HY330 – VLSI Digital systems

Exercise set 3

13/12/2019

Βαγενάς Αναστάσης

Exercise 1

In this exercise we want to implement a complete article with the following portals:

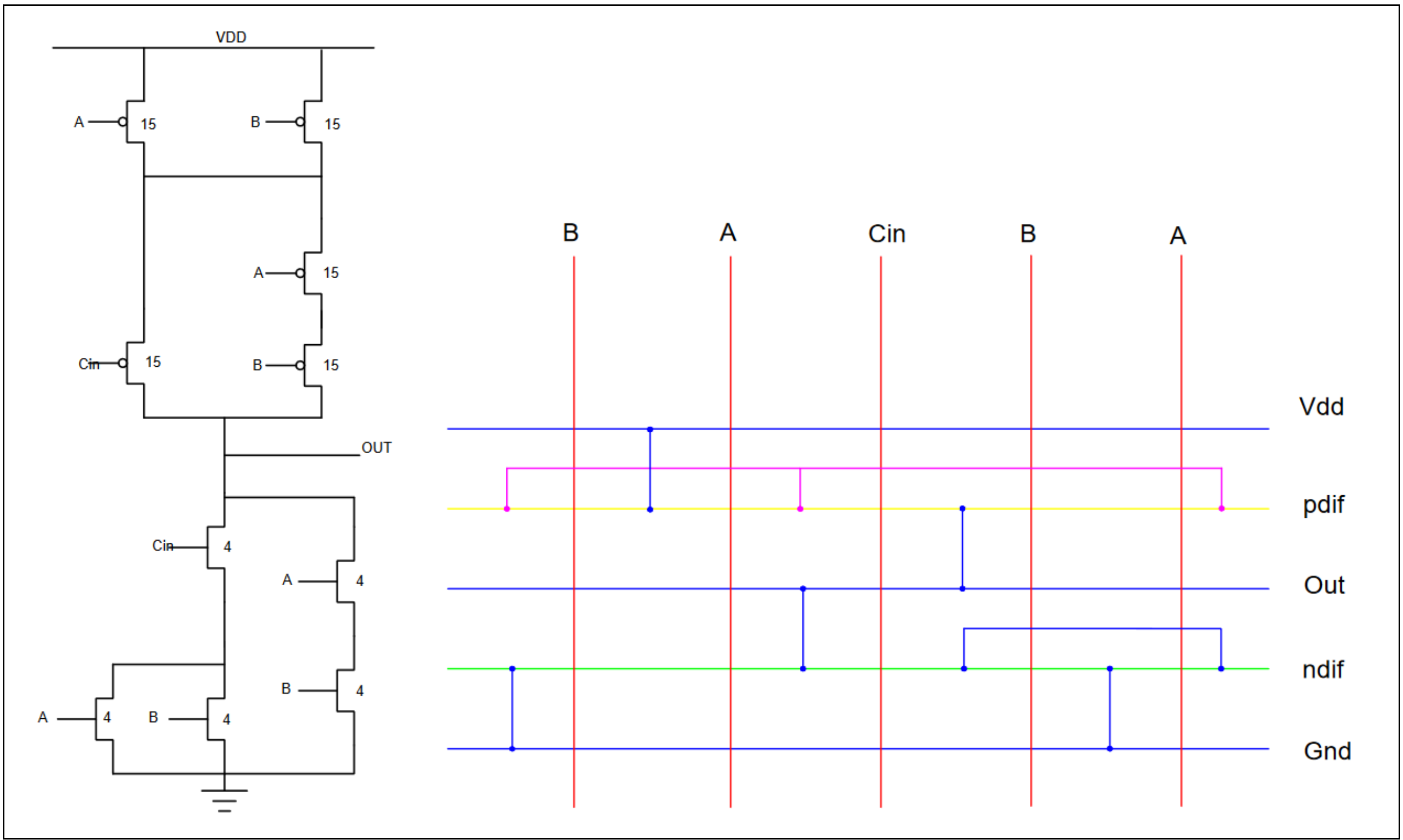
C'out= (AB+ACin+BCin)'

Cout= (C'out)'

