|  |  |
| --- | --- |
| 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):

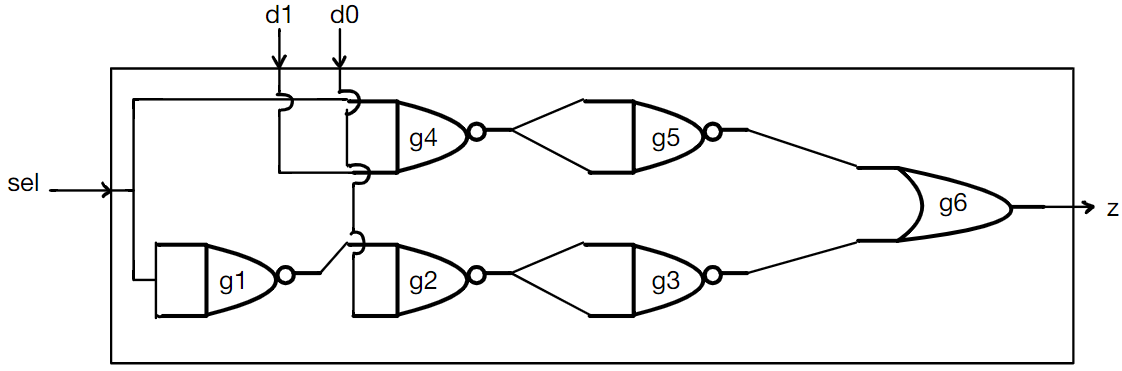
|  |  |  |  |
| --- | --- | --- | --- |
| **sel** | **d0** | **d1** | **z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 |

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:

The following table describes the times for the different gates:

|  |  |  |
| --- | --- | --- |
|  | tPDLH | tPDHL |
| NAND2 | 4 | 1 |
| OR2 | 1 | 2 |
| XNOR2 | 3 | 3 |

The following table describes the tPD of each possible path throughout the implementation above:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Path** | d0 | d1 | sel | tPD |
| d0 -> g2 -> g3 -> g6 | 0 -> 1 | 0 | 0 | t subscript P D H L end subscript open parentheses g 2 close parentheses space plus thin space t subscript P D L H end subscript open parentheses g 3 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space equals space 6 |
| 0 -> 1 | 1 | 0 | t subscript P D H L end subscript open parentheses g 2 close parentheses space plus thin space t subscript P D L H end subscript open parentheses g 3 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space equals space 6 |
| 1 -> 0 | 0 | 0 | t subscript P D L H end subscript open parentheses g 2 close parentheses plus t subscript P D H L end subscript open parentheses g 3 close parentheses plus t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 4 space plus space 1 space plus space 2 space equals space 7 |
| 1 -> 0 | 1 | 0 | t subscript P D L H end subscript open parentheses g 2 close parentheses space plus space t subscript P D H L end subscript open parentheses g 3 close parentheses space plus space t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 4 space plus thin space 1 space plus space 2 space equals space 7 |
| d1 -> g4 -> g5 -> g6 | 1 | 0 -> 1 | 1 | t subscript P D H L end subscript open parentheses g 4 close parentheses space plus space t subscript P D L H end subscript open parentheses g 5 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space equals space 6 |
| 0 | 0 -> 1 | 1 | t subscript P D H L end subscript open parentheses g 4 close parentheses space plus space t subscript P D L H end subscript open parentheses g 5 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space equals space 6 |
| 1 | 1 -> 0 | 1 | t subscript P D L H end subscript open parentheses g 4 close parentheses space plus space t subscript P D H L end subscript open parentheses g 5 close parentheses space plus space t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 4 space plus space 1 space plus space 2 space equals space 7 |
| 0 | 1 -> 0 | 1 | t subscript P D L H end subscript open parentheses g 4 close parentheses space plus space t subscript P D H L end subscript open parentheses g 5 close parentheses space plus space t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 4 space plus space 1 space plus space 2 space equals space 7 |
| sel -> g1 -> g2 -> g3 -> g6 (longest path to z) | 1 | 0 | 0 -> 1 | t subscript P D H L end subscript open parentheses g 1 close parentheses space plus space t subscript P D L H end subscript open parentheses g 2 close parentheses space plus space t subscript P D H L end subscript open parentheses g 3 close parentheses space plus space t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space plus space 2 space equals space 8 |
| 1 | 0 | 1 -> 0 | t subscript P D L H end subscript open parentheses g 1 close parentheses space plus space t subscript P D H L end subscript open parentheses g 2 close parentheses space plus space t subscript P D L H end subscript open parentheses g 3 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 4 space plus space 1 space plus space 4 space plus space 1 space equals space 10 |
| sel -> g4 -> g5 -> g6  (No change on longest path) | 0 | 1 | 0 -> 1 | t subscript P D H L end subscript open parentheses g 4 close parentheses space plus space t subscript P D L H end subscript open parentheses g 5 close parentheses space plus space t subscript P D L H end subscript open parentheses g 6 close parentheses space equals space 1 space plus space 4 space plus space 1 space equals space 6 |
| 0 | 1 | 1 -> 0 | t subscript P D L H end subscript open parentheses g 4 close parentheses space plus space t subscript P D H L end subscript open parentheses g 5 close parentheses space plus space t subscript P D H L end subscript open parentheses g 6 close parentheses space equals space 4 space plus space 1 space plus space 2 space equals space 7 |

