Journal of Engineering and Applied Sciences 11 (12): 2643-2650, 2016
ISSN: 1816-949X

Integrating FPGAs with Trigger Circuitry Core System Insertions for Observability in Debugging Process

[1]A. Murali, [1]K. Hari Kishore and [2]D. Venkat Reddy
[1]Department of ECE, KLEF, K.L. University, Green Fields, Guntur, Andhra Pradesh, India
[2]Department of ECE, MGIT, JNT University, Hyderabad, Telangana, India

Abstract: To overcome the lack of observability in FPGA-based prototypes, trace-buffer insertion plays an important role in the design. But it also has a disadvantage of which it leads to the recompilation of the entire system. In this study, we introduce how the incremental techniques are used to discard the necessity of recompilation process on the circuit design and also we propose the CAD optimizations to improve the special features, routing capacity and minimizing the delay impacts. The use of these technique implementations in this circuitry, fastens the magnitudes than a full compilation. In this scenario, the incremental trace insertion is notable as higher as 98 times faster than a full compilation of the design and 25% of the memory capacity is used for tracing. The incremental circuits are more helpful for the designers only to modify by inserting the trigger circuitry rather than compiling the entire design.

Key words: Incremental trace insertion, Field Programmable Gate-Array (FPGA) prototypes, trace-buffer, routing capacity, incremental compilation or recompilation, memory capacity

INTRODUCTION

FPGA-based prototyping enables to evaluate the complex designs of the hardware in direct way with certain speeds faster than it simulates. This approach has a disadvantage of unclear reports while debugging. Running at high speeds on FPGA (Graham *et al*., 2001) prototype have many real and exhaustive tests with high functionality parameters. To address the observability in unexpected behavior, some commercial tools like Altera's Signal TapII, Xilinx's ChipScope (Pro 12.3 2010) and tektronix (Mentor Certus ASIC multi-FPGA prototyping debug suit 2012) are essential to provide the ability to insert the Trigger Circuitry core systems in the FPGA-based prototypes. Trigger circuits are required to start and/or stop recording based on the values of selected signals in the circuit at compile time. To change the trigger behavior, re-compiling is a necessary process as the cost increases for each debug iteration. After the core system is inserted, the whole design can be compiled and the conditions stop/start are recorded accordingly. If any bug is found or suspected while in this process, the conditions can be changed that has been recorded and Recompilation must to done at this point. Compile times are still long, even the vendors gains at reducing compile times in the past which are significantly reduce the productivity of the debug (Incremental trace buffer insertion for FPGA debug, Very Lager Scale Integration VLSI, IEEE Transactions (on vol. 22 no 4 pp. 850-863 2014) and finally Towards Simulator-like observability For FPGAs).

To overcome this, we introduced a method called incrementally implementing the trigger functions using the logic elements and connecting these Trigger functions to both the input signals and the memory arrays used as trace buffers. The trace buffers are used to record a small set of data. They have a limited capacity.

This technique reclaims the unused logic and routing resources of FPGA without replacing the original circuit. The major idea is to develop this process without modifying the user circuit which has the limited routing flexibility. CAD Optimization is also useful to increase the success rate of insertions.

In Fig. 1, shows the flow of the incremental trigger insertions in the FPGA-based prototypes. The incremental trigger insertions allows the designer to modify the system without making any changes to the original circuit. The technique that we use in this study is trace-support added to the end of the normal compilation flow of the FPGA hence, it requires small changes to make use of trace instrumentation. Figure 2 shows the before and after incremental techniques for the FPGA compilation flow. The approaches in the techniques are to the open-source VPR6.0 which is a part of the academi

Corresponding Author: A. Murali, Department of ECE, RAGHU Engineering College, Visakhapatnam, Andhra Pradesh, India

