An Energy Behavior Supporting Simulator for Nonvolatile Processor Based Self-powered System.

ABSTRACT

Battery-less devices and energy harvesting techniques are more and more attractive in today's IoT area. However, the intermittent and low-intensity ambient power sources cause insecure quality of service (QoS) of these self-powered IoT applications, which makes it essential to pre-simulate the QoS of an self-powered system and its energy managing strategy under various power conditions. Unfortunately, existing simulator lacks of supports on peripherals in a self-powered system and requires great efforts to implement the hardware-based energy managing schemes, such as NVP, DVFS and near-threshold designs.

This paper proposes EBeSS, an system-level simulator based on GEM5, supporting easy-configurable energy managing strategies on both processor and peripherals. EBeSS develops a configurable simulation object, **virtual device**, to support the functional and energy-related behaviors of peripherals, and **energy message handling framework** to support configured energy managing strategies. A real hardware prototype validates that the error of EBeSS is less that xx%. And a self-powered system design flow is given to reveal the potential of EBeSS on exploring the design space of parameters of a self-powered system (such as capacitor and energy managing scheme) with optimal performance.

1 INTRODUCTION

Battery-less devices are more and more popular in today's IoT area for its light-weight charge-free energy supply system and long lifetime. However, the energy supply quality of these devices is insecure according to different energy sources and different working environment. For instance, solar-power is the most popular ambient energy source, whose power supply intensity various within orders of microwatt to milliwatt according to the time, season, weather condition, altitude, location (indoor/outdoor) and etc. Therefore, developers need to evaluate the quality of service (QoS) of a self-powered system under specific power supply condition before developing an energy harvesting application. Without proper software simulator, developers have to implement a hardware prototype with appropriate energy managing strategies to realize evaluate quality of service (QoS), which costs unacceptable efforts and time.

To tackle this problem, previous works develop different simulation strategies for different types of self-powered devices. Target the systems with pure software energy managing and scheduling algorithms, EDB [1] proposes an energy supporting debugger to simulate the performance of intermittent powered devices. For the devices with hardware optimizations, such nonvolatile processor (NVP) [4, 7–9], NVIO [3], NVRF [10], and near-threshold circuits [], Ma et al. [5] provides a simulating strategy based on verilog source code to process register-transfer level (RTL) evaluation. Furthermore, Gu et al. [2] models the architecture and energy managing behaviors of NVP [8] and proposes NVPsim, a simulator based on GEM5 to explore the processor-level parameter design space of NVP.

Although provided usable simulation approaches for processor and software energy managing algorithms, the above mentioned simulation tools has two main draw backs. Firstly, these simulators cannot support the functional and energy-related behaviors of peripherals. Peripherals, such as sensors, radio transceivers and accelerators, are playing irreplaceable roles in IoT area by taking charge of the interconnections between the system and the environment, and improve the QoS. With intermittent power supply, these external hardware modules may cause inconsistency problems and needs to be safely recovered after sudden outages. Secondly, neither RTL-level simulation [5] nor NVPsim [2] supports flexible and easy configured hardware energy managing techniques such as NVP [8] and dynamic voltage-frequency scaling (DVFS) [].

Focus on these problems, this paper proposes EBeSS, an <u>**E**</u>nergy <u>**Be**</u>havior <u>**S**</u>upporting <u>**S**</u>ystem-level simulator to explore the design space of a NVP based self-powered device. The contributions are listed as follows,

- Based on GEM5, EBeSS provides a configurable virtual device module to support the simulation of a variety of peripherals;
- EBeSS realizes flexible and friendly hardware energy managing behavior supports with the help of the energy message handling (EMH) framework to realize and manage the energy-related behaviors of each hardware module;
- With EBeSS, we provides an example of self-powered system developing and optimizing procedure by exploring the design space of capacitor and energy managing strategy selection with different power traces and benchmarks. The analysis reveals an optimal design edge to improve the performance of self-powered systems.

The rest of the paper is organized as follows. Sec. 2 proposes the architecture of the simulator and introduces the two techniques used to support the energy behavior exploration of NVP and peripherals. Sec. 3 validates the functionality of EBeSS with a real NVP based self-powered system prototype. Sec. 4 explores the performance of different energy behaviors with various power supply profile and application types. Finally, Sec. 5 summarizes the whole paper and looks ahead for more advanced usages of EBeSS.

