December 2016

Improving the Efficiency of Energy Harvesting Embedded System

Yukan Zhang
Syracuse University

Abstract

In the past decade, mobile embedded systems, such as cell phones and tablets have infiltrated and dramatically transformed our life. The computation power, storage capacity and data communication speed of mobile devices have increases tremendously, and they have been used for more critical applications with intensive computation/communication. As a result, the battery lifetime becomes increasingly important and tends to be one of the key considerations for the consumers. Researches have been carried out to improve the efficiency of the lithium ion battery, which is a specific member in the more general Electrical Energy Storage (EES) family and is widely used in mobile systems, as well as the efficiency of other electrical energy storage systems such as supercapacitor, lead acid battery, and nickel–hydrogen battery etc. Previous studies show that hybrid electrical energy storage (HEES), which is a mixture of different EES technologies, gives the best performance. On the other hand, the Energy Harvesting (EH) technique has the potential to solve the problem once and for all by providing green and semi-permanent supply of energy to the embedded systems. However, the harvesting power must submit to the uncertainty of the environment and the variation of the weather. A stable and consistent power supply cannot always be guaranteed. The limited lifetime of the EES system and the unstableness of the EH system can be overcome by combining these two together to an energy harvesting embedded system and making them work cooperatively.

In an energy harvesting embedded systems, if the harvested power is sufficient for the workload, extra power can be stored in the EES element; if the harvested power is short, the energy stored in the EES bank can be used to support the load demand. How much energy can be stored in the charging phase and how long the EES bank lifetime will be are affected by many

factors including the efficiency of the energy harvesting module, the input/output voltage of the DC-DC converters, the status of the EES elements, and the characteristics of the workload.

In this thesis, when the harvesting energy is abundant, our goal is to store as much surplus energy as possible in the EES bank under the variation of the harvesting power and the workload power. We investigate the impact of workload scheduling and Dynamic Voltage and Frequency Scaling (DVFS) of the embedded system on the energy efficiency of the EES bank in the charging phase. We propose a fast heuristic algorithm to minimize the energy overhead on the DC-DC converter while satisfying the timing constraints of the embedded workload and maximizing the energy stored in the HEES system. The proposed algorithm improves the efficiency of charging and discharging in an energy harvesting embedded system.

On the other hand, when the harvesting rate is low, workload power consumption is supplied by the EES bank. In this case, we try to minimize the energy consumption on the embedded system to extend its EES bank life. In this thesis, we consider the scenario when workload has uncertainties and is running on a heterogeneous multi-core system. The workload variation is represented by the selection of conditional branches which activate or deactivate a set of instructions belonging to a task. We employ both task scheduling and DVFS techniques for energy optimization. Our scheduling algorithm considers the statistical information of the workload to minimize the mean power consumption of the application while satisfying a hard deadline constraint. The proposed DVFS algorithm has pseudo linear complexity and achieves comparable energy reduction as the solutions found by mathematical programming. Due to its capability of slack reclaiming, our DVFS technique is less sensitive to small change in hardware or workload and works more robustly than other techniques without slack reclaiming.

Improving the Efficiency of Energy Harvesting Embedded System

by

Yukan Zhang

B.S., Nankai University, China, 2006
M.S., Binghamton University, 2009

Dissertation

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical & Computer Engineering.

Syracuse University

Decemeber 2016

Table of Content

Chapter 1 Introduction 1
1.1 An Overview of Energy Harvesting and Energy Storage Techniques 1
1.1.1 Energy Harvesting Techniques 2
1.1.2 Energy Storage Techniques 5
1.1.2.1 Conventional Energy Storage 5
1.1.2.2 Hybrid Electrical Energy Storage 8
1.2 Power Consumption on Conversion and Transfer 9
1.3 Power Management for Variable Workload 10
1.4 Thesis Contributions 11
Chapter 2 Improving Charging Efficiency with Workload Scheduling and DVFS 16
2.1 Related Work and System Model 22
2.1.1 Related Work 22
2.1.2 Energy Harvesting Embedded System Model 23
2.2 The DC-DC converter 23
2.2.1 The DC-DC Converter Model 24
2.2.2 Approximation of the Power Consumption of the DC-DC Converter 26
2.3 Task Scheduling for Efficient EES Charging 30
2.3.1 Problem Definition For Workload Schedule 30
2.3.2 Task Scheduling at the Fixed Harvesting Rate 31
2.3.2.1 Scheduling of Two Charging Phases with Very Short Duration 31
2.3.2.2 Scheduling of Multiple Charging Phases with Equally Short Durations 37
2.3.2.3 Scheduling of Arbitrary Charging Phases 39
2.3.3 Task Scheduling at Varying Harvesting Rate 39
2.3.3.1 Scheduling with Monotonically Decreasing Harvesting Rate 40
2.3.3.2 Scheduling with Monotonically Increasing Harvesting Rate 41
2.3.3.3 Neural Network-Based EES Bank Energy Prediction Model 42
2.3.4 Task Scheduling Algorithm to Improve Charging Efficiency 47
2.4 Experimental Results for Workload Schedule 48
2.4.1 Scheduling for Two Charging Phases 49
2.4.2 Scheduling Results for Multiple Tasks 51
2.5 DVFS Considering EES Charging Properties 54
2.5.1 EES Bank Charging Characteristics and Motivational Example 55
2.5.2 Problem Definition 57
2.5.3 A Dynamic Programming- Based DVFS Algorithm 58
2.6 Experimental Results For DVFS Algorithm 61
2.6.1 Evaluation of Proposed DVFS Algorithm 62
2.7 Chapter Summary 65

Chapter 3 Improving Energy Efficiency for Energy Harvesting Embedded Systems 67
3.1 Related Work 70
3.2 Architecture for Green Powered HEES 71
3.2.1 Energy Harvesting System 71
3.2.2 Balanced Reconfiguration of the HEES Bank 72
3.3 Efficient Heuristic Energy Management Algorithm 73
3.3.1 Problem Formulation 73
3.3.2 Observations of the DC-DC Converter 74
3.3.3 The Optimal Vcti for Discharging 76
3.3.4 Efficient Energy Management Algorithm 77
3.4 Experiment Results 80
3.4.1 Constant Input Power Results 81
3.4.2 Variable Sun Power Charging System 84
3.5 Chapter Summary 84
Chapter 4 Task Scheduling and Mapping for Applications on Heterogeneous Multi-Processor Systems 86
4.1 Related Works 89
4.2 Application and Hardware Architecture Models 91
4.3 Task Mapping and Scheduling 93
4.3.1 Minimum Average Makespan CTG Scheduling 94
4.3.2 Balancing Energy and Performance in a Heterogeneous Platform 96
4.3.3 Adding Control Edges to the CTG 98
4.4 DVFS Based on Slack Reclaiming 99
4.5 Experimental Results 106
4.5.1 Comparison with Random Mapping 110
4.5.2 Energy Consumption under Different System Utilization 111
4.5.3 The Effectiveness of Slack Reclaiming Algorithm 114
4.5.4 Sensitivity to Branch Probability Change 117
4.6 Conclusion 119
Chapter 5 Conclusions and Future Directions 121
5.1 Conclusion 121
5.2 Future Directions 123
Bibliography 126

Chapter 1 Introduction

Mobile and portable devices have become an indispensable part of our everyday life. They are used for more and more applications with intensive computation/communication, which are usually power-hungry. A recent survey shows that longer battery life is more important to user's satisfactory than any other features for mobile embedded systems such as smartphones and tablet computers [70]. The importance of battery life is ranked even before those features that the sellers always use to attract people, such as the ease of use and screen size. Moreover, the LG Electronics Smartphone "Low Battery Anxiety" Survey investigated a random sample of 2,000 smartphone users in U.S. It found that nearly nine out of 10 people "felt panic" when their phone battery drops to 20 percent or lower [71]. Therefore, despite the big screen size, powerful processor and fancy operating system, the truth is that the long battery life is the foundation of excellent user experience.

1.1 An Overview of Energy Harvesting and Energy Storage Techniques

In this section, we give a brief overview of general energy harvesting and energy storage techniques. Detailed information of existing works in each category will be presented later in the thesis.

1.1.1 Energy Harvesting Techniques

Energy harvesting, which derive energy from ambient background as energy source, is a promising technique to extend the battery life. These free and clean energy around us can potentially be converted to electric energy. For example, the solar power, wind power, human motion and radio frequency energy can be collected and converted to electrical power by different energy harvesting modules. Existing energy harvesters includes Photovoltaic (PV) cells for solar power, wind turbine for wind power, piezoelectric cantilevers for vibration, and electromagnetic generators for Radio Frequency (RF) energy.

Wind energy harvesting is a relatively matured technique and windfarms are now connecting to the grid. Large windfarms, e.g. the facility near Stockton, CA, contain up to 5000 modern wind turbines [72]. These large wind turbines can achieve efficiencies of up to 40%. The theoretic power P available in the wind impinging on a wind driven generator is given by: $P = \frac{1}{2}CA\rho v^3$, where C is an efficiency factor known as the power coefficient which depends on the machine design, A is the area of the wind front intercepted by the rotor blades (the swept area), ρ is the density of the air (averaging 1.225 Kg/m^3 at sea level) and v is the wind velocity [73]. Hence, the power collected related to area swept by the blades, the density of the air and to the cube of the wind speed. Wind power has the advantage that it is normally available 24 hours per day, unlike solar power which is only available during daylight hours. Unfortunately the availability of wind energy is less predictable than solar energy. And not all the wind power can be captured due to the conversion techniques. The efficiency of the wind turbine depends on the actual wind speed. For the most popular three-blade design, its maximum conversion efficiency is around 40% at an 18 mph wind. However, the wind speed is not stable around the optimal

conversion point. The average efficiency is around 20% [72]. In addition, the large-size wind turbine supporting high power system well is not suitable for low-power devices.

Radio frequency energy is broadcasted from billions of radio transmitters, including mobile phones, portable radios, mobile base stations, and broadcast stations. Radio frequency energy harvesting is poised to serve as an energy source to low-power devices, which makes these products more stable, reliable and eco-friendly. It supplies power to various kinds of wireless sensing node with 24-hour operation [85]. The Friis transmission equation relates the received (P_r) and transmitted (P_t) power with the distance R as: $P_r = P_t G_t G_r (\lambda/4\pi R)^2$ where Gt and Gr are antenna gains, and λ is the wavelength of the transmitted signal [74]. Hence, the converted power depends on the available RF power, the choice of antenna and frequency band. Also, the received signal strength, diminishes with the square of the distance. The energy transmitted from the RF sources can be very high up to 30W for 10GHz frequency, but only a small amount can be scavenged in the real environment. The rest is dissipated as heat or absorbed by other materials [75].

Kinetic energy like vibration can be harvested by micro power generators and can become power sources for fitness wristbands, wireless sensors and other portable applications. Micro power generators have three common mechanisms: piezoelectric, electromagnetic, and electrostatic. But low frequency vibration such as human motion is only sufficient to generate an extremely low output of power. To overcome this problem, [86] developed a new electrostatic micro power generator. This generator shows 40μW of power output at very low frequency vibration (2Hz, 0.4G). However, the low vibration energy still faces the problem of instability. In the Netherlands, A dance club has a floor that harnesses the energy created by the dancers' steps. Designed by a Dutch company the Sustainable Dance Club, the floor based on the piezoelectric

effect can generate 2 to 20 Watts of electricity, depending on the impact of the patrons' feet. The designers hope it can supply 60% of the club's electricity in one day. But for now, due to low frequency and instability, it's just enough to power LED lights in the floor.

In our work, we adopt the photovoltaic cells as the energy harvesting module. The solar energy harvesting is a popular and matured technology in both research and practice. Particularly, different sizes of solar panels are offered in the market from portable solar charger to solar panel backpack for mobile embedded systems. The U.S. solar market is set to grow a staggering 119 percent in 2015, published in conjunction with the Solar Energy Industries Association (SEIA). In 2016, the residential and commercial markets will also experience strong growth in 2016 [76]. Two main types of solar panel in the market are crystalline silicon and thin-film solar panels. In 2011, about 95% of all shipments by U.S. manufacturers to the residential sector were crystalline silicon solar panels. Crystalline silicon forms the basis of monocrystalline and polycrystalline silicon solar cells. Monocrystalline cells has high purity of silicon leading to high efficiency up to 40% but high cost, and polycrystalline cells lower the purity of silicon leading to related low efficiency of 16% and low cost. Much cheaper solar panel is thin-film solar panels with efficiency between 7%-10%. However it costs large space [77]. A monocrystalline 3VDC/40 mA miniature solar cell using a commercial solar module analyzer can achieve the output voltage of 1.958 V at 300 W/m^2 and 3.0V at 510 W/m^2 solar irradiation condition.

Factors may affect the solar power includes shadow effect caused by cloud and the ambient temperature which affects the output voltage. In order to guarantee a high conversion and transfer efficiency, Maximum Power Point Tracking (MPPT), which analyzes the output current- voltage curve of solar panel and apply the proper load to obtain maximum power for any given environmental conditions, and Maximum Power Transfer Tracking (MPTT), which makes

sure the maximum amount of power goes into the system while the power converter connected with the solar panel is considered, are usually performed [12].

1.1.2 Energy Storage Techniques

1.1.2.1 Conventional Energy Storage

One of the major obstacles of applying energy harvesting technique to real-time embedded system is that the harvesting power is not constant and may have large fluctuations. Due to many factors, for example, shadows, temperature and angle of sunlight affects the amount of solar energy that can be collected by the PV cells. For a practical implementation of such system, electrical energy storage element is usually employed as a buffer to compensate the power fluctuation caused by the energy harvesting. An Electrical Energy Storage (EES) element is an energy reservoir that can stores electrical energy when it is ample and supplies energy when necessary. With such an EES element integrated in the system, surplus energy can be stored while the harvesting rate is high. When the harvesting rate is low, the EES element can help the energy harvesting system or work as a single power source for the system.

Table 1.1 presents different types of ESS elements in the market and their properties. In this table, the power capacity is defined as output power per unit volume; the energy density is defined as energy stored per unit volume; the leakage means self-leakage of the EES element; the cycle life is defined as the number of complete charge/discharge cycles that the battery is able to support before its capacity falls under 80% of it's original capacity; the cycle efficiency gives the ratio between the power transferred and the power loss during the transfer; and the cost is the cost per unit of the storage [8].

EES elements can roughly be divided into two categories, batteries and supercapacitors. In the battery category, some common batteries include the Li-ion battery, the NiMH battery, the lead-acid battery and the NiCd battery are presented. They have different performance metrics, which leads to different usages. For example, the lead-acid battery has low cost but short cycle life. However, it can provide the high current demand, so it is used in the automobile to supply the power for starter. The NiCd battery is much smaller than the lead-acid battery. After being invented in 1988, due to its high cycle efficiency, the NiCd battery are used to account for 8% of all portable secondary (rechargeable) battery sales in the EU, and in the UK for 9.2% (disposal) and in Switzerland for 1.3% of all portable battery sales [79]. However, the NiCd battery is replaced by NiMH battery very quickly due to its heavy environmental hazard content. The NiMH battery is even smaller than the NiCd battery in size and is much safer. Now, it is widely in the high-drain devices like HD digital cameras and electric vehicles such as the General Motors EV1, Honda EV Plus, Ford Ranger EV and Vectrix scooter, and Hybrid vehicles such as the Toyota Prius, Honda Insight, Ford Escape Hybrid, Chevrolet Malibu Hybrid and Honda Civic Hybrid [80].

The most popular battery in current market is the Li-ion battery, which has high efficiency, high cycle life, but very high cost. It is usually made in a very small size and widely applied in the embedded or mobile systems, such as cell phones, laptops, tablets, digital camera and etc. However, because of its high cycle efficiency and very high energy density, Li-ion batteries are also used for propelling a wide range of electric vehicles such as aircraft, electric cars, hybrid vehicles, advanced electric wheelchairs, radio-controlled models, model aircraft and electrical robots including the Mars Curiosity rover [81].

Metrics	Batteries				Supercapacitor
	Li-ion	NiMH	Lead-acid	NiCd	
Power capacity	High	High	High	High	Ultra high
Energy density	Very high	Medium	Low	Low	Very low
Leakage	Low	Low	Low	Low	High
Cycle life	Medium	Short	Short	Medium	Very long
Cycle efficiency	High	Low	Low	High	Very high
Cost	High	Medium	Very low	Low	Very high

Table 1.1 Performance Metrics of Commercial Electrical Energy Storages [8]

Another kind of EES element, supercapacitor, currently is very popular in research. The supercapacitor has high power density, high cycle efficiency but low energy density. It typically stores 10 to 100 times more energy per unit volume than electrolytic capacitors, accepts and delivers charge much faster than batteries, and tolerates many more charge and discharge cycles than rechargeable batteries [84]. Moreover, the supercapacitor has less rate capacity effect compared with the battery, which make it able to provide high current to support the peak load demand. However, it is 10 times larger than conventional batteries for a given charge at the same capacity. In addition, it has very high self-leakage. It can lose all its energy in one day after being fully charged [8]. Hence, the supercapacitor is an ideal EES element for temporary energy storage. It is usually used for rapidly and frequently being charged and discharged, and also for shaving the peak current demand. Recently, more and more devices powered by supercapacitor are offered in the market. For example, portable speakers powered by supercapacitors were commercially available [82] as of 2013. As another example, a cordless electric screwdriver with supercapacitors for energy storage has about half the run time of a comparable battery model, but can be fully charged in 90 seconds [83]. Supercapacitors can provide backup or emergency shutdown power to low-power equipment such as RAM, SRAM, micro-controllers and PC Cards.

They are also the power source for low energy applications such as automated meter reading (AMR) equipment or for event notification in industrial electronics [84].

1.1.2.2 Hybrid Electrical Energy Storage

Each of the aforementioned homogenous EES elements has its own pros and cons. In 2010, the hybrid electrical energy storage system [9] has been proposed to overcome the drawbacks of the individual EES banks element while maintaining their strengths. The Hybrid Electrical Energy Storage (HEES) system connects different EES elements to achieve a good performance metric including high charge and discharge cycle efficiency, high energy density, high power delivery capacity, low cost per unit of storage, long cycle life and low leakage.

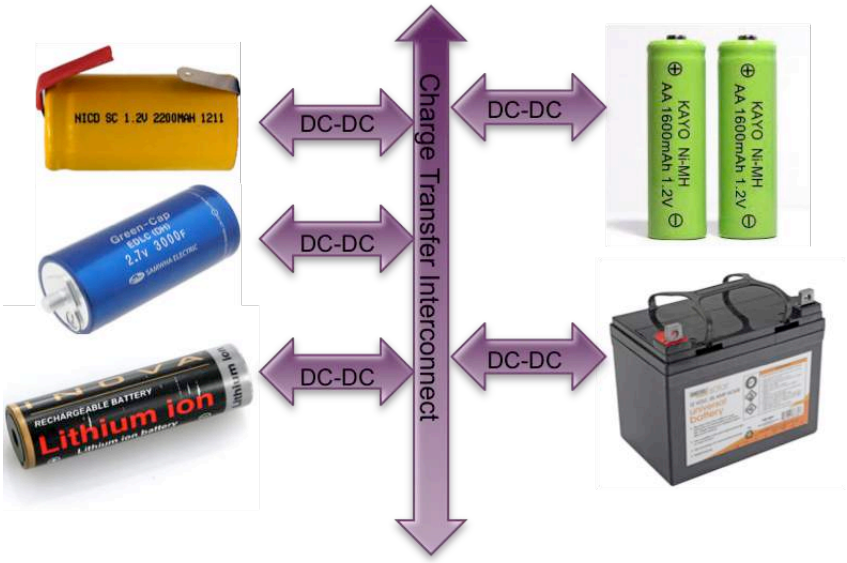


Figure 1.1 the Hybrid Electrical Energy Storage System

Figure 1.1 shows the HEES system proposed in [9]. In this block diagram, several different types of EES elements are connected with each other through charge transfer interconnect (CTI) and DC-DC converters. Each EES element is connected with two DC-DC

converters for both charging and discharging. These converters enable charge transfer incoming to and outgoing from EES elements through CTI by maintaining the voltage-level compatibility and supply a constant current and voltage. The power migration path in the HEES system can be simple wires, which only allow multiple-to-one charging/discharging operation, or expensive crossbar network that enables many-to-many charging/discharging operation. Several recent research works have addressed charge management in such system. For example, charging replacement scheme is discussed in [10]. It aims at selecting the EES bank(s) to be discharged to support a given load demand. Charge allocation scheme is also studied in [11], which solves an optimization problem to decide which EES bank to be used for energy storage when the system is in charge mode. Finally the charge migration problem is studied in [12] that determines how energy should be transferred internally from one EES bank to another.

Our work aims at integrating the HEES system with the energy-harvesting device as a power supply to support the load demand and extend the battery life.

1.2 Power Consumption on Conversion and Transfer

The DC-DC converters are important components in the system and their energy efficiency has drawn lots of attentions in recent works. According to a measurement performed on the power conversion circuits in the Qualcomm snapdragon processors, the voltage regulators such as DC-DC converters can consume around 25% to 40% power from battery. Figure 1.2 shows the measurement of power conversion efficiency of Qualcomm MDP MSM8660 [13]. Many works are proposed to reduce the power loss during the charge transfer on DC-DC converters in the energy storage system. In [14] [11][10][12], the author tried to minimize the power consumption on the DC-DC converter through controlling its input/output voltage/current.

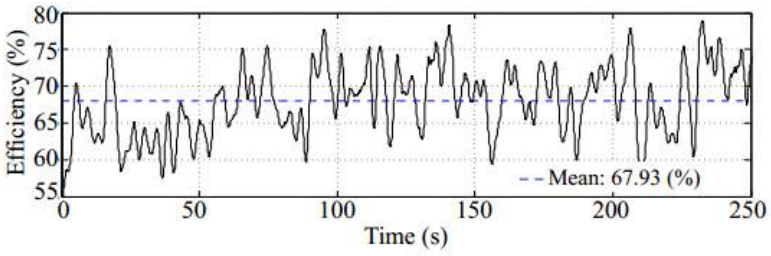


Figure 1.2 Measured of traces of the power conversion efficiency in the Qualcomm Snapdragon MDP MSM8660 [13]

In our work, the main approach to improve the energy efficiency is to reduce the energy overhead on the DC-DC converter during both charging and discharging phases while maximizing the energy stored in the HEES system and satisfying the task deadline constraints of the embedded workload.

1.3 Power Management for Variable Workload

With wide usage of green energy, energy harvesting system has been successfully applied in various applications, such as the solar powered laptop, SOL [78], and the autonomous space exploration rover, NASA Sprit, and the Opportunity mars rover [2]. Moreover, Internet giants like Google and Facebook announced plans on developing solar powered drones to deliver Internet access [3]. These robotic systems are able to carry relatively large size/capacity EES system to sustain their long operation cycles in extreme conditions and to store the extra energy harvested from environment; they are also integrated with embedded systems to do complex computation tasks, like data communication, image processing and recognition etc.

In addition to energy harvesting techniques and techniques to improve the efficiency of the power system, power management has been adopted to reduce application's energy demand

to extend the battery life while guarantee the system performance. These works include Dynamic Power Management (DPM) [15][16][17] and Dynamic Voltage and Frequency Scaling (DVFS) [18][19][20]. Most previous works [23][24][25][26][27][28][29] consider power aware task mapping, scheduling and DVFS for applications with fixed workload. However, this is not always true for real applications, whose workload depend on the input at runtime and may vary dramatically. In our work, we consider the power management techniques for applications with variable workload.

With the DC-DC converters presented in the system, existing power management techniques, which aim to minimizing the load energy consumption does not always result in the most harvesting energy stored in the EES bank. Because the power loss on the DC-DC converters depends on factors like the input power to the DC-DC converter, the terminal voltage of the EES bank, and etc., which will be affected by either DPM or DVFS. In this work, we also investigate the impact of workload scheduling and DVFS of the embedded system on the storage efficiency of the EES bank.

1.4 Thesis Contributions

In this thesis, we present several techniques to improve efficiency of the energy harvesting embedded system.

In the energy harvesting embedded system, if the harvested power is sufficient for the workload, extra power will be used to charge EES banks. How much power can be stored is affected by many factors including the efficiency of the energy harvesting module, the input/output voltage of the DC-DC converters, the status of the EES elements, and the characteristics of the workload.

This thesis investigates the impact of workload scheduling and DVFS of the embedded system on the storage efficiency of the EES bank. We present a simplified but accurate power consumption model of the DC-DC converter. Based on the approximated DC-DC converter model, we analytically prove that an optimal workload schedule is to always execute high power task first while the harvesting rate is fixed or monotonously decreases. Under the other conditions when the harvesting rate is changing, we developed an accurate energy prediction model to estimate the amount of energy that can be stored in the EES given its current status and the given workload. The model helps to choose a workload-scheduling algorithm that stores the maximum energy in the EES bank. The impact of DVFS of the workload on the storage efficiency of the EES bank is also considered. A dynamic programming based approach is presented to decide task execution voltage/frequency and maximize the energy stored in EES bank.

For those devices with more complex power supply system, which supports dynamic reconfiguration and variable CTI voltage, we then introduce a fast heuristic algorithm to search for the optimal HEES and CTI configuration that improves the efficiency of charge allocation and replacement. The goal of our algorithm is to minimize the energy overhead on the DC-DC converter and maximize the energy stored in the HEES system. Based on the previously developed DC-DC converter power consumption model, the optimal operating point of the system can be analytically solved. Integrated with the dynamic reconfiguration of the HEES bank, our algorithm provides energy efficiency improvement and run-time overhead reduction compared to previous approaches.

When the harvesting power is short, the load demands are supported by the EES bank. Many works have been proposed to reduce the energy demand of applications through DPM or

DVFS. Instead of the fixed workload, our research considers the uncertain workload on real time applications. One of the major influences of the workload fluctuation is the selection of conditional branches which activate or deactivate a set of tasks. In this work, we capture the workload dynamics using Conditional Task Graph (CTG) and propose a set of algorithms for the optimization of task mapping, task ordering and dynamic voltage/frequency scaling of CTGs running on a heterogeneous multi-core system. Our mapping algorithm balances the latency and the energy dissipation among heterogeneous cores in order to maximize the slack time without significant energy increase. Our scheduling algorithm considers the statistical information of the workload to minimize the mean power consumption of the application while maintaining a hard deadline constraint. The proposed DVFS algorithm has pseudo linear complexity and achieves comparable energy reduction as the solutions found by mathematical programming. Due to its capability of slack reclaiming, our DVFS technique is less sensitive to small changes in hardware or workload and works more robustly than the DVFS techniques without slack reclaiming.

The contributions of this thesis are outlined as the following:

- Although there have been many publications on battery aware task scheduling, most of them consider the non-linear property of battery when it is in discharge mode [33]. To the best of our knowledge, this is the first work that optimizes the task scheduling to increase the efficiency of EES charging process. Furthermore, many of the existing works on EES efficiency optimization either ignore the impact of DC-DC converters ([30] and [31]) or minimize the energy waste on DC-DC using dynamic voltage selection with an assumption of fixed task scheduling [14]. This is also the first work that investigates the impact of task scheduling on the energy efficiency with the consideration of the DC-DC converters. Our scheduling algorithm is rigorously

proved to be optimal under certain conditions. The scheduling algorithm under consistently decreases or fixed harvesting rate has low complexity as it only has to sort the tasks based on their power consumptions. For more general scenarios, we propose a neural network based energy prediction model that could accurately estimate the SOC of the EES bank given related parameters for the workload scheduling. Based on the energy prediction model, we also propose a dynamic programming based voltage/frequency scaling algorithm which maximizes the harvesting energy stored in the EES bank.

- Our work for improving energy efficiency in the energy harvesting embedded system considers both charge allocation and charge replacement together. Besides the voltage of charge transfer interconnect, and the active set of banks, the proposed algorithm utilizes the EES bank reconfiguration as the additional control knob for better energy efficiency. Based on the proposed approximation of the DC-DC power consumption model, our heuristic algorithm has very low complexity compared to the previously proposed algorithms, which solve convex optimization problem iteratively and use binary search for optimal solution. The simplicity of our algorithm makes it suitable for runtime energy management for systems where the operating conditions vary rapidly.

- This thesis addresses low power scheduling of conditional task graphs on heterogeneous multi-core platform. The proposed framework optimizes task mapping and scheduling simultaneously. We consider the application with conditional execution as a random procedure in which the branch selection probabilities can be profiled (or measured). Our algorithm utilizes such statistical information in each

optimization step to reduce the mean power consumption. The proposed slack reclaiming based DVFS algorithm has pseudo-linear complexity with the respect to the number of tasks in the CTG. It also performs more robustly than the DVFS solution found by mathematical programming. More generalized hardware and software models are targeted in this work. The hardware platform consists of heterogeneous processing elements with different power-performance tradeoff characteristics. The only assumption that is required by our DVFS algorithm is a convex relation between the power and performance for each PE, which is usually true for many devices. The software application consists of a set of tasks with data and control dependencies and hard deadline constraint.

Chapter 2 Improving Charging Efficiency with Workload Scheduling and DVFS

Energy harvesting system has been applied in various area, from small wireless sensors to space exploration rovers. Wireless sensor nodes spatially distributed to monitor physical motion, temperature change, sound, pressure, and etc., are hard to be accessed again for battery replacement, so they are designed to support energy harvesting. Energy harvesting system has also been successfully applied on autonomous space exploration rovers for several years, e.g. NASA Sprit and Opportunity mars rover [2].

Recently, Internet giants like Google and Facebook announced plans on developing solar powered drones to deliver Internet access [3]. These robotic systems are able to carry relatively large size/capacity Electrical Energy Storage (EES) system to sustain their long operation cycles in extreme conditions and to store the extra energy harvested from environment; they are also integrated with embedded systems to do complex computation tasks, like data communication, image processing and recognition etc. An ultra-low-power power management IC is designed for energy-scavenged wireless sensor nodes [42].

Such an energy harvesting embedded system is a complex platform consisting of many components that work collaboratively. And the energy efficiency of the system can be improved in many aspects and at different levels of abstraction. From the perspective of embedded system

optimization, the scheduling and operating frequency/voltage of the embedded processor should be carefully selected to satisfy the performance requirement as well as power/energy constraint. As mentioned above, since the peak of solar power harvesting mismatches the peak of the power consumption, we want to store the surplus harvested energy into the EES element as much as possible. Intuitively, we believe that less energy consumed on the load device, more energy will be stored in the EES bank. However, with the overhead of the DC-DC converters, the relation between energy consumed and stored is not straightforward.