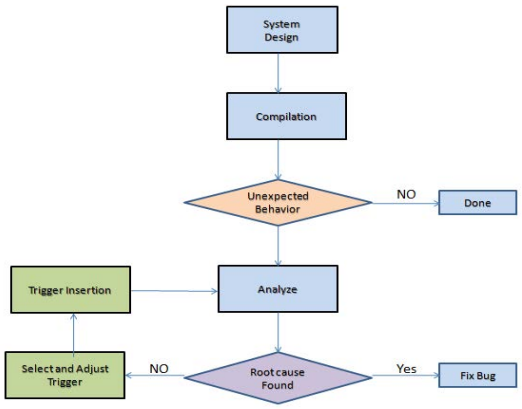

Fig. 1: Trigger insertion flow

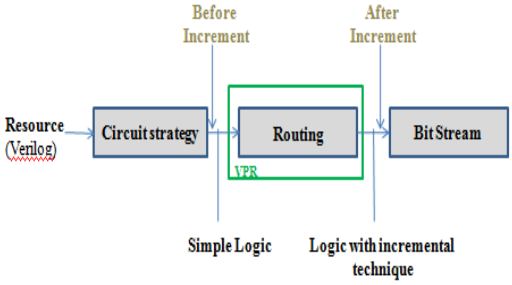

Fig. 2: FPGA (trace insertion) compilation flow in two levels(before and after incremental technique)

Verilog-To-Routing (VTR) with 1.0 version ("The VTR project: Architecture and CAD for FPGAs from verilog to routing" by Rose *et al.* (2012).

MATERIALS AND METHODS

Debugging procedure in circuitry: Trace buffers are on-chip RAM memories that store a history of a set of internal signals (referred to as trace signals) during at-speed device operation. Here, in the FPGA, the user circuit is implemented along with these Trace buffers and signals during the at-speed device operation. This is collectively known as the Debug Instrumentation. After compilation of the system, the internal signals which are referred as the Trace signals are then analyzed to determine the reason for unexpected behavior of the system. This method can trace only the small set of signals in a short period of time. Figure 3 shows a typical Debugging flow in FPGA.

It is essential to identify the events in the trace circuitry to make use of the trace buffer effectively. This circuitry uses to identify the recording signal stop at a point and is configures as a circular buffer as it continuously overwrites the stored data until the event occurs. The logic function is said to be the heart of the trigger circuitry as it indicates when the trigger should be

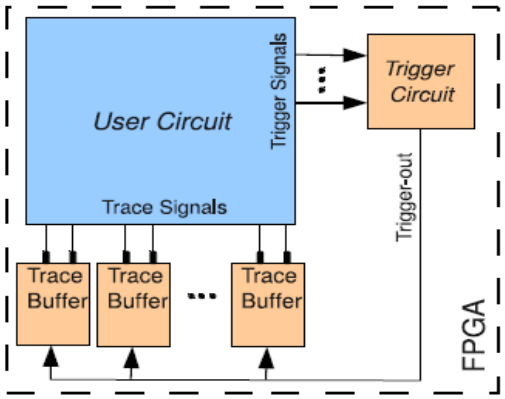

Fig. 3: Typical FPGA debugging flow of user circuit

occurred and relatively to the events in the user circuit. The circuit has mainly divided into three parts:

- Trigger function
- Inputs
- Single trigger output

In the normal traditional systems, usually modifications in trigger functions leads to recompilation of the entire system imposes to long compile times and also expensive. But in the method we introduced, the distribution of logic elements are desired to make up a trigger function across the logic elements which are left unused in the user circuitry. Using these logic elements in the three parts mentioned above overcome the routing failures. This contributes the distribution of the trigger logic over unused elements and to include the CAD optimizations in the trigger circuitry without recompiling the entire system.

Incremental distributed trigger insertion: In this section, we explain how the technique works in the distribution of the trigger logics to the Logic Elements (Les) in which the process is made in the routing phase. In LEs, there are Look-Up Tables (LUTs) and Registers. Let us assume that, the user selects a trigger function and its internal signals as the input without allowing to rip-up and re-routing the original circuit. This process of selection is known as the pack-place phase. Hence, it is crucial to consider the routability in trigger logic pack-place phase. There are two types of routing network connections. They are local network system and global networking system. and now, also assuming that FPGA has many Logic Clusters (LCs) and each of this have many LEs and are all connected to the local routing network. and LCs pins are connected to the global routing network. The process explained in three subsections.