2 EBESS DESIGN METHODOLOGY

This section demonstrates the design methodology of EBeSS. After introducing the overview of EBeSS structure, we illustrates the two key techniques embedded in the simulator. *Virtual Device* is a configurable module used to simulate the behaviors of peripherals. *Energy Message Handling Framework (EMHF)* provides an energy message interacting structure to manage the energy-related behaviors of each hardware module according to the energy managing strategies.

DAC'2018, May 2018, San Francisco, CA, US
.

2.1 Structure Overview of EBeSS

EBeSS is an GEM5-based system-level simulator for self-powered systems. EBeSS adopts the fundamental usage and logic of kernel architecture in GEM5 and develops two novel components to support peripherals and energy management. EBeSS enables the ability of simulating the energy-related behaviors of self-powered systems, such as non-volatile processors, multi-frequency systems, and multi-voltage-domain systems. The system architecture of EBeSS is shown in Fig. 1.

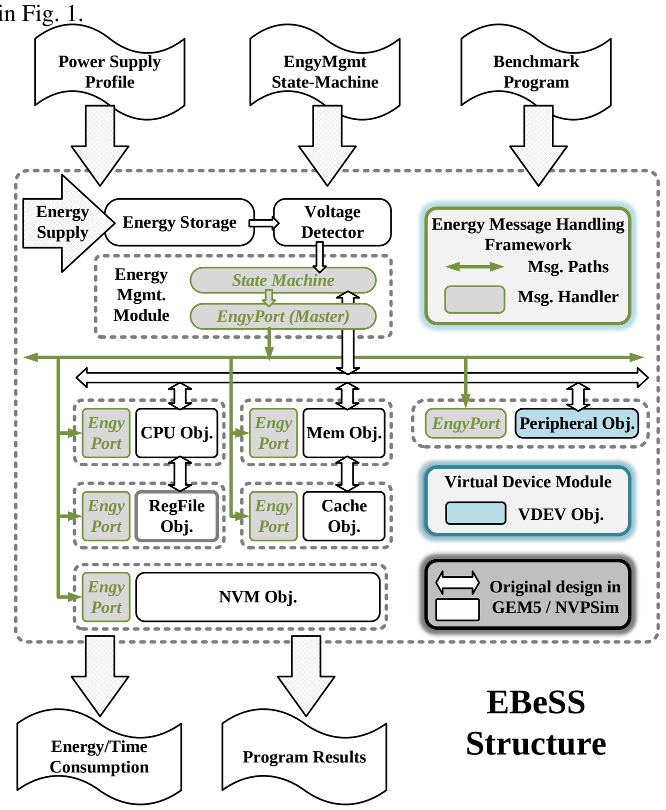


Figure 1: Simulator structure of EBeSS. Two new components are added based on original GEM5 architecture. EMHF generates and handles the energy messages according to energy conditions. The virtual device module creates an configurable object that supports the functionary and energy-related behaviors of peripherals.

To fill the blank of lacking peripheral supports, EBeSS adopts a new simulation object, *virtual device*, which defines the connections and functionary behaviors of various peripherals, as shown in Fig. 1. Virtual device is a configurable, pluggable module connected the system bus, which can be accessed by processor module via virtual addresses. Virtual device supports the fundamental operations of a peripheral, including the peripheral register read/write, initialization, activation and interrupts. Details of the virtual device module is discussed in Sec. 2.2.

EMHF provides a message-based energy managing framework to support configurable energy managing strategies, as shown in Fig. 1. EMHF contains three components, energy management module (EMM), energy ports and energy objects. EMM contains a state machine to generate the state conversion according to user's energy managing strategy. Energy ports define the energy message transmission rules and manage the interconnections between EMM and other hardware components. For each hardware component, EMHF reconstructs it with a new energy-related object, energy object (EnergyObject) allowing energy behaviors defined by energy management strategies as a reaction to energy condition changes. Details of the energy message handling framework is discussed in Sec. 2.3.

2.2 Peripheral Support: Virtual Device

Although GEM5 provides powerful simulation ability to estimate the functionalities of processor and memories, supports are still vacant for various peripherals, such as sensors and radio transmitter, which play indispensable roles in energy harvesting systems. To support the peripherals, this subsection first gives a general model of a peripheral and then introduces the proposed hardware component, virtual device.

