are similar to those for (srl) and (lwi).

To derive the base energy cost of instructions from its LBEP, we consider only the sample points in which the instruction is inserted between instructions of the same type. Thus, we obtain three base energy costs for each instruction, one for each case where it executes following instructions of one of three groups identified. For instance, from the energy profile of the load word instruction (lwi) given in Figure 1, we first consider the energy values corresponding to the lwi instruction inserted between pairs of logic or arithmetic instructions. The average of these energy estimates is recorded as the base energy cost of the load instruction following a logic or arithmetic operation. Similarly, estimates for the base energy cost of the lwi instruction following memory and shift instructions, are also evaluated. The second group of columns in Table 1 presents the three base energy costs of Microblaze instructions when they follow an instruction from one of the three groups. This constitutes the first parameter of the OVBM.

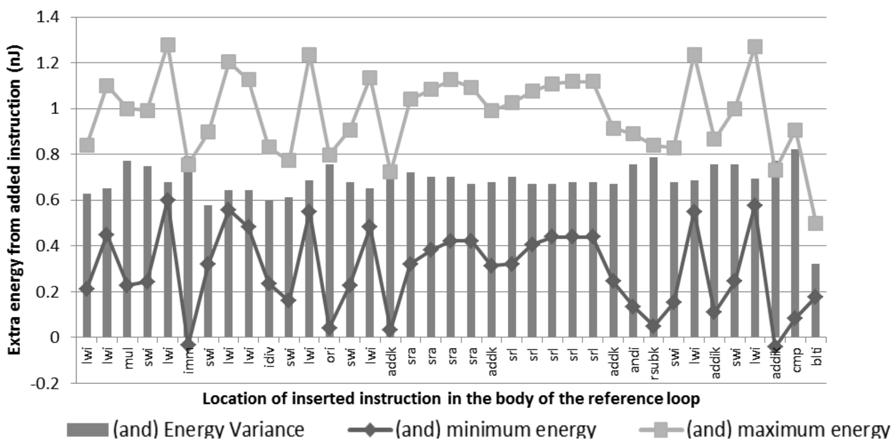

Fig. 2. Maximum energy variance, maximum, and minimum LBEP of an *and* instruction

3.2 Maximum Instruction Energy Variance

The dynamic power consumed by the soft processor cores implemented on an FPGA is dependent on the operand values of the instruction as discussed in Section 1. To incorporate the influence of operand values, we introduce a new parameter to the OVBM, called maximum instruction energy variance. It is defined as the maximum difference between the energy cost of an instruction with large operands and the instruction's base energy cost. To observe this variance, a copy of the energy benchmarks used to generate the LBEPs, described in the previous subsection, is created. The operand values of the inserted instructions are then changed to the maximum positive values of 0x7fffffff instead of zero, generating a maximum energy profile for the instruction. The two plots in Figure 2 illustrate the location-based maximum and base energy profiles of the (and) instruction. The difference between the profiles is presented as a bar graph in Figure 2.

As seen in Figure 2, the difference between the maximum energy and the base energy does not vary significantly with the location of the inserted instruction. The small variance is due to the different average one's densities of the instructions of the reference loop. The differences are averaged to obtain a single value of maximum energy variance for each instruction. The total energy consumed by an instruction i is modeled using equation (1).

$$E_i = E_{base}(i,j) + k(operand\ value\ property) \cdot EV(i) \tag{1}$$

$E_{base}(i,j)$ is the base energy of instruction i following instruction j. $EV(i)$ is the maximum energy variance of instruction i, and k is a factor that determines the what fraction of the maxim energy variance of instruction i given the operand values.