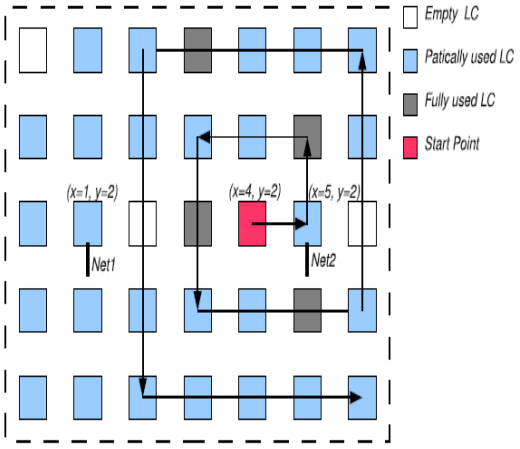

Fig. 4: Rotating placement strategy of finding the logic clusters

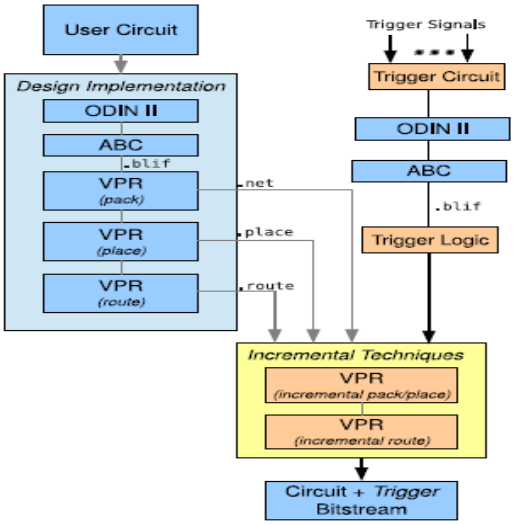

Fig. 5: CAD flow

Enhancing the routability process: Here, we show how selections are made and implementing the trigger logic is implemented with unused LEs. The main goal of this approach is to selecting the seed locations and positioning LEs in unused slots at the seed location. Firstly, selecting the seed location is mainly done at the closest point to the Logic cells since it impacts on the minimum wire-length and the time periods of the circuit. After insertion, the original circuit is fully mapped and routed, we can definitely come to know the positions and time periods of each input . Hence it is easy of finding the location for the seed by knowing the source location of each input net and weight these position by net's timings (with higher criticality and lower slack as in (V. Betz, Architecture and CAD for Deep-submicron FPGAs) (Rose *et al*., 1999). Criticality (Net1) = 0.1, Criticality (Net 2) = 0.3 Packer stars Packing from (x = 4, y = 2), three-times closer to Net2 than Net1. Packer will not consider fully used logic clusters.

Secondly, choosing the unused LEs utmost near to the seed location. For calculating the accurate results, the substitutions are in the form of an Algorithm 1. Figure 4 gives a clear view of this process. After choosing this start point in the trigger logic, the empty logic elements are then packed around this start point moving circular motion in forward direction as shown in Fig. 5. This rotating placement strategy keeps the logic elements in implementing the trigger logic as close together as possible.

Levels of congestion awareness in trigger pack-place: To avoid the routing failures, the routability in the pack-place phase is highly recommended. High usability of these routing resources may cause malfunctioning or failures in the trigger logic routing phase. Two-level congestion Awareness is added as a special phase to overcome the failures in the user circuit. They are LC-level congestion awareness and LE-level congestion awareness. This can be summarized in Algorithm 1.

Algorithm 1 : Two-level congestion aware pack-place algorithm

```
C = calculating starting point
while trigger_cells_to_place is not empty do
  if CL1(C) < threshold then
    T = select a logic element from
trigger_cells_to_place
    unused_LE_list = unused LEs in LC C
      foreach LE in unused_LE_list do
      if LE is not congested then
        pack T in LE of C
        remove T from trigger_cells_to_place
        if trigger_cells_to_place is empty then
          break
        T = select a logic element from trigger_cells_to_place
    C = go to next CLB
```

If the congestion value of LC is lower than the threshold, then the LEs may lead to have high congestions on their outputs. Due to this reason, these LEs are checked whether they are fully connected to the routing networks. Once if it finds the bugs like un congested LEs, pack-place inserts the trigger logic inside LE. To reach the goal of implementing the complete set of the logic functions, the algorithm goes on selecting the logic elements.

Incremental route-throughs in system: After the selection process is completed, these trigger inputs are routed in to each of their appropriate signals in the user circuit and the trigger outputs are routed to the memory.

To enhance the routability property, we use the unused LUTs as Route-throughs in the network.

RESULTS AND DISCUSSION

Evaluating the CAD optimizations:

Trigger insertion with incremental techniques in CAD flow: The known concept VPR of version 1.0. by Hung *et al.* (2013), The VTR Project FPGA 2012) which is the part of the academic scenario is used in the flow as shown in Fig. 5. Here, we proposed V6.0. of VPR to the design network.

