TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN

THIẾT KẾ VLSI

Đề tài:

THIẾT KẾ BASEBAND ĐIỀU CHẾ QPSK TRÊN PHẦN CỨNG BẰNG NGÔN NGỮ VERILOG

Nhóm sinh viên thực hiện:

Họ tên	MSSV	Lớp
Nguyễn Minh Hiếu	20151336	Điện tử 3 K60
Nguyễn Minh Hiếu	20151337	Điện tử 3 K60
Nguyễn Duy Quang	20152956	Điện tử 6 K60
Phan Văn Hòa	20151599	Điện tử 4 K60

GVHD: TS. Phan Xuân Vũ

Hà Nội, 12/2019

LỜI NÓI ĐẦU

Trong học phần Kiến trúc máy tính, để phục vụ cho bài tập lớn cuối kì nhóm em được giao đề tài "Thiết kế Baseband điều chế QPSK trên phần cứng bằng ngôn ngữ Verilog" để thực hiện. Đề tài này tuy hơi khó nhưng khá hay, vận dụng được nhiều kiến thức đã học để làm. Nhóm em xin chân thành cảm ơn thầy Phan Xuân Vũ đã hướng dẫn chỉ dạy trên lớp giúp chúng em hoàn thành bài tập lớn này.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1. CƠ SỞ LÍ THUYẾT

1.1 Giới thiệu kĩ thuật điều chế số

Trong ngành Điện tử - Viễn thông, điều chế là quá trình thay đổi một hoặc nhiều tính chất của sóng tuần hoàn, với tín hiệu điều chế chứa thông tin cần truyền. Hầu hết các hệ thống vô tuyến trong thế kỉ XX đều sử dụng điều chế tần số (Frequency Modulation - FM) hoặc điều chế biên độ (Amplitude Modulation – AM) để phát sóng vô tuyến.

Mục đích của điều chế tương tự là truyền tín hiệu băng cơ sở (baseband), hoặc tín hiệu thông thấp, qua băng thông tương tự ở tần số khác. Đối với điều chế số, mục đích của phương pháp này là để truyền một luồng bit số qua kênh truyền tương tự. Các kĩ thuật điều chế tương tự và số đã tạo điều kiện cho ghép kênh phân chia tần số (Frequency Division Multiplexing – FDM), trong đó một số tín hiệu thông thấp được truyền đồng thời trên cùng một thiết bị vật lí dùng chung, sử sụng các kênh băng thông riêng biệt (các tần số sóng mang khác nhau).

Trong bài tập lớn này, mục đích của điều chế số baseband là để truyền một luồng bit số qua kênh baseband, điển hình là dây đồng không được lọc như bus nối tiếp hoặc mạng cục bộ (Local Area Network. – LAN).

Trong điều chế số, tín hiệu sóng mang tương tự được điều chế bằng tín hiệu rời rạc. Các phương pháp điều chế số có thể được xem như sự chuyển đổi số sang tương tự và ngược lại, giải điều chế được xem là chuyển đổi tương tự sang số. Dưới đây là các kĩ thuật điều chế số phổ biến dựa trên khóa (keying):

- Khóa dịch biên độ (Amplitude-Shift Keying – ASK): Các biên độ được sử dụng là hữu hạn
- Khóa dịch biên tần số (Frequency-Shift Keying – FSK): Các tần số được sử dụng là hữu hạn
- Khóa dịch pha (Phase-Shift Keying – PSK): Các pha được sử dụng là hữu hạn
- Điều chế biên độ cầu phương (Quadrature Amplitude Modulation – QAM): Các pha và biên độ sử dụng là hữu hạn, với tối thiểu 2 pha và 2 tần số

1.2 Kĩ thuật khóa dịch pha cầu phương (QPSK)

1.2.1 Giới thiệu kĩ thuật PSK

Trong kĩ thuật này, mỗi pha được gán với một mẫu bit nhị phân duy nhất, với số bit mỗi mẫu bằng nhau. Xét sóng mang thông tin có dạng:

$$f_0(t) = A\cos(\omega t + \varphi_0) = A\cos\varphi(t)$$

Sử dụng khoảng di tần θ(t), ta có tín hiệu được điều chế

$$f_{PSK}(t) = A\cos\big(\varphi(t) + \theta(t)\big) = A\cos\varphi(t)\cos\theta(t) - A\sin\varphi(t)\sin\theta(t)$$

