ASSIGNMENT 2 STAGE 2 REPORT

Sreemanti Dey

January 2022

1 Objective

The modules that have been designed include ALU, Register File, Program Memory, Data Memory, Decoder, Condition-Checker, Flag-Update, Program-Counter and finally the processor.

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 a testbench for the final processor file that instantiates all the components that have been made in the 2 stages and the EPWave shows all the signals present.

I have run my code on a few assembly programs and the EPWave of all of them have been attached here.

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
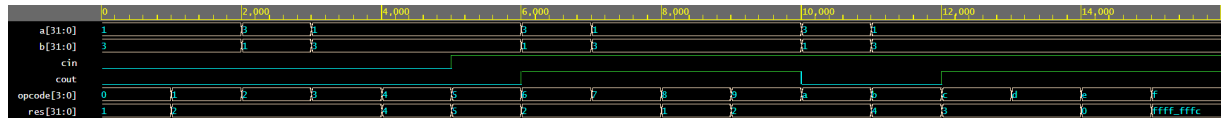

Figure 1: ALU

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
```

2

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
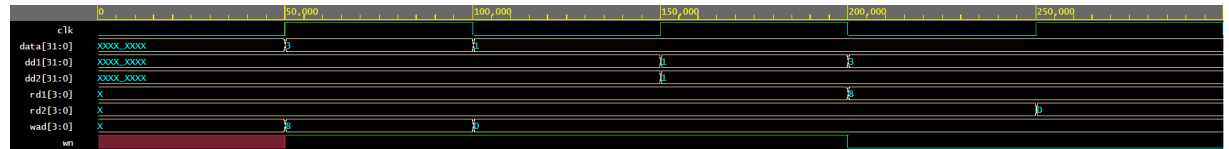

Figure 2: Register

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: ************************************************************
```

4

```
# 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

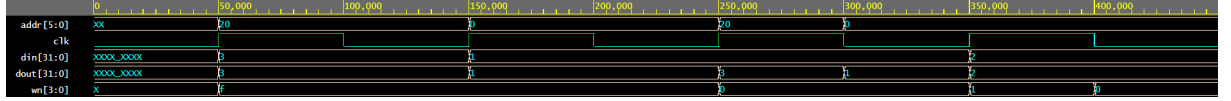Here is a picture of the simulation results I have achieved by EPWave

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
```

3.5   Condition-checker

3.5.1   Details

Condition-checking is done by a file called cond.vhd that checks various conditions and evaluates the value of predicate whether it is true or false.

6

3.5.2 Simulation result via EPWave

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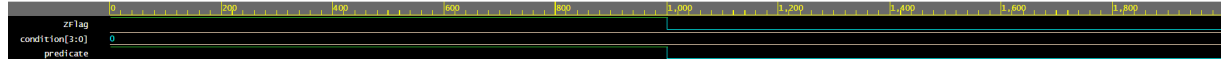

Figure 5: Condition Checker

3.5.3 Synthesis result

Here is 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                             9        210       4.29%
# Info: Global Buffers                  0        32        0.00%
# Info: LUTs                            4        63400     0.01%
# Info: CLB Slices                      1        15850     0.01%
# 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: cond    View: cond_arch
# Info: *********************************************************
# Info:  Number of ports :                   9
# Info:  Number of nets :                    21
# Info:  Number of instances :               13
# Info:  Number of references to this view :  0
# Info: Total accumulated area :
# Info:  Number of LUTs :                    4
# Info:  Number of Primitive LUTs :          4
# Info:  Number of accumulated instances :   13
```

3.6 PC

3.6.1 Details

Here I have 2 signals, named pcin and pcout, where pcout gets updated to the new value depending on if its a branch instruction or a normal instruction. The handling of the clock is kept outside this vhd file so that pcin gets the value of pcout only at rising edge of clock in the processor.

3.6.2   Simulation result via EPWave

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

Figure 6: Program Counter

3.6.3   Synthesis result

Here is 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                                97       210       46.19%
# Info: Global Buffers                     0        32        0.00%
# Info: LUTs                               63       63400     0.10%
# Info: CLB Slices                         16       15850     0.10%
# 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: pc    View: pc_arch
# Info: ****************************************************
# Info:  Number of ports :                    97
# Info:  Number of nets :                     336
# Info:  Number of instances :                275
# Info:  Number of references to this view :   0
# Info: Total accumulated area :
# Info:  Number of LUTs :                     63
# Info:  Number of Primitive LUTs :           63
# Info:  Number of MUX CARRYs :               58
# Info:  Number of accumulated instances :    275
```

3.7   Flag-Updater

3.7.1   Details

This file updates all flags based on the input instruction type, mainly this is based on the pdf file uploaded by sir that explains the circuit for maintaining various flags.

3.7.2 Flag-updater_tb

This tests the signals.

