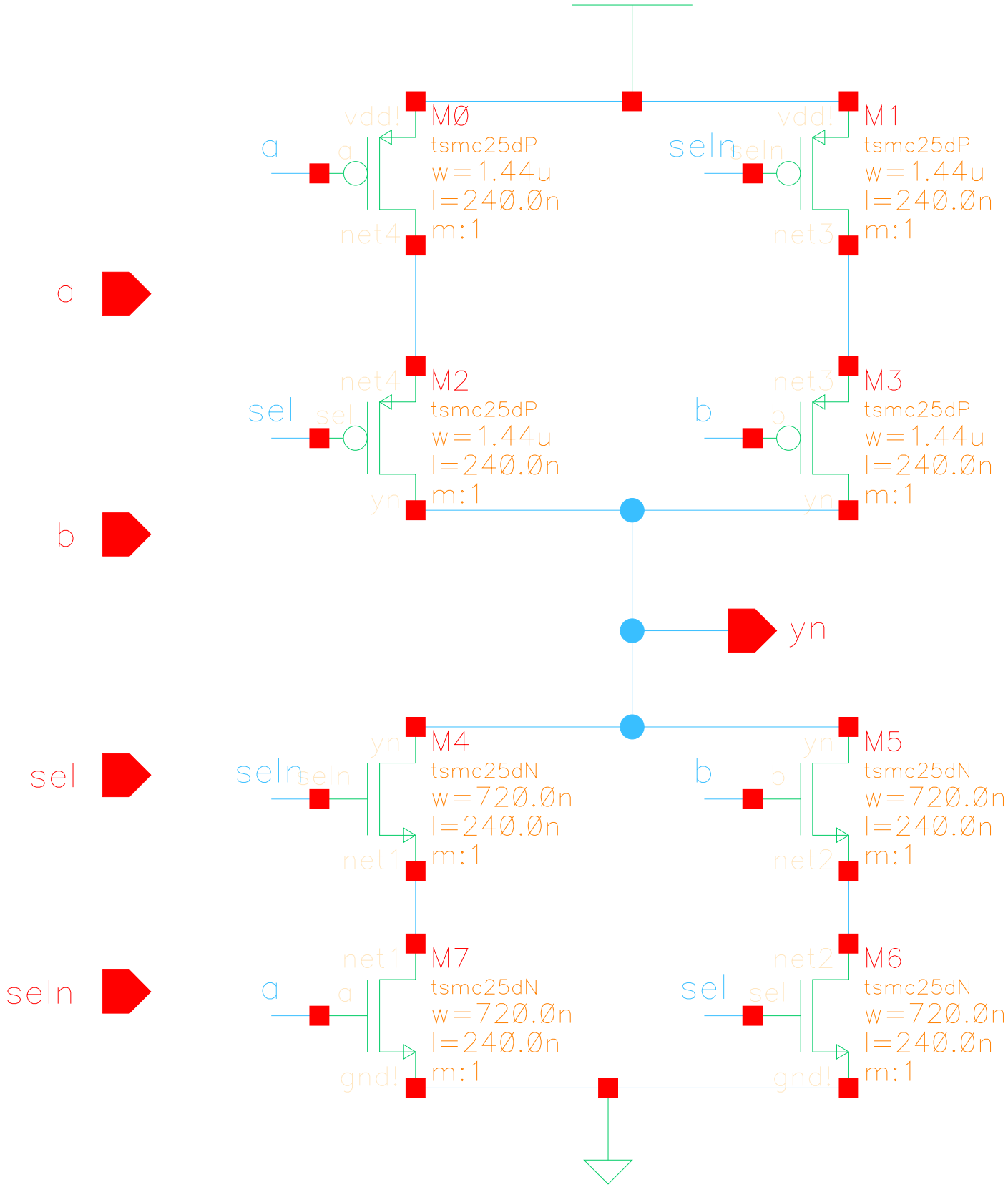
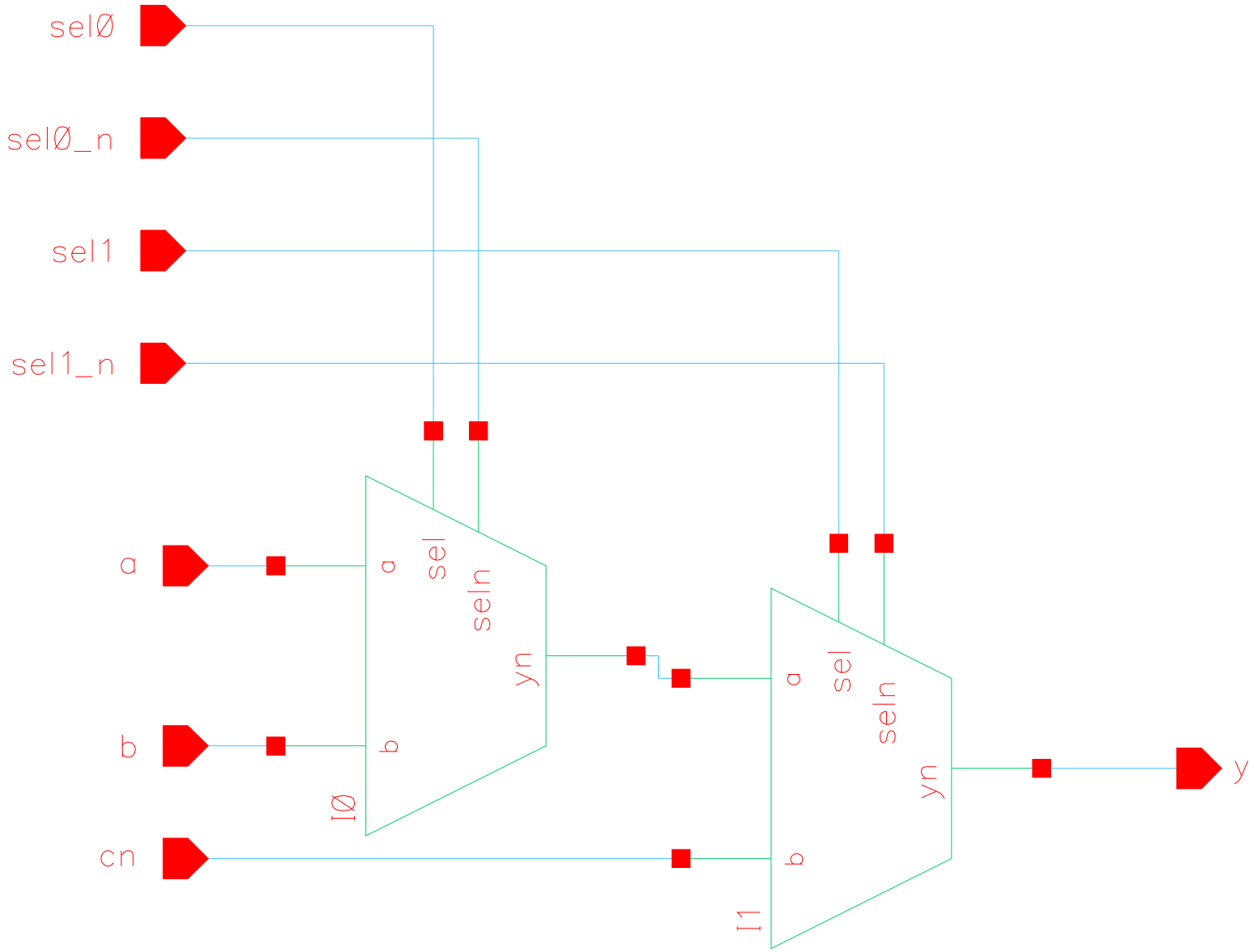
