EENG 5560 FINAL PROJECT Report Template

Name: Tejaswini Pulishetti & Vinay Reddy Padigela

Due: 02/04/2024

Table of Contents

[Design 3](#_Toc165573717)

[Block diagrams 3](#_Toc165573718)

[Overall design 3](#_Toc165573719)

[Subcomponents 6](#_Toc165573720)

[Design explanation 11](#_Toc165573721)

[Functionality 11](#_Toc165573722)

[Results 16](#_Toc165573723)

[Generated Schematics 16](#_Toc165573724)

[Waveforms 30](#_Toc165573725)

[31](#_Toc165573726)

[Table/Calculations 33](#_Toc165573727)

[Overall Design 33](#_Toc165573728)

[Subcomponent test cases 34](#_Toc165573729)

[MUX 4X1: 35](#_Toc165573730)

[MUX 2X1: 35](#_Toc165573731)

# Design

## Block diagrams

### Overall design

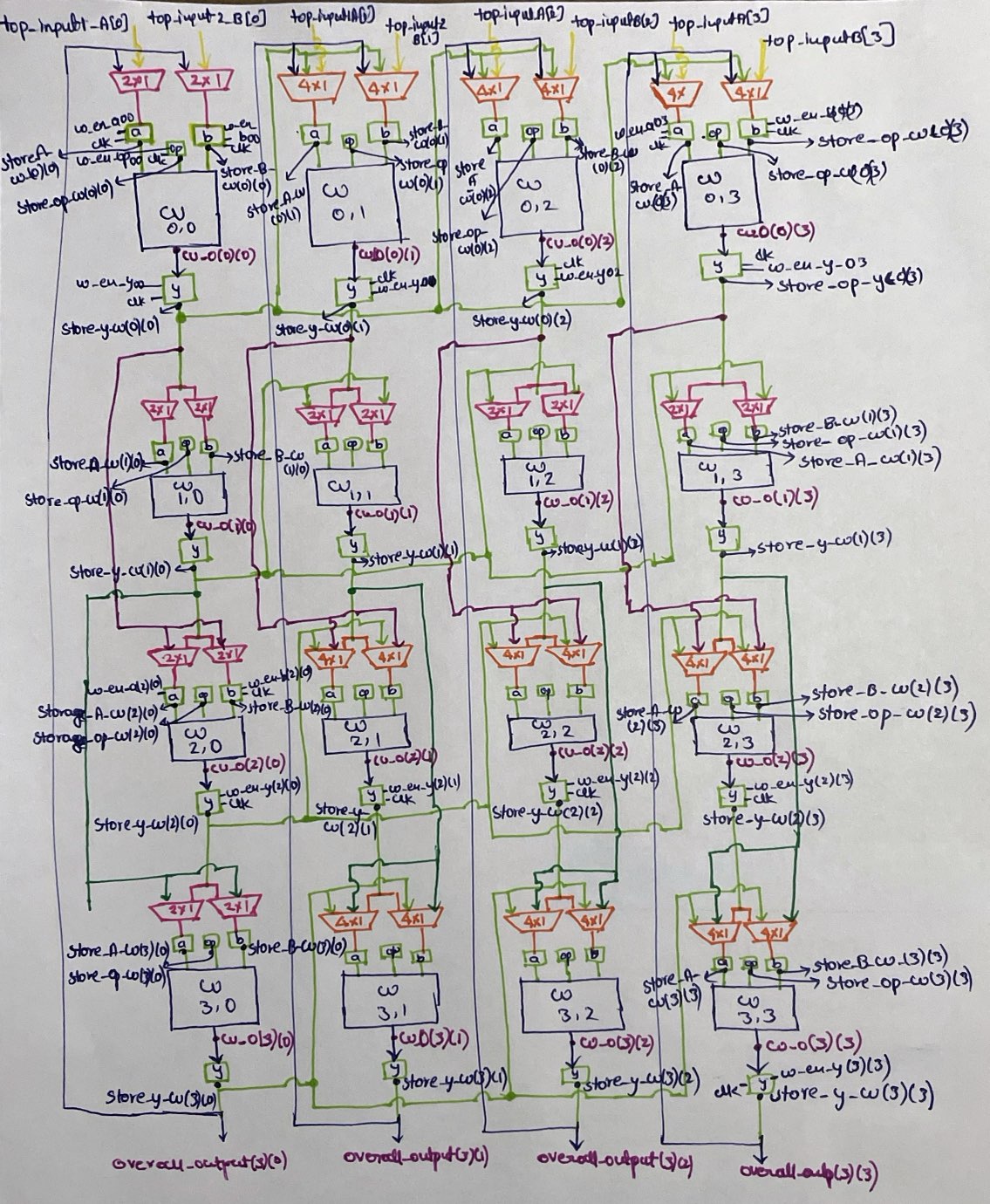


Figure 1-block diagram for top module

In this top module, it describes the overall design, where the pink color represents the 2x1 muxes, orange represents the 4x1 muxes, yellow represents the external inputs, green represents the storage units and the storage unit y output, being fed to the beside muxes for the CU operations as the inputs, dark green represents the feed back path from the storage y of the row 1 to the muxes input in the row 3, purple represent the feedback path from the row 0 to the muxes in the row 2, blue shows the feedback path from row 3 to the row 0 muxes. All the external input are named as top\_input1\_A, top\_input1\_B and represented in array. Overall\_output this gives the overall output, w\_en\_a, w\_en\_b, w\_en\_op, w\_en\_y all these represent the enables for the storage units of A,B,OP,Y . clk used for all the storage units to get triggered, where clock is common to whole top module . Output of the storage unit A is stored in store\_A\_CU, B in store\_B\_CU , OP in store\_op\_CU, Y in store\_y\_CU , CU\_O is the cu output, all the cu outputs and storage units outputs, mux outputs are declared as the intermediated signals.

**Overall component:** top\_module

**Parameters:** d\_w: integer := 4, rows: natural := 4, cols: natural := 4, declared and used

**Input ports:**

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| *top\_input1\_A,* *top\_input2\_B* | 4 | These are the external inputs for the whole design. |
| *Operations\_CU* | 4 | This provides the overall operations, which need to be done at the individual CU. |
| *mux\_two\_sel\_A,* *mux\_two\_sel\_B* | 4 | These are the select lines for the 2x1 mux operations |
| *mux\_four\_sel\_A,* *mux\_four\_sel\_B* | 4 | Select lines for the 4x1 mux operations. |
| *clk* |  | Over all design clock |
| *w\_en\_a* |  | Acts as the enable for a\_storage unit |
| *w\_en\_b* |  | Enable to b storage unit |
| *w\_en\_op* |  | Op storage unit enable |
| *w\_en\_y* |  | Storage unit enable for y |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| Overall\_output | 4 | These are the outputs, from the third stripe CU30,CU31,CU32,CU33. Output will have 4 bits each. |

Necessary intermediate signals:

We have used the intermediate signals for the mux 2x1 and 4x1 outputs individually for the two mux operations for every individual CU. In addition, we have even used the CU output signals, which can be given to the storage y , to store the output. we have created the storage output signals individually for the A,B,Op storage units.

Intermediate signals of top module for the CU at each row and column:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| CU\_O | 4 | This have the values of the CU outputs as each row and column. |

Intermediate signals of top module for 4x1 mux and 2x1 mux:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| mux\_four\_out\_A, mux\_four\_out\_B | 4 | These are used as intermediate signals for the 4x1 outputs |
| mux\_two\_out\_A, mux\_two\_out\_B | 4 | Output signals for the mux 2x1 |

Intermediate signals of top module for storage units:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| store\_A\_CU | 4 | Stores the output of the a storage unit |
| store\_B\_CU | 4 | Output signal for B storage unit |
| store\_y\_CU | 4 | Y storage unit output signal |
| store\_op\_CU | 4 | This acts as the output for the OP |

### Subcomponents

4\*1 mux:

we have used 4\*1 mux as one of the subcomponent in the top module. Where the mux is used for the selecting the inputs to the next stripe for both the input pins of the CU. So 4x1 is used for CU01,CU02,CU03,CU21,CU22,CU23,CU31,CU32,CU33 individually.

we have created a custom package by defining the inputs of the mux as **I0o,I1o,I2o,I3o** and the data type as **muxfour.**

we have instantiated it in the top module and used for individual CU input selection.

**Truth table for the mux4to1**

If I0o is selected then the output will be Input0, if I1o output is Input1,if I2o output is Input2, if I3o output is Input3. Here select input acts as the select line as it is defined directly in the custom package.

|  |  |  |
| --- | --- | --- |
| Select line | input | Resulting output |
| I0o | Input0 | Input0 |
| I1o | Input1 | Input1 |
| I2o | Input2 | Input2 |
| I3o | Input3 | Input3 |

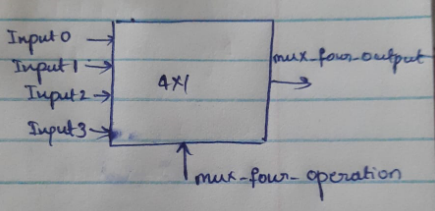


Figure 2 – 4\*1 mux block diagram

Subcomponent: 4\*1 mux :

Parameter : d\_w\_mux\_four: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| Input0,Input1,Input2,Input3 | 4 | mux inputs |
| mux\_four\_operation | 4 | Muxfour , declared in the custom package, used as Select line data type |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| mux\_four\_output | d\_w\_mux\_four - 1 downto 0 | mux output |

Subcomponent: Computational unit

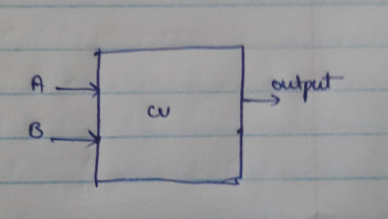


Figure 3- CU block diagram

For the CU operation, we have created the custom package, where all the operations are declared. But for the multiplication, we have used a variable and mapped it to the output , where the multiplication gives only the last 4 bits as the output. we have declared inputs as the unsigned bits.

Parameters: d\_w = 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| A,B | d\_w – 1 downto 0 | Data inputs |
| all\_operation | 4 | Accessing directly from the custom pack, which was created as all\_oper |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| output | d\_w - 1 downto 0 | Data output |

## 

Subcomponent: Mux 2x1:

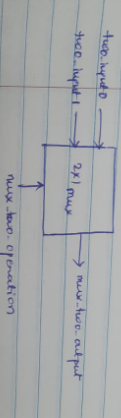


Figure 4- 2x1mux block diagram

we have used 2\*1 mux as subcomponent in the top module. Where the mux is used for the selecting the inputs to the next stripe for both the input pins of the CU. So 2x1 is used for CU00,CU10,CU11,CU12,CU13,CU20,CU30 individually.

we have created a custom package by defining the inputs of the mux as **T0o,T1o**and the data type as **muxtwo.**

we have instantiated it in the top module and used for individual CU input selection.

If T0o is selected then the output will be Two\_Input0, if T1o output is Two\_Input1. Here select input acts as the select line as it is defined directly in the custom package.

