TD ALM HARD

Le Multiplieur (document enseignant)

Comment calculer 6*5 ?

Pour l'instant entiers naturels

Exemple en base 2 de multiplication :

```
        110
*       101
_____________
        1 1 0
      0 0 0 .
    1 1 0 . .
_____________
    1 1 1 1 0
```

Notons en binaire A=a2,a1,a0 le multiplicande et B= b2,b1,b0 le multiplieur

S=s4,s3,s2,s1,s0

Combien de bits nécessaires pour coder le résultat ? si A et B sur n bits.

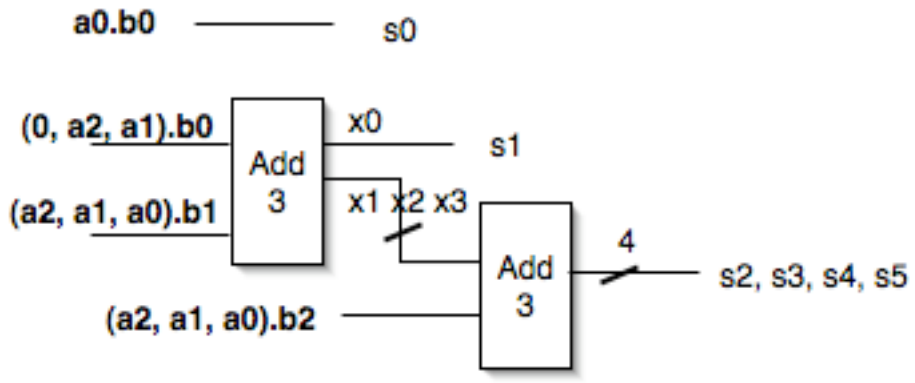
Analyse des opérations effectuées : Et booléen, additions et décalages

Réalisation à l'aide d'additionneurs.

Taille des additionneurs ?

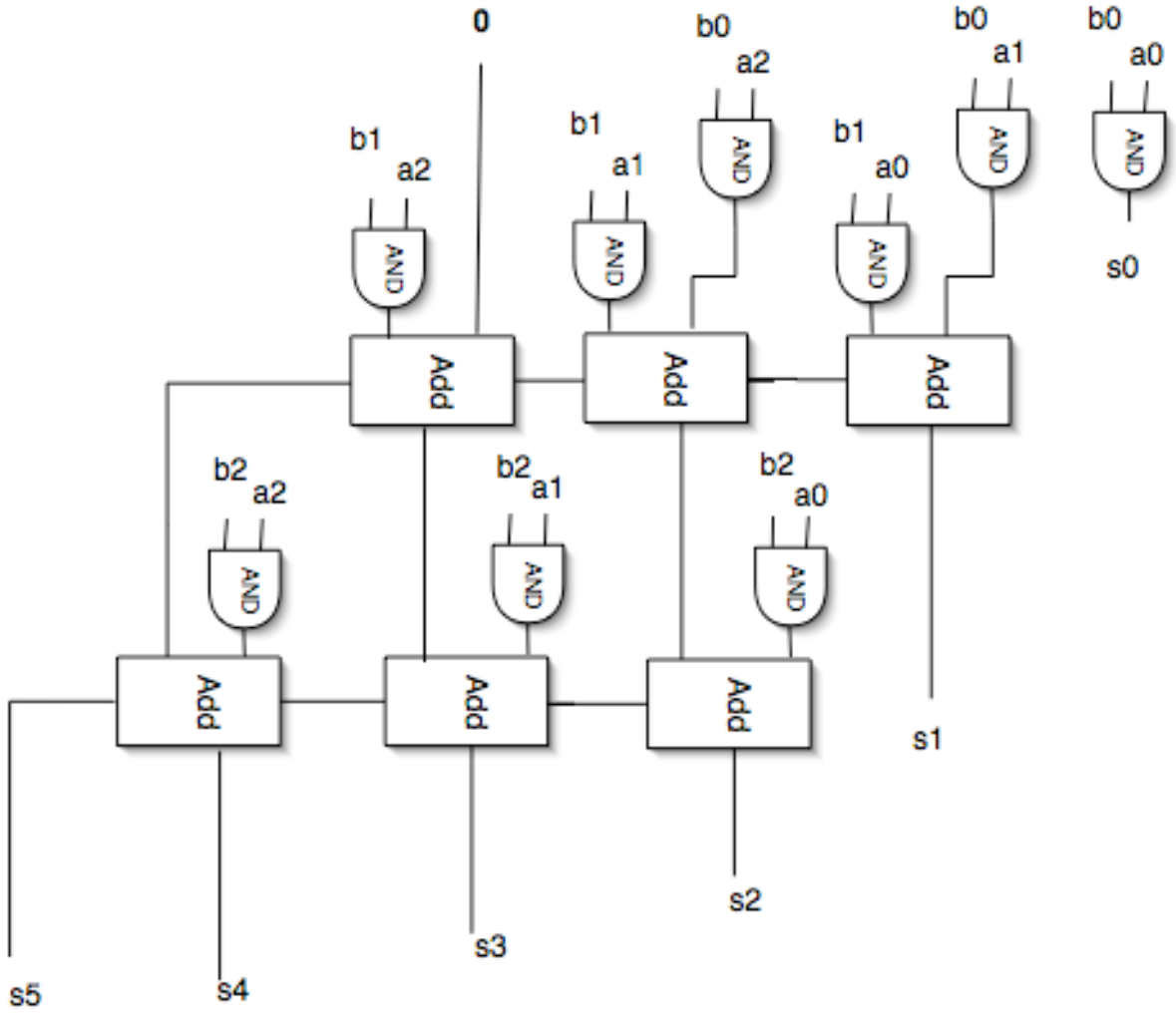
```
        111                         a2 a1 a0
*       111                 *       b2 b1 b0
_____________               _____________
    0 1 1 1                     0   b0.(a2 , a1 , a0)
+   1 1 1 .             +     b1.(a2 , a1 , a0)    0
_____________               _____________________
  1 0 1 0 1               x3    x2    x1     x0    b0.a0
  1 1 1 . .         +   b2.(a2 , a1 , a0)    0     0
_____________     _____________________________
1 1 0 0 0 1         s5  s4    s3    s2     s1    s0
```

