Design, Test and Evaluation of Trace-Buffer Inserted FPGA System

R.S. Karpagam and B. Viswanathan

Abstract Integrated circuits have more functionality and complexity, and to verify that these devices working properly in all scenarios is a difficult task. For using traditional verification techniques such as software simulation, designers are taking advantage of the significantly higher clock speeds which can be achieved by using field-programmable gate-array (FPGA)-based prototypes. However, the major challenge of using FPGAs for verification and debug is observability. Designers must use special techniques to observe the values of FPGA's internal signals. In this paper, a new method has been proposed for increasing the observability of FPGAs and demonstrates its feasibility. The new method incrementally inserts the Trace-Buffers controlled by a trigger into already placed-and-routed FPGA designs.

Keywords FPGA · Verification · Debug · Incremental synthesis · Observability

1 Introduction

Now, the integrated circuits (ICs) are reaching more than a billion transistor counts which make them insignificant to design these complex devices. Verifying whether these circuits are working properly under all operating conditions (expected and unexpected) is a difficult task. To overcome this, now many designers have turned to field-programmable gate-array (FPGA)-based prototyping to improve the verification coverage more than the achievable range using the traditional software simulations. FPGAs are an attractive platform for verification and debugging because they are much faster than simulation and less costlier than fabricating an ASIC prototyping. IBM researchers stated that FPGA prototype is 100 000 times

R.S. Karpagam (✉) · B. Viswanathan
Department of Electronics and Communication Engineering, SRM University, Chennai, India
e-mail: rasiaum@yahoo.com

B. Viswanathan
e-mail: viswanathan.b@ktr.srmuniv.ac.in

S.S. Dash et al. (eds.), *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, Advances in Intelligent Systems and Computing 394,
DOI 10.1007/978-81-322-2656-7_96

faster than software simulations and 400 times slower than the fabricated application-specific integrated circuit (ASIC) [1]. An ASIC prototype could achieve even greater speeds than an FPGA, but FPGAs have other advantages over ASIC prototypes. The lead time for fabricating an ASIC can be weeks or even months and can easily cost over 1 million USD [2]. The FPGA is used to identify the sources in correct behavior. Verification and debugging both increases the device capacity and limited due to on-chip observability. These make extensive use of software simulators. The simulation is extremely slow for large design. To overcome this problem, the prototype using Trace-Buffer is to be instrumented, for recording a subset of internal signals into on-chip memory to get subsequent analysis. To do so, ChipScope Pro, Certus, and SignalTap II can be used [3–6].

Trace-Buffer is formed from a memory resource on the FPGA. Trace-Buffer records a limited size history of the signals connected to them during regular device operation. This enables the designer to run the device normally. It can extract the signal history by the Trace-Buffers with techniques such as device reed back for off-line analysis [7].

The results and challenges of using the approach may differ between architectures. However, the approach could be used on any FPGA where it is possible to incrementally insert (place and route) additional logic and memory components in an already existing circuit.

The proposed approach has some specific advantages: (1) It has no impact on the placement or routing of the user circuit. (2) It requires no additional area. (3) It enables faster turn-around time for changing the observed signals or Trace-Buffer compared to traditional flows. (4) It increases FPGA observability by taking full advantages of all leftover memory.

2 Background

2.1 Enchancing Observability

A limited number of signals may be observed in FPGA's pins using an external logic analyzer. But, this is often inadequate and some of the pins may be used already. Methods to observe a large number of signals can be divided into two broad categories: scan-based and trace-based.

(1) *Scan-based*: Scan-based debug approaches are used to capture the state of an FPGA for inspection by serially shifting it out over an external pin or an interface such as JTAG. The state of the FPGA is the values in all memory elements on the chip, such as flip-flops and embedded blocks. For serially shifting out the Xilinx device uses the readback feature [7]. In [8], it was shown how readback data can be used for debugging in a combined simulation/hardware execution environment built on JHDL. However in [9], the average overhead for full scan is 84 % of additional area. A challenge with

scan-based approaches is that as devices increase in size and density, the time required to shift the entire FPGA state out proportionally increases [10].

