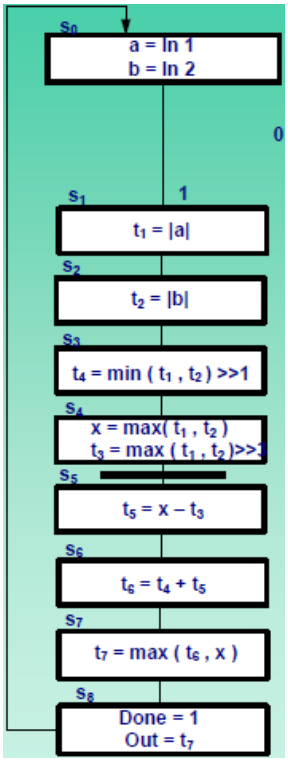
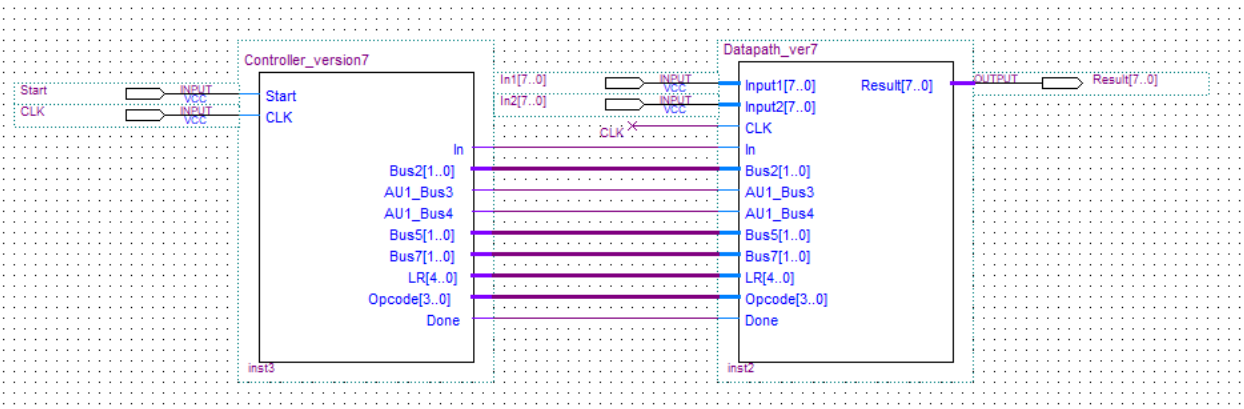
Datapath pipeline with pipelined functional units

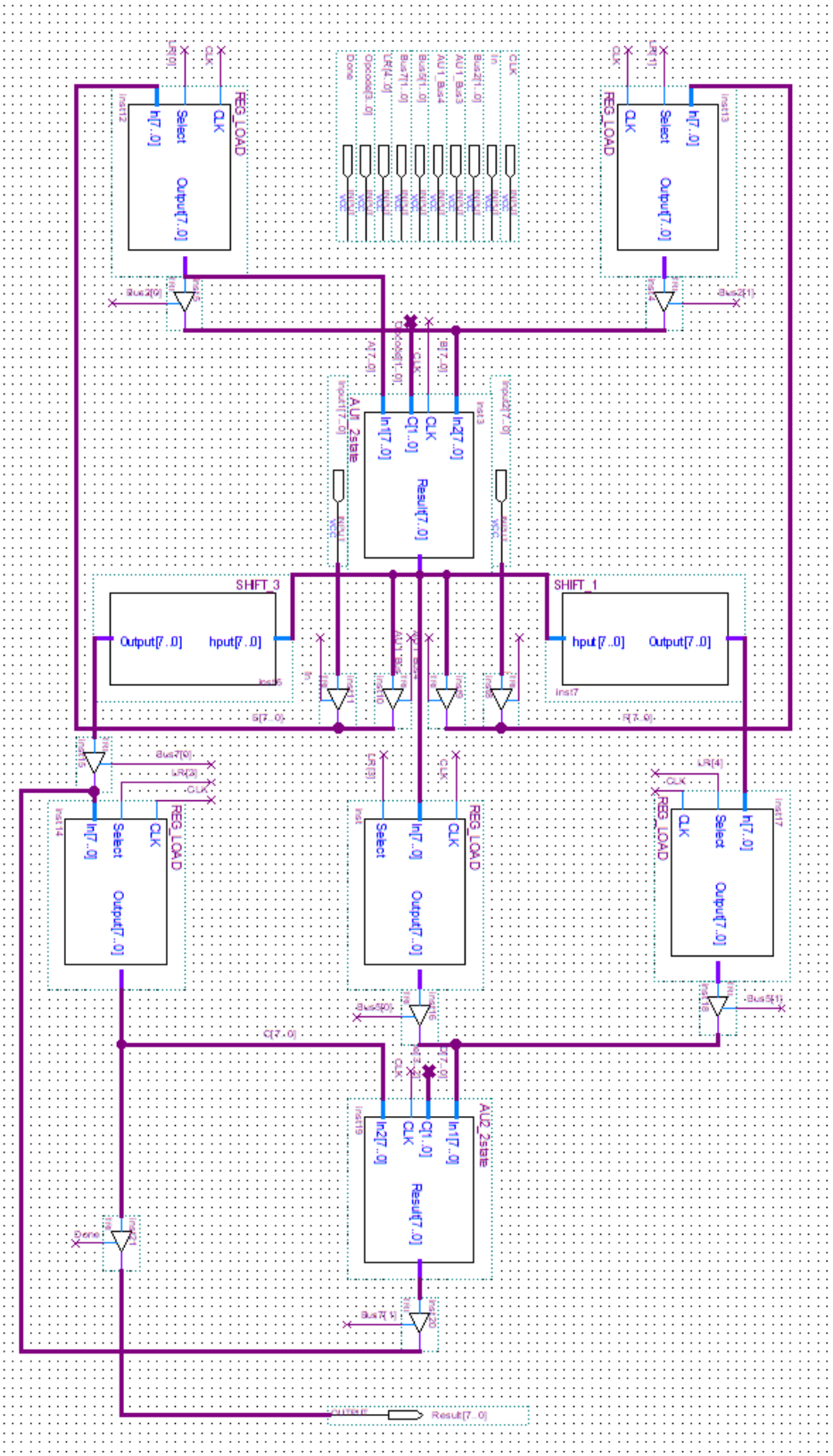
Algorithm:



Circuit (includes controller and datapath):



1. Datapath:


REG_LOAD
In[7..0]
Select
CLK
Output[7..0]
ALU_2state
In1[7..0]
In2[7..0]
Ct[1..0]
Result[7..0]
SHIFT_3
SHIFT_1
Output[7..0]
hput[7..0]
Result[7..0]

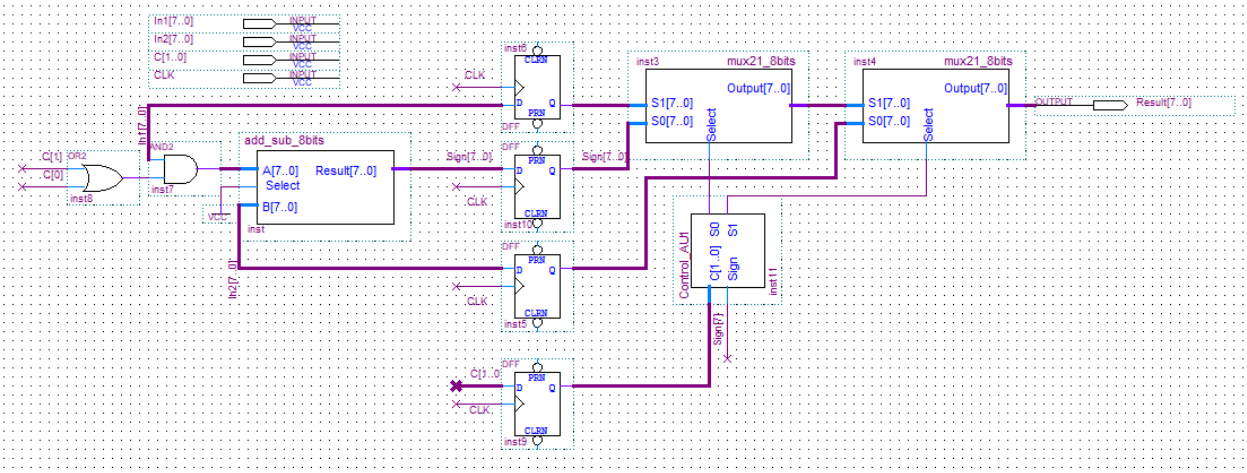
- The datapath has 22 inputs and 1 output, of which 2 inputs (8 bits) transmit the number to be calculated, the remaining 20 inputs are control signals (opcodes of AU blocks, read/write signals of registers, tries, CLK), the output for the final result.

- 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

-

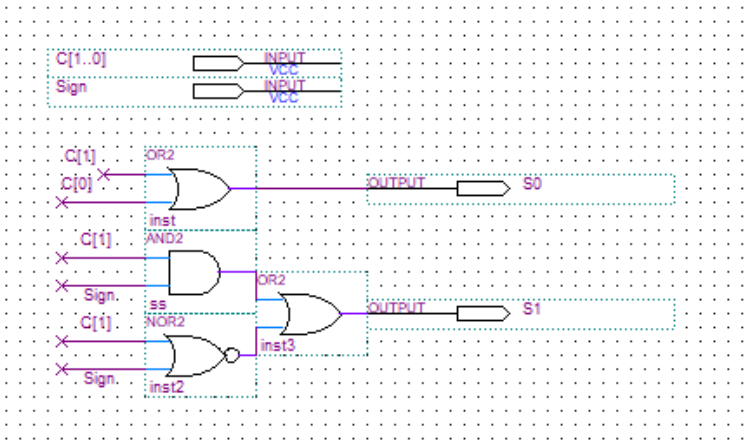
S1		C[1..0]			
		00	01	11	10
Sign	0	1	1		
	1			1	1

