Simulation 1
Dry Part

341312304	שרה גריפית
207223066	טל שמיר

2.1: Implementation of 2->1 Mux

A truth table for a mux with two inputs (d0 and d1), one selector (sel), and one output (z):

sel	d0	d1	z
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	1
1	0	0	0
1	0	1	1
1	1	1	1
1	1	0	0

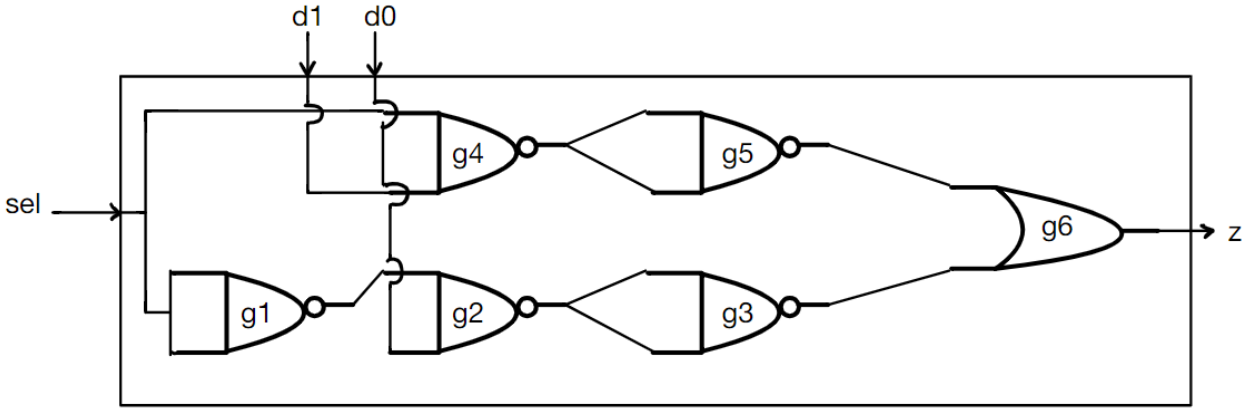
Let: w = sel, x = d0, y = d1

The following is the Karnaugh Map that corresponds to the above truth table:

w\xy	00	01	11	10
0	0	0	1	1
1	0	1	1	0

From the above truth table, we get the following minimal expression for mux: z = wy + w'x

Using the given gates, this is an implementation of mux with six logic gates:



Simulation 1
Dry Part

The following table describes the times for the different gates:

	t_{PDLH}	t_{PDHL}
NAND2	4	1
OR2	1	2
XNOR2	3	3

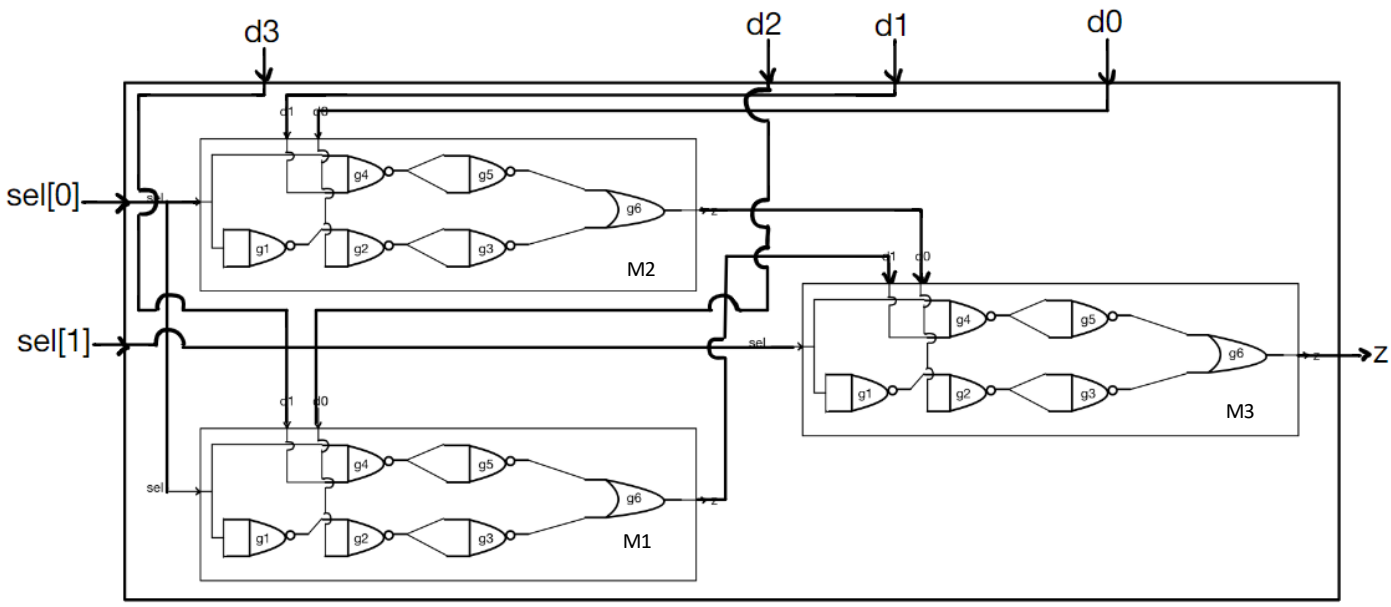
The following table describes the t_{PD} of each possible path throughout the implementation above:

Path	d0	d1	sel	t_{PD}
d0 -> g2 -> g3 -> g6	0 -> 1	0	0	$t_{PDHL}(g2) + t_{PDLH}(g3) + t_{PDLH}(g6) = 1 + 4 + 1 = 6$
	0 -> 1	1	0	$t_{PDHL}(g2) + t_{PDLH}(g3) + t_{PDLH}(g6) = 1 + 4 + 1 = 6$
	1 -> 0	0	0	$t_{PDLH}(g2) + t_{PDHL}(g3) + t_{PDHL}(g6) = 4 + 1 + 2 = 7$
	1 -> 0	1	0	$t_{PDLH}(g2) + t_{PDHL}(g3) + t_{PDHL}(g6) = 4 + 1 + 2 = 7$
d1 -> g4 -> g5 -> g6	1	0 -> 1	1	$t_{PDHL}(g4) + t_{PDLH}(g5) + t_{PDLH}(g6) = 1 + 4 + 1 = 6$
	0	0 -> 1	1	$t_{PDHL}(g4) + t_{PDLH}(g5) + t_{PDLH}(g6) = 1 + 4 + 1 = 6$
	1	1 -> 0	1	$t_{PDLH}(g4) + t_{PDHL}(g5) + t_{PDHL}(g6) = 4 + 1 + 2 = 7$
	0	1 -> 0	1	$t_{PDLH}(g4) + t_{PDHL}(g5) + t_{PDHL}(g6) = 4 + 1 + 2 = 7$
sel -> g1 -> g2 -> g3 -> g6 (longest path to z)	1	0	0 -> 1	$t_{PDHL}(g1) + t_{PDLH}(g2) + t_{PDHL}(g3) + t_{PDHL}(g6) = 1 + 4 + 1 + 2 = 8$
	1	0	1 -> 0	$t_{PDLH}(g1) + t_{PDHL}(g2) + t_{PDLH}(g3) + t_{PDLH}(g6) = 4 + 1 + 4 + 1 = 10$
sel -> g4 -> g5 -> g6 (No change on longest path)	0	1	0 -> 1	$t_{PDHL}(g4) + t_{PDLH}(g5) + t_{PDLH}(g6) = 1 + 4 + 1 = 6$
	0	1	1 -> 0	$t_{PDLH}(g4) + t_{PDHL}(g5) + t_{PDHL}(g6) = 4 + 1 + 2 = 7$

Simulation 1
Dry Part

2.2: Implementation of 4-1 mux using 2-1 mux:

Implementation of 4-1 mux using three 2-1 selectors:



The following is a table of the times of the different gates:

	t_{PDLH}	t_{PDHL}
NAND2	4	4
OR2	2	2
XNOR2	3	3

The following is a table showing the calculations of t_{PD} for changes on a selected input:

Path	d0	d1	d2	d3	sel[0]	sel[1]	t_{PD}
d0 -> NAND -> NAND -> OR -> NAND -> NAND -> OR	0 -> 1	0	0	0	0	0	20
	1 -> 0	0	0	0	0	0	20

Calculations for first change:

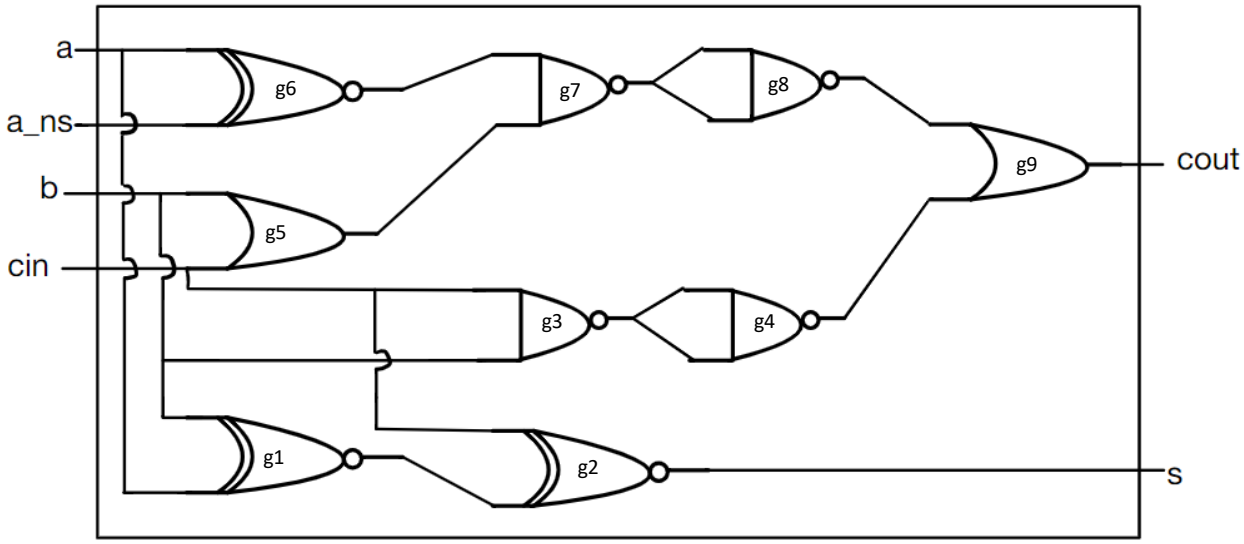
$$t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDLH}(OR) + t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDLH}(OR)$$
$$= 4 + 4 + 2 + 4 + 4 + 2 = 20$$

Calculations for second change:

$$t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR)$$
$$= 4 + 4 + 2 + 4 + 4 + 2 = 20$$

Simulation 1
Dry Part

2.3: Implementation of Full Adder/Subtracter using 9 logic gates:



The following is a table of the times used for the calculations:

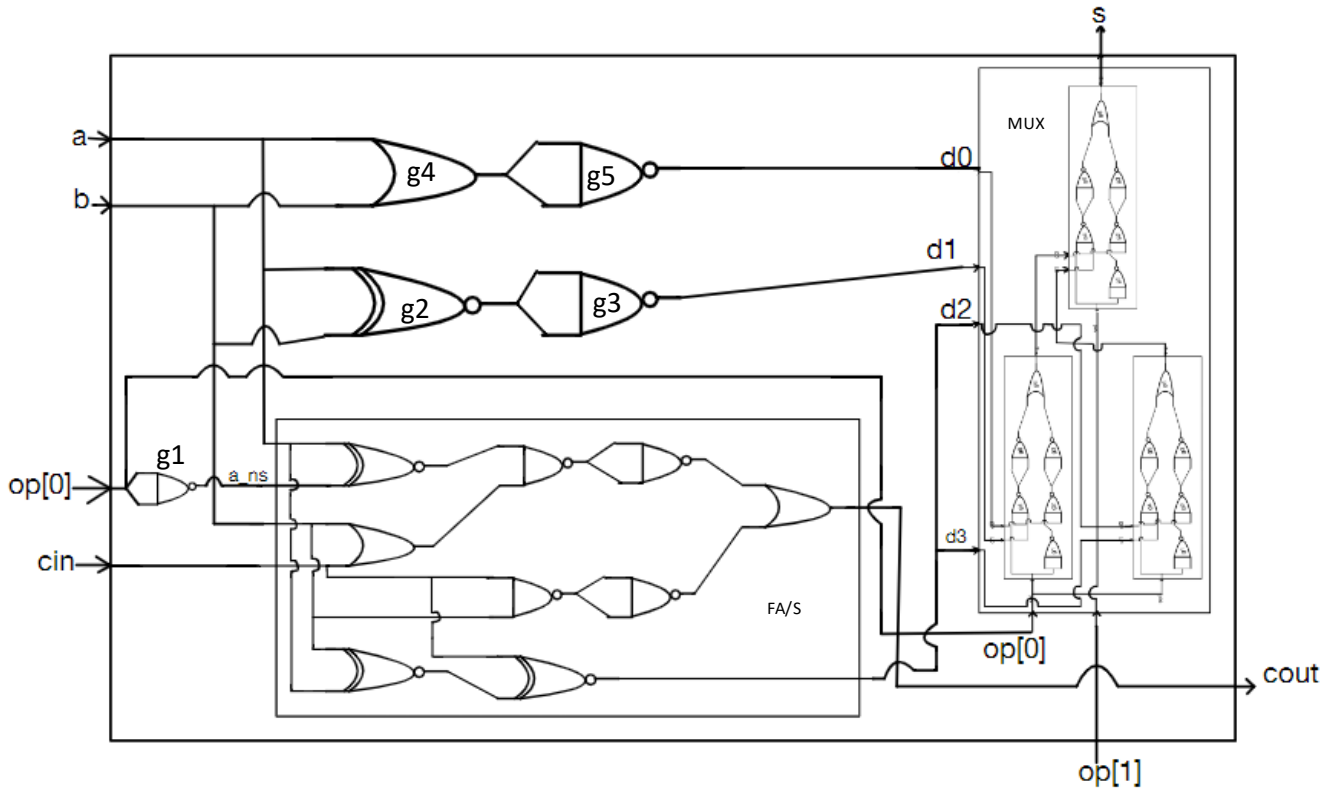
	t_{PDLH}	t_{PDHL}
NAND2	4	4
OR2	2	2
XNOR2	3	3

