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

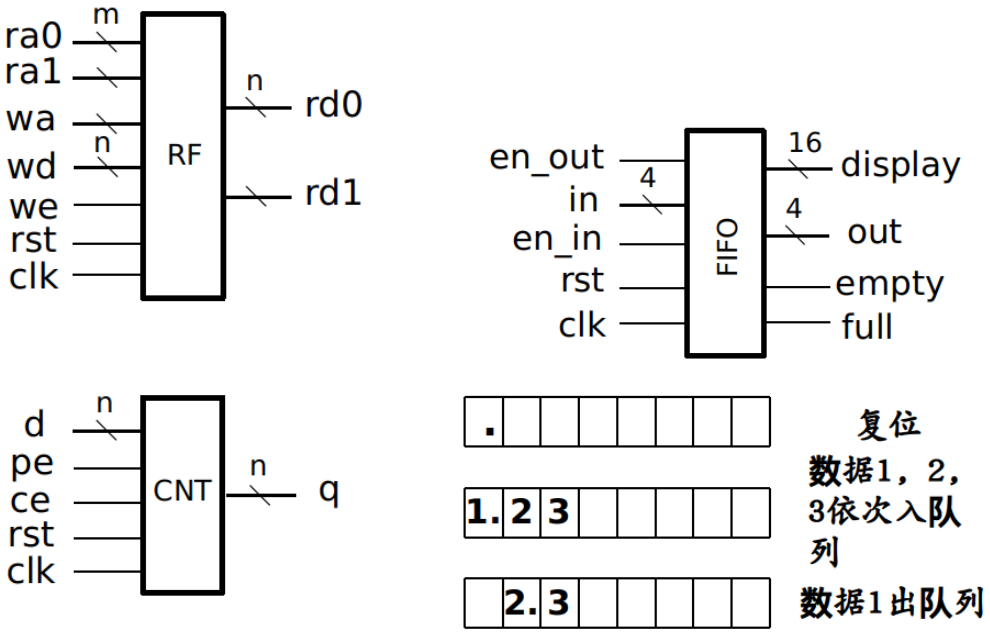

Figure 1: Lab Requirement

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：

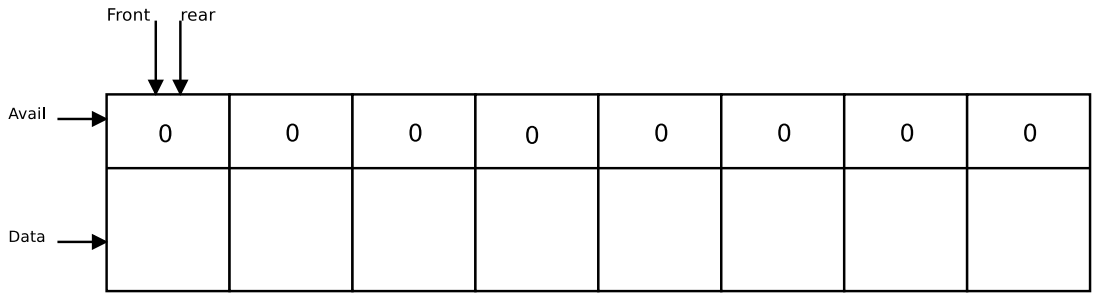

Figure 2: fifoshow1

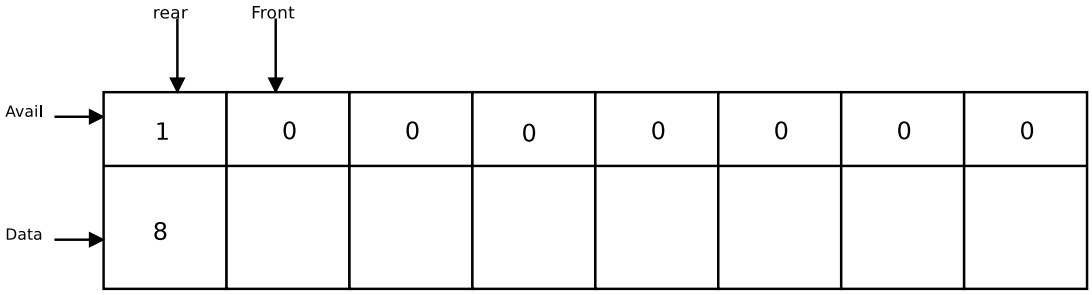

Figure 3: fifoshow2

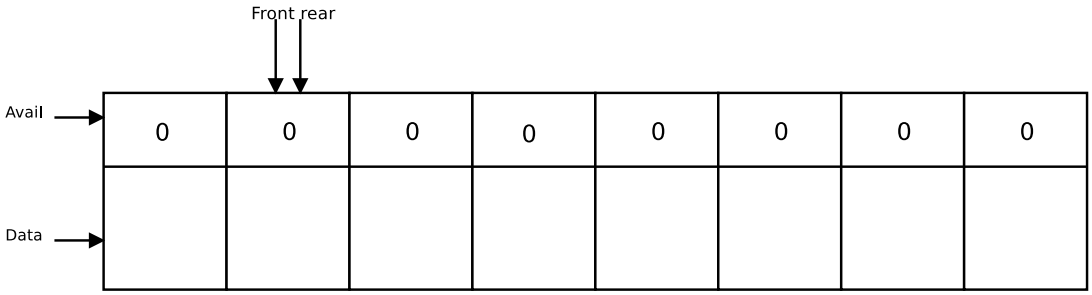

Figure 4: fifoshow3

front 指向对头后一个元素，rear 指向队尾元素。

4 仿真截图

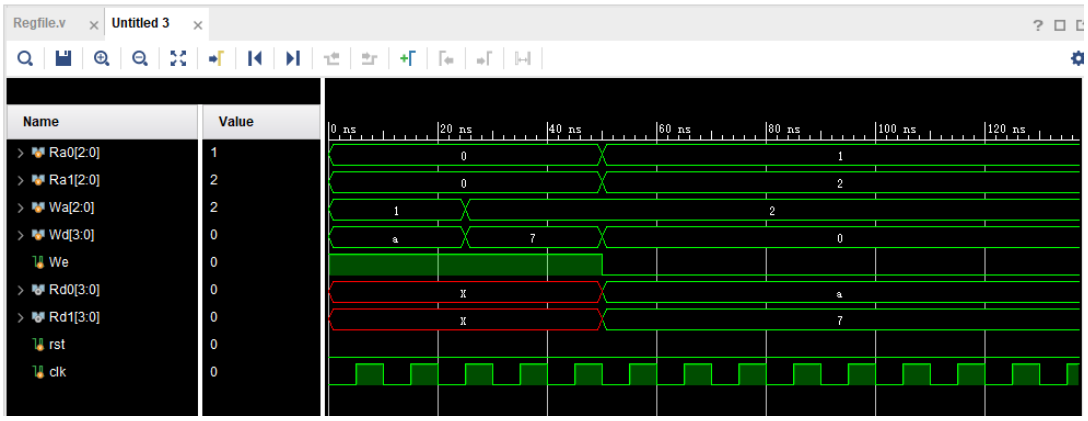

Figure 5: Regsim

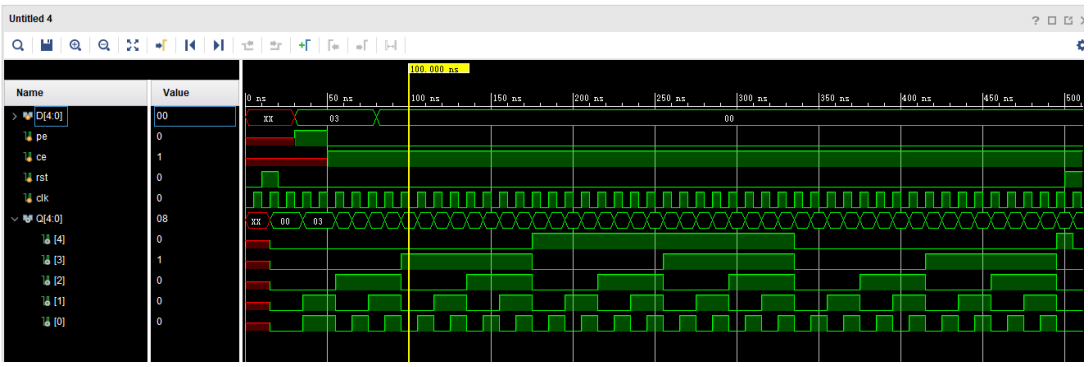

Figure 6: Countsim

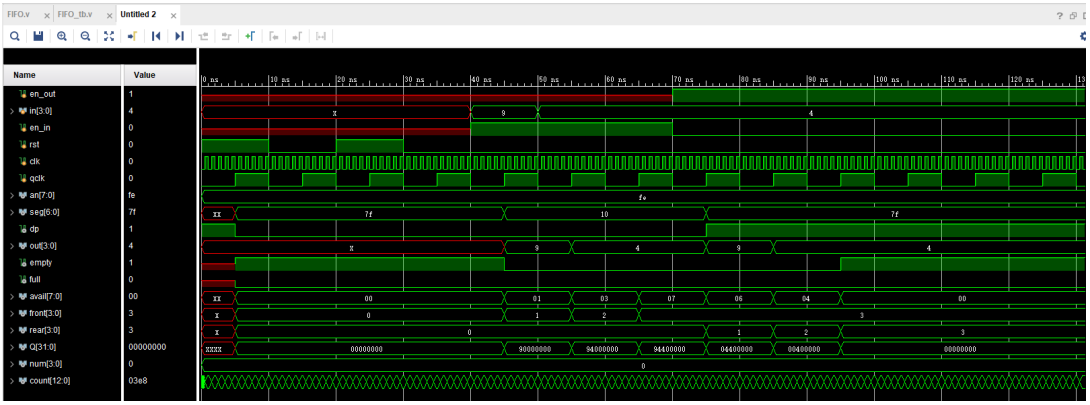

Figure 7: FIFOsim

5 性能评测截图

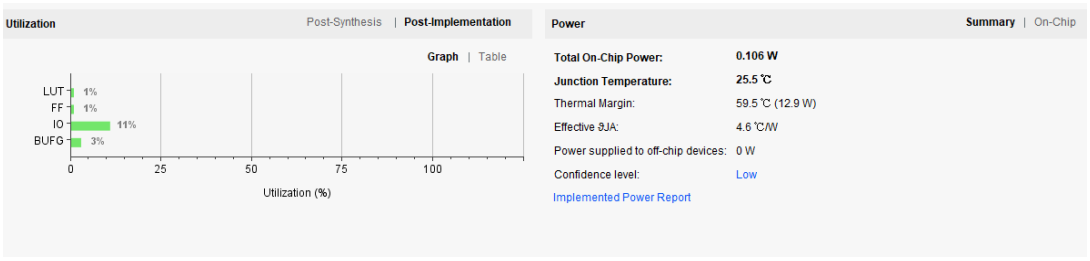

Figure 8: Regper1

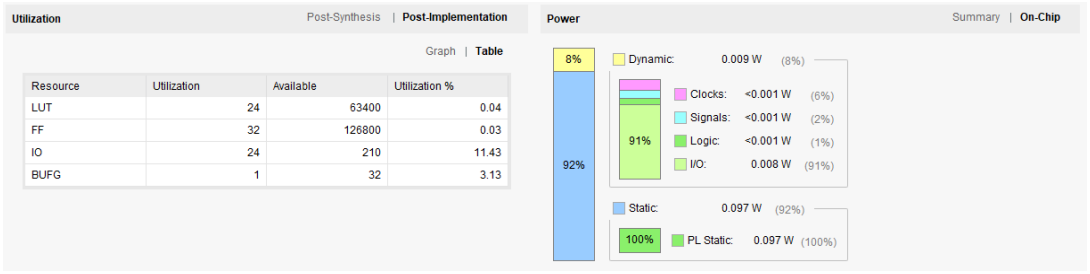

Figure 9: Regper2

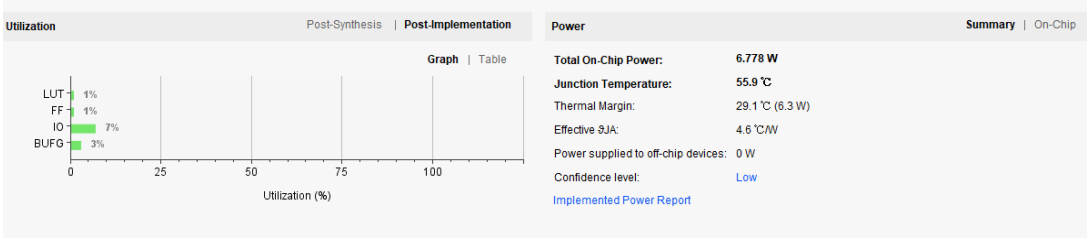