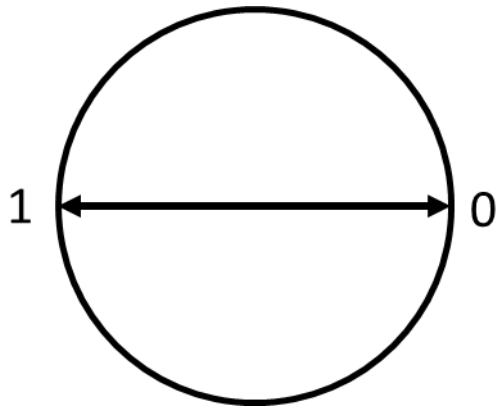
Đặt $s_1(t) = A\cos\theta(t)$, $s_2(t) = -A\sin\theta(t)$, ta có:

$$f_{PSK}(t) = s_1(t)\cos\varphi(t) + s_2(t)\sin\varphi(t)$$

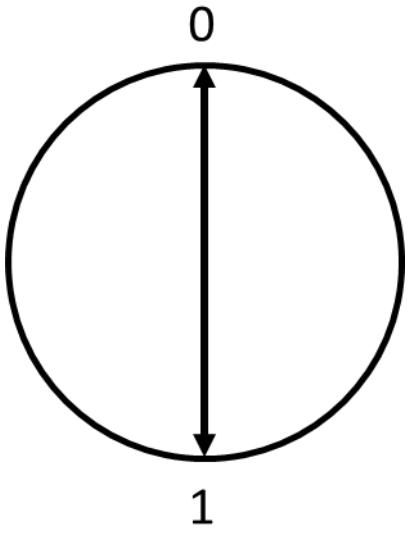
Như vậy, thay vì phát đi các tín hiệu (0,1) thì ta phát đi tín hiệu có thành phần là 2 hàm trực giao. Kĩ thuật PSK sử dụng N bit để mã hóa $M = 2^N$ được gọi là PSK-MP.

1.2.2 Kĩ thuật PSK-2P

Kĩ thuật này sử dụng 1 bit để mã hóa cho 2 pha. Có 2 kiểu điều chế như trên hình 1.1 và 1.2, tương ứng là các bảng mã hóa 1.1 và 1.2.



Hình 1.1 Điều chế PSK-2P (kiểu 1)



Hình 1.2 Điều chế PSK-2P (kiểu 2)

Bảng 1.1 Mã hóa PSK-2P (kiểu 1)

Bit mã hóa	θ(t)	$s_1(t)$	$s_2(t)$
0	0	A	0
1	π	-A	0

Bảng 1.2 Mã hóa PSK-2P (kiểu 2)

Bit mã hóa	θ(t)	$s_1(t)$	$s_2(t)$
0	π/2	0	A
1	3π/2	0	-A

Như vậy, với 2 kiểu điều chế PSK-2P, ta có tín hiệu đã được điều chế như sau:

- Kiểu 1:

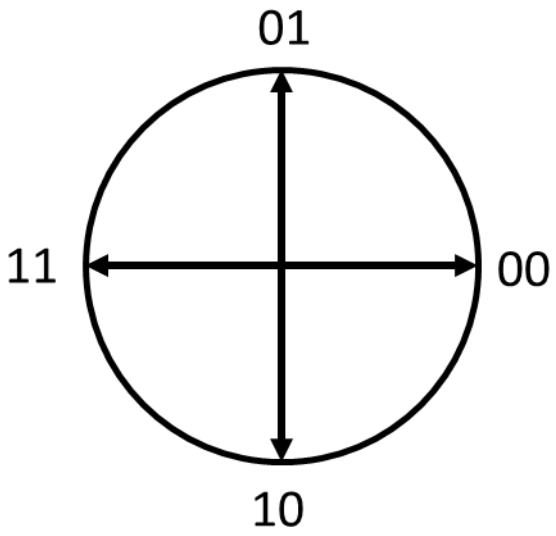
$$f_{PSK}(t) = \pm A\cos\varphi(t)$$

- Kiểu 2:

$$f_{PSK}(t) = \mp A\sin\varphi(t)$$

1.2.3 Kĩ thuật QPSK

PSK-4P hay QPSK (Quadrature PSK), sử dụng 2 bit để mã hóa cho 4 pha. Kĩ thuật này cũng có 2 kiểu điều chế nhưng trong bài tập lớn này nhóm em chỉ trình bày một kiểu trên hình 1.3, với điều kiện 2 mẫu mã hóa liên tiếp chỉ sai khác nhau 1 bit.



