**FPGA综合设计报告**

**Xilinx哈夫曼编码**

学 院（系）： 软件学院

专 业： 软件工程

学 生 姓 名： 曹煜政

学 号： 201593136

指 导 教 师： 王洁

评 阅 教 师： 王洁

完 成 日 期： 2018/6/27

大连理工大学

Dalian University of Technology

# 1原理分析与算法设计

## 1.1 Huffman Tree简介

    赫夫曼树（Huffman Tree），又称最优二叉树，是一类带权路径长度最短的树。假设有n个权值{w1,w2,...,wn}，如果构造一棵有n个叶子节点的二叉树，而这n个叶子节点的权值是{w1,w2,...,wn}，则所构造出的带权路径长度最小的二叉树就被称为赫夫曼树。

   树的带权路径长度的概念：树的带权路径长度指树中所有叶子节点到根节点的路径长度与该叶子节点权值的乘积之和，如果在一棵二叉树中共有n个叶子节点，用Wi表示第i个叶子节点的权值，Li表示第i个也叶子节点到根节点的路径长度，则该二叉树的带权路径长度 WPL=W1\*L1 + W2\*L2 + ... Wn\*Ln。

    根据节点的个数以及权值的不同，赫夫曼树的形状也各不相同，赫夫曼树具有如下特性：

* 对于同一组权值，所能得到的赫夫曼树不一定是唯一的。
* 赫夫曼树的左右子树可以互换，因为这并不影响树的带权路径长度。
* 带权值的节点都是叶子节点，不带权值的节点都是某棵子二叉树的根节点。
* 权值越大的节点越靠近赫夫曼树的根节点，权值越小的节点越远离赫夫曼树的根节点。
* 赫夫曼树中只有叶子节点和度为2的节点，没有度为1的节点。
* 一棵有n个叶子节点的赫夫曼树共有2n-1个节点。

## 1.2 Huffman Tree的构建

    赫夫曼树的构建步骤如下：

    1、将给定的n个权值看做n棵只有根节点（无左右孩子）的二叉树，组成一个集合HT，每棵树的权值为该节点的权值。

    2、从集合HT中选出2棵权值最小的二叉树，组成一棵新的二叉树，其权值为这2棵二叉树的权值之和。

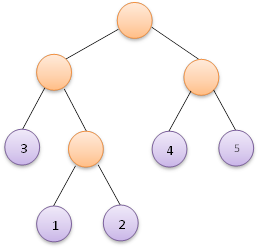
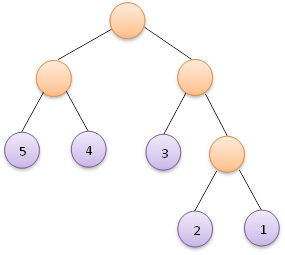
    3、将步骤2中选出的2棵二叉树从集合HT中删去，同时将步骤2中新得到的二叉树加入到集合HT中。

