Prelab07　Electronic Clocks

104061212　馮立俞

Top module—min_sec

```
`define BCD_WIDTH 4
module min_sec(
        input clk,
        input rst_n,
        output     [`BCD_WIDTH-1:0] min10,
        output    [`BCD_WIDTH-1:0] min1,
        output    [`BCD_WIDTH-1:0] sec10,
        output    [`BCD_WIDTH-1:0] sec1
        );

        wire MinCarry,SecCarry;         //60 sec -> 1 min

        sixtyup min(min10,min1,MinCarry,SecCarry,rst_n);
        sixtyup sec(sec10,sec1,SecCarry,clk,rst_n);

endmodule
```

Second level module—sixtyup

```
`define BCD_ZERO 4'd0
`define BCD_FIVE 4'd5
`define BCD_NINE 4'd9
`define BCD_WIDTH 4
`define ENABLED 1'b1
`define DISABLED 1'b0


module sixtyup(
        output [`BCD_WIDTH-1:0] sec1,
        output [`BCD_WIDTH-1:0] sec0,
          output cout1,
        input clk,
        input rst_n
        );

          wire cout0;
          reg ld_def;

          always@(sec1 or sec0)
               if(sec1 == `BCD_FIVE && sec0 == `BCD_NINE)       ld_def = `ENABLED;
               else  ld_def = `DISABLED;

          ucounter U1(sec1,cout1,clk,rst_n,cout0,ld_def,`BCD_ZERO);      //cout0 as
count enable
          ucounter U0(sec0,cout0,clk,rst_n,`ENABLED, ld_def,`BCD_ZERO);

endmodule
```

Building block—ucounter

```
`define BCD_WIDTH 4
`define ENABLED 1'b1
`define DISABLED 1'b0
`define BCD_NINE 4'd9
`define BCD_ZERO 4'd0
`define INCREMENT 1'b1

module ucounter(
        output reg [`BCD_WIDTH-1:0] value,
        output reg carry,
        input clk,
        input rst_n,
        input inc,
        input ld_def,
        input [`BCD_WIDTH-1:0] def_value
        );

        reg [`BCD_WIDTH-1:0]value_tmp;

        always@(value or inc or ld_def or def_value)
                if(inc == `DISABLED)            value_tmp <= value;//hold value
                else if(ld_def == `ENABLED)value_tmp <= def_value;      //load default
                else if(value==`BCD_NINE)       value_tmp <= `BCD_ZERO;
                else                                    value_tmp <= value + `INCREMENT;

        always@(inc or value or ld_def)
                if(inc == `ENABLED && value == `BCD_NINE || ld_def == `ENABLED)//X9 or 59
                        carry = `ENABLED;
                else  carry = `DISABLED;

        always@(posedge clk or negedge rst_n)   //combinational
                if(~rst_n)  value <= def_value;
                else                value <= value_tmp;

endmodule
```

Testbench

```
module min_secTest;

	// Inputs
	reg clk;
	reg rst_n;

	// Outputs
	wire [3:0] min10;
	wire [3:0] min1;
	wire [3:0] sec10;
	wire [3:0] sec1;

	// Instantiate the Unit Under Test (UUT)
	min_sec uut (
		.clk(clk),
		.rst_n(rst_n),
		.min10(min10),
		.min1(min1),
		.sec10(sec10),
		.sec1(sec1)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		rst_n = 0;

		// Wait 100 ns for global reset to finish
		#100;

		// Add stimulus here
		rst_n = 1;

		repeat(8000)
			#1 clk = ~clk;	//toggle 8000 times
```

```
    end

endmodule
```

Simulation Result