# 数字电路与数字系统实验

## EX06:计数器和时钟

191220029 傅小龙

周一5-6节班

[1830970417@qq.com](mailto:1830970417@qq.com)

2020年10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)……………………………………………………………………………………4

2.1 算术移位和逻辑移位寄存器

2.1.1 [模型概述](#_2.1模型概述)………………………………………………………………………4

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)………………………………………………………………………7

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 利用移位寄存器实现随机数发生器

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)…………………………………………………………………… 11

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)………………………………………………………………………14

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)算术移位和逻辑移位寄存器***

请根据表1，用 Verilog HDL 语言设计一个移位寄存器，并进行仿真查看移位寄存器的功能。

|  |  |
| --- | --- |
| 控制位 | 工作方式 |
| 000  001  010  011  100  101  110  111 | 清0  置数  逻辑右移  逻辑左移  算术右移  左端串行输入1位值，并行输出8位值  循环右移  循环左移 |

表1:移位寄存器的工作方式

***II)利用移位寄存器实现随机数发生器***

利用 8 位移位寄存器来实现一个简单的随机数发生器。参考教科书第 534 页 LFSR 反馈方程设计一个 n=8，共有 255 种状态的随机数发生器。请将 8 位二进制数以十六进制显示在数码管上，在 DE10-Standard 开发板上观察生成的随机数序列。系统需要能够自启动。

##### 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 算术移位和逻辑移位寄存器

###### 2.1.1模型概述

使用 Verilog HDL 实现一个8位移位寄存器. 通过LED灯显示寄存器内的8位二进制数. 该寄存器使用1Hz时钟信号来进行实际测试故需要一个时钟分频器. 对该寄存器的所有状态改变操作是同步的.

###### 2.1.2数字抽象

1. 输入:

控制信号[2:0]ctrl: 确定寄存器执行的功能.

数据输入[7:0]data\_in: 8位二进制码的输入.

时钟信号clk： 与时钟信号连接.

1. 输出:

数据输出[7:0]Q: 8位二进制码的输出

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 | [2:0]ctrl | KEY[2:0] |
| [7:0]data\_in | SW[7:0] |
| clk | LEDR[0]① |
| 输出 | [7:0]Q | LEDR[7:0] |

①:LEDR[0]为分频器产生的1Hz时钟信号输出

表 2-1-1:算术移位和逻辑移位寄存器输入输出信号与DE10平台信号对应关系

###### 2.1.3建立模型

**实现思路：**根据ex06.pdf 6.3.1节中给出的循环右移和算数右移的Verilog HDL语言实现样例，模100计数器的Verilog HDL实现如下：

module ShiftReg(ctrl, data\_in, clk, Q);

input [2:0]ctrl;

input [7:0]data\_in;

input clk;

output reg [7:0]Q;

always @(posedge clk) begin

case(ctrl)

0: begin//清零

Q <= 0;

end

1: begin //置数

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13  Q <= data\_in;  end  2: begin//逻辑右移  Q <= {1'b0, Q[7:1]};  end  3: begin//逻辑左移  Q <= {Q[6:0], 1'b0};  end  4: begin//算数右移  Q <= {Q[7], Q[7:1]};  end  5: begin //左端串行输入1位值，并行输出8位值  Q <= {data\_in[0], Q[7:1]};  end  6: begin //循环右移  Q <= {Q[0], Q[7:1]};  end  7: begin //循环左移  Q <= {Q[6:0], Q[7]};  end  default:Q <= Q;  endcase  end  endmodule |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |

参照exp05.pdf表5-2秒时钟生成代码，本实验所用到的分频器的实现如下：

|  |  |
| --- | --- |
| 1  module clock\_1(clk\_50M, clk\_o);  input clk\_50M;  output reg clk\_o;  integer cnt;    always @(posedge clk\_50M) begin  if(cnt == 25000000) begin  cnt <= 0;  clk\_o <= ~clk\_o;  end  else cnt <= cnt + 1;  end  endmodule |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |

###### 2.1.4分析/综合

分析/综合实验成功，如下图所示：

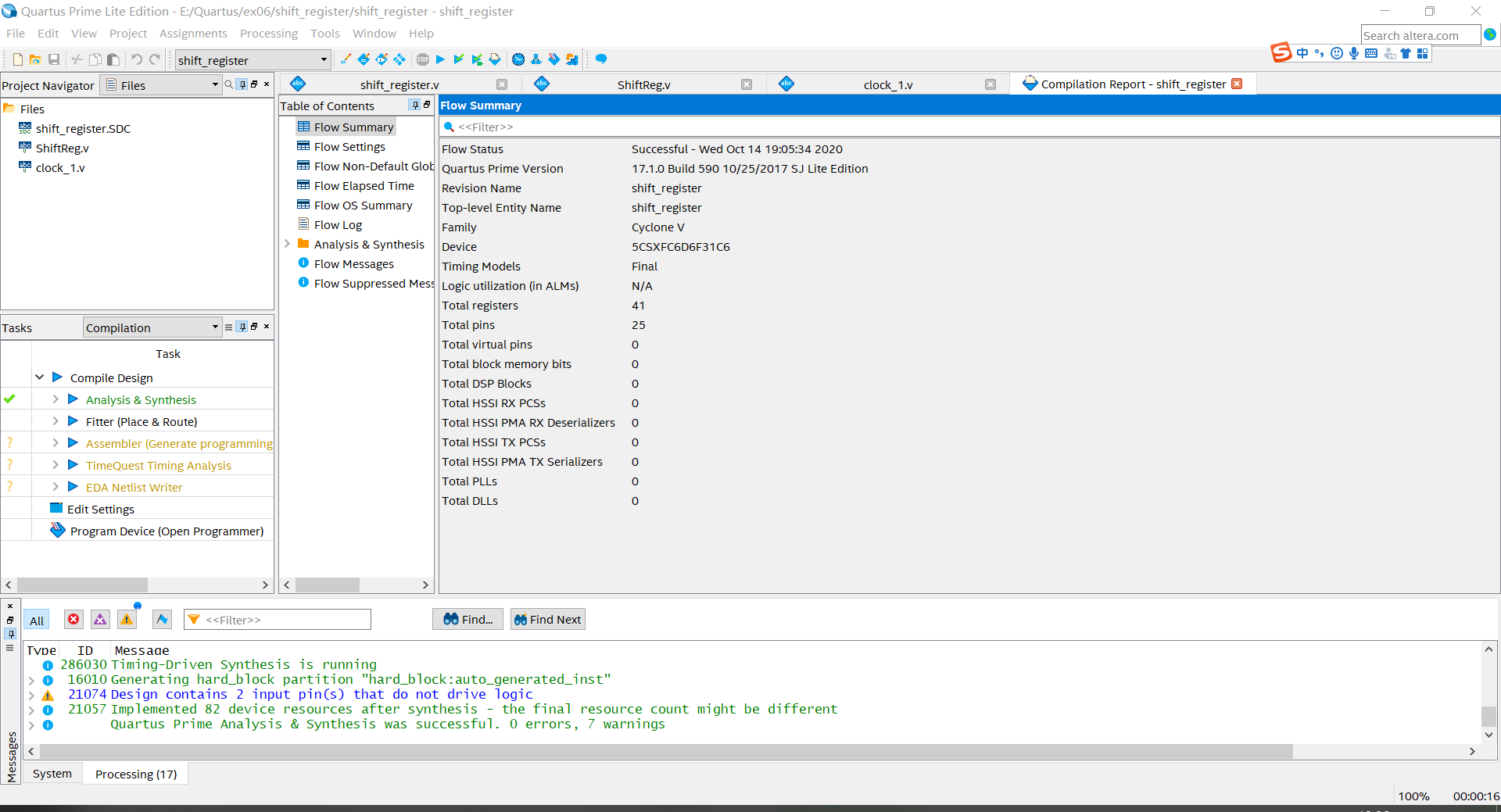
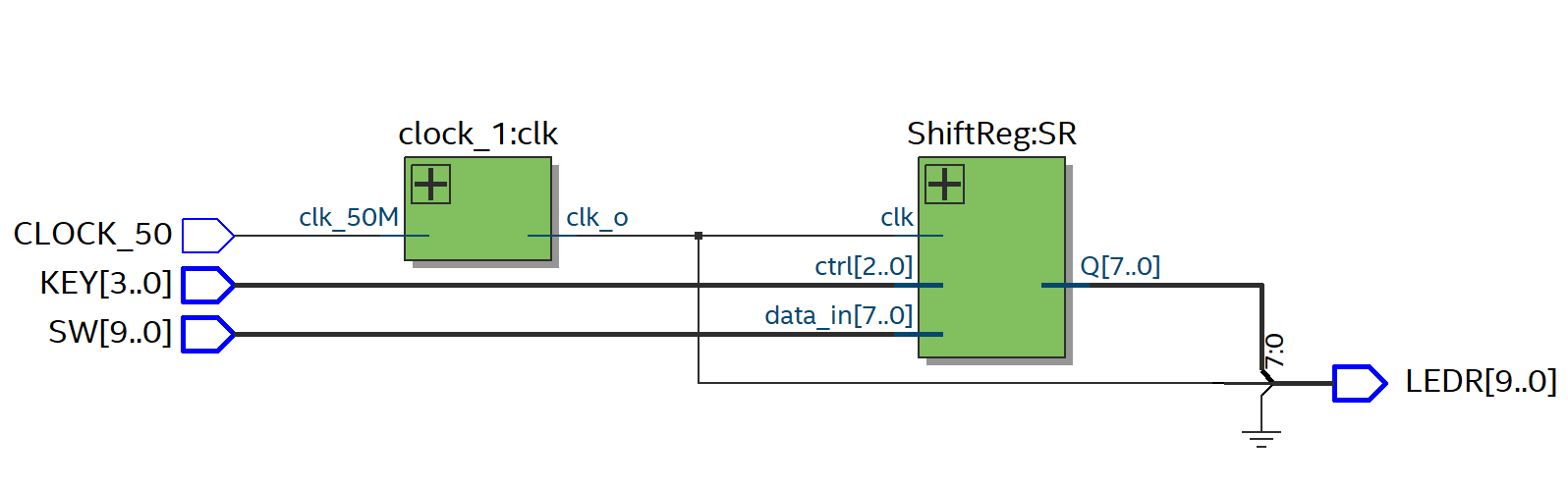
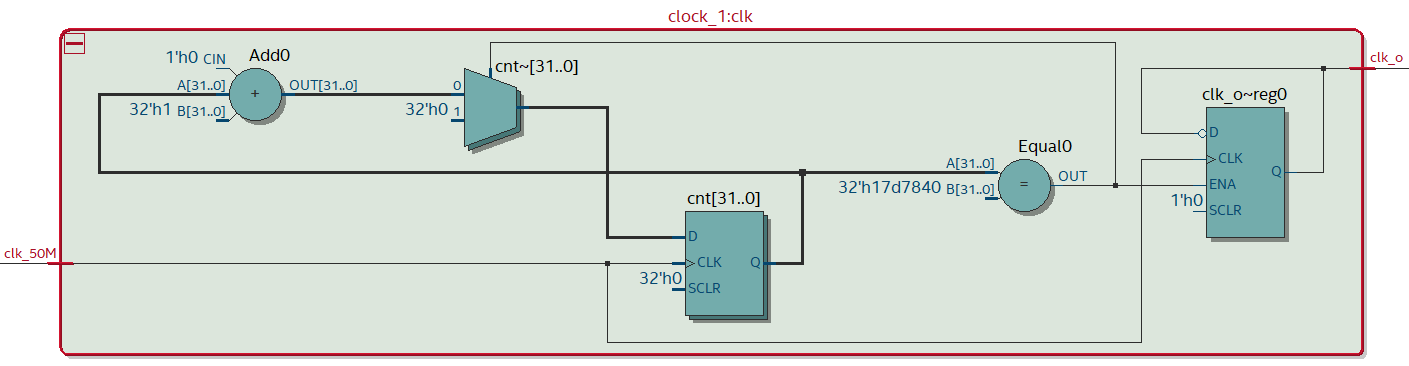


图2-1-1：分析/综合成功





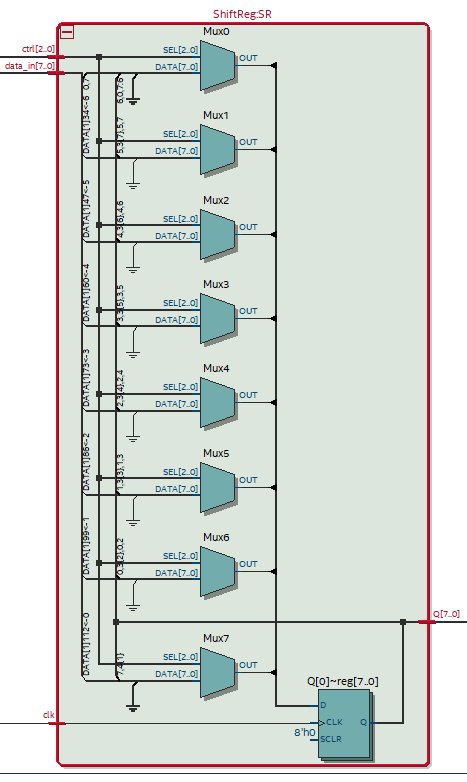


图 2-1-2:RTL视图

###### 2.1.5仿真测试

仿真测试中计数器模块的时钟信号输入与DE-10开发板的CLOCK\_50信号连接.

