TD1 : Logique Combinatoire

1 Exercices de base : manipulation de fonctions logiques

Q1 Montrez que l'opérateur "OU-EXCLUSIF" est le complément de l'opérateur "IDENTITE" (l'opérateur identité renvoie VRAI quand les opérandes sont égales).

Q2 Simplifiez de manière algébrique les expressions suivantes :

$$S_1 = a + a.b + \overline{a}.b.c + a.b.\overline{c}$$

$$S_2 = a.\overline{c} + b.\overline{c} + b.\overline{a}.c$$

$$S_3 = (a+b).\overline{c} + b.c.\overline{d} + \overline{a.(d+c)} + \overline{b+d)}$$

Tracez le logigramme de chacune des expressions précédentes.

Q3 Démontrez que l'opérateur "NON-OU" est une base des opérateurs (ou opérateur universel), c'est-à-dire qu'il est possible de construire toutes les opérations logiques à l'aide de cette unique porte logique.

Donnez les logigrammes des trois opérations NON, OU et ET à partir de portes NON-OU.

2 Transcodeur binaire → gray, gray → binaire

On désire réaliser un transcodeur, qui transforme le codage binaire en codage Gray pur d'un nombre compris entre 0 et 15.

Q4 Complétez le tableau ci-contre.

Q5 Déterminer les équations logiques des g_i en fonction des b_i.

Q6 Déterminer les équations logiques des g_i en fonction des b_i.

Q7 Construisez le logigramme du transcodeur binaire pur-> code Gray

Q8 Faites de même pour un transcodeur Gray-> binaire pur.

	b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

3 UAL 1 bit – 4 opérations

L'unité arithmétique et logique (UAL) constitue le module de calcul du micro-processeur. Il s'agit d'un circuit composé de portes logiques uniquement, auquel l'unité de commande du micro-processeur présente à chaque cycle de calcul les opérandes Y_1 et Y_2 des opérations à réaliser, ainsi que l'adresse

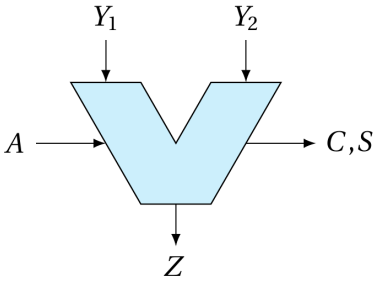


FIGURE 1 – Représentation de l'UAL à 1 bit et 4 opérations.

de l'opération, pour ensuite lire le résultat Z en sortie de l'UAL. Les UAL de petits micro-contrôleurs travaillent généralement sur des opérandes de 8 bits et disposent d'une centaine d'opérations.

L'objectif de l'exercice est de réaliser le logigramme d'une UAL à 1 bit et 4 opérations, c'est-à-dire capable de réaliser 4 opérations de type arithmétique (l'addition, la soustraction, la multiplication) ou de type logique (le ET logique, le OU logique) entre deux opérandes de 1 bit.

Le résultat Z, codé sur 1 bit, est complété par un registre d'état PSW comportant un bit C de retenue (*carry*) et un bit S de signe. La retenue vaut 1 si le résultat s'écrit sur 2 bits au lieu d'un, la retenue prenant alors la valeur du poids fort. Le bit de signe vaut 1 si le signe du résultat est négatif.

Q9 Montrer que pour une opération à 1 bit, la multiplication est équivalente au ET logique.

Q10 Déterminer le nombre n de bits d'adresse nécessaires pour adresser les opérations à réaliser.

Cette adresse sera notée A_i où i est le numéro du bit d'adresse (0 pour le poids faible et n pour le poids fort). L'ordre des opérations sera celui proposé ci-dessus.

Q11 Déterminer la liste des entrées et des sorties de l'UAL.

Q12 Déterminer la table de vérité de l'UAL.

Q13 Proposer le logigramme réalisant le calcul de Z dans l'UAL en fonction des opérandes et de l'adresse de l'opération à réaliser.

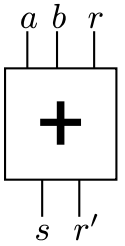
4 Additionneur 8 bits et additions sur les types int et long

L'addition de deux nombres entiers est une des opérations câblée dans les UAL. Selon la taille n des mots binaires traités par l'UAL (généralement 8 bits sur les petits microcontrôleurs et jusqu'à 64 bits pour les microprocesseurs d'ordinateurs), l'additionneur doit pouvoir traiter deux opérandes de n bits. Le résultats est un mot binaire pouvant nécessiter $n+1$ bits. Le $n+1$ième bit dépassant la taille des registres de travail, il est stocké dans un bit appelé "carry" (retenue) et noté C. Pour pouvoir traiter les nombres négatifs, le codage au format "complément à 2" ne nécessite aucune modification de l'addition, à l'exception du bit de signe qu'il faut distinguer de la carry. Pour simplifier, les nombres négatifs ne seront pas envisagés dans la suite et on s'intéressera à un additionneur 8 bits pour les entiers positifs.

