Expediting UCIe and Boot Verification using Distributed Ndie Simulations and Emulation for Multi-Chiplet AI SoCs

Harshal Kothari (Samsung Semiconductor India Research, Bengaluru, India)

Jerin M Jose (Samsung Semiconductor India Research, Bengaluru, India)

Jasobanta Sahoo (Samsung Semiconductor India Research, Bengaluru, India)

Ayush Agrawal (Samsung Semiconductor India Research, Bengaluru, India)

Madhukar Ramegowda (Samsung Semiconductor India Research, Bengaluru, India)

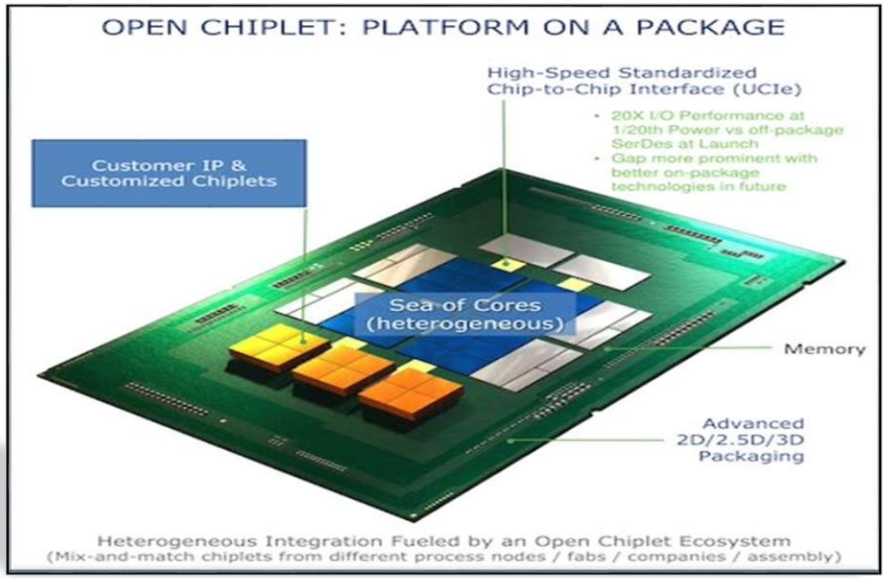
62 THE CHIPS TO SYSTEMS CONFERENCE

SAMSUNG

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

Mo’tivation: Mo’ size, mo’ problems?

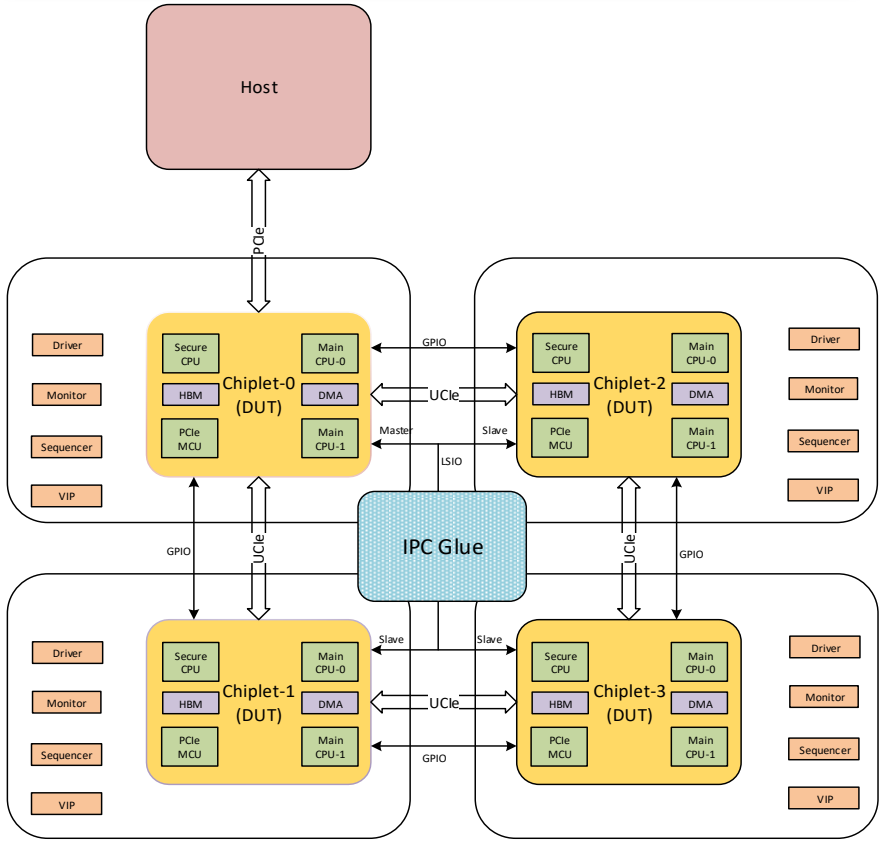
- Moore predicted “Day of reckoning” in 1965: *“It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected.”*
- Size of monolithic SoCs for Generative AI, hyperscalers & enterprise grade data-centre applications is becoming too big for manufacturability
- SoC functionalities may require different process nodes for optimal implementation
- Needed standardized die-to-die interconnects



