2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

An Accelerated System Level CPU Verification through Simulation-Emulation Co-Existence

Ruchi Misra, Samridh Deva, P Sai Krishna, Alok Kumar, Garima Srivastava
YoungSik Kim, Seonil Brian Choi

SAMSUNG

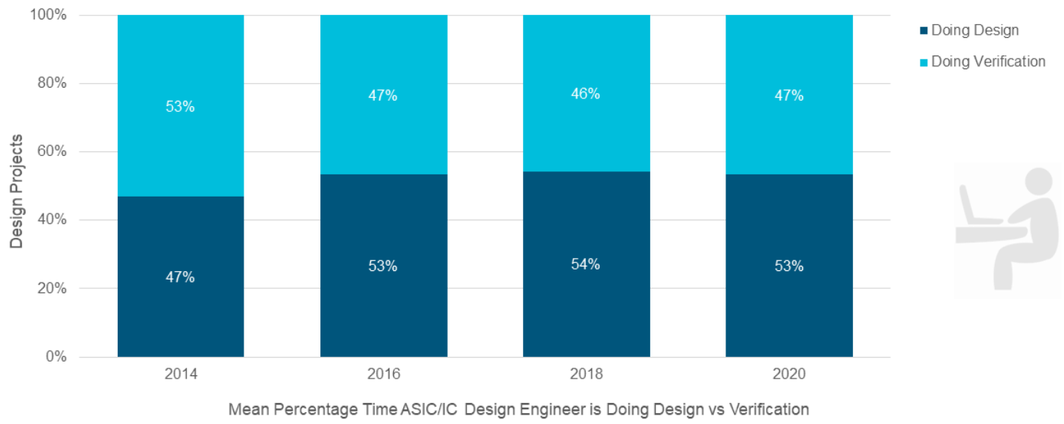
accellera SYSTEMS INITIATIVE

Outline

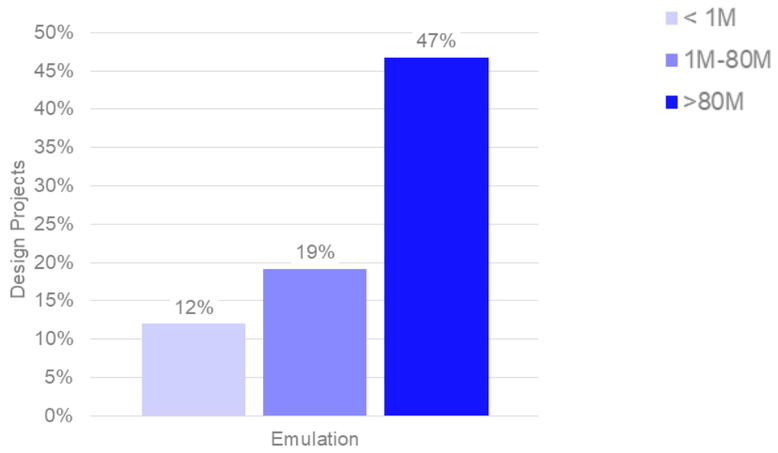
- Need for adoption of Emulation in early Design Verification
- Pre-Silicon Verification flow
- Emulator Compilation flow
- Methodology
- Case Studies
- Enhancements done
- Results
- Conclusion
- Future Scope and Care Abouts
- Acknowledgement

Need for adoption of Emulation in early Design Verification

- Very long simulation time
- Creating corner case scenarios
- Coverage Closure
- Security and Functional Safety
- Low Power Verification
- Huge Design Complexity
- Large T-A-T for iterations

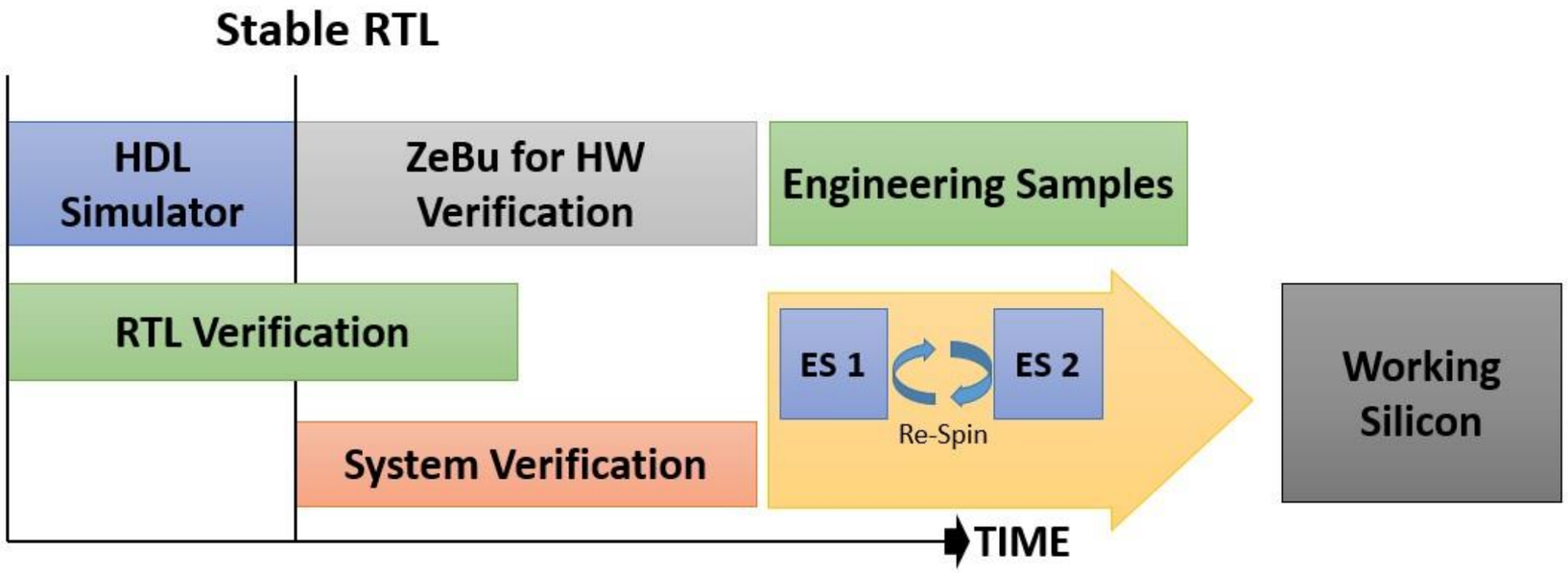


Mean Percentage Time ASIC/IC Design Engineer is Doing Design vs Verification



Source: Wilson Research Group and Mentor, A Siemens Business, 2020 Functional Verification Study

