《移动通信系统应用设计》任务书

学生姓名：钱旭　　专业班级：通信 ZY1701

指导教师：胡辑伟　　工作单位：信息工程学院

题　目：基于 FPGA 的 16QAM 调制解调算法设计

初始条件：

Quartus II、FPGA、Modelsim、Matlab

要求完成的主要任务：（包括课程设计工作量及其技术要求，以及说明书撰写等具体要求）

1、理解基于 FPGA 的 16QAM 调制解调算法基本原理，编写项目的设计方案

2、编写代码，进行调试，计算相关数据，实现相关功能，满足项目的要求

3、编写课程设计报告书，参考文献不得少于 5 篇

参考书：

与报告相关的参考书籍

[1] 郭田耘，徐文波等.无线通信 FPGA 设计[M].电子工业出版社，2004，10.

[2] 曹志刚，钱亚生. 现代通信原理[M]. 北京：清华大学出版社，2007.

[3] 戴振，李维英，蔡峰.基于 FPGA 的 16QAM 调制器设计与实现[J].电子元器件应用，2007(03):52-54.

时间安排：

1、理论讲解，老师布置课程设计题目，学生根据选题开始查找资料；

2、课程设计时间为 2 周。

（1）理解相关技术原理，确定技术方案， 时间 2 天；

（2）选择仿真工具，进行仿真设计与分析，时间 6 天；

（3）总结结果，完成课程设计报告，时间 2 天。

指导教师签名：　　年　月　日

系主任（或责任教师）签名：　　年　月　日

目录

摘要.....1
Abstract.....2
1 QAM 调制解调原理.....3
1.1 QAM 调制原理.....3
1.2 16QAM 调制.....4
1.3 16QAM 解调.....5
2 基于 FPGA 的 16QAM 模块设计.....5
2.1 16QAM 调制解调总体流程.....5
2.2 16QAM 调制.....6
2.2.1 数据源模块.....6
2.2.2 分频模块.....6
2.2.3 串并转换模块.....7
2.2.4 载波生成模块.....7
2.2.5 调制乘法模块.....8
2.2.6 16QAM 调制顶层模块.....9
2.3 16QAM 解调.....9
2.3.1 解调乘法模块.....9
2.3.2 低通滤波器模块.....10
2.3.3 定时判决模块.....11
2.3.4 并串转换模块.....11
2.3.5 16QAM 解调顶层模块.....12
2.4 16QAM 系统顶层模块.....12
3 仿真测试结果.....13
3.1 数据源模块仿真.....13
3.2 串并转换模块仿真.....13
3.3 载波生成模块仿真.....14
3.4 16QAM 调制模块仿真.....14
3.5 16QAM 解调模块仿真.....15
3.6 16QAM 顶层模块仿真.....15
4 小结.....16
参考文献.....17
附录.....18

摘要

QAM（Quadrature Amplitude Modulation）是一种的高效的调制方式，其幅度和相位同时变化，属于非恒包络二维调制，其比只利用单一维度空间资源的 PSK 和 ASK 调制方式频率利用率高。16QAM 是指包含 16 种符号的 QAM 调制方式，其调制的方法有正交幅度法和复合相移法两种方法。本文使用 FPGA 设计软件 Quartus II 和其仿真软件 Modelsim，使用硬件描述语言 Verilog，采用正交幅度法设计了一个 16QAM 的调制解调系统，并通过仿真观察和分析其波形。

关键词：16QAM　Verilog　FPGA

Abstract

QAM (Quadrature Amplitude Modulation) is an efficient modulation method with simultaneous amplitude and phase changes. It belongs to non-constant envelope two-dimensional modulation. It has higher frequency utilization than PSK and ASK modulation methods which only use single-dimensional spatial resources.16QAM refers to the QAM modulation method which contains 16 symbols. The modulation methods are orthogonal amplitude method and composite phase shift method.In this paper, a 16QAM modem system is designed by using the orthogonal amplitude method using the hardware description language Verilog, the design software Quartus II and its simulation software Modelsim, and its waveform is observed and analyzed by simulation.

Keywords: 16QAM Verilog FPGA

1 QAM 调制解调原理

1.1 QAM 调制原理

QAM 是用两路独立的数字基带信号对两个相互正相交的同频载波进行抑制载波的 DSB 调制，利用这种已调信号在同一带宽内频谱正交的性质来实现两路并行的数字信息传输。

M 进制正交振幅调制信号一般表示为（1.1），式子中A_n是基带信号幅度；φ_n是基带信号相位，它们分别可以取多个离散值；$g(t-nT_B)$是宽度为T_B的单个信号波形。

$$S_{MQAM}(t)=\sum_n A_n g(t-nT_B)\cos(\omega_c t+\varphi_n) \tag{1.1}$$

式（1.1）还可以变换为正交表示形式：

$$S_{MQAM}(t)=[\Sigma_n A_n g(t-nT_B)\cos\varphi_n]\cos\omega_c t-[\Sigma_n A_n g(t-nT_B)\sin\varphi_n]\sin\omega_c t \tag{1.2}$$

令$X_n=A_n cos\varphi_n$，$Y_n=Y_n sin\varphi_n$，则式（1.2）可以变为式（1.3）

$$S_{MQAM}(t)=[\Sigma_n X_n g(t-nT_B)]cos\omega_c t+[\Sigma_n Y_n g(t-nT_B)]sin\omega_c t \tag{1.3}$$

QAM 中的振幅还X_n和Y_n还可以表示为(1.4)，式子中 A 是固定振幅；c_n和d_n是由输入数据确定的离散值，它们决定了已调 QAM 信号在信号中的坐标点。

$$\begin{cases}X_n=c_n A\\ Y_n=d_n A\end{cases} \tag{1.4}$$

QAM 的调制方法如图 1.1 所示，本文采用正交幅度法，在发送数据时将数据分为两路，与两个正交的信号相乘，得到两个调制后的正交信号，求和后就是需要输出的 QAM 调制信号。信号矢量端点的分布图称为星座图，通常可以用星座图来描述 QAM 信号的信号空间分布状态。

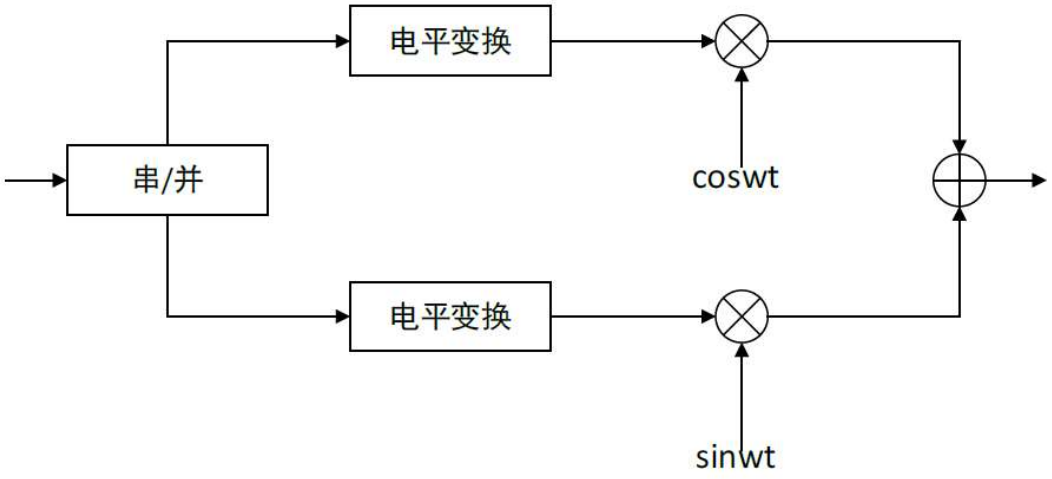

图 1.1 QAM 调制原理图

1.2 16QAM 调制

16QAM 信号在星座图上具有 16 个样点，每个样点表示一种矢量状态，16QAM 有 16 个态，每四位二进制数规定了 16 态中的一态，星座图中每个点的编码为格雷码的形式。16QAM 的星座图有方型和星型两种，在本文种使用方型的星座图来进行调制。

