ISSN 23499842(Online), Volume 1, Special Issue 2(ICITET 15), March 2015
International Journal of Innovative Trends and Emerging Technologies

IMPLEMENTATION OF TRACE BUFFER SIGNAL STATE DETECTION OF FPGA

S. SURIYAPRIYA[1], P. SATHIYA[2]
[1]PG Student, VLSI Design, [2]Assistant Professor,[1,2] Department of ECE,
[1,2]Surya Group of Institutions, India

Abstract*:*
The Integrated circuits (ICS) made up of multibillion transistor counts designing. Ics have more functionality and complexity. Verifying that the devices is difficult task. In using trace buffer for signal state it need to avoid full recompilation process. Signal state detection it can be a state by state to insert trace buffer into the circuit to store the error into the buffer*To* propose trace buffer into the circuit are used to record a limited set of internal signals it can be debugging. Giving challenge to this techniques are used to eliminate the need for full recompilation. Apply Incremental trace buffer insertion for Signal State can be 98 times faster than a full recompilation..Using circuit to insert the trace buffers into only a limited set of internal signals. Debugging process will be continuous It can be avoid full recompilation process. Error can be occur in a signal state to identify that the error location. Trace-insertion can be 98 times faster than a full recompilation process

I. INTRODUCTION

a) Need for Trace Buffer

Integrated circuits (ICs) have revolutionized the world of electronics. It is critical that IC's function correctly or there can be costly or even deadly consequences. Verification is an important part of IC product development and FPGAs are increasingly being used to do it. The major disadvantage of using FPGAs for verification or debug is observability, the ability to see FPGAs' internal values. Observability is key to verifying behavior and tracking down the cause of bugs. Simulators can provide full observability into all signals of a circuit, but on FPGAs and other physical prototypes only a small subset of signals can be observed through the external pins.

b) Incremental Trace Buffer

Trace buffers are formed from a memory resource on the FPGA. 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 examining trace buffer data. multiple trace buffers are distributed to observe and record nearby user signals. Distributing the trace buffers reduces the distance signals must be routed and improves circuit timing. Including more trace buffers will allow more signals to be observed. A centralized trigger unit controls the operation of all the trace buffers with a single output signal that halts their recording when necessary. 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. Intuitively, this will allow the trigger logic to take better advantage of any left-over space after mapping the user circuit, adapting to changes in the size or make-up of the trigger function. After placing the logic cells in unused locations, the logic cells must be connected to each other and to the user circuit using incremental routing techniques. Sample a subset of signals into on-chip memories. Capturing a sequence of states, at full speed. It does not cost and extra silicon area occupied. FPGAs commonly not filled to capacity.

c) Trace Buffer Insertion For Signal State

To propose trace buffer into the circuit are used to record a limited set of internal signals it can be debugging.

Giving challenge to this techniques are used to eliminate the need for full recompilation.Apply Incremental trace buffer insertion for Signal State can be 98 times faster than a full recompilation.

d) Integrated Circuits(Ics)

Integrated circuits (ICS) made up of multibillion transistor counts designing.Ics have more functionality and complexity.Verifying that the devices is difficult task.

In using trace buffer for signal state it need to avoid full recompilation process.Signal state detection it can be a state by state to insert trace buffer into the circuit to store the error into the buffer.

II. RELATED WORK AND CONTRIBUTIONS

The Explored the limitations of Applying incremental-synthesis techniques. To the insertion trace-buffers for improving on-chip observability and more flexibility. The advantages of For improving runtime and routability. The routing architecture employed is a modern unidirectional fully buffered network, which means that adding extra fan-out loads to existing nets will not affect the original circuit timing. It should be a faster compilation process

FPGA design validation and debugging can often be done by executing the hardware direct Our paper discusses how directly incrementing FPGA programming data, or bit streams are executing

