UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

Prova scritta del 9-2-2023

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 -4,-2,-1, 1, 3, 5, 6 o 7 e può assumere indifferentemente il valore 1 o 0 quando n vale -7, -6, o 2. F restituisce 1 per gli altri valori di n. Realizzare il circuito che implementa F usando le mappe di Karnaugh, e sintetizzando in forma SoP. Riportare i passaggi e disegnare il circuito derivato.

Soluzione:

La funzione F è definita dalla seguente tabella di verità, che riporta per ogni valore di N la relativa codifica in codice eccesso, ed il valore restituito da F.

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

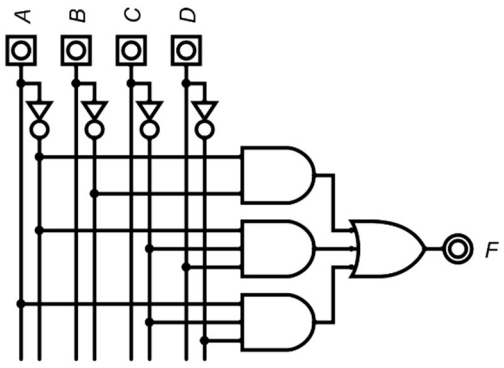
Dalla tabella si deriva la mappa di Karnaugh

F:

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

Da cui: $F = (\overline{A}\,\overline{B}) + (\overline{A}\,\overline{C}\,D) + (A\,\overline{C}\,\overline{D})$

Il circuito risulta pertanto:



Esercizio 2. Consideriamo una funzione booleana *F* definita come segue:

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

Ridurre F in forma minima mostrando e giustificando i passaggi, e disegnare il circuito che la implementa.

Soluzione:

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

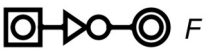
$= \overline{C}\,(A + B + C)\left(A + B + \overline{C}\right) + \overline{C}\,\overline{A} + \overline{C}\,\overline{B}$ (per assorbimento $B + C\overline{B} = B + C,\ A + \overline{A}B = A + B$)

$= \overline{C}\,(A + B) + \overline{C}\,\overline{A} + \overline{C}\,\overline{B}$ (per proprietà distributiva e inverso: $A + B + C\overline{C} = A + B$)

$= \overline{C}A + \overline{C}B + \overline{C}\,\overline{A} + \overline{C}\,\overline{B}$

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

$= \overline{C}$ (per inverso ed elemento nullo)

Il circuito risultante è: 

Esercizio 3. Indicare la codifica floating point IEEE 754 in singola precisione del numero -41,5625. Si ricorda che l'esponente va rappresentato su 8bit e la mantissa su 23bit. Riportare il procedimento di calcolo seguito.

Soluzione:

Segno: (-) → bit segno =1

Modulo parte intera= 41 → 101001

41	1
20	0
10	0
5	1
2	0
1	1
0	

Parte decimale: ,5625 → ,101

	,5625
1	,125
0	,25
0	,5
1	,0

41,5625 → 101001,1001 normalizzando otteniamo: $1{,}010011001 * 2^5$

Rappresento l'esponente 5 in codice eccesso 127 su 8 bit.

5+127= 132 → 10000100

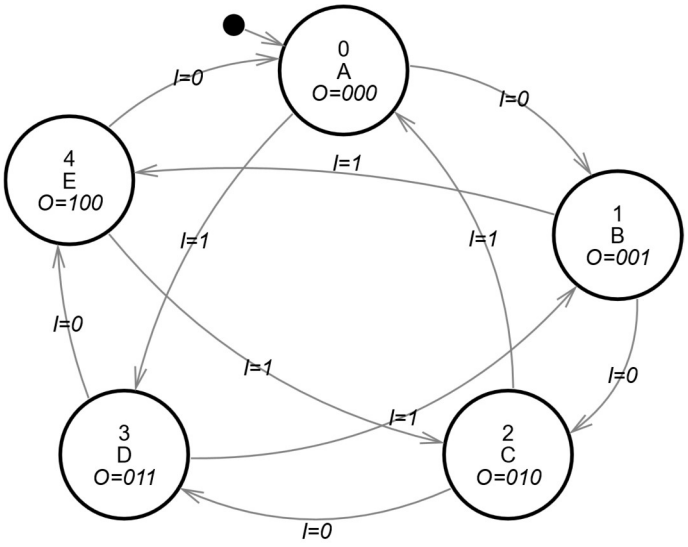
132	0
66	0
33	1
16	0
8	0
4	0
2	0
1	1
0	

Pertanto, la codifica IEEE 754 risulta: 1 10000101 01001100100000000000000

Esercizio 4. Progettare un circuito contatore sincrono modulo 5 che conta in avanti di 1 o di 3 in base ad un segnale di ingresso booleano I. Più precisamente, denotato V il valore del contatore, se I=0, V'=(V+1) mod 5, mentre, se I=1, V'=(V+3) mod 5. L'operazione di incremento è eseguita ad ogni ciclo di clock. Il circuito restituisce il valore memorizzato su un'uscita O su 3 bit. Realizzare il circuito impiegando la codifica di stato a singolo 1 (un solo bit per stato vale 1). Per la realizzazione circuitale è necessario usare Flip-Flop D, e possono essere usati blocchi funzionali di libreria.