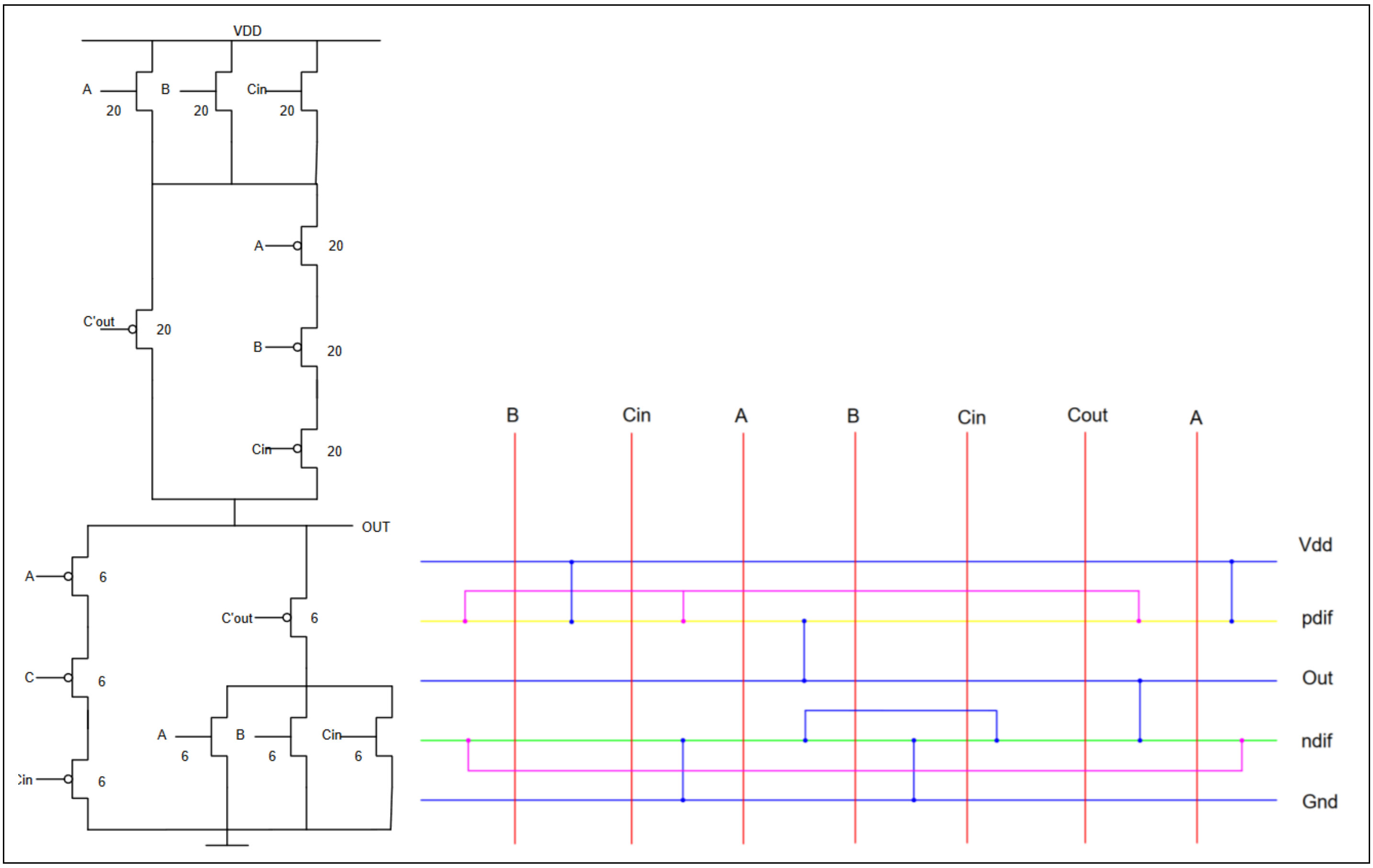
S'out= (ABCin+C'out(A+B+Cin))'

Sout= (S'out)'

