100% Visibility At MHz Speed: Efficient Soft Scan-Chain Insertion on AMD/Xilinx FPGAs

Hossein Omidian, Eddie Hung, and Dinesh Gaitonde

AMD, San Jose, USA

Abstract. FPGA-based prototyping has become an increasingly important part of the overall integrated circuit design and verification flow, providing the ability to test an integrated circuit running at (near) speed with realistic inputs and outputs. The reconfigurable aspect of FPGA technology makes them suitable for hardware emulation and prototyping, plus their nature of having over-provisioned resources — inherently necessary to support the late-binding of a wide range of applications — allows support for 'out-of-band' functionality such as debug. It is imperative that as much visibility into the inner state of the circuit is accessible in order for debugging to be effective. Full visibility for functional debug can be achieved by building a soft scan-chain out of LUTs and flip-flops, or by using hardened device readback capabilities that use the configuration network to exfiltrate circuit state. In this paper, we show how soft scan-chains can be efficiently and intelligently inserted to give 100% visibility into all user flip-flops of a design and demonstrate how performing parallel scan dumps can be more than 10x faster (reaching 1 MHz) than hardened readback when evaluated on industrial emulation designs in excess of 200K flip-flops.

Keywords: Emulation · Prototyping · Debug · Scan Chain · Readback

1 Introduction

Since the cost of fabricating a custom ASIC is so time-consuming and expensive after which changes (for example, to fix a design error or to insert some debug infrastructure) are not always possible, reconfigurable technology such as FPGA is widely used in this area. FPGAs are inherently flexible devices that are composed of programmable logic cells, memory and interconnect. This allows them to be customized and used in a broad range of applications including ASIC prototyping and hardware emulation [1].

A problem common to ASIC and FPGA technology is the lack of on-chip visibility for diagnosing erroneous behaviour. In ASICs, such errors can be caused by (a) fabrication defects or (b) functional bugs. Fabrication defects are caused by the imperfect nature of silicon fabrication process whereby, for example, a metal wire or a transistor is randomly manufactured incorrectly. To identify these cases, ASICs often employ 'scan flops' in place of regular flip-flops to enable manufacturing tests. A scan-flop behaves just as a regular flip-flop but with the

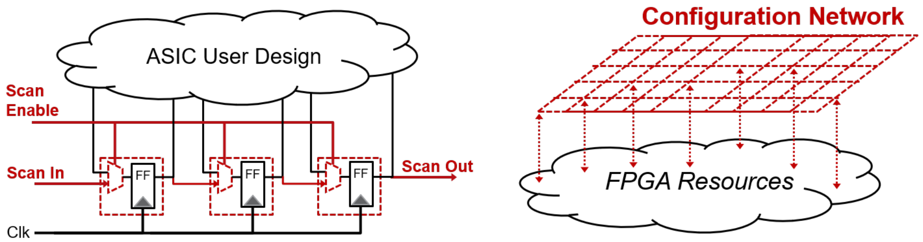


Fig. 1: *(left)* ASIC scan flops (scan-mux and regular flop) arranged into a chain. *(right)* Hardened FPGA config. network allowing both write and read back.

optional capability (achieved using a scan mux) that new values can instead be sequentially shifted in and existing values shifted out when arranged into a scan-chain as shown in Figure 1. The utility of a scan-chain is that post-fabrication, a known test pattern can be shifted into all flip-flops of a design, the clock advanced to capture the next state computed by the device, and then this newly captured state can be shifted out and compared with a known golden value. Deviations from this golden value would indicate a manufacturing failure.

After manufacturing tests, these same ASIC scan-chains can be reused to investigate *functional bugs* by following the same shift out method: halt the design at the clock cycle of interest and proceed to shift out all values on the scan-chain to gain a complete picture of all design state to aid debugging.

In contrast, even though FPGAs may suffer from the same issue of fabrication defects, their reconfigurable nature provide alternate ways to perform manufacturing tests without the overhead of hardened scan-chains as for ASICs. An unavoidable overhead that FPGAs do have to pay, however, is that of a configuration network. The purpose of this network is to transport all of the configuration necessary to implement a user design, such as all LUT contents, flip-flop initialization values, interconnect switch states, etc. to all locations of the device. Some FPGA vendors, such as Xilinx, allow this same configuration infrastructure to be re-purposed as a method of extracting user-state to aid in the investigation of functional bugs, in the same way as ASIC scan-chains. We refer to this FPGA capability as 'hardened readback'.

Hardened readback for functional debug shares the same limitations as for ASIC scan-chains: the design must be halted during the shift out procedure, for a length of time proportional to that necessary to perform configuration readback of all used flip-flop resources in the design or to unload the longest ASIC scan-chain. In this work, we show how the use of multiple soft scan-chains (i.e. scan-chains created out of regular LUT logic) can be used to dramatically reduce this overhead for functional debug; our main contributions are:

1. An approach for the efficient insertion of multiple soft scan-chains capable of acquiring 100% visibility into all flip-flops of a user design while still allowing such designs to continue operating in excess of 1 MHz while a typical emulation design operates between 1 to 10 KHz.

2. Integration of our techniques into a production quality and fully timing-driven commercial FPGA toolflow, one mindful of real-world considerations such as hold time requirements, clock skew, etc.
3. Robust evaluation on 29 industrial emulation designs containing multiple clock domains and more than 200,000 user flops, finding a 10x speedup over a hardened readback solution.

The remainder of the paper is organized as follows: Section 2 explains background and related studies. Section 3 describes the proposed approach of inserting soft scan-chains into a user's design. Section 4 provides experimental results and comparison of our approach with hardened readback. Finally, Section 5 concludes the paper.

1.1 Related Work

The novelty of our work is not in using soft-logic to implement scan functionality. Prior work from Wheeler et al. [2] examined the application of ASIC-style scan-flops (as per Fig. 1) to *replace* existing flip-flops (as opposed to our proposal of *shadowing* existing flops) to allow design state to be both observed as well as modified (where our shadow approach is unable to modify). The cost of this prior work is a reported 20% reduction in Fmax during normal operation, a 2.3x increase in LUT count, and the need to halt the design during the scan out procedure. Wheeler et al. state that is an acceptable overhead during development since this handicap is removed for the final production design. In contrast, our approach focuses on this development phase and we show that there is no Fmax degradation when scan functionality is not used, a temporary Fmax slowdown during scan-out, and no effect on the size of FPGA required, when evaluated on industrial designs from the emulation domain.

Work from Tiwari and Tomko [3] explores the use of soft scan-chains to implement software-like ”watch-point” functionality to detect when specific values appear on predetermined internal signals, after which the clock can be halted and the state of the design examined. Here, scan-chain functionality is used to update watch-point values efficiently. However, both prior works [2, 3] do consider using FPGA device readback to reduce the area overhead and for providing observability respectively, recognizing as we do, that readback is a slow operation.

2 Background

When a hardware design does not behave as expected, debugging is required to find the root cause of this erroneous behaviour. Key to the effectiveness of the functional debugging process is the visibility that the designer has into the internal signals of their circuit. Using software simulation, unlimited visibility is available but the speed at which large complex designs can be simulated is often many orders of magnitude slower than their target frequency. With real silicon, this frequency gap is significantly narrowed on FPGAs and may even be eliminated on ASICs, but the tradeoff is that visibility becomes severely limited. To overcome this limited visibility, designers must repurpose existing or insert new infrastructure to expose internal signal activity. There are two main categories for visibility infrastructure: scan-based and trace-based.

As described in Section 1, ASICs are often built with scan-chain capabilities to test that the silicon was manufactured correctly. Post manufacturing test, such functionality can be repurposed for debug. As long as the design can be halted at precisely the clock cycle of interest, by unloading all values of the scan-chain a designer can determine of all flip-flops in the design (and consequently, all intermediate combinatorial signals too). Equivalently, the same concept can be applied to FPGAs that support a hardened readback capability once the design is halted, its configuration network can instead be repurposed to read/scan out all flip-flop state. The design can then be advanced to the next state by single-stepping the clock, and further scan dumps performed to understand how the design evolves over time.

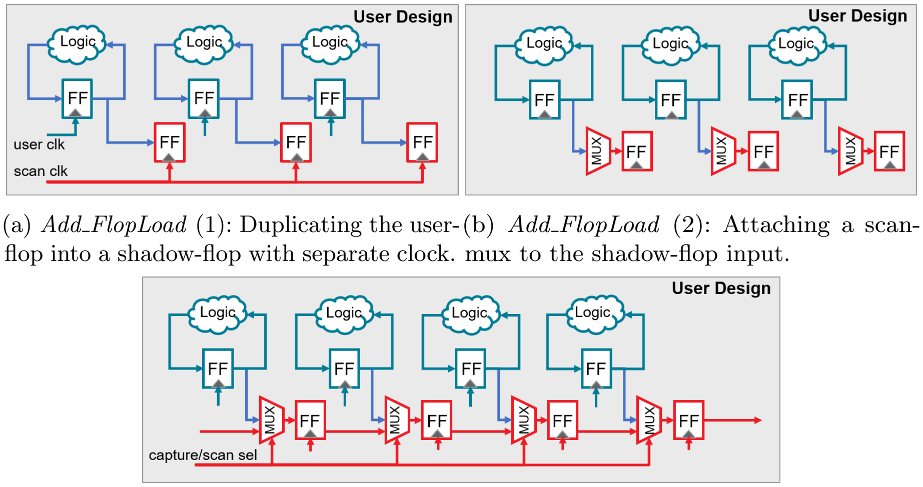
The disadvantage of a traditional scan-based approach is the time required to dump its contents. For FPGA technology, the max frequency of a user design with full readback $Fmax_{D-with-RB}$ is proportional to the number of flip-flop values that need to be dumped NF_{user}, the efficiency of hardware readback Eff, as well as output bandwidth of the configuration controller (on Xilinx devices, this is referred to as ICAP [4], $BW_{(ICAP)}$):

$$Fmax_{D-with-RB} \propto \mathit{Eff} \cdot \frac{BW_{ICAP}}{NF_{user}} \tag{1}$$

Eff here is a $(0, 1]$ scaling factor that reflects the overhead of using hardened readback. In Xilinx UltraScale+ devices, the configuration network operates at a frame granularity where each frame contains 2,976 bits of configuration data that must be atomically written or read [5]. Using hardened readback to extract the value of just one flip-flop value requires the entire frame to be read back, leading to an efficiency of 0.00034. The efficiency is improved, up to a limit, when multiple user flops that happen to be placed into the same frame are read back.

Trace-based approaches require the insertion of trace buffers and supporting logic to non-intrusively record a small subset of signal activity into on-chip memory [6]. The advantage of this method is that a design-under-trace need not to be halted in order to gain visibility, as well as being able to capture behaviour of the circuit over time without single stepping the clock. However, the disadvantages of trace infrastructure is that it does occupy precious on-chip memory and logic resources which can limit the amount of information that can be traced — both in terms of the number of signals that can be traced in parallel (corresponding to the width of the trace memory) as well as how many cycles of history can be captured (the depth of the trace memory).

Recent work by Attia and Betz [7] has demonstrated a compelling need to export the entire state of the design — that stored in user flip-flops as well as RAM contents — so that a faulty subset of the design already executing in an



(a) *Add_FlopLoad* (1): Duplicating the user-flop into a shadow-flop with separate clock.

(b) *Add_FlopLoad* (2): Attaching a scan-mux to the shadow-flop input.

(c) *Add_ScanChain*: Forming a chain of scan-mux to shadow-flop connections.

Fig. 2: Two-step scan-chain insertion: *Add_FlopLoad* followed by *Add_ScanChain*.

FPGA can be migrated into the much-slower (but more familiar and productive) software simulator to continue debug. To achieve this, they use Xilinx's hardened readback [4] capabilities, making it closer to the scan-based approach than a trace-based one.

In this work, we propose that a soft scan-chain approach be used to overcome the performance penalty incurred by continuously applying a hardened readback solution, yet without restricting the visibility available to a designer as would be necessary with a trace-based approach.

3 Soft Scan-Chain Methodology

The implementation and requirements of a soft FPGA scan-chain are very different to those in ASICs. Firstly, ASIC scan flops are typically highly optimized macro cells that can be used as drop-in replacements to regular flop cells with only a small area impact. For an FPGA, it is not practical to make all customers pay this area cost (along with even more area to expose the additional pins to the routing network) for a feature that many would not need, especially since manufacturing test is a FPGA vendor responsibility. Secondly, since an FPGA scan-chain is not used for manufacturing test there also does not exist the requirement to load new values into user-flops.

***Add_FlopLoad*:** Instead, FPGAs can use soft logic resources — lookup tables — to implement the 2:1 scan multiplexer functionality. Rather than add an extra 3-input LUT to every path leading into a flip-flop, we propose that each flop in the user design (henceforth referred to as a *user* flop) be replicated into a *shadow* flop, as shown in Fig 2a. Importantly, this shadow flop must be sensitive to a different clock than that used by the user flop for reasons explained in the following paragraph. A scan-mux can then be attached in front of the shadow flop, as per Fig 2b. Since user-flop controllability is not a necessity in FPGAs, along with the over-provisioning of flip-flop resources on FPGAs, a shadow flop is suitable here. Furthermore, adding an extra fanout to the output of the user flop, as opposed to adding extra logic to its input, also minimizes the impact on compilation quality and runtime. Both the shadow-flop and scan-mux insertion is accomplished in the *Add_FlopLoad* stage of our flow.

After capturing a design's state into shadow flops, all those captured values need to be stored or exported at every user clock cycle so that it may be analyzed or post processed. This can be done connecting the shadow flops into a serial chain (Fig. 2c) similar to a shift register; once the user clock is halted, advancing the scan clock will cause its contents to be dumped out one value at a time. Attaching the shadow flops to an separate scan clock independent from the user design is both necessary so that the scan-chain can be dumped without interfering the user design, and also beneficial since the scan dump procedure can also be safely operated at a higher frequency.

Figure 3a. shows a design with 6 flip-flops $FF_1, FF_2, ..., FF_6$ with values $D_1, D_2, ..., D_6$. After each user design clock cycle, the *Capturing_Value* process starts by saving each user flip-flop's value into their respective shadow flops. This step is done by selecting the top input of all scan-muxes. The *Capturing_Values* step was also shown as "Read" in the Figure 3a. waveform. After capturing values into the shadow flops, we move to the *Scan_Dump* mode to send the values out serially.

In the *Scan_Dump* mode, the bottom input for all scan-muxes are selected to enable shift out functionality[1]. As one can see in the waveform from Figure 3a. when scan functionality is desired the user clock period needs to exceed the time necessary to perform a scan dump when operating the scan clock at a different, faster period. Hence, the maximum frequency of a design with continuous scan dumps (Fmax) will be always dependent on the Fmax of the scan-chain as well as the number of scan flops that need to be unloaded.

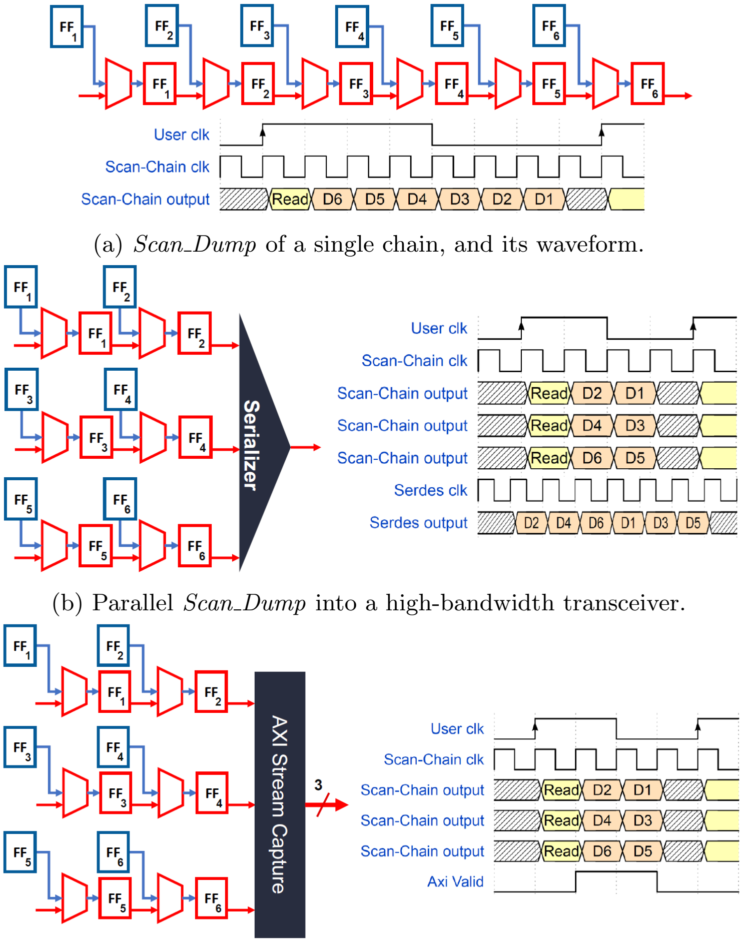
With NF_{scan} as the number of scan flops on the scan-chain (which in this work is equivalent to the number of user flops NF_{user}) Equation 2 shows the relationship between the Fmax of the slowed user design ($Fmax_{D\_with\_SC}$) and the scan-chain's Fmax ($Fmax_{scan}$).

$$Fmax_{D\_with\_SC} = \frac{Fmax_{scan}}{NF_{scan} + 2} \tag{2}$$

The +2 factor in the denominator represents a cycle to first read (capture) the user flop values into the shadow flops, and a cycle at the end to export the last value in the chain.

To improve the user-design-with-scan Fmax ($Fmax_{D\_with\_SC}$), it is possible to have more than one scan-chain and read out multiple in parallel. In other

[1] *Scan_Dump* can be done every cycle or once in while. For this study we focus on capturing and reading back flops every cycle since it covers both cases.



(a) *Scan_Dump* of a single chain, and its waveform.

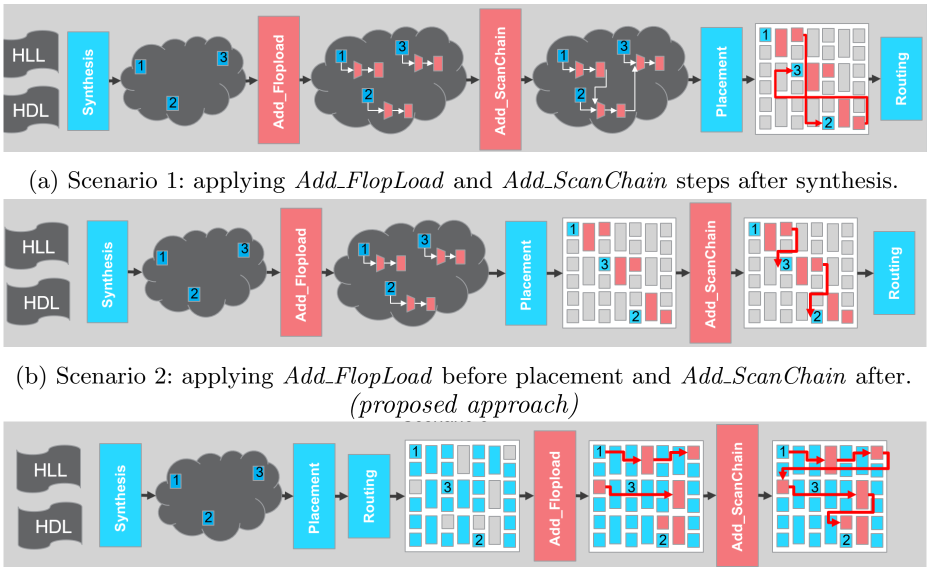
(b) Parallel *Scan_Dump* into a high-bandwidth transceiver.

(c) Parallel Multiple *Scan_Dump* into an AXI stream.

Fig. 3: Single and multiple/parallel *Scan_Dump* in which the scan clock operates at a multiple of the user clock.

words, we divide the set of all shadow flops into different scan-chains. Having more than one scan-chain leads us to have less number of flops in each chain (NF_{scan}) which leads to a higher $Fmax_{D\_with\_SC}$. Figures 3b. and 3c. have three scan-chains instead of one, with each containing 1/3 of all flops. This means we can shift out all scan flop values in only 2 cycles instead of 6 cycles, and $Fmax$ can be increased almost 3x.

When To Insert Scan-Chains: In our flow, the *Add_FlopLoad* step is followed by *Add_ScanChain*. The method taken by this latter step depends on when in the compilation flow it is applied, which we shall discuss first. Each of these previous steps can be applied to the user design at different stages of the flow: synthesis, placement or routing as illustrated in Figure 4.



(a) Scenario 1: applying *Add_FlopLoad* and *Add_ScanChain* steps after synthesis.

(b) Scenario 2: applying *Add_FlopLoad* before placement and *Add_ScanChain* after. *(proposed approach)*

(c) Scenario 3: applying *Add_FlopLoad*, *Add_ScanChain* after both placement and routing.

Fig. 4: Scenarios 1–3: scan application at various points of the compilation flow.

Scenario 1: apply both *Add_FlopLoad* and *Add_ScanChain* steps after synthesis and before placement. In this scenario, the place and route tool will have maximum flexibility to find optimum overall placement of the combined design (user design and the scan-chain). Since the placement algorithm places the design considering its routability, finding a performant routing will be more likely. For example, in case of possible routing congestion, the placer might spread out the design throughout the chip to ensure the router can find a high quality routing solution. On the other hand, there are two shortcomings in this approach. First, adding the full scan-chain to the user design before placement will bias the placer to give the user design and the scan-chain equal priority, which may lead to a subpar placement for the user logic. Second, the placer is given exactly one scan-chain ordering, with zero flexibility, which can also lead to a subpar placement for one or more scan-chain connections thus lowering $Fmax_{scan}$ and affecting overall system performance.

Scenario 2: applying *Add_FlopLoad* before placement, letting the tool place the design, and then applying *Add_ScanChain* to the placed result. In this scenario since all the shadow flops are only connected to user flops, the placer is going to place the design without being significantly affected by any scan-chain connections. The placer algorithm will simply see the shadow flops and scan muxes as floating logic attached only to the user flop's output, thus place the user flop as it would do normally and then place the shadow flop at a nearby location. Since *Add_ScanChain* is done after placement, the exact location of each shadow flop is known and this information can be used to find an efficient scan-chain order that minimizes the routing distance between shadow flops.

Scenario 3: performing *Add_FlopLoad* and *Add_ScanChain* both after placement/before routing, into just the FPGA resources left unused by the design. This scenario comes handy when the user design was anchored or floor-planned with a specific criteria. Adding the scan-chain after the placement technically doesn't affect user design's placement and will try to add scan logic into any unused resources left behind. However, finding unused LUT and compatible flip-flop resources near to user flops is far from guaranteed.

Experimentally, we have found that Scenario 2 performs best and is the focus for the remainder of this paper.

***Add_ScanChain*:** For Scenarios 2 and 3, *Add_ScanChain* is to be applied post placement. The main goal of this step is to maximize $Fmax_{scan}$ by reducing the total wirelength and worst-case delay of all shadow-flop to scan-mux paths across all connections within and between all scan-chains. Given a placed result where all scan-mux and shadow-flop locations are known, the problem is almost exactly that of the travelling salesman — starting at any shadow flop, determine the order in which all other scan-mux/shadow-flops are to be visited before finishing at a particular input pin, with no flop visited more than once and with the objective of minimizing the total travelled distance (equivalent to routed wirelength, minimizing which will improve the likelihood of finding a legal routing solution). An additional objective on top of the travelling salesman problem is to also minimize the maximum distance between any two flops, as that determines $Fmax_{scan}$.

Despite the (NP) difficulty of optimally solving the travelling salesman variant, experimentally we have found that a simple greedy heuristic was sufficient to achieve high performance. Starting from top-left of the chip, go down and find the nearest shadow flop and connect that to the scan-chain repeating until we hit the bottom of the chip. Then move right by one column and this time move to the top of the chip continuing to connect shadow flops in this way. This zig-zag move continues until all shadow flops are visited. Figure 5 shows an example of adding one scan-chain to 1% of the flops in a design. We picked 1% of flops randomly through out the design; 1% simply to make the figure clearer. The scan-chain is shown in purple color.

Similar to top-down approach, a left-right approach was also implemented. Experimentally, we observed that a top-down approach had slightly better results compared to left-right approach. We believe that top-down is more suitable for columnar FPGA architectures such as those from Xilinx.

Partitioning: So far we have talked about the two main steps for adding scan-chains to a user design and the different scenarios for when to do so. We also talked about the benefits of having more than one scan-chain In the following, we

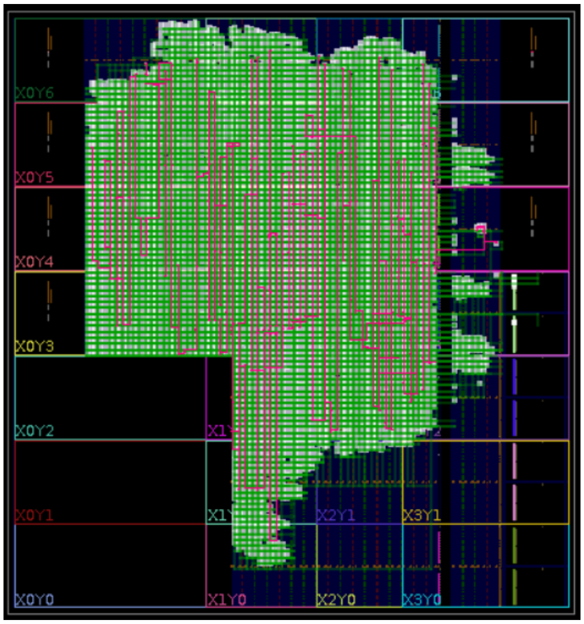


Fig. 5: Device view visualizing the connections made by one scan-chain visiting just 1% of the shadow flops in the design, using the top-down approach.

will explain different ways to break a single scan-chain into multiple and explain the tradeoffs in doing so.

As discussed, one dimension that can improve $Fmax_{D\_with\_SC}$ is by increasing the number of scan-chains, thus decreasing the amount of time required to dump their values (in parallel). Partitioning techniques can be used to cut the design into smaller partitions and assign a scan-chain for each partition. Partitioning can be define based on different parameters. One way of partitioning a design is considering the FPGA architecture and partition based off that. For example, Xilinx's latest FPGA devices use Stacked Silicon Interconnect (SSI) technology, which creates high-capacity FPGAs by combining multiple dies called Super Logic Regions (SLRs) [8]. Considering that crossing from one SLR to another incurs a significant wire delay, partitioning can be done along these lines.

Partitioning based on design hierarchy would be another method that generates multiple scan-chains within each hierarchical sub-tree. Under the assumption that the FPGA placer typically tries to place elements within the same hierarchy close to each other, partitioning along hierarchy lines can be beneficial for $Fmax_{scan}$ as well, and especially so for Scenario 1. Moreover, having scan-chains which stay within the same hierarchy can help the eventual post-processing and analysis steps too.

Both approaches were employed in this work.

Exporting Scan Data: Lastly, we must consider what to do with data from the scan-dump: sending it off-chip or to another module to do post-processing.

We consider two different ways to do it in this study. One approach is using a hardened high speed on-chip serializers such as Xilinx GT transceivers [9] to export this device off-chip. In this approach, $Fmax_{D\_with\_SC}$ is also dependent on the serializer's bandwidth (BW_{serdes}) as shown Figure 3b. Equation 3 captures this new consideration:

$$Fmax_{D\_with\_SC} = \max\left(\frac{Fmax_{scan}}{NF_{scan}+2}\,,\,\frac{BW_{serdes}}{NF_{scan}}\right) \tag{3}$$

Another approach to capture multiple scan-chain outputs is by having a soft logic shim implemented on the FPGA fabric which gathers their outputs, buffers them, and transmits it using the AXI stream protocol. In this study we also implemented a parameterized soft logic shim that our tool flow uses to receive all scan outputs. This soft logic shim uses the AXI Capture module. One shortcoming of this approach however is its area overhead, which scales with the number of parallel scan-chains that exist.

Trade-off and Optimization: As discussed before, overall system performance $Fmax_{D\_with\_SC}$ is a function of $Fmax_{scan}$ and the maximum number of shadow flops across all scan-chains. Therefore to improve system performance, it's possible to break down a big scan-chain into multiple smaller ones to reduce the maximum number of shadow flops in any one scan-chain. Although having several scan-chains can increase $Fmax_{D\_with\_SC}$, it also adds complexity to output capturing logic at the end. Moreover, having too many parallel scan-chains unloading at the same time might saturate the off-chip bandwidth. Our scan-chain insertion tool flow explores this problem space to find a trade off between number of parallel chain and the number of shadow flops in each while considering area usage/bandwidth capabilities of export logic. During scan-out, the Fmax of the user design must be slowed to $Fmax_{D\_with\_SC}$ as defined in Equation 2, which describes the frequency if the state of all user flops is to be scanned out at every cycle. Relaxing this requirement to a complete state dump every N cycles would improve $Fmax_{D\_with\_SC}$ by the same factor – in this mode, software simulation (along with a trace of any external stimulus) could be used to interpolate missing user flop values.

4 Experimental Results

Our experiments are carried out using the Xilinx Vivado toolflow (version 2021.2) targeting Xilinx UltraScale+ devices. We have developed a tool that analyzes post-synthesis, post-place or post-route netlist and finds a good tradeoff between area/speed for adding soft scan capability to the user design to enable hardware testing/emulation. Our tool flow explores area/speed tradeoffs to find how many scan-chains should be implemented, how many flops in each scan-chains are needed and how the design needs to be partitioned. After finding a good tradeoff, it applies the *Add_Flopload* and *Add_ScanChain* steps described in the previous section. Moreover, the tool determines an appropriate value for $Fmax_{scan}$ (and thus, computing $Fmax_{D\_with\_SC}$) and constrains both clocks accordingly. After adding the soft scan-chain, our tool flow adds all the necessary control units and soft IPs for parallel capture to send the test flops' values off-chip. We examined 29 industrial emulation designs ranging from approximately 100,000 flops to over 200,000 flops. In those 29 designs, we targeted a different numbers of user flops using a different number of scan-chains and let our tool insert the necessary logic and connections.

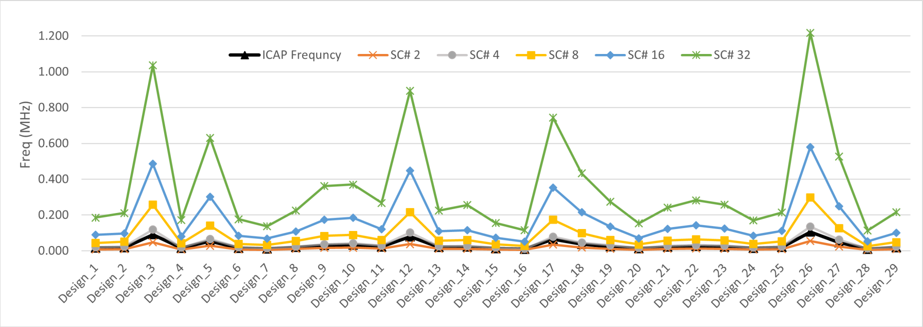


Fig. 6: Achievable user design Fmax with full per-cycle visibility — our work using soft-scan-chains being dumped continuously: SC#x ($Fmax_{D\_with\_SC}$); baseline using hardened readback[2]: ICAP ($Fmax_{D\_with\_RB}$).

Figure 6 shows the $Fmax_{D\_with\_SC}$ for different numbers of scan-chains while continuously dumping out all flop values in the design. To compare with the baseline approach, we also show an optimistic hardened readback approach[2] using the ICAP ($Fmax_{D\_with\_RB}$) and the configuration network to do so. As we can see after adding only four scan-chains to the design, $Fmax_{D\_with\_SC}$ exceeds that possible with the ICAP approach. By adding 32 scan-chains, on average the improvement over $Fmax_{D\_with\_RB}$ is 10x.

The bandwidth results for 29 designs in our design suite are shown in Figure 7. As discussed in the prior section, scan data needs to be transferred off-chip to be analyzed. We considered two approaches in this study to send the design status off-chip; 1) using GT ports and 2) dumping the values into DDR memory. Our tool flow, analyzes the bandwidth needed for sending the data off-chip and adds the necessary IP to the design. As mentioned, having more scan-chains is desirable to achieve higher $Fmax_{D\_with\_SC}$ but also requires more bandwidth to send the data off-chip. At 64 scan-chains, we exceed the bandwidth available supported by one GT resource. This means the tool flow needs to assign appropriate number of scan-chains to each GT based on the bandwidth. We face a similar limitation for DDR as well. A user needs to consider these limitations and force our tool

[2] The ICAP/$Fmax_{D\_with\_RB}$ results presented assume an optimistic but unrealistic value of $Efficiency = 1$ within Eqn. 1, meaning that hardened readback is capable of returning only user flop values. Even though this result is not attainable in current devices, we believe this reflects the upper-bound of what a configuration network based approach is capable of.

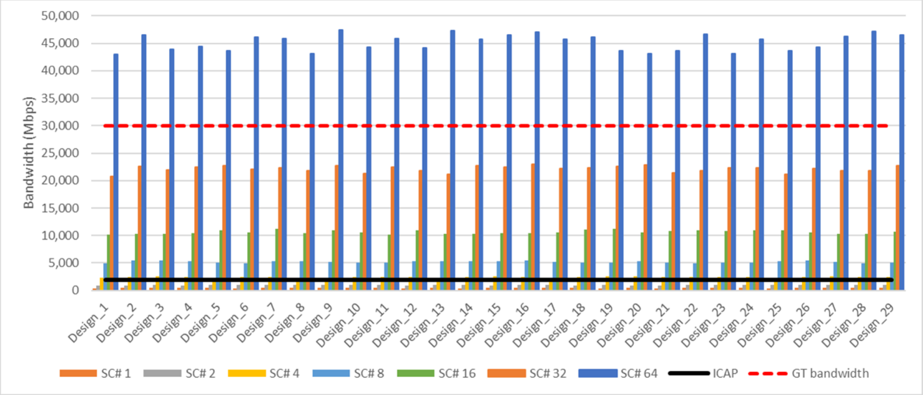


Fig. 7: Achievable soft scan/hardened readback^2 bandwidth, along with achievable off-chip bandwidth using GT transcievers.

to partition accordingly. This can be automated and will be addressed in future work.

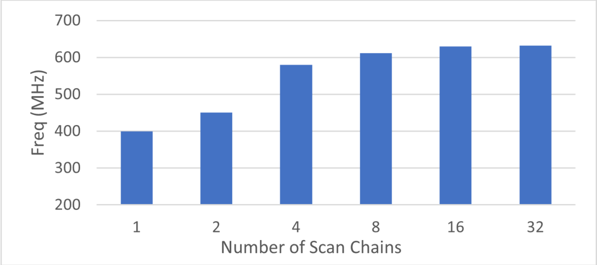


Fig. 8: Relationship of $Fmax_{scan}$ and number of scan-chains.

The average $Fmax_{scan}$ results for different numbers of scan-chains is shown in Figure 8. As we can see, by breaking a big scan-chain into a number of small chains, our tool flow can find a set of shadow flops closer to each other and create chains with lower delay.

Lastly, we added one long scan-chain using our tool flow, once for half of the flops in the design and once for all the flops in the design. We measured the placement and routing runtime and compared it with baseline (with no no scan-chains). We observed that place and route runtime for one long scan-chain is higher than having multiple shorter chains and we report the worst case scenario for runtime. Results of placement and routing runtime is shown in Figures 9 and 10 respectively. As we can see even for the worst case, the effect of adding scan-chains on place and route runtime is acceptable when gaining a 10x improvement for $Fmax_{D\_with\_SC}$ with only 32 scan-chains.

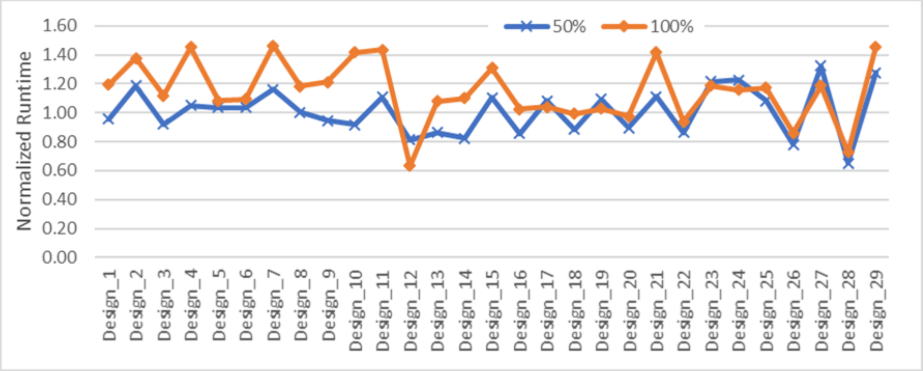


Fig. 9: Placement runtime, normalized to runtime for original user design.

