A DVS System Based on the Trade-off Between Energy Savings and Execution Time

M. Vasić, O. García, J.A. Oliver, P. Alou, J.A. Cobos
Universidad Politécnica de Madrid (UPM), Centro de Electrónica Industrial (CEI)
José Gutiérrez Abascal 2, 28006 Madrid, Spain
miroslav.vasic@upm.es

***Abstract*-DVS (Dynamic Voltage Scaling) is a technique used for reducing the power consumption of digital circuits. The power consumed by these circuits has a main component (dynamic power) that is proportional to the square of the supply voltage. Additionally, for every supply voltage, there is a maximum value of the clock frequency. The advantage of using DVS is that the supply voltage (and hence clock frequency) can be adjusted depending on the specific needs during execution. The DVS concept has been used in some commercial products like Transmeta's Crusoe [1], Intel Speed Step [2], AMD K6 [3], Hitachi SH4 [4], etc.**

This paper presents results obtained by using a DVS algorithm based on the workload estimation and trade-off between the execution time and power savings. It is discussed about influence of the power supply's slew rate, algorithms influence on the system performance and problems to estimate the processors workload. The DVS system is realized on Intel's PXA255 platform and energy savings have been calculated by measuring directly voltages and currents on the platform.

I. INTRODUCTION

One of the biggest concerns for the designers of portable devices is their autonomy. The autonomy greatly depends on the battery life, making the problem of device's low power consumption one of the most important ones. Low power is one of the concerns for much bigger systems as well, e.g. data centers where low power consumption reduces the cost on packages and heat sinks and its size and increases the circuit's reliability. The present technology for microprocessors and digital circuits is CMOS, and there can be distinguished three different mechanisms of power losses. The first one is due to the leakage that is present in these circuits, and this mechanism of power losses is getting more on the importance as the width of the channel of CMOS transistors decreases [5]. The next mechanism of power losses occurs every time when a CMOS changes its output value. For short period both transistors conduct and, therefore, there is a low resistance path from the power supply to the ground of the chip. Losses due to short circuit current are produced during this short period. The last mechanism (dynamic losses) is due to the parasitic capacitance at the output of each CMOS gate. Every change of the voltage on the output of one CMOS circuit means charging or discharging this capacitance, therefore changing its stored energy. These changes happen with a frequency of the system's clock, hence the power losses, in this case, depends on frequency lineally and on supply voltage as a quadratic function (proportional to the energy of the parasitic capacitor). This dependence can be presented as [6]:

$$P \propto CV_{DD}^2 f \qquad (1)$$

where *Vdd* is supply voltage, *f* system's clock frequency and *C* is the equivalent capacitance on the chip. The first two mechanisms depend on the supply voltage lineally, and they do not have such a great impact on total power loss as the last one. Due to this, it is possible to significantly decrease dynamic losses, and on that way the total losses as well, by decreasing the supply voltage. However, due to time delays of digital circuits, there is a correlation between the minimum supply voltage and the maximum clock frequency of the system [6]:

$$t_d = \mathrm{k}\frac{\mathrm{V_{DD}}}{(\mathrm{V_{DD}} - V_{th})^{\gamma}} \propto \frac{1}{\mathrm{f_{MAX}}} \qquad (2)$$

where *k*, V_{DD} and γ are constants that depend on the CMOS technology of the circuit. Hence, for each value of the supply voltage there is a maximum frequency of the system clock that guarantees correct operations of a digital system.

One of the techniques that are recently used to reduce consumed energy of microprocessors is Dynamic Voltage Scaling (DVS). DVS is a technique that offers adjustment of the voltage and the system's clock frequency depending on the task requirements during the execution time. Depending on that, whether the DVS technique is applied to a Real Time system or not, there can be distinguished two approaches to this idea. In Figure 1 it can be seen how the DVS concept is applied in a Real Time system. Each idle time of the processor is used in order to prolong execution of the active tasks, but again, the moment when every task has to begin is strictly obeyed. The task which is active in this way use "just enough" energy in order to finish its activities before the next one starts. By applying DVS in this way the system's performance will no be reduced.

