COD_LAB3 寄存器堆与计数器

黄业琦 PB17000144

April 7, 2019

Contents

1 实验目的

1.1 寄存器堆（Register File）

ra0, rd0; ra1, rd1：2 个异步读端口
wa, wd, we：1 个同步写端口

1.2 计数器（Counter）
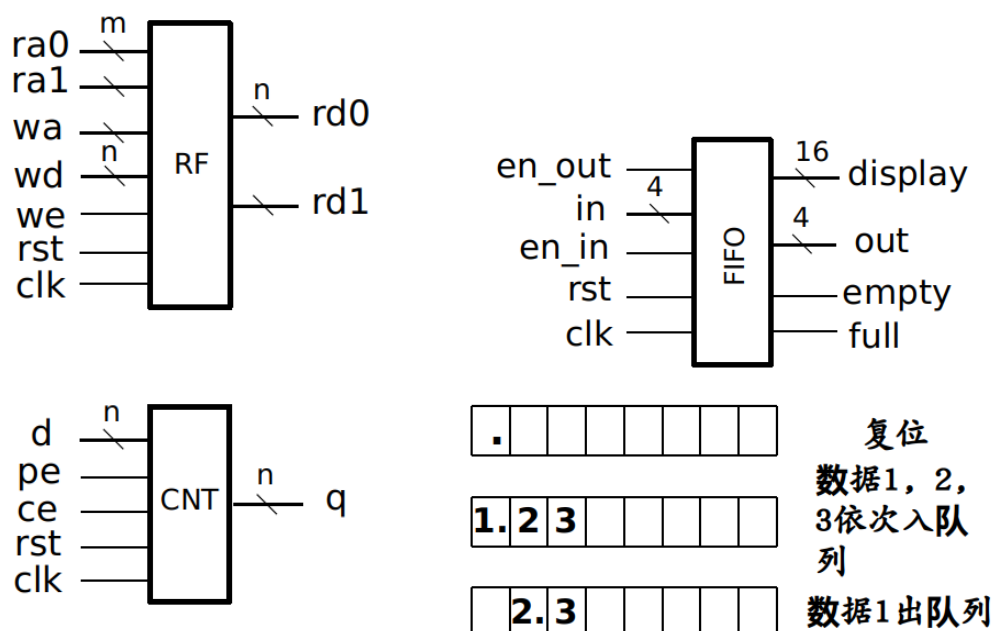
ce：计数使能，1: q=q+1
pe：同步装数使能，1: q=d
rst：异步清零，1: q=0

Figure 1: Lab Requirement

1.3 FIFO

最大长度为 8 的 FIFO 循环队列：用寄存器堆和适当逻辑实现。
en_out, en_in：出/入队列使能，一次有效仅允许操作一项数据
out, in：出/入队列数据
full, empty：队列空/满，空/满时忽略出/入队操作
display：8 个数码管的控制信号，显示队列状态

2 实验环境

Linux 下编程调试和仿真，使用 IVerilog，GtkWave 系列工具。
Windows 下用于生成比特流文件，使用 Vivado 2018.2，Verilog HDL
所有下载均在 Nexsy4-DDR 实验板完成

3 逻辑设计

3.1 Regfile Design

两个地址读入位置，直接用有带宽的向量作为存储寄存器。我们直接访问对应位置的
元素进行访问即可。

3.2 Counter Design

计数器，设计简单，与上学期的 lab8,9 一致，不再赘述。

3.3 FIFO_q Design

FIFO 设计：
我并没有单独调用我自己设计的 Regfile，因为队列的特殊性，我决定直接设计有带宽
的向量组直接使用。
这样大大降低了编程的复杂程度。
计数器我则用于分频，分八个段供八个七段数码管使用。
由于传向量组并不是 Verilog 的合法语法，有两种解决办法：

