J Supercomput
DOI 10.1007/s11227-007-0132-6

Energy-aware scheduling and simulation methodologies for parallel security processors with multiple voltage domains

Yung-Chia Lin · Yi-Ping You · Chung-Wen Huang · Jenq Kuen Lee · Wei-Kuan Shih · Ting-Ting Hwang

Abstract Dynamic voltage scaling (DVS) and power gating (PG) have become mainstream technologies for low-power optimization in recent years. One issue that remains to be solved is integrating these techniques in correlated domains operating with multiple voltages. This article addresses the problem of power-aware task scheduling on a scalable cryptographic processor that is designed as a heterogeneous and distributed system-on-a-chip, with the aim of effectively integrating DVS, PG, and the scheduling of resources in multiple voltage domains (MVD) to achieve low energy consumption. Our approach uses an analytic model as the basis for estimating the performance and energy requirements between different domains and addressing the scheduling issues for correlated resources in systems. We also present the results of performance and energy simulations from transaction-level models of our security processors in a variety of system configurations. The prototype experiments show that our proposed methods yield significant energy reductions. The proposed techniques will be useful for implementing DVS and PG in domains with multiple correlated resources.

Y.-C. Lin · Y.-P. You · C.-W. Huang · J.K. Lee (✉) · W.-K. Shih · T.-T. Hwang
Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan
e-mail: jklee@cs.nthu.edu.tw

Y.-C. Lin
e-mail: yclin@pllab.cs.nthu.edu.tw

Y.-P. You
e-mail: ypyou@pllab.cs.nthu.edu.tw

C.-W. Huang
e-mail: cwhuang@pllab.cs.nthu.edu.tw

W.-K. Shih
e-mail: wshih@cs.nthu.edu.tw

T.-T. Hwang
e-mail: tingting@cs.nthu.edu.tw

Keywords Security processor · Scheduling · Power management · Dynamic voltage scaling · Power gating · Parallel processing

1 Introduction

The development of techniques to reduce the power consumption of embedded system-on-a-chip (SOC) systems is receiving increasing attention. Pure software techniques [1] and hardware techniques [2, 3] can demonstrably reduce power requirements at the instruction and circuit levels, respectively. Several techniques involving hardware/software collaboration [4, 5] have been proposed to achieve power reduction at the architecture and system levels. Among the developed techniques, dynamic voltage scaling (DVS), power gating (PG), and multiple-domain partitioning are considered the most practical to achieve SOC designs with low power consumption.

DVS [6] can reduce both the dynamic and static power consumption. DVS reduces the dynamic power consumption P by dynamically scaling the supply voltage V_{dd} and the relative running frequency f of the processing element (PE) when maximum-speed operation is not demanded. DVS uses the following equations for the architecture-level estimation of dynamic power consumption:

$$P_{\text{dynamic}} = C \times \alpha \times f \times V_{\text{dd}}^2,$$

$$f = k \times (V_{\text{dd}} - V_{\text{th}})^2 / V_{\text{dd}},$$

where C is the switching capacitance, α is the switching activity, k is a proportionality constant specific to the CMOS technology, and V_{th} denotes the threshold voltage.

The leakage power at the architecture-level is usually estimated using the following equation [7]:

$$P_{\text{static}} = V_{\text{dd}} \times N \times k_{\text{design}} \times I_{\text{leakage}},$$

where N is the number of transistors, k_{design} is the effective transistor width of the cell which depends on the design, and I_{leakage} denotes the normalized leakage current that depends on the silicon technology, threshold voltage, and a sub threshold swing parameter. The equation above indicates that DVS is also effective for leakage power reduction, since it reduces the supply voltages.

Other useful techniques include PG and the use of multiple voltage domains (MVD). PG [8] is an effective technique for leakage power reduction as it reduces the number of active devices by using a sleep circuit that disconnects the supply power from inactive circuitry. The main difficulty in PG technology is issuing off/on commands at the appropriate times so as to minimize performance degradation and maximize the reduction in leakage power [9–13]. In its use of MVD or voltage islands [14], PG also gives more opportunity to reduce the voltage consumption at each decomposed power domain.

There are several issues associated with the application and integration of DVS and PG techniques in MVD with correlated resources that still need to be explored. This article investigates a scalable security processor (SP) as a case study to illustrate how to address the issues about the power-aware voltage/frequency assignment of MVD and the correlation of multiple resources among MVD. In this article, we present a practical scheme to effectively integrate DVS, PG, and the scheduling of multiple domain resources to achieve low energy consumption, for addressing the problem of power-aware task scheduling on scalable cryptographic processors that are equipped with heterogeneous distributed SOC designs.

Our testbed is a scalable SP that has been developed in collaboration with the VLSI design group at our university [15–20]. This project is aimed at producing a configurable prototype of high-performance low-power SPs, and it incorporates DVS, PG, and multiple-domain partitioning in the designed processors. The architecture of the SPs is that of a heterogeneous distributed embedded system in which the PEs are various cryptographic modules. Each cryptographic module is designed to have DVS and PG capabilities.

We propose a novel three-phase iterative scheme equipped with an analytic model that estimates the performance and energy requirements of different components in a system and addresses the scheduling issues for correlated resources in systems. The employed scheduling algorithm in our case study for SP utilizes a heuristic that integrates DVS and PG and thereby increases the total energy saving. We also present our methodologies used for the transaction-level modeling (TLM) of our SPs for both performance and energy simulations. This allows design-space explorations and experiments on a variety of system configurations. The simulators are written in SystemC, and they model the bus and controller at a cycle-accurate transaction level and the cryptographic engines at a timed-transaction level. These proposed techniques are essential to implementing DVS and PG with multiple domain resources that are correlated. Experiments performed in the simulation environments for our SPs have revealed that the discrepancies in the cycles and the power usage of our environment relative to a hardware RTL simulation result are less than 4.55% and 9.3%, respectively. The scheduling results of our proposed mechanisms show energy reductions of up to 32.41% without any degradation in the throughput of the SPs.

The remainder of the article is organized as follows. The architecture used in our target platform and the design of its power management are described in Sect. 2. The proposed iterative scheduling scheme and the corresponding analytical modeling-based approach to handling correlations among PEs and non-PEs are explained in Sect. 3. The implementations of the simulator with MVD and configurable functions, and the experimental setup and the results are described in Sect. 4. Section 5 reviews related works about MVD techniques and scheduling methods, and Sect. 6 concludes the article. Additionally, Appendix 1 details the joint variable-voltage scheduling with power gating which is employed in our case study, and Appendix 2 gives the explanation of the analytical models used in our proposed iterative approach.

2 Configurable SP with multiple-voltage-domain architecture

In this section we briefly describe a configurable architecture for SPs. Variations of this architecture have been used by many network-device manufacturers, such as Broadcom, Hifn [21], Motorola [22], and SafeNet. SPs may include the following key cryptographic functions:

- Data encryption (e.g. DES, 3DES, RC4, and AES).
- User authentication only (e.g. DSS and DSA).
- Hash function (e.g. SHA-1, MD5, and HMAC).
- Public-key encryption (e.g. RSA and ECC).
- Public-key exchange (e.g. Diffie-Hellman Key Exchange).
- Compression (e.g. LZS and Deflate).

The main feature of our SP is the scalable architecture, which is achieved by constructing internal buses therein. Therefore, versatile cryptographic engines can be integrated into the SP by adopting compatible bus interface wrappers. The other configurable parameters include the number of external buses, transfer engines, channels, and the internal buses. In the SP, a descriptor-base DMA controller is implemented to interpret the descriptors and manipulate the cryptographic engines to perform appropriate cryptographic operations. The processing flow and cryptographic operations are handled by descriptors to reduce the control signals from the main processor. The descriptor is a data structure that contains the type of encryption/decryption functions, the encryption key, the length of data, and data-address pointers. The descriptor also has a pointer to the next descriptor, so the DMA module could utilize the link list of descriptors to gather data without much overhead.

Figure 1 shows our architecture, consisting of a main controller, DMA modules, internal buses, and crypto modules [15–20]. The main controller has a slave interface to an external bus that accepts the control signals and returns the operation feedback via the interrupt port. In the main controller, the instruction decoder and the microprogram sequencer are in charge of descriptor decoding and signal passing. The resource-allocation module distributes the resources as the descriptor demands. Process-scheduler and power-management modules are added to the main controller for task scheduling and optimizing power consumption.

