使用NEO-6M作为GPS接收器模块，通过 FPGA 的数字信号处理器（DSP）功能，来快速且有效地处理 GPS 信号数据。最后使用 FPGA 的片上存储器来进行数据缓存和存储，用来作为其他功能的使用。

以下是一个简单的程序示例，使用 Terasic DE2-115 开发板上的 FPGA，通过 NEO-6M GPS 模块进行数据采样和数字信号处理，最终将数据存储在片上存储器中。

module GPS\_module(

input clk, // 输入时钟

input reset, // 复位信号

input [7:0] GPS\_data, // GPS 信号数据输入

output reg [7:0] GPS\_coord [511:0], // 存储 GPS 坐标的 RAM

output reg valid\_out // 用于指示是否读取了有效坐标

);

reg [2:0] state; // 状态寄存器

reg [7:0] data\_temp; // 采样数据缓存

reg [7:0] GPS\_coord\_temp; // GPS 坐标缓存

wire [7:0] DSP\_output; // DSP 模块输出

wire valid; // DSP 模块有效输出

// 实例化 DSP 模块

dsp\_module DSP\_inst (

.clk(clk),

.reset(reset),

.data\_in(data\_temp),

.data\_out(DSP\_output),

.valid(valid)

);

// 存储器控制时序

reg [8:0] counter;

wire [8:0] address;

assign address = counter << 1;

// 状态机

always @(posedge clk or posedge reset) begin

if (reset) begin

state <= 0;

valid\_out <= 0;

GPS\_coord <= 0;

counter <= 0;

end else begin

case (state)

0: begin // 采样状态

data\_temp <= GPS\_data;

state <= 1;

end

1: begin // DSP 模块处理状态

DSP\_inst.data\_in <= data\_temp;

state <= 2;

end

2: begin // 判断 DSP 模块输出有效状态

if (valid) begin

GPS\_coord\_temp <= DSP\_output;

state <= 3;

end else begin

state <= 0;

end

end

3: begin // 存储 GPS 坐标状态

GPS\_coord[address] <= GPS\_coord\_temp;

counter <= counter + 1;

if (counter == 255) begin

valid\_out <= 1;

end

if (counter == 511) begin

counter <= 0;

end

state <= 0;

end

default: state <= 0;

endcase

end

end

endmodule

```

上述程序实现了将 GPS 信号数据采样、数字信号处理（这里假设使用一个名为 DSP\_inst 的 DSP 模块，模拟处理过程并将处理结果存储在 DSP\_output 输出端口中）、判断 DSP 模块输出是否有效，以及将校准后的坐标存储到片上存储器中。具体实现中，使用了一个三位状态机，以及计数器和地址计算逻辑。

需要注意的是，上述程序示例仅仅是展示了一个较为简单的实现过程，实际情况中，您可能需要更加详细地考虑到项目需求中的各种参数，并进行相应的优化和调整才能完成该项目的设计。

Verilog 不支持在模块端口或函数参数中使用非压缩数组类型，在这种情况下，需要使用 SystemVerilog 中提供的 typedef 语法来定义一个新的类型，然后在模块端口中使用该类型。以下是一个可能的示例：

typedef logic [7:0] coord\_t; // 定义新的类型

module GPS\_module(

input clk,

input reset,

input [7:0] GPS\_data,

output reg valid\_out,

output coord\_t GPS\_coord [511:0] // 在模块端口中使用新类型

);

// ...

endmodule

在这个修改后的示例中，我们使用了 SystemVerilog 中的 typedef 语法定义了一个名为 coord\_t 的新类型，它是一个包含 8 位数据的逻辑数组。然后在 GPS\_module 模块中使用了该新类型来声明一个名为 GPS\_coord 的非压缩数组端口。这个修改后的版本在 Quartus 中应该能够通过编译。

请注意，这是一个参考示例，实际的修改可能需要根据您的具体情况进行进一步调整。同时，在实际设计过程中，使用一个堆栈或 FIFO 等数据结构来存储数据也是一种常见的选择。

