International Journal of Advance Research in Science and Engineering
Volume No.06, Special Issue No.(01), Nov 2017
www.ijarse.com
IJARSE
ISSN: 2319-8354

Data Encryption on FPGA using Huffman Coding

Sourav Singh[1], Kirti Gupta[2]

[12]Electronics and Communication Department,

Bharati Vidyapeeth's College of Engineering, New Delhi, (India)

ABSTRACT

The ultimate goal to design an encryption algorithm is to ensure security against unauthorized attacks. To accomplish this, code compression is a vital step performed in high-speed digital data transport. Huffman coding is one of the efficient coding scheme used to perform compression. A critical step in performing compression is building the dictionary. The better the dictionary used by compression, the higher the compression rate. Huffman coding codes values statistically according to their probability of occurrence. Short code words are assigned to highly probable values and long code words to less probable values. The reference design files show the efficient implementation of the algorithms on encryption of data.

Keywords: *Data security, Digital, FPGA, Huffman coding, Spartan, Verilog*

I. INTRODUCTION

Data Security is one of the area that has gained importance in digital communication. Encryption of data is usually done to achieve the purpose. By Data Encryption it means to perform mathematical calculations and application of algorithmic schemes so as to transform plain text into cipher text, a form that is non-readable to unauthorized parties. A recipient of an encrypted message at the receiver side uses a key which triggers the algorithm mechanism for the decryption of the data. The decryption refers the transformation of the received data into the original plaintext version. There are many different types of data encryption, but not all are reliable. Huffman coding is one of the best available techniques; presently available for data encryption. Huffman coding is used for lossless data compression, in image compression, in communication process for transmission of highly encrypted data. Huffman coding reduces the memory problem. [1-2].

A programmable integrated circuit named as Field-programmable gate array (FPGA) is designed to be configured by a customer or a designer after manufacturing—hence has been referred to as FPGA meaning "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL) such as the VHDL, the Verilog etc. Instead of being restricted to any predetermined hardware function, an FPGA allows to program product features and functions, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed. A FPGA contains many (64 to over 10,000) identical configurable logic blocks (CLB) that can be viewed as standard components. CLBs can be programmed for to exhibit different functionalities [3]. The individual CLBs are interconnected by a matrix of wires and programmable switches.

Paper is organized in the following manner. Firstly, a brief introduction about Huffman coding is presented in section II. Thereafter, the design approach at the software level and the hardware level is explained in section

III and IV. The simulation results on Active HDL and FPGA are presented in section V. Lastly the conclusion are drawn in section VI.

II. HUFFMAN CODING ALGORITHM

To attain lossless data compression an entropy encoding algorithm known as Huffman coding is commonly algorithm used. The algorithm was proposed by David Huffman when he was enrolled in a Ph.D course at MIT, and published in the 1952 paper [1]. It is a method for the construction of minimum-redundancy codes and is known as the most efficient known compression style. It uses a variable-length code table for encoding a source symbol, where the table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol [2-5].

A. Algorithm

1. The probability of each character is computed.
2. The data set is sorted in ascending order.
3. A new node where the left child is the lowest in the sorted list and the right is the second lowest in the sorted list is created.
4. Chop-off those two elements in the sorted list as they are now part of one node and add the Probabilities. The result is the probability for the new node.
5. An insertion sort on the list with the new node is performed.
6. STEPS 3- 5 are repeated until only one node is left.

A pictorial representation of the algorithm is shown in Fig. 1 for the ready reference.

Formalized description

Input.

Alphabet A= $\{a_1,a_2,\ldots,a_n\}$ which is the symbol alphabet of size n.

Set, W = $\{w_1,w_2,\ldots,w_n\}$ which is the set of the (positive) symbol weights (usually proportional to probabilities), i.e.

W_i=weight (a_i) $1\leq i\leq n$

Output $C(A,W)=(c_1,c_2,\ldots,c_n)$ Code, which is the set of (binary) codeword's, where c_i is the codeword for. a_i , $1\leq i\leq n$

Goal.

Let $L(C)=\sum W_i \times \text{length}(c_i)$, be the weighted path length of code C. Condition: $L(C)\leq L(T)$, for any code T(A,W).

