Low Power Process Assignment for Distributed Embedded Systems using Dynamic Voltage Scaling

Marcus T. Schmitz

Bashir M. Al-Hashimi[*]

Abstract

This paper presents an efficient algorithm for voltage scaling of an distributed embedded system taking communicating processes into account. The algorithm finds scaled voltages for each processes without restricting the applicable voltage levels apriori. In addition the algorithm is not limited by a fixed power consumption among processes. Furthermore we show the importance of a process optimisation which is optimised for the dynamic voltage scaling (DVS) technique. Various examples from the literature and randomly generate show the efficiency of the proposed scaling algorithm and the DVS optimised process assignment.

1 Introduction

Energy efficient design of embedded systems is essential due to various reasons. The increasing popularity of mobile applications, like cellular phones, PDAs, etc., forces system designers into the low power domain to extend the operational time of these portable systems limited by the capacity of batteries. In addition low power designs have become necessary to cope with environmental problems like global warming. Reducing for example the power dissipation of an commonly used automotive embedded system will eventually reduce CO_2 ejection.

Embedded systems are often build out of multiple processing elements (PEs) like general purpose processors, application specific instruction processors (ASIPs) and application specific integrated circuits (ASICs) to reduce the cost and the development time. This is because off the shelf components are often cheaper then ASICs and software is fast developed then hardware. But one drawback of programmable processors is their power inefficiency compared to ASICs which can be found in the region of 1 to 2 orders of magnitude [1]. Therefore various techniques have been proposed to reduce the power consumption of these components like dynamic power management (DPM) [2] and dynamic voltage scaling (DVS) [3], which either switch off components or scale their performance down, respectively.

Due to the dynamic energy reduction superiority of DVS over shutdown techniques [3] voltage scaling is receiving noticeable attention from the research community over the last few years. Starting from functional unit level voltage scheduling techniques [4, 5] to system-level scheduling techniques [6, 3, 7, 8, 9, 10] and also implementations of DVS processors which have shown the efficiency of this technique [11, 9].

[*]Marcus T. Schmitz and Bashir M. Al-Hashimi are with the Electronic Systems Design Group, Department of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, United Kingdom.

Many of these system-level scheduling approaches, which are of direct relevance to our research, are based on three major assumptions: a) the power dissipation is constant for all processes running on processing elements, b) the system is restricted to one DVS processor and c) the processes running on the processor are independent of each other.

These assumptions become unrealistic in modern distributed embedded system designs where processes running on multiple PEs communicate over different communication links. Furthermore modern PEs make use of power reduction techniques like gated clocks resulting in varying power consumptions depending on the computation carried out on them.

This paper describes a new voltage scaling technique for processes executing on a distributed embedded real-time system build out of DVS PEs (see figure 2). Moreover we investigate an energy optimised process assignment based on the proposed scaling algorithm.

2 Preliminaries

In this section we describe the system and the power model used in this paper. We also illustrate the importance of a DVS optimised process assignment by using a motivational example.

2.1 System model

Our technique operates on an embedded system specification which is given by a task graph and a fixed system architecture including DVS enabled PEs.

Task graph:

The functionality of a system can be described using task graphs (Figure 1) in which each node and edge represent a process and a data dependency, respectively. A process is a collection of computations associated with an execution time and dynamic power dissipation on each PE while each dependency inherits an amount of data which has to be transfered between the connected nodes. The period of the task graph describes the maximal time between two consecutive invocations of its source nodes. Each sink node has a deadline by which it has to finish its execution otherwise a real-time constraint is violated.

Architecture:

Each PE in the architecture is either a processor or an ASIC able to execute processes. ASICs are limited by the number of processes they can accommodate whereas processors are not. These elements are connected by communication links such that there is a direct link between each two PEs. The sending and receiving of data takes place over communication interfaces which are able to adapt the different operation frequen-