    4、重复步骤2和步骤3，直到集合HT中只含一棵树，这棵树便是赫夫曼树。

    假如给定如下5个权值：

https://img-blog.csdn.net/20140213234608531

    则按照以上步骤，可以构造出如下面左图所示的赫夫曼树，当然也可能构造出如下面右图所示的赫夫曼树，这并不是唯一的。

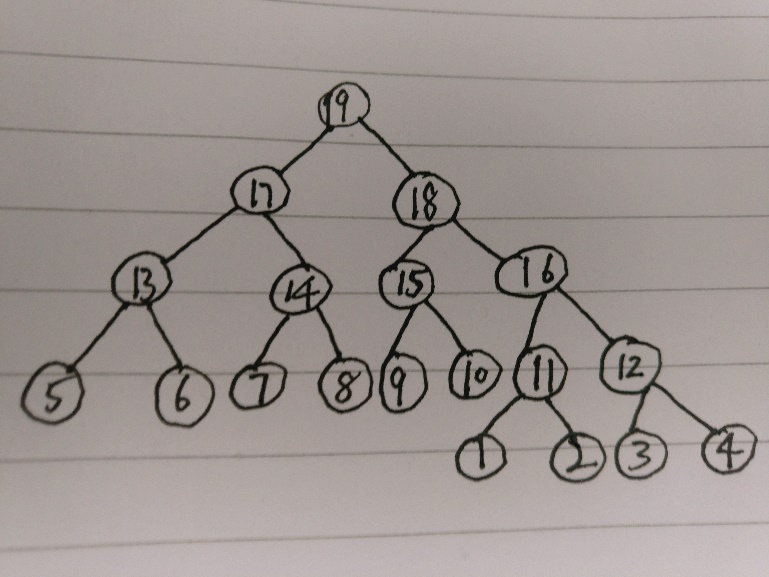
     

## 1.3编码算法设计

   先人为选定每个数字的频率，给出输入数据，然后统计频率后生成对应的树，再生成对应的编码和每个编码的长度，最后用输入的前几个数据检验算法的正确性。

频率表：0->22 1->22 2->23 3->24 4->25 5->26 6->27 7->28 8->29 9->30

Huffman树：



编码表: 0->1100 1->1101 2->1110 3->1111 4->000 5->001 6->010 7->011 8->100 9->101

前几个输入数据:999999998888888777777

前几个输出bit:101 101 101 101 101 101 101 101 100 100 100 100

100 100 100 011 011 011 011 011 011

# 2代码设计

## 2.1概述

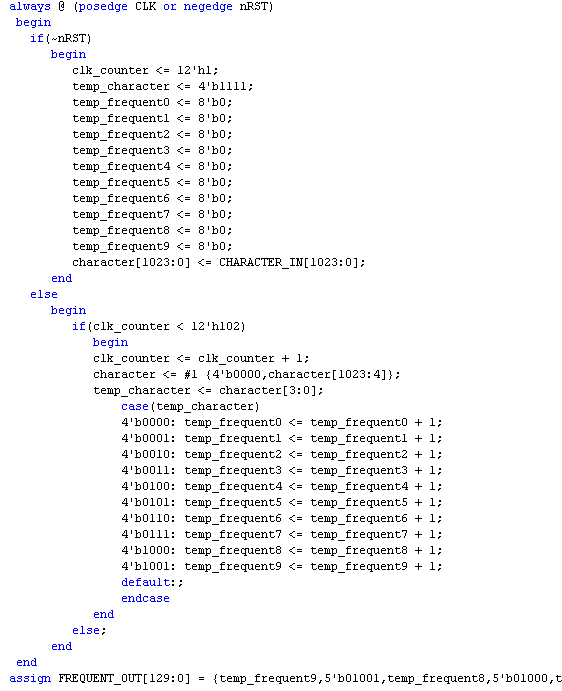
工程基于Virtex7,芯片选用XC7VX330T,速度-3。代码总共分为5个模块，共22个.v文件，每个模块对应一个test文件。只显示了部分主要代码。

## 2.2计数模块

功能：计数模块，负责将输入的256个数据进行计数，统计0~9数字出现的频率。

输入输出：输入为1024位的256个数据，输出为130位的计数后的10个数字。

逻辑设计：输出的每个数据前5位表示数字编号，后8位表示词频。

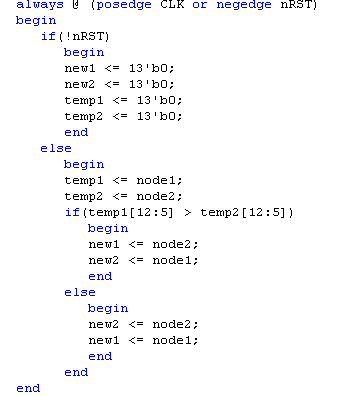


## 2.3 sort模块

功能：冒泡排序模块，将10个数字的权重进行排序

输入输出：输入为10个数据的权重，输出为排序后的10个数据的权重。

逻辑设计：首先将9个数据排序，再将第10个数据通过比较插入到前9个数中。9个数据的排序类似于10个数据排序，以此递归排序。最后是两个数据的排序。

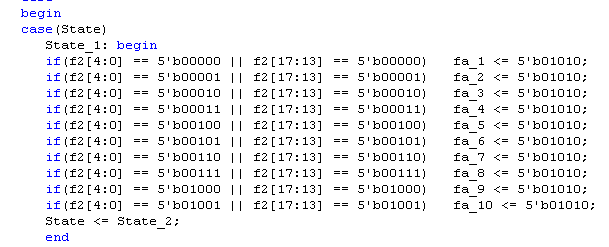
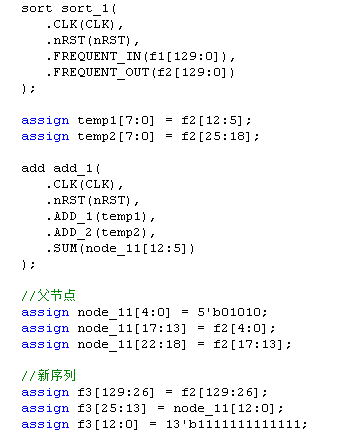


## 2.4 generate\_tree模块

功能：负责Huffman数的构建

输入输出：输入为计数后的各个数字的权重，输出为Huffman树的19个节点的信息，每个节点包括自身节点的编号，权重，左子树节点编号，右子树节点编号，父节点编号。

逻辑设计：首先对各个数字的权重进行排序。再从排序好的权重中选出最低的两个节点，进行相加操作，计算新节点的权重。新节点的编号从10开始一直到19，指定新节点的左右子树与分别为权重最低的两个节点。指定子节点的父节点为新节点的编号。再把新的权重进行排序…如此进行9次排序与相加操作后即构建出了Huffman树。

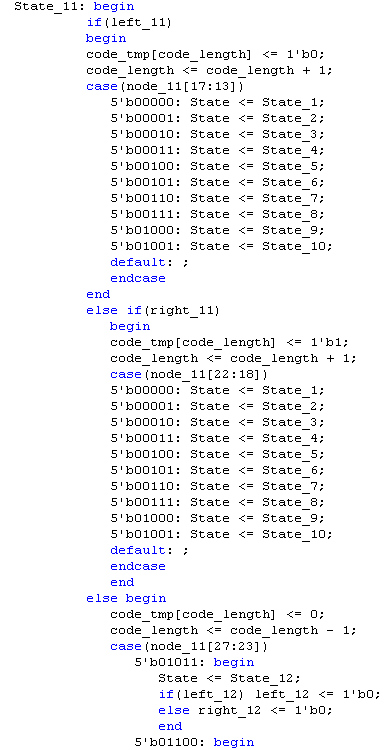
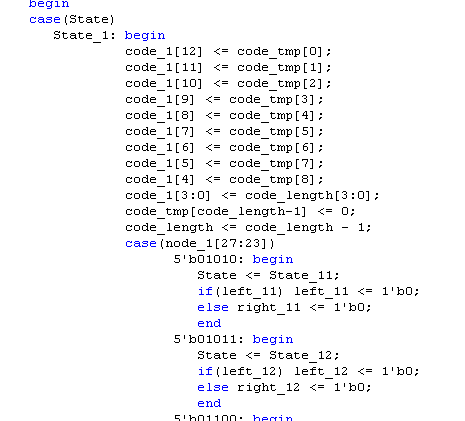