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

Diagram

Description automatically generated Implementation of 4-1 mux using three 2-1 selectors:

M3

M2

M1

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

|  |  |  |
| --- | --- | --- |
|  | tPDLH | tPDHL |
| NAND2 | 4 | 4 |
| OR2 | 2 | 2 |
| XNOR2 | 3 | 3 |

The following is a table showing the calculations of tPD for changes on a selected input:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Path** | **d0** | **d1** | **d2** | **d3** | **sel[0]** | **sel[1]** | **tPD** |
| 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 subscript P D H L end subscript open parentheses N A N D close parentheses space plus space t subscript P D L H end subscript open parentheses N A N D close parentheses space plus space t subscript P D L H end subscript open parentheses O R close parentheses space plus space t subscript P D H L end subscript open parentheses N A N D close parentheses space plus space t subscript P D L H end subscript open parentheses N A N D close parentheses space plus space t subscript P D L H end subscript open parentheses O R close parentheses
equals space 4 space plus space 4 space plus space 2 space plus space 4 space plus space 4 space plus space 2 space equals space 20

Calculations for second change:

t subscript P D L H end subscript open parentheses N A N D close parentheses space plus space t subscript P D H L end subscript open parentheses N A N D close parentheses space plus space t subscript P D H L end subscript open parentheses O R close parentheses space plus space t subscript P D L H end subscript open parentheses N A N D close parentheses space plus space t subscript P D H L end subscript open parentheses N A N D close parentheses space plus space t subscript P D H L end subscript open parentheses O R close parentheses
equals space 4 space plus space 4 space plus space 2 space plus space 4 space plus space 4 space plus space 2 space equals space 20

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

Diagram

Description automatically generated

g9

g8

g7

g6

g5

g4

g3

g2

g1

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

|  |  |  |
| --- | --- | --- |
|  | tPDLH | tPDHL |
| NAND2 | 4 | 4 |
| OR2 | 2 | 2 |
| XNOR2 | 3 | 3 |

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

Due to the fact that the tpd for low to high and high to low are equal for each logic gate, the times for changes from low to high and high to low are equal. Therefore, we will only write one of the two options in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Max Path** | **a** | **a\_ns** | **b** | **cin** | **tpd** |
| a -> XNOR -> NAND -> NAND -> OR -> cout | 0 -> 1 | 0 | 1 | 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 3 plus 4 plus 4 plus 2 space equals space 13 |
| a -> XNOR -> XNOR -> s | 0 -> 1 | 0 | 0 | 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses X N O R close parentheses space equals space 3 space plus space 3 space equals space 6 |
| a\_ns -> XNOR -> NAND -> NAND -> OR -> cout | 0 | 0 -> 1 | 1 | 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 3 plus 4 plus 4 plus 2 space equals space 13 |
| b -> OR -> NAND -> NAND -> OR -> cout | 0 | 0 | 1 -> 0 | 0 | t subscript P D H L end subscript open parentheses O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 2 plus 4 plus 4 plus 2 space equals space 12 |
| b -> XNOR -> XNOR -> s | 0 | 0 | 1 -> 0 | 0 | t subscript P D L H end subscript open parentheses X N O R close parentheses plus t subscript P D H L end subscript open parentheses X N O R close parentheses equals space 3 plus 3 space equals space 6 |
| cin -> OR -> NAND -> NAND -> OR -> cout | 0 | 0 | 0 | 1 -> 0 | t subscript P D H L end subscript open parentheses O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 2 plus 4 plus 4 plus 2 space equals space 12 |
| cin -> XNOR -> s | 0 | 0 | 0 | 1 -> 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses space equals space 3 |

