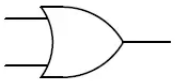
Segment 1 : Présentation du composant FPGA

1. Les fonctions logiques

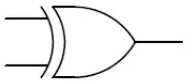
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

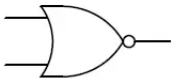


XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

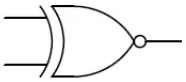
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



NOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

TD1

On suppose A = ‘0’ et B = ‘1’

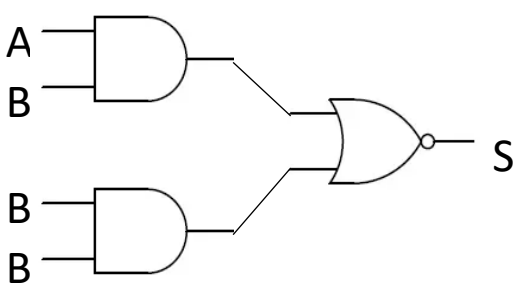
Quelle est la sortie des opérations suivantes ?

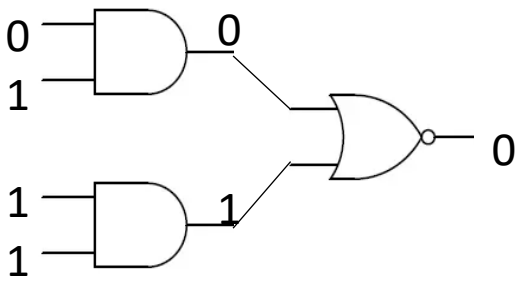
- **NOT((A AND B) OR (B AND B))**
- **(B XOR B) AND (A OR B)**

Représenter le schéma logique de ces opérations

Réponse

NOT((A **AND** B) **OR** (B **AND** B))



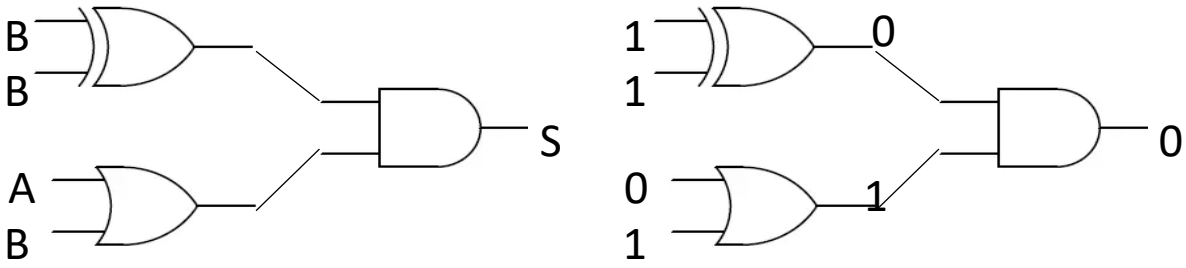


A=0 ; B=1 => A2=0

A1=1 toujours

S=0

(B **XOR** B) **AND** (A **OR** B)



TD2

La logique booléenne, réaliser l'addition « A + B » en utilisant de la logique booléenne, A et B sont deux chiffres représentés sur un 1 bit.

Dans un tableau représenter par colonne

Les tuples (combinaisons de valeurs) A et B en base binaire

Le résultat attendu S en base 10 (décimale) pour chaque tuple

Le résultat attendu S en base 2 (binaire)

Note : S sera représenté sur 2 bits

Réponse :

A	B	S (base 10)	S (base2)	S1	S0
0	0	0	00	0	0
0	1	1	01	0	1
1	1	2	10	1	0
1	0	1	01	0	1

Une addition base 2

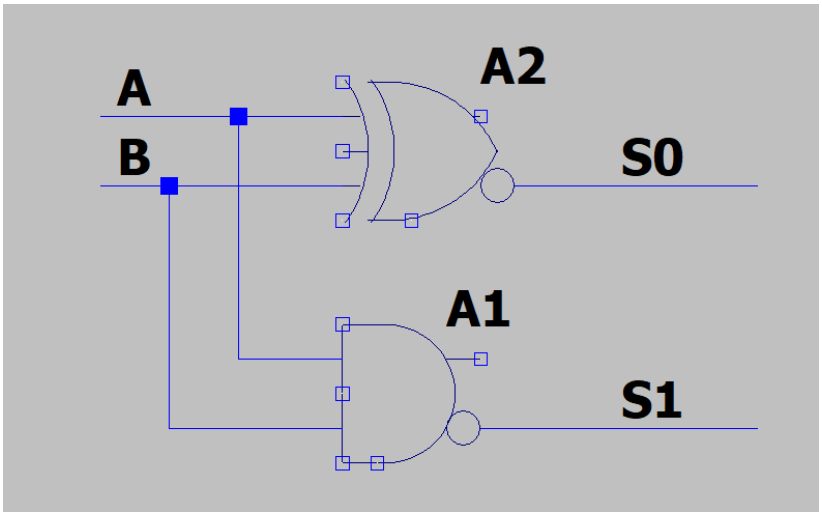
0+0=0 carried bit =0

0+1=1 carried bit=0

1+0=1 carried bit =0

1+1=0 carried bit =1

En comparant la tableau avec la table de vérité de porte logique, on a la porte logique AND pour la sortie S1 et porte logique XOR pour la sortie sS0.



TD3

Améliorer le circuit précédemment élaboré afin de lui ajouter une entrée pour le « carried bit input» que vous noterez C_{in}, S1 sera noté C_{out} pour « carried bit output » déterminer la table de vérité du circuit pour deux entrées A et B

Réponse :

A	B	Cin	Sum (base10)	Sum (base 2)	S	Cout (carried bit)
0	0	0	0	00	0	0
0	0	1	1	01	1	0
0	1	0	1	01	1	0
0	1	1	2	10	0	1
1	0	0	1	01	1	0
1	0	1	2	10	0	1
1	1	0	2	10	0	1
1	1	1	3	11	1	1

L'équation obtenue pour la sortie Cout est s'écrite :

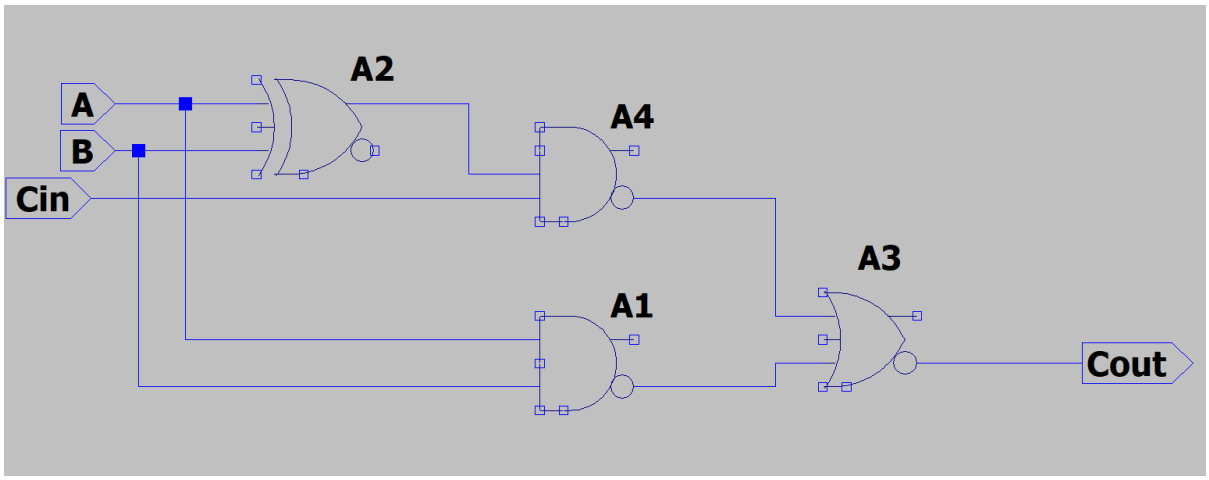
Cout= (nA.B.Cin)+(A.nB.Cin)+(A.B.nCin)+(A.B.Cin)=Cin.(nA.B+A.nB)+A.B.(Cin+nCin)

Cin+nCin=1 ;

nA.B+A.nB=A XOR B ;

Donc :

Cout=Cin.(A XOR B)+A.B



S=(nA.nB.Cin)+(nA.B.nCin)+(A.nB.nCin)+(A.B.Cin)=Cin.(nA.nB+A.B)+nCin.(nA.B+A.nB)

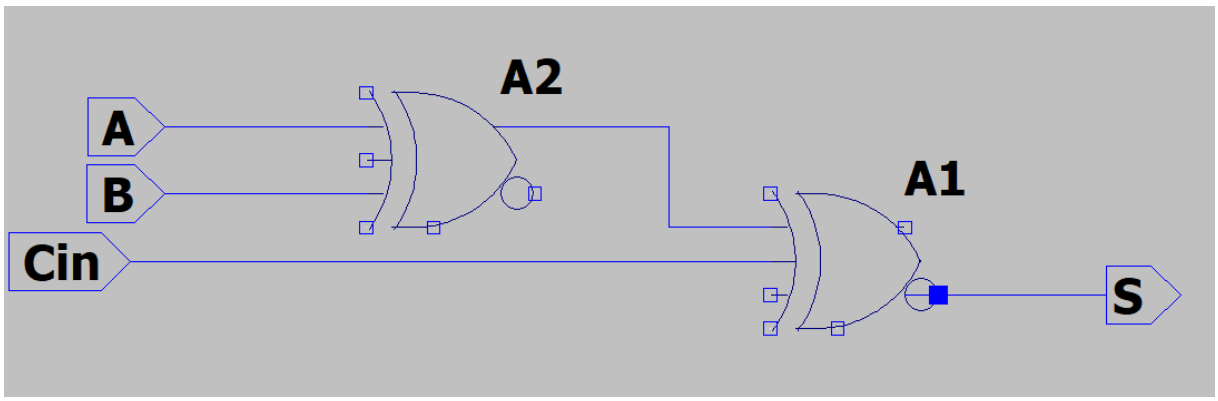
On a

nA.B+A.nB= A XOR B

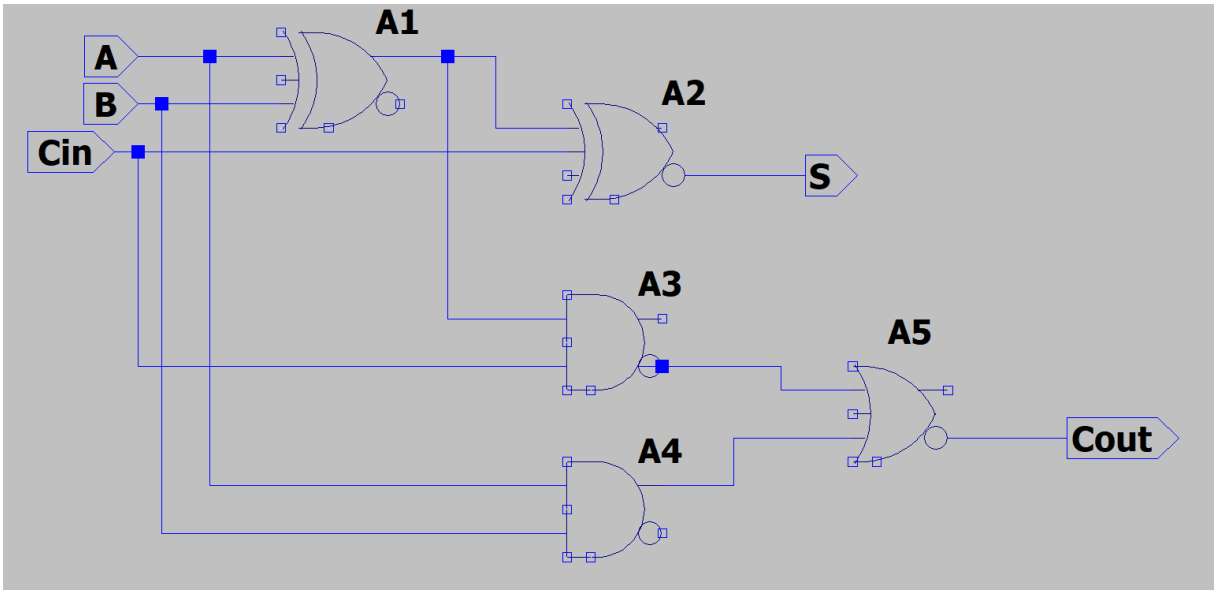
nA.nB+A.B=NOT(AXOR B)

donc

S=Cin.(NOT (A XOR B)) + nCin.(A XOR B) = C XOR (A XOR B)



