Energy Minimisation for Processor Cores using Variable Supply Voltages

Marcus T. Schmitz and Bashir M. Al-Hashimi

Department of Electronics and Computer Science
University of Southampton
Southampton, SO 17 1BJ, UK
{ms99r, bmah}@ecs.soton.ac.uk

Abstract

This paper presents an efficient technique for energy minimisation based on dynamically variable supply voltages. The efficiency is achieved by considering that power consumed by the core depends on the computation carried out on it. Energy is further minimised by allowing the voltage supply to vary continuously. The computational delay introduced due to supply voltage reduction is tolerated by exploiting the idle time while meeting the system time constraint. The energy minimisation problem is formulated and solved using a iterative improvement algorithm. Various examples with different task set complexities show the efficiency of the proposed technique.

1 Introduction

Reducing energy dissipation of embedded systems has emerged as a major task in system design. Portable devices (e.g. cellular phones, notebooks, PDAs etc.) draw energy from batteries with limited capacity which specifies the operational time between recharge phases. Extending these intervals increases the product value to the customer — an important purchasing factor. Faced with problems like global warming also environmental reasons are forcing more and more the design of low power consuming devices, such as general purpose computers and automotive embedded systems. Due to cost and power reduction embedded systems are being implemented as a single chip, called system-on-a-chip (SOC). This has become possible through the development of powerful CAD tools and the dramatically improved feature size allowing today more then 100 million transistors on a single die [1]. To cope with such a complexity designers try to adopt a core based system design methodology where the system is composed out of Intellectual Property (IP) cores (microcontrollers, ASICs, DSPs etc.), also called processing elements (PEs).

Recent advances of dc-dc converters [2] have made it possible to vary the supply voltage of computational PEs dynamically during system operation. Thus enabling to tradeoff performance against power dissipation [3]. Variable voltage scheduling has received noticeable attention from the research community over the last few years [4–9]. Most of the above mentioned publications [4–6, 8] consider the power dissipation (switching activity) of the PEs to be constant for each operation which is not the case for many modern VLSI circuits [10] using e.g. gated clocks. In [7] the general variable voltage scheduling problem, allowing variations of power, was formulated as an integer linear programming (ILP) problem under the constraint that possible supply voltages are fixed a priori. The feasibility of this formulation becomes impracticable due to two reasons. First the ever increasing demand of functionality increases the number of tasks describing the behaviour of the system and second it has become possible to implement dc-dc converter on the same chip resulting in the necessity to consider near continuously variations of the supply voltage.

In this paper we investigate the combination of allowing power variations among tasks [7] and not fixing the supply voltages a priori [8] in order to minimise energy of a processing element.

2 Preliminaries

This section gives the background of variable voltage scheduling and illustrates the motivation for the proposed method on an example.

2.1 System model

The system model considers the behaviour as a set of independent executing tasks [11]. All tasks have the same period, otherwise the least common multiple (LCM) theorem is applied [9]. The functionality has to be mapped (allocation and scheduling) to the PEs in such a way that the period (deadline) is met and the overall PE energy consumption is minimised. In our work we assume that the partitioning [12] and the scheduling [11] have already been performed. With each task there is an associated worst case execution time and an average power dissipation [13]. The average power is sufficient since no preemption is allowed.

2.2 Power model

Power dissipation P in a digital circuit [14] is caused by dynamic power P_{dyn} and static power P_{static}.

$$P = \underbrace{C \cdot N_{0\to1} \cdot V_{dd}^2 \cdot f}_{P_{dyn}} + \underbrace{(I_{sc} + I_{leak} + I_{st}) \cdot V_{dd}}_{P_{static}} \qquad (1)$$

where the dynamic power depends on load capacitance C, 0 to 1 switching activity $N_{0\to1}$, supply voltage V_{dd} and the frequency f at which the system runs. Static power is caused by the short circuit current I_{sc}, the leakage current I_{leak} and the static current I_{st} and is often considered to be negligible compared to the dynamic power. From *here* on only dynamic power is considered. It is evident that V_{dd} reduction lowers the power consumption, but unfortunately it increases the circuit delay d, which is inverse proportional to the operation frequency:

$$d = k \cdot \frac{V_{dd}}{(V_{dd} - V_t)^2} \qquad (2)$$

where k is a constant and V_t is the threshold voltage under which the circuit does not operate. The system performance S is the ratio of the delay at maximum supply voltage $d(V_{max})$ and the circuit delay at supply voltage $d(V_{dd})$. System performance and power depending on the supply voltage V_{dd} are given by [8]:

$$S(V_{dd}) = \frac{V_{max}}{(V_{max} - V_t)^2} \cdot \frac{(V_{dd} - V_t)^2}{V_{dd}} \quad (3)$$

$$P(V_{dd}) = \frac{V_{dd} \cdot (V_{dd} - V_t)^2}{V_{max} \cdot (V_{max} - V_t)^2} \cdot P(V_{max}) \quad (4)$$

From Equations (3) and (4) it is possible to derive the energy vs. supply voltage $E(V_{dd})$ and energy vs. performance $E(S)$ relationships as:

$$E(V_{dd}) = \frac{P(V_{max}) \cdot t(V_{max})}{V_{max}^2} \cdot V_{dd}^2 \quad (5)$$

$$E(S) = \frac{P(V_{max}) \cdot t(V_{max})}{V_{max}^2} \cdot V_0^2 \cdot \cdot \left(\frac{V_t}{V_0} + \frac{S}{2} + \sqrt{\frac{V_t}{V_o} \cdot S + \left(\frac{S}{2}\right)^2} \right)^2 \quad (6)$$

where $P(V_{max})$ and $t(V_{max})$ are the power dissipation and execution time at maximum supply voltage, respectively and V_0 equals $(V_{max} - V_t)^2 / V_{max}$. Figure 1 shows the normalised energy vs. supply voltage curve, show that it is always better to run a single task at a lower voltage. Note that Equation (6) differs from the one in [3], since we consider the use of gated clocks which shuts down unused resources.

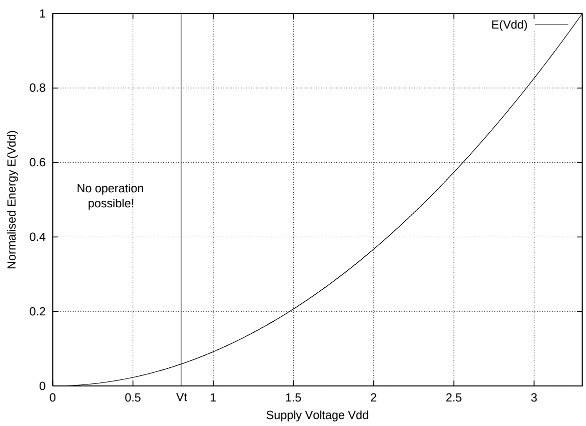

Figure 1: Energy vs. supply voltage ($V_t = 0.8V$)

2.3 Motivational example

To show the main idea of the proposed technique, consider the following example. Three sequential executing tasks T1, T3 and T5 need to be run on a single processing element using 3, 10, and $6s$ respectively, reflected in Table 1 and Figure 2. Assume that V_{max} of the PE is $3.3V$. The system time constraint requires that all the task have to be finished by the deadline $t_d = 22s$. The first part of the example considers a processor model with a fixed power dissipation independent of the computation carried out, so the processor runs constantly at $5W$. Since energy is the product of power and execution time, the total energy consumed by the processor is $5W \cdot 3s + 5W \cdot 10s + 5W \cdot 6s = 95J$, if the processing element

	Fixed power		*Varying power*	
Task	$P(V_{max})$	$t(V_{max})$	$P(V_{max})$	$t(V_{max})$
T1	$5W$	$3s$	$5W$	$3s$
T3	$5W$	$10s$	$2W$	$10s$
T5	$5W$	$6s$	$3W$	$6s$

Table 1: Power and execution time of T1,T3, and T5 when power is fixed and variable during the execution of tasks

is switched off after the computation. Energy dissipated by the PE could be reduced if its supply voltage is lowered at the expense of increased execution time. This increased computational time can be tolerated due to the availability of idle time which evokes out of the difference between required computation time and the time constraint of the application. Here it has been assumed that the time constraint is greater than the execution time of the tasks which is a valid assumption since it is very unlikely that an IP core matches exactly the required performance specification, leading to overdesign. Assuming this, the execution time can be increased from 19

