Appunti di Logica Ternaria: Operatori Diadici

Giuseppe Talarico 27 Gennaio 2014

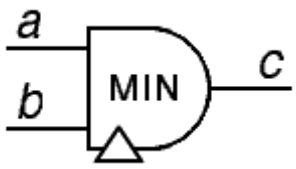
Nella logica ternaria, una tabella di verità con due ingressi ha nove righe, per cui ne consegue che il numero totale delle funzioni con due ingressi ternari è pari a 3^9 = 19'683. La enumerazione di un numero così alto di funzioni non è praticabile. In ogni caso, si può dimostrare che solo un numero limitatissimo di esse è necessario per realizzare una logica ternaria completa.

TAND o TMIN

La funzione ternaria TAND è true (1) se e solo se i due ingressi hanno valore true, false ($\bar{1}$) se uno dei due ingressi è false e unknown (0) negli altri casi. Essendo false < unknown < true l'and ternario può essere pensato come l' operatore che ritorna il minimo dei due valori di ingresso. Di seguito si riporta la tabella di verità, la funzione ed il simbolo:

TMIN		b		
		$\bar{1}$	0	1
a	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$
	0	$\bar{1}$	0	0
	1	$\bar{1}$	0	1

$c = a \wedge b = \mathrm{TMIN}(a, b)$



Come nella logica booleana la funzione TAND o TMIN è sia commutativa che associativa.

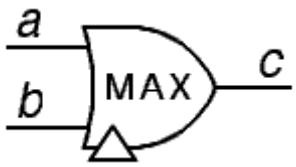
TOR o TMAX

La funzione ternaria TOR è true (1) se uno due ingressi ha valore true, false ($\bar{1}$) se entrambi i due ingressi sono false e unknown (0) negli altri casi. Essendo false < unknown < true l'or ternario può essere pensato come l' operatore che ritorna il massimo dei due valori di ingresso. Di

seguito si riporta la tabella di verità, la funzione ed il simbolo:

TMAX		b		
		$\bar{1}$	0	1
a	$\bar{1}$	$\bar{1}$	0	1
	0	0	0	1
	1	1	1	1

$c = a \vee b = \mathrm{TMAX}(a, b)$



La funzione TOR o TMAX è sia commutativa che associativa. Come nella logica booleana anche in quella ternaria valgono i due teoremi di De Morgan:

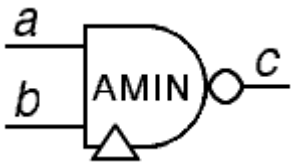
$$a \vee b = -(-a \wedge -b) \quad \text{ovvero:} \quad TMAX(a, b) = -\,\mathrm{TMIN}(-a, -b)$$
$$a \wedge b = -(-a \vee -b) \quad \text{ovvero:} \quad \mathrm{TMIN}(a, b) = -\,\mathrm{TMAX}(-a, -b)$$

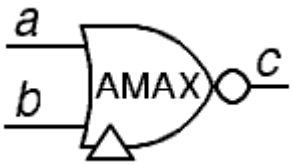
L' applicazione di questi teoremi porta ad affermare che gli operatori TMIN e TMAX possono ricavarsi l'uno dall' altro con l' ausilio dell'operatore monadico STI (Standard Ternary Inverter).

- Nella logica booleana, come è noto, ogni funzione combinatoria può essere espressa in termini di soli and or e not. Applicando DeMorgan and e not oppure or e not sono pure sufficienti per rappresentare qualunque funzione booleana.
- Nella logica ternaria vi sono delle funzioni monadiche come i decoder di 1 ("is true") e di $\bar{1}$ ("is false") e la funzione "decremento" che non possono essere espresse in termini di TMIN, TMAX e STI. Tuttavia è possibile dimostrare che aggiungendo una di queste funzioni a TMIN, TMAX e STI si può realizzare qualunque funzione ternaria.

TNAND e TNOR

Aggiungendo uno STI ai due operatori TAND e TOR si ottengono i due operatori TAntiMin e TAntiMax.