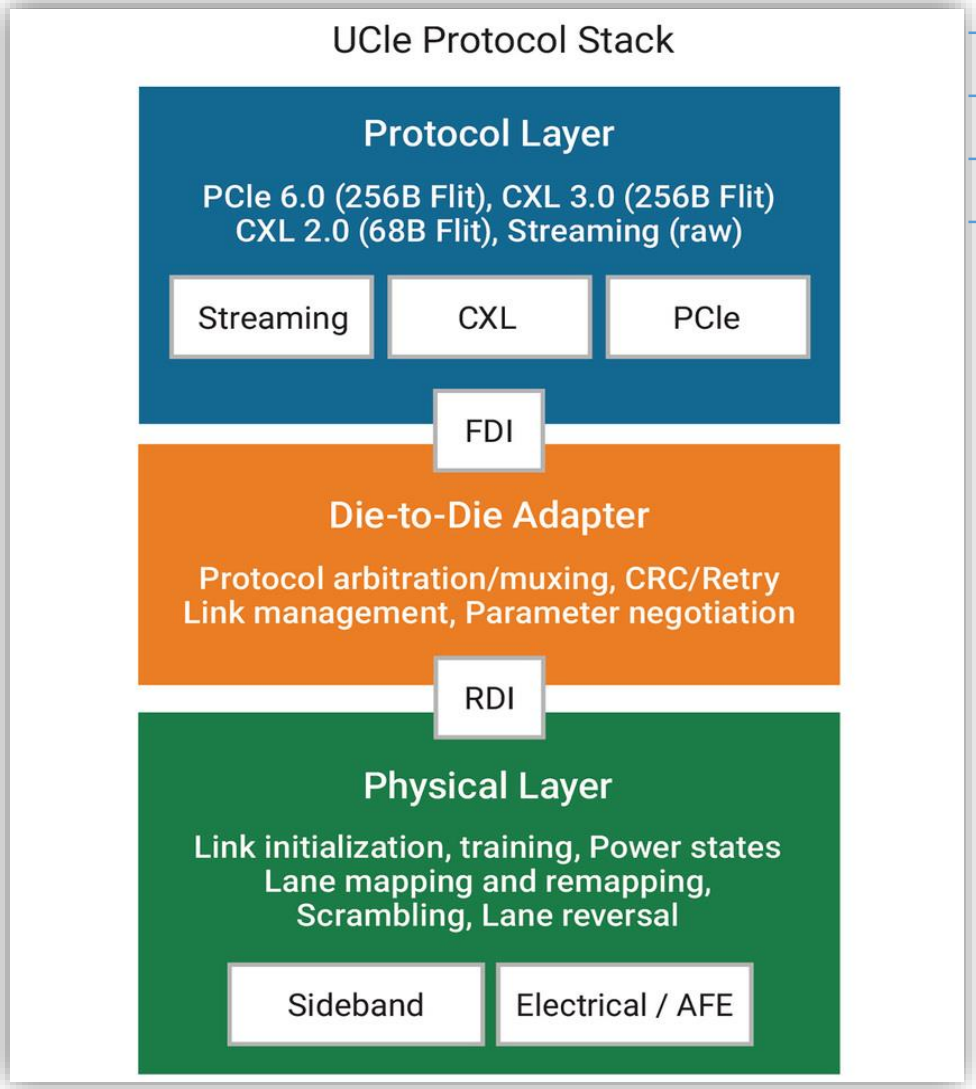
UCIe
Universal Chiplet Interconnect Express

Advantages of UCIe:

- Full stack interconnect integrating disintegrated dies
- Open industry standard - defines a complete solution for die-to-die interconnect, ensuring interoperability of compliant devices
- 130+ companies part of consortium- UCIe Gen 2.0 released
  - Broader reach than XSR/USR/BOW/OHBI/AIB
- High bandwidth, low latency, low power
- Supports multiple protocols
  - PCIe, CXL, Raw, Streaming



