iRazor: a Low Overhead Error Detection and Correction Scheme to Improve Processor Performance

Ricky Liou
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94709
Email: rliou92@berkeley.edu

Matthew Beach
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94709
Email: mathewbeach@berkeley.edu

Abstract—Technology scaling has increased the impact of process variations on the design margins of digital circuits, leading to circuits operating with higher percentages of safety margins. Current technology uses canary circuits, also known as replica circuits, or look up circuits in order to reduce the amount of margin at which the circuit operates, but these approaches still must account for worst case operating conditions and operate conservatively. iRazor is an error detection and correction (EDAC) scheme that allows the circuits to operate at points of first failure or even beyond. iRazor features a flip-flop that only adds three additional transistors to a conventional flip-flop and has much lower area and performance overheads than previous EDAC schemes. The iRazor flip flops will be implemented on the critical path of a three stage RISC-V processor using the 32/28 nm 241B process technology. Results will be compared against a RISC-V processor with no EDAC by comparing their energy efficiency and performance.

I. INTRODUCTION

As transistors scale to smaller dimensions, design uncertainties become more pronounced. Local temperature variations, supply voltage noise, inter and intra die process variations, etc. all contribute to operating a circuit at more conservative voltages and frequencies. However, as the transistors become smaller, operating at such points results in an increasingly larger percentage of performance loss and energy efficiency loss. Current methods of reducing design margins use adaptive online techniques that reduce the margins by adjusting the operating point.

The most popular implemented adaptive techniques include both look-up table circuits and canary circuits. The look-up table method operates the circuit at multiple known voltage and frequency points at which the circuit is known to function correctly. Based on the current conditions the operating point of the circuit can be adjusted. For example if the circuit does not require high performance, the voltage or frequency may be lowered. The main limitation of this method is that each operating point must also be sufficiently margined to ensure correct circuit operation.

Canary circuits allow reduced design margins by incorporating a replica circuit of the critical path in the die. The replica circuit is then able to track any variations that the critical path experiences. In doing so, the optimal operating point of the critical path circuit can be estimated in real time. The drawback of replica circuits is that they do not exactly mimic the critical paths. The replica circuits are located in different parts of the die and are thus unable to track any local fluctuations at the critical paths such as local temperature swings or supply voltage disturbances. Another disadvantage is that this method cannot respond quickly to short term transients that the critical path experiences.

An alternative to online methods is detecting errors and correcting them when they occur. In doing so, design margins can be completely eliminated, leading to large performance and energy efficiency gains. Traditional methods of error detection and correction involve large area overheads and additional circuit complexity that offset those gains, making them unattractive to implement. For those reasons EDAC methods have not been widely implemented on a large scale.

To improve on the current state of the art EDAC methods, we will be implementing the iRazor EDAC as shown in Figure 1. The iRazor error detection is achieved through a modified flip flop that only adds three transistors compared to a standard flip flop. The flip flop detects errors when the CTL transistor is off. In this error detection phase the virtual VVSS is floating, and any data transition will cause the voltage at VVSS to rise. If D experiences a data transition, there will be a brief short circuit current through the first tri state buffer A and feedback inverter B that pulls VVSS high. A skewed inverter monitors the VVSS and outputs an active low error signal.

Error correction in the iRazor is achieved by careful control of the CTL and local clock. The iRazor timing control block outputs a global clock that can be disabled if an error is detected. The CTL and local clock are then locally generated. For simplicity the control does not feature any pipeline rollback mechanism when an error occurs. Instead, the next clock cycle is deleted, and this gives the data enough time to propagate through. There is a fundamental tradeoff between the length of the detection window and the number of false positive errors. CTL must always have a falling edge after the rising edge of the clock in order to allow valid transitions of the data to
pass through. If the delay between the falling edge of CTL and rising edge of LCLK is small, there will be a larger chance of detecting a false positive, but the error detection period is longer. The error signals of all registers are propagated through a dynamic OR stage and the iRazor control block in the same clock cycle in which the error occurs. This presents a tight timing constraint to those logic.

II. CONTRIBUTION

In order to verify the effectiveness of iRazor, we will model it in a three-stage pipelined RISC-V processor and evaluate the new processors performance. We will replace the flip flops in the critical path with iRazor ones and add the associated timing logic. The performance will be measured from the maximum instructions per second (IPS) that can be achieved with the new processor.

\[ IPS = f_{clk} \times IPC \]

Instructions per cycle (IPC) will be defined as the number of instructions executed by the processor per the input clock. Both processors will run the same matrix multiplication program to ensure a valid comparison. The supply voltage will be held constant for each test, and the test will be run at different voltages. For the processor with EDAC, maximum instructions per second is achieved by increasing clock frequency, even past the point of first failure. When an error occurs, the instructions per cycle will be decreased. Thus, there will be an optimum point where the benefits of increasing the clock frequency will be offset by the decreased instructions per cycle.

