Fuse: A Technique to Anticipate Failures due to Degradation in ALUs

Jaume Abella, Xavier Vera, Osman Unsal[†], Oguz Ergin[‡], Antonio González
Intel Barcelona Research Center, Intel Labs – UPC
{jaumex.abella, xavier.vera, antonio.gonzalez}@intel.com

Abstract

This paper proposes the fuse, a technique to anticipate failures due to degradation in any ALU (Arithmetic Logic Unit), and particularly in an adder. The fuse consists of a replica of the weakest transistor in the adder and the circuitry required to measure its degradation. By mimicking the behavior of the replicated transistor the fuse anticipates the failure short before the first failure in the adder appears, and hence, data corruption and program crashes can be avoided. Our results show that the fuse anticipates the failure in more than 99.9% of the cases after 96.6% of the lifetime, even for pessimistic random within-die variations.

1. Introduction

As technology evolves, the geometry of transistors and wires shrinks. However, supply voltage does not scale at the same pace [23]; this causes transistors and wires to suffer higher current densities, which also imply higher temperatures. The increased current density and temperature translate into higher vulnerability of circuits. Under these conditions, transistors and wires will degrade faster and will be more prone to failures (higher failure rate per device). Furthermore, there will be an increased number of failures in the chip because of the larger number of such devices (transistors shrink but the chip size is expected to remain constant [23]).

The increasing unreliability of processors will make devices fail frequently during the normal lifetime of the processor. Moreover, transistor geometry may change significantly from one chip to another or even within the chip itself, in such a way that some components are prone to degrade faster than others. Similarly, dynamic variations of operating frequency, voltage and temperature may accelerate degradation significantly for some blocks. Thus, lifetime of blocks in a chip is unpredictable and mechanisms are required to detect failures before such failures produce crashes or data corruption.

Such unreliability can be addressed in several ways. One solution consists in testing the blocks for errors [11][18] and reconfigure the system accordingly. However, testing only avoids future crashes, but it does not prevent the system from crashing whenever failures show up for the first time.

Another set of solutions is based on detecting failures and avoiding data corruption. Memory-like structures, such as caches and register files, can be protected with ECC [9], which is useful to detect transient and permanent errors. ALUs are very likely to cause crashes and data corruption because most of the instructions use them, and thus, it is mandatory to protect them. However, combinational blocks like ALUs cannot use ECC or parity, and require more expensive techniques like reexecution or residue codes computation among others. Reexecution can be performed in a different instance of the same type of ALU [16][17] or in a special ALU devoted to error detection [4]. Both solutions are expensive either in terms of performance and/or extra hardware. Residue computation [5][12] is an alternative to detect failures in ALUs, especially in adders. Obtaining residue codes requires special hardware to compute the modulo function.

This paper proposes the fuse, a new technique to anticipate failures due to degradation in ALUs at a very low cost. In particular, we propose the design and implementation of a fuse to anticipate failures in a sparse tree adder [14], which is the one used in the Intel® Pentium® 4. The fuse is built as a replica of the weakest transistor in the adder and the circuitry required to measure its degradation. Whenever the fuse does not meet the delay constraints, it implies that the protected adder is *about* to fail, so it can be disabled or its frequency decreased [19] to prevent data corruption and program crashes.

The fuse is a very efficient solution in terms of hardware and power. We illustrate how to design and implement a fuse for a sparse tree adder, although the same idea can be extended to any other ALU without requiring any special property for the protected block, as it is the case for residue computation.

The rest of the paper is organized as follows. Section 2 introduces the main sources of failure affecting microprocessors. Section 3 presents the fuse, our technique to anticipate failures in adders. Section 4 presents the evaluation of the fuse. Section 5 reviews

[†] Osman Unsal is currently with the Barcelona Supercomputing Center, Spain (*osmal.unsal@bsc.es*)
[‡] Oguz Ergin is currently with the TOBB University of Economics and Technology, Ankara, Turkey (*oergin@etu.edu.tr*)

13th IEEE International On-Line Testing Symposium (IOLTS 2007)
0-7695-2918-6/07 $25.00 © 2007 IEEE

some related work. Future work is introduced in Section 6. Finally, section 7 draws the main conclusions of this work.

2. Sources of Wearout and Failure

There are many sources of failure (SOF) [24] affecting transistors and wires such as electromigration, stress migration, time-dependent dielectric breakdown, negative bias temperature instability (NBTI), etc. . Although our invention can be used to detect failures due to degradation for most of such SOF, we focus on a single SOF for the sake of illustration: NBTI. NBTI, which mainly affects PMOS transistors, has emerged in recent years as the most important SOF for transistors in sub-130nm technology as shown in literature [3]. Our simulations show a similar trend since transistors fail much earlier due to NBTI than due to any other SOF. Some molecules at the gate interface are broken when the input voltage of the gate is negative and the source voltage is positive. As a consequence, the threshold voltage increases and the gate becomes slower, leading to failures due to time constraints. NBTI depends on temperature, voltage, utilization and transistor geometry.