The DMA module integrates the master interfaces of external bus with the channels and the transfer engines. Each channel stores the header of its processing descriptor. The transfer engines pass the data from the external bus to dedicated cryp-

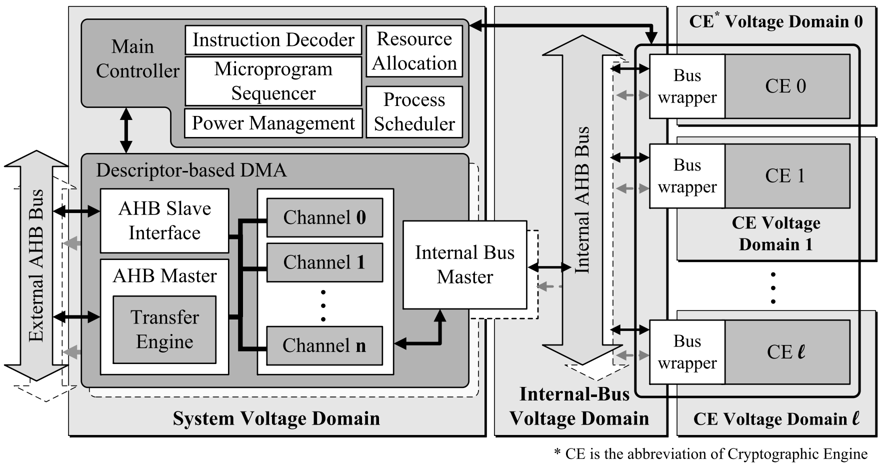


Fig. 1 Security processor (SP) architecture

Table 1 Voltage scaling delays of the DVG

Voltage: scaling up				Voltage: scaling down	
State	Delay (μs)	State	Delay (μs)	State	Delay (μs)
$V_{0\to1.2}$	20	$V_{1.2\to1.5}$	10	$V_{1.8\to1.5}$	8
$V_{0\to1.5}$	35	$V_{1.2\to1.8}$	30	$V_{1.8\to1.2}$	12
$V_{0\to1.8}$	63	$V_{1.5\to1.8}$	16	$V_{1.5\to1.2}$	5

tographic engines via the internal bus which is designed to support multiple layers for high-speed data transmission. Because the execution time of the cryptographic engine may be varied, this cryptographic engine will signal the main controller when operations are complete.

The SP is separated in several voltage domains, where each domain can operate at certain voltage/frequency. If communication signals cross between different voltage domains, we need level converters to downshift/upshift them [23–25]. Therefore, clustering modules with frequent communications into the same voltage domain can reduce the use of level shifters and thereby improve the delays and power consumption. The main controller and descriptor DMAs frequently share the descriptor information that is grouped in the system voltage domain. The internal buses and cryptographic engines have their own voltage domains. The power-management module in the main controller provides software-controllable voltage/frequency adjustment of voltage domains. Components controlled by the power-management module have four main power states: *Full*(1.8 V), *Low*(1.5 V), *Ultralow*(1.2 V), and *Sleep*(0 V). Cooperating with the process-scheduler module, tasks can be assigned power states from among *Full*, *Low*, and *Ultralow*, with PG as the *Sleep* modes.

For supplying multiple operating voltages, the power-management module controls a dynamic voltage generator (DVG) chip that provides the voltages to be used for DVS. The DVG works at 60 MHz and can output three supply voltages (1.2 V, 1.5 V, and 1.8 V) for the SP from a 3.3 V input. Table 1 lists the voltage scaling delays of the DVG. The DVG delays are estimated from the Cadence mixed-mode environment (Verilog_XL™). When the power-management module sets the power state to *Sleep*, it calls the clamping modules to gate off the power supply of specified cryptographic module without scaling the voltage.

To illustrate the kernel operations of SPs, Fig. 2 shows an AES operation in the architecture. The user program first calls the encryption libraries that pack the processing data as descriptors and activate the SP. When the SP controller is aware of a start signal, it retrieves the memory address of descriptors and makes a channel ready to receive the descriptor information. At the "setup channel data" phase, the main controller continually arranges the transfer engine and the master interface of the external bus to move the descriptor information into the channel.

Next, in the "AES operation" phase, after sequentially requesting the AES cryptographic engine, internal bus, transfer engine, and master interface of the external bus, the transfer engine will read data from memory and fill the buffer of the AES cryptographic engine. Once data transmission is complete, the main controller releases the requested resources except for the AES cryptographic engine. When the operation of the AES engine is complete, it will send a signal to the main controller. As the main

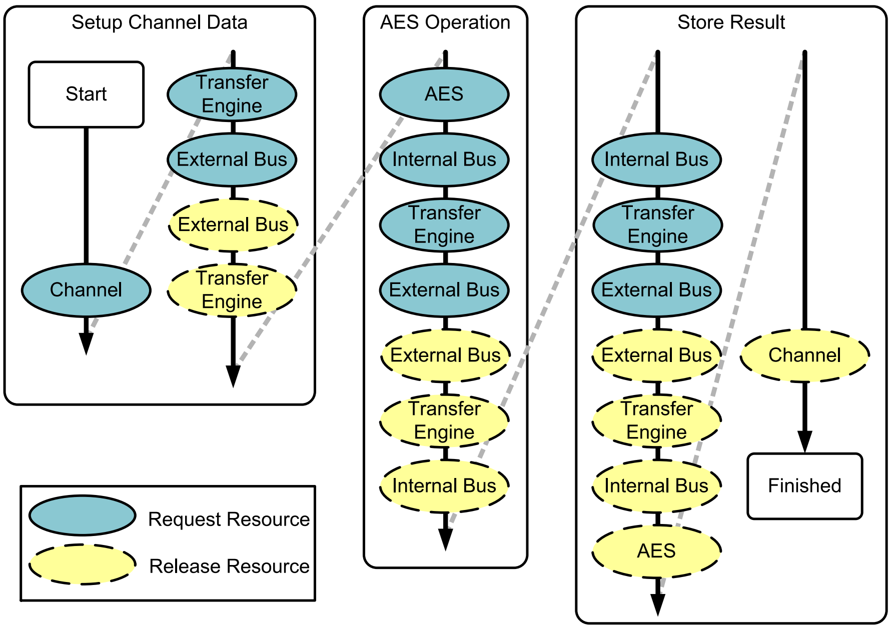


Fig. 2 AES operation on a descriptor-based SP

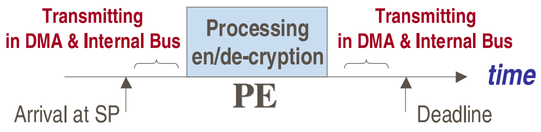
controller is aware of this state, it enters the "store result" phase in which it requests the internal bus, transfer engine, and external-bus-master interface to store the output data. The main controller then releases all the resources. The steps and phases in Fig. 2 can also be interleaved.

3 Power-aware scheduling approach

In this section we discuss the scheduling issues for lowering the power usage in our parallel security architectures, focusing on the problem of scheduling independent periodic tasks. We consider a distributed embedded system containing major PEs (cryptographic modules) which are capable of K-level supply voltages and the PG mode. Moreover, we assume the other non-PE components (such as buses and channels) are capable of DVS. Non-PEs may greatly correlate with the PEs and the correlation cannot be determined before scheduling. For example, in our SP design, a typical encryption or decryption task must take additional time slots beside the processing inside cryptographic engines as shown in Fig. 3. The additional time slots, which are engaged by the transmissions using the shared DMA channels and internal buses, will vary for each task because they depend on the scheduling of other tasks and the DVS/PG state of non-PEs.