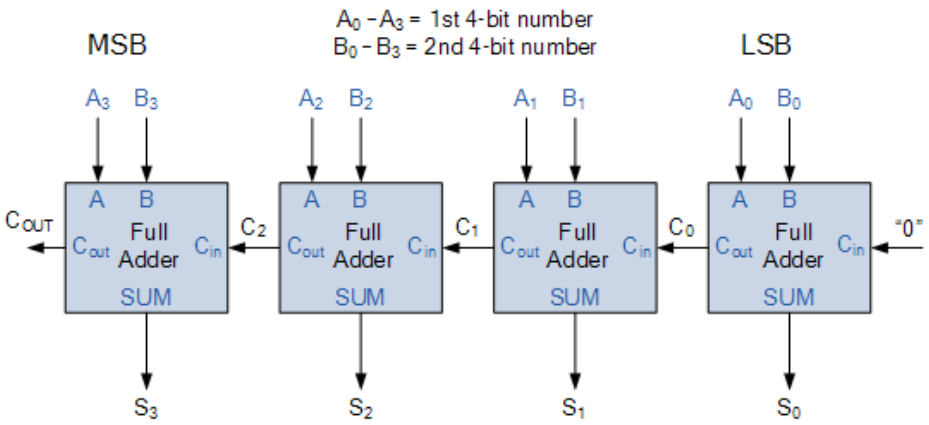
Le schéma d'un full-adder



TD 4

Réaliser un circuit combinant plusieurs full-adders pour additionner des chiffres sur 4 bits

Réponse



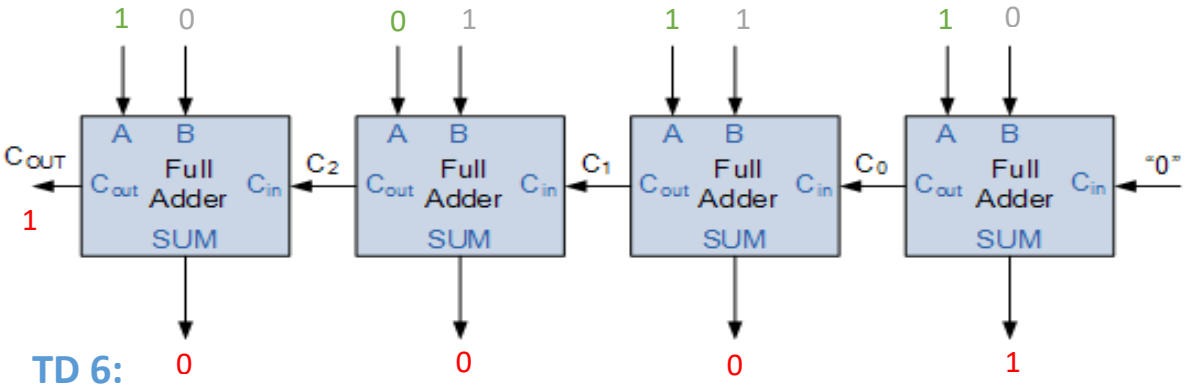
TD5

Présenter cette fonction sous la forme d'un paragraphe avec un exemple illustratif.

Réponse :

Soit A = 1011 dans base 2 ; A=B dans base 16 ; A=11 dans base 10

B = 0110 (base 2) B=6 (base 16) ; B=6 (base 10)



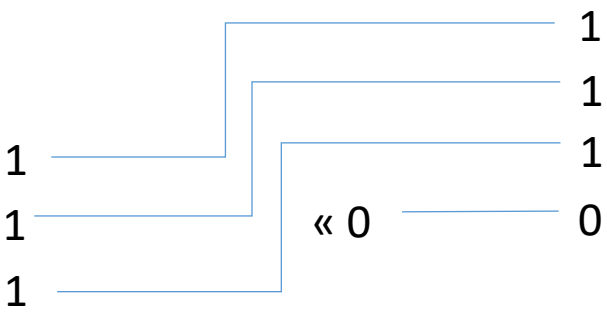
TD 6:

Proposer un circuit réalisant l'opération de multiplication par 2 d'une entrée sur 3 bits, la

Dec	A	B	C		x2	Out 3	Out2	Out1	Out0		/2	Out 3	Out2	Out1	Out0
0	0	0	0		0	0	0	0	0		0	0	0	0	0
1	0	0	1		2	0	0	1	0		0	0	0	0	0
2	0	1	0		4	0	1	0	0		1	0	0	0	1
3	0	1	1		6	0	1	1	1		1	0	0	0	1
4	1	0	0		8						2				
5	1	0	1		10						2				
6	1	1	0		12						3				
7	1	1	1		14						3				

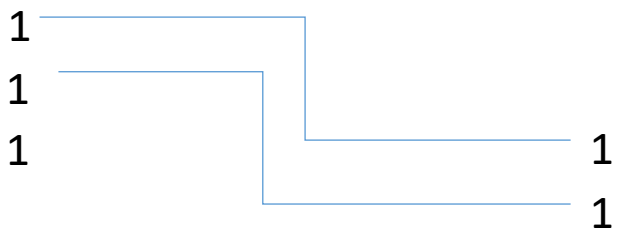
sortie sera donnée sur 4 bits (faire un arrondi inférieur).

Multiplication par 2



Décale vers à droite

Division par 2



Décale vers à gauche

TD 7

Étude opérations d'un calculateur type CPU, GPU ou micro-contrôleur. Supposons qu'un cycle d'inférence d'instruction prends environ 9 us (donc chaque instruction du pseudo code mettra 9 us à s'exécuter).

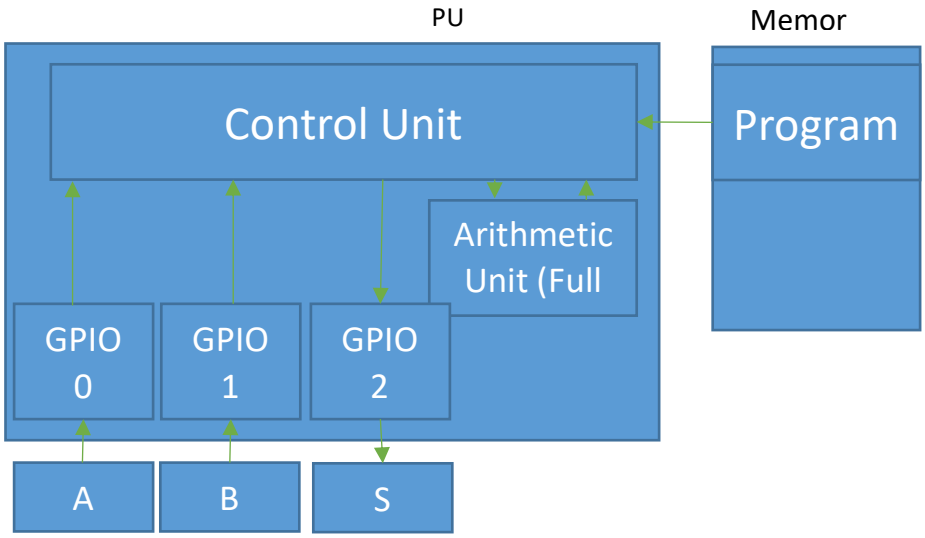
Calculer la latence du système, c'est-à-dire le temps écoulé entre le début de la première instruction du pseudo code et la fin de la dernière instruction

Calculer le temps qu'il faudrait pour répéter 500 fois le pseudo donné.

Réponse :

Le schéma du CPU

Les opérations d'un calculateur type CPU, GPU ou microcontrôleur sont effectué par 4 étapes successivement.



Pour un CPU, 4 étapes sont :

Lire GPIO 0

- Lire GPIO 1
- Appliquer Full-adder
- Mettre à jour GPIO 2

Chaque étape prend un temps latence de 9 us, le temps latence pour un cycle est calculé : 9us x 4 = 36 us. Pour une exécution de 500 itérations le temps latence est : 36us x 500 = 18 ms.

TD 8 :

Étude de cas addition deux chiffres par un FPGA

TD Supposons que A et B sont sur 4 bits chacun et que le temps de propagation d'une porte logique est de 5ps

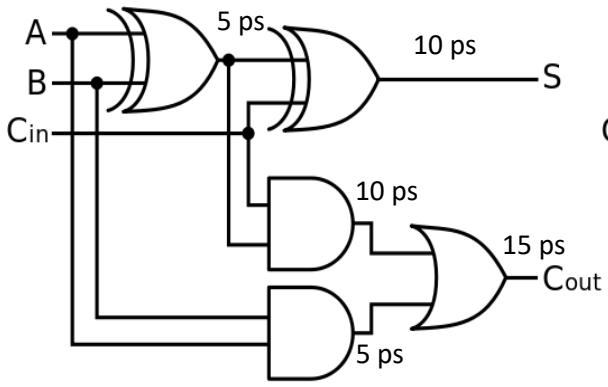
Calculer la latence du système, c'est-à-dire le temps écoulé entre le début de la première instruction du pseudo code et la fin de la dernière instruction

Calculer le temps qu'il faudrait pour répéter 500 fois le pseudo donné

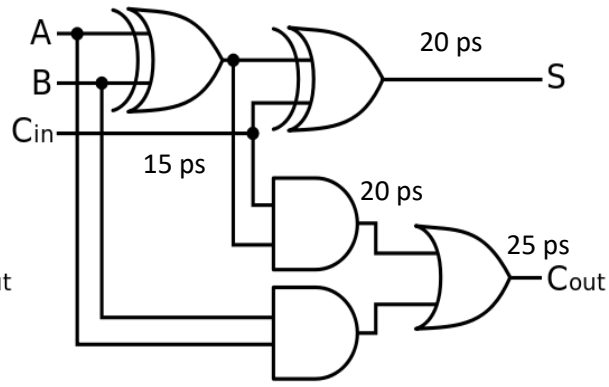
Réponse :

Dans un composant FPGA, on peut faire des calculs dit « à la volée », par exemple dans le schéma, les signaux A et B arrivent à une porte logique en même temps . Donc le temps latence du composant FPGA est réduit par rapport au temps latence du CPU, GPU ou microcontrôleur.

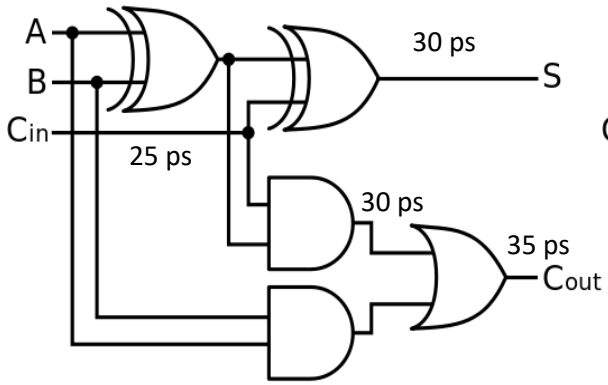
Nous étudions le temps latence pour un 4 bits full-adders :



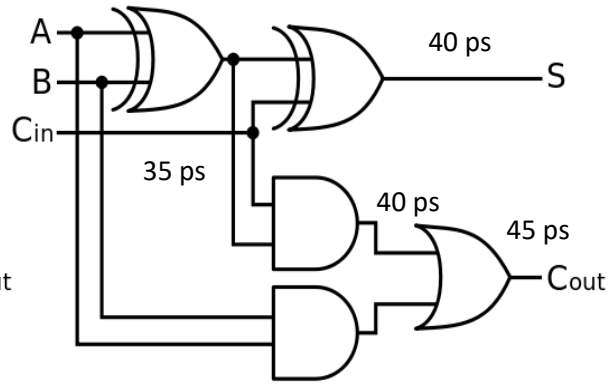
Adder 0



Adder 1



Adder 2



Adder 3

Le temps latence pour la porte S est de 40 ps, pour la porte C est de 45 ps. Le **chemin critique** est le chemin correspondent à porte avec le temps latence plus long, la porte C en ce cas.

