Spring 2011

# Design and evaluation of an adaptable vector coprocessor for multicores

Timothy William Steele
*New Jersey Institute of Technology*

## ABSTRACT

## DESIGN AND EVALUATION OF AN ADAPTABLE VECTOR COPROCESSOR FOR MULTICORES

**by**
**Timothy William Steele**

Future applications for multi-core processor systems will require increased signal processing power along with increased resource utilization and decreased power consumption. Conservative power consumption will be of paramount importance primarily for battery-powered portable multi-core platforms (e.g., advanced cell phones, tablet computers, etc.). This thesis investigates the robustness, efficiency and effectiveness of vector coprocessor sharing policies in multi-core environments. Vector coprocessor sharing is based on an innovative design for a vector lane that forms the building block for the creation of larger vector coprocessors. This innovative lane design contains a floating-point multiply unit, a floating-point add/subtract unit, a miscellaneous function unit, a load/store unit, and a vector register file. The design was prototyped and benchmarked on a field programmable gate array (FPGA) for a multitude of configurations to evaluate the performance and power consumption. The configurations included one or two host processors and two, four, eight, sixteen or thirty-two lanes. Sample applications in benchmarking were the fast Fourier transform, finite impulse response filter, matrix multiplication and LU matrix decomposition. As an additional experiment, a reconfigurable unit was added to the lane and configured as either a combined floating-point multiply/add or a floating-point divide to better match the needs of specific applications. The results show the versatility of the design towards high performance and controllable power consumption.

# DESIGN AND EVALUATION OF AN ADAPTABLE VECTOR COPROCESSOR FOR MULTICORES

by
**Timothy William Steele**

**A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering**

**Department of Electrical and Computer Engineering**

**May 2011**

## APPROVAL PAGE

**DESIGN AND EVALUATION OF AN ADAPTABLE VECTOR COPROCESSOR FOR MULTICORES**

**Timothy William Steele**

| | |
|---|---|
| Dr. Sotirios G. Ziavras, Thesis Advisor<br>Professor of Electrical and Computer Engineering, NJIT | Date |
| Dr. Sui-Hoi E. Hou, Committee Member<br>Associate Professor of Electrical and Computer Engineering, NJIT | Date |
| Dr. Jie Hu, Committee Member<br>Assistant Professor of Electrical and Computer Engineering, NJIT | Date |

## BIOGRAPHICAL SKETCH

**Author:** Timothy William Steele

**Degree:** Master of Science

**Date:** May 2011

**Undergraduate and Graduate Education:**

- Bachelor of Science in Electrical Engineering,
  Rochester Institute of Technology, Rochester, NY, USA, 1991

**Major:** Electrical Engineering

To my parents, William and Maral for making it all possible, to my wife, Rhonda for patience and support, and to my daughter, Megan for the future.

## ACKNOWLEDGMENT

First and foremost I offer my sincerest gratitude to my thesis advisor, Professor Sotirios G. Ziavras for introducing me to the topic of sharable vector coprocessors and their application to current and future design challenges. I am indebted to him for his time and patience while I completed this thesis.

It is my pleasure to thank Dr. Sui-Hoi E. Hou and Dr. Jie Hu for being members of my thesis committee. Their support and guidance are greatly appreciated.

My special thanks to Spiridon F. Beldianu and Christopher Dahlberg for their continuous assistance towards the completion of this thesis. Without their time and efforts I would have never succeeded at this task.

Lastly, I thank my family and friends for their encouragement and support while I was studying at New Jersey Institute of Technology.

## TABLE OF CONTENTS

**TABLE OF CONTENTS**
**(Continued)**

# LIST OF TABLES

## LIST OF FIGURES

**LIST OF FIGURES**
**(Continued)**

# CHAPTER 1

# INTRODUCTION

## 1.1 Need for Coprocessor Sharing in Multicore Processors

A shared vector coprocessor bank comprised of multiple vector lanes makes sense for multiple processor core architectures for several reasons. The first is that with appropriate resource allocation, a larger percentage of the entire coprocessor can be utilized at any given time. A number of lanes can be assigned to each processor based on the needs of the currently running application rather than on a design decision made during the architecture design. Any unused lanes would then be in a power-down mode, thus limiting power usage by keeping active only an optimum number of lanes. The second is that as the number of processor cores grows, the coprocessor bank can grow more slowly to meet the needs of the increased number of expected concurrent applications. This saves a significant number of transistors and a related amount of area on the die. The third is that an explicit vector design per core will not have a high utilization despite the rather frequent need to parallelize the processing of floating point data ever present in most digital signal processing applications.

## 1.2 Existing Approaches

Existing approaches in the literature, such as VIRAM [Kozyrakis and Patterson 2003], SODA [Lin et al. 2006] and AnySP [Woh et al. 2010] are designed as single microprocessor cores with attached vector operation support. These resources are not shared and are closely coupled with the microprocessor, thus limiting the possibility of taking advantage of parallelism between different threads while also underutilizing silicon resources. In addition, “soft” vector processor solutions have been investigated [Cho et al. 2006; Lin et al. 2006; Yiannacouras et al. 2008; Yu et al. 2009; Yang and Ziavras 2005] as coprocessor add-ons to an FPGA-based microprocessor. However, these designs are done using a fixed vector register length and as a result are not a generalized solution to the real issues of varying vector length for different applications or within the same application. A proposed architecture for simultaneous sharing and changeable vector register lengths [Beldianu and Ziavras 2011] forms the basis for the design utilized in this investigation. This adaptive vector processor sharing takes advantage of thread level parallelism by allowing multiple vector length instructions to pass through the lanes at the same time.

## 1.3 Motivations and Objectives

The motivation behind this effort is to provide a scalable, flexible solution to the problem of floating point vector processing for multicores. The design described within this paper is configurable to support between two and thirty-two processing lanes, with three distinct modes of sharing. The first mode is Coarse-grain Temporal Sharing (CTS), where all the available lanes are assigned to a microprocessor on an as-needed basis. The second mode is Vector Lane Sharing (VLS), where the available lanes are divided into two distinct groups, mimicking a dedicated vector coprocessor involving one half of the total vector lanes. The third mode is Fine-grain Temporal Sharing (FTS), where instruction requests from multiple cores are interleaved across the entire array of lanes [Beldianu and Ziavras 2011]. The objective is to benchmark these techniques using four different common signal processing applications in order to show how performance, cost and power consumption are affected by various lane configurations and the aforementioned vector lane sharing modes of operation.

# CHAPTER 2

# PROPOSED FRAMEWORK

## 2.1 Basic Architecture

The architecture proposed for this investigation is shown in Figure 2.1. It consists of two Xilinx™ MicroBlaze™ processors with associated program store memory connected through the Local Memory Bus (LMB). Attached to the common Processor Local Bus (PLB) is the array of Xilinx Block RAM (BRAM) memory blocks which form the interface to the Vector Processor (VP) lanes. Also connected to the PLB is the Hardware Internal Configuration Access Port (HWICAP) which allows either processor to reconfigure a predetermined portion of the lane known as the Reconfigurable Module (RM). Attached to each processor via a Fast Simplex Link (FSL) is a Vector Controller (VC) which handles scheduling of instructions to the lanes, and the flow of data into and out of the BRAM memory blocks. The two VC modules request access to the lanes through the Scheduler, which is granted in an arbitrated round-robin fashion. The Floating Point (FP) data flows between the BRAM memory blocks and the VP lanes through a Memory Crossbar (MC) which is configurable to support up to thirty-two lanes and thirty-two memory blocks. The MC also can function as a shuffle network to route data directly from one lane to another rather than requiring extra cycles to store the data in BRAM and retrieve it. For the applications used in this thesis, a minimum of eight memory blocks are used for the two and four lane cases due to the minimum required memory for the software as written.

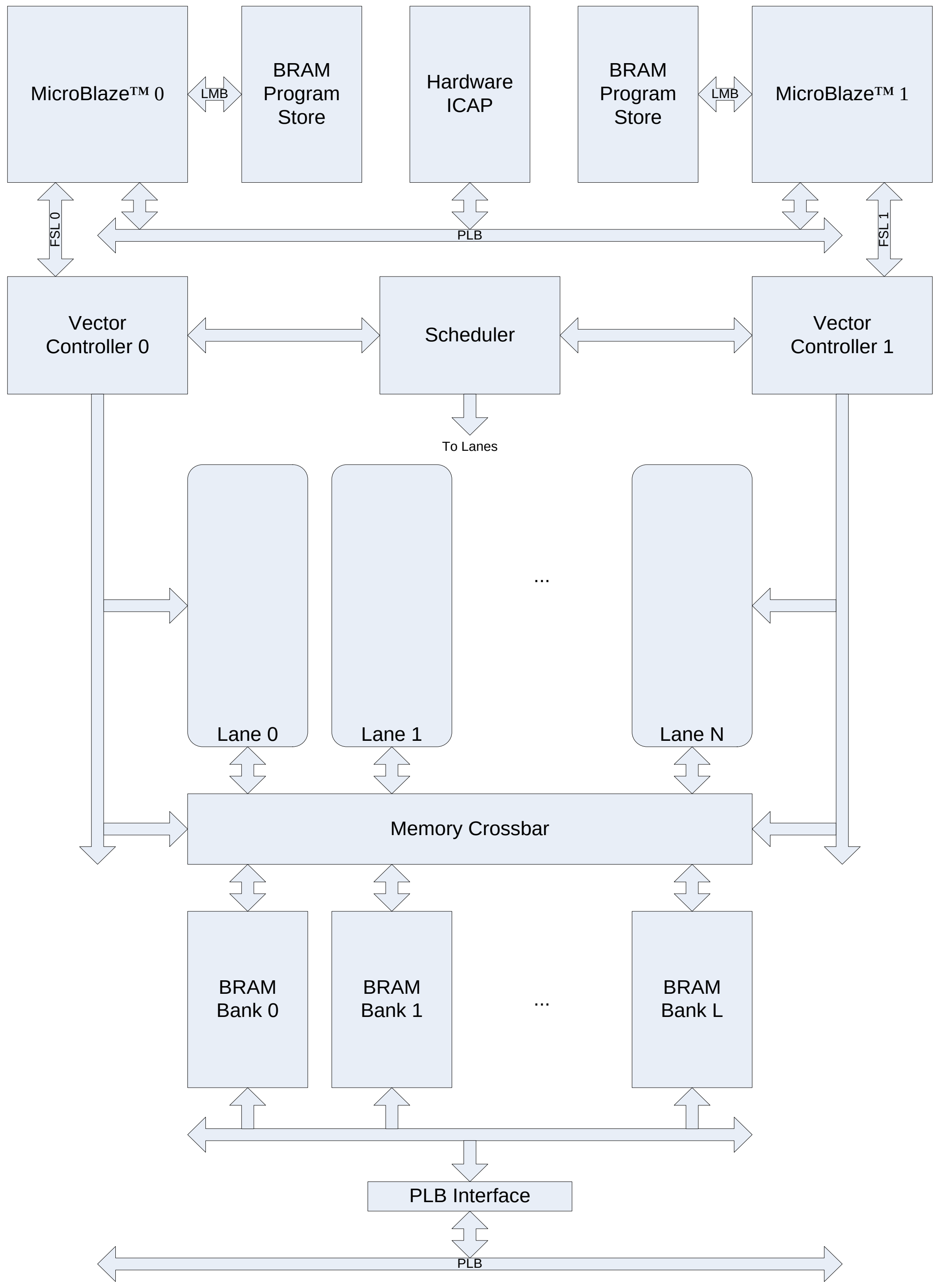

**Figure 2.1** Top level architecture.

## 2.2 Design Details

All the designs described below are written in synthesizable VHDL and targeted on a Xilinx Virtex-5 FPGA. The various floating point functions and ram blocks, as well as the MicroBlaze processors, are Xilinx core functions used for ease of synthesis.

### 2.2.1 Vector Controller (VC) Details

The Vector Controller (VC) receives instructions over the Xilinx Fast Simplex Link. It coordinates through the Scheduler when the commands are presented to the lanes and also determines how many of the lanes are dedicated to the current application. The controller and scheduler provide support for the three vector processor sharing architectures described in the introduction. The control signals from the VC and Scheduler provide all the information needed by the lane to indicate which VC currently controls the lane, the total number of lanes currently assigned to the VC, the index of the lane relative to the others attached to the VC to provide a continuous address space for the vector registers, and the number of register elements located in this lane’s Vector Register File. This information is stored in four discrete registers internal to each lane.

### 2.2.2 Vector Processor Lane Details

Each lane consists of a Load/Store function (on the left in Figure 2.2), an Arithmetic/Logic Unit (on the right), and a multi-port Vector Register File. Instructions coming into the lane from either Vector Controller are decoded, the respective operands fetched, and the dictated processing is done. Separate functional blocks in the ALU provide floating point multiplication, addition or subtraction, and miscellaneous functions such as negate, invert and move. A fourth block is added to take advantage of the capability of the Xilinx FPGA architecture to reprogram portions of the device while the system is operating. This reconfigurable module can be set at run time or at compile time to perform functions such as combined multiply and add/subtract, or divide, depending on the needs of the application.

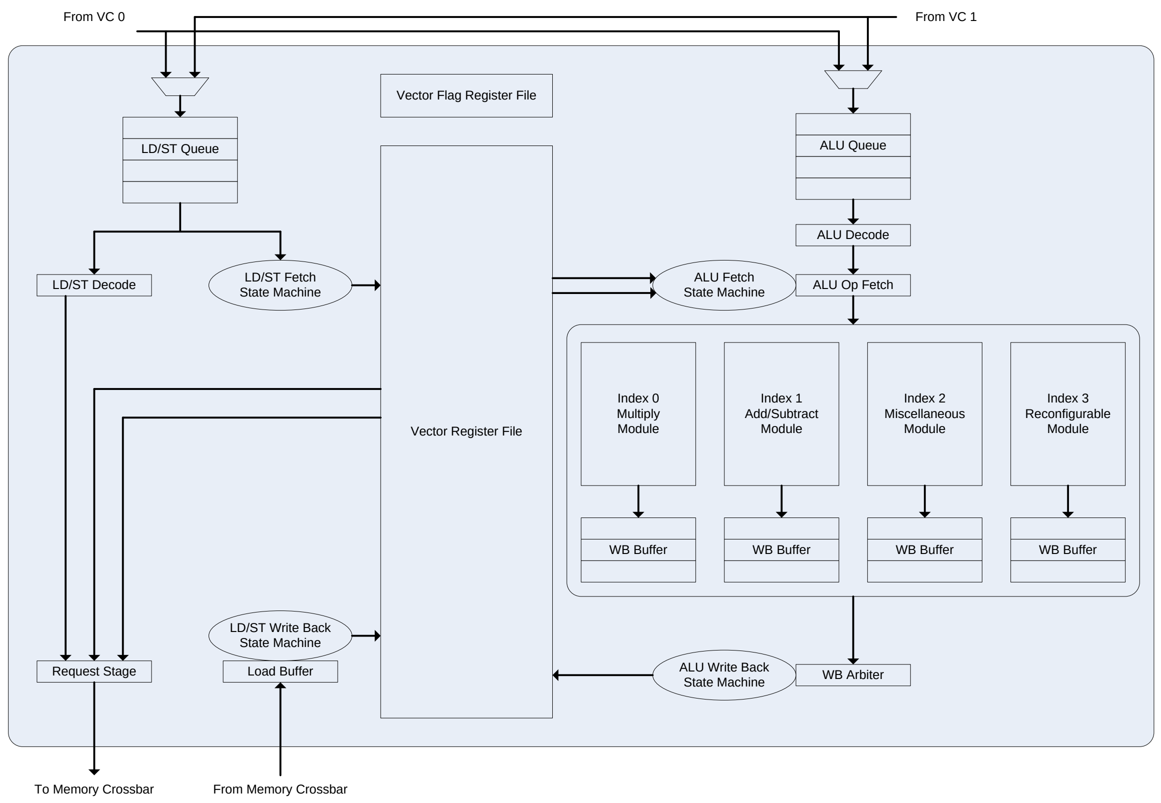

**Figure 2.2** Vector processor lane architecture.

**2.2.2.1 Multiply Module.** The Multiply module contains a Xilinx floating point core which performs a fully pipelined single precision multiplication with a latency of six clock cycles. This function supports three modes of operation: a vector times a vector, a vector times a scalar or a scalar times a vector. In addition, the module includes a write-back buffer because multiple instructions could finish on the same clock but only one write-back port is available to the vector register file. The results of the operation are stored in this buffer along with side information such as priority, an ignore flag, and a ready flag.

