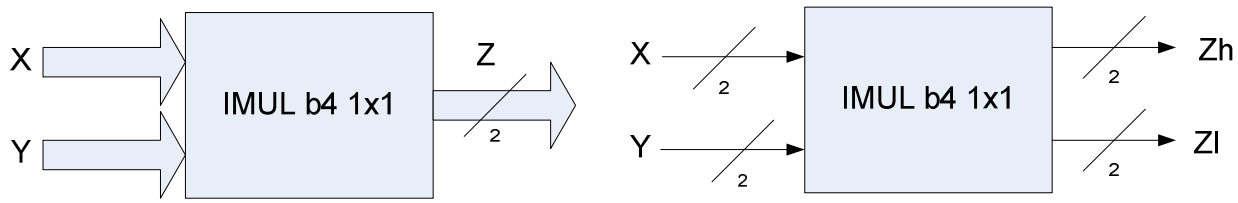
Esercizio 1
Descrivere e sintetizzare un *moltiplicatore ad 1 cifra in base 4 per interi in complemento alla radice*. Effettuare la sintesi a costo minimo delle uscite in forma SP. Individuare, classificare ed eliminare eventuali alee del 1° ordine. Calcolare il costo a porte della realizzazione priva di alee.

Soluzione
L'uscita di un moltiplicatore ad 1x1 cifra sta su 2 cifre in base 4. Ciascuna cifra in base 4 è codificata da due variabili logiche. La rete da sintetizzare è pertanto quella di figura, con 4 ingressi e 4 uscite.



L'intervallo di rappresentabilità per gli interi in complemento alla radice in base 4 su una cifra va da -2 a +1. Pertanto, la rete da sintetizzare è riassunta nella tabella sottostante, che ne include anche la mappa di Karnaugh (terza riga di ogni casella, in nero):

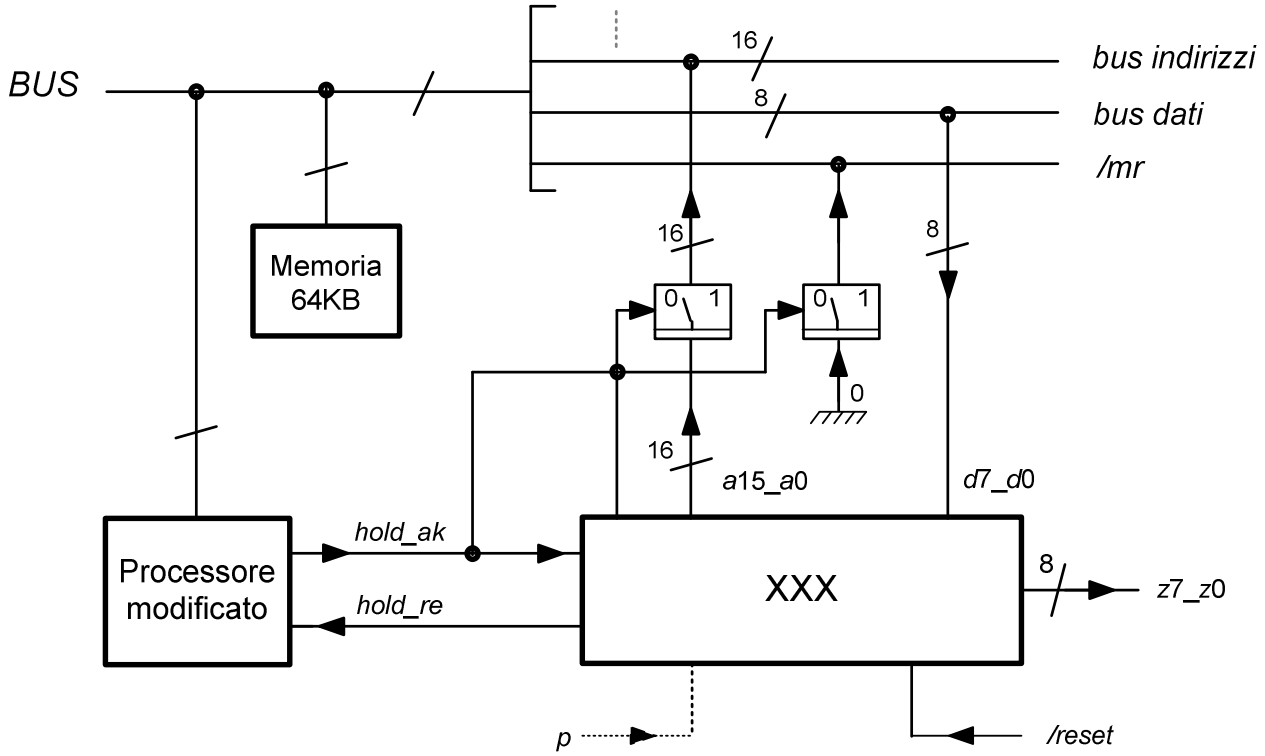
			0 0 00	+1 1 01	-1 3 11	-2 2 10	x X x_1x_0
0	0	00	0 0 0000	0 0 0000	0 0 0000	0 0 0000	
+1	1	01	0 0 0000	+1 01 0001	-1 33 1111	-2 32 1110	
-1	3	11	0 0 0000	-1 33 1111	+1 01 0001	+2 02 0010	
-2	2	10	0 0 0000	-2 32 1110	+2 02 0010	+4 10 0100	
y	Y			z ZhZl $z_3z_2z_1z_0$			

