תרגיל בית רטוב מספר 1 מערכות ספרתיות ומבנה מחשב

|  |  |
| --- | --- |
| 342816097 | גיליאן בן שושן |
| 330083858 | ראובן טימסיט |

**2. dry part**

**2.1**

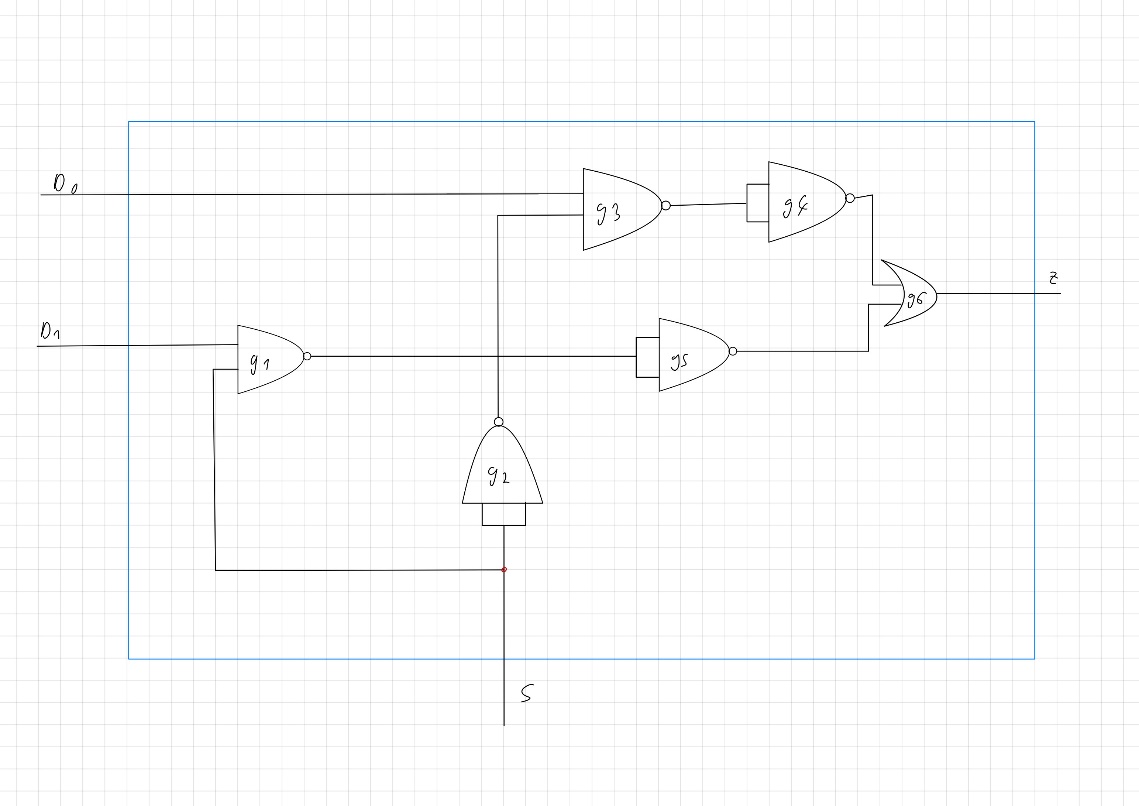
A B C D E F G H I

3 4 2 8 1 6 10 9 7

|  |  |  |
| --- | --- | --- |
|  | tPDLH | tPDHL |
| NAND2 | B=4 | E=1 |
| OR2 | C=2 | F=6 |
| XNOR2 | D=8 | G=10 |

We need to design MUX 2->1

So we need :



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Path gilian ruben** | **d0** | **d1** | **s** | **z** | **Tpd** |
| 1) d0->g3->g4->g6->Z | 0->1 | 0/1 | 1 | 0->1 | 13 / 7 |
| 2) d0->g3->g4->g6->Z | 1->0 | 0/1 | 1 | 1->0 | 13 / 11 |
| 3) s->g1->g5->g6->Z | 1 | 0 | 0->1 | 1->0 | 9 / 12 |
| 4) s->g1->g5->g6->Z | 0 | 1 | 1->0 | 1->0 | 12 / 11 |
| 5) s->g2->g3->g4->g6->Z | 0 | 1 | 0->1 | 0->1 | 13 / 7 |
| 6) s->g2->g3->g4->g6->Z | 1 | 0 | 1->0 | 1->0 | 13 / 11 |
| 7) d1->g2->g5->g7->g8->Z | 0/1 | 0->1 | 0 | 0->1 | 13 / 7 |
| 8) d1->g2->g5->g7->g8->Z | 0/1 | 1->0 | 0 | 1->0 | 13 / 11 |

**2.2**

We now working with those tpd

|  |  |
| --- | --- |
|  | **TPD** |
| **NAND** | 4 |
| **OR** | 6 |
| **XNOR** | 10 |

We want to create a mux 4->1 with 3 or less mux 2->1

The first mux take s[0] and chose between d0 and d1. 0 will chose d0 and 1 will chose d1