Peripheral Modeling Fig. **??** shows a general peripheral structure and its working flow. Most digital peripherals contain a digital part to connect to the system bus and a analog part to contact with the environment. The digital part contains a register file to store the command, execution status and data. The processor accesses the peripheral by writing commands to the *cmd_reg* and read/write the *data_reg*. When the bits in *cmd_reg* changes, the peripheral triggers an operation, such as initialization and execution. A peripheral needs to be initialized to ready state before executing a task. When the task is completed, the peripheral will trigger an interrupt to inform the processor.

Virtual Device Object EBeSS proposes a configurable virtual device object defining the peripheral behaviors and the interconnections between peripherals and the processor. Virtual device is a memory-like module that the processor can make a request by accessing the address of virtual device. Considering that most of the energy harvesting systems are light-weighted and lack of operating system supports to automatic allocate address spaces for virtual devices. EBeSS allows virtual devices to register virtual addresses in the configuration file. When the processor accesses a registered virtual address, the system will forward this access to the physical address of the related virtual device. The address of virtual devices consists of a control byte and a maximum 511B internal memory space bytes as data bytes, as shown in Fig. 2. In the control byte, four bits are used to control the executing status of peripherals. *addr1* is a configurable *init* bit to trigger the initialization operation. *addr2* is a read-only *ready* bit representing whether a device is initialized. *addr5* is a read-only *complete* bit and will be set and trigger an interrupt to the processor, when a virtual device operation is finished. *addr5* is a read-only *busy* bit representing whether the device is during execution and is not accessible if the bit is set. *addr7* is a configurable bit set by CPU to make a request to a virtual device.

Fig. **??** shows an example of temperature sensor.

2.3 Energy Message Handling Framework (EMHF)

EMHF is a message-based event handling framework to support the energy managing strategies in a self-powered system. The framework consists of three components: the informer, the connector and the reactor, as shown in Fig. 3. With these components, EMHF realize

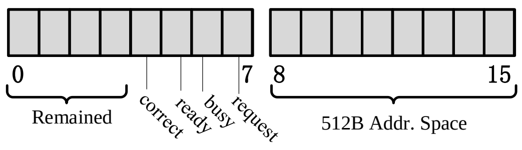
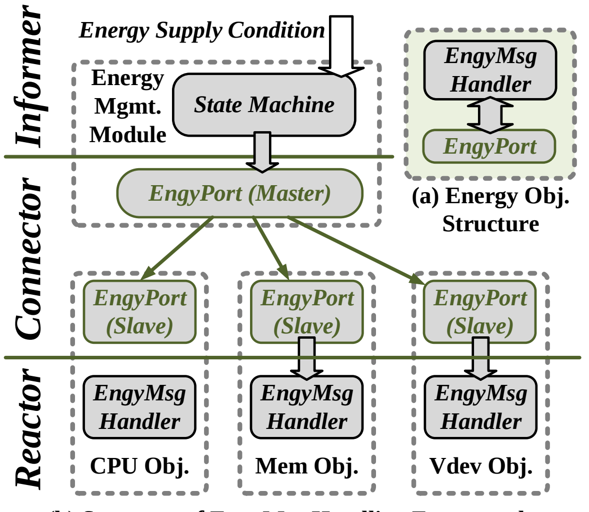


Figure 2: The bit-level explanation of virtual device address definition.

the functionality of energy harvesting, monitoring and all the detailed energy-related behaviors of each hardware component according to energy managing strategies.



(b) Structure of EngyMsg Handling Framework

Figure 3: Structure of the energy message handling framework. The framework contains three components, the informer, the connector and the reactor.

Informer: Energy Management Module. In the energy message handling framework, an energy management module (EMM) is developed to monitor energy changes and broadcast state conversion messages according to a user-defined energy managing strategy. EMM consists of a energy storage, a harvester, a consumer and a state machine, as shown in Fig. 4. During each time tick, the energy harvester collects the income energy from an energy profile and consumer will collect energy consumptions from all the other hardware modules. These energy changes are committed to the energy storage (capacitors). The state machine is configurable to realize and explore the optimal design of the user-defined energy managing strategies. When the supply voltage in the storage reaches certain thresholds, EMM determines state conversion messages according to the state machine and broadcast to the entire system.