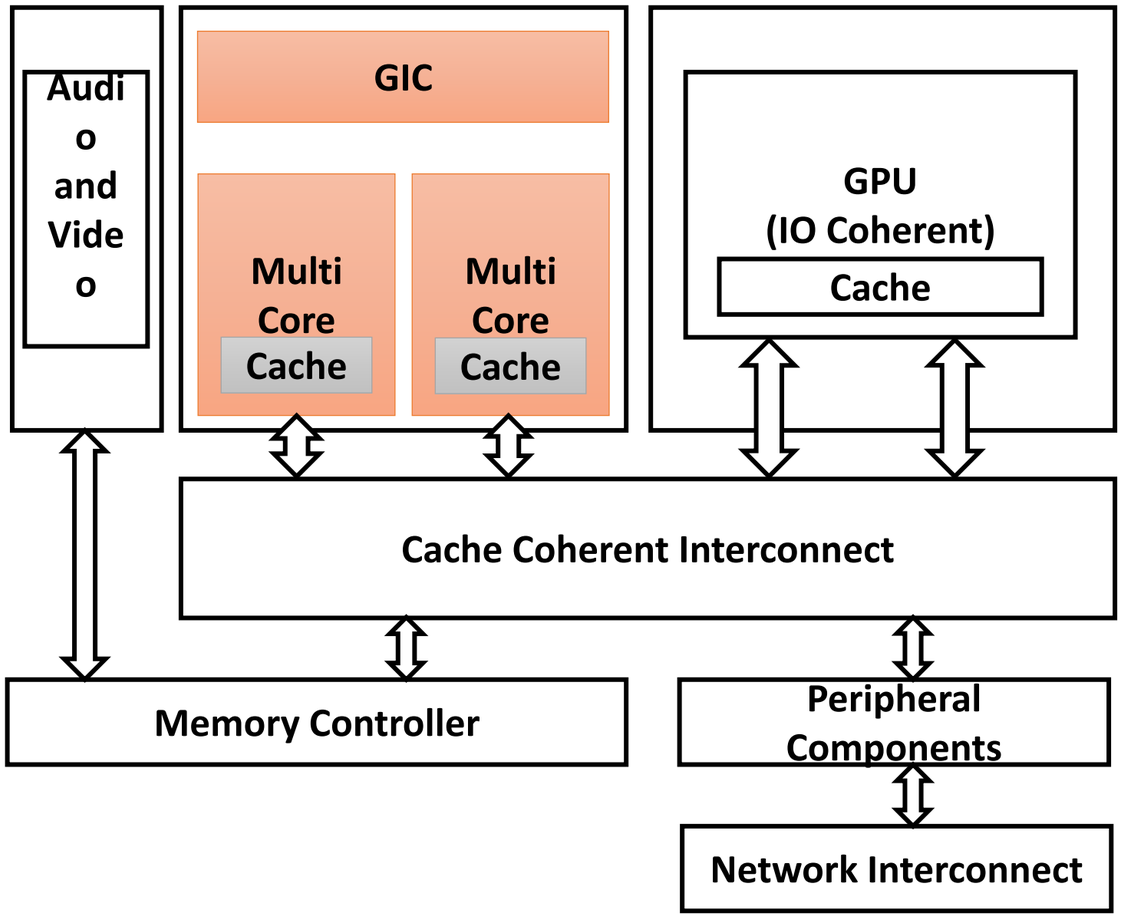
Introduction



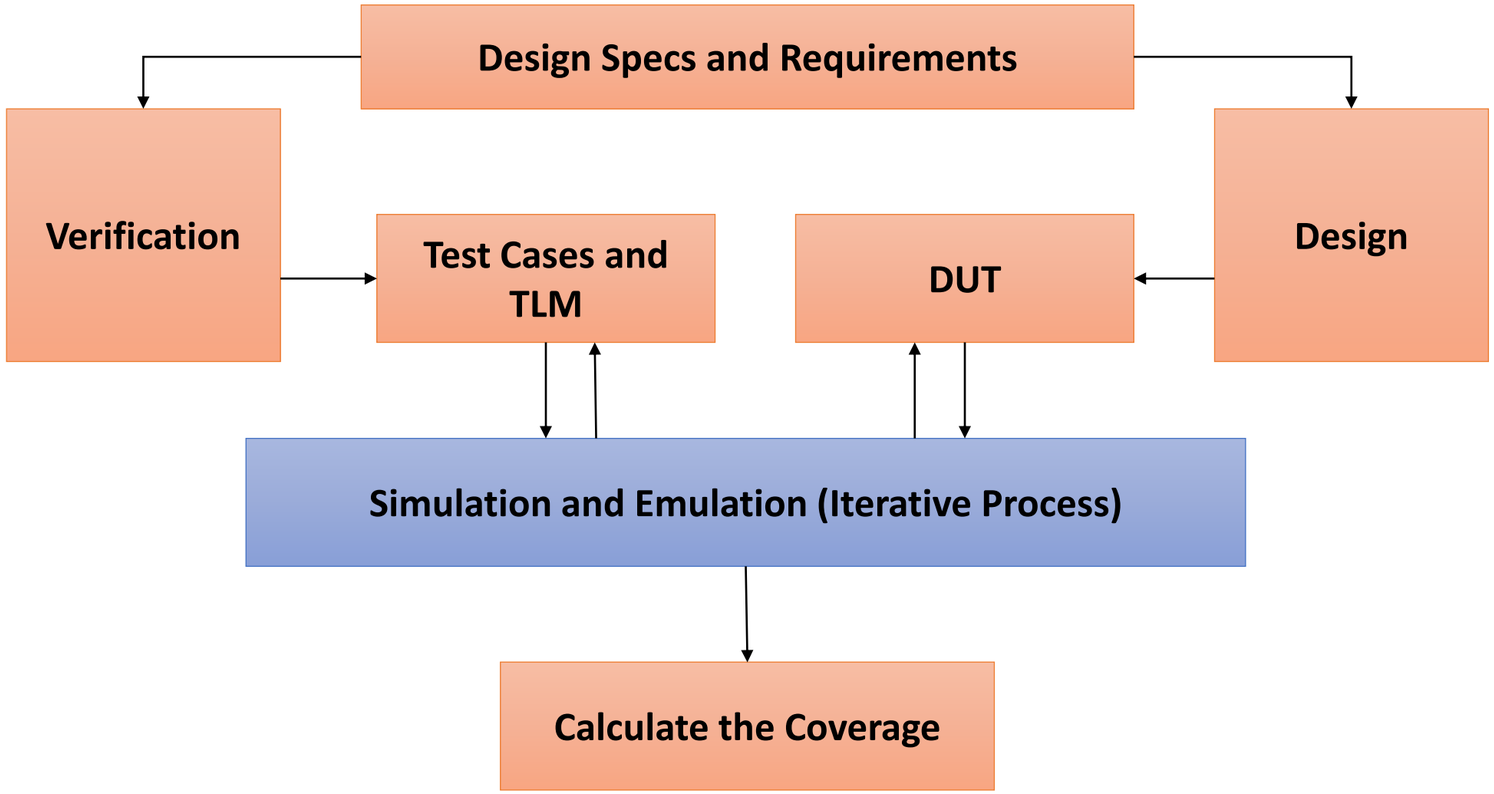
- Shrinking turn around times
- Need for faster simulations and quicker results
- Delay in time to market

