A Power-Aware Online Scheduling Algorithm for Streaming Applications in Embedded MPSoC

Tanguy SASSOLAS, Nicolas VENTROUX, Nassima BOUDOUANI and Guillaume BLANC

CEA List
contact: firstname.surname@cea.fr

Abstract. As application complexity grows, embedded systems move to multiprocessor architectures to cope with the computation needs. The issue for multiprocessor architectures is to optimize the processing resources usage and power consumption to reach a higher energy efficiency. These optimizations are handled by scheduling techniques. To tackle this issue we propose a global online scheduling algorithm for streaming applications. It takes into account data dependencies between pipeline tasks to optimize processor usage and reduce power consumption through the use of DPM and DVFS modes. An implementation of the algorithm on a virtual platform, executing a WCDMA application, demonstrates up to 45% power consumption gain while guaranteeing regular data throughput.

Index Terms - *scheduling, low-power, multiprocessor, streaming*

1 Introduction

As embedded applications become more complex, future embedded architectures will have to provide higher computing performances, while respecting strong surface and consumption constraints. Embedded devices will not only execute more computing intensive applications but also cross-domain ones, including telecom and video processing application . To cope with these demands an emerging trend in embedded system design lies in the conception of MultiProcessor Systems-on-Chips (MPSoC.)

These new architectures with a high density of processing elements have a strong energy dissipation. This dissipation must be taken into account to match an embedded-compliant power budget and to limit ageing phenomenon. To handle these thermal and energy issues, MPSoC designer integrate DVFS and DPM capabilities in their platform.

To leverage MPSoCs processing capabilities, applications need to be highly parallelized. A simple way to increase application parallelism and data throughput is to pipeline sequential applications into streaming ones. This applies to the WCDMA application whose parallelism can be drastically increased. Then pipeline stages must be efficiently allocated to the processing resources while taking into account data dependencies between them. As applications become more prone to execution time variation, online control solution are needed to dynamically schedule tasks and increase processor load. This variations can stem from the differences in input data for data processing application; or from the application structure itself. For instance the WCDMA application differently processes a pilot frame from a user frame.

Only a global scheduler with a complete view of the computation resource and task states can perform an optimal scheduling. The choice of global scheduling pushes forward the use of a central control solution. In addition, an online central control solution must react quickly to platform events. Therefore, online scheduling must remain simple and must find a balance between accuracy and execution speed. In this article, we propose an online power-aware scheduling algorithm that matches these conditions. This algorithm focuses on the scheduling of streaming applications. Our scheduling algorithm also tackles power consumption issues through an efficient use of Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) modes of the processing resources.

This paper is organized as follows: section 2 will study existing solutions in the field of power-aware streaming application scheduling. Then, section 3 will describe the proposed power-aware scheduling algorithm. Section 4 will detail implementation issues focusing on the simulation framework and the targeted MPSoC platform. Results will be presented in section 5 where the impact of our scheduling algorithm in terms of Quality of Service (QoS) and power consumption gain will be evaluated. Finally section 6 will discuss this new streaming application scheduling algorithm capabilities and its future improvements.

2 Related work

We focus our study on power-aware sheduling algorithms that rely on DVFS and DPM techniques [1]. First of all, we will briefly present the DPM and DVFS techniques and their impact on energy consumption. Then we will present a survey of previous works in the field of offline power-aware scheduling techniques for streaming processing. Finally we will expose online low-power scheduling techniques for dependant tasks.

The dissipated power in a CMOS design can be divided into two major sources: the dynamic power consumption and the static one. The dynamic consumption part is mainly due to transistor state switching and it can be drastically reduced by lowering the supply voltage. As the transistor delay is a function of the supply voltage, lowering the supply voltage imposes an adapted frequency reduction. This technique is called DVFS.

The static consumption is due to various current leakages in the transistor. The DVFS technique has some impact on the static power consumption thanks to the supply voltage reduction. Nonetheless this is not sufficient to drastically reduce static power consumption. To cut down static power consumption the only viable solution consists in switching off unused parts of a circuit. This technique is called DPM. Contrary to the DVFS technique the resource is made unavailable.