Energy efficiency will be measured in instructions per second per watt, or instructions per joule. The power consumption will be measured for multiple supply voltages while running at the optimum frequency to maximize IPS for that supply voltage. In the case of the baseline processor, the optimum frequency will just be the maximum frequency at which the processor still functions. For the EDAC processor, the optimum frequency will be found from the performance metric test. To achieve higher energy efficiency, the EDAC processor is able to tolerate lower supply voltages while operating at higher frequencies in order to have the same instructions per second. Operating at lower supply voltages will increase the error rate and thus reduce the instructions per cycle. Power consumption is found as:

\[ P = \alpha \times f_{clk} \times C \times V^2 \]

Because power is proportional to the voltage squared, it is beneficial power-wise to operate at reduced voltage at the cost of increased frequency.

III. EXPERIMENTAL SETUP

First, we will design and implement a single iRazor flip-flop in a transistor-level schematic using Cadence Virtuoso. We will then verify the functionality of the flip-flop by constructing a test bench. The original functionality of a flip-flop will be tested and verified initially. Next we will present to the flip-flop errors where the data arrives too late, violating the flip-flop’s set up time. We will sweep the data arrival time over a large range with respect to the clock while varying the error detection window. We will use the resulting ERR signals to optimize the design of the flip flop by trading off the number of false positives with the length of the detection window. Once the functionality of the flip-flop has been optimized through simulation, we will use Virtuoso to create a physical layout entry and timing characterization for the iRazor. The layout must pass a design rule check (DRC) and layout vs schematic check (LVS) using the Synopsys Hercules tool before we will be able to integrate the iRazor into a processor. Once the layout of the flip-flop is complete, a post layout SPICE netlist with parasitic resistance and capacitance will be extracted using Synopsys StarRCXT. The parasitic netlist will then be used to do final simulations and optimizations of the iRazor flip-flop.
The iRazor control block must be carefully tuned so that it can suppress the next clock pulse in the same clock cycle in which an error was detected. In order to ensure that the timing constraint will be met across all operating frequencies that we plan to test, we will design the control block to be able to operate at the highest clock frequency. We will write out the necessary logic functions in Verilog. After testing the correct operation of the behavioral code, we will synthesize, place and route, and import the control block into Virtuoso. In Virtuoso, we will do transistor level simulations to verify that the timing of the block will meet the timing constraint. At first, we will try to complete the control block using static logic. However, if static logic is not fast enough, we will switch to a faster architecture such as dynamic logic.

With the design of the iRazor complete, it will be implemented in a three stage RISC-V processor in order to quantify the performance benefits. First, a RISC-V processor will be pushed through a place-and-route flow, and its critical paths will be identified. The functionality of the processor will be verified by running a matrix multiplication C program. The time it takes to execute the C program will be measured as the performance benchmark figure of merit. Once the control processor has been established, all flip-flops on the critical path will be replaced by iRazor flip-flops in the Verilog source code. The new Verilog source code will also need to include the OR propagation of the error signals to the iRazor timing control as well as the timing control logic itself. Once the iRazor flip-flops and the supporting control circuitry have been implemented, the iRazor processor will be pushed through the same place-and-route flow as the control processor. The modified processor will then be subject to the same matrix multiplication C program.

IV. RESULTS

The processor with EDAC is expected to have a higher instructions per second than its counterpart at the same operating voltage. The processor with EDAC will be able to operate at a higher clock frequency. At the point of first failure, the failure rate is still expected to be low, meaning that there is an overall net gain by increasing the clock frequency to that point. If the clock frequency is increased too high the clock frequency might reach the point where other paths in the processor that do not have error correction become the critical path. If this is the case, then we will replace those flip-flops with error detecting ones as well.

V. CONCLUSION

In this paper, we will demonstrate that the implementation of iRazor EDAC flip-flops helped increase the performance of a RISC-V processor while adding minimal addition overhead when compared to a processor with no EDAC scheme. Implementation of the iRazor scheme is expected to result in performance gains of 30% over no EDAC scheme. To achieve these performance gains, overhead is expected to increase by 13.6% over no EDAC scheme. Future research includes optimizing the iRazor control scheme to increase the maximum supported amount of iRazor flip-flops and applying it to other clocked digital circuitry.

REFERENCES


