PHASE A

I) Clock Pulse Generator

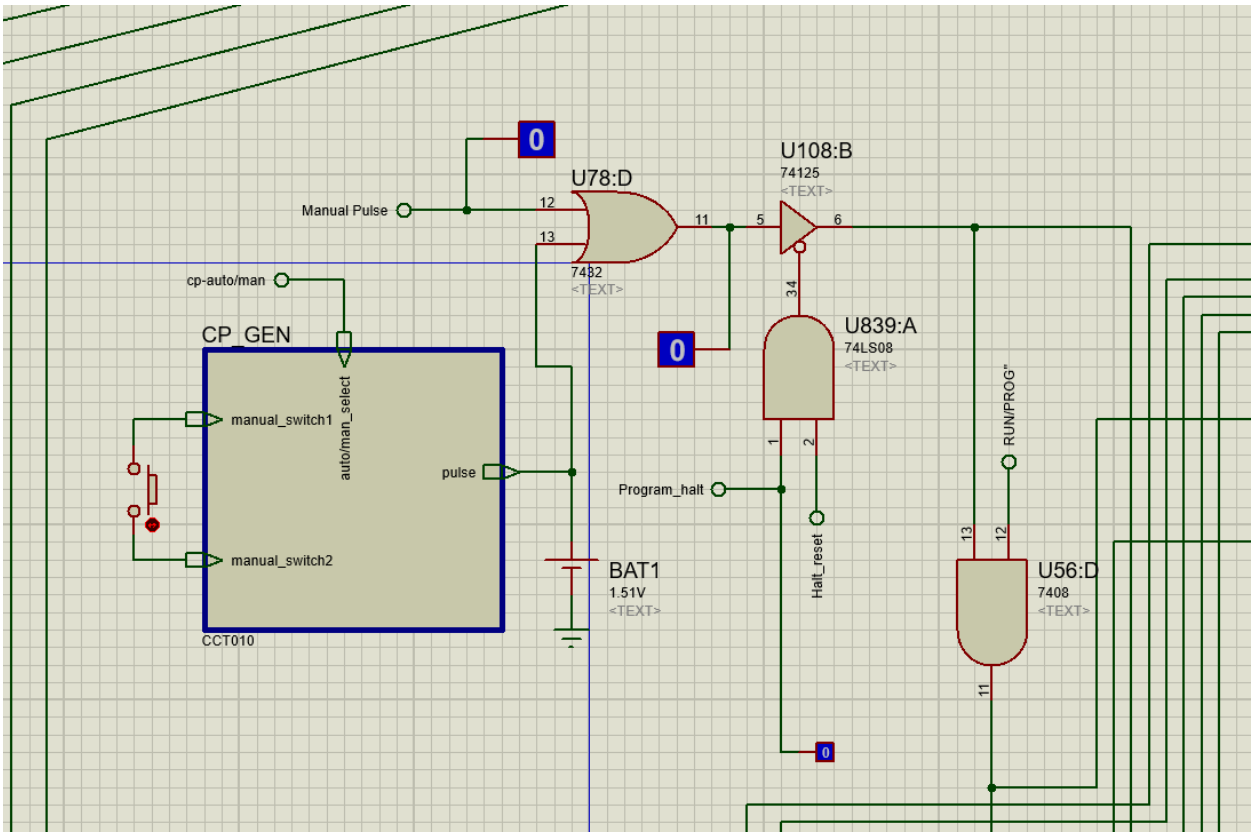


Figure: Clock Pulse Generator

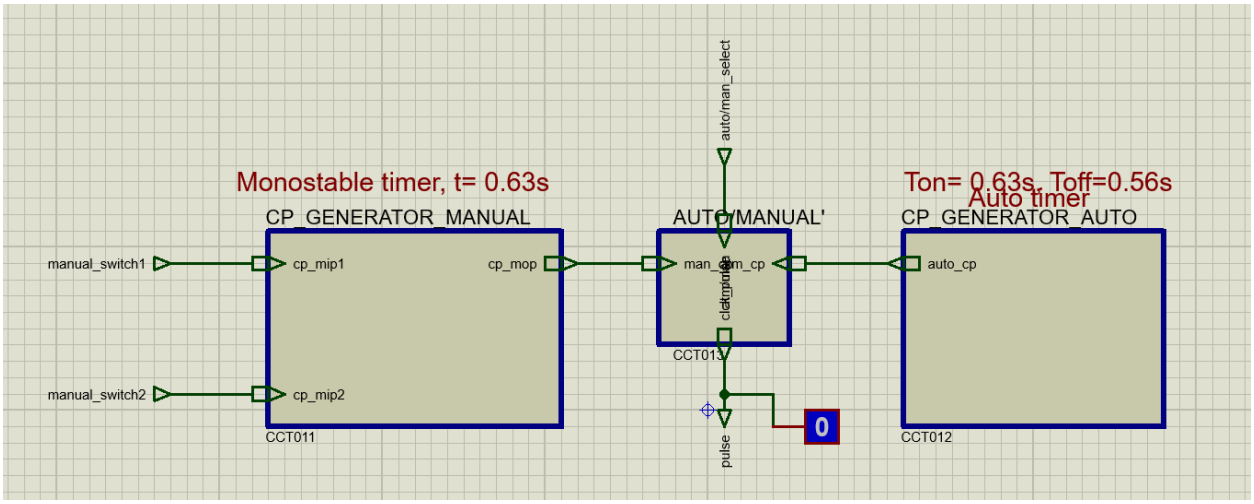


Figure: Clock Pulse Generator (Childsheet)

For clock pulse generator we used 555 timer IC. 555 timer was used to build astable multi vibrator and monostable multi vibrator and they were connected with a 2to 1 MUX to produce 1 clock pulse or train of clock pulse.

For monostable multivibrator:

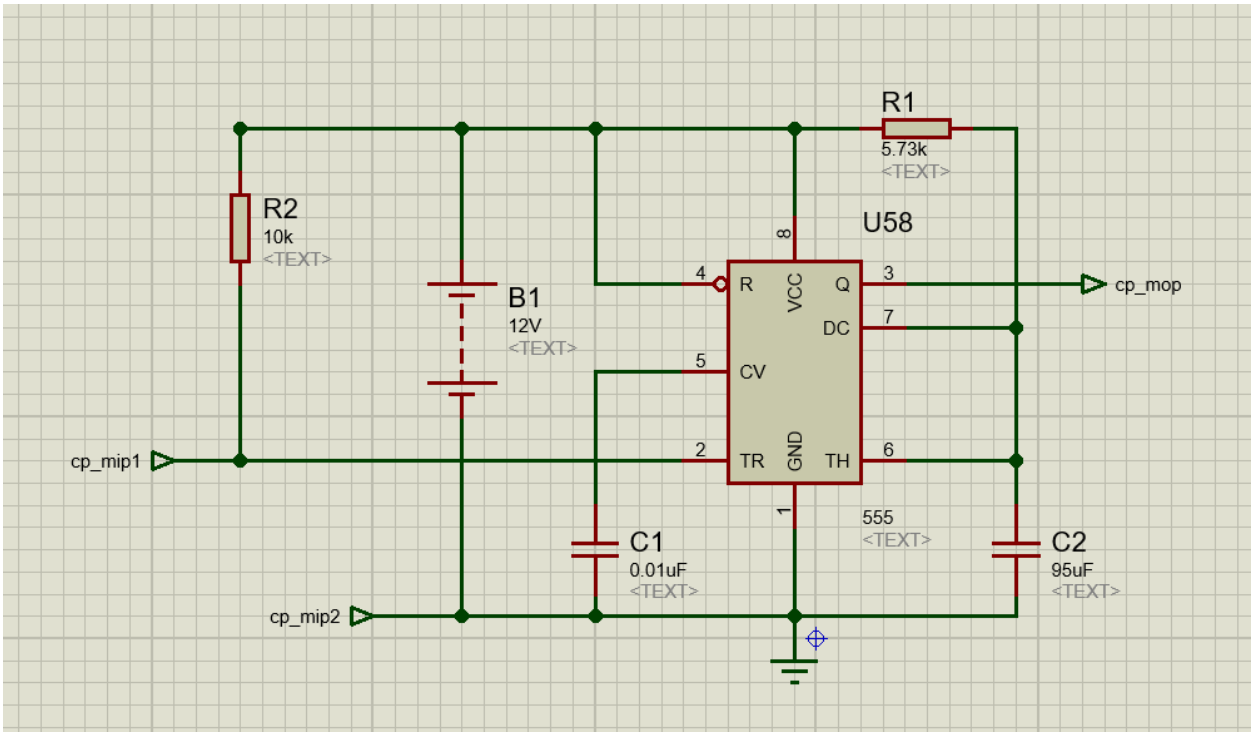


Figure: Monostable Multivibrator

The equation for time constant, T= 1.1R*C

For astable multivibrator:

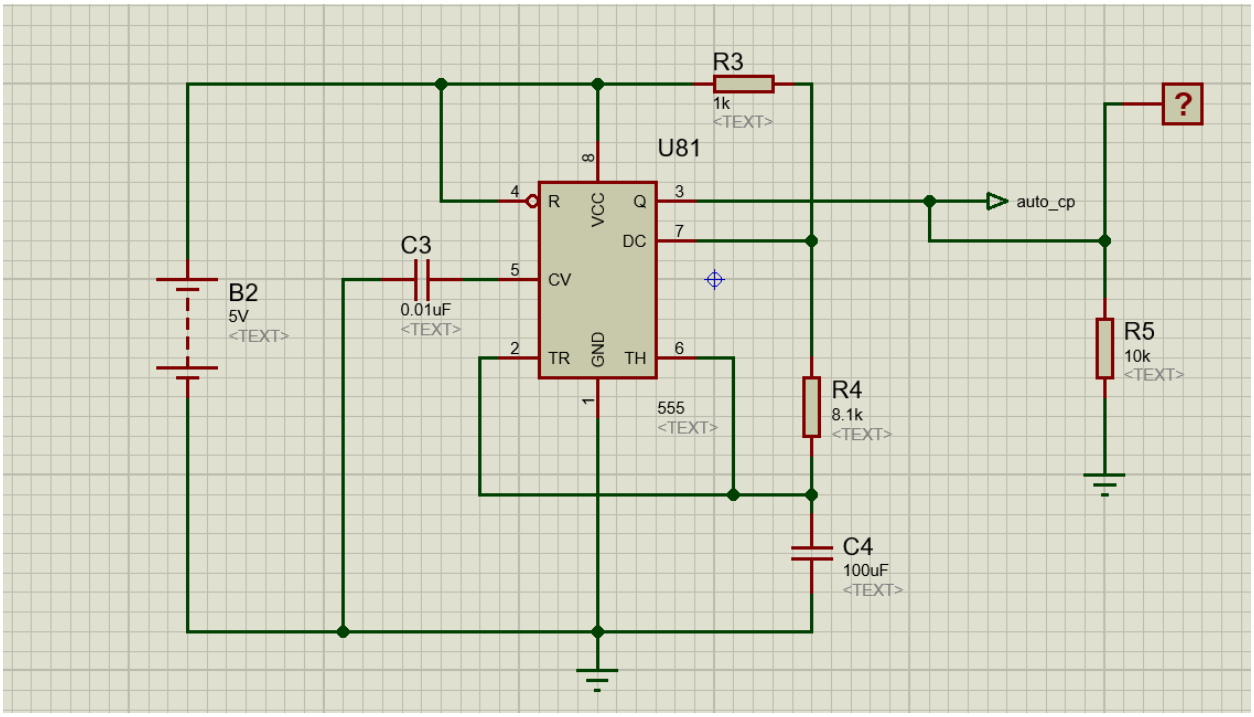


Figure: Astable Multivibrator

The eqution for charging time, $T_1 = 0.693(R_1 + R_2)*C$

And the equation for discharging time, $T_2 = 0.693*R_2*C$

Truth table:

Input	Output
1	Manual clock pulse
0	Automatic clock pulse

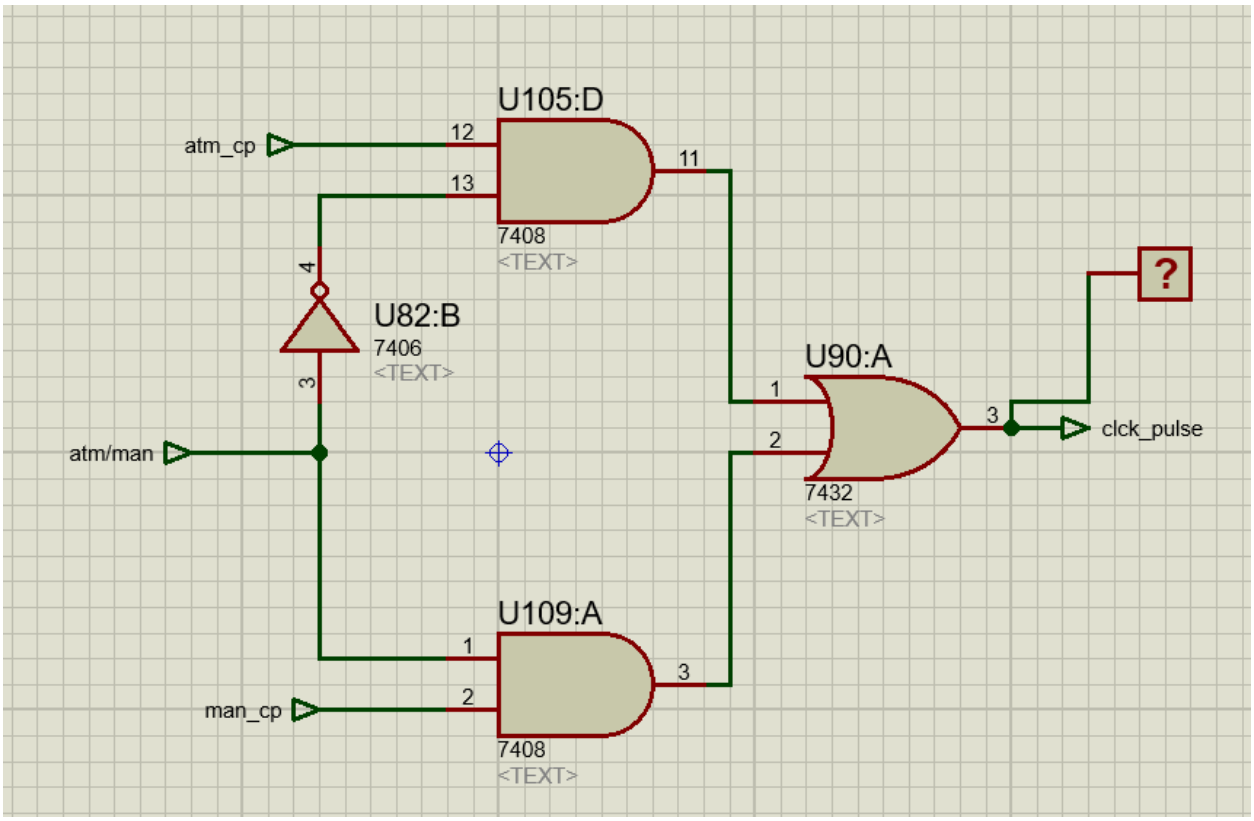


Figure: MUX for switching between Automatic or Manual clock pulse

II) Program Counter

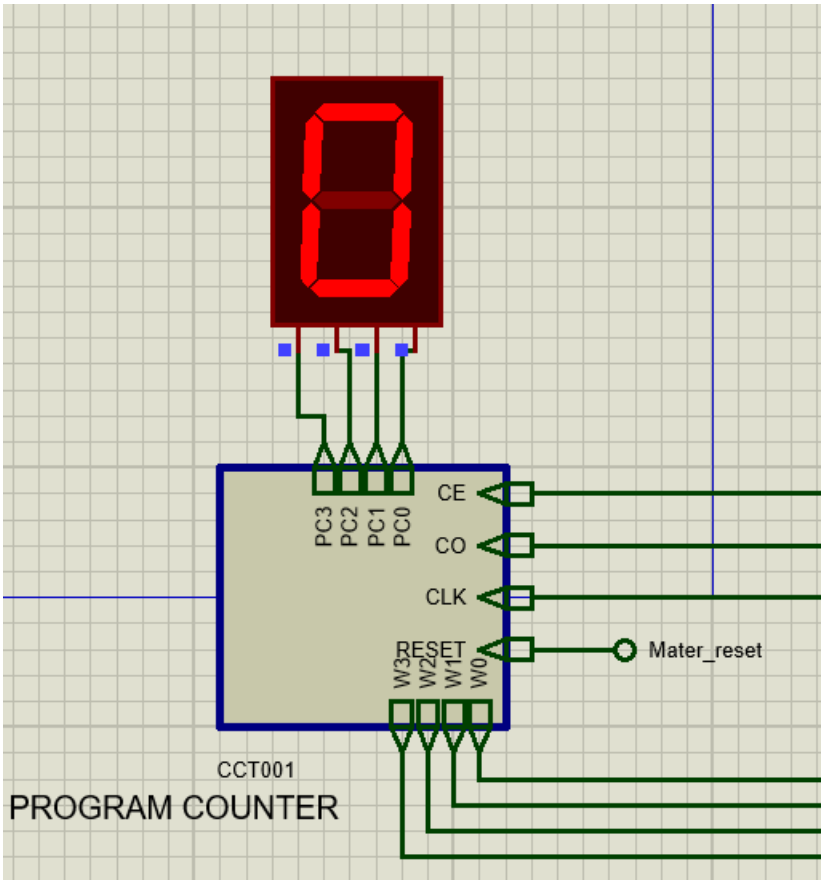


Figure: Program Counter

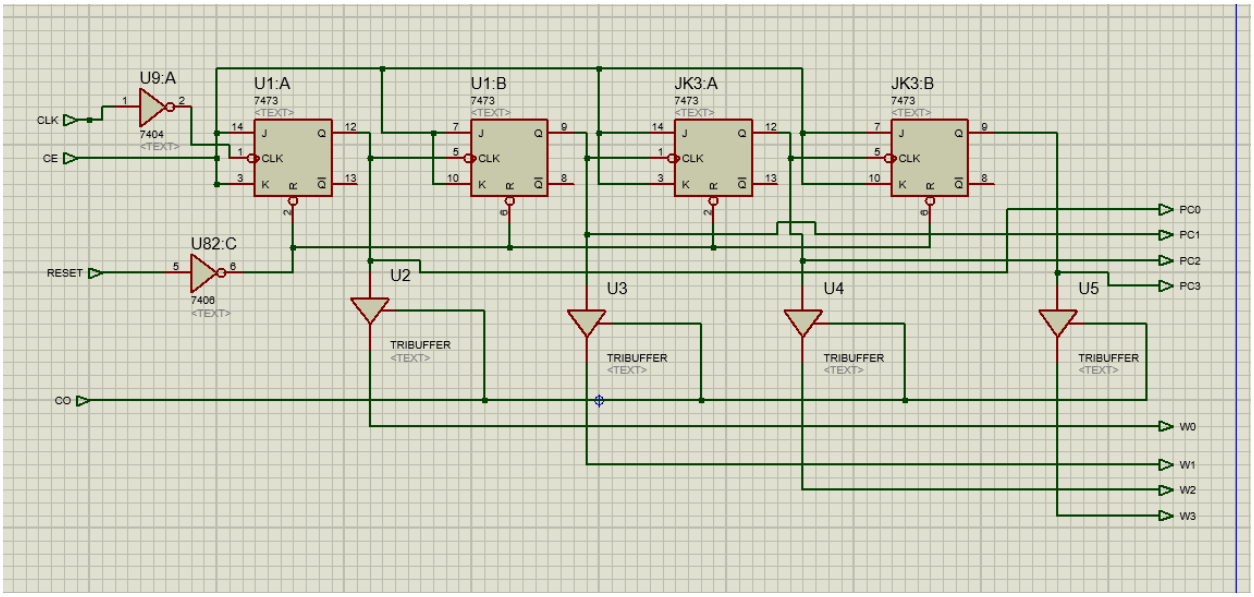


Figure: Program Counter(Childsheet)

The purpose of Program Counter is to select the next instruction automatically after the execution of each instruction.

Program Counter has been designed using four positive edge triggered JK Flip Flops.
CE pin increments the value of the program counter by 1 upon receiving a positive edge from the clock.
CO pin controls the flow of current program counter value from the program counter to the bus. This has been done by W0 to W3.
Reset pin resets the value of the program counter to 0000 when it receives logic 1.
The binary value of the program counter has been displayed in decimal form using a BCD to 7 Segment Decoder Display. This has been done by PC0 to PC3.

III) INPUT UNIT and MAR:

Input Unit:

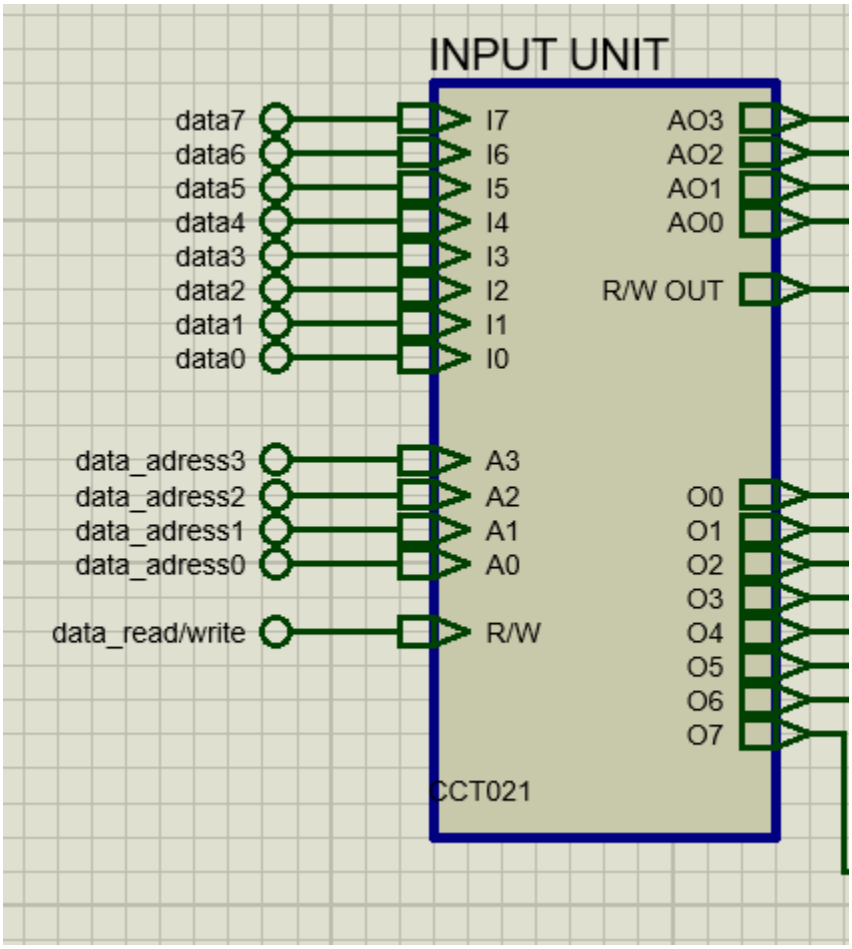


Figure: Input Unit

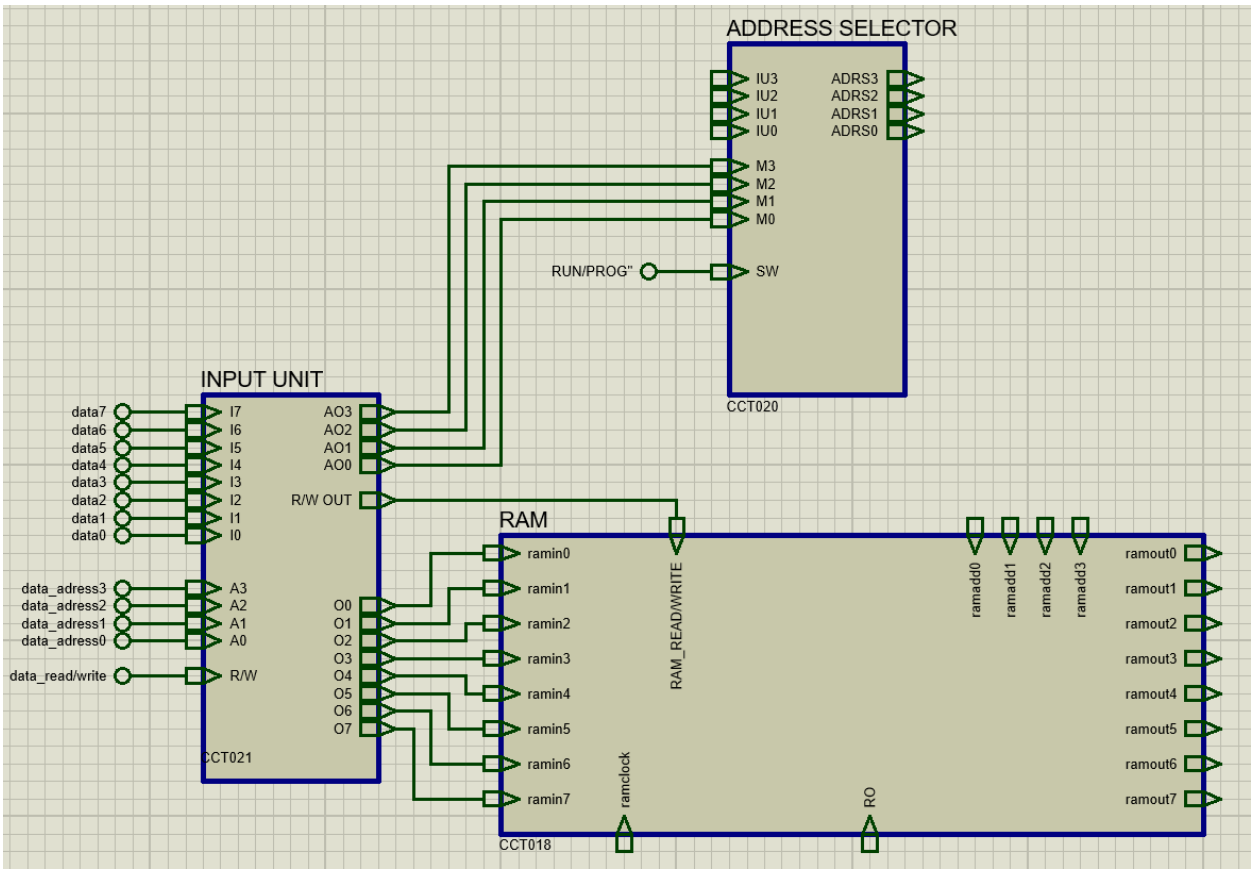


Figure: Adress Selector

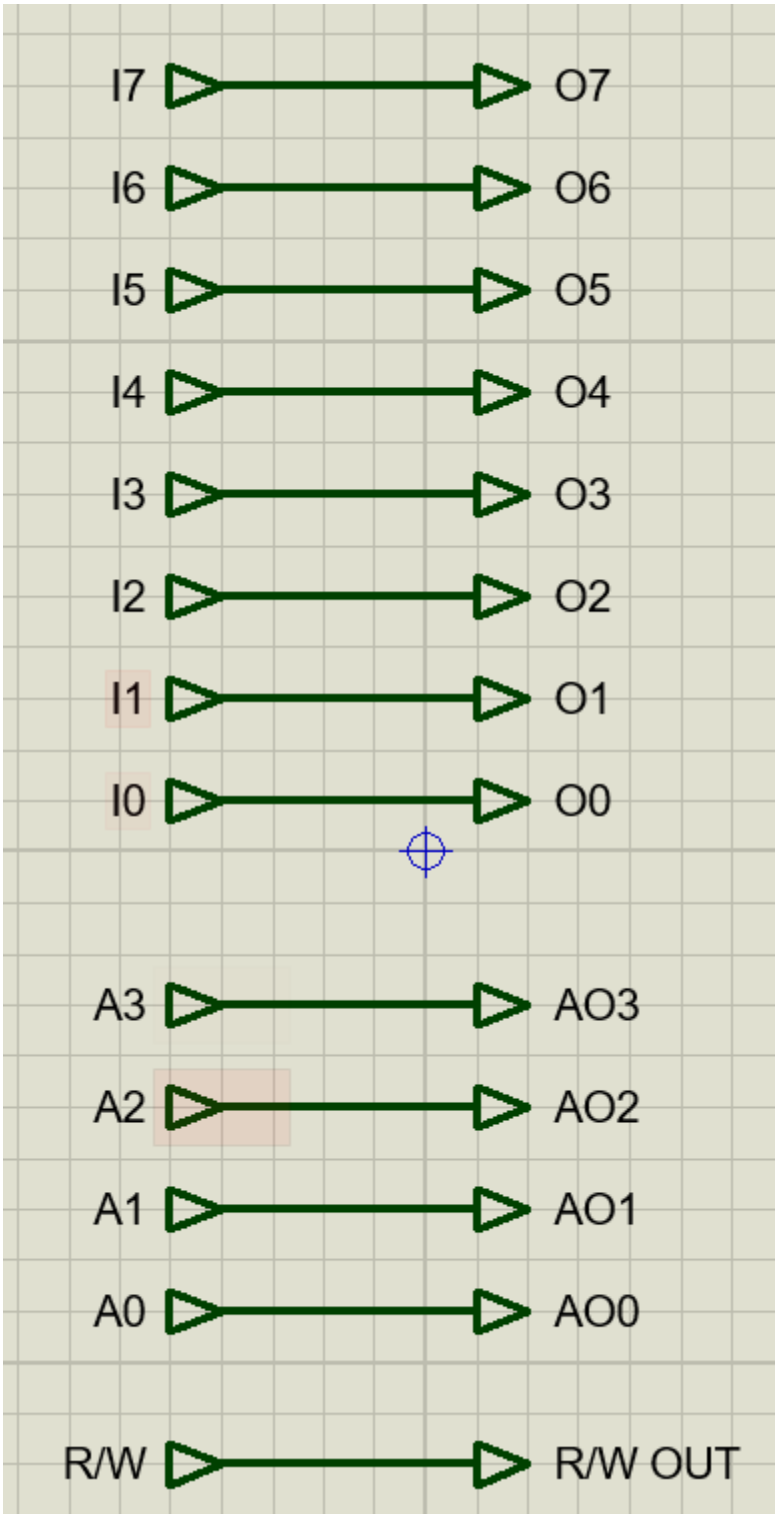


Figure: Input Unit Childsheet

Connections:

From>	Via>	Via>	To
data7	I7 of IU	O7	Ramin7 of RAM
Data6	I6 of IU	O6 of IU	Ramin6 of RAM
Data5	I5 of IU	O5 of IU	Ramin5 of RAM
Data4	I4 of IU	O4 of IU	Ramin4 of RAM
Data3	I3 of IU	O3 of IU	Ramin3 of RAM
Data2	I2 of IU	O2 of IU	Ramin2 of RAM
Data1	I1 of IU	O1 of IU	Ramin1 of RAM
Data0	I0 of IU	O0 of IU	Ramin0 of RAM
Data_adress3	A3 of IU	AO3 of IU	M3 of MAR
Data_adress2	A2 of IU	AO2 of IU	M2 of MAR
Data_adress1	A1 of IU	AO1 of IU	M1 of MAR
Data_adress0	A0 of IU	AO0 of IU	M0 of MAR
Data_read/write	R/W of IU	R/W OUT of IU	RAM READ_WRITE

MAR:

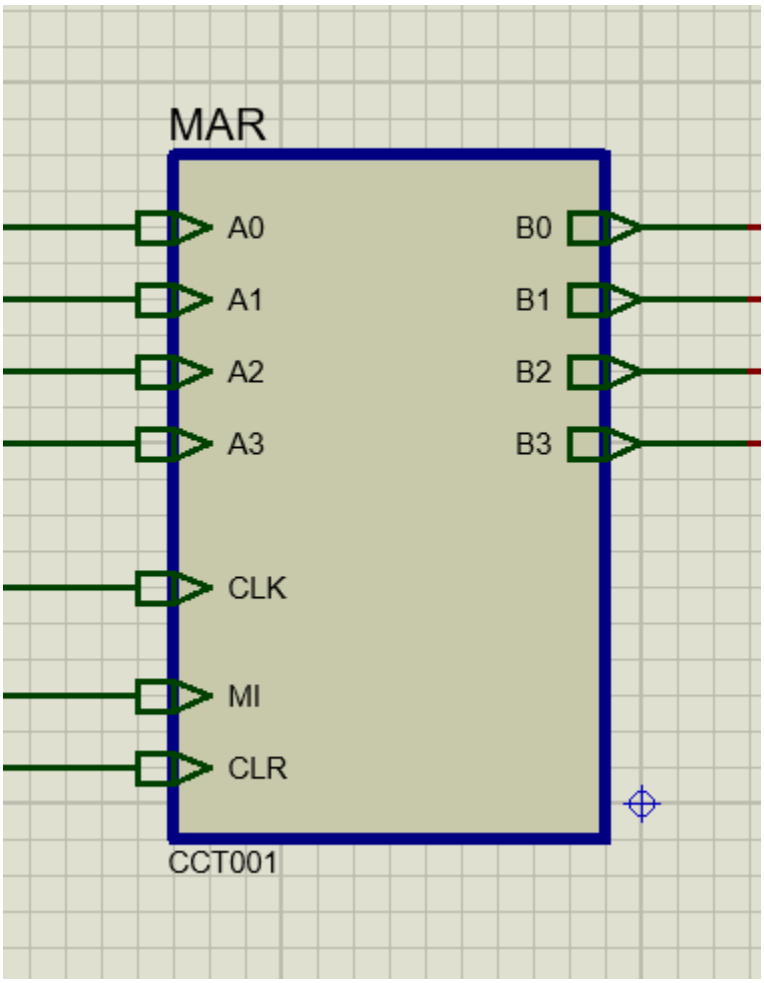
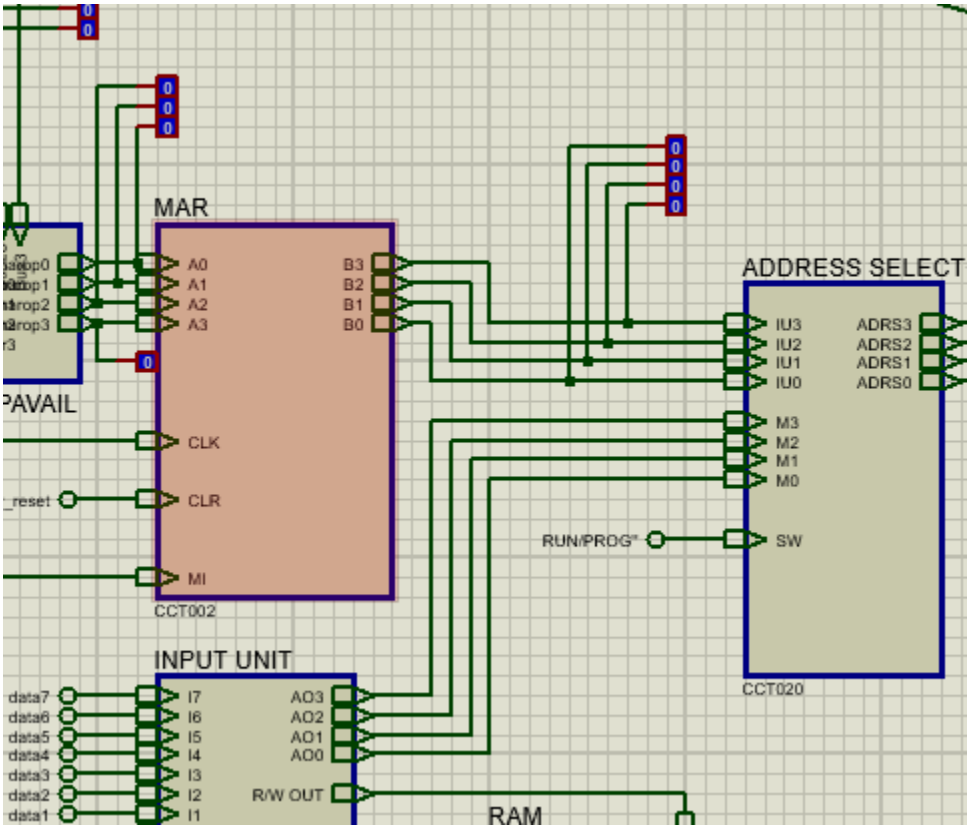
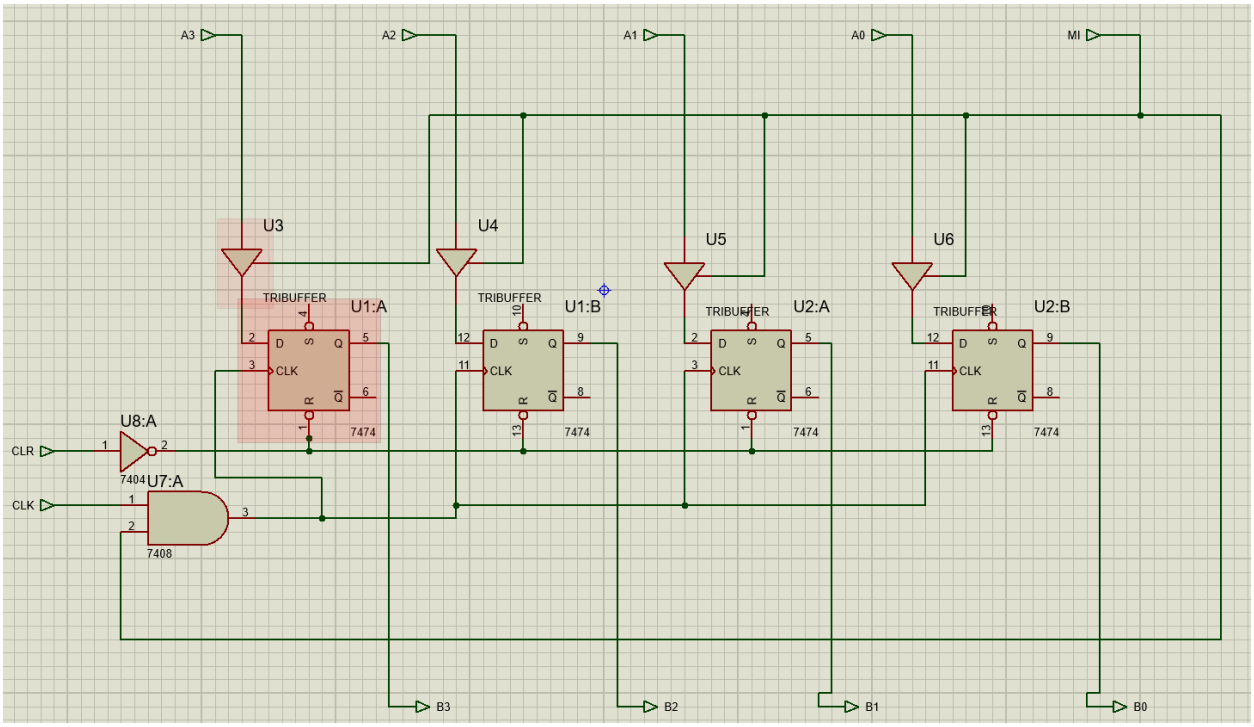


Figure: MAR


MAR
A0
A1
A2
A3
B3
B2
B1
B0
CLK
CLR
MI
CCT002
ADDRESS SELECT
IU3
IU2
IU1
IU0
ADRS3
ADRS2
ADRS1
ADRS0
M3
M2
M1
M0
RUN/PROG*
SW
CCT020
INPUT UNIT
data7
data6
data5
data4
data3
data2
data1
I7
I6
I5
I4
I3
I2
I1
AO3
AO2
AO1
AO0
R/W OUT
RAM


A3
A2
A1
A0
MI
U3
U4
U5
U6
TRIBUFFER
U1:A
U1:B
U2:A
U2:B
D
S
Q
CLK
R
7474
U8:A
7404
U7:A
7408
CLR
CLK
B3
B2
B1
B0

Connections:

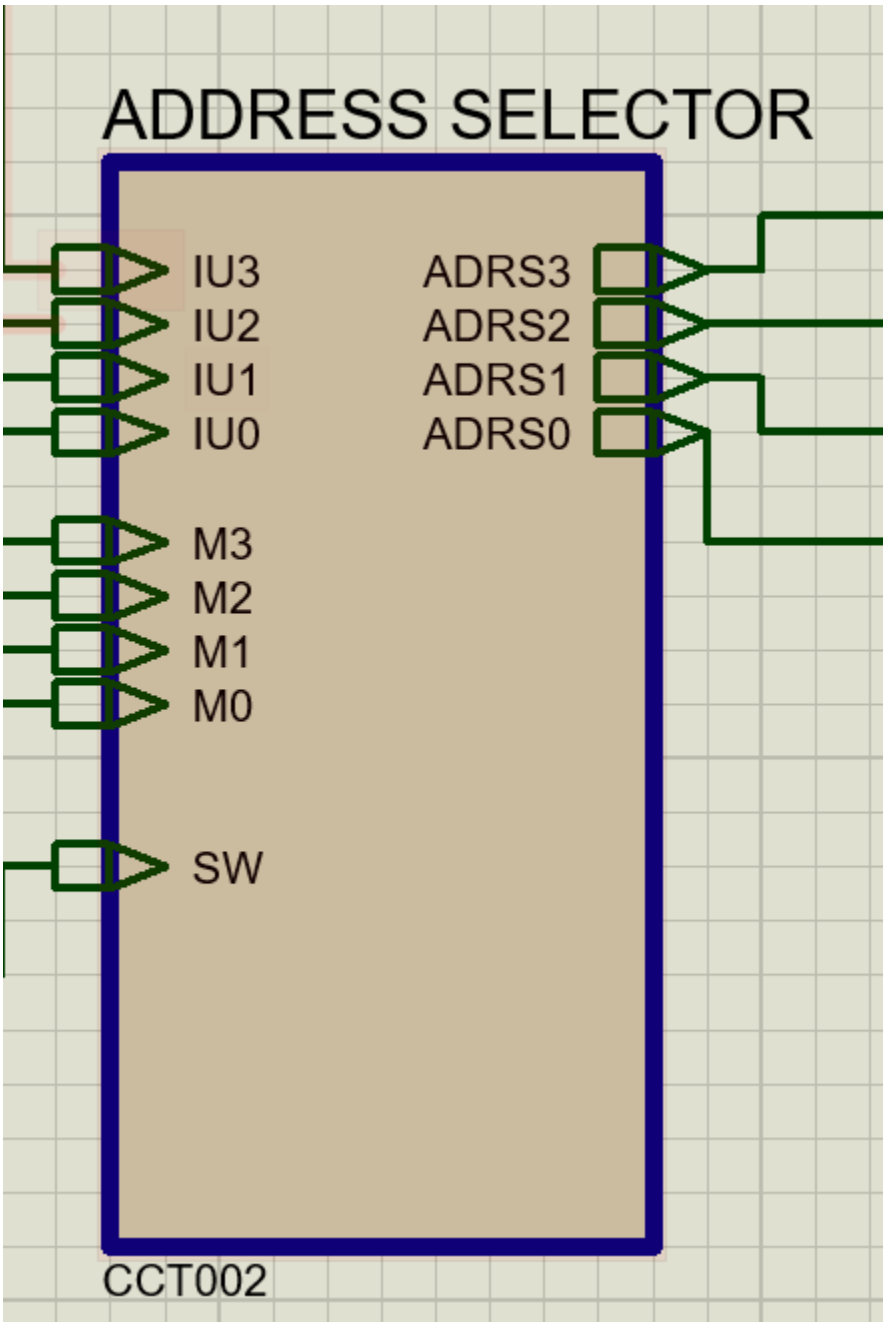
From>	TO	From	TO
Marop3	A3 of MAR	B3 of MAR	IU3 of IU
Marop2	A2 of MAR	B2 of MAR	IU2 of IU
Marop1	A1 of MAR	B1 of MAR	IU1 of IU
Marop0	A0 of MAR	B0 of MAR	IU0 of IU
U56:D (AND Gate)	CLK of MAR		
Mater_Reset	CLR of MAR		
Ctrl_ro	MI of MAR		

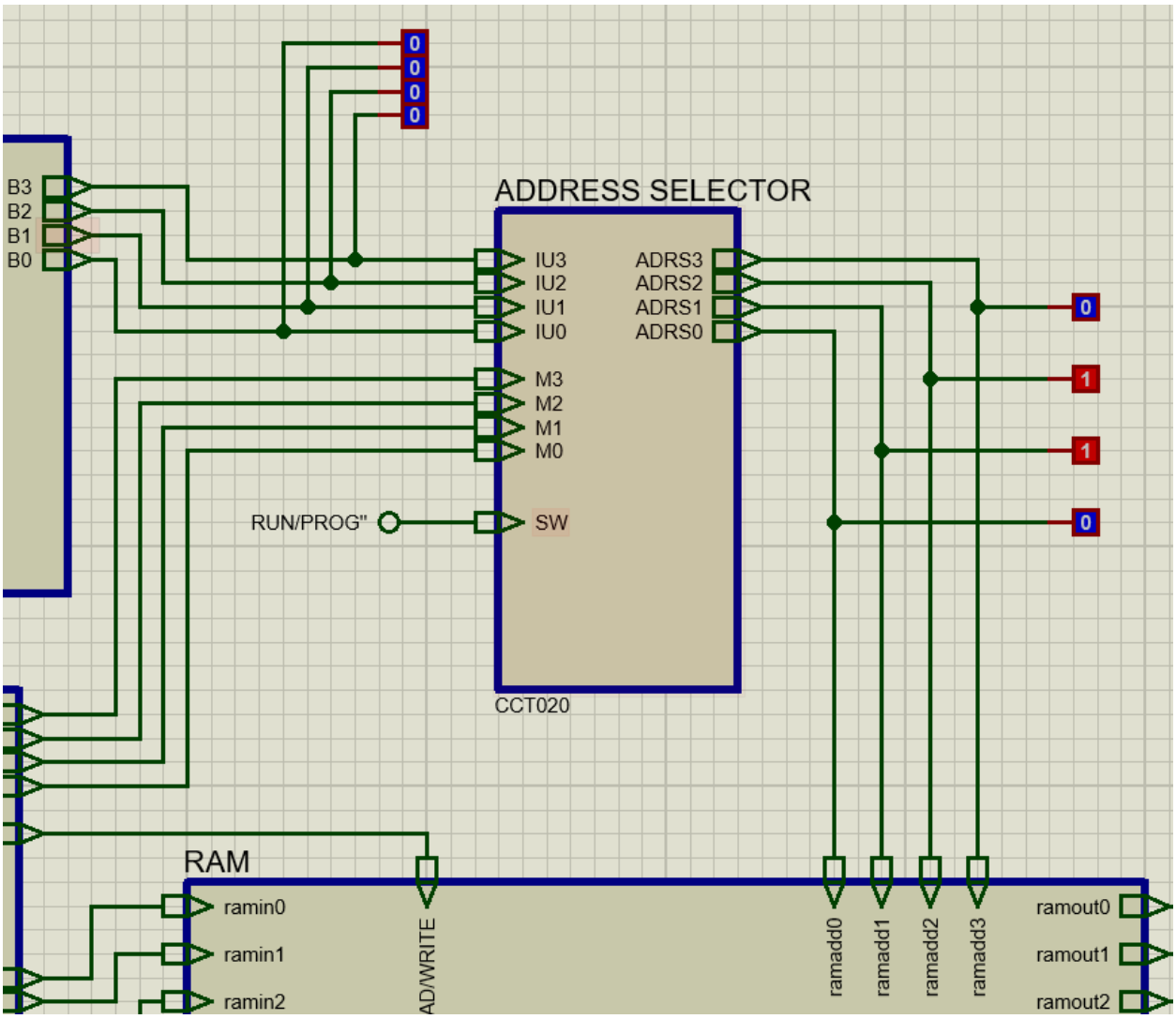
Truth Table:

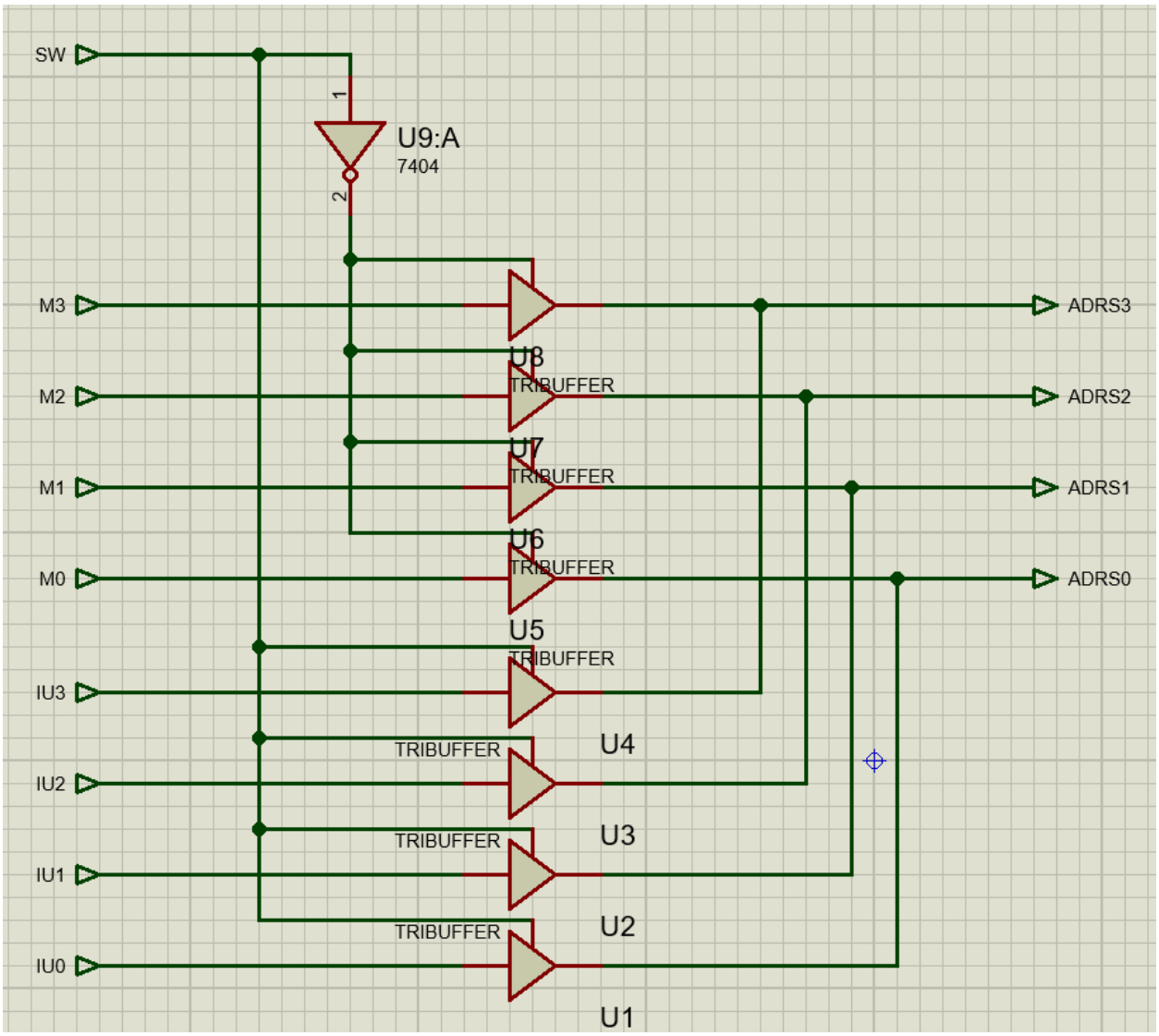
A0	A1	A2	A3	MI	CLR	B0	B1	B2	B3
0	0	0	0	1	0	0	0	0	0
0	0	0	1	1	0	0	0	0	1
0	0	1	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0	1	1
0	1	0	0	1	0	0	1	0	0
0	1	0	1	1	0	0	1	0	1
0	1	1	0	1	0	0	1	1	0
0	1	1	1	1	0	0	1	1	1
1	0	0	0	1	0	1	0	0	0
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	1	0	1	0
1	0	1	1	1	0	1	0	1	1
1	1	0	0	1	0	1	1	0	0
1	1	0	1	1	0	1	1	0	1
1	1	1	0	1	0	1	1	1	0
1	1	1	1	1	0	1	1	1	1
0/1	0/1	0/1	0/1	0	0	Previous state	Previous State	Previous State	Previous State
0/1	0/1	0/1	0/1	0/1	1	0	0	0	0

If Clock pulse is 0 then then also the output don’t change and it stores the previous state.

ADDRESS Selector:




0
0
0
0
B3
B2
B1
B0
ADDRESS SELECTOR
IU3
IU2
IU1
IU0
ADRS3
ADRS2
ADRS1
ADRS0
M3
M2
M1
M0
RUN/PROG"
SW
CCT020
0
1
1
0
RAM
ramin0
ramin1
ramin2
AD/WRITE
ramadd0
ramadd1
ramadd2
ramadd3
ramout0
ramout1
ramout2



Connections:

From	TO
B3 of MAR	IU3 of Address selector
B2 of MAR	IU2 of Address selector
B1 of MAR	IU1 of Address selector
B0 of MAR	IU0 of Address selector
AO3 of IU	M3 of Address selector
AO2 of IU	M2 of Address selector
AO1 of IU	M1 of Address selector
AO0 of IU	M0 of Address selector
ADRS3 of Address selector	Ramadd3
ADRS2 of Address selector	Ramadd2

ADRS1 of Address selector	Ramadd1
ADRS0 of Address selector	Ramadd0
SW of Address selector	RUN/PROG button

Truth Table:

SW	ADDRS3	ADDRES2	ADDRES1	ADDRES0
0	M3	M2	M1	M0
1	IU3	IU2	IU1	IU0

Phase B (Memory Unit)

I) Registers

Accumulator

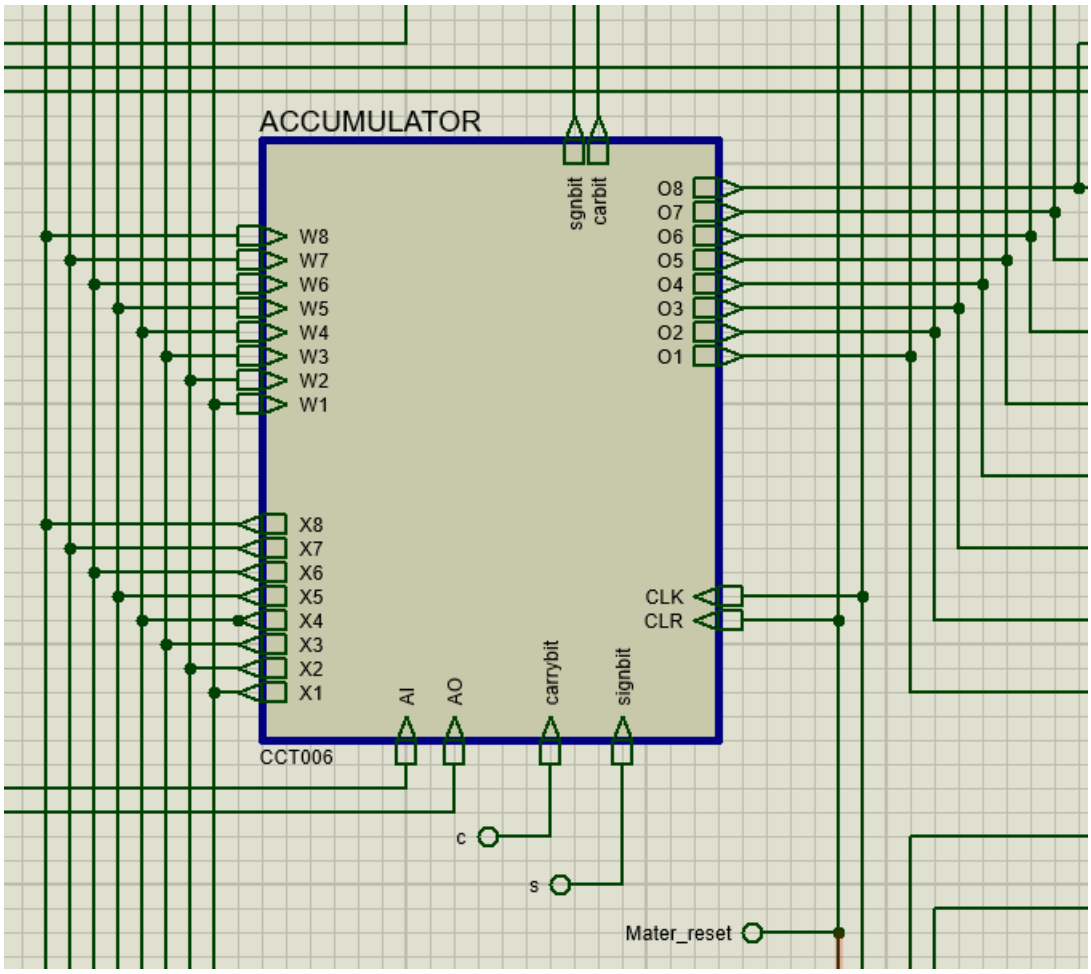


Figure: Accumulator

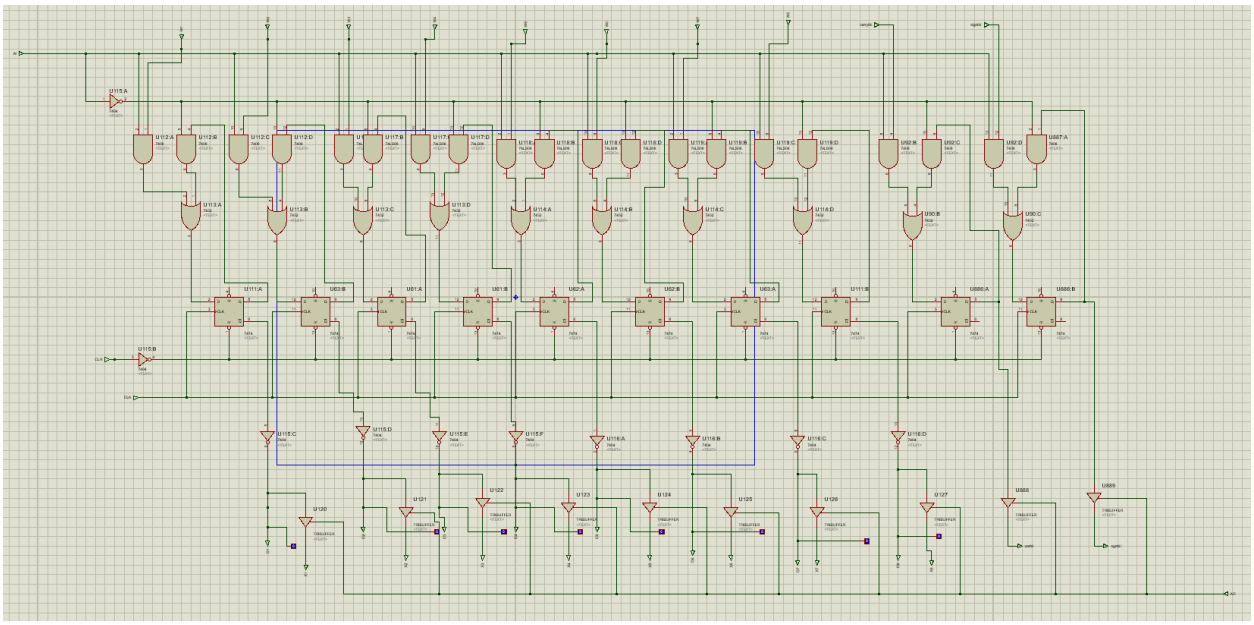


Figure: Accumulator(child sheet)

Accumulator is a register that stores the values obtained from the bus and sends it to the ALU for arithmetical operation.

It has been designed using 10 positive edge triggered JK flip flops (2 additional flip flops have been used for storing the sign bit and carry bit from the ALU directly). Here a combination of AND and OR gates have been used with the JK flip flops to store the previous output when and also to load new values when any one of them is necessary.
Here, AI pin controls the entrance of the values from the bus to the accumulator. This comes via W1 to W8.
AO pin (designed using tri buffers) controls the flow of the accumulator values to the bus via X1 to X8. Also, the output of accumulator is directly connected to the ALU without any enable pins via O1 to O8.

B Register

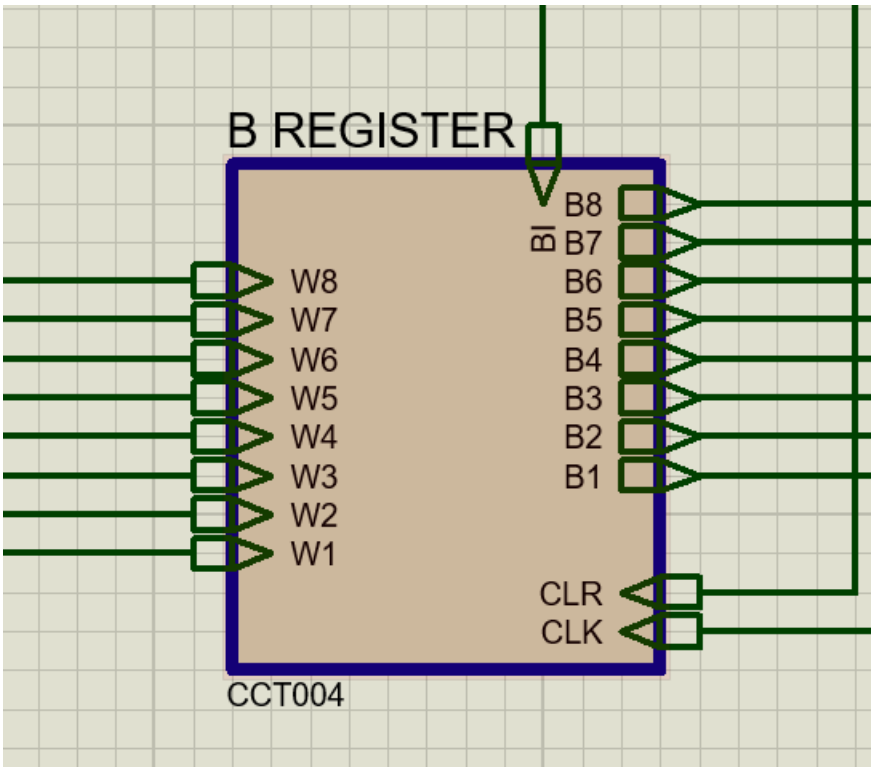


Figure: B Register

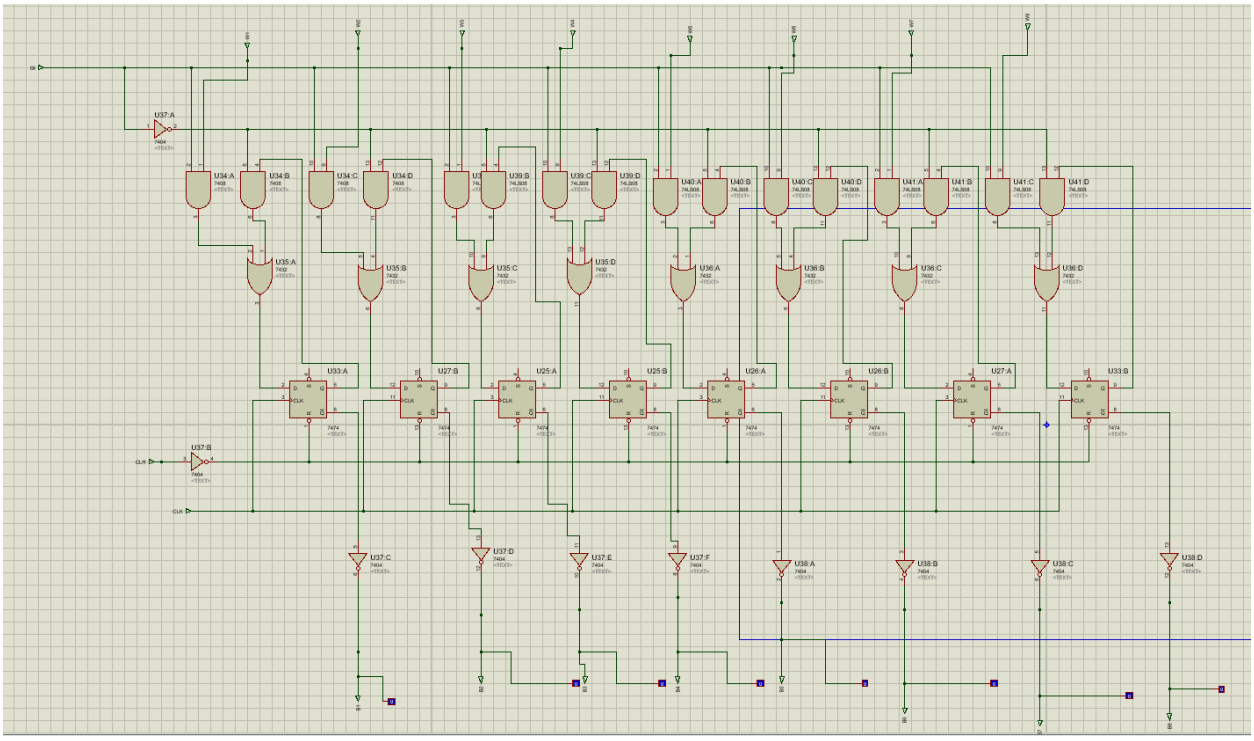


Figure: B Register (child sheet)

