SECTION OF ELECTRICAL ENGINEERING

EPFL STI – SEL
ELG
Station nº 11
CH-1015 Lausanne

Téléphone : +4121 693 1346
Fax :
E-mail : alexandre.levisse@epfl.ch

ADVANCED VLSI DESIGN

Introductory Laboratory Manual

Dynamic Logic

Purpose

By performing some predefined simulations,

- Understanding the working principles of basic dynamic logic gates.
- Observing the main problems of the dynamic gates which are
  1. Charge Sharing
  2. Capacitive Backgate Coupling
  3. Cascading Dynamic Gates
  4. Clock Feedthrough
  5. Leakage Current
- Thinking about the reasons of these problems by looking at the simulation waveforms
- Applying the solutions to these problems and observing the results
- Learning faster alternatives of dynamic logic
- Designing a simple circuit with dynamic logic techniques

Outcomes

- Understanding in detail how dynamic logic circuits work
- Learning the most common problems of dynamic gates
- Being ready to make a robust complex design by using dynamic logic

Contents

0. STARTING THE WORK

0.1. Connect to the server

During the spring 2025 semester, the EPFL EDA infrastructure evolves the classes to use the EPFL SCITAS HPC cluster. This migration enables strong improvements in terms of computation capabilities.
For this lab, we use the helvetios cluster which is reserved for education.

/!\ Disclaimer /!\
Generally, cluster time is billed on a pay-per-use model. However, the Helvetios cluster is available for the sole purpose of education without costs for the SEL section. If for some reason during the semester you are getting access to another cluster for a project in a lab, make sure you do not mix the cluster name, and not get your lab time billed to your hosting laboratory 😉

To connect to the helvetios cluster, from the linux VM, type the following command :

```
> ssh -X username@helvetios.hpc.epfl.ch
```

Where username is your gaspar username.

To setup the EPFL EDA environment on SCITAS, run the following command :

```
>/work/fvlsi/run_edadk
```

This command shall be run for every new terminal. It does the following :

- Connect a node in the helvetios cluster
- Configure the EPFL EDA environment
- With this command your cluster reservation has the following parameters :
    - 4cores and 16GB of RAM
    - A 12h per session uptime – this means that after 12h your connection to the node cluster will automatically close. You would need to run the *>/work/fvlsi/run_edadk* command again.

0.2. The Working Environment

Open a terminal in LINUX and in your home directory (or wherever you want to work), write the below instructions to create the project directory and install the necessary files for UMC 65nm technology:

```
>cd
>cp -r /education/classes/2024-2025/EE490b/EE490b_ADVLSI .
>cd EE490b_ADVLSI
>virtuoso &
```

- If you want to copy your libraries from your edauser account from the FVLSI class, you could rsync the EDATP design library from your edauser account in selsrv1 or 2. BUT ! make sure you

only copy the EDATP library and not the entire folder. **If you are not sure to understand what was written before, ask a TA to do it for you** 😉

Inside Cadence virtuoso, create the following libraries:

VLSI2 which contains the design (simple gates, more complex blocks etc.)
VLSI2_TB which contains the schematics for performing simulations on the blocks of ***VLSI2*** library.

Attach them to an existing technology library ***umc65ll***.

0.3. The Cells and the Naming Convention

In a standard cell library, the gates should have regular names so that they can be easily found when they are searched. In this work, you will give names to the logic gates similar to the example shown below:

DYN_N_F_ND2_X1
sThis name consists of 5 different properties of a gate. The first abbreviation which is ***DYN*** stands for the fact that the gate is a dynamic gate. ***N*** indicates that the logic tree is composed of nMOS devices (i.e. the transistors which perform the logic function are nMOS). ***F*** shows that the gate is footed (i.e. in the case of an N network, a clocked nMOS transistor exists at the bottom of the gate, under the nMOS logic tree; later, it will be shown that the clocked nMOS can be removed by applying some constraints. If there are some points which are not clear, you can refer to Figure 1: NA and NB are logic tree transistors; N0: (foot) clocked nMOS under nMOS logic tree). ***ND2*** specifies that the gate performs NAND logic function with 2 inputs. Finally, ***X1*** displays the driving strength of the gate. For the different options of the 5 different properties please observe the tables below:

Logic Type	Abbreviation
Static Logic	***STA***
Dynamic Logic	***DYN***
Domino Logic	***DOM***
Data Driven Dynamic Logic (D^3L)	***D3L***

Table 1: Abbreviations of the logic types to be used in the naming of the gates

Transistor Type of the Logic Tree	Abbreviation
nMOS	***N***
pMOS	***P***
not applicable	***X***

Table 2: Letters for different logic tree types to be used in the naming of the gates

Existance of Footing	Abbreviation
footed	***F***
unfooted	***U***
Not applicable	***X***

Table 3: Letters for indicating if the clocked tail transistor exists

Logic Function	Abbreviation
AND	***AN***
OR	***OR***
Exclusive OR	***XR***
Buffer	***BU***
NAND	***ND***
NOR	***NR***
XNOR	***XN***
Inverter	***IN***
Function A (arbitrary)	***FA***
Function B (arbitrary)	***FB***
…	***…***

Table 4: Abbreviations of the logic functions to be used in the naming of the gates

- *As an example of a standard cell library, you can take a look at the UMC65LL_UMK65LSCLLMVBBR_B03PB library available in your working environment. Note that every provider has its own way to define its naming conventions. Though they usually follow a common organization.*

1. THE PROBLEMS OF DYNAMIC GATES

In this part, you will perform some simulations on simple dynamic logic gates to observe the main problems of dynamic logic. You will also apply the solutions of these problems and observe if the solutions can correct the problems.

Create a new schematic view and call it ***DYN_N_F_ND2_X1***. Draw the dynamic NAND gate shown on Figure 1. Always use ***N_12_LLRVT*** and ***P_12_LLRVT*** transistors under the library ***umc65ll***. Do **NOT** forget to connect all bulk nodes of the nMOS devices to VSS and all bulk nodes of the pMOS devices to VDD pins. Set all the transistor dimensions to the values which are indicated on the figure. Do **NOT** forget to enter the label name **X** according to Figure 1. Connect the input pins A and B exactly same as in Figure 1 although the reverse connection results in the same functionality. Create the symbol view. During the creation of the symbol view, change the property of the NLPLabel from **[@partName]** to **[@cellName]**. You should do this each time you create the symbol view of a cell. By doing this, you will be able to see the correct cell name on a symbol view in case you copy a cell. On the symbol view, also place the CLK pin to the upmost point as shown on Figure 2. This way it is separated from the other input data signals.

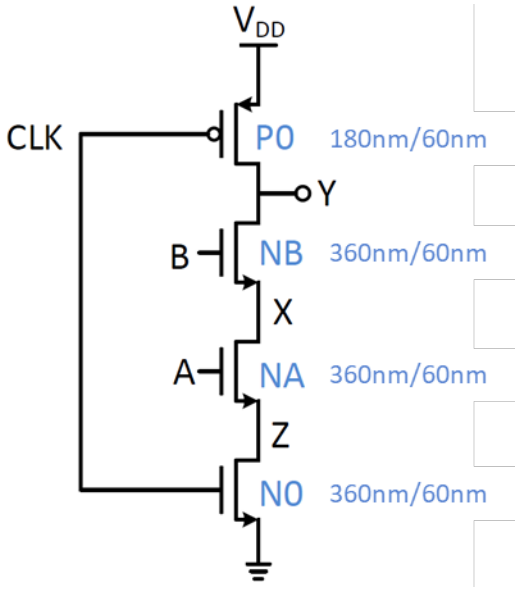


Figure 1: Dynamic NAND Gate
*(cell: **DYN_N_F_ND2_X1**, library: **VLSI2**)*

Under the library ***VLSI2_TB*** create a new schematic view and call it ***DYN_N_F_ ND2_X1_TB***. Draw the schematic of Figure 2 by providing the necessary net names. Edit the properties of the voltage sources according to Table 5.

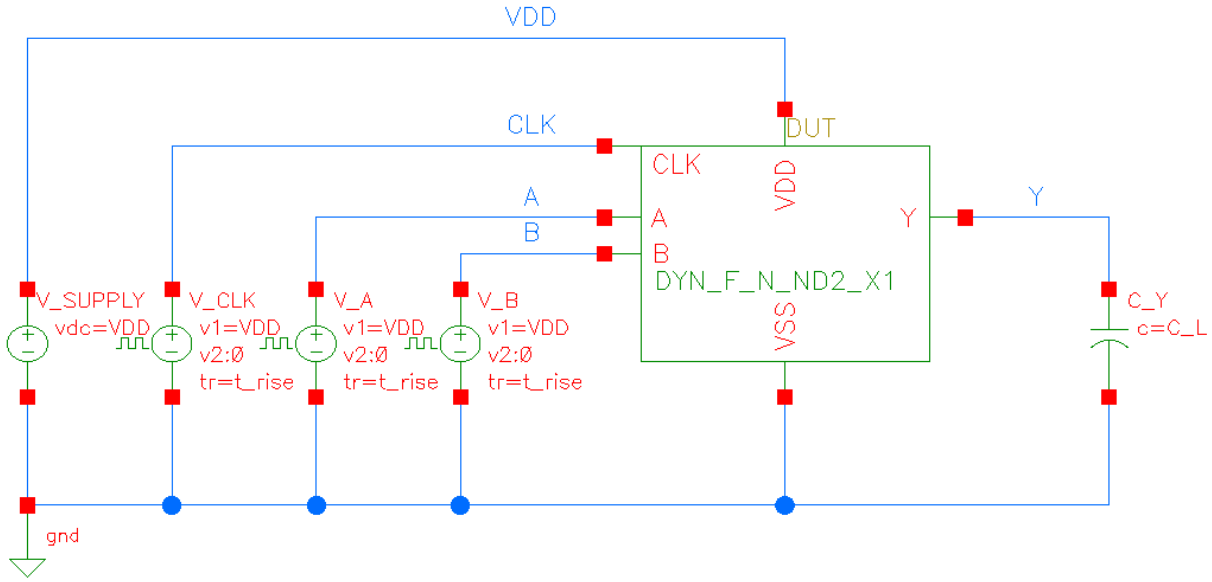


Figure 2: Dynamic NAND Gate Simulation Schematic
*(cell: **DYN_N_F_ND2_X1_TB**, library: **VLSI2_TB**)*