With debugging hardware can improve the debugging productivity Reduce a design's time to market. Design modification speed-ups ranging from about 6 to 19 times over more conventional techniques. Speculative (accurate) Debug Insertion for FPGAS in Using Testing and Verification of Integrated circuits(Ics).Speculative (accurate) Debug Insertion which a tool automatically predicts the Signals. What the signals will be First Debugging Process that the signals will be First compilation. If done correctly, this accelerate (Continuous) the debug process. This debugging task is difficult, primarily due to a lack of observability. Field Programmable Gate Arrays (FPGAs) are flexible, programmable devices Basic structure consists of an array of universal, programmable logic cells. Logic cells vary in their complexity, but the idea is to provide a universal design element. The software designer also present with views of the Technology. Routing FPGAs is a challenging problem of routing resources both wires and connection points. Slow implementations Caused by long wires. Delay is minimized by allowing the more critical signals. The original circuit mapping is fully preserved and incremental techniques are used to eliminate the need for a full recompilation, thereby accelerating the debugging process. By exploiting two opportunities available during trace-insertion: the ability to connect from any point of a signal to any trace-pin, and the internal symmetry of the FPGA architecture, we find that incremental trace-insertion can be 98 times faster than a full recompilation, return a routing solution with a shorter wire length, and have a negligible effect on the critical-path delay of the original circuit when reclaiming 75% of the leftover memory capacity for tracing.

III PROPOSED OF TRACE INSERTION

The proposed approach Using circuit to insert the trace buffers into only a limited set of internal signals. Debugging process will be continuous It can be avoid full recompilation process. Error can be occur in a signal state to identify that the error location. Trace-insertion can be 98 times faster than a full recompilation process Current FPGA trace-solutions such as Xilinx Chip Scope. Pro, Altera Signal Tap II, Synopsys Identify, and Tektronix.Certus [2]–[5] all operate primarily on the remapped circuit. That is, these tools will instrument the original user-circuits with trace-buffers and their connections before place-and routing the combined design, although several of these tool scan also support a limited amount of post mapping reconfiguration. A simplified illustration of what we defined to be the preamp, midmap, and postmap stages of the FPGA.Adding trace-instrumentation to the circuit after logic synthesis, where the circuit is already transformed from a high-level description (such as Verilog) into low-level FPGA primitives [lookup tables (LUTs)] means that the designer is restricted to only observing gate-level signals. These gate-level signals, through logic optimizations and technology mapping, may not have a direct correspondence to the original hard ware description language (HDL) signals. We believe several approaches exist to alleviate this mismatch: initially, unless register retiming is performed, both commercial and academic CAD tools are able to preserve the HDL-to-gate mapping for flip-flops in the circuit. Designers can therefore use the elements as fixed points of reference into their Verilog code, or to use the data collected for off-line simulation to compute all intermediate, combinational signals. Secondly, designers
are able to manually specify additional points of reference by using synthesis attributes to force the CAD tool to maintain this HDL-to-gate correspondence: (* syn_keep *) is supported by Simplify and Quartus II tools, while ISE users can apply theS (SAVE_NET) attribute to do so. Implicitly, existing trace IP
such as Chips cope Pro, SignalTap II, and Certus already do this when incrementing a circuit presynthesis