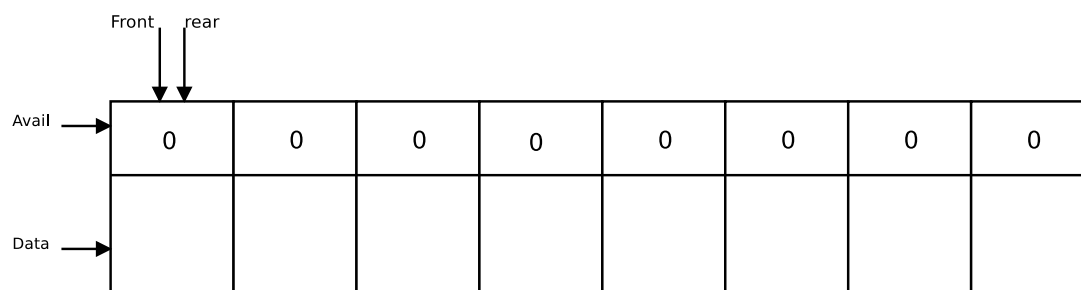1. 避免调用，合并程序。

2. 压缩向量，变为 1 维。
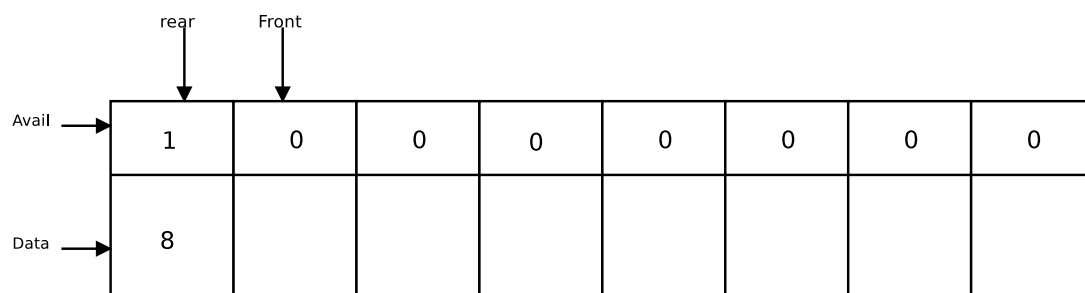
权衡选择第一种。入队出队操作见示意图：

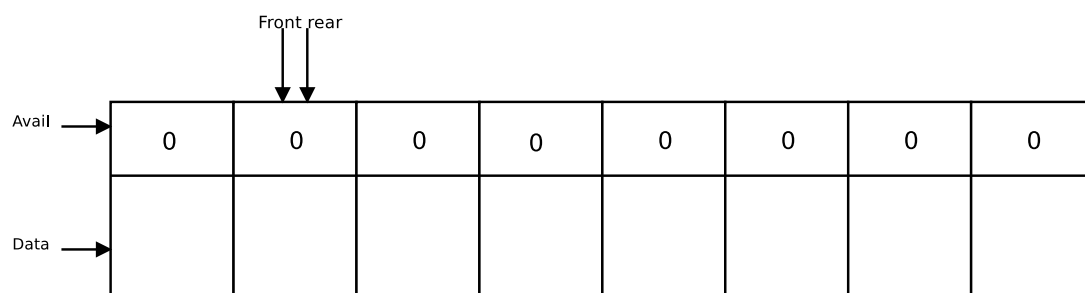Figure 2: fifoshow1

