Retour sur les opérateurs logiques

Par André C.

Les opérateurs logiques servent à deux choses principales: d'une part forcer certains bits à 0 ou 1 (par exemple pour la vidéo inverse ou pour le registre d'états #26A), d'autre part à élaborer des conditions d'exécution pour IF THEN et REPEAT UNTIL. Dans ce dernier cas, les tests arithmétiques <, >, = et leurs différentes combinaisons <>, <=, >= sont également utilisés.
Le Basic de l'Oric ne comporte que les 3 opérateurs logiques: AND, OR et NOT à partir desquels il est possible d'en émuler quelques autres comme NAND, XNOR et XOR. Une légère difficulté est rencontrée avec les expressions conditionnelles, car normalement FALSE vaut 0 et TRUE vaut 1, mais pour l'Oric si FALSE vaut bien 0, par contre TRUE vaut -1. Il s'en suit que les réactions des tests comme IF ... THEN et REPEAT ... UNTIL sont parfois surprenantes avec les opérateurs logiques (voir le paragraphe consacré à l'opérateur logique NOT ainsi que la conclusion de cet article).

Pour comprendre les recopies d'écran, voir page suivante le programme utilisé pour mes tests.

L'opérateur logique AND (ET)

Syntaxe a AND b où a et b sont, soit des expressions conditionnelles (comme A>B etc. ou TRUE ou FALSE), soit des nombres entiers (de 0 à 255).

Table de vérité AND

Entrées		Sortie
a	b	L
0	0	0
0	1	0
1	0	0
1	1	1

Le résultat est 1 seulement si les deux opérandes a et b sont à 1. AND fonctionne ici comme un produit arithmétique: L = ab.
Avec l'Oric, la table de vérité ci-dessus est vérifiée (voir la figure 1). Outre les valeurs 0 et 1, les entrées a et b peuvent être des nombres (de 0 à 255) (voir la figure 2). Dans ce dernier cas, chaque bit de l'octet en sortie répond aux bits correspondants en entrée, selon la table de vérité ci-dessus.

```
CHOIX:
1 AND
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 1 AND
a=? 0
b=? 0
 0
AND
a=? 0
b=? 1
 0
AND
a=? 1
b=? 0
 0
AND
a=? 1
b=? 1
 1
```

Figure 1

```
 1 AND
a=? 0
b=? 0
 0
AND
a=? 0
b=? 1
 0
AND
a=? 1
b=? 0
 0
AND
a=? 1
b=? 1
 1
AND
a=? 85
b=? 129
 1
AND
a=? 21845
b=? 33153

?ILLEGAL QUANTITY
Ready
```

Figure 2

Exemple:
```
a=0101 0101 (#55=85)
b=1000 0001 (#81=129)
L=0000 0001 (#01=1)
```

Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Fonction_ET

L'opérateur logique OR (OU inclusif)

Syntaxe L = a OR b

Table de vérité OR

Entrées		Sortie
a	b	L
0	0	0
0	1	1
1	0	1
1	1	1

Le résultat est 0 seulement si les deux opérandes a et b sont à 0. AND fonctionne comme une somme: L = a+b
Avec l'Oric, la table de vérité ci-dessus est vérifiée (voir la figure 3, page suivante). Outre les valeurs 0 et 1, les entrées a et b peuvent être des nombres (de 0 à 255) (voir la figure 3, page suivante). Dans ce dernier cas, chaque bit de l'octet en sortie répond aux bits correspondants en entrée, selon la table de vérité ci-dessus.
Exemple
```
a=0101 0101 (#55=85)
b=1000 0001 (#81=129)
L=1101 0101 (#D5=213)
```

```
100 PRINT"TESTS OPERATEURS LOGIQUES"
110 CLS:PRINT"CHOIX:"
120 PRINT"1 AND"
130 PRINT"2 OR"
140 PRINT"3 NOT"
150 PRINT"4 NAND"
160 PRINT"5 XNOR"
170 PRINT"6 XOR"
200 PRINT:GET R:PRINT R;
210 IF R=1 THEN 1000
220 IF R=2 THEN 2000
230 IF R=3 THEN 3000
240 IF R=4 THEN 4000
250 IF R=5 THEN 5000
260 IF R=6 THEN 6000
270 GOTO 200
1000 PRINT"AND"
1010 INPUT"a=";A
1020 INPUT"b=";B
1030 L=A AND B
1040 PRINT L
1050 GET R:IF R=1 THEN 1000 ELSE RUN
2000 PRINT"OR"
2010 INPUT"a=";A
2020 INPUT"b=";B
2030 L=A OR B
2040 PRINT L
2050 GET R:IF R=1 THEN 2000 ELSE RUN
3000 PRINT"NOT"
3010 INPUT"a=";A
3030 L=1-A:PRINT L
3035 ' Normal 0=FALSE et 1=TRUE
3040 L=NOT A:PRINT L
3045 ' Oric 0=FALSE et -1=TRUE
3050 GET R:IF R=1 THEN 3000 ELSE RUN
4000 PRINT"NAND"
4010 INPUT"a=";A
4020 INPUT"b=";B
4030 L=NOTA OR NOTB:PRINT L
4040 GET R:IF R=1 THEN 4000 ELSE RUN
5000 PRINT"XNOR"
5010 INPUT"a=";A
5020 INPUT"b=";B
5030 L=(A AND B) OR (NOTA AND NOTB)
5040 PRINT L
5050 GET R:IF R=1 THEN 5000 ELSE RUN
6000 PRINT"XOR"
6010 INPUT"a=";A
6020 INPUT"b=";B
6030 L=(A AND NOTB) OR (NOTA AND B)
6040 PRINT L
6050 GET R:IF R=1 THEN 6000 ELSE RUN
```

