BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG

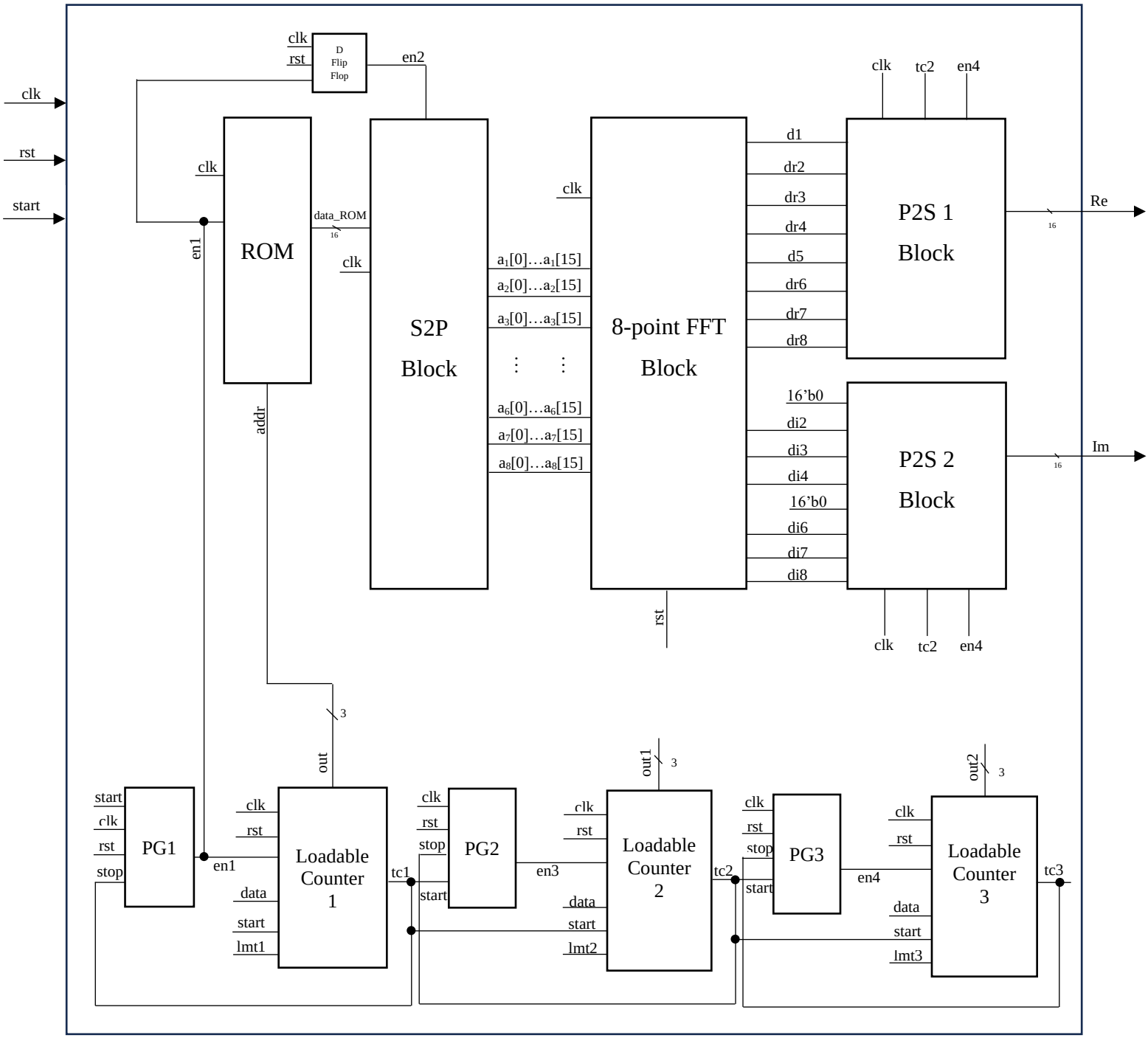
ĐỀ TÀI: THIẾT KẾ BỘ TÍNH TOÁN 8-POINT FFT

Môn	:	Thiết kế hệ thống & VMTH
Giáo viên hướng dẫn	:	TS. Võ Minh Huân
Lớp	:	ICSD336764_23_2_09
Nhóm sinh viên thực hiện	:	Nhóm 01 – Sáng thứ 5

- Phạm Quang Hợp 22119178
- Nguyễn Việt Anh 22119164
- Bùi Sỹ Vượng 22119159
- Nguyễn Trần Hùng Anh 22119163
- Lê Đình Thái Sơn 22119224

TP HỒ CHÍ MINH Tháng 04 năm 2024

1. TOP MODULE (FFT_processor)



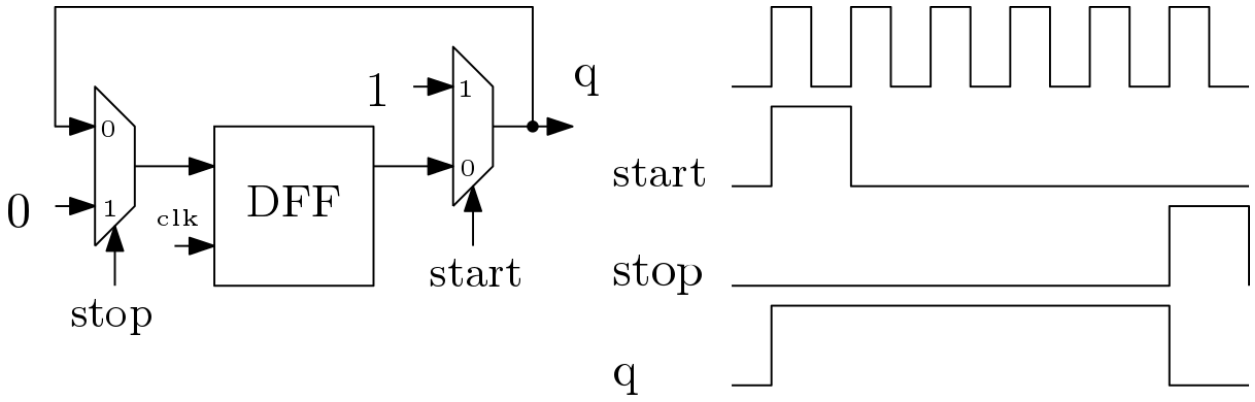
Bộ xử lý có các chân đầu vào là clk, reset và start. Bộ xử lý có hai vector đầu ra, một cho phần thực và một cho phần ảo. Độ rộng dữ liệu cố định 16 bits: 7 bits cho phần thực và 8 bits cho phần thập phân. MSB dùng cho dấu.

2. Tóm tắt mô tả hoạt động của toàn bộ hệ thống

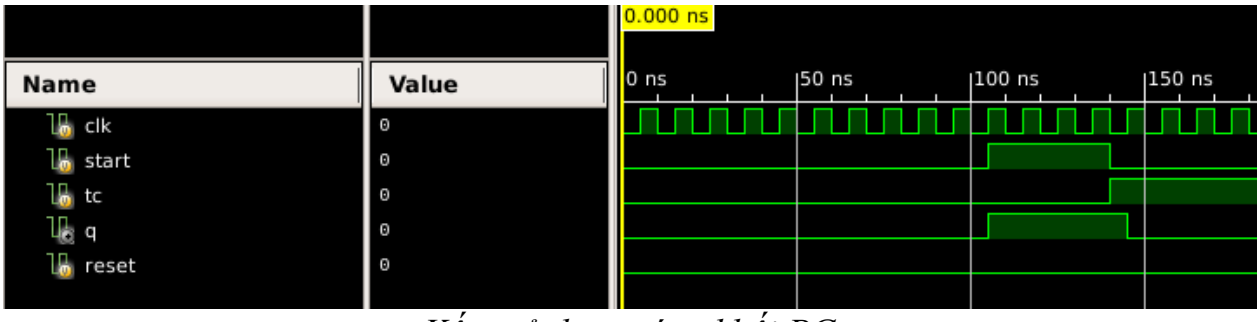
Xung start tạo ra tín hiệu en1 kích hoạt ROM và cũng đi đến khối S2P để chuyển từ tuần tự sang song song. Bộ đếm 1 cung cấp địa chỉ cho ROM. Bộ đếm 1 tạo ra tín hiệu tc1 sau 8 xung clock. Tín hiệu tc1 kích hoạt một khối PG2. PG2 và bộ đếm 2 được sử dụng để theo dõi độ trễ của khối FFT. Sau độ trễ của khối FFT, tín hiệu tc2 được tạo ra, làm kích hoạt bộ PG3 và bộ đếm 3. Tín hiệu tc2 và tín hiệu en4 hoạt động như tín hiệu load và p2s trong khối P2S ở trên. Tín hiệu tc2 được đồng bộ với đầu ra đầu tiên của khối FFT.