We use a simple example to show the detail. Assume that the input power from the energy harvesting module is 2W. Two different tasks will run on the energy dissipation module (EDM) sequentially with different power consumptions. The power consumption of high power workload is 1.5W and the power consumption of low power workload is 0.5W. Both tasks have duration 200 seconds. And the initial terminal voltage of the active EES bank is 0.5V. We simulate the charging process under two different workload schedules. If we scheduled the high power workload first, the system will charge the active EES bank using 0.5W charging power for the first 200 seconds and raise the terminal voltage of the EES bank to 0.8V. After finishing the high power workload, the EDM continues to execute the low power task. At this stage, the system charges the active EES bank using 1.5W charging power, and finally the terminal voltage is raised to 2.0V. On the other hand, if low power workload is scheduled at first, the system will charge the active EES bank using 1.5W, and it will raise the terminal voltage of the EES bank to 1.7V. After that when high power workload is running on the EDM, the system will charge the active EES bank using 0.5W charging power. However, at this point, input power is too small to charge the bank. The DC-DC converter connected with the EES bank for charging consumes almost all the charging power because the terminal voltage of the active EES bank is so high that

the converter works in the boost mode and has high power consumption overhead.

In this chapter, we investigate the impact of task scheduling and dynamic voltage and frequency scaling on the efficiency of an energy harvesting embedded system. More specifically, we consider the scenario where the harvested energy is more than enough to power the embedded system and the extra power will be used to charge an EES bank. Apparently, how much power is consumed by the embedded system directly affects the amount of remaining power that goes into the EES bank. For most EES modules (e.g. supercapacitors), their terminal voltage has a monotonically increasing relation with the amount of energy stored. From the example above, we know that different input power will increase energy storage at different rate and hence causes different terminal voltage. This affects the efficiency of the DC-DC converter, which matches the voltages between the EES and other system components. As a result, not all input energy can be stored in the EES and the efficiency of the EES charging process is affected. Our goal is to find a heuristic workload scheduling policy and Dynamic Voltage and Frequency Scaling (DVFS) strategy for the workload such that the most energy can be stored into the EES bank.

We consider the most basic scheduling problem that has only two tasks with equally short execution time and different power consumptions under the scenario of fixed energy harvesting rate. To avoid iteratively solving the complex optimization problem, we first derive a simplified yet accurate power consumption model of the DC-DC converter. Using the approximated DC-DC power model, we analytically prove that, in this special scenario, executing the high power task first enables more energy to be stored in EES. We then generalize this result to multiple tasks with arbitrary lengths under fixed and consistently decreasing energy harvesting scenario. We propose an accurate neural network based prediction model to estimate the state of charge

(SOC) of the EES bank given the input power to the EES bank, the initial SOC of the bank and the charging period. Based on the prediction model, we schedule the workload under all other conditions except when the harvesting rate is fixed or consistently decreasing. Combining both parts, we proposed a fast workload schedule to improve the efficiency of the EES bank effectively.

In this chapter, we also investigate the impact of DVFS of embedded task set on the efficiency of such energy harvesting embedded system. Our goal is to find an optimal DVFS assignment of the embedded workload such that the most energy can be stored into the EES bank for the future when harvesting rate is low.

The existence of DC-DC converters adds one more layer of complication to the problem for the following two reasons. First of all, the power consumption of DC-DC converters cannot be easily ignored and could be high if not configured properly. As a result, a DVFS scheme which minimizes the energy consumption of the embedded tasks does not necessarily minimize the overall energy consumption for the embedded system and DC-DC converters [43], which in turn does not result in the maximum energy stored in the EES bank. Second of all, as mentioned above, the power loss of the DC-DC converter is a complex non-linear function depending on the input/output terminal voltage and current. As a result, it is extremely difficult to use analytical approach to find out how much energy can be stored in the EES bank, which is a function of the integral of the DC-DC power loss, given the embedded task profile, the harvesting power profile and the initial condition. To circumvent these difficulties, previous works [14][11][41] propose approaches to optimize the instantaneous power efficiency of DC-DC converters. Their approaches assume that either input or output terminal voltage of the DC-DC converter is fixed and use binary search based method or convex optimization based method to find the optimal

voltage of another terminal. These approaches work very well if the terminal voltage is stable. However, if the terminal voltage varies dynamically, it is hard to find the optimal operation point to maximize the overall efficiency. And their sub-optimal approach is to divide the whole process into a number of small intervals and solve the instantaneous optimization problem periodically, which incurs a lot of computation overhead.

We tackle the first problem by considering the power consumption of DC-DC converters and the embedded system altogether. The power overhead of DC-DC converters is accurately accounted. Our DVFS scheme maximizes the energy stored in the EES bank. We tackle the second problem by using prediction model based on neural network. This prediction model would enable us to completely eliminate the time consuming process of periodically solving the binary search or convex optimization problem. This not only reduces the computation overhead but can also result in better solution. And base on this prediction model, we propose a dynamic programming based DVFS algorithm to optimize the energy stored in the EES bank.

The following summarizes the technical contributions of this work:

- Although there have been many publications on battery aware task scheduling, most of them consider the non-linear property of battery when it is in discharge mode [33]. To the best of our knowledge, this is the first work that optimizes the task scheduling and DVFS to increase the efficiency of EES charging process.
- We propose a neural network based energy prediction model which could accurately estimate the SOC of the EES bank given system variables. To our best knowledge, there is no existing work proposing the similar model to estimate the EES bank energy considering the DC-DC converters.

- We propose a fast and effective workload schedule based on the approximated DC-DC converter power consumption model and neural network prediction model to maximize the energy stored in the EES bank while guarantee the task set deadline.
- Base on the neural network prediction model, we propose a dynamic programming based voltage /frequency scaling algorithm which maximizes the harvesting energy stored in the EES bank. Unlike existing approaches which greedily optimize the instantaneous power input to the EES bank, the proposed approach considers the whole charging process and achieves better solution.
- Many of the existing works on EES efficiency optimization either ignore the impact of DC-DC converters ([41] and [44]) or minimize the energy waste on DC-DC using dynamic voltage selection with an assumption of fixed task scheduling [45]. This is the first work that investigates the impact of task scheduling and DVFS on the energy efficiency with the consideration of the DC-DC converters.

The rest of this chapter is organized as follows. The related work and system model are introduced in section 1. The DC-DC converter model and its approximation are presented in section 2. The proposed scheduling algorithm and neural network prediction model is in section 3. Experimental results and discussions for workload schedule are in section 4. The problem definition and motivational example for task DVFS are presented and the proposed dynamic programming based voltage/frequency assignment algorithm is described in section 5. Experimental results and discussions for task DVFS are presented in section 6. We conclude our work in section 7.

2.1 Related Work and System Model

2.1.1 Related Work

From the perspective of embedded system optimization, the scheduling and operating frequency/voltage of the embedded processor should be carefully selected to satisfy the performance requirement as well as power/energy constraint. For example, the authors in [30] proposed an energy-harvesting aware dynamic voltage frequency selection (EA-DVFS) algorithm, which adjusts the speed of task execution based on the current energy reserve in the system. In [31], the energy savings is further improved by exploring task slacks.

From the perspective of energy harvesting optimization, the operating conditions of the harvesting modules should be set carefully to generate more power. For example, in order to draw maximum amount of power from a photovoltaic (PV) array, various Maximum Power Point Tracking (MPPT) methods [36] have been proposed that dynamically adjust the output current to match the output impedance. In a recent work [44], the authors proposed a technique to improve the PV cells' efficiency under partial shading conditions.

The DC-DC converters are also important components in the system and their energy efficiency has drawn lots of attention in recent works. In [14], the authors proposed an analytic power model for DC-DC converters and based on this model, the authors found that the overall energy consumption of the embedded application and the converter is a convex function of the processor's supply voltage. Thus, the optimal operating voltage can be solved by minimizing a convex function. The DC-DC converter's power efficiency is also essential to the energy transfer between different components in the system. For example, [11][10][12] try to adjust the voltage of charge transfer interconnect (CTI) and active EES banks to reduce the power wasted on DC-

DC converters. And [40] proposed to reconfigure the parallel and serial connection of the EES bank to match the input and output voltage of the DC-DC converter to reduce the DC-DC converter power consumption.

2.1.2 Energy Harvesting Embedded System Model

Energy is harvested from the environment by Energy Harvesting Module (EHM) and distributed to other components. Any type of EHM can be integrated in our system. When the harvested power is more than sufficient for the embedded workload, the extra power will be stored in the EES bank for future use. In order to minimize the power wasted on the DC-DC converter, we assume only one EES bank is turned on at any time to accept the extra power. The efficiency of DC-DC converter increases when the difference between its input and output voltage reduces [40] by setting the voltage of the Charge Transfer Interconnect (V_{cti}) to be equal to the V_{dd} of the embedded processor, we can minimize the power wasted on ECM_{ED}. The very low V_{dd} level used by today's deep submicron technology will very likely keep V_{cti} lower than the terminal voltage of the EES bank, and consequently makes ECM_{ES} operate in the boost mode. Therefore, in this work, we confine our discussion to the scenario where ECM_{ES} operates in boost mode.

2.2 The DC-DC converter

Different components in the system cannot be connected to the charge transfer interconnect directly because these components usually have different operating voltage and current depending on their status. Therefore, the DC-DC power converters are placed between these components and the charge transfer interconnect for voltage and current regulation. A DC-

DC converter can provide desired output voltage or current level regardless the status of the storage banks or the loads.

2.2.1 The DC-DC Converter Model

In this work, we assume the converters in the system are all identical uni-directional switching buck-boost converters as shown in Figure 2.1 [40]. For each EES bank, there are two converters connected to it. One is for charging (charger, blue one) and the other is for discharging (discharger, green one). We assume at any given time, only one converter of these two is on, i.e. the bank is either charging or discharging. When the EES bank is charging, the I_{cti} and I_{bank} are positive while the EES is discharging, they are all negative.

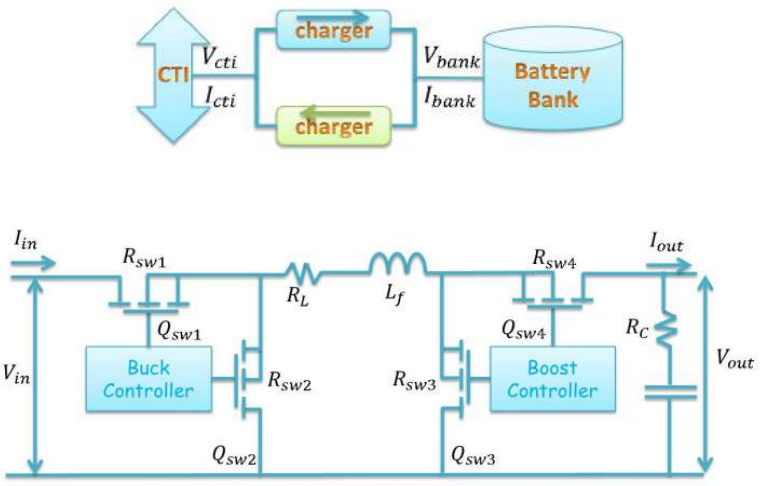


Figure 2.1 The DC-DC power converter structure

When the input voltage V_{in} is larger than the output voltage V_{out}, the DC-DC converter operates at buck mode. On the other hand, when $V_{out} < V_{in}$, the DC-DC converter operates at boost mode. The power consumption P_{dcdc} of the DC-DC converter strongly depends on V_{in}, V_{out} and the output current I_{out}. It consists of three components: conduction loss P_{cdct}, switching

loss P_{sw} and controller loss P_{ctrl}. If the DC-DC converter is operating at buck mode, the power components can be expressed as [11]:

$$P_{cdct} = I_{out}^2 \cdot (R_L + D \cdot R_{sw1} + (1 - D) \cdot R_{sw2} + R_{sw4}) + \frac{(\Delta I)^2}{12} \cdot (R_L + D \cdot R_{sw1} + (1 - D) \cdot R_{sw2} + R_{sw4} + R_c)$$

$$P_{sw} = V_{in} \cdot f_s \cdot (Q_{sw1} + Q_{sw2})$$

$$P_{ctrl} = V_{in} \cdot I_{ctrl}$$

Where $D = V_{out}/V_{in}$ is the duty ratio and $\Delta I = \frac{V_{out} \cdot (1-D)}{L_f \cdot f_s}$ is the maximum current ripple. f_s is the switching frequency; I_{ctrl} is the current flowing into the controller; R_L and R_C are the Equivalent Series Resistance (ESR) of inductor L and capacitor C, respectively; $R_{sw1\ldots4}$ and $Q_{sw1\ldots4}$ are the turn-on resistances and gate charges of the four switches in Figure 2.2, respectively. The power consumption on the DC-DC converter is the sum of P_{cdct}, P_{sw}, and P_{ctrl}.

If the DC-DC converter is operating at boost mode, the power components can be expressed as:

$$P_{cdct} = \frac{I_{out}^2}{D^2} \cdot (R_L + (1 - D) \cdot R_{sw3} + D \cdot R_{sw4} + R_{sw1} + D \cdot (1 - D) \cdot R_C) + \frac{(\Delta I)^2}{12} \cdot (R_L + (1 - D) \cdot R_{sw3} + D \cdot R_{sw4} + R_{sw1} + D \cdot R_C)$$

$$P_{sw} = V_{out} \cdot f_s \cdot (Q_{sw3} + Q_{sw4})$$

$$P_{ctrl} = V_{in} \cdot I_{ctrl}$$

Where $D = V_{in}/V_{out}$ and $\Delta I = \frac{V_{in} \cdot (1-D)}{L_f \cdot f_s}$.

2.2.2 Approximation of the Power Consumption of the DC-DC Converter

The above equation shows that the power consumption of the DC-DC converter P_{dcdc} is a function depends on the input voltage V_{in}, output voltage V_{out} and output current I_{out}. And the P_{dcdc} consists of three parts P_{cdct}, P_{sw} and P_{ctrl}. We further divided the P_{cdct} into two parts: P_{cdct1} and P_{cdct2}.

For buck mode:

$$P_{cdct1} = I_{out}^2 \cdot (R_L + D \cdot R_{sw1} + (1 - D) \cdot R_{sw2} + R_{sw4})$$

$$P_{cdct2} = \frac{(\Delta I)^2}{12} \cdot (R_L + D \cdot R_{sw1} + (1 - D) \cdot R_{sw2} + R_{sw4} + R_c)$$

For boost mode:

$$P_{cdct1} = \frac{I_{out}^2}{D^2} \cdot (R_L + (1 - D) \cdot R_{sw3} + D \cdot R_{sw4} + R_{sw1} + D \cdot (1 - D) \cdot R_C)$$

$$P_{cdct2} = \frac{(\Delta I)^2}{12} \cdot (R_L + (1 - D) \cdot R_{sw3} + D \cdot R_{sw4} + R_{sw1} + D \cdot R_C)$$

Because the resistance R_{sw1} and R_{sw2}, R_{sw3} and R_{sw4} have about the same value, we can simplify P_{cdct1} and P_{cdct2} as the following.

For buck mode:

$$P_{cdct1} = I_{out}^2 \cdot (R_L + R_{sw1} + R_{sw4})$$

$$P_{cdct2} = \frac{(\Delta I)^2}{12} \cdot (R_L + R_{sw1} + R_{sw4} + R_c)$$

For boost mode:

$$P_{cdct1} = \frac{I_{out}^2}{D^2} \cdot (R_L + R_{sw4} + R_{sw1} + D \cdot (1 - D) \cdot R_C)$$

$$P_{cdct2} = \frac{(\Delta I)^2}{12} \cdot (R_L + R_{sw4} + R_{sw1} + D \cdot R_C)$$

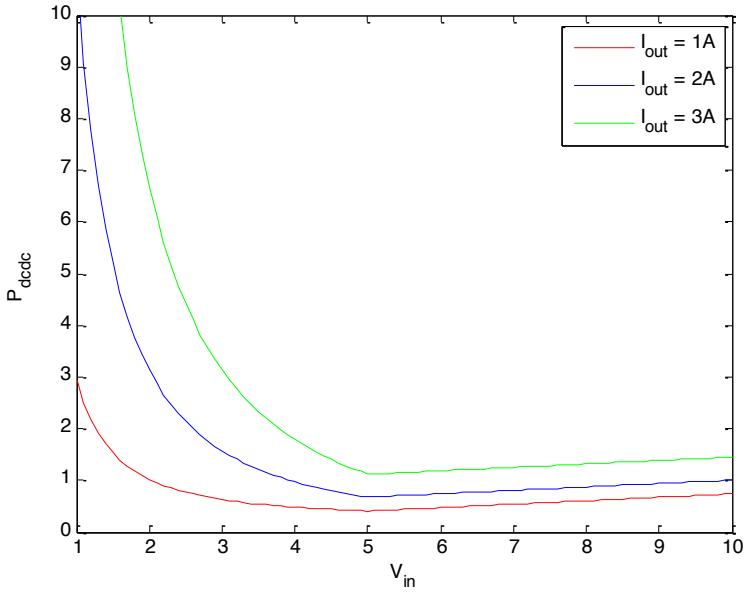


Figure 2.2 $\mathbf{P_{dcdc}}$ against $\mathbf{V_{in}}$ curve

As we can see, the power component P_{sw} and P_{ctrl} are linearly dependent on V_{in} and V_{out} in buck and boost mode respectively. P_{cdct1} is a constant in buck mode when I_{out} is fixed and a non-linear function of V_{in} and V_{out} in boost mode. P_{cdct2} is a non-linear functions of both V_{in} and V_{out}. To illustrate the relation between the DC-DC power consumption and the related variables, we fix the $V_{out} = 5V$ and plot the P_{dcdc} against V_{in} for I_{out} is 1A (red), 2A (blue) and 3A (green) in Figure 2.2.

As shown in the figure that when the DC-DC converter is operating at buck mode, i.e. $V_{in} > V_{out}$, the P_{dcdc} increases almost linearly against the V_{in}, which means the linear part is dominant in this region. Furthermore, we observe that when $V_{in} = V_{out}$, $D = \frac{V_{out}}{V_{in}} = 1, \Delta I = \frac{V_{out} \cdot (1-D)}{L_f \cdot f_s} = 0$, so $P_{cdct2} = 0$; when $V_{in} \to \infty$, $D = \frac{V_{out}}{V_{in}} = 0, \Delta I = \frac{V_{out} \cdot (1-D)}{L_f \cdot f_s} = \frac{V_{out}}{L_f \cdot f_s}$, so P_{cdct2} is a constant. In both cases, P_{cdct2} is is much smaller than P_{cdct1}. Therefore we could safely omit the non-linear part P_{cdct2}. Now P_{dcdc} only linearly depends on V_{in} and I_{out}^2 as following:

$$P_{dcdc} = P_{cdct1} + P_{sw} + P_{ctrl}$$

$$= I_{out}^2 \cdot (R_L + R_{sw1} + R_{sw4}) + V_{in} \cdot f_s \cdot (Q_{sw1} + Q_{sw2}) + V_{in} \cdot I_{ctrl} \quad (2.1)$$

On the other hand, when the converter is operating at boost mode, i.e. $V_{in} \leq V_{out}$, the power consumption increases non-linearly as the V_{in} decreases, which means that the non-linear parts (i.e. P_{cdct1} and P_{cdct2}) are dominant in this region. In boost mode, when $V_{in} = V_{out}$, P_{cdct2} is 0 while P_{cdct1} is non-zero. As $V_{in} \to 0$, $D \to 1$, so $P_{cdcd1} \to \infty$, and P_{dcdc2} is a finite number. This suggests that P_{dcdc2} can again be omitted compared to P_{cdcd1}. Therefore, we approximate the P_{dcdc} in boost mode as following:

$$P_{dcdc} = P_{cdct1} + P_{sw} + P_{ctrl}$$

$$= \left(\frac{I_{out} V_{out}}{V_{in}}\right)^2 \cdot (R_L + R_{sw4} + R_{sw1}) + V_{out} \cdot f_s \cdot (Q_{sw3} + Q_{sw4}) + V_{in} \cdot I_{ctrl} \quad (2.2)$$

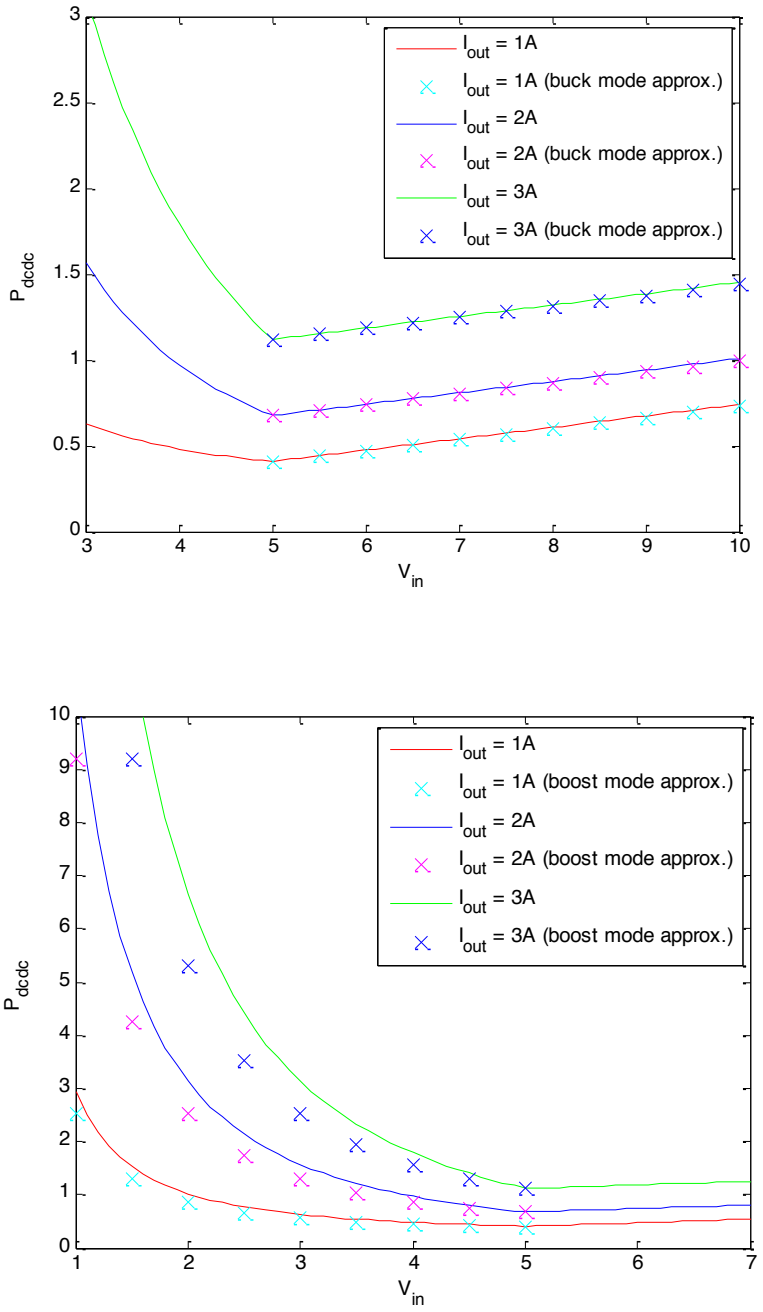


Figure 2.3 Fitting results of our approximation

Figure 2.3 compares the simplified P_{dcdc}(i.e. Equation (2.1)) and its original function given in Section 2.2.1. It shows that the approximation closely follows the original function.

2.3 Task Scheduling for Efficient EES Charging

2.3.1 Problem Definition For Workload Schedule

We define our scheduling problem as following. Given an energy harvesting embedded system, which is used to process n periodic tasks $T_1, T_2, \dots, T_n$. All tasks are immediately available at the beginning of a new period. Their durations are $t_1, t_2, \dots, t_n$. Tasks are independent, non-preemptive, and the processor works in single task mode. The power consumption of task T_i is denoted as P_i. At this time, we do not consider dynamic voltage and frequency scaling of the embedded processor; therefore, the power consumption of a task is assumed to be fixed during the runtime. We assume that the energy harvesting rate is higher than the power consumption of the EDM, therefore the EES bank works in charge mode. As we explained above, the DC-DC converter (ECM_{ES}) connecting the EES to the CTI operates in boost mode. Our goal is to find the optimal task schedule such that by the time when all tasks are completed the most energy can be stored into the EES bank.

We refer to the different charging characteristics as charging phases of the EES bank. When the harvesting rate is given, the identification of charging phases is solely determined by task execution. Given a set of tasks and an energy harvesting rate, executing the task with higher power consumption is equivalent to charging the EES bank using a lower input power, and vice versa. Therefore, the task scheduling can also be viewed as charge phase scheduling. In this work, we use the name highest power workload first (HPWF) to refer to the scheme that gives the top priority to execute the workload with the highest power consumption, and use the name lowest power workload first (LPWF) to refer to the scheme that gives the top priority to execute the workload with the lowest power consumption. We also use names highest power charge first

(HPCF) and lowest power charge first (LPCF) to refer to the scheduling scheme that generates a sequence of EES charging phases whose input power follows descending or ascending order respectively. It is obvious that when the harvesting rate is fixed, LPWF is equivalent to HPCF and HPWF is equivalent to LPCF.

Our task scheduling policy is explained in next sub-sections, the HPWF scheme is proved to be optimal at both the fixed and varying harvesting rate except one condition at the monotonically increasing harvesting rate. A neural network prediction model is introduced to fix this problem and applied in the our algorithm to find the task schedule such that by the time when all tasks are completed the most energy can be stored into the EES bank.

2.3.2 Task Scheduling at the Fixed Harvesting Rate

In this section, we first discuss the optimal charging phase scheduling when the harvesting rate is fixed and show that LPCF is superior to HPCF. This is proved first with the assumption that there are only two charging phases and both have the same durations that are extremely short. Then we extend this conclusion to scenarios with multiple charging phases that are equally short. Finally, we remove all constraint and generalize this conclusion to multiple charging phases with arbitrary duration. Based on this result, we further prove that HPWF is superior to LPWF.

2.3.2.1 Scheduling of Two Charging Phases with Very Short Duration

In this subsection, we consider two extremely short charging phases τ_1 and τ_2 with duration $\Delta t \to 0$, and their charging powers has the relation $P_{lo} < P_{hi}$. We denote the initial energy and voltage of the bank as (E_0, V_0). With the LPCF scheme, the bank is charged with P_{lo} in the first phase and then charged with P_{hi} in the second phase. The stored energy and voltage of

the bank at the end of phase 1 and phase 2 are denoted as (E_1^{LPCF}, V_1^{LPCF}) and (E_2^{LPCF}, V_2^{LPCF}) respectively. Similarly, with the HPCF scheme, the bank is charged with P_2 in the first phase and then is charged with P_1 in the second phase. The bank energy and voltage at the end of phase 1 and phase 2 are denoted as (E_1^{HPCF}, V_1^{HPCF}) and $\left(E_2^{HPCF}, V_2^{HPCF}\right)$ respectively.

Lemma 1. Given an EES with initial energy E_0 and initial voltage V_0. Charge this EES using two charging phases that has equal duration Δt that is extremely small. Denote the charging powers in the first and second phases as P_A and P_B respectively, and the bank voltage at the end of first and second phases as V_A and V_B respectively. At the end of second phase, the amount of energy stored in EES is:

$$E_1 = E_0 + \frac{\sqrt{\Delta(P_A, V_0)} - V_{in}^2}{2\alpha}\Delta t + \frac{\sqrt{\Delta(P_B, V_A)} - V_{in}^2}{2\alpha}\Delta t$$

where $\Delta(P, V) = V^4 + 4\alpha(P - \beta V - \gamma V)V^2$, and α, β, and γ are positive constant parameters for the given DC-DC converter.

Proof: Based on the DC-DC power approximation model, when in boost mode, P_{dcdc} can be written as

$$P_{dcdc} = \alpha\left(\frac{P_{out}}{V_{in}}\right)^2 + \beta V_{in} + \gamma V_{out} \tag{2.3}$$

Substitute P_{dcdc} with $P_{in} - P_{out}$ in the above equation and rearrange the equation, we obtain

$$\alpha P_{out}^2 + V_{in}^2 P_{out} + (\beta V_{in} + \gamma V_{out} - P_{in})V_{in}^2 = 0 \tag{2.4}$$

Although P_{out} can have two solutions, because $P_{out} > 0$, the only valid solution is

$$P_{out} = \frac{\left(\sqrt{V_{in}^4 + 4\alpha(P_{in} - \beta V_{in} - \gamma V_{out})V_{in}^2} - V_{in}^2\right)}{2\alpha} \tag{2.5}$$

Based on the definition of $\Delta(P, V)$, we have

$$P_{out} = \frac{\sqrt{\Delta(P_{in}, V_{out})} - V_{in}^2}{2\alpha}$$

The initial state of the bank is (E_0, V_0). After being charged using P_1 for Δt, the state of the bank becomes (E_A, V_A). Because Δt is very small, we assume the output voltage of the DC-DC converter is the same during this period. Also because P_{in} and V_{in} are constant values during the same charging phase, the output power is also a constant value and is denoted as $P_{out}(P_A, V_0)$. Then we have

$$E_A = E_0 + P_{out}(P_A, V_0) \cdot \Delta t = E_0 + \frac{\sqrt{\Delta(P_A, V_0)} - V_{in}^2}{2\alpha} \Delta t \tag{2.6}$$

For the same reason, in the second charging phase, we charge the bank using P_B for Δt and we have

$$E_B = E_A + P_{out}(P_B, V_A) \cdot \Delta t = E_A + \frac{\sqrt{\Delta(P_B, V_A)} - V_{in}^2}{2\alpha} \Delta t$$

$$= E_0 + \frac{\sqrt{\Delta(P_A, V_0)} - V_{in}^2}{2\alpha} \Delta t + \frac{\sqrt{\Delta(P_B, V_A)} - V_{in}^2}{2\alpha} \Delta t$$

Based on Lemma 1, we have the following theorem.

Theorem 1. Given that both charging phases have equal durations which are approaching to 0, the system follows LPCF scheme has more stored energy at the end of phase 2 than the system follows HPCF scheme, i.e. $E_2^{LPCF} > E_2^{HPCF}$.

Proof: For the LPCF scheme, we charge the bank using P_{lo} followed by P_{hi}. Both charging phases lasts for Δt. Based on Lemma 1 the final energy stored in the EES is:

$$E_2^{LPCF} = E_1 + P_{out}\left(P_{hi}, V_1^{LPCF}\right) \cdot \Delta t = E_1 + \frac{\sqrt{\Delta\left(P_{hi}, V_1^{LPCF}\right)} - V_{in}^2}{2\alpha}\Delta t$$

$$= E_0 + \frac{\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2}{2\alpha}\Delta t + \frac{\sqrt{\Delta\left(P_{hi}, V_1^{LPCF}\right)} - V_{in}^2}{2\alpha}\Delta t$$

,where V_1^{LPCF} is the bank voltage at the end of phase 1 for the system using LPCF scheme.

