UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

20 Novembre 2023

Esercitazione in aula

Esercizio 1. Sia F una funzione che riceve in ingresso un numero intero n rappresentato su 4 bit in codice eccesso 2^(k-1) con k=4. F assume valore 1 quando n vale -4, -1, 2, 4, 5, o 7 e può assumere indifferentemente il valore 1 o 0 quando n vale -2, 0, 3, o 6. F restituisce 0 per gli altri valori di n.

1. Realizzare il circuito che implementa F usando le mappe di Karnaugh sintetizzando in forma SoP. Riportare i passaggi e disegnare il circuito derivato.
2. Realizzare un circuito equivalente a quello derivato al punto 1 usando solo porte NAND.

Esercizio 2. Derivare la codifica floating point IEEE 754 in singola precisione del numero -432,3125. Si ricorda che l'esponente va rappresentato su 8bit e la mantissa su 23bit.

Esercizio 3. Si consideri la funzione booleana $F(A,B,C,D) = A\overline{B}C + (CB + AD)\overline{(A+B)} + \overline{(BC)}$. Ridurre F in forma minima riportando i passaggi, e disegnare il circuito che la implementa.

Esercizio 4. Progettare un confrontatore di numeri interi con segno rappresentati su 3 bit in codice eccesso 2^{k-1} con k=3 usando per la realizzazione circuitale un comparatore di interi senza segno a 2bit, ed eventuali ulteriori blocchi di libreria.

Esercizio 5. Progettare una ALU che permetta di eseguire le seguenti operazioni sugli ingressi A e B: somma A+B, differenza A-B, B<<2, B<<1, interpretando i valori sugli ingressi come interi con segno con codifica cp2. L'ALU genera il condition code V, che notifica eventuali overflow. Gli ingressi A e B e l'uscita U sono rappresentati su 4 bit, mentre il codice dell'operazione da eseguire è rappresentato su 2 bit.

Esercizio 1. Sia F una funzione che riceve in ingresso un numero intero n rappresentato su 4 bit in codice eccesso 2^(k-1) con k=4. F assume valore 1 quando n vale -4, -1, 2, 4, 5, o 7 e può assumere indifferentemente il valore 1 o 0 quando n vale -2, 0, 3, o 6. F restituisce 0 per gli altri valori di n.

1. Realizzare il circuito che implementa F usando le mappe di Karnaugh, sintetizzando in forma SoP. Riportare i passaggi e disegnare il circuito derivato.
2. Realizzare un circuito equivalente a quello derivato al punto 1 usando solo porte NAND.

Soluzione.

La funzione F è definita dalla seguente tabella di verità, che riporta per ogni valore di N rappresentabile su 4 bit in cp2, la relativa codifica, ed il valore restituito da F.

N	*A*	*B*	*C*	*D*	*F*
-8	0	0	0	0	0
-7	0	0	0	1	0
-6	0	0	1	0	0
-5	0	0	1	1	0
-4	0	1	0	0	1
-3	0	1	0	1	0
-2	0	1	1	0	X
-1	0	1	1	1	1
0	1	0	0	0	X
1	1	0	0	1	0
2	1	0	1	0	1
3	1	0	1	1	X
4	1	1	0	0	1
5	1	1	0	1	1
6	1	1	1	0	X
7	1	1	1	1	1

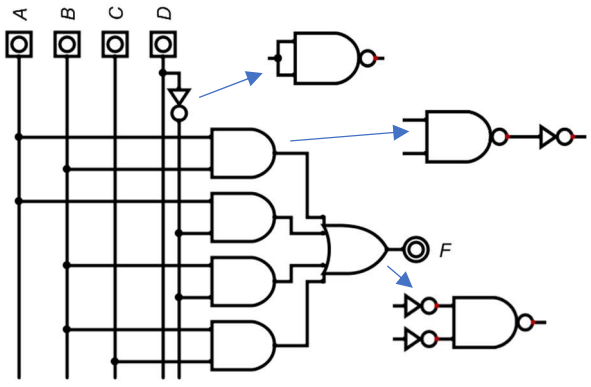
AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	1	X
11	1	1	1	X
10	X	0	X	1

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	1	X
11	1	1	1	X
10	X	0	X	1

La mappa di Karnaugh associata alla tabella ammette 2 soluzioni equivalenti:

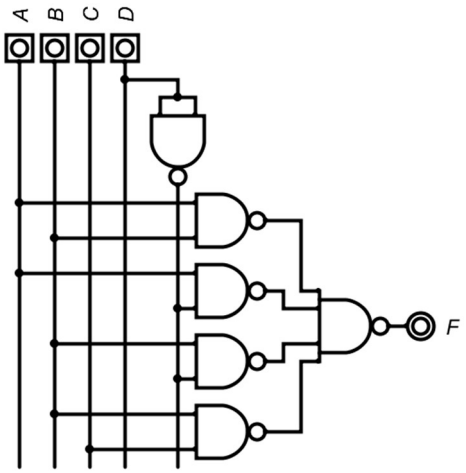
$F = AB + B\overline{D} + CB + AC$ $\qquad$ $F = AB + A\overline{D} + B\overline{D} + BC$

Implementando la prima soluzione otteniamo il seguente circuito:



Sostituiamo le porte AND, OR e NOT con composizioni di porte NAND che implementano il prodotto logico, la somma logica e l'inversione.

