УДК 004.318

DESIGNING FAULTS OF MODERN MICROPROCESSORS

Khafizov T.R.
Scientific supervisor – Associate professor Maksimova N. Y.

Siberian federal university

High-quality verification and testing is a vital step in the design of a successful microprocessor product. Designers must verify the correctness of large complex systems and ensure that manufactured parts work reliably in varied (and occasionally adverse) operating conditions. If successful, users will trust that when the processor is put to a task it will render correct results. If unsuccessful, the design can falter, often resulting in serious repercussions ranging from bad press, to financial damage, to loss of life. There have been a number of high-profile examples of faulty microprocessor designs. Perhaps the most publicized case was the Intel Pentium FDIV bug where an infrequently occurring functional design error caused erroneous results in some floating point divides. Functional design errors are not the only source of problems; the MIPS R10000 microprocessor was recalled due to manufacturing problems that resulted in unreliable operation.

The challenges that must be overcome to build a reliable microprocessor design are great. There are many sources of errors, each requiring careful attention during design, verification, and manufacturing. We broadly classify the faults that can reduce reliability into three categories: design faults, manufacturing faults, and operational faults.

Design faults are the result of human error, either in the design or specification of a system component that renders the part unable to correctly respond to certain inputs. The typical approach used to detect these bugs is simulation-based verification. A model of the processor being designed executes a series of tests and compares the model's results to expected results. Unfortunately, design errors sometimes slip through this testing process due to the immense size of the test space. To minimize the probability of undetected errors, designers employ various techniques to improve the quality of verification including co-simulation, coverage analysis, random test generation, and model-driven test generation.

Another popular technique, *formal verification*, uses equality checking to compare a design under test with the specification of the design. The advantage of this method is that it works at a higher level of abstraction, and thus can be used to check a design without exhaustive simulation. The drawback to this approach is that the design and the instruction set architecture it implements need to be formally specified before the process can be automated.

Complex modern designs have outpaced the capabilities of current verification techniques. For example, a microprocessor with 32 32-bit registers, 8k-byte instruction and data caches, and 300 pins cannot be fully examined with simulation-based testing. The design has a test space with at least 2132396 starting states and up to 2300 transition edges emanating from each state. While formal verification has improved detection of design faults, full formal verification is not possible for complex dynamically scheduled microprocessor designs. To date, the approach has only been demonstrated for in-order issue pipelines or simple out-of-order pipelines with small window sizes. Complete formal verification of complex modern microprocessors with out-of-order issue, speculation, and large instruction windows is currently an intractable problem.

Manufacturing defects arise from a range of processing problems that manifest during fabrication. For example, step coverage problems that occur during the metallization process may cause open circuits, or improper doping in the channel of CMOS transistors may cause a

change in the threshold voltage and timing of a device. *Nonconcurrent testing* techniques, which place the part into a special testing mode, are the primary vehicle for diagnosing these type of errors. Testing of the system is accomplished by adding special test hardware. Scan testing adds a MUX to the inputs of flip-flops that allow for reading and writing of latches during test mode. This method provides direct checking of flip-flop operation and indirect checking of combination logic connected to the scan latch. Using the scan chain, test vectors are loaded into the flip-flops and then combinational logic is exercised to determine if the implementation is faulty. Built in self test (BIST) adds specialized test generation hardware to reduce the time it takes to load latches with test vectors. BIST test generators typically employ modified linear shift feedback register (LSFR) or ROMs to generate key test vectors that can quickly test for internal logic defects such as single-stuck line faults. To obtain sufficient design coverage in current designs, BIST can take as much as 15% of the design area. A more global approach is taken by *IDDQ testing*, which uses onboard current monitoring to detect if there are any short-circuits. During testing, power supply currents are monitored while the system is exercised; any abnormally high current spikes are indicative of short-circuit defects. The advantage of IDDQ is that it is straightforward to test a large area all at once, however, it requires careful regulation to get the correct current limits.

