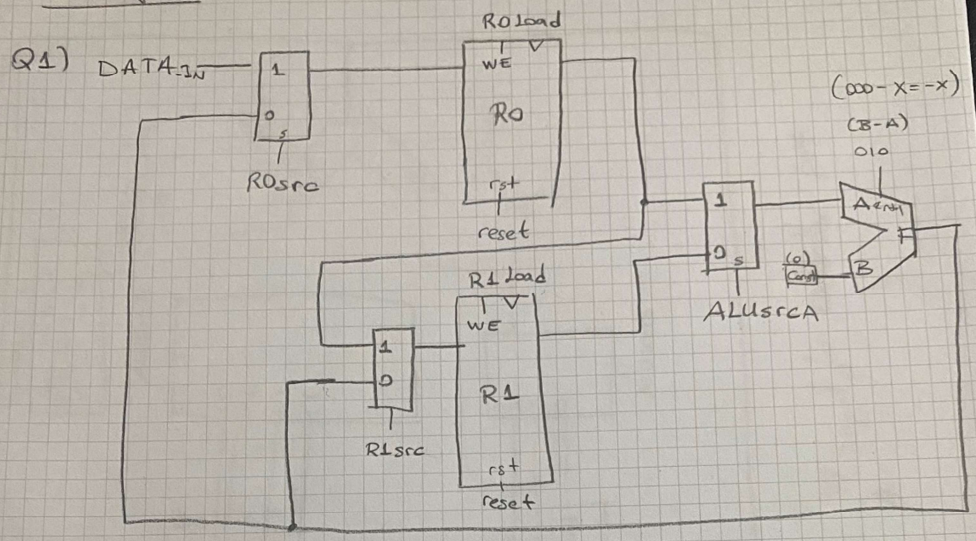
Berkay İPEK

EE446 -2304814

Laboratory Work 2 - Arithmetic Logic Processor Design

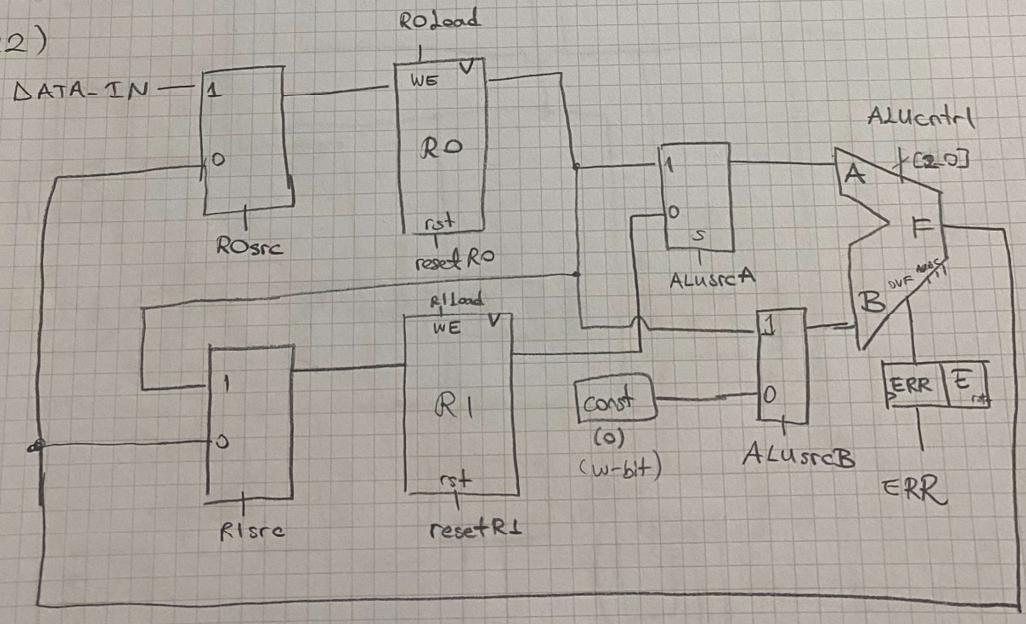
Datapath

Q1)

