lab04 时序与状态机

张立夫 PB15020718

实验目的

- 1. 了解并掌握时序逻辑电路设计
- 2. 掌握三段式状态机的设计与使用

实验平台

• 操作系统: Windows 10

● 编译仿真环境: Xilinx ISE 14.7

实验要求

- 使用状态机控制运算过程(数据读取,计算,数据写入),每部加法运算所用时钟数不允许超过五个。典型的,三个clk读取数据和操作符,一个clk计算,一个clk向ram写入结果。
- 仿真激励文件模块只允许出现clk和rst信号输入。
- 实现一个control模块,完成整个运算的控制。
- 实现一个顶层模块Top
- 调用Ram模块
 - 。调用RegFile
 - 。 调用ALU完成加法运算
 - 。 调用control模块, 完成运算控制

实验过程

- 1. 创建 ram
 - 。 新建 IP 核选择 RAM & ROM
 - 。 选择简单双端口 RAM (一个读端口,一个写端口)
 - 。 选择宽度 32bit 深度 256
 - 。 生成 RAM 模块
- 2. 创建控制模块 control.v
 - 。 定义有限状态机以下状态:
 - idle 初始状态,对内存进行初始化赋值
 - read0 读取第一个操作数,寄存器写入使能置一
 - read1 读取第二个操作数,记录当前操作数内存位置,内存写入使能置一
 - read2 读取操作符,操作符地址寄存器增一,寄存器写入使能置零
 - write 将运算结果写回内存,内存写入使能置零
 - stop 结束有限状态机循环,内存和寄存器写入使能置零
 - 。 定义初始化计数器 num , 在 idle 状态下进行初始化:
 - num < 13 给 RAM 地址 0 12 赋值 10 22

- num == 13 给 RAM 地址 13 赋值 23, 内存写入地址跳转至 100
- 13 < num < 20 给 RAM 地址 100 106 赋值 0 6
- num == 20 给 RAM 地址 107 赋值 -1
- num == 21 将 RAM 写入地址跳转至 199, 等待之后写入计算结果
- 。 该状态机为基于现态的状态机,实现实验要求仅需四个周期,分别是三个读周期与一个 写周期
- 。 状态循环如下:
 - 初始状态置为 idle
 - 当 num < 22 时, 为循环: idle -> idle
 - 当 num >= 22 后, 进入正常循环: idle -> read0
 - 正常循环周期: read0 -> read1 -> read2 -> write -> read0
 - 在 [read2] 状态下,如果读取的操作符为负一,即 [alu_op == -1] , 跳转至 [stop] 状态
- 3. 创建顶层文件 top.v
 - 。 提供时钟信号 clk 和复位信号 rst_n 的输入
 - 。 调用 RAM 模块 ram ,控制模块 control.v ,寄存器文件 regfile.v 和 ALU 模块 alu.v
 - 。 将由 ALU 输出的结果传入控制模块,由控制模块进行控制何时写入
- 4. 进行仿真

实验结果

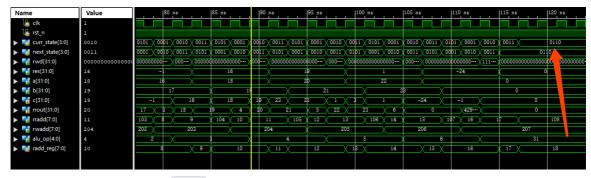
1. top.v 仿真:

| Name | Value | 0 ns | | 5 ns | | 10 | ns | | 15 | ns | | 20 |) ns | 25 n | s | | 30 ns | | . 1 | 35 ns | ١. | | 40 ns | | . 1 | 45 ns |
|---|---|------|----|------|------|----|------|-----|-------------|------|-----|-------|---|--------|--------|-------|--------|--------|-----|-------|-----|-----|--------|-----|-----|-------|
| ୀଲ clk | 1 | | | | | | | | | | | | | | | | | | | | П | | | | | |
| 1 rst_n rst_n | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| state[3:0] | 0000 | | | | | | | | | | | | 0000 | | | | | | | | | | | | | |
| mext_state[3:0] | 0000 | | | | | | | | | | | | 0000 | | | | | | | | | | | | | 0001 |
| rwd[31:0] | 000000000000000000000000000000000000000 | | | | | | | | | | 00 | 00000 | 000000000000000000000000000000000000000 | 000000 | 000000 | 00 | | | | | | | | | | |
| res[31:0] | 4 | 10 | 11 | 12 | 13 (| 14 | (15 | (16 | | 17 X | 18 | 19 | 20 21 | 22 | 23 | 3 (| X | 1 / | 2 | 3 | 4 | X | =X $=$ | 6 | -1 | 0 |
| ▶ 喘 a[31:0] | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| ▶ <table-of-contents> b[31:0]</table-of-contents> | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| ▶ 📆 c[31:0] | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| rrout[31:0] | 10 | (O)(| | | | | | | | | | | 10 | | | | | | | | | | | | | |
| rradd[7:0] | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| madd[7:0] | 104 | 0 | 1 | 2 | (3) | 4 | 5 | (6 | $\exists x$ | 7 X | 8 (| 9 | 10 / 11 | 12 | (12 | 3 / 1 | 00 (1 | 01 / 1 | 02 | 103 | 104 | (10 | 15 | 106 | 107 | 199 |
| alu_op[4:0] | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| sadd_reg[7:0] | 0 | | | | | | | | | | | | 0 | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |

- 。 res 为初始化向内存中写入数据
- 。 rwadd 为初始化向内存中写入的地址

| Name | Value | 50 ns | 55 ns | 60 ns | 65 ns | 70 ns | 75 ns | 80 ns | 85 ns | 90 ns | ٠ ا | 95 ns |
|-----------------|---|---|----------------|------------------|-------------------|----------------|---|------------------------|---|--------|---------------|-------|
| l₀ clk | 1 | | | | | | | | | | | |
| l rst_n | 1 | | | | | | | | | | | |
| state[3:0] | 0010 | 0 \ 0011 \ 0101 | 0001 0010 00 | 11 (0101 (0001 | 0010 (0011 (0 | 01 0001 0010 | 0011 0101 00 | 01 0010 0011 | 0101 (0001 | 0010 | 0011 0101 | 0001 |
| mext_state[3:0] | 0011 | 0 (0101 (0001) | 0010 (0011 (01 | 01 0001 0010 | 0011 (0101 (00 | 01 0010 0011 | 0101 0001 000 | 10 0011 0101 | 0001 (0010 | 0011 (| 0101 (0001) | 0010 |
| rwd[31:0] | 000000000000000000000000000000000000000 | 000000000000000000000000000000000000000 | 000\000\00 | 0000000000000000 | 000000000 | 00000000 \(000 | 000000000000000000000000000000000000000 | ···· \(0000··· \(00000 | 000000000000000000000000000000000000000 | 000 (0 | 000000000000 | 0000 |
| res[31:0] | 16 | | 0 | | \longrightarrow | 25 | -1 | Х | 16 | | 19 | |
| ▶ ■ a[31:0] | 18 | 0 \ 1 |) X | 12 | X | 14 | 16 | X | 18 | | 20 | |
| ▶ ■ b[31:0] | 19 | 0 X | 11 | / 1 | 3 | 15 | X | 17 | 1 | | X2 | 1 |
| ▶ ■ c[31:0] | 19 | 0 \ 10 | X | 0 | 25 27 | 29 / -1 | 1 / -1 | 16 | 18 | 19 (| 23 2 | 1 |
| rrout[31:0] | 20 | 10 \ 11 | 0 12 | 13 1 | 14 / 15 | 2 16 | 17 | 3 (18)(| 19 4 | 20 🗶 | 21 | 5 |
| rradd[7:0] | 11 | 1 100 | 2 / 3 | (101)(4) | 5 (1 | 02 (6) | (103) | 8 (9 | 104 10 | - 11 | 105 | 12 |
| rwadd[7:0] | 204 | 199 | 200 | X | 201 | 2 | 2 | 203 | \perp | 204 | | 205 |
| alu_op[4:0] | 4 | 10 | X | 0 | 1 | X | 2 | X | 3 | | 4 | |
| radd_reg[7:0] | 10 | 0 (1) | 2 | 4 | X 5 X | 6 | 7 (| 8 X 9 | 10 | | 11 / | 2 |
| | | | | | | | | | | | | |

- 。 a, b, c 为 ALU 运算操作数, alu_op 为运算符, res 为运算后写入结果
- 。 rradd 和 rwadd 为内存读写地址
- 。 curr_state 和 next_state 为当前和之后状态
- 。 一共使用四个周期实现



