CRC算法实现

# 概述

CRC即循环冗余校验码（Cyclic Redundancy Check）：是[数据通信](https://baike.so.com/doc/6408448-6622113.html)领域中最常用的一种查错[校验码](https://baike.so.com/doc/6735310-6949689.html)，其特征是信息字段和校验字段的长度可以任意选定。

循环冗余[校验码](https://baike.so.com/doc/6735310.html)（CRC）的基本原理是：在K位[信息码](https://baike.so.com/doc/56725.html)后再拼接R位的校验码，整个编码长度为N位，因此，这种编码也叫（N，K）码。对于一个给定的（N，K）码，可以证明存在一个[最高次幂](https://baike.so.com/doc/1886217.html)为N-K=R的多项式G(x)。根据G(x)可以生成K位信息的校验码，而G(x)叫做这个CRC码的生成多项式。 校验码的具体生成过程为：假设发送信息用信息多项式C(X)表示，将C(x)左移R位，则可表示成C(x)\*x的R次方，这样C(x)的右边就会空出R位，这就是校验码的位置。通过C(x)\*X的R次方除以生成多项式G(x)得到的余数就是校验码。

# 校验原理

CRC校验算法其根本思想就是先在要发送的帧后面附加一个数（这个就是用来校验的校验码，但要注意，这里的数也是二进制序列的，下同），生成一个新帧发送给接收端。当然，这个附加的数不是随意的，它要使所生成的新帧能与发送端和接收端共同选定的某个特定数整除（注意，这里不是直接采用二进制除法，而是采用一种称之为“模2除法”）。到达接收端后，再把接收到的新帧除以（同样采用“模2除法”）这个选定的除数。因为在发送端发送数据帧之前就已通过附加一个数，做了“去余”处理（也就已经能整除了），所以结果应该是没有余数。如果有余数，则表明该帧在传输过程中出现了差错。

