Energy Analysis of a Real-time Multiprocessor Control of Idle States

Jabran Khan, Sébastien Bilavarn, Khurram Bhatti and Cécile Belleudy
Laboratoire d'Electronique, Antennes et Télécommunication, Université de Nice, Sophia Antipolis, France

Keywords: Linux, Low Power, DPM.

Abstract: This paper focuses on the analysis of a dynamic low power switching technique called assertive dynamic power management (AsDPM) on ARM based platforms. The availability of ready tasks during the execution of a program is random. The choice to when exactly a ready task is executed on certain processor and how many processors are required for the remaining tasks can save a significant amount of energy utilization. This paper focuses on the energy efficiency of AsDPM strategy for real-time tasks, which decides when exactly a ready task shall execute; thereby reducing the number of active processors, which eventually reduces energy consumption. We will analyze the energy gains resulting from the implementation of this AsDPM power strategy for different ARM based multiprocessor platforms (ARM1176JZF-S, CortexA9). Results show significant amounts of gains up to 60% for different execution conditions*.

1 INTRODUCTION

As applications are becoming more and more complex, processing power is continuously increasing having a significant impact on embedded device battery life. The battery technology has not been able to match the advancement race in modern hardware devices, therefore puts more burden on implementation of new algorithms to cope with the demand. Dynamic power switching (DPS) that is the selective shutdown of system components that are idle or underutilized, has proven to be an effective technique for reducing power dissipation in such scenarios.

This paper presents a power optimization study for real-time embedded applications on ARM based platforms. Our goal is to implement a power management strategy on real development platforms in order to analyze and evaluate its operational behaviour. The work will mainly focus on finding out the conditions for energy gains for different platforms. An assertive dynamic power management (AsDPM) technique proposed by (Bhatti, 2009) is shown to be able to bring significant energy savings, while satisfying real time constraints for different applications. In this paper, we analyze the potential of AsDPM across different platforms based on recent generations of ARM processors. The availability to monitor the actual core power consumption inspired us to use the ARM based platforms in our work. We will also analyze the AsDPM strategy in different platform configurations (i.e. 2, 3 and 4 processors), to observe the efficiency of the strategy in a real multiprocessor environment.

The paper addresses these issues in the following manner. Section 2 state previous work and investigation efforts in energy and power management. Section 3 is divided into three parts; Section 3.1 explains briefly the AsDPM strategy and test applications used for the experiments, Section 3.2 puts light on different platforms used (i.e. ARM1176JZf-S, QEMU) and Section 3.3 focuses on the implementation of AsDPM on these platforms. In section 4, we detail the results and analyze the conditions of energy gains based on real measures and in section 5 we present our conclusions and future perspectives.

2 STATE OF ART

Research has been focused on estimating power and energy consumption using system level events

* This work is carried out under the COMCAS project (CA501), a project labeled within the framework of CATRENE, the EUREKA cluster for Application and Technology Research in Europe on NanoElectronics.

Khan J., Bilavarn S., Bhatti K. and Belleudy C..
Energy Analysis of a Real-time Multiprocessor Control of Idle States.
DOI: 10.5220/0004314001250130
In *Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems* (PECCS-2013), pages 125-130
ISBN: 978-989-8565-43-3

(Joseph and Martonosi, 2001), event counters (Benini et al., 1998) or at the instruction level (Tiwari, 1994). However, allowing OS and related software to gain control over power is really gaining more and more interest these days, as energy reduction is one of the prime concerns in embedded systems. At the processor level, two popular techniques are mainly employed: Dynamic Power Switching (DPS) to switch off the power supply of a part of the circuit (Benini et al., 2000), and Dynamic Voltage and Frequency Scaling (DVFS) to tune a processor clock speed and its corresponding voltage according to the requirements such as the workload (expected or actual) or the battery charge. Usually, the techniques based on DVFS are widely used to reduce power and energy consumption while DPS is used to solve thermal dissipation problems (Yang et al. 2009). (Benini and Micheli, 1997), as well as (Irani, Shukla and Gupta, 2003) have presented research on evaluating DPS techniques at processor level. Similarly, (Hwang and Wu, 2001) used a regressive analysis of the running tracks that rely on task activity prediction, to put hardware in sleep mode when possible.

