Ho Chi Minh City National University
University of Information Technology
Computer Engineering

REPORT

DIGITAL LOGIC DESIGN

LogicCore-9: A Verilog-Based Finite State Machine for Sequential Arithmetic and Bitwise Operations on Signed Operands

Class: CE118.N22.MTCL-EN

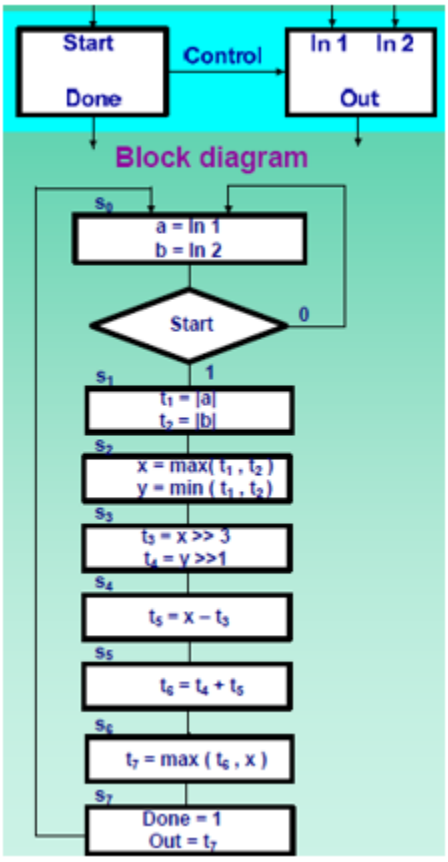
Lecturers: Hồ Ngọc Diễm

Student: Trương Thiên Quý - 23521321

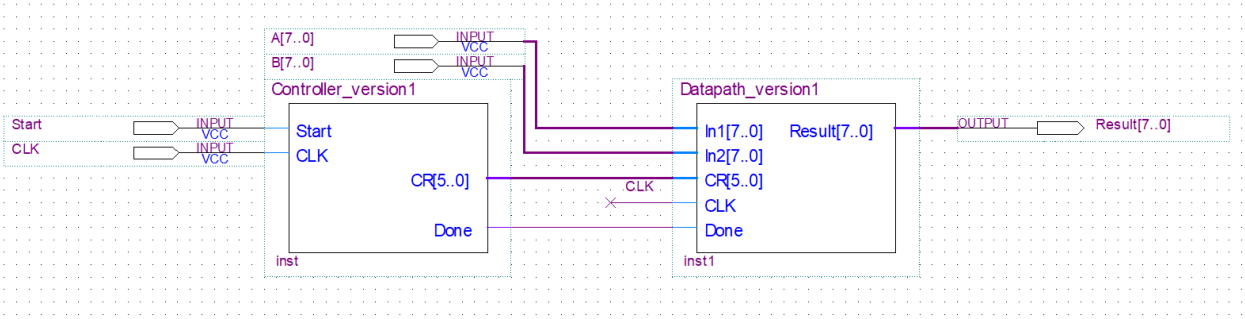
Ho Chi Minh City, 05/2024

Content

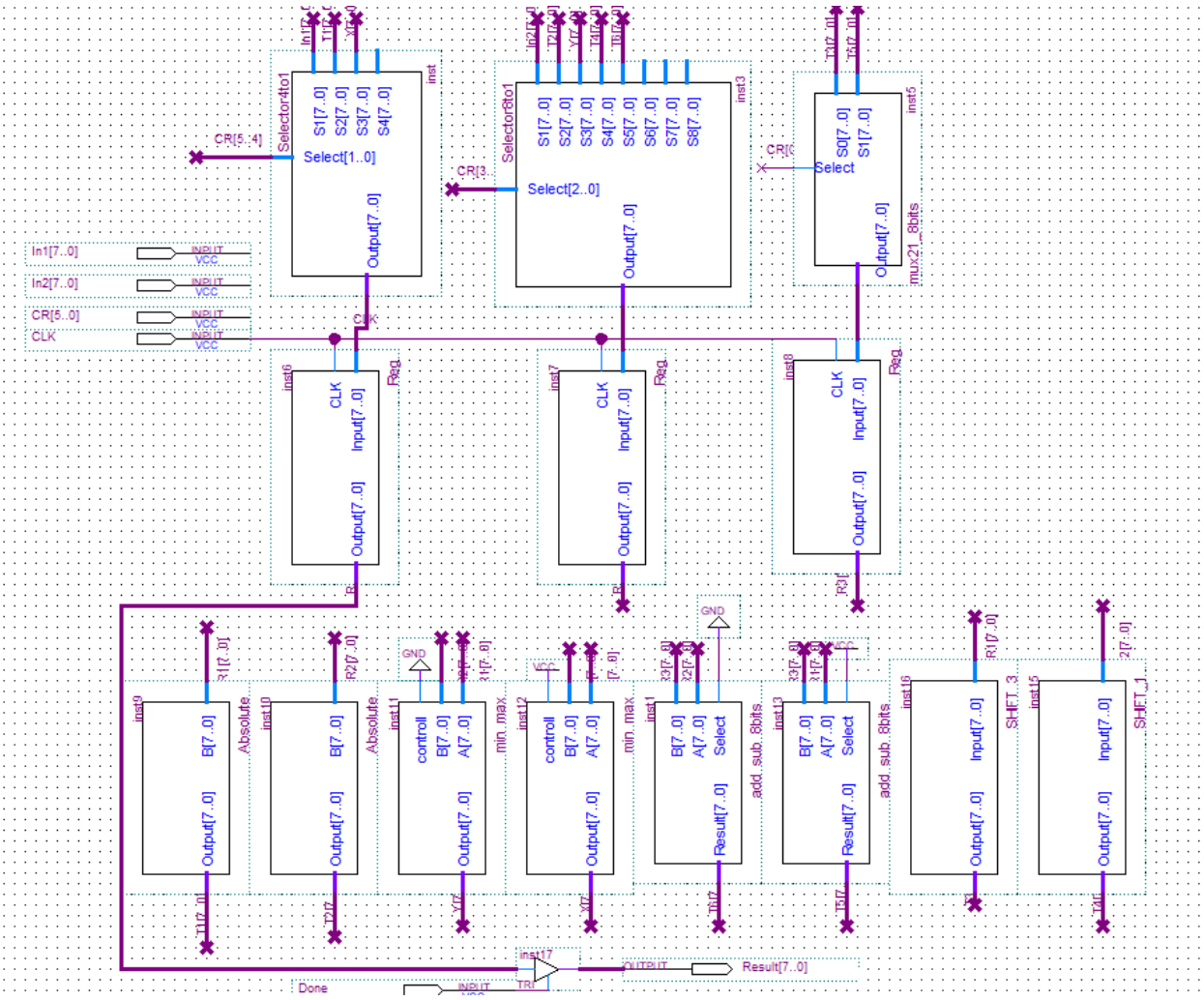
I. Version 1:



Circuit (includes controller and datapath):



1. Datapath:

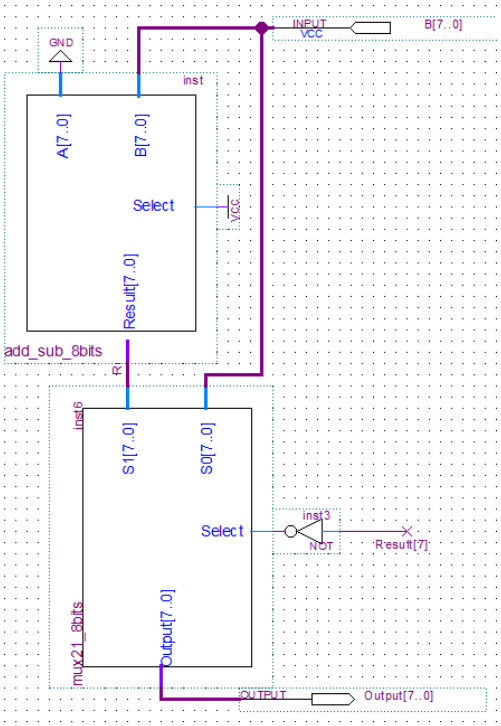


- Variable groups:
  - R1 = {In1, T1, X}
  - R2 = {In2, T2, Y, T4, T6}
  - R3 = {T3, T5}

Each group connect with 1 register

- **Absolute block (ABS): Use 1 Abs/Sub block and 1 Mux 2-1**

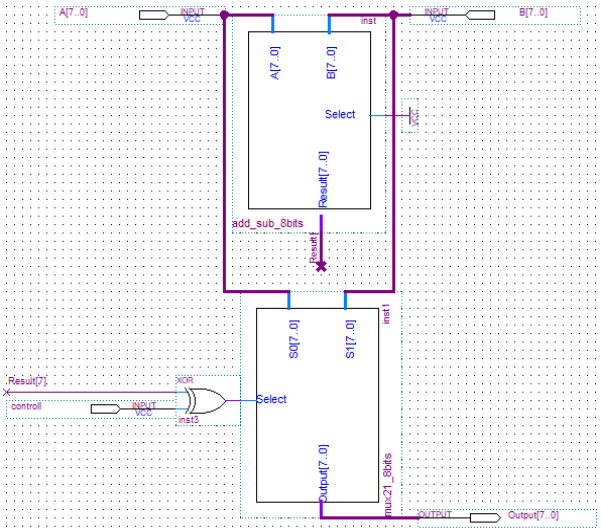
- Calculate 0 – number.
- If sign bit of result = 0, Output = this result.
- If sign bit of result = 1, Output = this number



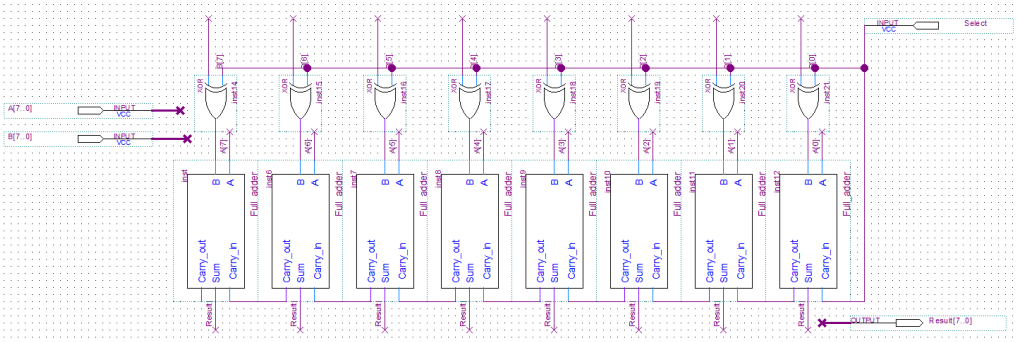
- **Min/Max Block: Use 1 Sub block and 1 Mux 2-1**

If Control = 0 output = min(A,B)

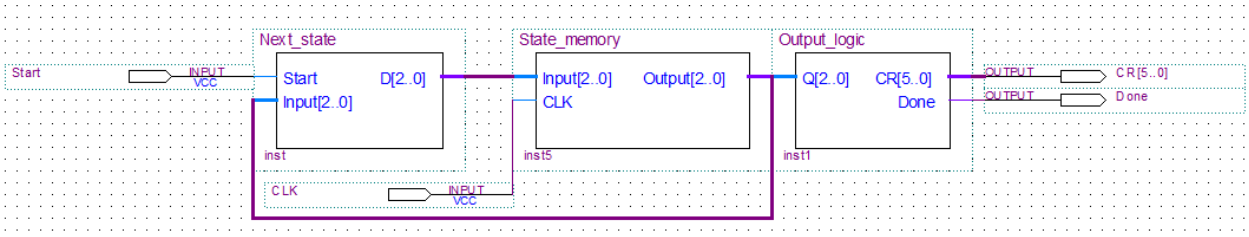
If Control =1, output = max(A,B)



- **Add/sub: If select = 0 : Calculate A+B ; Else Calculate A-B**



2. Controller:



Controller Block

- CR[5..4]: Control input of R1
- CR[3..1]: Control input of R2
- CR[0]: Control input of R3
- Done: Output in last state

- **Identify Next state:**

Q			Start	Q+		
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	0
0	1	0	X	0	1	1
0	1	1	X	1	0	0
1	0	0	X	1	0	1
1	0	1	X	1	1	0
1	1	0	X	1	1	1
1	1	1	X	0	0	0

Bảng xác định tín hiệu next state

- When start equal 1 ,jump to program.
- In state 7, get output.

Use K-Map:

Q0+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	1	0	0
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	1	1	0	0

$$Q_0^+ = \overline{Q_0}S + Q_1\overline{Q_0} + Q_2\overline{Q_0}$$

Q1+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	1	1
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	0	0	1	1

$$Q_1^+ = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

Q2+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	0	0
	0 1	0	0	1	1
	1 1	1	1	0	0
	1 0	1	1	1	1

$$Q_2^+ = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$

- Table of identify controll in each state:

State	CR[5..4] (R1)		CR[3..1] (R2)			CR[0] (R3)	Done
S0	0	0	0	0	0	x	0
S1	0	1	0	0	1	X	0
S2	1	0	0	1	0	x	0
S3	1	0	0	1	0	0	0
S4	1	0	0	1	1	1	0
S5	1	0	1	0	0	x	0
S6	1	0	x	x	x	x	0
S7	x	x	x	x	x	x	1

CR[0]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	0	0
	1	1	0	0	0

$$CR[0] = Q_2\overline{Q_1}\,\overline{Q_0}$$

CR[1]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	0	0
	1	1	0	0	0

$$CR[1] = Q_2\overline{Q_1}\,\overline{Q_0} + \overline{Q_2}\,\overline{Q_1}\,Q_0$$

CR[2]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	1
	1	1	0	1	1

$$CR[2] = Q_1 + \overline{Q_0}\,Q_2$$

CR[3]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	0	0
	1	0	1	1	0

$$CR[3] = Q_2\,Q_0$$

CR[4]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	0	0
	1	0	0	0	0

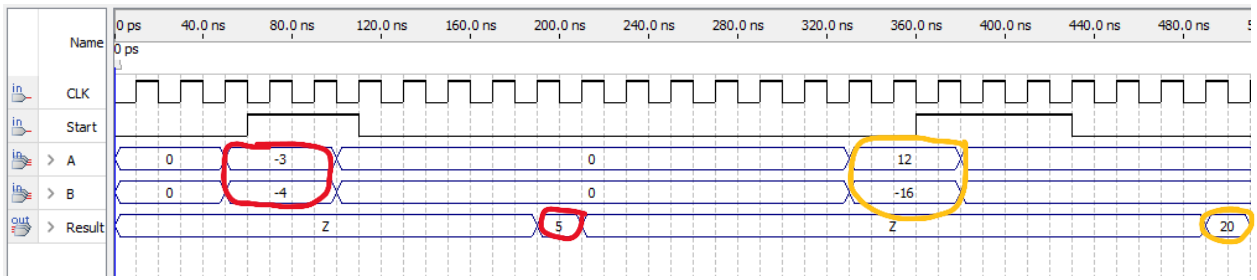
$$CR[4] = \overline{Q_2}\,\overline{Q_1}\,Q_0$$

CR[5]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	1
	1	1	1	1	x

$$CR[5] = Q_2 + Q_1$$

Done = Q2 Q1 Q0

3. Waveform:

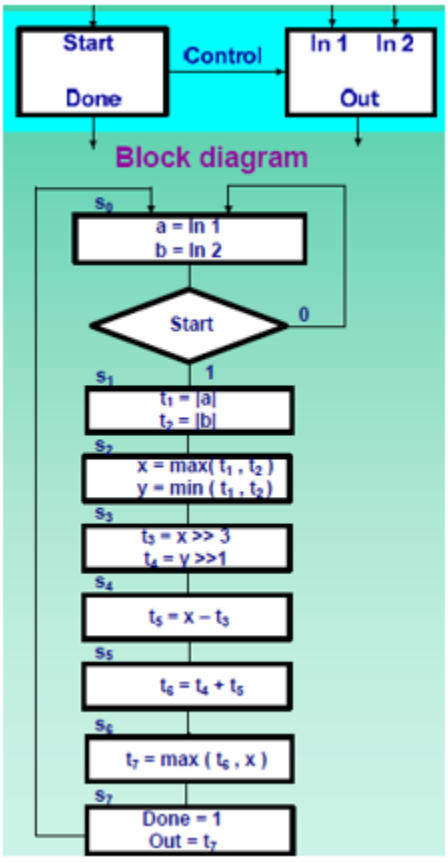


Every 7 cycles after the start signal turns on, output the result (S7), and after S7 reset back to S0.

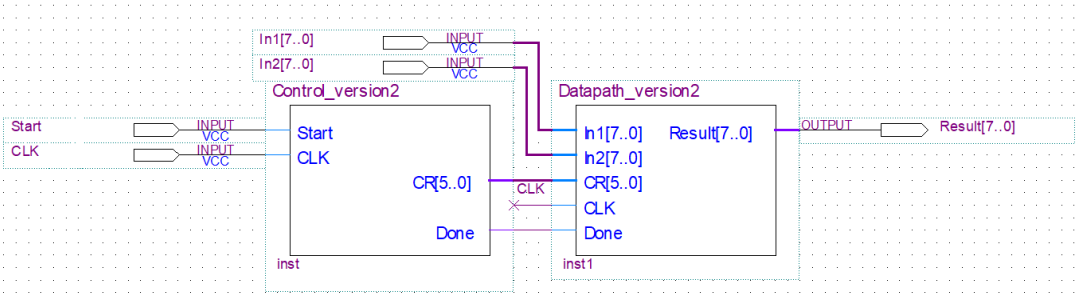
Test:

Input1 = -3 and Input2 = -4	Input1= 12 and Input2 = -16
S0: a = -3 , b = -4	S0: a = 12 , b = -16
S1: t1 = $\|a\|$ = 3, t2 = $\|b\|$ = 4	S1: t1 = $\|a\|$ = 12, t2 = $\|b\|$ = 16
S2: x = max(t1, t2)= 4, y = min(t1, t2)= 3	S2: x = max(t1, t2)= 16, y = min(t1, t2)= 12
S3: t3 = x >>3 = 0, t4 = y>>1 = 1	S3: t3 = x >>3 = 2, t4 = y>>1 = 6
S4: t5 = x – t3 = 4	S4: t5 = x – t3 = 4
S5: t6 = t5+ t4 = 5	S5: t6 = t5+ t4 = 20
S6: t7= max(t6, x) = 5	S6: t7= max(t6, x) = 20
S7: result = t7 = 5	S7: result = t7 = 20

II. Version 2

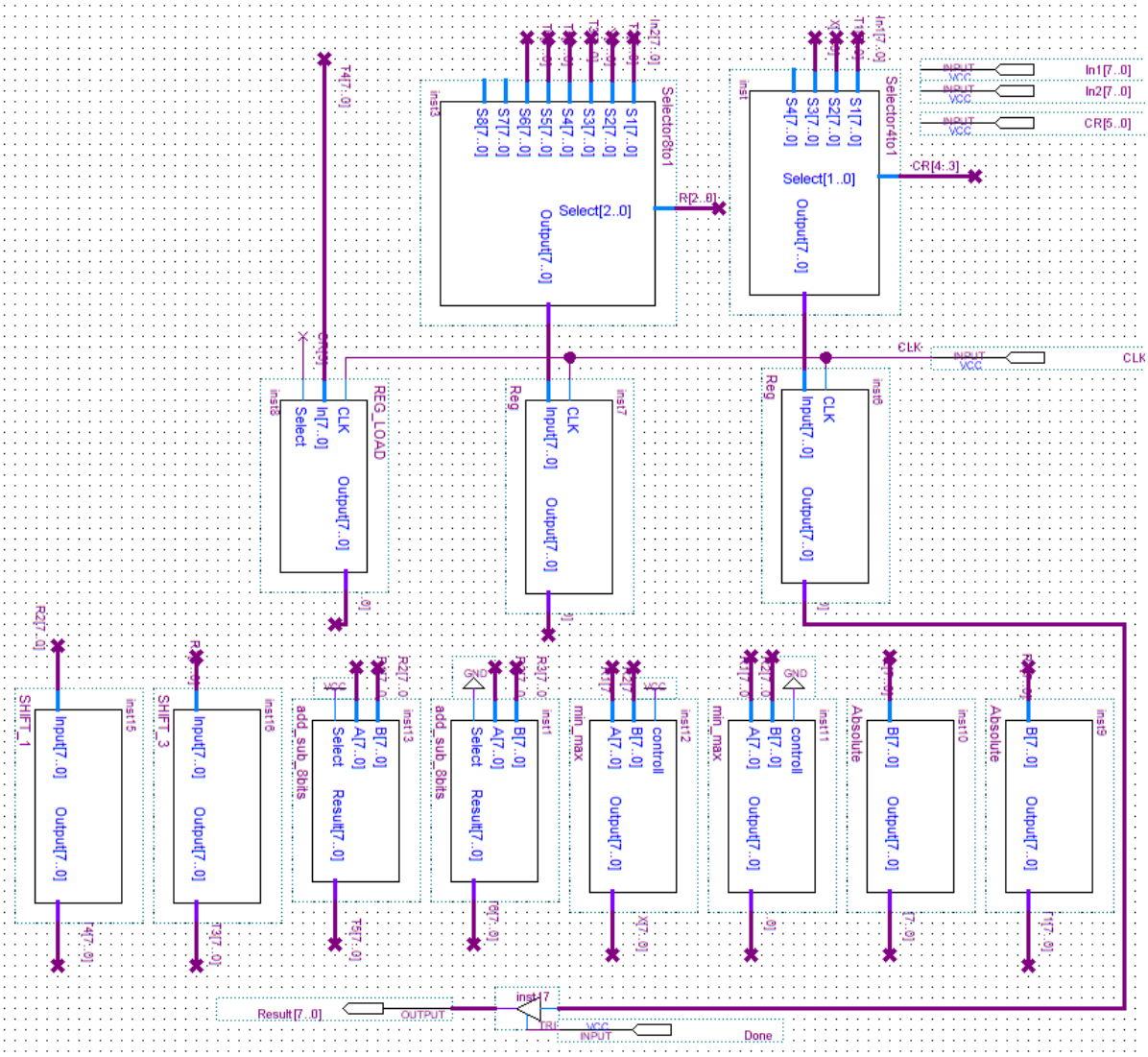


Circuit (includes controller and datapath):



version 2 is almost the same as version 1, but different in the group of variables

1. Datapath

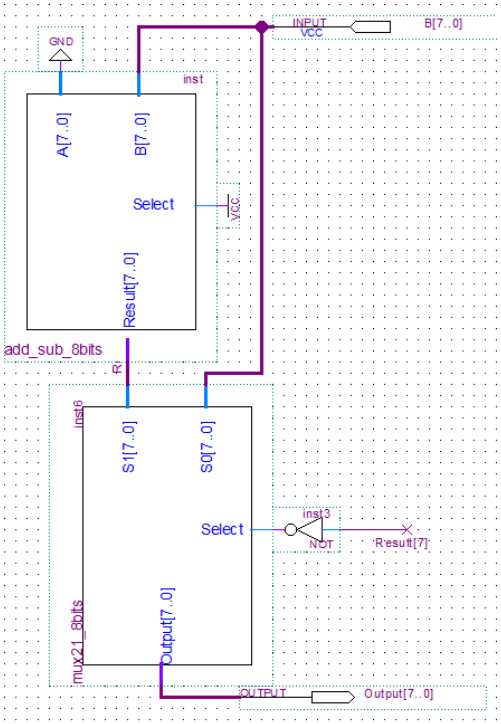


- Variable groups:
  - R1 = {In1, T1, X, T7}
  - R2 = {In2, T2, Y, T3, T5, T6}
  - R3 = {T4}

Each group connect with 1 register

- **Absolute block (ABS): Use 1 Abs/Sub block and 1 Mux 2-1**

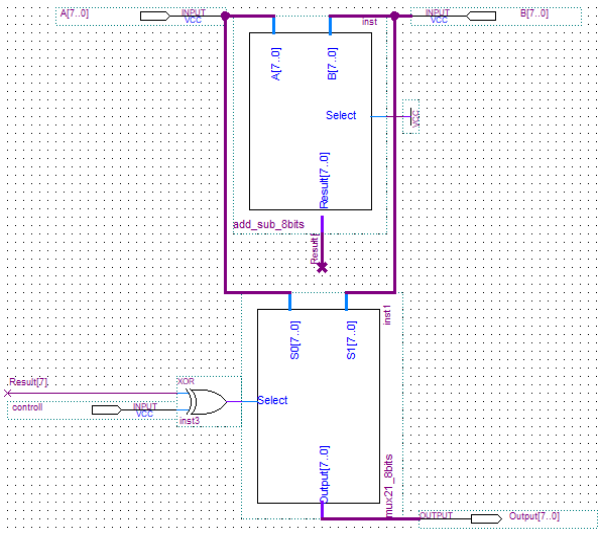
- Calculate 0 – number.
- If sign bit of result = 0, Output = this result.
- If sign bit of result = 1, Output = this number



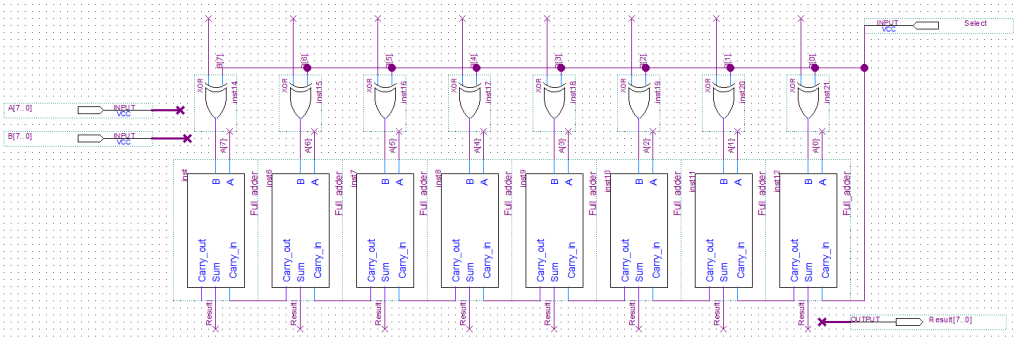
- **Min/Max Block: Use 1 Sub block and 1 Mux 2-1**

If Control = 0 output = min(A,B)

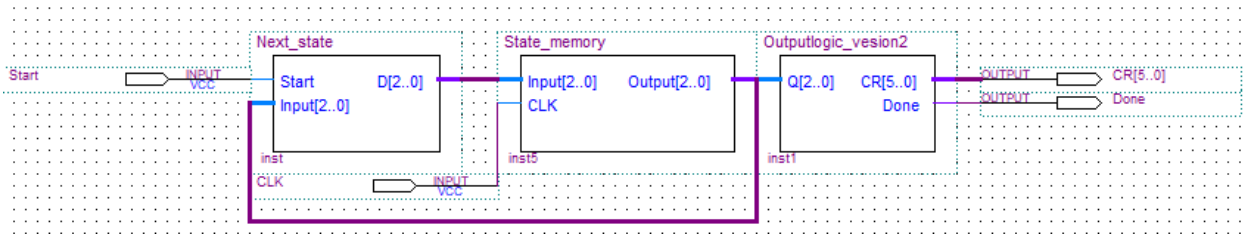
If Control =1, output = max(A,B)



- Add/sub: If select = 0 : Calculate A+B ; Else Calculate A-B



2. Controller



Controller Block

Controller of version2 differs from version1 in datapath block control signal

- CR[5]: Write Enable of R3
- CR[4..3]: Control input of R1
- CR[2..0]: Control input of R2
- Done: Output in last state

- **Identify Next state:**

Q			Start	Q+		
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	0
0	1	0	X	0	1	1
0	1	1	X	1	0	0
1	0	0	X	1	0	1
1	0	1	X	1	1	0
1	1	0	X	1	1	1
1	1	1	X	0	0	0

Bảng xác định tín hiệu next state

- When start equal 1 ,jump to program.
- In state 7, get output.