**Truth table for the mux2to1**

|  |  |  |
| --- | --- | --- |
| Select line | input | Resulting output |
| T0o | Two\_input0 | Two\_input0 |
| T1o | Two\_input1 | Two\_input1 |

Parameter : d\_w\_mux\_two: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| two\_Input0, two\_Input1 | 4 | mux inputs |
| mux\_two\_operation | 4 | muxtwo, declared in the custom package, used as Select line data type |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| mux\_two\_output | 4 | mux output |

Subcomponent: a\_storageunit

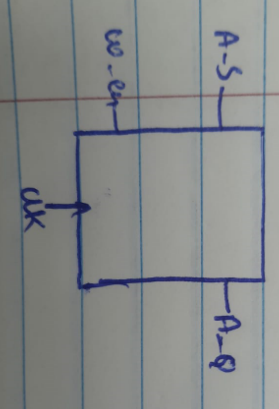


Figure 5- A storage unit block diagram

For the a storage, we have used the D flipflop concept, where the output will be available , only when the enable and CLK is available at 0 for enable and the clk for 1. And the same process of operation for the B storage, Y storage also.

Parameter : a\_d\_w: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| A\_S | 4 | A storage unit input |
| w\_en, clk |  | Enable and the clock for the storage to get triggered |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| A\_Q | 4 | Storage unit output |

Subcomponent: b\_storageunit

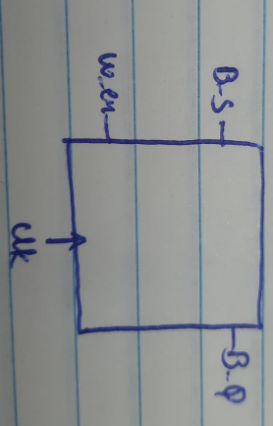


Figure 6- B storage unit block diagram

Parameter : b\_d\_w: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| B\_S | 4 | B storage unit input |
| w\_en, clk |  | Enable and the clock for the storage to get triggered |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| B\_Q | 4 | output |

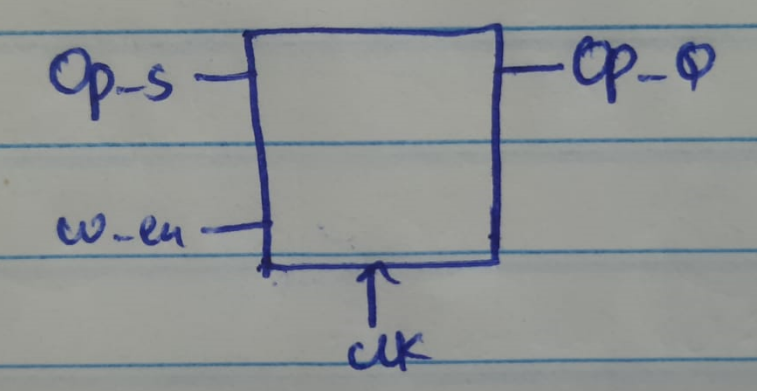
Subcomponent: Op\_storageunit

Figure 7- OP storage unit block diagram

Here we have used the custom package, all\_oper ,w hich is being used even for the CU operation, so that, both operate and the same time, when the OP value is selected and it is stored in the OP storage

Parameter : op\_d\_w: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| Op\_S | 4 | Op storage unit input will be taken from the custom pack,all\_oper |
| w\_en, clk |  | Enable and the clock for the storage to get triggered |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| OP\_S\_Q | 4 | output |

Subcomponent: y\_storage unit

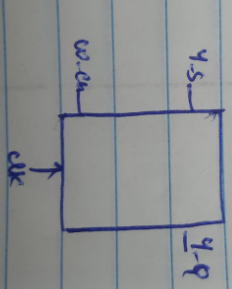


Figure 8- Y storage unit block diagram

Parameter : y\_d\_w: integer:= 4

Input ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| Y\_S | 4 | Input to storage |
| w\_en, clk |  | Enable and the clock for the storage to get triggered |

Output ports:

|  |  |  |
| --- | --- | --- |
| Port name | Bit width | Purpose |
| Y\_S\_Q | 4 | output |

## Design explanation

### Functionality

In this design we have used arrays concept, while port mapping each and every related mux and cu with the storage units. SO, for this, at first we have created a custom package, for the Cu overall operations along with the mux 2x1 and mux4x1, defined the arrays for all of them in the 1D and 2D format.

For the top module, external inputs, we have used the 1D array and the overall outputs also we have used the 1D array only, for rest operations, storage outputs, cu outputs, enabled and mux outputs we have used 2D arrays, clk we have used std\_logic itself.

So here the feedback loop is given from the row 3 to the row 0 , outputs of the row 0 is given to the input of mux for selection in the row2 along with the row 1 and cu20 as the adjacent output to cu21, cu22, cu23.

Row 1 output is given to the row3 as the input along with the row2 and the cu30 output to the cu31,cu32,cu33 as the input to the muxes.

All the above connections shows the multi level hierarchical connections. For this we have used mux 2x1 and mux 4x1 , where 2x1 is used as 4x4 matrix and 4x1 as the 3x3 matrix.

Individual Mux output is selected and given to the a storage and b storage independently and given to the CU, where the CU output is given as the input to the Y storage , where the output will be stored and it wait. All A ,B Op storage will operate at the same time, till then they will wait.

Once after the CU00 output it done, it is given to the adjacent CU of that row, and the same process for the row 1, row2, row3 also.

Due to all these wait times, each run will have different time delays to operate and complete, Delay will be different for each CU and for the entire row also.

During the first run, the values of row 3 will be stored at y\_storage 30,31,32,33 respectively and these values are used for the next run as the input for the row0, because of the feedback connections, but in the run2 all other values will be updated, due to the new operations taking place.

we have used the CU with below operations

|  |  |  |
| --- | --- | --- |
|  | Operation | Description |
| ANDo | AND | Performs bitwise **AND** operation |
| Oro | OR | Performs bitwise **OR** operation |
| NANDo | NAND | Performs bitwise **NAND** operation |
| NORo | NOR | Performs bitwise **NOR** operation |
| XORo | XOR | Performs bitwise **XOR** operation |
| XNORo | XNOR | Performs bitwise **XNOR** operation |
| ADDITIONo | Addition | Performs **addition** operation |
| SUBTRACTIONo | Subtraction | Performs **subtraction** operation |
| MULTIPLICATIONo | Multiplication | Performs **multiplication** operation |
| Greater\_thano | Greater\_than | Checks if **A** is greater **B** |
| Less\_thano | Less\_than | Checks if **A** is less than **B** |
| SLLo | SLL\_Shift\_output | shift left operation on **A**by the shift amount of **B** |
| SRLo | SRL\_Shift\_output | Perform arithmetic shift left operation on **A**by the shift amount of **B** |
| SRAo | SRA\_Shift\_output | Perform arithmetic shift right operation on **A**by the shift amount of **B** |
| ROLo | ROL\_Shift\_output | Perform rotate shift left operation on **A**by the shift amount of **B** |
| RORo | ROR\_Shift\_output | Perform rotate shift right operation on **A**by the shift amount of **B** |
| EQUALo | equal | Checks if **A** is equal **B** |
| NOTEQUALo | notEqual | Checks if **A** Is not equal **B** |
| Greater\_tha\_equal\_  too | Greater than or equal  to | Checks if **A** is greater than or equal to **B** |
| Less\_than\_equal\_t  oo | Less\_than\_equal\_to | Checks if **A less than equal to B** |
| SLAo | SLA\_Shift\_output | shift right operation on **A** by the shift amount of **B** |

*Table 1: CU operations*

Mux operations for all the CU:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Select input as per mux 4x1 | Select line selecting the other inputs basing on operation | Select line operation | output |
| cu00\_mux\_a | twoInput0 | top\_input1\_A(0) | T0o | mux\_two\_out\_A(0)(0) |
|  | twoInput1 | store\_y\_CU(3)(0) | T1o |  |
| cu00\_mux\_b | twoInput0 | top\_input2\_B(0) | T0o | mux\_two\_out\_B(0)(0) |
|  | twoInput1 | store\_y\_CU(3)(0) | T1o |  |
|  |  |  |  |  |
| cu01\_mux\_a | Input0 | store\_y\_CU(3)(1) | I0o | mux\_four\_out\_A(0)(0) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input1\_A(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| cu01\_mux\_b | Input0 | store\_y\_CU(3)(1) | I0o | mux\_four\_out\_B(0)(0) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input2\_B(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| cu02\_mux\_a | Input0 | store\_y\_CU(3)(2) | I0o | mux\_four\_out\_A(0)(1) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input1\_A(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| cu02\_mux\_b | Input0 | store\_y\_CU(3)(2) | I0o | mux\_four\_out\_B(0)(1) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input2\_B(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu03\_mux\_a | Input0 | store\_y\_CU(3)(3) | I0o | mux\_four\_out\_A(0)(2) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input1\_A(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| cu03\_mux\_b | Input0 | store\_y\_CU(3)(3) | I0o | mux\_four\_out\_B(0)(2) |
|  | Input1 | store\_y\_CU(0)(0) | I1o |  |
|  | Input2 | top\_input2\_B(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu10\_mux\_a | twoInput0 | (others => '0') | T0o | mux\_two\_out\_A(1)(0) |
|  | twoInput1 | store\_y\_CU(0)(0) | T1o |  |
| Cu10\_mux\_b | twoInput0 | (others => '0') | T0o | mux\_two\_out\_B(1)(0) |
|  | twoInput1 | store\_y\_CU(0)(0) | T1o |  |
|  |  |  |  |  |
| Cu11\_mux\_a | twoInput0 | store\_y\_CU(1)(0) | T0o | mux\_two\_out\_A(1)(1) |
|  | twoInput1 | store\_y\_CU(0)(1) | T1o |  |
| Cu11\_mux\_b | twoInput0 | store\_y\_CU(0)(1) | T0o | mux\_two\_out\_B(1)(1) |
|  | twoInput1 | store\_y\_CU(1)(0) | T1o |  |
|  |  |  |  |  |
| Cu12\_mux\_a | twoInput0 | store\_y\_CU(1)(0) | T0o | mux\_two\_out\_A(1)(2) |
|  | twoInput1 | store\_y\_CU(0)(2) | T1o |  |
| Cu12\_mux\_b | twoInput0 | store\_y\_CU(0)(2) | T0o | mux\_two\_out\_B(1)(2) |
|  | twoInput1 | store\_y\_CU(1)(0) | T1o |  |
|  |  |  |  |  |
| Cu13\_mux\_a | twoInput0 | store\_y\_CU(1)(0) | T0o | mux\_two\_out\_A(1)(3) |
|  | twoInput1 | store\_y\_CU(0)(3) | T1o |  |
| Cu13\_mux\_b | twoInput0 | store\_y\_CU(0)(3) | T0o | mux\_two\_out\_B(1)(3) |
|  | twoInput1 | store\_y\_CU(1)(0) | T1o |  |
|  |  |  |  |  |
| Cu20\_mux\_a | twoInput0 | store\_y\_CU(0)(0) | T0o | mux\_two\_out\_A(2)(0) |
|  | twoInput1 | store\_y\_CU(1)(0) | T1o |  |
| Cu20\_mux\_b | twoInput0 | store\_y\_CU(1)(0) | T0o | mux\_two\_out\_B(2)(0) |
|  | twoInput1 | store\_y\_CU(0)(0) | T1o |  |
|  |  |  |  |  |
| Cu21\_mux\_a | Input0 | store\_y\_CU(0)(1) | I0o | mux\_four\_out\_A(1)(0) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu21\_mux\_b | Input0 | store\_y\_CU(1)(1) | I0o | mux\_four\_out\_B(1)(0) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(0)(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu22\_mux\_a | Input0 | store\_y\_CU(0)(2) | I0o | mux\_four\_out\_A(1)(1) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu22\_mux\_b | Input0 | store\_y\_CU(1)(2) | I0o | mux\_four\_out\_B(1)(1) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(0)(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu23\_mux\_a | Input0 | store\_y\_CU(0)(3) | I0o | mux\_four\_out\_A(1)(2) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu23\_mux\_b | Input0 | store\_y\_CU(1)(3) | I0o | mux\_four\_out\_B(1)(2) |
|  | Input1 | store\_y\_CU(2)(0) | I1o |  |
|  | Input2 | store\_y\_CU(0)(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu30\_mux\_a | twoInput0 | store\_y\_CU(1)(0) | T0o | mux\_two\_out\_A(3)(0) |
|  | twoInput1 | store\_y\_CU(2)(0) | T1o |  |
| Cu30\_mux\_b | twoInput0 | store\_y\_CU(2)(0) | T0o | mux\_two\_out\_B(3)(0) |
|  | twoInput1 | store\_y\_CU(1)(0) | T1o |  |
|  |  |  |  |  |
| Cu31\_mux\_a | Input0 | store\_y\_CU(1)(1) | I0o | mux\_four\_out\_A(2)(0) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(2)(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu31\_mux\_b | Input0 | store\_y\_CU(2)(1) | I0o | mux\_four\_out\_B(2)(0) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(1) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu32\_mux\_a | Input0 | store\_y\_CU(1)(2) | I0o | mux\_four\_out\_A(2)(1) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(2)(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu32\_mux\_b | Input0 | store\_y\_CU(2)(2) | I0o | mux\_four\_out\_B(2)(1) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(2) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
|  |  |  |  |  |
| Cu33\_mux\_a | Input0 | store\_y\_CU(1)(3) | I0o | mux\_four\_out\_A(2)(2) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(2)(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |
| Cu33\_mux\_b | Input0 | store\_y\_CU(2)(3) | I0o | mux\_four\_out\_B(2)(2) |
|  | Input1 | store\_y\_CU(3)(0) | I1o |  |
|  | Input2 | store\_y\_CU(1)(3) | I2o |  |
|  | Input3 | (others => '0') | I3o |  |