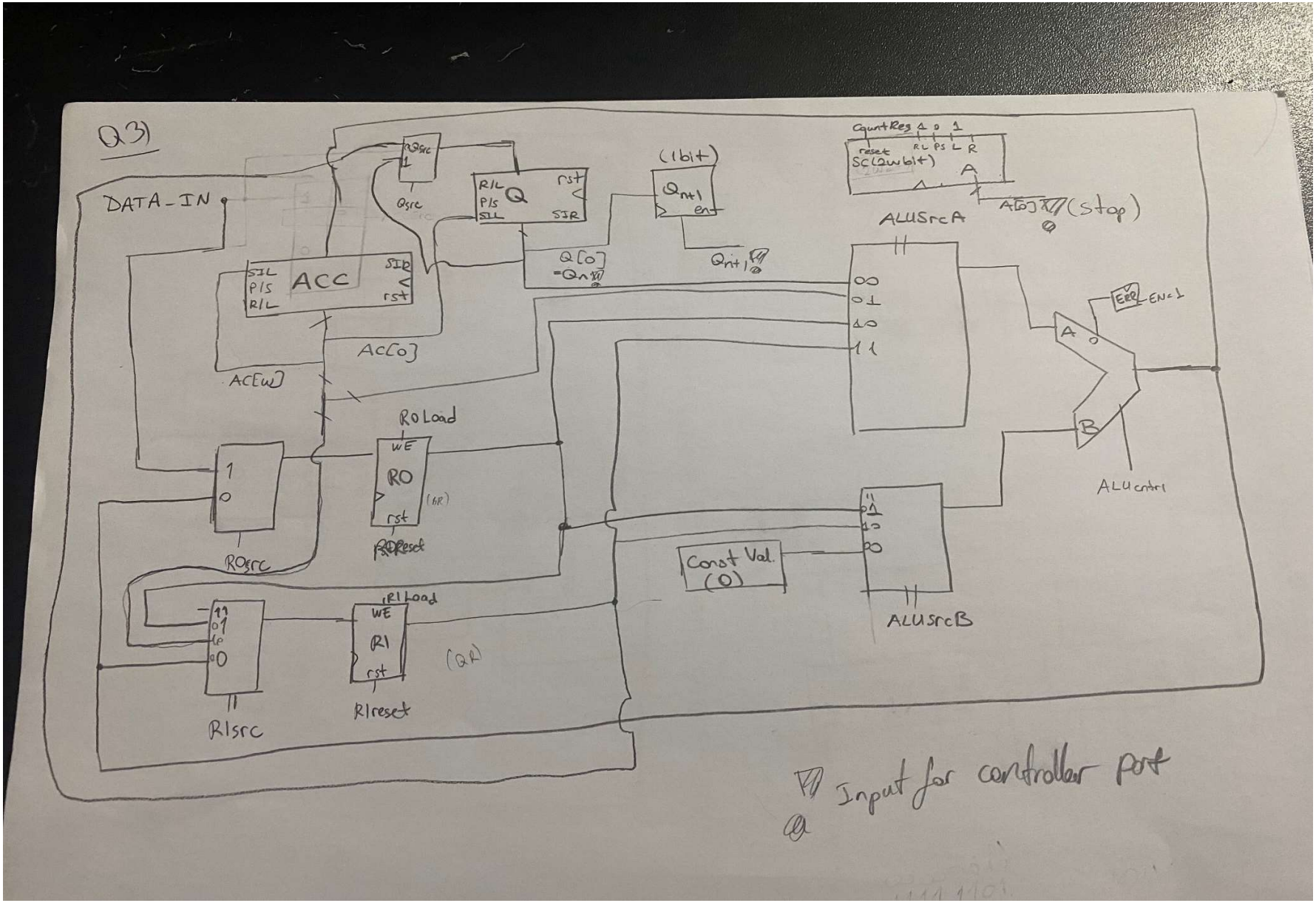


- ALU Cntrl == 010 Therefore it is not a control signal.

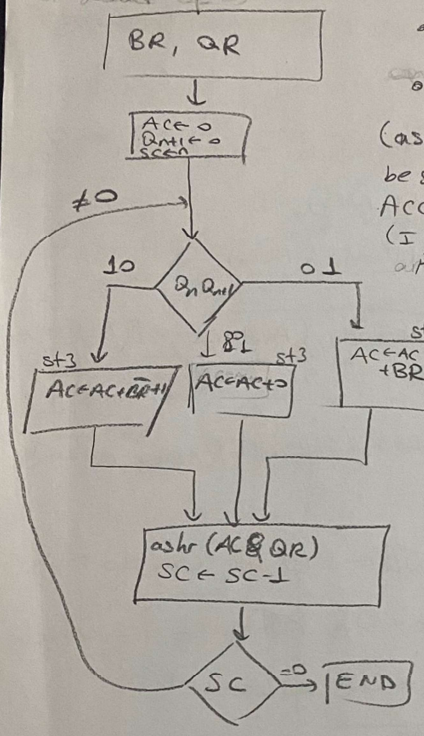
Q2)



- ALUcntrl, now, is a control signal. Since it will operate for different operations.