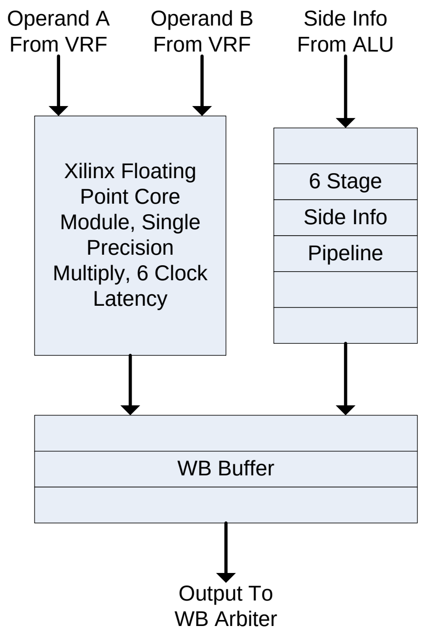
Operand A From VRF
Operand B From VRF
Side Info From ALU
Xilinx Floating Point Core Module, Single Precision Multiply, 6 Clock Latency
6 Stage Side Info Pipeline
WB Buffer
Output To WB Arbiter

**Figure 2.3** Multiply module architecture.

**2.2.2.2 Add/Subtract Module.** The Add/Subtract module contains a Xilinx floating point core which performs a fully pipelined single precision addition or subtraction with a latency of six clock cycles. This module supports the same three operating modes as the multiply module. In addition, the module includes an identical write-back buffer to the Multiply module.

**2.2.2.3 Miscellaneous Module.** The Miscellaneous module provides the capability to invert or negate a scalar or vector quantity, take the absolute value, or another path for data to move without using the Load/Store function. These functions take a single clock cycle to complete and the module includes a write-back buffer, which is the same as for the other functions.

**2.2.2.4 Reconfigurable Module (RM).** The Reconfigurable Module (RM) takes advantage of the ability in the Xilinx FPGA architecture to reprogram pre-defined areas of the device in the designed system. In this case, the floating point divide function (with a latency of six clock cycles) and a combined multiply (a vector times a scalar) and add/subtract (a vector added/subtracted with the result of the multiply) function (with a latency of eight clock cycles) take up approximately the same number of resources on the device. As a result, it is relatively straight forward to provide the configuration files for each possible use and allow the processor to program the lane with the function which makes the most sense for a particular application. Another possible function is the pipelined square root function. Reconfiguration can occur under the control of the microprocessor, with the configuration file stored externally to the FPGA. Since the largest of the proposed units uses approximately 980 slice registers, the region will consist of 25

Reconfigurable Frames. The time required to change from one function to another will take about 370 microseconds per lane, based on the information in the Xilinx Partial Reconfiguration User Guide. [Xilinx 2010]

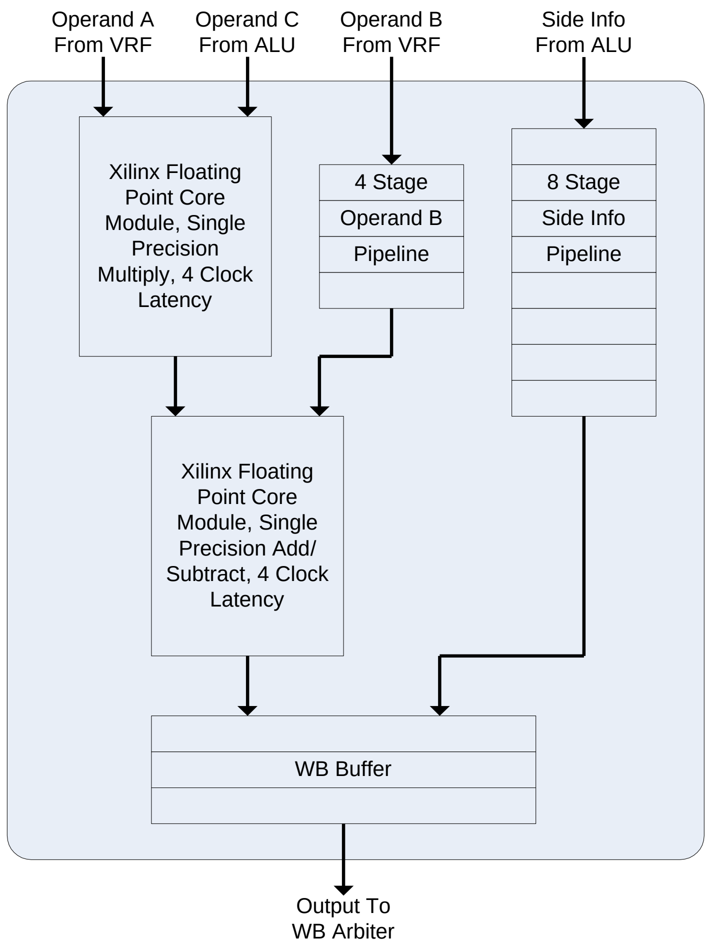
Operand A From VRF
Operand C From ALU
Operand B From VRF
Side Info From ALU
Xilinx Floating Point Core Module, Single Precision Multiply, 4 Clock Latency
4 Stage Operand B Pipeline
8 Stage Side Info Pipeline
Xilinx Floating Point Core Module, Single Precision Add/ Subtract, 4 Clock Latency
WB Buffer
Output To WB Arbiter

**Figure 2.4** Multiply/add module architecture.

**2.2.2.5 Load/Store Module.** The Load/Store module handles all data traffic into and out of the lane. It interfaces directly with the Memory Crossbar to load data into the Vector Register File or to return calculation results to the Vector Memory. This module also controls the shuffle function of the crossbar so that data can pass from one lane directly to another without passing through the Vector Memory.

**2.2.2.6 Vector Register File (VRF).** The Vector Register File (VRF) consists of 512 32-bit memory locations using the Xilinx Block RAM function (BRAM). Both the Load/Store and ALU sides of the lane require two read ports and one write port. This is handled in the FPGA by duplication of the BRAM and by running the interface at twice the processing clock rate. In addition, a 512x1 bit Flag register is included with each lane, as well as the four configuration registers described in the VC section, above.

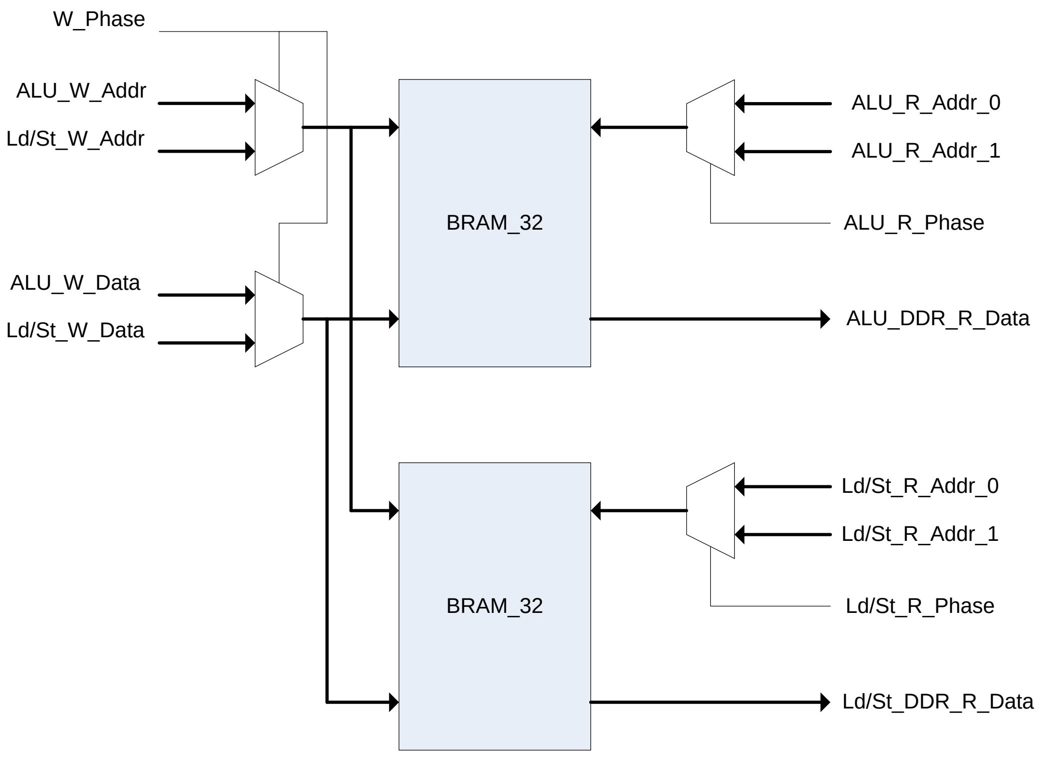

**Figure 2.5** Vector register file architecture.

### 2.2.3 Memory Crossbar Details

The Memory Crossbar provides a direct connection between the N lanes and the L BRAMs used in the Vector Memory. Access is arbitrated using a round-robin scheme for each input and output port and if no contention exists, all ports can be active on a single clock cycle. The architecture allows for the number of ports to be set to match the number of lanes and the number of block memories. The lane requests access to a specific BRAM and the arbiter acknowledges when the path through the crossbar is available. The BRAMs are set up as dual port devices, with one port dedicated to the lane through the crossbar, and the other dedicated to the PLB. The number of lanes does not need to match the number of BRAMs.

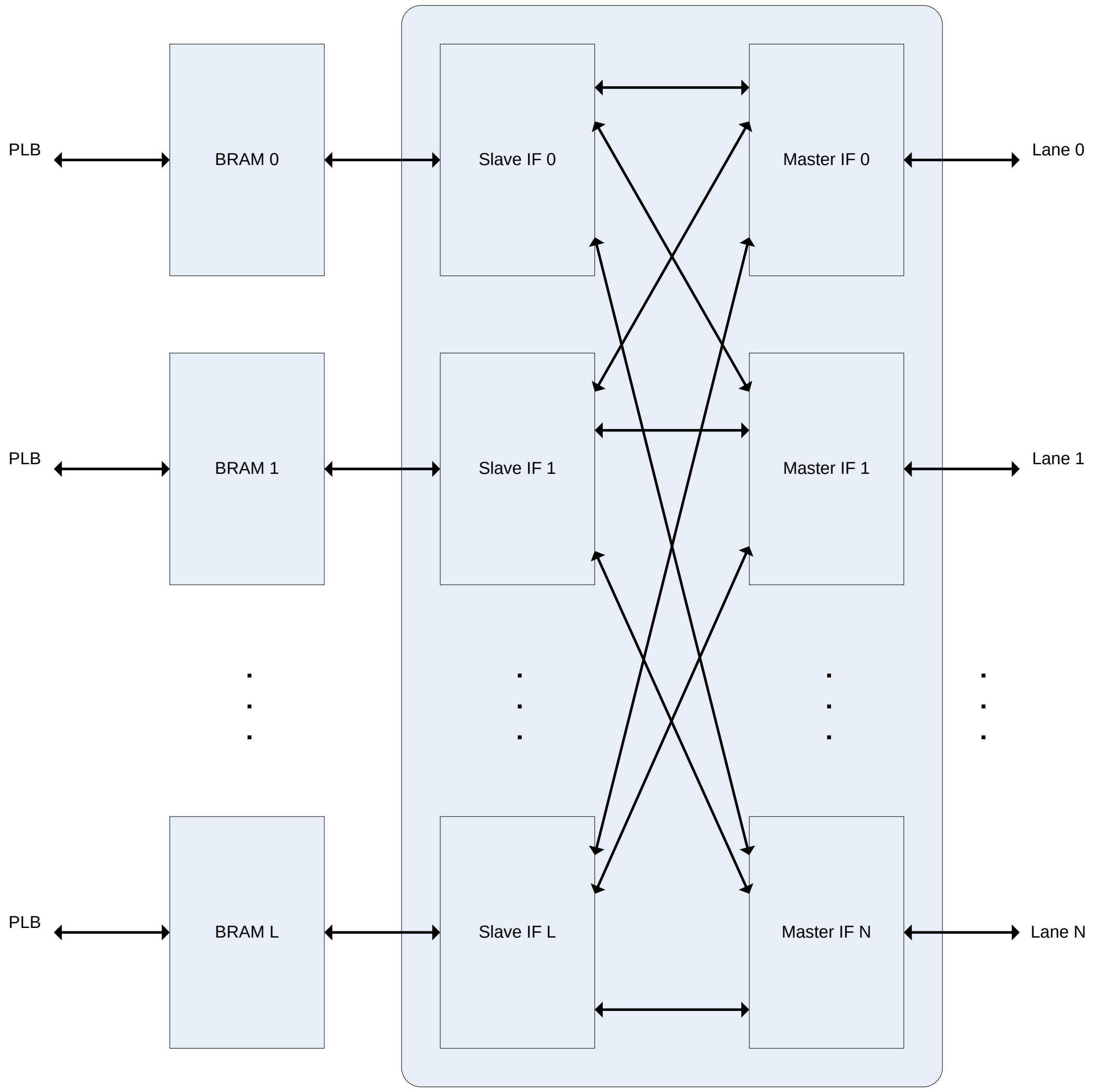

**Figure 2.6** Memory crossbar architecture.

# CHAPTER 3

# APPLICATION BENCHMARKING

## 3.1 Fast Fourier Transform (FFT)

The first application used to benchmark the performance of the multiple lane configurations was the 32-point decimation-in-time radix-2 butterfly fast Fourier transform. This was implemented using a five-stage butterfly where each stage includes complex multiplication and addition followed by a shuffle operation through the Memory Crossbar. Due to architectural limitations in the available size of vector lengths and number of vector registers, this application was not run on the two-lane version of the design. Two different scenarios were run and charted (see Tables 3.1 and 3.2, and Figures 3.1 and 3.2), first with simple processing (e.g., one complete FFT per pass through the loop) and, second with double processing of two complete FFTs per loop. The scenarios were run in each of the three lane sharing configurations and over the four instantiated lane conditions.

**Table 3.1** FFT 32, Simple

| | No. of Lanes | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFT 32 Simple | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| | CTS | 1.00 | 64.26 | 39.18 | 1.32 | 40.70 | 22.15 | 1.58 | 24.86 | 13.29 | 1.73 | 13.88 | 7.24 |
| | VLS | 1.23 | 72.30 | 43.07 | 2.00 | 34.24 | 33.30 | 2.63 | 40.52 | 20.44 | 3.17 | 24.93 | 11.11 |
| | FTS | 1.41 | 88.67 | 53.54 | 2.38 | 69.77 | 38.30 | 3.07 | 39.35 | 21.02 | 3.42 | 21.25 | 11.06 |

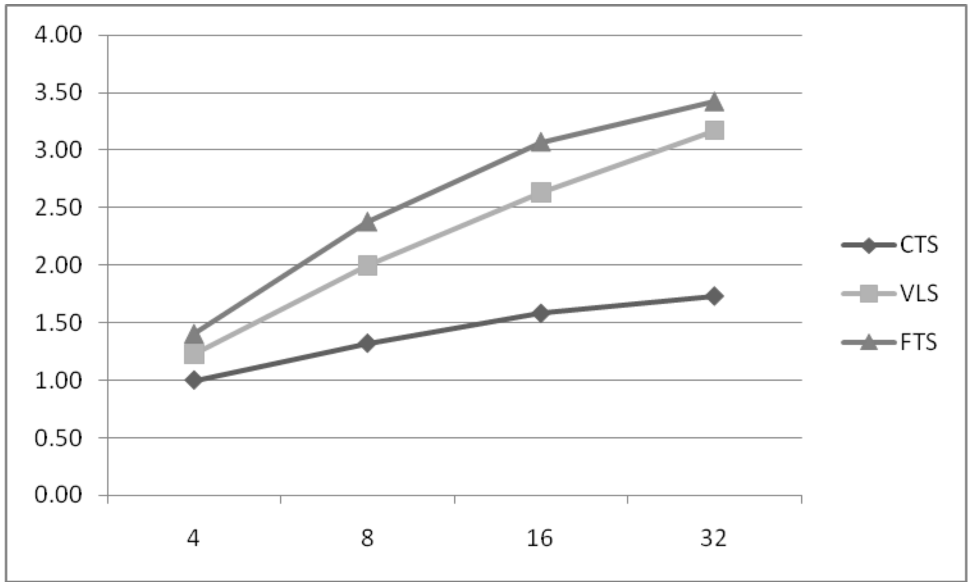

**Figure 3.1** FFT 32, simple.

**Table 3.2** FFT 32, Double

| FFT 32 Double | No. of Lanes | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| | CTS | 1.26 | 81.52 | 49.07 | 2.04 | 62.67 | 34.34 | 2.29 | 35.95 | 21.02 | 2.29 | 18.37 | 9.63 |
| | VLS | 1.37 | 79.99 | 47.10 | 2.53 | 44.49 | 42.92 | 4.08 | 62.79 | 26.37 | 4.57 | 35.94 | 14.21 |
| | FTS | 1.46 | 94.09 | 57.03 | 2.82 | 76.29 | 41.47 | 4.63 | 49.66 | 26.39 | 4.57 | 25.34 | 13.16 |

