同济大学计算机系

数字逻辑课程实验报告

学　　号 **2151140**

姓　　名 王谦

专　　业 信息安全

授课老师 郭玉臣

一、实验内容

在本次实验中，我们将使用 Verilog HDL 语言实现 D 触发器、JK 触发器以及 PC 寄存器的设计和仿真。

二、硬件逻辑图

（实验步骤中要求用 logisim 画图的实验，在该部分给出 logisim 原理图，否则该部分在实验报告中不用写）

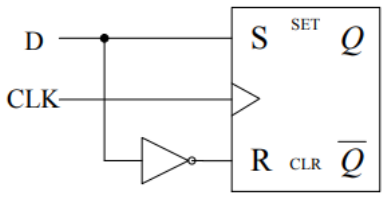
三、模块建模

（该部分要求对实验中建模的所有模块进行功能描述，并列出各模块建模的 verilog 代码）

1. 同步复位 D 触发器，当时钟信号上升沿时，RST_n 为低电平时复位，否则执行 D 触发器功能。

a）同步复位 D 触发器

```
module Synchronous_D_FF(
    input CLK,          //时钟信号，上升沿有效
    input D,            //输入信号 D
    input RST_n,        //复位信号，低电平有效
    output reg Q1,      //输出信号 Q
    output reg Q2       //输出信号 Q̄
);
```



输入		输出		说明
D	CLK	Q	$\bar{Q}$	
1	↑	1	0	置位（存 1）
0	↑	0	1	复位（存 0）

图 6.6.2 D 触发器逻辑图及其功能表

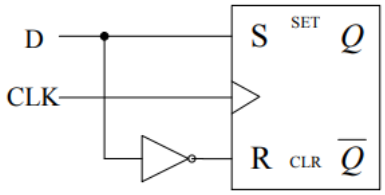
```
module Synchronous_D_FF(CLK,D,RST_n,Q1,Q2);
input CLK;
input D,RST_n;
output reg Q1,Q2;
```

```
always@(posedge CLK)
begin
if (~RST_n)
      begin Q1 = 1'b0;
      Q2 = !Q1; end
else
      begin Q1 = D;
      Q2 = !Q1; end
end
endmodule
```

2．异步复位 D 触发器，类似于 1，但是复位不再被时钟信号约束

b）异步复位 D 触发器

```
module Asynchronous_D_FF(
      input CLK,          //时钟信号，上升沿有效
      input D,            //输入信号 D
      input RST_n,        //复位信号，低电平有效
      output reg Q1,      //输出信号 Q
      output reg Q2       //输出信号 Q̄
);
```



输入		输出		说明
D	CLK	Q	$\bar{Q}$	
1	↑	1	0	置位（存 1）
0	↑	0	1	复位（存 0）

图 6.6.2 D 触发器逻辑图及其功能表

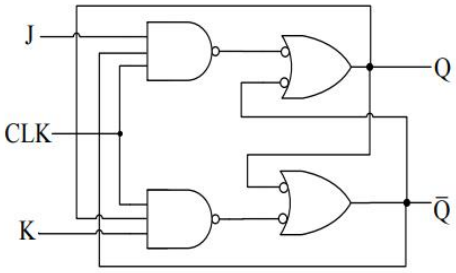
```
module Asynchronous_D_FF(CLK,D,RST_n,Q1,Q2);
input CLK;
input D,RST_n;
output reg Q1,Q2;

always@(posedge CLK or posedge RST_n)
begin
if (~RST_n)
      begin Q1 = 1'b0;
      Q2 = !Q1; end
else
      begin Q1 = D;
```

Q2 = !Q1; end
end
endmodule

3. 异步复位 JK 触发器

c) JK 触发器

JK 触发器是一种广泛应用的触发器类型。字母 J 和 K 只表示它们是两个数据输入端符号，没有什么特别含义。JK 触发器与 SR 触发器在置位、复位方面的功能是相同的，不同之处在于，JK 触发器改进了 SR 触发器存在的不稳定状态。当 J=1，K=1 时，对每一个连续的时钟脉冲，触发器可改变成相反状态或计数状态，这种工作方式称为交替操作。图 6.6.3 所示为 JK 触发器的内部逻辑及其功能表。



输入			输出		说明
J	K	CLK	Q	$\overline{Q}$	
0	0	↑	Q^n	$\overline{Q^n}$	保持
0	1	↑	0	1	置 0
1	0	↑	1	0	置 1
1	1	↑	$\overline{Q^n}$	Q^n	交替

图 6.6.3 JK 触发器逻辑图及其功能表

c) 异步复位 JK 触发器

```
module JK_FF(
    input CLK,        //时钟信号，上升沿有效
    input J,          //输入信号 J
    input K,          //输入信号 K
    input RST_n,      //复位信号，低电平有效
    output reg Q1,    //输出信号 Q
    output reg Q2     //输出信号 Q̄
);
```

提示：上升沿触发使用 always @ (posedge clk)

module JK_FF(CLK,J,K,RST_n,Q1,Q2);
input CLK;
input J,K,RST_n;
output reg Q1,Q2;

always@(posedge CLK or posedge RST_n)
begin
if(RST_n)
    begin Q1 = (J && ~Q1 || ~K && Q1);

```
        Q2 = ~Q1; end
else
        begin Q1 = 1'b0;
        Q2 = ~Q1; end
end
endmodule
```

