EENG 5560 FINAL PROJECT REPORT

Name: Anushree Chiganipalya Nagaraj (11708317)
Tharun Padmanabha Somashekar (11708316)

Due: 05/02/2024

Table of Contents

Design

Block diagrams

Overall design

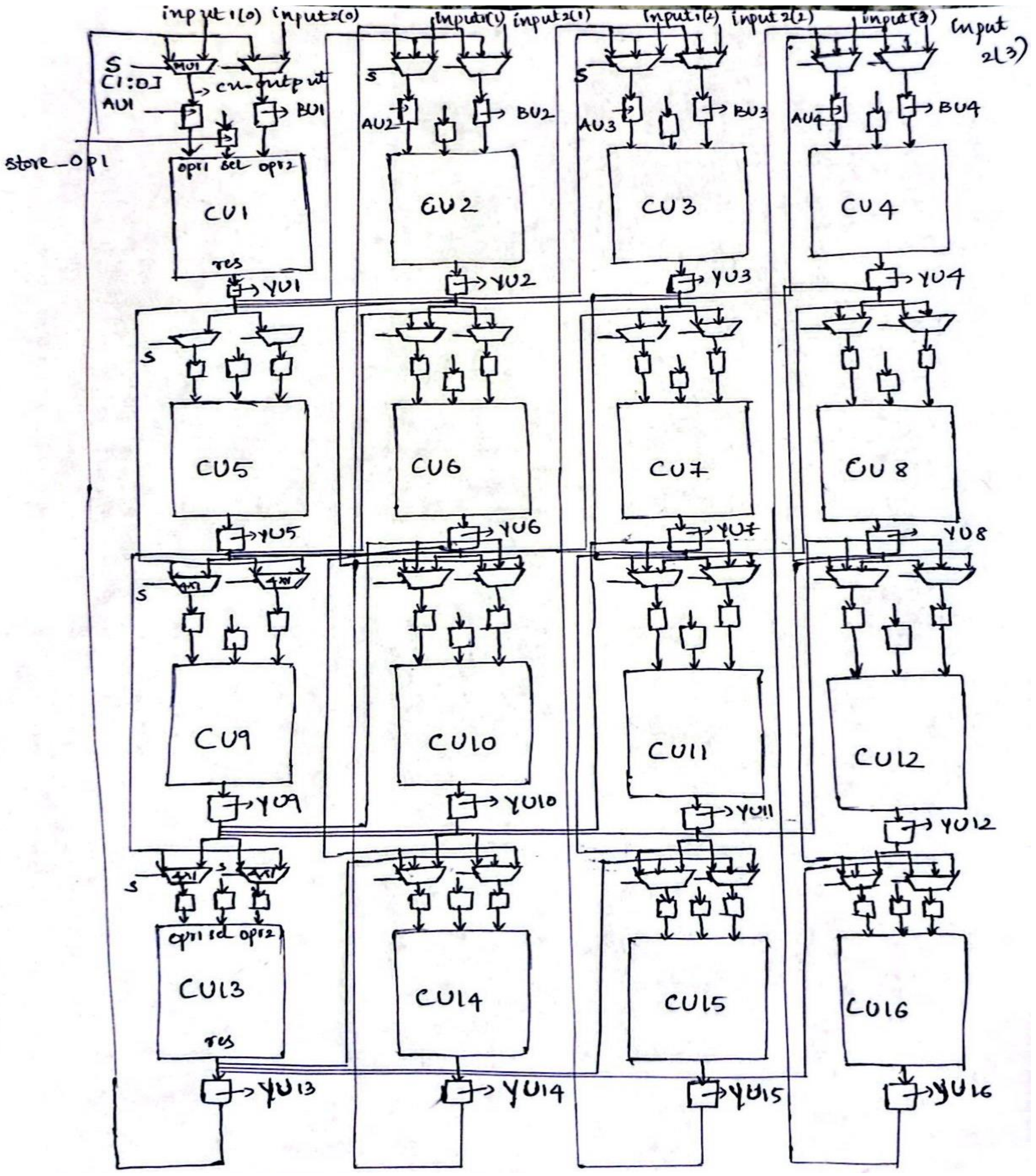


Figure 1- Overall component with subcomponents and intermediate signals

Note: In figure 1 for the CU 1 all the ports name are mentioned which is same for the rest of the CUs except for the index.

Overall component:

Parameters: d_w – data width (for inputs and outputs)

Input ports:

Port name	Bit width	Purpose
Input1 (0), Input2 (0), Input1 (1), Input2 (1), Input1 (2), Input2 (2), Input1 (3), Input2 (3)	d_w = 4	Data inputs 1 dimension array data inputs
Sel	5	2-dimension array for selecting operations for CU
Clk, w_en	1	These are used to determine when to read and when to write
Mux_sel	2	2-dimension array for selecting the mux output

Output ports:

Port name	Bit width	Purpose
Final_Output	d_w = 4	1 dimensional array for data output of final cu components
cu_Output	d_w = 4	2-dimensional array for data output of each cu components

Necessary intermediate signals:

Port name	Bit width	Purpose
A_output, B_output, Y_output	d_w = 4	These are 2d arrays we use them to store data and acts as Input to Instantiated CU's
res	d_w = 4	Data output Output from the CU acts as the input to the storage unit.

Subcomponents

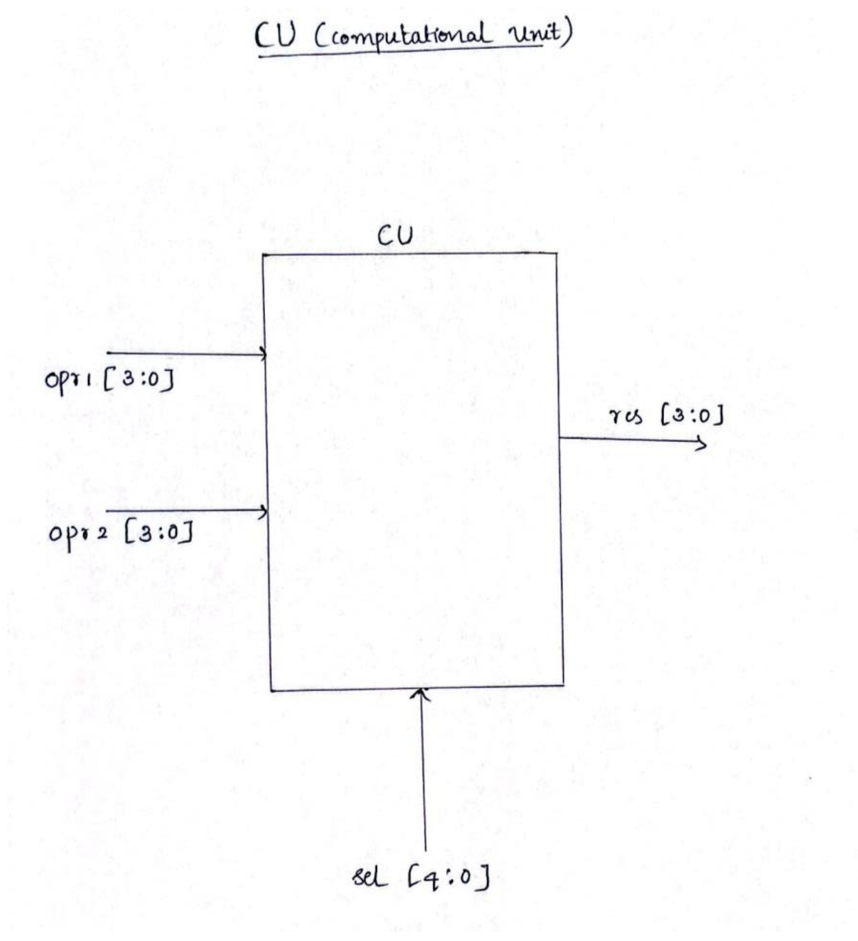


Figure 2 – Computational unit

Subcomponent: Computational unit

Input ports:

Port name	Bit width	Purpose
Opr1, opr2	d_w = 4	Data inputs
Sel	5	It selects the operation which will be performed on data inputs

Output ports:

Port name	Bit width	Purpose
res	d_w = 4	Data output

Necessary intermediate signals:

Port name	Bit width	Purpose
R_output	d_w = 4	It is 2d array each element of the array is act as input for storage units