*Table 3: mux operations*

Using all these selected output of the mux CU30,CU31,CU32,CU33 will be operated and gives the result in 4bits each.

# Results

## Generated Schematics

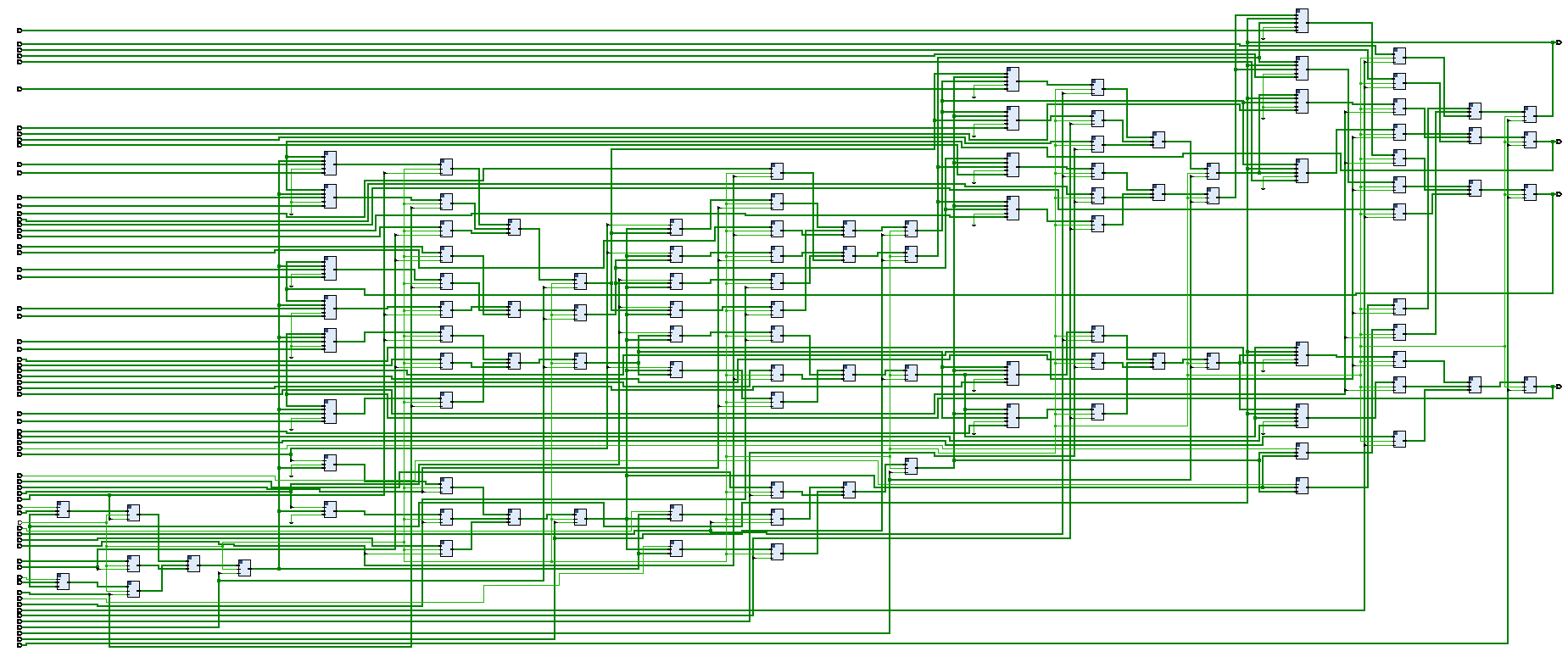
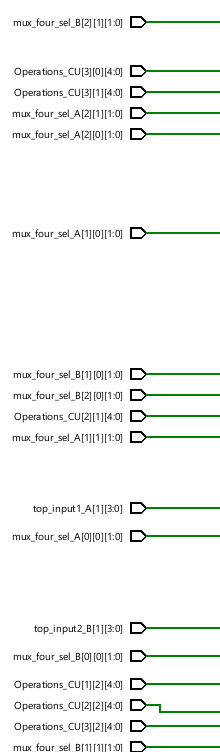


Figure 9 – top\_module schematic

This is the schematic for the top module where the final row 3 has the feed back, which is clearly visisble in the figure at the last side, for this we have used the arrays and the port mapping technique. top\_input1\_A, top\_input2\_B are the external input, while using them in arrays we have declared as top\_input1\_A(0), top\_input1\_A(1), top\_input1\_A(2), top\_input1\_A(3), top\_input2\_B(0), top\_input2\_B(1), top\_input2\_B(2), top\_input2\_B(3), this is a 1d array representation for the inputs. For 2D arrays representation, we have used as (0)(0) and defined as such depending in the row and column.



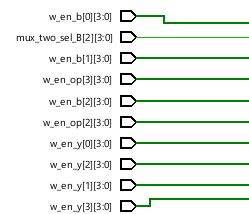


Figure 10 – top\_module schematic- part 1

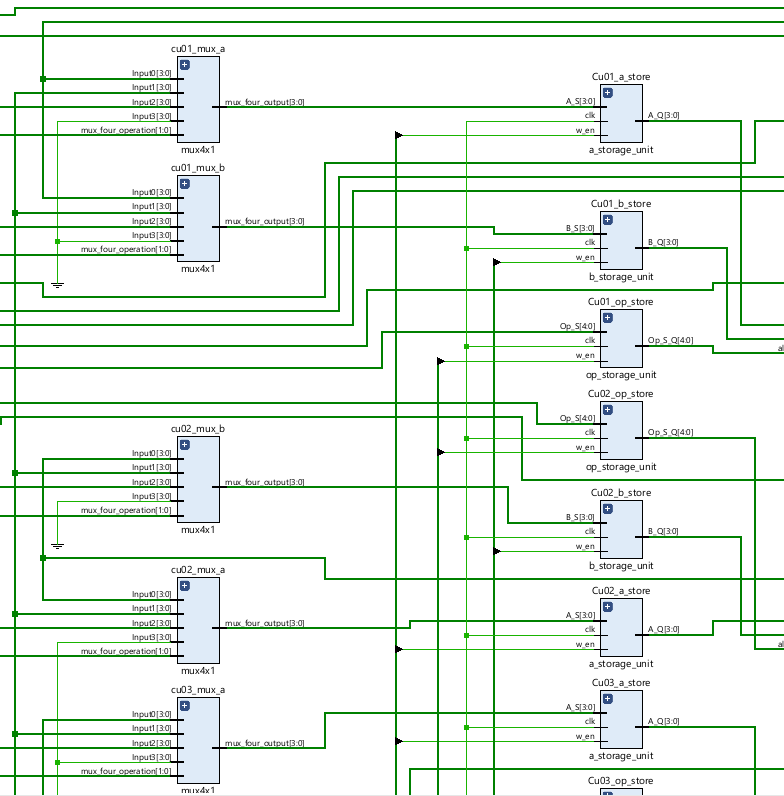
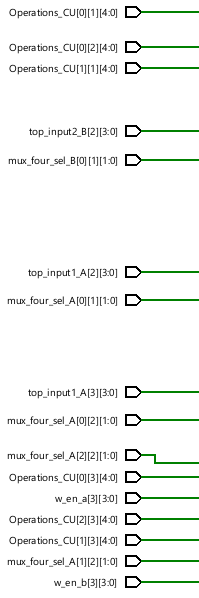