UCIe Subsystem Architecture

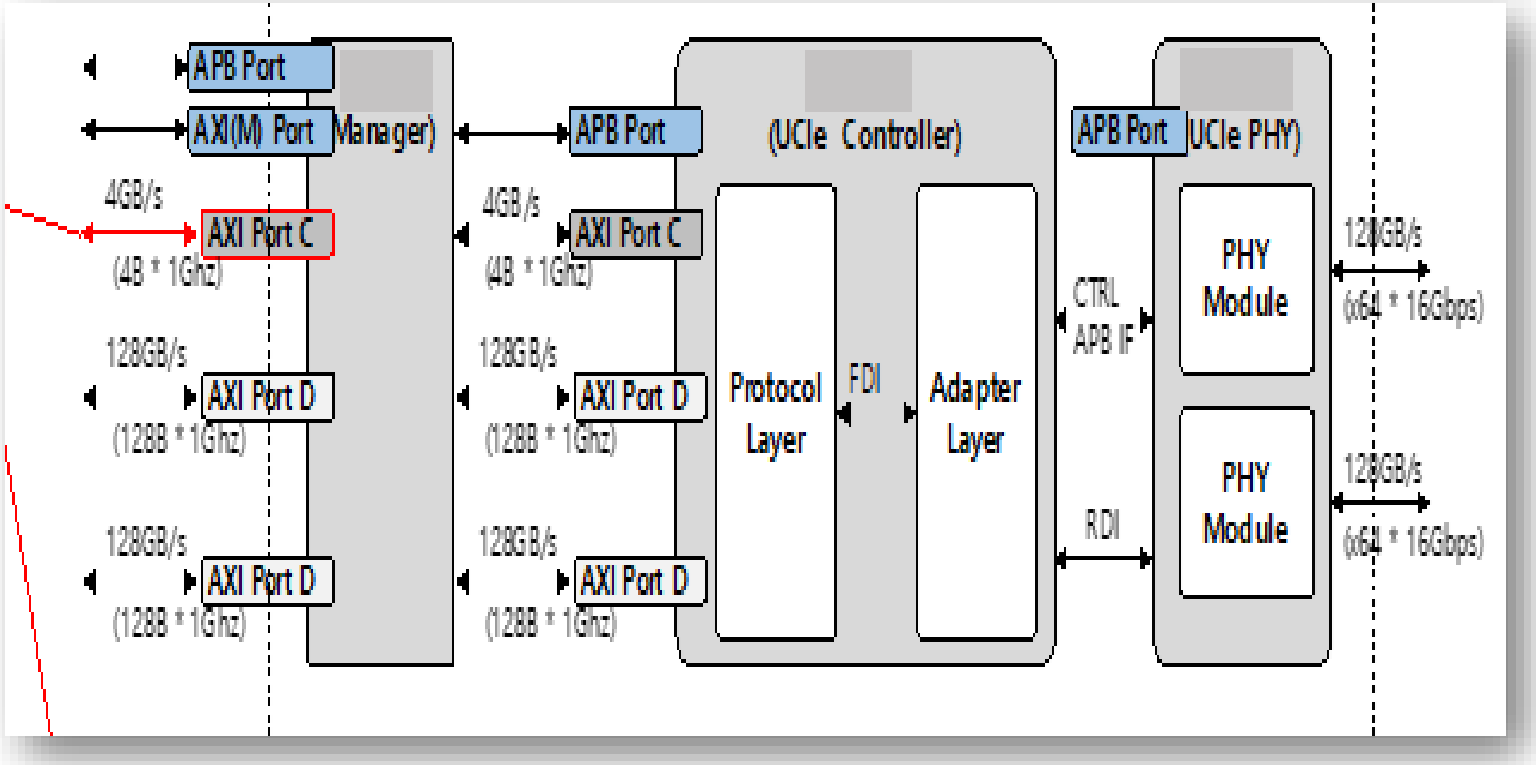


UCIe 1.1 Multi-module (2 PHY) Advanced Package Physical Layer on 2.5D IC

Tx and Rx x64 lanes (x2 PHY modules instances)

AXI4 x 6 upper layer with UCIe Streaming Protocol (1GHz)

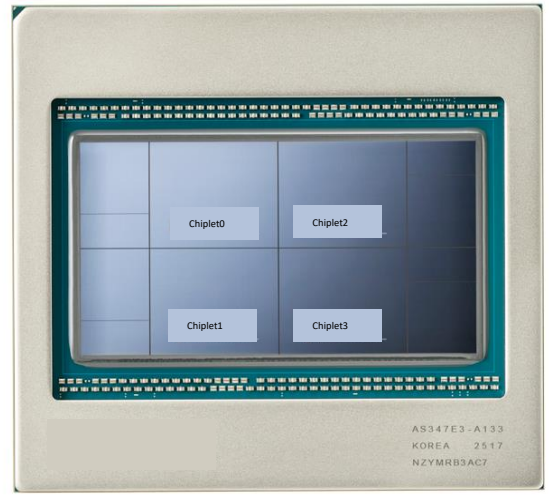
16Gbps per lane → Total Tx/Rx @ 2Tbps (bidir)



Why go distributed?

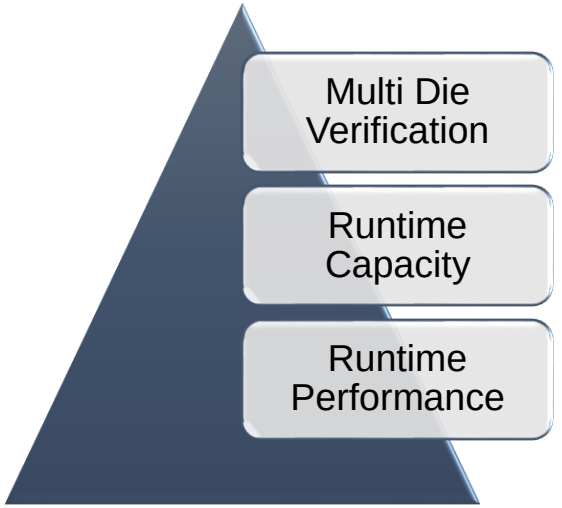
Challenge of scalability, performance while executing multi-die AI SoC design cycles efficiently

- Roadblocks for DV engineers:
  - Fitting the env into existing/limited resources (compute, memory, etc.)
  - Performance while simulating the usecases of inter-chiplet communication (high sim time for complex, across-the-die mission mode datapath tests ~300hrs)
  - Interposer design starts around die freeze stage
  - Verifying parallel LLM operations ranging 2 PFLOPS (FP8) compute across UCIe