(2) *Trace-based*: In a trace-based approach, a designer pre-inserts Trace-Buffers controlled by a trigger, also called embedded logic analyzers, into the circuit before compilation. The Trace-Buffers allow a window of the history of the chosen signals to be recorded as the circuit operates in real time. Trace-based debug has a drawback of requiring FPGA resources which can influence the placement and routing of the circuit [11] and limit the number of signals that can observe. The circuit may go through a time-consuming recompilation if the designer wishes to change the signals being observed or parameters of the Trace-Buffers or trigger. A number of automated techniques attempted to combat this [12].

2.2 Incremental Synthesis

Incremental synthesis allows us to overcome some drawbacks of the trace-based approaches. It is because instrumentation of incremental synthesis is not included in the compilation of the original circuit. The aim of incremental synthesis is to modify the functionality of a placed-and-routed circuit while preserving as much of the original solution as possible. FPGAs are well suited as there is often unused logic and routing resources leftover after a circuit has been complied. The main motivations behind this are (as shown in Fig. 1) to avoid a full compilation of the circuit when adding or making changes to the instrumentation, to preserve timing closure when undertaking engineering change orders, or for improved fault and defect resilience [13].

Graham et al. [14] demonstrated an approach which is more similar to this work; he put an unconnected embedded logic analysis in the FPGA prior to placing and routing, and afterward used a low-level bitstream modification to connect them to the desired signal [10]. However, their techniques relied on the Jbits API that was provided by Xilinx FPGA device, but similar APIs have not been for other

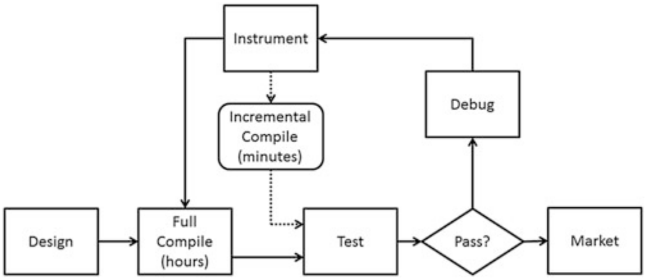

Fig. 1 Design and debug flow

commercial FPGAs. Graham system may not scale well for observing 1000s of nets, as only 128 nets were observed in their tests.

Commercial vendor tools also support the incremental compilation design. Altera's SignalTap II trace-IP solution and signal probe are specifically used to connect the I/O pins directly to the external analysis [4].

2.3 FPGA–ASIC Prototyping

FPGAs are integrated circuits fabricated to be configured by a designer after manufacturing. FPGAs are composed of reconfigurable logic, memory, and routing interconnects that can be configured to perform a huge variety of tasks. FPGAs can be used for one task and later be reprogrammed for a difficult task. They are different from application-specific integrated circuits which are typically hardwired for one task. However, the flexibility of FPGAs comes with some drawbacks. An ASIC consumes less area and power and operates faster than the equivalent circuit would on an FPGA. The specifics of FPGA architecture vary among vendors and product families. ASIC prototyping is the method to prototype ASIC design on FPGA for hardware verification. The need for verification of application-specific integrated circuit design is growing, due to the increased circuit complexity and time-to-market shrinking. Hardware platforms are becoming more prominent to test system designs at speed with on-chip bus clocks as compared to simulation clocks which may not provide an accurate reading of system behavior. These multi-gate designs usually are placed in multi-FPGA prototyping platform with six or more FPGAs, since they are unable to fit entirely onto a single FPGA. So the system RTL designs or netlist has to be partitioned onto each FPGA to be able to fit the design onto the prototyping platform.