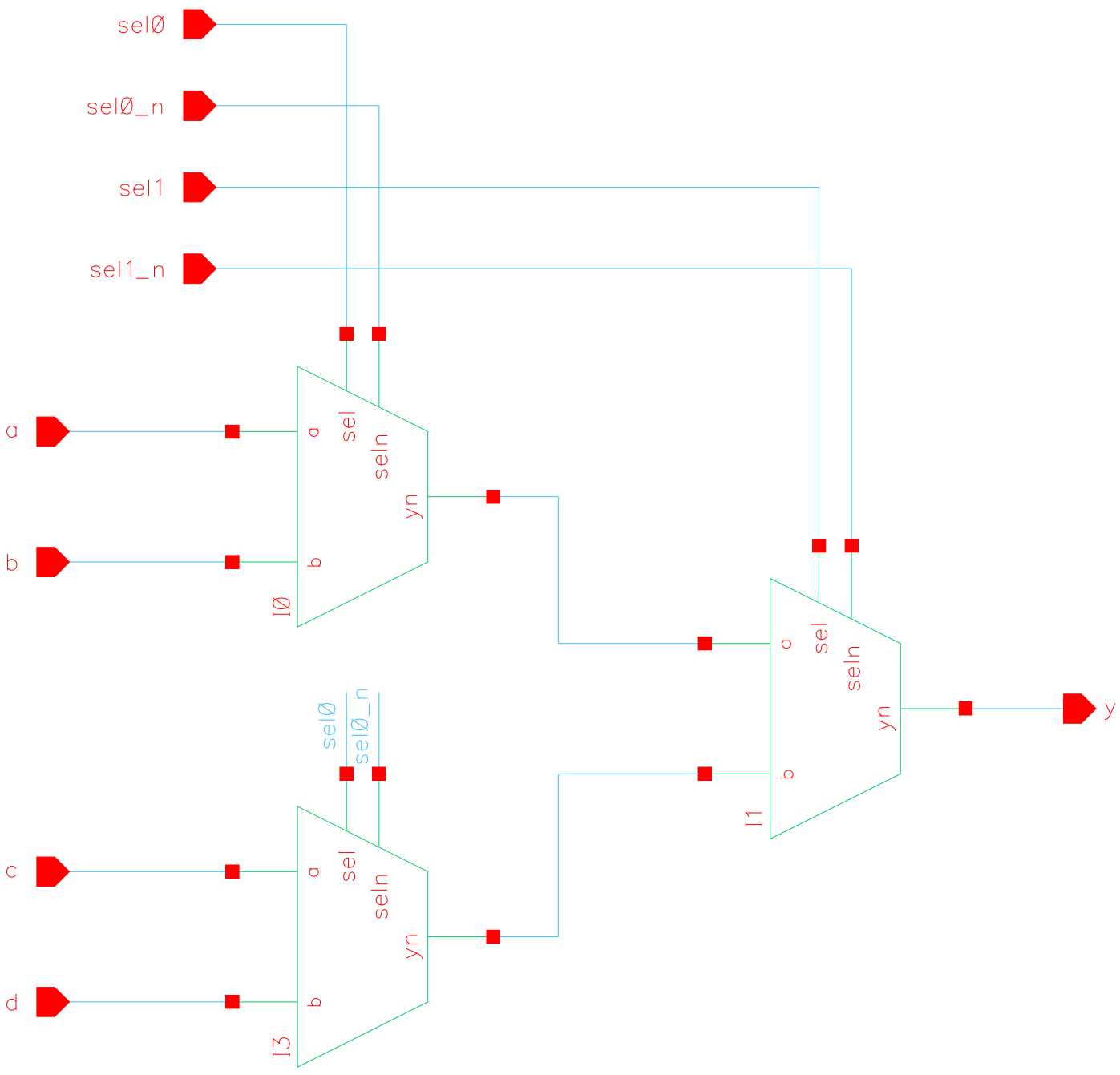