Operational faults are characterized as sensitivity of the chip to environmental conditions. It is useful to subdivide these type of errors into categories based on their frequency: permanent, intermittent, and transient faults. *Permanent faults* occur consistently because the chip has experienced an internal failure. Electro metal migration and hot electrons are two examples of permanent faults that can render a design irrevocably damaged. We also classify latch-up, which is caused by a unity gain in the bipolar transistor structures present in a CMOS layout, as a permanent fault; however, this fault can be cleared by powering down the system. Unlike permanent faults, *intermittent faults* do not appear continuously. They appear and disappear, but their manifestation is highly correlated with stressful operating conditions. Examples of this type of fault include power supply voltage noise or timing faults due to inadequate cooling.

Data-dependent design errors also fall into this category. These implementation errors are perhaps the most difficult to find because they require specific directed testing to locate.

Transient faults appear sporadically but cannot be easily correlated to any specific operating condition. The primary sources of these faults are single event radiation (SER) upsets. SER faults are the result of energized particle strikes on logic which can deposit or remove sufficient charge to temporarily turn the device on or off, possibly creating a logic error. While shielding is possible, its physical construction and cost make it an unfeasible solution at this time. *Concurrent testing* techniques are usually required to detect operational faults, since their appearance is not predictable. Three of the most popular methods are timers, coding techniques and multiple executions. Timers guarantee that a processor is making forward progress, by signaling an interrupt if the timer expires. Coding techniques, such as parity or ECC, use extra information to detect faults in data. While primarily used to protect storage, coding techniques also exist for logic. Finally, data can be checked by using a k-ary system where extra hardware or redundant execution is used to provide a value for comparison. Deep submicron fabrication technologies (i.e., process technologies with minimum feature sizes below 0.25um) heighten the importance of operational fault detection. Finer feature sizes result in smaller devices with less charge, increasing their exposure to noise-related faults and SER. If designers cannot meet these new reliability challenges, they may not be able to enjoy the cost and speed advantages of these denser technologies.

Many reliability challenges confront modern microprocessor designs. Functional design errors and electrical faults can impair the function of a part, rendering it useless. While functional and electrical verification can find most of the design errors, there are many

examples of non-trivial bugs that find their way into the field. Additional faults due to manufacturing defects and operation faults such as energetic particle strikes must also be overcome. Concerns for reliability grow in deep submicron fabrication technologies due to increased design complexity, additional noise-related failure mechanisms, and increased exposure to natural radiation sources.

To counter these reliability challenges, scientists proposed the use of dynamic verification, a technique that adds a checker processor to the retirement phase of a processor pipeline. If an incorrect instruction is delivered by the core processor the checker processor will fix the errant computation and restart the core processor using the processor's speculation recovery mechanism. Dynamic verification focuses the verification effort into the checker processor, whose simple and flexible design lends itself to high-quality functional verification and a robust implementation.

Designers presented analyses of our prototype physical checker design. Timing analyses indicate a fully synthesized and unpipelined 4-wide checker processor design in 0.25um technology is capable of running at 288 MHz They are currently hand optimizing this design through better layout and stage pipelining. Designers fully expect their follow on efforts will demonstrate that the checker design is also quite scalable. In addition, area and power analyses of their physical design were presented. Overall, the checker processor requires less than 6% the area and 1.5% the power of an Alpha 21264, confirming that this approach is low cost. Finally, there were presented novel extensions to baseline design that improve coverage for operational faults and manufacturing fault detection.

Designers feel that these results strengthen the case that dynamic verification holds significant promise as a means to address the cost and quality of verification for future microprocessors. Currently, they continue to refine their physical checker processor design, optimizing its layout and investigating technique to scale its performance through pipelining. In parallel with this effort, they are also examining how they might further leverage the fault tolerance of the core processor to improve core processor performance and reduce its cost. One such approach is to leverage the fault tolerance of the core to implement selftuning core circuitry. By employing an adaptive clocking mechanism, it becomes possible to overclock core circuitry, reclaiming design and environmental margins that nearly always exist.