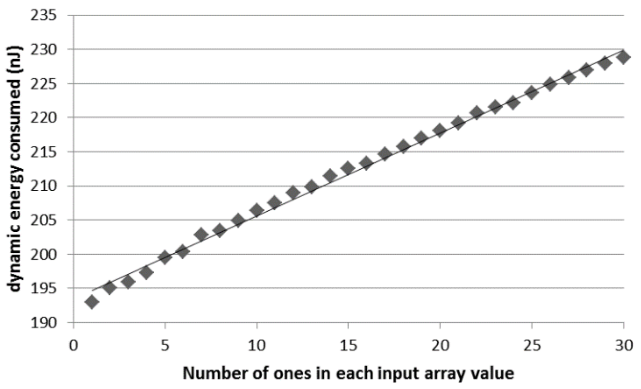

Fig. 3. Relation between the energy consumed running an application and the one's count of its input values

3.3 Energy Impact of Operand Values

As described in Section 1, the density of ones in the operand value impacts the energy consumed by soft-processors. To examine this correlation, we generated a different set of benchmarks where a fixed set of varying instructions operate on values with increasing densities. An application containing a source array of 10 elements and a loop with instructions is to load a value from the source array and perform several operations using it is first implemented. A total of 30 applications were then derived from it, in each the array values were initialized to different positive integers containing the same number of ones. The energy required to execute these application is estimated using reference estimation tools, plotted in Figure 3.

Figure 3 demonstrates a linear correlation of the energy consumption and operand densities. The lowest value, at 194.7 nJ, corresponds to the least dense operands. This value is in fact only 0.4% less than the sum of the base energies of the instructions in the benchmark (which evaluates to 195.4 nJ using values from Table 1). The additional energy consumed, beyond the base energy cost of 195 nJ, is the accumulation of energy consumed in the data-path signal result of the of instructions in the benchmark (the $k \cdot EV(i)$ term in equation (1)).The k factor in equation (1) is expressed as a linear function of the one's density as given in equation (2). The one's density of the operands of instruction i, is denoted by $OD(i)$ with slope m and y-intercept b.

$$k = m \cdot \left(OD(i)\right) + b \tag{2}$$

By substituting k using equation (2) into equation (1) we can express the estimated energy of a basic block of N instruction using equation (3) as the sum of the estimated energy consumed for each instruction making the basic block.

$$E_{est} = \sum_{i=1}^{N} E(i) = \sum_{i=1}^{N} E_{base}(i, i-1) + \sum_{i=1}^{N}(m \cdot OD(i) + b) \cdot \Delta E(i) \tag{3}$$

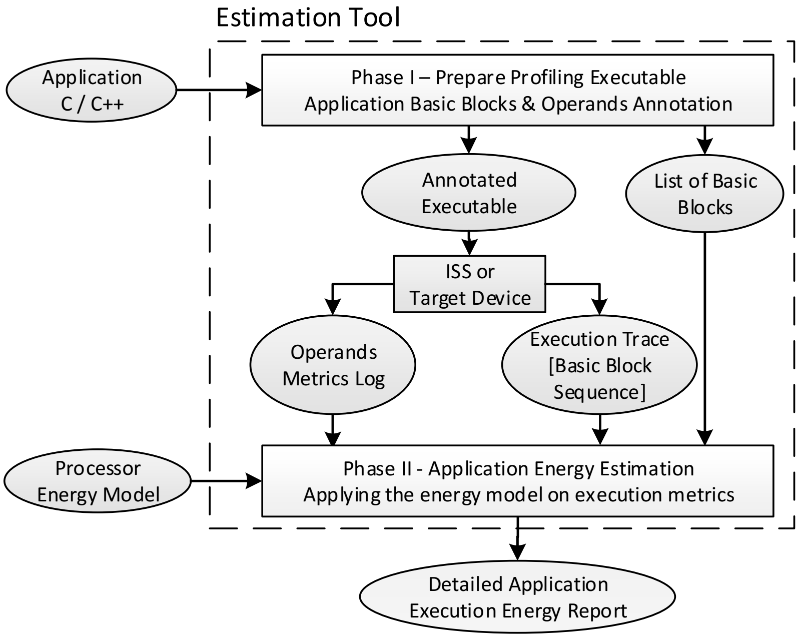
Estimation Tool
Application C / C++
Phase I – Prepare Profiling Executable Application Basic Blocks & Operands Annotation
Annotated Executable
List of Basic Blocks
ISS or Target Device
Operands Metrics Log
Execution Trace [Basic Block Sequence]
Processor Energy Model
Phase II - Application Energy Estimation Applying the energy model on execution metrics
Detailed Application Execution Energy Report