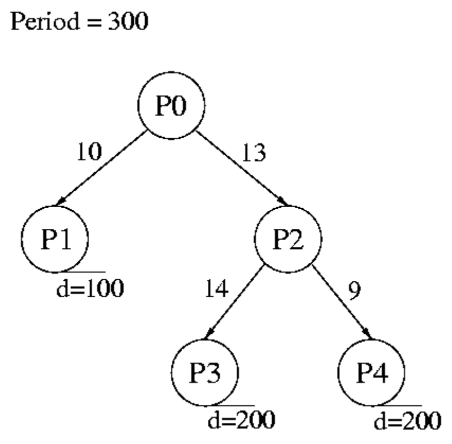

Figure 1: Multiple deadline task graph

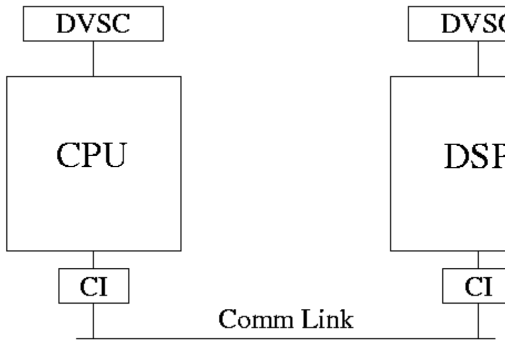

Figure 2: DVS distributed system

cies. To overcome the power inefficiency of processors they are assumed to be DVS controllable. An example architecture can be seen in Figure 2.

2.2 Power model

The major part of the power dissipated by digital circuits can be describe by the dynamic power equation (neglecting short circuit power), $P_{Dyn} = C \cdot N_{0\to 1} \cdot f \cdot V_{dd}^2$. Where C is the load capacitance, $N_{0\to 1}$ is the 0 to 1 switching activity, V_{dd} is the supply voltage and f is the operational frequency. It is obvious that reducing V_{dd} will reduce power in a quadratic manner but at the same time it will increase the circuit delay which is given by $d \propto V_{dd}/(V_{dd}-V_t)^2$. Where V_t is the threshold voltage of the circuit under which no operation is possible. Out of these equations we can derive the circuit speed S and dynamic power dissipation P at V_{dd}.

$$S(V_{dd}) = \frac{V_{max}}{(V_{max}-V_t)^2} \cdot \frac{(V_{dd}-V_t)^2}{V_{dd}} \quad (1)$$

$$P(V_{dd}) = \frac{V_{dd} \cdot (V_{dd}-V_t)^2}{V_{max} \cdot (V_{max}-V_t)^2} \cdot P(V_{max}) \quad (2)$$

These equations lead the the normalised energy versus speed function [10] which is plotted in figure 3.

2.3 Motivational example

We first outline the importance of an DVS optimised process assignment and give more information about the DVS scaling algorithm in section 3. To illustrate the impact of an DVS optimised process assignment on the energy consumption consider

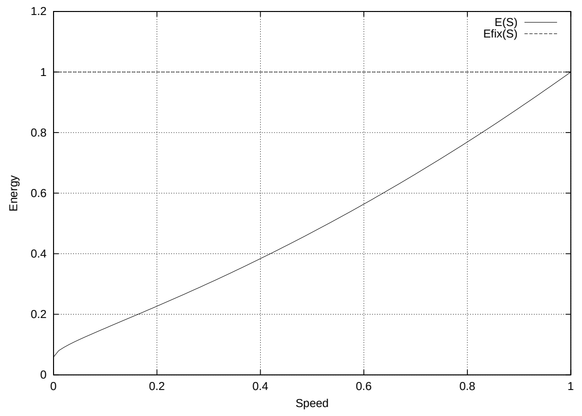

Figure 3: Energy versus speed function

the following example. The task graph of five processes $P0$ to $P1$ shown in figure 1 needs to be assigned to the architecture depictured in figure 2 such that the overall energy dissipation is minimised and no deadline d is violated. Table 1 gives the necessary information about the PEs, the processes and the communication links in terms of execution time t and power dissipation P at V_{max}. A method which is not aware of DVS would only consider the dynamic power of each process at maximal supply voltage, V_{max}, and therefore it would try to distribute the processes to PEs which are most power efficient for the particular process to minimise the value of the cost function. In our case processes 0 and 1 would be executed by the CPU while processes 2, 3, and 4 would run on the DSP. This process assignment results in a energy dissipation $E_{nonDVS} = 4595.7$ if no voltage scaling is applied at the final assignment. Using the DVS algorithm described in section 3 could lower the energy consumption to $E_{DVS} = 3395.7$.

But this assignment can be further improved if one would consider the DVS technique during the optimisation phase. Using the proposed DVS algorithm to calculate the cost to guide the optimisation results in the following process assignment. Process 0, 2, and 3 are moved to the CPU and process 1 and 4 are assigned to the DSP. The result of this assignment is a energy dissipation $E_{nonDVS} = 4923.4$ using no scaling of the PEs and a energy $E_{DVS} = 2719.3$ if the processing elements are scaled. This improves the consumed power by approx. 20% justifying a more costly objective function.