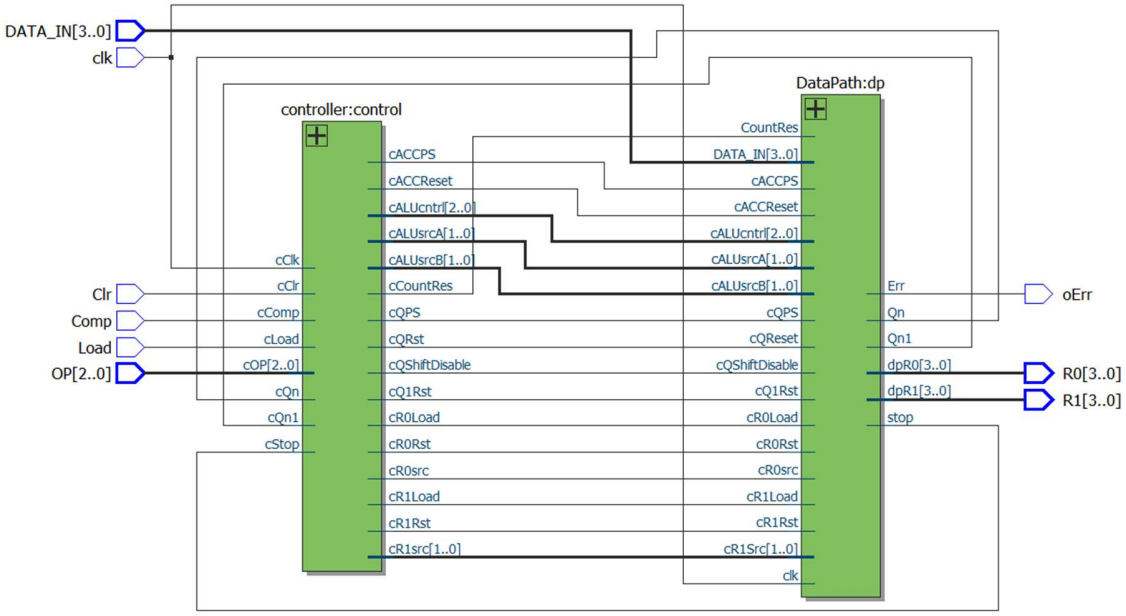

Q3)
DATA-IN
ACC
SIL
PIS
RIL
SIR
rst
Qsrc
RIL
PIS
SIL
Q
rst
SIR
(1bit)
Qnt1
ent
Q[0]
=Qnt1
Qnt1
QuntReg
reset
SC(2w bit)
RL PS L R
A
A[0]
(stop)
ALUSrcA
EER
EN=1
A
B
ALUcntrl
AC[0]
AC[w]
R0Load
WE
R0
rst
(IR)
R0reset
R0src
Const Val.
(0)
ALUSrcB
R1Load
WE
R1
rst
(QR)
R1reset
R1src
Input for controller port

- For Mult operation, I used (modified) booth's algorithm.



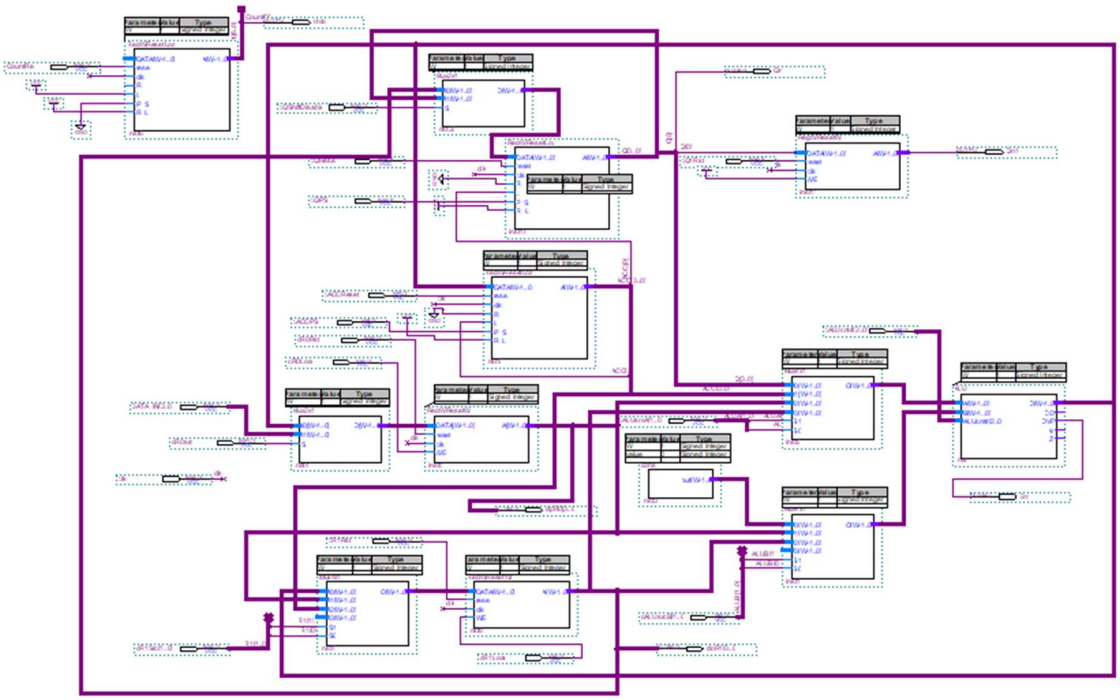
- I added AC ← AC + 0 state because
- After load operation, when MULL is desired (assume R0 = BR, R1 = QR). Qsrc will be selected as 0, so QR register will get QR. Acc will be reseted, and also SC will be cleared. (I used counter as shift register)
- Furthermore, $(\overline{BR}+1)$ will be calculated in this step. Since R0 has value of BR, just R1 ← 0000 - R0 operation will provide us loading R1 ← $\overline{BR}+1$
- According to Qn and Qn+1 input, controller will decide ALUsrc B. (To add BR, choose R0 and to add $\overline{BR}+1$, choose R1.) (If nothing then ALUsrcB will select 0) Then, do add operation and store it to ACC. While doing this step, Q should remain its content. Therefore, Qsrc will be 1 (which mean load itself.)
- To do ashr, AC will be shifted to right, (shift input is its signed bit) the last bit of ACC is the input of Qn. Qn+1
- SC => I used shift register, at the beginning, it is reseted, then "1" bit will be inserted from left input. This shift operation will be done in any cycle. Therefore, for each bit size, shifting will be done twice. That means, SC should 2w bit size so that we can understand operation is done. (in original, SC = 0)
- when everything is finished, R1 ← ACC & QR ← Q + 0 operation will be done

Q5)