4.PC 寄存器

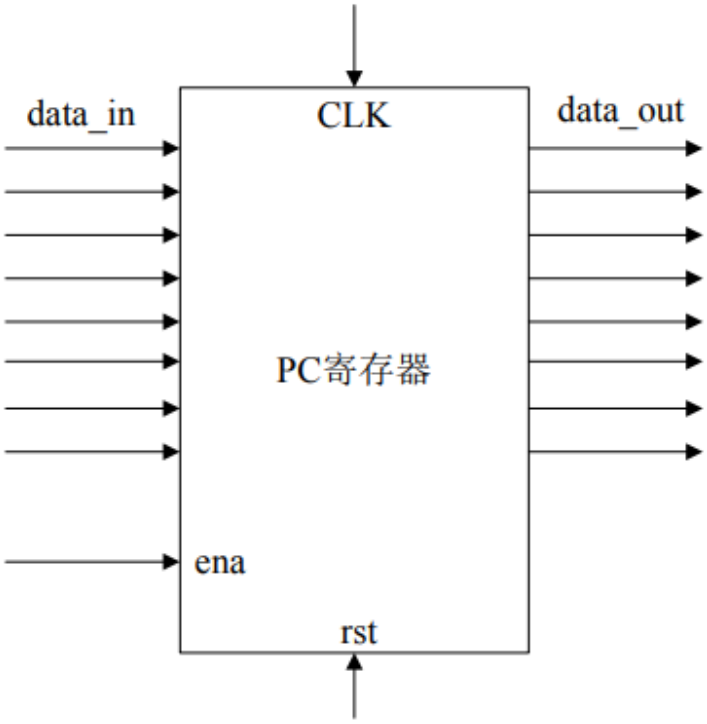


图 6.6.4 PC 寄存器功能图

```
module pcreg(
    input clk,    //1 位输入，寄存器时钟信号，上升沿时为 PC 寄存器赋值
    input rst,    //1 位输入，异步重置信号，高电平时将 PC 寄存器清零
                  //注：当 ena 信号无效时，rst 也可以重置寄存器
    input ena,    //1 位输入,有效信号高电平时 PC 寄存器读入 data_in
                  //的值，否则保持原有输出
    input [31:0] data_in, //32 位输入，输入数据将被存入寄存器内部
    output reg [31:0] data_out   //32 位输出，工作时始终输出 PC
                                 //寄存器内部存储的值
);
```

```
module pcreg(clk,rst,ena,data_in,data_out);
input clk, rst, ena;
input [31:0] data_in;
output [31:0] data_out;

reg [31:0] data = 32'b0;
always @(posedge clk or posedge rst)
    begin
        if(rst)
            begin
                data <= 32'h0000_0000;          //reset key
            end
        else
            begin
                if(ena) begin data <= data_in; end          //enable ,input
            end
    end
assign data_out = data;
endmodule
```

四、测试模块建模

（要求列写各建模模块的 test bench 模块代码）

1.

```
`timescale 1ns / 1ns
module Synchronous_D_FF_tb;
reg CLK;
reg D, RST_n;
wire Q1,Q2;

initial CLK = 0;
always #5 CLK = ~CLK;
initial
begin
    D = 1;
    RST_n = 1;
    #11 D = 0;
    #12 D = 1;
    #13
    RST_n = 0;
    #11 D = 0;
    #12 D = 1;
    #13
```

```
        RST_n = 1;
        #11 D = 0;
        #12 D = 1;
        #13
        RST_n = 0;
        #11 D = 0;
        #12 D = 1;
        #20 $stop;
end

Synchronous_D_FF
Synchronous_D_FF_init(
.CLK(CLK),
.D(D),
.RST_n(RST_n),
.Q1(Q1),
.Q2(Q2)
);
Endmodule
```

2.

```
`timescale 1ns / 1ns
module Asynchronous_D_FF_tb;
reg CLK;
reg D, RST_n;
wire Q1,Q2;

initial CLK = 0;
always #5 CLK = ~CLK;
initial
begin
        D = 1;
        RST_n = 1;
        #11 D = 0;
        #12 D = 1;
        #13
        RST_n = 0;
        #11 D = 0;
        #12 D = 1;
        #13
        RST_n = 1;
        #11 D = 0;
        #12 D = 1;
        #13
```

```
        RST_n = 0;
        #11 D = 0;
        #12 D = 1;
        #20 $stop;
end

Asynchronous_D_FF
Asynchronous_D_FF_init(
.CLK(CLK),
.D(D),
.RST_n(RST_n),
.Q1(Q1),
.Q2(Q2)
);
Endmodule
```

3.

```
`timescale 1ns / 1ns
module JK_FF_tb;
reg CLK;
reg J, K, RST_n;
wire Q1,Q2;

initial CLK = 0;
always #5 CLK = ~CLK;
initial
begin
        J = 0;K = 1;
        RST_n = 1;
        #11 J = 0;K = 0;
        #12 J = 1;K = 0;
        #13 J = 1;K = 1;
        #14
        RST_n = 0;
        #11 J = 0;K = 0;
        #12 J = 1;K = 0;
        #13 J = 1;K = 1;
        #14
        RST_n = 1;
        #11 J = 0;K = 0;
        #12 J = 1;K = 0;
        #13 J = 1;K = 1;
        #14
        RST_n = 0;
```

```
        #11 J = 0;K = 0;
        #12 J = 1;K = 0;
        #13 J = 1;K = 1;
        #14
        $stop;

end

JK_FF
JK_FF_init(
.CLK(CLK),
.J(J),
.K(K),
.RST_n(RST_n),
.Q1(Q1),
.Q2(Q2)
);
endmodule
```

4.

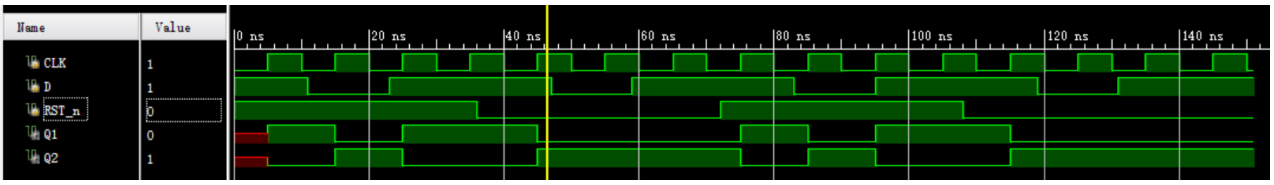
```
`timescale 1ns / 1ns
module pcreg_tb;
reg [31:0] data_in;
reg clk, rst, ena;
wire [31:0] data_out;

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

initial
begin
        rst = 0;ena = 1;
        data_in = 32'b10101010_10101010_10101010_10101010;
        #13 rst = 1; ena = 1;
        #13
        data_in = 32'b10111010_10111010_10111010_10111010;
        #13 rst = 0; ena = 0;
        #13 rst = 0; ena = 1;
        #13 rst = 1; ena = 0;
        #13
        $stop;
end

pcreg
```

```
pcreg_init(
.clk(clk),
.rst(rst),
.ena(ena),
.data_in(data_in),
.data_out(data_out)
);
endmodule
```

