Performance and Energy Bounds for Multimedia Applications on Dual-processor Power-aware SoC Platforms

ABSTRACT

The energy consumption of SoC architectures for multimedia processing is now as important as their performance because of the plethora of battery-operated devices running multimedia applications. In this paper, we present an analytical framework to evaluate the performance and the power of a dual-processor SoC architecture that supports dynamic frequency scaling in an integrated manner. As a result, we identify performance and energy bounds associated with platform configuration tradeoffs that includes operating frequencies of the processors, buffer sizes, buffer types and scheduling schemes. It is difficult and costly for simulation-based approaches to evaluate such tradeoffs because of the bursty nature of multimedia traffic and the high variability in the multimedia processing requirements. Furthermore, our model accounts for the impact of leakage power of platforms built with nanometer range process technologies.

1. INTRODUCTION

The convergence of hand-held devices has seen the emergence of multimedia applications to be an important class of applications running on such devices. The architecture of multimedia processing devices is typically that of a system-on-chip (SoC) platform. SoC platform designs often involve tradeoffs among conflicting factors such as performance, power consumption and cost. The nature of multimedia applications, in particular, the processing of variable bit rate streams, brings a new dimension to these tradeoffs.

Many new methodologies, languages and tools have been developed to improve the productivity of designers. Most of these [12, 13, 15] focus on the enhancement of the performance and the quality of the applications due to the complicated nature of the multimedia streams.

In the context of multimedia processing on hand-held devices, the issue of power consumption is as important as performance. Considerable amount of efforts have been expended to address the issue of power-awareness. Proposals include dynamic voltage and frequency scaling (DVFS) [4, 3, 10, 17], dynamic power management (DPM) [11] and energy consumption management [2]. Most of these techniques partition multimedia workloads in different ways in order to exploit the advantages of different scheduling methods to save energy (thereby extending battery life) with guaranteed quality of service for multimedia applications.

The issues of power and performance are tightly coupled. In this paper, we propose a methodology for considering these two issues as an integrated problem. We start with the following problem statement.

Problem Statement: Given a SoC platform containing multiple processors and buffers and an multimedia application running on the platform, we are interested in reducing the overall energy consumption without undermining quality of service guarantees. A guaranteed playback rate would be an example of the latter.

Many issues pertaining to this problem often pose serious challenges to platform designers. The following are some of the factors to be considered: multiple frequency settings of processors on the SoC platform, dynamic frequency and voltage scaling policies, processor customization to cater for applications, and the parameter extraction to characterize the burstiness of multimedia applications.

Some of the above issues have become industrial interests since the availability of the Intel XScale processor [16] and the Transmeta Crusoe processor [14]. These processors allow voltage and frequency values to be adjusted dynamically. Platform designers are interested in applying various DFS schemes to tradeoff performance and energy consumption.

Our contribution and relation to previous work: In this paper, we present an analytical framework which provides platform designers the capability to analyze both performance and power. Apart from a generalization of previous works [7, 8], this framework addresses the energy consumption issue in an integrated manner. Another important fact is that the leakage power starts to dominate power consumption as process technology progressed to below 90nm. Therefore, leakage power has to be considered in current real-time systems [5]. Our model goes beyond previous work [6] on modelling SoC platforms for multimedia applications by taking (i) the leakage power into consideration, (ii) frequency tuning of multiple (two) processors instead of a single processor, and (iii) a scheduling policy consisting of a choice of the duty cycle and scheduling period.

Our framework is built on top of a concept of *variability characterization curves* (VCCs) [7, 8]. VCCs focus on descriptions of the variances in the metrics characterizing multimedia applications.

To obtain the energy metrics, our VCC framework first applies performance analysis to the incoming and outgoing streams of the different PEs and buffers. With results from the performance analysis, the experimental results of [11] and the specifications [16, 14] are used to calculate the lower and upper bounds of energy dissipation of the PEs and the buffers. Consequently, the bounds of total energy costs of the whole SoC platform are obtained by summing up the energy consumption of all the PEs and buffers.

The effectiveness and usability of our framework will be illustrated with a case study. A simple dynamic frequency scaling (DFS) scheme will be tested in our experiments with multiple operating frequencies. The difference in the impacts of the scheme on both the total energy dissipation and the performance of the SoC platform with configuration tradeoffs will be clearly observed in the experimental results. We shall also illustrate the impact of the leakage power on the tradeoffs between adjustments for better performance and designs for energy consumption reductions.