Ce petit programme en Basic est très sommaire et n'a pour but que de faciliter l'entrée des opérandes des opérateurs logiques de l'Oric et d'afficher le résultat. Pas de fioritures donc, la présentation est brute de chantier!

Au programme, les opérateurs suivants:
1) AND
2) OR
3) NOT
4) NAND
5) XNOR
6) XOR

NB. a, b et L se réfèrent aux tables de vérité publiées dans l'article. A la fin de chaque test, le programme attend une touche. Si c'est '1', il reboucle sur le même test. Pour toute autre touche il repart au choix des opérateurs logiques.

Pour l'opérateur NOT, deux résultats sont calculés et affichés:
1) Le résultat normal utilisant la formule L=1-A
2) Le résultat donné par l'opérateur NOT de l'Oric. En effet, cet opérateur n'est pas orthodoxe et attribue la valeur -1 à TRUE au lieu de 1.

L'opérateur logique NAND n'existant pas dans le Basic de l'Oric, il est ici émulé par la formule: L=NOTA OR NOTB.

L'opérateur logique XNOR n'existant pas dans le Basic de l'Oric, il est ici émulé par la formule: L=(A AND B) OR (NOTA AND NOTB).

L'opérateur logique XOR n'existant pas dans le Basic de l'Oric, il est ici émulé par la formule: L=(A AND NOTB) OR (NOTA AND B).

Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Fonction_OU

L'opérateur logique NOT (NON)

Syntaxe L = NOT a

Table de vérité normale NOT	
Entrée a	Sortie L
0	1
1	0

Le résultat L est la valeur inverse de celle de l'opérande a. Seules les valeurs 0 (FALSE) et 1 (TRUE) sont autorisées en entrée. NOT correspond, en arithmétique, à L=1-a (1ère ligne de résultat sur les figures 5 et 6).
Sur la figure 5, on voit sur la 2ème ligne de résultat que le NOT de l'Oric donne des valeurs erronées (sauf avec 0). Avec l'Oric aussi, il faut indiquer FALSE et TRUE en entrée, mais si FALSE vaut bien encore 0, par contre le TRUE de l'Oric vaut -1.

Table de vérité NOT de l'Oric	
Entrée a	Sortie L
0	-1
-1	0

Sur la figure 6, on voit qu'avec les valeurs 0 (FALSE) et -1 (TRUE), l'Oric donne les bonnes réponses, respectivement -1 (TRUE) et 0 (FALSE). Dans les deux cas, soit application de la formule L=1-A (1ère ligne des résultats), soit de L=NOTA (2ème ligne des résultats), toute valeur autre que FALSE ou TRUE donne des résultats erronés (en tous cas je ne suis pas parvenu à les comprendre). Voir aussi la conclusion à la fin de cet article.

Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Fonction_NON

```
CHOIX:
1 AND
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 2 OR
a=? 0
b=? 0
 0
OR
a=? 0
b=? 1
 1
OR
a=? 1
b=? 0
 1
OR
a=? 1
b=? 1
 1
```

Figure 3

```
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 2 OR
a=? 0
b=? 0
 0
OR
a=? 0
b=? 1
 1
OR
a=? 1
b=? 0
 1
OR
a=? 1
b=? 1
 1
OR
a=? 85
b=? 129
 213
```

Figure 4

```
CHOIX:
1 AND
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 3 NOT
a=? 0
 1
-1
NOT
a=? 1
 0
-2
NOT
a=? 85
-84
-86
NOT
a=? 129
-128
-130
```

Figure 5

```
6 XOR

 3 NOT
a=? 0
 1
-1
NOT
a=? 1
 0
-2
NOT
a=? 85
-84
-86
NOT
a=? 129
-128
-130
NOT
a=? 0
 1
-1
NOT
a=? -1
 2
 0
```

