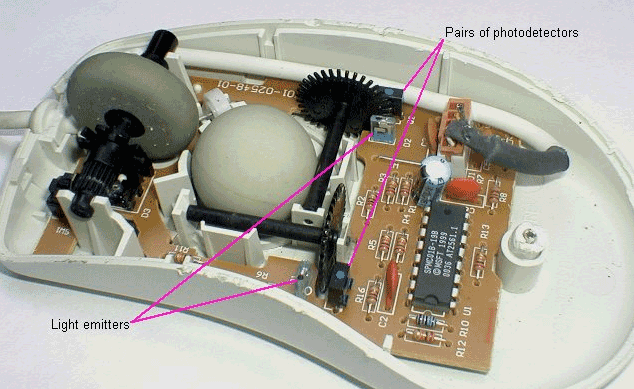
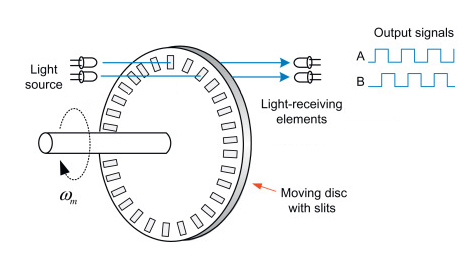
# Progetto di un decoder rotativo incrementale sincrono

Descrizione:

Un decoder incrementale rotativo è un dispositivo che ricevendo quale input due segnali alternati in quadratura, generati da un omologo “ENCODER incrementale rotativo”, viene ad incrementare, o rispettivamente a decrementare, un conteggio cumulativo totale (che verrà poi contestualizzato su delle opportune linee di uscita) e che riporta quanti passi in totale siano stati fatti in una certa direzione.   
Tale dispositivo è stato ampiamente usato in passato ad esempio sui mouse elettromeccanici, ma trova ancora moltepli applicazioni.



Mouse ottico con coppia di encoder incrementali rotativi



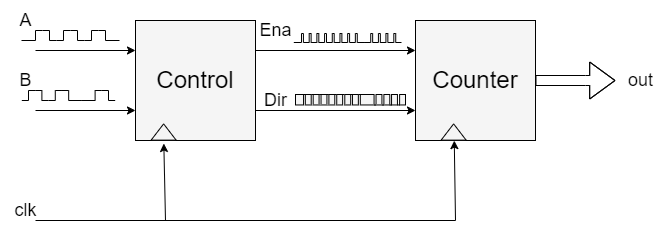
Funzionamento generale dell’encoder e forme d’onda generate

Lo scopo di questo esercizio e realizzare un dispositivo che ricevendo in ingresso le due forme d’onda in quadratura vada ad incrementare (o decerementare) un opportuno contatore in base ai fronti d’onda ricevuti, valutando al contempo se la forma d’onda A è in anticipo (oppure in rtitardo) rispetto la f.d.o. B ed agendo di conseguenza sul conteggio cumulativo.

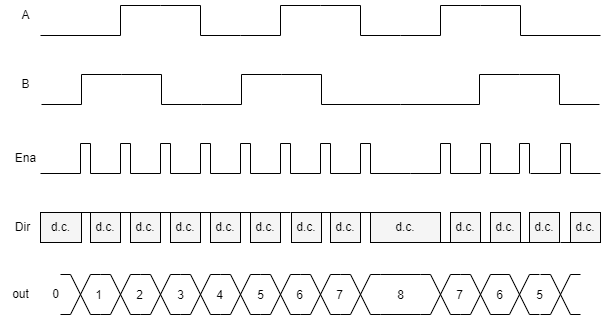
Individuare i segnali coinvolti e lo schema generale.

Visto dall’esterno il decoder presenta 2 segnali di ingresso (A e B) oltre ad un clock che opera un campionamento dei medesimi e da BUS composto da N linee d’uscita che rappresentano i bit del conteggio totale.

Nella realizzazione del sistema si possono individuare due blocchi:



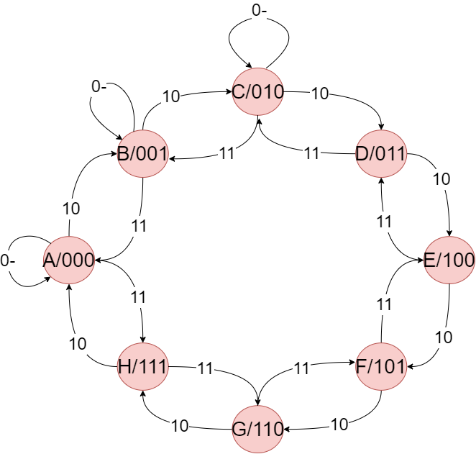
* Un contatore sincrono in grado di attivare il conteggio attraverso il segnale “Ena” (enable) solo quando sia opportuno (ovvero quando viene rilevato un fronte sulle linee A o B) ed inoltre in grado di invertire il senso del conteggio attraverso un opportuno segnale di controllo “Dir”.
* Un dispositivo che leggendo i segnali (A e B) in ingresso attivi opportunamente i segnali di controllo per il contatore stesso ovvero:
  + Attivi il segnale di “enable” quando rileva la presenza di un fronte o sul segnale A o sul segnale B (risulta infatti più opportuno incrementare/decrementare il conteggio tanto sui fronti di salita quanto di discesa di entrambi i segnali per migliorare la risoluzione del dispositivo)
  + Piloti opportunamente il segnale di “incremento/decremento” conteggio valutando lo sfasamento tra le due forme d’onda in ingresso. Tale segnale è di interesse per il contatore che segue solamente quando il segnale di enable è alto, mentre nelle altre condizioni (quando il contatore è fermo) il segnale di direzione non è di alcun interesse (don’t case).