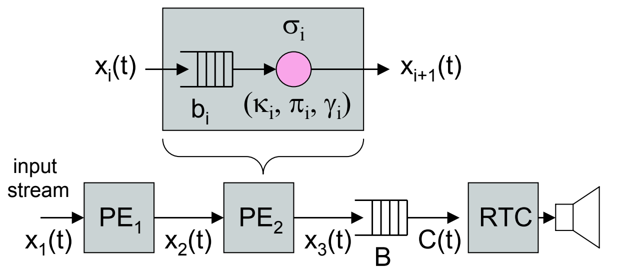

Figure 1: Stream processing on a dual-processor SoC platform

The remainder of the paper will begin with Section 2 to formulate the problems. Section 3 will detail descriptions of variability characterization curves. Section 4 will present our power model integrated with the performance analysis. In Section 5 is a case study to show the effectiveness of our integrated framework. This will be followed by a few concluding remarks.

2. PROBLEM FORMULATION

In this paper, we shall consider a dual processor SoC platform as a system-level model for multimedia stream processing. The SoC architecture modelled shown in Figure 1 has two processing elements (PEs) where different phases of a stream application are carried out. Such phases are actually pipelined tasks implemented on the two PEs. Each PE is assigned with one or more tasks. The first PE takes in the stream input to the platform, finishes the task(s) assigned to the PE on the input, then produces the outgoing stream to the second PE. Just like the first PE, the second PE processes the incoming stream by carrying out its task(s), then generates the output stream. Accepting the output of the second PE is a real-time client (RTC). Each PE has a FIFO buffer to store the incoming stream. The buffer in front of the RTC is called the *playout buffer*. We believe that our analysis of this dual processor architecture can be easily extended to a multi-processor pipeline.

We use $x_1(t)$ to denote the input stream of this application. This is the number of stream objects that reach PE_1 during the time interval $[0, t]$. Similarly, we use $x_2(t)$ to denote the outgoing stream of PE_1, which is the input to PE_2. PE_2 performs its task(s) on $x_2(t)$ and produces a third outgoing stream denoted by $x_3(t)$. $x_3(t)$ is the final processed stream that is stored in a playout buffer of size B. The playout buffer is consumed at a rate specified by $C(t)$, which represents the number of stream objects that are read from the playout buffer during the time interval $[0, t]$. The tasks on each PE will be specified by variability characterization curves (VCCs) in next section.

3. VARIABILITY CHARACTERIZATION CURVES

As described earlier, the performance issue is to guarantee the playback rate. Since the RTC plays back the video from the buffer in front of it, the playback rate is guaranteed if the playout buffer always contains enough data, i.e. it never underflows. Let B_0 denote the buffer fill level of the playout buffer and C denote the consumption rate of RTC assuming the initial buffer fill level is 0. Then the constraint on the playout buffer underflowing can be stated as:

$$x_3(t) \geq C(t) - B_0, \quad \forall t \geq 0 \tag{1}$$

Similarly, the constraint on the playout buffer overflow can be stated:

$$x_3(t) \leq C(t) + B_0, \quad \forall t \geq 0 \tag{2}$$

In general, the constraint that the internal buffer in the the ith PE should not overflow, is:

$$x_{i+1}(t) \geq x_i(t) - b_i, \quad \forall t \geq 0 \tag{3}$$

where b_i is the buffer fill level of the buffer of the ith PE.

All of the inequalities (1 2 3) have to be satisfied as the performance constraints in later analysis.

On each of PE_1 and PE_2, the stream s is scheduled by a scheduler, which apart from s also schedules other streams and real-time tasks possibly implemented on these processors. The *service* received by s at the two processors can be specified by service curves σ_1 and σ_2, where these functions depend on the scheduling policies used and their associated parameters. The variability associated with processing s on the two processors is specified by the VCCs $(\kappa_1, \pi_1, \gamma_1)$ and $(\kappa_2, \pi_2, \gamma_2)$.

More detailed definitions of workload curve $\gamma = (\gamma^l, \gamma^u)$, consumption and production curves $\kappa = (\kappa^l, \kappa^u)$ and service curve $\sigma = (\sigma^l, \sigma^u)$ can be referred in [9].

Number of activations $a(t)$: During the time interval $[0, t]$, the minimum and maximum number of activations of T that can be requested by the stream are equal to $\kappa_i^l(x_i^{\min}(t))$ and $\kappa_i^u(x_i^{\max}(t))$ respectively. Since the service guaranteed to the stream on the ith PE is σ_i, the minimum and maximum number of activations of T that are possible during $[0, t]$ are $\gamma_i^l(\sigma_i^l(t))$ and $\gamma_i^u(\sigma_i^u(t))$ respectively. Therefore, the minimum and the maximum number of activations of T that occur in $[0, t]$, which we denote using $a_i^{min}(t)$ and $a_i^{max}(t)$ respectively, can be computed as follows (see [1] for the mathematical background):

$$\begin{aligned} a_i^{min}(t) &= \kappa_i^l(x_i^{\min}) \otimes \gamma_i^l(\sigma_i^l)(t) \\ a_i^{max}(t) &= \kappa_i^u(x_i^{\max}) \otimes \gamma_i^u(\sigma_i^u)(t) \end{aligned} \tag{4}$$

Then clearly, $x_{i+1}^{\min} = \pi_i^l(a_i^{min}(t))$ and $x_{i+1}^{\max} = \pi_i^u(a_i^{max}(t))$. x_{i+1} serves as the input to PE_{i+1} and is possibly composed of stream objects of a different type, compared to the stream objects at the input of the ith PE.

