1. 设计一个序列检测器，要求检测器连续收到串行码1101后，输出检测标志1，否则输出0

Moore 型

S0: 默认状态

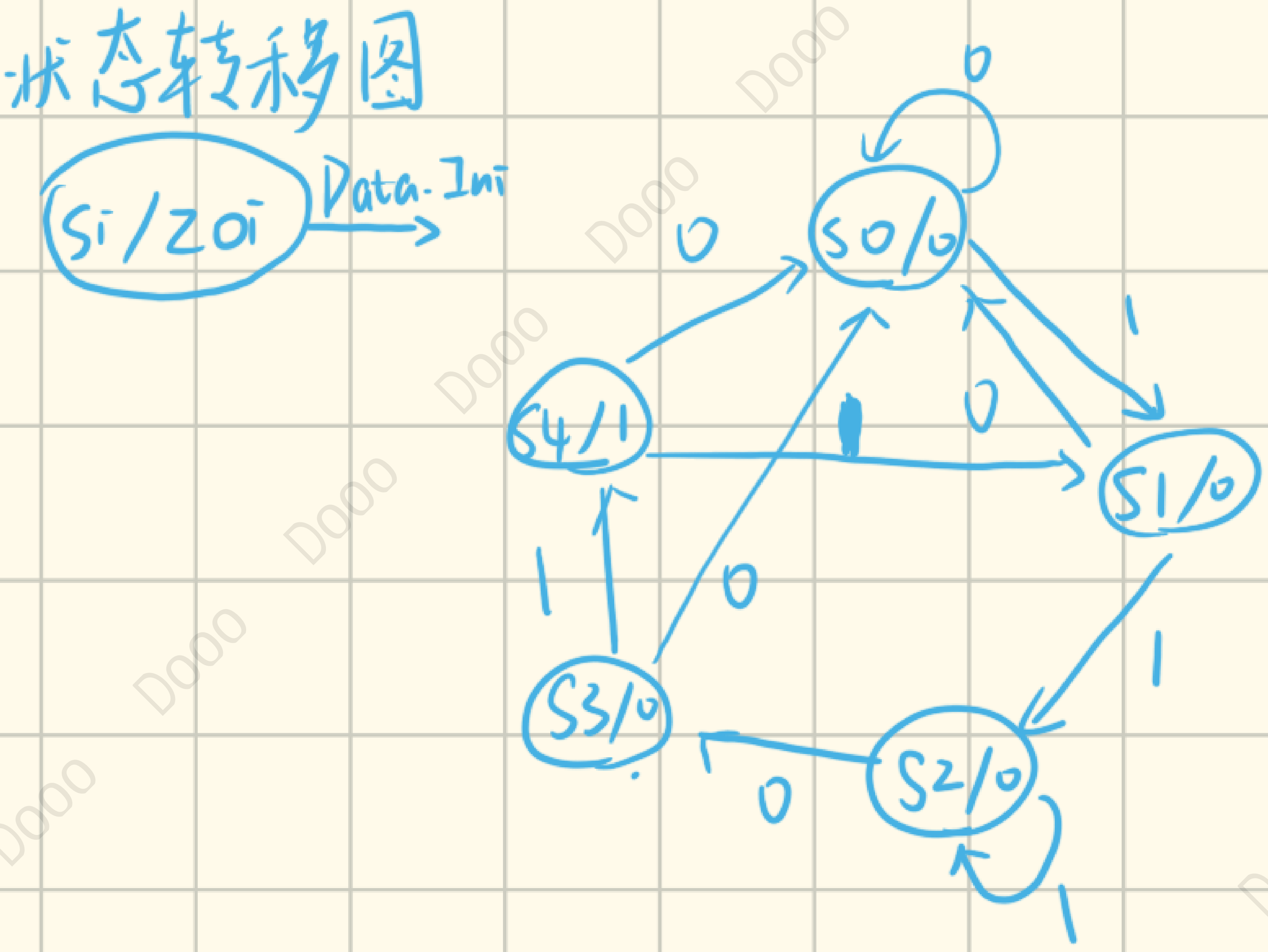
S1: 收到1

S2: 收到11

S3: 收到110

S4: 收到1101

状态转移图



HDL代码

```
'timescale 1ns/1ps
module seq_detect_moore(
  input clk,
  input rst_n,
  input din,
  output dout);
  localparam S0=3'b000,
             S1 = 3'b001,
             S2 = 3'b010,
             S3 = 3'b011,
             S4=3'b100;
  reg [2:0] state_current;
  reg [2:0] state_next;
```

```
① always @(posedge clk or negedge rst_n)
    begin
     if(!rst_n)
       state_current <= S0;
     else
```

```
state_current <= state_next;
end
```

```
②. always @ (*)
  begin
   case (state_current)
    S0: if (din)