Figure 10: Countper1

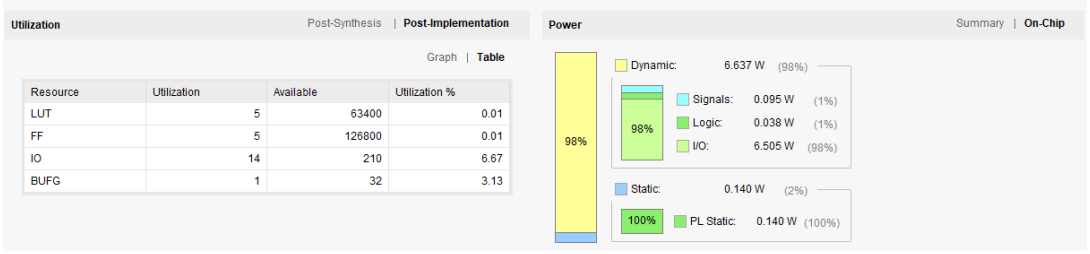

Figure 11: Countper2

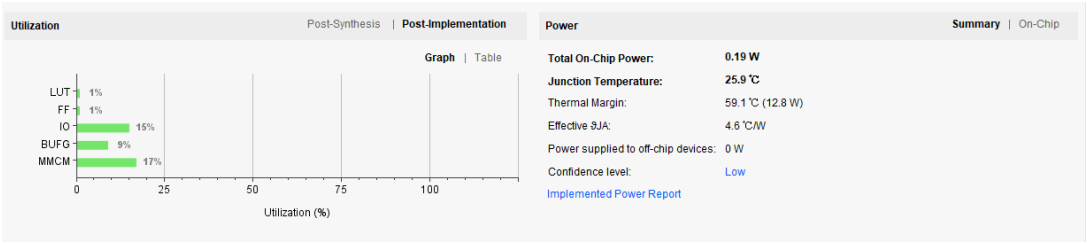

Figure 12: FIFOper1

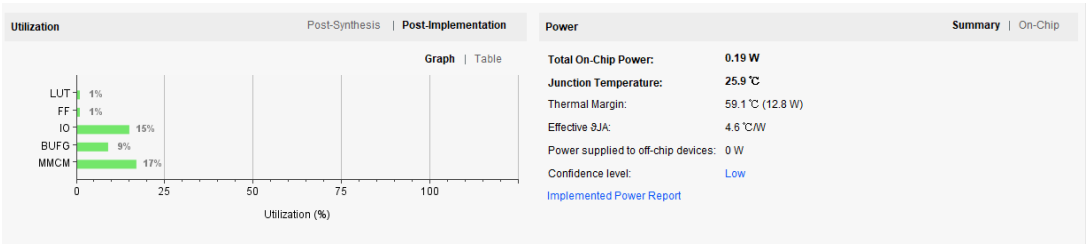

Figure 13: FIFOper1

6 实验代码

6.1 Regfile 代码

Listing 1: Regfile.v

```
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)
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

```
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;
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

```
1 `timescale 1ns / 1ps
2 //
    //////////////////////////////////////////////////////////////////////////////////
```

```
// Company:
// Engineer:
//
// Create Date: 2019/04/09 21:41:57
// Design Name:
// Module Name: counter