Di cui A e B sono i segnali in ingresso, Ena e Dir sono i segnali da inviare al contatore che ne determinano istante di incremento/decremento e direzione. Out è una possibile uscita desiderata che mantiene memoria del totale dei passi eseguiti dall’encoder.

1. Progetto del contatore

Il Contatore da sviluppare è una macchina sincrona rappresentabile col seguente diagramma di Moore



Si noti che:

* Il conteggio avviene in codice binario.
* Il numero di stati, per semplicità realizzativa è pari ad una potenza di due.
* Nel caso in esame utilizzeremo ad esempio 8 stati, ma vedremo come, una volta compresa la logica dell’architettura sarà possibile espandere quest’ultima a qualsiasi numero di stati che sia una potenza di due
* Il conteggio avviene circolarmente e senza alcuna saturazione: ad esempio nel conteggio incrementale ad 111 segue 000, il che ha senso ad esempio se si pensa che le uscite siano codificate in modalità “signed”, pertanto a “-1” segue 0. O nel conteggio decrementale a 000 segue 111 (a “0” segue “-1”)
* Anche se non completamente rappresentato nel grafo quando il segnale Ena=0, per OGNI stato, lo stato futuro coincide con lo stato attuale (ovvero il contatore non procede nel conteggio) ma si ferma allo stato attuale.
* Quando il segnale “Ena=1” il conteggio avviene in modo orario (incrementale) o antiorario (decrementale) a seconda del segnale “dir”.

Il Grafo di cui sopra trova corrispondenza nella seguente tavola di Huffman



Data la particolare corrispondenza tra le uscite desiderate nei vari stati, risulta piuttosto logico adottare una codifica per i vari stati a tre variabili (y1,y2,y3) che coincida con l’uscita desiderata. Si può quindi definire la seguente tavola di flusso. (Si noti che per questioni di visualizzazione ottimale secondo il codice Gray, sono state scambiate di posizione le linee corrispondenti alle coppie di stati CD e GH).



Supponendo di adottare dei Flip-Flop di tipo D per memorizzare le variabili di stato le eccitazioni da portare a questi coincidono con le variabili di stato future e queste possono essere raccolte in tre mappe di Karnaugh:



Adesso prima di passare all’individuazione delle tre funzioni con la semplificazione in base agli implicanti merita analizzare congiuntamente le tre mappe. Da esse si evince infatti che:

* Quando E=0 (parte sinistra della mappa): la variabile di stato futura coincide con la corrispondente variabile di stato attuale   
  (Y1’=Y1, Y2’=Y2,Y3’=Y3”), il che ci porta a scrivere, sfruttando il teorema di Shannon:
* La parte destra della tabella che rappresenta la funzione f(..) evidenziata qui sopra (relativa ad E=1) risulta invece indipendente da Y1 per le variabili Y2’ ed Y3’, mentre per Y1’ la parte superiore (con Y1=0) appare essere la duale della parte inferiore (Y1=1) il che suggerisce per quest’ultima la presenza di un’operazione di Xor (Y1, …). (Vedasi decomposizioni semplici)
* Le tre funzioni sarebbero quindi rispettivamente





Sebbene a prima vista le tre funzioni non mostrino grande similitudine, si noti che esse potrebbero anche essere scritte come:







Che da un punto di vista sia logico che euristico (e ricordandosi che fn() rappresenta la variabile di stato futura quando E=1) esse potrebbero essere descritte verbalmente nel seguente modo:

*“ la variabile di stato futura cambia stato (rispetto quella attuale) se e solo se tutte le variabili che la precedono (ovvero TUTTI ibit meno significativi) sono pari a 1 - nel caso di conteggio incrementale (D=1) oppure se tutte le variabili che la precedono (bit meno significativi) sono pari a 0 nel caso di conteggio decrementale.”*

Il che rispecchia perfettamente ciò che accade in un conteggio binario che ad esempio nel passaggio

0011010010111111 +1 = 0011010010000000

Che vede invertirsi tutte le variabili evidenziate in rosso in quanto per ciascuna di esse TUTTE le variabili che rappresentano i bit meno significativi che precedono sono pari a 1.

Similmente:

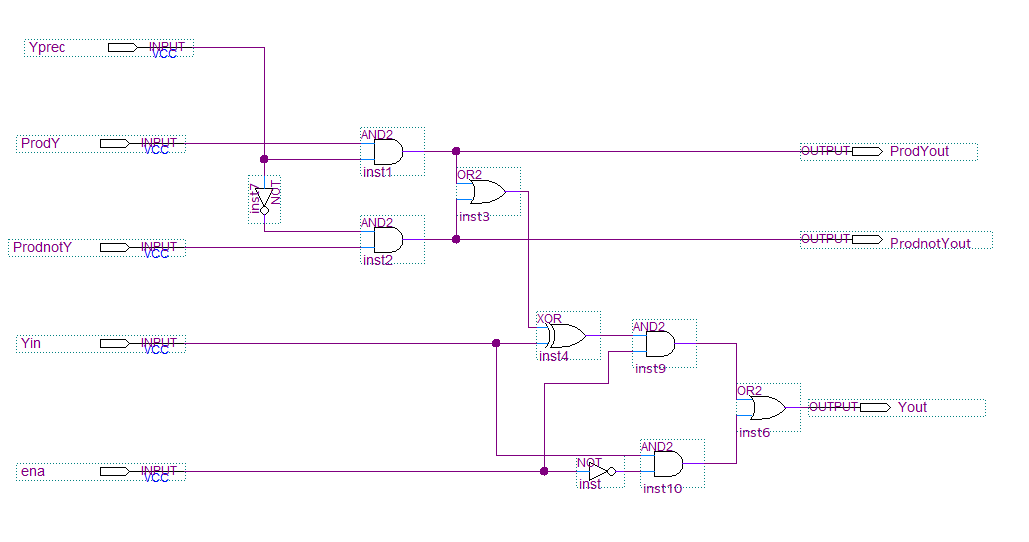
0011010010000000 – 1 = 0011010010111111

Nel conteggio decrementale si invertono quelle variabili che vedono TUTTE le variabili che rappresentano i bit meno significativi pari a zero.

A questo punto risulta evidente che ove si volesse realizzare un contatore con più bit le equazioni per i bit più significativi potrebbero essere realizzate attraverso una “generalizzazione” delle equazioni di cui sopra:

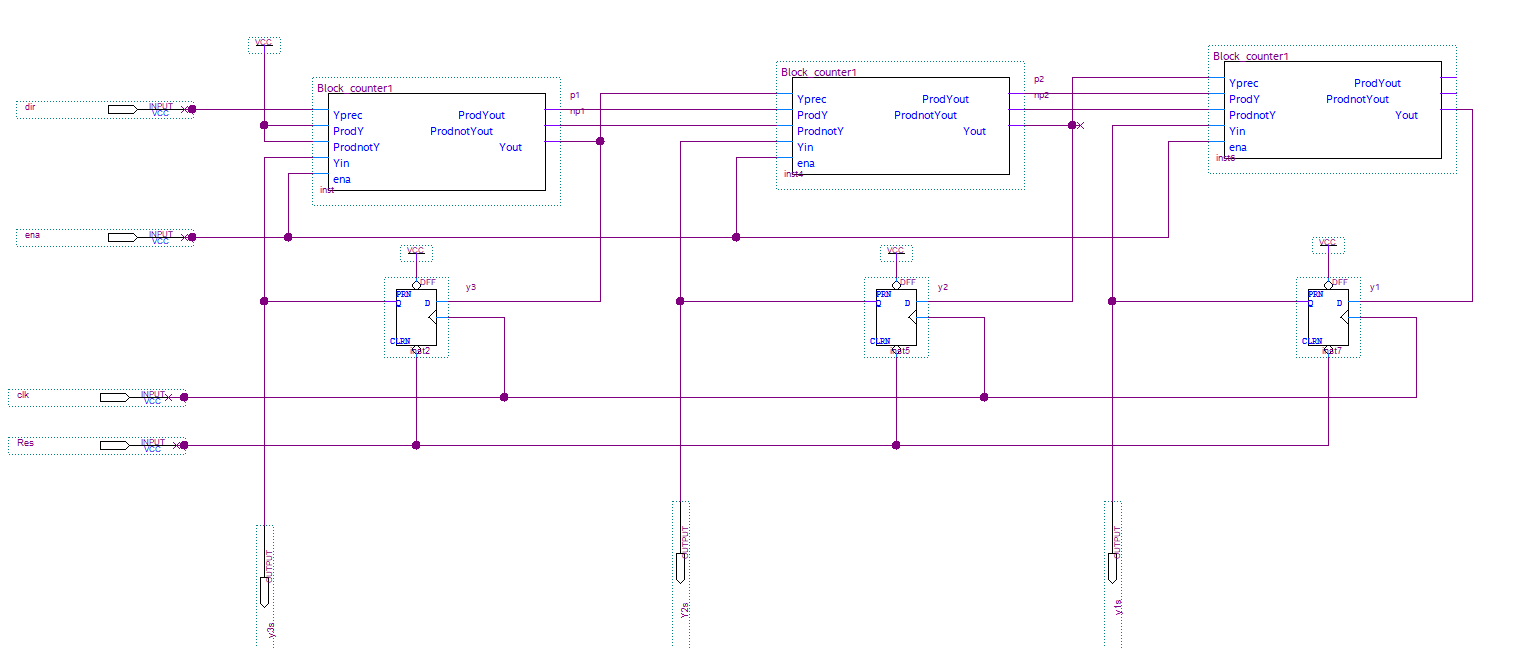


Uno schema “generico” che realizzi tale funzione generica può essere il seguente



Che opportunamente iterato genera ad ogni step sia il prodotto logico dei bit meno significativi con D sia il prodotto degli stessi negati con not(D). Entambi i prodotti vengono uniti nell’operazione di OR e quindi diretti alla porta Xor che provvederà eventualmente ad invertire il bit Yin quando la condizione espressa sopra (tutti 0 e D=0 oppure tutti 1 e D=1) sia verificata. Inoltre, in base al valore assunto da Ena viene portata in uscita o la variabile già presente nella cella di memoria oppure quella appena calcolata.

