**中国科学技术大学计算机学院**

**《计算机组成原理实验》报告**



实验题目：寄存器堆与存储器及其应用

学生姓名：\_\_\_吴毅龙\_\_\_\_\_\_

学生学号：\_\_\_PB19111749\_\_

完成日期：\_\_2021/4/12\_\_\_

计算机实验教学中心制

2020年09月

【实验题目】

寄存器堆与存贮器及其应用

【实验目的】

▶掌握寄存器堆（Register File）和存储器的功能、时序及其应用

▶熟练掌握数据通路和控制器的设计和描述方法

【实验环境】  
FPGAOL: fpgaol.ustc.edu.cn（或 Nexys4 DDR）  
Vivado

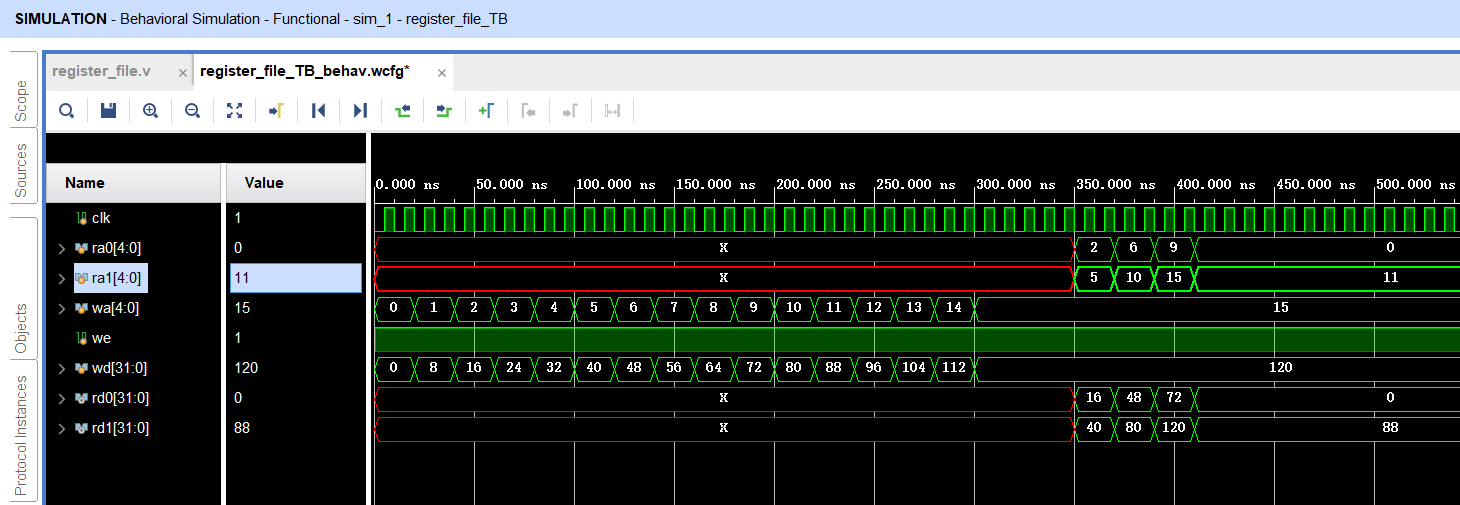
【实验练习】

1.行为方式参数化描述寄存器堆，功能仿真

如下为寄存器堆模块对应的代码：