。 箭头所指即位进入 stop 状态

2. 内存中数据:

| 0 10 11 12 13 4 14 15 16 17 8 18 19 20 21 12 22 23 0 0 16 0 0 0 0 20 0 0 0 0 24 0 0 0 0 28 0 0 0 0 32 0 0 0 0 36 0 0 0 0 40 0 0 0 0 44 0 0 0 0 44 0 0 0 0 52 0 0 0 0 64 0 0 0 0 64 0 0 0 0 72 0 0 0 0 88 0 0 <td< th=""><th></th><th>0</th><th>1</th><th>2</th><th>3</th></td<> | | 0 | 1 | 2 | 3 |
|---|-----|----|--------|----|-----|
| 4 14 15 16 17 8 18 19 20 21 12 22 23 0 0 16 0 0 0 0 20 0 0 0 0 24 0 0 0 0 28 0 0 0 0 32 0 0 0 0 36 0 0 0 0 40 0 0 0 0 40 0 0 0 0 44 0 0 0 0 48 0 0 0 0 52 0 0 0 0 60 0 0 0 0 60 0 0 0 0 62 0 0 0 0 63 0 0 0 0 64 0 0 0 0 80 | 0 | | | | |
| 12 22 23 0 0 20 0 0 0 0 24 0 0 0 0 28 0 0 0 0 32 0 0 0 0 36 0 0 0 0 40 0 0 0 0 44 0 0 0 0 52 0 0 0 0 52 0 0 0 0 60 0 0 0 0 60 0 0 0 0 64 0 0 0 0 68 0 0 0 0 72 0 0 0 0 80 0 0 0 0 84 0 0 0 0 88 0 0 0 0 96 0 0 0 0 104 <td< th=""><th>4</th><th></th><th></th><th></th><th></th></td<> | 4 | | | | |
| 16 0 0 0 0 20 0 0 0 0 24 0 0 0 0 28 0 0 0 0 32 0 0 0 0 36 0 0 0 0 40 0 0 0 0 44 0 0 0 0 48 0 0 0 0 52 0 0 0 0 56 0 0 0 0 60 0 0 0 0 64 0 0 0 0 68 0 0 0 0 72 0 0 0 0 80 0 0 0 0 84 0 0 0 0 92 0 0 0 0 96 0 0 0 0 100 0 | 8 | 18 | 19 | 20 | 21 |
| 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 12 | 22 | 23 | 0 | 0 |
| 24 0 0 0 0 0 0 28 0 | 16 | 0 | 0 | 0 | 0 |
| 28 | 20 | 0 | 0 | 0 | 0 |
| 32 0 | 24 | 0 | 0 | 0 | 0 |
| 36 | 28 | 0 | 0 | 0 | 0 |
| 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 32 | 0 | 0 | 0 | 0 |
| 44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 36 | 0 | 0 | 0 | 0 |
| 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 40 | 0 | 0 | 0 | 0 |
| 52 0 | | 0 | 0 | 0 | 0 |
| 56 0 | 48 | 0 | 0 | 0 | 0 |
| 60 0 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0 0 0 | 52 | 0 | 0 | 0 | 0 |
| 64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 56 | 0 | 0 | 0 | 0 |
| 68 0 0 0 0 0 0 0 72 0 0 0 0 0 75 0 0 0 0 0 0 0 0 0 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 72 0 0 0 0 76 0 0 0 0 80 0 0 0 0 84 0 0 0 0 88 0 0 0 0 92 0 0 0 0 96 0 0 0 0 100 0 1 2 3 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 124 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 76 0 0 0 0 80 0 0 0 0 84 0 0 0 0 88 0 0 0 0 92 0 0 0 0 96 0 0 0 0 100 0 1 2 3 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 80 1 0 | | 0 | 0 | 0 | 0 |
| 84 0 0 0 0 88 0 0 0 0 92 0 0 0 0 96 0 0 0 0 100 0 1 2 3 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 88 0 0 0 0 92 0 0 0 0 96 0 0 0 0 100 0 1 2 3 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 92 0 0 0 0 0 0 0 96 0 0 0 0 100 0 100 0 1 2 3 3 104 4 5 6 -1 108 0 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 124 0 0 0 0 0 0 128 0 0 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 96 0 0 0 0 0 0 100 0 100 0 1 2 3 3 104 4 5 6 -1 108 0 0 0 0 0 0 112 0 0 0 0 0 116 0 0 0 0 0 0 120 0 0 0 0 124 0 0 0 0 0 0 128 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 100 0 1 2 3 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | 0 | 0 | 0 | 0 |
| 104 4 5 6 -1 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | | 0 | 0 | 0 |
| 108 0 0 0 0 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | | 1 | 2 | 3 |
| 112 0 0 0 0 116 0 0 0 0 120 0 0 0 0 124 0 0 0 0 128 0 0 0 0 | | | | | -1 |
| 116 0 0 0 0 0 120 0 120 0 0 124 0 0 0 0 128 0 0 0 0 | | | 0 | 0 | 0 |
| 120 0 0 0 0 0 124 0 0 0 128 0 0 0 0 | | | | | |
| 124 0 0 0 0 0 128 0 0 0 | | | 0 | 0 | 0 |
| 128 0 0 0 | | | 0 | 0 | 0 |
| | | | 0 | 0 | 0 |
| memory X AN | 128 | 0 | 0 | 0 | 0 |
| <u> </u> | | | memory | | X m |

