

16 November 2022

Introduction to Hardware Verification.

Lars Viklund, ASIC Verification Manager, Axis Communications

www.axis.com



ASIC Development at Axis Communications



Axis Communications

- > Leader in network video surveillance
- > Swedish company
- > Part of Canon group
- > About 4000 employees worldwide







Axis ASIC Development

- > Long history of chip development
 - TGA/CGA Printer connectivity
 - ETRAX SoC for network connected devices
 - ARTPEC Network video
- > Why in house chip development?
 - Differentiation
 - Optimization for application
 - Access to technology
 - Unit cost







Axis ARTPEC Overview

- > Each generation offers more:
 - features
 - processing power
 - memory bandwidth
- > ARTPEC-1 (1999) First chip for network video
- > ARTPEC-2 (2003) MPEG encoding
- > ARTPEC-3 (2007) One chip network camera
- > ARTPEC-4 (2011) Wide Dynamic Range solution
- > ARTPEC-5 (2013) Forensic Capture
- > ARTPEC-6 (2017) Security
- > ARTPEC-7 (2019) LightFinder 2.0
- > ARTPEC-8 (2021) Deep learning on the edge
- > ARTPEC-9 (2023) ...



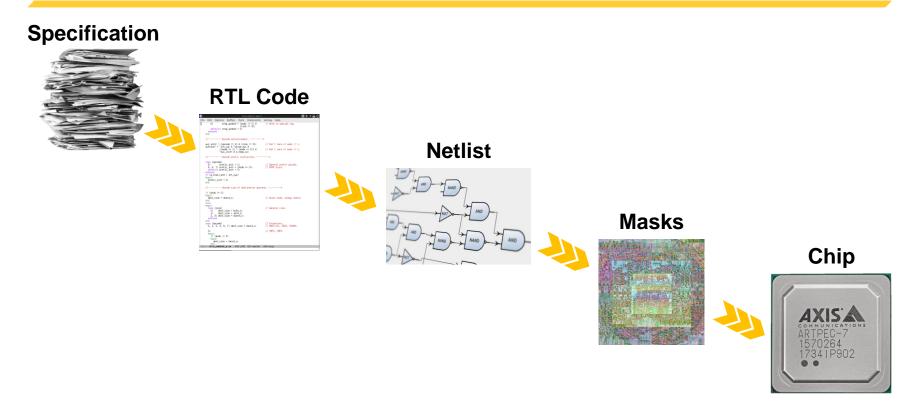




Overview of Hardware Verification



Digital ASIC Development Flow







Hardware Verification

- > Ensuring that the hardware design fulfills the requirements
- > Predominantly performed before tape-out
 - RTL
 - Netlists
 - UPF (power intent)
- > Verification vs. test in the context of ASIC





Hardware Verification vs. Software Testing

- > Same end objective
- > RTL is similar to software
- > However, hardware differs in that:
 - Very expensive to fix bugs after tape-out
 - Low abstraction level
 - Inherently parallel
 - Large state space





Requirements Specifications

- > Ideal world:
 - Complete
 - Unambiguous
 - Consistent
 - Appropriate level of detail
- > Reality:
 - Napkin sketches, gossip at the coffee machine, ...
 - Natural Language
 - Formal or informal
 - Models
 - Executable or not
 - Bit exact or not





Types of Requirements

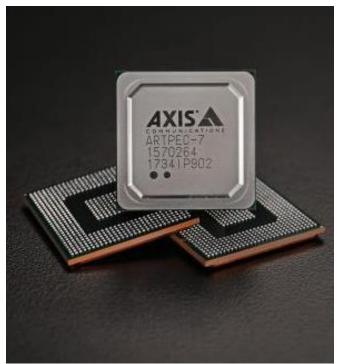
- > Functional Requirements
 - Specifies the behavior
- > Non-functional requirements
 - Performance
 - Reliability
 - Security
 - Safety
 - ...