Soluzione:

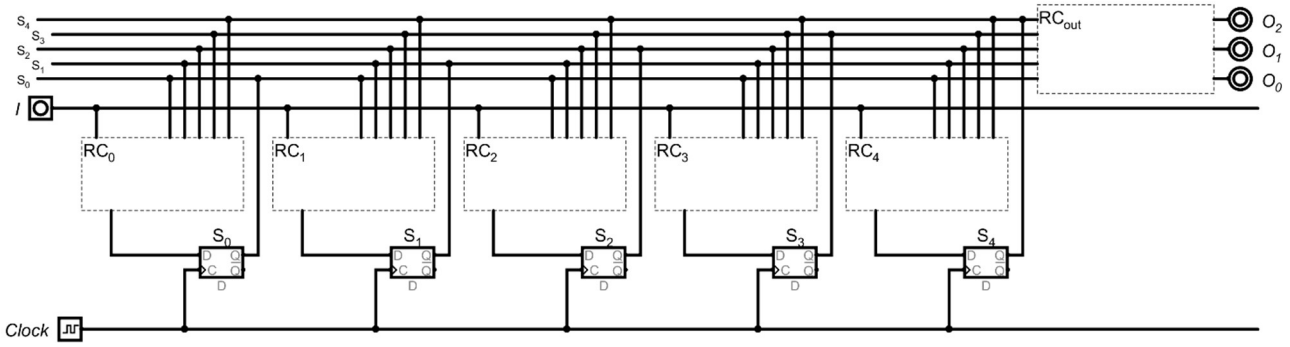
Il comportamento del circuito è modellato dalla FSM a fianco.



Ogni stato denota un valore di V (da 0 a 4), e riporta come segnali da inoltrare sull'uscita O la codifica binaria pura di V su 3 bit.

Con la codifica a singolo 1 gli stati sono codificati su 5 bit come segue: A: 00001, B: 00010, C: 00100, D: 01000, ed E: 10000

Lo schema di riferimento per la realizzazione del circuito sarà il seguente:



La rete combinatoria RC_{out} che calcola le uscite è definita partendo dalla seguente tabella di verità:

S_4	S_3	S_2	S_1	S_0	O_2	O_1	O_0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
1	0	0	0	0	1	0	0

da cui, sfruttando la codifica a singolo 1:

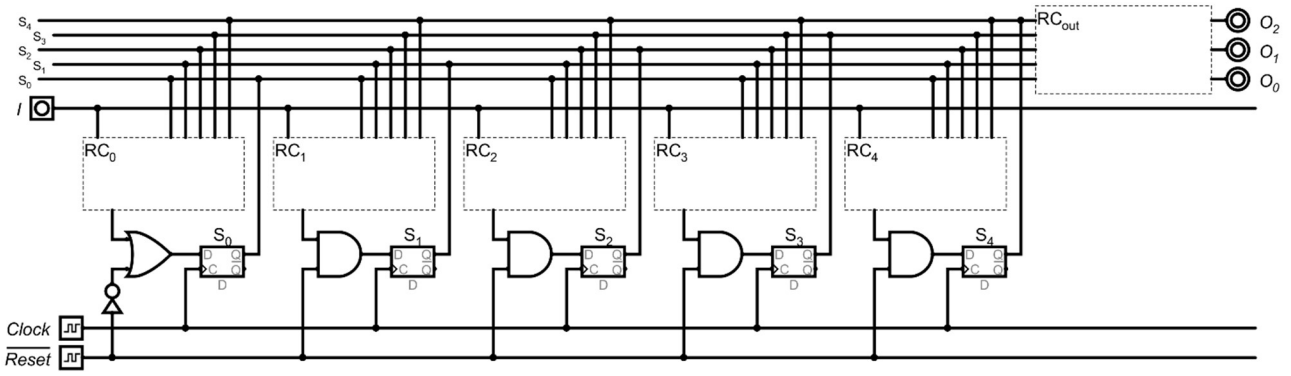
$O_2 = S_4$ $O_1 = S_2+S_3$ $O_0 = S_1+S_3$

Le reti combinatorie che gestiscono le transizioni di stato sono:

RC_0: $S_0' = 1 \Leftrightarrow S_4\overline{I} + S_2 I$ RC_1: $S_1' = 1 \Leftrightarrow S_0\overline{I} + S_3 I$ RC_2: $S_2' = 1 \Leftrightarrow S_1\overline{I} + S_4 I$