3. Khối Tạo Pha (PG)

Khối PG được vẽ dưới đây. Ngõ ra q mức cao khi một xung start kích vào. Đồng thời q mức thấp khi một xung stop được kích vào.



Input			Output
clk	start	stop	q
x	1	0	1
x	0	1	0
x	1	1	cấm

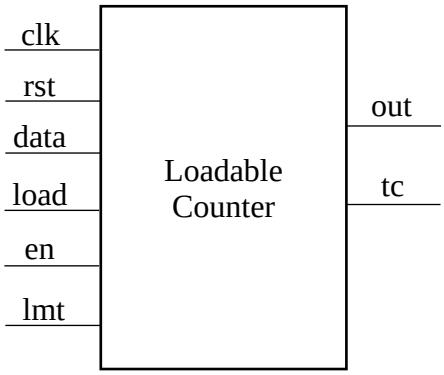


Kết quả dạng sóng khối PG

Nhận xét kết quả: đã đạt được mục tiêu thiết kế như trên lý thuyết đã đề cập.

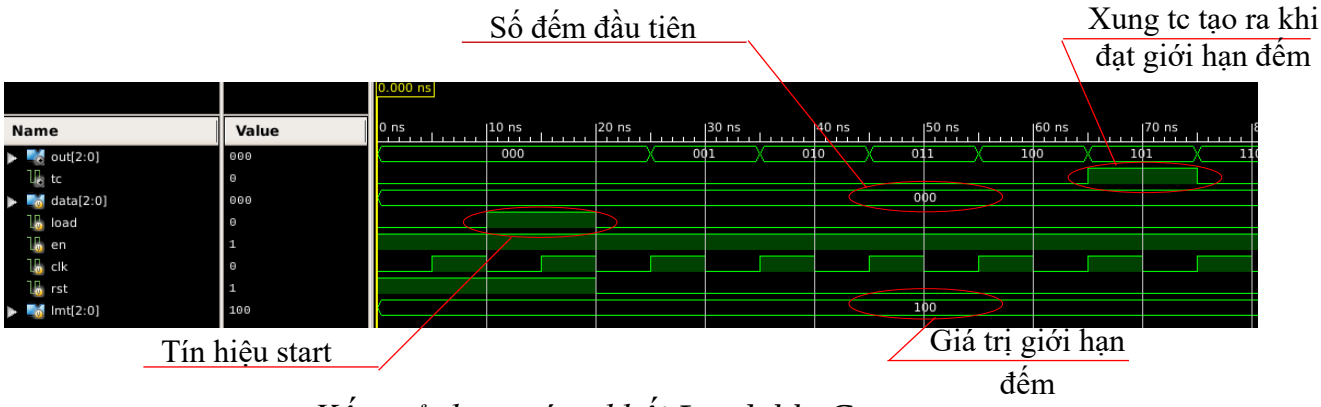
4. Bộ Đếm Có Thể Tải (Loadable Counter)

Chức năng: Bộ đếm có thể tải tạo ra tín hiệu Terminal Count (tc) khi đạt đến giá trị giới hạn đếm cụ thể (lmt).



Input		Output	
clk	Tạo xung clock	out	Ngõ ra làm địa chỉ cho khối ROM (ct1)
rst	Tạo xung reset		
data	Khởi tạo số đếm đầu tiên	tc	Làm xung start cho bộ PG kế tiếp, xung stop cho PG của nó
en	Kích xung, thực hiện đếm lên		
lmt	Giới hạn đếm, khi đạt đến bộ đếm tạo 1 xung tc		
load	Xung kích hoạt bộ đếm		

Input						Output	
clk	rst	load	data [2:0]	en	lmt [2:0]	out [2:0]	tc
0	0	1	000	1	100	000	0
1	0	0	000	1	100	001	0
2	0	0	000	1	100	010	0
3	0	0	000	1	100	011	0
4	0	0	000	1	100	100	1



Kết quả dạng sóng khối Loadable Counter

Nhận xét kết quả: đã đạt được mục tiêu thiết kế

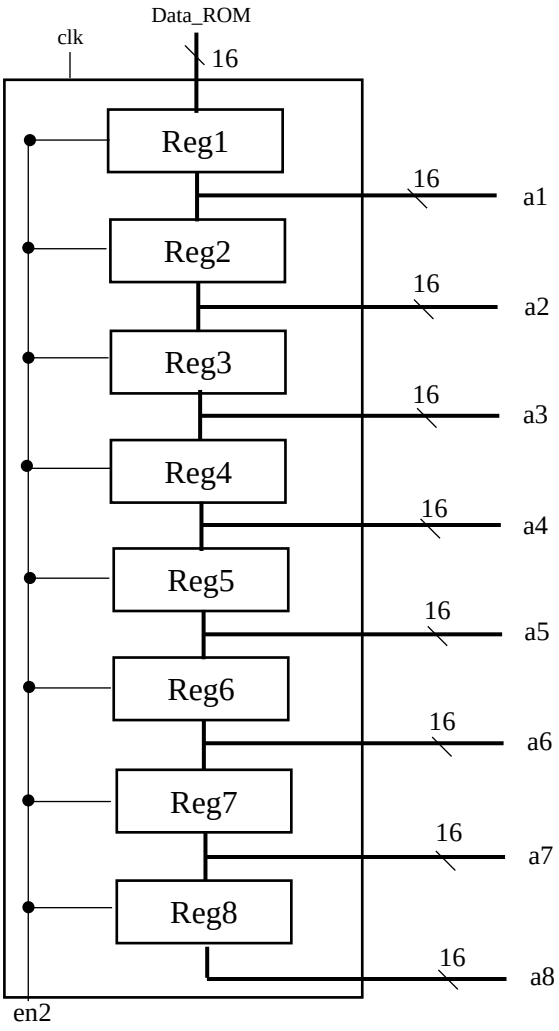
+ Bộ đếm đã thực hiện đếm với số đếm khởi tạo

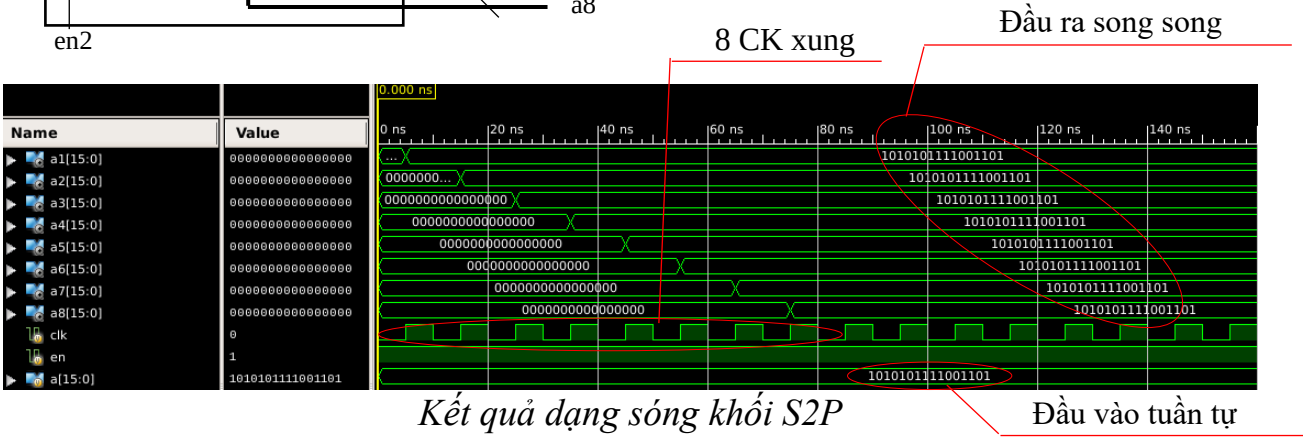
+ Bộ đếm đã tạo ra tín hiệu tc khi đạt đến giá trị ví dụ lmt = 3'b100

5. Khối S2P (chuyển đổi từ tuần tự sang song song)

Mô tả: Khối này có 8 thanh ghi độ rộng 16 bits. Mỗi thanh ghi được điều khiển bởi một chân kích hoạt điều khiển (en2). Dữ liệu đầu vào (data_ROM) chạy đến đầu ra chỉ khi tín hiệu kích hoạt (en2) mức cao.

Chức năng: chuyển đổi dữ liệu tuần tự thành song song trong 8 chu kỳ xung.