3. Fuse for a Sparse Tree Adder

This section presents the *fuse* for a sparse tree adder [14]. The fuse is used to detect when a failure due to degradation is about to happen in the adder, but it does not detect the *actual* failure. A fuse consists of a transistor and some extra logic to (i) degrade it properly and (ii) measure its degradation. When the fuse fails (i.e., it does not meet the delay constraints), the adder is considered to be unsafe. As a result, the system may disable the adder or try to find a frequency [19] such that the fuse (and thus the adder) still meets the delay constraints. Similarly, such scheme can be used to remove part of the guardband that has been set up to tolerate some extent of degradation.

A scheme of the adder can be found in Figure 1. The implementation used for the adder is the one provided by the authors of the adder and both the adder and the fuse have been designed in 65nm technology for Hspice-like simulation. The simulator is an Intel production aging simulator whose inputs are the spice description of the circuit, the technology description, the environmental parameters of the simulation (mainly temperature and voltage), the length of the simulation (how many years the circuit is expected to work), and the inputs for the circuit. Note that we do not feed the circuit with inputs for its whole lifetime. Instead, we provide the simulator with inputs for a given period of time, and the simulator repeats the circuit inputs for the whole lifetime. The experiment has been run sampling 1138 instructions requiring additions from a set of 569 traces from different workloads (see Table 1). Longer experiments have not been possible due to the large cost of such a long simulation at electrical level. Simulations have been conducted assuming a temperature of 110ºC, a supply voltage of 1.2V, and a lifetime of 7 years. Note that operating conditions (temperature and voltage) are those considered for TDP (Thermal Design Point), which are the ones used to decide whether a circuit passes or fails the test.

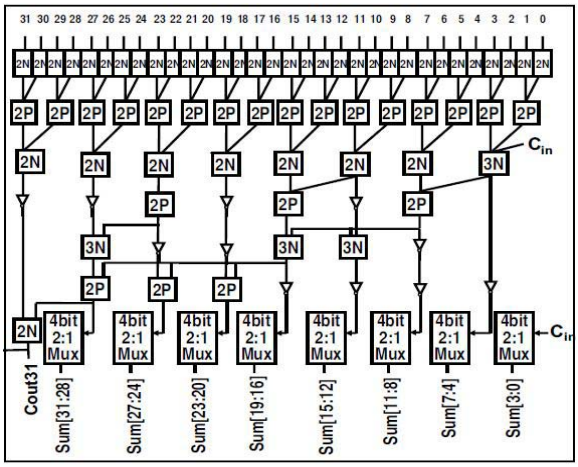

Figure 1. Schematic of a 32-bit sparse tree adder [14]

Table 1. Workloads

Benchmark suite	# traces	Description
Encoder	62	Audio/video encoding
SpecFP2000	41	Floating-point specs
SpecINT2000	33	Integer specs
Kernels	53	VectorAdd, FIRs
Multimedia	85	WMedia, photoshop
Office	75	Excel, Word, PowerPoint
Productivity	45	Internet contents creation
Server	55	TPC-C
Workstation	49	CAD, rendering
Spec2006	71	New spec release

The following sections identify and address the several key issues when designing the fuse. The first one is identifying the proper characteristics of the fuse; the second is designing the circuit in such a way that it is guaranteed that there is no transistor in the adder that will fail due to degradation before the fuse. Finally, we address the problem of pessimism: it must be minimized the time elapsed between the fuse failure and the time when the adder would have failed.

3.1 How to Identify the Required Fuse

The fuse replicates the weakest transistor of the adder. The weakness of transistors depends on different critical intrinsic SOF. Our simulations show that NBTI is much more significant than any other SOF affecting transistors for the evaluated technology. Thus, to design the fuse we must care about the weakest transistor due to NBTI.

We choose as fuse the transistor whose combination of physical characteristics and activity patterns gives the shortest lifetime in terms of NBTI. Vulnerability depends

on the length and width of the PMOS transistors, and its inputs (i.e. the longer, narrower, and more time with input to zero, the more vulnerable). Since all transistors in the adder have the same length, only their width and inputs must be considered. How to weight the different factors is still an open research topic [1][2][20][22]. However, our empirical results as well as the literature [10] show much higher dependency on the input than on the size of the transistor.

The weakest transistor for NBTI corresponds to the PMOS with smallest size and whose input is "0" more time than any other PMOS transistor. Regarding the size, the narrowest transistor has a width of 280nm (there are some tens of these transistors in the adder). On the other hand, the PMOS transistor whose input is "0" more time is one of those with smallest size, and therefore it is the weakest transistor in the adder (we refer to it as transistor T_{weak} for the sake of commodity). Such PMOS transistor (part of the "4bit 2:1 Mux" block outputting "Sum[3:0]" in Figure 1) has as input the clock signal when C_{in} is "1", otherwise its input is "0". C_{in} is "1" only for subtractions and additions with carry, which represent around 4-5% of the instructions requiring integer additions in the samples obtained from our traces. Hence, only about 2% of the time the PMOS transistor has input "1" (half of the time with the C_{in} set because of the clock signal), and 98% of the time the input is "0", which is the worst-case input.

