SAVS: A Self-Adaptive Variable Supply-Voltage Technique for Process- Tolerant and Power-Efficient Multi-issue Superscalar Processor Design

Hai Li
Qualcomm Inc.
5775 Morehouse Dr.
San Diego, CA, USA
Tel : +1-858-845-7393
e-mail: hail@qualcomm.com

Yiran Chen
Synopsys Inc.
700 East Middlefield Road
Mountain View, CA, USA
Tel : +1-650-584-4885
e-mail: yiran.chen@synopsys.com

Kaushik Roy
Purdue University
ECE Department
West Lafayette, IN, USA
Tel : +1-765-494-2361
e-mail: kaushik@ecn.purdue.edu

Cheng-Kok Koh
Purdue University
ECE Department
West Lafayette, IN, USA
Tel : +1-765-496-3683
e-mail: chengkok@ecn.purdue.edu

Abstract - ***Technology scaling and sub-wavelength optical lithography is associated with significant process variations. We propose a self-adaptive variable supply-voltage scaling (SAVS) technique for multi-issue out-of-order pipeline to improve parametric yield with minimal power dissipation. Our error-correction circuitry and recovery mechanism allow the proposed fault-tolerant pipeline to work at a dynamically tuned supply voltage with a very low error rate. Experiments on an 8-issue, out-of-order superscalar processor show that SAVS can achieve 93.3% yield with 8.66% total power reduction under a scaled V_{DD}, compared to the same yield achieved by conventional microarchitecture. The increased execution time is negligible (0.014%).***

1. Introduction

With technology scaling, power dissipation has become a limiting factor in high-performance microprocessor design. Among existing power management techniques, supply voltage (V_{DD}) scaling has been proven to be effective for both dynamic and leakage power reduction: with V_{DD} scaling, dynamic power decreases quadratically [1] and leakage power decreases exponentially [2], associated with the increase of circuit delay.

The latency increases of different circuit styles due to V_{DD} scaling are different [3]. Fig. 1 shows the simulation results of the relative latency increases of a gate-dominant circuit and an interconnect-dominant circuit when V_{DD} is scaled, under BPTM 70nm technology. Compared to interconnect-dominant circuit (e.g., result bus), the latency of gate-dominant circuit (or logic circuit, e.g., ALU), is less sensitive to V_{DD} scaling. Obviously, the circuit whose latency increases slower with V_{DD} scaling may get a larger benefit from V_{DD} scaling.

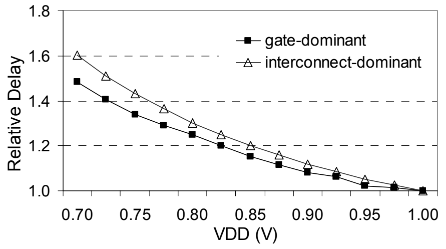


Fig. 1. V_{DD} scaling and relative delay

To ensure correct timing and consequently, acceptable chip yield at scaled V_{DD}, most of V_{DD} scaling techniques have to increase the clock period. The prolonged clock period leads to a longer execution time and consequently, the degradation of system performance. Hence, V_{DD} scaling is usually adopted in embedded systems that have relatively less stringent performance requirements [3][4]. For performance-oriented systems, V_{DD} scaling is applied only when the system has a low throughput. For example, processor can work at a scaled V_{DD} with a low clock frequency when pipeline idles during L2 cache misses [1]. The conflict between V_{DD} scaling and circuit delay (and consequently, the chip yield), severely limits the application of V_{DD} scaling technique in high-performance microprocessors.

In reality, due to the variability of device parameters (e.g. random doping, transistor dimension and threshold voltage variation) and the fluctuation of environment factors (e.g. temperature shifting and power supply voltage noise), a margin has to be added to the supply voltage adopted in the design time. This ensures a certain chip yield, which is defined as the ratio of the number of chips that work properly over the total number of chips, under a certain V_{DD}. However, the traditional corner-based V_{DD} selection, which assumes that all worst-case conditions occur simultaneously, may heavily overestimate the actual needed V_{DD}.

Several studies show that for logic circuits, the worst-case V_{DD} requirement may seldom occur. For example, [5] shows that for random input vectors, the average carry propagation length of a carry look-ahead adder (CLA) is much less than 1/3 of the longest one. Therefore, power management techniques that scale V_{DD} *below* the worst-case V_{DD} requirement, have been recently investigated: In [6] and [7], ALU works under a scaled V_{DD} and timing-errors due to incomplete operation of ALU are detected and corrected by/from a result checker or a shadow latch.

We note that timing-error correction techniques can be also adopted to tolerate circuit delay variation due to the process parameter fluctuations and environment factor variations. By detecting and correcting the incomplete operations of circuit at the scaled V_{DD}, timing-error correction mechanism can improve chip yield as well as reduce the power dissipation. This is especially important to the application of V_{DD} scaling in high-performance processors and is the motivation of our work.

In this paper, we propose a self-adaptive variable supply-voltage scaling (SAVS) technique that targets the process-tolerance and the power-efficiency in multi-issue high-performance microprocessors. Timing-error correction mechanism is applied to selected pipeline stages for chip yield enhancement by correcting the errant timing due to the delay variation. The selected stages can work at a scaled V_{DD} with a tolerable timing error rate while the chip yield is still maintained.

Simulations on 23 SPEC2000 benchmarks show that on average, SAVS can reduce up to 8.66% of microprocessor power with negligible instruction per cycle-based (IPC-based) performance degradation (0.014%) while maintaining a required chip yield of 93.3% and same clock frequency.