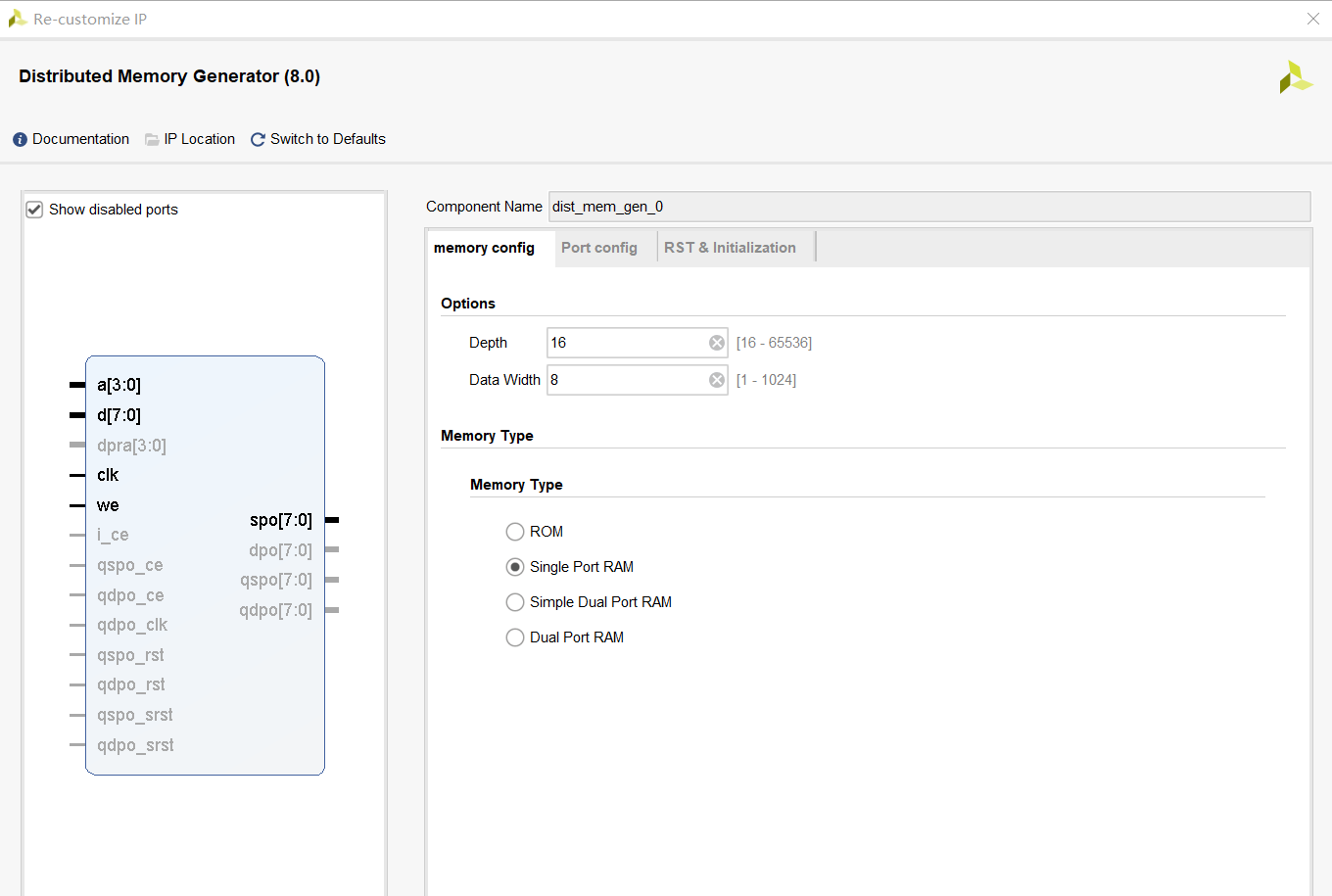
1. module register\_file
2. #(parameter WIDTH = 32)
3. (input clk,
4. input [4:0]ra0,
5. input [4:0]ra1,
6. input [4:0]wa,
7. input we,
8. input [WIDTH - 1:0]wd,
9. output [WIDTH - 1:0]rd0,
10. output [WIDTH - 1:0]rd1);
12. reg [WIDTH - 1:0] ram [15:0];
14. assign rd0 = ram[ra0];
15. assign rd1 = ram[ra1];
16. always@(posedge clk)
17. begin
18. **if**(we)
19. ram[wa] <= wd;
20. end
21. endmodule