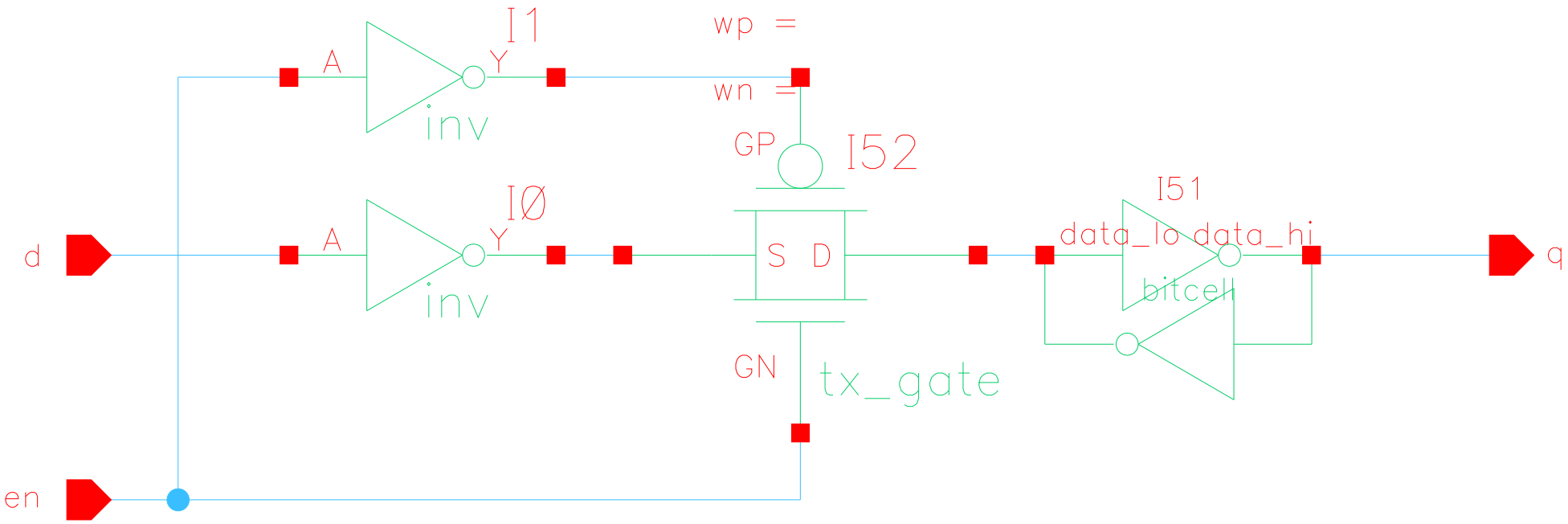
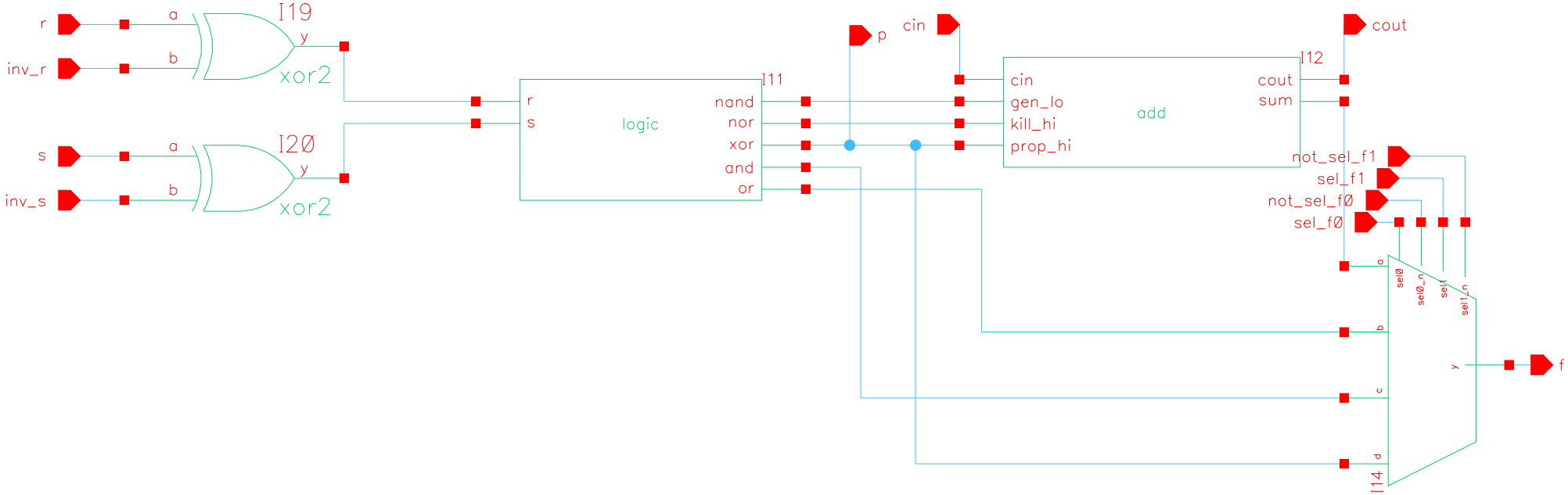