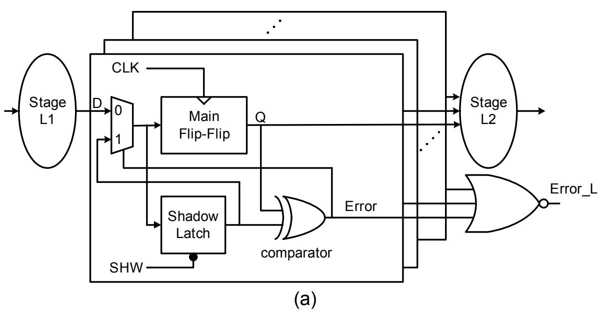
2. Self-Adaptive Variable Supply-Voltage Scaling (SAVS)

2.1. SAVS Mechanism

Shadow latch has been proven to be effective and economic for data retention and checking. For example, in [8], data is stored in shadow latch when circuit switches to power-saving mode. When circuit switches back to active mode, system status is restored from the data stored in shadow latch. In [7], a shadow flip-flop-based technique (called Razor latch) is proposed for timing-error detection and recovery. The mechanism of shadow flip-flop can be summarized as follows:

At the end of each clock cycle, the output of pipeline stage L1 is latched by the main flip-flop (FF) (Fig. 2(a)). When an errant output occurs, i.e., when operation latency L_{op} exceeds the original clock period T_{clk}, the incomplete output is latched by main FF at the end of clock cycle i+1. After time L_{op}-T_{clk}, operation in L1 completes and the output of L1 switches to the correct data. Time ΔT after the end of cycle i+1, detection signal SHW triggers shadow latch to capture the correct data. If the data captured by shadow latch is different from the data stored in main FF, an 'ERROR' signal is

The work is sponsored in part by Marco Gigascale Systems Research Center (GSRC) and Semiconductor Research Corp.



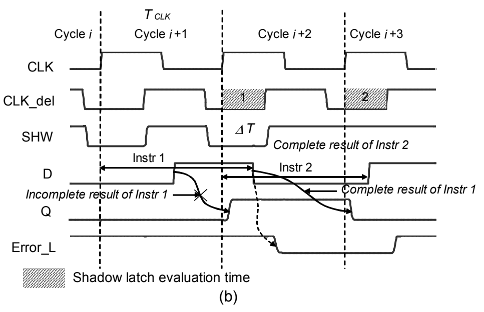
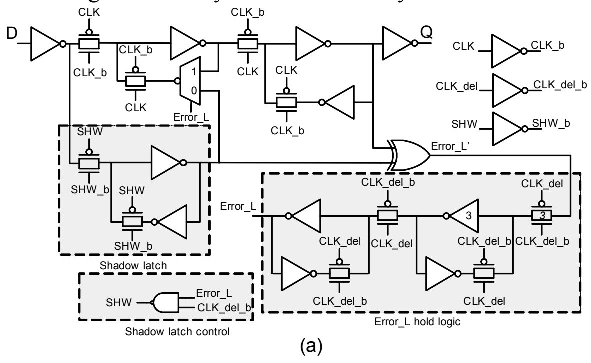

T CLK
Cycle i
Cycle i+1
Cycle i+2
Cycle i+3
CLK
CLK_del
SHW
D
Q
Error_L
ΔT
Instr 1
Instr 2
Complete result of Instr 2
Incomplete result of Instr 1
Complete result of Instr 1
Shadow latch evaluation time
(b)

Fig. 2. Shadow flip-flop mechanism for timing failure correction (a) Schematic (b) Timing diagram (From HSPICE)

generated in the subsequent cycle i+2 and the correct data is restored to main FF. Obviously, any operation with the latency longer than $T_{\mathrm{clk}}+\Delta T$ cannot be captured by shadow latch. In such a case, system may not recover from the timing-error. The corresponding timing diagram, which is extracted from HSPICE simulation, is shown in Fig. 2(b). After a timing-error has occurred at the end of cycle i+1, the errant output data of stage L1 is sent to the subsequent stage L2. Hence, the instruction executed in the subsequent stage L2 in cycle i+2 must be re-executed after getting the correct input from L1 at the beginning of cycle i+3. One cycle penalty is introduced in the procedure above since the execution of Instr 1 in stage L1 actually takes two clock cycles.



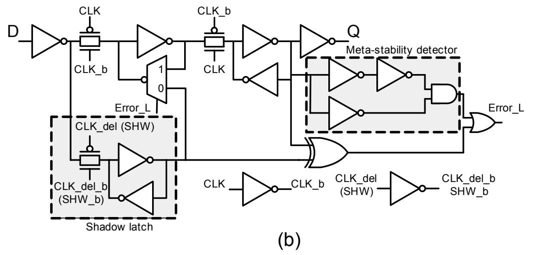


Fig. 3. Shadow flip-flop Design (a) Proposed Design (b) Original Design

In our SAVS technique, a modified robust shadow FF is developed for high-performance application and shown in Fig. 3(a). Compared to the shadow flip-flop design in [7] (shown in Fig. 3(b)), our shadow FF design has the following advantages:

1. Additional control transmission gate in the inverter loop of each latch and careful sizing up of inverters and transmission gates provide [9] more robust design to prevent the “drive fight” of two inverters and consequently, occurrence of meta-stability.