Other solution is to decrease speed for all tasks, and it is based on the trade-off between the power savings and execution time, Figure 2. By using lower frequencies of the CPU clock, and therefore lower supply voltages, application's time will increase, but the energy consumption will be lower. Naturally, DVS applied on this way cannot be used in a Real Time system, due to time constraints that are not satisfied.

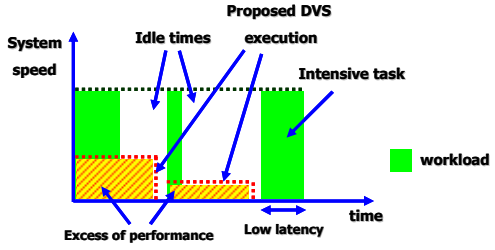

Fig. 1. DVS using CPU's idle time.

978-1-4244-2551-8/08/$20.00 ©2008 IEEE

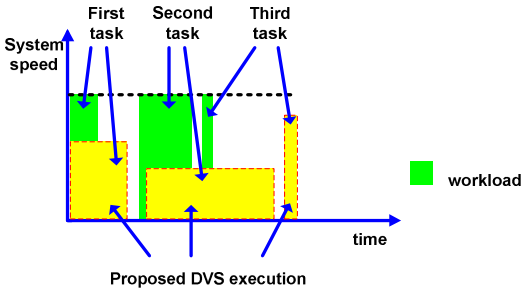

Fig.2. DVS by slowing down the running processes.

TABLE I
OVERVIEW OF POSSIBLE FREQUENCIES AND VOLTAGES IN THE SYSTEM.

	f_{CPU} (MHz) core	f_{INT} (MHz) internal bus	f_{EXT} (MHz) external bus	Core voltage (V)
f1	100	50	100	0.85
f2	200	100	100	1
f3	300	100	100	1.1
f4	400	200	100	1.3

DVS concept has been used in some commercial products like Transmeta's Crusoe [1], Intel's Speed Step [2], AMD K6 [3], Hitachi SH4 [4]. Depending on the type of application and the load that is processed, power savings can vary from 20% to 80% [7].

The focus of this paper is to provide information about the bottlenecks in a DVS system, to clarify how the system performance depends on the power supply, processor's phase locked loop (PLL) and DVS algorithm.

For the needs of testing a DVS system has been implemented with an algorithm which is based on the trade-off between the execution time and consumed energy.

II. SYSTEM DESCRIPTION

The hardware setup consists of an Intel's XScale PXA255 microprocessor. It can operate from 100MHz to 400MHz with a corresponding core supply voltage from 0.85V to 1.3V. The microprocessor has an internal and an external bus and both are used for communication with its external memory. Table 1 shows the frequency and voltage combinations for this system.

This microprocessor can change the working frequency by changing the value of Core Clock Configuration Register (CCCR) [8]. The supply voltage can be changed by setting the appropriate reference of specialized chip MAX1702. The chip contains three DC/DC converters that generate output voltages for the core, flash memory and I/O pins. The voltage reference is set by the microprocessor through the D/A converter, LTC1659. The block diagram of this part of the system is shown in Figure 3. At first glance, this could be a bottleneck of the implemented DVS system, because the voltage changes cannot have high dynamic, due to the time of the conversion needed by the D/A converter and high dynamics is what is needed by a DVS systems. Moreover, the power supply on the board we used was made in such a way that changes of the reference must be done in steps [9]. Due to fast changes of the reference signal, the DC/DC converter that supplies the core loses its control. All these restrictions lead to slow changes of the core's voltage. Figure 4 presents one voltage transition form 1V to 1.25V done in 13 steps. The size of the step is directly controlled inside the operating system.