Reactor: Energy Object & Energy Message Handler. Except for EMM, all the other hardware modules are defined as reactors, including the processor, memory and virtual devices. In GEM5, the native simulation object for these module supports only the functionary behaviors such as computing and memory accessing operations. To support the energy-related behaviors, EBeSS extend an energy object for each hardware module. An energy object contain an energy message handler, an consumer and is connected to

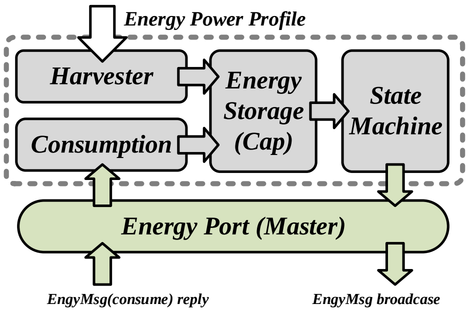


Figure 4: The Structure and work flow of the energy Management Module.

in EMHF framework via energy ports (introduced later), as shown in Fig. 3 (b). The energy message handler is a programmable module to realize the detailed reactions defined in an energy managing strategy. According to developers' needs, the handler will adjust the run-time states and parameters, such as reaction delay, consumption, frequency and register data, to reply the upcoming energy messages. The consumer is used to manage all the run-time energy consumptions, including execution consumption and leakages. Every tick or after a message reaction, the consumer should reply its energy consumption in a specific energy consuming message format to EMM to commit the energy costs. In future, EBeSS will open more authorities that the reactor replying more kinds of energy messages under the premise of system robustness.

Connector: The Energy Port. Energy Ports are interfaces that connect EMM and energy objects, and exchange energy messages.

Each EnergyObject is connected to a specific energy port. There are two types of energy ports: master ports and slave ports. Master ports can actively broadcast energy messages and the slave ports passively accept and reply the energy consuming message and the execution status of their local hardware modules. Master ports can only be connected to slave ports, and vice versa. Each master port can access multiple slaves and maintain a list of its slaves when a simulation is running, while a slave port can only track its unique master. The connections between those ports are established by users with simulation configuration profiles.

2.4 Implementation of A Simple System Energy Behavior

Based on the architecture of EBeSS, we implement a baseline energy behavior strategy, *2-thres*, based on the recover module in NVPSim [2]. The entire system consists of an EMM executing the 2-thres algorithm, a fully nonvolatile simple atomic NVP, a nonvolatile memory and volatile peripherals (virtual devices). In these modules, the behavior of the NVM is omitted because power outages will not affect its states.

2-thres is an energy management method with a power-on threshold (V_{on}) and a power-off threshold (V_{off}). When the supply voltage grows above V_{on}, a *power-on* message is broadcast to the system and all the modules start/restarts. When the supply voltage falls below V_{off}, a *power-off* message is broadcast to the system and all the modules are shut down.

Table 1: Parameter Settings of NVP System Prototype [4].

Param.	Oper. Freq.	power	Mem	RegFile	Cap.
Value	1MHz	0.16mW	512KB	128B	$10\mu F$

A fully nonvolatile simple atomic NVP is defined to match the 2-thres strategy. NVP will save all the states to nonvolatile memories when power failure takes place. In other words, the system pauses by consuming certain amount of energy after a certain time delay. When power-off message arrives, the handler replies with the energy consumption of the backup operation and submit the backup delay to the event queue of GEM5. Then EMM will update the energy storage voltage with the energy consumption and GEM5 will deschedule the events to pause the queue by inserting the backup delay. Similarly, when power-on message arrives, the handler replies the energy consumption of the restore operation and submit the restore delay.

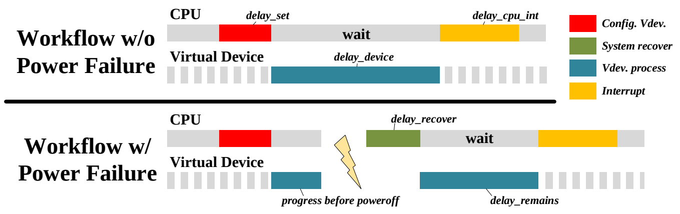


Figure 5: The work flow of virtual devices w and w/o power failures.

