数字系统设计 第六次实验课

主要内容： lab6

Lab 6: Dedicate Microprocessor Lab

(2+2 hours)

Goal

To learn how to implement a dedicated microprocessor.

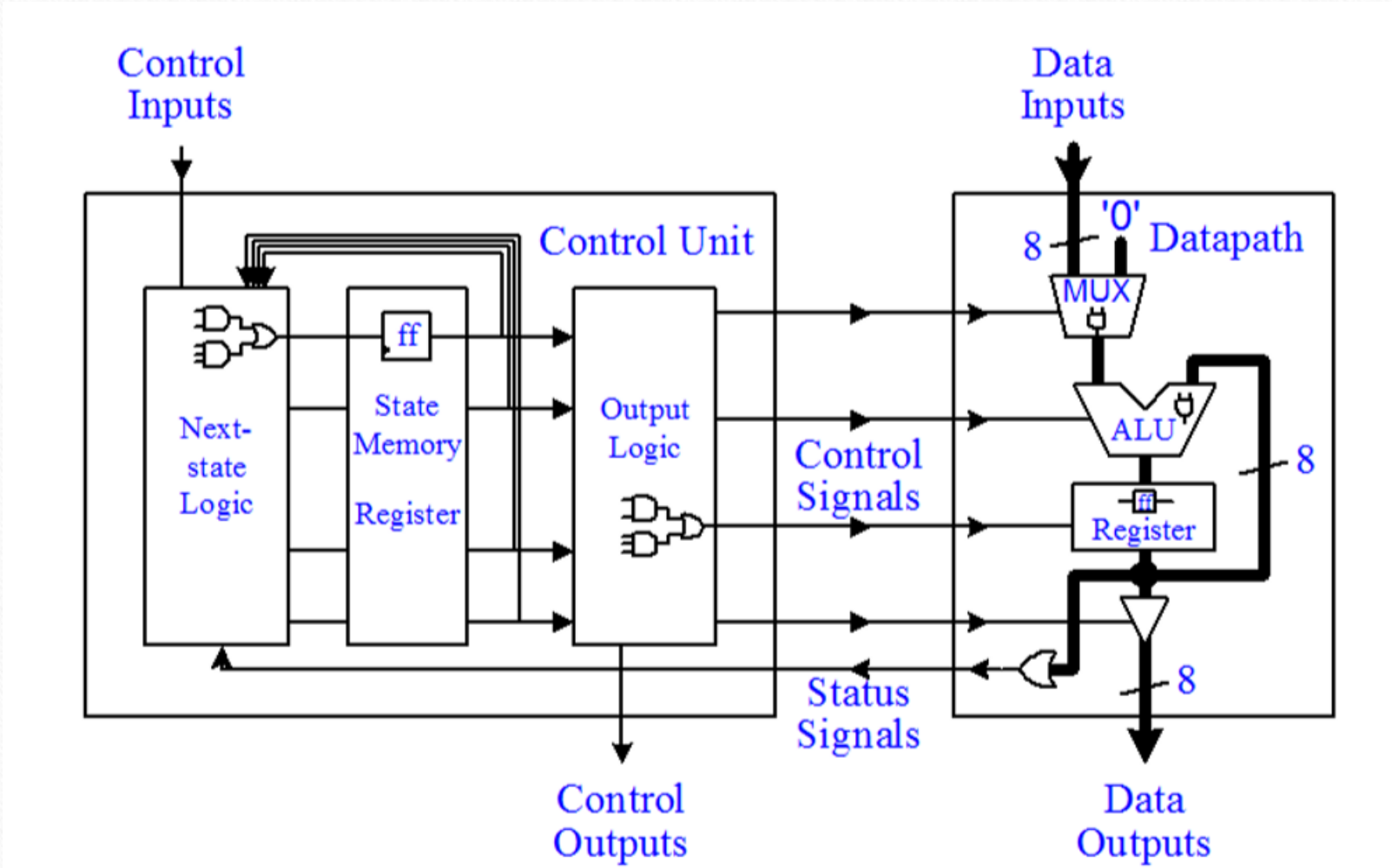
Procedure

1) Warm up: Implement the IF-THEN-ELSE example in Lecture 6 in your Basys 3 board. Connect the buttons and switches to the input pins of your module, and connect the leds to your output pins.