To efficiently handle the scheduling problems involving all the components in such a complicated system, we propose a three-phase iterative scheme for power-aware scheduling, namely the iterative analytical approach on MVD (IAA-MVD) scheduling method as follows:

Fig. 3 An illustration of correlation between non-PEs and PEs in SP


Transmitting in DMA & Internal Bus
Processing en/de-cryption
Transmitting in DMA & Internal Bus
time
Arrival at SP
PE
Deadline

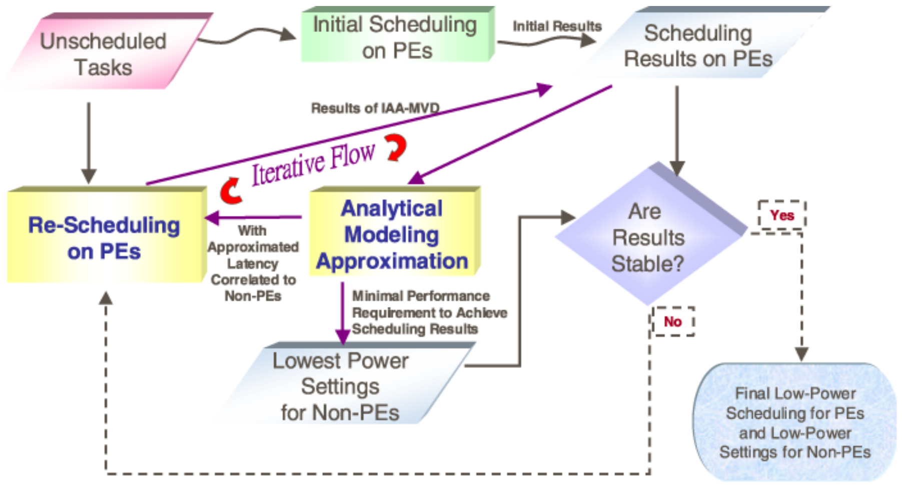

Unscheduled Tasks
Initial Scheduling on PEs
Initial Results
Scheduling Results on PEs
Results of IAA-MVD
Iterative Flow
Re-Scheduling on PEs
With Approximated Latency Correlated to Non-PEs
Analytical Modeling Approximation
Are Results Stable?
Yes
No
Minimal Performance Requirement to Achieve Scheduling Results
Lowest Power Settings for Non-PEs
Final Low-Power Scheduling for PEs and Low-Power Settings for Non-PEs

Fig. 4 Main procedures of the proposed IAA-MVD

1. Employ a scheduling method to schedule only the processing on PEs, including the settings of the running voltage/frequency of each task in major PEs and the appropriate times to invoke PG, and assume the maximum performance of non-PEs (such as bus and channels) to give initial values of additional latency caused by non-PEs for all task.
2. Apply analytic approximation techniques to rapidly determine the running voltages/frequencies of the remaining components in the system. Analytic methods also allow correct estimation of computation latency in PEs caused by these components in the system.
3. Reemploy the scheduling method involved in phase 1 with information generated in phase 2, and deliver the final scheduled setting of each task. Iteratively proceed with phases 2 and 3 until the scheduling results are constant.

The overall procedures of IAA-MVD are illustrated in Fig. 4. The scheduling method used in our IAA-MVD is not dedicated and most of the variable-voltage scheduling methods developed for multiple PEs may be incorporated appropriately into the proposed scheme. In the remainder of this section we brief the scheduling method used in our experiments, which is the extended work based on our previous research efforts [26], and present the second phase of the scheduling method.

3.1 Joint variable-voltage scheduling with power gating for PEs

It is known that the scheduling problem of a set of nonpreemptable independent tasks with arbitrary execution times on fixed-voltage multiprocessors is NP-complete [27].

The application of the reduction techniques reveals that the same scheduling problem on variable-voltage multiprocessors is NP-hard. To deal with the problem of reducing power consumption in real-time systems, we propose a heuristic algorithm to schedule tasks executing in a system with a variable supply voltage. The proposed scheduling algorithm is based on the EDF (Earliest Deadline First) [28] algorithm which, as the name implies, always executes the task with the earliest deadline.

The proposed scheduler maintains a list, called the *reservation list* [29], in which these tasks are sorted by deadlines. Since each periodic task arrives with a certain periodicity, we can obtain the arrivals and deadlines information of tasks in a given interval. Initially, all the tasks are in the list sorted by their deadlines, and the task with the earliest deadline is then picked for scheduling. The scheduler first checks if the task can be executed completely prior to its deadline at a lower voltage without influencing any unscheduled tasks in the reservation list. In this way the scheduler will determine how to schedule tasks at the lowest voltage as possible. The scheduler will also decide if idle PEs can be turned off so as to also maximize the static power reduction. Although the scheduler handles tasks with only homogeneous PEs, it can effectively handles the systems with heterogeneous PEs if tasks can be categorized and only performed on their dedicated types of PEs. In our SP, for instance, AES and RSA tasks can only be processed by AES cryptographic engines and RSA cryptographic engines, respectively. Therefore, we can separate the scheduling of these tasks with their dedicated PEs by our method. The detailed algorithm is explained in Appendix 1. As our focus will be on correlating scheduling with analytical models, we think this scheduling algorithm only represents one of the possible methods on the scheduler for major PEs.

3.2 Voltage/speed selection of non-PEs

We apply analytic modeling techniques to compute the suitable voltages for non-PE components (such as buses and transfer engines) such that the total performance of the system fits the scheduling results of major PEs.

We first describe the analytical model developed for th SP. Suppose the system has multiple PEs that are labeled with an index in the range $1, \dots, l$. Several channels labeled with one such index $(1, \dots, n)$ are built into the control unit for simultaneously accessing the PEs. Data are transferred between channels and PEs across a few internal buses, which are labeled with indexes in the range $1, \dots, m$. We can view each kth PE and jth internal bus as a server with a constant service rate of M_{s_k} and M'_{s_j} bits per second, respectively. Let $P_{i,k}$ be the probability that channel i makes its next service request to PE k. Define Φ_i as the average fraction of the time that the ith channel is not waiting for a service request to be completed from any of PEs and internal buses. Also, let $\Omega_{k,i}$ and $\Omega'_{j,i}$ be the fractions of time spent by the $i$thchannel waiting for service requests to PE k and internal bus j, respectively. Define $\frac{1}{M_{\mathrm{r}_{k,i}}}$ as the ratio of the time that descriptors of the ith channel spend performing overhead of PE service requests (not including the time waiting in queues and having requests serviced by the kth PE) to the total processing data size. Let

$$\eta_k = \sum_{i=1}^{n} P_{i,k} \frac{M_{\mathrm{r}_{k,i}}}{M_{\mathrm{s}_k}} \quad \text{and} \quad \lambda_k = \frac{(1+\epsilon_k) M_{\mathrm{s}_k}}{\sum_{j=1}^{m} M'_{\mathrm{s}_j}},$$

where ϵ_k is the average scaling ratio of the data size throughout the processing performed by the kth PE. The average time that each channel spends on initiating, host memory communication, and descriptor processing (Φ_i) is related to the time spent waiting ($\Omega_{k,i}$ and $\Omega'_{j,i}$) as follows:

$$\Phi_i + \sum_{k=1}^{l} \Omega_{k,i} + \sum_{j=1}^{m} \Omega'_{j,i} = 1,$$

$$\prod_{i=1}^{n} (1 - \Omega_{k,i}) + \eta_k \Phi_i = 1,$$

$$\prod_{i=1}^{n} (1 - \Omega'_{j,i}) + \sum_{k=1}^{l} \eta_k \lambda_k \Phi_i = 1.$$

The detailed model construction and proof are provided in Appendix 2.

Suppose tasks for l PEs are scheduled by the scheduler as described in Sect. 3.1. We can derive Φ_i, $\Omega_{k,i}$, and $\Omega'_{j,i}$ — the metrics of the expected performance — from the scheduling results: the average service rate M_{s_k} of the kth PE and $\eta_k \Phi_i$, which is semantically equal to the utilization of the kth PE due to task assignments. Assuming that the SP has n channels and m internal buses connecting PEs, we can select the appropriate voltages/frequencies of internal buses and transfer engines by solving the previous equations: we use the expected values of M'_{s_j} to evaluate the resulting Φ_i, $\Omega_{k,i}$, and $\Omega'_{j,i}$, and choose a minimal M'_{s_j} that maximizes the system load efficiency, for which Φ_i, $\Omega_{k,i}$, and $\Omega'_{j,i}$ are all positive and Φ_i should be as small as possible.

