

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。

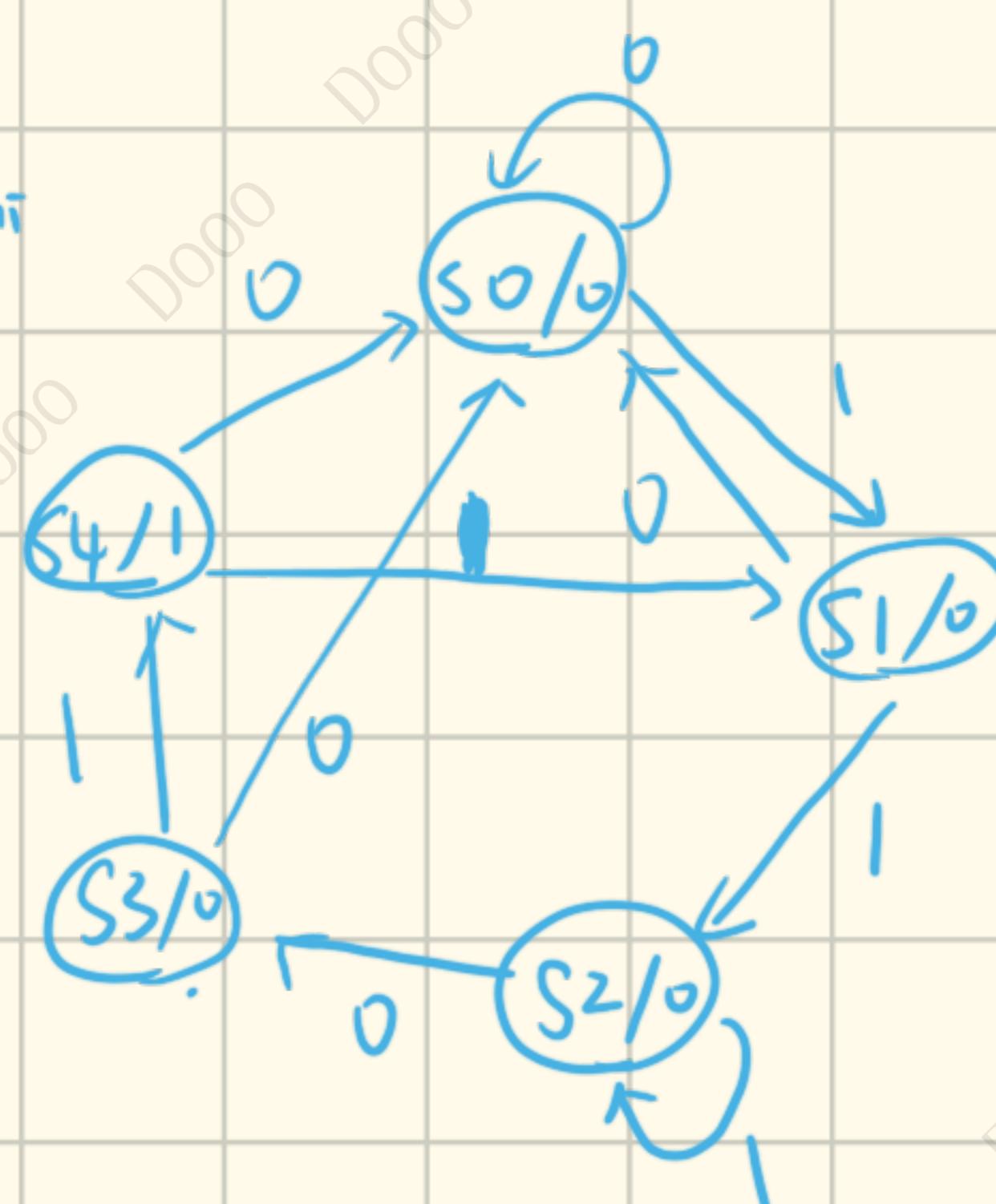
S2：收到 11。

S3：收到 110。

S4：收到 1101。

状态转移图

(S_i/Z_{oi}) → Data_Ini



HDL 代码。

'timescale 1ns/1ps

module seq_detect_moore(

input clk,

input rst_n,

input din,

output dout);

local param 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)

state_current <= state_next;

begin

end.

if (!rst_n)

state_current <= S0;

else

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

state_next = S0;

default : state_next = S0;

endcase

end

③ assign dout = (state_current == S4) ?