Properties	Instances				Units
Instance Name	V_SUPPLY	V_CLK	V_A	V_B	-
Cell Name	vdc	vpulse	vbit	vbit	-
DC voltage	VDD	-	-	-	V
Pattern Parameter Data	-	-	00010001	00000101	-
Voltage 1 / One Value	-	VDD	VDD	VDD	V
Voltage 2 / Zero Value	-	0	0	0	V
Period	-	T_CLK	T_CLK/2	T_CLK/2	s
Delay time	-	– t_rise/2	-	-	s
Rise time	-	t_rise	t_rise	t_rise	s
Fall time	-	t_fall	t_fall	t_fall	s

*Table 5: Properties of the voltage sources in the simulation schematic (cell: **DYN_N_F_ND2_X1_TB**, library: **VLSI2_TB**)*

For the capacitor at the output of the gate, define a variable **C_L** for the Capacitance entry by modifying the properties of the instance. Launch ADE L. Copy the variables from cell view and enter the values as in Table 6:

Variable	Value	Unit
C_L	0	F
T_CLK	2n	s
VDD	1.2	V
t_fall	15p	s
t_rise	15p	s

*Table 6: Values of the variables of the simulation schematic (cell: **DYN_N_F_ND2_X1_TB**, library: **VLSI2_TB**)*

Select the nets **CLK**, **A**, **B** any **Y** for plotting. By using ADE calculator, define a new output **delay_A_Y** which shows the delay time between the **2nd rising edge of signal A** and the **1st falling edge of signal Y**. Define the thresholds as VDD/2. You can use Table 7, if you do not know how to use the **delay** function of the calculator tool.

Signal1		VT("/A")	
Signal2		VT("/Y")	
Threshold Value 1	VAR("VDD")/2	Threshold Value 2	VAR("VDD")/2
Edge Number 1	2	Edge Number 2	1
Edge Type 1	rising	Edge Type 2	falling

*Table 7: Properties of the delay function (output: **delay_A_Y**)*

Select transient simulation and set the **Stop Time** as **4*VAR("T_CLK")**. Run the simulation. Check the functionality of the NAND gate. For checking the functionality, you should observe the voltage levels at the output during the CLK signal is high (which corresponds to the evaluation phase). Does the NAND gate perform the true functionality? What is the voltage value of the net Y at t = 5.5ns?

Redo the simulation by changing the C_L variable from 0 to 1f. Observe the voltage value of the net Y at time = 5.5ns once again. Note the delay value for **C_L** = 1f:

delay_A_Y: ..... [ps]	Dynamic NAND2

Try to understand the reason of this problem by performing the following steps:

- For the ***DYN_N_F_ND2_X1*** cell, plot the 5 transient voltages: inputs **A**, **B**, **CLK**; output **Y** and the net **X**.
- Zoom to t=5ns.
- Observe the transitions by looking at the dynamic NAND gate schematic on Figure 1 and considering the MOS parasitic capacitances seen at the nets **Y** and **X**, and considering the states of the transistors (either they are ON/OFF) (Do not forget the fact that the net Y at the output is a precharged floating node).

1.1. Charge Sharing

According to the working principle of a dynamic gate, the output node is charged during the precharge phase. During the evaluate phase, the pull-up device is OFF. By observing the schematic on Figure 3 and the corresponding signal diagram on Figure 4, it can be seen that starting with time t_0, the output node behaves as a precharged floating capacitor. If the output node is shorted at some time (t_1) to another capacitive node (X), the charge at the output node (Y) is shared between the X and Y nodes. This might result in wrong evaluation of the data as it propagates in the circuit. The solution is to charge all the floating nodes in the network as shown in Figure 5 by adding necessary pull-up devices. However, the additional device increases the parasitic capacitance of the gate and the need to discharge the X node will increase delay time. Note that for the Z node no precharge device is needed since it will be automatically shorted to the ground node during the evaluation phase.

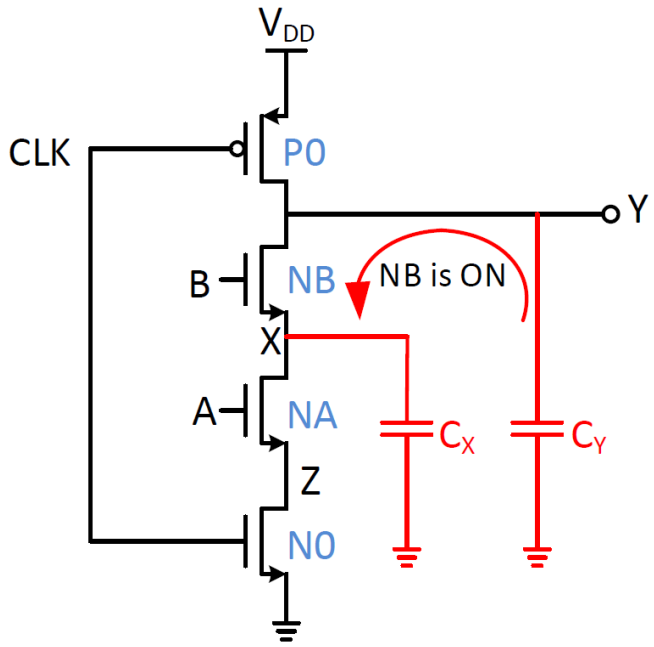


*Figure 3: Illustration of the **charge sharing** problem*
*(cell: **DYN_N_F_ND2_X1**, library: **VLSI2**)*

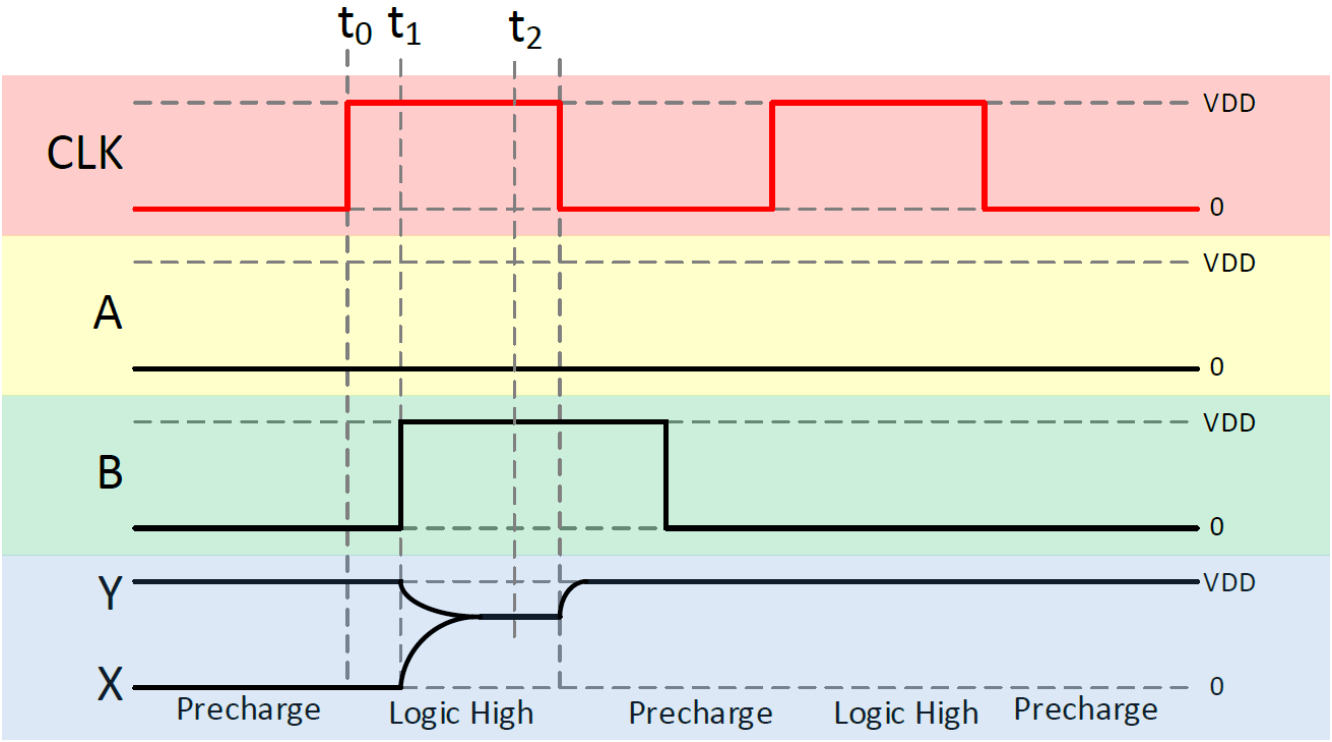


Figure 4: Signal flow diagram for the dynamic NAND gate of Figure 1 and 3
*(cell: **DYN_N_F_ND2_X1_TB**, library: **VLSI2_TB**)*