对于移位寄存器的各功能的相关测试样例/代码如下：

|  |  |
| --- | --- |
| 1  initial  begin  CLOCK\_50 = 0;  //置数  KEY[2:0] = 3'b001;  SW[7:0] = 8'b11011011; #7; |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7  //算数右移  //移动2位  KEY[2:0] = 3'b100; #20; //结果应为：11110110  //逻辑左移  //移动1位  KEY[2:0] = 3'b011; #10; //结果应为：11101100  //逻辑右移  //移动1位  KEY[2:0] = 3'b010; #10; //结果应为：01110110  //清零  KEY[2:0] = 3'b000; #10;  //串入并出  KEY[2:0] = 3'b101;  SW[0] = 0; #10;  SW[0] = 1; #10;  SW[0] = 1; #10;  SW[0] = 0; #10;  SW[0] = 0; #10;  SW[0] = 1; #10;  SW[0] = 0; #10;  SW[0] = 0; #10;  //结果应为：00100110  //循环右移  //移动3位  KEY[2:0] = 110; #30; //结果应为11000100  //循环左移  //移动3位  KEY[2:0] = 111; #30; //结果应为00100110  $stop;  end  always  begin  #5 CLOCK\_50 = ~CLOCK\_50;  end |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |
| 37 |  |
| 38 |  |
| 39 |  |
| 40 |  |

通过ModelSim执行上述测试代码得到的仿真结果如下：

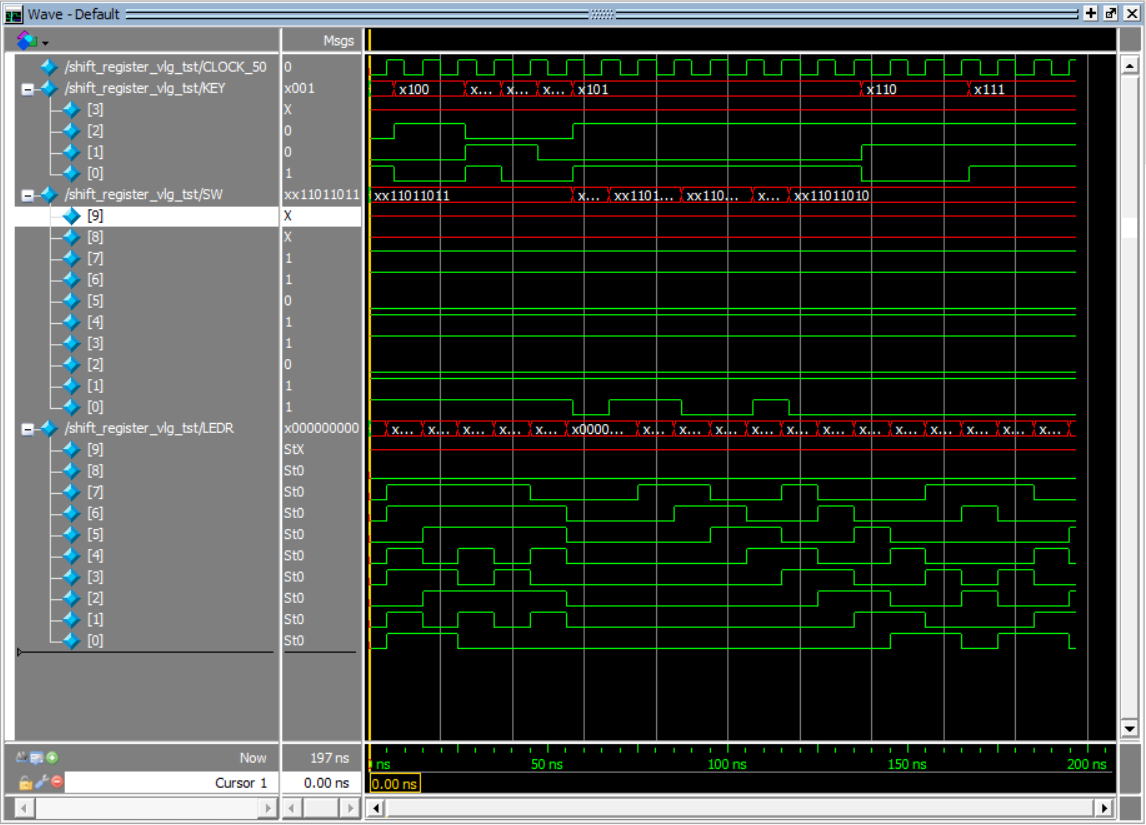
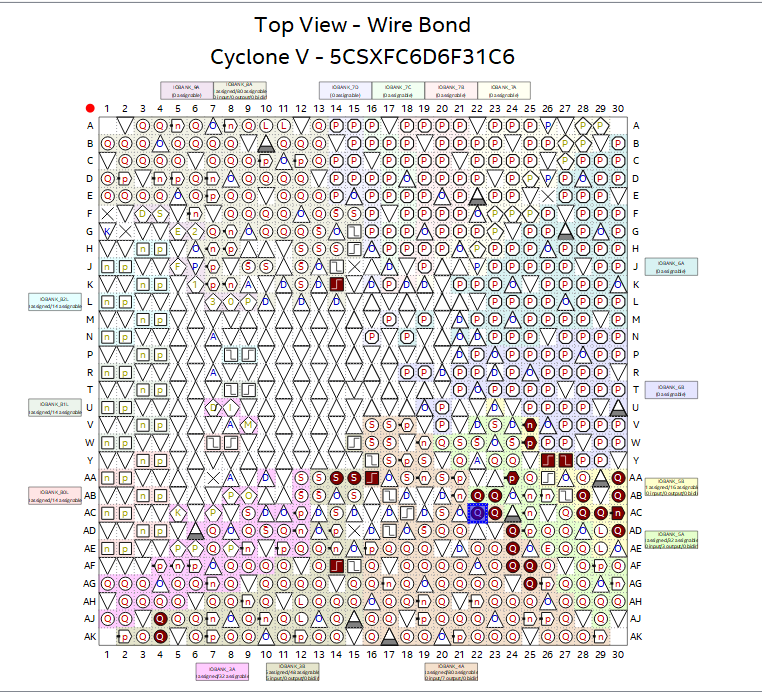


图 2-1-3仿真波形图

观察仿真波形图，将得到的LEDR[7:0]波形图与上述测试代码中的注释给出的结果相比对，发现相符合.