Verification Complexity is Increasing

- > Larger and more complicated designs
- > More elaborate non-functional requirements
 - Clock gating
 - Power gating
 - Security
 - Safety
 - ...
- > Stricter project deadlines
- > Increased costs for tape-outs







Some Data

Source:

Wilson Research Group and Siemens EDA, 2022 Functional Verification Study

- > 50%-60% median ASIC project time spent in verification
- > Mean peak number of ASIC engineers:
 - 50% increase in design engineers since 2007
 - 145% increase in verification engineers since 2007
- > 76% ASICs require 2 or more respins
- > Mask costs (vs 28nm):
 - 12nm 2.5X
 - 5nm 15X
 - 3nm 25X



Dealing with Verification Complexity

- > Work hierarchically
- > Minimize overlap in effort
 - Horizontally
 - Vertically
- > Use the best technique for each problem
 - Simulation
 - Formal
 - Specialized tools
- > Re-use whenever possible







When is Verification Done?

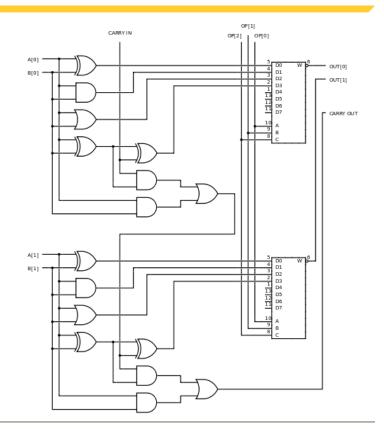






Verification Techniques

- > Dynamic Techniques
 - Simulation
 - Emulation and FPGA Prototyping
- > Static Techniques
 - Structural checks
 - Formal verification





SystemVerilog

- > Design Features
 - Synthesizable
 - Extends on Verilog
- > Verification Features
 - Classes
 - Constraints
 - Coverage
 - Assertions
- > Developed by the Accellera System Initiative
- > Std IEEE 1800-2017



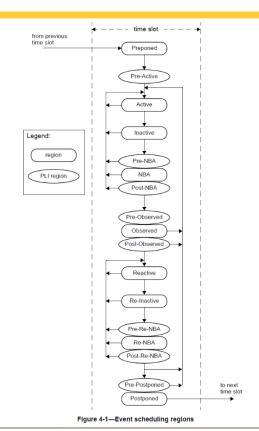


SystemVerilog Simulation

- > Execute in a software environment
- > Discrete event simulation
- > Four valued logic

- 0, 1, X, Z

- > Performed on RTL or Netlist
 - RTL Cycle behavior
 - Netlist With or without timing
- > Simulator can integrate other languages
 - Natively (VHDL, SystemC)
 - PLI or DPI





Test Benches

- > Instantiates the design under test (DUT)
- > Facilitates simulating the DUT
- > Models the environment of the DUT
 - Drives inputs
 - Monitors outputs
- > Does not have to be synthesizable

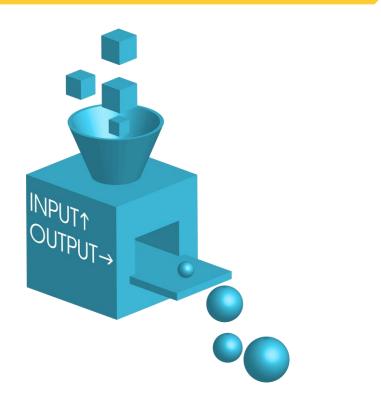






Modeling the Environment

- > Model what is outside the DUT
 - Other blocks on the chip
 - External devices
- > Model all legal behavior according to the specification of the interfaces
- > Model in layers
 - Handle complexity
 - Facilitate reuse