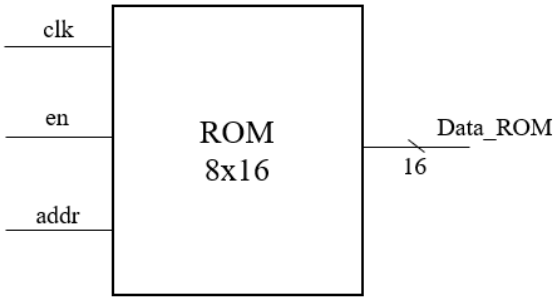




Kết quả dạng sóng khối S2P

Nhận xét kết quả: đạt được yêu cầu thiết kế chuyển đổi chuỗi dữ liệu 16 bits dạng tuần tự sang song song trong 8 chu kì xung. *Chú ý: trong testbench này chỉ có một chuỗi dữ liệu 16 bits, còn trong thiết kế có 8 chuỗi nên dạng tín hiệu đầu ra sẽ khác.*

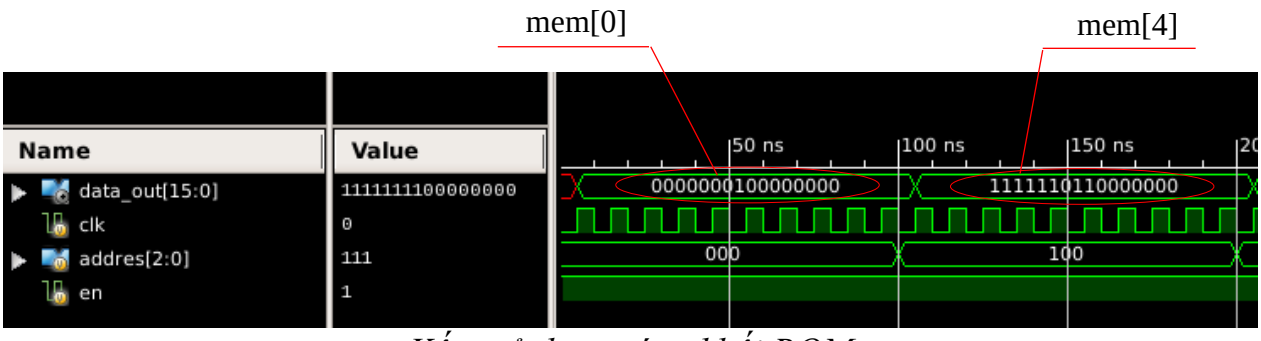
2. Khối ROM: cung cấp dữ liệu đầu vào cho thiết kế.




```
0000000100000000
0000000110000000
0000001000000000
0000001010000000
1111110110000000
1111111000000000
1111111010000000
1111111100000000
```

Mô tả In/Out:

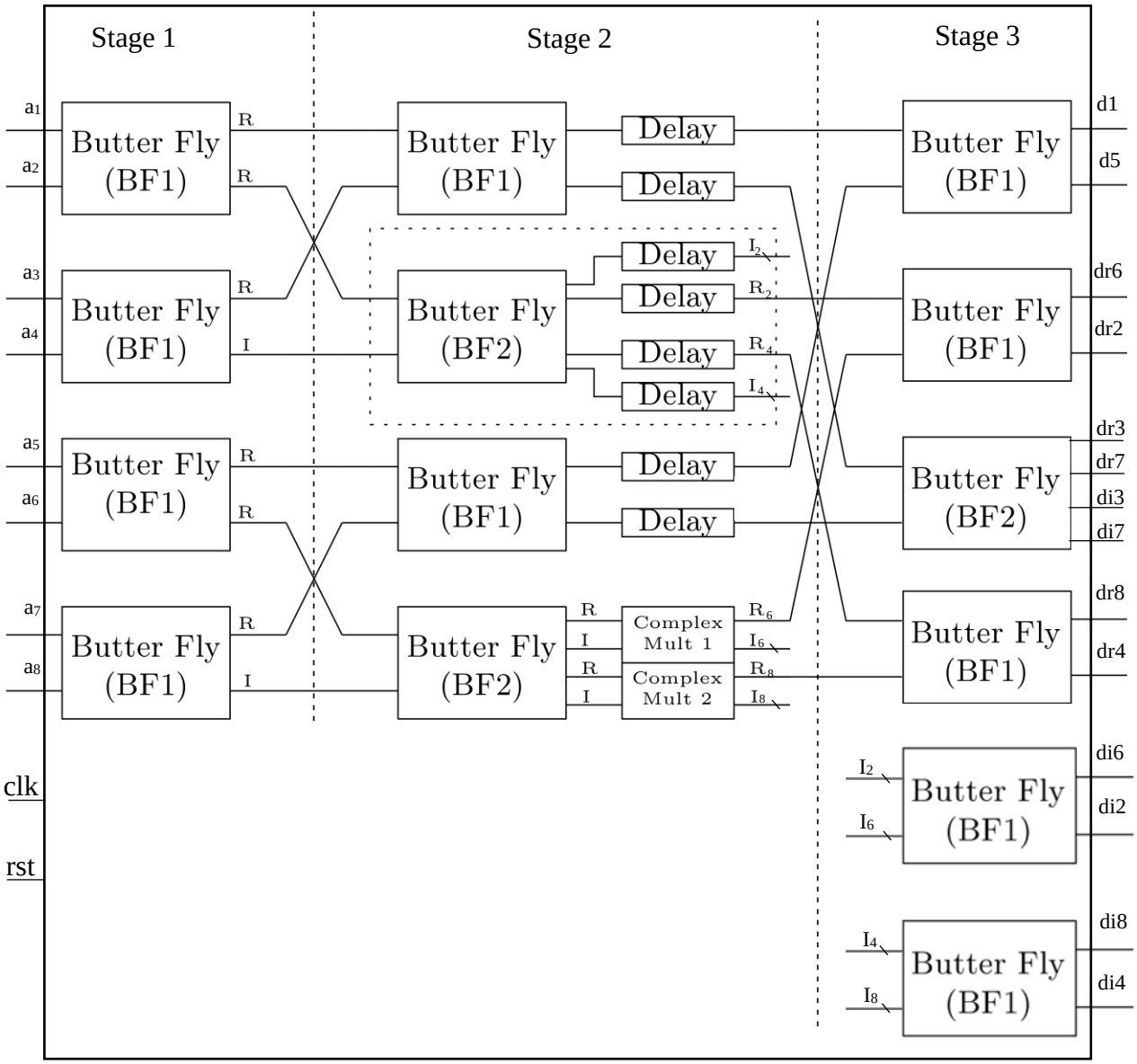
INPUT	OUTPUT
clk: tạo xung clock *en:* tín hiệu cho phép, sau mỗi en thì 1 chuỗi 16bit được lấy ra *addr:* cung cấp địa của chuỗi được lưu trong ROM	*Data_ROM:* chuỗi 16bit ngõ ra ứng với từng en và địa chỉ tương ứng



Kết quả dạng sóng khối ROM

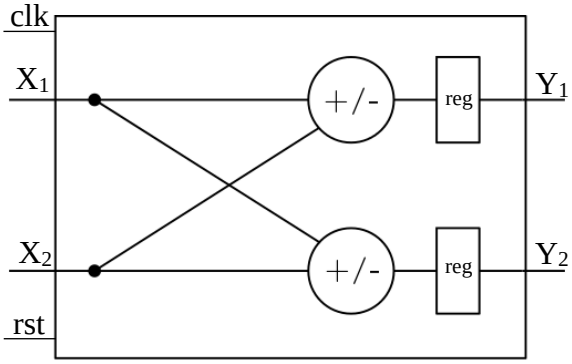
Nhận xét kết quả: đạt được yêu cầu thiết kế, ROM trích xuất dữ liệu đúng với các vị trí mem[0] và mem[4] như trên mảng dữ liệu mô tả.

7. Khối 8-point FFT

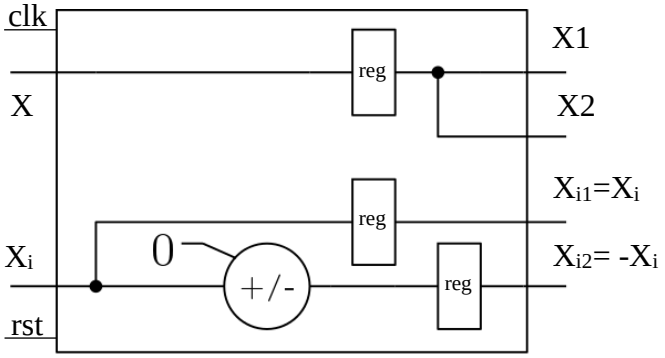


7.1. Các Khối Con Butterfly (BF1 và BF2)

<u>Mô tả chức năng</u>: Khối con butterfly cơ bản thực hiện hai loại phép toán cộng và trừ. Do đó, chúng em đã thiết kế khối BF1 thực hiện phép cộng hoặc phép trừ, **trong đó bộ (+/-) có tín hiệu chọn add: mức 0 thực hiện (+), mức 1 thực hiện (-)**. Còn khối BF2 ngoài tính toán phần real còn tính toán phần image, nhận 2 giá trị: real, image. Trả về 4 giá trị: real, real, image, -image.