Use K-Map:

Q0+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	1	0	0
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	1	1	0	0

$$Q_0^+ = \overline{Q_0}S + Q_1\overline{Q_0} + Q_2\overline{Q_0}$$

Q1+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	1	1
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	0	0	1	1

$$Q_1^+ = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

Q2+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	0	0
	0 1	0	0	1	1
	1 1	1	1	0	0
	1 0	1	1	1	1

$$Q_2^+ = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$

- Table of identify controll in each state:

State	CR[4..3] (R1)		CR[2..0] (R2)			CR[5] (R3)	Done
S0	0	0	0	0	0	0	0
S1	0	1	0	0	1	0	0
S2	1	0	0	1	0	0	0
S3	1	0	0	1	1	0	0
S4	1	0	1	0	0	1	0
S5	1	0	1	0	1	0	0
S6	1	0	x	x	x	0	0
S7	x	x	x	x	x	0	1

CR[0]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	1	0
	1	0	1	x	x

$CR[0] = Q_0$

CR[1]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	1
	1	0	0	x	x

$CR[1] = Q_1$

CR[2]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	1
	1	0	0	x	x

$CR[2] = Q_2$

CR[3]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	0	0
	1	0	0	x	x

$CR[3] = \overline{Q_2}Q_0\overline{Q_1}$

CR[4]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	1
	1	1	1	x	x

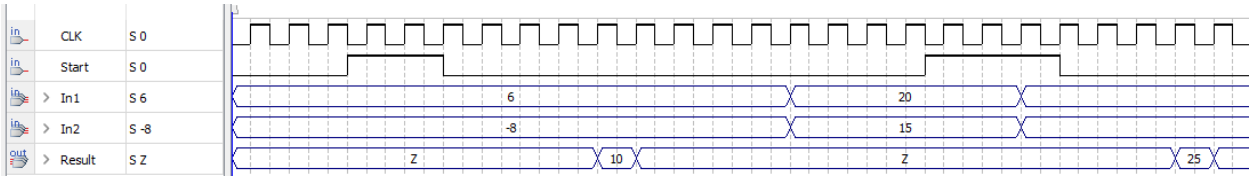
$CR[4] = Q_2 + Q_1$

CR[5]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	0	0
	1	1	0	x	x

$CR[5] = \overline{Q_0}Q_2\overline{Q_1}$

Done = Q2 Q1 Q0

3. Waveform

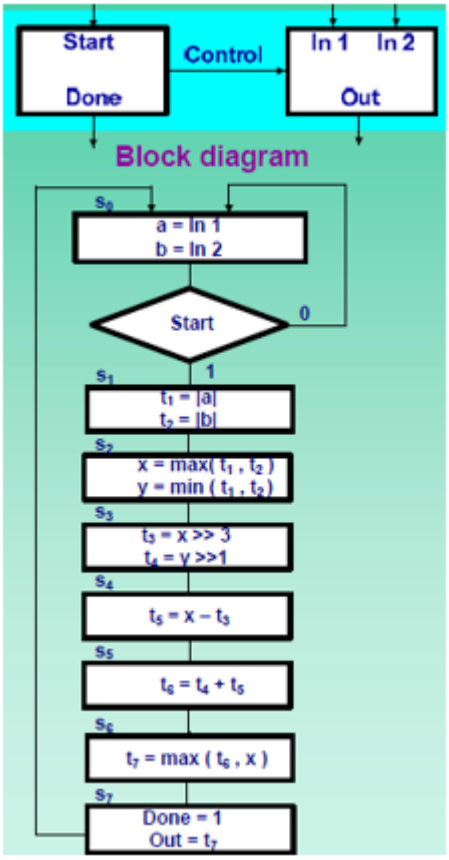


Every 7 cycles after the start signal turns on, output the result (S7), and after S7 reset back to S0.

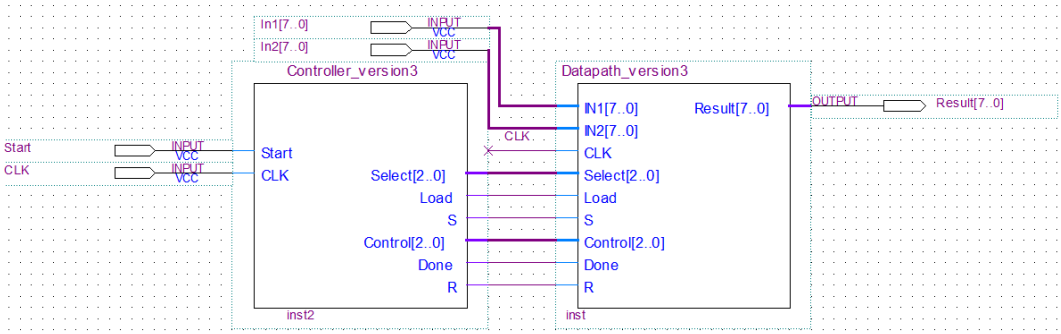
Test:

Input1 = 6 and Input2 = -8	Input1= 20 and Input2 = -15
S0: a = 6 , b = -8	S0: a = 20 , b = -15
S1: t1 = $\|a\|$ = 6, t2 = $\|b\|$ = 8	S1: t1 = $\|a\|$ = 20 (10100), t2 = $\|b\|$ = 15 (1111)
S2: x = max(t1, t2)= 8, y = min(t1, t2)= 6	S2: x = max(t1, t2)= 20, y = min(t1, t2)= 15
S3: t3 = x >>3 = 1, t4 = y>>1 = 3	S3: t3 = x >>3 = 2, t4 = y>>1 = 7
S4: t5 = x – t3 = 7	S4: t5 = x – t3 = 18
S5: t6 = t5+ t4 = 10	S5: t6 = t5+ t4 = 25
S6: t7= max(t6, x) = 10	S6: t7= max(t6, x) = 25
S7: result = t7 = 10	S7: result = t7 = 25

III. Version 3:

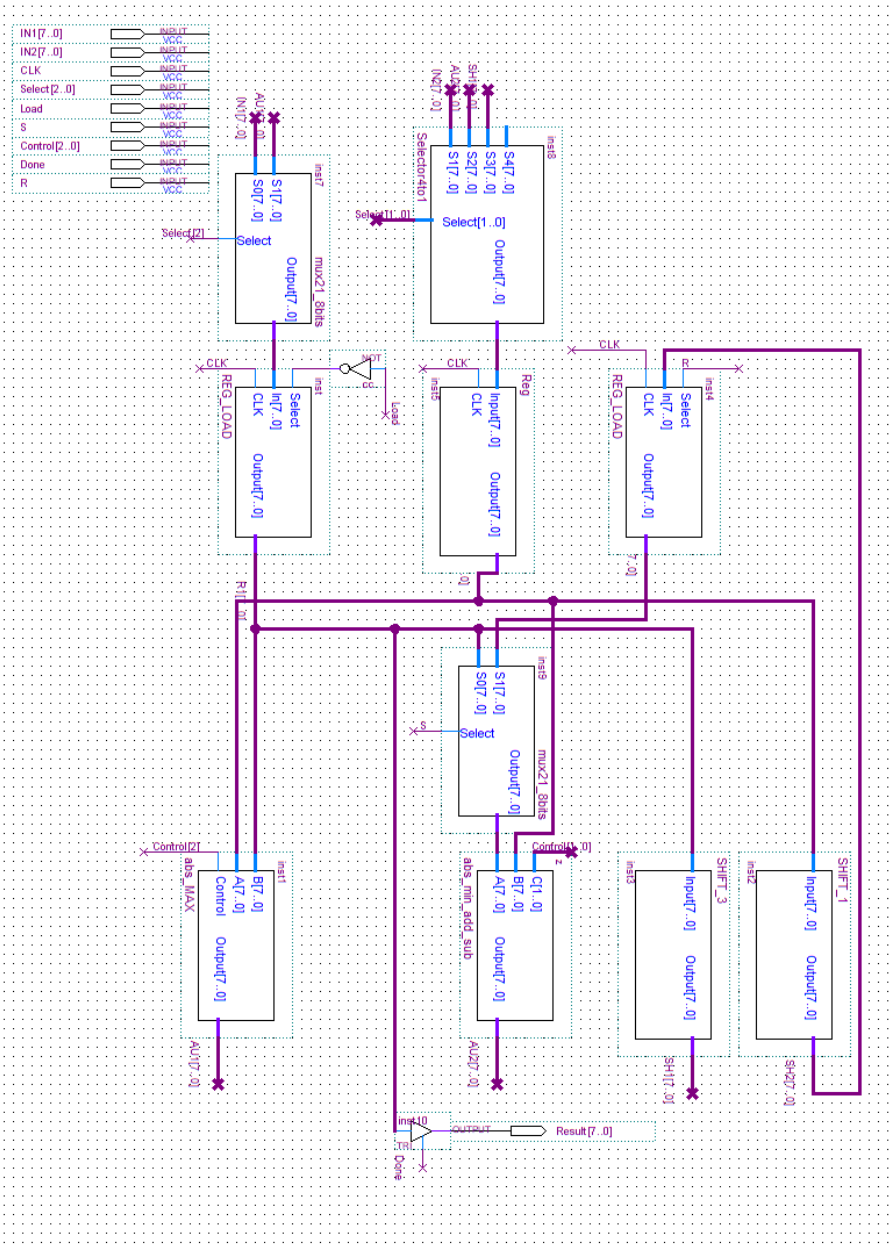


Circuit (includes controller and datapath):



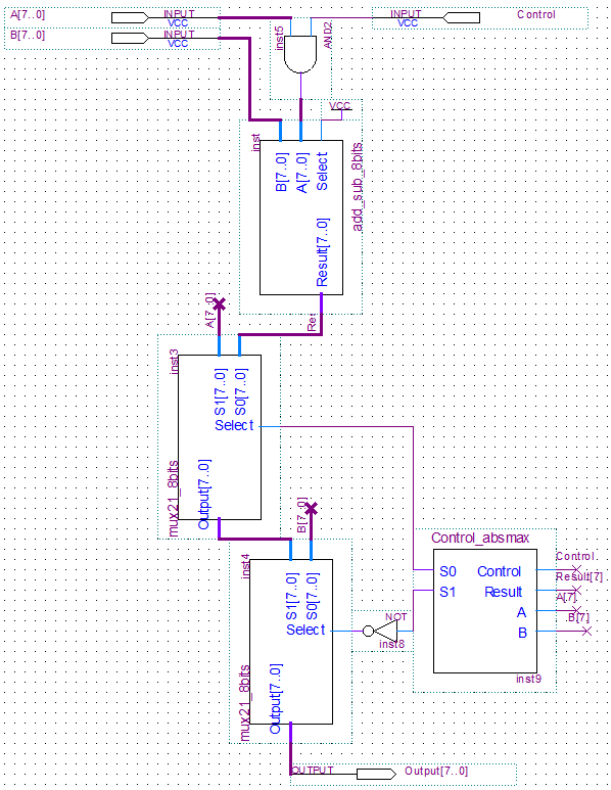
In version 3, We use “Functional logic sharing”, to reduce and reuse Functional logic.

1. Datapath



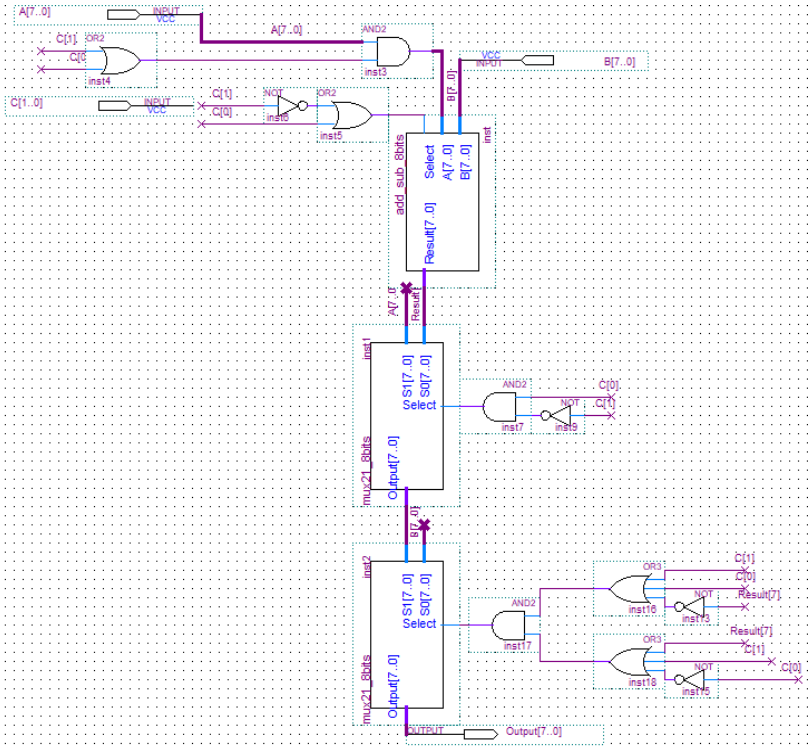
- Functional logic sharing:
  - ABS/Max
  - ABS/Min/+/-
  - Shift 3
  - Shift 1
- Groups:
  - R1: Input1, result of ABS/Max
  - R2: Input2, result of ABS/min/+/-, Shift 3
  - R3: Shift 1
- Use 1 mux 2-1 to select input for Block ABS/Min/+/-

- **ABS/max :**



- Use 1 Full adder/subtract, 2 Mux 2-1
- If Control = 0, Calculate ABS
    - If sign bit of result = 0, result = this Result
    - Else result = this number
- If Control = 1, Calculate Max
    - If sign bit of result = 0, result = A
    - Else result = B

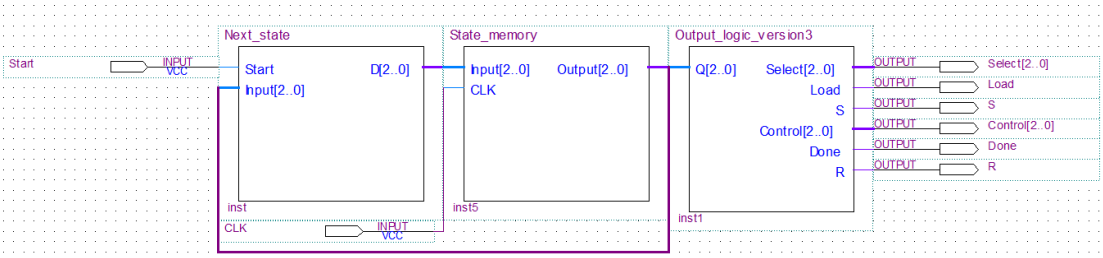
- **ABS/Min/+/-:**



- Use 1 Full adder/subtract, 2 Mux 2-1
- If Control = 00, Calculate ABS
    - If sign bit of result = 0, result = this Result

- Else result = this number
- If Control = 01, Calculate Max
  - If sign bit of result = 0, result = A
  - Else result = B
- If Control = 10, Calculate A+B
- If Control = 11, Calculate A-B

2. Controller



Controller Block

Controller of version2 differs from version1, version 2 in datapath block control signal

- Select[2]: Control input of R1
- Select[1..0]: Control input of R2
- Load: Write Enable of R1
- S: Select Input for Block ABS/Min/+/-
- Control[2]: Select function of ABS/Max
- Control[1..0] Select function of ABS/Min/+/-
- R: Write Enable of R3
- Done: Output in last state

- **Identify Next state:**

Q			Start	Q+		
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	0
0	1	0	X	0	1	1
0	1	1	X	1	0	0
1	0	0	X	1	0	1
1	0	1	X	1	1	0
1	1	0	X	1	1	1
1	1	1	X	0	0	0

Bảng xác định tín hiệu next state

- When start equal 1 ,jump to program.
- In state 7, get output.

Use K-Map:

Q0+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	1	0	0
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	1	1	0	0

$$Q_0^+ = \overline{Q_0}S + Q_1\overline{Q_0} + Q_2\overline{Q_0}$$

Q1+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	1	1
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	0	0	1	1

$$Q_1^+ = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

Q2+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	0	0
	0 1	0	0	1	1
	1 1	1	1	0	0
	1 0	1	1	1	1

$$Q_2^+ = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$

- Table of identify controll in each state:

State	Select[2..0]			S	Control[2..0]			Load	R	Done
	R1	R2			ABS/Max	ABS/Min/+/-				
S0	0	0	0	0	x	x	x	1	0	0
S1	1	0	1	0	0	0	0	1	0	0
S2	1	0	1	0	1	0	1	0	0	0
S3	1	1	0	0	x	x	x	0	0	0
S4	1	0	1	0	0	1	1	0	1	0
S5	1	0	1	1	0	1	0	1	0	0
S6	x	x	x	1	0	0	0	0	0	0
S7	x	x	x	1	x	x	x	x	0	1

Select[2]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	1	1
	1	1	1	1	1

Select [2] $= \overline{Q_2}Q_1\overline{Q_0}$

Select[1]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	1	0
	1	0	0	x	x

Select [1] $= Q_1Q_0$

Select[0]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	1	0	1
	1	1	1	x	x

Select [0]$= (Q_2 + Q_1 + Q_0)(Q_2 + \overline{Q_1} + \overline{Q_0})$

S		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	0	0	0	0
	1	0	1	1	1

S$= Q_2(Q_1 + Q_0)$

Control[2]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	x	0	x	1
	1	0	0	x	0

Control[2]$= Q_1\overline{Q_0}$

Control[1]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	x	0	x	0
	1	1	1	x	x

Control[1]$= Q_2$

Control[0]		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	x	0	0	1
	1	1	0	x	x

Control[0]$= Q_2\overline{Q_0} + Q_1\overline{Q_0}$

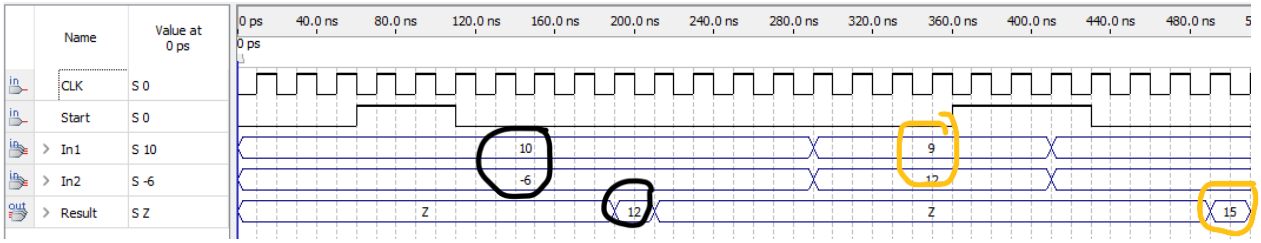
Load		Q1 Q0			
		0 0	0 1	1 1	1 0
Q2	0	1	1	0	0
	1	0	1	x	0

Load$= \bar{Q}_2\bar{Q}_1 + \bar{Q}_1Q_0$

R = $Q_2\bar{Q}_1\overline{Q_0}$

Done = Q2 Q1 Q0

3. Waveform



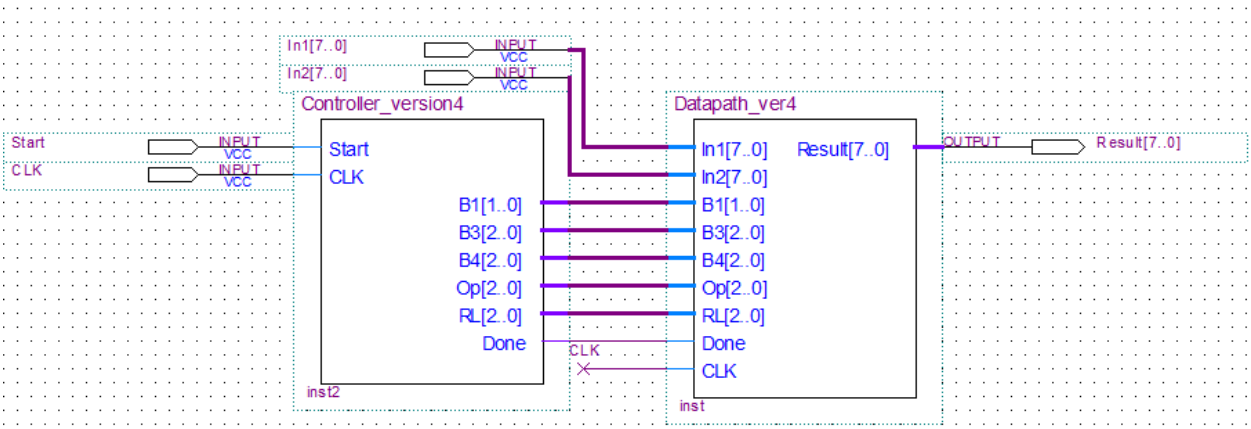
Every 7 cycles after the start signal turns on, output the result (S7), and after S7 reset back to S0 – same as Version1 and Version 2.

Test:

Input1 = 10 and Input2 = -6	Input1= 9 and Input2 = -12
S0: a = 10 , b = -6	S0: a = 9 , b = 12
S1: t1 = $\|a\|$ = 10(1010), t2 = $\|b\|$ = 6(110)	S1: t1 = $\|a\|$ = 9 (1001), t2 = $\|b\|$ = 12 (1100)
S2: x = max(t1, t2)= 10, y = min(t1, t2)= 6	S2: x = max(t1, t2)= 12, y = min(t1, t2)= 9
S3: t3 = x >>3 = 1, t4 = y>>1 = 3	S3: t3 = x >>3 = 1, t4 = y>>1 = 4
S4: t5 = x – t3 = 9	S4: t5 = x – t3 = 11
S5: t6 = t5+ t4 = 12	S5: t6 = t5+ t4 = 15
S6: t7= max(t6, x) = 12	S6: t7= max(t6, x) = 15
S7: result = t7 = 12	S7: result = t7 = 15

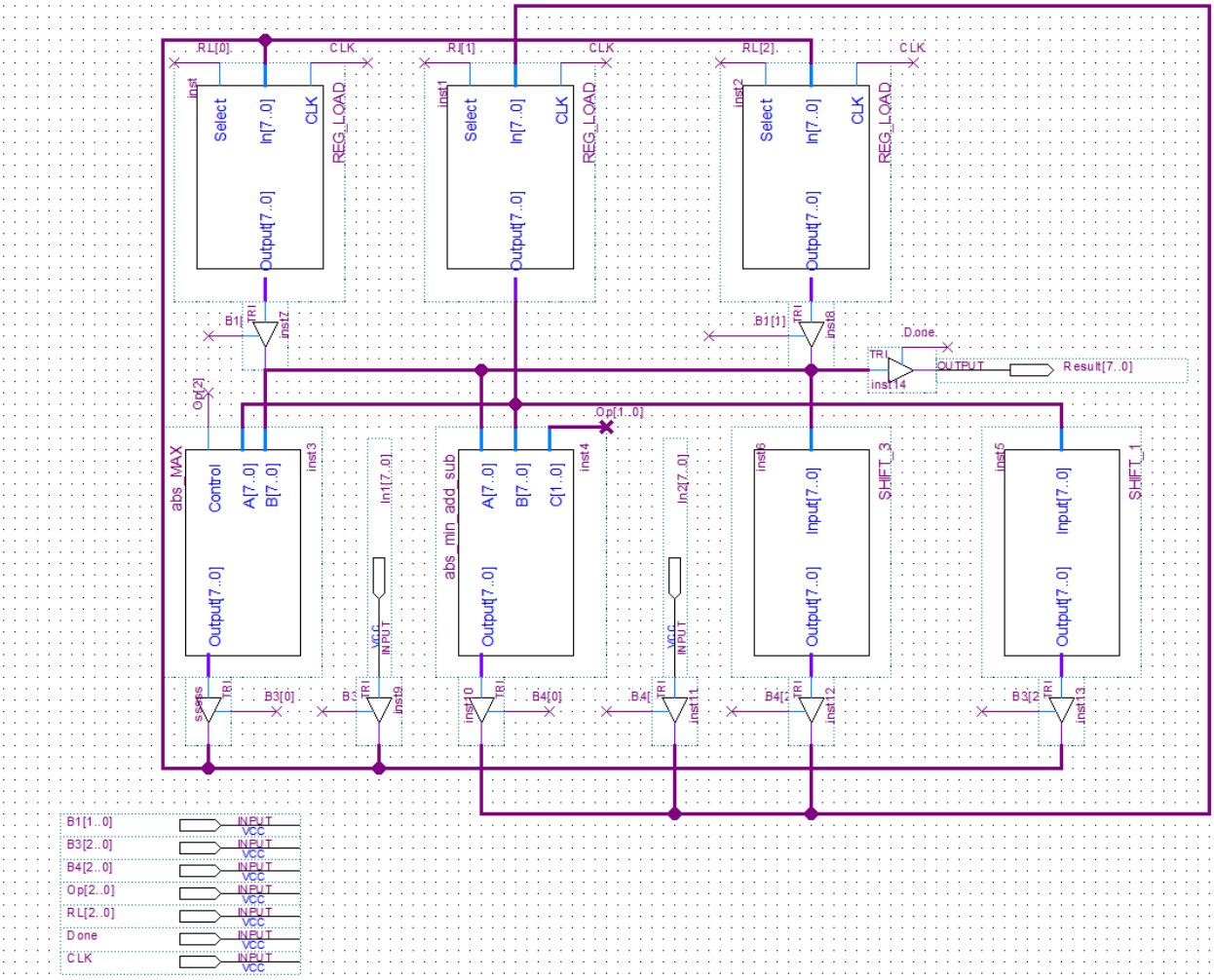
IV. Version 4:

Circuit (includes controller and datapath):



Version combine from Functional logic sharing and bus sharing.