如下为仿真结果：

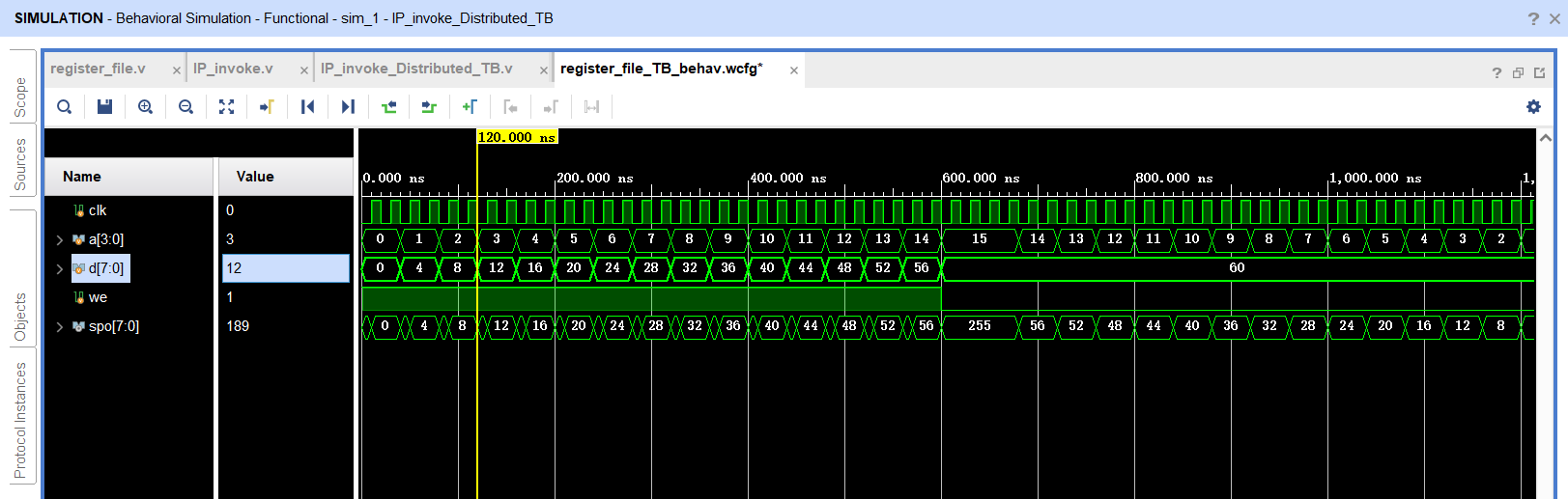


2.IP例化分布式和块式16x8位单端口RAM，功能仿真和对比

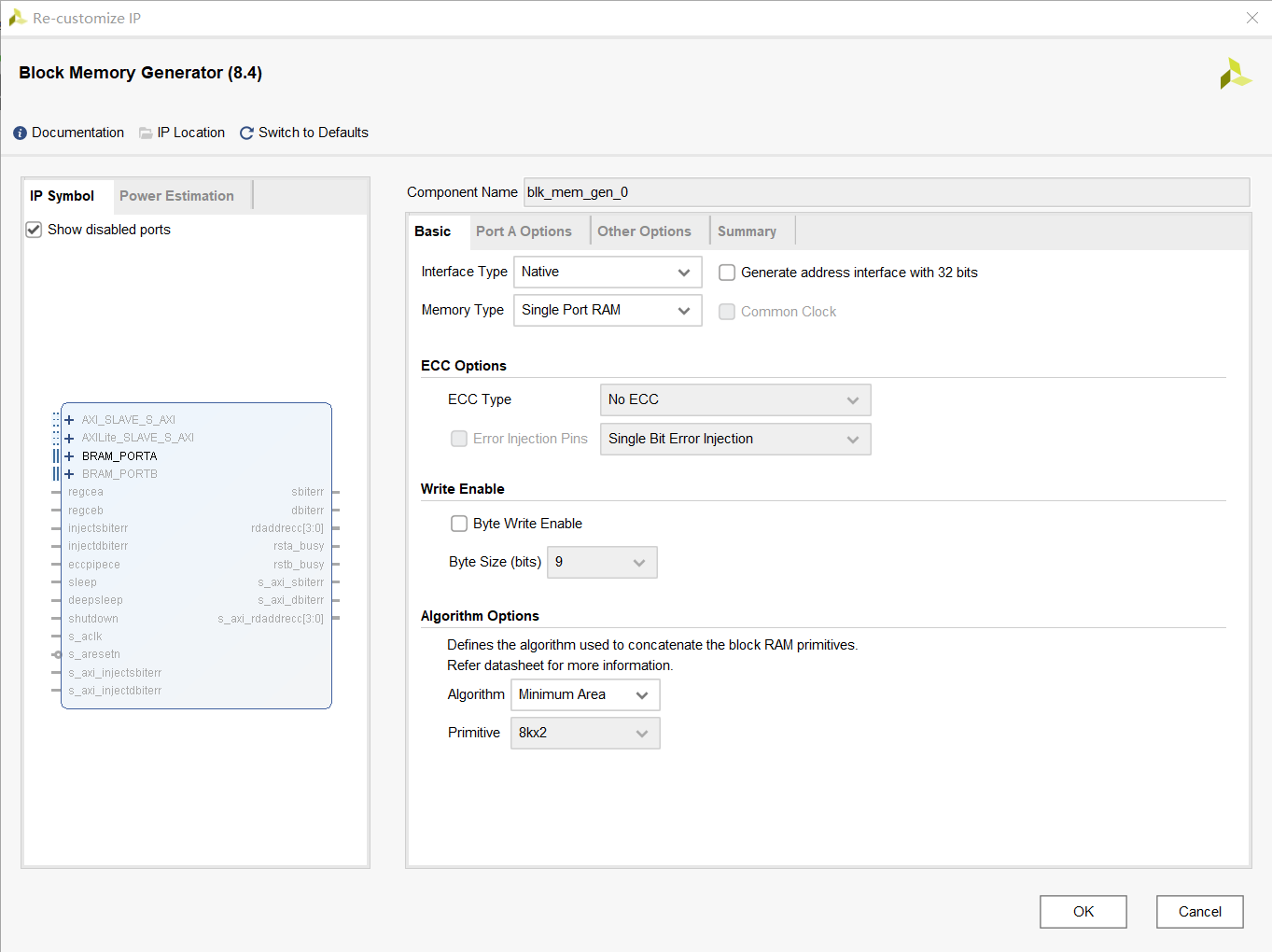
分布式RAM例化与功能仿真：



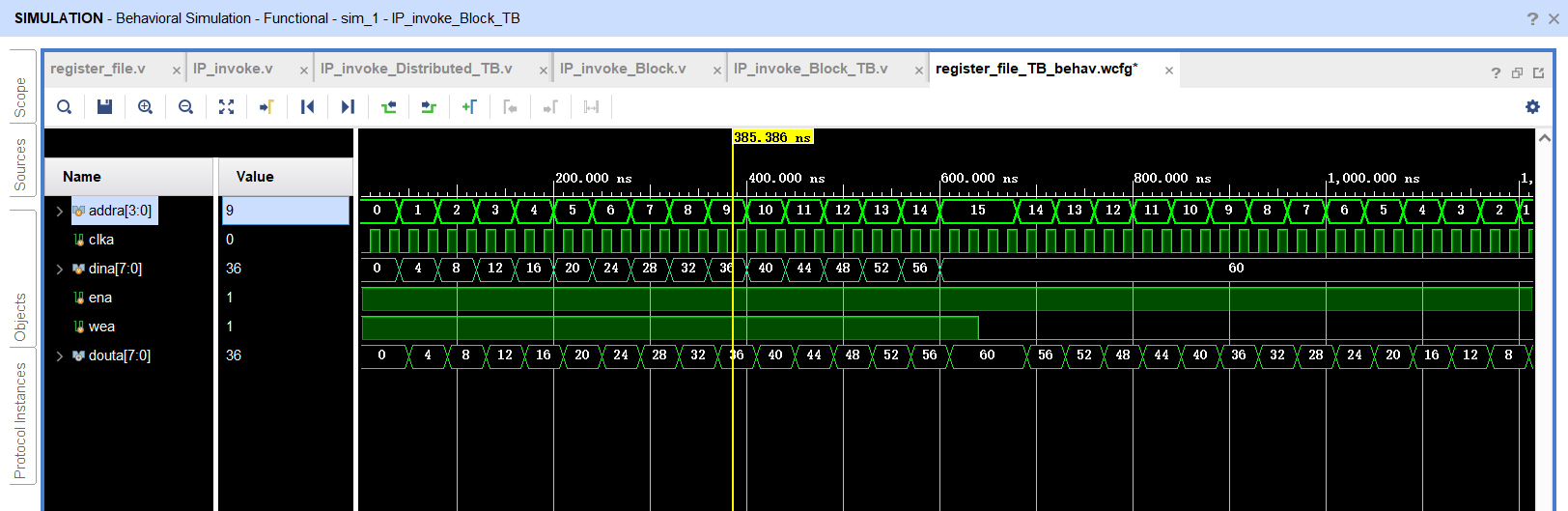
1. module IP\_invoke\_Distributed(
2. input clk,
3. input [3:0]a,
4. input [7:0]d,
5. input we,
6. output [7:0]spo);
7. dist\_mem\_gen\_0 IP\_invoke\_Distributed(
8. .a(a),
9. .d(d),
10. .clk(clk),
11. .we(we),
12. .spo(spo));
13. endmodule
14. module IP\_invoke\_Distributed\_TB();
15. reg clk;
16. reg [3:0]a;
17. reg [7:0]d;
18. reg we;
19. wire [7:0]spo;
20. IP\_invoke\_Distributed IP\_invoke\_Distributed(clk, a, d, we, spo);
21. initial
22. begin
23. clk = 0; a = 0; we = 1; d = 0;
24. #600 we = 0;
25. end
26. always #10 clk = ~clk;
27. initial
28. begin
29. repeat(15)
30. begin
31. #40 a = a + 1;
32. d = d + 4;
33. end
34. #40
35. repeat(15)
36. #40 a = a - 1;
37. end
38. endmodule