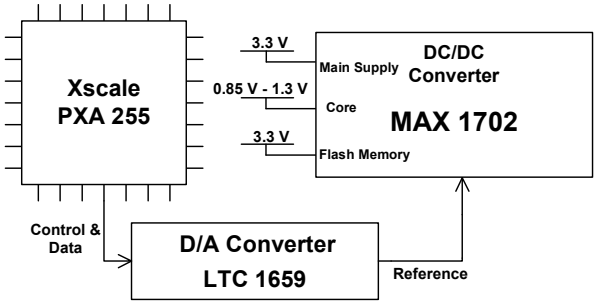

Figure 3. Block scheme of the system.

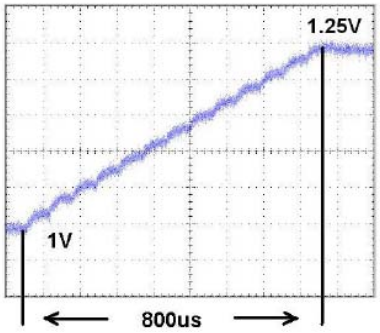

Figure 4. Transition of the core's voltage from 1V to 1.25V.

In the majority of the DVS algorithms, there is assumption that the time needed to perform the voltage and frequency change is negligible and that it does not decrease performance of the system. However, the measurements of the voltage transitions showed that from the processor's point of view this time might be too long, and that it could decrease the overall performance. Due to this, the low slew rate of the power supply was meant to be improved by using a very high dynamics DC/DC converter [10].

The selected operating system was Linux 2.6.13 with RTCore extension. The DVS algorithm was implemented as a real-time module of RTCore and acts as a periodic task. The used operating system is preemptive, i.e. the process with the highest level of priority is the active one. Thus, any process with higher priority can preempted the tested application. In order to facilitate the measurement of the power consumption and to be sure that the measured power is used by the tested application, and not by the other active applications, a little adaptation of the operating system was done.

The application that is tested is recognized directly in the scheduler of the operating system. When the scheduler marks this application as the active, one of the output pins is set to logical 1. Therefore, by measuring the time intervals when this pin has "active" value and at the same time the power consumed by the CPU the execution time and the energy consumed by the tested application can be determined. The measurement of the energy consumed by the DVS algorithm is done in the similar way. Using this method it is not necessary to know the priority level of tested application in order to measure its energy consumption, as the control signals show us when the tested application is active.

III. IMPLEMENTED ALGORITHM

The algorithm, which will make decisions about the necessary voltage and frequency, is the most important part of the system. The proposed algorithm is based on the decomposition of the CPU's work in workload on-chip and off-chip in order to control the execution time of the running application. The idea about the decomposition about the microprocessor's workload is presented in [7] and [11]. The workload can be presented as a sum of on-chip workload ($W_{on\text{-}chip}$) and off-chip workload ($W_{off\text{-}chip}$). On-chip workload is the number of CPU clock cycles needed to perform instructions which are executed inside the CPU only, and, on the other hand, off-chip workload is the number of external clock cycles needed to perform off-chip accesses (to fetch data from external memory).

Knowing the application's workload, clock and bus frequency, as well, it is possible to estimate the execution time of the running task. Hence, the problem is to estimate the workloads. Intel's family of XScale processors has a special Performance Monitoring Unit (PMU) that can monitor different CPU events as number of cache misses, number of executed instructions, number of CPU stall cycles and number of clock cycles. Using these variables it is possible to know in every moment the application's number of Stall cycles per Instruction (SPI), number of Data cache misses per Instruction (DPI) and number of Cycles per Instruction (CPI). Using the linear dependency between CPI and SPI (Figure 5), and the information about DPI, the execution time is estimated as it is explained in [7] and [11].