The second mux take s[0] and chose between d2 and d3. 0 will chose d2 and 1 will chose d3

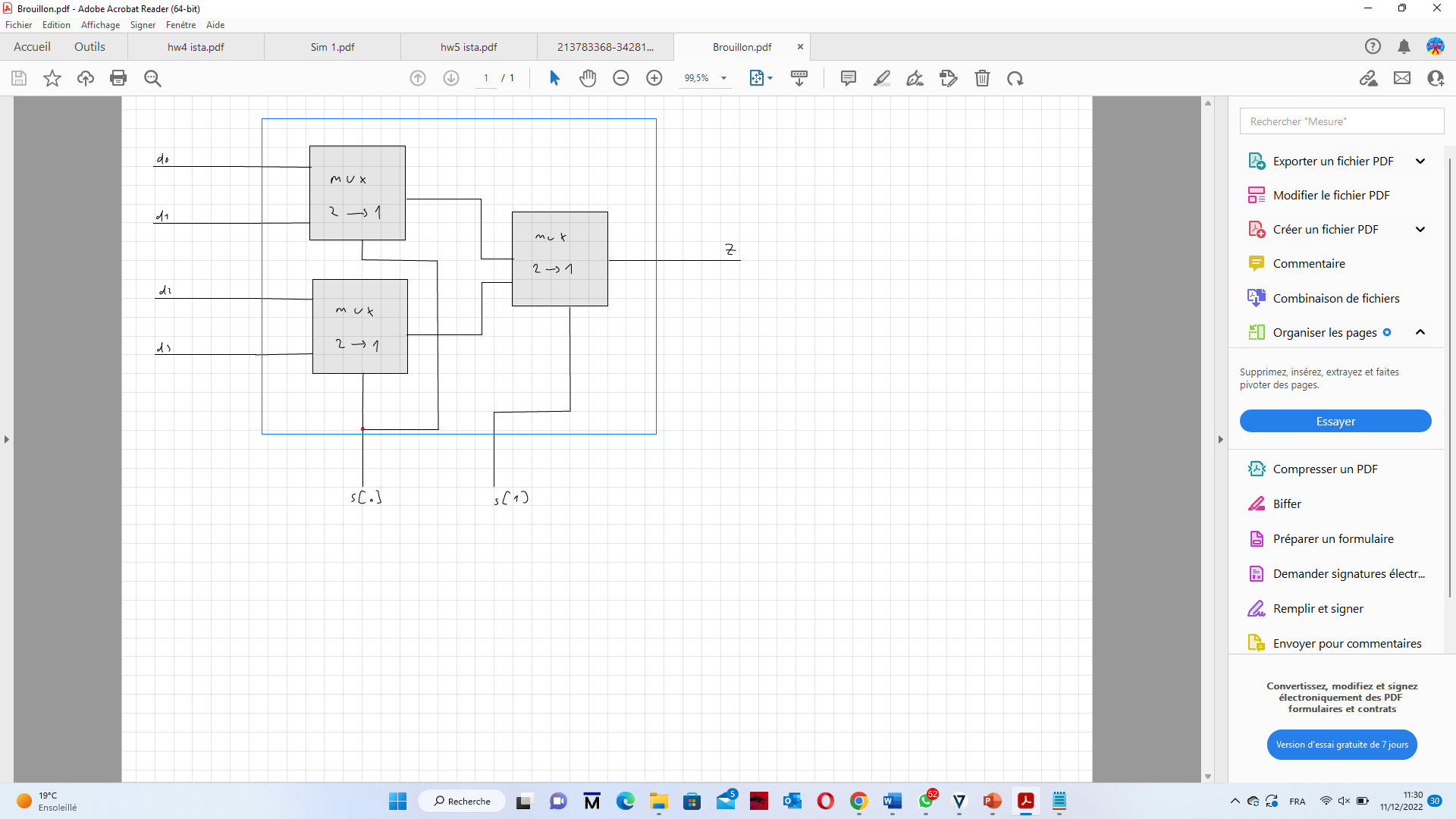
The third mux take s[1] and chose between the output of the two first. 0 will chose the output of d0 and d1 and 1 will chose d1 the output between d2 and d4

If S=00 the first mux will chose d0 with 0 and the second one will take s[1]=0 and the output will be d0

If S=01 the first mux will chose d1 with 1 and the second one will take s[1]=0 and the output will be d1

If S=10 the first mux will chose d2 with 0 and the second one will take s[1]=1 and the output will be d2

If S=11 the first mux will chose d3 with 1 and the second one will take s[1]=1 and the output will be d3