The work mode of the virtual devices is shown in Fig. 5. In a normal execution without power failures, the entire work flow contains three stages. First, NVP spends cycles (deley_set) to set up the configurations of a virtual device. Then, the virtual device executes a parallel task (delay_device) and triggers an interrupt request for NVP to reply the execution results (delay_cpu_int). The energy consumption of virtual device in each stage are committed to EMM. When power failure takes place, most of the existing peripherals, such as sensors and transmitters, will lose its volatile states. EBeSS pre-defines an ideal virtual device recovering model that the device can safely keep on the progress after a certain recover procedure (delay_recover) which is executed by NVP.

3 SIMULATOR VALIDATION

To validate the reliability of EBeSS, we compare the experiment results of EBeSS with an existing NVP system [4] which adopts the 2-thres scheme. The system parameters are listed in Tab. 1. The validation compares the execution time and energy consumption with different power profiles and benchmarks. The power profiles are real solar power traces in MIDC database from NREL Solar Radiation Research Laboratory [6]. By executing three benchmarks (*fft*, *queens* and *sort*), Tab. 2 shows that, the difference between the simulation result (time/energy) and the measured result are is within 7.88%/6.38% and the average difference is 3.45%/2.91%. Therefore, EBeSS is proved to have the ability of imitating the behavior of a real NVP based self-powered system prototype.

Table 2: Comparison Between EBeSS and NVP Prototype.

		FFT			QUEENS			SORT		
Power	Metrics	Sim.	Mea.	Err.	Sim.	Mea.	Err.	Sim.	Mea.	Err.
Trace 1	engy./uJ	4.21	4.02	4.51%	7.61	7.45	2.10%	29.2	26.9	7.88%
	time/ms	39.2	36.7	6.38%	77.4	74.3	4.01%	266	253	4.89%
Trace 2	engy./uJ	2.98	2.93	1.68%	5.14	4.97	3.31%	16.2	15.4	4.94%
	time/ms	21.2	20.6	2.83%	51.1	49.6	2.94%	172	166	3.49%
Trace 3	engy./uJ	2.19	2.15	1.83%	2.97	2.91	2.02%	10.9	10.6	2.75%
	time/ms	14.5	14.4	0.69%	24.5	24.5	0.00%	91.1	90.2	0.99%

4 DESIGN SPACE EXPLORATION ON EBESS

Based on EBeSS, this section explores the performance of three energy behaviors of NVP and peripherals under unstable power supply profiles.

4.1 Experiment Settings

The parameter settings are listed in Tab. 3. EBeSS build an ARM based core simulator which executes at 1MHz. This simulator contains a 16MB memory and a 2KB register file. Three peripherals are connected in this system, that are a temperature sensor, an accelerometer and a radio transmitter. The energy consumption values of these devices are listed in Tab. 3. The power profiles are square waves with different duty-cycles. The sample rate of each power profile is 10kHz.

Six benchmarks are used to explore the insight of energy behavior designs, as shown in Tab. 4. *fft*, *queens* and *sort* are three computing dominated benchmarks which is executed on CPU with a few memory accessing operations. *TmpSense*, *AccSense* and *AccTrans* are three peripheral related benchmarks whose I/O operation occupation successively increases.

Three energy behavior schemes are analyzed in the explorations, that are *2-thres*, *DVFS*, and *ModeCvrt*. 2-thres is the example given in Sec. 2.4, where two voltage thresholds are defined to manage system power on and off during execution. As shown in Fig. 7 (a), when the supply voltage grows above power-on threshold V_{on}, the system start to work. When the supply voltage falls below power-off threshold V_{off}, the system is shutdown. DVFS scheme provides three DVFS executing levels from high to low, where the levels with lower voltages are adopted in NVP when power supply voltage reduces. When the supply voltage changes across the thresholds ($V_{\text{d-1}}$, $V_{\text{d-2}}$, $V_{\text{d-3}}$), NVP changes the executing voltage and frequency as shown in Fig. 7 (b). ModeCvrt is a scheme that shuts down different hardware modules when the power supply is not enough. There are three working modes and a power off mode. When the supply voltage is higher than $V_{\text{m-3}}$, all the modules are active. When the supply voltage is between $V_{\text{m-3}}$ and $V_{\text{m-2}}$, the RF transmitter is shutdown. When the supply voltage is between $V_{\text{m-2}}$ and $V_{\text{m-1}}$, only the NVP is active and only computing operations are allowed. When the supply voltage is below $V_{\text{m-1}}$, the system is powered off.