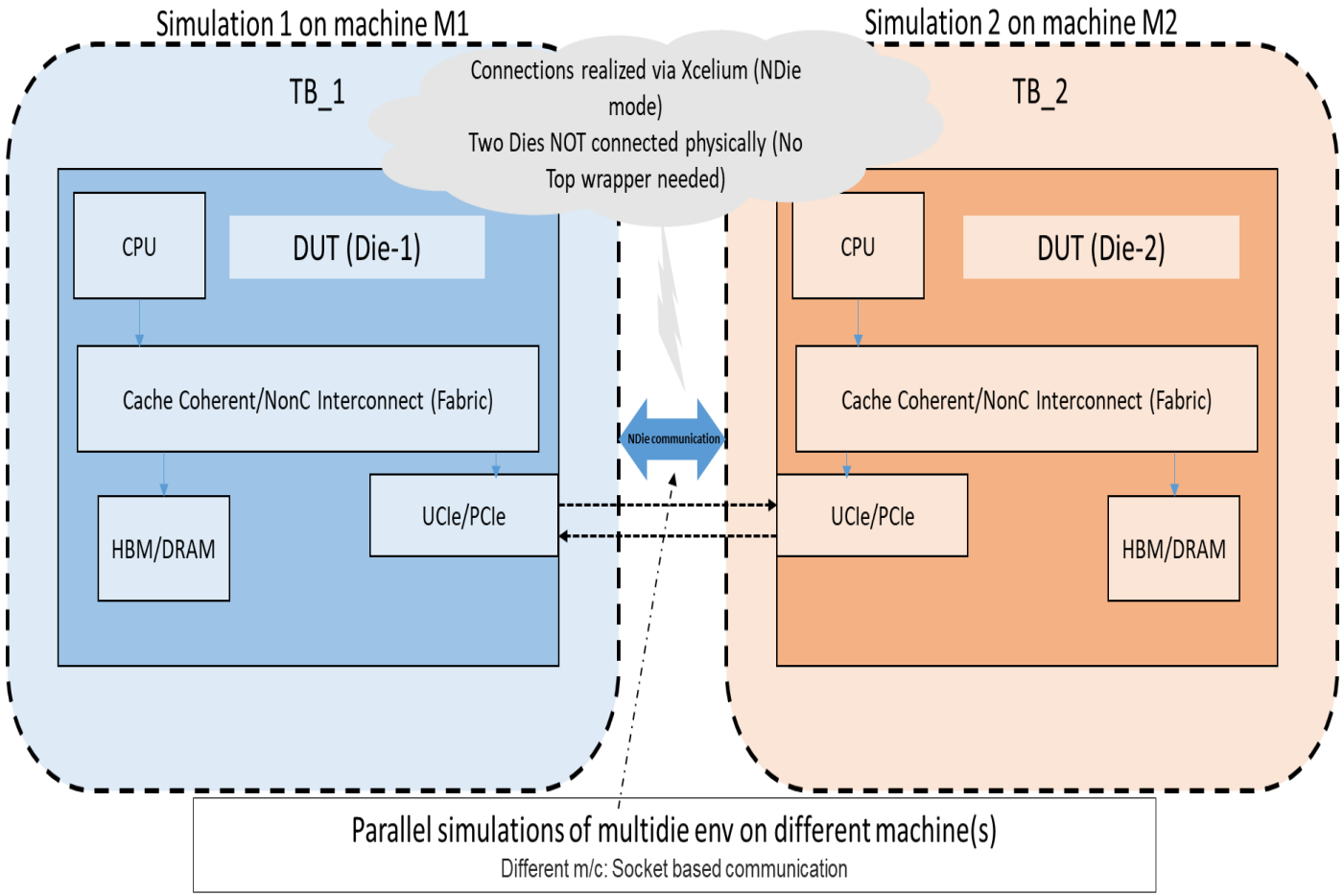


Ndie solution aims to cater to these challenges by:

- Running multi-die sims in parallel on separate resources/machines while keeping the functionality/intent intact
  - With sim across 2 or more dies getting executed in parallel on separate machines, it helps utilize the resources better
- Minimizing the efforts of DV engineer in creating a single env instantiating 2 or more dies to realize across-the-die scenarios
  - Since most of the multidie env have same design/TB for most of the dies, a simple config file can help the user connect the 2 env together w/o having to generate a top wrapper



Main Idea: Distributed Simulations Overview

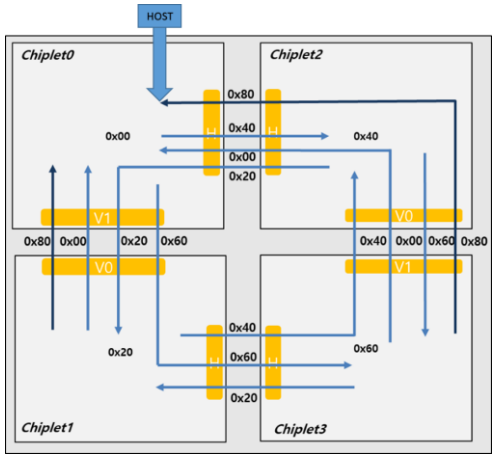



```
23 `ifdef DIE0                                                    1292 `ifdef DIE1
24 $mcs_register_output(638,top.dut.XUCIEV10_bu_m0_txio0); // DIE-0-1-2-3   1293 $mcs_register_input(638,top.dut.XUCIEV01_bu_m0_rxio63); // DIE-0-1-2-3
25 $mcs_register_output(639,top.dut.XUCIEV10_bu_m0_txio1); // DIE-0-1-2-3   1294 $mcs_register_input(639,top.dut.XUCIEV01_bu_m0_rxio62); // DIE-0-1-2-3
26 $mcs_register_output(640,top.dut.XUCIEV10_bu_m0_txio2); // DIE-0-1-2-3   1295 $mcs_register_input(640,top.dut.XUCIEV01_bu_m0_rxio61); // DIE-0-1-2-3
```

Key Features:

- Use multiple resources in simulating the multiple dies (one each for each die)
- Die-Die connection realized via tool config file (mcs)
- No need to create any wrapper for the two dies
- Helps reuse the same env for both the dies (Reduces TB development time from 3-4 weeks ➔2 days)
- Simple (extra) options to setup the env for NDie mode at elab and sim
- No need for any special settings to leverage the existing farm (LSF)
- No need for bigger memory/compute machines (4TB machines 11x costlier than 500G machines)

Main Idea: Distributed Simulations Implementation



3 parallel testbenches to thoroughly verify industry-first implementation:

1. UCIe with VIP (3rd party EDA Vendor)
2. UCIe B2B on single die (via external loopback)
3. Quad die testbench (final product DUT on single interposer)

NDIE is used to simulate long running sims from 2 & 3.
Experimentation showed that 50 ps async communication over the socket gave best performance

4 Homogeneous Chiplets in 1 package with SF4x

6 BLKs with UCIeA 1.1 subsystem (16Gbps) instance per die (total 24)

2 BLKs per die for AI/ML workload: 8 neural cores in each (total 64)

36GB Samsung HBM3E (9.6Gbps) per die (total 144GB)