块式RAM例化与功能仿真：

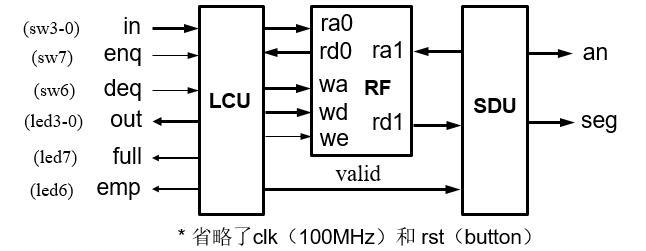


1. module IP\_invoke\_Block(
2. input [3:0]addra,
3. input clka,
4. input [7:0]dina,
5. input ena,
6. input wea,
7. output [7:0]douta);
8. blk\_mem\_gen\_0 IP\_invoke\_Block(
9. .addra(addra),
10. .clka(clka),
11. .dina(dina),
12. .douta(douta),
13. .ena(ena),
14. .wea(wea));
15. endmodule
16. module IP\_invoke\_Block\_TB();
17. reg [3:0]addra;
18. reg clka;
19. reg [7:0]dina;
20. reg ena;
21. reg wea;
22. wire [7:0]douta;
23. IP\_invoke\_Block IP\_invoke\_Block(addra, clka, dina, ena, wea, douta);
25. initial
26. begin
27. clka = 0; ena = 1; wea = 1; addra = 0; dina = 0;
28. #640 ena = 1; wea = 0;
29. end
31. always #10 clka = ~clka;
32. initial
33. begin
34. repeat(15)
35. begin
36. #40 addra = addra + 1;
37. dina = dina + 4;
38. end
39. #40
40. repeat(15)
41. #40 addra = addra - 1;
42. end
44. endmodule



3.设计FIFO队列电路的数据通路和控制器，结构化方式描述数据通路，Moore型FSM描述控制器

FIFO队列电路的数据通路：



//封装FIFO模块

1. module FIFO(
2. input enq,        //入队使能，一次有效仅允许操作一项数据 sw7
3. input deq,        //出队使能，一次有效仅允许操作一项数据 sw6
4. input [3:0]in,    //入队数据 sw3-0
5. input rst,        //复位信号 button
6. input clk,
8. output [3:0]out,  //出队数据 led3-0
9. output emp,       //队空信号 led6
10. output full,      //队满信号 led7
11. output [2:0]an,   //数码管选择
12. output [3:0]seg); //数码管数据
14. wire [2:0]ra0,ra1,wa;
15. wire [3:0]rd0,rd1,wd;
16. wire we;
17. wire [7:0]valid;  //有效标志
18. LCU LCU(.clk(clk),.rst(rst),.in(in),.enq(enq),.deq(deq),.out(out),.full(full),.emp(emp),.ra0(ra0),.rd0(rd0),.wa(wa),.wd(wd),.we(we),.valid(valid));
19. RF RF(.clk(clk),.ra0(ra0),.rd0(rd0),.wa(wa),.wd(wd),.we(we),.ra1(ra1),.rd1(rd1));
20. SDU SDU(.clk(clk),.ra1(ra1),.rd1(rd1),.an(an),.seg(seg),.valid(valid));
22. endmodule

//RF模块

1. module RF(
2. input we,                     //写使能
3. input clk,
4. input [2:0] wa, ra0, ra1,     //写端口地址，读端口0地址，读端口1地址
5. input [3:0] wd,               //写端口数据
6. output [3:0] rd0, rd1         //读端口0数据，读端口1数据
8. reg [3:0] regfile[0:7];
9. assign rd0 = regfile[ra0];
10. assign rd1 = regfile[ra1];
11. always @(posedge clk)
12. begin
13. **if**(we) regfile[wa] <= wd;
14. end
15. module

//LCU模块

1. module LCU(
2. input clk,
3. input rst,
4. input [3:0]in,            //入队数据
5. input enq, deq,
6. input [3:0]rd0,           //读端口0数据
8. output reg [3:0]out,      //出队列数据
9. output reg full, emp,     //队空与队满的标志
10. output [2:0]ra0,          //读端口0地址
11. output [2:0]wa,           //写端口地址
12. output reg [3:0]wd,       //写端口数据
13. output reg we,            //写使能信号
14. output reg [7:0]valid);   //有效标志，第i位对应Ri的状态
16. reg [2:0]cs, ns;          //当前状态，下一状态
17. reg [2:0]tail;            //队尾，指向入队列数据
18. reg [2:0]head;            //队头，指向出队列数据
19. assign ra0 = head;
20. assign wa = tail-1;
22. always@(posedge clk)
23. **if** (rst)   cs <= 3'b000;
24. **else**   cs <= ns;