Répéter 500 fois le calcul sur un full-adder de 4 bits, la latence est donc 500*45ps=22.5 us.

2. Architecture interner d'un FPGA

Un FPGA possède 3 composants principaux :

- Logic blocks (slices ou logique cellules)
- Programmable InterConnect (boite de routage)
- Input/Output Blocks

Une slice est regroupement de composants logiques bas niveau. Une Slice toujours contient :

-) Des LUTs (Look up table). La LUT permet de reproduire le comportement d'une porte logique.

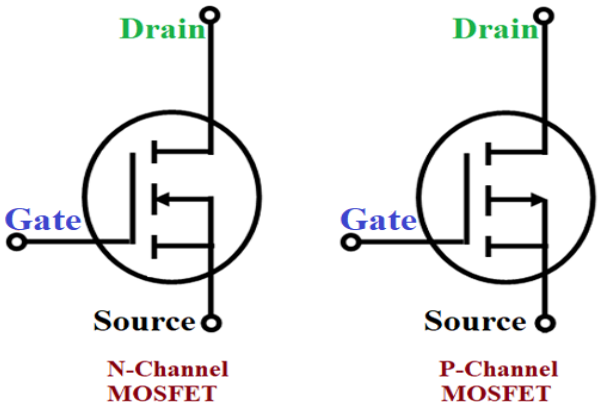
-) Le MUX (multiplexeur) permet de faire un aiguillage des signaux dans la **Slice** ou d'un Slice à une autre.

-) La FF, Flip Flop est un **registre** et permets la mémorisation d'un signal. Chaque FF permet de mémoriser 1 bit, c'est un composant **synchrone**.

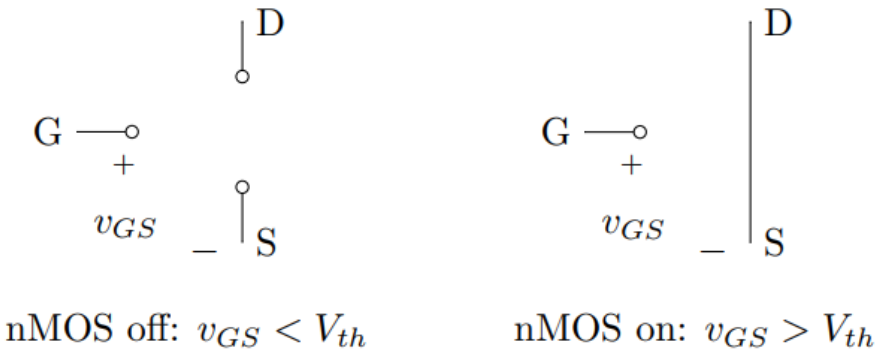
Électronique numérique fondamentale

Un transistor est un dispositif électronique utilisé pour permettre à un signal électrique de contrôler un autre signal électrique, généralement plus important en tension ou en courant. Les applications sont : amplificateurs ou « switches ».

MOS transistor ou MOSFET = metal–oxide–semiconductor field-effect transistor. Il exit 2 types de MOSFET : nMOS et pMOS

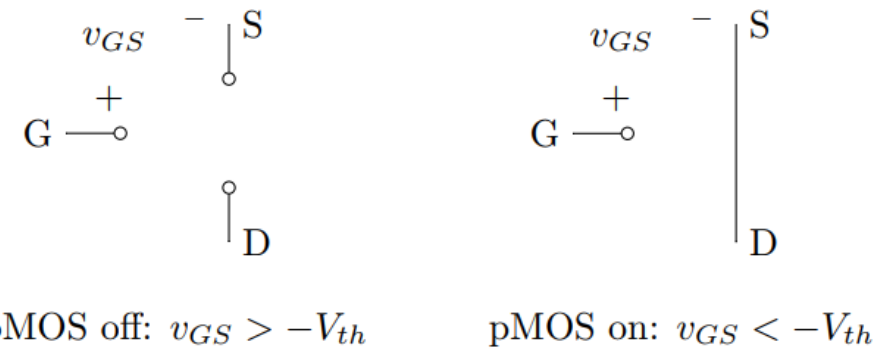


Le fonctionnement d'un transistor



Dans un nMOS, lorsque v_{GS} ($v_{GS}=v_G-v_S$) est supérieure à la tension de seuil V_{th} (Threshold) le transistor est fermé. Dans le cas contraire, le transistor est éteint et la connexion entre le drain et la source est ouverte.

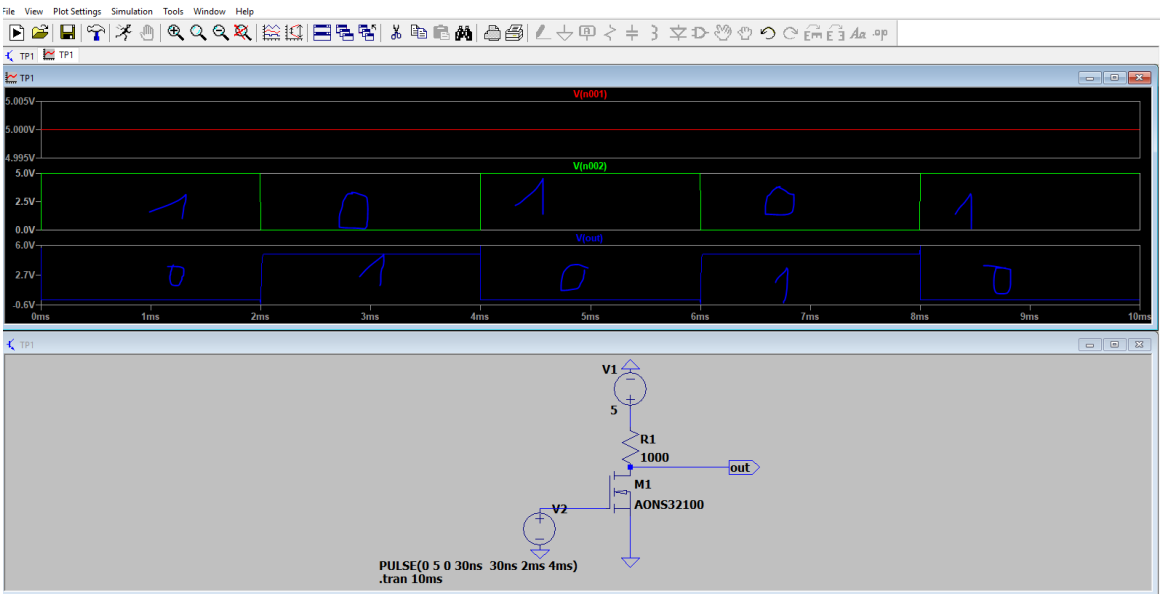
The pMOS is similar, except that it's flipped: it turns on when vGS < −Vth.



pMOS off: $v_{GS} > -V_{th}$ pMOS on: $v_{GS} < -V_{th}$

TD9

Modélisation de porte logique NOT et simulation sous Ltspice



Le circuit électrique comporte comme une porte logique NOT.

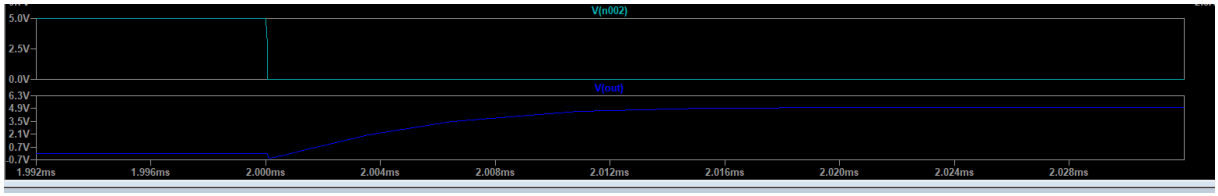
V_{th}=1.1 V pour nMOS AONS32100.

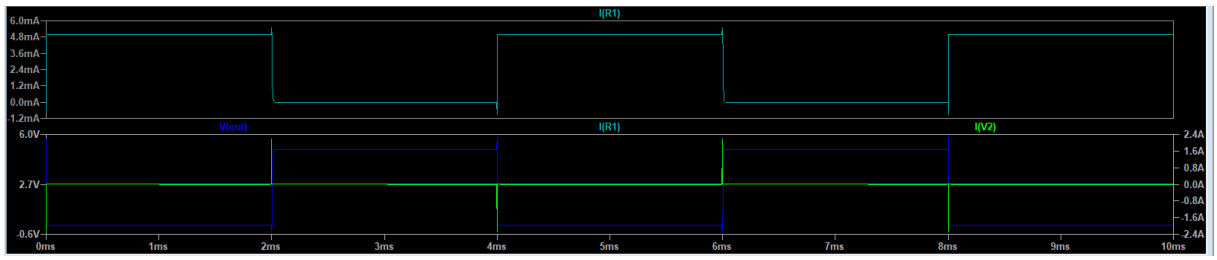
V_s=0 V toujours.

Si V_G= 5 V, V_{GS}=5 V > 1,1 V= V_{th}, nMOS est fermé. V_{out} connecte directement à la masse alors V_{out}= 0V.

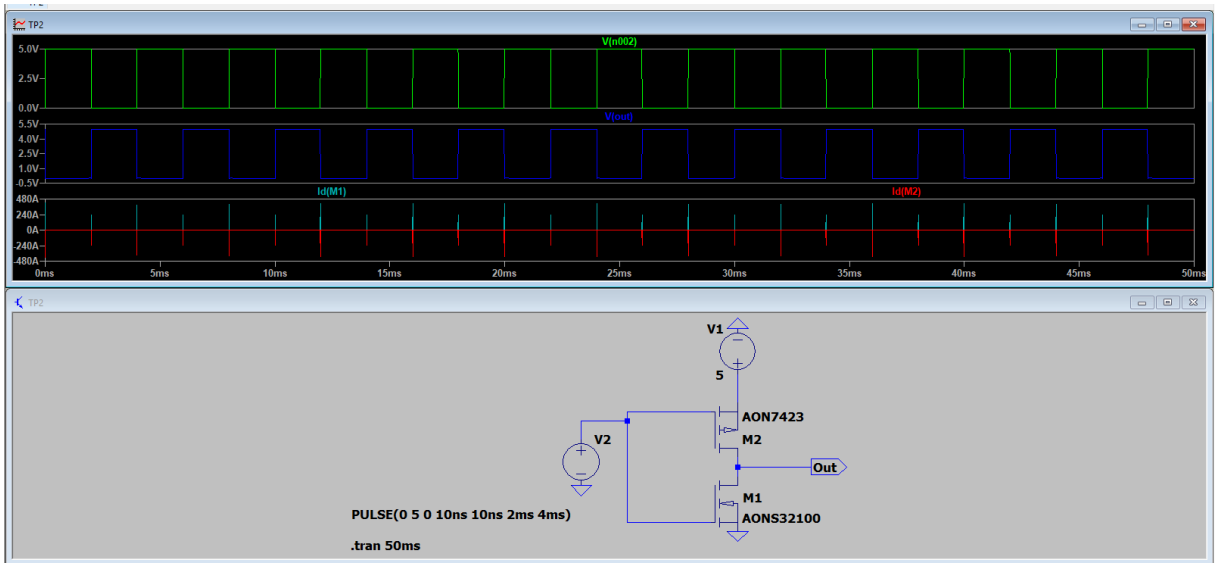
Si V_G= 0V, V_{GS}=0 V < 1,1 V= V_{th}, nMOS est ouvert V_{out}= V_1 – I.R avec I est l'intensité du courant qui passe la résistance R. Dans ce cas V_{out} # 0.

La latence de signal qui passe le nMOS, elle a ordre quelles us.





Le courant I (R) est toujours différent que zéro, la porte logique toujours consomme énergie. Le pic étroit sur le chronogramme de V_{out} se produit lorsque le nMOS change l'état : fermé--→ ouvert ou ouvert ---→ fermé.



Le circuit avec 2 MOSFET : un pMOS (AON7423) et un nMOS (AONS32100) se comporte comme une porte logique NOT également. L'avantage de ce circuit :

-) Le circuit consomme moins énergie par rapport au circuit précèdent : lorsque deux MOSFET grade ses états, le courant électrique du circuit est égale à zéro. Le circuit consomme l'énergie seulement quand deux MOSFET changent l'état.