B Register is an 8 bit register which stores the value of the output obtained from the bus (via W1 to W8) when BI pin is enabled. The value is then sent to the ALU (via B1 to B8) for arithmetical operation. The basic internal configuration of B Register is like that of the Accumulator.

Output Register

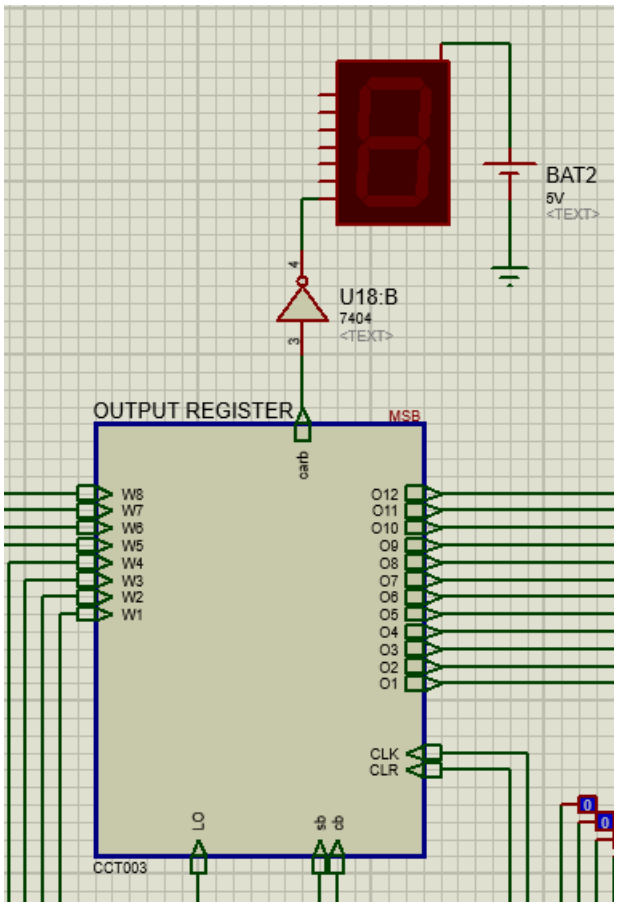


Figure: Output Register

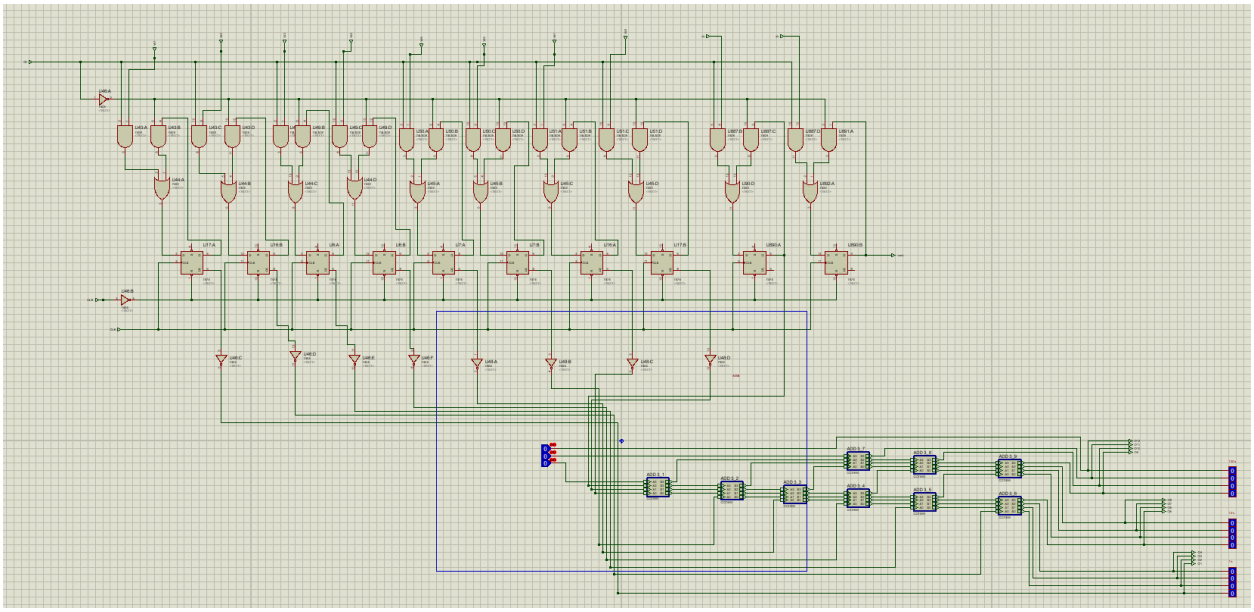


Figure: Output Register (Child Shee)

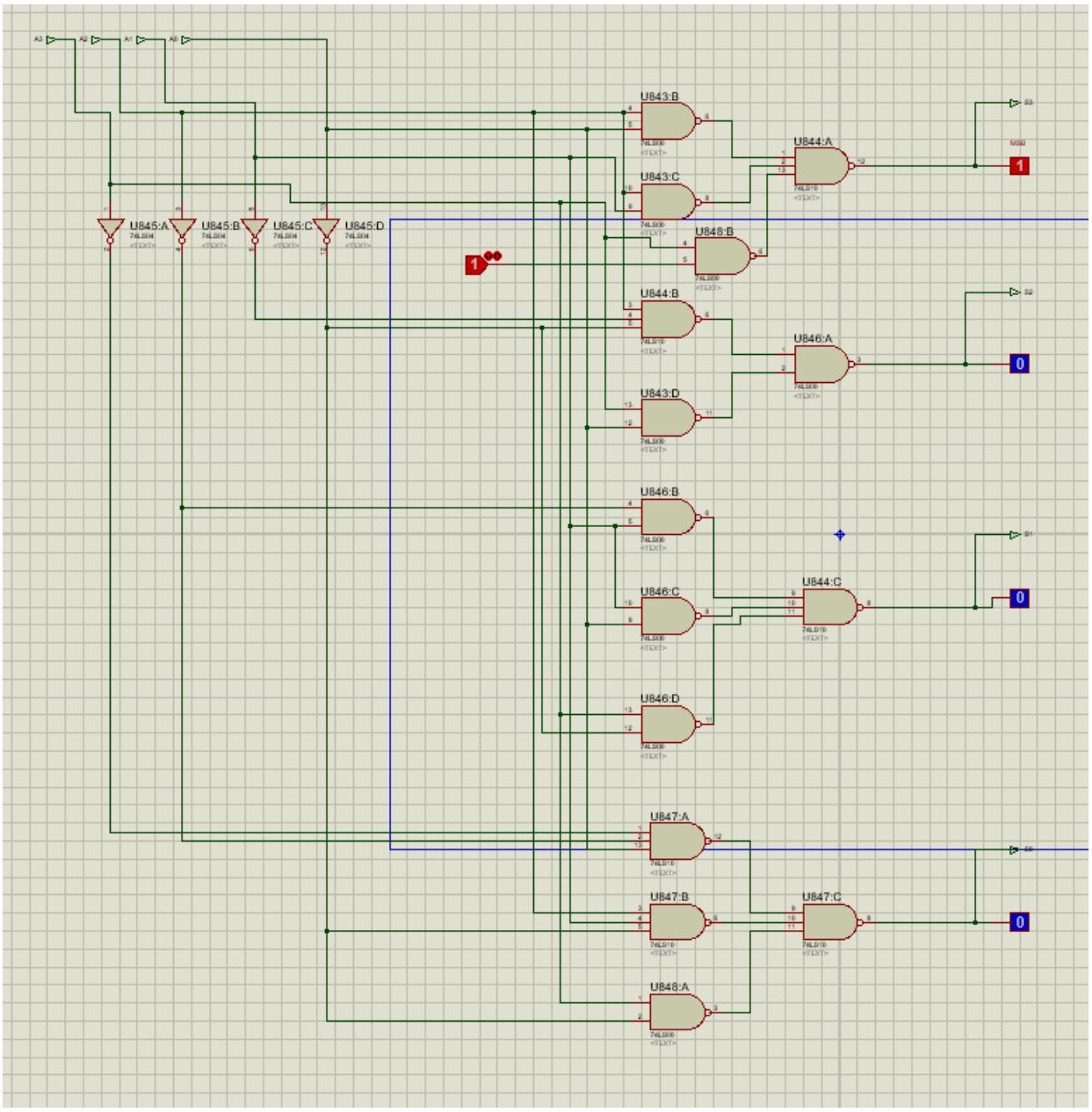


Figure: Checker and Adder Circuit (Child Sheet)

Output register stores the value of the output values obtained from the bus (W1 to W8) when OI is, and then sends them out to BCD to 7 segment decoder display via O1 to O8.

The basic internal building of the output register is similar to the B register. Here one carry bit (cb) and one sign bit(sb) has been taken

directly fed from the accumulator in parallel with the normal inputs W1 to W8.

The final binary result has been converted to decimal form using a technique called Double Dabble. Here each bit is shifted three times and then a checking is done to see if its greater than or equal to binary 101. If it is greater than binary 101 then 11 is added with those initial three shifted bits and then another shifting is done. If it is less 101 than no addition is done, and a regular shifting is done. The following truth table was developed for building the adder and checker circuit.

A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1010 to 1111				X	X	X	X

K-map was used to find out the Boolean Expression (SOP) of the outputs. The result has been given below:

S3= A2A0 + A2A1 + A3

S2= A2A1’A0 + A3A0

S1= A2’A1 + A1A0 + A3A0’

S0= A3’A2’A0 + A2A1A0’ + A3A0’.

Two level NAND implementation way was used to implement the above

combination circuit.

In this way the output binary number is checked and (added; if needed) shifted 9 times. A combinational circuit has been used to convert the 9(1 extra carry bit) bit binary number to BCD numbers. The converted BCD is then fed to the output unit.

II) Random Access Memory (RAM)