Note that it could be the case that the PMOS whose input is "0" more time than any other (T_{input}), and the smallest PMOS (T_{small}) were different transistors. In that case electrical simulation is required to identify the weakest PMOS transistor in the adder. In general, the fraction of time with the input set to "0" is much more relevant than the width of the transistor, and therefore, it is very likely that T_{input} is the weakest PMOS.

3.2 How to Implement the Fuse

In this section we discuss the mechanisms that degrade the fuse and the methodology to detect whether the fuse is working properly.

Input data for the adder vary dynamically. Usually, if the fuse is stressed assuming its worst-case input data all the time, it will fail long before the adder, becoming excessively pessimistic. Thus, the fuse has its own logic to determine when to stress it based on the inputs of the adder; this logic has to guarantee that no transistor in the adder degrades faster than the fuse, and at the same time, that the fuse will fail shortly before the adder would.

To set up the fuse of the sparse tree adder in Figure 1, it must be guaranteed that it is stressed at least as much time as the most stressed PMOS transistor, which is T_{weak}. This is achieved by enforcing the fuse to be stressed with the same input as T_{weak}. To avoid interfering the circuitry of the adder, T_{weak} input is not snooped but rather replicated, which just requires an AND gate with the clock signal and C_{in} as inputs.

In order to check whether the fuse meets the delay requirements, a ring oscillator [13] is used. Ring oscillators consist of an odd number of inverters arranged in a ring-manner so that the outputs of all inverters switch continuously. The switching time depends on the latency of the slowest component in the ring. Since the delay of each block is estimated at design time, the degradation of a fuse can be measured by integrating it in a ring oscillator and counting the number of times that the ring oscillates in a given number of cycles. If such number cannot be obtained statically, we can obtain it dynamically when the processor is turned on for the first time, because the fuse is not degraded yet. The number of cycles obtained must be decreased by the fraction of time that the adder (and hence, the fuse) is allowed to degrade while still considering that the circuit operates properly.

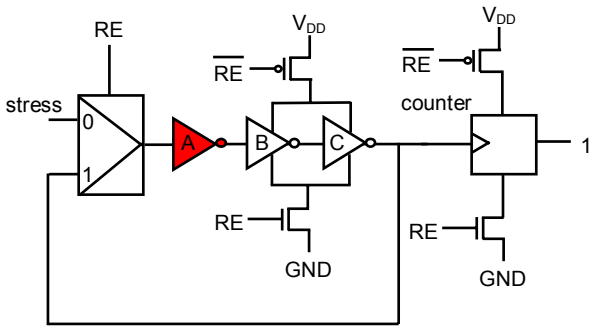

Figure 2. Fuse (in dark) and its ring oscillator to measure fuse delay (*RE* stands for *ring enable*)

Figure 2 presents an implementation of the circuit required for the fuse. The ring oscillator consists of 3 inverters (*A*, *B* and *C*). The transistor being used to detect failures is integrated in the inverter *A* of the ring oscillator. The circuit works as follows:

- During normal operation, *RE* signal (*ring enable*) is low. In this state the multiplexer (e.g. two pass-gates) drives the *stress* signal, and the inverter *A* works. The rest of the circuit is disabled. Should the output of the inverter *A* feed any circuit due to testing requirements, we can add a NMOS transistor whose source and drain are connected to ground and whose gate is connected to the output of *A* with a pass-gate that is activated during normal operation.
- Whenever it is time to check whether the fuse works properly, *RE* is high. In this state inverters *A*, *B* and *C* oscillate, and the counter tracks the number of oscillations. At the end of the checking period, the value of the counter is compared with the minimum number of oscillations computed at design time that are required to consider that the circuit works properly. If the number of oscillations is below it, the fuse does not fit its delay requirements, which means that the fuse fails.

One important observation must be done regarding the design of the ring oscillator. The inverters of the ring oscillator are sized in such a way that the slowest component in the ring is the device being stressed. For instance, for the design shown in Figure 2 it implies making *A* slower than *B* and *C*. This is achieved by using

larger, and hence, stronger transistors for inverters *B* and *C*.

Several other constraints must be taken into account when sizing transistors of inverters *B* and *C*. As told above, they must be larger than *A*, but they cannot be too large because *A* must be able to feed *B*, and *B* must be able to feed *C*. Finally, *C* must be able to feed *A* and the counter. At the end, they are sized in such a way that the output of *A* is amplified to feed the counter. The sizes of the NMOS and PMOS transistors of the three inverters as well as the total size of the transistors in the counter fed by *C* are detailed in Table 2. The PMOS transistor of inverter *A* is the replica of T_{weak} so its width is 280nm. As we can see, inverter size grows at a rate of 5X.