The above bounds can be used to compute the maximum backlog of stream objects at the input of the ith PE. This backlog denotes the minimum amount of buffer that must be provided in the ith PE to guarantee that an overflow does not occur. Hence, the buffer size b_i is derived from [1] as follows:

$$b_i = \sup_{t \geq 0}\{x_i^{min}(t) - \kappa_i^{l^{-1}}(a_i^{min}(t)), x_i^{max}(t) - \kappa_i^{u^{-1}}(a_i^{max}(t))\} \tag{5}$$

4. POWER MODEL

We consider the energy of PEs and buffers in two different ways separately. The PE's energy is modelled as the accumulation of the power consumption over the time. The buffer's energy is modelled as the effect of the amount of stored data [11]. We connect both the methods to our performance analysis in Section 3.

In our power model, we take a simple DFS scheme into our consideration. The simplest DFS scheduling policy is to switch the ith PE into the idle mode in one portion of every scheduling period (ρ_i) of the ith PE and switch the ith PE back to the active mode in the other portion. The scheduling period, measured in numbers of CPU cycles, is the unit in which the DFS scheme is applied. For example, if the scheduling period is 1,000 cycles, then under this simple DFS scheme with a *duty cycle* of 0.5, the processor would be active for 500 cycles, then idle for 500 cycles, then again active for 500 cycles and idle for 500 cycles and so on. The scheme does not affect the buffers since the buffers must be active constantly to keep the data stored and accessible.

In summary, a DFS policy in this paper consists of a pair of frequency choices for the two PEs, the duty cycle, and the scheduling period. The latter two are the same for both processors. We believe our model can be extended to allow for these last two parameters to be PE-specific too but that would complicate the discussion.

PE's energy $E_{p,i}$: The energy of the ith PE in the period $[0, t]$ comprises of two parts due to the scheduler's actions. The first part is the energy consumed in the active mode while the second is the energy consumed in the idle mode. We use $p_{p,i}^{active}$ to denote the power of the ith PE during an activation of T that occurs in $[0, t]$. $p_{p,i}^{active}$ is a constant when the ith PE is activated by T. $p_{p,i}^{idle}$ denotes the power of the ith PE in the idle mode.

In period $[0, t]$, the maximum and minimum lengths of activation durations in clock cycles on the ith PE are denoted with $L_{p,i}^{\max}$ and $L_{p,i}^{\max}$ respectively. The clock frequency of the ith PE is represented by Ω_i.

The maximum amount of active time in seconds is denoted by t_{active}^{max}. Similarly, t_{active}^{min} represents the minimum amount of active time in seconds. t_{active}^{max} and t_{active}^{min} are derived as follows:

$$\begin{aligned} t_{active}^{max} &= a_i^{max}(t)\ L_{p,i}^{\max}/\Omega_i \\ t_{active}^{min} &= a_i^{min}(t)\ L_{p,i}^{\min}/\Omega_i\ , \end{aligned} \quad (6)$$

where a_i^{max} and a_i^{min} are derived in Eq. 4. The impacts of the DFS scheme are directly on the service curves (σ_1 and σ_2), then reflected on a_i^{max} and a_i^{min}.

Then the amount of idle time in $[0, t]$ is $t - t_{active}^{max}$ or $t - t_{active}^{min}$.

Let $E_{p,i}^{\max}$ and $E_{p,i}^{\min}$ represent the maximum and minimum amounts of energy of the ith PE spent on activations of T in $[0, t]$. $E_{p,i}^{\max}$ and $E_{p,i}^{\min}$ are obtained as follows:

$$\begin{aligned} E_{p,i}^{\max} &= p_{p,i}^{active}\ t_{active}^{max} + p_{p,i}^{idle}(t - t_{active}^{max}) \\ E_{p,i}^{\min} &= p_{p,i}^{active}\ t_{active}^{min} + p_{p,i}^{idle}(t - t_{active}^{min})\ . \end{aligned} \quad (7)$$

