Huffman Coding Method Based on Parallel Implementation of FPGA

Yi Chen, Guo Chun Wan, Ling Yi Tang, and Mei Song Tong
Department of Electronic Science and Technology, Tongji University, Shanghai, China

Abstract— With the rapid development of science and technology, the requirement of data compression speed is getting higher and higher. In recent years, software implementation of image compression technology in the general-purpose computer or DSP chip cannot meet the real-time processing speed requirements. Field Programmable Gate Array (FPGA) is a new type of digital circuit. Each logic gate in the FPGA chip performs some logical operation at the same time every clock cycle. Obviously, FPGA is essentially a large-scale parallel hardware device. In this paper, the parallel processing features of FPGA and parallel Huffman coding are utilized. Different from the traditional sorting algorithm, this paper uses the parallel characteristics of FPGA, using parallel full comparison algorithm. By using this algorithm, it will take only one clock and a small number of clock cycles to get all the data sorted. And the FPGA implementation exports the results through a three-stage pipeline.

1. INTRODUCTION

Huffman coding is a most widely used lossless compression technique [1]. Compression speed depends on the speed of the processor [2]. As a kind of variable-length and lossless compression coding, Huffman Coding has higher compression efficiency than other codes [3]. With the development of science technology, higher demands are required for Huffman Coding. The general procedure of Huffman coding is to construct a Huffman tree step by step by comparing the weights, and then encode or decode that tree. One of the difficulties is to find out the two smallest weights by the sorting algorithms. There are a variety of traditional sorting algorithms, most of which are based on the comparison between two weights. The parallel full sorting algorithm is based on the parallel comparison of any two weights in the sequence. In this paper, we present an implementation of Huffman Coding based on FPGA involving the problems of coding efficiency, resource utilization and clock frequency. Taking advantage of the parallelism and pipelining techniques of FPGA, the Huffman Coding can be obtained during sorting. This sorting algorithm not only speeds up the coding process, but also makes the coding time fixed.

The proposed design is designed by using Verilog HDL. There is no any IP core in this design. In addition, the output coded data is exported in a continuous stream of bits, which contributes to a maximum compression. According to the results of the simulation, synthesizing and implementation by Vivado (IDE of Xilinx FPGA), the proposed method has achieved a LUT (look-up table) usage rate of 1.31% (833 LUTs), FF (flip-flop) of 0.44% (553 FFs) and the highest clock frequency of the circuit reaches to about 270 MHz ($T = 3.7$ ns). The experimental comparisons also show that, compared to the conventional methods, the proposed design has great improvements in the highest clock frequency and FPGA resource utilization at the same time. Moreover, the required clock cycle is a constant.

2. HUFFMAN CODING PRINCIPLE

In 1952, David A. Huffman put forward this concept while studying for Ph.D. at MIT. It was published in the article A Method for the Construction Minimum-Redundancy Codes. This method which called the best coding, commonly known as Huffman Coding, is based on the probability of occurrence of characters to construct a different prefix codeword with the shortest average length. The first step is to build a Huffman tree by comparing weights, and then encode and decode it according to the tree. A general algorithm for implementing Huffman encoding is shown below:

(1) Using sorting algorithm to obtain the sequence of each character, which from high to low;
(2) Adding the minimum two weights;
(3) The values obtained in step ii are sorted with the remaining weights;
(4) Repeating step ii until the total probability is one;

Through the above steps, a Huffman tree is built. Each character node is a leaf node. The coding procedure starts from the root node. The left-node codes 0 and the right-node codes 1. Traversing the binary tree, the coding of each character can be obtained.

3. DESIGN SCHEME

Whichever classical Huffman Coding or Canonical Huffman Coding, it is necessary to build a static Huffman tree to obtain the code length and code. This design fully considers the parallel processing of FPGA. A parallel comparison algorithm is utilized to get the sort results of all data. The time of the process is fixed. Coding and calculating the code length are performed simultaneously. At the end of the sorting, both the code length and coding value can be obtained at the same time.