Secure Root-of-trust boot processor and dual cluster 8 core CA73 per die

Package connects to Host (PCIe RC) via PCIe Gen5 EP

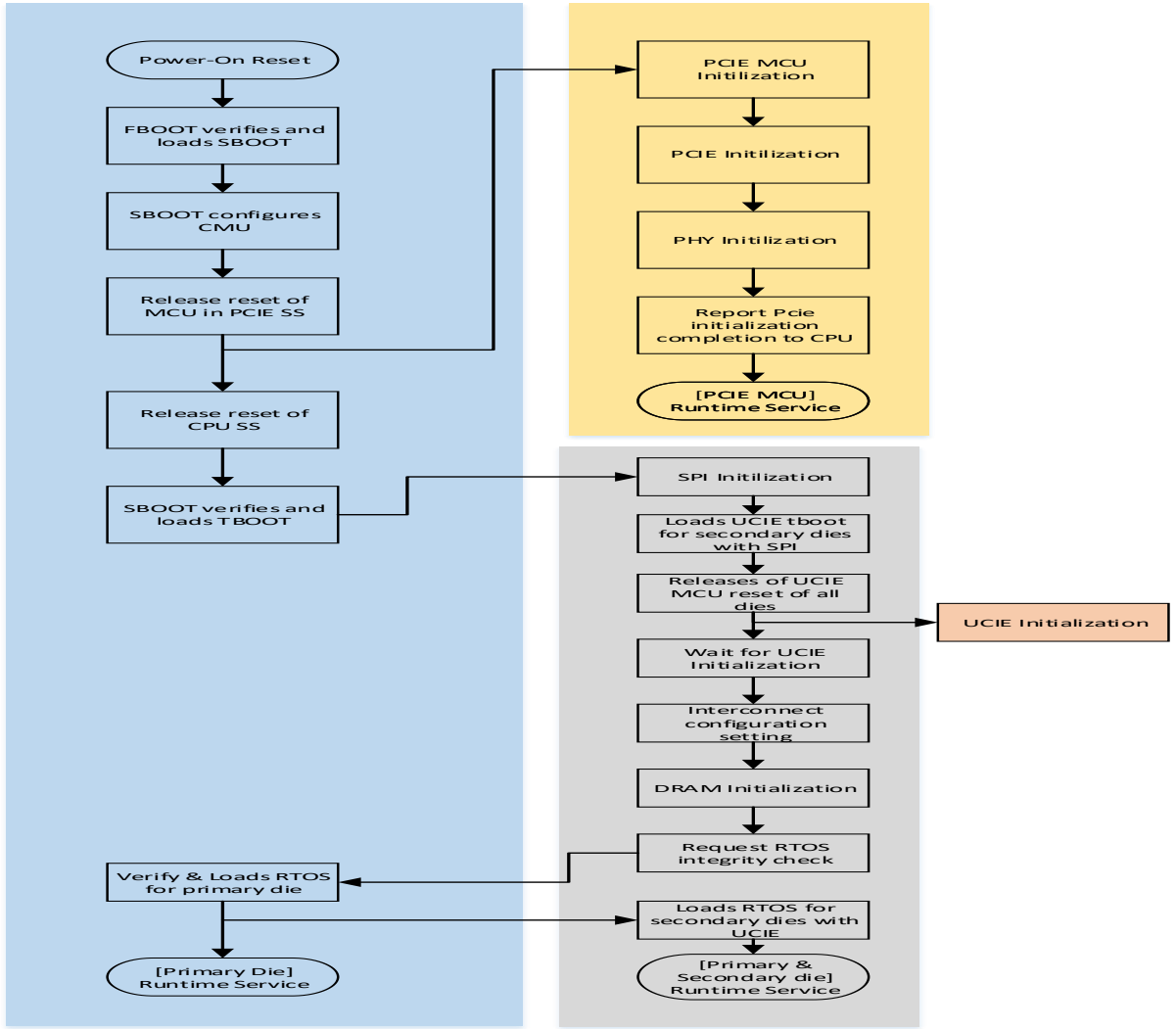
UCIe Sim	Runtime (hrs)	
(LTSM bypass init D2D HBM/MMIO transfers)	B2B	NDie 50ps
LSF (w/o dump)	10:14	4:39
LSF (w dump)	16:14	7:34
DM (w/o dump)	9:36	4:44
DM (w dump)	14:53	7:47

UCIe Sim	Runtime w Async Delay (hrs)				
(LTSM bypass init D2D HBM/MMIO transfers)	10ps	50ps	100ps	150ps	200ps
LSF (w/o dump)	10:14	4:39	5:00	4:44	5:56
LSF (w dump)	16:14	7:34	13:26	12:32	13:25
DM (w/o dump)	8:36	4:44	4:00	4:10	4:12
DM (w dump)	14:53	7:47	6:37	6:29	6:28

```
3 host name         : dspr0692
4 operating system  : Linux 4.18.0-372.75.1.el8_6.x86_64 #1 SMP Thu
5 TOOL: xrun(64)  24.06-a071-20240617: Started on Aug 19, 2024 at 19
6 xrun
7    -mcs_sim_index 0
8    -mcs_comm socket:@/user/jerin.jose21/mcs_temp_new_die.txt
9    -mcs_sim_count 4
10   -setenv MCS_ASYNC=50ps
11   -define DIE0
12   -define NDIE
13   -no_sharedlock2
e_with_dump/test-cpustub_rbc-soc_test_c-seq=soc_vseq_c=ucie_ltsm_bring
```

```
mcs: Running SIM 0 in delay async mode (50000 fs)
mcs [0]: produced socket-address-token @/user/jerin.jose21/mcs_temp_new_die.txt.0: 11.105.1.195:56391
mcs [0][1]: outputs: 624 vars, 624 comm-ids
mcs [0][1]: inputs: 624 vars, 0 comm-ids
mcs [0][1]: inouts: 0 vars
mcs [0][1]: num of used input vars=0 num of used output vars=0 num of used inouts of partner=0 num of used inouts by partner=0
mcs [0][2]: outputs: 624 vars, 624 comm-ids
mcs [0][2]: inputs: 624 vars, 312 comm-ids
mcs [0][2]: inouts: 0 vars
mcs [0][2]: num of used input vars=312 num of used output vars=312 num of used inouts of partner=0 num of used inouts by partner=0
mcs [0][3]: outputs: 624 vars, 624 comm-ids
mcs [0][3]: inputs: 624 vars, 312 comm-ids
mcs [0][3]: inouts: 0 vars
mcs [0][3]: num of used input vars=312 num of used output vars=312 num of used inouts of partner=0 num of used inouts by partner=0
```

Multi Chiplet Boot Flow



Boot Flow:

- The secure boot has three stages FBOOT, SBOOT and TBOOT, each stage bootloader of secure processor subsystem checks the integrity of the next stage bootloader.
- Primary and secondary dies perform boot steps(i.e., fboot, sboot) in parallel
- The primary die brings MCU in PCIe subsystem out of reset and establishes PCIe link.
- The primary die uses the QSPI channel to load UCIe tboot for secondary die and releases reset of the MCU in UCIe subsystem which performs UCIe initialization.
- The normal TBOOT, running by the primary die, proceeds with the rest of the initialization, including DRAM initialization.
- After DRAM initialization, the primary die loads RTOS into the DRAM for all dies.