###### 2.1.6分配引脚

引脚分配使用DE10\_Standard\_SystemBuilder生成。



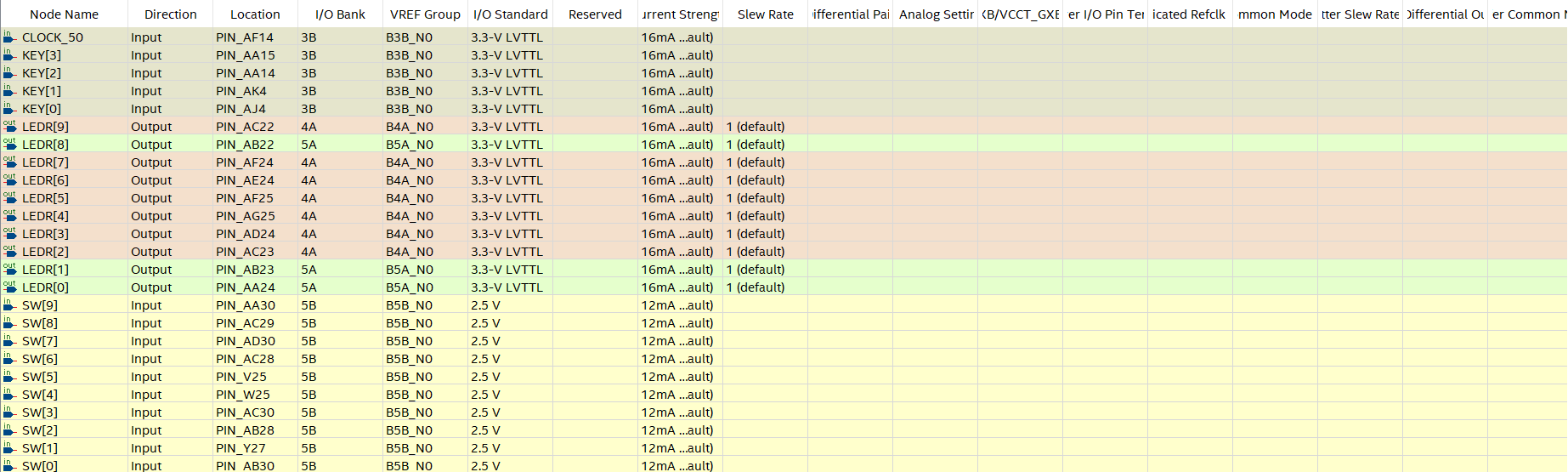


图2-1-4引脚分配图

###### 2.1.7全编译

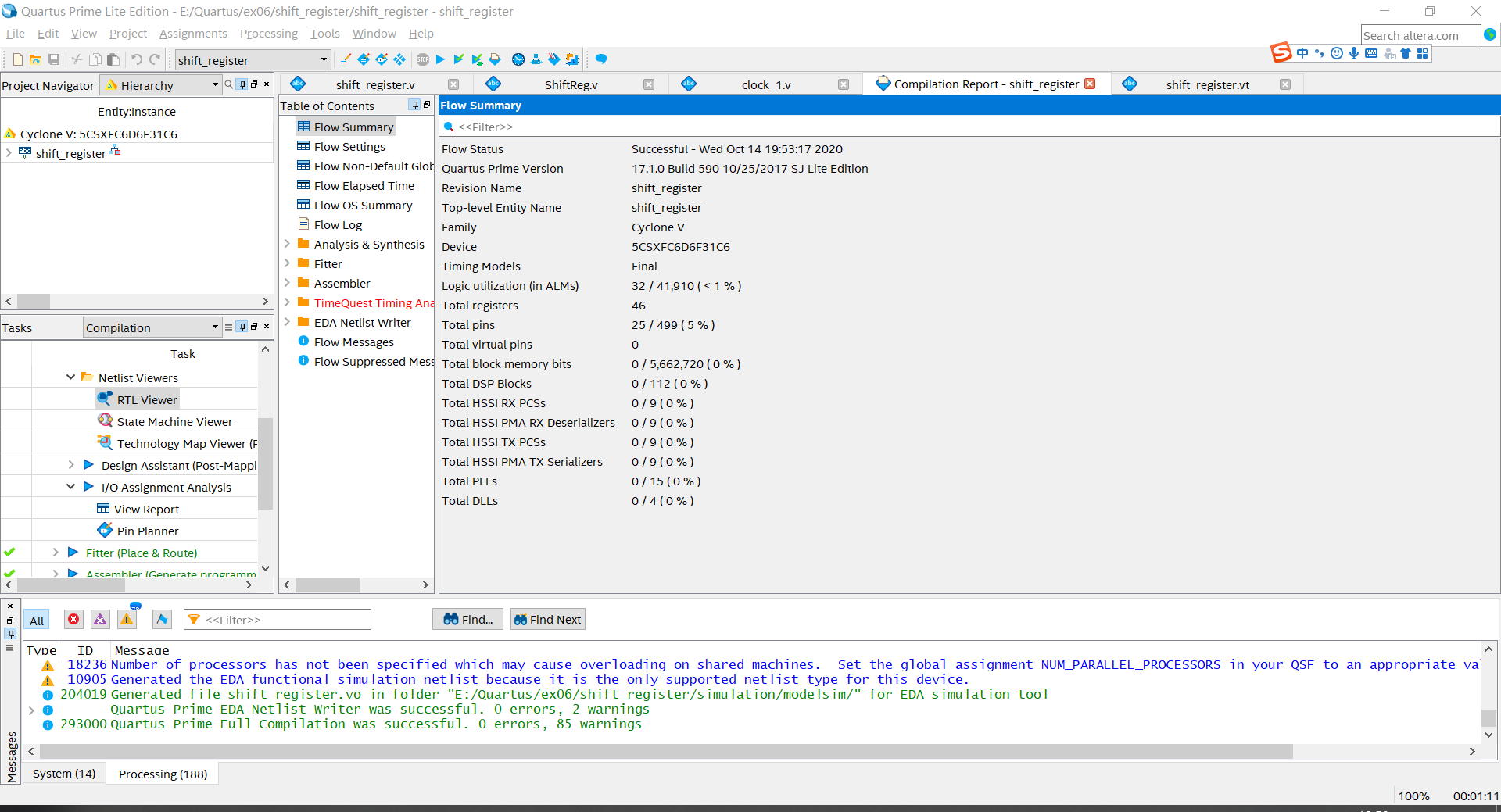


图2-1-6全编译成功

##### 2.2 利用移位寄存器实现随机数发生器

###### 2.2.1模型概述

使用 Verilog HDL 设计一个8位LFSR移位寄存器. 能够在时钟上升沿到来时随机生成0~254中的某一个数. 能够自启动. 随机数将以十六进制的形式显示在七段数码管上. 采用分频器产生的1Hz时钟信号工作. 具有置数功能和清零功能.

###### 2.2.2数字抽象

1. 输入:

时钟信号clk： 与分频器的输出时钟信号连接.

清零信号rs\_n: 同步清零信号，低有效.

置数信号load： 同步置数，为1时读入data\_in.

置数输入[7:0]data\_in： 置数的数据输入.

1. 输出:

数据输出[7:0]Q： 随机数输出.

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 | clk | LEDR[9]① |
| rs\_n | SW[9] |
| load | SW[8] |
| [7:0]data\_in | SW[7:0] |
| 输出 | [7:0]Q | LEDR[7:0] |