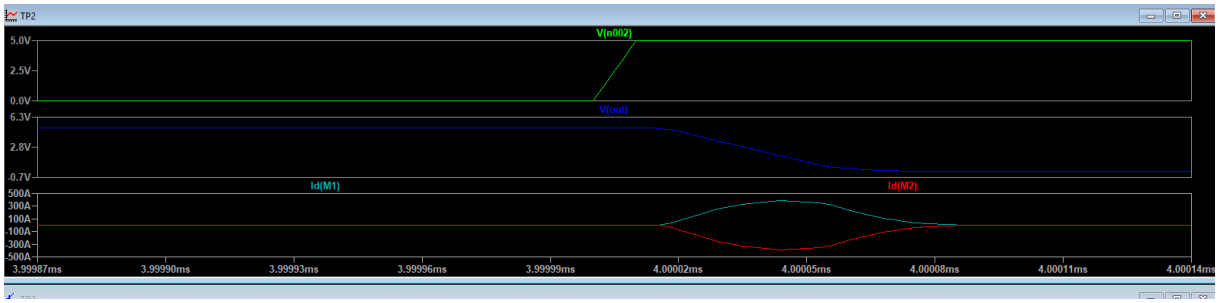
-) Le signal du sortie V_{out} ne se produit pas le peak lorsque les MOSFET changent l'état cas deux MOSFET consomme 2 courant avec le sens inverse.

V_G=5V, pour le pMOS, Vs= 5 V, donc v_{GS} = 0V > -V_{th}= -0.5 V alors pMOS est ouvert.

Pour le nMOS, Vs= 0 V, donc v_{GS} = 5V > V_{th}= 1.1 V, nMOS est fermé, V_{out} =0 V relie directement à la masse.

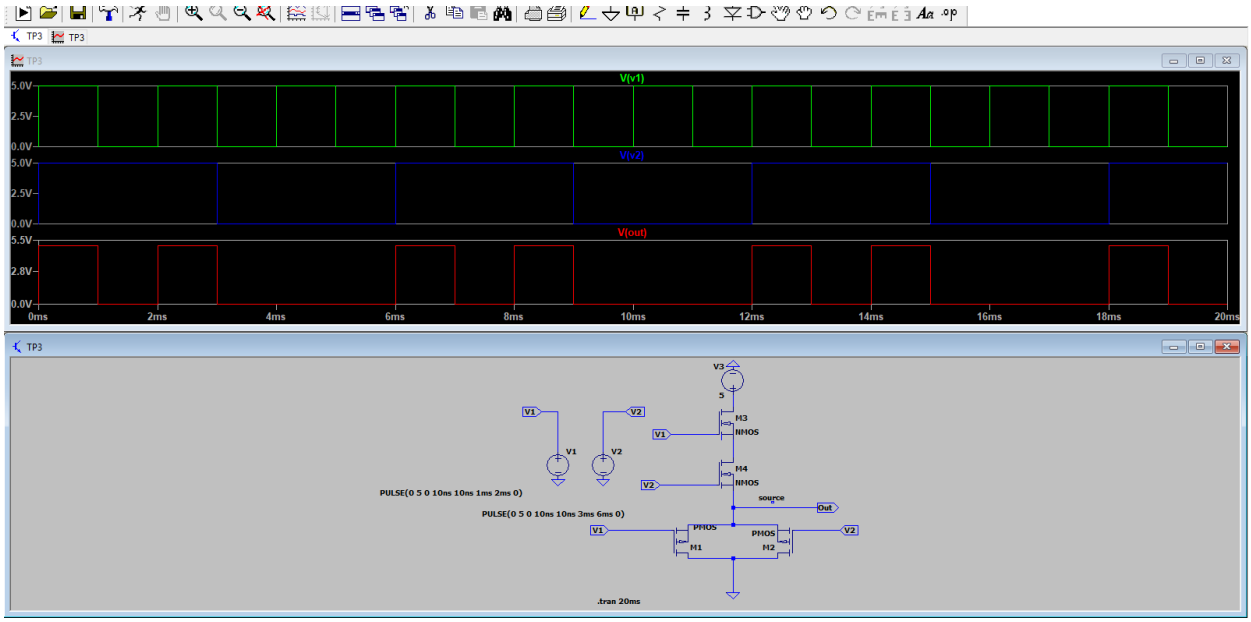
V_G= 0 V, pour le pMOS, v_{GS} = -5V < -V_{th}= -0.5 V alors pMOS est fermé, V_{out} # 0 V. Par contre, pour le n MOS v_{GS} = 0V < V_{th}= 1.1 V, le nMOS est ouvert.

La latence du circuit



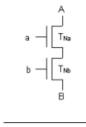
TD10 :

Modélisation de porte logique AND et simulation sous Ltspice



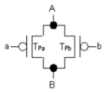
Le circuit se comporte comme une porte logique AND.

En utilisant la table vérité de deux transistors NMOS en série et deux transistors PMOS en parallèle, nous pouvons vérifier le signal V_{out} en fonction deux signaux rentrées V_1 et V_2.



Entrées		Transistors		Branche
a	b	T_{N_a}	T_{N_b}	F_{AB}
0	0	O	O	O
0	1	O	F	O
1	0	F	O	O
1	1	F	F	F

Figure : Deux transistors NMOS en série



Entrées		Transistors		Branche
a	b	T_{P_a}	T_{P_b}	F_{AB}
0	0	F	F	F
0	1	F	O	F
1	0	O	F	F
1	1	O	O	O

Figure : Deux transistors PMOS en parallèle

Cas V_1=5V, V_2=5V : M3 et M4 sont fermés, M1 et M2 sont ouverts ; donc V_{out} relie à V_3=5V.

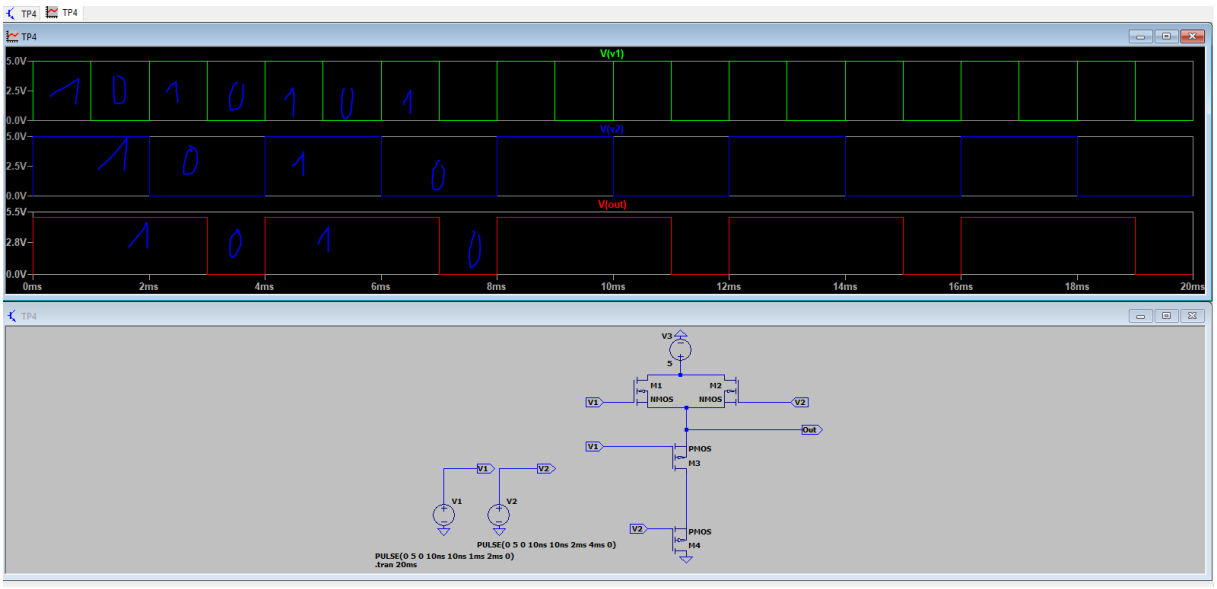
Cas V_1=0V, V_2=5V : M3 est ouvert, M4 est fermé, M1 est fermé, M2 est ouvert. V_{out} relie la masse dans ce cas. Le signal de sortie est de 0 V.

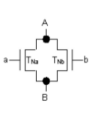
Cas V_1= 5V, V_2=0 V : M3 est fermé, M4 est ouvert, M1 est ouvert, M2 est fermé, V_{out} relie la masse dans ce cas. Le signal de sortie est de 0 V.

Cas V_1= 0V, V_2=0 V : M3 et M4 sont ouverts, M1 et M2 sont fermés. V_{out} relie la masse dans ce cas. Le signal de sortie est de 0 V.

TD 11:

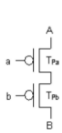
Modélisation de porte logique OR et simulation sous Ltspice





Entrées		Transistors		Branche
a	b	T_{N_a}	T_{N_b}	F_{AB}
0	0	O	O	O
0	1	O	F	F
1	0	F	O	F
1	1	F	F	F

Figure : Deux transistors NMOS en parallèle



Entrées		Transistors		Branche
a	b	T_{P_a}	T_{P_b}	F_{AB}
0	0	F	F	F
0	1	F	O	O
1	0	O	F	O
1	1	O	O	O

Figure : Deux transistors PMOS en série

V_1=5V, V_2=5V : M1 et M2 sont fermés, M3 et M4 sont ouverts. Donc V_{out} relie à V_3=5V.

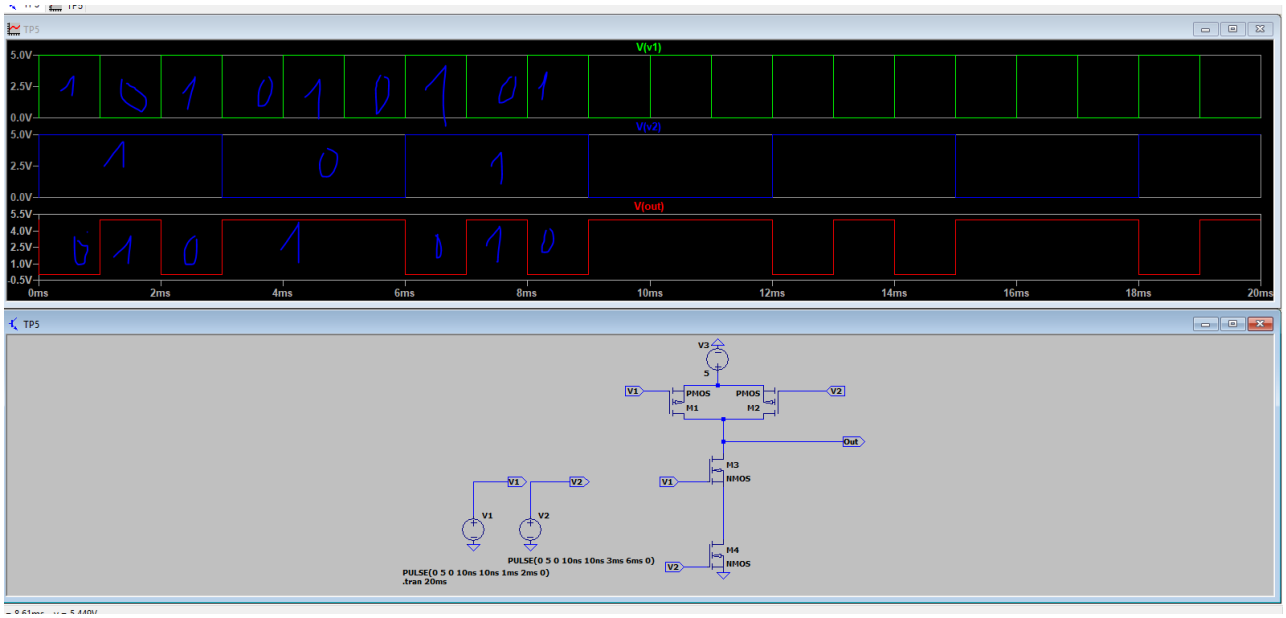
V_1=0V, V_2=5V : M1 est ouvert et M2 est fermé, M3 est fermé et M4 est ouvert. Donc V_{out} relie à V_3=5V.

