MAVO: An Automated Framework for ESL Design Monitor, Analyze, Visualize and Optimize

Yasaman Samei and Rainer Dömer

Technical Report CECS-14-12
November 20, 2014

Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
(949) 824-8919

ysameisy,doemer@uci.edu
http://www.cecs.uci.edu

MAVO: An Automated Framework for ESL Design Monitor, Analyze, Visualize and Optimize

Yasaman Samei and Rainer Dömer

Technical Report CECS-14-12
November 20, 2014

Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
(949) 824-8919

ysameisy,doemer@uci.edu
http://www.cecs.uci.edu

Abstract

Over the last decade, research in Electronic System Level (ESL) design has resulted in significant advances in addressing the rising design complexity and meeting the required performance constraints. Now a major concern of system-level design is the power reduction and energy dissipation of the system-on-chip which not only affects battery lifetime but also thermal aspects and reliability of the end product. Towards power-aware ESL design, we present MAVO, an automated framework to Monitor, Analyze, Visualize and Optimize both power and performance at the early stages of the design process. Using four techniques for pipeline balancing, Dynamic Voltage and Frequency Scaling (DVFS), power scheduling, and peak-power reduction, we minimize and balance the energy dissipation at the same time as addressing performance constraints. Using an image processing application, we demonstrate the benefits of automated power profiling and visualization in the MAVO framework resulting in a smoothed energy dissipation profile by 29% and with 16% power reduction without performance penalty.

Contents

List of Figures

List of Tables

MAVO: An Automated Framework for ESL Design Monitor, Analyze, Visualize and Optimize

Yasaman Samei and Rainer Dömer
Center for Embedded and Cyber-Physical Systems
University of California, Irvine
Irvine, CA 92697-2620, USA
ysameisy,doemer@cecs.uci.edu
http://www.cecs.uci.edu

Abstract

Over the last decade, research in Electronic System Level (ESL) design has resulted in significant advances in addressing the rising design complexity and meeting the required performance constraints. Now a major concern of system-level design is the power reduction and energy dissipation of the system-on-chip which not only affects battery lifetime but also thermal aspects and reliability of the end product. Towards power-aware ESL design, we present MAVO, an automated framework to Monitor, Analyze, Visualize and Optimize both power and performance at the early stages of the design process. Using four techniques for pipeline balancing, Dynamic Voltage and Frequency Scaling (DVFS), power scheduling, and peak-power reduction, we minimize and balance the energy dissipation at the same time as addressing performance constraints. Using an image processing application, we demonstrate the benefits of automated power profiling and visualization in the MAVO framework resulting in a smoothed energy dissipation profile by 29% and with 16% power reduction without performance penalty.

1 Introduction

ESL design has emerged around a decade ago and has been equipped with different System Level Description Languages (SLDL) and Electronic Design Automation (EDA) tools. However, the increasing complexity of SoCs extended the design space and consequently intensified the design productivity gap. Beside the difficulties in managing the size of design space, there are power, temperature and reliability concerns of the design as well. All these factors, make system design a challenging task. The fact that power optimization at system level can result in significant reduction up to an order of magnitude compared to power saving at lower levels(e.g. less than 10% at gate level), reveals the necessity for a effective and efficient system level power analysis [9]. System level design decisions can significantly increase the reliability and lifetime of a system and enhance any other power optimization applied at lower levels later. The main goal of evaluation at system level is to determine whether or not the proposed design meets the power, timing, temperature and reliability constraints and to identify the power saving opportunities. In order to explore the design space for better options, identify power saving solutions, detect peak power, and improve the reliability, a profound understanding of the power dissipation behavior of the design is required within reasonable time. The major design decisions such as component selection, scheduling, pipelining, and design configurations are essential to be made as soon as possible in the design process. Since changing them at lower levels is expensive in terms of both time and effort. Our proposed MAVO framework is designed to answer the need for system-level power and performance evaluation with minimal effort. The main contributions of this paper are:

- Provide a framework to monitor the system behavior in terms of power and performance, through both power and performance logs, and graphical visualization over time, in different part of HW and SW, as well as different design elements.