①:LEDR[9]为分频器产生的1Hz时钟信号输出

表 2-2-1:电子时钟输入输出信号与DE10平台信号对应关系

###### 2.2.3建立模型

**实现思路：**参考教科书P534LFSR反馈方程得到8位LFSR的递推方程为X8=X4⊕X3⊕X2⊕X0. 根据该方程，8位LFSR移位寄存器的Verilog HDL实现如下：

module RanGen(clk, rs\_n, load, data\_in, Q);

input clk;

input rs\_n;

input load;

input [7:0]data\_in;

output reg [7:0]Q;

always @(posedge clk) begin

if(!rs\_n) //清零信号

Q <= 0;

else if(load) //载入

Q <= data\_in;

else begin //线性移位

//X8=X4⊕X3⊕X2⊕X0

//加入全0状态

Q <= {(Q[4]^Q[3]^Q[2]^Q[0]) ^ (~(|Q[7:0])), Q[7:1]};

end

end

endmodule

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

###### 2.2.4分析/综合

分析/综合实验成功，如下图所示：

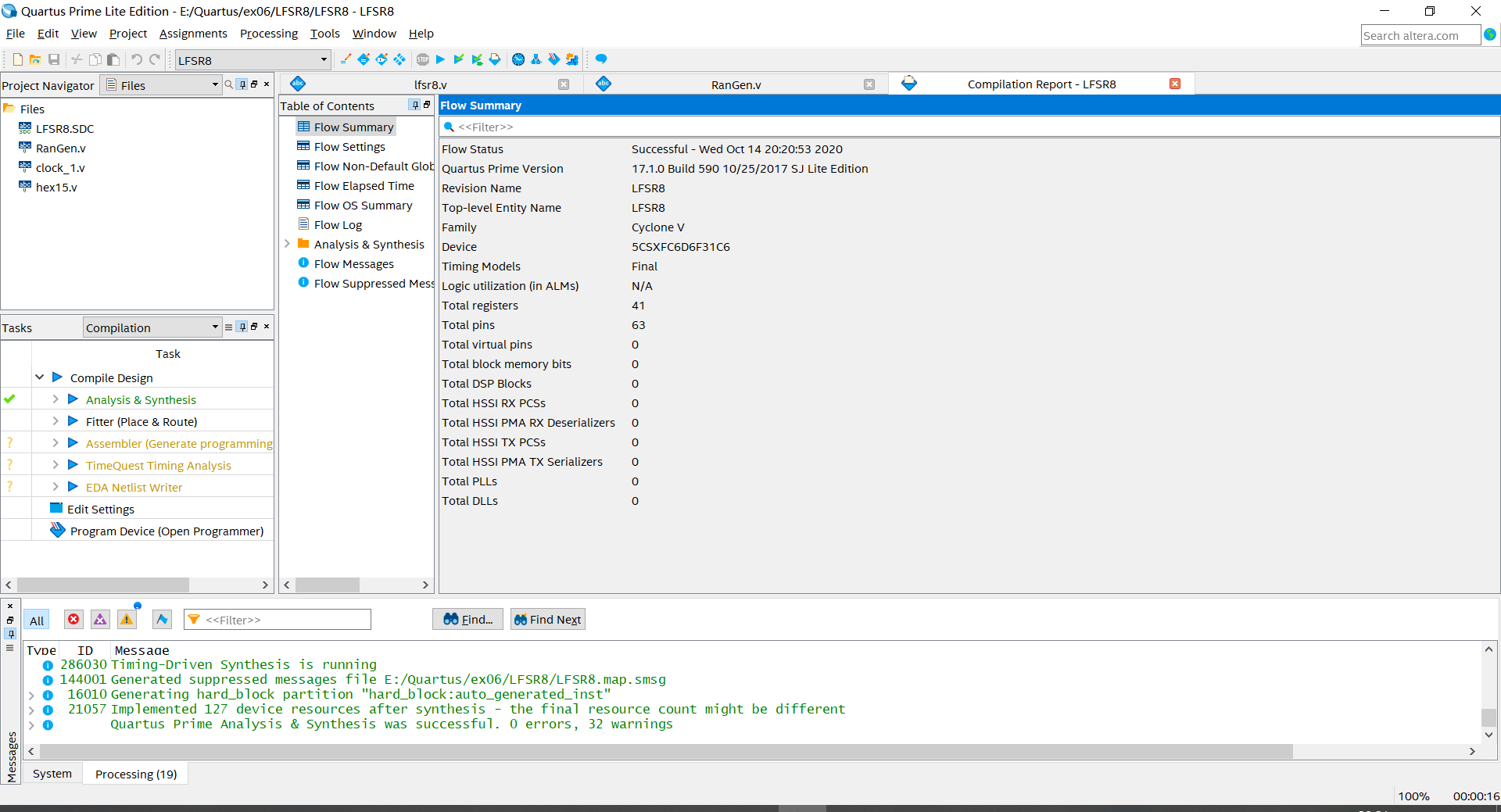
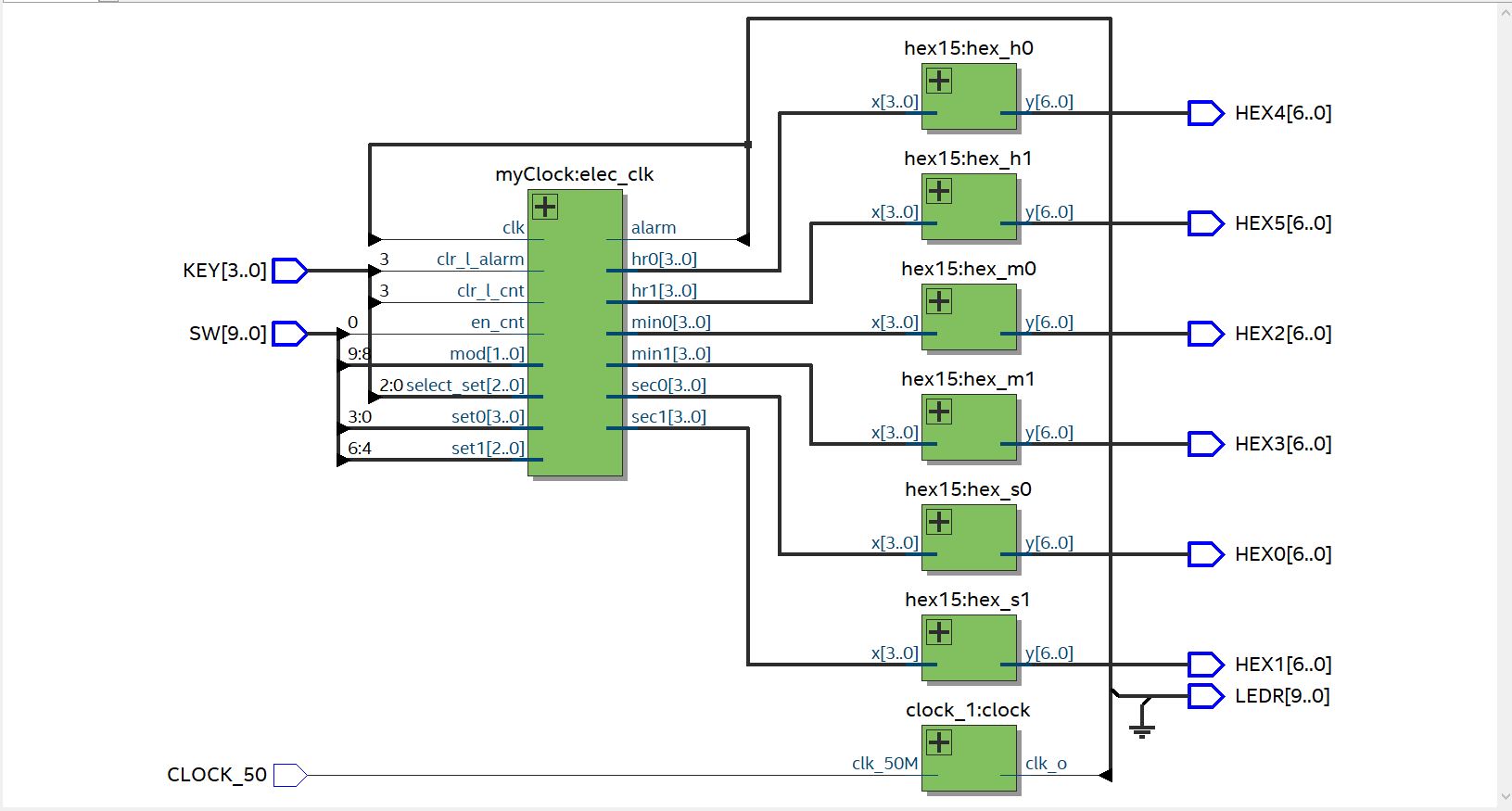
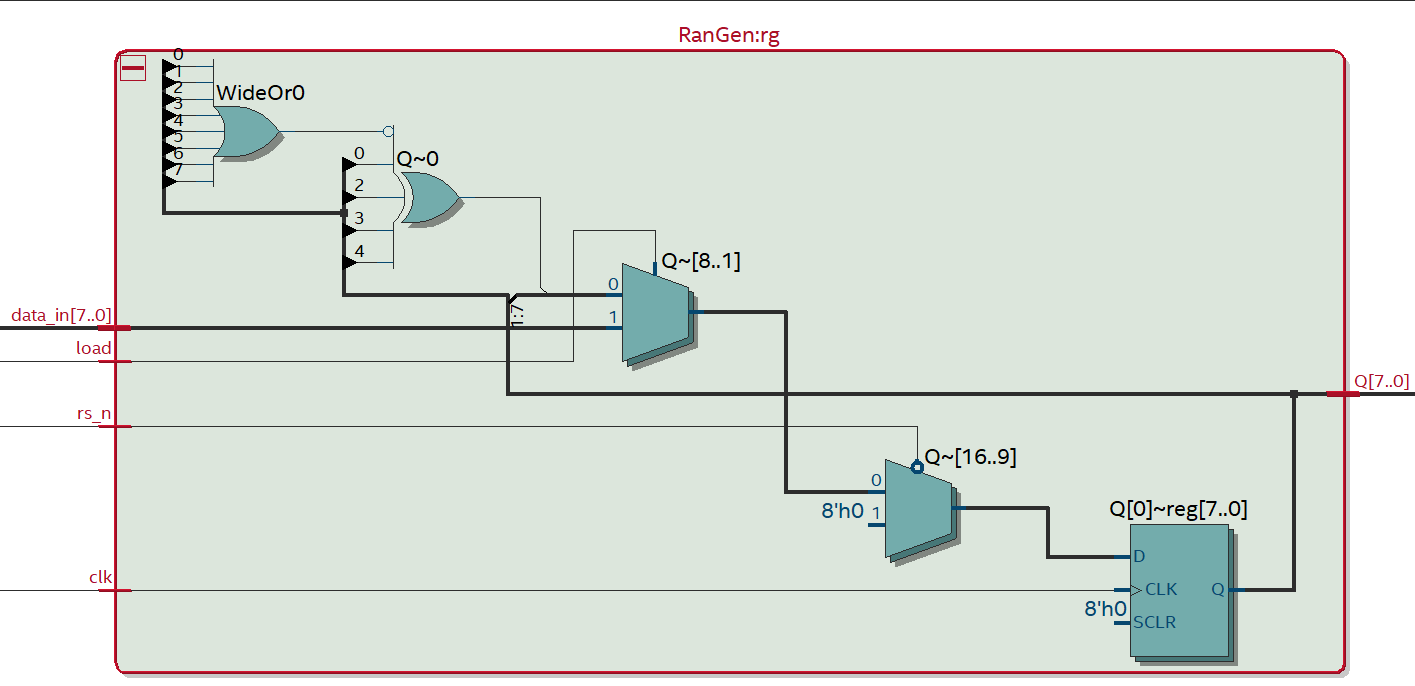
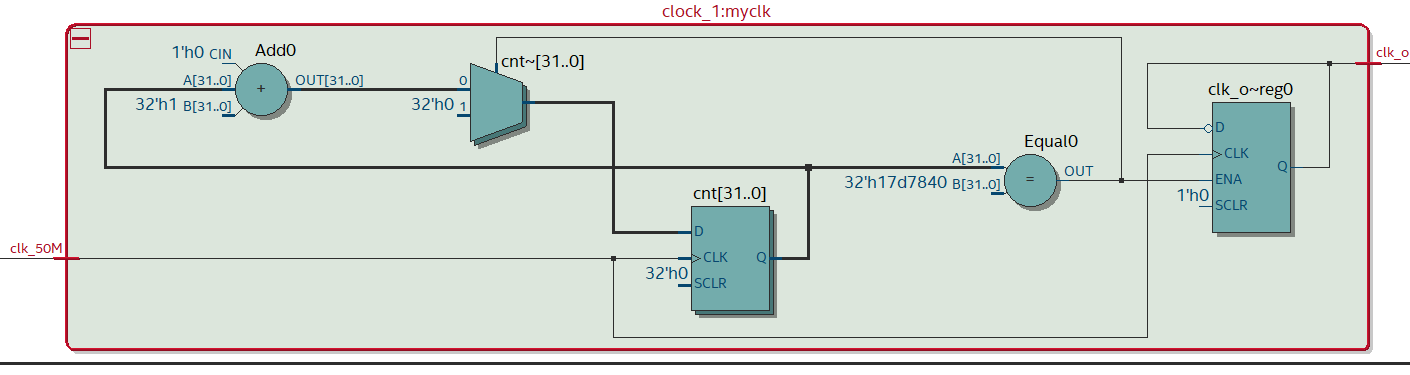