Figure 11 – top\_module schematic – part 2



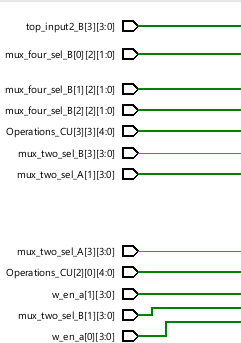


Figure 12 – top\_module schematic- part 3

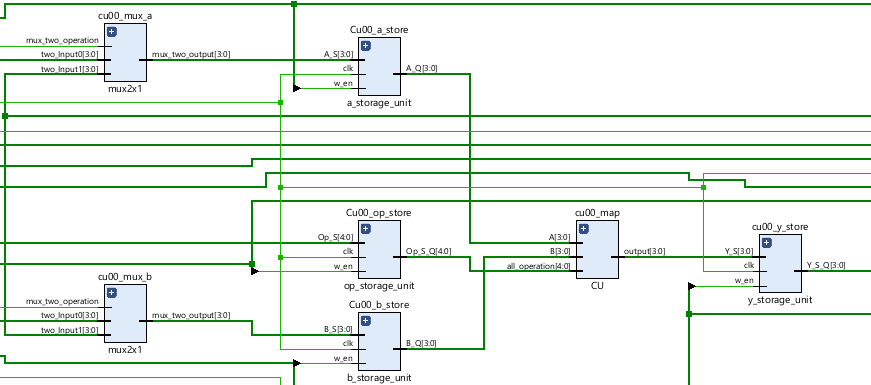


Figure 13 – top\_module schematic- part 4

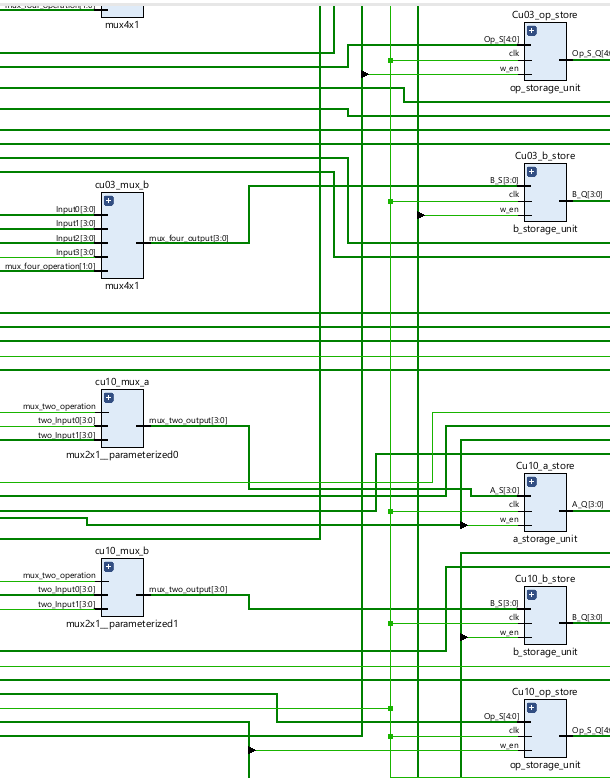


Figure 14 – top\_module schematic- part 5

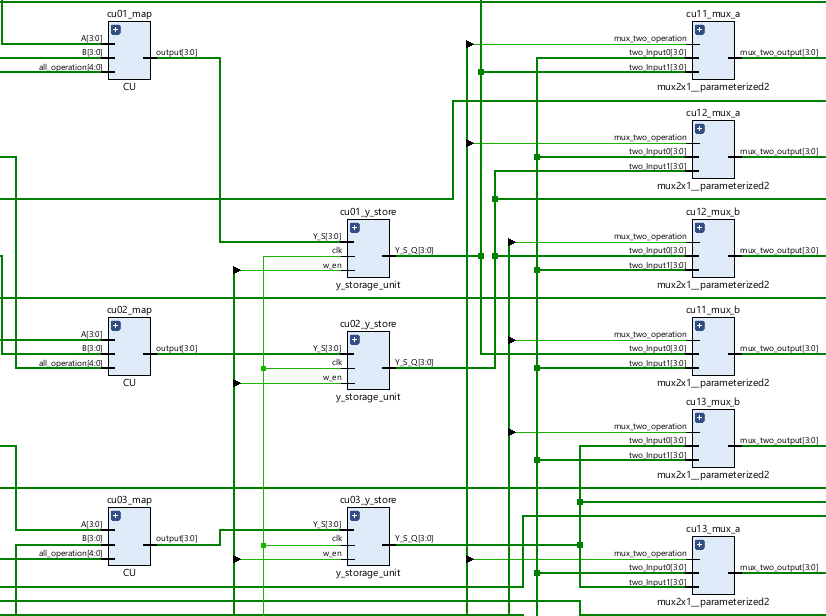


Figure 15 – top\_module schematic- part 6

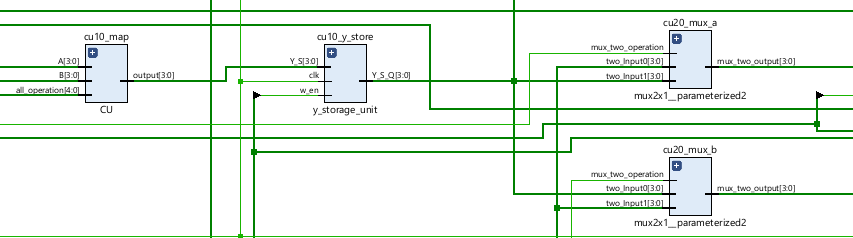


Figure 16 – top\_module schematic- part 7

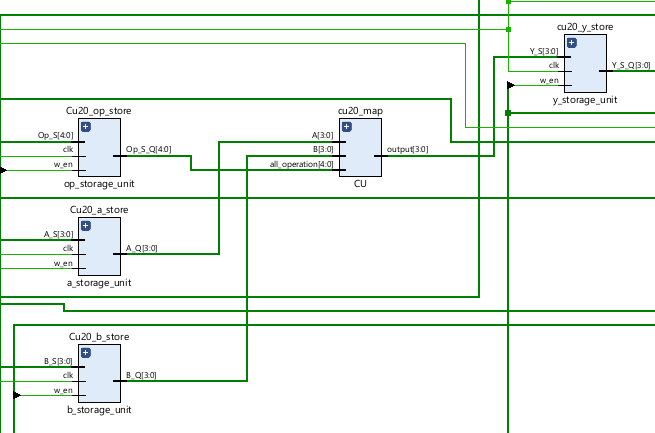


Figure 17 – top\_module schematic- part 8

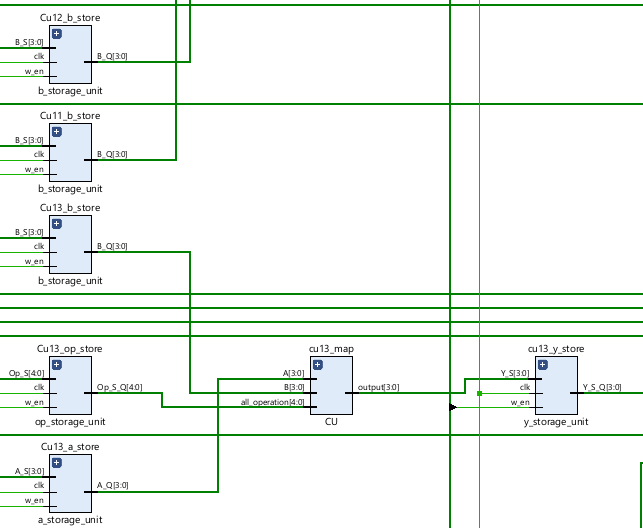


Figure 18 – top\_module schematic- part 9

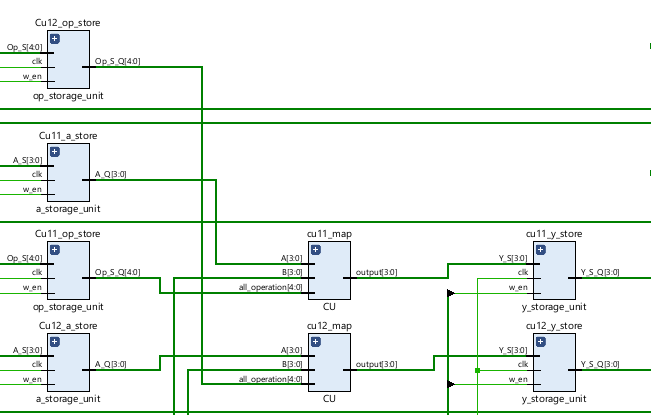


Figure 19 – top\_module schematic- part 10

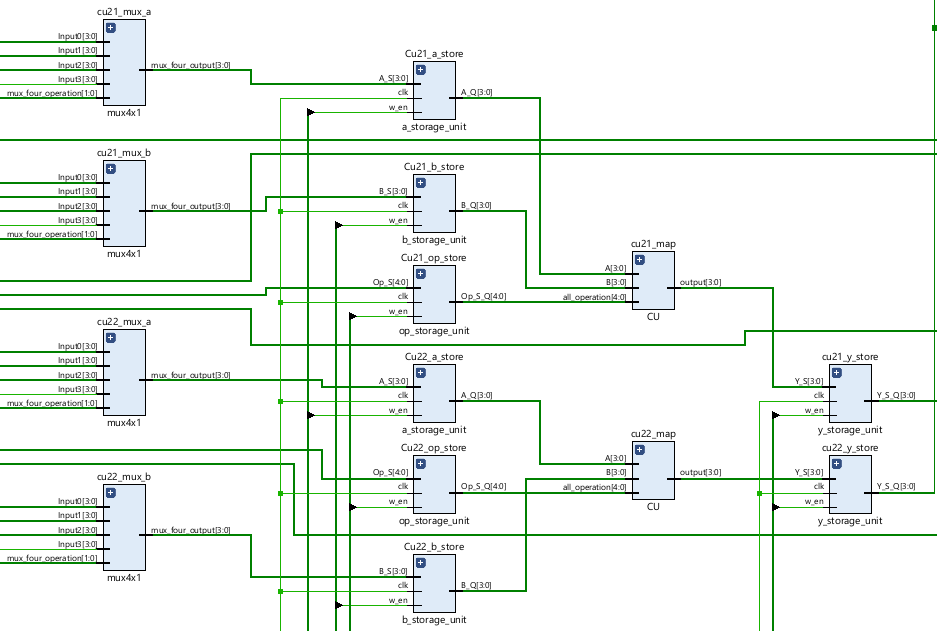


Figure 20 – top\_module schematic- part 11

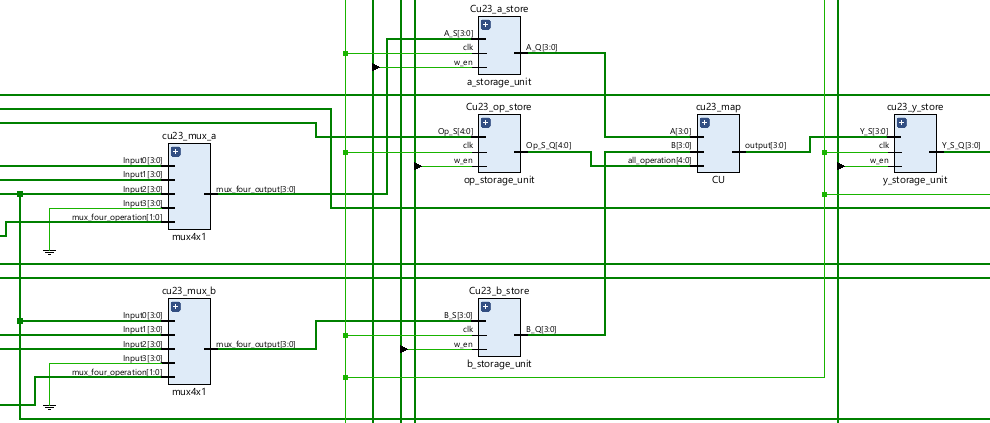


