2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Improve Emulator Test Quality By Applying Synthesizable Functional Coverage

Hoyeon Hwang, Taesung Kim, Sanghyun Park, Yong-Kwan Cho,
Dohyung Kim, Wonil Cho, Sanggyu Park

SAMSUNG

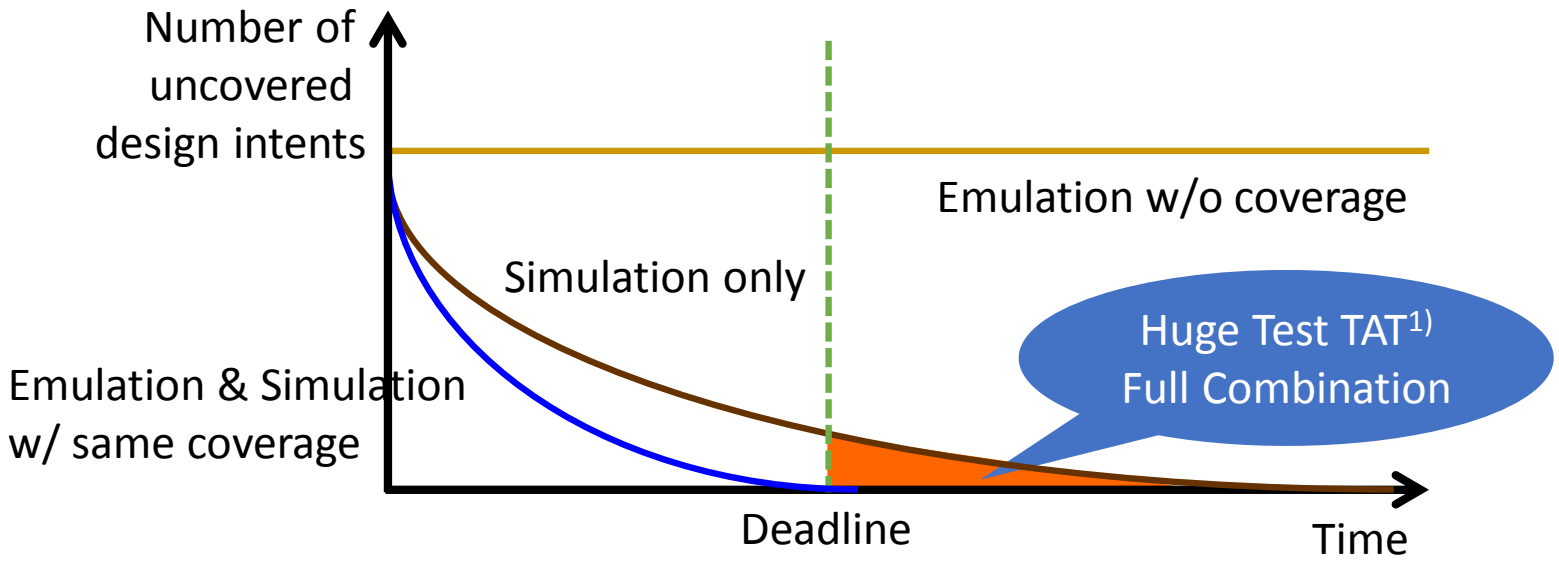
accellera
SYSTEMS INITIATIVE

Agenda

- Why functional coverage is essential in emulator?
- Syntax, Capacity & Speed considerations
- Functional coverage coding guidelines
- Integration functional coverage
- Merge coverage metrics data
- Two case studies
- Conclusion

Why functional coverage is essential in emulator?

