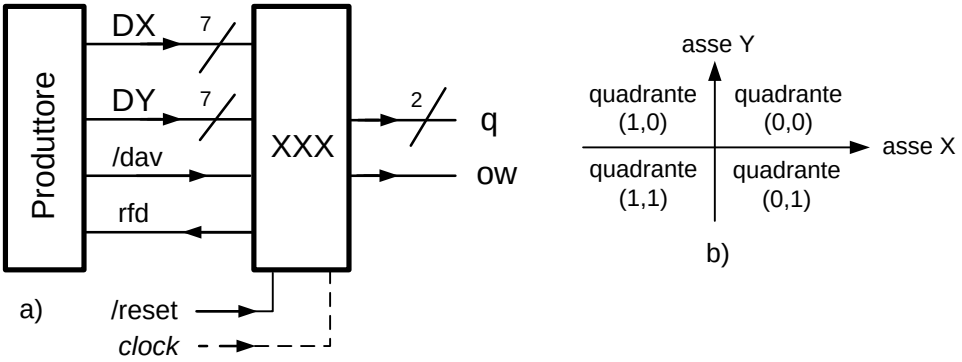
Compito di Reti Logiche, 16/7/19

Esercizio 1

Siano *x,y* le coordinate intere di un punto sul piano cartesiano, rappresentate in base 2 su 8 bit in complemento alla radice. Sintetizzare una rete combinatoria che prende in ingresso la rappresentazione delle coordinate di un punto sul piano e produce in uscita una variabile logica *z* che vale 0 se il punto è esterno ad un cerchio di raggio 16, ed 1 altrimenti.

Esercizio 2

L'unità XXX (vedi Fig. a) analizza le coordinate di un punto P in un piano cartesiano e genera una variabile a due bit *q* il cui valore indica in quale quadrante si trova il punto P (vedi Fig. b). L'Unità colloquia con un produttore che gli fornisce le rappresentazioni DX e DY di **due numeri interi a 7 bit**, che indicano la modifica da apportare alle rappresentazioni X e Y delle coordinate attuali *x* e *y* del punto P. Quando almeno una delle coordinate **non è più rappresentabile su 8 bit**, l'Unità pone a 1 la variabile di uscita *ow* e si ferma in attesa di un nuovo segnale di reset.



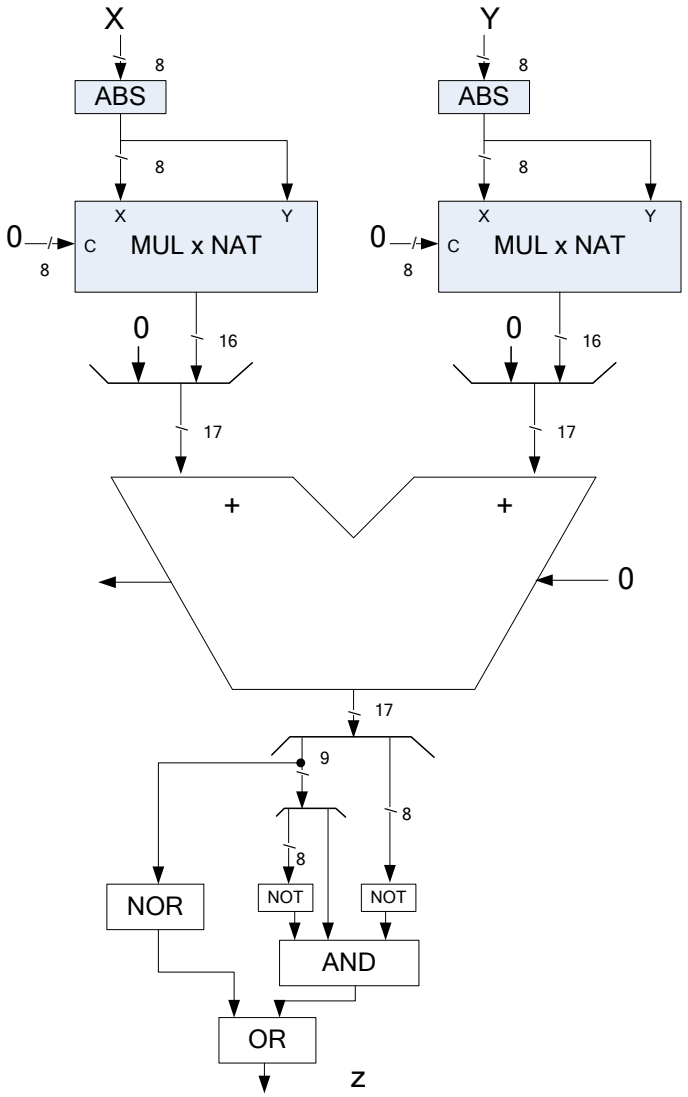
Si descriva e l'unità XXX e si sintetizzi e si disegni in dettaglio il circuito della parte operativa relativa al registro OW che supporta la variabile di uscita *ow*, riducendo il tutto a reti note.

Nota. Al reset iniziale il punto P va posizionato sull'origine degli assi. Tutte le rappresentazioni sono in complemento a 2; si consiglia di **rappresentare le coordinate *x* e *y* su su 9 bit** per semplificare il calcolo dell'overflow rispetto 8 bit. Si considerino i semiassi come appartenenti ai quadranti che ottimizzano la generazione della variabile *q*.

Soluzione Esercizio 1

La disuguaglianza da verificare è $x^2 + y^2 = |x|^2 + |y|^2 \leq 16^2 = 256$. La comparazione può essere ottimizzata come nel diagramma a fianco, in quanto:

a) $A < 256$ si testa guardando se sono a zero tutti i bit più significativi di A fino al bit 8 compreso
b) $A = 256$ si testa con una maschera a 17 bit, in cui tutti i bit sono a zero tranne il bit 8.



Esercizio 2 – soluzione

```
module XXX(DX,DY,dav_,rfd, q,ow, clock,reset_);
 input       clock, reset_;
 input [6:0] DX,DY;
 input       dav_;
 output      rfd;
 output[1:0] q;
 output      ow;
 reg[8:0]    X,Y;
 reg         RFD;   assign   rfd = RFD;
 reg[1:0]    Q;     assign   q = Q;
 reg         OW;    assign   ow = OW;
 reg[1:0]    STAR;  parameter S0=0,S1=1,S2=2,S3=3;
 always @(reset_==0) #1 begin X=0; Y=0; RFD=1; OW=0; Q=0; STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
  casex (STAR)
   S0: begin RFD<=1; STAR<=(dav_==1)?S0:S1; end
   S1: begin X<=X+{DX[6],DX[6],DX}; Y<=Y+{DY[6],DY[6],DY}; STAR<=S2; end
   S2: begin RFD<=0; Q<={X[8],Y[8]}; OW<=((X[8]==X[7])&(Y[8]==Y[7]))?0:1;
             STAR<=(dav_==0)?S2:S3; end
   S3: begin STAR<=(OW==1)?S3:S0; end
  endcase
endmodule
```