2) Manually design and implement on a FPGA a dedicated microprocessor to input one 8-bit value, and then determine whether the input value has an equal number of 0 and 1 bits. The microprocessor outputs a 1 if the input value has the same number of 0's and 1's; otherwise, it outputs a 0. For example, the number 10111011 will produce a 0 output; whereas, the number 00110011 will produce a 1 output. The algorithm is shown next. Draw the datapath and the corresponding FSM state diagram, FSM circuit, list the control words.

```
1     Count = 0              // for counting the number of 1 bits
2     INPUT N
3     WHILE (N ≠ 0){
4        IF (N(0) = 1) THEN  // least significant bit of N
5           Count = Count + 1
6        END IF
7        N = N >> 1           // shift N right one bit
8     }
9     OUTPUT (Count = 4) // output 1 if the test (Count = 4) is
                    true
```

3) Implement the datapath circuit and FSM circuit separately in Verilog module, connect them together by using a top module, and implement it in your Basys3 board. Connect the switches in board to your input pin, and connect the output pin to a LED.



Control unit

FSM输出控制字

Datapath

代码结构（FSM+D）

- Control_Unit（FSM）
- DataPath (D)
- （TOP）MicroProcessor

2) Manually design and implement on a FPGA a dedicated microprocessor to input one 8-bit value, and then determine whether the input value has an equal number of 0 and 1 bits. The microprocessor outputs a 1 if the input value has the same number of 0's and 1's; otherwise, it outputs a 0. For example, the number 10111011 will produce a 0 output; whereas, the number 00110011 will produce a 1 output. The algorithm is shown next. Draw the datapath and the corresponding FSM state diagram, FSM circuit, list the control words.

```
1     Count = 0                  // for counting the number of 1 bits
2     INPUT N
3     WHILE (N ≠ 0){
4        IF (N(0) = 1) THEN  // least significant bit of N
5           Count = Count + 1
6        END IF
7        N = N >> 1              // shift N right one bit
8     }
9     OUTPUT (Count = 4)  // output 1 if the test (Count = 4) is
                        true
```