## 2.5 generate\_code模块

功能：负责按照生成的Huffman树给数字0~9进行编号

输入输出：输入为Huffman树的19个节点的信息，输出为10个数字的编码。

逻辑设计：从根节点第19个节点开始，进行状态的切换，每个节点对应一个状态，总共19个节点。 先遍历它的左子树，切换到下一个状态，并记录编码的长度，编码设置为0。在下一个状态中如果是叶子节点，则保存本次编码，返回父节点，设置父节点的左子树为已经遍历过。再遍历右子树，如果左右子树都遍历过，则返回上上个节点，并设置遍历过的标志；下一个状态如果不是叶子节点，则先向左子树遍历，增加编码长度，并设置编码为0。若左子树已经编码过，则遍历右子树。

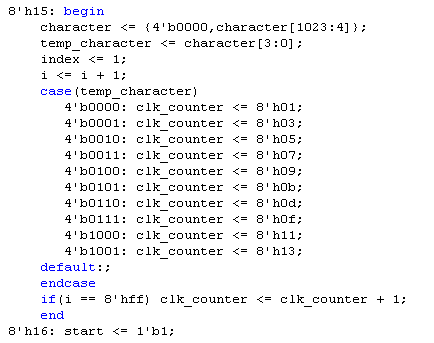
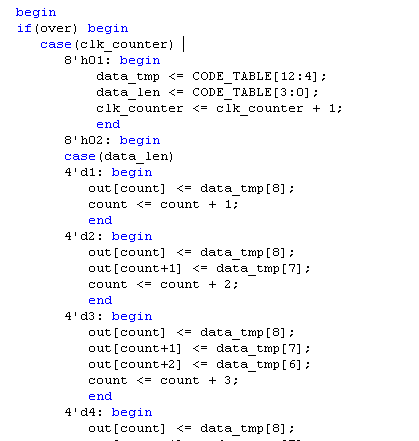


## 2.6 out\_data模块

功能：数据输出模块

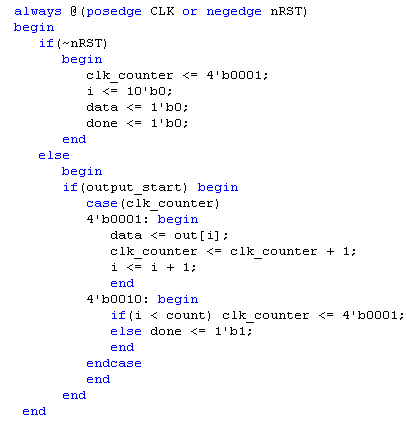
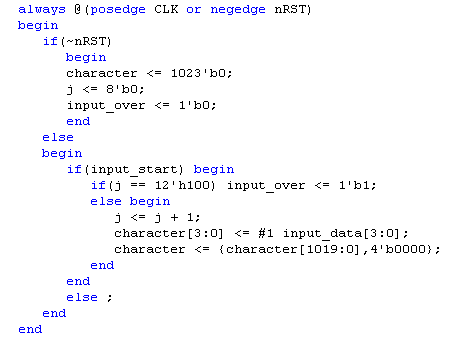
输入输出：输入是10个数字的编码和编码长度，输出是1024位编码与编码的有效长度。

