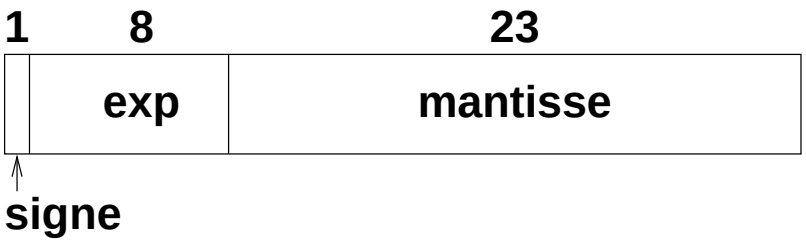
Représentation en virgule flottante (IEEE 754)

- C'est une façon de représenter un sous-ensemble des rationnels
- Le nombre est divisé en mantisse et exposant, les deux de tailles fixes
- La norme IEEE 754 définit 3 formats dont deux externes et un interne
- Les formats externes sont : simple précision (32 bits) et double précision (64 bits)
- Le format interne est utilisé à l'intérieur du processeur de calcul. La précision de ce format est de 80 bits (format étendu)
- La multiplication étant plus fréquente que l'addition, on n'utilise pas la représentation de complément à 2, mais signe+valeur absolue pour la mantisse
- Pour l'exposant on utilise une représentation similaire au complément à 2 appelée excédent 127 (simple précision) et excédent 1023 (double précision)

IEEE 754

Simple précision :



Double précision :

Le nombre représenté est $s \times m \times 2^e$

Représentation normalisée de la mantisse (IEEE 754)

- La valeur représentée est toujours supérieure à zéro
- De plus, la valeur est toujours normalisée ($1 \leq v < 2$). C'est toujours possible, car on peut toujours ajuster l'exposant
- La valeur représentée s'écrit donc : 1,...
- Puisque le premier chiffre est toujours 1, on ne le représente pas (il est implicite dans la représentation)
- En simple précision, la représentation 00000000000000000000000 indique donc la valeur 1, la représentation 10000000000000000000000 indique la valeur 1,5, etc.

Représentation normalisée de l'exposant (IEEE 754)

- L'opération la plus fréquente est l'addition/soustraction,
- mais on n'utilise pas le complément à 2
- Pour la simple précision, on utilise une représentation "excédent 127"

- Dans cette représentation la représentation est le nombre (positif) obtenu en additionnant 127 à l'exposant.
- De plus, les représentations 00000000 et 11111111 sont particulières
- La représentation 11111110 indique un exposant de 254 - 127 = 127
- La représentation 00000001 indique donc un exposant de 1 - 127 = -126
- Le plus petit nombre représentable est donc $1 * 2^{-126}$
- Le plus grand nombre représentable est $1,11111111111111111111111 * 2^{127}$
- La double précision est similaire

Représentations non normalisées

Il y a quatre représentations non normalisées :

- Représentation dénormalisée,
- Représentation de zéro,
- Représentation de l'infini,
- NaN (Not a Number)

Représentation dénormalisée

- Avec cette représentation le champ pour l'exposant est 00000000 (simple précision) ce qui signifie un exposant de -127
- La valeur de la mantisse est 0 < v < 1 avec tous les bits représentés
- Le plus grand nombre dénormalisé est donc $0.11111111111111111111111 * 2^{-127}$ soit presque 2^{-127}
- Le plus petit nombre dénormalisé est $0,00000000000000000000001 * 2^{-127}$ soit 2^{-150}

Représentation de zéro

- La valeur zéro est représentée par un champ exposant de 00000000 et un champ mantisse de 00000000000000000000000
- C'est l'extension logique de la représentation dénormalisée
- Il y a deux représentations pour zéro, dont une positive et une négative selon la valeur du champ signe

Représentation de l'infini

- Le champ exposant est 11111111 et le champ mantisse est 00000000000000000000000
- Cette valeur est générée par des opérations arithmétiques dont le résultat dépasse ce qui est représentable de façon normalisée
- Cette valeur est aussi acceptable en tant qu'opérande des opérations arithmétiques