- Present a comprehensive analysis of system behavior, based on power and performance reports
- Apply power management mechanisms and assist the designer to investigate further power saving solutions.

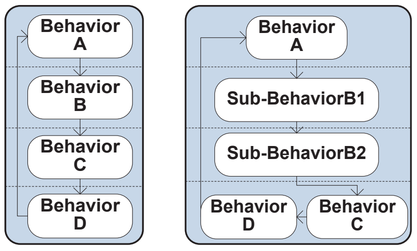
MAVO presents power and performance optimization techniques for design modification, voltage and frequency scaling, power aware scheduling, and dynamic power management with shut-off.

2 Motivation

Power and performance are major design concerns, and they directly effect on all other aspect of the design, such as area, temperature, reliability and lifetime of the device. However, evaluating and monitoring power and performance is a prime design challenge, particularly in multiprocessor SoCs. Therefore, a comprehensive analysis of energy dissipation within the system among HWs, communication elements, memories and SW processors is essential and can be achieved by profiling the simulation and applying power models. The features and functionality of MAVO are designed to fill this gap, at system level. The power optimization techniques have simple idea behind them, like voltage and frequency scaling or dynamic slack reclamation, however, in order to apply the techniques either statically in design phase, or dynamically during running time, a powerful platform is required to investigate the design rapidly and with adequate details.

2.1 Design Modification

Multiple techniques have been proposed for optimizing power dissipation. However, a low power design is mainly efficient due to its architecture and design model itself rather than the applied optimization techniques. For instance, the effect of having a system working in form of a pipeline configuration and balancing the pipeline stages, cannot made by applying a power optimization technique, such as DVFS.



(a) Original design (b) Modified design

Figure 1: Evaluation of different architectures for power and performance

Fig.1 shows two different design options of a design. The design in Fig.1(a) has 4 pipeline stages, *A*,*B*,*C* and *D*. After monitoring the power dissipation within different behaviors and availability of processing elements, the pipeline stages, and weight of their assigned behaviors, design has been modified as it is shown in Fig.1(b). Fig.1(b) shows an alternative with split stage *B* (*B1* and *B2*) and merged stage *C* and *D*. Without an infrastructure to monitor and profile the performance and power in each stage, it is impossible to apply these modification and decide which architecture is more efficient.

2.2 Voltage and Frequency Scaling

The fact that power is basically spending energy over time allows design optimization with respect to frequency, and supply voltage. We can reduce power dissipation and as a result develop more reliable designs by lowering the frequency or supply voltage within the defined deadline and without compromising the performance. Fig.2 illustrates the general idea of this scheme. The working frequency of PE_i is reduced

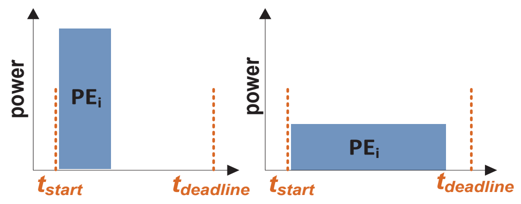


Figure 2: Adjusting *PE* clock frequency

to minimize power dissipation, while meeting the requested deadline.

2.3 Balancing power dissipation by scheduling

Throughout the life-time of a device it is important to balance power dissipation. This can effectively reduce the working temperature of the device, improve reliability, minimize faults, and extend the system life-time. MAVO supports monitoring the mode and the activity intervals of each design element, as well as the amount of their power consumption. Using this information, designer can easily examine scheduling alternatives and power saving opportunities via simple, yet effective design modification. Fig.3 demonstrates this by improved scheduling of the working intervals of two processing element,PE_i and PE_j, to balance the overall power usage, and reducing peaks and temperature at the same time.