Fig. 4. Proposed automated application analysis, annotation, and energy estimation tool

In order to derive the values for *m* and *b* for a given processor, we equated the linear approximation equation in (3) to the reference energy estimation for each of the 30 applications, generating a system of 30 linear equations. Substituting in the known values from the OVBM in Table 1, we are able to calculate the sum of base energies and energy variance for the fixed instructions in all 30 applications. The average one's densities of each instruction are found using the profiling stage of the estimation tool described Section 4 instead of the one's density of the input array. This results with an over-determined system of 30 equations and two unknowns, m and b. An approximate solution at m = 0.016 and b = -0.061 for the Microblaze processor is determined once and added to the model parameters.

4 Energy Estimation and Annotation

We have developed a tool to automatically apply the proposed model to an embedded application. The tool works in two phases as shown in Figure 4.

In the first phase, the tool annotates the source code with instructions aimed to identify the instructions that will execute, and run-time operand value metrics (one's density). It also identifies the basic-blocks of the application. After executing the annotated application, the logged profiling data is sent to the host PC for processing in the second phase. These annotations:

1. Record the execution sequence of the basic blocks,
2. Record the opcode of each instruction executed, as well as the values of their destination registers,
3. Calculate the average of the one's densities of all values used by each non-repeating shift instruction, and
4. The tool also generates a list of the basic blocks and the instructions in each basic block. Using an object-dump utility [17] the exact machine instructions of each basic block are identified

The second phase of the estimation tool uses the processor energy model, parameters obtained from the execution of the annotated application, and list of basic blocks, to estimate the energy consumed by each executed instruction using equation (3) described in Section 3. The estimated energy consumed by each basic block is then be evaluated as the sum of the estimated energy of all its instructions. The total energy consumption is then found using the estimated energy of the basic blocks and the execution trace.

Table 2. Baseline energy estimates using XPA [5]

Application	Time (μs)	Power (mW)	Energy (mJ)
Dhrystone	39.35	33.35	1.31
Quicksort	164.20	33.78	5.55
ReadBMPBlock	251.61	39.96	10.05
DCT	166.68	30.84	5.14
Quantize	58.20	25.52	1.49
Zigzag	25.33	30.98	0.78
Huffman Encode	471.95	40.70	19.21
JPEG	973.77	37.66	36.67

5 Experimental Results

To evaluate the proposed estimation method, we developed an OVBM of the Microblaze soft processor implementation on a Virtex5 FPGA as described in Section 3. The processor was implemented without cache, connected via a Local Memory Bus (LMB) to 64 kB block RAM, which stores the program and data of the application. The system clock is operating as a frequency of 125 MHz. To compare OVBM to the state of the art, we also developed three different energy models using the techniques surveyed in Section 2. We implemented the estimation tool described in Section 4 to automatically annotate and analyze applications targeted for a Microblaze processor. The automatic annotation tool was expanded to accept all generated energy models.

5.1 Estimation Accuracy

We examine the accuracy of the models using a set of 8 application benchmarks listed in Table 2 re used. The execution time and average dynamic power consumed by the processor for each benchmark were obtained using Xilinx XPA [5] and used to calculate the energy consumed by each benchmark. These estimations are highly accurate and can be used as baseline to compare the accuracy of all models. The benchmarks are Dhrystone, an implementation of the Quicksort algorithm, and five functions of a JPEG [19] encoder: Read BMP Block, *Discrete Cosine Transfer* (DCT), Quantize, Zigzag, and Huffman encode.

Table 3. Estimated energy in Millijoule (mJ) and estimation error, relative to references in Table 2, comparing the accuracy of proposed model to 1st, 2nd, and 3rd order energy models generated using previous work methods.