逻辑设计：先依次输出10个数字编码，再输出256个数据的编码



## 2.7 main模块

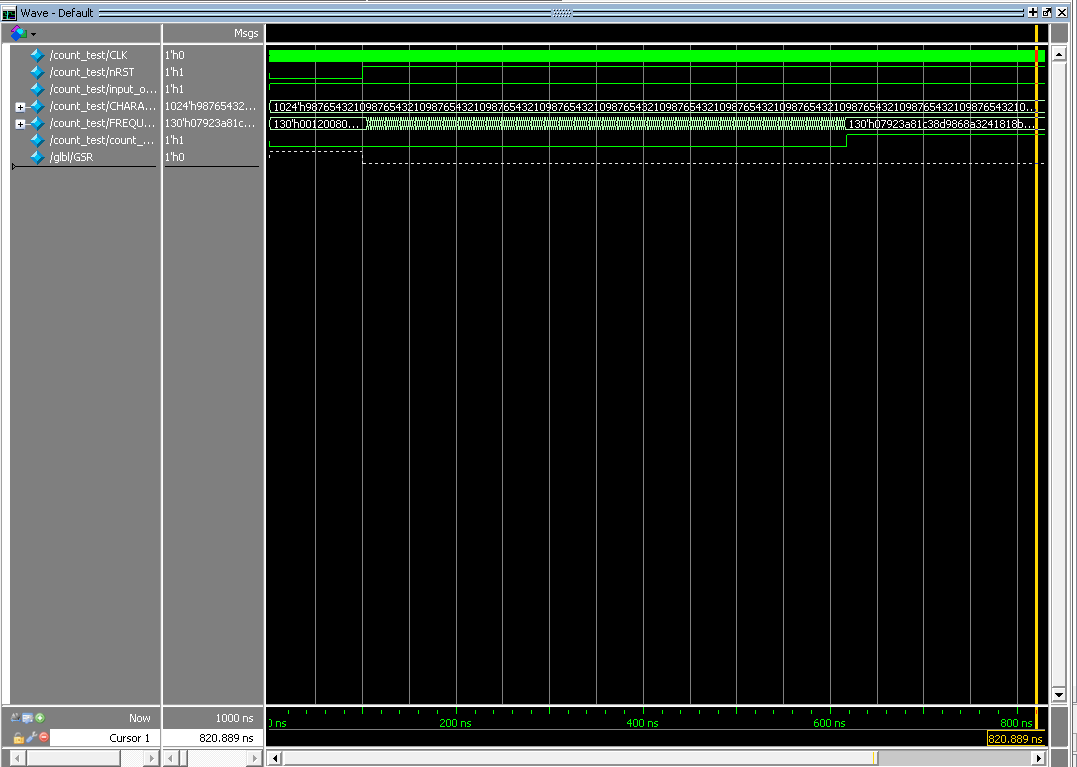
接收数据的输入，调用各模块，最后完成数据的按序一位输出。



# 3功能仿真

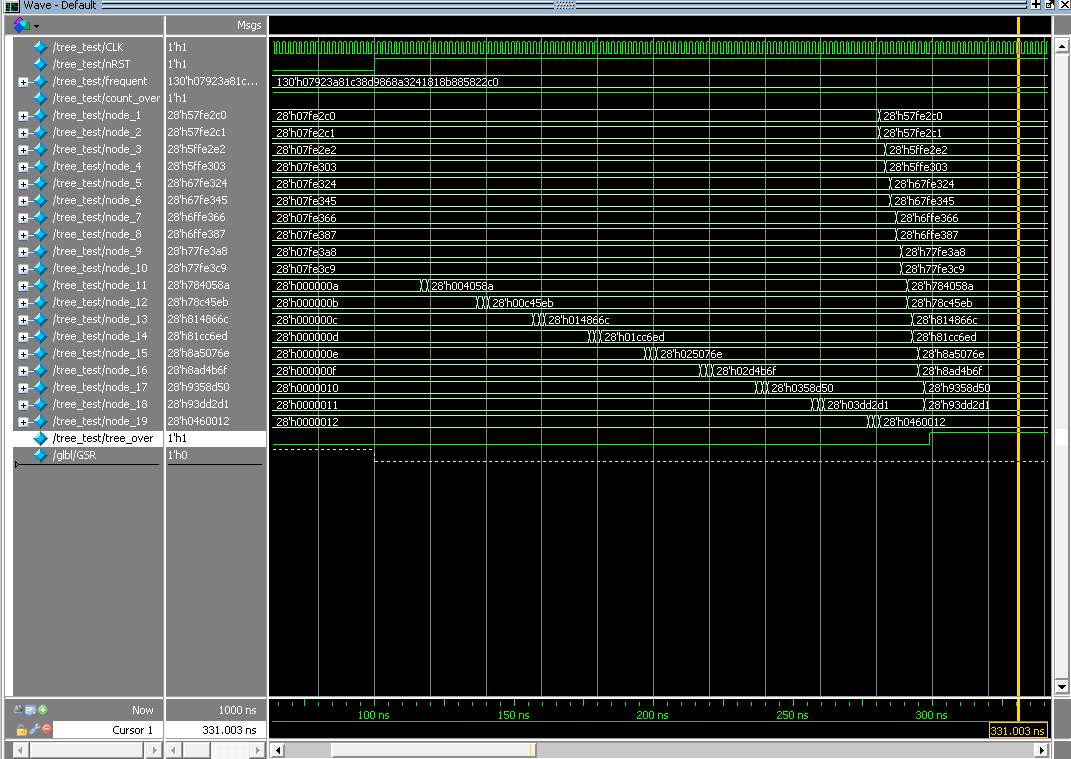
## 3.1 count模块

在复位和input\_over信号后，约经过520ns完成256个数据1024位的词频计数。置count\_over。