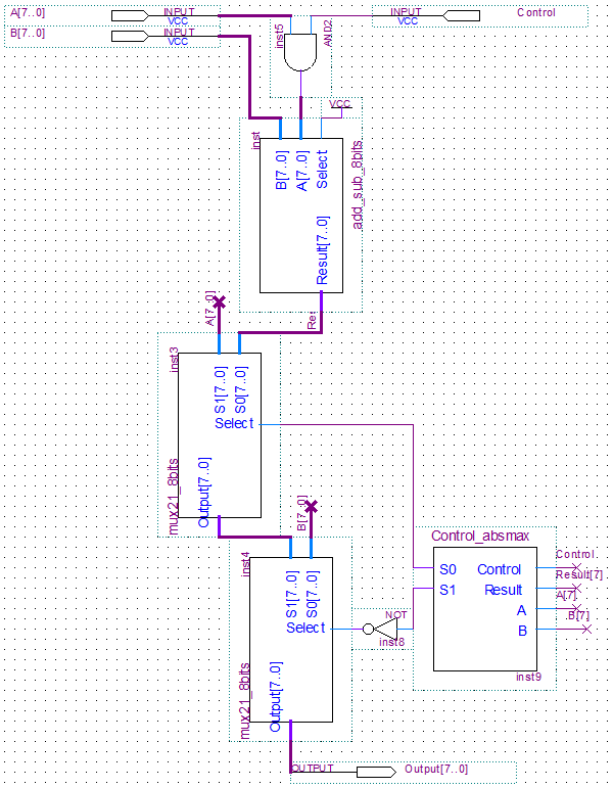
1. Datapath



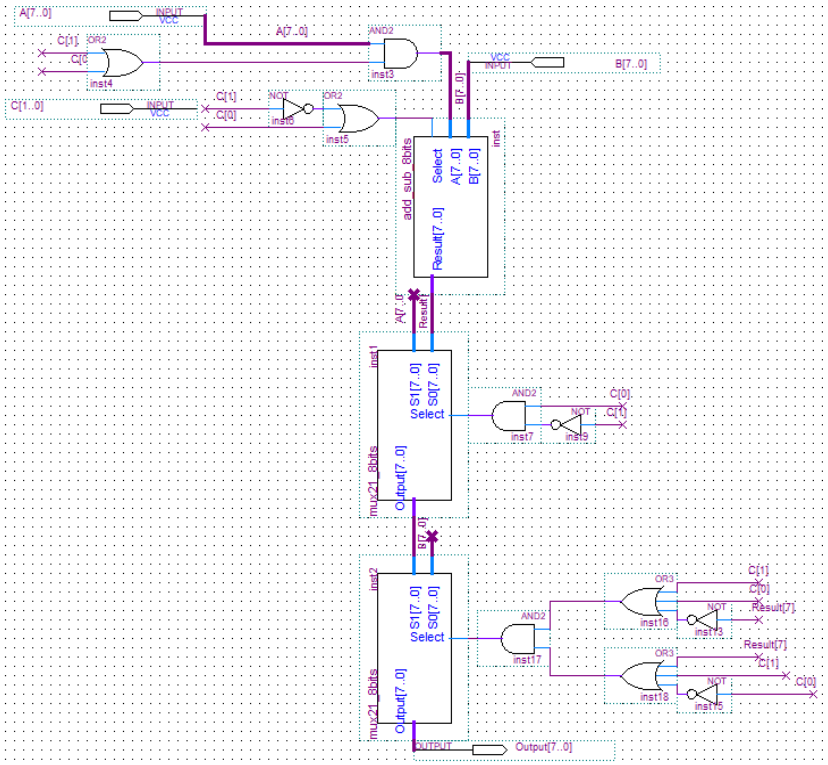
- Functional logic sharing:
  - ABS/Max
  - ABS/Min/+/-
  - Shift 3
  - Shift 1
- Bus Sharing:
  - Bus 1: R1, R3
  - Bus 2: R2
  - Bus 3 : ABS/Max, Input1, Shift 1
  - Bus 4: ABS/Min/+/-, Input2, Shift 3

- **ABS/max :**

- Use 1 Full adder/subtract, 2 Mux 2-1
- If Control = 0, Calculate ABS
    - If sign bit of result = 0, result = this Result
    - Else result = this number
- If Control = 1, Calculate Max
    - If sign bit of result = 0, result = A
    - Else result = B

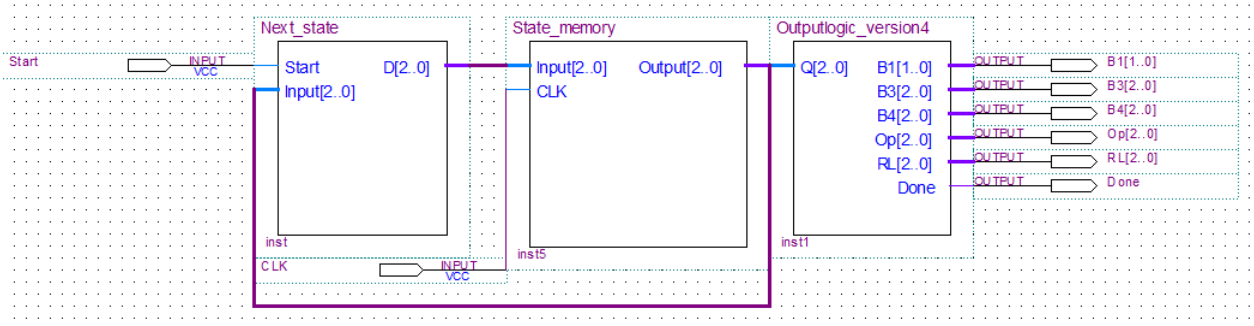


- **ABS/Min/+/-:**



- Use 1 Full adder/subtract, 2 Mux 2-1
- If Control = 00, Calculate ABS
    - If sign bit of result = 0, result = this Result
    - Else result = this number
- If Control = 01, Calculate Max
    - If sign bit of result = 0, result = A
    - Else result = B
- If Control = 10, Calculate A+B
- If Control = 11, Calculate A-B

2. Controller



Controller Block

Controller of version2 differs from version1, version 2 in datapath block control signal

- B1[1..0]: Control Bus1
- B3[2..0]: Control Bus3
- B4[2..0]: Control Bus4
- Op[2..0]: Select Function for Block ABS/Min/+/- and ABS/Max
- RL[2..0]: Write Enable for 3 Register
- Done: Output in last state

- **Identify Next state:**

Q			Start	Q+		
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	0
0	1	0	X	0	1	1
0	1	1	X	1	0	0
1	0	0	X	1	0	1
1	0	1	X	1	1	0
1	1	0	X	1	1	1
1	1	1	X	0	0	0

Bảng xác định tín hiệu next state

- When start equal 1 ,jump to program.
- In state 7, get output.

Use K-Map:

Q0+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	1	0	0
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	1	1	0	0

$$Q_0^+ = \overline{Q_0}S + Q_1\overline{Q_0} + Q_2\overline{Q_0}$$

Q1+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	1	1
	0 1	1	1	0	0
	1 1	1	1	0	0
	1 0	0	0	1	1

$$Q_1^+ = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

Q2+		Q0 Start			
		0 0	0 1	1 1	1 0
Q2 Q1	0 0	0	0	0	0
	0 1	0	0	1	1
	1 1	1	1	0	0
	1 0	1	1	1	1

$$Q_2^+ = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$

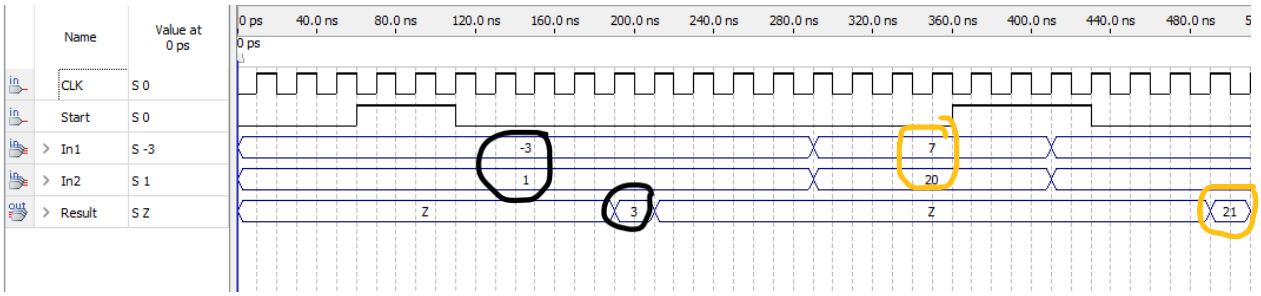
- Table of identify controll in each state:

State	Bus1		Bus3			Bus4			Opcode			RL		
	R1	R3	AU1	In1	SHIFT 1	AU2	In2	SHIFT 3	ABS/max	ABS/min/+/-		R1	R2	R3
S0	0	0	0	1	0	0	1	0	0	0	0	1	1	0
S1	1	0	1	0	0	1	0	0	0	0		1	1	0
S2	1	0	1	0	0	1	0	0	1	0	1	1	1	0
S3	1	0	0	0	1	0	0	1	0	0		0	1	1
S4	1	0	0	0	0	1	0	0	0	1	1	0	1	0
S5	0	1	0	0	0	1	0	0	0	1	0	0	1	0
S6	1	0	1	0	0	0	0	0	1	0		1	0	0
S7	1	0	0	0	0	0	0	0	0	0	0	0	0	0

*Note:

- AU1: ABS/Max; AU2: ABS/Min/+/-
- If Register need to load new value, Write Enable is turn on; Else Write Enable turn off (no change value in Register)

3. Waveform

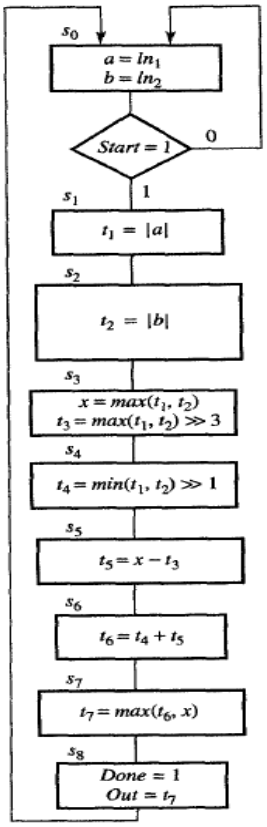


Every 7 cycles after the start signal turns on, output the result (S7), and after S7 reset back to S0 – same as Version 1, Version 2, Version 3.

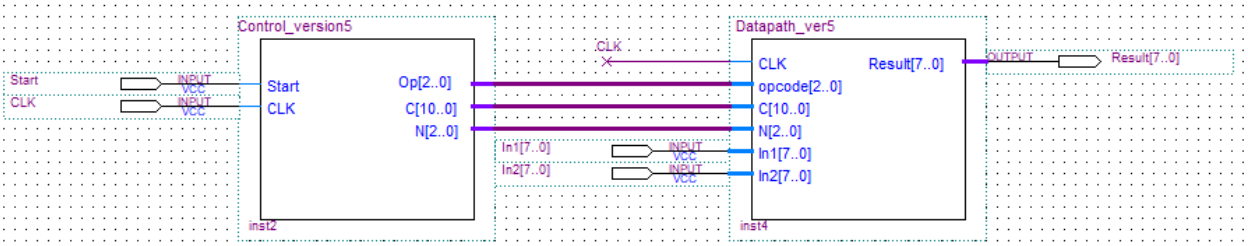
Test:

Input1 = -3 and Input2 = 1	Input1= 7 and Input2 = 20
S0: a = -3 , b = 1	S0: a = 7 , b = 20
S1: t1 = $\|a\|$ = 3(011), t2 = $\|b\|$ = 1(001)	S1: t1 = $\|a\|$ = 20 (10100), t2 = $\|b\|$ = 7 (111)
S2: x = max(t1, t2)= 3, y = min(t1, t2)= 1	S2: x = max(t1, t2)= 20, y = min(t1, t2)= 7
S3: t3 = x >>3 = 0, t4 = y>>1 = 0	S3: t3 = x >>3 = 2, t4 = y>>1 = 3
S4: t5 = x – t3 = 3	S4: t5 = x – t3 = 18
S5: t6 = t5+ t4 = 3	S5: t6 = t5+ t4 = 21
S6: t7= max(t6, x) = 3	S6: t7= max(t6, x) = 21
S7: result = t7 = 3	S7: result = t7 = 21

V. Version 5:

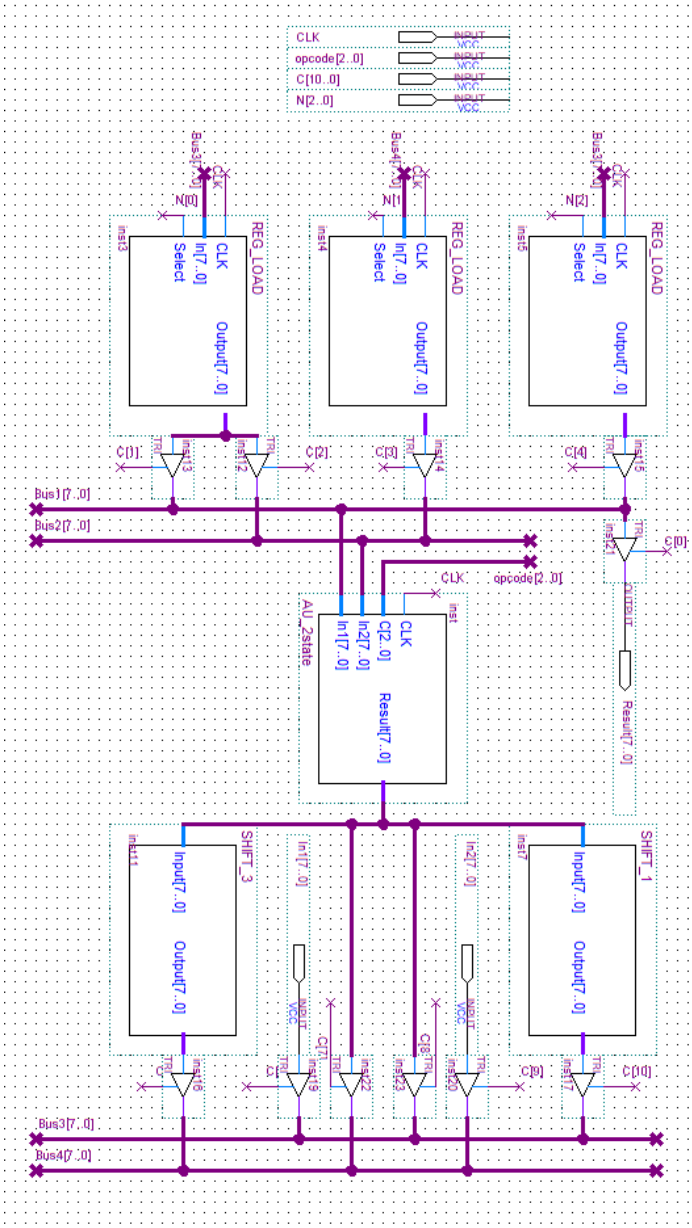


Circuit (includes controller and datapath):



Version 5 use method :"Functional unit pineline". Version 5 is different from the previous 4 versions, the number of states increased by 1.

1. Datapath



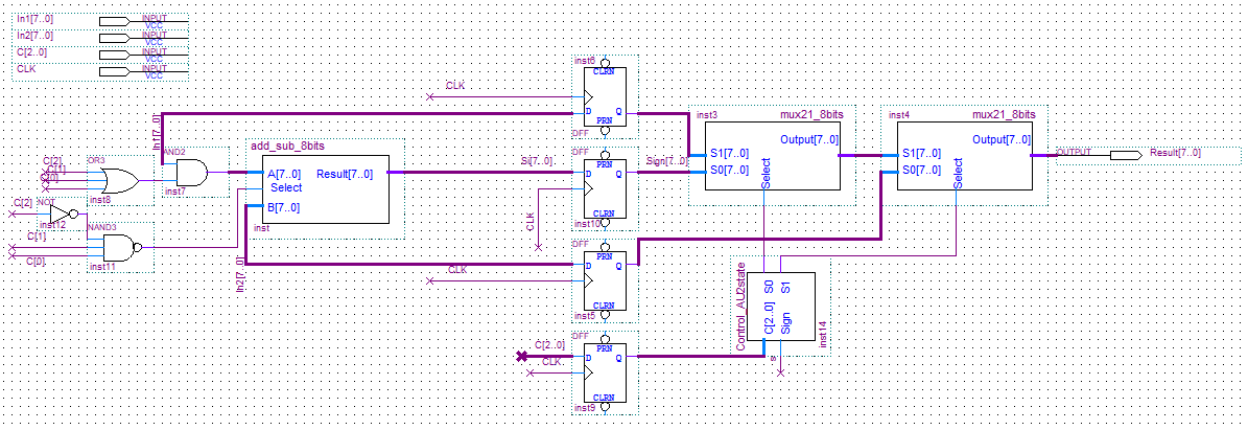
- The datapath has 20 inputs and 1 output, of which 2 inputs (8 bits) transmit the number to be calculated, the remaining 18 inputs are control signals (opcodes of AU blocks, read/write signals of registers, tries, CLK), the output for the final result.

- The datapath consists of 3 registers and an AU blocks, “Shift3” block and “Shift1” block. AU Block are controlled by 3 bit opcodes, the registers store the following values,

  R1 = [a, t1, x, t7]
  R2 = [b, t2, t3, t5, t6]
  R3 = [t4]

(About AU, we divide each calculation into 2 states to reduce the time each state)

- **AU (is used to calculating add/sub/abs/min/max and controlled by Opcode[2..0]):**

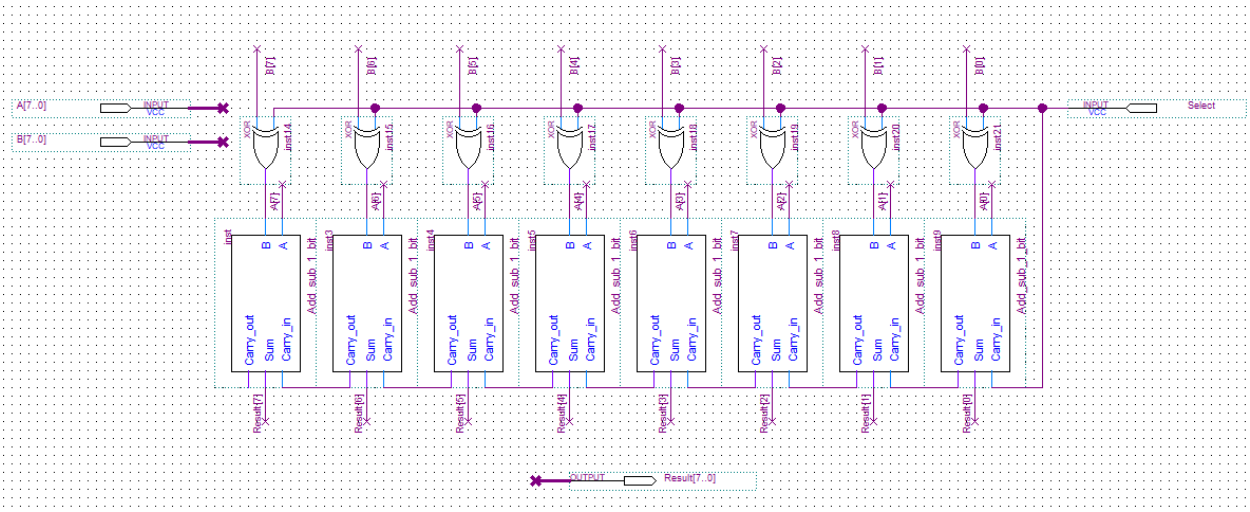


- Use 4 D-FF to devide each calculation into 2 states, the first D-FF save Input1, the 2nd D-FF save result of “Add_sub” Block, the highest bit of it is sign bit, the 3rd save Input2 and the last D-FF save control signal (C[1..0]).

- In A[7..0] of “Add_sub” Block, we connect with ((C[2] + C[1] + C[0]).In1), so only C[2] = C[1] = C[0] = 0 => A[7..0] = 0. Then result of “Add_sub” Block is 0-B this state is used to calculating Absolute.

- We use Result[7] as a sign bit to control 2 Mux Block following.

- “Add_sub” Block:



- “Control” Block:

Select	Function
0	Add
1	Sub

	Sign	IN			OUT			Function	Output of AU
		C[2]	C[1]	C[0]	S1	S0	Select		
A>B	0	0	0	0	0	x	1	abs	B
	0	0	0	1	1	1	1	max	A
	0	0	1	0	0	x	1	min	B
	0	0	1	1	1	0	0	add	Result of Add_sub
	0	1	0	0	1	0	1	sub	
	0	1	0	1	1	0	1		
	0	1	1	0	1	0	1		
	0	1	1	1	1	0	1		
A<B	1	0	0	0	0	x	1	abs	B
	1	0	0	1	0	x	1	max	B
	1	0	1	0	1	1	1	min	A
	1	0	1	1	1	0	0	add	Result of Add_sub
	1	1	0	0	1	0	1	sub	
	1	1	0	1	1	0	1		
	1	1	1	0	1	0	1		
	1	1	1	1	1	0	1		

S1		C[1]_C[0]			
		00	01	11	10
Sign_C[2]	00	0	1	1	0
	01	1	1	1	1
	11	1	1	1	1
	10	0	0	1	1

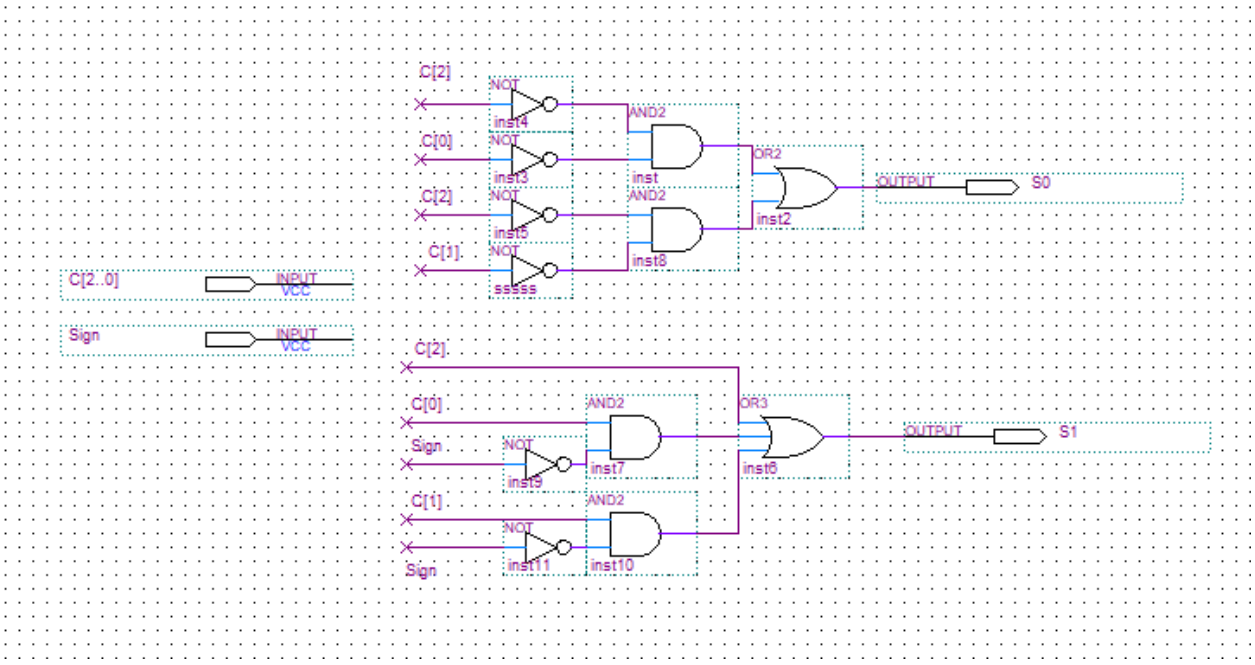
S0		C[1]_C[0]			
		00	01	11	10
Sign_C[2]	00	x	1		x
	01				
	11				
	10	x	x		1

S1 = C[2] + Sign.C[1] + Sign’.C[0]

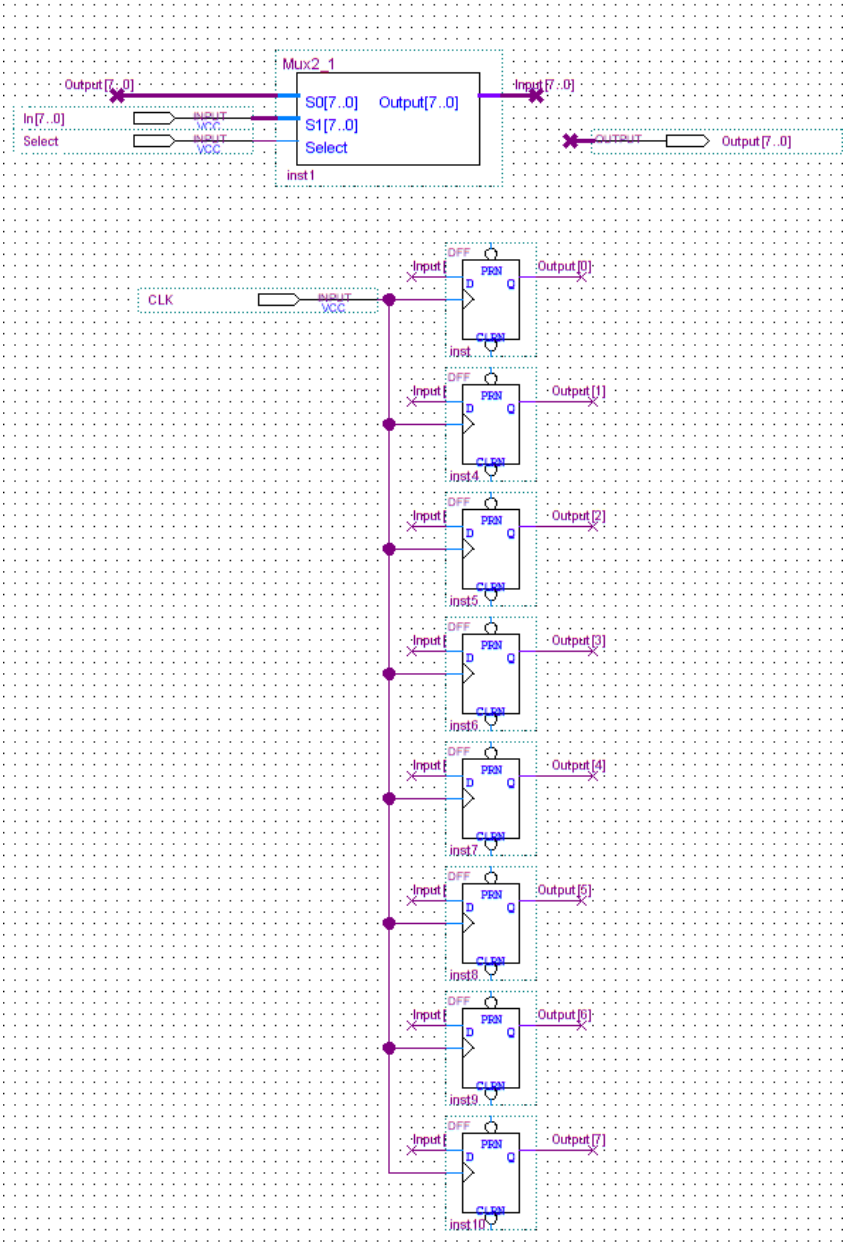
S0 = C[2]’.C[0]’ + C[2]’.C[1]’

Select		C[1]_C[0]			
		00	01	11	10
Sign_C[2]	00	1	1	0	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	0	1

Select = (C[2]’ x C[1] x C[0])’

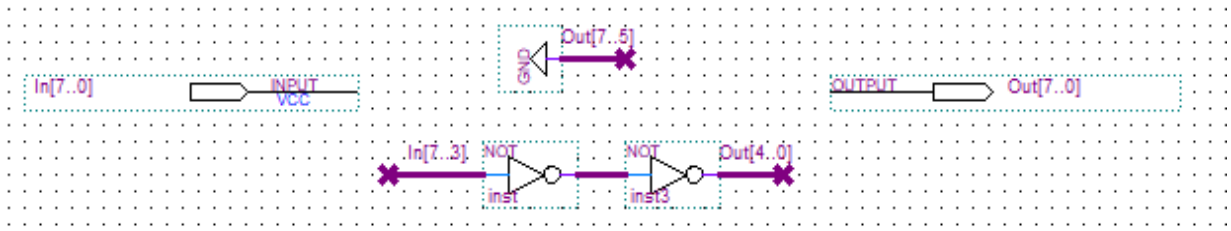


- **Register:**



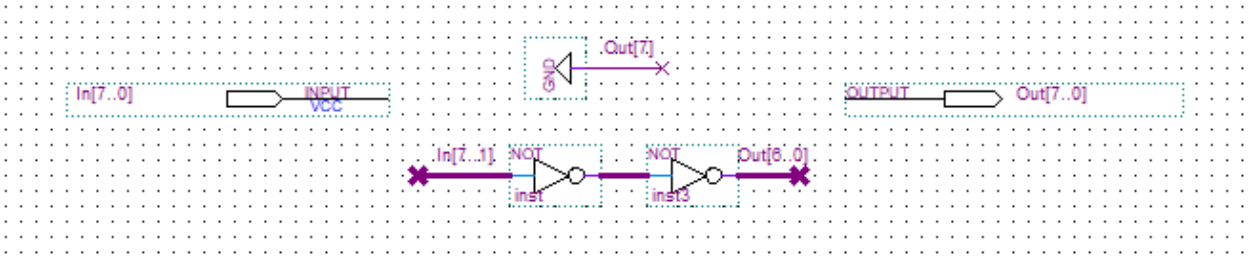
- Register if created by 8 D-FF and “Mux2_1”.
- The input Select decides that the register loads new value or not. If Select = 0 => no load new value and put previous value (S0) into D-FF so it can stores value, otherwise, Select = 1 => load new value (S1).
- In quartus, we can use only 1 DFF to save bus data.