If we consider the leakage power denoted by $p_{p,i}^{lk}$ and the energy overhead of mode switching denoted by $E_{p,i}^{sw}$, then Eq. 7 is modified as follows:

$$\begin{aligned} E_{p,i}^{\max} &= p_{p,i}^{active}\ t_{active}^{max} + p_{p,i}^{idle}(t - t_{active}^{max}) + p_{p,i}^{lk}\ t + E_{p,i}^{sw} \\ E_{p,i}^{\min} &= p_{p,i}^{active}\ t_{active}^{min} + p_{p,i}^{idle}(t - t_{active}^{min}) + p_{p,i}^{lk}\ t + E_{p,i}^{sw}\ , \end{aligned} \quad (8)$$

where $E_{p,i}^{sw}$ depends on the scheduling policy, and $p_{p,i}^{lk}$ is given by [5]:

$$p_{p,i}^{lk} = V_{dd}\ I_{subn} + |V_{bs}|\ I_j\ . \quad (9)$$

In Eq. 9, the subthreshold current (I_{subn}), the body bias voltage (V_{bs}) and the reverse bias junction current (I_j) are process technology specific. For a fixed process technology, the variables which platform designers can manipulate parts are the number of devices in the circuit (L_g), and the supply voltage (V_{dd}).

We assume that the switch from the active mode to the idle mode can be done instantaneously. The delay incurred by switching from the idle mode to the active mode is called *wake-up delay* denoted by D_{wakeup}. The number of occurrences of such switches is given by t/ρ_i. Then $E_{p,i}^{sw}$ is just:

$$E_{p,i}^{sw} = p_{p,i}^{idle}\ t/\rho_i\ D_{wakeup}\ . \quad (10)$$

Energy consumption of the buffers $E_{b,i}$: We use $p_{b,i}$ to denote the power consumption of buffer in front of the ith PE in $[0, t]$. $p_{b,i}$ is a constant in $[0, t]$ since the buffer is always busy refreshing the stream objects regardless of the status changes of the ith PE. We use $E_{b,i}$ to represent the energy amount of buffer in front of processing element the ith PE in $[0, t]$. The upper bound $E_{b,i}^{max}$ is calculated easily by:

$$E_{b,i}^{max} = p_{b,i}\ Q_i^{max}, \quad (11)$$

where Q_i^{max} is the maximum amount of stored data in bits in the buffer in front of the ith PE. This is simply the product of the size of the objects and the maximum object queue length, i.e. the minimum buffer size obtained from Eq. 5. There is no general form of the lower bound $E_{b,i}^{min}$ because of the possibility that the minimum amount of stored data drops to zero when an underflow case occurs. It is safer to assume the lower bound of Q_i^{min} to be zero.

Total energy consumption E: We sum up the amounts of energy of all the buffers and PEs in $[0, t]$ to obtain the maximum and minimum amounts of total energy $E^{\max}$ and $E^{\min}$:

$$\begin{aligned} E^{\max} &= \Sigma_i(E_{b,i} + E_{p,i}^{\max}) \\ E^{\min} &= \Sigma_i(E_{b,i} + E_{p,i}^{\min})\ . \end{aligned} \quad (12)$$

The impact of the DFS scheme of the PEs is on the service curves (σ_1 and σ_2), number of activations (a_i^{min} and a_i^{max}), energy amounts of PEs and the total energy.

5. CASE STUDY: MPEG-2 DECODER

We present our experiments on an MPEG-2 decoder as our case study in this section. One motivation of the experiments is to illustrate how to apply the analytical framework described previously. Another more important objective is to derive the trade-off curves so as to reveal how the energy consumption is associated with PE's operating frequencies and the DFS scheme.

Experiment Setup: For our experiments, we map an MPEG-2 decoder onto a high level SoC platform template shown in Fig. 1. The decoder is partitioned into two coarse grain partitions implemented on two processing elements, i.e. PE_1 and PE_2. The first partition includes the tasks of *variable length decoding* (VLD) and *inverse quantization* (IQ). The second partition performs the *inverse discrete cosine transform* (IDCT) and *motion compensation* (MC).

PE_1 and PE_2 work in a pipelined manner where synchronization is achieved via FIFO buffers. The input stream enters the FIFO buffer in front of PE_1. The original input stream composes of compressed bits. PE_1 reads the input, produces partially decoded stream data and writes the decoded data out to the FIFO buffers in front of PE_2. The input to PE_2 is a sequence of partially decoded *macroblocks*. PE_2 completes the decoding algorithm and saves completely decoded macroblocks in the playout buffer in front of RTC. In our experimental settings, we use a customized version of the SimpleScalar instruction set simulator to obtain the traces of cycle requirements of the MPEG-2 decoder tasks.

We use a simple DFS schemes which keep PEs active and idle at certain duty cycles. The DFS scheme can be implemented by exploiting the *active* and *idle* modes of the Intel 80200 processor [16] which is based on Intel XScale micro-architecture. The Intel 80200 processor provides a simple instruction to enter the idle mode. To wake up from the idle mode, a wake-up event in the form of either the assertion of a FIQ (fast interruption) or the IRQ (normal interruption) pins. This DFS scheme can also be implemented on the Transmeta Crusoe processor [14]. On the Crusoe processor, the active mode is called the *working* mode, while the idle mode is called the *auto halt* mode.