Dessin à l'aide d'additionneurs 3 bits :



Si on revient à l'additionneur 1 bits, on arrive au circuit régulier suivant (avec n=3):

		1	1	1				a2	a1	a0	
*		1	1	1		*		b2	b1	b0	
0	1	1	1			0	b0.a2	b0.a1	b0.a0		
+	1	1	1	.		+	b1.a2	b1.a1	b1.a0	0	
1	0	1	0	1		x3	x2	x1	x0	b0.a0	
1	1	1	.	.	+	b2.a2	b2.a1	b2.a0	0	0	
1	1	0	0	0	1	s5	s4	s3	s2	s1	s0



Et si le multiplicande est négatif ?

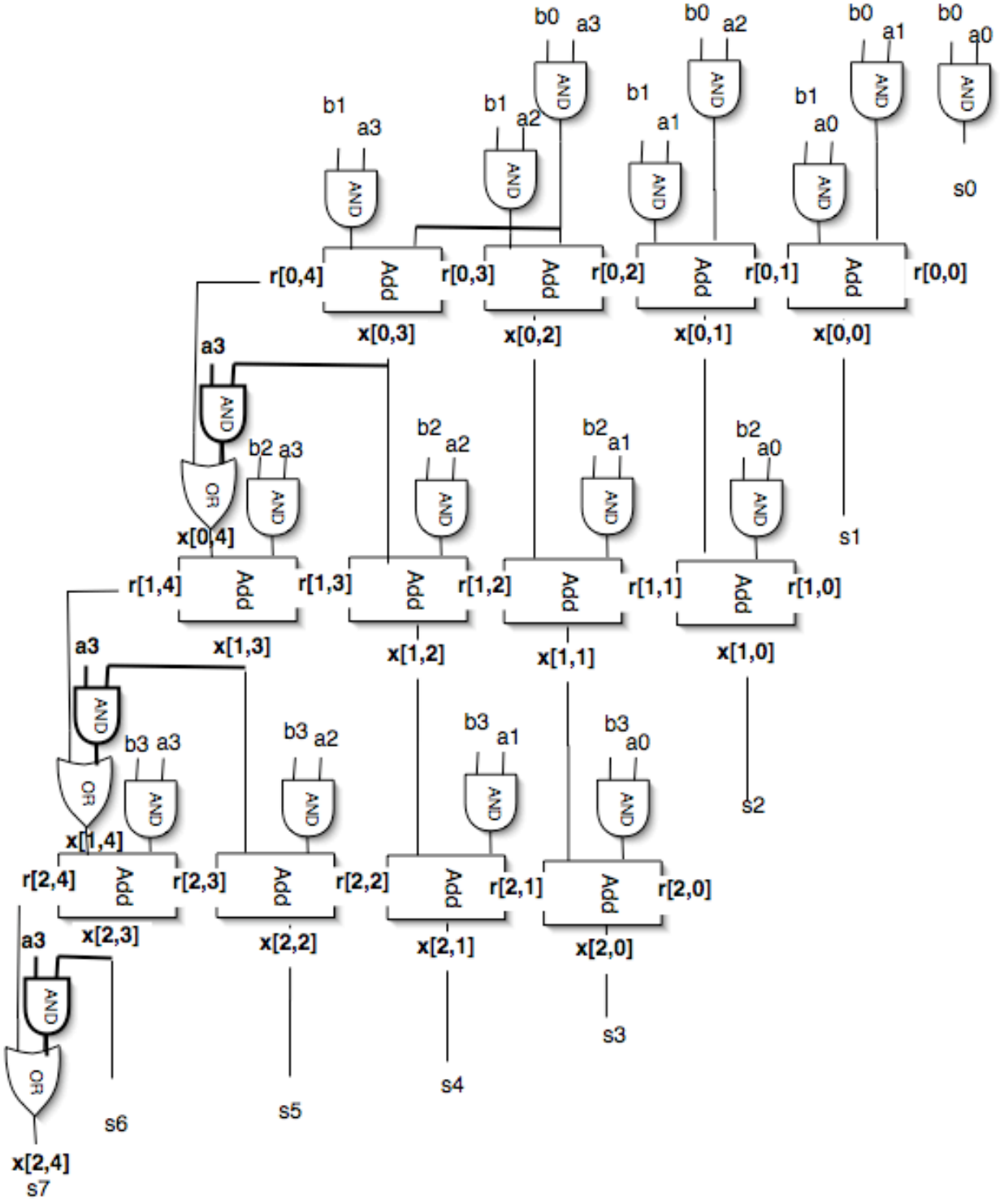
Le multiplicande est en complément à 2, le résultat aussi

Il faut propager le bit de poids fort si il est à 1 dans l'étage suivant pour garder des sommes intermédiaires négatives (et cela seulement si le multiplicande est négatif)

Exemple :

	111		a2 a1 a0
*	101	*	b2 b1 b0
	1 1 1 1		**b0.a2** b0.a2 b0.a1 b0.a0
+	0 0 0 .	+	b1.a2 b1.a1 b1.a0 0
	1 1 1 1 1		x3 x2 x1 x0 b0.a0
	1 1 1 . .	+	b2.a2 b2.a1 b2.a0 0 0
	1 1 1 0 1 1		s5 s4 s3 s2 s1 s0

Multiplieur 4 bits avec 1ere opérande et résultat en complément à 2 :



Description lustre :

Sur 4 bits avec variables intermediaire pour comprendre :

Réalisation en Lustre (à voir plus tard avec les étudiants quand ils auront appris le Lustre)
Faux générique sur 4 bits en base 2

```
--multiplieur 4 bits non générique realise a partir de and 2 entrees et d'add 1 bit
--2 opérandes n bits, résultats 2*n bits le tout en base 2