Hình 1.3 Điều chế QPSK

Bảng 1.3 là bảng mã hóa QPSK

Bảng 1.3 Mã hóa QPSK

Bit mã hóa	θ(t)	$s_1(t)$	$s_2(t)$
00	0	A	0
01	π/2	0	-A
11	π	-A	0
10	3π/2	0	A

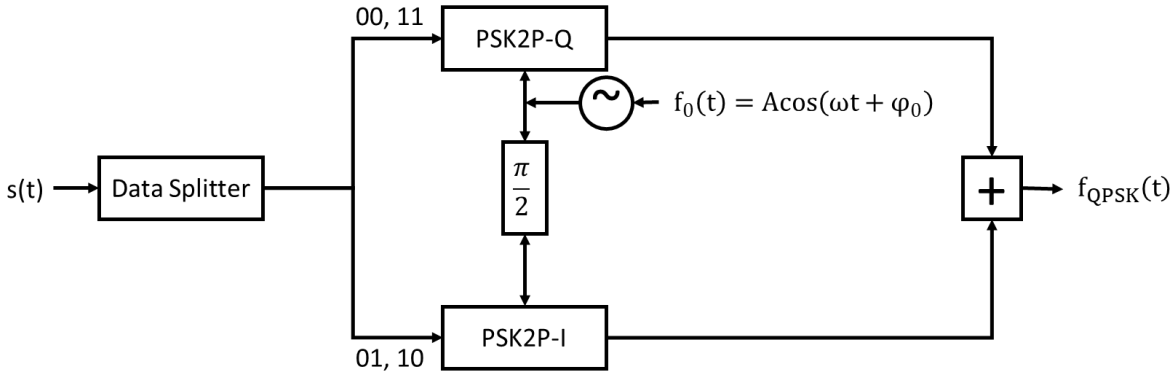
Như vậy ta có tín hiệu sau khi điều chế:

$$f_{QPSK}(t) = \pm A\cos\varphi(t) \mp A\sin\varphi(t)$$

Có thể thấy tín hiệu điều chế QPSK bằng tổng của 2 tín hiệu điều chế PSK, điều này có thể được áp dụng để xây dựng sơ đồ điều chế QPSK.

1.2.4 Sơ đồ điều chế QPSK

Sơ đồ điều chế QPSK được thể hiện trên hình 1.4.



Hình 1.4 Sơ đồ điều chế QPSK

Dòng bit đầu vào được đưa qua bộ phân tách dữ liệu, dữ liệu sau khi phân tách được đưa vào 2 bộ PSK2P-Q và PSK2P-I , kết hợp với bộ tạo dao động tạo ra sóng mang thông tin $f_0(t)$ để điều chế PSK-2P. Cuối cùng cộng tín hiệu từ 2 bộ PSK-2P ta thu được tín hiệu điều chế QPSK.

1.3 Truyền thông tin đi xa

Tín hiệu baseband sau khi điều chế có dải tần thấp, vì vậy không truyền được đi xa. Muốn truyền tín hiệu đi xa mà vẫn giữ được thông tin gốc ta cần sử dụng sóng mang (carrier wave). Sóng mang có biên độ bằng 1 nhưng tần số rất lớn. Giả sử ta có tín hiệu cần truyền đi là:

$$f_0(t) = A\cos(\omega_0 t)$$

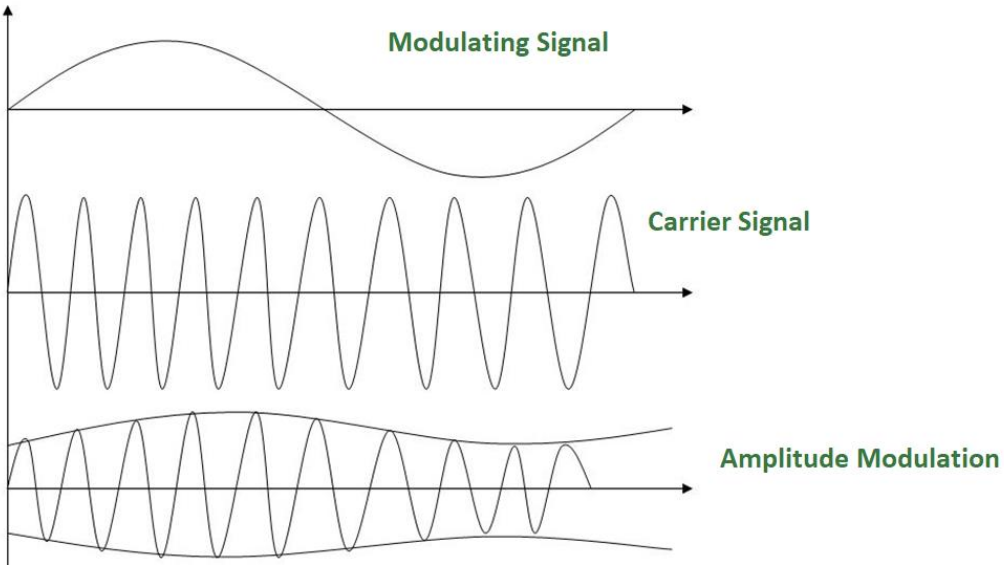
Tín hiệu sóng mang là:

$$f_c(t) = \cos(\omega_c t), \text{ với } \omega_c \gg \omega_0$$

Nhân tín hiệu cần truyền đi với sóng mang, ta có tín hiệu có thể truyền đi xa là:

$$f(t) = f_0(t)f_c(t) = A\cos(\omega_0 t)\cos(\omega_c t)$$

Tín hiệu này có biên độ bằng biên độ tín hiệu gốc nhưng tần số lớn hơn nhiều lần, đường bao của tín hiệu có dạng của tín hiệu gốc (hình 1.5). Phương pháp này còn được gọi là điều chế biên độ (Amplitude Modulation – AM).

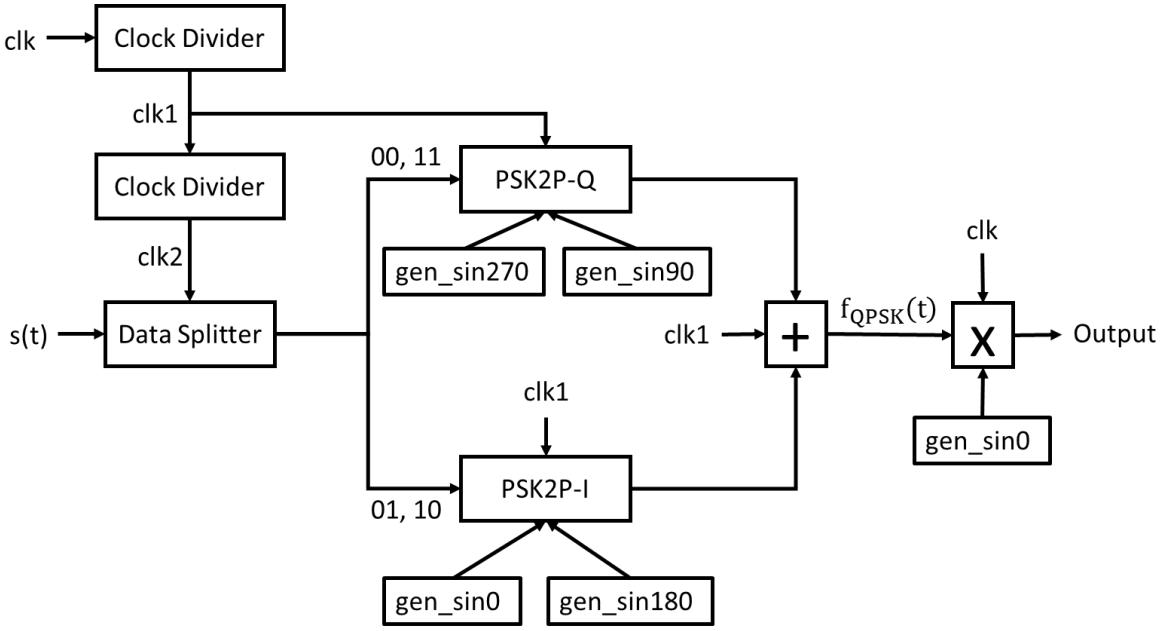


Hình 1.5 Điều chế AM để truyền thông tin đi xa

CHƯƠNG 2. THIẾT KẾ BASEBAND ĐIỀU CHẾ QPSK TRÊN PHẦN CỨNG

2.1 Thiết kế tổng quan

Dựa vào sơ đồ điều chế QPSK trên hình 1.4, ta có sơ đồ thiết kế tổng quan trên phần cứng (hình 2.1).



Hình 2.1 Sơ đồ thiết kế tổng quan trên phần cứng

Vì trên phần cứng không có các hàm sin, cos để tạo tín hiệu, vì vậy ta sử dụng các khối gen_sin0, gen_sin90, gen_sin180, gen_sin 270 để tạo ra các tín hiệu Asinφ(t), Acosφ(t), −A sinφ(t), −A cosφ(t) bằng cách sử dụng mảng có lưu sẵn các giá trị, Mỗi mảng gồm 64 phần tử có giá trị từ 0 đến 200 lưu giá trị các điểm của hàm sin, cos. Mỗi bộ chia tần số giảm tần số xuống 64 lần để đồng bộ các khối với nhau. Dựa theo sơ đồ thì tần số sóng mang gấp 64 lần tần số tín hiệu. Trên thực tế thì số lần này lớn hơn nhiều nhưng để dễ quan sát khi mô phỏng và tận dụng được khối chia tần số thì nhóm em chỉ thiết kế như vậy.