The checking period must be measured with a non-degrading circuit to ensure that the number of oscillations of the ring is tracked during a constant period of time during product lifetime. If there is no such device inside the chip, the system clock can be used as reference (e.g., the checking period can be the time elapsed between two consecutive clock interrupts).

Table 2. Transistor sizes for the ring oscillator and their extra fan-outs

Circuit	NMOS size (nm)	PMOS size (nm)	Total size (nm)
Inverter A	200	280	480
Inverter B	1000	1400	2400
Inverter C	5000	7000	12000
Counter	32540	12160	44700

The output of the inverters when they oscillate is shown in Figure 3. They oscillate between 0.2V and 0.8V for inverter *A*, between 0.1V and 1.1V for inverter *B*, and between 0V and 1.2V for inverter *C*. As we can see, their outputs are strong enough to feed all transistors at their output. Note that if further amplification of the signal is required, an even number of inverters can be added into the ring oscillator to produce stronger outputs.

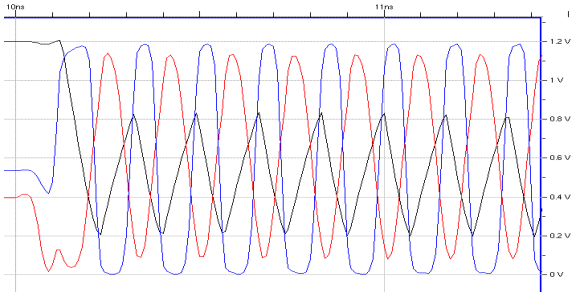

Figure 3. Behavior of the ring oscillator for the output of inverters *A* (black line), *B* (blue/dark gray line) and *C* (red/light gray line)

3.3 How to Make the Fuse Fail Short before the Adder

At first sight, it seems that ensuring that the fuse fails short before the adder would require stressing it just a bit more than the adder or using a slightly weaker replica of T_{weak}. However, variations [6][8] (i.e., process variations, voltage changes and temperature) may cause the circuit to behave differently from expected. When designing the fuse, variations may make the fuse weaker or stronger than the original transistor (T_{weak}). Thus, to ensure that the replica is weaker than T_{weak}, the replica width must be slightly reduced.

There are different sources of variation that may make the fuse and T_{weak} to behave and degrade differently:

- **Temperature variations**. Transistors at different locations in the chip may observe different temperature that may make them degrade differently. Fortunately, this is not the case for the fuse and the adder if both pieces of logic are placed one next to the other as explained later [7][8].
- **V_{DD} variations**. It is common observing V_{DD} changes during operation due to some blocks being turned on or off. That can make the V_{DD} to be different in different parts of the chip. Similarly to the case of temperature, this is not an issue for the fuse and the adder because both pieces of logic are placed one next to the other so that they observe the same V_{DD} [7][8].
- **Systematic within-die (WID) process variations**. Some transistors sized to be identical may have different sizes if they are in different parts of the chip due to systematic process variations. Such variations are regular and predictable, and may have some impact between distant parts of the chip ($\sigma/\mu \sim 3.5\%$-4% [7]), which is not the case for the fuse and the adder if they are placed one next to the other. Thus, systematic variations between the fuse and T_{weak} are negligible [7][8].
- **Random WID process variations**. Due to imperfections in the fabrication process there are some random variations affecting the size of transistors, and hence, their weakness. Such variations are in the order of 3.5%-4% for σ/μ [7][21]. Thus, the fuse must be made narrower (and hence, weaker) to take into account such variations to ensure that the fuse is weaker than T_{weak}, and therefore, the fuse fails before the adder and the failure is effectively anticipated. Note that the narrower the replica of T_{weak} is, the earlier it will fail. Thus, the replica of T_{weak} in the fuse must be slightly narrower than T_{weak} to anticipate the failure but not too much to not report the failure so early.

In summary, only random WID variations [6][8] are an issue for the design of the fuse. They can be handled by making the fuse a bit weaker by adjusting its size. In order to effectively remove temperature, voltage and systematic WID variations [6][8], the fuse must be placed as close as possible to the adder. Note that both the adder and the

fuse are very small pieces of logic, and hence, there are no variations other than random WID process variations. In this way, the fuse mimics the adder behavior more accurately. The weakness of the fuse must be increased by a factor that depends on the expected amount of random WID variations: a scenario with large random WID variations requires increasing more the extra weakness of the fuse, and therefore, it becomes more pessimistic. Detailed results about how much it must be narrowed to make it weaker are reported in the next section.

4. Fuse Evaluation

This section evaluates the *fuse* for the sparse tree adder. First, some results are presented to illustrate the theoretical impact of random WID variations in the lifetime and accuracy of the fuse. Later, circuit simulations considering variations for both the adder and the fuse show the effectiveness of the fuse as a mechanism to anticipate failures in the adder.