We are choosing the d1 entry and set all of the other to 0 and S=01

.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tpd** | **sel** | **d3** | **d2** | **d1** | **d0** | **Path** |
| 14+14=28 | 01 | 0 | 0 | 0->1 | 0 | d1(1)->g2(1)->g5(1)->g7(1)->g8(1)->Z(1)->d1(2)->g2(2)->g5(2)->g7(2)->g8(2)->Z |
| 14+14=28 | 01 | 0 | 0 | 1->0 | 0 | d1(1)->g2(1)->g5(1)->g7(1)->g8(1)->Z(1)->d1(2)->g2(2)->g5(2)->g7(2)->g8(2)->Z |

D1 0->1 : We need to use 2mux so d1 enter in the first one and the Tpd=14 (because we want to use max of tpd of each gate) and then he enter in the second mux and the Tpd is also 14 so total Tpd=28

D1 1->0 : In the same way we’re calculating the Tpd of the way of d1 form the input to the output and then total Tpd=28

**2.3**

We will built the Karnau map of COUT according to her truth table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ans,cin  b,a | **00** | 01 | 11 | 10 |
| 00 |  | 1 |  |  |
| 01 |  |  | 1 |  |
| 11 |  | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 |  |

So we have:

COUT = cin.b + ans’.cin.a’ + ans.cin.a + ans.b.a + ans’.b.a’

We have just the gate NAND, XNOR, and OR. So we can do:

𝑐𝑜𝑢𝑡=𝑁𝐴𝑁𝐷(𝑁𝐴𝑁𝐷(𝑏,𝑐𝑖𝑛),𝑁𝐴𝑁𝐷(𝑂𝑅(𝑏,𝑐𝑖𝑛),𝑋𝑁𝑂𝑅(𝑎,𝑎𝑛\_𝑠))

We will built the Karnau map of S according to her truth table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ans,cin  b,a | **00** | 01 | 11 | 10 |
| 00 |  | 1 | 1 |  |
| 01 | 1 |  |  | 1 |
| 11 |  | 1 | 1 |  |
| 10 | 1 |  |  | 1 |

About the exit S, we can see that the output is the same with addition or subtraction for the same input.  
So we studied in Lecture that it means that : S = XOR( XOR (a,b), cin )= XNOR( XNOR (cin,b), a )

Our solution contain 7 gates so respect the condition of the exercise.







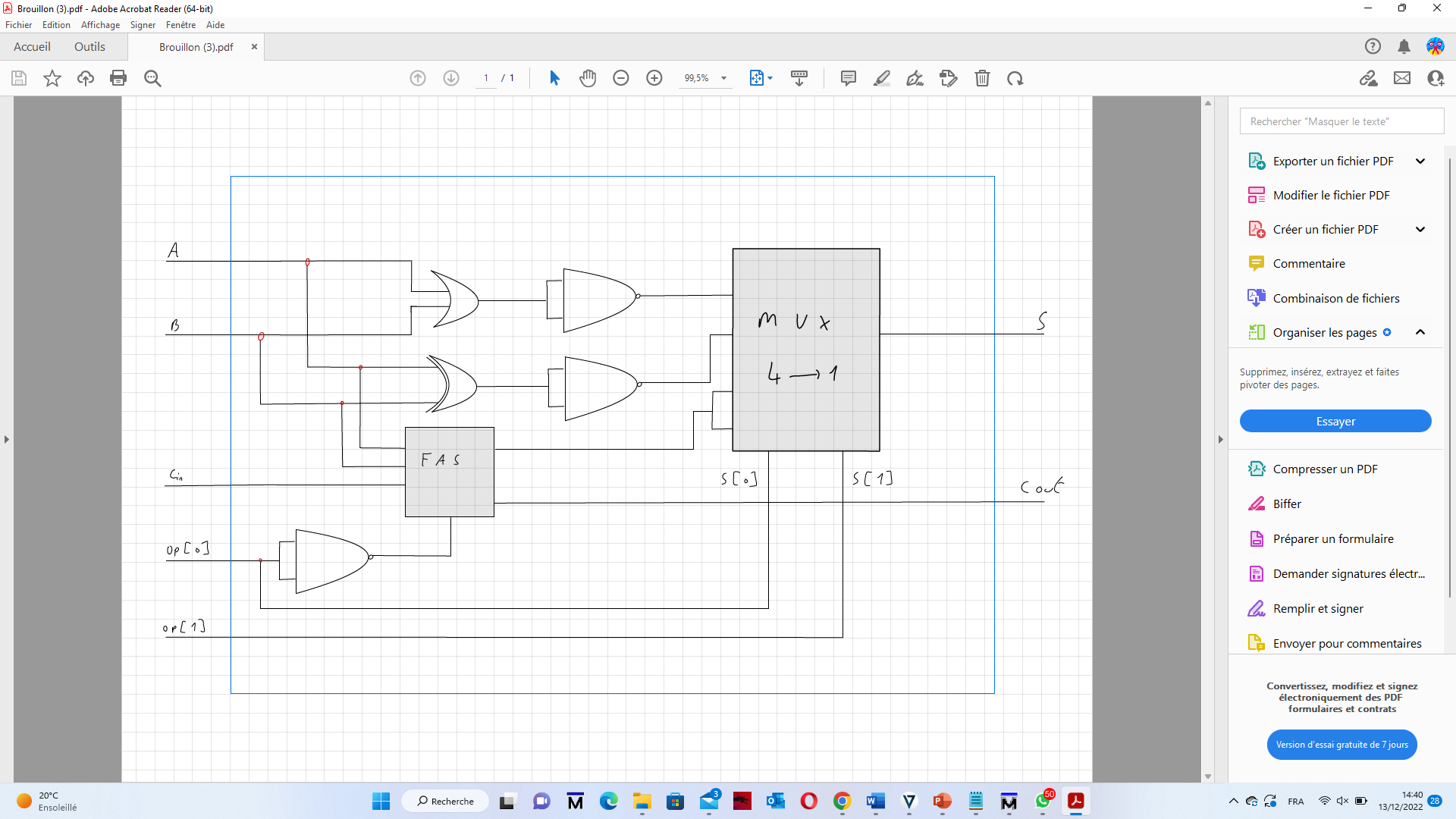
And we obtain:



|  |  |
| --- | --- |
| **PATH** | **TPD max** |
| a -> g6 -> S | 10 |
| a -> g1 -> g3 -> g5 -> g7 -> COUT | 18 |
| b -> g2 -> g6 -> S | 20 |
| b -> g3 -> g5 -> g7 -> COUT | 14 |
| cin -> g2 -> g6 -> S | 20 |
| cin -> g5 -> g7 -> COUT | 14 |
| ans -> S | 0 |
| ans -> g1 -> g3 -> g5 -> g7 -> COUT | 18 |

2.4



We use the gates, one FAS and one mux4->1 as asked.



1- a and b enter the gates OR and then the result enters NAND (which gives NOR) which gives at the end (a||b)’ .

2- a and b enter the gate XNOR then the result enters NAND (which gives XOR) give aꚚb !

3- a, b and cin enter FAS with op[0] as ans and gives the addition or the subtraction of a and b according to the opposite value of op[0] ,which gives also the value cout .

If op[0]=0 then it’ll add a and b. If op[1]=1 then it’ll substrate b to a.

4- Then we use a mux 4->1 with entries d0=(a||b)’  d1= aꚚb  d3=result of FAS(a,b,cin)

The mux will get out s equal to :

d0=(a||b)’  if op[1 :0]=00 .

d1= aꚚb   if op[1 :0]=01.

d3=a+b if op[1 :0]=10.

d4=a-b if op[1 :0]=11.

and cout as the result of the FAS .

Maximal tpd of every path:

|  |  |
| --- | --- |
| Path | Tpd |
| a->s | 42 |
| b->s | 48 |
| cin->s | 48 |
| a->cout | 18 |
| b->cout | 14 |
| cin->cout | 14 |
| op->s | 28 |
| op->cout | 22 |

1. a->s = 42

Tpd g4= XNOR = 10

Tpd g5= NAND= 4

Tpd MUX 0->1 entry d1 = 28

1. b->s = 48

Tpd FAS b->S 20

Tpd MUX 0->1 entry d1 = 28

1. cin->s = 48

Tpd FAS cin->s = 20

Tpd MUX 0->1 entry d3 = 28

1. a->cout= 18

Tpd FAS a->cout= 18

1. b->cout=14

Tpd FAS b->cout=14

1. cin->cout= 14

Tpd FAS cin->cout= 14

1. op[1 :0]->s=28

In function FAS we can see the ans doesnt change the output s so the tpd is 0.