// 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(
    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

```

```
52      assign Q = RegCount;
53  endmodule
```

Listing 4: counter_tb.v

```
1   `timescale 1ns / 1ps
2   //
        //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 2019/04/09 21:42:25
7   // Design Name:
8   // Module Name: counter_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  `define PARAN 5
23  `define BUSN 4:0
24
25  module counter_tb(
26
27      );
28
29      reg [`BUSN] D;
30      reg pe;
31      reg ce;
32      reg rst;
33      reg clk;
34      wire [`BUSN] Q;
35
36      initial rst<=0;
37      always #500 rst<=~rst;
38
39      initial clk<=0;
40      always #5 clk<=~clk;
41
42      initial
43      begin
```

```
44          #10 rst = 1;
45          #10 rst = 0;
46          #10 D  = 3;
47              pe  = 1;
48          #20 pe = 0;
49              ce  = 1;
50          #30 D  = 0;
51      end
52      counter DUT (D,pe,ce,rst,clk,Q);
53  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

```
1   module clk10HZ(
2       input clk5MHZ,
3       output reg Q
4       );
5       reg[0:23]count;
6       initial
7       begin
8           count=0;
9           Q=0;
10      end
11
12      always@(posedge clk5MHZ)
13              if(count>=24'd249999)
14              begin
15                  count<=24'b0;
16                  Q<=~Q;
17              end
18              else
19                  count<=count+24'h1;
20  endmodule
```

Listing 6: BCD27.v