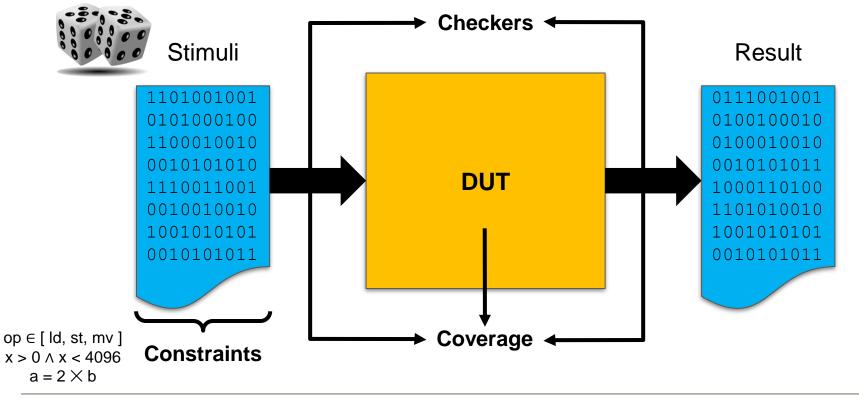






Coverage Driven Constrained Random Verification

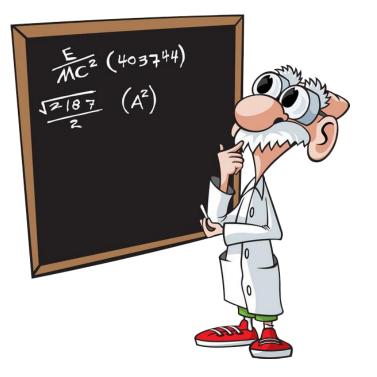
Coverage Driven Constrained Random Verification





SystemVerilog Constraints

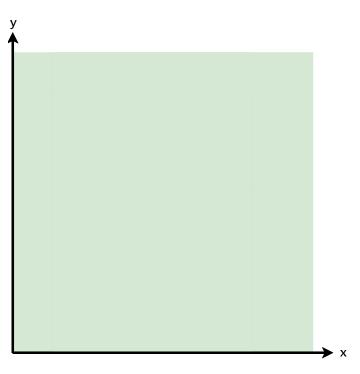
- > Defines legal values
- > Declarative
- > Compact
 - loops, conditionals, set operators, etc.
- > Constraint blocks are class members
- > Solver find random solutions satisfying the constraints





Constraints Example

```
rand logic [3:0] x;
rand logic [3:0] y;
constraint c_x { x inside {[2:12]}; }
constraint c_y { y < 14; }
constraint c_sum { x + y < 21; }
constraint c_x_mod { x % 8 != 0; }
```

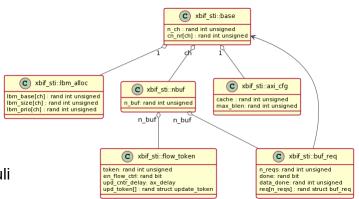






Modeling Stimuli

- > Model the input to the DUT as random variables
 - Configuration and data
 - Structure using object-orientation
 - Consider reuse of parts
- > Express legal stimuli using constraints
- > Abstract low-level stimuli when appropriate
 - To simplify constraints
 - To enable us to randomize interesting cases
 - Randomize high-level "knobs" and map to concrete stimuli
- > Appropriate granularity of randomization
 - Randomize at once or sequence the randomizations
 - Tradeoff between expressiveness and performance





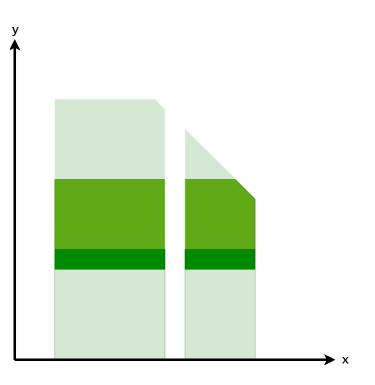
Controlling Random Distribution

- > Random distribution is rectangular by default
- > In many cases, we need other random distributions to hit all interesting cases
- > Define alternative sets of constraints
 - Use subclasses
- > Add constraints to limit the solution space
- > Add **dist** constraints to control random distribution of a variable





Distribution Constraint Example

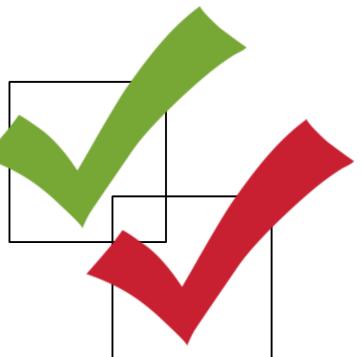






Checkers

- > Check that the DUT behaves correctly
- > Passive and independent of how the test drives the stimuli
- > Handle any legal stimuli
- > Can be split into multiple parts
 - Check different aspects
 - Check at different abstraction levels