Schematic and Transistor sizing/Diagram design



Schematic for C'out



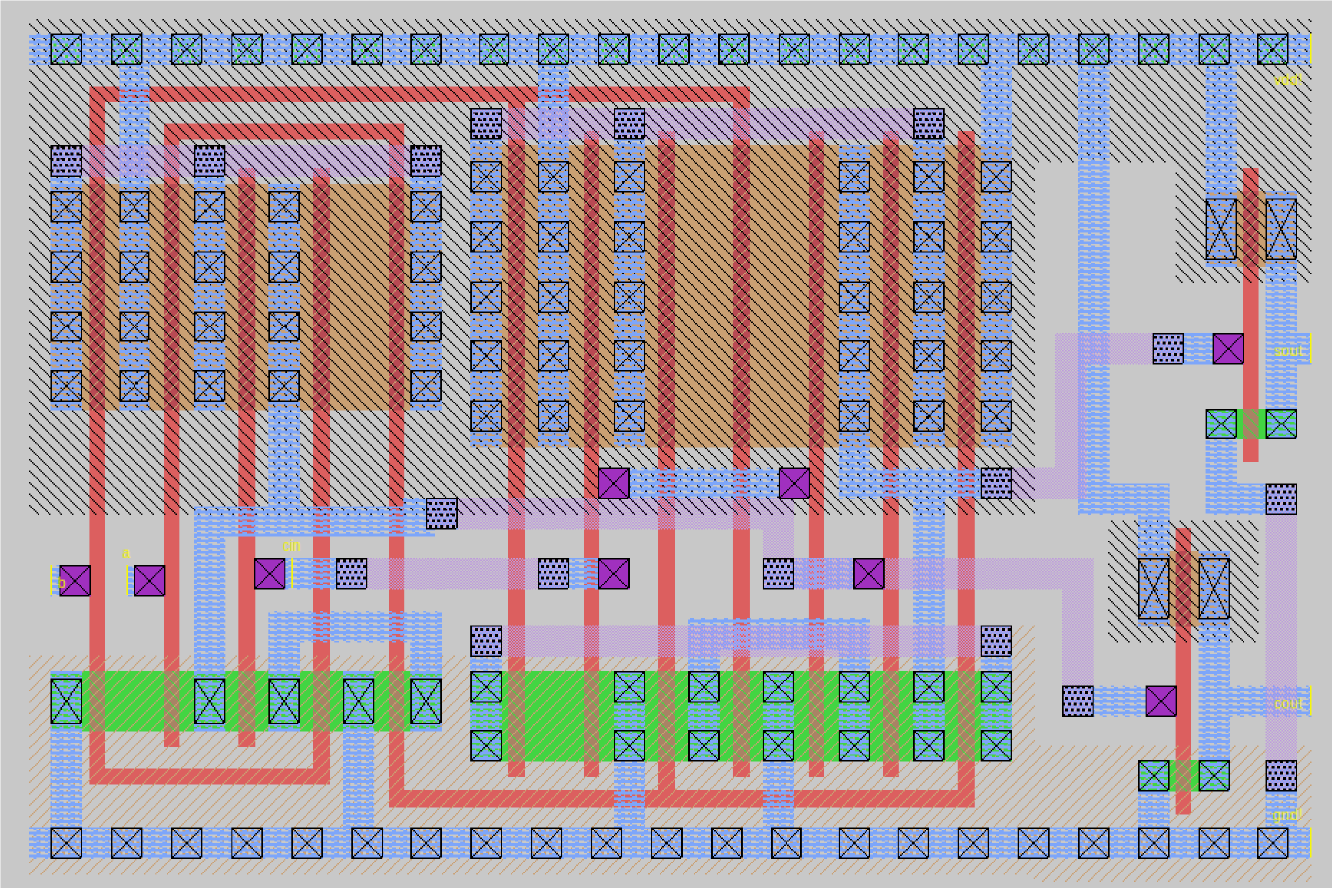
Schematic for S'out

For the whole circuit we need a total of 28 transistors.

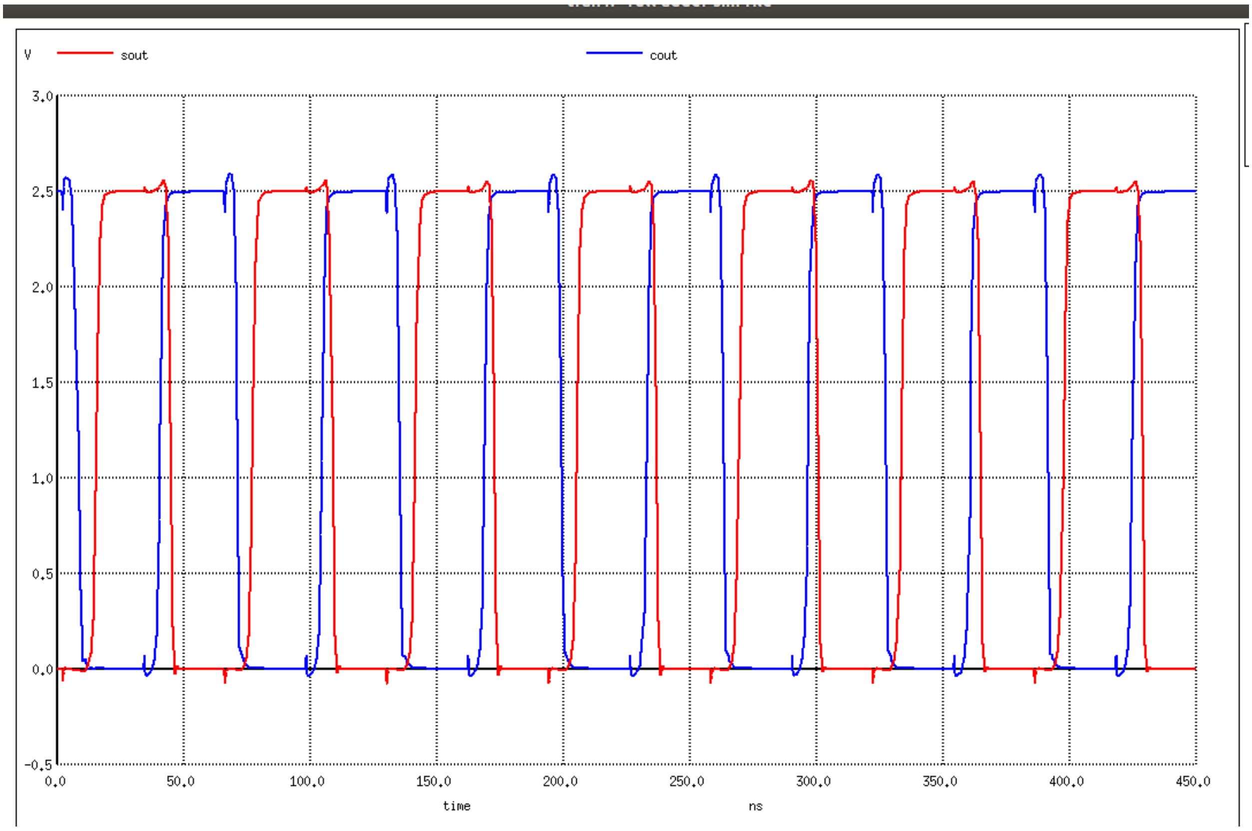
Από αυτά:

- The 4 are for the inverters for *S'out, C'out* and produce *Sout* και *Cout.*
- The 14 for *S'out.*
- The last 10 for *C'out* (we reduced the function from $AB+AC_{in}+BC_{in}$ to $AB+(A+B)C_{in}$.

Floorplan



Above the schematic of the full adder one bit.



Above the results of the simulation for input to the articulator A = 1, B = 0 and pulse Cin.

As Cin goes up and down we also see alternations in Sout and Cout.

Specifically, Cin starts from 0 and since A = 1 we will have output Sout = 1 and Cout = 0.

Conversely for Cin = 1 since A = 1 then we will have output Sout = 0 and Cout = 1, which is done in each case in the simulation.

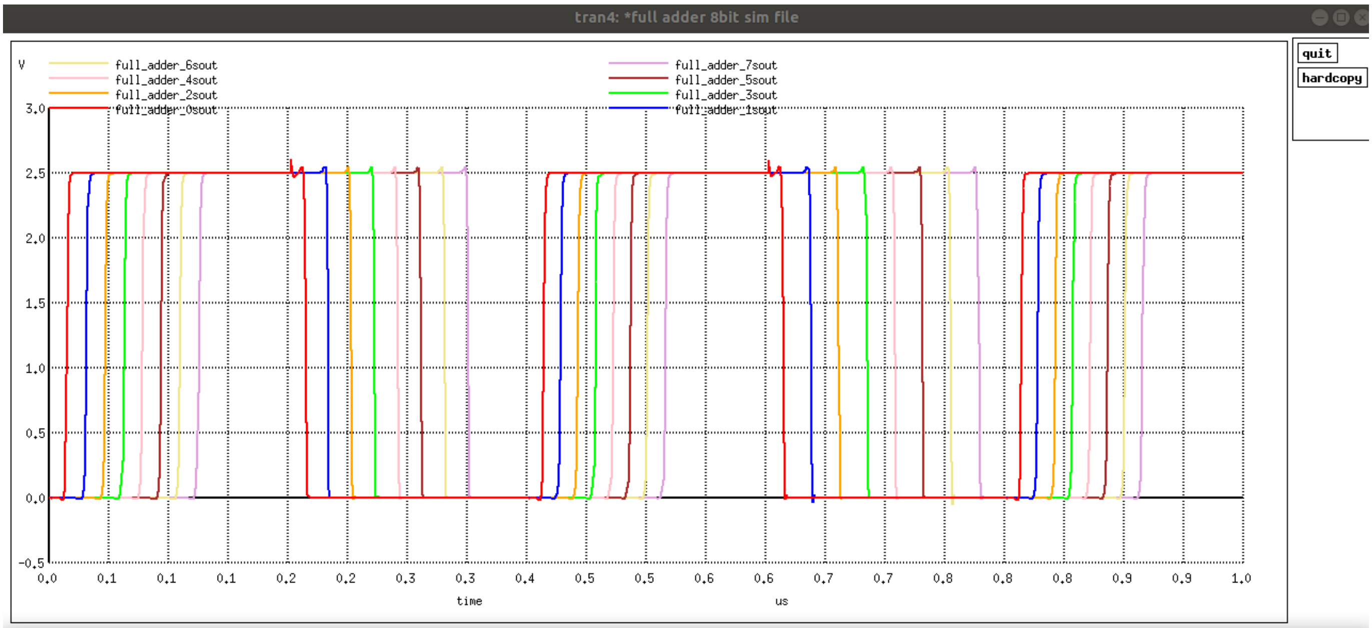
It is important to mention that as we expected Cout precedes Sout in the alternations since it is the signal that has the potential to affect Sout. Hence the delay that exists between the alternations.