```
1   `timescale 1ns / 1ps
2
3   module BCD27(
4       input [3:0]m,
5       output [6:0] out
6       );
7       reg[6:0]seg;
8       assign out = seg;
9       always@(m)
10      case(m)
```

```
11              4'b0000:seg=7'b1000000;
12              4'b0001:seg=7'b1111001;
13              4'b0010:seg=7'b0100100;
14              4'b0011:seg=7'b0110000;
15              4'b0100:seg=7'b0011001;
16              4'b0101:seg=7'b0010010;
17              4'b0110:seg=7'b0000010;
18              4'b0111:seg=7'b1111000;
19              4'b1000:seg=7'b0000000;
20              4'b1001:seg=7'b0010000;
21              default:seg=7'b1111111;
22              endcase
23 endmodule
```

Listing 7: counter.v

```
1  module counter #(parameter cnt=9)
2      (
3      input clk,
4      input enable,
5      input reset,
6      output carry,
7      output reg [3:0] Q
8      );
9      assign carry=(Q==cnt)?1:0;
10     initial
11     begin
12     Q<=0;
13     end
14     always@(posedge clk,posedge reset)
15     begin
16         if(reset)
17         Q<=0;
18         else if(enable)
19         if(Q>=cnt)
20             Q<=0;
21         else
22             Q=Q+1;
23     end
24
25 endmodule
```

Listing 8: FIFO.v

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////

3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2019/04/10 01:53:42
7  // Design Name:
```

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


module FIFO(
    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
```

```
57      if (rst)
58      begin
59        for (i = 0 ; i < 8 ; i = i + 1)
60        Q[i] = 0;
61
62        front = 0;
63        rear = 0;
64        avail = 0;
65        empty = 1;
66        full = 0;
67      end
68      else
69      begin
70        if (en_in)
71        begin
72            if (!avail[front])
73            begin
74                Q[front] = in;
75                avail[front] = 1;
76                front = (front + 1) % 8;
77            end
78
79            out = in;
80        end
81
82        if (en_out)
83        begin
84            if (avail[rear])
85            begin
86                out = Q[rear];
87                Q[rear] = 0;
88                avail[rear] = 0;
89                rear = (rear + 1) % 8;
90            end
91
92        end
93
94        if (avail == 8'b11111111)
95        full = 1;
96        else
97        full = 0;
98
99        if (avail == 8'b00000000)
100       empty = 1;
101       else
102       empty = 0;
103     end
104   end
105
106
107   reg [3:0] num;
```

```
108     wire [6:0] seg0[7:0];
109     reg [12:0] count ;
110     reg clk1;
111 
112     BCD27 B0(Q[0],seg0[0]);
113     BCD27 B1(Q[1],seg0[1]);
114     BCD27 B2(Q[2],seg0[2]);
115     BCD27 B3(Q[3],seg0[3]);
116     BCD27 B4(Q[4],seg0[4]);
117     BCD27 B5(Q[5],seg0[5]);
118     BCD27 B6(Q[6],seg0[6]);
119     BCD27 B7(Q[7],seg0[7]);
120 
121     initial
122     begin
123     an<=8'b11111111;
124     clk1<=0;
125     num<=0;
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];
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];
```

```
            if (!avail[6])
            seg<=7'b1111111;

            if (rear == 6)
            dp <= 0;
            else
            dp <= 1;
          end
        7:begin
            an <=8'b01111111;
            seg<=seg0[7];
            if (!avail[7])
            seg<=7'b1111111;

            if (rear == 7)
            dp <= 0;
            else
            dp <= 1;
          end
    endcase
endmodule
```

Listing 9: FIFO_tb.v

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

// Company:
// Engineer:
//
// Create Date: 2019/04/10 01:54:04
// Design Name:
// Module Name: FIFO_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////


module FIFO_tb;
    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;
47
48      initial
49      begin
50        rst = 1;
51        #10 rst = 0;
52        #10 rst = 1;
53        #10 rst = 0;
54
55        #10 en_in <= 1;
56            in <= 9;
57        #10 en_in <= 1;
58            in <= 4;
59        #20 en_in <= 0;
60            en_out <= 1;
61
62      end
63
64      FIFO DUT (en_out,in,en_in,rst,clk,qclk,an,
65              seg,dp,out,empty,full,avail,front,
66              rear,Q,num,count);
67  endmodule
```