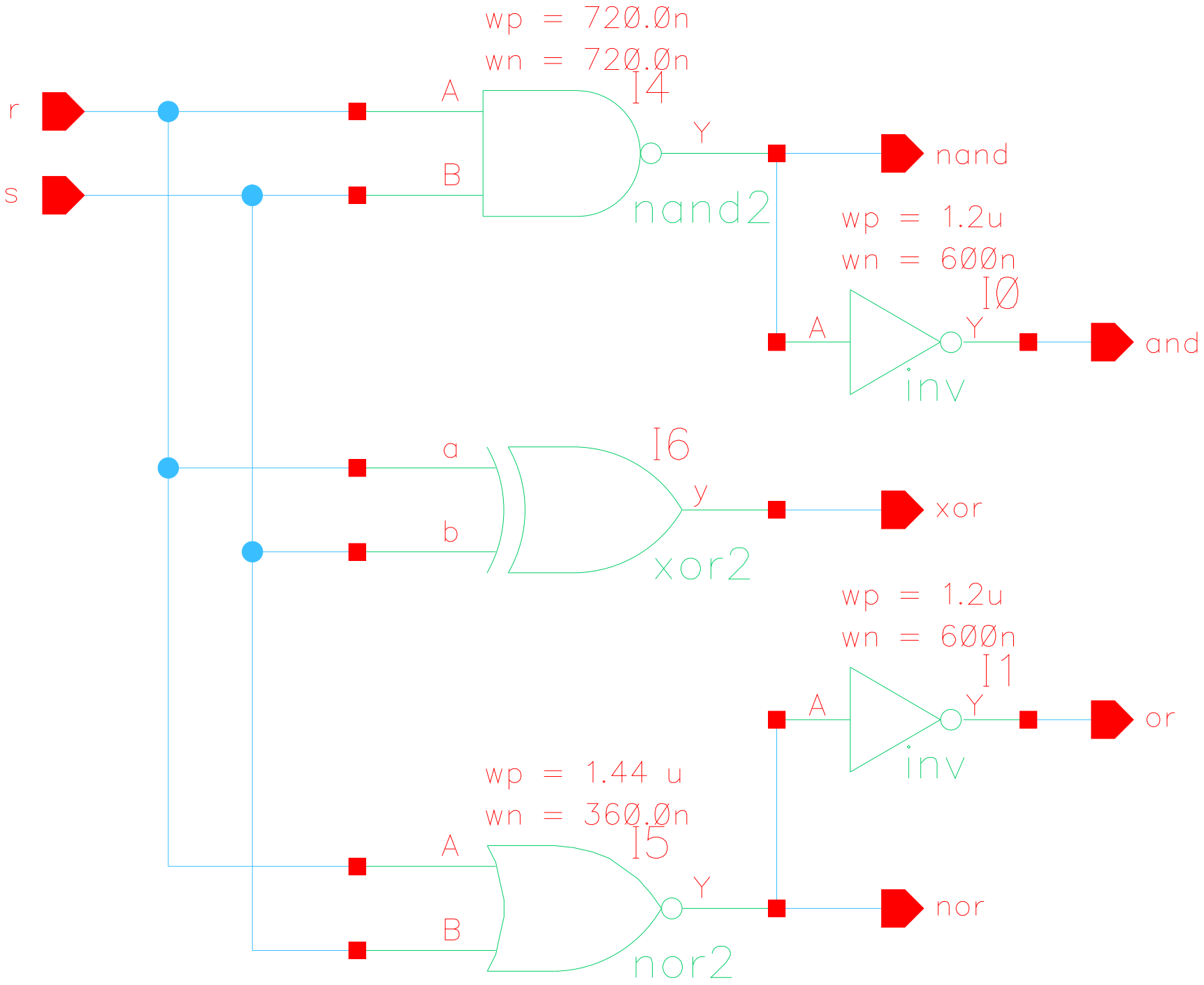

