Esercizio 1

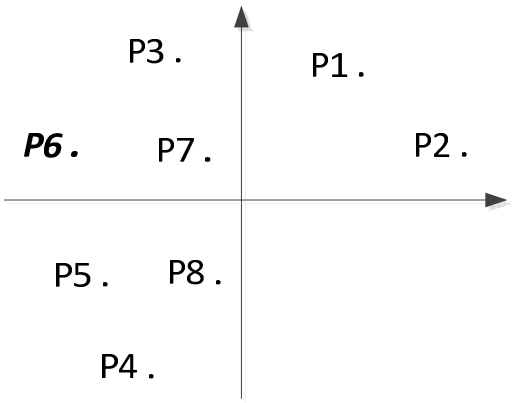
Sia dato un piano cartesiano con coordinate intere, rappresentate su *n* bit in complemento alla radice. Una rete WWW genera (rappresentazioni di) coordinate di punti del piano secondo un algoritmo ignoto.

Sintetizzare una rete di Mealy ZZZ che prende in ingresso coppie di coordinate *X*,*Y* prodotte da WWW. La rete ZZZ tiene l'uscita a 0 fintanto che i punti generati da WWW rimangono nello stesso quadrante o attraversano i quadranti in senso *antiorario*, senza saltare quadranti, altrimenti porta l'uscita ad 1 per un clock e ricomincia.

Ad esempio, se WWW produce le coordinate dei punti P1-P8 della figura sottostante, nell'ordine da 1 a 8, ZZZ porta l'uscita ad 1 quando vede in ingresso le coordinate del punto P6.

Si considerino i punti sugli assi e nell'origine come appartenenti ad un quadrante qualunque (quello che fa più comodo ai fini della soluzione dell'esercizio).

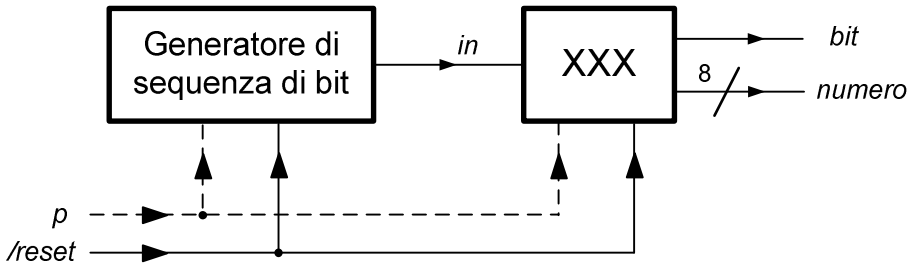
Descrivere esplicitamente qualunque rete non vista a lezione.



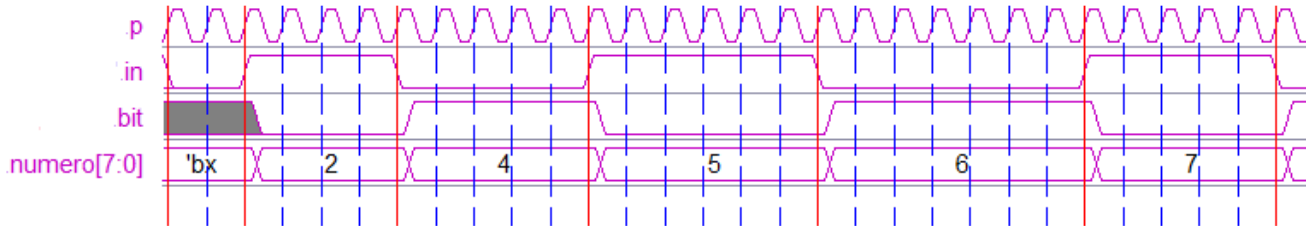
Esercizio 2

Descrivere e **sintetizzare** una Unità *XXX* che riceve una sequenza di bit da un *Circuito Generatore* e trasforma tale sequenza in una sequenza di coppie del tipo {*un bit, un numero naturale*} in accordo alle seguenti specifiche:

- Generatore di bit e Unità XXX **pilotati dallo stesso clock**;
- Numero massimo di 0 e di 1 consecutivi nella sequenza non superiore a 255;
- Primo bit ricevuto da XXX: 0;
- Prima coppia emessa da XXX: {0, numero di 0 consecutivi con cui inizia la sequenza di bit};
- Seconda coppia emessa da XXX: {1, numero di 1 consecutivi che segue gli 0 di cui sopra };
- Terza coppia emessa da XXX: {0, numero di 0 consecutivi che segue gli 1 di cui sopra };
- .....



Si noti che la prima coppia deve essere emessa quando *in* viene trovato ad 1 e deve essere mantenuta per tutto il tempo in cui *in* rimane ad 1. Poi, con le stesse modalità viene emessa la seconda coppia e così via, come mostrato nel seguente esempio.