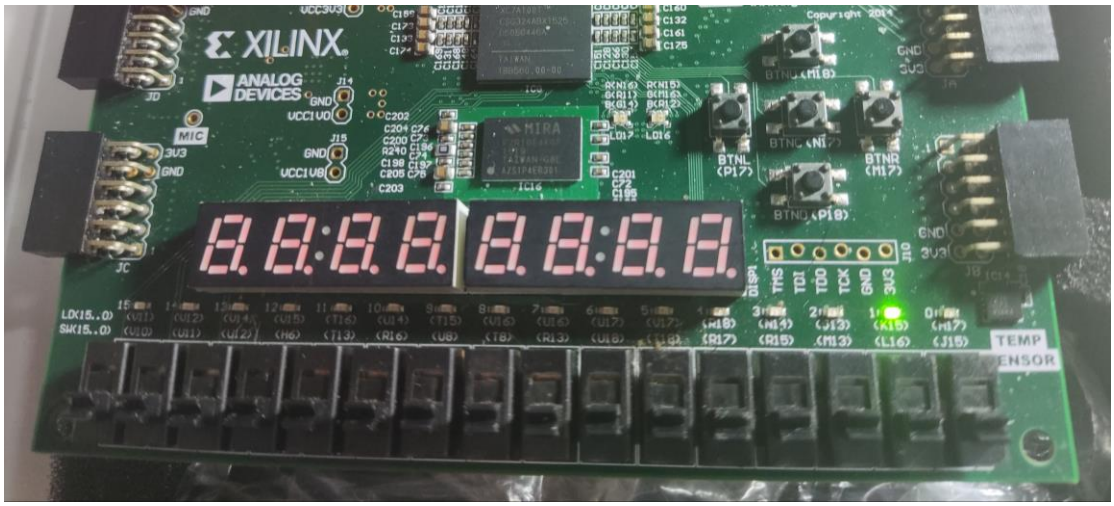
五、实验结果

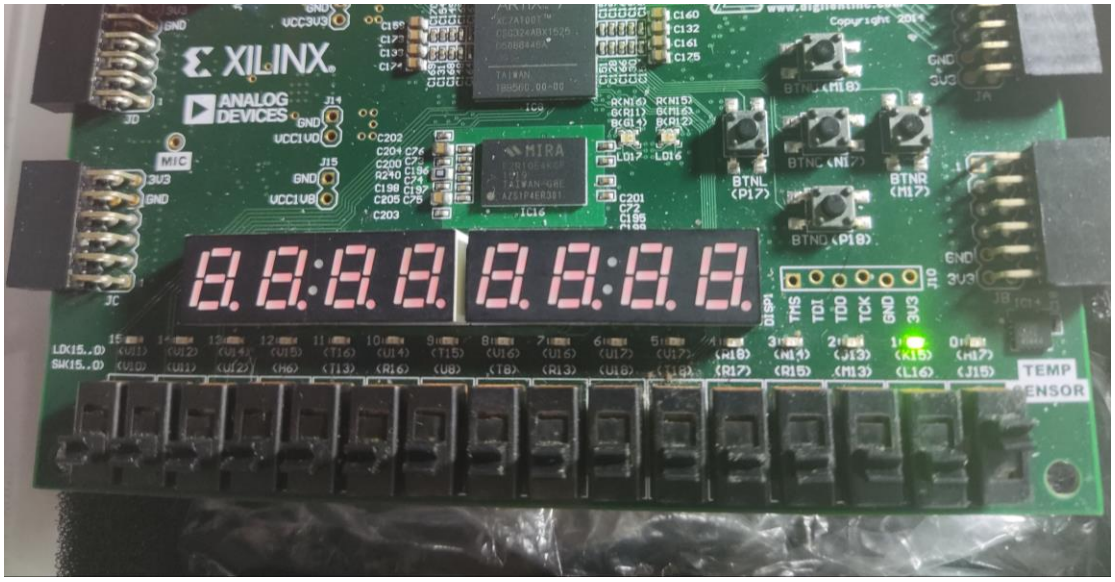
（该部分可截图说明，要求 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图（实验步骤中没有下板要求的实验，不需要下板贴图））

1.