2.2 Khối tách dữ liệu

Khối này tách luồng bit đầu vào thành các cặp (00, 11), (01,10) để đưa vào 2 bộ PSK2P-Q và PSK2P-I tương ứng. Thiết kế của khối này được thể hiện trên hình 2.2

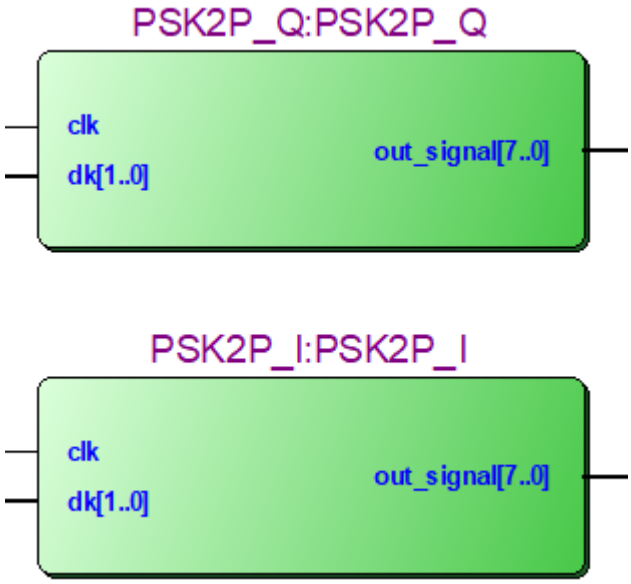
Hình 2.2 Khối tách dữ liệu

Đầu vào 2 bit được đưa đến đầu ra dk1 nếu có giá trị 00 hoặc 11, dùng để điều khiển bộ PSK2P-Q, hoặc đưa đến đầu ra dk2 nếu có giá trị 01 hoặc 10, dùng để điều khiển bộ PSK2P-I. Tốc độ đồng hồ của khối này là chậm nhất, cần phải qua 2 bộ chia tần số, vì khối này chỉ có nhiệm vụ đưa luồn bit đã tách vào các bộ điều chế, cần có thời gian để các bộ điều chế tính kết quả. Dưới đây là đoạn code Verilog thiết kế khối này.

```
module bitSplitter
	(
	input clk,
	input [1:0] in,

	output reg [1:0] dk1, dk2 //dk1: Q
				  //dk2: I
	);

	wire clk1;

	clk_divider clk_divider
	(
	.clk_in(clk),
	.clk_out(clk1)
	);

	always @(posedge clk1)
	begin
		if(in==2'b00 || in==2'b11)
		begin
			dk1=in;
			dk2=2'b00;
		end
		else
		begin
			dk1=2'b01;
			dk2=in;
		end
	end

endmodule
```

2.3 Các khối điều chế PSK-2P

Hai khối điều chế PSK2P-Q và PSK2P-I có thiết kế tương tự nhau, chỉ khác ở chỗ khối PSK2P-Q được thiết kế theo kiểu 2, còn khối PSK2P-I được thiết kế theo kiểu 1. Thiết kế 2 khối này được thể hiện trên hình 2.3.



Hình 2.3 Các khối điều chế PSK-2P

Đầu vào là tín hiệu điều khiển 2 bit dùng để xác định dạng sóng đầu ra dựa vào lí thuyết PSK-2P ở mục 1.2.2. Dưới đây là đoạn code Verilog thiết kế 2 khối này.

```
module PSK2P_Q
      #(
      parameter bit_length=8,
      parameter number_path=64
      )
   (
      input clk,
      input [1:0] dk,
      output [7:0] out_signal
      );

      wire [7:0] out90, out270;
      reg [7:0] out_next, out_reg;

      always @(posedge clk)
      begin
                  out_reg <= out_next;
      end

      always @*
      begin
            out_next=out_reg;
            case(dk)
            2'b00:
```

