A Survey on Low-Power Techniques for Single and Multicore Systems

Mahroo Zandrahimi
Delft University of Technology
Delft, the Netherlands
m.zandrahimi@tudelft.nl

Zaid Al-Ars
Delft University of Technology
Delft, the Netherlands
z.al-ars@tudelft.nl

ABSTRACT

This paper surveys state of the art low-power techniques for both single and multicore systems. Based on our proposed power management model for multicore systems, we present a classification of total power reduction techniques including both leakage and active power. According to this classification, three main classes are discussed: power optimization techniques within the cores, techniques for the interconnect and techniques applicable for the whole multicore system. This paper describes several techniques from these classes along with a comparison. For the whole multicore system, we focus on adaptive voltage scaling and propose a comprehensive taxonomy of adaptive voltage scaling techniques, while considering process variations.

1. INTRODUCTION

Power has been one of the primary design constraints and performance limiters in the semiconductor industry such that reducing power consumption can extend battery lifetime of portable systems, decrease cooling costs, as well as increase system reliability.

The continuous progress in microprocessors has been maintained mostly by technology scaling, which results in exponential growth both in device density and performance. However, as the technology scaling enters nanometer regime, CMOS devices are facing many problems such as increased leakage currents, large parameter variations, low reliability and yield [1]. The inability to continue to lower the supply voltage halted the ability to increase the clock speed without increasing power dissipation. Therefore, in order to avoid encountering a stall in the future growth of computing performance, high performance microprocessors had to enter the multicore era [2]. However, the growth in the number of cores causes super-linear growth in non-core area and power; accordingly, the power dissipation problem did not disappear in the new multicore regime [3, 4]. Therefore, in addition to a focus on multicore design and parallel processing, we need research and development on much more power-efficient computing systems at various levels of abstraction.

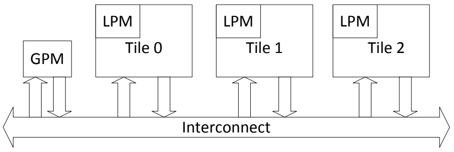

Figure 1: System model block diagram

There are various power reduction techniques published in the literature. This paper provides a survey of these techniques. Fig. 1 displays a system model that will be considered in this survey. The model consists of a number of tiles (either a processor or memory), each of which contains a local power management (LPM) unit for local power optimizations. The model also contains a global power management (GPM) unit, which aims to reduce power considering all tiles and interactions among them. The figure also shows the interconnect, which is used for the interaction among tiles and GPM. Notably, techniques used for LPM are applicable to both single and multicore systems. Based on Fig. 1, power reduction techniques can be applied to either the tiles or the interconnects, whether inside or outside the cores.

A high-level taxonomy of the power reduction techniques for both single and multicore systems is illustrated in Fig. 2. Many techniques have been proposed to achieve power reduction at different levels of abstraction, some of which require modification of the process technology, achieving power reduction during fabrication/design stage. Others are run-time techniques that require architectural support, and in some cases, technology support as well. Based on Fig. 2, there are different techniques which aim to reduce power either during fabrication/design or runtime in the tiles. Power consumption of single and multicore systems can also be reduced in the interconnects or through adaptive voltage scaling techniques in the local and global power management units to dynamically manage power during run-time. The contributions of this survey are as follows:

• We propose a comprehensive taxonomy of power reduction techniques for both tiles and the interconnect as well as run-time techniques for adaptive voltage scaling.

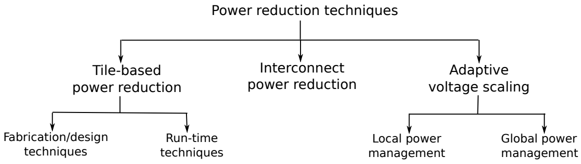
Power reduction techniques
Tile-based power reduction
Interconnect power reduction
Adaptive voltage scaling
Fabrication/design techniques
Run-time techniques
Local power management
Global power management

Figure 2: Taxonomy of total power reduction

- We discuss several techniques from each class in the taxonomy along with examples as well as reported power reduction values.

- We address various design and manufacturing issues, which degrade the effectiveness of power reduction techniques such as process and environmental variations and describe several low-power techniques considering these effects.