M0
tsmc25dP
w=1.44u
l=240.0n
m:1
M1
tsmc25dP
w=1.44u
l=240.0n
m:1
M2
tsmc25dP
w=1.44u
l=240.0n
m:1
M3
tsmc25dP
w=1.44u
l=240.0n
m:1
M4
tsmc25dN
w=720.0n
l=240.0n
m:1
M5
tsmc25dN
w=720.0n
l=240.0n
m:1
M7
tsmc25dN
w=720.0n
l=240.0n
m:1
M6
tsmc25dN
w=720.0n
l=240.0n
m:1
vdd!
gnd!
a
b
sel
seln
yn
net1
net2
net3
net4


sel0
sel0_n
sel1
sel1_n
a
b
cn
I0
a
b
sel
seln
yn
I1
a
b
sel
seln
yn
y


sel0
sel0_n
sel1
sel1_n
a
b
I0
a
sel
seln
yn
b
c
d
I3
a
sel
seln
yn
b
sel0
sel0_n
I1
a
sel
seln
yn
b
y


I1
wp =
A
Y
wn =
inv
GP
I52
I0
I51
d
A
Y
S D
data_lo
data_hi
q
inv
bitcell
GN
tx_gate
en


r
inv_r
a
b
I19
y
xor2
s
inv_s
a
b
I2Ø
y
xor2
r
s
logic
I11
nand
nor
xor
and
or
p
cin
cin
gen_lo
kill_hi
prop_hi
add
I12
cout
sum
cout
not_sel_f1
sel_f1
not_sel_fØ
sel_fØ
a
b
c
d
sel0
sel0_n
sel1
sel1_n
y
I14
f


wp = 720.0n
wn = 720.0n
I4
A
B
Y
nand2
r
s
nand
wp = 1.2u
wn = 600n
I0
A
Y
inv
and
I6
a
b
y
xor2
xor
wp = 1.2u
wn = 600n
I1
A
Y
inv
or
wp = 1.44 u
wn = 360.0n
I5
A
B
Y
nor2
nor

ECE 425, Sp 2023

Introduction to VLSI System Design

MP2

---

2. List the eight ALU functions and explain how you implemented each of them.

I created a table that shows all possible I_{3-5} control signals and the function each combination goes with. Then I created new columns for when I need to invert either input to calculate the function, 0 if I don't need to invert, 1 if I do. Then I labeled each function with the logic block used to calculate the result of the function, ADD if I use the ADD logic block, OR for OR gate, etc. Then I created the sel_f signal based on what the inputs to the 4x1 MUX were that selected the function output.

