UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

ARCHITETTURE DEGLI ELABORATORI

Prova scritta del 26-1-2023

Cognome e nome Matricola

Esercizio 1.

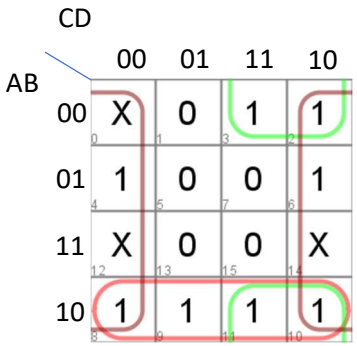
Sia F una funzione che riceve in ingresso un numero intero n rappresentato su 4 bit in codice eccesso 2^{k-1} con k=4. F assume valore 0 quando n vale -7,-3,-1, 5 o 7 e può assumere indifferentemente il valore 1 o 0 quando n vale -8, 4 o 6. F restituisce 1 per gli altri valori di n. Realizzare il circuito che implementa F usando le mappe di Karnaugh, 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.

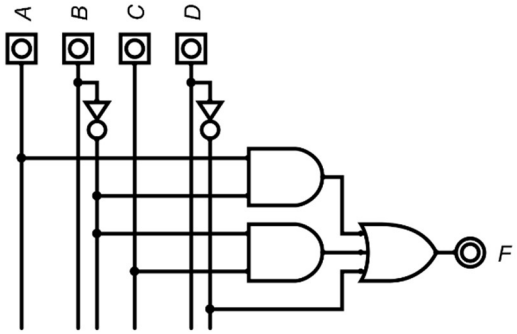
N	*A*	*B*	*C*	*D*	*F*
-8	0	0	0	0	X
-7	0	0	0	1	0
-6	0	0	1	0	1
-5	0	0	1	1	1
-4	0	1	0	0	1
-3	0	1	0	1	0
-2	0	1	1	0	1
-1	0	1	1	1	0
0	1	0	0	0	1
1	1	0	0	1	1
2	1	0	1	0	1
3	1	0	1	1	1
4	1	1	0	0	X
5	1	1	0	1	0
6	1	1	1	0	X
7	1	1	1	1	0

Dalla tabella si deriva la seguente mappa di Karnaugh:



F = A !B + C !B + !D

Il circuito risulta pertanto:



Esercizio 2.

Sia F una funzione booleana definita come segue: $F = B\left(A + \overline{D}C + D\right) + \overline{C} + AB$. Ridurre F in forma minima mostrando i passaggi, e mostrare il circuito che la implementa.

Soluzione:

$=B(A + C + D) + \overline{C} + AB$ (per assorbimento $\overline{D}C + D = C + D$)

$=AB + BC + BD + \overline{C}$ (proprietà distributiva e idempotenza $AB + AB = AB$)

$=AB + BD + B + \overline{C}$ (assorbimento $BC + \overline{C} = B + \overline{C}$)

$=(A + D + 1)B + \overline{C}$ (proprietà distributiva)

$=B + \overline{C}$ (elemento nullo)

Il circuito risultante è:

Esercizio 3.

Derivare il valore decimale del numero rappresentato dalla seguente codifica floating point IEEE 754 in singola precisione.

1	10001001	10011011001100000000000

Riportare il procedimento di calcolo seguito.

Soluzione:

Bit segno : 1 → segno (–)

Esponente: 10001001 → $2^7+2^3+2^0-127=128+8+1-127=10$

Mantissa: 1,100110110011 (parte intera implicita a 1, poiché il numero è in forma normalizzata)

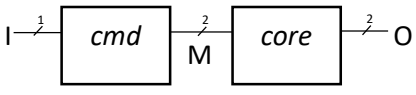
Il valore denotato in forma normalizzata in base 2 corrisponde a -1,100110110011 * 2^{10}

Sposto la virgola verso destra di 10 posizioni e derivo il valore decimale

-11001101100,11 = $(-1)(2^{10}+2^9+2^6+2^5+2^3+2^2+2^{-1}+2^{-2})$= -(1024+512+64+32+8+4+0.5+0.25)= -1644,75

Esercizio 4.

Si vuole realizzare un circuito sequenziale sincrono che gestisce una lampada led a 3 colori dotata di un pulsante di controllo. Il circuito è composto da 2 moduli, denotati *cmd*, e *core*, interconnessi come mostrato nel seguente schema.