初始 RST_n 为低电位有效，故 D 无法产生影响：

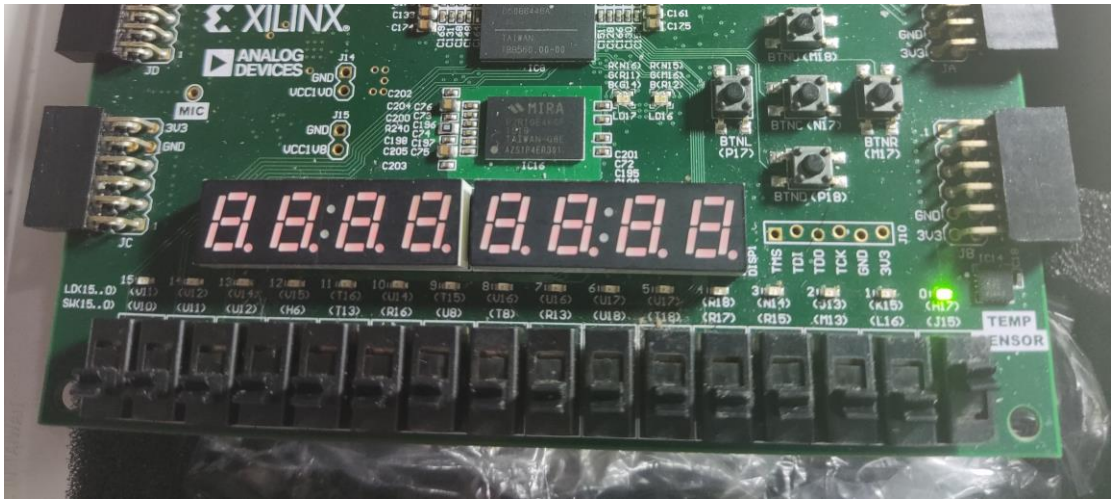




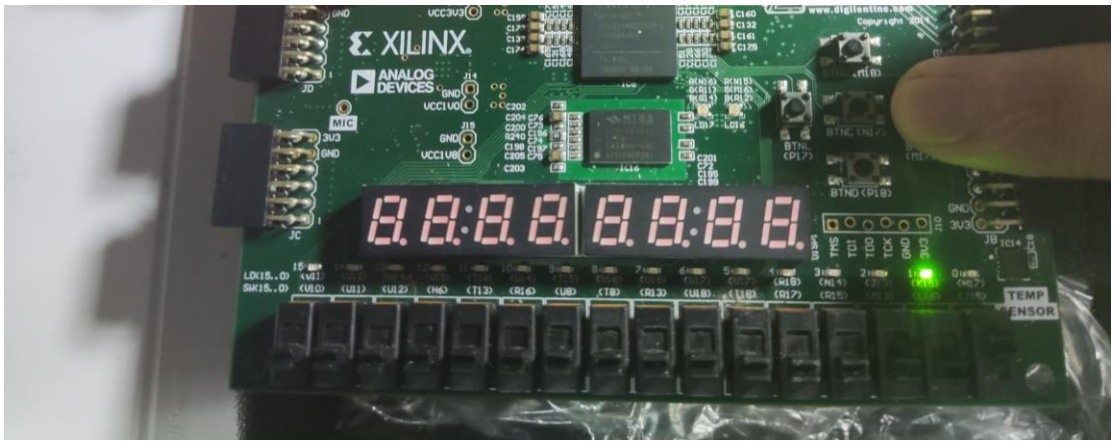
即使时钟信号到来也无法产生影响：

当 RST_n 处于高电位时，时钟上升沿时，D 有效：

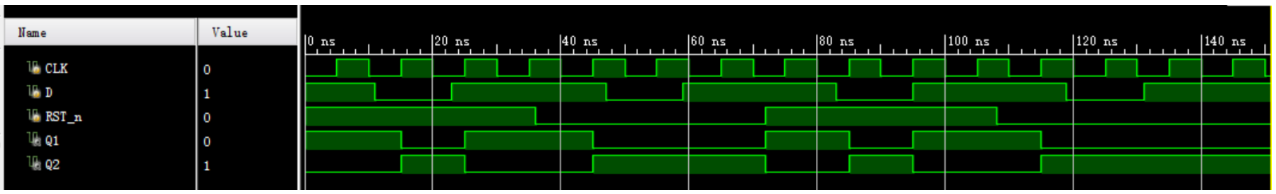
RST_n 回到低电位，但是时钟上升沿没有到来，输出维持之前的状态：



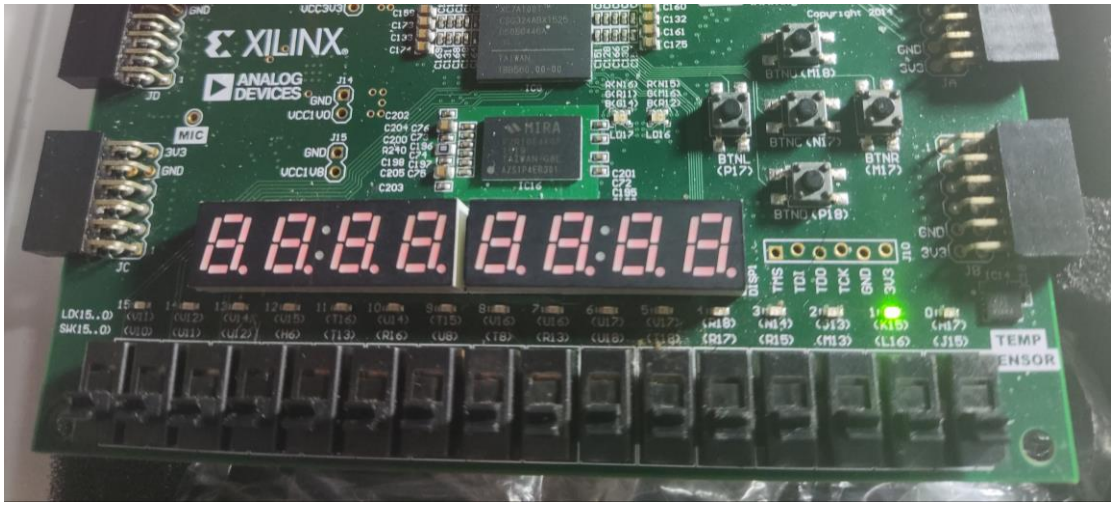
时钟上升沿到来，RST_n 生效，输出被重置：

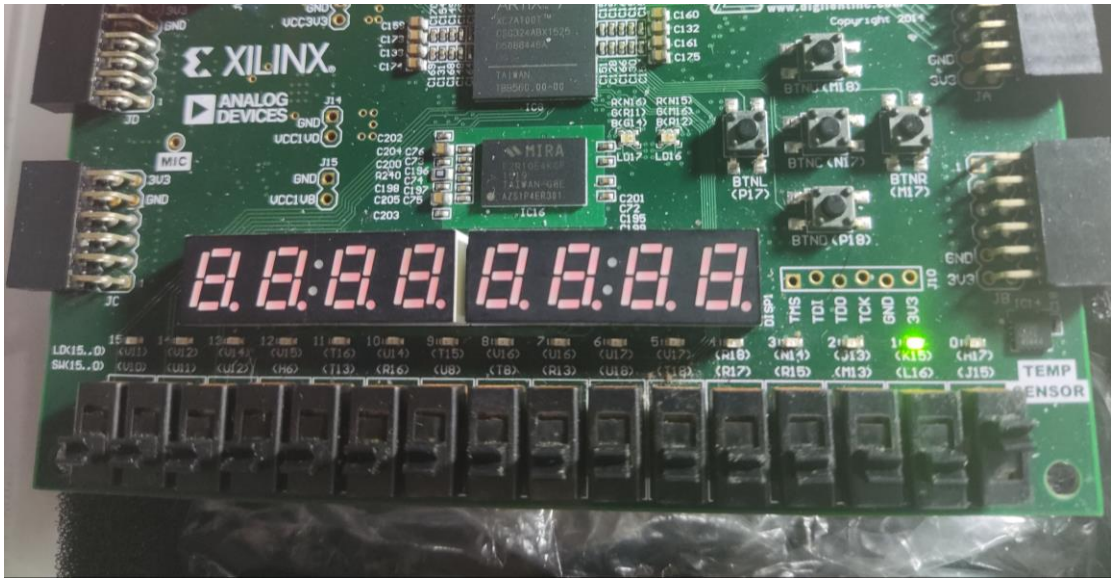


2.



初始 RST_n 为低电位有效，故 D 无法产生影响：

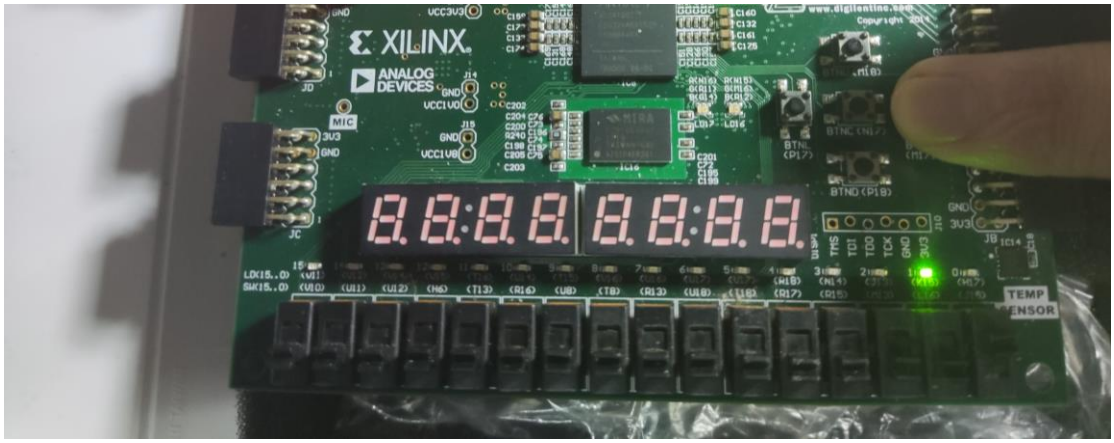




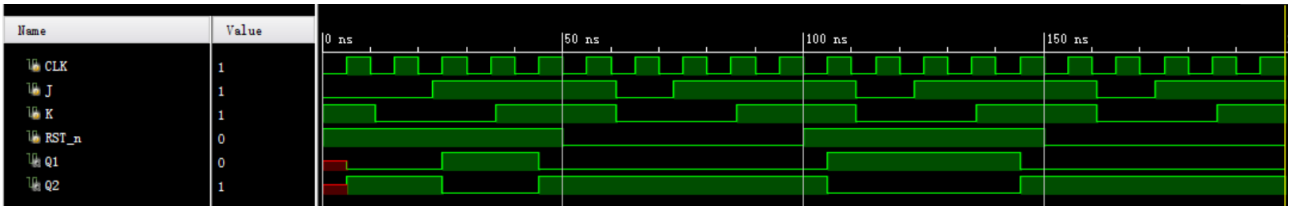
即使时钟信号到来也无法产生影响：

当 RST_n 处于高电位时，时钟上升沿时，D 有效：

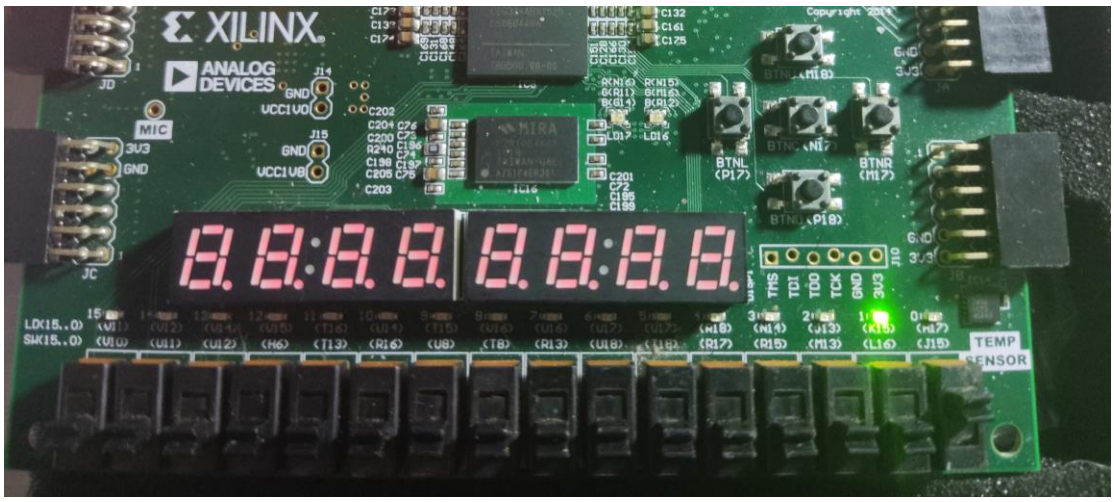
RST_n 回到低电位，虽然时钟上升沿没有到来，但由于异步复位，重置返回初始状态：



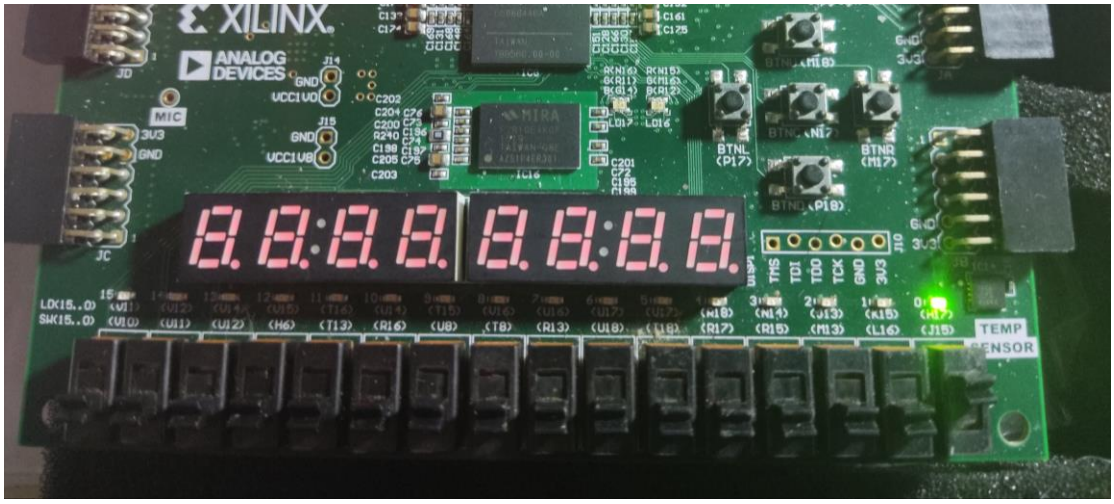
3.



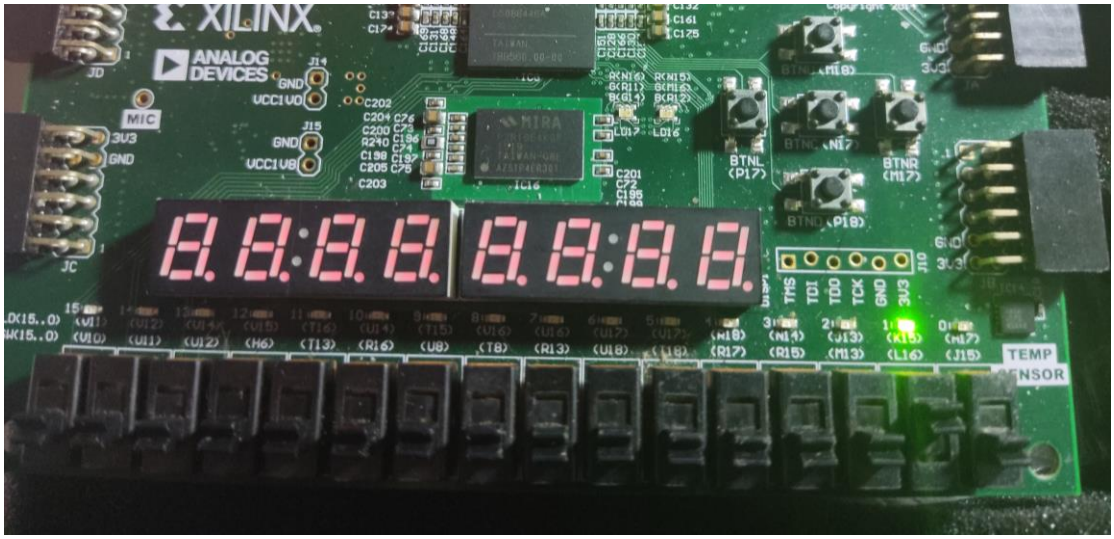
J=1,K=0，但 RST_n 低电位，处于复位状态：



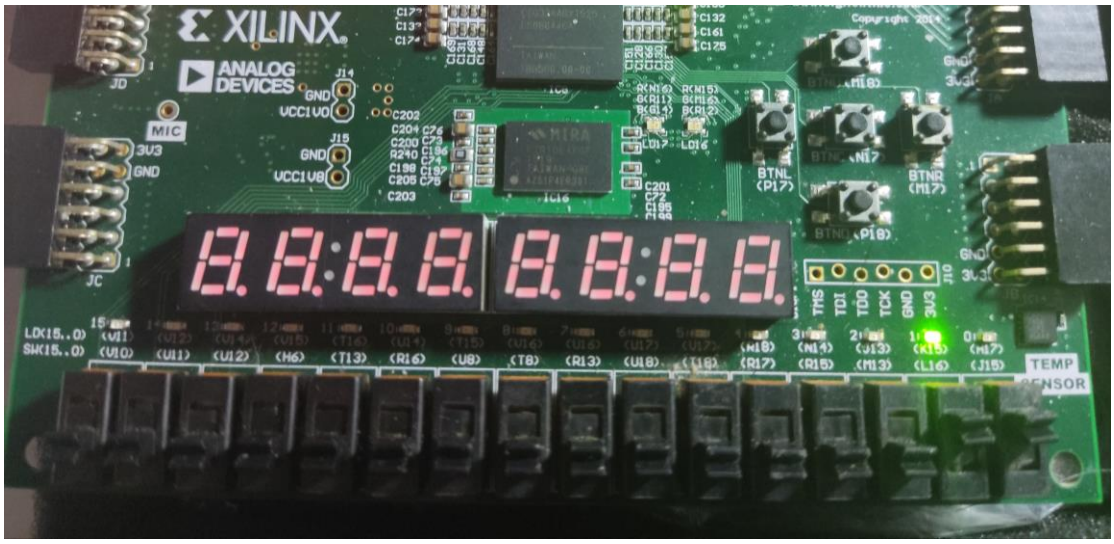
RST_n 变为高电位时，触发器生效：



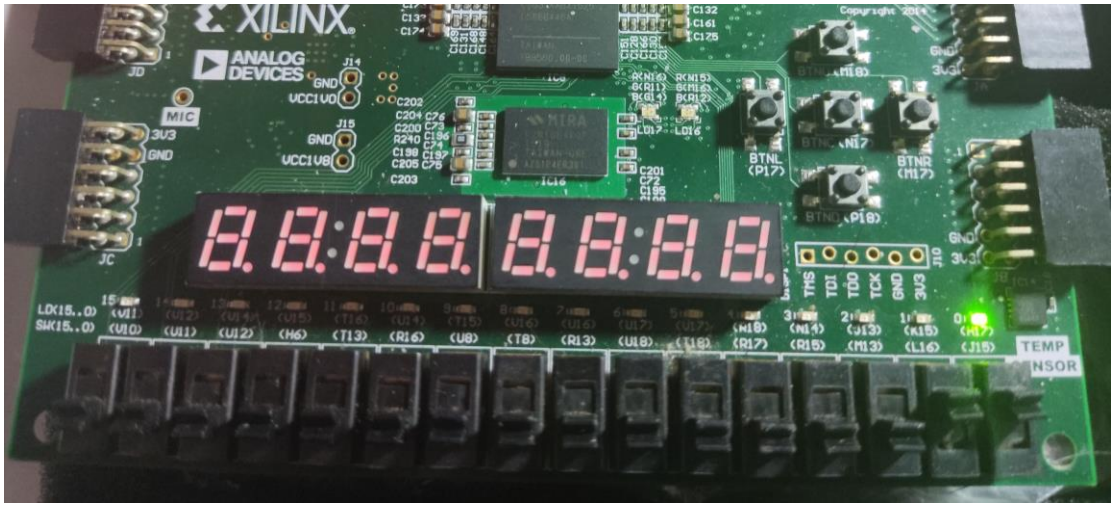
J=0,K=1，不论复位与否，始终保持 Q=0：



J=1,K=1，复位状态或生效状态由 Q=1 转变而来均为此种情况：



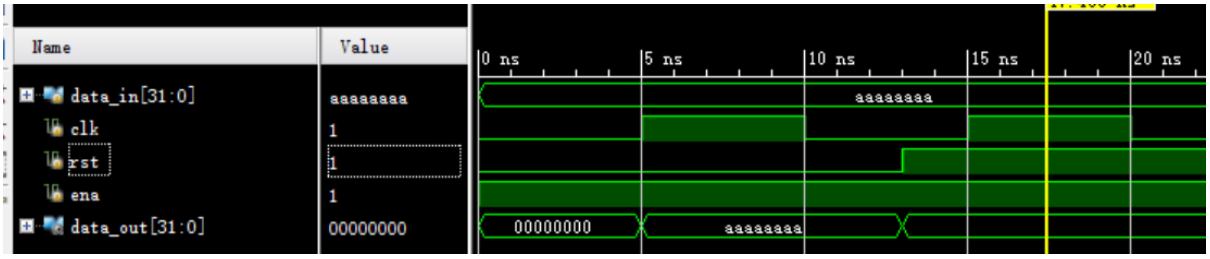
J=1,K=1，解除复位状态，由 Q=0 转变而来：

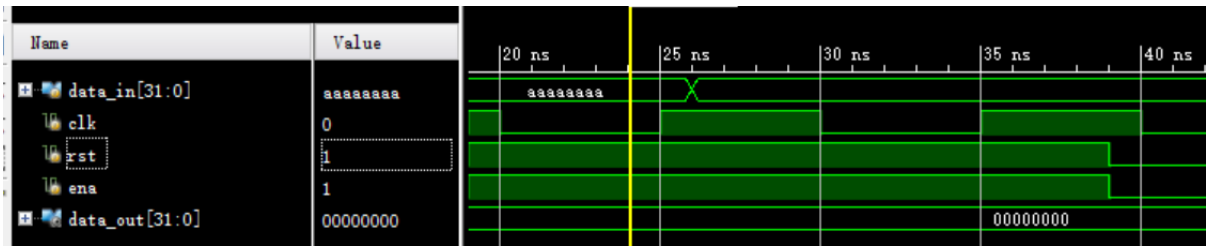


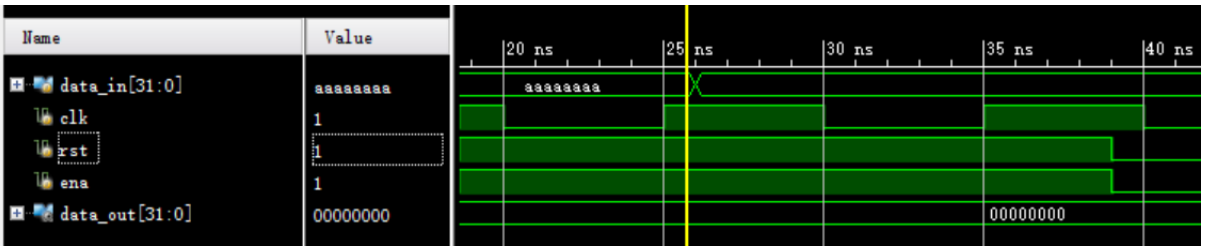
4.

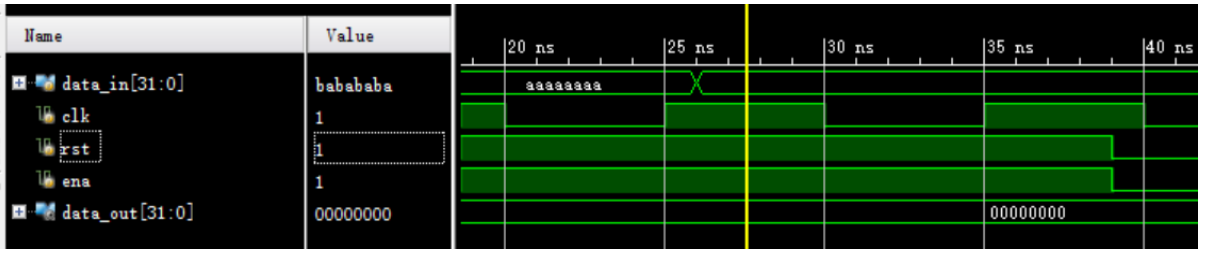
```
module pcreg(
    input clk,    //1 位输入，寄存器时钟信号，上升沿时为 PC 寄存器赋值
    input rst,    //1 位输入，异步重置信号，高电平时将 PC 寄存器清零
                  //注：当 ena 信号无效时，rst 也可以重置寄存器
    input ena,    //1 位输入,有效信号高电平时 PC 寄存器读入 data_in
                  //的值，否则保持原有输出
    input [31:0] data_in, //32 位输入，输入数据将被存入寄存器内部
    output reg [31:0] data_out   //32 位输出，工作时始终输出 PC
                                 //寄存器内部存储的值
);
```

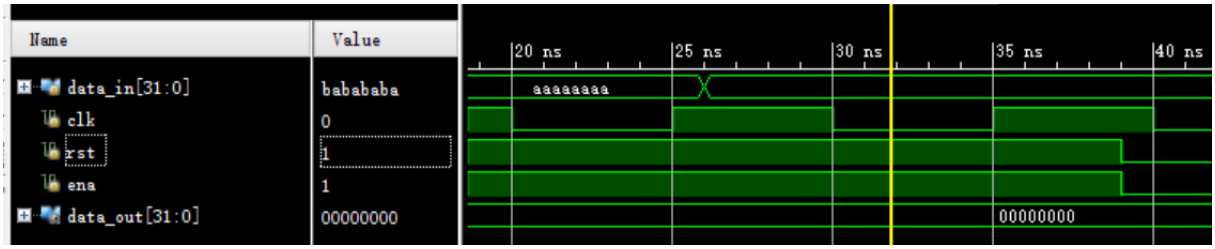
以下过程自初始开始，可以通过观察 value 得到变化情况：

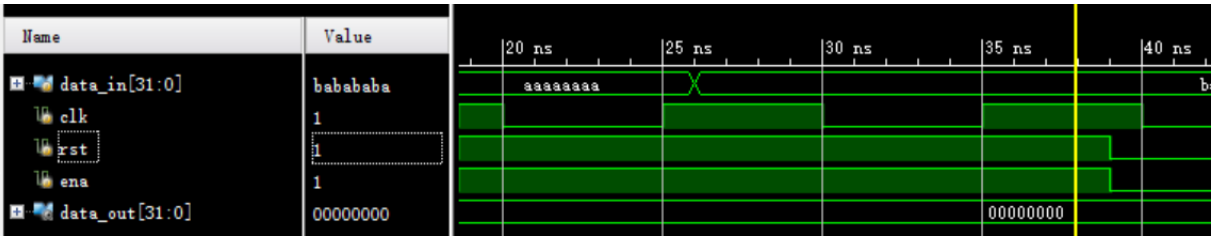

Name
Value
data_in[31:0]
aaaaaaaa
clk
1
rst
1
ena
1
data_out[31:0]
00000000
0 ns
5 ns
10 ns
15 ns
20 ns
aaaaaaaa
00000000
aaaaaaaa