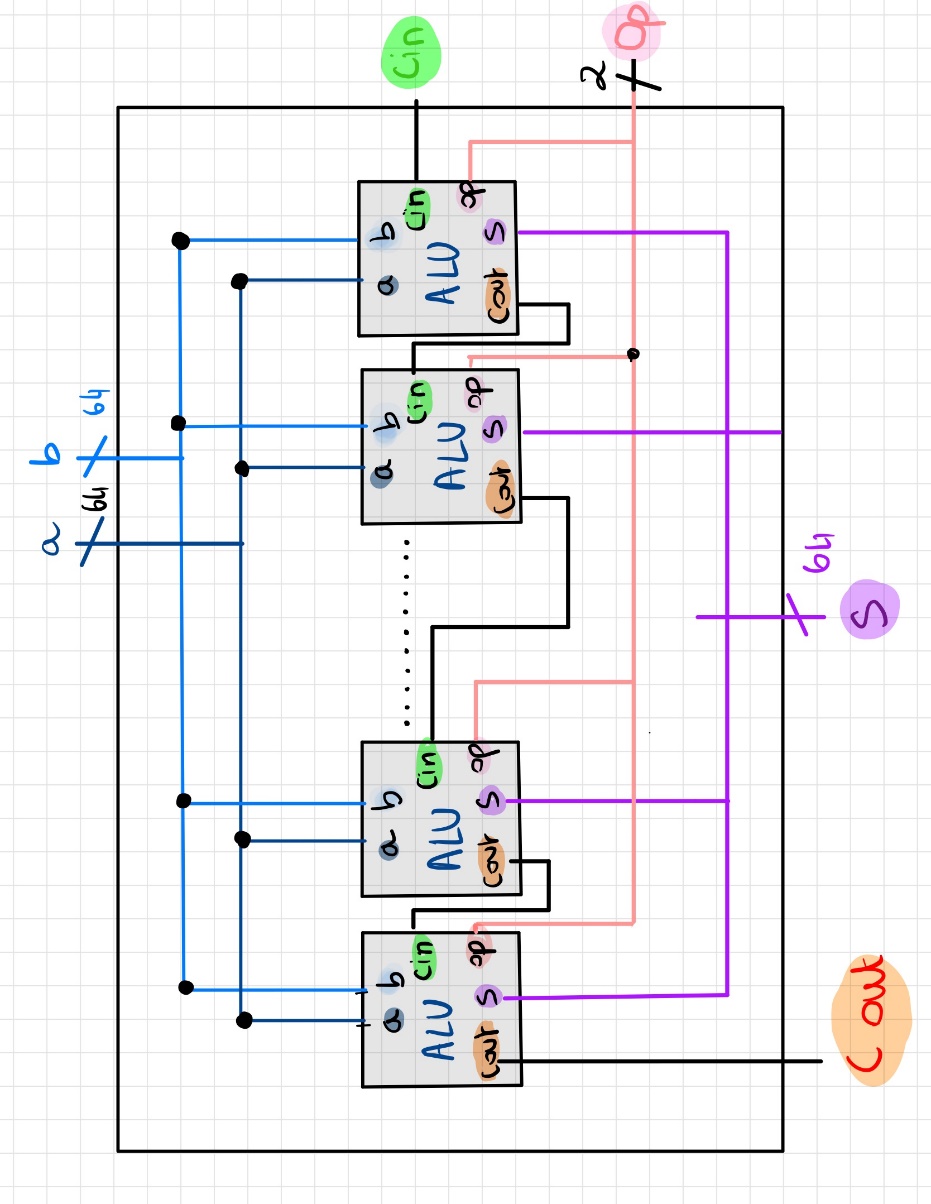
So the only Tpd comes from the mux4. d2 and d3 are the same so the first mux2 doesnt change. But if we change from d1 to d0 the first mux2 changes and the last too so the mux changes from value 1 to 0 for the values of a,b chosen. Tpd=22\*2

1. op[1 :0]->cout=22

tpd NAND=4

tpd FAS ans->cout=18

**2.5**



We will built ALU of the size of 64 bits, with the help of what we built in question 2.4 and with modules that in our system as we can see in the schema.

The TPD max have a path split in 3 parts.

* The first part is the one who begin with OP and finish in COUT of the first module. (22)
* The second part is the 62 elements of ALU who are going through CIN and finish in COUT. (62\*14)
* The third part is the path who begin with the element CIN of the last module, and finish in S of the same module. 48

To conclude the TPD max is:

Tpd ( op->cout ) + 62\*tpd ( cin->cout ) + tpd ( cin->sum ) = 28 +62\*14 + 48 = 938.

**WET PART**

**3.1** cf file mux2.sv

**3.2** cf file mux4.sv

**3.3**

We choose the input:

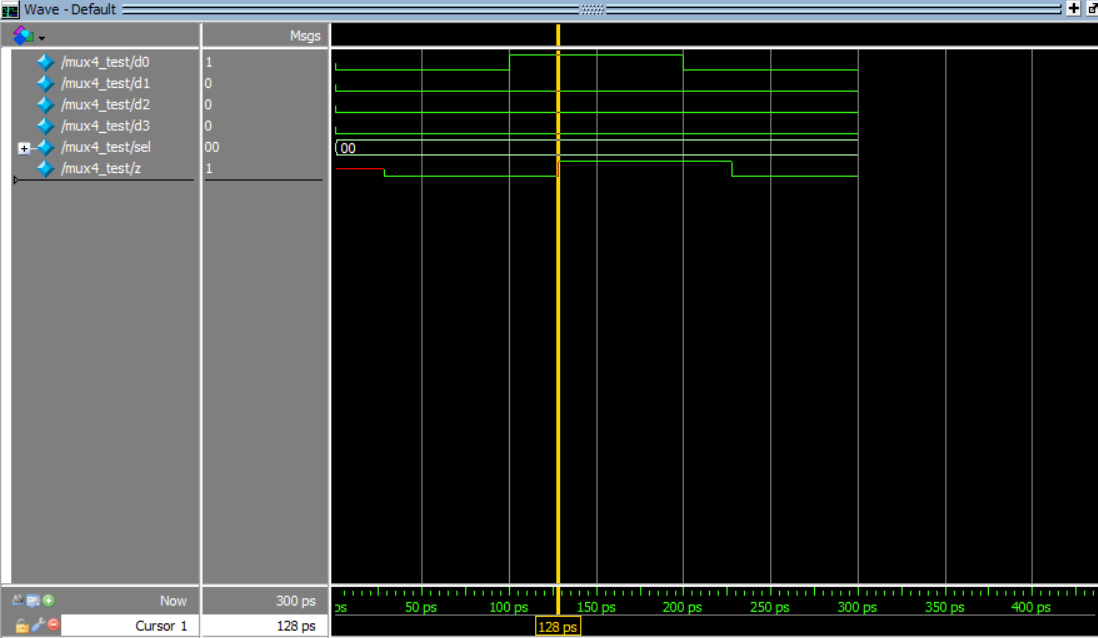
D0=D1=D2=D3=sel[0]=sel[1]=0

We are switching D0 to 1. In question 2.2 we calculated that the max time is 28.

We put the same input in ModelSim.

* We can see that in time 100 tps, d0 switch from0 to 1. And we see that the switch is happened in time 128 tps. So there is 28 tps between them.
* We can see that in time 200 tps, d0 switch from 1 to 0. And we see that the switch is happened in time 228 tps. So there is 28 tps between them.

And it’s exactly what we have calculated in question 2.2 .





**3.4** cf file fas.sv

**3.5**

**Path A🡪SUM :**

We choose the input:

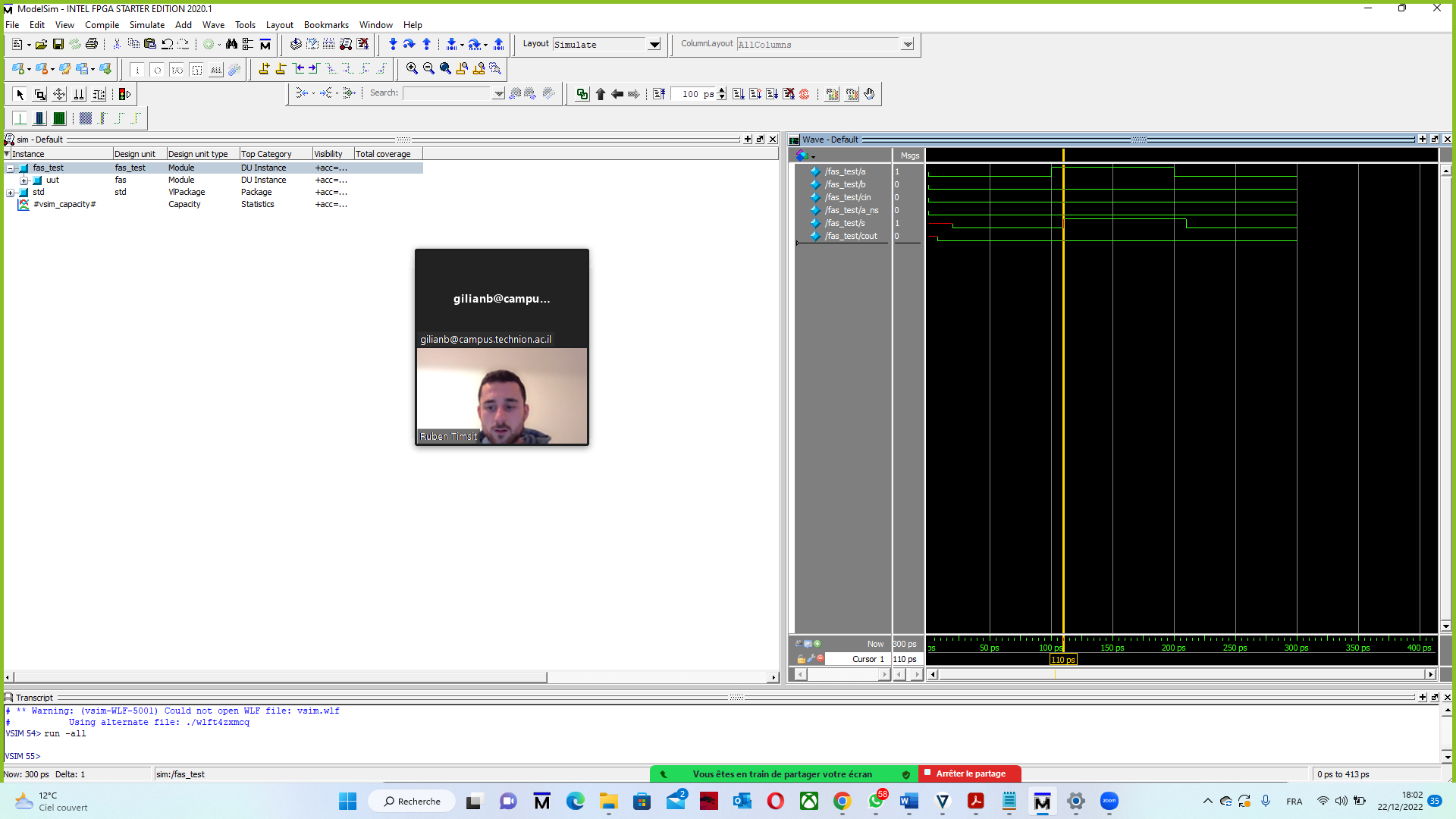
A=B=CIN=ANS=0

We are switching A to 1. In question 2.3 we calculated that the max time is 10.