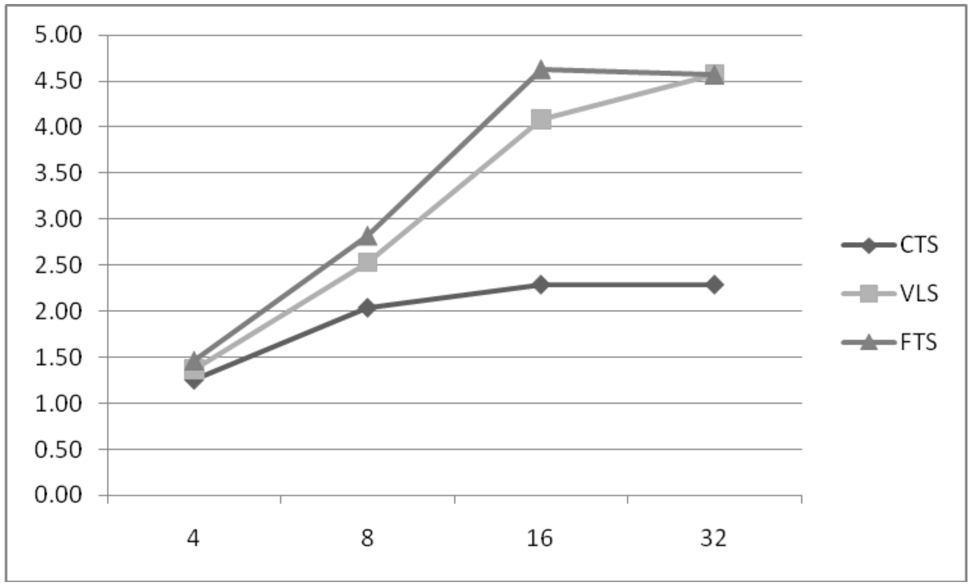

**Figure 3.2** FFT 32, double.

## 3.2 Finite Impulse Response Filter (FIR)

The second application used to benchmark the multiple lane configurations was the 32-tap finite impulse response filter, implemented using the outer product format [Sung and Mitra 1987] which avoids the reduction operation. Three different scenarios were run and charted (see Tables 3.3, 3.4, and 3.5, and Figures 3.3, 3.4, and 3.5), first with a vector length of 32 and no loop unrolling, second with a vector length of 64 and no loop unrolling, and third with a vector length of 64 and unrolling the loop three times (for a total of four passes through the loop). The scenarios were run in each of the three lane sharing configurations and over the five instantiated lane conditions. An additional set of runs was done using the RM configured as a combined Multiply/Add functional unit and the application was changed to take advantage of this where possible. The results are compared for three scenarios, first with a vector length of 32 and no loop unrolling, the second with the same vector length and unrolling the loop three times and the third with a vector length of 128 and unrolling the loop three times. All three scenarios were run on the three lane sharing configurations and the first two over the five instantiated lane conditions (see Tables A.1, A.2 and A.3, and Figures A.1, A.2 and A.3). The third was run with four, eight, sixteen and thirty-two lanes due to the limit of available vector registers with two lanes.

**Table 3.3** FIR 32, VL=32, No Loop Unroll

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIR 32 | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.00 | 37.90 | 19.82 | 1.39 | 26.39 | 13.80 | 1.74 | 16.34 | 8.44 | 1.98 | 9.37 | 4.75 | 2.13 | 5.03 | 2.55 |
| No Loop Unroll | VLS | 1.28 | 48.43 | 25.28 | 2.00 | 37.80 | 19.19 | 2.79 | 26.42 | 13.44 | 3.47 | 16.45 | 8.36 | 3.95 | 9.37 | 5.11 |
| | FTS | 1.95 | 73.91 | 38.68 | 2.77 | 52.53 | 26.74 | 3.47 | 32.90 | 16.71 | 3.95 | 18.74 | 9.51 | 4.25 | 10.07 | 4.76 |

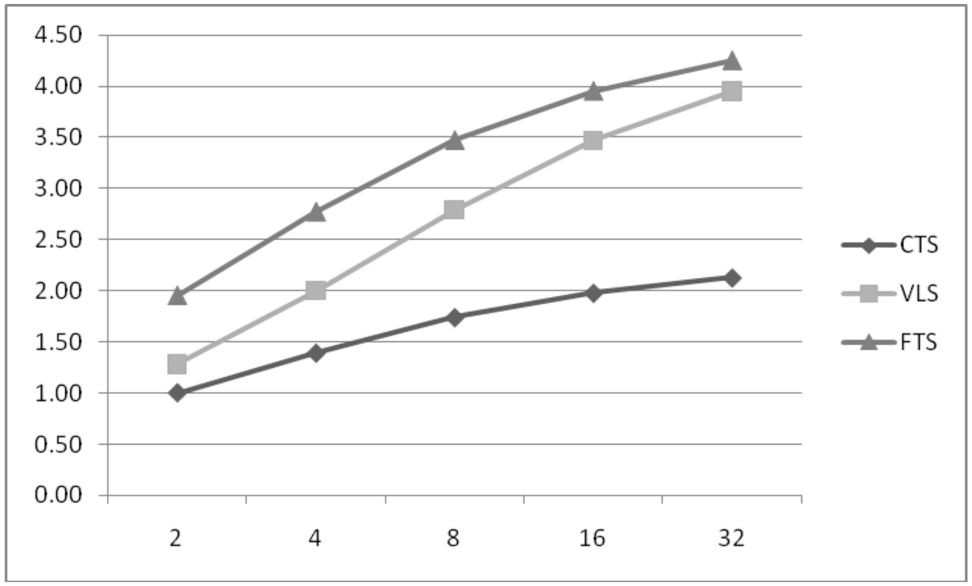

**Figure 3.3** FIR32, VL=32, no loop unroll.

**Table 3.4** FIR 32, VL=64, No Loop Unroll

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIR 32 | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=64 | CTS | 1.28 | 49.35 | 25.86 | 2.00 | 37.87 | 19.80 | 2.79 | 26.42 | 13.42 | 3.47 | 16.44 | 4.75 | 3.96 | 9.37 | 4.76 |
| No Loop Unroll | VLS | 1.48 | 56.11 | 28.82 | 2.56 | 48.41 | 24.61 | 3.99 | 37.81 | 19.25 | 5.58 | 26.42 | 8.36 | 6.94 | 16.45 | 8.36 |
| | FTS | 2.24 | 84.92 | 44.25 | 3.90 | 73.88 | 37.58 | 5.54 | 52.60 | 26.71 | 6.94 | 32.86 | 9.51 | 7.91 | 18.74 | 9.51 |

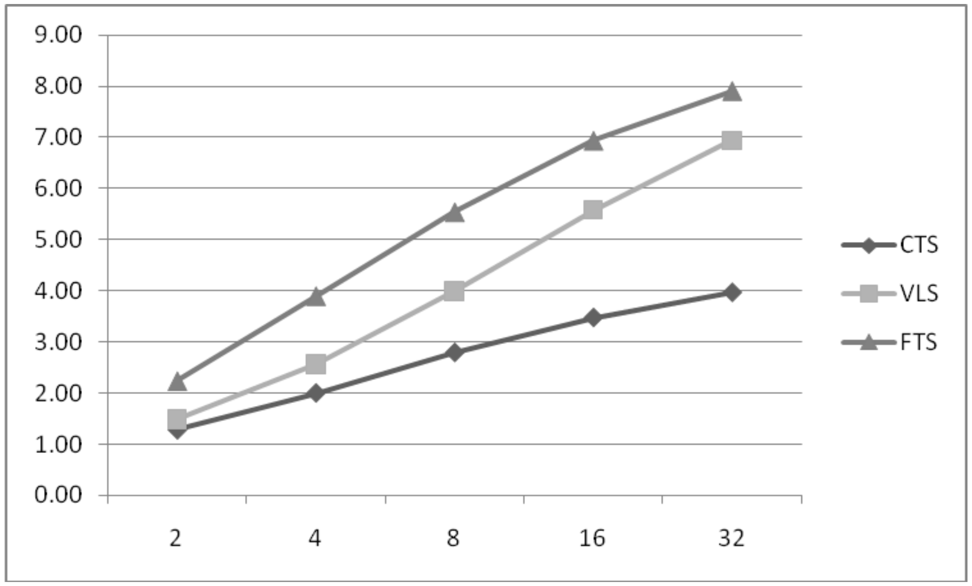

**Figure 3.4 FIR32**, VL=64, no loop unroll.

**Table 3.5** FIR 32, VL=64, Unroll 3 Times

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIR 32 | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=64 | CTS | 2.36 | 89.50 | 46.66 | 4.32 | 81.89 | 41.92 | 6.19 | 58.63 | 29.79 | 7.64 | 36.18 | 18.39 | 8.65 | 20.49 | 10.41 |
| Unroll 3 Times | VLS | 2.47 | 93.36 | 48.60 | 4.72 | 89.34 | 45.93 | 8.60 | 81.50 | 41.52 | 12.29 | 58.22 | 35.65 | 15.19 | 35.99 | 20.58 |
| | FTS | 2.61 | 99.85 | 51.45 | 5.29 | 99.56 | 50.89 | 10.07 | 95.37 | 48.35 | 14.70 | 70.16 | 29.64 | 17.08 | 40.45 | 18.30 |

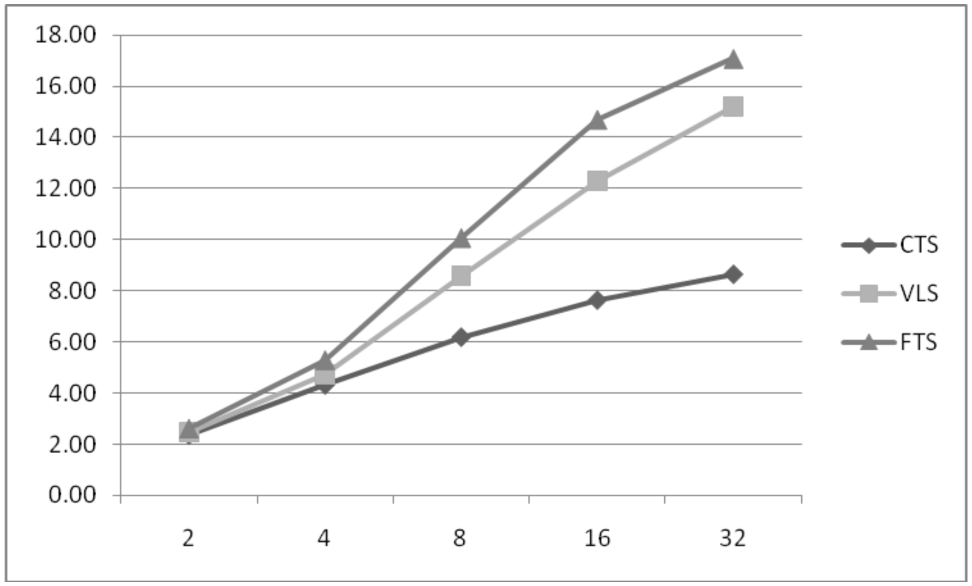

**Figure 3.5** FIR32, VL=64, unroll 3 times.

## 3.3 Matrix Multiplication (MM)

The third application is matrix multiplication, which uses the same procedure as the FIR filtering. The Single-precision real Alpha X plus Y (SAXPY) algorithm is run in a loop to obtain one row result for each pass. Two different scenarios were run and charted (see Tables 3.6 and 3.7, and Figures 3.6 and 3.7), first with a vector length of 32 and unrolling the loop once, and second with a vector length of 64 and unrolling the loop once. The scenarios were run in each of the three lane sharing configurations and over the five instantiated lane conditions. An additional set of runs was done using the RM configured as a combined Multiply/Add functional unit and the application was changed to take advantage of this where possible. The results are compared for two scenarios, first with a vector length of 32 and unrolling the loop one time and second with a vector length of 64 and unrolling the loop one time. The scenarios were run for all three lane sharing configurations and over the five instantiated lane conditions (see Tables B.1 and B.2, and Figures B.1 and B.2).

**Table 3.6** MM, VL=32, Unroll 1 Time

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MM | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.00 | 71.10 | 72.21 | 1.47 | 52.22 | 53.08 | 1.91 | 33.94 | 34.50 | 2.09 | 18.67 | 18.91 | 2.20 | 9.80 | 9.96 |
| Unroll 1 Time | VLS | 1.17 | 82.98 | 84.63 | 2.00 | 71.04 | 72.36 | 3.04 | 53.51 | 68.20 | 3.67 | 32.86 | 33.32 | 4.18 | 18.58 | 19.38 |
| | FTS | 1.38 | 98.13 | 99.87 | 2.49 | 88.74 | 90.34 | 3.79 | 67.09 | 54.50 | 4.08 | 36.57 | 37.13 | 4.36 | 19.27 | 19.59 |

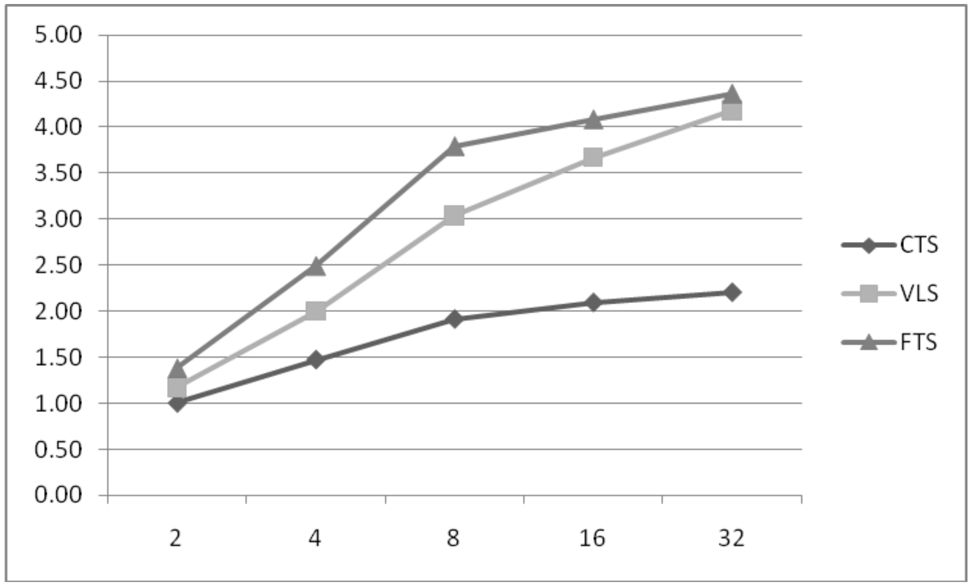

**Figure 3.6** MM, VL=32, unroll 1 time.

**Table 3.7** MM, VL=64, Unroll 1 Time

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MM VL=64 Unroll 1 Time | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| | CTS | 1.16 | 82.59 | 84.29 | 1.96 | 69.18 | 70.25 | 2.83 | 70.13 | 68.96 | 3.48 | 30.95 | 31.53 | 3.94 | 17.61 | 17.91 |
| | VLS | 1.28 | 90.72 | 92.14 | 2.32 | 82.47 | 84.08 | 3.96 | 71.68 | 70.31 | 5.66 | 50.04 | 50.88 | 6.96 | 31.16 | 31.68 |
| | FTS | 1.38 | 98.35 | 100.00 | 2.75 | 97.73 | 99.51 | 4.95 | 89.46 | 87.98 | 6.84 | 61.03 | 61.98 | 7.84 | 34.97 | 35.52 |

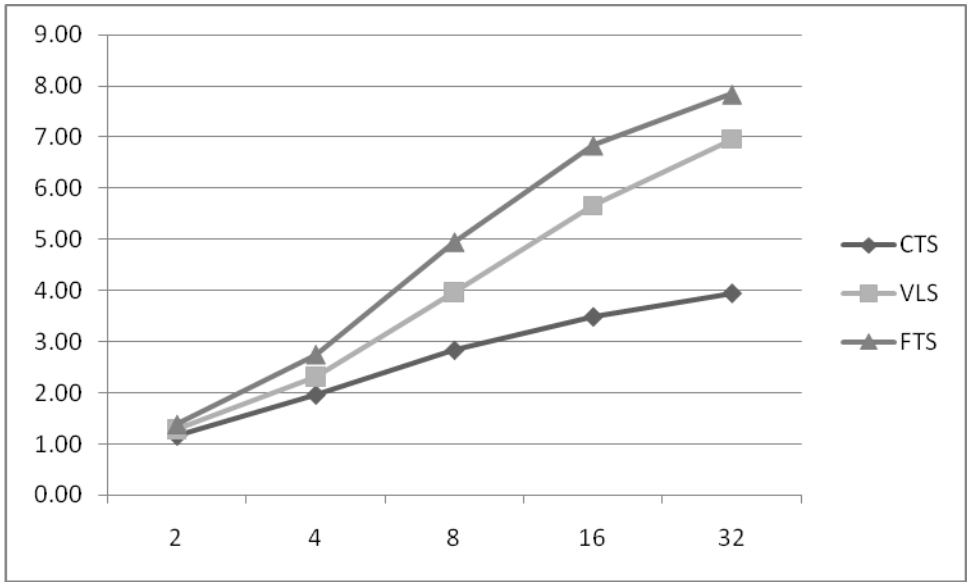

**Figure 3.7** MM, VL=64, unroll 1 time.

## 3.4 LU Decomposition (LU)

The fourth application is LU decomposition, where the Lower and Upper diagonal matrices are generated from a dense 128x128 element matrix using the Doolittle algorithm [Golub and Van Loan 1996]. Two different scenarios were run and charted (see Tables 3.8 and 3.9, and Figures 3.8 and 3.9), first with a vector length of 64 and no loop unrolling, and second with a vector length of 32 and no loop unrolling. The scenarios were run in each of the three lane sharing configurations and over the five instantiated lane conditions. An additional set of runs was done using the RM configured for a Divide function, the application was changed to take advantage of this where possible, and these results are compared for two scenarios, first with a vector length of 64 and no loop unrolling and second with a vector length of 32 and no loop unrolling. The scenarios were run for the CTS and FTS lane sharing configurations and over the four, eight, sixteen, and thirty-two lane cases (see Tables C.1 and C.2, and Figures C.1 and C.2). The two lane case was not used because once the application was modified to use the Divide function too many vector registers were required. The VLS lane sharing configuration was not run for the same reason.