The rest of this paper is organized as follows. Section 2 presents low-power techniques that are applied either during fabrication/design or run-time stage to the tiles. Section 3 discusses interconnect low-power techniques that are applied dynamically during run-time. Section 4 specifically focuses on adaptive voltage scaling, which is widely used for run-time power optimization under process variations. Finally Section 5 concludes the paper.

2. TILE-BASED POWER REDUCTION

In this section we discuss the fabrication/design as well as run-time techniques for power reduction in the tiles for both single and multicore systems from architecture level to circuit level.

Power consumption of the tiles of single and multicore systems can be diminished at different levels of abstraction from system to layout, among which we will investigate various techniques at architecture, gate, and circuit levels in details. Fig. 3 illustrates a taxonomy of techniques for power reduction in the tiles from architecture to circuit level.

Based on Fig. 3, the tile power at architecture level can be cut back through low power control logic designs, low power memory hierarchies, and low power processor architectures. To explain low power control logic designs, assume the control logic of a processor as a finite state machine (FSM), which activates the appropriate circuitry for each state. Accordingly, optimizations in FSMs can be done for power reduction. Encoding FSM states to minimize the switching activity, or decomposing the FSM into sub-FSMs and activating only the circuitry needed for the currently executing sub-FSM are some examples of FSM optimizations throughout the processor [6]. A summary of attainable power reduction from this and other techniques is given in Table 1. Applying both of these techniques at the same time reduces power from 30-90%, while increasing area from 20-120%.

Another architecture level solution could be designing low power memories and memory hierarchies. Power dissipation in memories can be diminished in two ways, either by reducing the power dissipated in a memory access, or by reducing the number of memory accesses [5]. Moreover, splitting memory into smaller sub-systems is an effective way to reduce power consumed in a memory access. This can be done by partitioning memory into smaller, independently accessible components in different granularities so that only the needed circuitry is activated in each memory access [7]. A combination of subbanking, multiple line buffers and bit-line segmentation can reduce the on-chip cache power dissipation by as much as 75% in a technology-independent manner without compromising the processor cycle time. Augmenting the memory hierarchy with specialized cache structures is another popular method to save power by reducing memory hierarchy accesses. A simple example is a trace cache, which stores traces of instructions in their executed order rather than their compiled order. Hence, if an instruction sequence is already in the trace cache, it does not need to be fetched from the instruction cache and can be decoded directly from the trace cache [8]. However, conventional trace caches (CTC) may increase power in the fetch unit because of the simultaneous access to both the trace cache and the instruction cache. Dynamic direction prediction-based trace cache (DPTC), which avoids simultaneous accesses to the trace cache and the instruction cache achieve 38.5% power reduction over CTC, while only trading a 1.8% performance overhead compared to CTC [8].

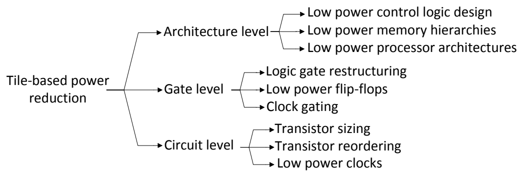

Figure 3: Taxonomy of tile-based power reduction

Another method to save power at architecture level is through adaptive processor architectures, which aim to save power by activating minimum hardware resources needed for the code that is executing. Adaptive caches and adaptive instruction queues are two examples. In an adaptive cache, storage elements (lines, blocks, or sets) can be selectively activated based on the workload. One example of such a cache is the drowsy cache whose lines can be placed in a drowsy mode where they dissipate minimal power, but retain data during drowsy mode and can be activated instantly [9]. In adaptive instruction queues, only the partitions that contain the currently executing instructions are activated at any time. For example, the heuristic proposed in [10], periodically measures the IPC (instructions per cycle) over fixed length intervals. If the IPC of the current interval is smaller than the previous interval, the size of the instruction queue is increased to enhance the throughput. The drowsy cache technique reduces power up to 53% with a performance overhead of 4.06-12.46%. Also, the adaptive instruction queue method achieves up to a 70% power reduction, while the complexity of the additional circuitry needed to achieve this result is almost negligible.

At gate level, logic gate restructuring is one simple method for power reduction. The idea is that since there are many ways to build a circuit out of logic gates, thus, how to arrange the gates and their input signals is important to power consumption [5]. Another possible solution is using low power flip-flops. Power consumption in flip-flops consists of the power dissipated in the clock signal, internal switching, and output transitions. Most of these low power designs for flip-flops reduce the switching activity or the power dissipated by the clock signal. Another method, which is very effective for power reduction at gate level is clock gating. Since clock is always active, and makes two transitions per cycle, it consumes about 40% of total processor power, so clock gating which inhibits clock to unused blocks is useful for power reduction.