图2-2-1：分析/综合成功





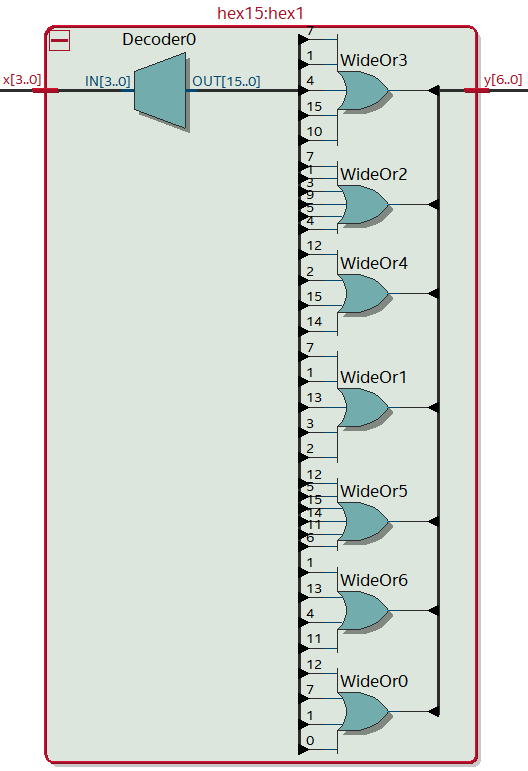


图 2-2-2:RTL视图

###### 2.2.5仿真测试

由于该模型生成的随机数有255个，仿真测试结果的可读性较差，不对该模型进行仿真测试.