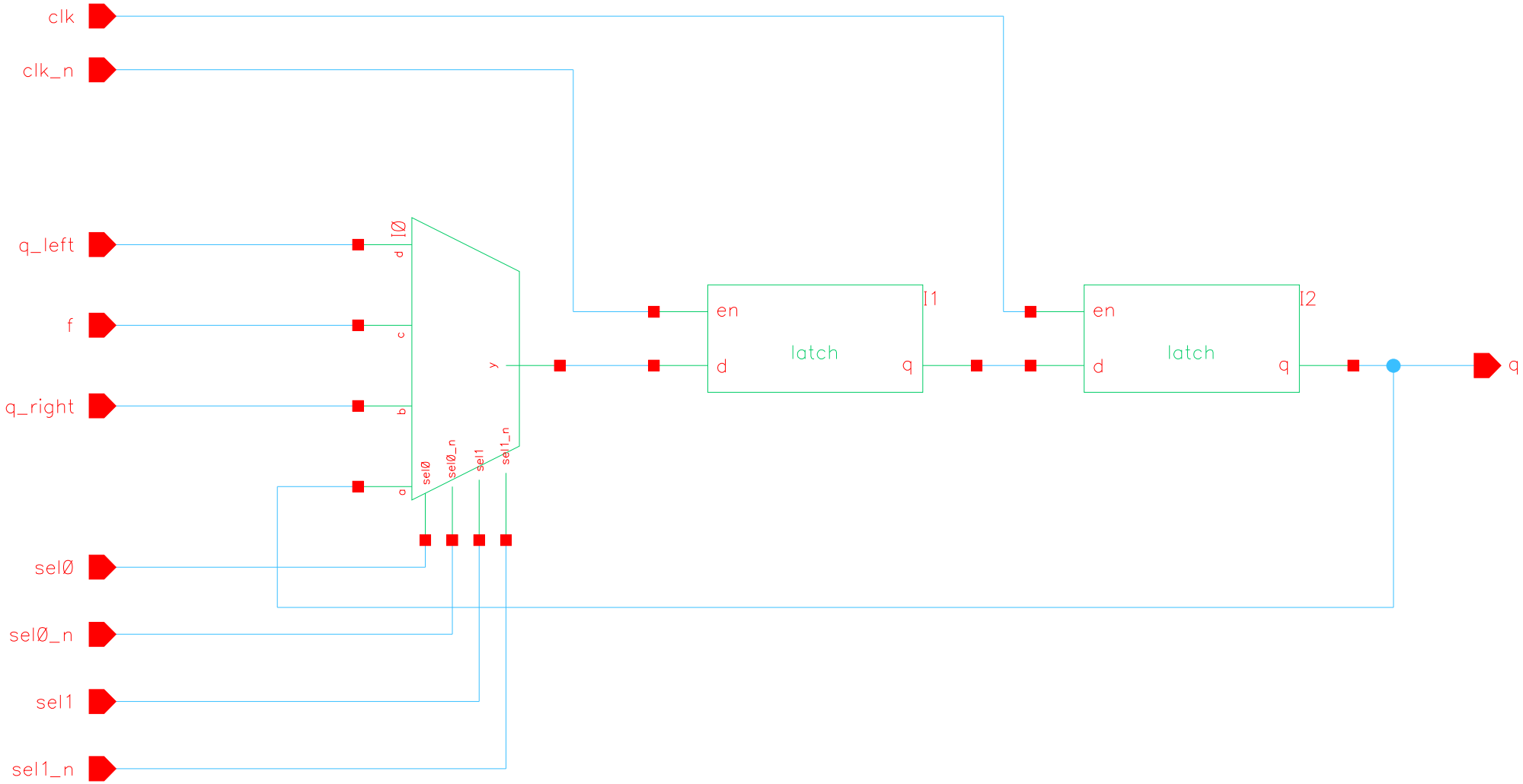
Table 1

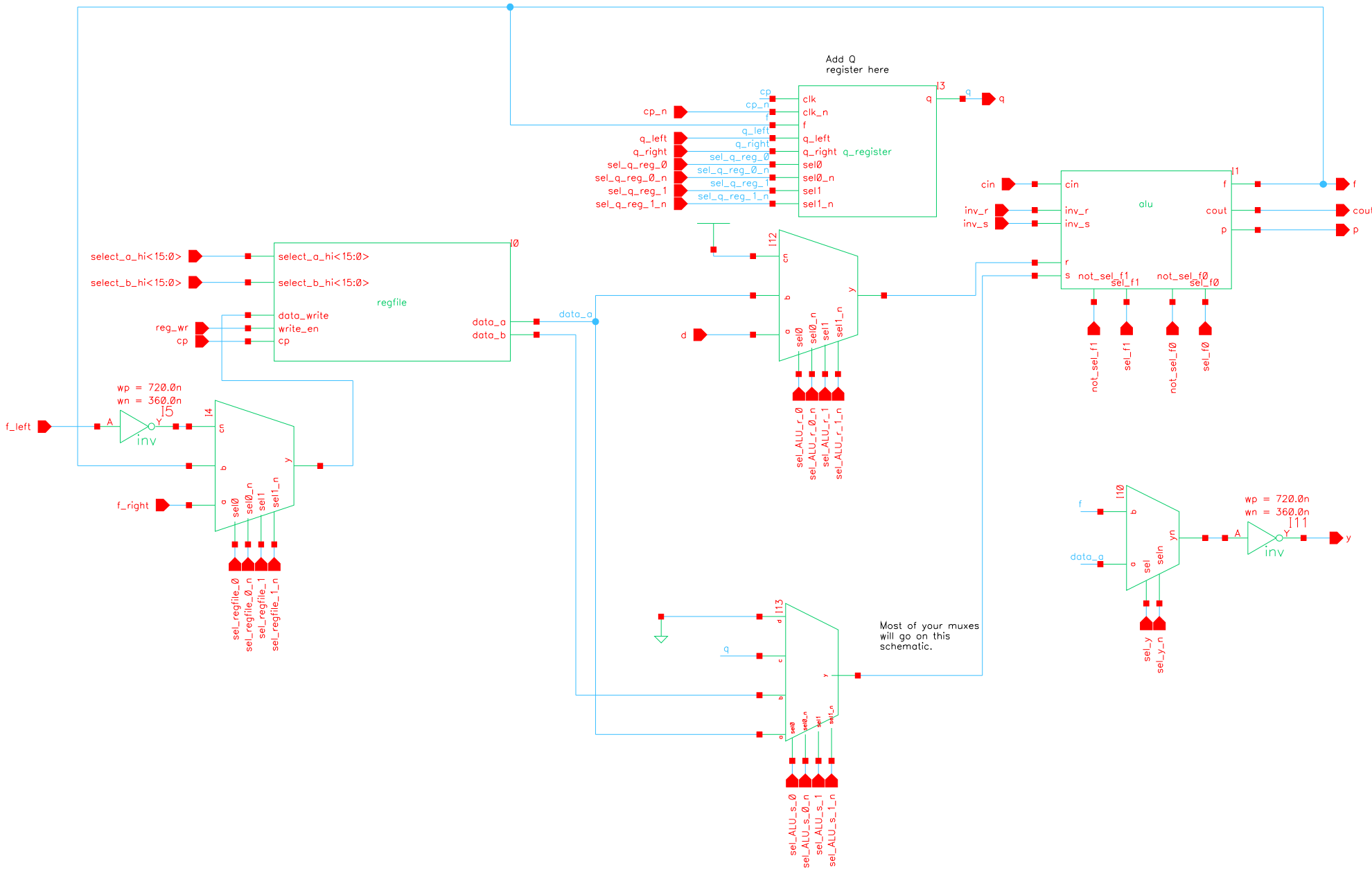
I_5	I_4	I_3	function	inv_r	inv_s	operation	sel_f[1:0]
0	0	0	$R + S$	0	0	ADD	00
0	0	1	$S - R$	1	0	ADD	00
0	1	0	$R - S$	0	1	ADD	00
0	1	1	$R \vee S$	0	0	OR	01
1	0	0	$R \wedge S$	0	0	AND	10
1	0	1	$\overline{R} \wedge S$	1	0	AND	10
1	1	0	$R \oplus S$	0	0	XOR	11
1	1	1	$R \overline{\oplus} S = R \oplus \overline{S}$	0	1	XOR	11

