UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

Prova scritta del 9 Gennaio 2024

Cognome e nome Matricola

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 0 quando n vale -6, -4, 4 o 5 e assume il valore 1 quando n vale -7, -2, -1, 1, 3 o 6. F restituisce indifferentemente 0 o 1 per gli altri valori di n.

a. Realizzare il circuito che implementa F usando le mappe di Karnaugh, sintetizzando in forma PoS. Riportare i passaggi e disegnare il circuito derivato.
b. Realizzare un circuito equivalente a quello derivato al punto a) usando solo porte NOR.

Soluzione:

La funzione F è definita dalla seguente tabella di verità, che riporta per ogni valore di N la relativa rappresentazione in codice eccesso 2^{k-1} con k=4, ed il valore restituito da F.

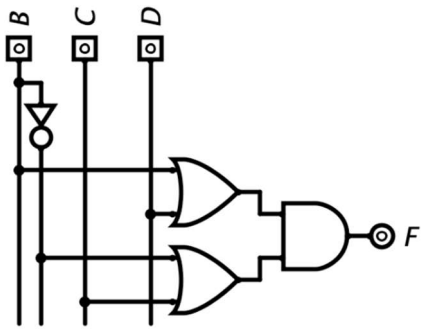
N	A	B	C	D	F
-8	0	0	0	0	X
-7	0	0	0	1	1
-6	0	0	1	0	0
-5	0	0	1	1	X
-4	0	1	0	0	0
-3	0	1	0	1	X
-2	0	1	1	0	1
-1	0	1	1	1	1
0	1	0	0	0	X
1	1	0	0	1	1
2	1	0	1	0	X
3	1	0	1	1	1
4	1	1	0	0	0
5	1	1	0	1	0
6	1	1	1	0	1
7	1	1	1	1	X

Dalla tabella si deriva la seguente mappa di Karnaugh:

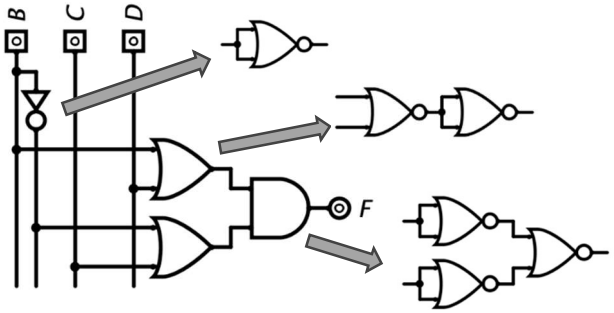
AB \ CD	00	01	11	10
00	X	1	X	0
01	0	X	1	1
11	0	0	X	1
10	X	1	1	X

$$F = (B + D)\left(\overline{B} + C\right)$$

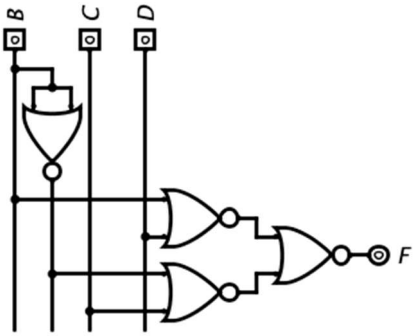
Il circuito che implementa F è il seguente:



Sostituiamo le porte AND, OR e NOT con composizioni di porte NOR 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 NOR.



Esercizio 2.

Derivare il valore decimale del numero N rappresentato con la seguente codifica floating point IEEE 754 in singola precisione: 1 10000101 11110000000000000000000

Soluzione:

Bit di segno (1): segno (-)

Esponente 10000101 in codice eccesso 127: (128+4+1)-127= 133-127=6

Il numero è in forma normalizzata, pertanto la mantissa è implicitamente dotata di parte intera valorizzata 1. La mantissa pertanto vale: 1,1111

N vale (-1)* 1,1111 * 2^6. Spostando la posizione della virgola di 6 posizione verso destra ottengo: (-1) 1111100= (-1)* $2^6+2^5+2^4+2^3+2^2$ = (-1)* (64+32+16+8+4)= -124

Esercizio 3.