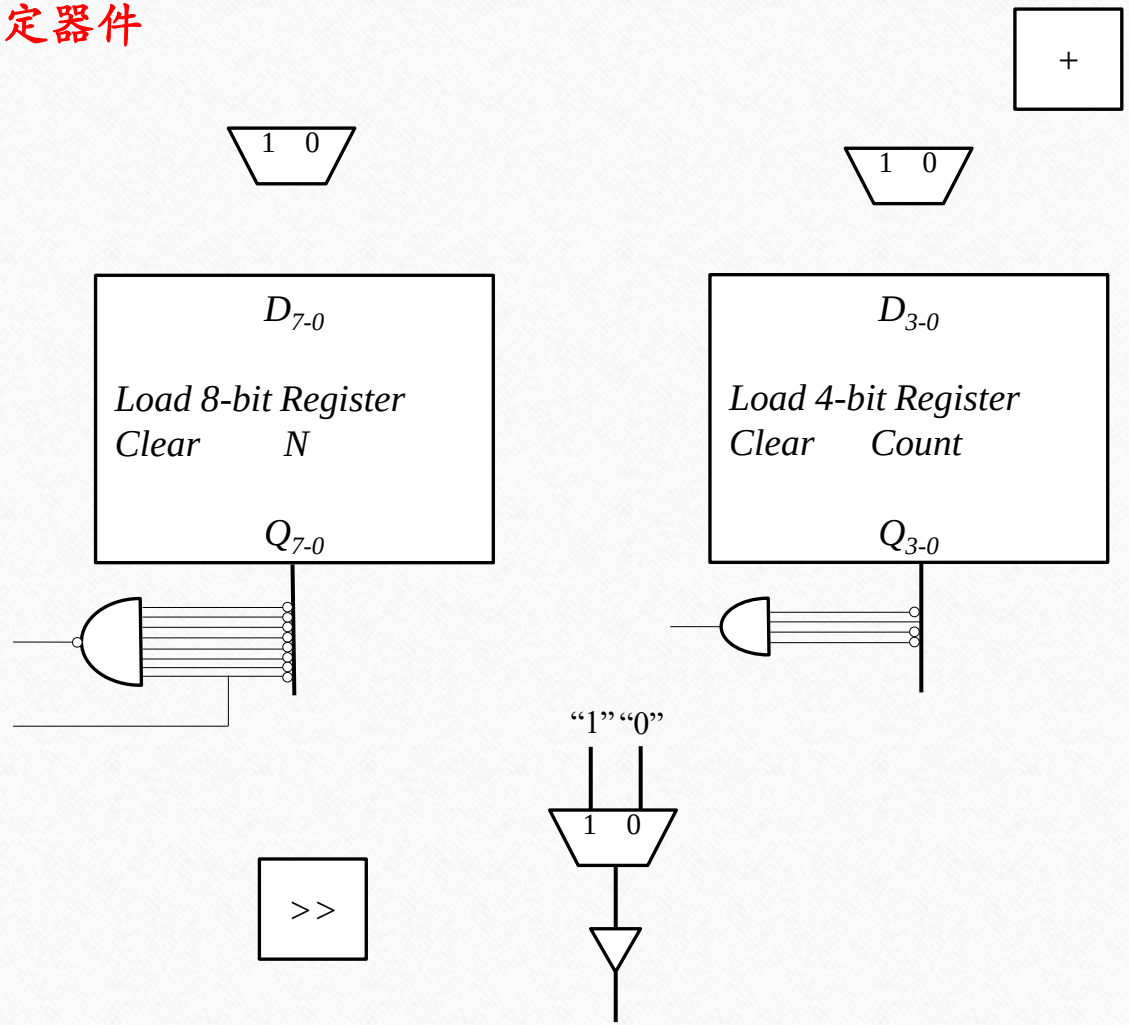
step1： 画出数据通路

step2：控制单元：画出状态转换图

step3：控制单元：输出控制字

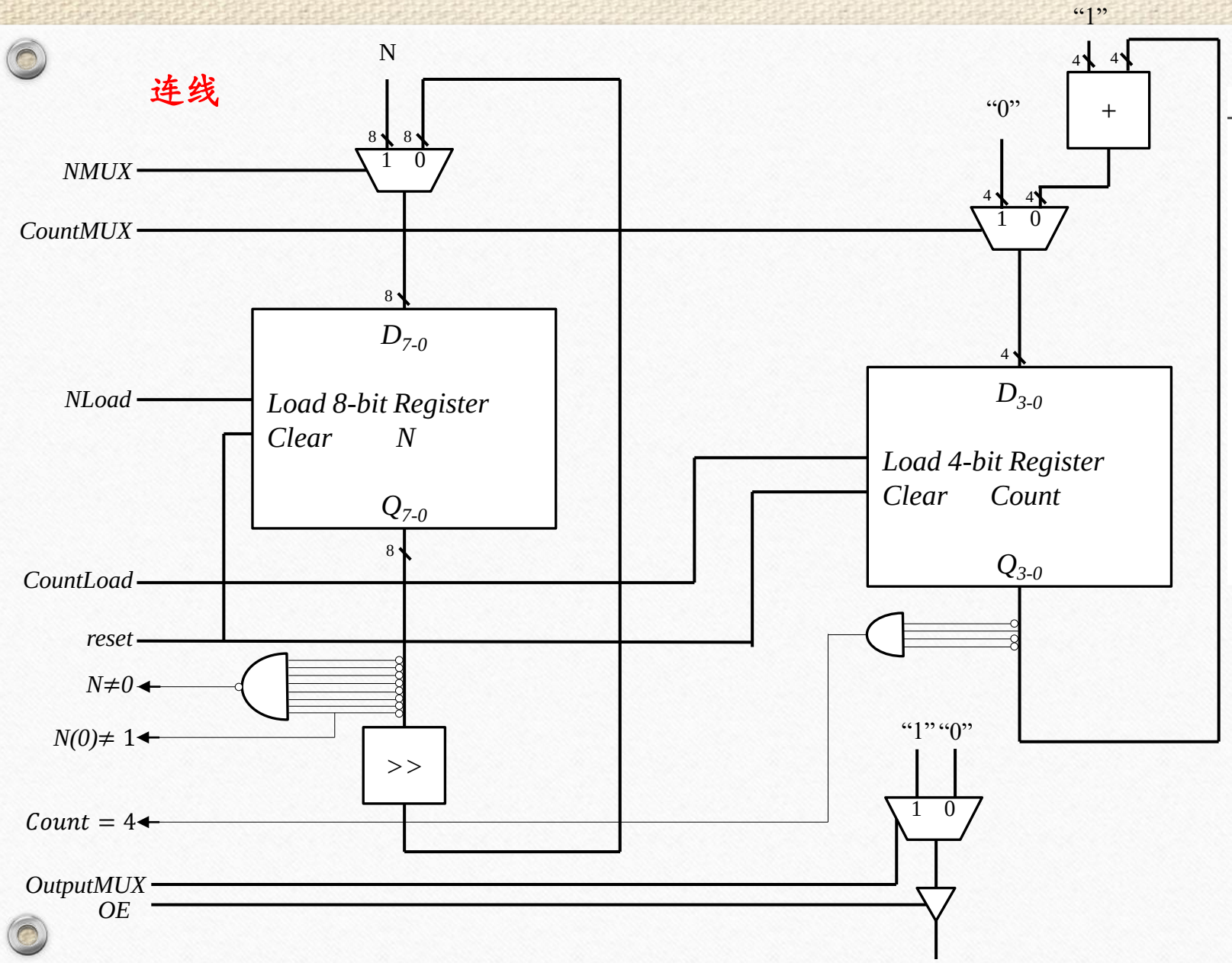
step1：画出数据通路

确定器件



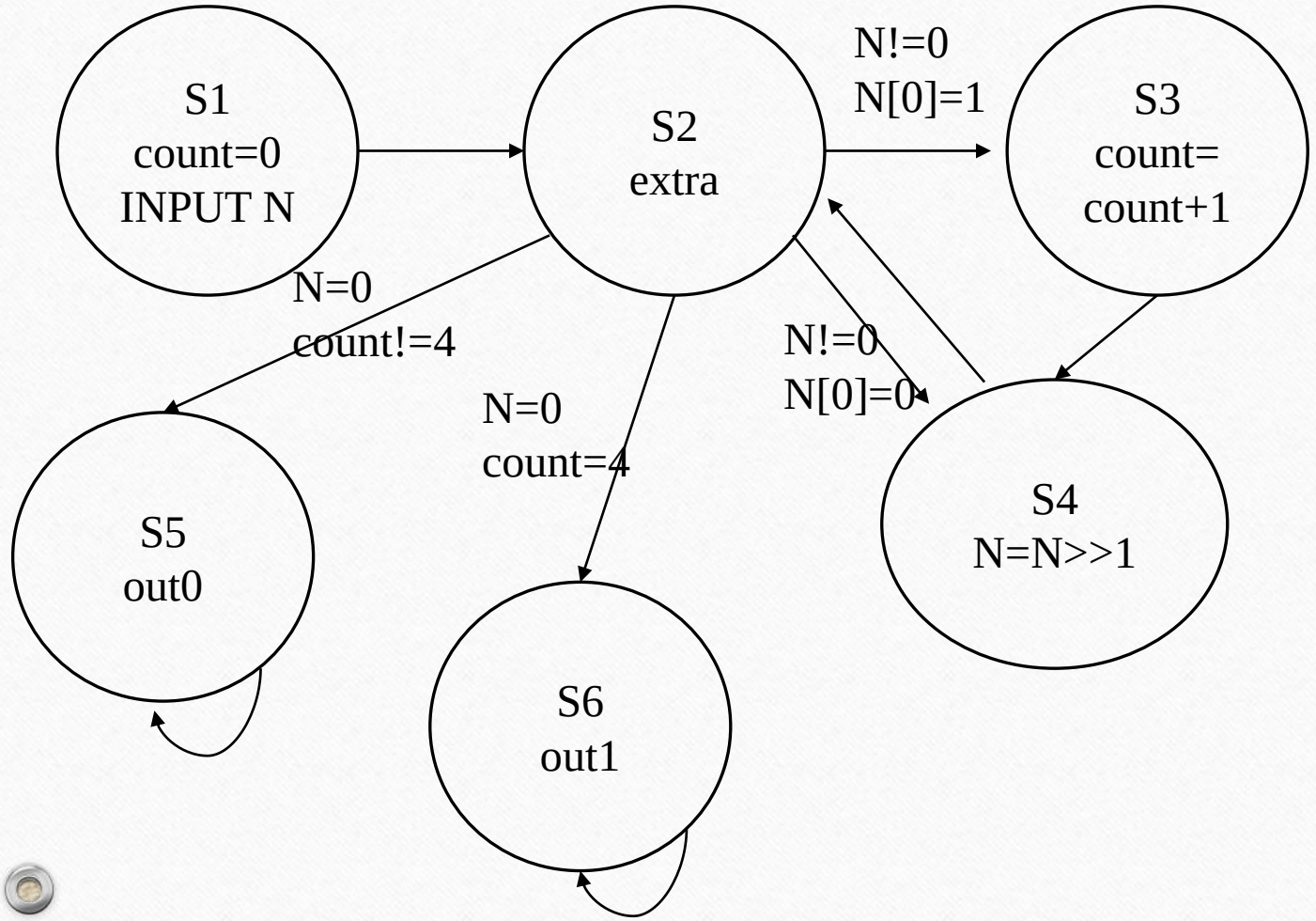

```
Count = 0                    //
INPUT N
WHILE (N ≠ 0){
    IF (N(0) = 1) THEN /
        Count = Count + 1
    END IF
    N = N >> 1               /
}
OUTPUT (Count = 4)  // ou
```