The data from the PMU are taken periodically, and at the beginning of each period, they are used to estimate the application's workload. The estimation is done by applying the linear regression to the collected data. By being able to estimate application's workload, it is feasible to control application's execution time by changing the CPU clock frequency. Applying lower frequencies and, therefore, lower voltages, the power consumption can be reduced. The needed frequency and the voltage are calculated at the beginning of each time period. In [7] and [11] power savings up to 80% are achieved. However, those solutions do not have in mind the influence of the finite number of CPU frequencies. If the calculated frequency is 135MHz, for example, the applied frequency will be 100MHz, because it is the closest one (the CPU has a finite number of clock frequencies, as it is shown in Table 1). Therefore, the active application will run slower than it is supposed, and this error is not taken into account in the next frequency calculation, i.e. the algorithm works in open loop, Figure 6.

In this paper a similar algorithm is proposed, but using the applied frequency as a feedback (Figure 7) to compensate the finite number of CPU frequencies.

In the proposed algorithm if the calculated frequency for two time intervals is, for example, 175 MHz. First is applied the frequency of 200 MHz, because it is the closest one, and then the one of 100 MHz. The algorithm would do it in the manner that the application lasts as if it was running 175 MHz all the time.

Due to the increased execution time it is necessary to define the application's time performance loss as follows, [7]:

$$PF \cdot T_{f_{MAX}^{CPU}} = T_{f^{CPU}} - T_{f_{MAX}^{CPU}} \quad (1)$$

where $T_{f_{MAX}^{CPU}}$ stands for the execution time at the CPU's maximum frequency, and $T_{f^{CPU}}$ is the execution time at the CPU frequency of f^{CPU}. Thus, PF shows how much the execution time of the tested application is longer than the time when the application is executed with maximal speed. For example, in the case of PF=0.2, execution time of the application with the frequency of CPU's clock of f^{CPU} is 20% longer than the time in the case when the maximal frequency is applied.

In order to calculate the frequency it is necessary to have a valid system model. Using the definitions from Table 2 and relationship between $f_{CPU,}$ f_{EXT} and f_{INT} and it can be shown that the CPU frequency is given by:

$$f_{CPU} = \frac{f_{MAX}(1+\beta)}{\mathrm{PF_{DEMANDED}} + 1 + \beta(4\alpha + 2\alpha')(\frac{PF_{DEMANDED}}{M'\alpha + N'\alpha'} + \frac{1}{M\alpha + N\alpha'})} \quad (2)$$

Detailed explanation of the system model can be found in [12].

As it was afore mentioned, the algorithm is trying to foresee the future workload by analyzing the data from the past, therefore this type of algorithms is usually referred to as "history" based. The error in the estimation of the application's performance produced by this method depends on the dynamics of the tested application, and this approach cannot be used for real time systems, since the algorithm does not have any information about the time constraints of running application.

The algorithm proposed in this work, compared with the algorithms in [7] and [11] , is aware of system limitations, i.e. finite number of possible clock frequencies and the control of execution time is performed in closed loop, while in [7] and [11] the algorithm works in open loop. Additionally, comparing the proposed model with the system model in [7], the proposed model clearly includes influence of the external and internal buses on application's execution time.

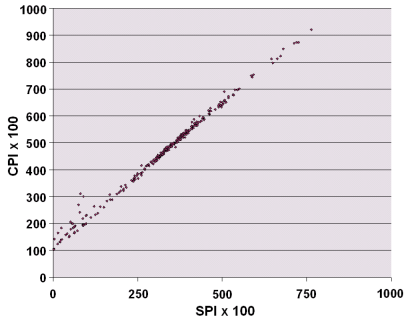

Figure. 5 Linear dependency between CPI and SPI for gzip application.

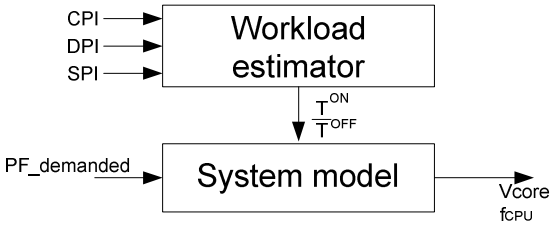

Figure 6. Block diagram of algorithm's control in open loop.