**Table 3.8** LU Decomposition, VL=64, No Loop Unroll

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LU Decomp | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=64 | CTS | 1.00 | 53.20 | 54.24 | 1.83 | 40.28 | 40.87 | 1.81 | 20.11 | 20.40 | 1.64 | 10.06 | 10.20 | 1.83 | 5.03 | 5.10 |
| No Unrolls | VLS | 1.33 | 59.02 | 60.49 | 2.40 | 53.24 | 54.71 | 3.67 | 39.62 | 41.05 | 3.67 | 19.86 | 20.50 | 3.67 | 9.97 | 10.25 |
| | FTS | 1.75 | 90.19 | 93.38 | 3.58 | 79.71 | 82.41 | 3.62 | 39.93 | 41.35 | 3.62 | 20.00 | 20.63 | 3.62 | 10.00 | 10.32 |

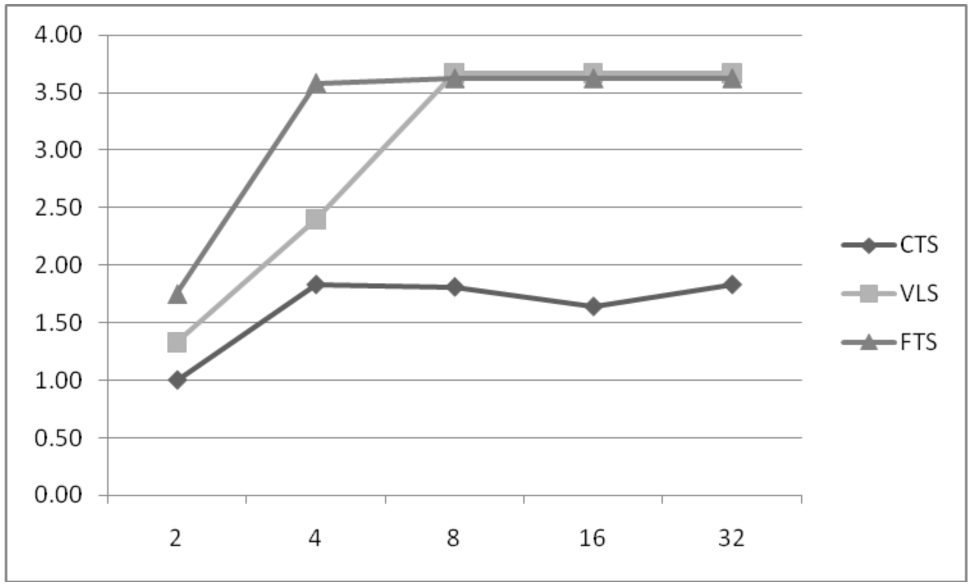

**Figure 3.8** LU decomposition, VL=64, no loop unroll.

**Table 3.9** LU Decomposition, VL=32, No Loop Unroll

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LU Decomp | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.29 | 39.84 | 41.31 | 1.83 | 19.90 | 20.61 | 1.81 | 9.58 | 10.33 | 1.72 | 4.98 | 5.15 | 1.83 | 2.49 | 2.58 |
| No Unrolls | VLS | 2.40 | 52.28 | 56.25 | 3.67 | 38.25 | 42.03 | 3.62 | 19.05 | 21.03 | 3.67 | 9.55 | 10.49 | 3.62 | 4.77 | 5.24 |
| | FTS | 3.58 | 76.57 | 84.82 | 3.71 | 38.62 | 42.89 | 3.62 | 18.74 | 21.10 | 3.62 | 9.67 | 10.71 | 3.58 | 4.84 | 5.35 |

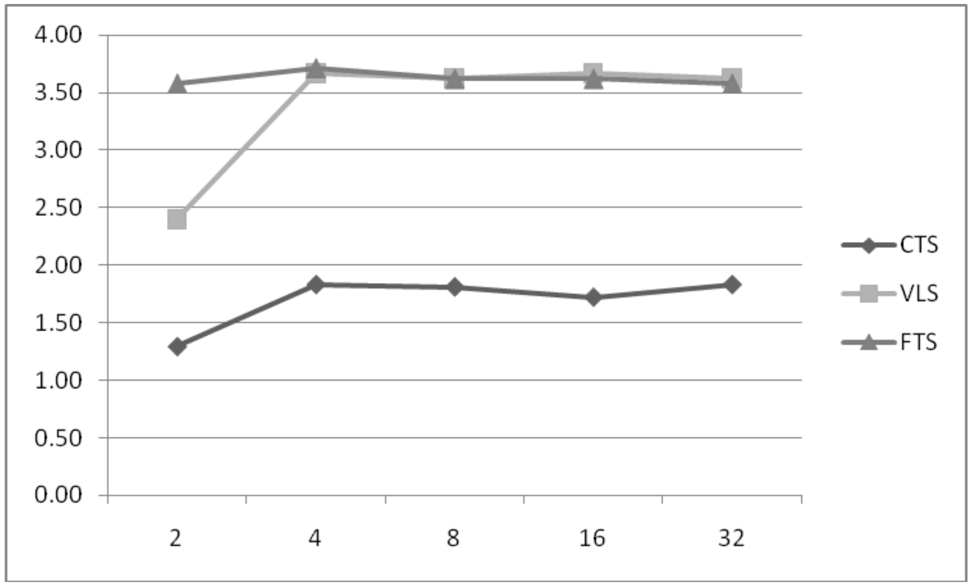

**Figure 3.9** LU decomposition, VL=32, no loop unroll.

For each combination of lane sharing configuration and number of lanes in shown in the tables, three metrics are displayed. The first is performance (Perf.), which is the ratio of the time needed to complete one computational element for a given application on the slowest case to the time needed to complete one computational element for that location in the table. As an example, in Table 3.1 the FTS configuration in the eight lane case performs 2.38 times faster than the CTS configuration in four lanes. The second metric is ALU utilization (ALU Util.), which is a measure of the percentage of the total available capacity of the arithmetic/logic unit used by this application in this VP configuration. A higher value means that the VP is taking advantage of data parallelism and has fewer gaps in the computation pipeline. The third metric is Load/Store utilization (LD/ST Util.), which measures the percentage of the total load/store unit capacity used by this application. As with the ALU utilization, a higher value means that the VP is taking advantage of parallelism in the application but it is instruction parallelism that is being indicated.

# CHAPTER 4

# ANALYSIS OF RESULTS

## 4.1 FFT Application Results

All of the relative performance results shown in the charts are normalized with respect to the four-lane, one MicroBlaze, simple case. This application could not be run on the two-lane version of the design because it requires more than sixteen vector registers with thirty-two elements per lane which cannot be supported because the product of the number elements per lane and number of vector registers must be less than or equal to the available memory locations in the VRF, which is 512. As can be seen from Figures 3.1 and 3.2, the performance grows as the number of lanes increases. However, this growth is not linear due to limitations in keeping the execution pipelines of the vector lanes full.

The performance for the Simple CTS case (the reference for normalization) grows by 73% when the number of lanes increases from four to thirty-two, but that requires an 8x increase in computational resources. The ALU utilization starts at 64.26% for the four-lane case and decreases to 13.88% for the thirty-two-lane case due to the lack of sufficient instruction and data parallelism in the single application. The Load/Store utilization starts at 39.18% for the four-lane case and decreases to 7.24% for the thirty-two-lane case for the same reason.

The performance for the Simple VLS case starts out 23% better than the CTS case, and grows by 158% from the four-lane case to the thirty-two-lane version because of the improvement in resource utilization. The ALU utilization starts at 72.30% for the four-lane case and decreases to 24.93% for the thirty-two-lane case because the two microprocessors cannot keep the pipeline full as the number of available lanes increases. The Load/Store utilization starts at 43.07% for the four-lane case and decreases to 11.11% for the thirty-two-lane case, which is not much better than the CTS case.

The performance for the Simple FTS case starts out 41% better than the CTS case, and grows by 143% from the four-lane case to the thirty-two-lane case. The ALU utilization starts at 88.67% for the four-lane case and decreases to 21.25% for the thirty-two-lane case because even two microprocessors equally sharing the resources cannot achieve high utilization running this application. The Load/Store utilization starts at 53.54% for the four-lane case and decreases to 11.06% for the thirty-two-lane case.

The performance for the Double CTS case starts out 26% better than the Simple CTS case, and grows by 57% when the number of lanes increases from four to thirty-two because of the increase in data parallelism over the Simple CTS case. The ALU utilization starts at 81.52% for the four-lane case and decreases to 18.37% for the thirty-two-lane case for the same reason as the Simple CTS case. The Load/Store utilization starts at 49.07% for the four-lane case and decreases to 9.63% for the thirty-two-lane case, again somewhat better than the Simple CTS case but not a sufficient increase in parallelism to justify thirty-two-lanes for this application.

The performance for the Double VLS case starts out 37% better than the Simple CTS case, and grows by 234% from the four-lane case to the thirty-two-lane version due to increased utilization from two processors. The ALU utilization starts at 72.30% for the four-lane case and decreases to 24.93% for the thirty-two-lane case. The Load/Store utilization starts at 43.07% for the four-lane case and decreases to 11.11% for the thirty-two-lane case.

The performance for the Double FTS case starts out 41% better than the Simple CTS case, and grows by 217% from the four-lane case to the sixteen lane case. There in an anomaly in that the performance improvement for the thirty-two-lane case actually decreases by 1.3% compared with that of the sixteen lane case. This is caused by a large number of the instructions issued to the lane requiring two clock cycles (multiplying a vector quantity by a scalar value which is included as the second 32 bits of the instruction). In this case the pipeline of the lane cannot hide the extra clock because each lane is processing only one element. The ALU utilization starts at 94.09% for the four-lane case and decreases to 25.34% for the thirty-two-lane case. The Load/Store utilization starts at 57.03% for the four-lane case and decreases to 13.16% for the thirty-two-lane case.

The performance increase is best for the FTS sharing case for both the Simple and the Double applications, and the decreased utilization percentages for the higher lane cases indicates that more threads or applications from additional processors could be supported with little decrease in performance. In addition, the performance increases more for the Double version, indicating that applications which more fully utilize the available resources because of improved data parallelism will show a larger payback in results.

## 4.2 FIR Filter Application Results

All of the relative performance results shown for this application are normalized with respect to the two-lane, one MicroBlaze, vector length 32, no loop unrolling case. As can be seen from Figures 3.3, 3.4 and 3.5, the performance improvement grows as the number of lanes increases. As was the case for the FFT application, this improvement is not linear.

The performance for the vector length 32 with no loop unrolling CTS case (the reference for normalization) grows by 113% when the number of lanes increases from two to thirty-two, but that requires a 16x increase in computational resources. The ALU utilization starts at 37.90% for the two-lane case and decreases to 5.03% for the thirty-two-lane case due to a lack of parallelism. The Load/Store utilization starts at 19.82% for the two-lane case and decreases to 2.55% for the thirty-two-lane case for the same reason.

The performance for the vector length 32 with no loop unrolling VLS case starts out 28% better than the CTS case, and grows by 209% from the two-lane case to the thirty-two-lane version due to an increase in data traffic from two separate microprocessors. The ALU utilization starts at 48.43% for the two-lane case and decreases to 9.47% for the thirty-two-lane case due to no increase in data or instruction parallelism. The Load/Store utilization starts at 25.28% for the two-lane case and decreases to 5.11% for the thirty-two-lane case.

The performance for the vector length 32 with no loop unrolling FTS case starts out 95% better than the CTS case, and grows by 118% from the two-lane case to the thirty-two-lane case because the two microprocessors are sharing the available resources equally which improves the utilization. The ALU utilization starts at 73.91% for the two-lane case and decreases to 10.07% for the thirty-two-lane case which is better than the CTS case but still shows improvements in the application are possible. The Load/Store utilization starts at 38.68% for the two-lane case and decreases to 4.76% for the thirty-two-lane case.

The performance for the vector length 64 with no loop unrolling CTS case starts out 28% better than the vector length 32 with no loop unrolling CTS case, and grows by 209% when the number of lanes increases from two to thirty-two due to the increased data parallelism from the doubling of the vector length. The ALU utilization starts at 49.35% for the two-lane case and decreases to 9.37% for the thirty-two-lane case. The Load/Store utilization starts at 25.86% for the two-lane case and decreases to 4.76% for the thirty-two-lane case.

The performance for the vector length 64 with no loop unrolling VLS case starts out 48% better than the vector length 32 with no loop unrolling CTS case, and grows by 369% from the two-lane case to the thirty-two-lane version because of the combination of improved parallelism and dividing the lanes between two microprocessors. The ALU utilization starts at 56.11% for the two-lane case and decreases to 16.45% for the thirty-two-lane case. The Load/Store utilization starts at 28.82% for the two-lane case and decreases to 8.36% for the thirty-two-lane case.

The performance for the vector length 64 with no loop unrolling FTS case starts out 124% better than the vector length 32 with no loop unrolling CTS case, and grows by 253% from the two-lane case to the thirty-two-lane case, again because of increased parallelism and because of the sharing configuration. The ALU utilization starts at 84.92% for the two-lane case and decreases to 18.74% for the thirty-two-lane case. The

Load/Store utilization starts at 44.25% for the two-lane case and decreases to 9.51% for the thirty-two-lane case.

The performance for the vector length 64 with three times loop unrolling CTS case starts out 136% better than the vector length 32 with no loop unrolling CTS case, and grows by 267% when the number of lanes increases from two to thirty-two because now instruction parallelism has been increased by unrolling the loop. The ALU utilization starts at 89.50% for the two-lane case and decreases to 20.49% for the thirty-two-lane case. The Load/Store utilization starts at 46.66% for the two-lane case and decreases to 10.41% for the thirty-two-lane case.

The performance for the vector length 64 with three times loop unrolling VLS case starts out 147% better than the vector length 32 with no loop unrolling CTS case, and grows by 515% from the two-lane case to the thirty-two-lane version, again because of improved parallelism and dividing the lanes between two processors. The ALU utilization starts at 93.36% for the two-lane case and decreases to 35.99% for the thirty-two-lane case. The Load/Store utilization starts at 48.60% for the two-lane case and decreases to 20.58% for the thirty-two-lane case.

The performance for the vector length 64 with three times loop unrolling FTS case starts out 161% better than the vector length 32 with no loop unrolling CTS case, and grows by 554% from the two-lane case to the thirty-two-lane case due to data and instruction parallelism and improved sharing of resources between the two microprocessors. The ALU utilization starts at 99.85% for the two-lane case and decreases to 40.45% for the thirty-two-lane case which shows that it is possible to keep the pipeline nearly completely full on the ALU side. The Load/Store utilization starts at

51.45% for the two-lane case and decreases to 18.30% for the thirty-two-lane case in part because this application has about one half the Load/Store utilization compared with the ALU utilization for each case.

As with the FFT application above, the performance increase is best for the FTS sharing case for all three applications, and the decreased utilization percentages for the higher lane cases indicates that more threads or applications from additional processors could be supported with little decrease in performance. In addition, performance increases significantly for the vector length 64 version, and even more for the three times loop unrolling version, indicating that designing applications to exhibit more parallelism results in large performance gains.

## 4.3 FIR Filter MADD Results

The normalization of performance results for this section match that of the FIR Filter. The addition of the combined Multiply/Add function does result in significant performance increases, as can be seen in Figures 3.6, 3.7 and 3.8.

The performance for the vector length 32 with no loop unrolling CTS MADD case starts out 46% better than the vector length 32 with no loop unrolling CTS case, and grows by 203% from the two-lane case to the thirty-two-lane case because the replacement of two separate instructions (one multiply and one add/subtract) with one instruction balances the utilization between the ALU and the Load/Store units. The relative increase in performance for each lane configuration remains almost the same as well, decreasing from 46% for the two-lane case to 42% for the thirty-two-lane case. The ALU utilization starts at 28.11% for the two-lane case and decreases to 3.64% for the thirty-two-lane case. The Load/Store utilization starts at 28.08% for the two-lane case and decreases to 3.64% for the thirty-two-lane case.