3 Incremental Trace-Insertion

This section describes the proposed method for increasing FPGA observability to simply debug and verify. It describes how those components are incrementally inserted into the design. First, an overwrite of the proposed method that describes how it fits into Xilinx design flow is given. Then, the two major steps placement and routing are used for incrementally synthesizing the debug system.

3.1 Trace-Insertion

To increase the observability of FPGA circuits, a Trace-Buffer and a trigger unit are inserted incrementally into already placed-and-routed designs. Nets from the original circuit shall be incrementally connected to the Trace-Buffers to be observed and recorded. The trigger unit will control the Trace-Buffers to record until it meets the specified trigger condition.

The impact on the original circuit's area, placement, routing, and timing will be reduced when incrementally inserting the debug system. From the perspective of the original circuit, the debug systems have no area overhead. The original circuit will already be placed and routed and thus will already have claimed whatever area of the FPGA it needs. The debug system is inserted into whatever FPGA area has been left unused by the original circuit. Thus, the amount of Trace-Buffers and trigger logic to insert will be influenced by the area of the original circuit.

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 proposed work goal is different than this "general-purpose incremental synthesis" because its only desire is to observe signals. The placement and routing of the Trace-Buffers and trigger unit are restricted to resources unused by the existing circuit. The original circuit will be left completely intact.

3.2 Xilinx Design Flow

The proposed work is inserted between the place and route (PAR) and BitGen stages. When inserting instrumentation, the NCD representation of the circuit produced by the PAR process is converted to a XDL file. Trace-insertion modifies the XDL file to insert the Trace-Buffers and trigger unit and creates a new XDL that includes the modifications. This XDL file can be converted back to an XCD, and the normal Xilinx flow may continue. BitGen can create a bit file that can configure the FPGA with the circuit that includes the debug system. The ncd2xdl and xdl2ncd conversions are done (as shown in Fig. 2). A XDL file representing the circuit to instrument is one of the inputs to trace-insertion. Trace-insertion requires two lists: a list of the nets to trace and a list of the nets to connect to the trigger unit inputs. There are also other parameters of trace-insertion that can be adjusted such as Trace-Buffer width, trigger width, or number of trigger unit slices.

4 Primary Results

This section describes the methodology for testing the flexibility of the proposed incremental trace-based debug system. The runtimes of the incremental insertion on a set of benchmarks and the effects on minimum period are presented. The results of the tests are presented and demonstrated that the proposed system is feasible and can observe thousands of signals.

To demonstrate the feasibility of our proposed debug system, the effects of using it to trace up to 100 % of the flip-flops and latches in three benchmark circuits

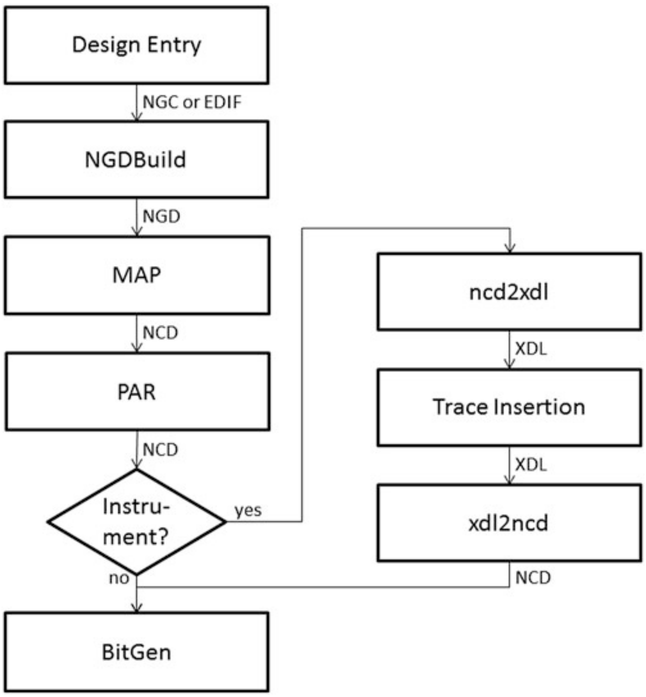

Fig. 2 Xilinx design flow

(when Trace-Buffer capacity permits). We chose to trace the flip-flops and latches because given the values of them the other intermediate values of a circuit could be calculated using techniques such as those in [15, 16].