Transistor sizing reduces the width of transistors based on the fact that reducing the width of transistors causes an increase in transistor delay, which leads to dynamic power reduction. Thus, the transistors that lie away from the critical paths of a circuit are usually the best candidates for this technique. Algorithms for applying this technique usually associate with each transistor a tolerable delay, which varies depending on how close the transistor is to the critical path. These algorithms then try to scale each transistor to be as

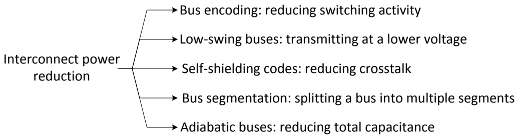
Interconnect power reduction
Bus encoding: reducing switching activity
Low-swing buses: transmitting at a lower voltage
Self-shielding codes: reducing crosstalk
Bus segmentation: splitting a bus into multiple segments
Adiabatic buses: reducing total capacitance

Figure 4: Taxonomy of Interconnect power reduction

small as possible without violating its tolerable delay [11]. Up to 15.3% power reduction can be achieved when 20% of the transistors are resized.

At circuit level, transistor reordering rearranges transistors to minimize their switching activity as their arrangement in a circuit affects power consumption [13, 14]. Another method is using low power clocks such as half-frequency and half-swing clocks, which reduce frequency and voltage respectively. Traditionally, hardware events such as register file writes occur on a rising clock edge. Half-frequency clocks synchronize events using both edges, and they tick at half the speed of regular clocks, thus cutting clock switching power in half. Reduced-swing clocks also often use a lower voltage signal, and hence reduce power quadratically [12]. As can be seen in Table 1, with transistor reordering, power can be reduced by up to 18% with minimum area and no performance overhead. The half-swing clocking scheme cuts power back by up to 67% in the whole chip and 75% in the clocking circuitry with minimal speed degradation.

3. INTERCONNECT POWER REDUCTION

Interconnects dissipate power due to switching of interconnect capacitances. Since efforts to improve chip performance lead to smaller chips with more transistors and more densely packed wires carrying larger currents [15], there arise additional sources of power consumption such as crosstalk. Therefore, power dissipating in interconnects has become one of the important contributors to total chip power consumption. Several methods have been proposed to cut back power consumption in interconnects, each of which tries to reduce power by focusing on a different aspect of power dissipation in the interconnect as depicted in Fig. 4.

A popular way to diminish interconnect power consumption is to reduce switching activity using intelligent bus encoding systems such as bus-inversion, which ensures that at most half of the bus wires switch during a bus transaction [16]. However, because of the cost of the logic required to invert the bus lines, this technique is mainly used in external buses rather than the internal chip interconnect. For every data transmission, the number of wires that switch depends on the current and previous values transmitted. If the Hamming distance between these values is more than half the number of wires, then most of the wires on the bus will switch current. To prevent this, bus-inversion transmits the inverse of the intended value and asserts a control signal alerting recipients of the inversion. For example, if the current binary value to transmit is 110 and the previous was 000, the bus instead transmits 001, the inverse of 110. This technique decreases the I/O peak power dissipation by 50% and the I/O average power dissipation by up to 25%.

Low swing buses transmit the same information but at a lower voltage [17]. Traditionally, logic one is represented by $+5$ volts and logic zero is represented by -5 volts. However, in a low-swing system, logic one and zero are encoded using lower voltages, such as $+300$mV and -300mV. The input signal is split into two signals of opposite polarity bounded by a smaller voltage range. The receiver sees the difference between the two transmitted signals as the actual signal and amplifies it back to normal voltage. This system has several advantages in addition to reduced power consumption. It is immune to crosstalk and electromagnetic radiation effects. Since the two transmitted signals are close together, any spurious activity will affect both equally without affecting the difference between them. However, the costs of increased hardware at the encoder and decoder should be considered. These buses decrease power from 62-78% with approximately 45% performance overhead.

As mentioned above, another source of power consumption in interconnects is crosstalk, which is false activity caused by activity in neighboring wires. One way of reducing crosstalk is to insert a shield wire between adjacent bus wires [18]. Since the shield remains deasserted, no adjacent wires switch in opposite directions, however, this solution doubles the number of wires. Another alternative is using coding systems which are resistant to crosstalk such as self-shielding codes [19, 20]. Just like traditional bus encoding system, a value is encoded and then transmitted. However, this system avoids opposing transitions on adjacent bus wires.