The main drawback of these two techniques lies in the timing and consumption mode switching penalties. If the timing penalties for the DVFS are rather constrained, it is not the same for the DPM where wake-up time can reach a hundred milliseconds (136ms for the PXA270 [2].) Therefore, for a processor implementing both techniques, the issue is to find when reducing the voltage and frequency couple is more energy efficient than running at full speed then switch of the processor. This matter is summarized in Fig 1. For a given technological process, the issue is thus to evaluate the duration of future inactivity periods of the resource. Having introduced the DVFS and DPM technique and the optimization problem they imply, we will now present offline low-power scheduling technique for streaming applications.

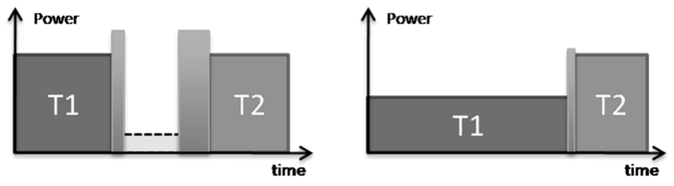


Fig. 1. DPM (left)and DVFS (right) technique timing issues.

Given the fact that scheduling on a multiprocessor environment is an NP complete problem [3], adding power consumption optimization to the problem makes the issue of power-aware scheduling for multiprocessor harder to solve. Streaming application can be seen as a set of tasks linked by their data dependencies. Thus, scheduling dependent tasks allows to schedule streaming applications. Many offline solutions have been proposed to solve this optimality issue assuming task dependencies and their execution lengths were available. They mainly vary in the way they describe the problem, changing which parameters have to be taken into account, and the computing optimization method used to solve the problem like in [4].

To the authors' knowledge no previous work has been done to find an offline low-power multiprocessor scheduling dedicated to streaming application. Nonetheless an interesting line of work has been developed with the same scope but for monoprocessor environment. In [5] the authors study the power optimization by using DVFS technique on a streaming application described as a directed acyclic graph with a constant output rate. Their solution allows to find the lower consumption scheduling given buffer size or finding the buffer size given a power budget. A similar approach is taken in [6] with DPM utilization. To meet more realistic application they describe the production rate as a random variable following a given probability rule. Nonetheless, variations in the effective execution time limit the performance of offline solutions. To handle this dynamism, online low-power solution have been proposed for streaming applications.

Many online solutions have been designed for the case of independent tasks [7, 8] but they cannot apply for streaming applications. Online scheduling that handle task dependency issues are uncommon. Interesting solutions for dependant task scheduling have been proposed by [9, 10]. Nonetheless, these solutions rely on a partitionning of resources. Partionning solution are necessarily sub-optimal as they only handle resources separately. A global scheduling can potentially reach a better resource usage.

We remind for the reader's knowledge a few online power management techniques used for mono processor architecture in the case of streaming application described with a Directed Acyclic Graph (DAG). In [11] the author take into account potential blocking communication between tasks to always run the data producer at full speed in that case but lower the energy consumption otherwise. [12] presents another example of inter task communication buffer size optimization, with this time an online scheduler handling slack time accumulated with buffer use. None of the strategies listed above take into account the online scheduling of streaming applications that allow a pipelined execution and potential output rate improvements in an MPSoC environment.

3 Power-aware streaming application scheduling

We believe that a more power-efficient scheduling for dynamic streaming applications can be found by the use of an online global scheduling. In this section, we will first remind the application description used by our algorithm. Then we will explain the grounds of our algorithm, before presenting it in detail.

Our scheduling algorithm has been written to handle streaming applications described in a specific way. An application is a set of tasks with consumer/producer relationships. Data is transferred from a producer task to a consumer task through a circular buffer. Only one task can write on a buffer while it can be read by multiple consumer tasks. This creates a divergence in the data flow. A consumer task can also read multiple input buffers, creating a convergence in the data flow. This allows the description of parallelism in the processing flow of a given data.

