电子技术实验-FPGA第二次作业

UNIkeEN

Lab2-1 静态显示 Pro

源文件代码

学号尾号偶数，偶数时显示小数点

```verilog
module seg_static_pro
#(
    parameter   CNT_MAX = 25'd24_999_999
)
(
    input   wire            sys_clk     ,
    input   wire            sys_rst_n   ,

    output   reg    [3:0]   sel         ,
    output   reg    [7:0]   seg
);

reg     [24:0]  cnt;
reg             cnt_flag;
reg     [3:0]   data;

//cnt
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt <= 25'd0;
    else if(cnt == CNT_MAX)
        cnt <= 25'd0;
    else
        cnt <= cnt + 25'd1;

//cnt_flag
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt_flag <= 1'b0;
    else if(cnt == CNT_MAX - 25'd1)
        cnt_flag <= 1'b1;
    else
        cnt_flag <= 1'b0;

//data
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        data <= 4'd0;
    else if((cnt_flag == 1'b1) && (data == 4'b1111))
        data <= 4'd0;
    else if(cnt_flag == 1'b1)
        data <= data + 4'd1;
    else
        data <= data;
```

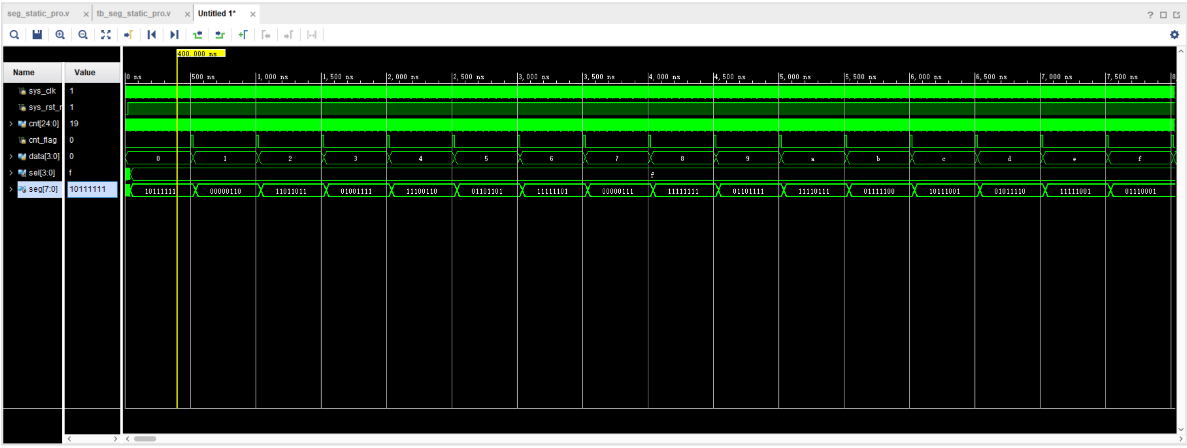
```
//sel
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        sel <= 4'b0000;
    else
        sel <= 4'b1111;

//seg
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        seg <= 8'b0000_0000;
    else
        case(data)
            4'd0:       seg <= 8'b1011_1111;
            4'd1:       seg <= 8'b0000_0110;
            4'd2:       seg <= 8'b1101_1011;
            4'd3:       seg <= 8'b0100_1111;
            4'd4:       seg <= 8'b1110_0110;
            4'd5:       seg <= 8'b0110_1101;
            4'd6:       seg <= 8'b1111_1101;
            4'd7:       seg <= 8'b0000_0111;
            4'd8:       seg <= 8'b1111_1111;
            4'd9:       seg <= 8'b0110_1111;
            4'd10:      seg <= 8'b1111_0111;
            4'd11:      seg <= 8'b0111_1100;
            4'd12:      seg <= 8'b1011_1001;
            4'd13:      seg <= 8'b0101_1110;
            4'd14:      seg <= 8'b1111_1001;
            4'd15:      seg <= 8'b0111_0001;
            default:    seg <= 8'b1000_0000;
        endcase
endmodule
```

仿真文件代码

```
`timescale 1ns/1ns

module tb_seg_static_pro();

reg     sys_clk;
reg     sys_rst_n;

wire    [3:0]   sel;
wire    [7:0]   seg;

initial
    begin
        sys_clk = 1'b1;
        sys_rst_n <= 1'b0;
        #20
        sys_rst_n <= 1'b1;
    end

always #10 sys_clk = ~sys_clk;
```

```
seg_static_pro
#(
    .CNT_MAX(25'd24)
)
seg_static_pro_inst
(
    .sys_clk      (sys_clk    ),
    .sys_rst_n    (sys_rst_n  ),

    .sel          (sel        ),
    .seg          (seg        )
);

endmodule
```

