IJRSET March 2016 Volume 3, Issue 3 Pages: 22-30

ISSN 2394-739X

International Journal for Research in Science Engineering and Technology

SIMULATION ENHANCEMENT OF IMPROVING FPGA DEBUG METHODOLOGIES BY USING HAMMING SEC-DAED-TAED CODE

[1] A. Inna Mary, [2] Dr. VanajaShivakumar
[1, 2] M. Tech Department of ECE, Professor & HOD
[1, 2] Hindustan University,
[1, 2] Padur, Tamilnadu,
[1, 2] India.

Abstract:-

In the debug system, logic analyzer is built into FPGA. The debug module allows non-interfering real time debugging of software for the SoC microcontroller. Post –processing and analyzing the data is invaluable for system debugging. The prototype using trace-buffers to note a subset of internal signals into an on-chip memory for subsequent analysis. In this paper, high frequency methods are not suited for monitoring the fault occurrence in trace buffers. Therefore, low frequency method which is used to detect and correct the faults in trace buffers. The internal stages of the circuits are monitored and identified the faults and these faults are stored in the separate memory for analyzing the signals. In the proposed, the two methodologies are used.1) Finite State Machine (FSM) and 2) Hamming codes. The clock pulses are then divided by using clock divider. The finite state machine is used to monitor the errors and also the hamming codes are used to detect and correct the errors. Thus the tracing buffers are performed to monitor the signal state on FPGA.

Keywords: - Finite State Machine (FSM), Field Programmable Gate Array (FPGA), Central Processing Unit (CPU).

1. INTRODUCTION

The capacities of FPGAs grow, fortifying that a design is functionally correct (verification and validation) and detecting the sources of any observed incorrect behavior (debugging) has become increasingly difficult. Logic verification is known to be one of the major challenges of contemporary chip design. Simulation based techniques have the most common verification approaches due to their notable flexibility. Field programmable gate array device interconnect is designed to match chip interconnect exactly and the partitioning of the chip logic onto the multiple FPGA devices generally done by hand. To increasing the device capacity as well as limited on-chip observability. Verification and debugging both make comprehensive use of software simulators. Simulation gives full-chip visibility and fast turnarounds between design changes. However, the simulation of big designs can be extremely slow. [1] For an example, Intel announced that software simulations of

their core i7 chip ran one billion times slower than on real silicon, with the addition of all their simulation efforts on a big server farm culminating in no more than a few minutes of actual of actual chip operation [2]. The design can be run at speed meaning much deeper traces are possible. Testing the FPGA in-situ may allow realistic input stimuli, the device can be connected to the other chips in the target system. A primary challenge is the limited observability of the signals on-chip with hardware validation. FPGA allow a "snapshot" of all state bits to be taken, which can be read-out using a JTAG interface, however, this does not permit easily for tracing signals over time. The trace buffers are memories that record the activities of particular signals over a number of cycles. Trace buffer technology has been proposed to track a small number of internal state elements within a capture window when the chip is operating. These signals are selected for tracing at the design stage and the traces are examined at the post-silicon stage to debug logic errors. The collected traces are used to restore as many other state elements within the capture window. In addition to trace buffers, these tools instantiate connections that connects the traced signals to these buffers. This has some drawbacks: 1) recompilation can be slow, especially in prototyping systems consisting of multiple FPGAs, 2) a recompilation may cause timing differences which may obscure the bug that was being sought and 3) additional routing stress may cause a previously routable design to become unroutable.

In this paper, finite state machine and hamming code methodologies are used to monitor the error and also detect and correct the error occurs in trace buffers. At low frequency, which is used to detect and correct the faults. The detected and corrected faults that are stored in the separate memory for analyzing the signals.

2. LITERATURE REVIEW

[Ko. H.F. et al, 2009] described the algorithms in silicon debug for state restoration and trace-signal selection for data acquisition.. Silicon debug can be divided into two main steps data acquisition and analysis. An accelerated algorithm for restoring circuit state elements from the traces collected during a debug session by exploiting bitwise parallelism is presented. New metrics that guide the automated selection of trace signals, which can improvethe real-time observability during in-system debug, was also introduced. To detct and correct the faults that escape pre-silicon verification here, the accelerated algorithms for restoring circuit state elements. State restoration, the algorithm only needs to check if data can be redesigned at a circuit node and no branching and backtracking will be done if unsuccessful, undefined values will be concluded. [Raman .S. et al, 2011] explained timing constrained routing algorithm for symmetrical FPGAs which incorporates a novel incremental routing strategy. Experimental results confirm that the algorithm reduces delay along the longest path in the circuit, uses routing resources efficiently and requires low CPU time. The incremental routing technique has a significant effect on both the timing performance and routability. The benchmark circuits demonstrate that the algorithm performs very well with respect to timing, routability, resource utilization and computation. [Hung .E. et al , 2012] explained the capacities of FPGA increases its verification and validation is difficult so, on chip observability is used in order to enhance on-chip observability , but doing so often requires re-compiling the whole design for each new trace configuration. To explore the limitations of incremental-synthesis for trace –buffer insertion, and to execute CAD optimizations exclusive to this application for improving runtime and routability is presented. During incremental tracing, there

exists far more flexibility than with functional design changes- a traced signal is not constrained to reaching one particular sink.