NaN, Not a Number

Certaines opérations, par exemple la division de l'infini par l'infini, donnent un résultat indéfini, dont la représentation contient le champ exposant 11111111 et n'importe quelle configuration de bits sauf 00000000000000000000000 du champ mantisse

Arithmétique à virgule flottante

- Pour l'addition et la soustraction, il faut d'abord ajuster l'exposant du plus petit des deux opérandes pour que les deux exposants soit identiques
- Il suffit alors de décaler la mantisse du plus petit nombre à droite (right shift) (y compris le bit implicite) et d'incrémenter son exposant
- Si les exposants sont tràs différents, on risque de perdre des bits significatifs du plus petit nombre
- Pour la soustraction, si les deux nombres diffàrent de peu, il peut y avoir une perte considérable de bits significatifs du résultat

Multiplication et division à virgule flottante

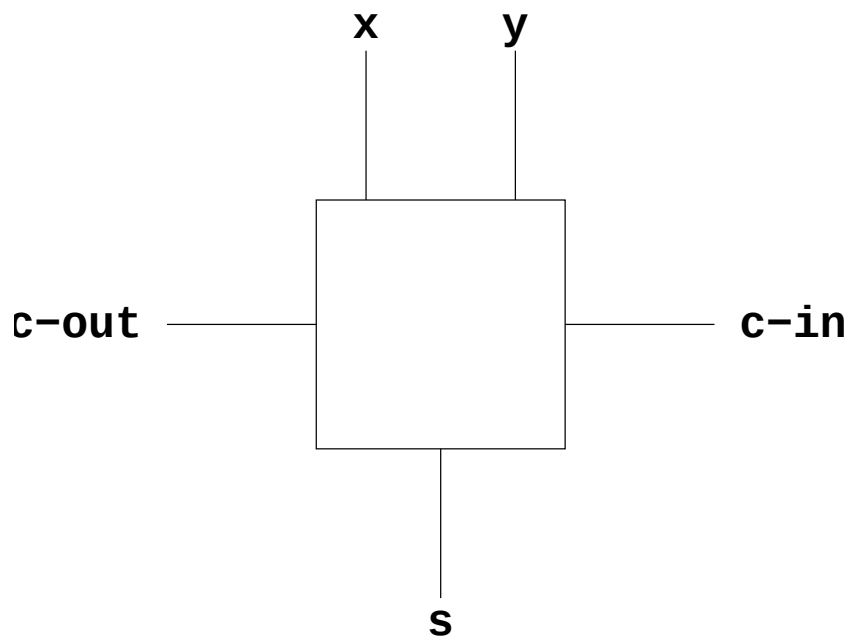
- Pour la multiplication, il suffit de multiplier les mantisses et d'additionner les exposants. Si l'on utilise un additionneur normal, il faut soustraire du résultat pour obtenir la bonne représentation de l'exposant. Il faut aussi éventuellement normaliser le résultat, car la mantisse du résultat peut Ã^{a}tre
- Pour la division, il faut diviser les mantisses et soustraire les exposants supérieure ou égale à 2

Circuits pour l'arithmétique binaire (addition et soustraction)

