**霍夫曼编码器硬件实现**

201311期毕业设计论文

姓名： 李佳泽

学号：

时间：2014年3月11日

北京至芯科技FPGA培训

**【综述】**

哈夫曼编码(Huffman Coding)是一种编码方式，哈夫曼编码是可变字长编码(VLC)的一种。uffman于1952年提出一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长 度最短的码字，有时称之为最佳编码，一般就叫作Huffman编码。　哈夫曼压缩是个无损的压缩算法，一般用来压缩文本和程序文件。它属于可变代码长度算法一族。意思是个体符号（例如，文本文件中的字符）用一个特定长度的位序列替代。因此，在文件中出现频率高的符号，使用短的位序列，而那些很少出现的符号，则用较长的位序列。以此来提高代码的压缩效率或用于实现视频编码的传输效率。

**【课题内容】**

本课题是一个Huffman编码器的硬件实现架构，用于对输入的视频码流进行Huffman编码压缩，以利于后续传输的效率的提高，本架构可对8种字符以内的任意长度码流进行Huffman编码，输入待编码字符位宽为8位。

**【技术背景】**

1951年，[霍夫曼](http://zh.wikipedia.org/wiki/%E9%9C%8D%E5%A4%AB%E6%9B%BC" \o "霍夫曼)和他在[MIT](http://zh.wikipedia.org/wiki/MIT)[信息论](http://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF%E8%AE%BA)的同学需要选择是完成学期报告还是期末[考试](http://zh.wikipedia.org/wiki/%E8%80%83%E8%A9%A6)。导师[Robert M. Fano](http://zh.wikipedia.org/w/index.php?title=Robert_M._Fano&action=edit&redlink=1)给他们的学期报告的题目是，查找最有效的二进制编码。由于无法证明哪个已有编码是最有效的，霍夫曼放弃对已有编码的研究，转向新的探索，最终发现了基于有序频率[二叉树](http://zh.wikipedia.org/wiki/%E4%BA%8C%E5%8F%89%E6%A0%91" \o "二叉树)编码的想法，并很快证明了这个方法是最有效的。由于这个算法，学生终于青出于蓝，超过了他那曾经和[信息论](http://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF%E8%AE%BA)创立者[克劳德·香农](http://zh.wikipedia.org/wiki/%E5%85%8B%E5%8A%B3%E5%BE%B7%C2%B7%E9%A6%99%E5%86%9C)共同研究过类似编码的导师。霍夫曼使用自底向上的方法构建二叉树，避免了次优算法[香农-范诺编码](http://zh.wikipedia.org/wiki/%E9%A6%99%E5%86%9C-%E8%8C%83%E8%AF%BA%E7%BC%96%E7%A0%81)的最大弊端──自顶向下构建树。

**【技术目的】**

在[计算机](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA" \o "计算机)[数据处理](http://zh.wikipedia.org/w/index.php?title=%E8%B3%87%E6%96%99%E8%99%95%E7%90%86&action=edit&redlink=1)中，霍夫曼编码使用[变长编码表](http://zh.wikipedia.org/w/index.php?title=%E8%AE%8A%E9%95%B7%E7%B7%A8%E7%A2%BC%E8%A1%A8&action=edit&redlink=1)对源符号（如文件中的一个字母）进行编码，其中[变长编码表](http://zh.wikipedia.org/w/index.php?title=%E8%AE%8A%E9%95%B7%E7%B7%A8%E7%A2%BC%E8%A1%A8&action=edit&redlink=1" \o "变长编码表（页面不存在）)是通过一种评估来源符号出现机率的方法得到的，出现机率高的字母使用较短的编码，反之出现机率低的则使用较长的编码，这便使编码之后的字符串的平均长度、[期望值](http://zh.wikipedia.org/wiki/%E6%9C%9F%E6%9C%9B%E5%80%BC)降低，从而达到[无损压缩](http://zh.wikipedia.org/wiki/%E6%97%A0%E6%8D%9F%E5%8E%8B%E7%BC%A9)数据的目的。