2.4 Voltage scaling problem formulation

The object of the DVS is to minimise the power consumed by all the processes executed by the PEs of the system while meeting all the timing constrains. To formulise the problem we have to define the following sets of processes and edges: P is the set of all N processes p, P_D the set of all deadline processes p_d, PE the set of all processing elements pe and C_p is the set of all ingoing edges c of the process p. t_S and t_E are describing start and end times, respectively and t_d reflects the deadline of a process.

minimise

$$E = \sum_{p=1}^{N} E_p(V_{dd_p}) \quad (3)$$

	P0		*P1*		*P2*		*P3*		*P4*		
	t	P	t	P	t	P	t	P	t	P	P_{stat}
CPU	26.86	35.38	27.79	23.79	58.08	35.08	24.68	30.53	–	–	1.97
DSP	–	–	11.61	20.20	78.84	21.93	15.89	20.79	36.33	21.59	2.27

	t_{aver}	P_{aver}	P_{stat}	*PacketSize*
CommLink	0.3	2.90	2.51	7

Table 1: Process and link characteristics

subject to

$$\forall p_d \in P_D : t_{E_{p_d}} \leq t_{d_{p_d}} \tag{4}$$

$$\forall p \in P : \forall c \in C_p : t_{E_c} \leq t_{S_p} \tag{5}$$

$$\forall pe \in PE : \forall p_{pe} \in P_{PE} : t_{p_{pe}} \leq t_{S_{p_{pe+1}}} \tag{6}$$

Find for all processes of the system a scaled voltage V_{dd_p} such that the energy dissipated is minimised, no deadline processes finishes it execution after its deadline, all the ingoing communications of a processes are finished before its execution starts and all processes on a single PE have no overlapping executions.

3 Proposed algorithm

In this section we describe a algorithm which is able to scale the scheduled processes of an multiple PE architecture by exploiting slack and idle times of the system. The assigned voltages are not fixed apriori, unlike [4, 5, 6, 7]. Also variations in the average power dissipated by each process are permitted, unlike [4, 5, 6, 3]. In addition non of the previous approaches considered distributed PEs. The algorithm takes advantage of the fact that reducing the voltage also improves the power dissipation, similar to the approach presented in [10]. The pseudo code of the algorithm is shown in figure 4. We explain now the proposed algorithm by going through one iteration of its execution. The algorithm is supplied with the input information given in figure 4 including the step width Δt which is the quantum of time by which a process can be extended during each iteration. The first step of the algorithm is to calculate a ΔE value for each process which is the energy reduction of a process when it is extended by Δt. This is followed by the generation of an *Unblocked* process list which includes all processes which can be extended without violating any deadline. For all the processes in the *Unblocked* list the dependencies on other processes and communications are calculated when these processes are extended by Δt. Now the process with the highest ΔE improvement in the *Unblocked* list is scaled and its ΔE value is updated. Since the other processes and communications might be influenced we have to change there start and end times. The algorithm removes now all unextendable processes from the *Unblocked* list. If there are unblocked processes left a new iteration is invoked beginning with a new search for the process with the highest ΔE improvement. After no processes are left which can be extended the algorithm terminates and returns the scaled voltages of each process. The overall consumed dynamic energy is calculated in addition which is the value used by the assignment optimisation algorithm.

```
Algorithm: SCALE_VOLTAGES
Input: - for all processes $p \in P$
       - PEAssignment $PEA_p$
       - Start Time $t_S$
       - End Time $t_E$
       - average power $P_p$
       - deadline $t_{dp}$ if a deadline process
       - step width $\Delta t$

Output: - Energy optimised voltages for each process $V_d dp$
        - Dissipated energy

  1. calculate $\Delta E$ for all processes extended by $\Delta t$
  2. initialise Unblock list with $\Delta t$ extendable processes
  3. set $\Delta t$ dependencies on other processes and communications
  4. while Unblock processes left
       (a) Find processes with highest $\Delta E$
       (b) scale the found process
       (c) update $\Delta E$ of the found process
       (d) update all other influence processes and communications
       (e) remove all unextendable processes from the Unblock list
```

Figure 4: Proposed DVS algorithm for distributed embedded systems

4 Experimental Results

To demonstrate the efficiency of the proposed technique to reduce the energy consumption of a distributed embedded system we have carried out a number of experiments. Five task graphs, three randomly generated (*ms1*,*ms2*,*ms3*) and two taken from the literature [12] (*hou1*, *hou2*), have been assigned to random fixed architectures.