# CRC校验码求法

现假设选择的CRC生成多项式为G（X） = X4 + X3 + 1，要求出二进制序列10110011的CRC校验码。下面是具体的计算过程：

①将多项式转化为二进制序列，由G（X） = X4 + X3 + 1可知二进制一种有五位，第4位、第三位和第零位分别为1，则序列为11001

②多项式的位数位5，则在数据帧的后面加上5-1位0，数据帧变为101100110000，然后使用模2除法除以除数11001，得到余数。

③将计算出来的CRC校验码添加在原始帧的后面，真正的数据帧为101100110100，再把这个数据帧发送到接收端。

④接收端收到数据帧后，用上面选定的除数，用模2除法除去，验证余数是否为0，如果为0，则说明数据帧没有出错。

## 模二除法

110110110101

——————————

11001 /101100110000

11001

--------------------

011110

11001

--------------------

001111

11001

--------------------

10110

11001

--------------------

011111

11001

--------------------

001100

11001

--------------------

10101

11001

--------------------

011000

11001

--------------------

000010

11001

--------------------

11011

11001

--------------------

000100

11001

--------------------

11101

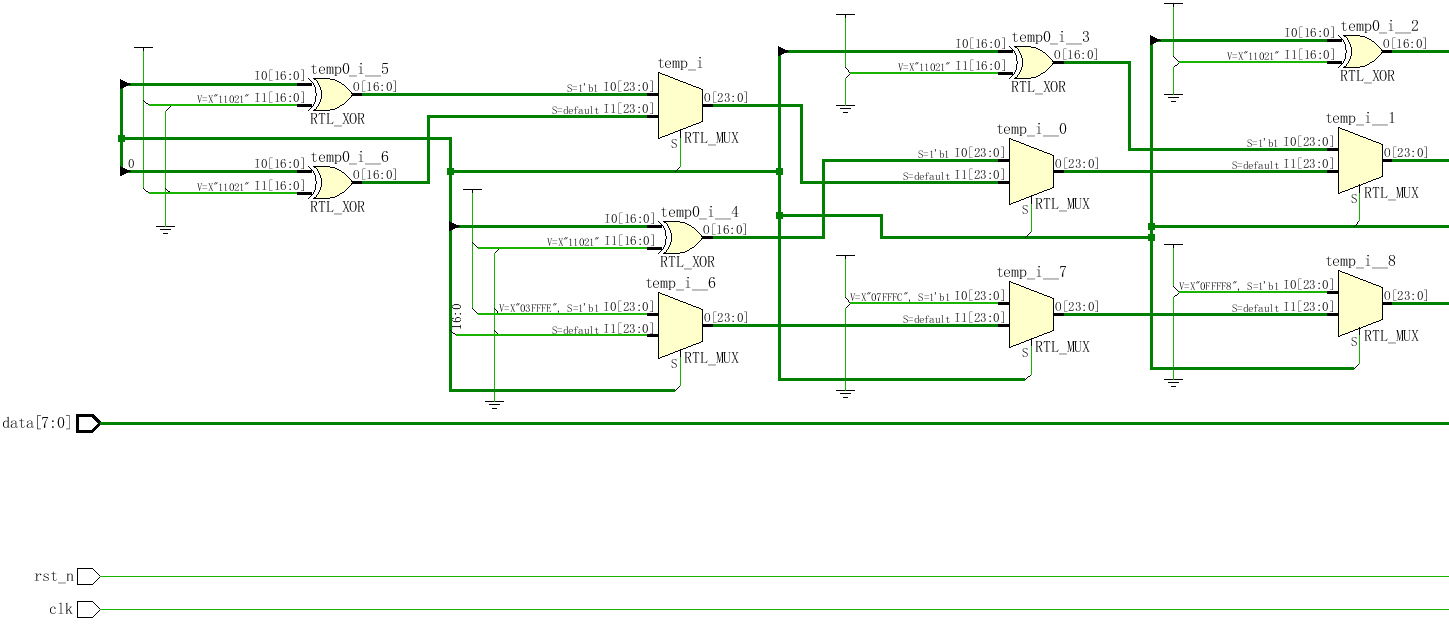
11001

--------------------

0100

# verilog实现CRC算法

本设计中实现了求得8bit信息序列的CRC校验码，生成多项式取g(x)=X^16+X^12+X^5+1，对应的生成序列为1\_0001\_0000\_0010\_0001，输入的8bit序列data左移16位后得到stemp={data，16‘b0000\_0000\_0000\_0000} ，每次异步复位信号rst\_n复位时将croc的值清零并把线型变量stemp的值打入寄存器temp中，然后通过时序电路将temp与生成多项式对应的序列进行多次按位异或，最终得到一个小于生成序列的temp后，temp[15：0]的值即为CRC校验序列，并把它赋给输出croc。本设计顶层RTL视图如下：

**

## CRC算法实现代码

module crc3 (clk,data,rst\_n,crc );

input [7:0] data;

input clk;

input rst\_n;

output reg [15:0] croc=0;

wire [23:0] stemp;

reg [23:0] temp=0;

parameter polynomial=17'b1\_0001\_0000\_0010\_0001;

assign stemp={data,16'b0000000000000000};

always @ (posedge clk or negedge rst\_n)

begin

if(!rst\_n)

begin

croc<=0;

temp<=stemp;

end

else

begin

if(temp[23]) temp[23:7]<=temp[23:7]^polynomial;

else if(temp[22]) temp[22:6]<=temp[22:6]^polynomial;

else if(temp[21]) temp[21:5]<=temp[21:5]^polynomial;

else if(temp[20]) temp[20:4]<=temp[20:4]^polynomial;

else if(temp[19]) temp[19:3]<=temp[19:3]^polynomial;

else if(temp[18]) temp[18:2]<=temp[18:2]^polynomial;

else if(temp[17]) temp[17:1]<=temp[17:1]^polynomial;

else if(temp[16]) temp[16:0]<=temp[16:0]^polynomial;

else croc<=temp[15:0];

end

end

endmodule

## 测试脚本

`timescale 1ns/1ps

module tb\_crc3();

reg clk;

reg [7:0] data;

reg rst\_n;

wire [15:0] crc;

crc3 crc3\_inst (

.clk (clk),

.data (data),

.rst\_n (rst\_n),

.crc (crc)

