Alinx AX7035t 开发板实验

——基于埃氏筛法的 6 位素数计算与显示

```
////////////////////////////////////////////////////////////////////////////////
module top
(
   input            clk,
   input            rstn,
   output  [3:0]    led,
   input   [3:0]    key,
   output  [5:0]    seg_sel,
   output  [7:0]    seg_dig
);
wire [3:0]           key_signal;
wire [3:0]           key_pulse;
wire                 rstn_signal;
wire                 tick;
wire                 one_second;
wire                 select;
wire                 reset;
wire [47:0]          seg;
wire [19:0]          cnt_20b;
wire [23:0]          cnt_24d;

//1秒计时器
Count_to_one_second timer(clk,one_second);

//按键除抖及脉冲
Killshake Killshake(clk,rstn,rstn_signal);
genvar j;
generate for(j = 0; j < 4; j = j + 1) begin
   Killshake Killshake (clk,key[j],key_signal[j]);
   Edgedetect Edgedetect (key_signal[j],clk,key_pulse[j]);
end
endgenerate

//Led控制及模式选择
ledcontrol ledcontrol(clk,rstn_signal,key_pulse,led);
modecontrol modecontrol(clk,rstn_signal,one_second,led,key_pulse,reset,select,tick);

//埃氏筛法
isprime solver(clk,reset,tick,select,cnt_20b);

//显示模块
binary_20b_to_bcd_6d transformer(cnt_20b,cnt_24d);
genvar i;
generate for(i=0; i<6; i=i+1) begin
   led7seg_decode d(cnt_24d[i*4 +: 4], 1'b1, seg[i*8 +: 8]);
```

```verilog
end
endgenerate
seg_driver #(6) driver(clk, rstn_signal, 6'b111111, seg, seg_sel, seg_dig);

endmodule
/////////////////////////////////////////////////////////////////////////////////////
module ledcontrol
(
   input           clk,
   input           rstn_signal,
   input   [3:0]   key_pulse,
   output  [3:0]   led
);

reg    [3:0]    led_r;
always @(posedge clk or negedge rstn_signal) begin
   if(~rstn_signal) begin
      led_r <= 4'b1110;
   end
   else if(~key_pulse[0]) begin
      led_r <= 4'b1110;
   end
   else if(~key_pulse[1]) begin
      led_r <= 4'b1101;
   end
   else if(~key_pulse[2]) begin
      led_r <= 4'b1011;
   end
   else if(~key_pulse[3]) begin
      led_r <= 4'b0111;
   end
end
assign led = led_r;
endmodule
/////////////////////////////////////////////////////////////////////////////////////
module modecontrol
(
   input           clk,
   input           rstn_signal,
   input           one_second,
   input  [3:0]    led,
   input  [3:0]    key_pulse,
   output          reset,
   output          select,
   output          tick
);

reg                 tick_r;
reg                 select_reg;
reg   [1:0]         reset_r;
```

```

always @(posedge clk) begin
   if(~led[0]) begin
      tick_r <= one_second;
      select_reg<=1;
   end
   else if(~led[1])begin
      tick_r <= one_second;
      select_reg<=0;
   end
   else if(~led[2])begin
      tick_r <= 1;
      select_reg<=1;
   end
   else if(~led[3])begin
      tick_r <= 1;
   select_reg<=0;
   end
   else begin
      tick_r<=0;
   end
end

always @(posedge clk) begin
   reset_r<={reset_r[0],(&key_pulse)&rstn_signal};
end

assign tick = tick_r;
assign select=select_reg;
assign reset=reset_r[1];

endmodule
////////////////////////////////////////////////////////////////////////////////////
module Edgedetect
(
   input   key,
   input   clk,
   output  pulse
);

reg     key_prev;
reg     pulse_reg;

always @(posedge clk) begin
   key_prev <= key;
   if (key_prev & ~key)
      pulse_reg <= 0;
   else
      pulse_reg <= 1;
   end
```

```verilog

assign pulse = pulse_reg;

endmodule
/////////////////////////////////////////////////////////////////////////////////
module Killshake
(
input   clk,
input   key,
output  signal
);

parameter   DEBOUNCE_TIME = 1000000;
reg [19:0]  count;
reg         key_state;
reg         signal_reg;

always @(posedge clk) begin
   if (key == key_state) begin
   if (count < DEBOUNCE_TIME)
      count <= count + 1;
   else
      signal_reg <= key_state;
   end
   else begin
      count <= 0;
      key_state <= key;
   end
end

assign signal = signal_reg;

endmodule
/////////////////////////////////////////////////////////////////////////////////
module seg_driver #(parameter NPorts=8)
(
input                       clk, rstn,
input [NPorts-1:0]          valid_i,
input [NPorts*8-1:0]        seg_i,
output reg [NPorts-1:0]     valid_o,
output [7:0]                seg_o
);

reg [14:0]          cnt;
reg [NPorts-1:0]    sel;
always @(posedge clk or negedge rstn)
   if(~rstn)
      cnt <= 0;
   else
      cnt <= cnt + 1;
```

```

always @(posedge clk or negedge rstn)
   if(~rstn)
      sel <= 0;
   else if(cnt == 0)
      sel <= (sel == NPorts - 1) ? 0 : sel + 1;

always @(sel, valid_i) begin
   valid_o = {NPorts{1'b1}};
   valid_o[sel] = ~valid_i[sel];
end

assign seg_o = ~seg_i[sel*8+:8];

endmodule
////////////////////////////////////////////////////////////////////////////////////
module led7seg_decode
(
   input [3:0] digit,
   input valid,
   output reg [7:0] seg
);
always @(digit)
   if(valid)
   case(digit)
      0: seg = 8'b00111111;//0
      1: seg = 8'b00000110;//1
      2: seg = 8'b01011011;//2
      3: seg = 8'b01001111;//3