Bus segmentation is another effective technique for interconnect power reduction. In a traditional shared bus architecture, the entire bus is charged and discharged upon every access. Segmentation splits a bus into multiple segments connected by links that regulate the traffic between adjacent segments. Links connecting paths essential to a communication are activated independently, allowing most of the bus to remain powered down. Ideally, devices communicating frequently should be in the same or nearby segments to avoid powering many links. There are different algorithms for partitioning a bus into segments to benefit from this property as much as possible [21]. This technique achieves 24.6-37.21% power reduction with 6% area overhead.

Another solution to reduce power in interconnects is to reduce total capacitance, which is the principal behind adiabatic circuits [22]. These circuits reuse existing electrical charge to avoid creating new charge. In a traditional bus, when a wire becomes deasserted, its previous charge is wasted. A charge-recovery bus recycles the charge for wires about to be asserted. The saved power depends on transition patterns. No energy is saved when all lines rise. The most energy is saved when an equal number of lines rise and fall simultaneously. The biggest drawback of adiabatic circuits is the delay for transferring shared charge. This technique can achieve 28% power reduction.

4. ADAPTIVE VOLTAGE SCALING

With the on going scaling of CMOS technologies, variations in process, supply voltage, and temperature (PVT) have become serious concern in integrated circuit design. Depending on their spatial correlation, process variations can be divided into three groups. Die-to-die (D2D) variations have a correlation distance larger than the die size, i.e., all transistors on a chip are affected the same way. Within-die (WID) variations have a correlation distance smaller than the chip size. Random variations are not correlated at all; every transistor is affected individually. Environmental variations such as power supply noise and crosstalk have also gained significance with increasing current densities and reduced geometric dimensions [32].

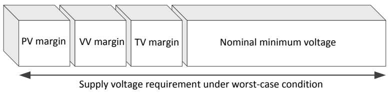

Figure 5: Schematic of the worst-case guardbanding approach (PV, VV, and TV stand for process, voltage, and temperature variations, respectively)

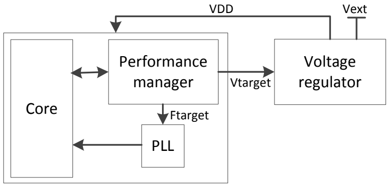

Figure 6: Architecture of an AVS system

Therefore, an individual safety margin for each variation source is added on the top of supply voltage needed for the nominal case as depicted in Fig. 5. However, this classical worst-case analysis is quite pessimistic and leads to power and performance be wasted. To overcome this problem, various adaptive design strategies have been proposed. The basic idea is to adapt the supply voltage to the optimal value, based on the current operation conditions of the system so that power is saved; variations are compensated, while maintaining the desired performance.

In this section, LPM techniques which are used in both single and multicore systems are explored. Specifically we focus on adaptive voltage scaling, which is widely used for run-time power optimization under process variations. In addition, we discuss GPM techniques which are specialized for multicore systems.

4.1 Local power management unit

Adaptive voltage scaling (AVS) systems are very efficient in saving power since the supply voltage has a profound impact on the operating frequency and power consumption of an integrated circuit. Typically, logic delay increases as V_{DD} reduces and power consumption increases super linearly with V_{DD}. Whenever maximum performance is not required, supply voltage can be scaled so that power can be saved while the system can still meet the timing constraints. Fig. 6 shows the overall architecture of an AVS system [28]. The performance manager predicts performance requirements. Once performance requirement is determined, the performance manager sets the voltage and frequency just enough to accomplish the performance target of the system. The target frequency is sent to the phase-locked loop (PLL) to accomplish frequency scaling. Based on the target voltage, the voltage regulator is programmed to scale the supply voltage up/down until target voltage is achieved.