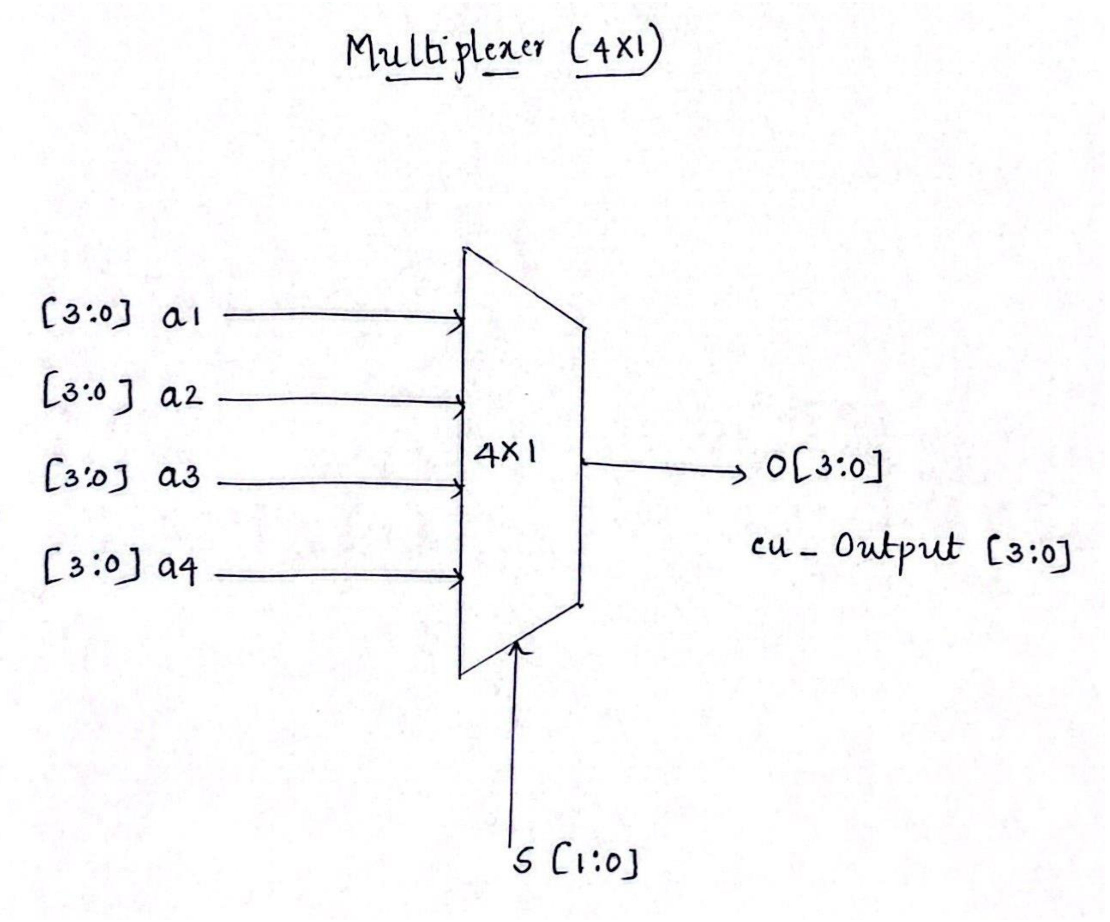


Figure 3 – 4x1 multiplexer

Subcomponent: 4x1 mux

Input ports:

Port name	Bit width	Purpose
a1, a2, a3, a4	d_w = 4	Data inputs
S	2	Select line, selects which of the 4 data inputs to send to data output

Output ports:

Port name	Bit width	Purpose
cu_output	d_w = 4	Data output

Necessary intermediate signals:

Port name	Bit width	Purpose
mux_output	d_w = 4	It is 2d array in which element will act as input for the storage unit

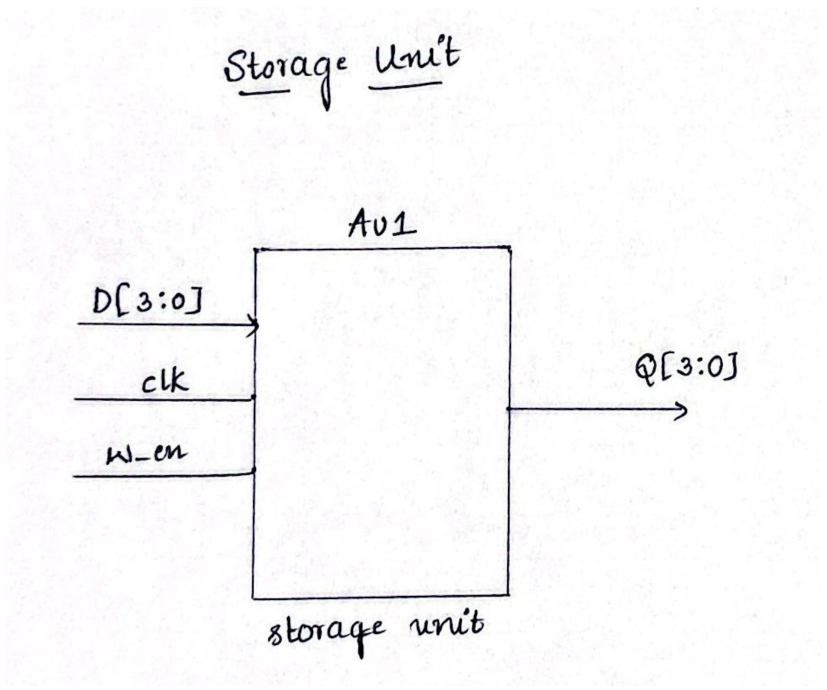


Figure 4 – Storage Unit for AU, BU, YU

Subcomponent: Storage Unit

Input ports:

Port name	Bit width	Purpose
D	d_w = 4	Data inputs
w_en	1	These are used to determine when to read and when to write
clk	1	Used for data synchronization

Output ports:

Port name	Bit width	Purpose
Q	d_w = 4	Data output

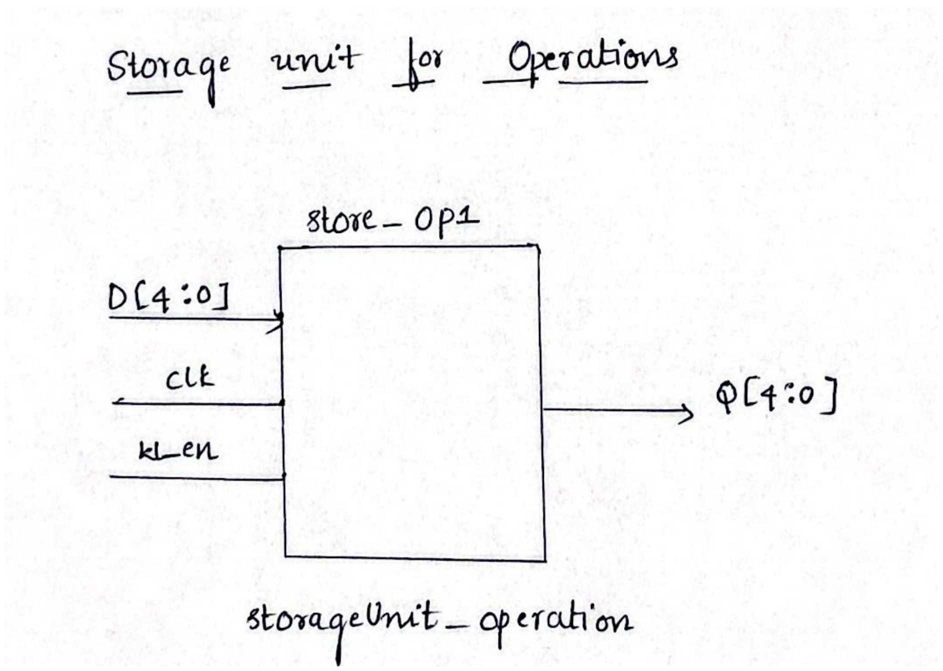


Figure 5 – Storage Unit to store Operation

Subcomponent: Storage Unit Operation

Input ports:

Port name	Bit width	Purpose
D	d_w = 5	Data inputs
w_en	1	These are used to determine when to read and when to write
clk	1	Used for data synchronization

Output ports:

Port name	Bit width	Purpose
Q	d_w = 5	Data output

Necessary intermediate signals:

Port name	Bit width	Purpose
Sel	d_w = 5	Selects the operation which needs to be performed by CU. Select line for CU
operation	5	2d array we use it to store operations

Design explanation

Functionality

The top-level design is a CU matrix completed with computation unit, multiplexer, and storage unit networks to handle data routing. Where the Configuration includes:

16 Computational Units (CUs): Key processing elements organized in a 4x4 matrix. They execute input from data and process it for specific purposes. 32 Multiplexers instances of 4x1. 64 Storage units are instantiated among them 48 storage units are of 4 bits used to store the output of multiplexer and used as input for CU units. 16 storage units are of 5 bits instantiated for storing operations which user provides and acts as input for CUs. 4x1 selects the direction of input signals towards CUs and storage units are used to store the data for CUs especially those belonging to a single row.