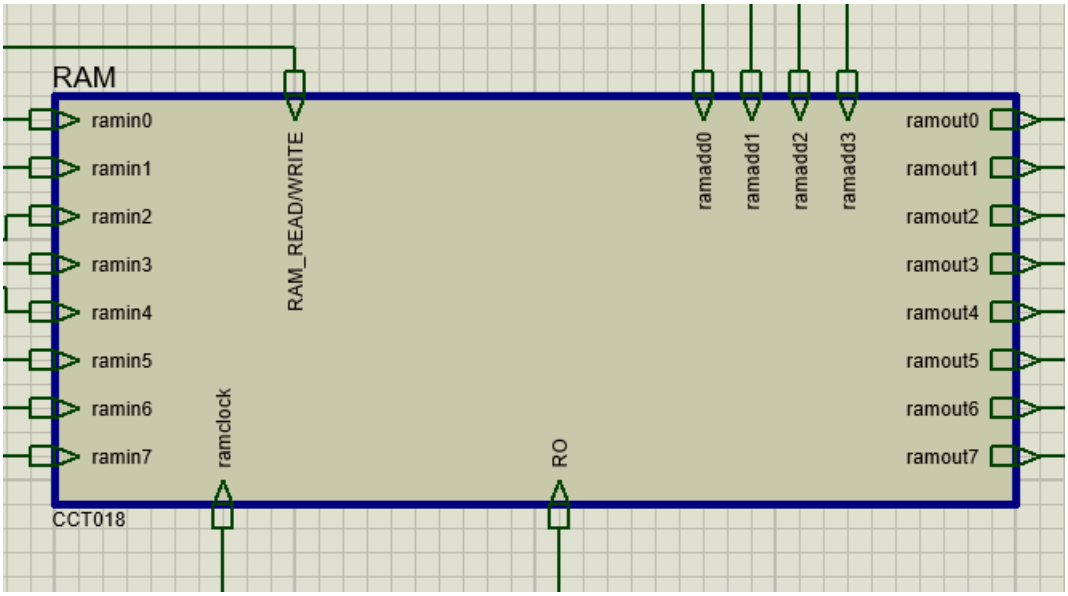


Figure: RAM

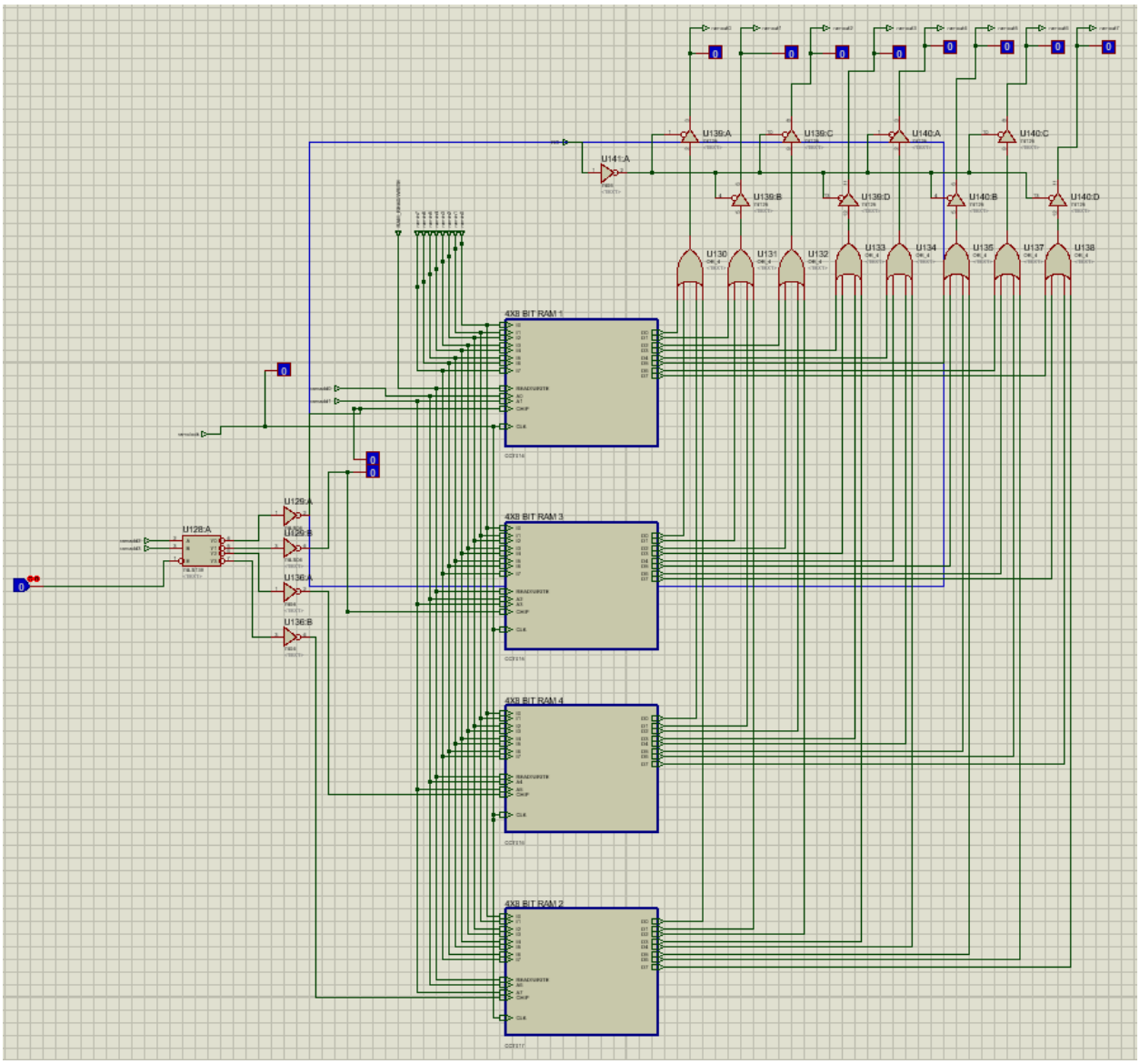


Figure: RAM Childheet

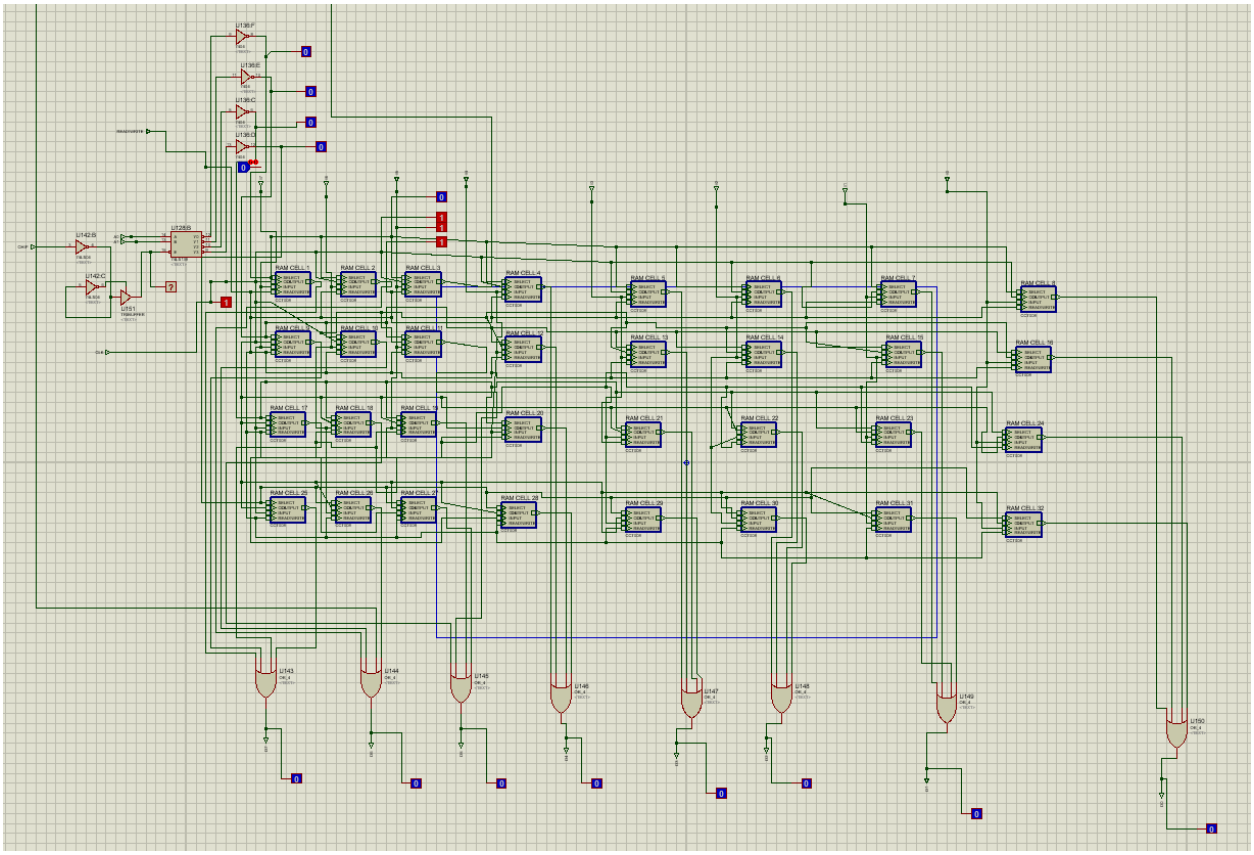


Figure: RAM Chip (Childsheet)

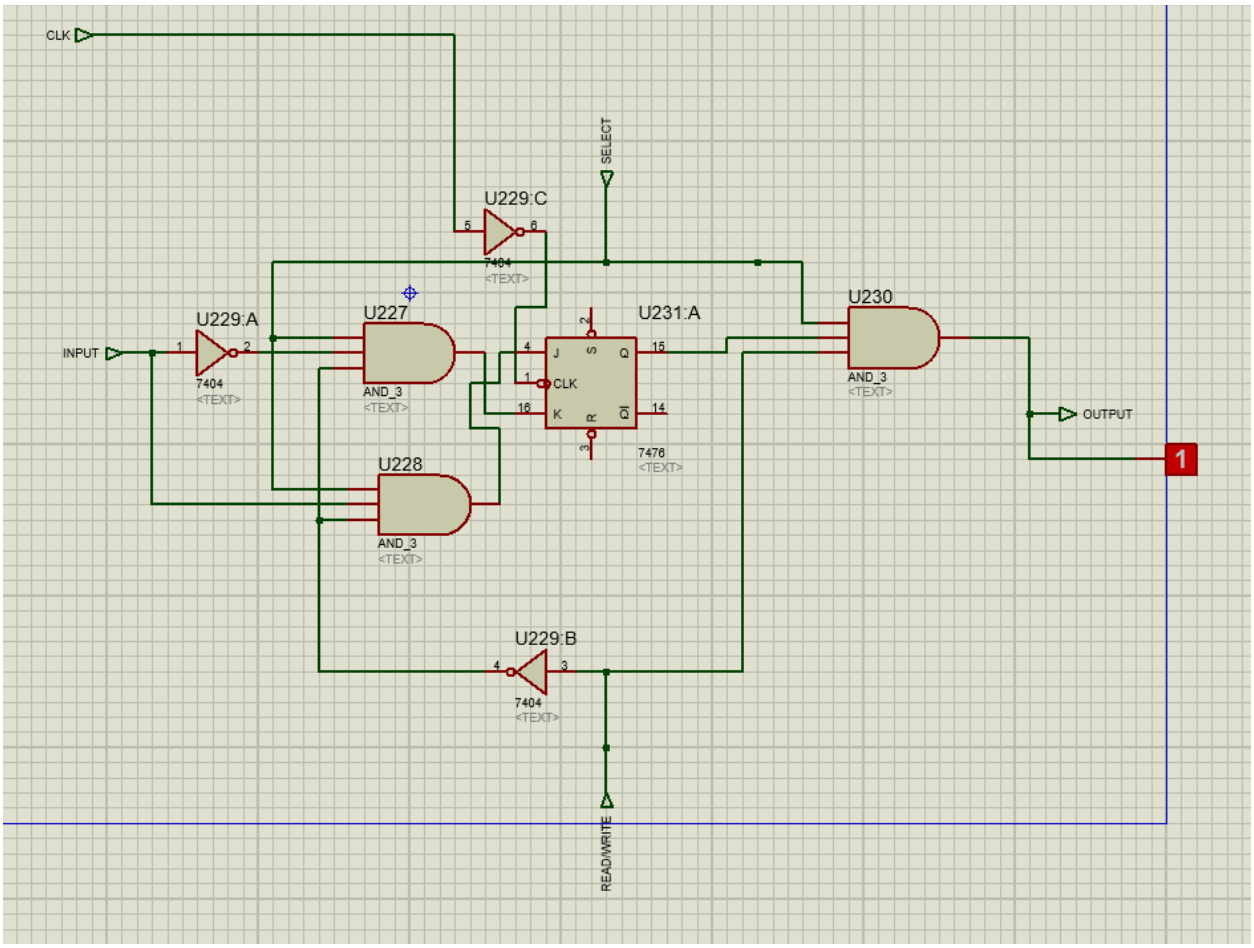


Figure: RAM Cell (Childsheet)

RAM stores the instructions and data to be executed before the computer is run. It is a 16x8 bit ram, meaning it has 16 address locations and each address location can store 8 bits of binary data. The location of the address has been selected by means of an address selector. The RAM has been designed using four 4x8 bit RAM chips. Each of the RAM chips have been designed using 1 bit RAM cell. The 1-bit RAM cell has been designed using positive edge triggered JK flip flop. A level triggered flip flop could have been used but the problem with this is that, while giving the input sometimes garbage values get stored. Also while switching between different address location, every time the read and write mode has to be toggled, otherwise the input of the previous address gets stored in the next address. The advantage of

edge trigger is that all the inputs can be given in write mode and neatly each input can be given using a manual pulse ensuring no garbage values gets stored. Also switching between different addresses is easy as we can do it and simply give a manual pulse after switching. Each of the four ram chips has been selected using a 2 into four decoder. The output of the RAM is controlled by the RO pin.

Phase C

Arithmetic and Logic Unit(ALU)

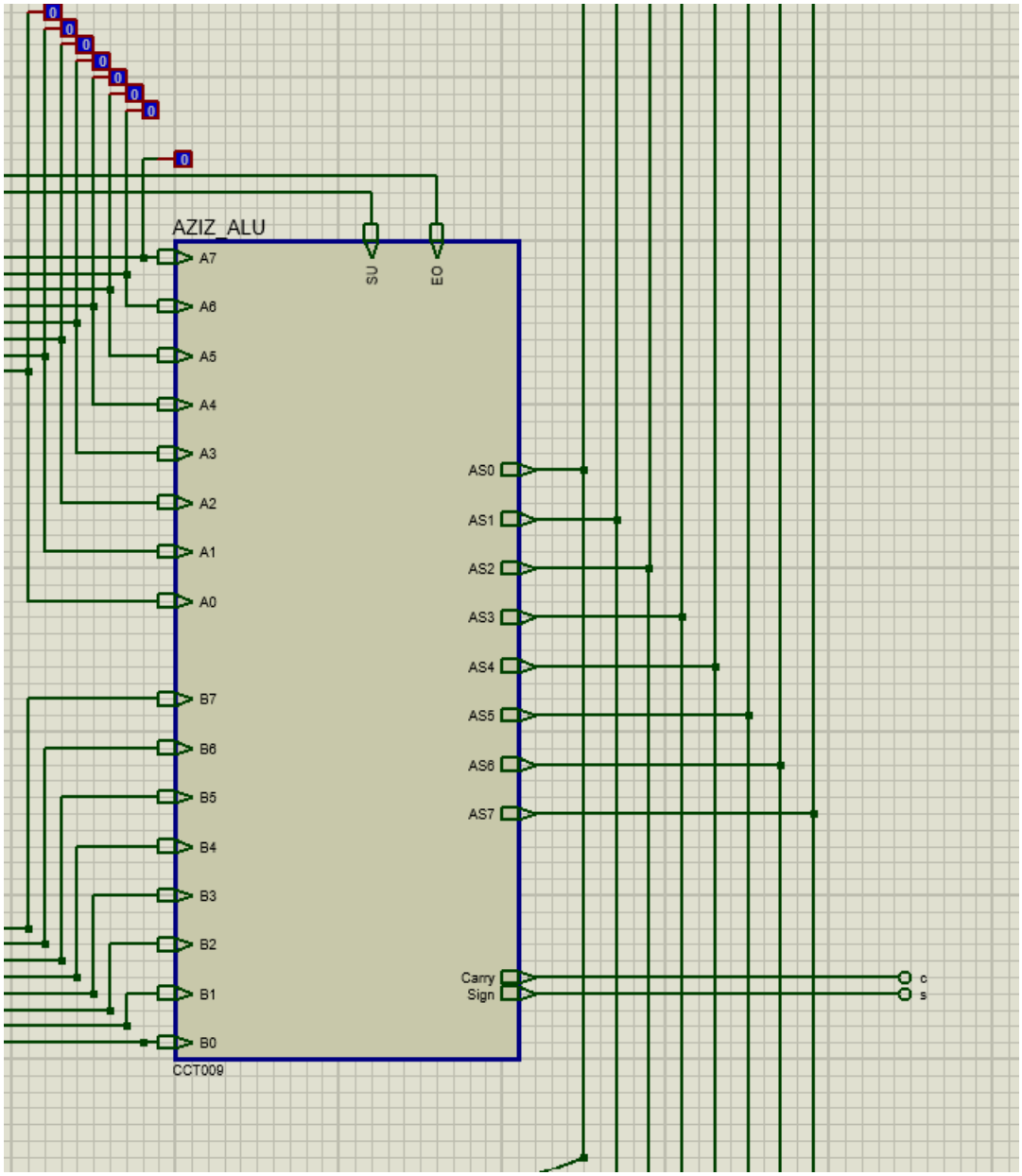


Figure: Arithmatical Logic Unit

The purpose of ALU is to add or subtract numbers, so full adder was built:

Truth table for 2 full bit adder:

Input			Output	
A	B	Carry	Sum	Carry Out
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

So Sum= A ⊕ B ⊕ Carry

Carry Out= A*B+ Carry* A ⊕ B

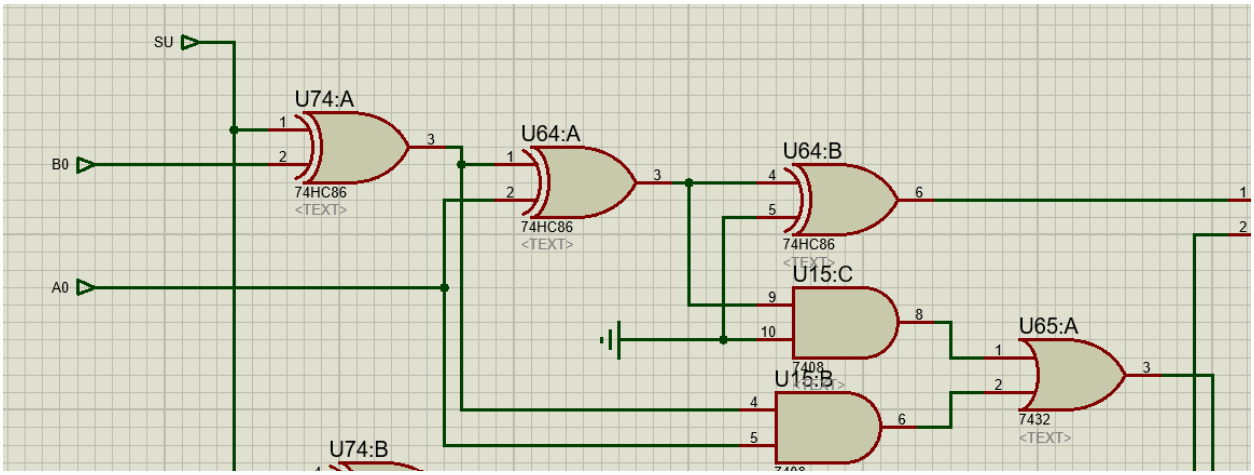


Figure: 1 bit adder and subtractor

8 no of 2 bit full bit adder were cascaded to get 8 bit full adder.