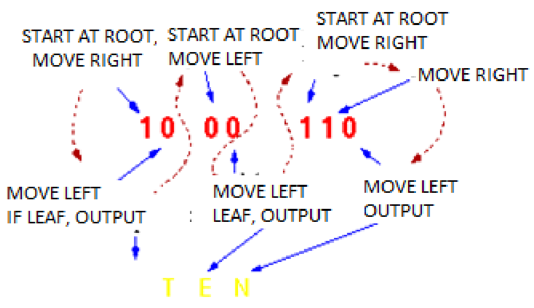

Fig. 1. An example of data encryption using Huffman Alogorithm

III. DESIGN APPROACH IN VERILOG HDL

In this manuscript, Verilog, standardized as IEEE 1364, is a hardware description language (HDL) has been used to model electronic systems. Verilog commonly employed in the design and verification of digital circuits at the register-transfer level of abstraction. A block diagram for the Huffman Coding module is shown in Fig. 2. The inputs to the block are a,b,c,d and clk whereas the 8 lines forms the encrypted data lines of the module. Based on the above algorithm, a sub-section of the program is given below which scans the array or string & count the frequency of each alphabet. It then assigns a appropriate Huffman code to the respective alphabet.

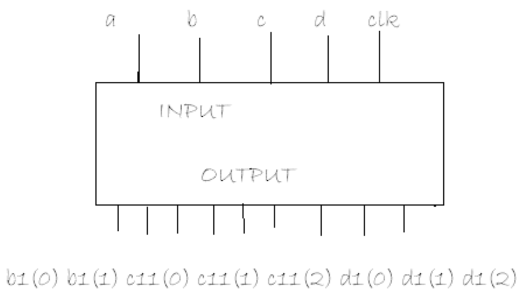

Fig.2. Block Diagram of the Huffman module

A subsection of the Huffman code in Verilog

```
Always @ (posedge CLK)
   Begin
      If ((a==1'b0) && (b==1'b0))
        Begin
          c1=c1+1;
        End
      Else if ((a==1'b0) && (b==1'b1))
        Begin
          c2=c2+1;
        End
      Else if ((a==1'b1) && (b==1'b0))
```

```
            Begin
                        c3=c3+1;
            End
   Else if ((a==1'b1) && (b==1'b1))
            Begin
                        c4=c4+1
```

IV. DESIGN APPROACH FOR FPGA

After calculating the frequency of each alphabet, the algorithm will check the alphabet with highest frequency and check; if its frequency is greater than the sum of rest of three alphabets. If it is found correct then it will assign bit '1' to it, else it will assign it bit '0'. Now, going to next frequency; it will first apply the compliment of highest frequency and then the same procedure is followed for it and rest of the alphabets. A synthesizable code written in Verilog to implement the functionality on FPGA is as follows:

```
Always@ (c1 or c2 or c3 or c4)
Begin
        If (c1>= (c2+c3+c4))
           a1=1'b1;
        Else
          a1=1'b0;
         If (c2>= (c3+c4))
               b1 [0] =1'b1;
          b1 [1] = (~a1);
    End
         Else
     Begin
           b1 [0] =1'b0;
           b1 [1] = (~a1);

   End
   If (c3>=c4)
    Begin
              c11 [0] =1'b1;
              c11 [1] = (~b1 [0]);
              c11 [2] = (~a1);
              d1 [0] = (~c11 [0]);
              d1 [1] = (~b1 [0]);
              d1 [2] = (~a1);
    End
```

```
Else
    Begin
            c11 [0] =1'b0;
            c11 [1] = (~b1 [0]);
            c11 [2] = (~a1);
            d1 [0] = (~c11 [0]);
            d1 [1] = (~b1 [0]);
            d1 [2] = (~a1);
End
    Endmodule
```

V. SIMULATION RESULTS

This section first present the results of Huffman coding on Active HDL. Thereafter, the results of the synthesizable code on FPGA: SPARATAN-II is demonstrated.

5.1 Result on Active-HDL

Active-HDL is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It shows the frequency of each character in string. Greater the number of counts to a respective character, which means maximum, is its frequency and vice-versa.

