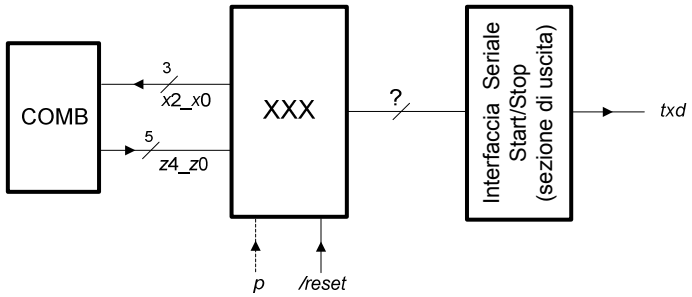
Esercizio 1
Sia dato un numero intero *x*. Descrivere e sintetizzare la rete che prende in ingresso la rappresentazione di *x* in complemento alla radice su 16 bit in base 2 e produce in uscita la rappresentazione in modulo e segno di *x* in base 10, secondo la codifica BCD. Descrivere esplicitamente qualunque rete non presentata a lezione.

Esercizio 2
Descrivere e sintetizzare l'unità *XXX* avente le specifiche che seguono.



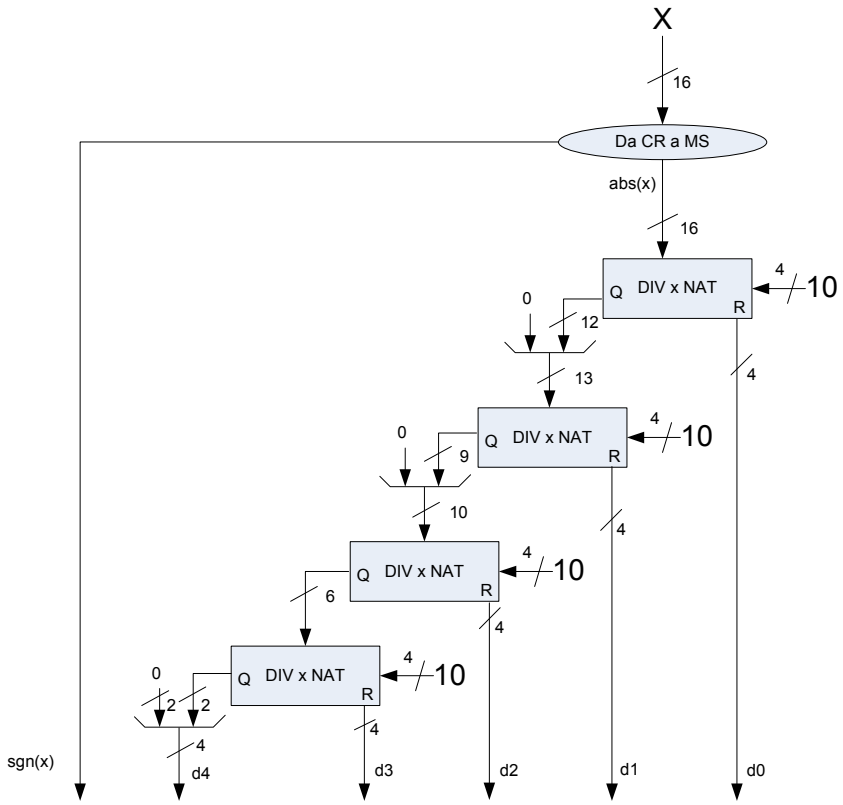
COMB è una rete combinatoria da ispezionare. L'unità *XXX* emette, tramite l'**interfaccia** seriale di uscita, la tabella di verità della rete *COMB*. In dettaglio: fa emettere dall'interfaccia seriale otto trame in ognuna delle quali i tre bit più significativi del byte trasportato sono uno stato di ingresso della rete *COMB* e i cinque bit meno significativi sono il corrispondente stato di uscita di detta rete. Se ad esempio la tabella di verità della rete *COMB* fosse quella riportata in figura, i byte trasportati dalle trame sarebbero `00011001,00101000, ...`

x2_x0	z4_z0
000	11001
001	01000
…	….

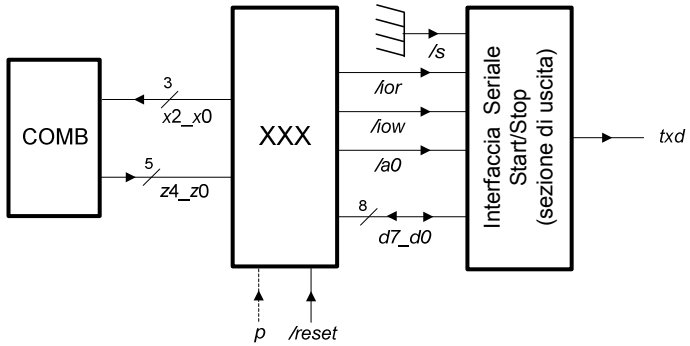
Terminata l'ispezione di *COMB*, l'unità *XXX* si blocca fino a nuovo reset asincrono.

Note: si supponga che il tempo di risposta della rete combinatoria *COMB* sia inferiore al periodo del clock. Non si inseriscano stati di wait nella gestione dell'interfaccia.

Soluzione
Dato che $ABS(x) \le 2^{15} = 32768$, il modulo può essere rappresentato su 5 cifre in base 10. Le cifre in base 10 $d_4 \ldots d_0$ si trovano dividendo successivamente il modulo per 10 (divisione naturale), avendo cura di dimensionare correttamente i divisori, estendendo quando necessario. La rete risultante è quella di figura.



Una possibile soluzione




```
module XXX(x2_x0,z4_z0,a0,ior_,iow_,d7_d0,p,reset_);
 input         p,reset_;
 output [2:0]  x2_x0;
 input  [4:0]  z4_z0;
 output        a0,ior_,iow_;
 inout[7:0]    d7_d0;

 reg    [2:0]  X2_X0;   assign x2_x0=X2_X0;
 reg           A0;      assign a0=A0;
 reg           IOR_;    assign ior_=IOR_;
 reg           IOW_;    assign iow_=IOW_;
 reg           DIR;
 reg    [7:0]  MBR;     assign d7_d0=(DIR==1)?MBR:'HZZ;

 reg [2:0]     STAR;
 parameter S0=0,S1=1,S2=2,S3=3,S4=4,S5=5,S6=6,S7=7;

 always @(posedge p or negedge reset_)
  if (reset_==0) begin DIR<=0; IOR_=1; IOW_=1; X2_X0<=0; MBR='HFF; STAR=S0;
end else #3
  casex(STAR)
   // Test sul flag di buffer di uscita vuoto
   S0: begin DIR<=0; A0<=0; STAR<=S1; end
   S1: begin IOR_<=0; STAR<=S2; end
   S2: begin IOR_<=1; STAR<= (d7_d0[5]==1)?S3:S1; end
   // Emissione
   S3: begin DIR<=1; A0<=1; MBR<={X2_X0,z4_z0}; STAR<=S4; end
   S4: begin IOW_<=0; STAR<=S5; end
   S5: begin IOW_<=1; X2_X0<=(X2_X0)+1; STAR<=(X2_X0==7)?S6:S0; end
   // Blocco
   S6: begin STAR<=S6; end
  endcase
endmodule
```