We summarize the Intel 80200 processor's power parameters in Table 1. The wake-up delay of the processor is of the same value as the interrupt latency since the wake-up is made via an interrupt. The minimum interrupt latency of Intel 80200 processor is 3 clock cycles. The power characteristics of Transmeta Crusoe processor

Symbol	Parameter	Max.	Units
I_{CC} Active Mode	Core Current 733Mhz at 1.5v I_{CC}	720	mA
	400Mhz at 1.3v I_{CC}	410	mA
I_{CC} Idle Mode	at 1.5v I_{CC}	190	mA
	at 1.3v I_{CC}	135	mA

Table 1: Intel 80200 XScale processor power characteristics.

V_{dd}	Ω_i(Mhz)	$p_{p,i}^{active}$(**W**)	$p_{p,i}^{idle}$(**W**)	$p_{p,i}^{lk}$(W)
0.55	100	0.08	0.05	0.18
0.60	167	0.10	0.11	0.20
0.65	233	0.18	0.15	0.25
0.70	300	0.28	0.24	0.29
0.75	367	0.37	0.34	0.34
0.80	433	0.52	0.45	0.40
0.85	500	0.68	0.59	0.46
0.90	567	0.85	0.75	0.53
0.95	633	1.08	0.93	0.62

Table 2: Transmeta Crusoe processor power characteristics.

are shown in Table 2. The wake-up latency of Transmeta Crusoe processor is 260ns.

For the buffer's power specifications, we used the experimental values of $p_{b,i}$ previously reported in the literature [11]. A typical value for 64 MB SDRAM buffer is 0.012W/MB. The bounds of energy consumption of buffers are calculated by Eq. 11 with the values of $p_{b,i}$ and results of Eq. 5.

The input video clip for the experiments with Intel XScale processor parameters is the "susi" MPEG-2 video clip which has a constant bit rate is 8 Mbps. The input video clip for the experiments with Transmeta Crusoe processor parameters is the "automotive" MPEG-2 video clip that has a constant bit rate of 4 Mbps. To obtain the cycle requirements of the different tasks, the MPEG-2 decoder process is profiled with the models of PEs based on a customized version of the SimpleScalar instruction set simulator. The VCC curves $(\kappa_1, \pi_1, \gamma_1)$, $(\kappa_2, \pi_2, \gamma_2)$ are calculated off-line with the cycle requirements.

The output of the VLD+IQ task on PE_1 is characterized by $\gamma_i = (\gamma_i^l, \gamma_i^u)$ curves, i.e. the lower and upper bounds of the production rate. The VCC π_i curves are straight lines whose slopes are determined by the number of stream objects consumed or produced by each task activation.

The client of the play-out buffer is a video playback device which has a frame rate of 25 frames per second and a resolution of 704 by 576 pixels.

Underflow possibilities associated with PE's scheduling periods and DFS duty cycle: In this section, we start with the experimental results on the associations between underflow possibilities and PE's scheduling periods based on the parameters of the Intel 80200 XScale processor. The processor is manufactured using Intel's 0.18-micron processor technology with a minimum supply voltage of 1.1v. The sum of input and output leakage current is 440μA [16]. Consequently, the leakage power is less than 0.001 mW. At this level, the leakage power is negligible, especially compared with processors built using more recent technologies such as 70nm and 90nm. For example, if projected to a 70nm process technology, the Transmeta Crusoe processor will have a leakage power over 100mW [5]. These scaled parameters will be the inputs to our experiments on associations between underflow possibilities and duty cycle of DFS schemes.

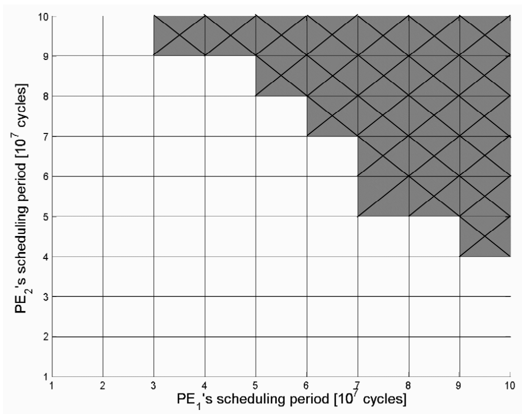

Figure 2: Underflow possibilities associated with scheduling periods (ρ) of PE_1 & PE_2 with a duty cycle of 0.5 at 733Mhz

According to Eq. 7, the bounds on the energy consumption of the PEs can be obtained given the VCC curves, $p_{p,i}^{active}$ and $p_{p,i}^{idle}$. We derive the experimental values of $p_{p,i}^{active}$ and $p_{p,i}^{idle}$ by multiplying I_{CC} (processor core current) and voltage values listed in Table 1.