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

Diagram

Description automatically generated

g5

g4

g3

g2

g1

MUX

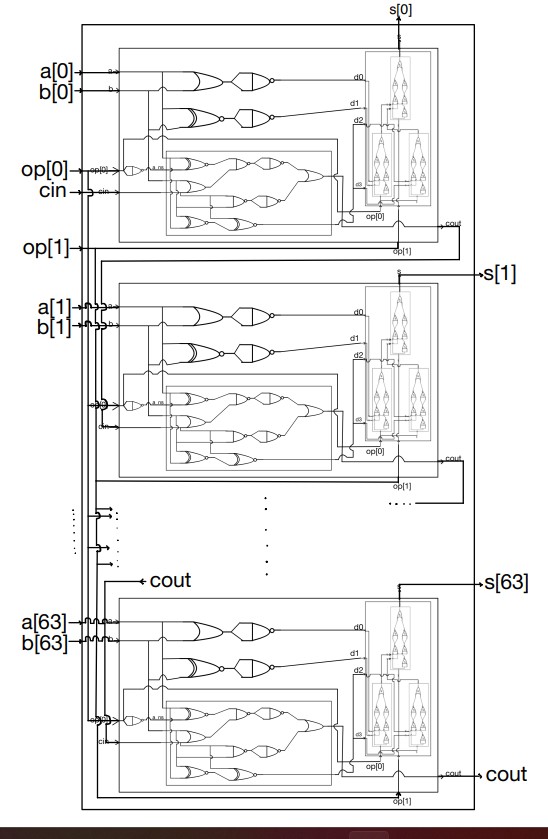
FA/S

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

Due to the fact that the tpd for low to high and high to low are equal for each logic gate, the times for changes from low to high and high to low are equal. Therefore, we will only write one of the two options in the table below.