We have chosen a genetic algorithm to optimise the process assignment leading to satisfactory results. The results are shown in table 2. Considering the lowest complexity example *ms3* which is identical to the task graph given in figure 1 and the architecture shown in figure 2 shows the improvements which can be achieved. A process assignment only considering the dynamic power dissipation at V_{max} would optimise the value in the fourth column and results in $E_{nonDVS} = 4595.7$. If this assignment would be scaled using the proposed scaling algorithm

		non DVS optimised assignment			*DVS optimised assignment*		
Name	*Process #*	*DVS scaled*	*unscaled*	*time*	*DVS scaled*	*unscaled*	*time*
ms3	5	3395.7	4595.7	0.01s	2719.3	4923.4	0.01s
ms2	36	6539.6	11577.0	1.67s	4359.1	13477.6	77.11s
ms1	72	54214.2	72955.6	100.45s	50539.5	88805.6	8550.01s
hou1	20	12033.6	13722.2	0.16s	10604.3	13199.0	0.29s
hou2	20	53025.6	68419.4	18.44s	46852.0	77593.4	1018.54s

Table 2: Energy dissipation of DVS optimised and non DVS optimised process assignments

the energy dissipation could be lowered to $E_{DVS} = 3395.7$. The column *DVS optimised assignment* shows the results obtained if the DVS technique is used throughout the proposed assignment optimisation to guide the genetic algorithm. Although E_{nonDVS} increases to 4923.4 the energy dissipation using DVS is further reduced to $E_{DVS} = 2719.3$, improving the energy consumption approximately about 20%. We have also verified the efficiency of the proposed DVS optimised process assignment using more complex examples. In all cases the proposed method was able to improve the energy dissipation between 6 and 33%, which shows clearly the advantage of the DVS optimised assignment. Due to the increase complexity ($O(N^2)$) of the cost function which is used by the genetic algorithm to guide the optimisation the synthesis time increases in a polynomial manner with the number of processes in the hyperperiod task graph.

5 Conclusion and future work

This paper has presented an efficient algorithm to solve the DVS scheduling problem of an distributed embedded system. Variations in the power dissipation among processes and inter process communications are consider. Furthermore we have shown the importance of considering DVS as part of the process assignment step of an distributed embedded system to further lower the energy consumption. The proposed technique was able to reduce the energy of an DVS enabled embedded system. Working progress will investigate the allocation step to determine the number and types of PEs used in the system. Also a more advanced DVS scheduling heuristic could further improve the results.

Acknowledgements

The authors wish to acknowledge the financial support of EPSRC in this project.

References

[1] Thomas D. Burd and Robert W. Brodersen. Processor Design for Portable Systems. *VLSI Journal of Signal Processing*, 13(2):203–222, August 1996.

[2] Luca Benini and Giovanni De Micheli. System-Level Power Optimizatin: Techniques and Tools. *ACM Transcations on Design Automation of Electronic Systems*, 5(2):115–192, April 2000.

[3] Inki Hong, Darko Kirovski, Gang Qu, Miodrag Potkonjak, and Mani B. Srivastava. Power Optimization of Variable-Voltage Core-Based Systems. *IEEE Trans. Computer-Aided Design*, 18(12):1702–1714, Dec 1999.

[4] Salil Raje and Majid Sarrafzadeh. Variable Voltage Scheduling. In *Proceedings 1995 International Symposium on Low Power Design*, pages 9–14, Apr 1995.

[5] Jui-Ming Chang and Massoud Pedram. Energy Minimization Using Multiple Supply Voltages. *IEEE Trans. on VLSI Systems*, 5(4):436–443, Dec 1997.

[6] Yann-Hang Lee and C.M. Krishna. Voltage-Clock Scaling for Low Energy Consumption in Real-Time Embedded Systems. In *Sixth International Conference on Real-Time Computing Systems and Applications*, 1998.

[7] Tohru Ishihara and Hiroto Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proceedings 1998 International Symposium on Low Power Design*, pages 197–202, 1998.

[8] Youngsoo Shin and Kiyoung Choi. Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems. In *Proceedings of the 36th Design Automation Conference*, pages 134–139, 1999.

[9] Trevor Pering, Thomas D. Burd, and Robert B. Brodersen. Voltage Scheduling in the lpARM Microprocessor System. In *Proceedings 2000 International Symposium on Low Power Design*, pages 96–101, June 2000.

[10] M. T. Schmitz and B. M. Al-Hashimi. Energy Minimisation for Processor Cores using Variable Supply Voltages. In *IEE Systems on a chip Workshop, Cork*, pages 10/1–10/4, Sept 2000.

[11] Vadim Gutnik and Anantha Chandrakasan. Embedded Power Supply for Low-Power DSP. *IEEE Trans. on VLSI Systems*, 5(4), 425–435 1997.

[12] Junwei Hou and Wayne Wolf. Process Partitioning for Distributed Embedded Systems. In *Proceedings of 4th International Workshop on Hardware/Software Codesign, Codes/CASHE'96*, pages 70 – 76, March 1996.