3.7.3 Simulation result via EPWave

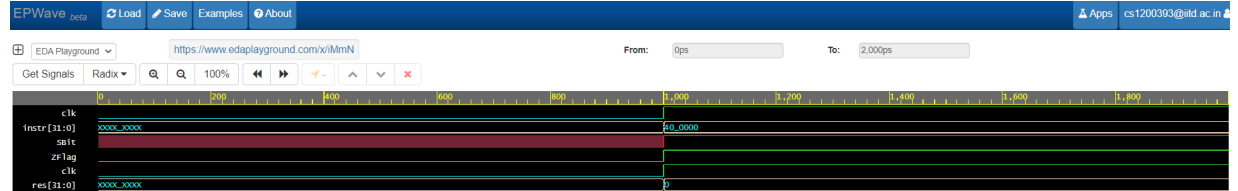Here is a picture of the simulation results I have achieved by EPWave

Figure 7: Flag Updater

3.7.4 Synthesis result

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

```
# Info: Resource                               Used    Avail   Utilization
# Info: ----------------------------------------------------------------
# Info: IOs                                     74      210     35.24%
# Info: Global Buffers                          1       32       3.12%
# Info: LUTs                                    13      63400    0.02%
# Info: CLB Slices                              3       15850    0.02%
# Info: Dffs or Latches                         4       126800   0.00%
# Info: Block RAMs                              0       135      0.00%
# Info: DSP48E1s                                0       240      0.00%
# Info: ----------------------------------------------------------------
# Info: ********************************************************
# Info: Library: work    Cell: flagupd    View: flag_arch
# Info: ********************************************************
# Info:  Number of ports :                         74
# Info:  Number of nets :                          127
# Info:  Number of instances :                      75
# Info:  Number of references to this view :         0
# Info: Total accumulated area :
# Info:  Number of Dffs or Latches :                4
# Info:  Number of LUTs :                          13
# Info:  Number of Primitive LUTs :                14
# Info:  Number of LUTs with LUTNM/HLUTNM :         2
# Info:  Number of accumulated instances :         75
# Info: ***************************
```

9

3.8 My Types

This is the file uploaded on moodle and I have used it to enumerate the DP opcodes and the types of instructions.

3.9 Decoder

This is the file uploaded on moodle and I have used it for my instruction decoder.

4 Processor

This is the main file, that integrates all the entities that have been made in this stage. I have instantiated all the components in the file and then I have defined signals that will take care of mapping of the components. The signals include the various flags, the predicate, the input and output for data memory, program memory and register file and also for the ALU operands. Overall, I have made a process that has clock in its sensitivity list and it conditionally makes write enable as on or off for various instructions to function properly. SBit is always 0 for mov, add, sub while it is always 1 for cmp. Also we conditionally give either instruction(15 to 12) to register in case of a DT instruction else we give instruction(3 to 0) to register. Also, we give similar control signals for write back in register file as well as for input to data memory. Finally I update PC value at the end.

4.1 Testbench

I have hardcoded some value in the loop that updates the clock every cycle, based on the number of instructions in the instruction memory, also adding some extra in case there are loops.

4.2 EPWave for prog1

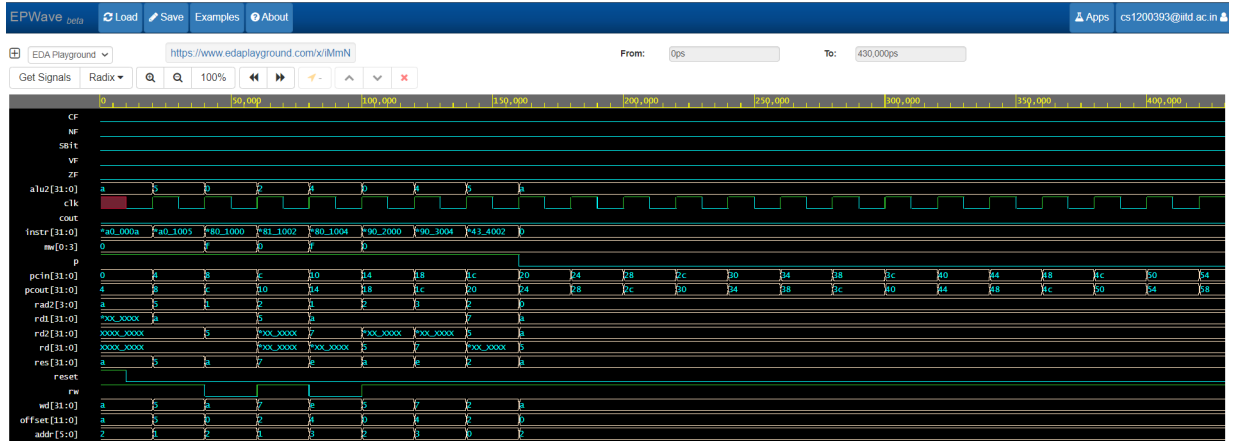Here is a picture of the simulation results I have achieved by EPWave for prog1.s