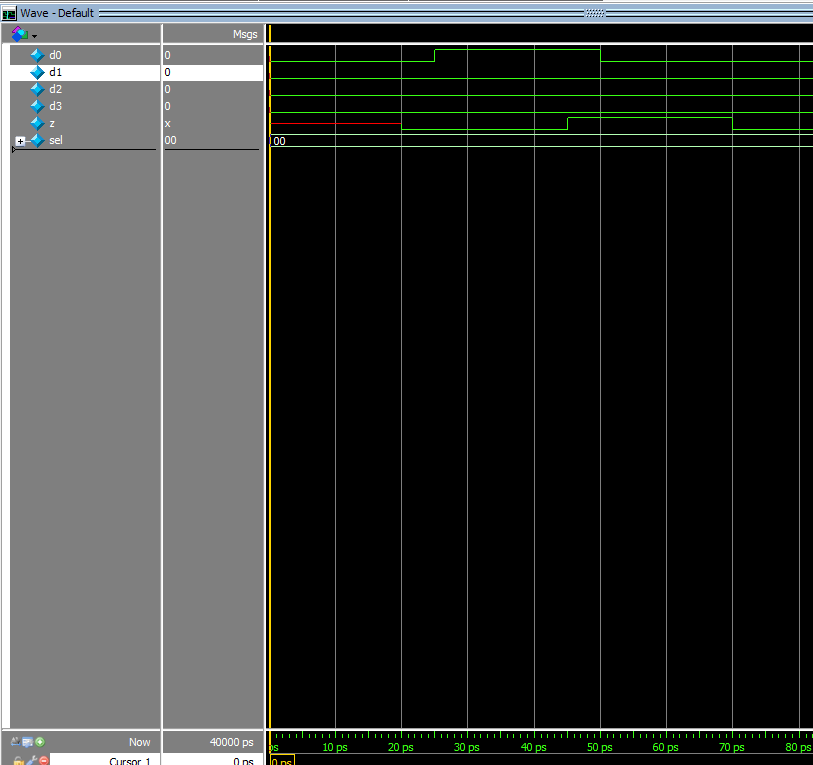
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Max Path** | **a** | **b** | **cin** | **op[0]** | **op[1]** | **tPD** |
| a -> XNOR -> NAND->MUX-> s | 0 -> 1 | 0 | 0 | 1 | 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus 2 open parentheses t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses O R close parentheses close parentheses equals space 3 plus 4 plus 2 open parentheses 4 plus 4 plus 2 close parentheses space equals space 27 |
| a -> FA/S -> cout | 0 -> 1 | 1 | 0 | 1 | 0 | t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 3 plus 4 plus 4 plus 2 space equals space 13 |
| b -> XNOR -> NAND ->MUX-> s | 0 | 1-> 0 | 0 | 1 | 0 | t subscript P D L H end subscript open parentheses X N O R close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus 2 open parentheses t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses close parentheses equals 3 plus 3 plus 4 plus 4 plus 2 plus 4 plus 4 plus 2 equals 27 |
| b -> FA/S -> cout | 0 | 1 -> 0 | 0 | 1 | 0 | t subscript P D H L end subscript open parentheses O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 2 plus 4 plus 4 plus 2 space equals space 12 |
| cin -> FA/S -> Mux -> s | 0 | 0 | 0->1 | 1 | 1 | t subscript P D L H end subscript open parentheses X N O R close parentheses plus 2 open parentheses t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses O R close parentheses close parentheses equals space 3 plus 2 open parentheses 4 plus 4 plus 2 close parentheses equals space 23 |
| cin -> FA/S -> cout | 0 | 0 | 1->0 | 1 | 0 | t subscript P D H L end subscript open parentheses O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals space 2 plus 4 plus 4 plus 2 space equals space 12 |
| op[0] -> Mux -> s | 0 | 0 | 0 | 1->0 | 0 | t subscript P D L H end subscript open parentheses N A N D close parentheses plus 2 open parentheses t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses O R close parentheses close parentheses equals space 4 space plus space 2 open parentheses 4 plus 4 plus 2 close parentheses space equals space 24 |
| op[0] -> NAND-> FA/s-> cout | 0 | 1 | 0 | 1->0 | 1 | t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses X N O R close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses O R close parentheses equals space 4 plus 3 plus 4 plus 4 plus 2 space equals space 17 |
| op[1] -> Mux -> s | 0 | 0 | 0 | 0 | 0->1 | t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D L H end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses N A N D close parentheses plus t subscript P D H L end subscript open parentheses O R close parentheses equals 4 plus 4 plus 4 plus 2 space equals space 14 |

