module gps\_decoder(

input rx\_clk,

input rx\_data,

input reset,

output [7:0] latitude,

output [7:0] longitude,

output [7:0] altitude,

output [7:0] speed,

output [7:0] heading,

output [7:0] utc\_hour,

output [7:0] utc\_min,

output [7:0] utc\_sec,

output [7:0] sat\_count,

output [7:0] hdop

);

/\*

\*定义常量

\*/

localparam BYTE\_COUNT = 66; //NMEA消息的字节数

localparam DATA\_START = 8; //数据开始的字节

/\*

\*定义寄存器

\*/

reg [7:0] buffer [BYTE\_COUNT-1:0];

reg [7:0] buffer\_pointer = 0;

reg ready = 0;

reg process = 0;

/\*

\*定义部分GPS消息的字段

\*/

reg [2:0] message\_id;

reg [7:0] utc\_time[3];

reg [7:0] latitude\_b[3];

reg [7:0] longitude\_b[3];

reg [7:0] altitude\_b[3];

reg [7:0] speed\_b[3];

reg [7:0] heading\_b[3];

reg [7:0] sat\_count\_b;

reg [7:0] hdop\_b[2];

/\*

\*状态机状态

\*/

localparam IDLE = 0;

localparam RX = 1;

localparam PARSE = 2;

localparam ERROR = 3;

reg [1:0] state = IDLE;

/\*

\*等待启动

\*/

always @(posedge rx\_clk) begin

if (reset) begin

state <= IDLE;

buffer\_pointer <= 0;

ready <= 0;

process <= 0;

end else begin

case(state)

IDLE: begin

if (rx\_data == $8A) begin

buffer\_pointer <= 0;

state <= RX;

end

end

RX: begin

buffer[buffer\_pointer] <= rx\_data;

buffer\_pointer <= buffer\_pointer+1;

state <= (buffer\_pointer == BYTE\_COUNT) ? PARSE : RX;

end

PARSE: begin

ready <= 1;

state <= IDLE;

process <= 1;

end

ERROR: begin

state <= IDLE;

buffer\_pointer <= 0;

end

endcase

end

end

/\*

\*数据解析

\*/

always @(posedge rx\_clk) begin

if (process) begin

process <= 0;

if (buffer[3] == 'R' && buffer[4] == 'M' && buffer[5] == 'C') begin

message\_id = 0;

sat\_count\_b = "00000000";

end else if (buffer[3] == 'G' && buffer[4] == 'G' && buffer[5] == 'A') begin

message\_id = 1;

sat\_count\_b = buffer[8];

end else if (buffer[3] == 'G' && buffer[4] == 'L' && buffer[5] == 'L') begin

message\_id = 2;

sat\_count\_b = buffer[8];

end else if (buffer[3] == 'G' && buffer[4] == 'S' && buffer[5] == 'A') begin

message\_id = 3;

end else if (buffer[3] == 'G' && buffer[4] == 'S' && buffer[5] == 'V') begin

message\_id = 4;

end else if (buffer[3] == 'G' && buffer[4] == 'S' && buffer[5] == 'T') begin

message\_id = 5;

end else begin

message\_id = -1;

end

case(message\_id)

0: begin

//提取速度和航向信息

speed\_b[0] = buffer[DATA\_START+4];

speed\_b[1] = buffer[DATA\_START+5];

heading\_b[0] = buffer[DATA\_START+6];

heading\_b[1] = buffer[DATA\_START+7];

end

1, 2: begin

//提取UTC时间，纬度，经度，海拔信息

utc\_time[0] = buffer[DATA\_START];

utc\_time[1] = buffer[DATA\_START+1];

utc\_time[2] = buffer[DATA\_START+2];

latitude\_b[0] = buffer[DATA\_START+2];

latitude\_b[1] = buffer[DATA\_START+3];

latitude\_b[2] = buffer[DATA\_START+4];

longitude\_b[0] = buffer[DATA\_START+5];

longitude\_b[1] = buffer[DATA\_START+6];

longitude\_b[2] = buffer[DATA\_START+7];

altitude\_b[0] = buffer[DATA\_START+8];

altitude\_b[1] = buffer[DATA\_START+9];

altitude\_b[2] = buffer[DATA\_START+10];

end

3: begin

//提取卫星ID信息

end

4: begin

//提取卫星信息

end

5: begin

//提取HDOP信息

hdop\_b[0] = buffer[DATA\_START];

hdop\_b[1] = buffer[DATA\_START+1];

end

default: begin

//无法识别的消息

end

endcase

//格式转换

speed = {speed\_b[0], speed\_b[1]};

heading = {heading\_b[0], heading\_b[1]};

sat\_count = sat\_count\_b;

hdop = {hdop\_b[0], hdop\_b[1]};