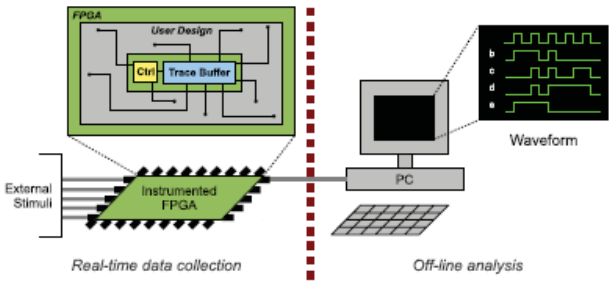


Fig 1. Trace-based debugging

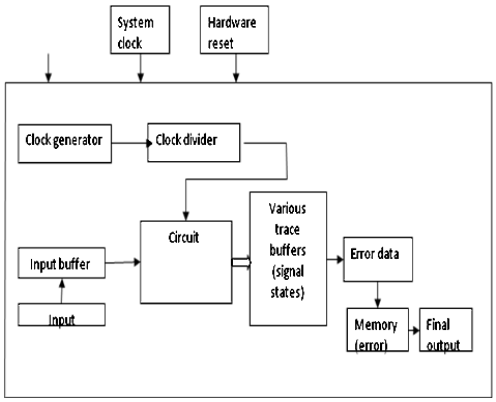


Fig 2 Block Diagram for Trace Buffereing

Clock generator: From the clock generator, the input clock pulse gets generated.
Clock divider: The input data is given to the input buffer where these inputs are stored. Circuit: This is then given to circuit and this produces the corresponding output. This circuit's internal signal states are stored using a trace buffer. Various Trace Buffers: Various trace buffers are used for storing the various signal states. Error Memory: If any error occurs, it will be stored in a separate memory and monitored it cannot be Recompilation process: 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.
Scan-based techniques: Scan-based techniques rely on the serial connection of all of the flip-flops in a circuit

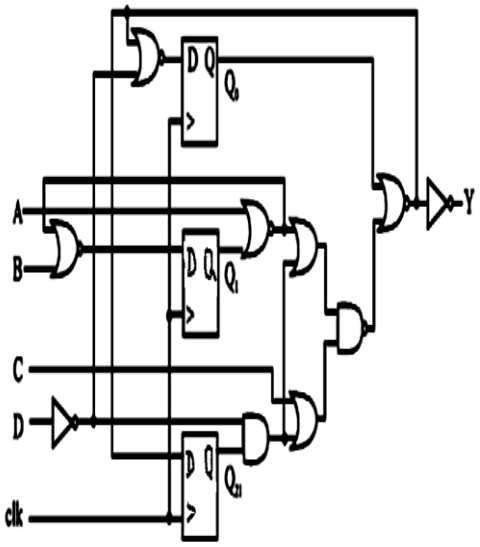


Fig 3. Circuit Diagram for Trace buffer

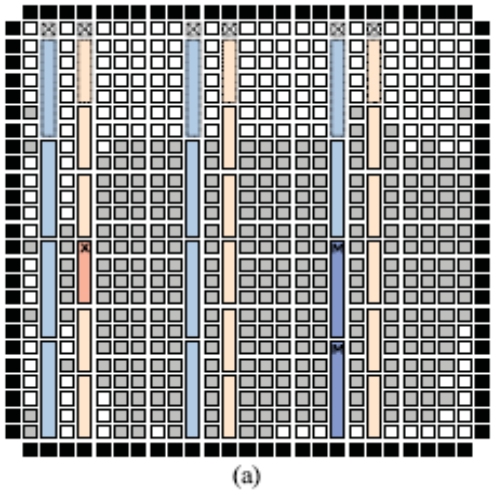


Fig 4 Baseline uninstrumented circuit

Enhancing Observability:
Methods to enhance device visibility during hardware validation.
Incremental synthesis:
Incremental synthesis is to allow the functionality of a fully place-and-routed circuit. Implementation to be modified. In this paper, we refer to the use of incremental techniques for transparently inserting trace-buffers as incremental-tracing.

