I. Examen 7 Janvier : Correction

Exercice I.1

Soit $f(x_1, ..., x_4)$ La fonction booléenne suivante : f donne vrai si au max une des entrées est vraie.

– Donnez la table de vérité.

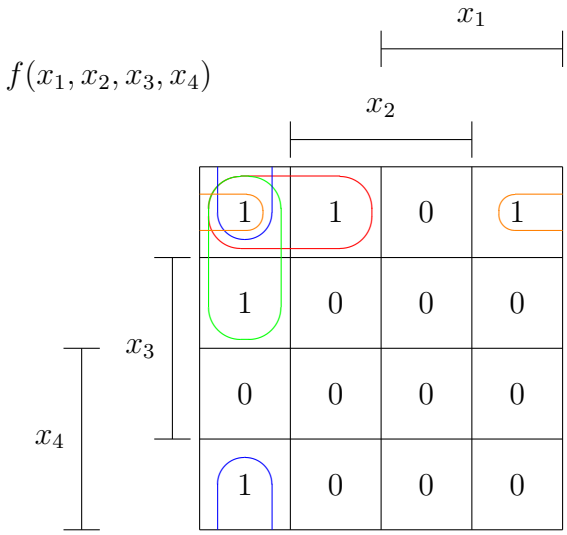
Correction

x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0

x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

– Donnez une représentation en table de Karnaugh. Indiquez une couverture de la table pour la forme normale disjonctive.

Correction



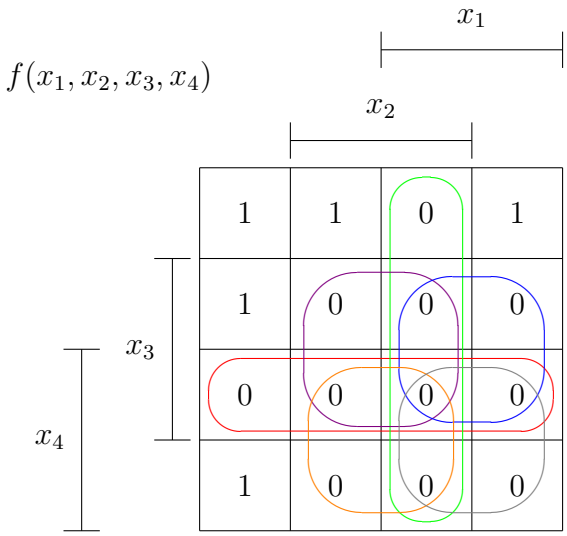
– Donnez les formes normales conjonctives disjonctives et conjonctives.

Correction

La forme disjonctive :

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1}.\overline{x_2}.\overline{x_4}) + (\overline{x_1}.\overline{x_3}.\overline{x_4}) + (\overline{x_1}.\overline{x_2}.\overline{x_3}) + (\overline{x_2}.\overline{x_3}.\overline{x_4})$$

Pour la forme conjonctive il suffit de reprendre la table de Karnaugh et d'entourer les '0'.



$$f(x_1, x_2, x_3, x_4) = \overline{(x_1 \cdot x_2) + (x_3 \cdot x_4) + (x_1 \cdot x_3) + (x_1 \cdot x_4) + (x_2 \cdot x_4) + (x_2 \cdot x_4)}$$
$$= (\overline{x_1} + \overline{x_2}) \cdot (\overline{x_3} + \overline{x_4}) \cdot (\overline{x_1} + \overline{x_3}) \cdot (\overline{x_1} + \overline{x_4}) \cdot (\overline{x_2} + \overline{x_4}) \cdot (\overline{x_2} + \overline{x_4})$$

– Généraliser pour $f_n(x_1, ..., x_n)$ qui serait vraie si au max une des entrées est vraie.

Correction

Pour la forme normal disjonctive on remarque qu'on prend tout les termes sauf un dans chaque groupe de conjonction. On peut en déduire la formule suivante :

$$f(x_1, ..., x_n) = (\overline{x_2} \cdot \overline{x_3} \cdot ... \cdot \overline{x_n}) + (\overline{x_1} \cdot \overline{x_3} \cdot ... \cdot \overline{x_n}) + ... + (\overline{x_1} \cdot ... \cdot \overline{x_{n-1}})$$

Car il faut au moins que tout les autres soient à '0' sauf un. Pour la forme conjonctive nous remarquons qu'on de prendre toutes les combinaisons par deux des complémentaires.