Detailed Configuration

Computational Units of the Network (4x4 Matrix): 4 CUs are in each row of the matrix. These CUs are then ordered in the raster-scan order and consecutively numbered from CU (0,0) to CU (3,3), where the first index corresponds to the row and the second to the column.

4x1 Multiplexers: There are 32 occurrences of 4x1 MUX. Each of the CUs has a corresponding 4x1 MUX which selects from its various inputs, including external data (first row only) and the first row of CUs will get feedback from the final output which acts as the input to the first row of. Green and pink lines are used for vertical connectivity, green is used to connect from bottom layer (3,0) to above layer (0,0) and pink lines represent the connectivity from above layer (1,0) to below layers (3,0), red lines are used to connect (0,0) to (2,0).

Blue lines are used to connect to horizontal for example (0,0) to (0,1), (0,2), (0,3) All this selection is under the control of specified MUX_SEL lines.

Storage unit: Positioned to deal with each one of the rows of the mux outputs and operations and output for CUs. Each storage units consolidates the outputs from the mux in architecture, facilitating either internal data routing within the matrix or preparing for final output delivery.

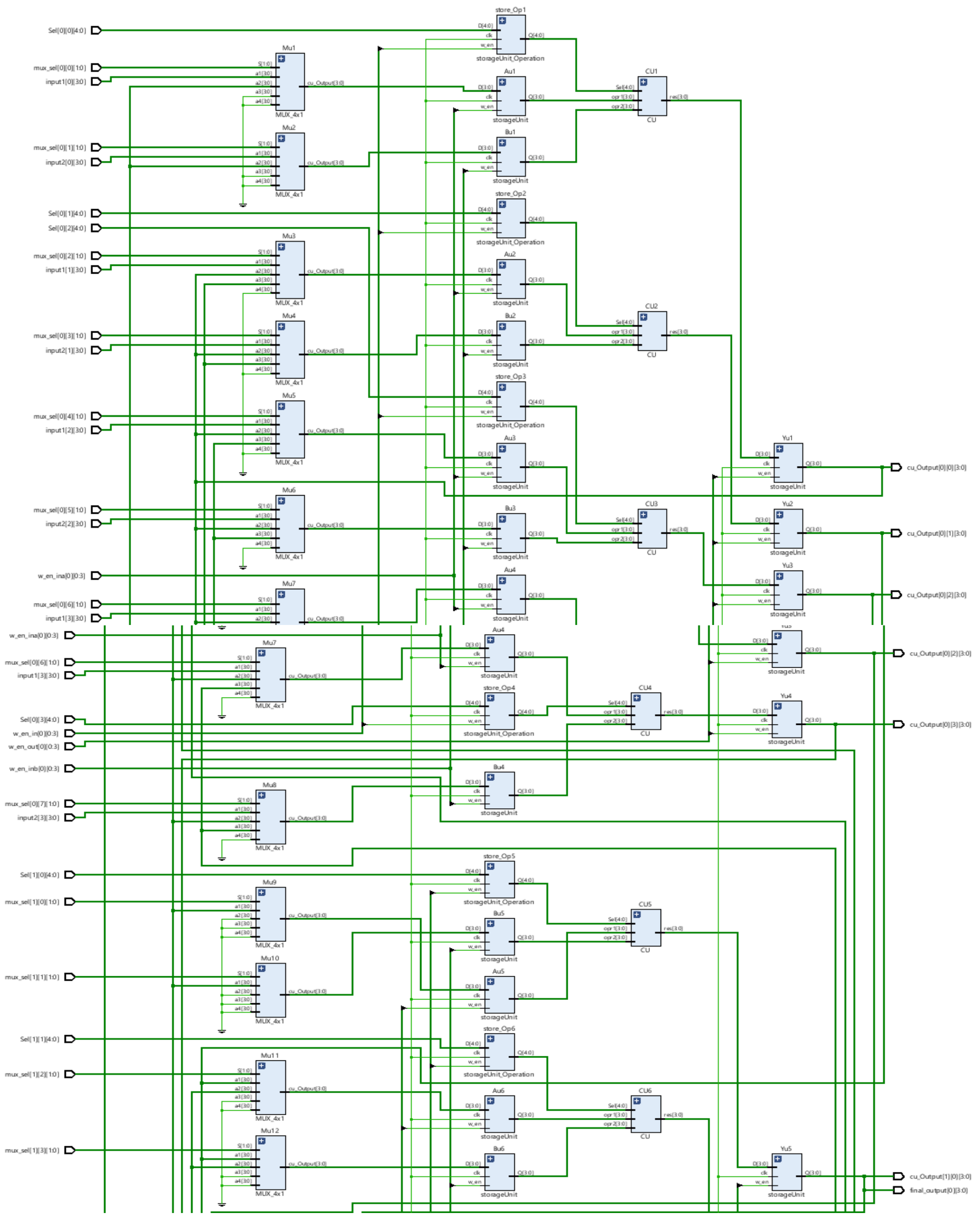
Data flow and processing functions Input Handling: The 4x1 multiplexers take care of the inputs by selecting the required data according to the operational requirement of the system at a particular point in time. The multiplexers can be dynamically reconfigured to be either inputs selected from outside sources or from the outputs of other CUs based on processing requirements.

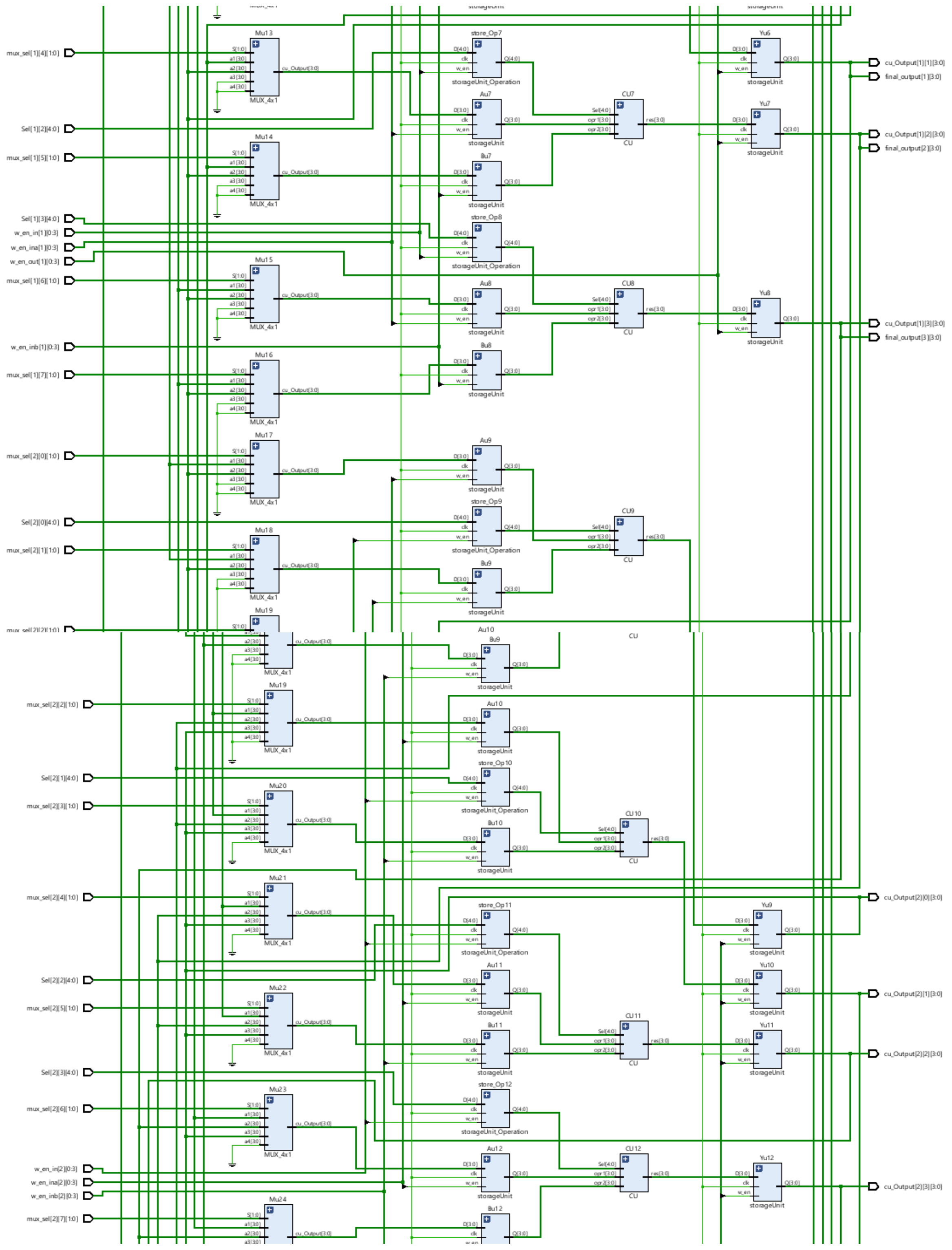
Every computational unit (CU) (Only for the first row) is waiting for the possible feedback from Last row CUs in the matrix and forms the processing chain of data. This interconnection of each CU the output of one serving as input to another provides that a CU may be able to adjust or vary its operations during implementation depending upon the results equipped by its precursor or by parallel units. This will provide an interdependency that will allow complex multi-stage processing strategies. From the above explanation the output of CU will greatly depend on the computed outputs from other CUs in the system improving the functionality and efficiency of the matrix.