We put the same input in ModelSim.

* We can see that in time 100 tps, A switch from0 to 1. And we see that the exit SUM switch in time 110 tps. So there is 10 tps between them.
* We can see that in time 200 tps, A switch from 1 to 0. And we see that the exit SUM switch in time 210 tps. So there is 10 tps between them.

And it’s exactly what we have calculated in question 2.3



**Path B🡪SUM :**

We choose the input:

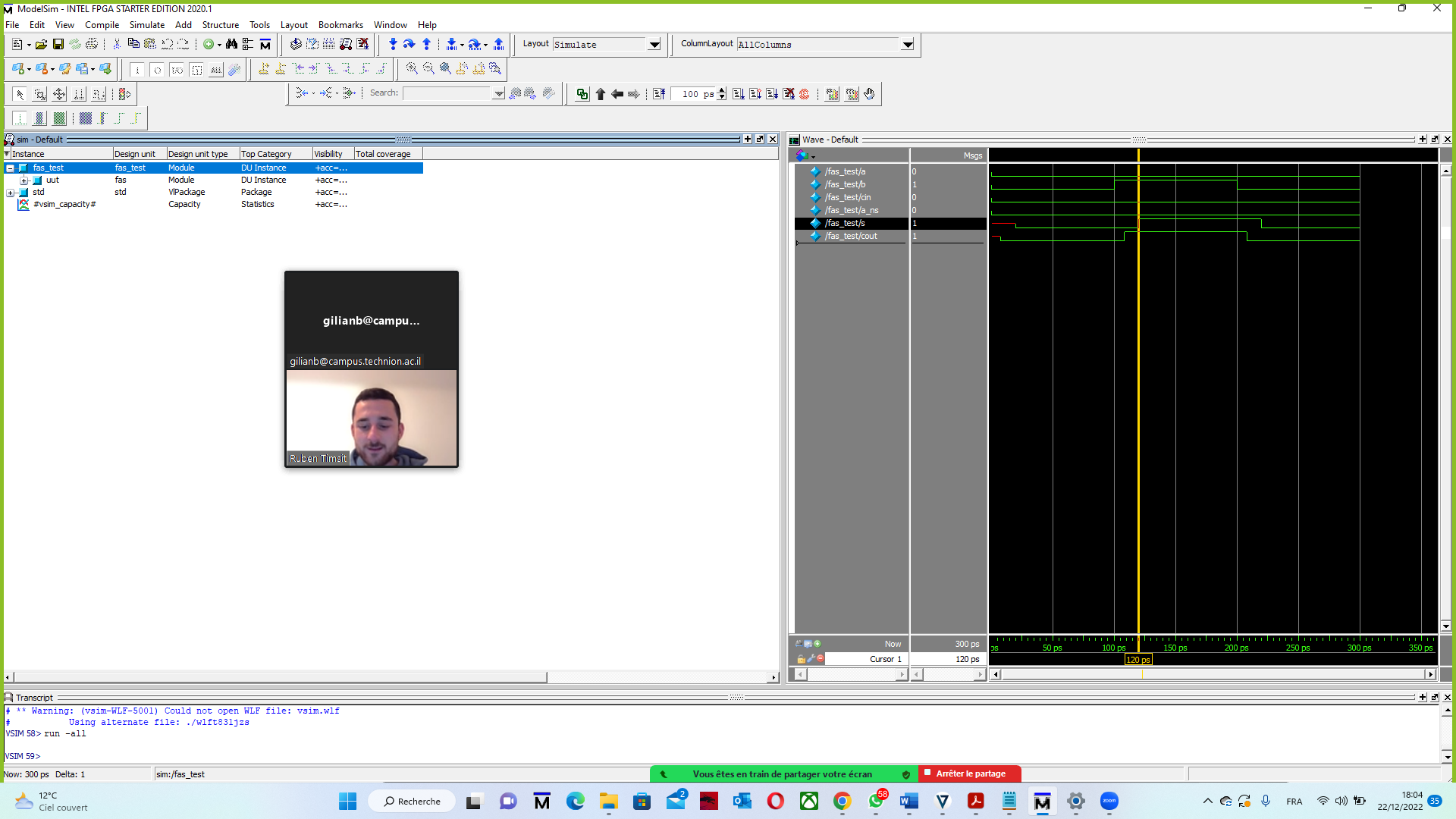
A=B=CIN=ANS=0

We are switching B to 1. In question 2.3 we calculated that the max time is 20.