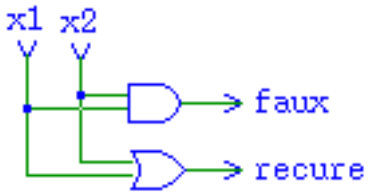
$$f(x_1, ..., x_n) = (\overline{x_1} + \overline{x_2}) \cdot (\overline{x_1} + \overline{x_3}) \cdot ... \cdot (\overline{x_1} + \overline{x_n}) \cdot (\overline{x_2} + \overline{x_3}) \cdot ... \cdot (\overline{x_{n-1}} + \overline{x_n})$$

Car il ne peut y en avoir deux égaux à '1' ce qui mettrait un groupe disjonction à '0'.

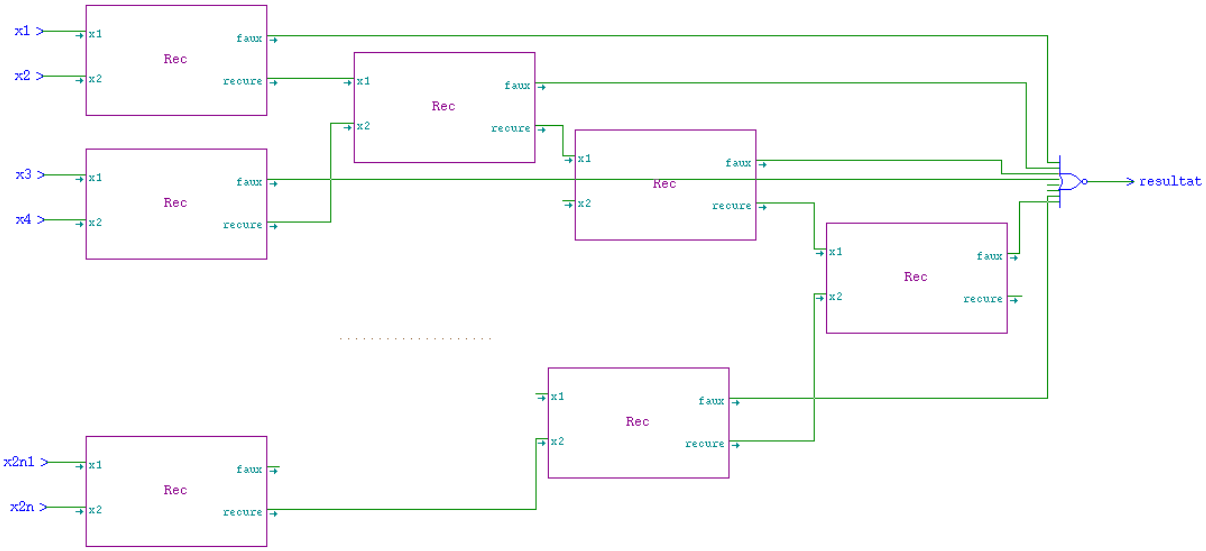
– Bonus 1 : Donner un circuit combinatoire pour f_{2^n} qui n'a qu'un nombre linéaire (en 2^n) de portes. Indication : construire d'une façon récursive des circuit C_n avec deux sorties, une pour f_{2^n} et une pour $g_{2^n}(x_1, ..., x_n)$ qui est vraie ssi toutes les entrées sont fausses.

Correction

Tout d'abord f_{2^n} veux dire qu'on va découper par paquets de deux la fonction



réalisant la même chose mais sur deux fois moins d'élément. Je propose une fonction élémentaire qui prends deux bits et dit si les deux sont à 1 ou l'un des deux ou aucun. C'est ce que réalise ce module : Faux ⇒ que deux sont à "1" et recure ⇒ l'un des deux est à un. Ainsi on si on a aucun faux cela veux dire qu'on a rencontrer au maximum un bit à "1". On les câble donc comme dessous :



– Bonus 2 : Si vous avez trouvé le circuit pour le Bonus 1, qu'elle est la taille de l'expression booléenne qui correspond au circuit ?