Figure 21 – top\_module schematic- part 12

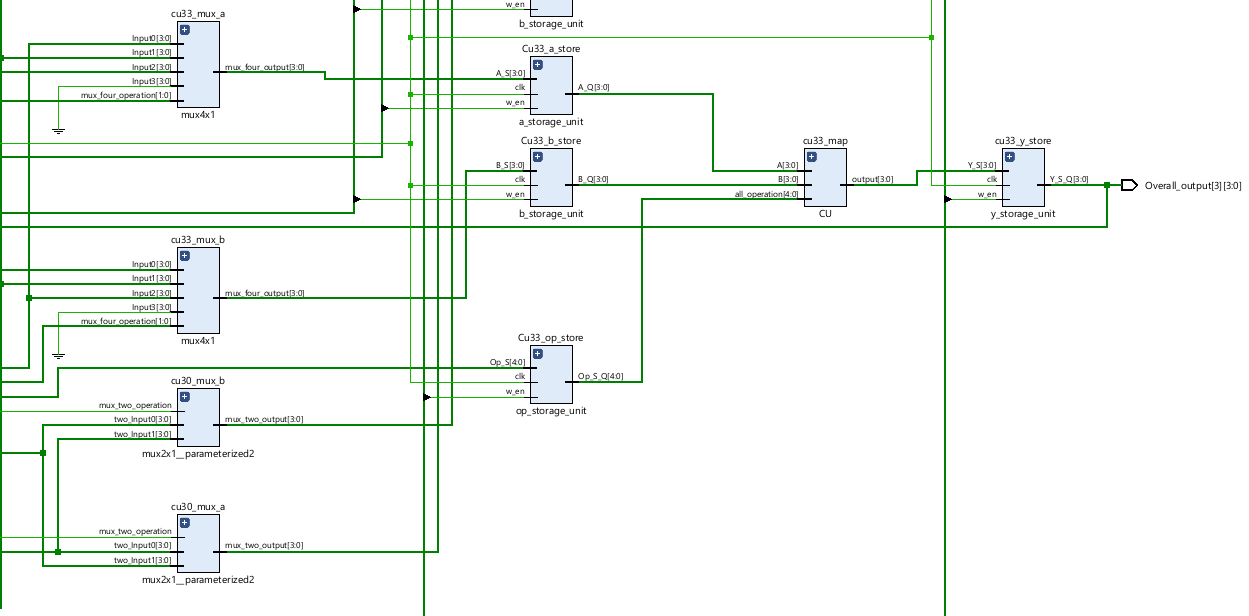


Figure 22 – top\_module schematic- part 13

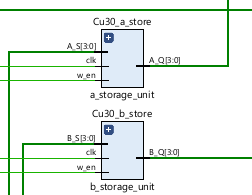


Figure 23 – top\_module schematic- part 14

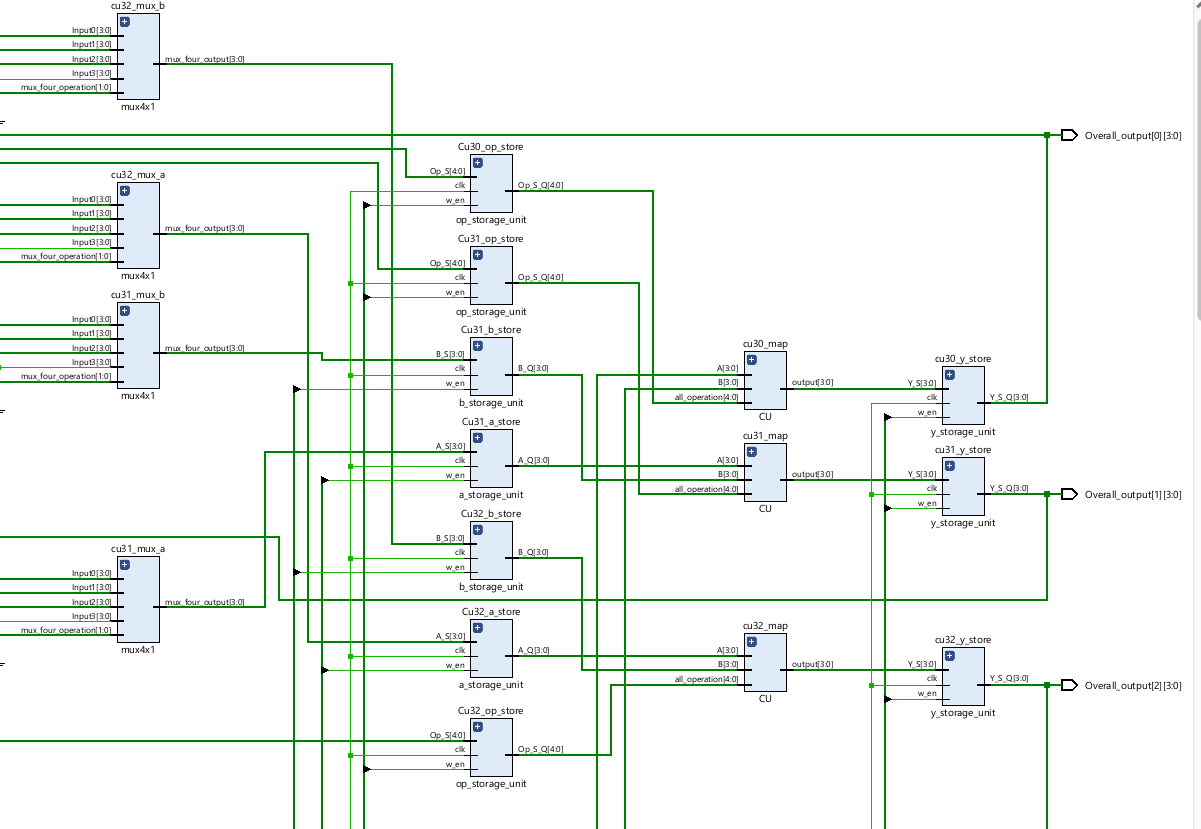


Figure 24 – top\_module schematic- part 15

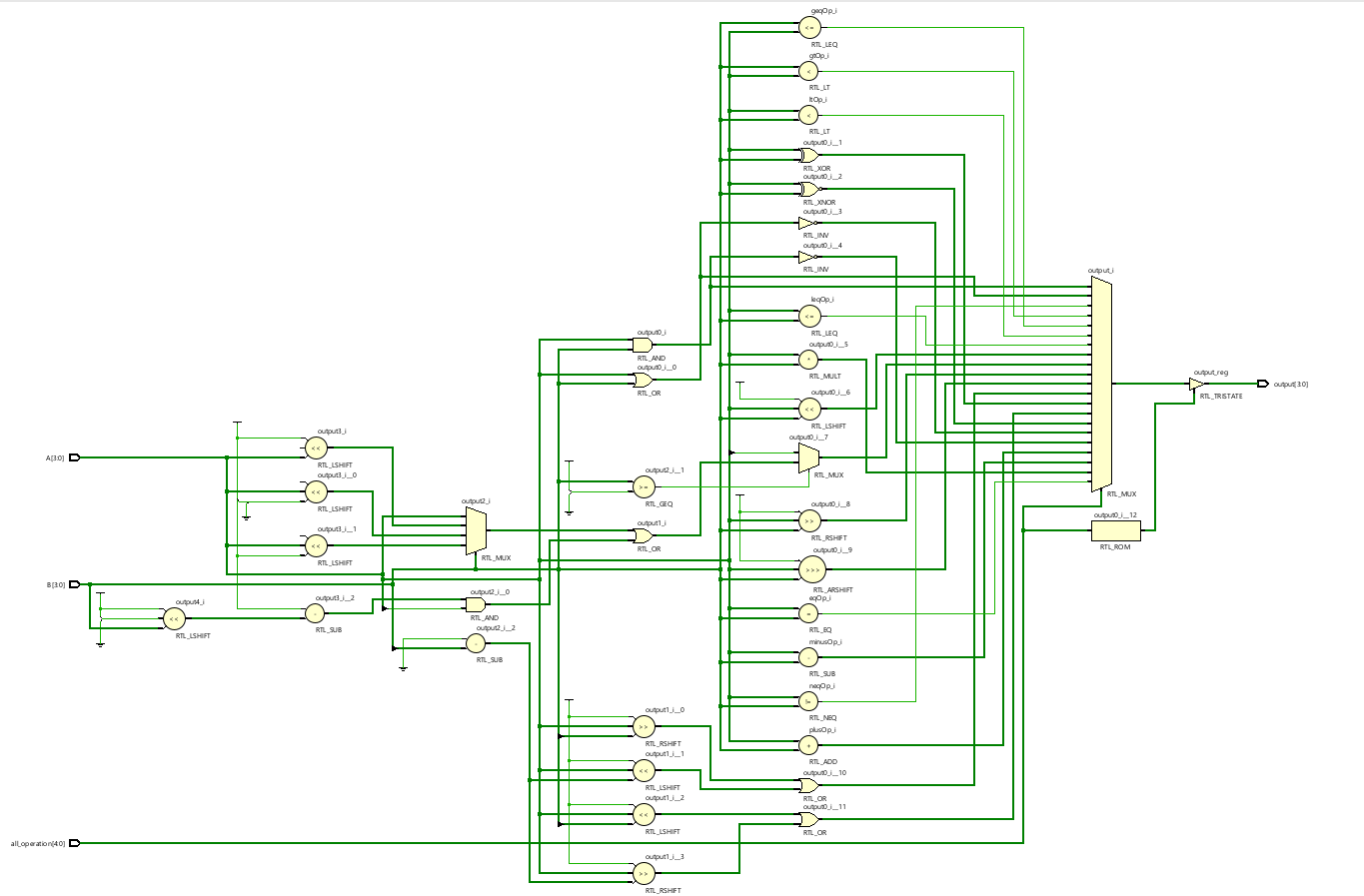


Figure25 – CU schematic

In the CU we have used A,B as the input and the output is declared as output

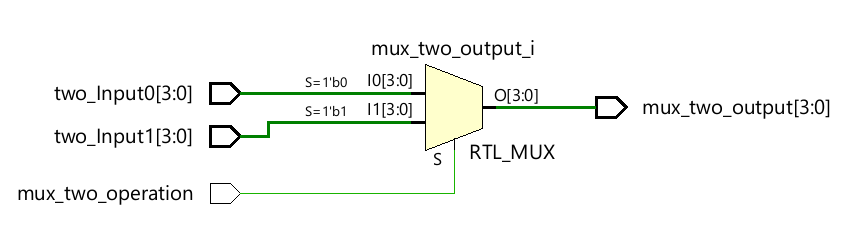


Figure26 – 2x1 mux schematic

For the mux 2x1 we have used input as the two\_input0 and two\_input1 , where mus\_two\_operation is the select line and the mux\_two\_output is the output

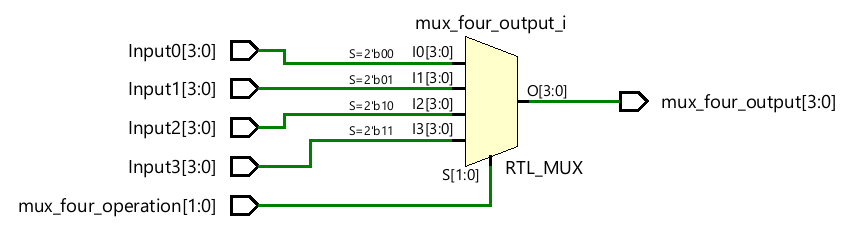


Figure 27 – 4x1 schematic

Input0, input1,input2,input3 are the inputs to the 4x1 mux, where the mux\_four\_operation is the select line and the output is mux\_four\_output.

