**实验三 时序电路设计实验报告**

**姓名**：  **学号**：1120190699

**班级**： **手机**：

1. **实验题目**

设计一个串行数据子序列检测器。当连续输入 4 个或 4 个以上的 0 时，输出为 1，其他情况下输出为 0。

1. **电路设计**
   1. **规范化**

串行数据子序列检测器是一个检测输入序列中是否带有连续4个或以上0的时序电路。这个电路的输入为标量X，代表串行数据的当前值；电路的输出为标量Z，默认值为0。电路初始时X=1，Z=0，处于状态S；如果输入X=1，不管处于何种状态，立刻回到状态S；如果在状态S下，输入X=0，转移到状态A；如果在状态A下，输入X=0，转移到状态B；如果在状态B下，输入X=0，转移到状态C；如果在状态C下，输入X=0，转移到状态D；如果在状态D下，输入X=0，则输出Z=1。

* 1. **形式化**

输入变量：X。X=0代表当前在序列中读到了一个0，X=1代表当前在序列中读到了一个1。

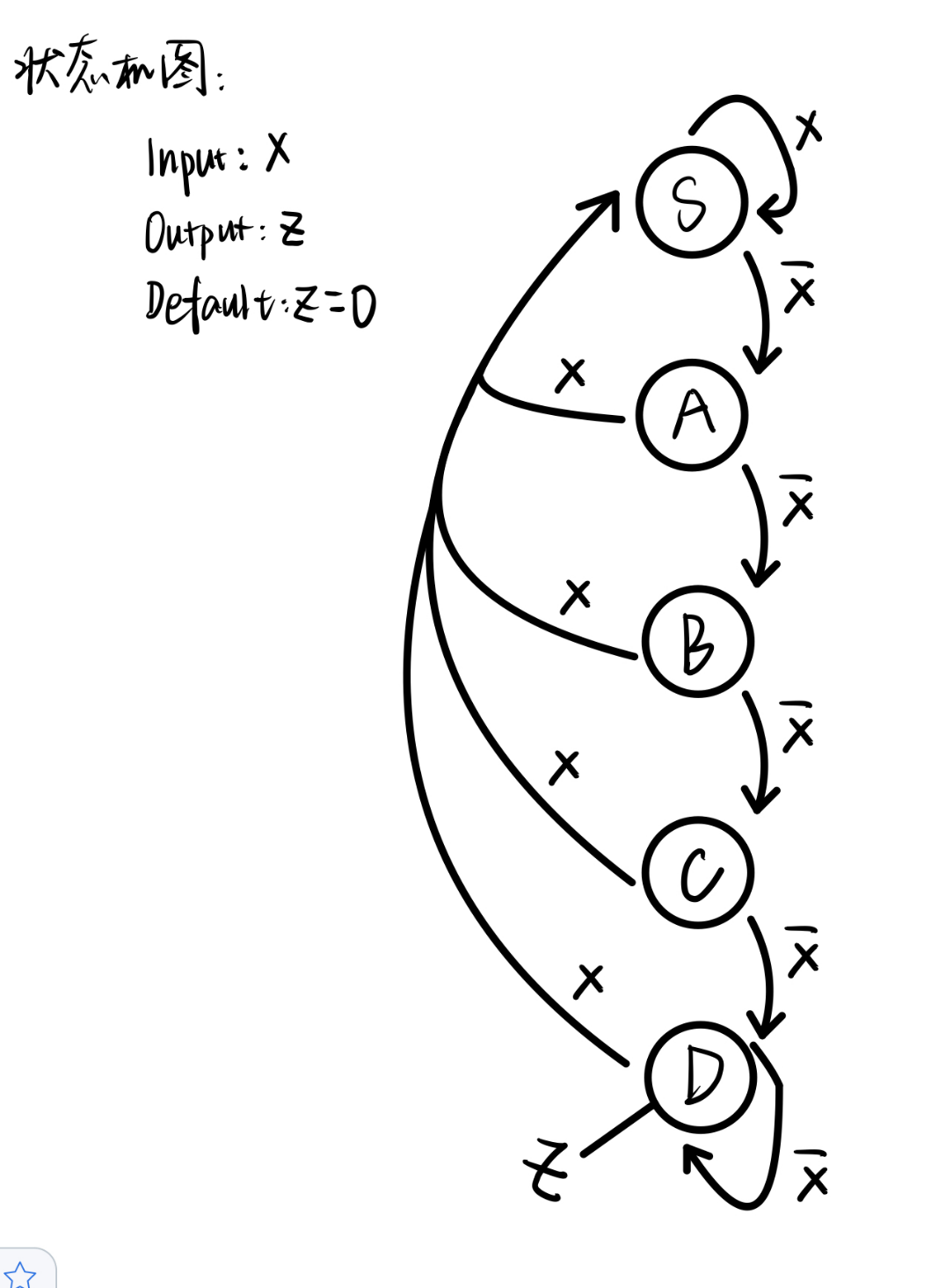
输出变量：Z。Z=0代表还没有连续输入4个或4个以上的0，Z=1代表连续输入了4个或4个以上的0。

状态：S：初始状态，代表一个0都没有输入，这可能发生在序列输入前或者序列输入了一个1。S->A,X=0。

A：代表输入了一个0。A->B,X=0。A->S,X=1。

B：代表输入了两个0。B->C,X=0。B->S,X=1。

C：代表输入了三个0。C->D,X=0。C->S,X=1。

D：代表输入了四个0。D->S,X=1。X=0，Z=1。

* 1. **状态分配**

S：000

A：001

B：010

C：011

D：100

1. **电路实现**

/\*通过定义三个独立的、并行的进程来实现状态图，因此使用事件控制语句\*/

module SeqDetect(clk,reset,X,Z);