例如，在英文中，e的出现[机率](http://zh.wikipedia.org/wiki/%E6%A9%9F%E7%8E%87" \o "机率)最高，而z的出现概率则最低。当利用霍夫曼编码对一篇英文进行压缩时，e极有可能用一个[比特](http://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)来表示，而z则可能花去25个[比特](http://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)（不是26）。用普通的表示方法时，每个英文字母均占用一个字节（[byte](http://zh.wikipedia.org/wiki/Byte" \o "Byte)），即8个[比特](http://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)。二者相比，e使用了一般编码的1/8的长度，z则使用了3倍多。倘若我们能实现对于英文中各个字母出现概率的较准确的估算，就可以大幅度提高无损压缩的比例。

霍夫曼树又称最优二叉树，是一种带权路径长度最短的二叉树。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根结点为0层，叶结点到根结点的路径长度为叶结点的层数）。树的路径长度是从树根到每一结点的路径长度之和，记为WPL=（W1\*L1+W2\*L2+W3\*L3+...+Wn\*Ln），N个权值Wi（i=1,2,...n）构成一棵有N个叶结点的二叉树，相应的叶结点的路径长度为Li（i=1,2,...n）。可以证明霍夫曼树的WPL是最小的。

**【方案简介】**

假定我们要处理的数据码流为“F”“F”“O”“O”“O”“R”“R”“R”“G”“G”“G”“G”“E”“E”“E”“E”“E”“T”“T”“T”“T”“T”“T”“T”。我们用二叉树的递归推到得到对应的二叉树如图1-1所示，由节点出发沿着树枝统计“0”和“1”，即可得到对应字符的霍夫曼编码。如果要用硬件实现，比较好的方式是采用FSMD的方案，有状态机控制不同的模块协同工作，结构清晰明了，可使整个架构和代码易与维护和交接，是系统设计的常用方式。本架构即采用了FSMD的方案，具体实施步骤如下：

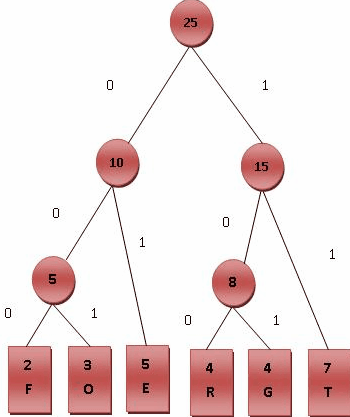


图1-1

1. 首先我们需要对输入的码流进行分组统计，以得到如下表格：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码字符 | T | R | R | O | E | F |
| 出现频次 | 7 | 4 | 4 | 3 | 5 | 2 |