Based on this simulator and the configurations, we analyzes the performance of these energy behavior designs with different power profiles and applications.

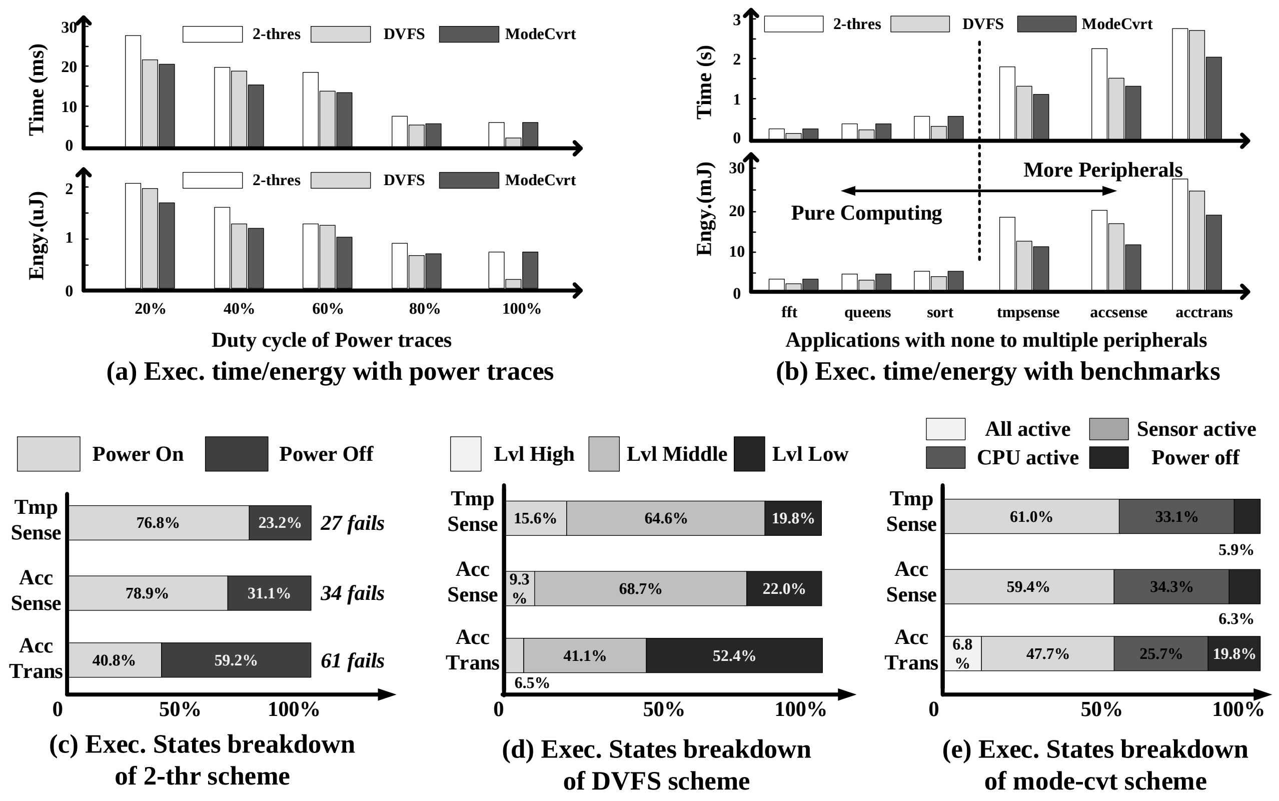


Figure 6: Experiment Results. Figure (a) is the time and energy comparison of the energy behavior schemes executed under the power traces with different duty cycle. Figure (b) compares the execution results by executing 6 benchmarks which has different peripheral occupation. Figure (c)(d)(e) are execution states break downs of the three schemes executing three peripheral-contained benchmarks.

Table 3: Simulator Settings of the Explorations.

