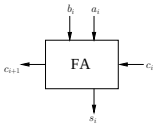
Arithmétique des ordinateurs – TD 01: Addition

{ christoph.lauter, matthieu.gallet } @ens-lyon.fr

30 janvier et 4 février 2008

1 Propagation de la retenue - retenue conditionnelle



1. Rappelez le fonctionnement d'un full-adder binaire.

2. Implémentez un full-adder binaire à base de portes NAND.

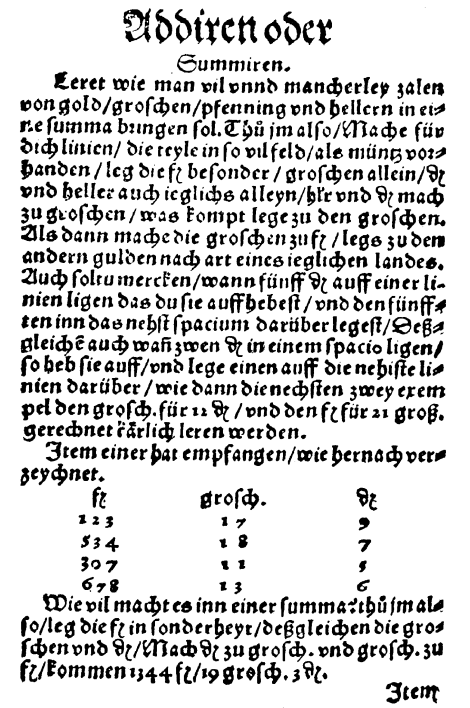
3. Construisez un additionneur 8 bits à base de full-adders. Évaluez le nombre de portes NAND nécessaires.

4. Formez une rangée de 8 personnes au tableau. Chacun d'entre vous jouera le rôle d'un full-adder décimal. Additionnez 52671772 et 12535497. A quel problème vous voyez-vous confrontés ?

5. Démontrez ou refutez l'assertion suivante : quelle que soit la base β d'un système de numération à position, la retenue d'une position à la suivante est d'au plus 1.

6. Coupez votre rangée en deux parties. Recrutez encore 4 personnes pour une deuxième demi-rangée. Additionnez $52671772 = 5267 \cdot 10^4 + 1772$ et $12535497 = 1253 \cdot 10^4 + 5497$ en supposant que la retenue de la position 3 à la position 4 est soit 0 soit 1.

7. Construisez un additionneur 8 bits qui fonctionne en temps logarithmique. Evaluez le nombre de portes NAND nécessaires.



Addiren oder
Summiren.

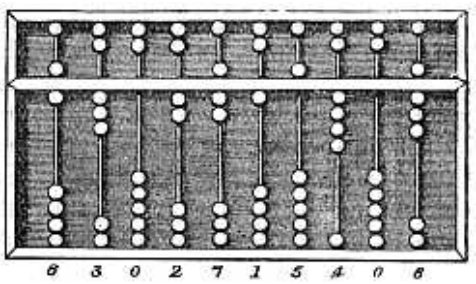
Leret wie man vil vnnd mancherley zalen von gold/groſchen/pfenning vnd hellern in eine ſumma bringen ſol. Thů jm alſo/Mache für dich linien/ die reyle in ſo vil feld/als münz vorhanden / leg die ß beſonder / groſchen allein/ß vnd heller auch ieglichs alleyn/hlr vnd ₰ mach zu groſchen / was kompt lege zu den groſchen. Als dann mache die groſchen zu ß / legs zu den andern gulden nach art eines ieglichen landes. Auch ſoltu mercken/wann fünff ₰ auff einer linien ligen das du ſie auffhebeſt / vnd den fünfften inn das nehſt ſpacium darüber legeſt/Deßgleichē auch wañ zwen ₰ in einem ſpacio ligen/ ſo heb ſie auff/vnd lege einen auff die nehiſte linien darüber / wie dann die nechſten zwey exempel den groſch. für 12 ₰ / vnd den ß für 21 groß. gerechnet ẽrlich leren werden.

Item einer hat empfangen/wie hernach verzeychnet.

ß	groſch.	₰
123	17	9
534	18	7
307	11	5
678	13	6

Wie vil macht es inn einer ſumma: thů jm alſo/leg die ß in ſonderheyt/deßgleichen die groſchen vnd ₰/Mach ₰ zu groſch. vnd groſch. zu ß/kommen 1344 ß/19 groſch. 3 ₰.

Item



2 Addition carry-save

8. Examinez le boulier représenté. Expliquez pourquoi la position des boules correspond effectivement au nombre 6302715478.

9. A quoi les boules tout en haut servent-elles ? Pourtant aucune n'est descendue !

10. Dessinez un boulier de ce type à 8 positions et posez l'addition de 52671772 et 12535497. Essayez de procéder de la façon la plus parallèle possible.

11. Rappelez ce que vous avez appris en cours pour la représentation carry-save ! En vous servant du full-adder, dessinez un additionneur de deux nombres binaires avec une sortie carry-save ! Quel est le temps de calcul de votre additionneur ?

12. Tant que vous y êtes, dessinez un additionneur de deux nombres en représentation carry-save avec une sortie en carry-save qui ait la même complexité ! Il paraît qu'on puisse faire avec seulement 2 full-adders par position.

13. Donnez un circuit qui prend un nombre en représentation carry-save et qui le transforme en représentation binaire « normale » ! Quelle est sa complexité en temps ?

14. Imaginez qu'on rajoute encore des boules à valeur 5 à notre boulier pour en avoir k. Combien de nombres en notation à position pouvez-vous additionner maintenant avant de devoir faire une propagation de retenue ?

15. Certaines bibliothèques de calcul sur des grands nombres (GMP, SCSLib) sont basées sur le carry-save logiciel, c'est-à-dire sur le retardement des propagation de retenue. Dans ces bibliothèques, de grands nombres x sont représentés comme des tableaux de nombres x_i de la façon suivante :

$$x = \sum_{i=0}^{n} x_i \cdot 2^{b \cdot i}$$

où b est un entier plus petit que la largeur l des mots machine. Généralement, deux opérations sont donc implémentées : *l'addition carry-save* qui ne propage pas les retenues et *la propagation de la retenue* explicite. Donnez du pseudo-code pour chacune de ces opérations !

16. Pour quel type d'utilisation une telle bibliothèque carry-save est plus rapide qu'un bibliothèque de grands nombres implémentée de façon naïve ? Pensez par exemple au produit scalaire ou bien à la multiplication elle-même.

17. Pourquoi le parallélisme sert-il aussi dans une implémentation logicielle ?

3 Addition carry-skip

Étudions maintenant les principes de génération et de propagation de la retenue. Considérons deux nombres $A = (a_{n-1}a_{n-2}\dots a_0)$ et $B = (b_{n-1}b_{n-2}\dots b_0)$. On constate que :

- si $a_i = b_i = 0$, alors on a toujours $c_{i+1} = 0$,
- si $a_i = b_i = 1$, alors on a toujours $c_{i+1} = 1$,
- si $a_i \neq bi$, alors on a toujours $c_{i+1} = c_i$.

18. Imaginez un additionneur utilisant ces remarques pour accélérer l'ensemble du processus. Quel serait l'intérêt d'un tel additionneur ? Quels en seraient ses défauts ?