The performance for the vector length 32 with no loop unrolling VLS MADD case starts out 87% better than the vector length 32 with no loop unrolling CTS case, and grows by 202% from the two-lane case to the thirty-two-lane case. The relative increase in performance for each lane configuration remains almost the same as well, decreasing from 46% for the two-lane case to 43% for the thirty-two-lane case when compared with the vector length 32 with no loop unrolling VLS case. The ALU utilization starts at 35.97% for the two-lane case and decreases to 6.79% for the thirty-two-lane case. The Load/Store utilization starts at 36.13% for the two-lane case and decreases to 6.79% for the thirty-two-lane case.

The performance for the vector length 32 with no loop unrolling FTS MADD case starts out 191% better than the vector length 32 with no loop unrolling CTS case, and grows by 108% from the two-lane case to the thirty-two-lane case. The relative increase in performance for each lane configuration remains almost the same as well, decreasing from 49% for the two-lane case to 43% for the thirty-two-lane case when compared with the vector length 32 with no loop unrolling FTS case. The ALU utilization starts at 55.98% for the two-lane case and decreases to 7.28% for the thirty-two-lane case. The Load/Store utilization starts at 55.92% for the two-lane case and decreases to 7.28% for the thirty-two-lane case.

In all three cases, the change in the application to take advantage of the additional MADD function decreases the ALU utilization relative to the case without the MADD function and increases the Load/Store utilization until both percentages are approximately equal.

The performance for the vector length 32 with three times loop unrolling CTS MADD case starts out 270% better than the vector length 32 with no loop unrolling CTS case, and grows by 74% from the two-lane case to the thirty-two-lane case because of the increased instruction parallelism arising from loop unrolling. The relative increase in performance for each lane configuration does not remain the same, decreasing from 71% for the two-lane case to 39% for the thirty-two-lane case when compared with the vector length 32 with three times loop unrolling CTS case because the boost from the added instruction combined with the boost from loop unrolling becomes less effective as the number of lanes increases. This is shown by the large decreases in utilization for both the ALU and the Load/Store units. The ALU utilization starts at 83.82% for the two-lane case and decreases to 9.13% for the thirty-two-lane case. The Load/Store utilization starts at 71.19% for the two-lane case and decreases to 7.74% for the thirty-two-lane case.

The performance for the vector length 32 with three times loop unrolling VLS MADD case starts out 311% better than the vector length 32 with no loop unrolling CTS case, and grows by 213% from the two-lane case to the sixteen lane case. The performance remains constant for sixteen and thirty-two-lanes. The relative increase in performance for each lane configuration does not remain the same, decreasing from 74% for the two-lane case to 49% for the thirty-two-lane case when compared with the vector length 32 with three times loop unrolling VLS case. The ALU utilization starts at 92.86% for the two-lane case and decreases to 18.26% for the thirty-two-lane case. The

Load/Store utilization starts at 78.79% for the two-lane case and decreases to 15.47% for the thirty-two-lane case.

The performance for the vector length 32 with three times loop unrolling FTS MADD case starts out 321% better than the vector length 32 with no loop unrolling CTS case, and grows by 205% from the two-lane case to the eight lane case. The performance remains flat from eight to thirty-two-lanes. The relative increase in performance for each lane configuration does not remain the same, increasing from 60% for the two-lane case to 73% for the eight lane case and then decreasing to 39% for the thirty-two-lane case when compared with the vector length 32 with three times loop unrolling VLS case. The ALU utilization starts at 97.76% for the two-lane case and decreases to 18.26% for the thirty-two-lane case. The Load/Store utilization starts at 83.24% for the two-lane case and decreases to 15.46% for the thirty-two-lane case.

In all three cases, the change in the application to take advantage of the additional MADD function increases the ALU utilization relative to the case without the MADD function and increases the Load/Store utilization but both percentages remain unequal. This is the explanation for the tailing off of the performance increases as the number of lanes is increased.

The performance for the vector length 128 with three times loop unrolling CTS MADD case starts out 723% better than the vector length 32 with no loop unrolling CTS case, and grows by 213% from the four-lane case to the thirty-two-lane case due to the combination of increased data and instruction parallelism. The relative increase in performance for each lane configuration does not remain the same, increasing from 74% for the four-lane case to 90% for the sixteen lane case and then decreasing to 68% for the thirty-two-lane case when compared with the vector length 128 with three times loop unrolling CTS case. The ALU utilization starts at 93.52% for the four-lane case and decreases to 36.48% for the thirty-two-lane case. The Load/Store utilization starts at 79.42% for the four-lane case and decreases to 31.58% for the thirty-two-lane case. These are still relatively high utilization numbers even for the thirty-two-lane case, indicating the increased performance from improved parallelism in the application.

The performance for the vector length 128 with three times loop unrolling VLS MADD case starts out 759% better than the vector length 32 with no loop unrolling CTS case, and grows by 420% from the four-lane case to the thirty-two-lane case. The relative increase in performance for each lane configuration does not remain the same, decreasing from 74% for the four-lane case to 62% for the sixteen lane case and then increasing to 82% for the thirty-two-lane case when compared with the vector length 128 with three times loop unrolling VLS case. The ALU utilization starts at 97.42% for the four-lane case and decreases to 64.69% for the thirty-two-lane case. The Load/Store utilization starts at 82.84% for the four-lane case and decreases to 55.21% for the thirty-two-lane case. These are high utilization numbers up through the thirty-two-lane case which explains the large performance increases.

The performance for the vector length 128 with three times loop unrolling FTS MADD case starts out 791% better than the vector length 32 with no loop unrolling CTS case, and grows by 477% from the two-lane case to the thirty-two-lane case. The relative increase in performance for each lane configuration does not remain the same, decreasing from 71% for the four-lane case to 59% for the eight lane case and increasing to 75% for the thirty-two-lane case when compared with the vector length 128 with three times loop unrolling VLS case. The ALU utilization starts at 98.85% for the four-lane case and decreases to 60.83% for the thirty-two-lane case. The Load/Store utilization starts at 83.61% for the four-lane case and decreases to 51.53% for the thirty-two-lane case.

In all three cases, the change in the application to take advantage of the additional MADD function increases the ALU utilization relative to the case without the MADD function and increases the Load/Store utilization but both percentages remain unequal. This is the explanation for the tailing off of the performance increases as the number of lanes is increased.

## 4.4 MM Application Results

All of the relative performance results shown in the charts are normalized with respect to the two-lane, one MicroBlaze, vector length 32, unroll the loop one time case. As can be seen from Figures 3.9 and 3.10, the performance grows as the number of lanes increases. However, this growth is not linear due to limitations in keeping the execution pipelines of the vector lanes full.

The performance for the vector length 32 one time unroll CTS case (the reference for normalization) grows by 120% when the number of lanes increases from two to thirty-two. The ALU utilization starts at 71.10% for the two-lane case and decreases to 9.80% for the thirty-two-lane case. The Load/Store utilization starts at 72.21% for the two-lane case and decreases to 9.96% for the thirty-two-lane case.

The performance for the vector length 32 one time unroll VLS case starts out 17% better than the CTS case, and grows by 257% from the two-lane case to the thirty-two-lane version. The ALU utilization starts at 82.98% for the two-lane case and decreases to

18.58% for the thirty-two-lane case. The Load/Store utilization starts at 84.63% for the two-lane case and decreases to 19.38% for the thirty-two-lane case.

The performance for the vector length 32 one time unroll FTS case starts out 38% better than the CTS case, and grows by 216% from the two-lane case to the thirty-two-lane case. The ALU utilization starts at 98.13% for the two-lane case and decreases to 19.27% for the thirty-two-lane case. The Load/Store utilization starts at 99.87% for the two-lane case and decreases to 19.59% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll CTS case starts out 16% better than the vector length 32 one time unroll CTS case, and grows by 240% when the number of lanes increases from two to thirty-two. The ALU utilization starts at 82.59% for the two-lane case and decreases to 17.61% for the thirty-two-lane case. The Load/Store utilization starts at 84.29% for the two-lane case and decreases to 17.91% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll VLS case starts out 28% better than the vector length 32 one time unroll CTS case, and grows by 444% from the two-lane case to the thirty-two-lane version. The ALU utilization starts at 90.72% for the two-lane case and decreases to 31.16% for the thirty-two-lane case. The Load/Store utilization starts at 92.14% for the two-lane case and decreases to 31.68% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll FTS case starts out 38% better than the vector length 32 one time unroll CTS case, and grows by 468% from the two-lane case to the thirty-two-lane case. The ALU utilization starts at 98.35% for the four-lane case and decreases to 34.97% for the thirty-two-lane case. The Load/Store utilization starts at 100.00% for the four-lane case and decreases to 35.52% for the thirty-two-lane case.

The performance increase is best for the FTS sharing case for both vector lengths, and the decreased utilization percentages for the higher lane cases indicates that more threads or applications from additional processors could be supported with little decrease in performance. In addition, the performance increases more for the version with the larger vector length due to increased data parallelism. The Load/Store utilization is slightly higher than the ALU utilization, indicating that this application relies more heavily on the Load/Store unit than on the ALU.

## 4.5 MM MADD Results

The normalization of performance results for this section match that of the Matrix Multiplication application. As can be seen from Figures 3.11 and 3.12, the performance grows as the number of lanes increases. However, the addition of the MADD unit does not provide as significant performance gains as it did in the case of the FIR application.

The performance for the vector length 32 one time unroll CTS MADD case starts out 3% worse than the CTS case, and grows by 164% when the number of lanes increases from two to thirty-two. This is due to the decreased ALU utilization (almost by a half) when compared with the CTS case. The ALU utilization starts at 34.45% for the two-lane case and decreases to 5.73% for the thirty-two-lane case. The Load/Store utilization starts at 69.93% for the two-lane case and decreases to 11.65% for the thirty-two-lane case, which is almost the same as the utilization for the CTS case (slightly lower for two-lanes and higher by 2% for thirty-two-lanes).

The performance for the vector length 32 one time unroll VLS MADD case starts out 15% better than the CTS case, and grows by 322% from the two-lane case to the thirty-two-lane version, again starting slightly worse for two-lanes versus the VLS case and improving up through the thirty-two-lane case. The ALU utilization starts at 40.79% for the two-lane case and decreases to 10.79% for the thirty-two-lane case. The Load/Store utilization starts at 82.97% for the two-lane case and decreases to 21.94% for the thirty-two-lane case.

The performance for the vector length 32 one time unroll FTS MADD case starts out 38% better than the CTS case, and grows by 266% from the two-lane case to the thirty-two-lane case, showing no performance change for two-lanes compared with the FTS case and improving up through the thirty-two-lane case. The ALU utilization starts at 49.00% for the two-lane case and decreases to 11.17% for the thirty-two-lane case. The Load/Store utilization starts at 99.85% for the two-lane case and decreases to 22.70% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll CTS MADD case starts out 14% better than the vector length 32 one time unroll CTS case, and grows by 302% when the number of lanes increases from two to thirty-two. This is worse performance by 2% at two-lanes compared with the vector length 64 one time unroll CTS case, but shows improvement over the thirty-two-lane case. The ALU utilization starts at 40.76% for the two-lane case and decreases to 10.24% for the thirty-two-lane case. The Load/Store utilization starts at 82.75% for the two-lane case and decreases to 20.78% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll VLS MADD case starts out 26% better than the vector length 32 one time unroll CTS case, and grows by 501% from the two-lane case to the thirty-two-lane version, again showing a slight (2%) decrease in performance relative to the same configuration without the MADD at two-lanes but ultimately showing an increase in performance at thirty-two-lanes. The ALU utilization starts at 44.71% for the two-lane case and decreases to 16.76% for the thirty-two-lane case. The Load/Store utilization starts at 91.41% for the two-lane case and decreases to 34.05% for the thirty-two-lane case.

The performance for the vector length 64 one time unroll FTS MADD case starts out 39% better than the vector length 32 one time unroll CTS case, and grows by 556% from the two-lane case to the thirty-two-lane case. This time, the two-lane case starts with a slight (1%) increase when compared with the vector length 64 one time unroll FTS application and shows improved performance through thirty-two-lanes. The ALU utilization starts at 49.08% for the four-lane case and decreases to 19.89% for the thirty-two-lane case. The Load/Store utilization starts at 100.00% for the four-lane case and decreases to 40.50% for the thirty-two-lane case.

In both cases, the change in the application to take advantage of the additional MADD function decreases the ALU utilization relative to the case without the MADD function by about half and increases the Load/Store utilization slightly. This makes the two percentages unequal.

## 4.6 LU Decomposition Application Results

All of the relative performance results shown for this application are normalized with respect to the two-lane, one MicroBlaze, vector length 64 case. As can be seen from Figures 3.13 and 3.14, the performance improvement grows as the number of lanes increases up to the four-lane version for most cases (eight lanes for the vector length 64 VLS case). This is because the limiting factor in this application is the number of floating point divides required for each pass through the processing loop. These divisions are done by the MicroBlaze rather than by the lane and take either 28 or 30 clock cycles to complete, depending on the optimization used during synthesis of the microprocessor.

The performance for the vector length 64 no unroll CTS case grows by 83% when the number of lanes increases from two to four and is basically constant up to thirty-two-lanes. The ALU utilization starts at 53.20% for the two-lane case and decreases to 5.03% for the thirty-two-lane case. The Load/Store utilization starts at 54.24% for the two-lane case and decreases to 5.10% for the thirty-two-lane case.

The performance for the vector length 64 no unroll VLS case starts out 33% better than the CTS case, grows by 176% from the two-lane case to the eight lane version and is constant from then on. The ALU utilization starts at 59.02% for the two-lane case and decreases to 9.97% for the thirty-two-lane case. The Load/Store utilization starts at 60.49% for the two-lane case and decreases to 10.25% for the thirty-two-lane case.

The performance for the vector length 64 no unroll FTS case starts out 75% better than the CTS case, and grows by 107% from the two-lane case to the four-lane case and is constant from then on. The ALU utilization starts at 90.19% for the two-lane case and decreases to 10.00% for the thirty-two-lane case. The Load/Store utilization starts at 93.38% for the two-lane case and decreases to 10.32% for the thirty-two-lane case.

The performance for the vector length 32 no unroll CTS case starts out 29% better than the vector length 64 no unroll CTS case, grows by 42% when the number of lanes increases from two to four, and remains basically constant from then on. The ALU utilization starts at 39.84% for the two-lane case and decreases to 2.49% for the thirty-two-lane case. The Load/Store utilization starts at 41.31% for the two-lane case and decreases to 2.58% for the thirty-two-lane case.

The performance for the vector length 32 no unroll VLS case starts out 140% better than the vector length 64 no unroll CTS case, and grows by 53% from the two-lane case to the four-lane version with no increase from then on. The ALU utilization starts at 52.28% for the two-lane case and decreases to 4.77% for the thirty-two-lane case. The Load/Store utilization starts at 56.25% for the two-lane case and decreases to 5.24% for the thirty-two-lane case.

The performance for the vector length 32 no unroll FTS case starts out 258% better than the vector length 64 no unroll CTS case, and grows by 4% from the two-lane case to the four-lane case with no additional increase beyond that. The ALU utilization starts at 76.57% for the four-lane case and decreases to 4.84% for the thirty-two-lane case. The Load/Store utilization starts at 84.82% for the four-lane case and decreases to 5.35% for the thirty-two-lane case.

## 4.7 LU Decomposition DIV Results

The normalization of performance results for this section match that of the LU Decomposition. As can be seen from Figures 3.15 and 3.16, the performance improvement grows as the number of lanes increases in a similar curve to the rest of the applications described above. This is because the floating point divide functions have been removed from the MicroBlaze and are performed in the lane. However, there are a limited number of these divides so the performance increase in the VP is not large. Overall application performance increases significantly because of removing the MicroBlaze bottleneck from the application, replacing an un-pipelined 28 or 30 clock latency divide function with a pipelined 6 clock latency function in the lane, but that performance is not measured in this thesis. The two-lane cases for the two sharing configurations (CTS and FTS) and all the lane cases for the VLS configuration could not be run due to an architectural limitation related to the number of available vector registers, similar to the issue with the FFT application, above.

The performance for the four-lane vector length 64 no unroll CTS DIV case starts out 86% better than the two-lane vector length 64 no unroll CTS case and grows by 60% when the number of lanes increases from four to thirty-two. The ALU utilization starts at 42.96% for the four-lane case and decreases to 10.27% for the thirty-two-lane case. The Load/Store utilization starts at 44.21% for the four-lane case and decreases to 10.49% for the thirty-two-lane case.