Similarity, with the HPCF scheme, the final energy in EES is:

$$E_2^{LPCF} = E_1 + P_{out}\left(P_{hi}, V_1^{LPCF}\right) \cdot \Delta t = E_1 + \frac{\sqrt{\Delta\left(P_{hi}, V_1^{LPCF}\right)} - V_{in}^2}{2\alpha}\Delta t$$

After eliminating the common terms we can see that, in order to prove $E_2^{LPCF} > E_2^{HPCF}$, we only need to prove the following inequality:

$$\sqrt{\Delta(P_{lo}, V_0)} + \sqrt{\Delta\left(P_{hi}, V_1^{LPCF}\right)} > \sqrt{\Delta(P_{hi}, V_0)} + \sqrt{\Delta\left(P_{lo}, V_1^{HPCF}\right)} \quad (2.7)$$

Before proving (2.7), let us first find the relation between V_0 and V_1^{LPCF} or V_1^{HPCF}. For simplicity, we use supercapacitor as the EES bank in our proof. (The same discussion can be applied to any EES bank as long as its terminal voltage is a monotonically increasing function of its energy.)

Let P denote the input power, V_0 and V_1 denote the initial bank voltage and the bank voltage after first charging phase. The energy of a supercapacitor can be represented as $E = \frac{1}{2}CV^2$. We used this to replace E in (2.6), and have

$$\frac{1}{2}CV_1^2 = \frac{1}{2}CV_0^2 + \frac{\sqrt{\Delta(P\ \ , V_0)} - V_{in}^2}{2\alpha}\Delta t$$

or equivalently

$$C(V_1 - V_0)(V_1 + V_0) = \frac{\sqrt{\Delta(P\ \ , V_0)} - V_{in}^2}{\alpha}\Delta t$$

Because $\Delta t \to 0$, we could assume that $V_1 \approx V_0$, and $V_1 + V_0 \approx 2V_0$. Therefore, we have

$$V_1 = V_0 + \frac{\sqrt{\Delta(P\ \ , V_0)} - V_{in}^2}{2\alpha C V_0}\Delta t$$

Therefore, we have:

$$V_1^{LPCF} = V_0 + \frac{\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2}{2\alpha C V_0}\Delta t$$

and

$$V_1^{HPCF} = V_0 + \frac{\sqrt{\Delta(P_{hi}, V_0)} - V_{in}^2}{2\alpha C V_0}\Delta t$$

We expand the V_1^{LPCF} and V_1^{HPCF} in $\Delta(P_{hi}, V_1^{LPCF})$ and $\Delta(P_{lo}, V_1^{HPCF})$:

$$\Delta(P_{hi}, V_1^{LPCF}) = V_{in}^4 + 4\alpha\left(P_{hi} - \beta V_{in} - \gamma(V_0 + \frac{\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2}{2C\alpha V_0}\Delta t)\right)V_{in}^2$$

$$= \Delta(P_{hi}, V_0) - \frac{2\gamma(\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2)V_{in}^2}{CV_0}\Delta t$$

$$\Delta(P_{lo}, V_1^{HPCF}) = V_{in}^4 + 4\alpha\left(P_{lo} - \beta V_{in} - \gamma(V_0 + \frac{\sqrt{\Delta(P_{hi}, V_0)} - V_{in}^2}{2C\alpha V_0}\Delta t)\right)V_{in}^2$$

$$= \Delta(P_{lo}, V_0) - \frac{2\gamma(\sqrt{\Delta(P_{hi}, V_0)} - V_{in}^2)V_{in}^2}{CV_0}\Delta t$$

We plug the above two equations, the left side becomes:

$$\sqrt{\Delta(P_{lo}, V_0)} + \sqrt{\Delta\left(P_{hi}, V_1^{LPCF}\right)} =$$

$$\sqrt{\Delta(P_{lo}, V_0)} + \sqrt{\Delta(P_{hi}, V_0) - \frac{2\gamma(\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2)V_{in}^2}{CV_0}\Delta t} \tag{2.8}$$

While the right side becomes:

$$\sqrt{\Delta(P_{hi}, V_0)} + \sqrt{\Delta\left(P_{lo}, V_1^{HPCF}\right)} =$$

$$\sqrt{\Delta(P_{hi}, V_0)} + \sqrt{\Delta(P_{lo}, V_0) - \frac{2\gamma(\sqrt{\Delta(P_{hi}, V_0)} - V_{in}^2)V_{in}^2}{CV_0}\Delta t} \tag{2.9}$$

Then we take square on both (2.8) and (2.9), and denote $\Delta(P, V_0)$ as $f(P)$, we have

$$f(P_{lo}) + f(P_{hi}) - \frac{2\gamma\left(\sqrt{f(P_{lo})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0} + 2\sqrt{f(P_{lo})f(P_{hi}) - \frac{2f(P_{lo})\gamma\left(\sqrt{f(P_{lo})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0}}$$

and

$$f(P_{lo}) + f(P_{hi}) - \frac{2\gamma\left(\sqrt{f(P_{hi})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0} + 2\sqrt{f(P_{lo})f(P_{hi}) - \frac{2f(P_{hi})\gamma\left(\sqrt{f(P_{hi})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0}}$$

Eliminate the common terms in the above two equations, to prove (2.7), it is sufficient to prove the two inequalities below

$$-\frac{2\gamma\left(\sqrt{f(P_{lo})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0} > -\frac{2\gamma\left(\sqrt{f(P_{hi})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0}$$

$$-\frac{2f(P_{lo})\gamma\left(\sqrt{f(P_{lo})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0} > -\frac{2f(P_{hi})\gamma\left(\sqrt{f(P_{hi})} - V_{in}^2\right)V_{in}^2\Delta t}{CV_0}$$

Here we need to state two properties of f(P).

- Because $\frac{df(P)}{dP} = 4\alpha > 0$, f(P) is an increasing function of P. So $f(P_{hi}) > f(P_{lo})$.
- Based on Equation (3), we have

$$\frac{\sqrt{f(P_{lo})} - V_{in}^2}{2\alpha} = \frac{\sqrt{\Delta(P_{lo}, V_0)} - V_{in}^2}{2\alpha} = P_{out} > 0$$

Therefore, $\sqrt{f(P_{hi})} > \sqrt{f(P_{lo})} > V_{in}^2$. And consequently

$$f(P_{lo})\left(\sqrt{f(P_{lo})} - V_{in}^2\right) < f(P_{hi})\left(\sqrt{f(P_{hi})} - V_{in}^2\right).$$

Given these two properties, it is not difficult to see that the two inequalities holds for $P_{hi} > P_{lo}$.

Combining the discussions above, we proved Theorem 1.

2.3.2.2 Scheduling of Multiple Charging Phases with Equally Short Durations

In this subsection, we consider the scheduling problem for arbitrary number of extremely short charging phases $\tau_1, \tau_2, \dots, \tau_n$ with duration $\Delta t \to 0$, and their charging powers are $P_1, \dots, P_n$. Before giving the theorem, we first give a lemma.

Lemma 2. Given two identical EES bank B_1 and B_2 with initial energy $E_1 < E_2$. After charging them using the same power P for the same duration T, the energy in B_1 is less than or equal to the energy in B_2, i.e. $E_1' \leq E_2'$.

Proof: Assume it takes time t to charge B_1 from E_1 to E_2 using P. Then after time $T - t$, bank B_2 reaches E_1', which equals the final energy of B_1 after time T. Continue to charge B_2 for time t, we have $E_2' \geq E_1'$.

Theorem 2. Given n charging phases $\tau_1, \tau_2, \dots, \tau_n$, with duration $\Delta t \to 0$, scheduling them based on the ascending order of their power maximizes the amount of energy stored in EES. Such scheduling policy is referred as LPCF (i.e. Lowest Power charging First).

Proof: We will prove this theorem using induction and contradiction. We know from Theorem 1 that the statement is true when $n = 2$. Assume LPCF scheme is the optimal scheduling policy for any i charging phases as long as $i \leq N$, we are going to prove LPCF scheme is the best for $N + 1$ charging phases.

Assume the energy optimal scheduling for the $N + 1$ charging phases are $S_{opt} = \{\tau_1, \tau_2, \dots, \tau_{N+1}\}$, and their power are not in the ascending order. This means that there are two tasks τ_i and τ_{i+1} such that $P_i > P_{i+1}$. Then we construct a new schedule by switching phases τ_i and τ_{i+1} and get $S' = \{\tau_1, \dots, \tau_{i+1}, \tau_i, \dots, \tau_{N+1}\}$. Note that after charging phase τ_{i-1}, the energy stored by both S_{opt} and S' are the same, because the first $i - 1$ charging phases are the same for the two schedules. For the next two charging phases, schedule $\{\tau_i, \tau_{i+1}\}$ is worse than schedule $\{\tau_{i+1}, \tau_i\}$ because $P_{i+1} < P_i$. Therefore, at the end of $(i + 1)$th charging phase, schedule S' stores more energy than the schedule S_{opt}. Because the remaining $N - i - 1$ charging phases are the same for both schedules, based on Lemma 2, S' stores more energy than S_{opt} at the end of all charging phases. This contradicts the assumption that S_{opt} is the energy optimal scheduling. Therefore, for $N + 1$ charging phases, the optimal scheduling is still the LPCF scheme.

From the above discussions, we have seen that for arbitrary number of very short charging phases, the LPCF scheme is the most energy efficient scheduling.

2.3.2.3 Scheduling of Arbitrary Charging Phases

In this subsection, we consider the scheduling problem for multiple charging phases with arbitrary duration. We claim that the LPCF scheme is still the best scheduling. This can be proved by dividing charging phases into very small slices such that $t_i = N_i \Delta t$, where t_i is the duration of the ith charging phase. We consider each slice as a sub-phase. All sub-phases belonging to the ith phase have the same charging power P_i.

Based on the discussion of previous sections, to reach the highest energy efficiency, these sub-phases should be arranged based on LPCF scheme. All sub-phases having the lowest charging power will be scheduled first, followed by the sub-phases having the second lowest charging power. This is equivalent as scheduling the charging phases from low power to high power, in another word, to execute tasks with highest power consumptions first.

2.3.3 Task Scheduling at Varying Harvesting Rate

It is easy to know that when the energy harvest rate is fixed, task scheduling and charging phase scheduling have direct correspondence. HPWF is the optimal task scheduling because it leads to LPCF, which has been proved to be optimal in previous sections. In this sub-section, we will show that HPWF is the optimal even if the energy harvesting rate is time varying except one condition at monotonically increasing harvesting rate which will be solved by neural network prediction model.

2.3.3.1 Scheduling with Monotonically Decreasing Harvesting Rate

We will show that the HPWF scheduling is still optimal when the harvesting rate is monotonically decreasing. Similar to Section 2.3.2, we first consider two very short tasks t_{hi} and t_{lo}, with duration $\tau \to 0$ and power consumption R_{hi} and R_{lo}. Without loss of generality, we assume $R_{hi} > R_{lo}$. The harvesting power during time period $[0,\ \tau]$ and $[\tau,\ 2\tau]$ is denoted as Q_1 and Q_2 respectively. We assume harvesting power is more than enough to power either one of the two tasks, i.e. $\min(Q_1, Q_2) > \max(R_{hi}, R_{lo})$. We focus our discussion to the case where the harvesting energy is decreasing, i.e. $Q_1 > Q_2$.

Based on the relations between $R_{hi},\ R_{lo}$ and Q_1, Q_2, we have two possible cases:

Case1: $Q_1 - R_{hi} > Q_2 - R_{lo}, Q_1 - R_{lo} > Q_2 - R_{hi}$.

Case 2: $Q_1 - R_{hi} < Q_2 - R_{lo}, Q_1 - R_{lo} > Q_2 - R_{hi}$.

We denote $P_1 = Q_1 - R_{hi},\ P_2 = Q_2 - R_{lo}$ and $P_1' = Q_1 - R_{lo,}\ P_2' = Q_2 - R_{hi}$. It is easy to see that if we execute task t_{hi} during the period $[0,\ \tau]$ and task t_{lo} during the period $[\tau, 2\tau]$, we are also charging the EES with input power P_1 during the period $[0,\ \tau]$ and input power P_2 during the period $[\tau, 2\tau]$. On the other hand, if we execute t_{lo} followed by task t_{hi}, we are charging the EES with P_1' for the duration $[0,\ \tau]$ and P_2' for the duration $[\tau, 2\tau]$. Furthermore, $P_1 + P_2 = P_1' + P_2' = P$.

We will first prove that, under case 1, HPWF is better than LPWF, i.e. charging EES with (P_1, P_2) is better than charging with (P_1', P_2').

Based on the lemma 1, using HPWF, the final energy stored in EES bank can be calculated as:

$$E_2 = E_0 + \frac{\sqrt{\Delta(P_1, V_0)} - V_{in}^2}{2\alpha}\Delta t + \frac{\sqrt{\Delta(P_2, V_1)} - V_{in}^2}{2\alpha}\Delta t$$

We are interested in finding the derivative of E_1' against P_1, i.e. dE_1'/dP_1. Based on the definition of $\Delta(P_1, V_0)$ and given that $P_2 = P - P_1$, and E_0, V_{in} do not depend on P_1, the derivative is:

$$\frac{dE_1'}{dP_1} = \frac{1}{\sqrt{\Delta(P_1, V_0)}}\Delta t - \frac{1 + \frac{\gamma}{CV_0\sqrt{\Delta(P_1, V_0)}}}{\sqrt{\Delta(P_2, V_1)}}\Delta t$$

Based on the definition of $\sqrt{\Delta(P_1, V_0)}$, we have $\sqrt{\Delta(P_1, V_0)} > \sqrt{\Delta(P_2, V_1)}$ because $P_1 > P_2$. Since $\gamma > 0$, we also have $1 + \frac{\gamma}{CV_0\sqrt{\Delta(P_1, V_0)}} > 1$. Together, we have $dE_1'/dP_1 < 0$, when $P_1 > P_2$. This means E_1' is a decreasing function against P_1, if $P_1 > P_2$. Because $P_1' > P_1$, charging with (P_1, P_2) is better than charging with (P_1', P_2'). So for case 1, HPWF is better than LPWF.

For case 2, we note that, first, charging with (P_1, P_2) is better than (P_2, P_1), because (P_1, P_2) is LPCF. Then based on the discussion for case 1, we know (P_2, P_1) is better than (P_1', P_2'), because $P_2 < P_1'$. Therefore HPWF is still better for case 2.

We have proved that HPWF scheduling is optimal for two very short tasks under monotonically decreasing harvesting power. This result can be extended to any number of tasks with arbitrary duration by using the same techniques in Section 2.3.2.

2.3.3.2 Scheduling with Monotonically Increasing Harvesting Rate

In this case, it is very difficult to analytically prove, for any given two tasks, which order is better. To see this, we again use the same notations as in the last section: $R_{hi} < R_{lo}$ and $Q_1 < Q_2$. Again, we have two possible cases:

Case1: $Q_1 - R_{hi} < Q_2 - R_{lo}, Q_1 - R_{lo} > Q_2 - R_{hi}$.

Case 2: $Q_1 - R_{hi} < Q_2 - R_{lo}, Q_1 - R_{lo} < Q_2 - R_{hi}$.

We again denote the $P_1 = Q_1 - R_{hi}$, $P_2 = Q_2 - R_{lo}$ and derive $\frac{dE_1'}{dP_1}$, which is exactly the same as in the last section. But now, since $P_1 < P_2$, the sign of $\frac{dE_1'}{dP_1}$ depends on the parameters, i.e. P_1, P_2, V_0, C and γ, and cannot be guaranteed. Therefore, it is necessary to utilize a prediction model to estimate which execution order is better.

2.3.3.3 Neural Network-Based EES Bank Energy Prediction Model

The power consumption of DC-DC converters P_{dcdc} (as well as the power goes into the bank) has a non-linear relation on the input/output voltage/current and it is very difficult to find an analytic expression of it with dependent parameters. The energy stored in the EES bank depends on the integral of P_{dcdc}, and is even harder to solve it analytically and efficiently. In this work, we overcome these difficulties by employing a neural network [47] to predict the energy stored in the EES bank based on system variables. Neural network is well known for its capability of accurately capturing the non-linear relation between its inputs and outputs. Although the training process is usually time consuming, however, it is an offline procedure and needs to be done only once. The recall process of the model has very low complexity. It only involves one light matrix-vector multiplication and one vector inner product for a neural network structure as shown in Figure 2.4. Hence, the runtime overhead is negligible. Furthermore, the

model treats the EES bank as a black-box, which enables it to be applied for any type of EES bank even without the knowledge of the detailed charging characteristics.

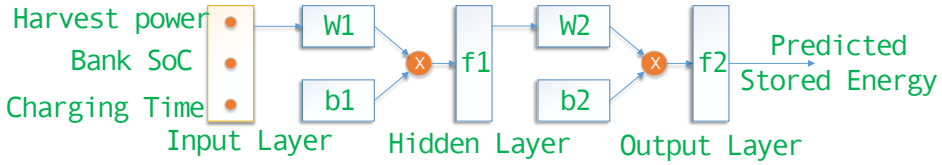


Figure 2.4 Neural Network Prediction Model

In this work, we adopt a two-layer neural network model as shown in Figure 2.4 to predict the energy stored in the EES bank. The input of the predictor is a set of variables that have direct impact on the energy stored in the EES. It includes: (1) The input power to DC-DC converters P_c. P_c determines the power goes out of the DC-DC converter and into the EES bank. It is the difference between the harvesting power and the load power. Including P_c enables the model implicitly takes the load into account. (2) The initial energy level (SOC) of the EES bank E_{init}. E_{init} affects the efficiency of DC-DC converters as well as the final energy of the EES bank. (3) The charging duration t_c. As shown in Figure 2.4, the model has 3 layers: one input layer, one hidden layer and one output layer. The inputs to the neural network model are s dimensional vectors ($s = 3$). There are m neurons in the hidden layer and one neuron in output layer. We set m to be 7 to get a good tradeoff between prediction accuracy and computation complexity. The $\mathbf{W_1}$, $\mathbf{W_2}$, b_1, b_2 are m by s hidden layer weight matrix, 1 by m output layer weight matrix, 1 by m bias vector and 1 by 1 bias constant respectively. They will be learned during training process and used as known parameters later the neural network model is applied to make prediction. The transfer functions for the hidden layer (f_1) and the output layer (f_2) are the tansig function and the purelin function respectively as shown in the figure. Let P_0 denote the

input vector, the output of hidden layer can be calculated as $P_1 = \text{tansig}(\mathbf{W_1} \times P_0 + b_1)$, and the output for the output layer is $P_2 = \text{purelin}(\mathbf{W_2} \times P_1 + b_2)$, which is the predicted energy stored in the bank. The neural network predictor is trained using the memory efficient Levenberg-Marquardt algorithm [47].

In general, the more training data we have, the more accurate prediction model we could obtain. However, more training data means longer training time. In order to limit the amount of training data yet still be able to get accurate prediction model, it is desirable to reduce the dimension of the input vector. Therefore, instead of feeding the charge time directly to the neural network model, we fix it to a constant (say 10 seconds) during the training process, which is, in fact, eliminating one dimension (the charging time). In the prediction process, we interpolate the charging time to get the predicted energy. Here is one example, we want to predict the final energy of a charging period of 23 seconds, we would call the prediction model 2 and 3 times to obtain the prediction results for 20 and 30 seconds. Then we use linear interpolation to get the results for 23 seconds.

More numerical details about generating the training data can be found in accuracy evaluation are as following. During prediction process, we feed the three inputs E_{init}, P_c and t_c to the trained model and obtain the final energy E_{end} shown in the following equation (2.10).

$$E_{\text{end}} = \text{nnet}(E_{\text{init}}, P_c, t_c) \tag{2.10}$$

We charge the supercapacitor bank with 11 different input powers ranging from 0W to 4W with a step of 0.4W. Each charging process starts when supercapacitor bank voltage is 1.0V and lasts for 4000 seconds. We record the bank energy every 5 seconds and feed these data into the neural network model. Even with these limited training data, the neural network is still able

to learn an effective model and make accurate prediction on input power it has never seen as shown later in this section. This feature is particularly useful considering the harvesting power can have big variations. Each training data is a 3 dimensional vector, with the input being input power to the bank, current bank energy and charging duration (10 sec), and output being the bank energy after 10 seconds. During the neural network training process, we use 80% data as training data and leave 10% data as validation data and 10% data as testing data. In addition to this set of testing data, we generate another set of validation data similar as how we generate the training data. The only difference is that the input power is randomly chosen in the range [0W, 4W].

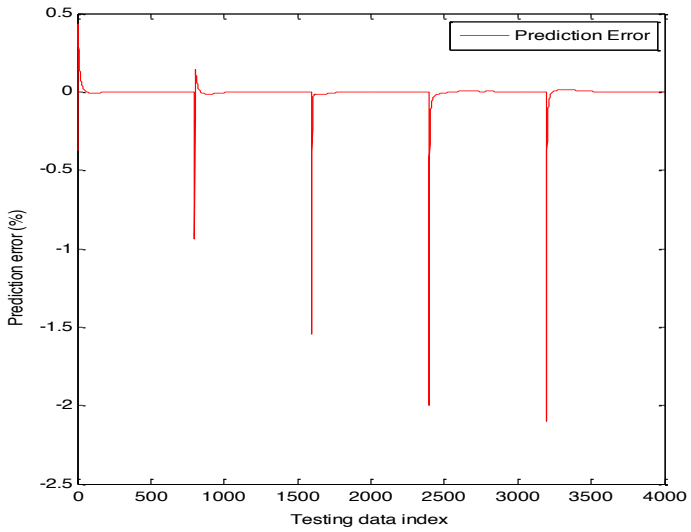


Figure 2.5 Prediction Error for the Validation Data Set

Figure 2.5 presents the prediction error for 4000 validation data, which are generated using 5 levels of input powers randomly selected in the range [0W, 4W], so each level of input power generates 800 testing data. Please note, these 4000 validation data is completely irrelevant with the testing data generated during the training process. The prediction error is defined as the

percentage difference between the predicted energy and the real energy stored in the EES bank. It clearly shows the accuracy of the prediction model. Most of the testing errors are less than 0.5%, and all of them are less than 2.5%. We notice that the prediction model produces bigger error when the bank energy is low, because at this time the energy change is more sensitive to the charging power, i.e. the slope of the energy increasing curve is sharp in this area (left part of Figure 2.5). When bank energy is high, more charging power will be wasted on DC-DC converters and the energy increasing rate reduces.

Figure 2.5 only shows the results of one step prediction, i.e. given the current bank energy and charging power, we predict the bank energy after charging for 10 seconds. This might work for a system with all tasks of about several tens of seconds. However, for the predictor to be truly useful, it must be able to accurately predict EES energy for variable charging durations. Figure 2.6 shows the prediction results when we use the model iteratively to estimate a charging period of 4000 seconds for three input powers. The results show that the prediction tracks the real energy accurately.

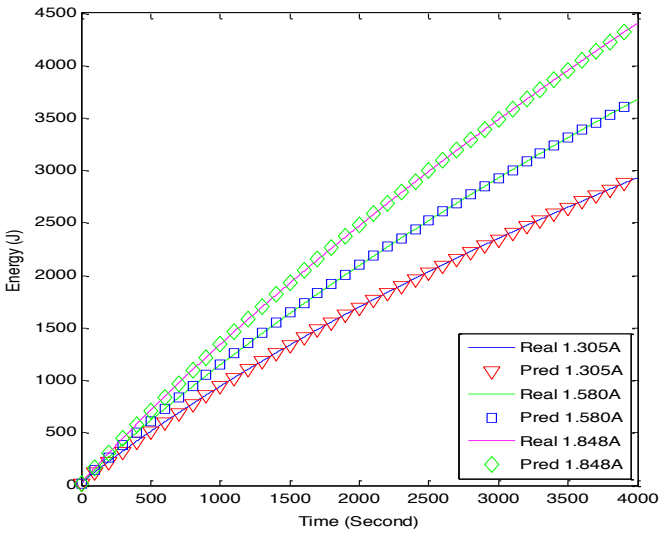


Figure 2.6 Multiple step prediction performance

2.3.4 Task Scheduling Algorithm to Improve Charging Efficiency

In this section, we present our heuristic task scheduling algorithm. The core idea is to consider low power charging first policy first, while leaving the unfitting case to the neural network predictor. We inspect each adjacent task pair, and their corresponding solar power profile. If the predicted solar power is monotonic non-increasing during the execution of the two tasks, then based on previous theorems, it ensures that the execution order which leads to the LPCF is optimal. If solar power is monotonically increasing, we apply the neural network to find out which of the two execution orders leads to more energy. The following pseudo code summarizes the task scheduling algorithm.

Algorithm 2.1: Task Scheduling Algorithm

```
    // Input: Two tasks (t1, t2), Predicted solar level S
    // Output: optimal execution order of (t1, t2)
1.  if S is monotonic non-increasing:
2.  return LPCF schedule
3.  if S is monotonic decreasing and only one order leads to LPCF:
4.  return LPCF schedule
5.  e1 = predictEnergy(execute t1 first)
6.  e2 = predictEnergy(execute t2 first)
7.  if e1 >= e2
8.  return execute t1 first
9.  else
10. return execute t2 first
```

Algorithm 2.1 Task Scheduling Algorithm

Algorithm 2.2: Task Scheduling Algorithm for N tasks

```
    // Input: N tasks (t1, t2, …, tN), Predicted solar level S
    // Output: optimal execution order of (t1, t2, …, tN)
11. while (true)
12.     swap = 0
13.     for (i = 0; i < N-1; i++)
14.         if (need to swap ti and ti+1)
15.             swap++;
16.      if (swap == 0)
```

17. break;

Algorithm 2.2 Task Scheduling Algorithm for N tasks

For a group of tasks, algorithm 2.2 shows that we examine each adjacent pair of tasks and swap them if necessary according to algorithm 2.1 until there is no adjacent pair of tasks needs to be swapped.

The above presented algorithm is an effective heuristic algorithm. First of all, in the case of stable harvesting rate or monotonous decreasing harvesting rate, it guarantees the optimality. Secondly, even with monotonous increasing harvesting rate or irregular harvesting rate, it can still use the accurate prediction to find a high quality schedule. In fact, without a prediction model, it is impossible to use bank energy as the objective in scheduling algorithm, because bank energy cannot be computed analytically.

2.4 Experimental Results for Workload Schedule

To demonstrate the effectiveness of the proposed algorithm, we implement a C++ simulator to model the energy harvesting embedded system. We assume the system has one customized supercapacitor [4] with 40F capacitance and 15V rated voltage as the EES element. This configuration is similar to the one used in [5]. We obtain the parameters of the DC-DC power converter model from [41]. These parameters are obtained from the datasheets of the real devices. We assume the V_{dd} of the embedded system is 1.0V and the V_{cti} is operated at 1.0V to match V_{dd}. The initial bank terminal voltage is also set to 1.0V.

2.4.1 Scheduling for Two Charging Phases

In the first set of experiments, we examine the impact of the scheduling for two charging phases with different charging power. Because there are only two charging phases, only two possible schedules are available: the LPCF and the HPCF scheme.

We first set the input power of one charging phase to be 0.5W and sweep the other from 0.2W to 1.0W. Note that the input power here is the extra harvested power after supplying the embedded system. We skip the case when both charging phases are 0.5W. We also set the duration of both charging phases to be 30 min. This duration is similar to the military radio application [10]. Table 2.1 Shows the energy stored in the EES element for the two scheduling schemes. As we can see, the LPCF scheme always performs better than the HPCF scheme as expected, and the difference could be up to 24.6%.

Power (W)	0.2	0.3	0.4	0.6	0.7	0.8	0.9	1.0
HPCF (J)	419.9	419.9	495.8	711.4	789.5	868.3	946.8	1069.2
LPCF (J)	442.9	496.9	562.5	773.3	910.5	1046.2	1179.9	1311.4
Impr. (%)	5.46%	18.34%	13.45%	8.70%	15.33%	20.49%	24.61%	22.65%

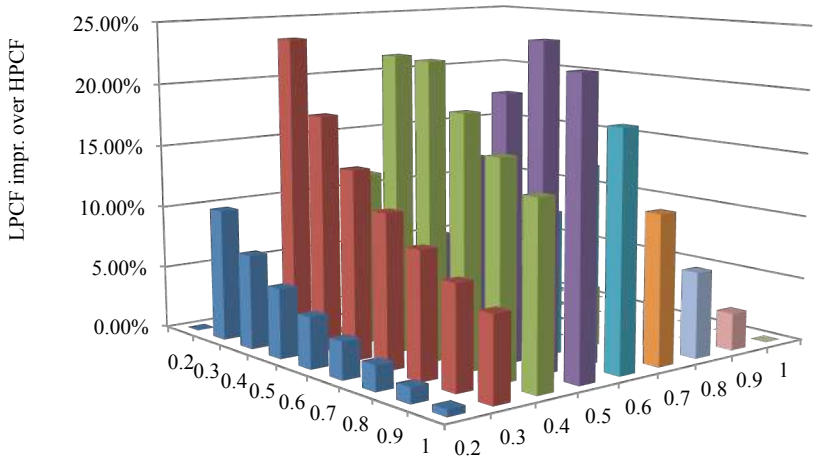
Table 2.1 Comparison of Energy Stored by HPCF & LPCF

As the input power increases in one of the charging phases and remains fixed in the other, the total energy stored in the EES element generally increases for both HPCF and LPCF. However, we also note that for the HPCF scheme, when the incoming power in one of the charging phase changes from 0.2W to 0.3W, the stored energy remains the same and all input power is consumed by the DC-DC converter. In fact, there is a minimum input power that is required to charge the EES bank. It can be estimated by setting $P_{in} = P_{dcdc}$ and $P_{out} = 0$ in our approximation model given by Equation (2.3). The minimum required input power is a function of V_{in} and V_{out}:

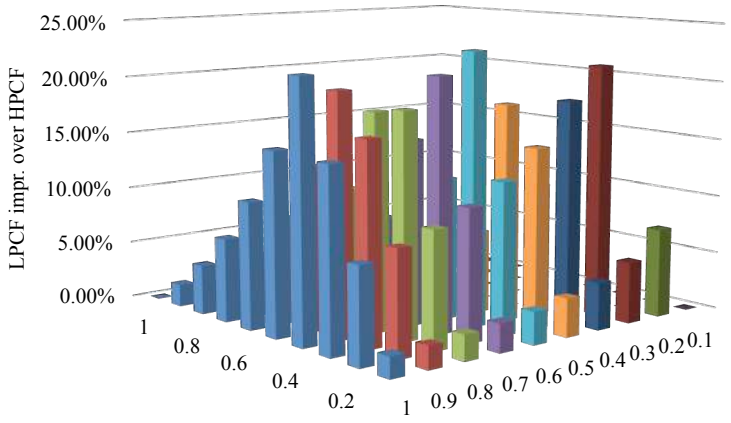
$$P_{in} = \beta V_{in} + \gamma V_{out}$$

which means we need more input power in order to break even of the consumption in the DC-DC converter when the terminal voltage of EES bank (i.e. V_{out}) increases. This can be viewed as an intuitive explanation of why LPCF is always better than the HPCF scheme. With the HPCF scheme, high power charging phase will first raise the terminal voltage of the EES bank to a high level, which reduces the opportunity for the bank to be charged in the future.