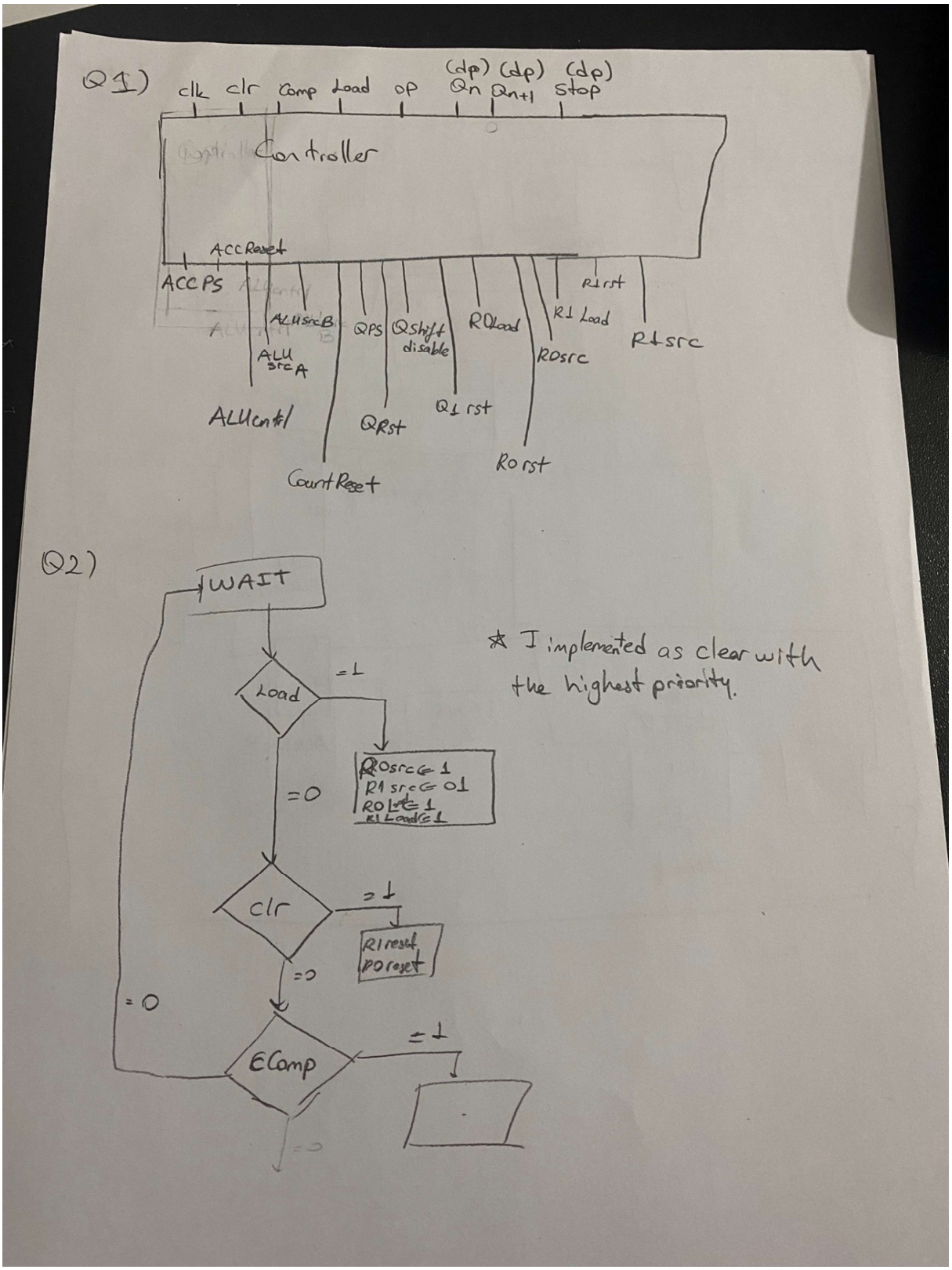


Q6)

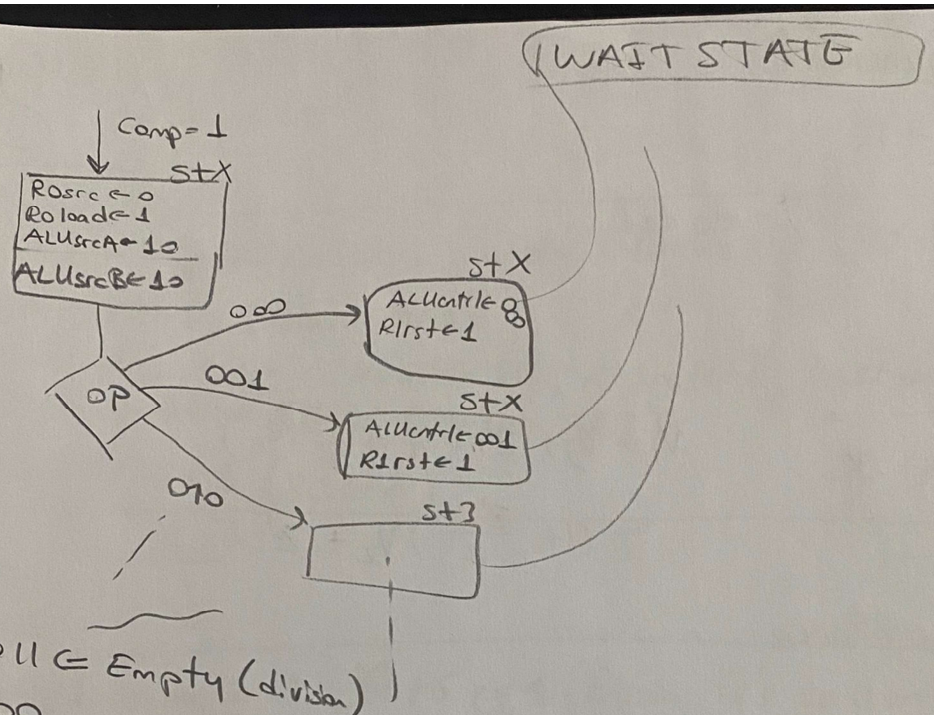
See bdf file in code zip also.



