EARLY PERFORMANCE ESTIMATION OF IMAGE COMPRESSION METHODS ON SOFT PROCESSORS

Adam Powell, Christos-S. Bouganis, Peter Y.K. Cheung

Department of Electronics and Electrical Engineering
Imperial College London
{adam.powell08,christos-savvas.bouganis,p.cheung}@imperial.ac.uk

ABSTRACT

This paper presents a power and execution time estimation framework for an FPGA-based soft processor when considering the implementation of image compression techniques. Using the proposed framework, a quick power consumption and execution time estimate can be obtained early in the design phase allowing system designers to estimate these performance metrics without the need of implementing the algorithm or generating all possible soft processor architectures. This estimate is performed using both high-level algorithm parameters and soft processor architecture parameters. For system designers this can result in fast design space exploration. The model can predict the execution time of an algorithm with an average of 139% less relative error than predictions using only architecture parameters with the same framework.

1. INTRODUCTION

Image compression methods on embedded systems are becoming more important due to the availability of cheap high-resolution embedded cameras and the growing need to store an even larger set of images in the device. Soft processors allow for quick design and use of microprocessors in FPGA applications such as image compression. However, it is difficult for the embedded system designer to know how a particular image compression method is going to perform early in the design process because specific parameters of the architecture may not yet be known. Ideally, the designer would know how a compression algorithm would perform in terms of power consumption and execution time without the need of implementing this algorithm or knowing specific architecture parameters.

Exploration of the soft processor architecture space is a lengthy process; based on the number of architecture combinations, the generation of all possible architectures can take up to 25 days in some cases. By having a profile of power consumption and execution time as a function of high-level parameters that capture the main computational and memory access characteristics of the algorithm and the architectural parameters of the hardware, early optimization and design decisions can be performed. This allows for a much earlier estimate than previous work at the expense of a constrained application space and accuracy of the predictions.

A popular method for soft processor power modeling is done on the instruction-level and is covered in these works [1–3]. The energy to execute a given instruction is profiled and stored in a look-up table. Therefore the energy for a program execution is the sum of required energy for the executed instructions. Component-level modeling is covered in these works [4, 5]. These methods profile the average consumption of various system components such as caches and arithmetic units. The power consumption of these individual components depends on the usage frequency of these components. Hybrid methods which combine the two paradigms have also been examined [6, 7]. Both component and instruction level modeling require the implementation of the algorithm and detailed knowledge of the underlying system architecture.

To our knowledge, this is the first attempt at predicting power consumption and execution time using both algorithmic and system-level parameters on a soft processor at this level of abstraction.

The compression methods and selection of their parameters are discussed in Section 2, followed by the actual construction of the model in Section 3. Finally, the evaluation process for the model is discussed in Section 4.

2. ALGORITHM PARAMETER SELECTION

The algorithm parameters used for modeling must be high-level and need to be easily extractable by an engineer familiar with the main features of the algorithm without having to provide an actual implementation of it.

2.1. Domain-specific Knowledge

The envisaged high-level models are challenging to construct due to the high level of abstraction that is targeted which removes many descriptive aspects of the system in

Table 1. Algorithms and parameters used in modeling

Algorithm	Mem./Ari. Ratio	Block size $log_2 n$	Precision	Mult.?	Div.?	Block dependency
JPEG Slow Integer	3	3	Fixed	Yes	Yes	No
JPEG Fast Integer	4	3	Fixed	Yes	No	No
JPEG Floating point	3	3	Floating	Yes	Yes	No
JPEG 2000 Reversible	3	6	Fixed	No	Yes	No
JPEG 2000 Irreversible	3	6	Fixed	Yes	Yes	No
JPEG XR	5	2	Fixed	No	No	Yes
Fractal Compression	1	2	Floating	No	Yes	Yes
Vector Quantization	1	1	Fixed	Yes	No	Yes

question. By using domain-specific knowledge, key characteristics of the algorithm can be used that add low-level descriptiveness to the high-level parameters.

The proposed framework takes advantage of the regularity of image compression methods; nearly all image compression methods perform a relatively small set of functions repeatedly on the entire image. This set of functions usually works on the data in *blocks* of the original image. This characteristic allows the algorithm parameters to focus on a small set of instructions and to disregard low-level implementation details such as control statements.

2.2. Parameter Selection

The following section describes the selection of descriptive parameters for the image compression methods used in the construction of this model. The algorithms and their parameters are shown in Table 1.

2.2.1. Proportion of memory to arithmetic operations

Each compression algorithm has different characteristics in terms of both the number of arithmetic operations and the types of these operations. Consideration must be given to the proportion of arithmetic operations to memory operations. This gives an indication on how reliant the algorithm will be on caching or low-latency arithmetic units.