Butter Fly (BF1)



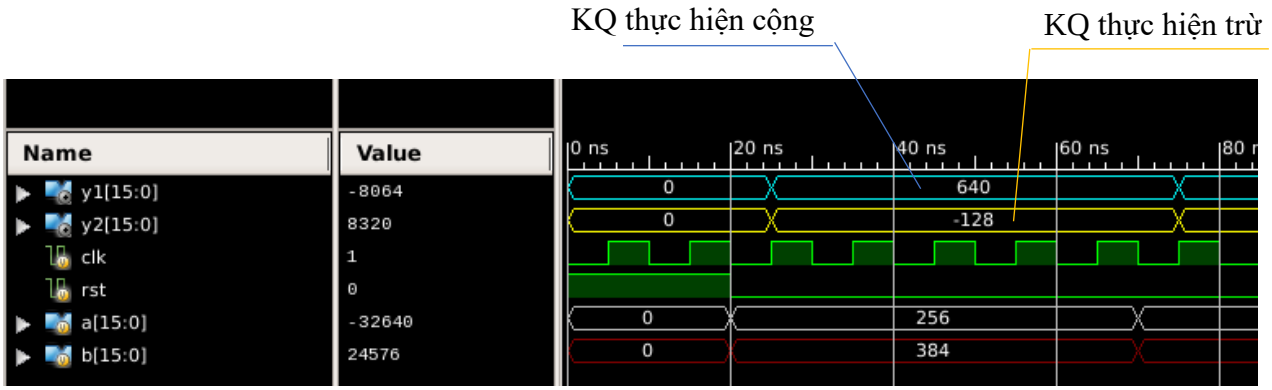
Butter Fly (BF 2)

Mô tả In/Out khối BF1:

Input				Output	
clk	rst	X_1 [15:0]	X_2 [15:0]	Y_1 **(add=0)** [15:0]	Y_2 **(add=1)** [15:0]
x	0	00000001 00000000	00000001 10000000	00000010 10000000	10000000 10000000
x	0	10000000 10000000	01100000 00000000	01011111 10000000	11100000 10000000

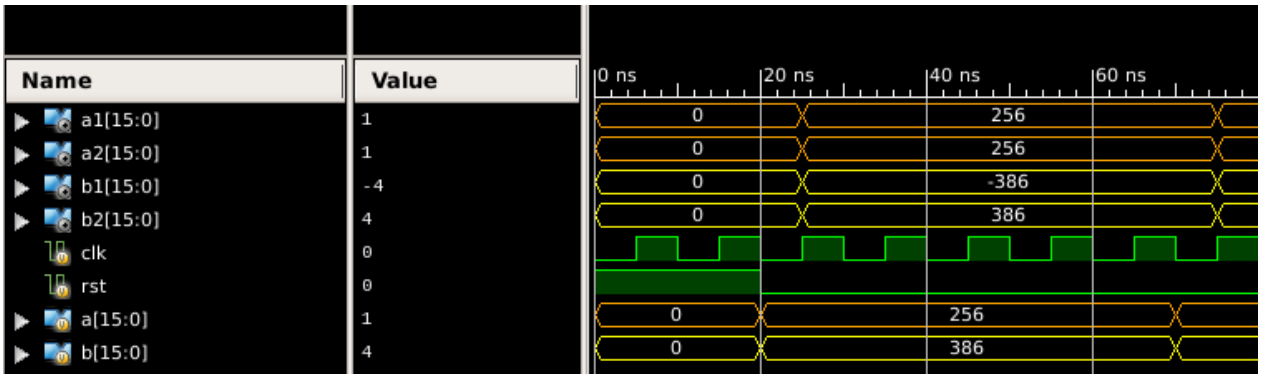
Mô tả In/Out khối BF2:

Input				Output			
clk	rst	X [15:0]	X_i [15:0]	X1 [15:0]	X2 [15:0]	X_{i1} [15:0]	X_{i2} [15:0] **(add=1)**
x	0	00000001 00000000	00000001 10000010	00000001 00000000	00000001 00000000	00000001 10000010	10000001 10000010



Kết quả dạng sóng khối BF1

Nhận xét kết quả: đạt được yêu cầu thiết kế, thực hiện đúng phép cộng hoặc phép dựa trên tín hiệu chọn add đã trình bày ở trên.



Kết quả dạng sóng khối BF2

Nhận xét kết quả: đạt được yêu cầu thiết kế, nhận 2 giá trị đầu vào. Real là a = 256 trả về hai giá trị a1 và a2 giống nó. Image là b = 386 trả về b2 = b và b1 = - b.

Chú ý: Do phần mềm không hỗ trợ mô phỏng số thập phân nên các kết quả mô phỏng sẽ là các số nguyên (chỉ test phép tính đúng của các khối).

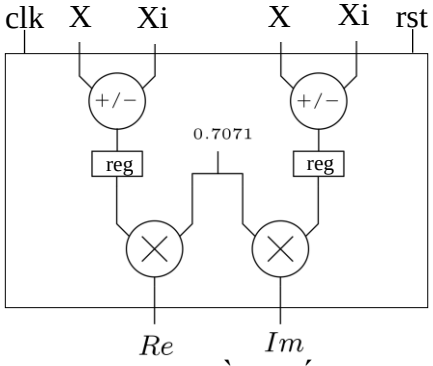
7.2. Bộ Nhân Số Phức (Complex Mult)

Chức năng: Bộ nhân số phức nhân hai số phức. Phép nhân hai số phức có thể được biểu diễn như sau:

$$(a+jb)\times(c+jd) = ac+jad+jbc-bd = (ac-bd)+j(ad+bc) \qquad (1)$$

Trong trường hợp của chúng em, vì (a+jb) là biến số nhưng (c+jd)=$\frac{\sqrt{2}}{2}(1-j)$ hoặc (c+jd)=$\frac{\sqrt{2}}{2}(-1-j)$ trong tính toán FFT. Do đó,

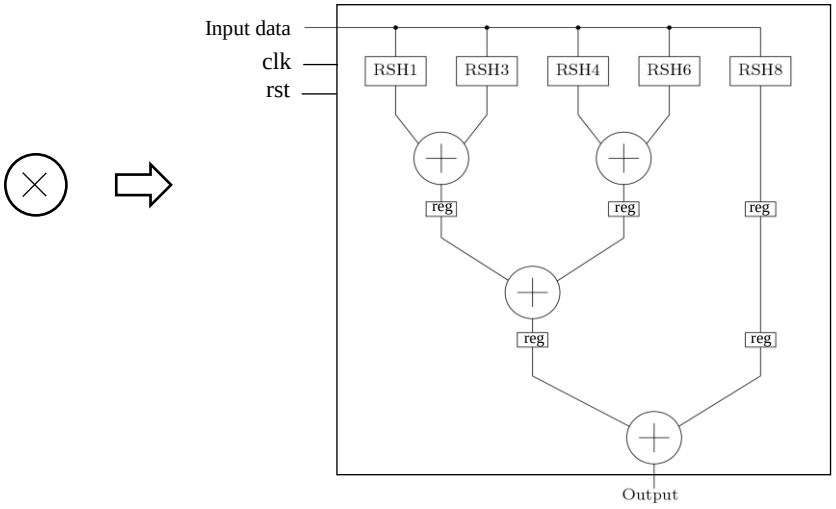
$$(1) \Leftrightarrow \begin{cases} (a+jb)(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j) = \frac{\sqrt{2}}{2}(a+b)-\frac{\sqrt{2}}{2}j(a-b) \\ (a+jb)(-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j) = -\frac{\sqrt{2}}{2}(a-b)-\frac{\sqrt{2}}{2}j(a+b) \end{cases}$$

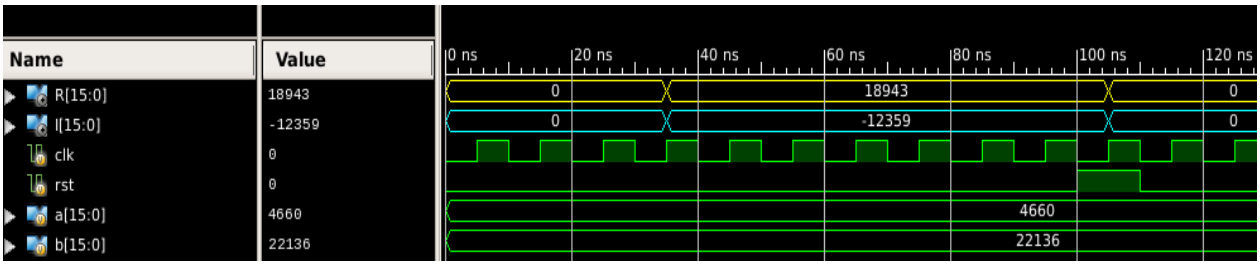


Trong bộ nhân hằng số, phép nhân với hằng số 0.7071 được biểu diễn như sau:

$$0.7071x=(2-1+2-3+2-4+2-6+2-8)x$$

Các khối RSH là các khối dịch input data sang phải 1 or 3 or 4 or 6 or 8 bits.