The design contains a three-stage state machine, which controls the switching between different modules and implements functions. The state machine consists of nine states. Respectively as: input statistics data state, parallel full comparison state, get semi accumulated value state, get full accumulated value state, get sorted results state, add two minimum weights state, wait for a cycle state, output code value state and output done state. The simplified algorithm flowchart is shown in Figure 1. Among that the left part is the parallel comparison sorting, the right part is parallelism coding process.

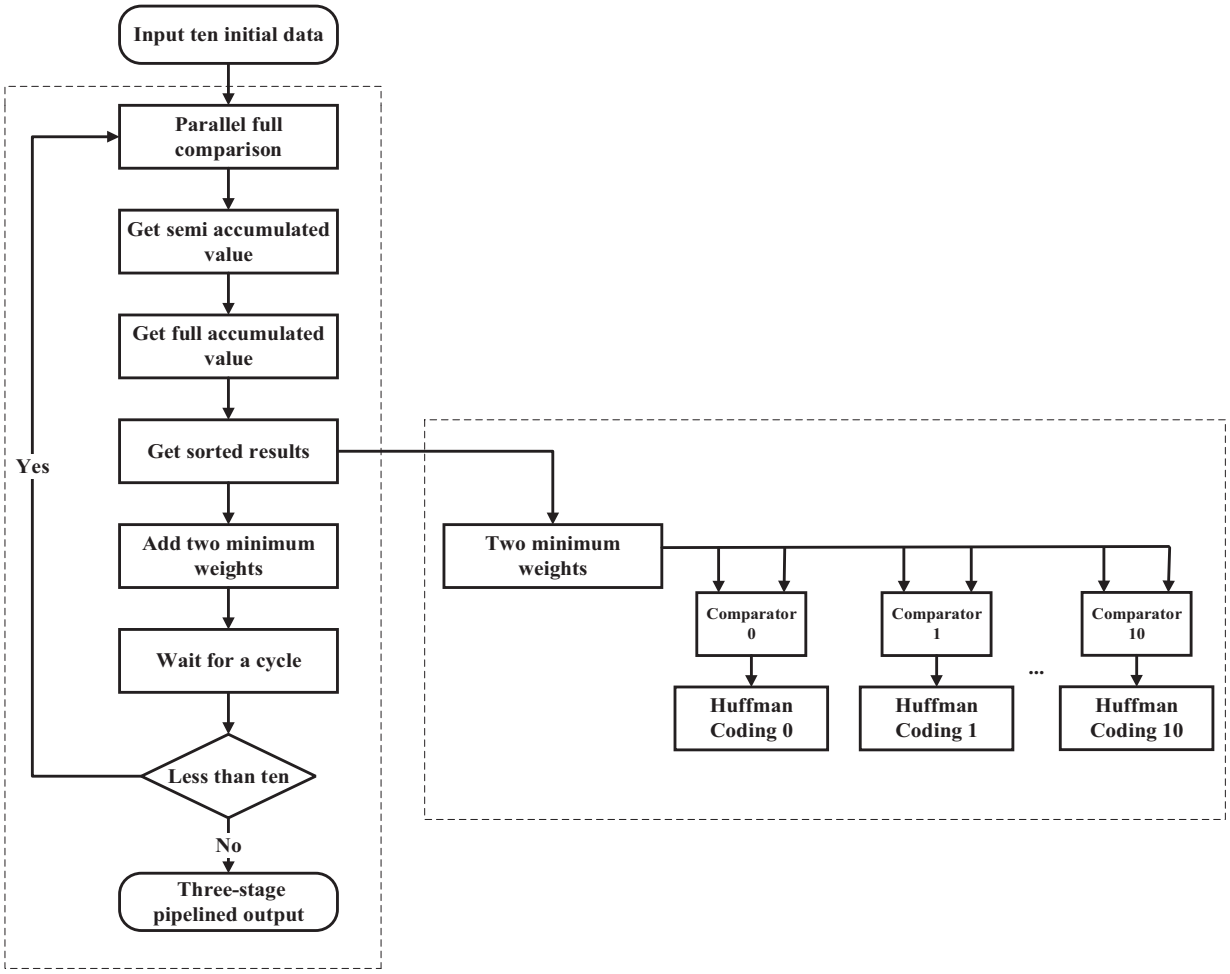

Figure 1: Programming flowchart.

