ASSIGNMENT 2 STAGE 1 REPORT

Sreemanti Dey

January 2022

1 Objective

The modules that have been designed include ALU, Register File, Program Memory and Data Memory.

2 Assumptions

VHDL
edaplayground
Aldec Riviera Pro 2020.04 used for simulation
Mentor Precision 2021.1 used for synthesis

3 Implementation details

I have made 4 VHDL files for implementing ALU, Register, data memory and program memory and I have written testbenches for all these vhdl files individually.
I have used the run.do file that was available on edaplayground for synthesis.
Note: If the test benches are to be run on edaplayground, the name of the testbench has to be specified in top entity.

3.1 ALU

3.1.1 Details

I have written case statements inside a process that takes in a,b,cin,opcode in its sensitivity list. I have declared a variable temp that contains the result of the operation performed on zero-extended a and zero-extended b, from where we give MSB to carry out and the rest to the result. In case of operations that don't change carry, carry is don't care value, so I have just set value of carry out to carry in.

3.1.2 ALU_tb

This is the testbench I have made for testing the various opcodes in ALU. It tests all the opcodes with different values of a, b and cin and throws an assertion error if the result does not match with the intended result.

3.1.3 Simulation result via EPWave

Here is a picture of the simulation results I have achieved by EPWave

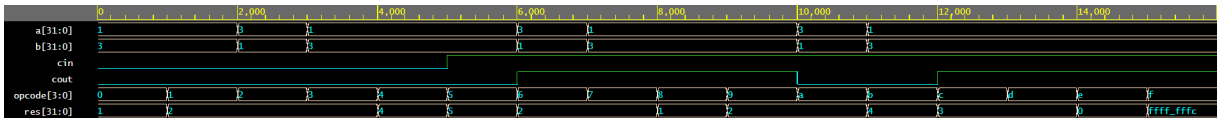

Figure 1: ALU

3.1.4 Synthesis result

Here is a picture of the synthesis results - that includes number of LUTs used, number of IOs, buffers and so on.

```
# Info: Device Utilization for 7A100TCSG324
# Info: ***********************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: -----------------------------------------------------------------------
# Info: IOs                               102     210      48.57%
# Info: Global Buffers                    0       32        0.00%
# Info: LUTs                              167     63400     0.26%
# Info: CLB Slices                        41      15850     0.26%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: DSP48E1s                          0       240       0.00%
# Info: -----------------------------------------------------------------------
# Info: *****************************************************
# Info: Library: work    Cell: ALU    View: beh_ALU
# Info: *****************************************************
# Info:  Number of ports :                          102
# Info:  Number of nets :                           405
# Info:  Number of instances :                      304
# Info:  Number of references to this view :          0
# Info: Total accumulated area :
# Info:  Number of LUTs :                           167
# Info:  Number of Primitive LUTs :                 168
# Info:  Number of LUTs with LUTNM/HLUTNM :           2
# Info:  Number of MUX CARRYs :                      64
# Info:  Number of accumulated instances :          401
```

3.2 Register

3.2.1 Details

Register File contains an array of 16 std logic vectors of 32-bits each. Since there are 16 vectors, I have made address of 4 bits. Its inputs include two read addresses, one write address, one data input, one write enable and a clock. There are two data outputs on which contents of the array elements selected by read addresses are continuously available. If write enable is active, at clock edge the input data gets written in the array element selected by write address.

3.2.2 Register_tb

This is the testbench I have made for testing the read and write operations of the register file. I have written some data into the register first at different addresses and after that I have checked using read operation whether the data that is present is the correct one or not. It throws an assertion error if the result does not match with the correct data.

3.2.3 Simulation result via EPWave

Here is a picture of the simulation results I have achieved by EPWave

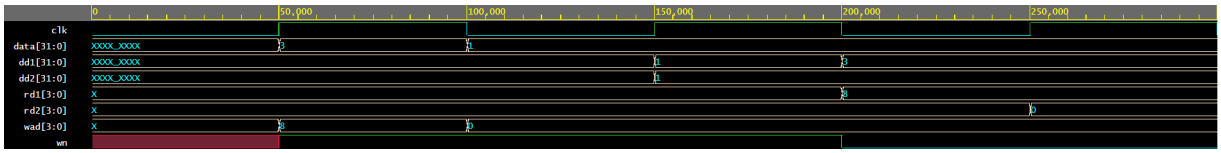

Figure 2: Register

3.2.4 Synthesis result

Here are my synthesis results - that includes number of LUTs used, number of IOs, buffers and so on.

```
# Info: Device Utilization for 7A100TCSG324
# Info: ***************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ---------------------------------------------------------------
# Info: IOs                               110     210      52.38%
# Info: Global Buffers                    1       32        3.12%
# Info: LUTs                              48      63400     0.08%
# Info: CLB Slices                        12      15850     0.08%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: Distributed RAMs
# Info:    RAM32M                         10
```

```
# Info:    RAM64M                          2
# Info: DSP48E1s                           0       240       0.00%
# Info: ------------------------------------------------------------------
# Info: ******************************************************************
# Info: Library: work    Cell: regtr    View: register_arch
# Info: ******************************************************************
# Info:  Number of ports :                              110
# Info:  Number of nets :                               220
# Info:  Number of instances :                          111
# Info:  Number of references to this view :              0
# Info: Total accumulated area :
# Info:  Number of LUTs :                                48
# Info:  Number of Primitive LUTs :                      48
# Info:    Number of LUTs as Distributed RAM :           48
# Info:  Number of accumulated instances :              123
```