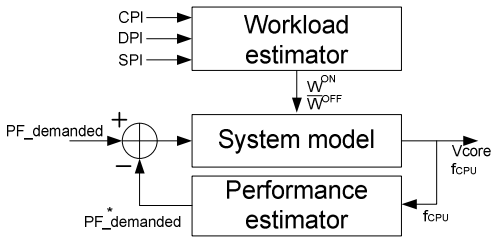

Figure 7. Block diagram of algorithm's control with feedback.

TABLE 2
DEFINITIONS OF THE VARIABLES USED IN THE SYSTEM MODEL

Variable	Definition
f_{EXT}	the frequency of the external bus
f_{INT}	the frequency of the internal bus
t_{ON}	W_{ON}/f_{CPU}, the time needed to execute on-chip workload
t_{OFF}	W_{OFF}/f_{BUS} the time needed to execute off-chip accesses
T	$t_{ON}+ t_{OFF}$, the task's execution time
f_{BUS}	$f_{INT}/ \alpha + f_{EXT}/ (1-\alpha)$, $\alpha \in (0,1)$
α'	1-α
B	t_{OFF}/t_{ON}
M	f_{CPU} / f_{EXT} in the next time interval
N	f_{CPU} / f_{INT} in the next time interval
M'	f_{CPU} / f_{EXT} in the previous time interval
N'	f_{CPU} / f_{INT} in the previous time interval

The good behavior of the algorithm depends strongly on the good estimation of the workload and the good system model. If the estimation is done poorly, applied frequency will not provide the demanded execution time.

IV. Experimental Results

The set of applications we used to test our DVS system consists of: gzip (compression of files), bfish (file encoding) and cjpeg (compression of photos).

The first experiments were done in order to see if the frequency feedback was necessary. Figure 8 presents (measured) time performance loss vs. demanded when the frequency feedback is not applied. Figure 9 shows the same dependency for the same application, but this time when the frequency feedback is present. As it can be seen for higher values of time performance loss the system with feedback works better. For instance, when the demanded PF is 50%, the difference between the demanded time performance loss and the actual one is about 10% in the system without feedback. The reason for the difference between the demanded PF and actual PF lies in the accumulated error between applied clock frequency and the calculated one. In some cases the average value of this error is close to zero, e.g. when the demanded PF is 20% in Figure 8, but in other it can accumulate and produce huge errors, as we have seen. Therefore, it is necessary to provide a feedback to the DVS algorithm.

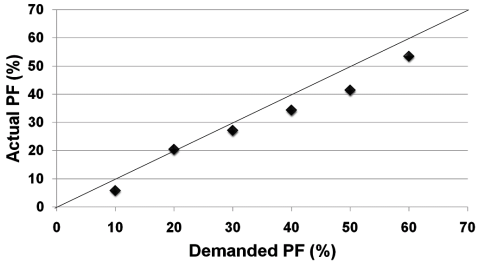

Figure 8. Actual PF Vs. Demanded PF without the frequency feedback.

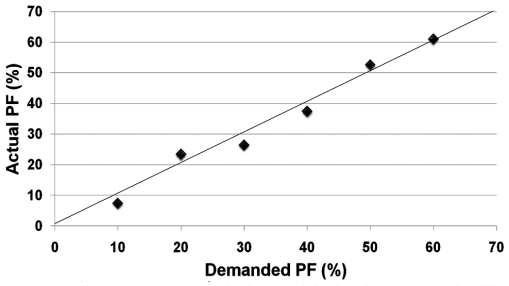

Figure 9. Actual PF Vs. Demanded PF with the frequency feedback.