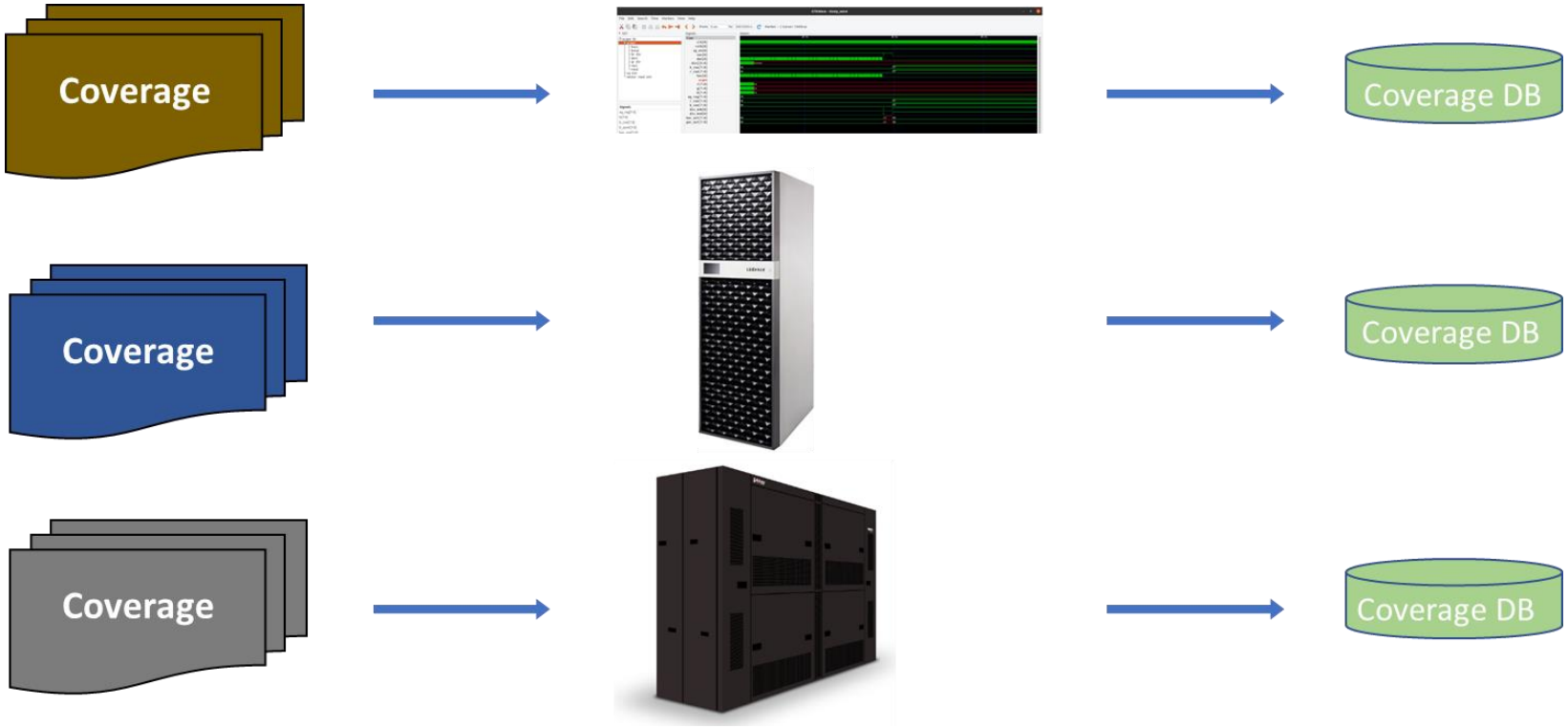
- The way how to quantitatively evaluate verification quality in emulator



1) TAT: Turn-around-time

Syntax, Capacity & Speed considerations

Syntax (difference) Capacity (limitation) Speed (reduction)



Reflect considerations in coding guidelines

Syntax (difference) Capacity (limitation) Speed (reduction)