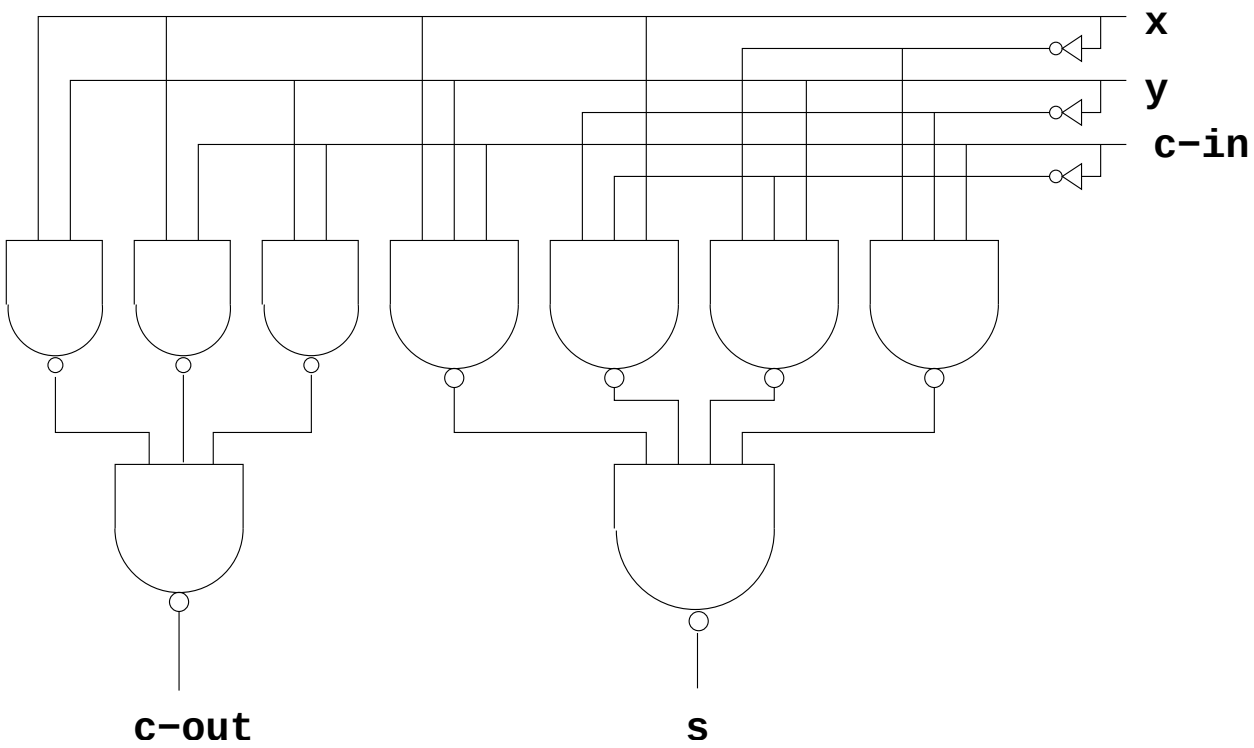
- L'utilisation de la méthode générale pour la construction d'un circuit combinatoire n'est pas possible
- MÃªme avec minimisation, un circuit à deux niveaux peut avoir trop de portes (selon le nombre de bits de la représentation)
- On utilise un circuit combinatoire itératif
- C'est la répétition d'un circuit combinatoire normal
- Pour l'addition est soustraction, on peut commencer par un circuit combinatoire normal pour chaque bit significatif des opérandes
- Puis, on répète ce circuit autant de fois que le nombre de bits le nécessite

Additionneur (1 bit)