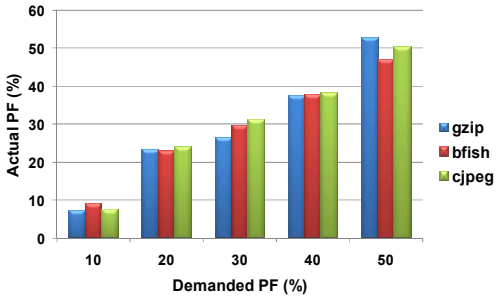

Figure 10. Actual (Measured) Vs. Demanded PF.

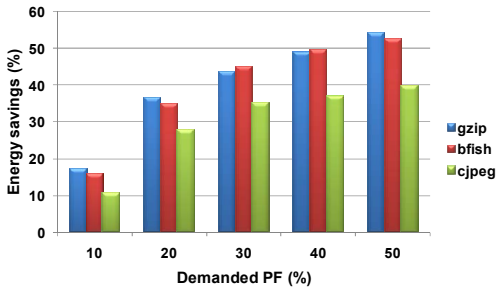

Figure 11. Achieved energy savings Vs. Demanded PF.

Figure 10 shows the actual performance loss of the system for the tested applications in the case when the feedback is applied. It can be seen that is very close to the values that are demanded, all values are in ±5% of $PF_{DEMANDED}$.

Once, when the control of time performance has been assured it is important to see what the energy savings are.

Figure 11 presents the energy saving and it can be noticed that they are asymptotically drawing near certain value. The reason for this is that the energy savings cannot be higher than in the case of minimal CPU frequency. By increasing the $PF_{DEMANDED}$, the average CPU frequency is getting closer to 100 MHz (the minimal frequency), so that energy savings are drawing near the maximum savings, which are limited by the hardware.

V. Algorithm's Imapct on the System Performance

The next step was to show the influence of the implemented algorithm on the system performance. As it was aforementioned, the implemented DVS algorithm acts as a periodic system task. We have conducted a series of tests to find the optimal period for the algorithm regarding the best time execution control. The best results are obtained for the periods around 20ms. For shorter periods the algorithm does not act correctly. The reason for this lies in the nature of the algorithm itself. If the small period is

applied it can be seen that CPI and DPI have high dynamics, and this high dynamics leads to great oscillations in the estimation of the workload, because the algorithm uses several last measurements to estimate it by regression. On the other hand, for longer periods this high dynamics is "filtered", the measured CPI is closer to its current averaged value, therefore, the applied regression will provide better estimation of the current workload.

Measuring the extra execution time and extra energy needed by the implemented algorithm and frequency/voltage changes it was shown that both of them are lower than 2% of the time and energy when the tested application runs at the maximum speed, Figures 12 and 13. The measurements were conducted when the period of the algorithm was 20ms. The time and the energy spent during frequency/voltage transitions are negligible in this case.

VI. SLEW RATE EFFECT

Once we obtained the results about the influence of the implemented algorithm on system performance, we tried to improve them by increasing the slew rate of the power supply. The slew rate of the power supply can be changed inside the operating system by changing the voltage step we have mentioned earlier. On our surprise, we could change only the rising slew rate. When the core's voltage tends to be lower both switches of the synchronous buck converter, which is used to supply the core, are open and the core is supplied by the output capacitor (220uF) until the moment when the output voltage falls to the desired voltage. This can be clearly seen in Figure 14. The green trace represents the current of supply's inductor, and it can be seen that at the beginning of the voltage/frequency transition current drops, because of the frequency change. Then, the PLL needs to stabilize, and when the frequency is locked, the voltage transition begins. However, during the voltage transition there is not any current in the inductor, hence, it can be concluded that the processor is supplied from the output capacitor. The converter starts to regulate again in the moment when the supply's output reaches the target voltage.

Although the slew rate was not as we expected, by analyzing the waveforms and using some control signals that we implemented inside the operating system, we could estimate with high accuracy the additional execution time and energy that we would have in the case of the demanded slew rate. The results in the case of *bfish* are summarized in Figures 15 and 16.

The results show that although higher slew rate leads to less additional energy and time there is not significant improvement of system behaviour for higher slew rates.