We put the same input in ModelSim.

* We can see that in time 100 tps, B switch from0 to 1. And we see that the exit SUM switch in time 120 tps. So there is 20 tps between them.
* We can see that in time 200 tps, B switch from 1 to 0. And we see that the exit SUM switch in time 220 tps. So there is 20 tps between them.

And it’s exactly what we have calculated in question 2.3

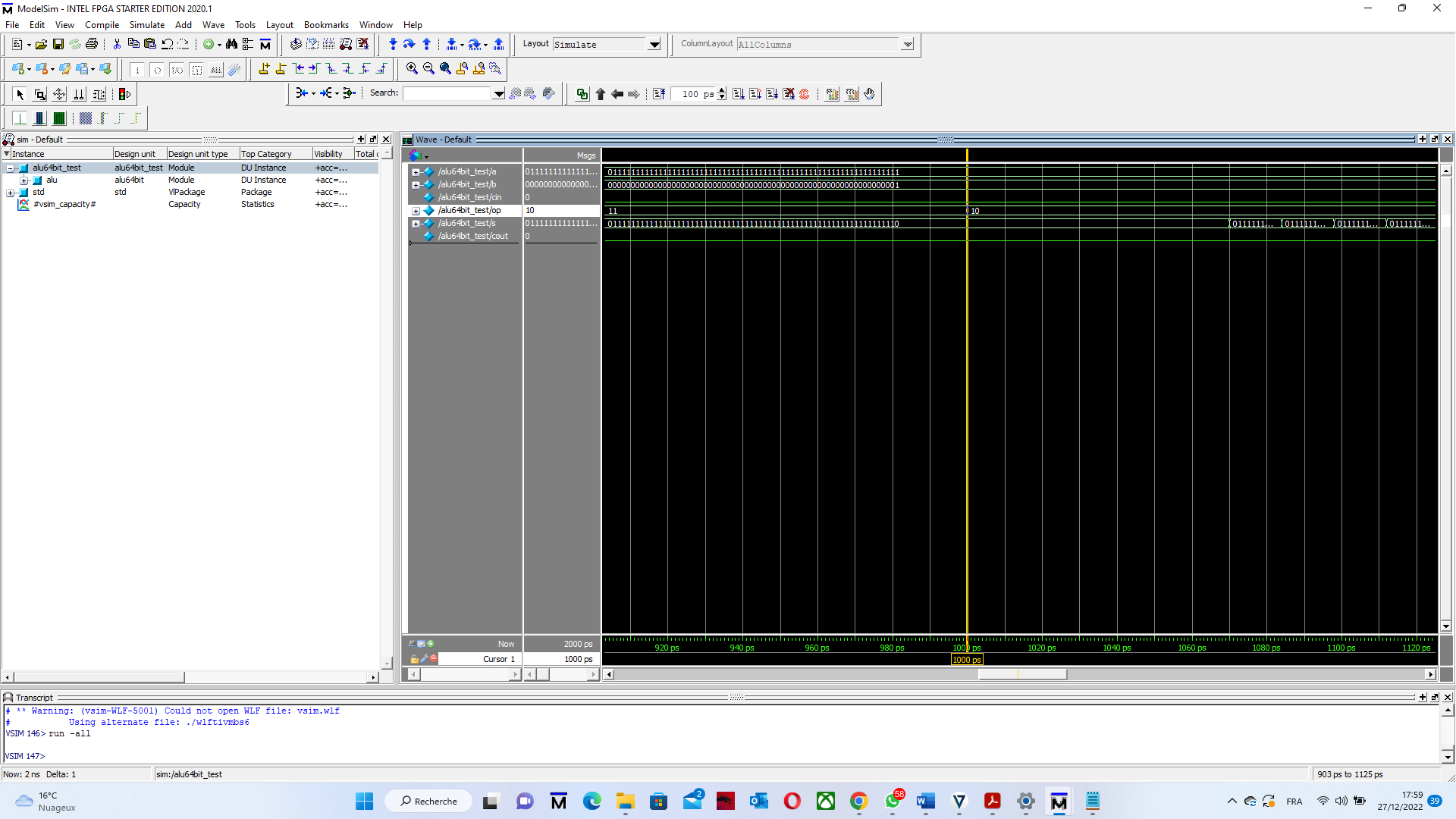


**3.6** cf file alu1bit.sv

**3.7** cf file alu64bit.sv

**3.8**

We took the same enters as we said in 2.5. we can see here that we are waiting for 1000 times unit until the op field change from 11 to 10 we want to change the op field because this is the field that have the longest tpd in alu1bit



We can see that after op change from 11 to 10 we have to wait 938 times(1938-1000) unit more until the exit s change to something stable. And this is exactly what we have calculated in 2.5.