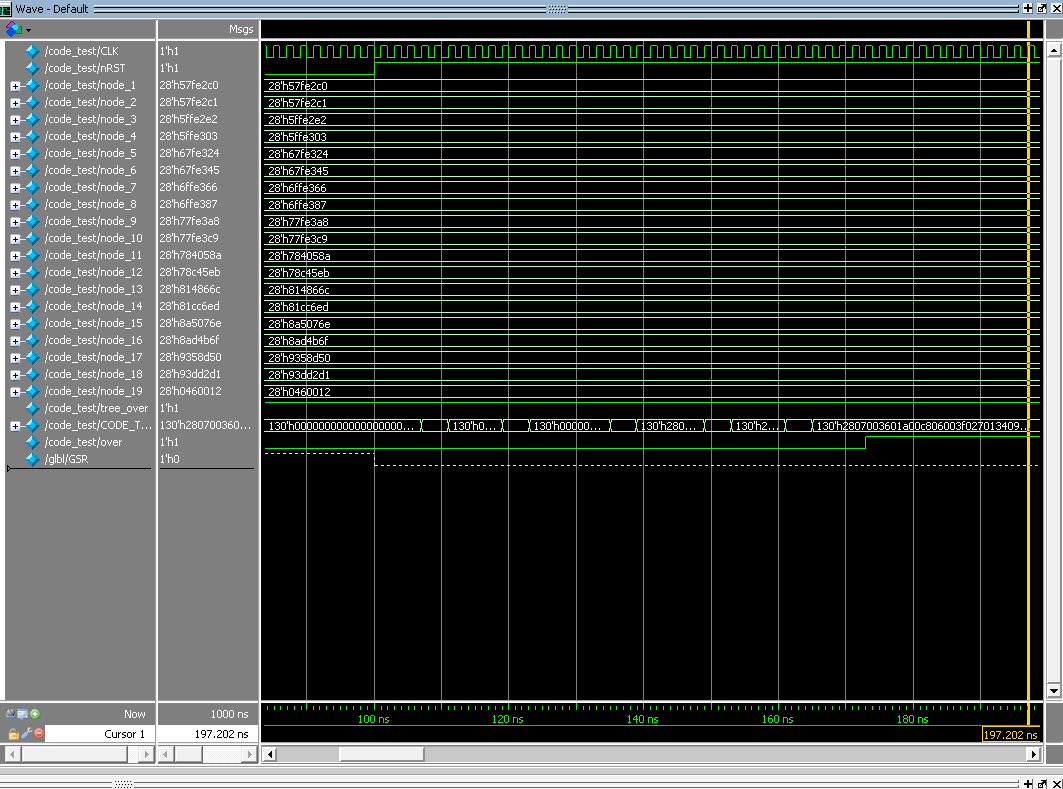
## 3.2 generate\_tree模块

在复位信号后进行9次排序与相加，大约200ns后完成Huffman树的生成，置tree\_over。



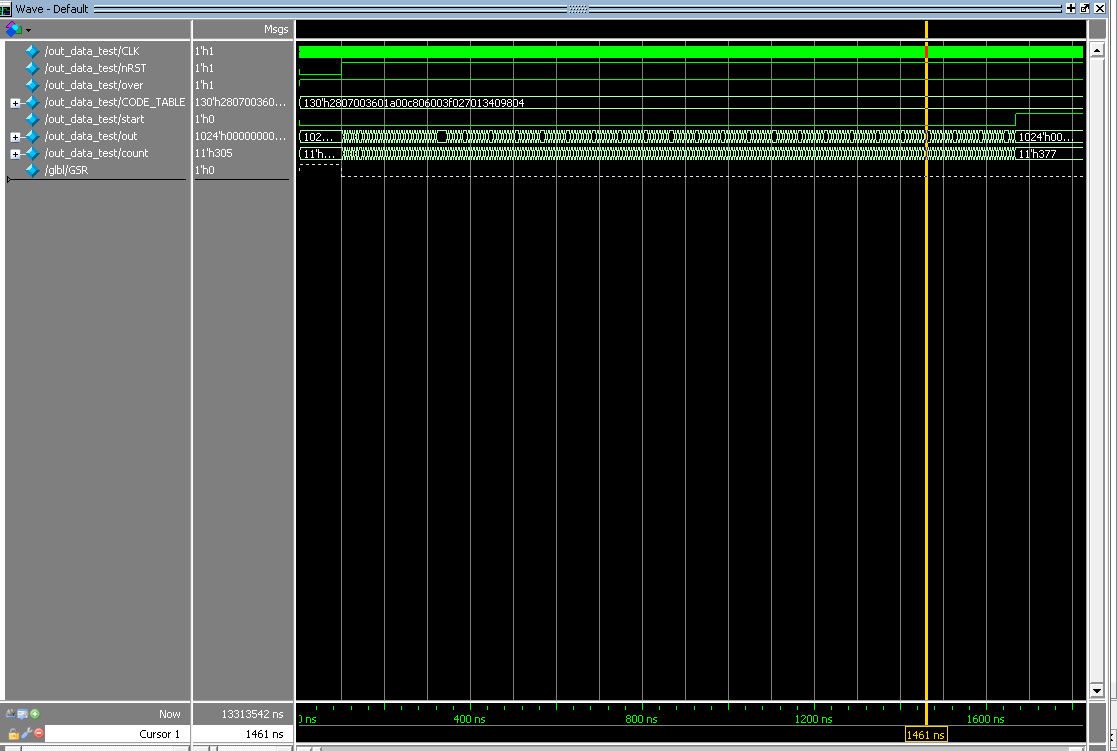
## 3.3 generate\_code模块

在时钟复位信号和tree\_over信号后大约75ns完成10个数据的编码工作，并置over信号。



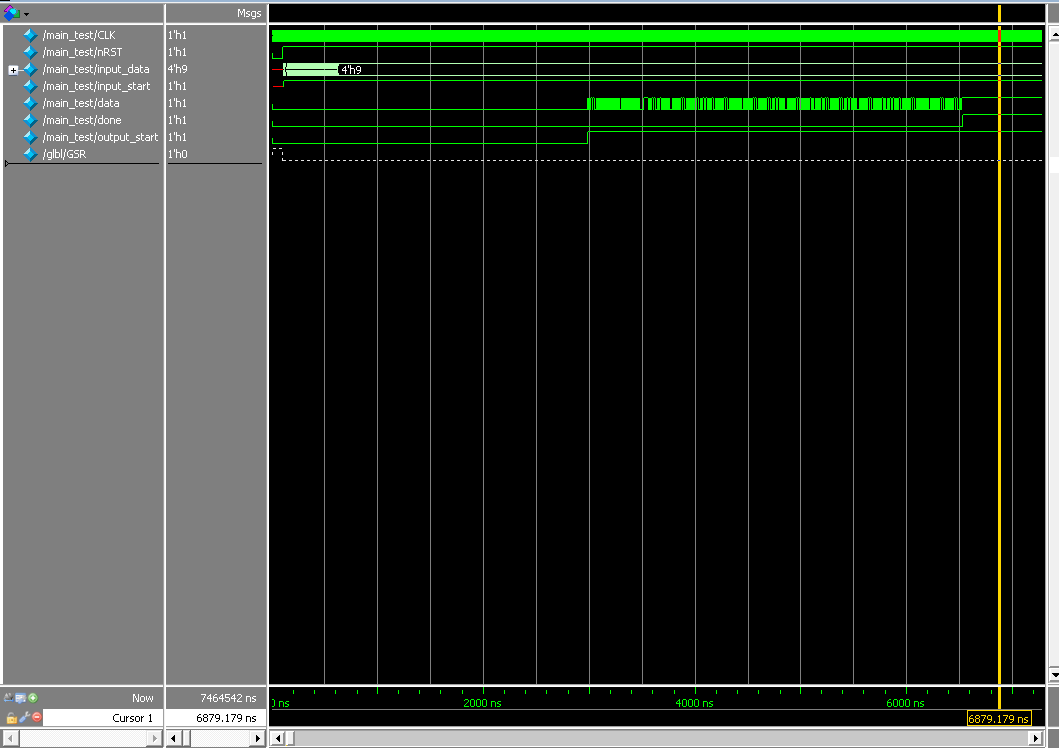