x	y	$c-in$	s	$c-out$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

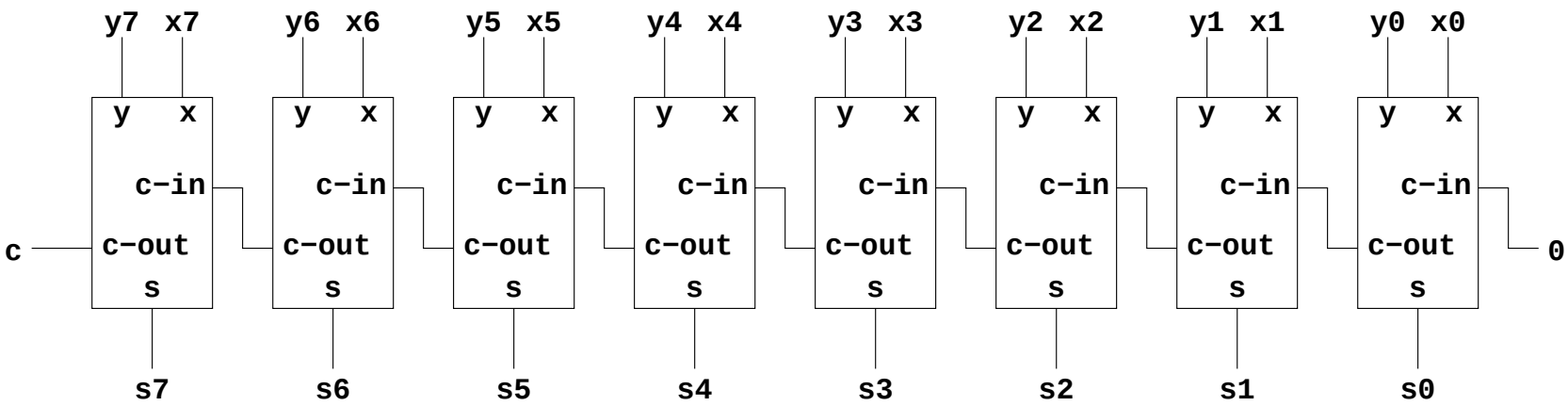


Additionneur, circuit



Additionneur à n bits (exemple $n = 8$)

Il suffit maintenant de répéter le circuit n fois comme ceci :

