World Journal of Science and Technology 2011, 1(8): 89-93
ISSN: 2231 – 2587
www.worldjournalofscience.com

FPGA IMPLEMENTATION OF SELECTIVE HUFFMAN CODING FOR HIGH SPEED DATA COMPRESSION AND DECOMPRESSION

Satish Kannale [1], Kavita Khare[2], Deepak Andore[3] and Mallikarjun Mugli [4]

[1,3,4] Department of Telecommunication Engineering, Rural Engineering College Bhalki, India
[2]Department of Electronics and Communication Engineering, MANIT, Bhopal, India
Corresponding author e-mail: ksatish_d@rediffmail.com

Abstract

This paper proposes a new test data compression/decompression method for system on chip. The method is based on analyzing the factors that influence test parameters: compression ratio, area overhead and test application time. To improve compression ratio, the new method is based on selective Huffman coding for reducing the amount of test data that must be stored on a tester and transferred to each core in a system- on- chip (SOC) during manufacturing test. The test data bandwidth between the tester and the SOC is bottleneck that can result in long test times when testing complex SOCs that contain many cores. In the proposed scheme, the test vectors for the SOC are stored in compressed form in the tester memory and transferred to the chip where they are decompressed and applied to the cores. A small amount of on –chip circuitry is used to decompress the test vectors. Given the set of test vectors for a core, a modified Huffman code is carefully selected so that it satisfies certain properties. These properties guarantee that the code words can be decoded by a simple pipelined decoder (placed at the serial input of the core's scan chain) that requires very small area. Results indicate that the proposed scheme can provide test data compression nearly equal to that of an optimum Huffman code with much less area over head for the decoder. An experimental comparison on bench mark circuits validates the proposed method.

Keywords: ATE, CUT, MISR, SOC, TAT, TDCE

Introduction

Due to the increased complexity of System on-a-Chip (SOC), testing is an important factor that drives time-to-market and hence the cost of design[3]. In order to accomplish this goal, chip designers often use pre designed and pre verified cores to develop systems-on-a-chip (SOC) devices. With time, these devices have become extremely complex. This high level of integration has allowed vendors to drive down the effective manufacturing costs. However, it has also rapidly increased the complexity of testing these chips. One of the increasingly difficult challenges in testing SOCs is dealing with the large amount of test data that must be transferred between the tester and the chip [5]. Each core in an SOC has a given set of test vectors that must be applied to it. The test vectors must be stored on the tester and then transferred to the inputs of the core during modular testing. As more and more cores (each with its own test set) are placed on a single chip, the amount of total test data for the chip increases rapidly [1].Test data volume is a major problem encountered in the testing of system-on-a-chip (SOC) designs [4]. This poses a serious problem because of the cost and limitations of automated test equipment (ATE). Testers have limited speed, channel capacity and memory. In general, the amount of time required to test a chip depends on how much test data needs to be transferred to the chip and how fast the data can be transferred (i.e., the test data bandwidth to the chip). This depends on the speed and channel

capacity of the tester and the organization and characteristics of the scan chains on the chip. Both test time and test storage is major concerns for SOCs from a test economics point of view.

In this paper, we present a new Stastical compression/decompression scheme to reduce the amount of test data that needs to be stored on the tester and transferred to the chip is been proposed here. The idea is to store the test vectors for a core in the tester memory in compressed form and then transfer the compressed vectors to the chip, where a small amount of on-chip circuitry is used to decompress the test vectors. Instead of having to transfer each entire test vector from the tester to the core, a smaller amount of compressed data is transferred. The approach presented here significantly reduces both test storage requirements and the overall test time.

Transferring compressed test vectors takes less time than transferring the full vectors at a given bandwidth. However, in order to guarantee a reduction in the overall test time, the decompression process should not add delay (which would subtract from the time saved in transferring the test data). Moreover, the on-chip decompression circuitry must be small so that it does not add significant area overhead.

Methodology Adopted

By implementing this technique, is achieved an enhancement to the existing coding method by integrating the both VLC and FLC feature to improve the efficiency of a data compression system. This technique implements a new hybrid method having the feature of both FLC and VLC and compare the performance of the proposed system with existing Huffman coding system. The performance analysis of the two systems is to be analyzed based on the timing consideration and resource utilization system.

Statistical Coding

The compression/decompression scheme described in this project is based on statistical coding. In statistical coding, variable length code words are used to represent fixed-length blocks of bits in a data set. For example, if a data set is divided into four-bit blocks, then there are 16 unique four-bit blocks. Each of the 16 possible four-bit blocks can be represented by a binary codeword. The size of each codeword is variable (it need not be four bits). The idea is to make the code words that occur most frequently have a smaller number of bits, and those that occur least frequently to have a larger number of bits. This minimizes the average length of a codeword. The goal is to obtain a coded representation of the original data set that has the smallest number of bits. A Huffman code is an optimal statistical code that is proven to provide the shortest average codeword length among all uniquely decodable variable length codes.