The performance for the four-lane vector length 64 no unroll FTS DIV case starts out 225% better than the two-lane CTS case and grows by 76% from the four-lane case to the thirty-two-lane case. This is lower performance for the four-lane case than the FTS configuration without DIV, but improvement occurs at the eight lane case and continues through the thirty-two-lane case. The ALU utilization starts at 72.68% for the four-lane case and decreases to 19.61% for the thirty-two-lane case. The Load/Store utilization starts at 75.23% for the four-lane case and decreases to 20.09% for the thirty-two-lane case.

The performance for the four-lane vector length 32 no unroll CTS DIV case starts out 134% better than the two-lane vector length 64 no unroll CTS case and grows by 30% when the number of lanes increases from four to thirty-two. The ALU utilization starts at 25.74% for the two-lane case and decreases to 4.47% for the thirty-two-lane case. The Load/Store utilization starts at 27.80% for the two-lane case and decreases to 4.86% for the thirty-two-lane case.

The performance for the four-lane vector length 32 no unroll FTS DIV case starts out 409% better than the two-lane vector length 64 no unroll CTS case and grows by 46% from the four-lane case to the thirty-two-lane case. The ALU utilization starts at 51.28% for the four-lane case and decreases to 9.89% for the thirty-two-lane case. The Load/Store utilization starts at 54.79% for the four-lane case and decreases to 10.61% for the thirty-two-lane case.

In all four cases with the DIV function, the utilization percentages for both the ALU and Load/Store units are approximately equal and are from 33% (for four-lanes) to 104% (for thirty-two-lanes) higher than the equivalent run without the DIV function. This is the explanation for the performance improvements.

## 4.8 FPGA Resource Utilization

The Xilinx synthesis tool was run on ten of the configurations (see Table 4.1 below): once for each number of lanes without the addition of the RM and once for each number of lanes with the RM included and configured for the MADD function. The DIV function is slightly smaller (approximately 100 slice registers fewer than the 980 required for the MADD function) so it did not make sense to run this configuration through the tool. As can be seen from the table, the amount of resources used increased nearly linearly, approximately doubling for each increase in the number of lanes. The growth varies from a low of 72.6% between the two and four-lane cases without RM to 119.5% between the sixteen and thirty-two-lane cases without RM. The number of BRAMs does not increase the same way for the two and four-lane cases because the minimum number of RAMs in the VM is eight. The numbers in the table do not include the resources required for the MicroBlaze processors or the associated program memory and related logic.

**Table 4.1** FPGA Synthesis Results

| Logic Utilization in XC5VLX110T | 2-Lane w/ MADD | 2-Lane | 4-Lane w/ MADD | 4-Lane | 8-Lane w/ MADD | 8-Lane | 16-Lane w/ MADD | 16-Lane | 32-Lane w/ MADD | 32-Lane | Available Resources |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of Slice Registers | 10495 | 8575 | 18703 | 14798 | 35101 | 27308 | 66306 | 50607 | 142248 | 111112 | 69120 |
| Number of Slice LUTs | 8391 | 5911 | 15099 | 10164 | 28891 | 19058 | 60690 | 41030 | 151899 | 112603 | 69120 |
| Number of fully used LUT-FF pairs | 4548 | 3290 | 8121 | 5516 | 15577 | 10456 | 28979 | 18411 | 67258 | 46489 | 11196 |
| Number of bonded IOBs | 230 | 230 | 230 | 230 | 230 | 230 | 231 | 231 | 232 | 232 | 680 |
| Number of Block RAM/FIFO | 18 | 18 | 20 | 20 | 24 | 24 | 48 | 48 | 96 | 96 | 148 |
| Number of BUFG/BUFGCTRLs | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 32 |
| Number of DCM_ADVs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 12 |
| Number of DSP48Es | 6 | 6 | 12 | 12 | 24 | 24 | 48 | 48 | 96 | 96 | 64 |
| Frequency (in MHz) | 225.2 | 229.4 | 228.8 | 228.4 | 220.4 | 220.4 | 175.4 | 174.8 | 145.1 | 145.1 | |

## 4.9 FPGA Power Consumption

The dynamic power consumption of each lane configuration is estimated based on a formula generated empirically by Spiridon Beldianu and discussed in a forthcoming paper. The formula, shown below, calculates the dynamic power dissipated during the active operation of the lanes and is based on the utilization of both the arithmetic/logic unit and the load/store unit. In addition, the formula includes a factor for the vector memory and crossbar units. From this power value, the dynamic energy use for each computational element can be calculated by multiplying the power by the time to complete one element.

$$P^{d}_{TOTAL} \cong M\left[\left(K_{ALU} + K_{VRF}/2\right)U_{ALU} + \left(K_{LDST} + K_{VRF}/2\right)U_{LDST}\right] + K_{MC\_VM} \cdot U_{LDST} \qquad (1)$$

Where:

$P^{d}_{TOTAL}$ is the total dynamic power

$M$ is the number of lanes

$K_{ALU}$ is a constant for the ALU equal to 0.3723 mW/%, 0.4739mW/% for MADD and DIV cases

$K_{LDST}$ is a constant for the LD/ST equal to 0.0967 mW/%

$K_{VRF}$ is a constant for the VRF equal to 0.2818 mW/%

$K_{MC\_VM}$ is a constant for the MC/VM equal to 1.5197 mW/% (2, 4 and 8 lanes), 3.0394 mW/% (16 lanes), and 6.0788 mW/% (32 lanes)

$U_{ALU}$ is the percent utilization of the ALU for a particular scenario

$U_{LDST}$ is the percent utilization of the LD/ST for a particular scenario

As can be seen in Table 4.2 the dynamic power increases as the number of lanes increases and as the sharing configuration changes from CTS to VLS and to FTS. More interesting is the dynamic energy required for computation of one element for the FIR 32 application. For a constant number of lanes, the dynamic energy per element is approximately constant for the same lane architecture (without or with RM). It decreases by about 19% from two to four-lanes and about 13% from four to eight lanes. In this application, the dynamic energy decreases significantly for the RM architecture when compared with the case without the RM. The decrease is approximately 25% for two-lanes, 31% for four-lanes, and 35% for eight, sixteen and thirty-two-lanes. Also interesting to note is that the dynamic energy per element stops decreasing as the number of lanes increases beyond eight. This is due to the application reaching the maximum possible level of parallelism, so an increased number of available lanes does not improve the energy used.

In Table 4.3 the dynamic power and energy usage is compared for the MM scenarios. The energy per computational element values are much larger than those for the FIR scenarios because one element for MM is the calculation of an entire row, rather than the calculation of a single filter value for FIR. The power increases as the number of lanes increases and as the sharing configuration changes. As with the FIR scenarios, the dynamic energy values are basically constant for a set number of lanes. It decreases by approximately 26% from two to four-lanes and about 17% from four to eight lanes. Again, the RM architecture provides a significant improvement in dynamic energy usage, from about 17% for two-lanes to about 22% for four-lanes and about 27% for eight, sixteen and thirty-two-lanes.

Table 4.4 shows the comparison of power and energy usage for the LU Decomposition scenarios. Of most interest is the increase of dynamic energy required per computational element for the RM architecture over the version without the additional module. This is due to moving the divide function from the MicroBlaze to the lanes, which increases the total amount of work done by the lanes. In this case, a fair comparison of efficiency would have to include the power and energy used by the microprocessor to carry out the floating point divide functions.

**Table 4.2** FIR Filter Dynamic Power Results

| | No. of Lanes | 2 | | 4 | | 8 | | 16 | | 32 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) |
| | CTS | 78.4 | 53.8 | 88.2 | 43.5 | 96.0 | 38.0 | 109.4 | 38.0 | 117.6 | 37.9 |
| FIR 32 | CTS - MADD | 90.6 | 42.6 | 95.6 | 32.6 | 99.9 | 27.5 | 113.2 | 27.5 | 121.3 | 27.5 |
| VL=32 | VLS | 100.1 | 53.8 | 125.0 | 43.0 | 154.3 | 38.0 | 192.2 | 38.0 | 218.9 | 38.0 |
| No Unroll | VLS - MADD | 116.3 | 42.7 | 138.6 | 32.6 | 161.7 | 27.5 | 199.9 | 27.5 | 226.5 | 27.5 |
| | FTS | 153.0 | 53.8 | 173.8 | 43.1 | 192.1 | 38.0 | 218.8 | 38.0 | 235.2 | 38.0 |
| | FTS - MADD | 180.3 | 42.5 | 190.2 | 32.5 | 199.6 | 27.5 | 226.1 | 27.5 | 242.7 | 27.5 |
| | CTS | 169.6 | 53.9 | 196.2 | 43.5 | 211.2 | 38.0 | 239.3 | 38.0 | 256.1 | 38.0 |
| FIR 32 | CTS - MADD | 245.1 | 45.5 | 299.9 | 35.5 | 285.6 | 30.5 | 285.5 | 30.5 | 285.4 | 30.5 |
| VL=32 | VLS | 185.0 | 53.8 | 274.1 | 42.8 | 342.6 | 38.0 | 422.8 | 38.3 | 478.9 | 38.0 |
| Unroll 3 Times | VLS - MADD | 271.3 | 45.3 | 385.7 | 35.5 | 513.8 | 31.2 | 571.8 | 30.5 | 570.8 | 30.5 |
| | FTS | 203.3 | 53.0 | 324.5 | 44.0 | 410.3 | 37.9 | 478.3 | 38.0 | 512.0 | 38.0 |
| | FTS - MADD | 286.4 | 46.8 | 426.8 | 35.6 | 570.5 | 30.4 | 570.8 | 30.5 | 570.6 | 30.5 |
| | CTS | | | 297.0 | 43.2 | 478.3 | 38.0 | 685.1 | 38.0 | 845.6 | 38.0 |
| FIR 32 | CTS - MADD | | | 426.0 | 35.5 | 664.0 | 30.5 | 1050.8 | 30.5 | 1142.6 | 30.5 |
| VL=128 | VLS | | | 312.3 | 43.3 | 521.9 | 38.0 | 953.2 | 38.0 | 1367.1 | 38.3 |
| Unroll 3 Times | VLS - MADD | | | 444.1 | 35.5 | 713.7 | 30.5 | 1248.6 | 30.7 | 2027.3 | 31.2 |
| | FTS | | | 331.9 | 43.6 | 582.9 | 37.8 | 1112.9 | 37.9 | 1639.1 | 38.3 |
| | FTS - MADD | | | 449.5 | 34.6 | 766.4 | 31.3 | 1377.7 | 28.8 | 1901.0 | 25.4 |

**Table 4.3** MM Dynamic Power Results

| | No. of Lanes | 2 | | 4 | | 8 | | 16 | | 32 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) |
| MM VL=32 Unroll 1 Time | CTS | 217.0 | 2503.2 | 238.2 | 1864.3 | 257.6 | 1552.1 | 282.1 | 1554.0 | 297.0 | 1560.1 |
| | CTS - MADD | 181.8 | 2160.3 | 196.6 | 1529.9 | 212.3 | 1219.3 | 257.8 | 1218.6 | 272.1 | 1224.0 |
| | VLS | 253.9 | 2508.2 | 324.4 | 1871.9 | 406.4 | 1540.9 | 497.5 | 1564.2 | 570.0 | 1571.8 |
| | VLS - MADD | 215.6 | 2166.0 | 257.2 | 1531.3 | 308.6 | 1216.3 | 420.2 | 1216.2 | 512.3 | 1219.5 |
| | FTS | 299.9 | 2507.0 | 405.2 | 1878.9 | 509.3 | 1552.3 | 554.1 | 1567.1 | 584.3 | 1547.4 |
| | FTS - MADD | 259.4 | 2167.5 | 359.9 | 1542.3 | 420.8 | 1212.0 | 514.3 | 1232.6 | 530.1 | 1211.1 |
| MM VL=64 Unroll 1 Time | CTS | 252.8 | 2516.1 | 315.4 | 1859.5 | 380.1 | 1544.0 | 469.6 | 1556.3 | 534.0 | 1563.7 |
| | CTS - MADD | 214.1 | 2161.8 | 252.3 | 1526.2 | 300.8 | 1211.5 | 401.9 | 1212.8 | 485.6 | 1224.0 |
| | VLS | 276.9 | 2502.0 | 376.8 | 1875.0 | 534.5 | 1558.6 | 758.7 | 1546.9 | 944.7 | 1564.9 |
| | VLS - MADD | 237.3 | 2165.7 | 303.6 | 1532.7 | 401.9 | 1226.4 | 592.7 | 1211.0 | 795.2 | 1211.3 |
| | FTS | 300.9 | 2508.4 | 446.3 | 1872.7 | 667.9 | 1553.9 | 924.7 | 1558.0 | 1059.7 | 1558.3 |
| | FTS - MADD | 260.5 | 2169.9 | 365.6 | 1532.6 | 551.1 | 1217.4 | 798.3 | 12109.1 | 945.1 | 1196.1 |

**Table 4.4** LU Decomposition Dynamic Power Results

| | No. of Lanes | 4 | | 8 | | 16 | | 32 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) | Power (in mW) | Energy (in nJ) |
| LU Decomp VL=64 No Unrolls | CTS | 183.6 | 114.5 | 152.3 | 96.3 | 152.3 | 106.0 | 152.3 | 95.1 |
| | CTS - DIV | 214.8 | 132.3 | 262.5 | 128.1 | 311.2 | 131.9 | 345.4 | 135.4 |
| | VLS | 244.3 | 116.3 | 303.5 | 94.7 | 303.2 | 94.6 | 303.8 | 94.8 |
| | FTS | 367.1 | 117.5 | 305.2 | 96.4 | 305.2 | 96.5 | 305.2 | 96.4 |
| | FTS - DIV | 364.5 | 128.3 | 499.4 | 123.9 | 597.9 | 126.8 | 660.4 | 132.1 |
| LU Decomp VL=32 No Unrolls | CTS | 91.7 | 57.2 | 75.8 | 47.9 | 76.1 | 50.5 | 76.1 | 47.5 |
| | CTS - DIV | 131.9 | 64.4 | 134.4 | 57.0 | 153.6 | 60.2 | 154.4 | 58.0 |
| | VLS | 182.3 | 56.9 | 150.1 | 47.4 | 150.1 | 46.8 | 150.1 | 47.4 |
| | FTS | 185.2 | 57.0 | 149.1 | 47.1 | 152.6 | 48.2 | 152.5 | 48.8 |
| | FTS - DIV | 261.4 | 73.2 | 261.7 | 55.5 | 305.5 | 61.1 | 339.6 | 65.2 |

# CHAPTER 5

# CONCLUSIONS

This thesis presents a shared vector coprocessor bank comprised of multiple vector lanes. Three sharing configurations, Coarse-grained Temporal Sharing, Vector Lane Sharing and Fine-grained Temporal Sharing, were investigated to determine the possible improvements in both performance and energy efficiency. In addition, five different numbers of lanes (two, four, eight, sixteen and thirty-two) were also evaluated. Finally, an additional Reconfigurable Module was added to each lane and configured to best support the benchmarking application currently being run to determine possible improvements from this feature.

It was shown that because of the increased utilization of the lanes, FTS sharing provided the greatest improvement, followed by VLS. It was also shown that while adding the RM for the FIR Filter application provided significant improvement in both performance and energy usage, the same function in the MM application only provided modest performance improvement along with a similarly better dynamic energy usage. Finally, the addition of the RM to the LU Decomposition application provided a minor performance increase but also increased the dynamic energy per element because of moving the divide function from the microprocessor to the lanes.

Increases in instruction parallelism from loop unrolling and data parallelism from longer vector lengths were shown in the analysis to improve performance by a larger margin than an increased number of lanes alone while not causing an increase in energy per element usage. This is due to the larger utilization percentage of both the Load/Store and Arithmetic/Logic units in each lane. The impact of changing the sharing configuration and thus increasing utilization was larger than that of improved parallelism for a given application, showing that FTS followed by VLS is a better way to improve performance than changing the application.

Future work will focus on alternative functions for the RM such as the floating point square root, as well as continued improvements to the scheduling and sharing portions of the design with the goal of dynamically optimizing either performance or energy usage depending on the current operating conditions. In addition, priority will be included in the scheduling function so that a higher priority task or thread can preempt a lower priority one either from the same microprocessor or from another attached microprocessor. Finally, simulation of the implementation in an ASIC environment rather than a Xilinx FPGA architecture will be investigated to see if any improvements can be made by increasing the number of read and write ports on the Vector Register File, increasing the data bus size from 32 to 64 bits, or adding the capability to perform double precision operations, as examples.

# APPENDIX A

# FIR FILTER MADD BENCHMARKS

The three charts and figures for the FIR Filter with MADD RM scenarios will be found in this appendix.