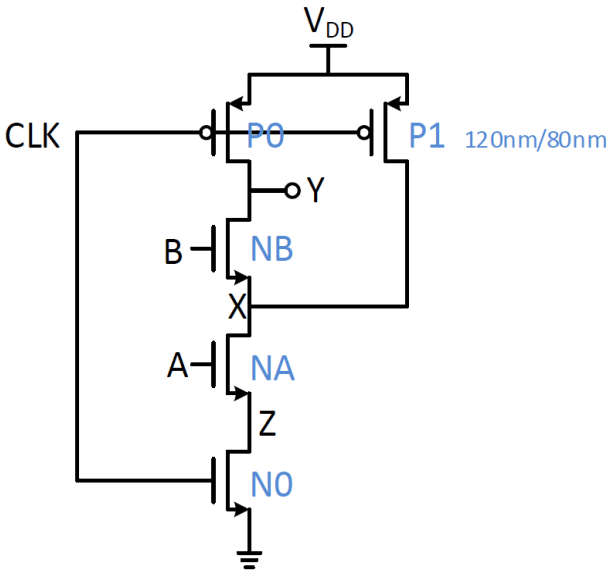


Figure 5: Dynamic NAND Gate with pull-up devices at all nodes
*(cell: **DYN_N_F_ND2_X1**, library: **VLSI2**)*

Case Study 1: Two different simulations were run on the circuit of Figure 2. The below data is observed:
Simulation 1: C_L = 1fF, $V_Y(t_2)$ = 1.058mV at steady state
Simulation 2: C_L = 2fF, $V_Y(t_2)$ = 1.118mV at steady state
Estimate the values of the parasitic capacitors C_X and C_Y which are on Figure 3. Note that C_X and C_Y are the intrinsic capacitances of the dynamic NAND gate and capacitor C_L is external parameter.

Open the schematic view of the NAND gate and insert a pull-up device for the X node as shown on Figure 5. Redo the simulation with **C_L** = 1f. Do you observe the true functionality now? What can be the reason of the increase on the delay time?

delay_A_Y: ..... [ps]	Dynamic NAND2 with pull-up device

Do **NOT** forget to save the state of the simulation before closing ADE L window.

Now we would like to make an AND gate by cascading a dynamic inverter to the NAND gate. Create a schematic view and call it ***DYN_N_F_IN1_X1***. The inverter should look similar to the one on Figure 6. You can simply copy the cell ***DYN_N_F_ND2_X1*** and modify it to save some time; if you do so, do **NOT** forget to modify also the symbol view.

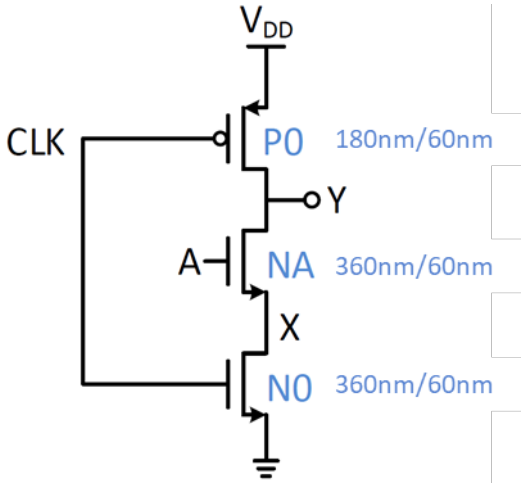


Figure 6: Dynamic Inverter Gate
*(cell: **DYN_N_F_IN1_X1**, library: **VLSI2**)*

Create a new cell and call it ***DYN_N_F_AN2_X1***. Cascade the NAND gate and the inverter to make an AND gate as in Figure 7. Do **NOT** forget to create also the symbol view.

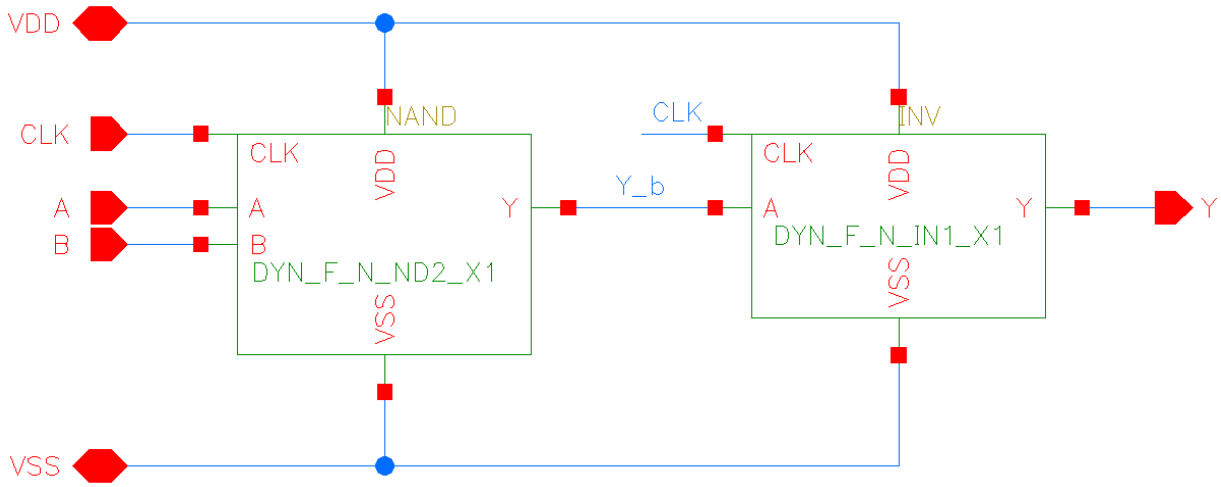


Figure 7: Dynamic AND Gate Schematic View

*(cell: **DYN_N_F_AN2_X1**, library: **VLSI2**)*

To simulate this gate you can simply make a copy from the cell ***DYN_N_F_ND2_X1_TB***. Call the new cell ***DYN_N_F_AN2_X1_TB***. Perform the same transient simulation of the NAND gate, this time for the AND gate, with exactly the same values of the variables and the same configuration (adjust **C_L** to 1fF). Select the nets **CLK**, **A**, **B** and also the net **Y_b** inside the schematic view of ***DYN_N_F_AN2_X1*** (which corresponds to the output of the dynamic NAND gate). Do **NOT** plot the net **Y** for now. Run the simulation.

Does it look like the output of a correctly functioning NAND gate?

Try to understand the reason of this problem by performing the following steps:

- On the schematic view of the ***DYN_N_F_IN1_X1*** cell, select the three transient voltages input **A**, output **Y** and the net **X** and plot them on the same graph.
- Zoom to t=1ns
- Observe the transitions by looking at the inverter schematic on Figure 6 (***DYN_N_F_IN1_X1*** on Cadence) and considering the MOS parasitic capacitances (Do not forget the fact that the input of the inverter is a precharged floating node during the evaluation phase).

1.2. Capacitive Backgate Coupling

In some cases, a voltage change at the output of a gate might change the voltage levels at some of its inputs. The 2 main reasons of this problem are

- The parasitic capacitances between the output node and the input nodes
- The fact that the input nodes are precharged to V_{DD} and not driven by any gate during the evaluation phase (i.e. they are floating).

Therefore, these precharged nodes are quite prone to be affected by the voltage variations of some nodes which are capacitively coupled. Figures 8 and 9 illustrate the situation. On Figure 8, a dynamic NAND gate is driving a static NAND gate and Figure 9 shows the signal levels at some nodes of the gates which are shown on Figure 8. Just after the evaluation phase starts, the node Y_1 is disconnected from any node and it becomes floating. At some point, the B input of the static NAND starts to rise which results in discharging the nodes X and Y_2 between the time intervals $[t_0, t_1]$. Since these nodes are coupled to the floating node Y_1, the dynamic node voltage is sagged depending on the capacitance ratios. If this variation is extreme and Y_1 falls down more than the critical point that N1 is turned OFF and P1 is turned ON, the output Y_2 might rise more than a critical value. This results in the propagation of the wrong data and the design fails to work correctly. To prevent this problem, the simplest solution is to insert a static weak pull-up device to the output node Y_1 (to the Y output for the ***DYN_N_F_ND2_X1*** cell) according to Figure 10. This solution works well but when the Y_1 is pulled down, there will be static power consumption. The capacitive backgate coupling might also result from closely coupled routings on the layout; therefore, during the design of the layout the critical nodes (usually the floating nodes during evaluation) should be carefully considered.