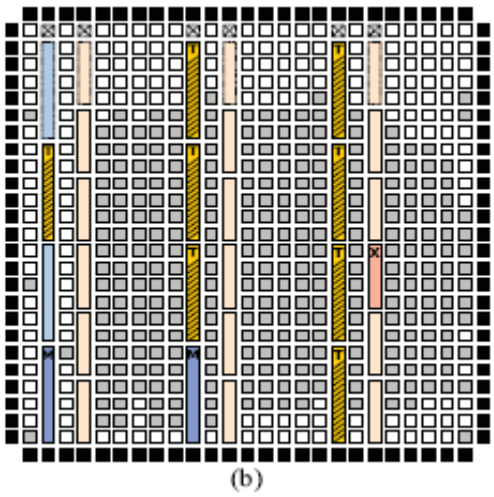


Fig 5 Instrumentation without incremental techniques

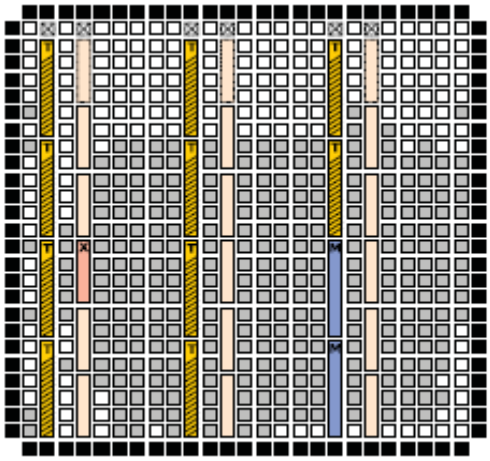


Fig 6 Instrumentation with incremental-techniques

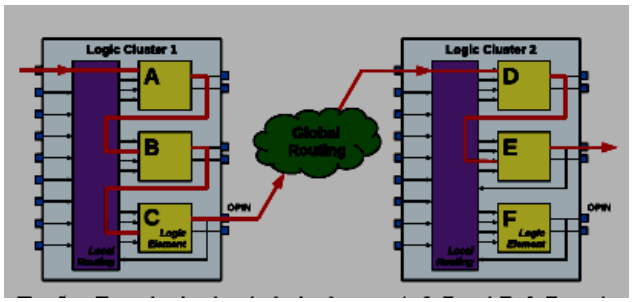


Fig 7 Example signal path; logic elements A & B and D & F can be incrementally swapped for greater trace-flexibility.

During post map trace-insertion, one rather unique opportunity not previously available when mapping the original user logic exists: a selected signal needs only be incrementally routed to any free trace-buffer input pin for its signal values to be observed. This differs from user-logic in that signals do not need to be connected to all of its sinks to create a valid routing solution; for incremental-tracing, by treating all inputs-pin of all trace-buffers of the FPGA as potential sinks, a connection to any pin is sufficient to allow observability.Fig. 7 shows this concept—any trace-pin connection along the dashed routes (or along any other combination of routes not shown) will be sufficient. Recall that, because of our assumptions, we can convert every unused memory block into a trace-buffer. In addition, for nets that utilize the fully buffered global interconnect such as that shown, any point of the net can be tapped to make this connection. This many-to-many capability provides two advantages: 1) significantly improved routing flexibility and 2) CAD runtime, both of which are especially important given the self-imposed constraint that during post map insertion, we prevent any existing user-routing from being ripped up. Routing algorithms can then be modified to search for any trace-pin and finish as soon as one is found. These algorithms are commonly coupled with Pathfinder with which our techniques are also compatible.

