Macro 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 Mohamed
SEC:2, BN:22

April 25, 2020

Contents

0.1 Introduction

All macro modules are fully documented in the main document, see sections ‘0.14‘ , ‘0.5‘ .

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

0.2 IO

Implemented in ‘src/io.vhdl‘ , follows the subsection ‘0.14.1‘ in main refernce.

0.2.1 Subunits

IO is composed of the subunits in ‘src/io_subunits‘ :

- ‘src/io_subunits/decompressor.vhdl‘ :
  - at each positive edge:
    - takes the compressed input and decompress it into internal 128 bit buffer, and andvances fill_i (initialzed with 0) (see section ‘0.16 Decompression‘ in main document).
    - checks the current buffer state, if fill_i - flush_i ¿ 32, that means that buffer has more thant 32 bit of uncompressed data, it flushes them.
  - uses couple of integer adders, and range_extractor to figure out indices of buffer to fill.
- ‘src/io_subunits/range_extractor.vhdl‘ : given a compressed 4bit packet and the previous range_extractor output, outputs the indices in the buffer to iterate over so the decompressor knows where to fill the data for this packet.
- ‘src/io_subunits/next_adr.vhdl‘ : given output from decmpressor (which can stay without change for couple of cycles because the buffer is filled), the next_adr unit (nau) updates the address of the data, and figures out whether it has reached the final address or not. NAU advances the address by 1 in case float64 mode, otherwise it advances the address by 2 because of how data is padded in solver and interpolator. When

NAU detects that address of some variable has finished (based on values extracted from the header (the first decompressed word)), NAU advances the address to the beginning of the next variable (see section '0.13 Input Format' in main document).

- IO subunits also include integer operator units (adders, multipliers, incrementors ... .) as following:
  - 'src/io_subunits/decrementor.vhdl'
  - 'src/io_subunits/full_adder.vhdl'
  - 'src/io_subunits/half_adder.vhdl'
  - 'src/io_subunits/incrementor.vhdl'
  - 'src/io_subunits/int_adder.vhdl'
  - 'src/io_subunits/int_multiplier.vhdl'

  They are used in multiple units accross the project.

0.2.2 Synthesis Statistics

Area	Power	Time
cell=271 Area per cell =12357	12356.2	242

0.3 Solver

Implemented in 'src/solver.vhdl' and has some of its procedures defined in 'src/solver_pkg.vhdl' .

- 'src/solver.vhdl' : holds solver main process, which includes fixed and variable step orchestration FSMs, main data loading and reset/error handling. Also, the file contains some matrix manipulation procedures that aid the solver functionalities.
- 'src/solver_pkg.vhdl' : holds packages containing sub-procedures of solver. Procedures are divided into: Memory IO procedures, Matrix Manipulation procedures and Utils for sending data on output bus and address calculation.

- The directory ‘solver_subtest‘ holds files for individual components test and integration.

All design details of solver are in subsection ‘0.14.2‘ in main document.

0.3.1 Synthesis Statistics

Area	Power	Time
TODO	TODO	TODO

0.4 Interpolator

Implemented in ‘src/interp.vhdl‘ .

‘src/interp.vhdl‘ : contains the whole implementation of the interpolator. The implementation is basically a main process that handles data loading, reset, errors and main orchestration FSM. The sub-functionalities are done through a set of procedures for: Memory IO, data outage on bus and range finding of t_s.

All design details of solver are in subsection ‘0.14.2‘ in main document.

0.4.1 Synthesis Statistics

Area	Power	Time
TODO	TODO	TODO