2. In the Error_L hold logic in Fig. 3(a), signal Error_L is triggered by a delayed clock signal CLK_del, based on the result of comparison of the values stored in the main slave latch and the shadow latch. This design avoids the false switching of Error_L due to any glitch at the circuit output D in shadow latch evaluation time 1 (see Fig. 2(b)).

3. When an errant output is detected, the FF structure of Error_L control logic keeps Error_L at logic ZERO until the next shadow latch evaluation time 2 (see Fig. 2(b)) completes. This mechanism masks the evaluation signal SHW of shadow latch in shadow latch evaluation time 2 and prevents the complete result of Instr 1 at Q from being corrupted by the result of Instr 2 (if Instr 2 is a long-latency instruction and switches in the shadow latch evaluation time 2).

We note that only *one* such a flip-flop-based Error_L control logic is required by the whole pipeline stage L1 in Fig. 2(a): The Error_L signal in Fig. 3(a) is the Error_L signal in Fig. 2(a), which indicates any Error in any output bits of stage L1. The incurred area/power overhead is negligible.

For the non-critical pipeline stages whose latency is always short enough (less than one clock cycle at scaled V_{DD}), no error correction circuitry is required. Moreover, the output bits that are not located in the critical data paths also do not require error correction circuit. Here the critical data paths are defined as the data paths that may not complete execution within one clock cycle at the scaled V_{DD}, due to process variations or other environmental factors.

As pointed out in [7], in Fig. 2(b), if the execution time (T_S) of Instr 2 in the cycle i+2 is shorter than ΔT, the complete output of Instr 1 may be corrupted by the result of Instr 2. Hence, buffers need to be added at some inputs of stage L1 to ensure $T_S > \Delta T$ for the output bits with shadow latch. Such input buffers do not increase the critical path of stage L1.

The simulation of a 32-bit CLA under BPTM 70nm process [10] considering Vt variations shows that that only 7 output bits may generate errors when scaling the V_{DD} from 1.0V to 0.725V for a chip yield of 93.3%. More details of experiment setup are given in Section 4.1. Because the scaled V_{DD} applied to the stages with SAVS technique (SAVS stages) is different from the normal V_{DD} of other stages, FFs with level-conversion function (FFLC) [11] may be required at the output of SAVS stages. Our simulation shows the modification of the execution/bypass stage in SAVS technique results in about 6% area overhead, with respect to the conventional execution/bypass stage design in an 8-issue out-of-order superscalar microarchitecture [12][13].

2.2. Principle of SAVS

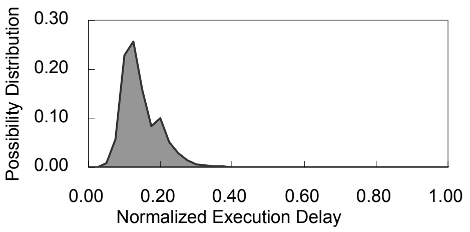


Fig. 4. Delay distribution of 32-bit CLA

Fig. 4 shows the delay distribution of the 32-bit CLA over 8192 random inputs. We can observe that at most of time, the delay of CLA is far shorter than the longest possible delay (normalized to 1 in Fig. 4). *Very few operations really go through the longest data path of CLA*. If we lower the V_{DD} and are able to correct all possible timing errors due to process variations or environmental factor fluctuations, power dissipation can be lowered without any degradation of chip yield while maintaining the same working frequency.

For some interconnect-dominant circuits, the delay also varies from case to case. Fig. 5 shows the layout of an 8-way bypass mechanism in execution/bypass stage of pipeline. Result bus bypasses the data between 8 different ALUs. To reduce the bypass delay, Register File (REG) is located at one end [12][13].

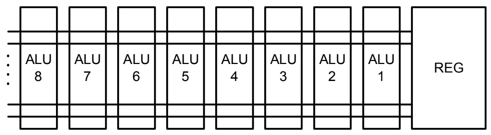


Fig. 5. Layouts for 8-way bypassing

Bypass delay includes bypass logic delay and interconnect delay. The bypass delays between different ALU pairs in Fig. 5 are shown in Fig. 6 (delays are normalized over the longest bypass delay occurring between ALU8 and ALU1). We use the dimensions of ALUs given in [12] and carefully scale them to 70nm technology. Repeaters are inserted in some result bypass buses to reduce the long RC delay of long metal interconnect.

It is known that for superscalar pipeline, the usage of ALUs is limited by the ILP (instruction level parallelism). Most of time, bypassing is constrained among first several ALUs that are in physical proximity in layout. The corresponding bypass delays are much shorter than the longest one (from ALU8 to ALU1).

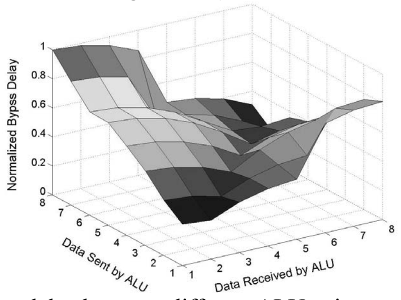


Fig. 6. Bypass delay between different ALU pairs

We note that the latency of circuit does *not* rely on the input vector when: 1) in gate-dominant circuit, every operation has to go through the longest data path; 2) in interconnect-dominant circuit, the data transmission is point-to-point, i.e., transmitter- receiver pair. In such cases, V_{DD} scaling may result in a high error rate with a fixed clock frequency. SAVS may not be applicable.