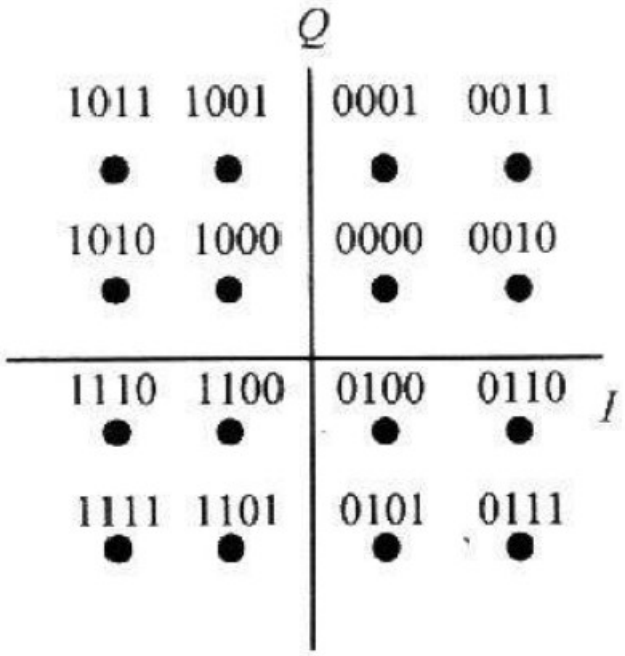

图 1.2 16QAM 方型星座图

图 1.2 为 16QAM 的方型星座图，其对应的编码如表 1.1 所示，其中 I 对应第 0 位和第 4 位，Q 对应第 1 位和第 3 位。经过编码表映射后得到的数据对两个正向交的载波进行相乘，总而实现调制。

表 1.1 16QAM 编码表

$b_0b_1b_2b_3$	I	Q	$b_0b_1b_2b_3$	I	Q
0000	1	1	1000	-1	1
0001	1	3	1001	-1	3
0010	3	1	1010	-3	1
0011	3	3	1011	-3	3
0100	1	-1	1100	-1	-1
0101	1	-3	1101	-1	-3
0110	3	-1	1110	-3	-1
0111	3	-3	1111	-3	-3

1.3 16QAM 解调

16QAM 信号采用正交相干解调的方法解调，解调器首先对接收到的信号分为两路，一路与$cos\omega_t$相乘，一路与$sin\omega_t$相乘，进行相干解调。将相乘后的信号经过一个低通滤波器进行滤波，消除高频分量，获得滤波后的低频信号，滤波输出后进行抽样判决，恢复出原先的电平信号，再进行并串转换获得原先输入。16QAM 的解调流程如图 1.3 所示。

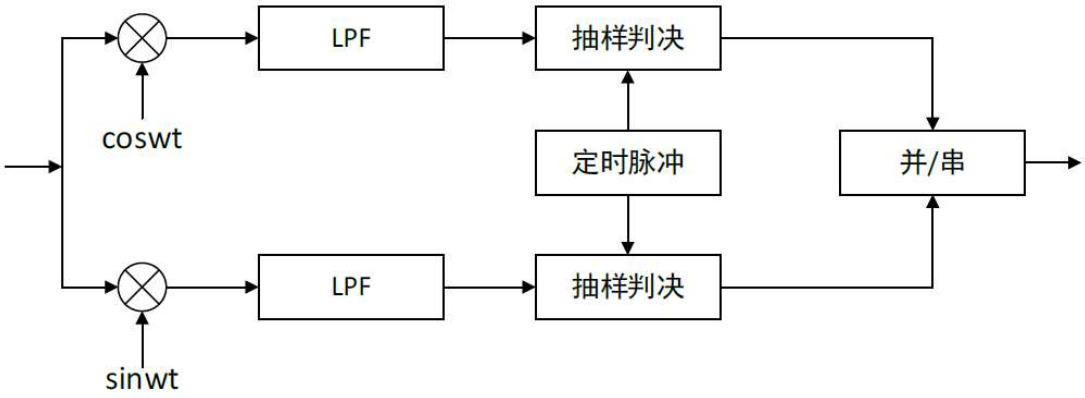

图 1.3 16QAM 解调流程图

2 基于 FPGA 的 16QAM 模块设计

2.1 16QAM 调制解调总体流程

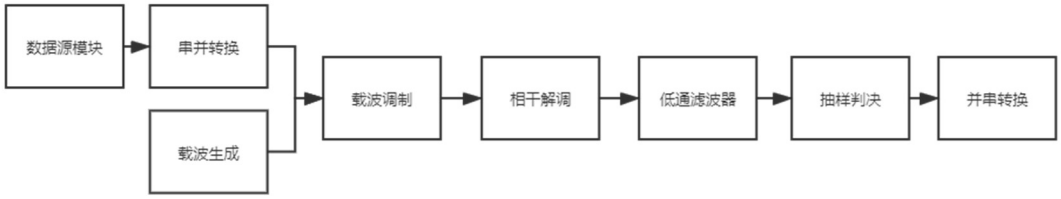

图 2.1 16QAM 整体模块图

数据源模块将输出原始的二进制序列，串并转换模块将每 4 位串行输入转换为 4 位，使用 Matlab 对两个载波的波形进行采样，生成.mif 文件，导入到 Quarrtus II 生成的 ROM 的 IP 核中，对其例化并且读取，输出相互正交的正交和余弦信号。调制模块中将串并转换的 4 位信号分为 I、Q 两路，分别与两路载波信号相乘，相乘后再求和输出。相干解调模块中将调制信号分为两路，分别乘以正交的正弦与余弦信号，再通过经过 Matlab 设计的低通滤波器，得到的滤波输出经过抽样判决得到 4 位并行输出，最后经过并行转串行转换，获得原串行数据。