Because this is a high-level parameter, it is a qualitative evaluation of the number of arithmetic operations per memory access. The values shown in Table 1 were calculated using profiling tools. The number of memory accesses is much higher than arithmetic operations so this parameter is a measure of the relative degree of this comparison versus other algorithms. Using this model in practice, this value would be estimated by the engineer.

2.2.2. Arithmetic hardware

Another important aspect of computations is the presence of arbitrary (that is, non-2^n) multiplication or division. In the soft processor, the hardware multipliers available can be either the dedicated multiplier units or multipliers implemented in the FPGA logic (in addition to software emulation). Arbitrary division can be implemented in either hardware or software. Floating-point hardware support is not included in this study.

2.2.3. Block characteristics

Lastly, the block size will give an indication of the reliance of the algorithm on both instruction and data caches. Additionally, data dependency between blocks should introduce a greater reliance on larger data cache sizes in order to reduce average access time for memory.

3. MODEL CONSTRUCTION

To model the interactions between the algorithm and the soft processor, as well as their effects on power consumption and execution time, an appropriate modeling technique must be chosen.

It is expected the final model will be non-linear due to the non-linear characteristics of cache performance and multiplier types, for example.

In this work, the modeling technique chosen is based on a type of decision tree called a regression tree [8]. A regression tree is used to predict a continuously valued response variable using an input vector of continuous or discrete predictor variables. More about decision and regression trees can be seen in [8].

For this model, a regression forest (that is, multiple regression trees) was used. A separate regression forest is created for each performance metric (discussed more later). One advantage is the reduction in error that comes from using more than one tree. Breiman [9] has shown that the error decreases asymptotically as the number of trees increases.

The data used in the learning and testing of a regression forest was a set of vectors where each vector was the particular algorithm parameters followed by the parameters of the architecture on which the algorithm was executed. The response variable vector differed based on the particular forest

Table 2. NIOS 2 parameter summary

Category	Parameter	Choices
Data cache	Cache size	7
	Line size	3
	Burst transfers	2
Instruction cache	Cache size	7
	Burst transfers	2
Multipliers	Multiplier type	2
Division	HW or SW division	2
Clock frequency	50 MHz	
Total combinations		2,352

being created (e.g. power or execution time). Every algorithm was executed and measured on every combination of architecture parameters.

4. MODEL EVALUATION

To evaluate this model, actual power measurements were taken for the execution of the algorithms on the soft processor. In this work, the Altera NIOS 2 soft processor was chosen as the platform, although other soft processors could easily be used instead. All possible adjustable NIOS 2 architecture parameters are shown in Table 2.

Each algorithm performed compression on a grayscale image of size 128 x 128 pixels which was taken arbitrarily from the FERET face dataset.

To measure power, the Cyclone III starter kit offers two sense resistors that allow the measurement of the current drawn by the FPGA core and the memory plus I/O devices. Execution time was measured using the NIOS 2-compatible version of gprof, the GNU profiling tool. This tool provides a function-by-function breakdown of execution time.

A regression forest is only able to predict one response variable so a forest was created for each metric under investigation. These metrics are FPGA core power, memory and I/O power, and execution time. In addition to these, there is combined power, which is the sum of the FPGA core power and the memory devices power measurements. The second is energy-per-execution which is the product of the combined power and execution time measurement. This is essentially a measure of the efficiency of an algorithm.

4.1. Results

As this model is at a much higher level than other power estimation techniques, it is difficult to produce a meaningful comparison to previous work. The ability to estimate performance at a very early stage in design when neither the algorithm nor hardware is implemented is expected to come at the cost of higher error in the estimation. Consequentially, the gain provided by this higher level model compared to previous work is difficult to quantify. Therefore, the comparison will be done to models created with the same construction technique described previously but using only architecture parameters. This would be a typical way of intuitively predicting performance of an unknown algorithm (e.g. assuming that using a larger cache will guarantee a faster execution time). This comparison will demonstrate the gain of using algorithm parameters for generating an accurate high-level estimation of power consumption and execution time.

As mentioned above, five models were generated for the five metrics in question. These five models will be considered a *model set*. For brevity, the model set generated using only the architecture parameters will be called the *architecture set* and the model set generated using both architecture and algorithm parameters will be called the *full set*.

To test the models, the “Leave One Out” method was used with all algorithms in turn. This means the model was constructed using all but one of the algorithms and then tested against the excluded algorithm. This shows the ability of the model to “predict” the excluded algorithm. In practice, a system designer would use this same model to predict a particular algorithm’s performance based on their particular set of parameters.

The error metric used to calculate the accuracy is the Relative Error (RE) of the predicted value versus the measured value.

The distribution of the RE in each model is shown in the histogram in Figure 1 along with the mean RE in Table 3.

4.1.1. Comparison

Table 3 shows the comparison of the mean relative error (MRE) for the architecture and full sets. This is the mean error of all predictions for each of the five models.