2.3. Overview of SAVS in High-Performance Microprocessors

Fig. 7 depicts the general pipeline model for a superscalar processor [13]. The delays of each stage in the 8-issue baseline superscalar pipeline are carefully analyzed in [7], [13] and [14]. The delays of every stage in terms of FO4 delay, which measures the delay of an inverter driving a-fanout-of-four, are shown in Table I for 70nm technology. It has been shown that in 0.18µm technology and beyond, for the superscalar microprocessor whose issue width is equal to or more than 6, execution/bypass stage becomes the critical stage (e.g., execution/bypass stage with a delay of 18+18=36 FO4 in Table I in an 8-issue pipeline) [13][14]. The corresponding simulation parameters of baseline pipeline will be shown in Section 4.3.1.

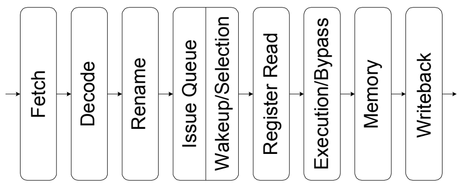


Fig. 7. Baseline superscalar model

Instruction cache (I-cache), data cache (D-cache) and register files involve RAM (random access memory) structure [14]. As the analysis in Section 2.2, due to different interconnect lengths from the active cell to sense amplifier, cache core access delay (along wordline and bitline) relies on the physical position of data cell in RAM array. However, the cache core access time usually occupies limited percentage (less than 35%) of total cache access time [15]. The delays of other peripheral circuitries, including decoder, sense amplifier and data bus, are insensitive to the data address in RAM. Therefore, SAVS may *not* be applicable to fetch, register read, D-cache and writeback stages since the delays of those RAM-based stages (30 FO4 for Fetch (I-Cache) and D-Cache, 24 FO4 for register read and writeback) are close to the clock period (36 FO4).

The register rename logic is used to translate logical register designators into physical register designators. Its two functions – mapping and check-pointing – are also usually implemented by RAM structure [13][14]. However, the delay of rename stage (18 FO4) is only around half of the clock period (36 FO4). Therefore, SAVS can be applied to rename stage with error-free operations (at scaled V_{DD} of SAVS: 0.725V in our simulation, Section 4.1).

Table I
Stage Delays of 8-issue Superscalar Pipeline

	Delay (FO4)		Delay (FO4)
Fetch	30	Decode	18
Rename	18	Register Read	24
Issue Queue	12	Execution	18
Selection	12	Bypass	18
Load/Store. Queue	12	Writeback	24
I-Cache	30	D-Cache	30

The complexity and the latency of decode stage greatly depends on ISA (instruction set architecture) and circuit design. In many practical microprocessor design, decode stage latency is shorter than the execution time of ALU [7][16]. In the execution/bypass stage of multi-issue out-of-order pipelines, bypass logic delay is equal to or greater than ALU delay [12][13][14]. In such a case, the decode delay is around half of clock cycle. Consequently, SAVS *may* be applied to decode stage with error free operations. However, to be conservative, we did *not* apply SAVS to decode stage in our design.

The issue queue is a CAM (content addressable memory) structure [13][14]. In every entry of issue queue, the content is the decoded instruction and the tag is the sources of instruction's operands. Wakeup logic works as follows: The tags associated with the executed results of ALU are broadcasted to all entries in issue queue. If each tag in an entry matches one tag associated with the execution results, the corresponding instruction is ready to be issued in the next cycle. Wakeup logic latency is mostly determined by the length of issue queue and the physical position of stored instruction in issue queue.

Selection logic is composed of stacked arbiters to select instructions to be issued from the pool of ready instructions in issue queue. The number of instructions to be issued in the next cycle determines the latency of selection logic: more arbiters are required when more instructions are to be issued [17]. Although the latency of wakeup/selection stage is input vector (address) dependent, the extremely high-cost recovery mechanism makes the application of SAVS difficult: after an instruction is issued, it is popped out from the issue queue right away. Recovery from such errors may need to restore the issued instructions back to issue queue, and, introduce large power/area overhead and design complexity. To be conservative, we did *not* apply SAVS to wakeup/selection stage.

The load/store queue in memory stage is also a CAM structure. The delay of writing into or reading from memory queue (12 FO4) is around 1/3 of the delay of critical stage (36 FO4). Hence, SAVS can be applied to memory stage (not cache).

Because of the increasing gap between the delays of execution/bypass stage and other stages with technology scaling, the yield of pipeline is mainly determined by the timing error in execution/bypass stage. In execution/bypass stage, each of the ALU execution and data bypass takes about half of clock period [12][13][14]. Correcting the timing error in execution/bypass stage at the scaled V_{DD} can improve the yield of execution/bypass stage and consequently, the yield of whole pipeline, with minimal power dissipation. Carefully choosing the range of V_{DD} scaling ensures that the error rate of execution/bypass stage and the incurred performance penalty are low while other pipeline stages still work with error free.

We note that the discussion above may not be applicable to some deeper pipelines since the design objective of deep pipeline is to uniform the delay of each stage of pipeline [14]. However, deep pipeline has been proven unsuitable to scaled technology because of the extremely high power dissipation and a large number of critical paths [18].

In summary, for a multi-issue out-of-order pipeline, SAVS can be easily applied to rename, execution/bypass stages and load/store queue. V_{DD} scaling causes the timing-errors only in execution/bypass stage. Hence a timing-error correction circuitry is required in execution/bypass stage. Although other stages keep working at the conservatively high V_{DD} (it is important if cache is another critical stage), our experimental results in Section 4 show a significant power reduction with negligible performance penalty and energy overhead, when SAVS is applied.