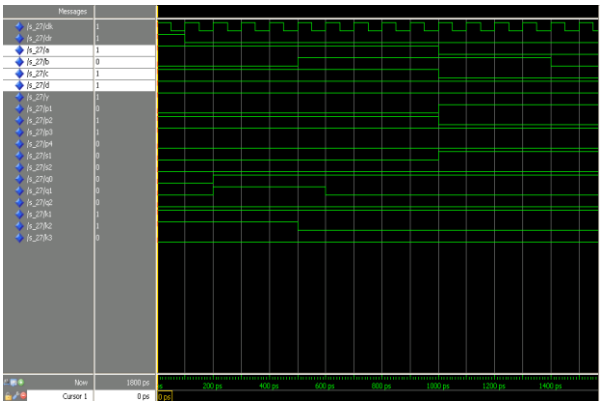
TABLE I

Comparison table

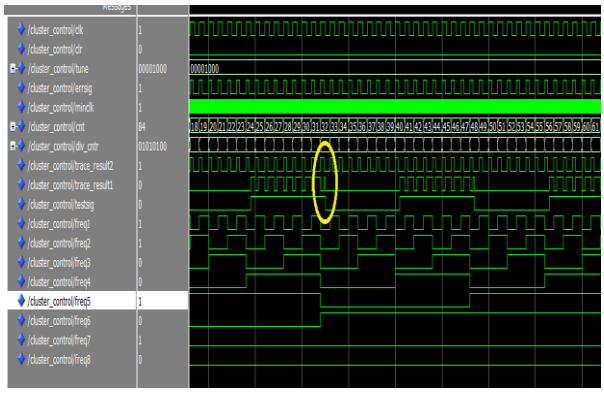
S.NO	PARAMETERS	EXISTING SYSTEM	PROPOSED SYSTEM
1.	AREA	] 702.51	3/154.08
2	POWER	95.4mW	65.45mW

IV SIMULATION RESULT & VERIFICATIONS

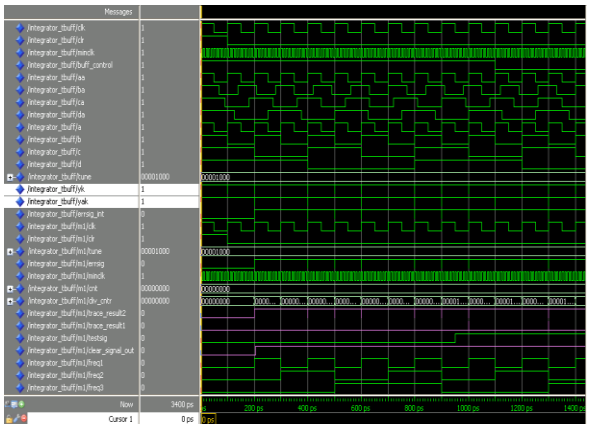
a) Simulated Output For S27 Circuit For Fugal Debug



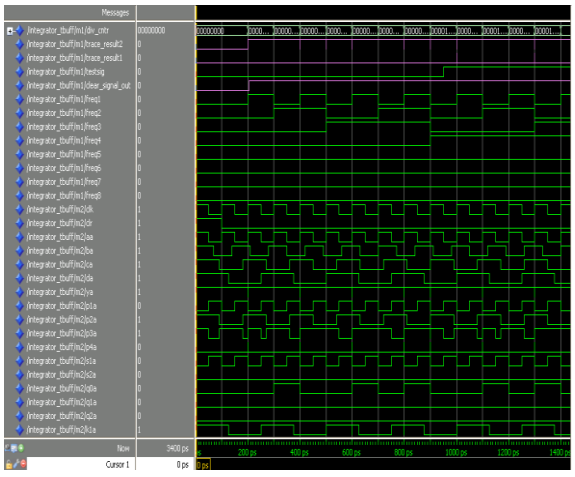
b) Simulated Output Design And Analysis Of Trace Buffers



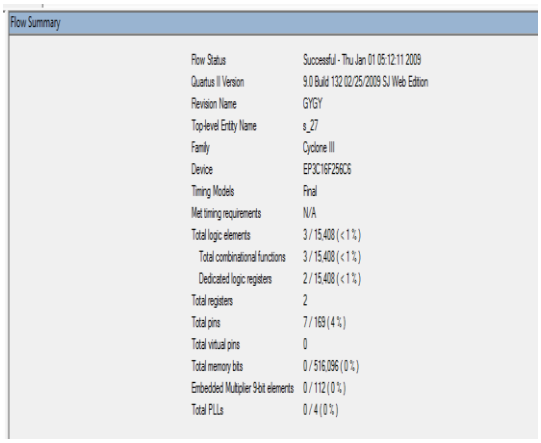
c) *Simulated Output Analysis Of Errors And Memory Module*