Figure 3: fifoshow2

Figure 4: fifoshow3

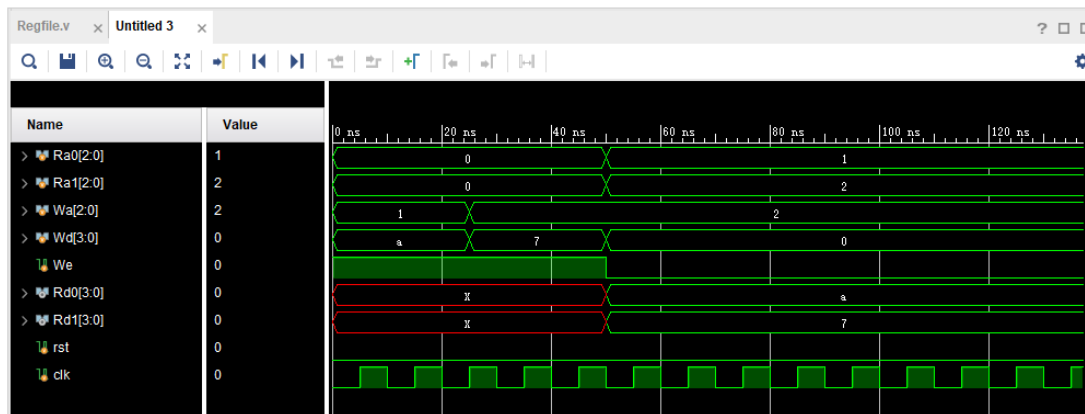front 指向对头后一个元素，rear 指向队尾元素。

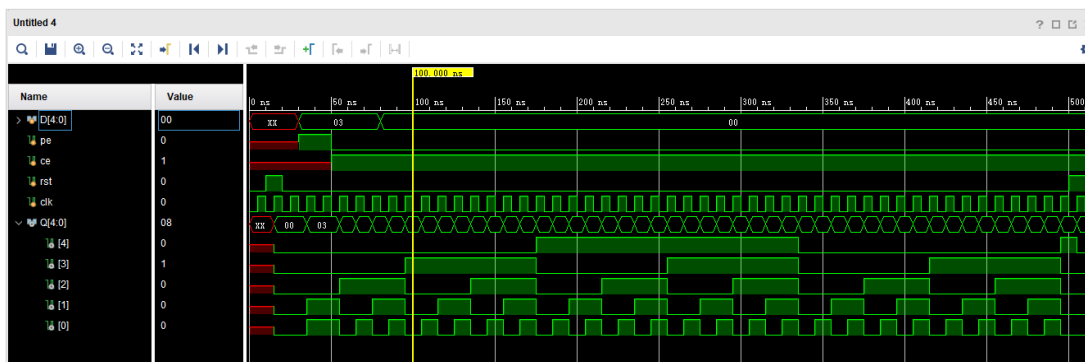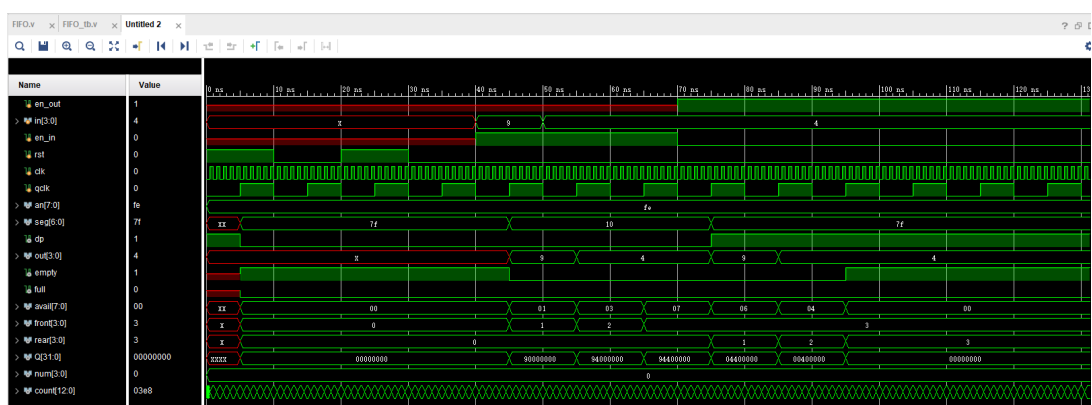# 4 仿真截图