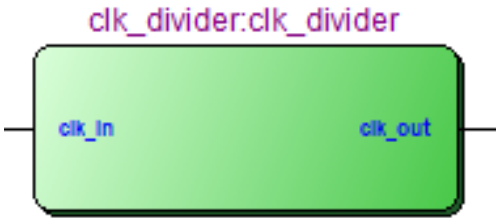
```
                out_next=out90;
            2'b11:
                out_next=out270;
            default:
                out_next=8'b00000000;
            endcase
      end

      gen_sin90
      #(
      .bit_length(bit_length),
      .number_path(number_path)
      )
      sin90
      (
      .clk(clk),
      .dk(dk),
      .out_sin(out90)
      );

      gen_sin270
      #(
      .bit_length(bit_length),
      .number_path(number_path)
      )
      sin270
      (
      .clk(clk),
      .dk(dk),
      .out_sin(out270)
      );

      assign out_signal=out_reg;

endmodule
```

```
module PSK2P_I
      #(
      parameter bit_length=8,
      parameter number_path=64
      )
   (
      input clk,
      input [1:0] dk,
      output [7:0] out_signal
      );

      wire [7:0] out0, out180;
      reg [7:0] out_reg, out_next;

      always @(posedge clk)
      begin
                  out_reg <= out_next;
      end

      always @*
      begin
            out_next=out_reg;
            case(dk)
            2'b01:
                  out_next=out180;
```

```
            2'b10:
                  out_next=out0;
            default:
                  out_next=8'b00000000;
            endcase
      end

      gen_sin180
      #(
      .bit_length(bit_length),
      .number_path(number_path)
      )
      sin180
      (
      .clk(clk),
      .dk(dk),
      .out_sin(out180)
      );

      gen_sin0
      #(
      .bit_length(bit_length),
      .number_path(number_path)
      )
      sin0
      (
      .clk(clk),
      .dk(dk),
      .out_sin(out0)
      );

      assign out_signal=out_reg;

endmodule
```

2.4 Khối chia tần số

Để đồng bộ hoạt động của các khối với nhau, khối chia tần số được sử dụng để giảm tốc độ đồng hồ đi 64 lần. Thiết kế khối chia tần số được thể hiện trên hình 2.4.



Hình 2.4 Khối chia tần số

Phía dưới là code Verilog thiết kế khối này.

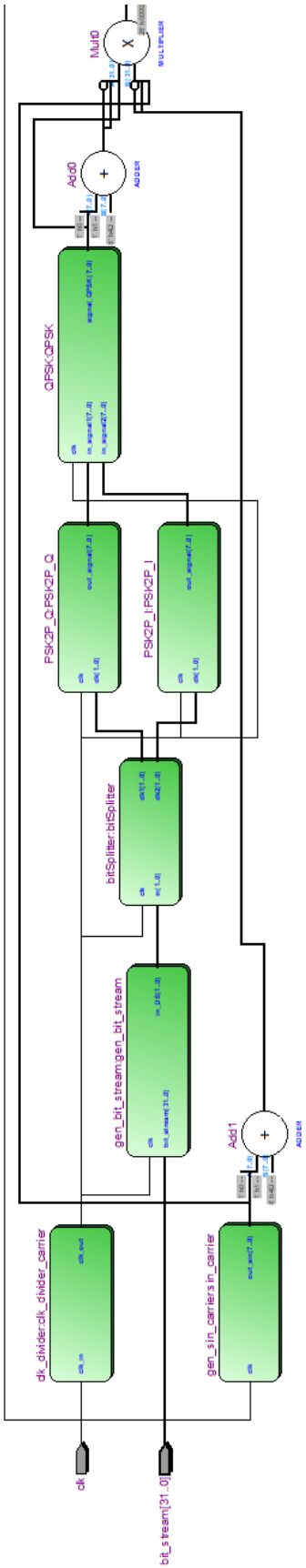
```
module clk_divider
      (
      input clk_in,
      output clk_out
      );

      integer count=0;
      reg t=0;

      always @(posedge clk_in)
      begin
            count=count+1;
            if(count==32)
            begin
                  t=~t;
                  count=0;
            end
      end

      assign clk_out=t;

endmodule
```