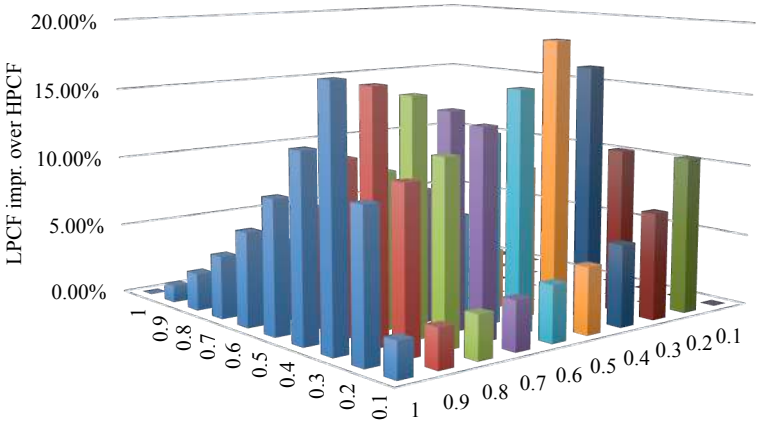
In the next experiment, we sweep the charging power from 0.2W to 1.0W with a step of 0.1W for both phases. The duration of each phase is still kept at 30 min. Figure 2.7a) shows the improvement of LPCF over HPCF for all possible combinations of charging phases. We could see that LPCF performs better than HPCF in all cases. And the average improvement is 10.41%. When the duration of the charge phase is set to 20min and 10min, the average improvement is 9.14% and 6.86% respectively.



a) Comparison of energy stored after 30min.



b) Comparison of energy stored after 20min.



c) Comparison of energy stored after 10min

Figure 2.7 Improvement of LPCF over HPCF when sweep power from 0.2w to 1w

2.4.2 Scheduling Results for Multiple Tasks

		1.2 W			1.4 W			1.6 W		
	Sec	100	300	600	100	300	600	100	300	600
8 tasks	HPCF (J)	384.3	835.1	1364.5	514.4	1174.0	1935.4	644.2	1545.0	2600.1
	LPCF (J)	405.7	939.6	1598.6	535.7	1294.2	2238.7	663.6	1652.1	2901.3
	avg (J)	393.3	879.6	1458.4	524.0	1227.7	2065.1	653.0	1593.3	2735.6
	Im. H	5.58%	12.5%	17.2%	4.15%	10.2%	15.7%	3.01%	6.94%	11.6%
	Im. A	3.15%	6.83%	9.61%	2.23%	5.42%	8.41%	1.62%	3.69%	6.06%

6 tasks	HPCF (J)	389.3	807.1	1262.8	488.5	1086.9	1783.2	585.4	1364.1	2307.4
	LPCF (J)	400.2	873.0	1450.9	498.4	1145.4	1951.3	594.6	1417.1	2459.4
	avg (J)	394.7	839.0	1354.4	493.4	1115.4	1865.1	589.9	1390.0	2381.8
	Im. H	2.81%	8.17%	14.9%	2.03%	5.39%	9.43%	1.56%	3.89%	6.59%
	Im. A	1.42%	4.05%	7.12%	1.03%	2.69%	4.62%	0.79%	1.95%	3.26%
10 tasks	HPCF (J)	493.8	1154.6	1921.9	656.7	1592.	2677.4	816.4	2049.8	3536.0
	LPCF (J)	519.8	1275.2	2208.9	679.9	1716.3	3012.5	837.6	2160.9	3840.4
	avg (J)	507.3	1215.3	2065.4	668.8	1657.0	2852.5	827.5	2107.9	3696.8
	Im. H	5.27%	10.5%	14.9%	3.53%	7.78%	12.5%	2.59%	5.42%	8.61%
	Im. A	2.47%	4.93%	6.94%	1.66%	3.58%	5.61%	1.22%	2.51%	3.88%

Table 2.2 Scheduling of multiple tasks with constant solar power

In the second set of experiments, we examined the impact of task scheduling for multiple tasks on the energy stored in the EES bank. Various system configurations are tested.

Table 2.2 shows the result of scheduling of multiple tasks with constant solar power. The number of tasks in each configurations are set to 6, 8 and 10; the task durations varies from 100s, 300s to 600s; and the harvesting power varies from 1.2W, 1.4W to 1.6W. The power consumption of each task is uniformly distributed between [0.2W, 1.0W]. We compare our LPCF scheme with HPCF scheme and the average results of 400 random generated schedules.

Table 2.2 reports the amount of stored energy for EES under LPCF, HPCF and the average random scheduling for all test cases. It also reports the relative improvement of LPCF over HPCF and random scheduling. We could see from this table that the LPCF scheme consistently outperforms the HPCF as well as the random scheduling policy in all test cases. And the improvement can be up to 17.15% over the HPCF scheme and 9.61% over the random scheduling. The HPCF always stores the least energy, which is even less than the worst cases in

the random schedules. We observe that in some settings where the phase duration is short, the terminal voltage of the EES bank stays relatively low all the time. These cases are in general less sensitive to the scheduling order, so the difference between the three scheduling schemes is small. This explains the trend that the results for cases with 600s phase duration are generally better than the cases with 100s phase duration. For the similar reason, when the harvesting power increases, the system becomes less sensitive to the charging phase order. On the other hand, some settings are more sensitive to the scheduling. For example when the harvesting power is 1.2W and duration is 600s. In these cases, inappropriately charging the EES bank with high power first will raise the terminal voltage of EES bank and prevent it to be further charged at low power phase.

Num_ of_ Tasks	Charging Power	[1.5, 1.8]	[1.6, 2.0]	[1.8, 2.2]
10	lpwf	700.50761	861.67737	932.63295
	random	724.44418	880.80102	952.58636
	sched	750.30561	903.81158	975.7946
	Impr.L	8.29%	5.53%	5.18%
	Impr.R	4.14%	2.95%	2.72%
15	lpwf	957.73372	1129.22774	1263.49074
	random	1004.50876	1169.45109	1305.75648
	sched	1047.28694	1211.83534	1343.98414
	Impr.L	10.44%	8.03%	6.92%
	Impr.R	4.73%	3.96%	3.17%
20	lpwf	1244.41919	1441.21392	1738.8753
	random	1313.55971	1510.44985	1800.48796
	sched	1384.86407	1578.11509	1866.45214
	Impr.L	12.27%	10.21%	7.78%
	Impr.R	5.88%	4.80%	3.88%

Table 2.3 Scheduling of multiple tasks with varying solar power

In this set of experiments, we examined the impact of task scheduling for multiple tasks on the energy stored in the EES bank when the harvesting rate is varied. The numbers of tasks in each configuration are set to 10, 15 and 20; the task duration is 100s; and the harvesting power varies within the range of [1.5W, 1.8W], [1.6W, 2.0W], [1.8W, 2.2W]. The power consumption of each task is uniformly distributed between [1.0W, 2.0W]. We compare our proposed task scheduling algorithm with LPWF scheme and the average results of 100 random generated schedules.

Table 2.3 reports the amount of stored energy for EES under LPWF, the average random scheduling and the proposed scheduling algorithm for all test cases. It also reports the relative improvement of the proposed scheduling algorithm over LPWF and random scheduling. We could see from this table that our proposed algorithm consistently outperforms the LPWF as well as the random scheduling policy in all test cases. And the improvement can be up to 12.27% over the LPWF scheme and 5.88% over the random scheduling.

2.5 DVFS Considering EES Charging Properties

Dynamic voltage and frequency scaling is an effective energy reduction technique and has been widely applied to embedded systems. Traditional DVFS aims at reducing workload power consumption without violating the performance constraint. In an energy harvesting system, during charging mode, reducing workload power is equivalent to increasing the charging power. Due to the variable power overhead of DC-DC converter, when to reduce the workload power consumption and how much should it be reduced will have different impact to the amount of energy that can be charged into the EES. In this section, we consider DVFS during the charging mode. We continue using the same energy harvesting embedded system model and DC-DC

converter model as discussed in previous sections. And we consider the scenario that the harvesting energy is sufficient to support the load device and charge the EES bank.

2.5.1 EES Bank Charging Characteristics and Motivational Example

As we discussed before, that assumption that all of the excessive harvested energy will be stored into the EES system is not valid because the power consumed by the DC-DC converter is non-negligible [43]. Therefore, in addition to the factors like the harvesting energy from the renewable energy source and the energy consumption of the workload applications, the amount of energy that can be stored in the EES banks is also affected by the power consumption of DC-DC converter.

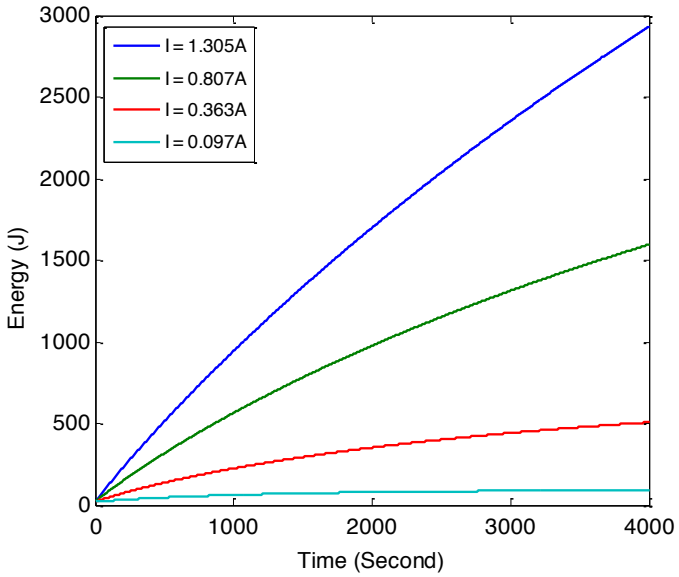


Figure 2.8 EES Bank Charging Characteristics

The power consumption of the DC-DC converter is not a constant, and it depends on the terminal voltages of its input and output. This suggests even charging with constant voltage and constant current, the terminal voltage as well as the stored energy of the supercapacitor bank

behind the DC-DC converter will not grow linearly. Figure 2.8 shows the super-capacitor bank energy growth as a function of time charged with different constant current based on DC-DC converter model in [41]. The concave curve clearly shows that the energy growth rate keeps decreasing because the bank terminal voltage increases with charging. This indicates that more power is consumed by the converter. One observation from the figure is that, for a given charging power, the bank will eventually reach a state such that charging efficiency is very low. Conversely, for a given bank terminal voltage, there is a minimum requirement on the charging power such that any power below this threshold is not able to charge the bank efficiently. Such charging characteristic motivates our power management strategy. It is our hypothesis that the optimal DVFS assignment, which maximizes the energy stored in the bank, might not necessarily minimize the energy consumed by the load tasks; under certain circumstance, it is necessary to speed up the load tasks and leave more idle slacks so that harvesting power could be fully used to charge the bank.

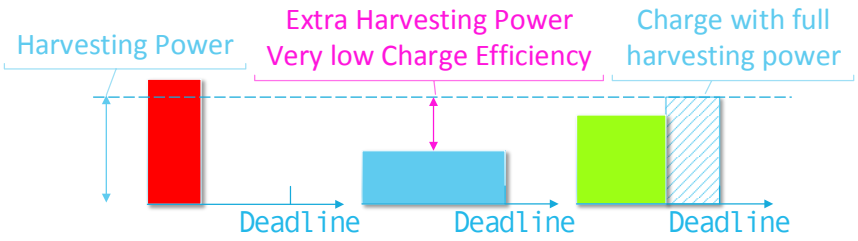


Figure 2.9 The Motivation Example

Figure 2.9 illustrates a motivational example when intelligently investing more energy to speed up load tasks is beneficial to conserve more harvesting energy in the EES bank. Three DVFS settings are presented in this figure. The left one uses the most aggressive voltage/frequency combination, consuming more power than that of harvested by the system, and is simply infeasible. The middle one minimizes the load task energy by running them at the

lowest voltage/frequency that just meets the scheduling deadline. Unfortunately, the surplus harvesting power is too low to charge the bank efficiently for current terminal voltage, and most of them are wasted. In this scenario, only the last DVFS setting could meet the scheduling deadline, while still being able to charge the supercapacitor bank with extra harvesting power in the idle slack. On the other hand, if the harvesting power is large enough to support the aggressive DVFS setting, then the left scheme leaves longer slack for harvesting power to be fully utilized to charge the bank and might be the best solution. Or if initial the bank terminal voltage is lower, then the extra power form the middle DVFS setting is probably able to charge the bank with high efficiency. And this setting might become the optimal solution as well. Obviously, the optimal DVFS strategy depends not only on the power consumption and deadline of the load tasks but also on the harvesting power and current SOC of the EES bank, which affects charging efficiency. The accurate estimation of energy stored into the bank plays a critical role in obtaining the best DVFS assignment. In the following section, we propose a dynamic programming based algorithm to find the optimal DVFS assignment.

2.5.2 Problem Definition

We define the DVFS assignment problem as the following. Consider an energy harvesting embedded system with K voltage/frequency settings $VF = \{(v_1, f_1), (v_2, f_2), \dots, (v_K, f_K)\}$ and a nominal setting $(v_{nom}, f_{nom}) \in VF$, which is used to process N tasks $\Gamma = \{\tau_1, \tau_2, \dots, \tau_N\}$ with a common deadline D. In this work, we assume that task execution order is fixed and only consider DVFS. For a task τ_n, it's execution length and power consumption under V/F setting k are $VL(n, k)$ and $VP(n, k)$. The next task τ_{n+1} is available when previous task τ_n finishes. Yet τ_{n+1} does not have to start immediately when τ_n finishes. Our task model is similar to that of [31]. We assume that the energy harvesting rate is higher than the

power consumption of the load, therefore the EES bank works in charge mode. Our goal is to find the optimal DVFS setting as well as the start time for each task such that by the deadline D the most energy can be stored into the EES bank.

2.5.3 A Dynamic Programming- Based DVFS Algorithm

Our goal is to find the optimal voltage/frequency assignment for each task such that all the tasks are finished within the given deadline and the extra energy stored in the EES bank is maximized. The proposed voltage/frequency assignment algorithm is based on dynamic programming (DP) and it has polynomial complexity. The algorithm runs every time when a new task set arrives. The symbols are summarized in Table 2.4.

Symbol	Definition
$VF(n,t)$	Voltage of task τ_n
$E(n,t)$	Energy stored in the EES bank when task τ_n finishes
$S(n,t)$	Start time of task τ_n
v_{nom}	Nominal voltage
P_h	Harvesting power
$VP(i,k)$	Power of real task τ_i running at kth voltage level
$VL(i,k)$	Execution length of real task τ_i running at kth voltage level
N, T, K	Total number of tasks, timestamps and voltage/frequency levels
n, t, k	Index of tasks, timestamps and voltage/frequency levels

Table 2.4 The List of Symbols

We divide the deadline D of the task set into T timestamps: $\{0, 1, \dots, T\}$. The task set is $\Gamma = \{\tau_1, \tau_2, \dots, \tau_N\}$. And then we build a two dimension table schedTable of size $(N)\times(T+1)$. Each entry (n,t) in this table contains 3 variables: $VF(n,t)$, $E(n,t)$ and $S(n,t)$. For a task τ_n, $VF(n,t)$ equals the operating voltage of task if first n tasks can be finished exactly at time t, otherwise we set $VF(n,t)$ to NULL. If $VF(n,t)$ is NULL, then other two variables in the same table entry will be assigned NULL as well. And we mark such a table entry as invalid. On the

other hand, if $VF(n,t)$ is not NULL, then $E(n,t)$ is the energy stored in the EES bank when task τ_n finishes and $S(n,t)$ is the start time of task τ_n, which also equals to the finish time of task τ_{n-1}. It is easy to see that if table entry (n,t) is valid, then table entry $(n-1, S(n,t))$ is also a valid entry. And we utilize this property as the recurrence relation to fill in the schedTable table as shown in algorithm 2.3.

Algorithm 2.3: Constructing the DP table schedTable

Input: task set Γ, deadline D, with two tables $VL(n,k)$, $VP(n,k)$.
Output: Fulfilled DP table

```
(1)   /* Initilize DP Table: entry (i, j) = NULL; */
(2)   /* Fill the first row for idle task τ0 */
(3)   for (t = 0; t <= T; ++t) {
(4)     S(0, t)=0; VF(0, t)=vnom;
(5)     E(1, t)=nnet(Einit, Ph, t);
(6)   }
(7)   /* Fill the rest of the table for task τ1 to τ2N */
(8)   for (i = 1; i <= N; ++i) {
(9)     for (j = 0; j <= T; ++j) { // real tasks
(10)      if (entry(i-1,j) != NULL) {
(11)        for (k = 0; k < K; ++k) {
(12)          t = j + VL(i,k);
(13)          Epred = nnet(E(i-1,j),Ph-VP(i,k),VL(i,k));
(14)          if ( (entry(i,t)==NULL) || (Epred > E(i,t)) )
(15)            S(i,t)=j; VF(i,t)=k; E(i,t)=Epred;
(16)          }  // if line 13
(17)        }  // for k line 10
(18)      }  // if line 9
(19)    } // for j line 8
(20)  } // for i line 7
```

Algorithm 2.3 Constructing the DP Table schedTable

The input to Algorithm 2.3 is the task set $\Gamma = \{\tau_0, \tau_1, \tau_2, \ldots, \tau_N\}$, the deadline T, along with the two N by K tables VP and VL. Each entry of the former table, $VP(n,k)$, is the power consumption of task τ_n under voltage k, and each entry of the latter one, $VL(n,k)$ is the length of task τ_n under voltage k. Initially, all entries in the schedTable are reset to NULL. Algorithm 2.3 starts by filling the first row of schedTable, which is for the first task τ_1. Only K entries in this

table will be valid, because the execution length of τ_1 has only K possibilities. It sets the variables in the table entry $(1, \mathrm{VL(k)})$ to be $\mathrm{VF}(1, \mathrm{VL(k)}) = \mathrm{k}$, $\mathrm{S}(1, \mathrm{VL(k)}) = 0$, and $\mathrm{E}(1, \mathrm{VL(k)}) = \mathrm{nnet}\,(\mathrm{E_{init}}, \mathrm{VP(k)}, \mathrm{VL(k)})$, that is the predicted energy stored in the EES bank when idle task τ_1 finishes at time $\mathrm{VL(k)}$. Then it continues to fill the table entry for task τ_2 to τ_N. For a task τ_i, it looks for valid table entry in row $\mathrm{i}-1$. If $(\mathrm{i}-1, \mathrm{t})$ is a valid table entry, then the algorithm updates the table entry $(\mathrm{i}, \mathrm{t} + \mathrm{VL}\,(\mathrm{i}, \mathrm{k}))$ for $\forall \mathrm{k} = 1, \ldots, \mathrm{K}$, only when this entry is not valid or previous predicted energy is less than current predicted energy (line 8 ~ 18). The complexity of the Algorithm 2.3 is O (TNK) and the memory requirement for the table is proportional to the table size, which is $\mathrm{N}\,(\mathrm{T}+1)$. And each table entry only stores 3 floating numbers, which is 24 Byte.

After the DP table schedTable is fulfilled, we use Algorithm 2.4 to back trace the optimal scheduling. It starts on the last row N and finds a valid table entry (N, t) with the maximum energy $\mathrm{E}\,(\mathrm{N}, \mathrm{t})$. $\mathrm{VF}(\mathrm{N}, \mathrm{t})$ is the voltage/frequency setting for task τ_N in the optimal schedule. Algorithm 2.4 pushes the table entry (N, t) into a stack and goes to DP table entry $(\mathrm{N}-1, \mathrm{S}\,(\mathrm{N}, \mathrm{t}))$, which must be a valid entry for task $\tau_{2\mathrm{N}-1}$ based on the recurrence relation mentioned in Algorithm 2.4. And $\mathrm{VF}\,(\mathrm{N}-1, \mathrm{S}\,(\mathrm{N}, \mathrm{t}))$ is the voltage/frequency setting for task $\tau_{\mathrm{N}-1}$ in the optimal schedule. The algorithm continues until all rows have been visited. The complexity for Algorithm 2.4 is $\mathrm{O}\,(\mathrm{T}+\mathrm{N})$.

Algorithm 2.4: Back tracing to find the energy optimal schedule

Input: Fulfilled DP table from Algorithm 1
Output: Energy optimal schedule

```
1.  /* Find table entry with max energy in row N */
2.  entry(N,t) = {(N, t) | E(N, t) is maximum};
3.  stack.push_back(entry(N, t));
4.  for (i = N; i >= 1; --i) {
5.    t = S(i,t);
```

```
6.    stack.push_back(entry(i-1, t));
7.  }
8.  return stack;
```

Algorithm 2.4 Back tracing to find the energy optimal schedule

2.6 Experimental Results For DVFS Algorithm

In order to demonstrate the effectiveness of the proposed dynamic programming based DVFS algorithm, we implement a C++ simulator to model the energy harvesting embedded system. We assume the system has one customized supercapacitor bank [4] with 40F capacitance and 15V max voltage as the EES element. This configuration is similar to [40][5], and can be applied to relatively large system like rovers and drones. Yet as discussed in Section 3.3.4.2, the prediction model is not limited to the supercapacitor bank and can be applied to any type of EES bank. We obtain the parameters of the DC-DC power converter model from [41]. These parameters are obtained from the datasheets of the real devices.

We assume that the V_{dd} of the embedded system supports 3 voltage levels from 0.8V to 1.0V with a step of 0.1V. We also assume that the power consumption of the embedded system is proportional to the cubic of the operating voltage and the operating frequency is linearly proportional to the voltage. Please note that our algorithm does not depend on this assumption to work. It only needs to know the power consumption of the embedded system for each task at different operating conditions (VF settings), and will pick the optimum VF setting for each task.

The V_{cti} is assumed to operate at 1.0V, which means that the input voltage to DC-DC converters is 1.0V. Please note that we do not consider V_{cti} optimization techniques [11][48] in this work. However, those techniques are also orthogonal to our DVFS algorithm and can be combined together.

Finally, we assume that the harvesting power to the system is 4W, because there are a lot of portable solar panels available in the market which could output around this range. The variation of harvesting power will not affect our DVFS algorithm, because it only limits the voltage/frequency settings feasible to a task and the algorithm will always only choose valid voltage/frequency setting for a task. Therefore, we assume that harvesting rate is constant for simplicity. Since we only consider EES charging problem in this work, we assume the task power is always less than the load power. Otherwise it will become an EES discharging problem.

2.6.1 Evaluation of Proposed DVFS Algorithm

In this section, we evaluated our dynamic programming based DVFS algorithm. We compare the amount of energy stored in the EES banks when all tasks finish. The supercapacitor bank is assumed to be pre-charged to 2.5V. The processor of the load device can select the execute voltage among 0.8V, 0.9V and 1.0V.

In the first experiment, there are 6 tasks in this task set. The load task power is generated randomly in a range of 1.0W to 2.0W. When the voltage is 1.0V, the execution time of each task is 100s. And the charging power is fixed at 2.0W.The basic idea of the baseline algorithm 1 is first to find the optimal frequency that minimizes the energy consumption of the load tasks assuming that the frequency can be adjusted continuously. If the optimal frequency is not supported by the system, it chooses between two adjacent frequency levels to approximate the optimal frequency while making the task set meet the deadline. The idea of the baseline algorithm 2 is to choose the minimum frequency which makes the task set meet the deadline at the beginning of each task.

WCET/DEADLINE	VS Baseline1	VS Baseline2
~0.88	9.99%	9.99%
~0.86	17.36%	11.13%
~0.85	23.60%	10.17%
~0.84	22.48%	3.93%
~0.83	27.14%	8.67%
~0.82	31.89%	6.30%
~0.81	34.13%	2.94%

Table 2.5 Comparison of different slow down ratios

Table 2.5 shows the results of the first experiment. We set the deadline so that the baseline algorithm 1 needs to use at least 0.9V to execute the task set. For example, when we set the deadline 680, the lowest voltage for the task set is 0.89V. Hence, the baseline algorithm needs to select 0.9 V to guarantee the task set can be finished on time. In the experiment, we extend the deadline before the baseline algorithm 1 could select 0.8 to execute the task set and still guarantee the deadline is met. The experiment shows that our algorithm can outperform the baseline algorithm 1 up to 34.13% in this case. And when the lowest voltage is further from 0.9V, our algorithm works better than the baseline algorithm1. That's because the baseline algorithm 1 just consider to minimize the power dissipation on the load devices, but our algorithm consider to maximize the energy stored in the EES bank directly while reducing the power consumption on the load device. The results based on baseline algorithm 2 are better than baseline algorithm 1, because it begins to slow down part of the tasks when the slow down ratio is set to 0.86. It means that part of tasks use 0.9V and the others use less than 0.9V. However, our DVFS algorithm still outperforms the baseline algorithm 2 in all cases, and the improvement can be as large as 11%. Baseline algorithm 2 adjusts the task execution speed only based on the task order and available slack, without considering the critical power information. Therefore, it cannot find the optimal

speed for each task. Whereas our algorithm take account of the power consumption of each task in the task set, which leads to more energy saved in EES bank.

In the second experiment, we randomly generate task sets with 6, 8, and 10 tasks with task power in a range of 1.0W to 2.0W, and the execution length of each task is still 100 seconds when the voltage is 1.0V. Also, we first assume that the baseline algorithm needs use 0.9V to execute the task set. In two cases, we set the two deadlines so that the lowest voltage for one is around 0.85V, and the lowest voltage for the other is around 0.81V. We compare our algorithm with the baseline algorithms and the "no-DVFS" policy which runs the tasks without any DVFS. Apparently, "no-DVFS" policy will finish the task set first. Then the energy harvested in the slack between the finish time and deadline will be fully used to charge the EES banks. This table shows our algorithm stored up to 44.34% more energy compare to "no-DVFS".

Num Tasks	Algorithms	W/D ≈ 0.85	W/D ≈ 0.81
6	Impr.no-DVFS	39.50%	44.34%
	Impr.BL1	9.86%	16.08%
	Impr.BL2	3.63%	1.55%
8	Impr.no-DVFS	37.63%	40.90%
	Impr.BL1	10.60%	15.21%
	Impr.BL2	4.16%	1.78%
10	Impr.no-DVFS	41.21%	43.99%
	Impr.BL1	11.33%	10.66%
	Impr.BL2	3.62%	1.51%

Table 2.6 Comparison of different task sets

Table 2.7 shows the percentage improvement of energy stored in the supercapacitor bank of our DVFS algorithm over the baseline algorithms when the ratio between the load task power and harvested power changes. The execution length of each task is 100 seconds when the voltage is 1.0V. This table clearly shows the trend that when average ratio between task power and harvested energy is increasing, our algorithm can store more energy compare to the baseline

algorithms. When task power ranges from 60% ~ 70% of harvesting power (first column), the improvement is minimum; and when it ranges from 70% ~ 80% (last column), the improvements are larger. That's because the voltage/frequency of each task needs to be assigned more and more carefully when the extra power decreases. Otherwise, the extra harvesting power saved by running the load tasks at lower voltage is more likely to be wasted on the DC-DC converters, like the middle example shown in Figure.

Num Tasks	Charging Power	[2.8, 3.2]	[2.6, 3.0]	[2.5, 2.8]
6	Impr.no-DVFS	22.42%	25.78%	29.39%
	Impr.BL1	5.71%	6.44%	7.19%
	Impr.BL2	1.93%	2.22%	2.54%
8	Impr.no-DVFS	22.20%	26.25%	29.17%
	Impr.BL1	5.93%	6.83%	7.50%
	Impr.BL2	2.13%	2.51%	2.77%
10	Impr.no-DVFS	24.31%	27.86%	31.02%
	Impr.BL1	6.67%	7.50%	8.12%
	Impr.BL2	2.00%	2.25%	2.38%

Table 2.7 Comparison of different energy harvesting ratios

2.7 Chapter Summary

In this chapter, we investigate the effects of workload scheduling on the efficiency of the EES charge process in an energy harvesting embedded system. We found that low power first scheme always performs better than the high power first scheme when the harvesting rate is fixed or monotonically decreases. It is proved using an approximated but accurate power model of the DC-DC converter. We proposed a neural network based prediction model which could accurately estimate bank energy given input power, initial bank energy and charging time. Under other situations, we use the energy prediction model to help task scheduling. We integrate both parts to build a fast and effective task scheduling algorithm such that the energy stored in the

EES bank is minimized and the task set deadline is guaranteed. Experimental results show that the HPWF outperforms the LPWF by up to 24.61% for two charging phases. For multiple tasks, HPWF outperforms LPWF by up to 12.27% and outperforms the random scheduling scheme by up to 5.88%. We also investigate the effects of DVFS on the efficiency of supercapacitor bank charging process in an energy harvesting embedded system. Based on the same energy prediction model, we propose a dynamic programming based DVFS algorithm which performs robustly regardless the variation of bank SOC, task load power consumption and schedule deadline compare to two baseline algorithms. Results show that up to 34% more harvesting energy could be stored into the EES bank.

Chapter 3 Improving Energy Efficiency for Energy Harvesting Embedded Systems

As mentioned above, environmental energy harvesting is a promising technique for sustainable operation of embedded system (e.g. wireless sensor node). It can potentially provide the possibility of unlimited lifetime [34]. However, one of the major constraints of applying energy harvesting technique to real-time embedded system is the uncertainty and large variation in harvesting rate [31]. For a practical implementation of such system, Electrical Energy Storage (EES) element is usually employed to compensate the power fluctuation caused by the energy harvesting. With such an EES element integrated in the system, excessive energy can be stored while the harvesting rate is high. When the harvesting rate is low, the EES element can be a supplement energy source for the embedded system to work properly.

Traditional EES systems are mainly homogenous and have the same type of storage banks. Recently, the Hybrid Electrical Energy Storage (HEES) system [9] has been proposed to overcome the drawbacks of different types of EES banks while exposing their strengths. These EES banks have different characteristics such as cycle efficiency, leakage current, cycle life, storage cost and volumetric energy density, power rating and so on. These EES elements are connected via Charge Transfer Interconnect (CTI) and DC-DC power converters to enable power transfer. Due to the combination of various EES elements, HEES system may show good

performance metric including high energy density, high power delivery capacity, low cost, and low leakage.

Several research works have addressed different problems on charge transfer including charge replacement, charge allocation and charge migration. One drawback of previous works is the high complexity of the algorithm, which might not be feasible for online applications, because the status of the EES banks, the power input to the system and the load characteristics could change rapidly. When these system status change, the control parameters have to be recomputed, which could incur large runtime overhead.

In this chapter, we propose a fast heuristic algorithm to improve the energy efficiency for charge allocation and replacement in an EHS/HEES equipped embedded system. We observed that the major energy loss in the system during charge allocation and replacement is on the DC-DC power converters, which are important components in the system. So the goal of our algorithm is to minimize the energy overhead on the DC-DC converter while maximizing the energy stored in the HEES system and satisfying the task deadline constraints of the embedded workload. We divide the energy efficiency optimization problem into two parts. When the harvesting power is high enough, the problem becomes charge allocation problem. In addition to supplying the embedded workload, excessive energy will be stored in the HEES system. On the other hand, when the harvesting power is low, the problem becomes charge replacement problem, because the energy is drawn from the HEES to supply the embedded load.