Realizzare una ALU dotata di un ingresso A, e di un'uscita U, entrambi su 8 bit con rappresentazione cp2, e che in base ad un ingresso F gestisce le operazioni: -1-A, A<<1, e A mod 2. La ALU genera un bit di esito V, che notifica eventuali overflow.

Soluzione:

Specifichiamo i valori di F per cui eseguire le varie operazioni, servono 2 bit.

1. F=00: -1-A
2. F=01: A <<1
3. F=10 e 11: A mod 2

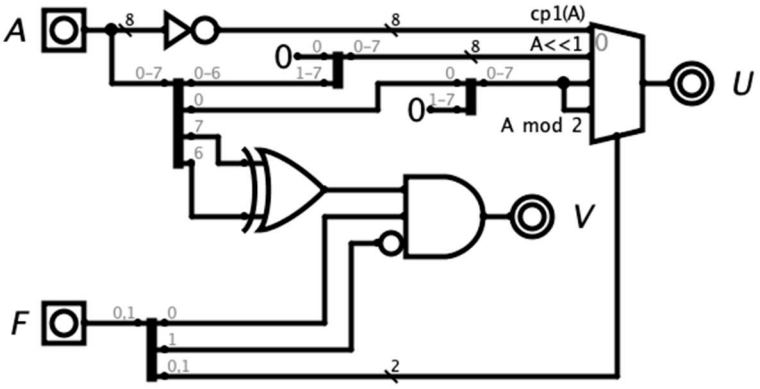
Non serve un sommatore per gestire -1-A (F=00). Infatti, -1-A è esprimibile come cp1(A) -1 +1, che equivale a cp1(A).

Inoltre, relativamente all'operazione A mod 2, osservo che il resto della divisione di A per 2 corrisponde al LSB(A) esteso a 0 su 8 bit.

Per regolare il valore da inoltrare in uscita uso un MUX con ingresso di selezione su 2 bit che assume il valore di F, e ingressi dati su 8 bit

L'unica operazione che può generare overflow è l'operazione A<<1. Più precisamente, genera overflow se il bit con indice 7 e 6 di A denotano valori differenti. Pertanto, derivo che V=1 sse (F=01) AND (B7 XOR B6).

Il circuito risulta pertanto:



Esercizio 4.

Progettare un circuito sequenziale sincrono dotato di un ingresso I da 1 bit. Sul fronte di discesa di ogni ciclo di clock il circuito memorizza il segnale presente su I. Il circuito genera un segnale O che indica se gli ultimi valori memorizzati sono 1111 (O=1 sse è stata identificata la sequenza 1111).

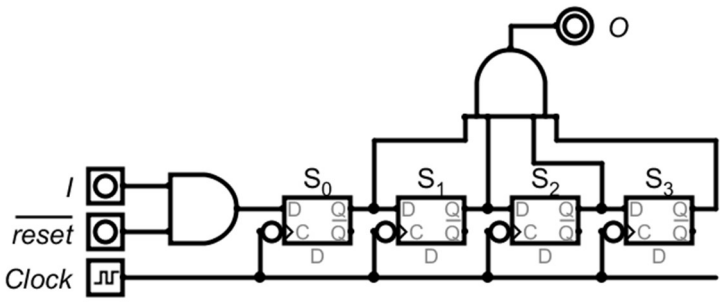
Per la realizzazione circuitale è necessario usare Flip-Flop D. E' possibile mostrare uno schema circuitale ad alto livello di astrazione, rappresentando come blocchi le reti combinatorie che compongono il circuito, ma è necessario specificare le funzioni logiche implementate da tali reti.

Soluzione:

Uso un registro a scorrimento da 4 bit sincrono sul fronte di discesa e con ingresso seriale. S_0, S_1, S_2 ed S_3 rappresentano l'ultimo, il penultimo, il terzultimo e il quartultimo valore memorizzato.