- **Shift 3 (8 bits):**



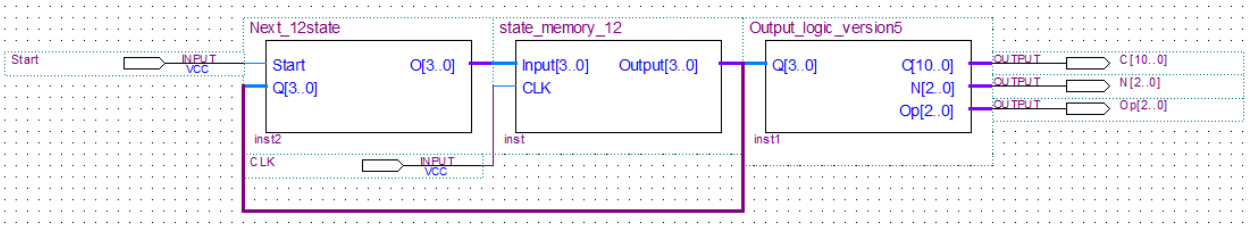
Connect bits [7..3] of input to bits[4..0] of output, remaining bits of output are GND.

- **Shift 1 (8 bits):**



Connect bits [7..1] of input to bits[6..0] of output, remaining bit of output is GND.
We can change 2 "Not" logic gate by 1 "carry" logic gate

2. Controller



Note (according to datapath):

- **Bus3:**

- C[6]: control tri allows transferring data from Input1 to Bus_3.
- C[8]: control tri allows transferring data from AU to Bus_3.
- C[10]: control tri allows transferring data from Shift1 to Bus_3.

(in the same time only Input1 or AU or Shift1 load data to Bus_3)

- **Bus2:**

- C[2]: control tri allows loading data from Register1 to Bus_2.
- C[3] control tri allows loading data from Register2 to Bus_2.
  (in the same time only register1 or register2 load data to Bus_2)

- **Bus4:**

- C[5]: control tri allows transferring data from Shift3 to Bus_4.
- C[7]: control tri allows transferring data from AU to Bus_4.
- C[9]: control tri allows transferring data from Input2 to Bus_4.
  (in the same time only Input2 or AU or Shift3 load data to Bus_4)

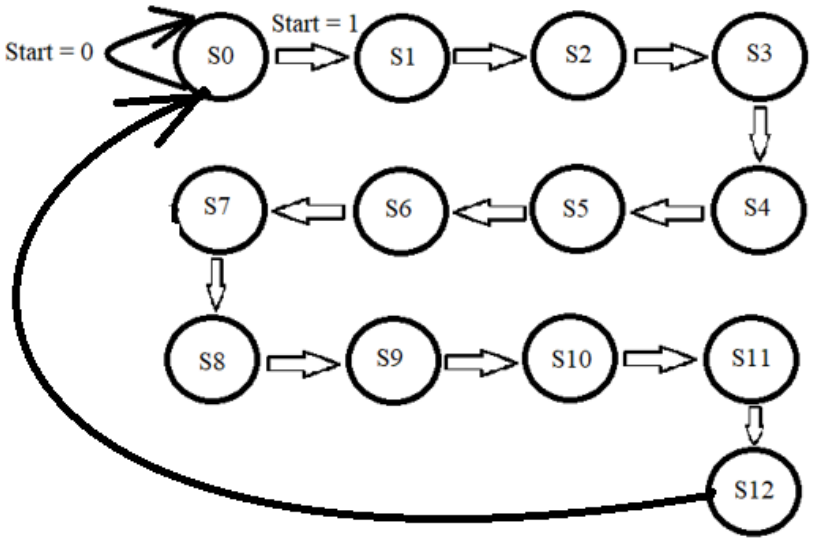
- **Bus1:**

- C[1]: control tri allows transferring data from Register1 to Bus_1.
- C[4]: control tri allows transferring data from Register3 to Bus_1.
  (in the same time only Register1 or Register3 load data to Bus_1)
- C[0]: when state is S12 (after successfully calculated final result we will have the output is result, in other states output is "X")

- **Write Enable:**

- N[0]: Allows to load data to Register1.
- N[1]: Allows to load data to Register2.
- N[2]: Allows to load data to Register3.
- Opcode[2..0]: control AU.

- **Next state:**



Temporarily removed Start to easy to draw K_map, after that we or(+) Q_{0+} with Start.S_0, because Start = 1 only in case that input(Q3_Q2_Q1_Q0) = 0000 and output(Q3+_Q2+_Q1+_Q0+) = 0001:

STATE	IN					OUT			
	Start	Q3	Q2	Q1	Q0	Q3	Q2	Q1	Q0
S0	0	0	0	0	0	0	0	0	0
S0	1	0	0	0	0	0	0	0	1
S1	X	0	0	0	1	0	0	1	0
S2	X	0	0	1	0	0	0	1	1
S3	X	0	0	1	1	0	1	0	0
S4	X	0	1	0	0	0	1	0	1
S5	X	0	1	0	1	0	1	1	0
S6	X	0	1	1	0	0	1	1	1
S7	X	0	1	1	1	1	0	0	0
S8	X	1	0	0	0	1	0	0	1
S9	X	1	0	0	1	1	0	1	0
S10	X	1	0	1	0	1	0	1	1
S11	X	1	0	1	1	1	1	0	0
S12	X	1	1	0	0	0	0	0	0
S13	X	1	1	0	1	X	X	X	X
S14	X	1	1	1	0	X	X	X	X
S15	X	1	1	1	1	X	X	X	X

Q3+	Q1_Q0			
Q3_Q2	00	01	11	10
00				
01			1	
11		X	X	X
10	1	1	1	1

Q2+	Q1_Q0			
Q3_Q2	00	01	11	10
00			1	
01	1	1		1
11		X	X	X
10			1	

$Q_{2+} = Q_3'.Q_2.Q_1' + Q_2'.Q_1.Q_0 + Q_2.Q_1.Q_0'$

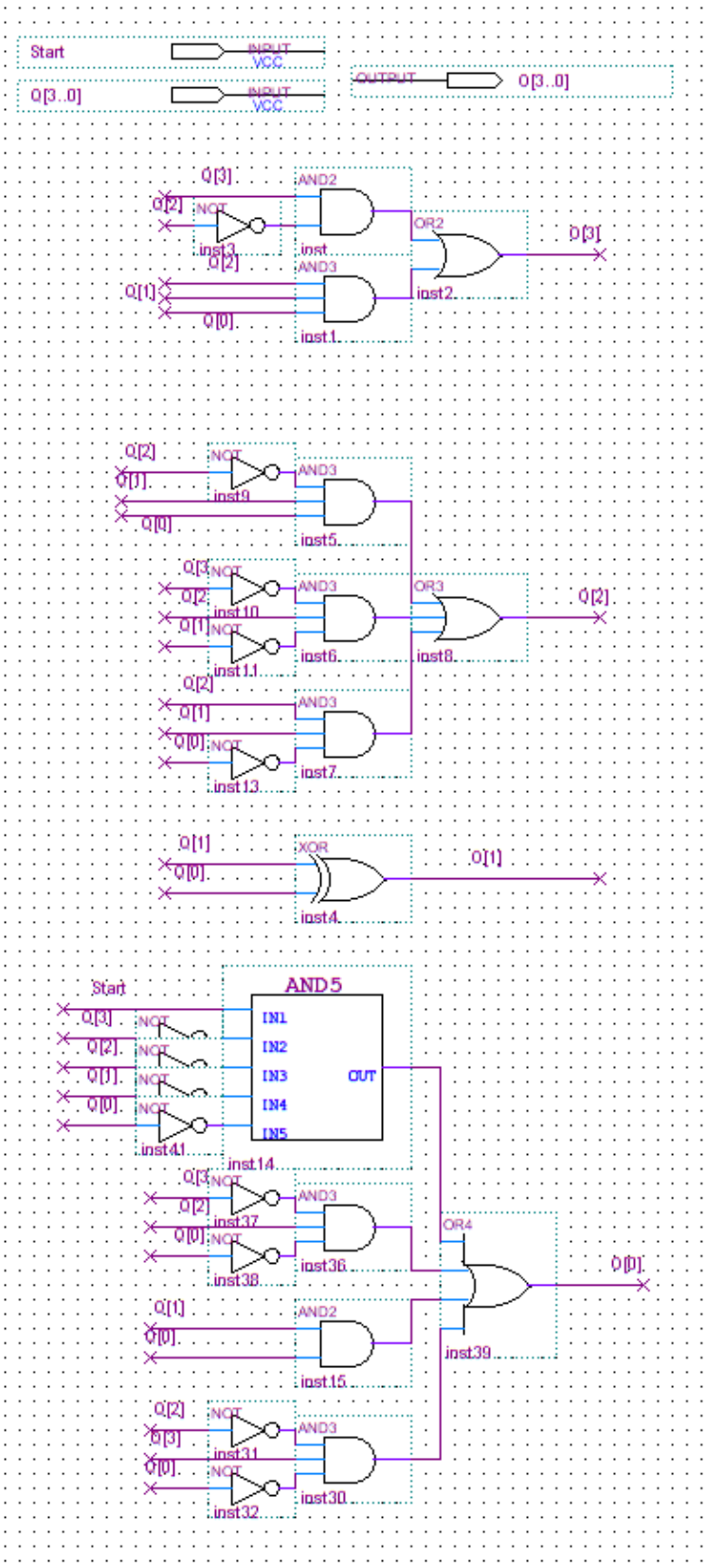
$Q_{3+} = Q_3.Q_2' + Q_2.Q_1.Q_0$

Q1+	Q1_Q0			
Q3_Q2	00	01	11	10
00		1		1
01		1		1
11		X	X	X
10		1		1

$Q_{1+} = Q_1 \oplus Q_0$

$Q_{0+} = Q_1.Q_0 + Q_3.Q_2'.Q_0' + Q_3'.Q_2.Q_0' + Start.Q_3'.Q_2'.Q_1'.Q_0'$

Q0+	Q1_Q0			
Q3_Q2	00	01	11	10
00				1
01	1			1
11		X	X	X
10	1			1


Start
INPUT
VCC
Q[3..0]
INPUT
VCC
OUTPUT
Q[3..0]
Q[3]
Q[2]
NOT
inst3
AND2
inst
OR2
Q[3]
Q[2]
AND3
Q[1]
Q[0]
inst1
inst2
Q[2]
NOT
Q[1]
Q[0]
inst9
AND3
inst5
Q[3]
Q[2]
NOT
inst10
AND3
OR3
Q[2]
Q[1]
NOT
inst11
inst6
inst8
Q[2]
Q[1]
Q[0]
NOT
inst13
AND3
inst7
Q[1]
Q[0]
XOR
Q[1]
inst4
Start
Q[3]
Q[2]
Q[1]
Q[0]
NOT
inst41
AND5
IN1
IN2
IN3
IN4
IN5
OUT
inst14
Q[3]
Q[2]
Q[0]
NOT
inst37
inst38
AND3
inst36
OR4
Q[0]
Q[1]
Q[0]
AND2
inst15
inst39
Q[2]
Q[3]
Q[0]
NOT
inst31
inst32
AND3
inst30

- **Decoder from state to controll:**

STATE	IN				OUT															
	Q3	Q2	Q1	Q0	I1B3	I2B4	R1B1	R3B1	R1B2	R2B2	S3B4	S1B3	AU1B3	AU1B4	LR1	LR2	LR3	OP[2]	OP[1]	OP[0]	DONE
S0	0	0	0	0	1	1									1	1					
S1	0	0	0	1					1									0	0	0	
S2	0	0	1	0						1			1		1			0	0	0	
S3	0	0	1	1										1		1					
S4	0	1	0	0			1			1								0	0	1	
S5	0	1	0	1			1			1	1			1	1	1		0	1	0	
S6	0	1	1	0			1			1		1	1				1	1	X	X	
S7	0	1	1	1									1			1					
S8	1	0	0	0				1		1								0	1	1	
S9	1	0	0	1										1		1					
S10	1	0	1	0			1			1								0	0	1	
S11	1	0	1	1									1		1						
S12	1	1	0	0			1														1
S13	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
S14	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
S15	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

I1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01				
Q3_Q2 11		X	X	X
Q3_Q2 10				

I1B3 = $Q_3'.Q_2'.Q_1'.Q_0'$

I2B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01				
Q3_Q2 11		X	X	X
Q3_Q2 10				

I2B4 = $Q_3'.Q_2'.Q_1'.Q_0'$

R1B1	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	X	X	X
Q3_Q2 01	1	1	X	1
Q3_Q2 11	1	X	X	X
Q3_Q2 10	0	X	X	1

R1B1 = $Q_2 + Q_1$

R3B1	Q1_Q0			
	00	01	11	10
Q3_Q2 00	0	0	0	0
Q3_Q2 01	0	0	0	0
Q3_Q2 11	0	0	0	0
Q3_Q2 10	1	0	0	0

R3B1 = $Q_3.Q_2'.Q_1'.Q_0'$

R1B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00	0	1	0	0
Q3_Q2 01	0	0	0	0
Q3_Q2 11	0	0	0	0
Q3_Q2 10	0	0	0	0

R1B2 = $Q_3'.Q_2'.Q_1'.Q_0$

R2B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	0	X	1
Q3_Q2 01	1	1	X	1
Q3_Q2 11	X	X	X	X
Q3_Q2 10	1	X	X	1

R2B2 = $Q_3 + Q_2 + Q_1$

S3B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01		1		
Q3_Q2 11				
Q3_Q2 10				

S3B4 = $Q_3'.Q_2.Q_1'.Q_0$

AUB4	Q1_Q0			
	00	01	11	10
Q3_Q2 00	0	X	1	X
Q3_Q2 01	X	0	X	X
Q3_Q2 11	X	X	X	X
Q3_Q2 10	X	1	X	X

AUB4 = $Q_3 + Q_1$

S1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				1
Q3_Q2 11				
Q3_Q2 10				

S1B3 = $Q_3'.Q_2.Q_1.Q_0'$

AUB3	Q1_Q0			
	00	01	11	10
Q3_Q2 00	0	X	X	1
Q3_Q2 01	X	1	1	0
Q3_Q2 11	X	X	X	X
Q3_Q2 10	X	X	1	X

AUB3 = $Q_3'.Q_2'.Q_1 + Q_0$

LR1	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			1
Q3_Q2 01		1		
Q3_Q2 11		X	X	X
Q3_Q2 10			1	

LR1 = $Q_3'.Q_2'.Q_0' + Q_2.Q_1'.Q_0 + Q_3.Q_1.Q_0$

LR2	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1		1	
Q3_Q2 01		1	1	
Q3_Q2 11		X	X	X
Q3_Q2 10		1		

LR2=$Q_3'.Q_2'.Q_1'.Q_0' + Q_2.Q_0 + Q_3.Q_1'.Q_0 + Q_3'.Q_1.Q_0$

LR3	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				1
Q3_Q2 11		X	X	X
Q3_Q2 10				

LR3 = $Q_2.Q_1.Q_0'$

OP[2]	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	0	X	0
Q3_Q2 01	0	0	X	1
Q3_Q2 11	X	X	X	X
Q3_Q2 10	0	X	X	0

OP[3] = $Q_2.Q_1$

OP[1]	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	0	X	0
Q3_Q2 01	0	1	X	X
Q3_Q2 11	X	X	X	X
Q3_Q2 10	1	X	X	0

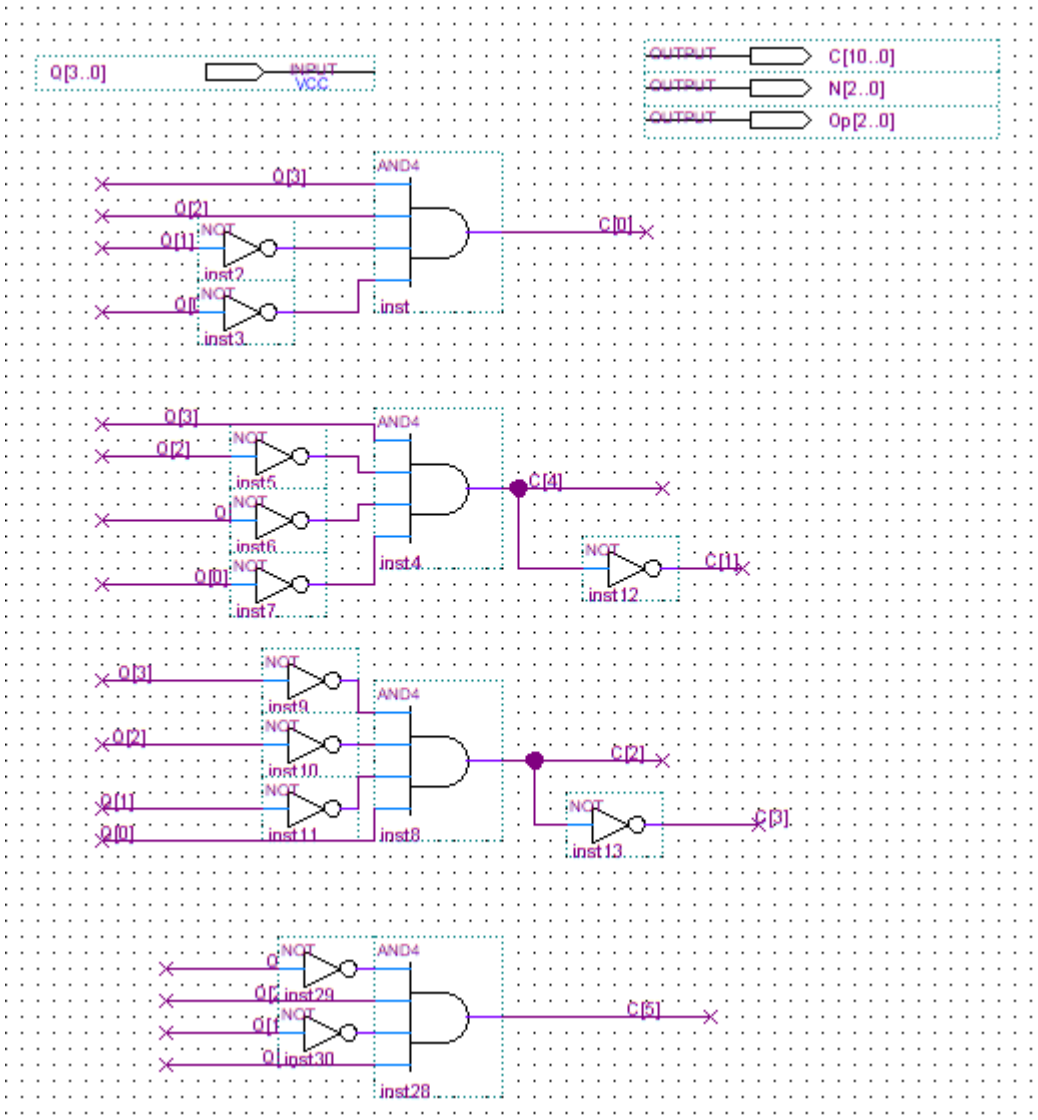
OP[1] = $Q_2.Q_0 + Q_3.Q_1'$

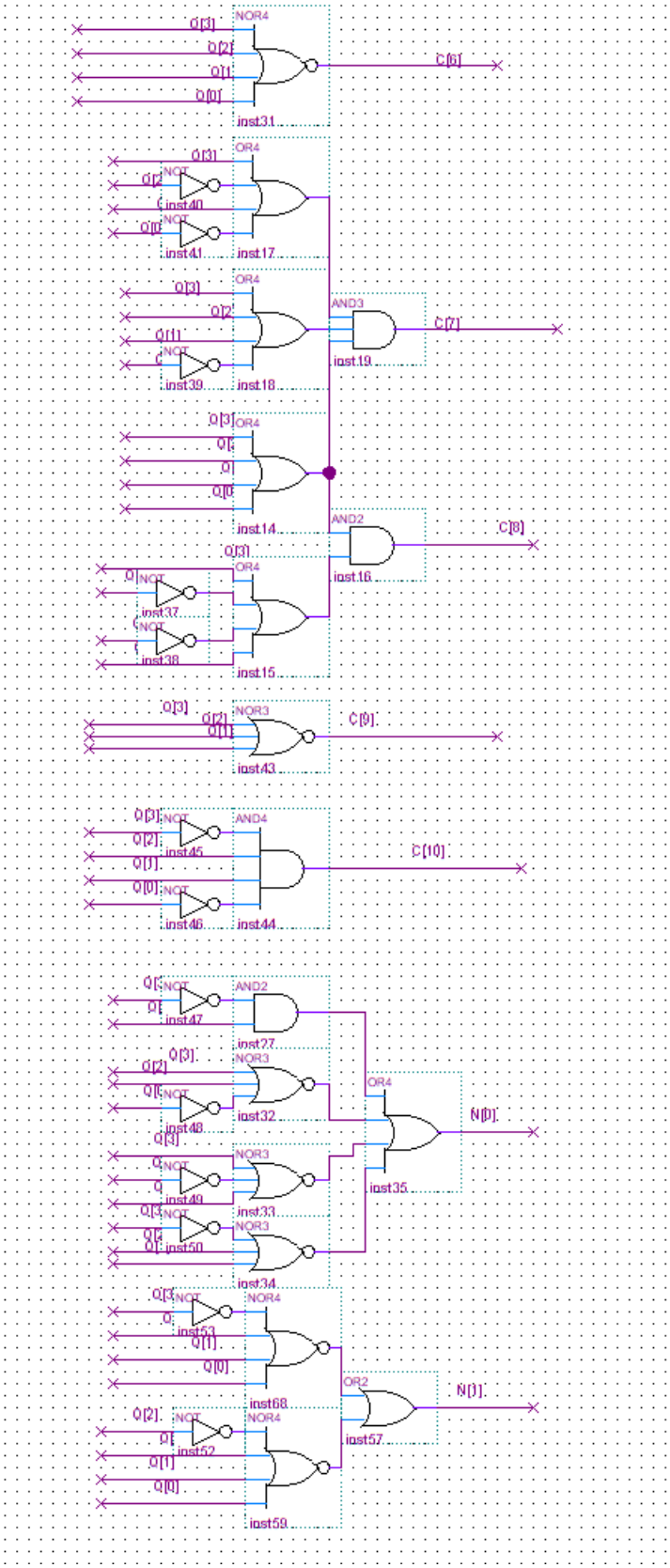
OP[1]	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	0	X	0
Q3_Q2 01	1	0	X	X
Q3_Q2 11	X	X	X	X
Q3_Q2 10	1	X	X	1

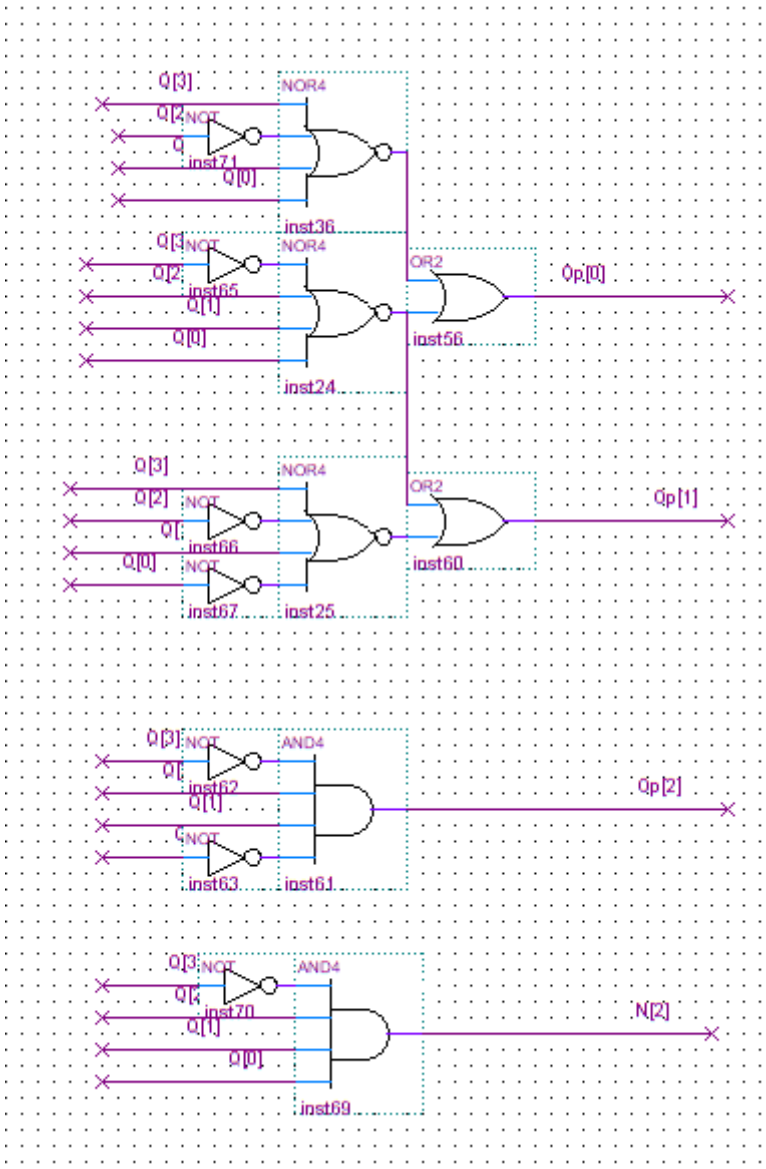
OP[0] = $Q_3 + Q_1'.Q_0'$

OP[1]		Q1_Q0			
		00	01	11	10
Q3_Q2	00				
	01				
	11	1	X	X	X
	10				

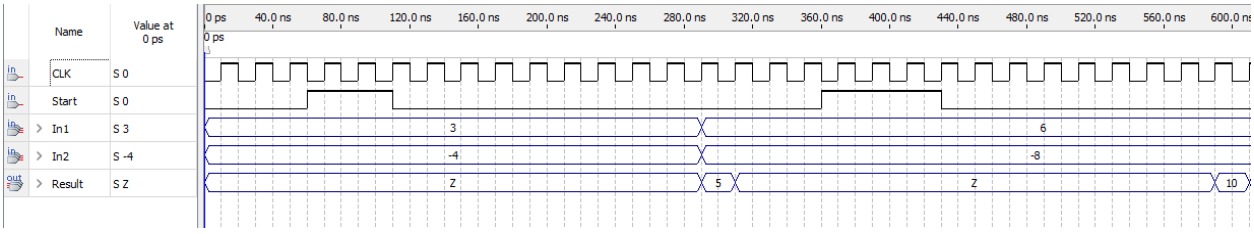
DONE = $Q_3.Q_2$




NOR4
Q[3]
Q[2]
Q[1]
Q[0]
C[6]
inst31
OR4
NOT
inst40
inst41
inst17
AND3
C[7]
inst19
inst39
inst18
inst14
AND2
C[8]
inst16
inst37
inst38
inst15
NOR3
C[9]
inst43
AND4
C[10]
inst45
inst46
inst44
inst47
inst27
inst48
inst32
N[0]
inst35
inst49
inst33
inst50
inst34
inst53
inst68
N[1]
inst57
inst52
inst59


Q[3]
Q[2]
Q[1]
Q[0]
NOT
NOR4
inst71
inst36
OR2
Qp[0]
inst56
inst65
inst24
inst66
inst67
inst25
Qp[1]
inst60
AND4
inst62
inst63
inst61
Qp[2]
inst70
inst69
N[2]

3. Waveform

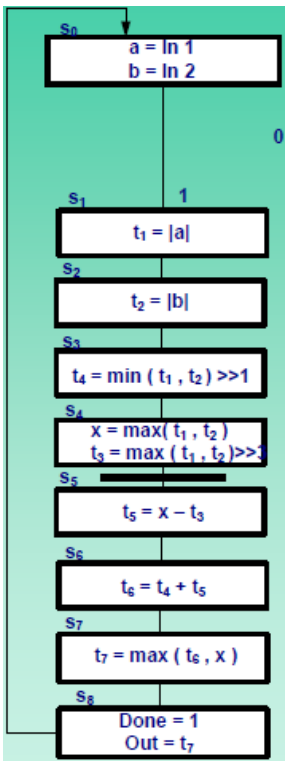

Name
Value at 0 ps
CLK
S 0
Start
S 0
In1
S 3
In2
S -4
Result
S Z
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
400.0 ns
440.0 ns
480.0 ns
520.0 ns
560.0 ns
600.0 ns
3
6
-4
-8
Z
5
Z
10