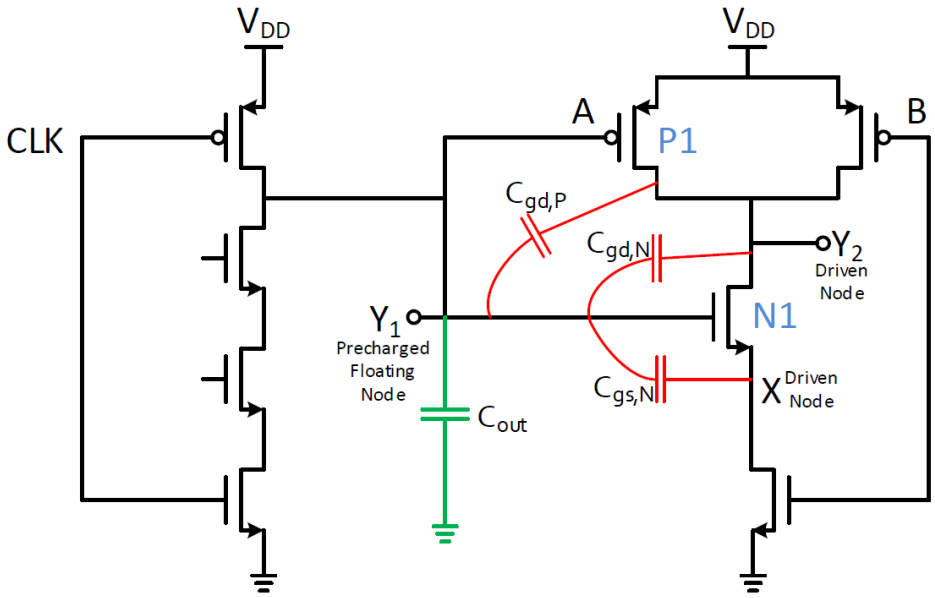


*Figure 8: Illustration of the **capacitive backgate** problem*

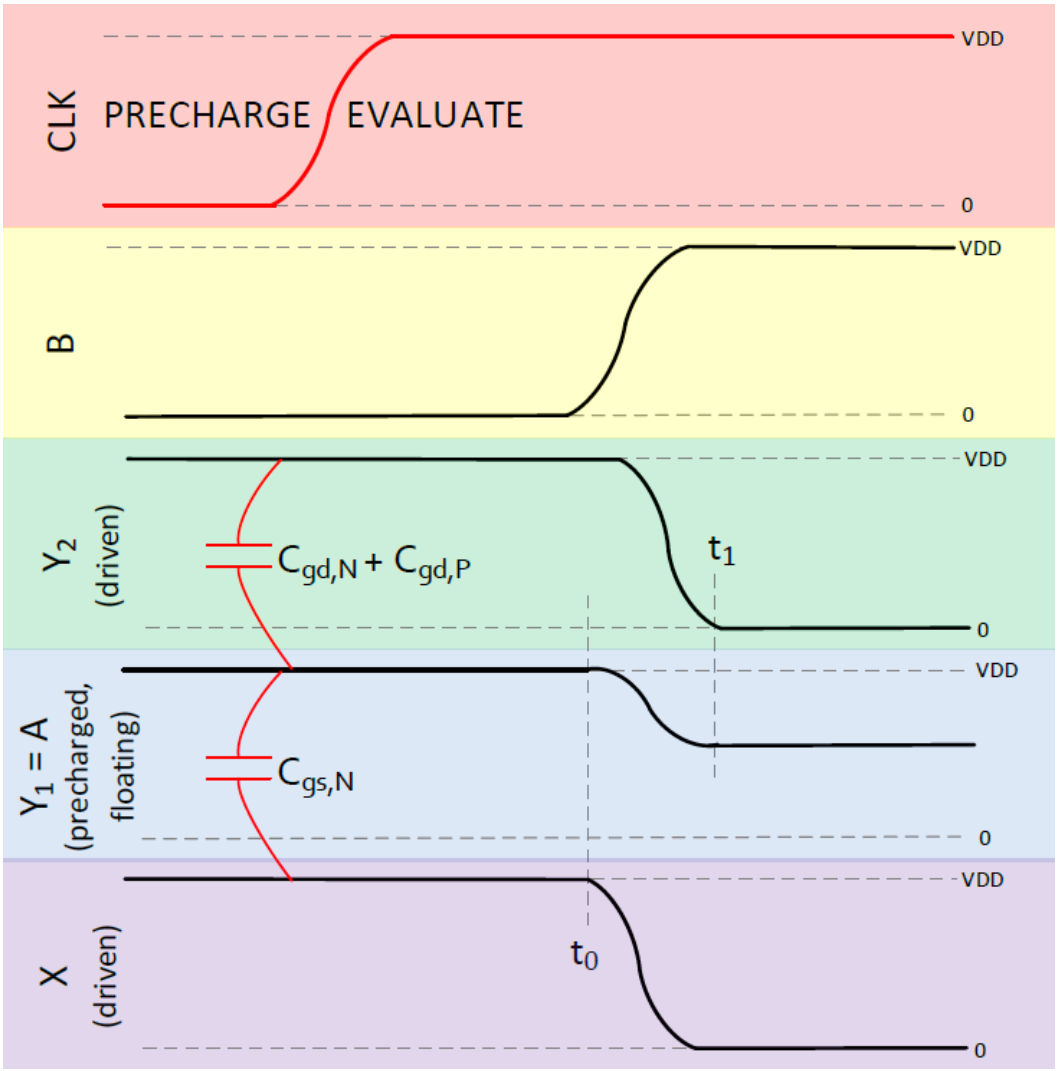


Figure 9: Signal flow diagram for the 2nd N-network dynamic gate of Figure 8

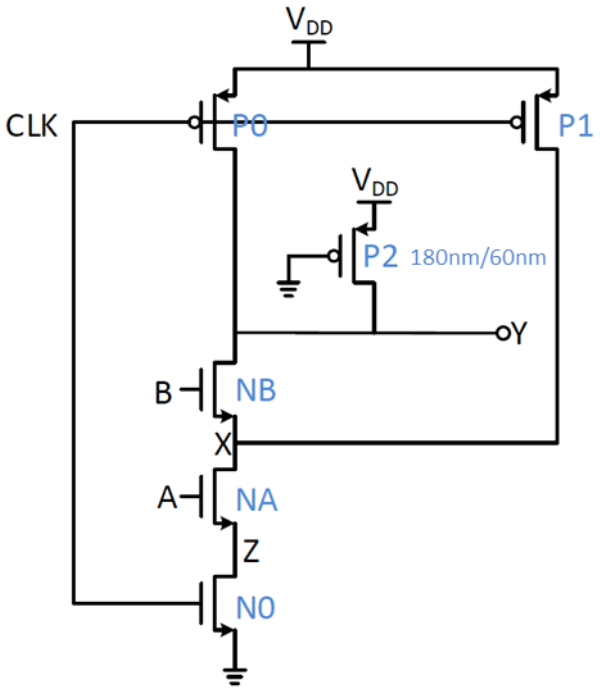


*Figure 10: A simple solution for the **capacitive backgate problem** (cell: **DYN_N_F_ND2_X1**, library: **VLSI2**)*

Case Study 2: According to the Figures 8 and 9 it is given that
$V_{Y1}(t < t_0) = V_{Y2}(t < t_0) = V_X(t < t_0) = 1.2V$
$V_{Y2}(t_1 < t) = V_X(t_1 < t) = 0V$ and $V_{Y1}(t_1 < t) = 0.84V$
On Figure 8, N1 is on and P1 is off for all the times.
Ignore rest of the parasitic capacitances other than the ones shown on Figure 8.
$C_c = C_{gd,P} + C_{gd,N} + C_{gs,N}$
Estimate the ratio C_{out}/C_c.

Modify the schematic view of the ***DYN_N_F_ND2_X1*** cell according to Figure 10 and redo the simulation. Do you observe the true logical output values on **Y_b** net?

Now, remove the pull-up transistor P2 because this is not the best solution since it brings some static power consumption. After removing the pull-up device redo the simulation and this time plot also the **Y** net. Check the functionality of the AND gate **only for t=7.5ns**. Does it give the right output? If not, for this time instant, the problem cannot be the capacitive backgate coupling because the net **Y_b** is logically correct and the net **Y** is simply the inverse of the net **Y_b**.

Try to understand the reason of this problem by performing the following steps:

- On the schematic view of the ***DYN_N_F_IN1_X1*** cell, select the three transient voltages for the inputs **A** and **CLK** and the output **Y**. Plot them on the same graph.
- Zoom to t=7ns.
- Observe the transitions by looking at the inverter schematic on Figure 6 (***DYN_N_F_IN1_X1*** on Cadence) and considering the states of the three transistors on that schematic (either they are ON/OFF).

1.3. Cascading Dynamic Gates

It is impossible to connect the output of a dynamic gate with an nMOS logic tree to the output of another dynamic gate which also has nMOS logic tree (i.e. nMOS inputs). The reason can be clarified by observing the Figures 11 and 12: Figure 11 shows a dynamic gate with nMOS logic tree which drives the same type of dynamic gate (forbidden case) and Figure 12 shows the signal flow. On Figure 11, the outputs Y_1 and Y_2 are both precharged to high voltage value. Therefore, just after the CLK signal goes high, the output node Y_2 will discharge since all of the inputs of the Gate 2 are precharged to high (Figure 12). Consequently, if some of the inputs of Gate 2 changes after some time, the output Y_2 can never be charged again because there is no pull-up path during the evaluation phase. The situation can be fixed by cascading a static inverter for **ALL** of the dynamic gates in a design to provide logical low nodes for the next stage nMOS logic tree gates as in Figure 13. These types of gates are called **domino logic gates** (including the static inverter).