Mux	
Select	Result
0	S0
1	S1

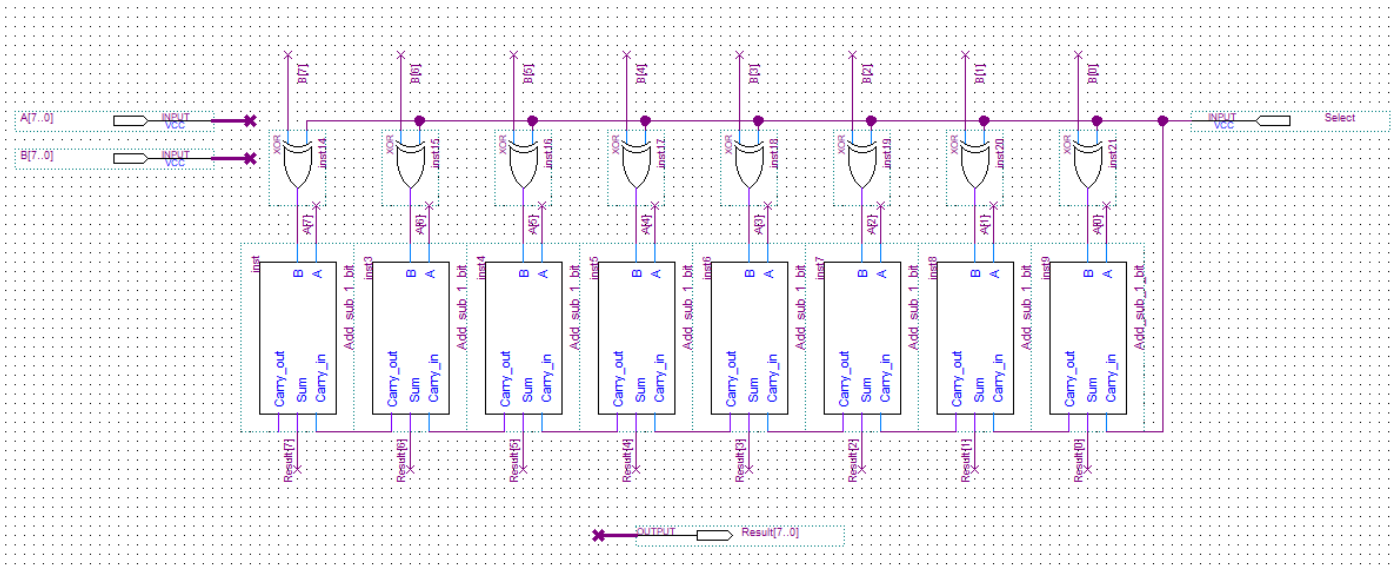
S0		C[1..0]			
		00	01	11	10
Sign	0	0	1	x	x
	1	x	x	1	1

S1 = C[1] + C[0]

S0 = Sign.C[1] + Sign’.C[0]’

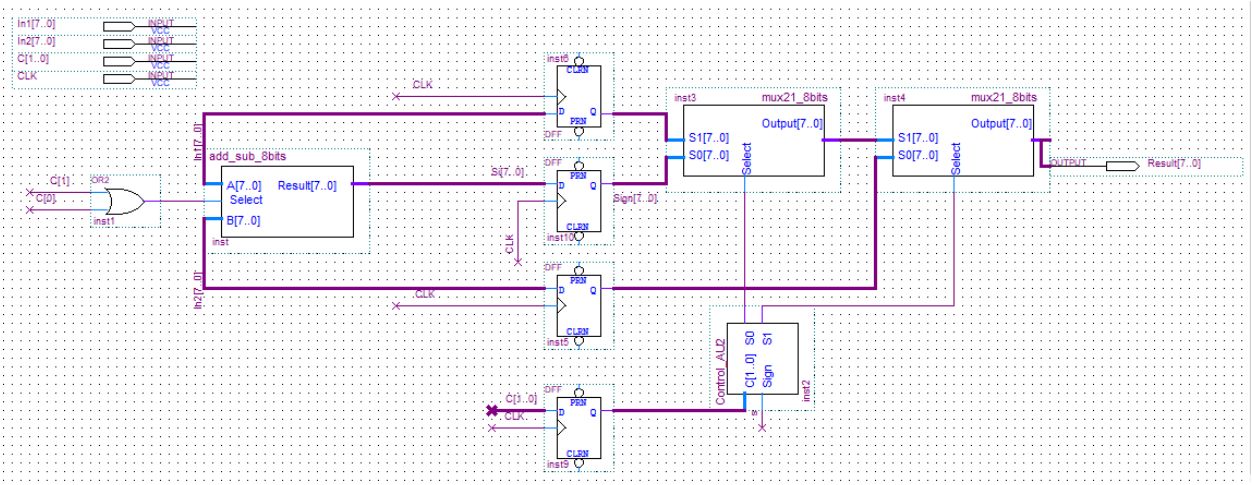


Add_sub:



Select	Function
0	Add
1	Sub

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

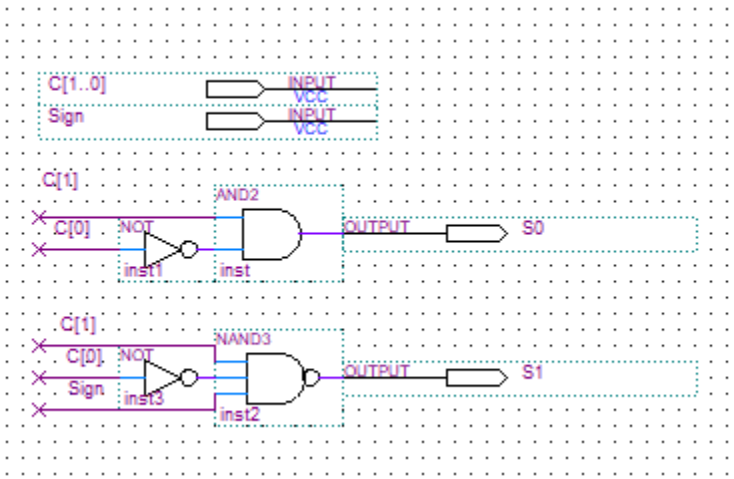
	Control AU1							Output of AU1
	Input			Output				
	Sign	C[1]	C[0]	S1	S0	Select	Funtion	
A>B	0	0	0	1	0	0	Add	Result of Add_sub
	0	0	1	1	0	1	Sub	Result of Add_sub
	0	1	0	1	1	1	Max	A
	0	1	1	1	0	1	Sub	Result of Add_sub
A<B	1	0	0	1	0	0	Add	Result of Add_sub
	1	0	1	1	0	1	Sub	Result of Add_sub
	1	1	0	0	x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

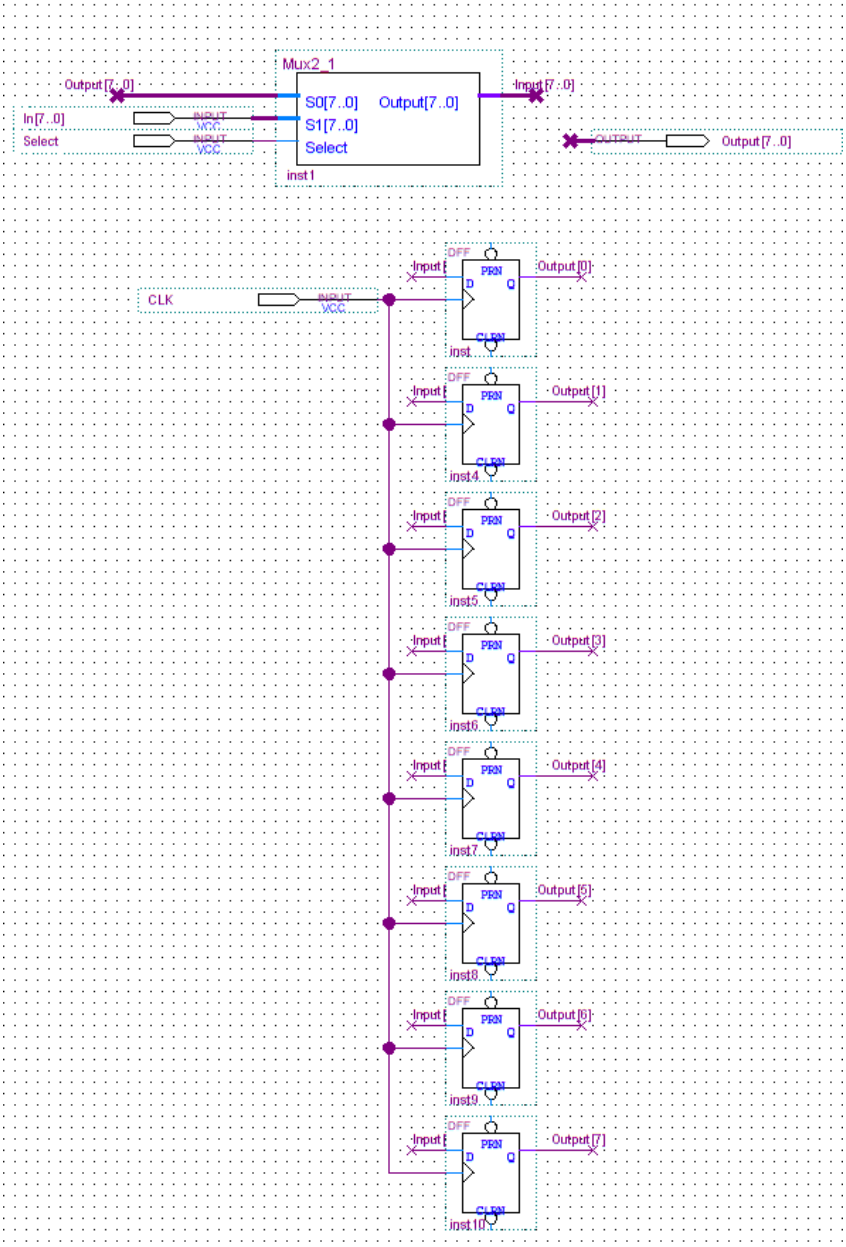
Select		C[1..0]			
		00	01	11	10
Sign	0	0	1	1	1
	1	0	1	1	1