2.2 16QAM 调制

2.2.1 数据源模块

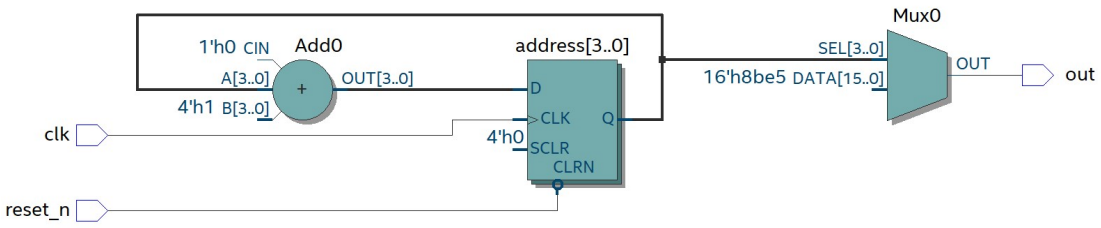

图 2.2 数字信号源模块

图 2.2 为数据信号源模块的 RTL 图，使用一组序列 1010_0111_1101_0001 作为需要传输的信号。输入参数为 clk 时钟信号和 reset_n 复位信号（低电平有效），out 为输出的信号。每次时钟跳变就读出其中的一位数据。读完 16 位数据后重新读取第 1 位数据。

2.2.2 分频模块

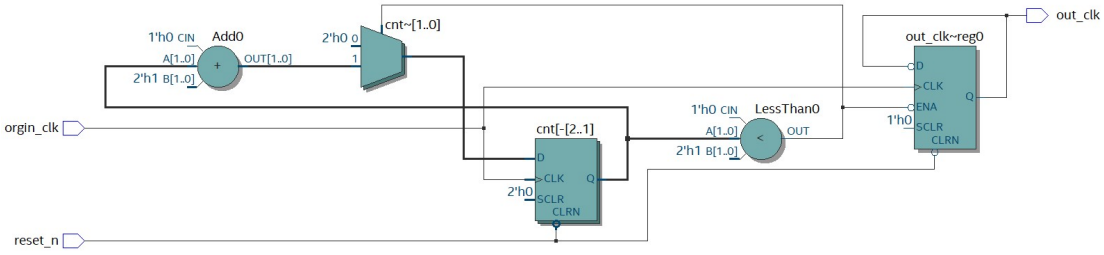

图 2.3 分频模块

图 2.3 为分频模块的 RTL 图。因为需要分频的地方比较多，所以使用 parameter 关键词进行参数化设计，在例化的时候指定分频系数。其中输入参数 orgin_clk 为输入时钟，reset_n 为复位信号（低电平有效），输出参数为 out_clk 为分频后的时钟。在调制模块中需要将 4 位串行数据转换位 4 位并行数据，因此设置分频系数为 4 分频。

2.2.3 串并转换模块

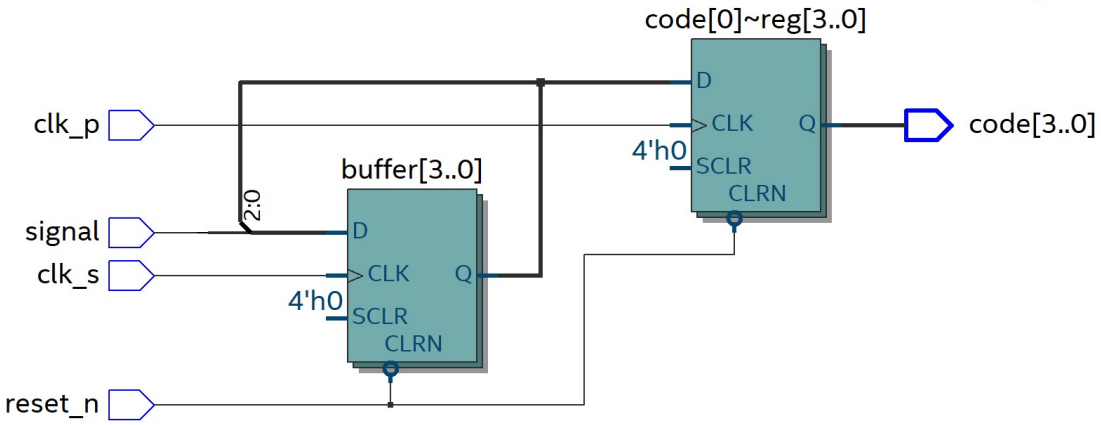

图 2.4 串并转换模块

图 2.4 为串并转换模块的 RTL 图，输入参数 clk_p 为并行数据时钟，signal 为串行数据输入，clk_s 为串行数据时钟，reset_n 为复位信号（低电平有效），code 为并行输出。设计时设置并行时钟为串行时钟的四分频，并在模块内使用一个 4 位宽的 buffer 缓存来存放串行数据。每遇到一个串行时钟上升沿的时候，将一位串行数据存入缓存，当遇到并行时钟上升沿的时候将缓存输出，达到串行数据转并行数据的效果。

2.2.4 载波生成模块

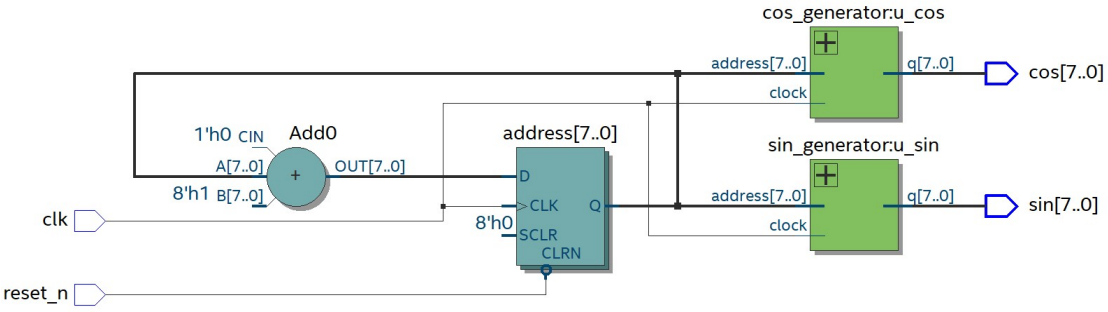

图 2.5 载波生成模块

图 2.5 为载波生成模块的 RTL 图，输入参数 clk 为输入时钟，reset_n 为复位信号，cos 和 sin 分别为输出的余弦和正弦载波。利用 Matlab 对正弦波进行采样，采样 256 个点，采样位宽为 8 位，得到.mif 文件，将 mif 文件导入生成的 ROM IP 核里面，得到两个包含正余弦载波信息 ROM 的模块，例化模块并且根据时序读出，这样就可以得到正余弦载波。

