2015 10th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

Online Self Adjusting Progressive Age Monitoring of Timing Variations

Somayeh Sadeghi-Kohan[1], Mehdi Kamal[1], John McNeil[2], Paolo Prinetto[3], Zain Navabi[1]

[1]School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

[2]School of Electrical and Computer Engineering, Worchester Polytechnique Institute, Worchester, USA

[3]Politecnico di Torino, Dip. di Automatica e Informatica, Corso Duca degli Abruzzi 24, I-10129 Torino TO, Italy

[1](sm.sadeghi79, mehdikamal, navabi @ut.ac.ir), [2] (mcneill@ece.wpi.edu), [3](Paolo.Prinetto@polito.it)

***Abstract*— Transistor and interconnect wearout is accelerated with transistor scaling that results in timing variations. Progressive age measurement of a circuit can help a better prevention mechanism for reducing more aging. This requires age monitors that collect progressive age information of the circuit. This paper focuses on monitor structures for implementation of progressive age detection. The monitors are self-adjusting that they adjust themselves to detect progressive changes in the timing of a circuit. Furthermore, the monitors are designed for low hardware overhead, and certainty in reported timing changes.**

***Keywords*— self-adjusting monitors, phase-shift clock, aging phenomena**

I. INTRODUCTION

Technology scaling and transistor's size reduction increase variations in various physical parameters like transistor threshold voltage and timing. Although static variations are detectable after chip fabrication, dynamic variations could only emerge at the operational phase of the system at runtime. These variations result in cores to operate at a slower speed than they were originally designed for. Two main physical phenomena that progressively cause slowing down cores are Bias Temperature Instability(BTI) and Hot-Carrier Injection (HCI) [2,3], which are caused by stress on transistors due to continuous movement of charges [1]. These effects have direct influence on current and electrical parameters of transistors and consequently are emerged as slower transistors and therefore, slower cores. Aging phenomena reduces the life time of a core and its mean time to failure (MTTF). Clearly, the aging process has important effects on the design and cost of processors. Therefore, we should monitor aging rate of a system.

In this paper, we propose a new self-adjusting age monitoring method that provides the required information about the present state of the core (new slack time) and also the aging rate. In our proposed method, the output of the critical paths are captured using an age monitoring clock (that has a rising edge that occurs before that of the system clock), and the captured data is compared to data captured at the rising edge of the system clock. This mechanism enables our proposed monitoring scheme to detect progressive changes in delay of individual cores. Therefore, this information can be used by an aging prevention method for mitigating the wear-out of a core.

In the rest of this paper, details of the proposed clocking structure and age monitoring structure are presented in Section II, followed by the conclusions in Section III.

II. AGE MONITOR

In this section, we primarily focus on processor core monitors. It needs to be reminded that we are dealing with a many-core, and identical logic clusters in all cores are being monitored by identical monitor structures. Furthermore, we are assuming that an individual core is a pipe-line processor, whose timing bottle-neck is its ALU. For the processor, delay of a critical path section of its ALU must be monitored.

A. Pipe stage hardware

The dotted areas of Figure 1 show pipeline stage of a pipeline processor between pipe-stage registers *i* and *j* (PSR_i, PSR_j). The combinational cloud between these registers is marked, and in most processors, this represents the processors ALU that has the longest delay of all pipe stages. Pipe stage registers use the system clock that we assume a 50% duty cycle clock and rising-edge trigger registers.

System clock frequency is set for the slowest path in the circuit, which means that the output of the pipe stage combinational cloud will be ready some time before the next edge of the system clock, thus slack time. Changes in this slack time present processor's age.

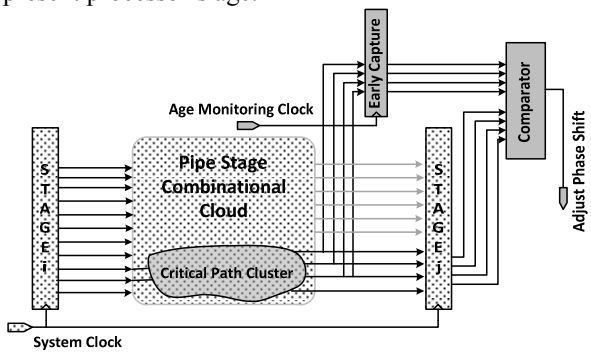

Figure 1. Pipeline-stage, and monitoring

As shown in Figure 1, for measuring processor's age, a critical section of the combinational cloud is selected. This section that we refer to as *critical path cluster* is selected based on activity, aging and a balanced delay between variations paths. The selection mechanism for this cluster is discussed in the next section. Age monitoring uses a clock that lags behind the system clock by the amount of slack time of the critical path cluster. The original after manufacturing delay of this cluster is measured, and the age monitoring clock (AM-Clock) is adjusted accordingly. Figure 2 shows timing diagram and slack time of the critical path cluster of Fig. 2. The AM-clock is created by a phase shift of the system clock.

The gray areas of Figure 1 show parts that are considered and are added to the pipeline stage for measuring the age of a processor core. Initially, the at-birth critical path delay of the critical path cluster is measured, and the AM-clock is set accordingly. Every time this pipe stage is exercised, the output of the critical path cluster is clocked in the pipe stage $Register_j$ (RSR_j) by the system clock, and in the Early stage Register by the AM-clock. Captured data are then compared. Because of

978-1-4799-1999-4/15/$31.00 ©2015 IEEE

allowed slack time, initially the comparator shown indicates that data in the two registers are equal. However, as the processor core ages, delay of its critical path increases, and the comparator output indicates the discrepancy of data captured in ESR and PSR_j. when this happens, the AdjustPhaseShift (APS) signal becomes 1, to readjust the AM-Clock. Readjusting is done by increasing the phase shift between the system clock and AM-Clock. This reduces the slack time, allowing more time for the critical path cluster.