Apart from the voltage/frequency selection, the proposed analytic modeling techniques is used to revise the latency parameters in the schedulers during phases 2 and 3 of the IAA-MVD scheduling. In realistic environments of the considered systems, the computation times of tasks in PEs should actually include the latency due to data transmission and bus contention. The data-transmission latency can be calculated based on the data size, bus speed, and detailed transfer operations during scheduling. The bus-contention latency, however, cannot be correctly estimated if any run-time information is lacking. Thus we shall use the worst-case estimation of the latency to calculate the task computation time in order to avoid missing any deadlines under run-time conditions. In the first phase of IAA-MVD scheduling scheme, we use the maximum performance settings of internal buses and transfer engines (which relax the slack-time computation) to schedule tasks. We conservatively assume that the worst case for each task is waiting for all tasks in PEs except the one in which it is scheduled to complete their data transmission with the maximum time spent by the largest possible data transmission. The proposed analytic approximation phase of voltage selection estimates the possible low-power settings of internal buses and transfer engines that match the scheduling results, and is also able to estimate a more-accurate worst-case latency in each PE than the theoretical one using $\Omega'_{j,i}$ and $\eta_k \Phi_i$, which would reflect the possible worst-case latency in the scheduling results. We then perform the third phase of the IAA-MVD scheduling scheme that uses values derived in the second phase to obtain the final scheduling results. Due to the monotonicity property of PE usage in our scheduling algorithms in Sect. 3.1, iteratively performing phases 2 and 3 of the IAA-MVD scheduling will converge on a stable scheduling result.

4 Experiments

In this section, we first evaluate the accuracy of performance and power of the SP simulator. We then show how our IAA-MVD scheduling methods can be used for energy reductions to a set of benchmarks.

4.1 Security processor simulation methodology

The simulators for performance evaluations and design space explorations are implemented in SystemC. SystemC can model the hardware/software co-design at the (untimed) functional level, the transaction level [30], and the pin level (RTL level). In the simulator, the bus and controller are modeled at a cycle-accurate transaction level and the cryptographic engines are modeled at a timed-transaction level.

In our experiments, we evaluated the precision of our architecture-level energy simulator, and used this model as a basis for evaluating our proposed scheduling methods.

The first experiment evaluated the cycle accuracy of our configurable SystemC TLM simulation using the Verilog RTL simulation by Cadence NC_Verilog. The architecture comprised one internal bus, one RSA module, two AES modules, two HMAC modules, one RNG module, and one cryptographic DMA with four channels. In RSA operation patterns, the maximum error in the cycles is 0.14% and the mean error is 0.05%, where the error is measured as the difference between two simulation cycles of SystemC TLM and Verilog RTL divided by the simulation cycles of Verilog RTL. In our AES operation patterns, the maximum error is 4.55% and the mean error is 0.21%. The mean errors of the HMAC and RNG modules are less than 0.2%.

In Table 2, the RSA and AES power model in SystemC which established in the simulator are compared with the power results of PrimePower™ (Synopsys) in the Verilog RTL. The errors in the mean powers of the RSA and AES are 0.57% and 9.30%, respectively, where the error is measured as the difference between two power results of SystemC TLM and Verilog RTL divided by the simulation cycles of Verilog RTL.

4.2 Iterative analytical approaches

We evaluated our proposed scheduling methods by implementing a randomized security-task generator to generate benchmark descriptor files for the simulator. The

Table 2 Power values for RSA and AES

Type	Avg. power (SystemC TLM)	Avg. power (Verilog RTL)	Error avg.(%)
RSA	0.2232 W	0.2245 W	0.57
AES	0.0176 W	0.01941 W	9.30

Table 3 Benchmark settings and results

Suite		1	2	3	4	5	6	7	8	9
Arrival distribution		Uniform			Normal			Exponential		
Number of jobs		300								
Jobs/time (µs)		1500		375	1500		375	1500		375
AES:RSA		30:1								
Max data size (bytes)		1280								
Maximum AES deadline (µs)		3072	3430		3072	3430		3072	3430	
Maximum RSA deadline (µs)		13312	15872		13312	15872		13312	15872	
Iterate only once	dynamic energy reduction (%)	28.14	28.21	27.24	16.93	31.33	16.13	16.03	29.05	30.28
	leakage energy reduction (%)	82.33	83.52	83.40	83.57	83.53	83.67	83.51	83.67	83.38
	total energy reduction (%)	28.30	28.38	27.41	17.13	31.49	16.34	16.24	29.22	30.44
Iterate till end	dynamic energy reduction (%)	30.41	32.19	29.01	21.34	34.75	19.37	17.49	32.47	35.08
	leakage energy reduction (%)	83.61	83.77	83.97	84.65	83.98	85.06	84.48	84.37	84.77
	total energy reduction (%)	30.58	32.35	29.17	21.53	34.91	19.57	17.69	32.63	35.23

generator produced simulated operating-system-level jobs of decryption/encryption, with each job having operation types, data sizes, keys and content, arrival times, and deadlines randomized on the basis of an adjustable configuration of job-arrival distributions, job numbers, job density, ratio of distinct operation types, job-size variance, and job-deadline variance. Each generated job was then converted by the generator to the corresponding descriptors that could be executed by the simulator. In our preliminary experiments, we assumed that the SP comprised six AES modules, two RSA modules, five internal buses, and eight channels and transfer engines. The generated benchmarks come from nine test suites with the task-generator configurations listed in Table 3. They can be divided into three types of arrival distribution. Each distribution type has three suites with different task slackness dependent on the job density and job deadline range: the first suite features a high density and a short deadline, the second features a high density and a long deadline, and the third features a low density and a long deadline.

We generated 100 distinct descriptor files for each suite and computed their average energy consumptions for different components from the results of the simulator, as shown in Fig. 5. The bars labeled by **N** are the scheduling results without power management, and others labeled by **P** and **PI** are the results with enabling our proposed power management by iterating only once and iterating till the results unchanged respectively. The numbers of iterations in the results labeled by **PI** vary with respect to the workloads, averaging between five and six. The energy overhead of applying DVS and PG is too low to appear clearly on the charts, as is the leakage of RSA modules. The top chart gives the energy reduction for AES modules, the middle

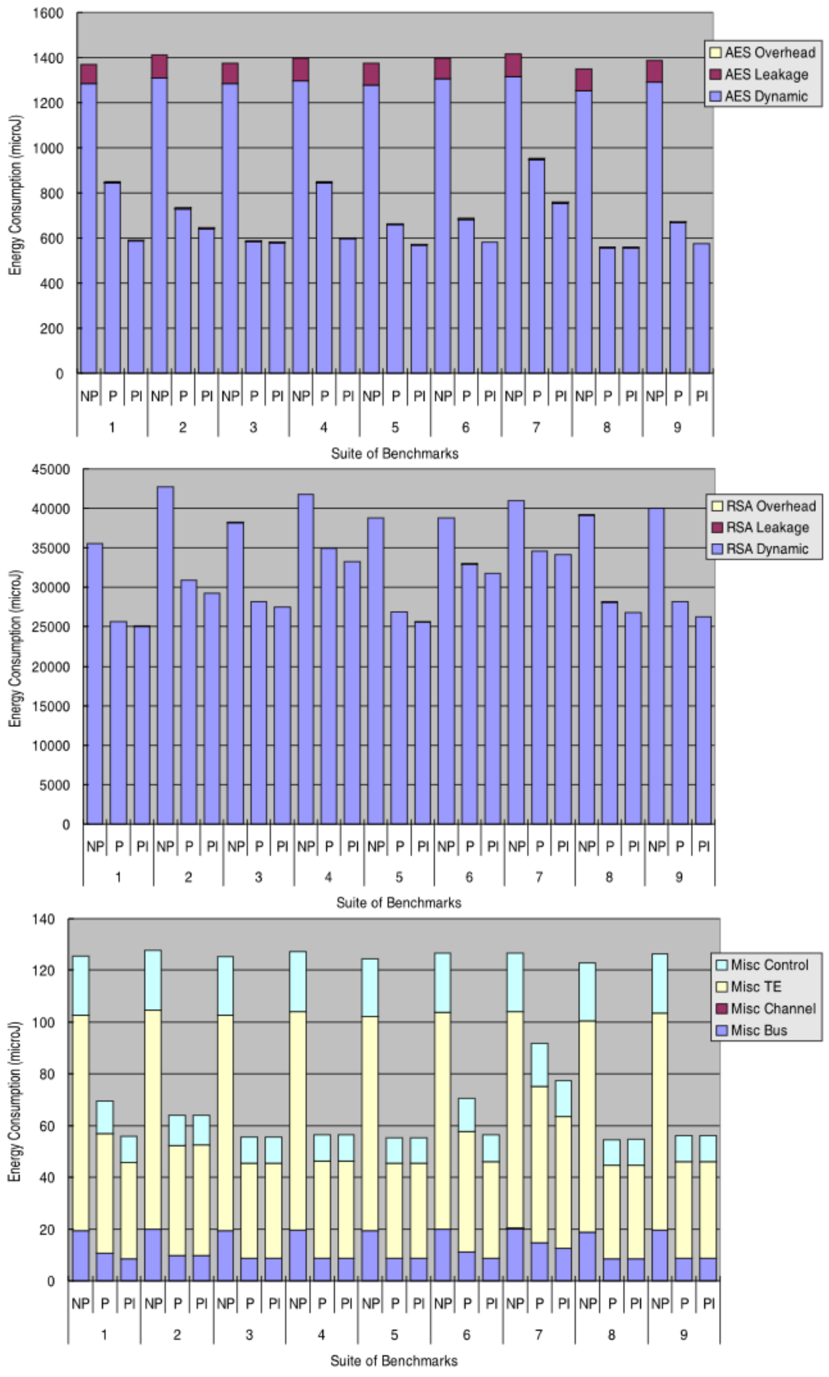


Fig. 5 Energy consumption estimated by the simulator with ("P"—iterate only once; "PI"—iterate till end) and without ("N") our proposed power management