Let us consider the first row of the CUs where the CUs will get input from the external and from the final output from the last row CUs as the feedback (Only for the 1st row). Where the 4x1 mux selects the input from the various inputs. Output from mux is given as input to the storage unit (AU & BU) which stores the data in read mode and during the write mode the stored data given as input to the CU and the storage unit for operation used to store operations which selects operation for the CU (Acts as select line for CU). The output from the CU will be stored in the storage unit (YU) and given as input to the different CUs.

<u>Results</u>

Generated Schematics





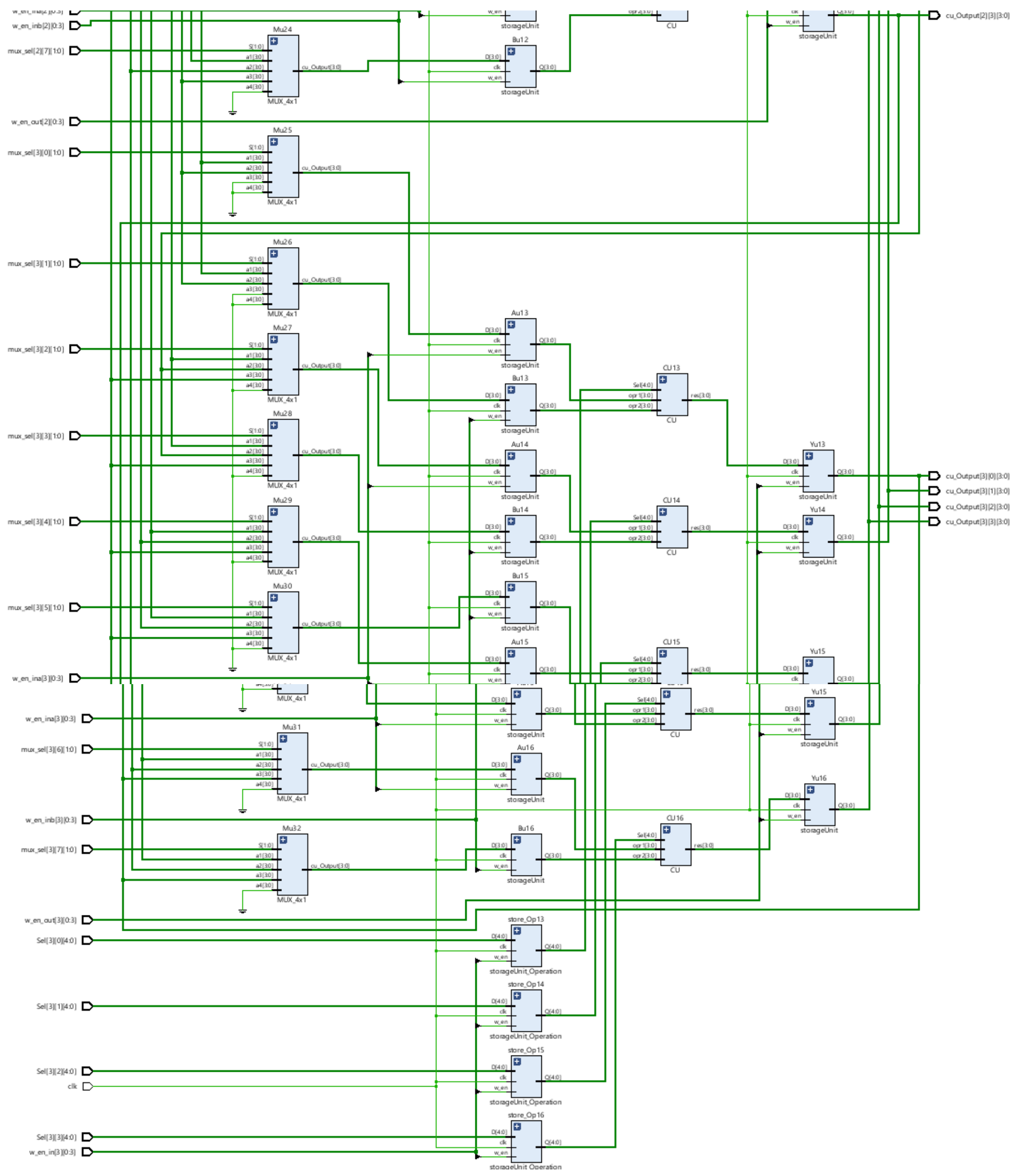


Figure 6 – RTL schematic of overall component

The above RTL schematic consists of 16 Computational Units, 32 Multiplexers, 64 storage units among them 48 are 4 bits sized and 16 are 5 bits sized. RTL and block diagram connections are matched and verified.

Where as the 1 CU consists of two of 4x1 muxes and which selects the input among various inputs (For 1st row will get input from external and feedback from the final output). It has 2 storage unit (AU and BU) to store data from muxes and 1 storage unit (Store_op) selects the operation to be performed by the CU. And the output of the CU will get stored in the storage unit (YU).

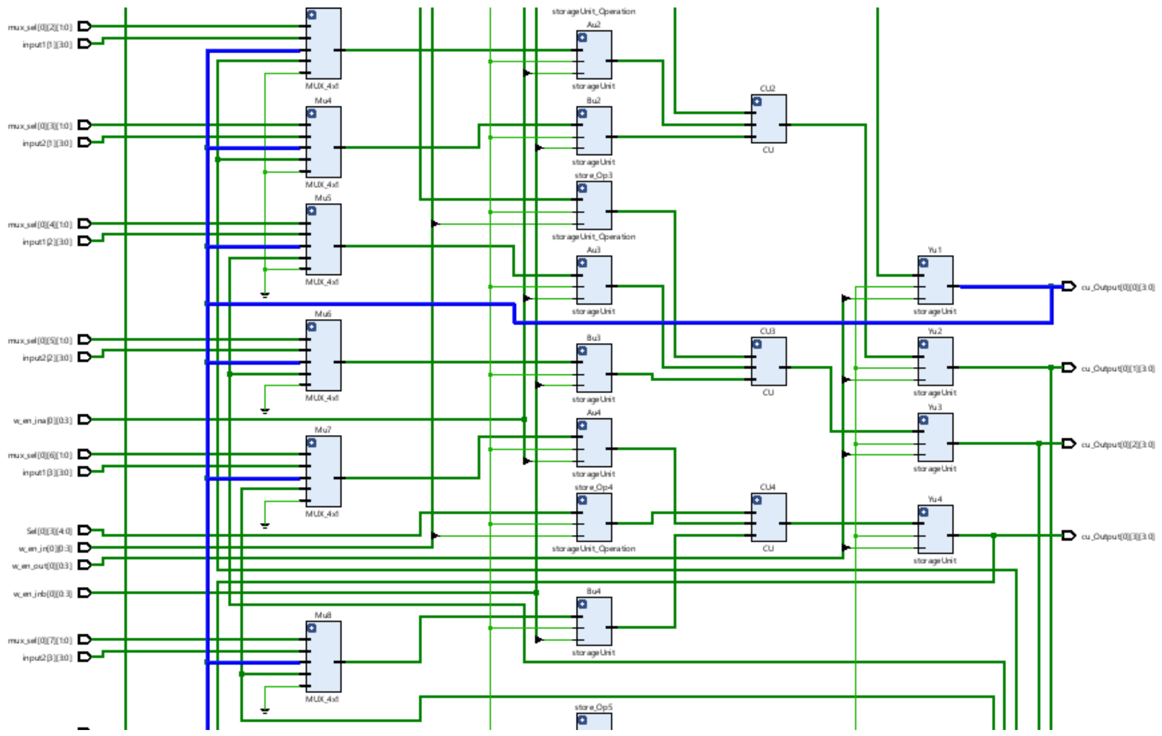


Figure 7 – Horizontal connectivity

The above shows the horizontal connectivity where the output from the CU1 is stored in the YU1 storage unit which is given as input the muxes (muxes 3 ,4, 5, 6, 7,8).