Controller Design



Q3)



011 ∈ Empty (division)

100

ALUcntrl ∈ 100
RIrst ← 1

101

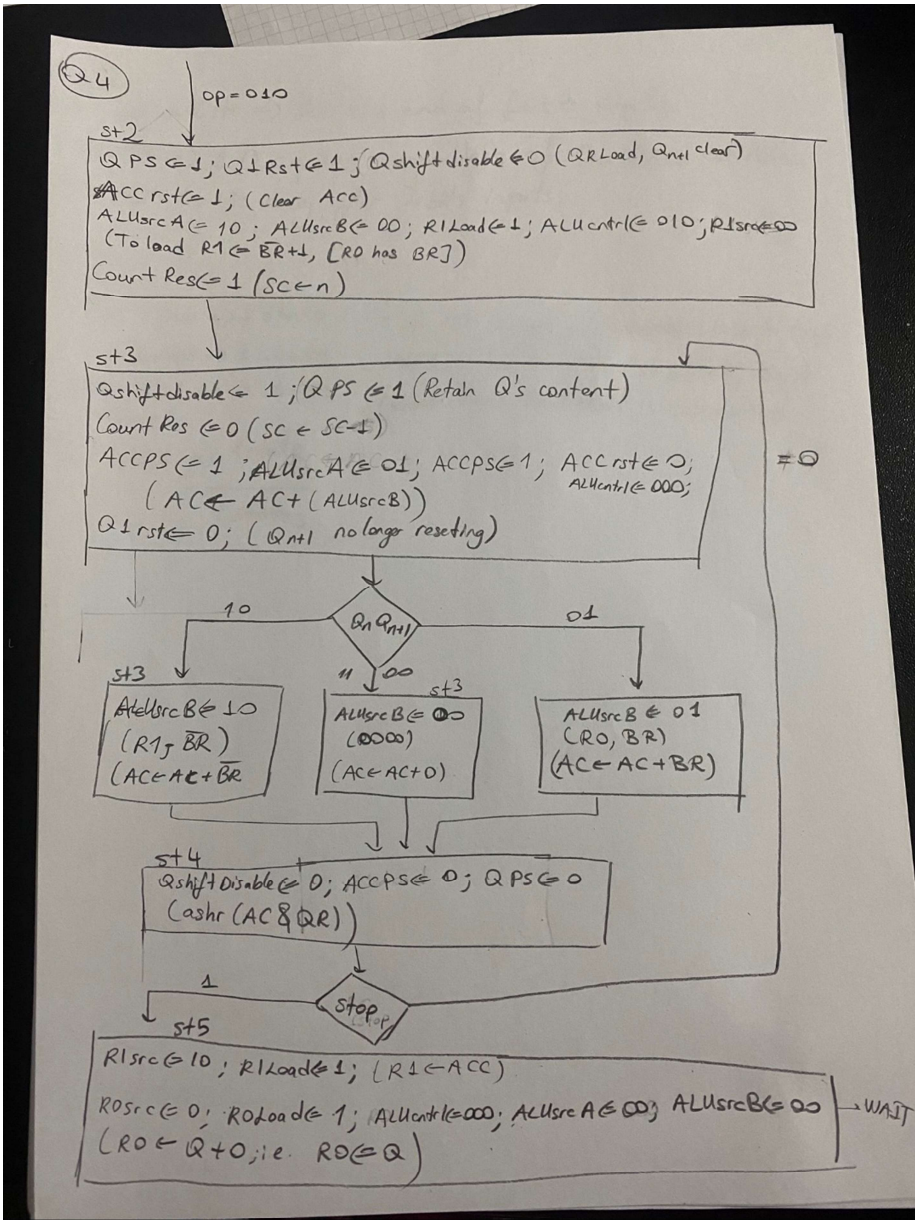
ALUcntrl ∈ 101
RIrst ∈ 1

110

ALUcntrl ∈ 110
RIrst ∈ 1

111

ALUcntrl ∈ 011
RIrst ∈ 1


Q4
op=010
st2
QPS ⟵ 1; Q1Rst ⟵ 1; Qshiftdisable ⟵ 0 (QRLoad, Qn+1 clear)
ACC rst ⟵ 1; (Clear Acc)
ALUsrcA ⟵ 10; ALUsrcB ⟵ 00; R1Load ⟵ 1; ALUcntrl ⟵ 010; R1src ⟵ 00
(To load R1 ⟵ BR+1, [R0 has BR])
Count Res ⟵ 1 (SC ⟵ n)
st3
Qshiftdisable ⟵ 1; QPS ⟵ 1 (Retain Q's content)
Count Res ⟵ 0 (SC ⟵ SC-1)
ACCPS ⟵ 1; ALUsrcA ⟵ 01; ACCPS ⟵ 1; ACC rst ⟵ 0; ALUcntrl ⟵ 000;
(AC ⟵ AC+(ALUsrcB))
Q1 rst ⟵ 0; (Qn+1 no longer resetting)
≠0
QnQn+1
10
01
11
00
st3
ALUsrcB ⟵ 10
(R1, BR)
(AC ⟵ AC+BR
st3
ALUsrcB ⟵ 00
(0000)
(AC ⟵ AC+0)
ALUsrcB ⟵ 01
(R0, BR)
(AC ⟵ AC+BR)
st4
Qshiftdisable ⟵ 0; ACCPS ⟵ 0; QPS ⟵ 0
(ashr (AC & QR))
1
stop
st5
R1src ⟵ 10; R1Load ⟵ 1; (R1 ⟵ ACC)
R0src ⟵ 0; R0Load ⟵ 1; ALUcntrl ⟵ 000; ALUsrcA ⟵ 00; ALUsrcB ⟵ 00
(R0 ⟵ Q+0; i.e. R0 ⟵ Q)
WAIT

Q6)

- Except Multi => At the end of first cycle.
- Multi => 10 cycles regardless of Qn Qn+1 conditions.