/\*声明并定义输入输出，默认输入都是wire型变量，

实际上输入也只能是wire型变量

而输出因为需要在always语句内部被赋值，

而在always内部被赋值的只能是reg类型变量，

所以只能将它定义为reg型变量\*/

input clk,X,reset;

output reg Z;

/\*用3bit位的寄存器数组来存储当前状态和下一状态\*/

reg [2:0] state,next\_state;

//用参数语句为S、A、B、C、D五个状态赋状态代码，它们都是3bit位的

parameter S=3'b000,A=3'b001,B=3'b010,C=3'b011,D=3'b100;

/\*第一个进程：描述用于存储序列识别器状态的状态寄存器

因为是时序逻辑，用if-else语句、并行执行的非阻塞赋值语句

每次clk从0变为1，或者当reset为1，该进程将会被执行

如果此时reset为1，还未开始检测输入序列，初始化当前状态为状态S

如果reset为0，代表仿真已经开始，在时钟的上升触发沿

将下一状态的值赋给当前状态\*/

always @(posedge clk or posedge reset)

begin

if (reset)

state<=S;

else

state<=next\_state;

end

/\*第二个进程：描述下一状态函数，定义状态转移条件

由于该进程是组合逻辑，用顺序执行的阻塞赋值语句，

无论何时只要输入发生变化或者状态发生变化，进程就会被执行

因为这里需要用到多路判断，所以用case语句而不是if-else

这里实现的是基于当前状态和输入X判断下一状态

其实就是实现了状态机图里面的每一条线\*/

always @(X or state)

begin

case(state)

S: next\_state = X ? S : A;

A: next\_state = X ? S : B;

B: next\_state = X ? S : C;

C: next\_state = X ? S : D;

D: next\_state = X ? S : D;

endcase

end

/\*第三个进程：描述输出函数，给输出Z赋值

由于该进程是组合逻辑，用顺序执行的阻塞赋值语句

执行进程的条件和第二个进程相同

本电路中，输出行为是TCI

在状态S、A、B、C中，输出都是默认值0

在状态D中，如果输入X=0，那么输出1\*/

always @(X or state)

begin

case(state)

S: Z=1'b0;

A: Z=1'b0;

B: Z=1'b0;

C: Z=1'b0;

D: Z=X ? 1'b0 : 1'b1;

endcase

end

endmodule

1. **电路验证**
   1. **TestBench**

module testbench();

/\*线型变量Z存储被测module反馈的输出

寄存器变量clk，X，reset为被测module提供输入\*/

wire Z;

reg clk,X,reset;

/\*定义被测序列为一个11 bit位的数组

存入寄存器内由低位到高位存储

定义一个integer在for循环语句中使用

初始化integer为0\*/

reg [0:10] test\_sequence=11'b100\_0010\_0010;

integer i=0;

SeqDetect test(clk,reset,X,Z);

/\*提供时钟周期信号，每隔50纳秒改变一次\*/

always begin

clk=1'b1;

#50;

clk=1'b0;

#50;

end

/\*仿真开始时，进程激活reset，使状态机初始化，处于状态S

reset失效后，用for循环语句每隔100纳秒

加载存储在test\_sequence中的X输入值

输入值加载的时间是在时钟的上升沿

最后用stop语句结束仿真\*/

initial begin

reset=1'b1;

X=1'b0;

#20;

reset=1'b0;

for(i=0;i<11;i=i+1)

begin

X=test\_sequence[i];

#100;

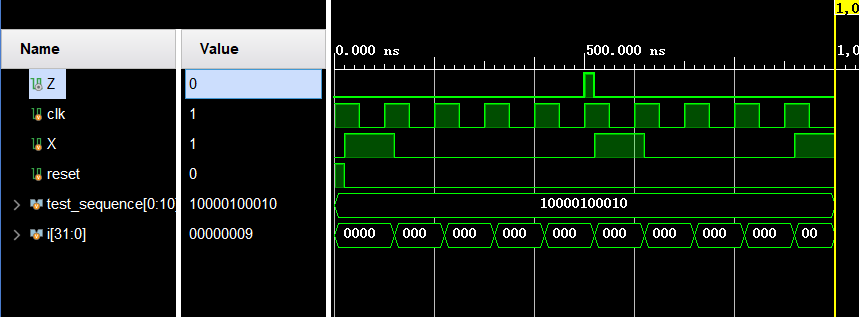
end

$stop;

end

endmodule

* 1. **仿真结果**



仿真期间，clk信号每隔50纳秒改变一次，波形中也可以看出clk的值做1-0周期性重复。

仿真开始前20纳秒，reset为1，此时序列检测状态机被初始化，处于状态S，module在状态S中Z输出为0。此时testbench中初始化X输入为0，被激活的序列检测状态机在状态S下输入X=0，因此转移到状态A。

在reset信号失效后，testbench开始加载存储在test\_sequence中的X输入值，加载时总是在clk的上升沿，因此序列检测状态机的状态总会由于X输入的改变而改变。输入值每隔100纳秒变换到下一位，可以看见i值每隔100纳秒就进行一次加一变换。首先加载的是最低位test\_sequence[0]，波形中也可见X=1，序列检测状态机处于状态S。接下来加载的是test\_sequence[1]，X=0，一直到test\_sequence[4]，输入都是0。在第500纳秒的时钟上升沿，序列检测状态机经历状态A、B、C，改变到了状态D，则输出Z=1。但是马上输入X= test\_sequence[5]=0，所以输出Z=0。后面一直没有出现连续的四个0，所以Z=0。

1. **实验心得**