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]’

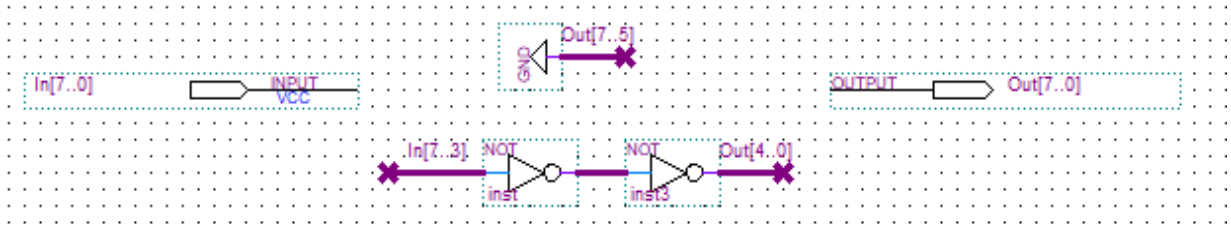


- **Register:**



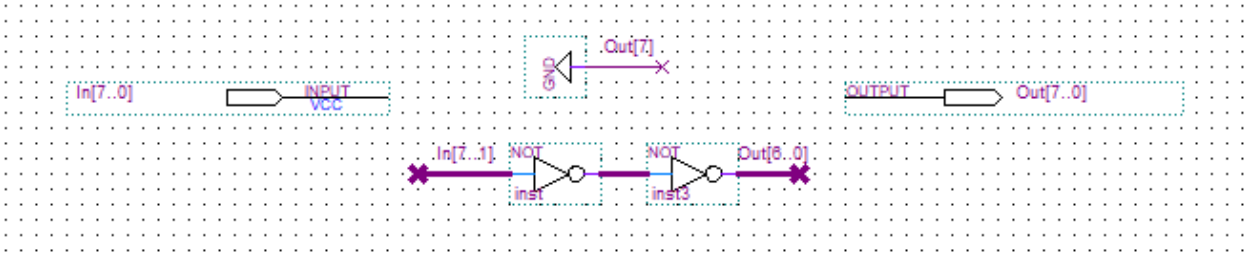
- Register if created by 8 D-FF and “Mux2_1”.
- The input Select decides that the register loads new value or not. If Select = 0 => no load new value and put previous value (S0) into D-FF so it can stores value, otherwise, Select = 1 => load new value (S1).

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



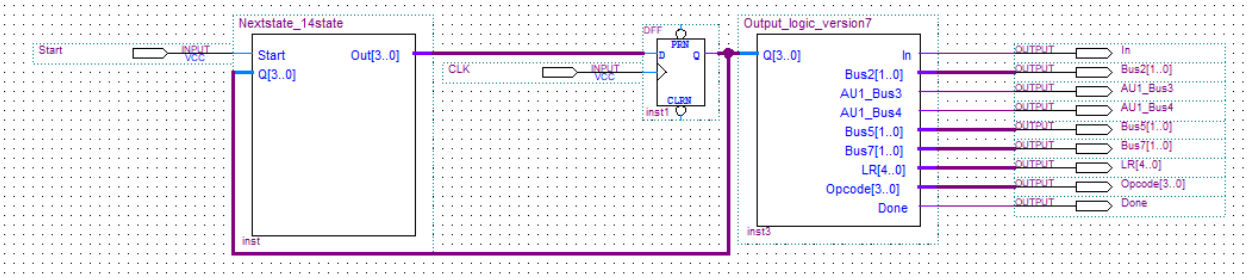
Connect bits [7..3] of input to bits[4..0] of output, remaining bits of output are GND.