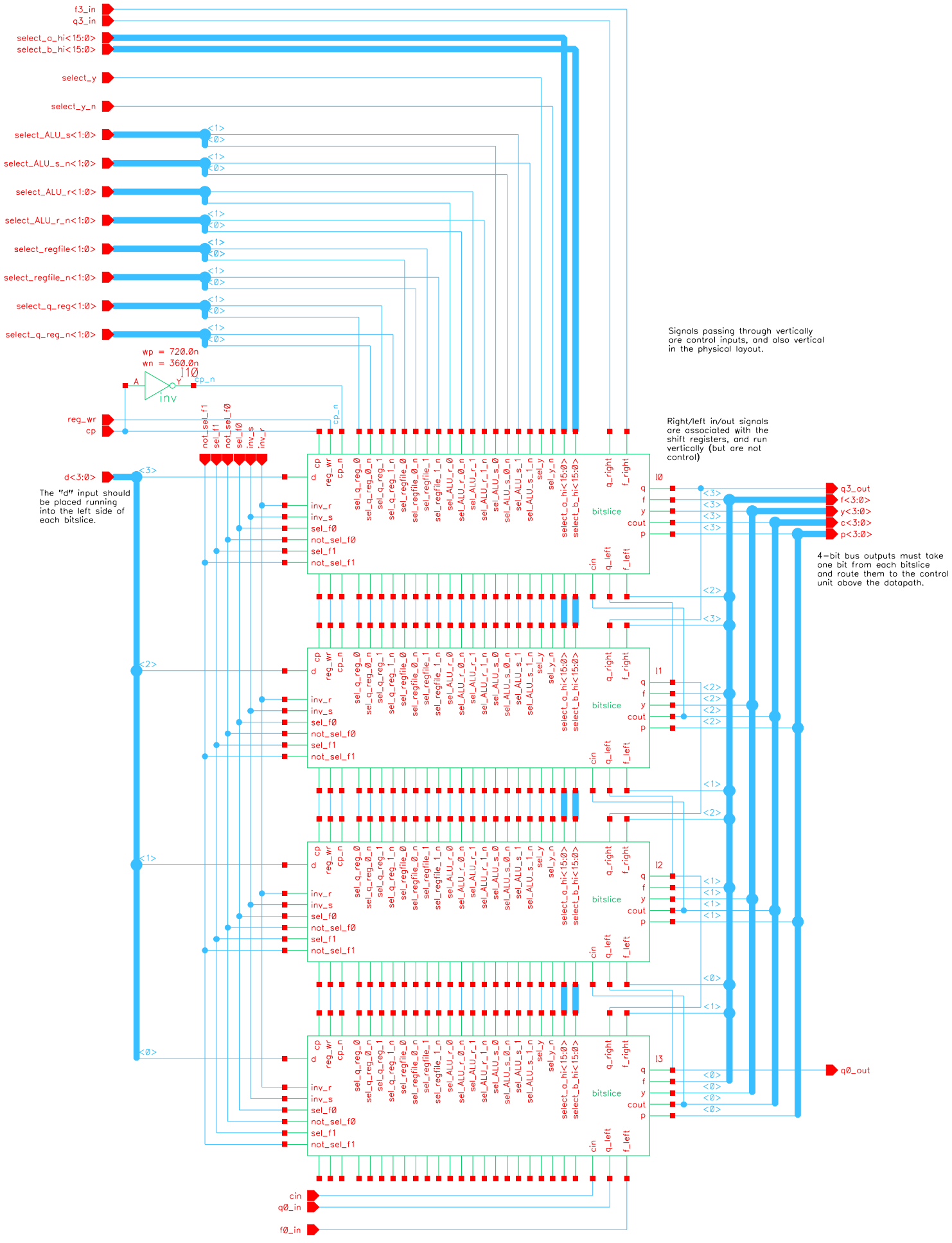
Using the values found in *Table 1*. I created functions for inv_r, inv_s, sel_f[0], and sel_f[1] in terms of I_5, I_4 and I_3, which are:

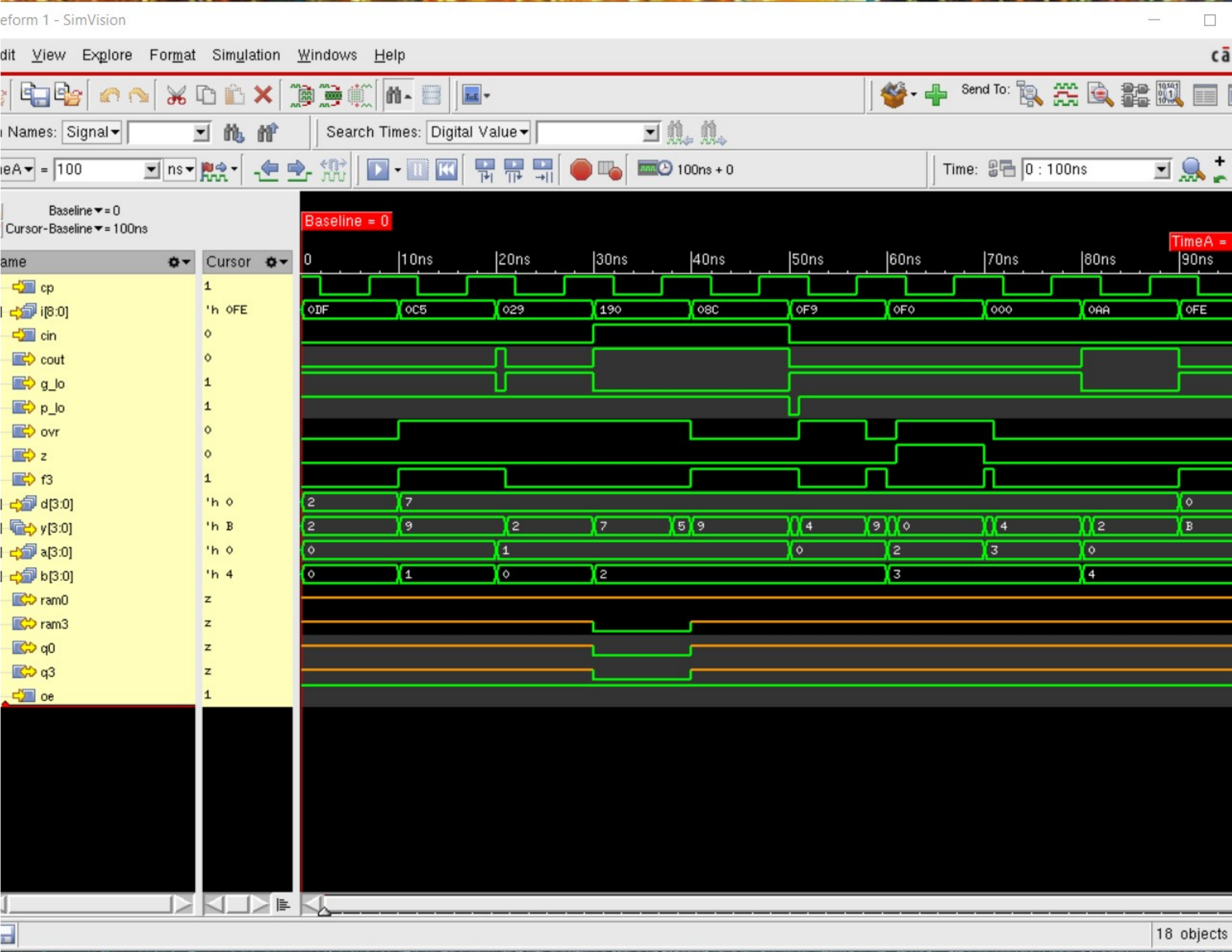
sel_r = $\sim I_4 \& I_3$
inv_s = $\sim I_5 \& I_4 \& \sim I_3$ || $I_5 \& I_4 \& I_3$
sel_f[0] = $I_4 \& I_3 || I_5 \& I_4$
sel_f[1] = I_5