Figure 5: Regsim

Figure 6: Countsim

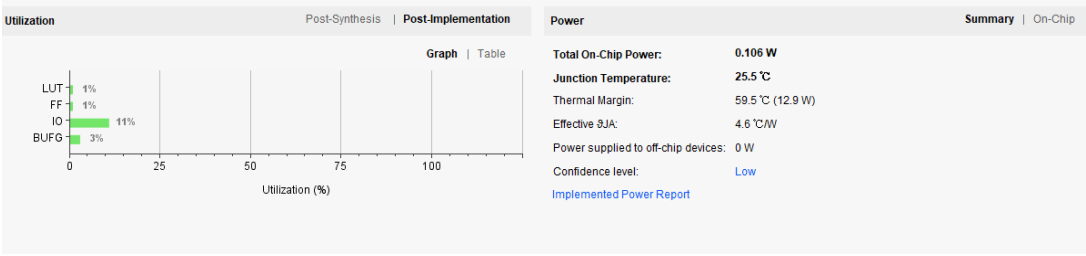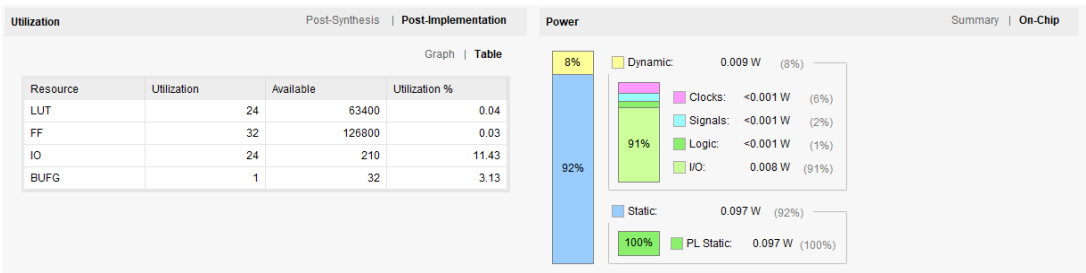Figure 7: FIFOsim

5 性能评测截图

Figure 8: Regper1

Figure 9: Regper2
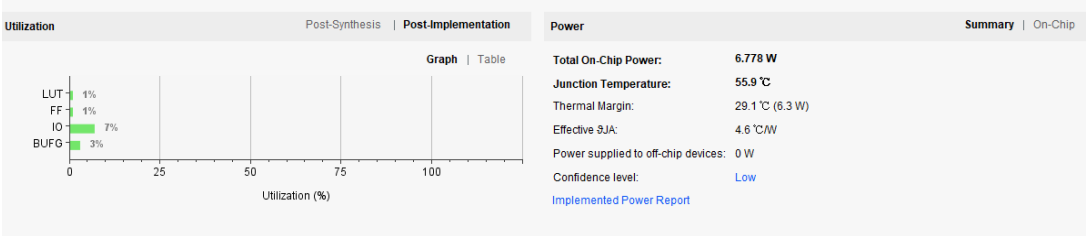
Figure 10: Countper1
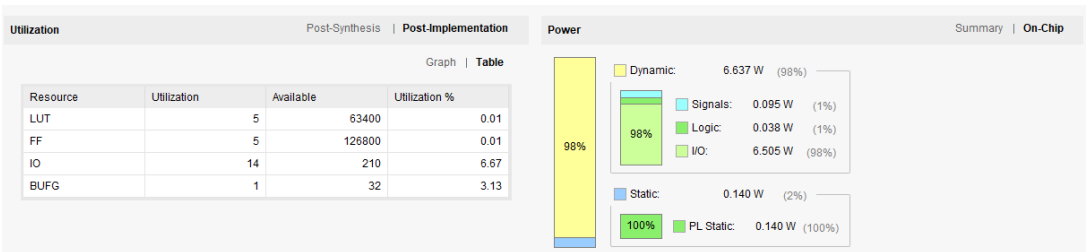
Figure 11: Countper2
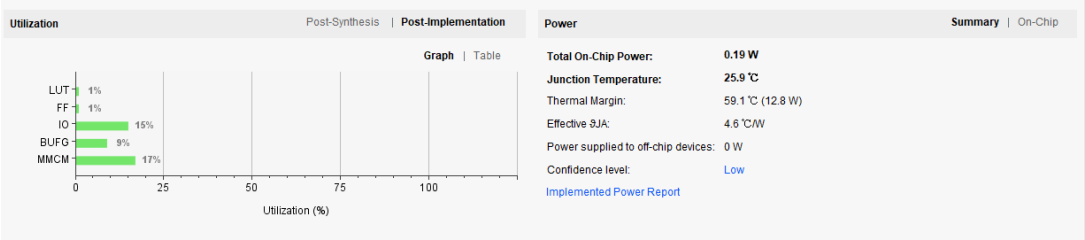
Figure 12: FIFOper1
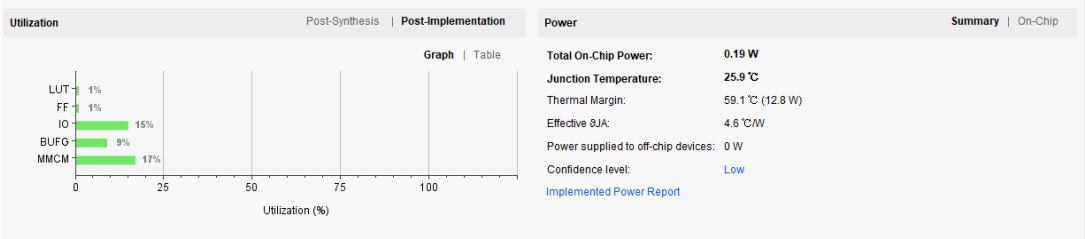
Figure 13: FIFOper1