4.1 Analytical Evaluation: Lifetime and Accuracy of the Fuse

In order to measure the lifetime of the fuse and the adder, as well as the accuracy of the fuse, three different scenarios have been considered for random WID variations [6][8] between the adder and the fuse: optimistic (σ/μ=2%), pessimistic (σ/μ=5%), and highly pessimistic (σ/μ=8%). According to our simulations and state-of-the-art measurements [10], a transistor whose width decreases by x% has a lifetime (x/5)% shorter. For instance, using a transistor in the fuse 10% narrower than its counterpart in the adder implies that its lifetime is 2% shorter.

Figure 4 shows (a) the expected lifetime of the fuse for a sparse tree adder whose lifetime is 7 years, and (b) the probability of having a failure in the adder before the fuse signals an error for different percentages of width decrease (from 0% to 30%).

For the pessimistic scenario, the probability of anticipating the failure[1] is 99.98%[2] for the fuse when adding 20% weakness (20% narrower fuse). The failure is anticipated on average around 0.25 years[3] (after 6.75 years in-use). For the optimistic scenario, the coverage is 99.9998% for the fuse when its weakness is increased only by 10%. The pessimism is 0.12 years on average (the failure is detected after 6.88 years in-use). Thus, by adding an extra weakness of 20% (or only 10% for the optimistic scenario), the adder practically never fails before the fuse when considering both optimistic and pessimistic scenarios for WID variations, and the failure is anticipated few weeks before the actual failure in the adder shows up. For evaluation purposes we have included a highly pessimistic scenario with random WID variations of σ/μ=8%. In this case, a coverage of 99.93% can be achieved by increasing the weakness of the fuse by 30%. In this case the pessimism is 0.36 years on average (the failure is detected after 6.64 years in-use).

The overhead in terms of area and energy is described in the next subsection. Finally, since the lifetime of the circuits is expected to be several years, the interval to check the fuse can be any order of magnitude (minutes, hours or days) without compromising their accuracy/well-function. The time required to check the fuse is just a few cycles.

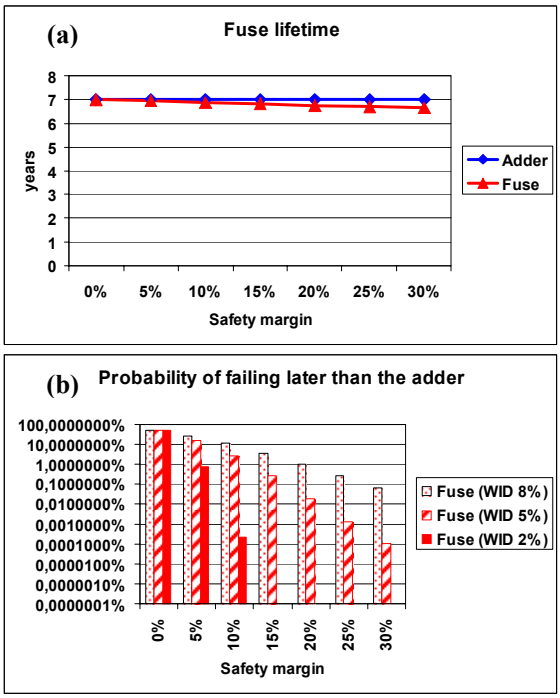

Figure 4. Lifetime and detection capabilities of the adder fuse

4.2 Empirical Evaluation

Electrical level simulations have been run for the adder and the fuse with in-home simulators based on state-of-the-art degradation models for transistors.

Figure 5 shows the output of the ring oscillator (output of inverter *C* in Figure 2) before and after degradation. We can observe that for a given period of time (around 1ns), there are 7 oscillations before degradation (black line) and only 6 after 7 years of degradation at TDP conditions (red/gray line). This information is caught by the counter, whose content after the measurement is used to decide whether the ring oscillates fast enough or not. If it is the latter case, then the fuse fails, and therefore the adder is very likely to fail soon.

[1] We refer to this probability as *coverage* in the rest of the paper
[2] Note that values above 99.9% are very high in the reliability community
[3] We refer to the time of failure anticipation as *pessimism* in the rest of the paper

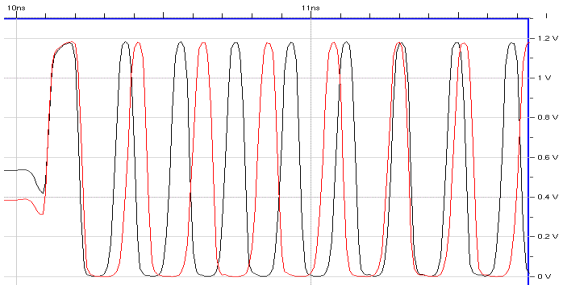

Figure 5. Waveforms for the ring oscillator before (black line) and after degradation (red/gray line)