2.2.5 调制乘法模块

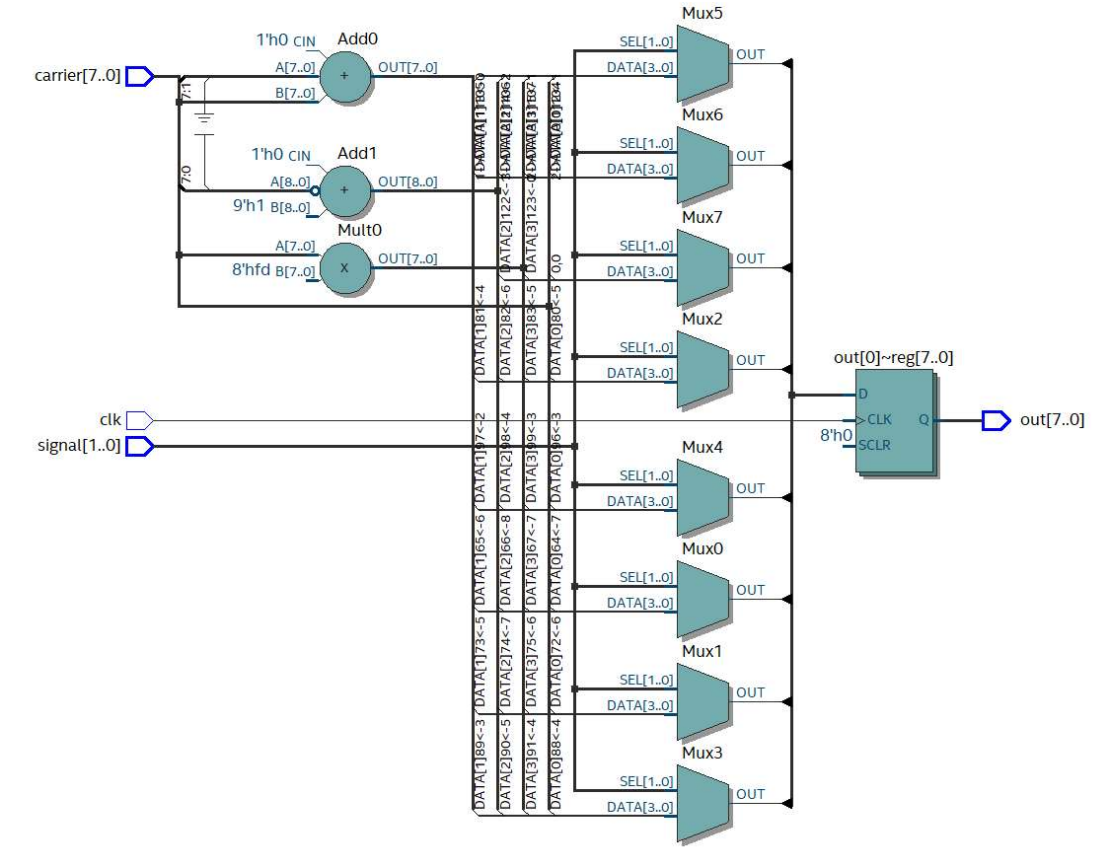

图 2.6 调制乘法模块

图 2.6 为调制乘法模块的 RTL 图，signal 为两位 I/Q 数字信号输入，clk 为时钟信号，carrier 为载波信号输入，out 为乘法输出。根据 16QAM 星座图的编码，设置输出倍率，当 signal 为 00 的时候，输出原先的载波；为 01 的时候，输出原来的 3 倍；10 的时候，输出原来的-1 倍；11 的时候输出原来的-3 倍。I 路和 Q 路调制方法一样，因此两路各自例化一个乘法模块就可以完成调制。

2.2.6 16QAM 调制顶层模块

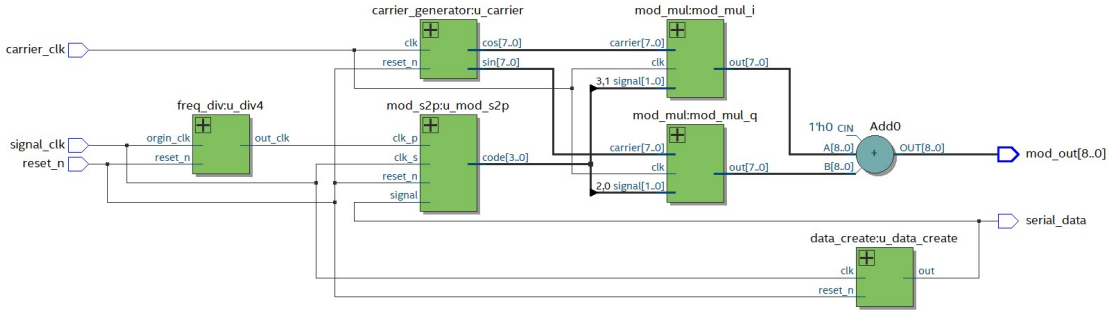

图 2.7 16QAM 调制顶层模块

图 2.7 为 16QAM 调制的顶层模块的 RTL 图，输入参数 carrier_clk 为载波输入时钟，signal_clk 为数字信号源输入时钟，reset_n 为复位信号（低电平有效），mod_out 为 16QAM 的调制输出，serial_data 为数字信号源产生的串行数据。I、Q 两路经过相乘后，再进行相加得到调制后的输出。I、Q 两路的输出信号为 8 位有符号信号，为了防止加法溢出因此设置输出为 9 位有符号信号。

2.3 16QAM 解调

2.3.1 解调乘法模块

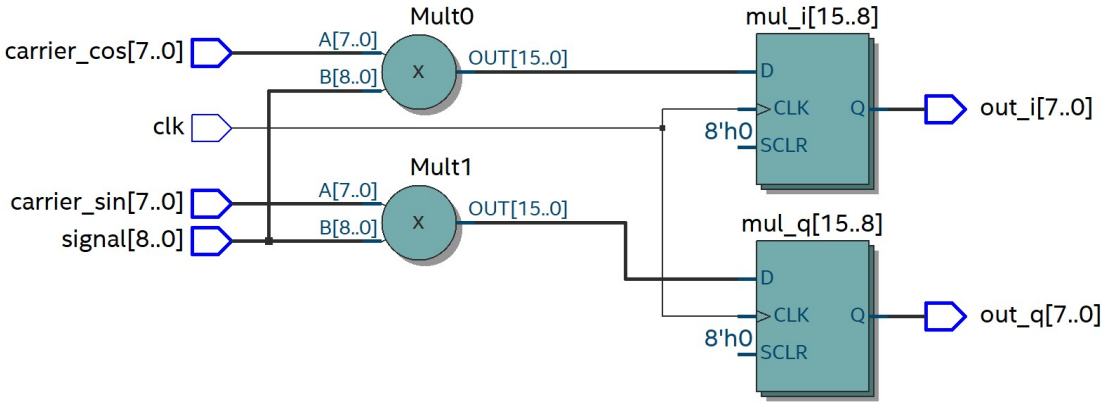

图 2.8 解调乘法模块

图 2.8 为解调乘法模块的 RTL 图，输入参数 carrier_cos 和 carrier_sin 为与调制信号同频率的正余弦载波，clk 为输入的载波频率，signal 为经过调制后的载波信号，out_i 和 I 路乘法输出，out_q 为 Q 路乘法输出。