。 初始化后的数据

| 200 | 0 | 25 | -1 | 16 |
|-----|----|-------------|----------|-----|
| 204 | 19 | 1 | -24 | 0 |
| 208 | 0 | 0 | 0 | 0 |
| 212 | 0 | 0 | 0 | 0 |
| 216 | 0 | 0 | 0 | 0 |
| 220 | 0 | 0 | 0 | 0 |
| 224 | 0 | 0 | 0 | 0 |
| 228 | 0 | 0 | 0 | 0 |
| 232 | 0 | 0 | 0 | 0 |
| 236 | 0 | 0 | 0 | 0 |
| 240 | 0 | 0 | 0 | 0 |
| 244 | 0 | 0 | 0 | 0 |
| 248 | 0 | 0 | 0 | 0 |
| 252 | 0 | 0 | 0 | 0 |
| ANA | | Default.wcf | * | × E |

。 运算结果

实验总结

首先实现了四个周期完成,后发现计算过程无需占用一个周期,故将运算周期去除,达到四个周期实现。

附录

源代码

top.v:

```
1
   `timescale 1ns / 1ps
 2
 3
   module top(
 4
        input clk,
 5
        input rst_n
 6
   );
 7
   wire [31:0] rrout, a, b, c, res, rwd;
   wire [7:0] rradd, rwadd;
10
   wire [5:0] wAddr;
   wire [4:0] alu_op;
11
12
   wire wEna;
13
   control con(clk, rst_n, c, rrout, alu_op, rradd, rwadd, wAddr, a, b,
14
    rwd, res, rwe, wEna);
15
    ram rr(
16
        .clka(clk),
17
        addra(rwadd),
```

```
18
        .dina(res),
19
        addrb(rradd),
20
        .clkb(clk),
21
        wea(rwe),
22
        .doutb(rrout)
23
   );
   regfile r(clk, rst_n, wAddr, rwd, wEna);
24
25
    alu alu1(a, b, alu_op, c);
26
27
   endmodule
```

control.v:

```
`timescale 1ns / 1ps
1
 2
 3
   module control(
4
        input clk,
5
        input rst_n,
6
        input [31:0] c,
 7
        input [31:0] rrout,
8
        output reg signed [4:0] alu_op,
9
        output reg [7:0] rradd,
10
        output reg [7:0] rwadd,
11
        output reg [5:0] wAddr,
12
        output reg [31:0] a,
13
        output reg [31:0] b,
        output reg [31:0] rwd,
14
15
        output reg [31:0] res,
16
        output reg rwe,
17
        output reg wEna
18
   );
19
20
   parameter idle = 3'h0;
21
    parameter read0 = 3'h1;
22
   parameter read1 = 3'h2;
23
    parameter read2 = 3'h3;
24
    parameter write = 3'h5;
25
    parameter stop = 3'h6;
26
27
    reg [3:0] curr_state, next_state;
28
    reg [4:0] num;
29
    reg [7:0] radd_reg, op_reg;
30
31
32
33
    always@(posedge clk or negedge rst_n) begin
34
        if (~rst_n)
35
            curr_state <= idle;</pre>
36
        else
37
            curr_state <= next_state;</pre>
38
   end
39
40
   always@(*) begin
41
        case(curr_state)
```

```
42
             idle:
43
                 if (num < 22) next_state = idle;</pre>
44
                 else next_state = read0;
45
             read0:
46
                 next_state = read1;
47
             read1:
48
                next_state = read2;
49
            read2:
50
                if (alu_op == -1) next_state = stop;
51
                 else next_state = write;
52
            write:
53
                next_state = read0;
54
            stop:
55
                next_state = stop;
56
             default:
57
                next_state = idle;
58
        endcase
59
    end
60
61
    always@(posedge clk or negedge rst_n) begin
62
        if (∼rst_n) begin
             rradd <= 0;</pre>
63
             rwadd <= 0;
64
65
            op_reg <= 8'd100;
            wAddr <= 0;
66
67
            rwe <= 1;
68
            wEna <= 0;
69
            num <= 0;
70
            res <= 10;
71
            rwd <= 0;
72
            a <= 0;
73
            b \ll 0;
74
            alu_op <= 0;
75
             radd_reg <= 0;
76
        end
77
        else if (curr_state == idle) begin
78
             if (num < 13) begin
79
                 rwadd <= rwadd + 1;
80
                 num <= num + 1;
81
                 res <= res + 1;
82
            end
             else if (num == 13) begin
83
                 rwadd <= 100;
84
85
                 num <= num + 1;
86
                 res <= 0;
87
            end
88
             else if (num < 20) begin
89
                 rwadd <= rwadd + 1;
90
                 num \ll num + 1;
91
                 res <= res + 1;
92
            end
93
             else if (num == 20) begin
                 rwadd <= rwadd + 1;
94
95
                 res <= -1;
96
                 num \ll num + 1;
```

```
97
              end
 98
              else if (num == 21) begin
 99
                  rwadd <= 199;
100
                  res <= 0;
                  num <= num + 1;
101
102
              end
103
          end
         else if (curr_state == read0) begin
104
105
              alu_op <= rrout;</pre>
              rradd <= rradd + 1;</pre>
106
              wAddr <= 0;
107
108
              wEna <= 1;
109
              rwd <= rrout;
110
          end
111
          else if (curr_state == read1) begin
112
              res <= c;
113
              radd_reg <= rradd;</pre>
114
              wAddr <= wAddr + 1;
115
              a <= rrout;
116
              rwd <= rrout;
117
              rwe <= 1;
118
          end
119
          else if (curr_state == read2) begin
120
              radd_reg <= radd_reg + 1;</pre>
121
              rradd <= op_reg;</pre>
122
              b <= rrout;
123
              op_reg <= op_reg + 1;
124
              wAddr <= wAddr + 1;
125
              wEna <= 0;
126
127
         end
128
         else if (curr_state == write) begin
129
              rwe <= 0;
              rwadd <= rwadd + 1;
130
131
              rradd <= radd_reg;</pre>
132
          end
133
          else if (curr_state == stop) begin
134
              rwe <= 0;
135
              wEna <= 0;
136
          end
137
     end
138
139
     endmodule
```