Reference Models

- > Check by score boarding the output from the DUT against a reference model
- > Abstraction level may vary
 - Cycle exact or transaction level
- > Accuracy may vary
 - Bit exact or approximative







Coverage

- > Primary metric for measuring completeness
- > Two main types:
 - Code Coverage
 - Functional Coverage
- > Collected during simulation by the simulator
- > Analyzed to determine which aspects have not been fully verified yet





Code Coverage

- > Implicit
- > Based on the code
- > Different types:
 - Line Coverage
 - Branch Coverage
 - Toggle Coverage
 - Condition Coverage
 - FSM Coverage





Functional Coverage

- > Explicit
- > Based on specification
- > Defines interesting cases
- > Sample values during simulation:
 - Configuration
 - Input data
 - Output data
 - Internal state
 - Combinations of the above
- > Use knowledge about the design
- > Consider observability

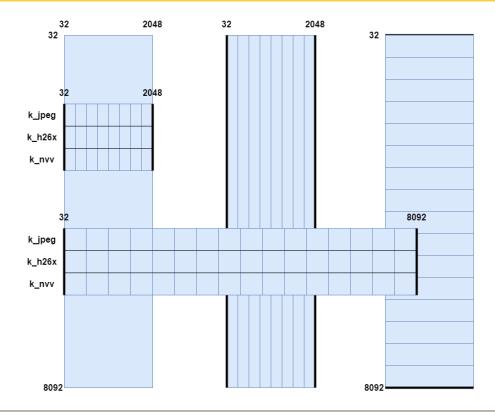






```
covergroup cg_upsamp_cfg;
  cp_subsamp: coverpoint subsamp {
    bins jpeg = {k_jpeg};
bins h26x = {k_h26x};
bins nnv = {k_nnv};
illegal_bins ib = default;
  cp_xs: coverpoint xs {
   bins min = {
                                      32};
     bins max
                                   2048};
                         =
    bins xsize [8] = \{[32:2048]\};
    illegal bins bad = default;
  cp_ys: coverpoint ys {
   bins min = {
                                      32};
     bins max
                                   8192
                =
    bins ysize [16] = \{[32:8192]\};
    illegal bins bad = default;
  cc ss xs : cross cp subsamp, cp xs;
  cc ss ys : cross cp subsamp, cp ys;
endgroup: cg upsamp cfg;
```