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



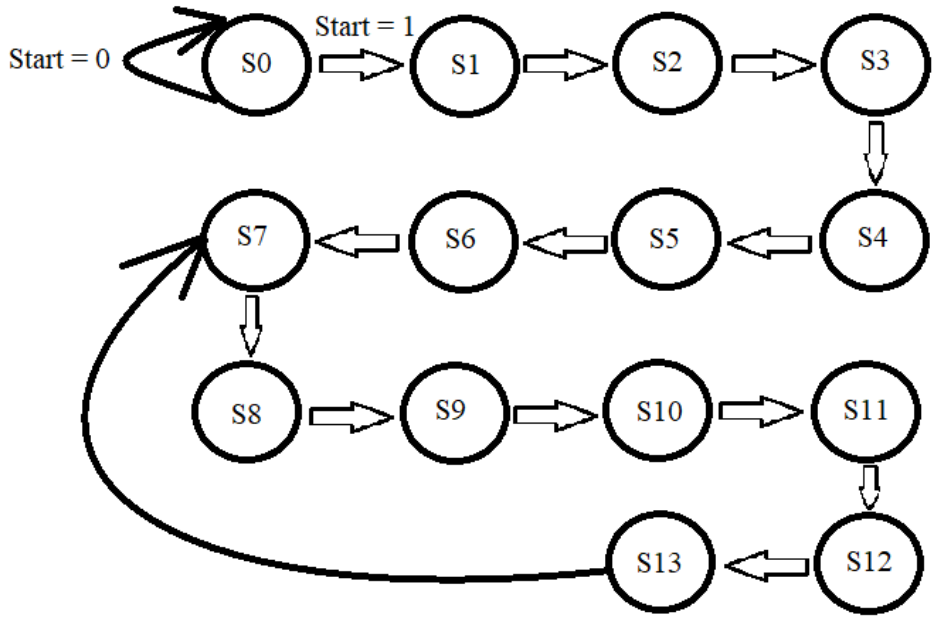
Connect bits [7..1] of input to bits[6..0] of output, remaining bit of output is GND.

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.

- **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			
Q3_Q2	00	01	11	10
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			
Q3_Q2	00	01	11	10
00			1	
01	1	1		1
11	1	1	X	X
10			1	

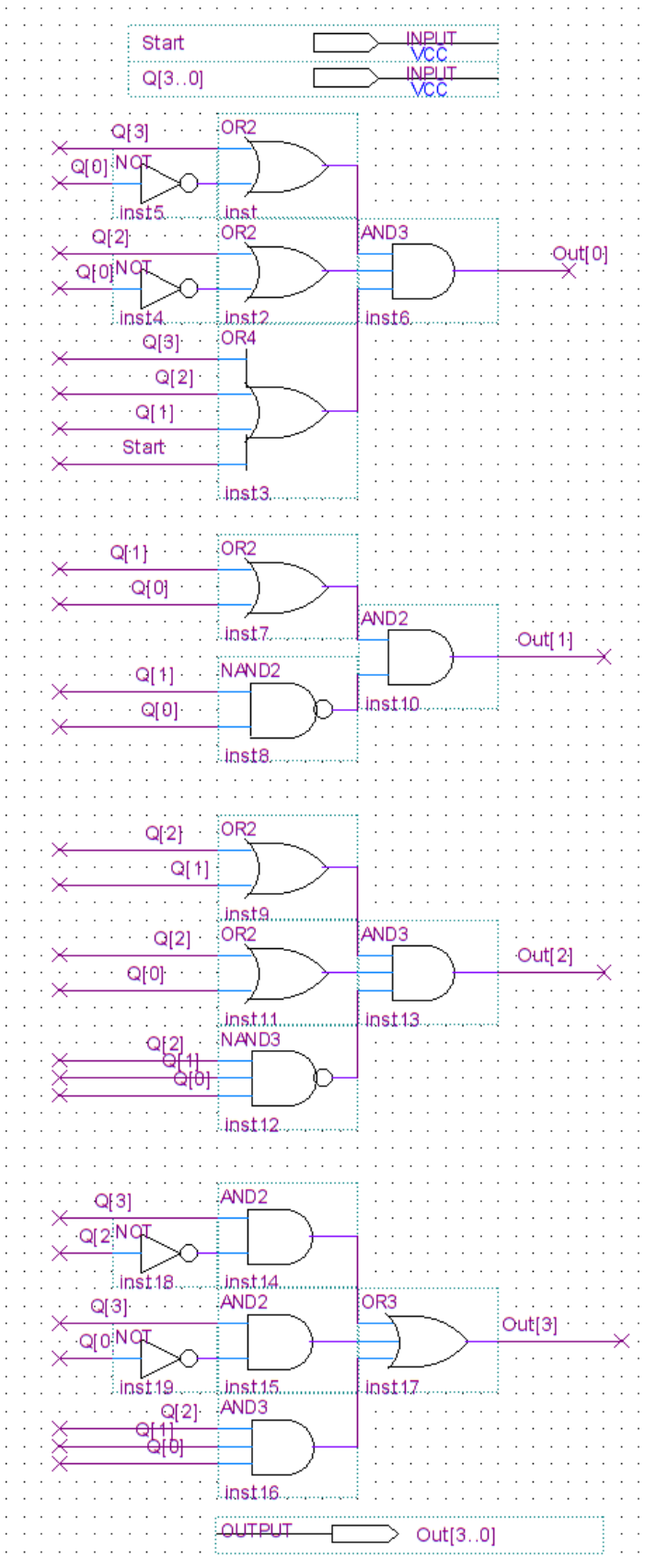
$Q_{2+} = (Q_2+Q_1) \times (Q_2+Q_0) \times (Q_2.Q_1.Q_0)'$