A Huffman code is obtained by constructing a Huffman tree. The path from the root to each leaf gives the codeword for the binary string corresponding to the leaf. An example of constructing a Huffman code can be seen in Table 1 for Fig 1. An example of a test set divided into four-bit blocks is shown in Fig.1. Table 1 shows the frequency of occurrence of each of the possible blocks (referred to as symbols). There are a total of 60 four-bit blocks in the example in Fig.1& Fig. 2 shows the Huffman tree for this frequency distribution and the corresponding code words are shown in Table 1.

An important property of Huffman codes is that they are prefix-free. No codeword is a prefix of another codeword. This greatly simplifies the decoding process. The decoder can instantaneously recognize the end of a codeword uniquely without any look ahead. Table 1 below shows the Statistical Coding Based on Symbol Frequencies for Test Set given in fig 1.

The amount of compression that can be achieved with statistical coding depends on how skewed the frequency of occurrence is for the different code words. If all of the code words occur with equal frequency, then no compression can be achieved. It is well known, however, that the test vectors in a test set tend to have a lot of correlations.

Test 1:	1111	0101	0011	1111	1011	1110	1101	1011
Test 2:	1111	1111	1111	1111	0000	0000	0000	0000
Test 3:	1001	0010	0110	0111	1110	1101	0110	1110
Test 4:	1111	1111	1110	0110	1111	1001	1111	1011
Test 5:	1111	1111	0111	1010	1111	1111	0111	1111
Test 6:	0010	1110	1000	0100	1111	1111	1111	1111
Test 7:	0110	1011	1011	1011	1101	0111	1011	0111
Test 8:	1100	1000	1010	0111	0101	1011	1111	1101
Test 9:	1011	0100	1101	1101	1110	1111	1111	1111
Test10:	1111	1011	0111	0101	1111	0111	1111	1101
Test11:	1100	1101	1001	1110	1110	1101	1011	0110
Test12:	0111	0010	1111	1111	0111	1111	1011	1111

Fig. 1 Huffman tree for the code shown in Table1

Table 1. Statistical Coding based on Symbol Frequencies

Symbol	Freq.	Pattern	Huffman code	Selective code
S0	30	1111	11	11
S1	12	1011	100	101
S2	10	0111	010	100
S3	09	1101	000	01101
S4	08	1110	1011	01110
S5	05	0110	0011	00110
S6	04	0000	10101	00000
S7	03	0010	01111	00010
S8	03	1001	01110	01001
S9	03	0101	01101	00101
S10	02	1100	01100	01100
S11	02	0100	00101	00100
S12	02	1000	00100	01000
S13	02	1010	101001	01010
S14	01	0011	101000	00011

Results Simulation Waveforms

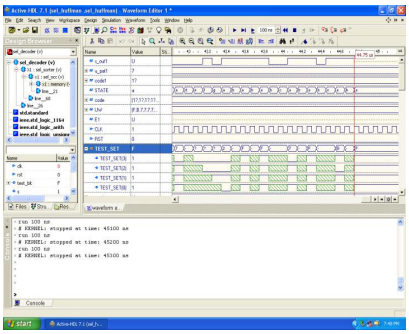

Fig 2.1. Simulation result showing Decoding operation for Huffman Decoder

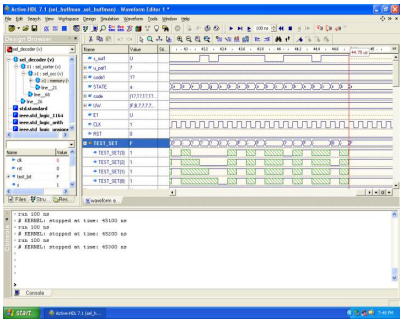

Fig 2.2.Simulation result showing decoding operation for Selective Huffman decode

Logic Placement

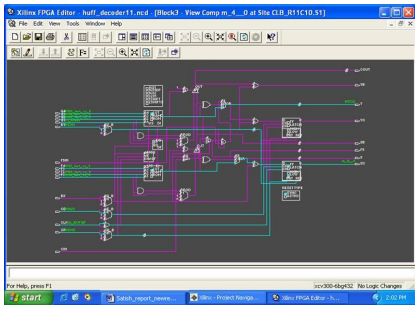

Fig 3.1. showing the Huffman design as shown above,targeting(Xcv300-6bg432) FPGA generated on FPGA Editor tool.