Motivation

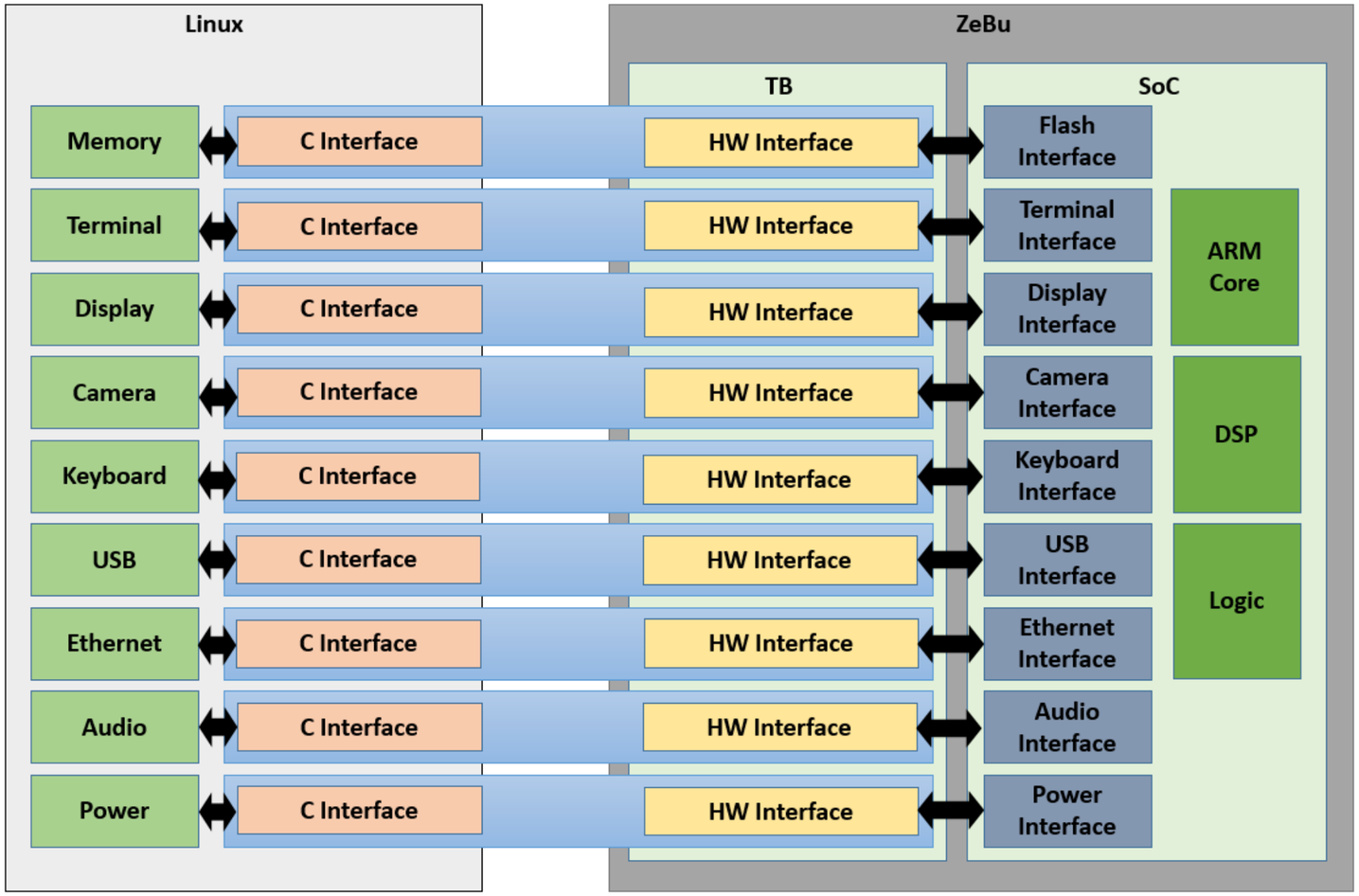
- Extremely fast, full System-on-Chip testing
- Emulation enables longer test cases to be run in less time
- Accelerated Verification flow
- Quicker debug iterations



