# 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](mailto:1830970417@qq.com)

2020年9月14日

# **目录**

1. [实验内容](#_实验内容)……………………………………………………………………………………3
   1. [实验要求](#_1.1实验要求)………………………………………………………………………………3
   2. [实验工具](#_1.2实验工具)………………………………………………………………………………3
2. [实验过程](#_实验过程)……………………………………………………………………………………3

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](#_2.1模型概述)………………………………………………………………………3

2.1.2 [数字抽象](#_2.2数字抽象)………………………………………………………………………4

2.1.3 [建立模型](#_2.3建立模型)………………………………………………………………………4

2.1.4 [分析/综合](#_2.4分析/综合)…………………………………………………………………… 5

2.1.5 [仿真测试](#_2.5仿真测试)………………………………………………………………………6

2.1.6 [分配引脚](#_2.6分配引脚)………………………………………………………………………9

2.1.7 [全编译](#_2.7全编译)…………………………………………………………………………10

2.2 设计一个同步清零和异步清零的D触发器

2.2.1 [模型概述](#_2.2.1模型概述)………………………………………………………………………10

2.2.2 [数字抽象](#_2.2.2数字抽象)………………………………………………………………………10

2.2.3 [建立模型](#_2.2.3建立模型)………………………………………………………………………11

2.2.4 [分析/综合](#_2.2.4分析/综合)…………………………………………………………………… 12

2.2.5 [仿真测试](#_2.2.5仿真测试)………………………………………………………………………14

2.2.6 [分配引脚](#_2.2.6分配引脚)………………………………………………………………………15

2.2.7 [全编译](#_2.2.7全编译)…………………………………………………………………………16

三、[实验总结](#_实验总结)……………………………………………………………………………………16

#### 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

#### 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：

1. 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.

1. 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：

module blocing\_assign(data, clk, en, out\_lock1, out\_lock2);

input data;

input clk;

input en;

output reg out\_lock1;

output reg out\_lock2;

always @(posedge clk)

if(en)

begin

out\_lock1 = data;

out\_lock2 = out\_lock1;

end

else

begin

out\_lock1 = out\_lock1;

out\_lock2 = out\_lock2;

end

endmodule

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |

阻塞赋值触发器的Verilog HDL实现如下：

|  |  |
| --- | --- |
| 1  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 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |

###### 2.1.4分析/综合

分析/综合实验成功，如下图所示：

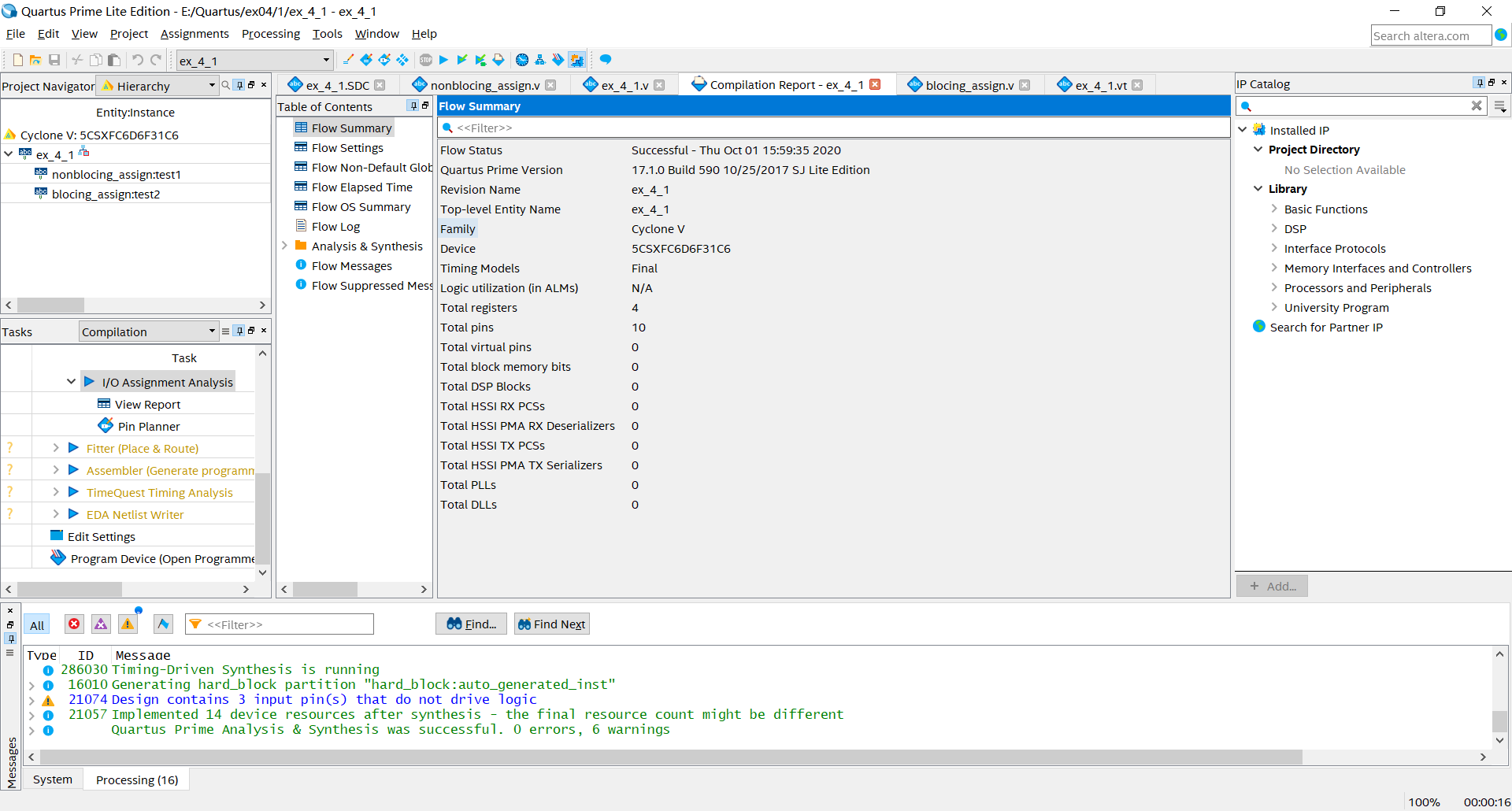
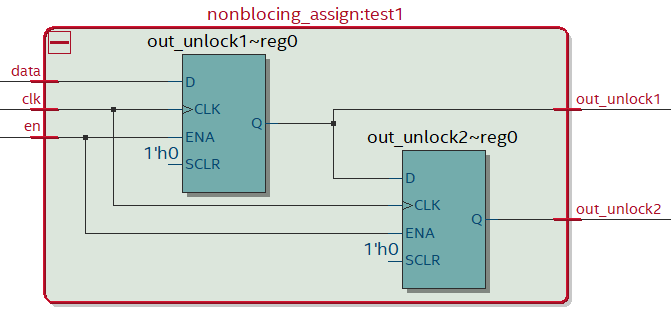


图2-1-1：分析/综合成功



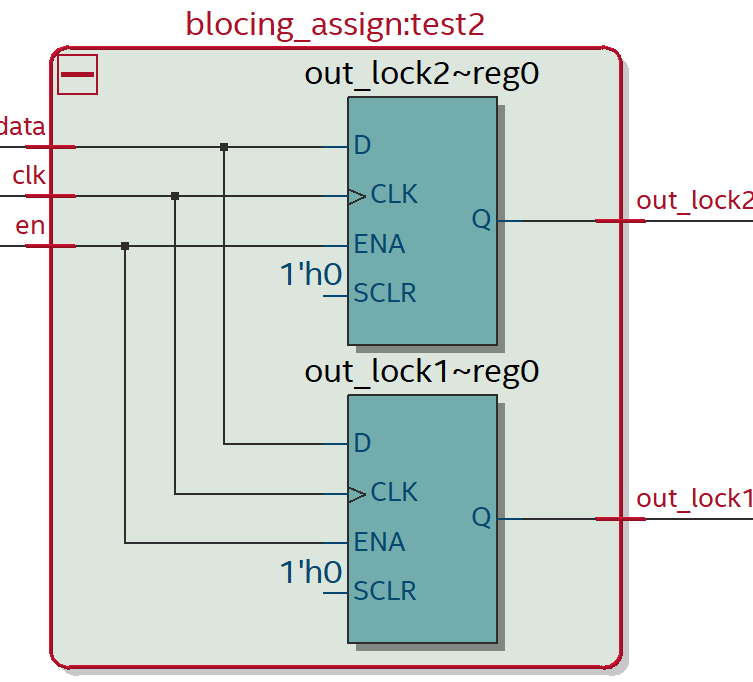


图 2-1-2:RTL视图

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

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：

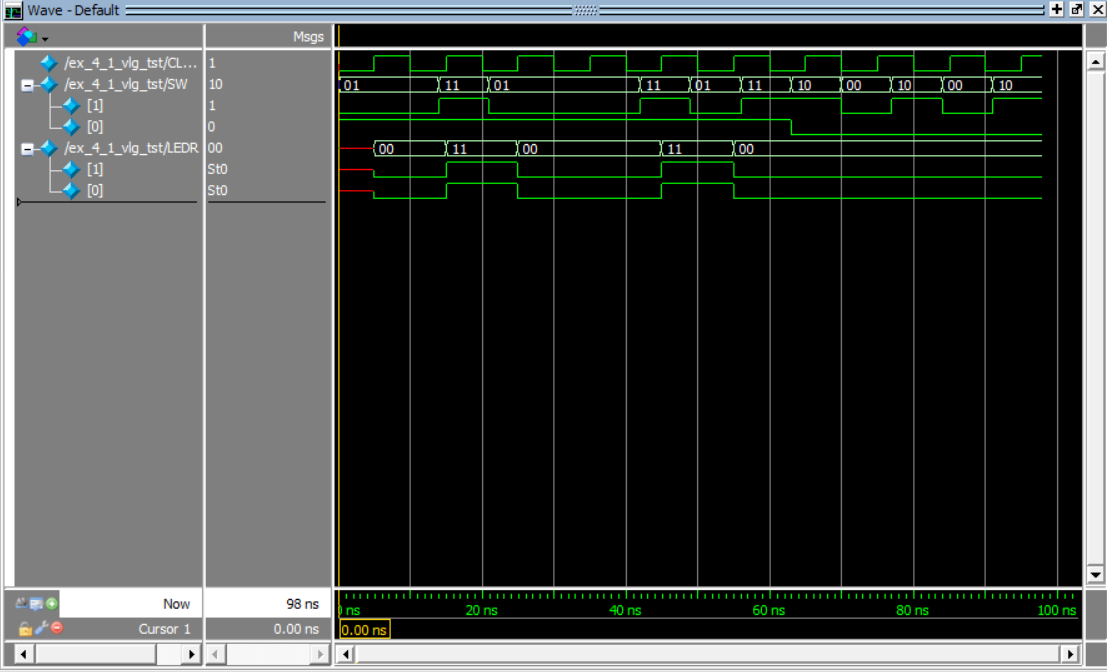


图 2-1-3阻塞式赋值触发器仿真波形图

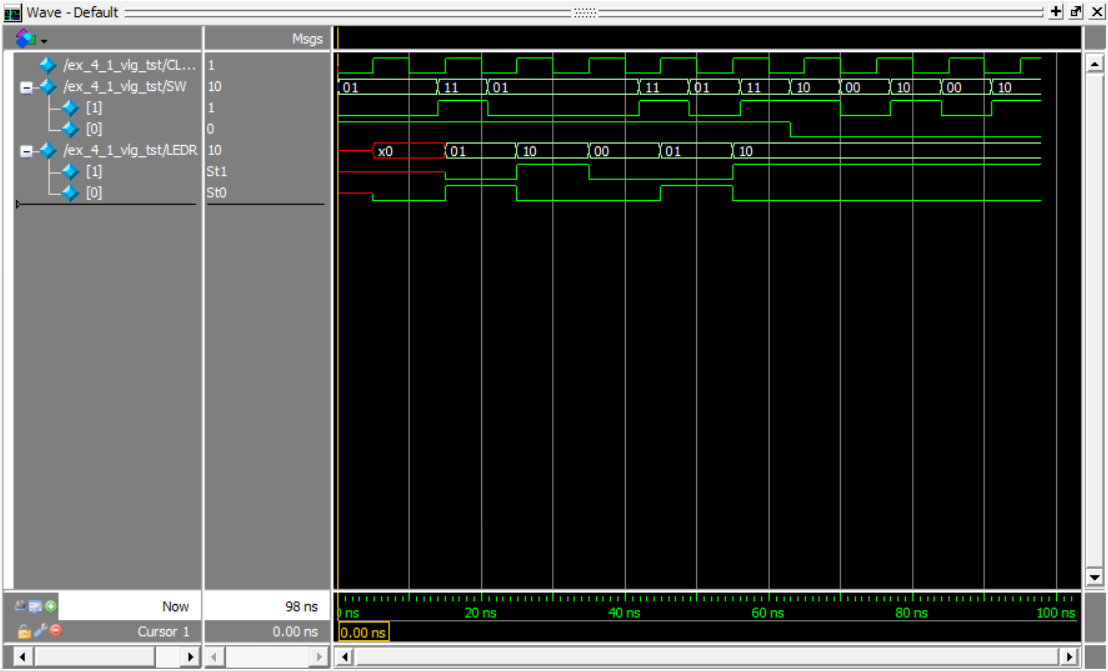


图 2-1-4：非阻塞式赋值触发器仿真波形图

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 |

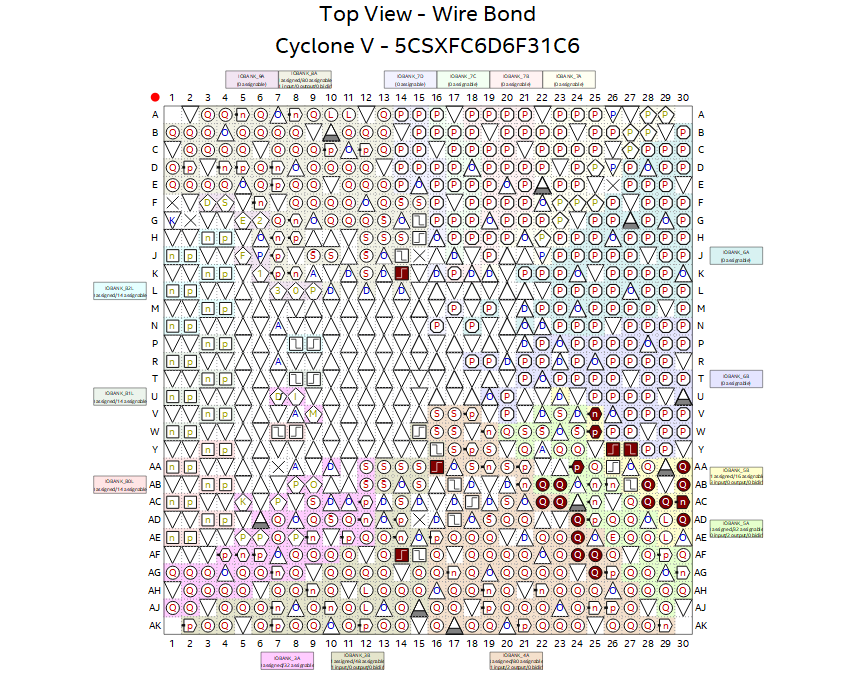
表 2-1-2:非阻塞赋值触发器行为表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。



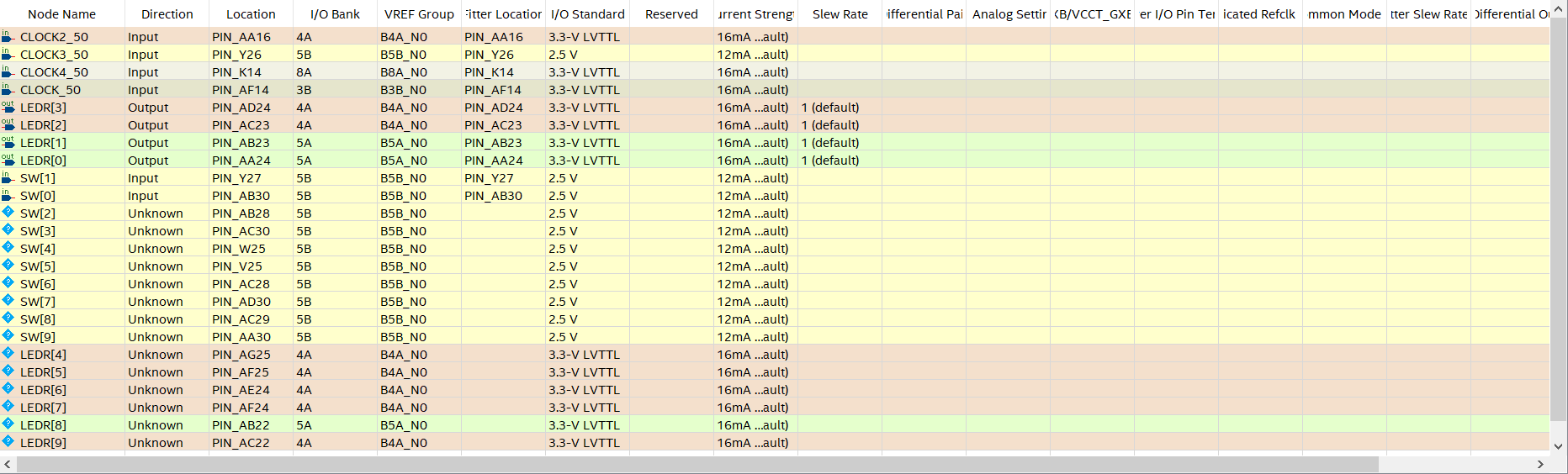


图2-1-5引脚分配图

###### 2.1.7全编译

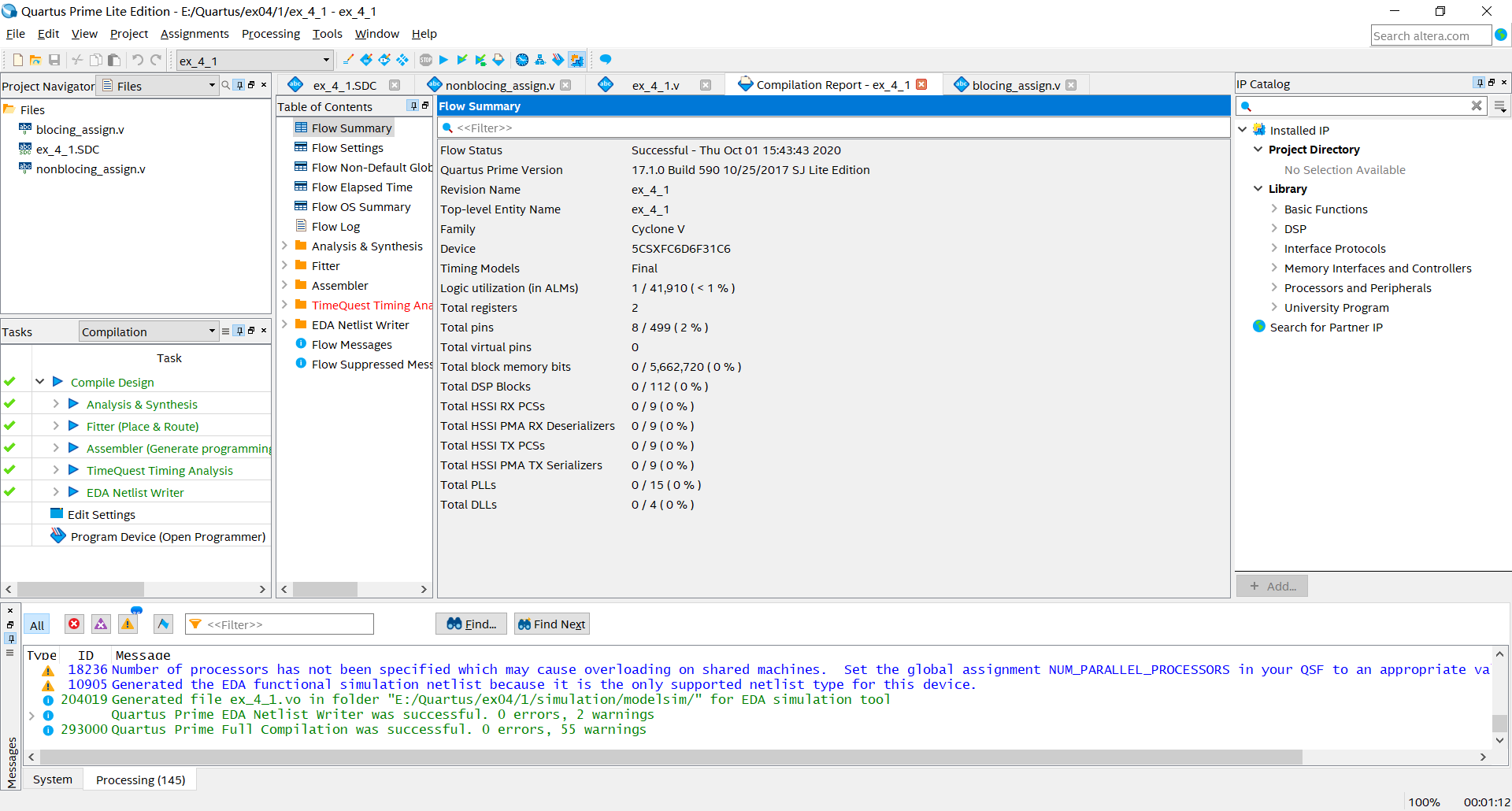


图2-1-6全编译成功

##### 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**

1. 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.

1. 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.

1. 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：

module synDff(D, clr\_n, clk, Q);

input D;

input clr\_n;

input clk;

output reg Q;

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 | always @ (posedge clk) begin |
| 7 | if(!clr\_n) begin  Q <= 0;  end  else Q <= D;  end  endmodule |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |

1. **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

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |

###### 2.2.4分析/综合

分析/综合实验成功，如下图所示：

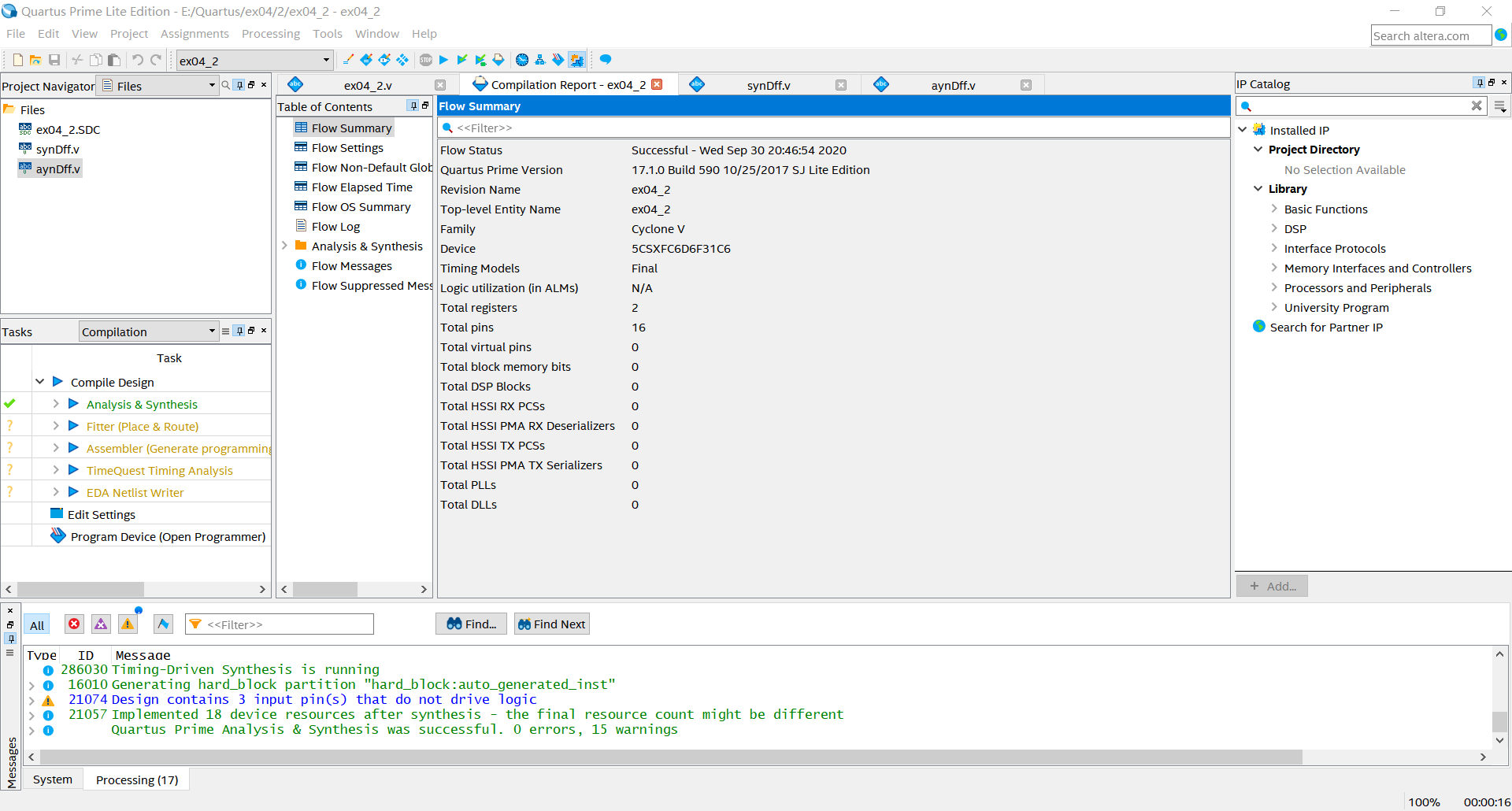
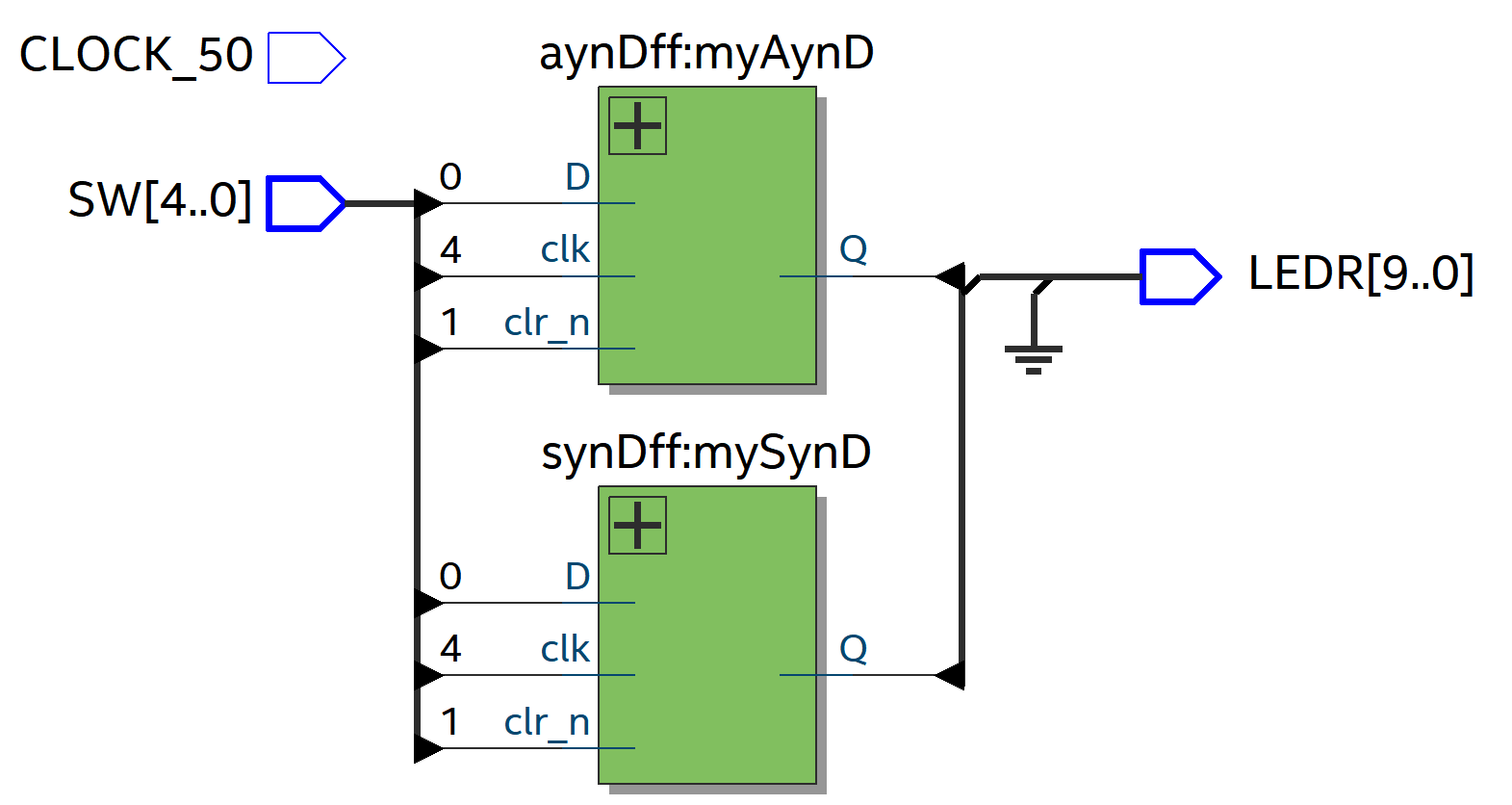


图2-2-1：分析/综合成功



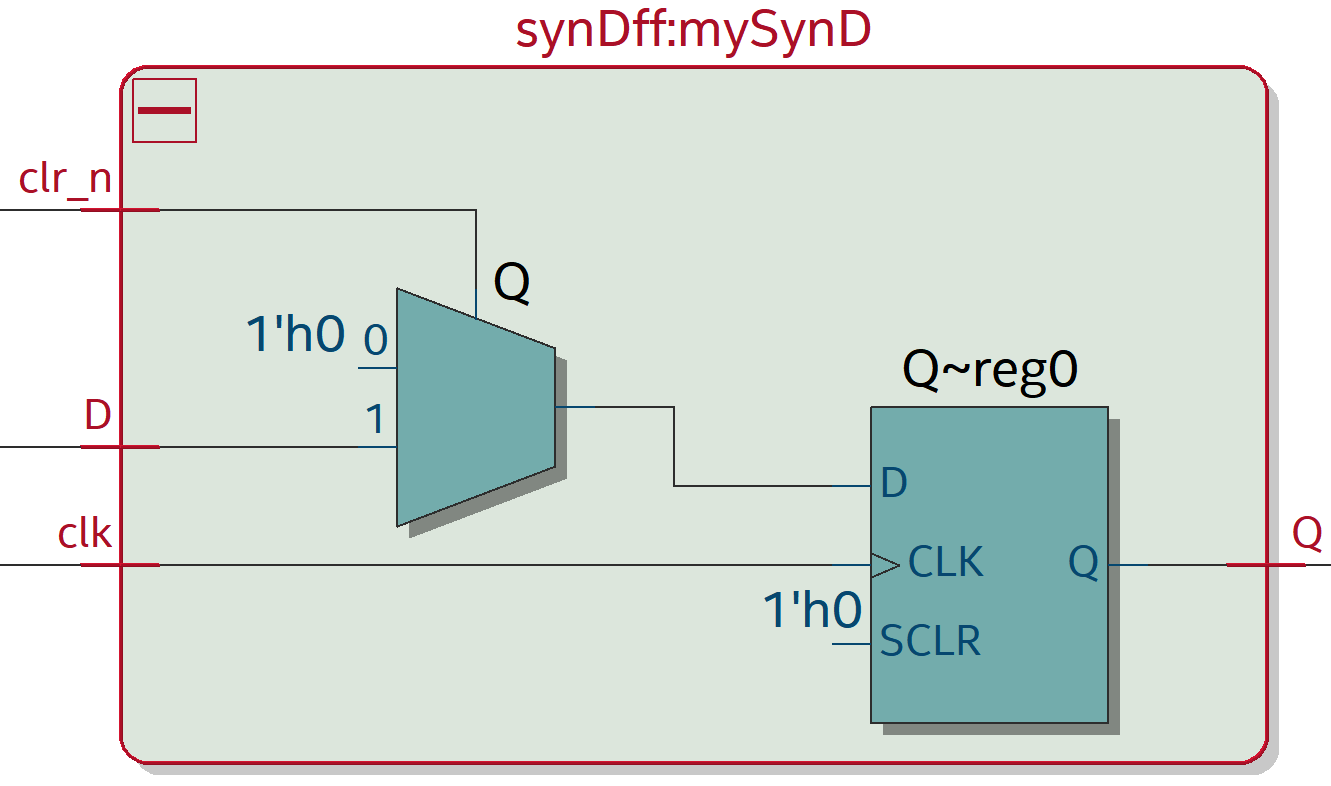
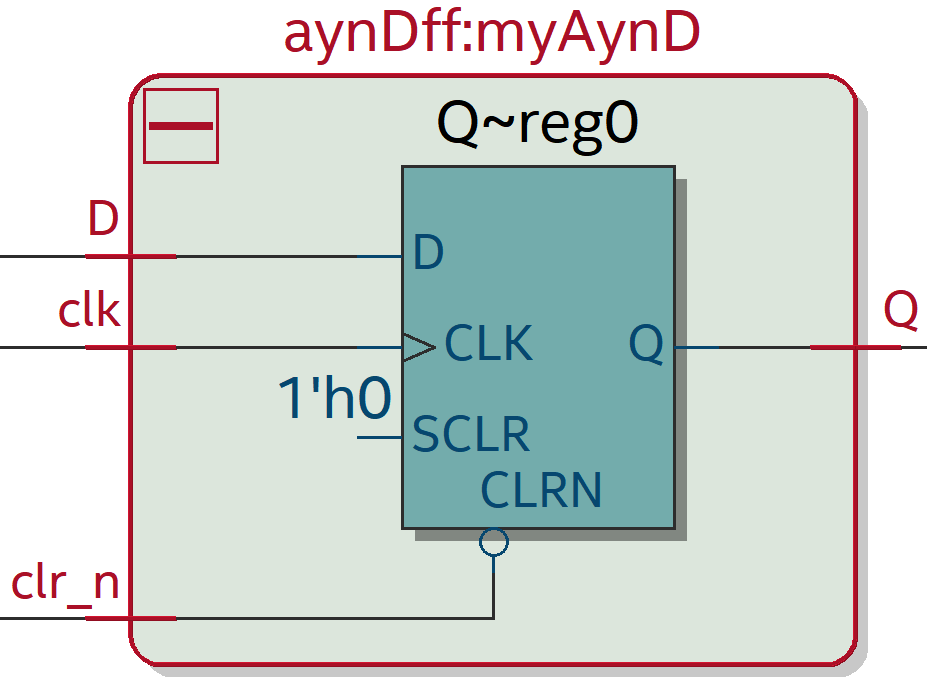


图 2-2-2:RTL视图

###### 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  begin // code that executes only once  CLOCK\_50 = 0;  SW[0] = 1; SW[1] = 1; #12;  SW[1] = 0; #7;  SW[1] = 1; #7;  $stop;  // --> end  end  always  begin // code executes for every event on sensitivity list  #5 CLOCK\_50 = ~CLOCK\_50;  // --> end  end |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |

运行仿真模拟后得到的波形图如下：

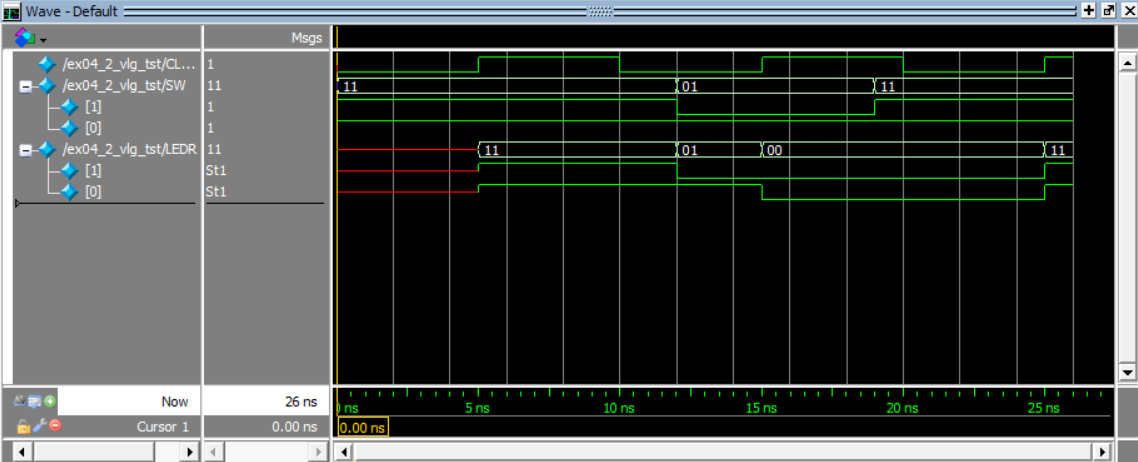
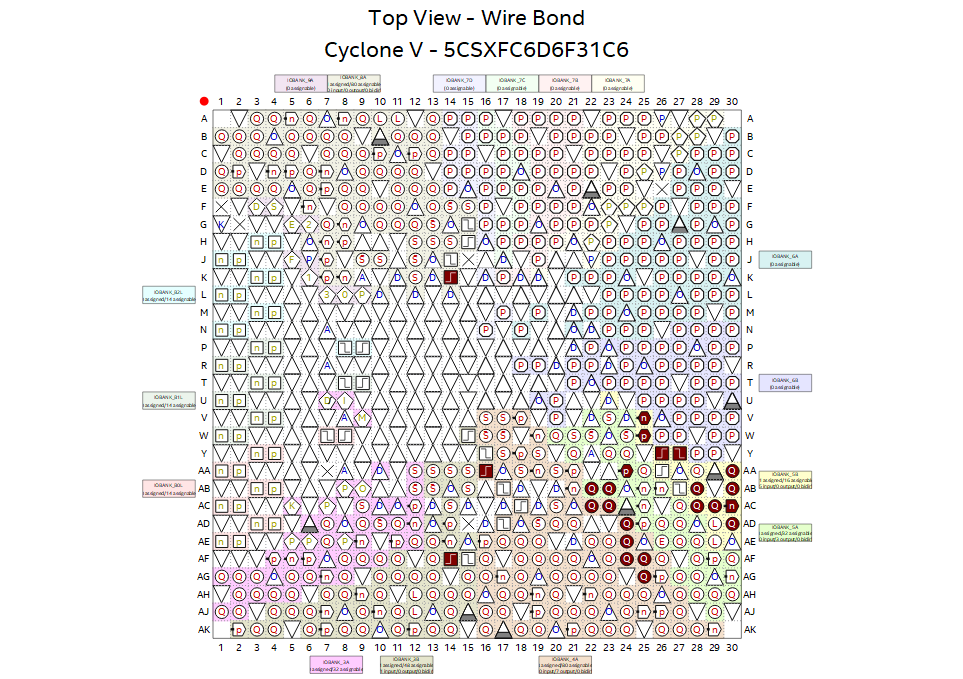


图 2-2-3:仿真波形图

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。



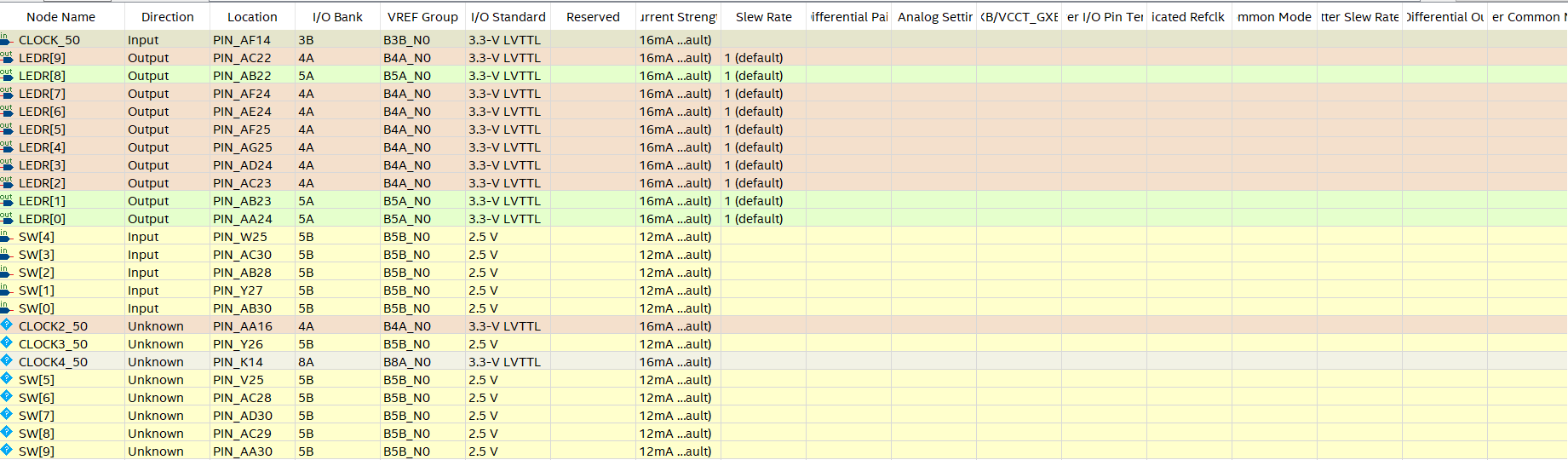


图2-2-3引脚分配图

###### 2.2.7全编译

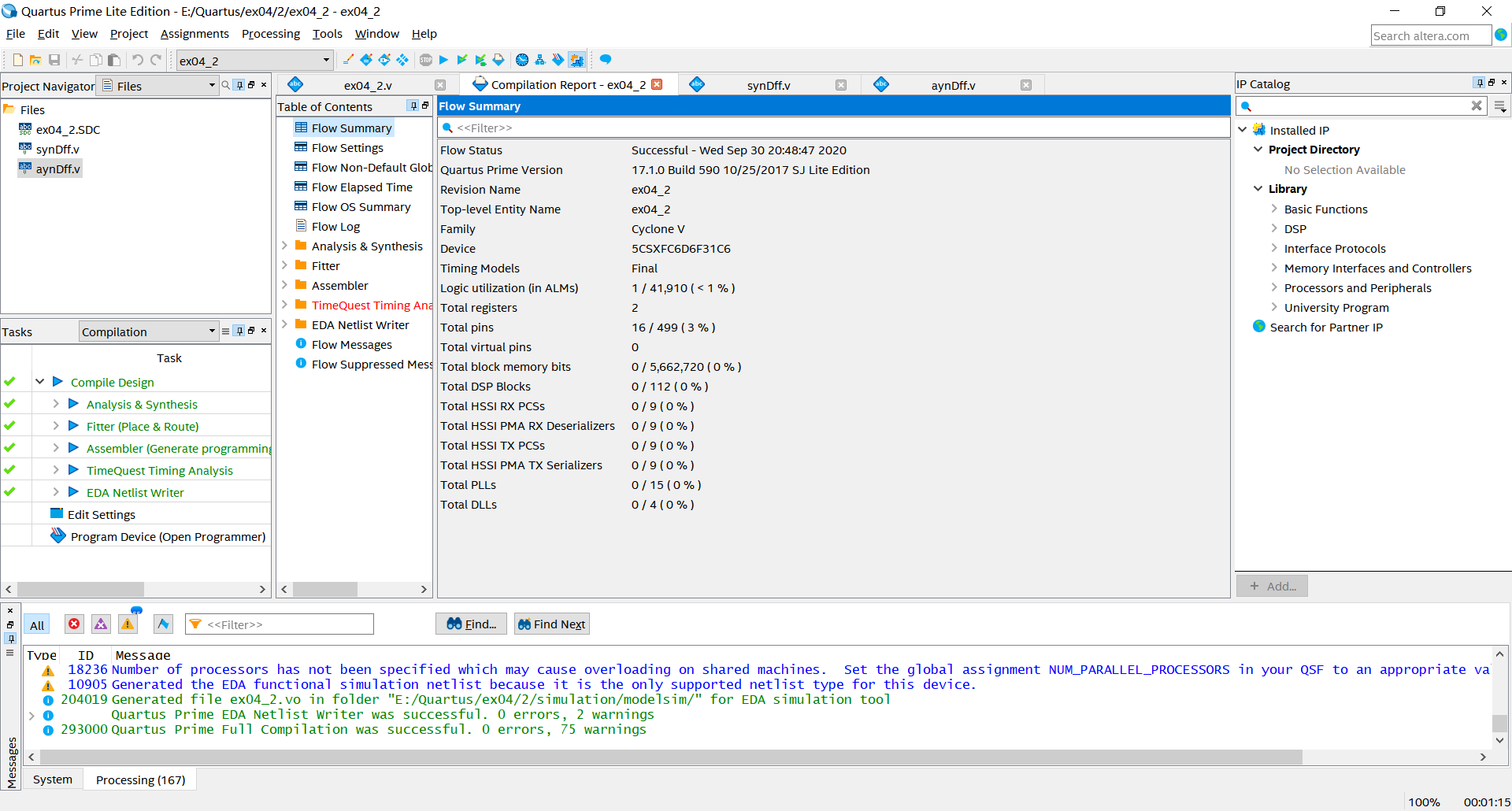


图2-2-4全编译成功

#### 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）