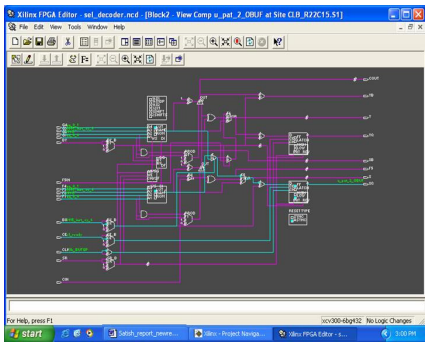

Fig 3.2. showing the selective design as shown above, targeting (Xcv300-bg432-6) FPGA generated on FPGA Editor tool.

FPGA Layout Editor

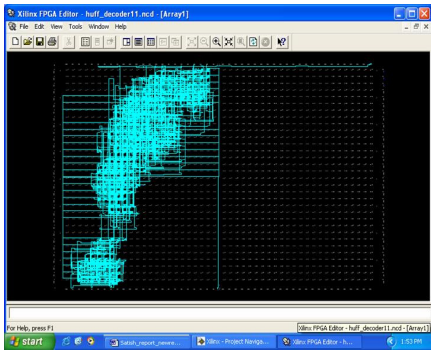

Fig 4.1. showing the implementation of Huffman design on to the targeted FPGA (Xcv300-6bg432) generated on FPGA Editor Tool.

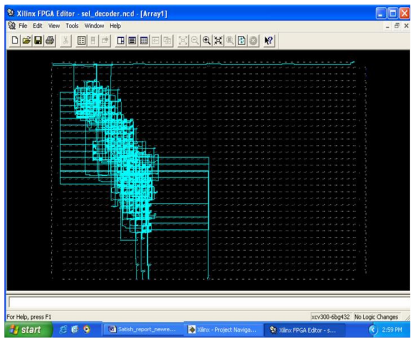

Fig 4.2. showing the implementation of selective huffman design on to the targeted FPGA (Xcv300-6bg432) generated on FPGA Editor Tool

Comparison between Huffman and Selective Huffman Coding

Sl No	Parameter	Huffman code	Selective Huffman code	Remark
1	Simulation time	46350ns	44750ns	1600ns faster
2	speed	slower	faster	
3	No of decoder states	14 states	7 states	
4	Decoder circuitry	Complex	Simpler	
5	Registers used	156	51	
6	Counters used	21	21	
7	Multiplexures used	20	15	
8	Comparators used	17	15	
9	BELS	1041	707	
10	Device utilization	More	less	
11	No of memory cells saved (input 384bits)	66 bits (318 bits)	38 bits (346 bits)	
12	Compression ratio	17.2%	9.9%	
13	If 1GB input data	-----	Save 20 sec time than Huffman coding	

Conclusion

The proposed design hass been implemented on VHDL and functionally verified on Active tool. From the obtained results it is concluded that; Statistical coding provides a powerful way to compress test data. It provides the advantage of reducing the time for transferring test data from the tester to the CUT with very high speed of decoding operation. Compared to the Huffman coding the memory used for storing the compressed test is higher in selective coding thus increasing the required memory to store the compressed test set, but at the same time the test bits are so encoded that it drastically reduces the time taken to decode the test set for decoding.

References

1. **Abhijit Jas, Jayabrata Ghosh-Dastidar, Mom Eng Ng, and Nur A. Touba, (2003).** "An Effective Test vector compression Scheme using Selective Huffman Coding" IEEE Transactions on CAD of Integrated circuits and systems Vol.22,no.6.

2. **P. Gonciari, B. M. Al-Hashimi, and N. Nicolici, (2002).** "Improving compression ratio, area overhead, and test application time for systems-on-a-chip test data compression/decompression," in Proc. Design Automation Test Eur. pp. 604–611.

3. **J E. J. Marinissen, Y Zorian, R. Kapur, T.Taylor, and L.Whetsel. (1997).**Towards a

Standard for Embeded Core test: An Examle.In Proceedings IEEE International Test Conference, pp. 616-627.

4. **Y.Zorian, E.J.Marinissen and S.Dey,(1998).** "Testing embedded-core based system chips'', proc.Int.Test Conf., pp.130-143.

5. **R. Chandramouli and S. Pateras,(1996).** "Testing systems on a chip," IEEE Spectrum, pp. 42–47.

6. **A. Chandra and K. Chakrabarty.(2001).** Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression. In Proceedings IEEE VLSI Test Symposium, 114–121.

7. **A. Chandra and K. Chakrabarty.(2001).** System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes. IEEE Transactions on Computer-Aided Design, 20:113–120.

8. **A. Jas and N. A. Touba, (1998).** "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in Proc. Int.Test Conf. pp. 458–464.

9. **A. Jas, J. Ghosh-Dastidar, and N. A. Touba, (1999).** "ScanVector compression/ decompression using statistical coding," in Proc. VLSI Test Symp. pp. 114–120.