6 实验代码

6.1 Regfile 代码

Listing 1: Regfile.v

```verilog
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/04/09 11:02:39
7  // Design Name:
8  // Module Name: Regfile
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 `define PARA_M 3 //address 3bits
24 `define PARA_N 4 //data    4bits
25 `define BUSM   2:0
26 `define BUSN   3:0
27 `define BUSREG 7:0
28 (* use_dsp = "yes" *)
29
30 module Regfile(
31     input  [`BUSM] Ra0,
32     input  [`BUSM] Ra1,
33     input  [`BUSM] Wa,
34     input  [`BUSN] Wd,
35     input          We,
36     input          rst,
37     input          clk,
38     output [`BUSN] Rd0,
39     output [`BUSN] Rd1
40     );
41
42     reg [`BUSN] Regs [`BUSREG];
43     integer i;
44
45     always @(posedge clk or negedge rst)
```

```verilog
46     begin
47       if (rst)
48       begin
49         for (i = 0 ; i < 8 ; i = i + 1 )
50             Regs[i] <= 0;
51       end
52       else
53       if (We)
54       Regs[Wa] = Wd;
55     end
56
57     assign Rd0 = Regs[Ra0];
58     assign Rd1 = Regs[Ra1];
59 endmodule
```

Listing 2: RegSim.v

```verilog
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/04/09 20:21:46
7  // Design Name:
8  // Module Name: RegSim
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////
21
22 `define PARA_M 3 //address 3bits
23 `define PARA_N 4 //data    4bits
24 `define BUSM   2:0
25 `define BUSN   3:0
26 `define BUSREG 7:0
27 module RegSim;
28  reg  [`BUSM] Ra0;
29  reg  [`BUSM] Ra1;
30  reg  [`BUSM] Wa;
31  reg  [`BUSN] Wd;
32  reg          We;
33  wire [`BUSN] Rd0;
34  wire [`BUSN] Rd1;
35  reg rst;
```

9

```verilog
36  reg clk;
37
38  initial rst<=0;
39  always #500 rst<=~rst;
40
41  initial clk<=0;
42  always #5 clk<=~clk;
43
44  initial
45  begin
46    Ra0=0;
47    Ra1=0;
48    Wa =1;
49    We =1;
50    Wd =10;
51    #25
52
53    Ra0=0;
54    Ra1=0;
55    Wa =2;
56    We =1;
57    Wd =7;
58    #25
59
60    Ra0=1;
61    Ra1=2;
62    Wa =2;
63    We =0;
64    Wd =0;
65    #25;
66  end
67
68  Regfile DUT (Ra0,Ra1,Wa,Wd,We,rst,clk,Rd0,Rd1);
69  endmodule
```

6.2 Counter 代码

Listing 3: counter.v

```verilog
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/04/09 21:41:57
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
```

```verilog
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

`define PARAN 5
`define BUSN  4:0

module counter(
    input [`BUSN] D,
    input pe,
    input ce,
    input rst,
    input clk,
    output [`BUSN] Q
    );

    reg [`BUSN] RegCount;

    always @(posedge clk or negedge rst)
    begin
      if (rst)
        RegCount = 0;
      else
      begin
        if (pe)
          RegCount = D;
        else
        begin
          if (ce)
          RegCount = RegCount + 1;
        end
      end
    end

    assign Q = RegCount;
endmodule
```

Listing 4: counter_tb.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/04/09 21:42:25
// Design Name:
```

```verilog
// Module Name: counter_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