Q1+	Q1_Q0			
Q3_Q2	00	01	11	10
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			
Q3_Q2	00	01	11	10
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 control:**

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

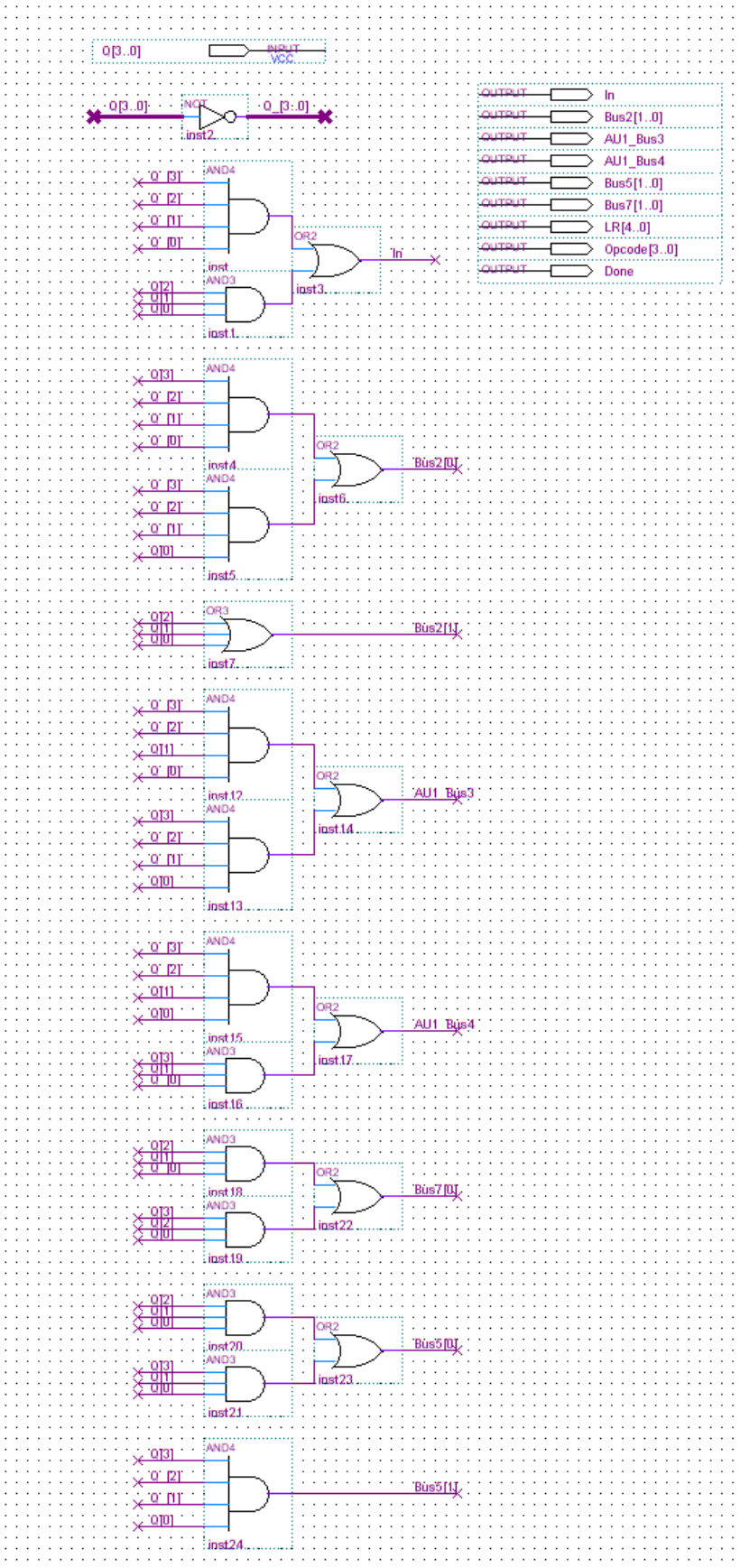
OP[1] = $Q_3'.Q_1'.Q_0' + Q_3.Q_1.Q_0$

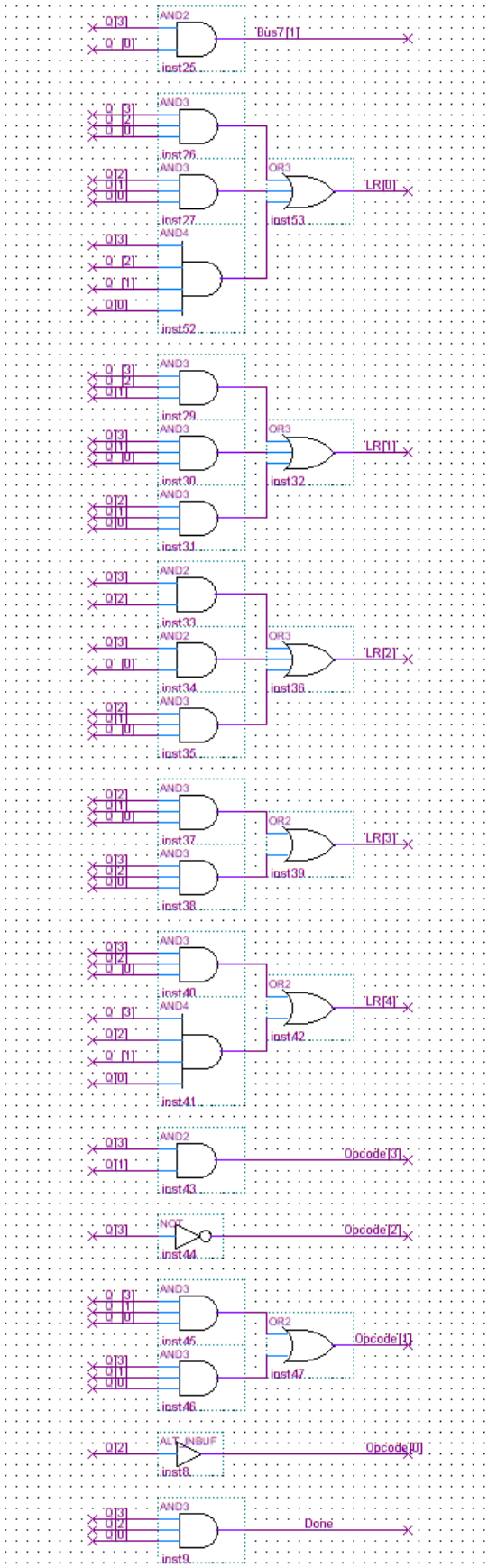
OP[0]	Q1_Q0			
Q3_Q2	00	01	11	10
00	X	0	0	0
01	X	1	X	X
11	1	X	X	X
10	0	0	X	0

OP[0] = Q_2

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

DONE = $Q_3.Q_2.Q_0$

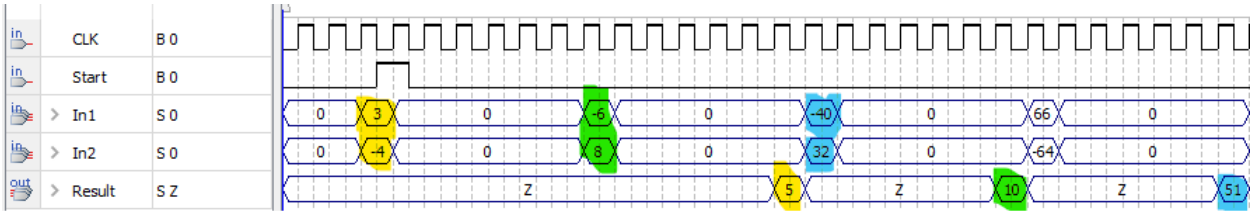

Q[3..0]
INPUT
VCC
Q[3..0]
NOT
Q_[3:.0]
inst2
OUTPUT
In
OUTPUT
Bus2[1..0]
