## 实验一报告 基于Verilog HDL的数字电路设计（一）

### 实验目的

使用 Verilog HDL 语言设计以下功能模块，并在 FPGA 或仿真环境中验证其逻辑正确性：

1. 8位带进位输入/输出的全加器
2. 3-8线译码器
3. 8选1多路选择器
4. 4位可配置移位寄存器（含去抖模块）

### 实验一：8位带进位输入输出的全加器

#### 实验原理

全加器实现两个 8 位输入的加法，并处理进位输入（cin）与输出（sum[8]）。通过控制信号 sel 选择是否累加先前存储的值，实现简单的带状态计算功能。

#### 程序代码

module full\_adder\_8bit(

input clk, // 系统时钟

input rst, // 异步复位，高电平有效

input [7:0] a, // 8位输入

input sel, // 控制信号：

// sel = 0：更新存储值并输出当前输入

// sel = 1：使用存储值与当前输入相加

input cin, // 进位输入

output reg [8:0] sum // 9位输出（含进位）

);

reg [7:0] stored\_a;

always @(posedge clk or posedge rst) begin

if (rst) begin

stored\_a <= 8'b0;

sum <= 9'b0;

end else if (sel == 1'b0) begin

stored\_a <= a;

sum <= {1'b0, a} + cin;

end else begin

sum <= a + stored\_a + cin;

end

end

endmodule

#### 实验结果及分析

如图，加数已被设置为4，被加数为8，进位1，结果为8+4+1=13。

#### 问题与解决

* **问题**：初始未考虑复位条件。
* **解决**：添加异步复位逻辑。

### 实验二：3-8线译码器

#### 实验原理

译码器将3位二进制输入转换为8位输出，每次仅一个输出位为0，其余为1。设定特定使能条件。

#### 程序代码

module Decoder(

input [2:0] data\_i, // 3位输入数据

input [2:0] en\_i, // 3位使能信号

output reg [7:0] data\_o // 8位输出

);

always @(\*) begin

if(en\_i[0] || en\_i[1] || !en\_i[2])

data\_o = 8'b1111\_1111; // 使能无效，全部高电平

else

case (data\_i)

3'b000: data\_o = 8'b1111\_1110;

3'b001: data\_o = 8'b1111\_1101;

3'b010: data\_o = 8'b1111\_1011;

3'b011: data\_o = 8'b1111\_0111;

3'b100: data\_o = 8'b1110\_1111;

3'b101: data\_o = 8'b1101\_1111;

3'b110: data\_o = 8'b1011\_1111;

3'b111: data\_o = 8'b0111\_1111;

endcase

end

endmodule

#### 实验结果及分析

如图，选择位输入101，5号位灯熄灭。

### 实验三：8选1多路选择器

#### 实验原理

根据 3 位选择信号 sel，从 8 位输入 in 中选出一个输出。

#### 程序代码

module MUX(

input [2:0] sel, // 选择信号

input [7:0] in, // 8路输入

output reg out // 单比特输出

);

always @(sel) begin

case(sel)

3'b000: out <= in[0];

3'b001: out <= in[1];

3'b010: out <= in[2];

3'b011: out <= in[3];

3'b100: out <= in[4];

3'b101: out <= in[5];

3'b110: out <= in[6];

3'b111: out <= in[7];

endcase

end

endmodule

#### 实验结果及分析

如图，状态位为101（图误），输出了5位上的数据。

### 实验四：4位移位寄存器（含可配置功能与按键去抖）

#### 实验原理

该模块支持：

* 可配置的移位方向（左/右）
* 移位模式（循环/逻辑补0）
* 支持按钮控制加载/移位动作
* 使用 debounce 去抖模块稳定按键信号

#### 程序代码（分模块）

##### 1. debounce（去抖模块）

module debounce(

input clk,

input rst,

input noisy\_in,

output reg clean\_out

);

reg [127:0] shift\_reg;

always @(posedge clk or posedge rst) begin

if (rst) begin

shift\_reg <= 128'hFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF;

clean\_out <= 0;

end else begin

shift\_reg <= {shift\_reg[126:0], noisy\_in};

if (shift\_reg == 128'hFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF)

clean\_out <= 1;

else if (shift\_reg == 128'h0000000000000000000000000000000)

clean\_out <= 0;

end

end

endmodule

##### 2. 移位寄存器模块

module configurable\_shift\_reg\_4bit(

input clk,

input rst,

input start,

input load,

input dir,

input mode,

input [3:0] d,

output reg [3:0] q

);

reg start\_d, load\_d;

wire start\_pulse = ~start\_d & start;

wire load\_pulse = ~load\_d & load;

always @(posedge clk or posedge rst) begin

if (rst) begin

start\_d <= 0;

load\_d <= 0;

end else begin

start\_d <= start;

load\_d <= load;

end

end

always @(posedge clk or posedge rst) begin

if (rst)

q <= 4'b0000;

else if (load\_pulse)

q <= d;

else if (start\_pulse) begin

if (dir) // 左移

q <= mode ? {q[2:0], q[3]} : {q[2:0], 1'b0};

else // 右移

q <= mode ? {q[0], q[3:1]} : {1'b0, q[3:1]};

end

end

endmodule

##### 3. 顶层模块（整合输入输出）

module top(

input clk\_100mhz,

input rst,

input btn\_start\_raw,

input btn\_load\_raw,

input dir,

input mode,

input [3:0] d,

output [3:0] q

);

wire btn\_start\_clean;

wire btn\_load\_clean;

debounce u\_debounce\_start(

.clk(clk\_100mhz),

.rst(rst),

.noisy\_in(btn\_start\_raw),

.clean\_out(btn\_start\_clean)

);

debounce u\_debounce\_load(

.clk(clk\_100mhz),

.rst(rst),

.noisy\_in(btn\_load\_raw),

.clean\_out(btn\_load\_clean)

);

configurable\_shift\_reg\_4bit u\_shift\_reg(

.clk(clk\_100mhz),

.rst(rst),

.start(btn\_start\_clean),

.load(btn\_load\_clean),

.dir(~dir), // 与硬件方向开关对应

.mode(mode),

.d(d),

.q(q)

);

endmodule

#### 实验结果与测试图

如图所示。

#### 问题与解决办法

| **问题描述** | **解决方法** |
| --- | --- |
| 按钮误触反复触发 | 添加 debounce 去抖模块 |
| 多模块连接逻辑不清 | 分层设计 + 顶层整合 |
| 忘记复位条件 | 每个 always 块加 rst 分支 |