Pre-Silicon Verification Flow



Emulation Platform



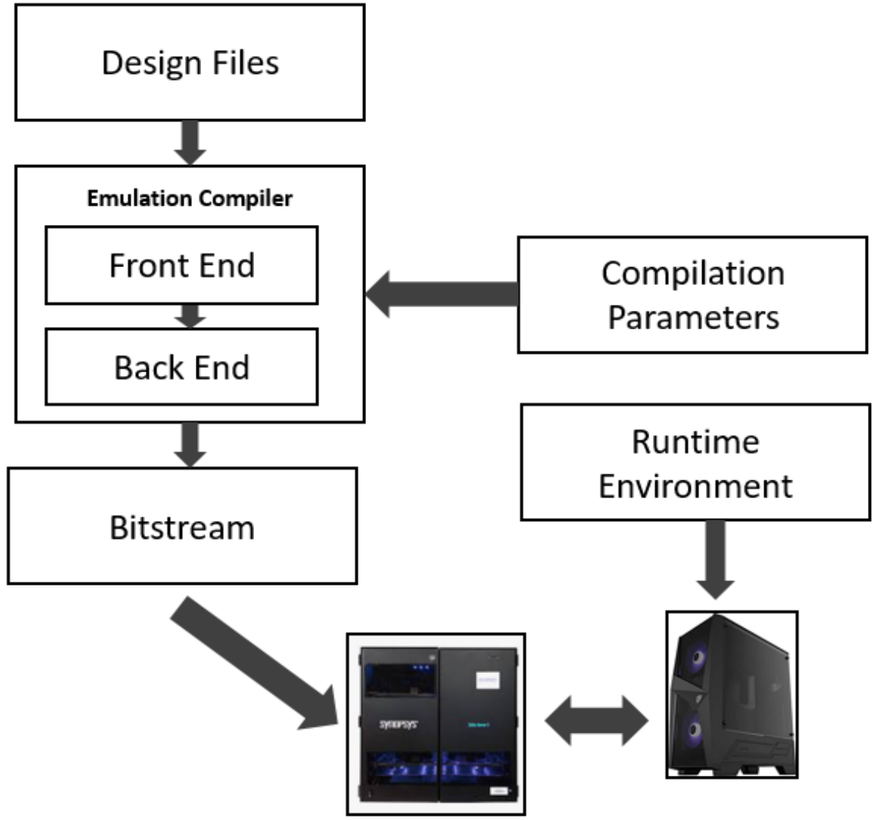
Emulation Compilation Flow

Front end compilation

System level back compilation

Core level back-end compilation

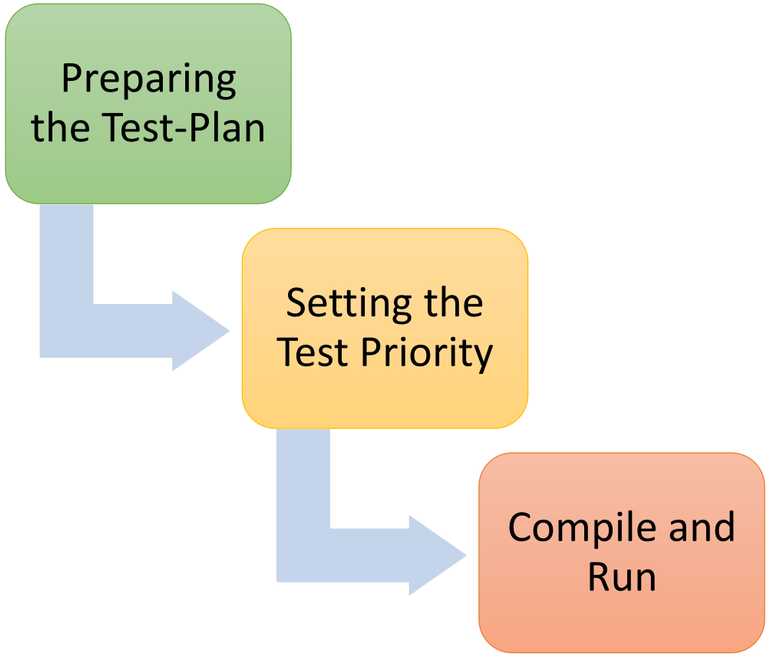
Place and Route

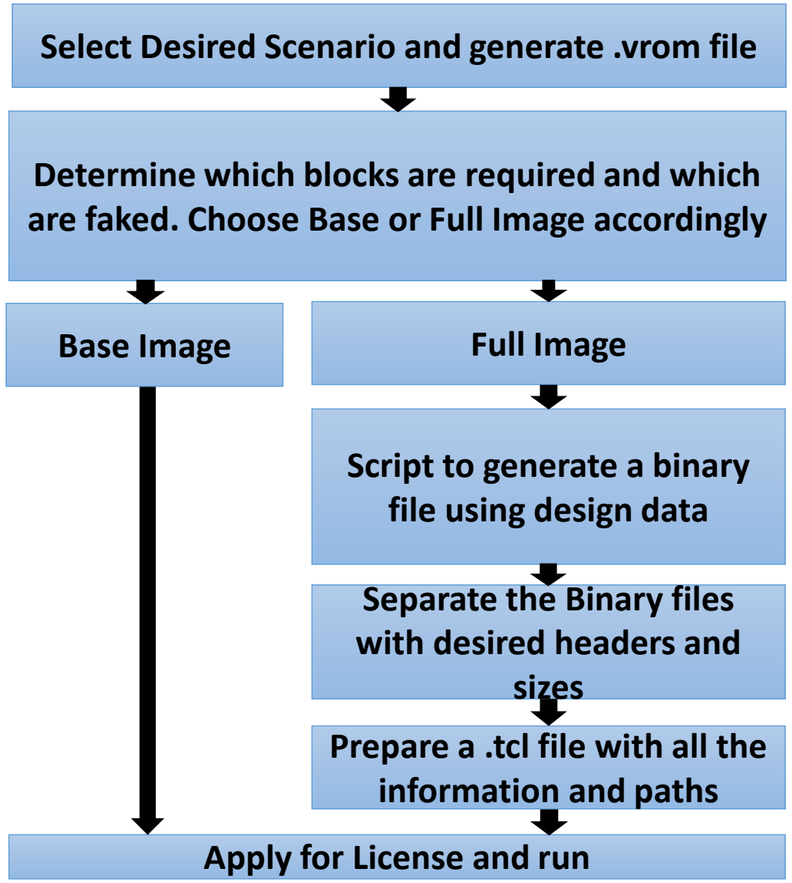


Motivation for Power Aware Verification at SoC

- Power-aware Verification of SoCs has been around for over a decade ever since IEEE released the UPF standard.
- Today, we not only have multi-core SoC designs but also multi-cluster SoCs with dual architecture processors implemented on the same die.
- Needless to say, this increase in complexity due to SoC architecture evolution has seeped into the requirements of power-aware verification and is fast becoming the long-pole of the verification lifecycle.
