Molding Functional Coverage 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

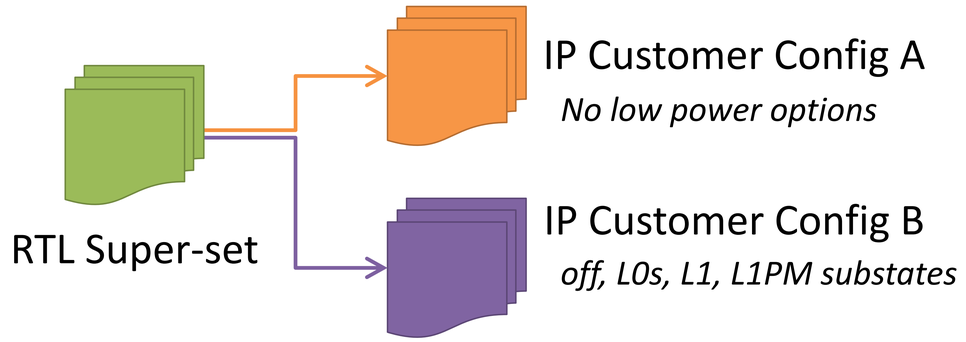
Karishma Dhruv
San Jose, CA
karishma.dhruv@gmail.com

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

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

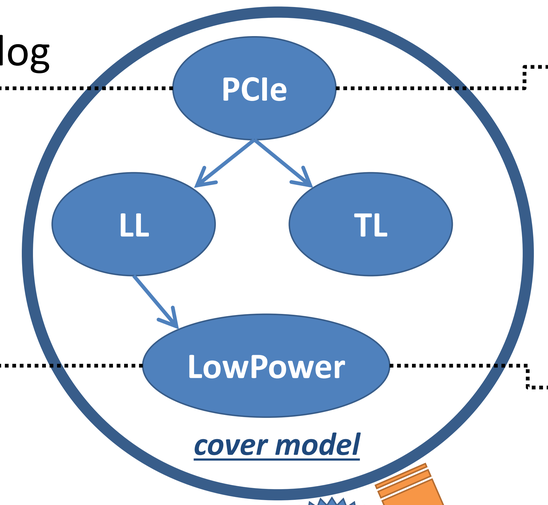
Name	Symbol Name	Range	Description
Low Power	C_lowpower	off, L0s_en, L1_en, L1PMss_en	IP supported low power modes

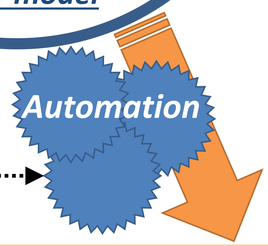
Define applicable coverage
Filters-out invalid cross scenarios in covergroups

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





Customer IP Setup

Mode Variables

Name	Symbol Name	Range	Signal
Low Power	M_lowpower	off, L0s_en, L1_en, L1PMss_en	int CFG::LP

Explicitly define coverage reporting space
Crossed with child covergroup scenarios

Cover Groups

Name	RX Datapath Refined					
Symbol Name	rx_datapath_cg					
Description	Required RX path only important scenarios to sim/verify					
Point Types						*Comment*
Points	Data	Control	ltssm_state	M_lowpower	C_lowpower	
pkt_delim_cross	$STP, $SDP, $END	1	L0			*Cover packet delimiters*
L0s_wake_rx_cross	$COM	1	L0s_rx_FTS	L0s_en	L0s_en	*Cover FTS receive*

```
covergroup rx_datapath_cg;
 coverpoint Data {
   bins Data_0[] = { 8'hFB, 8'hFC, 8'hFD };
 }
 coverpoint Control {
   bins Control_0 = { 1 };
 } // contd...
```

```
// ...contd
 ltssm_state: coverpoint tb.ltssm_o
   bins ltssm_state_0 = { L0 };
 }
 c: cross Data, Control, ltssm_state {
   bins pkt_delim_cross_0 =
        binsof(Data.Data0) &&
        binsof(Control.Control_0) &&
        binsof(ltssm_state.ltssm_state_0);
}
endgroup
```