VHDL IMPLEMENTATION OF PIPELINED DLX MICROPROCESSOR

IGNATIUS EDMOND ANTHONY

UNIVERSITI TEKNOLOGI MALAYSIA

ABSTRACT

The 32-bit load/store DLX processor architecture is a generic RISC processor designed by Hennessy and Patterson for pedagogical purposes. The DLX processor design abstracts many features of general-purpose commercial processors, and is a well-understood computer architecture, providing a good architectural model for study, not only because of the popularity of this type of machine, but also because it is easy to understand. Utilizing open source hardware such as the DLX core yields the apparent advantage of free-for-all distribution as well as having source codes that are is available and open, allowing for source code modification at-will. This project aims to continue previous work on integration of the DLX core by adding instruction pipelining which was excluded from the previous project's scope due to complexity and time limitations. Instruction execution speedup and performance was left on the table to be dealt with in future work. Since the DLX microprocessor was, by nature, a 5-stage pipelined microprocessor, it can be expected that the core's performance on instruction execution can be sped up with a pipeline implementation. Comparison between the non-pipelined and pipelined DLX were also performed to verify this instruction execution speedup expectation.

ABSTRAK

Senibina pemproses DLX merupakan suatu pemproses generik RISC 32-bit yang direkacipta oleh Hennnesy and Patterson bagi tujuan peyelidikan and pendidikan. Senibina pemproses DLX merangkumi pelbagai ciri-ciri and fungsi pemproses umum di pasaran, dan bukan sahaja merupakan senibina komputer yang mudah difahami, tetapi juga amat popular. Menggunakan pemproses sumber terbuka atau *open-core* seperti mesin DLX ini memberi kelebihan dalam tersedianya kod-kod sumber secara terbuka yang membenarkan dan memudahkan pengubahsuaian untuk keperluan projek. Matlamat projek in adalah untuk meneruskan projek sebelumnya di mana pemproses DLX dan *Wishbone interface* diintegrasikan, tetapi dengan menambah fungsi *pipelining* untuk pemprosesan suruhan. Dengan penambahan ciri ini, adalah dijangka bahawa tempoh pemprosesan suruhan dapat disingkatkan memandangkan cara pemprosesan suruhan dalam mesin DLX dilakukan dalam lima peringkat. Perbandingan prestasi pemproses DLX sebelum and selepas implementasi ciri *pipelining* turut dilaksanakan dalam projek ini untuk mengesahkan jangkaan awal.

TABLE OF CONTENTS

CHAPTER 1

PROJECT OVERVIEW

1.1 Background

The 32-bit load/store DLX processor architecture is a generic RISC processor designed by Hennessy & Patterson for pedagogical purposes. The DLX processor design abstracts many features of general-purpose commercial processors, and is a well-understood computer architecture.

The DLX provides a good architectural model for study, not only because of the popularity of this type of machine, but also because it is easy to understand. Like most load/store machines, the DLX emphasizes a simple load/store instruction set, design for pipelining efficiency, an easily decoded instruction set and efficiency as a compiler target.

The Wishbone Bus is an open source hardware computer bus intended to let the parts of an integrated circuit communicate with each other. The aim is to allow the connection of differing cores to each other inside of a chip or system-on-chip.

Utilizing open source hardware such as the DLX core and Wishbone bus (or its competitor – the AMBA bus) yields benefits which include solutions for most of the problems associated with proprietary cores. Besides the apparent advantage of free-for-all distribution, utilizing open source hardware standards and open microprocessor cores tout:

- Each core will have a larger user base, which will ensure better support, better documentation and better implementation examples to work from.
- The source is available, so any developer can find out what he or she needs to know about the core.
- Eventually, as cores and standards for them are developed, cores will become more standards-compliant than proprietary cores
- Allows for source code modification at-will, which enables designers to fine-tune and tweak any design for any design constraint – gate count, performance, power, etc.

While previous work on integration of the DLX core and Wishbone bus interface has been undertaken and completed, instruction pipelining was excluded from the project's scope due to complexity and time limitations. A DLX microprocessor core with a non-pipelined instruction execution data path was used to showcase the microprocessor core-Wishbone bus integration functionality on the FPGA.

Since the previous project's focus was on functionality, instruction execution speedup and performance was left on the table to be dealt with in future work. Since the DLX microprocessor was, by nature, a five-stage pipelined microprocessor, it can be expected that the core's performance on instruction execution can be sped up with a pipeline implementation.

1.2 Objectives

The objective of this project is to design a five-stage pipelined DLX microprocessor for the purpose of instruction execution speedup and performance improvement.

As part of the project, the performance of instruction execution speedup of the enhanced DLX processor with pipelined-instruction execution versus the non-pipelined DLX will be evaluated and analyzed using a predetermined test suite which will consist of several small programs.

Finally, exploration and investigation will be done on several design-for-test (DFT) feature integrations into the DLX-Wishbone system for improved system debug-ability as future work.

1.3 Scope of Work

This project is focused on the incremental enhancement of an existing DLX processor design with Wishbone bus interface integration in UTM [2], to modify the data path unit and controller unit for pipelined instruction execution in five stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM) and Write-Back (WB).