The runtime is the time taken to place and route the Trace-Buffers and trigger unit. The time was determined by storing the current system time between major steps in the program and then comparing the times after routing has completed. The minimum period was determined by the Xilinx ISE tool, which analyses the timing of the FPGA circuit.

The important characteristics of the benchmarks are shown in Table 1. The benchmarks circuits were synthesized, and placed and routed with Xilinx ISE tools for the Virtex-2pro FG896 embedded development platform which contains a XC2VP30 FPGA. The benchmark circuits are available as part of the VTR project and represent realistic, sizable, heterogeneous designs that include Monte Carlo simulation for a financial application and linear system solvers. This table assumes a Trace-Buffer width of 16 is being used, so “Max-Traces” is calculated by multiplying the number of unused BRAMs by 16.

For the tests, there are several parameters are kept constant. The width of Trace-Buffers is fixed at 16, meaning 16 traces can be connected to each Trace-Buffer. At this width, the Trace-Buffers have a depth of 1024. The traces are routed in random order. The trigger unit is fixed at 256 inputs and 100 slices. The

Table 1 Uninstrumented benchmark summary

Circuit	LUTs	FFs	Slices	BRAMs	DSPs	lOBs	Signals traced	Max-Traces
BGM	20,511	3,672	11,708	0	44	289	3,672	4,870
Stereovision0	5,328	3,733	4,884	0	0	366	3,733	4,870
Stereovision1	9,299	9,078	7,222	0	88	278	9,078	4,870
Total available	27,392	27,392	13,696	136	136	416	–	4,870

trigger unit slices are placed in locations where the original circuits has not used any routing in the interconnect tile associated with the slices to avoid routing congestion.

The slices of the trigger unit are assumed to be placed within the same region. This ensures the trigger unit has good timing performance and its internal signals only have to be routed short distances.

4.1 Runtime Proportion

Figure 3 shows the total runtime for each of the benchmarks when the max numbers of Trace-Buffer inputs are routed. It should be noted that all benchmarks were able to successfully route all flip-flops. This is an interesting result and demonstrates that the Virtex-2pro architecture has enough routing resources to support our method. Most of the benchmarks take about a minute for the entire PAR process.

All the runtimes are less than the time it would take to recompile the entire circuit. Large and complicated circuits are known to take hours to recompile. The compile times of the benchmarks used in this work are shown in Table 2. The table also shows the average trace-insertion runtimes from Fig. 2 and the difference between the compile time and insertion runtime. None of compile times are in hours but even the shortest times are longer than all the runtime for trace-insertion, so incremental insertion decreases turnaround time for even these circuits also, if

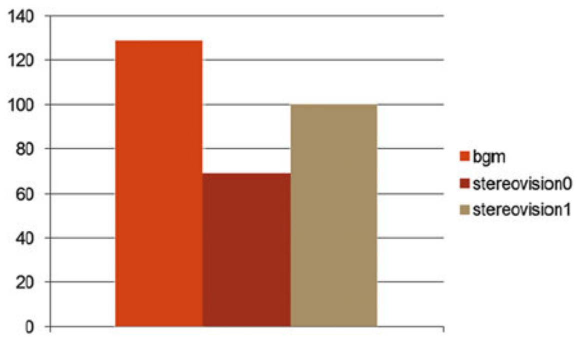

Fig. 3 Trace-insertion runtime

Table 2 Benchmark compile times in minutes and seconds

Circuit	BGM (mins:sec)	Stereo0	Stereo1
Compile time	08:55	05:10	11.06
Trace-insertion	02:05	01:25	01:43
Difference	10:50	03.45	10.23

incremental were not used then the compile time would be higher than the values shown in the Table 2 because the Xilinx tool would have to place and route the Trace-Buffer and trigger unit in additional to original circuit.