      4: seg = 8'b01100110;//4
      5: seg = 8'b01101101;//5
      6: seg = 8'b01111101;//6
      7: seg = 8'b00000111;//7
      8: seg = 8'b01111111;//8
      9: seg = 8'b01101111;//9
      default: seg = 0;
   endcase
   else seg = 8'd0;
endmodule
////////////////////////////////////////////////////////////////////////////////////
module isprime #(parameter N=999999)
(
   input clk,rstn,tick,
   input select,
   output [19:0] cnt_20b
);
reg [19:0]      w_addr;
reg             w_data;
reg             wea;
reg [19:0]      r_addr;
```

```
wire              r_data;

reg [19:0]        i;
reg [19:0]        j;
reg               en;
reg               done;


reg [19:0]        cnt_temp_reg;
reg [19:0]        cnt_20b_reg;

reg [2:0]         timer;
reg               hold;

always @(posedge clk or negedge rstn) begin
   if(!rstn) begin
      cnt_temp_reg<=(select)?2:N;
      cnt_20b_reg<=(select)?2:N;
      wea<=0;
      i<=2;
      j<=0;
      en<=0;
      done<=0;
      timer<=0;
      hold<=1;
   end
   else if (i*i<=N) begin
      if(en==0)begin
         r_addr<=i;
         if(timer>2)begin
            timer<=0;
            hold<=0;
         end
         else begin
            timer<=timer+1;
            hold<=1;
         end
         if(!hold) begin
            if (r_data==0) begin
               en<=1;
               j<=i+i;
            end
            else begin
               i<=i+1;
            end
         end
      end
      else if(en==1) begin
         if(j<N)begin
            wea<=1;
```

```
                w_addr<=j;
                w_data<=1;
                j<=j+i;
            end
            else begin
                wea<=0;
                en<=0;
                i<=i+1;
            end
        end
    end
    else begin
        done<=1;
        if(done) begin
            if (((cnt_temp_reg<N)&(select))|((cnt_temp_reg>=2)&(~select))) begin
                r_addr<=cnt_temp_reg;
                if(hold) begin
                    if(timer>2)begin
                        timer<=0;
                        hold<=0;
                    end
                    else begin
                        timer<=timer+1;
                    end
                end
                else begin
                    if ((~r_data)&tick) begin
                        cnt_20b_reg<=cnt_temp_reg;
                        cnt_temp_reg<=(select)?cnt_temp_reg+1:cnt_temp_reg-1;
                        hold<=1;
                    end
                    else if ((r_data)) begin
                        cnt_temp_reg<=(select)?cnt_temp_reg+1:cnt_temp_reg-1;
                        hold<=1;
                    end
                    else if((~r_data)&(~tick)) begin
                        cnt_temp_reg<=cnt_temp_reg;
                    end
                end
            end
        end
    end
end
assign cnt_20b=cnt_20b_reg;
ram_ip ram_ip_inst_1
(
.clka      (clk          ),
.wea       (wea          ),
.addra     (w_addr       ),
.dina      (w_data       ),
```

```verilog
.clkb      (clk           ),
.addrb     (r_addr        ),
.doutb     (r_data        )
);
endmodule
////////////////////////////////////////////////////////////////////////////////////
module binary_20b_to_bcd_6d #(parameter N = 20,parameter M = 24)
(
   input  [N-1:0]  input_20b,
   output [M-1:0]  output_6d
);
reg [3:0]      digits [5:0];

integer i;
always @(input_20b) begin
   for (i = 0; i < 6; i = i+1) begin
      digits[i] = 4'd0;
   end
   for(i = N-1; i >= 0; i = i-1) begin  //加3移位法
      if (digits[0] >= 4'b0101) digits[0] = digits[0] + 4'b0011;
      if (digits[1] >= 4'b0101) digits[1] = digits[1] + 4'b0011;
      if (digits[2] >= 4'b0101) digits[2] = digits[2] + 4'b0011;
      if (digits[3] >= 4'b0101) digits[3] = digits[3] + 4'b0011;
      if (digits[4] >= 4'b0101) digits[4] = digits[4] + 4'b0011;
      if (digits[5] >= 4'b0101) digits[5] = digits[5] + 4'b0011;

      digits[5][3:0] = {digits[5][2:0], digits[4][3]};
      digits[4][3:0] = {digits[4][2:0], digits[3][3]};
      digits[3][3:0] = {digits[3][2:0], digits[2][3]};
      digits[2][3:0] = {digits[2][2:0], digits[1][3]};
      digits[1][3:0] = {digits[1][2:0], digits[0][3]};
      digits[0][3:0] = {digits[0][2:0], input_20b[i]};
   end
end

assign output_6d ={digits[5],digits[4],digits[3],digits[2],digits[1],digits[0]};

endmodule
////////////////////////////////////////////////////////////////////////////////////
module Count_to_one_second #(parameter Count_To = 50_000_000)
(
   input clk,
   output one_second
);

reg [31:0]  counter;
reg         one_second_r;

always @(posedge clk) begin
   if((counter < Count_To) && ~one_second_r) begin
```

```
            counter <= counter + 1;
        end
        else if((counter < Count_To) && one_second_r) begin
            one_second_r <= 0;
            counter <= counter + 1;
        end
        else begin
            counter <= 0;
            one_second_r <= 1;
        end
    end

    assign one_second = one_second_r;

    endmodule
```