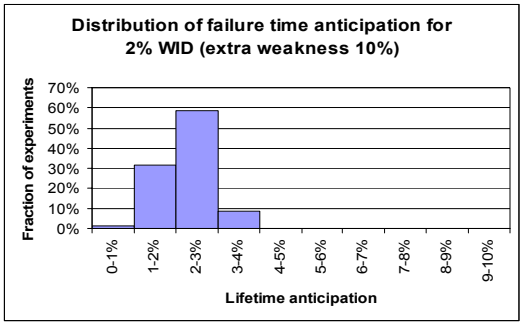

Figure 6. Failure anticipation for optimistic random WID variations of σ/μ=2%

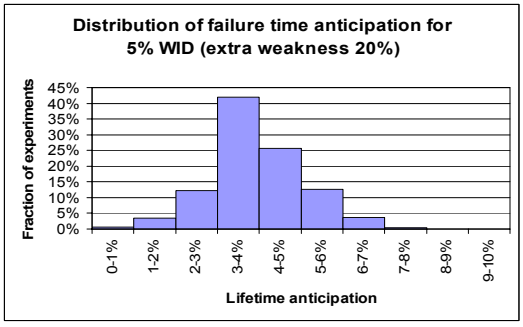

Figure 7. Failure anticipation for pessimistic random WID variations of σ/μ=5%

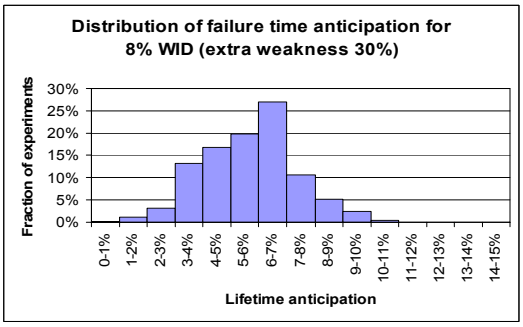

Figure 8. Failure anticipation for highly pessimistic random WID variations of σ/μ=8%

In order to evaluate the real anticipation of failures provided by the fuse and how often it anticipates the failure of the adder, three Monte-Carlo experiments have been set up. Each experiment consists of 10K individual samples where transistor geometry is set according to the random WID variations for both the adder and the fuse. Once we have the 10K samples, the lifetime for both the adder and the fuse can be measured with the Hspice-like simulator for all samples. Then, we count how many times out of the 10K samples the fuse anticipates the failure on time and how much the failure is anticipated. Each experiment has two parameters: the level of variations and the decrease in size of the PMOS transistor of inverter *A* in the fuse (the replica of the weakest transistor in the adder). We have set up experiments with optimistic random WID variations (2%) and 10% size decrease, pessimistic random WID variations (5%) and 20% size decrease, and highly pessimistic random WID variations (8%) and 30% size decrease, which are expected to be safe the very most of the times (the coverage is higher than 99.9% in all cases). Results are presented in Figure 6, Figure 7 and Figure 8. Our results show that for 10K samples the failure is always anticipated. On average, the failure is anticipated by 1.7% of the lifetime for the optimistic scenario (e.g., the failure would be detected after 6.88 years for an adder whose lifetime is 7 years), and only in some cases the failure is anticipated by 4% of the lifetime. For the pessimistic scenario failures are anticipated by 3.4% of the lifetime on average, and failures are anticipated by up to 7% of the lifetime in more than 99% of the cases. Finally, even for the highly pessimistic scenario failures are anticipated only by 5.2% of the lifetime on average, and failures are anticipated by up to 10% of the lifetime in more than 99% of the cases.

Note that recommended assumptions for today's random WID variations are 3.5-4% [7][21], so the lifetime anticipation will be between 2% and 3%, with a coverage higher than 99.9%. Both the accuracy and the lifetime anticipation are noticeable because the failure is anticipated right before it actually happens in the adder with very high coverage. If further accuracy is desired, some more lifetime should be sacrificed, which could be the case of highly reliable servers. Conversely, if lower lifetime anticipation is desired, some coverage will be lost, which could be the case for low-end computers.

The overhead of the fuse in terms of area and energy is rather small. Our results for the fuse of the sparse tree adder show that the area overhead is around 4% of the adder area. Most of the overhead corresponds to the counter, which occupies most of the area. Such overhead can be further reduced if several adders are in place because we could set up only one counter for the fuses of the different adders, and it could be used in a round-robin fashion by the fuse of each adder. The power overhead is negligible because most of the time the ring oscillator is off (it is used only every several minutes or hours), and stressing the fuse just requires plugging a signal that hardly switches to inverter *A*.

0-7695-2918-6/07 $25.00

5. Related Work

Detecting failures in circuits has been a concern for years, so the literature in this area is abundant. Nowadays, processors go through the burn-in process where they are stressed at extreme environmental conditions (very high temperature and voltage) for some hours or days to weed out those that would have failed prematurely. During burn-in, processors are tested with some on-chip special hardware that can be also used to test processors during normal life. Main approaches for processor testing [11][18] consist of mechanisms to detect and correct failures once they have appeared. Typically, these mechanisms take control of the circuit and test those inputs that are expected to identify most of the errors, comparing the obtained outputs with the expected outputs. None of these mechanisms anticipates failures. Hence, if failures are frequent, the system becomes unreliable.