Microprocessor manufacturers have provided different solutions to make use of the DPS and DVFS techniques. For example, (ARM, 2006) provided a policy manager called Intelligent Energy Manager (IEM), which handle system configuration according to the actual and/or predicted workload. (Intel, 2004) proposed a similar technology Enhanced Intel SpeedStep (EIST), that is integrated in the Pentium M-series processors to manage power. A variety of power management strategies are also available today in popular operating systems to control the power consumption of the CPU and its devices. For example Linux OS, by the help of ACPI, provides governors to use DPS and DVFS techniques. Similarly in Windows OS, we have different schemes (Max Battery, Performance, etc.) to manage power and energy, with the help of ACPI. These strategies have an advantage of being applicable in all cases (general purpose), but the drawback is probably a certain level of inefficiency. Most of these strategies are defined on the basis of overall workload on which their efficiency varies. However, there are very few techniques that provide power management within an application while it is executing.

The use of application specific power management strategy provides an extra room for power management by utilizing the idle time more efficiently. (Cheng and Goddard, 2006) showed that DPS techniques achieve energy conservation in embedded computing systems by selectively putting its components into power-efficient states, sufficient to meet functional requirements. In our work, we will implement a DPS based AsDPM technique, that mainly considers the processors for power and energy consumption, during execution of a certain application. It works on the principle of admission control for ready tasks by delaying the execution of ready tasks as much as possible. This controls the maximum number of active/running processors in the system at any time instant. Next section details the implementation of this technique.

3 AsDPM IMPLEMENTATION

In this section, we describe a real implementation of a power management strategy called AsDPM for multiprocessor low power scheduling. With this implementation, experiments and simulations have been carried out with mainly two objectives in mind. (a) To compare the efficiency and behavior of our real ARM1176JZF-S platform with the virtual QEMU_ARM1176 platform, (b) To implement our strategy on multiprocessor platforms to verify the feasibility and correctness of our AsDPM strategy on different multi-core platforms.

3.1 AsDPM Strategy and Test Applications

AsDPM strategy is DPS based power strategy in which the number of processors to use, depends upon the amount of remaining tasks and their deadlines. AsDPM technique exploits the idle time intervals within an application. Conventional DPS techniques can exploit idle intervals only once they occur on a processor. Upon detecting idle time intervals, these techniques decide whether to transition target processor(s) to power-efficient state. AsDPM technique, on the other hand, aggressively extracts most of the idle time intervals from some processors and clusters them on some other processors of the platform to elongate the duration of idle time.

At every scheduling event, the strategy performs a test of whether the remaining tasks to be executed are schedulable on either one processor, two or more processors. The required numbers of processors for the remaining tasks are calculated based on this test. Afterwards the highest priority task using EDF scheduler (having shortest deadline) is allocated to the first processor and so on. If system requires only one processor, the second highest priority task is executed on the same processor after finishing the first one. If system requires two processors, the first higher priority task is executed on the first processor, the second priority task on second and similarly the process goes on until the completion of the remaining tasks. By this way we execute our program on the least number of processors required, hence minimizing the energy utilization. At the end as higher priority tasks finish earlier, meaning the one on processor one will complete earlier. Therefore when a scheduling event occurs, the task on second processor is moved to the first processor for its completion. Hence we minimize the total number of needed processors after each scheduling event as well as at the end.

To evaluate AsDPM strategy in the real implementation world, we have used four test applications and three different platform configurations. *Example.1* and the video encoder H.264 example consists of four tasks running on two processor configuration. *Example.3* contains six tasks to be executed on three processors and *Example.4* contains eight tasks and needs four processors. We used an ARM1176JZF-S based real platform and two QEMU (Gligor et al., 2010) based virtual platforms to test the scheduler. However the virtual platforms should not be confused with high level application simulators. The QEMU based virtual platforms used in our work are provided by our COMCAS project partners TIMA labs. The virtual platforms (QEMU_ARM1176 and QEMU_CortexA9) performance and functionalities match to those of the real hardware platform baseboards.