);

initial

begin

clk=0;

end

initial

begin

rst\_n=0;

data=8'b0;

#1000;

data=8'b10110110;

rst\_n=1;

#50;

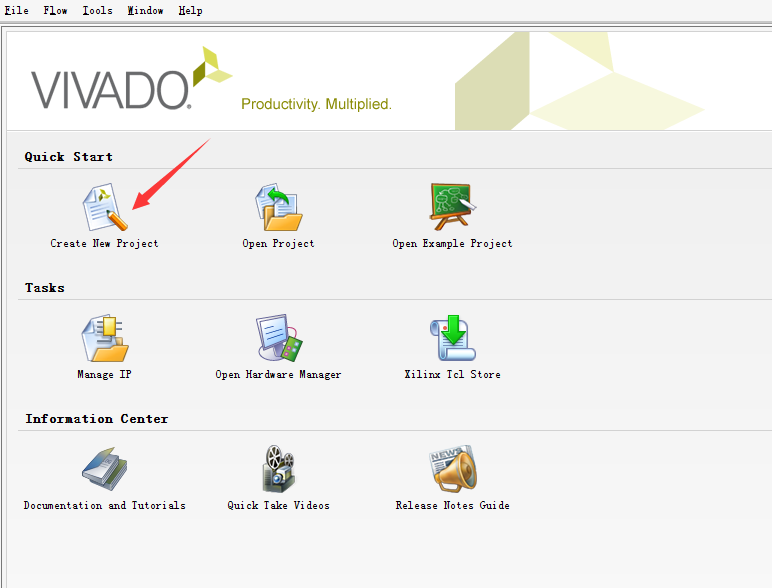
rst\_n=0;

#50;

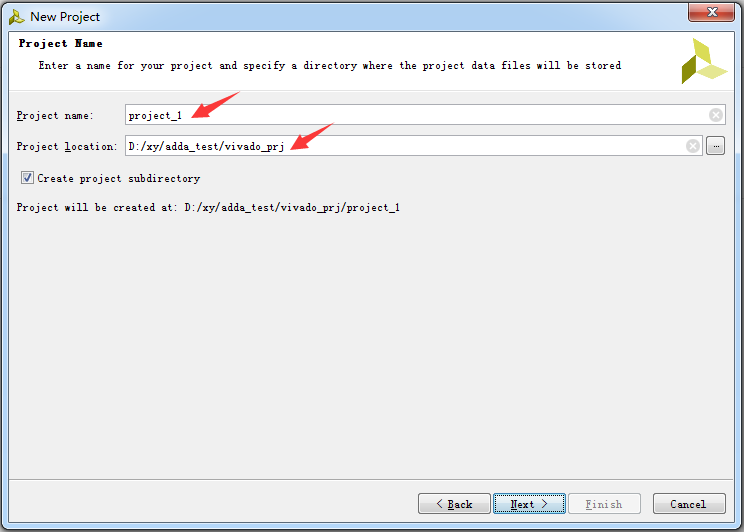
rst\_n=1;

# VIVADO CRC算法工程创建

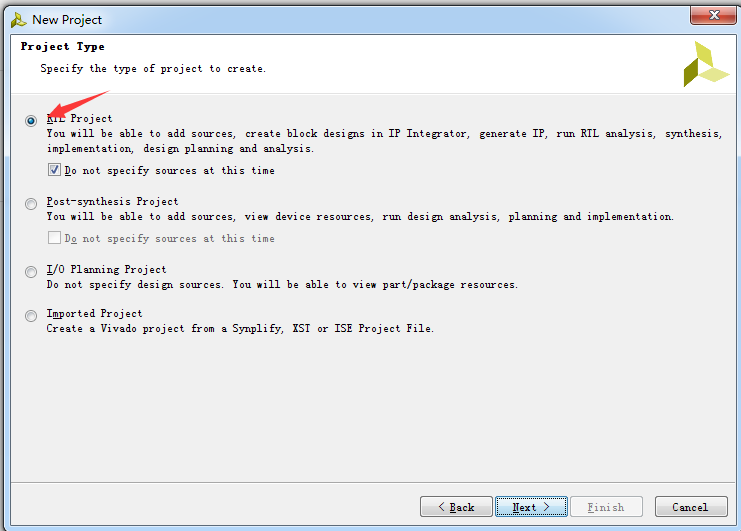
①双击进入VIVADO软件界面，选择“Creat new project”