[Asaad .S. et al, 2012] described the cycle-accurate and cycle-reproducable large-scale platforms of FPGA that is designed from the ground up to accelerate logical verification of the Bluegene/ Q compute node ASIC, a multi-processor SOC implemented in IBM's 45nm SOI CMOS technology. The challenges for constructing such large-scale platforms of FPGA, including design partitioning, clocking & synchronization, and debugging support, as well as addressing these challenges without sacrificing cycle accuracy and cycle reproducibility.

[Li .M. 2014] explained the post silicon debug is the last step of debug process for faster and accurate debug an hybrid approach is employed. Here, the State Restoration Ratio (SRR) is used to measure the quality of a set of selected trace signal .Metric-based algorithms utilize metrics which permit approximating the capability of a candidate trace signal to restore the untraced state elements while taking into account the restoration that can be made from a subset of already –selected trace signals.

3. FI NITE STATE MACHINE

Finite State Machines (FSM), used in sequential logic, have outputs that depend on both the current input as well as the history of the input.FSM is composed of a combinational logic unit and flip-flops placed in such a way as to maintain state information. FSM generating sequences of control signals instructs data path what to do next. In Mealy machine, sequential system where output depends on current input and state. In Moore machine, sequential system where output depends only on current state. Finite state machine is used to monitor the faults in trace buffer. The state diagram of finite state machine as shown in fig.1.

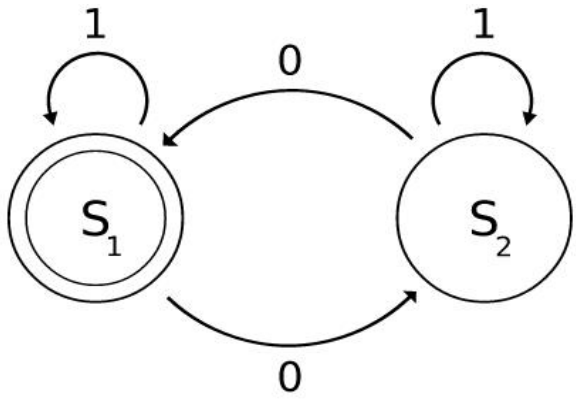

Figure .1 State diagram of finite state machine

4. HAMMING ENCODER AND DECODER

Hamming code is an Error Correction Codes (ECCs) in which single bit error can be successfully detected and corrected. It can detect up to three adjacent bit errors but can correct only single bits error. In mathematically, hamming codes are characterized by the following parameters,

$$n = 2^m - 1 \quad (1)$$

$$k = n - m \quad (2)$$

$$d_{\min} = 3 \quad (3)$$

where, n is the block size, k is the number of information bits, d_{min} is the hamming minimum distance and m is the parity check bits. For a single bit error correction, d_{min} is three that means three parity bits are used to detect and correct a single bit error. The parity bit is further extended for more number of adjacent bit error detection. The process of hamming encoder is as follows: (1) Number the all position of input bits starting from 1 to m, where m is the last position of the bit. (2) Convert all the positions are in their binary form as 1, 10, 11, 100, 101 etc. (3) All bit positions that are powers of two are considered as a parity bits.

Step 1: Check 1 bit and skip 1 bit steps are followed such as 1, 3, 5, 7, 9.....

Step 2: Check 2 bits and bound 2 bits steps are followed such as 2, 3, 6, 7, 10,

11…..

Step 3: Check 4 bits and skip 4 bits steps are followed such as 4, 5, 6, 7, 12,

13, 14, 15.....

Step 4: Check 8 bits and skip 8 bits steps are followed such as 8-15, 24-31,

40-47.....

Parity check bits for input bits are obtained when combined results coming from all steps. The code word for input bits is obtained when locating parity check bits into respective places (that are powers of two). A lexicographic hamming matrix for 8-bits is illustrated in equation (4). The transpose of lexicographic hamming matrix is known as Syndrome matrix. Hamming distance of this vector is 4. Therefore, from this matrix, we can detect up to two bit error and can correct a single bit error.

$$H=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

Hamming decoding process is as follows:

Step 1: To multiply the code word with syndrome matrix.