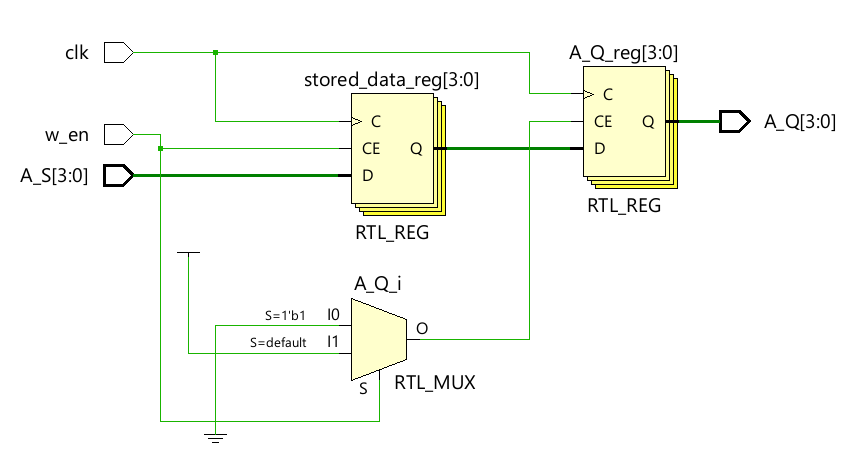


Figure 28 – a\_storageunit schematic

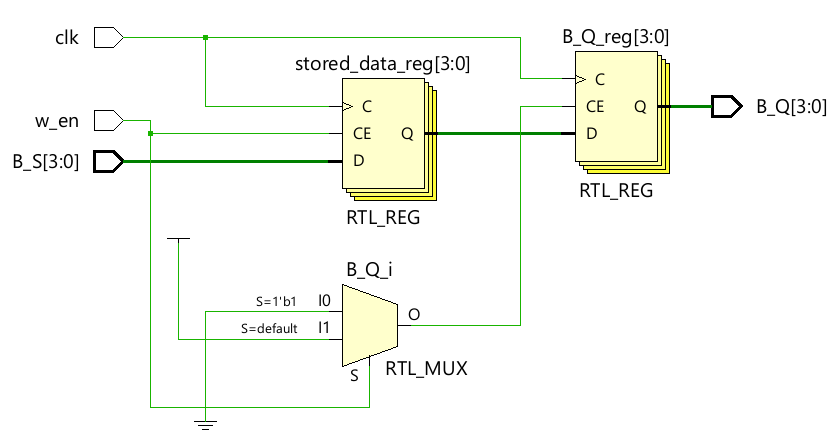


Figure 29 – b\_storageunit schematic

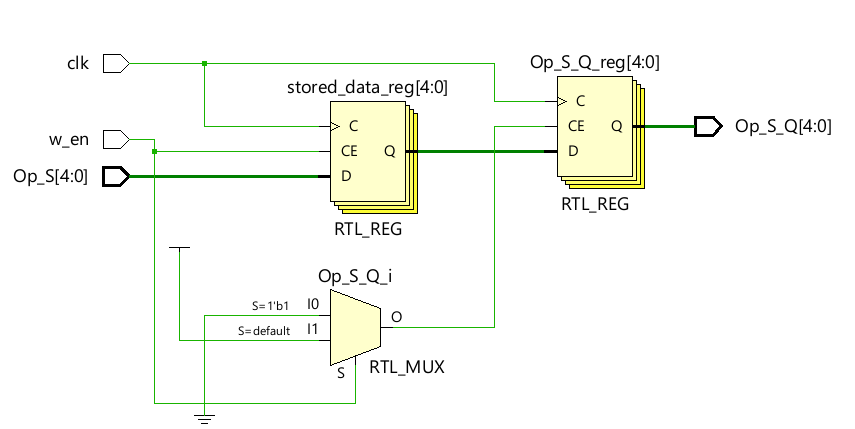


Figure 30 – op\_storageunit schematic

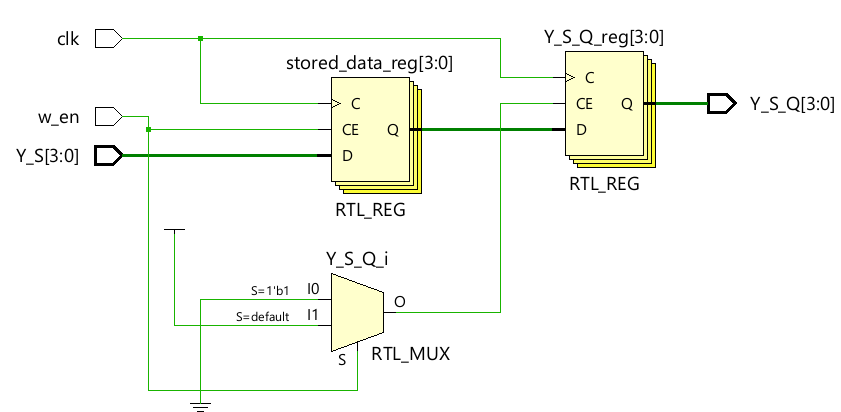


Figure 31 – y\_storageunit schematic

## Waveforms

Figure 32 – CU testbench

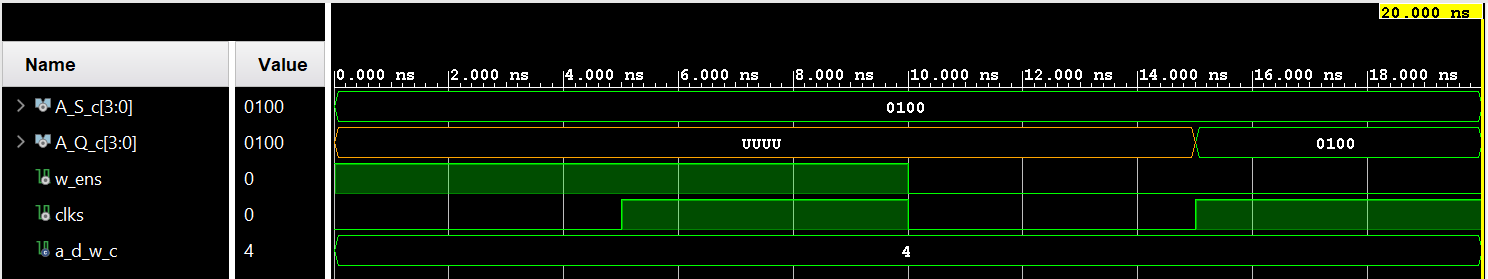


Figure 33 – a\_storage\_unit testbench

