An Efficient Test Time Model for Optimizing Tessent SSN for a 3D Design

Vasubabu Ravipati, Shyam Kallepalli and Lance C Cheney

May 15, 2022

An Efficient Test Time Model for Optimizing Tessent SSN for a 3D Design

***Abstract*—**This paper presents an efficient scheme to calculate and optimize test time when using Tessent Streaming Scan Network (SSN) by bypassing the traditional vendor solution of patterns re-targeting process. SSN retargeting process requires stand-alone patterns of all partitions present in SSN network and availability of full chip SoC Netlist model which is too late in the design cycle. Many "case" studies can be performed for estimating test volume/test time, easily without hardware changes using the described algorithm in the paper. This methodology helps to update test network design for optimal test time and volume in very early phase of the project, thereby reducing iterative and late change costs for DFT design.

Keywords*— *SSN, pattern re-targeting, test time, test volume, 3D

I. Introduction

With each generation, SoC designs have increased in size and complexity. To maintain same or lesser tester time & tester volume than previous generation SoC designs, the use and optimization of a scan test system fabric is a must. Test volume directly depends on test fabric design spec therefore, it is required that the design team analyze, estimate, and react to test time growths in a timely fashion and do the necessary changes to the test fabric spec early in design cycle. With classical designs this meant tuning the mapping and muxing of test-port IOs to individual scan endpoints throughout the design process as partitions were added, deleted, or changed. Streaming Scan Network (SSN) is a new structural scan test system from Siemens which promises a portable and scalable architectural solution and eliminates top-level IO mapping and muxing. It delivers efficient scan vectors by enabling parallel testing of all partitions in the design [1] and decouples the size and design of the test-port from the internal scan construction. However, using only the tooling supplied by the vendor, scan pattern volume and test time estimation with SSN fabric requires a SoC-level ATPG re-targeting flow which takes too long due to the late availability of full chip soc netlist model and results in late feedback to design. In this paper we would like to share a methodology for estimating pre-silicon pattern volume and test time for any given SSN specification without the use of a retargeting flow. We assess test time with various bus widths, regrouping of partitions across SSN networks based on bandwidth usage, and the results of optimization of the SSN network distribution. We are using SSN architecture for an SoC scan testing, and this paper details the method we have implemented to optimize test time in very early phase of design cycle.

II. SoC SSN IMPLEMENTATION OVERVIEW

In our 3D stacked SoC design, we have a top-die which contains IO buffers that are routed as package balls using micro-bumps (ubumps) and through-silicon vias (TSV) routing through the bottom/base-die [2]. The top die and base die contain multiple physical partitions, all of which must be tested via scan. In this design, we created four parallel SSN networks to address all the partitions: the Top Left network (TLNW) and Top Right network (TRNW) for top-die, and the Base Left network (BLNW) and Base Right network (BRNW) for base-die, as shown Figure 1. Parallel networks are created to ease the scan routing and timing convergence, as we have limited routing channel availability and in due consideration of the IO placement. For TLNW and TRNW the SSN data bus and clock inputs are routed from package balls through TSV, to probe-able ubumps in base-die, through IO buffers in top-die and onto the SSN fabric in the top die. We have added repeater stages for SSN data bus out of TLNW and TRNW prior to IO buffers in the top-die; these are routed to ubumps in the base-die and TSVs to package balls. The base die networks are available independently via probe-able ubumps at sort but are connected only through the top die network after die stacking. For package-level testing the base-die sort content must be updated through retargeting flows to comprehend the additional repeater stages in top-die. Figure 1 depicts custom muxing logic in the top die to stitch the TLNW into the BLNW and similarly for TRNW into BRNW. By enabling this feature, we convert 4 parallel SSN networks to 2 parallel SSN networks during package testing. This helps evaluate future SoC designs where we could be limited by IO buffer availability which is beyond the scope of this paper.

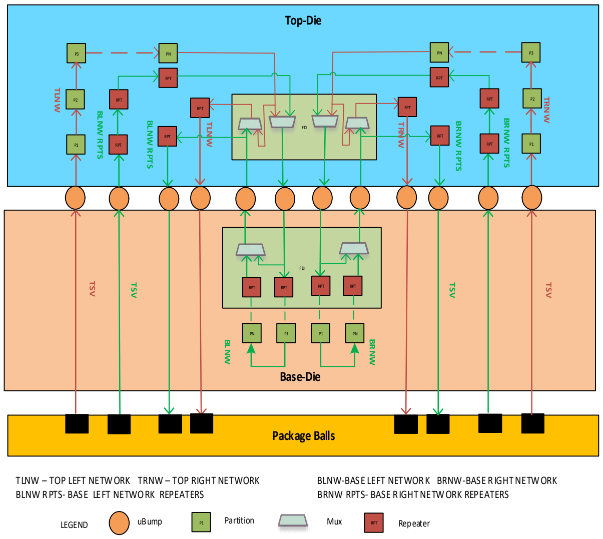

Top-Die
P3
PN
RPT
P2
P1
TLNW
BLNW RPTS
FDI
TRNW
BRNW RPTS
Base-Die
BLNW
BRNW
TSV
Package Balls
TLNW – TOP LEFT NETWORK TRNW – TOP RIGHT NETWORK
BLNW RPTS- BASE LEFT NETWORK REPEATERS
BLNW-BASE LEFT NETWORK BRNW-BASE RIGHT NETWORK
BRNW RPTS- BASE RIGHT NETWORK REPEATERS
LEGEND
uBump
Partition
Mux
Repeater

Figure 1: SoC SSN Arch Overview

In legacy GPIO flows, partitions always have different shift counts even with good planning. This results in a lot of wasted bandwidth due to padding that is required to synchronize the capture cycles across partitions. SSN alleviates this wastage by varying the amount of data sent to each partition and enabling independent captures between partitions. This is done by using a packet-based system where each packet contains a variable amount of scan data for each partition under test. Packets can span multiple test-port clock cycles and each controller manages its transitions between shift and capture modes independently, which reduces the droop effect of scan shift. The number of bits allocated for each SSH is dependent on the total amount of data it must consume for the duration of the test but cannot exceed the number of bits used for one shift operation [3]. Additionally, each packet delivered on the network must contain at least one bit of data for every SSH block that is enabled for that test [3]. The final test volume is dependent on the configuration of all of the enabled SSH blocks in the system: each blocks' total shift cycles and their channel counts. A model to convert these inputs into a test volume estimate can be used to quickly provide feedback on the effects of a channel count change, partition-level content truncation or growth, and to estimate test time before a full retargeting run can be completed. Under a standard flow, partition-level ATPG patterns are generated, then retargeted to SoC level, with all partitions enabled in parallel. This provides a test time estimate and details about the packet utilization. We can analyze the resulting pattern and regroup the partitions under a single SSH and/or adjust the SSN bus width for each network. This information is fed into SSN RTL spec and new RTL is then generated. The final optimized test time and optimized network can be achieved within a few iterations. The biggest drawbacks are that the iteration loop is very long, intrusive to the design cycle, and each iteration requires a new model for retargeting, as shown in the Figure 2.

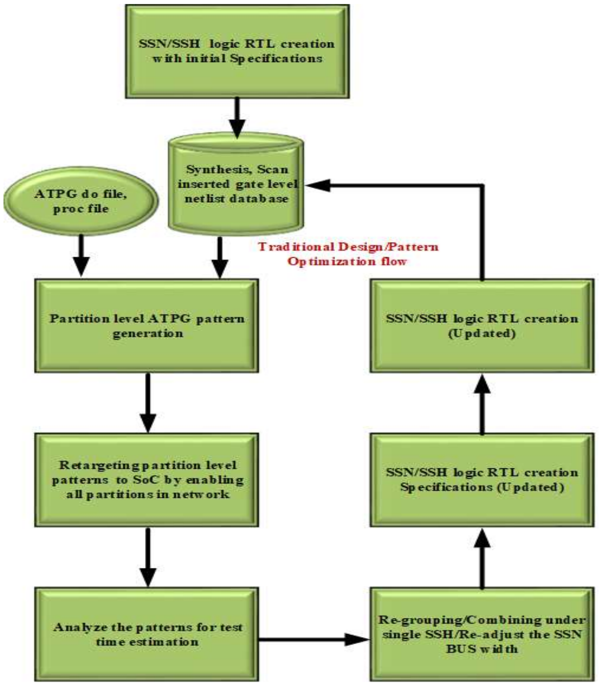

SSN/SSH logic RTL creation with initial Specifications
Synthesis, Scan inserted gate level netlist database
ATPG do file, proc file
Traditional Design/Pattern Optimization flow
Partition level ATPG pattern generation
SSN/SSH logic RTL creation (Updated)
Retargeting partition level patterns to SoC by enabling all partitions in network
SSN/SSH logic RTL creation Specifications (Updated)
Analyze the patterns for test time estimation
Re-grouping/Combining under single SSH/Re-adjust the SSN BUS width

Figure 2: Traditional SSN design optimization flow

With offline modeling for test time and SSN network optimization we bypass the retargeting flow altogether and feedback to design is much faster and minimizes the design rework, with shorter iterations as shown in Figure 3. We can evaluate test time for other SSN grouping and EDT channel count permutations without making RTL or hardware changes, and use the results for the final optimal network.

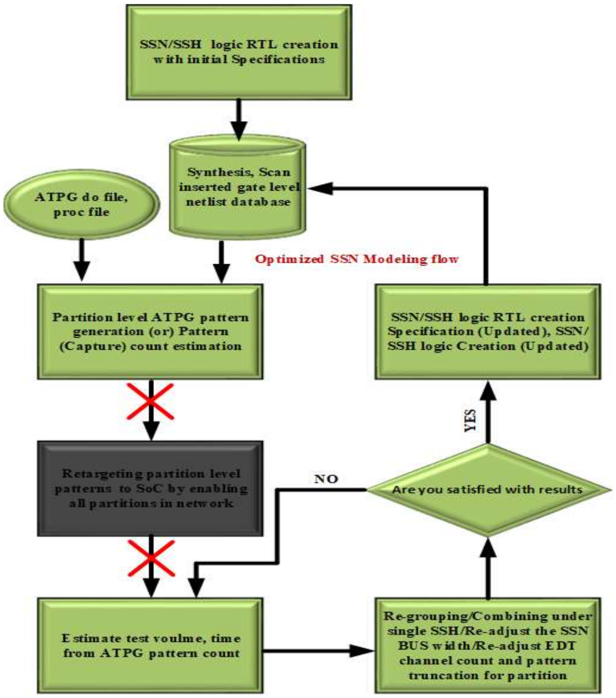
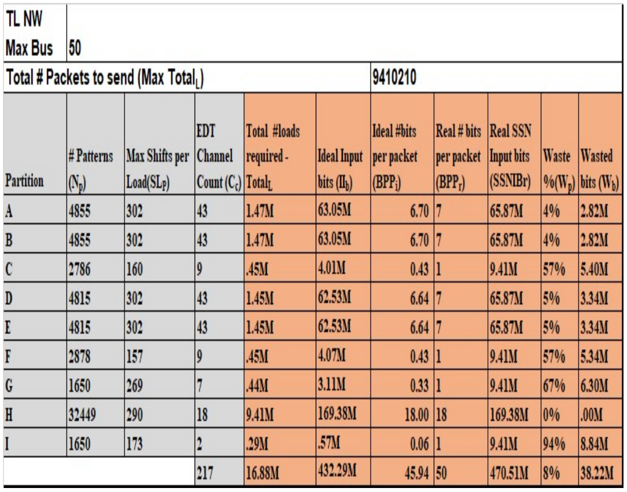

SSN/SSH logic RTL creation with initial Specifications
Synthesis, Scan inserted gate level netlist database
ATPG do file, proc file
Optimized SSN Modeling flow
Partition level ATPG pattern generation (or) Pattern (Capture) count estimation
SSN/SSH logic RTL creation Specification (Updated), SSN/SSH logic Creation (Updated)
Retargeting partition level patterns to SoC by enabling all partitions in network
YES
NO
Are you satisfied with results
Estimate test voulme, time from ATPG pattern count
Re-grouping/Combining under single SSH/Re-adjust the SSN BUS width/Re-adjust EDT channel count and pattern truncation for partition

Figure 3: Optimized SSN Modeling flow

SSN Packet sizing and Wastage Calculation:

Our vision was to create a spreadsheet model that could be used for rapid what-if evaluation of changes to the design and content: EDT channel counts, capture/pattern count truncation, chain count adjustment, SSH sharing, etc. We would then run partition-level ATPG to get initial capture count estimates and permute variables until an optimal solution was found, at which point the changes could be committed to the RTL design. The key to this evaluation was building the algorithm which determines the total packet size and number of bits in that packet that are used by each partition. Table 1 shows the details of pattern volume of the 9 partitions in TLNW, including the number of scan captures/Patterns (N_p), number of shifts per capture for load/unload (SL_p) and the EDT channel count (C_c). We used Symmetric EDT for all partitions and did not use the on-chip compare feature. The "# Patterns" (Np) multiplied with "Max Shifts per load" (SL_p) provides the approximate "shifts/loads required" ($Total_L$) for each partition, which can be multiplied by the "Channel Count" (C_c) to get the ideal number of input bits (II_b) required for each partition's test. Because each packet can contain at most one shift for each SSH, the number of packets required is constrained by the partition with the highest number of loads ($MaxTotal_L$)required, which in this example was partition "H" @9.41Million loads. We then calculate the number of bits per packet that should be sent to each partition using the ratio of its $Total_L$ vs the now-determined $MaxTotal_L$ packet count, multiplied by its channel count (C_c), ($Total_L$ / $MaxTotal_L$) * C_c which is shown here in the "# bits per packet (ideal)" BPP_i column. Fractions are always rounded up to determine the actual number of bits used for each SSH, providing us the final "Real bits per packet" BPP_r. The actual number of input bits delivered by the SSN bus ($SSNIB_r$) for each partition is calculated by multiplying the partition's "# bits per packet (real)" with "Total #packets to send" (BPPr * $MaxTotal_L$). For example: partition "I" has $SSNIB_r$ input volume of 9.4 megabits. Wastage percentage (W_p) is then calculated based on the total input bits ideal II_b (0.57M) vs total SSN bits real $SSNIB_r$ (9.41M) for each partition. This results in maximum wastage of 94% for partition "I" in this network. Similarly, Wasted Bits (W_b) per partition is determined as delta between $SSNIB_r$ and II_b as shown in Table 1. Understanding this algorithm aids in determining the right solution to improve the overall efficiency of the system. In this case SSN has done a good job of controlling data delivery and the overall waste is fairly small at just 8%, and the test volume (and thus test time) is dominated by partition H. Since SSN can easily support internal changes in partition EDT channel counts without affecting test-port or fabric design, a range of options are available at relatively low design cost to minimize vector volume. It takes just seconds to do what-if analyses such as considering the effects of doubling the channel and chain count for partition H assuming no increase in captures/#patterns (4% reduction in total test volume), or of reducing the number of channels allocated to partitions with low capture/#pattern counts, such as "F" or "C" (which has no effect).

Table 1: TLNW SSN packet size and wastage calculation



TL NW Max Bus	50									
Total # Packets to send (Max $Total_L$)						9410210				
Partition	# Patterns (N_p)	Max Shifts per Load(SL_P)	EDT Channel Count (C_c)	Total #loads required - $Total_L$	Ideal Input bits (II_b)	Ideal #bits per packet (BPP_i)	Real # bits per packet (BPP_r)	Real SSN Input bits (SSNIBr)	Waste %(W_p)	Wasted bits (W_b)
A	4855	302	43	1.47M	63.05M	6.70	7	65.87M	4%	2.82M
B	4855	302	43	1.47M	63.05M	6.70	7	65.87M	4%	2.82M
C	2786	160	9	.45M	4.01M	0.43	1	9.41M	57%	5.40M
D	4815	302	43	1.45M	62.53M	6.64	7	65.87M	5%	3.34M
E	4815	302	43	1.45M	62.53M	6.64	7	65.87M	5%	3.34M
F	2878	157	9	.45M	4.07M	0.43	1	9.41M	57%	5.34M
G	1650	269	7	.44M	3.11M	0.33	1	9.41M	67%	6.30M
H	32449	290	18	9.41M	169.38M	18.00	18	169.38M	0%	.00M
I	1650	173	2	.29M	.57M	0.06	1	9.41M	94%	8.84M
			217	16.88M	432.29M	45.94	50	470.51M	8%	38.22M

This calculation of wasted bits and wasted bits percentage per partition can be extended to all networks using simplified formula as shown below.

$W_p = (1 – (II_b/SSNIB_r))$ *100

Substituting for IIb in above equation we get:
$W_p = (1 – ((Total_L * C_c)/SSNIB_r))$ *100

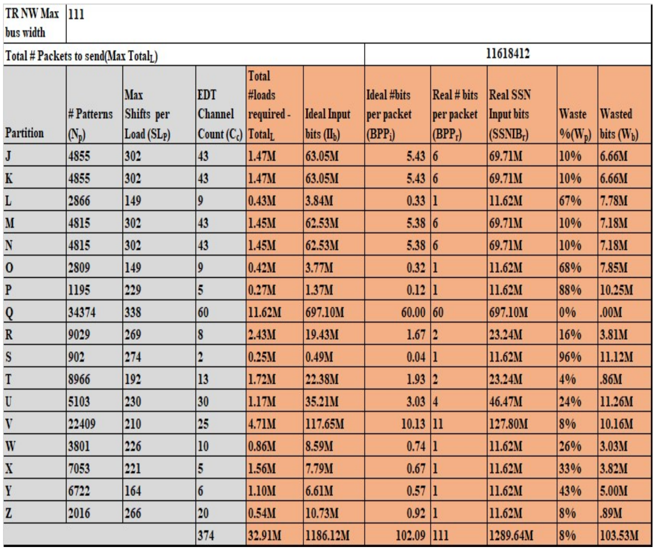
Substituting for TotalL and SSNIBr we get:

$W_p = (1- ((N_p*SL_p*C_c)/BPP_r*MTotal_L)))$ *100

Similarly Wasted Bits Per partition is determined by,
$W_{b =} (BPP_r*MTotal_L) – (N_p*SL_p*C_c)$

Table 2 below shows the TRNW's pattern volume of all 17 partition as estimated by our model using the algorithms described earlier. In this case also SSN has done a good job of controlling data delivery and the overall waste is fairly small at just 8%, but the test volume (and thus test time) is dominated by partition "Q", which requires ($MaxTotal_L$) of 11.62M loads.

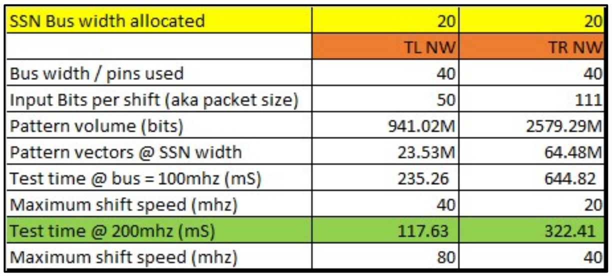
Table 2: TRNW SSN packet size and wastage calculation



TR NW Max bus width	111									
Total # Packets to send(Max $Total_L$)						11618412				
Partition	# Patterns (N_p)	Max Shifts per Load (SL_P)	EDT Channel Count (C_c)	Total #loads required - $Total_L$	Ideal Input bits (II_b)	Ideal #bits per packet (BPP_i)	Real # bits per packet (BPP_r)	Real SSN Input bits ($SSNIB_r$)	Waste %(W_p)	Wasted bits (W_b)
J	4855	302	43	1.47M	63.05M	5.43	6	69.71M	10%	6.66M
K	4855	302	43	1.47M	63.05M	5.43	6	69.71M	10%	6.66M
L	2866	149	9	0.43M	3.84M	0.33	1	11.62M	67%	7.78M
M	4815	302	43	1.45M	62.53M	5.38	6	69.71M	10%	7.18M
N	4815	302	43	1.45M	62.53M	5.38	6	69.71M	10%	7.18M
O	2809	149	9	0.42M	3.77M	0.32	1	11.62M	68%	7.85M
P	1195	229	5	0.27M	1.37M	0.12	1	11.62M	88%	10.25M
Q	34374	338	60	11.62M	697.10M	60.00	60	697.10M	0%	.00M
R	9029	269	8	2.43M	19.43M	1.67	2	23.24M	16%	3.81M
S	902	274	2	0.25M	0.49M	0.04	1	11.62M	96%	11.12M
T	8966	192	13	1.72M	22.38M	1.93	2	23.24M	4%	.86M
U	5103	230	30	1.17M	35.21M	3.03	4	46.47M	24%	11.26M
V	22409	210	25	4.71M	117.65M	10.13	11	127.80M	8%	10.16M
W	3801	226	10	0.86M	8.59M	0.74	1	11.62M	26%	3.03M
X	7053	221	5	1.56M	7.79M	0.67	1	11.62M	33%	3.82M
Y	6722	164	6	1.10M	6.61M	0.57	1	11.62M	43%	5.00M
Z	2016	266	20	0.54M	10.73M	0.92	1	11.62M	8%	.89M
			374	32.91M	1186.12M	102.09	111	1289.64M	8%	103.53M

Table 3 details the test time calculation for both TLNW and TRNW assuming the SSN Bus width allocated as 20 in/20 out and omitting SSN IJTAG setup test time which would be a be constant for the network and cannot be modulated by internals of the network. Test time is approximated here by dividing the total pattern volume (bits) with the bus width to get pattern vectors @ SSN width, and then multiply by bus frequency. Calculations are done for both 100mhz and 200mhz bus frequencies. With our estimated SSN bus POR @200mhz, test time for TLNW is ~117ms and TRNW will be ~ 322ms. In our architecture both these networks are in exercised in parallel, so the maximum test time is limited by TRNW (322ms).

Table 3: Test Time calculation for given SSN bus width



SSN Bus width allocated	20	20
	TL NW	TR NW
Bus width / pins used	40	40
Input Bits per shift (aka packet size)	50	111
Pattern volume (bits)	941.02M	2579.29M
Pattern vectors @ SSN width	23.53M	64.48M
Test time @ bus = 100mhz (mS)	235.26	644.82
Maximum shift speed (mhz)	40	20
Test time @ 200mhz (mS)	117.63	322.41
Maximum shift speed (mhz)	80	40

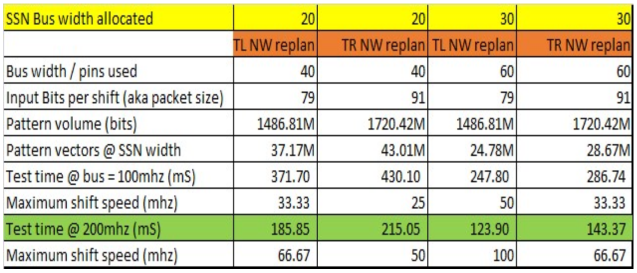
III. RESULTS AND SUMMARY

TRNW is the test time limiter for Top-die SSN scan testing, with partition "Q" requiring 11.62M packets of desired bus width of 111 in/out bits, while the TLNW is idle for more than 60% of the scan testing. We evaluated the below options to optimize the test time.

1. Rebalancing TRNW vs. TLNW, by moving some of the physical routable partitions to TLNW
2. Increasing the SSN Bus Width by using additional IOs
3. Combining adjoining partitions into a single SSH, targeting those with a large % of wastage bits

In option 1, we moved partitions R thru Z into TLNW which are physically accessible without any timing limitations. Using our test time model, we observed ~33% reduction in test time(~107ms). There is an ~68ms of test time increase for TLNW but is not of concern since it is not a limiter as both TLNW, TRNW runs in parallel during silicon scan testing. As seen in the Table 4 below in this option, the input bits for the TLNW changed to 79 as ideal case. With the partition moved to TLNW the total wastage of bits has reduced considerably. In option 2, we evaluated the feasibility of sharing leftover IO's and able to increase the SSN Bus Width from 20 in/out to 30 in/out. Using our test time model, we were able to quickly evaluate the test time benefits and was able to roll in the required design changes without waiting for ATPG re-targeting flow results. Similarly, we have evaluated for base-die SSN networks but haven't found any significant improvements that would dictate a design change.

Table 4: Test time calculation for given SSN bus width 20,30 and with TL NW, TR NW replan



SSN Bus width allocated	20	20	30	30
	TL NW replan	TR NW replan	TL NW replan	TR NW replan
Bus width / pins used	40	40	60	60
Input Bits per shift (aka packet size)	79	91	79	91
Pattern volume (bits)	1486.81M	1720.42M	1486.81M	1720.42M
Pattern vectors @ SSN width	37.17M	43.01M	24.78M	28.67M
Test time @ bus = 100mhz (mS)	371.70	430.10	247.80	286.74
Maximum shift speed (mhz)	33.33	25	50	33.33
Test time @ 200mhz (mS)	185.85	215.05	123.90	143.37
Maximum shift speed (mhz)	66.67	50	100	66.67

IV. CONCLUSIONS AND FUTURE PLAN

The SSN Bus Width and Test time optimization model which we created for our SoC design is scalable and re-usable for all SoC's which are using the SSN scan solution. This can also model asymmetric/dual EDT which can be useful when SSN is operating in on-chip compare mode. This modeling could be automated to populate the input data such as number of captures/Patterns, EDT channel count, max loads per partition by directly reading such information from design and atpg pattern generation area. We have also observed that Tessent SSN retargeting introduced bandwidth throttling that limited the usefulness of two parallel independent networks during retargeting flow as shown in Figure 4. Ideal scenario for two independent SSN networks patterns re-targeting as separate runs shown in Figure 5. More optimized testing solution requires enhancement to vendor tooling such that multiple parallel networks can independently tested as depicted in Figure 6. We anticipate more designs to adopt SSN with these enhancements.

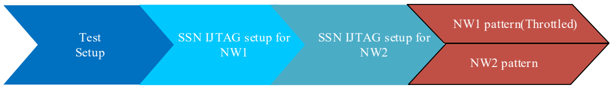


Figure 4: NW1, NW2 pattern retargeting by Tessent tool

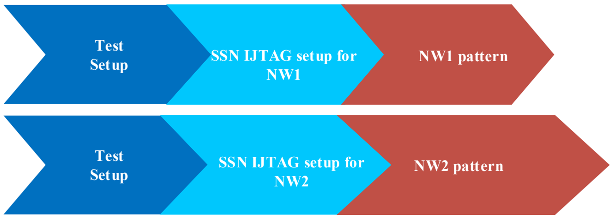


Figure 5: Standalone NW1, NW2 pattern re-targeting

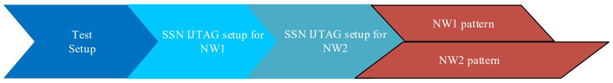


Figure 4: Ideal scenario and requirement for NW1, NW2 pattern retargeting

V. ACKNOWLEDGEMENTS

We would like to thank all our colleagues in AXG DFT team who worked on Streaming Scan Network implementation to enable us to come with this algorithmic test time optimization model without going through SoC level pattern re-target flow. We would like to extend our thanks to Siemens AE's who provided the training on Streaming Scan Network solution.

VI. REFERENCES

[1] J-F. Côté, et.al., "Streaming Scan Network (SSN): An Efficient Packetized Data Network for Testing of Complex SoCs," ITC, 2020

[2] D. B. Ingerly, et.al., "Foveros: 3D Integration and the use of Face-to-Face Chip Stacking for Logic Devices", IEEE, 2019

[3] Siemens Tessent Streaming Scan Network (SSN) User manual