The following is a table showing the maximum time for each input:

Max Path	**a**	**a_ns**	**b**	**cin**	**t_{pd}**
a -> XNOR -> NAND -> NAND -> OR -> cout	0 -> 1	0	1	0	$t_{PDHL}(XNOR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) = 3+4+4+2 = 13$
a -> XNOR -> XNOR -> s	0 -> 1	0	0	0	$t_{PDHL}(XNOR) + t_{PDLH}(XNOR) = 3 + 3 = 6$
a_ns -> XNOR -> NAND -> NAND -> OR -> cout	0	0 -> 1	1	0	$t_{PDHL}(XNOR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) = 3+4+4+2 = 13$
b -> OR -> NAND -> NAND -> OR -> cout	0	0	1 -> 0	0	$t_{PDHL}(OR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) = 2+4+4+2 = 12$
b -> XNOR -> XNOR -> s	0	0	1 -> 0	0	$t_{PDLH}(XNOR) + t_{PDHL}(XNOR) = 3+3 = 6$
cin -> OR -> NAND -> NAND -> OR -> cout	0	0	0	1 -> 0	$t_{PDHL}(OR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) = 2+4+4+2 = 12$
cin -> XNOR -> s	0	0	0	1 -> 0	$t_{PDHL}(XNOR) = 3$

Simulation 1
Dry Part

2.4: Implementation of ALU using one mux, one full adder/subtracter, and 5 logic gates:



The following is a table showing the maximum time for each input and output:

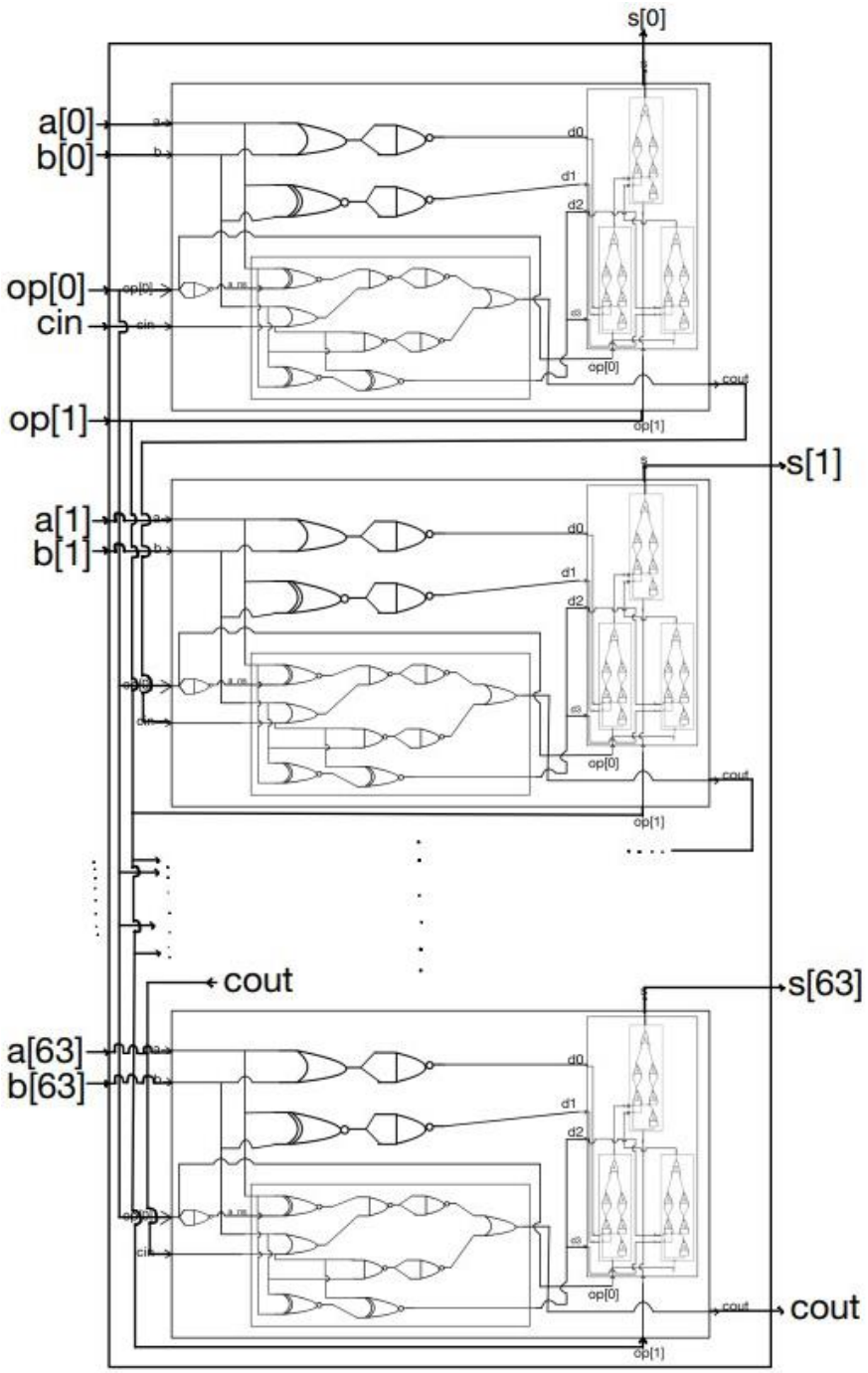
Max Path	a	b	cin	op[0]	op[1]	t_{PD}
a -> XNOR -> NAND->MUX-> s	0 -> 1	0	0	1	0	$t_{PDHL}(XNOR) + t_{PDLH}(NAND) + 2\left(t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDLH}(OR)\right)$ $= 3+4+2(4+4+2) = 27$
a -> FA/S -> cout	0 -> 1	1	0	1	0	$t_{PDHL}(XNOR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR)$ $= 3+4+4+2 = 13$
b -> XNOR -> NAND->MUX-> s	0	1-> 0	0	1	0	$t_{PDLH}(XNOR) + t_{PDHL}(NAND) + 2\left(t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR)\right)$ $=3+3+4+4+2+4+4+2=27$
b -> FA/S -> cout	0	1 -> 0	0	1	0	$t_{PDHL}(OR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR)$ $= 2+4+4+2 = 12$
cin -> FA/S -> Mux -> s	0	0	0->1	1	1	$t_{PDLH}(XNOR) + 2\left(t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDLH}(OR)\right)$ $= 3+2(4+4+2) = 23$

Simulation 1
Dry Part

cin -> FA/S -> cout	0	0	1->0	1	0	$t_{PDHL}(OR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR)$ $= 2+4+4+2 = 12$
op[0] -> Mux -> s	0	0	0	1->0	0	$t_{PDLH}(NAND) + 2\left(t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDLH}(OR)\right)$ $= 4 + 2(4+4+2) = 24$
op[0] -> NAND-> FA/s-> cout	0	1	0	1->0	1	$t_{PDLH}(NAND) + t_{PDHL}(XNOR) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDLH}(OR)$ $= 4+3+4+4+2 = 17$
op[1] -> Mux -> s	0	0	0	0	0->1	$t_{PDHL}(NAND) + t_{PDLH}(NAND) + t_{PDHL}(NAND) + t_{PDHL}(OR) =$ $4+4+4+2 = 14$

2.5: The following is an implementation of an ALU with 64-bit data inputs.