3.2 Platforms used

We have used in total three platforms to study the energy behavior of AsDPM. The first one is a real ARM1176JZF-S platform and the two others are QEMU based virtual platforms: one composed of ARM1176 processors and other of CortexA9 processors. We used the virtual platforms due to the unavailability of platforms supporting multi-core execution and processor power measurement features at the same time. The latest embedded Linux (2.6.33) is used as an operating system for the platforms. The programs are compiled using the code sorcery cross compiler for the target platforms. The code is compatible on both real hardware and virtual platforms. Similarly it can be used and run by any other platform using Linux OS by cross compiling for that specific platform. However such systems (Laptops, PCs etc.) do not allow access to the real time processor power utilization (energy consumption could not analyzed).

To evaluate our strategy, we first experimented an EDF scheduler using a mono-processor platform with power measurement facilities (ARM1176JZF-S) in order to compare with an identical virtual platform, and to quantify the accuracy of QEMU power estimations. Afterwards, we used two virtual platforms in multi-core configurations (ARM1176, CortexA9) to evaluate the energy gains in multi-core execution scenarios. We will analyze and compare the percentage energy gains with and without our strategy implementations. As an illustration the energy consumed by our applications using EDF scheduler is compared with the energy consumed by the applications with our AsDPM strategy.

3.2.1 ARM1176JZF-S

The platform baseboard PB ARM1176JZF-S contains an ARM1176JZF-S core and a Virtex-4 XC4VLX40 FPGA. It also contains the Intelligent Energy management (IEM) technology which is in charge of controlling the power supply. The platform baseboard also contains the main memory system i.e. 128MB of 3bit wide Mobile DDR RAM, 8MB of 32-bit PSRAM and two 64MB of 32-bit NOR flash, bus control (AMBA AXI) and other peripherals with their controllers (implemented on the FPGA). The ARM1176JZF-S processor has a maximum frequency of 265 MHz It also contains built-in registers connected directly to processor to monitor core current and voltage hence the power consumption of the main core. A Linux driver has been developed in order to poll these registers at regular interval of times and provide reliable power consumption profiles.

3.2.2 QEMU Platforms

QEMU is a generic and open source machine emulator and Virtualizer. When used as a machine emulator, QEMU can run the OS and programs made for one machine (e.g. on PC) on a different machine (e.g. an ARM board). In our case, we have two virtual platform configurations namely QEMU_ARM1176 and QEMU_CortexA9. QEMU_ARM1176 matches the specifications of a real ARM1176JZF-S platform whereas QEMU_CortexA9 matches the specifications of the ARM CortexA9 platform. The processor frequency along with the corresponding power levels with and without load are shown in Table 1. These values have been derived directly from the real measures on an ARM1176JZF-S PB and on a Snowball platform with Dual Cortex A9 (ST-Ericson, 2011). The differences in power consumption, shown in table 1, also affect the energy consumption by the platforms. The platform characteristics plays an important role in overall energy gains as discussed in detail by (Khan and Bilavarn, 2012).

Table 1: Power consumption for QEMU_ARM1176 and QEMU_CortexA9 platforms.

QEMU Platforms	Frequency MHz	Power (No Load) mWatt	Power (Load) mWatt
ARM1176	265	252	330
CortexA9	1000	90	320

The availability of the Linux *CPUIdle governor* allows DPS switching in ARM1176JZF-S platform, however, QEMU platforms allows DPS and power consumption estimates using its own customizable drivers. We have thus developed a control driver *PM_Driver* to change the required processor state to our desired level (i.e. Idle, Sleep, Running) and *PM_Monitor* driver to monitor power and energy. A test example is executed on both real ARM1176JZF-S and QEMU_ARM1176 platform before implementing the AsDPM strategy in a multi-processor configuration. The example application is similar to those used for analyzing the AsDPM strategy later on. The actual execution (AET) of the test example was changed from best case to worst case. The energy and timing results for both platforms are summarized in Table 2.

Table 2: Energy and Performance Analysis of real ARM1176JFZ-S and QEMU_ARM1176 platform.