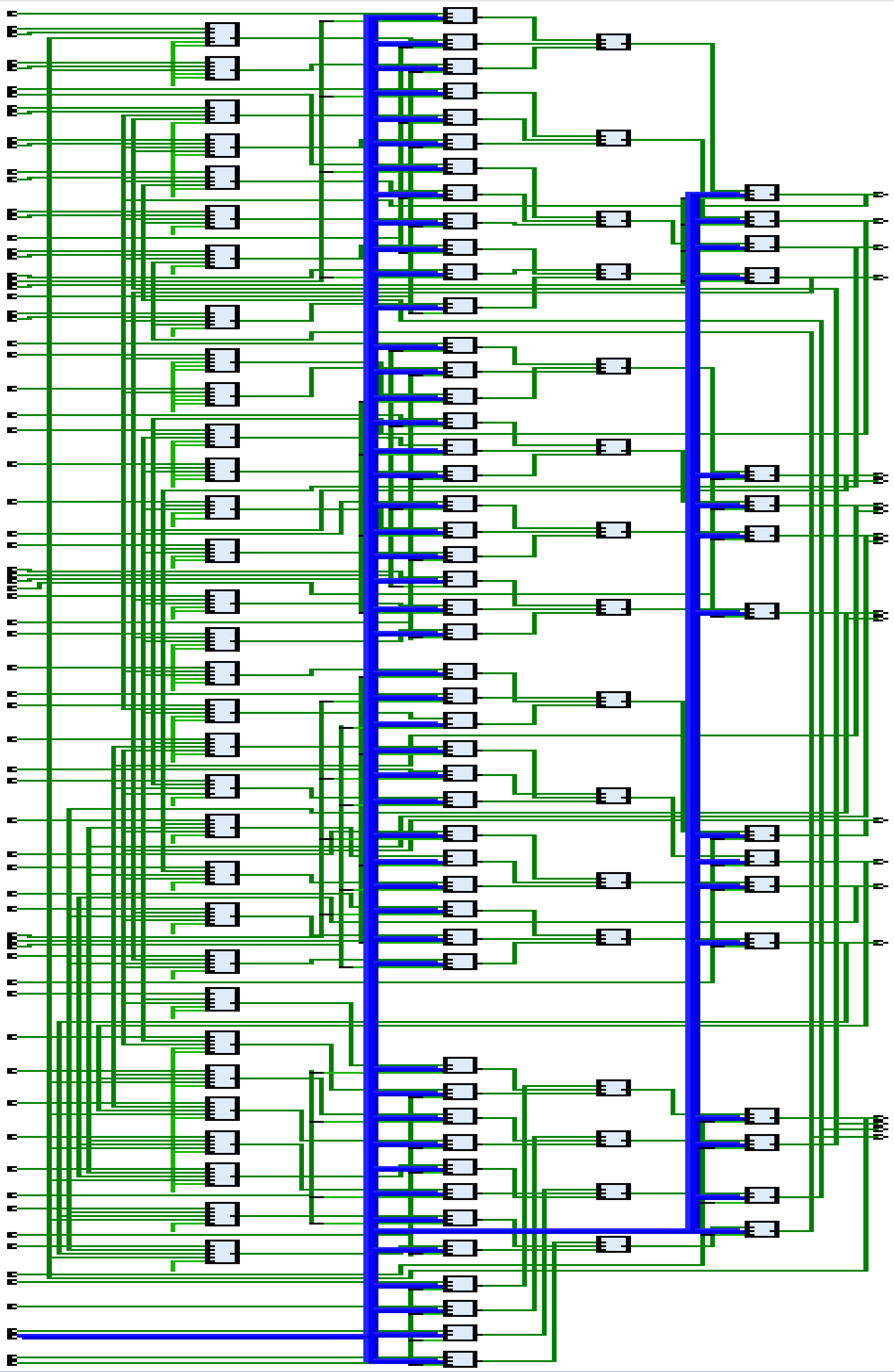


Figure 8 – Multi level connectivity

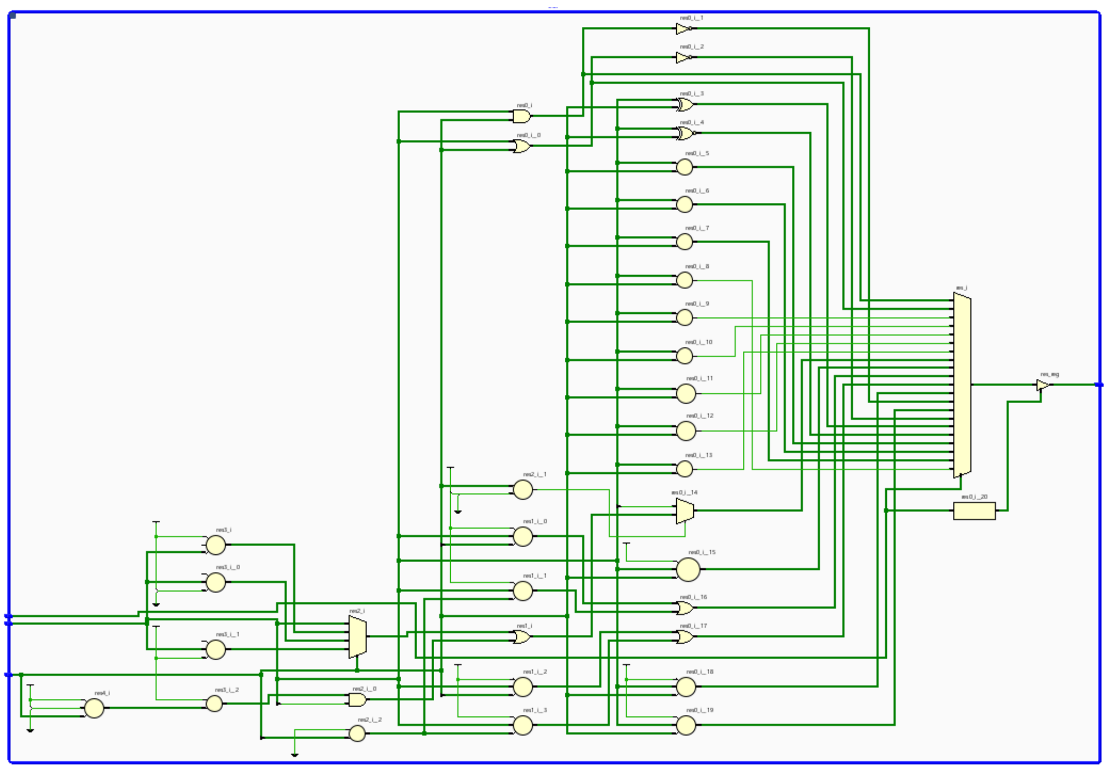


Figure 9 – RTL schematic of CU

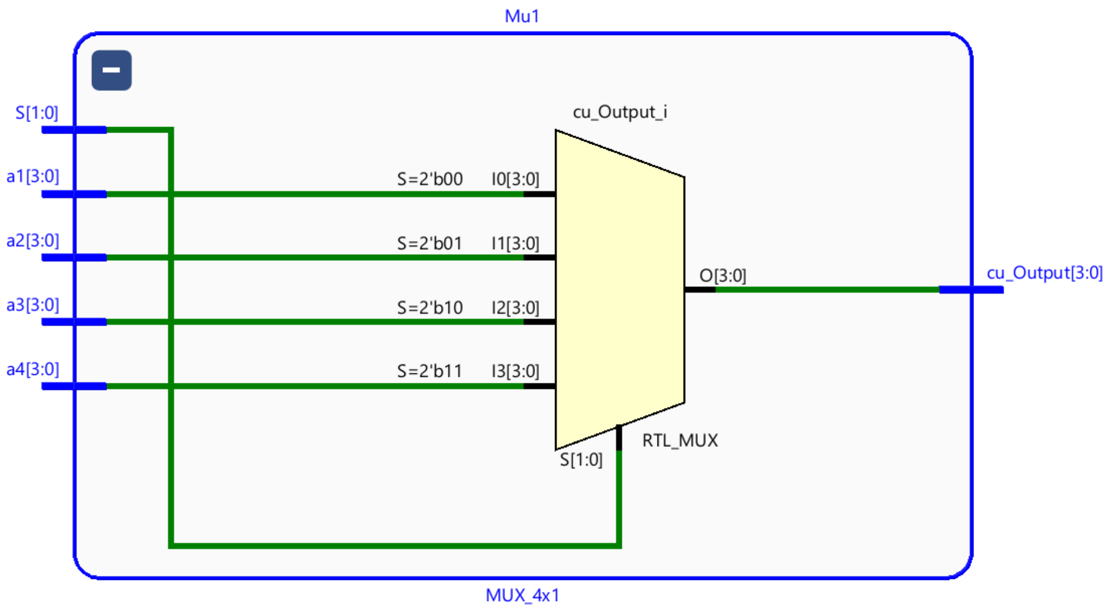