Emulation setup

Design Env Development

- Design and Test environment development
- Top TB, Clock generation, accessing runtime signals

Compilation

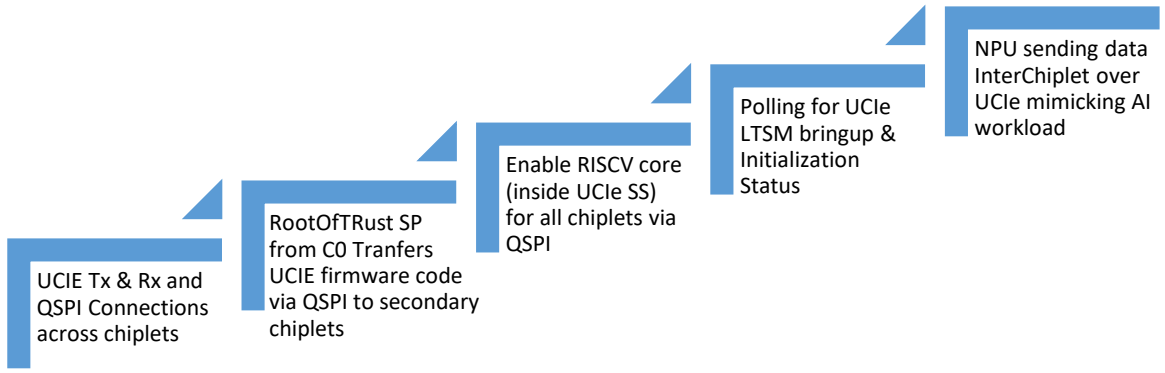
- Compilation setup – CompilersOption , Makefile
- Launching vlan_dut , vlan_tbtop(top file), ixcom compile, xrun tb compilation for front end synthesis, backend compilation and P&R for emulator

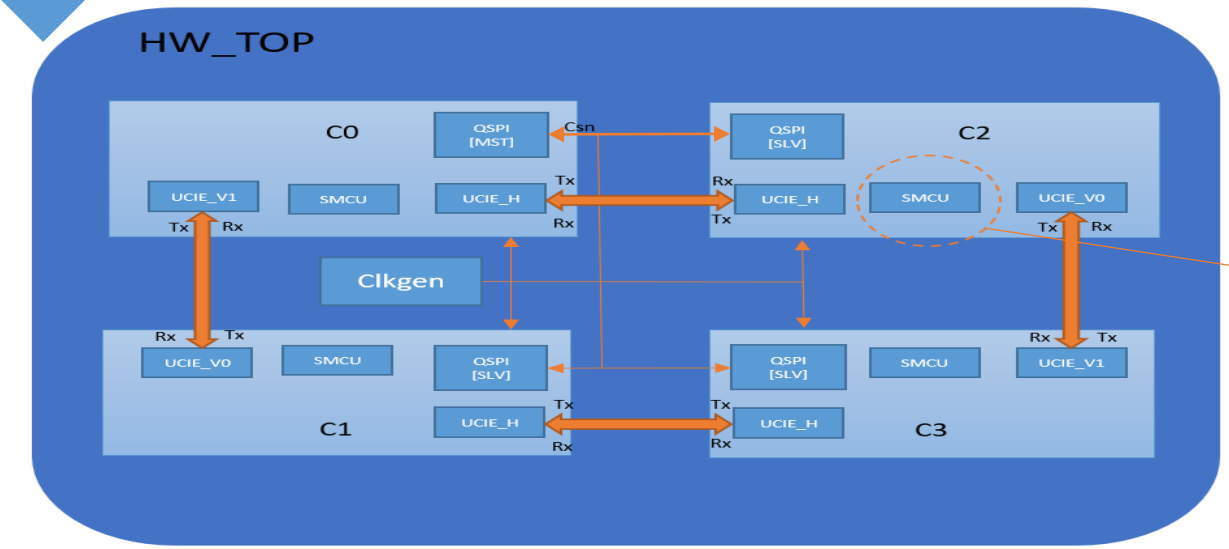
Run Time

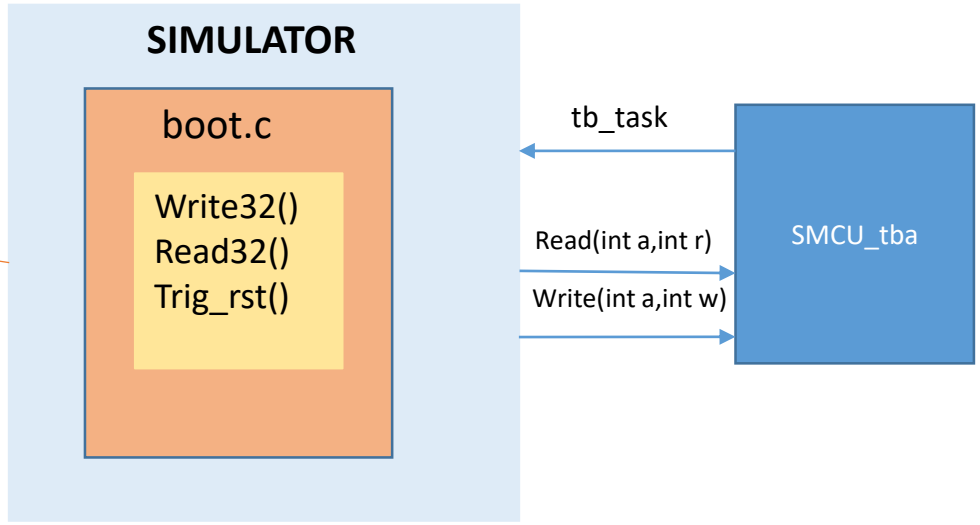
- Loading image onto Emulator
- Reset release of DUT & backdoor loading of memory
- Sanity tests
- Waveform Capture