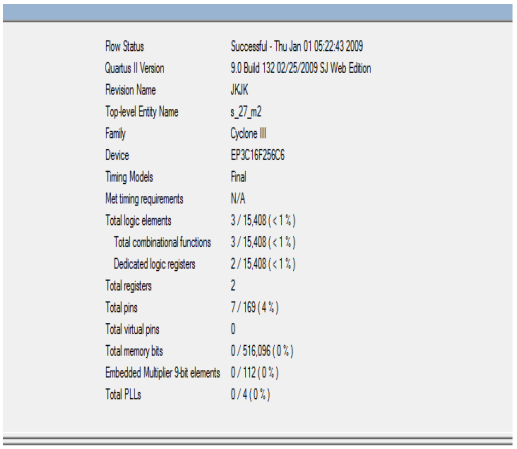
d) *Simulated Output Design And Analysis Of Integration Module*



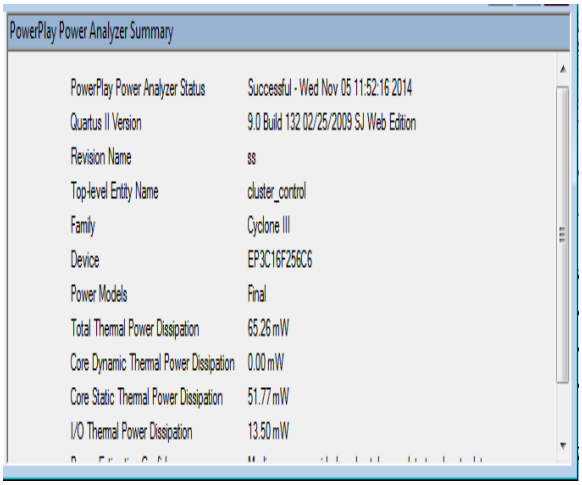
e) *Design And Analysis Of A Circuit For FPGA Debug*



f) Design And Analysis Of Trace Buffers



g) Design And Analysis Of Errors And Memory Module



V. CONCLUSION

This paper has presented a high-speed BCH based hamming decoder for correcting multiple error as well as single-bit errors in serial and parallel manner. When a multiple-bit error occurs including a double-adjacent error. High-speed correction of double-adjacent errors is included in the proposed scheme, because double-adjacent and single-bit errors are much more frequent in a memory system. The RTL schematic diagrams are show in the figure. Successful implementation of the BCH based Hamming Code Single, Two, Three, Four Bit Error Detection and Correction using the Xilinx ISE.12.4 version and output wave forms are shown in figure above. Check the output waveform is shown in figure. This Hamming Encoder can be implemented for any Memory circuit and in communication devices to prevent Data loss. The experimental results showed that the proposed scheme has a lower complexity in terms of area compared with the normal BCH based techniques as well as more number of bits are correctly serially in fast manner with less mathematically complexity.

.

REFERENCES

[1] E. Hung and S. J. E. Wilton, in Proc. limitations of incremental nal-tracing for fpga debug Int. Conf. Field Program. Logic Appl., Aug. 2012, pp. 49–56.

[2] P. Graham, B. Nelson, and B. Hutchings, “Instrumenting Bitstreams for Debugging FPGA Circuits,” in Proc. 9th Annu. IEEE Symp. Field-Program. Custom Comput. Mach., Mar. 2001, pp. 41–50.

[3] S. Trimberger, Field-Programmable Gate Array Technology. Norwell, MA, USA: Kluwer, 1994.

[4] L. McMurchie and C. Ebeling, “PathFinder: A negotiation-based performance-driven router for FPGAs,” in Proc. ACM 3rd Int. Symp. Field-Program. Gate Arrays, 1995, pp. 111–117.

[5] R. Y. Rubin and A. M. DeHon, “Timing-driven pathfinder pathology and remediation: Quantifying and reducing delay noise in VPR-pathfinder,” in Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays, 2011, pp. 173–176.