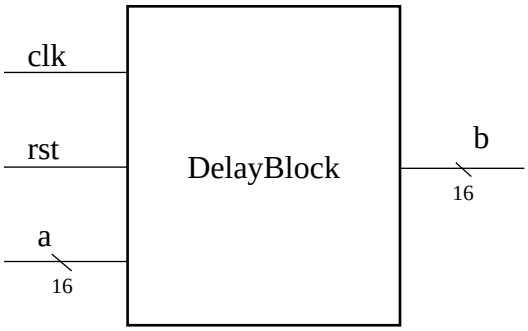
Kết quả dạng sóng khối Complex_Mult

Nhận xét kết quả: đã đạt được yêu cầu thiết kế, ví dụ với a = 4660, b = 22136

+ R = (a + b) x 0.7071 = (4660 + 22136) x 0.7071 = 18943

+ I = (a - b) x 0.7071 = (4660 - 22136) x 0.7071 = -12359

7.3. Khối Trễ (DelayBlock)

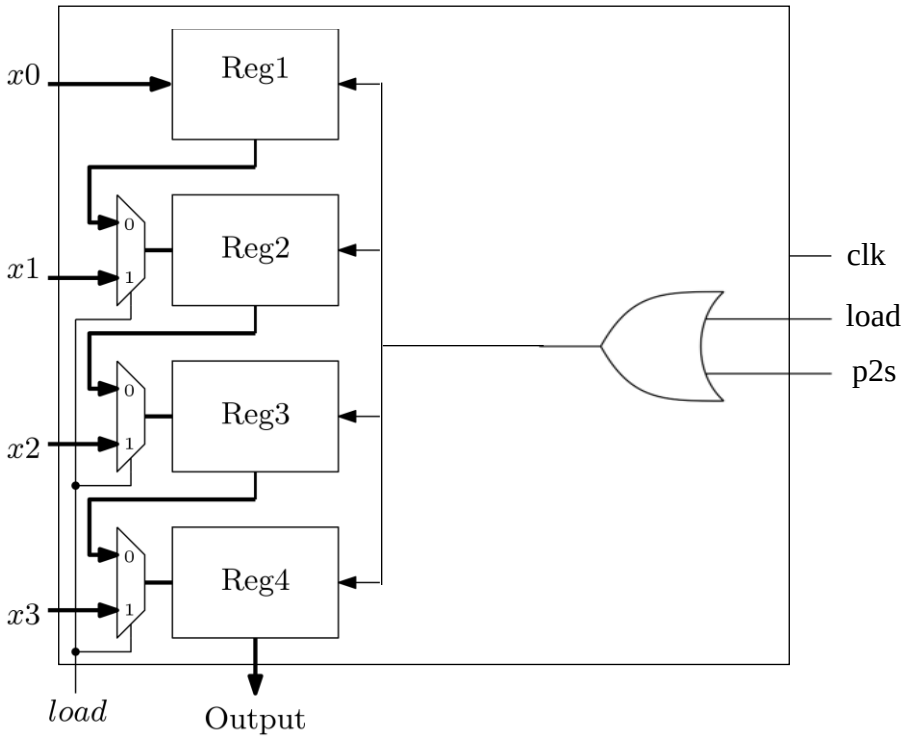


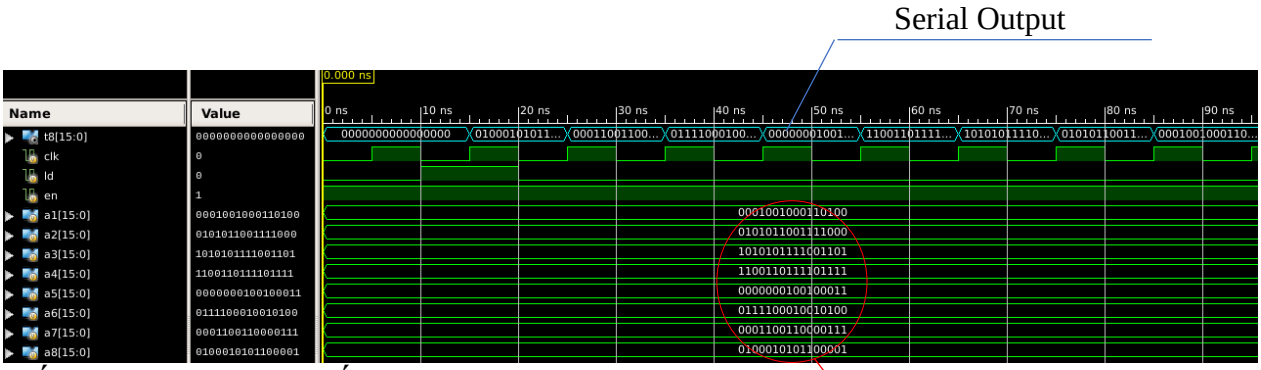
Chức năng: các khối trễ được thêm vào để đồng bộ hóa các hoạt động, 4 thanh ghi 16bits được kết nối tuần tự để tạo thành các khối trễ.

8. Khối P2S (chuyển đổi từ song song sang tuần tự)

Chức năng: chuyển đầu ra song song từ khối FFT thành dạng tuần tự do thiếu chân đầu ra hoặc nhiều hệ thống chấp nhận dữ liệu tuần tự.

Mô tả: Sơ đồ khối P2S được hiển thị dưới cho 4 mẫu dữ liệu, đối với báo cáo là 8 mẫu. Xung load trước tiên tải các mẫu dữ liệu song song vào các thanh ghi thông qua các Mux. Sau đó, tín hiệu p2s làm cho chúng trở thành dạng tuần tự.





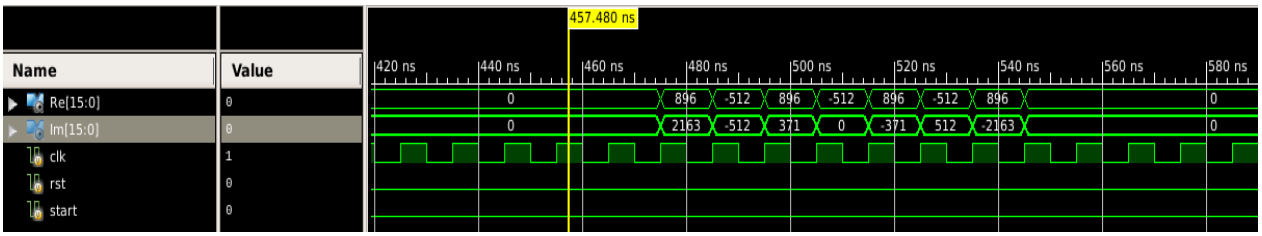
Kết quả dạng sóng khối P2S

Nhận xét kết quả: đạt được yêu cầu thiết kế chuyển đổi 8 chuỗi dữ liệu 16bit dạng song song thành dạng tuần tự ở ngõ ra khi có 1 xung load kích vào.

9. Testcase mô tả hệ thống

Để kiểm tra tính đúng đắn của một bộ tính toán FFT 8 điểm, chúng em đã thực hiện đưa mẫu dữ liệu [15:0] mem [0:7] sau đây vào hệ thống

```
0000000100000000
0000000110000000
0000001000000000
0000001010000000
1111110110000000
1111111000000000
1111111010000000
1111111100000000
```



Kết quả dạng sóng của hệ thống

Nhận xét kết quả: Thiết kế đã bảo đảm logic của một bộ FFT khi so sánh với kết quả của bộ FFT khác được đăng tải trên mạng với cùng số liệu đầu vào.

respectively. The tc2 signal is synced with the first output of FFT block.

FFT Processor Output and Design Performance

The output of the FFT processor is shown in Fig. 12 according to the example shown [1].

Input data samples are real but output is imaginary.