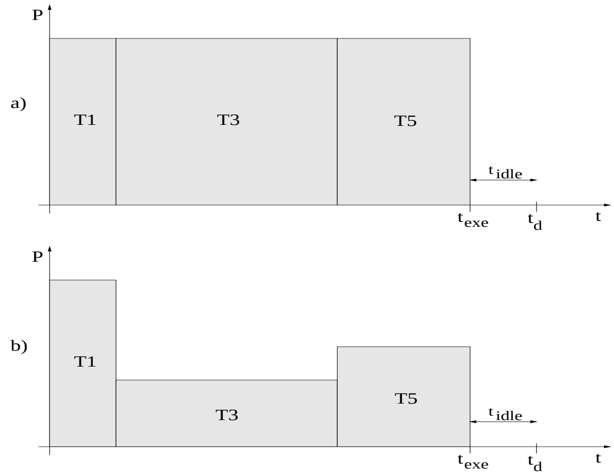

Figure 2: Schedule of tasks on a PE considering the a) fixed power model and the b) variable power model

to $22s$ allowing the voltage supply to be reduced from 3.3 to $3.03V$, computed by Equation (3). From Equation (4) it follows that energy is reduced from 95 to $79.79J$. This example has assumed that the power dissipated by the processor core is independent of the operations involved in the computation, which is only valid for a certain type of processor cores [15]. Whereas the power dissipation of many modern IP cores *does* depend on the computation carried out [10], as shown in Table 1 and Figure 2b) for example. This shows that the power varies among the tasks between 2 to $5W$. Computing the energy consumption under consideration of the varying power model leads to $E = 5W \cdot 3s + 2W \cdot 10s + 3W \cdot 6s = 53J$. Similar to the previous example we could find a single supply voltage such that the deadline is just meet, by extending the execution time from 19 to $22s$, resulting in $V_{dd} = 3.03V$ and $E = 44.52J$. However this is not the optimal solution (minimising energy) since a different but unique supply voltage for each task could further improve the energy consumption which is main aim of the proposed technique. For example: Running the tasks at $V_{dd_1} = 2.79V$, $V_{dd_2} = 3.12V$, and

$V_{dd_3} = 3.01V$ would result in an total energy dissipation of $E = 43.55J$ and still meet the deadline $t_d = 22s$. Note that all the above mentioned results were obtained assuming that the threshold voltage V_t is $0.8V$.

2.4 *Problem formulation*

As given in subsection 2.1 the system is modelled out of independent executing tasks therefore the PEs do not influence each other and the minimised energy consumption E will be achieved when all PEs consume a minimum amount of energy. Hence it is adequate to formulate the optimisation problem for a single PE.

minimise

$$E = \sum_{i=1}^{N} E_i(S_i) \tag{7}$$

subject to

$$\sum_{i=1}^{N} \frac{t_i(V_{max})}{S_i} \leq t_d \tag{8}$$

where N is the number of task.

> *Find for all task of the processing element a performance factor $S_i \in \mathbb{R}^+ : 0 < S_i \leq 1$ such that deadline t_d is not violated and the energy consumption is minimised.*

Given the motivational example in subsection 2.3 and problem formulation above the aim of the following section is to introduce an effective and efficient iterative improvement technique to solve the energy optimisation problem by assigning appropriate voltages to each task under consideration of power variations among tasks.