Step 2: Results from step 1 called as syndrome vector. If all the bits of syndrome vector are zero, then there will be “no error” in the decoded output. If any one of the bits is non-zero, then there will be “error” in transmission. If the syndrome vector is in from (0000) to (1100), then there will be a “single error” in transmission and with the help of syndrome vector, we can easily correct it. If the syndrome vector is in from (1101) to (1111), then there will be only detect the double bit adjacent error, but cannot correct it.

Using an example for Hamming code (12, 8), data bits (01011100) are coded as (100010101100). Multiplication of code word (100010101100) and syndrome matrix is given as (0000), which is called as syndrome vector. All the bits of syndrome vector are zero; hence, there will be no error in data transmission. If the fifth bit of code word is flipped as 0. Hence, code word as (100000101100), then the syndrome vector obtained as (0101). The syndrome vector takes 5th location of syndrome matrix. Hence, there is possibility to change the fifth location of code word. In this way hamming codes can detect and correct a single bit error.

5. SYSTEM ANALYSIS

A. Trace Buffer

On the FPGA, trace buffers are formed from a memory resource .Trace buffers record a limited-size history of the signals connected to them during regular device operation. Designers verify functionality or hunt bugs by properly adjusting the trigger conditions and analyzing the trace buffer data. Multiple trace buffers are distributed to observe and record nearby user signals. The trigger unit requires a region of logic to detect conditions. Incremental distributed trigger insertion, we distribute the logic elements that make up the trigger function across logic elements that are not used by the user circuit. These logic elements may not be contiguous. Functional definition errors can be doubly difficult to find since the designer has not understand a particular requirement, so the fault can be overlooked even when looking carefully at the important details of the design. An example of a functional definition error would be where a state

machine transition doesn't end up in the right state.

B. Incremental Trace Insertion

The proposed trace buffers and a trigger unit are inserted incrementally in an already placed-and-routed design, to increase the observability of FPGA circuits. It refers to the already placed-and-routed design as the original circuit or user circuit and the trace buffers and trigger that are incrementally inserted as the debug system. The incrementally inserting the debug system because it will reduce the impact on the original circuit's area, placement, routing, and timing. Incremental insertion allows us to adjust the size of the debug system to fit into whatever the original circuit does not use. The amount of trace buffers and trigger logic we can insert will be influenced by the area of the original circuit.

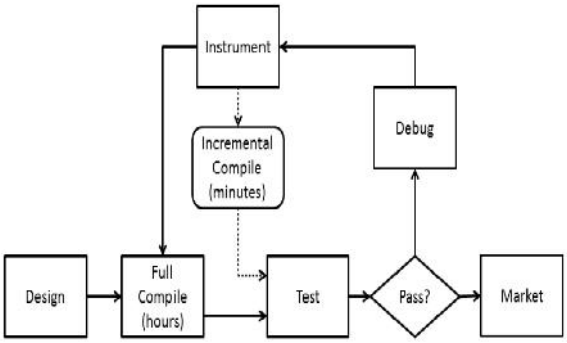

Figure.2. Design and debug flow demonstrating how incremental compile reduces the time

The fig.2 shows how the time taken for the execution has been reduced after using incremental technique. The goal of incremental synthesis is generally to modify the functionality of an existing circuit with minimal changes to its current placement and routing. The placement and routing of trace buffers and trigger unit is restricted to resources unused by the circuit.

C. Block diagram of FPGA debug trace buffer

The fig.3 is the block diagram of FPGA debug trace buffer. From the clock generator, the input clock pulse gets generated. These clock pulses are the divided using a clock divider. The input data is given to circuit and this produces the corresponding output. The internal states of the circuits are stored using a trace buffer. Various trace buffers are used for storing the various signal states. If any error occurs, it will be stored in a separate memory and monitored. The power supply is the input supply given to the hardware. The system clock is the basic clock pulse given to the hardware and the hardware reset is the initial condition for resetting of the hardware.