###### 2.2.6分配引脚

引脚分配使用DE10\_Standard\_SystemBuilder生成。

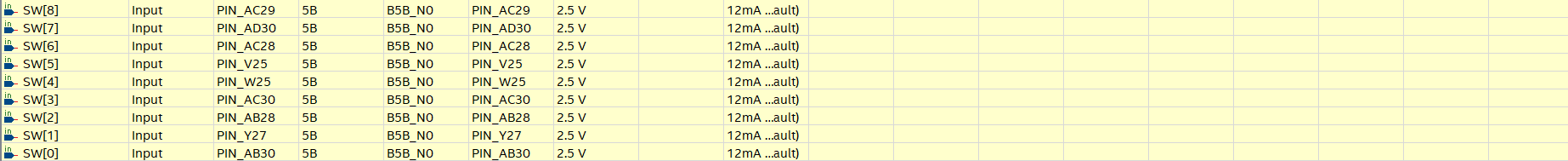
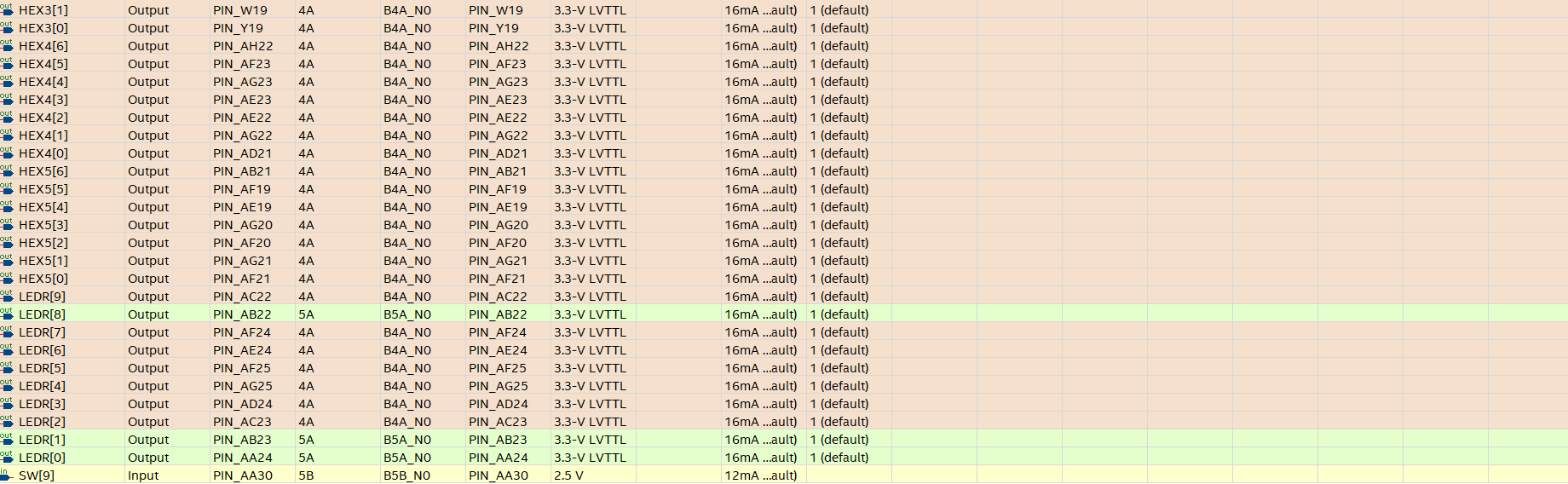
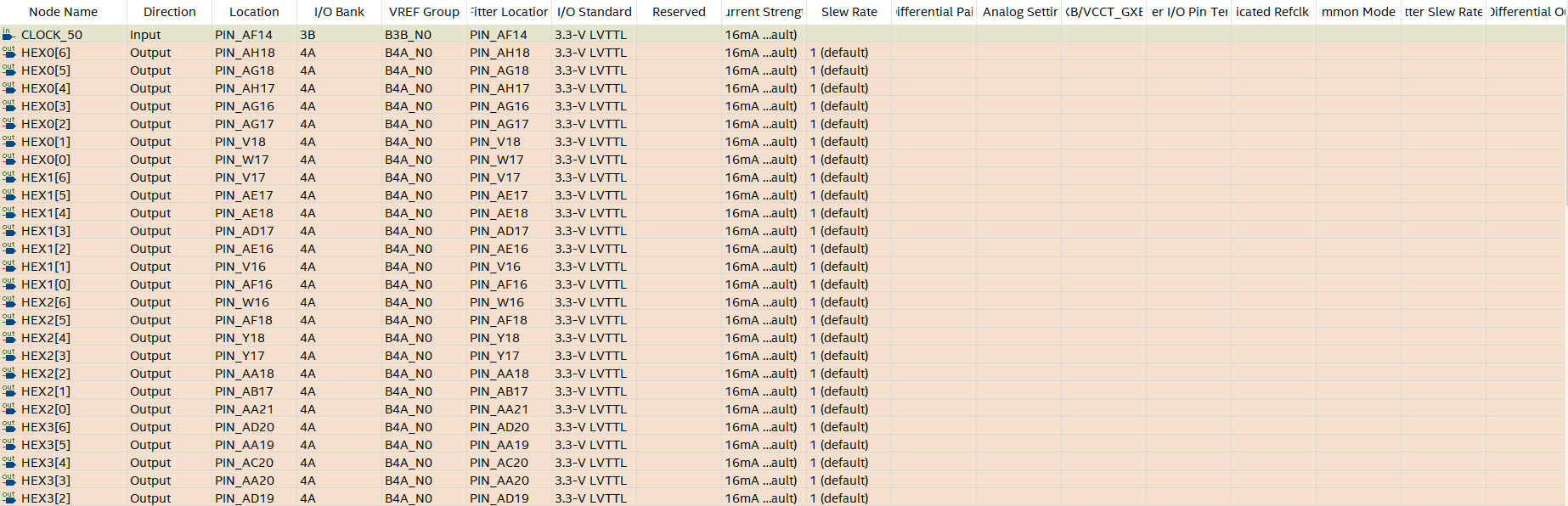
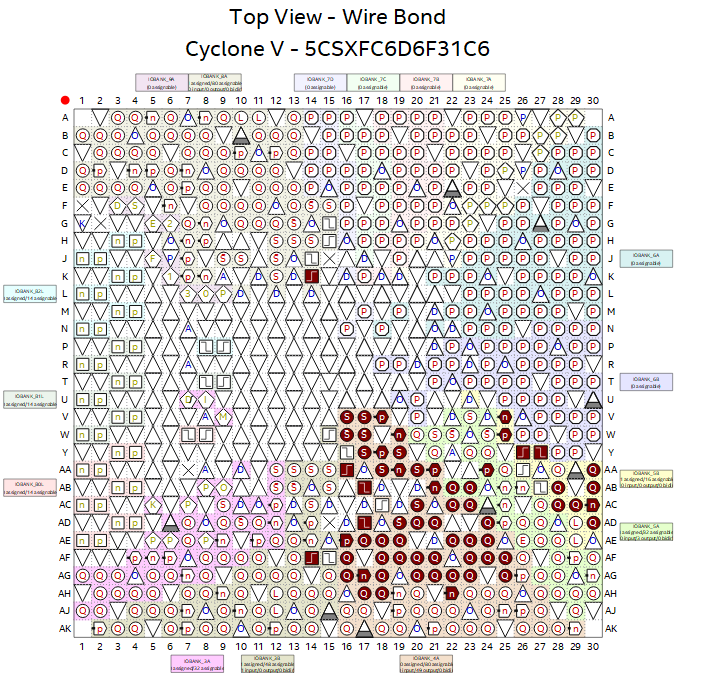