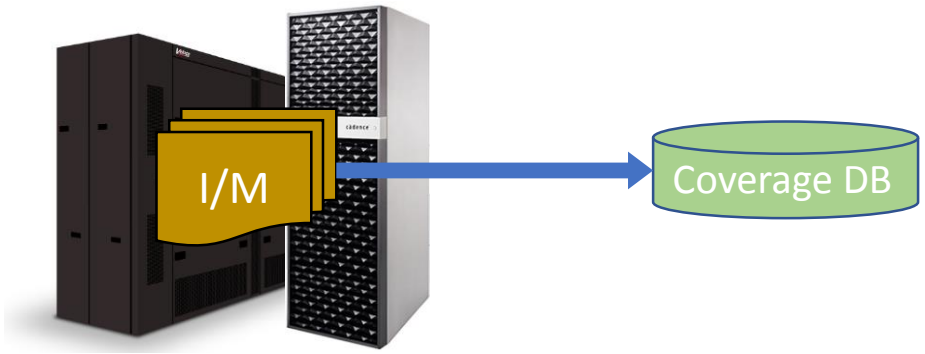
Find common syntax

Use less resource
→ more coverage

Speed considerations

- With 191,797 bins, the coverage data dump only takes additional 1 to 2 minutes.

	C-vendor emulator		M-vendor emulator	
	w/o Func. Cov.	w/ Func. Cov.	w/o Func. Cov. (Throughput: 90.47%)	w/ Func. Cov. (Throughput: 90.59%)
Run-time Average	12min 25sec	13min 13sec	26min 31sec	24min 27sec
Coverage DB Dump Time	N/A	1min 2sec	N/A	1min 20sec



Functional coverage coding guidelines

- Use the emulator’s capacity more efficiently
- Consider synthesizable in emulator
- Use common expression, which can be applied all kinds of emulators

Guideline: Use minimum resource (1/4)

- "*ignore_bins*" usage example in cross coverage
  - Prefer to apply "*ignore_bins*" in coverpoint, not in cross coverage

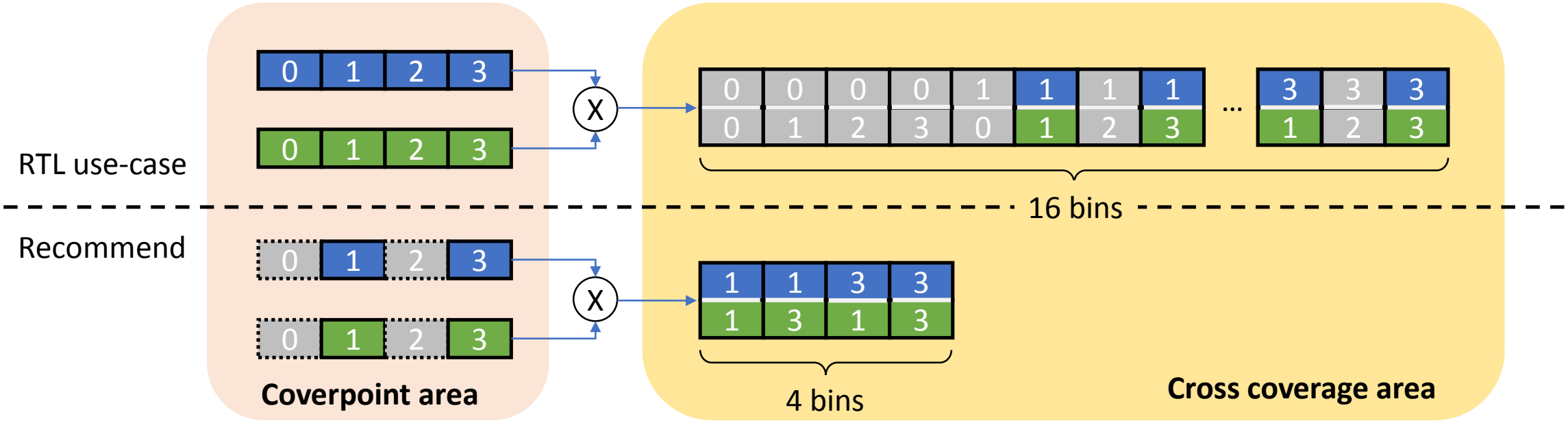
RTL simulation general usage	C-vendor emulator (recommend)

covergroup CG @(posedge clk);
  A: coverpoint alpha; // 4 bins
  B: coverpoint bravo; // 4 bins
  odd_combinations cross A, B { // 4 bins
  ignore_bins odd_A = binsof(A) intersect {0, 2};
  ignore_bins odd_B = binsof(B) intersect {0, 2};
  }
endgroup
	covergroup CG @(posedge clk);
	  A: coverpoint alpha {
	  ignore_bins odd = {0, 2};
	  }
	  B: coverpoint bravo {
	  ignore_bins odd = {0, 2};
	  }
	  odd_combinations: cross A, B;
	endgroup