- We divided the power-aware verification requirements into a number of test categories and successfully accelerated the power-aware verification lifecycle with the use of emulators by incorporating UPF design libraries and power domain control into the design images used for emulation-based verification.
- This has resulted in up to 40x savings in time and resources.

Basic Methodology





Case-Study 1: Complex Coherency Scenario

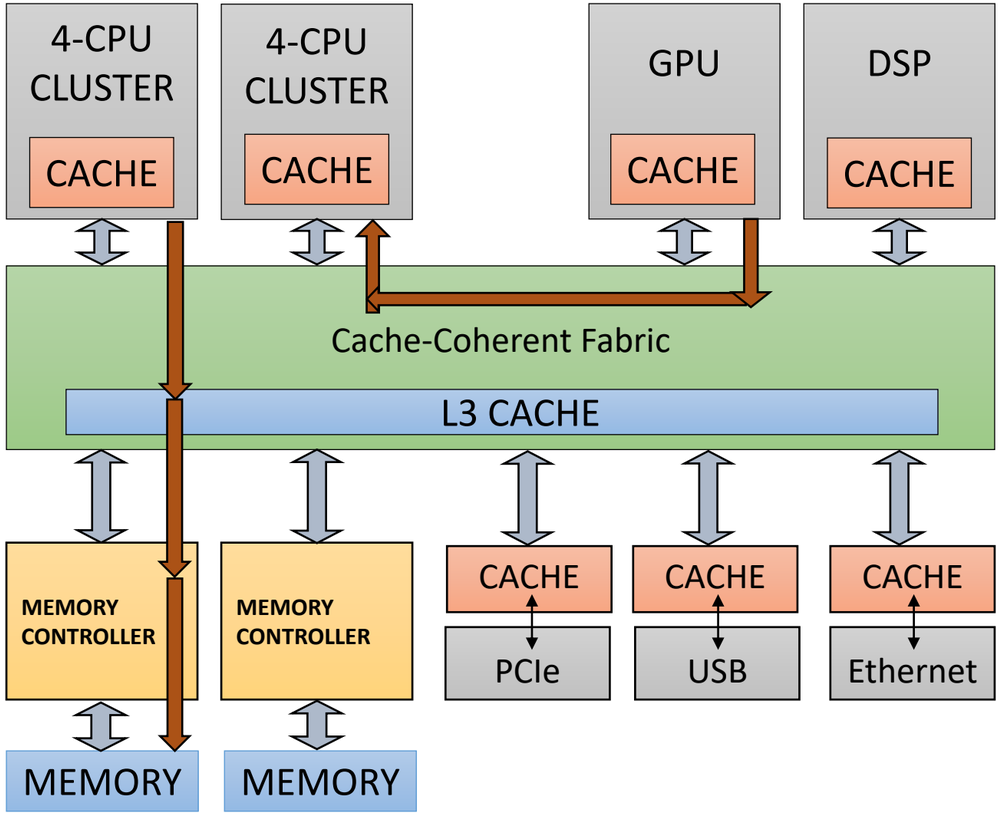
SoC teams are connecting multiple clusters together

Processor vendor may not provide coherent connection mechanism for multiple clusters

System level Cache coherency is fundamentally had to verify !

Stressed scenarios mixing these:

- Page Table Properties
- Memory Ordering
- Power Scaling
- Cache-line Evictions



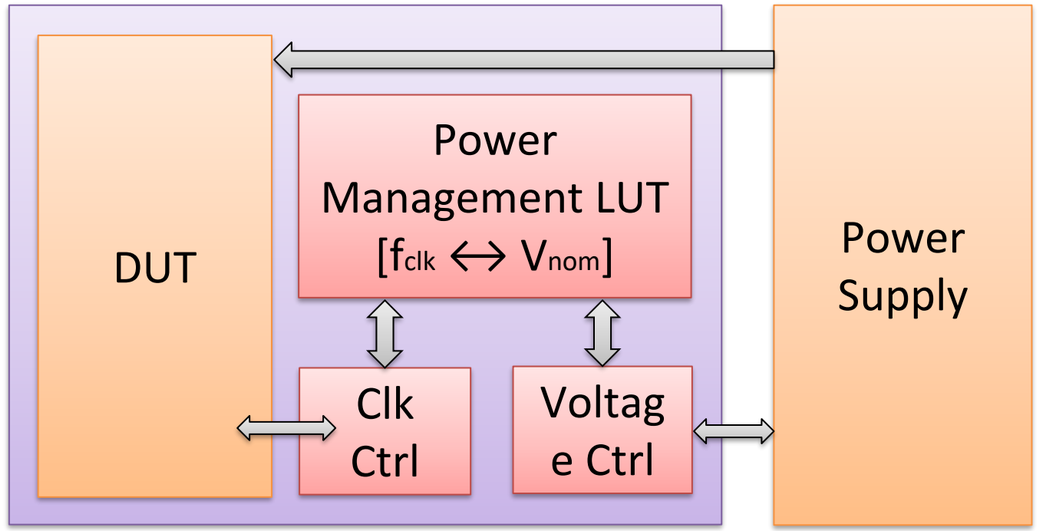
Case-Study 2: Dynamic Voltage and Frequency Scaling