2.3.2 低通滤波器模块

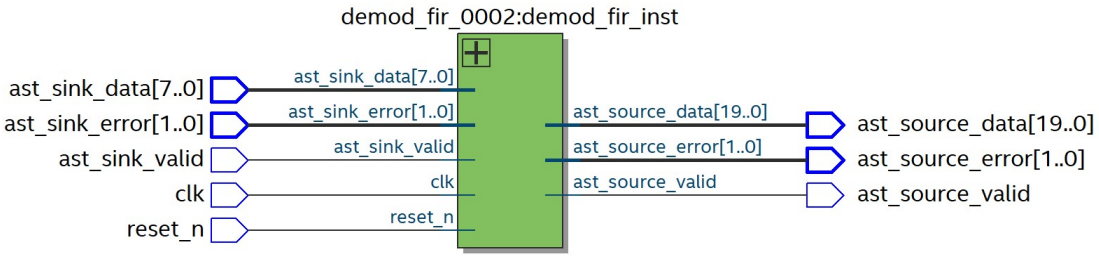

图 2.9 低通 FIR 模块

图 2.9 为低通 FIR 滤波器的 RTL 图，在本次设计中输入的载波频率为 50MHz，而采样点数为 256 个，因此输出的载波频率约为 195KHz，设置串行数据的时钟为 10KHz，滤波输入的信号频率大约有 10KHz、195KHz、390KHz，滤波结果只需要 10KHz。借助 Matlab 的 Filter Designer 工具设计滤波器，设计结果如图 2.10 所示，采样频率 Fs 为 500KHz，通带频率为 10KHz，截止频率为 100KHz，获得滤波器的阶数和参数，并导入到 Quarrtus II 生成的 IP 核里面。输入参数 clk 为时钟频率，也就是采样频率，reset_n 为复位信号，ast_sink_vaild 为输入有效信号，ast_sink_error 为输入错误类型，ast_sink_data 为输入的待滤波数据，输出参数 ast_source_data 为滤波结果，ast_source_error 是输出的错误类型，ast_source_valid 为输出是否有效。

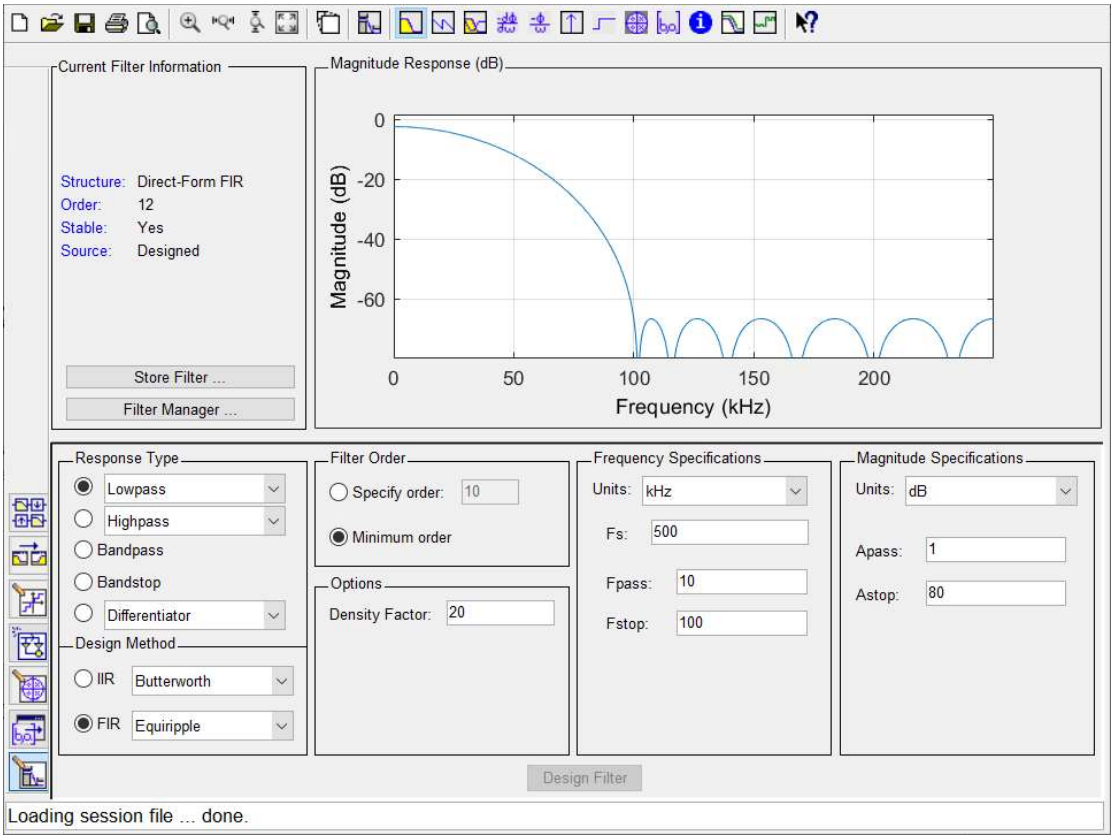

图 2.10 Matlab 设计 FIR 滤波器

2.3.3 定时判决模块

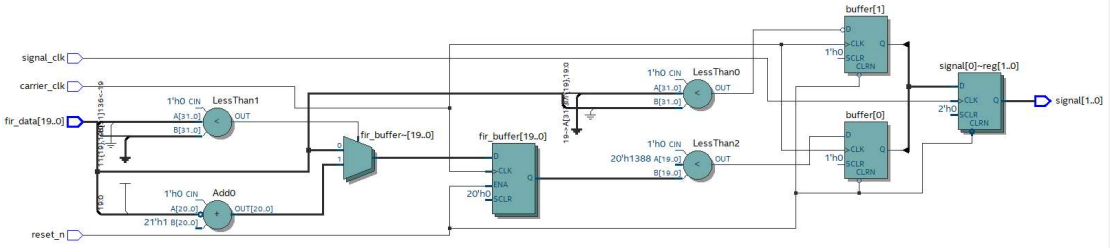

图 2.11 定时判决模块

图 2.11 为定时判决模块的 RTL 图，输入参数 signal_clk 为判决时钟，carrier_clk 为载波的频率，fir_data 为滤波器的输出，reset_n 为复位信号（低电平有效），输出参数 signal 为判决输出。模块内设置一个缓存，在每次载波时钟上升沿的时候对其判决并存入缓存中，当判决时钟上升沿的时候将判决输出。

2.3.4 并串转换模块

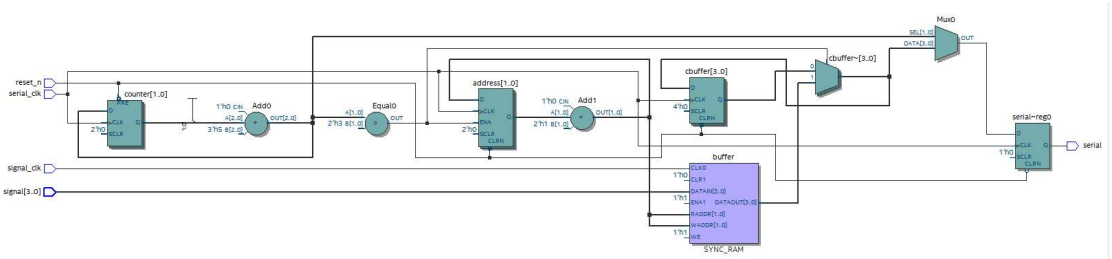

图 2.12 并串转换模块

图 2.12 为并串转换模块的 RTL 图，输入参数 reset_n 为复位信号，serial_clk 为串行时钟频率，signal 为并行数据，signal_clk 为输入并行时钟的频率。输出参数 serial 为串行输出。设计该模块时使用了一个缓存，在每个并行数据的时钟上升沿的时将并行数据存入缓存，当串行时钟的上升沿到来时，将按序将串行数据输出。