ATTENZIONE

- Non ci si preoccupi se viene commesso un errore nel conteggio dei primi bit 0 dopo il reset.
- Si **tenga presente** che ad **ogni clock arriva un nuovo bit** e che quindi l'Unità *XXX* **non deve perdere il passo.**

Come verifica, si tracci un diagramma di temporizzazione in cui si evidenzi l'evoluzione dei vari registri dell'Unità *XXX*, nel caso che la sequenza di bit ricevuta sia **001100011...** (la sequenza che deve essere emessa è ovviamente **{0,2} {1,2} {0,3} {1,2} ...)**

Esercizio 1 - Soluzione

Se le coordinate sono in complemento alla radice, sotto l'ipotesi semplificativa sui punti che si trovano sugli assi, il quadrante di un punto è individuato dal bit più significativo di ciascuna coordinata (gli altri *n*-1 sono irrilevanti). Chiamando per semplicità $a = X_{n-1}, b = Y_{n-1}$, la corrispondenza tra i quadranti ed i bit più significativi delle coordinate, è la seguente:

1° quadrante (Q1): $a = 0, b = 0$

2° quadrante (Q2): $a = 1, b = 0$

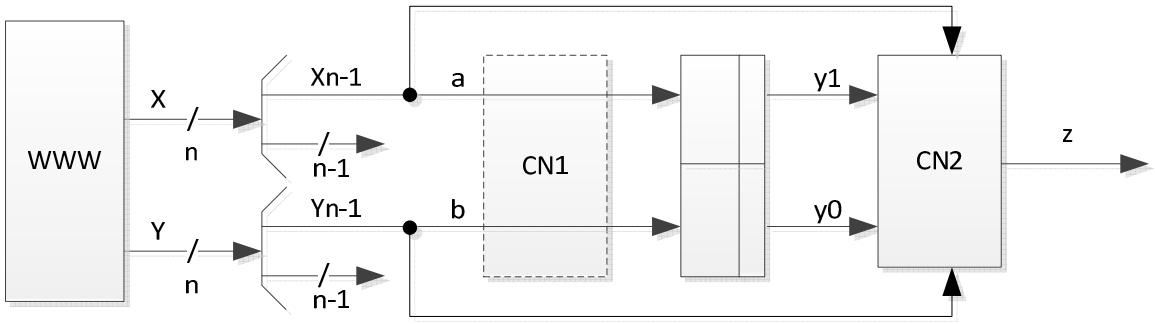
3° quadrante (Q3): $a = 1, b = 1$

4° quadrante (Q4): $a = 0, b = 1$

Pertanto ZZZ può essere descritta come RSS di Mealy con due ingressi *a, b*, quattro stati (Q1-Q4) ed un'uscita *z*. La rete ha la seguente tabella di flusso:

ab	00	01	11	10
Q1	Q1/0	Q4/1	Q3/1	Q2/0
Q2	Q1/1	Q4/1	Q3/0	Q2/0
Q3	Q1/1	Q4/0	Q3/0	Q2/1
Q4	Q1/0	Q4/0	Q3/1	Q2/1

Dalla tabella di flusso, osservando che tutti gli stati su una colonna sono identici, si evince che la rete CN1 non dipende dalle variabili di stato, ma solo dagli ingressi. Codificando i quattro quadranti nel modo più ovvio (Q1=00, Q2=10, Q3=11, Q4=01), si ottiene inoltre che $y_1 = a$, $y_0 = b$, dove y_1, y_0 sono le variabili di stato. In questo caso, quindi, CN1 è un cortocircuito.



La seguente tabella di verità per la rete CN2 è la seguente:

y1y0 \ ab	00	01	11	10
00	0	1	1	0
01	0	0	1	1
11	1	0	0	1
10	1	1	0	0

Una sintesi a costo minimo di CN2 (soggetta ad alee) è la seguente:

$$z = \overline{y_1} \cdot \overline{y_0} \cdot b + \overline{y_1} \cdot y_0 \cdot a + y_1 \cdot y_0 \cdot \overline{b} + y_1 \cdot \overline{y_0} \cdot \overline{a}$$

Esercizio 2 – soluzione

```
module XXX(in, bit,numero, p,reset_);
 input       p,reset_;
 input       in;
 output      bit;
 output[7:0] numero;
 reg BIT;              assign bit=BIT;
 reg [7:0]   NUMERO;  assign numero=NUMERO;
 reg [7:0]   COUNT;
 reg STAR;   parameter S0=0,S1=1;
 always @(posedge p or negedge reset_)
  if (reset_==0) begin COUNT<=0; STAR<=S0; end else #3
  casex(STAR)
   S0: begin COUNT<=(in==0)?(COUNT+1):1; BIT<=(in==0)?BIT:0;
       NUMERO<=(in==0)?NUMERO:COUNT; STAR<=(in==0)?S0:S1; end
   S1: begin COUNT<=(in==1)?(COUNT+1):1; BIT<=(in==1)?BIT:1;
       NUMERO<=(in==1)?NUMERO:COUNT; STAR<=(in==1)?S1:S0; end
  endcase
endmodule
```