Figure 10 – RTL schematic of 4x1 mux

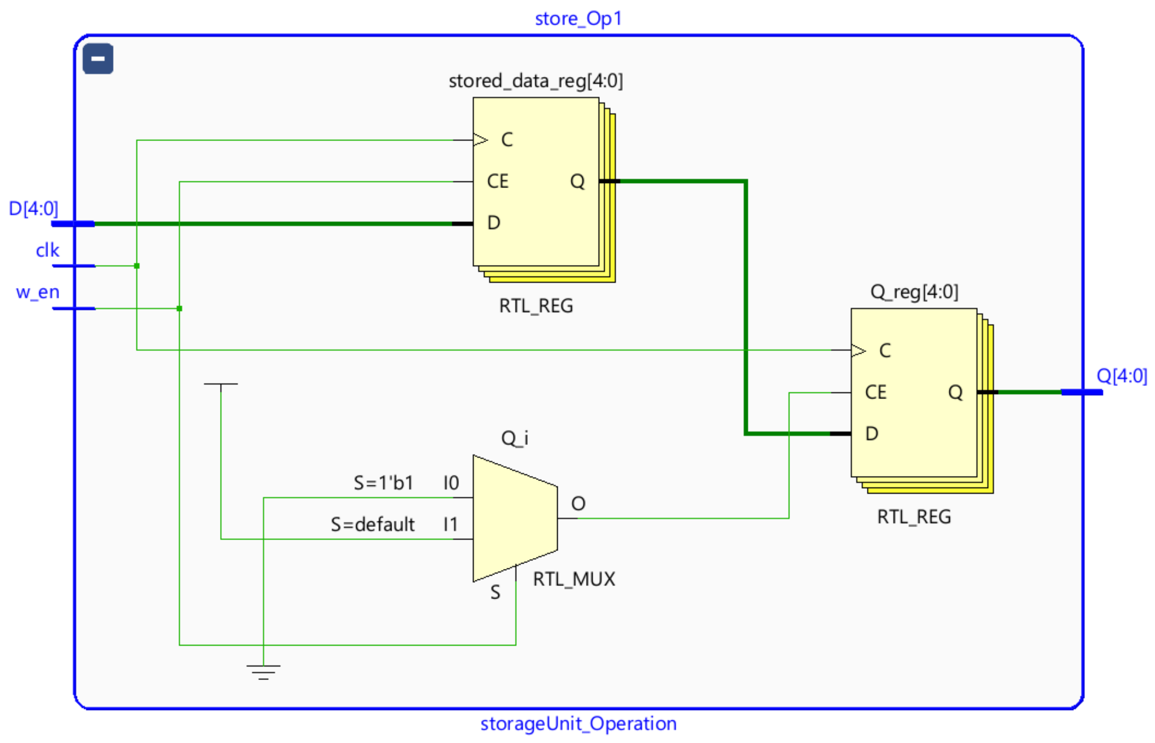


Figure 11 – RTL schematic of Storage unit to store operations

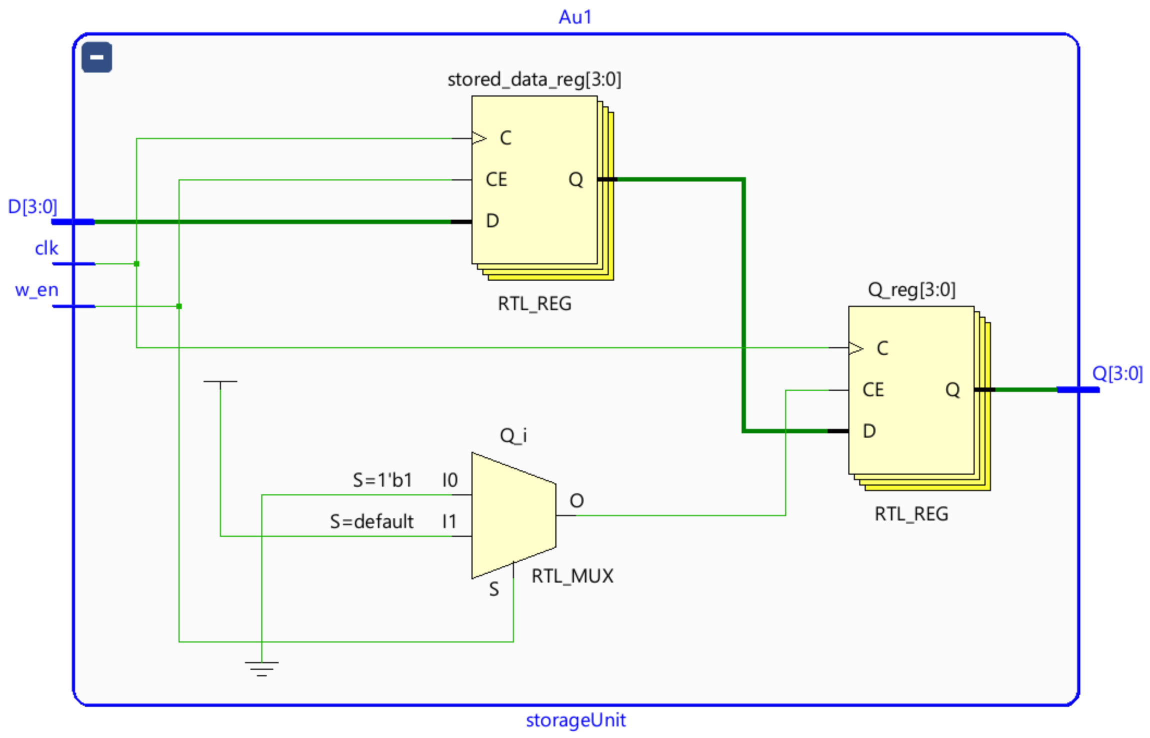
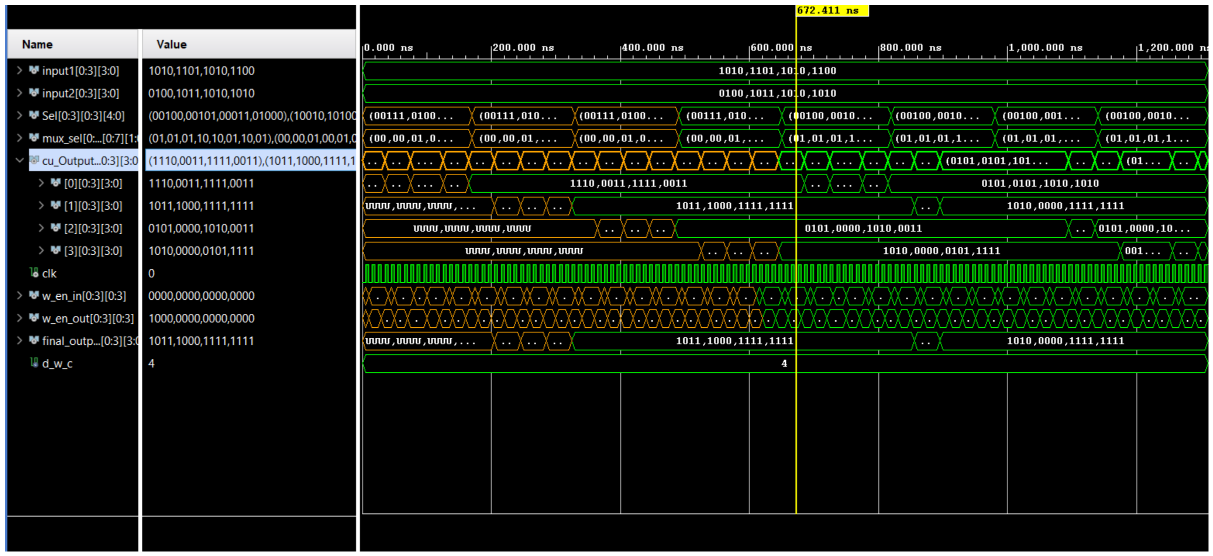