Given the previously described application model, one can make a few observations. A streaming application throughput is constrained by the duration of its slowest stage. As a result other pipeline stages can be slowed down to meet the same output rate as the slowest stage. This can be performed by using a slower DVFS mode for the resources with a too high output rate. Besides, tasks that are further in the pipeline stream than the slowest task are to be blocked waiting for data. These tasks should be preempted if other tasks can execute instead, or the resource should be shut down if not. This implies the use of DPM functionalities. Given these observations, our algorithm will use DVFS to balance the pipeline stage length and DPM to shut down unused resources. Our objective is to maintain the same data throughput as if the task were executing at full speed while making substantial energy saving.

To be able to balance an application pipeline, we need additional information on the dynamic output rate of a task. Thus we introduce monitors on every communication buffer. For every buffer we specify how many dataset it can contain. We also specify two thresholds. When the higher threshold is reached we assume that the producer is executing to fast. When the lower threshold is reached we assume that the producer is not executing fast enough. A specific event is sent to the scheduler when a threshold is crossed. It contains the writing task identifier. An event is also sent when a task is blocked reading an empty buffer, as well as when a task is blocked writing a full buffer. The buffer monitors are summarized in Fig. 2. One objective of balancing pipeline stage length is to prevent buffers from getting full, which would block the producer. And to never reach an empty buffer, which would block the consumer and could result in an increase of the data processing length.

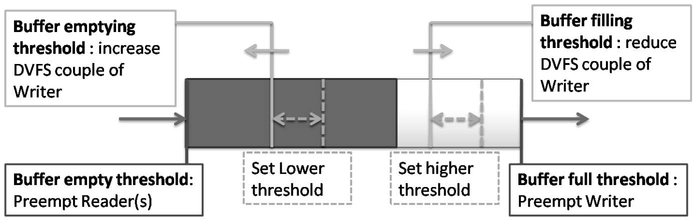

Buffer emptying threshold : increase DVFS couple of Writer
Buffer filling threshold : reduce DVFS couple of Writer
Buffer empty threshold: Preempt Reader(s)
Set Lower threshold
Set higher threshold
Buffer full threshold : Preempt Writer

Fig. 2. Summary of buffer monitors and scheduling implications

To keep our scheduling algorithm as simple as possible the task priorities are made of a static and a dynamic part. We will list the different priority parts by level of importance. First we check the blocked task status, as we do not want to give the priority to a blocked task. Then the application priority is taken into account. After that, we study pipeline position priority. Every task is given a priority depending on its position in the streaming pipeline. This allows to give the priority to tasks handling older dataset, i.e the ones that are deeper in the pipeline. Finally for tasks that have the same pipeline position priority, we give the priority to the task with the emptier buffer. The complete scheduling loop is described in Algorithm 1.

4 Implementation

To study and validate our algorithm we implemented it on a virtual MPSoC. In this section we will first present the SESAM simulation framework. Then, we will describe the specificities of the simulated MPSoC. Finally we will shortly present the WCDMA application used for our performance analysis.

SESAM [13] is a tool that has been specifically built up to ease the design of asymmetric multiprocessor architectures. This framework is described with the SystemC description language, and allows MPSoC exploration at the TLM

Algorithm 1 The Power-Aware Streaming Application Scheduling Loop

```
1: procedure SCHEDULING($task\_to\_schedule[nb\_tasks]$, $status\_proc[nb\_proc]$)
   ◇ First we take into account buffer events
2:     for all tasks to schedule do
3:         if $task$ is waiting for data then
4:             remove task from task_to_schedule
5:         else if $task$ output buffer reached Higher Threshold then
6:             reset $task$'s buffer priority bit
7:         else if $task$ output buffer reached Lower Threshold then
8:             set $task$'s buffer priority bit
9:         end if
10:    end for
   ◇ Then we order the tasks by priority
11:    $ordered\_tasks[nb_proc] \leftarrow$ sort_task_by_priority($task\_to\_schedule$)
   ◇ We handle already in execution tasks to limit preemption/migration
12:    for all $task$ already in execution in $ordered\_tasks$ do
13:        remove $task$ from ordered_tasks
14:        remove $proc$ executing $task$ from $free_proc$
15:    end for
   ◇ We allocate tasks not in execution on any processor yet
16:    for all $task$ left in $ordered\_tasks$ do
17:        execute $task$ on $free_proc$
18:    end for
   ◇ Finally we handle the consumption
19:    for all $proc$ do
20:        if $proc$ is free then
21:            $proc\_mode \leftarrow idle\_mode$
22:        else if $Task$ on $proc$ reached $lower\_threshold$ then
23:            $proc\_mode \leftarrow turbo\_mode$
24:        else if $Task$ on $proc$ reached $higher\_threshold$ then
25:            $proc\_mode \leftarrow half\_mode$
26:        end if
27:    end for
28: end procedure
```