For the client of the real-time system, the quality of service is ensured if there is no underflow at the playout buffer in the whole decoding process. In our model framework, the output of PE_2 is accumulated at the playout buffer. Hence, underflow can be observed by monitoring the fill level of the playout buffer.

When both PEs are running at 733Mhz with a DFS duty cycle of 0.5, underflow occurs for larger PE scheduling periods. Intuitively, since the PEs are switched to the idle mode for a long period of time, the chance of an underflow increases.

Fig. 2 details the relationship between underflow and the PE's scheduling periods. The darker crossed blocks in the figure represent the combinations of PE scheduling periods which result in underflow. Such results allow platform designers to make tradeoffs between the energy consumption and the possibilities of underflow by carefully selecting the combinations of scheduling periods.

We shall now turn our attention to the impact of different duty cycles using the parameters scaled up to the 70nm technology from specifications of the Transmeta Crusoe processor. A slower bit rate input stream 4 Mbps "automotive" is chosen to study the design tradeoffs for platforms running at low frequencies and supply voltages.

Fig. 3 shows that combinations of DFS policies with lower duty cycles produce underflow cases in the playout buffer. To prevent underflows, higher duty cycles should be chosen.

Minimizing energy in design space consisting of frequency combinations and DFS duty cycles: Platform designers tend to assume that minimizing frequency is the best means to reduce the energy consumption. However, this practice might be wrong because we also need to consider leakage power and the energy consumption of the buffers. This is especially true for processors built with technology below 90nm and operating in the lower operating frequencies where dynamic power is less dominating. In the experiments for Fig. 4 and Fig. 5, we applied the same frequency steps to both PEs. For each step of operation frequency of PEs and DFS policy, we derive the maximum energy consumption of PEs by bringing parameters from Table 2 into Eq. 8. We then summed up the maximum energy consumption of PEs with the energy consumption of buffers in Eq. 11 to obtain the bounds of maximum

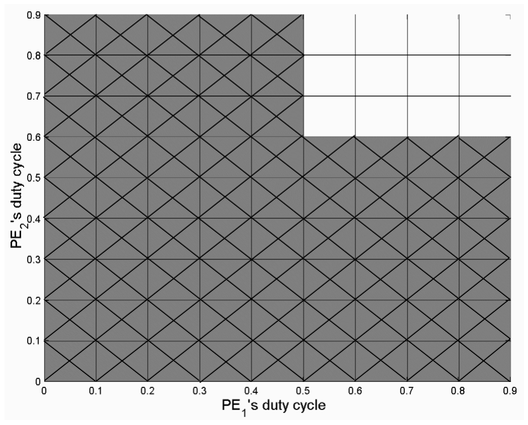

Figure 3: Underflow possibilities associated with varying duty cycles at 633Mhz

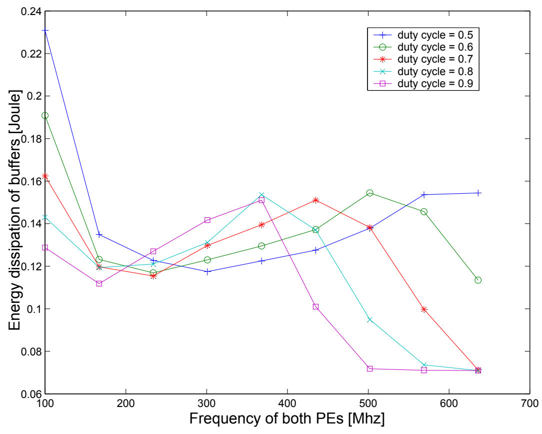

Figure 4: Bounds of buffer's maximum energy associated with the same frequencies of PEs with SDRAM buffers under varying duty cycle

total energy consumption without considering underflow.

It is interesting to observe that the curves in Fig. 4 and Fig. 5 have minimum points at the frequency of 167Mhz which is not the minimum frequency. Fig. 4 shows how DFS policies and frequencies affect the bounds of the maximum energy of buffers. For SDRAM only memory system, Fig. 5 shows how the bounds of maximum total energy consumption. The minimal result is actually the effect of the leakage power since the leakage power gradually dominates the dynamic power when the frequency and voltage decrease to that point. Furthermore, higher buffer fill levels in the buffers incur increases in the buffers' energy consumption.