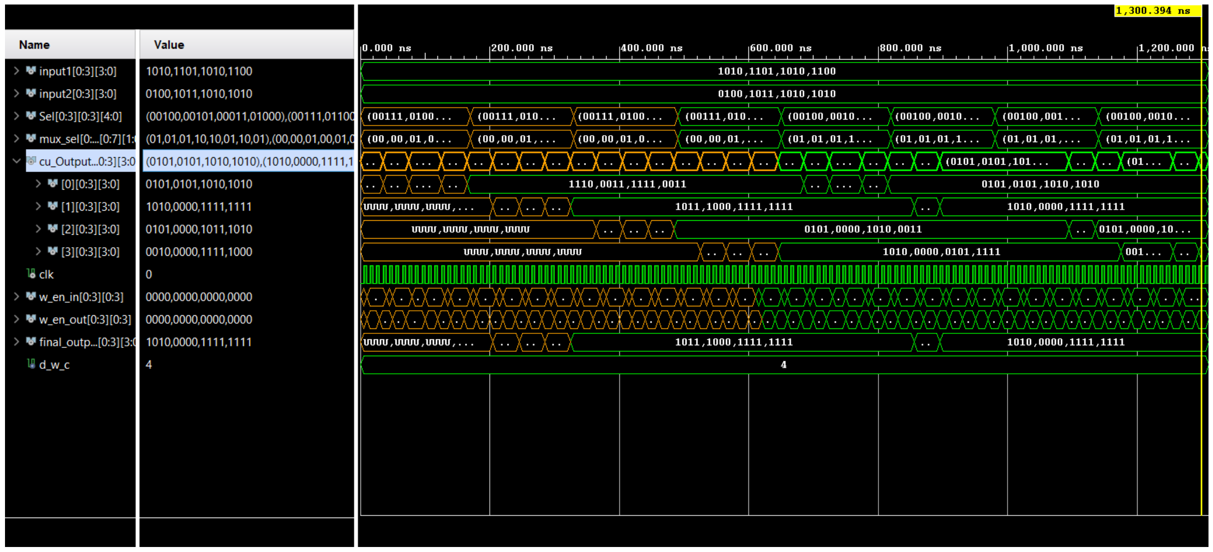
Figure 12 – RTL schematic of Storage unit to store data from multiplexer and CU

Waveforms



Name	Value
input1[0:3][3:0]	1010,1101,1010,1100
input2[0:3][3:0]	0100,1011,1010,1010
Sel[0:3][0:3][4:0]	(00100,00101,00011,01000),(10010,10100
mux_sel[0:...[0:7][1:	(01,01,01,10,10,01,10,01),(00,00,01,00,01,0
cu_Output...0:3][3:0	(1110,0011,1111,0011),(1011,1000,1111,1
[0][0:3][3:0]	1110,0011,1111,0011
[1][0:3][3:0]	1011,1000,1111,1111
[2][0:3][3:0]	0101,0000,1010,0011
[3][0:3][3:0]	1010,0000,0101,1111
clk	0
w_en_in[0:3][0:3]	0000,0000,0000,0000
w_en_out[0:3][0:3]	1000,0000,0000,0000
final_outp...[0:3][3:0	1011,1000,1111,1111
d_w_c	4

Figure 13 - Run 1 Waveforms and values



Name	Value
input1[0:3][3:0]	1010,1101,1010,1100
input2[0:3][3:0]	0100,1011,1010,1010
Sel[0:3][0:3][4:0]	(00100,00101,00011,01000),(00111,01100
mux_sel[0:...[0:7][1:	(01,01,01,10,10,01,10,01),(00,00,01,00,01,0
cu_Output...0:3][3:0	(0101,0101,1010,1010),(1010,0000,1111,1
[0][0:3][3:0]	0101,0101,1010,1010
[1][0:3][3:0]	1010,0000,1111,1111
[2][0:3][3:0]	0101,0000,1011,1010
[3][0:3][3:0]	0010,0000,1111,1000
clk	0
w_en_in[0:3][0:3]	0000,0000,0000,0000
w_en_out[0:3][0:3]	0000,0000,0000,0000
final_outp...[0:3][3:0	1010,0000,1111,1111
d_w_c	4

Figure13 – Run 2 Waveforms and values

Note: On the first day of the demo, we have shown the Run 1 output Since we were getting partial output for Run2. On the other day we have shown the overall output for both Runs.

Table/Calculations

Overall Design

Run1 (0ns to 645ns):

CU#	SourceA	SourceB	A	B	Oper	Calculated Op	Simulated Op	Match
CU1	External	External	1010	0100	ADD	1110	1110	Yes
CU2	CU1	External	1110	1011	SUB	0011	0011	Yes
CU3	CU1	CU1	1110	1110	XNOR	1111	1111	Yes
CU4	External	CU1	1100	1110	ROR	0011	0011	Yes
CU5	CU1	CU1	1110	1110	ROL	1011	1011	Yes
CU6	CU5	CU2	1011	0011	LSL	1000	1000	Yes
CU7	CU5	CU3	1011	1111	LTE	1111	1111	Yes
CU8	CU5	CU5	1011	1011	EQ	1111	1111	Yes
CU9	CU1	CU5	1110	1011	XOR	0101	0101	Yes
CU10	CU9	CU2	0101	0011	LSR	0000	0000	Yes
CU11	CU9	CU7	0101	1111	NAND	1010	1010	Yes
CU12	CU8	CU4	1111	0011	AND	0011	0011	Yes
CU13	CU9	CU5	0101	1011	ROR	1010	1010	Yes
CU14	CU10	CU13	0000	1010	GTE	0000	0000	Yes
CU15	CU7	CU11	1111	1010	SUB	0101	0101	Yes
CU16	CU13	CU8	1010	1111	OR	1111	1111	Yes

Run2(645ns to 1295ns):

CU#	SourceA	SourceB	A	B	Oper	Calculated Op	Simulated Op	Match
CU1	CU13	CU13	1010	1010	NOR	0101	0101	Yes
CU2	CU1	CU14	0101	0000	XOR	0101	0101	Yes
CU3	CU15	CU1	0101	0101	NAND	1010	1010	Yes
CU4	CU16	CU1	1111	0101	SUB	1010	1010	Yes
CU5	CU1	CU1	0101	0101	ADD	1010	1010	Yes
CU6	CU5	CU2	1010	0101	EQ	0000	0000	Yes
CU7	CU5	CU3	1010	1010	XNOR	1111	1111	Yes
CU8	CU5	CU4	1010	1010	LTE	1111	1111	Yes
CU9	CU5	CU1	1010	0101	ROL	0101	0101	Yes
CU10	CU9	CU2	0101	0101	LT	0000	0000	Yes
CU11	CU3	CU7	1010	1111	SUB	1011	1011	Yes

CU12	CU8	CU4	1111	1010	AND	1010	1010	Yes
CU13	CU5	CU9	1010	0101	MULT	0010	0010	Yes
CU14	CU6	CU13	0000	0010	LSR	0000	0000	Yes
CU15	CU7	CU13	1111	0010	ASR	1111	1111	Yes
CU16	CU12	CU13	1010	0010	LSL	1000	1000	Yes

Responsibilities of each team member

We both worked together most of the time, Instantiation part we divided. Although we encountered a lot of errors in testbench we figured it out together. Run 1 manual calculation and testing done by Anushree and Run 2 manual calculation and testing done by Tharun and we verified it together. As for the report part we divided equally.