          state_next = S1;
         else
         state_next = S0;
    S1: if (din)
         state_next = S2;
         else
         state_next = S0;
    S2: if (din)
         state_next = S2;
         else
         state_next = S3;
    S3: if (din)
         state_next = S4;
         else
         state_next = S0;
    S4: if (din)
         state_next = S1;
         else
          stato_next = S0;
   default : state_next = S0;
   endcase
  end
```

```
③ assign dout = (state_current == S4) ?
                                        1'b1 : 1'b0 ;

  endmodule
```

设计一个序列检测器，要求检测器连续收到串行码1101后，输出检测标志1，否则输出0

Mealy型.

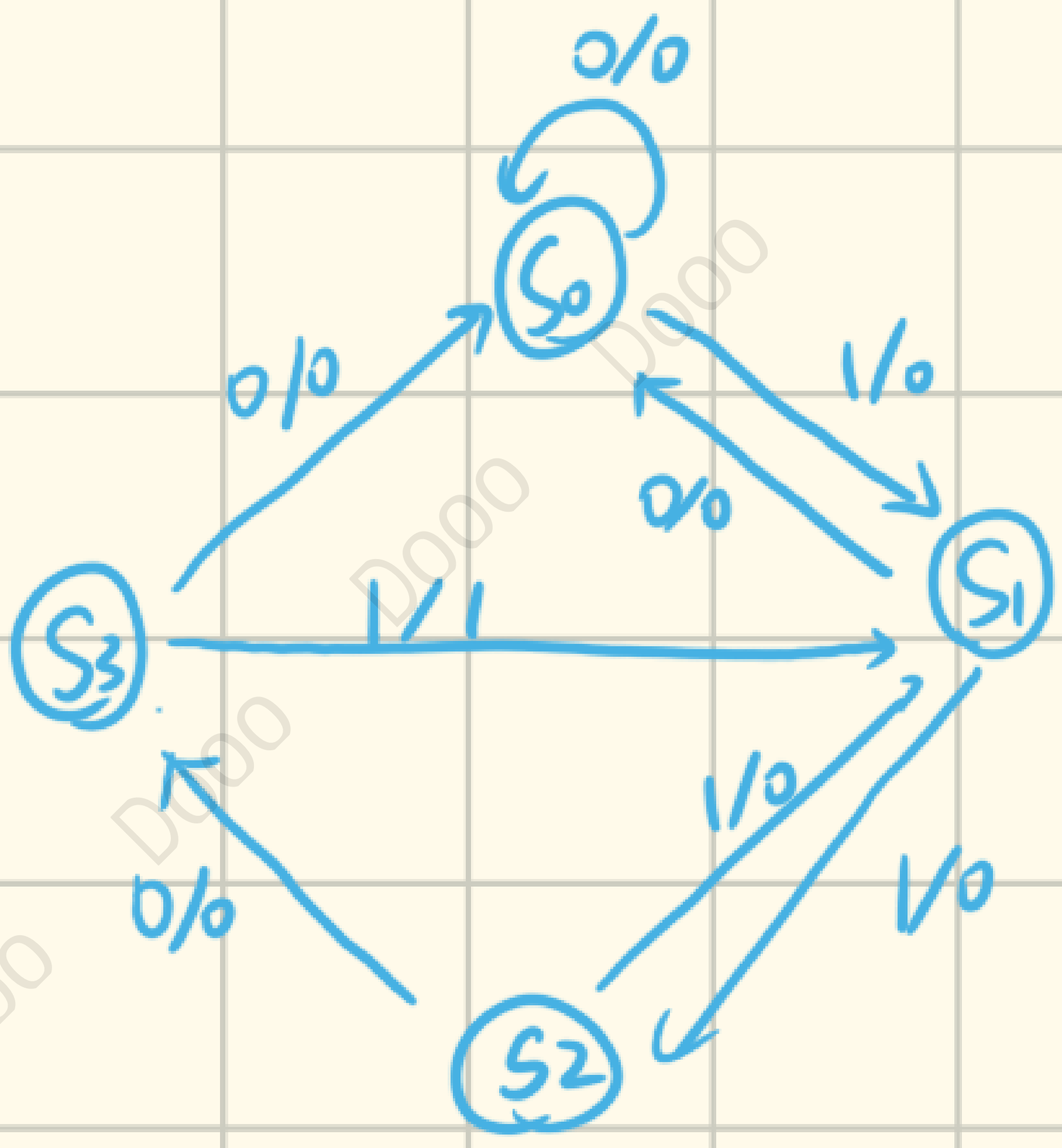
S0:默认状态.