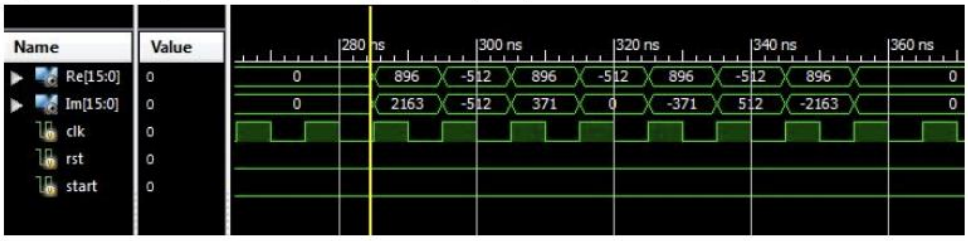


Fig.12: Simulation Result

The FFT processor is implemented on SPARTAN 3E starter kit. The design performance is shown below in the following table.

(kiểm chứng kết quả bọn em chưa so sánh với lý thuyết mà chỉ đề cập đến hình ảnh minh chứng trên vì Môn xử lí tín hiệu số chưa học đến phần FFT)

BẢNG PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ HOÀN THÀNH CỦA CÁC THÀNH VIÊN TRONG NHÓM

MSSV	Họ và tên	Nhiệm vụ	Kết quả	Ký tên
22119178	Phạm Quang Hợp (nhóm trưởng)	Bộ điều khiển (PG, bộ đếm), FFT_8point; stage 1-2-3; FFT_processor	Hoàn thành tốt, đúng hạn	
22119164	Nguyễn Việt Anh	Delayblock, Complex_mul	Hoàn thành tốt, đúng hạn	
22119259	Bùi Sỹ Vượng	Khối P2S (và các muxes 16bits)	Hoàn thành tốt, đúng hạn	
22119163	Nguyễn Trần Hùng Anh	Buterfly 1, 2 Khối ROM	Hoàn thành tốt, đúng hạn	
22119224	Lê Đình Thái Sơn	Khối S2P (và các register 16bits)	Hoàn thành tốt, đúng hạn	

PHỤ LỤC

Các code Synthesis và Testbench

FFT_processor (top module)

```
`timescale 1ns / 1ps
module FFT_processor(clk,rst,start,Re,Im);
input clk,rst,start;
output [15:0] Re,Im;
wire [15:0] data_out,a1,a2,a3,a4,a5,a6,a7,a8,
        d1,dr2,di2,dr3,di3,dr4,di4,d5,dr6,di6,dr7,di7,dr8,di8;
wire [2:0] addr,out1,out2;
wire en1,en2,en3,en4,tc1,tc2,tc3;
parameter zro = 16'b0;
ROM mem(clk,addr,data_out,en1);
S2P blk1(clk,en2,data_out,a8,a7,a6,a5,a4,a3,a2,a1);
FFT_8point blk2(clk,rst,a1,a2,a3,a4,a5,a6,a7,a8,
                d1,dr2,di2,dr3,di3,dr4,di4,d5,dr6,di6,dr7,di7,dr8,di8);
P2S blk3(clk,tc2,en4,d1,dr2,dr3,dr4,d5,dr6,dr7,dr8,Re);
P2S blk4(clk,tc2,en4,zro,di2,di3,di4,zro,di6,di7,di8,Im);
///control part.....
cnt4 ct1(addr,3'b000,start,en1,clk,rst,tc1,3'b110);
pg p1(start,tc1,en1,clk,rst);
DFF d11(en2,clk,rst,en1);
cnt4 ct2(out1,3'b000,tc1,en3,clk,rst,tc2,3'b111);
pg p2(tc1,tc2,en3,clk,rst);
cnt4 ct3(out2,3'b000,tc2,en4,clk,rst,tc3,3'b110);
pg p3(tc2,tc3,en4,clk,rst);
endmodule
```

KHỐI ROM

///**Synthesis…**

```
`timescale 1ns / 1ps
module ROM(clk,addres,data_out,en);
input clk,en;
input [2:0] addres;
output reg [15:0] data_out;
reg [15:0] mem [0:7];
initial begin
        $readmemb("input.txt", mem);
end
always@(posedge clk)
        begin
                if(en)
                        data_out <= mem[addres];
                else
                        data_out <= data_out;
                end
endmodule
```

///**Testbench…**

```
`timescale 1ns / 1ps

module ROM_tb;
        reg clk;
        reg [2:0] addres;
        reg en;
        wire [15:0] data_out;
```

```
        ROM uut (
                .clk(clk),
                .addres(addres),
                .data_out(data_out),
                .en(en)
        );
        always #5 clk = ~clk;
        initial begin
                clk = 0;
                addres = 0;
                en = 1;
                #100;
                addres = 4;
                #100;
                addres = 7;
        end
endmodule
```

KHỐI S2P

```
///Synthesis…
`timescale 1ns / 1ps
module S2P(clk,en,a,a1,a2,a3,a4,a5,a6,a7,a8);
input [15:0] a;
output [15:0] a1,a2,a3,a4,a5,a6,a7,a8;
input clk,en;
fdce16 f1(a,clk,en,a1);
fdce16 f2(a1,clk,en,a2);
fdce16 f3(a2,clk,en,a3);
```

```
fdce16 f4(a3,clk,en,a4);
fdce16 f5(a4,clk,en,a5);
fdce16 f6(a5,clk,en,a6);
fdce16 f7(a6,clk,en,a7);
fdce16 f8(a7,clk,en,a8);
endmodule
```

///Module_fdce16…

```
`timescale 1ns / 1ps
module fdce16(a,clk,en,y);
        input [15:0] a;
        input clk,en;
        output [15:0] y;
fdce d1(y[0],clk,en,a[0]);
fdce d2(y[1],clk,en,a[1]);
fdce d3(y[2],clk,en,a[2]);
fdce d4(y[3],clk,en,a[3]);
fdce d5(y[4],clk,en,a[4]);
fdce d6(y[5],clk,en,a[5]);
fdce d7(y[6],clk,en,a[6]);
fdce d8(y[7],clk,en,a[7]);
fdce d9(y[8],clk,en,a[8]);
fdce d10(y[9],clk,en,a[9]);
fdce d11(y[10],clk,en,a[10]);
fdce d12(y[11],clk,en,a[11]);
fdce d13(y[12],clk,en,a[12]);
fdce d14(y[13],clk,en,a[13]);
fdce d15(y[14],clk,en,a[14]);
```

```
fdce d16(y[15],clk,en,a[15]);
endmodule
```

///Module_fdce…

```
`timescale 1ns / 1ps
module fdce(q,clk,ce,d);
   input d,clk,ce;
   output reg q;
initial begin q=0; end
always @ (posedge (clk)) begin
        if (ce)
                q <= d;
        else
                q <= q ;
        end
endmodule
```

///Testbench_S2P…

```
`timescale 1ns / 1ps
module S2P_tb;
        reg clk;
        reg en;
        reg [15:0] a;
        wire [15:0] a1;
        wire [15:0] a2;
        wire [15:0] a3;
        wire [15:0] a4;
        wire [15:0] a5;
        wire [15:0] a6;
```

```
        wire [15:0] a7;
        wire [15:0] a8;
        S2P uut (
                .clk(clk),
                .en(en),
                .a(a),
                .a1(a1),
                .a2(a2),
                .a3(a3),
                .a4(a4),
                .a5(a5),
                .a6(a6),
                .a7(a7),
                .a8(a8)
        );
        always #5 clk = ~clk;
  initial begin
     clk = 0;
     en = 1;
     a = 16'habcd;
        end
endmodule
```

KHỐI 8-point FFT

///fft_8point…

```
`timescale 1ns / 1ps
module FFT_8point(clk,rst,a1,a2,a3,a4,a5,a6,a7,a8,
 d1,dr2,di2,dr3,di3,dr4,di4,d5,dr6,di6,dr7,di7,dr8,di8);
```

```
input [15:0] a1,a2,a3,a4,a5,a6,a7,a8;
wire [15:0] b1,b2,b3,b4,b5,b6,b7,b8;
wire [15:0] c1,cr2,ci2,c3,cr4,ci4,c5,cr6,ci6,c7,cr8,ci8;
output [15:0] d1,dr2,di2,dr3,di3,dr4,di4,d5,dr6,di6,dr7,di7,dr8,di8;
input clk,rst;
FFT_stage1 stg1(clk,rst,a1,a2,a3,a4,a5,a6,a7,a8,
                    b1,b2,b3,b4,b5,b6,b7,b8);
FFT_stage2 stg2(clk,rst,b1,b3,b2,b4,b5,b7,b6,b8,
                   c1,cr2,ci2,c3,cr4,ci4,c5,cr6,ci6,c7,cr8,ci8);
FFT_stage3 stg3(clk,rst,c1,c5,cr2,ci2,cr6,ci6,c3,c7,cr4,ci4,cr8,ci8,
                  d1,dr2,di2,dr3,di3,dr4,di4,d5,dr6,di6,dr7,di7,dr8,di8);
endmodule
```