Application	1st Order		2nd Order				3rd Order				OVBM	
	E	Err	E	Err	E*	Err	E	Err	E*	Err	E	Err
Dhrystone	1.31	0.0%	3.6	171%	1.31	0.0%	3.3	155%	1.31	0.0%	1.30	-0.7%
Quicksort	5.48	-1.3%	16	185%	5.07	-8.7%	13	128%	4.95	-10.7%	5.37	-3.2%
ReadBMPBloc	8.39	-16%	25	145%	7.90	**-21%**	22	116%	8.50	**-15%**	8.82	**-12.3%**
DCT	5.56	8.2%	18	253%	5.82	13.2%	18	253%	7.12	**38.5%**	4.96	-3.5%
Quantize	1.94	**30%**	6.4	329%	2.04	**38%**	4.0	169%	1.57	5.4%	1.47	-0.9%
Zigzag	0.84	7.7%	2.3	195%	0.74	-5.3%	2.3	194%	0.90	15.3%	0.78	-0.6%
Huffman Enc.	15.74	**-18%**	51	164%	16.26	-15%	48	148%	18.68	-2.7%	17.64	-8.2%
JPEG	32.48	-11.4%	102	179%	32.76	-11%	94	156%	36.77	0.3%	33.67	-8.2%
Average error		13.4%		203%		16%		165%		12.6%		4.7%
Std. Deviation of error		9.5%		60%		11%		43%		12.8%		4.3%

Table 3 presents the dynamic energy estimations obtained using the four models for the examined benchmarks. The first order model uses the average dynamic power consumed by Microblaze executing the Dhrystone benchmark as the average power parameter. The estimated energy consumed by the remaining benchmarks are the products of the execution times by the average power. Naturally, the error in estimating the Dhrystone benchmark using this method is zero. The error in estimating the other benchmarks ranges between -18% and 30.7%, when compared to the reference values in Table 2. It is important to note that the accuracy of the first order model depends on the choice of the average power paramter used. In this instance, using the average power of Dhrystone resulted in reasonable estimates for benchmarks like quicksort and ReadBMPBlock, however, as it's tested with more applications, the unpredictablitiy of this method becomes apparent.

The second and third order models, initially, produced large errors. The energy for an instruction in the second order model is obtained using low level simulations of the given instruction in a loop as done in [3], [13, 14, 15, 16]. Similarly, the energy for a pair of instructions in the third order model is obtained using low level simulation of the instruction pair in a loop. Clearly, this technique does not produce accurate estimates of the energy cost of an instruction. As suggested in [3], the models are calibrated using the Dhrystone estimation error. However, despite calibration, the second and third order models generated worst case errors of up to 37.6%, and 38.5% respectively. The average errors were also high at 16% and 12.6%. Furthermore, the high deviation of estimate errors further demonstrates that these models cannot be used with confidence.

The energy estimates generated using our approach are given in the final column, titled OVBM. It outperforms other models, both in terms of average accuracy and estimation confidence. The worst case error obtained is -12.3%, with an average error of only 4.7%. Therefore, dynamic energy estimates obtained using the proposed approach can be used with confidence for early design space exploration as well as system and software reconfiguration and optimization efforts.

Table 4. Execution time of PVBM-based estimation tool compared to Post-place-and-route simulation (XPA)

Application	**OVBM Tool (Seconds)**			**XPA (Minutes)**
	Phase 1 + 2 (Host)	**Execution** (Target)	**Total**	
Dhrystone	0.03	7.49	7.53	72
Quicksort	0.01	23.08	23.09	147
ReadBMPBlock	0.21	5.88	6.08	202
DCT	0.03	10.85	10.88	148
Quantize	0.01	8.40	8.41	85
Zigzag	0.01	4.41	4.42	65
Huffman Encode	0.07	65.04	65.11	340
JPEG	0.28	104.24	104.52	638

5.2 Estimation Speed

In addition to having a higher average accuracy and confidence over other instruction-level models, the proposed estimation technique generates energy estimates within seconds. Table 4 compares the time required by our tool to XPA [5] to derive the presented estimations. The simulation host was utilized a Nehalem based Intel i7 quad-core processor and 16 GB of DDR3 RAM. The time presented includes the time required to execute both estimation phases presented in Section 4 and the execution time of the annotated application. In our experiments, we used a Microblaze implementation on a Xilinx Virtex5 FPGA development board to run the annotated executable. The total time required to complete the two phases running on the host for the JPEG benchmark was under one second. The time needed to run the annotated executable and transfer the logs to the host was under two minutes. As such, the total estimation time this benchmark was under two minutes. In contrast, it took over 10 hours to obtain the baseline estimate using XPA. As such, our model demonstrates 3 orders of magnitude speedup over post-place-and-route models.