CPU	ISA	Freq.	power	Mem	RegFile	Cap.
	ARM	1MHz	0.16mW	16MB	2KB	10μF
Peripheral	Tmp. Sensor		Acc. Sensor		RF Trans.	
	Active	Idle	Active	Idle	Active	Idle
	6.55mW	0.33mW	2.86mW	0.50mW	39.6mW	0.65mW

Table 4: Benchmarks Used in Energy Behavior Exploration.

App.	Type	Hardware Module Used			
		CPU&Mem.	Tmp. Sen.	Acc. Sen.	Trans.
fft	comp.	√	×	×	×
queens	comp.	√	×	×	×
sort	comp.	√	×	×	×
TmpSense	comp.	√	√	×	×
AccSense	comp.	√	√	√	√
AccTrans	comp.	√	√	√	√

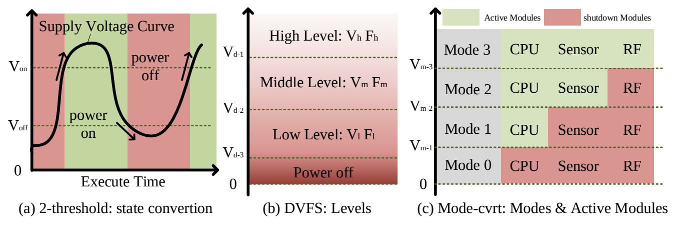


Figure 7: The work flow of virtual devices w and w/o power failures.

4.2 Impact of Power Profiles on Energy Behavior Design

Power supply is one of the most critical impact that affect the execution efficiency of the energy behavior schemes. By scanning the duty cycle of the square wave, power traces with different energy intensity is generate to power the system. Fig. 6 (a) shows the energy consumption and the execution time to complete the TmpSense benchmark. With more efficient power supply, DVFS achieve higher executing speed than the other two schemes. An interesting observation is that, although more energy is cost to complete the same task with high DVFS level, DVFS scheme can still achieve higher energy efficiency. The reason is that, other schemes waste more energy when the capacitor is fully charged, while DVFS has the ability to take advantage of the sufficient supply power which finally increases the energy efficiency.

When the energy supply is insufficient, DVFS and ModeCvrt has similar performance to deal with the low power supply. With the low DVFS level, the system keeps progress with low CPU power consumption and low execution speed. On the other side, ModeCvrt has the ability to use peripheral more energy efficiently by reducing the power leakages when these devices are not executing. Generally, compared with 2-thres, DVFS and ModeCvrt has 22.1% and 17.7% of performance improvement in averages, and can reduce the power consumption by 24.2% and 21.9% in average.

4.3 Impact of Applications on Energy Behavior Design

Application is another important factor that affect the executing performance and energy efficiency of the energy behaviors. Applications in the energy harvesting system contains the computing operations executed on NVP and the I/O operations executed on the peripherals. Since the energy behavior schemes have different adjustment priorities, applications with different I/O operation occupations will have different performance improvements adopting these energy behavior designs.

In this part, benchmark with different I/O operation occupations are used. Fig. 6 (b) shows the time and energy comparison of the energy behavior schemes with these benchmarks. Results show that, with more peripherals used, ModeCvrt has significant energy saving and executing speed improvements. In this condition, DVFS has little improvements because the power consumption of NVP is not dominating when more peripherals are used. DVFS is superior in executing the pure computing benchmarks because its objective is to adjust the execution states of the processor. However, with no peripheral, ModeCvrt lose its advantages to execute the computing tasks and has similar performance with 2-thres.

Fig. 6 (c-f) show the execution state breakdowns of the three energy behavior schemes executing three benchmarks. 2-thres scheme leads to lots of power failures more than other two modules. Because of large energy consumption, AccTrans spends most of the time in power off mode to charge the energy storages. DVFS is also affected by peripherals. Although DVFS focus on the execution of CPU, more peripherals affect the total energy consumption and leads to more low voltage execution time. ModeCvrt performs better because the power gating strategy is more suitable for the peripherals by reducing the peripheral leakages.

4.4 Exploration Conclusion

In conclusion, experiment explores the affect of the power supply density and the application types. Results show that, DVFS has the ability to take better usage of the too low and the too high energy supply. On the other side, ModeCvrt and DVFS have the advantages in adjusting the energy usage of CPU and peripherals, respectively. When the computing operations is dominating, DVFS has better performance to take fully usage of power supply. When the I/O operations are dominating, ModeCvrt is the better choice to avoid the leakages of the idle peripherals.