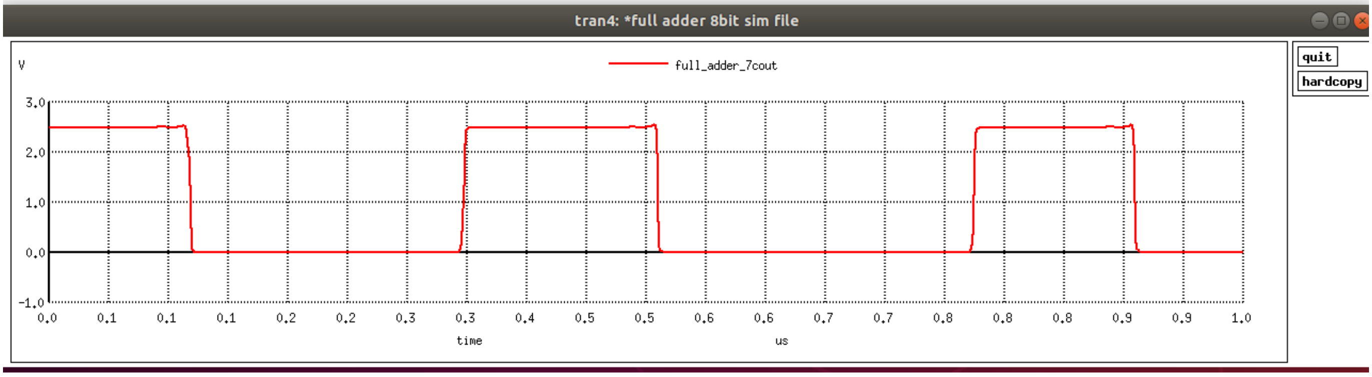
Floorplan (for 8bit)

For this part we made the appropriate connections between Cin and Cout and the intermediate adders.

So the only exceptions are Cin of full_adder_0 cell and the Cout of full_adder_7 cell (more and better can be seen in the .mag file that will be accompanied by the report).

Down below the results of the simulation:





First, we add the numbers A = 11111111, B = 0000000 (0/1) and Cin = 0 where essentially the LSB of B goes from 0 to 1. We do this because it will help us to have overflow and every bit of it Sout to change state.

As shown in the pictures we first have (at time 0) we have B = 00000001 so all Sout [7: 0] = 0 and Cout = 1 which is done. Then after the LSB of B falls to 0 we will have a reset of Sout [7: 0] to 1. Indeed from full_adder_0 (the LSB of the adder) we see the change in Sout and the each Sout up to full_adder_7 / Sout we have a change to 1. Finally, full_adder_7 / cout also goes to 0 (it happens slightly before the change of Sout to full_adder_7)

Exercise 2

In Exercise 2 we have to design and simulate a pass logic latch at spice level.

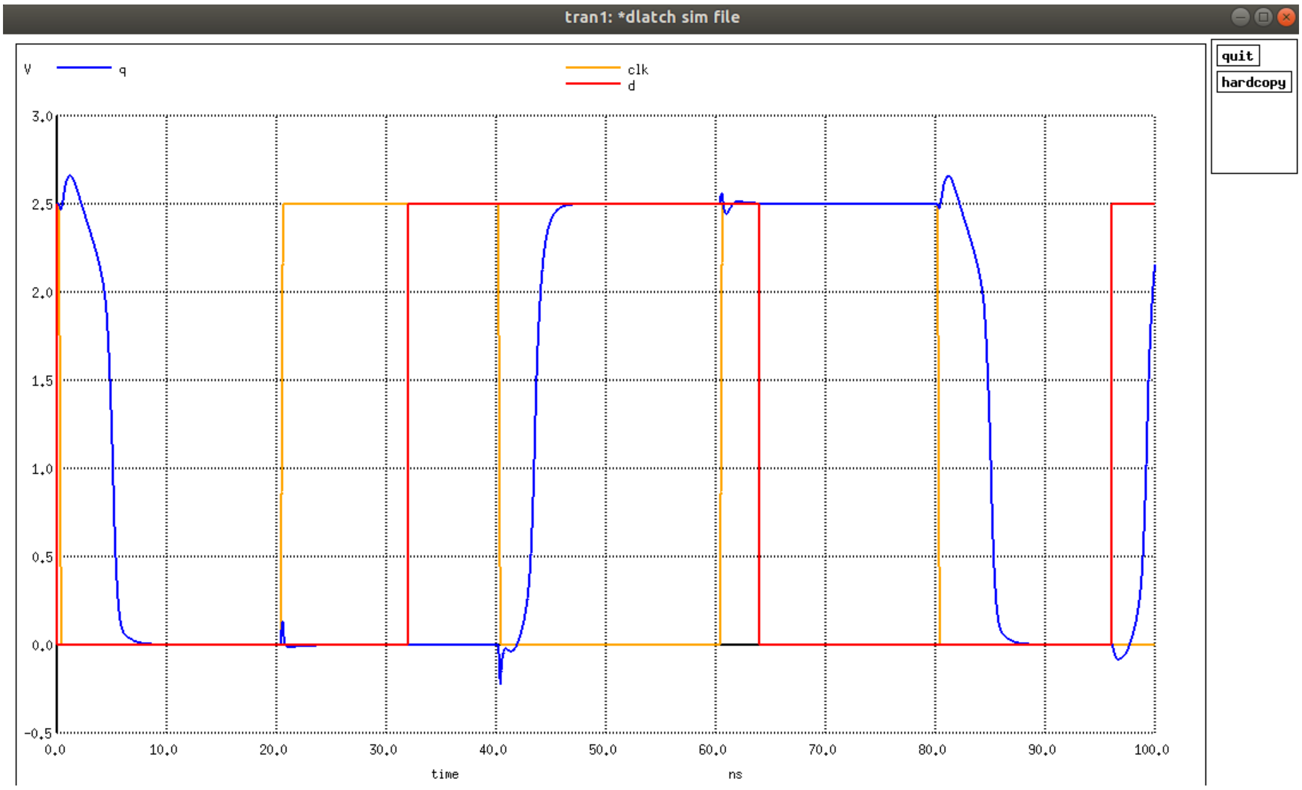
To create it in it we use the card **.subckt**, where we put the inverters, the sram cell (the inverters with the feedback) and the pass gate.