However, it is only for 4 bits inputs.

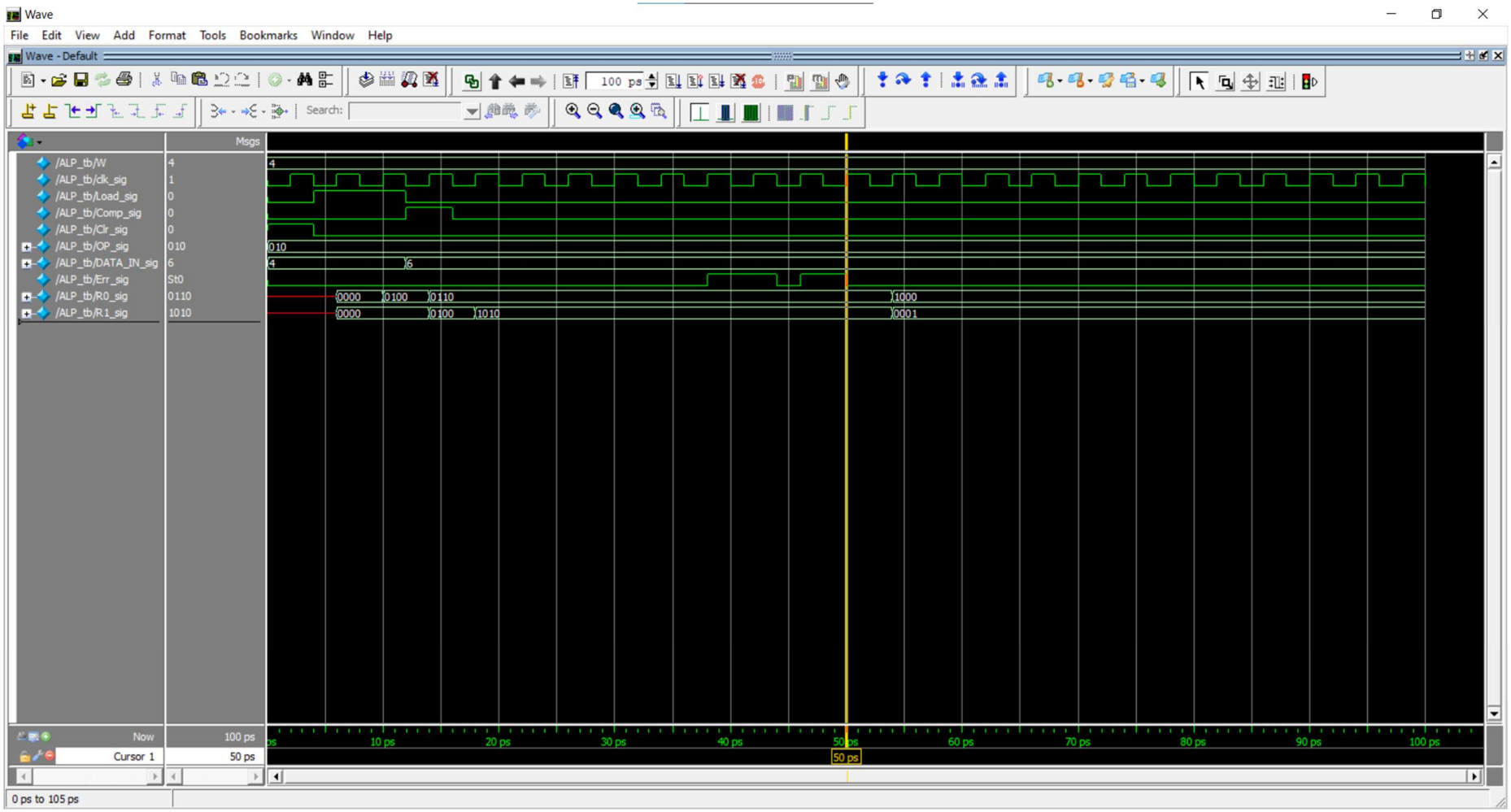
Q7)

- For clear => 1 state
- For Load => 1 state
- For MULT => 4 states (Comp)
- For wait => 1 state