O=1 sse $S_3 S_2 S_1 S_0$

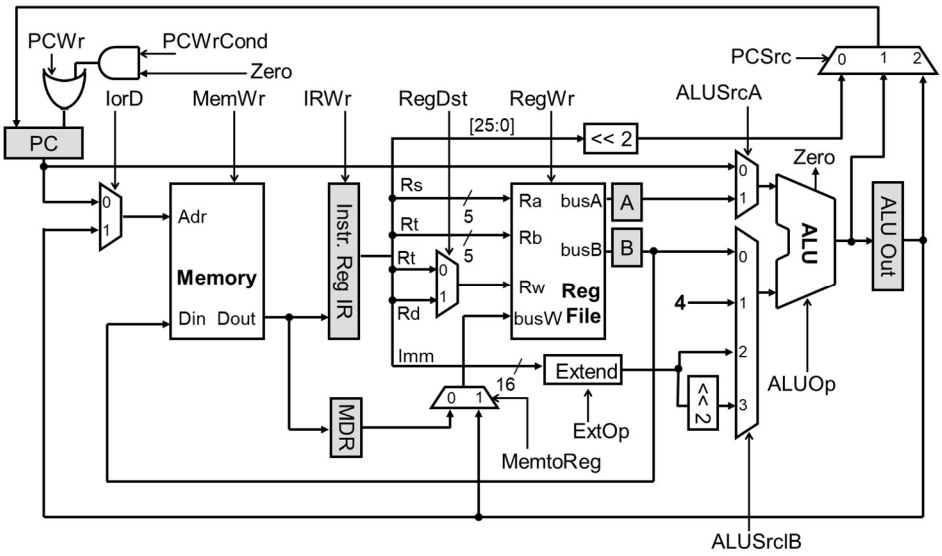
Gestisco l'inizializzazione del circuito con un ingresso reset attivo basso, è suffiente operare sul flip-flop S_0 connesso ad I.



Esercizio 5.

Facendo riferimento al datapath multiciclo del MIPS in figura, descrivere brevemente le fasi del ciclo di esecuzione dell'istruzione: `sw rt, rs, imm16` indicando, per ogni fase d'esecuzione l'uso che viene fatto dei registri, e motivando le risposte date.

Si ricorda che l'istruzione *sw* (store word) copia in memoria la parola presente nel registro *rt* del register file. La copia viene effettuata all'indirizzo denotato dalla somma tra il contenuto del registro *rs* ed *imm16* esteso in segno.



Fase	**Trasferimenti tra registri**	**Motivo** *(breve spiegazione, max 3 righe)*
1)	IR ← Mem[PC] PC ← PC+4	…
2)	A ← R[IR[rs]] B ← R[IR[rt]]	
3)	ALUout ←A + SignExt(IR[imm16])	
4)	Mem[ALUout] ← B	
5)		

Esercizio 6. Si consideri un sistema di memoria composto da una memoria di lavoro di 256 MB, e da una cache di 512KB. Il sistema gestisce blocchi di 256 parole di 64 bit. Assumendo che la memoria sia indirizzata a livello di byte, si chiede di indicare la struttura degli indirizzi in scenari dove la cache impiegata è: a) una cache a indirizzamento diretto, b) una cache completamente associativa, e c) una cache set-associativa a 8 vie. In particolare, indicare quali bit di un indirizzo denotano i tag, quali gli indici di linea (se presenti), e quali il byte offset nei blocchi. Riportare i procedimenti di calcolo seguiti.

Soluzione.

Mem size: 256 MB= 2^{28} byte → Indirizzi a livello di byte su 28 bit ($\log_2$ Mem_size)

Block size: 256*8 byte=2^{11} byte → Byte offset a livello di blocco su 11 bit

Cache size: 512 KB= 2^{19} byte

a) Num linee= 2^{19} byte/2^{11} byte=2^8 → servono 8 bit per specificare l'indice di linea
Tag su 9 bit *(28 – (8 + 11))*
Schema indirizzo: 9 bit *Tag* | 8 bit *Line idx* | 11 bit *Byte offset*

b) Tag riferito su 17 bit *(28 – 11)*
Schema indirizzo: 17 bit *Tag* | 11 bit *Byte offset*

c) Set size=8*2^{11} byte= 2^{14} byte
Num set=2^{19} byte/2^{14} byte=2^5 → servono 5 bit per specificare l'indice del set
Tag riferito su 12 bit *(28 – (5 + 11))*
Schema indirizzo: 12 bit *Tag* | 5bit *Set idx* | 11 bit *Byte offset*