AET	Energy mJ	Time ms	Power mW	% Error	
				Energy	Time
Conf 1	2730.72	309.99	8.809	0.31	0.034
	2722.29	308.93	8.812		
Conf 2	2792.58	317.05	8.808	0.31	0.034
	2783.96	315.96	8.811		
Conf 3	2801.42	318.05	8.808	0.31	0.023
	2792.77	317.00	8.810		
Conf 4	2810.25	319.02	8.809	0.31	0.023
	2801.58	317.96	8.811		
Conf 5	2819.09	309.99	8.810	0.31	0.011
	2810.39	318.96	8.811		

Analysis proved that both the platforms showed similar behavior in terms of energy consumption and performance. There was a negligible error of 0.03% in timing analysis. Similarly, the energy consumed was also similar on both platforms with an error or 0.31% (8 to 10 milli joules that can be neglected).

3.3 AsDPM Implementation and Energy Measurements

In order to implement the AsDPM strategy on the considered platforms, a *PM_Scheduler* program (containing AsDPM strategy) is loaded to the platforms. The drivers to choose between different power C-states and to monitor power and measure energy are also loaded at the start of this program. The *PM_Monitor* driver was also able to measure the instantaneous power consumption and the mean power between two defined points (i.e. from the start of simulation to end). Consequently it can also derive the corresponding energy consumption. When the entire execution of the test application (i.e. *Example.1, 2, 3* or H264 encoder etc.) is completed, the power management program *PM_scheduler* calls the *PM_Monitor* driver to stop and return the mean power and energy by the processor(s) to the console. The measured values are stored in a file for future analysis.

To experiment with the AsDPM strategy, we will first implement three test examples (*Example.1*, *Example.2* and *Example.*3) in order to analyze the energy and application behavior in different multi-core configuration (2, 3 and 4 processors). Afterwards we will use the H.264 encoder examples on both platforms in order to measure and analyze energy gains for the video encoder. The examples used are based on the algorithm defined by (Bhatti, 2009). In our work, we focus on analyzing the behavior of the AsDPM strategy on different platforms, the parameters affecting the energy gain, and compatibility with different multi-core configurations. Therefore we will not detail the example structure in our work. However some of the application parameters are listed on which the energy gain depends like worst case execution time (WCET), Actual execution time (AET), Best case execution time (BCET) etc. We have used variable length example applications as the total execution time does not affect the overall energy consumption.

4 RESULTS

We started implementing our examples on both QEMU_ARM1176 and QEMU_CortexA9 platforms by varying our actual execution time (AET) from best case execution time (BCET) to worst case execution time WCET. By this means, we can have a range of minimum and maximum energy gains for our strategy. In *Example.1*, the tasks are defined in such a way that the first two tasks are parallel and the third and fourth tasks are randomly chosen. By this way the *Example.1* requires two processors for execution at the beginning. Afterwards the *PM_Scheduler* takes the decision of either using one or two processors for the remaining tasks by minimizing the number of processors. The percentage energy gain with and without our AsDPM strategy is shown in Figure 1 for both platforms. Figure 1, shows the results of the implementations of *Example.1* on both QEMU_ARM1176 and QEMU_CortexA9 platforms. The energy gains are between 24.88% to 34.67% for QEMU_ARM1176 and 12.58% to 28.11% for QEMU_CortexA9 platform.

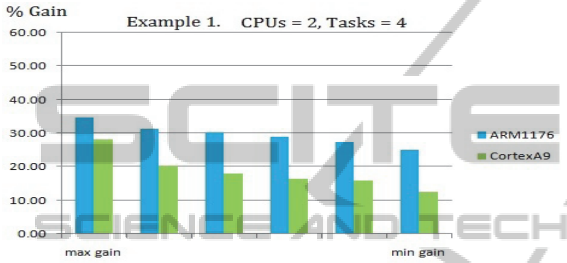

Figure 1: Percentage Energy gain for *Example 1*.

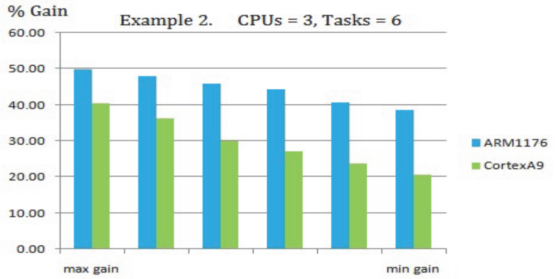

Figure 2: Percentage Energy gain of *Example 2*.

Figure 2 shows the energy gain of *Example.2* for both platforms. It consists of 6 tasks executed on platform configuration having three processors. Results show energy gains ranging between 38.65% to 49.88% for QEMU_ARM1176 and between 20.51% to 40.55% for QEMU_CortexA9 platform. Similarly, figure 3 shows results of *Example.3* having eight tasks implemented on platform having four processors. Percentage energy gain ranges between 50.64% to 59.18% for QEMU_ARM1176 and between 30.95% to 47.69% for QEMU_CortexA9 platform. We have therefore shown the compatibility of AsDPM strategy for different multi-core scenarios (up to 4 processors). The results provided significant energy gains as well.

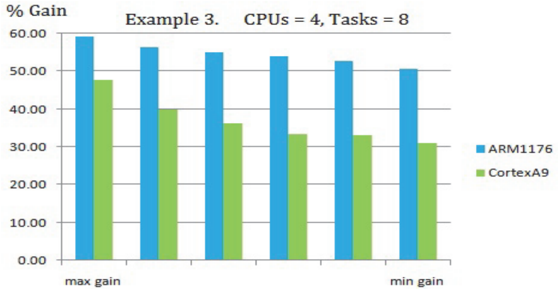

Figure 3: Percentage Energy gain of *Example 3*.

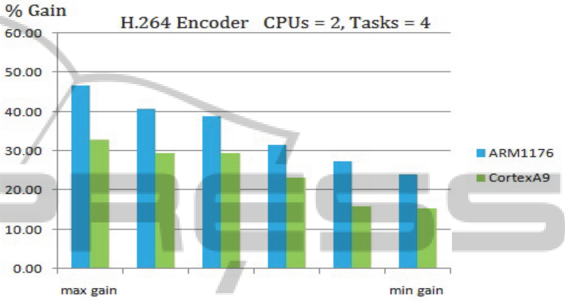

Figure 4: Percentage Energy gain of H.264 Encoder.

In Figure 4, savings for an H.264 encoder containing four tasks implemented on two processor configuration is shown. The percentage energy energy gain ranges between 24.05% to 46.73% for QEMU_ARM1176 platform and 15.32% to 32.72% for QEMU_CortexA9 platform.

It should be noted that the percentage energy gains are higher in case of QEMU_ARM1176 platform in comparison to QEMU_CortexA9 platform. The reason is related to the operating points of the ARM1176 core (Idle vs. Load power level). Table 3, provides few results obtained by changing AET for the H.264 encoder example for both platforms.

Table 3: Energy consumption of H.264 encoder on QEMU_ARM1176 and QEMU_Cortex A9 Platforms

AET	QEMU_ARM1176		QEMU_CortexA9	
	Energy (mJ)	% Gain	Energy (mJ)	% Gain
Conf 1	7350.12		5080.23	
	3915.76	46.73	3417.94	32.72
Conf 2	7866.55		5746.56	
	4660.13	40.76	4052.16	29.49
Conf 3	8022.68		6461.38	
	6092.99	24.05	5471.44	15.32

Results show the total energy consumed by the H.264 encoder example, with and without the

AsDPM strategy. QEMU_ARM1176 consumes more power while the processor is in running state whereas QEMU_CortexA9 consumes much less power (Table 1). The difference between load power and idle power of both platforms explains the differences in percentage energy gains. If a processor on one platform consumes more power while executing certain application than on another, it will provide more power savings when it is idle as shown in Table 3. However in terms of total energy consumption by an application, the QEMU_Cortex platform is more efficient. As an illustration, the total energy consumption of H.264 Encoder for QEMU_ARM1176 with and without AsDPM strategy is 6.62 Joules and 8.02 Joules respectively. However the same energy consumption for QEMU_CortexA9 platform was 5.7Joules and 6.2 Joules (much lower due to efficient platform).