2.3.5 16QAM 解调顶层模块

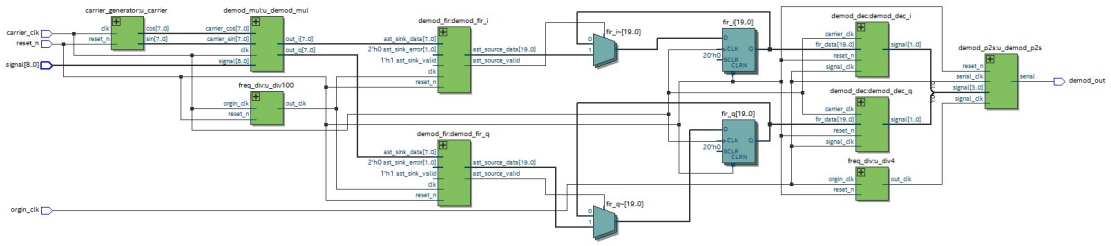

图 2.13 16QAM 顶层模块

图 2.13 为 16QAM 顶层模块的 RTL 图，输入参数 carrier_clk 为载波信号的时钟输入，reset_n 为复位信号，signal 为 16QAM 输入信号，origin_clk 为串行输出的时钟。将上文中的解调模块都进行例化，并将 I、Q 两路信号解调信号合并重新排序后，输入到并串转换模块中，得到解调后的串行信号。

2.4 16QAM 系统顶层模块

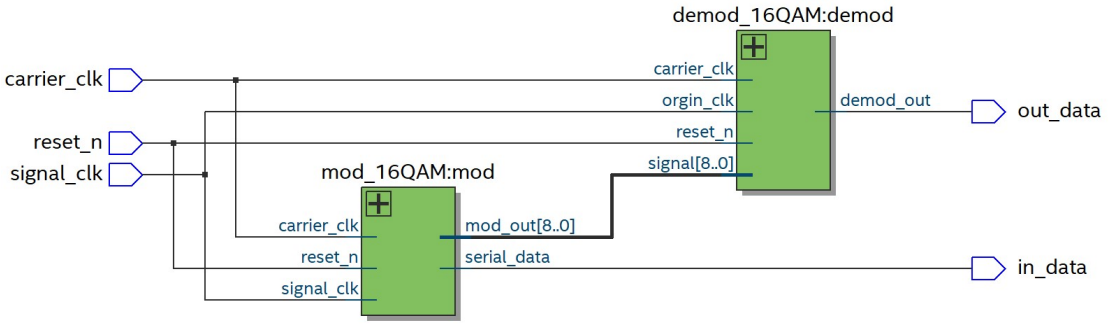

图 2.14 16QAM 系统顶层模块

图 2.14 为 16QAM 系统顶层模块的 RTL 图，输入参数 carrier_clk 为载波时钟，reset_n 为复位信号，signal_clk 为输出串行数据的时钟，输出参数 in_data 为调制的数据，out_data 为解调出的数据。

