Novel Approach for Verification of Multi Die Booting Using Disruptive Distributed Simulation Methodology

Manikanta Gummadidala (Samsung-India), Pradeep Kumar Sahoo (Samsung-India), Sunil Shrirangrao Kashide (Samsung-India), M Nitin Kumar (Samsung-India), Chethan Kumar G (Cadence-India)

***Abstract*- The BootROM, implemented as a small mask ROM or write-protected flash memory embedded within the processor chip is responsible for executing the first code when system is powered-up or reset. BootROM code is responsible for fetching all software binaries from external devices which includes the BL (Boot Loader), authenticate it and keep it in the system RAM. To enhance system security, the BootROM integrates AES (Advanced Encryption Standard) and SHA (Secure Hash Algorithm) in the chip, checks the loaded BL for security, and provides personalized key management for the chips to prevent from cyber-attack. The ultimate goal of BOOTING is to fetch all BLs from external device and to bring up the OS (Operating System). In the ever evolving era of chip manufacturing, conventional approaches of achieving functionality, form factor, cost and much demanding power/performance goals pave the way of transitioning to smaller process nodes [1]. However, manifold increase in compute/performance requirement has pushed the monolithic System on Chip (SOC) to sizes that are challenging to fabricate with acceptable yields. Additionally, the diminishing returns of advanced nodes have made it economically impractical to accommodate all of the required logic, IO and memory for compute intensive applications within the limits of manufacturing equipment [1].**

To address these limitations, chip designers are embraced with multi-chip and multi-die designs where larger designs are partitioned into smaller designs often referred as chiplets. These chiplets are then integrated into a single package for multi-die and different packages for multi-chip to achieve the desired form factor and power goals. Additionally, these approaches provide both scalability and flexibility to cater to different market segments and specific needs. Despite clear advantages of Multi-die/Multi-chip systems, numerous novel and distinctive challenges need to be addressed. Key verification challenges include availability of a scalable test bench to realize multi-chip/multi-die simulation, infrastructure limitation, homogeneous(multi-die) and heterogeneous(multi-chip) design and increase in simulation run time with debug complexities. This paper proposes a unique scalable approach called "DISTRIBUTED SIM" to overcome all mentioned challenges within constrained verification timeframe. DISTRIBUTED SIM provides a distributed simulation approach where each individual die/chiplet has its own simulation and all simulations communicate with each other over socket at a specified time. Preliminary result shows a 10x improvement in test bench development, 20-30 % improvement in simulation time, 15 % improvement in processor usage and 10x reduction in resource requirement over a conventional single wrapper approach of multi-die/multi-chip simulation. All these promising figures claim DISTRIBUTED SIM as a robust approach towards multi-die/multi-chip verification.

I. Introduction

The function of BootROM in multi die environment is to Boot up all the chips/dies securely and bring up the OS. This involves copying of binaries from external storage to the system memories of primary and the secondary die.

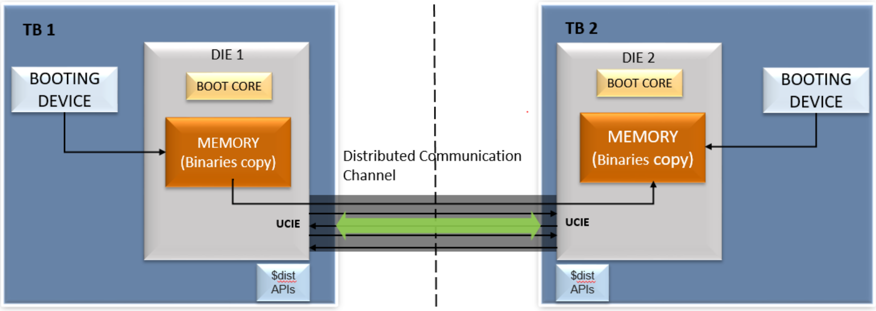


Fig.1 Multi-die Booting

The die to die connection is through UCIe interface. The primary die is connected to the storage device such as UFS (Universal Flash Storage). Primary die will copy all binaries from UFS to its system memory and copies to the secondary die system memory over UCIe link.

In modern VLSI, Moore's law enabled semiconductor companies to double the number of gates in an Integrated Circuit(IC) every two years since several decades. As Moore's law slows, device scaling for monolithic system on chip(SOCs) is noticeably less while the cost of newer, more complex process nodes continues to increase steadily [1].

