GPS定位的型号为NEO-6M，对于接收到的GPS定位信息进行解码，并提取出所需的信息。然后将这些信息存储到FPGA开发板上的内存中，以便于其他模块的使用

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

先定义了常量和寄存器，并在状态机的不同状态中对接收到的数据进行相应的解析和处理。接下来，我们分别对于速度、航向、UTC时间、纬度、经度、海拔和卫星数等信息进行了提取和格式转换，以便于存储到FPGA开发板上的内存中。请根据实际情况进行修改和优化。