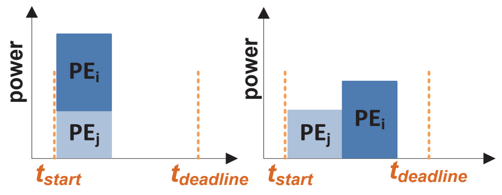


Figure 3: Scheduling Power Dissipation with MAVO framework

2.4 Smoothing Power Spikes

The peak power of the design is among the factors that directly influences the reliability, thermal limitations, cost and size of the device [8]. Fig.4 illustrates the general idea of eliminating low and high spikes. The unwanted power dissipation behavior can be avoided by scaling frequency within the involve units. In an ideal design, peak power should be limited to certain range. In MAVO we are using a simple method to monitor different active process of the design and scale down the frequency, in order to avoid out of range peak power.

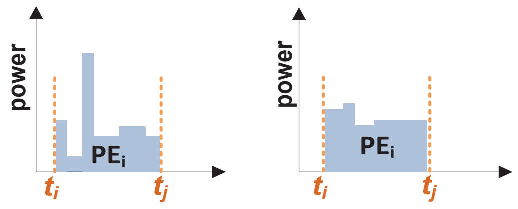


Figure 4: Smoothing power spikes with MAVO framework

2.5 Power shutoff

Finally, in order to reduce static power, a common Dynamic Power Management(DPM) technique is to shutoff the inactive devices. MAVO also supports this approach.

3 Related works

There has been large body of work and research efforts on low-power design, power optimization techniques, and power aware EDA tools, for design at different levels of abstraction. For power and performance estimation at system level, two common steps within all these studies are: generating power models and tracing model simulation to extract information needed by power models. A functional level power analysis approach is used in PETS [10]. PETS uses generic power models while extracting micro architectural activity to tackle the accuracy-speed tradeoff. COMPLEX [5] is a framework for HW/SW codesign at system levels and allows applying hybrid combination of power models from various works for different design components. Wattch [1] and Simple-Power [14] are cycle-accurate power estimators with low speed. A power, performance and area estimator with built-in power models for all types of HW units is presented in McPAT [7]. McPAT has to be used along with a simulator as well. Although power models can work with negligible error, the main limitation at system level is to rapidly collect detailed power traces for different applications. To alleviate this problem, simulators Sniper [2] as an interval simulator and Multi2Sim [13] as functional simulator are

proposed. There are also works that extend SLDLs with additional libraries and allows the designer to insert power data/functions to design manually, such as PowerSC [6] and TLM POWER3 [4].
These power estimators at system level generate general power reports in form of average power consumption, performance, or the trace of total power of the design only. However, our MAVO framework enables the designer to concentrate on any HW or SW part of the design, their working intervals, their power consumption over time, and modes of operation, with user-defined granularity. This feature allows to make critical design decisions and find power optimization solutions easy and rapidly with clear understanding of the design.

4 Approach:MAVO Framework

An overview of the design flow using MAVO is presented in Fig.5. The main developed components consist of a *Monitor* [11], *PowerAnalyzer* API [12], power-time model *Annotator*, and an interactive power-performance *Optimizer*. We developed the *Monitor* and the *PowerAnalyzer* API [12], and evaluated them in terms of accuracy and fidelity for ESL power estimation [11]. Our results confirms that the *Monitor* along with the *PowerAnalyzer* deliver rapid estimates with high fidelity and at minimal cost. In this work we automated the monitoring and power analysis. We also integrated the visualization and interactive optimization capabilities.
As it is shown in Fig.5, the design process at system level starts with a specification model, that reflects the functional behavior of the system, without any notion of time nor power. Next, Processing Ele-