`define PARAN 5
`define BUSN  4:0

module counter_tb(

    );

    reg [`BUSN] D;
    reg pe;
    reg ce;
    reg rst;
    reg clk;
    wire [`BUSN] Q;

    initial rst<=0;
    always #500 rst<=~rst;

    initial clk<=0;
    always #5 clk<=~clk;

    initial
    begin
        #10 rst = 1;
        #10 rst = 0;
        #10 D   = 3;
            pe  = 1;
        #20 pe  = 0;
            ce  = 1;
        #30 D   = 0;
    end
    counter DUT (D,pe,ce,rst,clk,Q);
endmodule
```

6.3 FIFO 代码

这里的注释部分为调试使用，代码的 36-41 行的取消注释，54 行取消注释，47 行换为 46 行，可以进行仿真。

Listing 5: clk10HZ.v

```verilog
module clk10HZ(
    input clk5MHZ,
    output reg Q
    );
    reg[0:23]count;
    initial
    begin
        count=0;
        Q=0;
    end

    always@(posedge clk5MHZ)
            if(count>=24'd249999)
            begin
                count<=24'b0;
                Q<=~Q;
            end
            else
                count<=count+24'h1;
endmodule
```

Listing 6: BCD27.v

```verilog
`timescale 1ns / 1ps

module BCD27(
    input [3:0]m,
    output [6:0] out
    );
    reg[6:0]seg;
    assign out = seg;
    always@(m)
    case(m)
                4'b0000:seg=7'b1000000;
                4'b0001:seg=7'b1111001;
                4'b0010:seg=7'b0100100;
                4'b0011:seg=7'b0110000;
                4'b0100:seg=7'b0011001;
                4'b0101:seg=7'b0010010;
                4'b0110:seg=7'b0000010;
                4'b0111:seg=7'b1111000;
                4'b1000:seg=7'b0000000;
                4'b1001:seg=7'b0010000;
                default:seg=7'b1111111;
                endcase
endmodule
```

Listing 7: counter.v

```verilog
module counter #(parameter cnt=9)
    (
    input clk,
    input enable,
    input reset,
    output carry,
    output reg [3:0] Q
    );
    assign carry=(Q==cnt)?1:0;
    initial
    begin
    Q<=0;
    end
    always@(posedge clk,posedge reset)
    begin
        if(reset)
        Q<=0;
        else if(enable)
        if(Q>=cnt)
            Q<=0;
        else
            Q=Q+1;
    end

endmodule
```

Listing 8: FIFO.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/04/10 01:53:42
// Design Name:
// Module Name: FIFO
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module FIFO(
```

```verilog
    input en_out,
    input [3:0] in,
    input en_in,
    input rst,
    input clk,
    input qclk,
    output reg [7:0] an,
    output reg [6:0] seg,
    output reg dp,
    output reg [3:0] out,
    output reg empty,
    output reg full/*,
    output [7:0] avail,
    output [3:0] front,
    output [3:0] rear,
    output [31:0] queue,
    output [3:0] num,
    output [12:0] count*/
    );

    wire clk5MHZ;
    wire clkp;
    //assign clk5MZ=clk;
    clk_wiz_0 cw1(clk5MHZ,clk);

    reg [3:0] Q [7:0];
    reg [7:0] avail;
    reg [3:0] front;
    reg [3:0] rear;
    integer i;
    //assign queue = {Q[0],Q[1],Q[2],Q[3],Q[4],Q[5],Q[6],Q[7]};
    always @(posedge qclk or negedge rst)
    begin
      if (rst)
      begin
        for (i = 0 ; i < 8 ; i = i + 1)
        Q[i] = 0;

        front = 0;
        rear  = 0;
        avail = 0;
        empty = 1;
        full  = 0;
      end
      else
      begin
        if (en_in)
        begin
            if (!avail[front])
            begin
                Q[front] = in;
```

```verilog
                    avail[front] = 1;
                    front = (front + 1) % 8;
                end

                out = in;
            end

        if (en_out)
        begin
            if (avail[rear])
            begin
                out = Q[rear];
                Q[rear] = 0;
                avail[rear] = 0;
                rear = (rear + 1) % 8;
            end

        end

        if (avail == 8'b11111111)
        full = 1;
        else
        full = 0;

        if (avail == 8'b00000000)
        empty = 1;
        else
        empty = 0;
    end
end


reg [3:0] num;
wire [6:0] seg0[7:0];
reg [12:0] count ;
reg clk1;

BCD27 B0(Q[0],seg0[0]);
BCD27 B1(Q[1],seg0[1]);
BCD27 B2(Q[2],seg0[2]);
BCD27 B3(Q[3],seg0[3]);
BCD27 B4(Q[4],seg0[4]);
BCD27 B5(Q[5],seg0[5]);
BCD27 B6(Q[6],seg0[6]);
BCD27 B7(Q[7],seg0[7]);

initial
begin
an<=8'b11111111;
clk1<=0;
num<=0;
```

```verilog
126        count<=0;
127    end
128
129    always@(posedge clk5MHZ)
130        if(count>=13'd1999)
131        begin
132            clk1<=~clk1;
133            count=13'b0;
134            num = (num+1)%8;
135        end
136        else
137        count=count+1;
138
139    always@*
140    case(num)
141        0:begin
142            an <=8'b11111110;
143            seg<=seg0[0];
144            if (!avail[0])
145            seg<=7'b1111111;
146
147            if (rear == 0)
148            dp <= 0;
149            else
150            dp <= 1;
151          end
152        1:begin
153            an <=8'b11111101;
154            seg<=seg0[1];
155            if (!avail[1])
156            seg<=7'b1111111;
157
158            if (rear == 1)
159            dp <= 0;
160            else
161            dp <= 1;
162          end
163        2:begin
164            an <=8'b11111011;
165            seg<=seg0[2];
166            if (!avail[2])
167            seg<=7'b1111111;
168
169            if (rear == 2)
170            dp <= 0;
171            else
172            dp <= 1;
173          end
174        3:begin
175            an <=8'b11110111;
176            seg<=seg0[3];
```