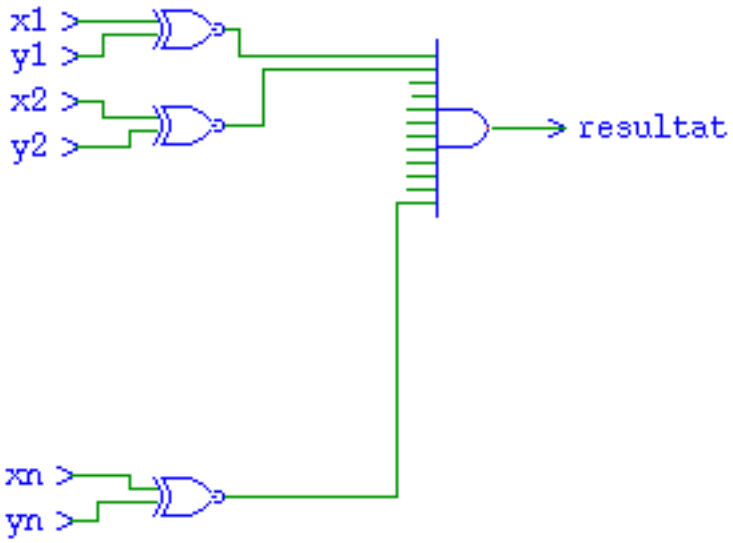
Correction

Ce qui nous intéresse c'est de faire une opération OR sur tout les "faux". Nous savons qu'il y a $log_2(n)$ niveaux. A chaque niveau nous utilisons tout les littéraux. Par exemple les fils partant du premier niveau donneront : x_1 et x_2 **ou** x_3 et x_4 ... Ceux du deuxième niveau donneront : [(x_1 ou x_2) et (x_3 ou x_4)] **ou** [(x_5 ou x_6) et (x_7 ou x_8)] ... .
On obtient donc une taille de : $nlog_2(n)$.

Exercice I.2 Dessiner un circuit pour $eq(x_1, ..., x_n, y_1, ..., y_n)$ qui est vraie ssi les deux vecteurs $(x_1, ..., x_n)$ et $(y_1, ..., y_n)$ sont égaux.

Correction

Ici je propose d'utiliser une porte $nXOR$ afin de vérifier que chaque paire x_i, y_i sont égaux. En finissant avec une grosse porte AND.



Qu'elle est la taille de la forme normale disjonctive ?

Correction

Ici on vérifie toutes les combinaisons qui donne un 1. C'est à dire ($x_1 = 1$ et $y_1 = 1$) ou ($x_1 = 0$ et $y_1 = 0$) C'est à dire qu'il y a 2^n combinaisons possible. donc 2^n conjonctions contenant $2n$ éléments ce qui nous fait $n2^{n+1}$ littéraux.

Même question pour la forme normale conjonctive.

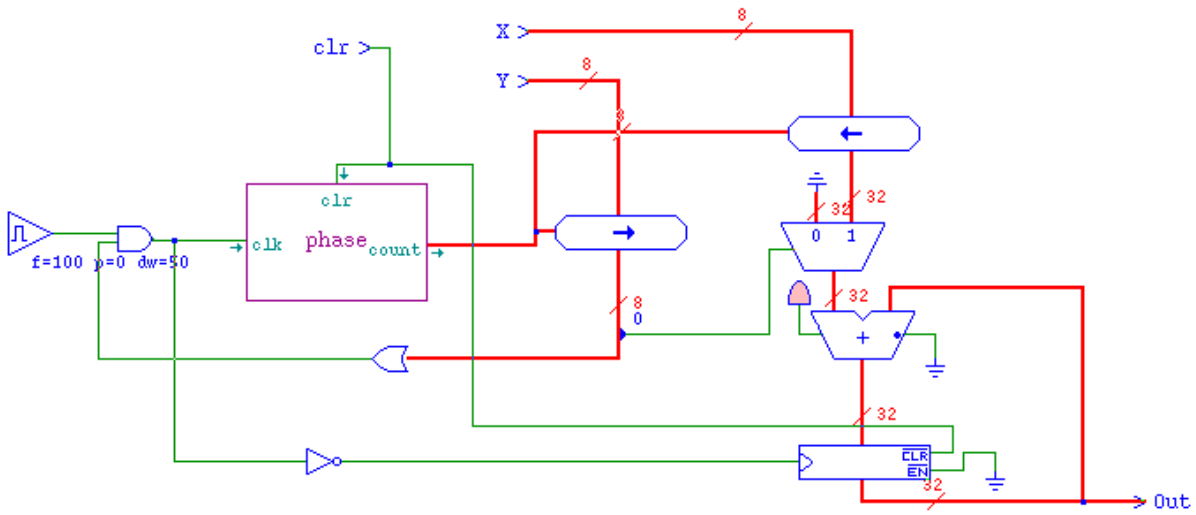
Correction

Pour la forme normale conjonctive on vérifie uniquement que les couples x_i, y_i soit égaux donc on a des couples de type : $(\overline{x_i} + y_i).(x_i + \overline{y_i})$. Donc la taille est de $2n$ littéraux.

Exercice I.3 Expliquez la multiplication « par addition et décalage ». Indiquez la forme d'un circuit synchrone pour la multiplication des deux nombre de n bits qui utilise l'addition (circuit combinatoire), le décalage, un registre et un compteur de phases.

Correction

Le principe de la multiplication décalage est de faire la somme des produit de chaque chiffres de Y avec X en décalant. La formule est : $\sum_{i=1}^{n} X \times y_i \times 2^i$.



Dans ce circuit X et Y sont nos deux nombres. Le compteur de phase commence à 0 et s'incrémente au front montant. Le registre mémorise en front descendant. On commence avec un clk à '1'.