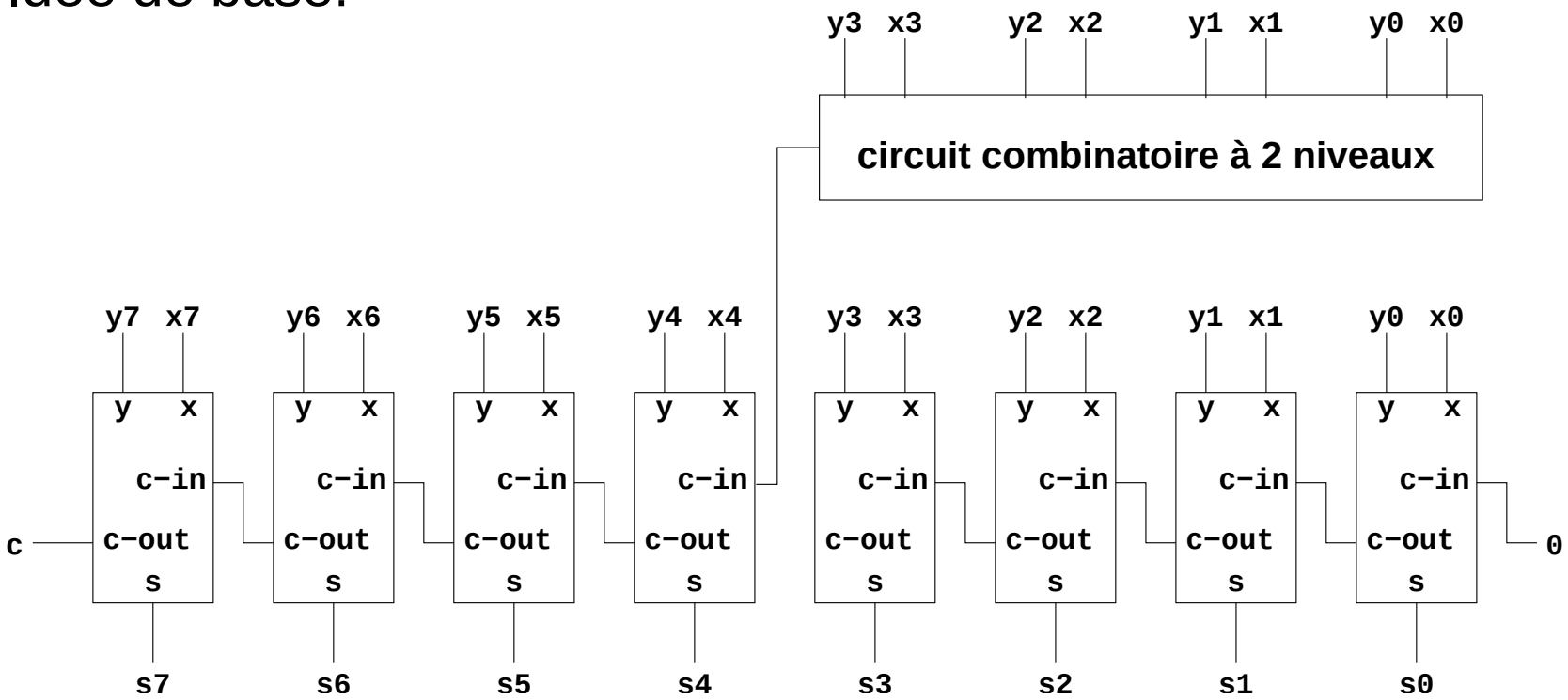


Analyse du circuit d'addition

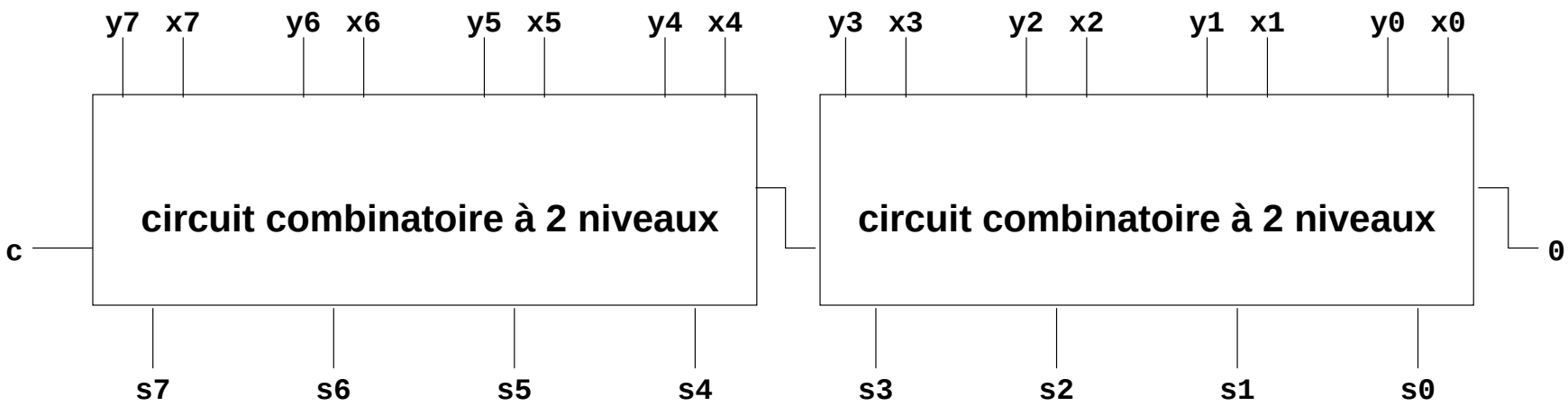
- La profondeur (et donc le délai de calcul de la sortie) est proportionnelle au nombre de bits significatifs
- Le nombre de portes est proportionnel au nombre de bits significatifs
- Pour diminuer la profondeur, on peut imaginer plusieurs solutions intermédiaires :
  - Circuits d'addition de plusieurs bits à la fois (par exemple 4)
  - Circuits pour l'accélération du calcul de la retenue

Accélération du calcul de la retenue

Idée de base:



Addition de plusieurs bits à la fois



Addition et soustraction

- Pour calculer $x - y$, on calcule $x + (-y)$.
- En représentation en complément à 2, on peut calculer $-y$ comme $\overline{y} + 1$
- On peut donc calculer $x + (\overline{y} + 1) = (x + \overline{y}) + 1$
- Le calcul de l'inverse de y se fait avec des portes de type "ou-exclusif"
- L'addition de 1 se fait avec l'entrée $c - in$