Il circuito del contatore a tre bit completo con le tre celle di memoria realizzate attraverso tre FlipFlop di tipo D potrebbe pertanto essere quindi il seguente:



Che grazie alla sua natura ricorsiva può essere iterato N volte in base al numero di bit richiesti.

Si noti che al primo stadio dove la cella di logica combinatoria deve fornire come uscite combinatorie solamente D e not(D) (oltre evidentemente all’eccitazione del Flip-Flop) si porta sulla variabile d’ingresso Yprec il segnale di direzione e sulle due linee ove si “accumulano” i prodotti dei bit meno significativi il valore fisso “1”.

I Flip Flop scelti presentano inoltre una linea di Reset asincrona (in questo caso in logica negata - ovvero attiva quando bassa) che consente di forzare il circuito in uno stato iniziale noto. C’è inoltre a disposizione una linea di “Set/Preload” asincrona che però non viene impiegata in questo contesto e viene mantenuta costantemente alta.

Il circuito descritto in Verilog HDL è composto da due parti:

Una prima parte che descrive la logica combinatoria sequenziale, denominata “Block\_counter1”

module Block\_counter1(

ProdY,

ProdnotY,

ena,

Yin,

Yprec,

ProdYout,

ProdnotYout,

Yout

);

input wire ProdY;

input wire ProdnotY;

input wire ena;

input wire Yin;

input wire Yprec;

output wire ProdYout;

output wire ProdnotYout;

output wire Yout;

wire WIRE\_0;

wire WIRE\_1;

wire WIRE\_2;

wire WIRE\_3;

wire WIRE\_4;

wire WIRE\_5;

wire WIRE\_6;

wire WIRE\_7;

assign ProdYout = WIRE\_3;

assign ProdnotYout = WIRE\_2;

assign WIRE\_0 = ~ena;

assign WIRE\_3 = ProdY & Yprec;

assign WIRE\_5 = Yin & WIRE\_0;

assign WIRE\_2 = WIRE\_1 & ProdnotY;

assign WIRE\_4 = WIRE\_2 | WIRE\_3;

assign WIRE\_7 = WIRE\_4 ^ Yin;

assign Yout = WIRE\_5 | WIRE\_6;

assign WIRE\_1 = ~Yprec;

assign WIRE\_6 = WIRE\_7 & ena;

endmodule

Ed una seconda denominata “Counter” che importa la precedente descrizione in forma di “block-box” (ovvero di “istanza”), definendone solamente i collegamenti alle varie linee e descrive inoltre la presenza dei FlipFlop D in maniera puramente comportamentale (always at posedge …) senza addentrarsi nella loro eventuale struttura interna.

module Counter(dir,ena,clk,Res,Y2s,y3s,y1s);

input wire dir;

input wire ena;

input wire clk;

input wire Res;

output wire Y2s;

output wire y3s;

output wire y1s;

wire np1;

wire np2;

wire p1;

wire p2;

wire y1;

wire y2;

wire y3;

wire SYNTHESIZED\_WIRE\_5;

reg DFF\_inst2;

wire SYNTHESIZED\_WIRE\_2;

reg DFF\_inst5;

wire SYNTHESIZED\_WIRE\_3;

reg DFF\_inst7;

wire SYNTHESIZED\_WIRE\_4;

assign Y2s = DFF\_inst5;

assign y3s = DFF\_inst2;

assign y1s = DFF\_inst7;

assign SYNTHESIZED\_WIRE\_5 = 1;

assign SYNTHESIZED\_WIRE\_2 = 1;

assign SYNTHESIZED\_WIRE\_3 = 1;

assign SYNTHESIZED\_WIRE\_4 = 1;

Block\_counter1 b2v\_inst(

.Yprec(dir),

.ProdY(SYNTHESIZED\_WIRE\_5),

.ProdnotY(SYNTHESIZED\_WIRE\_5),

.Yin(DFF\_inst2),

.ena(ena),

.ProdYout(p1),

.ProdnotYout(np1),

.Yout(y3));

always@(posedge clk or negedge Res or negedge SYNTHESIZED\_WIRE\_2)

begin

if (!Res)

begin

DFF\_inst2 <= 0;

end

else

if (!SYNTHESIZED\_WIRE\_2)

begin

DFF\_inst2 <= 1;

end

else

begin

DFF\_inst2 <= y3;

end

end

Block\_counter1 b2v\_inst4(

.Yprec(y3),

.ProdY(p1),

.ProdnotY(np1),

.Yin(DFF\_inst5),

.ena(ena),

.ProdYout(p2),

.ProdnotYout(np2),

.Yout(y2));

always@(posedge clk or negedge Res or negedge SYNTHESIZED\_WIRE\_3)

begin

if (!Res)

begin

DFF\_inst5 <= 0;

end

else

if (!SYNTHESIZED\_WIRE\_3)

begin

DFF\_inst5 <= 1;

end

else

begin

DFF\_inst5 <= y2;

end

end

Block\_counter1 b2v\_inst6(

.Yprec(y2),

.ProdY(p2),

.ProdnotY(np2),

.Yin(DFF\_inst7),

.ena(ena),

.Yout(y1));

always@(posedge clk or negedge Res or negedge SYNTHESIZED\_WIRE\_4)

begin

if (!Res)

begin

DFF\_inst7 <= 0;

end

else

if (!SYNTHESIZED\_WIRE\_4)

begin

DFF\_inst7 <= 1;