//UTC时间转换

utc\_sec = ((utc\_time[0] & 4'h0F) \* 10) + (utc\_time[0] & 4'hF0) / 16;

utc\_min = ((utc\_time[1] & 4'h0F) \* 10) + (utc\_time[1] & 4'hF0) / 16;

utc\_hour = ((utc\_time[2] & 4'h0F) \* 10) + (utc\_time[2] & 4'hF0) / 16;

//纬度格式转换

latitude = {(latitude\_b[0] & 4'h0F) \* 10 + (latitude\_b[0] & 4'hF0) / 16,

(latitude\_b[1] & 4'h0F) \* 10 + (latitude\_b[1] & 4'hF0) / 16,

((latitude\_b[2] & 2'hFC) / 4'h04) \* 10 + (latitude\_b[2] & 2'h03),

(latitude\_b[2] & 4'h0C) \* 10 + (latitude\_b[3] & 4'hF0) / 16,

((latitude\_b[3] & 4'h0F) \* 10 + (latitude\_b[3] & 4'hF0) / 16) / 6};

//经度格式转换

longitude = {(longitude\_b[0] & 3'h0F) \* 100 + (longitude\_b[0] & 3'hF0) / 16 \* 10 + (longitude\_b[1] & 4'hF0) / 4'h10,

(longitude\_b[1] & 4'h0F) \* 10 + (longitude\_b[2] & 4'hF0) / 16,

((longitude\_b[2] & 4'h0F) \* 10 + (longitude\_b[3] & 4'hF0) / 16) / 6,

(longitude\_b[3] & 2'hFC) / 4'h04};

altitude = {(altitude\_b[0] & 4'h0F) \* 10000 + (altitude\_b[0] & 4'hF0) \* 1000 + (altitude\_b[1] & 3'hF0) \* 100 + (altitude\_b[2] & 4'hF0) / 6 \* 10 + (altitude\_b[2] & 4'h0F)};

end

end

endmodule

module gps\_processing(

input rx\_clk,

input rx\_data,

input reset,

output reg [7:0] latitude,

output reg [7:0] longitude,

output reg [7:0] altitude,

output reg [7:0] speed,

output reg [7:0] heading,

output reg [7:0] sms\_send //发送短信的信号

);

localparam BYTE\_COUNT = 66;

localparam DATA\_START = 8;

localparam STOP\_RATIO = 1.5; //轨迹限制比例

localparam MAX\_LATITUDE = 100; //纬度最大值

localparam MIN\_LATITUDE = 10; //纬度最小值

localparam MAX\_LONGITUDE = 115; //经度最大值

localparam MIN\_LONGITUDE = 75; //经度最小值

//定义寄存器

reg [7:0] buffer [BYTE\_COUNT-1:0];

reg [7:0] buffer\_pointer = 0;

reg ready = 0;

reg process = 0;

reg [7:0] sms = 0;

//定义GPS数据

reg [7:0] old\_latitude;

reg [7:0] old\_longitude;

reg [7:0] old\_speed;

reg [7:0] old\_heading;

//定义轨迹范围

reg [7:0] latitude\_max;

reg [7:0] latitude\_min;

reg [7:0] longitude\_max;

reg [7:0] longitude\_min;

//初始化寄存器

initial begin

old\_latitude = 0;

old\_longitude = 0;

old\_speed = 0;

old\_heading = 0;

latitude\_max = MAX\_LATITUDE;

latitude\_min = MIN\_LATITUDE;

longitude\_max = MAX\_LONGITUDE;

longitude\_min = MIN\_LONGITUDE;

end

//等待启动

always @(posedge rx\_clk) begin

if (reset) begin

buffer\_pointer <= 0;

ready <= 0;

process <= 0;

latitude <= 0;

longitude <= 0;

altitude <= 0;

speed <= 0;

heading <= 0;

end else begin

case(process)

