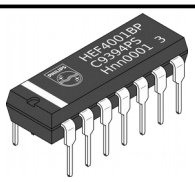
Structure Machine 2

CORRIGE de Série de TD1

https://elearning.univ-bejaia.dz/course/view.php?id=5349

Circuits logiques combinatoire (partie1)

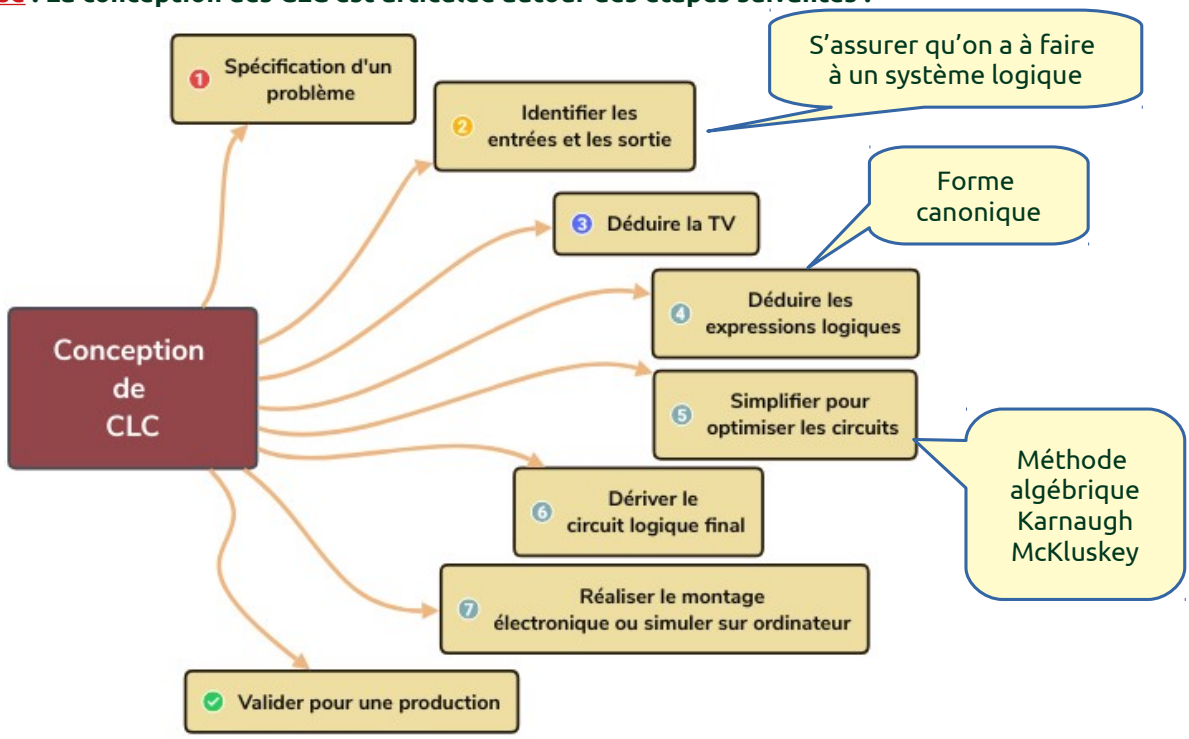


Séance de TD n°1

Objectif : Comprendre les fondements de la conception de circuits logiques combinatoires (CLC) et identifier les 2 grandes catégories de circuits logiques tout en expliquant le principe des circuits logiques combinatoires. Les étudiants, devraient aussi être capables d'énumérer les 3 classes de CLC et de citer quelques exemples de circuits dans chacune de ces classes. Enfin, il devraient être capables de faire l'analyse et la synthèse d'un CLC simple.

Q1 : Indiquez les étapes de conception des circuits logiques combinatoires (CLC) :

Réponse : La conception des CLC est articulée autour des étapes suivantes :



Q2 : Indiquez les 2 grandes catégories de circuits logiques existants

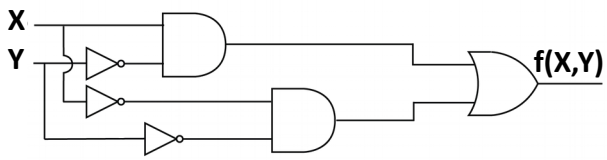
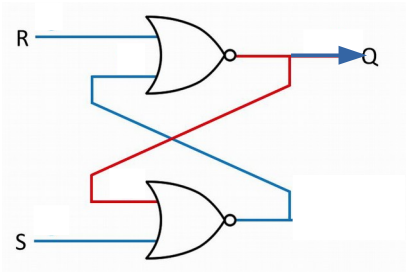
Réponse : Les circuits logiques peuvent être classés en circuits logiques combinatoires et circuits logiques séquentiels.