3. Implementation of SAVS

3.1. Pipeline recovery mechanism

In Fig. 2(b), the errant outputs of execution/bypass stage at the end of cycle i+1 may be bypassed back to the inputs of the execution/bypass stage and also sent to the memory stage in the subsequent cycle i+2. In such a case, the operations committed in execution/bypass stage and memory stage in the current cycle i+2 must be re-executed (or re-sent) in the following cycle i+3. The error recovery mechanism of the pipeline needs to ensure that: (1) no new instructions are issued out from the issue queue until the re-executions complete; (2) the incorrect execution results in the previous cycle is flushed out from the pipeline; and (3) no errant register or cache writing is committed.

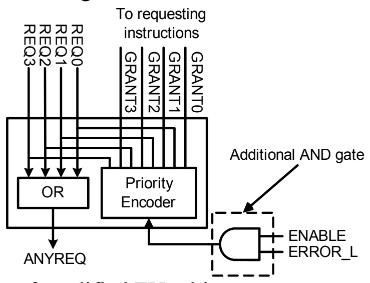


Fig. 8. Diagram of modified FU arbiter

Fig. 8 shows the modified FU arbiter for SAVS scheme. The ready instructions in issue queue can be issued out only when the ENABLE signal of the corresponding FU is raised to logic ONE [13][14][17]. In Fig. 2(b), the ERROR_L is pulled down to logic ZERO once an error in execution/bypass stage is detected in cycle i+2. We piggyback on this ERROR_L signal to block the ENABLE signal and prevent instructions from issuing out in cycle i+3. No extra performance penalty is introduced.

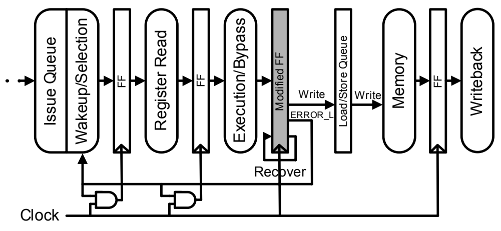


Fig. 9. Clock-gating-based stalling scheme

During the re-execution of instructions, the output latches of wakeup/selection stage and register read stage must be stalled at the end of cycle i+2. The data stored in the latches of register file stage and the correct data bypassed from execution/bypass stage are resent into execution/bypass stage for re-execution in cycle i+3. In modern high-performance microprocessors, the output of execution/bypass stage is sent to a load/store queue before it is written into the memory. Hence, the errant entry of load/store queue can easily recover by using the modified FFs of execution/bypass stage (see Section 2.1). A clock-gating-based stalling scheme is shown in Fig. 9. Compared to the modified in-order pipeline in [7], the additional pipeline stage between execution/bypass stage and memory stage (to prevent writing the errant data to the memory) is removed.

The whole pipeline is separated by issue queue into the front-end and the back-end. The stages in the front-end – fetch, decode and rename – are not stalled when the re-execution is committed. New instructions can still be fetched and sent to issue queue as long as issue queue is not full. Unlike the global clock-gating scheme implemented in an in-order pipeline [7], in our design, clock-gating is applied to the latches of only two back-end stages: wakeup/selection and register read. These two circuit blocks are located in physical proximity in the floorplan. The clock-gating control signal, which is the ERROR_L signal generated by timing-error correction circuit, can easily reach those latches by the end of current cycle.

Pipeline writes the results to register file at writeback stage. Since the information of date correctness has been available at the end of memory stage, errant writing to register file is avoided.

For pipelined multi-cycle ALU, e.g., multiplier and floating point ALU, the timing-error correction circuit is implemented on the output of each stage. Moreover, similar clock-gating scheme can be applied to every slice of multi-cycle ALU for the recovery from any possible timing error in the slice of multi-cycle ALU.

3.2. Supply-Voltage Scaling Control

3.2.1. Supply-voltage control scheme

Due to bypass mechanism, even if only one ALU operation is incorrect in the previous cycle, all the instructions being executed in the present cycle need to be re-executed. Hence, the error-induced performance and energy overheads are determined only by the error rate of program (ER_p). We also refer the error rate of an ALU as ER_a. If each ALU's error rate is independent of others', ER_p approximately equals $ER_a \cdot$IPC when ER_a is small.

Because of the process variations and the time-varying environmental factors, ER_p changes from program to program or even from portion to portion within the same program. Hence, V_{DD} must be automatically adjusted to achieve an acceptable chip yield with acceptable performance and energy overheads.

In our design, we select two switching thresholds, ER_{high} and ER_{low}, and monitor the ER_p of the running program. V_{DD} does not change when ER_p is between ER_{high} and ER_{low}. Otherwise, we decrease V_{DD} to reduce power consumption if ER_p is lower than ER_{low}, or increase V_{DD} to avoid unacceptable performance/energy overhead if ER_p is higher than ER_{high}.

However, our V_{DD} adjustment mechanism *never* reduces V_{DD} under a critical voltage level $V_{DD\text{-}min}$, and hence, guarantees a desired chip yield. This constraint is *very crucial* when the chip yield is considered. More details will be given in Section 4.2.

3.2.2. Limitations of the V_{DD} scaling

V_{DD} ramping is limited by two factors: the response time of voltage regulator and the power supply noise tolerance. [7] described a typical commercial voltage regulator that takes 10's of microseconds to adjust V_{DD} by 100mV. Moreover, V_{DD} shifting results in the charging/discharging of the intrinsic capacitance of circuits and consequently, current surge in power supply network. V_{DD} ramping must be slow enough so that the induced power supply noise is below the allowed threshold. In our experiments, we conservatively set V_{DD} ramping rate to 5mV/µs.