Figure 8: Program 1

4.3 EPWave for prog2

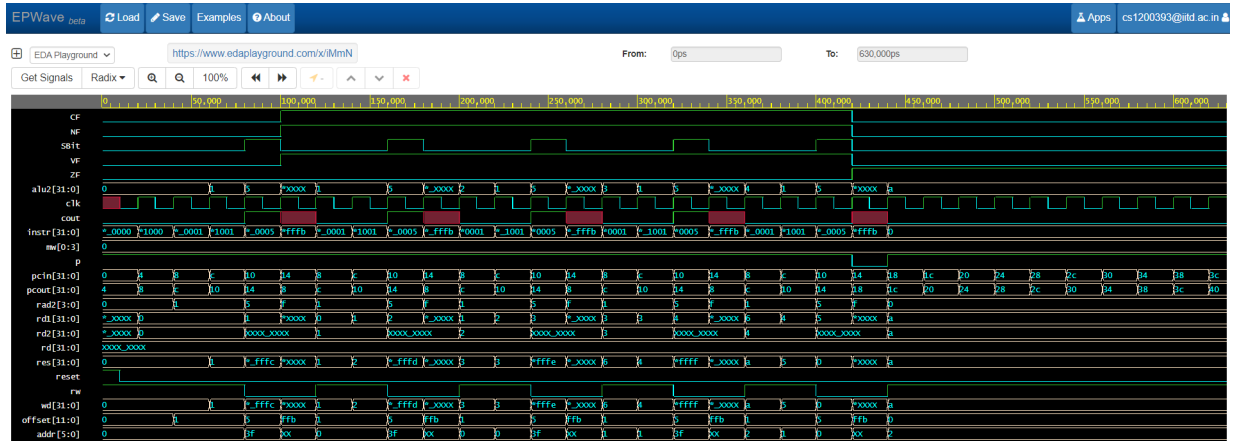Here is a picture of the simulation results I have achieved by EPWave for prog2.s

Figure 9: Program 2

4.4 EPWave for prog3

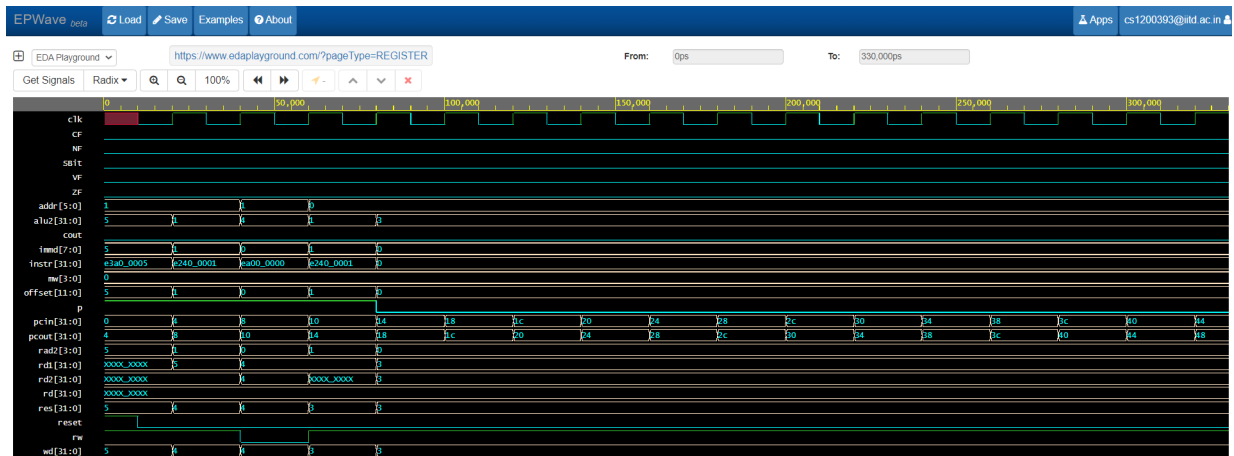Here is a picture of the simulation results I have achieved by EPWave

Figure 10: Program 1

4.5 Synthesis results

```
# Info: Device Utilization for 7A100TCSG324
# Info: *****************************************************************
# Info: Resource                            Used    Avail   Utilization
# Info: -----------------------------------------------------------------
# Info: IOs                                  2       210      0.95%
# 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: Processor    View: beh_Processor
# Info: *****************************************************************
# Info:  Number of ports :                          2
# Info:  Number of nets :                           0
# Info:  Number of instances :                      0
# Info:  Number of references to this view :        0
# Info: Total accumulated area :
# Info:  Number of gates :                          0
# Info:  Number of accumulated instances :          0
# Info: ****************************
# Info:  IO Register Mapping Report
# Info: ****************************
# Info: Design: work.Processor.beh_Processor
```

```
# Info: +----------+-----------+----------+----------+----------+
# Info: | Port     | Direction |   INFF   |  OUTFF   |  TRIFF   |
# Info: +----------+-----------+----------+----------+----------+
# Info: | clk      | Input     |          |          |          |
# Info: +----------+-----------+----------+----------+----------+
# Info: | reset    | Input     |          |          |          |
# Info: +----------+-----------+----------+----------+----------+
# Info: Total registers mapped: 0
```