------------------------------------------------------------
-- Additionneur 1 bit
node add1bit( x, y, rin : bool)
          returns
         (s, rout : bool);
let
      s = x xor y xor rin;
      rout = (x and y) or (x and rin) or (y and rin);
tel
------------------------------------------------------------
--------multiplieur 4 bits
--Voir le dessin 1er indice : colonne, 2eme: ligne
node multnbits(const nn: int; A, B: bool^nn) returns(S: bool^(2*nn));
var x0,x1,x2, r0,r1,r2: bool^(nn+1); --resultat et retenues intermediaires
let
--premier etage
r0[0]= false;
(x0[0..nn-2], r0[1..nn-1])= add1bit( B[0]^(nn-1) and A[1..nn-1], B[1]^(nn-1) and A[0..nn-2],
r0[0..nn-2]);
(x0[nn-1],r0[nn])=add1bit(B[1] and A[nn-1], 0, r0[nn-1]);
x0[nn]= r0[nn];

-- etages de 1 a nn-2
r1[0]= false;
r2[0]= false;
```

```
x1[nn]=r1[nn];
x2[nn]=r2[nn];

(x1[0..nn-1],r1[1..nn])= add1bit(B[2]^nn and A[0..nn-1], x0[1..nn], r1[0..nn-1]);
(x2[0..nn-1],r2[1..nn])= add1bit(B[3]^nn and A[0..nn-1], x1[1..nn], r2[0..nn-1]);

--les sorties
S[0]= A[0] and B[0];
S[1]= x0[0];
S[2]= x1[0];
S[nn-1..(2*nn-1)]= x2[0..nn];
tel;
-- instantiation du multnbits
const a=4;
node instmultnbits(op1, op2: bool^a)
returns(res: bool^(2*a));
let
res= multnbits(a, op1, op2);
tel;
```