OUTPUT
AU1_Bus3
OUTPUT
AU1_Bus4
OUTPUT
Bus5[1..0]
OUTPUT
Bus7[1..0]
OUTPUT
LR[4..0]
OUTPUT
Opcode[3..0]
OUTPUT
Done
AND4
Q[3]
Q[2]
Q[1]
Q[0]
OR2
In
inst
AND3
inst3
inst1
inst4
inst6
Bus2[0]
inst5
OR3
Bus2[1]
inst7
inst12
inst14
AU1_Bus3
inst13
inst15
inst17
AU1_Bus4
inst16
inst18
Bus7[0]
inst22
inst19
inst20
Bus5[0]
inst23
inst21
Bus5[1]
inst24


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

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”)

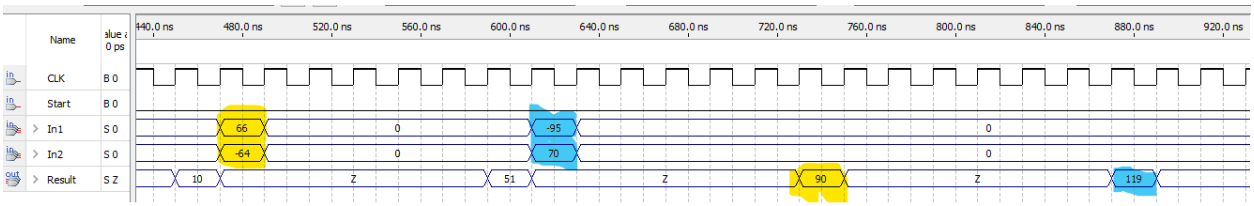
3. Waveform (input and output according to color respectively):




```
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
```