Some mechanisms have been proposed to detect failures before committing wrong results in such a way that program crashes and data corruption are avoided. Mechanisms for storage structures like caches and register files consist in generating a signature of any value when it is written and checking such signature when the value is read. If the actual signature of the data and the one previously stored do not match, a failure has been found. Testing techniques can be used to check whether the failure was a soft error or a hard error. Signatures can be very simple like parity (a single bit), which does not provide error correction, or more complex like ECC [9].

Techniques to protect combinational blocks such as ALUs have been also proposed. Such techniques are based on computing a property of the output based on the inputs and checking whether the output has this property or not. Among those techniques residue codes are the most popular solution. Residue codes [5][12] are based on computing the residue of the result in two different ways: as the modulo of the result itself, and as the modulo of operating the residues of the input operands:

$$(a\ op\ b) \bmod x = ((a \bmod x)\ op\ (b \bmod x)) \bmod x \qquad (1)$$

In equation (1), a and b stand for the input operands, and op stands for the ALU operation (addition, subtraction, multiplication, etc.). Whenever both residues do not match, there is an error and the ALU can be tested to find out whether the error is soft or hard. Residue codes involve a tradeoff between accuracy and cost. In order to be effective and not very expensive, x must be of the form 2^n-1. For instance, if x is 7 residue codes are 3-bit codes, and the coverage is 87.5%. Such coverage can be increased by using larger values of x, and hence, larger residue codes, but the cost is high to compute such larger residues. Such logic is expensive in terms of area and power (especially much more than the fuse), and its coverage is largely below the coverage of the fuse unless extremely large and expensive residue codes are used. For instance, the expected coverage for the fuse in the pessimistic scenario is 99.98%, which would require 12-bit residues (x = 4095).

A different scheme that detects failures is DIVA [4], which uses a simple in-order core as a checker for an out-of-order core. Such checker may be a significant hardware overhead if it has to keep pace with the out-of-order core for high ILP programs. On the other hand, although the fuse is presented only for a sparse tree adder, fuses can be used for any ALU in the processor, with negligible hardware and energy overhead, and without introducing any performance loss.

Similarly to DIVA, redundant multithreading (RMT) [16][17] is a solution to detect failures based on reexecution either in the same core (SMT core) or in another core (CMP processor). Again, although RMT is a generic solution to detect errors in any ALU, its performance cost is huge because the performance of a thread is wasted to run the replicated thread, and the original thread observes some performance drop because of the synchronization and resource sharing of both threads. Moreover, if both threads run in the same core, hard errors cannot be detected for non-replicated ALUs (e.g., multipliers, dividers) and may not be detected for replicated ALUs (e.g., adders) if both instances of an instruction are executed in the same ALU. Comparatively, the coverage of the fuse to detect failures is huge, its hardware cost very low and there is no performance loss. Although the fuse does not detect soft errors in ALUs, some other techniques can be used for such purpose [15].

6. Future Work

This paper contributes presenting the fuse as a mechanism to accurately and timely anticipate failures due to NBTI aging in a sparse tree adder. We plan to extend such technique to deal with other ALUs in the processor and other types of adders. Furthermore, blocks such as the decoding logic can be also protected with fuses. So far, there is no technique to anticipate failures in combinational logic where residue codes do not work unless the hardware is replicated.

In this work we have considered degradation due to NBTI. However, we plan to extend fuses to deal with other sources of failure affecting transistors and wires. The concept of the fuse holds the same for wires, and the weakest wire can be replicated and integrated into the ring oscillator between two of the inverters to measure its degradation. In this way, there will be several fuses for each block protecting it from the main sources of failure. Whenever any of these fuses anticipates a failure due to degradation, the protected block is considered unsafe and it can be disabled.

7. Conclusions

Faster device degradation is one of the main challenges that future technology is posing. In the forthcoming process generations, it will be crucial to anticipate failures due to degradation to prevent frequent system crashes and data corruption.

State-of-the-art solutions to detect failures in ALUs detect them a posteriori or are very expensive in terms of performance and/or hardware cost. To deal with these issues we propose a new technique to anticipate failures (*fuse*) in ALUs. The fuse is a small piece of hardware that replicates the weakest transistor in the block and mimics its behavior to anticipate failures due to degradation. Our evaluation shows that even if random WID variations are high, the fuse is very effective. Even in a pessimistic scenario, the fuse only anticipates the failure by 3.4% of the adder lifetime (less than three months for an adder whose lifetime is 7 years), its coverage is very high (99.98%), and area and power overheads are low. Although we propose the fuse for a sparse tree adder, this solution can be extended to deal with any other combinational block such as other adders, multipliers, dividers, decode logic, etc.