4. Experimental results

4.1. Yield Analysis with Timing-Error Correction Mechanism

Our Monte-Carlo simulation on chip yield is conducted by HSPICE with BPTM 70nm Technology. Without loss of generality, only the variations of Vt and interconnect width are considered in our experiments. The STDs of both inter-die and intra-die Vt variations are set to 30mV [19]. The lumped RC model of interconnect [20] is used while the STD of interconnect width for each segment of lumped RC model is set to 10% of the nominal width. The spatial correlation coefficient between the intra-die Vt variations of any two transistors and the one between the widths of any two interconnect segments are both set to 0.4. Execution/bypass (E/B) stage is implemented with 32-bit CLA and the corresponding bypass logic. Fig. 10 shows the yield of E/B stage under different V_{DD}'s. Clock period is selected to ensure that when E/B stage operates at the normal V_{DD} (1.0V), chip yield is 93.3% (*Φ(1.5)*, where *Φ* is the cumulative distribution function of a standard normal distribution).

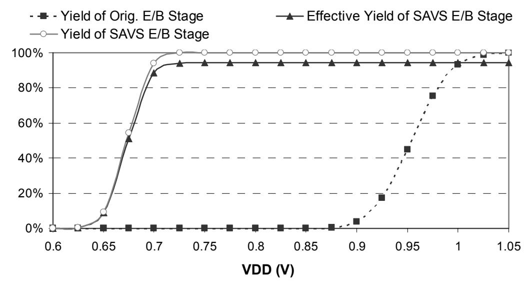


Fig. 10. Yield of ALU under different V_{DD}'s.

To take account of the area overheads of timing-error correction circuit, we introduce the effective yield, defined as:

$$Y_{chip}^{eff} = \frac{A_{orig}}{A_{orig} + A_{ovhd}} Y_{chip} \tag{1}$$

Here A_{orig} and A_{ovhd} are the area of E/B stage and the area overhead incurred by timing-error correction circuit, respectively. Y_{chip} is the yield of whole E/B stage and Y_{chip}^{eff} is the effective yield. The effective yields of the E/B stage with SAVS under different V_{DD}'*s* are also shown in Fig. 10. Here the switching time of the error-detection signal *ΔT* is set at half clock cycle (see Fig. 2(b)). The whole pipeline's yield is largely determined by the yield of E/B stage, which is the critical stage for sub-0.18μm technology and beyond [13][14]. Hence, the yield of whole chip can be improved significantly by SAVS with the negligible area overhead (compared to the whole chip area of pipeline). *It should be noted that yield considered in this paper is only due to SAVS.* In reality, the exact yield calculation for a system depends on multitude of factors and is a lot more involved.

Increasing V_{DD} can also enhance the chip yield, albeit with significant power penalty. Although the highest possible effective yield of modified E/B stage is limited by the area overhead of timing-error correction circuit, when V_{DD} is low (less than the actual needed V_{DD}, say, 1.0V), the effective yield of modified E/B stage is much higher than the yield of non-modified ones. For an acceptable yield, 93.3% (say), the modified E/B stage requires only V_{DD}=0.725V while the non-modified one requires V_{DD}=1.0V. We select the minimal V_{DD} that ensure the desired chip yield in SAVS technique as the critical V_{DD} of SAVS ($V_{DD\text{-}min}$).

4.2. Error Rate with Aggressively Scaled V_{DD}

In our error rate simulation, if the latency of an operation at the scaled V_{DD} exceeds original clock cycle (considering flip-flop set up time and hold time), it is counted as an error. Obviously, the actual error rate of a circuit greatly depends on the process parameter and environmental factor variations. Fig. 11 shows the error rates of a CLA-based ALU with 8192 random input vectors at different V_{DD}'s: when the *Vt*'*s* of all transistors in the 32-bit CLA: 1) equal the designed value (-designed). Here, the designed value is the *Vt* adopted in design time; 2) +42.4mV deviation from the designed value toward 0V (-slow); 3) -42.4mV deviation from the designed value toward 0V (-fast). We note that the error rate of ALU actually relies on the particular benchmark.

The error rate of bypass logic is different for various benchmarks. Based on the simulation results in Fig. 6, we use Wattch [21] to simulate the data bypass in an 8-way, out-of-order pipeline for 23 SPEC2000 benchmarks [22]. Results show that benchmark *art* has the highest error rate of bypass logic.

Fig. 11 also shows the error rate of E/B stage (including both ALUs and bypass logic): when the interconnect width of bypass logic: 1) equals the designed value (-designed); 2) -10% deviation from the designed value toward 0 (-slow); 3) +10% deviation from the designed value toward 0 (-fast). Here, we assume that the longest latencies of ALU and data bypass take half clock period at V_{DD}=1.0V, respectively [12][13][14]. The behaviors of ALU and bypass logic are based on 8192 random input vectors for benchmark *art*.

