VLSI Design of CRC-Based Fingerprinting on MIPS8 Architecture

Georgi Kostadinov, Xinchi Chen, Kaushik Boga, Mojing Liu
Department of Electrical and Computer Engineering
McGill University

Abstract—Traditional error mitigation techniques such as Error Correcting Code (ECC) and Dual Modular Redundancy (DMR) in Lockstep provide error detection at great cost of power, area and performance. In this paper, we present the implementation and verification of an Execution Fingerprinting Unit using CRC16 operating on a MIPS8 architecture. Our design provides a 255 times decrease in comparison overhead compared to DMR Lockstep and 300MHz max operating frequency unpipelined.

1 Introduction

TRADITIONAL error mitigation techniques such as Error Correcting Code (ECC) provide limited error coverage rate at the cost of performance and area overhead. Another solution widely used in industry, Dual Modular Redundancy (DMR) in Lockstep, consists of two redundant cores executing the same application in lockstep and validating the results only if the comparison passes. However large amount of computational ressource is wasted for comparison as not to mention the overwhelming complexcity of clock synchronization due to lockstep.

2 Background

Fingerprinting first uses Distributed Temporal Redundancy, which allows processors to execute out of sync[1], thus breaking the lockstep. This will create much more freedom in terms of schedulabilty, letting us apply a second technique called Relaxed Dedication which allows the processor to also execute non critical tasks[2]. As the processors are out of sync,however the execution data still needs to be verified through redundant exeuction data. Directly saving the redundant data would not be ressource efficient, hence fingerprinting compresses this block of data into a single word called fingerprint. The compression algorithm can be chosen by the designer and will have different impact in terms of error coverage and detection.

3 Implementation

Fingerprinting can be implemented in three steps. First, the execution data of the processor needs to be extracted and fed into the fingerprinting circuit. Second, this data is compress using CRC16 to generate the fingerprints, which will be explained in details in this section. At last, fingeprints needs to be stored for later comparison for error detection. The implementation of fingerpriting was based on the MIPS8 architecture and a standard cell library provided by Harvey Mudd College[3] using Electric VLSI Design System[4]. The goal of this work is to prove the feasibility of using fingerprinting for error detection Therefore error coverage analysis is not in the scope of this work.

3.1 Execution Information Extraction

Only memory write address and data were originally available through the MIPS8 architecture. Fingerprinting additional execution has been proven to improve error dectection coverage and latency[**?**]. Therefore we added to the export list the results of the ALU within the datapath. However, only ALU results that are written to the register file are needed and therefore a control signal composed of *memtoreg* and *regwrite* were used to signal a useful ALU result to be fingerprinted. As the MIPS8 core within this project cannot ouput a valid ALU result and memory write data, hence an additional multiplexer was added in order to help choosing the right set of signals for the fingerprinting circuit.

3.2 Compressing the Data: The CRC Fingerprint

The Cyclic Redundancy Check algorithm can be used to compress data for later verification. CRC is a good choice for fingerprinting, because it is a simple and widely used algorithm that was designed with error detection in mind. Although CRC is more ideal for transmission channel error detection, as it can guarantee resilience to burst errors to within a given number of corrupt bits, it can still offer strong error detection for randomly distributed error when compared to other techniques such as Fletcher's Checksum.

A 16-bit wide CRC was chosen, as it offers much stronger error detection characteristics than a lower bit alternative, but is not too large to be considered overkill for a MIPS-8 core. The 0x1755b polynomial was used, since it is a good choice for larger block sizes[**?**]. A set of

logical equations for each output bit were found[5] and implemented using combinational logic. The produced combinational circuit was linked to a register to store the CRC result after each iteration, as its value is required for the next calculation.