end

else

begin

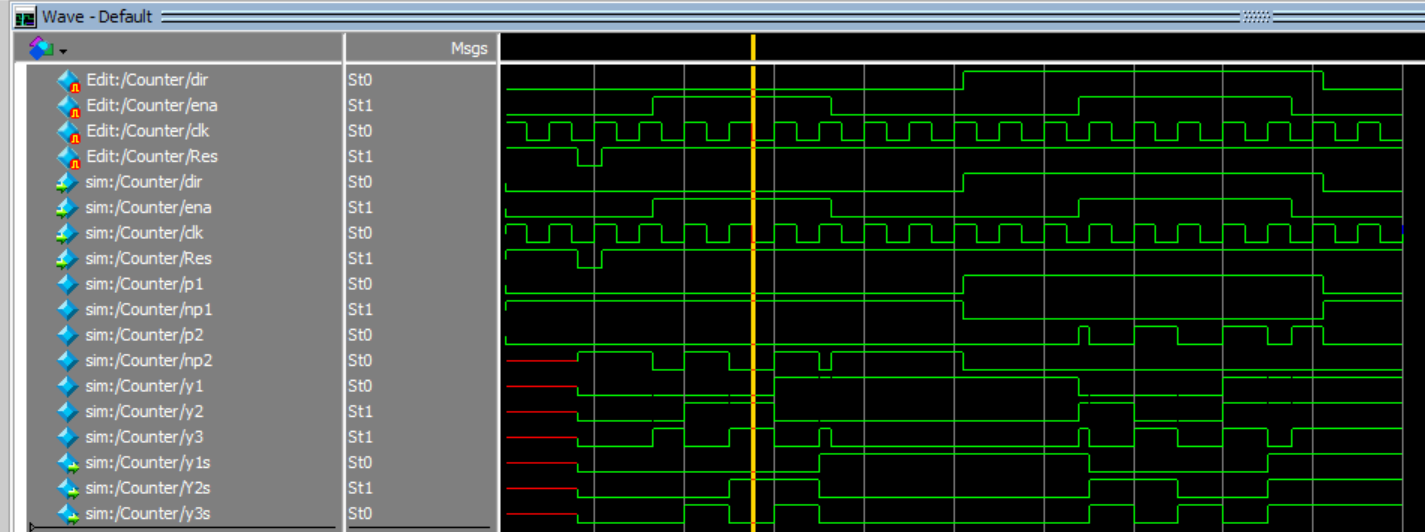
DFF\_inst7 <= y1;

end

end

endmodule

Il Contatore completo opportunamente simulato attraverso idonei segnali in ingresso dimostra un funzionamento corretto.



Si noti la differenza tra le variabili Yn e le variabili Yns dove le prime rappresentano l’eccitazione del FF e le seconde lo stato futuro. Le seconde “copiano” le prime solo all’arrivo del fronte di clock (istante di sincronismo).

1. Controllore

La seconda parte del progetto prevede di sviluppare un blocco che acquisendo in ingresso le forme d’onda in quadratura generi i segnali opportuni da inviare al contatore.

Si noti che la sequenza dei segnali presente in ingresso segue la sequenza

… 00 – 01 – 11 – 10 – 00 - …

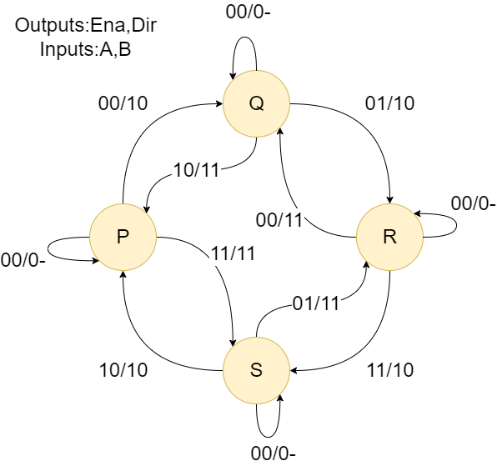
quando il dispositivo elettromeccanico gira in un senso mentre genera la sequenza

… 00 – 10 – 11 – 01 – 00 - …

quando gira in senso inverso

Si vuole pertanto realizzare un sistema che ogni qualvolta vi sia un cambio degli ingressi nella sequenza generi un impulso da inviare al piedino di enable, mentre sia in grado di distinguere se lo step eseguito sia stato fatto in un senso o nell’altro per poter controllare il segnale “Dir”.

Si può pertanto realizzare una macchina a 4 stati uno per ogni step della sequenza che vada ad interpretare i possibili passaggi di stato e che per ciascuno di essi generi i segnali corretti. La sua rappresentazione secondo Mealey potrebbe essere la seguente:



Si noti che:

* Il passaggio da uno stato all’altro avviene solamente quando si ha una variazione degli ingressi
* Fintanto che gli ingressi sono costanti il sistema si pone in uno stato e vi rimane
* il segnale Ena viene attivato solo durante i passaggi di stato, mentre rimane a 0 quando gli ingressi rimangono stabili
* Il segnale di direzione viene attivato se la transizione porta ad uno stato che segue in senso antiorario, mentre viene messo a 0 se il passaggio è ad uno stato che segue in senso orario.
* Se i segnali d’ingresso non cambiano e lo stato è stabile il segnale di direzione è ininfluente (essendo il segnale di enable disattivato.