27. always@(posedge clk)
28. **case**(cs)
29. 3'b000:begin      //初始化
30. valid <= 8'b0;
31. tail <= 0;
32. head <= 0;
33. full <= 0;
34. emp <= 0;
35. we <= 0;
36. end
37. 3'b001:begin      //入队一个数据
38. wd <= in;
39. valid[tail] <= 1;
40. tail <= (tail + 1) % 8;
41. full <= ((tail + 1) % 8 == head) & !emp;
42. emp <= 0;
43. we <= 1;
44. end
45. 3'b010:begin       //出队一个数据
46. out <= rd0;
47. valid[head] <= 0;
48. head <= (head + 1) % 8;
49. emp <= (tail == (head + 1)) & !full;
50. full <= 0;
51. we <= 0;
52. end
53. **default**:
54. we <= 0;
55. endcase

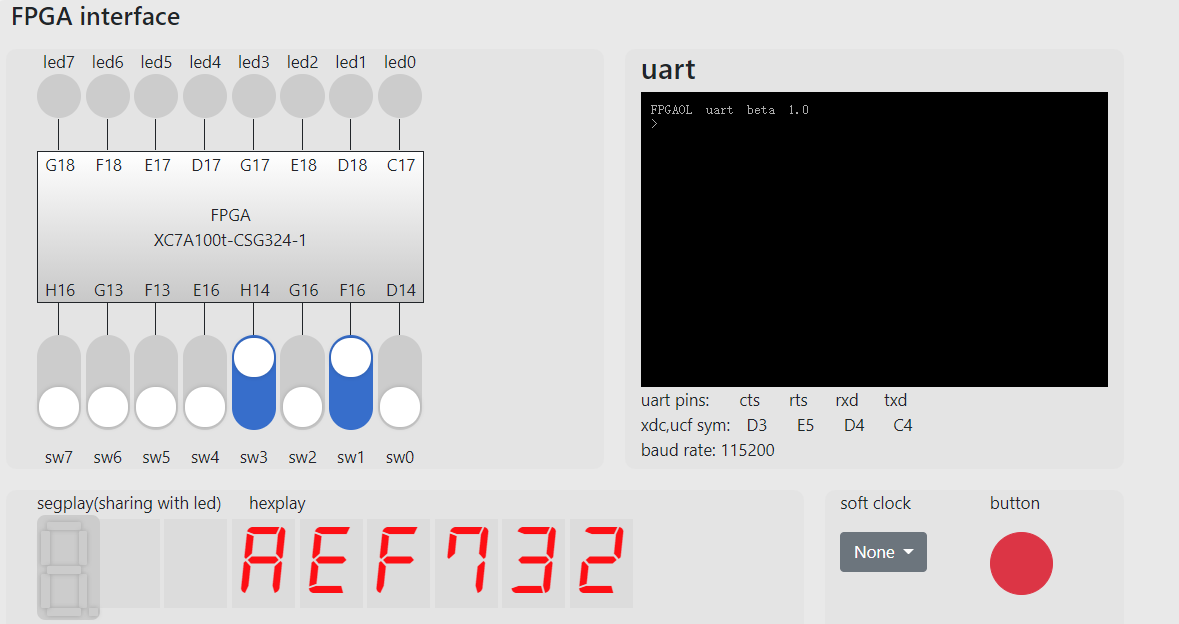
58. always@(\*)
59. begin
60. **case** (cs)
61. 3'b000:begin
62. **if**(enq & !full)      //当入队信号有效且队列不满时可以入队
63. ns = 3'b001;     //下一状态转到数据入队状态
64. **else** **if**(deq &! emp)  //当出队信号有效且队列不空时可以出队
65. ns = 3'b010; //下一状态转到数据出队状态
66. **else** **if**((enq & full)|(deq & emp))  //当入队信号有效且队列满或当出队信号有效且队列空时
67. ns = 3'b011;
68. **else** ns = 3'b000;
69. end
70. 3'b001:ns = 3'b011;
71. 3'b010:ns = 3'b011;
72. 3'b011:begin
73. **if**(!enq &! deq)
74. ns = 3'b100;
75. **else**
76. ns = 3'b011;
77. end
78. 3'b100:begin
79. **if**((enq & full)|(deq & emp))
80. ns = 3'b011;
81. **else** **if**(enq & !full)
82. ns = 3'b001;
83. **else** **if**(deq  & !emp)
84. ns = 3'b010;
85. **else** ns = 3'b100;
86. end
87. **default**:
88. ns = 3'b000;
89. endcase
90. end
92. endmodule