0: begin

if (rx\_data == $8A) begin

buffer\_pointer <= 0;

process <= 1;

end

end

1: begin

buffer[buffer\_pointer] <= rx\_data;

buffer\_pointer <= buffer\_pointer+1;

process <= (buffer\_pointer == BYTE\_COUNT) ? 2 : 1;

end

2: begin

ready <= 1;

buffer\_pointer <= 0;

process <= 0;

end

default: begin

process <= 0;

end

endcase

end

end

//数据解析

always @(posedge rx\_clk) begin

if (ready) begin

ready <= 0;

if (buffer[3] == 'G' && buffer[4] == 'G' && buffer[5] == 'A') begin

//提取UTC时间，纬度，经度，海拔信息

latitude[0] = buffer[DATA\_START+2];

latitude[1] = buffer[DATA\_START+3];

latitude[2] = buffer[DATA\_START+4];

longitude[0] = buffer[DATA\_START+5];

longitude[1] = buffer[DATA\_START+6];

longitude[2] = buffer[DATA\_START+7];

altitude[0] = buffer[DATA\_START+8];

altitude[1] = buffer[DATA\_START+9];

altitude[2] = buffer[DATA\_START+10];

end else if (buffer[3] == 'R' && buffer[4] == 'M' && buffer[5] == 'C') begin

//提取速度和航向信息

speed[0] = buffer[DATA\_START+4];

speed[1] = buffer[DATA\_START+5];

heading[0] = buffer[DATA\_START+6];

heading[1] = buffer[DATA\_START+7];

end

end

end

//判断是否越界并发短信

always @(posedge rx\_clk) begin

if (old\_latitude != 0 && old\_longitude != 0) begin

//当前轨迹范围

if (latitude > latitude\_max || latitude < latitude\_min || longitude > longitude\_max || longitude < longitude\_min) begin

sms\_send <= 1;

sms <= 1;

end else begin

sms\_send <= 0;

end

//更新轨迹范围

if (old\_latitude > latitude) begin

latitude\_max <= old\_latitude + (old\_latitude - latitude) / STOP\_RATIO;

end else if (old\_latitude < latitude) begin

latitude\_min <= old\_latitude - (latitude - old\_latitude) / STOP\_RATIO;

end

if (old\_longitude > longitude) begin

longitude\_max <= old\_longitude + (old\_longitude - longitude) / STOP\_RATIO;

end else if (old\_longitude < longitude) begin

longitude\_min <= old\_longitude - (longitude - old\_longitude) / STOP\_RATIO;

end

end else begin

sms\_send <= 0;

end

old\_latitude <= latitude;

old\_longitude <= longitude;

old\_speed <= speed;

old\_heading <= heading;

end

//短信发送

always @(posedge rx\_clk) begin

if (sms) begin

//发送短信

//连接SIM900A

//填充短信内容：http://maps.google.com/maps?q=latitude,longitude

//发送短信

//拨通号码

//......

sms <= 0;

end

end

endmodule

module sim900a\_sms(

input rx\_clk,

input rx\_data,

input reset,

output reg led, //指示灯

output reg alert //警报

);

localparam BYTE\_COUNT = 160;

localparam SMS\_HEADER\_LENGTH = 6; //短信头部长度

//定义寄存器

reg [7:0] buffer [BYTE\_COUNT-1:0];

reg [7:0] buffer\_pointer = 0;

reg ready = 0;

reg process = 0;

reg [1:0] sms\_state = 0;

reg [7:0] sms\_content [BYTE\_COUNT-1:0];

reg [7:0] sms\_pointer = 0;

//初始化寄存器

initial begin

buffer\_pointer = 0;

ready = 0;

process = 0;

sms\_state = 0;

sms\_content = '{';

sms\_pointer = 0;

led = 0;

alert = 0;

end

//等待启动

always @(posedge rx\_clk) begin

if (reset) begin

buffer\_pointer <= 0;

ready <= 0;

process <= 0;

sms\_state <= 0;

sms\_pointer <= 0;

end else begin

case(process)