La sintesi a costo minimo in forma SP è la seguente:

$$z_3 = x_1 \cdot \overline{y_1} \cdot y_0 + y_1 \cdot \overline{x_1} \cdot x_0$$
$$z_2 = x_1 \cdot \overline{y_1} \cdot y_0 + y_1 \cdot \overline{x_1} \cdot x_0 + x_1 \cdot \overline{x_0} \cdot y_1 \cdot \overline{y_0}$$
$$z_1 = x_1 \cdot \overline{y_1} \cdot y_0 + y_1 \cdot \overline{x_1} \cdot x_0 + x_0 \cdot y_1 \cdot \overline{y_0} + y_0 \cdot x_1 \cdot \overline{x_0}$$
$$z_0 = x_0 \cdot y_0$$

La realizzazione di costo minimo è priva di alee del 1° ordine. Il costo a porte totale è pari a 9, in quanto gli implicanti usati per la sintesi di z_3 possono essere utilizzati per la sintesi di z_2, z_1.

Esercizio 2



Il processore visto a lezione è stato modificato nel seguente modo:

1) Indirizza una memoria lineare da 64Kbyte
2) Anche i piedini che ne permettono il collegamento al bus indirizzi e al bus per il comando di lettura in memoria sono supportati da porte a tre stati, cosicché il processore può isolarsi completamente anche da questi fasci di fili ponendo tali porte in alta impedenza;
3) È stata aggiunta la variabile di ingresso *hold_re* e la variabile di uscita *hold_ak*;
4) Quando *hold_re* viene messa a 1 dalle circuiterie esterne, il processore si **blocc**a, cioè si isola dal bus dati, dal bus indirizzi e dal bus per il comando di lettura in memoria, non compie alcuna evoluzione e notifica ciò ponendo *hold_ak* a 1; quando *hold_re* viene riportato a 0 il processore si **sblocca**, cioè rimuove il suo isolamento, notifica ciò ponendo *hold_ak* a 0 e riprende la sua normale evoluzione.

Descrivere e sintetizzare l'Unità *XXX* in modo che essa, partendo dalla condizione di reset iniziale emetta tramite z7_z0 il contenuto della memoria **ad un ritmo di un byte ogni 20 cicli di clock**. Per **ogni accesso** alla memoria, *XXX* si preoccupa di bloccare il processore e di sbloccarlo quando ha finito il ciclo di lettura del byte. Come si vede dalla Figura, quando *hold_ak* vale 0, l'Unità *XXX* non carica il bus.

Ipotesi semplificative: Si consideri il processore molto più veloce di *XXX* per cui il tempo in cui esso risponde ai segnali che gli giungono tramite *hold_re* è di pochi cicli del clock di *XXX*. Si consideri anche la memoria sufficientemente veloce da non richiedere cicli di wait. Si considerino pertanto 20 cicli di clock ampiamente sufficienti a *XXX* per effettuare tutte le operazioni richieste per la lettura e l'emissione di un byte.

Soluzione

```
module XXX(data,address,hold_re,hold_ak,z7_z0,p,reset_);
 input         p,reset_;
 input   [7:0] data;
 output [15:0] address;
 output        hold_re;
 input         hold_ak;
 output [7:0]  z7_z0;
 reg [15:0]    MAR;   assign   address=MAR;
 reg           HOLD_RE;  assign   hold_re=HOLD_RE;
 reg [7:0]     OUT;      assign   z7_z0=OUT;
 reg [7:0]     MBR;
 reg [3:0]     COUNT;
 reg [1:0]     STAR; parameter S0=0,S1=1,S2=2,S3=3;
 parameter Num_Periodi=10;

 always @(posedge p or negedge reset_)
  if (reset_==0) begin=0; MAR=0; COUNT=Num_Periodi; STAR<=S0; end
  else #3
   casex(STAR)
    S0:  begin COUNT<=COUNT-1; HOLD_RE<=1; STAR<=(hold_ak==0)?S0:S1; end
    S1:  begin COUNT<=COUNT-1; MBR<=data; HOLD_RE<=0; MAR<=MAR +1; STAR<=S2; end
    S2:  begin COUNT<=COUNT-1; STAR<=(hold_ak==1)?S2:S3; end
    S3:  begin COUNT<=(COUNT==1)?Num_Periodi:(COUNT-1); OUT<=(COUNT==1)?MBR:OUT;
         STAR<=(COUNT==1)?S0:S3; end
 endcase
endmodule
```