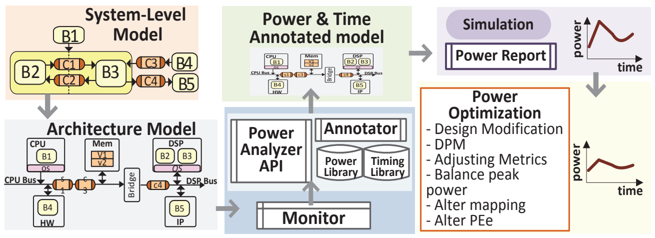


Figure 5: Design flow with MAVO framework

ment (PE) allocations, behavior mappings, scheduling and communication refinements are performed. All these design decisions generate a large design space which needs to be minimized through elimination, based on design constraints. All invalid design options are pruned from the design space and the best design options go to power optimization and performance improvement phase. In this work, we implemented the system level specification models in SpecC language, and the System-on-Chip Environment (SCE) [3] is used for component allocation, architecture mapping, and refinements. Although we picked SpecC language and SCE as the design environment, MAVO can be used in SystemC or other design frameworks similarly. Moreover, each module of MAVO, *Monitor*, *PowerAnalyzer* API, *Annotator*, and power-performance *Optimizer*, can be used separately as needed or integrated into other tools as well. Once the architecture model is ready, the subsequent steps are profiling the model, annotating power and time related functions to the design, and finally the power and performance analysis.

4.1 Monitor

The system specification model represents the functionality of the system. The system level design languages allow the designer to specify the design with timing notion, different communication attributes such as channels, queues, and the format of behaviors executions such as parallel, pipeline, sequential or finite state machines . For power and timing analysis, designer needs to specify the structural model in which processing elements, communication elements, and memories are allocated, and mappings are defined for specification model components. In order to generate the power and timing reports and perform analysis, *Monitor* [11] profiles the trace of different operations executed on processing elements and memory accesses, besides the amount and type of data being transferred over the channels. These traces can be collected with different levels of granularity, for instance, with highest granularity for the whole design, or lower, the trace of every component and communication channel of the design. In this work, we are monitoring with granularity of every basic block. The basic block in this work is defined as in compilers design. The traces of operations, data accesses, and transferred data are generated dynamically through simulation of system level architecture model.

4.2 Power and Performance Annotator

In order to perform power and timing aware simulation, and design exploration, the collected traces are annotated to the structural model. In order to maintain accuracy, the traces are associated with every basic block of the model. Therefore, an *Annotator* is designed to insert power and timing information. Nevertheless, the *Annotator* can support the annotation with higher granularity of behaviors and components as well. The back annotated information includes execution delay, static, and dynamic power dissipated within the corresponding basic block, considering the type of the design component that block is mapped to, its configured operational mode, as well as the communication transactions through the assigned communication unit. In order to process these values, *PowerAnalyzer* API [12] is applied. The *Annotator* is linked to *PowerAnalyzer* API in order to apply the power functions and use generated values for annotations. An example of a basic block annotated with time and power information is presented below.

```
{//basic block: Bi
 {
  Label i:
   waitfor time;
   dissipate(PowerMeterA, Dynamic Power,Static
       Power);
 }
 ...
 d++;
 Ch1.send(d);
}
```

As shown, the *dissipate* function represents the amount of power spent in basic block B_i, in form of dynamic and static power, as well as the *PowerMeterA* that monitors B_i power dissipation. MAVO can support the dynamic and static power monitoring separately, so the designer can focus on any of them as needed. The static power represents leakage power, and because of shrinking in transistor size to sub-micron, static power needs to be investigated as carefully as dynamic power.

4.3 PowerAnalyzer API

The *PowerAnalyzer* API is developed to complement system level models with power notion. *PowerAnalyzer* API [12] is implemented in *C++* and can be used along with any ANSI-C based SLDL. In this API, a set of power related functions is provided to add the dimension of power to system level design. Using the provided functions, the user can specify power related activities and analyze power dissipation in different processing elements, communication elements, behaviors, and globally, both manually or automatically using the *Annotator*.
In order to analzye power consumption, PowerMeters are used. Power dissipation is measured using PowerMeters and can be evaluated for any subsection of the design. As shown in last section, the dissipate function captures the dynamic and static power dissipation of a block which is monitored by a *PowreMeterA*. PowerMeters can get allocated for any component, behavior or block of the design automatically. However, designer can define more PowerMeters for any part of the design for further analysis if needed. The PowerMeters are mainly useful for post-simulation power analysis. For effective power analysis and improving the reliability of system, designer need to monitor global power consumption of the system for peak powers, and temperature analysis. However, for power optimization investigation and balancing peak power, detailed power dissipation traces of design elements, beside behaviors status, is required. For instance, in a design with multiple processing elements, in order to balance the power consumption within elements and schedule their mapped behaviors, it is required to monitor elements activity intervals as well as the amount of their power consumption, apart from profiling entire design power dissipation. Similarly, in order to identify a solution for smoothing a peak power in certain processing element, designer need to study active behaviors on that element during the peak.
To generate a comprehensive analysis of power and performance, the *PowerAnalyzer* API and *Annotator* are working alongside to use monitored power log information, and annotate power and timing related functions generated by *PowerAnalyzer* API. To calculate the dynamic and static power values, user need to

provide power model and configuration files of each element.