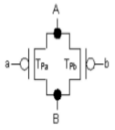
V_1=5V, V_2=0V : M1 est fermé et M2 est ouvert, M3 est ouvert et M4 est fermé. Donc V_{out} relie à V_3=5V.

V_1=0V, V_2=0V : M1 et M2 sont ouverts, M3 M4 sont fermés. Donc V_{out} relie à la masse, V_{out}=0V.

TD 12 :

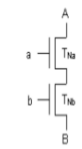
Modélisation de porte logique NAND et simulation sous Ltspice





Entrées		Transistors		Branche
a	b	T_{P_a}	T_{P_b}	F_{AB}
0	0	F	F	F
0	1	F	O	F
1	0	O	F	F
1	1	O	O	O

Figure : Deux transistors PMOS en parallèle



Entrées		Transistors		Branche
a	b	T_{N_a}	T_{N_b}	F_{AB}
0	0	O	O	O
0	1	O	F	O
1	0	F	O	O
1	1	F	F	F

Figure : Deux transistors NMOS en série

V_1=5V, V_2=5V : M1 et M2 sont ouverts, M3 et M4 sont fermés. Donc V_{out} relie à la masse.

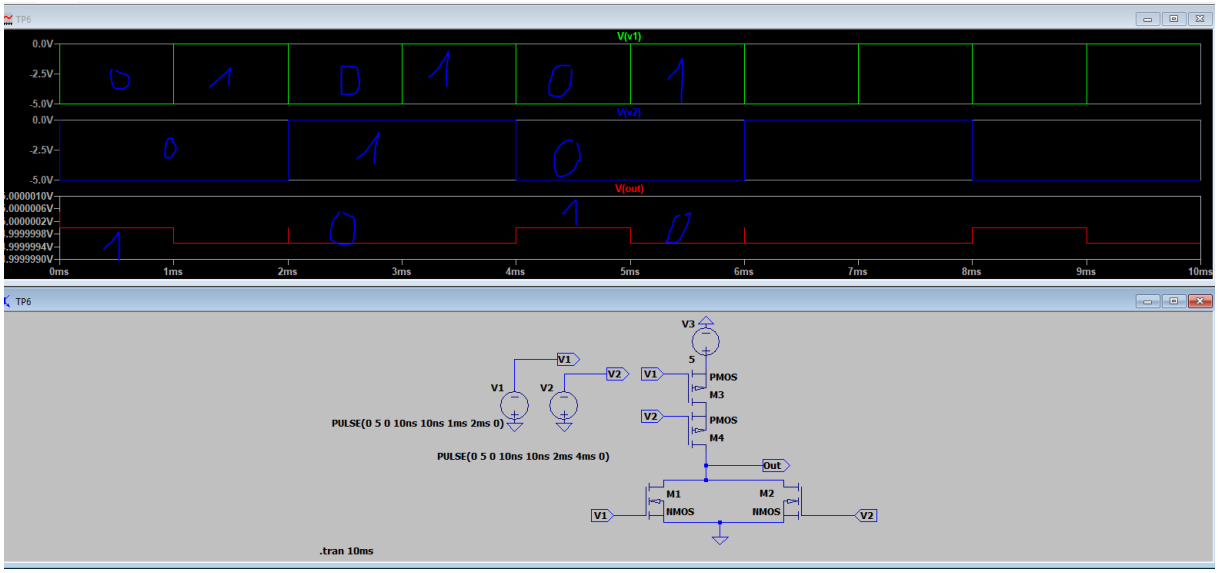
V_1=0V, V_2=5V : M1 est fermé et M2 est ouvert, M3 est ouvert et M4 est fermé. Donc V_{out} relie à V_3=5V.

V_1=5V, V_2=0V : M1 est ouvert et M2 est fermé, M3 est fermé et M4 est ouvert. Donc V_{out} relie à V_3=5V.

V_1=0V, V_2=0V : M1 et M2 sont fermés, M3 M4 sont ouverts. Donc V_{out} relie à V_3=5V. Donc V_{out} relie à V_3=5V.

TD 13

Modélisation de porte logique **NOR** et simulation sous Ltspice



Entrées		Transistors		Branche
a	b	T_{N_a}	T_{N_b}	F_{AB}
0	0	O	O	O
0	1	O	F	F
1	0	F	O	F
1	1	F	F	F

Figure : Deux transistors NMOS en parallèle

Entrées		Transistors		Branche
a	b	T_{P_a}	T_{P_b}	F_{AB}
0	0	F	F	F
0	1	F	O	O
1	0	O	F	O
1	1	O	O	O

Figure : Deux transistors PMOS en série

V_1=0V, V_2=0V : M3 et M4 sont fermés, M1 et M2 sont ouverts. Donc V_{out} relie à V_3=5V. Donc V_{out} relie à V_3=5V

V_1=5V, V_2=0V : M1 est fermé et M2 est ouvert, M3 est ouvert et M4 est fermé. Donc V_{out} relie à la masse.

V_1=0V, V_2=5V : M1 est ouvert et M2 est fermé, M3 est fermé et M4 est ouvert. Donc V_{out} relie à la masse.

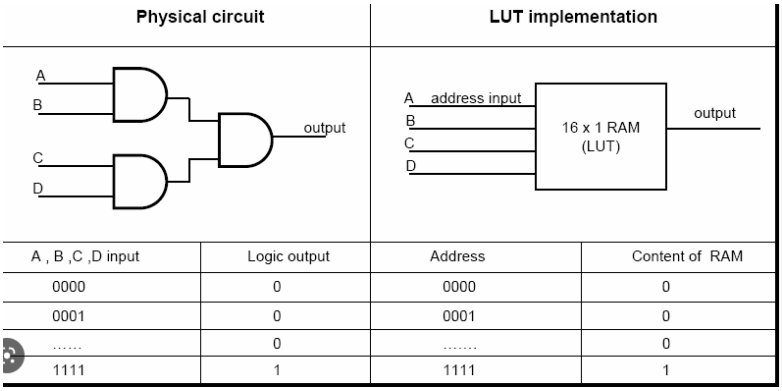
V_1=5V, V_2=5V : M1 et M2 sont fermés, M3 et M4 sont ouverts. Donc V_{out} relie à la masse.

TD 14:

Modélisation de portes logiques et simulation sous Ltspice

Si les portes logiques sont assemblées avec des transistors **gravés dans le silicium**, pourquoi dans un FPGA toutes les portes logiques sont **reconfigurables** ?

Ce sont les LUTs qui permettent cette reconfiguration du composant, physiquement parlant, une LUT est une nano unité de mémoire qui contient la table de vérité d'un circuit. La LUT étant une mémoire, possède un bus d'adresse sur lequel on connecte nos entrées.



A , B ,C ,D input	Logic output	Address	Content of RAM
0000	0	0000	0
0001	0	0001	0
.....	0	.....	0
1111	1	1111	1

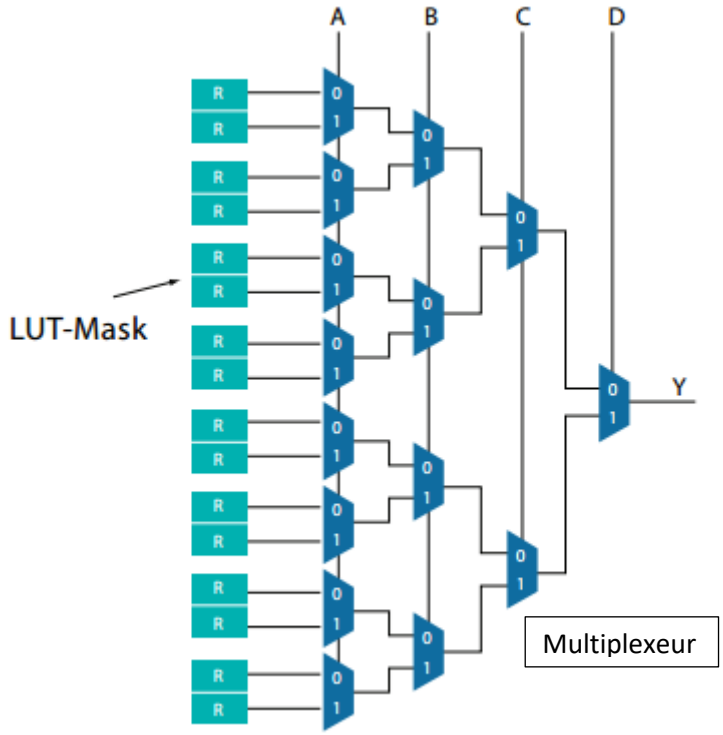
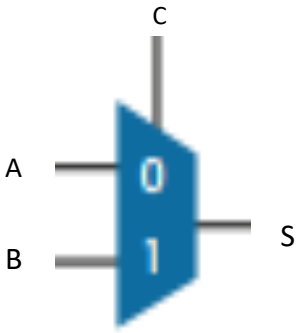
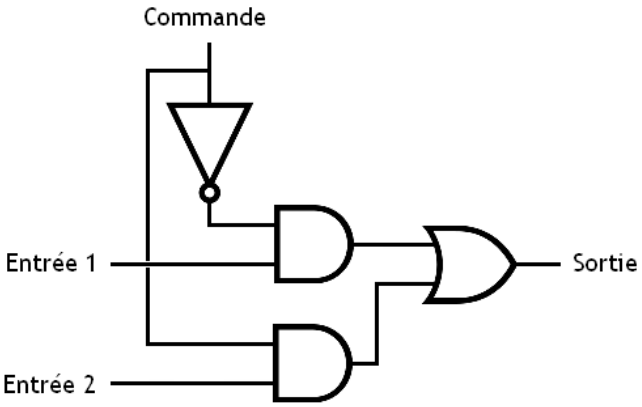


Schéma représentant une nano unité de mémoire.

La table vérité du multiplexeur

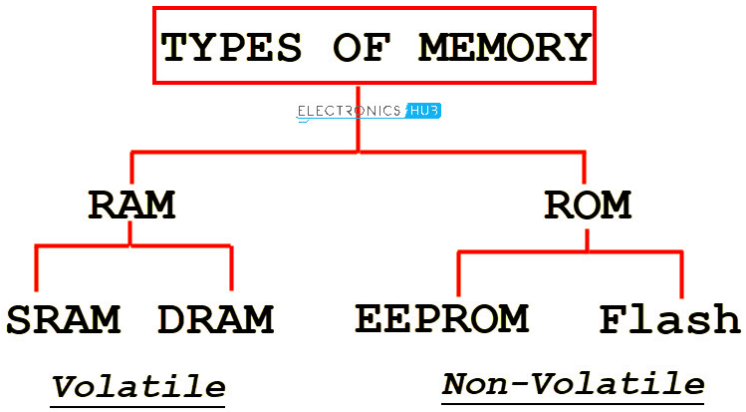


C	S
0	A
1	B



Le circuit physique d'un multiplexeur à deux entrées.

Plus part du temps, on utilise de SRAM (Static Random Acces Memorie) pour constituer une LUT. Mais sur certaines applications on peut employer du flash notamment pour que le temps de programmation du FPGA soit le plus court possible et donc qu'il soit **opérationnel le plus tôt possible.** Pour enregistrer le bitstream et donc pour programmer les LUTs, tous les types de mémoire peuvent être utilisé; avec leur avantages et inconvénients de chacun.