Two parallel testbenches development for faster bringup :

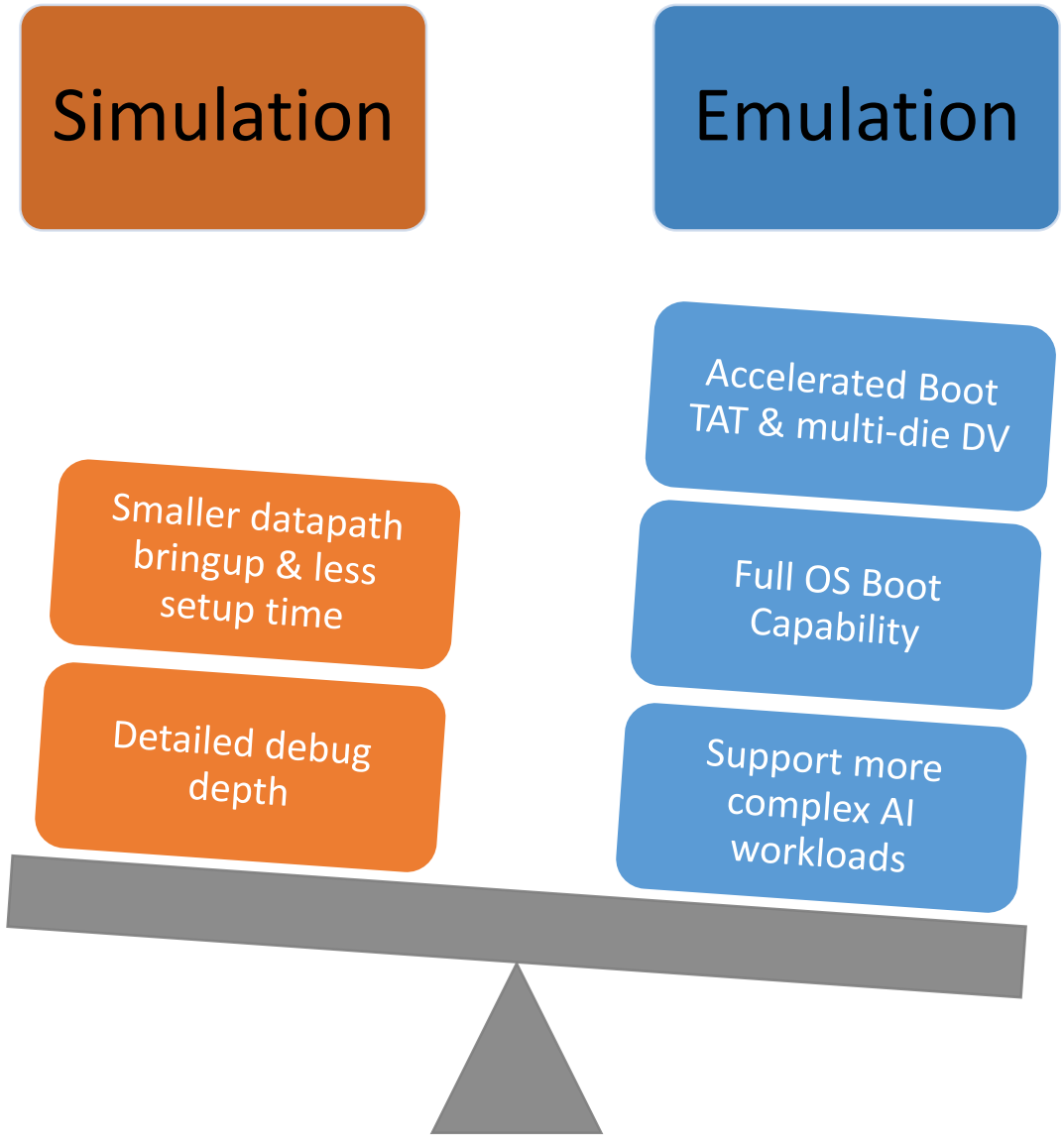
1. Secure Processor replaced with transaction based acceleration for UCIe & interchiplet AI workload transfer verification (No dependency on secure processor firmware)
2. Real Secure Processor based multi-die boot-up







Emulation implementation



- Simulation is useful for early RTL bringup and unit level verification, but struggles with scale and speed for full boot processes
- Emulation bridges that gap by enabling high performance, system level and software aware boot verification
- DUT of 350mm^2 x 4 on SF4x node implemented on PZ3
- 1.63B gate count (With Hybrid emulation) with 18 boards
- Able to emulate entire multi-chiplet boot-up scenarios with concurrently running chiplets
- Parallel image development on Palladium for DV acceleration and ZeBu for SW bringup
- Tremendous savings over Quad die simulation TB:

Product Use-Case Scenario	Quad Die Sim (hrs)	Emulation (mins)
RAM based boot	130	6
OTP based bootloader	96	5
QSPI Flash based bootloader	620	30

Evidence & Results

S.No	Scenario	Runtime QD_TB (hr)	Runtime NDIE (hr)	Improvement
1	UCIe LTSM FW bringup HBM/MMIO 2 die access from Die0 -> 1 (Single Link V10 to V01) - FULL LTSM	118	65	1.82x
2	UCIe LTSM FW bringup HBM/MMIO 2 die access Die3 <-> 2 (Both Link V10, V11 to V01, V00) - FULL LTSM	126	70	1.8x
3	UCIe LTSM FW bringup HBM/MMIO 3 die access Die0 -> 2 -> 3 (All 3-2 and 2-0 links) with chiplet routing update	165	79	~2.2x
4	UCIe LTSM FW bringup HBM/MMIO 4 die access Die0 -> 1 -> 3 -> 2 (All 8 : 0-1, 1-3, 2-3 and 0-2 links) - FAST_SIM	243	82	~3x
5	UCIe LTSM FW bringup HBM/MMIO 4 die access Die0 -> 1 -> 3 -> 2 (All 8 : 0-1, 1-3, 2-3 and 0-2 links) - FULL LTSM	275	92	~3x