4.4 Power Optimizer

The power *Optimizer* is an infrastructure for close evaluation and analysis of the design, through power reports, and identifying power and performance optimization opportunities. These opportunities can be in form of design alteration e.g. changing the weight of computations in different blocks of the design, altering algorithms, changing execution methods like parallelism or pipelining, communication policies, components allocations, and PE mappings. The other group of power saving solutions, can be power optimization methods such as dynamic voltage and frequency scaling, dynamic power management, scheduling or load balancing.
The main role of *Optimizer* is to assist the designer with optimization decisions. *Optimizer* supports generating power and performance analysis for any time interval or subsection of the design to allow the designer to evaluate the design and explore other design options rapidly. For frequency scaling, scheduling and balancing peak power, *Optimizer* can help further and show the working intervals of each element, and involved design elements in peak power.The *Optimizer* assesses PowerMeters and provides numerical logs of power over time. In order to control the size these log files and adjust the precision of this analysis, the user can pick the sampling frequency. The user can also specify any simulation interval to monitor as well. Most importantly, the user can view graphical power dissipation over time and zoom in for specific intervals of any design elements and behaviors. Moreover, the *Optimizer* support merging the reports or stacking up the power dissipation values in different PowerMeters over time.

5 Case Study: Canny Edge Detector

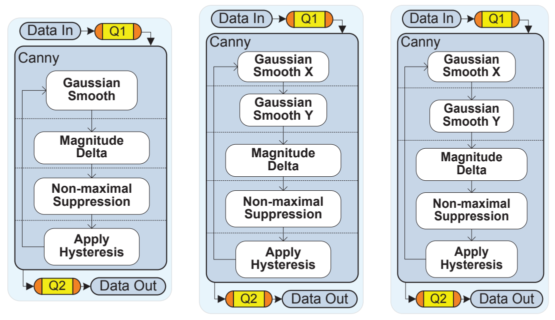
We have investigated the MAVO framework with a Canny edge detector. Canny is a real-life image processing application implemented in 4-stage pipeline configuration. The model was examined on an ARM based processor with two custom HW units for input and output, and 2 HWs for pipeline. All units are communicating through the AMBA BUS.

5.1 Canny: Design Modification

The 4-stage pipeline architecture is suggested by edge detection algorithm which has four major steps. However, after viewing the power and timing reports from MAVO, it become apparent that this architecture is imperfect in term of the pipeline load in each stage. Table 1 shows the power and time consumption of each pipeline stage.

Design	Stage1		Stage2		Stage3		Stage4		Stage5	
	Time (ms)	Power (mW)	Time (ms)	Power (mW)	Time (ms)	Power (mW)	Time (ms)	Power (mW)	Time (ms)	Power (mW)
4-Stage	537	328.3	184.4	77.1	353	72.5	142	38.5	-	-
5-Stage	226	174.6	237.8	149.6	184.4	77.1	353	72.5	142	38.5
3-Stage	226	174.6	237.8	149.6	688.8	188.2	-	-	-	-

Table 1: The delay & average power of pipeline stages



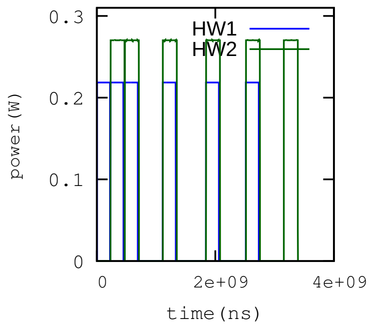
(a) 4-stage pipeline (b) 5-stage pipeline (c) 3-stage pipeline