TD 15

Construire un full-substractor 4bits

Réponse

La table de vérité d'un full-substractor

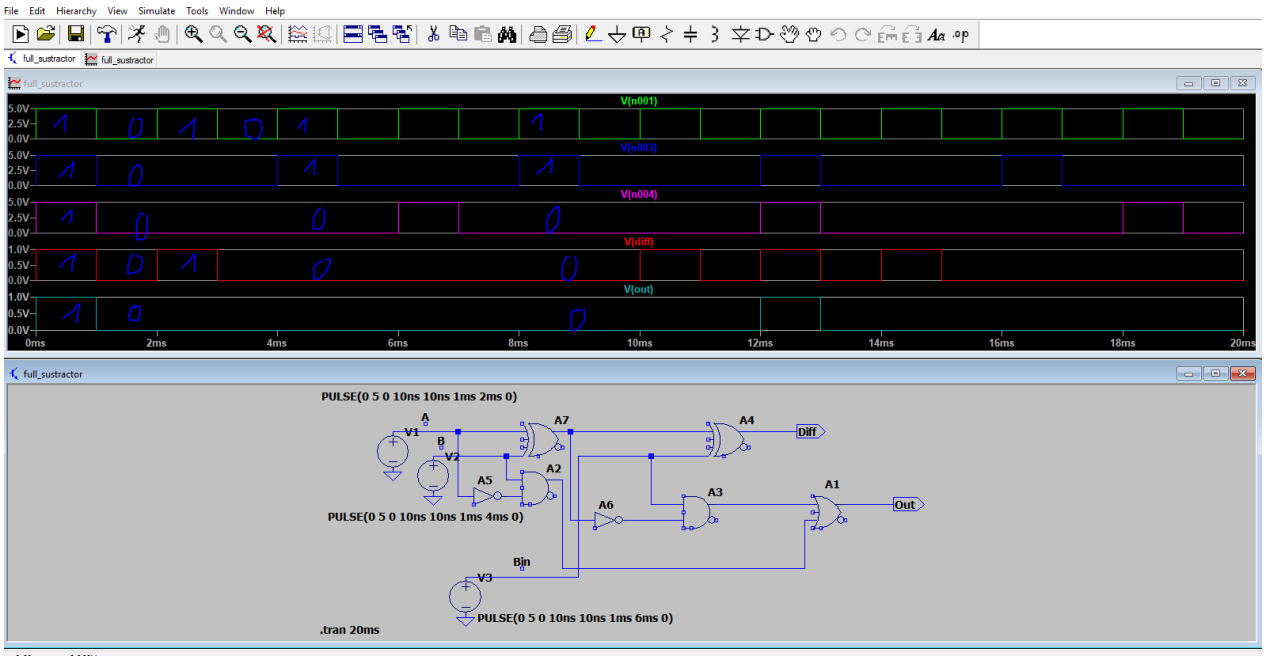
Inputs			Outputs	
A	B	B_{in}	D	B_{out} (carried bit)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	1	1	1	1
1	1	0	0	0
1	0	0	1	0
1	0	1	0	0

Note : 0-1= 1 et carried bit =1.

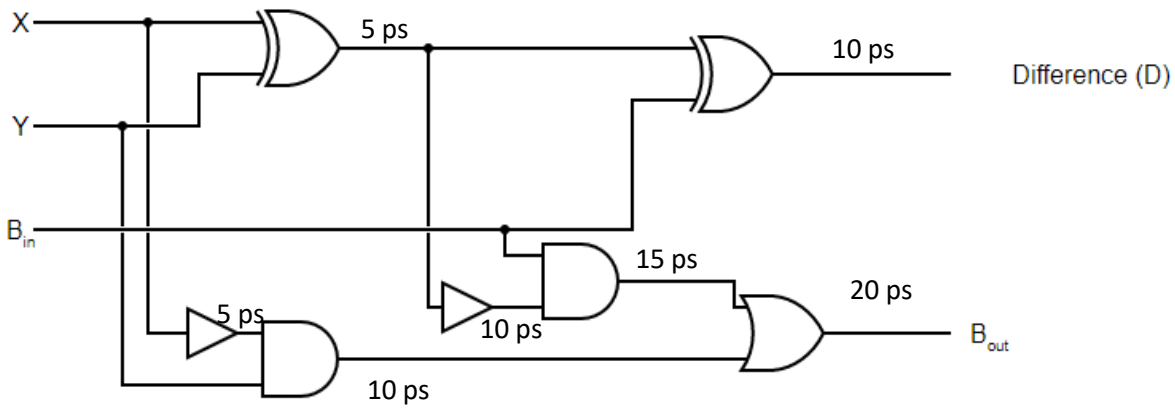
Les équations logiques pour le full-substractor

D= A XOR B XOR B_{in}

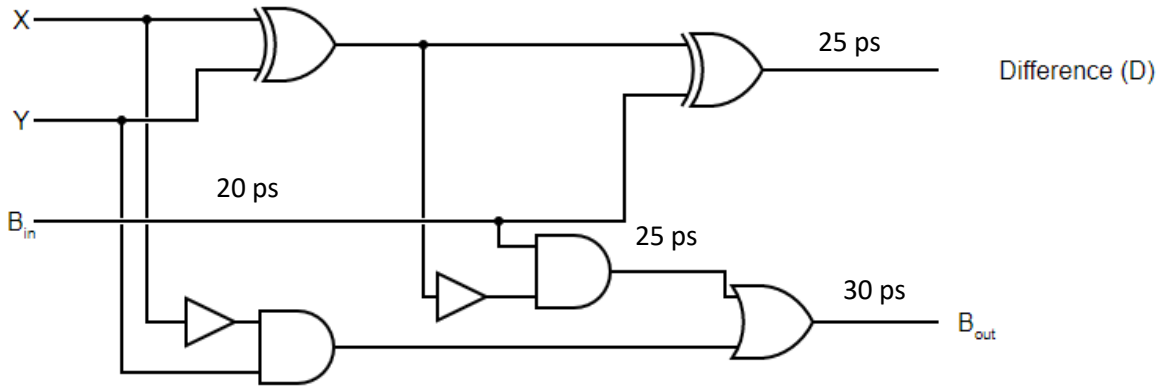
B_{out} = not(A). B_{in} + not(A).B + B. B_{in}



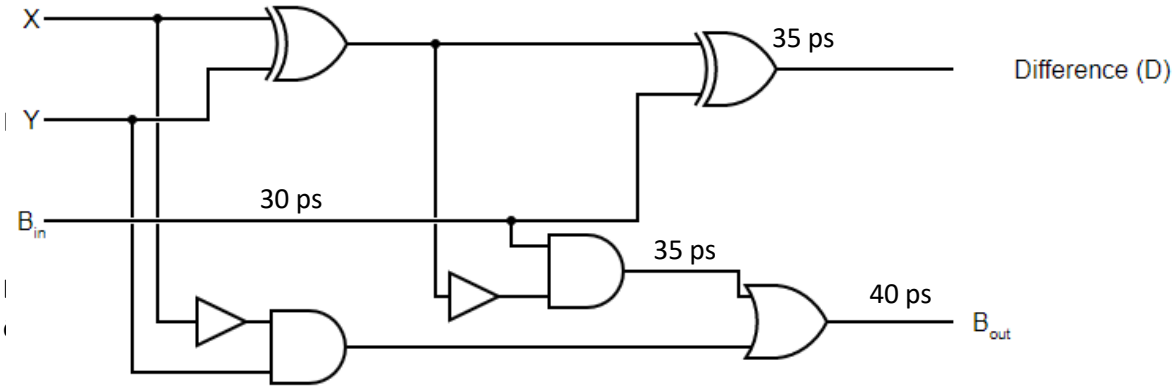
Le schéma électrique et la table de vérité sont réalisés par LT-spice.



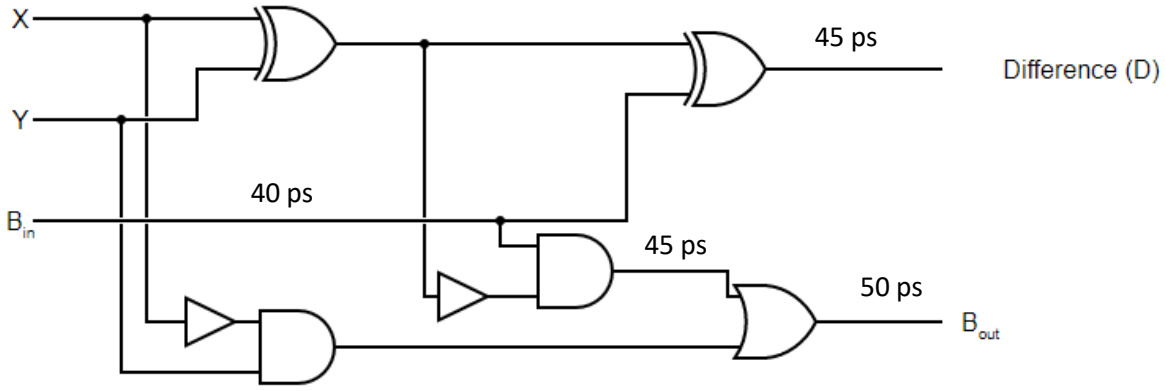
Subtractor 0



Subtractor 1



Subtractor 2



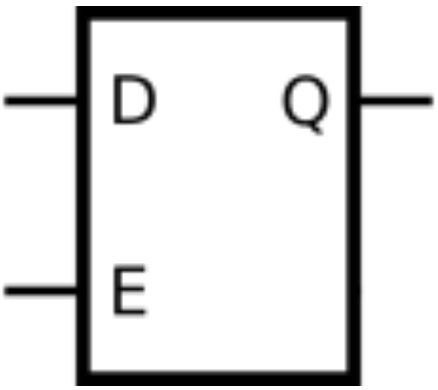
Subtractor 3

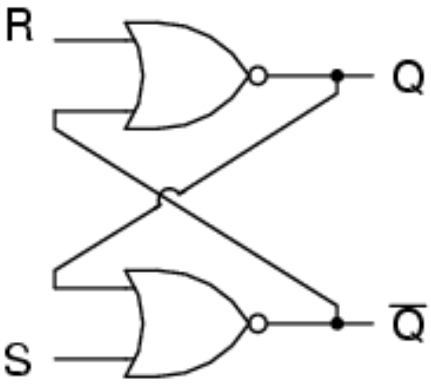
Le chemin critique est le chemin du porte B_{out}.

3. Logique numérique synchrone

La bascule (le latch)

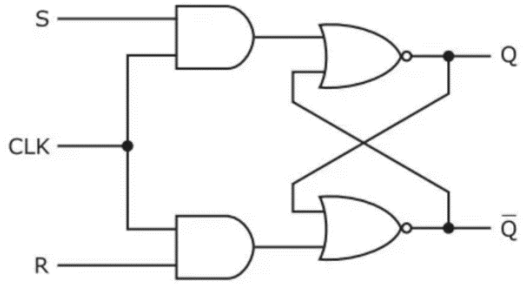
Un latch est un élément de circuit numérique qui a deux états stables et peut être utilisé **pour stocker des informations.** Les deux **états stables** d'un latch sont communément appelés états set et reset. Lorsqu'on entre dans l'état latch (S = 0 et R = 0) alors la sortie Q conserver sa valeur d'état précédent

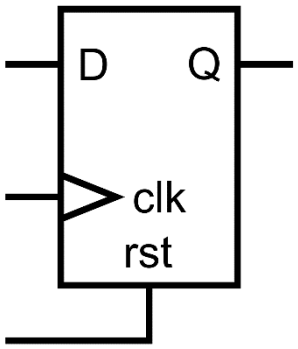




S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

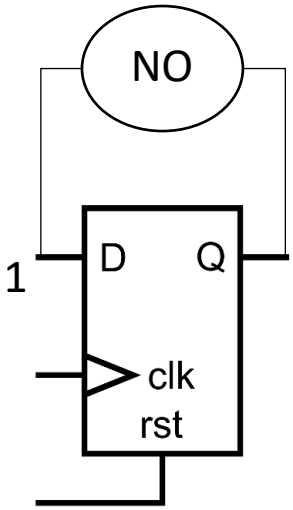
Le registre (flip flop) : On ajoute un étage de portes AND et un nouveau signal CLK qui correspond à notre horloge

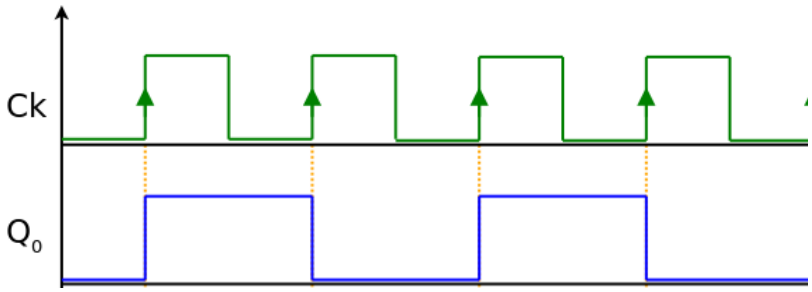




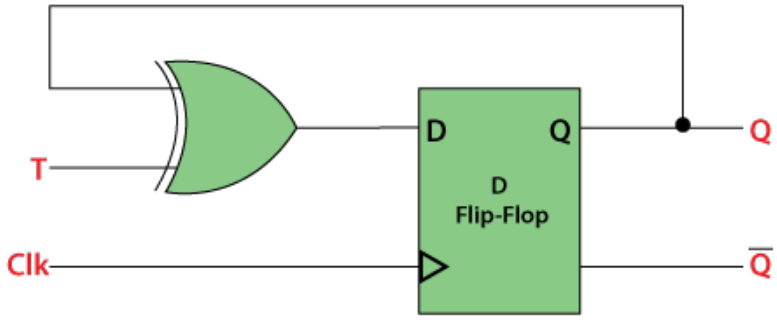
Le registre (flip flop) : la D-flipflop

Le registre va alterner entre la valeur 0 et 1 (porte not dans la contre-réaction), **sur front d'horloge.** On peut décider que la mise à jour soit faite sur front montant, descendant ou les deux.