**Table A.1** FIR 32, VL=32, No Loop Unroll, MADD

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIR 32 | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.00 | 37.90 | 19.82 | 1.39 | 26.39 | 13.80 | 1.74 | 16.34 | 8.44 | 1.98 | 9.37 | 4.75 | 2.13 | 5.03 | 2.55 |
| No Loop Unroll | CTS - MADD | 1.46 | 28.11 | 28.08 | 2.02 | 19.42 | 19.39 | 2.49 | 12.00 | 11.98 | 2.83 | 6.79 | 6.79 | 3.03 | 3.64 | 3.64 |
| MADD | VLS | 1.28 | 48.43 | 25.28 | 2.00 | 37.80 | 19.19 | 2.79 | 26.42 | 13.44 | 3.47 | 16.45 | 8.36 | 3.95 | 9.37 | 5.11 |
| | VLS - MADD | 1.87 | 35.97 | 36.13 | 2.92 | 28.14 | 28.11 | 4.03 | 19.40 | 19.40 | 4.98 | 12.00 | 11.98 | 5.65 | 6.79 | 6.79 |
| | FTS | 1.95 | 73.91 | 38.68 | 2.77 | 52.53 | 26.74 | 3.47 | 32.90 | 16.71 | 3.95 | 18.74 | 9.51 | 4.25 | 10.07 | 5.11 |
| | FTS - MADD | 2.91 | 55.98 | 55.92 | 4.01 | 38.62 | 38.58 | 4.98 | 23.96 | 23.93 | 5.64 | 13.56 | 13.57 | 6.06 | 7.28 | 7.28 |

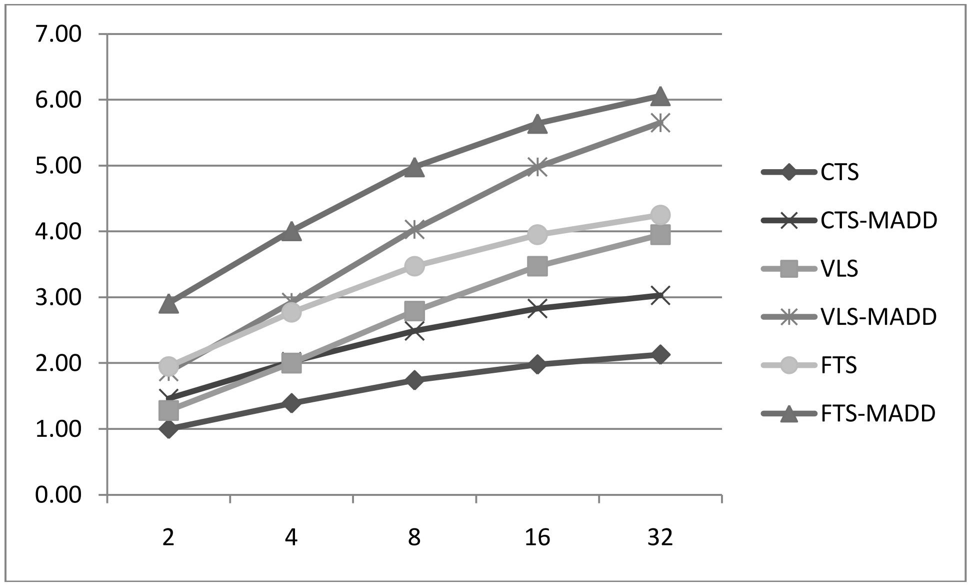

**Figure A.1** FIR 32, VL=32, no loop unroll, MADD.

**Table A.2** FIR 32, VL=32, Unroll 3 Times, MADD

| No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| CTS | 2.16 | 81.86 | 42.91 | 3.10 | 58.67 | 30.59 | 3.82 | 36.17 | 18.35 | 4.32 | 20.49 | 10.41 | 4.63 | 10.96 | 5.57 |
| CTS - MADD | 3.70 | 83.82 | 71.19 | 5.81 | 65.94 | 55.82 | 6.43 | 36.56 | 30.98 | 6.43 | 18.27 | 15.28 | 6.43 | 9.13 | 7.74 |
| VLS | 2.36 | 89.39 | 46.76 | 4.40 | 81.88 | 42.94 | 6.18 | 58.63 | 29.84 | 7.59 | 36.19 | 18.40 | 8.65 | 20.50 | 10.42 |
| VLS - MADD | 4.11 | 92.86 | 78.79 | 7.45 | 84.76 | 71.91 | 11.30 | 65.79 | 55.66 | 12.86 | 36.60 | 30.99 | 12.86 | 18.26 | 15.47 |
| FTS | 2.63 | 99.71 | 50.66 | 5.07 | 98.23 | 49.80 | 7.42 | 70.23 | 35.70 | 8.65 | 40.94 | 20.80 | 9.26 | 21.91 | 11.14 |
| FTS - MADD | 4.21 | 97.76 | 83.24 | 8.22 | 93.82 | 79.43 | 12.86 | 73.04 | 61.82 | 12.86 | 36.53 | 30.94 | 12.86 | 18.26 | 15.46 |

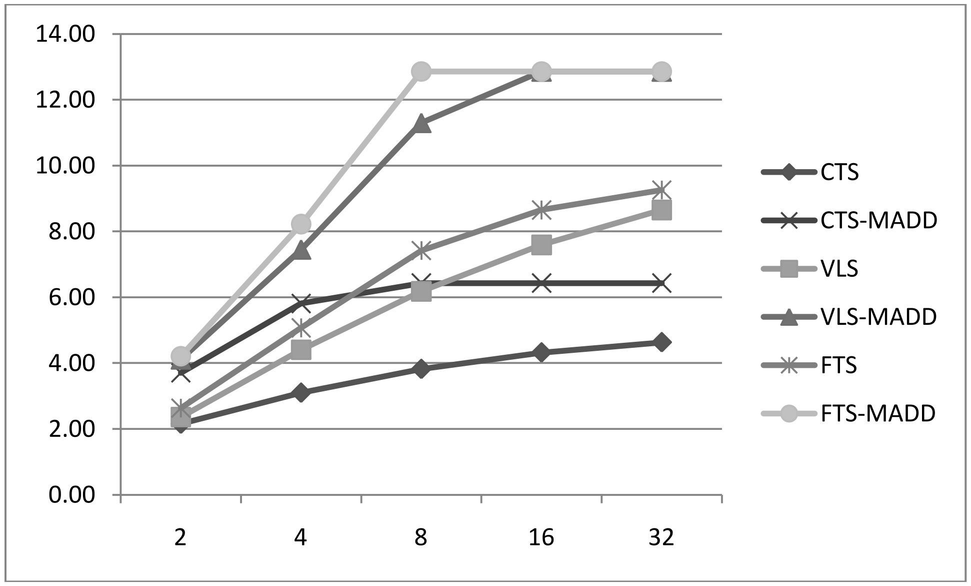

**Figure A.2** FIR 32, VL=32, unroll 3 times, MADD.

**Table A.3** FIR 32, VL=128, Unroll 3 Times, MADD

|  | No. of Lanes | 4 |  |  | 8 |  |  | 16 |  |  | 32 |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIR 32 VL=128 Unroll 3 Times MADD |  | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
|  | CTS | 4.72 | 89.57 | 45.86 | 8.64 | 81.83 | 41.66 | 12.38 | 58.63 | 29.83 | 15.28 | 36.18 | 18.41 |
|  | CTS - MADD | 8.23 | 93.52 | 79.42 | 14.94 | 84.91 | 72.10 | 23.62 | 67.11 | 57.16 | 25.72 | 36.48 | 31.08 |
|  | VLS | 4.94 | 93.74 | 48.58 | 9.44 | 89.39 | 45.35 | 17.23 | 81.62 | 41.43 | 24.52 | 58.51 | 29.74 |
|  | VLS - MADD | 8.59 | 97.42 | 82.84 | 16.08 | 91.36 | 77.38 | 27.95 | 79.98 | 67.58 | 44.65 | 64.69 | 55.21 |
|  | FTS | 5.22 | 99.85 | 51.42 | 10.58 | 99.65 | 50.86 | 20.14 | 95.30 | 48.35 | 29.41 | 70.15 | 35.65 |
|  | FTS - MADD | 8.91 | 98.85 | 83.61 | 16.82 | 97.99 | 83.24 | 32.89 | 88.23 | 74.59 | 51.44 | 60.83 | 51.53 |

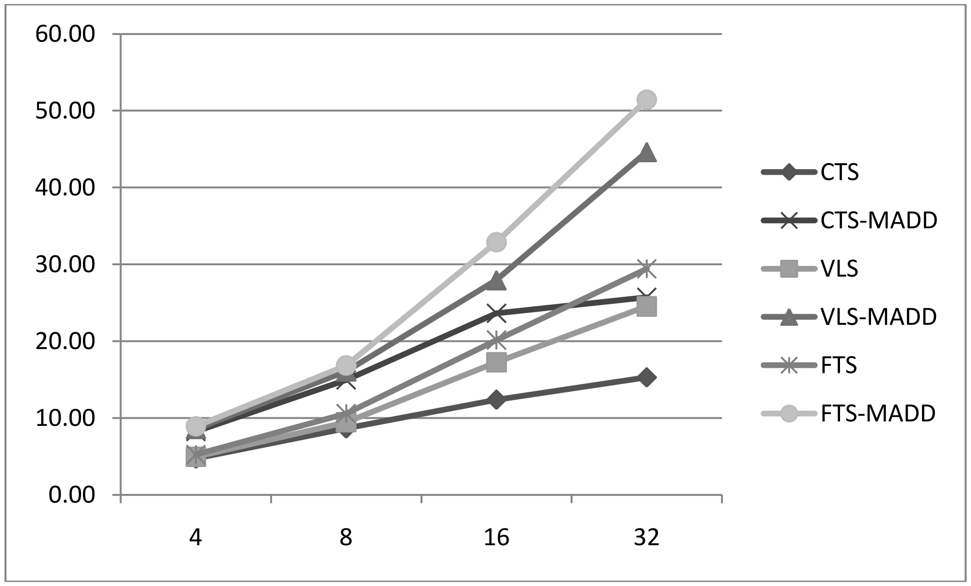

**Figure A.3** FIR 32, VL=128, unroll 3 times, MADD.

# APPENDIX B

## MM MADD BENCHMARKS

The two charts and figures for the MM with MADD RM scenarios will be found in this appendix.

**Table B.1** MM, VL=32, Unroll 1 Time, MADD

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MM | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.00 | 71.10 | 72.21 | 1.47 | 52.22 | 53.08 | 1.91 | 34.50 | 33.94 | 2.09 | 18.61 | 18.91 | 2.20 | 9.80 | 9.96 |
| Unroll 1 Time | CTS - MADD | 0.97 | 34.45 | 69.93 | 1.48 | 26.31 | 53.40 | 2.01 | 17.89 | 36.36 | 2.44 | 10.86 | 22.08 | 2.56 | 5.73 | 11.65 |
| MADD | VLS | 1.17 | 82.98 | 84.63 | 2.00 | 71.04 | 72.36 | 3.04 | 54.45 | 53.51 | 3.67 | 32.86 | 33.32 | 4.18 | 18.58 | 19.38 |
| | VLS - MADD | 1.15 | 40.79 | 82.97 | 1.94 | 34.42 | 69.88 | 2.93 | 26.03 | 52.83 | 3.99 | 17.71 | 35.98 | 4.85 | 10.79 | 21.94 |
| | FTS | 1.38 | 98.13 | 99.87 | 2.49 | 88.74 | 90.34 | 3.79 | 68.20 | 67.09 | 4.08 | 36.57 | 37.13 | 4.36 | 19.27 | 19.59 |
| | FTS - MADD | 1.38 | 49.00 | 99.85 | 2.69 | 48.14 | 97.80 | 4.01 | 35.43 | 72.14 | 4.81 | 21.68 | 44.03 | 5.05 | 11.17 | 22.70 |

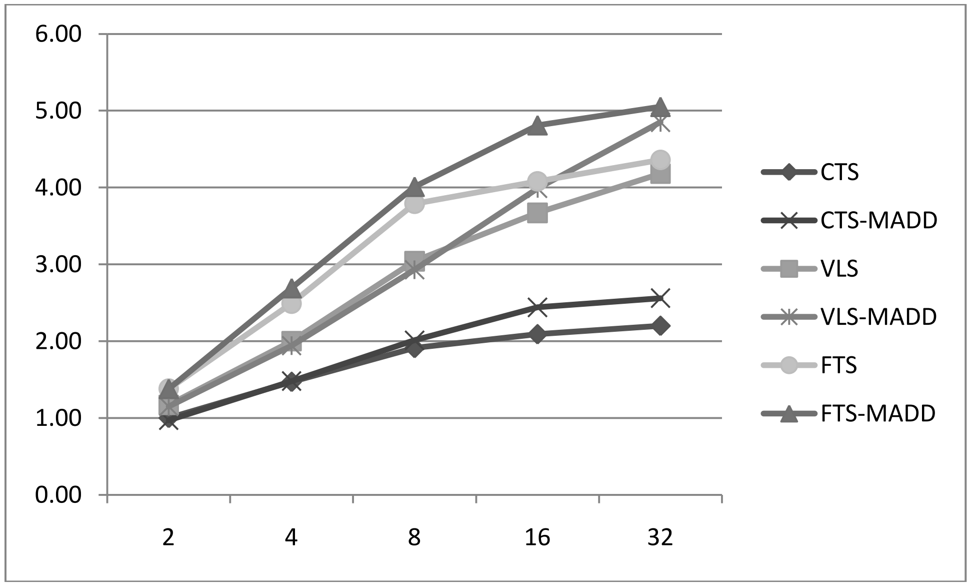

**Figure B.1** MM, VL=32, unroll 1 time, MADD.

**Table B.2** MM, VL=64, Unroll 1 Time, MADD

| | No. of Lanes | 2 | | | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MM | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=64 | CTS | 1.16 | 82.59 | 84.29 | 1.96 | 69.18 | 70.25 | 2.84 | 70.13 | 68.96 | 3.48 | 30.95 | 31.53 | 3.94 | 17.61 | 17.91 |
| Unroll 1 Time | CTS - MADD | 1.14 | 40.76 | 82.25 | 1.91 | 33.75 | 68.56 | 2.86 | 25.36 | 51.51 | 3.82 | 16.94 | 34.41 | 4.58 | 10.24 | 20.78 |
| MADD | VLS | 1.28 | 90.72 | 92.14 | 2.32 | 82.47 | 84.08 | 3.96 | 71.68 | 70.31 | 5.66 | 50.44 | 50.88 | 6.96 | 31.16 | 31.68 |
| | VLS - MADD | 1.26 | 44.71 | 91.41 | 2.29 | 40.59 | 82.56 | 3.78 | 33.88 | 68.82 | 5.65 | 24.97 | 50.77 | 7.57 | 16.76 | 34.05 |
| | FTS | 1.38 | 98.35 | 100.00 | 2.75 | 97.73 | 99.51 | 4.96 | 89.46 | 87.98 | 6.84 | 61.03 | 61.98 | 7.85 | 34.97 | 35.52 |
| | FTS - MADD | 1.39 | 49.08 | 100.00 | 2.75 | 48.80 | 99.47 | 5.22 | 46.63 | 94.42 | 7.62 | 33.65 | 68.36 | 9.12 | 19.89 | 40.50 |

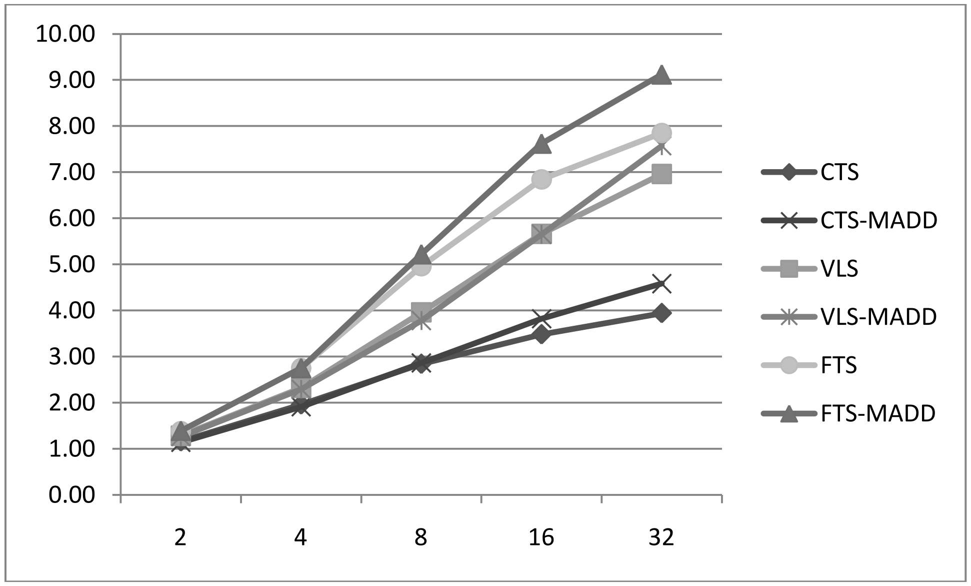

**Figure B.2** MM, VL=64, unroll 1 time, MADD.

# APPENDIX C

## LU DIV BENCHMARKS

The two charts and figures for the LU Decomposition with DIV RM scenarios will be found in this appendix.