However, the demand for faster, better and smarter silicon is only growing as the exploding market of smart everything and ubiquitous artificial intelligence (AI) drive a massive push for faster speeds and more transistors [1]. To quench the ever increasing thirst of quicker time-to-market, semiconductor companies across globe are revolutionizing both architectural function and form with new-multi-die/multi-chip systems. Segregating a larger monolithic chip design into smaller, proven dies results in 1) higher yields 2) lower silicon costs over time 3) required form factor and performance 4) customized design 5) faster time to market. These multifaceted advantages make multi-chip/multi-die design as a global choice in semiconductor industry.

However, mentioned benefits of multi-chip/multi-die designs come along with numerous new challenges. One of the biggest challenge is the verification of multi-chip/multi-Die design. Standalone BootROM verification for a single die takes a lot of time in complex SOCs. Integrating two die in a single testbench (TB) has its disadvantages. The addressable challenges are

1) TB development: Unified Testbench development time for each additional chiplet/die
2) Performance: Instantiating multiple homogeneous and/or heterogeneous designs in a single wrapper causes considerable impact on testbench size and simulation performance (Long run time)
3) Infrastructure limitation: Need machines with higher compute & memory to build/run multidie simulations
4) Design Cycle: With long runs it has implication on bugs found after 2 weeks. Rerun will take another 2 weeks, which will effect project timelines.
5) Customer requirement: Scalable TB development is not possible when there is Additional Chiplet/Die requirement from Customer.
6) Heterogeneous Die's Tb development: It is not possible to verify two different designs in same database with same testbench.
7) Lack of Maturity in design as it's a relatively novice domain causing repeated design logic update during the course of execution
8) Communication across dies or chiplets.
9) Debug complexity as the simulation across chiplets or dies integrated together.

II. Distributed simulation

Below is the left shift approach that is used or proposed to address all the above mentioned challenges.

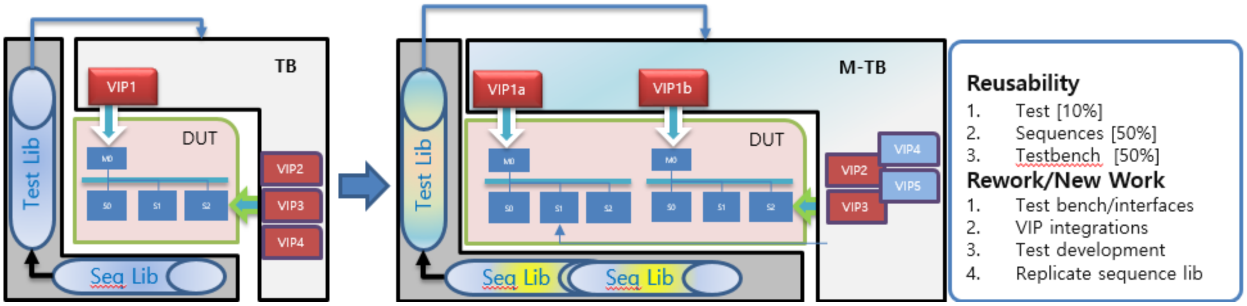


Fig.2 Legacy Multi Die environment

Fig.2 represents the transition from single die environment to conventional multi-die environment for two dies. In summary, Fig.1 describes the below challenges

a) Designs of two dies are instantiated in a single wrapper
b) Two different test benches are developed separately with integrating corresponding VIPs
c) Both the design and test benches get compiled and simulated together under a single simulation
d) As multiple design and test benches integrated together, the overall database size increases. Hence, compilation and simulation requires bigger resource with bigger memory. Simulation run time is also a concern especially for heterogeneous systems where communication need to happen between different simulations involving different DUT and test bench.
e) In addition, the test bench development time and effort increases linearly with increase in number of dies which affects scalability and reusability.

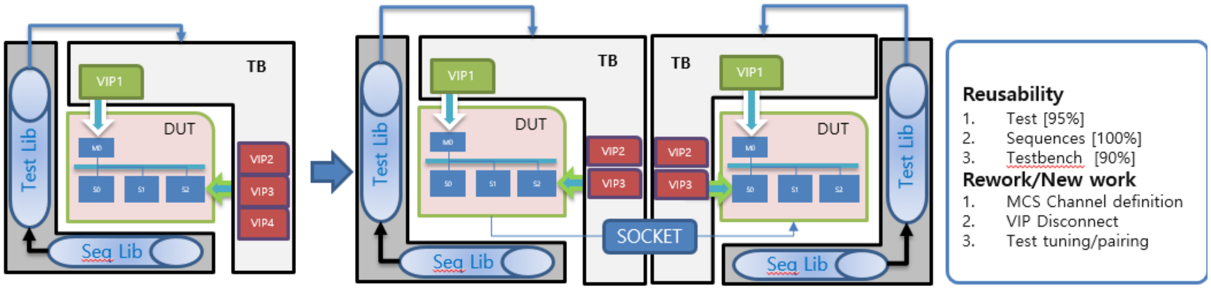


Fig.3 Distributed Sim simulation

Distributed simulation approach can be used instead to address the described challenges in Fig.2. Fig.3 describes the distributed simulation approach and its advantages over conventional approach, which is highlighted below.