i'b1:i'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。

状态转移图。

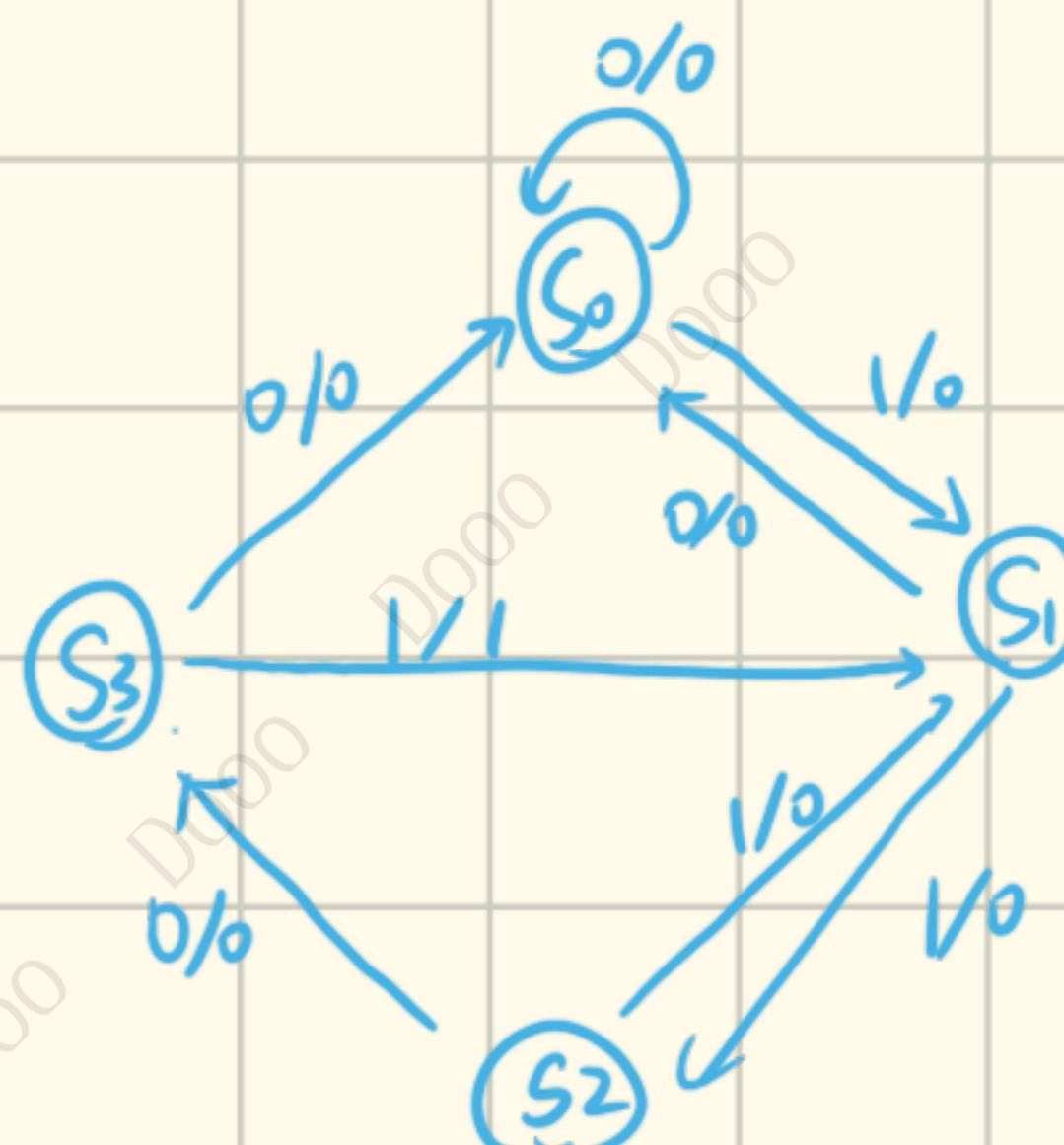
S0：默认状态。



S1：收到 1。

S2：收到 11。

S3：收到 110。



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@(*)

③ assign dout = ((state_current == S3) & (din == 1'b1)) ?

1'b1 : 1'b0;

begin

case (state_current) endmodule.