仿真运行关键波形截图



Name	Value
sys_clk	1
sys_rst_n	1
cnt[24:0]	19
cnt_flag	0
data[3:0]	0
sel[3:0]	f
seg[7:0]	10111111

Lab2-2 动态显示

源文件代码

`seg_dynamic.v`

```
module seg_dynamic
(
    input    wire              sys_clk      ,
    input    wire              sys_rst_n    ,

    output   wire     [7:0]    seg          ,
    output   wire     [7:0]    sel
);

wire     [15:0]   data;

data_gen
#(
    .CNT_MAX   (23'd4_999_999),
    .DATA_MAX  (16'hFFFF)
)
data_gen_inst
(
    .sys_clk    (sys_clk  ),
    .sys_rst_n  (sys_rst_n),

    .data       (data     )
);

data_show data_show_inst
(
    .sys_clk     (sys_clk    ),
    .sys_rst_n   (sys_rst_n  ),
    .data        (data       ),

    .seg         (seg        ),
    .sel         (sel        )
);

endmodule
```

`data_gen.v`

```
module data_gen
#(
    parameter   CNT_MAX  = 23'd4_999_999,
    parameter   DATA_MAX = 16'hFFFF
)
(
    input   wire            sys_clk      ,
    input   wire            sys_rst_n    ,

    output   reg    [15:0]  data
);

reg     [22:0]  cnt_50ms;
reg             cnt_flag;

//cnt_50ms
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt_50ms <= 23'd0;
    else if(cnt_50ms == CNT_MAX)
        cnt_50ms <= 23'd0;
    else
        cnt_50ms <= cnt_50ms + 23'd1;

//cnt_flag
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt_flag <= 1'b0;
    else if(cnt_50ms == CNT_MAX - 23'd1)
        cnt_flag <= 1'b1;
    else
        cnt_flag <= 1'b0;
```

```
//data
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        data <= 16'h0;
    else if((data == DATA_MAX)&&(cnt_flag == 1'b1))
        data <= 16'h0;
    else if(cnt_flag == 1'b1)
        data <= data + 16'd1;
    else
        data <= data;

endmodule
```