regfile.v :

```
1
   `timescale 1ns / 1ps
2
3
   module regfile(
4
       input
                       clk,
5
       input
                       rst_n,
6
       input
             [5:0]
                       wAddr,
7
       input
             [31:0]
                       wDin,
8
       input
                       wEna
```

```
9
   );
10
11
    reg [31:0] data [0:63];
12
    integer i;
13
14
    always@(posedge clk or negedge rst_n) begin
15
        if(∼rst n) begin
            data[0] <= 32'h0002;</pre>
16
            data[1] <= 32'h0002;
17
18
            for(i = 2; i < 64; i = i + 1)
19
                 data[i] <= 0;
20
        end
21
        if(wEna)
22
            data[wAddr] = wDin;
23
    end
24
25
   endmodule
```

alu.v:

```
1
   `timescale 1ns / 1ps
2
3
   module alu(
4
       input signed [31:0] alu_a,
 5
        input signed [31:0] alu_b,
 6
       input [4:0] alu_op,
 7
       output reg signed [31:0] alu_out
8
   );
9
   parameter A_NOP = 5'h00; //空运算
10
11
   parameter A_ADD = 5'h01; //符号加
   parameter A_SUB = 5'h02; //符号减
12
13
   parameter A_AND = 5'h03; //与
   parameter A_0R = 5'h04; //或
14
15
   parameter A_XOR = 5'h05; //异或
   parameter A_NOR = 5'h06; //或非
16
17
18
   always@(*) begin
19
     case (alu_op)
20
       A_NOP: alu_out = 32'h0;
21
       A_ADD: alu_out = alu_a + alu_b;
22
       A_SUB: alu_out = alu_a - alu_b;
23
       A_AND: alu_out = alu_a & alu_b;
24
       A OR: alu out = alu a | alu b;
       A_XOR: alu_out = alu_a ^ alu_b;
25
26
       A_NOR: alu_out = \sim (alu_a \mid alu_b);
27
       default: alu_out = alu_a;
28
     endcase
29
   end
30
31
   endmodule
```

```
1
    `timescale 1ns / 1ps
 2
 3
   module test;
 4
 5
        // Inputs
 6
        reg clk;
 7
        reg rst_n;
8
9
        // Instantiate the Unit Under Test (UUT)
10
        top uut (
11
            .clk(clk),
12
            .rst_n(rst_n)
        );
13
14
        always #1 clk = \simclk;
15
        initial begin
16
            // Initialize Inputs
17
            clk = 0;
18
            rst_n = 0;
19
            // Wait 100 ns for global reset to finish
20
21
            #2;
22
            rst_n = 1;
23
            repeat(500) @(posedge clk);
24
            // Add stimulus here
25
26
        end
27
28
   endmodule
```