Fig. 6 shows the association between the bound of maximum total energy and combinations of frequencies of PEs along with SDRAM buffers and a DFS scheme whose duty cycle is 0.9. The scheduling period is 50 million cycles. It is noteworthy that there is an area surrounded by points $(100, 100)$, $(367, 100)$, $(367, 500)$ and $(100, 500)$ on the surface where combinations of frequencies produce underflows in the playout buffer. Since the surface almost monotonously increases with the frequencies except for the starting point, designers should choose frequency combinations along the boundary of the area to minimize energy without violating the performance constraint. There are 9 points along the boundary of the area, among which $(367, 500)$ is associated with the minimum

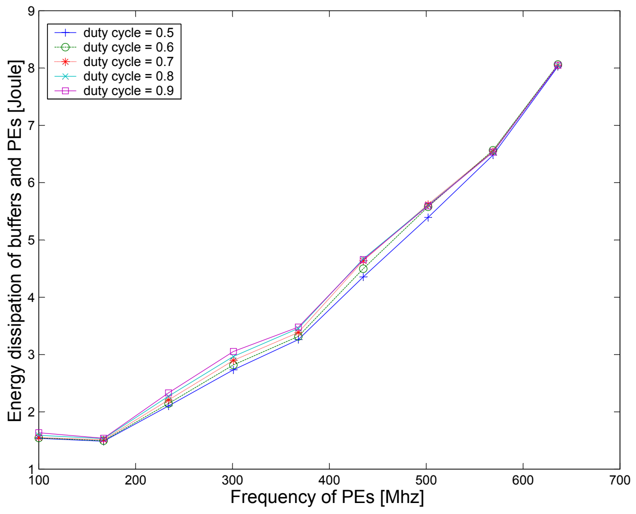

Figure 5: Bounds of maximum total energy consumption associated with the same frequencies of PEs with SDRAM buffers under varying duty cycle

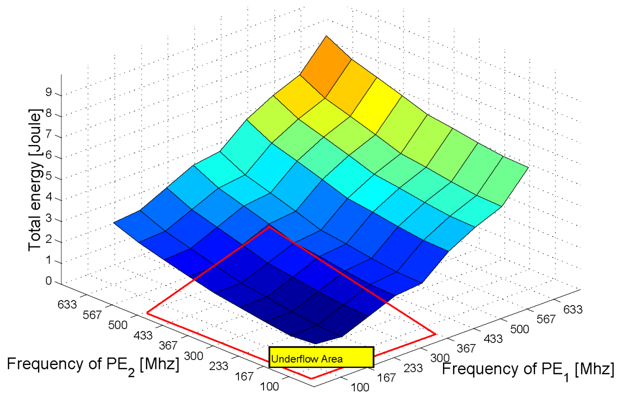

Figure 6: Bounds of maximum total energy consumption associated with combinations of frequencies of PEs with SDRAM buffers and a duty cycle of 0.9

energy among the 9 points. In other words, the frequency combination $(367, 500)$ Mhz gives the minimum of maximum total energy, 4.32 joules, without underflow.

For DFS policies whose duty cycle is less than 0.9, the underflow area grows. For example with a duty cycle of 0.8 (with the same scheduling period of 50 million cycles) we see a larger area surrounded by points $(100, 100)$, $(433, 100)$, $(433, 567)$ and $(100, 567)$. Among the points on the boundary, the (433,567) Mhz gives the minimum of the maximum total energy of 5.56 joules without underflows.

A higher duty cycle clearly lowers the speed requirement on the processors as more cycles are available to the application. However, in practice, designers may have to use the duty cycle of less than 1.0 to satisfy other design constraints such as allowing other tasks to run on the PEs. Our analytical model can be used in one of three practical situations:

- Assuming that the duty cycle is fixed because other tasks have to run on the PEs, then our model can help locate the frequency combination of the PEs that will meet the perfor-

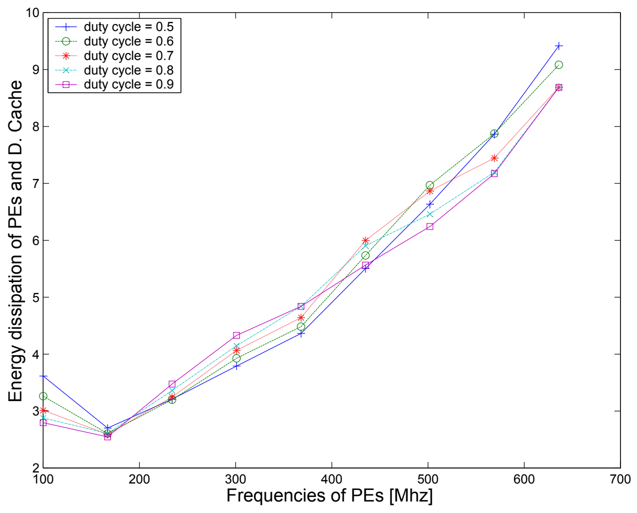

Figure 7: Bounds of maximum total energy consumption associated with the same frequencies of PEs with data cache buffers under varying duty cycle

mance constraint of no underflow.

