实验8+. 有限状态机 入门

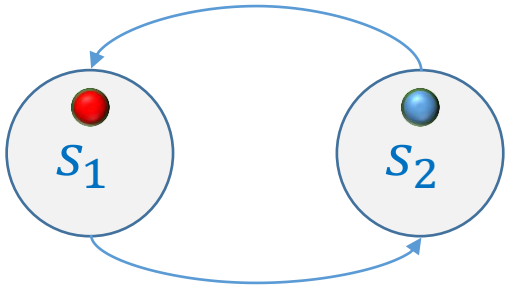
xgsun@fudan.edu.cn

孙晓光

2023-11-5

有限状态机 Finite State Machine

- 电路设计**经典方法**，尤其适于设计控制模块，易于FPGA实现。
- 用于：对**有限内部状态相互转换的系统**进行建模。
- 常为同步时序，在时钟信号的触发下完成各状态之间的转换，并产生相应的输出。
- 由**组合逻辑**(状态译码、产生输出信号) + **时序逻辑**(存储状态)构成。



FSM设计要点

- 状态机有3部分：
  - ① **当前状态** PS
  - ② **下一状态** NS
  - ③ **输出逻辑** OL
- Verilog有4种描述方式：
  - 三段式：①、②、③各用一个 always / assign 描述；
  - **两段式：** ① + ② 、③ *或* ②、① + ③ 各用一个always；
  - 一段式：① + ② + ③ 只用一个always。
- **多余状态**要明确定义，或者用case语句中的default。
- **初始状态**：电路复位后所处的状态。实用的状态机都应有**复位信号**。

rotatingLED

旋转的警灯

```
1  // 单色警灯：红灯旋转亮
2  module rotatingLED( input logic clk,
3                      input logic reset,
4                     output logic RedLED1,
5                     output logic RedLED2 );
6      parameter  S0 = 0,  S1 = 1,  S2 = 2;
7      logic [1:0]  state;
8
9      always_ff @(posedge clk)
10         if (!reset) state <= S0;
11         else case (state)
12                  S0 : state <= S1;
13                  S1 : state <= S2;
14                  S2 : state <= S1;
15              default: state <= S0;
16              endcase
17
18     always_comb
19         case (state)
20             S0 : begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
21             S1 : begin RedLED1 = 1'b1; RedLED2 = 1'b0; end // 红1亮
22             S2 : begin RedLED1 = 1'b0; RedLED2 = 1'b1; end // 红2亮
23         default: begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
24         endcase
25 endmodule
```

二段式描述

① 当前状态

② 下一状态

③ 输出逻辑

rotatingLED

旋转的警灯

```
// 单色警灯：红灯旋转亮
module rotatingLED( input logic clk,
                    input logic reset,
                   output logic RedLED1,
                   output logic RedLED2 );
    parameter  S0 = 0,  S1 = 1,  S2 = 2;
    logic [1:0]  state;

    always_ff @(posedge clk)
        if (!reset) state <= S0;
        else case (state)
                S0 : state <= S1;
                S1 : state <= S2;
                S2 : state <= S1;
            default: state <= S0;
            endcase

    always_comb
        case (state)
            S0 : begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
            S1 : begin RedLED1 = 1'b1; RedLED2 = 1'b0; end // 红1亮
            S2 : begin RedLED1 = 1'b0; RedLED2 = 1'b1; end // 红2亮
        default: begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
        endcase
endmodule
```

① 当前状态

② 下一状态

③ 输出逻辑

```
module Clk6Hz(input  logic clk100MHz,
              output logic clk6Hz  );

    logic [23:0] count;

    always @(posedge clk100MHz)
        count <= count + 1;

    assign clk6Hz = count[23];
endmodule
```

Top文件

```
module rotatingLED_Top(
    input  logic CLK100MHZ,
    input  logic BTNC,
    output logic LED17_R,
    output logic LED16_R  );

    logic clk6Hz;

    Clk6Hz C6(CLK100MHZ, clk6Hz);

    rotatingLED A1(.clk(clk6Hz),
                   .reset(!BTNC),
                   .RedLED1(LED17_R),
                   .RedLED2(LED16_R) );
endmodule
```

旋转的警灯

```
1  // 单色警灯：红灯旋转亮
2  module rotatingLED( input logic clk,
3                      input logic reset,
4                     output logic RedLED1,
5                     output logic RedLED2 );
6      parameter  S0 = 0,  S1 = 1,  S2 = 2;
7      logic [1:0]  state;
8
9      always_ff @(posedge clk)
10         if (!reset) state <= S0;
11         else case (state)
12                 S0 : state <= S1;
13                 S1 : state <= S2;
14                 S2 : state <= S1;
15             default: state <= S0;
16             endcase
17
18     always_comb
19         case (state)
20             S0 : begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
21             S1 : begin RedLED1 = 1'b1; RedLED2 = 1'b0; end // 红1亮
22             S2 : begin RedLED1 = 1'b0; RedLED2 = 1'b1; end // 红2亮
23         default: begin RedLED1 = 1'b0; RedLED2 = 1'b0; end // 全灭
24         endcase
25 endmodule
```

① 当前状态

② 下一状态

③ 输出逻辑