Timing and Cycles. for all clk = 5 + 5 = 10, RUN 1

Cu(0,0) (or) Cu1 [Row1]

output Time → 37.223 ns ÷ 10 = 3.7 ⇒ 4 Clock cycles

Cu(0,1) (or) Cu2 [Row 1]

output Time → 75ns ÷ 10 = 7.5 = 8 clock cycles

Cu(0,2) (or) Cu3 [Row1]

output Time → 125 ÷ 10 = 12.5 = 13 clock cycles

Cu (0,3) (or) Cu4 [Row1]

output Time → 165 ÷ 10 = 16.5 ⇒ 17 clock cycles.

— for Row 1 Complete – 165 ÷ 10 = 16.5 cycles —

Cu (1,0) (or) Cu 5 [Row 2]

output Time → 205 ÷ 10 = 20.5 = 21 clock cycles

Cu (1,1) (or) Cu6 (Row 2)

output Time → 245 ÷ 10 = 24.5 = 25 clock cycles.

Cu(1,2) (or) Cu7 (Row 2)

output Time → 285 ÷ 10 = 28.5 = 29 clock cycles

Cu (1,3) (or) Cu8 (Row 2)

output Time → 325 ÷ 10 = 32.5 = 33 clock cycles

— for Row2 Complete – 325 ÷ 10 = 33 clock cycles —

cu(2,0) (or) cu9 (Row3)

output Time → 365 ÷ 10 = 36.5 = 37 clock cycles

cu(2,1) (or) cu10 [Row 3]

output Time → 405 ÷ 10 = 40.5 = 40 clock cycles

cu(2,2) (or) cu11 (Row 3)

output Time → 445 ÷ 10 = 44.5 = 45 clock cycles

cu(2,3) (or) cu12 [Row 3]

output Time → 485 ÷ 10 = 48.5 = 49 clock cycles.

— Row3 Complete → 485 ÷ 10 = 49 clock cycles ——

cu (3,0) (or) cu 13 (Row 4)

output → 525 ÷ 10 = 52.5 = 53 clock cycles

cu (3,1) (or) cu 14 (Row 4)

output → 565 ÷ 10 = 56.5 = 57 clock cycles.

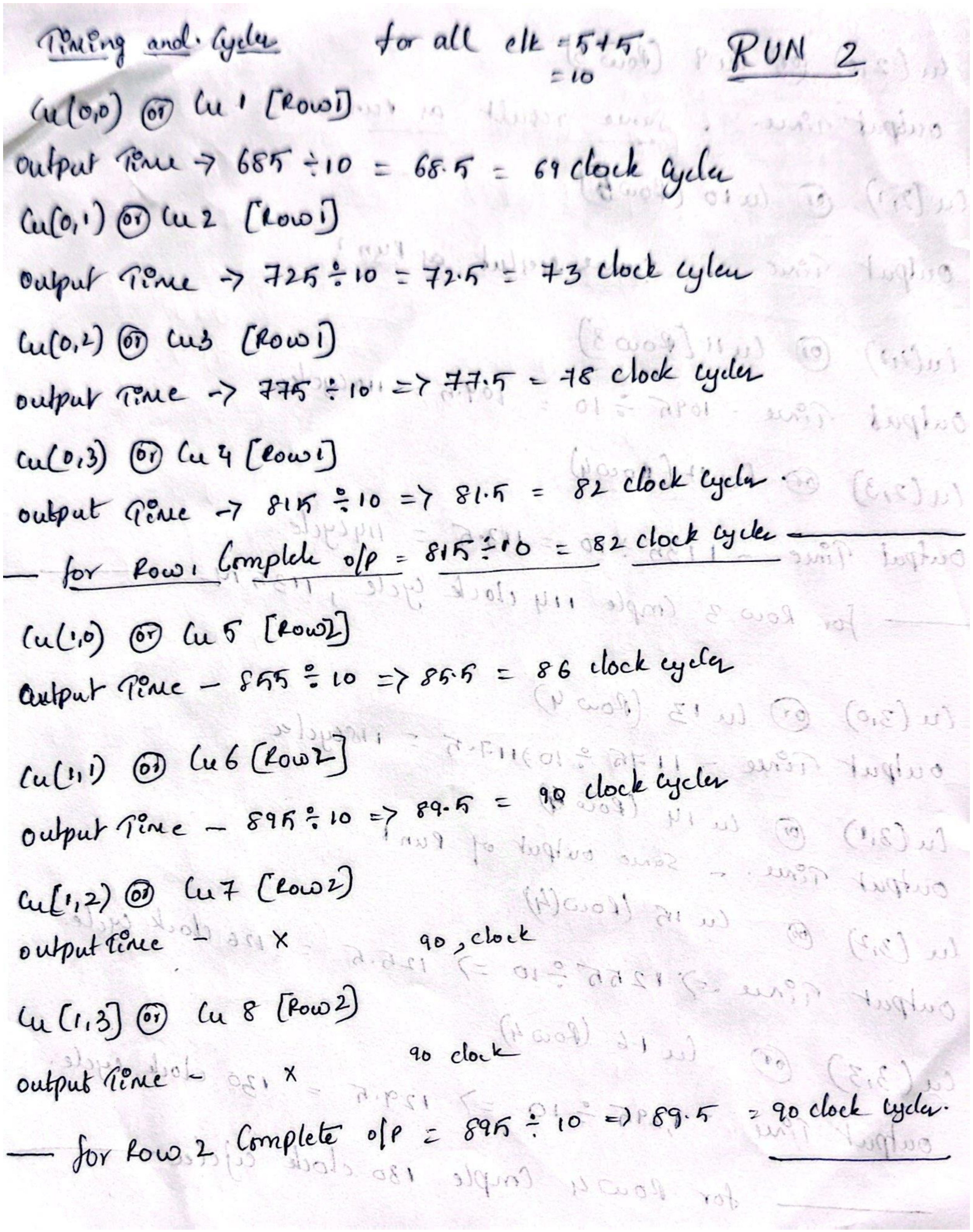
cu (3,2) (or) cu 15 (Row 4)

output → 605 ÷ 10 = 60.5 = 61 clock cycles

cu(3,3) (or) cu 16 (Row 4)

output → 645 ÷ 10 = 64.5 = 65 clock cycles.

—— Row 4 Complete = 645 ÷ 10 = 65 clock cycles ——



Timing and Cycles for all clk = 5+5 = 10 RUN 2

Cu(0,0) or Cu 1 [Row 1]

Output Time → 685 ÷ 10 = 68.5 = 69 clock cycles

Cu(0,1) or Cu 2 [Row 1]

Output Time → 725 ÷ 10 = 72.5 = 73 clock cycles

Cu(0,2) or Cu3 [Row 1]

output Time → 775 ÷ 10 => 77.5 = 78 clock cycles

Cu(0,3) or Cu 4 [Row 1]

output Time → 815 ÷ 10 => 81.5 = 82 clock cycles.

— for Row 1 Complete o/p = 815 ÷ 10 = 82 clock cycles

Cu(1,0) or Cu 5 [Row 2]

Output Time – 855 ÷ 10 => 85.5 = 86 clock cycles

Cu(1,1) or Cu6 [Row 2]

output Time – 895 ÷ 10 => 89.5 = 90 clock cycles

Cu[1,2) or Cu 7 (Row 2)

output Time – X 90, clock

Cu (1,3] or Cu 8 [Row 2]

output Time – X 90 clock

— for Row 2 Complete o/p = 895 ÷ 10 => 89.5 = 90 clock cycles.

Cu(2,0) (or) Cu 9 (Row 3)

output Time. - Same result as Run 1

Cu(2,1) (or) Cu 10 (Row 3)

output Time - Same result as Run 1

Cu(2,2) (or) Cu 11 (Row 3)

output Time - 1095 ÷ 10 = 109.5 = 110 cycles

Cu(2,3) (or) Cu 12 (Row 4)

output Time. - 1135 ÷ 10 = 113.5 = 114 cycle.

——— for Row 3 Cmple 114 clock cycle, 1135 ns ———

Cu(3,0) (or) Cu 13 (Row 4)

output Time - 1175 ÷ 10 => 117.5 = 118 cycle

Cu(3,1) (or) Cu 14 (Row 4)

output Time. - Same output of Run 1

Cu(3,2) (or) Cu 15 (Row (4)

output Time => 1255 ÷ 10 => 125.5 = 126 clock cycle.

Cu(3,3) (or) Cu 16 (Row 4)

output Time. - 1,295 ÷ 10 => 129.5 = 130 clock cycle.

——— for Row 4 Cmple 130 clock cycle ———