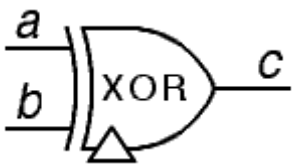
Le tabelle di verità e le funzioni si lasciano agli studenti come esercizio.

TXOR

La funzione ternaria dell'OR esclusivo si può ricavare da quelle già introdotte, infatti vale: $a \oplus b = (a \wedge -b) \vee (b \wedge -a)$. Di seguito si riporta la tabella di verità, la funzione diadica ed il simbolo.

c	b = $\bar{1}$	0	1
$\bar{1}$	$\bar{1}$	0	1
a 0	0	0	0
1	1	0	$\bar{1}$

$c = a \oplus b = \mathrm{TXOR}(a, b)$



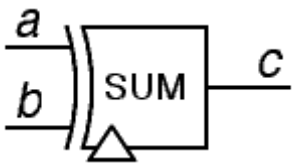
La funzione TXOR è sia commutativa che associativa. Come per il NOT, nella logica ternaria, esistono funzioni analoghe all'OR esclusivo.

TSUM

Nella logica booleana lo xor viene utilizzato nel semi-sommatore per calcolare la somma (modulo 2) dei due bit d'ingresso. Il TXOR non fa la stessa cosa per cui si definisce un altro operatore detto TSUM che calcola la somma (modulo 3) con valori compresi in $\{\bar{1},0,1\}$. La tabella di verità, la funzione ed il simbolo, che richiama lo xor, sono qui appresso riportati:

c	b		
	$\bar{1}$	0	1
$\bar{1}$	1	$\bar{1}$	0
0	$\bar{1}$	0	1
1	0	1	$\bar{1}$

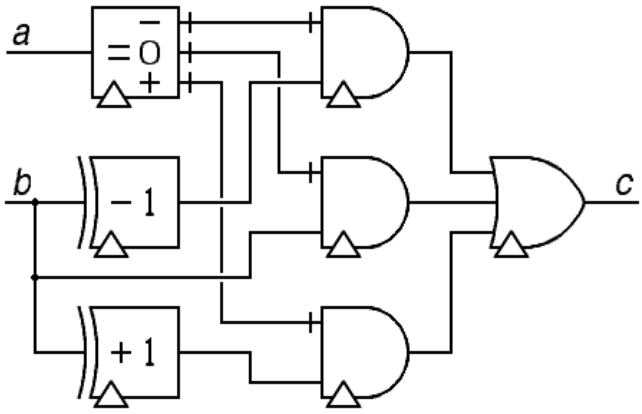
$c = a + b = \mathrm{TSUM}(a,b)$



La funzione TSUM è sia commutativa che associativa. L'operatore TSUM, non è strettamente necessario, poiché è ottenibile da quelli già introdotti. Infatti, vale, come è facile dimostrare:

$a + b = ((a = -1) \wedge (b - 1)) \vee ((a = 0) \wedge b) \vee ((a = +1) \wedge (b + 1))$

da cui si ottiene il seguente circuito:



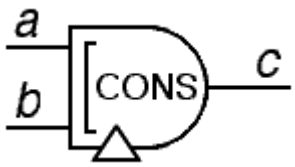
I collegamenti con il taglietto stanno ad indicare, come già detto, segnali a due livelli e quindi booleani. Nello schema, l' input "a" controlla un MUX 3:1 (realizzato dal decoder, da tre TAND e dal TOR) che lascia passare in uscita l'input "b" decrementato, "b" o "b" incrementato a seconda del valore del trit "a".