3 Proposed algorithm

A iterative improvement algorithm for the variable voltage problem is formalised in this section. It assigns voltages to each task in the schedule of the processor core such that the energy is minimised. The selected voltages are *not* fixed a priori, unlike [4–7] and the average power consumption of the processor can vary among the tasks, unlike [4–6, 8]. The algorithm takes advantage of the fact that reducing the supply voltage, i.e. slowing down the PE, results always in better energy consumption (Figure 1). The algorithm is shown in Figure 3. We now explain the algorithm using the example introduced in subsection 2.3. Consider the task set shown in Figure 2b. Obviously, the usable time to optimise the energy is $t_{idle} = t_d - t_{exe} = 3s$. In its first step the algorithm divides this idle time into Q equal time pieces, where Q is the number of iterations used by the algorithm to converge. Its value is set apriori as an input to the algorithm and in this example it is $Q = 12$, resulting in $\Delta t = 0.25s$. In the second step an initial lookup table is generated. This table contains the energy savings ΔE of each task when its execution time is extended by Δt. Asides from that the new execution times have to be stored to allow the updating of ΔE during the iterations. So this initial table would look like Table 2. In each iteration the algorithm goes through three steps shown in step 3 of Figure 3. The first iteration of our example would select task T1, extend it to $3.50s$ and change ΔE to $1.14J$. The loop in step 3 finishes after all Q time pieces of size Δt are distributed among the tasks. In the final step the algorithm calculates the supply voltages for each task out of equation (3) using the execution time at maximal supply voltage $t_i(V_{max})$ and the found extended execution time $t_i(V_{max}) + \sum \Delta t_i$ reduced by Δt.

Algorithm: **SCALE_VOLTAGES**
Input: - task execution times at V_{max} $\boldsymbol{t_i(V_{max})}$
- average task power dissipation at V_{max} $\boldsymbol{P_i(V_{max})}$
- single deadline (period) $\boldsymbol{t_d}$
- number of iterations $\boldsymbol{Q}$

Output: - Energy optimised voltages for each task $\boldsymbol{V_{dd_i}}$

1. calculate $\Delta t = \frac{t_{idle}}{Q}$
2. generate initial lookup table consisting of ΔE_i for all tasks when extended by Δt, additionally the new execution times $t_i(V_{max}) + \Delta t$ are stored
3. $\forall Q$
   (a) find highest ΔE in the lookup table
   (b) give time piece Δt to that task
   (c) update best task lookup table entry (energy and time)
4. $\forall i \in N$ compute V_{dd_i} out of $S_i = \frac{t_i(V_{max})}{t_i(V_{max}) - \Delta t + \sum \Delta t_i}$

Figure 3: Proposed iterative improvment algorithm for task voltage assignment

	Initial		*1. iteration*	
Task	ΔE	$t_i + \Delta t$	ΔE	$t_i + \Delta t$
T1	**1.37J**	**3.25s**	**1.14J**	**3.50s**
T3	0.59J	10.25s	0.59J	10.25s
T5	0.87J	6.25s	0.87J	6.25s

Table 2: Initial and first updated lookup table

4 Experimental Results

To demonstrate the efficiency of the proposed technique in reducing the energy consumption of a PE a number of examples are given. Nine randomly generated task set examples (*t_10_1* to *t_100_3*) ranging from 10 to 100 tasks were generated, resulting in the total execution time and energy consumption at V_{max} as shown in Table 3. Table 4 shows the results of applying the proposed algorithm to these various task sets when the time constraint of the application increase by 10, 20 and 50% with respect to the execution time at V_{max}. As it can be seen that the proposed technique reduces the energy consumption of all task sets. For the task sets example *t_100_1* the energy consumption was reduced from $2815J$ to $2393.23J$ at the expense of increased computation time from $560s$ to $616s$ which can be tolerated due to the system specified time constraint $t_d = 560s + 10\% = 616s$. The computed supply voltages range was 2.79 to $3.3V$. The energy could be further improved by relaxing the time constraint to $t_d = 560 + 50\% = 840s$ resulting in $E = 1635.97J$ and a supply voltage range from 2.24 to $3.3V$. The proposed iterative

Name	*No. of tasks*	*Exec. Time*	*Energy*
t_10_1	10	51	263
t_10_2	10	86	724
t_10_3	10	82	148
t_50_1	50	292	1759
t_50_2	50	533	5645
t_50_3	50	379	774
t_100_1	100	560	2815
t_100_2	100	1081	12236
t_100_3	100	744	1559