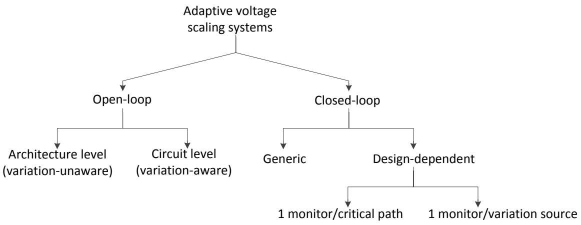
Adaptive voltage scaling systems
Open-loop
Closed-loop
Architecture level (variation-unaware)
Circuit level (variation-aware)
Generic
Design-dependent
1 monitor/critical path
1 monitor/variation source

Figure 7: Taxonomy of adaptive voltage scaling systems

Thus, accurate circuit performance estimation is required so that the actual performance of the core running under the scaled voltage is monitored to guarantee a fail-safe operation, while maintaining the required performance [28]. A taxonomy of AVS systems is illustrated in Fig. 7. Based on whether the performance estimation is done early in manufacturing or during run-time, these techniques can be classified as either open or closed-loop [25]. The following subsections explore the commonly used AVS techniques.

4.1.1 Open-loop adaptive voltage scaling

A typical open-loop adaptive voltage scaling system creates a pre-characterized LUT to find the corresponding minimum voltage for a given frequency target. Conventionally, the voltage levels for each domain, as well as the mapping between frequencies and voltages are determined at architecture level without considering variations. One example is the three domain dynamic voltage frequency scaling (DVFS) power management scheme proposed in [26]. In this architecture level technique, the voltage and frequency of each power domain are dynamically scaled according to the performance requirement of each domain. They assumed that each domain has a specific requirement of voltage and frequency due to different workloads that they execute. Using three power domains diminishes power by up to 65% compared to a single domain, while imposes 2.6% area and 9.5% power overhead on the system.

However, with the increasing effect of process variations as a result of technology scaling, the research has become more focused towards the variation-aware adaptive voltage scaling techniques at circuit level. A technique proposed in [27], utilizes a user and process driven dynamic voltage and frequency scaling scheme to adapt voltage to the frequency of a microprocessor in real-time according to processor needs. User-driven frequency scaling (UDFS) uses direct user feedback to determine the processor frequency for individual users. Process-driven voltage scaling (PDVS) creates an LUT which maps frequency and temperature to the operating minimum voltage considering process variations. Using both of these techniques at the same time reduces power by up to 50% for single task and 70% for multi-task workloads compared to Windows XP DVFS. However, since these techniques do not have a feedback mechanism, the LUT is heavily guard-banded to ensure reliable system operation which results in performance and energy wastes. At the same time, characterizing the LUT is a time consuming and expensive procedure. Thus, closed-loop schemes which take advantage of feedback mechanisms during run-time are more efficient in saving power.

4.1.2 Closed-loop adaptive voltage scaling

A closed-loop adaptive voltage scaling system adjusts supply voltage by probing actual chip performance using on-chip monitors, thus, margin required by open-loop systems can be recovered. To track timing performance of a chip, many approaches have been proposed. Based on Fig. 7, in terms of design point of view, performance monitors are classified into design dependent and generic[24].

Generic performance monitors

Generic performance monitors range from simple inverter-based ring oscillators [29] to more complex process-specific ring oscillators (RO) [30] and also alternative monitoring

Table 1: Reported power reduction values

Reference	Section	Technique	Power reduction	Comments
[6]	II.A	Encoding FSM & decomposition to sub-FSMs	30% to 90%	20% to 120% area overhead
[7]	II.A	Splitting memory into smaller sub-systems	75%	sub-banking, bit-line segmentation, multiple-line buffers - no performance overhead
[8]	II.A	DPTC	38.5%	1.8% performance overhead over CTC
[9]	II.A	Drowsy cache	53%	4.06% to 12.46% performance overhead
[10]	II.A	Adaptive instruction queue	70%	Complexity of additional circuitry is negligible
[5]	II.A	Clock gating	up to 40%	small area overhead
[11]	II.A	Transistor sizing	up to 15.3%	20% of transistors are resized
[14]	II.A	Transistor reordering	18%	minimum area overhead, no performance overhead
[12]	II.A	Half-swing clock	67%-75%	small speed degradation
[16]	II.B	Bus inversion	50%-25%	peak and average power reduction of I/O
[17]	II.B	Low swing bus	62% to 78%	45% performance overhead
[21]	II.B	Bus segmentation	24.6% to 37.21%	6% area overhead
[22]	II.B	Adiabatic bus	28%	-
[26]	III.A	Three domain DVFS	65%	power overhead: 9.5%, area overhead: 2.6% compared to single domain
[27]	III.A	UDFS-PDVS	50% to 75%	compared to Windows XP DVFS
[33]	III.A	Universal delay line	13% to 27%	area overhead: 0.01%, power overhead is negligible
[23]	III.A	In-situ delay monitoring (over-critical)	13.5%	compared to the worst-case design, prediction error rate: 1.10^{-15}
[32]	III.A	In-situ delay monitoring (regular)	14%	power overhead: 0.5%, area overhead: 10%
[34]	III.A	Critical path replica	11% to 78%	highly dependent on the benchmark
[36]	III.A	RCP	31%	smaller guard-band than critical path replica, prediction error rate: 2.8%