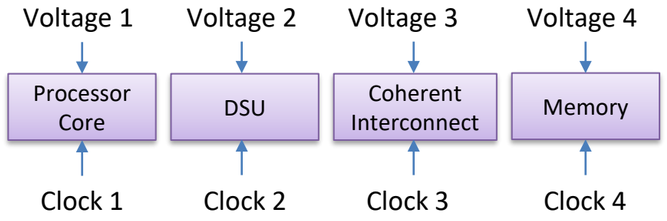
Ways to reduce power consumption of chips on the fly

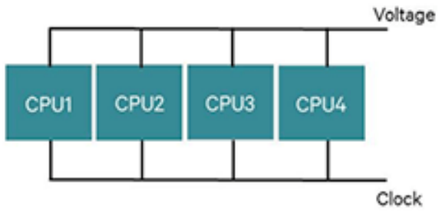
Different Design Blocks at SoC work on different frequency

Verification team has to traverse through all the frequencies for testing

Testing/Scaling of frequencies for memory interface, cores, coherent interconnects etc take huge simulation time !

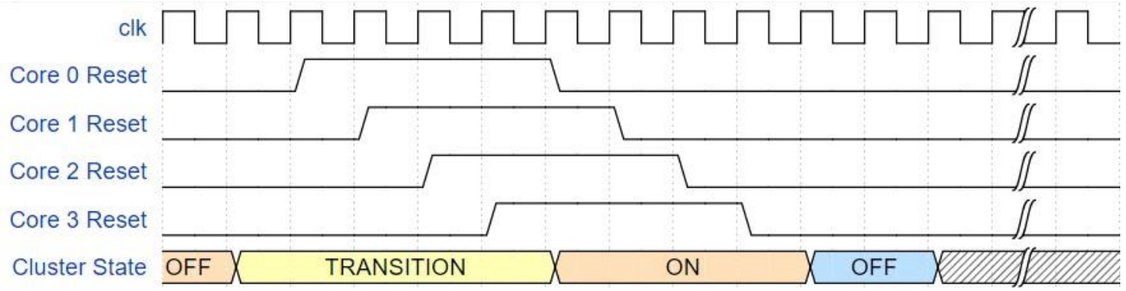


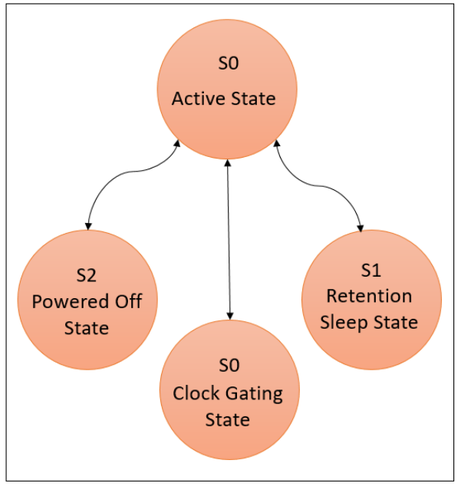




Case-Study 3: Low Power Scenarios

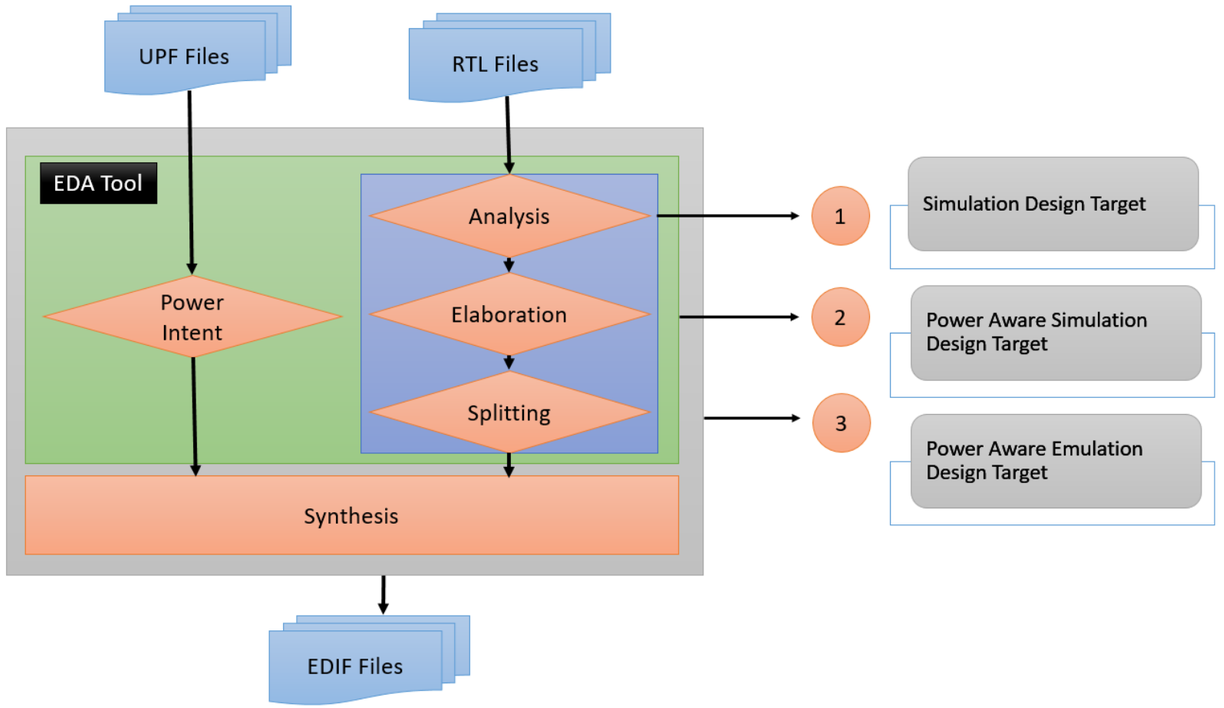
- ✓ Various low power scenarios at Emulation level
- ✓ Core/Cluster Clock Gating schemes, Memory Retention schemes as well as complete power down and sleep scenarios for various blocks of SoC.
- ✓ Figure shows different processor states possible and the waveform shows a typical Power Down scenario for Cores and Cluster which can be verified using emulator in an efficient manner.
- ✓ These low power scenarios are also run in Power-Aware setup with UPF incorporated.