These functions, along with their complements, were implemented in control.v and send to the datapath via pins.


clk
clk_n
q_left
f
q_right
sel0
sel0_n
sel1
sel1_n
I0
d
c
b
a
y
sel0
sel0_n
sel1
sel1_n
I1
en
d
latch
q
I2
en
d
latch
q
q


Add Q
register here
I3
cp
cp_n
f
q_left
q_right
sel_q_reg_0
sel_q_reg_0_n
sel_q_reg_1
sel_q_reg_1_n
clk
clk_n
f
q_left
q_right
sel0
sel0_n
sel1
sel1_n
q_register
q
q
I0
select_a_hi<15:0>
select_b_hi<15:0>
reg_wr
cp
data_write
write_en
cp
data_a
data_b
regfile
I12
cn
b
a
y
d
sel0
sel0_n
sel1
sel1_n
sel_ALU_r_0
sel_ALU_r_0_n
sel_ALU_r_1
sel_ALU_r_1_n
I1
cin
inv_r
inv_s
r
s
not_sel_f1
sel_f1
not_sel_f0
sel_f0
alu
f
cout
p
wp = 720.0n
wn = 360.0n
I5
inv
A
Y
f_left
I4
f_right
sel_regfile_0
sel_regfile_0_n
sel_regfile_1
sel_regfile_1_n
I13
Most of your muxes
will go on this
schematic.
I10
sel_y
sel_y_n
I11
y


f3_in
q3_in
select_a_hi<15:0>
select_b_hi<15:0>
select_y
select_y_n
select_ALU_s<1:0>
select_ALU_s_n<1:0>
select_ALU_r<1:0>
select_ALU_r_n<1:0>
select_regfile<1:0>
select_regfile_n<1:0>
select_q_reg<1:0>
select_q_reg_n<1:0>
wp = 720.0n
wn = 360.0n
I10
inv
reg_wr
cp
d<3:0>
The "d" input should be placed running into the left side of each bitslice.
Signals passing through vertically are control inputs, and also vertical in the physical layout.
Right/left in/out signals are associated with the shift registers, and run vertically (but are not control)
bitslice
I0
I1
I2
I3
4-bit bus outputs must take one bit from each bitslice and route them to the control unit above the datapath.
q3_out
f<3:0>
y<3:0>
c<3:0>
p<3:0>
q0_out
cin
q0_in
f0_in


eform 1 - SimVision
dit View Explore Format Simulation Windows Help
Send To:
Names: Signal
Search Times: Digital Value
100 ns
100ns + 0
Time: 0 : 100ns
Baseline = 0
Cursor-Baseline = 100ns
Baseline = 0
TimeA =
0 10ns 20ns 30ns 40ns 50ns 60ns 70ns 80ns 90ns
ame
Cursor
cp 1
i[8:0] 'h 0FE
cin 0
cout 0
g_lo 1
p_lo 1
ovr 0
z 0
f3 1
d[3:0] 'h 0
y[3:0] 'h B
a[3:0] 'h 0
b[3:0] 'h 4
ram0 z
ram3 z
q0 z
q3 z
oe 1
0DF 0C5 029 190 08C 0F9 0F0 000 0AA 0FE
18 objects

==========

```
Compare set name:       c1 (hierarchical)
Compare command:        compare . -start 0ns -end 100ns -pos 2ns -neg 2ns \
                        -maxerrors 200
Successful compares:    22/24
Expression miscompares: 10
```