5 CONCLUSION

Target on energy behavior design exploration, this paper proposes a NVP based energy harvesting system simulator, EBeSS, which supports flexible energy behavior configuration of both NVP and peripherals. With the help of EBeSS, we explores the performance and energy efficiency of three existing energy behaviors with different power profiles and application types. The experiments shows the insights that both DVFS and ModeCvrt schemes have advantages in different power supply and application conditions. Furthermore, EBeSS shows the potential to efficiently and correctly optimize the design parameter settings and validate the data consistency and correctness of the energy management strategies in the transiently powered energy harvesting systems.

REFERENCES

[1] Alexei Colin, Graham Harvey, Brandon Lucia, and Alanson P. Sample. 2016. An Energy-interference-free Hardware-Software Debugger for Intermittent Energy-harvesting Systems. *ASPLOS 2016, Acm Sigops Operating Systems Review* 50, 2 (2016), 577–589.

[2] Yizi Gu, Yongpan Liu, Yiqun Wang, Hehe Li, and Huazhong Yang. 2016. NVPsim: A simulator for architecture explorations of nonvolatile processors. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 147–152.

[3] Zewei Li, Yongpan Liu, Daming Zhang, Chun Jason Xue, Zhangyuan Wang, Xin Shi, Wenyu Sun, Jiwu Shu, and Huazhong Yang. 2016. HW/SW co-design of nonvolatile IO system in energy harvesting sensor nodes for optimal data acquisition. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 154.

[4] Yongpan Liu, Zewei Li, Hehe Li, Yiqun Wang, Xueqing Li, Kaisheng Ma, Shuangchen Li, Meng Fan Chang, Sampson John, and Yuan Xie. 2015. Ambient energy harvesting nonvolatile processors: From circuit to system. In *Design Automation Conference*. 150.

[5] Kaisheng. Ma, Yang Zheng, Shuangchen Li, Karthik Swaminathan, Xueqing Li, Yongpan Liu, Jack Sampson, Yuan Xie, and Vijaykrishnan Narayanan. 2015. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 526 – 537.

[6] Measurement and Instruction Data Center (MIDC). [n. d.]. http://www.nrel.gov/midc/srrl_bms/. ([n. d.]).

[7] Fang Su, Yongpan Liu, Yiqun Wang, and Huazhong Yang. 2016. A Ferroelectric Nonvolatile Processor with 46 Îijs System-Level Wake-up Time and 14 Îijs Sleep Time for Energy Harvesting Applications. *IEEE Transactions on Circuits Systems I Regular Papers* PP, 99 (2016), 1–12.

[8] Yiqun Wang, Yongpan Liu, Shuangchen Li, Daming Zhang, Bo Zhao, Mei-Fang Chiang, Yanxin Yan, Baiko Sai, and Huazhong Yang. 2012. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *2012 Proceedings of ESSCIRC (ESSCIRC)*. IEEE, 149–152.

[9] Zhibo Wang, Yongpan Liu, Albert Lee, Fang Su, Chieh-Pu Lo, Zhe Yuan, Jingyang Li, Chien-chen Lin, Wei-Hao Chen, Hsiao-Yun Chiu, Wei-En Lin, Ya-Chin King, Chrong-Jung Lin, Pedram Amiri, Kang-Lung Wang, Meng-Fan Chang, and Huazhong Yang. 2017. A 65-nm ReRAM-Enabled Nonvolatile Processor With Time-Space Domain Adaption and Self-Write-Termination Achieving >4x Faster Clock Frequency and >6x Higher Restore Speed. In *IEEE Journal of Solid State Circuits (JSSC)*. IEEE.

[10] Zhibo Wang, Fang Su, Zeiwei Li, Xueqing Li, Ryuji Yoshimura, Takashi Naiki, Takashi Tsuwa, Takahiko Saito, Zhongjun Wang, Koji Taniuchi, Meng-Fan Chang, Huazhong Yang, and Yongpan Liu. 2017. A 130nm FeRAM-based parallel recovery nonvolatile SOC for normally-OFF operations with 3.9x faster running speed and 11x higher energy efficiency using fast power-on detection and nonvolatile radio controller. In *Symposia on VLSI Technology and Circuits*. IEEE, C336–C337.