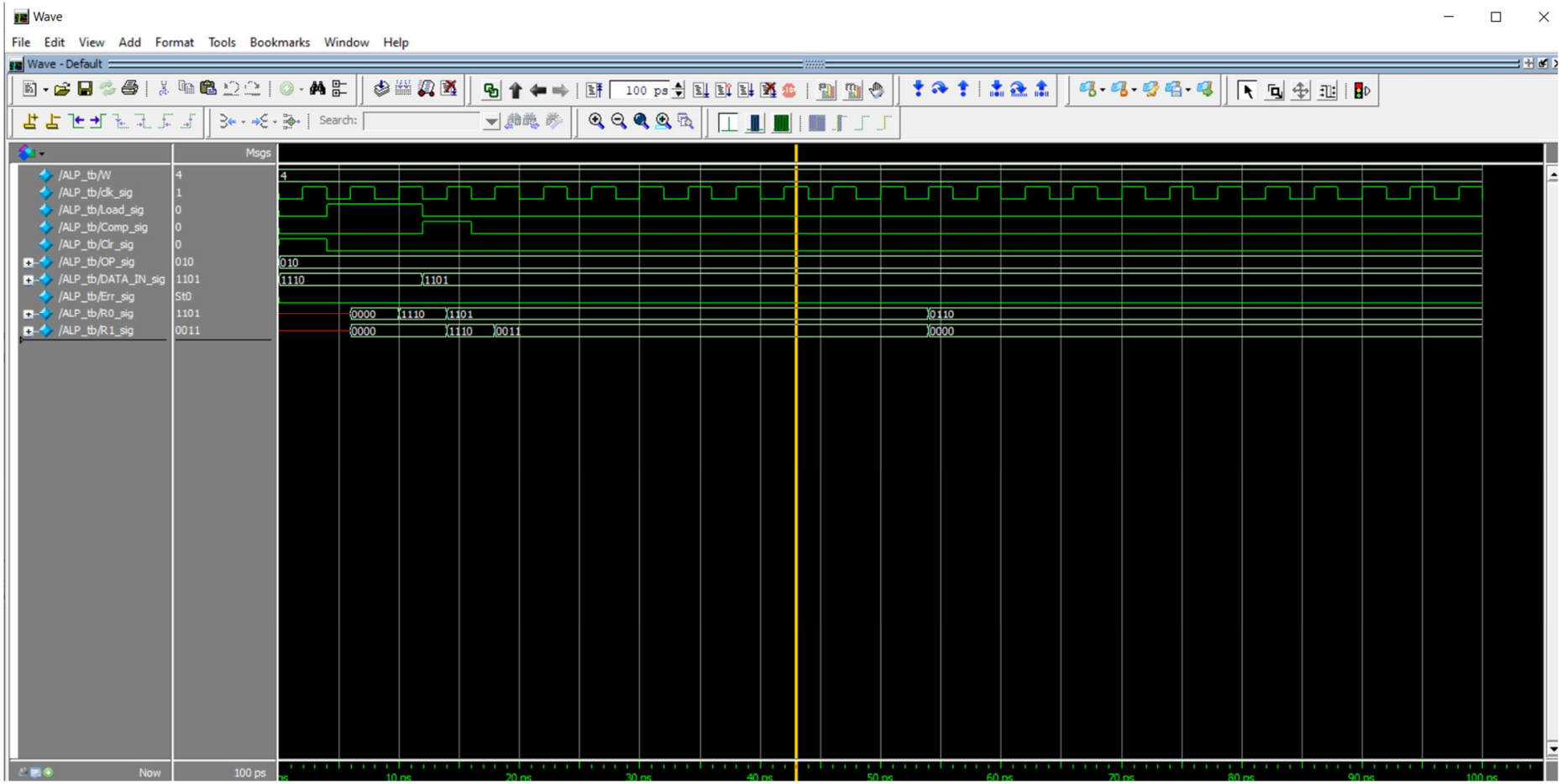
7 states

- If OP changed, my design would remain same, as if it enteres "s+3", "s+4", it will remain there until result is calculated.