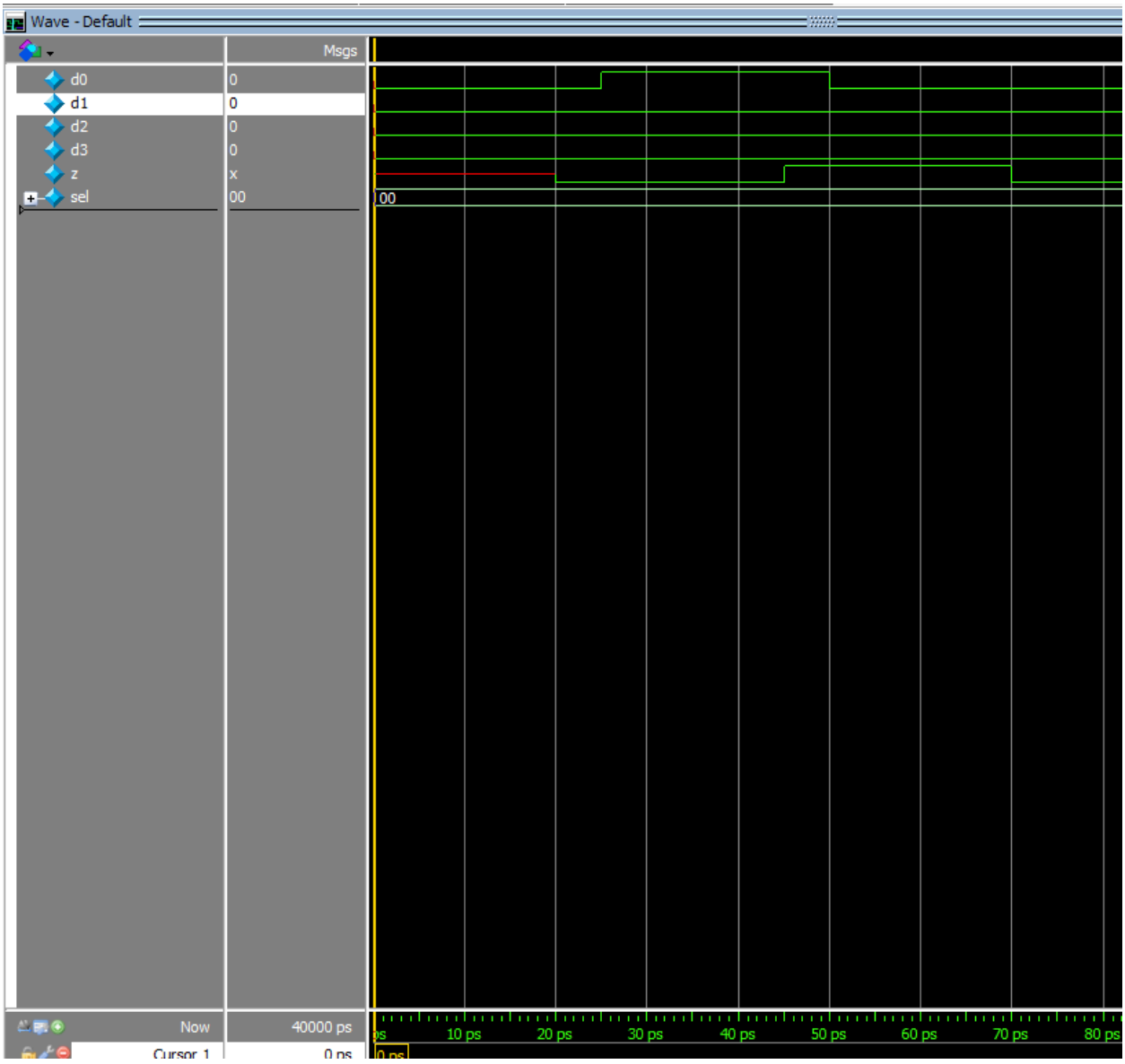
Simulation 1
Dry Part



Max Path	a	b	cin	op[0]	op[1]	t_{PD}
op[0] -> cout0 -> cin1 -> cout1 -> … -> cin63 ->s[63]	10…00	00…1	0	0 -> 1	1	$t_{PD}(op[0]->cout0) + 62t_{PD}(cin->cout) + t_{PD}(cin63->s[63]) = 17 + 62 \cdot 12 + 23 = 784$