Q3 : Expliquer la différence entre « **analyse** » et « **synthèse** » d'un circuit logique

Réponse :

- **L'analyse d'un circuit logique consiste à démarrer d'un logigramme en établissant les équations booléennes de ses sorties puis de simplifier et d'essayer d'interpréter ce que fait exactement le circuit.**
- **La synthèse d'un circuit logique démarre d'un problème et a pour finalité de trouver sa solution en concevant un circuit logique. Pour cela une méthode est utilisée pour les CLC et une autre pour les circuits logiques séquentiels. Pour les CLC, voir la réponse à la question Q1 !**

Q4 : Indiquez si les circuits ci-dessous sont des CLC ? Justifiez votre réponse :

Circuit 1	Circuit 2
	 R et S sont les entrées Q est une sortie

Réponse :

- **Le circuit 1 représente une fonction qui dépend exclusivement des entrées (dites combinatoires) X et Y. Dans ce circuit la valeur de la sortie ne dépend pas de son état précédent (la sortie n'est pas mémorisé et n'est pas réinjectée en entrée!). Il s'agit donc d'un circuit logique combinatoire (CLC)**
- **Le circuit 1 représente une sortie (fonction) qui dépend des entrées R et S, mais aussi de cette même sortie qui est réinjectée en entrée. Ce circuit n'est certainement un circuit logique combinatoire. Nous verront au second chapitre du cours qu'il s'agit d'un circuit séquentiel et plus précisément une bascule (nous allons l'étudiant plus tard!).**

Q5 – Citez 3 classes circuits logiques combinatoires (CLC)

Réponse : En se référant à votre cours, vous découvrirez qu'on peut distinguer 3 types de CLC :

- **Les circuits de calcul arithmétiques et logiques,**
- **les circuits d'aiguillage et de transmission de données,**
- **les convertisseurs de codes.**

Q6 – Citez 3 exemples de CLC permettant la transmission de données

Réponse : En se référant à votre cours, vous découvrirez qu'on vous a donnée 4 exemple de circuits de transmission (ou d'aiguillage ou de sélection) de données : le multiplexeur (MUX), le démultiplexeur, le décodeur et l'encodeur.

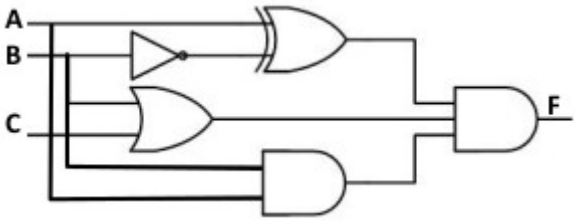
Q7 – Citez 3 exemples de CLC permettant le calcul arithmétique et logique

Réponse : Additionneur, Soustracteur, Comparateur.

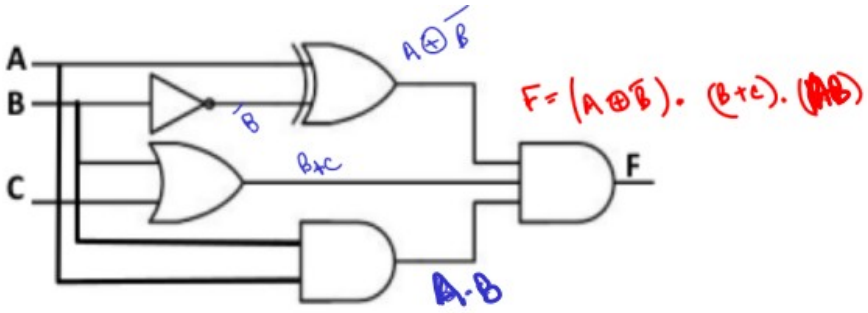
Q8 – Citez 3 exemples de CLC permettant la conversion de code

Réponse : Conversion binaire vers BCD, Afficheur 7 segments, Transcodeur

Q9 – Analyser le circuits suivant :



Réponse : On exprime les termes algébrique à la sortie de chaque porte logique en commençant des entrées jusqu'à arrivé à la sortie :



Nous obtenons l'équation de la sortie (de la fonction). Il faut la simplifier :

$$
\begin{aligned}
F &= (A \oplus \bar{B}) \cdot (B+C) \cdot (A \cdot B) \\
&= (A \oplus \bar{B}) \cdot AB\,(B+C) \\
&= (A \oplus \bar{B}) \cdot (AB \cdot B + ABC) \\
&= (A \oplus \bar{B}) \cdot (AB + ABC) \\
&= (\bar{A}\bar{B} + AB)\,(AB + ABC) \\
&= \bar{A}\bar{B} \cdot AB + \bar{A}\bar{B} \cdot ABC + AB \cdot AB + AB \cdot ABC \\
&= 0 + 0 + AB + ABC \\
&= AB(1+C) = AB
\end{aligned}
$$

Nous déduisons que notre circuit réalise une fonction « ET logiques » entre la « négation de A » avec les variables B et C.