- 两个Register （count、N）
- 一个移位寄存器
- 一个加法器
- ？个mux
- 其他



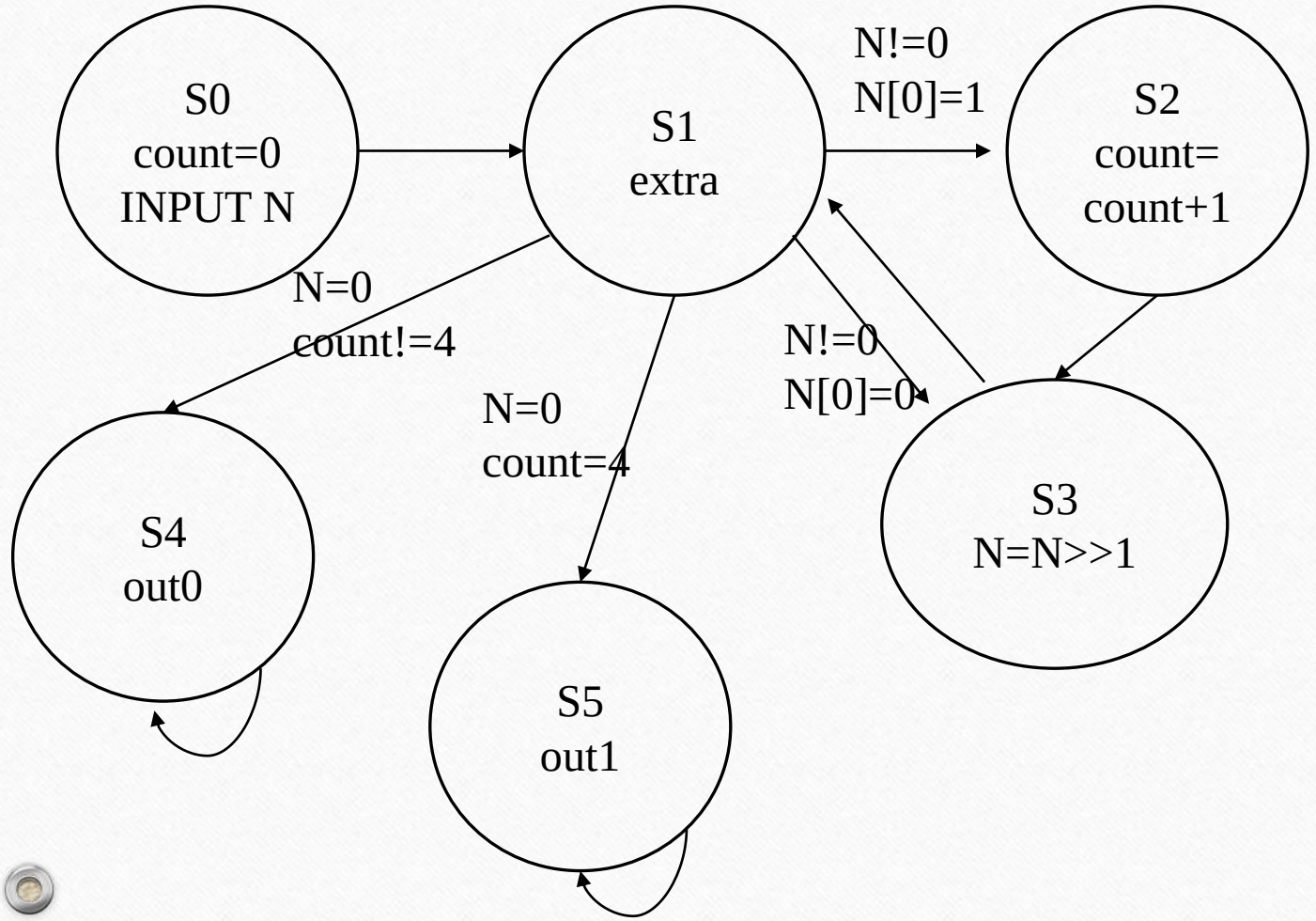

```
Count = 0                    //
INPUT N
WHILE (N ≠ 0) {
    IF (N(0) = 1) THEN /
        Count = Count + 1
    END IF
    N = N >> 1               /
}
OUTPUT (Count = 4)  // ou
```