The flow does not need any modifications to initial packing, placements and the routing of user circuit. Instead the trigger circuit is parsed and is converted to the technology-mapped netlist. and then, the process of packing and placing phases are performed by the algorithm. After the insertion process, the RAM memories are then claimed as the trace buffers. The final step is the direct search algorithm is then implemented for routing the trigger circuitry in VRP. In this method, we avoid to select the routing nodes that are fully occupied in the user circuit by modifying this algorithm for the best results.

Obtaining and validating the benchmark values by the logic circuitry system: The above process flow as described is then characterized by the logic cluster size as N = 10 and the look-up table size K = 6 and the cluster I/O flexibilities as $F_{c\text{-}In} = 0.15$, $F_{c\text{-}Out} = 0.10$ and channel segment length as L = 4. Table 1 and 2 shows the benchmark summary of the signals in the trigger circuitry.

For each of the benchmark, an FPGA size has chosen to be the smallest square that fits to the circuitry. The minimum channel width W_{min} (which has no debug instrumentation) for which the circuit is fully routed. The number of nets that are listed are from which the trigger inputs are chosen. The percentages of the Free LCs that are completely empty and the LEs shows the unused which are partially used in the user circuit. When running the VRP, the default feature (allow-unrelated-clustering) is disabled and is as like packing in a commercial Tool (Hung and Wilton, 2011, 2014).

The trigger function is a bitwise and that is between 32-1024 trigger signals which are randomly selected from the Global Nets. As for the multiple runs of each test is done, the different set of trigger signals are then chosen for each run.

Changes noticeable in the circuit by method implemetation: In this study, we explain the investigation process by using the methodology that is described in the previous study. Here, to know certain effects that are in the routing resource availability when the insertion is done, the W_{min} is then increased by some percentage, let us assume it to be as like W_{min}+20% as in Fig. 6. At first, the trigger circuits with various sizes (32-1024) are inserted into each benchmark. In this case all the

Table 1: Benchmark summary: all signals available to be used for trigger circuitry

	I/O		Logic cluster		DSP		RAM	
Circuit	Used	All	Used	All	Used	All	Used	All
Bgm	289	2400	4111	4200	11	162	0	120
LU8PEEng	216	1962	2667	2745	8	120	45	80
LU32PEEng	216	3552	9105	9213	32	378	150	252
Mcml	70	3200	7350	7400	30	325	38	208
mkDelayWorker32B	1064	1344	916	1302	0	50	41	42
mk PKtMerge	467	832	18	494	0	18	15	16
or 1200	779	800	288	475	1	18	2	12
Raygentop	544	640	277	280	9	15	1	9
Stereovision0	354	1312	1240	1271	0	50	0	30
Stereovision1	278	1632	1889	1938	60	72	0	56
Stereovision2	331	2848	5889	5963	213	242	0	154

Table 2: Signal percentages available for trigger circuitry

Circuit	6-Input LUTs	FFs	W_{min}	Global nets	Free LCs (%)	Free LEs (%)
Bgm	32384	5362	80	23476	2	25
LU8PEEng	22632	6630	92	16643	3	18
LU32PEEng	76211	20898	128	55873	1	17
Mcml	101858	53736	86	52226	1	9
mkDelayWorker32B	5590	2491	76	4686	30	47
mk PKtMerge	232	36	44	515	96	22
or 1200	3054	691	74	2602	39	11
Raygentop	2148	1423	60	2126	1	35
Stereovision0	11460	13405	52	8358	2	31
Stereovision1	10290	11789	90	11368	3	52
Stereovision2	29943	18416	92	35386	1	57