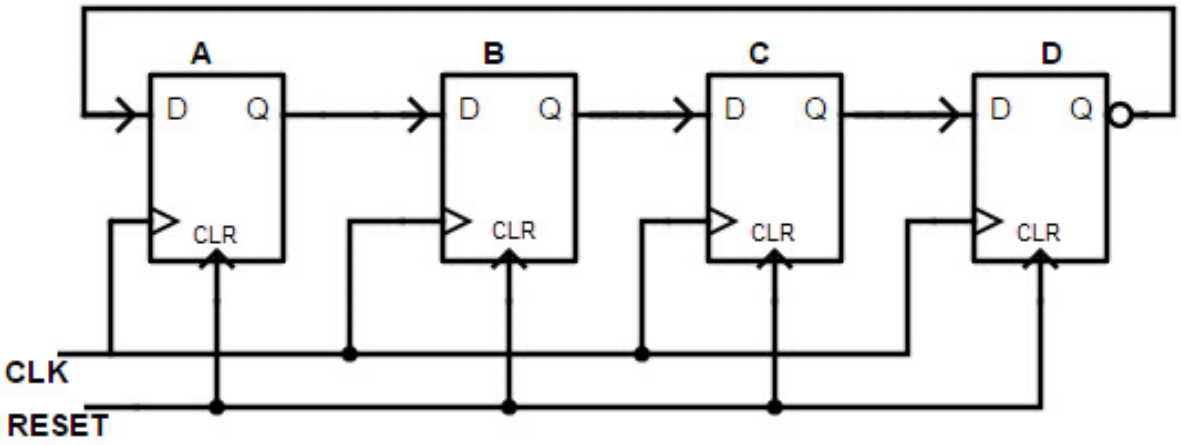
Le registre (flip flop) : La T-Fliflop



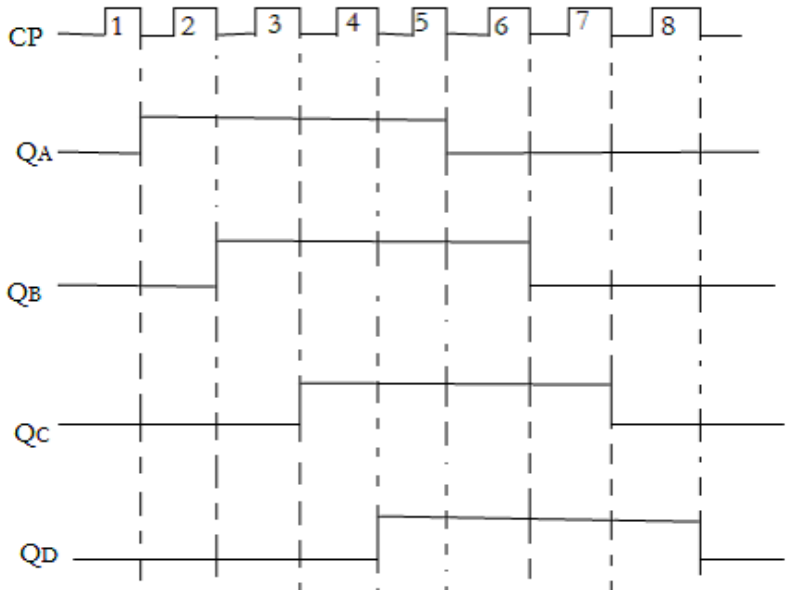
Sur front front montant d'horloge, si l'entrée du registre est à 1, alors la valeur courante du registre est inversée. Sinon, la valeur du registre est inchangée.

Les compteurs

Compteur 4bit, schéma RTL et chronogramme

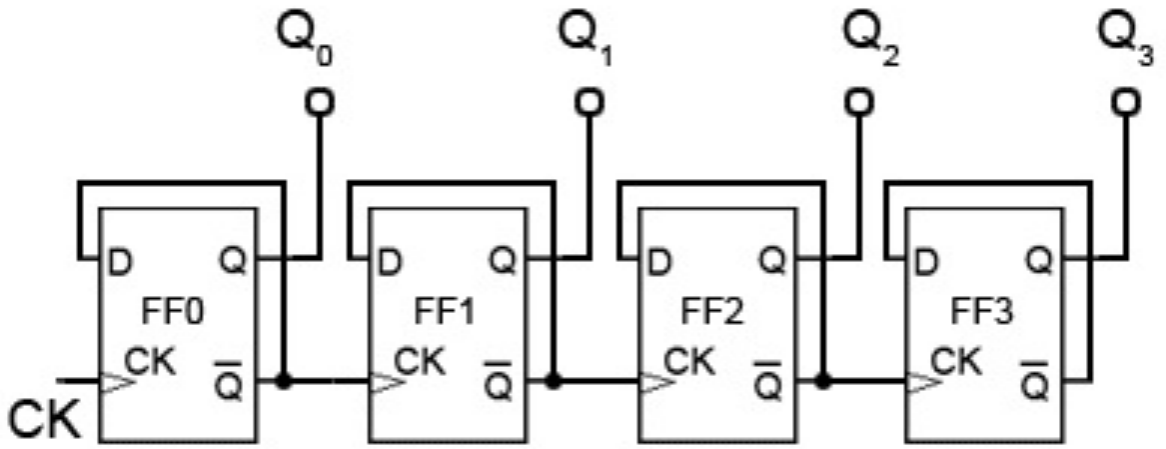


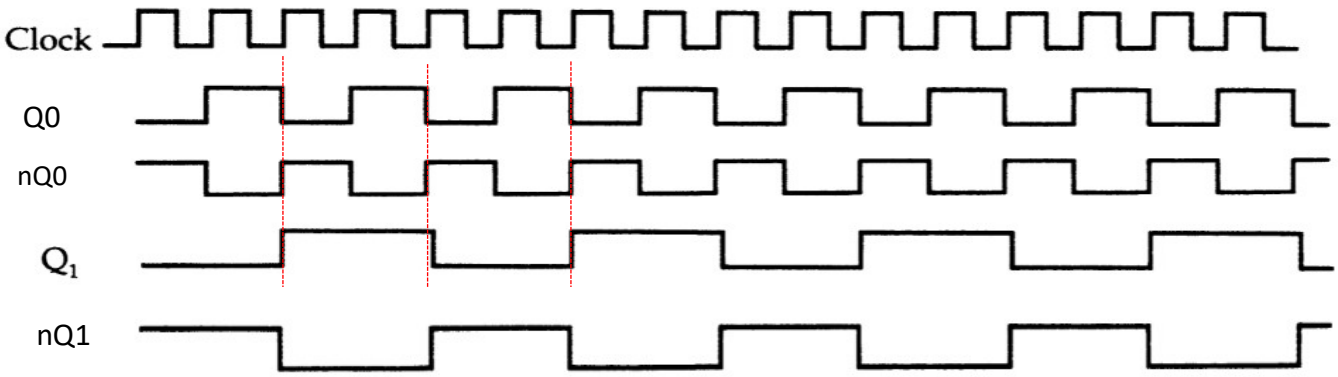
Q_A	Q_B	Q_C	Q_D
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
repeat			



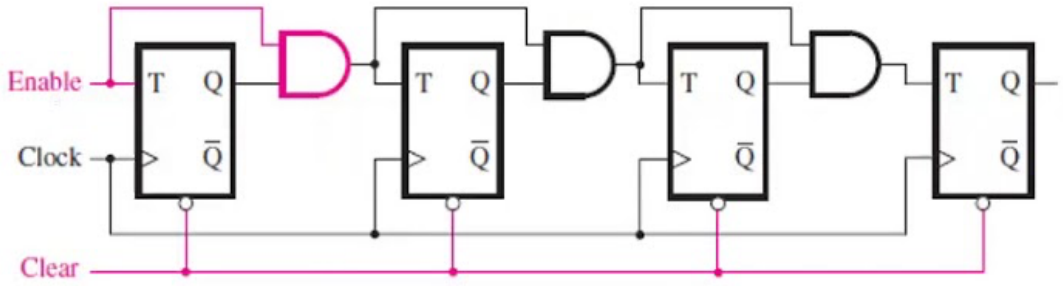
Les compteurs : Ripple counter

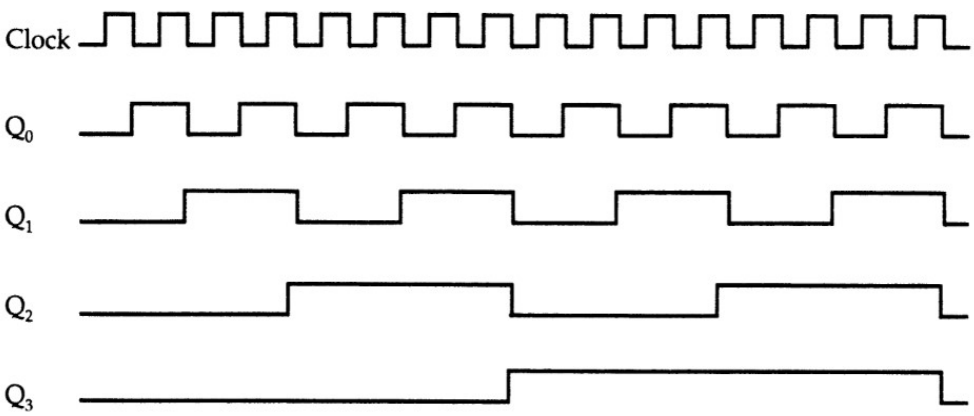
Compteur 4bit, base de D-flip-flops schéma RTL, «Ripple counter »





Les compteurs





Machines à états

TD 16:

Cahier des charges :

La machine à états doit être capable de contrôler l'ouverture et la fermeture d'une porte de garage.

Les entrées sont constituées d'un bouton poussoir pour l'ouverture (B0), un bouton poussoir pour la fermeture (B1), et un détecteur de fin de course pour indiquer la position de la porte. On notera S0 et S1 les signaux représentant respectivement « porte ouverte » et « porte fermée ».

Les sorties sont constituées d'un moteur pour actionner la porte, A0 et A1 pour respectivement ouvrir et fermer la porte

Le fonctionnement de la machine à états doit respecter les règles suivantes :

-) Si la porte est fermée et qu'on appuie sur le bouton d'ouverture, la porte doit s'ouvrir.

-) Si la porte est ouverte et qu'on appuie sur le bouton de fermeture, la porte doit se fermer.

-) Si la porte est en train de s'ouvrir et qu'on appuie sur le bouton de fermeture, la porte doit s'arrêter.