a) Simulation per die or chiplet can be independent and parallel compiled and simulated on multiple machines.

b) These simulations can communicate with each other over socket when transactions happen between them. Each simulation behaves as simulation for a single monolithic SoC. Hence, multi-die or multi-chip testbench can leverage from the existing monolithic SOC testbench with incremental update. This reduces the testbench development turnaround time (TAT) considerably or approximately same as the existing monolithic SoC testbench. Experiments show a 10x reduction in testbench development time as compared to conventional single wrapper multi-die or multi-chip approach for a dual die system.

c) As per die or chiplet simulation can happen in separate machine, the compute resource limitation can be overcome as same machines used for monolithic SoC runs can be used for multi-die or multi-chip runs.

d) Distributed Sim provides a scalable testbench solution. The same testbench can be scaled to N number of dies or chiplets as the testbench remains same as per single monolithic SOC with incremental update.

e) The Distributed Sim solution is independent of homogeneous or heterogeneous design. For homogeneous designs, the same testbench can be scaled to multiple dies. For heterogeneous designs, different test benches can be scaled for different designs.

This paper focuses on distributed simulation approach so called Distributed Sim hereafter for addressing multi-die or multi-chip verification challenges.

III. Distributed Sim Methodology

Distributed Sim technology allows multiple independent simulations across homogeneous or heterogeneous designs to communicate with each other mimicking the behavior of a single unified simulation. Fig.4 represents the connectivity between dual chiplets in the Distributed Sim distributed simulation approach

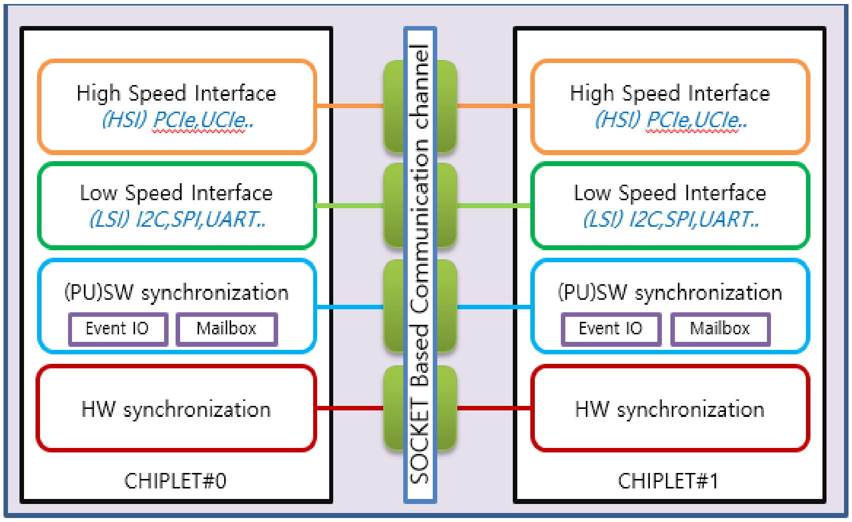
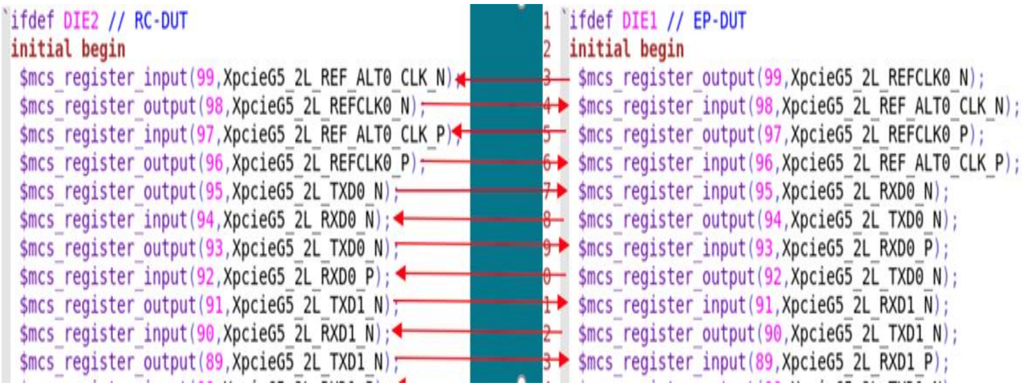


Fig.4 DISTRIBUTED SIM Connectivity

As shown in fig.4, the two chiplets simulating on two independent servers communicate via socket through shared memory. The socket needs to be configured based on the type of transaction between chiplets or dies. As shown in fig.3, the communication between chiplets can be through high-speed interfaces like UCIE or PCIE, low speed interfaces like SPI, UART or I2C, unidirectional or bidirectional synchronization through mailboxes or IO events and other timer based hardware synchronizations.