level with fast and cycle accurate simulation. Besides, SESAM uses approximate-timed TLM with explicit time to provide a fast and accurate simulation of complex NoC communications [14]. It performs simulations with an accuracy of 90% compared to fully cycle accurate models. In addition, the programming model of SESAM is specifically adapted to dynamic applications and global scheduling methods. It is based on the explicit separation of the control and the computation parts.

The processing elements of the SESAM simulator are functional Instruction Set Simulators (ISS) generated by the ArchC tool. Thus, we extended the ArchC ISS to integrate DVFS and DPM models to the SESAM environment. To avoid multiple context switches and accelerate simulation, every ArchC ISS executes multiple instructions at a time then waits for the time it should have spent executing them. For every DVFS mode, we calculate the smallest couple (a, b) so that a/b equals the DVFS mode slowing factor. Then, we multiply the number of instructions to be executed by a and the time to wait for these instructions by b. We also calculate the energy spent during the execution of a set of instruction and keep the total energy consumption for each ISS. A DVFS mode switch is modelled as an interruption for the ISS. When it occurs, the ISS computes the time and energy spent in its previous mode. Then, it waits for the adequate switching latency, takes into account its switching energy penalty and finally resumes its execution with the (a, b) couple of the new DVFS mode. So as to model realistic processors we used the PXA270 Power State Machine (PSM) values [2]. We chose to use only two DVFS modes, *Turbo* and *Half-turbo*, and one DPM mode, *Deep Idle*, as they have acceptable switching latencies compared to our task execution times.

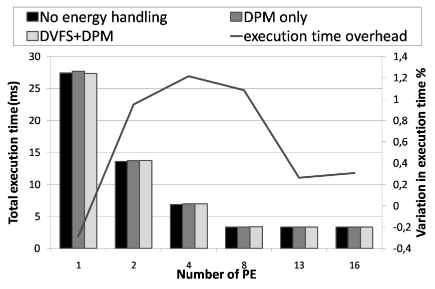
To perform a realistic analysis of our scheduling algorithm we modelled with the SESAM simulator an asymmetric MPSoC platform. This platform is build of a set of Processing Elements (PE) made of a processor equipped with a TLB, a 1KB instruction cache and a 1KB data one. They are connected to a set of shared 2ns-latency L2 memory through a 2ns-latency multibus. Communication between tasks are made possible thanks to HAL functions. Data coherency is guaranteed by a memory Management Unit (MMU). The buffers used for our algorithm are modelled using a specific HAL and the buffer thresholds are handled by the MMU. Preemption and migration of tasks are possible and their costs is reduced thanks to the shared memory and the virtualization of the memory space enabled by the use of TLBs [13].

The central controller is made of a processor with its own caches and memory. It is connected to the PEs and the MMU through another timed multibus. Its specific HAL enables to send configuration, execution, preemption or consumption mode switch orders. It can also be interrupted by any PE to be informed of a task execution end. The MMU also interrupts the controller whenever a task is blocked (or no longer blocked) waiting for input data or output space, as well as when a buffer threshold is crossed. We did not set the number of PE so as to study how our scheduling algorihtm can cope with different processor loads.

