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 TMR, which allows processors to execute out of sync, thus breaking the lockstep. This will create much more freedom in terms of schedulabilty, letting us apply a second technique called RD which allows the processor to also execute non critical tasks. Both techniques come from papers that prof Meyer has published and I invite you all to have a look. So back to fingerprinting, as the processors are out of sync, in order to compare the execution data, we need to save it somewhere. However, directly saving this data would take a look of space, hence we compress it 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

3.1 Execution Information Extraction

The data that we decided to fingerprint is memory write address and data as well as register data updates. We had to add additional exports to the original chip, which turned out to be a perfectl fit using all the pins. Thus we were able to maintain the original MIPS8 packaging, eliminating the cost of a new package,making use of the 9 remaining pins that were available. Eight of them are for the 8 bit register data and one is a control signal that tells us if a register was written.

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[2]. A set of logical equations for each output bit were found[3] 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

Every 1000 or so cycles, the CRC circuit generates a fingerprint. To avoid lockstep execution with a parallel processor, a limited set of these fingerprints can be stored in a buffer and read out later for comparison. A buffer size to store 8 16-bit fingerprints was chosen. Since the goal of the project was not memory design, a generic flipflop based shift register circuit was quickly implemented. The generic flipflop takes 2 clocks, en and reset as inputs and as a result needed 30 transistors per bit. This resulted in a significantly big buffer (needing 3872 transistors) relative to the size of the fingerprinting circuit and hence will skew the energy or performance savings primarily proposed. Hence the shift register was replaced with a FIFO built with SRAM.

The FIFO features a single read/write per cycle composed of 6T SRAM cells, 2 3-bit counters for pointers to memory, row decoder to drive specific word lines, bitline conditioning and read/write driver logic to operate the SRAM cells. This implementation needed 1374 transistors laid on a 45 lamda pitch and doesnt move data around every clock cycle. So, the area and power consumption is much smaller, allowing for a more compelling case 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.

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.

The fifo was validated in a timing analyser tool (IRSIM) which uses the linear delay model to approximate switching delay. Once deemed functional, it is connected to the rest of the circuit and simulated together in IRSIM. 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

5.2 Operating Frequency

6 Evaluation

One of the key observations from this project is to understand 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 been tested for any inconsistencies. Further, the fifo could have been optimized further, performing a path effort analysis for a faster design with lower power consumption.

7 Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] 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.

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