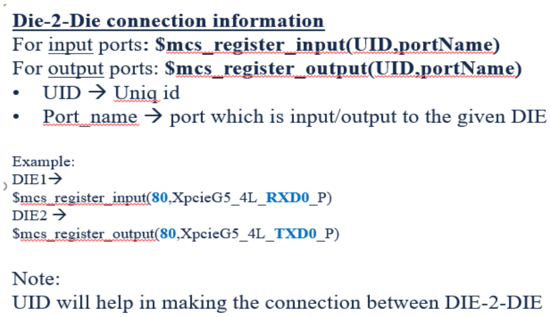


```
`ifdef DIE2 // RC-DUT
initial begin
 $mcs_register_input(99,XpcieG5_2L_REF_ALT0_CLK_N);
 $mcs_register_output(98,XpcieG5_2L_REFCLK0_N);
 $mcs_register_input(97,XpcieG5_2L_REF_ALT0_CLK_P);
 $mcs_register_output(96,XpcieG5_2L_REFCLK0_P);
 $mcs_register_output(95,XpcieG5_2L_TXD0_N);
 $mcs_register_input(94,XpcieG5_2L_RXD0_N);
 $mcs_register_output(93,XpcieG5_2L_TXD0_N);
 $mcs_register_input(92,XpcieG5_2L_RXD0_P);
 $mcs_register_output(91,XpcieG5_2L_TXD1_N);
 $mcs_register_input(90,XpcieG5_2L_RXD1_N);
 $mcs_register_output(89,XpcieG5_2L_TXD1_N);
```

```
`ifdef DIE1 // EP-DUT
initial begin
 $mcs_register_output(99,XpcieG5_2L_REFCLK0_N);
 $mcs_register_input(98,XpcieG5_2L_REF_ALT0_CLK_N);
 $mcs_register_output(97,XpcieG5_2L_REFCLK0_P);
 $mcs_register_input(96,XpcieG5_2L_REF_ALT0_CLK_P);
 $mcs_register_input(95,XpcieG5_2L_RXD0_N);
 $mcs_register_output(94,XpcieG5_2L_TXD0_N);
 $mcs_register_input(93,XpcieG5_2L_RXD0_P);
 $mcs_register_output(92,XpcieG5_2L_TXD0_N);
 $mcs_register_input(91,XpcieG5_2L_RXD1_N);
 $mcs_register_output(90,XpcieG5_2L_TXD1_N);
 $mcs_register_input(89,XpcieG5_2L_RXD1_P);
```

Fig.5 example of communication file

Fig.4 depicts the socket configuration between two chiplets communicating over high speed interface PCIE. This configuration file contains chiplet based input output port information for PCIE protocol. The details of the connectivity syntax are explained below.



Die-2-Die connection information

For input ports: **$mcs_register_input(UID,portName)**

For output ports: **$mcs_register_output(UID,portName)**

- UID → Uniq id
- Port_name → port which is input/output to the given DIE

Example:

DIE1→

$mcs_register_input(**80**,XpcieG5_4L_**RXD0**_P)

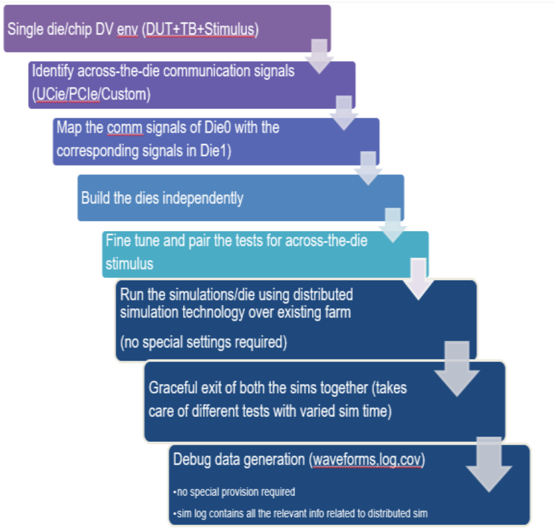
DIE2 →

$mcs_register_output(**80**,XpcieG5_4L_**TXD0**_P)

Note:

UID will help in making the connection between DIE-2-DIE

As shown above, input port XpcieG5_4L_RXD0_P of DIE1 is connected to output port Xpcie_4L_TXD0_P of DIE2. For bidirectional ports $mcs_register_inout(UID,portName) gets used.