3.3: Testbench of MUX4:

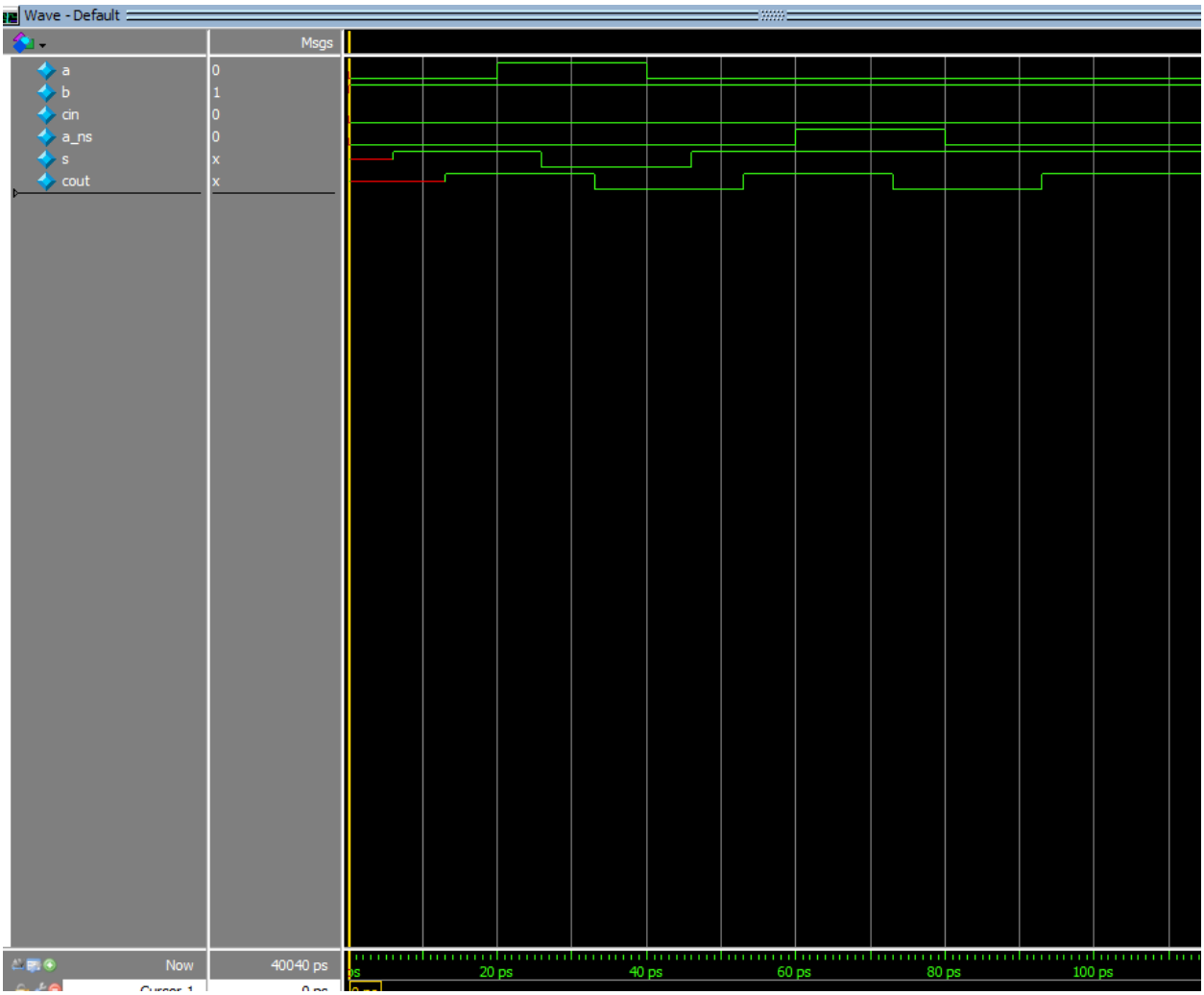
Simulation 1
Dry Part



We defined a wait time of 25 ps in our testbench, and as seen in the photo the output changes after 20 ps each time. This change matches our calculations from question 2.2. Until 20 ps, the output z was undefined (value of x) in accordance with our calculations from before. After 20 ps, the out put changed to the value of 0, according to the initial input received (all inputs are 0). At 25 ps, we changed the input and only 20 ps later, at the time 45 ps, the output changed. Lastly, at the time 50 ps, we changed the input again (back to 0) and the output changed again at the time 70 ps, again with a delay of 20 ps.

3.5: Testbench of FAS:

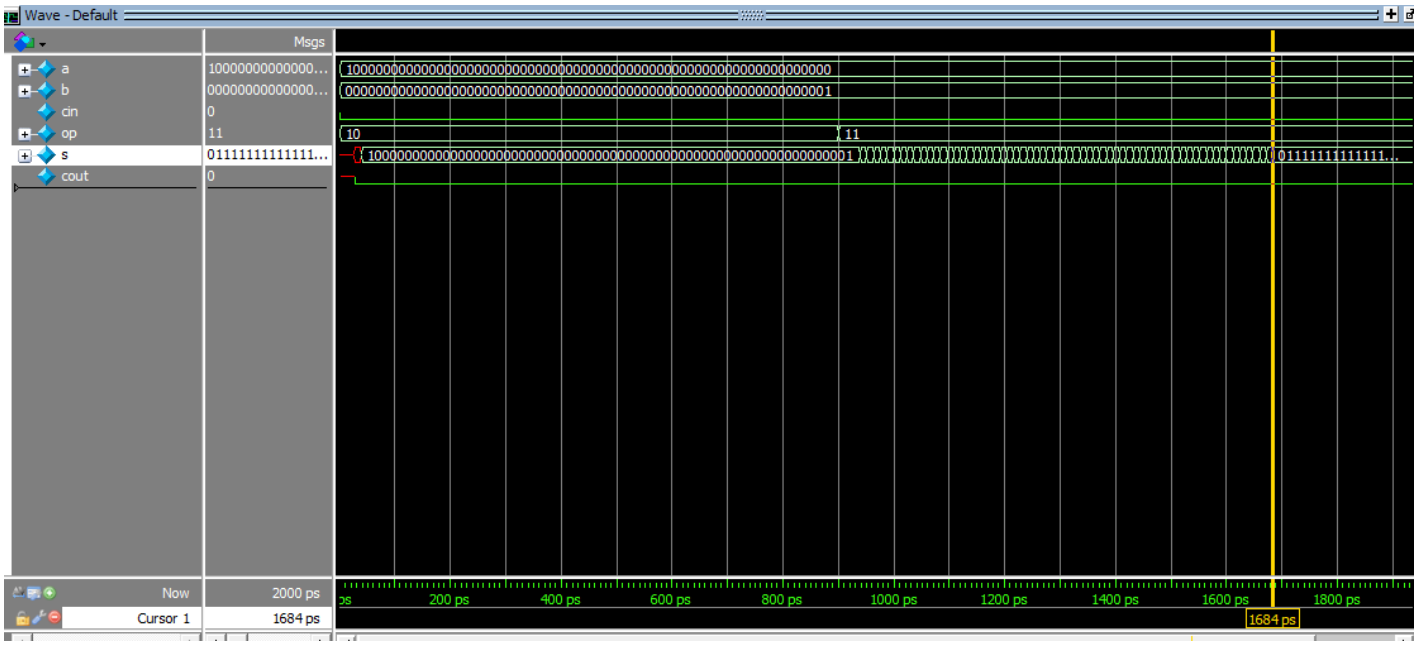
Simulation 1
Dry Part



We defined a wait time of 20 ps for our testbench. As seen in the photo, the outputs s and cout start off uninitialized (value of x). After 6 ps, s changes to match the input, and after 13 ps cout changes accordingly. These times match our calculations from question 2.3. At the time 20 ps the input a changes from 0 to 1, and at the time 26 ps, s changes accordingly. Cout changes at the time 33 ps. At the time 40 ps, we change a back to 0. At the time 46 ps s changes, and at the time 53 ps cout changes. At 60 ps, the value of a_ns is changed, and s is no longer affected. At 73 ps cout changes according to the new value of a_ns which matches our calculations. At the time 80 ps, we changed a_ns back, and cout changed after a delay of 13 ps.

3.8: Testbench of ALU64bit:

Simulation 1
Dry Part



In our testbench we defined a waiting period of 900 ps. At the time 40 ps, the values s and cout are initialized. At time 900 ps, the input is changed, and at the time 1684 ps, the value of s changes accordingly (cout has no change in value). This matches our previous calculations of the maximum tpd it takes for the outputs to change (784 ps).