3.1. Parallel Comparison Sorting

Most traditional sorting algorithms are based on the comparisons between two numbers. Our parallel complete sorting algorithm is based on the parallel comparison of any two numbers in the sequence [3]. Since all the data are processed at the same time, it will take a large amount of processing space. Therefore, this sorting algorithm is a trade space for time strategy [4]. The middle six states of the left part of the Figure 1 show the detail of the sorting algorithm.

A set of data that is first compared between the two data and gets a result of 0 or 1. Typically, 90 instantiation modules are needed. However, the remaining result can be obtained from the previous comparative results. Therefore, we can get the same result by only instantiating 45 modules. It only takes one cycle to get the comparative results for that parallel processing in FPGA. The

algorithm has the following advantages:

(1) Compared with other sorting algorithms, this algorithm can obtain two minimum weights at one time;

(2) By parallel processing, it only takes four clock cycles to achieve the sorting of the data. The time complexity is fixed;

(3) A module needs one clock cycle and the whole sorting module only takes 54 clock cycles;

(4) Data processing is carried out in pipelining without any additional control logic;

3.2. Accumulated Values

For the sake of getting higher a frequency, only half of the comparative results is processed in a cycle. In the next cycle, each half of results is added, which get the total accumulated values of the ten data. The accumulated values are shown in Table 1.

Table 1: Accumulated values.

X / Y	0	1	2	3	4	5	6	7	8	9	SUM
0		0	0	0	0	0	0	0	0	0	0
1	1		1	1	1	1	1	1	1	1	9
2	1	0		1	1	1	1	1	1	1	8
3	1	0	0		0	1	0	1	1	1	5
4	1	0	0	1		1	1	1	1	1	7
5	1	0	0	0	0		0	1	1	0	3
6	1	0	0	1	0	1		1	1	1	6
7	1	0	0	0	0	0	0		0	0	1
8	1	0	0	0	0	0	0	1		0	2
9	1	0	0	0	0	1	0	1	1		4

According to the result of the total value of the previous cycle, we get the value of data. The new weight of data is obtained by the addition of the two original minimum weights followed by the comparison of all weights on the next cycle. Repeat is needed for nine times with this step before getting all the Huffman Coding. Aimed to improve the timing, nothing is to be done but waiting at the last cycle.

3.3. Coding Process

As shown in the following Figure 2, the new coding result is the same as the traditional Huffman Coding. However, all data can be encoded at the same time by using of FPGA parallelism. In this way, it doesn't need extra clock cycle. The parallelism coding process is shown in the right part of Figure 1. The following tables from Table 2 to Table 4 show parallelism coding process. There are only two minimum data 0 and 7 coding at Table 2. However, all ten data are coded at the same cycle and some of them keep the old values. Table 4 shows all of ten data coded at the same time as they share a same root node.

Table 2: The first coding result.

Subscript	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Code Length	1	0	0	0	0	0	0	1	0	0
Coding Result	0	0	0	0	0	0	0	1	0	0

Table 3: The second coding result.

Subscript	0	1	2	3	4	5	6	7	8	9
Code Length	1	0	0	0	0	1	0	1	1	0
Coding Result	0	0	0	0	0	1	0	1	0	0

Table 4: The last coding result.