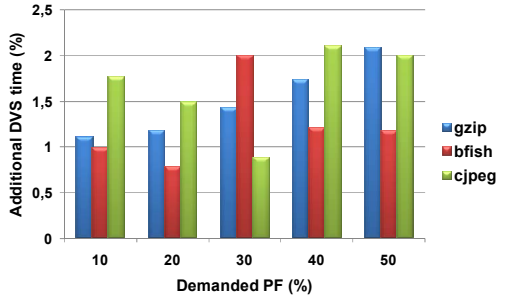

Figure 12. Additional execution time due to DVS transitions.

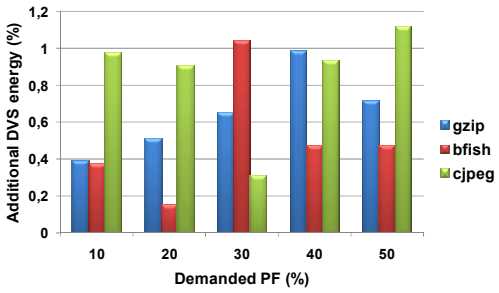

Figure 13. Additional energy due to DVS transitions.

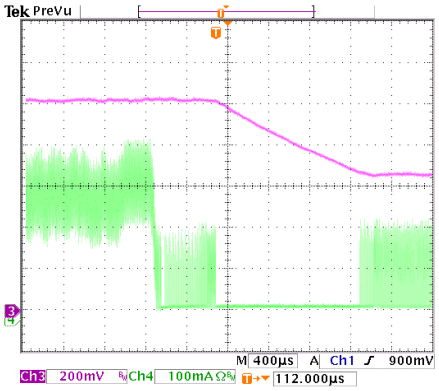

Figure 14. Core voltage transition (pink trace) and output inductor current (green trace).

The main reason for this is relatively small number of voltage/frequency changes (no more than 100 for one application) and the fact that each transition is composed of the time needed for the voltage change and the time spent by PLL to lock to the new frequency. By measuring these times it is determined that, approximately, 80% of the time needed by one transitions is spent by PLL. Due to strong influence of PLL the slew rate of the power supply does not have great influence on the execution time of the algorithm, so that we did not want to change the power supply with the faster one, because the system would be more complex and there would not be much benefit.

VII. PROBLEMS TO ESTIMATE PROCESSOR'S WORKLOAD

In [12] was explained that the main drawback of this algorithm is that it is necessary to adjust the value of the constant α from the equation (2). This constant represents the ratio between the time needed by internal bus data transfer, and the time needed by external bus data transfer. Unfortunately, the value of this constant differs depending on the type of application. Even more, for some applications we could not adjust the value of the constant, e.g. when we tried to compress a black and white photograph into jpg format, or to compress a pdf file using gzip.

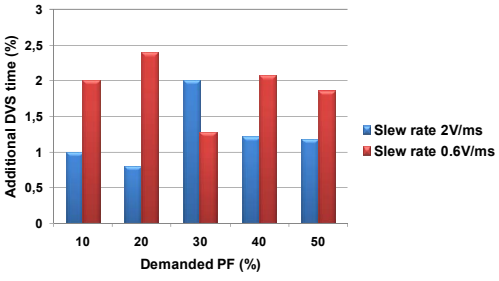

Figure 15. Additional execution time due to DVS transitions for different slew rates of power supply.

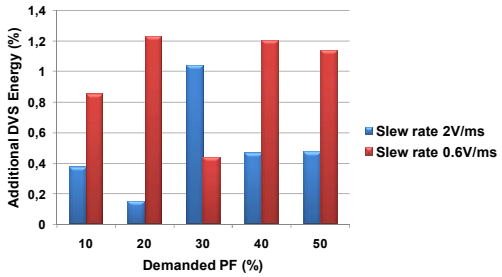

Figure 16. Additional energy due to DVS transitions for different slew rates of power supply.