chart gives the energy reduction for RSA modules, and the bottom chart shows the energy reduction for non-PE components which are assigned by our analytic approximation phase, for all benchmark suites. The simulations also finally confirm that no deadline is missed using the latency approximation by our analytical modeling techniques. Although the energy consumptions is dominated by RSA operations in our experimental architecture and workloads, the charts show that our scheme performs well for all components in the system. Moreover, combining PG with DVS scheduling reduces the leakage by up to 85.06% (as indicated in Table 3), which is expected to become even more important as the leakage power increases in chips constructed using CMOS processes down to below 0.13 μm [31, 32]. Table 3 also gives the overall energy reduction for all test suites. The bottom row of the table indicates that a total energy reduction of up to 35.23% was achieved by our power-aware scheduling scheme. Furthermore, the results by using only one iteration show that limiting the number of iterations may still provide adequate energy reduction, which could be feasible for incorporating methods of online scheduling.

5 Related work and discussion

5.1 Low-power design

In hardware design, the operating voltage and clock rate are the two primary factors affecting the power consumption and processing speed. There is a great demand for a hardware device to exhibit sufficient performance with low power consumption. Clock gating (CG) [2] refers to inactivating the input clocks in a hardware circuit when there is no work to be done. For sequential circuits, the clock signal is considered to be a major contributor to the dynamic power dissipation since the clock is the only input that switches all the time and usually is highly loaded, hence the CG technique is quite helpful. Frequency scaling (FS) [33] supports different operating frequencies. When the operating requirements are low, the frequency is scaled down to reduce the number of wasted/useless cycles whilst keeping the supply constant. The PG [8] technique clamps the supply voltage of the hardware module. It uses high-V_{th} transistors to switch the supply voltage and reduce the leakage power. DVS supports different voltage levels and switches the operating voltage during run-time. It must employ multiple-V_{th} transistors and the DVG. Because frequency scaling accompanies voltage scaling at run-time, this technique is also called as dynamic voltage and frequency scaling (DVFS) [34]. In the multiple clock domain (MCD) [35] technique, the hardware is separated into modules based on certain attributes, with each domain having its own clock. The synchronization of domains and power reduction in this technique are referred to as The globally-asynchronous locally-synchronous (GALS) design is one of the popular techniques to implement MCD systems, and the available power-reduction techniques are also beneficial for this kind of implementation [36]. The supply voltage of MCD may be uniform. If the voltage level differs between the domains, the hardware design is in MVD or multiple voltage islands [14]. To support the required throughput with lower power consumption in the design with multiple voltage domains, techniques such as CG, PG, FS, and DVFS could be applied.

5.2 Variable-voltage scheduling

Variable-voltage scheduling manages tasks with execution deadlines and reduces the energy consumption by lowering the running voltage/frequency, whilst ensuring that all tasks finish execution before their deadlines. Several variable-voltage scheduling techniques have been developed to exploit DVS for power reduction in real-time systems. For example, workload descriptions have been used to statically schedule tasks on the basis of energy efficiency for variable-voltage processors [37], and a heuristic non-preemptive scheduling algorithm have been proposed for independent tasks executing on variable-voltage processors [38]. DVS scheduling for dependent tasks in distributed embedded systems has also been studied [39, 40]. However, none of the developed techniques considers task scheduling employing the advantages of both DVS and PG to jointly reduce dynamic power dissipation and static power leaks, or addresses the issue that in a complex SOC design there may be additional DVS-enabled components that correlate the operation of DVS-enabled PEs and affect the task scheduling for the entire system. Also, some researches have been exploited to use analytical techniques for minimizing energy in a two-device data flow chain [41], and optimizing energy consumption on systems of multiple voltage islands based on rate and latency constraints [42], while we primarily focus on the systems with task-based scheduling problems on PEs, differentiated from their studies.

6 Conclusions

The growing complexity and numbers of transistors involved in the SOC designs with multiple domains demand new and more-effective power-reduction techniques than ever. In this article, we have presented a practical method to integrate and extend the existing dynamic and static power-reduction mechanisms for increasing the power efficiency in complex distributed embedded systems. A novel three-phase iterative scheme was proposed to effectively estimate the performance and energy requirements of various components correlated in systems. By using of an analytical approximation, this approach selects voltages/frequencies of minor components (and not major PEs) so as to decrease the complexity of the overall scheduling problem in systems of with multiple correlated domains. We have demonstrated the proposed method on a fast SP capable of executing various cryptographic computations in parallel, which is a significant application of a complex SOC design. The experimental results reveal that our power-management scheme achieves a significant power reduction on our testbed, and may also profit other complex SOC designs.

Appendix 1 Joint variable-voltage scheduling with power gating

Slack-time computation

We first define slack time in the scheduling. Suppose we are going to schedule task T_i, and there are still $(n-i)$ unscheduled tasks (i.e., $T_{i+1}, T_{i+2}, \dots, T_n$) in the reservation list. The slack time $\delta_i(V)$ is the maximum period allowed for T_i while the remaining $(n-1)$ tasks are scheduled at supply voltage V in reverse order. To obtain the information for T_i, we first build a pseudo scheduler for the $(n-i)$ tasks with the following behaviors. The $(n-i)$ tasks are scheduled in a reversed manner, in which deadlines are treated as arrivals and the arrivals as deadlines, and starts from the point of the latest deadline (i.e., d_n is the deadline of T_n) via the well-known earliest deadline first (EDF) algorithm [28]. We then record the time of the end point of the pseudo schedule as $\lambda_i(V)$.

The slack time of the pseudo schedule at a supply voltage V can be obtained from the following equation:

$$\delta_i(V) = \lambda_i(V) - \text{Maximum}(a_i, f_{i-1}),$$

where a_i is the arrival time of T_i, f_{i-1} is the finishing time of the last task T_{i-1}. Figure 6 gives an example of the slack-time computation, in which there are four tasks in the reservation list. Here two reservation lists are maintained: one is created by a pseudo scheduler to schedule tasks at the lowest voltage, and the other is compiled by the highest-voltage scheduler. The slack time $\delta_i(V_H)$ and $\delta_i(V_L)$ is the time from the finishing time of the last task to the end point of the reservation list from the highest- and lowest-voltage schedulers, respectively. If we consider the overhead of DVS, the highest-voltage scheduler should add the maximum time-overhead of DVS to f_{i-1} to compute $\delta_i(V_\mathrm{H})$. During the scheduling an exception is flagged if any deadline cannot be met when scheduling at the highest voltage. This is because the forward and backward scheduling are equivalent on the qualification of time-constrained tasks, and hence if there is no backward scheduling there is also no forward scheduling. However, when the low voltage is supplied, we ignore deadline misses in the pseudo scheduling.