We observed that when the input and output voltage of the DC-DC converter matches, the power consumption of the converter is minimized. Therefore, our algorithm tries to match the terminal voltages of the DC-DC converters in the system for both charge allocation and

replacement. Compare to the previous works [11][10][12][35], in addition to voltage of charge transfer interconnect (V_{cti}) and the active set of banks, our algorithm has one more control knob, i.e. the structure of EES bank. We utilize the EES bank reconfiguration technique to dynamically adjust the connection of the EES bank so that the bank terminal voltage could better match the V_{cti} .

The following summarizes the key characteristics of this chapter:

- Unlike the previous works [11][10][12][35], which only consider one sub-problem of the HEES system, our work considers both charge allocation and charge replacement together. In addition to V_{cti} and the active set of banks, the proposed algorithm utilizes the EES bank reconfiguration as additional control knob for better energy efficiency.

- Based on the proposed approximation of the DC-DC power consumption model, our heuristic algorithm has very low complexity compared to the previously proposed algorithms which solve the convex optimization problem iteratively and use binary search for optimal V_{cti}. The simplicity of our algorithm makes it suitable for runtime energy management for systems where the operating conditions vary rapidly.

- Compared to the previous proposed approach, our algorithm can achieve up to 41.23% improvement in energy efficiency for constant power charging and achieve up to 124.05% improvement in energy efficiency for a system charging and discharging under real solar power profile.

The rest of this chapter is organized as follows. Existing works on charge transfer in HEES systems are introduced in section 1. The energy harvesting system model and some assumptions are presented in section 2. The proposed heuristic energy management algorithm is described in section 3. Experimental results and discussions are presented in section 4. We conclude our work in section 5.

3.1 Related Work

In [10], the goal of charge replacement problem is to select which EES banks to be discharged and determine the discharging current of each selected EES bank to support a given load demand. In contrast with drawing power out of the HEES bank, the goal of charge allocation [11] is to solve the dual problem when the external power comes into the HEES system. The goal of charge migration problem [12] is to transfer energy internally from one EES bank to another, while improving the energy efficiency and reducing the energy loss during the transfer process. The HEES system could also be integrated with energy harvesting devices like PV cells [35] to store the maximum amount of solar energy. The main control knobs in these problems are the voltage of charge transfer interconnect V_{cti}, the set of active EES banks and the bank current. The algorithms for finding the optimal control variables in these problems are similar. The outer loop of these algorithms is to select the best V_{cti} for a given active set of EES banks using binary search, while the inner loop is to select the optimal set of active EES banks. In the inner loop, a convex optimization problem is solved iteratively until the set of active banks and the bank terminal voltages converge.

3.2 Architecture for Green Powered HEES

In this section, we introduce the system architecture of energy harvesting embedded system. We use the same energy harvesting and storage system model as the one in previous chapter. And we use our approximated DC-DC converter power consumption model in this work.

Figure 3.1 shows a block diagram of the system architecture considered in this work. The system consists of the following components, an Energy Harvesting Module (EHM), several heterogeneous Electrical Energy Storage (EES) banks and the embedded systems, i.e. the Energy Dissipation Module (EDM). All these components are connected together through Charge Transfer Interconnect (CTI) and DC-DC converters. We refer those DC-DC converters as Energy Conversion Modules (ECM). The DC-DC converters connected to EHM, EDM and ESS are called ECM_{EH}, ECM_{ED}, and ECM_{ES} respectively. The ECM_{ES} can further be divided into two function units, ECM_{ES_charge} and $ECM_{ES\_discharge}$, which involve in ESS charge and discharge processes.

3.2.1 Energy Harvesting System

On the left part of the system, energy is harvested from the environment by the energy harvesting system and distributed to other components of the system. Although any EHS can be integrated in our system, in this work, we assume that they are a set of photovoltaic (PV) arrays. The PV array exhibits non-linear current-voltage characteristics so that the output current has to be adjusted dynamically to match the output impedance to draw the maximum amount of power from the PV system. This technique is called Maximum Power Point Tracking (MPPT) and there are several methods proposed by previous works that could achieve the MPPT, e.g. the perturb and observe (P&O) method [36], the incremental conduction method [37] and the ripple

correlation control method [38]. When the power consumption of the charger is taken into consideration, a Maximum Power Transfer Tracking (MPTT) method is proposed [39] to guarantee the maximum amount of power input into the system. In this work, we assume the MPTT method is applied to the EHS so that the maximum amount of power goes into the system.

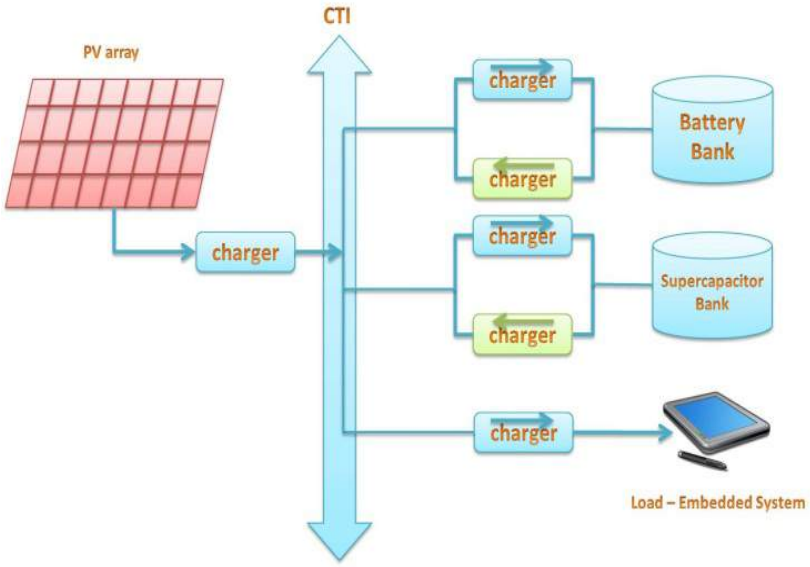


Figure 3.1 Energy Harvesting Embedded System Architecture

3.2.2 Balanced Reconfiguration of the HEES Bank

In this work, we consider two types of EES banks, one is supercapacitor bank and the other is battery bank. We assume the storage cells in an EES bank are identical and have same terminal voltages at all time in this work. As we mentioned, the power consumption of DC-DC converter depends on its input voltage V_{in} and output voltage V_{out}. And we will show in next section, the further apart between V_{in} and V_{out}, the higher P_{dcdc} will be. Therefore, we use EES bank reconfiguration technique to match the V_{in} and V_{out}. Let N be the number of cells in an EES bank, we define that a balanced configuration (m, n) of an EES bank has m cells in series and n cells in parallel such that $N = m \times n$. For example, for a four-cell bank, the balanced

configurations are (4,1), (2,2) and (4,1). Figure 3.2 shows the three possible balanced reconfiguration of the four-cell EES bank. Each cell has three switches: one series switch (S-switch, orange one) and two parallel switches (P-switches, green ones) except for the last cell. In the (4,1) configuration, all the S-switches are on while all the P-switches are off. On the other hand, in the (1,4) configuration, only the P-switches are on while all S-switches are off.

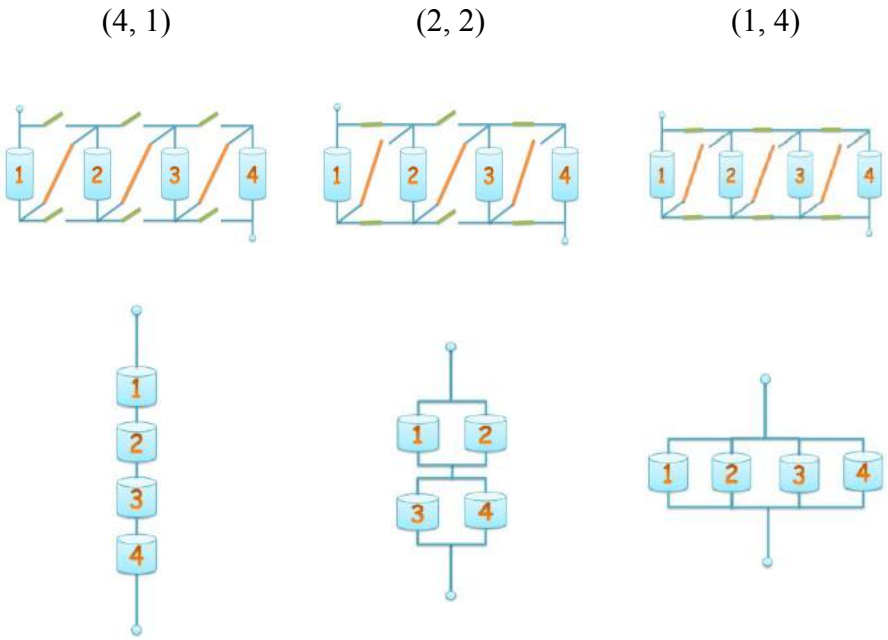


Figure 3.2 Reconfiguration of an EES Bank

3.3 Efficient Heuristic Energy Management Algorithm

3.3.1 Problem Formulation

For an energy harvesting embedded system modeled in the last section, energy is harvested from the renewable source and is either consumed immediately by the embedded system (load) or stored in the HEES bank for future use. As the energy flow through the system, some portion of the energy inevitably lost in the system. The major loss in such a system is

caused by the energy consumption of the DC-DC converters. Therefore, to improve the energy efficiency of the whole system, we have to reduce the energy wasted on the DC-DC converters.

In such a system, there are several control knobs we could manipulate to achieve the energy efficiency. For example, the voltage of the CTI V_{cti}, the configuration of the HEES bank and operating conditions of the embedded system. We formally define our problem as following.

Given: the harvesting power profile G(t), the load application characteristics.

Goal: dynamically adjust the V_{cti}, the configuration of the HEES bank and the operating voltage and current of the embedded system such that the energy wasted on the DC-DC converters is minimized and the energy stored in the HEES bank is maximized. And the deadline of the embedded application is satisfied.

3.3.2 Observations of the DC-DC Converter

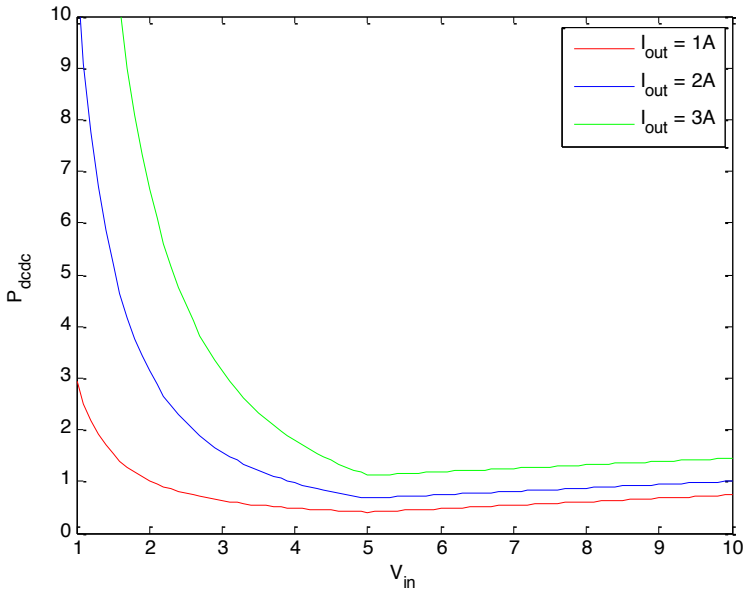


Figure 3.3 $\mathbf{P_{dcdc}}$ against $\mathbf{V_{in}}$ curve

To illustrate the relation between the DC-DC power consumption and the related variables, we fix the $V_{out} = 5V$, and according to the approximated DC-DC converter power model proposed in the last chapter, plot the P_{dcdc} against V_{in} for I_{out} is 1A (red), 2A (blue) and 3A (green) in Figure 3.3.

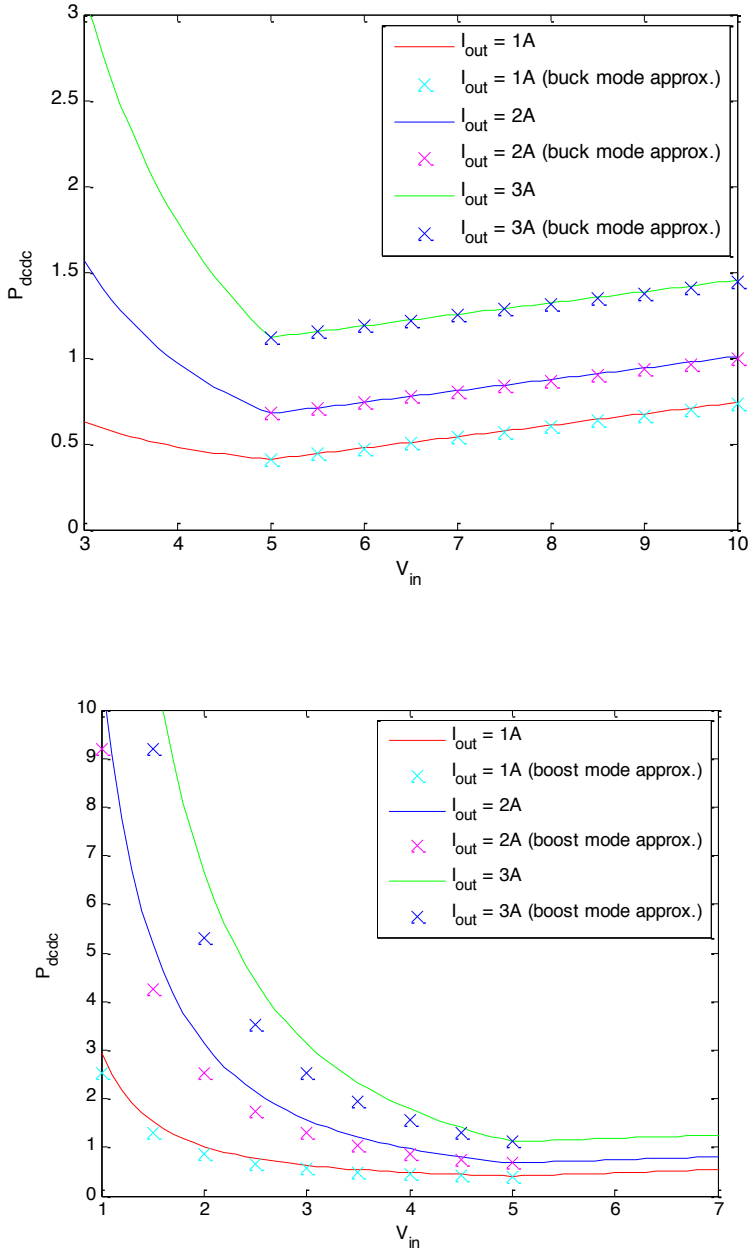


Figure 3.4 Fitting results of our approximation

Figure 3.4 compares the simplified P_{dcdc}(i.e. Equation (2.1)) and its original function given in Section 2.2.

From these figures, we have several observations based on the above analysis:

- Firstly, for the given V_{out} and I_{out}, P_{dcdc} reaches its minimum at $V_{in} = V_{out}$. According to this observation, we tried to minimum power loss on DC-DC converters connecting with EES banks through keeping reconfiguring the EES bank to make its terminal voltage match the CTI voltage as close as possible during charging and discharging

- Secondly, as the V_{in} moves away from V_{out}, P_{dcdc} increases much faster in the boost mode than in the buck mode. These two observations suggest that it is better to operate the DC-DC converter at buck mode and set the V_{in} close to V_{out}.

- Thirdly, P_{dcdc} increases as the I_{out} increases.

3.3.3 The Optimal V_{cti} for Discharging

Given the load power consumption characteristics and current status of one energy storage bank, we are interested in finding the optimal V_{cti} to maximize the discharging efficiency, i.e. to minimize the power consumption on the two DC-DC converters, (dcdc1 connects the bank and the CTI and dcdc2 connects the CTI and the load). Based on the approximation of P_{dcdc} in the last section, we can derive an analytic expression for the optimal V_{cti} in discharging process. Assume both the converters are operating in the buck mode, the operating voltage and current for the load is V_{load} and I_{load}, and the EES bank voltage is V_{bank}. Then the power consumption of the two DC-DC converters are given as

$$P_{dcdc1} = \alpha V_{bank} + \beta I_{cti}^2$$

$$P_{dcdc2} = \alpha V_{cti} + \beta I_{load}^2$$

We could substitute the I_{cti} with $(P_{dcdc2} + P_{load})/V_{cit}$, because the power provided by the CTI is consumed by the load and dcdc2. Our goal is to find the best V_{cti} to minimize $P_{dcdc} = P_{dcdc1} + P_{dcdc2}$. We differentiate P_{dcdc} with the variable V_{cti} and set it to 0 as following:

$$\frac{\partial P_{dcdc}}{\partial V_{cti}} = \alpha - \frac{2\alpha\beta\gamma}{V_{cti}^2} - \frac{2\alpha\gamma^2}{V_{cti}^3} = 0$$

i.e.

$$V_{cti}^3 - 2\beta\gamma V_{cti} - 2\gamma^2 = 0$$

In the above equation, $\gamma = \beta I_{load}^2 + P_{load}$. Because it is a cubic equation, it is analytically solvable [7], and we could find the optimal V_{cti} by solving the above equation. Let V_{opt} denote the optimal V_{cti} , one observation that can be made from the above analysis is that the value of V_{opt} is only determined by the load characteristics and does not depend on the status of the bank V_{bank}, as long as V_{bank} is high enough to make both DC-DC converters operates at the buck mode. However the P_{dcdc} does increase as the V_{bank} increase. This also suggests that we should set the V_{bank} to be the lowest level that is greater than V_{opt}. This provides a guideline for the reconfiguration of the HEES banks.

3.3.4 Efficient Energy Management Algorithm

In this section, we describe our efficient energy management algorithm. Our algorithm can be divided into two parts. The first part is called charging process, which is used when the harvesting power is high (e.g. during the noon) and in addition to supplying the load, there is

extra energy that can be stored in the HEES bank. The second part is called discharging process which is used when harvesting power is low or even zero, e.g. in the early morning or during the midnight. In this case, energy has to be drawn from the HEES system.

The main idea of our algorithm is to save the energy wasted on DC-DC converter to improve the energy efficiency by dynamically adjusting the operating voltage and frequency of the load, the voltage of interconnect V_{cti} and the configuration of the HEES system. We use one bank from the HEES system as the source for charging or discharging at a time so that only one DC-DC converter will be turned on, therefore the energy consumed by DC-DC converter can be reduced. We use the supercapacitor banks as the primary charging/discharging source while leaving the battery bank as the secondary source because supercapacitors have very high cycle efficiency and bigger voltage swings than the batteries so that the input and output voltage of the DC-DC convert can be matched better. In addition, the supercapacitor has no rate capacity effects as the battery.

We summarize our charging process in Algorithm 3.1. For a given workload, we first find the optimal V_{load} and I_{load} to minimize the energy consumption of the embedded application. Any existing optimization algorithm can be integrated because this step is independent to other parts of the algorithm. If the supercapacitor bank is not fully charged, we will charge the supercapacitor bank, otherwise we will charge the battery bank.

As we show in Section 3.3.2, the P_{dcdc} is minimized when V_{in} matches V_{out}, and buck mode is more energy efficient than boost mode. So the ideal charging situation would be $V_{cti} = V_{load} = V_{bank}$. But this might not always be possible depending on the amount of energy stored in the HEES bank. Therefore, we select a configuration that gives the highest bank voltage

$V_{bank,cap}$ that is smaller than V_{load} (line 3). In this case, if we set $V_{cti} = V_{load}$, one converter is operating at buck mode and another converter's input voltage and output voltage are same. Therefore the energy on DC-DC converters is reduced. However, if we are not able to find a configuration with $V_{bank,cap}$ less than V_{load}, then we will set all capacitors connecting in parallel to reduce the bank terminal voltage and set $V_{cti} = V_{bank,cap}$. In this case, the DC-DC converter on the supercapacitor bank side (i.e. dcdc1) has matching input and output voltage while the converter on the load side (i.e. dcdc2) operating at buck mode (line 6). If the supercapacitor bank is fully charged, we will charge the battery bank. The configuration for battery bank (line 8~11) is similar as the configuration for supercapacitor bank (line 3~6).

Algorithm 3.1: Charging Process

- Find the most energy efficient V_{load} and I_{load} setting for current load
- **if** (supercapacitor bank is not full) // Charge the supcap bank
- (s, p) = Choose a config. s.t. $V_{bank,cap} \leq V_{load}$ and V_{bank} is closet to V_{load}
- **if** $((s, p) == \emptyset)$
- $(s, p) = (1, N_{cap})$
- $V_{cti} = \max(V_{bank,cap}, V_{load})$
- **else** // Charge the battery bank
- (s, p) = Choose a config. s.t. $V_{bank,bat} \leq V_{load}$ and V_{bank} is closet to V_{load}
- **if** $((s, p) == \emptyset)$
- $(s, p) = (1, N_{bat})$
- $V_{cti} = \max(V_{bank,bat}, V_{load})$
- **return** V_{cti}, (s, p);

Algorithm 3.1 Algorithm for Charging Process

The discharging process is shown in Algorithm 3.2. Again, we first find the optimal V_{load} and I_{load} that minimize the energy consumption of the embedded application. Next, we use the method discussed in Section 3.3.3 to find the optimal CTI voltage V_{opt} and set $V_{cti} = V_{opt}$ (line 2). Then we will discharge from the supercapacitor bank if it is not empty. From the discussion in Section 3.3.3, it is better to operate the DC-DC converter at lower input voltage for energy

efficiency as long as it is in buck mode. Therefore, we select a configuration such that gives the lowest bank terminal voltage $V_{bank,cap} \geq V_{opt}$ (line 4). If there is no such feasible configuration, we will connect all supercapacitors in series. We will switch to battery bank for discharging until the supercapacitor bank is depleted. The configuration process for battery bank (line 8~10) is similar as that of supercapacitor bank (line 4~6).

Algorithm 3.2: Discharging Process

1. Find the most energy efficient V_{load} and I_{load} setting for current load
2. Find the optimal CTI voltage V_{opt} for (V_{load}, I_{load}), set $V_{cti} = V_{opt}$
3. **if** (supercapacitor bank is not empty) // Discharge from supcap bank
4. $(s, p) =$ Choose a config. s.t. $V_{bank,cap} \geq V_{opt}$ and $V_{bank,cap}$ is closet to V_{opt}
5. **if** $((s, p) == \emptyset)$
6. $(s, p) = (N_{cap}, 1)$
7. **else** // Discharge from the battery bank
8. $(s, p) =$ Choose a config. s.t. $V_{bank,bat} \geq V_{opt}$ and $V_{bank,bat}$ is closet to V_{opt}
9. **if** $((s, p) == \emptyset)$
10. $(s, p) = (N_{bat}, 1)$
11. **return** V_{cti}, (s, p);

Algorithm 3.2 Algorithm for Discharging Process

3.4 Experiment Results

To demonstrate the effectiveness of the proposed algorithm, we implement a C++ simulator to model the HEES system. The HEES consists of one supercapacitor bank and one battery bank. The supercapacitor bank consists of 4 capacitors with 100F capacitance [40][35]. We obtain other components like battery and DC-DC power converter model parameters from [41]. These parameters are obtained from the datasheets of the real devices. We assume the load power consumption is constant for a given period.

3.4.1 Constant Input Power Results

In this section, we compare the energy efficiency of our heuristic algorithm and that of the algorithm proposed in previous works [11][10][12][35]. The previous work is referred as Adaptive V_{cti}. As mentioned before, this algorithm uses binary search to adaptively adjust V_{cti} to minimize the energy consumed on the DC-DC converters. It does not consider the reconfiguration of the HEES.

Charging Power	(4, 1) (J)	(2, 2) (J)	(1, 4) (J)	Our (J)
1.75w	169.92	189.75	179.67	194.75
1.25w	15.05	14.77	13.49	19.05

Table 3.1 Energy stored in HEES for constant power input

Table 3.1 shows the amount of energy stored in the HEES system after it is charged using constant power input for 10 minutes. In this set of experiments, because there are 4 supercapacitors in the supercapacitor bank, the possible configurations of the bank are $(4,1)$, $(2,2)$ and $(4,1)$ as shown in the example of section **Error! Reference source not found.**. We onsider two scenarios when the input power to the system is 1.75w and 1.25w. In the former case, our heuristic algorithm stores 14.61%, 2.64% and 8.39% more energy compared to the Adaptive V_{cti} algorithms whose potential configurations are $(4,1)$, $(2,2)$ and $(4,1)$ respectively. While in the latter case, the proposed algorithm stores 26.58%, 28.96% and 41.23% more energy. As we could see, one supercapacitor configuration might work better than others at certain power input level, when the power input level changes, their efficiency will degrade. However, our algorithm works consistently better than the fixed configuration. Although the Adaptive V_{cti} algorithm always finds the optimal V_{cti} for a fixed configuration, it can still produce large imbalance between the input and output voltage of the DC-DC converters. Therefore it cannot minimize the power consumed on the converters. On the contrary, through reconfiguration, our algorithm can

always keep the terminal voltage of the DC-DC converters better matched. Thus it is more effective in reducing the energy wasted on the converters.

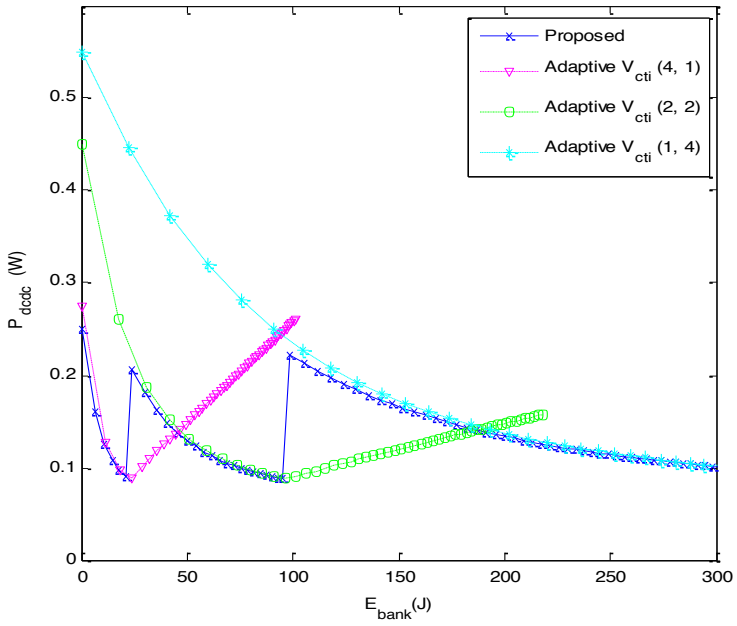


Figure 3.5 $\mathbf{P_{waste}}$ against the Energy Stored in the EES Bank

Figure 3.5 further explains why our algorithm can improve energy efficiency. This figure shows the wasted power against the energy stored in the supercapacitor bank during the charging process. Wasted power consists of the power consumed on the DC-DC converter and on the internal resistance of the EES bank. Apparently, when the EES bank does not store too much energy, it is better to connect them in series (i.e. (4, 1) configuration) to produce higher terminal voltage so that input and output voltage of the DC-DC converter can be better matched. As the charging process continues, the bank energy as well as the terminal voltage increases, the (4, 1) configuration first enters the boost mode, and the P_{waste} starts to increase. In this time, it is better to reconfigure the bank to (2, 2). Overall, our algorithm is able to track the most energy efficient bank configuration during the charging process.

Our algorithm could achieve better energy efficiency during discharge process as well. In the second set of experiment, we start using the energy stored in the HEES system to provide the load demands after the 10 minutes charging mentioned before. Our algorithm could last 17.24%, 4.11% and 32.59% longer compared to Adaptive V_{CTI} with $(4,1)$, $(2,2)$ and $(1,4)$ configurations respectively when the input power is 1.75W, and last 23.15%, 44.57% and 72.73% longer when in 1.25W.

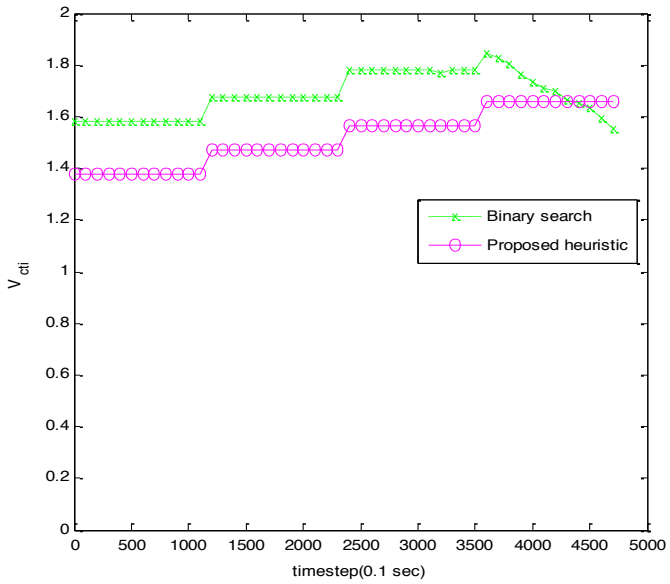


Figure 3.6 $\mathbf{V_{cti}}$ Comparison between Our Proposed Heuristic and Binary Search

Figure 3.6 shows the comparison of V_{cti} change between the base line Adaptive V_{cti} approach and our approach based on the approximation mode during the discharge process. In this experiment, we fix the bank configurations of both approaches to be $(4, 1)$. And the load changes from 1.0w to 1.3w with a period of 120 seconds. The Adaptive V_{cti} policy uses binary search method to find the V_{cti} which minimizes the energy wasted on DC-DC converters. Although the V_{cti} is optimal, the searching process is time consuming and might not be

applicable in time critical applications. Based on our approximation of the DC-DC converter power model, V_{cti} can be computed directly. As the figure shows, our method tracks the optimal V_{cti} pretty well.

3.4.2 Variable Sun Power Charging System

Config.	(4, 1) (J)	(2, 2) (J)	(1, 4) (J)	Our (J)
Energy	325.504	558.459	711.643	729.287
Impr.	124.05%	30.59%	2.48%	-

Table 3.2 Energy stored in HEES for real solar irradiation

In this section, we integrated a real solar irradiation profile [39] as the power input to our system. We start charging the system from 11:00am to 12:00pm when the solar irradiance reaches its maximum value. As Table 2.2 shows, in this charging profile, our algorithm outperforms the baseline algorithm with previously mentioned three configurations by 124.05%, 30.59% and 2.48% respectively. During the charging process, the bank terminal voltages of configuration (4, 1) and (2, 2) exceeds the input voltage of the DC-DC converters for a long time, which means they spend a large portion of time in boost mode which has low energy efficiency. As a result, large amount of energy wasted on DC-DC converters and less amount of energy is stored in the HEES bank for future use. We again compare the discharging time after this charging process. The proposed algorithm last 140.22%, 34.02% and 5.29% longer compared to (4,1), (2,2) and (4,1) configuration respectively.