structure such as PLLs [31]. Although generic monitors are very simple to design and can be used in any product without customizations, they are inadequate to capture design characteristics, and there will be a large error in the measurements due to the difference in gate structure between the actual critical path and the delay monitor. So, delay estimation using generic monitors is less accurate and sometimes incurs larger margins. However, the generic performance monitor proposed in [33] tries to minimize the errors due to gate structure difference by utilizing certain chain of delay gates, as well as the errors due to the within die variations by distributing monitors among the chip. Each performance monitor, which is called a universal delay line, contains a ring oscillator and a counter. The ring oscillator is designed with double stacked NMOSs and PMOSs since this gate structure is the most dominant component in the critical path delay, which minimizes the error due to the gate structure difference. This technique decreases power by up to 27% with a negligible area overhead.

Design-dependent performance monitors

According to Fig. 7, some of the design-dependent techniques implement one monitor per variation source, while the others implement one monitor per critical path. One group of methods that utilize one monitor per critical path are based on in-situ delay monitors, which are special latches or flip-flops, included at the end of critical paths to report the timing behavior of the circuit in order to form a closed loop configuration for voltage adaptation [32]. Circuit delay characterization using in-situ delay monitors can be done in two different ways. The first is by observing the regular operation of a circuit and to detect timing errors in the circuit itself during operation. With the error information, the critical supply voltage, that is the minimum supply voltage which is needed for correct operation, can be determined. The second possibility is to observe an over-critical system. Here, a test module which is always slower than the most critical part of the chip is observed, and as soon as the test module fails, the system detects a late data transition called a pre-error[23]. The regular in-situ method achieves 14% power reduction using two power switches, while imposing 10% area overhead and 0.5% power overhead to the system. The over-critical method compared to the worst-case design reduces power to 13.5% with a negligible error rate of 10^{-15}.

Another approach implements replica-paths, representing the critical paths of the circuit, thus, with varying operating conditions, the timing of the replica-path will change similarly to the actual critical path. So, the timing information of the replica-path can be used to control the supply voltage adaptively. Alternatively, the critical path replica can be replaced by fan-out of 4 (FO4) ring oscillator [34] or a delay line [35]. A safety margin is added to account for any mismatch between the ring oscillator (or the delay line) and the actual critical path. The FO4 technique achieves 11% power reduction for compute-intensive code; the power decreased by up to 78 % for non-speed-critical applications compared to operating at a fixed supply voltage.

Several methods have been proposed which implement one monitor per variation source. For instance, the method presented in [36] synthesizes a single representative critical path (RCP) for post-silicon delay prediction. The RCP is designed such that it is highly correlated to all critical paths for some expected process variations. For both this and the critical path replica method, it is essential to guard-band the prediction. However, the RCP approach with 2.8% predication error rate requires a guard-band 31% smaller than the critical path replica method.

Although design-dependent monitors show good estimation accuracies, most of them rely on monitoring and characterization of one unique critical path, however, due to the increasing effect of process variations, finding one unique critical path is a hard task to do. Depending on the operating point, process corner, and workload many different timing paths might become critical, therefore, for real circuits the concept of finding one critical path and create a critical path replica as a performance monitor is too simplistic. Moreover, techniques that have one monitor per critical path incur high area overhead as well as long design turnaround time to the system[24].

4.2 Global power management unit

We discussed various types of performance monitors used for AVS to locally manage power within each core. All the mentioned types of performance monitors are applicable for both single as well as multicore processors. As we discussed earlier in this section, process variations are static during operation and manifest themselves as D2D, WID variations, while temperature and voltage variations are dynamic. These affect both single as well as multicore processors. However, as the individual core size becomes smaller, there arises another source of process variations that specifically affects multicore systems, called core-to-core variations (C2C). C2C variations occur due to spatially correlated WID variations, for example due to non-uniformity in the lithographic exposure field [38]. Thus, multicore processors are still threatened by increasing power consumption due to PVT variations since they require large design margins in the supply voltage resulting in large power consumption.