2.5 Baseband điều chế QPSK hoàn chỉnh

Hai tín hiệu từ hai khối PSK-2P được cộng lại với nhau, sau đó nhân với sóng mang có tần số lớn hơn 64 lần, ta thu được tín hiệu có thể truyền đi xa. Ghép các khối đã thiết kế lại với nhau, ta có Baseband điều chế QPSK hoàn chỉnh. Dưới đây là đoạn code Verilog ghép các module và điều chế AM. Cuối cùng ta có thiết kế hoàn chỉnh trên hình 2.5.

```
module Baseband_QPSK
      (
      input clk,
      input [31:0] bit_stream,

      output [15:0] Baseband_out
      );

      wire [1:0] in_DS, dk1, dk2;
      wire [7:0] signal_Q, signal_I, signal_QPSK, signal_carrier;
      reg [15:0] bb_out_reg, bb_out_next;

      clk_divider clk_divider_carrier
      (
      .clk_in(clk),
      .clk_out(clk_QPSK)
      );


      gen_bit_stream gen_bit_stream
      (
      .clk(clk_QPSK),
      .bit_stream(bit_stream),
      .in_DS(in_DS)
      );

      bitSplitter bitSplitter
      (
      .clk(clk_QPSK),
      .in(in_DS),
      .dk1(dk1),
      .dk2(dk2)
      );

      PSK2P_Q
      #(
      .bit_length(8),
      .number_path(64)
      )
      PSK2P_Q
   (
      .clk(clk_QPSK),
      .dk(dk1),
      .out_signal(signal_Q)
      );

      PSK2P_I
      #(
      .bit_length(8),
      .number_path(64)
```

```
      )
      PSK2P_I
   (
      .clk(clk_QPSK),
      .dk(dk2),
      .out_signal(signal_I)
      );

      QPSK QPSK
      (
      .clk(clk_QPSK),
      .in_signal1(signal_Q),
      .in_signal2(signal_I),
      .signal_QPSK(signal_QPSK)
      );

      gen_sin_carrier
      #(
      .bit_length(8),
      .number_path(64)
      )
      sin_carrier
      (
      .clk(clk),
      .out_sin(signal_carrier)
      );

      always @(posedge clk)
      begin
                  bb_out_reg <= bb_out_next;
      end

      always @*
      begin
            bb_out_next = (signal_QPSK-100)*(signal_carrier-100)+10000;
      end

      assign Baseband_QPSK_out=bb_out_reg;

endmodule
```