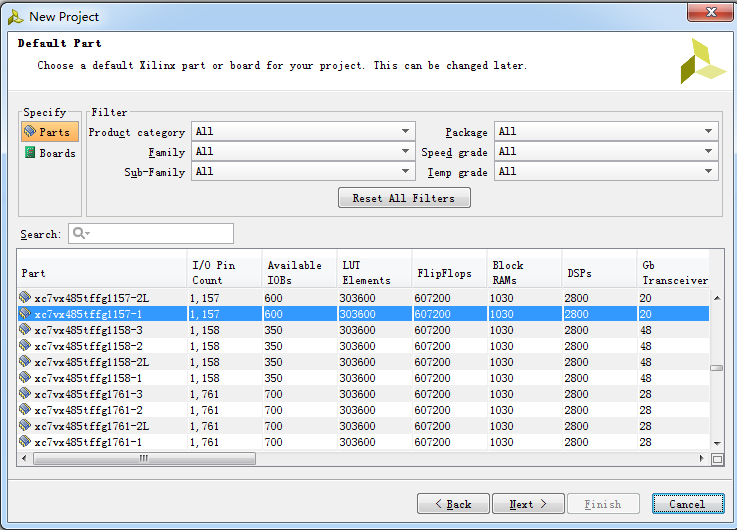
②在“Project name”界面，输入工程名，和工程文件路径。点击进入下一步；



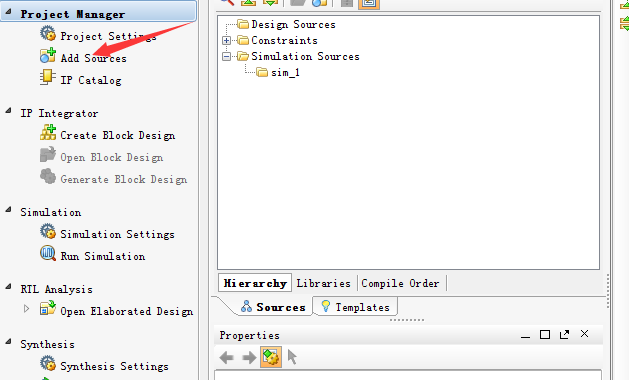
③在“Project type”界面，选择“RTL project”。点击“Next”进入下一步。



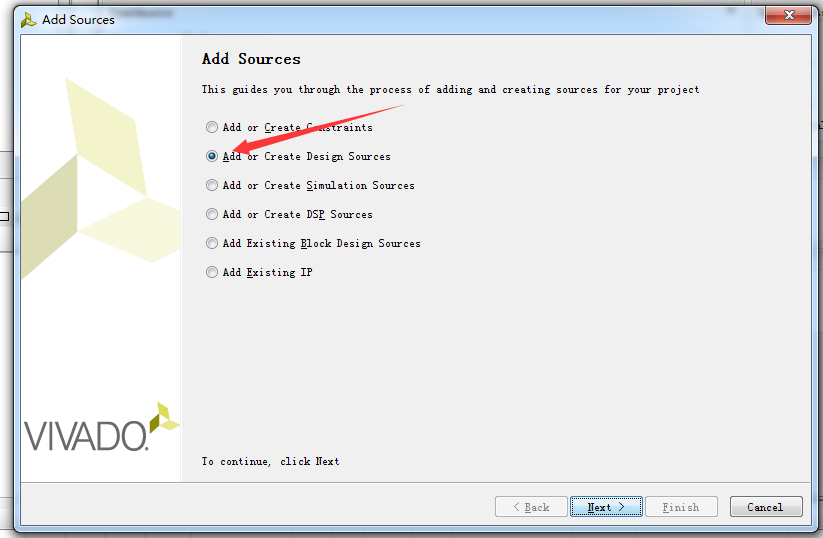
④进入芯片型号选择界面，这里选择“XC7S25FTGB196-1”芯片,然后进入下一步；