Q14 Refaire une opération d'addition en binaire entre les nombres 0b10011111 et 0b01111010 pour se remémorer les opérations élémentaires sur chaque bit et le rôle de la retenue. Vérifier que l'addition de

2 nombres sur 8 bits peut renvoyer un nombre sur 9 bit mais qu'il est impossible d'aboutir à un nombre sur 10 bits.

Q15 On souhaite réaliser un module logique élémentaire, permettant d'additionner 2 bits a et a et une retenue r, et renvoyer en résultat le bit de somme s et la retenue r' à prévoir pour l'opération suivante. Établir la table de vérité du module logique élémentaire d'addition puis exprimer les fonctions logiques $S(A,B,R)$ et $R'(A,B,R)$ et tracer leurs logigrammes.



Q16 Assembler les modules logiques élémentaires d'addition pour réaliser un additionneur 8 bits admettant en entrée le mot binaire $A = 0b\,a_7a_6a_5a_4a_3a_2a_1a_0$ et $B = 0b\,b_7b_6b_5b_4b_3b_2b_1b_0$, et renvoyant en sortie le mot binaire $0b\,C\,s_7s_6s_5s_4s_3s_2s_1s_0$ où S est le mot binaire de 8 bits stocké dans le registre de résultat et C est le 9ème bit de "carrry".

Lorsque le module réalisé est cablé dans l'UAL, il permet de calculer l'addition du mot binaire en parallèle, en un seul cycle d'horloge. Il est aussi possible de réaliser l'opération en série, en 8 cycles d'horloge. La figure 2 montre le câblage du module logique élémentaire où les 8 bits de A et B sont successivements présentés en entrée, du bit de poids faible au bit de poids fort. On rappelle que le registre à décalage D recopie l'entrée sur la sortie au moment d'un front montant (et assure un décalage d'un cycle d'horloge entre r' et r).

Q17 Tracer le logigramme des sorties s et r' suite aux entrées CLK, a et b donnés sur le chronogramme.

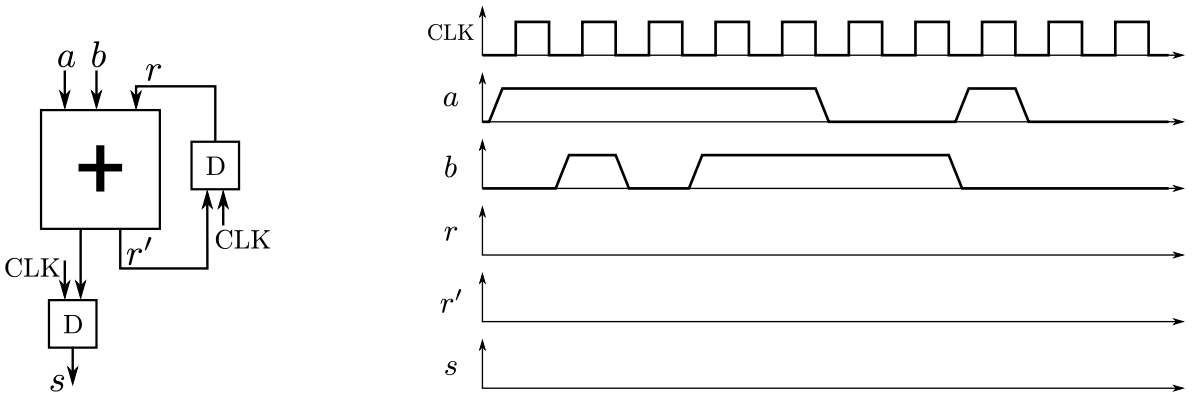


FIGURE 2 – Additionneur série

Le traitement sériel est évidement 8 fois plus long que le traitement parallèle, mais il n'est pas limité à 8 bits. En réalité, pour additionner sur un microcontrôleur 8 bits un entier int (16 bits) ou long (32 bits), les deux approches sont utilisées : traitement en série d'addition d'octets (chaques octets additionnés en paralllèle), la retenue étant reportée d'une addtion à la suivante. Il faut donc 2 cycles d'horloge pour une addition de type int et 4 pour le type long. À 16 MHz, il est possible de traiter 8 millions d'additions de type int par seconde.

5 Bascules RS, JK et D

Les bascules sont très utilisées dans les systèmes numériques, pour la réalisation de registres mémoire ou le transfert synchronisé des données. La bascule D, la plus utilisée, est réalisée à partir d'une bascule JK, elle même réalisée à partir de deux bascules RS. Le chronogramme du document réponse est à votre disposition pour les différents chronogrammes à tracer dans cet exercice.

5.1 Bascule RS Sychrone

La bascule dispose de 3 entrées R, S et CLK, et deux sorties Q_1 et Q_2. En fonctionnement normal, $Q_2 = \overline{Q}_1$. Le logigramme est indiqué figure 3.

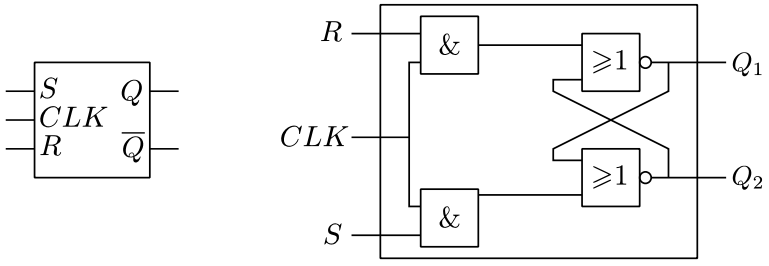


FIGURE 3 – Bascule RS

Q18 Proposer une table de vérité indiquant le comportement de la bascule RS en fonction des entrées, et vérifier que lorsque $R = S = 0$, l'état de la bascule s'exprime en fonction de $Q = Q_1$, et que $Q_2 = \overline{Q}_1$ dans toutes les situations sauf une que l'on précisera. Cette situation est évitée autant que possible dans les systèmes numériques.

Q19 Proposer un chronogramme représentant un signal d'horloge rectangulaire et représentant un fonctionnement normal de la bascule ou S passe à 1 puis revient à 0, puis R passe à 1 et revient à 0, en représentant l'état des sorties Q et $\overline{Q}$. Que se passe-t-il si S ou R change d'état pendant la phase où CLK est à 1 ?

5.2 Bascule JK

La bascule RS synchrone permet de mémoriser de l'information mais ne permet pas d'assurer le synchronisme au sens où elle est "transparente" durant la phase où l'horloge est à 1 : son état peut changer sans front d'horloge. La bascule JK résoud ce problème.

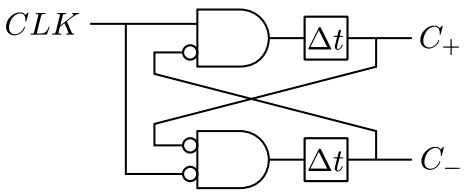


FIGURE 4 – Décalage d'horloge pour la bascule JK.

Q20 Étudier le schéma figure 4 où le bloc Δt est un bloc introduisant un petit retard de quelques nanosecondes de la sortie par rapport à l'entrée. Proposer un chronogramme montrant que C_+ et C_- sont des copies de CLK (à un petit retard près) assurant que les deux signaux ne sont jamais à 1 en même temps.

Q21 Étudier le schéma de la bascule JK figure 5 où les portes OUI et NON sont affectées d'un léger retard comme étudié à la question prédédentes. Vérifier que les deux bascules RS ne sont jamais transparentes en même temps, ce qui assure l'opacité de la bascule JK.

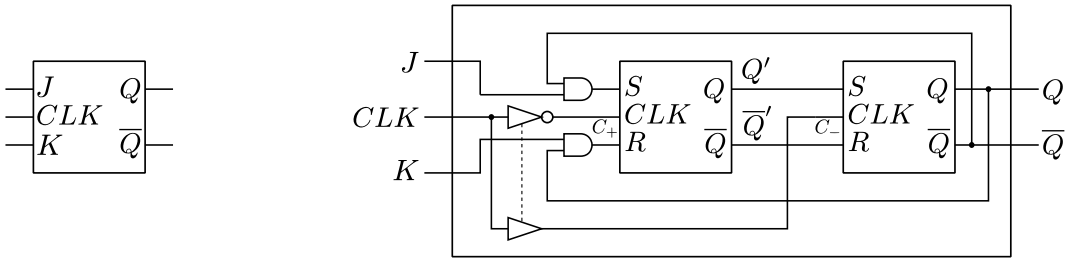


FIGURE 5 – Bascule JK

Remarque : la première bascule RS est dite "maitre" et la seconde est dite "esclave" au sens où elle recopie la sortie du maitre pendant que celui-ci est en phase bloquée.

Q22 Proposer un chronogramme montrant l'évolution de CLK, Q' et Q dans le cas d'un fonctionnement normal ou J passe à 1 puis revient à 0, puis K passe à 1 et revient à 0.

Q23 Proposer un chronogramme montrant l'évolution de CLK, Q' et Q dans le cas particulier où $J = K = 1$. Comment peut-on appeler le composant réalisé dans cette situation ?

Remarque : dans cette dernière situation, il est possible de réaliser des timers (compteurs) sur plusieurs bits facilement !

5.3 Bascule D

La bascule D permet de mémoriser un bit de donnée sur front montant de l'horloge uniquement. Il s'agit simplement d'une bascule JK où l'entrée K est le complément de J (figure 6).

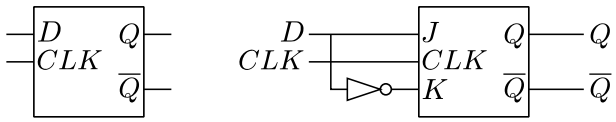
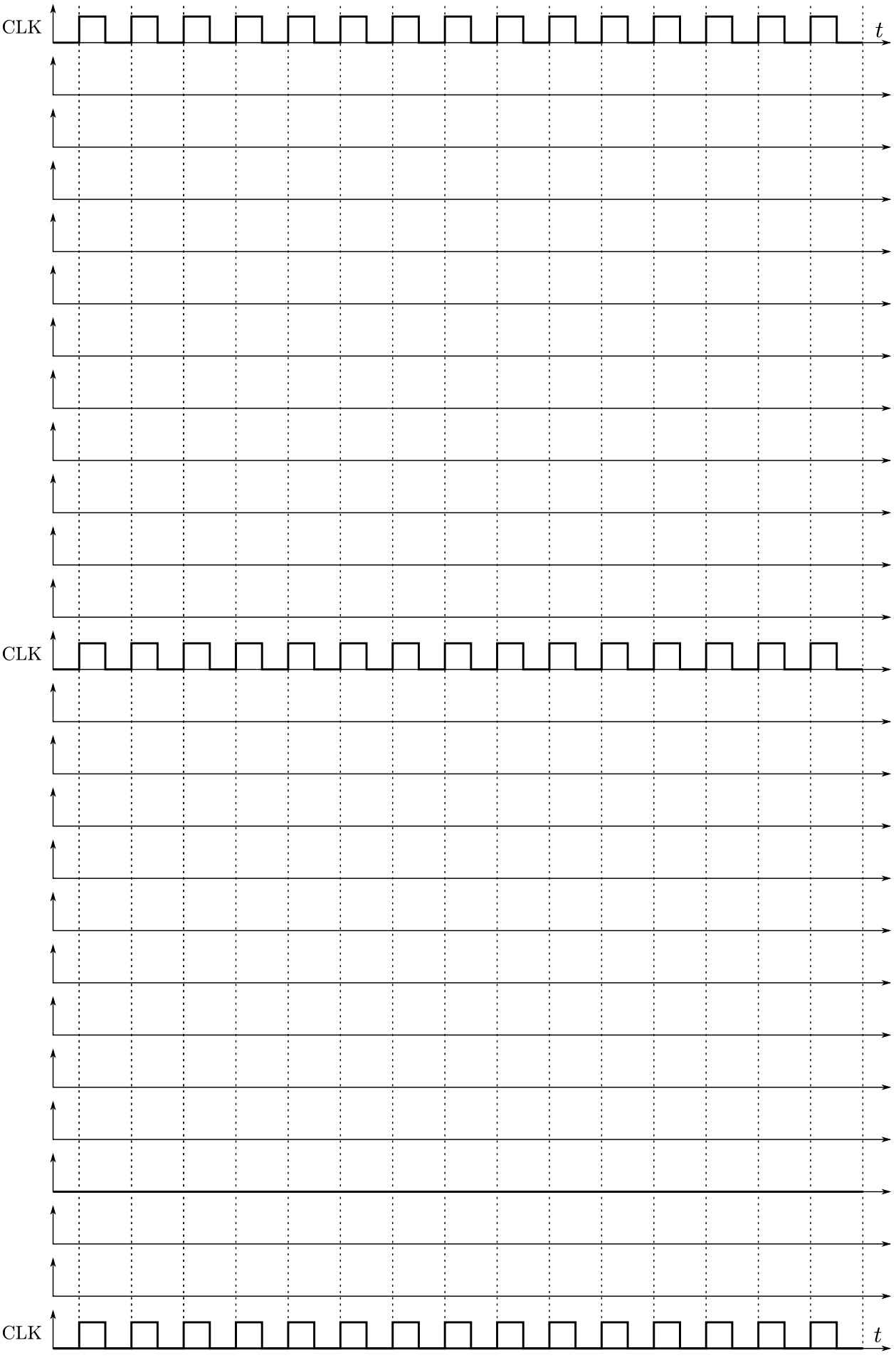


FIGURE 6 – Bascule D

Q24 Proposer un chronogramme montrant l'évolution de la sortie Q pour les entrées CLK et D en fonctionnement normal.

Q25 Proposer un logigramme comportant 3 bascules D en série, partageant le même signal d'horloge. Donner le logigramme des 3 sorties des bascules en fonctionnement normal où l'entrée de la première bascule passe à 1 puis revient à 0. Comment peut-on appeler le composant réalisé ?


CLK
t
CLK
CLK
t