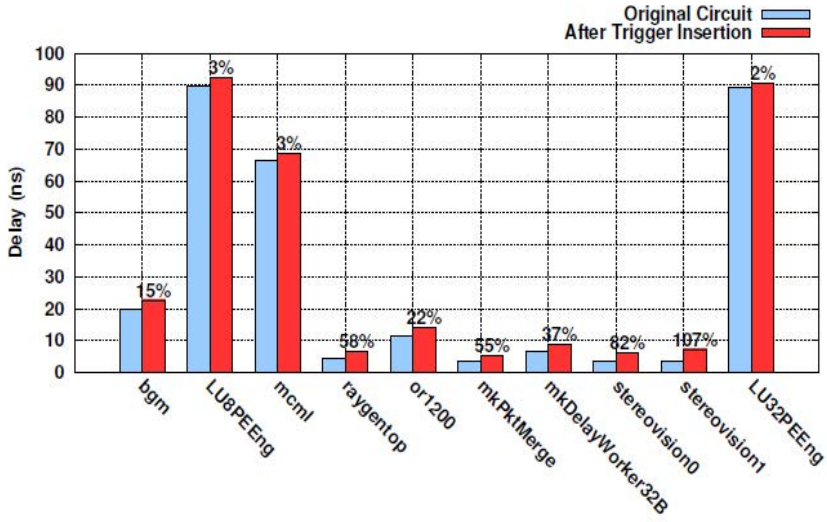

Fig. 6: Circuit path delay increases after trigger insertion for W_{min}+20%

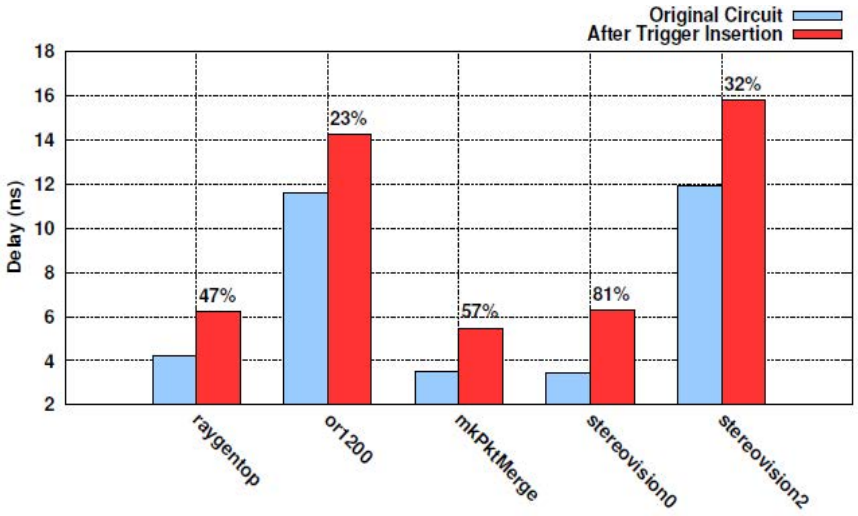

Fig. 7: Circuit path delay increases after trigger insertion for W_{min}+30%

input triggers functions (32-1024) are successfully inserted for all benchmarks with some expectations. The circuits (LU8PEEng, LU32PEEng and mcml) have critical path delays (90, 90 and 67 ns, respectively) and inserting the trigger logic increases the delay (about 3, 2 and 2%, respectively). These incremental delays are not that notable and are temporarily exist during the debugging process. The debugging instrumentation is not required as the operating frequency of the circuit can be set as normal.

As because of the high congestion and routing more than estimated all free memories (154), the insertion process here is utterly failed. Even the circuit stereovision1 has a critical path delay as only 3.5 ns and the trigger logic increases its delay on average by 107% which is temporary aspect. But in other circuits as stereo vision 0 stereo vision 2, mkPktMerge and raygentop are failed since all the benchmarks are too small to support such large triggering functions and also inserting the triggering functions with 1024 size failed for 0r1200 for the same reason. To determine the working of our technique in the circuit at less routing congestion situations, here again we repeat the same process with increase of minimum channel Width as W_{min}+30%. To differentiate these values, ure Fig. 7 gives a complete reference of the percentages of each circuits with the original circuit and the trigger inserted circuit.