0: begin

if (rx\_data == $0A) begin

buffer\_pointer <= 0;

process <= 1;

end

end

1: begin

buffer[buffer\_pointer] <= rx\_data;

buffer\_pointer <= buffer\_pointer+1;

process <= (buffer\_pointer == BYTE\_COUNT) ? 2 : 1;

end

2: begin

ready <= 1;

buffer\_pointer <= 0;

process <= 0;

end

default: begin

process <= 0;

end

endcase

end

end

//接收短信

always @(posedge rx\_clk) begin

if (ready && buffer[1] == 'C' && buffer[2] == 'M' && buffer[3] == 'T' && buffer[4] == 'I') begin

//提取短信号码

reg [19:0] phone\_number;

phone\_number[0] = '+';

phone\_number[1] = buffer[SMS\_HEADER\_LENGTH];

phone\_number[2] = buffer[SMS\_HEADER\_LENGTH+1];

phone\_number[3] = buffer[SMS\_HEADER\_LENGTH+2];

phone\_number[4] = buffer[SMS\_HEADER\_LENGTH+3];

phone\_number[5] = buffer[SMS\_HEADER\_LENGTH+4];

phone\_number[6] = buffer[SMS\_HEADER\_LENGTH+5];

phone\_number[7] = buffer[SMS\_HEADER\_LENGTH+6];

phone\_number[8] = buffer[SMS\_HEADER\_LENGTH+7];

phone\_number[9] = buffer[SMS\_HEADER\_LENGTH+8];

phone\_number[10] = buffer[SMS\_HEADER\_LENGTH+9];

phone\_number[11] = buffer[SMS\_HEADER\_LENGTH+10];

phone\_number[12] = buffer[SMS\_HEADER\_LENGTH+11];

phone\_number[13] = buffer[SMS\_HEADER\_LENGTH+12];

phone\_number[14] = buffer[SMS\_HEADER\_LENGTH+13];

phone\_number[15] = buffer[SMS\_HEADER\_LENGTH+14];

phone\_number[16] = buffer[SMS\_HEADER\_LENGTH+15];

phone\_number[17] = buffer[SMS\_HEADER\_LENGTH+16];

phone\_number[18] = buffer[SMS\_HEADER\_LENGTH+17];

phone\_number[19] = buffer[SMS\_HEADER\_LENGTH+18];

if (phone\_number == "+1234567890") begin //判断短信是否来自指定号码

for (int i = 0; i < buffer\_pointer; i = i+1) begin

if (sms\_state == 0 && buffer[i] == ' ') begin

sms\_state <= 1;

end else if (sms\_state == 1 && buffer[i] != ' ') begin //提取短信指令

sms\_content[sms\_pointer] <= buffer[i];

sms\_pointer <= sms\_pointer+1;

sms\_state <= 2;

} else if (sms\_state == 2 && buffer[i] != '\r') begin

sms\_content[sms\_pointer] <= buffer[i];

sms\_pointer <= sms\_pointer+1;

end else if (sms\_state == 2 && buffer[i] == '\r') begin

sms\_state <= 3;

end else if (sms\_state == 3 && buffer[i] == '\n') begin

if (sms\_pointer == 7 && sms\_content[0] == 'g' && sms\_content[1] == 'o' && sms\_content[2] == ' ' &&

sms\_content[3] == 'h' && sms\_content[4] == 'o' && sms\_content[5] == 'm' && sms\_content[6] == 'e') begin

led <= 1;

} else if (sms\_pointer == 4 && sms\_content[0] == 's' && sms\_content[1] == 'e' && sms\_content[2] == 'e' &&

sms\_content[3] == 'k') begin

alert <= 1;

}

sms\_state <= 0;

sms\_pointer <= 0;

sms\_content <= '{';

end

end

end

ready <= 0;

end

end

endmodule

module sos\_send (

input clk, // 输入时钟信号

input reset\_n, // 复位信号

input button, // 按钮

output reg led, // LED指示灯

output reg sim\_tx // SIM900A 发送数据序列

);

// 定义常量和参数

parameter BAUD\_RATE = 9600;

localparam GPS\_DATA\_LENGTH = 20;

localparam SMS\_MAX\_LENGTH = 160;

localparam SOS\_TEXT = "SOS - HELP NEEDED! \n";

localparam ERROR\_MSG = "ERROR: UNABLE TO RETRIEVE GPS DATA \n";