//SDU模块

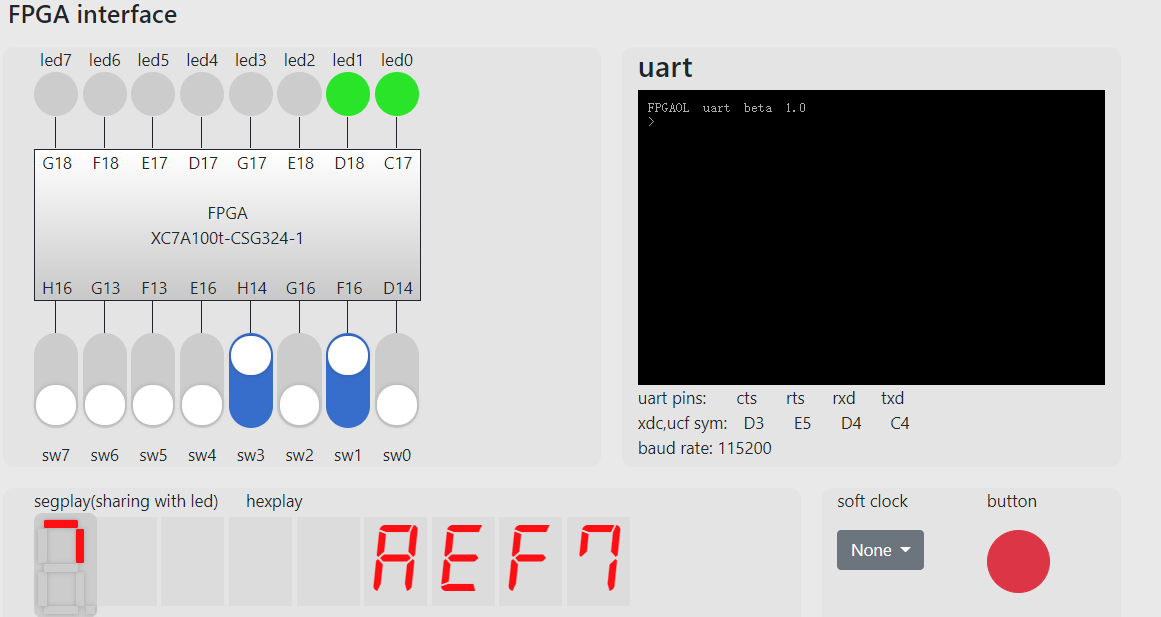
1. module SDU(
2. input clk,
3. input [3:0]rd1,       //读端口1数据
4. input [7:0]valid,     //有效标志
5. output reg [2:0]ra1,  //读端口1地址
6. output reg[2:0]an,    //数码管选择
7. output reg[3:0]seg);  //数码管数据
8. reg [9:0]cnt;
10. initial begin
11. ra1 = 0;
12. cnt = 0;
13. end
15. always@(posedge clk)
16. begin
17. cnt <= cnt + 1;
18. **if**(cnt == 0)begin
19. ra1 <= ra1 + 1;
20. **if**(valid[ra1])begin
21. an <= ra1;
22. seg <= rd1;
23. end
24. **else** begin
25. an <= an;
26. seg <= seg;
27. end
28. end
29. end
30. endmodule