Cross coverage bin count: 4	Cross coverage bin count: 4
Emulator internal memory: 16	Emulator internal memory: 4

Guideline: Use minimum resource (2/4)

- Emulator resource management for cross coverage
  - Drive the coverpoint hit counter from the cross data



Guideline: Use minimum resource (3/4)

- One same coverpoint usage in many cross coverages
  - Separate cross coverages into different covergroups

RTL simulation general usage

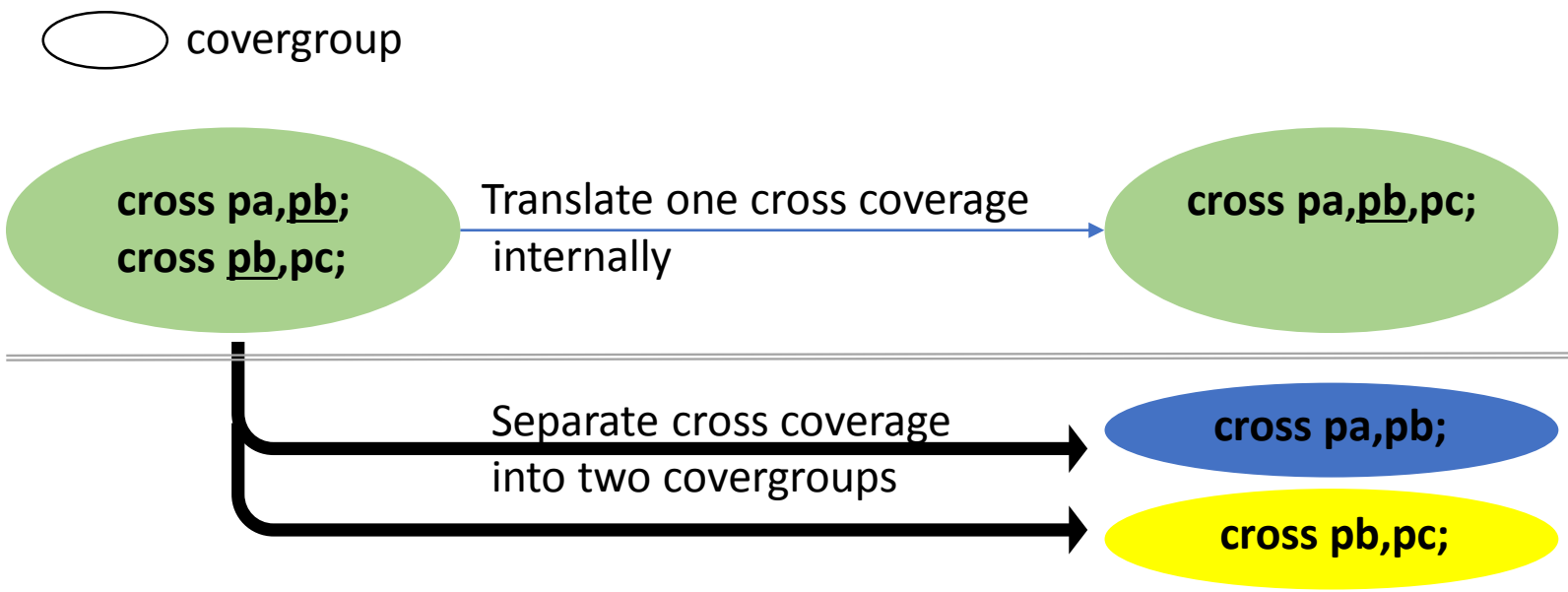
```
reg [2:0] a,b,c;

covergroup CG_A;
  pa: coverpoint a;
  pb: coverpoint b;
  pc: coverpoint c;
  cross pa,pb;
  cross pb,pc;
endgroup
```

C-vendor emulator (recommend)

```
reg [2:0] a,b,c;

covergroup CG_B_0;
  pa: coverpoint a;
  pb: coverpoint b;
  cross pa,pb;
endgroup

covergroup CG_B_1;
  pb: coverpoint b;
  pc: coverpoint c;
  cross pb,pc;
endgroup
```

Guideline: Use minimum resource (4/4)

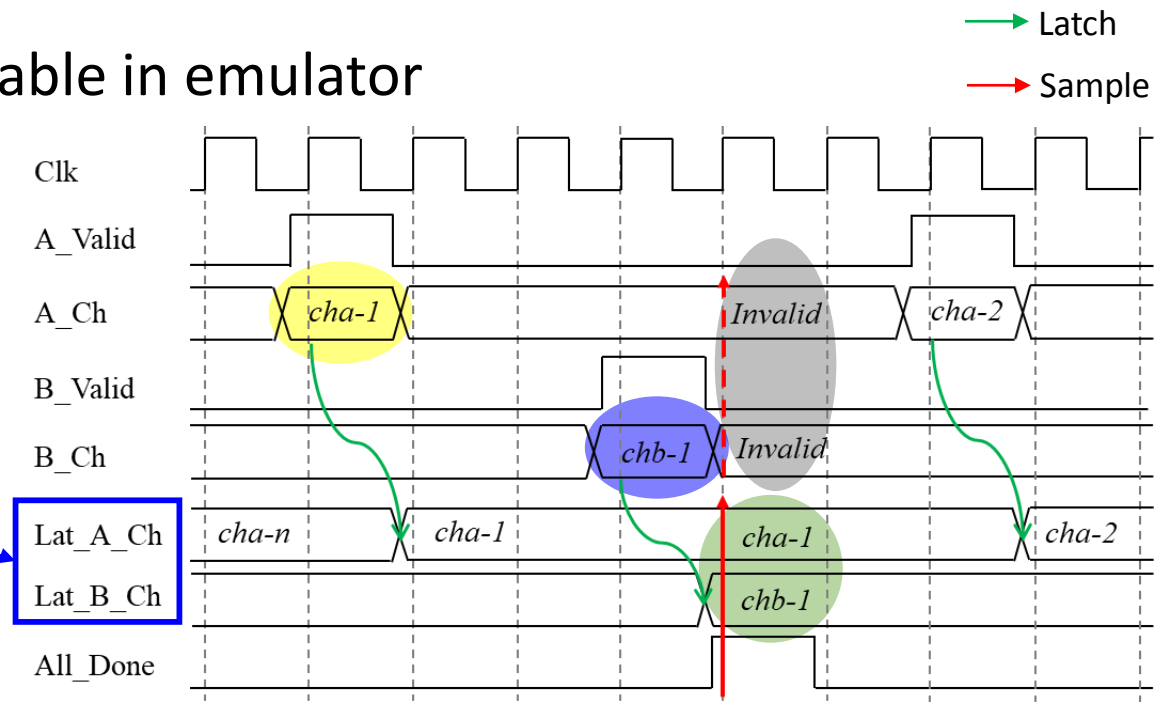
- Emulator internal memory management
  - Smaller internal memory counter(IMC) per one covergroup is required smaller space



	IMC / Covergroup	Total Bin Count
cross pa,pb,pc;	2^3	2^3
cross pa,pb; / cross pb,pc;	**2^2**	2^3

Guideline: Consider synthesizable

- Sampling
  - The "*sample*" method is not synthesizable in emulator
  - Define special sample event
    like *@(posedge All_done)*
  - Need extra signals for cross coverage