Dynamic power management of multicore processors is extremely important because it allows power savings when not all cores are used. AVS is one of the techniques that is widely used for power reduction in multicore processors. The per-core performance data collected by performance monitors is sent to the global power management unit to decide the supply voltage. AVS for multicore processors can be performed at various levels of granularity: 1) Per-chip, the supply voltage is set globally for the whole chip, 2) Per-core, the supply voltage is set for each core, which means that only cores that require higher frequency are set to the higher supply voltage, while other cores operate at lower supply voltage or are completely shut down, 3) cluster-level, the voltage is set for each cluster which one or more cores are associated with.

5. CONCLUSION

This paper presented a classification of power reduction techniques in single and multicore systems. Three main classes have been discussed: the techniques which aim to reduce power either during fabrication/design or runtime in the tiles, run-time power reduction techniques for interconnects, and adaptive voltage scaling techniques to dynamically manage power during run-time. In addition, a number of design and manufacturing issues (such as process and temperature variations) have been taken into consideration. The paper also discussed a number of examples for each of the classes and presented the published power reduction numbers reported by their respective papers. A summary of these numbers has been listed along with the trade-offs in performance and/or area overhead incurred as a result.

Acknowledgments

This work is carried out under the BENEFIC project (CA505), a project labelled within the framework of CATRENE, the EUREKA cluster for Application and Technology Research in Europe on NanoElectronics.

6. REFERENCES

[1] Y. B. Kim, *Challenges for Nanoscale MOSFETs and Emerging Nanoelectronics*, Trans. On Electrical and Electronic Materials, vol. 11, pp. 93-105,2010.

[2] S. H. Fuller and L. I. Millett, *The Future of Computing Performance: Game Over or Next Level?*, The National Academy of Sciences, 2011.

[3] L. Spracklen and S. G. Abraham, *Chip Multithreading: Opportunities and Challenges*, in HPCA, pp. 248-252, 2005.

[4] H. Esmaeilzadeh, et. al, *Dark Silicon and the End of Multicore Scaling*, in ISCA, vol. 46, pp. 5âĂŞ26, 2011.

[5] V. Venkatachalam and M. Franz, *Power Reduction Techniques for Microprocessor Systems*, ACM Computing Surveys, vol. 37, no. 3, pp. 195-237, 2005.

[6] F. Gao and J. P. Hayes, *ILP-based optimization of sequential circuits for low power*, in ISLPED, pp. 140-145, 2003.

[7] K. Ghose, B. Kamble, *Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation*, in ISLPED, Pages 70-75, 1999.

[8] J. Hu, et. al, *Using Dynamic Branch Behavior for Power-efficient Instruction Fetch*, in ISVLSI, pp. 127-132, 2003.

[9] S. N. Kim, et. al, *Drowsy Instruction Caches: Leakage Power Reduction Using Dynamic Voltage Scaling and Cache Sub-bank Prediction*, in MICRO, pp. 219-230, 2002.

[10] A. Buyuktosunoglu, et. al, *An Adaptive Issue Queue for Reduced Power at High Performance*, Proc. of the Inter. Workshop on Power-aware Computer Systems, pp. 25-39, 2001.

[11] J. Ebergen, J. Gainsley, and P. Cunningham, *Transistor sizing: How to control the speed and energy consumption of a circuit*, Proc. of the Inter. Symp. on Asynchronous Circuits and Systems, pp. 51-61, 2004.

[12] H. Kojima, S. Tanaka, and K. Sasaki, *Half-Swing Clocking Scheme for 75% Power Saving in Clocking Circuitry*, in JSSC, vol. 30, no. 4, pp. 432-435, 1995.

[13] E. Kursun, S. Ghiasi, and M. Sarrafzadeh, *Transistor Level Budgeting for Power Optimization*, in ISQED, pp. 116-121, 2004.

[14] A. Sultania, D. Sylvester, and S. Sapatnekar, *Transistor and Pin Reordering for Gate Oxide Leakage Reduction in Dual Tox circuits*, in ICCD, pp. 228-233, 2004.

[15] K. Banerjee and A. Mehrotra, *Global interconnect warming*, IEEE Circuits and Devices Magazine, pp. 16-32, 2001.