图2-2-3引脚分配图

###### 2.2.7全编译

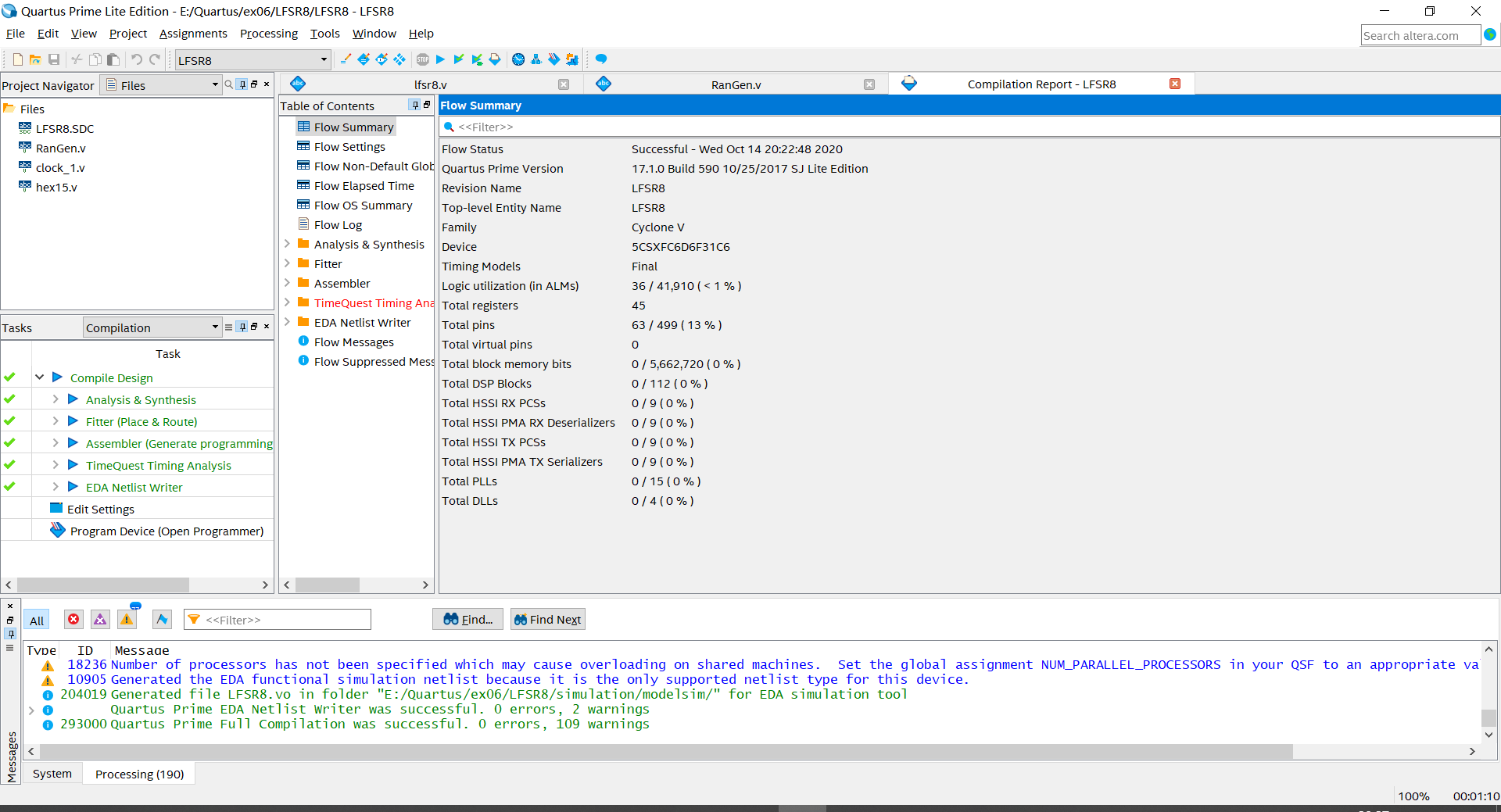


图2-2-4全编译成功

#### 实验总结

本次实验主要学习实践了移位寄存器的设计和应用，同时了解了Switch的按键消抖的原因和原理. 本实验报告中的模型的时钟输入选用1Hz分频器的输出信号，故没有实际使用.

实验的第二部分实现了将移位寄存器改造为随机数生成器，但LFSR反馈方程对于随机数的生成仍有一定的规律可循.

#### 附

##### 4.1 关于思考题

*生成的伪随机数序列仍然有一定的规律，如何能够生成更加复杂的伪随机数序列？*

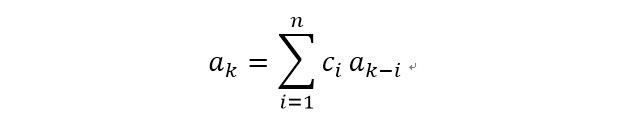
答：下面给出两种其他构造随机数序列的方式：

①：

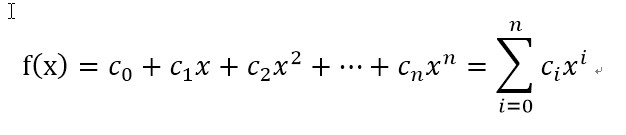
可根据m序列的相关原理构造伪随机数序列.

m序列相关原理：

递推方程：



特征方程：



若一个n次多项式f(x)满足下列条件：  
1）f(x)为既约的；  
2）f(x)可整除(x^m +1),m=2^n -1；  
3）f(x)除不尽(x^q +1),q<m；  
则称f(x)为本原多项式。

8位二进制数的本原多项式为 x8+x4+x3+x2+1. 由此可以得到新的随机数生成方式如下：（修改2.2.3中的代码第16行）

|  |  |
| --- | --- |
| 16 | Q <= {(Q[4]^Q[5]^Q[6]^Q[0]), Q[7:1]}; |

该随机数生成方式需要在开始前对Q进行初始化为一个非全0数.

②：

可以通过外接设备收集环境噪音再加以一定的计算实现随机数序列.随机性更强. 由于该方式的实现复杂，故实现方式不在此报告中具体给出.