///module_stage1…

```
`timescale 1ns / 1ps
module FFT_stage1(clk,rst,a1,a2,a3,a4,a5,a6,a7,a8,
                    b1,b2,b3,b4,b5,b6,b7,b8);
input [15:0] a1,a2,a3,a4,a5,a6,a7,a8;
output [15:0] b1,b2,b3,b4,b5,b6,b7,b8;
input clk,rst;
BF1 m1(clk,rst,a1,a5,b1,b2);
BF1 m2(clk,rst,a3,a7,b3,b4);
BF1 m3(clk,rst,a2,a6,b5,b6);
BF1 m4(clk,rst,a4,a8,b7,b8);
endmodule
```

///module_BF1…

```
`timescale 1ns / 1ps
```

```
module BF1(clk,rst,a,b,y1,y2);
input clk,rst;
input [15:0] a,b;
output [15:0] y1,y2;
wire [15:0] s1,s2;
addsub ad1(1'b0,a, b, s1);
addsub ad2(1'b1,a, b, s2);
reg16 f1(y1,clk,rst,s1);
reg16 f2(y2,clk,rst,s2);
endmodule
```

///**module_addsub…**

```
`timescale 1ns / 1ps
module addsub (add,a, b, s);
  input add;
  input [15 : 0] a;
  input [15 : 0] b;
  output [15 : 0] s;
  assign s = (add)? (a - b) : (a + b);
endmodule
```

///**module_reg16…**

```
`timescale 1ns / 1ps
module reg16(y,clk,reset,a);
    input [15:0] a;
    output [15:0] y;
    input clk,reset;
DFF d1(y[0],clk,reset,a[0]);
DFF d2(y[1],clk,reset,a[1]);
```

```
DFF d3(y[2],clk,reset,a[2]);
DFF d4(y[3],clk,reset,a[3]);
DFF d5(y[4],clk,reset,a[4]);
DFF d6(y[5],clk,reset,a[5]);
DFF d7(y[6],clk,reset,a[6]);
DFF d8(y[7],clk,reset,a[7]);
DFF d9(y[8],clk,reset,a[8]);
DFF d10(y[9],clk,reset,a[9]);
DFF d11(y[10],clk,reset,a[10]);
DFF d12(y[11],clk,reset,a[11]);
DFF d13(y[12],clk,reset,a[12]);
DFF d14(y[13],clk,reset,a[13]);
DFF d15(y[14],clk,reset,a[14]);
DFF d16(y[15],clk,reset,a[15]);
endmodule
```

///**module_DFF…**

```
`timescale 1ns / 1ps
module DFF(q,clk,reset,d);
  input d,clk,reset;
  output reg q;
initial begin q=0; end
always @ (posedge (clk)) begin
        if (reset)
                q <= 0;
        else
                q<= d ;
        end
```

```
endmodule
```

///Testbench_BF1…

```
`timescale 1ns / 1ps
module BF1_tb;
	reg clk;
	reg rst;
	reg [15:0] a;
	reg [15:0] b;
	wire [15:0] y1;
	wire [15:0] y2;
	BF1 uut (
		.clk(clk),
		.rst(rst),
		.a(a),
		.b(b),
		.y1(y1),
		.y2(y2)
	);
	always #5 clk = ~clk;
	initial begin
		clk = 0;
		rst = 1;
		a = 0;
		b = 0;
		#20;
		rst = 0;
		a = 258;
```

```
                b = 384;
                #50;
                a = -250;
                b = -300;
        end
endmodule
```

///module_stage2…

```
`timescale 1ns / 1ps
module FFT_stage2(clk,rst,a1,a2,a3,a4,a5,a6,a7,a8,
                    c1,cr2,ci2,c3,cr4,ci4,c5,cr6,ci6,c7,cr8,ci8);
input [15:0] a1,a2,a3,a4,a5,a6,a7,a8;
output [15:0] c1,cr2,ci2,c3,cr4,ci4,c5,cr6,ci6,c7,cr8,ci8;
input clk,rst;
wire [15:0] b1,b2,b3,b4,b5,br6,bi6,b7,br8,bi8;
BF1 m1(clk,rst,a1,a2,b1,b3);
BF2 m2(clk,rst,b2,b4,cr2,cr4,ci2,ci4);
BF1 m3(clk,rst,a5,a6,b5,b7);
BF2 m4(clk,rst,a7,a8,br6,br8,bi6,bi8);
Delayblock df1(clk,rst,b1,c1);
Delayblock df2(clk,rst,b3,c3);
Delayblock df3(clk,rst,a3,b2);
Delayblock df4(clk,rst,a4,b4);
Delayblock df5(clk,rst,b5,c5);
Delayblock df6(clk,rst,b7,c7);
complex_mul cm1(clk,rst,br6,bi6,cr6,ci6);
complex_mul1 cm2(clk,rst,br8,bi8,cr8,ci8);
endmodule
```

///**module_BF2…**

```
`timescale 1ns / 1ps
module BF2(clk,rst,a,b,a1,a2,b1,b2);
input [15:0] a,b;
output [15:0] a1,a2,b1,b2;
input clk,rst;
wire [15:0] s;
parameter zro = 16'b0000000000000000;
reg16 f1(a1,clk,rst,a);
addsub ad1(1'b1,zro, b, s);
reg16 f2(b1,clk,rst,s);
reg16 f3(b2,clk,rst,b);
assign a2 = a1;
endmodule
```

(module reg16 và addsub như trên)

///**testbench_BF2…**

```
`timescale 1ns / 1ps
module BF2_tb;
        reg clk;
        reg rst;
        reg [15:0] a;
        reg [15:0] b;
        wire [15:0] a1;
        wire [15:0] a2;
        wire [15:0] b1;
        wire [15:0] b2;
        BF2 uut (
```

```
        .clk(clk),
        .rst(rst),
        .a(a),
        .b(b),
        .a1(a1),
        .a2(a2),
        .b1(b1),
        .b2(b2)
    );
    always #5 clk = ~clk;
    initial begin
        clk = 0;
        rst = 1;
        a = 0;
        b = 0;
        #20;
        rst = 0;
        a = 16'h0100;
        b = 16'h0182;
        #50;
        a = 16'h0001;
        b = 16'h0004;
    end
endmodule
```

///module_DelayBlock…

```
`timescale 1ns / 1ps
module Delayblock(clk,rst,a,b);
```

```
input [15:0] a;
output [15:0] b;
input clk,rst;
wire [15:0] a1,a2,a3,a4;
reg16 f1(a1,clk,rst,a);
reg16 f2(a2,clk,rst,a1);
reg16 f3(a3,clk,rst,a2);
reg16 f4(b,clk,rst,a3);
endmodule
```

(module reg16 như trên)

///module_Complex_mul…

```
`timescale 1ns / 1ps
module complex_mul(clk,rst,a,b,R,I);
input [15:0] a,b;
output [15:0] R,I;
input clk,rst;
wire [15:0] s1,s2,s3,s4,s5,s6;
addsub ad1(1'b0,a, b, s1);
addsub ad2(1'b1,a, b, s2);
reg16 f1(s3,clk,rst,s1);
reg16 f2(s4,clk,rst,s2);
Constant_mul cm1(clk,rst,s3,s5);
Constant_mul cm2(clk,rst,s4,s6);
reg16 f3(R,clk,rst,s5);
reg16 f4(I,clk,rst,s6);
endmodule
```

(addsub và reg16 như trên)

///**module_Constant_mul…**

```
`timescale 1ns / 1ps
module Constant_mul(clk,rst,a,op);
input [15:0] a;
output [15:0] op;
input clk,rst;
wire [15:0] t1,t2,t3,t4,t5,s1,s2,s3,s4,s5,s6,s7,s8;
rsh1 m1(a,t1);
rsh3 m2(a,t2);
rsh4 m3(a,t3);
rsh6 m4(a,t4);
rsh8 m5(a,t5);
addsub ad1(1'b0,t1, t2, s1);
addsub ad2(1'b0,t3, t4, s2);
reg16 f1(s3,clk,rst,s1);
reg16 f2(s4,clk,rst,s2);
reg16 f3(s5,clk,rst,t5);
addsub ad3(1'b0,s3, s4, s6);
reg16 f4(s7,clk,rst,s6);
reg16 f5(s8,clk,rst,s5);
addsub ad4(1'b0,s7, s8, op);
endmodule
```

(addsub và reg16 như trên)