3.3 Storing the Fingerprint: Shift Register vs SRAM

The CRC block generates a fingerprint every 256 memory/ALU operations executed by the MIPS processor. To avoid lockstep execution with a parallel processor, a limited set of these fingerprints are stored in a buffer and read out later for comparison, to detect whether the 256 batch of operations were executed correctly. A buffer size to store 8 16-bit fingerprints was chosen. Since the goal of the project was not memory design, rather to demonstrate the efficacies of fingerprinting, a generic flipflop based shift register circuit was quickly implemented. The generic flipflop takes ph1, ph2, enable and reset as inputs and as a result needed 30 transistors per bit. Though the layout was very modular using the standard cell library components, and hence implementation quick, this resulted in a significantly big buffer (needing 3872 transistors) relative to the size of the fingerprinting circuit. This will skew the energy or performance savings primarily proposed. The shift register also suffers from the hold time constraint with a really short contamination delay between two flipflops. So, this implementation would have needed a thorough timing analysis to ensure timing wont fail and is a poor choice of design for a buffer. Hence the shift register was replaced by a FIFO built with SRAM during our optimization phase. The FIFO features a single read/write per cycle composed of 6T SRAM cells, 2 3-bit counters, row decoder, bitline conditioning and read/write driver logic. Unlike the shift register, the FIFO moves a pointer to the data, instead of moving all the data physically. Two 3-bit counters are used to keep track of read and write pointers to the rows of bits. 3 2:1 multiplexers choose between the two pointers to drive the row decoder depending on whether data is to be read from or written to the SRAM. The row decoder composed of a 2 stage design; 8 3-in NANDs and 8 2-in NORs to drive the word lines. NOR stage makes the row decoder output ph1 qualified to drive the word lines only when ph1 is high. The 16 bitline conditioning blocks sitting on top of the SRAM slices, composed of 2 pmos transistors clocked with ph2b (dynamic logic), precharge the bitlines when ph1 is low, ready for the read/write operation. The read driver is a pair of high skew inverters which detect the voltage on the bitlines and drives the output to the rails or ground. The write driver consists of a series pair of nmos transistors on each bitline. The nmos transistors pull the bit line to ground if writeq1 and datas1 are both high and the bitb line behaves conversely. This implementation needed 1374 transistors laid on a 45 lamda pitch and did not involve moving data around every clock cycle. So, the area and power consumption is much smaller and hence realistic, allowing a more compelling case to be made for the fingerprinting circuit.

3.4 The Final Design

The fingerprinting system as a whole includes a multiplexer, a counter, and additional glue logic to integrate the CRC circuit into a fully functional design. The MIPS core serves as the external controller for the fingerprinting system, issuing the appropriate signals whenever new data is ready to be fingerprinted. A multiplexer performs the selection of data to be fingerprinted based on the CPUs signals. A counter is used to count each time new data comes in and it determines the data compression ratio of the fingerprint. When the desired amount of data has been fingerprinted, the counter signals the output buffer block to store the new fingerprint, and it also signals the CRC block to start computation for a new block. The CRC block needs to be able to handle data coming in at the same time as the counter's control signal, so additional logic was required to handle such a scenario.

Wiring proved to be a challenge when implementing the layout. If the design was to be repeated, more attention would be placed on wire planning and pitch matching than on logic minimization, as a decrease in functional blocks or transistor count may not necessarily lead to improvements in area, and pitch matching can allow for menial tasks such as wire routing to be left to automation tools rather than the designer. Long and complex wires also decrease the modularity of a design, introduce additional parasitic resistances and capacitances, and make the system more vulnerable to noise, so it is generally a good idea to try and minimize them.

4 Experimental Setup

4.1 Schematic Verification

Benchmarks were written in system verilog and simulated using ModelSim for functional verification. Each major circuit component was first simulated on the schematic level before implementing the layout. This not only aids in finding and fixing bugs, but also simplifies the component integration process. Random test vectors were generated to verify timing sequence and operation of complex circuits. For the final circuit, the modified mips core was hooked up to the fingerprinting circuit for functional simulation. A benchmark was written in MIPS assembly to run on the mips core, and the produced waveform was studied to ensure proper circuit operation.

4.2 Layout Verification

4.3 Timing Analysis

4.3.1 SRAM timing

A 6T SRAM cell functions due to the 6 specially sized transistors that enable read and write stability. Since

Modelsim is a functional/logic simulator which doesnt consider transistor sizes, fifo cannot be tested in Modelsim with the rest of the circuit. Ideally, an SRAM cell model must be built which mimics the SRAM behaviour using logic. Since this would be time consuming, the entire circuit was instead tested with the earlier shift register implementation for functional verification. The fifo was validated in a timing analyser tool (IRSIM) which uses the linear delay model to approximate switching delay.

The functionality was validated by observing internal signals in the FIFO block to verify each component behaved as intended with the right set of inputs. Once verified individually, two sets of tests were performed. The first test checked whether the FIFO can write 8 fingerprints in a row sequentially and read them out sequentially after some arbitrary time. The other test alternates write followed by a subsequent read. This test is more realistic considering fingerprints are generated one every few hundred/ thousand clock cycles. Once validated, the SRAM was connected to the entire circuit and a timing simulation was performed to evaluate the delay. This exercise exposed the limitations of Modelsim for testing and allowed the team to appreciate the importance of timing analysis for VLSI designs.