It can be seen that power estimation using either architecture or the full set of parameters provide similar accuracy. This is may be partly due to the fact that dynamic power consumption is the smaller part of FPGA power consumption compared to static power [10]. Also, it shows that the prediction of the combined power is more accurate than that of either the core or memory power alone. This is likely due to the architecture parameters affecting both core and memory devices. For example, a larger cache will increase power consumption in the core but reduce power consumption in the memory because it is used less. The results also demonstrate that estimation of execution time and energy-per-execution is improved by the use of algorithm parameters.

4.1.2. Prediction error distribution

Focusing on the full set error only (Figure 1), the distribution of power estimate error is localized around the mean for

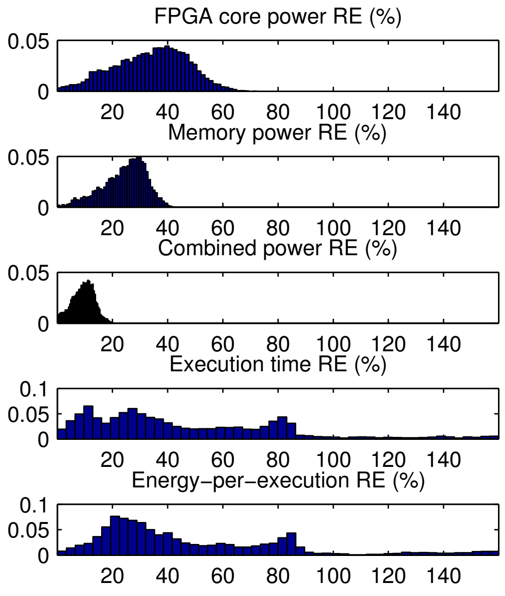

Fig. 1. Error histogram for the full set

all power models. The distribution of power estimate errors appear Gaussian in nature and therefore are well-behaved. Because the combined power is the most accurate, the parameters are able to predict total system interaction better than core or memory power separately.

However, the distribution of error for the execution time (and consequentially energy-per-execution) model is concentrated in the lower errors but the mean is much higher due to a number of problematic combinations of architectures and algorithms. Analysis of the error shows that the highest error in predicting power (core, IO, or combined) happens when the data cache size is at its smallest (512 bytes) but is relatively less sensitive to other parameters. This error increase is likely due to the line replacement behavior; lines will be replaced constantly with such a small size which causes erratic behavior that is difficult for the high-level parameters used here to capture.

The same statement holds true in predicting execution time and energy-per-execution with the addition of the error being dependent on the size of the instruction cache as well.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented predictive models for image compression techniques running on soft processors. It has the ability to predict power and execution time from high-level algorithm and architecture parameters. The modeling was done using regression forests that are able to generate a descriptive model of data exhibiting non-linear characteristics.

Table 3. Model set accuracy versus all algorithms

Model	**Mean Relative Error**	
	Architecture	Full
Core Power	33.4%	31.9%
Memory Power	24.2%	33.7%
Combined Power	9.4%	9.4%
Execution Time	545%	169%
Energy-per-execution	495%	170%

While these models exhibit some high levels of error, the results are promising and this error should decrease with the addition of more algorithms to adequately explore this parameter space.

These predictions happen very early in the design phase long before an algorithm implementation is possible. This also removes the need to spend 25 days generating all architecture combinations to see how a *specific* algorithm performs on a soft processor. Ultimately, this allows for fast design space exploration in the very early stages of design for current and future image compression algorithms.

References

[1] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *VLSI Systems, IEEE Transactions on*, 1994.

[2] J. Ou and V. Prasanna, "Rapid energy estimation of computations on fpga based soft processors," in *SOC Conference, 2004. Proceedings*.

[3] S. Sultan and S. Masud, "Rapid software power estimation of embedded pipelined processor through instruction level power model," in *Performance Evaluation of Computer Telecommunication Systems. Symposium on*, 2009.

[4] X. Wang, S. Ziavras, and J. Hu, "System-level energy modeling for heterogeneous reconfigurable chip multiprocessors," in *Computer Design. International Conference on*, 2006.

[5] J. Laurent, *et al.*, "Functional level power analysis: An efficient approach for modeling the power consumption of complex processors," in *DATE, 2004. Proceedings*.

[6] Y. Fei, *et al.*, "A hybrid energy-estimation technique for extensible processors," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2004.

[7] P. Zipf, *et al.*, "A power estimation model for an fpga-based softcore processor," in *FPL, 2007. Proceedings*, 2007.

[8] L. Breiman, *Classification and Regression Trees*, 1984.

[9] L. Breiman, "Random forests," *Machine Learning*, 2001.

[10] T. Tuan and B. Lai, "Leakage power analysis of a 90nm fpga," in *CICC, 2003. Proceedings*.