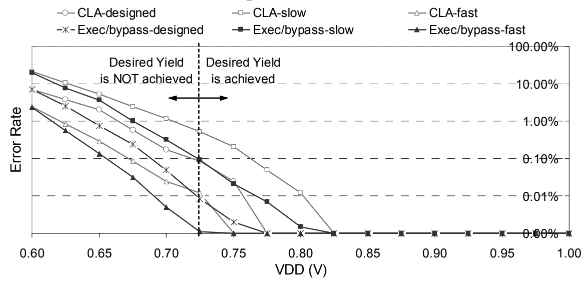


Fig. 11. Error rate of CLA and execution/bypass stage

We point out that, when V_{DD} is scaled under 0.725V, the error rate may still be at a low level: when V_{DD} is 0.7V, the corresponding error rate of E/B stage is only 0.05%. The incurred performance and power overheads is still small. However, the over scaled V_{DD}, i.e., 0.7V, cannot guarantee a desired chip yield (say, 93.3%): Some errors cannot be detected and recovered by the error correction circuitry (Section 2.1). Thus, when chip yield is considered, a pure error-rate-based (or performance/energy penalty-oriented) dynamic V_{DD} adjustment mechanism [7] is not sufficient: to maintain the desired chip yield, V_{DD} should *not* be scaled under the critical V_{DD} of SAVS (0.725V in our simulation).

In our simulations in Section 4.3, we assume all *Vt*'s of transistor and the bypass interconnect width in microprocessors equal the designed (nominal) value, The corresponding error rates of E/B stage are shown as curve "Exec/bypass-designed" in Fig. 11. [7] shows that the random input-based simulation overestimates the error rate of an ALU in reality. Since we also adopted the error rate of bypass logic in benchmark *art* to come up with the curve "Exec/bypass-designed", in our simulation, the performance penalty and energy overheads due to the errant operations are actually overestimated. Consequently, for the error-rate-driven V_{DD} scaling, power reduction is underestimated.

[7] also shows that complex logic structures, such as multipliers, generally have lower error rate than adder-based ALU's. In our simulation, we assume that integer multiplier and floating-point ALU have the same error rate as that of CLA at the same V_{DD}. This assumption does not underestimate the error rate or overestimate the effectiveness of our methodology.

4.3. System-Level Simulations

4.3.1. Simulation environment

Table II
Baseline Processor Configuration

Processor	8-way issue, 128 RUU, 64 LSQ, 8 integer ALUs, 2 integer mul/div units, 4 FP ALUs, 4 FP mul/div units, uses clock gating (DCG) and s/w prefetching
Branch prediction	8K/8K/8K hybrid predictor; 32-entry RAS, 8192-entry 4-way BTB, 8 cycle misprediction penalty
Caches	64KB 2-way 2-cycle I/D L1, 2MB 8-way 12-cycle L2, both LRU
MSHR	IL1 - 32, DL1 – 32, L2 – 64
Memory	Infinite capacity, 400 cycle latency
Memory bus	32-byte wide, pipelined, split transaction, 4-cycle occupancy

We used a modified version of Wattch to simulate an 8-way, out-of-order SAVS processor, which is summarized in Table II, under BPTM 70nm process. A deterministic clock gating (DCG) technique [23] is involved in all simulations. The extra power dissipation of the timing-error correction circuitry and the additional energy overhead due to the re-execution of errant operation have been included in our power saving and overhead estimation.

Alpha-SPEC2000 binaries that are pre-compiled with SPEC *peak* setting are accepted in our simulation. We used *ref* inputs, fast-forwarded 2 billion instructions, and simulated 500 million instructions.

In our simulation, ER_{high} and ER_{low} are set to 0.1% and 0.05%, respectively. V_{DD} ramping rate is set to 5mV/µs. Under the assumption of the 2GHz clock frequency, it takes 2000 cycles to ramp the V_{DD} up/down by 5mV.

4.3.2. Effectiveness of SAVS

Fig. 12 depicts the simulation results of SAVS pipeline on 23 SPEC2000 benchmarks. The gray bar (with Y-axis on the left) represents the total microprocessor power savings ratio (including the cache power), with respect to the 8-issue baseline processor under V_{DD}=1.0V. The line (with Y-axis on the right) shows the percentage of the execution time increase of SAVS pipeline, with respect to the 8-issue baseline pipeline under V_{DD}=1.0V. The X-axis shows benchmarks.

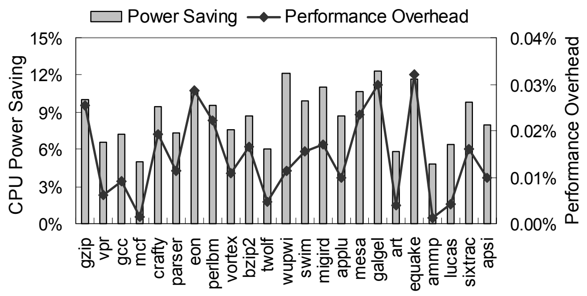


Fig. 12. Power savings and performance overhead of SAVS

The simulation results show that SAVS effectively reduces the power consumption of processor with negligible performance penalty while maintaining the same chip yield: on average, SAVS improves 8.66% of processor power consumption by paying only 0.014% performance penalty (the maximum performance overhead is within 0.032%). The average energy reduction is 8.64% while the chip yield is maintained at 93.3%.

The benchmarks with high ILP achieve higher power savings. For example *wupwise*, whose IPC is 4.10, achieves 12.1% reduction of processor power dissipation. The power savings for the benchmarks with low ILP, such as *ammp*, are insignificant. It can be explained as following: under clock gating scheme, for the program with higher ILP, ALUs are used more frequently and consume more power. Hence, more power saving can be achieved when SAVS is applied.