`data_show.v`

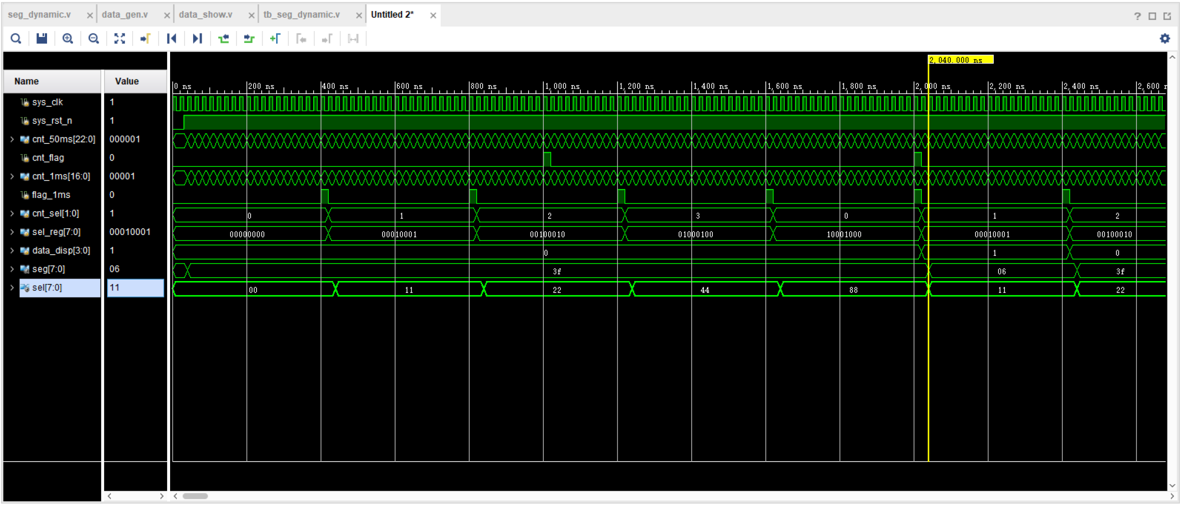
```
module data_show
(
    input   wire            sys_clk     ,
    input   wire            sys_rst_n   ,
    input   wire    [15:0]  data        ,

    output   reg    [7:0]   seg         ,
    output   reg    [7:0]   sel
);

parameter   CNT_MAX = 17'd99_999;

reg     [16:0]  cnt_1ms;
reg             flag_1ms;
reg     [1:0]   cnt_sel;
reg     [7:0]   sel_reg;
reg     [3:0]   data_disp;

//cnt_1ms
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        cnt_1ms <= 17'd0;
    else if(cnt_1ms == CNT_MAX)
        cnt_1ms <= 17'd0;
    else
        cnt_1ms <= cnt_1ms + 17'd1;

//flag_1ms
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        flag_1ms <= 1'b0;
    else if(cnt_1ms == CNT_MAX - 17'd1)
        flag_1ms <= 1'b1;
    else
        flag_1ms <= 1'b0;

//cnt_sel
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
```

```
        cnt_sel <= 2'd0;
    else if((cnt_sel == 2'd3) && (flag_1ms == 1'b1))
        cnt_sel <= 2'd0;
    else if(flag_1ms == 1'b1)
        cnt_sel <= cnt_sel + 1'd1;
    else
        cnt_sel <= cnt_sel;

//sel_reg
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        sel_reg <= 8'b0000_0000;
    else if((cnt_sel == 2'd0)&&(flag_1ms == 1'b1))
        sel_reg <= 8'b0001_0001;
    else if(flag_1ms == 1'b1)
        sel_reg <= sel_reg << 1;
    else
        sel_reg <= sel_reg;

//data_disp
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        data_disp <= 4'd0;
    else if(flag_1ms == 1'b1)
        case(cnt_sel)
            2'd0:       data_disp <= data[3:0];
            2'd1:       data_disp <= data[7:4];
            2'd2:       data_disp <= data[11:8];
            2'd3:       data_disp <= data[15:12];
            default:    data_disp <= 4'd0;
        endcase
    else
        data_disp <= data_disp;


//seg
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        seg <= 8'b0000_0000;
    else
        case(data_disp)
            4'd0:       seg <= 8'b0011_1111;
            4'd1:       seg <= 8'b0000_0110;
            4'd2:       seg <= 8'b0101_1011;
            4'd3:       seg <= 8'b0100_1111;
            4'd4:       seg <= 8'b0110_0110;
            4'd5:       seg <= 8'b0110_1101;
            4'd6:       seg <= 8'b0111_1101;
            4'd7:       seg <= 8'b0000_0111;
            4'd8:       seg <= 8'b0111_1111;
            4'd9:       seg <= 8'b0110_1111;
            4'd10:      seg <= 8'b0111_0111;
            4'd11:      seg <= 8'b0111_1100;
            4'd12:      seg <= 8'b0011_1001;
            4'd13:      seg <= 8'b0101_1110;
            4'd14:      seg <= 8'b0111_1001;
            4'd15:      seg <= 8'b0111_0001;
            default:    seg <= 8'b0000_0000;
        endcase

//sel
always@(posedge sys_clk or negedge sys_rst_n)
    if(sys_rst_n == 1'b0)
        sel <= 8'b0000_0000;
    else
        sel <= sel_reg;

endmodule
```

仿真文件代码

```
`timescale 1ns / 1ps

module tb_seg_dynamic();

reg             sys_clk     ;
reg             sys_rst_n   ;

wire    [7:0]   seg         ;
wire    [7:0]   sel         ;

initial
    begin
        sys_clk = 1'b1;
        sys_rst_n <= 1'b0;
        #30
        sys_rst_n <= 1'b1;
    end

always #10 sys_clk = ~sys_clk;

defparam    seg_dynamic_inst.data_show_inst.CNT_MAX = 17'd19;
defparam    seg_dynamic_inst.data_gen_inst.CNT_MAX = 23'd49;

seg_dynamic seg_dynamic_inst
(
    .sys_clk     (sys_clk   ),
    .sys_rst_n   (sys_rst_n ),

    .seg         (seg       ),
    .sel         (sel       )
);

endmodule
```

仿真运行关键波形截图



Name	Value
sys_clk	1
sys_rst_n	1
cnt_50ms[22:0]	000001
cnt_flag	0
cnt_1ms[16:0]	00001
flag_1ms	0
cnt_sel[1:0]	1
sel_reg[7:0]	00010001
data_disp[3:0]	1
seg[7:0]	06
sel[7:0]	11