3.5 Chapter Summary

In this chapter, we proposed an efficient heuristic algorithm for the energy efficiency optimization problem of charge allocation and replacement in an EHS/HEES equipped embedded system. Our goal is to minimize the energy wa sted on the DC-DC converters while

maximizing the energy stored in the HEES system. To avoid the complexity of iteratively searching the optimal operating point, we developed an approximation power model of the DC-DC converter. Based on this model, our algorithm tries to match the input and output voltages of the DC-DC converters by dynamically adjusting the V_{dd} of the embedded load, the charge transfer interconnect voltage V_{cti} and the storage bank terminal voltage using reconfiguration. Experimental results show that our algorithm could store up to 124.05% more energy during charging process and last up to 140.22% longer.

Chapter 4 Task Scheduling and Mapping for Applications on Heterogeneous Multi-Processor Systems

When the harvesting power is short, the load demands are supported by the EES bank only. Many works have been proposed to reduce the power dissipation on the battery-powered devices and systems. In this chapter, we will discuss the task mapping, scheduling and Dynamic Voltage and Frequency Scaling (DVFS) algorithms on the heterogeneous system, which is one of the solutions to raise the system performance and improve the energy efficiency.

Many real time applications demonstrate uncertain workload that varies during runtime. One of the major influences of the workload fluctuation is the selection of conditional branches which activate or deactivate a set of instructions belonging to a task. Many techniques have been proposed that perform power aware task mapping, ordering and DVFS for applications with fixed workload [23][49][50]. However, they may not achieve their best performance with real life applications that demonstrate workload variations. This work considers task scheduling and voltage selection of applications with variable workload on a heterogeneous multiprocessor system-on-chip. We focus on the set of applications which can be decomposed into repeated

tasks with relatively constant execution time. The uncertainty of the workload is reflected by the random activation and deactivation of certain tasks during run- time and it is captured by branch selections among tasks. We assume that such branch information is observable to the scheduling software to assist runtime scheduling and power management of the system. Examples of such task level branching include branches that enable or disable IDCT function during MPEG decoding or branches that select different modulation schemes for preamble and payload based on 802.11b physical layer standard. We model an application with dynamic branch selections using a conditional task graph (CTG). In this work, we capture the workload dynamics using Conditional Task Graph (CTG) and propose a set of algorithms for the optimization of task mapping, task ordering and dynamic voltage/frequency scaling of CTGs running on a heterogeneous multicore system.

We study the power aware task mapping and scheduling techniques for the CTGs with a particular interest in their performance on the heterogeneous multi-core system. It is important to investigate the low power techniques for the heterogeneous multi-core system, not only because it is the architecture of many embedded computing systems such as the Pasta sensor node [51], the LEAP system [52], and the mPlatform [53], but also because, due to the within-die variation and transistor wear-out, a homogenous multi-core system will exhibit heterogeneity over time [54].

The proposed framework optimizes the task mapping, task ordering and dynamic voltage/frequency scaling of CTGs running on a heterogeneous multi-core system. Due to the probabilistic branching, the execution of CTG is a random process. Our goal is to minimize the mean power consumption instead of the worst case power consumption of the application while meeting the hard real-time constraint. Our mapping algorithm balances the latency and the

energy dissipation among heterogeneous cores in order to maximize the slack time without significant energy increase. Our scheduling algorithm minimizes the mean power consumption of the application while maintaining a hard deadline constraint. Our DVFS algorithm dynamically distributes slacks during runtime, and achieves comparable energy reduction as the solutions found by non-linear mathematical programming. Due to its capability of slack reclaiming, our DVFS technique is less sensitive to the slight change in hardware or task execution and works more robustly than the conventional DVFS techniques. Compared to the existing works, the uniqueness of our approach can be summarized as the following:

- The proposed framework optimizes task mapping and scheduling simultaneously. We modify the Dynamic Level based Scheduling (DLS) algorithm [55] to find mapping and scheduling of CTGs on a heterogeneous platform. The complexity of task mapping and ordering is O(MN) where M is the number of processing elements (PEs) in the system and N is the number of tasks in the task graph.
- We consider the application with conditional execution as a random procedure in which the branch selection probabilities can be profiled (or measured). Our algorithm utilizes such statistical information in each optimization step to reduce the mean power consumption.
- The proposed slack reclaiming based DVFS algorithm has pseudo-linear complexity with the respect to the number of tasks in the CTG. It also performs more robustly than the DVFS solution found by mathematical programming.

The rest of this section is organized as follows: section 1 presents related works; section 2 introduces the application and hardware architecture models; section 3 and section 4 present in detail the underlying scheduling, mapping and DVFS algorithms. Section 5 gives the

experimental results. Finally, section 6 concludes the work.

4.1 Related Works

Task mapping, ordering, and DVFS for CTGs have been studied in many literatures. In [56], Dolif et al. propose an exact analytic formulation of the stochastic objective function for conditional task mapping and scheduling based on task graph analysis. Their approach considers stochastic branching probabilities and improves the overall performance by optimizing the bus activity. In [57], Eles et al. present an approach to minimize the worst case delay, which considers the communication workload and branch control. A recursive algorithm is proposed to generate the schedule table, which stores the start time of each task at different execution scenarios. In [58], Xie et al. propose a task mapping and scheduling algorithm which is sensitive to mutually exclusive tasks in the CTG. Their algorithm exploits the processor sharing for mutually exclusive tasks to minimize the worst case delay of the CTG. All these works do not minimize energy dissipations.

Wu et al. [59] utilized a scheduling table for runtime DVFS. The slack time of each task can be derived from the table. A genetic algorithm based task mapping algorithm is proposed for further energy savings. An implicit assumption of this technique is that all the conditional branches will be selected with equal probability, which might not be realistic for a real system. Furthermore, the GA based task mapping algorithm has very high complexity because the inner loop of this algorithm needs to perform the task ordering and stretching for the entire CTG.

Shin et al. [60] proposed an algorithm for task scheduling and DVFS of CTG which considers the run-time behavior. The algorithm is based on a modified list scheduling and non-

linear programming (NLP) based voltage/frequency optimization. Because the algorithm schedules a task with different start time and speed under different branch selection scenarios, it is referred as condition aware scheduling. However, their task mapping is assumed to be fixed.

In [61] and [62] we proposed a Communication Aware Probability-based (CAP) scheduling algorithms for CTGs on a DVFS enabled multiprocessor platform. However, the algorithm assumes that all the PEs have the same energy-performance characteristics. Furthermore, its DVFS algorithm relies on evaluating all the paths, which has exponential complexity. In this work, we completely revised the CAP algorithm and consider a heterogeneous system where each PE has unique power-performance characteristics. The complexity of the DVFS algorithm is also significantly reduced.

Power aware task mapping, scheduling and DVFS techniques for heterogeneous multiprocessor system has been studied in [23][24][25][26][27][28][29]. In [23], a two-phase algorithm is proposed for distributed real-time embedded system. A static scheduling algorithm first generates the schedule for all tasks in the CTG, and then a dynamic scheduling algorithm determines the speed ratio for mutual exclusive tasks during the runtime. In [24], the authors focus on the critical paths and distribute the slack time over the tasks on the critical path to achieve energy savings. Both works assume that only processors have DVFS capability and their power-performance characteristics are identical. [25] and [26] study the energy-efficient software partition in heterogeneous multiprocessor system. In [25], Goraczko et al. formulate the software partitioning problem as an integer linear programming (ILP) constrained by the deadline. The solution of the ILP gives the number of processors needed by the system and task mapping information. In [26], Yang et al present an approximate algorithm for task partition to find near-optimal task mapping and scheduling with minimum energy consumption. In both

these works, tasks are assumed to be independent to each other. A leakage aware DVFS algorithm is proposed in [27]. Unlike our work, it only considers heterogeneous system with only 2 types of processing elements (PE), one with high performance and high power consumption while the other with low performance and low power consumption. The objective is to maximize the number of tasks mapped on the PEs with low performance while satisfy the deadline constraint. Deadline Monotonic (DM) rule is used in the task scheduling while Random Algorithm (RA), First-Fit Decreasing Density (FFDD), Genetic Algorithm (GA) and Simulated Annealing (SA) are used for task mapping. In [28], Azevedo et al propose to do intra task DVS based on the program checkpoint, while in [29], Yang et al rely the Pareto curve profiling for a task graph to do online task mapping and scheduling. Both of these works do not consider energy and performance heterogeneity of the multiprocessor platform.

4.2 Application and Hardware Architecture Models

The CTG that we are using is similar to the one specified in [60]. A CTG is a directed acyclic graph $< V, E >$. Each vertex $\tau \in V$ represents a task. An edge $e=(\tau_i, \tau_j)$ in the graph represents that the task τ_i must complete before task τ_j can start. A conditional edge e is associated with a condition C(e). The tail node of a conditional edge is called branch fork node. We use prob(e) to denote the probability that the condition C(e) is true; and we use $succ(\tau)$ and $pred(\tau)$ to denote the set of successors and predecessors of a taskτ. We assume that all tasks have the same deadline.

A node is activated when all its predecessors are completed and the conditions of the corresponding edges are satisfied. The condition that the task τ_i is activated is called the

activation condition and denoted as $X(\tau_i)$. It can be written as $\vee_{\tau_k \in pred(\tau_i)} \left(C(\tau_k, \tau_i) \wedge X(\tau_k)\right)$, where "$\wedge$" and "$\vee$" represent logic operations "AND" and "OR".

A minterm m is a possible combination of all conditions of CTG. We use M to denote the set of all possible minterms of CTG. A task τ is associated with a minterm m if $X(\tau)$ is true when m is evaluated to be 1. The set of minterms with which τ is associated is referred as activation set of τ and denoted as $\Gamma(\tau)$. Two tasks τ_i and τ_j are mutually exclusive if they cannot be activated at the same time, i.e. $\Gamma(\tau_i) \cap \Gamma(\tau_j) = \phi$. The set of all unique $\Gamma(\tau)$, $\forall \tau \in V$, is called activation space and denoted as T.

The amount of data that pass from one task to another is also captured by the CTG. Each edge (τ_i, τ_j) in the CTG associates with a value $Comm(\tau_i, \tau_j)$ which gives the communication volume. Finally, we assume a periodic graph and use a common deadline for the entire CTG.

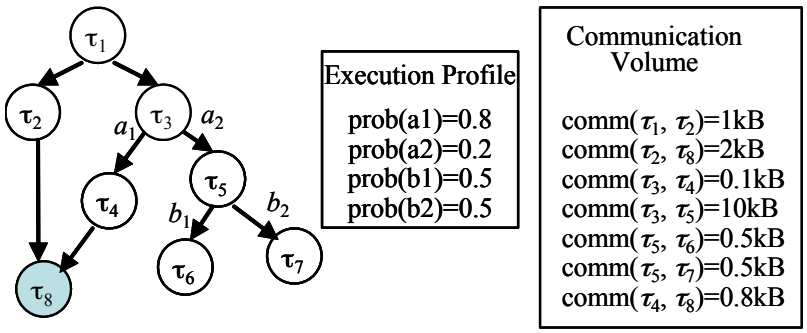


Figure 4.1 an example of a CTG

Figure 4.1 shows an example of a CTG. The edges coming out from τ_3 and τ_5 are conditional edges therefore nodes τ_3 and τ_5 are branch fork nodes. The symbol marked beside a conditional edge denotes the condition under which the edge will be activated. For example, $C(\tau_3, \tau_4) = a_1$. There are total of 4 minterms in the CTG and $M = \{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}$. We have

$\Gamma(\tau_1) = \Gamma(\tau_2) = \Gamma(\tau_3) = \{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}$, $\Gamma(\tau_4) = \Gamma(\tau_8) = \{a_1b_1, a_1b_2\}$, $\Gamma(\tau_5) = \{a_2b_1, a_2b_2\}$, $\Gamma(\tau_6) = \{a_2b_1\}$, $\Gamma(\tau_7) = \{a_2b_2\}$. The execution profile and communication volume are given beside the CTG. The activation space is:

$$T = \{\{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}, \{a_1b_1, a_1b_2\}, \{a_2b_1, a_2b_2\}, \{a_2b_1\}, \{a_2b_2\}\}$$

4.3 Task Mapping and Scheduling

Our task mapping and scheduling is based on a modified Dynamic Level based Scheduling (DLS) [55] algorithm that finds task mapping and ordering simultaneously. Our goal is to implement an on-line scheduling, mapping DVFS algorithm that adaptively change with the branch probability. That's why we choose the DL-based approach. The DLS algorithm is a list scheduling algorithm. It maintains a ready list that stores tasks whose predecessors have been scheduled and mapped. For each task τ_i in the ready list, the dynamic level $DL(\tau_i, p_j)$ is calculated using the following equation:

$$DL(\tau_i, p_j) = SL(\tau_i) - AT(\tau_i, p_j) \quad (4.1)$$

where p_j is one of the processing elements, $SL(\tau_i)$ is the static level of task τ_i, which is equal to the longest distance from node τ_i to any of the end nodes in the task graph (i.e. the longest remaining execution time), $AT(\tau_i, p_j)$ is the earliest time that task τ_i can start at processor p_j, it is calculated as $AT(\tau_i, p_j) = \max\lfloor DA(\tau_i, p_j), TF(p_{j)}\rfloor$, where $DA(\tau_i, p_j)$ is the earliest time that all data required by node τ_i is available at the j^{th} PE with the consideration of both computation and communication delay, and $TF(p_j)$ is the time that the last task assigned to the j^{th} PE finishes its execution. The pair of (τ_i, p_j) which gives the maximum dynamic level will be selected and the mapping is performed accordingly. The task is scheduled to be started at the time

$\max\left\lfloor DA(\tau_i, p_j), TF(p_{j)}\right\rfloor$. After that, the ready list is updated and the dynamic level of each task in the ready list is re-calculated. Because we need to calculate the DL for each task processor pair in the system, the complexity of the DLS algorithm is O(|V|*N), where |V| is the number of tasks and N is the number of PEs in the system.

The original DLS algorithm does not handle mutual exclusive tasks. Its goal is to minimize the makespan of the application. Although minimal makespan scheduling usually enables more aggressive DVFS and hence lower power consumption in a homogenous multi-core system, this is not always true in a heterogeneous system. In this work we adopt the general flow of the DLS algorithm but modify the way that the dynamic level (DL) is calculated to consider the mutual exclusiveness among conditional tasks, the probability of branch selection and the energy performance heterogeneity among processors.

4.3.1 Minimum Average Makespan CTG Scheduling

Our first step is to modify the DL function to explore the mutual exclusiveness among conditional tasks and consider the branch selection probability. The goal of such modification is to minimize the average makespan instead of the worst case makespan of the application. In the modified algorithm, the static level (SL) of a task τ is used to represent the average remaining execution time when τ has just started.

The static level of a non-branching node is the maximum static level of its successors plus the average WCET (denoted as $WCET_{avg}$) of itself. Let $Succ(\tau_i)$ be the set of successors of τ_i, equation (4.2) calculates the SL of a non-branching node.

$$SL(\tau_i) = WCET_{avg}(\tau_i) + \max_{\tau_j} SL(\tau_j), \tau_j \in Succ(\tau_i) \qquad (4.2)$$

The static level of a branch fork node is the mean of the static level of all its successors plus the $WCET_{avg}$ of itself. Let c_{ij} denote the condition of edge (τ_i, τ_j), equation (11) calculates the static level of a branch fork node.

$$SL(\tau_i) = WCET_{avg}(\tau_i) + \sum_j prob(c_{ij}) SL(\tau_j), \tau_j \in Succ(\tau_i) \quad (4.3)$$

where $prob(c_{ij})$ is the probability that condition c_{ij} is true.

For a task τ_i without successors, its static level equals to its average worst case execution time: $SL(\tau_i) = WCET_{avg}(\tau_i)$.

The dynamic level of task-processor pair (τ_i, p_j) is calculated as the following equation.

$$DL(\tau_i, p_j) = SL(\tau_i) - AT(\tau_i, p_j) + \delta(\tau_i, p_j) \quad (4.4)$$

The term $\delta(\tau_i, p_j)$ is the difference between $WCET_{avg}(\tau_i)$ and $WCET(\tau_i, p_j)$ which accounts for the performance heterogeneity. Adding this offset ensures correct evaluation of a task's DL for different processors since SL is computed using the average WCET. $AT(\tau_i, p_j)$ is the earliest time that task τ_i can start on processor p_j. It must satisfy the following two conditions:

- At time $AT(\tau_i, p_i)$ all data required by τ_i is available at p_i, i.e. $AT(\tau_i, p_i) \geq DA(\tau_i, p_i)$.
- If a task τ_j is scheduled during the interval $[AT(\tau_i, p_i), AT(\tau_i, p_i) + WCET(\tau_i, p_i)]$, then τ_j and τ_i are mutually exclusive. This condition allows two mutually exclusive tasks to share the same processor at the same time, and thus making the schedule more efficient.

The data available time of a task τ_i on processor p_j is calculated as the latest time when its predecessor completed execution plus the time to transfer the data, i.e. $\mathrm{DA}(\tau_i, p_j) = \max_{\tau'}[\mathrm{ET}(\tau') + \mathrm{comm}(\tau', \tau_i)/\mathrm{BW}(p', p_j)]$, $\forall \tau' \in \mathrm{pred}(\tau_i)$, where p' is the processor where τ' is mapped to and ET(τ') is the end time of task τ'. Here we assume point-to-point communication between two processors. Please note that we can extend the framework to systems with shared communication links by considering all data communications as separate tasks and all communication links as separate resources and schedule communication tasks in the similar way as the computation tasks. Please note that our DLS algorithm could also handle the overhead of reconfiguring voltage and frequency by adding the overhead to the variable $\mathrm{AT}(\tau_i, p_j)$.

4.3.2 Balancing Energy and Performance in a Heterogeneous Platform

In a homogenous system, because all PEs are identical in energy performance characteristics, the total energy dissipation is determined solely by the slowdown ratio. Minimizing the average makespan will generally produce more slacks for each task and enable a lower execution speed for more energy reduction. Such monotonic relation does not exist in heterogeneous system, where the application energy dissipation does not only depend on the slowdown ratio, but also on which core the application is mapped to.

There is a fundamental tradeoff between energy and performance in a heterogeneous system. Mapping a task τ to a faster PE could result better performance and shorter execution time but will incur higher energy consumption. To compensate the extra energy incurred by task τ, later tasks have to choose slower PEs to run and result longer latency. The performance gain by running task τ at faster speed will diminish. From this perspective, higher energy

consumption is equivalent to longer execution latency. Our modified DLS algorithm is already capable of handling performance heterogeneity among PEs (by introducing $\delta(\tau_i, p_j)$ in Equation (4.4)), to further consider energy heterogeneity in the algorithm, we propose a heuristic method to convert energy heterogeneity into performance heterogeneity, and handle them in a unified way.

In the rest of this sub-section, we use $\mathrm{WCET}_{\mathrm{avg}}(\tau)$ and $\mathrm{WCET}(\tau, p)$ to denote the average WCET of task τ and the WCET of task τ on processor p. $E_{total}(p,T)$ and $E_{avg}(T)$ to denote the overall cycle energy of processor p and the average cycle energy of all PEs when their clock period is T. We use prob(τ) to denote the probability that task τ is activated. Finally, we use $\Delta E(\tau, p)$ to denote the difference between the energy dissipation of task τ running on p and the mean energy dissipation of τ running on the heterogeneous system:

$$\Delta E(\tau, p) = [E_{total}(p, T_{min}) \cdot WCET(\tau, p) - E_{avg}(T_{min}) \cdot WCET_{avg}(\tau)] * prob(\tau)$$

Here $T_{min} = {}^{1}/_{f_{max}}$ is the period of the fastest clock. Note that $\Delta E(\tau, p)$ represents the extra energy of running task τ on p, and it could be positive or negative. In order to make up for this extra energy, we must run the entire application at a lower clock frequency and lower supply voltage (or a higher clock frequency and higher supply voltage if $\Delta E(\tau, p)$ is negative.) Assume that setting the clock period to T' can balance the energy. Then we have,

$$\Delta E(\tau, p) = \sum_{\tau' \in V} WCET_{avg}(\tau') * [E_{avg}(T_{min}) - E_{avg}(T')] \qquad (4.5)$$

Here we use $\sum_{\tau \in V} WCET_{avg}(\tau) * E_{avg}(T)$ to approximate the total energy of the application when all processors are running at clock frequency 1/T. This is only a rough estimation of the

true energy dissipation of the application because not all tasks have been scheduled and mapped yet. Solving Equation (4.5) we get the T':

$$T' = {E_{avg}}^{-1}[E_{avg}(T_{min}) - \frac{\Delta E(\tau, p)}{\sum_{\tau' \in V} WCET_{avg}(\tau')}]$$

To make up for the extra energy incurred by mapping τ to processor p, the total execution time of the application must be extended by $(T' - T_{min}) \sum_{\tau} WCET_{avg}(\tau)$. Therefore, we adjust the WCET of τ and calculate its effective WCET as:

$$WCET_{eff}(\tau, p) = WCET(\tau, p) + \lambda(T' - T_{min}) \sum_{\tau} WCET_{avg}(\tau) \qquad (4.6)$$

The parameter λ is introduced to provide a tradeoff between performance and energy. It will be referred as the balance factor in the rest of the chapter. When λ is 0, the algorithm is reduced to the minimum average makespan scheduling as we introduced in Section 4.3.1 regardless of energy heterogeneity among processors. When λ is very large, the algorithm maps a task to the minimum energy processor, without making effort to reserve slacks for the DVFS algorithm which will be applied later.

4.3.3 Adding Control Edges to the CTG

After task mapping and ordering, the original CTG must be modified to reflect the new precedence constraints. Edges representing control dependencies will be inserted. These control edges will be used to find the execution paths for DVFS control. a control edge (τ_i, τ_j) must have the following properties:

- Both τ_i and τ_j are mapped to the same PE.
- Task τ_i is executed before task τ_j.

- If there is another control edge (τ_k, τ_j), then τ_k and τ_i do not have identical activation set, i.e. $\Gamma(\tau_k) \neq \Gamma(\tau_i)$.
- If a task τ_k is mapped to the same PE as τ_i and τ_j, and $\Gamma(\tau_k) = \Gamma(\tau_i)$, then τ_k is executed either after τ_j or before τ_i.

Algorithm 4.1: Relink CTG

```
1.  for each task τ∈V {
2.      for each activation set Γ∈T {
3.          τ_link = NULL;
4.          for each task τ' executed before τ  on the same PE {
5.              if Γ(τ')=Γ and τ' is executed before τ_link then τ_link = τ';
6.          }
7.          if τ_link ≠ NULL then add control edge from τ_link  to τ ;
8.      }
9.  }
```

Algorithm 4.1. Relink CTG

Algorithm 4.1 gives the algorithm that adds control edges to the CTG. For each task τ and each unique activation set Γ, it searches for the latest task whose activation set is Γ and is executed before τ. Let C denote the number of unique activation sets in the CTG, the worst case complexity of Algorithm 1 is $O(C *|V|^2)$. The modified CTG will be used by the DVFS algorithm introduced in the next section.

4.4 DVFS Based on Slack Reclaiming

Although it gives the optimal solution, solving the above mentioned NLP is time consuming. In the next, we will introduce a heuristic DVFS algorithm based on slack reclaiming (SR). In the rest of the chapter, it is referred as SR_DVFS. In order to apply the slack reclaiming algorithm during the runtime, the following information is needed for each task: overall average

remaining execution time ($ARET_\tau$), the maximum remaining execution time ($MxRET_\tau$), the average remaining execution time along the most critical path on different PEs ($ARET\_PE_\tau[i]$, $1 \leq i \leq N$), and a look-up-table of slack distribution rules (LUT_τ). We divide the total available slack into multiple discrete levels. Given the available slack l, the element $LUT_\tau[l]$ specifies the amount of slacks (out of l) that should be distributed to the PE that τ is mapped to.

Algorithm 4.2: Calc. ARET, MxRET

```
• Topological sort the modified CTG;
• for each task τ starting from the lowest topological order {
•     ARET_τ = 0;
•     for each minterm m∈Γ(τ) {
•         max_aret = 0;
•         for each successor τ_i of τ  with m∈Γ(τ_i) {
•             if(max_aret < ARET_τi +comm(τ, τ_i )/BW(τ, τ_i ))
•                 max_aret = ARET_τ i +comm(τ, τ_i )/BW(τ, τ_i );
•          }
•         ARET_τ = pr(m)/pr(τ )* max_aret;
•     }
•     ARET_τ = ARET_τ  + WCET(τ, p_τ);
•     MxRET_τ = 0;
•     for each successor τ_i of τ  {
•         if(MxRET_τ < MxRET_τi +comm(τ, τ_i )/BW(τ, τ_i )+WCET(τ, p_τ))
•             MxRET_τ = MxRET_τi +comm(τ, τ_i )/BW(τ, τ_i )+WCET(τ, p_τ);
•     }
• }
```

Algorithm 4.2 Algorithm for ARET, MxRET calculation

Algorithm 4.2 gives the algorithm that calculates the values of ARET and MxRET of each task. The algorithm processes the tasks based on the reverse topological order of the modified CTG. Steps 3 through 10 in the algorithm calculates the ARET of a task τ. For each minterm m, the ARET of τ is the maximum value of its successor's ARET plus the data communication time. The overall ARET of τ is the sum of the ARETs under each minterm m

weighted by the condition probability that minterm m is true given that task τ is activated (i.e. pr(m)/pr(τ)). Finally, the ARET of task τ should include the execution time of τ itself as specified in step 10. Steps 12 through 17 in the algorithm calculate the MxRET. It is equal to the largest MxRET of τ's successors plus the data communication time. Let S denote the maximum number of immediate successors of a task, the worst case complexity of the algorithm is O(MS|V|).

The values of ARET and MxRET are used to determine the remaining slack that will be distributed to the current task and the following tasks. In a heterogeneous multi-core system, where each PE has different power-performance tradeoff characteristics, the slacks will be distributed to different PEs non-uniformly. Therefore, we need to know not only the overall average remaining execution time, but also the remaining execution time on different PEs.

Based on Algorithm 4.2, the ARET of a task reflects the longest path in the modified CTG. Considering only the longest path while ignoring the power-performance tradeoff efficiency of each PE will not result the best slack distribution policy. Let $f_{EPT}(p)$ denote the energy-performance tradeoff (EPT) factor of processor p. It is calculated as $f_{EPT}(p) = \frac{dE_{total}(T)}{dT}\Big|_{T=1/f_{max}}$. We choose the path that has the highest EPT as the most critical path and collect the average remaining execution time on different PEs along this critical path.

Algorithm 4.3 gives the algorithm that calculates the variables ARET_PE[i] for each PE i. For each task τ, a variable $AR_{EPT}(\square)$is maintained. It records the average total EPT of the remaining tasks along the most critical path (i.e. the path with the highest EPT efficiency). The ARET_PE of a task τ under minterm m is determined by the ARET_PE of its successor that is

active under the same minterm and has the largest remaining EPT. The overall ARET_PE of a task is the sum of its ARET_PEs under different minterms weighted by the conditional probability that the minterm m is active given the condition that the task τ is going to be executed. Similar to Algorithm 6, the worst case complexity of Algorithm 4.3 is also O(MS|V|).

Algorithm 4.3: Calc. ARET_PE

- **for** each task τ starting from the lowest topological order {
- $ARET\_PE_\tau[i] = 0, 0 \le i \le N;$
- **for** each minterm $m \in \Gamma(\tau)$ {
- $eng = 0;$
- Find τ_i with the largest $eng_{\tau I}$, $\tau_I \in succ(\tau)$ and $m \in \Gamma(\tau_i)$
- $eng_\tau = pr(m)/pr(\tau) * eng_{\tau i};$
- $ARET\_PE_\tau[i] = ARET\_PE_\tau[i] + pr(m)/pr(\tau) ARET\_PE_{\tau i}[i], 0 \le i \le N;$
- }
- $ARET\_PE_\tau[p_\tau] = ARET\_PE_\tau[p_\tau] + WCET(\tau, p_\tau);$
- $eng_\tau = eng_\tau + WCET(\tau, p_\tau) * E_{total}(p_\tau)$
- }

Algorithm 4.3 Algorithm for ARET_PE Calculation

As we can see, the ARET_PE shows the computing time distribution along the path with the highest EPT efficiency. This information determines how the average available slack will be distributed.

Let $E_{p_i}(T_{clk} \cdot L)$ denote the energy dissipation over L clock cycles on processor p_i when the clock period is set to T_{clk}. We know that $E_{p_i}(T_{clk} \cdot L) = LE_{total}(p_i, T_{clk})$. Replacing $T_{clk} \cdot L$ with a new variable D, $E_{p_i}(D)$ gives the energy dissipation of p_i running for a duration D at clock speed $1/T_{clk}$. D is a linear function of T_{clk} and $E_{p_i}(D)$ is a convex function of D. Assume that, scaling the voltage and frequency extends task execution time from D_1 to D_2. Let $ES_i(D_1, D_2)$ denote the energy saving, i.e. $ES_i(D_1, D_2) = E_{p_i}(D_1) - E_{p_i}(D_2)$.

Theorem 2. The energy savings of the optimal continuous DVFS is incremental, i.e. $ES_i(D, D+Y) = ES_i(D, D+X) + ES_i(D+X, D+Y)$, $\forall X \in (0, Y)$. Furthermore, the energy saving $ES_i(D, D+X), X > 0$ is a decreasing function of D.

Proof: The first part of the theorem can be proved from the definition of the function ES(). An optimal DVFS with continuous voltage and frequency levels will set the T_{clk} to be D/L. The left side of the equation is:

$$ES_i(D, D+Y) = CE_{total}(D/L) - CE_{total}((D+Y)/L)$$

The right side of the equation is:

$$\begin{aligned} &ES_i(D, D+X) + ES_i(D+X, D+Y) \\ &\quad = CE_{total}(D/L) - E_{total}((D+X)/L) + CE_{total}((D+X)/L) \\ &\quad - CE_{total}((D+Y)/L) \end{aligned}$$

The left and right sides equal to each other.

To prove the second part of the theorem, we only need to prove that $ES_i(D, D+X) - ES_i(D+Y, D+Y+X) > 0, \forall X, Y > 0$.

The left part of the inequality is:

$$ES_i(D, D+X) - ES_i(D+Y, D+Y+X) = E_{p_i}(D) - E_{p_i}(D+X) - E_{p_i}(D+Y) + E_{p_i}(D+Y+X)$$

Because $E_{p_i}()$ is a convex function, we know that:

$$E_{p_i}(D) + E_{p_i}(D+Y+X) \square E_{p_i}(D+X) + E_{p_i}(D+Y).$$

The second part of the theorem is also proved.