Figure 6: Canny Architecture

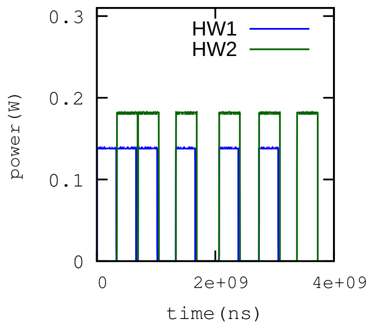
The power and timing results reveal that the *Gaussian Smooth* behavior is computationally expensive and power hungry. In order to balance the pipeline we modified the design to a 5-stage pipeline (CannyA), splitting the Gaussian Smooth behavior in to *X* and *Y* dimensions. In this work the stage 1 and stage 2 of the pipeline are mapped to custom HWs, and rest of the stages are mapped to ARM processor. The applied mappings, makes the Canny architecture works as a 3-stage pipeline configuration. Fig.6 shows the architecture of Canny before and after the modification. Next we evaluate Canny for power optimization. Fig.11(a) shows the power dissipation in each of the design elements.

5.2 Canny: Adjusting Frequency

Using the reports and graphs, the working frequency and supply voltage of each unit can be optimized. In Fig.11(a), a power saving opportunity can be detected for HW1 and HW2, which finish their tasks earlier than the rest of stages. In turn, we lower the frequency of HW1 and HW2 within the performance constraints (CannyB).



(a) Before



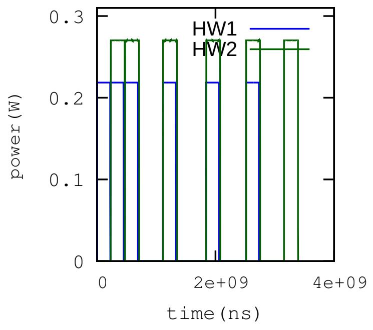
(b) After

Figure 7: Adjusting Frequency for HW1 and HW2 using MAVO

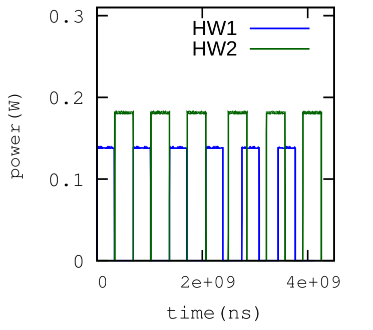
By extending the processing time in stage 1 and 2, the simulation time gets extended as well. This is due to the fact that initially it takes longer to fill the pipeline, however, the pipeline throughput performance remains the same.

5.3 Canny: Power Aware Scheduling

In order to balance power dissipation in whole device, we can schedule the work period of the units such that they have minimum overlaps. For HW1 and HW2 the result of this modification (CannyC) is shown in Fig.8.



(a) Before



(b) After

Figure 8: Adjusting work period for HW1 and HW2 using MAVO

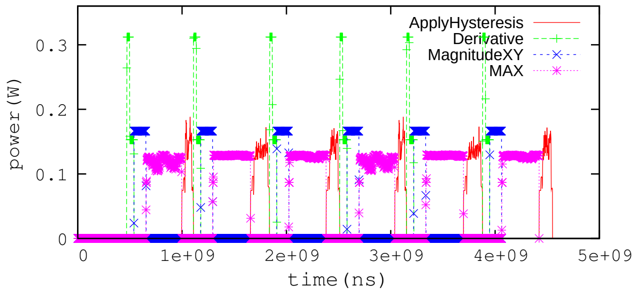
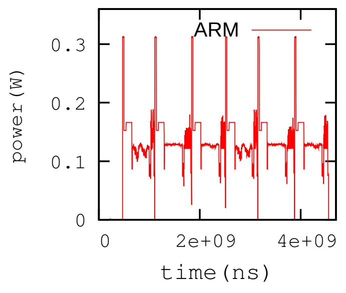


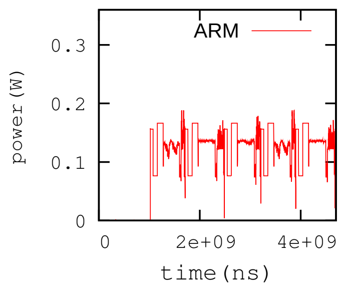
Figure 9: Active processes power dissipation in CPU

5.4 Canny: Smoothing Power Spikes

The total power results from MAVO show that the device is experiencing high peak powers, where the



(a) Before



(b) After

Figure 10: Smoothing power dissipation using MAVO

peak is more than the double the average. In order to smoothen the dissipation MAVO identifies the active processes during the peaks as shown in Fig.9.