///**module_rsh1…**

```
`timescale 1ns / 1ps
module rsh1(a,b);
    input [15:0] a;
```

```
    output [15:0] b;
assign {b[15],b[14:0]}= {a[15],a[15:1]};
endmodule
```
///**module_rsh3…**
```
`timescale 1ns / 1ps
module rsh3(a,b);
    input [15:0] a;
    output [15:0] b;
assign {b[15:12],b[11:0]}= {a[15],a[15],a[15],a[15:3]};
endmodule
```
///**module_rsh4…**
```
`timescale 1ns / 1ps
module rsh4(a,b);
    input [15:0] a;
    output [15:0] b;
assign {b[15:11],b[10:0]}= {a[15],a[15],a[15],a[15],a[15:4]};
endmodule
```
///**module_rsh6…**
```
`timescale 1ns / 1ps
module rsh6(a,b);
    input [15:0] a;
    output [15:0] b;
assign {b[15:9],b[8:0]}= {a[15],a[15],a[15],a[15],a[15],a[15],a[15:6]};
endmodule
```
///**module_rsh8…**
```
`timescale 1ns / 1ps
module rsh8(a,b);
```

```
    input [15:0] a;
    output [15:0] b;
assign {b[15:7],b[6:0]}= {a[15],a[15],a[15],a[15],a[15],a[15],a[15],a[15],a[15:8]};
endmodule
```

///testbench_Complex_mul…

```
`timescale 1ns / 1ps
module Complex_Mul_tb;
        reg clk;
        reg rst;
        reg [15:0] a;
        reg [15:0] b;
        wire [15:0] R;
        wire [15:0] I;
        complex_mul uut (
                .clk(clk),
                .rst(rst),
                .a(a),
                .b(b),
                .R(R),
                .I(I)
        );
        always #5 clk = ~clk;
        initial begin
                clk = 0;
                rst = 0;
                a = 16'h1234;
                b = 16'h5678;
```

```
            #100;
            rst = 1;
            #10;
            rst = 0;
        end
endmodule
```

///module_stage3…

```
`timescale 1ns / 1ps
module FFT_stage3(clk,rst,a1,a2,ar3,ai3,ar4,ai4,a5,a6,ar7,ai7,ar8,ai8,
                  b1,br2,bi2,br3,bi3,br4,bi4,b5,br6,bi6,br7,bi7,br8,bi8);
input [15:0] a1,a2,ar3,ai3,ar4,ai4,a5,a6,ar7,ai7,ar8,ai8;
output [15:0] b1,br2,bi2,br3,bi3,br4,bi4,b5,br6,bi6,br7,bi7,br8,bi8;
input clk,rst;
////for Real part
BF1 m1(clk,rst,a1,a2,b1,b5);
BF1 m2(clk,rst,ar3,ar4,br6,br2);
BF2 m3(clk,rst,a5,a6,br3,br7,bi3,bi7);
BF1 m4(clk,rst,ar7,ar8,br8,br4);
////for imagnary part
BF1 m5(clk,rst,ai3,ai4,bi6,bi2);
BF1 m6(clk,rst,ai7,ai8,bi8,bi4);
endmodule
```

(module BF1 và BF2 như trên)

///testbench_8point_FFT…

```
`timescale 1ns / 1ps
module FFT_tb;
        reg clk;
```

```
        reg rst;
        reg start;
        wire [15:0] Re;
        wire [15:0] Im;
        FFT_processor uut (
                .clk(clk),
                .rst(rst),
                .start(start),
                .Re(Re),
                .Im(Im)
        );
        always #5 clk = ~clk;
        initial begin
                // Initialize Inputs
                clk = 0;
                rst = 0;
                start = 0;
                // Wait 100 ns for global reset to finish
                #300;
                start = 1;
                #10;
                start = 0;
        end
endmodule
```

KHỐI P2S

///Synthesis…

```
`timescale 1ns / 1ps
```

```
module P2S(clk,ld,en,a1,a2,a3,a4,a5,a6,a7,a8,t8);
input clk,ld,en;
input [15:0] a1,a2,a3,a4,a5,a6,a7,a8;
output [15:0] t8;
wire [15:0] p1,p2,p3,p4,p5,p6,p7;
wire [15:0] t1,t2,t3,t4,t5,t6,t7;
fdce16 f1(a1,clk,en,t1);
mux16 mx1(t1,a2,ld,p1);
fdce16 f2(p1,clk,en,t2);
mux16 mx2(t2,a3,ld,p2);
fdce16 f3(p2,clk,en,t3);
mux16 mx3(t3,a4,ld,p3);
fdce16 f4(p3,clk,en,t4);
mux16 mx4(t4,a5,ld,p4);
fdce16 f5(p4,clk,en,t5);
mux16 mx5(t5,a6,ld,p5);
fdce16 f6(p5,clk,en,t6);
mux16 mx6(t6,a7,ld,p6);
fdce16 f7(p6,clk,en,t7);
mux16 mx7(t7,a8,ld,p7);
fdce16 f8(p7,clk,en,t8);
endmodule
```

(module fdce16 như trên)

///**module_mux16…**

```
`timescale 1ns / 1ps
module mux16(A,B,S,Y);
    input [15:0] A,B;
```

```
    output [15:0] Y;
    input S;
assign Y = (S)? B : A;
endmodule
```

///testbench_P2S…

```
`timescale 1ns / 1ps
module P2S_tb;
        reg clk;
        reg ld;
        reg en;
        reg [15:0] a1;
        reg [15:0] a2;
        reg [15:0] a3;
        reg [15:0] a4;
        reg [15:0] a5;
        reg [15:0] a6;
        reg [15:0] a7;
        reg [15:0] a8;
        wire [15:0] t8;
        P2S uut (
                .clk(clk),
                .ld(ld),
                .en(en),
                .a1(a1),
                .a2(a2),
                .a3(a3),
                .a4(a4),
```

```
        .a5(a5),
        .a6(a6),
        .a7(a7),
        .a8(a8),
        .t8(t8)
    );
    always #5 clk = ~clk;
    initial begin
        clk = 0;
        ld = 0;
        en = 1;
        a1 = 16'h1234;
        a2 = 16'h5678;
        a3 = 16'hABCD;
        a4 = 16'hCDEF;
        a5 = 16'h0123;
        a6 = 16'h7894;
        a7 = 16'h1987;
        a8 = 16'h4561;
        #10;
        ld = 1;
        #10;
        ld = 0;
    end
endmodule
```

BỘ ĐIỀU KHIỂN

///**module_pg…**

```
`timescale 1ns / 1ps
module pg(start,tc,q,clk,reset);
        input start,tc,clk,reset;
        output  q;
        wire t1,t2;
        parameter vdd=1'b1;
        parameter gnd=1'b0;
        mux M1(t2,vdd,start,q);
        mux M2(q,gnd,tc,t1);
        DFF d2(t2,clk,reset,t1);
endmodule
```

///**module_mux…**

```
`timescale 1ns / 1ps
module mux(A,B,S,Y);
   input  A,B;
   output  Y;
   input S;
assign Y = (S)? B : A;
endmodule
```

(module DFF như trên)

///**testbench_pg…**

```
`timescale 1ns / 1ps
module pg_tb;
        reg start;
        reg tc;
        reg clk;
        reg reset;
```

```
        wire q;
        pg uut (
                .start(start),
                .tc(tc),
                .q(q),
                .clk(clk),
                .reset(reset)
        );
        always #5 clk = ~clk;
        initial begin
                start = 0;
                tc = 0;
                clk = 0;
                reset = 0;
                #105;
                start = 1;
                #35;
                start = 0;
                tc = 1;
        end
endmodule
```

///module_cnt4…

```
`timescale 1ns / 1ps
module cnt4(out,data,load,en,clk,rst,tc,lmt);
output [2:0] out;
output reg tc;
input [2:0] data;
```

```
input load, en, clk,rst;
reg [2:0] out;
input [2:0]lmt;
initial begin out=3'b000;
        tc=0; end
always @(posedge clk)
        if (rst) begin
                out <= 3'b000 ;
        end
        else if (load) begin
                out <= data;
        end
        else if (en)
                out <= out + 3'b001;
        else out <= out;
always @(posedge clk)
        if (out ==lmt)
                tc<=1;
        else tc<=0;
endmodule
```

///**testbench_cnt4…**

```
`timescale 1ns / 1ps
module cnt4_tb;
        reg [2:0] data;
        reg load;
        reg en;
        reg clk;
```

```
reg rst;
reg [2:0] lmt;
wire [2:0] out;
wire tc;
cnt4 uut (
        .out(out),
        .data(data),
        .load(load),
        .en(en),
        .clk(clk),
        .rst(rst),
        .tc(tc),
        .lmt(lmt)
);
always #5 clk = ~clk;
  initial begin
      clk = 0;
      rst = 1;
      load = 0;
      en = 1;
      data = 3'b000;
      lmt = 3'b100;
      #20;
      rst = 0;
   end
        initial begin
      #10;
```

```
        load = 1;
        #10;
        load = 0;
        #10;
        en = 1;
    end
endmodule
```