Hình 2.5 Baseband điều chế QPSK hoàn chỉnh

CHƯƠNG 3. MÔ PHỎNG BASEBAND ĐIỀU CHẾ QPSK

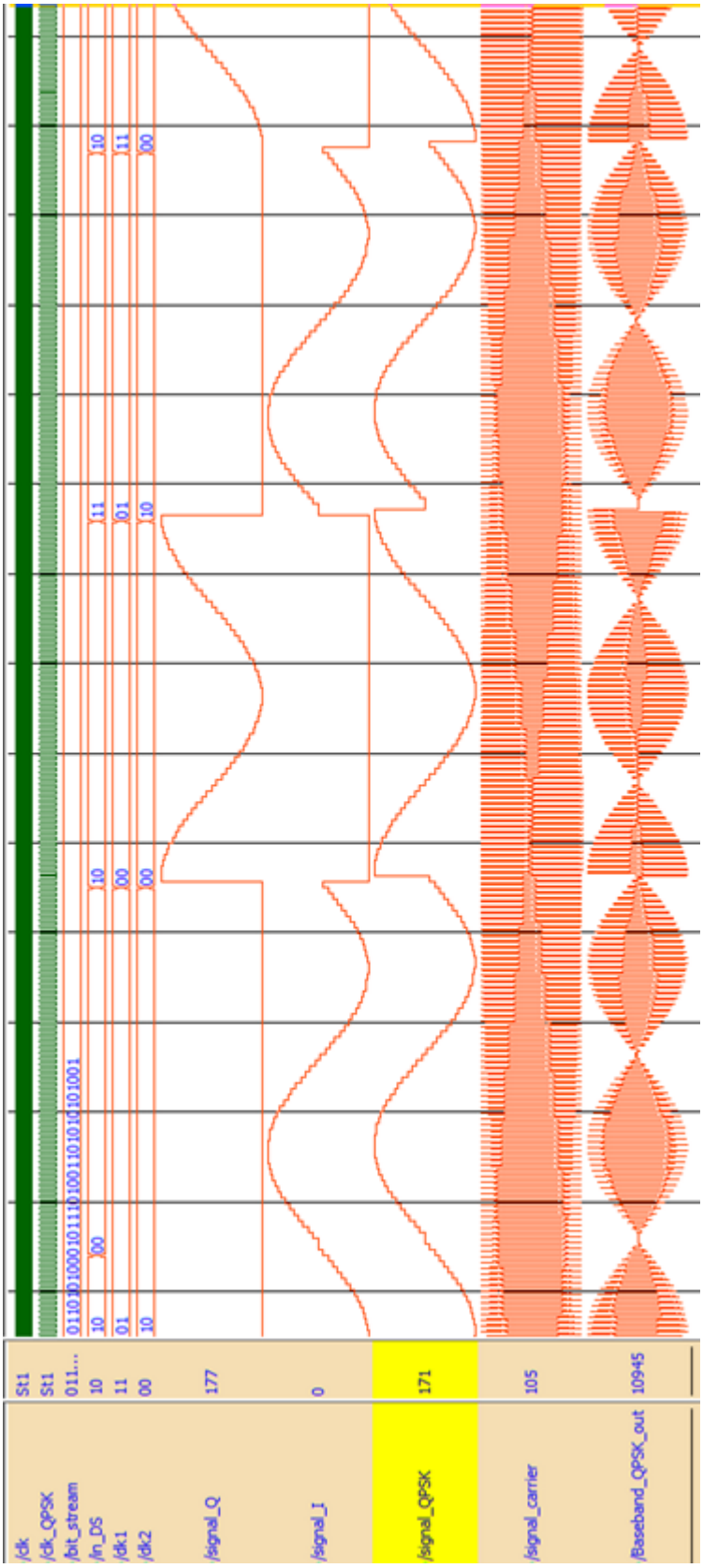
3.1 Thiết kế testbench

Trong thiết kế testbench, nhóm em tạo xung đồng hồ chu kì 1 μs, tương ứng với tần số 1 MHZ, cùng với đó là tạo một dòng bit đầu vào gồm 32 bits để kiểm tra hoạt động của hệ thống. Dưới đây là code Verilog file testbench.

```
`timescale 100ms/1us

module Baseband_QPSK_tb();
      reg  clk;
      reg [31:0] bit_stream;
      wire [15:0] Baseband_QPSK_out;

      Baseband_QPSK Baseband_QPSK_tb
      (
      .clk(clk),
      .bit_stream(bit_stream),
      .Baseband_out(Baseband_QPSK_out)
      );

      initial
      begin
            clk = 1'b0;
            forever #5000 clk = ~clk;
      end

      initial
      begin
            bit_stream=32'b01101010001011101001101010101001;
  end

endmodule
```

3.2 Kết quả mô phỏng

Kết quả mô phỏng được thể hiện trên hình 3.1



Hình 3.1 Kết quả mô phỏng Baseband điều chế QPSK

Dòng 32 bits được đưa vào hệ thống, sau đó từng cặp 2 bits liên tiếp được tách ra và trở thành các tín hiệu điều khiển dk1, dk2 theo thiết kế của bộ tách dữ liệu ở mục 2.2. Các tín hiệu signal_Q và signal_I là các tín hiệu điều chế từ các bộ điều chế PSK2P-Q và PSK2P-I tương ứng. Tín hiệu signal_QPSK là tổng của 2 tín hiệu điều chế trên. Tín hiệu signal_carrier là sóng mang có tần số rất lớn. Sau khi nhân tín hiệu điều chế QPSK với sóng mang, ta thu được tín hiệu Baseband_QPSK_ out. Các tín hiệu trong mô phỏng đều có kết quả đúng như lí thuyết.

KẾT LUẬN

Như vậy em đã thực hiện thành công việc thiết kế, mô phỏng Baseband điều chế QPSK trên phần cứng sử dụng ngôn ngữ Verilog. Việc thiết kế trên phần cứng thực sự gặp nhiều khó khăn do phải biểu diễn tín hiệu tương tự dưới dạng số và trên phần cứng không hỗ trợ các hàm lượng giác. Đây là một bài tập tương đối phức tạp, đòi hỏi nhiều thời gian, công sức để nghiên cứu thực hiện. Qua bài tập lớn nay đã giúp nhóm em hiểu sâu hơn về code Verilog, ứng dụng được lí thuyết thông tin số vào triển khai hệ thống tín hiệu tương tự trên phần cứng. Một lần nữa nhóm em xin cảm ơn thầy Phan Xuân Vũ đã hướng dẫn tận tình nhóm em hoàn thành bài tập lớn này

TÀI LIỆU THAM KHẢO

[1] Ian Glover and Peter Grant, *Digital Communications*, Prentice Hall, 2000

[2] Pong P. Chu, *FPGA Protyping by Verilog Examples*, John Wiley & Sons Inc, 2008

[3] https://en.wikipedia.org/wiki/Modulation, truy cập lần cuối ngày 22/12/2019

PHỤ LỤC

Toàn bộ mã nguồn của bài tập lớn này nằm trong đường link sau:

https://drive.google.com/open?id=1QOpb2LUG3WlinEGYF3RAbuQebYT6nRyJ