5 CONCLUSIONS AND FUTURE PERSPECTIVES

We have presented and analyzed the effectiveness of a DPS based AsDPM power strategy on different applications (including video encoding) for different ARM based platforms. We have also validated the execution of the real ARM1176JZF-S platform with our virtual QEMU_ARM1176 platform. The results show the same behavior on both platforms having a negligible deviation of 0.03% of time and 0.31% (few milli-joules) for energy consumption. We have also shown that our strategy is compatible with different configuration of multi-core platforms (i.e. QEMU_ARM1176 and QEMU_CortexA9) and provided significant energy gains ranging between minimum gains of 12.58% to a maximum gain of 60% under different operating conditions. Using the virtual platform, we have thus explored the efficiency of the DPS strategy for different applications implemented under several platform configurations (2, 3 and 4 processors). The power strategy provides gain that ranges up to 60% depending upon the different values of the actual execution time as well as the number of processors.

Other perspectives of this work are to implement and study power strategies like presented by (Khan, 2012) and (Chéour, 2011) on real hardware platforms boards like the ARM1176JZF-S and the ARM11 CortexA9 in order to explore their effectiveness in the real development world.

REFERENCES

M. K. Bhatti, M. Farooq, C. Belleudy, M. Auguin, O.Mbarek, 2009, "AsDPM strategy for globally scheduled RT multiprocessor systems", In *proceedings of 19th international conference of ICSD,PATMOS'09.*

R. Joseph, M.Martonosi, 2001, “Runtime Power Estimation in High Performance Microprocessors”, *Symposium on Low Power Electronics and Design.*

L. Benini, A. Bogliolo, and G. De Micheli, 1998, “Monitoring system activity of OS-directed dynamic power management”, *International Symposium on Low Power Electronics and Design*.

V. Tiwari, S. Malik, and A. Wolfe, 1994, “Power analysis of embedded software: a first step towards software power minimization”, IEEE Transactions on Very Large Scale Integration (VLSI) systems.

L. Benini, G. De Michelli, 2000, “A survey of Design Techniques for System-Level DPM”, *IEEE Transactions on VLSI systems*, Vol.8, No.3.

C. Y. Yang, J. J. Chen, T. W. Kuo, L. Thiele, 2009, “An Approximation Scheme for Energy-Efficient Scheduling of Real-Time Tasks in Heterogeneous Multiprocessor Systems”, *DATE '09*.

L. Benini and G. De Micheli, 1997, “*Dynamic Power Management: Design Techniques and CAD Tools*”, Springer; 1st edition.

S. Irani, S. K. Shukla, and R. K. Gupta, 2003, “Online strategies for DPM in systems with multiple power-saving states”. *ACM Trans. Embed. Syst.*

C. H. Hwang, A.C-H.Wu, 2001, “A predictive system shutdown method for energy saving of event driven Computation”, In *proceedings of TODAES’01*.

ARM, 2006, “Intelligent Energy Manager (IEM) in the ARM1176JZF-S Development Chip. Application Note 172. http://infocenter.arm.com.

INTEL, 2004, *Enhanced Intel Speed Step (EIST) for multi-core processors with unified Level 2 cache*. http://www.intel.com/support/processors.

H. Cheng and S. Goddard, 2006, “Online energy-aware i/o device scheduling for hard real-time systems”, In *proceedings of DATE ’06*, Belgium.

M. Gligor, N. Fournel, F. Pétrot, F. Colas-bigey, A.M. Fouilliart, P.Teninge, M.Copolla, 2010, “Practical Design space exploration of Handheld devices using Virtual Platform”, *Lecture notes in CS*; Vol 5953/2010.

ST-Erricson, 2011, “*Hardware Reference Manual SKY-S9500-ULP-CXX*” (aka Snowball PDK-SDK), Revision 1.0.

J. Khan, S.Bilavarn, C. Belleudy, 2012, “Impact of Operating Points on DVFS Power Managemeng”, in *7th International conference DTIS’12*), Tunisia.

J. Khan, S. Bilavarn, C.Belleudy, 2012, "Energy analysis of a DVFS based power strategy on ARM platforms" in *11th IEEE conference FTFC*, Paris France.

R. Chéour, S. Bilavarn, M. Abid, 2011, “Exploitation of the EDF scheduling in the wireless sensors networks”, *Measurement Science Technology Journal (IOP)*, Special Issue: Devices, Signals and Materials.