4.2 Minimum Period

The minimum period of each benchmark before and after instrumentation is shown in Fig. 4. The BGM benchmark circuit has a minimum period of 13.216 ns, and instrumentation increases this on an average by about 18 %. The stereovision benchmarks' minimum periods increase much more than other benchmarks. Stereovision0 jumps from the minimum period of 8.392–12.751 ns over 51 %. The percentage is even greater for stereovision1 which goes from 7.846 to 12.667 ns. The minimum period increases if the delay of any of the paths we insert is greater than the original minimum period. Circuits with the higher minimum are less likely to experience any increase because the inserted path may be longer without becoming a critical path.

The BGM sometimes had an increase in its minimum period and sometimes did not. The variation in period is due to the randomization in Trace-Buffer placement. The critical path is the control signal that is an output of the trigger unit and an input to the Trace-Buffers. The trigger unit is often placed near the edge of the FPGA because closer to center there are not enough free slices to place it. The

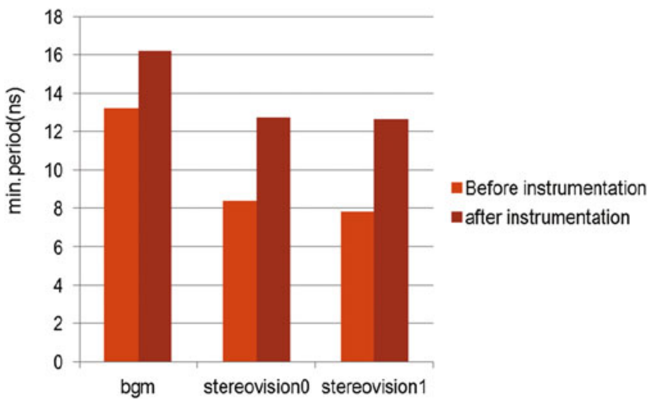

Fig. 4 Minimum period of each benchmark before and after instrumentation

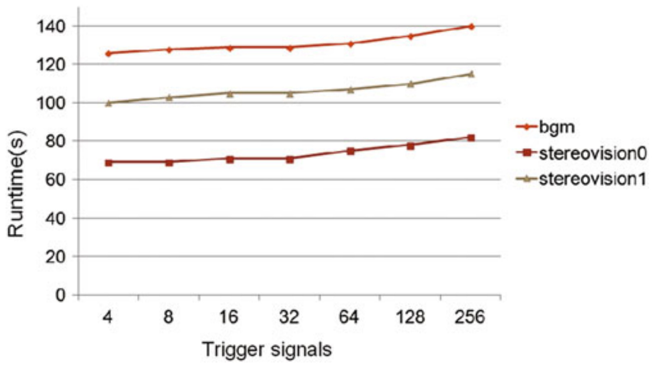

Fig. 5 Trigger signals influence on runtime

Trace-Buffers are spread throughout the FPGA. Thus, some of them are located long distance from the trigger units.

The critical path is the delay between the trigger unit and one of the distant Trace-Buffers. Figure 5 shows the runtimes for the different trigger widths. The runtime increases exponentially with the number of trigger signals, but it is mostly flat until the widths are 1024 or higher. The exponential increase comes because of the additional clock routing that is required with more slices.

Designer should avoid using trigger units that require thousands of input signals if runtime issue for the benchmarks used in our test runtime did not increase much if the runtime of input signals was 512 or below. Beyond that, the runtime increases quickly due to exponential curves. However, improving the clock router may eliminate the exponential increase in the scan time or at least keep the increase flat for even larger number of signals. Designers should also keep in mind the amount of the slices the original design uses. None of the benchmarks used here had a problem placing the trigger units of 100's of slices.