5 Results

5.1 Area Overhead and Scaling

	Shift register	FIFO
Transistor count	3872	1374
Area(lamda squared)	2328 X 1000	1200 X 730

The SRAM based design needs 3 times fewer transistors compared to the FF based design, consequently needing 3 times less area. The area could have been smaller if the row decoder was built using a 45 lamda pitch, if the 3-bit counters were built using latches rather than the generic FFs and if the white space could have been eliminated by using a less modular design. The SRAM block and consequently the read and write drivers could have been tighter if there was an additional layer to route vdd and gnd wires.

The fingerprinting circuit can be tuned for various modes of operation, and some decisions have an impact on area and cost. The counter determines the compression ratio, so increasing the counter's modulus will decrease the area required to store the fingerprints for the same amount of CPU execution information. Another important factor is the choice of execution data to fingerprint. At a minimum, the fingerprinting circuit needs to verify memory writes and adresses for data corruptions. However, if aditional data, such as ALU operations, are also fingerprinted, the amount of execution data will increase, and so will the area required to store all fingerprints. The impact of fingerprinting additional data would vary depending on the target application, but for the MIPS benchmark that we tested, the area required for storage gets tripled if ALU operations are fingerprinted in addition to memory write operations.

5.2 Operating Frequency

As this is a transistor-level design, besides functional simulation, a timing performance simulation may also be required in order to have a more precise system performance prediction. Irsim switch-level digital circuit simulator, and it will use extracted capacitance and lumped resistance values to produce a timing analysis simulation for a certain design.

In order to perform the Irsim simulation, a Irsim deck, a prm file and a testbench file are needed. The irsim deck, which contains all the connection information of the transistors in the schematic(or layout) design, can be automatically generated from a schematic design in Electric. The prm file contains all the resistance and capacitance information of transistors, and the testbench can be written based on Irsim test-language-grammer.

Here is a sample C++ based irsim testbench generator, which is used to test the multiplexer.

The components of the finger-printing design, including the multiplexer, the fingerprinting code generator, and the counter, are tested individually in the first place. The delay of each component can be found by observing the time difference between a rising clock edge and its next output value transition. As there are some slight variance for the delay at different clock edges, an average delay is calculated by observing multiple clock rising edges. Here are the average delay of each component.

TABLE 1
Delay of Each Components

Mux	Counter	Fingerprint	Shift Register
0.146ns	0.967ns	3.812ns	0.699ns

Next, a slowest bit analysis is performed on the CRC code generator. As all the 16 bits in fingerprinting block are processed in parallel, each bit is tested individually for the range of delay data. From the test, it is shown that the longest delay is 3.4ns, while the shortest is 2.27. And the difference is 1.13ns.

After that, the whole design is tested. From the test, it is shown that a reset signal from the counter will always slows down the CRC code output. In order to find the worst case scenario of the system delay, only clock edge with a reset signal are analysis. In order to ensure the worst case scenario is reliable, the difference from slowest bit is added to the worst case delay observed from the collected data.

In clonsion, the whole system delay is calculated to be 8.03ns. Based on this result, the maximum system operating frequency is estimated to be 120 Mhz.

6 CONCLUSION

One of the key observations from this project is the importance of understanding the limitations of tools used. For example, we learnt after the project that NCC may not be reliable for large designs. So, while SRAM was tested for timing using the schematic description, the layout though passing NCC, may be incorrect. Hence, if more time was available, the SRAM fifo layout would have also been thoroughly tested for any inconsistencies. Further, the fifo could have been optimized further, for example by performing a path effort analysis for the row decoder. This would have shortened the time required to charge the capacitance of all the 16 SRAM cells and allowed for a higher operational frequency. The project would have benefited with more time evaluating the circuit in IRSIM because a timing analysis exposes various vulnerabilities of the design that Modelsim cannot check for.

REFERENCES

[1] Brett H. Meyer, Benton Calhoun, John Lach, Kevin Skadron, *Cost-effective Safety and Fault Localization using Distributed Temporal Redundancy*,CASES'11, October 2011.

[2] Brett H. Meyer, Nishant George, Benton Calhoun, John Lach, Kevin Skadron, *Reducing the Cost of Redundant Execution in Safety-Critical Systems using Relaxed Dedication*, DATE'11, March 2011.

[3] Havery Mudd College *E158 Spring 2007 MIPS Project* [Online]. Available: http://www4.hmc.edu:8001/Engineering/158/07/project/index.html

[4] Static Free Software, *Electric* [Online]. Available: http://www.staticfreesoft.com/electric.html
P. Koopman and T. Chakravarty, *Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks*, Proceedings of the 2004 International Conference on Dependable Systems and Networks, 2004.

[5] E. Stavinov, *A Practical Parallel CRC Generation Method*, Feature Article, pp. 38-45, Jan. 2010.