4.FIFO队列电路下载至FPGA中测试

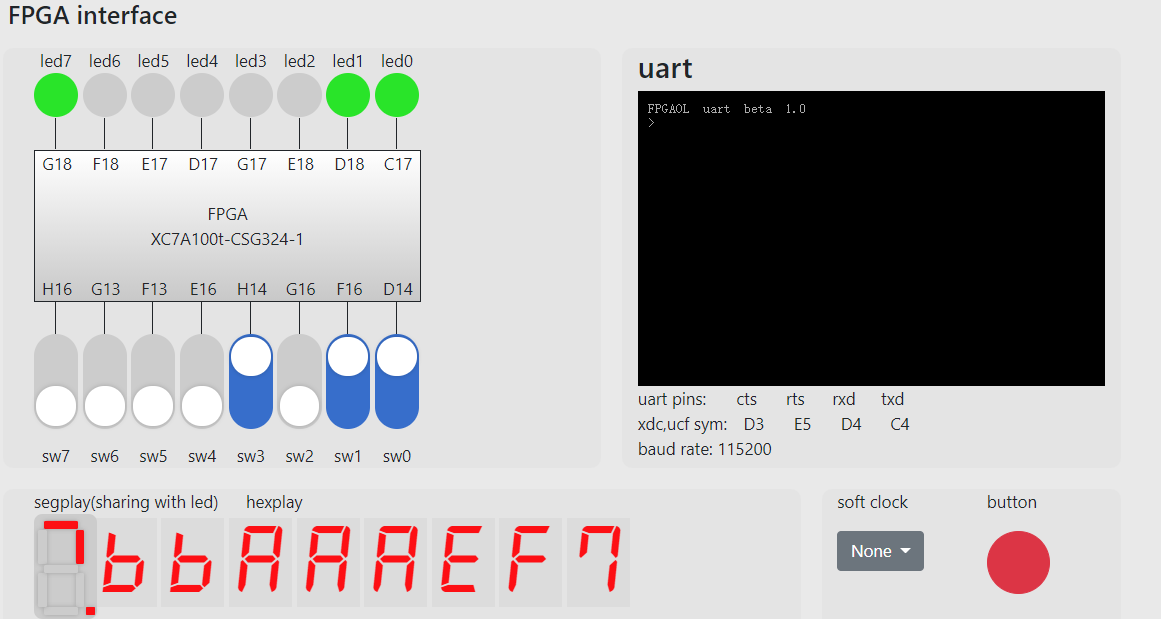
数据入队：



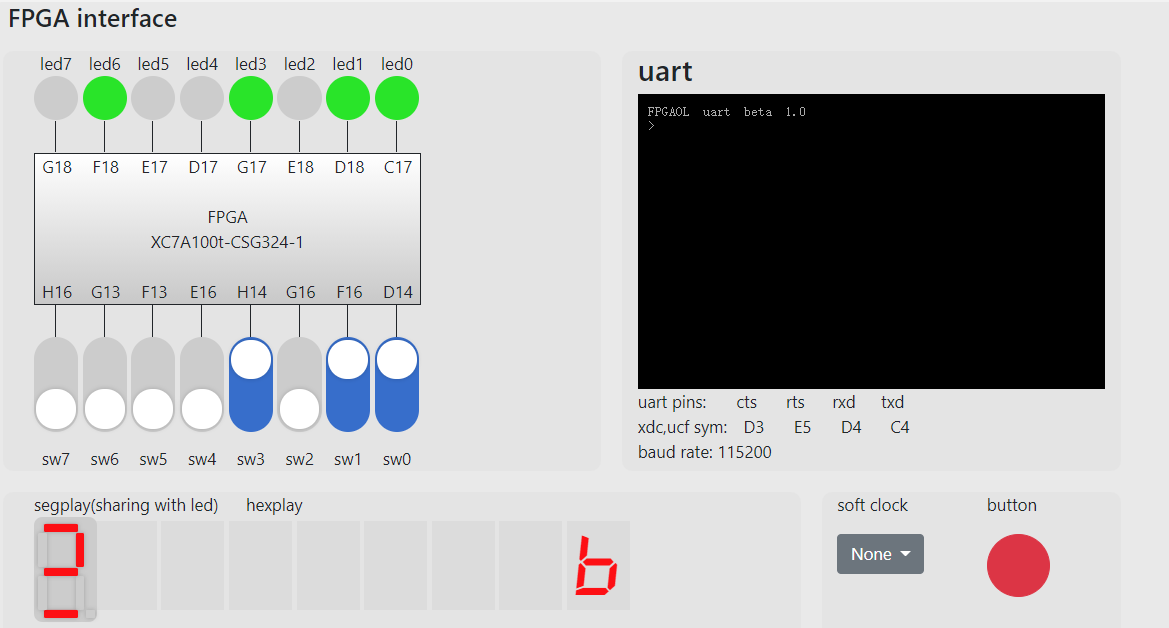
数据出队：



队满指示：



队空指示：



【总结与思考】

通过本次实验，我利用FPGAOL平台进行实验，并例化使用IP核，实现了FIFO队列。本次实验难度不大，根据实验说明的指导就可以完成实验操作，实验题目的难度层层递进，有基础操作的考核，也有所学知识的综合，难易结合，既有复习又有思考，让所学在实践中得以运用，加深了我对逻辑电路知识的理解。希望今后实验可以保持本次实验中详细实验指导描述的优点，辅助完成每项试验内容。