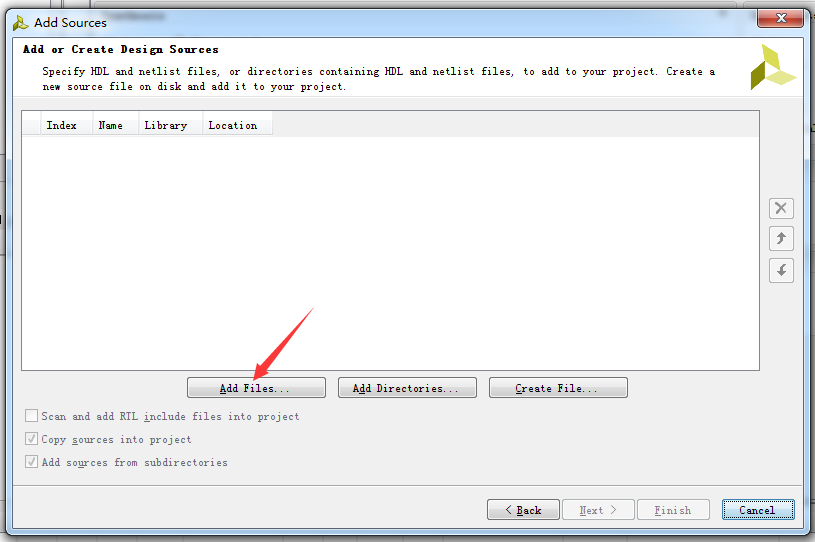
⑤工程创建完成，随后向工程内添加设计文件。



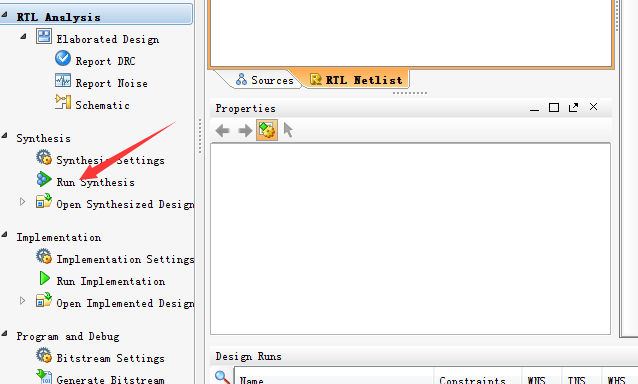
⑥选择“Add or Creat Design Sources”，添加CRC算法实现代码文件。



⑦因为在工程创建前，已在第三方文本编辑软件“Ultraedit”上完成设计文件的编写，所以在“Add source”界面中，我们选择“Add files”将设计文件路径指定到工程中；如果没有设计文件，我们选择“Creat file”新建一个设计文件；