The property of XOR gate:

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

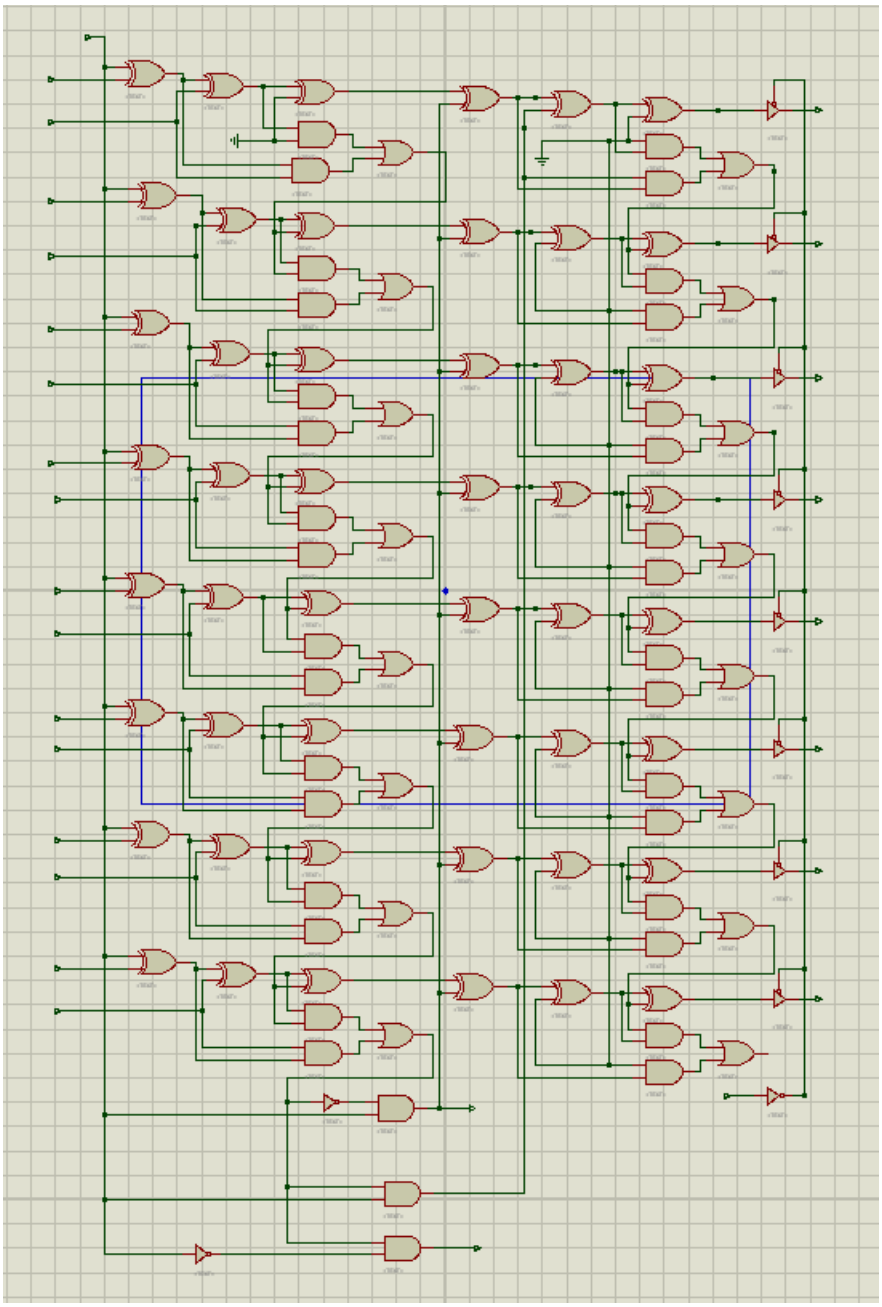


Figure: 8 bit full adder and subtractor(binary result with carry and sign bit)

So from the properties XOR gate, when one input is 0 the output is same but when one input is 1 the output is 1's complement of the input. Using this property we conducted addition when controlling bit SU= 0 and when SU=1, we got 1's complement of the input from B

register by passing through an XOR gate and giving SU bit to all the XOR gates.

From the added result we gave it to bus when SU=0 but when SU=1 we converted the result to binary. To do that when we have carry =1, we discarded carry and added 1 with another 8 bit full adder. When SU=0 and carry=0 we used an XOR gate on the output of the first adder to get 1's complement of the result and the binary value.

For Sign bit and carry bit:

Input		Output	
SU	Carry	Sign bit	Carry bit
0	0	0	0
0	1	0	1
1	0	1	X
1	1	0	X

So sign bit = SU*Carry'

Carry bit= SU'*Carry

All the output pins are connected to tribuffer which gives the output to bus when EO=1

Phase D (Control Unit)

I) INSTRUCTION REGISTER

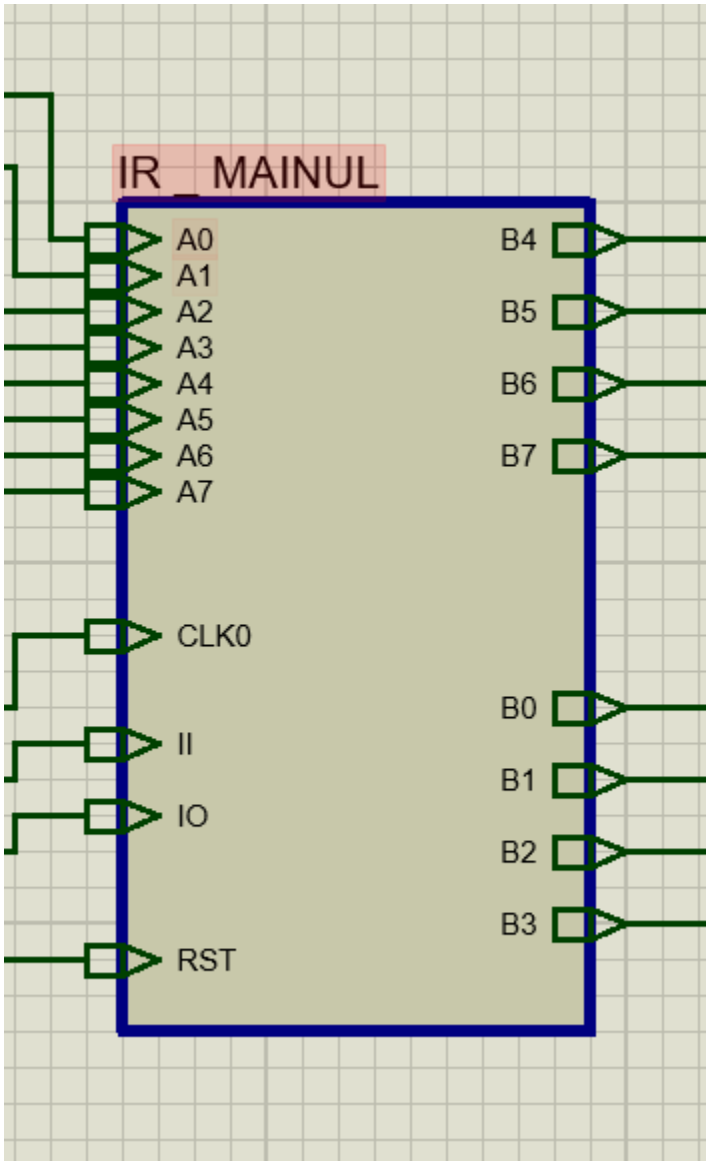


Figure: Instruction Register

Truth Table:

II	IO	RST	B0	B1	B2	B3	B4	B5	B6	B7
0	0	0	undefined	undefined	undefined	undefined	Prev. state	Prev. state	Prev. state	Prev. state
0	0	1	undefined	undefined	undefined	undefined	0	0	0	0
0	1	0	prev. state	Prev. state	Prev. state	Prev. state	Prev. state	Prev. state	Prev. state	Prev. state
0	1	1	0	0	0	0	0	0	0	0
1	0	0	Undefined	Undefined	Undefined	Undefined	A4	A5	A6	A7
1	0	1	undefined	undefined	undefined	undefined	0	0	0	0
1	1	0	A0	A1	A2	A3	A4	A5	A6	A7
1	1	1	0	0	0	0	0	0	0	0

Connections:

From	TO	From	TO
Bus-line 00	A0 of IR	B0 of IR	Bus-line 00
Bus-line 01	A1 of IR	B1 of IR	Bus-line 01
Bus-line 02	A2 of IR	B2 of IR	Bus-line 02
Bus-line 03	A3 of IR	B3 of IR	Bus-line 03
Bus-line 04	A4 of IR	B4 of IR	Opcode0 of CS
Bus-line 05	A5 of IR	B5 of IR	Opcode1 of CS
Bus-line 06	A6 of IR	B6 of IR	Opcode2 of CS
Bus-line 07	A7 of IR	B7 of IR	Opcode3 of CS
Mater_reset	RST of IR	Ctrl_clk of ‘CS’ ,MARCLK0 of ‘IR’, CLK of ‘CP’ shorted	
Ctrl_io	MARIO of IR	Ctrl_ii	MARII of IR

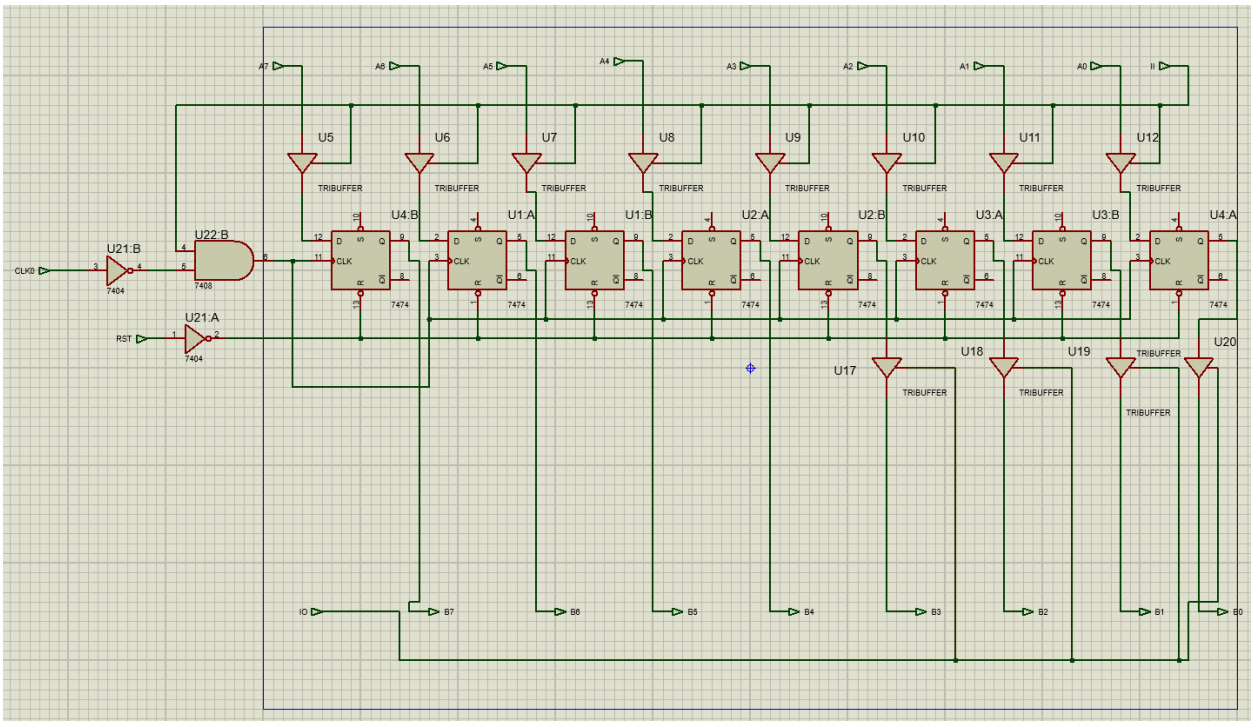


Figure: Instruction Register Childsheet

II) Controller Sequencer:

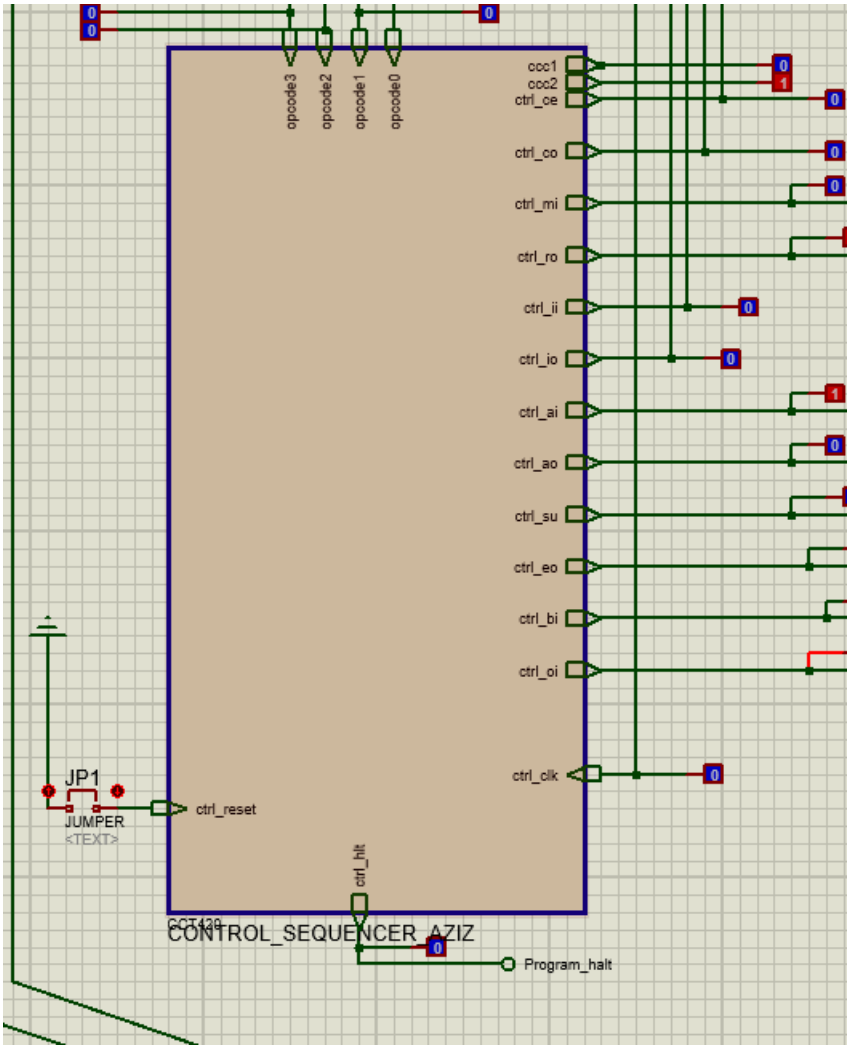


Figure: Controller Sequencer

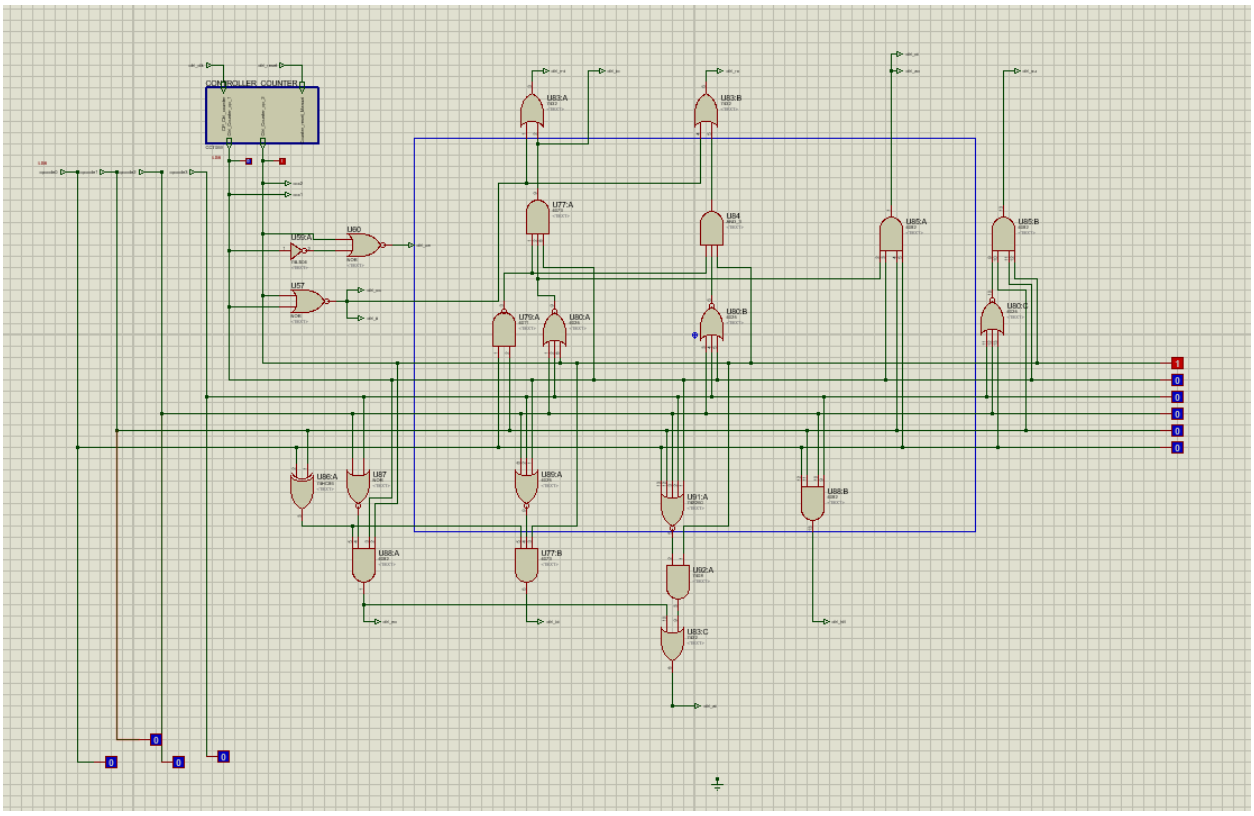


Figure: Controller Sequencer Childsheet

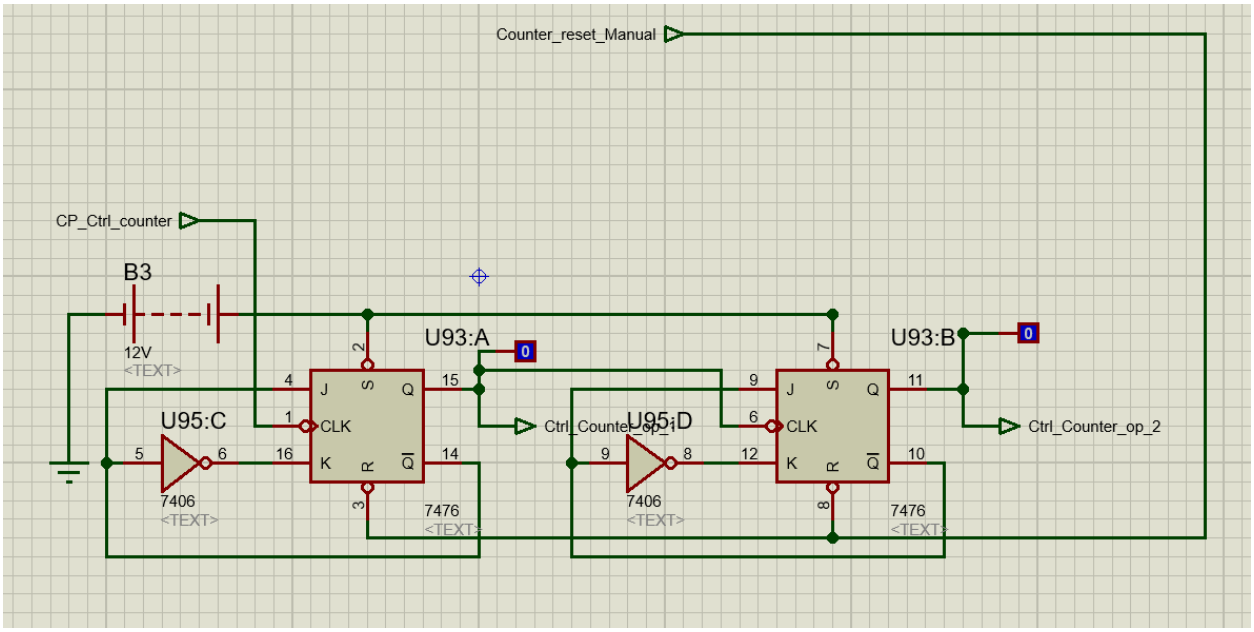


Figure: T State counter

Ctrl_count			Instruct.reg.				Out.											
Q2	Q1		OP3	OP2	OP1	OP0	CE	CO	MI	RO	II	IO	AI	AO	SU	EO	BI	OI
0	0		x	x	x	x		1	1	1	1							
0	1		x	x	x	x	1											
0	1		0	0	0	0			1				1					
1	0		0	0	0	0				1				1				
1	1		0	0	0	0												
0	1		0	0	0	1			1				1					
1	0		0	0	0	1				1							1	
1	1		0	0	0	1								1				
0	1		0	0	1	0			1				1					
1	0		0	0	1	0				1							1	
1	1		0	0	1	0								1		1		
0	1		0	0	1	1									1			1
1	0		0	0	1	1												
1	1		0	0	1	1												
0	1		1	1	1	1	main pulse stop											

Figure: Truth Table

1. CE = $\bar{Q}_2 \cdot Q_1$

2. CO = $\bar{Q}_2 \cdot \bar{Q}_1$ = AA $\bar{Q}_2 Q_1$

3. MA = $Q_2 \cdot \bar{Q}_1 + (\overline{OP_3}\ \overline{OP_2}\ \overline{OP_1}\ \overline{OP_0} + \overline{OP_3}\ \overline{OP_2}\ \overline{OP_1}\ OP_0 + \overline{OP_3}\ \overline{OP_2}\ OP_1\ \overline{OP_0})$

$= AA + Q_1 \overline{(Q_2 + OP_2 + OP_3)}\ \overline{(OP_0 \times OP_1)}$

$= AA + AO$

4. RO = AA + $Q_2\ \overline{Q_1}\ \overline{OP_3}\ \overline{OP_2}\ (\overline{OP_1} + \overline{OP_0})$

$= AA + Q_1 \overline{(Q_1 + OP_2 + OP_2)}\ \overline{(OP_1 \times OP_0)}$

5. Since From truth table,

AO = $\overline{(Q_2 + OP_2 + OP_3)} \cdot \overline{(OP_0\ OP_0 + OP_1)} \times Q_1$

so, AO can be taken from 2nd part of MA

6. AA = EO + $Q_2 \bar{Q}_1\ \overline{(OP_0 + OP_1 + OP_2 + OP_3)}$

$= EO + Q_2 \overline{(Q_1 + OP_0 + OP_1 + OP_2 + OP_3)}$

7. $AO = \overline{(Q_2 + OP_3 + OP_2)}$ $Q_1 \times OP_1 \times OP_0 = OS$

8. $SV = \overline{(OP_3 + OP_2 + OP_0)} \times OP_1 \times Q_2 \times Q_1$

9.
$EO = Q_2 Q_1 \overline{OP_3}\ \overline{OP_2}\ (OP_0 \oplus OP_1)$

$= (Q_2 Q_1)\ \overline{(OP_3 + OP_2)}\ (OP_0 \oplus OP_1)$

10.
$BS = Q_2\ \overline{(Q_1 + OP_3 + OP_2)}\ (OP_0 \oplus OP_1)$

Figure: Controller Sequencer Boolean Expressions

Controller sequencer includes a time counter for determining the microinstructions. Using the output (2) of these counter and from the output taken from the instruction register, a combinational circuit was designed which turns on control bit at different micro instruction.

Phase E

I) Integration with 8-bit Bus

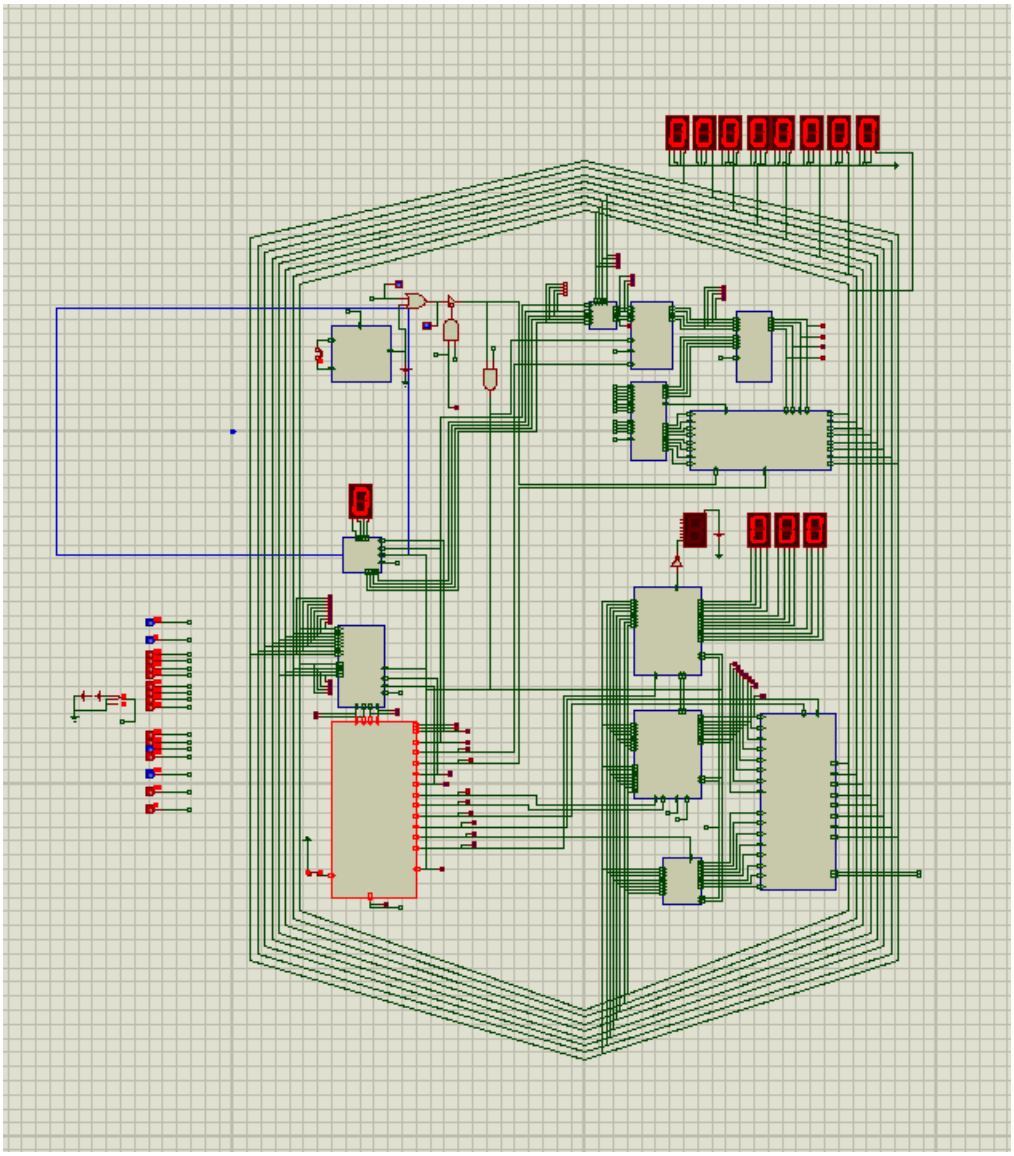


Figure: Integration with 8-bit Bus

II) Output Unit

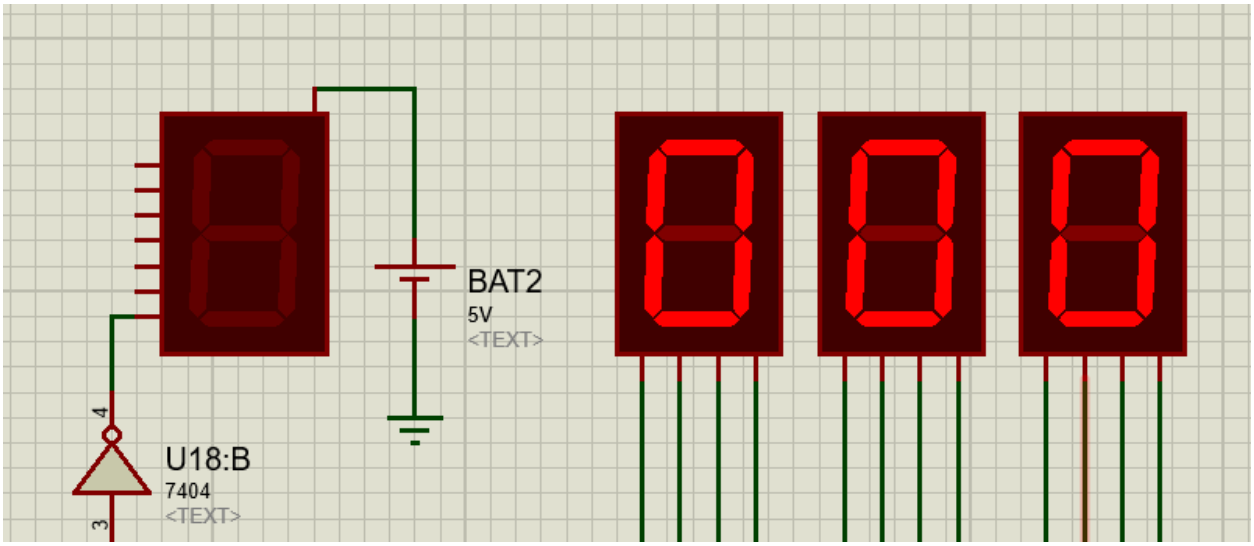


Figure: Output Unit

The final result has been displayed in Decimal number using a BCD display. For showing a negative sign(if the result of subtraction is negative) a common anode was used which has been powered by a 5V battery. The internals of the output display unit has been explained in the Output Register Part.

Problems:

*For clock pulse generator, we were getting a fatal error message and the simulation was aborted when we tried run the program in automatic pulse from the starting. To solve that we used a pull down resistor in the output of astable mutivibrator. Still we did not get acquired result as it only gave stable 0 state but not 1. So we used a voltage source in the output to boost the output voltage level.

*In the ram we did not use the subcircuit name as accordingly so we suffered from unusual behavior of ram.

*In reducing the microinstruction when the input from program counter was connected with MAR with the bus there was clash of values, so to get acquired result we used a MUX.

*In reducing microinstruction, we also faced problems as in the first T-state from negative pulse to positive there is a garbage value and since the IR triggered before RAM sending desired output in the BUS, we got garbage instruction stored in the IR. To resolve it, we used multiple number(14) of NOT gates to delay the triggering and to get acquired instruction stored in the IR.

*After integrating all the components in the project we did not firstly include the output of ALU in the BUS. So the Accumulator after always stored garbage value which was resolved when the output was given.

*Despite producing the sign bit and carry bit from the ALU, the value of sign bit and carry bit was not being stored in the ALU. So these two bits were send directly to the accumulator and the output unit.