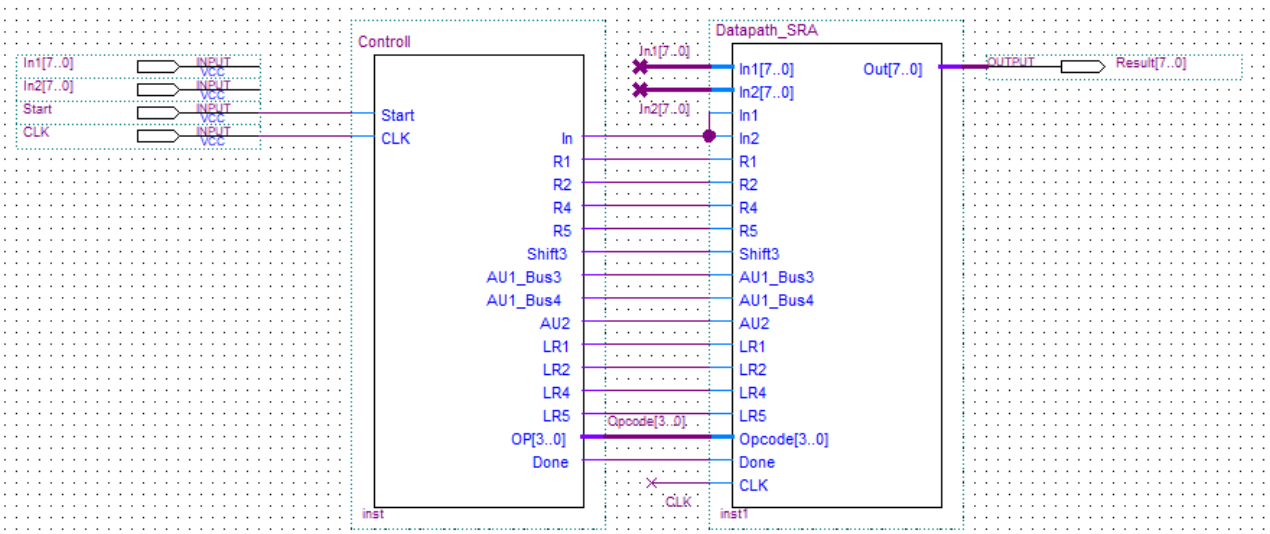
Input1 = -3 and Input2 = 4	Input1= 6 and Input2 = -8
S0: a = 3 , b = -4	S0: a = 6 , b = -8
S1: t1 = $\|a\|$ = 3(011),	S1: t1 = $\|a\|$ = 6 (110),
S2: t2 = $\|b\|$ = 1(100)	S2:t2 = $\|b\|$ = 8 (1000)
S3: x = max(t1, t2)= 4, t3 = x>>3 =0	S3: x = max(t1, t2)= 8, t3 = x>>3 =1
S4: t4 = min(t1, t2)>>1= 1	S4: t4 = min(t1,t2)>>1 = 3
S5: t5 = x – t3 = 4	S4: t5 = x – t3 = 7
S6: t6 = t5+ t4 = 5	S5: t6 = t5+ t4 = 10
S7: t7= max(t6, x) = 5	S6: t7= max(t6, x) = 10
S8: result = t7 = 5	S7: result = t7 = 10

VI. Version 6:

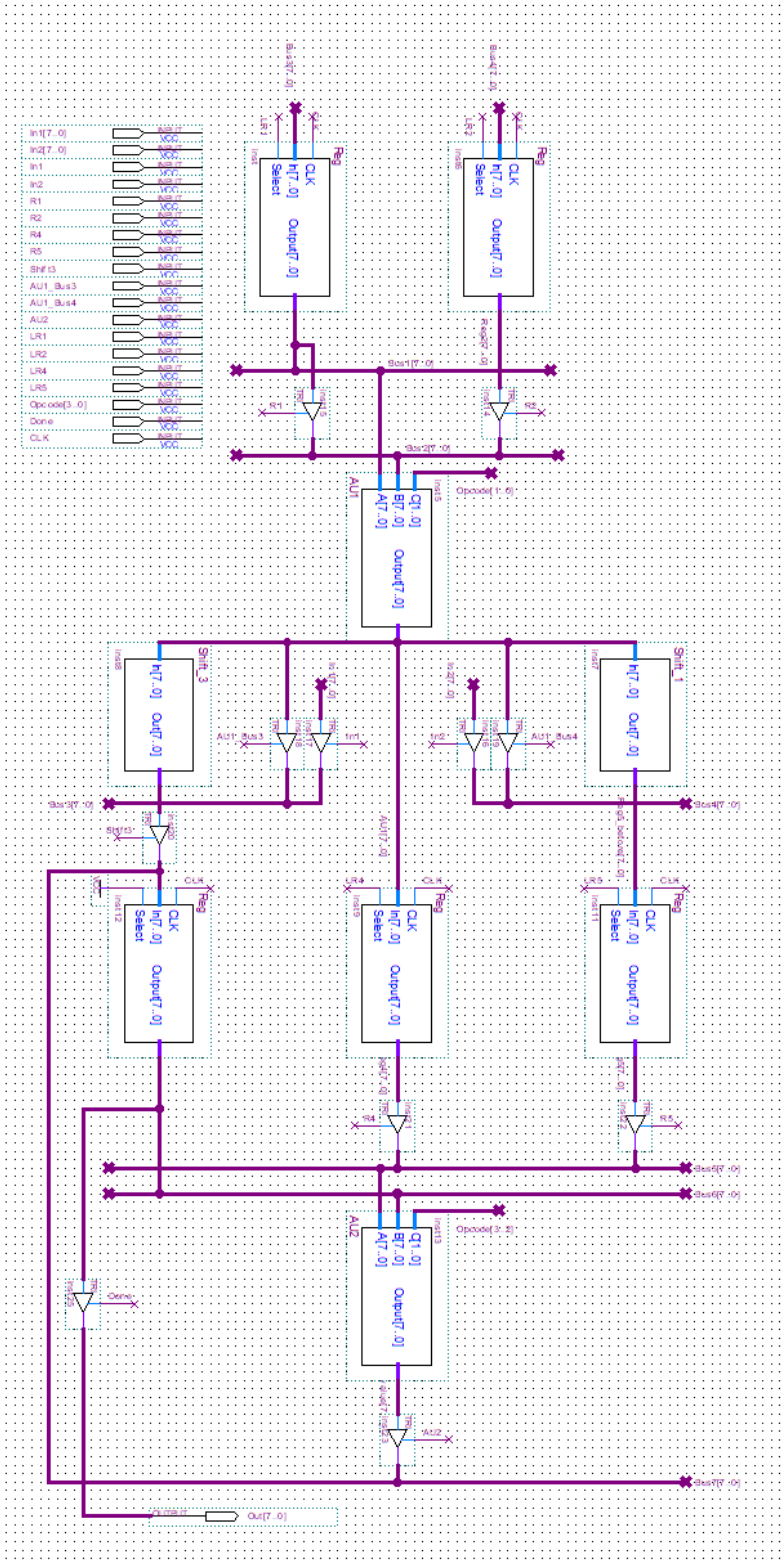
Version 5 is update of version4 , and it use datapath pineline.

Circuit (includes controller and datapath):





1. Datapath



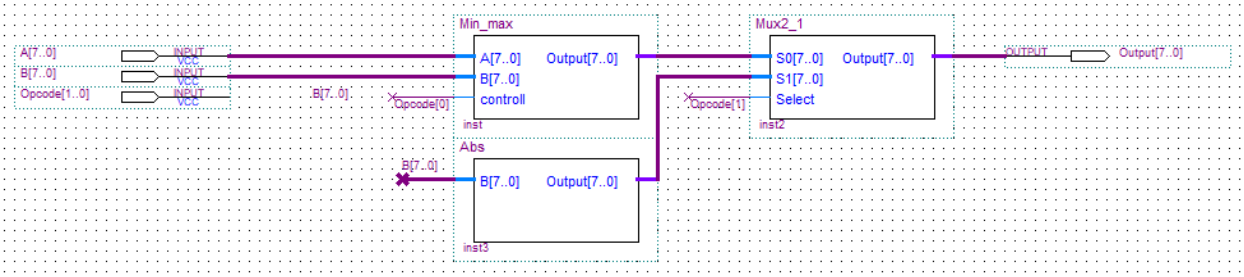
- The datapath has 19 inputs and 1 output, of which 2 inputs (8 bits) transmit the number to be calculated, the remaining 17 inputs are control signals

(opcodes of AU blocks, read/write signals of registers, tries), the output for the final result.

- The datapath consists of 7 registers and 2 AU blocks, “Shift3” block and “Shift1” block. “AU1” and “AU2” are controlled by 4 bit opcodes (Opcode[3..2] control AU2, Opcode[1..0] control AU1), the registers store the following values, “AU1” and “AU2” perform the following funtions:

  R1 = [a, t1] R3 = [t3, t5, t6, t7]
  R2 = [b, t2] R4 = [x]
  AU1 = [abs/min/max] R5 = [t4]
  AU2 = [+/-/max]

- **AU1 (is used to calculating abs/min/max and controlled by Opcode[1..0]):**



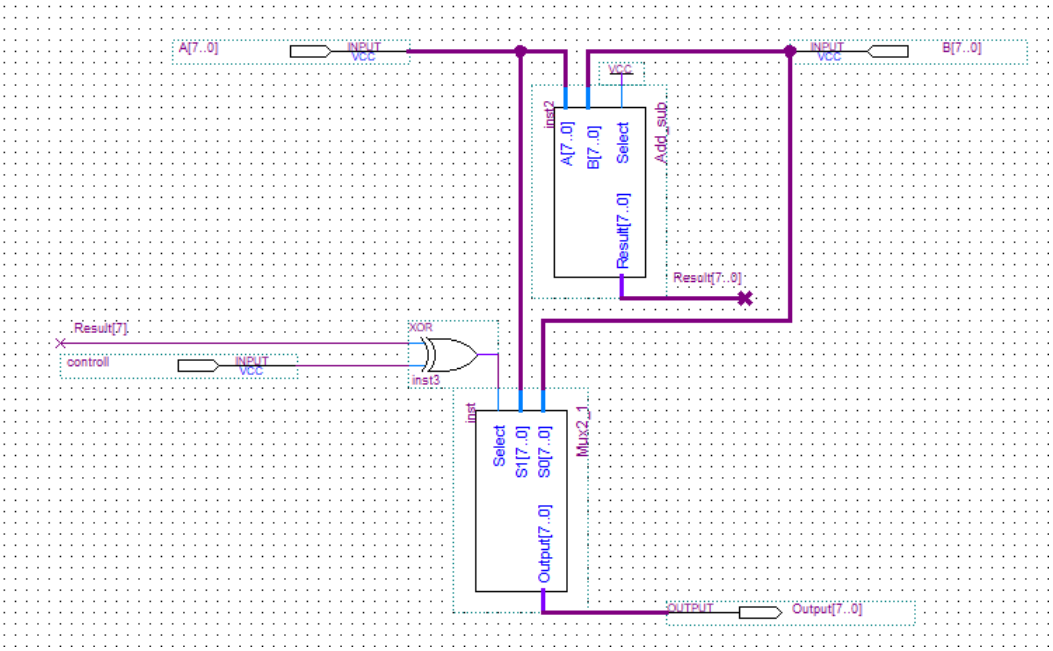
AU1	
Opcode	Function
00	min
01	max
10	abs
11	abs

- Opcode[1] connects to "Mux2_1" block to choose result of “Abs” block or “Min_max” block, Opcode[0] connects to "Min_max" block to control *min* or *max* calculation.

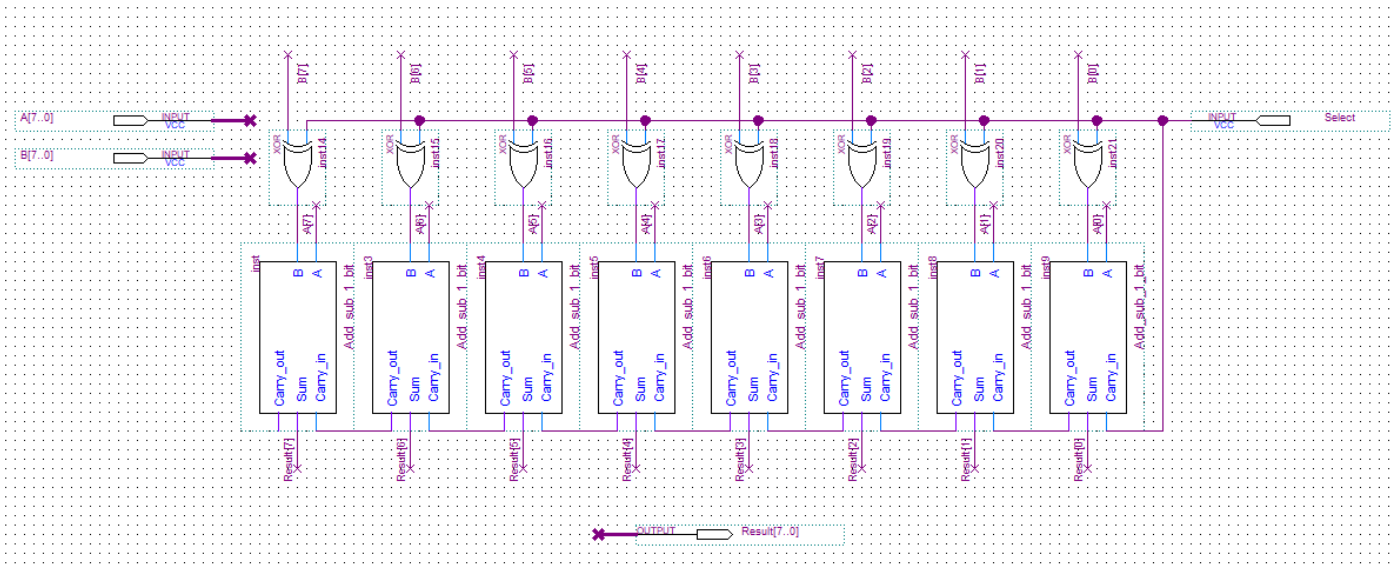
- If Opcode[1] = 1 then “AU1” performs *absolute* calculation, otherwise Opcode[1] = 0, “AU1” will choose the result calculated in the "Min_max" block.

- The "Min_max" block is controlled by Opcode[0], if Opcode[0] = 0 calculates *min*, Opcode[0] = 1 calculates *max*.

- **Min_Max:**



Add_sub:



Select	Function
0	Add
1	Sub

Connect vcc to Select of “Add_sub” block to calculate A-B. Then take the sign bit against the controller signal to decide whether to choose A or B to match the signal. In “Mux21” connect A to S1, B to S0:

Mux	
Select	Result
0	S0
1	S1

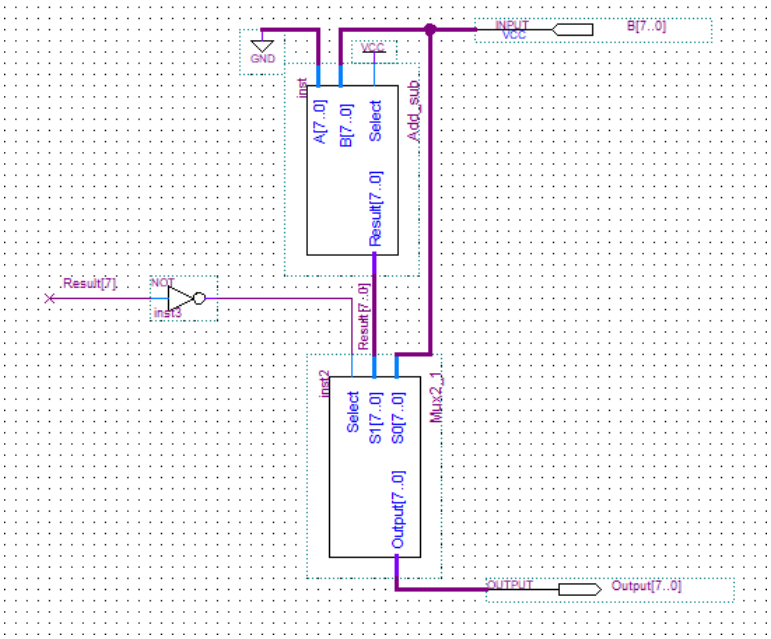
	Result[7]	Control		Select	Function
S1>=S0	0	0		0	min
	0	1		1	max
S1<S0	1	0		1	min
	1	1		0	max

Select = Control XOR Result[7]

- A>=B => A-B>0 => Result[7] = 0
  - Control = 0 => Select = Control XOR Result[7]
    = 0 XOR 0 = 0 => Select S0 (B) (min)
  - Control = 1 => Select = Control XOR Result[7]
    = 0 XOR 1 = 1 => Select S1 (A) (max)

- A<B => A-B<0 => Result[7] = 1
  - Control = 0 => Select = Control XOR Result[7]
    = 0 XOR 1 = 1 => Select S1 (A) (min)
  - Control = 1 => Select = Control XOR Result[7]
    = 1 XOR 1 = 0 => Select S0 (B) (max)

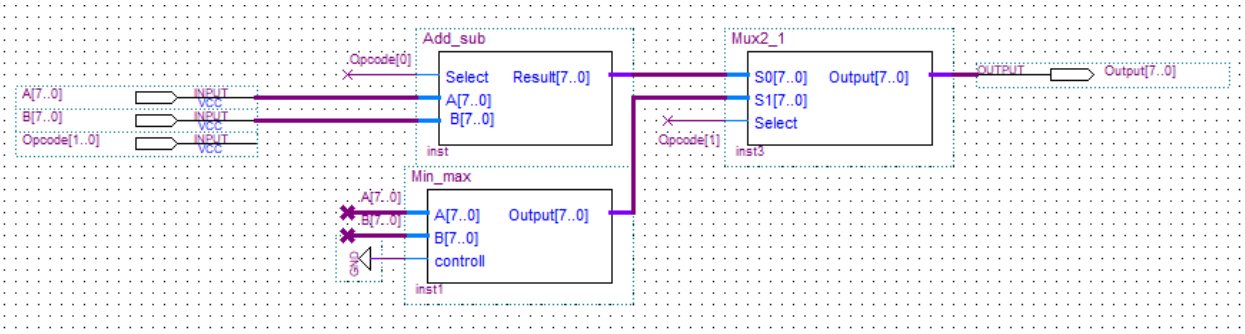
- **Abs:**



Connect vcc to Select and GND to A[7..0] of “Add_sub” block to calculate 0-B .

- If B < 0 then 0 – B > 0, so Result[7]=0 => Select = Result[7]’=1 => Output = S1 = 0 - B = -B.
- If B > 0 then 0 – B < 0, so Result[7]=1 => Select = Result[7]’ = 0 => Output = S0 = B.

- **AU2:**



AU2	
Opcode	Funtion
00	+
01	-
10	max
11	max

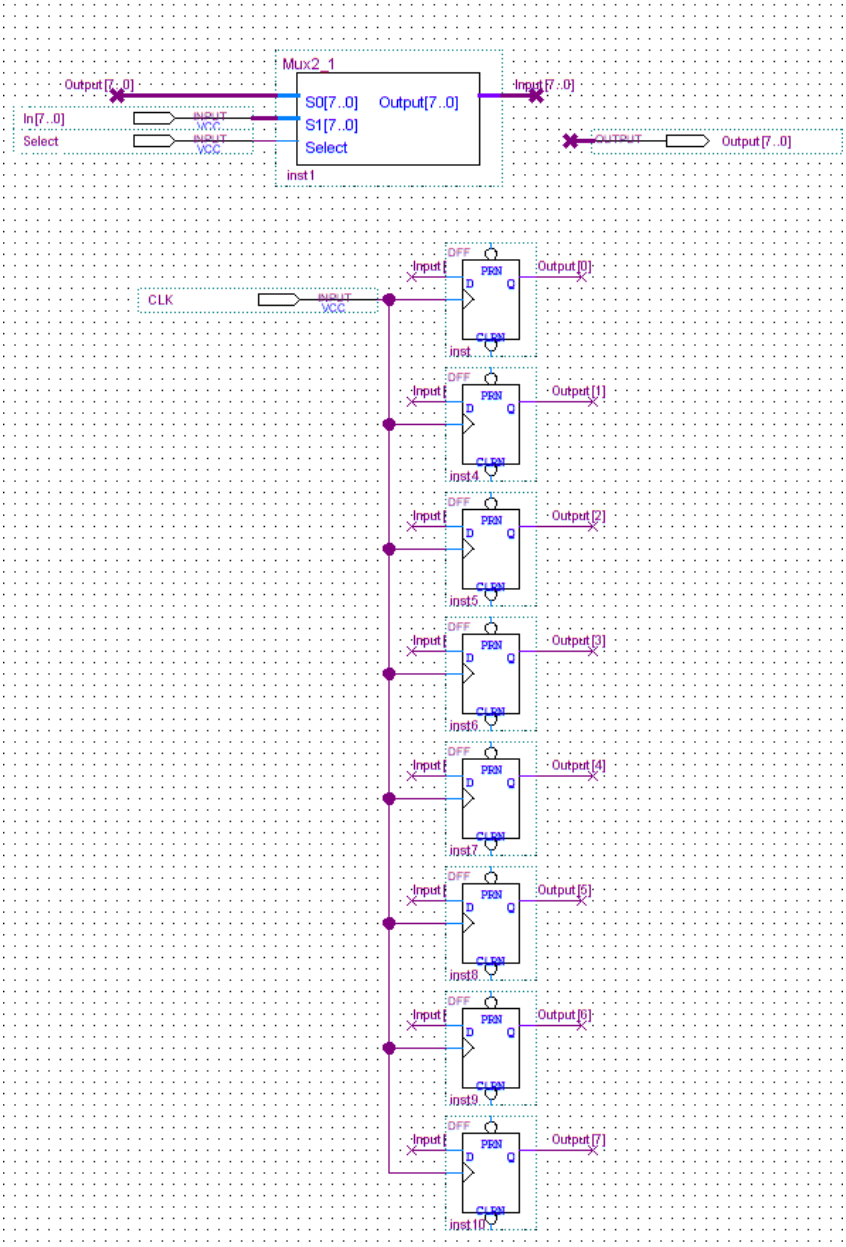
- Opcode[1] connects to "Mux2_1" block to choose result of “Add_sub” block or “Min_max” block.

- If Opcode[1] = 1 then “AU2” will choose the result calculated in the "Min_max" block, otherwise Opcode[1] = 0, “AU2” will choose the result calculated in the "Add_sub" block.

- Connect GND to always calculate *max* and the "Add_sub" block is controlled by Opcode[0], if Opcode[0] = 0 calculates *addition*, Opcode[0] = 1 calculates *subtraction*.

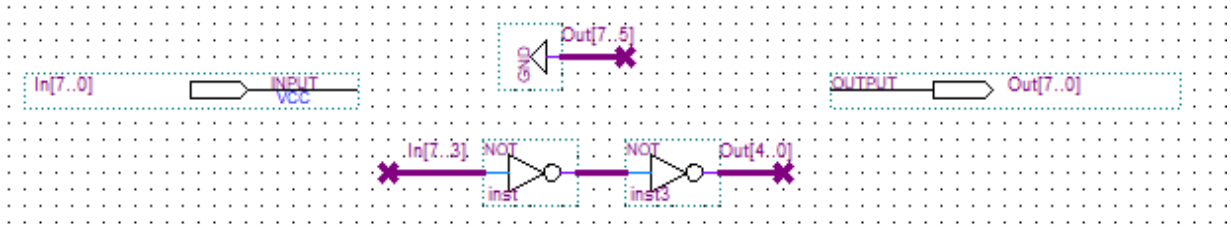
  (“Add_sub” block and “Min_max” block are explained in “AU1”)

- **Register:**



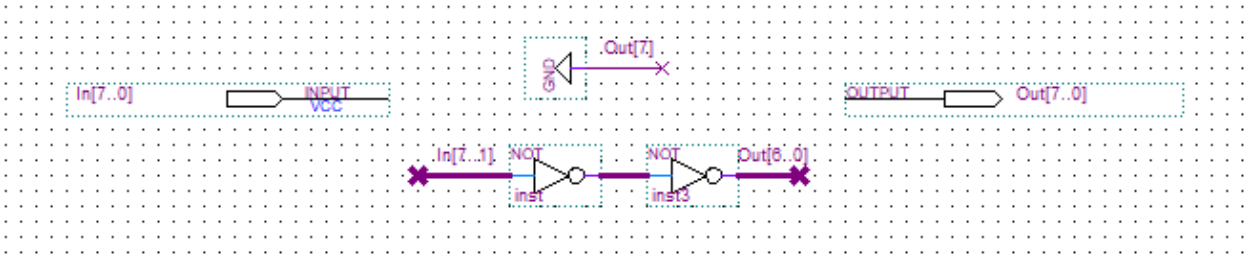
- Register if created by 8 D-FF and “Mux2_1”.
- The input Select decides that the register loads new value or not. If Select = 0 => no load new value and put previous value (S0) into D-FF so it can stores value, otherwise, Select = 1 => load new value (S1).

- **Shift 3 (8 bits):**



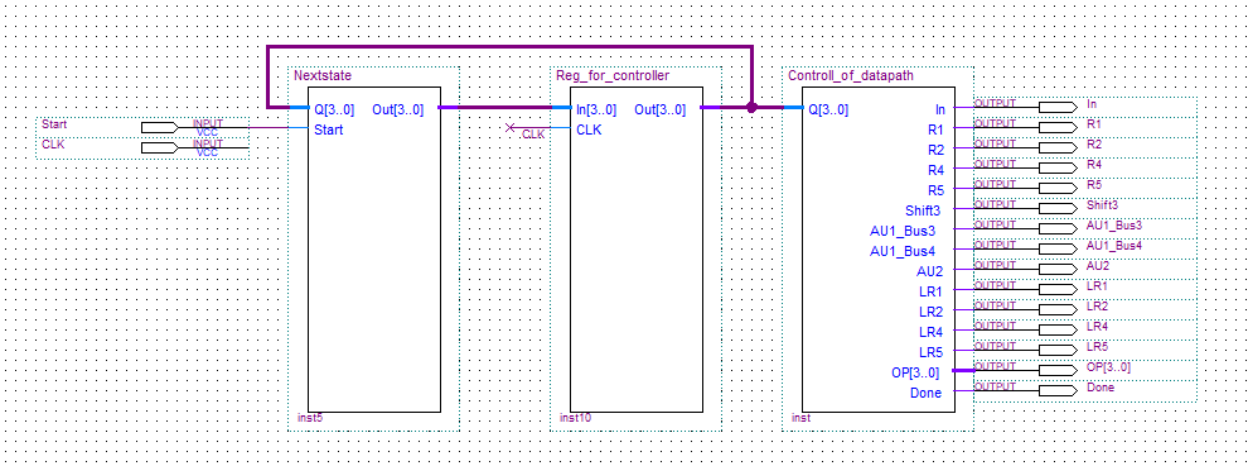
Connect bits [7..3] of input to bits[4..0] of output, remaining bits of output are GND.