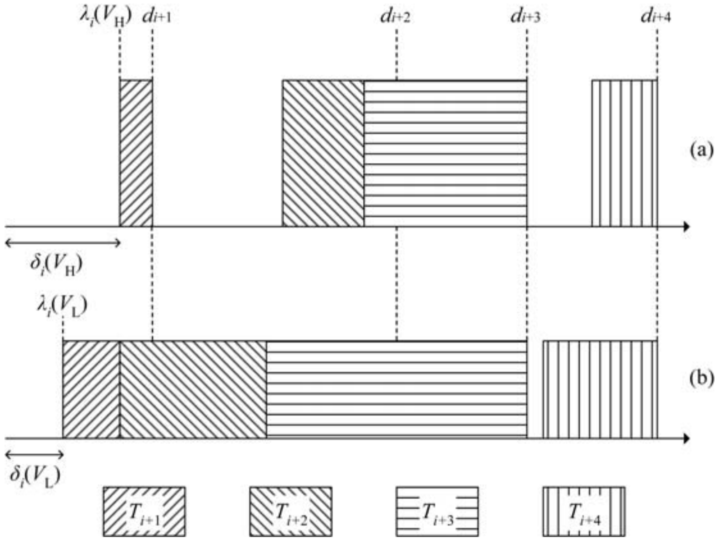


Fig. 6 Examples of slack-time computation while scheduling T_i: (**a**) tasks performed at the highest voltage; (**b**) tasks performed at the lowest voltage

Scheduling algorithm

The proposed scheduling algorithm is based on the EDF algorithm which, as the name implies, always executes the task with the earliest deadline. Figure 7 lists the algorithm. Assume that there are n periodic tasks to be scheduled. First, we sort the tasks in ascending order by deadlines, namely $T_1, T_2, \dots, T_n$, and put them in a list of unscheduled tasks, i.e., the reservation list. We then extract each task from the list on the basis of the schedule. Suppose the system provides m PEs, and each PE is capable of K-level supply voltages, where level 1 represents the lowest voltage and level K represents the highest voltage. In order to reduce the complexity and expense of maintaining K reservation lists, we maintain two reservation lists: one for the pseudo scheduler at the lowest voltage and the other for the scheduler at the highest voltage. Steps 1–3 in Fig. 7 describe these procedures. For utilizing PG capabilities, we attempted to both make tasks run continuously and concatenate the idle time because PG mechanisms cost much more than DVS in terms of both performance and power. Next, in step 4 we compute the slack time for task T_i with both $\delta_i(V_\mathrm{H})$ and $\delta_i(V_\mathrm{L})$.

Real-time scheduling algorithm with variable-voltage reservation lists in multiple PEs

```
 Input:  $n$ unscheduled periodic tasks and $m$ PEs
Output:  Schedule of gating commands and the $n$ tasks
         with variable supply voltages at $PE_{1,\dots,m}$

1. Sort tasks by deadlines in ascending order; i.e., $T_1, T_2, \dots, T_n$.
2. Put them in the *reservation list* of
   the target PE ($PE_j$). Initially, $j = 1$.
3. Remove the first task, namely $T_i$, that has the earliest deadline from
   the *reservation list*. Repeat steps 3–6 while the list is not empty.
4. Compute the slack time for task $T_i$ with both the highest and
   lowest voltage pseudo schedulers, i.e., $\delta_i(V_\mathrm{H})$ and $\delta_i(V_\mathrm{L})$.
5. Compute the computation time of $T_i$ at the highest
   and lowest voltages, i.e., $c_i(V_\mathrm{H})$ and $c_i(V_\mathrm{L})$.
6. Letting $o_t(i)$ be the voltage scaling time, schedule $T_i$ using the following rules:
   - If $c_i(V_\mathrm{L}) + o_t(i) \le \delta_i(V_\mathrm{L})$, schedule $T_i$ for $PE_j$ at $V_\mathrm{L}$ if possible[†].
   - If $\delta_i(V_\mathrm{L}) < c_i(V_\mathrm{L}) + o_t(i) \le \delta_i(V_\mathrm{H})$, call the *decision algorithm*.
   - If $c_i(V_\mathrm{L}) + o_t(i) > \delta_i(V_\mathrm{H})$ and
     - if $c_i(V_\mathrm{H}) + o_t(i) \le \delta_i(V_\mathrm{H})$, schedule $T_i$ for $PE_j$ at $V_\mathrm{H}$.
     - if $c_i(V_\mathrm{H}) + o_t(i) > \delta_i(V_\mathrm{H})$, put $T_i$ in an unscheduled list $L_\mathrm{un}$.
7. Check the idle time of $PE_j$ and insert gating commands if possible[‡].
8. If $PE_j$ is the last available PE and the list $L_\mathrm{un}$ is not empty,
   then report a possible failure of real-time scheduling.
9. If the list $L_\mathrm{un}$ is not empty, let $j = j + 1$ and use the list
   $L_\mathrm{un}$ as the *reservation list* of the target $PE_j$. Next, go to step 3.
10.If $j < m$, then gate off $PE_{j+1}, \dots, PE_m$ all the time.

[†]Schedule $T_i$ at $V_\mathrm{L}$ if deadline is met and the energy overhead is acceptable.
[‡]Gate on/off if tasks are unaffected and the energy overhead is acceptable.
```

Fig. 7 Reservation-list scheduling algorithm for variable-voltage problems in multiple PEs

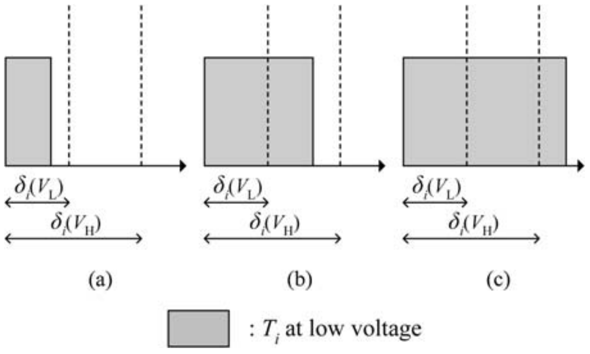


Fig. 8 Scenarios of scheduling task T_i

The slack time $\delta_i(V)$ represents the maximum time interval allowed for task T_i to execute while all the remaining tasks in the reservation list are scheduled in reverse order with supply voltage V. In step 5 we compute the computation time of task T_i at both the highest and lowest voltages, denoted as $c_i(V_H)$ and $c_i(V_L)$. In step 6 we compare $c_i(V_H)$ and $c_i(V_L)$ with $\delta_i(V_L)$ and $\delta_i(V_H)$ to decide which voltage should be applied to the task. This algorithm results in three possible scenarios, as depicted in Fig. 8:

1. $c_i(V_L)$ plus the time overhead of voltage scaling is smaller than or equal to $\delta_i(V_L)$. If the energy overhead of voltage scaling is less than the energy saving, we can schedule task T_i at the lowest voltage without affecting any future task because there are no overlaps between task T_i and the unscheduled tasks while those tasks are assumed to be executed at the lowest voltage.
2. $c_i(V_L)$ plus the time overhead of voltage scaling is larger than $\delta_i(V_L)$ and smaller than or equal to $\delta_i(V_H)$. In this case we call the decision algorithm described in Sect. 6 to determine the voltage at which task T_i should be scheduled. This algorithm weights the alternatives to optimize the overall costs, using a criterion such as the power consumption.
3. $c_i(V_L)$ plus the time overhead of voltage scaling is larger than $\delta_i(V_H)$. This means that it is impossible for task T_i to complete executing by its deadline at any voltage lower than the highest voltage, and hence it must be scheduled at the highest voltage. If task T_i is unschedulable for the current PE, we put it in a new list called L_{un} that contains all unschedulable tasks.

In step 7 we check the remaining idle time between the scheduled tasks in the current PE and determine PG commands to be inserted if this reduces the energy consumption. In steps 8 and 9, if the list L_{un} generated in step 6 is not empty, we use this list as the reservation list for the next-available PE and schedule it using the procedures in steps 3–6. If no PE is available for scheduling, the scheduler should report failure. At the last step we turn off all unused PEs via PG to maximize both the static and dynamic power savings.