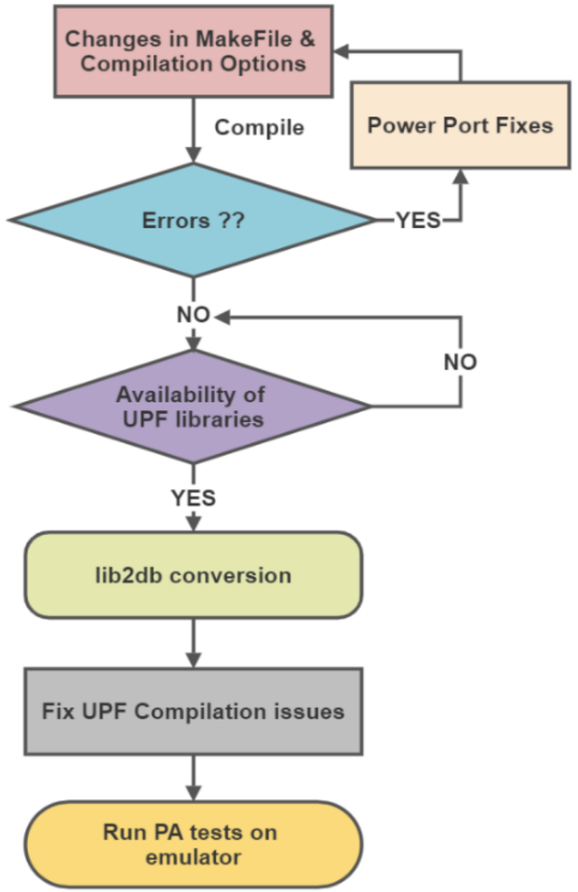
Methodology : Low Power Set-up

- Both UPF files and RTL files are fed to the EDA tool which understands the power intent of the design through UPF files and functional intent from design files.
- After the process of analysis, elaboration and splitting, and then synthesis, EDIF files are produced which is a vendor-neutral file format used to store netlists.
- The power-aware emulation model compilation flow has:
  - Similar UPF constructs are supported between simulation and emulation environments.
  - Debug continuity across PA-Simulation and PA-Emulation.



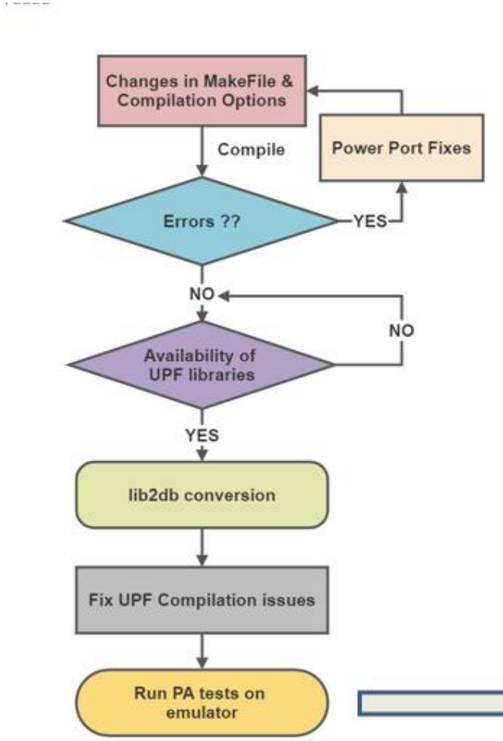
Compiling a PA Emulation Design Target

- ❑Makefile changes and identifying compilation options that need to change for running low-power scenarios.
- ❑While compiling, we need to fix any errors that may come due to any power related ports for any IP instances.
- ❑We also need to ensure the availability of complete list of UPF libraries to be included.
- ❑Next step is to use the available utility to convert present libraries to Emulation compatible libraries.
- ❑After that we can run lib to DB conversion script and convert libraries to the format needed.
- ❑After all this, once we fix all the UPF compilation issues related to version or any specific tool option, we can run our basic single/multi core power down scenarios to verify that the intent of the test has been achieved.



Power Aware Compilation Flow

- This methodology ensures that all synthesizable design components are re-used between PA Simulation and PA Emulation.
- This means now we can use both platforms to complement each other in the Validation lifecycle with a high degree of confidence.
- We have efficiently offloaded long running power aware scenarios to the PA Emulation environment to bring down the testing time to a fraction of what it was in the PA Simulation environment.




```
module moduleName (
    // regular ports
    input x1,
    input x2,
    ...
    ...
    output a1,
    output a2,
    ...
    ...
    //power ports
    `ifdef UPF_ENABLED
    inout power_port1,
    inout power_port2,
    inout power_port3,
    ...
    `endif
);

endmodule
```

```
extern "C" void power_ctrl_pd1 (const int power_on )
{
    if (power_on)
    {
        //logic to set voltage of pd1 to 0.75
        setVolt("top.dut.VDD_pd1", 0.75);
    }
    else{
        //logic to turn-off pd1
        setVolt("top.dut.VDD_pd1", 0);
    }
}
```

Some Common Issues

✓ ***Clash of Rules and Design Attributes amongst various IPs:***

Enabling/disabling of power well biases varies from one IP to another and sometimes clashes with the power bias enabled/disabled rule at the SoC level. To resolve this, we had set enable_bias to false for all IPs and the SoC top.

```
Error- [UPF_ATTR_VALUE_MISMATCH] Attribute value different from parent
/disk/path/to/ip/A/ipA_top.upf, 20
Attribute 'enable_bias' has been specified in an ancestor hierarchy as
'false'. Cannot specify attribute as 'true' in scope
```

✓ ***Corrupt cell libraries:***

Cell libraries for various IPs have vastly different release schedules and hence during integration, sometimes we can end up with stale and corrupt lib paths which need to be updated manually.

```
Error- [CFCILFBI] Cannot find cell in liblist
/disk/path/to/ip_A/design/files/RTL/ip_A_prj_<size>_ram_<attr>.v, 162
Cell 'ip_A_<process>_<size>_rtl_top' cannot be found in liblist for
binding instance
`top.dut.<ip_A_soc_hierarchy>.cell_name
Liblist: WORKLIB_ip_A DEFAULT
```

Some Enhancements Done

1 Single Step Run of the Scenario

- Automate the process of launching the run
- Waveform dump using the script
- Binary file split and use for full image runs

2 Changing signal value during run time

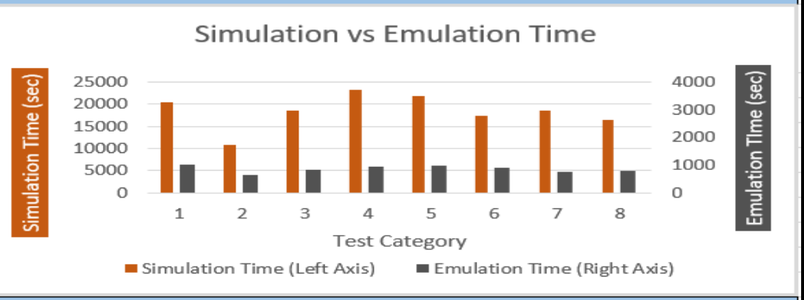
- Deposit
- Inject
- Force

3 Enable Checkers/Monitors

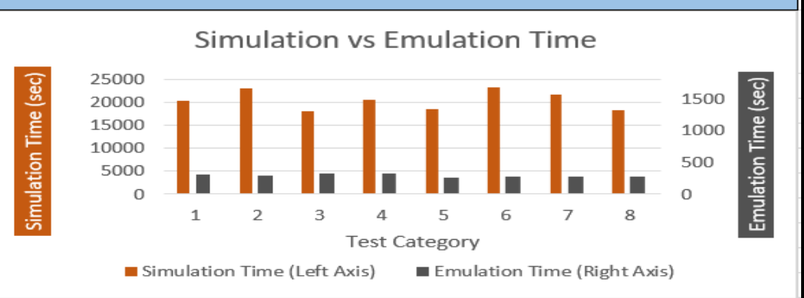
- Some SV-C handshakes should be enabled in Emulator run
- Specific Tracer/Monitor enablement

Results

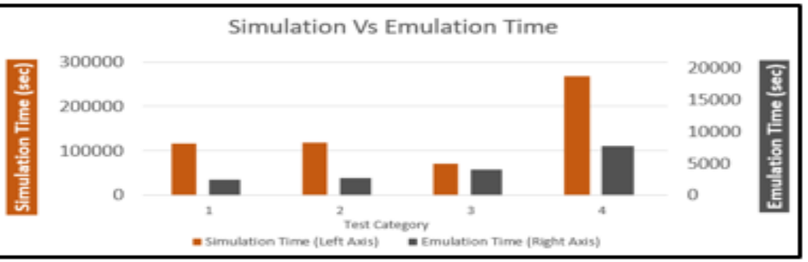
CADENCE PLATFORM				
Test Category	No. of clock cycles Millions	RTL Simulation Time (sec)	Emulation Run Time (sec)	Speedup %
Category 1	8.641	20529	1020	95.0314
Category 2	5.182	10715	645	93.9804
Category 3	6.526	18654	840	95.4969
Category 4	7.951	23279	960	95.8761
Category 5	7.375	21782	971	95.5422
Category 6	7.250	17458	911	94.7818
Category 7	6.404	18580	765	95.8827
Category 8	5.530	16381	785	95.2079



SYNOPSYS PLATFORM				
Test Category	No. of clock cycles Millions	RTL Simulation Time (sec)	Emulation Run Time (sec)	Speedup %
Category 1	9.199	20492	307	98.5019
Category 2	9.056	23235	288	98.7605
Category 3	7.374	18065	322	98.2175
Category 4	8.641	20529	331	98.3876
Category 5	6.526	18654	268	98.5633
Category 6	7.951	23279	275	98.8187
Category 7	7.375	21782	273	98.7467
Category 8	7.374	18432	275	98.5080



Test Category	RTL Simulation Time (sec)	Emulation Run Time (sec)	Speedup %
Category 1	116877	2417	97.9320
Category 2	119090	2733	97.7051
Category 3	70703	4027	94.3043
Category 4	267216	7782	97.0877



Conclusion

- Multiple long-running scenarios can be smoothly ported onto an emulation platform and are much easier and faster than simulation.
- The experiments were carried out for several functional and low power scenarios along with simulation for the same scenario on same design drop for comparison.
- The run times were recorded and analyzed. The corresponding plots are shown in Figure 16 as an indicator of the results.
- By using this method, we significantly reduced the number of iterations needed, debug time, and the verification schedule. Current results are indicative and based on two emulator platforms from vendors like Cadence and Synopsys.
- We also narrowed down the areas of critical bugs since we specifically aim to find relevant bugs early through this environment and not the complete coverage.
- Further enhancing the process by adding automation, verification engineer can maneuver between the two environments very easily as per the requirement.
- The results achieved in some of the Exynos Mobile SoCs and Automotive SoCs were demonstrative of the fact that we saved at least 50X time in closure of certain critical long pole features.

BENEFITS

- Huge saving in SOC simulation run time
- Scalable to various platforms across different designs
- Major use case in low power functionalities
- Has the potential to be applied to IO Coherent scenarios

LIMITATIONS

- Licenses are limited for parallel runs
- Debug prints, logs and checkers are difficult to integrate and use
- Randomization of signals and memories is difficult
- Complete waveform dump with all hierarchies and full time duration limits the performance

Future Scope

- 4 State runs on Emulator
- Unified compile flow
- Hybrid Emulation
- Enabling SV assertions in ZeBu

Care abouts

- Usage of software code
- Choosing the scenarios wisely
- ZTDB/FSDB dump duration
- Additional task of model enablement

Acknowledgement

The authors would like to thank Samsung Semiconductors India Research for enabling the work mentioned in this paper. We would also like to thank DVCon Europe team for giving us the opportunity to participate in the conference and present our work.