Dopo aver rimosso le doppie negazioni otteniamo il seguente circuito implementato usando solo porte NAND.



Esercizio 2. Derivare la codifica floating point IEEE 754 in singola precisione del numero -432,3125. Si ricorda che l'esponente va rappresentato su 8bit e la mantissa su 23bit.

Soluzione.

Segno: (-) → bit segno =1

Modulo parte intera= 432 → 110110000

432	0
216	0
108	0
54	0
27	1
13	1
6	0
3	1
1	1
0	

Parte decimale: ,3125 → 0101

	,3125
0	,625
1	,25
0	,5
1	,0

432,3125 → 110110000,0101 normalizzando otteniamo: $1{,}101100000101 * 2^8$

Rappresento l'esponente 8 in codice eccesso 127 su 8 bit.

8+127= 135 → 10000111

135	1
67	1
33	1
16	0
8	0
4	0
2	0
1	1
0	

Pertanto la codifica IEEE 754 di 432,3125 risulta: 1 10000111 10110000010100000000000

Esercizio 3. Si consideri la funzione booleana $F(A,B,C,D) = A\overline{B}C + (CB + AD)\overline{(A+B)} + \overline{(BC)}$. Ridurre F in forma minima riportando i passaggi, e disegnare il circuito che la implementa.

Soluzione.

$F = A\overline{B}C + (CB + AD)\overline{(A+B)} + \overline{(BC)} =$

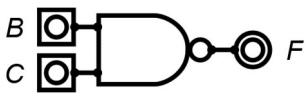
$= A\overline{B}C + (CB + AD)\overline{A}\,\overline{B} + \overline{B} + \overline{C}$ *(De Morgan)*

$= A\overline{B}C + \left(CB\overline{A}\,\overline{B} + AD\overline{A}\,\overline{B}\right) + \overline{B} + \overline{C}$ *(Distributiva)*

$= A\overline{B}C + \overline{B} + \overline{C}$ *(Inverso)*

$= \overline{B} + \overline{C}$ *(Assorbimento)*

$= \overline{BC}$ *(De Morgan)*



Esercizio 4. Progettare un confrontatore di numeri interi con segno rappresentati su 3 bit in codice eccesso 2^{k-1} con k=3 usando per la realizzazione circuitale un comparatore di interi senza segno a 2bit, ed eventuali ulteriori blocchi di libreria.

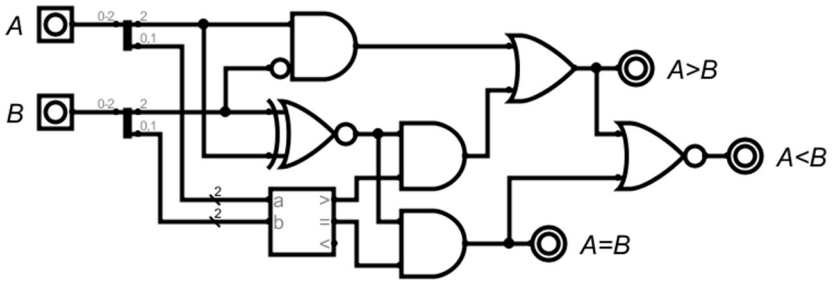
Soluzione.

Riferiamo con A e B gli ingressi su 3 bit del circuito, e con A>B, A=B, e A<B le uscite. Il circuito integra un confrontatore di interi senza segno a 2 bit che opera sui 2 bit meno significativi dei fasci di A e B.

L'uscita A=B è asserita quando i bit di pari posizione nei fasci di A e B denotano lo stesso valore. Più precisamente, A=B sse $A_2=B_2$ e dalla comparazione dei 2 bit meno significativi di A e B risulta che $A_1A_0=B_1B_0$.

A>B quando A denota un valore positivo e B negativo, ovvero quando $A_2=1$ e $B_2=0$, oppure quando $A_2=B_2$ e dalla comparazione dei 2 bit meno significativi di A e B risulta che $A_1A_0>B_1B_0$

Infine, A<B sse nessuna delle altre uscite è asserita.



Esercizio 5. Progettare una ALU che permetta di eseguire le seguenti operazioni sugli ingressi A e B: somma A+B, differenza A-B, B<<2, e B<<1, dove i valori sugli ingressi sono da interpretare come interi con segno con codifica cp2. L'ALU genera il condition code V, che notifica eventuali overflow. Gli ingressi A e B e l'uscita U sono rappresentati su 4 bit, mentre il codice dell'operazione da eseguire è rappresentato su 2 bit.