Q10 – Faire la synthèse d'un circuit de détection de débordement. Je vous renvoi au cours du premier semestre pour vous revoir la problématique de débordement de capacité lors des calculs arithmétiques. Je rappel juste qu'il s'agit de dire si un débordement se produit à l'issue d'un calcul entre deux nombres A et B. Vous devez vous basez sur les signes des 2 nombres A et B et sur le signe du résultats R du calcul entre A et B. Je vous rappel aussi qu'il faut appliquer la méthode de conception de CLC (cerner le problème en identifiant les variables et les fonctions, établir la TV, extraire la FCD, simplifier puis dessiner le logigramme). A vous de jouer !

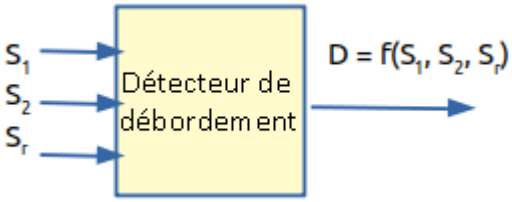
Réponse : A la lecture de l'énoncé de notre problème, on voit que nous avons une seule sorite (fonction) à déterminer. Il s'agit d'une fonction qui à « 1 », elle indique qu'il y a débordement de capacité (lors d'un calcul) et à « 0 », elle indique que le calcul n'a pas engendré de débordement de capacité. On voit bien qu'il s'agit d'une fonction booléenne. Appelons cette fonction *« D »*

Voyons de quoi dépend notre fonction D ?

En analysant notre énoncé, on voit bien que *D* dépend des signes des 2 nombres *A* et *B* impliqués dans un calcul et du signe du résultat de ce calcul.

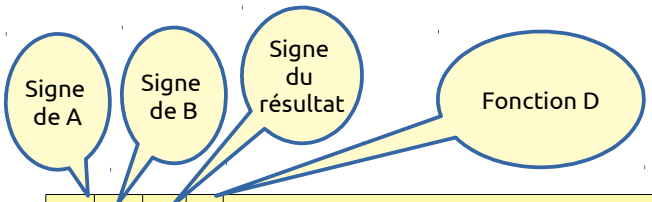
- **Appelons le signe du premier nombre S_1**
- **Appelons le signe du premier nombre S_2**
- **Appelons le signe du résultat S_r**

Nous pouvons, donc, écrire : *$D = f(S_1, S_2, S_r)$*



Nous sommes bien en présence d'un système logique : une fonction booléenne et des variable booléennes :

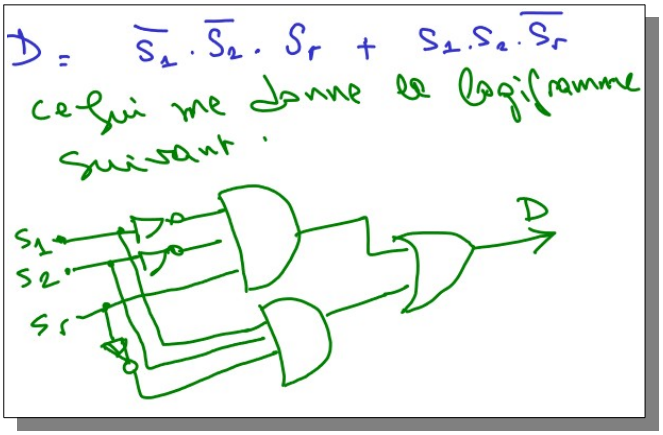
Trouvons, maintenant la relation qui existe entre la sortie D et les entrées (S_1, S_2, S_r). Nous ferons cela en dressant une table de vérité :



	S1	S2	Sr	D	Observation
m_0	0	0	0	0	Le signe de A et B sont positifs et le signe du résultat aussi ⇒ pas de débordement
m_1	0	0	1	1	**Le signe de A et B sont positifs et le signe du résultat négatif ⇒ débordement**
m_2	0	1	0	0	Le signe de A différent de celui de B ⇒ impossible d'avoir un débordement
m_3	0	1	1	0	Le signe de A différent de celui de B ⇒ impossible d'avoir un débordement
m_4	1	0	0	0	Le signe de A différent de celui de B ⇒ impossible d'avoir un débordement
m_5	1	0	1	0	Le signe de A différent de celui de B ⇒ impossible d'avoir un débordement
m_6	1	1	0	1	**Le signe de A et B sont négatifs et le signe du résultat positif⇒ débordement**
m_7	1	1	1	0	Le signe de A et B sont négatifs et le signe du résultat aussi ⇒ pas de débordement