5 Conclusion

A new incremental trace-based method is used for increasing the observability of FPGAs. The method incrementally inserts the Trace-Buffers and the trigger unit in an already placed-and-routed FPGA circuits. A unique characteristic of the method is a centralized trigger unit that controls all the distributed Trace-Buffers. Advantages of this incremental method include not affecting the placing and routing of the user circuit taking full advantage of leftover BRAMs to observe more signals and decreasing turnaround time when changes are made to the debug system. The method could instrument 100 % of the flip-flops given that enough Trace-Buffer capacity exists. This was done on a commercial Xilinx Virtex-2pro FPGA further distinguishing from others. The time it takes to perform the method was less than

5 min for all benchmarks. This means that a designer could insert or change circuit instrumentation for debugging relatively fast. One drawback of our method is that it can increase minimum period for some circuits. This occurs if the delay of any of the parts we insert is greater than the current minimum period. So, the pipelining method can be used to improve the minimum period if needed.

References

1. Asaad S, Bellofatto R, Brezzo B, Haynes C, Haynes M, Kapul M, Parker B, Roewer T, Saha P, Takken T, Tierno J. A Cycle-accurate, cycle reproducible multi-FPGA system for accelerating multi-core process simulation. In: Proceedings of ACM/SIGDA international symposium field program gate arrays;2012. p. 153–62.
2. Kottolli A. The economics of structured-and standard-cell-asic designs. Open-Silicon: Technical Solutions Engineer; 2006.
3. Xilinx. Chipscope pro 12.3, software and core, user guide;2012. San Jose, CA, USA. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/chipscope_pro_sw_core_ug029.pdf.
4. Altera. Quartus II handbook version 11.1 vol. 3: Verification;2011. San Jose, CA, USA. http://www.altera.com/literature/hb/qts/qts_qii5v3.pdf.
5. Synopsys. Identify: Simulator-Like Visibility into Hardware Debug, Mountain View;2011. OR, USA. http://www.synopsys.com/Tools/Implementation/CapsuleModule/identify_ds.pdf.
6. Tektronix. Certus debug suite;2012. Beaverton, OR, USA. http://www.tek.com/sites/tek.com/files/media/media/resources/Certus-ASSIC-Prototyping-Debug-Solution-Datasheet_54W-28030-2_8.pdf.
7. Xilinx. Virtex-2pro fpga configuration user guide; 2012.
8. Hutchings BL, Torresen J. Unifying simulation and execution in a design environment for fpga systems. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2011;9(1):201–5.
9. Wheeler T, Graham P, Nelson BE, Hutchings B. Using design-level scan to improve fpga design observability and controllability for functional verification. In: Proceedings of 11th international conference on field-programmable logic and applications, FPL'01;2001. Springer-Verlag. p. 483–92.
10. Iskander YS, Patterson CD, Craven SD. Improved abstractions and turnaround time for fpga design validation and debug. In: 2011 international conference on field programmable logic and applications (FPL);2011. p. 518–23.
11. Hung E, Wilton SJE. Incremental Trace-Buffer insertion for fpga debug. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2014;22(4):850–63.
12. Hung E, Wilton SJE. On evaluating signal selection algorithms for post-silicon debug. In: Proceedings of international symposium quality electronic design, Mar 2011. p. 290–6.
13. Ko HF, Nicolici N. Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug. IEEE Trans Comput-Aided Design Integr Circ Syst. 2009;28(2):285–97.
14. Graham P, Nelson B, Hutchings B. Instrumenting bitstreams for debugging fpga circuits. In: The 9th annual IEEE symposium on field-programmable custom computing machines, FCCM'01;2001. p. 41–50.
15. Prabhakar S, Hsiao M. "Using non-trivial logic implications for Trace-Buffer-based silicon debug. In: In Asian test symposium 2009, ATS'09. p. 131–6.
16. Cheng W, Chuang C, Liu CJ. An efficient mechanism to provide full visibility for hardware debugging. In: Proceedings of 2006 IEEE international symposium on circuits and systems, Iscas 2006; 2006. p. 814.