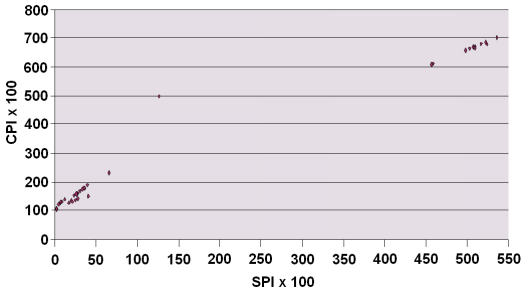

Figure 17. CPI Vs. SPI when the algorithm does not work correctly

The main cause of this problem can be in the statistics of the tested application. If we compare the statistics for gzip application in the case when we want to compress a txt file and when we want to compress a pdf file, we can see a significant difference between these two processes. In Figure 17 CPI vs. SPI diagram for gzip compressing a pdf file is presented. The lineal dependency between CPI and SPI is not obvious, and the measured points are separated. After a series of tests we came with assumption that the algorithm did not model the time needed to read a source file, and later to write the results to an output file, but it is yet to be proved.

VIII. CONCLUSIONS

In this paper an algorithm for DVS system on microprocessor platform is presented. The goal of the implemented algorithm is to improve the energy efficiency by demanding some acceptable performance loss. According to the experimental results, it is possible to save up to 50% of the CPU energy with 50% performance loss. The power savings vary with the processors load, i.e. the active application. The algorithm is strongly dependent on good estimation of the core's workload, and it could be a problem to estimate it in some cases. In order to compensate the finite number of possible frequencies for the system clock the algorithm should work with the frequency feedback. The feedback should compensate the error that could be accumulated due to the difference between demanded and selected frequency. The additional time and energy consumed by the implemented algorithm does not need more than 2% of application's time and energy. The analysis of the extra time needed for DVS transitions has been performed, showing that it does not need a voltage supply with fast transitions, due to long duration of frequency changes.

REFERENCES

[1] Transmeta's Design guides and Datasheets
[2] http://support.intel.com/support/processors/mobile/pentiumiii/sb/CS-007509.htm
[3] Mobile AMD-K6 Processor Power Supply Design, Application Note
[4] Kawaguchi, H.;Shin Y.;Sakurai T. "uITRON-LP: Power-Conscious Real-Time OS Based on Cooperative Voltage Scaling for Multimedia Applications" *IEEE transactions on multimedia*, Vol.7,No.1,Feb 2005
[5] Elgharbawy W.M.; Bayoumi M.A. "Leakage sources and possible solutions in nanometer CMOS technology" Circuits and Systems Magazine, 2005
[6] Soto A.;Alou P.;Cobos J.A.;Uceda J. "The future DC-DC converter as an enabler of low energy consumption systems with dynamic voltage scaling" IECON 02, vol. 4,
[7] Choi, K.; Soma, R.; Pedram, M "Dynamic Voltage and Frequency Scaling based on workload Decomposition" *Proceedings of the 2004 International Symposium on Low Power Electronics and Design, ISPLED, 2004*
[8] Intel PXA255 Processor, Developer's manual, Jan. 2004.
[9] Viper Intel PXA255 XScale RISC based PC/104 single board computer, technical manual
[10] Soto A.;Castro A.;Alou P.;Cobos J.A.;Uceda J.;Lotfi A "Analysis of the Buck Converter for Scaling the Supply Voltage of Digital Circuits" *APEC '03, Eighteen Annual IEEE, vol. 2, February 2003*.
[11] Choi, K.; Soma, R.; Pedram, M. "Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Tradeoff Based on the Ratio of Off-Chip Access to On-Chip Computation Times" *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, No. 1, Jan. 2005
[12] Vasic M.;Garcia O.;Alou P;Oliver J.A.; Cobos J.A. "Trade-off between energy savings and execution time applying DVS to a microprocessor", *5th international Conference on Integrated Power Electronics System, CIPS 2008*