Single die/chip DV env (DUT+TB+Stimulus)
Identify across-the-die communication signals (UCie/PCIe/Custom)
Map the comm signals of Die0 with the corresponding signals in Die1)
Build the dies independently
Fine tune and pair the tests for across-the-die stimulus
Run the simulations/die using distributed simulation technology over existing farm
(no special settings required)
Graceful exit of both the sims together (takes care of different tests with varied sim time)
Debug data generation (waveforms,log,cov)
•no special provision required
•sim log contains all the relevant info related to distributed sim

Fig.6 Distributed Sim execution flow for dual die

Fig.6 represents the implementation flow for distributed simulation approach for Multi-Die/Multi-Chip verification.

A. The monolithic single die design verification environment is taken. This contains single DUT instance and corresponding test bench
B. Based on the type of communication between dies or chiplets, the port information of corresponding IP is identified. For Ex. if cross chiplet communication happens over PCIE, the ports of PCIE IP have to be considered
C. Port mapping between dies is done through UID and $mcs_register as shown in Fig.4
D. The dies are then compiled independently along with mapping files
E. The required single die stimulus is then fine-tuned for cross die simulation. Either the same stimulus can be used for all dies or different stimulus can be used per die.
F. The simulation per die is getting run using distributed simulation technology over existing farm by providing simple option to the simulator to link the multiple simulations per die.
G. The simulations can run in lockstep mode with zero latency between runs or in asynchronous mode with specific latency. All simulations exit gracefully taking into consideration different simulation time
H. All debug information including waves, coverage and logs remain same as single die simulation.

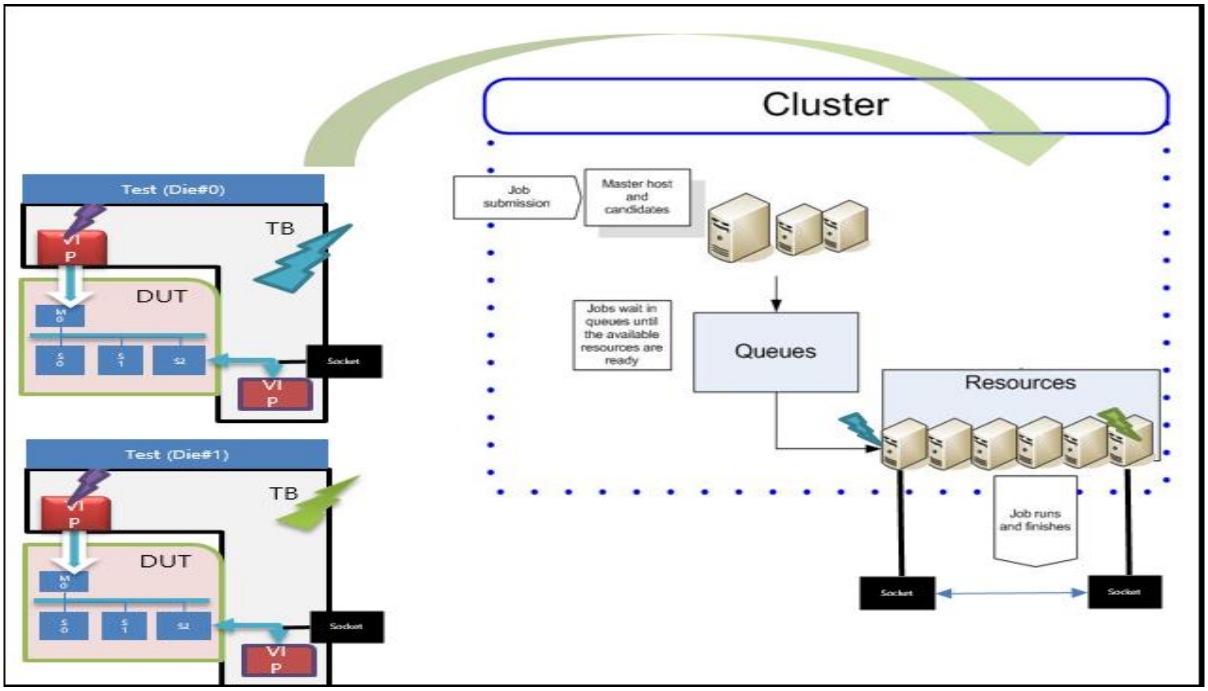
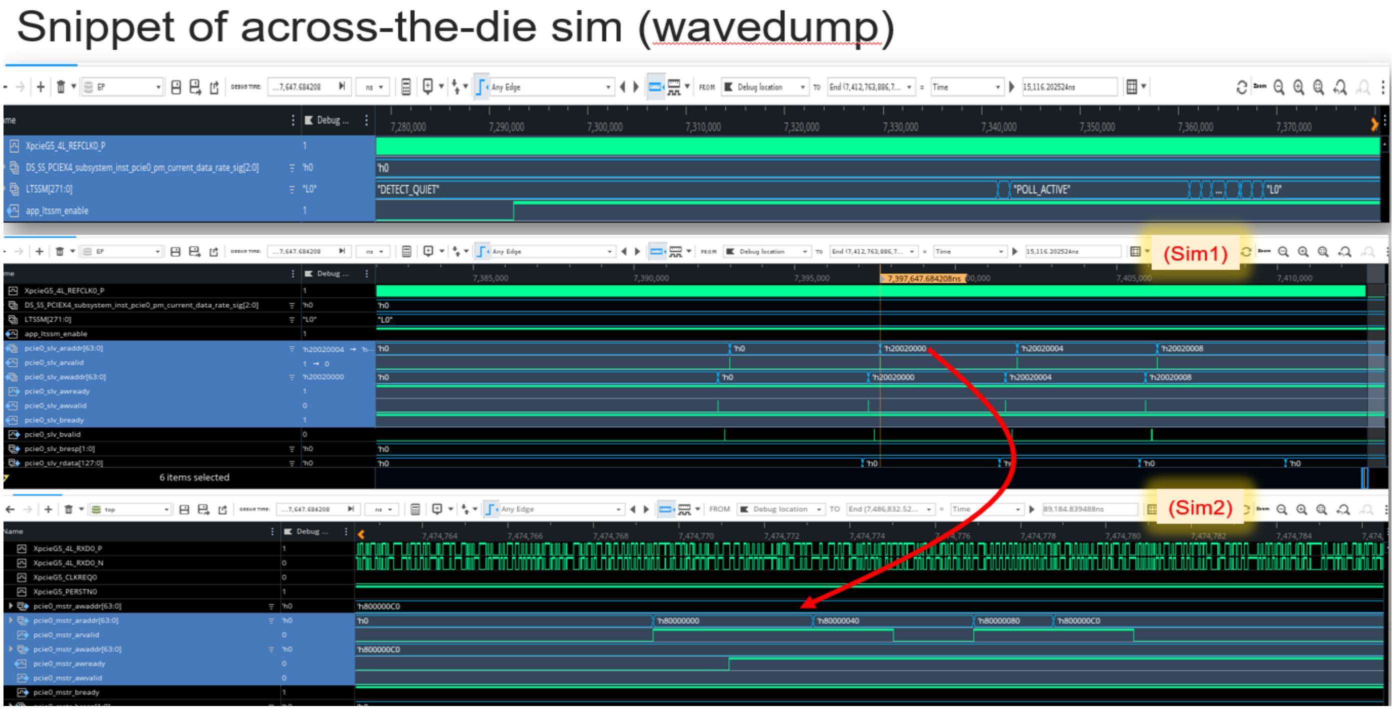