Decision algorithm

Assume that we are scheduling task T_i and that the computation time of T_i at the low voltage, $c_i(V_L) + o_t(i)$ is larger than $\delta_i(V_L)$ and smaller than or equal to $\delta_i(V_H)$. Another viewpoint is that the finishing time of task T_i at the low voltage falls within the

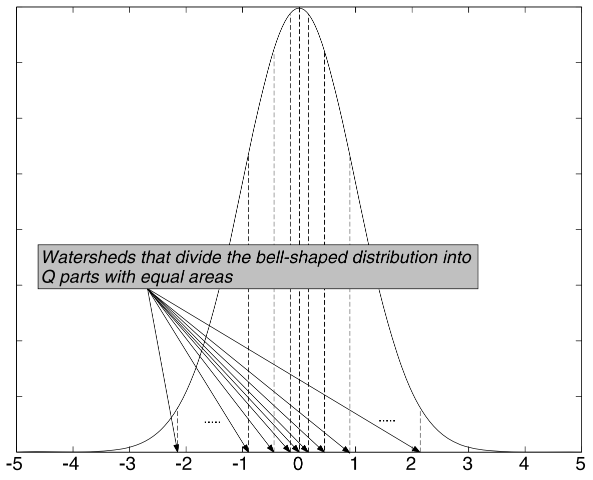


Fig. 9 Watersheds of a population

region bounded by $\lambda_i(V_L)$ and $\lambda_i(V_H)$. To achieve the objective of power reduction, we propose several algorithms for deciding at which voltage tasks should be scheduled when weighting trade-offs between tasks. We use a probability density function,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad \text{where } -\infty < x < \infty,$$

which defines the probability density function for the value x of a random observation from the population [43], to divide the population of a group into Q equal parts in terms of area under the distribution, and then schedule tasks at levels corresponding to the parts that the tasks belong to. In other words, let $W^1, W^2, \ldots,$ and W^{Q-1} be a demarcation that separates the population into Q parts (as shown in Fig. 9); a task will be scheduled at level t if its value falls between W^{t-1} and W^t. The detailed algorithms are described as follows:

1. *Reservation list with first-come first-served scheduling*
   Tasks are always scheduled at the lowest voltage possible without missing deadlines. This algorithm does not apply a cost model to the decision.
2. *Reservation list with average power consumption*
   We use the switching activity α_i to select the voltage level for T_i. We schedule a task at level $(Q - \tau + 1)$ if

$$\int_{-\infty}^{W_\alpha^\tau} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(W_\alpha^\tau-\mu)^2}{2\sigma^2}} = \frac{\tau}{Q},$$

   where W_α^τ denotes the τth watershed of the population of switching activities of tasks.

Appendix 2 Analytical models

Here we detail the analytic model developed for the security architectures given in Sect. 2 as follows. We consider the typical execution process of an operation in the system described in Sect. 2. The execution of each operation can be viewed as a procedure in which a channel requests an internal bus twice to serve the data transmission and requests a PE to manipulate the data. Assume that each channel execution can be treated as an exponentially distributed random process that produces sets of service requests with three correlated operations in the following fixed order: two for the internal bus, and one for the PE. Following the notation in Sect. 3.2, let $system\_cycles_i$ be the total time spent by the ith channel on transmitting over the system bus (including accessing the host memory, descriptor processing, and idling). We can now define $request\_cycles_i$, which has two elements: (i) the total time spent by the ith channel on preparing the PE request, the internal bus request, and processing time, and (ii) the total time for the data to traverse the channel, internal buses, and PEs. Now let $channel\_cycles_i$ be

$$channel\_cycles_i = system\_cycles_i + request\_cycles_i.$$

We define $M_{\mathrm{r}_{k,i}}$ as

$$M_{\mathrm{r}_{k,i}} = \frac{data\_amount_{k,i}}{channel\_cycles_i},$$

which is the ratio of the amount of data requested to the time that descriptors of the ith channel spend performing the overhead of PE service requests, not including the time spent waiting in queues and having requests serviced by the kth PE.

If we neglect the interaction between channels and assume that all internal buses are utilizable by all channels and PEs, we have the following analytic model developed on top of a previous parallelizing theorem [44, 45]:

Theorem 1 *Let*

$$\eta_k = \sum_{i=1}^{n} P_{i,k} \frac{M_{\mathrm{r}_{k,i}}}{M_{\mathrm{s}_k}} \quad \textit{and} \quad \lambda_k = \frac{(1+\epsilon_k) M_{\mathrm{s}_k}}{\sum_{j=1}^{m} M'_{\mathrm{s}_j}},$$

where ϵ_k is the average scaling ratio of the data size throughout the processing by the kth PE. The average time that each channel spends performing initiation, host memory communication, and descriptor processing (Φ_i) is related to the time spent waiting ($\Omega_{k,i}$ and $\Omega'_{j,i}$) as follows:

$$\Phi_i + \sum_{k=1}^{l} \Omega_{k,i} + \sum_{j=1}^{m} \Omega'_{j,i} = 1,$$

$$\prod_{i=1}^{n} (1 - \Omega_{k,i}) + \eta_k \Phi_i = 1,$$

$$\prod_{i=1}^{n} (1 - \Omega'_{j,i}) + \sum_{k=1}^{l} \eta_k \lambda_k \Phi_i = 1.$$

Proof The first equation simply infers time conservation. Letting $C_{k,i}$ be the average channel-i-to-PE-k request cycle time for the system and *total_cycle* be the total operation time per request yields

$$\frac{1}{C_{k,i}} = \frac{data\_amount_{k,i}}{total\_cycles} \tag{1}$$

on average. By observing the workloads, we can compute $M_{\mathrm{r}_{k,i}}$ (which is the ratio of the amount of requested data to the channel cycles). Based on the definition of $M_{\mathrm{r}_{k,i}}$ and equation (1), we obtain

$$\frac{1}{M_{\mathrm{r}_{k,i}} C_{k,i}} = \frac{channel\_cycles_i}{total\_cycles} = \Phi_i. \tag{2}$$

Moreover, we define

$$\delta_{k,i} = \begin{cases} 1 & \text{if channel } i \text{ is not waiting for module } k, \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\delta'_{j,i} = \begin{cases} 1 & \text{if channel } i \text{ is not waiting for bus } j, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Let μ_k be the probability that PE k is busy and μ'_j be the probability that internal bus j is busy. We have

$$\mu_k = 1 - E(\delta_{k,1}\delta_{k,2}\cdots\delta_{k,n}), \tag{5}$$

$$\mu'_j = 1 - E(\delta'_{j,1}\delta'_{j,2}\cdots\delta'_{j,n}), \tag{6}$$

where $E(\nu)$ is the expected value of the random variable ν. Therefore, $\mu_k M_{\mathrm{s}_k}$ and $\mu'_j M'_{\mathrm{s}_j}$ are the rates of completed requests to PE k and internal bus j, respectively. When the system is in equilibrium, $\mu_k M_{\mathrm{s}_k}$ is equivalent to the rate of submitted requests to PE k, and $\mu'_j M'_{\mathrm{s}_j}$ is equivalent to the rate of submitted requests to internal bus j. Since $\sum_{i=1}^{n} \frac{P_{i,k}}{C_{k,i}}$ is the total rate of submitted requests to PE k from all channels,we have the equivalence

$$\sum_{i=1}^{n} \frac{P_{i,k}}{C_{k,i}} = \mu_k M_{\mathrm{s}_k}. \tag{7}$$

Likewise, $\sum_{k=1}^{l} \sum_{i=1}^{n} \frac{P_{i,k}}{C_{k,i}}$ is the average rate of submitted requests to all internal buses from all channels. Due to the law of data indestructibility, $\sum_{k=1}^{l} \sum_{i=1}^{n} \frac{P_{i,k}(1+\epsilon_k)}{C_{k,i}}$ is the average rate of submitted requests to all internal buses from all channels and all PEs. Accordingly, we have the following equivalence:

$$\sum_{k=1}^{l} \sum_{i=1}^{n} \frac{P_{i,k}(1+\epsilon_k)}{C_{k,i}} = \sum_{j=1}^{m} \mu'_j M'_{\mathrm{s}_j}. \tag{8}$$

By combining (2), (5), (6), (7), and (8), we get

$$\begin{cases} \eta_k = \sum_{i=1}^{n} P_{i,k} \frac{M_{\mathrm{r}_{k,i}}}{M_{\mathrm{s}_k}}, \\ E(\delta_{k,1}\delta_{k,2}\cdots\delta_{k,n}) + \eta_k \Phi_i = 1, \end{cases} \qquad \begin{cases} \lambda_k = \frac{(1+\epsilon_k) M_{\mathrm{s}_k}}{\sum_{j=1}^{m} M'_{\mathrm{s}_j}}, \\ E(\delta'_{j,1}\delta'_{j,2}\cdots\delta'_{j,n}) + \sum_{k=1}^{l} \eta_k \lambda_k \Phi_i = 1. \end{cases} \tag{9}$$

Nevertheless, since both $\delta_{k,i}$ and $\delta'_{j,i}$ are binaries, we have by symmetry

$$E(\delta_{k,i}) = 1 - \Omega_{k,i} \quad \text{and} \quad E(\delta'_{j,i}) = 1 - \Omega'_{j,i}$$

for each channel i. We now make a critical approximation by assuming that all the channels have noncorrelated activities, and get

$$\begin{aligned} E(\delta_{k,1}\delta_{k,2}\cdots\delta_{k,n}) &= E(\delta_{k,1})E(\delta_{k,2})\cdots E(\delta_{k,n}) = \prod_{i=1}^{n}(1-\Omega_{k,i}), \\ E(\delta'_{j,1}\delta'_{j,2}\cdots\delta'_{j,n}) &= E(\delta'_{j,1})E(\delta'_{j,2})\cdots E(\delta'_{j,n}) = \prod_{i=1}^{n}(1-\Omega'_{j,i}). \end{aligned} \tag{10}$$

The result follows. □

Acknowledgements This work was supported in part by Ministry of Economic Affairs under grant no. 95-EC-17-A-01-S1-034 and 96-EC-17-A-01-S1-034, by National Science Council under grant no. 95-2220-E-007-001 and 95-2220-E-007-002 in Taiwan.