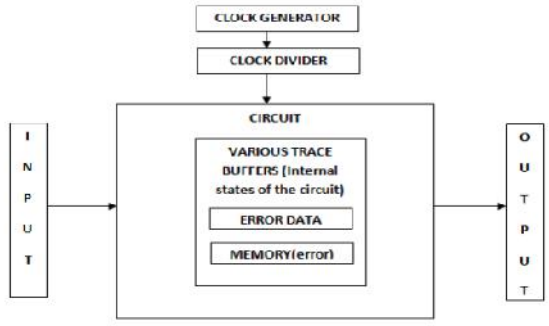

Figure .3. Block diagram of FPGA debug trace buffer

D. Internal Circuit diagram of trace buffers

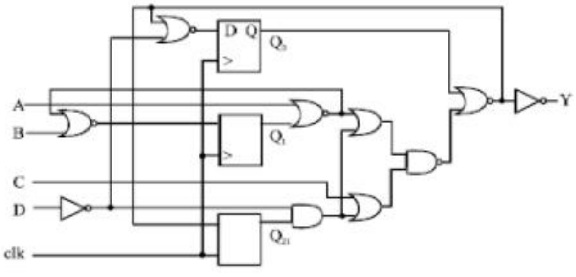

Figure.4 Internal Circuit Diagram of Trace Buffers

The fig.4 shows the internal circuit diagram of trace buffers. Here the circuit is used in sequential circuit. Sequential logic is a type of logic circuit, in which the output depends on present input and previous output. In a combinational logic, the output depends on the present input. Sequential logic has state (memory) while the combinational logic does not. A sequential circuit is to construct finite state machines, a basic building block in all digital circuitry, as well as memory circuits. Digital sequential circuits are divided into two types, synchronous and asynchronous types. The state of the device changes only at discrete times in response to a clock signal, in synchronous circuit. The state of the device can change at any time in response to changing inputs , in asynchronous circuits. Memory is not used in combinational circuit. Hence the previous state of input does not have any effect on the present state of the circuit. But sequential circuit has memory. Based on the input, the output can vary. This type of circuits uses previous input, output, clock and a memory element.

6. PROPOSED HAMMING CODE BASED FPGA DEBUG TRACE BUFFER

In this paper, proposed hamming code based FPGA debug trace buffer is designed. In the hamming code, which is used to detect and correct the error in trace buffer. The internal stages of the circuits are monitored and identified the faults and these faults are stored in the separate memory for analyzing the signals. Proposed hamming code based PPGA debug trace buffer is shown in fig.5.The internal circuit of three outputs are given to the input of 3:8 decoder. The 3:8 decoders are given to the hamming encoder. Hamming encoder is used to detect and correct the SEC-DAED-TAED errors. In hamming encoder, 8 bit is converted into 12 bit and the 12 bit is given to the input of hamming decoder. Hamming decoder is given to the 8:3 encoder. Finally, 8:3 encoders is given to the detected and corrected output. When compared to finite state machine, hamming code gives a better performance. The faults are monitored and corrected in trace buffer only at low frequency.

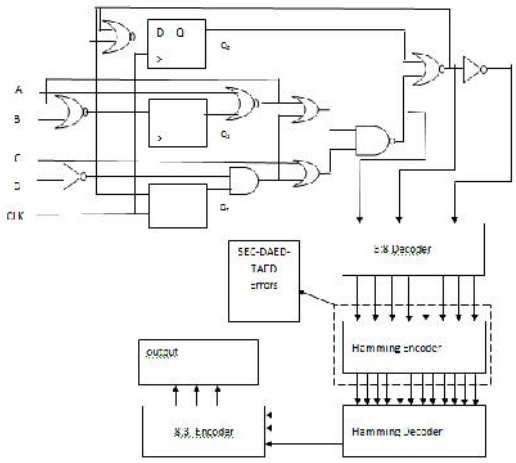

Figure.5 Proposed Hamming code based FPGA debug trace buffer

7. RESULTS AND DISCUSSION

The design of proposed hamming code based FPGA debug trace buffer has been made by using Verilog Hardware Description Language (Verilog HDL). The simulation results have been evaluated by using ModelSim 6.3C. The simulation output for generating glitches at frequency_3(2MHz) is shown in fig.6.

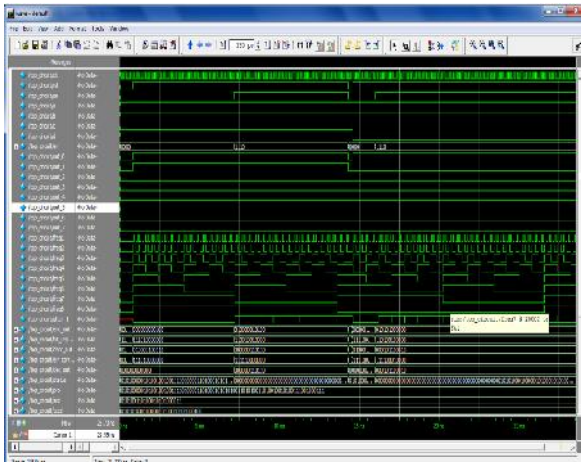

Figure.6 Simulation output for generating glitches at frequency_3 (2MHz)

The simulation output for generating glitches at frequency_4(1MHz) is shown in fig.7.The simulation output for detecting adjacent bit error using hamming code is shown in fig.8. Designing the circuit and debugging it and locating the errors by means of signal tracing. The signal tracing concept is applied to analyze the internal signals. The occurrences of glitches, the trace buffers are used for storing the internal signals during debug. Trace buffers are also used for locating errors.