3 仿真测试结果

3.1 数据源模块仿真

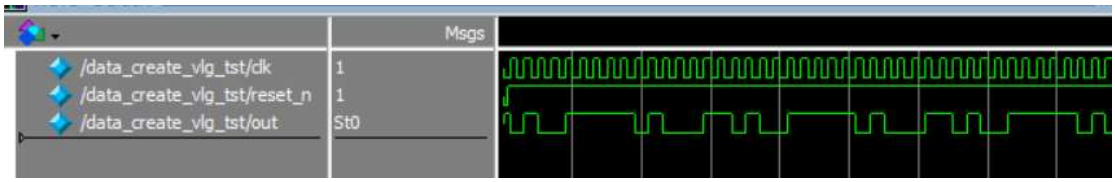

图 3.1 数据源模块仿真

图 3.1 为数据源模块的仿真图，可以很明显的看出，输出的串行数据是 1010_0111_1101_0001，与预期符合。

3.2 串并转换模块仿真

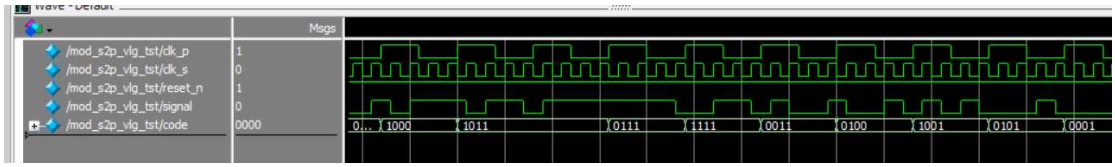

图 3.2 串并行模块仿真

图 3.2 为串并行模块的仿真时序图，可以很明显的看出并行输出就是由串行输出转换而来，而且慢一个并行时钟周期。

3.3 载波生成模块仿真

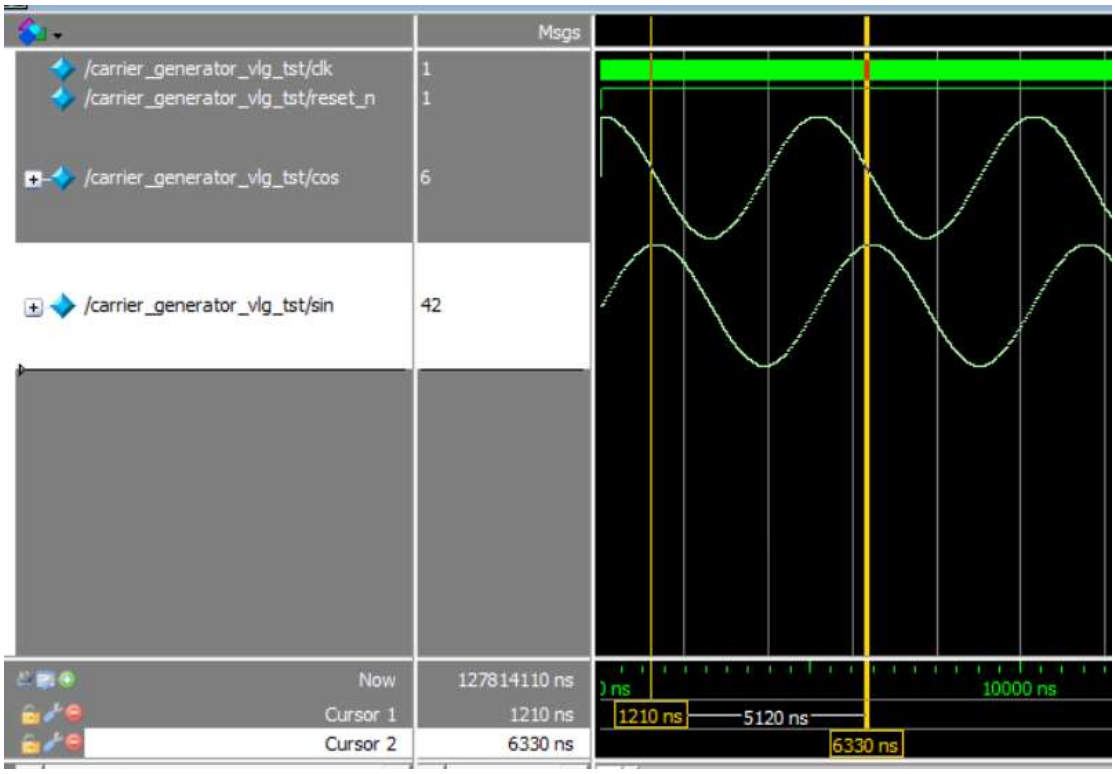

图 3.3 载波生成模块仿真图

图 3.3 为载波生成模块仿真图，模块输入时钟为 50MHz，生成了两路相互正交的正余弦载波信号。载波信号的周期为 5120ns，符合设计预期。

3.4 16QAM 调制模块仿真

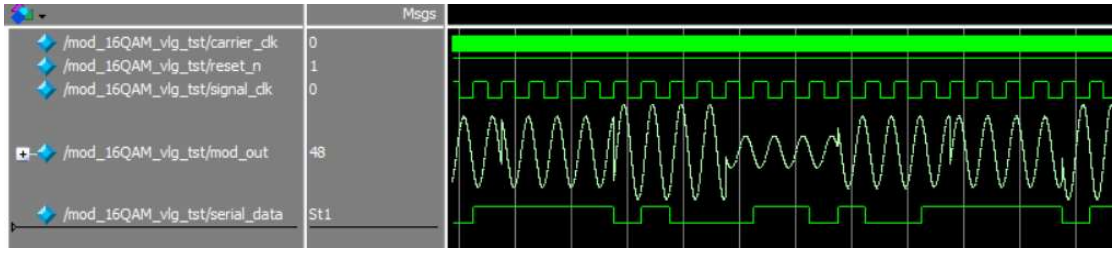

图 3.4 16QAM 调制模块仿真

图 3.4 为 16QAM 调制模块的仿真图，可以看到调制信号与预期一致，符合设计要求。

3.5 16QAM 解调模块仿真

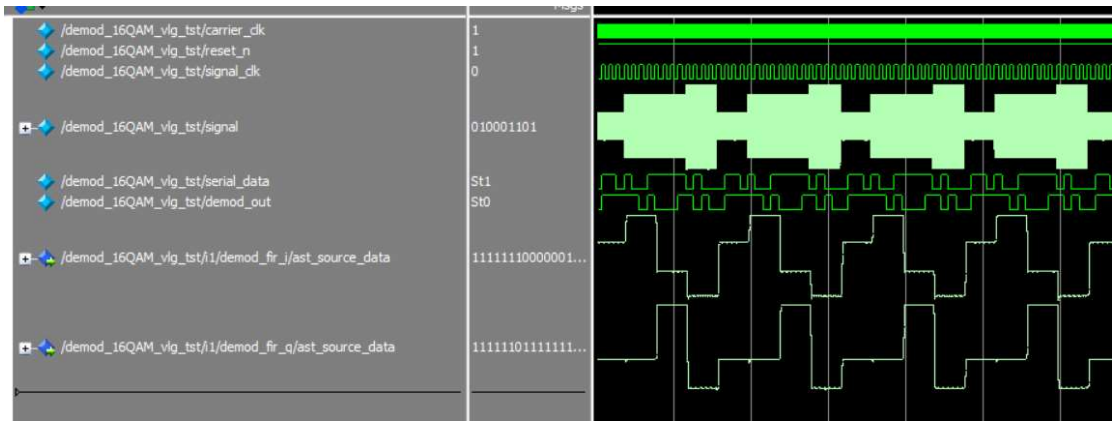

图 3.5 16QAM 解调模块仿真

图 3.5 为 16QAM 的解调仿真图，其中可以看到解调出来的数据与调制的数据一致，但是慢几个周期，也可以看到滤波器的输出与预期一致，解调符合预期。

3.6 16QAM 顶层模块仿真

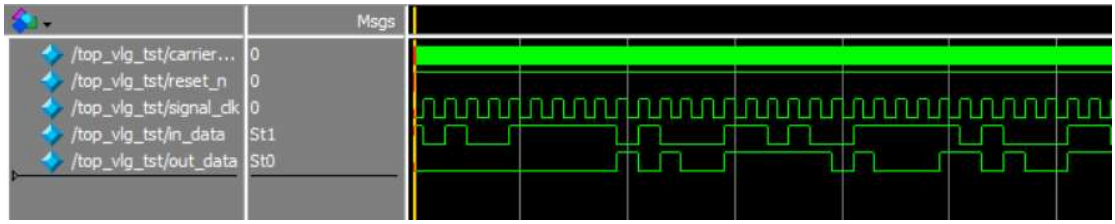

图 3.6 16QAM 顶层模块仿真

图 3.6 为 16QAM 顶层模块的仿真图，可以看到解调输出数据与调制的输入数据一致，但是慢几个信号周期，主要为中间的缓存影响，与预期的调试解调结果一致。

4 小结

本文实现了一种基于 FPGA 的 16QAM 调制解调系统的设计，使用 Verilog 与 Matlab 实现了 16QAM 的调制解调功能，经过验证，该设计能够正确完成对信号的调制解调，与预期结果相符，满足题目设计要求。

经过这次 16QAM 调制解调系统的设计，我对 Quartus II 和 Modelsim 有了更深的了解，能够熟练运用 Quartus II 设计模块、调用 IP 核，以及借助 Matlab 进行辅助设计。同时能够熟练 Quartus II 和 Modelsim 进行联合仿真。随着我查阅了更多的资料，我对于 16QAM 的调制解调有了更加深刻的了解。随着我对模块的编写调试，解决一个个 BUG，我对于 Verilog 进行设计有了更加深刻的理解。

本次课程设计不仅让我学会了关于 16QAM 调制解调系统的设计，也使我对 FPGA 的功能有了更清楚的认识，也激发了我对 FPGA 更加深入学习的兴趣。

参考文献

[1] 郭田耘，徐文波等.无线通信 FPGA 设计［M］.电子工业出版社，2004，10.

[2] 曹志刚，钱亚生．现代通信原理［M］．北京：清华大学出版社，2007.

[3] 戴振，李维英，蔡峰.基于 FPGA 的 16QAM 调制器设计与实现[J].电子元器件应用，2007(03):52-54.

[4] 彭国晋，文孟军.基于 FPGA 的 16QAM 调制解调系统的设计与实现[J]. 福建质量管理, 2018(14).

[5] 易阳，陈安，何雨东，等.基于 Quartus 的 16QAM 调制系统设计[J].现代信息科技，2019，003(002):P.39-41.

附录

16QAM 调制顶层模块代码

```
/*
* 16QAM 调制
*/
module mod_16QAM(signal_clk,carrier_clk,reset_n,mod_out,serial_data);
input signal_clk,carrier_clk,reset_n;
output signed [8:0] mod_out;
output wire serial_data; //串行数据

wire [3:0] p_data; //并行数据

//载波
wire signed [7:0] carrier_sin;
wire signed [7:0] carrier_cos;

//调制后载波
wire signed [7:0] carrier_i;
wire signed [7:0] carrier_q;
wire [1:0] signal_I; //I 信号
wire [1:0] signal_Q; //Q 信号

wire clk_4div; //四分频

assign signal_I = {p_data[3],p_data[1]};
assign signal_Q = {p_data[2],p_data[0]};
//载波输出
assign mod_out = carrier_i + carrier_q;

//四分频
freq_div #(.DIV(4)) u_div4(
    .orgin_clk(signal_clk),
    .reset_n(reset_n),
    .out_clk(clk_4div)
```

```
);

//数据源
data_create u_data_create(
    .clk(signal_clk),
    .reset_n(reset_n),
    .out(serial_data)
);

//串并转换
mod_s2p u_mod_s2p(
    .clk_s(signal_clk),
    .clk_p(clk_4div),
    .reset_n(reset_n),
    .signal(serial_data),
    .code(p_data)
);

// 载波生成
carrier_generator u_carrier(
    .clk(carrier_clk),
    .reset_n(reset_n),
    .sin(carrier_sin),
    .cos(carrier_cos)
);


//载波 I
mod_mul   mod_mul_i(
    .clk(carrier_clk),
    .signal(signal_I),
    .carrier(carrier_cos),
    .out(carrier_i)
);
```

```
//载波 Q
mod_mul mod_mul_q(
    .clk(carrier_clk),
    .signal(signal_Q),
    .carrier(carrier_sin),
    .out(carrier_q)
);

Endmodule
```

