Université Joseph Fourier – Grenoble 1
ALM MIAGE L3
Examen première session
Décembre 2013
Durée : 3h00 en tout (~1h00 Archi, ~1h Système, ~1h Réseau).
1 document A4 commun ASR autorisé. Sujet Archi sur 1 recto-verso.

I. Arbres de Wallace (barème indicatif sur 7 points : 3 points)

Q1. Rappeler le schéma de la cellule élémentaire du circuit d'addition FA : ($Add_{1\ bit}$: A, B, C_{in} → C_{out}, S)

Ce circuit (FA) peut être vu comme un compteur à trois entrées (E_0 E_0' E_0'') qui donne sur 2 bits en sortie (S_1 S_0) le nombre d'entrées à 1 (S_0 : bit de poids 0 du résultat, S_1 : bit de poids 1 du résultat).

Il s'agit d'un arbre de Wallace élémentaire à 3 entrées, que nous noterons W_3 dans la suite.

Plus généralement, les arbres de Wallace sont des compteurs de complexité logarithmique (pour le temps de calcul en fonction du nombre d'entrées) construit récursivement à partir d'arbres de Wallace de taille plus petite : à partir de 2 arbres de Wallace à $n = 2^P-1$ entrées, p sorties et de p FA ($=W_3$) on peut construire un arbre de Wallace deux fois plus gros (à $2^{P+1}-1$ entrées). L'idée générale repose sur une décomposition par dichotomie : pour compter n entrées, on divise les entrées en 2 groupes égaux, on compte chaque groupe avec un arbre de Wallace et on additionne les deux résultats. Remarque : l'addition finale se fait avec des FA (c'est-à-dire avec des arbres de Wallace élémentaires W_3). L'ensemble constitue un arbre de Wallace.

Dans le détail, les $2^{P+1}-1$ entrées d'un arbre de Wallace $W2^{P+1}-1$ sont réparties en 2 sous-groupes de 2^P-1 entrées ; pour avoir le bon nombre d'entrées, on ajoute 1 entrée supplémentaire (E_{sup}). Récursivement, le nombre de bits à 1 de chaque sous-groupe est comptabilisé avec un arbre de Wallace $W2^p-1$. Les deux résultats s'expriment sur p bits, ils sont additionnés avec E_{sup} qui a le rôle de retenue entrante de l'additionneur final : i.e. Les bits de poids faibles des résultat des deux sous-groupes (S_0 et S_0') sont combinées avec l'entrée supplémentaire E_{sup} à l'aide d'un arbre W_3 pour donner le début du résultat (le bit de poids faible) ; le bit de poids 1 de ce premier W_3 est combiné avec les bits de poids 1 des résultats des deux arbres $W2^p-1$ avec un second arbre W_3 pour donner le bit de poids 1du résultat et ainsi de suite on cascade les W_3 jusqu'aux derniers bits de poids forts des deux groupes.

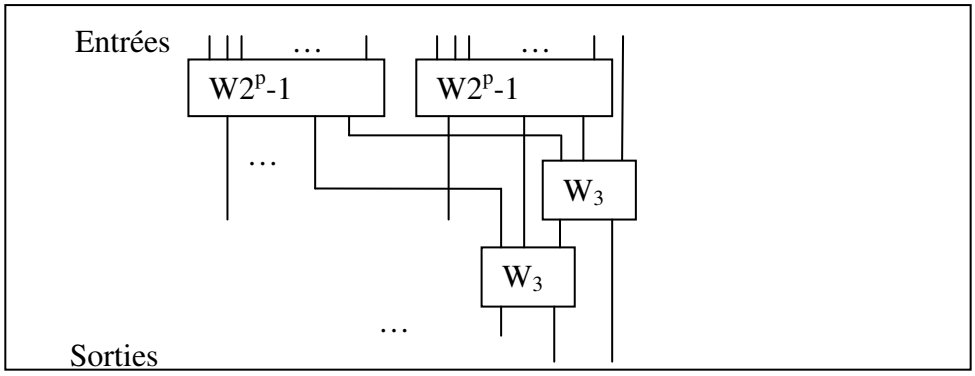


Fig : Construction de $W2^{p+1}-1$ à partir de $W2^p-1$ et de W_3.

Les arbres de Wallace peuvent être utilisés en particulier dans les multiplieurs naïfs sur n bits, pour exécuter les n additions à effectuer en phase finale après les produits élémentaires (par 0/1) et les décalages effectués dans une première phase.

Q2. Déterminer comment faire W_7 à partir de 2 W_3 (pour faire un premier décompte sur deux sous-groupes de 3 bits pris parmi les 7 bits d'entrées) et d'autres W_3 pour comptabiliser le dernier bit d'entrée (E_{sup}) et combiner les deux résultats précédents sous la forme d'un additionneur. Déterminer précisément la complexité en temps de calcul de chaque sortie de l'arbre de Wallace W_7.

Q3. Généraliser, en regardant éventuellement la construction de W15 à partir de W7 et la complexité du circuit obtenu, pour obtenir la complexité en temps de calcul d'un arbre de Wallace $W2^p-1$.