Soluzione.

Associamo un codice comando ad ogni operazione, e per minimizzare il numero di blocchi esprimiamo tutte le operazioni come somme usando un unico sommatore a 4 bit.

F=00 → A+B F=01 → A+cp1(B)+1 F=10 → 0+(B<<1) F=11 → 0+(B<<2)

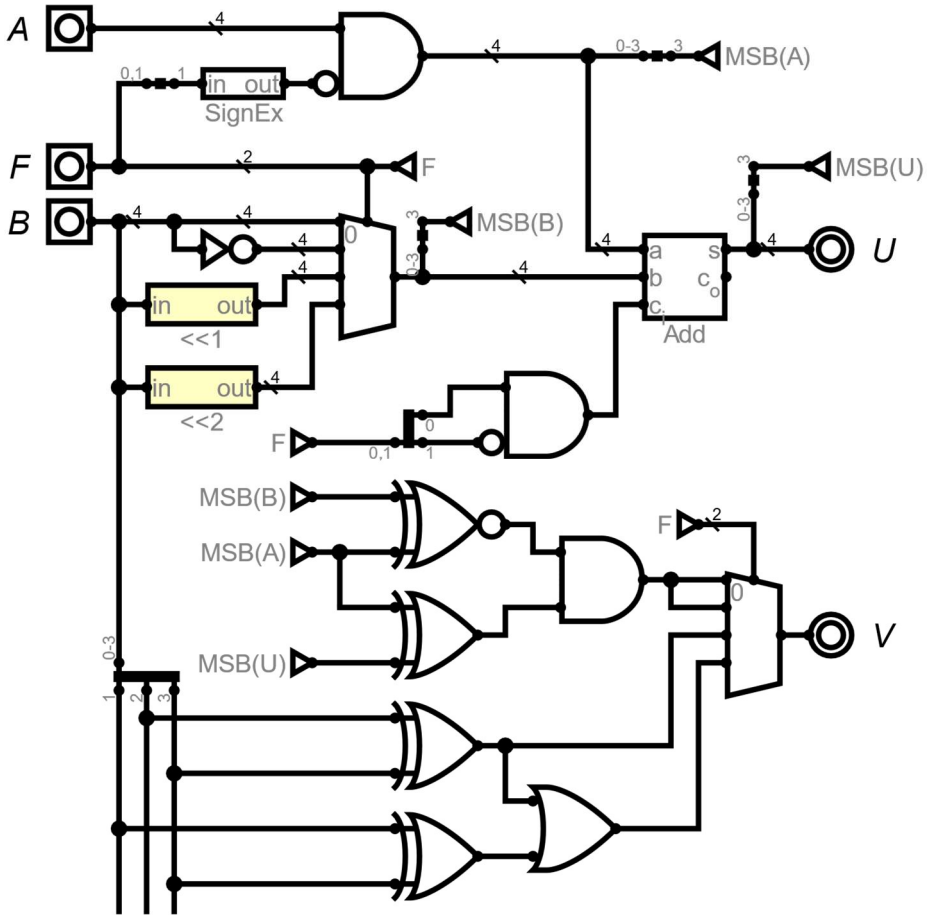
Il primo addendo di ogni somma corrisponde ad A quando MSB(F)=0, e a 0 quando MSB(F)=1. Per ottenere il primo addendo posso pertanto estendere in segno MSB(F), su 4 bit, invertirlo, e congiungerlo ad A.

In base ad F, il secondo addendo può corrispondere a B, cp1(B), B<<1, o B<<2. La scelta del secondo operando avviene per mezzo di un MUX a 4 bit, con F collegato all'ingresso di selezione a 2 bit del MUX.

Infine, l'operazione A-B (F=01) richiede di completare l'inversione di segno di B sommando 1. Sfruttiamo a tal fine l'ingresso C_{in} del sommatore. C_{in} vale 1 solo quando $F_1=0$ e $F_0=1$.

Il controllo dell'overflow avviene in base dell'operazione eseguita. Con le operazioni A+B e A+cp1(B)+1 (ovvero con F=00 e F=01) si verifica overflow quando i MSB dei due operandi sono tra loro uguali, ma diversi dal MSB della somma.

Nell'esecuzione di B<<1 (F=10) si genera overflow quando i due bit più significativi di B specificano un valore differente ($B_3!=B_2$). Infine, durante l'esecuzione di B<<2 (F=11) si verifica overflow quando ($B_3!=B_2$) oppure quando ($B_3!=B_1$).


A
F
B
SignEx
in out
<<1
<<2
MSB(A)
MSB(B)
MSB(U)
F
U
V
Add
a
b
c
s
c
o