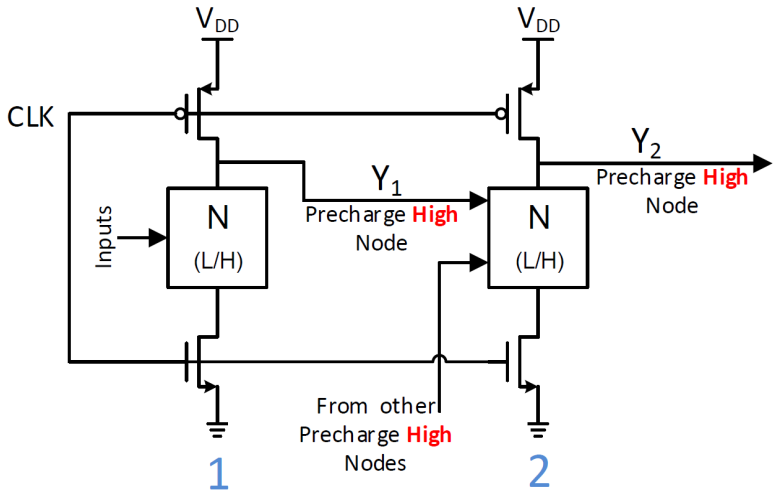


*Figure 11: Illustration of the **cascading dynamic gates** problem*

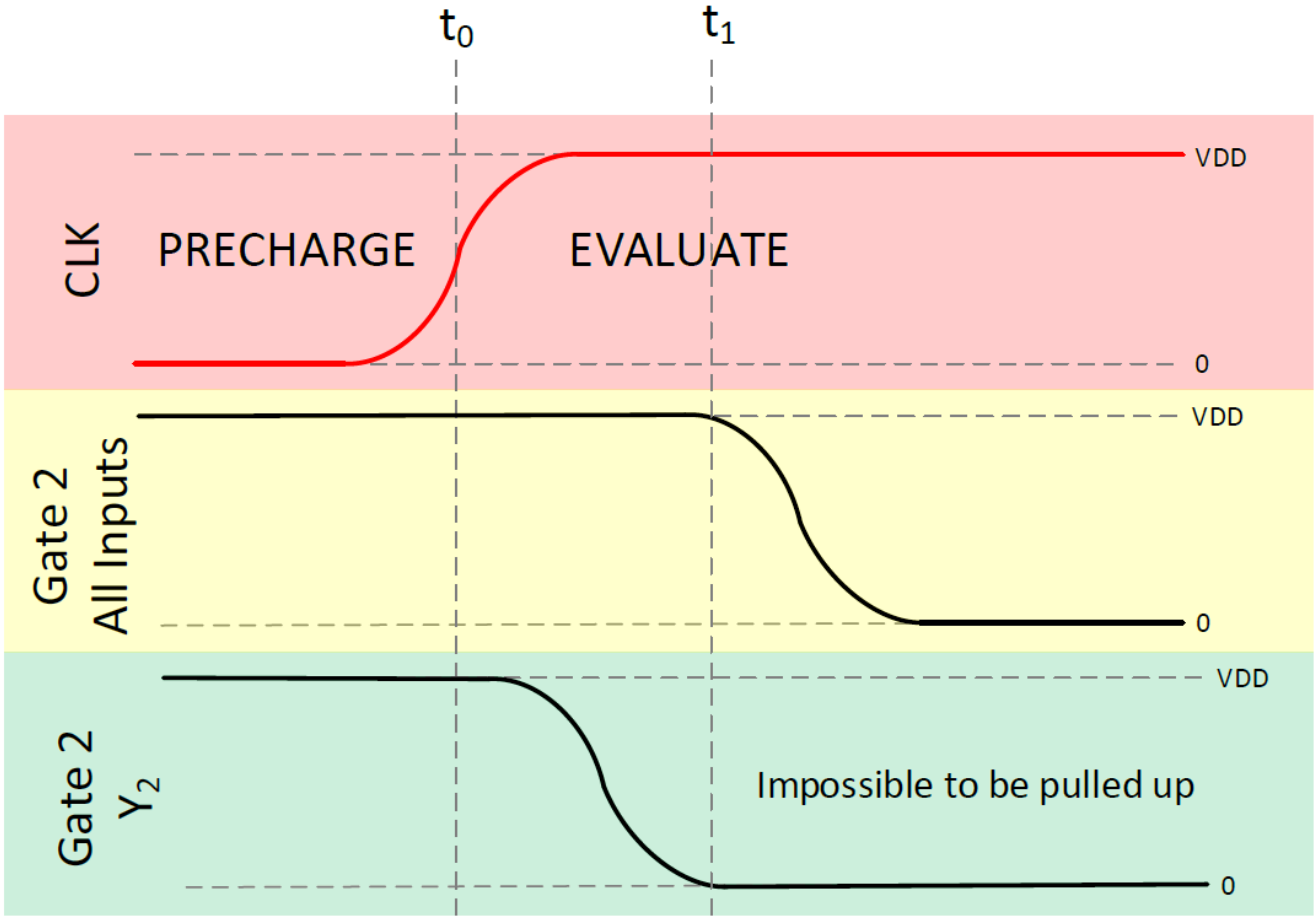


*Figure 12: Signal flow diagram for the 2nd N logic tree dynamic gate of Figure 11 (cell: **DYN_N_F_AN2_X1_TB**, library: **VLSI2_TB**)*

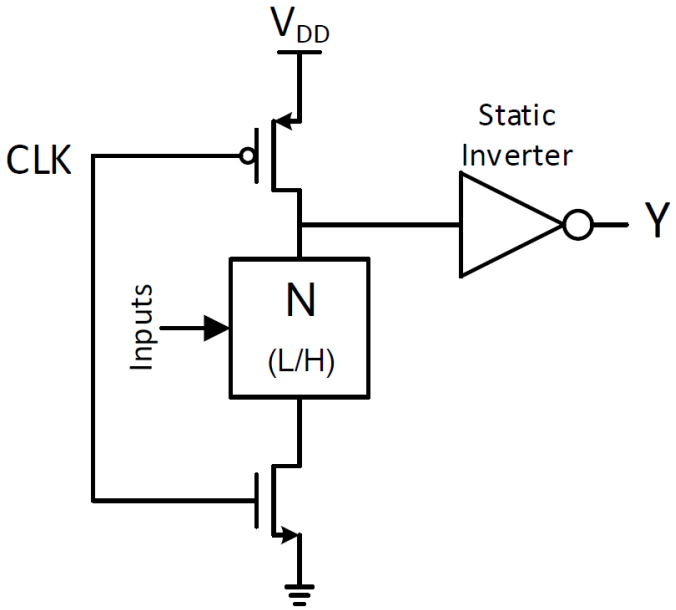


Figure 13: Domino Logic Gate

Create a new schematic view for the static inverter, call it ***STA_X_X_IN1_X2*** and size it according to Figure 14. After that, create a new schematic view and call it ***DOM_N_F_AN2_X1*** under the library ***VLSI2*** and the corresponding simulation schematic ***DOM_N_F_AN2_X1_TB*** under the library ***VLSI2_TB***. You can simply copy the cells ***DYN_N_F_AN2_X1*** and ***DYN_N_F_AN2_X1_TB*** and make necessary modifications. At the end, ***DOM_N_F_AN2_X1*** cell should be the dynamic NAND gate (***DYN_N_F_ND2_X1***) cascaded by the static inverter (***STA_X_X_IN1_X2***). Please do **NOT** forget that the dynamic AND gate (***DYN_N_F_AN2_X1***) and the corresponding simulation schematic cell (***DYN_N_F_AN2_X1_TB***) are useless; but do **NOT** delete them for remembering the related problems in the future.

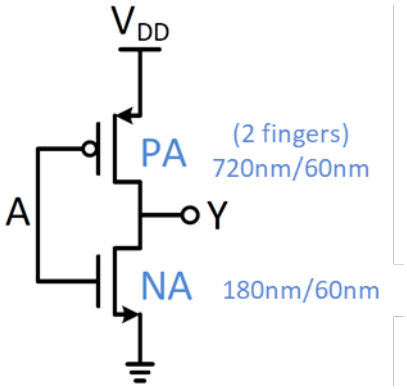


Figure 14: Static inverter
*(cell: **STA_X_X_IN1_X2**, library: **VLSI2**)*

Case Study 3: Normally, for a static inverter, the pMOS aspect ratio is adjusted so that it is around 2.5 times the aspect ratio of the nMOS due to mobility issues. However, on Figure 1, the pMOS device is 4 times larger than the nMOS device. What can be the reason of this? (Hint: Think about the possible transitions that can occur at the output of the inverter when it is cascaded to the output of an N logical tree dynamic gate.)

Open the simulation schematic ***DOM_N_F_AN2_X1_TB*** and load the same simulation state of ***DYN_N_F_AN2_X1_TB***. Perform the simulation and also measure the delay between the second rising edge of A and the first rising edge of Y. Plot both the transient voltages of the nets **Y_b** and **Y**. Do you observe the correct logical values?

delay_A_Y: ..... [ps]	Dynamic AND2

Measure the voltage level for the net **Y_b** at t = 1.2ns. Does it show some variations when compared to the logical high level (V_{DD})?

Try to understand the reason of this problem by performing the following steps:

- On the schematic view of the ***DOM_N_F_ND2_X1*** cell, plot the three transient voltages for the inputs **A**, **B**, **CLK** and the output **Y_b**.
- Zoom to t = 1ns.
- By considering the MOS parasitic capacitances seen between the output nodes and the input nodes observe the transitions of the signals.

1.4. Clock Feedthrough

The main reason of this problem is the fact that the output node has no pull-down path at the low to high clock transition (i.e. the transition from precharge phase to evaluation phase). The Figures 15 and 16 would be helpful to observe the situation. By the time the clock signal makes a low to high transition at t_o (Figure 16), it affects the voltage at the output node due to the coupling capacitance ($C_{gd,P0}$). This results in a stationary increase at the precharged output node. When one of the input signals change in a way that the output falls at t_1, it should fall down with a higher swing (ΔV_{out}), which increases the delay of the gate.

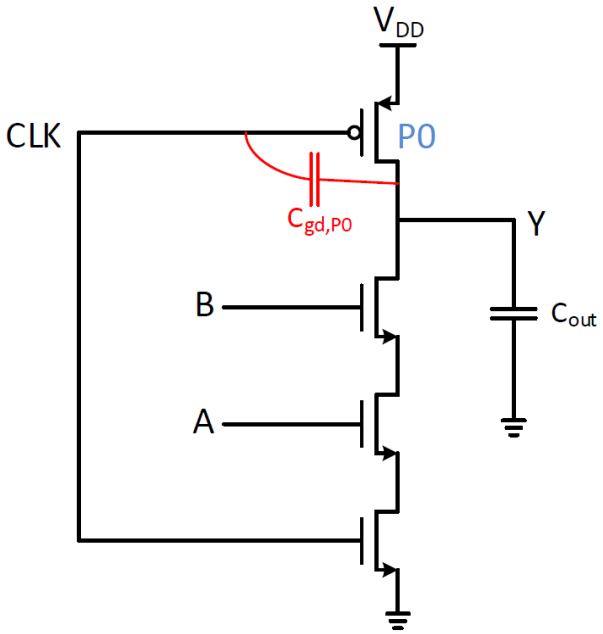


Figure 15: Clock feedthrough problem
*(cell: **DYN_N_F_AD2_X1_TB**, library: **VLSI2_TB**)*

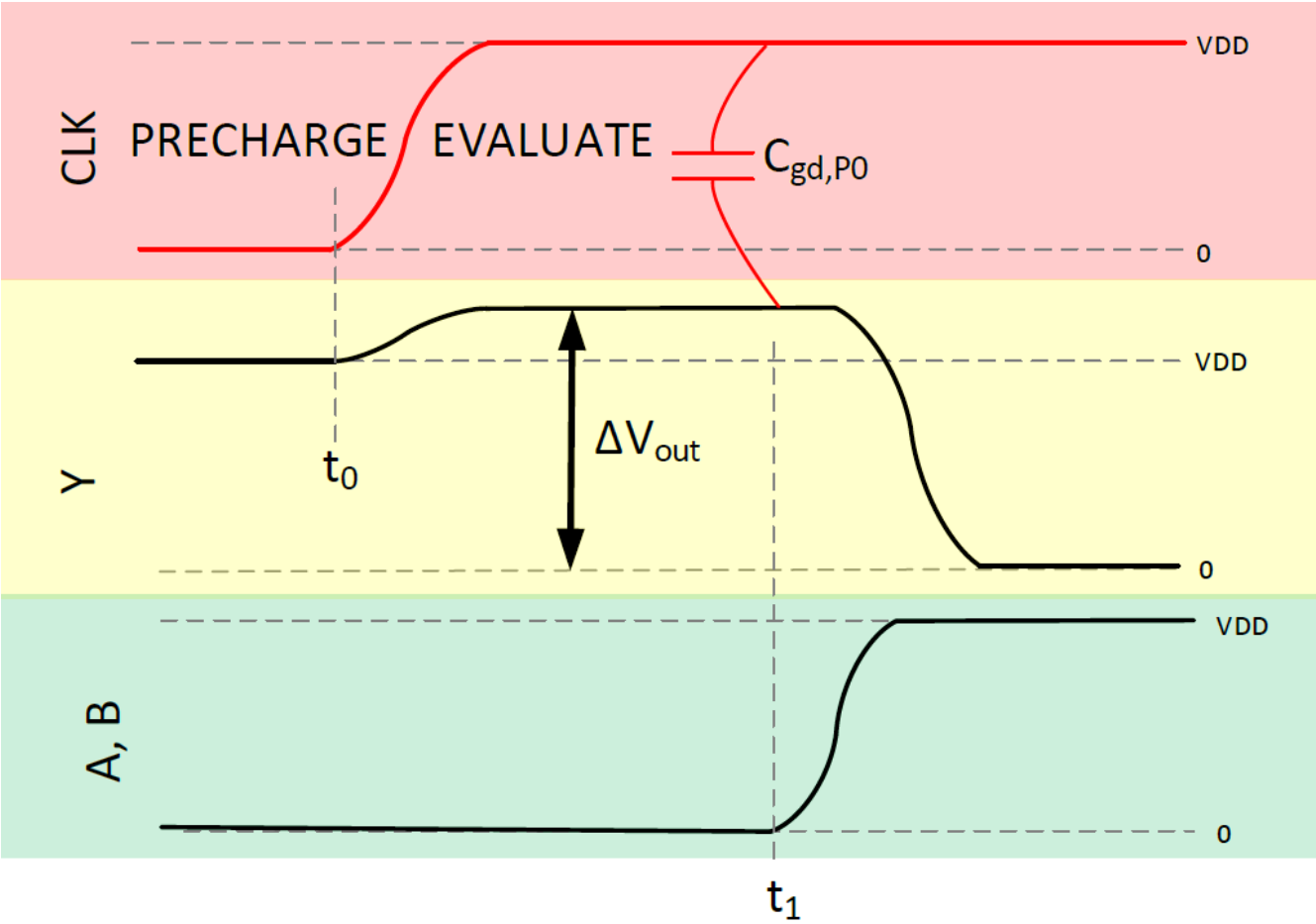


*Figure 16: Signal flow diagram corresponding to the dynamic gate of Figure 15 (cell: **DYN_N_F_AD2_X1_TB**, library: **VLSI2_TB**)*

Case Study 4: On Figure 15 and Figure 16 it is given that C_{out} = 500 aF, $V_Y(t_1)$ = 1.4V, V_{DD} = 1.2V Can you estimate the value of $C_{gd,P0}$?

Change the variable **T_CLK** to 2µs and redo the same simulation. Observe the transition voltage of net **Y** for t = 3.5ns and t = 5.5ns. Does it give the right output value? If you don't see a difference, try with LL_LVT transistors in place of the LL_RVT ones, to exemplify the issue. After this test, put back LL_RVT MOS. You could also try to increase the temperature, it will make the leakage higher.

- Note that in a real design, one would need to make sure that the circuit works in all the corners (temperature, VT variations, voltage variations). Thereby in reality, it is highly likely that a circuit working in TT corner, at nominal vdd and 25C will not work in the corners, so the design needs to be adapted accordingly.

1.5. Leakage Current

For very low clock frequencies, the charge at the output node might leak through the nMOS transistors. This problem can be solved by inserting a keeper transistor at the output node as in Figure 17. This application also solves the problems 2 and 4 because the output node is driven by the weak keeper and it is not floating any more. Also the pull-up devices for the other floating nodes can be removed (Remember problem 1).

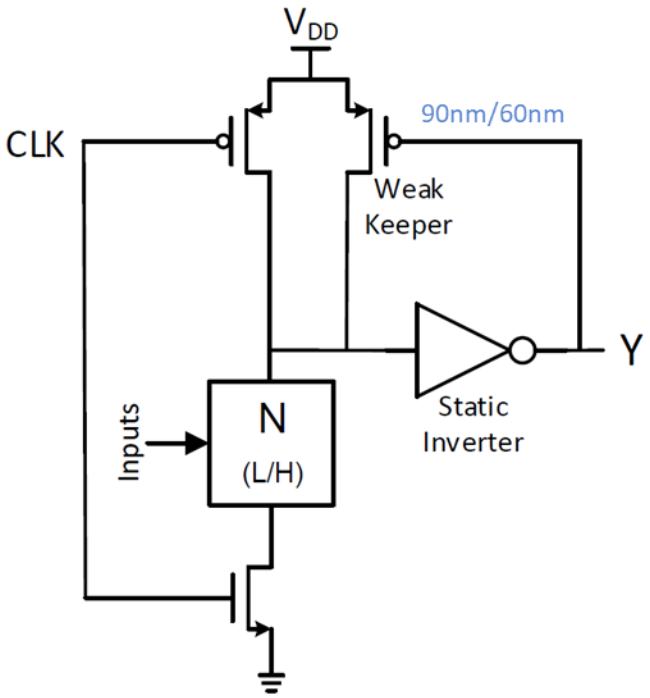


Figure 17: Driving the dynamic output node of a domino gate by a weak keeper.

Modify the domino AND gate (***DOM_N_F_AD2_X1***) by inserting a weak keeper according to Figure 17 and redo the same simulation. Try to observe the problems 2, 4 and 5. This is the most robust form of a dynamic logic gate.

delay_A_Y: ..... [ps]	Dynamic AND2 with weak keeper

2. FASTER DYNAMIC LOGIC SOLUTIONS

In this part you will learn some techniques to increase the driving capability of the dynamic gates and decrease the logic depth of a circuit which is implemented by dynamic logic gates.

2.1. NP Domino Logic (Footed)

For an N-type dynamic logic gate, all of the inputs should be precharged to zero before the evaluation phase starts. Naturally, the output node is precharged to one (Figure 18-a). Due to this fact, N-type gates are often referred as **input precharged low, output precharged high (L/H)** gates. Therefore, it is impossible to drive an N-type dynamic gate by another N-type dynamic gate because of these incompatibility issues at the level of precharging at the inputs and the outputs.

For a P-type dynamic logic gate on the other hand, all of its inputs should be precharged to one and the output is precharged to zero. A P-type dynamic logic gate can be seen on Figure 18-b. Analogous to the N-type gates, the P-type gates are often referred as **input precharged high, output precharged low (H/L)** gates. It is also impossible to drive a P-type dynamic gate by another P-type dynamic gate. Nevertheless, N-type gates can drive P-type gates and vice versa. Thus, instead of placing an inverter at the output of a dynamic gate as in Domino Logic, the logic can be constructed by placing N-type and P-type gates consecutively as in Figure 19. This type of dynamic implementation is called **NP Domino Logic** or **NORA (NO RAce) Logic** (There is also a third name which is **Zipper Logic** with a slight difference).

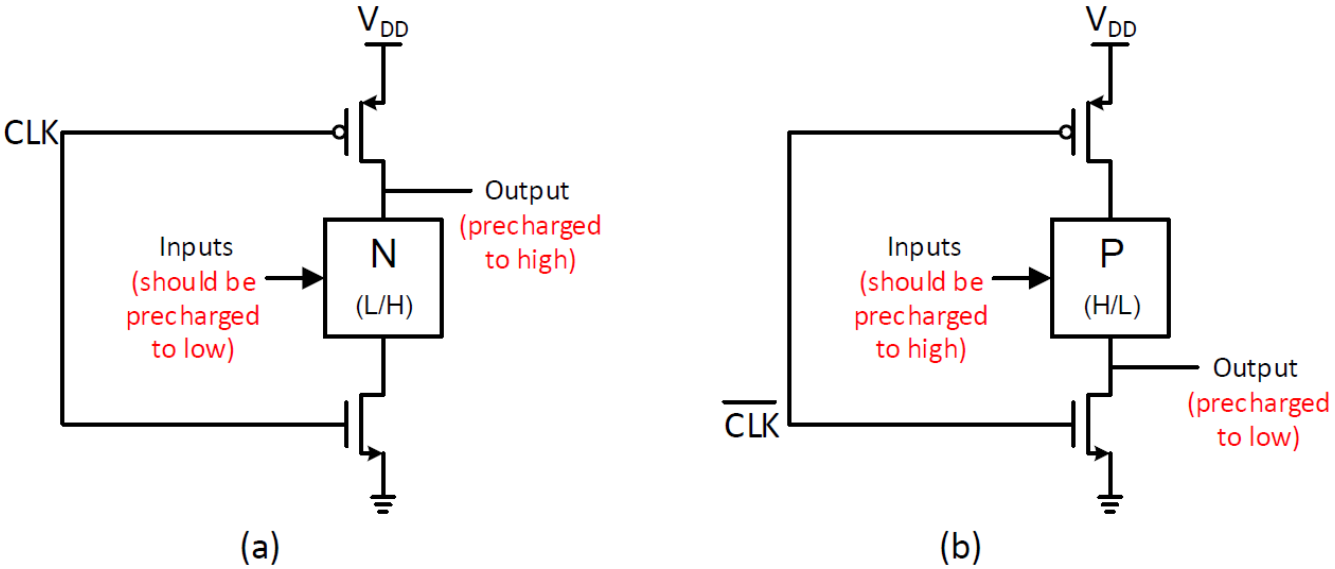


Figure 18: (a) N-type dynamic logic gate, (b) P-type dynamic logic gate

The NP Domino Logic can decrease the logic depth and area significantly because there is no need for

the inverters at the outputs.

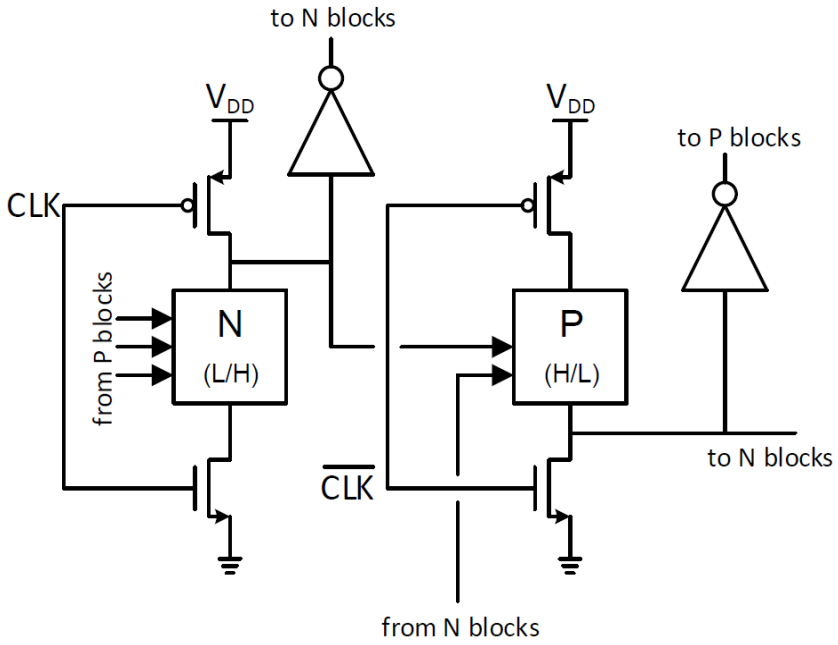


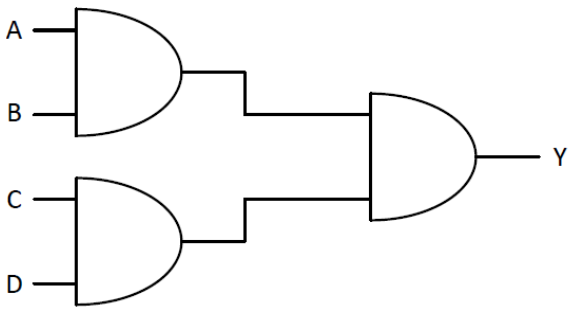
Figure 19: NP Domino Logic (footed)

The difference between Domino Logic and NP Domino Logic can be easily seen with a simple example about an AND function with 4 inputs. By using Domino Logic, the AND4 gate can be implemented as in Figure 20-a. It can be seen that there are three AND2 gates. To implement the AND2 gate, physically we need a NAND2 gate and an inverter. Thus, the logic depth is already 2 in a single gate. The reason is that with Domino Logic, only **non-inverting gates** (where the input and the output cannot rise or fall at the same time for a gate) can be implemented and the depth of a non-inverting gate is by default 2.

By using NP Domino Logic on the other hand, the AND4 function can be implemented with 2 NAND2 and a NOR2 gate as in Figure 20-b. The NAND and the NOR are **inverting gates**. Their logic depths are 1 and by using them the AND4 gate can be implemented with a logic depth of 2. Both the area and the speed can be optimized to a better level than the Dynamic Logic counterpart of the AND4 gate.

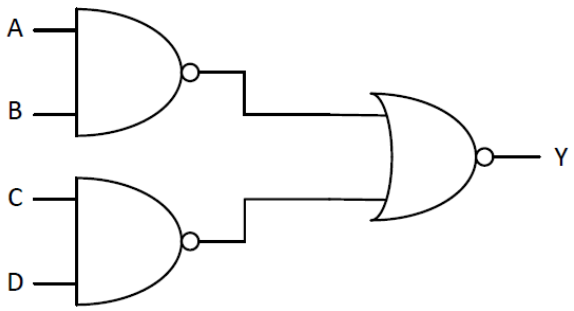
Note that using De Morgan's Law <u>in the reverse fashion</u> is quite helpful for the optimization when NP Domino Logic is used:

$Y = (A \bullet B) \bullet (C \bullet D)$



(a)

$Y = \overline{\overline{(A \bullet B)} + \overline{(C \bullet D)}}$



(b)

Figure 20: AND4 implementation with (a) Domino Logic, (b) NP Domino Logic

Create a new schematic and call it ***DYN_NP_F_AN4_X1***. This cell will be the footed NP Domino Logic Implementation of the AND4 function (Figure 20-b). For the NAND2 gates use ***DYN_N_F_ND2_X1*** and for the 2 input NOR function, create a new cell which is called ***DYN_P_F_NR2_X1***. From this point, do NOT forget to apply the solution of the charge sharing problem. Note that you will implement a P-type NOR2 gate (for the second stage). For the transistor sizes, use the same sizes which are indicated on Figure 1 and Figure 5 (do **NOT** forget that you are implementing a P-type gate, so use 500nm/80nm pMOS devices for the logic tree and small nMOS devices). For the cell ***DYN_NP_F_AN4_X1***; ***NP*** means that the first stage is N-type and the second stage is P-type.

Create also the simulation schematic called ***DYN_NP_F_AN4_X1_TB*** under the library ***VLSI2_TB***. Use the same simulation configuration which was shown on the Table 5 and 6 except the *Pattern Parameter Data* entries of the bit sources (You can simply load the saved states from other cells). Adjust the bit sources according to Table 8:

Instance (vbit)	Pattern Parameter Data
V_A	00010001000100010001000100010001
V_B	00000101000001010000010100000101
V_C	00000000010101010000000001010101
V_D	00000000000000000101010101010101

Table 8: Bit Source Configuration
*(cell: **DYN_NP_F_AD4_X1_TB**, library: **VLSI2_TB**)*

Adjust also the **Stop Time** of the transient simulation as **16*VAR("T_CLK")** to see all the possible input vectors. Check the functionality and measure the delay time between the 8th rising edge of the input A and the following rising edge of the output Y.

delay_A_Y: ..... [ps]	NP Domino AND4 (footed)

2.2. NP Domino Logic (Unfooted)

You can see the nMOS footing device at the bottom of the circuit on Figure 18-a and the pMOS footing device at the top of the circuit on Figure 18-b. The main purpose of the footing devices during the precharge phase can be defined in the following:

- For an nMOS device (Figure 18-a, bottom) to switch off the pull down path (evaluation path)
- For a pMOS device (Figure 18-b, top) to switch off the pull up path (evaluation path)

However, NP Domino Logic structure guarantees that the inputs of nMOS devices (N-type gates) are always precharged to low and the inputs of pMOS devices (P-type gates) are always precharged to high **for the stages higher than 1st one**; thus, the gates are by default precharged to the correct values just after the precharge phase is terminated and this state is preserved until there is an event at the input

of the gate. Therefore, for the higher stages, these devices can be removed and unfooted gates can be safely used (Figure 21).

