Descrizione e sintesi di un modulo Produttore che genera numeri dispari crescenti sostenendo un handshake /dav,rfd con un modulo Consumatore

Descrizione

```
module Produttore(dav_,rfd, numero, clock,reset_);
 input        clock,reset_;
 input        rfd;
 output       dav_;
 output [7:0] numero;
 reg          DAV_;     assign dav_=DAV_;
 reg [7:0]    NUMERO;   assign numero=NUMERO;
 reg [1:0]    STAR;     parameter S0=0,S1=1,S2=2,S3=3;
 always @(reset_==0) begin DAV_<=1; NUMERO<=1; STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
   casex(STAR)
    S0:  begin DAV_<=0; STAR<=(rfd==1)?S0:S1; end
    S1:  begin DAV_<=1; STAR<=(rfd==0)?S1:S2; end
    S2:  begin NUMERO<=NUMERO+2; STAR<=(NUMERO==255)?S3:S0; end
    S3:  begin STAR<=S3; end
   endcase
endmodule
```

Sintesi: Primo Passo

```
module Produttore(dav_,rfd, numero, clock,reset_);
 input        clock,reset_;
 input        rfd;
 output       dav_;
 output [7:0] numero;
 reg          DAV_;     assign dav_=DAV_;
 reg [7:0]    NUMERO;   assign numero=NUMERO;
 reg [1:0]    STAR;     parameter S0=0,S1=1,S2=2,S3=3;

 //reg DAV_
 always @(reset_==0) begin DAV_<=1; end
 always @(posedge clock) if (reset_==1) #3
   casex(STAR)
    S0:     begin DAV_<=0; end
    S1:     begin DAV_<=1; end
    S2,S3:  begin DAV_<=DAV_; end
   endcase

 //reg NUMERO
 always @(reset_==0) begin NUMERO<=1; end
 always @(posedge clock) if (reset_==1) #3
   casex(STAR)
    S0,S1,S3:  begin NUMERO<=NUMERO; end
    S2:        begin NUMERO<=NUMERO+2; end
   endcase

 //reg STAR
 always @(reset_==0) begin STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
   casex(STAR)
    S0:  begin STAR<=(rfd==1)?S0:S1; end
    S1:  begin STAR<=(rfd==0)?S1:S2; end
    S2:  begin STAR<=(NUMERO==255)?S3:S0; end
    S3:  begin STAR<=S3; end
   endcase
endmodule
```

Sintesi: Secondo Passo

```
module Produttore(dav_,rfd, numero, clock,reset_);
 input        clock,reset_;
 input        rfd;
 output       dav_;
 output [7:0] numero;
 reg          DAV_;     assign dav_=DAV_;
 reg [7:0]    NUMERO;   assign numero=NUMERO;
 reg [1:0]    STAR;     parameter S0=0,S1=1,S2=2,S3=3;

 //reg DAV_
 wire b1,b0; assign {b1,b0}=(STAR==S0)?'B00: (STAR==S1)?'B01:'B1X;
 always @(reset_==0) begin DAV_<=1; end
 always @(posedge clock) if (reset_==1) #3
   casex({b1,b0})
    'B00:  begin DAV_<=0; end
    'B01:  begin DAV_<=1; end
    'B1?:  begin DAV_<=DAV_; end
   endcase

 //reg NUMERO
 wire b2; assign b2=(STAR==S2)?1:0;
 always @(reset_==0) begin NUMERO<=1; end
 always @(posedge clock) if (reset_==1) #3
    casex(b2)
     0:  begin NUMERO<=NUMERO; end
     1:  begin NUMERO<=NUMERO+2; end
    endcase

 //reg STAR
 wire c0;    assign  c0=(rfd==1)?1:0;
 wire c1;    assign  c1=(NUMERO==255)?1:0;
 always @(reset_==0) begin STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
    casex(STAR)
     S0:  begin STAR<=(c0==1)?S0:S1; end
     S1:  begin STAR<=(c0==1)?S2:S1; end
     S2:  begin STAR<=(c1==1)?S3:S0; end
     S3:  begin STAR<=S3; end
    endcase
endmodule
```

// Riassunto delle variabili di comando

```
{b2,b1,b0}=(STAR==S0)?'B000:
           (STAR==S1)?'B001:
           (STAR==S2)?'B11X:
                      'B01X;
```

//Riassunto e espressioni algebriche per le variabili di condizionamento:

```
  c0=(rfd==1)?1:0 cioè
  c0=rfd;

  c1=(NUMERO==255)?1:0 cioè
  c1= NUMERO[7] & NUMERO[6] & ... NUMERO[2] & NUMERO[1] & NUMERO[0]
```

Sintesi: Ultimo Passo

```
module Produttore(dav_,rfd,numero, clock,reset_);
 input        clock,reset_;
 input        rfd;
 output       dav_;
 output [7:0] numero;
 wire b2,b1,b0,c1,c0;
 Parte_Operativa PO(dav_,rfd,numero,b2,b1,b0,c1,c0, clock,reset_);
 Parte_Controllo PC(b2,b1,b0,c1,c0, clock,reset_);
endmodule

module Parte_Operativa(dav_,rfd,numero,b2,b1,b0,c1,c0, clock,reset_);
 input        clock,reset_;
 input        rfd;
 output       dav_;
 output [7:0] numero;
 input        b2,b1,b0;
 output       c1,c0;
 reg          DAV_;     assign dav_=DAV_;
 reg [7:0]    NUMERO;   assign numero=NUMERO;
 assign c1 = NUMERO[7] & NUMERO[6] & NUMERO[5] & NUMERO[4] &
             NUMERO[3] & NUMERO[2] & NUMERO[1]&NUMERO[0];
 assign c0 = rfd;

 //reg DAV_
 always @(reset_==0) begin DAV_<=1; end
 always @(posedge clock) if (reset_==1) #3
   casex({b1,b0})
    'B00:  begin DAV_<=0; end
    'B01:  begin DAV_<=1; end
    'B1?:  begin DAV_<=DAV_; end
   endcase

 //reg NUMERO
 always @(reset_==0) begin NUMERO<=1; end
 always @(posedge clock) if (reset_==1) #3
   casex(b2)
    0:  begin NUMERO<=NUMERO; end
    1:  begin NUMERO<=NUMERO+2; end
   endcase
endmodule

module Parte_Controllo(b2,b1,b0,c1,c0, clock,reset_);
 input     clock,reset_;
 input     c1,c0;
 output    b2,b1,b0;
 reg [1:0] STAR;     parameter S0=0,S1=1,S2=2,S3=3;
assign {b2,b1,b0}=(STAR==S0)?'B000:(STAR==S1)?'B001:(STAR==S2)?'B11X:'B01X;
 //reg STAR
 always @(reset_==0) begin STAR<=S0; end
 always @(posedge clock) if (reset_==1) #3
   casex(STAR)
     S0:  begin STAR<=(c0==1)?S0:S1; end
     S1:  begin STAR<=(c0==1)?S2:S1; end
     S2:  begin STAR<=(c1==1)?S3:S0; end
     S3:  begin STAR<=S3; end
   endcase
endmodule
```