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

2. 设计一个序列检测器。要求检测器连续收到串行码 1111 后，输出检测标志，否则输出 0。

Moore 型。

S_0 : 默认状态。

S_1 : 收到 I

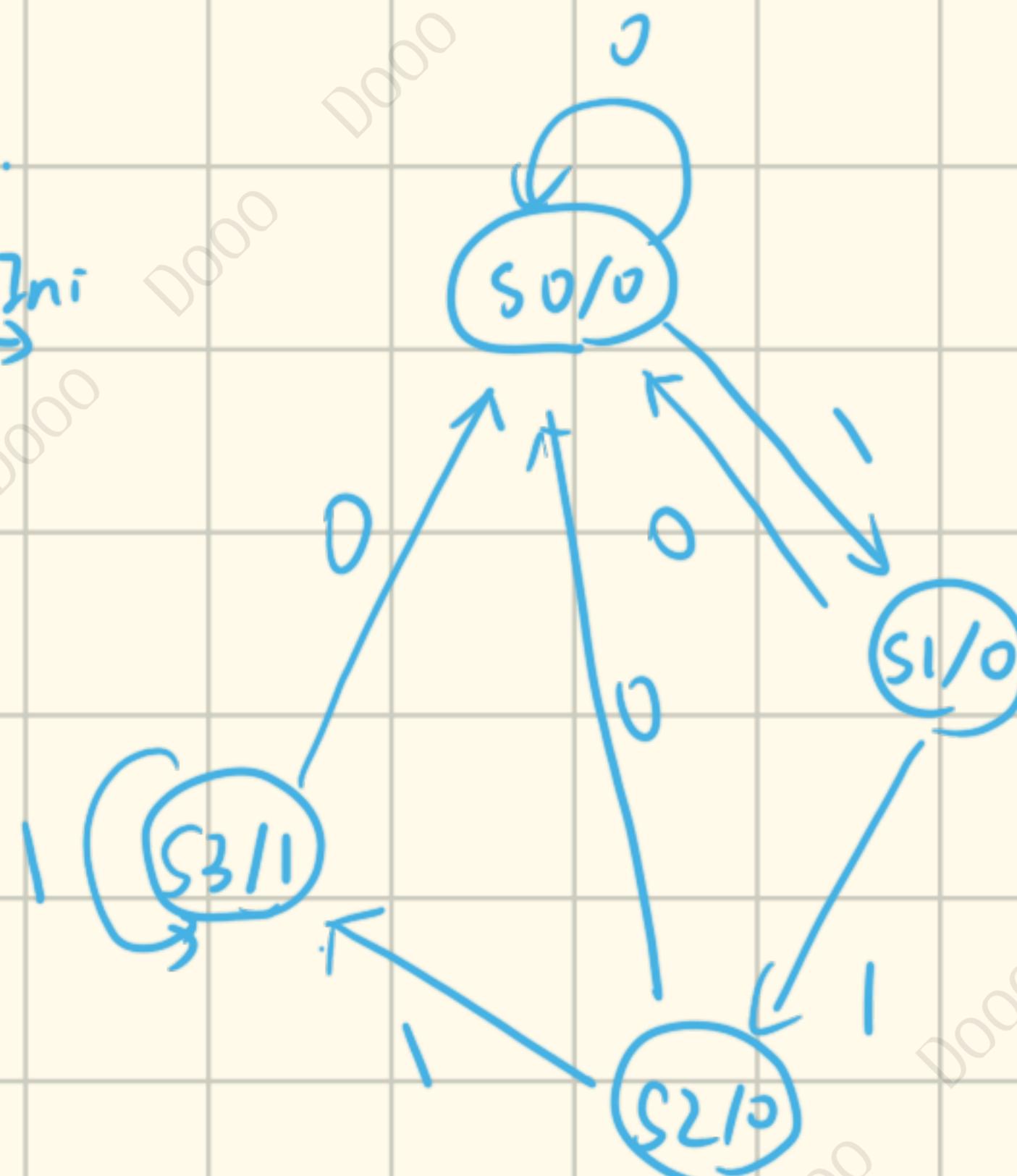
S_2 : 收到 II

S_3 : 收到 III

状态转移表。

现态	次态		输出
	$x=0$	$x=1$	
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_3	0
S_3	S_0	S_3	1