Fig.7 Distributed Sim verification environment

Fig.7 represents overall verification environment for distributed simulation approach applied to dual dies. The details are summarized below.

a) The same single die test bench involving same dut and vip (virtual IPs) is used for both die0 and die1(Multi-Die).
b) The different single die test bench involving different dut and vip can also be used for die0 and die1(Multi-Die1)
c) Two separate simulations are initiated using distributed simulation technology approach
d) The simulations wait in queue to get allocated to a server.
e) Using socket, each simulation has server information of other simulation where the run gets launched
f) Once the server gets allocated to both runs, the simulations communicate with each other through socket.
g) The two simulations wait for each other once both gets completed and then exit gracefully.

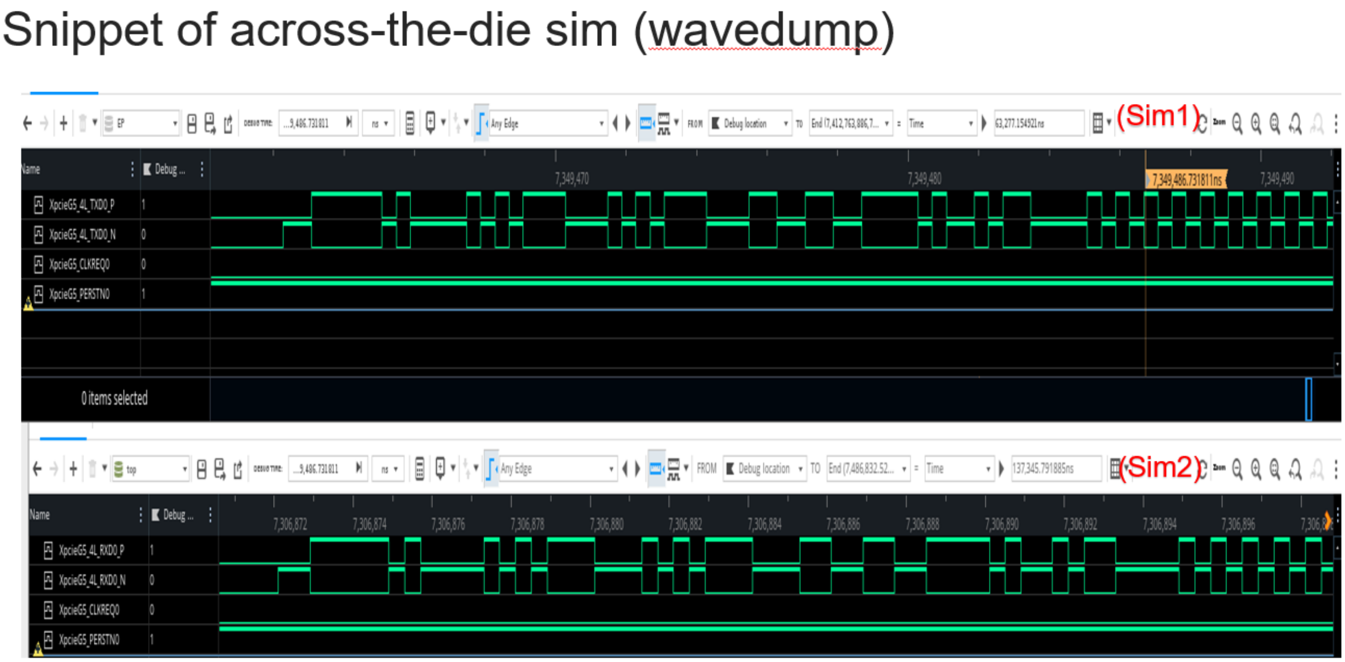
IV. Wave Analysis



Snippet of across-the-die sim (wavedump)

Fig.8 Dual Die simulation

Fig. 8. shows the wave snippet for simulations of both chiplets. “SIM1” represents the wave of CHIP0 simulation and “SIM2” represents the wave for CHIP1 simulation. The red arrow mark on the snippet represents sync between both simulations through master and save ports.



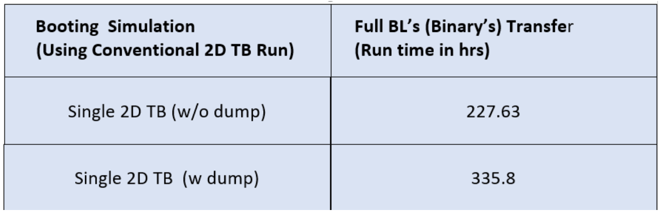
Snippet of across-the-die sim (wavedump)

Fig.9 communication signal dump

Fig.9 shows the wave snippet of common communication ports between dies required for linking both simulations. Here, “SIM1” represents CHIP0 simulation whereas “SIM2” represents CHIP1 simulation. The snippet shows RX and TX port behavior from both simulations.

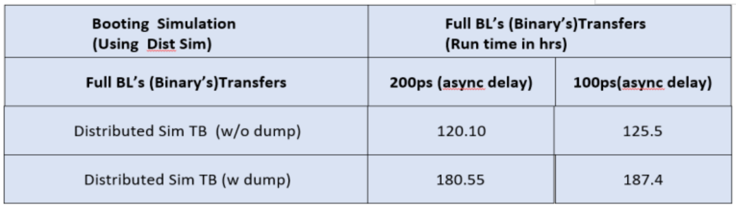
V. Results

TABLE I
Distributed Sim result_1 on 2-Die Automotive SoC



Booting Simulation (Using Conventional 2D TB Run)	**Full BL's (Binary's) Transfer (Run time in hrs)**
Single 2D TB (w/o dump)	227.63
Single 2D TB (w dump)	335.8

TABLE II
Distributed Sim result_2 on 2-Die Automotive SoC



Booting Simulation (Using Dist Sim)	**Full BL's (Binary's)Transfers (Run time in hrs)**	
Full BL's (Binary's)Transfers	**200ps (async delay)**	**100ps(async delay)**
Distributed Sim TB (w/o dump)	120.10	125.5
Distributed Sim TB (w dump)	180.55	187.4

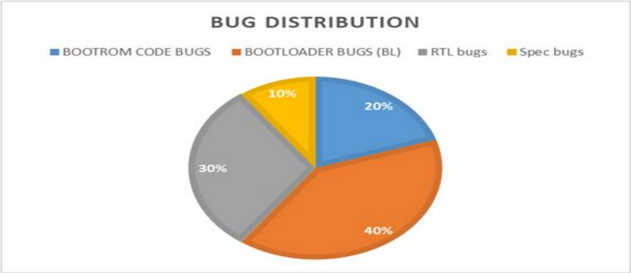


Fig.10 Results of adopting Distributed Sim approach

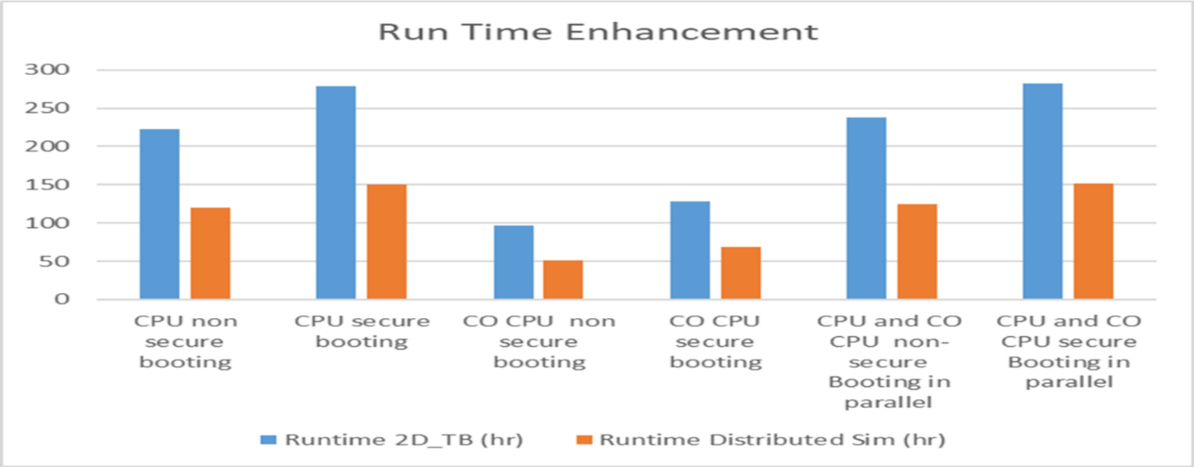


Fig.11 Results of adopting Distributed Sim approach

Abbreviations and Acronyms

Refer to the mentioned abbreviations used throughout the paper.

SOC (System On Chip), DUT (Design Under Test), CPU (Central Processing Unit), VIP (Virtual Intellectual Property), LSF (Load Sharing Facility), AI (Artificial Intelligence), IC (Integrated Circuit), UID (Unique Identification), TB (Test Bench)

VI. CONCLUSION

(i) Over all 90%, DV effort reduced in setting up TB for different DIEs

(ii) Completion of execution with critical bug findings in very less time

(iii) DISTRIBUTED SIM provides a promising methodology to verify MULTIDIE BOOTROM with reduced development

(iv) Simulation time leading to early bug hunting and on time closure of verification

REFERENCES

[1] K. A. Bowman, A. R. Alameldeen, S. T. Srinivasan and C. B. Wilkerson, "Impact of Die-to-Die and Within-Die Parameter Variations on the Clock Frequency and Throughput of Multi-Core Processors," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 12, pp. 1679-1690, Dec. 2009

[2] S. S. Rao, B. Thangavelu, S. C. P, S. Joshi, A. Kumar and G. Srivastava, "A Safe, Secure and Coherent Multi-chip Architecture for ISO Compliant Automotive Solutions," 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 2025, pp. 1-6.

[3] K. Zhou, C. Sha and Y. Liu, "New Generation Test Framework Solution for Complicated Multi-Die Chip on ATE," 2024 Conference of Science and Technology for Integrated Circuits (CSTIC), Shanghai, China, 2024, pp. 1-3.

[4] H. Löhr, A. -R. Sadeghi and M. Winandy, "Patterns for Secure Boot and Secure Storage in Computer Systems," 2010 International Conference on Availability, Reliability and Security, Krakow, Poland, 2010, pp. 569-573.

[5] S. Penta et al., "Performance Evaluation of UCIe-based Die-to-Die Interface on Low-Cost 2D Packaging Technology," 2024 IEEE 74th Electronic Components and Technology Conference (ECTC), Denver, CO, USA, 2024, pp. 274-278.

[6] A. Saiki, Y. Omori and K. Kimura, "Parallel Verification in RISC-V Secure Boot," 2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Singapore, 2023, pp. 568-575.

[7] Chang Yifeng and Yang Yintang, "Application and simulation technology of multichip module packaging," Proceedings. 2005 International Conference on Communications, Circuits and Systems, 2005., Hong Kong, China, 2005, pp. 1214.