6 Conclusion

We presented a novel dynamic energy modeling technique for soft processors in FPGA based on the operand values of instructions. We showed that energy estimates obtained from our model are significantly more accurate than the state of the art energy models. The energy model can be used for early software optimization, system architecture reconfiguration and customization as well as design space exploration. In the future, we expect to validate our model with more applications and other embedded processors.

References

1. Bazzaz, M., Salehi, M., Ejlali, A.: An accurate instruction-level energy estimation model and tool for embedded systems. IEEE Trans. On Instrumentation and Measurement **62**(7) (2013)
2. Xilinx Inc, Power Methodology Guide UG786 (2011). http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug786_PowerMethodology.pdf
3. Sinha, A., Chandrakasan, A.P.: JouleTrack-a web based tool for software energy profiling. In: Proceedings, Design Automation Conference pp. 220–225 (2001)
4. Krishnaswamy, A., Gupta, R.: Dynamic coalescing for 16-bit instructions. ACM TECS **2005**, 3–37 (2005)
5. Xilinx Inc, XPower Analyzer (2011). http://www.xilinx.com/products/design_tools/logic_design/verification/xpower.htm
6. Synopsys Inc. Power Compiler (2012). http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/PowerCompiler.aspx
7. Weicker, R.P.: Dhrystone: a synthetic systems programming benchmark. Comm. of the ACM **27**, 1013–1030 (1984)
8. Xilinx Inc.: LogiCORE IP, MicroBlaze micro controller system. http://www.xilinx.com/tools/microblaze.htm
9. Gurumurthi, S., Sivasubramaniam, A., Irwin, M.J., Vijaykrishnan, N., Kandemir, M.: Using complete machine simulation for software power estimation: The SoftWatt approach. HPCA, pp. 141–150 (2002)
10. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: A framework for architectural-level power analysis and optimizations. ISCA, pp. 83–94 (2000)
11. Ahmadinia, A., Ahmad, B., Arslan, T.: A state based framework for efficient system-level power estimation of costum reconfigurable cores. In: Proceedings, SOC Conf., pp. 1–4 (2008)
12. Chen, J., Thiele, L.: Task partitioning and platform synthesis for energy efficiency. In: Proceedings. RTCSA, pp. 393–402 (2009)
13. Brandolese, C., et al.: Software energy estimation based on statistical characterization of intermediate compilation code. Symposium. Low Power Electronics and Design (ISLPED), pp. 333–338 (2011)
14. Tiwari, V., Tien-Chien Lee, M.: Power analysis of a 32-bit embedded microcontroller. ASPDAC, pp. 141–148 (1995)
15. Tien-Chien Lee, M., Tiwari, V., Malik, S., Fujita, M.: Power analysis and minimization techniques for embedded DSP software. IEEE VLSI, (5), pp. 123–135 (1997)
16. Roy, S., Bhatia, R., Mathur, A.: An accurate energy estimation framework for VLIW processor cores. In: Proceedings. ICCD, pp. 464–469 (2006)
17. Kim, J., Kang, K., Shim, H., et al.: Fast estimation of software energy consumption using IPI(inter-prefetch interval) energy model. In: VLSI-SoC, pp. 224–229 (2007)
18. Sailer, P.M., Philip, M., Kaeli, R.: The DLX Instruction Set Architecture Handbook (1st ed.). Morgan Kaufmann Publishers Inc., CA, USA (1996)
19. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. On Consumer Electronics **38**, xviii–xxxiv (1992)