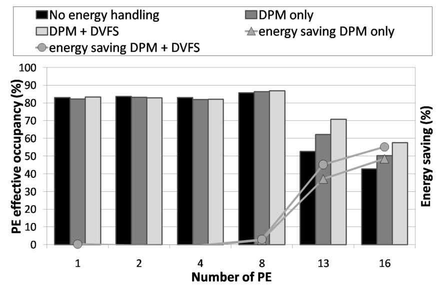
To evaluate our algorithm impact on a streaming application, we used a well-known telecommunication application: a WCDMA encoder/decoder [15]. The application was pipelined and implemented on the simulated target MPSoC. The WCDMA application integrates an encoder followed by a decoder and is consequently built of 13 tasks. This allows having more tasks than resources on the SCMP platform to stress the potential scheduling anomalies. This application is characterized by an unbalanced pipeline whose slowest tasks are the FIR filters. In addition dynamism, is found in the task execution length as pilot frame get processed instead of actual data.

5 Results

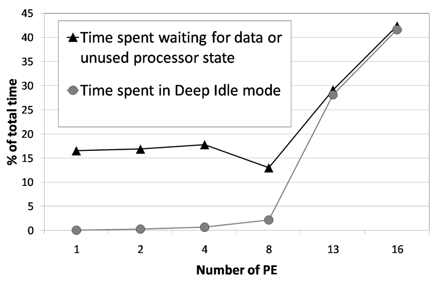
To study the impact of our scheduling algorithm we chose to compare it to two simpler versions of the algorithm. The first version does not handle power issues.



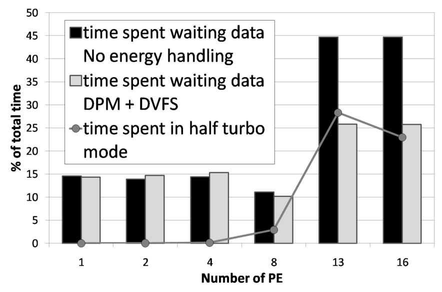
(a)



(b)



(c)



(d)

Fig. 3. Figure (a),(b),(c) and (d) were obtained with the same WCDMA application sending 256 frames. The communication buffers were 8-frame long and had a higher threshold identical to the lower one and set to 2 frames. (a) Total execution time for the WCDMA application in function of the number of processing resources and the scheduling algorithm used; execution time overhead of our solution compared to the *no energy handling* algorithm. (b) Total processor effective occupancy and energy saving in function of the number of processing resources and the scheduling algorithm used; (c) Average time spent in Deep Idle mode compared to the time spent in unused state or waiting for data for a processor when using our proposed algorithm; (d) Comparison of the average time a processor spends waiting for data in the case of the no power saving algorithm and of our solution (DPM+DVFS): influence of the Half-Turbo mode usage on blocking states.

It simply schedules tasks relying on pipeline stage position and blocked states. All processor are kept in *Turbo* mode. It is referred as the *no energy handling* scheduling. The second version is called *DPM-only* scheduling. This corresponds to a naive power-aware approach. Here unused resources and resources executing blocked tasks are put to *Deep Idle* mode. Finally our proposed algorithm will be referred as *DPM + DVFS* scheduling.

As shown in figure 3(a) the total execution time of the WCDMA application is not affected by our scheduling algorithm no matter how many processing resources there are. The variation in execution time is always maintained below 1.2%. In addition our algorithm allowed a good acceleration of the processing for streaming applications.

While we managed to maintain the execution time of the scheduling without energy awareness, Fig. 3(b) shows that substantial energy savings were made. As soon as processor effective occupancy drops it is directly compensated by our power saving method. With 13 processors we reduced the power consumption by 45%. In addition, our method obtains better results than the *DPM-only* scheduling which only reaches 37% energy saving in that case.

Fig. 3(c) illustrates how our scheduling algorithm uses the DPM mode in a real application case. The figure shows that when processors spend little time waiting for data or in unused state (below 17%), the *Deep Idle* mode is seldom used. When the wasted time increases the DPM usage curve follows the unused or blocked processor curve as planned. In fact, when the number of processing elements is little, there is often another task ready to be executed immediately. For low PE numbers the wasted time corresponds to the control overhead. The controller lacks reactivity to reach higher computing performance or power saving.