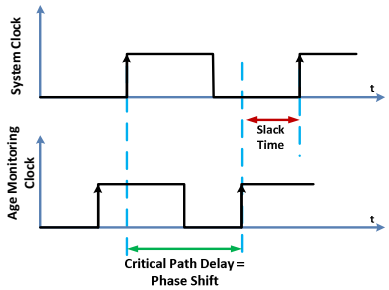

Figure 2. Critical path cluster slack time

B. Readjusting age monitoring clock

Structure shown in Figure 3 takes input from the comparator of Fig. 1 (left-hand side), and readjusts the AM-Clock, which is feedback to Figure 1 for further sampling of critical path cluster. In addition to AM-Clock output (on the right-hand side), this circuit outputs Progressive Age Indication (PAI) and the initial at-birth process characteristic of the core. The original slack time is based on core process characteristic (CPC), and the reduction in slack time (larger phase shift) is indicated by the PAI output. These outputs become available for system level scheduling decision making.

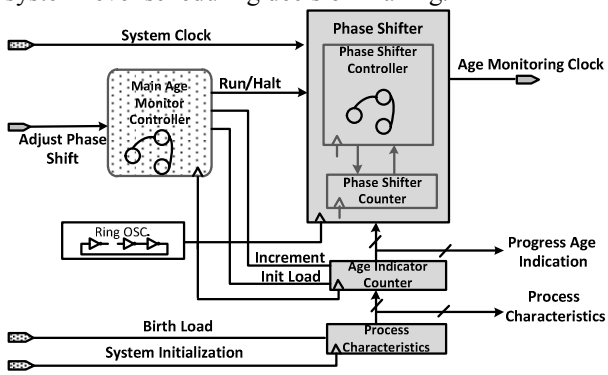

Figure 3. Readjusting age monitor clock

Readjusting AM-Clock is done by phase-shifting the system clock. The amount of phase shift is decided by increments of a fast clock that is generated by a Ring Oscillator shown on the left-side of Fig. 3. The Age Indicator Counter (AIC) on the lower right side of Figure 3 contains the count of Ring Oscillator pulses for this phase shifting. Initially, this counter is loaded from CPC (core process characteristic), that is initialized by system initialization at the birth time. As the core ages, the AIC counter is incremented causing a longer phase shift, and a shorter slack time.

Figure 4 shows the main age monitor controller that is shown in the upper left of Fig. 3. The controller waits for an unequal signal from the comparator of Fig. 1. When received, it halts the phase shifter, and issues Adjust-Phase-shift to increment the AIC counter. It then resumes the phase-shifter operation. This controller works with the Ring Oscillator clock.

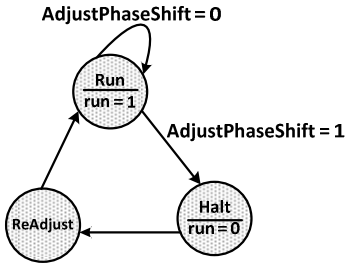

Figure 4. Age monitor controller

C. Phase shifter

The phase shifter that is shown on the upper right side of Fig. 3 takes the system clock as input and produces AM-clock by shifting the system clock *n*Ring Oscillator cycles (see timing diagram of Fig. 2). Number *n* is given to the phase-shifter block as an input. This block has a counter to count down *n,* and a controller that produces falling and rising edges of the Age Monitor clock. Figure 5 shows the AM-Clock that is generated from the system clock in terms of the Ring Oscillator clock.

Parameter *m* is the number of Ring Oscillator clock cycles that fit in a system clock cycle.When the phase-shifter is in the*run* mode, it waits for the rising edge (falling edge) of system clock and produces the falling edge (rising edge) of the AM-clock *n – m/2* Ring Oscillator cycles later.

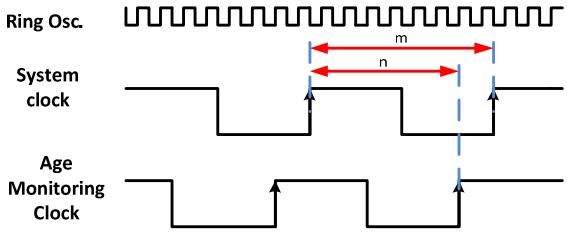

Figure 5. Phase shifter waveform

The combination of hardware blocks in Figures 1 and 3, and controllers of Figures 2 and 4 show the progressive age of a processing core. Age monitoring causes the age monitor clock to keep shifting in time as the core ages. When AM-Clock edge reaches the system clock edge, slack time becomes zero, and core will be marked as defective.

III. Conclusions

For balancing aging in a many-core, age monitors that provide time-dependent aging information are added to individual cores. This information together with the initial state of cores provide online age and rate of aging of individual cores. This information becomes available to a top-level task scheduler assigning tasks to cores of a many-core. With this information, the scheduler is able to dynamically assign tasks not only to consider present age of a core, but also how fast core is aging. Our focus in this paper was on online hardware of the monitors and their application. The task of developing the top-level task scheduler is left as a future work.

IV. References

[1] A. Tiwari, et al., "Facelift: Hiding and slowing down aging in multicores," in MICRO Conference, 2008.

[2] K. Bernstein, et al.,"High-performance CMOS variability in the 65-nm regime and beyond" in IBM Journal of Res. and Dev., 2006.

[3] A. Snavely, et al. "Symbiotic job scheduling for a simultaneous multithreaded processor," in Architectural Support for Programming Languages and Operating Systems, 2000.