TCONS

Nella logica booleana, la funzione coincidenza ovvero l' inverso dello xor (xnor) è true se i due input sono uguali e false quando sono differenti. Nella logica ternaria la naturale estensione di un operatore siffatto porta al TConsensus la cui uscita è true se i due input sono true (1), false se i due input sono false ($\bar{1}$) e unkown (0) in tutti gli altri casi. Di seguito si riporta la tabella di verità, la funzione diadica ed il simbolo:

c / a \ b	$\bar{1}$	0	1
$\bar{1}$	$\bar{1}$	0	0
0	0	0	0
1	0	0	1

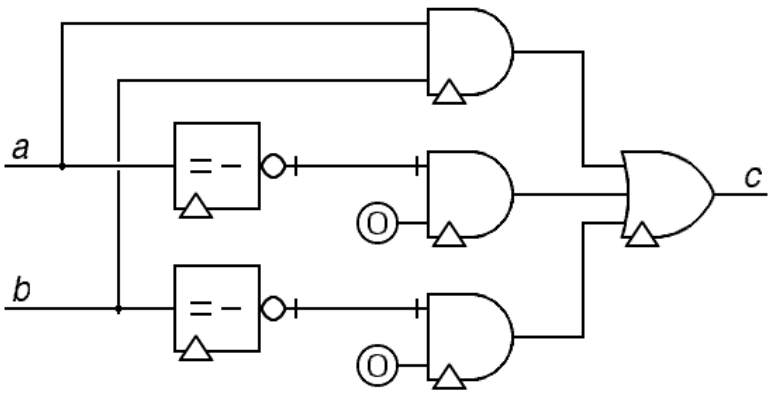
$c = a \boxtimes b = \mathrm{TCONS}(a, b)$



Il simbolo include elementi presi in prestito dallo xor e dall' and booleani. La similitudine con l' and si ha se si considerano equivalenti false e unknown. E' possibile dimostrare che la funzione TCONS può essere descritta da:

$$a \boxtimes b = (a \wedge b) \vee ((a \neq -1) \wedge 0) \vee ((b \neq -1) \wedge 0)$$

che porta al circuito:



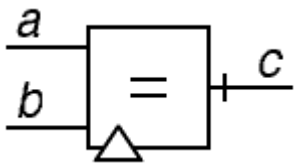
TEQUAL

Le funzioni monadiche di decodifica hanno una implementazione anche come funzione diadica: la funzione di uguaglianza TEQUAL. Il valore della funzione è true se e solo se i valori dei due trit d'ingresso sono uguali.

Di seguito si riportano la tabella di verità, la funzione ed il simbolo diadico:

c		b		
		$\bar{1}$	0	1
a	$\bar{1}$	1	$\bar{1}$	$\bar{1}$
	0	$\bar{1}$	1	$\bar{1}$
	1	$\bar{1}$	$\bar{1}$	1

$c = (a = b) = \text{TEQUAL}(a, b)$



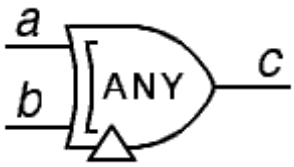
Da notare che questa funzione considera due input di valore unknown come due valori uguali !

TANY

L' operatore ternario "*accept anything*" dà in uscita un valore unknown (0) solo se entrambi gli inputs sono unknown {0,0} oppure discordanti: {1,$\bar{1}$} o {$\bar{1}$,1} Negli altri casi l' uscita coincide con l'input "non-unknown" .

c		b		
		$\bar{1}$	0	1
a	$\bar{1}$	$\bar{1}$	$\bar{1}$	0
	0	$\bar{1}$	0	1
	1	0	1	1

$c = a \boxplus b = \text{TANY}(a, b)$



Balanced Half Adder Ternario

Come primo esempio di circuito ternario consideriamo il semisommatore nel sistema ternario bilanciato, la cui tabella di verità è la seguente:

a_i	c_i	Somma s_i	Riporto c_{i+1}
$\bar{1}$	$\bar{1}$	1	$\bar{1}$
$\bar{1}$	0	$\bar{1}$	0
$\bar{1}$	1	0	0
0	$\bar{1}$	$\bar{1}$	0
0	0	0	0
0	1	1	0
1	$\bar{1}$	0	0
1	0	1	0
1	1	$\bar{1}$	1

Osservando la colonna Somma si riconosce che essa è nient'altro che la funzione diadica TSUM(a_i,c_i), mentre la colonna Riporto è la TCONS (consensus) per cui il circuito logico è il seguente: