[Public]

Test Time Optimization: A Novel Staggered-capture Architecture Using A Token-passing Architecture

Khushboo Agarwal
Circuit Technology DFT
Advanced Micro Devices, Inc.
Santa Clara, California, USA
Khushboo.Agarwal@amd.com

Ari Shtulman
Circuit Technology DFT
Advanced Micro Devices, Inc.
Fort Collins, Colorado, USA
Ari.Shtulman@amd.com

Ravindra Mathanker
Circuit Technology DFT
Advanced Micro Devices, Inc.
Bangalore, India
Ravindra.Mathanker@amd.com

Ahmet Tokuz
Circuit Technology DFT
Advanced Micro Devices, Inc.
Santa Clara, California, USA
Ahmet.tokuz@amd.com

Kristian Yordanov
Circuit Technology DFT
SiBiZ
Plovdiv, Bulgaria
Kristian.Yordanov@amd.com

Ivan Marinov
Circuit Technology DFT
SiBiZ
Plovdiv, Bularia
Ivan.Marinov@amd.com

Miroslav Marev
Circuit Technology DFT
SiBiZ
Plovdiv, Bulgaria
Miroslav.Marev@amd.com

***Abstract*—As the scale of System-on-Chip (SoC) designs increases, scan test data volume becomes huge. Structural/Scan-based patterns are a major contributor of test time due to their sheer volume. Hence, there is an increasing need for reduction/optimization of these patterns. Scan pattern count grows exponentially for designs with many asynchronous clock domains where only one clock is pulsed per pattern. One way to reduce this pattern count is by staggering multiple clock pulses during the capture window in a single pattern, hence packing more detected faults per pattern. Staggered Capture is a well-known methodology designed to reduce pattern count in the presence of multiple asynchronous clock domains. However, challenges emerge due to the complexity involved in pulsing multiple clock domains at appropriate times within the same capture window. If the clock pulses land too close together, it can result in timing violations causing the patterns to fail on silicon. A common design method is to use counters loaded with a different value per clock domain, but this is complex and error prone. The technique proposed in this paper solves this challenge of manual and error-prone delay calculations by building a hardware solution which will cause staggering of pulses reliably. We propose passing a token from one clock domain to another to initiate capture pulse generation. This is like a domino effect which will create clock pulses in a staggered manner without any overlap. Once we trigger the first OCC (On-Chip Clock generator) and it finishes generating the capture pulses, it passes a token to trigger the second OCC and so on creating a domino effect. This automatically ensures that clock pulses are spaced far enough apart preventing a timing violation. Patterns generated using this technique will "successfully" pulse several OCCs in a staggered fashion for the asynchronous clock domains in the same pattern and hence coverage per pattern will increase. This will result in a smaller pattern set for the same test coverage reducing overall test cost and test time. Furthermore, test time will be used "more efficiently" to run other kinds of tests at the initial stages of ATE testing and help improve our Known Good Die (KGD)/DPPM metrics. We will also discuss that this is a relevant application where AI/ML can be used to guide the ordering as well as number of domains to stagger to generate most optimal test patterns.** (*Abstract*)

***Keywords*—Staggered capture, test time, test data volume** (*key words)*

I. Introduction (*Heading 1*)

Designers are packing more and more functionality into a single SoC leading to an increase in SoC size and complexity. Different components of SoC have several test modes and test requirements. This leads to increase in the scan test-modes and patterns required to robustly test the design. Scan test data volume grows exponentially for large designs with multiple asynchronous clock domains. This is because asynchronous clock domains are expected to be pulsed in a one-hot fashion which means only one clock is pulsed during capture in a given scan pattern. Therefore, more patterns will be generated to cover all the clock domains in the design. One way to reduce pattern count would be to pulse multiple clock domains in the same patterns, with the Automatic Test Pattern Generation (ATPG) tool being aware of the sequence of operations to correctly predict cross-domain interactions, and thus detecting more faults per pattern and amortizing the cost of the load-unload process on either side of the capture window. However, these clock pulses must be spaced (staggered) far apart from each other so as to not cause any timing violation or silicon failures.

In this paper we propose a built-in hardware methodology of token passing between the clock domains to achieve this stagger. This is a like a domino effect where the next clock domain can only pulse after the previous one is finished and a deterministic number of dead cycles have been passed successfully.

All designs use On-Chip Clock generators (OCCs) to emit the desired number of clock pulses during the capture window. Generally, designs have a dedicated OCC instantiated per asynchronous clock domain which means each clock domain has its own OCC. Figure 1 shows a simplified OCC architecture:

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

[Public]

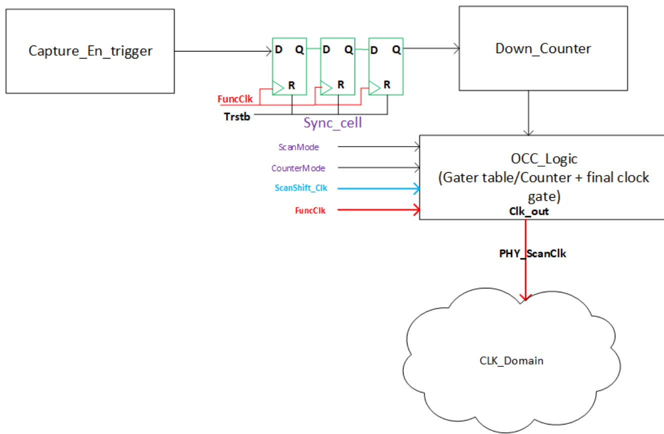

Figure 1: Simplified OCC architecture

The "Capture_En_trigger" is the OCC trigger signal which gets asserted to start the generation of capture pulses. This can come from the ATE or generated internally using ScanShiftEn de-assertion. Once this signal is asserted, it gets synchronized into the functional clock domain and then starts the down counter to start counting down. The down-counter can be of any size, making this solution scalable. The initial counter value can be loaded/configured via shift as the counter flops are on scan chains. Once the counter reaches 0 it triggers the capture pulse generation for this clock domain. The number of capture pulses to be generated is already loaded into the gater table flops at the end of shift. After the clock pulses have been generated, the gater table flops get loaded with all 0's and no further clock pulses will be allowed to pass through. This commences the clock pulse generation for the current OCC in the current pattern. Now we must go into the shift phase for the next pattern and trigger the next OCC and so on.

In current designs this needs to be done for each OCC (clock domain) on a per pattern basis. Only one OCC can be enabled in a given pattern. This causes the pattern count to grow exponentially for the designs with many asynchronous clock domains (and hence OCCs). This leads to large test time and test cost for scan pattern application on ATE.

If we wish to enable multiple OCCs in the same pattern in a staggered fashion on the existing architecture, we will have to program the down-counter in each OCC to a specific value such that it can finish emitting the clock pulses before the next OCC starts emitting its clock pulses. But since there are multiple OCCs and a varied range of frequencies in the design it would get very complex and error prone to achieve the desired stagger. Especially because each OCC gets triggered at the same time via a common ATE signal and we are only relying on the counters in the OCCs to achieve the stagger. This could be solved if there was a way to delay the triggering of the OCCs themselves, allowing us to control their trigger in a certain order. This paper proposes exactly that.

II. Proposed Staggered Clocking Architecture

Figure 2 shows a modified OCC architecture which supports a built-in hardware methodology of token passing between the clock domains to achieve an OCC stagger reliably in a domino fashion.

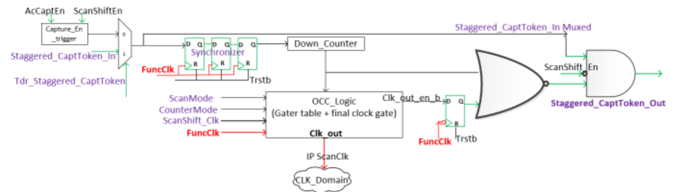

Figure 2: Proposed OCC architecture

"Capture_En_trigger" for each OCC can either come from the ATE signal or can be generated from the previous OCC. As shown in **Figure 2** there is a mux between the ATE-generated "Capture_En_trigger" and the Staggered_CaptToken_In coming from the previous OCC. This muxed signal is synchronized into the functional clock domain. Additionally, each OCC now generates a Staggered_CaptToken_Out signal which is used by the following OCC as a trigger. This signal can get asserted ONLY under the following conditions:

1. ScanShift_En has been de-asserted (we are in capture phase)
2. Down Counter for the current OCC has expired
3. Gater table flops are all set to 0 (i.e. current OCC finished generating clock pulses)
4. Trailing edge of final clock pulse has been generated from the current OCC

After all four conditions have been met, the OCC asserts Staggered_CaptToken_Out which triggers the next OCC and so on until the last OCC in the chain has been successfully triggered for capture pulse generation. The down-counter flops in each OCC can still be configured to a desired value (but could be same for each OCC in this scheme) to achieve the maximum stagger between clock pulses. We do not need to calculate a unique counter value as the OCC trigger is now based on the incoming token from the previous OCC in a domino fashion.

Now this is happening in the capture window of the same scan pattern, so we do not need to go into the shift phase until all the OCCs have been triggered in the current pattern. This allows us to pulse several clock domains reliably in the same pattern and achieving pattern count/test time reduction.

Figure 3 below shows 3 OCCs in a design and how they are connected in a chain for the staggered capture**. Figure 4** shows the waveform for staggered capture.

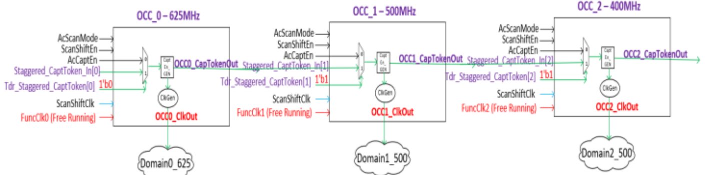

Figure 3: Staggered OCC blocks

[Public]

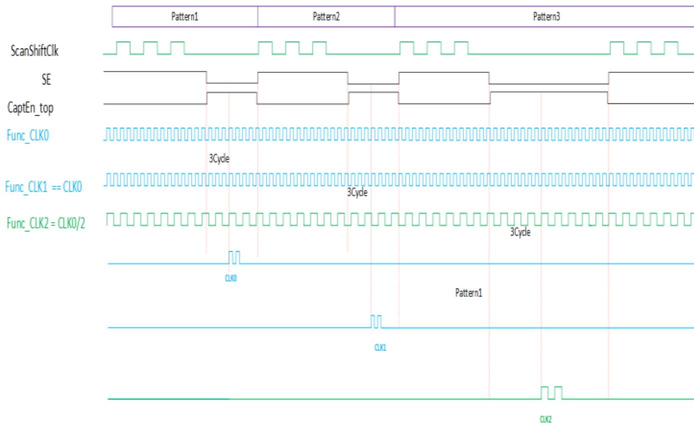

Figure 4: Staggered Capture wave form

As shown in Figure 3, the design can integrate all OCCs in a single chain. During ATPG, the first OCC is configured to be triggered via the asynchronous ATE signal, all subsequent OCCs in the chain will use Staggered_Token_In from their respective previous OCC as a trigger. This will create the desired domino effect for staggered capture-based pattern generation. Additionally, if needed, this chain can be broken down into several short segments using TDR programming, providing full flexibility during ATPG on how many OCCs we want to stagger. We could even decide to not use the staggering function at all and trigger each OCC independently. If any given OCC (in a chain) is bypassed (not used) during pattern generation, it will immediately pass the incoming token to the next OCC (without wasting tester cycles for token passing). Down-counter values in each OCC can still be programmed to create configurable amounts of delay before the OCC is triggered even after the arrival of a token from the previous OCC.
This scheme can be used for both stuck-at as well as at-speed pattern generation in a staggered fashion.

III. Results

The staggered capture token passing technique was applied to several circuits for both stuck-at and at-speed testing. The resulting test pattern count, test time, and test coverage were compared to a one-hot scan capture approach where each clock domain was captured independently in its own pattern. Each circuit contained different numbers of clock domains with differing scan capture frequencies and distributions of logic per clock domain. As a result, we were able to apply this technique to a broad range of scenarios. The testcases are described in **Table 1**.

Ckt	Scan Chain Length	Shift Freq. (MHz)	OCCs	Functional Clock Freq. (MHz)	Dominant OCC as % of Total Patterns (Stuck-At)	Dominant OCC as % of Total Patterns (At-Speed)
A	250	200	34	810	33.95%	43.28%
B	240	200	19	1250	38.06%	47.76%
C	130	200	5	570	63.64%	66.67%
D	194	200	9	1000	63.65%	84.59%

Table 1: Description of the 4 sample circuits used for data collection.

The pattern count, test time, and test coverage information for each circuit using a traditional one-hot scan capture technique is shared in **Table 2** and **Table 3:**

Ckt.	Pattern Count	Test Time (ns)
A	26,736	121,907,802
B	14,016	59,426,895
C	6,902	23,682,374
D	20,930	90,826,440

Table 2: At-speed pattern count and test time using one-hot scan capture technique.

Ckt.	Pattern Count	Test Time (ns)
A	25,560	103,004,460
B	12,685	52,822,885
C	6,131	20,032,952
D	8,444	34,357,320

Table 3: Stuck-at pattern count, test time, and coverage data using one-hot scan capture technique.

We applied the technique to at-speed as well as stuck-at pattern generation scenarios. As expected, enabling staggered capture resulted in pattern count and test time reduction. However, the degree to which we see that reduction was dependent on several factors. Those factors differed between stuck-at and at-speed pattern generation modes. We will discuss the results and the implications below. The at-speed test pattern generation experiments and corresponding test pattern count, test time, and test coverage results are shown in **Table 4**.

Ckt	OCCs per segment	Patt Count	Test Time (ns)	Patt. Count Delta	Test Time Delta	Test Cov. Delta
A	8/8/7/7/4	15,807	80,140,500	(40.88%)	(34.26%)	0.94%
B	6/5/4/4	9,329	42,323,825	(33.44%)	(28.78%)	(0.17%)
C	5	4,862	17,555,674	(29.55%)	(25.85%)	0.02%
D	5/4	18,208	84,474,060	(13.00%)	(6.99%)	(0.21%)

Table 4: At-speed pattern count, test time, and coverage data comparing one-hot capture to staggered capture using token passing.

While the data shows significant test pattern and test time reduction, there is also a clear correlation between the amount of improvement and the presence of a dominant clock domain. Circuit D has an extremely dominant at-speed scan OCC that is responsible for 84.59% of the total test patterns. As a result, staggering this extremely dominant OCC with the others does not produce much savings. The reason for this is that to achieve a similar test coverage, the tool could not eliminate many patterns from the dominant clock domain. However, when the patterns are more evenly distributed across the various clock domains, staggering is much more effective. This can be seen by looking at the dramatic increase in test pattern and test time savings seen in circuits A and B, relative to Circuits C and D. One important observation is that even with an extremely dominant OCC, there is clear test time savings using this token-passing technique. This is due to the relative length of a scan capture cycle relative to that of a scan shift (load-unload) cycle. Having a longer scan capture window will not out-weigh the

impact of reducing the number of total patterns. The stuck-at pattern generation experiments discussed below will reinforce the impact of a dominant clock domain.

The stuck-at test pattern generation experiments and corresponding test pattern count, test time, and test coverage results are shown in **Table 5**. The staggered OCC chains are stitched in the same way in this experiment as they were for the at-speed pattern generation.

Ckt	OCCs per segment	Patt Count	Test Time (ns)	Patt. Count Delta	Test Time Delta	Test Cov. Delta
A	8/8/7/7/4	11,254	52,215,500	(55.97%)	(49.31%)	(0.09%)
B	6/5/4/4	10,120	44,700,465	(20.22%)	(15.38%)	(0.02%)
C	5	4,902	17,737,834	(20.05%)	(11.46%)	(0.72%)
D	5/4	8,729	38,400,730	3.26%	10.53%	(0.44%)

Table 5: Stuck-at pattern count, test time, and coverage data comparing one-hot capture to staggered capture using token passing.

When looking at the same OCC staggering configuration in the stuck-at test mode, we observed that the test pattern and test time savings were not as evident in some cases. There were a few factors that contributed to this. Firstly, circuits A and B have longer scan chains which means that their shift time is longer. Therefore, the increase of the scan capture window had less of an impact on total test time relative to circuits C and D which have shorter scan chain lengths (and therefore a shorter shift time). Also, when a circuit contains a large number of clock domains and a single clock domain is not particularly dominant, the pattern count and test time savings are substantial. Circuit A, which contained 34 unique clock domains and a maximum pattern count of 34% for any one domain, we see over 50% savings on pattern count and almost 50% improvement on test time. As the pattern dominance of a single OCC increases, the savings in both pattern count and test time seen during staggering reduces. Circuit B has an OCC responsible for 38% of the patterns with the other test patterns distributed across the clock domains of the remaining 18 OCCs. In this case, there is still plenty of pattern count (over 20%) and test time (over 15%) improvement seen.

However, in circuits C and D which have a significantly dominant OCC in terms of pattern count, the benefit of the staggered capture isn't as evident. Circuit C, while showing a pattern count savings of 20% and test time savings of about 11%, shows a reduction in test coverage. The test coverage loss calls into question the validity of the pattern count and test time savings. Meanwhile, Circuit D did not show an improvement in either pattern count OR test time. It is also noteworthy that the test coverage was reduced in Circuit D as well. Our suspicion in these cases was that due to the dominance of a single OCC, staggering that OCC with others could result in some additional complexity in test cube generation which could reflect the test coverage loss.

To address the test coverage reduction, we took circuits C and D and re-configured our OCC staggering solution to separate out the dominant OCC into its own named-capture procedure, and stagger only the remaining non-dominant clock domain captures. **Table 6** shows the results of this experiment.

Ckt	OCCs per segment	Patt Count	Test Time (ns)	Patt. Count Delta	Test Time Delta	Test Cov. Delta
C	4/1	5,992	20,191,634	(2.26%)	0.78%	0.09%
D	4/4/1	7,756	33,555,620	(8.15%)	(2.33%)	(0.03%)

Table 6: Stuck-at pattern count, test time, and coverage data comparing one-hot capture to staggered capture where only the dominant clock domain is captured separately, and the remaining clock domains are staggered using token passing.

As can be seen in **Table 6** above, Circuit C's test coverage loss is completely recovered, while test time is now relatively in-line and pattern count is slightly improved relative to the baseline one-hot capture. Meanwhile, Circuit D now sees a noticeable improvement in pattern count of over 8% and a slight test time improvement of over 2%. Circuits D's test coverage delta compared to traditional one-hot coverage improved dramatically from a coverage drop of 0.44% to a very slight test coverage loss. The removal of the dominant OCC from the staggered capturing appears to have eased the strain on the ATPG tool.

The question remains, however, can Circuits C and D benefit from staggered capture in both pattern count AND test-time while keeping test coverage in line with the baseline case? To this point, we have not observed so. One more configurability capability of the staggered capture token passing technique was necessary to figure that out. What would happen if we shortened the staggered OCC chains even more? This would reduce the capture window cycle count substantially. We attempted this for both Circuits C and D and the results can be seen in **Table 7** below.

Ckt	OCCs per segment	Patt Count	Test Time (ns)	Patt. Count Delta	Test Time Delta	Test Cov. Delta
C	2/2/1	5,714	19,124,958	(6.80%)	(4.53%)	0.08%
D	2/2/2/2/1	8,501	35,276,980	0.67%	2.61%	(0.02%)

Table 7: Stuck-at pattern count, test time, and coverage data comparing one-hot capture to staggered capture with a shortened OCC staggering and the dominant OCC capturing without being staggered.

Reducing the staggered OCC chain length showed some interesting results. Circuit C has finally showed an improvement in all three metrics; pattern count reduced by 6.8%, test time reduced by 4.53%, and the test coverage actually increased by 0.08%. Clearly, for circuit C, this is the best way to configure the stuck-at scan capture staggering. For circuit D, however, we see that the results are not as good as the previous experiment. This means that shortening the OCC staggering is not the best choice in the case of circuit D. However, we can see that there is a configuration where circuit D does achieve savings in pattern count and test time (shown in **Table 7**) with just a minor cost to stuck-at coverage.

[Public]

IV. Conclusion

In this paper, we leveraged the software configurability of the staggered capture token passing approach by adjusting the OCC staggered chain connectivity. We observed that for at-speed scan, this approach shows a clear test pattern and test time savings for all four tested cases in the basic configuration. When generating stuck-at patterns, we can conclude that the pattern count and test time savings can still be achieved in all four cases tested, however configurability becomes more important. In the absence of a single dominant clock domain that is responsible for a majority of the test patterns, the test cost savings of staggered capture are relatively easy to realize. However, there are corner cases where the length of a staggered OCC chain (how many OCCs are staggered in a given pattern) or the exclusion of certain clock domains from the staggering scheme can significantly alter the results. Thus, the true strength of the token-passing staggered capture approach presented in this paper is the configurability of the OCC staggering through test initialization.

While we explored a significant number of configurations in this paper, there is still more work to be done. In the future, we intend to explore the impact to pattern count and test time when stitching the OCCs into a staggered chain in different orders. It is likely that capturing on flops in one domain relative to another domain when those clock domains are interacting could yield different results. With the emergence of artificial intelligence (AI), we will also explore the possibility of leveraging AI tools to determine the ideal staggered chain length and order. Finally, we'd like to study any potential IR drop impacts of the staggered capture approach relative to the one-hot capture base-line discussed in this paper.