step2：画出状态转换图（控制单元）



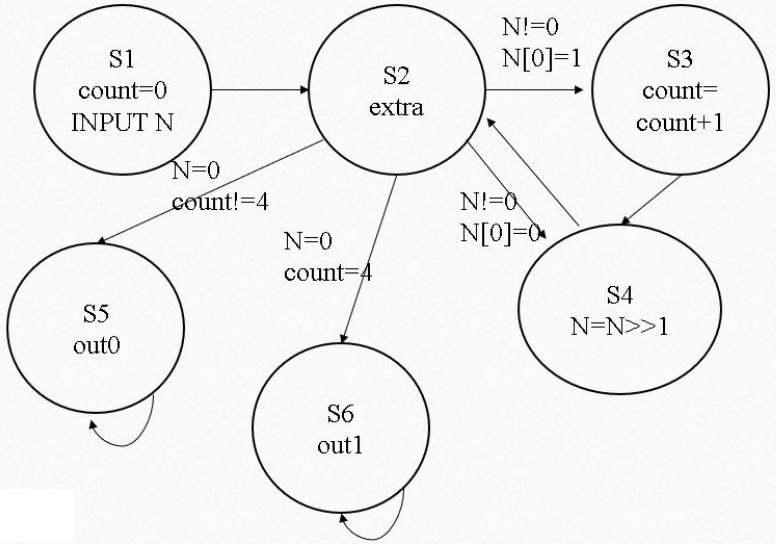

```
Count = 0                    //
INPUT N
WHILE (N ≠ 0){
    IF (N(0) = 1) THEN /
        Count = Count + 1
    END IF
    N = N >> 1               /
}
OUTPUT (Count = 4) // ou
```