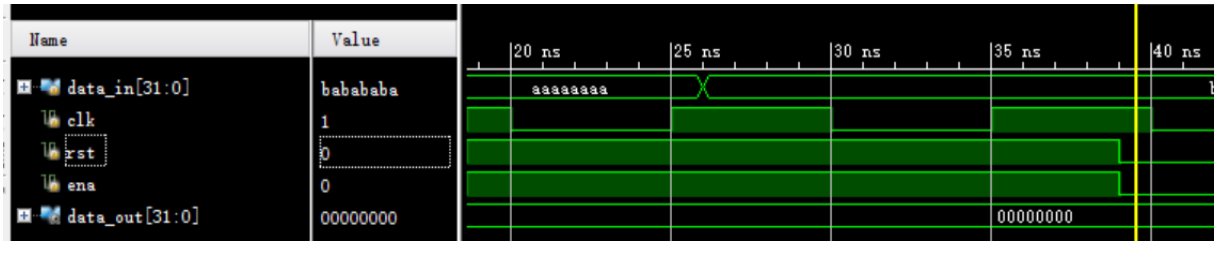

Name
Value
data_in[31:0]
aaaaaaaa
clk
0
rst
1
ena
1
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000

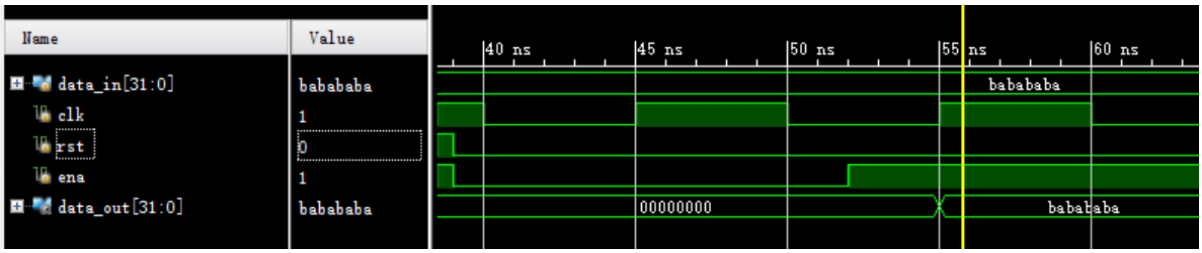

Name
Value
data_in[31:0]
aaaaaaaa
clk
1
rst
1
ena
1
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000

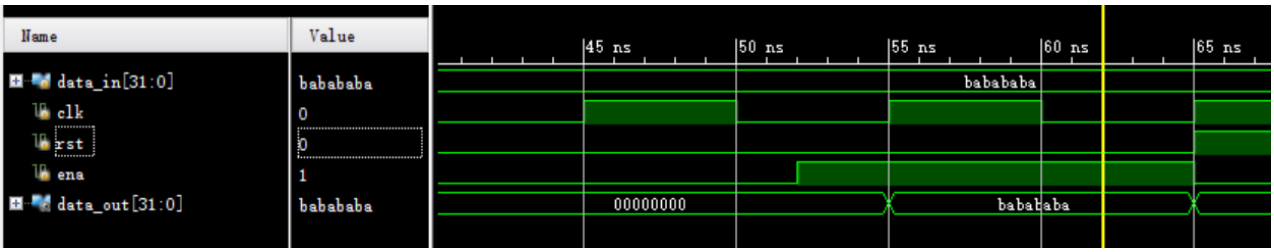

Name
Value
data_in[31:0]
babababa
clk
1
rst
1
ena
1
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000

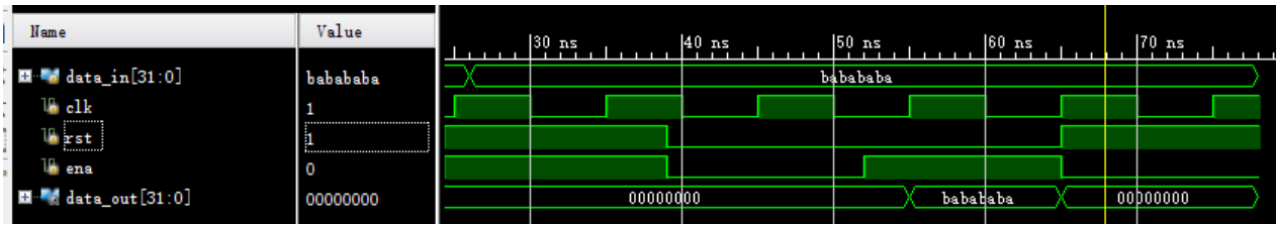

Name
Value
data_in[31:0]
babababa
clk
0
rst
1
ena
1
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000


Name
Value
data_in[31:0]
babababa
clk
1
rst
1
ena
1
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000


Name
Value
data_in[31:0]
babababa
clk
1
rst
0
ena
0
data_out[31:0]
00000000
20 ns
25 ns
30 ns
35 ns
40 ns
aaaaaaaa
00000000


Name
Value
data_in[31:0]
babababa
clk
1
rst
0
ena
1
data_out[31:0]
babababa
40 ns
45 ns
50 ns
55 ns
60 ns
babababa
00000000
babababa


Name
Value
data_in[31:0]
babababa
clk
0
rst
0
ena
1
data_out[31:0]
babababa
45 ns
50 ns
55 ns
60 ns
65 ns
babababa
00000000
babababa


Name
Value
data_in[31:0]
babababa
clk
1
rst
1
ena
0
data_out[31:0]
00000000
30 ns
40 ns
50 ns
60 ns
70 ns
babababa
00000000
babababa
00000000