17

```verilog
177             if (!avail[3])
178             seg<=7'b1111111;
179
180             if (rear == 3)
181             dp <= 0;
182             else
183             dp <= 1;
184          end
185      4:begin
186             an <=8'b11101111;
187             seg<=seg0[4];
188             if (!avail[4])
189             seg<=7'b1111111;
190
191             if (rear == 4)
192             dp <= 0;
193             else
194             dp <= 1;
195          end
196      5:begin
197             an <=8'b11011111;
198             seg<=seg0[5];
199             if (!avail[5])
200             seg<=7'b1111111;
201
202             if (rear == 5)
203             dp <= 0;
204             else
205             dp <= 1;
206          end
207      6:begin
208             an <=8'b10111111;
209             seg<=seg0[6];
210             if (!avail[6])
211             seg<=7'b1111111;
212
213             if (rear == 6)
214             dp <= 0;
215             else
216             dp <= 1;
217          end
218      7:begin
219             an <=8'b01111111;
220             seg<=seg0[7];
221             if (!avail[7])
222             seg<=7'b1111111;
223
224             if (rear == 7)
225             dp <= 0;
226             else
227             dp <= 1;
```

```verilog
228          end
229      endcase
230  endmodule
```

Listing 9: FIFO_tb.v

```verilog
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/04/10 01:54:04
7  // Design Name:
8  // Module Name: FIFO_tb
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module FIFO_tb;
24     reg en_out;
25     reg [3:0] in;
26     reg en_in;
27     reg rst;
28     reg clk;
29     reg qclk;
30     wire [7:0] an;
31     wire [7:0] seg;
32     wire dp;
33     wire [3:0] out;
34     wire empty;
35     wire full;
36     wire [7:0]avail;
37     wire [3:0]front;
38     wire [3:0]rear;
39     wire [31:0] Q;
40     wire [3:0] num;
41     wire [12:0] count;
42
43     initial clk<=0;
44     always #0.5 clk<=~clk;
45     initial qclk<=0;
46     always #5 qclk<=~qclk;
```

```verilog
    initial
    begin
      rst = 1;
      #10 rst = 0;
      #10 rst = 1;
      #10 rst = 0;

      #10 en_in <= 1;
          in <= 9;
      #10 en_in <= 1;
          in <= 4;
      #20 en_in <= 0;
          en_out <= 1;

    end

    FIFO DUT (en_out,in,en_in,rst,clk,qclk,an,
                seg,dp,out,empty,full,avail,front,
                rear,Q,num,count);
endmodule
```