Algorithm 4.4: Generate slack distribution table for task τ

- Input: $ARET_\tau$, $ARET\_PE_\tau[i]$, $ES_i()$, S_{max} (maximum slow down ratio)
- **for** each PE i with non-zero $ARET\_PE_\tau[i]$ {
- $ES_i[l] = ES_i(ARET\_PE_\tau[i]+l, ARET\_PE_\tau[i]+l+1)$,

  $1 \le l < \min(\text{deadline}, S_{max} * ARET\_PE_\tau[i])$; }

- Clear $LUT_\tau[]$, idx = 0, slack = 0;
- **While** (not all $Es_i[]$ arrays are empty) {
- Compare the leading elements in $ES_i[]$ arrays, $1 \le i \le N$;
- Assume the largest value located in the pth array (i.e. $ES_p[0]$);
- If $p = p_\tau$ then $LUT_\tau[idx++] = ++slack$;
- else $LUT_\tau[idx++] = slack$;
- delete the leading element in $ES_p[]$ and point to the next one;
- }

Algorithm 4.4 Slack Distribution LUT Generation

Based on these properties, we design the algorithm that generates the slack distribution table shown in Algorithm 4.4. The input of the algorithm is the average remaining execution time (i.e. $ARET_\tau$), the remaining execution time on different PEs (i.e. $ARET\_PE_\tau[i]$, $1 \le i \le N$), the energy saving characteristics of each PE, $ES_i()$, and the maximum slow down ratio S_i of each PE i. The algorithm increases the slack allocation to each PE with uniform time step and records the corresponding energy saving in an array $ES_i[]$ (Steps 2~4). This procedure ends when the maximum slow down ratio is reached. The value of $ES_i[j]$ tell us the energy saving of extending the total execution time of all tasks on PE_i from D+j-1 to D+j, where D is the minimum execution time. Based on Theorem 2, the array $ES_i[]$ is decreasing. Using the information in $ES_i[]$, in Steps 7~11, we distributed slacks one unit by one unit to the PEs that has the highest energy saving. The amount of slacks distributed to p_τ is stored in array $LUT_\tau[idx]$. The ith

element of $LUT_\tau[]$ gives the amount of slacks p_τ receiving when the total available slack is i. Let T denote the minimum time steps in slack distribution. The maximum length of each LUT array is bounded by $\text{deadline} \cdot S_{max}/\mathcal{T}$. The worst case complexity of Algorithm 8 is $O(N \cdot \text{deadline} \cdot \frac{S_{max}}{\mathcal{T}})$.

Algorithm 4.5 gives the SR_DVFS algorithm. It is executed during runtime. Before executing a task τ, the SR_DVFS algorithm calculates the available slack (step 1), then it uses the look-up-table to determine how many slacks should be distributed to the current PE where task τ is mapped to (Step 2). The execution speed (i.e. s_τ) is calculated in Step 3 of the algorithm. We then check the longest path to see if this solution violates the deadline constraint (Step 4). If the answer is false, the algorithm returns, otherwise we recalculate the available slack by considering only the longest path (Step 5) and repeat the previous steps. Note that Algorithm 4.5 has a constant complexity; however, it must be executed each time before a task is executed.

Algorithm 4.5: SR_DVFS (τ, t): τ -- current task id, t -- current time

Calculate the available slack t_{sl} = deadline – t - $ARET_\tau$;

1. The slack distributed to p_τ is $t_{sl}(\tau) = LUT(t_{sl})$;
2. $s_\tau = (t_{sl}(\tau) + ARET\_PE_\tau)/ ARET\_PE_\tau$;
3. **if** ($WCET(\tau, p_\tau) * (s_\tau - 1) + MxRET_\tau + t >$ deadline) {
4. t_{sl} = deadline – t – $MxRET_\tau$;
5. $t_{sl}(\tau) = LUT(t_{sl})$;
6. $s_\tau = (t_{sl}(\tau) + ARET\_PE_\tau)/ ARET\_PE_\tau$;
7. }

Algorithm 4.5 SR_DVFS Algorithm

If the application consists of single path across different PEs, then the SR_DVFS gives the optimal solution. However, parallel execution paths usually exist in a multi-core system and there are synchronization events among these execution paths. Therefore, the SR_DVFS is only

a greedy heuristic. Our experimental results show that compared to the NLP based DVFS, the system using SR_DVFS algorithm consumes 4.1% more energy in average.

The effectiveness of the SR_DVFS relies on two important conditions: 1. the power consumption of a PE must be a convex function of its clock period; 2. the PEs must support continuous voltage and frequency scaling. The first condition can be satisfied by most existing microprocessors with support of DVFS. The second assumption is not quite realistic, because many real life microprocessors only support discrete DVFS. However, it has been pointed out in [100] that using intra-task DVFS, we can approximate any voltage and frequency setting using 2 discrete voltage and frequency levels. Even if the intra-task DVFS is not available, the SR-DVFS algorithm works fine under discrete voltage-frequency because it reclaims the slack that cannot be utilized by previous tasks due to the voltage (frequency) round up. Note that the NLP based approach cannot handle discrete voltage (frequency) level due to the high complexity. As we will show in Section VI, compared to a DVFS choice that simply rounds up the voltage and frequency solution of the NLP to the nearest level, the slack reclaiming algorithm reduces the energy by 41.37%.

4.5 Experimental Results

We assume that there are three different kinds of PEs in the system. They are XScale 80500, XScale PXA255 and PowerPC processors. These processors operate at different voltage and frequency, and they have different power/performance characteristics. We obtained their cycle energies under different voltage and frequencies from [64], [65] and [66] respectively. We use curve fitting to approximate the cycle energy as a continuous function of the cycle period and use this model to predict the cycle energy for any cycle period which is between the maximum

and minimum supported frequency. We summarize the processors' parameters in Table 4.1 and plot the cycle energy curves in Figure 4.2.

To account for different graph structures and complexities which resemble numerous real applications, we carried experiments on CTGs modified from random task graphs generated by TGFF [63]. The MPSoC consists of either 3 PEs (one of each aforementioned mentioned type) or 4 PEs (with an additional Xscale 80500 processor). Table 8 gives the summary of some statistics of the 6 test cases including the number of tasks, the number of PEs, and the number of branch fork nodes in the CTG. The first row of the table gives the ID of each test case, which will be used in the rest of the section.

Type	Max_V (Volt)	Max_f (MHz)	Min_V (Volt)	Min_f (MHz)
PPC	1.9	333	1	33
Xscale	1.49	733	0.91	333
PXA255	1.3	400	0.85	100

Table 4.1 Processor parameters

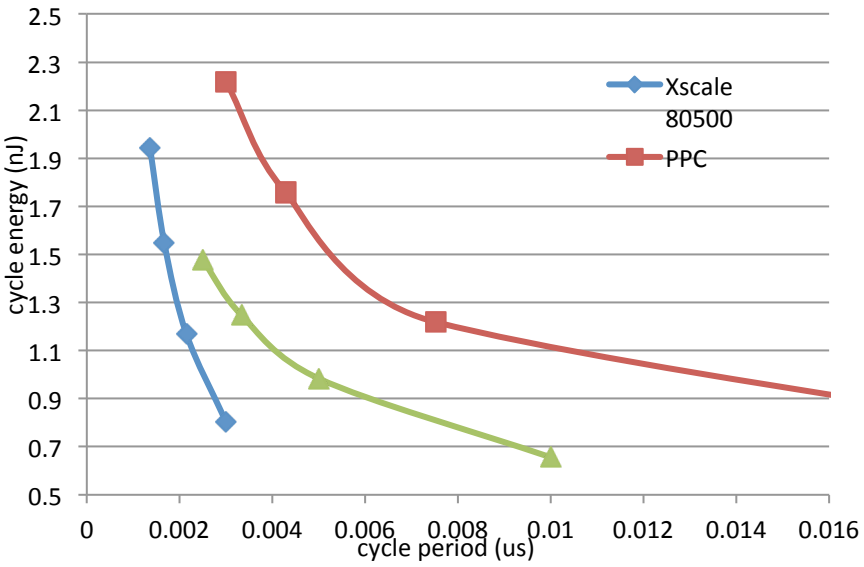


Figure 4.2 Processor Cycle Energy Curves

In addition to these 6 cases, and in order to test the scalability of the proposed algorithms and its performance on large applications and systems, we also construct a large CTG with 86 tasks and allocate them on a 16-PE MPSoC system. We call the test case CTG-large.

Furthermore, we apply our algorithm on a real-world application, i.e. the MPEG decoder. The MPEG decoding process keeps varying according to the contents of the visual scene. Each video frame in the encoded video stream is composed of various macroblocks that represents 16x16 pixel area of the image. The macroblocks are further classified into I, P and B blocks. Different types of macroblock require different decoding procedure. For example, the I-blocks will perform the IDCT function while the B-blocks may skip the IDCT. The CTG for MPEG decoding process have 46 tasks. Among these tasks, 9 of them are branch nodes. Due to space limitations we cannot show the whole graph of the decoding process. We map the MPEG decoding CTG to a system consists of 6 PEs. We call the test case CTG-mpeg.

Test Case ID	0	1	2	3	4	5
Number of nodes in CTG	12	25	16	15	15	25
Number of PEs	3	3	3	4	4	4
Number of branch fork nodes	1	3	1	2	1	3
Probability of each branch fork nodes	(0.9,0.1)	(0.9,0.1) (0.9,0.1) (0.9,0.1)	(0.9,0.1)	(0.9,0.1) (0.9,0.1)	(0.8,0.1,0.1)	(0.8,0.2) (0.8,0.2) (0.8,0.2)

Table 4.2 Summary of generated test cases

We define the system utilization as $U = \sum_{\tau_i \in V} WCET(\tau_i)/(D \cdot N)$ where D is the application deadline and N is the number of PEs. Obviously, the effectiveness of the low power scheduling algorithm should change as the system utilization varies.

We refer to our algorithm as Balanced Energy Slack Scheduling (BESS) as it considers both the energy and performance in a heterogeneous system. For each CTG, the algorithm

generates a task mapping, a schedule and a voltage/frequency selection. We call the combination a Solution.

Sometime, the mapping between tasks and PEs are predetermined. We can extend BESS to find the task execution order for the CTGs with fixed mapping by considering only those given task-PE pairs in DL calculation. We refer to this degenerated version of BESS as BESS-FM (BESS with Fixed Mapping). For comparison purpose, we implemented NLP based DVFS similar to the algorithm described in [90]. By default, the BESS algorithm uses NLP based DVFS. We use BESS-SR to refer to the BESS algorithm utilizing the slack reclamation based DVFS.

One of the major questions of BESS is how to determine the value of λ in Equation (4.5). When λ is 0, the algorithm degenerates into the minimum average make span scheduling; and when λ is very large, the algorithm maps a task to the minimum energy processor without considering the slacks for the DVFS algorithm. Strictly speaking, the optimum value of λ is a function of many parameters, including the topology of the CTG, the branch probability, the performance and power characteristics of the multi-core system, and the system utilization, etc. For those 8 test cases, we vary their branch probability and deadline (which consequently affects the system utilization) and generate a large number of different scenarios. For each scenario, we sweep λ to find its optimal value. We found that (1) the best value of λ is always within a small range for majority of the scenarios; (2) some values of λ consistently outperform other values for all system utilization and branch probability settings, and the difference between these local maximum points is not significant.

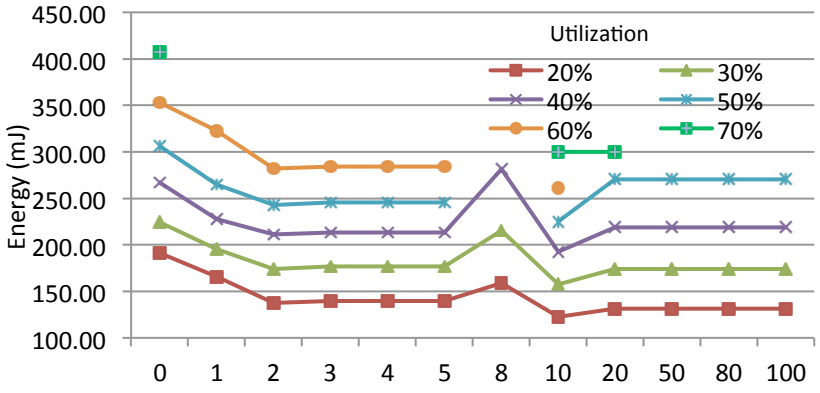


Figure 4.3 The Effect of λ on Energy Consumption

As an example, Figure 4.3 shows the energy as a function of λ for CTG0 under different system utilization from 20% to 70%. Please note that not all λ have feasible solutions for all system utilizations, for example, there is feasible schedule when the system utilization is at 70% and λ is -between 1 and 8. As shown in the figure, most of the energy curves have two local minima. The first one occurs at $\lambda \in [2, 5]$ while the second occurs at $\lambda=10$. The first local maximum has slightly higher energy than the second local maximum. However, overall, both local minima give much lower energy than other data points in the energy curve. Similar trend is observed for other test cases. Based on this observation, we use a fixed λ for each CTG in our experiments. It is set to be the best value that minimizes the energy of the CTG at 50% utilization and equal branch selection probability. As we will show later, the task mapping obtained by BESS algorithm outperforms the best random mapping in 10,000 samples. Given the limited range of the optimal value of λ, searching for the best λ is much easier than searching for the best mapping directly.

4.5.1 Comparison with Random Mapping

In the first experiment, we evaluate the quality of the task mapping function in BESS. For each test case, large numbers of random mappings are generated and scheduled using BESS-FM. The best result is compared to the result found by the original BESS, which optimizes both task mapping, ordering and DVFS.

Table 4.3 compares the system energy dissipation of running the solution generated by BESS, the best results and average results of 10,000 random mappings scheduled using BESS-FM and the improvement of BESS over the best and average random mapping. As we can see, even with 10,000 random samples, we cannot find a task mapping that is as energy efficient as the one found by BESS.

CTG#	BESS (mJ)	Best of 10,000 Random Mapping (mJ)		Average of 10,000 Random Mapping (mJ)	
		BESS-FM	Impr.	BESS-FM	Impr.
CTG0	157.38	163.97	4.02%	291.54	46.02%
CTG1	101.01	120.88	16.44%	158.72	36.36%
CTG2	196.99	378.24	47.92%	763.07	74.18%
CTG3	298.25	320.77	7.02%	506.21	41.08%
CTG4	512.08	558.99	8.39%	829.40	38.26%
CTG5	918.38	1042.85	11.94%	1250.61	26.57%
CTG-large	1114.44	2604.5	57.21%	3686.68	69.77%
CTG-mpeg	92.99	110.29	15.69%	162.92	42.92%

Table 4.3 Comparison to Random Mapping

4.5.2 Energy Consumption under Different System Utilization

In the following experiments, we will demonstrate the necessity of considering delay and energy jointly during every step of the resource management algorithm, and will compare the solution of our algorithm with four other scheduling algorithms: the Minimum Make Span (MMS), the Minimum Energy (MNRG), the Earliest Start Time (EST), and one of the previous works in reference [79]. These 4 algorithms are very similar to BESS. The only deviation is how the dynamic level (i.e. DL) is calculated. The MMS algorithm uses the original WCET instead of

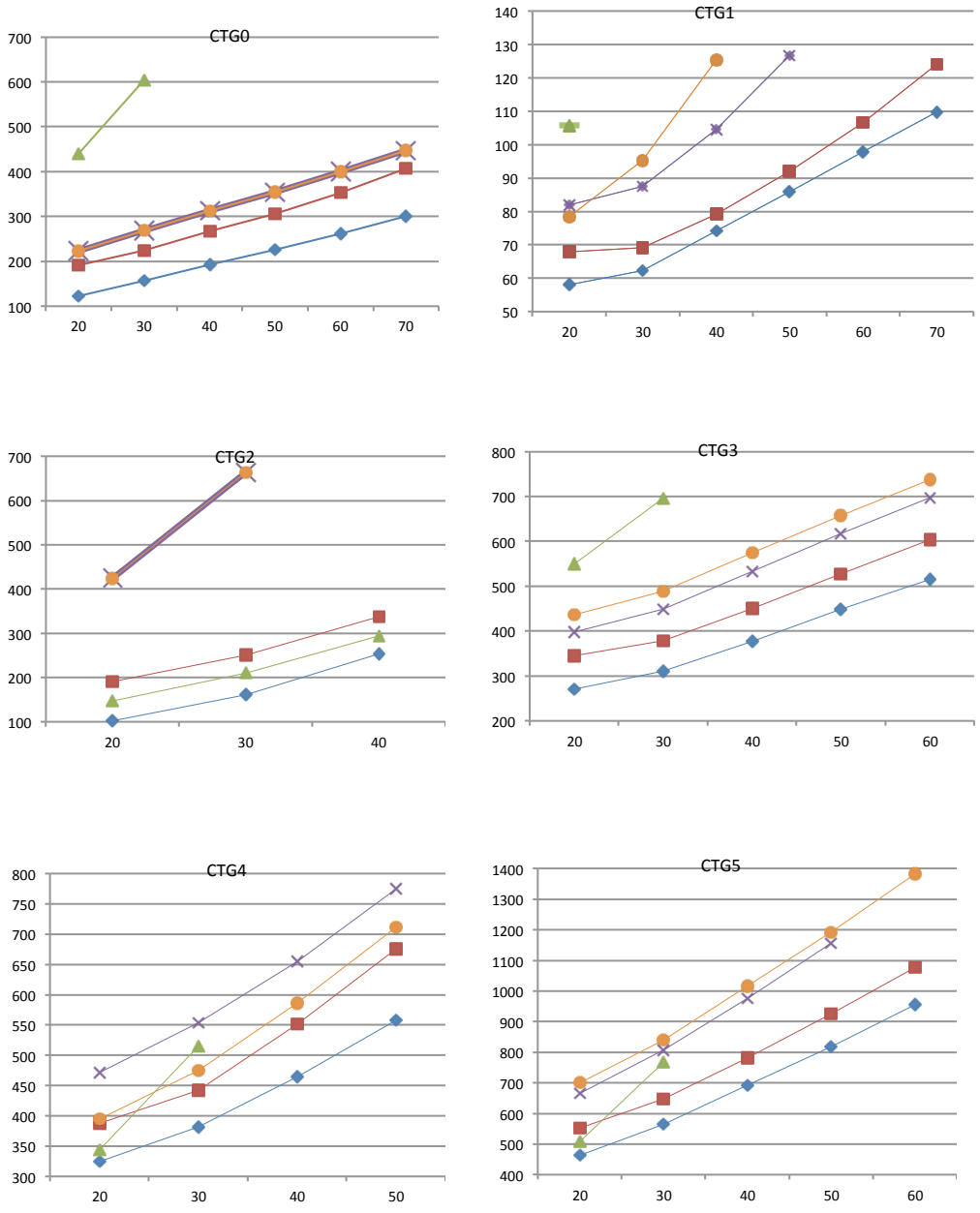
the effective WCET that converts energy heterogeneity to performance heterogeneity. Therefore, it maps tasks to minimize the average make span as stated in Section 4.3.1. Please note that the MMS algorithm is equivalent to the CAP algorithm [91][92]. The MNRG algorithm calculates the dynamic level as: $\mathrm{DL}(\tau_i, p_j) = -\mathrm{WCET}(\tau_i, p_j) \times \mathrm{E}_{\mathrm{total}}(p_j, f_{max})$; therefore, it maps a task to the PE that is the most energy efficient for this task. Finally the EST algorithm calculates the dynamic level as: $\mathrm{DL}(\tau_i, p_j) = -\mathrm{AT}(\tau_i, p_j)$, therefore, it maps tasks to where they can start the earliest.

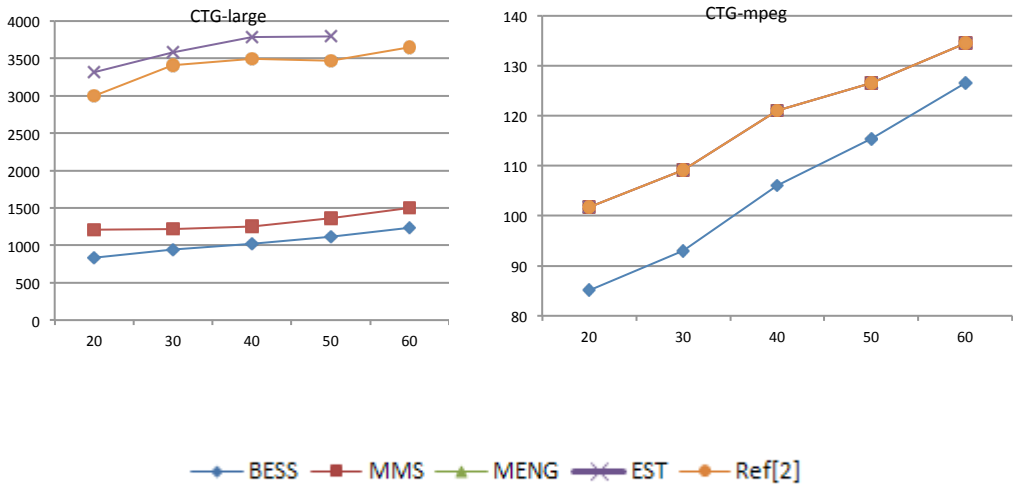
To the best of our knowledge, there is no existing work that considers the exact same problem as this work (i.e. task mapping, scheduling and DVFS for CTG in a heterogeneous multiprocessor platform.) In order to compare with existing work, we modified the algorithm proposed by Zhang et. al. in [79] to consider conditional branches. Their algorithm is modified so that the slack is replaced by average slack and mutually exclusive tasks can share the same PE at the same time.

Figure 4.4 shows the energy consumption of the 8 test cases under those 5 scheduling algorithms. In those figures X-axis represents the percentage system utilization and Y-axis represents the energy consumption. Please note that not all scheduling algorithms have feasible solutions for all system utilizations. As we expected, when the system utilization increases, the energy consumptions also increase because tasks have to run at higher speed to guarantee the deadline.

As shown in the figure, in general, MNRG is ineffective as it cannot find feasible solution for most of the test cases. The BESS algorithm is able to find better solutions than other four algorithms in all cases because it is not only able to leave enough slack for DVFS but also

able to explore the energy heterogeneity. Because [79] approach does not consider power/performance heterogeneity, it could not achieve the same energy savings as BESS. Overall, comparing to BESS, MMS, MNRG, EST and [79] consume 25.3%, 85.7%, 69.8% and 70.3% more energy for test case 0-5 respectively.



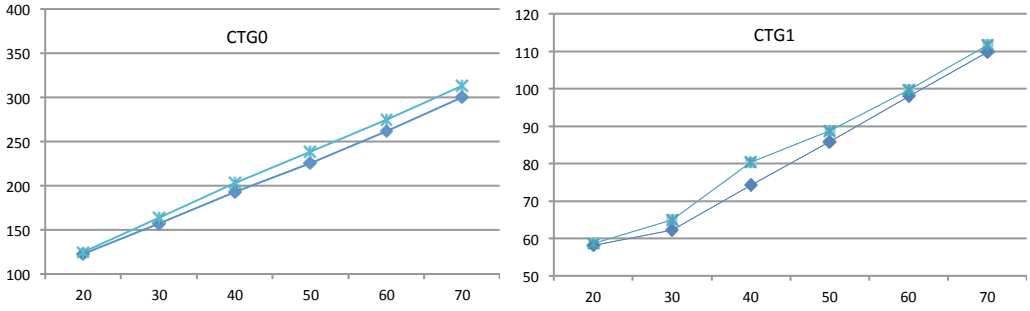


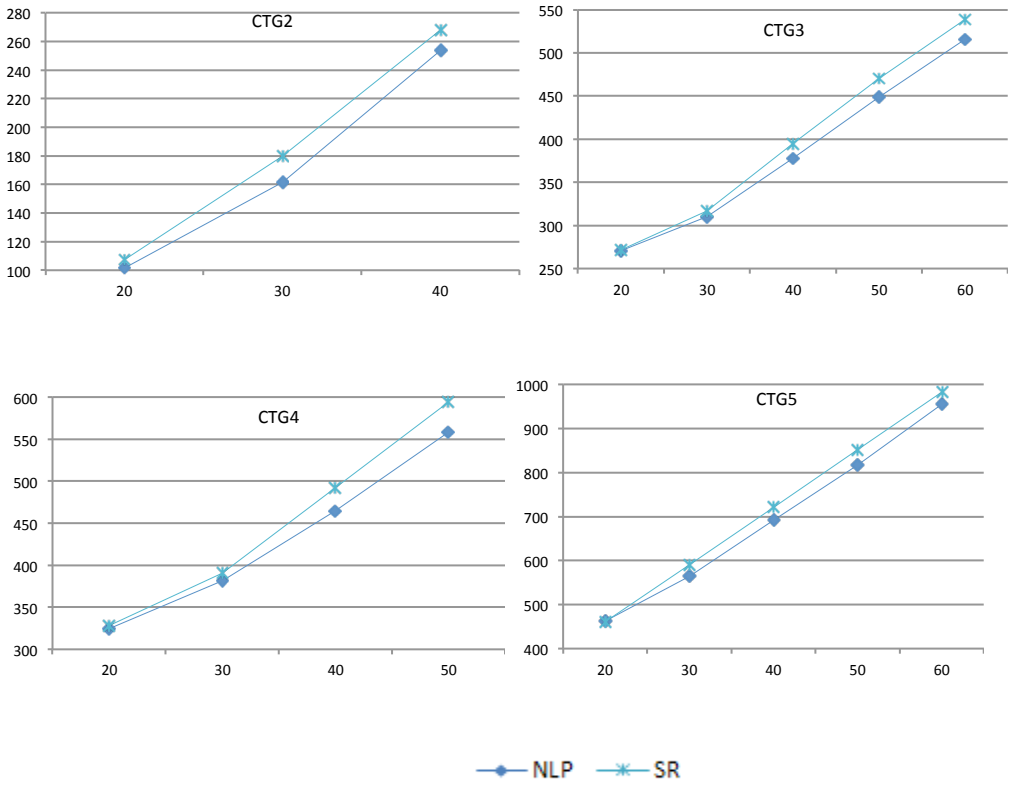
x-axis: utilization, y-axis: energy consumption (mJ)

Figure 4.4 Energy Consumption under Different System Utilization

Please note that because the search spaces of test case CTG-large and CTG-mpeg are too large, the NLP based DVFS is not able to find a solution within a reasonable time, while our SR algorithm only takes several seconds. Therefore, we integrate the SR based DVFS to all the mapping and scheduling algorithms in CTG-large and CTG-mpeg test case. As shown in this figure, in CTG-large case, BESS algorithm outperforms MMS, EST and [79] by 21.6%, 73.0%, 69.9%. In CTG-mpeg case, MMS, EST and [79] algorithm have very similar energy consumption, while the BESS algorithm outperforms them by 11.64%.

4.5.3 The Effectiveness of Slack Reclaiming Algorithm





x-axis utilization, y-axis energy consumption (mJ)

Figure 4.5 Comparison of NLP and SR Based DVFS

In order to evaluate the performance of our slack reclamation (SR) based DVFS heuristic, we developed a fast performance evaluation framework based on OMNeT++ [105]. OMNeT++ is a discrete event simulation environment, and its simulation kernel is written in C++. Users can define the behavior of their own components in the network and describe the network topology and communication overhead using a high level language provided by the OMNeT++.

We again vary the system utilization from 20% to 70% and compare the quality of DVFS solutions found by the SR and NLP algorithm. The results are shown in Figure 4.5. The same task mapping and ordering found by BESS are used for both DVFS algorithms.

As shown in these figures, the SR algorithm tracks the NPL based DVFS algorithm very well and gives close energy consumptions. In average, systems using SR based DVFS method only consume 4.1% more energy than the system using NLP based DVFS.

CTGs	Continuous DVFS (mJ)		Discrete DVFS (mJ)	
	NLP	SR	NLP	SR
CTG 0	192.64	202.66	415.29	218.92
CTG 1	73.19	80.34	183.86	92.82
CTG 2	253.59	268.27	321.29	270.44
CTG 3	377.38	394.75	614.2	418.48
CTG 4	464.83	492.13	859.52	491.86
CTG 5	697.23	721.79	1985.3	775.60

Table 4.4 Comparison of Continuous and Discrete DVFS

Table 4.4 shows the comparison between the continuous and discrete DVFS. As shown in the table, the performance of NLP based DVFS is severely degraded when only discrete voltage and frequency level is supported by the system, while the SR algorithm dynamically adjusts the slowdown ratio based on current remaining slack. On average, the discretized slack reclaiming algorithm reduces the energy dissipation by 41.37% comparing to discretized NLP algorithm. Please note that for test case CTG-large CTG-mpeg, the only feasible DVFS algorithm is Slack Reclaiming based algorithm, thus in this subsection, we do not have results for them.

In the next experiment we restrict the PE's voltage and frequency to those discrete levels specified in Table 4.1. With NLP based DVFS, in order to guarantee deadline, the voltage and frequency are always round up to the nearest level. With SR based DVFS, the extra slack generated by rounding up of current task will be reclaimed by future tasks

4.5.4 Sensitivity to Branch Probability Change

In previous experiments, we assume that the branch probabilities are known and fixed during run time. In the next set of experiments, we examine the sensitivity of BESS to the change of branch probability. The results will help us to understand the performance of BESS under inaccurate software model or insufficient profiling information.

The solution of BESS consists of 3 parts: task mapping, task ordering and voltage/frequency selection. When the branch probability of an application changes from A to B during the runtime, we have 4 options: (1) Continue using the initial scheduling found based on branch probability A. This option will be referred as "No update" as it updates nothing. (2) Continue using the original task mapping and ordering, but update the voltage and frequency selection using the new branch probability B. This option will be referred as "DVFS only". (3) Continue using the original task mapping, but update the task ordering and voltage/frequency selection. It will be referred as "DVFS+SCH". (4) Update everything in solution including task mapping, ordering and voltage/frequency selection. We refer to this option as "All update". In the rest of the chapter, we refer to the original branch probability A as the "biased branch probability" and the updated branch probability B as the "actual branch probability". We refer to the first 3 scheduling options that update none or only part of the solution as biased scheduling and the last scheduling option as the unbiased scheduling.

We use CTG0 to illustrate the impact of branch probability on the quality of solution. We set the actual branch probability B to (0.1, 0.9). Figure 17 shows the energy dissipation of the system under 3 biased scheduling algorithms with the biased probability A varying from (0.1, 0.9) to (0.9, 0.1). All 3 curves give the same best result when the biased probability A equals to the

actual branch probability B, which stands for the unbiased scheduling option (i.e. "ALL") that updates everything in the solution.