Example waveform to sample two data(A_ch, B_ch) path information

Guideline: Use common expression

- Number of bins
  - Do not exceed 4,096 bins(12bit, [11:0]) per one coverpoint

RTL simulation general usage

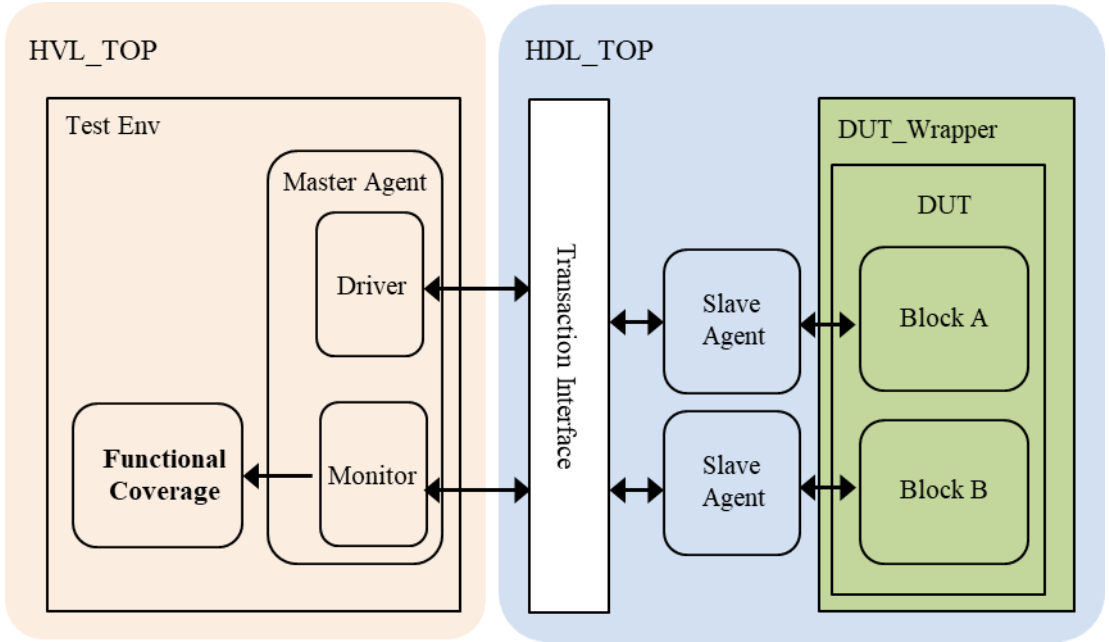
```
covergroup CG_IMG_SIZE@(posedge clk);
  P: coverpoint pre_img_size[12:0]; // Max size: 8192
  S: coverpoint scaled_img_size[12:0];
endgroup
```

M-vendor emulator (recommend)

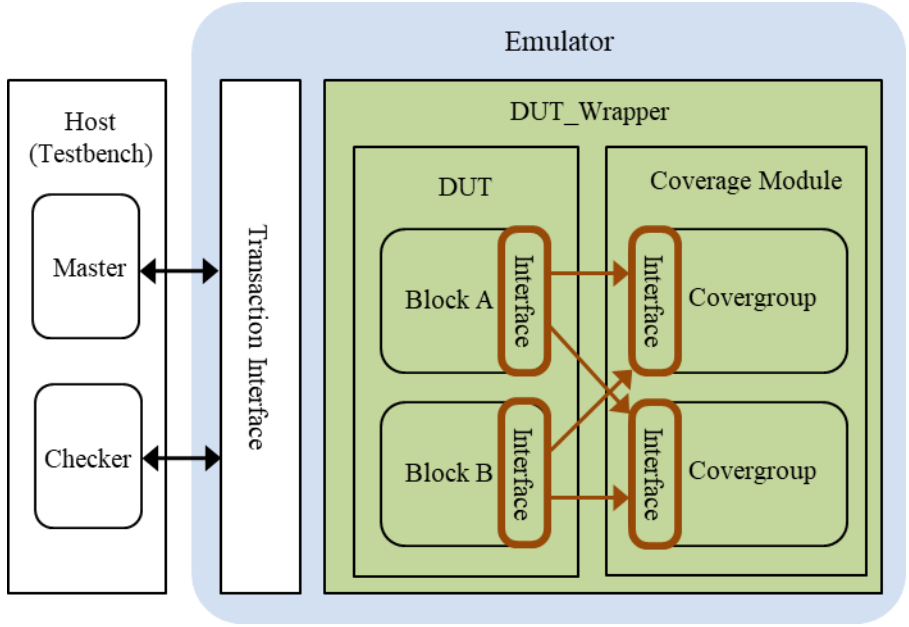
```
covergroup CG_PRE_IMG_SIZE@(posedge clk);
  P_0: coverpoint pre_img_size[11:0];
  P_1: coverpoint pre_img_size[12];
  P: cross P_1, P_0;
endgroup

covergroup CG_SCL_IMG_SIZE@(posedge clk);
  S_0: coverpoint scaled_img_size[11:0];
  S_1: coverpoint scaled_img_size[12];
  S: cross S_1, S_0;
endgroup
```