Il modulo *cmd* è un circuito che elabora i segnali ricevuti dal pulsante di controllo, ed invia al modulo *core* segnali che codificano la richiesta di impostare un comportamento della lampada. Invece, il modulo *core* è un circuito che, in base alle richieste del modulo *cmd* e alla configurazione corrente dei led, genera segnali per comandare l'accensione di una luce colorata o lo spegnimento dei led. Più precisamente:

a. Il modulo *cmd* è un circuito sequenziale che implementa un contatore sincrono modulo 4. Il modulo *cmd* è dotato: i) di un ingresso I, che convoglia i segnali generati dal pulsante di controllo, e ii) di un'uscita M, su 2 bit, connessa all'ingresso del modulo *core*, su cui viene mandato il valore memorizzato dal contatore. Ad ogni ciclo di clock, alla pressione del pulsante (ovvero, quando I=1) viene incrementato il valore del contatore. I segnali inoltrati su M codificano la richiesta di impostare un comportamento della lampada.

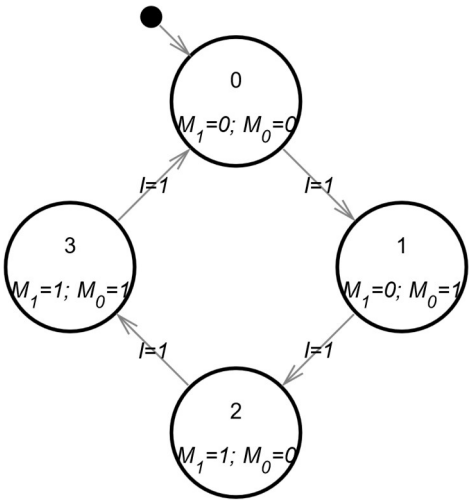
b. Il modulo *core* è un circuito sequenziale sincrono dotato: i) di un'uscita O su 2 bit tramite cui viene inoltrato al led il comando di accensione di una luce colorata o di spegnimento, e ii) di un ingresso M su 2 bit, su cui vengono indirizzate le richieste di impostare un comportamento.
   - Quando M vale 00 la lampada deve essere mantenuta spenta. Per implementare tale comportamento, *core* deve generare in uscita il segnale O=00.
   - Quando M vale 01, deve essere mantenuta accesa la luce di colore bianco. In questo caso, *core* deve generare in uscita O=01.
   - Quando M vale 10, ad ogni ciclo di clock deve essere impostato un colore diverso, seguendo la sequenza periodica: bianco, rosso, verde, e luci spente. Per implementare tale comportamento *core* dovrà cambiare ad ogni ciclo di clock i segnali in uscita: O=01 per il bianco, O=10 per il rosso, O=11 per il verde, e O=00 per le luci spente.
   - Quando M vale 11, deve essere mantenuta l'ultima configurazione impostata (lampada accesa con luce bianca, o rossa, o verde, oppure luci spente). In questo ultimo caso, *core* dovrà mantenere stabile il valore dei segnali già presenti su O alla ricezione di M=11.

Si chiede di progettare i 2 moduli mostrando i relativi schemi circuitali. È sufficiente denotare come blocchi le reti combinatorie che compongono i circuiti specificando le funzioni logiche implementate da tali reti. Per la realizzazione circuitale devono essere impiegati dei FlipFlop-D, e possono essere usati blocchi funzionali di libreria.

Soluzione:

a) Modulo *cmd*

Il circuito *cmd* si comporta come un contatore modulo 4 con comando di incremento. Il comportamento è modellato dalla seguente FSM. In ogni stato viene anche riferito il comando inoltrato sulle uscite M_1 e M_0 al modulo *core*.



La funzione di transizione è specificata dalla seguente tabella di verità.

