Micro Modules Design Report

Team #5

Mohamed Shawky
SEC:2, BN:16

Remonda Talaat
SEC:1, BN:20

Evram Youssef
SEC:1, BN:9

Mahmoud Adas
SEC:2, BN:21

Reham Gamal
SEC:1, BN:21

Mazen Amr
SEC:2, BN:8

Mohamed Ahmed Ibrahim
SEC:2, BN:9

Mahmoud Ahmed
SEC:2, BN:20

April 25, 2020

Contents

0.1 Introduction

This report documents some of the implementation details in the micro modules, plus it contains the synthesis statistics for those modules.

0.2 FPU (Float/Fixed Point Unit)

The fpu is a component that mux-es between an adder/subtractor, multiplier and divider, all of which operate only on floats/fixed point numbers.

The fpu itself is not instantited in the project, but rather its subunits.

All fpu subunits follow the fpu specs (including all ports except 'operation' and all logic and state) in '0.14.4' in the main document.

FPU is implemented in 'src/fpu.vhdl' , while its subunits are implemented in 'src/fpu_subunits' .

Each fpu subunit has multiple architectures. They are:

- 'with_operators' : copied from phase1, used temporarily in macro modules while other architectures are developed.
- 'first_algo' .
- 'sec_algo' : divider does have only first_algo and with_operators.

0.2.1 FPU Adder/Subtractor

Synthesis Statistics

Architecture	Area	Power	Time
first algo	cell=255 Area per cell =322	1013.76	1201
sec algo	cell=321 Area per cell =376	830.51	1005

0.2.2 FPU Multiplier

Synthesis Statistics

Architecture	Area	Power	Time
sec algo	1745	3186.622559	59739.5

0.2.3 FPU Divider

Synthesis Statistics

Architecture	Area	Power	Time
first algo	3445	3557.846191	57189.6

0.3 RAM

RAM is generic in:

- number of words
- width of address bus
- width of data bus

RAM has seperate bus for read and write, and performas both on falling edge of the clock.

RAM also has asyncrounous reset signal that performs parallel load into all internal lateches.

0.3.1 Synthesis Statistics

Area	Power	Time
cells=8910 Area per cell=26279	4437.828	122.4

0.4 CPU

Composed of 3 parts:

1. ‘scripts/preprocessor.py‘ : reads input json and outputs ‘.in‘ file that contains ASCII 0s and 1s in lines, each line is 32 byte in width.
2. ‘test/cpu_tb.vhdl‘ : testbench that reads ‘.in‘ files in ‘input‘ directory and performs the main simulation by feeding the compressed data and switching state input and receiving output data and writing them back to ‘out/*.out‘ files.

   Some lines in ‘cpu_tb.vhdl‘ are commented for this phase, because they are integration points for next phases.

3. ‘scripts/output_fromatter.py‘ : convert ASCII output (0s and 1s) from ‘cpu_tb.vhdl‘ into human readable json.