Figure 6

L'opérateur logique NAND (NON-ET)

Syntaxe normale: L = a NAND b
Emulation sur Oric: L = NOT a OR NOT b
Le résultat est 0 seulement si les deux opérandes a et b sont à 1. Il faudrait plutôt écrire "Le résultat est FALSE seulement si les deux opérandes a et b sont TRUE. En effet, l'opérateur NOT de l'Oric fonctionne de manière spéciale (voir le paragraphe précédent consacré à NOT). Notre fonction NAND ne marchera que si on met en entrée

Table de vérité normale NAND		
Entrées		Sortie
a	b	L
0	0	1
0	1	1
1	0	1
1	1	0

Table de vérité NAND de l'Oric		
Entrées		Sortie
a	b	L
0	0	-1
0	-1	-1
-1	0	-1
-1	-1	0

FALSE (0) ou TRUE (-1). On retrouve bien -1 NAND -1 = 0 et on obtient bien -1 (TRUE) pour les 3 autres combinaisons (voir la figure 7). Dans les autres cas on aura des réponses erronées (voir la figure 8). Si on place des entiers (0 à 255) en entrées, ils seront considérés comme si on avait tapé le chiffre 1 (figure 8).
Conclusion: à utiliser avec précaution.
Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Fonction_NON-ET

L'opérateur logique XNOR (NON-OU exclusif)

Syntaxe normale: L = a XNOR b
Emulation sur Oric: L = (a AND b) OR (NOT a AND NOT b)
L = 0 si et seulement si a différent de b.
Ce qui correspond à L = (a AND b) OR (NOTa AND NOTb)
La fonction XNOR peut ainsi être émulée avec les opérateurs AND, OR et NOT.

Table de vérité normale XNOR

Entrées		Sortie
a	b	L
0	0	1
0	1	0
1	0	0
1	1	1

Avec l'Oric, même remarque que pour NAND. Si les entrées sont limitées à 0 (FALSE) et -1 (TRUE).Les figures 9 et 10 montrent que ça marche selon la table de vérité suivante:

Table de vérité XNOR de l'Oric

Entrées		Sortie
a	b	L
0	0	-1
0	-1	0
-1	0	0
-1	-1	-1

Si on place des entiers (0 à 255) en entrée, l'Oric donne des résultats erronés (en tous cas je ne suis pas parvenu à les comprendre, sinon que le résultat trouvé = résultat normal - 256) (voir la figure 11 page suivante).
Exemple:
```
a=0011 0011 (#33=51)
b=1000 0000 (#80=128)
L=0100 1100 (#4C=76 et non -180!)
```

```
CHOIX:
1 AND
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 4 NAND
a=? 0
b=? 0
-1
NAND
a=? -1
b=? 0
-1
NAND
a=? 0
b=? -1
-1
NAND
a=? -1
b=? -1
 0
```
Figure 7

```
b=? 0
-1
NAND
a=? 0
b=? -1
-1
NAND
a=? -1
b=? -1
 0
NAND
a=? 1
b=? 1
-2
NAND
a=? 0
b=? 1
-1
NAND
a=? 1
b=? 0
-1
NAND
a=? 85
b=? 129
-2
```
Figure 8

```
CHOIX:
1 AND
2 OR
3 NOT
4 NAND
5 XNOR
6 XOR

 5 XNOR
a=? 0
b=? 0
-1
XNOR
a=? 0
b=? 1
-2
XNOR
a=? 0
b=? -1
 0
XNOR
a=? 1
b=? 0
-2
XNOR
a=? 1
b=? 1
```
Figure 9

```
b=? 0
-1
XNOR
a=? 0
b=? 1
-2
XNOR
a=? 0
b=? -1
 0
XNOR
a=? 1
b=? 0
-2
XNOR
a=? -1
b=? 0
 0
XNOR
a=? 1
b=? 1
-1
XNOR
a=? -1
b=? -1
-1
```
Figure 10

Conclusion: Egalement à utiliser avec précaution.
Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Coincidence

L'opérateur logique XOR (OU exclusif)

Syntaxe normale: L = a XOR b
Emulation sur Oric: L = (a AND NOT b) OR (NOT a AND b)
Le résultat est 1 si un et un seul des opérandes A et B est 1 ou encore le résultat est 1 si les deux opérandes A et B ont des valeurs distinctes.
Ce qui correspond à L = (a AND NOT b) OR (NOT a AND b). La fonction XOR peut ainsi être émulée avec les opérateurs AND, OR et NOT.
Cette fois, chose curieuse, malgré l'utilisation de la commande NOR, l'émulation de XOR sur l'Oric fonctionne correctement (voir la figure 12 page suivante). La table de vérité ci-dessus est vérifiée et qui plus est, si a et b sont des octets de 0 à 255, le résultat est correct! (Figure 12 page suivante).