// 定义寄存器和变量

reg [31:0] timer = 0; // 按钮计时器

reg [31:0] gps\_timer = 0; // GPS 模块计时器

reg [31:0] gps\_div; // GPS 模块波特率调整

reg [7:0] buffer[GPS\_DATA\_LENGTH-1:0]; // GPS 数据缓存

reg [7:0] sms\_buffer[SMS\_MAX\_LENGTH-1:0]; // 短信缓存

reg [7:0] sms\_counter = 0; // 短信计数器

reg [7:0] buffer\_count = 0; // GPS 数据计数器

reg [1:0] state = 0; // 状态机，用来保存当前状态6

reg gps\_ready = 0; // GPS 连接状态

reg sms\_sent = 0; // 短信发送状态

reg gps\_read = 0; // GPS 数据读取状态

reg can\_send = 0; // 是否可以发送短信的状态

reg tx\_ready = 0; // SIM900A 发送缓冲区状态

// 状态机

always @(posedge clk, negedge reset\_n) begin

if (~reset\_n) begin

state <= 0;

gps\_timer <= 0;

gps\_read <= 0;

buffer\_count <= 0;

sms\_counter <= 0;

timer <= 0;

sms\_sent <= 0;

gps\_ready <= 0;

can\_send <= 0;

end else begin

case (state)

0: begin // 空闲状态

timer <= 0;

led <= 0;

if(button == 0) begin

state <= 1;

timer <= 0;

end

end

1: begin // 按钮按下状态

timer <= timer + 1;

led <= 1;

if(timer >= 10000000) begin

state <= 0;

end

if(button == 1) begin

state <= 0;

end

end

2: begin // 检查GPS状态状态

gps\_timer <= gps\_timer + 1;

if(gps\_timer >= 1000000) begin

gps\_ready <= 1;

sms\_counter <= 0;

state <= 3;

end

end

3: begin // 读取GPS数据状态

if(~gps\_read) begin

buffer[buffer\_count] <= SIM900A\_DATA\_IN;

buffer\_count <= buffer\_count + 1;

if(buffer[buffer\_count-1] == 0x0A && buffer[buffer\_count-2] == 0x0D) begin

gps\_ready <= 0;

gps\_read <= 1;

buffer\_count <= 0;

sms\_counter <= 0;

state <= 4;

end

end

end

4: begin // 拼接短信并预处理状态

sms\_buffer <= SOS\_TEXT;

sms\_counter <= strlen(SOS\_TEXT);

if(~gps\_ready) begin

state <= 5;

end else begin

for(i=0; i<buffer\_count; i=i+1) begin

sms\_buffer[sms\_counter+i] <= buffer[i];

end

sms\_counter <= sms\_counter + buffer\_count;

buffer\_count <= 0;

if(sms\_counter + 14 > SMS\_MAX\_LENGTH) begin

state <= 5;

end else begin

state <= 2;

end

end

end

5: begin // 发送错误信息状态

sms\_buffer <= ERROR\_MSG;

sms\_counter <= strlen(ERROR\_MSG);

can\_send <= 1;

gps\_read <= 0;

state <= 6;

end

6: begin // 等待SIM900A发送缓存区准备状态

if(tx\_ready) begin

if(can\_send) begin

sms\_sent <= 1;

for(i=0; i<strlen(sms\_buffer); i=i+1) begin

sim\_tx <= sms\_buffer[i];

end

sim\_tx <= 0x1A; // 发送结束

end

state <= 0;

end

end

endcase

end

end

// 波特率调整

always @(posedge clk, negedge reset) begin

if(~reset) begin

gps\_div <= 0;

end else begin

if(gps\_div >= clk/BAUD\_RATE) begin

gps\_div <= 0;

gps\_read <= 1;

end else begin

gps\_div <= gps\_div + 1;

gps\_read <= 0;

end

end

end

// SIM900A 发送缓存区状态检查

always @(posedge clk, negedge reset) begin

if(~reset) begin

tx\_ready <= 0;

end else begin

if(sim\_tx == 0) begin

tx\_ready <= 1;

end else begin

tx\_ready <= 0;

end

end

end

endmodule