**Table C.1** LU Decomp, VL=64, No Loop Unroll, DIV

| | No. of Lanes | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LU Decomp | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=64 | CTS | 1.83 | 40.28 | 40.87 | 1.81 | 20.11 | 20.40 | 1.64 | 10.06 | 10.20 | 1.83 | 5.03 | 5.10 |
| No Unrolls | CTS - DIV | 1.86 | 42.96 | 44.21 | 2.34 | 31.18 | 31.94 | 2.69 | 18.48 | 18.93 | 2.92 | 10.27 | 10.49 |
| DIV | FTS | 3.58 | 79.71 | 82.41 | 3.62 | 39.93 | 41.35 | 3.62 | 20.00 | 20.63 | 3.62 | 10.00 | 10.32 |
| | FTS - DIV | 3.25 | 72.68 | 75.23 | 4.61 | 59.31 | 60.79 | 5.40 | 35.50 | 36.38 | 5.72 | 19.61 | 20.09 |

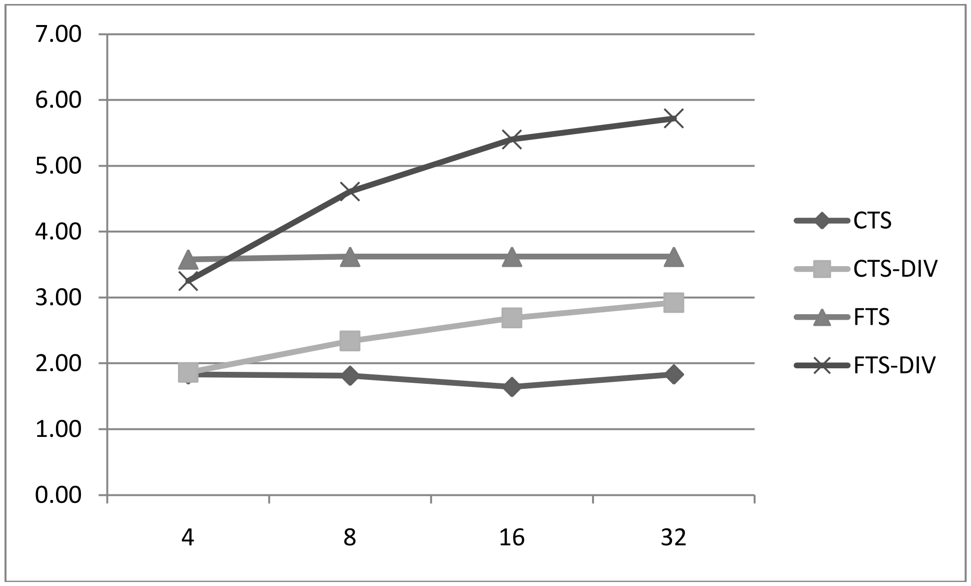

**Figure C.1** LU Decomp, VL=64, no loop unroll, DIV.

**Table C.2** LU Decomp, VL=32, No Loop Unroll, DIV

| | No. of Lanes | 4 | | | 8 | | | 16 | | | 32 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LU Decomp | | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. | Perf. | ALU Util. | LD/ST Util. |
| VL=32 | CTS | 1.83 | 19.90 | 20.61 | 1.81 | 9.58 | 10.33 | 1.72 | 4.98 | 5.15 | 1.83 | 2.49 | 2.58 |
| No Unrolls | CTS - DIV | 2.34 | 25.74 | 27.80 | 2.70 | 15.62 | 16.85 | 2.92 | 8.90 | 9.66 | 3.04 | 4.47 | 4.86 |
| DIV | FTS | 3.71 | 38.62 | 42.89 | 3.62 | 18.74 | 21.10 | 3.62 | 9.67 | 10.71 | 3.58 | 4.84 | 5.35 |
| | FTS - DIV | 4.09 | 51.28 | 54.79 | 5.40 | 30.65 | 32.46 | 5.72 | 17.87 | 18.98 | 5.96 | 9.89 | 10.61 |

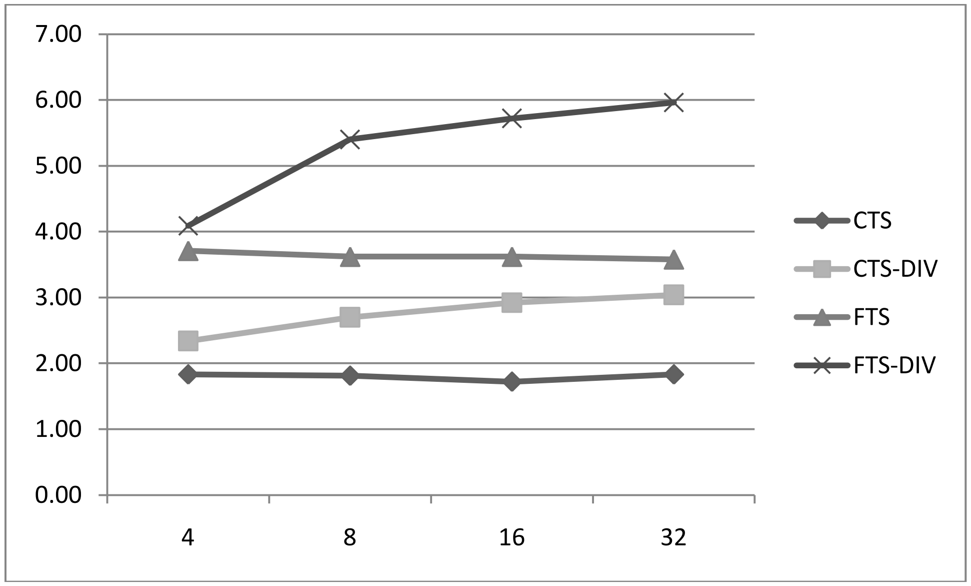

**Figure C.2** LU Decomp, VL=32, no loop unroll, DIV.

# APPENDIX D

# SCENARIOS RUN ON SYSTEM

The complete list of scenarios run on the system as part of the research in this thesis will be found in this appendix.

## D.1 Fast Fourier Transform (FFT)

### D.1.1 03_fft32_1mb_simple_v01

CTS; single pass through FFT; run on four, eight, sixteen and thirty-two lanes.

### D.1.2 04_fft32_1mb_double_v01

CTS; double pass through FFT; run on four, eight, sixteen and thirty-two lanes.

### D.1.3 05_fft32_2mb_simple_v01

FTS; single pass through FFT; vector length 32; run on four, eight, sixteen and thirty-two lanes.

### D.1.4 06_fft32_2mb_double_v01

FTS; double pass through FFT; vector length 32; run on four, eight, sixteen and thirty-two lanes.

### D.1.5 07_fft32_2mb_simple_sl_v01

VLS; single pass through FFT; vector length 32; run on four, eight, sixteen and thirty-two lanes.

### D.1.6 08_fft32_2mb_double_sl_v01

VLS; double pass through FFT; vector length 32; run on four, eight, sixteen and thirty-two lanes.

## D.2 Finite Impulse Response (FIR)

### D.2.1 12_fir32_vl32_unroll4_1mb

CTS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.2 13_fir32_vl64_unroll4_1mb

CTS; unroll loop three times; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.3 14_fir32_vl128_unroll4_1mb

CTS; unroll loop three times; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.2.4 15_fir32_vl256_unroll4_1mb

CTS; unroll loop three times; vector length 256; run on eight, sixteen and thirty-two lanes.

### D.2.5 16_fir32_vl64_unroll4_2mb

FTS; unroll loop three times; vector length 64; run on eight, sixteen and thirty-two lanes.

### D.2.6 17_fir32_vl128_unroll4_2mb

FTS; unroll loop three times; vector length 128; run on four, eight, sixteen and thirty-two lanes.

### D.2.7 18_fir32_vl256_unroll4_2mb

FTS; unroll loop three times; vector length 256; run on eight, sixteen and thirty-two lanes.

### D.2.8 31_01_fir32_vl32_nounroll_1mb

CTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.9 31_02_fir32_vl64_nounroll_1mb

CTS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.10 31_03_fir32_vl128_nounroll_1mb

CTS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.2.11 31_04_fir32_vl256_nounroll_1mb

CTS; no loop unroll; vector length 256; run on eight, sixteen and thirty-two lanes.

### D.2.12 33_01_fir32_vl32_nounroll_2mb

FTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.13 33_02_fir32_vl64_nounroll_2mb

FTS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.14 33_03_fir32_vl128_nounroll_2mb

FTS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.2.15 33_04_fir32_vl256_nounroll_2mb

FTS; no loop unroll; vector length 256; run on eight, sixteen and thirty-two lanes.

### D.2.16 34_01_fir32_vl32_unroll2_2mb

FTS; unroll loop once; vector length 32; run on two and four lanes.

### D.2.17 34_02_fir32_vl64_unroll2_2mb

FTS; unroll loop once; vector length 64; run on two and four lanes.

### D.2.18 35_01_fir32_vl32_nounroll_2mb_sl

VLS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.19 35_02_fir32_vl64_nounroll_2mb_sl

VLS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.20 35_03_fir32_vl128_nounroll_2mb_sl

VLS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.2.21 37_01_fir32_vl32_unroll4_2mb_sl

VLS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.22 37_02_fir32_vl64_unroll4_2mb_sl

VLS; unroll loop three times; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.2.23 37_03_fir32_vl128_unroll4_2mb_sl

VLS; unroll loop three times; vector length 128; run on four, eight, sixteen and thirty-two lanes.

### D.2.24 110_39_01_fir32_vl32_unroll4_2mb_no_madd

FTS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

## D.3 Finite Impulse Response (FIR) with MADD

### D.3.1 110_12_fir32_vl32_unroll4_1mb

CTS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.1.

### D.3.2 110_14_fir32_vl128_unroll4_1mb

CTS; unroll loop three times; vector length 128; run on four, eight, sixteen and thirty-two lanes; equivalent to D.2.3.

### D.3.3 110_17_fir32_vl128_unroll4_2mb

FTS; unroll loop three times; vector length 128; run on four, eight, sixteen and thirty-two lanes; equivalent to D.2.6.

### D.3.4 110_31_01_fir32_vl32_nounroll_1mb

CTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.8.

### D.3.5 110_33_01_fir32_vl32_nounroll_2mb

FTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.12.

### D.3.6 110_35_01_fir32_vl32_nounroll_2mb_sl

VLS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.18.

### D.3.7 110_37_01_fir32_vl32_unroll4_2mb_sl

VLS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.21.

### D.3.8 110_37_03_fir32_vl128_unroll4_2mb_sl

VLS; unroll loop three times; vector length 128; run on four, eight, sixteen and thirty-two lanes; equivalent to D.2.23.

### D.3.9 110_38_01_fir32_vl32_unroll4_2mb

FTS; unroll loop three times; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.2.24.

## D.4 Matrix Multiplication (MM)

### D.4.1 53_01_matmul_vl128_unroll_1mb

CTS; unroll loop once; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.4.2 53_02_matmul_vl64_unroll_1mb

CTS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.4.3 53_03_matmul_vl32_unroll_1mb

CTS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.4.4 54_01_matmul_vl128_unroll_2mb

FTS; unroll loop once; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.4.5 54_02_matmul_vl64_unroll_2mb

FTS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.4.6 54_03_matmul_vl32_unroll_2mb

FTS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.4.7 55_01_matmul_vl128_unroll_4lanes_2mb

VLS; unroll loop once; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.4.8 55_02_matmul_vl64_unroll_4lanes_2mb

VLS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.4.9 55_03_matmul_vl32_unroll_4lanes_2mb

VLS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

## D.5 Matrix Multiplication (MM) with MADD

### D.5.1 120_53_02_matmul_vl64_unroll_1mb

CTS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.2.

### D.5.2 120_53_03_matmul_vl32_unroll_1mb

CTS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.3.

### D.5.3 120_54_02_matmul_vl64_unroll_2mb

FTS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.5.

### D.5.4 120_54_03_matmul_vl32_unroll_2mb

FTS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.6.

### D.5.5 120_55_02_matmul_vl64_unroll_4lanes_2mb

VLS; unroll loop once; vector length 64; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.8.

### D.5.6 120_55_03_matmul_vl32_unroll_4lanes_2mb

VLS; unroll loop once; vector length 32; run on two, four, eight, sixteen and thirty-two lanes; equivalent to D.4.9.

## D.6 LU Decomposition (LU)

### D.6.1 60_01_LUDecomp_origMatSize128_matSize128_vl128_1mb

CTS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.6.2 60_02_LUDecomp_origMatSize128_matSize64_vl64_1mb

CTS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.6.3 60_03_LUDecomp_origMatSize128_matSize32_vl32_1mb

CTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.6.4 61_01_LUDecomp_origMatSize128_matSize128_vl128_2mb

FTS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.6.5 61_02_LUDecomp_origMatSize128_matSize64_vl64_2mb

FTS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.6.6 61_03_LUDecomp_origMatSize128_matSize32_vl32_2mb

FTS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

### D.6.7 62_01_LUDecomp_origMatSize128_matSize128_vl128_2mb_vls

VLS; no loop unroll; vector length 128; run on eight, sixteen and thirty-two lanes.

### D.6.8 62_02_LUDecomp_origMatSize128_matSize64_vl64_2mb_vls

VLS; no loop unroll; vector length 64; run on two, four, eight, sixteen and thirty-two lanes.

### D.6.9 62_03_LUDecomp_origMatSize128_matSize32_vl32_2mb_vls

VLS; no loop unroll; vector length 32; run on two, four, eight, sixteen and thirty-two lanes.

## D.7 LU Decomposition (LU) with DIV

### D.7.1 130_60_05_LUDecomp_origMatSize128_matSize128_vl64_1mb

CTS; no loop unroll; vector length 64; run on four, eight, sixteen and thirty-two lanes; equivalent to D.6.2.

### D.7.2 130_60_08_LUDecomp_origMatSize128_matSize128_vl32_1mb

CTS; no loop unroll; vector length 32; run on four, eight, sixteen and thirty-two lanes; equivalent to D.6.3.

### D.7.3 130_61_05_LUDecomp_origMatSize128_matSize128_vl64_2mb

FTS; no loop unroll; vector length 64; run on four, eight, sixteen and thirty-two lanes; equivalent to D.6.5.

### D.7.4 130_61_08_LUDecomp_origMatSize128_matSize128_vl32_2mb

FTS; no loop unroll; vector length 32; run on four, eight, sixteen and thirty-two lanes; equivalent to D.6.6.

## REFERENCES

Beldianu, S.F. and Ziavras, S.G. 2011. On-chip Vector Coprocessor Sharing for Multicores. In *Proceedings of 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*. IEEE Computer Society proceedings.

Cho, J., Chang, H., and Sung, W. 2006. An FPGA based SIMD processor with a vector memory unit. In *Proceedings of IEEE International Symposium on Circuits and Systems*. IEEE 525-528.

Golub, G. H. and Van Loan, C. F. 1996. Matrix Computations 3rd Ed. Johns Hopkins, Baltimore, USA.

Kozyrakis, C. and Patterson, D. 2003. Scalable, vector processors for embedded systems. *IEEE Micro*. 23, 6, 36-45.

Lin, Y., Lee, H., Woh, M., Harel, Y., Mahlke, S., Mudge, T., Chakrabarti, C., Flautner, K. 2006. SODA: A low-power architecture for software radio. In *Proceedings 33rd Annual International Symposium on Computer Architecture*. IEEE, Boston, MA, 89-101.

Sung, W. and Mitra, S. K. 1987. Implementation of digital filtering algorithms using pipelined vector processors. *Proceedings of the IEEE*. IEEE, 75, 9, 1293-1303.

Woh, M., Seo, S., Mahlke, S., Mudge, T., Chakrabarti, C., and Flautner, K. 2010. AnySP: Anytime Anywhere Anyway Signal Processing. *IEEE Micro*. 30, 1, 81-91.

Xilinx Inc. 2010. Partial Reconfiguration of Virtex FPGAs in ISE 12. Xilinx, http://www.xilinx.com/support/documentation/white_papers/wp374_Partial_Reconfig_Virtex_FPGAs.pdf

Yang, H. and Ziavras, S. 2005. FPGA-Based Vector Processor for Algebraic Equation Solvers. In *Proceedings of IEEE International Systems-On-Chip Conference*. IEEE, Herndon, VA, 115-116.

Yiannacouras, P., Steffan, J. G. and Rose, J. 2008. VESPA: Portable, Scalable and Flexible FPGA-Based Vector Processors. In *Proceedings of International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. ACM, Atlanta, GA.

Yu, J., Eagleston, C., Chour, C. H.-Y. Perreault, M., and Lemieux, G. 2009. Vector Processing as a Soft Processor Accelerator. *ACM Transactions on Reconfigurable Technology and Systems*. ACM, 2, 2, 1-34.