RC_3: $S_3' = 1 \Leftrightarrow S_2\overline{I} + S_0 I$ RC_4: $S_4' = 1 \Leftrightarrow S_3\overline{I} + S_1 I$

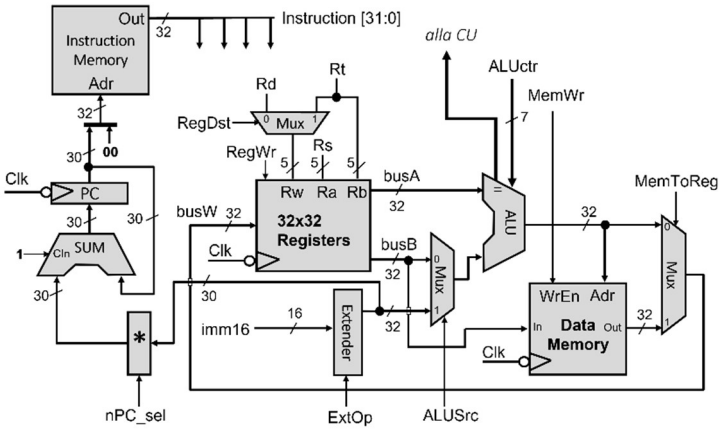
Infine, per far si che il circuito si comporti correttamente usiamo un segnale di Reset attivo basso per impostare lo stato iniziale:



Esercizio 5. Facendo riferimento al datapath a singolo ciclo del MIPS riportato in figura, si chiede di indicare nella seguente tabella il valore dei segnali scambiati tra l'unità di controllo e le componenti del datapath durante l'esecuzione dell'istruzione *andi rt, rs, imm16* (and immediate) giustificando la risposta data.

La semantica di *andi rt, rs, imm16* in RTL è definita come segue:

R[rt]← R[rs]*and* ZeroExt(imm16)
PC ← PC+4



Segnale	Valore	Motivo (breve spiegazione, max 3 righe)
RegDst	1	…
RegWr	1	
ALUctr	And	
MemWr	0	
MemToReg	0	
ALUSrc	1	
ExtOp	0 (zero)	
nPC_sel	0	

Esercizio 5. Si consideri un sistema di memoria caratterizzato da una memoria di lavoro di 8 KB indirizzata a livello di Byte, e una cache set associativa a 4 vie. La cache gestisce 8 blocchi, riferiti A, B, C, D, E, F, G e H, ripartiti in tal ordine tra i set, seguendo l'ordine crescente dell'indice del set. Ogni blocco è composto da 32 parole da 64 bit. Considerando la sequenza di richieste alla memoria riportata in tabella, e assumendo per la gestione dei blocchi in cache la politica LRU, si chiede di completare la tabella che illustra il comportamento della cache nel rispetto delle indicazioni seguenti:

- Nella colonna "esito" riportare H (hit) se il blocco richiesto si trova nella cache, M (miss) se invece il blocco deve essere caricato dalla memoria.
- Nelle colonne "dati" deve essere riportato l'indice del blocco della memoria (in decimale), presente nel corrispondente blocco della cache.
- Nella colonna "azione indicare il blocco a cui si accede in caso di hit, o in cui si caricano i dati della memoria in caso di miss e l'indice del blocco caricato (in decimale).

		Blocco A			**Blocco B**			**Blocco C**			**Blocco D**			**Blocco E**			**Blocco F**			**Blocco G**			**Blocco H**			
	Esito	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Valido**	**Etichetta**	**Dati**	**Azione**
Situazione iniziale		0			1	1101	26	1	1001	18	1	1000	16	0	1001	19	1	0000	1	1	1011	23	1	1000	17	--
1) Richiesta *addr* 1001 1 11100010	M													1	1001	19										Carico in E (set 1) il blocco 19
2) Richiesta *addr* 1000 0 10001011	H																									Accedo al blocco D nel set 0
3) Richiesta *addr* 0001 0 10010011	M	1	0001	2																						Carico in A (set 0) il blocco 2
4) Richiesta *addr* 1001 0 01111011	H																									Accedo al blocco C nel set 0

Mem size: 8KB (2^{13} *Byte*) → indirizzamento a livello di byte su 13 bit;
Block size: 32 word * 64 bit= 256 byte (2^5*2^3 *byte*) → byte offset nel blocco su 8 bit
Cache size (senza tag): 8*256 byte = 2KB (2^{11} *byte*);
Set size= 1KB (*4 * 256 byte*);
Num set: 2KB/1KB=2 → set index su 1 bit;
Tag su 4 bit (*13-(8+1)*);
Scomposizione indirizzo memoria: 4bit tag + 1bit set idx + 8bit byte offset;
Indice del blocco in memoria riferito dai 5 bit più significativi (*i bit di tag e set idx*).