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



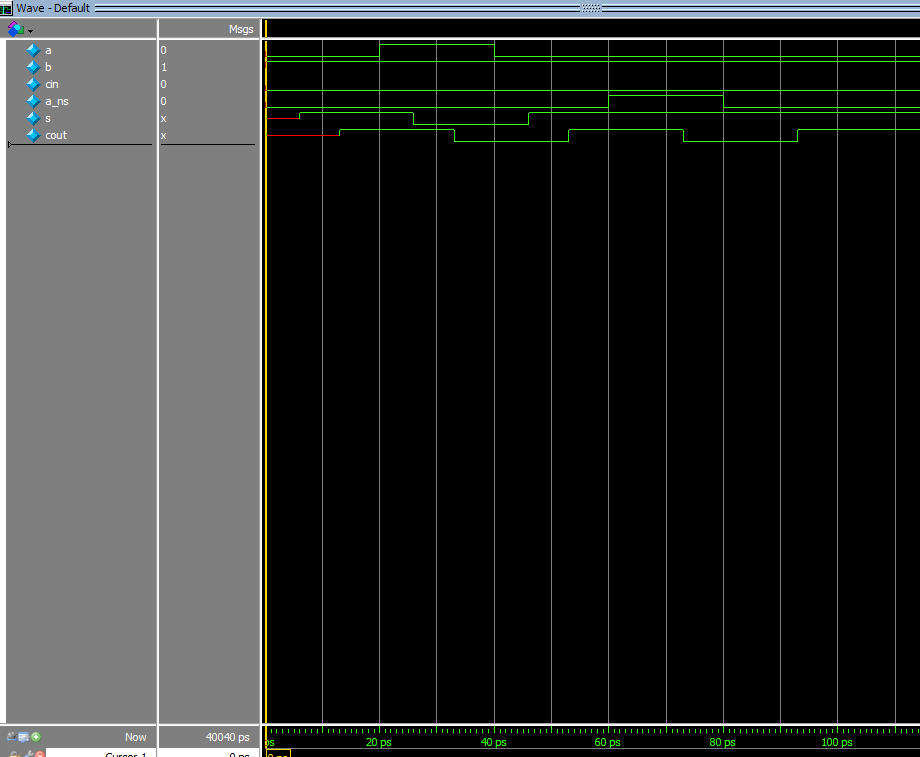
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Max Path** | **a** | **b** | **cin** | **op[0]** | **op[1]** | **tPD** |
| op[0] -> cout0 -> cin1 -> cout1 -> … -> cin63 ->s[63] | 10…00 | 00…1 | 0 | 0 -> 1 | 1 | t subscript P D end subscript open parentheses o p open square brackets 0 close square brackets minus greater than c o u t 0 close parentheses plus space 62 t subscript P D end subscript open parentheses c i n minus greater than c o u t close parentheses space plus space t subscript P D end subscript open parentheses c i n 63 minus greater than s open square brackets 63 close square brackets close parentheses equals space 17 plus 62 times 12 space plus space 23 space equals space 784 |

3.3: Testbench of MUX4:



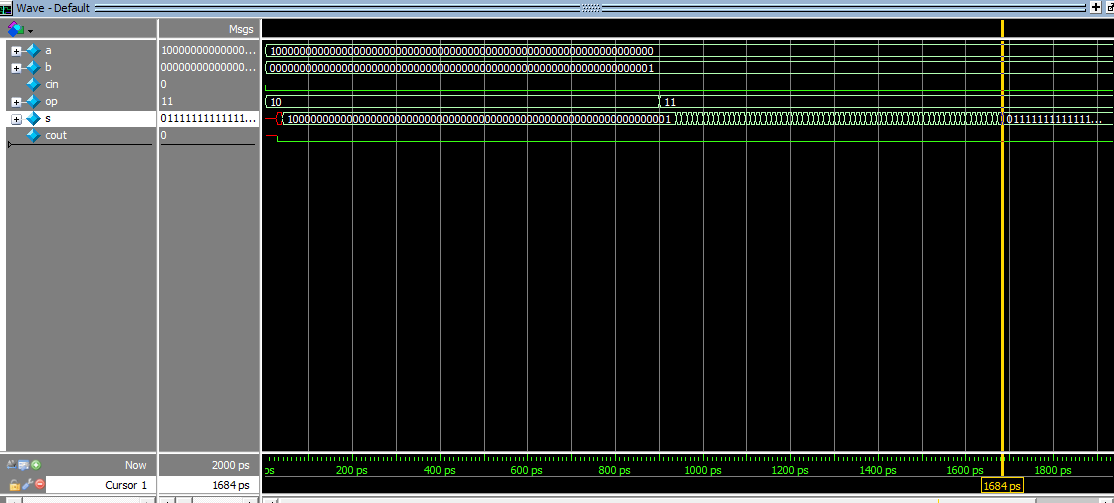
We defined a wait time of 25 ps in our testbench, and as seen in the photo the output changes after 20 ps each time. This change matches our calculations from question 2.2. Until 20 ps, the output z was undefined (value of x) in accordance with our calculations from before. After 20 ps, the output changed 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:



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:



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 output s to change (784 ps).