Here we decide to scale the frequency for the involved behaviors. Fig.10 shows the results (CannyD). A block performing the floating point operations is responsible for power peaks. We lower the frequency of CPU here and in order to maintain the performance, another integer intensive behavior is scaled with higher frequency instead. Table 2 shows the power and performance of each models. The canny example has been tested with 6 images and it was expected to generated one image in every 0.8 seconds. As shown, MAVO power savings resulted in an optimized design with no performance penalty. The optimized design experiences power fluctuations 8% less, based on comparing the standard deviation of power reports and the power changes range has been reduced by 29%. The difference between the minimum and the maximum of power dissipation over time, considered as the power changes range of the design.

Model	Pipeline Throughput (ms)	Power Range (min,max)	Relative Fluctuation	Power (mW)	Power Saving
CannyA	688.8/800 (86%)	(0.003,0.312)	100%	374.504	+0%
CannyB	689/800 (86%)	(0.003,0.312)	100%	357.113	+5%
CannyC	701.8/800 (88%)	(0.003,0.312)	100%	329.903	+12%
CannyD	738.8/800 (92%)	(0.001,0.204)	71%	315.511	+16%

Table 2: Timing and Performance for Canny Edge Detector after applying each technique

6 Conclusion & Future Work

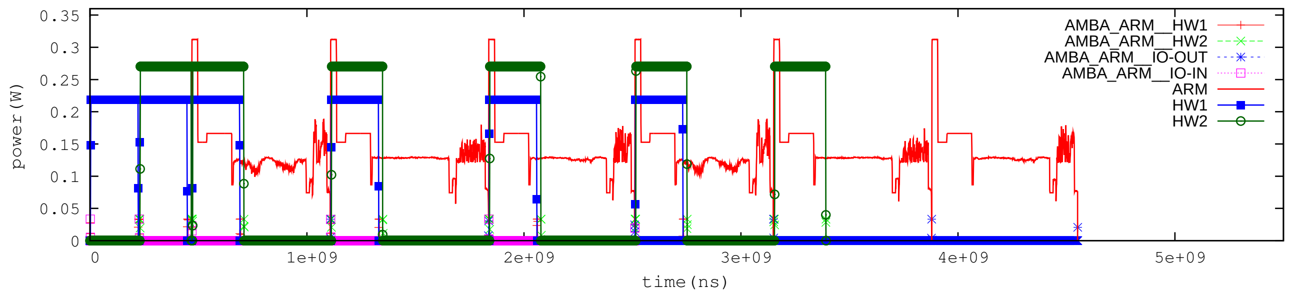
This paper presents a framework to *Monitor*, *Analyze*, *Visualize* and *Optimize* power and performance for low power ESL design. MAVO is a simulation based power and performance estimator and optimizer. A *Monitor* for profiling the system model simulation, an API called *PowerAnalyzer* for computing power numbers using power models, a model *Annotator* for back annotating the power values automatically, and an *Optimizer* to apply power optimization techniques, are developed and integrated in MAVO.
The Canny edge detector has been studied using MAVO, and it is optimized up to 16% for power without compromising performance. The power range has been reduced by 29%. Future work will address integrating an efficient dynamic power manager with thermal and reliability analyzer.