8. Acknowledgements

This work has been partially supported by the Spanish Ministry of Education and Science under grant TIN2004-03702. The authors would like to thank James Tschanz and Subhasish Mitra for their feedback in the early stages of this work, and Alex Piñeiro for his thorough reading of the draft of this paper.

References

[1] W. Abadeer, W. Ellis. Behavior of NBTI under AC Dynamic Circuit Conditions. In International Reliability Physics Symposium (IRPS) 2003.

[2] M.A. Alam. A Critical Examination of the Mechanics of Dynamic NBTI for PMOSFETs. In International Electron Devices Meeting (IEDM) 2003.

[3] M.A. Alam, S. Mahapatra. A Comprehensive Model of PMOS NBTI Degradation. In the Journal of Microelectronics Reliability, vol. 45, no. 1, January 2005.

[4] T. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In International Symposium on Microarchitecture (MICRO) 1999.

[5] A. Avizienis. Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design. IEEE Transactions on Computers, vol. 20, no. 11, November 1971.

[6] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, V. De. Parameter Variations and Impact on Circuits and Microarchitecture. In Design Automation Conference (DAC) 2003.

[7] K.A. Bowman. Intel Guidelines for Variations. Personal communication.

[8] K.A. Bowman, S.G. Duvall, J.D. Meindl. Impact of Die-to-Die and Within-Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration. In IEEE Journal of Solid-State Circuits, vol. 37, no. 2, February 2002.

[9] C.L. Chen, M.Y. Hsiao. Error-Correcting Codes for Semiconductor Memory Applications: A State of the Art Review. In Reliable Computer Systems – Design and Evaluation, Digital Press, 2nd edition, 1992.

[10] S.S. Chung, C.H. Yeh, H.J. Feng, C.S. Lai, J.J. Yang, C.C. Chen, Y. Jin, S.C. Chen, M.S. Liang. Impact of STI on the Reliability of Narrow-Width pMOSFETs with Advanced ALD N/O Gate Stack. In IEEE Transactions on Device and Materials Reliability, vol. 6, no. 1, March 2006

[11] R.R. Fritzemeier, H.T. Nagle, C.F. Hawkins. Fundamentals of Testability – A Tutorial. In IEEE Transactions on Industrial Electronics, vol. 36, no. 2, May 1989.

[12] G.G. Langdon, C. K. Tang. Concurrent Error Detection for Group Look-ahead Binary Adders. IBM Journal of Research and Development, vol. 14, no. 5, September 1970.

[13] J.G. Maneatis, M.A. Horowitz. Precise Delay Generation Using Coupled Oscillators. In IEEE Journal of Solid-State Circuits, vol 28, no 12, December 1993.

[14] S. Mathew, M. Anders, B. Bloechel, T. Nguyen, R. Krishnamurthy, S. Borkar. A 4GHz 300mW 64b Integer Execution ALU with Dual Supply Voltages in 90nm CMOS. In International Solid-State Circuits Conference (ISSCC) 2004.

[15] S. Mitra, M. Zhang, S. Waqas, N. Seifert, B. Gill, K.S. Kim. Combinational Logic Soft Error Correction. In Workshop on System Effects of Logic Soft Errors (SELSE) 2006.

[16] S. K. Reinhardt, S. S. Mukherjee. Transient Fault Detection via Simultaneous Multithreading. In International Symposium on Computer Architecture (ISCA) 2000.

[17] E. Rotenberg. AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors. In Dependable Systems and Networks (DSN) 1999.

[18] E. Schuchman, T.N. Vijaykumar. Rescue: A Microarchitecture for Testability and Defect Tolerance. In International Symposium on Computer Architecture (ISCA) 2005.

[19] G. Semeraro, G. Magklis, R. Balasubramonian, D.H. Albonesi, S. Dwarkadas, M.L. Scott. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. In International Symposium on High-Performance Computer Architecture (HPCA) 2002.

[20] Y. Sun, K.L. Pey, C.H. Tung, S. Lombardo, F. Palumbo, L.J. Tang, M.K. Radhakrishnan. Geometry Dependence of Gate Oxide Breakdown Evolution. In International Conference on Physical and Failure Analysis (IPFA) 2004.

[21] X. Vera, O. Unsal, A. González. X-Pipe: An Adaptive Resilient Microarchitecture for Parameter Variations. In Workshop on Architectural Support for Gigascale Integration (ASGI) 2006 (in conjunction with ISCA 2006).

[22] E. Wu, J. Suñe, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, D. Harmon. Interplay of Voltage and Temperature Acceleration of Oxide Breakdown for Ultra-thin Gate Oxides. IBM Journal of Research and Development, vol 46 no 2/3, 2002.

[23] Semiconductors Industry Association. International Technology Roadmap for Semiconductors 2003. http://public.itrs.net/Files/2003ITRS/Home2003.htm

[24] Toshiba Corporation. Semiconductor Reliability Handbook (discrete devices), January 2002. http://www.semicon.toshiba.co.jp/eng/prd/common/data/semi.html