状态转移图。



Mealy 型。

S_0 : 默认状态。

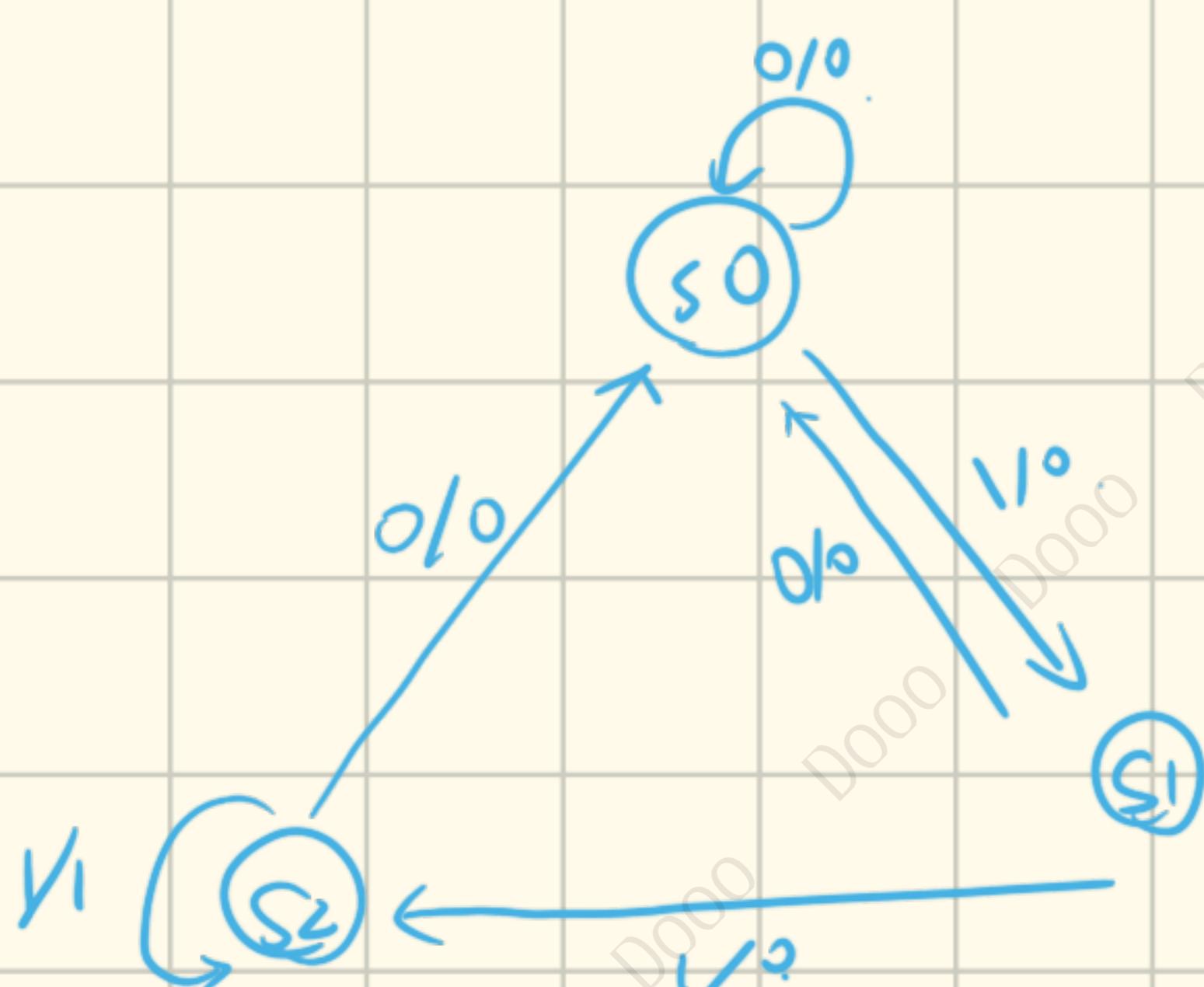
S_1 : 收到 I

S_2 : 收到 II

状态转移图。

S_i X/Y

现态	次态		输出
	$x=0$	$x=1$	
S_0	S_0	S_1	0 0
S_1	S_0	S_2	0 0
S_2	S_0	S_2	0 1



HDL Fib. Moore 型

'timescale 1ns/1ps

②. always @ (*)

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) state_next=S0;

begin

if (!rst-n)

state_current <= S0;

else

state_current <= state_next;

end

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 틀 2. 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;

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;

① always @posedge clk or @nedge rst-n) S2: if (din)

begin

if (!rst-n)

state-current <= S0;

state-next = S2;

else

state-next = S0;

else

default: state-next = S0;

state-current <= state-next;

endcase

end.

③ assign dout = ((state-current == S2) & (din == 1'b1)) ? 1'b1 : 1'b0;

endmodule.