## 3.4 out\_data模块

当复位信号和编码结束信号置一时，开始进行256个输入数据的编码，经过约1600ns完成。



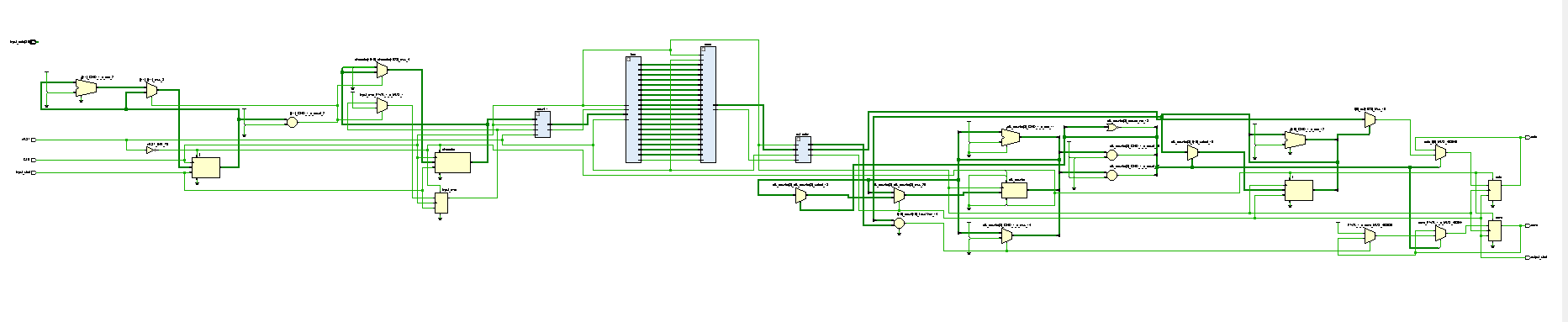
## 3.4 Main模块

如图所示，复位信号后，先是input\_start信号置一，开始输入数据。输入完之后进行Huffman编码。编码完成后，置output\_start信号为一，开始输出数据。先输10个数的编码，再输出256个数据的编码。输出结束后置done信号为一。结束本次编码周期。

