MULTI LEVEL HUFFMAN TEST DATA COMPRESSION WITH MULTIPLE SCAN CHAIN METHOD

[1]Srikanth Immareddy, [2] Sravan Kumar Talusani

[1] *Electronics and Communication Engineering Dept, Methodist College of Engineering and Technology, Hyderabad 500001, India.*

[2] *Electronics and Communication Engineering Dept, Methodist College of Engineering and Technology, Hyderabad 50001, India.*

Abstract

The present paper focuses on the compression system using Huffman compression algorithm for data compression. The rate of data decompression is more time consuming during the decoding of these compressed data. Various works have proposed for the enhancement of compressed data decoding. In this paper the realization and implementation of a High speed decoding Huffman coding system is proposed and the proposed architecture shows an improvement with respect to the speed of decoding operation based on multiple scan method and multi level decoder partitioning concept. As a apart of the above work the existing Huffman coding system is also implemented and its performance in terms of speed and area are compared with the proposed work.

Keyword: Data compression, Huffman coding, Multi-level, Multiple-scan high speed Decoder

I. INTRODUCTION

Data compression is one of the major challenges involved in testing and has been difficult to master and also tough to implement. Coding is often referred as a special representation of data, without loss of required information. The Information theory is defined to be the study of efficient coding and it affects the transmission speed, memory required for storage and error probability. A new data compression algorithm is being presented in this work analyzing the existing Huffman coding system. Data compression involves in transforming a stream of bits into a new stream of bits, by reducing the length using the Huffman coding algorithm.

The problem of compressing the test data and reducing test time for core-based Silicon on chip has been discussed in several different angles in recent Literature [1], [6], [7], [8]. The time required for test data between a workstation across a network and a tester off the chip is reduced using scan chain architectures maximizing the bandwidth utilization are presented in [1]. Novel approaches for compressing test data using the Burrows–Wheeler transform and run-length coding were presented in [7], [8]. A new scan vector using cyclical decompressor and run-length coding is described for compression and decompression of scans vectors in [9]. A modular built-in self-test (BIST approach that allows sharing of BIST control logic among multiple cores is presented in [3]. A novel Technique for combining BIST and external testing across multiple Cores are described in [12]. The statistically encoding test data idea was presented in [2]. A statistical coding using comma codes similar to Huffman codes in described for nonscan BIST scheme circuits is presented in [5]. A technique that uses a linear combinational expander circuit is presented in [2]. The use of Frequency directed run length (FDR) and Golomb codes for test data compression has been demonstrated [4]–[6]. A Parallel Serial Full Scan (PSFS) approach for reduction of test time in cores is presented in [5].A virtual scan chains for specially designed cores is presented in [5] for reduction in test time and test data. Various types of run length coding have been discussed in the above surveys which are not the efficient coding method.

A Huffman code with variable length input is proposed in [11] for System on chip SOC test data compression. Increase in test data bandwidth has been presented in [9] using a technique for reusing scan chains for other cores in an SOC. An automatic test pattern generation (ATPG) techniques have been described in [10] for producing test cubes that are suitable for encoding. A fixed-to-fixed block-encoding scheme is described in [7]. A fault simulation-based technique to reduce the entropy of the test vector set by pattern transformation is described in [8]. Such transformations increase the amount of compression that can be achieved on the transformed test set using statistical coding. ATPG algorithms for producing test vectors that can more effectively be compressed using statistical codes have been described in [2]. Huffman coding algorithms are used for automatic test pattern generation for system on chip have been discussed.

A test-data compression method based on multi level Huffman coding, has been presented for IP cores of unknown structure, that improves compression result, area over ahead for decompressor is presented in[13]. A multilevel Huffman test-data

compression approach for IP cores with multiple scan chains has been presented with a decompression architecture that can generate clusters of test bits in parallel. This new method reduces test-application times and achieves high compression ratios, and also the sizes of the encoded data blocks are made independent of the slice size that can be applied to multiple scan-chain cores has been presented in [14]. An efficient decoder for decompressing the compressed data is implemented using Xilinx and simulated on Active HDL enhances the speed of decoding operation has been presented in [15]. The present paper implements and realizes the high speed Huffman decoder with the concept of multi level Huffman using the multi scan chain method.

II. DATA COMPRESSION

Data compression is a way of encoding digital data so it takes up less storage space and requires less network bandwidth to be transmitted. An effective encoding technique minimizes the average length of a data. Consider data containing m unique patterns A_1, $A_{2....}A_m$ with probabilities of occurrence $P_1,P_2,\ldots P_m$ respectively[3]. The entropy H is defined intuitively as the minimum average number of bits required to represents a pattern, is given by

$$H(DS) = -\sum P_i \log_2 P_i \qquad (1)$$

Therefore, an optimal encoding technique is one in which the average number of bits needed to represent a pattern is closest to the entropy bound. Among all statistical encoding techniques the Huffman code is having the shortest average codeword that assign a unique binary codeword to each pattern [12]. In fact, if L_H is the average length of a Huffman-encoded pattern in D_S, then

$$H(DS) \leq L_H \leq H(DS) + 1. \qquad (2)$$

In addition, Huffman codes possess the prefix-free property, i.e., no codeword is the prefix of a longer codeword [15]. It can be shown that this property is important for decoding. Table-III illustrates the Huffman code for an example data set with sixteen unique patterns out of a total of 384. Column 1 of Table-III lists the sixteen patterns, column 2 lists the corresponding number of occurrences (Occ) of each pattern Xi, and column 3 lists the corresponding probability of occurrence p_i, given by Occ /D_S. Finally, column 4 gives unique pattern to each corresponding Huffman code. Using a state diagram approach, as shown in Figure 1, performs Huffman decoding. In conventional Huffman decoding each incoming bit is compared with entries of the lookup table of the code vectors. The original or the encoder transmitted data from is retrieved from the lookup table, which consists of all the unique words and their corresponding code vectors.

Once the Huffman coding is performed, the encoded data is serially transmitted to the decoder. In a conventional decoder, these single bit data is accepted and the state of decoder is changed based on the data that can be either '0' or'1'. Once the data is received at the decoder, it is compared with each entries of the code vectors form the lookup table. If the received data matches with any of the code vectors in the lookup table, the unique word of the matched code vector corresponding to the transmitter encoder data is decoded by the decoder. If a mismatch occurs, the decoder accepts the next received data and repeats the above procedure. When all the states of the decoder are traversed and a unique word of the received data pattern is identified, the control is transferred to the initial state, indicating that the unique word for the data pattern has been matched and the decoder wait for the next incoming data. If unmatched, control is transferred to next state as shown in the figure and again process repeats. The decoding process continues till the encoder stops transmitting the data.

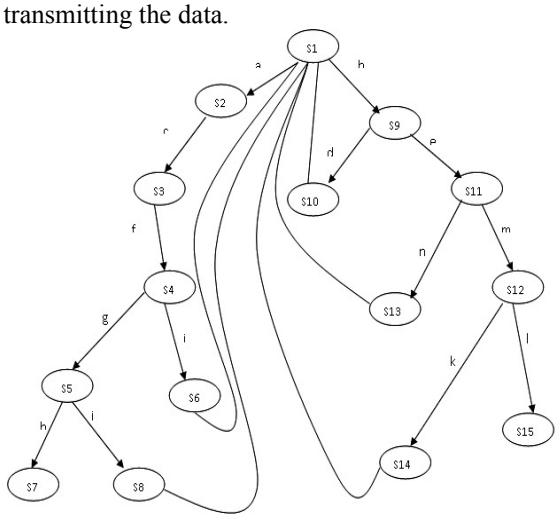

Fig. 1. State Transition of a conventional Huffman Decoder

Once the complete data has been received, decoding of the compressed data is done as said earlier by the state diagram approach and the decoded data is decompressed and the complete process of decoding is called as decompression of data. Figure 1 shows the state diagram for Huffman decoding. Each bit of data after reception is compared with the code vectors of the look up table if a match occurs between the test code and code word of LUT then the state is returned to its initial state returning the corresponding symbol for that code word. Else on not matching the state moves to next state. For example the transition of state starts with 'S1' state called initial state and

proceeds to next states based on the code bits received. On a match, the state 'S1' results a symbol 'a' or 'b' starting again with initial state. If a non-match occurs the state goes to the next state depending on the received code bit.

III. HIGH SPEED HUFFMAN DECODER ARCHITECTURE

The block diagram of high speed Huffman decoder architecture is shown in Figure 2. The proposed decoder receives the input bit stream in serial from the encoder. The received serial data is passes to the N-bit shifter which converts the serial data into parallel data. The length decoder and the symbol encoder accepts the parallel data from the Shifter, decodes the original data back from the decoding code words. The code words of equal length are shorted and aligned in the look up table (LUT) of a codebook. The symbol decoder consists of partitioned codebooks of equal length code words.

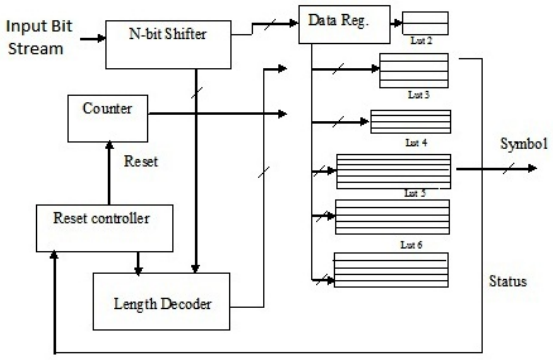

Fig. 2. Architecture of High speed Huffman Decoder

The length decoder calculates the length of the received code word and the high-speed decoder scans the LUT with the received code depending on its length. The scanning time of tracing is reduced as each LUT consists of specific length codes. For Example, if the code word for '1111' is '11' during encoding the '1111' will be represented as '11' and transmitted. In the decoding of a normal tree based Huffman decoder the two clock cycles are utilized in storing these two bits. As the actual data is of size four bits, additionally 16 clock cycles are utilized by the system for tracing out the code word from code book. Hence a tree based decoder utilizes 34 clock cycles (2+16+16=34) for decoding.

Whereas the proposed Multi-scan decoder utilizes 2 clock cycles for scanning LUT-1 which is of one bit length, if the matching fails, the next code bit is latched i.e. '11' and the scanning is performed on LUT-2 which is of length 2. It takes maximum of four clock cycles for scanning LUT-2.Hence it utilizes maximum of 8 (2+2+4) clock cycles, which is comparatively faster than the existing conventional Huffman decoding system.

The counter is used in controlling the decoding operation. Selection of the LUT depends on the length decoder in the symbol decoder unit. The selection of the code word is done sequential manner based on the count value from the counter unit. The length decoder output helps in selecting one among the six LUT's for tracing the code word from the LUT. The reset generation unit generates the reset signal for the counter and the length decoder unit; they get reset to initial value. The reset generator generates the reset signal when the status signal goes high. The status signal is generated from the symbol decoder when a match of code word occurs. Under every 6-clock cycles the shift register receives 6 coded bits and transfers it to data register of symbol decoder and length decoder.

While the symbol decoder process on data registers value the shift register keep receiving the next code bits from the encoder unit. This gives the decoder unit to work simultaneously while receiving the data bits. This gives a faster operation compared to existing tree based decoding where the encoded bits are latched for decoding. The output of the symbol decoder and the received code bits are applied to the length decoder which generates the count length starting from L1 to the maximum of the received code bit length. Depending on either L1 or L2 or L3 etc one of the LUT among the 6 LUT is selected.

The count generation is stopped by length decoder once it receives a reset signal; this completely disables the decoding system from further decoding. For example, if the Code words received at the end are not of 6-bit length, in such cases the decoder generates a count up to the existing code bits only. Table 1 generates length count from L1 to L6 for selecting the LUT.

TABLE I
TRUTH TABLE FOR LENGTH DECODER.

L1	L2	L3	L	L5	L6
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

IV. SIMULATION RESULTS

The proposed architecture is simulated on Active HDL 8.1 v, with the data bits as given in table II. The sample data used for simulation are the compacted test cubes.

TABLE II
DATA DIVIDED INTO 4-BIT BLOCKS

1111	0101	0011	1111	1011	1110	1101	1011
1111	1111	1111	1111	0000	0000	0000	0000
1001	0010	0110	0111	1110	1101	0110	1110
1111	1111	1110	0110	1111	1001	1111	1011
1111	1111	0111	1010	1111	1111	0111	1111
0010	1110	1000	0100	1111	1111	1111	1111
0110	1011	1011	1011	1101	0111	1011	0111
1100	1000	1010	0111	0101	1011	1111	1101
1011	0100	1101	1101	1110	1111	1111	1111
1111	1011	0111	0101	1111	0111	1111	1101
1100	1101	1001	1110	1110	1101	1011	0110
0111	0010	1111	1111	0111	1111	1011	1111

This section illustrates the statistical encoding for the given set of data. To encode the given data, it is divided into 4-bit blocks; each block is a 4-bit pattern. The dataset consist of 15 unique words divided in 4-bit block each. To reduce the number of unique words the data is partitioned into size of 4, which reduces the complexity of the decompression circuit. This also increases the speed of decompression. Each block pattern is mapped into a variable length code word. The length of the code word depends on the probability with which each pattern occurs in the data set.

TABLE III
PROBABILITY AND CODE TABLE

i	X_i	Occurrence	Probability	Huffman code
1	1111	30	0.3125	11
2	1011	12	0.1250	100
3	0111	10	0.1042	010
4	1101	09	0.098	000
5	1110	08	0.0833	1011
6	0110	05	0.0521	0011
7	0000	04	0.0417	10101
8	0010	03	0.0313	01111
9	1001	03	0.0313	01110
10	0101	03	030313	01101
11	1100	02	0.0208	01100
12	0100	02	0.0208	00101
13	1000	02	0.0208	00100
14	1010	02	0.0208	101001
15	0011	01	0.0104	101000
16	0001	0	0	UUUU
# States of FSM				15

The Table III shows the probabilities (pi) of occurrence of each unique pattern [xi] in the data of Table II. Following simulation results shows the data compression and decompression using Huffman coding. The result shows the compression in memory location required for data storage.

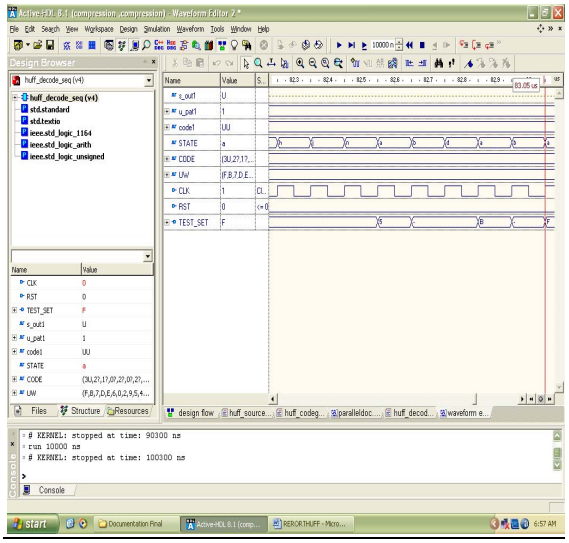

Fig.3. Simulation of the existing Huffman decoding system.

The Figure 3 shown above indicates the simulation result obtained for the existing Huffman coding system. From the simulation result it is observed that a total of 83.15µs is taken for the decoding of complete encoded data set. The signal 'Test set' shows the decoded data set retrieved for the obtained encoded data bits.

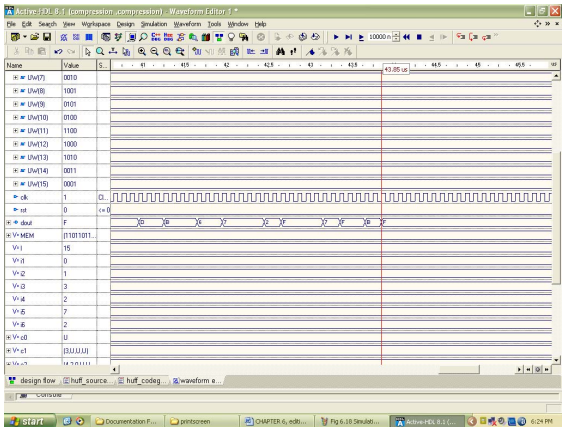

Fig.4.Simulation waveform of the High speed Huffman decoding system.

The Figure 4 shows the simulation result for the final timing simulation result obtained for the decoding of the complete data set. From the simulation result it is obtained that a total of 43.75µs simulation time period is taken for the decoding of

the complete data set. It is observed that about 50% of the time is reduced during the decoding process of the given data set using high speed Huffman decoder.

V. IMPLEMENTATION RESULTS

The proposed High speed decoding Huffman coding system design and the existing Huffman coding are implemented on to the targeted FPGA Cyclone EP1C20F324C6 using Quartus II. The chip planner resource utilization of the targeted FPGA for the high speed Huffman decoder system results about 63 % utilization, i.e. use a total of 13,280 Logic cells out of available 20,600 logic cells is shown in figure5.

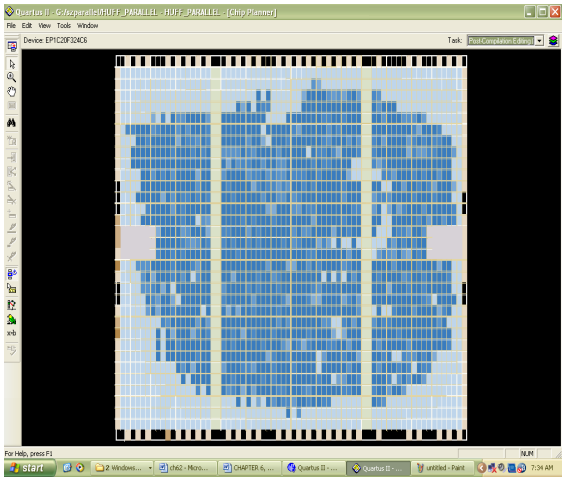

Fig.5. FPGA Chip planner resource utilization.

The Routing is carried out using FPGA Editor. It is observed that the average interconnects usage is 31% of the available device resources and the Peak interconnects usage is 57% of the available device resources.

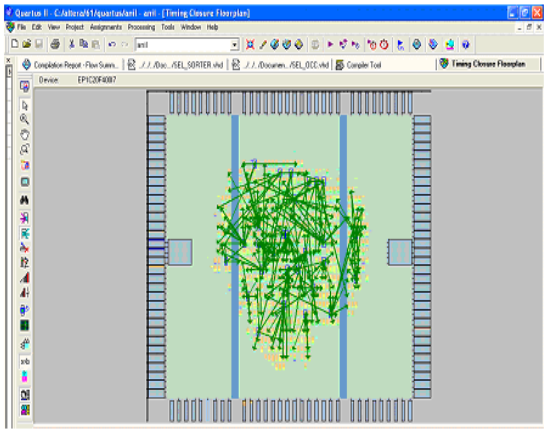

Fig.6. Timing closure floor plan using Chip Planner.

The Floor plan timing closure is shown in Figure-6. The timing closure report gives the Total cell delay = 102.658 ns (33.86 %) and the Total interconnect delay = 187.876 ns (63.14 %) for the Proposed system.

TABLE IV
COMPARISON OF IMPLEMENTATION RESULTS

Sl No	Parameter	Existing system	Proposed system
1	Logic cell	347	563
2	LC register	738	905
3	LUT only LC	128	218
4	Register only LC	133	160
5	LUT/ Register LC's	185	185
6	Critical Path Delay	316 nsec	309 ns
7	Total cell delay	112 nsec	103 ns
8	Total interconnect Delay	191 nsec	187 nsec

From the Table IV it can be observed that the Proposed System requires more resources on the FPGA that the Existing Huffman system.

VI. CONCLUSION

In this paper a high speed Huffman decoder is implemented using the concept of efficient partitioning of the multiple length code to isolated code register based on their length. The multi level Huffman decoder with multiple scan improves the scanning time for a test data set having less code length. We conclude that the High speed Huffman decoding system takes 50 % less time for decoding when simulated using Active HDL 8.1v and 33% more FPGA resources when implemented using Quartus II implementation tool compared to the existing Huffman decoding system.

REFERENCES

[1] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based ICs," in Proc. Int. Test Conf., 1998, pp. 448–457.

[2] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," in Proc. Design Automation Conf., 2001, pp. 151–155.

[3] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in Proc. Int Symp. Circuits Syst., 1989, pp.1929–1934.

[4] A. Chandra and K. Chakrabarty, "Test data compression for system-on-a-chip using Golomb codes," in Proc. VLSI Test Symp., 2000, pp.113–120.

[5] Efficient test data compression and decompression for system-on-a-chip using internal scan chains and Golomb coding," in Proc. Design, Automation, Test Eur., 2001.
[6] "Frequency-directed run-length codes with application to system-on-a-chip test data compression," in Proc. VLSI Test Symp., 2001, pp. 42–47.
[7] "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes," in Proc.Design Automation Conf., 2002, pp. 673–678.
[8] R. Chandramouli and S. Pateras, "Testing systems on a chip," IEEE Spectrum, pp. 42–47, Nov. 1996.
[9] R. Dorsch and H.-J. Wunderlich, "Reusing scan chains for test pattern decompression," in Proc. European Test Workshop, 2001, pp. 124–132.

[10] "Tailoring ATPG for embedded testing," in Proc. Int. Test Conf., 2001, pp. 530–537.
[11] P. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Improving compression ratio, area overhead, and test application time for systems-on-a-chip test data compression/decompression," in Proc. Design Automation Test Eur., 2002, pp. 604–611.
[12] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in Proc. Int. Conf. Computer-Aided Design, 1998, pp. 283–289.
[13] Xrysovalantis Kavousianos , Emmanouil Kalligeros, and Dimitris Nikolos," Multilevel Huffman Coding: An Efficient Test-Data Compression Method for IP Cores", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, VOL. 26, NO. 6, June 2007.
[14] Xrysovalantis Kavousianos , Emmanouil Kalligeros, and Dimitris Nikolos," Multilevel-Huffman Test-Data Compression for IP Cores With Multiple Scan Chains", IEEE Transactions on VLSI SYSTEMS, VOL. 16, NO. 7, July 2008.
[15] K. Ashok Babu and V. Satish Kumar, "Implementation of Data Compression Using Huffman Coding", International Conference on Methods and Models in Computer Science ICM2CS-2010, 2010,pp 65-70

SRIKANTH IMMAREDDY Received the Bachelor degree in Electronics and Communication Engineering (ECE) from the JNTU Hyderabad India and Master degree in Digital Systems from Osmania University, India in 2009.
He is working as Assistant Professor in ECE Dept at Methodist College of Engineering and Technology, India His main interests are in the fields of Very Large Scale Integration and Digital Signal Processing.

SRAVANKUMAR TALUSANI Received the Bachelor degree in Electronics and Communication Engineering (ECE) from Osmania University, India and Master degree in Digital Systems & Computer Electronics from JNTU Ananthapur, India.
He is working as Assistant Professor in ECE Dept at Methodist College of Engineering and Technology, India His major interests are in the fields of Analog Electronics and Signal Processing.