Finally Fig. 3(d) studies the impact of DVFS modes usage on the application execution. We compare the execution of our algorithm to the *no energy handling* scheduling. The analysis shows that when DVFS mode are used they drastically reduce the amount of time spent in blocking states (42% reduction for 13 processors). Thus, our algorithm succeeds to balance the streaming pipeline stage execution length efficiently when the processor usage drops. As a result the processor load is increased with our algorithm compared to the *no energy handling* scheduling as shows Fig. 3(b).

6 Conclusion

In this paper we presented a new power-aware scheduling algorithm for pipelined application in MPSoC environments. The algorithm was implemented on a virtual MPSoC platform simulated with the SESAM environment. Substantial energy consumption gain was made compared to a classic data dependency scheduling that only takes into account blocking states. For a WCDMA application executing on a platform with 13 PE our scheduling algorithm reduced the processing resources power consumption by 45%. In addition the use of DVFS and DPM did not impact the application execution speed. The variation in execution speed were maintained below 2%. Moreover, our algorithm succeeded to maintain a high processor load. As a result, our algorithm allows a good acceleration of the execution speed of streaming applications in MPSoCs while efficiently managing power consumption issues through the use of DVFS and DPM capabilities. In addition, as our algorithm is fully online and can handle the scheduling of more tasks than processor, we can manually shut down some processing resources to lower the power budget while guaranteeing a correct execution.

7 Acknowledgements

Part of the research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement no. 100029.

References

1. V. Venkatachalam and M. Franz. Power Reduction Techniques For Microprocessor Systems. *ACM Computing Surveys (CSUR)*, 37(3):195–237, 2005.
2. *Intel PXA27x Processor Family, Electrical, Mechanical, and Thermal Specification*, 2005.
3. M. L. Dertouzos and A. K. Mok. Multiprocessor Online Scheduling of Hard-Real-Time Tasks. *IEEE Transactions on Software Engineering*, 15(12):1497–1506, 1989.
4. L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation, Scheduling and Voltage Scaling on Energy Aware MPSoCs. In *Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pages 44–58, 2006.
5. Y.-H. Lu, L. Benini, and G. De Micheli. Dynamic Frequency Scaling with Buffer Insertion for Mixed Workloads. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):1284–1305, 2002.
6. N. Pettis, L. Cai, and Y.-H. Lu. Statistically Optimal Dynamic Power Management for Streaming Data. *IEEE Transactions on Computers*, 55(7):800–814, 2006.
7. K. H. Kim, R. Buyya, and J. Kim. Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters. In *IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 541–548, 2007.
8. F. Zhang and S. T. Chanson. Power-Aware Processor Scheduling under Average Delay Constraints. In *IEEE Real Time on Embedded Technology and Applications Symposium (RTAS)*, pages 202–212, 2005.
9. P. Choudhury, P. P. Chakrabarti, and R. Kumar. Online Dynamic Voltage Scaling using Task Graph Mapping Analysis for Multiprocessors. In *International Conference on VLSI Design (VLSID)*, pages 89–94, 2007.
10. S. Hua, G. Qu, and S. S. Bhattacharyya. Energy-Efficient Embedded Software Implementation on Multiprocessor System-on-Chip with Multiple Voltages. *ACM Transactions on Embedded Computing Systems (TECS)*, 5(2):321–341, 2006.
11. F. Zhang and S. T. Chanson. Blocking-Aware Processor Voltage Scheduling for Real-Time Tasks. *ACM TECS*, 3(2):307–335, 2004.
12. C. Im, H. Kim, and S. Ha. Dynamic Voltage Scheduling Technique for Low-Power Multimedia Applications Using Buffers. In *ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 34–39, 2001.
13. N. Ventroux, A. Guerre, T. Sassolas, L. Moutaoukil, C. Bechara, and R. David. SESAM: an MPSoC Simulation Environment for Dynamic Application Processing. In *IEEE International Conference on Embedded Software and Systems (ICESS)*, 2010.
14. A. Guerre, N. Ventroux, R. David, and A. Merigot. Approximate-Timed Transactional Level Modeling for MPSoC Exploration: A Network-on-Chip Case Study. In *IEEE Euromicro Symposium on Digital Systems Design (DSD)*, pages 390–397, 2009.
15. A. Richardson. *WCDMA Design Handbook.* 2006.