II. Calcul par la méthode de Newton-Raphson (barème indicatif sur 7 points : 1.5 points)

Q1. Proposer un circuit réalisant l'algorithme suivant de calcul de la racine carrée par la méthode de Héron (une exemple d'utilisation de la méthode de Newton-Raphson). Vous supposerez que des circuits arithmétiques d'addition, de soustraction, de multiplication et de division sont disponibles.

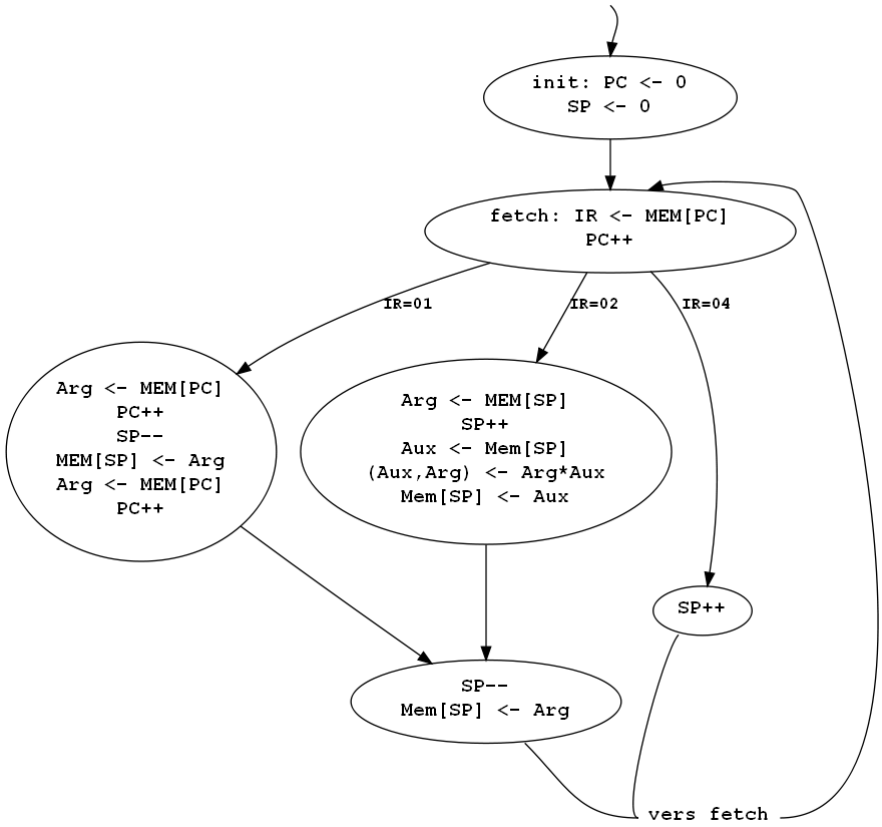
Init : A←entrée ; X←1 ;
Tant que abs(X*X-A)<1 **faire** : X ← (X + A/X) / 2 ; **fin-tant-que**
Fin : **Si** X*X>A **alors** retourne X-1 **sinon** retourne X.

III. Automate de contrôle, multiplication et division (barème indicatif sur 7 points : 2.5 points)

Une machine 8 bits à pile (pointeur de pile **SP** sur l'élément en haut de pile, empiler décroit **SP**) est donnée par un automate d'interprétation dont une partie est donnée ci-après. Cette machine a la particularité de pouvoir faire des multiplications d'entiers sur 8 bits avec un résultat non tronqué sur 16 bits (8 bits de résultat de poids fort dans le registre **Aux**, 8 bits de résultat de poids faible dans le registre **Arg**).

Q1. Simuler l'exécution du programme donné en mémoire ci-après, en utilisant l'automate décrit en vis-à-vis.

Automate de contrôle :



Mémoire :

Adresse	Contenu (en hexadécimal)
00	01
01	14
02	3C
03	02
04	04
05	00
…	…
FE	00
FF	00

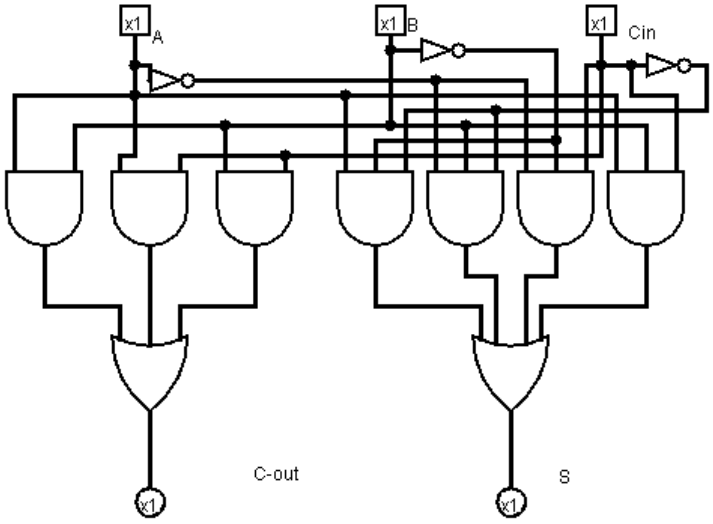
Le programme ci-dessus a peut-être fourni en haut de pile le résultat de la division 60 / 13 (en hexadécimal, 3C / 0D) et pourtant le programme ne comporte par la valeur 13 (0D). La méthode utilisée consiste à remplacer 60 / 13 par (60*256) / (13*256), en le transformant en 60 * (256 / 13) * (1/256), c'est-à-dire à effectuer une multiplication de 60 par 20 (arrondi par défaut de 256 / 13), puis à faire un décalage d'un octet. (Poids Fort, Poids faible) → (00, Poids Fort).