I	S_1	S_0	S'_1	S'_0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

da cui si derivano le seguenti mappe di Karnaugh per il calcolo di S'_1 e S'_0

S'_1 S_1S_0

I \ S_1S_0	00	01	11	10
0	0	0	1	1
1	0	1	0	1

S'_1= !IS_1 + S_1!S_0 + I!S_1S_0

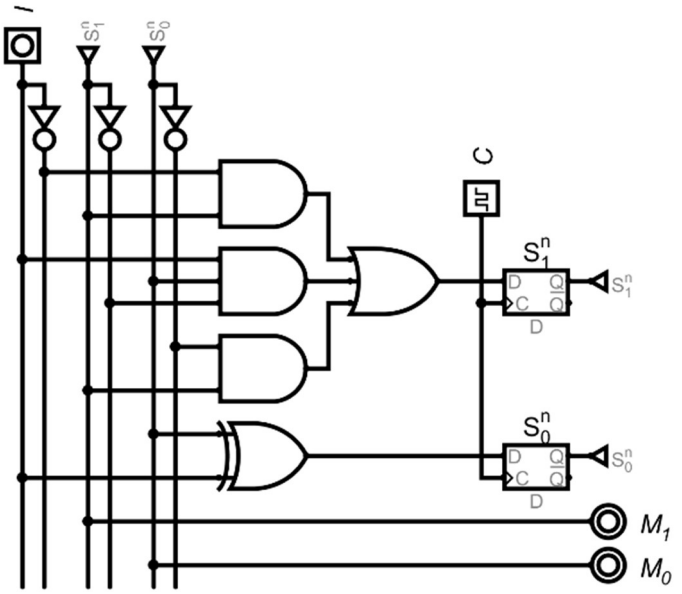
S'_0 S_1S_0

I \ S_1S_0	00	01	11	10
0	0	1	1	0
1	1	0	0	1

S'_0=!I S_0 + I !S_0 = I xor S_0

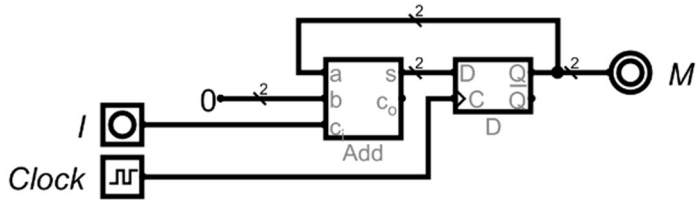
Per il calcolo delle uscite si osserva che: $O_1=S_1$ e $O_0=S_0$

Il circuito risultante è pertanto il seguente:



In alternativa allo svolgimento proposto sopra, *cmd* può essere facilmente realizzato usando un sommatore a 2 bit e un registro parallelo a 2 bit.

Quando I=1, ad ogni ciclo di clock il sommatore incrementa di 1 il valore memorizzato e propagato su M. Quando questo valore corrisponde a 11, l'incremento successivo porta a resettare il contatore memorizzando come nuovo valore 00 (l'incremento genera un traboccamento, visibile su c_{out}, che viene però ignorato; vengono pertanto memorizzati i 2 bit meno significativi del risultato atteso 100, ovvero 00). Di contro, quando I=0, il valore memorizzato non viene modificato.



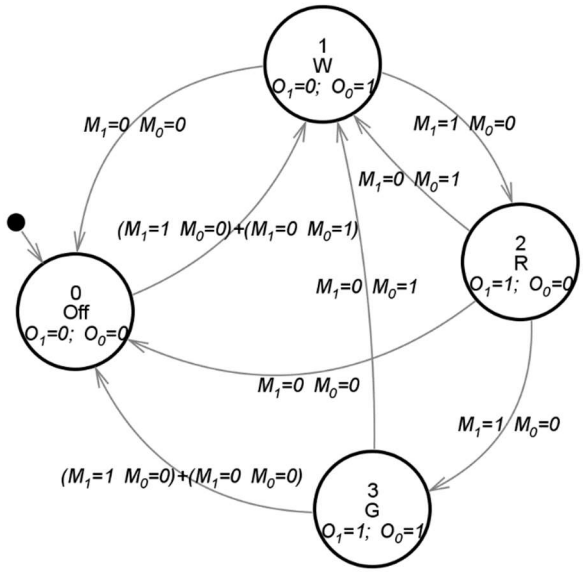
b) Modulo core

Il comportamento del circuito *core* è modellato dalla seguente FSM. Ogni stato denota una configurazione della lampada. Off (codificato 00), rappresenta lo stato in cui le luci sono spente, W (codificato 01), R (codificato 10), e G (codificato 11) sono stati in cui la luce accesa è rispettivamente di colore bianco, rosso e verde.

In ogni stato viene anche riferito il comando inoltrato sulle uscite O_1 e O_0 per comandare l'accensione o lo spegnimento delle luci colorate.

La funzione di transizione è definita dalla tabella di verità:

S_1	S_0	M_1	M_0	S'_1	S'_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	1	1



da cui si derivano le seguenti mappe di Karnaugh per il calcolo di S'_1 e S'_0

S'_1 M_1M_0

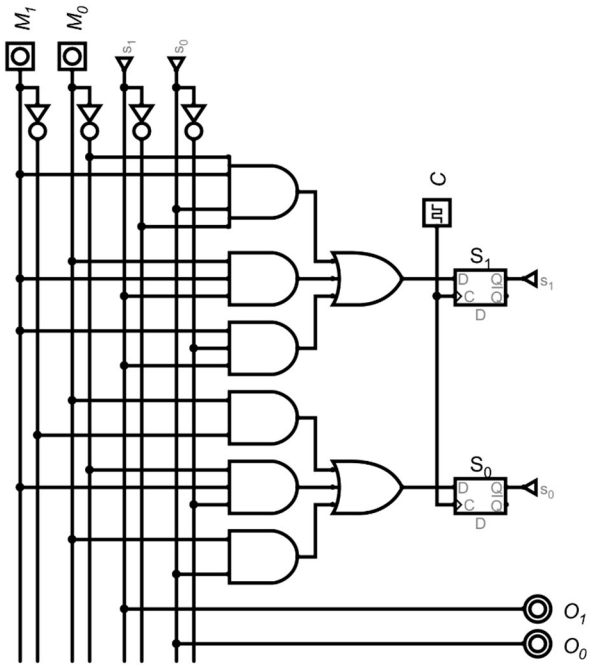
S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	1	0
10	0	0	1	1