The reference source code for this project is referenced from the DLX project by the University of Stuttgart, Germany which implements a non-pipelined, non-synthesizeable flavour of the DLX processor.

Implementation of the DLX enhancements would involve coding in hardware description language VHDL. Altera's Quartus 6.1 Web Edition is the tool of choice for design entry, logic synthesis (compilation) and simulation.

In the work flow of the project, the main emphasis would be on adding the pipelining capability into the DLX processor's datapath unit as well as any incremental changes in the control unit to support the pipelined execution. The Wishbone bus interface integration into the design will be used as-is from the previous project with the expectation that any previous issues have been resolved.

While the eventual target implementation of the design would be in ASIC, this project's implementation level will only be restricted to functional and timing simulation within Altera's Quartus software.

As a final outcome, the implementation is validated and verified for functionality correctness on through simulation. Performance analysis and evaluation would be carried out using a predetermined suite of small programs to be executed on the integrated DLX system.

1.4 Expected Results

The pipelined DLX processor is successfully designed and simulated in Altera's Quartus software. The processor implementation is validated and verified using FFT computation task or other simple sorting programs.

The enhanced pipelined-DLX will offer at least 1.5X instruction execution speedup versus non-pipelined core measured by CPU cycle time required to complete predefined computation task-list on both processors. This will serve as the baseline expectation to justify that the additional logic overhead incurred due to the pipelined-datapath translates to real-world performance speedup.

1.5 Report Layout

The layout of this report would be as follows:-

Chapter 1:	Brief overview of project, including objectives and work scope.
Chapter 2:	Literature review of other existing DLX projects undertaken by other universities, design considerations for pipelining, and microprocessor selection considerations.
Chapter 3:	Overview of the DLX processor and instruction set architecture, pipelining concepts and design considerations.
Chapter 4:	Design workflow, methodology and tools.
Chapter 5:	Pipelined DLX microprocessor components and design.
Chapter 6:	Results and performance analysis.
Chapter 7:	Conclusion and future work recommendations.

CHAPTER 7

CONCLUSION AND FUTURE WORK RECOMMENTATIONS

7.1 Conclusion

The pipelined DLX processor was successfully designed and implemented using VHDL based loosely on the previous DLX project carried out in UTM. The redesigned DLX processor was successfully simulation using Altera's Quartus 2 7.2 Web Edition software suite.

The pipelined DLX utilized a five-stage instruction pipeline (instruction fetch, instruction decode, execute, memory and write-back) to operate on instruction fed into the processor.

This project consisted of several phases of work. In this first part of the project, much effort was been spent to understand the DLX architecture and source code, as well as delving into pipelining concepts and design considerations. The next stage involved successfully re-simulated and verifying the previous non-pipelined DLX in Quartus. This is imperative to verifying the functionality of the previous design can be reproduced, as well as strengthens knowledge of the DLX architecture and familiarizing with the Quartus design tools.

The two main components of the DLX – the datapath and the control unit - were redesigned to enable pipelined execution of instructions. Once the VHDL coding of the submodules were completed, all the sub-blocks were integrated and validated in Quartus. A lot of time and effort was spent iterating between coding the blocks, integrating the design and validating the implementation in Quartus.

Once the design was functionally validated, performance analysis work was carried out, focusing on comparing the previous non-pipelined DLX versus the redesigned pipelined-DLX processor.

The pipelined DLX showcased a 25% instruction execution speedup measured by number of clock cycles, as compared to the non-pipelined DLX that came at a cost of 23% increase in logic element utilization as measured by Quartus tool. Finally, exploration of possible future work to further improve the performance of the pipelined DLX processor was done.

Throughout the design, implementation and validations stages of the project, numerous hindrances were encountered, which includes lack VHDL coding proficiency, familiarizing with the Quartus tool, as well as handling branch instructions in the pipelined architecture. Each setback was handled meticulously and diligently.

In summation, a wealth of knowledge was gained from this project which could not be lesson-taught. In depth knowledge of computer architecture, VHDL coding, pipelining concepts and implementation, branching handling in microprocessor design as well as generic skills were among the expertise acquired through this project.

7.2 Recommendations for Future Work

Among several possible future work recommendations presented here, the most pivotal would be a full scale implementation of the pipelined DLX on FPGA (and possibly fabricated ASIC). With the design implemented on FPGA, real world performance of the pipelined DLX can be measures, particularly with the presence of real external memory interactions. This will introduce the need for better timing synchronization and handling between the DLX processor and the external memory.

Another possible path to explore in further improving the performance of the DLX processor would be the introduction of a branch-prediction algorithm and hardware module. This can be realized through multiple fetches from memory per load cycle and a sub-module that looks-ahead two instruction in advance to prepare for branch instructions in the program. This will eliminate the 3-cycle penalty paid whenever an instruction in the pipeline is decoded as a branch, resulting in the entire pipeline being flushed.

The addition of a cache (either data cache or instruction cache) would also significantly increase the performance of the DLX processor. In this case, an instruction cache would be more ideal to work with the pipelined architecture since latency to fetch the instructions from the memory can be reduced to a minimum if instructions are cached, regardless of whether the branch prediction unit is implemented.