La tavola di Huffman per la macchina sopra descritta è la seguente:



Si noti in particolare che alcune celle sono rimaste in condizione “don’t care” in quanto durante un funzionamento normale non si prevede possa presentarsi tale situazione. Peraltro potrebbe capitare che tale situazione invece si verifichi in caso di malfunzionamento del dispositivo elettro-meccanico che genera le due f.d.o in ingresso. Pertanto se si volesse migliorare il funzionamento del dispositivo si potrebbe prevedere in tali condizioni di raggiungere uno stato d’errore ove viene segnalato ad esempio su di un’uscita dedicata (Error) il malfunzionamento. Per semplicità in questo progetto tale possibilità non verrà considerata.

Come da prassi bisogna stabilire una opportuna codifica degli stati che potrebbe essere per semplicità

P: 00 – Q:01 – R:11 – S:10

La tavola di flusso diventa quindi:



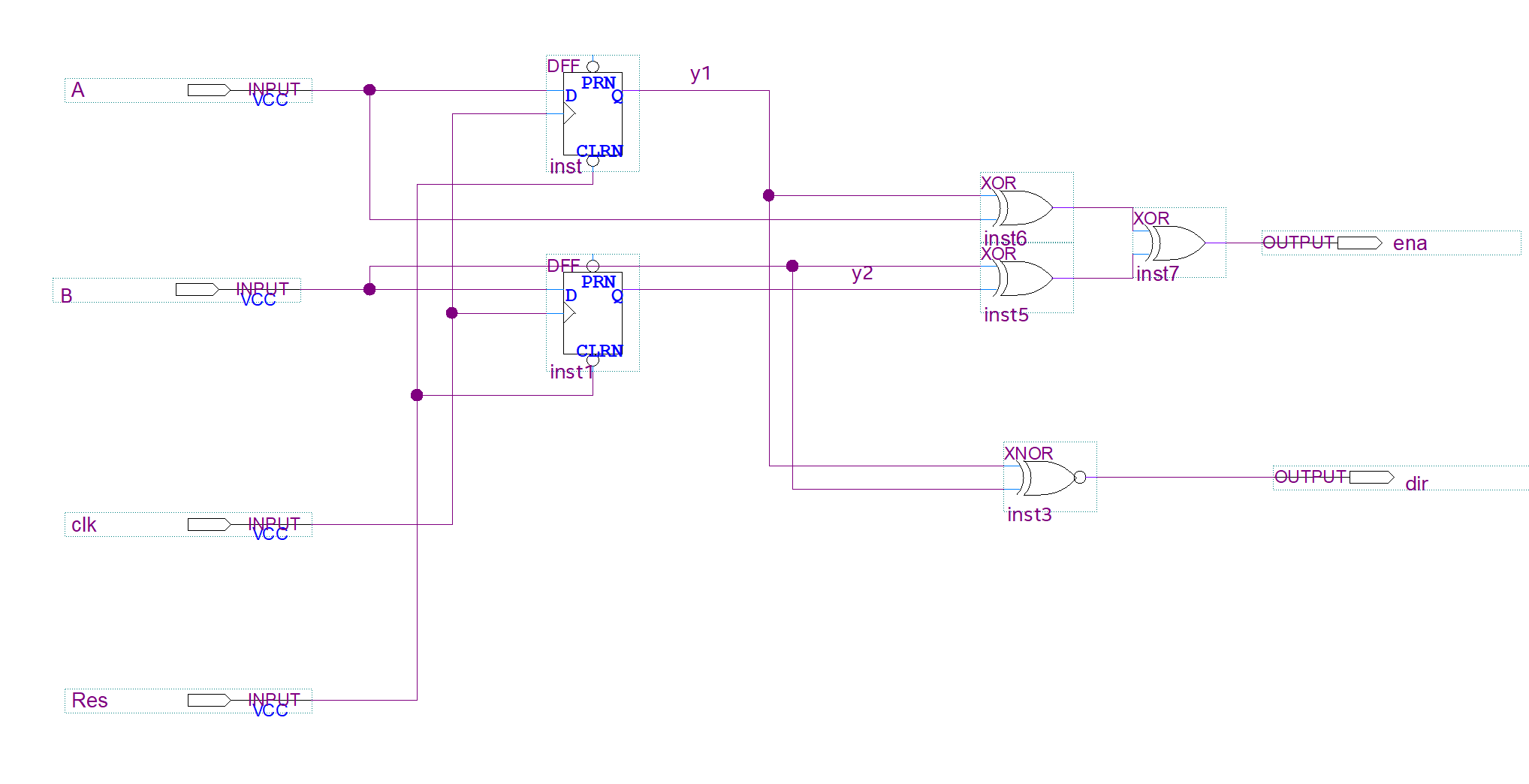
Se andiamo ad impiegare dei FlipFlop di tipo D le variabili di eccitazione da portare a queste coincidono con le variabili di stato future desiderate e nel caso specifico

Inoltre le funzioni di uscita (Ena e Dir) possono essere riassunte nelle seguenti mappe di Karnaugh:



Che portano ad evidenziare le seguenti funzioni d’uscita:

Che potrebbero essere realizzate col seguente circuito (ove sono stati aggiunti opportunamente i segnali impliciti di clock e di Reset asincrono)