Integration functional coverage

- Coverage module is placed into DUT wrapper



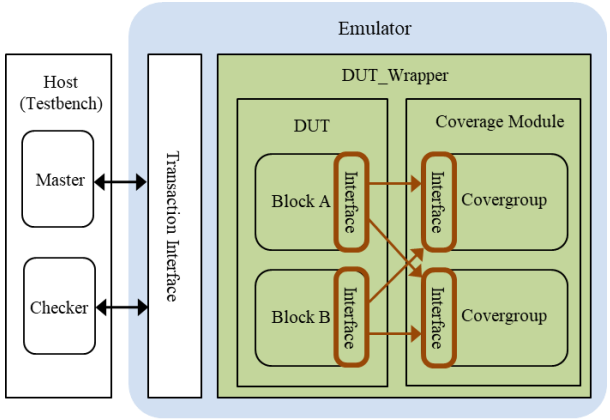
[Previous]



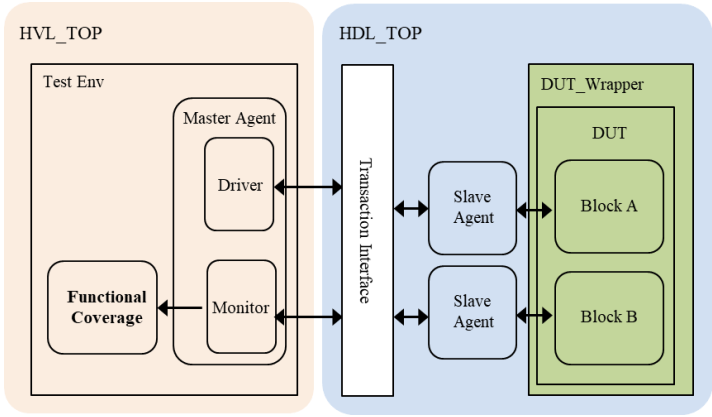
[Proposed]

Integration functional coverage: Benefits

- Why coverage module is placed into DUT wrapper ?
  - Re-use the functional coverage both simulation and emulation
    - Easy to merge coverage database
  - Lower performance impact when pull signals from DUT



[Proposed]



[Previous]

Merge coverage metrics data

- Conditions to merge coverage with simulation's coverage data
  - A block hierarchy including functional coverage should be same
  - Can be merged within same vendor

- For next step, share of coverage data across hardware accelerators using UCIS (Unified Coverage Interoperability Standard)

Two case studies

- TAT reduction of corner case hunting
  - Define functional coverage with 2,632 bins

Bin Count	Simulation	Emulation
2,632	157,920 hours [1)]	421 hours [2)]

[1)] 2,632 bins * 60 hours. Assume that 1test per 1bin is required.

[2)] 2,632 bins * 10 mins. Assume that 1test per 1bin is required.

- Measure full combination coverage
  - Define functional coverage with 4,846 bins
  - Able to verify all combinations without missing cases

Conclusion

- Functional coverage is essential in emulator
- Proposed the common functional coverage coding guidelines
- Showed effectiveness for TAT reduction and suitability for full combination case applying functional coverage in emulator

Questions