S1:收到1

S2:收到11

S3:收到110.

状态转移图.



HDL代码.

```
'timescale 1ns/1ps
module seq_detect_mealy(
 input clk,
 input rst_n,
 input din,
 output dout);
 localparam s0=2'b00,
            s1=2'b01,
            s2=2'b10,
            s3=2'b11;
 reg[1:0] state_current;
 reg [1:0] state_next;
① always @(posedge clk or negedge rst_n)
   begin
     if(!rst_n)
       state_current <= S0;
     else
       state_current <= state_next;
   end.
```

```
② always@(*)
   begin
     case (state_current)
     S0: if(din)
           state_next = S1;
          else
           state_next = S0;
     S1: if(din)
           state_next = S2;
          else
           state_next = S0;
     S2: if(din)
           state_next = S2;
          else
           state_next = S3;
     S3: if(din)
           state_next = S0;
          else
           state_next = S0;
     default: state_next = S0;
     endcase
     end
```

```
③ assign dout = ((state_curt==S3)&(din==1'b1)) ?
                                               1'b1 : 1'b0;

   endmodule.
```

2. 设计一个序列检测器. 要求检测器连续收到串行码{111}后, 输出检测标志1, 否则输出0。

Moore 型.

S0: 默认状态.

S1: 收到1

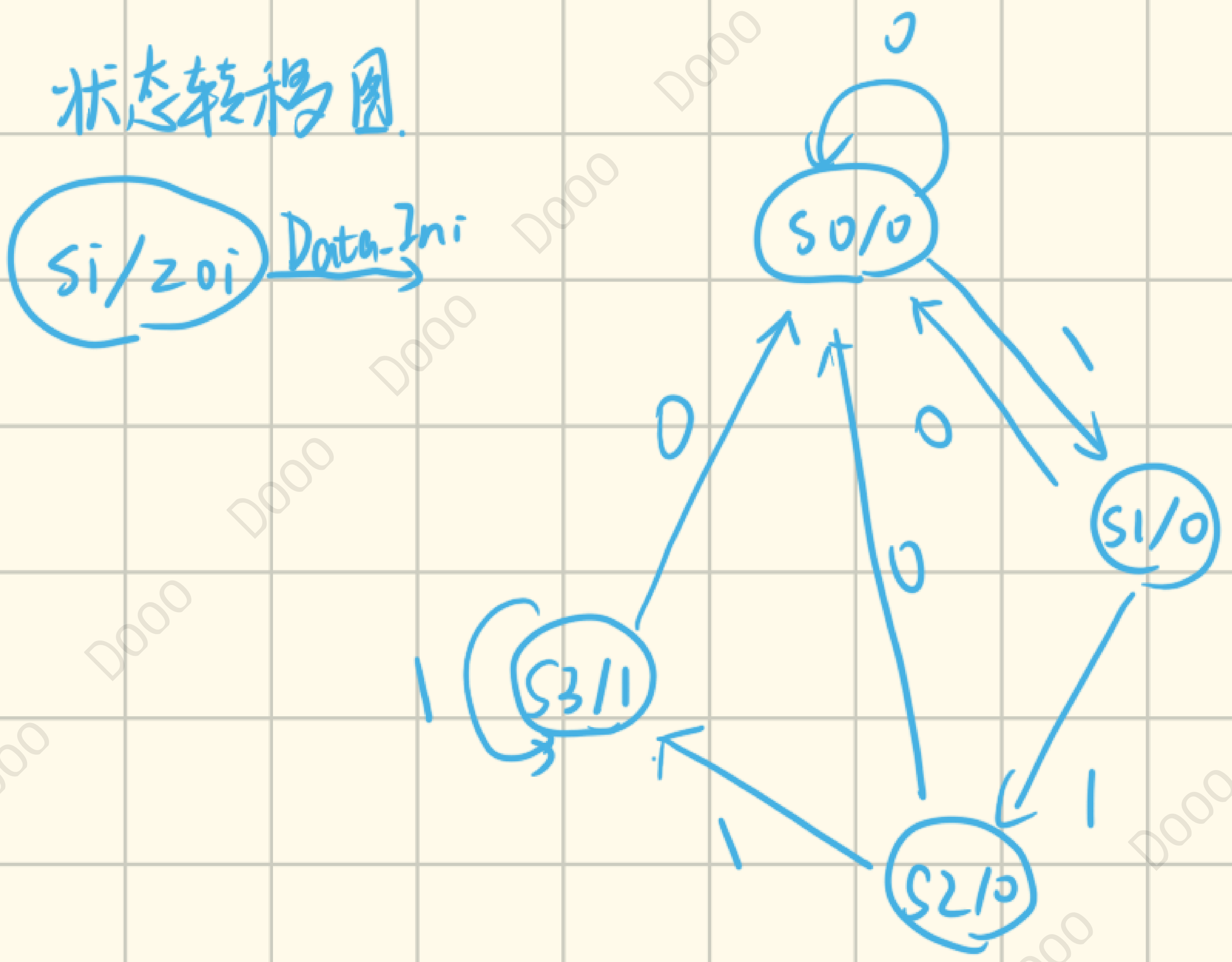
S2: 收到11

S3: 收到111.

状态转移表.

现态	次态 x=0	次态 x=1	输出
S0	S0	S1	0
S1	S0	S2	0
S2	S0	S3	0
S3	S0	S3	1

状态转移图.



Mealy 型.

S0: 默认状态.