References

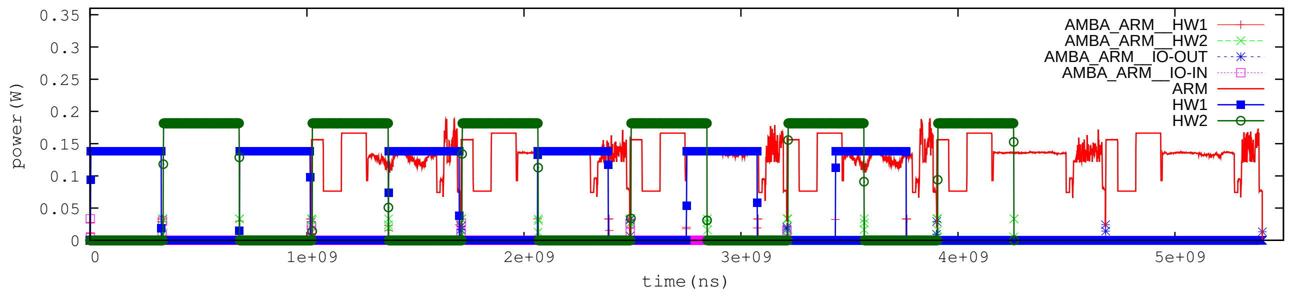
[1] David Brooks, Vivek Tiwari, and Margaret Martonosi. *Wattch: a framework for architectural-level power analysis and optimizations*, volume 28. ACM, 2000.

[2] Trevor E Carlson, Wim Heirman, and Lieven Eeckhout. Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 52. ACM, 2011.

[3] Rainer Dömer, Andreas Gerstlauer, Junyu Peng, Dongwan Shin, Lukai Cai, Haobo Yu, Samar



(a) Canny edge detector



(b) Optimized Canny Edge Detector

Figure 11: Power dissipation of Canny Edge Detector visualized and optimized by MAVO

Abdi, Daniel D Gajski, et al. System-on-chip environment: a SpecC-based framework for heterogeneous MPSoC design. *EURASIP Journal on Embedded Systems*, 2008.

[4] David Greaves and Mehboob Yasin. TLM POWER3: Power estimation methodology for SystemC TLM 2.0. In *Models, Methods, and Tools for Complex Chip Design*, pages 53–68. Springer, 2014.

[5] Kim Grüttner, Philipp A Hartmann, Kai Hylla, Sven Rosinger, Wolfgang Nebel, Fernando Herrera, Eugenio Villar, Carlo Brandolese, William Fornaciari, Gianluca Palermo, et al. The complex reference framework for hw/sw co-design and power management supporting platform-based design-space exploration. *Microprocessors and Microsystems*, 37(8):966–980, 2013.

[6] Felipe Klein, Rodolfo Azevedo, Luiz Santos, and Guido Araujo. Systemc-based power evaluation with PowerSC. *Electronic System Level Design*, pages 129–144, 2011.

[7] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 469–480. IEEE, 2009.

[8] Massoud Pedram. Power minimization in ic design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(1):3–56, 1996.

[9] Jan Rabaey. *Low power design essentials*. Springer, 2009.

[10] Santhosh-Kumar Rethinagiri, Oscar Palomar, Osman Unsal, Adrian Cristal, Rabie Ben-Atitallah, and Smail Niar. Pets: Power and energy estimation tool at system-level. In *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, pages 535–542. IEEE, 2014.

[11] Yasaman Samei and Rainer Dömer. Automated Estimation of Power Consumption

for Rapid System Level Design. In *Performance Computing and Communications Conference (IPCCC), 2014 IEEE 33rd International*. IEEE, 2014.

[12] Yasaman Samei and Rainer Dömer. PowerMonitor: A Versatile API for Automated Power-Aware ESL Design. In *Specification & Design Languages (FDL), 2014 Forum on*. IEEE, 2014.

[13] R Ubal, J Sahuquillo, S Petit, and P López. Multi2sim: A simulation framework to evaluate multicore-multithread processors. In *IEEE 19th International Symposium on Computer Architecture and High Performance computing, page (s)*, pages 62–68, 2007.

[14] Wu Ye, Narayanan Vijaykrishnan, Mahmut Kandemir, and Mary Jane Irwin. The design and use of simplepower: a cycle-accurate energy estimation tool. In *Proceedings of the 37th Annual Design Automation Conference*, pages 340–345. ACM, 2000.