- Si le premier bit de Y est à 1 on mets X u registre sinon on mets 0.
- On descend on mémorise la valeur précédente 0 puis la valeur calculé juste avant selon le premier bit.
- On monte, le compteur de phase passe à 1 ce qui décale d'un bit à droite le nombre X (multiplie par deux) et décale à gauche Y (divise par 2)

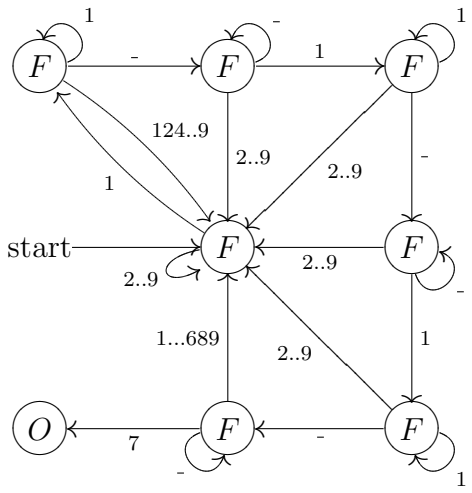
Et recommence : on teste le bit 0 de Y décalée et etc. A la fin quand la valeur Y est nulle la porte AND à droite stop les cycle d'horloge.

Exercice I.4 Un circuit pour un verrou électronique reçoit d'un clavier des chiffres $0, ..., 9$ (si une touche est appuyée) et « _ » (pas de touche appuyée), il ya pour sortie « fermé » et « ouvert ». Dessiner la machine de Moore pour la verrou qui ouvre sur la combinaison de chiffres $3, 1, 1, 7$. Pas de circuit demandé, juste la machine.

Correction

Pour cet exercice il suffit de faire un chemin de la séquence que l'utilisateur devra faire pour avoir l'état ouvert. Chaque déviation du chemin a pour effet de retourner à l'étape initiale. Ici la séquence est "3_1_1_7_". On peut dire que quand il appuis sur le dernier 7 de la séquence on est dans l'état acceptant.

Voici l'automate réalisant cela :



Exercice I.5 Réaliser en assembler PIC16 les fonctions (appelées avec « Call ») qui prennent leur argument dans W et qui rendre le résultat aussi dans W

- Une fonction qui renvoie le nombre de '1' dans la représentation binaire de l'argument (par exemple, pour l'argument 5, le résultat est 2 pour l'argument 11, le résultat est 3, etc.).
- Une fonction qui prend (dans W) l'adresse d'un registre et qui renvoie le nombre de '1' dans la représentation binaire de ce registre.
- Une fonction qui multiplie l'argument par 2, une autre qui multiplie l'argument par 3.

S'il faut des registres auxiliaires pour réaliser ces fonction, indiquer le, mais utiliser les avec des noms.

Correction

Pour ce code on déclare les registres number et counter.

```
compte  MOVWF   number      ; On met le nombre dans le registre number
        CLRF    counter     ; On met le compteur à 0
        CLRW                ; On met W à 0 et il le restera jusqu'à la fin de la boucle
boucle  BCF     STATUS, C   ; On met C à 0 afin d'éviter l'injection de 1 dans le nombre
        BTFSC   number, 0   ; On test le bit 0 du nombre.
                            S'il est à 0 on passe l'instruction suivante
        INCF    counter, f  ; S'il est à 1 on incrémente le compteur
        RRF     number, f   ; On décale les bits de number vers la droite.
        SUBWF   number, f   ; On teste si le nombre est null ici W est toujours null
        BTFSS   STATUS, Z   ; S'il est null on saute le goto
        GOTO    boucle      ; On revient à boucle
        MOVF    counter, f  ; On charge le compteur dans W
        RETURN              ; On fini la fonction
```

```
compteindr  MOVWF   FSR       ; On met l'adresse dans FSR
            MOVF    INDF,W    ; On récupère la valeur pointée
            CALL    compte    ; On appelle la fonction précédente
            RETURN            ; On fini la fonction.
```

Ce code fonctionne si et seulement si l'utilisateur a réglé IRP dans $STATUS$ correctement.

Ici on déclare le registre tmp.

```
mult2   MOVWF   tmp       ; On met le nombre dans tmp
        ADDWF   tmp,W     ; Ajoute une fois tmp à W
        RETURN            ; On a fini
mult3   MOVWF   tmp       ; On met le nombre dans tmp
        ADDWF   tmp,W     ; Ajoute une fois tmp à W
        ADDWF   tmp,W     ; Ajoute une fois tmp à W
        RETURN            ; On a fini
```

Est-ce qu'il y a des exception au fonctionnement de vos fonctions ?