- **Shift 1 (8 bits):**



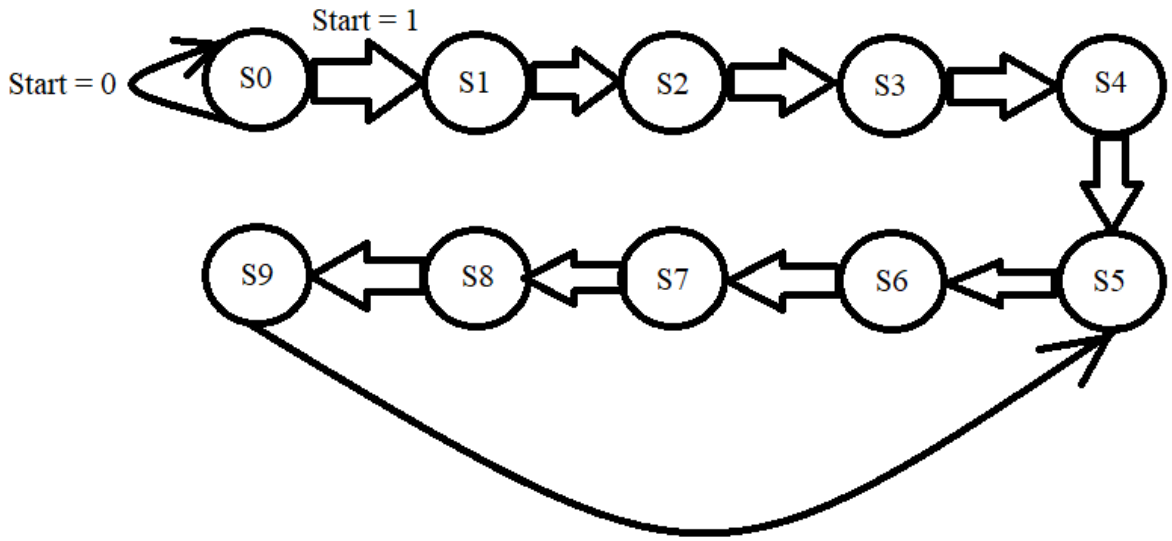
Connect bits [7..1] of input to bits[6..0] of output, remaining bit of output is GND.

2. Controller:



Controller includes “Next_state” block, Register always load new value and a block to decode from state into control signal to control datapath.

- **Next state:**



	IN					OUT				
	Start	Q3	Q2	Q1	Q0	Q3+	Q2+	Q1+	Q0+	
S0	0	0	0	0	0	0	0	0	0	S0
S0	1	0	0	0	0	0	0	0	1	S1
S1	X	0	0	0	1	0	0	1	0	S2
S2	X	0	0	1	0	0	0	1	1	S3
S3	X	0	0	1	1	0	1	0	0	S4
S4	X	0	1	0	0	0	1	0	1	S5
S5	X	0	1	0	1	0	1	1	0	S6
S6	X	0	1	1	0	0	1	1	1	S7
S7	X	0	1	1	1	1	0	0	0	S8
S8	X	1	0	0	0	1	0	0	1	S9
S9	X	1	0	0	1	0	1	0	1	S5
S10	X	1	0	1	0	X	X	X	X	
S11	X	1	0	1	1	X	X	X	X	
S12	X	1	1	0	0	X	X	X	X	
S13	X	1	1	0	1	X	X	X	X	
S14	X	1	1	1	0	X	X	X	X	
S15	X	1	1	1	1	X	X	X	X	

Temporarily removed Start to easy to draw K_map, after that we or(+) Q_{0+} with Start.S_0, because Start = 1 only in case that input(Q3_Q2_Q1_Q0) = 0000 and output(Q3+_Q2+_Q1+_Q0+) = 0001:

Q3+		Q1_Q0			
		00	01	11	10
Q3_Q2	00				
	01			1	
	11				
	10	1			

$Q_{3+} = Q_3'.Q_2.Q_1.Q_0 + Q_3.Q_2'.Q_1'.Q_0'$

Q2+		Q1_Q0			
		00	01	11	10
Q3_Q2	00			1	
	01	1	1		1
	11				
	10		1		

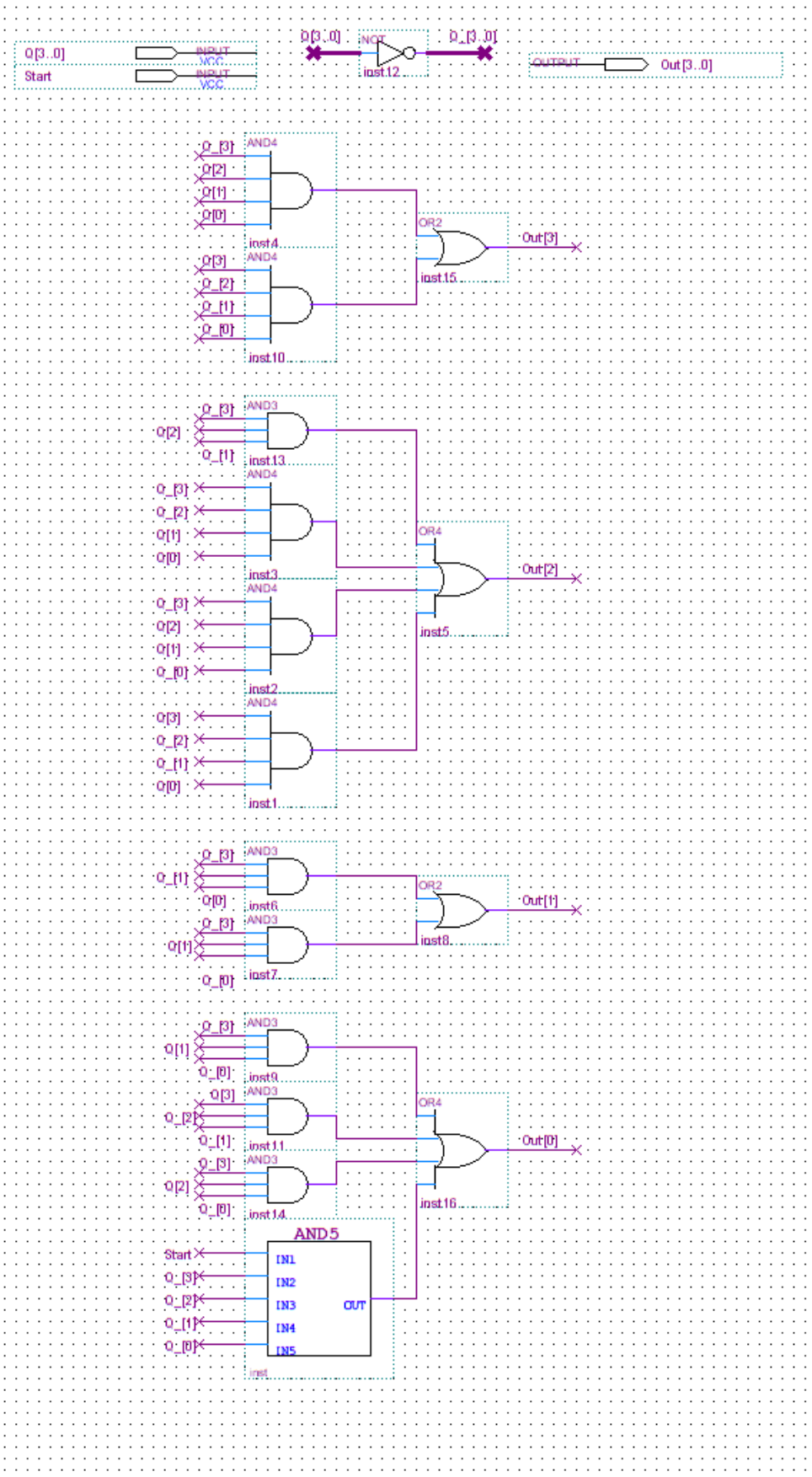
$Q_{2+} = Q_3.Q_2.Q_1' + Q_3'.Q_2'.Q_1.Q_0 + Q_3'.Q_2.Q_1.Q_0'$
$+ Q_3.Q_2'.Q_1'.Q_0$

Q0+		Q1_Q0			
		00	01	11	10
Q3_Q2	00				1
	01	1			1
	11				
	10	1	1		

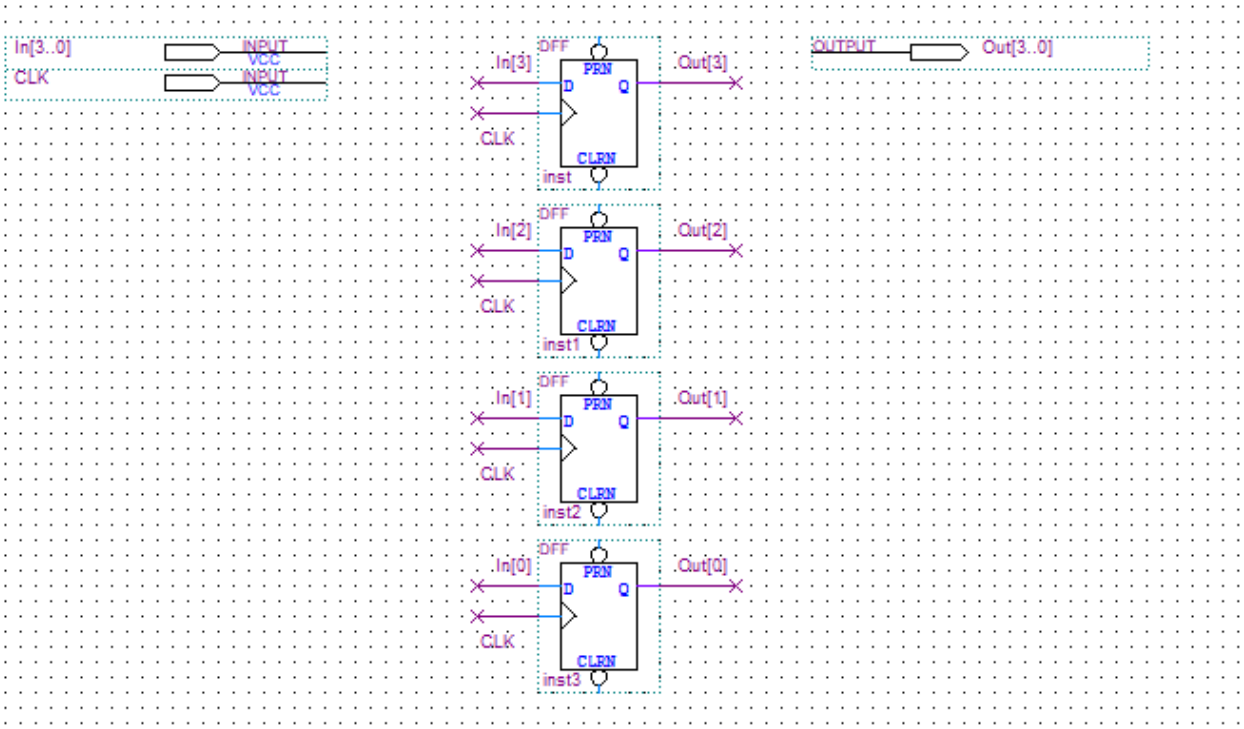
$Q_{1+} = Q_3.Q_1'.Q_0 + Q_3'.Q_1.Q_0'$

Q1+		Q1_Q0			
		00	01	11	10
Q3_Q2	00		1		1
	01		1		1
	11				
	10				

$Q_{0+} = Q_3'.Q_1.Q_0 + Q_3.Q_2'.Q_1' + Q_3'.Q_2.\ Q_0'$
$+ \text{Start}.\ Q_3'.Q_2'.Q_1'.Q_0'$


Q[3..0]
INPUT
VCC
Start
INPUT
VCC
Q[3..0]
NOT
inst12
Q_[3..0]
OUTPUT
Out[3..0]
AND4
Q_[3]
Q[2]
Q[1]
Q[0]
inst4
AND4
Q[3]
Q_[2]
Q_[1]
Q_[0]
inst10
OR2
Out[3]
inst15
AND3
Q_[3]
Q[2]
Q_[1]
inst13
AND4
Q_[3]
Q_[2]
Q[1]
Q[0]
inst3
AND4
Q_[3]
Q[2]
Q[1]
Q_[0]
inst2
AND4
Q[3]
Q_[2]
Q_[1]
Q[0]
inst1
OR4
Out[2]
inst5
AND3
Q_[3]
Q_[1]
Q[0]
inst6
AND3
Q_[3]
Q[1]
Q_[0]
inst7
OR2
Out[1]
inst8
AND3
Q_[3]
Q[1]
Q_[0]
inst9
AND3
Q[3]
Q_[2]
Q_[1]
inst11
AND3
Q_[3]
Q[2]
Q_[0]
inst14
AND5
Start
Q_[3]
Q_[2]
Q_[1]
Q_[0]
IN1
IN2
IN3
IN4
IN5
OUT
inst
OR4
Out[0]
inst16

- **Register always load:**



Using this register to sync with CLK.

- **Decoder from state to control:**

STATE	IN				OUT																	
	Q3	Q2	Q1	Q0	I1B3	I2B4	R1B2	R2B2	AU1B3	AU1B4	S3B7	R4B5	R5B5	AU2B7	LR1	LR2	LR4	LR5	OP[3]	OP[2]	OP[1]	OP[0]	DONE	
S0	0	0	0	0	1	1									1	1			X	X	X	X		
S1	0	0	0	1			1		1						1				X	X	1	X		
S2	0	0	1	0				1		1						1			X	X	1	X		nth pair
S3	0	0	1	1				1										1	X	X	0	1		
S4	0	1	0	0				1			1						1		X	X	0	0		
S5	0	1	0	1	1	1		1				1		1	1	1			0	1	X	X		
S6	0	1	1	0			1		1				1	1	1				0	0	1	X		
S7	0	1	1	1				1		1		1		1		1			1	X	1	X		(n+1)th pair
S8	1	0	0	0				1										1	X	X	0	1	1	
S9	1	0	0	1				1			1						1		X	X	0	0		
S10	1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S11	1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S12	1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S13	1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S14	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
S15	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

I1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01		1		
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

I1B3 = $Q_3'.Q_2'.Q_1'.Q_0' + Q_2.Q_1'.Q_0$

I2B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01		1		
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

I2B4 = $Q_3'.Q_2'.Q_1'.Q_0' + Q_2.Q_1'.Q_0$

R1B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00		1		
Q3_Q2 01				1
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

R1 = $Q_3'.Q_2'.Q_1'.Q_0 + Q_2.Q_1.Q_0'$

R2B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00			1	1
Q3_Q2 01	1	1	1	
Q3_Q2 11	X	X	X	X
Q3_Q2 10	1	1	X	X

R2B2 = $Q_3 + Q_1.Q_0 + Q_2 \oplus Q_1$

AU1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00		1		
Q3_Q2 01				1
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

AU1B3 = $Q_3'.Q_2'.Q_1'.Q_0 + Q_2.Q_1.Q_0'$

AU1B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00				1
Q3_Q2 01			1	
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

AU1B4 = $Q_2.Q_1.Q_0 + Q_2'.Q_1.Q_0'$

S3B7	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01	1			
Q3_Q2 11	X	X	X	X
Q3_Q2 10		1	X	X

S3B7 = $Q_2.Q_1'.Q_0' + Q_3.Q_1'.Q_0$

R4B5	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01		1	1	
Q3_Q2 11	X	X	X	X
Q3_Q2 10			X	X

R4B5 = $Q_3'.Q_2.Q_0$

R5B5	Q1_Q0			
Q3_Q2	00	01	11	10
00				
01				1
11	X	X	X	X
10			X	X

R5B5 = $Q_2.Q_1.Q_0'$

AU2B7	Q1_Q0			
Q3_Q2	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10			X	X

AU2B7 = $Q_3'.Q_2.Q_1 + Q_3'.Q_2.Q_0$

LR1	Q1_Q0			
Q3_Q2	00	01	11	10
00	1	1		
01		1		1
11	X	X	X	X
10			X	X

LR1 = $Q_3'.Q_2'.Q_1' + Q_2.Q_1'.Q_0 + Q_2.Q_1.Q_0'$

LR2	Q1_Q0			
Q3_Q2	00	01	11	10
00	1			1
01		1	1	
11	X	X	X	X
10			X	X

LR2 = $Q_3'.Q_2'.Q_0' + Q_2.Q_0$

LR4	Q1_Q0			
Q3_Q2	00	01	11	10
00				
01	1			
11	X	X	X	X
10		1	X	X

LR4 = $Q_2.Q_1'.Q_0' + Q_3.Q_0$

LR5	Q1_Q0			
Q3_Q2	00	01	11	10
00			1	
01				
11	X	X	X	X
10	1		X	X

LR5 = $Q_3.Q_1'.Q_0' + Q_2'.Q_1.Q_0$

OP[3]	Q1_Q0			
Q3_Q2	00	01	11	10
00	X	X	X	X
01	X	0	1	0
11	X	X	X	X
10	X	X	X	X

OP[3] = $Q_1.Q_0$

OP[2]	Q1_Q0			
Q3_Q2	00	01	11	10
00	X	X	X	X
01	X	1	X	0
11	X	X	X	X
10	X	X	X	X

OP[2] = Q_0

OP[1]	Q1_Q0			
Q3_Q2	00	01	11	10
00	X	1	0	1
01	0	X	1	1
11	X	X	X	X
10	0	0	X	X

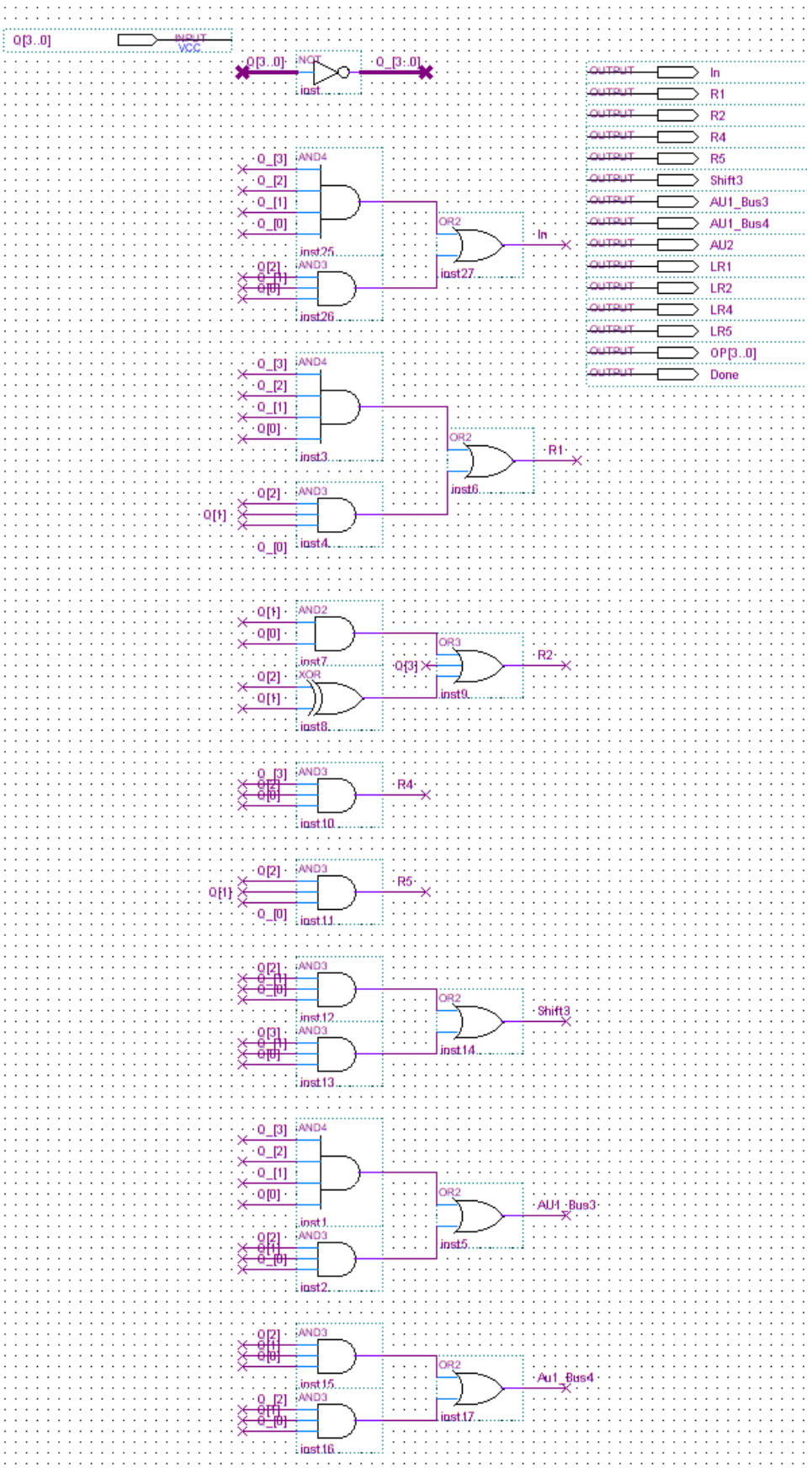
OP[1] = $Q_3'.Q_2'.Q_1' + Q_2.Q_0 + Q_1.Q_0'$

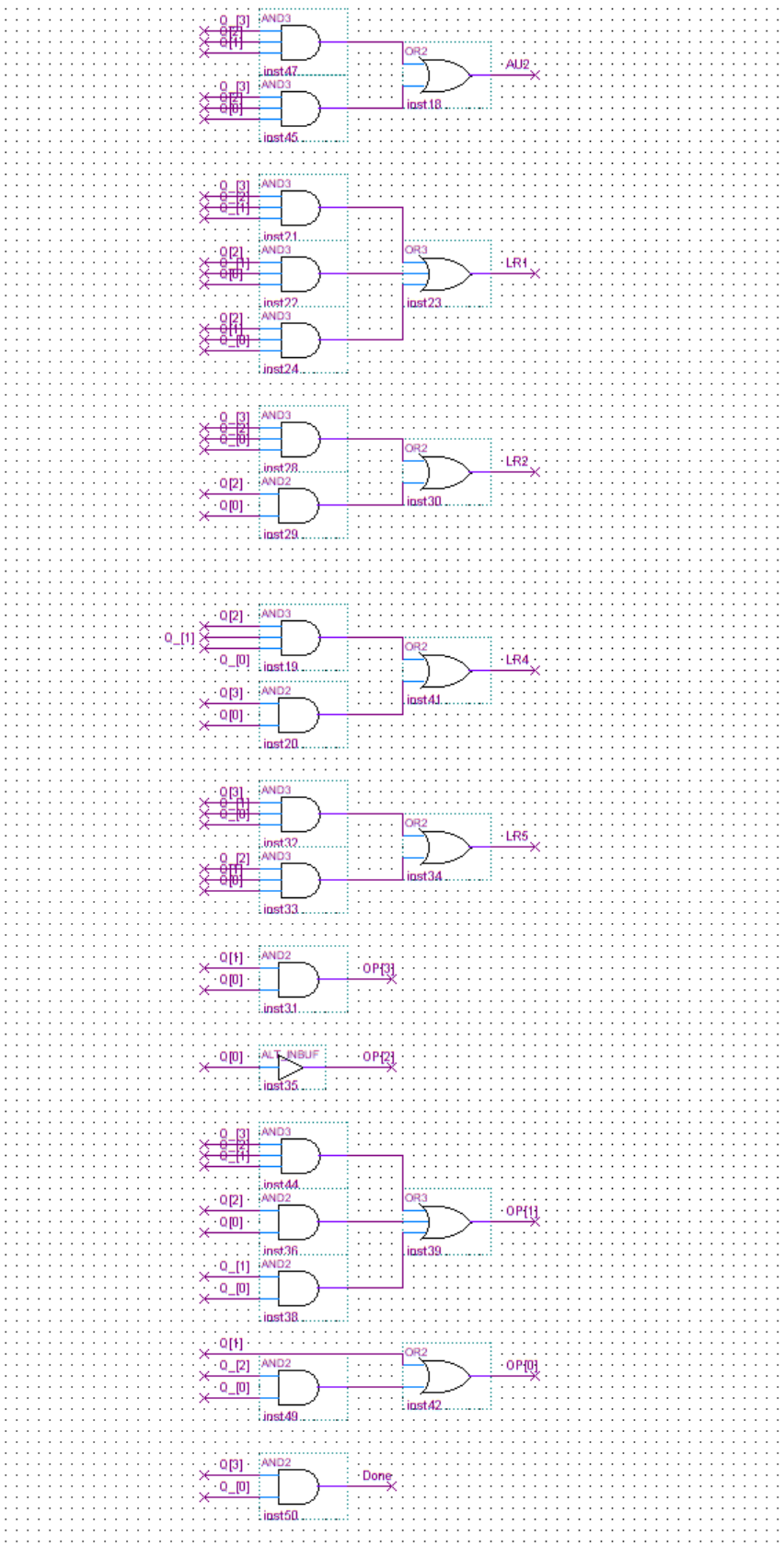
OP[0]	Q1_Q0			
Q3_Q2	00	01	11	10
00	X	X	1	X
01	0	X	X	X
11	X	X	X	X
10	1	0	X	X

OP[0] = $Q_1 + Q_2'.Q_0'$

DONE	Q1_Q0			
Q3_Q2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	0	X	X

DONE = $Q_3.Q_0'$

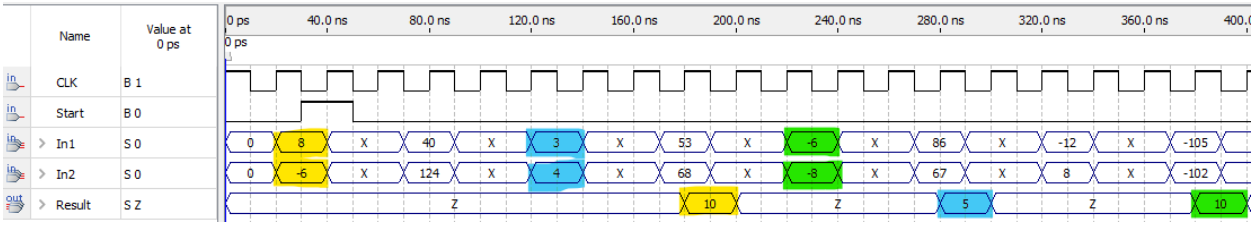

Q[3..0]
INPUT
VCC
Q[3..0]
NOT
inst
Q_[3..0]
OUTPUT
In
R1
R2
R4
R5
Shift3
AU1_Bus3
AU1_Bus4
AU2
LR1
LR2
LR4
LR5
OP[3..0]
Done
Q_[3]
Q_[2]
Q_[1]
Q_[0]
AND4
inst25
AND3
inst26
OR2
inst27
In
Q[1]
inst3
inst6
R1
inst4
inst7
inst9
R2
XOR
inst8
inst10
R4
inst11
R5
inst12
inst14
Shift3
inst13
inst1
inst5
AU1_Bus3
inst2
inst15
inst17
Au1_Bus4
inst16


AND3
inst47
O_[3]
O_[2]
O_[1]
AND3
inst45
O_[3]
O_[2]
O_[0]
OR2
inst18
AU2
AND3
inst21
O_[3]
O_[2]
O_[1]
AND3
inst22
O_[2]
O_[1]
O_[0]
AND3
inst24
O_[2]
O_[1]
O_[0]
OR3
inst23
LR1
AND3
inst28
O_[3]
O_[2]
O_[0]
AND2
inst29
O_[2]
O_[0]
OR2
inst30
LR2
AND3
inst19
O_[2]
O_[1]
O_[0]
AND2
inst20
O_[3]
O_[0]
OR2
inst41
LR4
AND3
inst32
O_[3]
O_[1]
O_[0]
AND3
inst33
O_[2]
O_[1]
O_[0]
OR2
inst34
LR5
AND2
inst31
O_[1]
O_[0]
OP[3]
ALT_INBUF
inst35
O_[0]
OP[2]
AND3
inst44
O_[3]
O_[2]
O_[1]
AND2
inst36
O_[2]
O_[0]
AND2
inst38
O_[1]
O_[0]
OR3
inst39
OP[1]
AND2
inst49
O_[1]
O_[2]
O_[0]
OR2
inst42
OP[0]
AND2
inst50
O_[3]
O_[0]
Done

Note (according to datapath):