References

1. Lee CR, Lee JK, Hwang TT, Tsai SC (2003) Compiler optimizations on vliw instruction scheduling for low power. ACM Trans Des Automat Electron Syst 8(2):252–268
2. Devadas S, Malik S (1995) A survey of optimization techniques targeting low power vlsi circuits. In: Proceedings of the design automation conference, pp 242–247
3. Singh D, Rabaey J, Pedram M, Catthoor F, Rajgopal S, Sehgal N, Mozdzen T (1995) Power conscious cad tools and methodologies: a perspective. Proc IEEE 83:570–594
4. Hsu CH, Kremer U, Hsiao M (2001) Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. In: Proceedings of the 2001 international symposium on low power electronics and design
5. Azevedo A, Issenin I, Cornea R, Gupta R, Dutt N, Veidenbaum A, Nicolau A (2002) Profile-based dynamic voltage scheduling using program checkpoints. In: Proceedings of the conference on design, automation and test in Europe
6. Weiser M, Welch B, Demers A, Shenker S (1994) Scheduling for reduced CPU energy. In: Proceedings of USENIX symposium on operating systems design and implementation (OSDI), pp 13–23
7. Butts JA, Sohi GS (2000) A static power model for architects. In: Proceedings of the international symposium on microarchitecture, pp 191–201
8. Powell MD, Yang SH, Falsafi B, Roy K, Vijaykumar TN (2000) Gated-vdd:a circuit technique to reduce leakage in deep-submicron cache memories. In: Proceedings ISLPED
9. You YP, Huang CW, Lee JK (2005) A sink-n-hoist framework for leakage power reduction. In: Proceedings EMSOFT
10. You YP, Lee CR, Lee JK (2002) Compiler analysis and support for leakage power reduction on microprocessors. In: Proceedings LCPC
11. You YP, Lee CR, Lee JK (2006) Compilers for leakage power reductions. ACM Trans Des Autom Electron Syst 11(1):147–166

12. Duarte D, Tsai Y, Vijaykrishnan N, Irwin MJ (2002) Evaluating run-time techniques for leakage power reduction. In: Proceedings ASPDAC
13. Rele S, Pande S, Onder S, Gupta R (2002) Optimizing static power dissipation by functional units in superscalar processors. In: Proceedings of the international conference on compiler construction, pp 261–275
14. Lackey DE, Bednar PSZTR, Stout DW, Gould SW, Cohn JM (2002) Managing power and performance for system-on-chip designs using voltage islands. In: Proceedings of the 2002 IEEE/ACM international conference on computer-aided design, pp 195–202
15. Su CY, Hwang SA, Chen PS, Wu CW (1999) An improved Montgomery algorithm for high-speed rsa public-key cryptosystem. IEEE Trans Very Large Scale Integr Syst 7:280–284
16. Hong JH, Wu CW (2003) Cellular array modular multiplier for the rsa public-key cryptosystem based on modified booth's algorithm. IEEE Trans Very Large Scale Integr Syst 11:474–484
17. Lin TF, Su CP, Huang CT, Wu CW (2002) A high-throughput low-cost aes cipher chip. In: 3rd IEEE Asia–Pacific conference ASIC
18. Su CP, Lin TF, Huang CT, Wu CW (2003) A highly efficient aes cipher chip. In: ASP-DAC
19. Wang MY, Su CP, Huang CT, Wu CW (2004) An hmac processor with integrated sha-1 and md5 algorithms. In: ASP-DAC
20. Lee MC, Huang JR, Su CP, Chang TY, Huang CT, Wu CW (2002) A true random generator design. In: 13th VLSI design/CAD symposium
21. Hifn (2003) 7954 security processor Data Sheet
22. Gammage N, Waters G (2003) Securing the smart network with Motorola security processors
23. Chang JM, Pedram M (1997) Energy minimization using multiple supply voltages. IEEE Trans Very Large Scale Integr Syst 5(4)
24. Yu CC, Wang WP, Liu BD (2001) A new level converter for low-power applications. In: The 2001 IEEE international symposium on circuits and systems, pp 113–116
25. ARM (2004) Intelligent energy controller technical overview
26. You YP, Lee CR, Lee JK (2001) Real-time task scheduling for dynamically variable voltage processors. In: Proceedings of the IEEE workshop on power management for real-time and embedded systems
27. Stankovic JA, Spuri M, Natale MD, Buttazzo G (1995) Implications of classical scheduling results for real-time systems. Computer 28(6):16–25
28. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard read-time environment. J ACM 20(1):46–61
29. Shih WK, Liu JWS (1996) On-line scheduling of imprecise computations to minimize error. SIAM J Comput 25(5):1105–1121
30. Pasricha S (2002) Transaction level modeling of soc with Systemc 2.0. Technical report, Synopsys Users Group Conference
31. Semiconductor Industry Association (2003) International technology roadmap for semiconductors 2003 edition. Technical report
32. Doyle B, Arghavani R, Barlage D, Datta S, Doczy M, Kavalieros J, Murthy A, Chau R (2002) Transistor elements for 30 nm physical gate lengths and beyond. Intel Technol J 6:42–54
33. Kim JM, Chae SI (1996) New mpeg2 decoder architecture using frequency scaling. In: IEEE international symposium on circuits and systems. ISCAS '96, vol 4, pp 253–256
34. Pouwelse J, Langendoen K, Sips H (2001) Dynamic voltage scaling on a low-power microprocessor. In: 7th ACM international conference on mobile computing and networking (Mobicom), Rome, Italy, pp 251–259
35. Semeraro G, Magklis G, Balasubramonian R, Albonesi D, Dwarkadas S, Scott M (2002) Dynamic frequency and voltage control for a multiple clock domain microarchitecture
36. Iyer A, Marculescu D (2002) Power and performance evaluation of globally asynchronous locally synchronous processors. In: Proceedings 29th annual international symposium on computer architecture, pp 158–168
37. Pouwelse J, Langendoen K, Sips H (2001) Energy priority scheduling for variable voltage processors. In: Proceedings ISLPED
38. Hong I, Kirovski D, Qu G, Potkonjak M, Srivastava MB (1999) Power optimization of variable-voltage core-based systems. IEEE Trans Comput Aided Des 18(12):1702–1714
39. Luo J, Jha NK (2000) Power-conscious joint scheduling of periodic task graphs and aperiodic tasks in distributed real-time embedded systems. In: Proceedings ICCAD, pp 357–364
40. Luo J, Jha N (2002) Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems. In: Proceedings ASPDAC

41. Rao R, Vrudhula S (2004) Energy optimization for a two-device data flow chain. In: Proceedings of the 2004 international conference on computer aided design, pp 268–274
42. Niyogi K, Marculescu D (2005) Speed and voltage selection for gals systems based on voltage/frequency islands. In: Proceedings of the ACM/IEEE Asia–Pacific design automation conference, China
43. Lapin LL (1997) Modern engineering statistics. Wadsworth, Belmont
44. Hwang K, Briggs F (1984) Computer architecture and parallel processing. McGraw–Hill, New York
45. Bodin F, Windheiser D, Jalby W, Atapattu D, Lee M, Gannon D (1990) Performance evaluation and prediction for parallel algorithms on the bbn gp1000. In: Proceedings of the 4th ACM international conference on supercomputing, pp 401–403