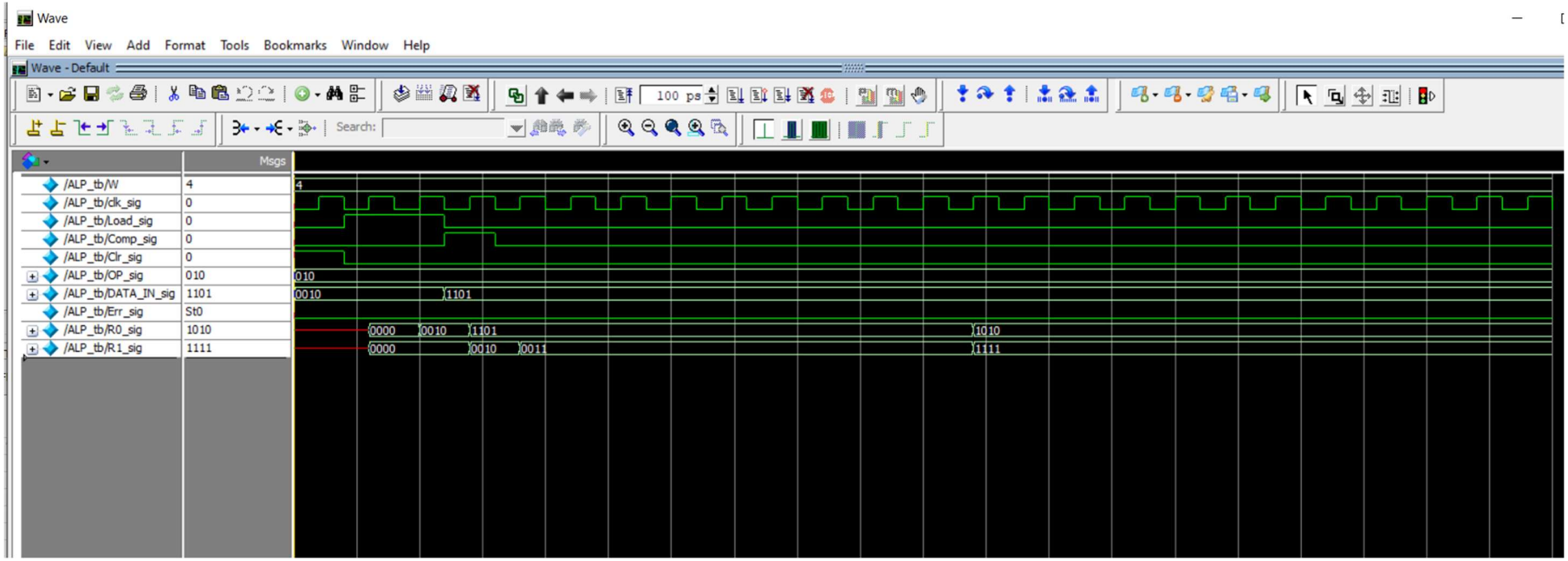
SIMULATION RESULT



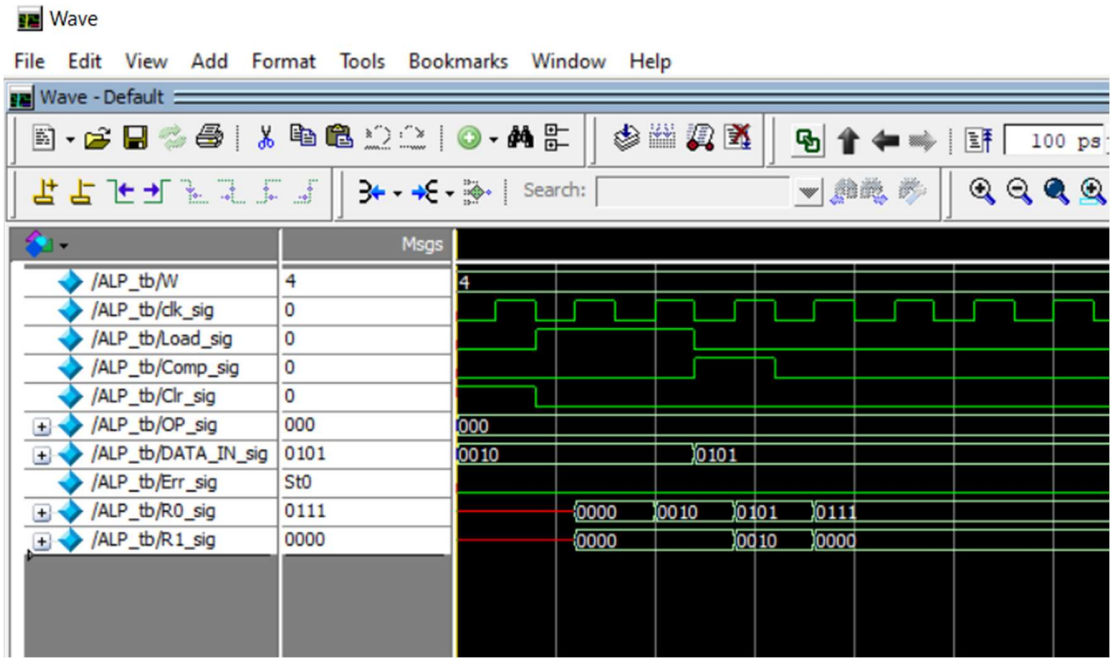
After multiplying 4 and 6, Result is in {R1,R0} = 00011000 = $(24)_{10}$



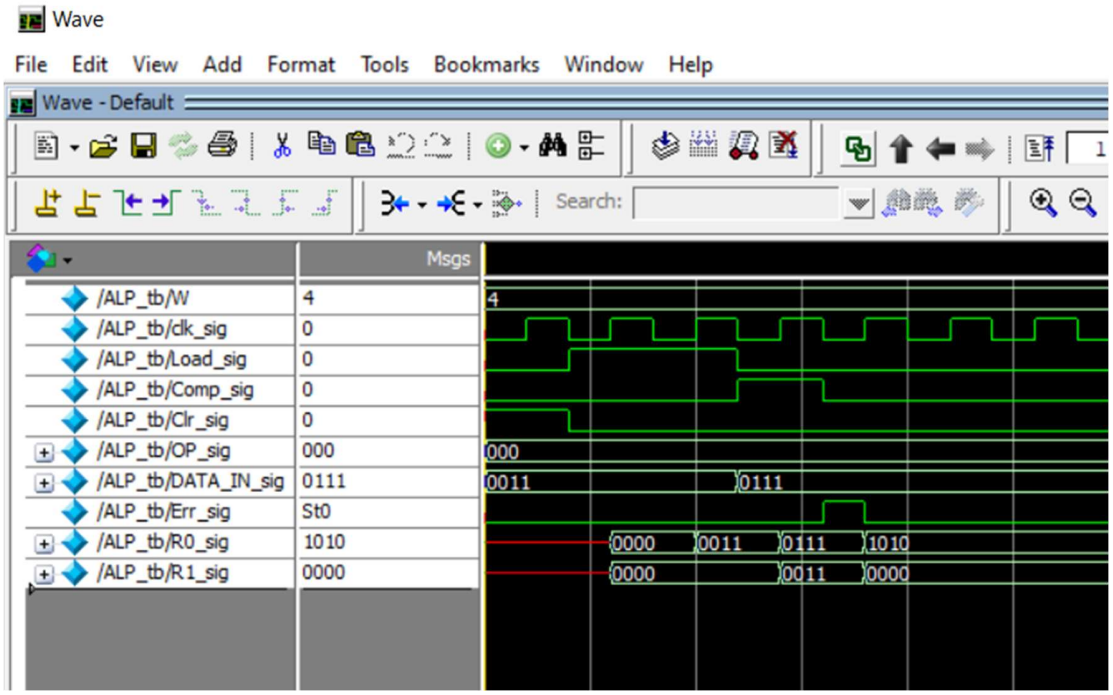
After multiplying -3 (1101) and -2(1110), Result is in {R1,R0} = 00000110 = $(6)_{10}$



After multiplying -3 (1101) and 2(0010), Result is in {R1,R0} = 11111010 = $(-6)_{10}$



After adding 2 (0010) and 5(0101), Result is in R0 = 0111 = $(7)_{10}$



After adding 3 (0011) and 7(0111), Result is in R0 =1010 = $(-6)_{10}$ with an ERROR SIGNAL.