3.3 Program-Memory

3.3.1 Details

Program memory contains an array of 64 std logic vectors of 32-bits. Since there are 64 std logic vectors hence I have provided a 6-bit input for my address. Program Memory has one read port. Read operations are modeled such that read is unclocked (like a combinational circuit).

3.3.2 Program-Memory_tb

This is the testbench I have made for testing the read operation of the program memory. The initialised values in my program memory have been tested here. It throws an assertion error if the result does not match with the correct data.

3.3.3 Simulation result via EPWave

Here is a picture of the simulation results I have achieved by EPWave

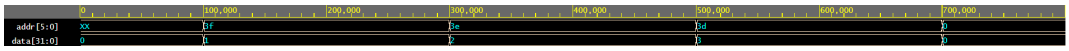

Figure 3: Program Memory

3.3.4 Synthesis result

Here is a picture of the synthesis results - that includes number of LUTs used, number of IOs, buffers and so on.

```
# Info: Device Utilization for 7A100TCSG324
# Info: ******************************************************************
```

```
# Info: Resource                          Used    Avail   Utilization
# Info: ---------------------------------------------------------------
# Info: IOs                               38      210      18.10%
# Info: Global Buffers                    0       32        0.00%
# Info: LUTs                              0       63400     0.00%
# Info: CLB Slices                        0       15850     0.00%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: DSP48E1s                          0       240       0.00%
# Info: ---------------------------------------------------------------
# Info: ************************************************
# Info: Library: work    Cell: pm    View: pm_arch
# Info: ************************************************
# Info:  Number of ports :                          38
# Info:  Number of nets :                           33
# Info:  Number of instances :                      33
# Info:  Number of references to this view :         0
# Info: Total accumulated area : unknown
```

3.4 Data-Memory

3.4.1 Details

Data memory contains an array of 64 std logic vectors of 32-bits. Data Memory has one read port and one write port. Since there are 64 std logic vectors hence I have provided a 6-bit input for my address. Read/write operations are modeled such that write is clocked whereas read is unclocked (like a combinational circuit). I have provided 4 write enable signals to provide byte level write operation, i.e. write enable is a std logic vector of 4 bits where a set bit implies that part of the input data will be written.

3.4.2 Data-Memory_tb

This is the testbench I have made for testing the read and write operations of the data memory. I have written some data into the memory first at different addresses and after that I have checked using read operation whether the data that is present is the correct one or not. It throws an assertion error if the result does not match with the intended result.

3.4.3 Simulation result via EPWave

Here is a picture of the simulation results I have achieved by EPWave

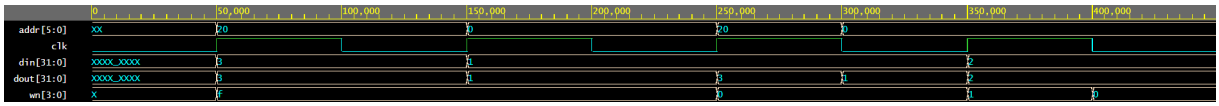

Figure 4: Data Memory

3.4.4 Synthesis result

Here is my synthesis results - that includes number of LUTs used, number of IOs, buffers and so on.

```
# Info: ************************************************************
# Info: Device Utilization for 7A100TCSG324
# Info: ************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ------------------------------------------------------------
# Info: IOs                               75      210      35.71%
# Info: Global Buffers                    1       32        3.12%
# Info: LUTs                              32      63400     0.05%
# Info: CLB Slices                        8       15850     0.05%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: Distributed RAMs
# Info:    RAM64X1S                       32
# Info: DSP48E1s                          0       240       0.00%
# Info: ------------------------------------------------------------
# Info: ****************************************************
# Info: Library: work    Cell: dm    View: dm_arch
# Info: ****************************************************
# Info:  Number of ports :                          75
# Info:  Number of nets :                           150
# Info:  Number of instances :                      76
# Info:  Number of references to this view :        0
# Info: Total accumulated area :
# Info:  Number of LUTs :                           32
# Info:  Number of Primitive LUTs :                 32
# Info:    Number of LUTs as Distributed RAM :      32
# Info:  Number of accumulated instances :          107
```