- Assuming that for some reasons, one or both of the frequencies of the PEs are fixed. Our model will find the lowest duty cycle that will meet the performance constraint of no underflow.

- Assuming that the frequencies and duty cycle are fixed, a procedure similar to that for obtaining Fig. 6 can be used to obtain the optimal scheduling period.

For SoC platforms consisting of data cache instead of the SDRAM, we assume the data cache is ten times more energy costly than the SDRAM. (Or put in another way, assuming that the buffers' power consumption is 10 times higher than our estimates.) In this case, the impact of DFS policies on the total energy is more significant as shown in Fig. 7.

An important observation is that DFS schemes minimizing the energy vary with different frequencies. In the frequency range between 300 Mhz and 467 Mhz, the DFS scheme whose duty cycle is 0.5 gives the minimum energy. However, for the frequency range between 533 MHz and 633 MHz, it is the duty cycle of 0.9 that gives the minimum energy.

6. CONCLUSION

In this paper we presented an analytical framework based on the *variability characterization curves* abstraction. In this framework, we not only identified performance and dynamic power bounds of buffer-constrained SoC platforms in an integrated manner, but also considered the leakage power which is the becoming important for technologies under 90nm.

With parameters of Intel XScale processor and Transmeta Crusoe processor, we applied the framework to study the impact of DFS schemes with varying duty cycle and scheduling periods on the real-time performance constraint. We observed that the scheduling periods have more irregular impacts on underflow possibilities than the duty cycle.

Using our model, we further evaluated the impact of DFS schemes on the total energy. Assuming a lower quality of service, we found that, especially in the low frequency domain, the lowest processor frequency does not guarantee minimum energy because leakage and buffer energy plays a significant role. We also observed that DFS schemes have more significant impacts on platforms containing data cache than platforms with SDRAM only. More importantly, the best DFS scheme changes with frequencies of PEs.

Our model can be used to find the optimal frequency combinations of the PEs assuming a fixed duty cycle, or the optimal duty cycle assuming one or both frequencies of PEs are fixed, or the optimal scheduling period if duty cycle and PE frequencies are fixed.

7. REFERENCES

[1] J.-Y. Le Boudec and P. Thiran. *Network Calculus - A Theory of Deterministic Queuing Systems for the Internet*. LNCS 2050, 2001.

[2] L. Cai and Y.H. Lu. Energy management using buffer memory for streaming data. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 24(2):141 – 152, 2005.

[3] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a MPEG decoder. In *IEEE/ACM International Conference on (ICCAD)*, pages 732 – 737, November 2002.

[4] Kihwan Choi, R. Soma, and M. Pedram. Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding. In *Proc. 41th Design Automation Conference (DAC-41)*, pages 544 – 549, June 2004.

[5] R. Jejurika, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proc. 41st Design Automation Conference (DAC-41)*, 2004.

[6] Yanhong Liu, Alexander Maxiaguine, Samarjit Chakraborty, and Wei Tsang Ooi. Processor frequency selection in energy-aware soc platform design for multimedia applications. In *IEEE Real-Time Systems Symposium (RTSS)*, Lisbon, Portugal, December 2004. IEEE.

[7] A. Maxiaguine, S. Künzli, S. Chakraborty, and L. Thiele. Rate analysis for streaming applications with on-chip buffer constraints. In *ASP-DAC*, 2004.

[8] A. Maxiaguine, S. Künzli, and L. Thiele. Workload characterization model for tasks with variable execution demand. In *7th DATE*, March 2004.

[9] Alexander Maxiaguine, Yongxin Zhu, Samarjit Chakraborty, and Weng-Fai Wong. Tuning soc platforms for multimedia processing: Identifying limits and tradeoffs. In *Proc. Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2004.

[10] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Multimedia for tiny devices: Integrated power management for video streaming to mobile hand held devices. In *ACM Multimedia (MM)*, pages 582 – 591, 2003.

[11] Nathaniel Pettis, Le Cai, and Yung-Hsiang Lu. Dynamic power management for streaming data. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 62 – 65, Newport Beach CA, USA, August 2004. ACM.

[12] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. A framework for evaluating design tradeoffs in packet processing architectures. In *39th Design Automation Conference (DAC)*, 2002.

[13] W. Thies, M. Karczmarek, and S. Amarasinghe. StreamIt: A language for streaming applications. In *11th Conf. on Compiler Construction*, LNCS 2304, pages 179–196, 2002.

[14] Transmeta Crusoe Processor, Transmeta Inc. `http://www.transmeta.com/technology`.

[15] G. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for MPEG-2 video applications. *IEEE Transactions on VLSI*, 12(1), January 2004.

[16] Intel 80200 processor based on intel xscale microarchitecture. http://www.scarpaz.com/processors/intel27341404-Intel80200-Datasheet.pdf.

[17] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time cpu scheduling for mobile multimedia systems. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.