- I1B3: control tri allows transferring data from Input1 to Bus_3.
- AU1B3: control tri allows transferring data from AU1 to Bus_3.
  (in the same time only Input1 or AU1 load data to Bus_3)
- R1B2: control tri allows loading data from Register1 to Bus_2.
- R2B2: control tri allows loading data from Register2 to Bus_2.
  (in the same time only register1 or register2 load data to Bus_2)
- I2B4: control tri allows transferring data from Input2 to Bus_4.
- AU1B4: control tri allows transferring data from AU1 to Bus_4.
  (in the same time only Input2 or AU1 load data to Bus_4)
- S3B7: control tri allows transferring data from Shift3 to Bus_7.
- AU2B7: control tri allows transferring data from AU2 to Bus_7.
  (in the same time only Shift3 or AU2 load data to Bus_7)
- R4B5: control tri allows transferring data from Register4 to Bus_5.
- R5B5: control tri allows transferring data from Register5 to Bus_5.
  (in the same time only Register4 or Register5 load data to Bus_5)
- LR1: Allows to load data to Register1.
- LR2: Allows to load data to Register2.
- LR3: Allows to load data to Register3.
- LR4: Allows to load data to Register4.
- LR5: Allows to load data to Register5.
- OP[3..0]: Opcode (OP[3..2] control AU2, OP[1..0] control AU1).
- Done: when state is S8 (after successfully calculated final result we will have the output is result, in other states output is “X”)

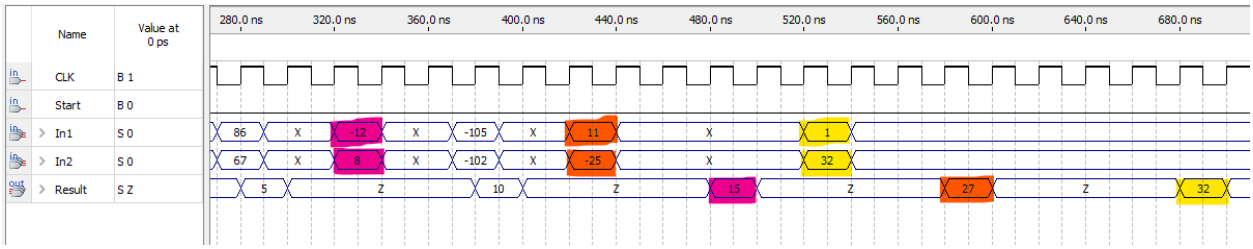
1. Waveform (input and output according to color respectively):




```
a = 8
b = -6
t1 = |a| = 8
t2 = |b| = 6
t4 = min(t1, t2)>>1 = 3
x = max(t1, t2) = 8
t3 = max(t1, t2)>>3 = 1
t5 = x - t3 = 7
t6 = t4 + t5 = 10
t7 = max(t6, x) = 10
Result = t7 = 10
```

```
a = 3
b = 4
t1 = |a| = 3
t2 = |b| = 4
t4 = min(t1, t2)>>1 = 1
x = max(t1, t2) = 4
t3 = max(t1, t2)>>3 = 0
t5 = x - t3 = 4
t6 = t4 + t5 = 5
t7 = max(t6, x) = 5
Result = t7 = 5
```

```
a = -6
b = -8
t1 = |a| = 6
t2 = |b| = 8
t4 = min(t1, t2)>>1 = 3
x = max(t1, t2) = 8
t3 = max(t1, t2)>>3 = 1
t5 = x - t3 = 7
t6 = t4 + t5 = 10
t7 = max(t6, x) = 10
Result = t7 = 10
```

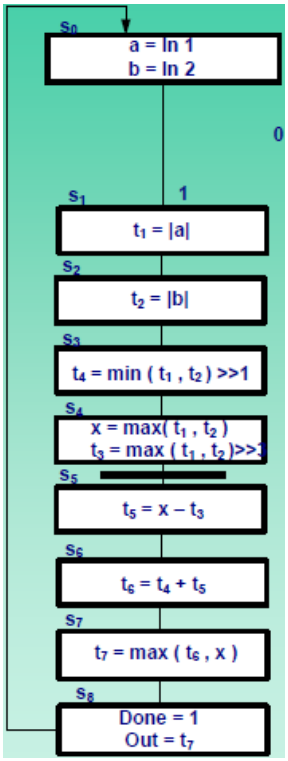



```
a = -12
b = 8
t1 = |a| = 12
t2 = |b| = 8
t4 = min(t1, t2)>>1 = 4
x = max(t1, t2) = 12
t3 = max(t1, t2)>>3 = 1
t5 = x - t3 = 11
t6 = t4 + t5 = 15
t7 = max(t6, x) = 15
Result = t7 = 15
```

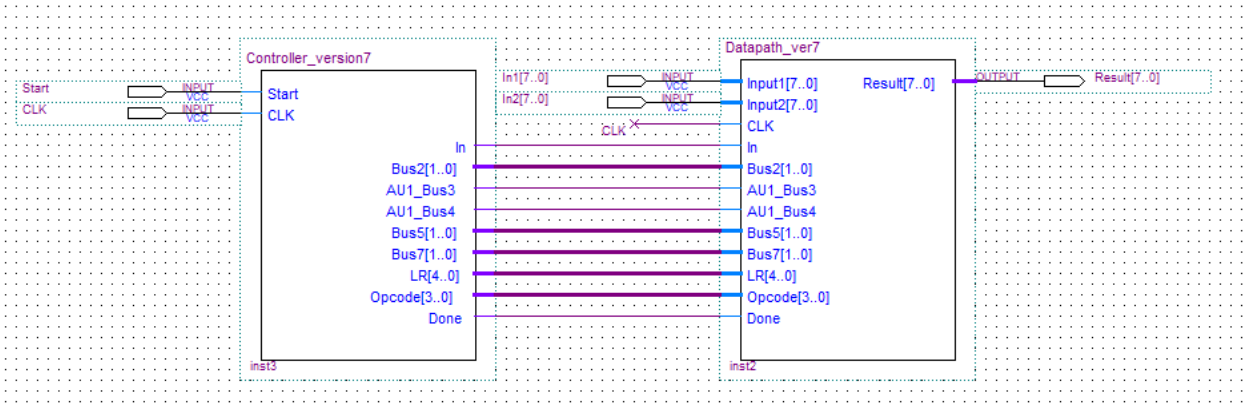
```
a = 11
b = -25
t1 = |a| = 11
t2 = |b| = 25
t4 = min(t1, t2)>>1 = 5
x = max(t1, t2) = 25
t3 = max(t1, t2)>>3 = 3
t5 = x - t3 = 22
t6 = t4 + t5 = 27
t7 = max(t6, x) = 27
Result = t7 = 27
```

```
a = 1
b = 32
t1 = |a| = 1
t2 = |b| = 32
t4 = min(t1, t2)>>1 = 0
x = max(t1, t2) = 32
t3 = max(t1, t2)>>3 = 4
t5 = x - t3 = 28
t6 = t4 + t5 = 28
t7 = max(t6, x) = 32
Result = t7 = 32
```

VII. Version 7:

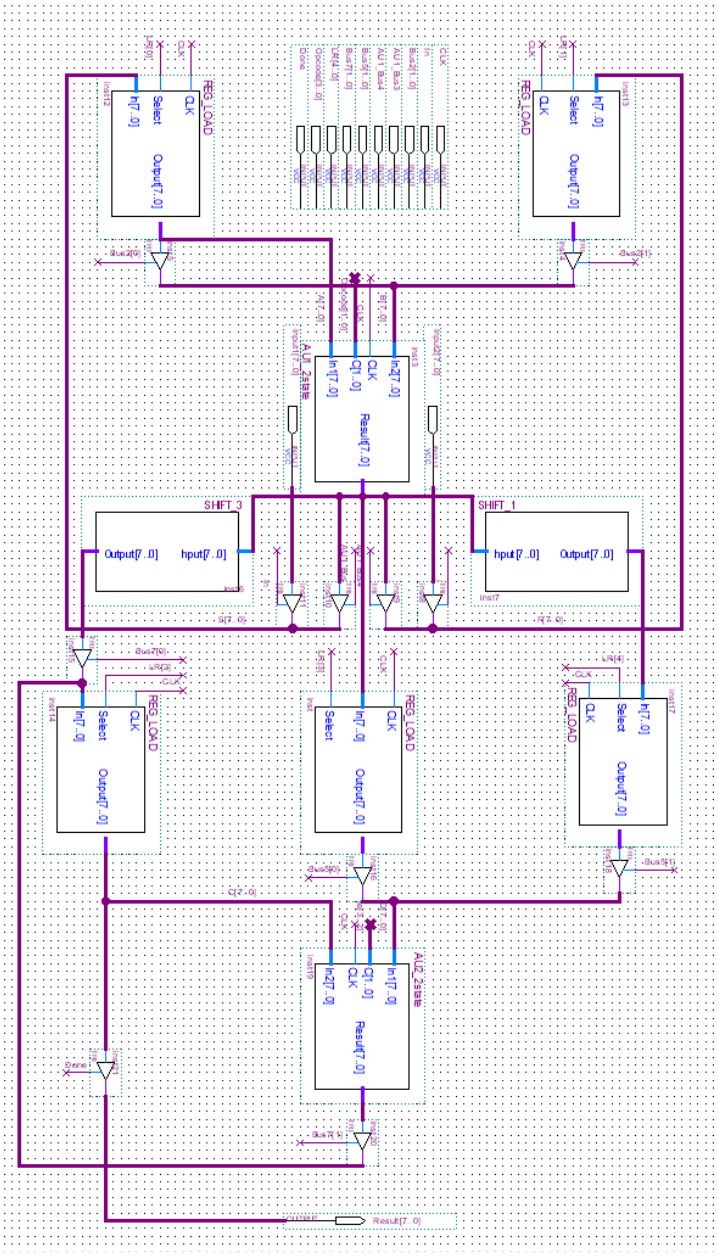


Circuit (includes controller and datapath):



Version 7 is combination of version 5 and version 6 (Functional logic pineline + Datapath pineline).

1. Datapath



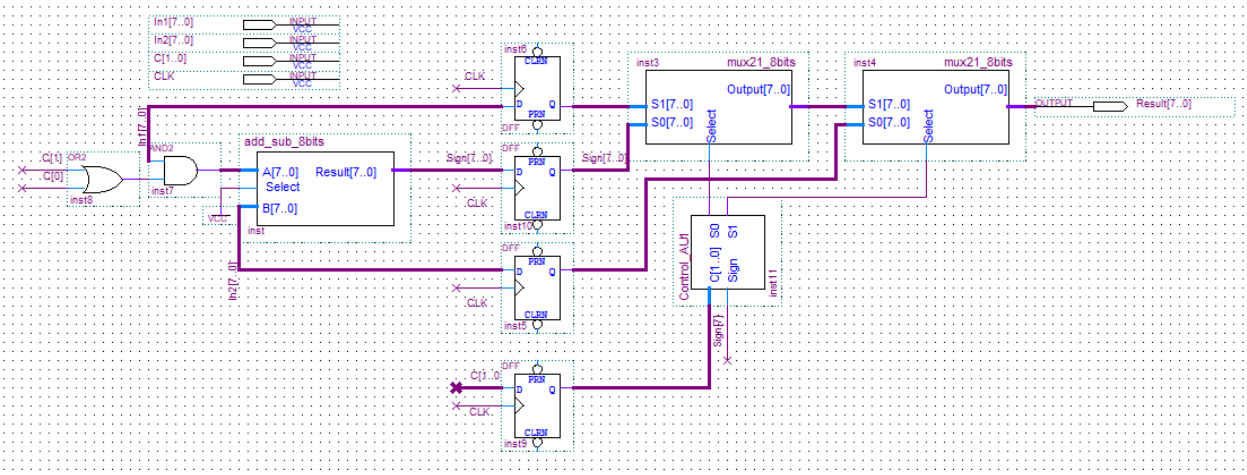
- Datapath in version7 use AU1 block (ABS/Min/Max) 2 state and AU2 Block (Max/add/Sub) 2 state (Functional unit pineline + datapath pineline)
    - The datapath has 22 inputs and 1 output, of which 2 inputs (8 bits) transmit the number to be calculated, the remaining 20 inputs are control signals (opcodes of AU blocks, read/write signals of registers, tries, CLK), the output for the final result.

- The datapath consists of 5 registers and 2 AU blocks, “Shift3” block and “Shift1” block. “AU1” and “AU2” are controlled by 4 bit opcodes (Opcode[3..2] control AU2, Opcode[1..0] control AU1), the registers store the following values, “AU1” and “AU2” perform the following funtions:

  R1 = [a, t1] R3 = [t3, t5, t6, t7]
  R2 = [b, t2] R4 = [x]
  AU1 = [abs/min/max] R5 = [t4]
  AU2 = [+/-/max]

(About AU, we divide each calculation into 2 states to reduce the time each state)

- **AU1 (is used to calculating abs/min/max and controlled by Opcode[1..0]):**



AU1	
Opcode	Function
00	abs
01	max
10	min
11	min

- Use 4 D-FF to devide each calculation into 2 states, the first D-FF save Input1, the 2nd D-FF save result of “Add_sub” Block, the highest bit of it is sign bit, the 3rd save Input2 and the last D-FF save control signal (C[1..0]).

- Connect Select of “Add_sub” Block to alway calculate A-B.

- In A[7..0] of “Add_sub” Block, we connect with ((C[1] + C[0]).In1), so only C[1] = C[0] = 0 => A[7..0] = 0. Then result of “Add_sub” Block is 0-B, we use Result[7] as a sign bit to control 2 Mux Block following.

- “Control” Block:

	Control AU1						Output of AU1
	Input			Output			
	Sign	C[1]	C[0]	S1	S0	Funtion	
A>B	0	0	0	1	0	abs	Result
	0	0	1	1	1	max	A
	0	1	0	0	x	min	B
	0	1	1	0	x	min	B
A<B	1	0	0	0	x	abs	B
	1	0	1	0	x	max	B
	1	1	0	1	1	min	A
	1	1	1	1	1	min	A

-

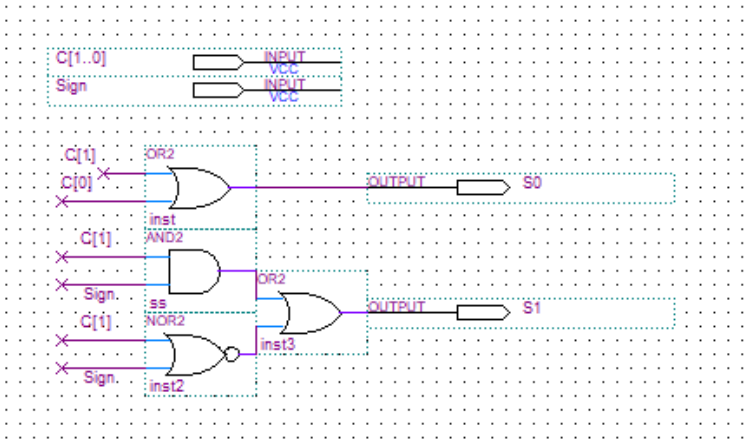
S1		C[1..0]			
		00	01	11	10
Sign	0	1	1		
	1			1	1

Mux	
Select	Result
0	S0
1	S1

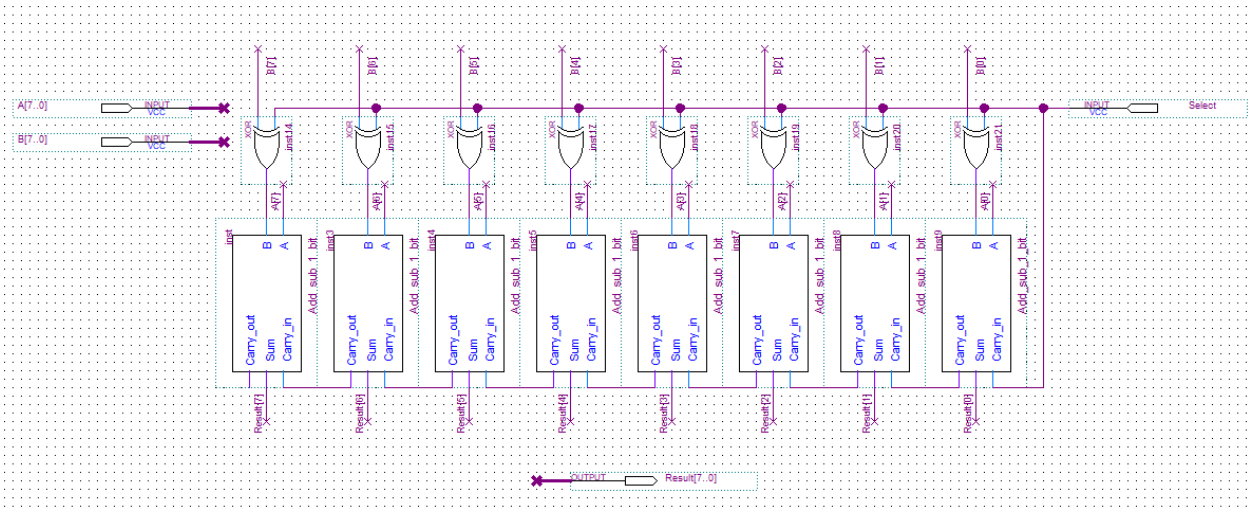
S0		C[1..0]			
		00	01	11	10
Sign	0	0	1	x	x
	1	x	x	1	1

S1 = C[1] + C[0]

S0 = Sign.C[1] + Sign’.C[0]’



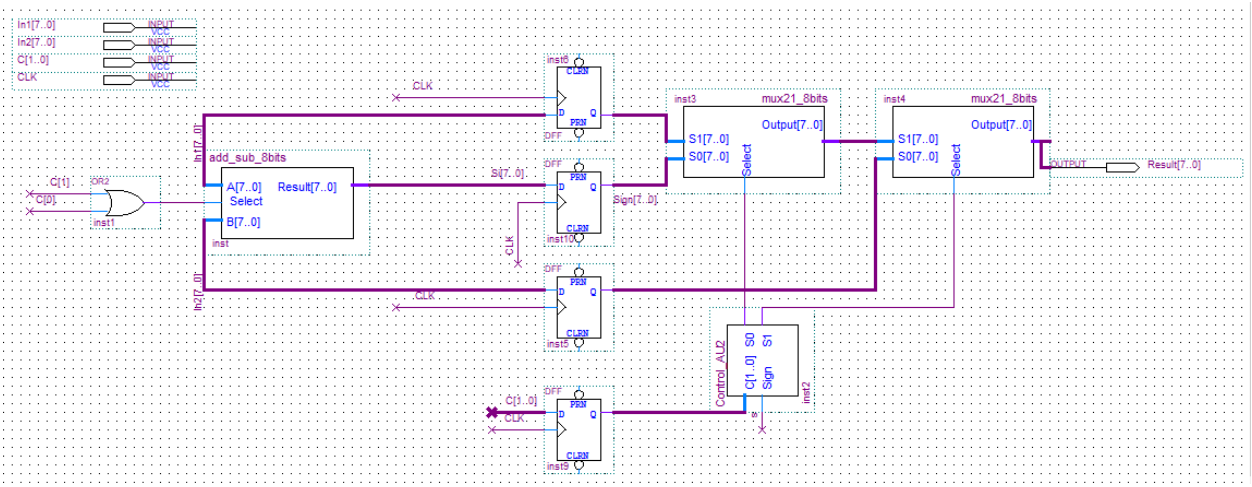
Add_sub:



- **AU2 (is used to calculating add/sub/max and controlled by**

Select	Function
0	Add
1	Sub

Opcode[3..2]:



AU2	
Opcode	Function
00	Add
01	Sub
10	Max
11	Sub

- The same as AU1, only different “Control” block and the Select of “Add_sub” Block:

- “Control” Block:

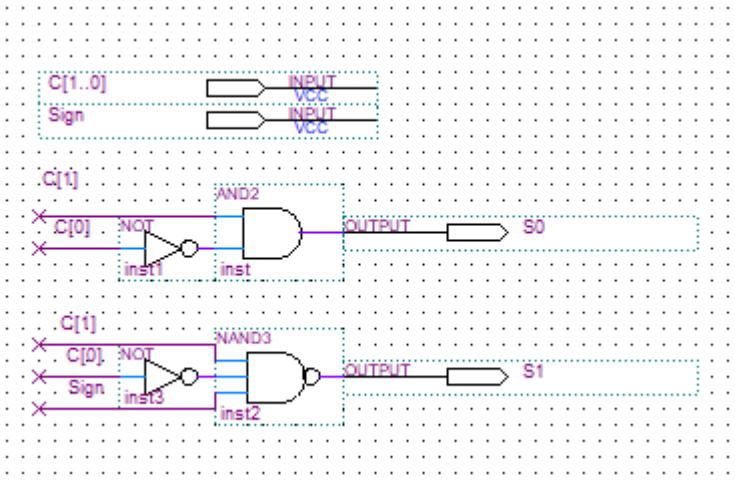
	Control AU1							Output of AU1
	Input				Output			
	Sign	C[1]	C[0]		S1	S0	Select	Funtion	
A>B	0	0	0		1	0	0	Add	Result of Add_sub
	0	0	1		1	0	1	Sub	Result of Add_sub
	0	1	0		1	1	1	Max	A
	0	1	1		1	0	1	Sub	Result of Add_sub
A<B	1	0	0		1	0	0	Add	Result of Add_sub
	1	0	1		1	0	1	Sub	Result of Add_sub
	1	1	0		0	x	1	Max	B
	1	1	1		1	0	1	Sub	Result of Add_sub

S1		C[1..0]			
		00	01	11	10
Sign	0	1	1	1	1
	1	1	1	0	1

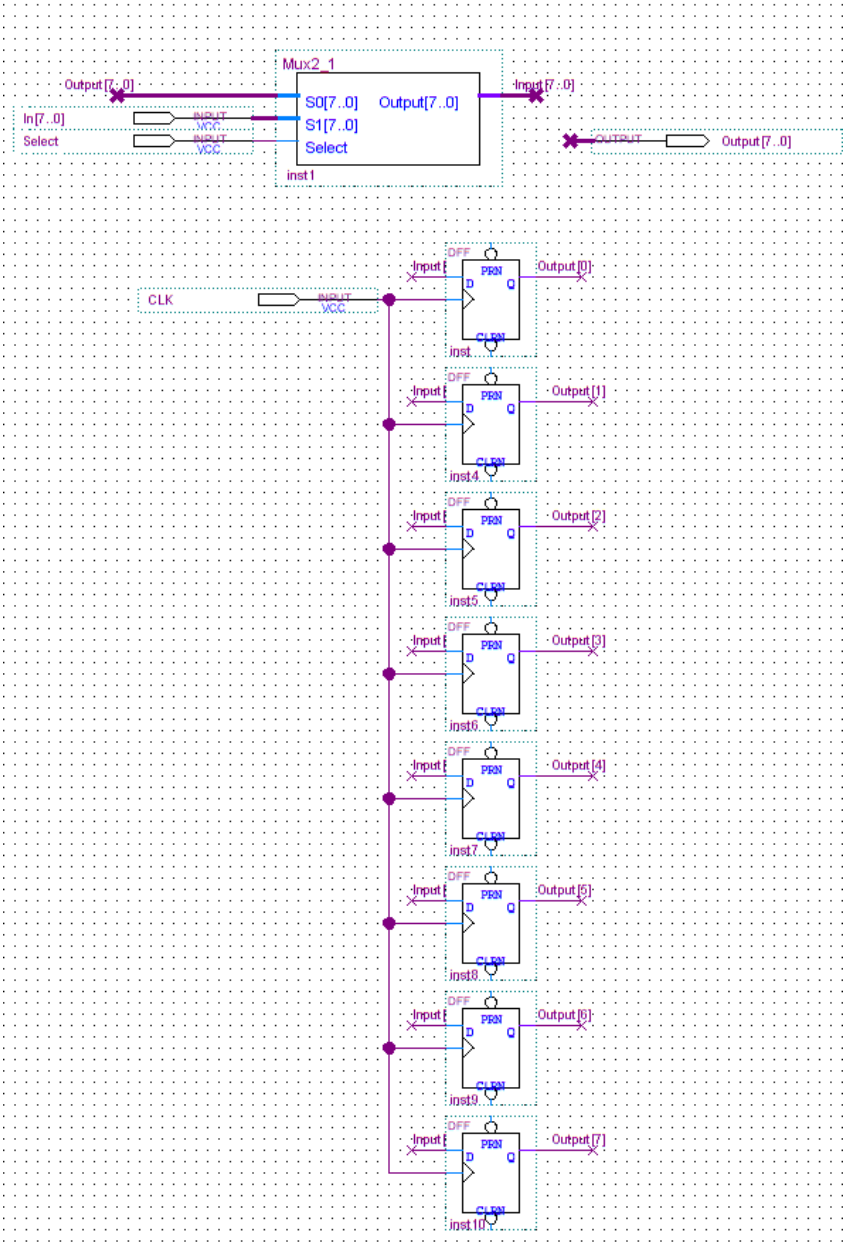
Select		C[1..0]			
		00	01	11	10
Sign	0	0	1	1	1
	1	0	1	1	1

S0		C[1..0]			
		00	01	11	10
Sign	0	0	0	0	1
	1	0	0	0	x

S1 = (C[1].C[0]’.Sign)’ Select = C[1] + C[0] S0 = C[1].C[0]’

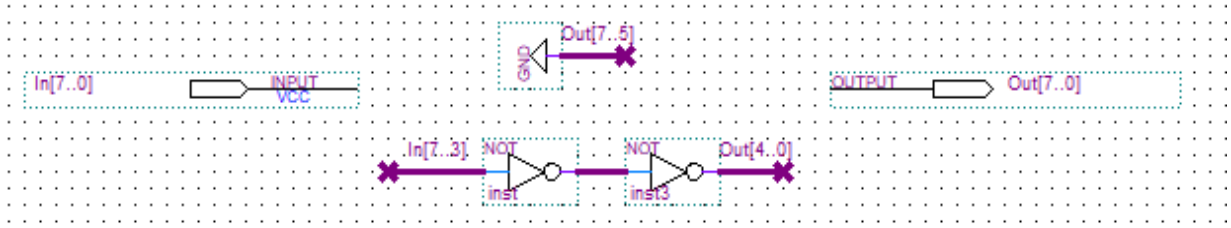


- **Register:**



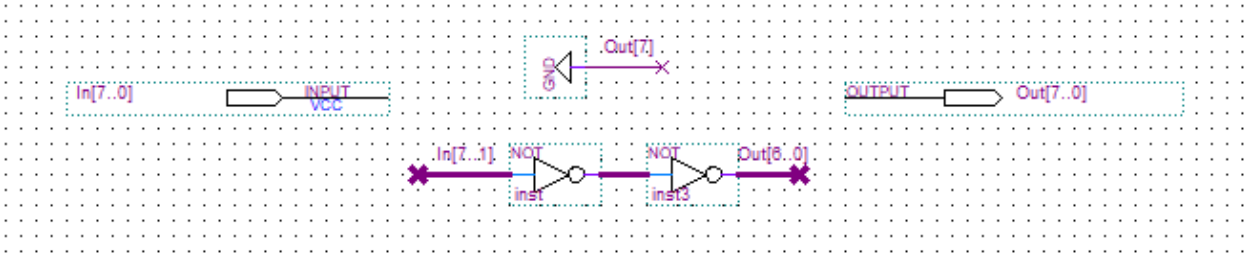
- Register if created by 8 D-FF and “Mux2_1”.
- The input Select decides that the register loads new value or not. If Select = 0 => no load new value and put previous value (S0) into D-FF so it can stores value, otherwise, Select = 1 => load new value (S1).

- **Shift 3 (8 bits):**



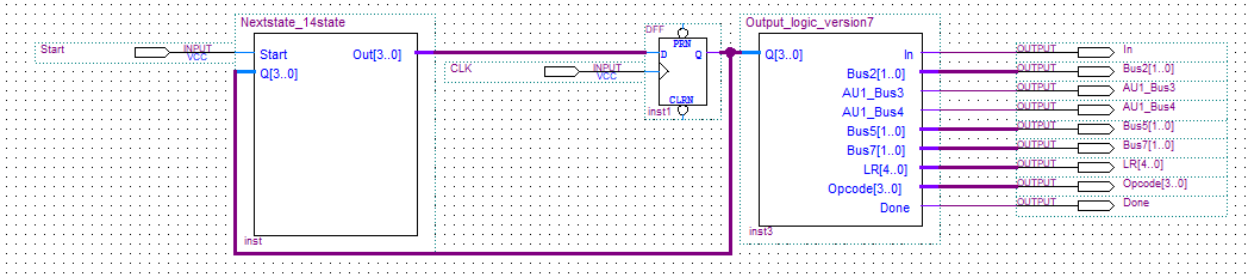
Connect bits [7..3] of input to bits[4..0] of output, remaining bits of output are GND.