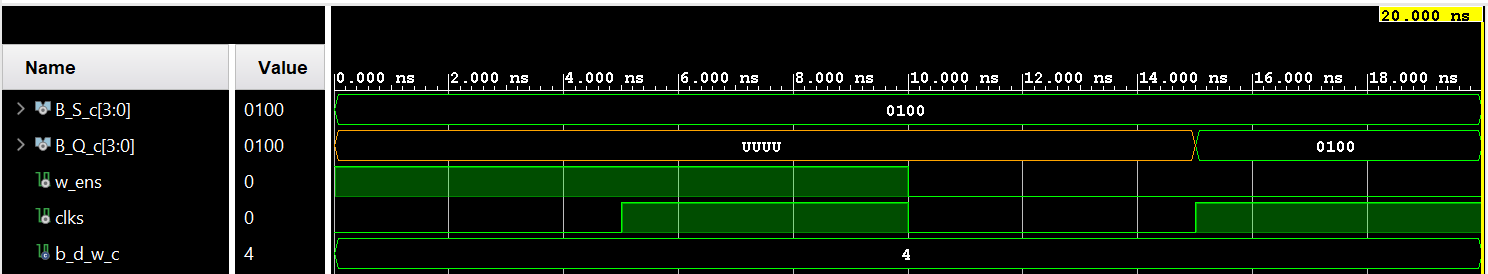


Figure 34 – b\_storage\_unit testbench

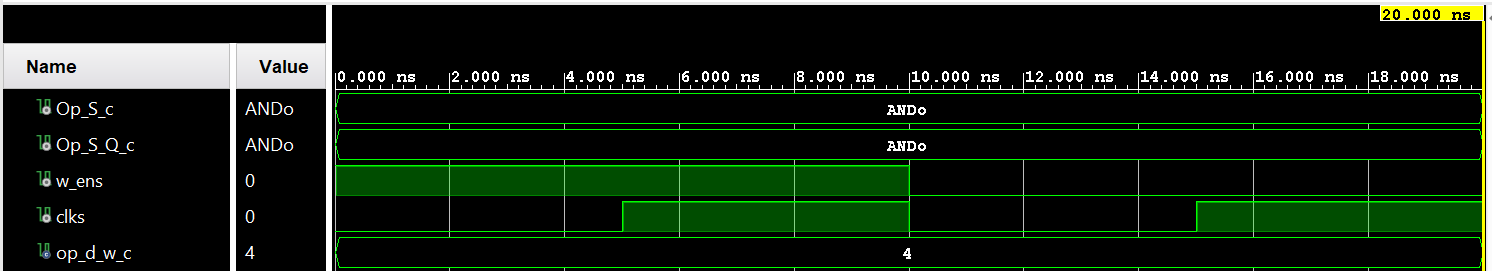


Figure 35 – op\_storage\_unit testbench

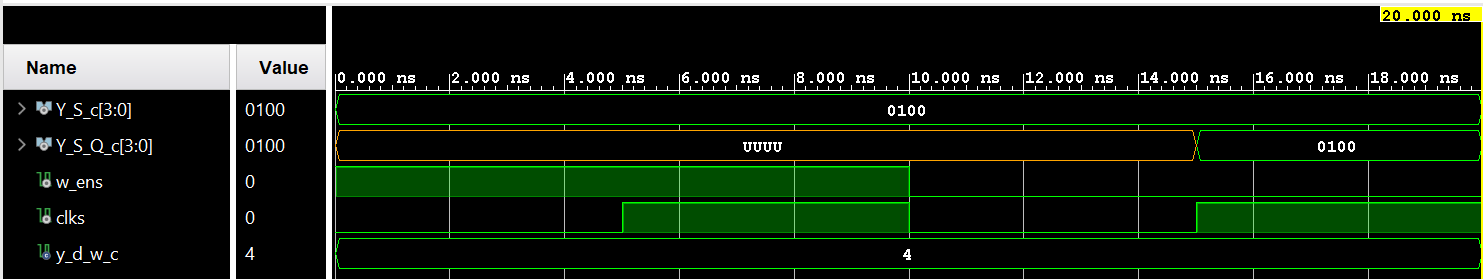


Figure 36 – y\_storage\_unit testbench

## 

Figure 37 – 2x1MUX testbench

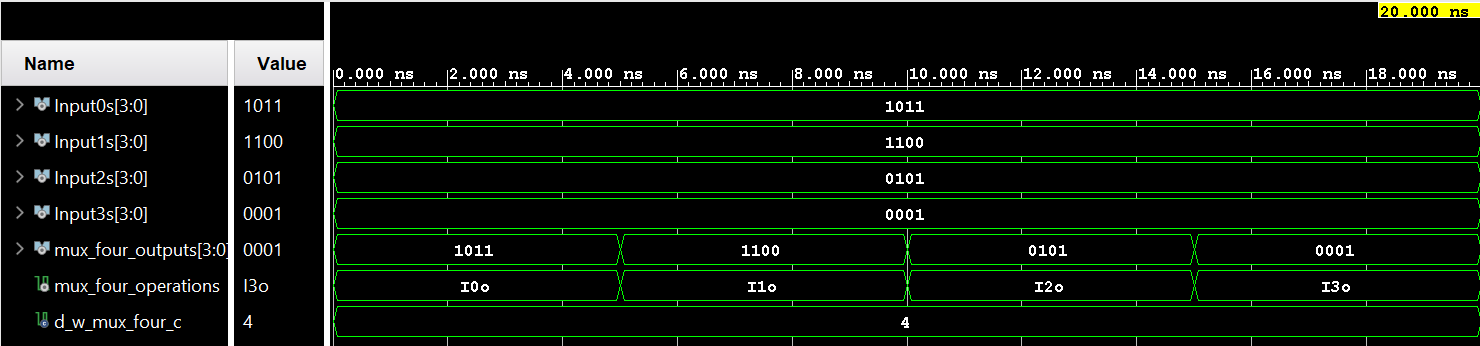


Figure 38 – 4x1MUX testbench

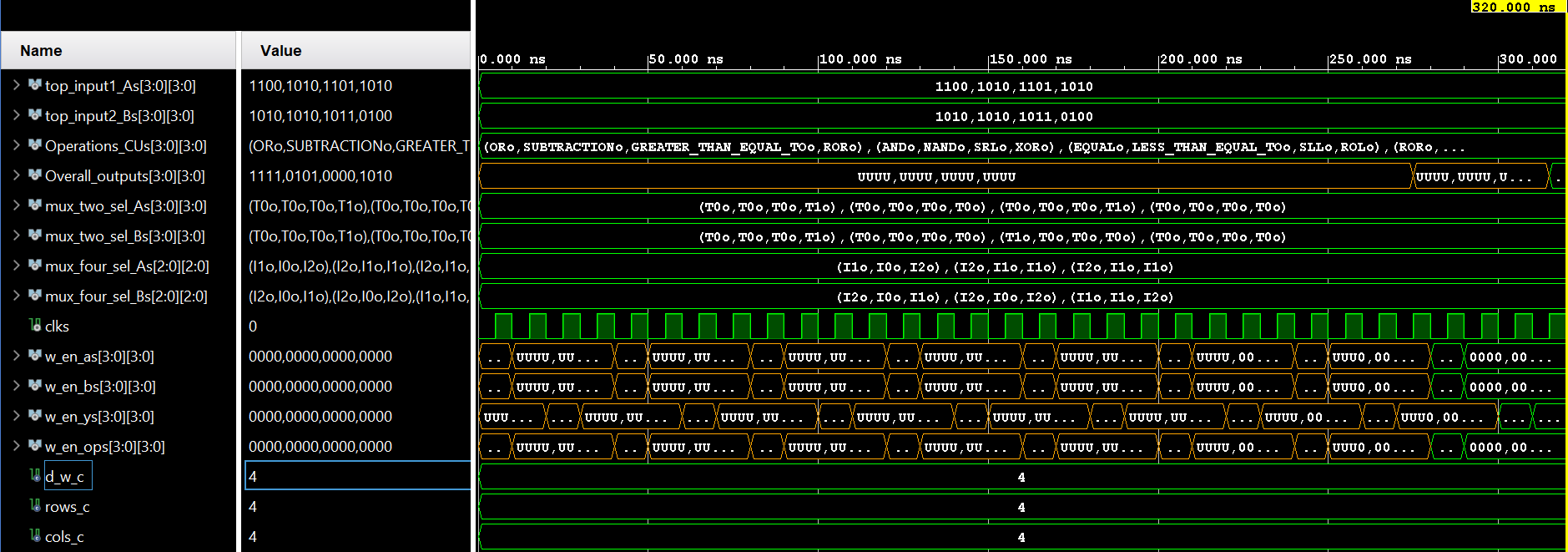
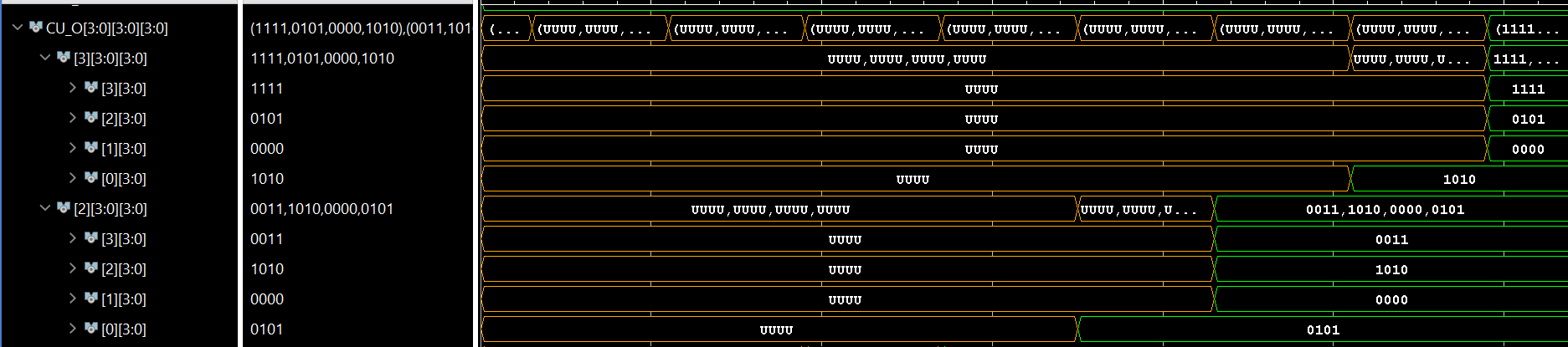