Q2. Transformer le graphe de l'automate pour introduire une instruction permettant d'effectuer cette suite d'opérations en une seule instruction **MultShift**. Dans le détail, **MultShift** est une instruction avec un paramètre enregistré dans l'octet suivant l'instruction, **MultShift Arg** doit réaliser la suite d'opération suivante : dépiler le haut de pile dans **Aux**, multiplier **Arg** et **Aux**, empiler l'octet de poids fort du résultat de la multiplication (l'octet de poids faible du résultat de la multiplication n'est pas utile).

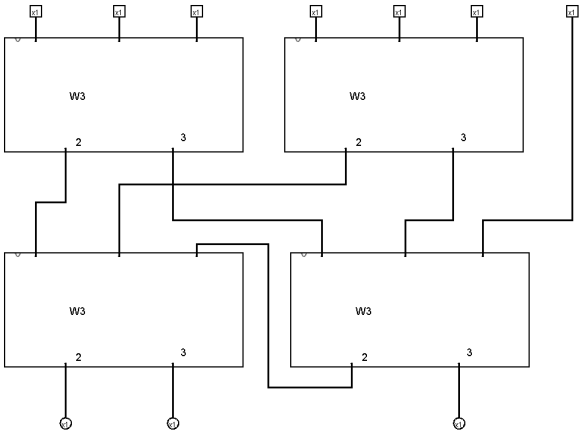
Éléments de solution.

Exo 1.

Q1. Full Adder (W3)

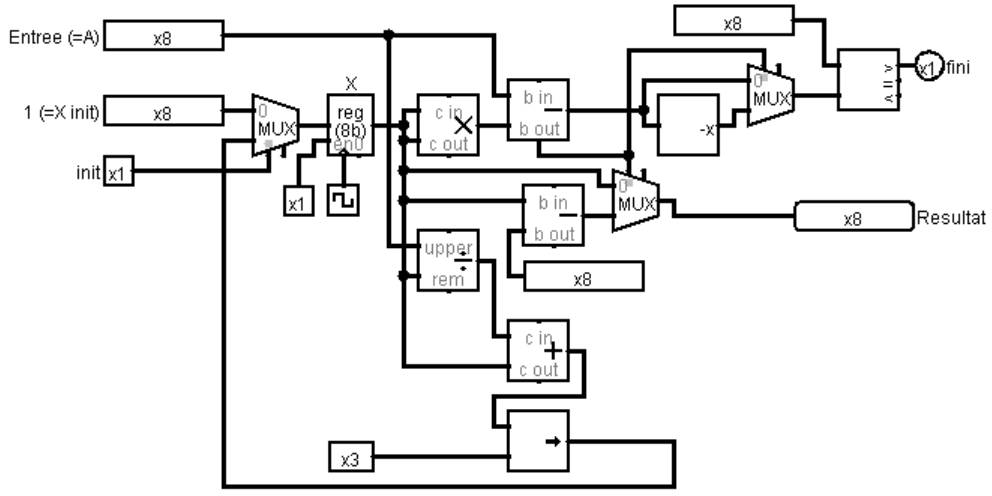


Q2. W7 avec les complexité 7 8 6 (dans cet ordre) pour les 3 sorties. En haut les deux comptages récursifs, en bas l'additionneur.



Q3. Compléxité en général : à chaque niveau supplémentaire, il y a une cellule W3 en plus en hauteur sur le chemin de la première sortie (et sur chaque chemin), et une cellule en plus en largeur pour l'additionneur final. Donc pour passer de $W2^p$-1 à $W2^{p+1}$-1 il faut ajouter 5 délais. W3 : 3 délais, W7 : 8 délais, W15 : 13 délais max, …Complexité général en temps de calcul proportionnelle à P pour $W2^p$-1.

Exo 2. Circuit pour des entiers (mais algo plutôt prévu pour des réels), à part l'entrée et le résultat, les constantes de ce circuits valent 1.



Exo 3.
Q1. Avec le nommage des états ci-dessous, on a l'exécution :
1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; 10 ; 3 ; 4 ; 11 ; 12 ; 13 ; 14 (la multiplication : 04 . B0 ←14 * 3C) ; 15 ; 16 ; 17 ; 3 ; 4 ; 18 fin avec sur la pile la valeur 04 (SP=FF ; MEM[FF]=04).

Q2. Une solution qui factorise la récupération de l'argument (5 ; 6) et la sauvegarde du résultat (16 ; 17). Rem. : on pouvait factoriser d'autres états, mais pour un graphe moins simple.