Vrai générique (plus compliqué tableau de tableau) :

```
--multiplieur n bits realise a partir de and 2 entrees et d'add 1 bit
--2 opérandes n bits, résultats 2*n bits
--le multiplicande est en complément à 2, il peut etre négatif
--le multiplieur est en base 2
--Le resultat est en complement à 2
-----------------------------------------------------------------
--tableau de tableau m*n
--S[i][j]= A[j]and B[i])
--réalisation recursive

node tabandrec(const n,m : int; A : bool^n; B: bool^m)
```

```
returns(S: bool^n^m);
--attention tableau de tableau : 1er indice contenu des tableau, 2eme indice :tableau de tableau
--indice inverse dans l'utilisation par rapport à la declaration: ici m tableau de n élements: S[0..m-1,0..n-1]
--attention ecriture S[0..m-1][0..n-1] autre semantique

let
-- attention utiliser with au lieu de if sinon le compilateur produit un code qui boucle ??
S=with (m=1) then ([B[0]^n and A])
        else  ( tabandrec(n, m-1, A , B[0..m-2])|[B[m-1]^n and A]);
        --| est l'opération de concatenation de tableau
tel;
-------------------------------------------------------------
--------multiplieur base sur des tableaux de tableaux
--Voir le dessin 1er indice : colonne, 2eme: ligne

node multnbits(const nn: int; A, B: bool^nn)
returns(S: bool^(2*nn));
var r, x: bool^(nn+1)^nn-1; --resultat et retenues intermediaires
taband:bool^nn^(nn-2); --tableau des and ai bj
let

--premier etage---------
r[0,0]= false;
(x[0,0..nn-2], r[0,1..nn-1])= add1bit( B[0]^(nn-1) and A[1..nn-1], B[1]^(nn-1) and A[0..nn-2], r[0,0..nn-2]);
(x[0, nn-1],r[0, nn])= add1bit(B[1] and A[nn-1], A[nn-1] and B[0], r[0,nn-1]);
-- remplace A[nn-1]and B[0] par 0 pour un multiplicande en base 2 (non signe)
x[0,nn]= r[0,nn] or ((x[0,nn-1])and(A[nn-1]));
-- remplace x[0,nn-1] and A[nn-1] par 0 pour un multiplicande en base 2 (non signe)

-- etages de 1 a nn-2-------
```

```
r[1..nn-2,0]= false^(nn-2);--attention parenthese obligatoire

x[1..nn-2,nn]=r[1..nn-2,nn] or (x[1..nn-2,nn-1]and (A[nn-1])^(nn-2)) ;
-- remplace x[1..nn-2,nn-1] and (A[nn-1])^(nn-2) par 0 pour un multiplicande en base 2 (non signe)

taband=tabandrec(nn, nn-2, A[0..nn-1],B[2..nn-1]);
(x[1..nn-2,0..nn-1],r[1..nn-2,1..nn])= add1bit(taband, x[0..nn-3,1..nn], r[1..nn-2,0..nn-1]);

--les sorties
S[0]= A[0] and B[0];
S[1..nn-1]= x[0..nn-2,0];
S[nn..(2*nn-1)]= x[nn-2,1..nn];
tel;

-- instantiation du multnbits generique
const a=3;
node instmultnbits(op1, op2: bool^a)
returns(res: bool^(2*a));
let
res= multnbits(a, op1, op2);
tel;
```