A partir de la table de vérité ci-dessus, on trouve facilement l'équation algébrique (la forme canonique disjonction FCD) de notre sortie D : *$D = \sum(1, 6) = m_1+m_6$*

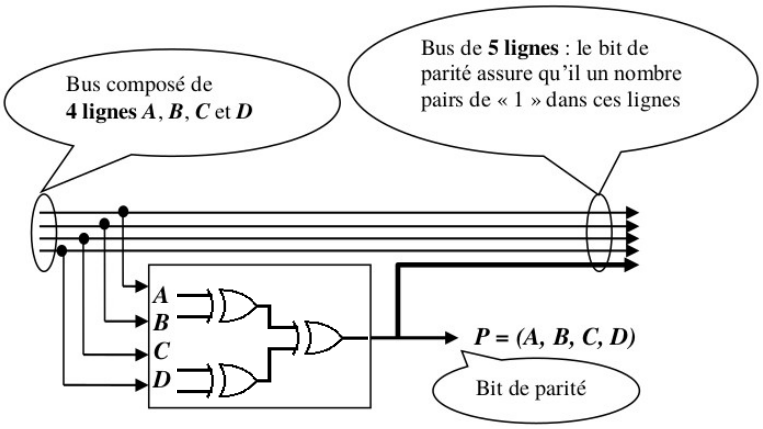
Voici l'équation et le logigramme de notre sortie D : →



Séance n°2

Objectif : A l'issue de cette séance de TD, les étudiants devraient être capables de faire la synthèse de quelques circuits logiques combinatoires en se basant sur une spécification d'un problème. De plus ils doivent être capables d'effectuer un montage en cascades de circuits logiques combinatoires (détecteur d'erreur).

Q12 : Faire la synthèse d'un circuits générateur de parité : On supposera qu'un générateur de parité est définit comme un circuit logique permettant d'assurer que le nombre de lignes à « 1 » dans un bus soit **toujours pair**. En supposant que vous avez un bus composé de 4 lignes (***A***, ***B***, ***C*** et ***D***), le générateur de parité ajoute à ce bus une 5ème ligne nommé « ***P*** » dont l'état logique est positionné de sorte à garantir un nombre pair « 1 » dans ce bus. Nous rappelons qu'un bus est un ensemble de fils qui transportent chacun un bit d'information.

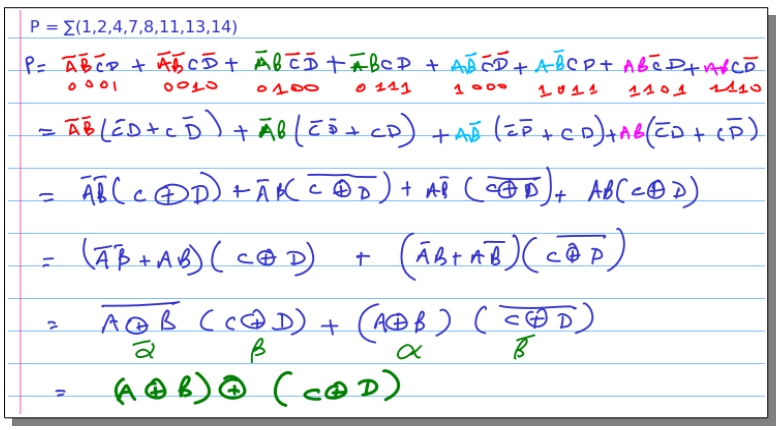


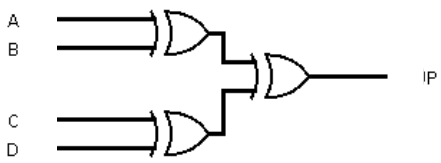
	ABCD	P
m_0	0000	0
m_1	0001	1
m_2	0010	1
m_3	0011	0
m_4	0100	1
m_5	0101	0
m_6	0110	0
m_7	0111	1
m_8	1000	1
m_9	1001	0
m_{10}	1010	0
m_{11}	1011	1
m_{12}	1100	0
m_{13}	1101	1
m_{14}	1110	1
m_{15}	1111	0

Réponse : Comme indiqué su le schéma ci-dessus, la fonction P dépend de 4 entrées A, B, C et D. Elle est à 1 si le nombre de « 1 » composée par les 4 variables A, B , C et D est impaire. Ceci garanti que le nombre total de « 1 » composé par les lignes A, B, C, D et P est pair !

Voici la table de vérité qui découle de ce fonctionnement : →

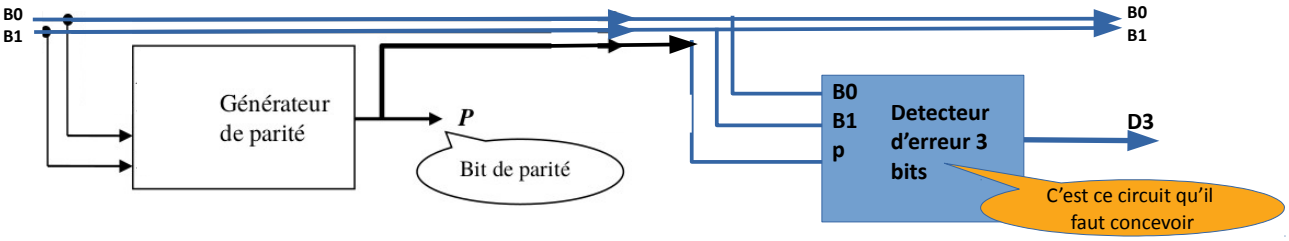
P = ∑(1,2,4,7,8,11,13,14)



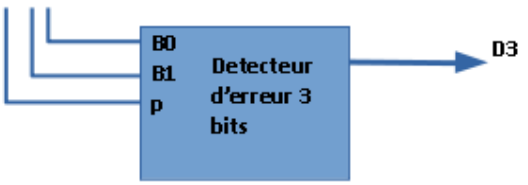


Q13 : Faire la synthèse d'un circuits de détection d'erreurs sur un bus de transmissions de données. On supposera que dans ce bus on a assuré, à l'émission, que le nombre de « 1 » est pair. Bien évidement si la transmission s'est déroulée correctement, le nombre de « 1 » dans le bus restera à « 1 » à la réception sauf si une erreur s'est produits lors de la communication. A vous concevoir ce détecteur d'erreur pour un bus de **3 bits** (2 bits de données et un bit de parité) puis pour un bus de **4 bits**. Enfin vous devez proposer un détecteur d'erreur pour un bus de **n+1 bits** (n bits pour les données et 1 bit de parité).

Réponse : Nous supposons que nous avons un bus avec *n bits* de données et 1 bits de parité. Nous avons vue dans la question précédente que le bit de parité est positionné de sorte à toujours avoir un nombre pairs de « 1 » dans le bus (de n+1 bits). A la réception des données (à l'autre bout du bus), s'il n'y a pas d'erreur de communication, il faut que le nombre de « 1 » dans le bus reste toujours pair. On se base sur cet fait pour vérifier qu'il n'y a pas justement des erreurs lors de la communication (transfert de données à travers le bus). Il s'agit donc de concevoir un circuit de détection d'erreur en se basant sur un générateur de parité. Voici le schéma global pour un bus à 3bits (2 bits de données et un bit de parité) :



On voit bien, dans ce problème, que la détection d'erreur dépend des *n+1* bits de notre bus. Pour n=3, nous nommons notre fonction de détection d'erreur *D3* (pour dire détection d'erreurs sur un bus à 3 bits) :

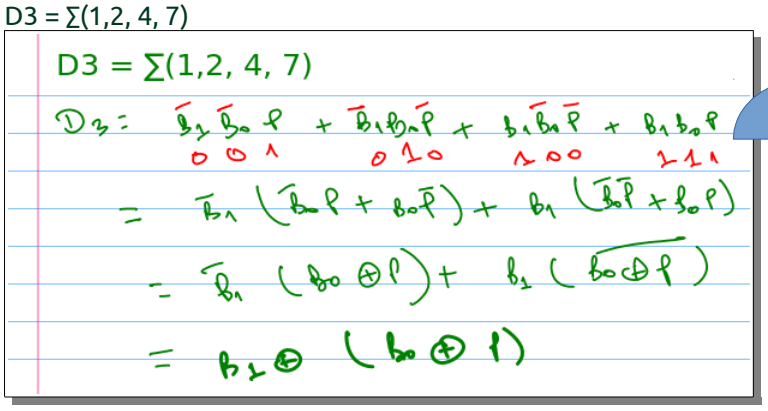


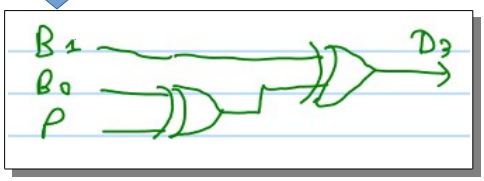
$D3 = f(B_1, B_0, P)$

Voici la table de vérité

	B1	B0	P	D3	Observation : On doit s'attendre à un nombre pairs de bits dans le bus !
m_0	0	0	0	0	
m_1	0	0	1	1	Ici nous avons un nombre impaire de « 1 » dans le bus, donc il y a erreur de transmission
m_2	0	1	0	1	Ici nous avons un nombre impaire de « 1 » dans le bus, donc il y a erreur de transmission
m_3	0	1	1	0	
m_4	1	0	0	1	Ici nous avons un nombre impaire de « 1 » dans le bus, donc il y a erreur de transmission
m_5	1	0	1	0	
m_6	1	1	0	0	
m_7	1	1	1	1	Ici nous avons un nombre impaire de « 1 » dans le bus, donc il y a erreur de transmission

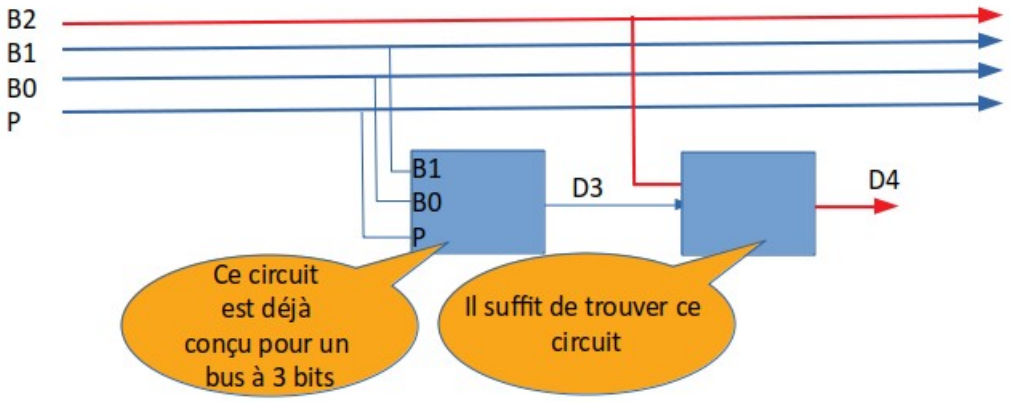
$D3 = \sum(1,2,4,7)$





Nous avons conçu un circuit de détection d'erreur pour un bus à 3 bits (B1, B0, P) , maintenant si on rajoute un quatrième bits, comment allons nous concevoir notre circuit ?

Nous allons profiter du fait que nous avons déjà trouvé un circuit de détection d'erreur (détection de parité) pour un bus à 3 bits (notre fameuse fonction *D3* !). Si l'on rajoute un 4ème bits, nommons notre nouvelle fonction *D4* pour dire détection d'erreur pour un bus à 4 bits (voir schéma suivant) :



Il s'agit donc de trouver la fonction D4 = f(B2, B1, B0, P) = g(B2, D3)
Il faut noter que D3 à « 1 » indique qu »il y a une erreur de transmission, donc un nombre impairs de « 1 » dans les 3 premiers bits (B1, B0 et P) du bus .

Voici la table de vérité

B2	D3	D4	Observation	
0	0	0	Ici D3 indique un nombre pair dans les 3 premiers bits (B1, B0 et P) sachant qu'ici B2 à 0 donc le nombre de « 1 » dans bus global reste pairs donc pas d'erreur !	pair + pair = pair
0	1	1	Ici D3 indique un nombre impair dans les 3 premiers bits (B1, B0 et P) Ici B2 à « 0 » donc le nombre de « 1 » dans bus global est impairs donc il y a une erreur !	impair + pair = impair
1	0	1	Ici D3 indique un nombre pair dans les 3 premiers bits (B1, B0 et P) Ici B2 à « 1 » donc le nombre de « 1 » dans bus global est impairs donc il y a une erreur !	pair + impair = impair
1	1	0	Ici D3 indique un nombre impair dans les 3 premiers bits (B1, B0 et P) et B2 à 1 donc on a, au final, un nombre pairs de « 1 » dans le bus, donc il y a une erreur !	impair + impair = pair

D4 = g(B2, D3) = ∑(1,2) = m1+m2 = $B2 \oplus D3$
Ce qui nous donne la fonction D4 de détection d'erreur pour un bus de 4 bits est un simple OU exclusif entre le bit de poids fort (le bit supplémentaire B2) et D3 qui est la fonction de détection d'erreur de niveau inférieur (pour un bus à 3 bits).