Descritto in Verilog HDL nel seguente modo

module decoder\_rot(A,B,clk,Res,ena,dir);

input wire A;

input wire B;

input wire clk;

input wire Res;

output wire ena;

output wire dir;

reg y1;

reg y2;

wire WIRE\_0;

wire WIRE\_1;

always@(posedge clk or negedge Res)

begin

if (!Res)

begin

y1 <= 0;

end

else

begin

y1 <= A;

end

end

always@(posedge clk or negedge Res)

begin

if (!Res)

begin

y2 <= 0;

end

else

begin

y2 <= B;

end

end

assign dir = y1 ~^ B;

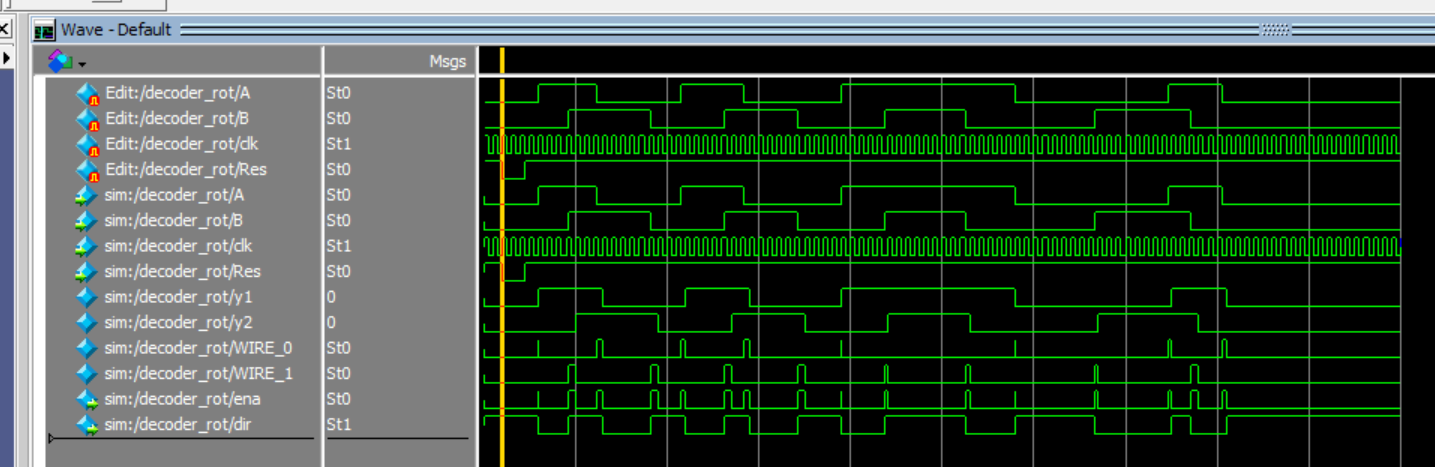
assign WIRE\_1 = B ^ y2;

assign WIRE\_0 = y1 ^ A;

assign ena = WIRE\_0 ^ WIRE\_1;

endmodule

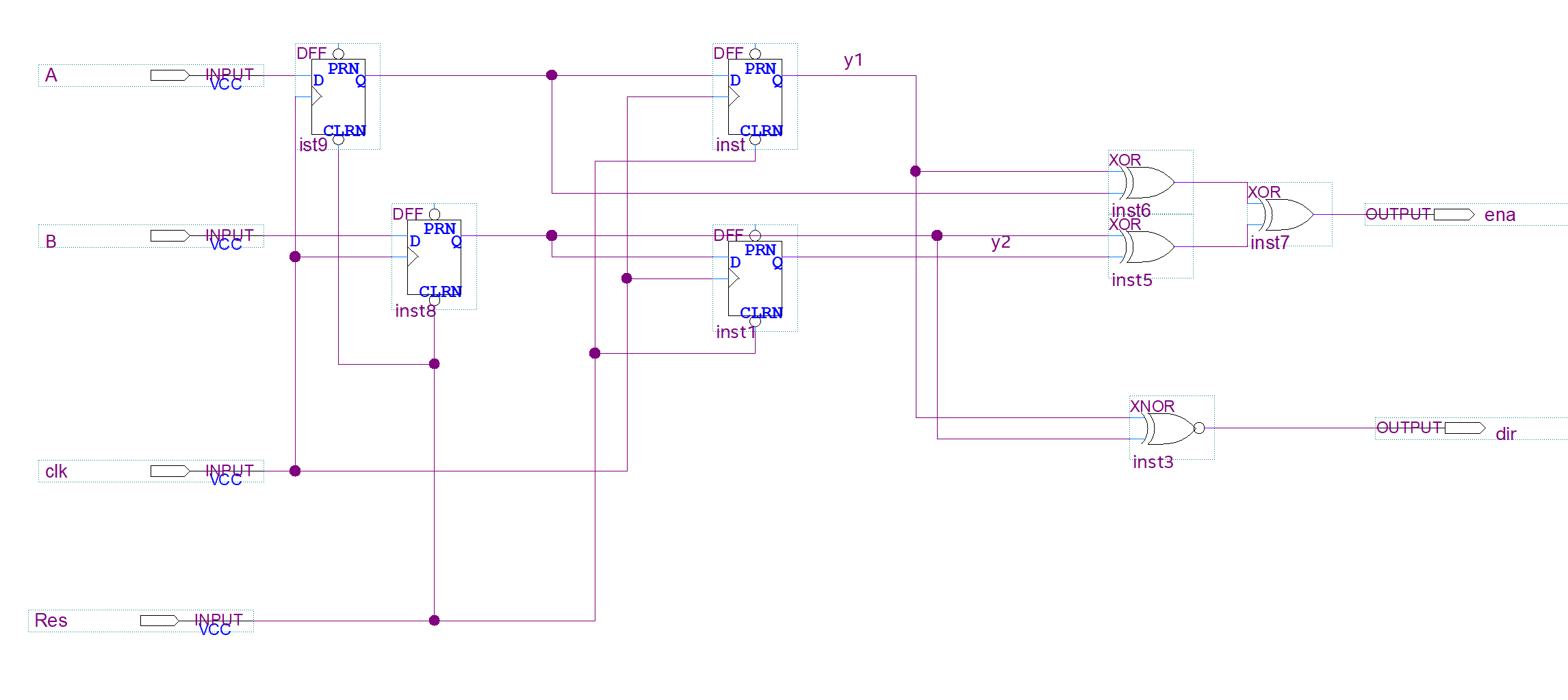
Che opportunamente simulato fornisce un discreto risultato:



Ad ogni fronte dei segnali A o B vi è un opportuno segnale di attivazione (ena) ed in corrispondenza a questi il segnale di direzione indica correttamente se si sta procedendo col conteggio in maniera incrementale o decrementale.

Risulta peraltro altresì evidente che a seguito della natura asincrona tra il segnale di clock ed i segnali A e B può capitare che i fronti di questi segnali siano troppo vicini gli uni agli altri, il che porta a generare degli impulsi che possono essere di durata inconsistente per essere opportunamente analizzati dal contatore che segue, creando delle condizioni di metastabilità.

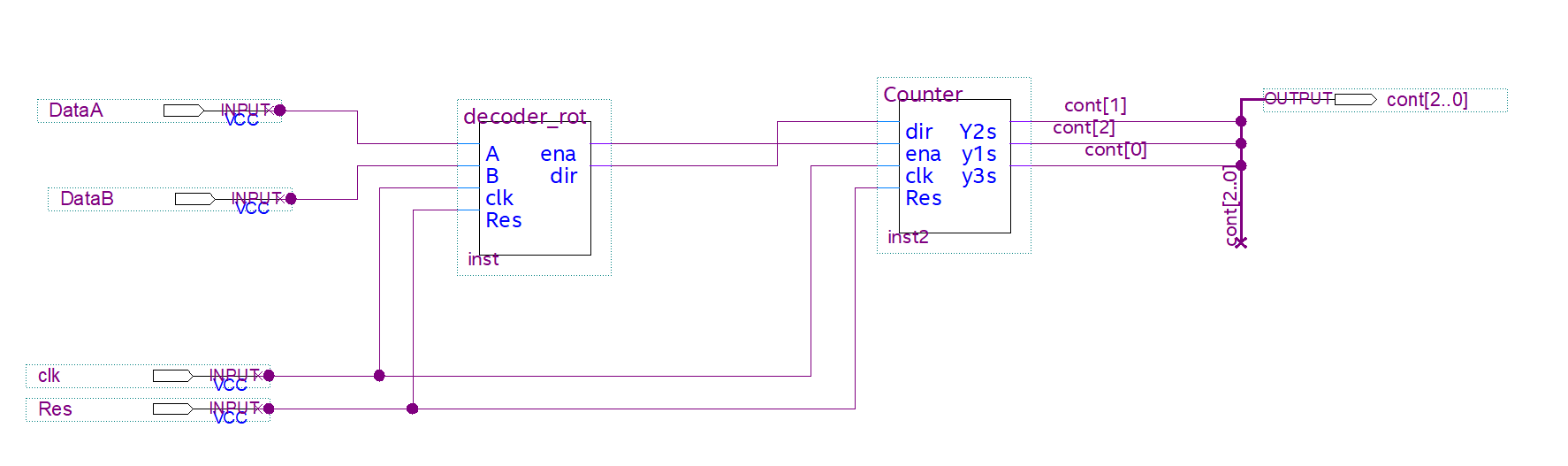
Per evitare ciò una precauzione che si è soliti adottare è quella di aggiungere un opportuno circuito di sincronismo che effettui un campionamento anche nei segnali di ingresso A e B che garantisca una loro congruità col segnale di clock.



Così facendo si realizzano dei segnali il cui funzionamento è più prevedibile ed idoneo per pilotare altri dispositivi sincroni in cascata, quali ad esempio nel nostro caso il contatore.



Giunti a questo punto non rimane che collegare tra loro i due blocchi, magari riunendo i vari bit di conteggio in un unico BUS che ne rappresenti numericamente il valore in forma decimale o binaria



Che in Verilog HDL viene descritto da:

module full\_encoder(DataA,DataB,clk,Res,cont);

input wire DataA;

input wire DataB;

input wire clk;

input wire Res;

output wire [2:0] cont;

wire [2:0] BUS;

wire WIRE\_0;

wire WIRE\_1;

decoder\_rot b2v\_inst(

.A(DataA),

.B(DataB),

.clk(clk),

.Res(Res),

.ena(WIRE\_1),

.dir(WIRE\_0));

Counter b2v\_inst2(

.dir(WIRE\_0),

.ena(WIRE\_1),

.clk(clk),

.Res(Res),

.Y2s(BUS[1]),

.y1s(BUS[2]),

.y3s(BUS[0]));

assign cont = BUS;

E simularne il funzionamento