Subscript	0	1	2	3	4	5	6	7	8	9
Code Length	4	3	3	3	3	4	3	4	4	3
Coding Result	0011	101	001	100	110	1111	010	1011	0111	000

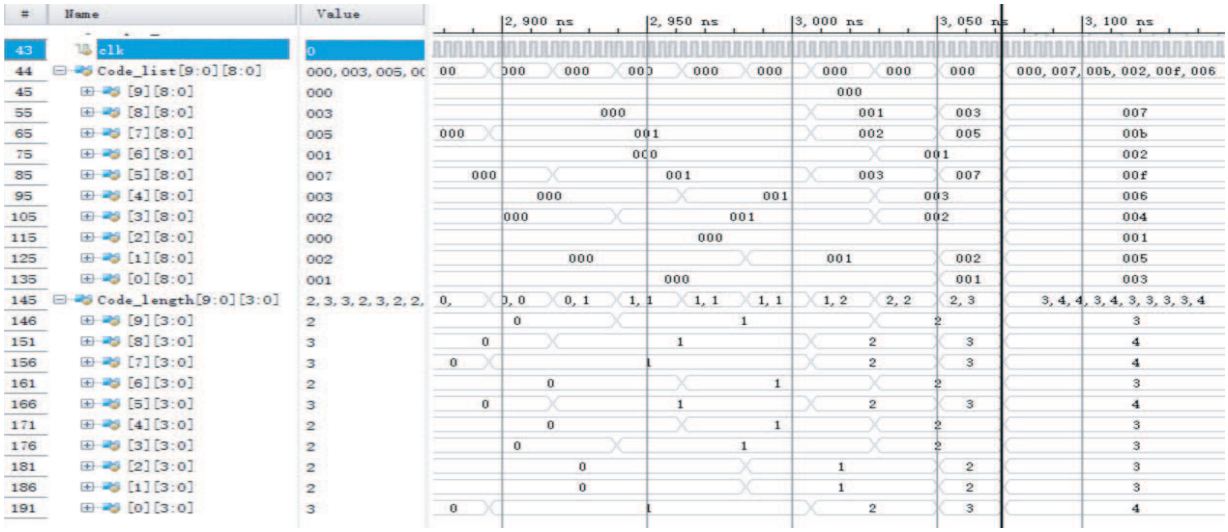

Figure 2: Waveform of the coding process.

4. SIMULATION RESULT

Considering there are some limitations that arise in Huffman Coding [5]. This experiment is based on the design suite Vivado, developed by Xilinx, a world's leading provider of FPGA. XC7A100T-1CSG324C is chosen as the target device. Both in the constraint file and testbench, the clock frequency is 270 MHz, which means that the period is about 3.7 ns. The simulation results shown in Figure 2 prove that the designed circuit can meet the requirements.

Table 5: Resource utilization.

Resource	Utilization	Available	Utilization%
LUT	833	63400	1.31
LUTRAM	24	19000	0.13
FF	553	136800	0.44
IO	10	210	4.76
BUFG	1	32	3.13

5. CONCLUSION

This paper implements the design of Huffman Coding based on FPGA. By taking full advantage of FPGA parallelism and pipelining technology, we can get the coding value while sorting. In this way, the total algorithm is fixed to 59 cycles regardless of different input weights. According to the experimental results, which are showed in Table 5, only about 1.31% of the LUT and 0.44% of the FF are used, and the maximum clock frequency is raised to about 270 MHz.

REFERENCES

1. Sharma, M., "Compression using Huffman coding," *International Journal of Computer Science and Network Security*, Vol. 10, No. 5, 133–141, 2010.
2. Pujar, J. H. and L. M. Kadlaskar, "A New lossless method of image compression and decompression using Huffman coding technique," *Journal of Theoretical and Applied Information Technology*, Vol. 15, No. 1, 18–23, 2010.
3. Ye, Z. H. and K. Q. Shen, *Information Theory and Coding*, Publishing House of Electronics Industry, 2013.
4. Shi, T. W. and J. C. Jin, "Parallel sorting algorithm based on FPGA," *Digital Technology and Applications*, Vol. 18, No. 2, 2013.
5. Kavousianos, X., E. Kalligeros, and D. Nikolos, "Optimal selective huffman coding for test-data compression," *IEEE Transactions on Computers*, Vol. 56, No. 8, 1146–1152, Aug. 2007.