HW1_ Return Address Predictor Design

郭宗信, 111550105

***Abstract*—This report analyzes and designs a branch predictor for the Aquila system to improve CoreMark performance. We disabled the Branch Prediction Unit (BPU) or adjusted the Branch History Table (BHT) size to study their impact. Branch statistics, including hit and miss rates, were analyzed. Finally, a return address predictor was designed to enhance prediction accuracy and reduce mispredictions. The report is limited to two pages and outlines key design strategies and results.**

***Keywords*—Profiling, FPGA, Aquila core, CoreMark, real-time analysis, gprof, hardware profiling.**

I. Introduction

In modern processors, branch prediction plays a vital role in maintaining pipeline efficiency, particularly in systems like Aquila. The accurate prediction of branch targets, including return addresses, can significantly impact performance benchmarks such as CoreMark. This report focuses on Part 1, where we analyze the existing branch predictor in Aquila, exploring how disabling the Branch Prediction Unit (BPU) or adjusting the Branch History Table (BHT) size influences CoreMark performance. Additionally, we investigate branch statistics, including hit and miss rates, and propose a return address predictor architecture to enhance prediction accuracy and minimize pipeline flushes caused by mispredictions.

II. Methodologies

A. *Profiler in verilog*

A hardware profiler is implemented in Verilog within the Aquila SoC to track clock cycles for specific CoreMark functions. Profiling counters monitor the program counter (PC) during function execution, counting the cycles spent in each function. The captured data is read out using the Xilinx Integrated Logic Analyzer (ILA), allowing precise performance analysis of functions at the hardware level.

III. Implementation

The distribution of instruction types (JAL, JALR, load, store) gives us a clear view of the workload characteristics of CoreMark. Understanding the proportion of branch instructions versus regular instructions is crucial for fine-tuning the branch prediction mechanism.

IV. Discussion

The current implementation of the `profiler` provides essential insights into the processor's performance under different branch prediction configurations. Key findings include:

A. ***Branch Prediction Accuracy****:*

By analyzing the `branch_mispredictions` counter, we can evaluate the accuracy of the BPU and determine how mispredictions affect the overall pipeline. This data is critical for optimizing the BPU and improving CoreMark scores.

B. ***Impact of PC Range****:*

By focusing the profiling on the specific PC range relevant to CoreMark, we ensure that the performance evaluation is targeted and provides meaningful data about the benchmark's behavior.

C. ***Instruction Distribution****:*

The distribution of instruction types (JAL, JALR, load, store) gives us a clear view of the workload characteristics of CoreMark. Understanding the proportion of branch instructions versus regular instructions is crucial for fine-tuning the branch prediction mechanism.