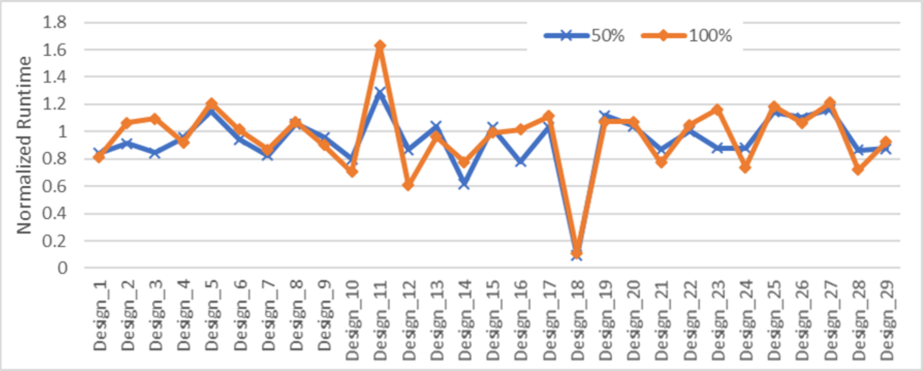


Fig. 10: Routing runtime, normalized to runtime for original user design.

Conclusion

FPGA prototypes have become an increasingly important part of the overall integrated circuit design and verification flow, providing the ability to test an integrated circuit running at (near) speed with realistic inputs and outputs. This make FPGAs great platforms for hardware emulation and provides visibility into many signals. This paper presents a soft scan-chain methodology for FPGA technology which can be applied to user design to give full and continuous visibility into all flop values, in a way that reduces its Fmax impact drastically compared to a hardened readback approach using the FPGA's built-in configuration network. Our tool flow analyzes the user design, explores its area/time tradeoffs, and partitions the scan connections into multiple parallel chains automatically in order to obtain an efficient solution. We evaluated our tool flow on a production-quality toolflow, using realistic industrial designs, and across a variety of different scan configurations to find the approach with the highest Fmax. Our findings show that by inserting only 32 parallel scan chains, post-placement, we can achieve a 10x higher Fmax compared to the baseline readback approach, allowing 100% visibility into designs able to continue running beyond 1 MHz.

Future Work: We plan to extend our work to the AMD/Xilinx Versal FPGA architecture [10] and leverage its high-bandwidth hardened Network-on-Chip (NoC) for on- and off-chip movement of scan data. Also, we plan to add and evaluate an automatic pipeline insertion to improve long connections within scan-chains, improving $Fmax_{scan}$ at the expense of efficiency-loss due to redundant flops, as well as to modify the insertion methodology to be congestion-aware so that routing runtime can be reduced.

A second direction would be to examine Scenario 3 (post routing insertion) in more detail since this scenario provides the benefit of leaving the user design fully untouched — such a concept that may require different algorithms could be explored using the open-source RapidWright framework [11]. Lastly, we intend to investigate how a hybrid implementation of using hardened readback (for reading Block RAM contents as well as any hard-to-reach shadow flops) in combination with our proposed soft scan-chain can lead to an even more efficient solution.

References

1. W. Lo, C. Choy, and C. Chan, “Hardware emulation board based on fpgas and programmable interconnections,” in *Proceedings of IEEE 5th International Workshop on Rapid System Prototyping*, 1994, pp. 126–130.
2. T. Wheeler, P. Graham, B. Nelson, and B. Hutchings, “Using Design-Level Scan to Improve FPGA Design Observability and Controllability for Functional Verification,” in *Field-Programmable Logic and Applications*, 2001.
3. A. Tiwari and K. Tomko, “Scan-chain based watch-points for efficient run-time debugging and verification of fpga designs,” in *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, 2003.*, 2003, pp. 705–711.
4. Xilinx, “LogiCORE IP AXI HWICAP,” 2020. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg134-axi-hwicap
5. Xilinx-Inc, “UltraScale Architecture Configuration,” 2022. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug570-ultrascale-configuration
6. D. Holanda Noronha, R. Zhao, Z. Que, J. Goeders, W. Luk, and S. Wilton, “An Overlay for Rapid FPGA Debug of Machine Learning Applications,” in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019.
7. S. Attia and V. Betz, “StateMover: Combining Simulation and Hardware Execution for Efficient FPGA Debugging,” in *FPGA ’20: The 2020 ACM/SIGDA International Symposium on FPGA, Seaside, CA, USA, February 23-25, 2020.*
8. K. Saban, “Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency,” *Xilinx, White Paper*, 2011.
9. Xilinx, “UltraScale Architecture GTY Transceivers,” 2020. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug578-ultrascale-gty-transceivers
10. AMD, “Versal: The First Adaptive Compute Acceleration Platform (ACAP),” 2022.
11. C. Lavin and A. Kaviani, “RapidWright: Enabling Custom Crafted Implementations for FPGAs,” in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018, pp. 133–140.