Table 3: Evaluation task sets at V_{max}

Name	$E(t_d = 10\%)$	$E(t_d = 20\%)$	$E(t_d = 50\%)$
t_10_1	226.57	201.66	156.64
t_10_2	629.02	558.46	434.78
t_10_3	127.52	114.25	88.98
t_50_1	1503.09	1333.48	1025.08
t_50_2	4847.74	4322.84	3329.15
t_50_3	672.27	601.00	468.52
t_100_1	2393.23	2124.49	1635.97
t_100_2	10486.05	9347.34	7240.25
t_100_3	1364.32	1220.15	949.83

Table 4: Energy optimised results after running the proposed algorithm with 1 million iterations on the evaluation task sets

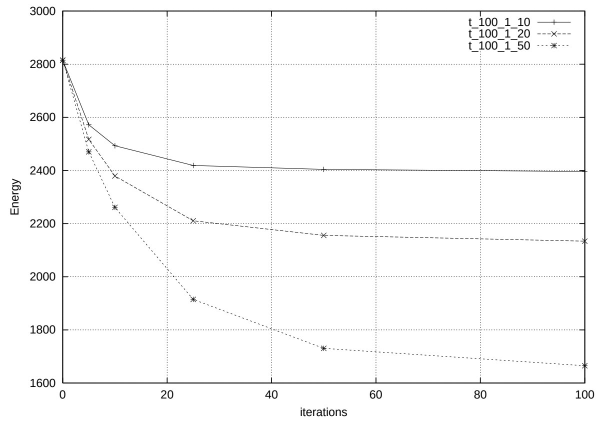

Figure 4: Convergence of the proposed algorithm shown on task set *t_100_1* when the idle time is 10%, 20% and 50% of the execution time

improvement algorithm has a time complexity polynomial to the number of tasks N, while its space complexity is linear. Figure 4 shows that the proposed algorithm converges to a near optimal solution in a small amount of iterations as the energy values approaches a constant value. Here the number of iterations reflects the computational time of the algorithm. On a PentiumIII/500MHz PC running Linux each iteration took on average 1.47 to 4.59ns for 10 and 100 tasks, respectively. Running the proposed algorithm with 1000 iterations (execution time $< 5ms$) resulted for all given task sets in solution within a 1% margin with respect to the results shown in Table 4.

5 Conclusion and future work

This paper has presented an efficient technique to solve the variable voltage energy minimisation problem under consideration of variations in the power dissipation depending on task. Furthermore the possibility of continuously variable voltages were exploited to reduce the energy consumption. It was shown that energy can be reduced by 12 to 42% if the idle time varies between 10 and 50% of the execution time.

Working progress will investigate the feasibility of scheduling algorithm for different system models.

Acknowledgements

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

References

[1] *International Technology Roadmap for Semiconductors.* http://notes.sematech.org/ntrs/PublNTRS.nsf.

[2] W. Namgoong, M. Yu, and T. Meng. A high-efficiency variable-voltage cmos dynamic dc-dc switching regulator. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 380–381, 1997.

[3] Vadim Gutnik and Anantha Chandrakasan. Embedded power supply for low-power dsp. *IEEE Trans. on VLSI Systems*, 1997.

[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), 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*, 1999.

[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] 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.

[9] 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.

[10] Vivek Tiwari, Sharad Malik, and Andrew Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Trans. on VLSI Systems*, Dec 1994.

[11] Alan Burns and Andy Wellings. *Real-Time Systems and Programming Languages.* Addison-Wesley, 1989.

[12] F. Vahid, D.D. Gajski, and J. Gong. A Binary-Constraint Search Algorithm for Minimizing Hardware During Hardware/Software Partitioning. In *Proc. European Design Automation Conference (EURO-DAC94)*, pages 214–219, Grenoble, France, September 1994. IEEE Computer Society, IEEE Computer Society Press.

[13] Darko Kirovski and Miodrag Potkonjak. System-level synthesis of low-power hard real-time systems. In *Proceedings of the 34th Design Automation Conference*, pages 697–702, 1997.

[14] Anantha P. Chandrakasan and Robert W. Brodersen. *Low Power Digital CMOS Design.* Kluwer Academic Publisher, 1995.

[15] V. Tiwari and M. Lee. Power analysis of a 32-bit embedded microcontroller. In *Asia and South Pacific Design Automation Conference*, pages 141–148, Aug 1995.