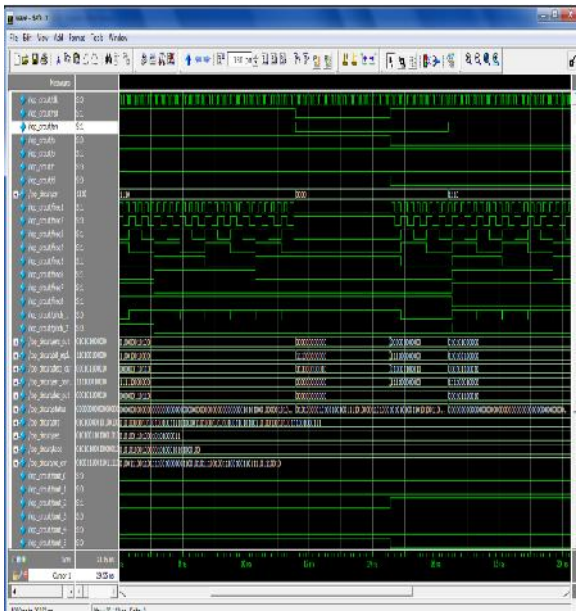

Figure.7 Simulation output for generating glitches at frquency_4 (1MHZ)

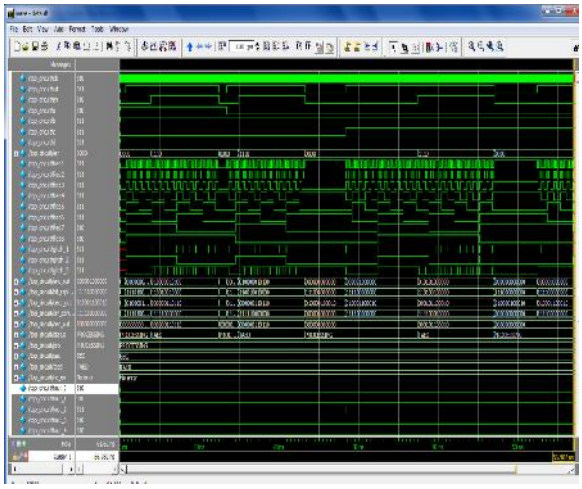

Figure.8 Simulation output for detecting adjacent bit error using hamming code

CONCLUSION

In this paper, the proposed hamming code based FPGA debug trace buffer has been designed through Very Large Scale Integration (VLSI) design environment. In these design is used to detect and correct the errors in a trace buffer. Finite state machine is used to only detect the error. Hamming code which is used to detect and correct the error. Hamming codes are detected and corrected the SEC-DAED-TAED code. These codes are used to avoid the wrong information passes through the space communication. The SEC-DAED-TED is required to add one parity bit, which takes more amount of computational time for detecting the error. When compared to FSM, hamming code gives a better performance. For improving the FPGA debug method, by using hamming SEC-DAED-TAED codes. The corrected errors are stored in a memory of the circuit. The trace buffers are performed to monitor the signal state of FPGA.

REFERENCES

[1] E. Hung and S. J. E. Wilton, “Limitations of incremental signal tracing for FPGA debug,” in Proc. Int. Conf. Field Program. Logic Appl., Aug. 2012.

[2] H. F. Ko and N. Nicolici, “Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 2,, Feb. 2009

[3] Min Li and Azadeh Davoodi, “A Hybrid Approach for Fast and Accurate Trace Signal Selection for Post-Silicon Debug”2013

[4] S. Raman, c.l liu (2011), “A Timing-Constrained Incremental Routing Algorithm for Symmetrical FPGAs Symmetrical FPGAs”

[5] S. Asaad, R. Bellofatto, B. Brezzo, C. Haymes, M. Kapur, B. Parker, T. Roewer, P. Saha, T. Takken, and J. Tierno,

"A cycle-accurate, cyclereproducible multi-FPGA system for accelerating multi-core processor simulation," in Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays, 2012

[6] Xilinx. (2012, Dec.). ChipScope Pro 12.3, Software and Cores, User Guide, Sanjose,CA,USA[Online].Available:http://www.xilinx.com/support/document.