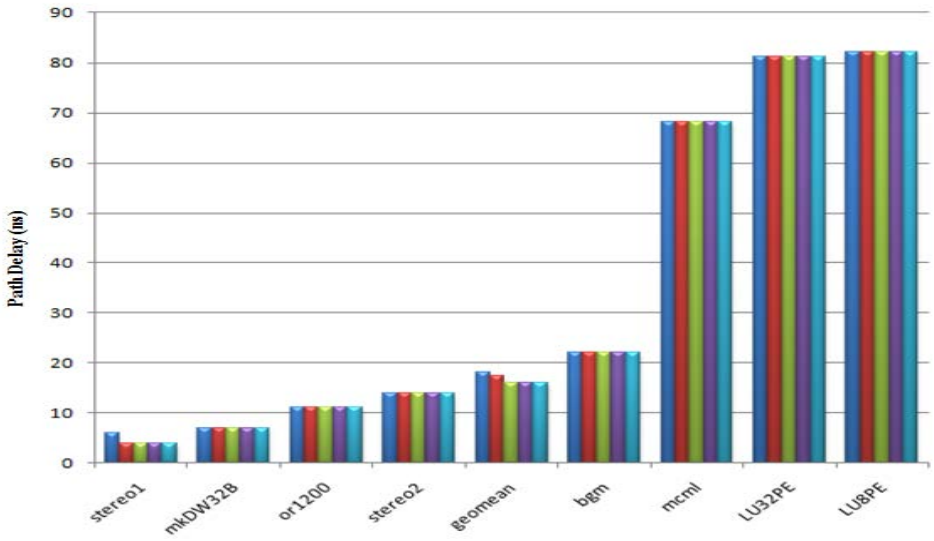

Fig. 8: Comparing the circuit path delays for minimum channel Width (W_{min}) at trace demand 0.75

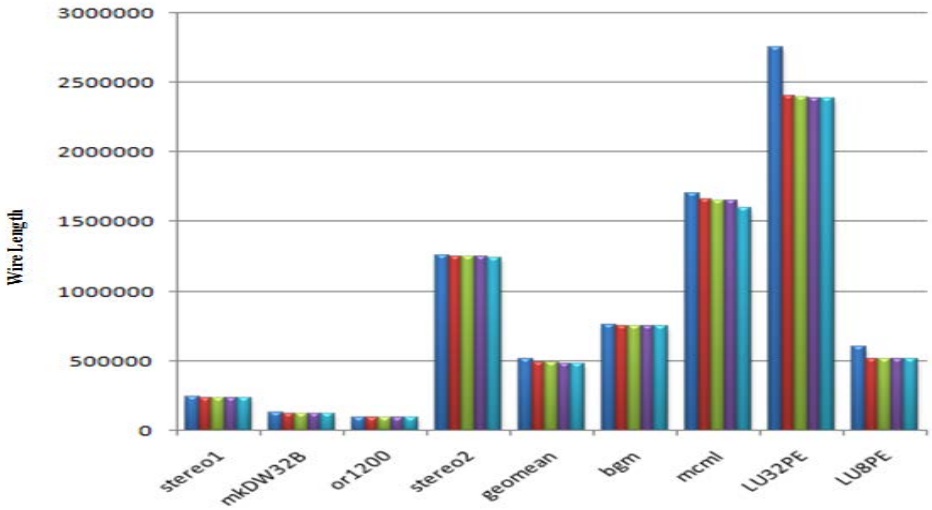

Fig. 9: Total wire length required for each of the circuit with minimum channel width at trace demand 0.75

It is experimentally found that the congestion threshold of 60% for trigger functions with sizes of 32- 512 is sufficient for successfully insertions. For sizes of 1024, 50% of threshold is highly necessary to use in the circuit. The impact on run-time is also examined, and is mentioned clearly in Fig. 8. At average, incremental-distributed trigger insertion is 80X faster than full recompilation.

As we fix the minimum channel lengths to the increments (0, 20, 30, 40 and 50%), it represents that the implementation of the circuit is done by utilizing the number of FPGA routing resources. The value of the trace demand at some point be constant (say 0.75), requires the trace wires of shorter length as shown in Fig. 9. For the higher rates of trace demand values (say 0.5 and lesser), they have the shorter trace-wire length and even at maximum demand value (say 1.0) it is still within 2% after incremental technique. From Fig. 10, it is clearly shown that as the channel width increases, the instrumental wire length decreases.

Despite of increase in wire length observed in the above result, the difference in the path delay is less effective on each circuit. The experimental results shows the trace insertion is higher at 0.6% when compared with the original circuit. Figure 10 shows the impact of trace insertion has great impact on the circuit with various channel width values (Fig. 11).