S'_0 M_1M_0

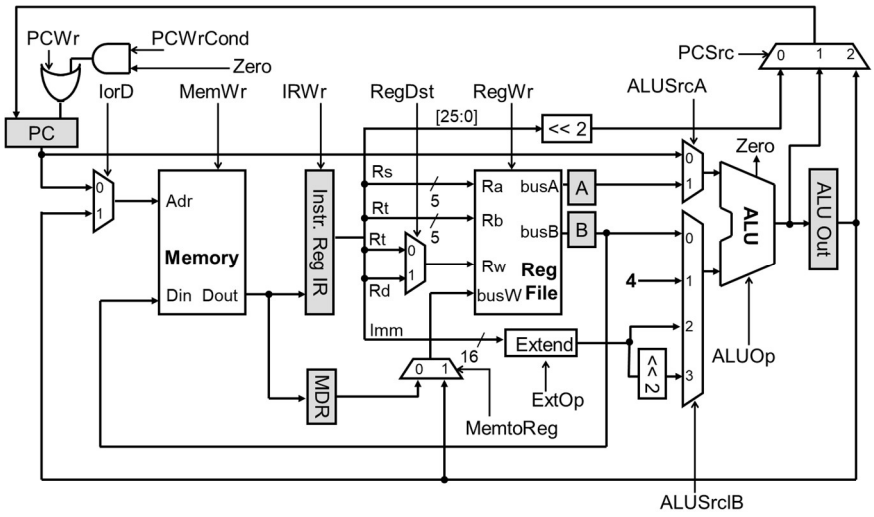
S_1S_0	00	01	11	10
00	0	1	0	1
01	0	1	1	0
11	0	1	1	0
10	0	1	0	1

$S'_1 = M_1S_1!S_0 + M_1M_0S_1 + M_1!M_0!S_1S_0$ $\quad$ $S'_0 = !M_1M_0 + M_1!M_0!S_0 + M_0S_0$

Per il calcolo delle uscite non serve specificare tabelle di verità, poiché in ogni stato della FSM proposta le uscite O_1 e O_0 denotano il bit più significativo e meno significativo della codifica di stato. Si osserva quindi che: $O_1=S_1$ e $O_0=S_0$. Il circuito risulta pertanto:



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 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[Im16])	…
4)	Mem[ALUOut] ← B	…
5)	--	

Esercizio 6.

Si consideri un sistema di memoria caratterizzato da una memoria di lavoro di 2 KB indirizzata a livello di Byte, e una cache ad indirizzamento diretto, inizialmente vuota, di dimensione 128 byte. Il sistema gestisce blocchi composti da 8 parole da 16 bit. Considerando la sequenza di richieste alla memoria riportata in tabella, 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).

		Linea 0			**Linea 1**			**Linea 2**			**Linea 3**			**Linea 4**			**Linea 5**			**Linea 6**			**Linea 7**			
	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**
1) Richiesta *addr* 0011 011 1001	M										1	0011	27													Carico il blocco 27 nella linea 3
2) Richiesta *addr* 0111 111 0001	M																						1	0111	63	Carico il blocco 63 nella linea 7
3) Richiesta *addr* 1011 001 1111	M				1	1011	89																			Carico il blocco 89 nella linea 1
4) Richiesta *addr* 1010 101 1110	M																1	1010	85							Carico il blocco 85 nella linea 5

Mem size: 2KB (2^{11} *byte*) → indirizzamento a livello di byte su 11 bit;
Block size: 8 word * 16 bit= 16 byte ($2^{3}*2$ *byte*) → byte offset nel blocco su 4 bit
Cache size (senza tag): 128 byte (2^{7} *byte*);
Num linee = num blocchi = 128/ 16 = 8 → line idx su 3 bit;
Tag su 4 bit (*11-(4+3)*);
Scomposizione indirizzo memoria: 4bit tag + 3bit line idx + 4bit byte offset; Indice del blocco in memoria riferito dai 7 bit più significativi (i bit di *tag e line idx*).