<u>Généralisation pour un bus de n+1 bits</u> :
On peut généraliser ce principe pour un bus à n+1 bits. Ainsi, si je suppose que j'ai déjà un circuit de détection pour les n premier bits (B_{n-1}, B_{n-2}, … B_0, P), si je rajoute un bits supplémentaire B_n. Je peux faire le même raisonnement que précédemment :
La fonction que je veux calculer est $Dn+1 = f(B_{n-1}, B_{n-2}, \dots B_0, P) = g(B_n, D_n)$. Nous aurons exactement la table de vérité ci-dessus. On déduit donc que $D_{n+1} = g(B_n, D_n) = B_n \oplus D_n$

voici au final le résultat :

$$D_3 = B_1 \oplus (B_0 \oplus P)$$
$$D_4 = B_3 \oplus D_3$$
$$D_5 = B_4 \oplus D_4$$
$$\dots \dots$$
$$D_n = B_{n-1} \oplus D_{n-1}$$
$$D_{n+1} = B_n \oplus D_n$$

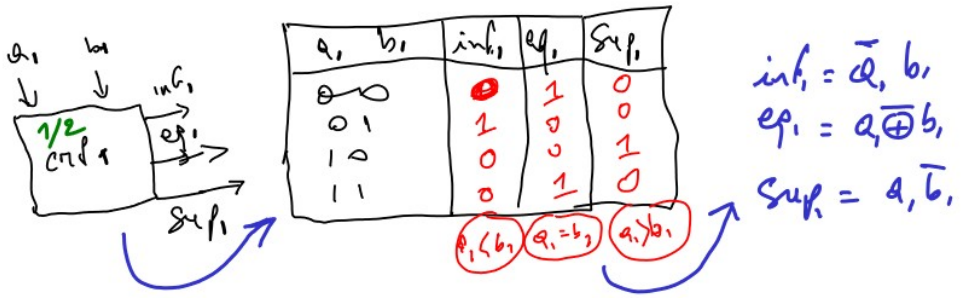
Objectifs des séances 3 et 4 : A l'issue des séances de TD 3 et 4, les étudiants devraient être capables de faire la synthèse de quelques circuits logiques combinatoires et d'effectuer des montages en cascades (comparateur, décodeur). Enfin, ils doivent aussi être capable de générer des fonction quelconques en utilisant des décodeur et des démultiplexeurs.

Séance n°3

Q14 : Dessiner un circuit additionneur de 2 mots A et B de 4 bits chacun (on met à votre disposition des 1/2 additionneur des des additionneurs complets)

Q15 : Faire la synthèse d'un comparateur entre **2 bits**. On vous indique qu'un comparateur 2 bits doit avoir deux entrées (b1 et b2) et 3 sorites : inf1, sup1 et equ1. Voir le schéma suivant :

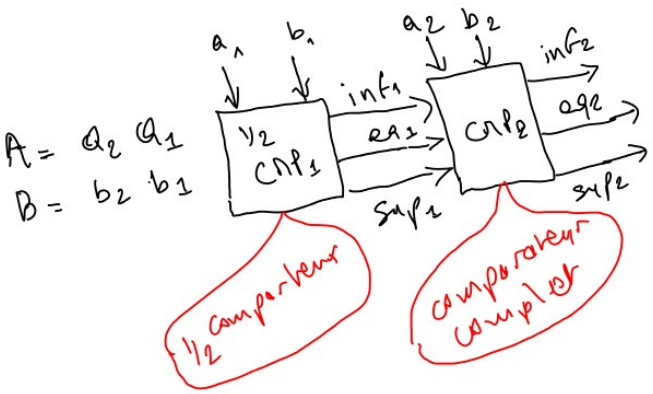
Réponse :



Q16 : Faire la synthèse d'un comparateur entre **4 bits** en vous basant sur un comparateur 2 bits de la question précédente

Réponse :

A = (a2, a1)
B = (b2, b1)



Pour que A soit inférieur à B, il suffit que le bit de poids fort de A soit inférieur à celui de B. Mais lorsque ces 2 bits sont égaux, il faut que le bit de niveau inférieur de A soit inférieur à celui de B. Ce qui donne :

- A<B ⇔ inf2 = (a2<b2) OU ((a2=b2) ET (a1<b1))
- en d'autres termes : A<B ⇔ inf2 = (a2<b2) OU ((a2=b2) ET (inf1))