- **Shift 1 (8 bits):**



Connect bits [7..1] of input to bits[6..0] of output, remaining bit of output is GND.

2. Controller



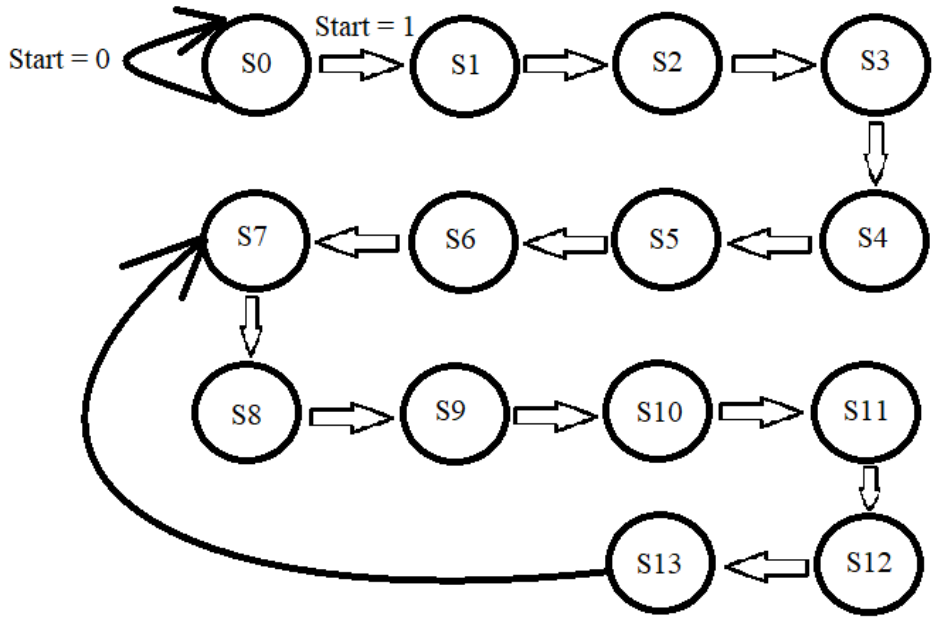
Controller includes “Next_state” block, Register always load new value and a block to decode from state into control signal to control datapath.

Note (according to datapath):

- I1B3: control tri allows transferring data from Input1 to Bus_3.
- AU1B3: control tri allows transferring data from AU1 to Bus_3.
  (in the same time only Input1 or AU1 load data to Bus_3)
- R1B2 (Bus2[0]): control tri allows loading data from Register1 to Bus_2.

- R2B2 (Bus2[1]): control tri allows loading data from Register2 to Bus_2.
  (in the same time only register1 or register2 load data to Bus_2)
- I2B4: control tri allows transferring data from Input2 to Bus_4.
- AU1B4: control tri allows transferring data from AU1 to Bus_4.
  (in the same time only Input2 or AU1 load data to Bus_4)
- S3B7: control tri allows transferring data from Shift3 to Bus_7.
- AU2B7: control tri allows transferring data from AU2 to Bus_7.
  (in the same time only Shift3 or AU2 load data to Bus_7)
- R4B5 (Bus5[0]): control tri allows transferring data from Register4 to Bus_5.
- R5B5 (Bus5[1]): control tri allows transferring data from Register5 to Bus_5.
  (in the same time only Register4 or Register5 load data to Bus_5)
- LR1: Allows to load data to Register1.
- LR2: Allows to load data to Register2.
- LR3: Allows to load data to Register3.
- LR4: Allows to load data to Register4.
- LR5: Allows to load data to Register5.
  (LR5-LR1 corresponding to LR[4..0])
- Opcode[3..0]: OP[3..2] control AU2, OP[1..0] control AU1.
- Done: when state is S8 (after successfully calculated final result we will have the output is result, in other states output is "X")

- **Next state:**



STATE	IN					OUT			
	Start	Q3	Q2	Q1	Q0	Q3+	Q2+	Q1+	Q0+
S0	0	0	0	0	0	0	0	0	0
S0	1	0	0	0	0	0	0	0	1
S1	X	0	0	0	1	0	0	1	0
S2	X	0	0	1	0	0	0	1	1
S3	X	0	0	1	1	0	1	0	0
S4	X	0	1	0	0	0	1	0	1
S5	X	0	1	0	1	0	1	1	0
S6	X	0	1	1	0	0	1	1	1
S7	X	0	1	1	1	1	0	0	0
S8	X	1	0	0	0	1	0	0	1
S9	X	1	0	0	1	1	0	1	0
S10	X	1	0	1	0	1	0	1	1
S11	X	1	0	1	1	1	1	0	0
S12	X	1	1	0	0	1	1	0	1
S13	X	1	1	0	1	0	1	1	1
S14	X	1	1	1	0	X	X	X	X
S15	X	1	1	1	1	X	X	X	X

Temporarily removed Start to easy to draw K_map, after that we or(+) Q_{0+} with Start.S_0, because Start = 1 only in case that input(Q3_Q2_Q1_Q0) = 0000 and output(Q3+_Q2+_Q1+_Q0+) = 0001:

Q3+		Q1_Q0			
		00	01	11	10
Q3_Q2	00				
	01			1	
	11	1		X	X
	10	1	1	1	1

$Q_{3+} = Q_3.Q_2' + Q_3.Q_0' + Q_2.Q_1.Q_0$

Q2+		Q1_Q0			
		00	01	11	10
Q3_Q2	00			1	
	01	1	1		1
	11	1	1	X	X
	10			1	

$Q_{2+} = (Q_2+Q_1) \times (Q_2+Q_0) \times (Q_2.Q_1.Q_0)'$

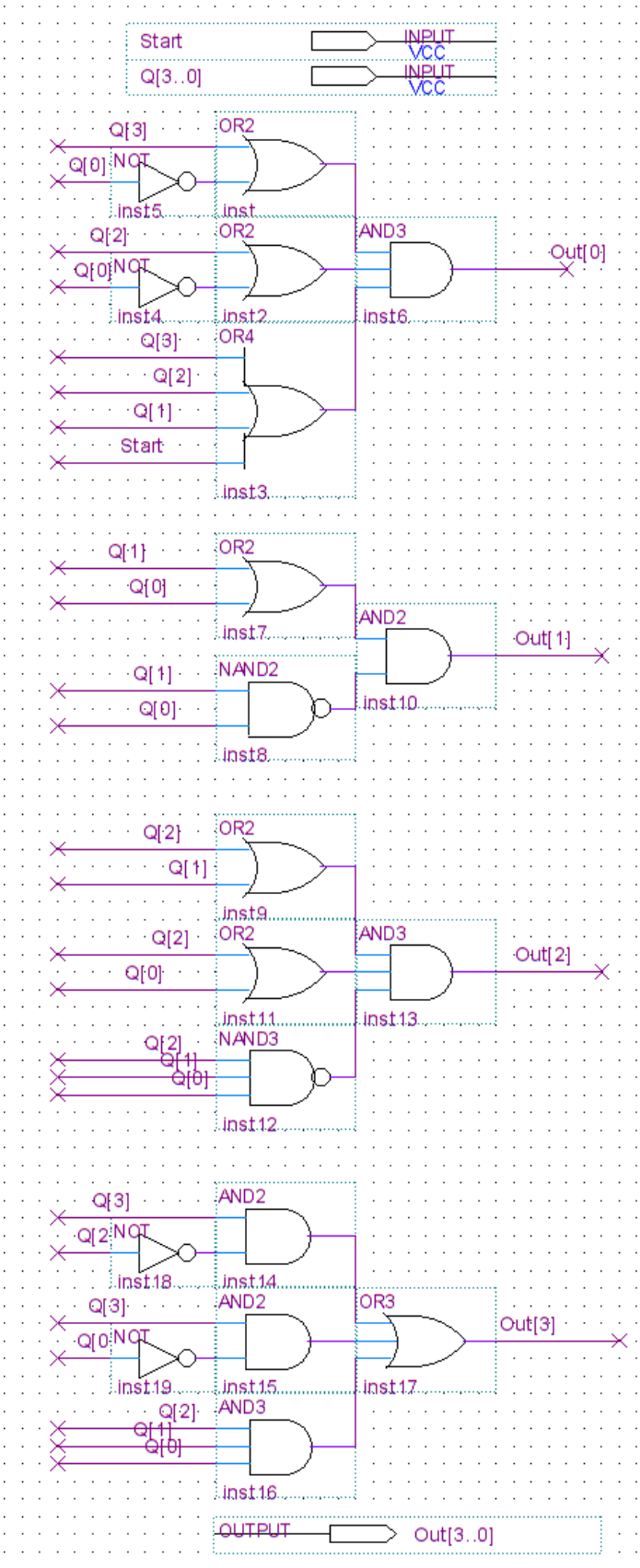
Q1+		Q1_Q0			
		00	01	11	10
Q3_Q2	00	1			1
	01		1		1
	11		1	X	X
	10		1		1

$Q_{1+} = (Q_1+Q_0) \times (Q_1.Q_0)'$

Q0+		Q1_Q0			
		00	01	11	10
Q3_Q2	00				1
	01	1			1
	11		1	X	X
	10	1			1

$Q_{0+} = Q_1.Q_0' + Q_3'.Q_2.Q_0' + Q_3.Q_2'.Q_0' + Q_3.Q_2.Q_0$

$+ \text{Start}.\ Q_3'.Q_2'.Q_1'.Q_0'$


Start
INPUT
VCC
Q[3..0]
INPUT
VCC
Q[3]
OR2
Q[0]
NOT
inst5
inst
Q[2]
OR2
AND3
Out[0]
Q[0]
NOT
inst4
inst2
inst6
Q[3]
OR4
Q[2]
Q[1]
Start
inst3
Q[1]
OR2
Q[0]
inst7
AND2
Out[1]
Q[1]
NAND2
Q[0]
inst10
inst8
Q[2]
OR2
Q[1]
inst9
Q[2]
OR2
AND3
Out[2]
Q[0]
inst11
inst13
Q[2]
NAND3
Q[1]
Q[0]
inst12
Q[3]
AND2
Q[2]
NOT
inst18
inst14
Q[3]
AND2
OR3
Out[3]
Q[0]
NOT
inst19
inst15
inst17
Q[2]
AND3
Q[1]
Q[0]
inst16
OUTPUT
Out[3..0]

- **Decoder from state to controll:**

STATE	IN				OUT																		
	Q3	Q2	Q1	Q0	I1B3	I2B4	R1B2	R2B2	AU1B3	AU1B4	S3B7	R4B5	R5B5	AU2B7	LR1	LR2	LR3	LR4	LR5	OP[3]	OP[2]	OP[1]	OP[0]	DONE
S0	0	0	0	0	1	1									1	1						X	X	
S1	0	0	0	1			1															0	0	
S2	0	0	1	0				1	1						1							0	0	
S3	0	0	1	1				1		1						1						0	0	
S4	0	1	0	0																		1	X	
S5	0	1	0	1				1											1			0	1	
S6	0	1	1	0							1						1	1				X	X	
S7	0	1	1	1	1	1						1			1	1				0	1	X	X	
S8	1	0	0	0			1							1			1					0	0	
S9	1	0	0	1				1	1				1		1					0	0	0	0	
S10	1	0	1	0						1				1		1	1					0	0	
S11	1	0	1	1				1				1								1	0	1	X	
S12	1	1	0	0				1						1			1		1			0	1	
S13	1	1	0	1							1						1	1				X	X	1
S14	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
S15	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

I1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01			1	
Q3_Q2 11			X	X
Q3_Q2 10				

I1B3 = $Q_3'.Q_2'.Q_1'.Q_0' + Q_2.Q_1.Q_0$

I2B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			
Q3_Q2 01			1	
Q3_Q2 11			X	X
Q3_Q2 10				

I2B4 = $Q_3'.Q_2'.Q_1'.Q_0' + Q_2.Q_1.Q_0$

R1B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00		1		
Q3_Q2 01				
Q3_Q2 11			X	X
Q3_Q2 10	1			

R1B2 = $Q_3'.Q_2'.Q_1'.Q_0 + Q_3.Q_2'.Q_1'.Q_0'$

R2B2	Q1_Q0			
	00	01	11	10
Q3_Q2 00	0	X	1	1
Q3_Q2 01	1	1	X	X
Q3_Q2 11	1	X	X	X
Q3_Q2 10	0	1	1	X

R2B2 = $Q_2 + Q_0 + Q_1$

AU1B3	Q1_Q0			
	00	01	11	10
Q3_Q2 00				1
Q3_Q2 01				
Q3_Q2 11			X	X
Q3_Q2 10		1		

AU1B3 = $Q_3'.Q_2'.Q_1.Q_0' + Q_3.Q_2'.Q_1'.Q_0$

AU1B4	Q1_Q0			
	00	01	11	10
Q3_Q2 00			1	
Q3_Q2 01				
Q3_Q2 11			X	X
Q3_Q2 10				1

AU1B4 = $Q_3.Q_1.Q_0' + Q_3'.Q_2'.Q_1.Q_0$

S3B7	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				1
Q3_Q2 11		1	X	X
Q3_Q2 10				

S3B7 = $Q_2.Q_1.Q_0' + Q_3.Q_2.Q_0$

R4B5	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01			1	
Q3_Q2 11			X	X
Q3_Q2 10			1	

R4B5 = $Q_3.Q_1.Q_0 + Q_2.Q_1.Q_0$

R5B5	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				
Q3_Q2 11			X	X
Q3_Q2 10		1		

R5B5 = $Q_3.Q_2'.Q_1'.Q_0$

AU2B7	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				
Q3_Q2 11	1		X	X
Q3_Q2 10	1			1

AU2B7 = $Q_3.Q_0'$

LR1	Q1_Q0			
	00	01	11	10
Q3_Q2 00	1			1
Q3_Q2 01			1	
Q3_Q2 11			X	X
Q3_Q2 10		1		

LR1 = $Q_3'.Q_2'.Q_0' + Q_2.Q_1.Q_0 + Q_3.Q_2'.Q_1'.Q_0$

LR2	Q1_Q0			
	00	01	11	10
Q3_Q2 00		1	1	
Q3_Q2 01			1	
Q3_Q2 11			X	X
Q3_Q2 10				1

LR2 = $Q_3'.Q_2'.Q_1 + Q_2.Q_1.Q_0 + Q_3.Q_1.Q_0'$

LR3	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				1
Q3_Q2 11	1	1	X	X
Q3_Q2 10	1			1

LR3 = $Q_3.Q_2 + Q_3.Q_0' + Q_2.Q_1.Q_0'$

LR4	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01				1
Q3_Q2 11		1	X	X
Q3_Q2 10				

LR4 = $Q_3.Q_2.Q_0 + Q_2.Q_1.Q_0'$

LR5	Q1_Q0			
	00	01	11	10
Q3_Q2 00				
Q3_Q2 01		1		
Q3_Q2 11	1		X	X
Q3_Q2 10				

LR5 = $Q_3.Q_2.Q_0' + Q_3'.Q_2.Q_1'.Q_0$

OP[3]	Q1_Q0			
	00	01	11	10
Q3_Q2 00	X	X	X	X
Q3_Q2 01	X	X	0	X
Q3_Q2 11	X	X	X	X
Q3_Q2 10	X	0	1	X

OP[3] = $Q_3.Q_1$

OP[2]		Q1_Q0			
		00	01	11	10
Q3_Q2	00	X	X	X	X
	01	X	X	1	X
	11	X	X	X	X
	10	X	0	0	X

OP[2] = Q_3'

OP[1]		Q1_Q0			
		00	01	11	10
Q3_Q2	00	X	0	0	0
	01	1	0	X	X
	11	0	X	X	X
	10	0	0	1	0

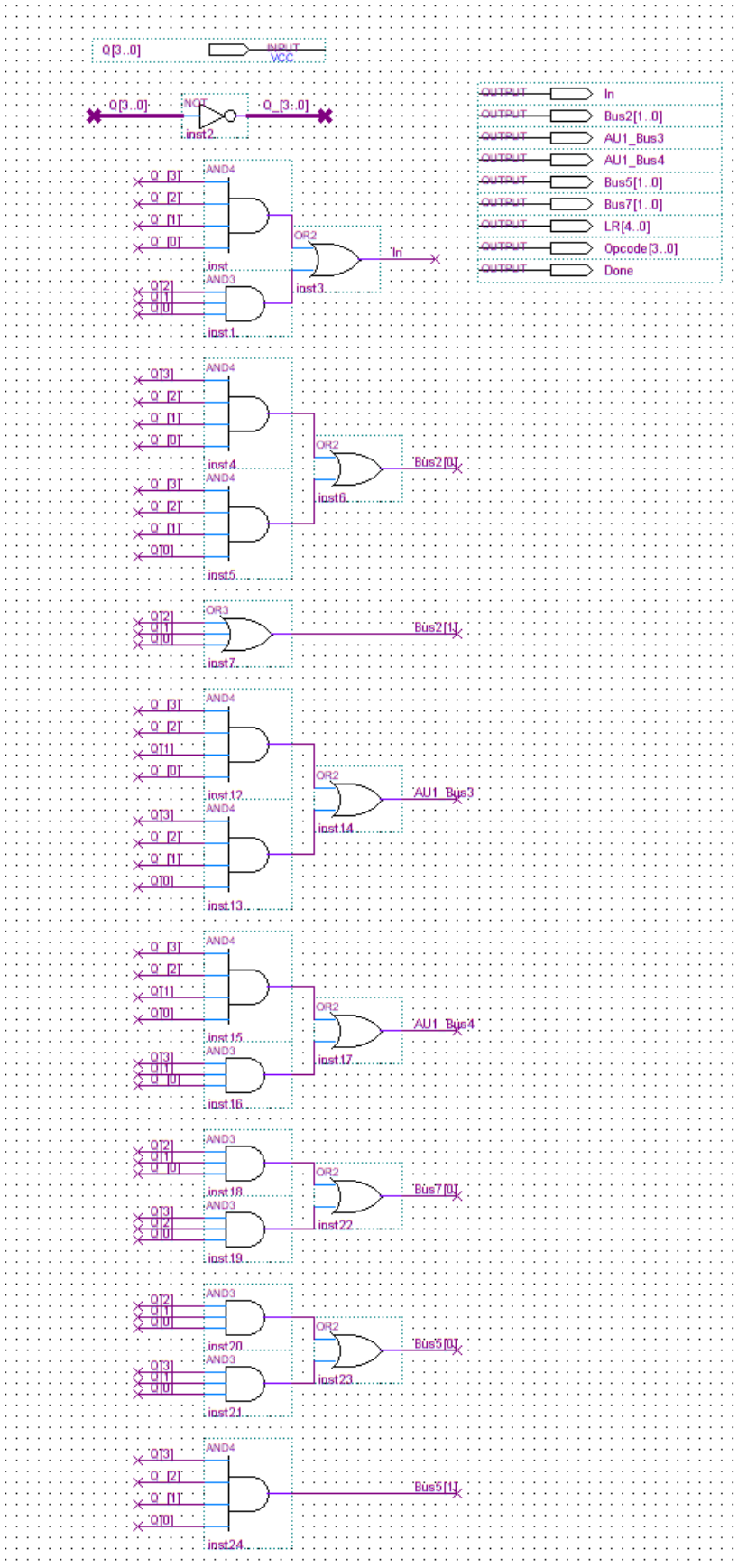
OP[1] = $Q_3'.Q_1'.Q_0' + Q_3.Q_1.Q_0$

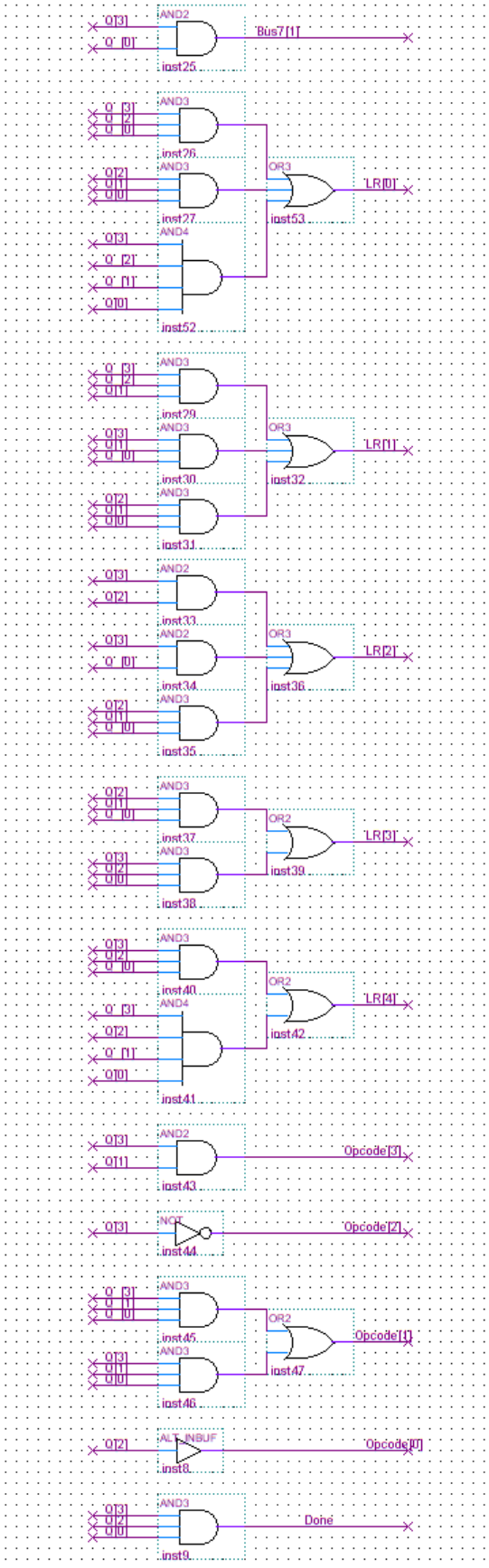
OP[0]		Q1_Q0			
		00	01	11	10
Q3_Q2	00	X	0	0	0
	01	X	1	X	X
	11	1	X	X	X
	10	0	0	X	0

OP[0] = Q_2

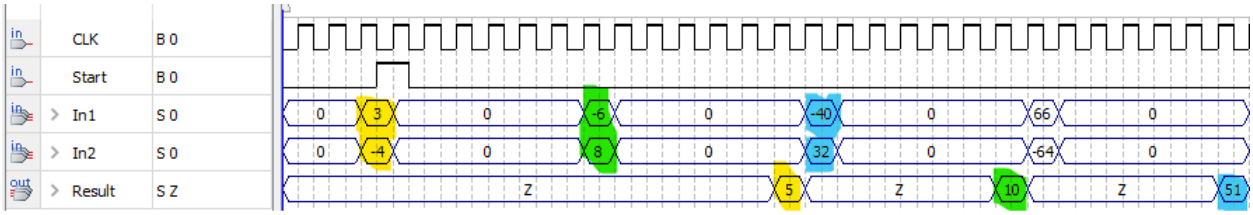
DONE		Q1_Q0			
		00	01	11	10
Q3_Q2	00				
	01				
	11		1	X	X
	10				

DONE = $Q_3.Q_2.Q_0$


O[3..0]
INPUT
VCC
O[3..0]
NOT
O_[3:0]
inst2
OUTPUT
In
Bus2[1..0]
AU1_Bus3
AU1_Bus4
Bus5[1..0]
Bus7[1..0]
LR[4..0]
Opcode[3..0]
Done
AND4
AND3
OR2
OR3
inst
inst1
inst3
In
inst4
inst5
inst6
Bus2[0]
inst7
Bus2[1]
inst12
inst13
inst14
AU1_Bus3
inst15
inst16
inst17
AU1_Bus4
inst18
inst19
inst22
Bus7[0]
inst20
inst21
inst23
Bus5[0]
inst24
Bus5[1]


O[3]
O[0]
AND2
inst25
Bus7[1]
O[3]
O[2]
O[0]
AND3
inst26
O[2]
O[1]
O[0]
AND3
inst27
OR3
inst53
LR[0]
O[3]
O[2]
O[1]
O[0]
AND4
inst52
O[3]
O[2]
O[1]
AND3
inst29
O[3]
O[1]
O[0]
AND3
inst30
OR3
inst32
LR[1]
O[2]
O[1]
O[0]
AND3
inst31
O[3]
O[2]
AND2
inst33
O[3]
O[0]
AND2
inst34
OR3
inst36
LR[2]
O[2]
O[1]
O[0]
AND3
inst35
O[2]
O[1]
O[0]
AND3
inst37
OR2
inst39
LR[3]
O[3]
O[2]
O[1]
AND3
inst38
O[3]
O[2]
O[0]
AND3
inst40
OR2
inst42
LR[4]
O[3]
O[2]
O[1]
O[0]
AND4
inst41
O[3]
O[1]
AND2
inst43
Opcode[3]
O[3]
NOT
inst44
Opcode[2]
O[3]
O[1]
O[0]
AND3
inst45
OR2
inst47
Opcode[1]
O[3]
O[1]
O[0]
AND3
inst46
O[2]
ALT_INBUF
inst8
Opcode[0]
O[3]
O[2]
O[0]
AND3
inst9
Done

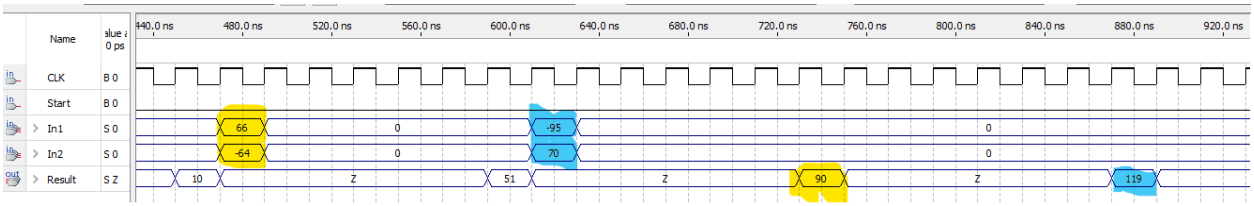
3. Waveform




```
a = 3
b = -4
t1 = |a| = 3
t2 = |b| = 4
t4 = min(t1, t2)>>1 = 1
x = max(t1, t2) = 4
t3 = max(t1, t2)>>3 = 0
t5 = x - t3 = 4
t6 = t4 + t5 = 5
t7 = max(t6, x) = 5
Result = t7 = 5
```

```
a = -6
b = 8
t1 = |a| = 6
t2 = |b| = 8
t4 = min(t1, t2)>>1 = 3
x = max(t1, t2) = 8
t3 = max(t1, t2)>>3 = 1
t5 = x - t3 = 7
t6 = t4 + t5 = 10
t7 = max(t6, x) = 10
Result = t7 = 10
```

```
a = -40
b = 32
t1 = |a| = 40
t2 = |b| = 32
t4 = min(t1, t2)>>1 = 16
x = max(t1, t2) = 40
t3 = max(t1, t2)>>3 = 5
t5 = x - t3 = 35
t6 = t4 + t5 = 51
t7 = max(t6, x) = 51
Result = t7 = 51
```




```
a = 66
b = -64
t1 = |a| = 66
t2 = |b| = 64
t4 = min(t1, t2)>>1 = 32
x = max(t1, t2) = 66
t3 = max(t1, t2)>>3 = 8
t5 = x - t3 = 58
t6 = t4 + t5 = 90
t7 = max(t6, x) = 90
Result = t7 = 90
```

```
a = -95
b = 70
t1 = |a| = 95
t2 = |b| = 70
t4 = min(t1, t2)>>1 = 35
x = max(t1, t2) = 95
t3 = max(t1, t2)>>3 = 11
t5 = x - t3 = 84
t6 = t4 + t5 = 119
t7 = max(t6, x) = 119
Result = t7 = 119
```