表：1-1

对此，我们需要准备对应的分组统计模块，命名为group模块，该模块可对输入的码流进行分组统计，得到如上码表。模块如图1-1所示。

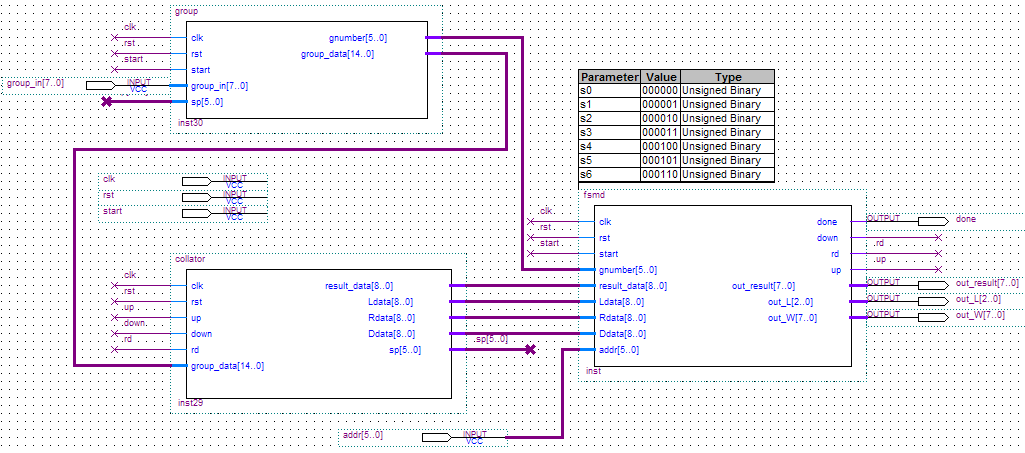


图1-1

1. 然后我们需要对“表：1-1”得到的字符频次进行排序，以得到由小到大的序列，对此，我们准备排序模块，命名为collator。模块如图1-1所示。

collator模块进行降序排序时由主状态机模块指挥，当接收到降序排列信号down有效时即进行降序排序，当排序完成时，主状态机将down置为无效，collator模块停止工作，保持现有状态，得到排序完成表格如表2-1所示。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码字符 | F | O | R | G | E | T |
| 出现频次 | 2 | 3 | 4 | 4 | 5 | 7 |

表：2-1

1. 当排序完成以后，需要进行二叉树的递归，最后得到相关字符的二叉树码表，对此，我们构建主状态机模块，由主状态机指挥collator模块和主状态机配合完成生成二叉树的工作，主状态机模块如图1-1所示。主状态机简明工作状态如表3-1所示：

|  |  |  |
| --- | --- | --- |
| 状态 | 状态描述 | 转移条件 |
| S0 | 开始分组 | Start开始 |
| S1 | 分组完成 | Start结束 |
| S2 | 开始排序 | 计数到待排序数量 |
| S3 | 排序完成 | 无条件转移 |
| S4 | 读取排序结果保存 | 计数到待排序数量 |
| S5 | 开启逆向排序 | 无条件转移 |
| S6 | 生成二叉树 | 计数到节点数量 |

表3-1

当主状态机完成二叉树后，done信号拉高，由外部测试模块读出生成的二叉

树表格。结果如表3-2所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码字符 | F | O | R | G | E | T |
| 编码长度 | 3 | 3 | 3 | 3 | 2 | 2 |
| 编码值 | 0000000 | 0000001 | 0000100 | 0000101 | 0000001 | 0000011 |

表3-2

1. 至此，本架构的工作已经完成，得到相应编码表，后续工作还需完善内容为将表3-2的编码进行提取，进一步得到下表的内容，方案之一是使用指针的方法进行取值，例如：由上表可知“F”的编码长度为3，即取编码值的后3位“000”即为字符“F”的霍夫曼编码。可在本架构的顶层模块进一步添加指针提取模块完成后续工作。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编码字符 | F | O | R | G | E | T |
| 编码值 | 000 | 001 | 100 | 101 | 001 | 011 |

表4-1

**【方案优化】**

本架构对于霍夫曼编码使用硬件架构实现，省去了对堆栈等环节的处理，相对于软件遍历等方式可大幅提高处理速度。对于8种字符综合后占用逻辑单元1400左右。应用时如对速度要求不是特别严格的场合可对架构进行优化，将二叉树模块改成流水线架构，实现逻辑单元复用，可进一步降低逻辑单元的占用。

**【应用展望】**

随着信息技术的飞速发展，各种各样的信息需要传输，传输信息就要得先经过编码，然后再译码，可见编码技术的提高对整个信息产业有着举足轻重的作用。霍夫曼编码是一种可变的无损压缩方法，其效率也比较高，所以在当今网络传输中意义重大。霍夫曼树是一棵最有二叉树，在各种程序设计中都用到它来降低程序运行的时间复杂度。在高清视频等应用日记普及的今天，许多软件的处理方式已经无法满足应用需求，于是硬件的优势便显现出来，FPGA具有的并行处理的天生优势，用硬件架构的方式思考，用硬件的优势去解决问题必会得到越来越广的应用。

**【附件】：**

文件夹名称：Huffman\_lijiaze

图片名称：原理图

模块文件： 1、group.v

2、collator.v

3、fsmd.v

4、top.v

5. top\_tb.v