In the sizes of transistors we have chosen:

- For the inverter driven by D and the inverter driven by output Q we have W_{pmos} = 27u and W_{nmos} = 9u.
- For the feedback inverter we have W_{pmos} = 9u and W_{nmos} = 3u.
- For the pass gate we have W_{pmos} = 18u and W_{nmos} = 6u.

Then, we connect them properly to create the required circuit. Below is the truth table:

CLK	D	Q
0	0	0
1	0	Q
0	1	1
1	1	Q



The latch is active on the negative edge of the clock, so by dropping the clock to 0, we will have toggle the output Q based on D.

In the simulation image, in the first part (from 0-40ns) the input D is 0 and at the beginning Q changes and remains at 0 for the whole period of the clock.

Then, when the clock goes back to 0 we see input D again, which has gone to 1 and we change the Q from 1 to 0. This remains at 1 for the specific clock period (40ns-80ns).

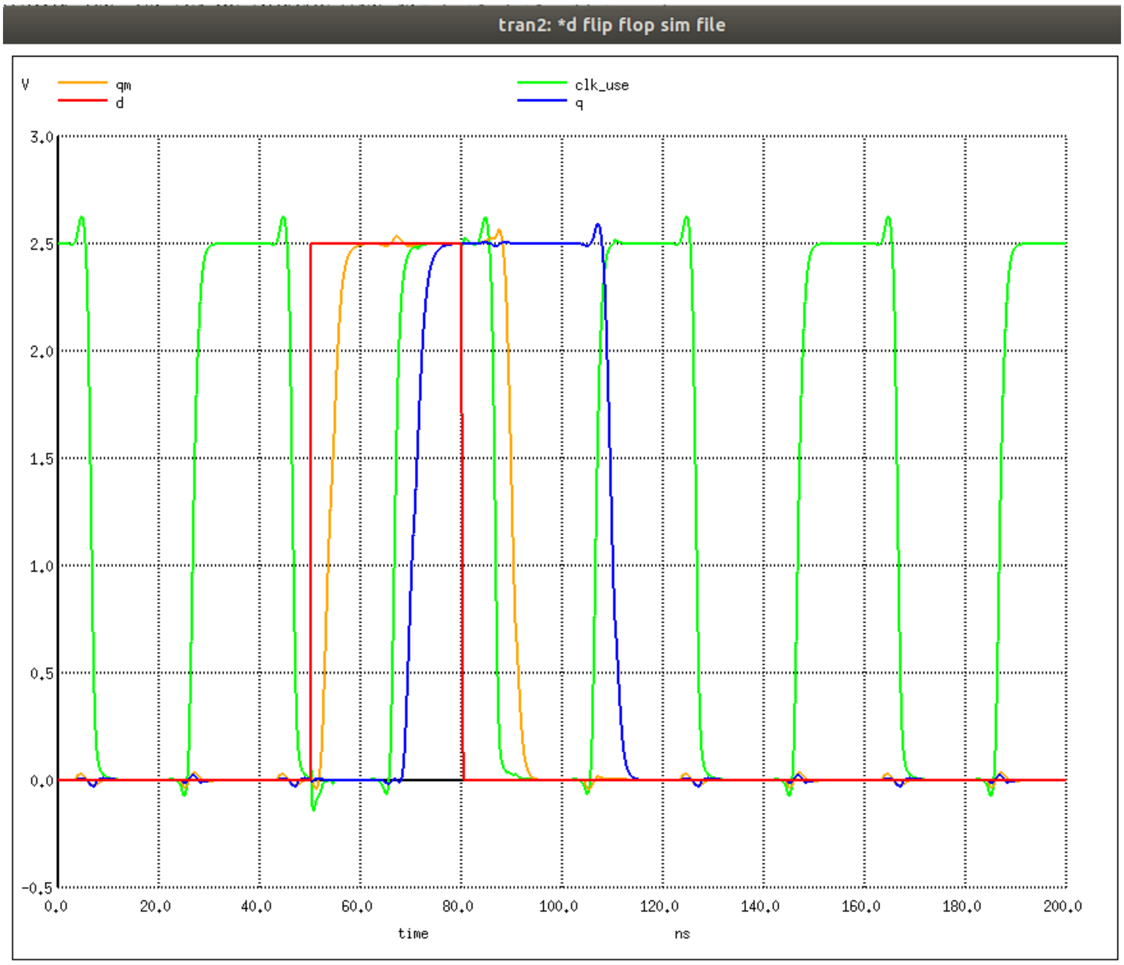
Exercise 3

In Exercise 3 we have to design and simulate a flip-flop at spice level.

First, we simulate the truth table:

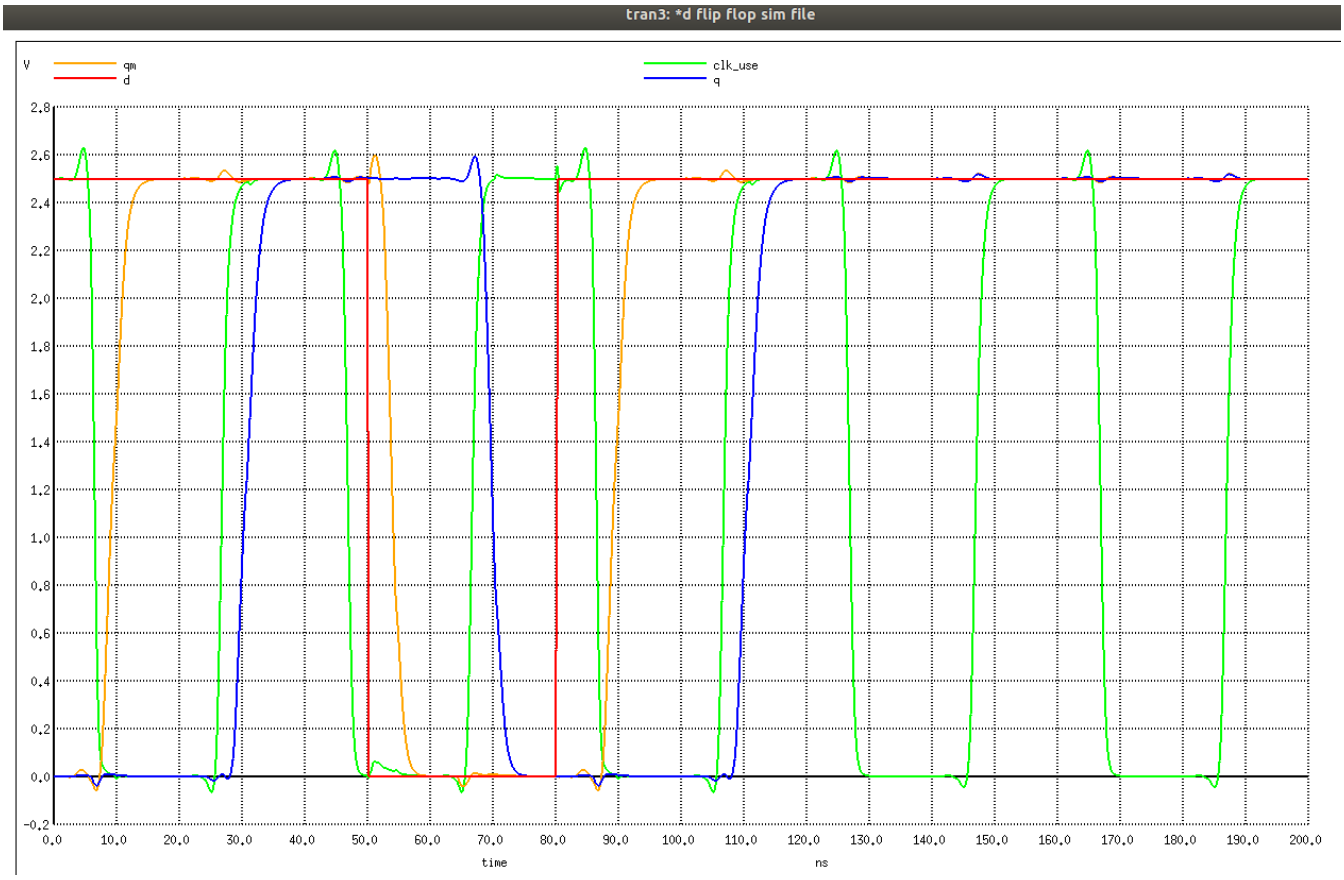
CLK	D	Q
↑	0	0
↑	1	1
else	x	Q

For the first case:



We have a pulse at input D that passes through output Q only after the clock is at the positive edge (the clock is clk_use) and holds the data until the next edge. At the next edge since D is 0 the value of Q returns to 0.

We do the same for the reverse case where we have similar results.



Now based on the first case (ie the positive pulse D) we make measurements on the times required with load Cg1 and ascent / descent time 200ps:

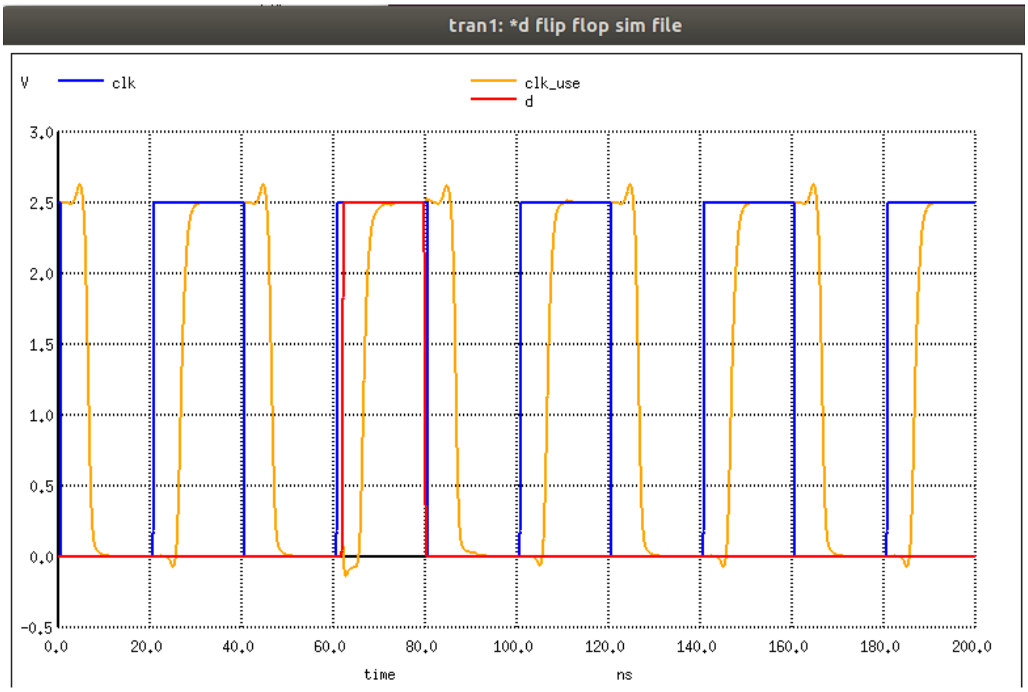
$t_{rise(qm)}$	3.67ns
$t_{fall(qm)}$	3.01ns
$t_{rise(q)}$	3.68ns
$t_{fall(q)}$	2.94ns
Clock to q	9.68ns
Hold Time	5.88ns

And for Cg1 load and rise / fall time 400ps:

$t_{rise(qm)}$	3.67ns
$t_{fall(qm)}$	3.01ns
$t_{rise(q)}$	5.09ns
$t_{fall(q)}$	3.95ns
Clock to q	10.54ns
Hold time	5.91ns

So the setup is 0 in both cases.

This can be seen from the pictures below:



Due to the long delay between the external clk clock and the clock used by the flip-flop, clk_use, the setup time is essentially negative (if we count in terms of the clk clock and not the clk_use).

In this image while the pulse D comes after the clock clk again passes to the output Q. This is not wrong since the flip-flop sees the clock clk_use which is what causes the changes in it. So the setup time in clk_use is respected and the flip-flop works properly.

So what we measure is the hold time which is the time which the data must be constant after the arrival of the clk edge. When we observe an increase of about 5% in clock_to_q then we will have the hold time, ie the time from 50 % of clk value up to 50% of clk_use.

Finally, with the increase of the descent as well as the load time we will see an increase in the time that the output Q changes, since it has to drive additional load.

It is worth mentioning that the ascent time of Qm does not change in any case since it is an internal node of Flip-Flop.