Figure 39 – run 1- part 1



*Figure 40 – run 1- part 2*

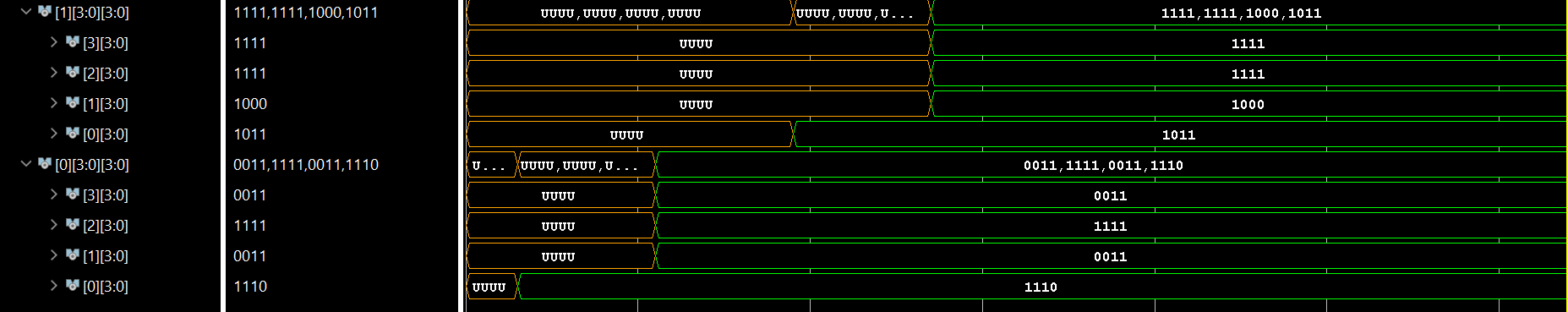


Figure 41 – run 1- part 3

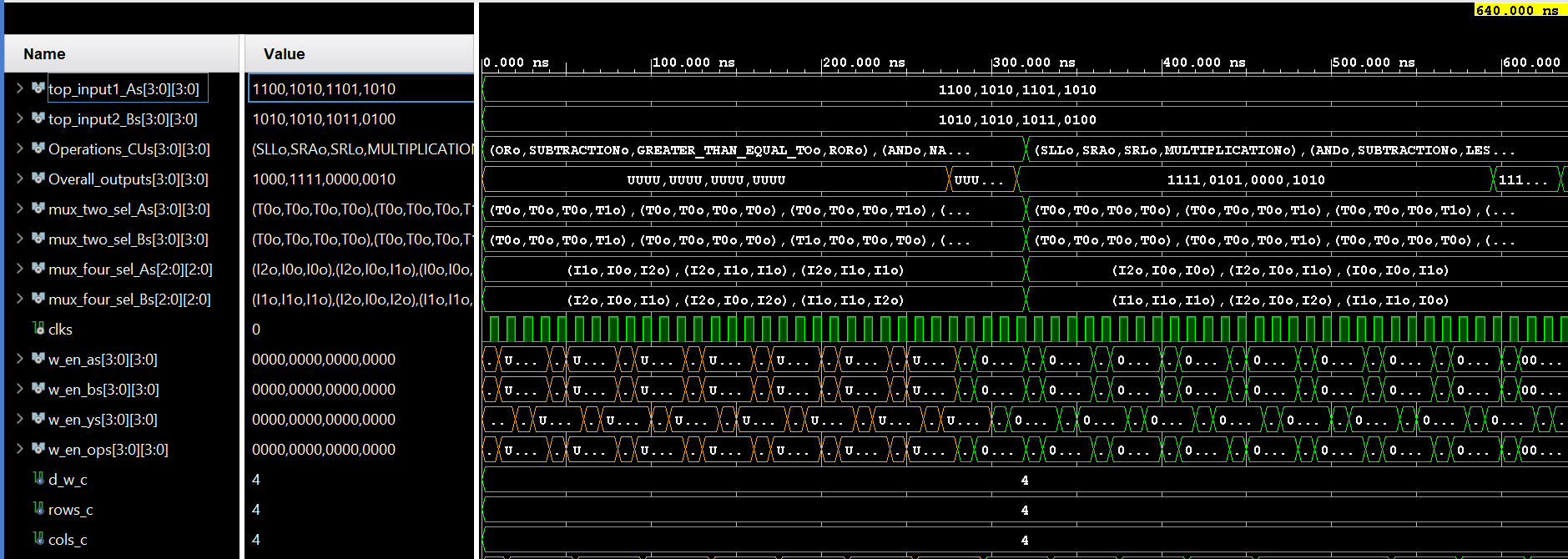


Figure 42 – run 2- part 1

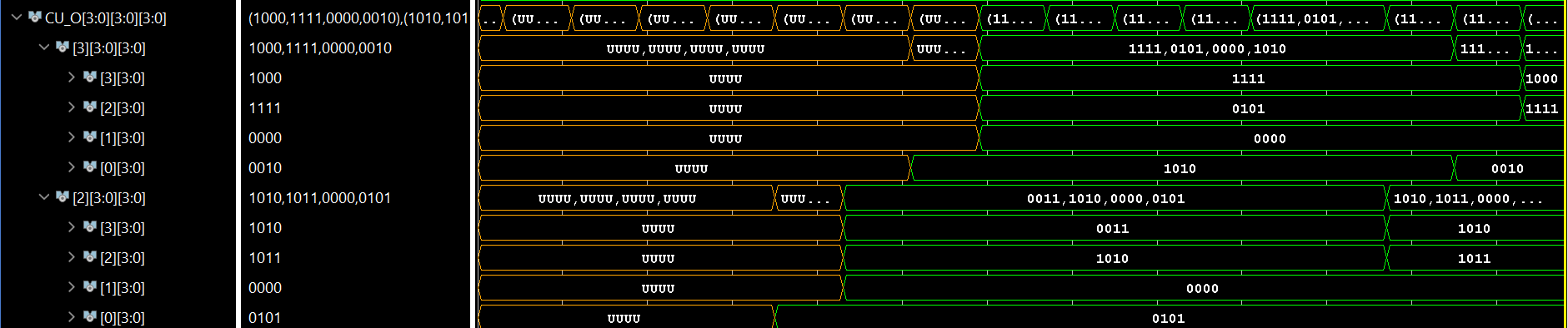


Figure 43 – run 2- part 2

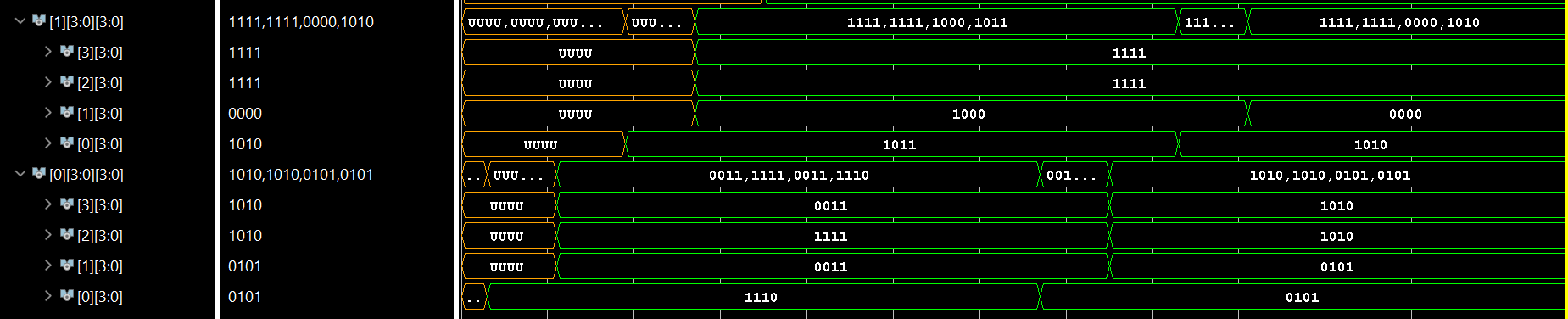


Figure 44 – run 2- part 3

## Table/Calculations

### Overall Design

Run1 DFG1 :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CU# | SourceA | SourceB | Input1 | Input2 | Oper | Calculated  Op | Simulated Op | Match |
| CU (0,0) | External input | External input | 1010 | 0100 | ADD | 1110 | 1110 | Yes |
| CU (0,1) | CU(0,0) | External input | 1110 | 1011 | SUB | 0011 | 0011 | Yes |
| CU(0,2) | CU(0,0) | CU(0,0) | 1110 | 1110 | XNOR | 1111 | 1111 | Yes |
| CU (0,3) | External input | CU(0,0) | 1100 | 1110 | ROR | 0011 | 0011 | Yes |
| CU (1,0) | CU(0,0) | CU(0,0) | 1110 | 1110 | ROL | 1011 | 1011 | Yes |
| CU (1,1) | CU(1,0) | CU(0,1) | 1011 | 0011 | LSL | 1000 | 1000 | Yes |
| CU(1,2) | CU(1,0) | CU(0,2) | 1011 | 1111 | LTE | 1111 | 1111 | Yes |
| CU (1,3) | CU(1,0) | CU(1,0) | 1011 | 1011 | EQ | 1111 | 1111 | Yes |
| CU (2,0) | CU(0,0) | CU(1,0) | 1110 | 1011 | XOR | 0101 | 0101 | Yes |
| CU (2,1) | CU(2,0) | CU(0,1) | 0101 | 0011 | LSR | 0000 | 0000 | Yes |
| CU (2,2) | CU(2,0) | CU(1,2) | 0101 | 1111 | NAND | 1010 | 1010 | Yes |
| CU (2,3) | CU(1,3) | CU(0,3) | 1111 | 0011 | AND | 0011 | 0011 | Yes |
| CU (3,0) | CU(2,0) | CU(1,0) | 0101 | 1011 | ROR | 1010 | 1010 | Yes |
| CU (3,1) | CU(2,1) | CU(3,0) | 0000 | 1010 | GTE | 0000 | 0000 | Yes |
| CU(3,2) | CU(1,2) | CU(2,2) | 1111 | 1010 | SUB | 0101 | 0101 | Yes |
| CU (3,3) | CU(3,0) | CU(1,3) | 1010 | 1111 | OR | 1111 | 1111 | Yes |

Run2 DFG2 :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CU# | SourceA | SourceB | Input1 | Input2 | Oper | Calculated  Op | Simulated Op | Match |
| CU (0,0) | CU(3,0) | CU(3,0) | 1010 | 1010 | NOR | 0101 | 0101 | Yes |
| CU (0,1) | CU(0,0) | CU(3,1) | 0101 | 0000 | XOR | 0101 | 0101 | Yes |
| CU(0,2) | CU(3,2) | CU(0,0) | 0101 | 0101 | NAND | 1010 | 1010 | Yes |
| CU (0,3) | CU(3,3) | CU(0,0) | 1111 | 0101 | SUB | 1010 | 1010 | Yes |
| CU (1,0) | CU(0,0) | CU(0,0) | 0101 | 0101 | ADD | 1010 | 1010 | Yes |
| CU (1,1) | CU(1,0) | CU(0,1) | 1010 | 0101 | EQ | 0000 | 0000 | Yes |
| CU(1,2) | CU(1,0) | CU(0,2) | 1010 | 1010 | XNOR | 1111 | 1111 | Yes |
| CU (1,3) | CU(1,0) | CU(0,3) | 1010 | 1010 | LTE | 1111 | 1111 | Yes |
| CU (2,0) | CU(1,0) | CU(0,0) | 1010 | 0101 | ROL | 0101 | 0101 | Yes |
| CU (2,1) | CU(2,0) | CU(0,1) | 0101 | 0101 | LT | 0000 | 0000 | Yes |
| CU (2,2) | CU(0,2) | CU(1,2) | 1010 | 1111 | SUB | 1011 | 1011 | Yes |
| CU (2,3) | CU(1,3) | CU(0,3) | 1111 | 1010 | AND | 1010 | 0100 | Yes |
| CU (3,0) | CU(1,0) | CU(2,0) | 1010 | 0101 | MULT | 0010 | 0010 | Yes |
| CU (3,1) | CU(1,1) | CU(3,0) | 0000 | 0010 | LSR | 0000 | 0000 | Yes |
| CU(3,2) | CU(1,2) | CU(3,0) | 1111 | 0010 | ASR | 1111 | 1111 | Yes |
| CU (3,3) | CU(2,3) | CU(3,0) | 1010 | 0010 | LSL | 1000 | 1000 | Yes |

As can be seen by the table, the waveform output values for both test cases match the calculated values. This means the design works properly.

### Subcomponent test cases

***CU:***

Test Case cu:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | OPERATION | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| 1010 | 0011 | ANDo | 0010 | 0010 | Pass |
| ORo | 1011 | 1011 | Pass |
| NANDo | 1101 | 1101 | Pass |
| NORo | 0100 | 0100 | Pass |
| XORo | 1001 | 1001 | Pass |
| XNORo | 0110 | 0110 | Pass |
| ADDITIONo | 1101 | 1101 | Pass |
| SUBTRACTIONo | 0111 | 0111 | Pass |
|  | MULTIPLICATIONo | 1110 | 1110 | Pass |
|  | GREATER\_THANo | 1111 | 1111 | Pass |
| LESS\_THANo | 0000 | 0000 | Pass |
|  | SLLo | 0000 | 0000 | Pass |
| SRLo | 0001 | 0001 | Pass |
| SRAo | 1111 | 1111 | Pass |
| ROLo | 0101 | 0101 | Pass |
| RORo | 0101 | 0101 | Pass |
| EQUALo | 0000 | 0000 | Pass |
| NOTEQUALo | 1111 | 1111 | Pass |
| GREATER\_THAN\_EQUAL\_TOo | 1111 | 1111 | Pass |
|  |  | LESS\_THAN\_EQUAL\_TOo | 0000 | 0000 | Pass |
|  |  | SLAo | 0000 | 0000 | Pass |

### MUX 4X1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | SELECT LINE | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| INPUT0 = 1011 | I0o | 1011 | 1011 | PASS |
| INPUT1 = 1100 | I1o | 1100 | 1100 | PASS |
| INPUT2 = 0101 | I2o | 0101 | 0101 | PASS |
| INPUT3 = 0001 | I3o | 0001 | 0001 | PASS |

### MUX 2X1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | SELECT LINE | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| two\_Input0 = 1011 | T0o | 1011 | 1011 | PASS |
| two\_Input1 = 1100 | T1o | 1100 | 1100 | PASS |

***A\_storageunit:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INPUT | W\_en | Clk | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| 0100 | 1 | 0 | uuuu | uuuu | PASS |
|  | 1 | 1 | uuuu | uuuu | PASS |
|  | 0 | 0 | uuuu | uuuu | PASS |
|  | 0 | 1 | 0100 | 0100 | PASS |

***B\_storageunit:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INPUT | W\_en | Clk | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| 0100 | 1 | 0 | uuuu | uuuu | PASS |
|  | 1 | 1 | uuuu | uuuu | PASS |
|  | 0 | 0 | uuuu | uuuu | PASS |
|  | 0 | 1 | 0100 | 0100 | PASS |

***Y\_storageunit:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INPUT | W\_en | Clk | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| 0100 | 1 | 0 | uuuu | uuuu | PASS |
|  | 1 | 1 | uuuu | uuuu | PASS |
|  | 0 | 0 | uuuu | uuuu | PASS |
|  | 0 | 1 | 0100 | 0100 | PASS |

***Op\_storageunit:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INPUT | W\_en | Clk | EXPECTED OUTPUT | STIMULATED OUTPUT | TEST PASS OR FAIL |
| ANDo | 1 | 0 | ANDo | ANDo | PASS |
|  | 1 | 1 | ANDo | ANDo | PASS |
|  | 0 | 0 | ANDo | ANDo | PASS |
|  | 0 | 1 | ANDo | ANDo | PASS |

***Individual Responsibilities:***

***Tejaswini Pulishetti:***

* *Custom package*
* *Mux 4x1*
* *Y storage unit*
* *A storage unit*
* *Top module*

***Vinay Reddy Padigela:***

* *Mux 2x1*
* *B storarge unit*
* *Op storage unit*
* *CU*
* *Top module testbench(80 % vinay + 20% tejaswini)*