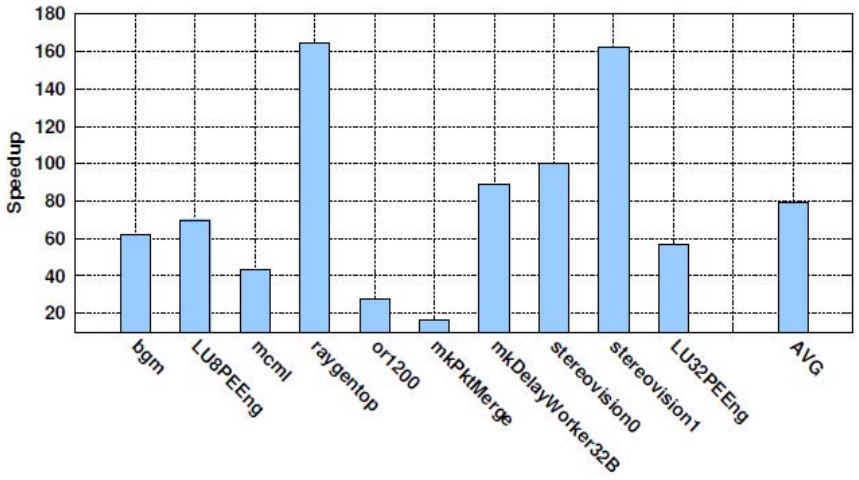

Fig. 10: Comparing the trigger insertion runtime with recompilation process

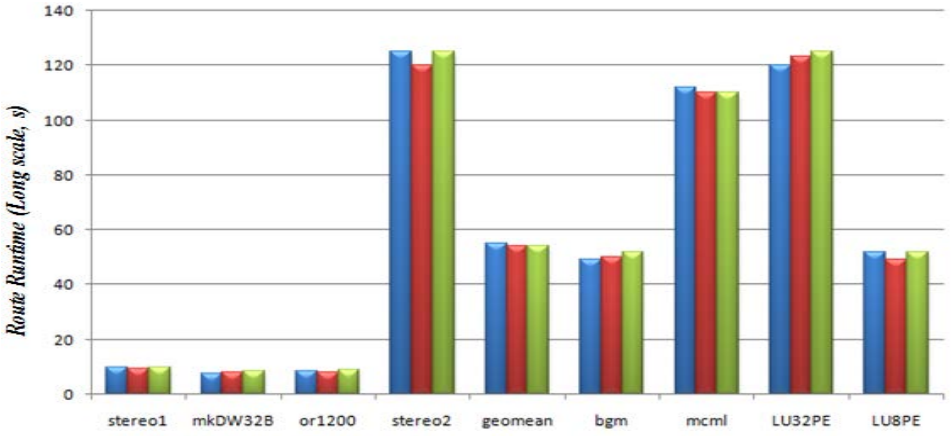

Fig. 11: Route runtime (long scale, with trace demand of 0.75 varies for all circuits with different W_{min} values

Features that made a difference in the circuit: The features like wire length, delay path, route runtime vary at different values for all circuits at minimum and maximum trace demand. Also logic utilization, memory utilization and the maximum operating frequencies differ for certain value. Especially reclaiming the 100% memory utilization is noticed for tracing and the memory blocks shows flexibility even at trace demand <1.0.

CONCLUSION

In this research, incremental compilation techniques are mainly used to insert the trigger circuitry into the design without requiring a full recompilation process. We have undergone the successful results by using these special techniques. Insertions with a short critical path (below 18 ns) increases the path delay.

REFERENCES

Graham, P., B. Nelson and B. Hutchings, 2001. Instrumenting bitstreams for debugging FPGA circuits. Proceeding of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, March 29-April 2, 2001, IEEE, Brigham, Utah, ISBN:0-7695-2667-5, pp: 41-50.

Hung, E. and S.J. Wilton, 2011. Speculative debug insertion for FPGAs. Proceeding of the 2011 21st International Conference on Field Programmable Logic and Applications, September 5-7, 2011, IEEE, Vancouver, BC, ISBN:978-1-4577-1484-9, pp: 524-531.

Hung, E. and S.J. Wilton, 2014. Incremental trace-buffer insertion for FPGA debug. IEEE. Trans. Very Large Scale Integr. Syst., 22: 850-863.

Hung, E., F. Eslami and S.J. Wilton, 2013. Escaping the academic sandbox: Realizing VPR circuits on Xilinx devices. Proceeding of the 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, April 28-30, 2013, IEEE, Vancouver, Canada, ISBN:978-0-7695-4969-9, pp: 45-52.

Rose, J., J. Luu, C.W. Yu, O. Densmore and J. Goeders *et al.*, 2012. The VTR project: Architecture and CAD for FPGAs from verilog to routing. Proceedings of the ACM-SIGDA International Symposium on Field Programmable Gate Arrays, February 22-24, 2012, ACM, California, USA., ISBN:978-1-4503-1155-7, pp: 77-86.

Rose, J., V. Betz and A. Marquardt, 1999. Architecture and Cad for Deep-Submicron Fpgas. Kluwer Academic Publishers, Norwell, Massachusetts,.