[16] M. Stan and W. Burleson, *Bus-invert coding for low-power i/o*, in TVLSI, pp. 49-58, 1995.

[17] H. Zhang, J. Rabaey, *Low-swing interconnect interface circuits*, in ISLPED, pp. 161-166, 1998.

[18] C. N. Taylor, S. Dey, and Y. Zhao, *Modeling and Minimization of Interconnect Energy Dissipation in Nanometer Technologies*, in DAC, pp. 754-757, 2001.

[19] B. Victor and K. Keutzer, *Bus encoding to prevent crosstalk delay*, in ICCAD, pp. 57-63, 2001.

[20] K. N. Patel and I. L. Markov, *Error correction and crosstalk avoidance in dsm busses*, Proc. of ACM Inter. Workshop on System-Level Interconnect Prediction, pp. 9-14, 2003.

[21] W. B. Jone, et. al, *Design theory and implementation for low-power segmented bus systems*, in TODAES, vol. 8, issue 1, pp. 38-54, 2003.

[22] B. Biship, M. .J .Irwin, *Databus Charge Recovery: Practical Considerations*, In ISLPED, pp. 85-87, 1999.

[23] M. Wirnshofer, et. al, *A Variation-Aware Adaptive Voltage Scaling Technique based on In-Situ Delay Monitoring*, in DDECS, pp. 261-266, 2011.

[24] T. Chan, et. al, *DDRO: A Novel Performance Monitoring Methodology Based on Design-Dependent Ring Oscillators*, in ISQED, pp. 633-640, 2012.

[25] T. Chan and A. B. Kahng, *Tunable Sensors for Process-Aware Voltage Scaling*, in ICCAD, pp. 7-14, 2012.

[26] J. Lee, B. G. Nam, and H. J. Yoo, *Dynamic Voltage and Frequency Scaling (DVFS) Scheme for Multi-Domains Power Management*, in ASSCC, pp. 360-363, 2007.

[27] B. Lin, et. al, *User and Process-Driven Dynamic Voltage and Frequency Scaling*, in ISPASS, pp. 11-22, 2009.

[28] M. Elgebaly and M. Sachdev, *Variation-Aware Adaptive Voltage Scaling System*, in TVLSI, vol. 15, no. 5, pp. 560-571, 2007.

[29] T. Yamagishi, et. al, *An Area-Efficient, Standard-Cell Based On-Chip NMOS and PMOS Performance Monitor for Process Variability Compensation*, Proc. of IEEE Inter. Conf. on Cool Chips, pp. 1-3, 2012.

[30] M. Bhushan, et. al, *Ring Oscillators for CMOS Process Tuning and Variability Control*, in TSM, Vol. 19, No. 1, pp. 10-18, 2006.

[31] K. Kang, et. al, *On-Chip Variability Sensor Using Phase-Locked Loop for Detecting and Correcting Parametric Timing Failures*, in TVLSI, vol. 18, no. 2, pp. 270-280, 2010.

[32] M. Eireiner, et. al, *In-Situ Delay Characterization and Local Supply Voltage Adjustment for Compensation of Local Parametric Variations*, in JSSC, vol. 42, no. 7, pp. 1583-1592, 2007.

[33] Y. Ikenaga, et. al, *A 27% Active-Power-Reduced 40-nm CMOS Multimedia SoC with Adaptive Voltage Scaling Using Distributed Universal Delay Lines*, in JSSC, vol. 47, no. 4, pp. 832-840, 2012.

[34] TD. Burd, T. Pering, A. Stratakos, and R. Brodersen, *A dynamic voltage scaled microprocessor system*, in ISSCC, pp. 294 âĂŞ 295, 2000.

[35] J. Kim and M. A. Horowitz, *An efficient digital sliding controller for adaptive power-supply regulation*, in IJSSC, vol. 37, no. 5, pp. 639-647, 2002.

[36] Q. Liu and S. S. Sapatnekar, *Capturing Post-Silicon Variations Using a Representative Critical Path*, in TCAD, vol. 29, no. 2, pp. 211-222, 2010.

[37] L. Xie and A. Davoodi, *Representative Path Selection for Post-Silicon Timing Prediction under Variability*, in DAC, pp. 386-391, 2010.

[38] E. Humenay, D. Tarjan, and K. Skadron, *Impact of Process Variations on Multicore Performance Symmetry*, in DATE, pp. 1-6, 2007.