Functional Coverage Planning and Automation for Highly Configurable IP

Jeremy Ridgeway
Avago Technologies, Ltd.
Fort Collins, CO
jeremy.ridgeway@avagotech.com

Kavitha Chaturvedula
Avago Technologies, Ltd.
San Jose, CA
kavitha.chaturvedula@avagotech.com

Karishma Dhruv
San Jose, CA
karishma.dhruv@gmail.com

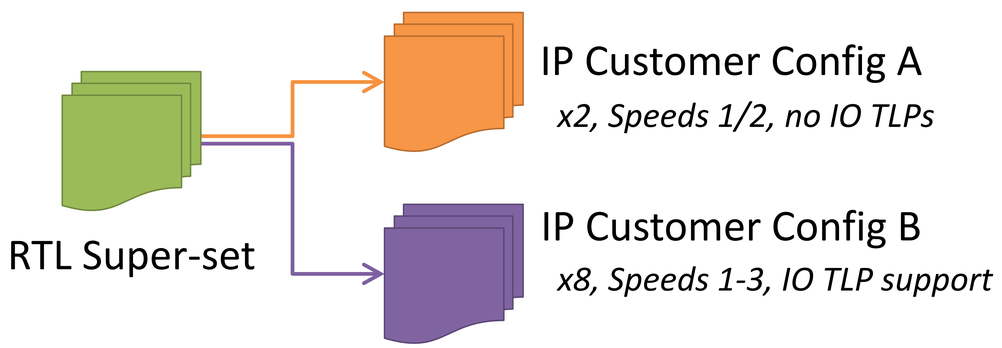
Is IP Configuration A covered in *all* operating modes?

Functional coverage for RTL source may not be applicable in all configurations

Employ (lots of) waivers?

Configuration may have multiple top-level modes of operation

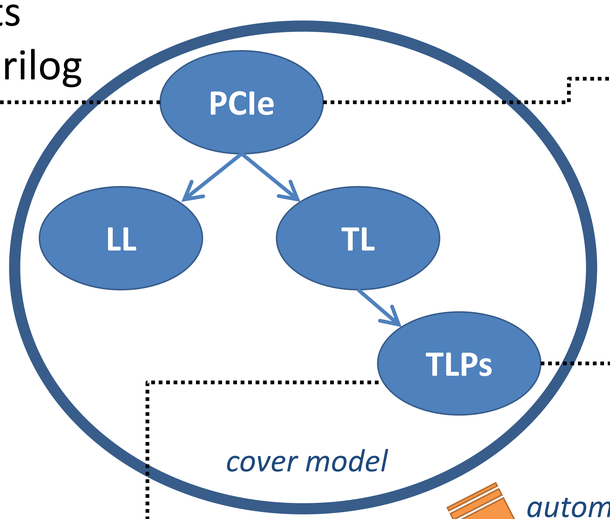
Need to cross with mode in every *covergroup.*



Use a Flexible Hierarchical Functional Coverage Model!

Directed Acyclic Graph: children inherit from parents

Automation processes model to generate SystemVerilog



Configuration Variables

Name	Symbol Name	Range	Description
Link Width	C_WIDTH	1, 2, 4, 8	IP supported widths
Link Speed	C_SPEED	1, 2, 3	IP supported speeds
IO Packet Support	C_IO_TLP	0, 1	IP supports IO TLPs

Define applicable coverage

Filters-out invalid cross scenarios in covergroups

Cover Variables

Name	Symbol Name	Range	Description
TLP Type Field	TLP_TYPE	MsgD, IO_RD, ...	Type of TLP transmitted
Payload length	HDR_DLEN	[0:4096]	Payload length specified in TLP header

Defines point-of-observation coverage

Used in (any number of) covergroups as contextual cross scenarios

Mode Variables

Name	Symbol Name	Range	Signal
Link Width	M_WIDTH	1, 2, 4, 8	int CFG.M_WIDTH
Link Speed	M_SPEED	1, 2, 3	int CFG.M_SPEED

Explicitly define coverage reporting space

Crossed with child covergroup scenarios

Cover Groups

Name	Expected Packet Types		
Symbol Name	pkt_types_cg		
Description	Specific packet type coverage scenarios		
Points	TLP_TYPE	HDR_DLEN	C_IO_TLP
c_0	MsgD	0	*
c_1	MsgD	{[1:1024]}, {[1025:4096]}	*
c_2	IO_RD, IO_WR		1

```
covergroup pkt_types_cg;
 coverpoint TLP_TYPE {
   bins T0 = { MsgD };
 }
 coverpoint HDR_DLEN {
   bins H0 = { 0 };
   bins H1 = { [1:1024] };
   bins H2 = { [1025:4096] };
 }
 // contd...
```

```
 coverpoint M_WIDTH {
   bins W0 = { 1 };
   bins W1 = { 2 };
 }
 coverpoint M_SPEED {
   bins S0 = { 1 };
   bins S1 = { 2 };
   bins S2 = { 3 };
 }
 // contd...
```

```
 c: cross TLP_TYPE, HDR_DLEN, M_WDITH, M_SPEED {
   bins c_0_0 = binsof(T0 && H0 && W0 && S0);
   ...
   bins c_1_0 = binsof(T0 && H1 && W0 && S0);
   ...
   bins c_1_1 = binsof(T0 && H2 && W0 && S0);
   ...
   bins c_1_x = binsof(T0 && H2 && W1 && S2);
 }
endgroup
```