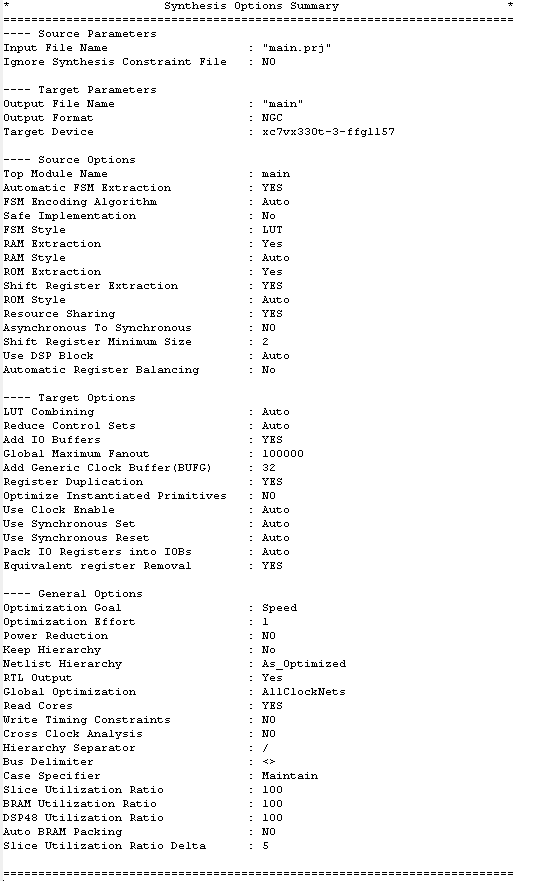


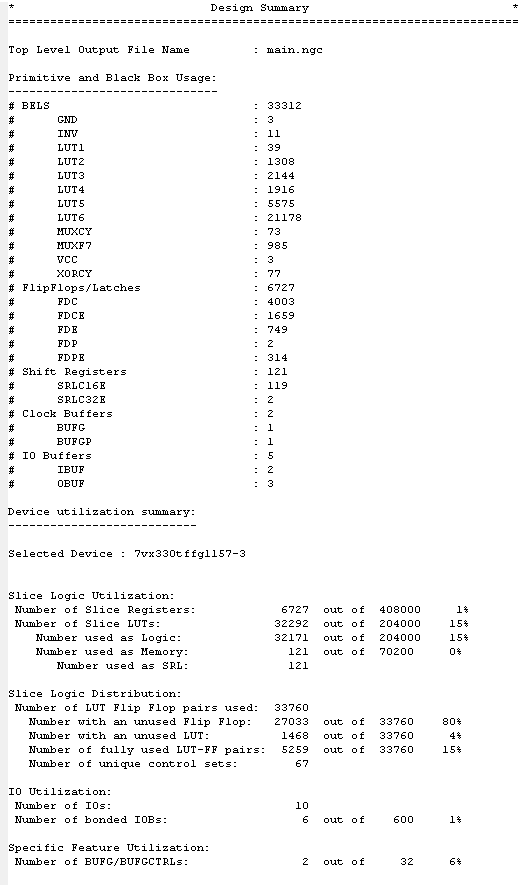
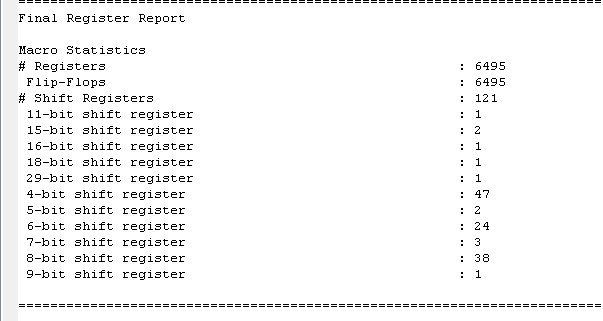
# 4逻辑综合报告