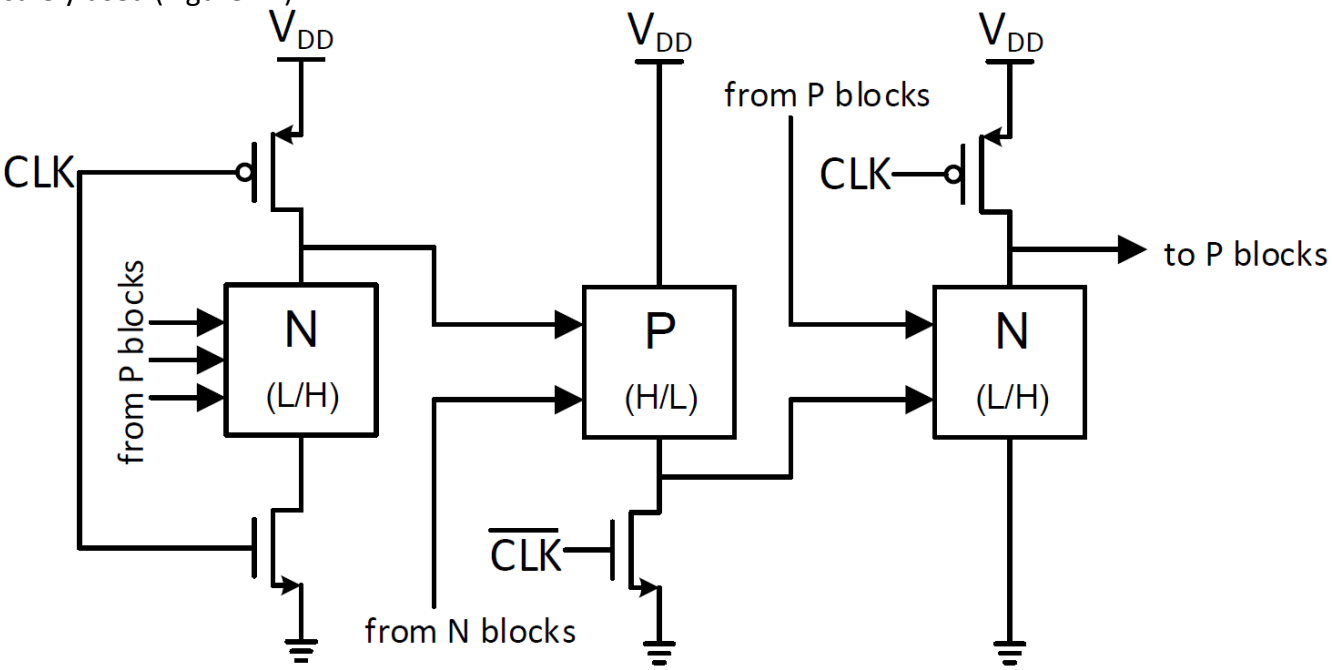


Figure 21: NP Domino Logic (unfooted)

Create a new schematic and call it ***DYN_NP_U_AN4_X1***. This cell will be the unfooted NP Domino Logic Implementation of the AND4 function (Figure 20-b and Figure 21). For the NAND2 and NOR2 gates create and use ***DYN_N_U_ND2_X1*** and ***DYN_P_U_NR2_X1*** cells and construct them.

Create also the simulation schematic called ***DYN_NP_U_AN4_X1_TB*** under the library ***VLSI2_TB***. Use the same simulation configuration of ***DYN_NP_U_AN4_X1_TB***. Note that you are going to simulate an NP Domino Logic circuit where **the first stage is also <u>unfooted</u>**. However, the input voltage levels are given such that they are always zero during the precharge phase. Therefore, the simulation will work correctly. Check the functionality of the gate and note the delay:

delay_A_Y: ..... [ps]	NP Domino AND4 (unfooted)

Case Study 5: Fill in the blanks
The unfooted NP Domino Logic is faster than the footed counterpart because • …… …… …… • …… …… …… The unfooted NP Domino Logic is more power efficient than the footed counterpart because • …… ……

……

- ……
  ……
  …..

Another superiority of unfooted NP Domino Logic on the footed counterpart is …… ……
……..

3. A SMALL DESIGN EXAMPLE

Design a digital circuit that performs the following logic function:

Y(A,B,C,D,E,F) = ABDE + CDE + ABDF + CDF

Design Constraints:

- Use NP Domino Logic
- Use unfooted gates as much as possible
- Use at most 3 gates (one gate is composed of at most one load and one logic tree and an additional footing device if necessary)
- For none of the gates, do NOT use more than 2 stacked transistor for the logic tree
- Do **NOT** forget to put the precharge devices to prevent charge sharing problem both for N-type and P-type gates
- Have a critical path delay less than **60ps** while driving a load capacitance of **5fF**.

READ PART 3 UNTIL THE END BEFORE STARTING YOUR DESIGN.

Step 1: Simplify the logic function to be suitable for NP Domino Logic Implementation (Hint: Use De Morgan's Law **in a reverse fashion** to see how **inverting gates** can be constructed).

Step 2: Draw the corresponding schematic and indicate the critical path.

Step 3: Create a simulation schematic and make necessary adjustments to observe the same waveform of Figure 22.

Step 4: After observing the right functionality, size the transistors to get a critical path delay less than 60ps under 5fF load.
Critical Path Delay: …… [ps]

Step 5 (BONUS): Draw the layout of this simple circuit.

Step 6 (BONUS): Perform post-layout simulations and write the new critical path delay with the

additional parasitic capacitances.
Critical Path Delay: …… [ps]

CALL AN ASSISTANT AND SHOW YOUR FINAL RESULTS OF PART 3

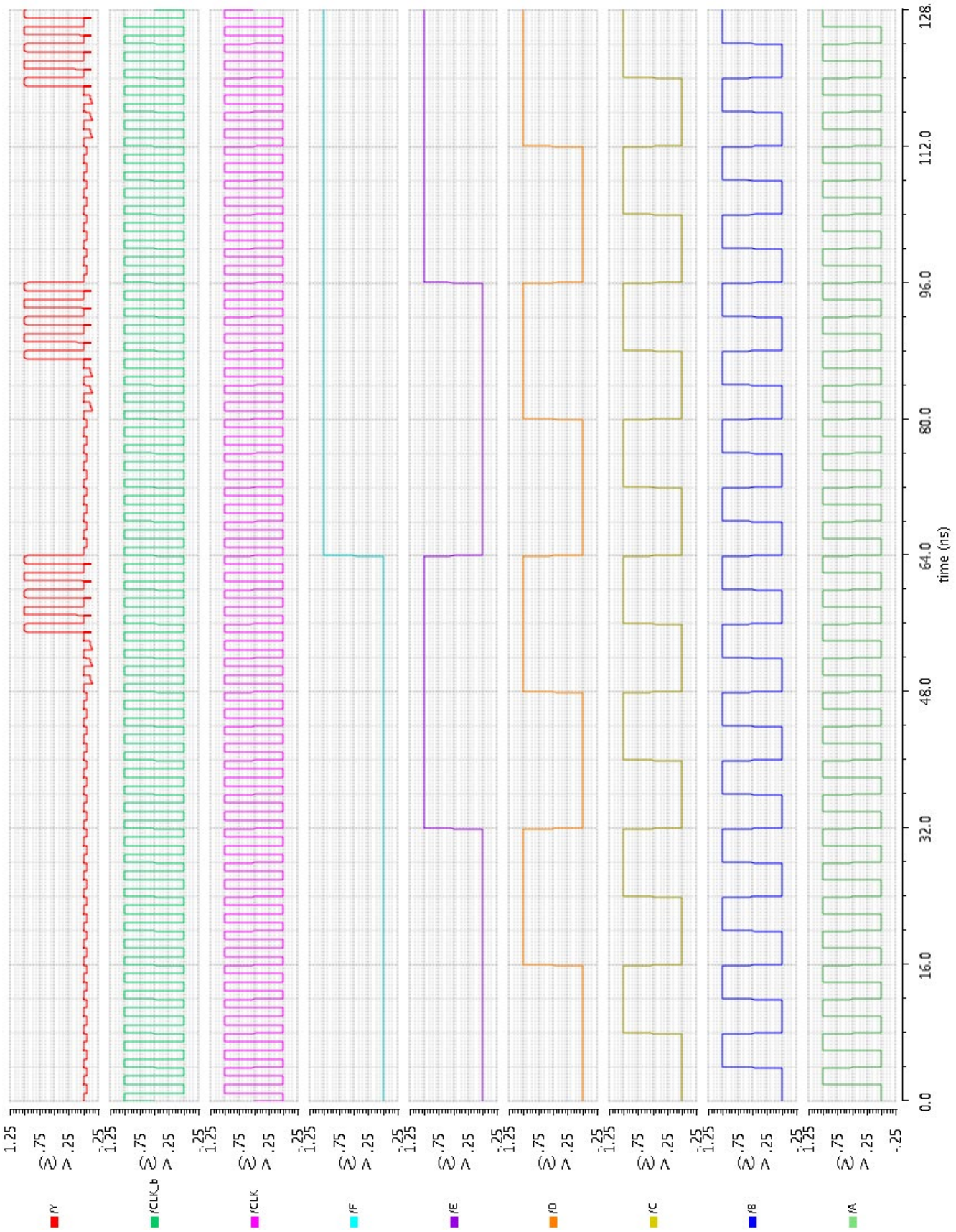


Figure 22: Expected waveforms of the simulation (in your case at VDD=1.2V)