We point out that the application of SAVS in the critical stage of pipeline decreases the critical V_{DD} for whole system. Hence, the V_{DD} applied to other non-SAVS stages can be reduced to lower level that is determined by the new critical stage(s) of pipeline (for example, 0.85V in our simulation), without incurring additional performance, power and yield penalties. This fact provides an new methodology to design a high performance, power efficient, pipelined system under scaled technology.

5. Conclusion

In this paper, we proposed an error-tolerant self-adaptive variable supply-voltage scheme (SAVS) for multi-issue out-of-order superscalar microprocessors. Besides maintaining the chip yield, SAVS can effectively reduce power dissipation of high performance microprocessor with negligible performance degradation. SAVS is suitable for scaled technologies where process parameter fluctuations and environment factor variations are significant. SAVS also provides an alterative choice to simultaneously achieve the high throughput and low power in pipelined systems under scaled technologies.

Reference

[1] H. Li, *et. al.*, VSV: L2-Miss-Driven Variable Supply-Voltage Scaling for Low Power, *36th IEEE/ACM Int'l Symp. on Microarch.,* pp. 19-28, Dec. 2003.

[2] S. Mukhopadhyay, *et. al.*, Gate Leakage Reduction for Scaled Devices using Transistor Stacking, *IEEE Trans. on Very Large Integ. (VLSI) Sys.*, Vol. 11-4, pp. 716-730, Aug. 2003.

[3] T. D. Burd, *et. al.,* A dynamic Voltage Scaled Microprocessor System, *IEEE Jour. of Solid State Ckts.*, Vol. 35-11, pp. 1571-1580, Nov. 2000.

[4] J. Pouwelse, *et al.*, Dynamic Voltage Scaling on a Low-power Microprocessor, *Mobile Comp. Conf,* pp. 251-259, Jul. 2001.

[5] R. Ramachandran, *et al.*, Carry Logic, *Wiley Encyclopedia of Electr. and Electron. Engineering*, Edited by Joh G. Webster, 1999.

[6] T. Liu and S.-L. Lu, Performance Improvement with Circuit-level Speculation, *33rd IEEE/ACM Int'l Symp. on Microarch.,* pp. 348-355, Dec. 2000.

[7] D. Ernst, *et. al.,* Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *36th IEEE/ACM Int'l Symp. on Microarch.,* pp. 7-18, Dec. 2003.

[8] D. Lammers, TI Moves Ahead with 65-nm Chips by Next Year, *EE Times*, Mar. 22, 2004.

[9] L. Kim and R. W. Dutton, Metastability of CMOS Latch/Flip-Flop, *IEEE Jour. of Solid State Ckts.*, Vol. 25-4, pp. 942-951, Aug. 1990.

[10] BPTM, http://www-device.eecs.berkeley.edu/~ptm

[11] K.Usami, *et al.*, Design Methodology of Ultra Low-power MPEG4 Codec Core Exploiting Voltage Scaling Techniques, *35th Des. Auto. Conf.*, pp. 483-488, June, 1998.

[12] Eric S. Fetzer, *et. al.*, A Fully Bypassed Six-Issue Integer Datapath and Register File on the Itanium-2 Microprocessor, *IEEE Journal of Solid State Circuits*, Vol. 37-11, pp. 1433-1440, Nov. 2002.

[13] S. Palacharla, *et al.*, Quantifying the Complexity of Superscalar Processors. *Technical report CS-TR-96-1038*, Dept. of CS., Univ. of Wisconsin, 1996.

[14] Z. Chishti, and T. N. Vijaykumar, Wire Delay Is Not a Problem for SMT (in the near future)*, 31st Annual Int'l Symp. on Comp. Arch.*, pp. 40-51, Jun. 2004.

[15] A. Agarwal, *et al.*, A Single-Vt Low-Leakage Gated-Ground Cache for Deep Submicron, *IEEE Jour. of Solid State Ckts.,* Vol.38-2, pp. 319-328, Feb. 2003.

[16] S. Virtanen and J. Lilius, The TACO Protocol Processor Simulation Environment, *9th Int'l. Symp. on Hardware/Software Codesign*, pp. 201–206, Apr. 2001.

[17] Y. Chen, *et al.*, Integrated Architectural/Physical Planning Approach for Minimization of Current Surge in High Performance Clock-gated Microprocessors, *Int'l Symp. on Low Power Electr. Des. 2003,* pp. 229-234, Aug. 2003.

[18] T. Karnik, Probabilistic and Variation-Tolerant Design: Key to Continued Moore's Law Scaling, *Invited talk in ACM/IEEE Int'l TAU Workshop on Timing Issues*, Feb. 2004.

[19] A. Bhavnagarwala, *et. al.*, The impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability, *IEEE Jour. of Solid State Ckts.,* Vol. 36, No. 4, pp. 658-665, Apr. 2001.

[20] Y. Chen, *et. al.*, Model Reduction in the Time-domain Using Laguerre Polynomials and Krylov Methods, *2002 Des. Auto. and Test in Euro. Conf. and Exhi.,* pp. 931-935, Mar. 2002.

[21] D. Brooks, *et. al.*, Wattch: A Framework for Architectural-level Power Analysis and Optimizations, *27th Int'l Symp. on Comp. Arch.*, pp. 83-94, June 2000.

[22] http://www.eecs.umich.edu/~chriswea/benchmarks/spec2000.html

[23] H. Li, *et. al.*, Deterministic Clock Gating for Microprocessor Power Reduction, *9th Int'l Symp. on High-Perf. Comp. Arch.*, pp. 113-122, Feb. 2003.