Table de vérité XOR		
Entrées		Sortie
a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

Exemple:
```
a=0101 0101 (#55=85)
b=1000 0001 (#81=129)
L=1101 0100 (#D4=212)
```

Au-dessus de 255, ça continue à marcher jusqu'à une certaine limite que je n'ai pas cherché à définir, car cela présente peu d'intérêt. Au-delà de cette limite, on obtient une ?ILLEGAL QUANTITY ERROR.

Il est TRES heureux que l'émulation de XOR fonctionne correctement sur l'Oric, car c'est un opérateur logique de base et on peut s'étonner qu'il ne fasse pas partie de la panoplie de base!

Pour d'autres informations voir:
https://fr.wikipedia.org/wiki/Fonction_OU_exclusif

Conclusions

Pas de problème avec AND, OR et XOR [émulé par (a AND b) OR (NOT a AND NOT b)].

Les constantes TRUE (VRAI) et FALSE (FAUX) ont des valeurs qui dépendent des systèmes. Dans le système de Boole, TRUE=1 et FALSE=0. Ainsi on a bien NOT TRUE = FALSE et NOT FALSE = TRUE.

Mais pour l'Oric, TRUE=-1 et FALSE=0. Et on a bien NOT TRUE = 0 et NOT FALSE = -1 (figure 13 et 14). D'après le manuel de l'Atmos (page 214), toute valeur non nulle est considérée par l'Oric comme TRUE.

Mais curieusement NOT22 = -23 (22, non nul, devrait être considérée comme TRUE et NOT22 devrait donc donner FALSE soit 0). Y-a comme une bogue!

Je ne vais pas vous entraîner dans le labyrinthe des tests où l'on observe que NOT+valeur négative (dont -1, soit TRUE) donne une valeur positive ou nulle et NOT+valeur positive ou nulle (soit FALSE) donne une valeur négative (figures 13 et 14). N'en tirez aucune conclusion! Evitez d'utiliser NOT dans un test conditionnel de type IF THEN ou REPEAT UNTIL car NOT donne à peu près n'importe quoi. Tenez-vous en à "toute valeur non nulle est considérée par l'Oric comme TRUE".

```
b=? -1
 0
XNOR
a=? 1
b=? 0
-2
XNOR
a=? -1
b=? 0
 0
XNOR
a=? 1
b=? 1
-1
XNOR
a=? -1
b=? -1
-1
XNOR
a=? 51
b=? 128
-180
XNOR
a=? 85
b=? 129
-213
```
Figure 11

```
6 XOR

 6 XOR
a=? 0
b=? 0
 0
XOR
a=? 0
b=? 1
 1
XOR
a=? 1
b=? 0
 1
XOR
a=? 1
b=? 1
 0
XOR
a=? 51
b=? 128
 179
XOR
a=? 85
b=? 129
 212
```
Figure 12

```
Ready
?NOTTRUE
 0

Ready
?NOT-1
 0

Ready
?NOT22
-23

Ready
?NOT-23
 22

Ready
?NOT128
-129

Ready
?NOT-129
 128

Read
```
Figure 13

```
Ready
?NOTFALSE
-1

Ready
?NOT 0
-1

Ready
?NOT255
-256

Ready
?NOT256
-257

Ready
?NOT#55
-86

Ready
?NOT#5555
-21846

Read
```
Figure 14

```
LIST

  100 IF TRU
  200 IF-1 T
  300 IF NOT
  400 IF -22
  500 IF 22
  600 IF FAL
  700 IF 0 T
  800 IF FAL

Ready

RUN
TRUE
-1
NOT0
-22
22
FALSE=0

Ready
```
Figure 15

Avec IF THEN l'action après le THEN n'est effectuée que si la condition est TRUE. Sur la figure 15, la commande PRINT est exécutée pour les valeurs TRUE, -1, -22 et NOT0 (donc -1). La commande PRINT n'est pas exécutée avec les valeurs FALSE (ligne 600) ou 0 (ligne 700). La valeur 22 (non nulle, donc TRUE) entraîne bien l'exécution du PRINT! Rien d'étonnant non plus à ce que FALSE=0 (c'est vrai) entraîne l'exécution du PRINT!

Quoi qu'il en soit, utilisez l'opérateur NOT avec prudence (idem avec les émulations de NAND et XNOR). Et rappelez-vous toujours que "Toute valeur différente de zéro est TRUE" et que "Toute condition TRUE entraîne l'exécution situé après le THEN" (je ne vais pas plus loin à cause de la bogue IF THEN bien connue).