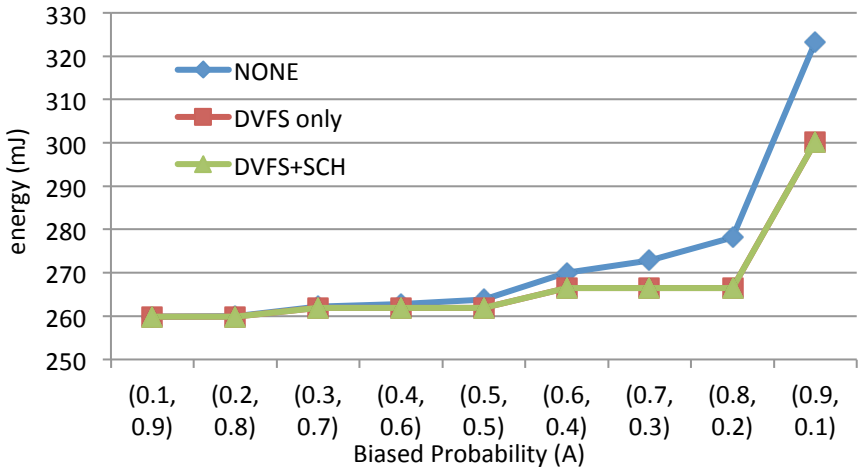


Figure 4.6 Energy Consumption under Inaccurate Branch Probability

From this figure, we can see that the more accurate probability information we use in scheduling, the more energy savings we can achieve. We also observe that when the biased probability is close to the actual probability, the performance of the biased scheduling and the unbiased scheduling are close to each other. For example, when the biased probability goes from (0.1, 0.9) to (0.6, 0.4), the energy difference between the biased and the unbiased solution is less than 4%. On the other hand, when the biased probability is far from the real probability, the energy difference between the biased and the unbiased solution can be as large as 25%. The above observation implies that if the branch probability changes during run time, as long as the difference between the actual probability and the biased probability is small, the energy consumption will not significantly deviate from the optimum value. Therefore, we do not need to rerun the entire scheduling process (which can be very time consuming for systems with large number of tasks and PEs) to adapt the probability change.

Bias Distance	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6
DVFS+SCH	0.14%	0.30%	0.69%	1.12%	1.71%	2.33%	3.16%	5.12%
DVFS only	0.22%	0.34%	0.74%	1.19%	1.79%	2.44%	3.32%	5.12%
No update	0.27%	0.54%	1.06%	1.72%	2.57%	3.54%	4.85%	7.47%

Table 4.5 Solution Quality Degradation as the Bias Distance Increases

The similar trend is found for other CTGs. First, we define the Bias Distance to quantify the difference between the actual branch probability and the biased probability. Assume a CTG has n conditional edges and the actual branch probabilities for these branches are $B = (b_1, b_2, \dots, b_n)$, the biased branch probabilities are $A = (a_1, a_2, \dots, a_n)$. Then the distance between A and B is defined as the L1-norm of the vector $\mathrm{A} - \mathrm{B}$, i.e. $\mathrm{D}(\mathrm{A}, \mathrm{B}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} |\mathrm{a_i} - \mathrm{b_i}|$. A large bias distance means that the biased probability used for optimization is far from the actual probability during runtime, which, intuitively, could induce low quality solution. This trend is shown in Table 4.1. For each CTGs, we select one branch fork node and vary its real branch probabilities to generate test cases whose biased distance ranging from 0.2 to 1.6. For each test case, the energy dissipation of the system under 3 biased scheduling algorithms is recorded. Table 4.1 gives the percentage energy increase of biased solutions over the unbiased solution. As expected, when the bias distance increase, the solution quality degrades and energy consumption increases. Furthermore, performing "DVFS only" and "DVFS+SCH" give very similar results in terms of energy savings. This indicates that without correct task mapping, updating the task ordering does not make much difference.

4.6 Conclusion

In this chapter, we proposed a framework for simultaneous task mapping and ordering followed by DVFS of control intensive real-time applications modeled as probabilistic

conditional task graph. The goal of our scheduling algorithm is to minimize the mathematical expectation of the energy by utilizing the branch selection probability. The proposed mapping and scheduling algorithm balances the energy and performance of tasks running on a heterogeneous multi-core platform. The proposed slack reclaiming DVFS algorithm effectively distributes the slack based on current branch selection at run time and achieves similar energy reduction as mathematical programming based DVFS with much lower complexity.

We compared our proposed mapping and ordering algorithm with the minimum makespan, the minimum energy, the earliest start time and the mapping and ordering algorithm proposed in [79] on 8 CTGs, and the proposed algorithm can reduce 25.3%, 85.7%, 69.8% and 70.3% energy consumption respectively. The proposed Slack Reclaiming DVFS algorithm could find good solution which only consumes 4% more energy compared to the solution found by Nonlinear Mathematical Programming based DVFS when the operating voltage can be adjusted continuously. When the operating voltage is discrete, our Slack Reclaiming DVFS algorithm reduces the energy dissipation by 41.37% comparing to discretized NLP algorithm.

Chapter 5 Conclusions and Future Directions

The battery life limits the customer's experience while using the portable devices, and the life of battery limits the life of portable devices. Energy harvesting systems have been adopted as a clean and semi-permanent power source for some portable devices in order to extend the life of portable devices and improve the customer's experience. However, due to the instability of the harvesting energy, the electrical energy storage has been used to store extra energy when the harvesting rate is high and supports the devices with stored energy when the harvesting rate is low. The popularity of these types of devices in commercial market and in research community motivates us to propose several techniques to improve energy efficiency on the energy harvesting embedded system.

5.1 Conclusion

First, we investigate the effects of workload scheduling on the efficiency of the EES charge process in an energy harvesting embedded system. Our analytic algorithm always charges the EES bank using the lower power first scheme at fixed harvesting rate and monotonically decreasing harvesting rate. We found that high power workload first scheme always performs better than the low power workload first scheme. The algorithm has been proved by using the approximated but accurate power model of the DC-DC converter. We proposed a neural network based prediction model to predict the energy conserved in the EES bank after one charge phase. It is adopted in the workload scheduling algorithm to solve the

cases that do not fit the HPWF scheme. Experimental results show that the HPWF outperforms the LPWF by up to 24.61% for two charging phases. For multiple tasks, our workload scheduling outperforms LPWF by up to 12.27% and outperforms the random scheduling scheme by up to 5.88%. Then, we investigate the effects of DVFS on the efficiency of supercapacitor bank charging process in an energy harvesting embedded system. Our algorithm used the dynamic programing based DVFS policy based on the prediction of the stored energy in the EES bank. For prediction, we adopted the proposed neural network based prediction model which could accurately estimate bank energy given input power, initial bank energy and charging time. Based on this prediction model, we propose a dynamic programming based DVFS algorithm which performs robustly regardless the variation of bank SOC, task load power consumption and schedule deadline compare to a baseline algorithm. Results show that up to 80% more harvesting energy could be stored into the EES bank.

Secondly, as realized the DC-DC converter is playing an important role on the energy efficiency in the energy harvesting embedded system, we proposed an efficient heuristic algorithm for the energy efficiency optimization problem on the DC-DC converter for charge allocation and replacement in an EHS/HEES equipped embedded system. In order to avoid the complexity of iteratively searching the optimal operating point, we adopted the proposed approximation power model of the DC-DC converter in the framework. Our goal is to minimize the energy wasted on the DC-DC converters while maximizing the energy stored in the HEES system during the charge transfer. The heuristic algorithm is divided into the charging part and discharging part, which tries to match the input and output voltages of the DC-DC converters by dynamically adjusting the V_{dd} of the embedded load, the charge transfer interconnect voltage V_{cti} and the storage bank terminal voltage using reconfiguration. Experimental results show that our

algorithm could store up to 124.05% more energy during charging process and last up to 140.22% longer.

Finally, in order to reduce the power consumption when the harvesting rate is short and the EES banks support the load devices alone, we proposed a framework for simultaneous task mapping and ordering followed by DVFS of control intensive real-time applications modeled as probabilistic conditional task graph. The goal of our scheduling algorithm is to minimize the mathematical expectation of the energy by utilizing the branch selection probability. The proposed mapping and scheduling algorithm balances the energy and performance of tasks running on a heterogeneous multi-core platform. The proposed slack reclaiming DVFS algorithm effectively distributes the slack based on current branch selection at run time and achieves similar energy reduction as mathematical programming based DVFS with much lower complexity. The proposed algorithm can reduce 25.3%, 85.7%, 69.8% and 70.3% energy consumption respectively compared to the baseline algorithms. The proposed Slack Reclaiming DVFS algorithm could find good solution which only consumes 4% more energy compared to the solution found by Nonlinear Mathematical Programming based DVFS when the operating voltage can be adjusted continuously. When the operating voltage is discrete, our Slack Reclaiming DVFS algorithm reduces the energy dissipation by 41.37% comparing to discretized NLP algorithm.

5.2 Future Directions

In the future, according to different demands on energy and different characteristics of power consumption, if it is a challenge to design an energy harvesting system and energy storage system for a specific device. Some problems are as following:

- Which type of EES bank is best to be a main storage or a secondary storage bank? It is important to choose among different EES elements for constructing a HEES system. For example, in the smart building, lead acid battery may be used as the main storage due to its low cost, and li-ion battery may be the secondary storage to reserve the energy for a long time because of its low self-leakage. [67] However, for a mobile device, lead acid battery is too large to pack onto it. Under this situation, while li-ion battery is using a main energy storage, supercapacitors can be adopted as the backup power source or charging buffer.
- Deciding the size of each EES elements is also application dependent. If the HEES system is applied on a mobile embedded system, it should be large enough to support the system and small enough to be carried on. [68] tries to solve this problem on the electric vehicles. The vehicle has the limitation on its own size, and it is not energy efficient if it costs more energy to carry the EES bank itself. The embedded system faces the similar problem.
- With the rise of requirements, it is necessary to charge and discharge several banks simultaneously. Hence, how to do the charge management under this scenario attracts researcher's attention. It is one solution if we use several charge transfer interconnects. Another solution is to build a router into the converter to distribute the energy. This idea is from the routers in the FPGA distributing the tasks [69].

It is impossible to design a commercial energy harvesting system or an energy storage system without referring to specification of the load applications and considering the energy efficient as well as the capital cost. We believe that research in these areas not only enhance the

user experience with embedded devices but also help conserve the resource and improve the living environment.

Bibliography

[1] http://mspoweruser.com/idc-survey-finds-windows-phone-buyers-chose-their-phones-based-on-os-ahead-of-ios-or-android/

[2] http://mars.nasa.gov/mer/home/

[3] http://www.wired.com/2014/09/facebook-drones-2/

[4] http://www.tecategroup.com/ultracapacitors-supercapacitors/custom-modules.php

[5] http://www.tecategroup.com/capacitors/datasheets/powerburst/PBD.pdf

[6] OMNeT++: http://www.omnetpp.org/

[7] Nathan Jacobson, Basic algebra, vol. 1 (2nd ed.), Dover, 2009.

[8] Q Xie, D Zhu, Y Wang, M Pedram, Y Kim, N Chang, "An efficient scheduling algorithm for multiple charge migration tasks in hybrid electrical energy storage systems" Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific

[9] M. Pedram, N. Chang, Y. Kim and Y. Wang, "Hybrid electrical energy storage systems," in Proc. of International Symposium on Low Power Electronics and Design, Aug. 2010, pp. 363-368.

[10] Q. Xie, Y. Wang, M. Pedram. Y. Kim, D. Shin, and N. Chang, "Charge replacement in hybrid electrical energy storage systems," Proc. of Asia and South Pacific Design Automation Conf., Jan. 2012.

[11] M. Pedram. "Charge allocation for hybrid electrical energy storage systems," Proc. of International Conf. on Hardware/Software Codesign and System Synthesis, Oct. 2011.

[12] Q. Xie, Y. Wang, Y. Kim, N. Chang and M. Pedram, "Charge allocation for hybrid electrical energy storage systems," in Proc. of the International Conference on Hardware/Software Codesign and System Synthesis, Oct. 2011.

[13] Y. Wang, Y. Kim, Q. Xie, N. Chang, and M. Pedram. “Charge migration efficiency optimization in hybrid electrical energy storage (HEES) systems,” Proc. of Int'l Symp. on Low Power Electronics and Design, Aug. 2011

[14] W. Lee, Y. Wang, D. Shin, N. Chang, and M. Pedram. “Power Conversion Efficiency Characterization and Optimization for Smartphones,” Proc. of Int'l Symp. on Low Power Electronics and Design, Jul. 2012.

[15] Y. Choi, N. Chang and T. Kim, "DC–DC Converter-Aware Power Management for Low-Power Embedded Systems," in IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol.26, no.8, pp. 1367-1381, Aug. 2007.

[16] Y. H. Lu, L. Benini, and G. De Micheli, “Low-power task scheduling for multiple devices,” Proc. of International Workshop on HW/SW CoDesign, 2000.

[17] S. Liu, Q. Qiu, and Q. Wu, “Task merging for dynamic power management of cyclic applications in real-time multi-processor systems,” in ICCD, 2006.

[18] Y. Wang, Q. Xie, A. Ammari, and M. Pedram, “Deriving a near-optimal power management policy using model-free reinforcement learning and Bayesian classification,” in DAC, 2011.

[19] F. Yao, A. Demers, and S. Shenker, “A scheduling model for reduced CPU energy,” IEEE Foundations of Computer Science, 1995.

[20] T. Ishihara and H.Yasurra, “Voltage scheduling problem for dynamically variable voltage processors,” Proc. Of International Symposium on Low Power Electronics Design, August, 1998.

[21] G. Quan and X. S. Hu, “Minimum energy fixed-priority scheduling for variable voltage processors,” in DATE, 2002.

[22] X. Lin, Y. Wang, S. Yue, N. Chang, and M. Pedram. “A Framework of Concurrent Task Scheduling and Dynamic Voltage and Frequency Scaling in Real-Time Embedded Systems with Energy Harvesting,” Proc. of Int'l Symp. on Low Power Electronics and Design, Sep. 2013.

[23] J. Luo and N. K. Jha, “Static and Dynamic Variable Voltage Scheduling Algorithms for Real-time Heterogeneous Distributed Embedded Systems,” in Proc. International Conference on VLSI Design, Jan. 2002, pp. 719-726.

[24] Y. Liu, B. Veeravalli and S. Viswanathan, “Novel Critical-Path based Low-Energy Scheduling Algorithms for Heterogeneous Multiprocessor Real-Time Embedded Systems,” Proc. of the 13th International Conference on Parallel and Distributed Systems, 2007.

[25] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, “Energy-optimal Software Partitioning in Heterogeneous Multiprocessor Embedded Systems,” Proc. Design Automation Conference, 2008.

[26] C. Y. Yang, J. J. Chen, T.W. Kuo, and Lothar Thiele, “An Approximation Scheme for Energy-Efficient Scheduling of Real-Time Tasks in Heterogeneous Multiprocessor Systems,” Proc. of Design, Automation & Test in Europe Conference & Exhibition, 2009.

[27] J. Goossens, D. Milojevic, and V. Nelis, “Power-Aware Real-Time Scheduling upon Dual CPU type Multiprocessor Platforms,” Proc. of the 12th International Conference on Principles of Distributed Systems, 2008.

[28] Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, A. Nicolau, “Profile-based Dynamic Voltage Scheduling using Program Checkpoints,” in Proc. of DATE, 2002.

[29] P. Yang, F. Catthoor, “Pareto-optimization-based run-time task scheduling for embedded systems,” in Proc. of CODES+ISSS, 2003.

[30] S. Liu, Q. Qiu, Q. Wu, “Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting”, in Proc. of Design Automation and Test in Europe, Mar. 2008.

[31] S. Liu, Q. Wu and Qinru Qiu, “An Adaptive Scheduling and Voltage/Frequency SelectionAlgorithm for Real-time Energy Harvesting Systems”, in Proc. Of Design Automation Conference, Jul. 2009.

[32] S. Liu, J. Lu, Q. Wu, and Q. Qiu, "Harvesting-aware power management for real-time systems with renewable energy," IEEE TVLSI, 2012.

[33] P. Stanley-Marbell and D. Marculescu, "Dynamic fault-tolerance and metrics for battery powered, failure-prone systems," in Proc. of Int'l Conference on Computer Aided Design, Nov. 2003.

[34] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman and Mani Srivastava, “Design considerations for solar energy harvesting wireless embedded systems,” In Proc. of the 4th intern. Symp. on Information processing in sensor networks (IPSN '05), 2005.

[35] Y. Wang, X. Lin, Y. Kim, N. Chang, and M. Pedram, "Enhancing Efficiency and Robustness of a Photovoltaic Power System Under Partial Shading," in Proc. of the 13th Int'l Symposium on Quality of Electronic Design, Mar. 2012.

[36] D. Hohm and M. Ropp, "Comparative study of maximum power point tracking algorithms using an experimental, programmable, maximum power point tracking test bed," in IEEE PSC, 2000.

[37] T. Esram, J. Kimball, P. Krein, P. Chapman, and P. Midya, "Dynamic maximum power point tracking of photovoltaic arrays using ripple correlation control," IEEE T. on Power Electonics, 2006.

[38] F. Liu, S. Duan, F. Liu, B. Liu, and Y. Kang, “Avariable step size INC MPPT method for PV systems,”IEEE T. on Industrial Electron., 2008.

[39] Y. Kim, N. Chang, Y. Wang, and M. Pedram, "Maximum power transfer tracking for a photovoltaic-supercapacitor energy system," in Proc. of Symposium on Low Power Electronics and Design, Aug. 2010, pp. 307-312.

[40] Y. Kim, S. Park, Y. Wang, Q. Xie, N. Chang, M. Poncino, and M. Pedram, "Balanced reconfiguration of storage banks in a hybrid electrical energy storage system," Proc. of Int'l Conference on Computer Aided Design, Nov. 2011.

[41] D. Shin, Y. Wang, N. Chang, and M. Pedram, "Battery-supercapacitor hybrid system for high-rate pulsed load applications," in Proc. of Design Automation and Test in Europe, Mar. 2011.

[42] M. D. Seeman, S. R. Sanders, J. M. Rabaey, “An ultra-low-power power management IC for energy-scavenged wireless sensor nodes”, Power Electronics Specialists Conference, 2008. PESC 2008. IEEE

[43] W. Lee, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Optimizing the Power Delivery Network in a Smartphone Platform," in IEEE Trans. Computer-aided Design of Integrated Circuit and System (TCAD), vol.33, no.1, pp.36-49, Jan. 2014.

[44] S. Park, B. Koh, Y. Wang, J. Kim, Y. Kim, Pedram. M and N. Chang, "Maximum Power Transfer Tracking in a Solar USB Charger for Smartphones," in Proc. of IEEE Intern. Symposium on Low Power Electronics and Design (ISLPED), Sep. 2013.

[45] Y. Zhang, Y. Ge and Q. Qiu, "Improving Charging Efficiency with Workload Scheduling in Energy Harvesting Embedded Systems," in Proc. of Design Automation Conference (DAC), May 2013.

[46] X. Lin, Y. Wang, S. Yue, D. Shin, N. Chang and M. Pedram, "Near-optimal, dynamic module reconfiguration in a photovoltaic system to combat partial shading effects." in Proc. of Design Automation Conference, Jun. 2012.

[47] Y. Ge, Q. Wu and Q. Qiu, "A Multi-Agent Framework for Thermal Aware Task Migration in Many-core Systems," in IEEE Trans. on VLSI systems, Vol.8, No. 5, pp. 535-551, Dec. 2012.

[48] Y. Zhang, Y. Ge and Q. Qiu, "Improving Energy Efficiency for Energy Harvesting Embedded Systems," in Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 2013.

[49] Y. Zhang, X. Hu, and D. Z. Chen, "Task Scheduling and Voltage Selection for Energy Minimization," in Proc. Design Automation Conference, Jun. 2002, pp. 183-188.

[50] J. Hu and R. Marculescu, "Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints," in Proc. Conference and Exhibition on Design, Automation and Test in Europe, Feb. 2004, pp. 234-239.

[51] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. "A modular power-aware microsensor with > 1000x dynamic power range," Proc. of Information Processing in Sensor Networks, April 2005.

[52] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. J. Kaiser, "The low power energy aware processing (leap) embedded networked sensor system," Proc. of International Conference on Information Processing in Sensor Networks, 2006.

[53] D. Lymberopoulos, B. Priyantha, and F. Zhao. Mplatform, "A reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes," Proc. of Information Processing in Sensor Networks, 2007.

[54] D. Roberts , R. G. Dreslinski , E. Karl , T. Mudge , D. Sylvester and D. Blaauw, "When Homogeneous becomes Heterogeneous -- Wearout Aware Task Scheduling for Streaming Applications," Proc. of the Workshop on Operationg System Support for Heterogeneous Multicore Architectures, September, 2007.

[55] G.C. Sih and E.A. Lee. "A Compile Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architecture," IEEE Transactions on Parallel and Distributed Systems, Volume 4, Issue 2, Feb. 1993, pp. 175-187.

[56] E. Dolif, M. Lombardi, M. Ruggiero, M. Milano and L. Benini, "Communication-aware Stochastic Scheduling Framework for Conditional Task Graphs in Multi-Processor Systems-on-Chip," in Proc. International Conference on embedded software, Sep. 2007, pp. 47-56.

[57] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli, and P. Pop, "Scheduling of Conditional Process Graphs for the Synthesis of Embedded Systems," in Proc.

Conference and Exhibition on Design, Automation and Test in Europe, pp. 132-139, Feb. 1998.

[58] Y. Xie and W. Wolf, "Allocation and Scheduling of Conditional Task Graph in Hardware/Software Co-synthesis," in Proc. Conference and Exhibition on Design, Automation and Test in Europe, pp. 620-625, Mar. 2001.

[59] D. Wu, B.M. Al-Hashimi and P. Eles, "Scheduling and Mapping of Conditional Task Graph for the Synthesis of Low Power embedded Systems," IEEE Proceedings of Computers and Digital Techniques, Volume 150, Issue 5, Sep. 2003, pp. 262-273.

[60] D. Shin and J. Kim, "Power-Aware Scheduling of Conditional Task Graphs in Real-Time Multiprocessor Systems," in Proc. International Symposium on Low Power Electronics and Design, Aug. 2003, pp. 408-413.

[61] P. Malani, P. Mukre, Q. Qiu and Q. Wu, "Adaptive Scheduling and Voltage Scaling for Multiprocessor Real-time Applications with Non-deterministic Workload," in Proc. Design Automation and Test in Europe, Mar. 2008.

[62] P. Malani, P. Mukre and Q. Qiu, "Power Optimization for Conditional Task Graphs in DVS Enabled Multiprocessor Systems," in Proc. International Conference on VLSI-SoC, Oct. 2007.

[63] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in Proc. International Workshop on Hardware/Software Codesign, Mar. 1998, pp. 15-18.

[64] K. Choi, W-C. Cheng, and M. Pedram "Frame-based dynamic voltage and frequency scaling for an MPEG player." Journal of Low Power Electronics, American Scientific Publishers, Vol. 1, No. 1, Apr. 2005, pp. 27-43.

[65] G. Dhiman and T. Rosing, “Dynamic voltage frequency scaling for multi-tasking systems using online learning”, Proc. of Intern. Symp. on Low Power Electronics Design, Aug. 2007.

[66] J. Lu and Q. Qiu, “Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting”, Proc. of International Green Computing Conference, Jul. 2011.

[67] D. Zhu, S. Yue, Y. Wang, Y. Kim, N. Chang, and M Pedram, “"Designing a Residential Hybrid Electrical Energy Storage System Based on the Energy Buffering Strategy," Proc. of the Int'l Conference on Hardware/Software Codesign and System Synthesis (CODES), Sep. 2013.

[68] D Zhu, S Yue, Y Wang, N Chang, and M Pedram. "Cost-effective design of a hybrid electrical energy storage system for electric vehicles," Proc. of the Int'l Conference on Hardware/Software Codesign and System Synthesis (CODES), Oct. 2014.

[69] Y. Kim, S. Park, N. Chang, Q. Xie, Y. Wang, and M. Pedram, “Networked Architecture for Hybrid Electrical Energy Storage Systems,” Proc. of 49th Design Automation Conf., Jun. 2012.

[70] http://www.popsci.com/environment/article/2009-01/harvesting-energy-humans

[71] http://www.prnewswire.com/news-releases/low-battery-anxiety-grips-9-out-of-ten-people-300271604.html

[72] http://zebu.uoregon.edu/disted/ph162/l11.html

[73] http://www.mpoweruk.com/wind_power.htm

[74] P. Nintanavongsa, “A survey on RF energy harvesting: circuits and protocols,” in Energy Procedia 56:414–422 · December 2014

[75] A. M. Zungeru, L. Ang, S. Prabaharan, K. Seng, “Radio Frequency Energy Harvesting and Management for Wireless Sensor Networks,” Green Mobile Devices and Networks: Energy Optimization and Scavenging Techniques, 2011

[76] http://www.seia.org/news/us-solar-market-set-grow-119-2016-installations-reach-16-gw

[77] http://energyinformative.org/best-solar-panel-monocrystalline-polycrystalline-thin-film/#best-solar-panel-type-for-home-use

[78] http://store.solaptop.com/

[79] https://en.wikipedia.org/wiki/Nickel%E2%80%93cadmium_battery

[80] https://en.wikipedia.org/wiki/Nickel%E2%80%93metal_hydride_battery

[81] https://en.wikipedia.org/wiki/Lithium-ion_battery

[82] "Helium Bluetooth speakers powered by supercapacitors," Gizmag.com. Retrieved 2013-11-29.

[83] "Coleman FlashCell Cordless Screwdriver Recharges In Just 90 Seconds," OhGizmo! 2007-09-11. Retrieved 2013-05-29

[84] https://en.wikipedia.org/wiki/Supercapacitor

[85] A. N. Parks, A. P. Sample, Y. Zhao, J. R. Smith, “A Wireless Sensing Platform Utilizing Ambient RF Energy,” IEEE WiSNET

[86] Y. Naruse, N. Matsubara, K. Mabuchi, M. Izumi and S. Suzuki, “Electrostatic micro power generation from low-frequency vibration such as human motion,” Journal of Micromechanics and Microengineering, Volume 19, Number 9, 2009

YUKAN ZHANG

17502 Rosa Drew Ln Apt 12D · Irvine, CA, 92612 · (607) 232-1627 · yzhan158@syr.edu

EDUCATION

- Syracuse University· PhD, Computer Engineering· Cumulative GPA: 3.87/4.0
- Binghamton University, SUNY· MS, Computer Engineering, Dec 2009· Overall GPA: 3.925/4.0
- Nankai University, Tianjin, P. R. China · BS, Electrical Engineering, Jun 2006

SKILLSET

- **Programming Languages:** C, C++, Shell Scripting, C#, JAVA, Matlab, Python
- **Hardware description languages:** Verilog, VHDL, Spice, Assembly Language
- **Software:** GCC, Vi, Visual Studio, Cadence Virtuoso, SoC Encounter, Synopsys Hspice, Design Compiler, PrimeTime, Mentor Graphics, Modelsim, Calibre, MATLAB
- **Operating Systems:** Unix, Linux, Windows

INDUSTRIAL ENGINEERING EXPERIENCE

Hardware Engineer· Shanghai Contec Microelectronic Co., Ltd.· www.contec.com ·7/2006- 6/2007

- Designed circuits of computer add-on boards using EDA tools.
- Read datasheets and selected suitable components to use in a design.
- Estimated costs, reliability, and safety factors.
- Developed product testing tools using Visual Studio.
- Wrote product specifications using Microsoft Office.

ENGINEERING EXPERIENCE

Research Assistant ·AMPS Lab College of Engineering and Computer Science, Syracuse University

- Research interests lie in the field of power management in green computing.
- Power model analysis and simulation.
- Power management on multiprocessors and embedded systems.
- Energy efficiency optimization for energy harvesting embedded systems.

RESEARCH PROJECTS

Energy efficiency optimization for an energy harvesting embedded system

- Developed a C++ simulator to model an energy harvesting embedded system.
- Used C++ polymorphism feature to model the similarity and difference of the behaviors of different energy storage banks.
- Analyzed and improved the model of power converters and reduced the power loss on them.
- Proposed an energy storage bank reconfiguration algorithm and a task scheduling algorithm to improve the energy efficiency.

Energy-aware task allocation and scheduling for conditional task graphs on multiprocessor platform

- Analysis the power consumption model for heterogeneous multiprocessor SoC, including Intel Xscale, PXA and IBM PowerPC.
- Achieved 10% average power savings compared to existing scheduling techniques while completed the tasks within their deadlines.
- Implemented task allocation and scheduling algorithm under Linux environment.
- Designed a conversion program which automatically generates multiprocessor simulation environments according to user specification and algorithm output.

- Validated the system and algorithm using OMNet++ network simulator framework.

COURSE PROJECTS

- Designed layout of 16-bit carry-bypass adder and 16-bit register file using TSMC 20process and Cadence Virtuoso Design tool; verified functionality and did timing analysis in Hspice simulator.
- Used Verilog to realize the system based on wireless LAN 802.11g application.
- Designed a 16 bit microprocessor using Verilog and simulated the functionality in Modelsim.
- Designed BSB recall operation using C on IBM cell processor.
- Used Cadence schematic tool to design and simulate a phase lock loop.
- Analyzed performance of a sensor station package data processing system using PPC ISS in C and VHDL.

PUBLICATIONS

[1] Yang Ge, Yukan Zhang, Parth Malani, Qinru Qiu, Qing Wu, "Low Power Task Scheduling and Mapping for Applications with Conditional Branches on Heterogeneous Multi-processor System", in Journal of Low Power Electronics, 8(5), 535-551, Dec. 2013.
[2] Yang Ge, Yukan Zhang, Qinru Qiu, "A Game Theoretic Resource Allocation for Overall Energy Minimization in Mobile Cloud Computing System", in International Symposium of Low Power Electronics and Design, Aug. 2012.
[3] Yang Ge, Yukan Zhang, Qinru Qiu, "Improving energy efficiency for energy harvesting embedded systems", in Asia and South Pacific Design Automation Conference, Jan. 2013.
[4] Yukan Zhang, Yang Ge, Qinru Qiu, "Improving charging efficiency with workload scheduling in energy harvesting embedded systems," in Design Automation Conference, Jun. 2013.
[5] Yang Ge, Yukan Zhang, Qinru Qiu," Distributed Task Migration in a Homogeneous Many-core System for Leakage and Fan Power Reduction", JOLPE, Vol. 10, N° 4, December 2014.

TEACHING EXPERIENCE

Teaching Assistant · Watson School of Engineering and Applied Science, Binghamton University

Computer Organization and Microprocessors	**Spring/2010**
Digital System Design I	**Fall/2010**
Computer Communication and Networking	**Spring/2011**

- Assisted students with their lab procedure
- Hold weekly office hour to answer student questions
- Graded lab reports and course projects

OTHER WORK EXPERIENCE

Server · Food court in Student Union, Binghamton University · Fall/2007 – Fall/2008