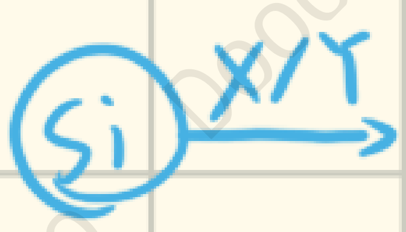
S1: 收到1

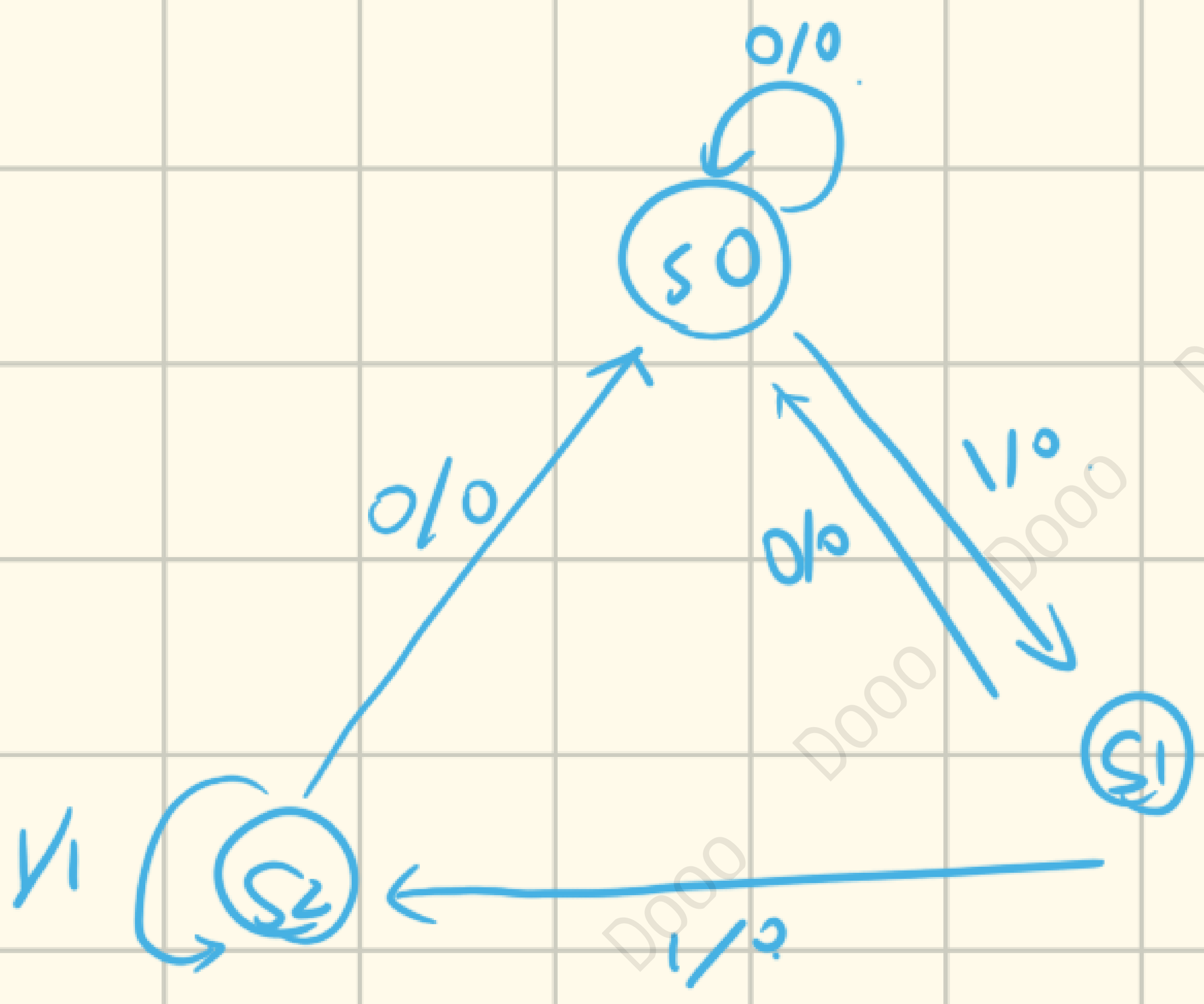
S2: 收到11.

状态转移表.

现态	次态 x=0	次态 x=1	输出 x=0	输出 x=1
S0	S0	S1	0	0
S1	S0	S2	0	0
S2	S0	S2	0	1

状态转移图.





HDL代码. Moore 型.

```
'timescale 1ns/1ps
module seq_detect_moore(
  input clk,
  input rst_n,
  input din,
  output dout);
  localparam s0=2'b00,
             s1=2'b01,
             s2=2'b10,
             s3=2'b11;
  reg[1:0] state_current;
  reg [1:0] state_next;
① always @(posedge clk or negedge rst_n)
  begin
    if(!rst_n)
      state_current <= s0;
    else
      state_current <= state next;
  end
```

```
② always @(*)
   begin
   case(state_current)
   s0: if(din)
         state_next= s1;
       else
         state_next=s0;
   s1: if(din)
         state_next=s2;
       else
         state_next=s0;
   s2: if(din)
         state_next=s3;
       else
         state_next=s0;
   s3: if(din)
         state_next=s3;
       else
         state_next=s0;
   endcase
   end.
```

```
③ assign dout =(state_current== s3)? 1'b1 : 1'b0;
  endmodule.
```

HDL代码. Mealy型.

```
'timescale 1ns/1ps
module seq-detect-mealy(
  input clk,
  input rst-n,
  input din,
  output dout);
  localparam S0=2'b00,
             S1=2'b01,
             S2=2'b10,
  reg [1:0] state_current;
  reg [1:0] state_next;
① always @(posedge clk or negedge rst_n)
  begin
    if(!rst_n)
     state_current<=S0;
    else
     state_current<=state_next;
    end.
```

```
② always @(*)
   begin
    case(state_current)
     S0: if(din)
           state_next = S1;
         else
           state_next = S0;
     S1: if(din)
           state_next=S2;
         else
           state_next=S0;
     S2: if(din)
           state_next=S2;
         else
           state_next=S0;
     default: state_next = S0;
    endcase
   end.
```

```
③ assign dout = ((state_current==S2)&(din=1'b1))? 1'b1:1'b0;
  endmodule.
```