Table 1: Run time comparison of NDIE Dis. Sim. v/s Quad Die sim

Emulation was deployed to accelerate real Si use case verification like multi-chip boot scenarios and FW based UCIe LTSM init

```
=================================================================
                Statistics for Overall Performance
=================================================================
Top Level Design Name: xcva_top
Design Utilization: 69.5
Instruction Usage: 75.04
Max emulator operating speed is 646 kHz
Design is scheduled in 1400 steps
The number of domains: 144
The number of boards: 18
Total gate count originally: 1599046400
  --(including memory access instrumentation: 1601800435)
Total gate count after transformations: 1117693238
  --(including memory access instrumentation: 1120444530)
=================================================================
```

Parameters	Simulation	Emulation
Compile build-up time	1hr 10mins	4hrs 29mins
Sequence Generation Time	2hrs	2hrs 30mins
Run time (test duration- 15ms)	165hrs	8mins (>1000x faster!)

Table 2: Run time comparison of Hybrid Emulation v/s Quad Die sim

Summary

- Closed UCIe verification on SF4X: Many IP/SoC bugs found around use-case features (Fig 1)
- Significant performance gain and reduced TAT of multi-die DV
- N-Die Dis. Sim. up to ~3x faster than quad-chiplet DUT run on single xrun/simv thread on single LSF machine
- Verified huge multi-chiplet scenarios with same testbench- 3 months left shift even before interposer design ready
- TB development time reduced from 3-4 weeks to 2 days
- Scalable across n-dies- works like a charm with 4 dies
- $125k saving by avoiding costly big mem/dedicated machines
- >1000x improvement with emulation deployment
- Waveform debug difficult on Dis. Sim.
  AI debug utility development WIP with EDA vendor

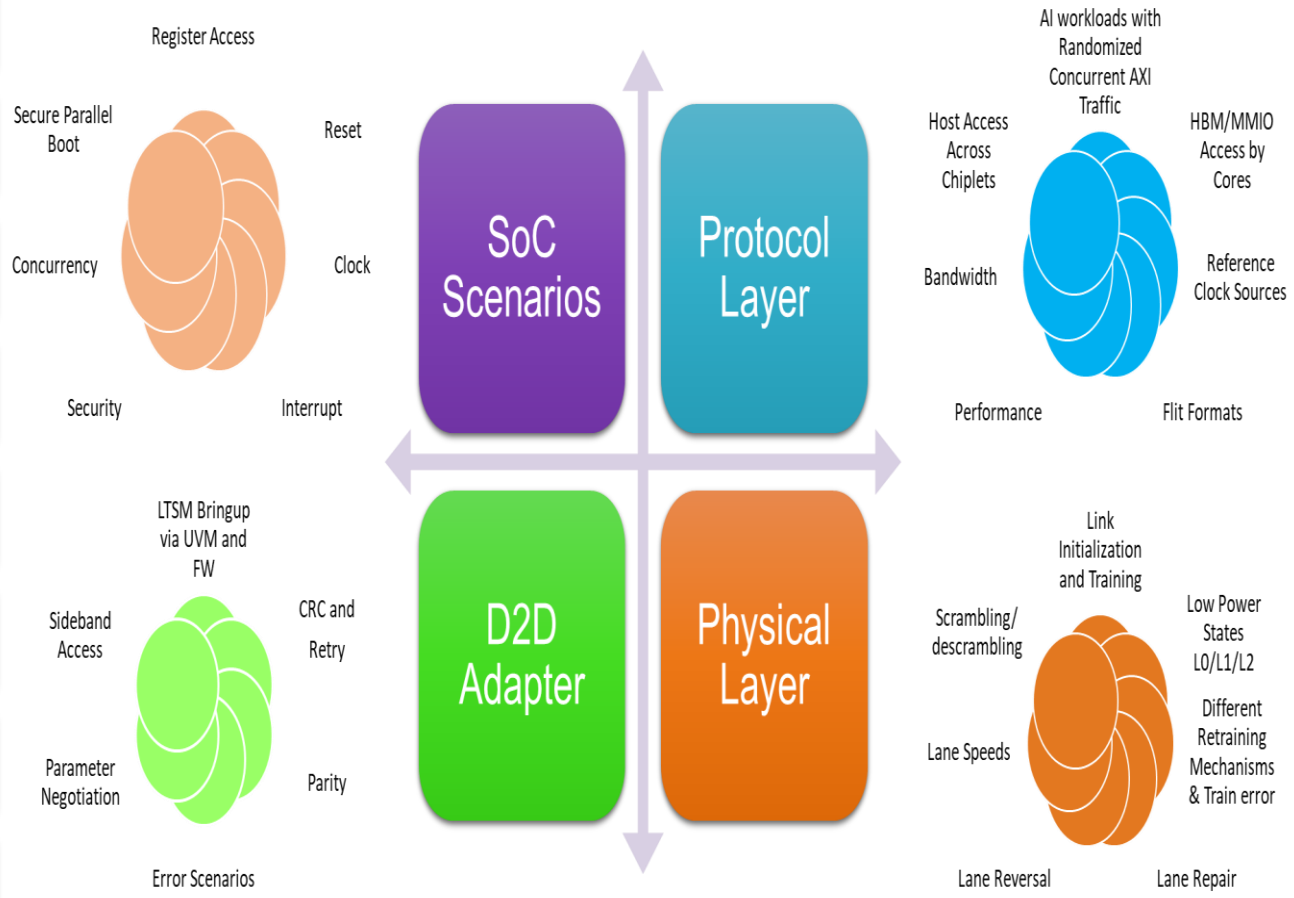


Fig.1: Verified attributes across NDIE Dis. Sim./Emulation