## 4.1生成电路

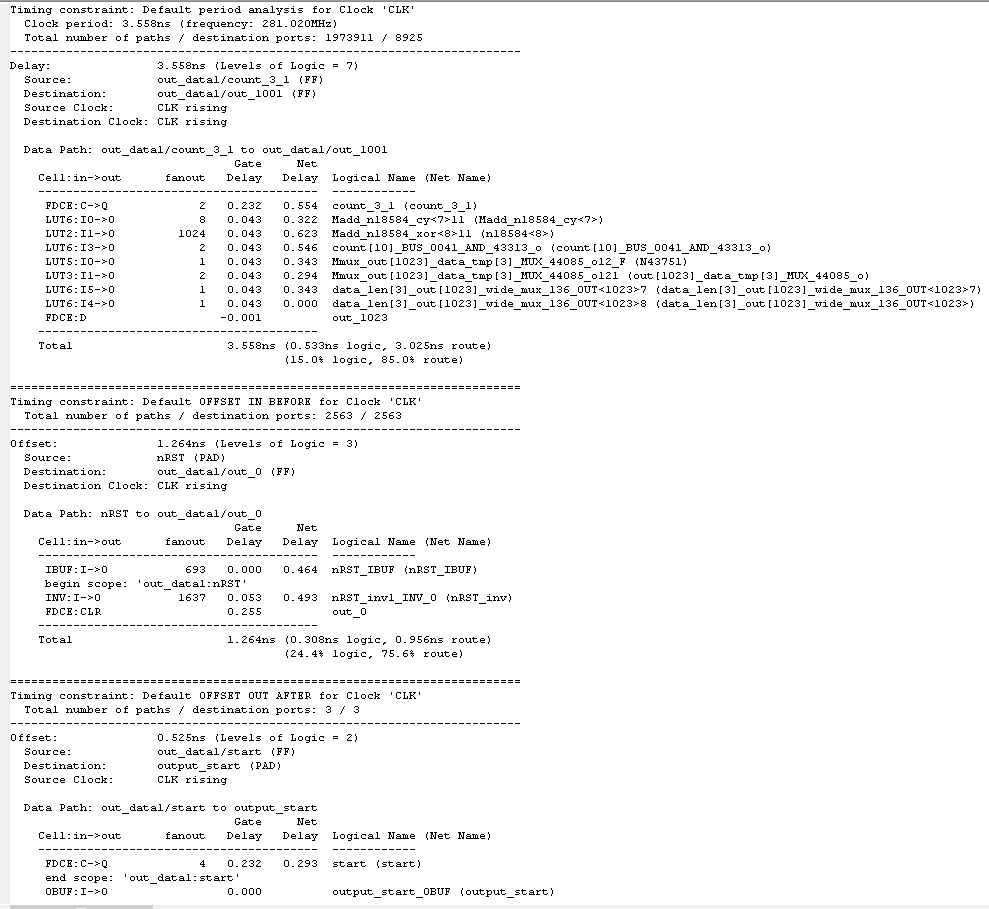


## 4.2综合选项

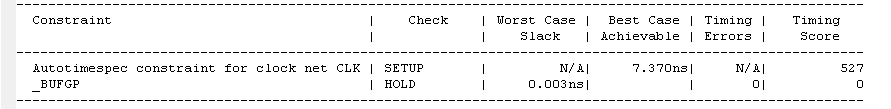




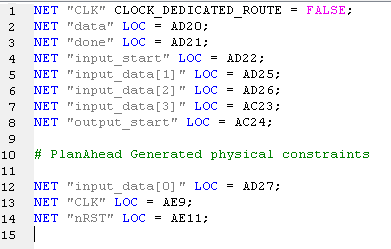
## 4.3时钟约束



## 4.4 worst slack

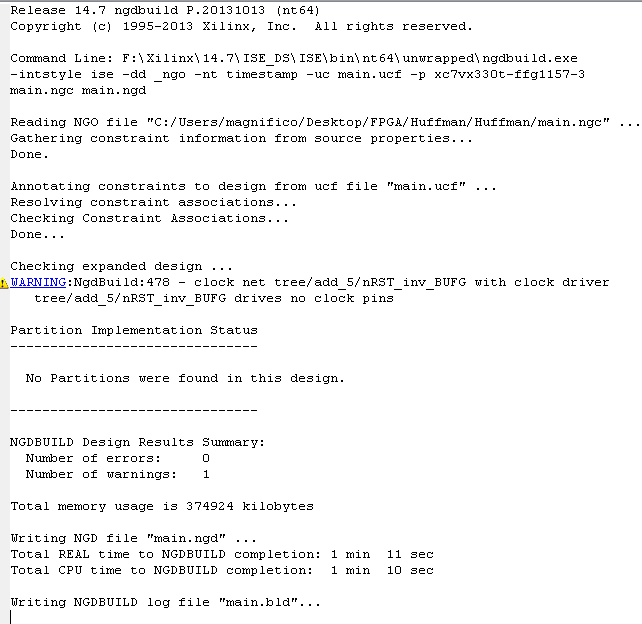


# 5 UCF约束

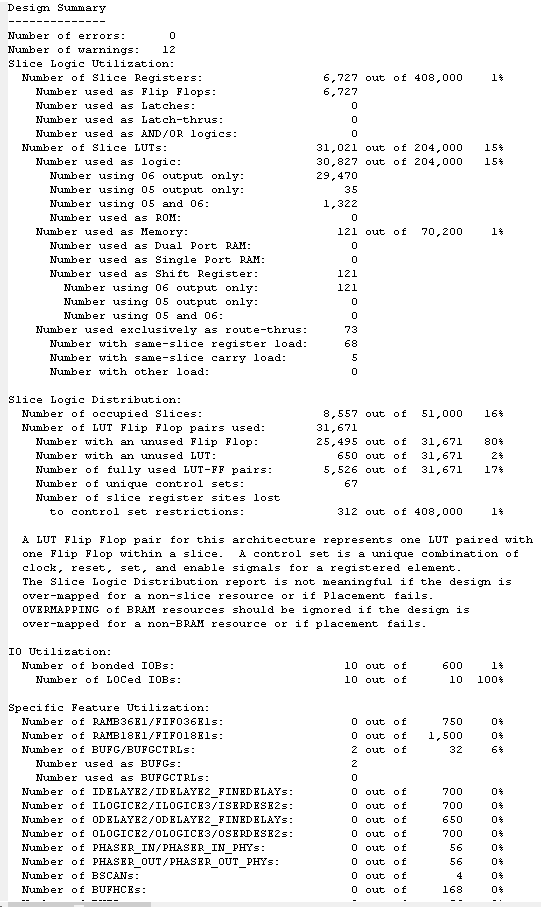


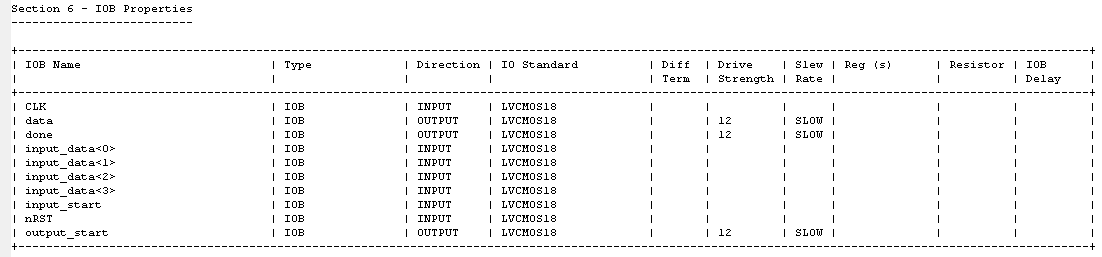
# 6实现报告

## 6.1 translate report



## 6.2 map report





## 6.3 P&R