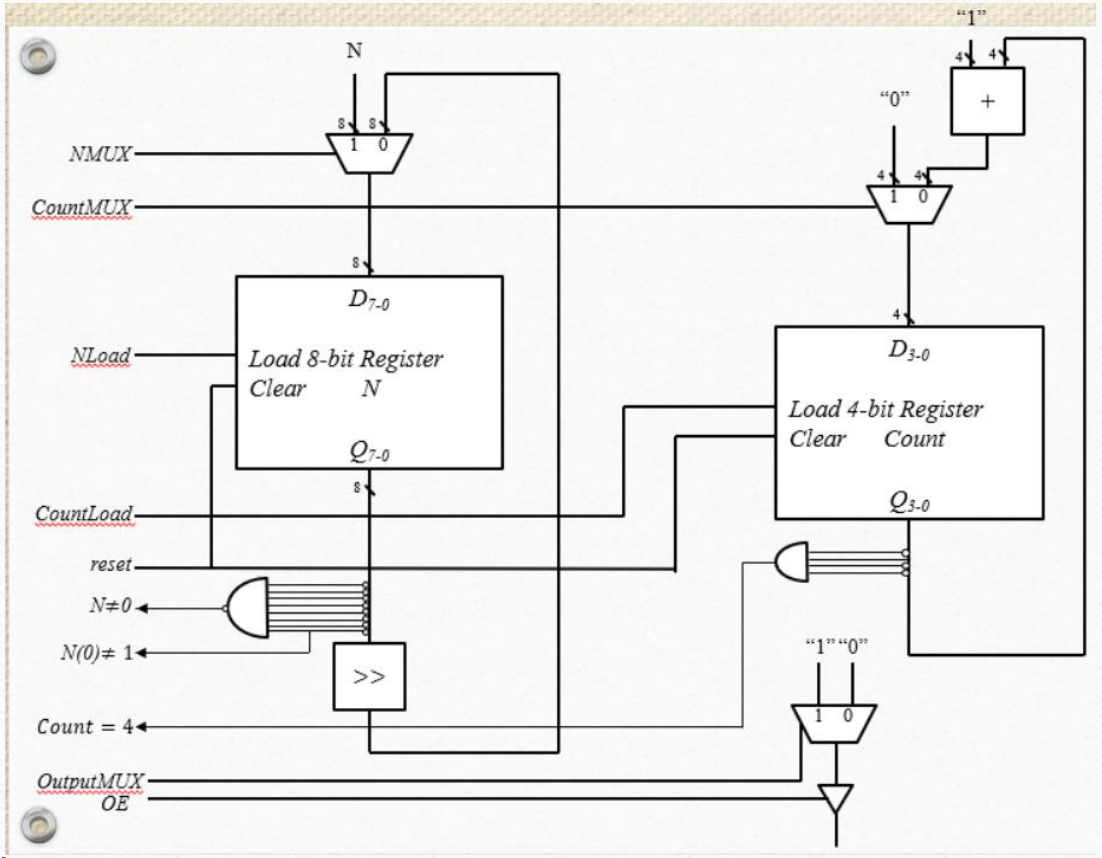
step2：画出状态转换图（控制单元）




```
Count = 0                    //
INPUT N
WHILE (N ≠ 0){
    IF (N(0) = 1) THEN /
        Count = Count + 1
    END IF
    N = N >> 1               /
}
OUTPUT (Count = 4) // ou
```

step3：输出控制字（控制单元）





状态反馈信号(count, N)

状态转换 ⟵ ⟶ 数据通路

控制字

step3：输出控制字（控制单元）

- 根据数据通路所需、确定每个状态的控制字

step3：输出控制字（控制单元）

	NMUX	*CountMUX*	*NLoad*	*CountLoad*	*OutputMUX*	*OE*
S1 count=0 INPUT N	*1*	*1*	*1*	*1*	*X*	*0*
S2 extra	*X*	*X*	*0*	*0*	*X*	*0*
S3 count= count+1	*X*	*0*	*0*	*1*	*X*	*0*
S4 N=N>>1	*0*	*X*	*1*	*0*	*X*	*0*
S5 out0	*X*	*X*	*0*	*0*	*0*	*1*
S6 out1	*X*	*X*	*0*	*0*	*1*	*1*

要点：

- 根据数据通路，搭建datapath模块(组合逻辑)
- 根据状态转换图写有限状态机（control unit模块），每个状态下输出特定控制字（时序逻辑）
- 用一个顶层模块将control unit模块和datapath模块连接起来。

要求：

- 实验报告（状态图、datapath、control word、仿真和实验结果）
- 验收

顶层代码

```
module Dedicate_Microprocessor(
    input clk,
    input reset,
    input [7:0] N,
 //   output [7:0] check,
    output led
    );

    wire NMUX, CountMUX, NLoad, CountLoad, OE, OutputMUX;
    wire NnotOne, NnotZero, CountFour;

    Control c1(.clk(clk), .rst(reset), .NnotZero(NnotZero), .NnotOne(NnotOne), .C

    Datapath d1(.clk(clk), .N(N), .rst(reset), .NMUX(NMUX), .CountMUX(CountMUX),

  //  assign check = N;

endmodule
```

寄存器代码

```
always @(posedge clk or posedge rst) begin

    if(rst) out <= 0;
    else if(load) out <= ina;

end
```

移位代码

```
    assign shiftedN = preN >> 1;
```