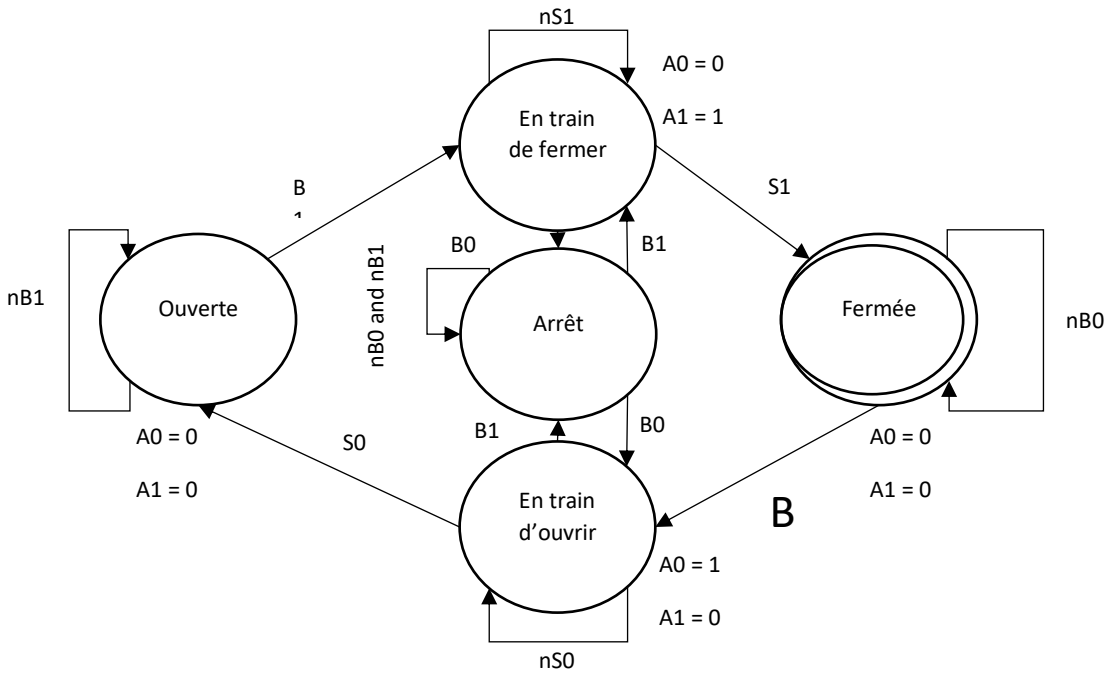
-) Si la porte est en train de se fermer et qu'on appuie sur le bouton d'ouverture, la porte doit s'arrêter.

Réponse :

Le nombre d'états nécessaires est de quatre : "fermée", "en train de s'ouvrir", "en train de se fermer" et "ouverte".

Les transitions entre les états sont les suivantes :

- De l'état "fermée" à l'état "en train de s'ouvrir" si le bouton d'ouverture est enfoncé et que la porte n'est pas déjà en train de s'ouvrir.
- De l'état "en train de s'ouvrir" à l'état "ouverte" lorsque le détecteur de fin de course indique que la porte est complètement ouverte.
- De l'état "ouverte" à l'état "en train de se fermer" si le bouton de fermeture est enfoncé et que la porte n'est pas déjà en train de se fermer.
- De l'état "en train de se fermer" à l'état "fermée" lorsque le détecteur de fin de course indique que la porte est complètement fermée.
- De l'état "fermée" à l'état "en train de se fermer" si le bouton de fermeture est enfoncé et que la porte est déjà en train de s'ouvrir.
- De l'état "ouverte" à l'état "en train de s'ouvrir" si le bouton d'ouverture est enfoncé et que la porte est déjà en train de se fermer.



TD 17 :

Consignes :

Considérons un système de verrouillage à code. Le système possède un pavé numérique composé de 10 boutons, chacun étiqueté de 0 à 9, pour entrer un code de verrouillage composé de 4 digits.

Lorsque l'utilisateur appuie sur un bouton deux signaux sont générés, un signal de validité ainsi qu'un signal sur 4bits correspondant au digit, si quatre chiffres ont été entrés, le système doit vérifier si le code est correct.

Si le code est correct, le système doit déverrouiller la porte pendant 10 secondes. La porte est par défaut verrouillée.

Note : vous disposez d'une horloge cadencée à 100Mhz

Définir

Les états

Les sorties et les entrées

Les parties opératives du système pilotant / générant les entrées si besoin

Les transitions entre état

Réponse :

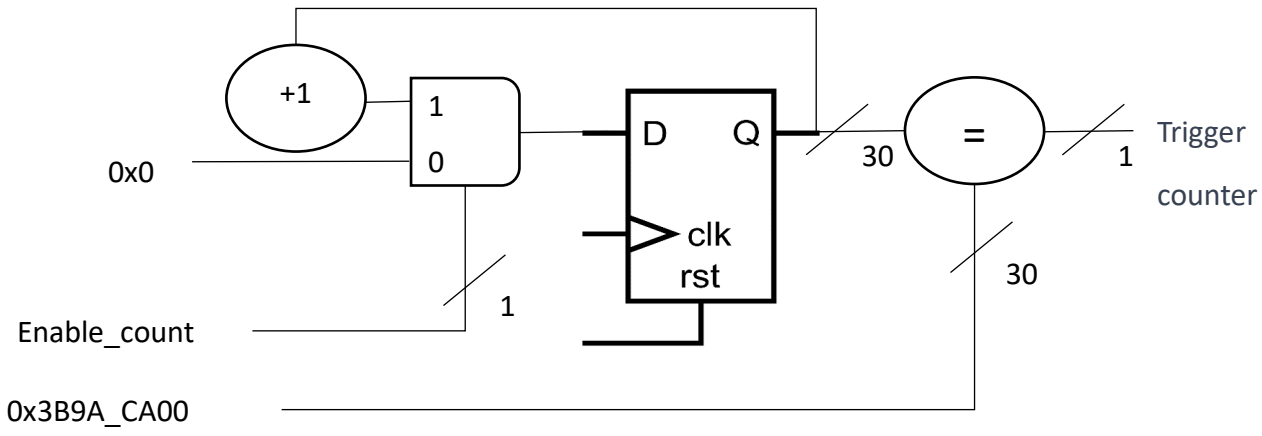
Un jeu d'état possible pour ce problème est

- Attente de la saisie du premier chiffre (état initial)
- Attente de la saisie du deuxième chiffre
- Attente de la saisie du troisième chiffre
- Attente de la saisie du quatrième chiffre
- Comparaison du code
- Ouverture porte
- Fermeture porte

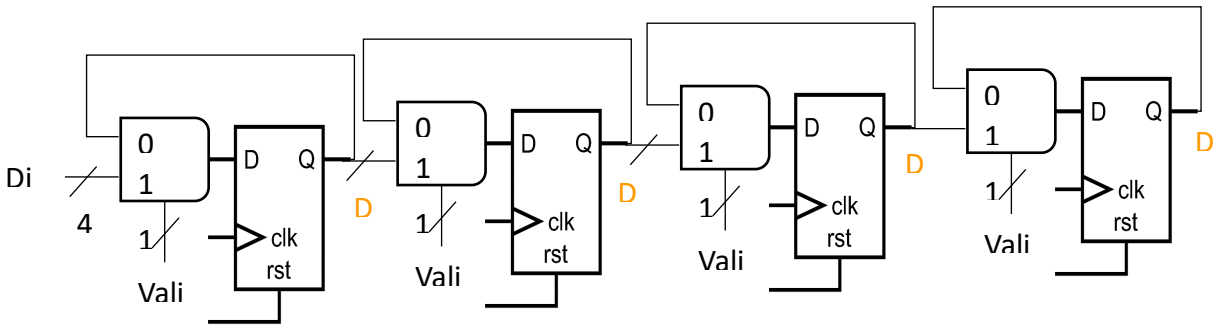
La Sortie du système est noté « O » et pilotera l'ouverture de la porte

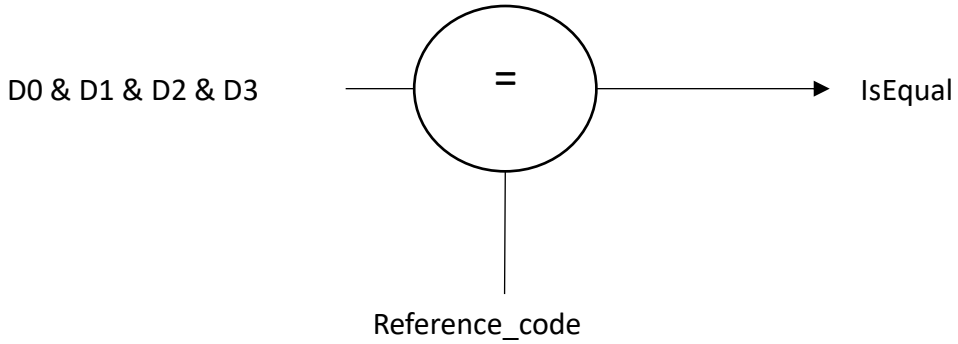
L'entrée « Valid » générée par le tableau de commande pour signifier le passage d'un état de saisie vers un autre.

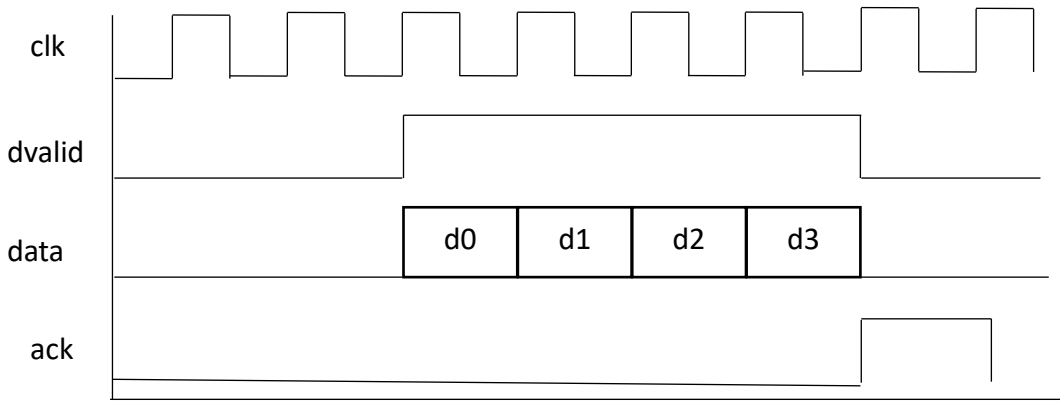
Un signal « Triggercounter » au moyen d'un compteur de temporisation sur 30 bits, ce dernier devra compter 1 000 000 000 cycles d'horloges (10 secondes d'attente); nous utiliserons un comparateur pour déterminer la fin de comptage.



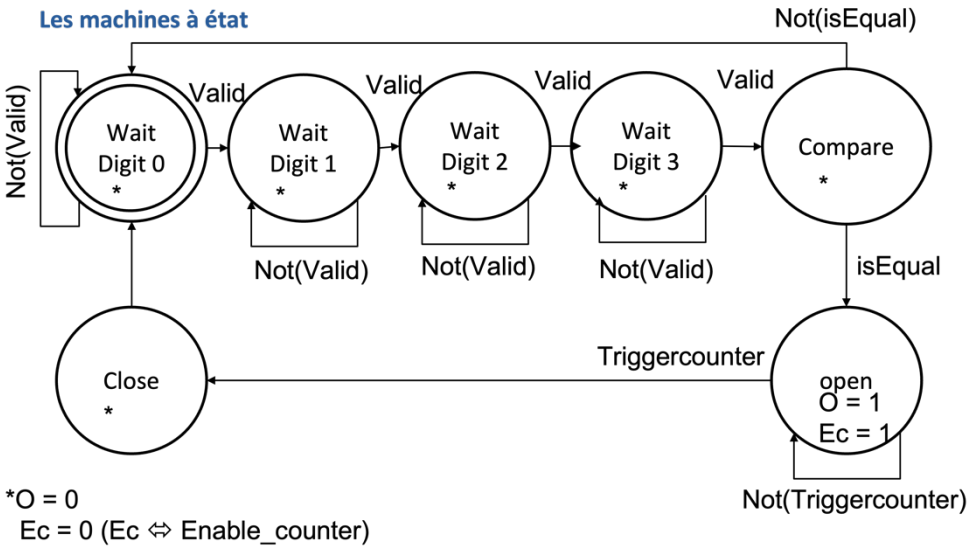
Un signal « IsEqual » pour vérifier que le code entré est bien celui demandé. Pour cela, nous utiliserons un registre à décalage pour stocker la valeur de la séquence de 4 digits. Le décalage aura lieu si et seulement si le signal « Valid » est actif







Les machines à état



*O = 0
Ec = 0 (Ec ⇔ Enable_counter)