16QAM 解调顶层模块代码

```
/*
* 16QAM 解码
*/
module demod_16QAM(carrier_clk,orgin_clk,reset_n,signal,demod_out);
input carrier_clk,reset_n,orgin_clk;
input signed [8:0] signal; //调制输出
output demod_out;

wire [3:0] p_data; //并行数据


wire signed [7:0] mul_i;
wire signed [7:0] mul_q;

//载波
wire signed [7:0] carrier_sin;
wire signed [7:0] carrier_cos;


wire clk_4div; //四分频
wire clk_500k;

wire [1:0] i_data;
```

```
wire [1:0] q_data;

wire fir_q_vaild;
wire fir_i_vaild;

//滤波器
reg signed [19:0] fir_i;
reg signed [19:0] fir_q;
wire signed [19:0] fir_i_temp;
wire signed [19:0] fir_q_temp;

assign p_data = {i_data[1],q_data[1],i_data[0],q_data[0]};

//四分频
freq_div #(.DIV(4)) u_div4(
    .orgin_clk(orgin_clk),
    .reset_n(reset_n),
    .out_clk(clk_4div)
);

//100 分频
freq_div #(.DIV(100)) u_div100(
    .orgin_clk(carrier_clk),
    .reset_n(reset_n),
    .out_clk(clk_500k)
);

// 载波生成
carrier_generator u_carrier(
    .clk(carrier_clk),
    .reset_n(reset_n),
    .sin(carrier_sin),
    .cos(carrier_cos)
);
```

```
//乘法
demod_mul u_demod_mul(
    .clk(carrier_clk),
    .carrier_cos(carrier_cos),
    .carrier_sin(carrier_sin),
    .signal(signal),
    .out_i(mul_i),
    .out_q(mul_q)
    );

//i fir
demod_fir demod_fir_i(
    .clk(clk_500k),
   .reset_n(reset_n),
    .ast_sink_data(mul_i),
    .ast_sink_valid(1'b1),
    .ast_sink_error(2'b00),
    .ast_source_data(fir_i_temp),
    .ast_source_valid(fir_i_vaild),
    .ast_source_error()
);

always @(posedge carrier_clk or negedge reset_n) begin
    if(!reset_n) begin
        fir_i <= 20'b0;
    end else if (fir_i_vaild) begin
        fir_i <= fir_i_temp;
    end else begin
        fir_i <= fir_i;
    end
end

//q fir
demod_fir demod_fir_q(
    .clk(clk_500k),
```

```
    .reset_n(reset_n),
    .ast_sink_data(mul_q),
    .ast_sink_valid(1'b1),
    .ast_sink_error(2'b00),
    .ast_source_data(fir_q_temp),
    .ast_source_valid(fir_q_vaild),
    .ast_source_error()
);

always @(posedge carrier_clk or negedge reset_n) begin
    if(!reset_n) begin
        fir_q <= 20'b0;
    end else if (fir_q_vaild) begin
        fir_q <= fir_q_temp;
    end else begin
        fir_q <= fir_q;
    end
end

//i 判决
demod_dec demod_dec_i(
    .fir_data(fir_i),
    .reset_n(reset_n),
    .carrier_clk(carrier_clk),
    .signal_clk(orgin_clk),
    .signal(i_data)
);

//q 判决
demod_dec demod_dec_q(
    .fir_data(fir_q),
    .reset_n(reset_n),
    .carrier_clk(carrier_clk),
    .signal_clk(orgin_clk),
    .signal(q_data)
```

```
);

demod_p2s u_demod_p2s(
      .serial_clk(orgin_clk),
      .signal_clk(clk_4div),
      .reset_n(reset_n),
      .signal(p_data),
      .serial(demod_out)
);

endmodule
```

16QAM 调制测试仿真文件

```
`timescale 10 ns/ 10 ns
module mod_16QAM_vlg_tst();
// constants
// general purpose registers

// test vector input registers
reg carrier_clk;
reg reset_n;
reg signal_clk;
// wires
wire [8:0]    mod_out;
wire serial_data;

// assign statements (if any)
mod_16QAM i1 (
// port map - connection between master ports and signals/registers
      .carrier_clk(carrier_clk),
      .mod_out(mod_out),
      .serial_data(serial_data),
      .signal_clk(signal_clk),
      .reset_n(reset_n)
);
```

```
initial
begin
// code that executes only once
// insert code here --> begin

// --> end
      #0 reset_n = 1'b0;
     #0 carrier_clk = 1'b0;
      #0 signal_clk = 1'b0;
      #1 reset_n = 1'b1;
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
     #2 carrier_clk = ~carrier_clk; //25M

// --> end
end

always
begin
     #500 signal_clk =~signal_clk; //100k
end
endmodule
```

解调模块仿真

```
`timescale 10 ns/ 10 ns
module demod_16QAM_vlg_tst();
// constants
// general purpose registers

// test vector input registers
```

```
reg carrier_clk;
reg reset_n;

reg signal_clk;
// wires
wire [8:0] signal;
wire serial_data;

// assign statements (if any)
demod_16QAM i1 (
// port map - connection between master ports and signals/registers
      .carrier_clk(carrier_clk),
      .orgin_clk(signal_clk),
      .reset_n(reset_n),
      .demod_out(demod_out),
      .signal(signal)
);

mod_16QAM i2(
      .signal_clk(signal_clk),
      .carrier_clk(carrier_clk),
      .reset_n(reset_n),
      .serial_data(serial_data),
      .mod_out(signal)
);
initial
begin
// code that executes only once
// insert code here --> begin
     #0 reset_n=1'b0;
      #0 signal_clk=1'b0;
      #0 carrier_clk=1'b0;
      #2 reset_n = 1'b1;
// --> end
end
```

```
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
      #1 carrier_clk = ~carrier_clk;
// --> end
end

always
begin
     #5000 signal_clk =~signal_clk;
end
endmodule
```

本科生课程设计成绩评定表

姓　　名		学　　号	
专业、班级			

课程设计题目：

课程设计答辩或质疑记录：

成绩评定依据：

评　分　项　目	最高分限	实际得分
设计说明书（论文）文理通顺、书写工整、图纸整洁。	10	
方案设计合理，有一定创新，计算正确，论证充分。	20	
答辩时态度谦虚有礼。	10	
能够及时正确地回答教师所提出的问题。	30	
仿真设计及其运行情况	10	
设计报告规范性	20	
总分	100	

最终评定成绩（以优、良、中、及格、不及格评定）

指导教师签字：_____

年　　月　　日