数字电路与数字系统实验

EX04:触发器和锁存器

191220029 傅小龙
周一 5-6 节班
1830970417@qq.com
2020 年 9 月 14 日

目录

一、实验内容

1.1 实验要求

I)分析阻塞和非阻塞 RTL 视图和仿真结果

1. 新建工程，用阻塞赋值语句设计两个触发器；保存 Verilog 语言文件。

2. 在 Tools 栏，点击 Netlist Viewers 栏下的 RTL Viewer 查看生成的 RTL Schematic，看看在用阻塞赋值语句生成两个触发器的实际电路原理。

3. 新建另一个工程，用非阻塞赋值语句实现两个触发器，重复上述步骤，比较两种触发器实现方式在硬件电路实现上的异同。

II)设计一个同步清零和一个异步清零的 D 触发器

查阅资料，分析同步清零和异步清零的不同，并请在一个工程中设计两个触发器，一个是带有异步清零端的 D 触发器，而另一个是带有同步清零端的 D 触发器。

1.2 实验工具

软件环境：

设计、编译、仿真：Quartus Prime Version 17.1.0 Build 590 10/25/2017 SJ Lite Edition
DE10_Standard_SystemBuilder

硬件环境：DE-10 Standard 开发平台

FPGA 芯片：Cyclone V 5CSXFC6D6F31C6

二、实验过程

2.1 分析阻塞和非阻塞 RTL 视图和仿真结果

2.1.1 模型概述

使用 Verilog HDL 分别用阻塞式赋值和非阻塞式赋值的方式实现 2 个触发器.

相关原理：阻塞赋值语句在 Verilog HDL 中以“=”的形式体现，其作用为立即将表达式右侧的值赋给左侧. 而非阻塞赋值语句在 Verilog HDL 中以“<=”形式体现，其作用为将表达式右边的值在 always 块执行完毕后的一个极小延迟内赋值给左侧.

2.1.2 数字抽象

要设计的触发器的输入和输出的意义是相同的，故这里只给出非阻塞赋值触发器的输入输出：

I) 输入:

数据输入信号 data： 当时钟信号有效时该值将会被存储在触发器中.

时钟信号 clk： 与系统时钟连接，控制触发器的行为.

使能信号 en： 当 en 为 1 时触发器正常工作，否则不工作.

II) 输出:

out_unlock1：第一个触发器的数据输出.

out_unlock2：第二个触发器的数据输出.

下表\图给出了以上输入输出信号在 DE10 平台对应的信号：

	信号名称	DE-10平台信号
输入	data	SW[1]
	clk	CLOCK_50
	en	SW[0]
输出	out_unlock1	LEDR[0]
	out_unlock2	LEDR[1]

表 2-1-1:非阻塞赋值触发器输入输出信号与 DE10 平台信号对应关系

2.1.3 建立模型

实现思路：参照 exp04.pdf 4.2 节的例 1、例 2.

阻塞赋值触发器的 Verilog HDL 实现如下：

```
1  module blocing_assign(data, clk, en, out_lock1, out_lock2);
2      input data;
3      input clk;
4      input en;
5      output reg out_lock1;
6      output reg out_lock2;
7
8      always @(posedge clk)
9          if(en)
10             begin
11                 out_lock1 = data;
12                 out_lock2 = out_lock1;
13             end
14         else
15             begin
16                 out_lock1 = out_lock1;
17                 out_lock2 = out_lock2;
18             end
19 endmodule
```

阻塞赋值触发器的 Verilog HDL 实现如下：

```
module nonblocing_assign(data, clk, en, out_unlock1, out_unlock2);
    input data;
    input clk;
    input en;
    output reg out_unlock1;
    output reg out_unlock2;

    always @(posedge clk)
        if(en)
            begin
                out_unlock1 <= data;
                out_unlock2 <= out_unlock1;
            end
        else
            begin
                out_unlock1 <= out_unlock1;
                out_unlock2 <= out_unlock2;
            end
endmodule
```

2.1.4 分析/综合

分析/综合实验成功，如下图所示：

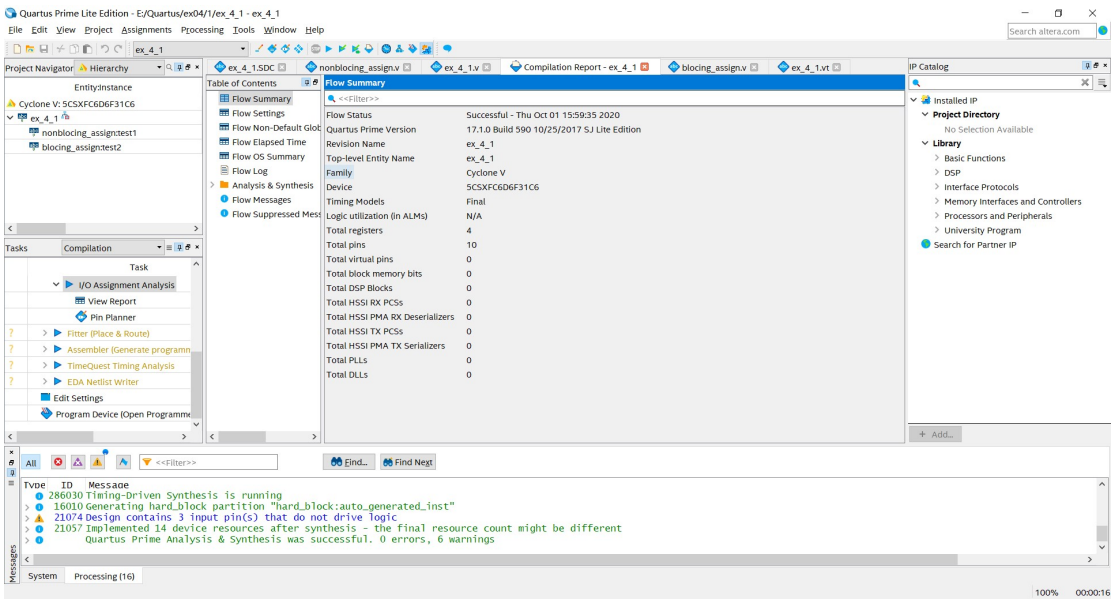


图 2-1-1：分析/综合成功

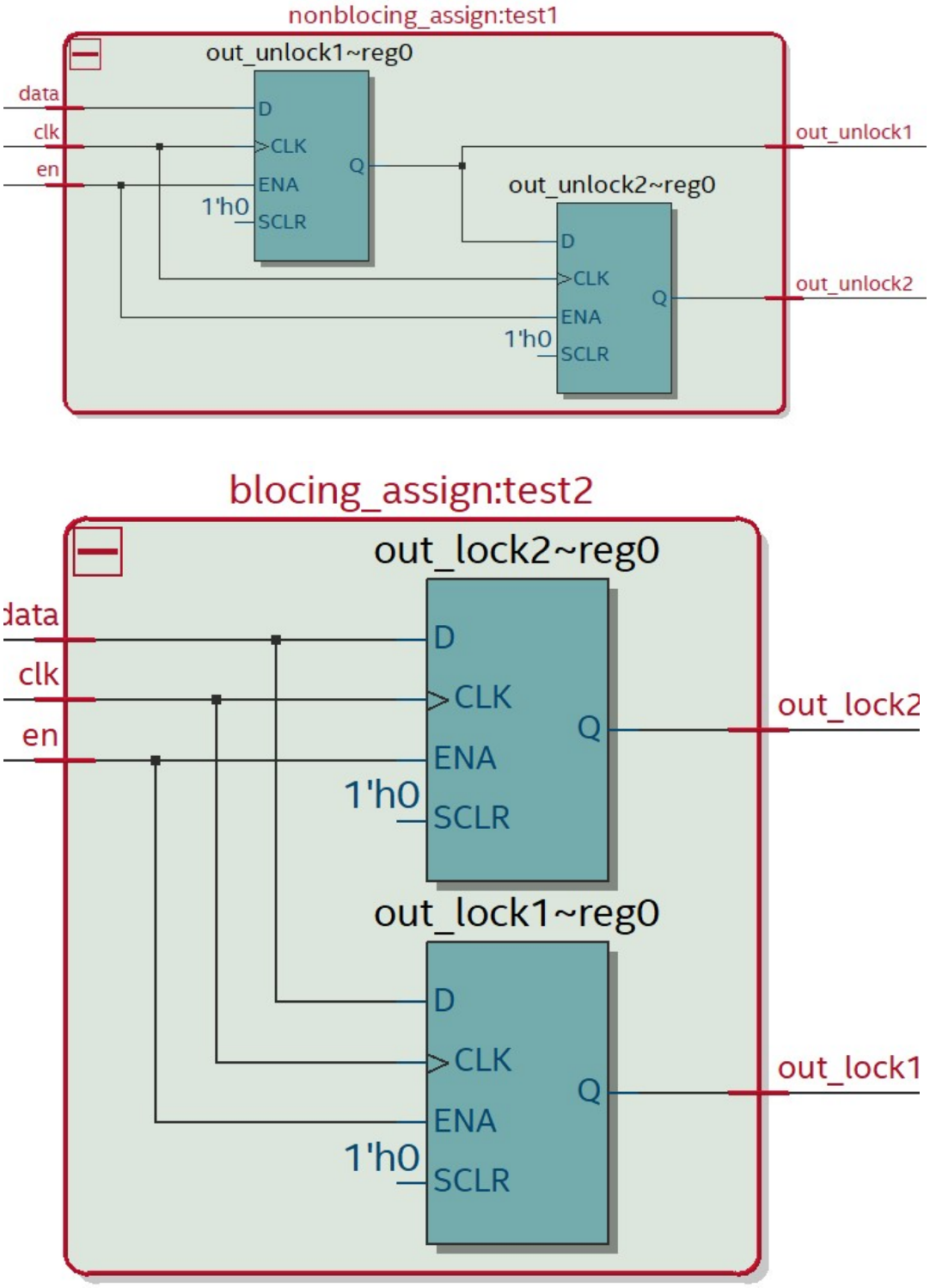


图 2-1-2:RTL 视图

2.1.5 仿真测试

测试样例/代码如下：

```
1    CLOCK_50 = 0; SW[0] = 1; SW[1] = 0; #7;
2        SW[1] = 0; #7;
```

```
3      SW[1] = 1; #7;
4             SW[1] = 0; #7;
5          SW[0] = 1; #7;
6             SW[1] = 0; #7;
7             SW[1] = 1; #7;
8             SW[1] = 0; #7;
9             SW[1] = 1; #7;
10         SW[0] = 0; #7;
11            SW[1] = 0; #7;
12            SW[1] = 1; #7;
13            SW[1] = 0; #7;
14            SW[1] = 1; #7;
15         $stop;
```

阻塞赋值和非阻塞赋值的触发器通过 ModelSim 执行上述测试代码得到的仿真结果如下：

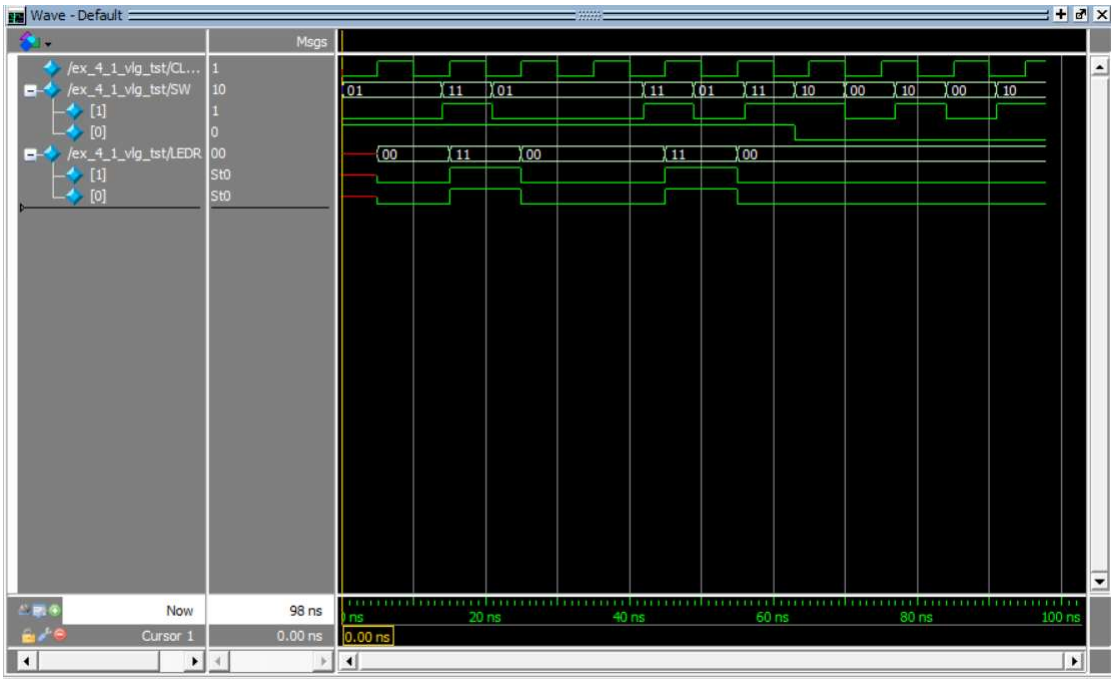


图 2-1-3 阻塞式赋值触发器仿真波形图

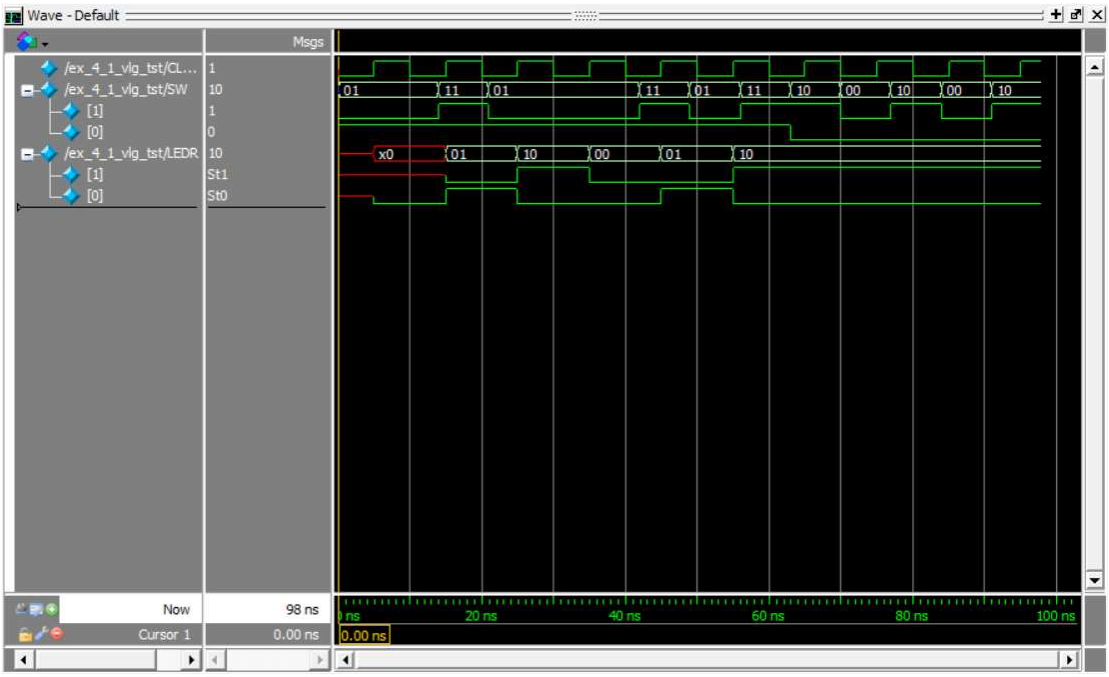


图 2-1-4：非阻塞式赋值触发器仿真波形图

观察上面的两个波形图发现非阻塞式赋值的触发器在时钟上升沿到来时进入 always 语句块已采样输入数据的值，但这个值将在 always 语句块结束后才会赋给 out_unlock1(LEDR[0])，故 out_unlock2(LEDR[1])在 always 语句块结束后将得到 out_unlock1 原来的值. 而阻塞赋值的触发器的输出端 out_unlock1 是在 always 语句块中立刻得到输入数据的值，out_unlock2 在 always 语句块中立即得到 out_unlock1 的值. 这也与 2.1.1 节中对阻塞和非阻塞式赋值特点的概述相符合.

下表给出了非阻塞式赋值和阻塞式赋值的触发器的行为表：

输入			输出	
数据输入端 input_data	使能信号 en	时钟信号 clk	数据输出 out_unlock1	数据输出 out_unlock2
x	0	x	lastout_unlock1	lastout_unlock2
x	1	0	lastout_unlock1	lastout_unlock2
x	1	1	lastout_unlock1	lastout_unlock2
0	1	↓	0	lastout_unlock1
1	1	↓	1	lastout_unlock1

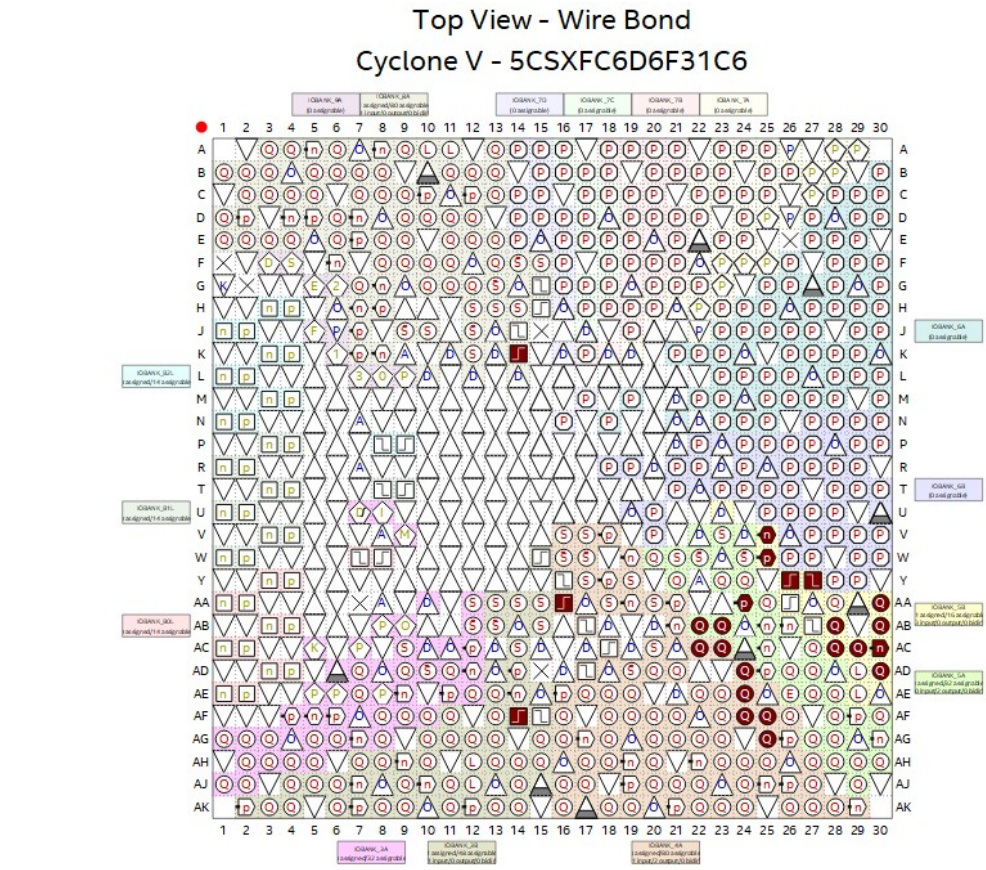
表 2-1-2:非阻塞赋值触发器行为表

输入			输出	
数据输入端 input_data	使能信号 en	时钟信号 clk	数据输出 out_unlock1	数据输出 out_unlock2
x	0	x	lastout_unlock1	lastout_unlock2
x	1	0	lastout_unlock1	lastout_unlock2
x	1	1	lastout_unlock1	lastout_unlock2
0	1	↓	0	0
1	1	↓	1	1

表 2-1-3:阻塞赋值触发器行为表

2.1.6 分配引脚

引脚分配使用 DE10_Standard_SystemBuilder 生成。



Node Name	Direction	Location	I/O Bank	VREF Group	itter Locatior	I/O Standard	Reserved	rrent Streng	Slew Rate	ifferential Pai	Analog Settir	(B/VCCT_GXE	er I/O Pin Ter	icated Refclk	mmon Mode	tter Slew Rate	)ifferential O
CLOCK2_50	Input	PIN_AA16	4A	B4A_N0	PIN_AA16	3.3-V LVTTL		16mA ...ault)									
CLOCK3_50	Input	PIN_Y26	5B	B5B_N0	PIN_Y26	2.5 V		12mA ...ault)									
CLOCK4_50	Input	PIN_K14	8A	B8A_N0	PIN_K14	3.3-V LVTTL		16mA ...ault)									
CLOCK_50	Input	PIN_AF14	3B	B3B_N0	PIN_AF14	3.3-V LVTTL		16mA ...ault)									
LEDR[3]	Output	PIN_AD24	4A	B4A_N0	PIN_AD24	3.3-V LVTTL		16mA ...ault)	1 (default)								
LEDR[2]	Output	PIN_AC23	4A	B4A_N0	PIN_AC23	3.3-V LVTTL		16mA ...ault)	1 (default)								
LEDR[1]	Output	PIN_AB23	5A	B5A_N0	PIN_AB23	3.3-V LVTTL		16mA ...ault)	1 (default)								
LEDR[0]	Output	PIN_AA24	5A	B5A_N0	PIN_AA24	3.3-V LVTTL		16mA ...ault)	1 (default)								
SW[1]	Input	PIN_Y27	5B	B5B_N0	PIN_Y27	2.5 V		12mA ...ault)									
SW[0]	Input	PIN_AB30	5B	B5B_N0	PIN_AB30	2.5 V		12mA ...ault)									
SW[2]	Unknown	PIN_AB28	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[3]	Unknown	PIN_AC30	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[4]	Unknown	PIN_W25	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[5]	Unknown	PIN_V25	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[6]	Unknown	PIN_AC28	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[7]	Unknown	PIN_AD30	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[8]	Unknown	PIN_AC29	5B	B5B_N0		2.5 V		12mA ...ault)									
SW[9]	Unknown	PIN_AA30	5B	B5B_N0		2.5 V		12mA ...ault)									
LEDR[4]	Unknown	PIN_AG25	4A	B4A_N0		3.3-V LVTTL		16mA ...ault)									
LEDR[5]	Unknown	PIN_AF25	4A	B4A_N0		3.3-V LVTTL		16mA ...ault)									
LEDR[6]	Unknown	PIN_AE24	4A	B4A_N0		3.3-V LVTTL		16mA ...ault)									
LEDR[7]	Unknown	PIN_AF24	4A	B4A_N0		3.3-V LVTTL		16mA ...ault)									
LEDR[8]	Unknown	PIN_AB22	5A	B5A_N0		3.3-V LVTTL		16mA ...ault)									
LEDR[9]	Unknown	PIN_AC22	4A	B4A_N0		3.3-V LVTTL		16mA ...ault)									

图 2-1-5 引脚分配图

2.1.7 全编译

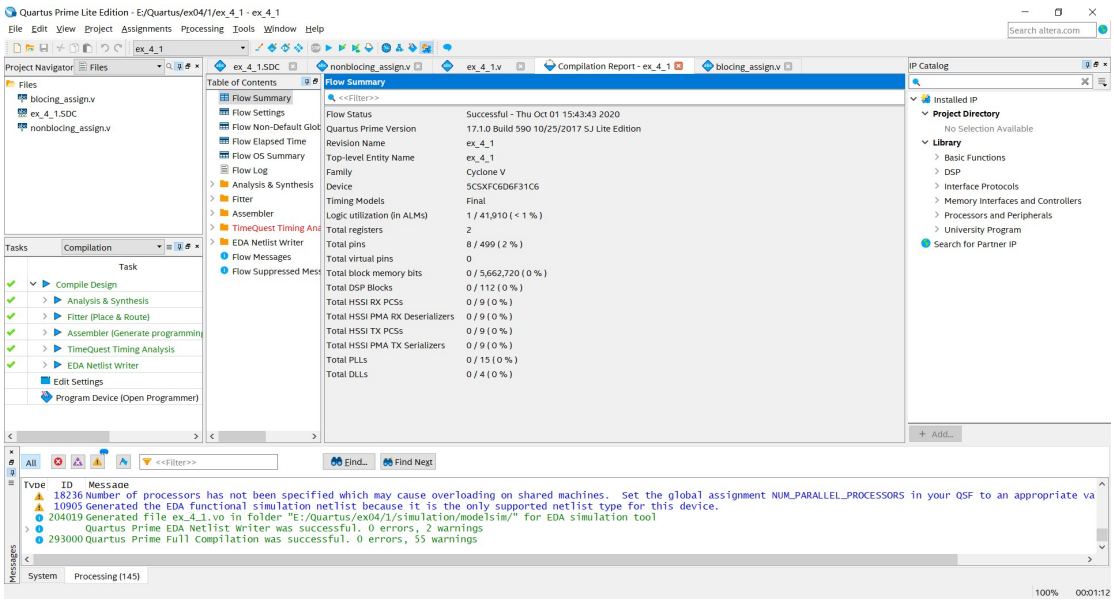


图 2-1-6 全编译成功

2.2 设计一个同步清零和一个异步清零的 D 触发器

2.2.1 模型概述

使用 Verilog HDL 分别实现同步清零和异步清零的 D 触发器. 同步清零 D 触发器的清零操作在时钟信号有效时才会执行，而异步清零 D 触发器的清零操作与时钟信号无关.

2.2.2 数字抽象

A)同步清零 D 触发器

I) 输入:

数据输入端 D： 当时钟信号有效时 D 的值将会输入到触发器中被存储起来.

清零信号 clr_n：当清零信号和时钟信号同时有效时，D 触发器会将内部存储置 0.

时钟信号 clk： 与系统时钟连接，控制触发器的行为.

II) 输出:

数据输出端 Q：以 4 位二进制补码的形式输出运算结果.

下表给出了以上输入输出信号在 DE10 平台对应的信号：

	信号名称	DE-10平台信号
输入	D	SW[0]
	clr_n	SW[1]
	clk	SW[4]
输出	Q	LEDR[0]

表 2-2-1:同步清零 D 触发器输入输出信号与 DE10 平台信号对应关系

B)异步清零 D 触发器

I) 输入:

数据输入端 D： 当时钟信号有效时 D 的值将会输入到触发器中被存储起来.

清零信号 clr_n：当清零信号有效时，D 触发器会将内部存储置 0.

时钟信号 clk： 与系统时钟连接，控制触发器的行为.

III) 输出:

数据输出端 Q：以 4 位二进制补码的形式输出运算结果.

下表给出了以上输入输出信号在 DE10 平台对应的信号：

	信号名称	DE-10 平台信号
输入	D	SW[0]
	clr_n	SW[1]
	clk	SW[4]
输出	Q	LEDR[1]

表 2-2-2:异步清零 D 触发器输入输出信号与 DE10 平台信号对应关系

2.2.3 建立模型

A) 同步清零 D 触发器

下表给出了同步清零 D 触发器输出与输入的关系

输入			输出
数据输入端 D	清零信号 clr_n	时钟信号 clk	数据输出端 Q
x	x	0	lastQ
x	x	1	lastQ
0	1	↓	0
1	1	↓	1
x	0	↓	0
x	0	↓	0

表 2-2-3:同步清零 D 触发器行为表

实现思路：同步清零 D 触发器对内部存储的修改仅由时钟信号的上升沿触发，故使用 always 语句监测 clk 信号的上升沿，若 clk 上升沿到来再根据数据输入和清零信号执行相应的操作.

同步清零 D 触发器的 Verilog HDL 实现如下：

```
1   module synDff(D, clr_n, clk, Q);
2       input D;
3       input clr_n;
4       input clk;
5       output reg Q;
6   always @ (posedge clk) begin
```

```
            if(!clr_n) begin
                Q <= 0;
        end
        else Q <= D;
    end
endmodule
```

B) 异步清零 D 触发器

下表给出了异步清零 D 触发器输出与输入的关系

输入			输出
数据输入端 D	清零信号 clr_n	时钟信号 clk	数据输出端 Q
x	x	0	lastQ
x	x	1	lastQ
0	1	↓	0
1	1	↓	1
x	0	x	0
x	0	x	0

表 2-2-4:异步清零 D 触发器行为表

实现思路：异步清零 D 触发器对内部存储的修改和时钟信号的上升沿、清零信号触发，又由于清零信号低有效，故使用 always 语句监测 clk 信号的上升沿和清零信号的下降沿.若清零信号的下降沿到来则将触发器内部存储置零，否则若 clk 上升沿到来则将数据输入到触发器内部存储.

异步清零 D 触发器的 Verilog HDL 实现如下：

```
module aynDff(D, clr_n, clk, Q);
    input D;
    input clr_n;
    input clk;
    output reg Q;

    always @ (negedge clr_n or posedge clk) begin
        if(!clr_n) begin
                Q <= 0;
            end
        else Q <= D;
    end
endmodule
```

2.2.4 分析/综合

分析/综合实验成功，如下图所示：

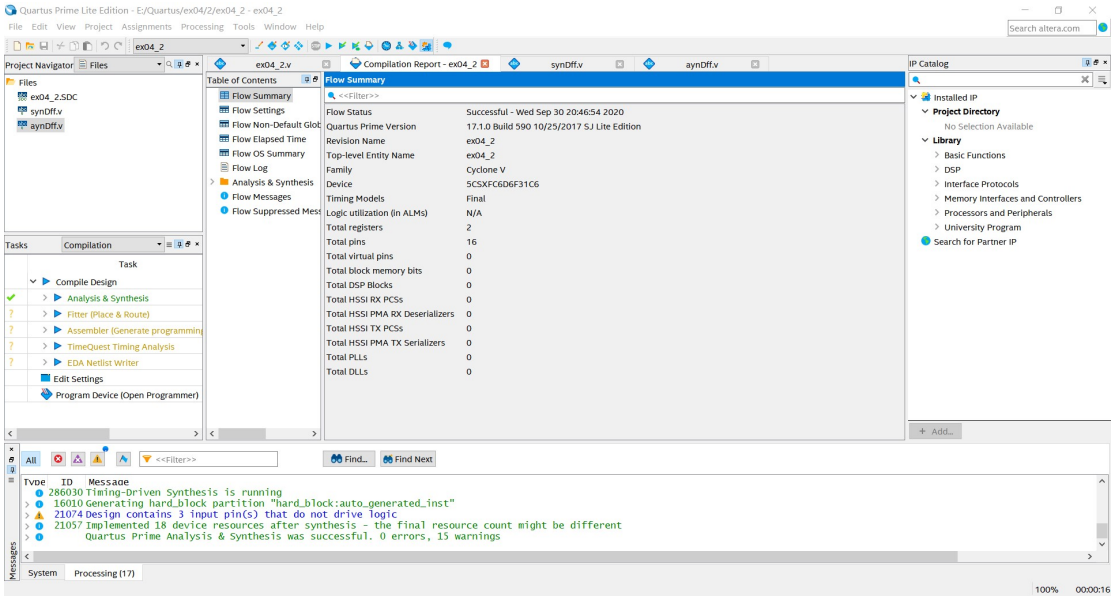


图 2-2-1：分析/综合成功

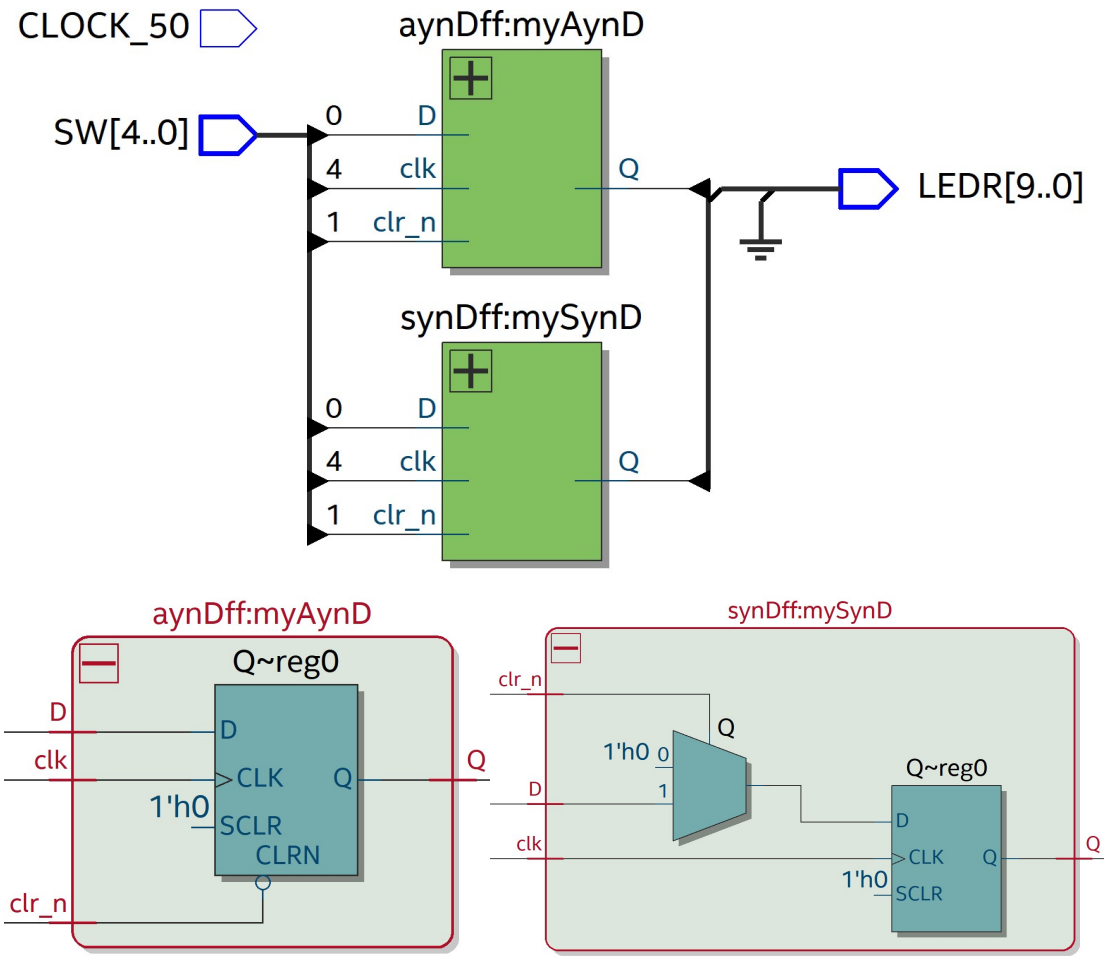


图 2-2-2:RTL 视图

2.2.5 仿真测试

根据同步/异步清零 D 触发器的逻辑功能，给出如下测试代码，从赋值、清零考察模型设计的正确性.

```
1    initial
2    begin          // code that executes only once
3      CLOCK_50 = 0;
4           SW[0] = 1; SW[1] = 1; #12;
5           SW[1] = 0; #7;
6           SW[1] = 1; #7;
7        $stop;
8    // --> end
9    end
10   always
11   begin   // code executes for every event on sensitivity list
12     #5 CLOCK_50 = ~CLOCK_50;
13   // --> end
14   end
```

运行仿真模拟后得到的波形图如下：

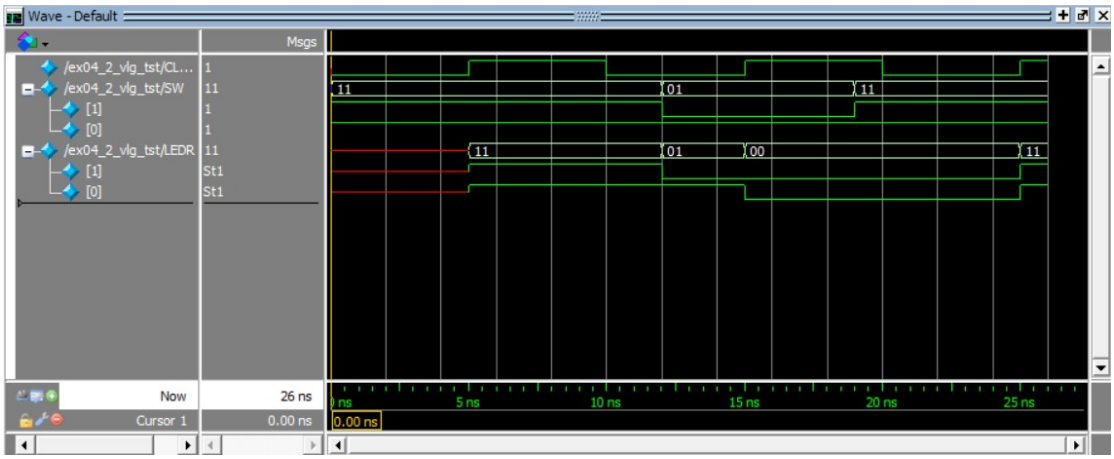
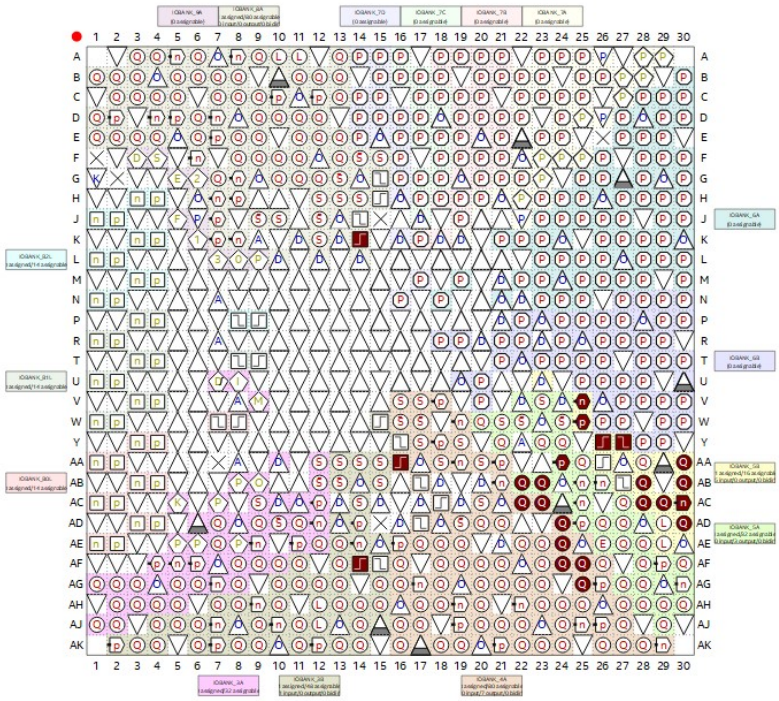


图 2-2-3:仿真波形图

发现在清零信号（SW[1]）为 0 时（上图中的 12ns 时刻），异步清零 D 触发器的数据输出（LEDR[1]）立即置 0，而同步清零 D 触发器的数据输出（LEDR[0]）在时钟上升沿到来时（上图中的 15ns 时刻）才被置 0，符合设计需求.

2.2.6 分配引脚

引脚分配使用 DE10_Standard_SystemBuilder 生成。



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	urrent Streng	Slew Rate	ifferential Pai	Analog Settir	(B/VCCT_GXE	er I/O Pin Ter	icated Refclk	mmon Mode	tter Slew Rate	)ifferential Ou	er Common I
CLOCK_50	Input	PIN_AF14	3B	B3B_N0	3.3-V LVTTL		16mA ...ault)										
LEDR[9]	Output	PIN_AC22	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[8]	Output	PIN_AB22	5A	B5A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[7]	Output	PIN_AF24	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[6]	Output	PIN_AE24	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[5]	Output	PIN_AF25	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[4]	Output	PIN_AG25	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[3]	Output	PIN_AD24	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[2]	Output	PIN_AC23	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[1]	Output	PIN_AB23	5A	B5A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
LEDR[0]	Output	PIN_AA24	5A	B5A_N0	3.3-V LVTTL		16mA ...ault)	1 (default)									
SW[4]	Input	PIN_W25	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[3]	Input	PIN_AC30	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[2]	Input	PIN_AB28	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[1]	Input	PIN_Y27	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[0]	Input	PIN_AB30	5B	B5B_N0	2.5 V		12mA ...ault)										
CLOCK2_50	Unknown	PIN_AA16	4A	B4A_N0	3.3-V LVTTL		16mA ...ault)										
CLOCK3_50	Unknown	PIN_Y26	5B	B5B_N0	2.5 V		12mA ...ault)										
CLOCK4_50	Unknown	PIN_K14	8A	B8A_N0	3.3-V LVTTL		16mA ...ault)										
SW[5]	Unknown	PIN_V25	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[6]	Unknown	PIN_AC28	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[7]	Unknown	PIN_AD30	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[8]	Unknown	PIN_AC29	5B	B5B_N0	2.5 V		12mA ...ault)										
SW[9]	Unknown	PIN_AA30	5B	B5B_N0	2.5 V		12mA ...ault)										

图 2-2-3 引脚分配图

2.2.7 全编译

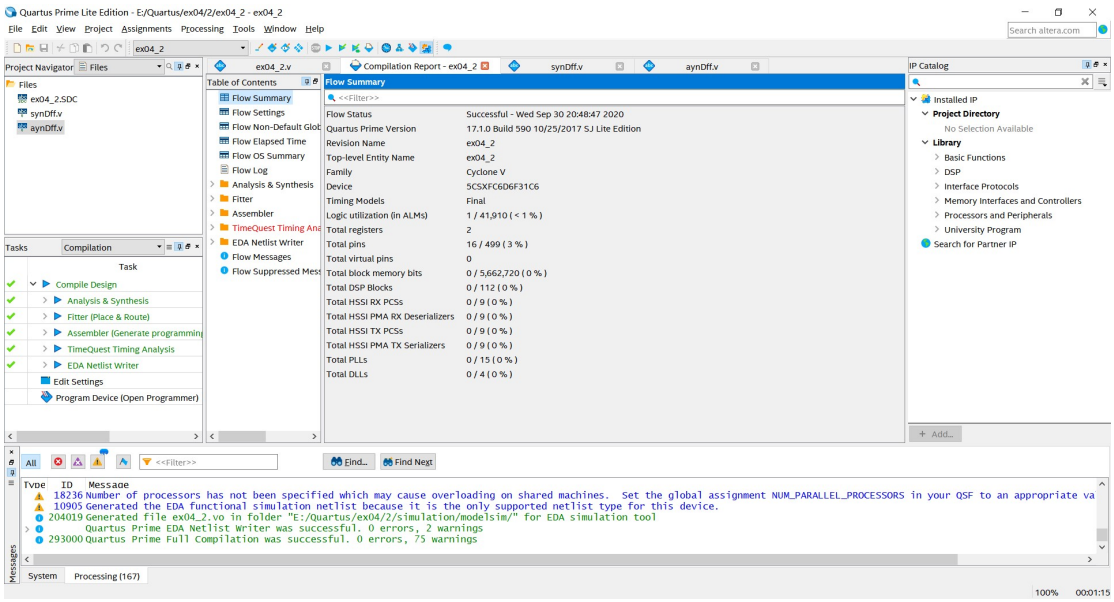


图 2-2-4 全编译成功

三、实验总结

本次实验主要学习实践了阻塞和非阻塞赋值的区别和同步、异步清零触发器的区别和实现.

非阻塞赋值的方式主要用于时序逻辑电路的建模，其作用在于在 always 语句块之后将结果给表达式右侧的值赋给左侧的值. 同步/异步清零的区别在于清零操作是否受到时钟信号的影响. 异步清零由于不受时钟信号影响故 always 语句也需要监视清零信号的对应变化（具体是上升还是下降沿由清零信号的有效值决定）