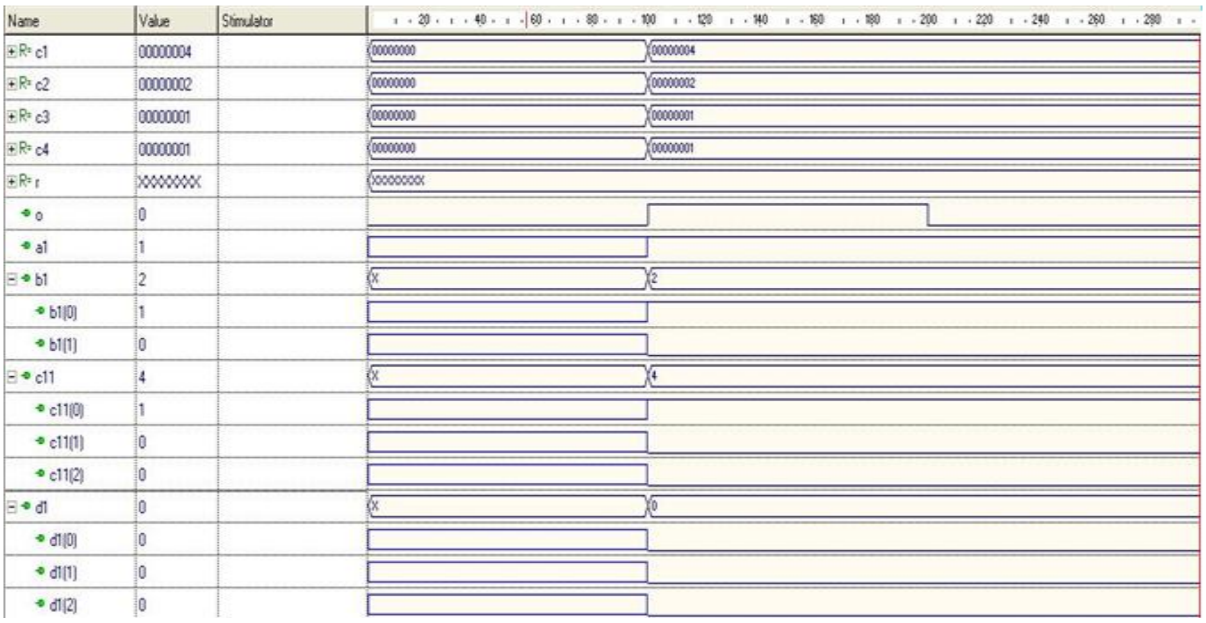

Fig.3. Output Waveforms on Active HDL

5.2Result on FPGA

The field-programmable gate array (FPGA) is a semiconductor device that can be programmed after manufacturing. With each clock pulse, the FPGA first count the frequency of each character and then gives first as the character with maximum frequency and in second clock pulse it gives the output with second highest frequency and so on.

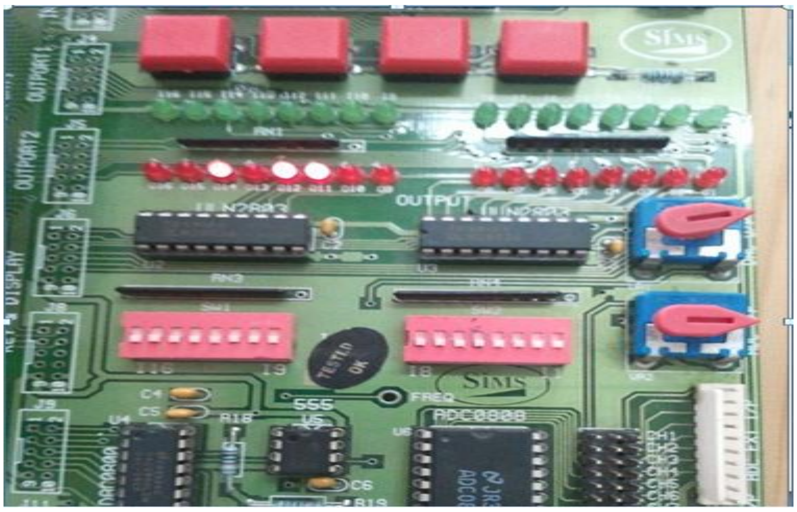

Fig. 4. FPGA output of Huffman Coding

VI. CONCLUSION

The principal goal guiding the design of any encryption algorithm must be security against unauthorized attacks. The critical part of compression is building the dictionary. The better the dictionary used by compression, the higher your compression rate. Huffman coding is used to code values statistically according to their probability of occurrence. Short code words are assigned to highly probable values and long code words to less probable values. We first calculated the frequency of each alphabet and then encrypted them using Huffman coding and make it synthesizable for FPGA. This Huffman coding application note describes the Huffman coding algorithm.

REFERENCES

[1] Daugman, J. G. Entropy reduction and decorrelation in visual coding by oriented neural receptive fields. IEEE Transactions on Biomedical Engineering, 36(1), 1989, 107-114.

[2] Vitter, J. S. Design and analysis of dynamic Huffman codes. Journal of the ACM (JACM), 34(4), 1987, 825-845.

[3] Monmasson, E., & Cirstea, M. N. FPGA design methodology for industrial control systems—A review. IEEE transactions on industrial electronics, 54(4), 2007, 1824-1842.

[4] Katona, G., & Nemetz, O. Huffman codes and self-information. IEEE Transactions on Information Theory, 22(3), 1976, 337-340.

[5] Bell, T. C., Cleary, J. G., & Witten, I. H. Text compression. Prentice-Hall, Inc. 1990.