D'où l'équation suivante :

$$inf_2 = \bar{a}_2 . b_2 + (a_2 \oplus b_2) . inf_1$$

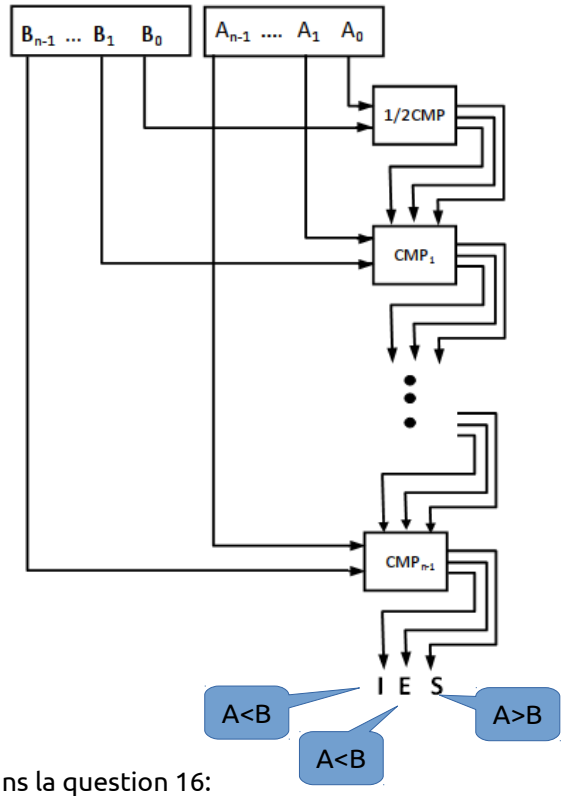
Pour que A soit égale à B, il faut que les bits de A soient respectivement égaux à ceux de B. Ce qui nous donne ceci :

- A=B ⇔ (a2=b2) ET (a1=b1)
- D'où l'équation suivante : $eq_2 = (a_2 \oplus b_2) . eq_1$

Pour que A soit supérieur à B il suffit que A ne soit pas inférieur à B et au même il faut qu'il ne soit pas supérieur à B. Ce qui nous donne :

- A>B ⇔ NON (A=B) ET NON(A<B)
- D'où l'équation suivante : $sup_2 = \overline{inf}_2 . \bar{eq}_2$

Q17 : Généralisation : Faire la synthèse d'un comparateur entre n bits en vous basant sur un comparateur 2 bits de la question précédente



Réponse : En supposant que je veux réaliser un comparateur qui compare 2 mots A et B sur n bits chacun. Voici le schéma global de ce circuit qui est basé sur 2 circuits de base : un demi comparateur noté (1/2CMP) et un comparateur complet (note CMPi) qui prend en compte les « i » bits précédents des 2 mots A et B. En fait, on va procéder comme on avait fait avec l'additionneur. Voici le schéma global de notre comparateur :

Nous avons déjà conçu le circuit 1/2CMP. C'est la réponse à la question 15.

La question 16 (Comparateur 4 bits ou le nombre de bits de A est 2 et celui de B est 2) nous avions trouvé les équations suivantes :

$$inf_2 = \bar{a}_2 . b_2 + (a_2 \oplus b_2) . inf_1$$
$$eq_2 = (a_2 \oplus b_2) . eq_1$$
$$sup_2 = \bar{inf}_2 . \bar{eq}_2$$

On peut se baser sur le même raisonnement que celui utilisé dans la question 16:

Supposons que :

- A = (a_n , a_{n-1}, … a_2, a_1)
- B = (b_n , b_{n-1}, … b_2, b_1)

Pour que A soit inférieur à B, il suffit que le bit de poids fort de A soit inférieur à celui de B. Mais lorsque ces 2 bits sont égaux, il faut que les nombre formé des bits de niveau inférieur de A soit inférieur à celui composé des bis bits de niveau inférieur de B.
ce qui donne : A<B ⇔ inf_n = (a_n<b_n) OU ((a_n=b_n) ET inf_{n-1})

ce qui donne :

$$inf_n = \bar{a}_n . b_n + (a_n \oplus b_n) . inf_{n-1}$$

Pour que A soit égale à B, il faut que les bits de A soient respectivement égaux à ceux de B. Ce qui nous donne ceci :

$$eq_n = (a_n \oplus b_n) . eq_{n-1}$$

Pour que A soit supérieur à B il suffit que A ne soit pas inférieur à B et au même il faut qu'il ne soit pas supérieur à B. Ce qui nous donne :

$$sup_n = \bar{inf}_n . \bar{eq}_n$$

Séance n°4

Q18 : Faire la synthèse d'un décodeur 3⇒8 à base de décodeur 2⇒4

Q19 : Générer la fonction F(x,y,z)=∑(1,5,6) à base de DEC et DeMUX

Q20 : Faire l'analyse du circuit suivant :