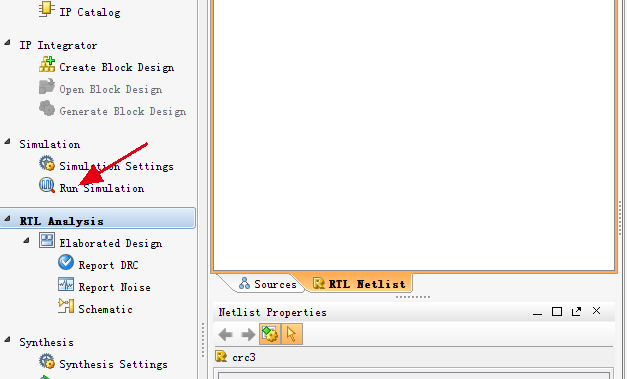


⑧选择“Run Synthesis”进行编译，如编译通过，证明设计语法正确。

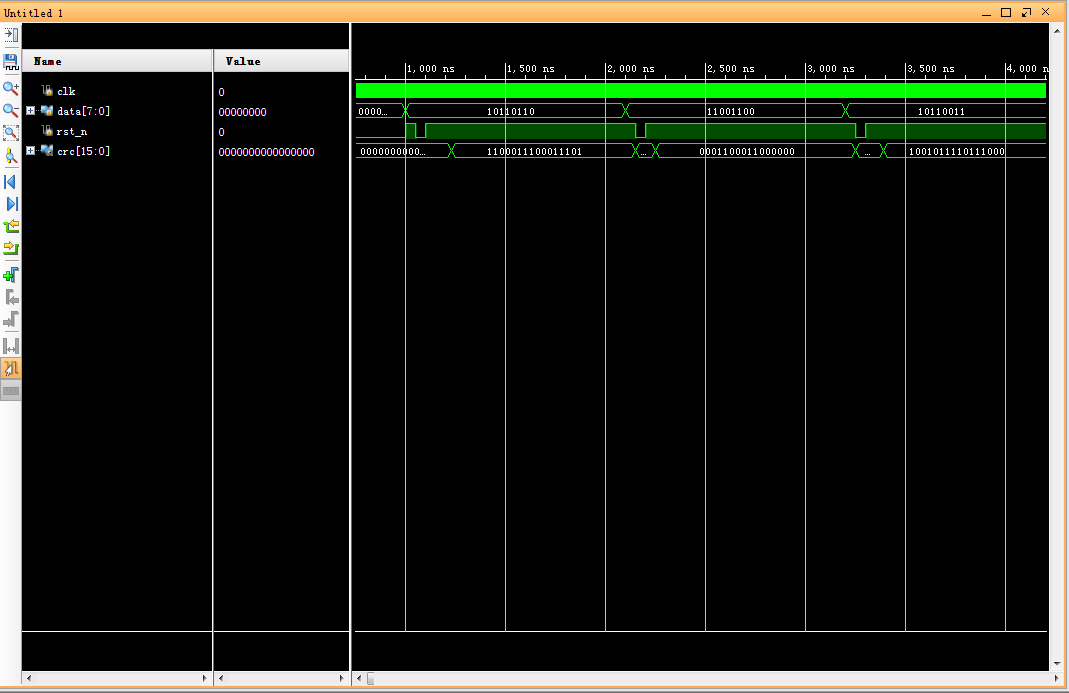


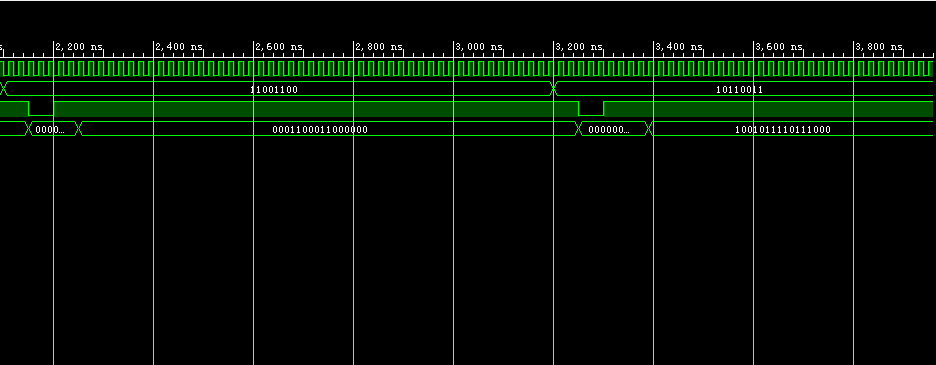
⑨根据①----⑦步骤将仿真文件“tb\_crc3.v”添加到工程。

⑩选择“Run Simulation”进行算法的仿真验证。



仿真波形如下图所示：





通过仿真波形图可以看出，在输入8位数据位10110110时，经过CRC校验算法后，输出数据位16位1100011100011101，输入11001100时，输出0001100011000000，根据第三节“CRC校验码求法”进行计算，与仿真结果一致，证明本设计成功实现了CRC算法功能。