Functional Coverage Example



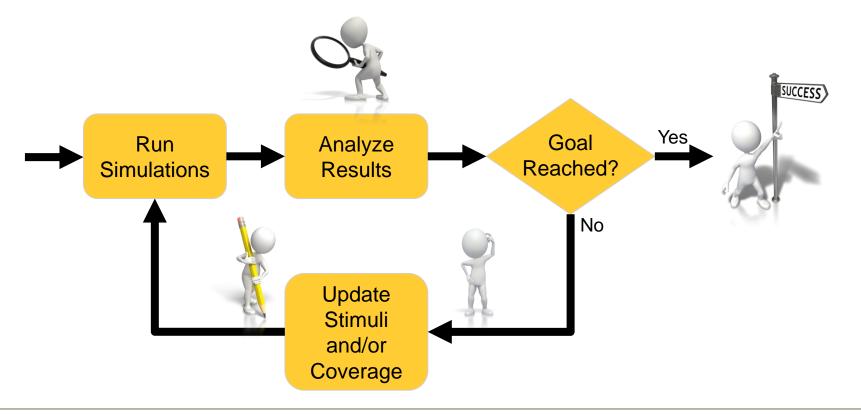
Coverage Reports

and the second se										sic-appd1 s			vdCoverag	* 12 00	b db v	+tsp : grsh — Konsole Hitis			Mon Aug 09, 17:51
View Plan Exclusion Tools Window He	selp														-				P. Menu
😐 🗔 🕃 🏅 🖬 🛥 🗇 🗙	2:000	Tree 1 10 10		3 (B)-1															
				* . m									P. 0	Center	1.60	nit:mmap_data_cov.cg_data 🦸	18-0	Conductant	18-0
rchy Modules Groups Asserts Statistics	Thata													Unco		Jh/larsviemil/src/common/mmap/tb/top/mmap_data_cov.s			· Cross 20 Pro
	(Bass												- 11					Cover Group Item Score Instances U+C U C	X Goal Weight
J) -			p. Group Score:99.97% U		1.2. 6.1	RE30 N-0							- 10	0	37	covergroup cg_data(); cp_index: coverpoint this.data.index (- G Suntt mmap_ 99.48% 230 2 2:	
		Avg. Gri	up Inst. Score:99.98% L	+C:26261 1	U12 C12	6259 X12									39	bins b1_index[3] = ([0:2]);		- C cp_index 100.00% 3 0	d 100% 1
roup		Score		U+C U		×	Goal V	At sight	Least Per	inst Over	lap AutoB	ini Missis	Commer	10	40) cp_start: coverpoint this.data.start (0 100% 1
Sunki/h/larsviemil/src/common/mmap	pith/topimmap_data	cov.	100.00%	6 210	0	210	0							1	42	bins bi_start[64] = cp_start;		- Of cp. queue. 100.00% 30 0 3	0 100% 1
Suniti/Pylarsw/emil/src/common/me			100.00%		0	26	0 100%	1	3	1		64 e			43) // The coverage for max size of allocated region is not neces		- Ca cp size 55.88% 64 2 6	
- Ga Suniti/hvlarsw/emil/src/common/mm			100.00%		0	74	0 100%	1	1	1		64 6			-44	ry when coverage model has separate bins for		- C cp_start 100.00% 64 0 6	0 100% 1
- Go \$uniti/9xlarsv/emil/src/common/inm			100.00%		0	110	0 100%	1	1	1		64 (45	// nize and start that defined for remapping areas and has b	binn 冒	Cr. cp_start_p 100.00% 65 0 6	0 100% 1
Sunit ax_axi_data_rov cg_addr_bit: \$P			100.00%		0	184	0 100%	1	1	1		64 (for maximum of intert + size)			
Sunt ax_axi_data_cov cg_burst SHAP			100.00%			6804	0 100%	1	1	1		64 (47	Dinp bi_size[64] = cp_size;			
Con Sunit ax_axi_data_cov cg_data_bit SH		size	100.00%		0	324	0 100%	1	1	1		64 6			48 49			e.	
Sunt:ax_axi_pin_cov#i64,36,4.4):cg_n		-	100 00%		0	2	0 100%	1	1	1		64 (a	49	<pre>cp_irq_level: coverpoint this.data.irq_level { bins bi_level[] = ([1:this.data.q_size]);</pre>		Status Bin Name Type At Least Size Hit Count	
- 55 \$unit:ax_axi_pin_cov#164,36,4,41::cg_t	timing		100.00%	3 (9)	0	70	0 100%	1	1	1		64 6			51	1			A
Sunit mmap_data_cov cg_axi_ont			100.00%	3	0	3	0 100%	1	1	D		64 6		0	52 53	<pre>cp_queue_map_mize1 coverpoint this.data.queue_map.mize() { bins bi_qmap_mize(] = ((1:this.data.mar_mize)))</pre>		busize[5:0000005:5/////// User 1 1	
Sunt mmap_data_cox cg_axi_mmap			100.00%	1024		1024	0 100%	1	1	0		64 6			54	1		- W bi_size[64000000 67ffffff] User 1 1	
Sunit mmap_data_cov cg_data			99.46%	230	2	228	0 100%	1	1	0	1	64 6	4		55			- V bi_size[680000000 6bffffff] User 1 1 1	
- Gr cp_index			200.00%	3	0	3	0 100%	1	1		1	0		0	56	<pre>cp_start_p_size: coverpoint (1'h0, this.mask(this.data.sta } + (1'h0, this.mask(this.data.size)) {</pre>	arta	- zz bi_size]6c000000 6ffffffff User 1 1	
- O cp_irg_level			100.00%	4	0	4	0 100%	1	1			0			57	bins bi_sps[64] = [[36"h0_0000_0001:36"hf_ffff_fff]];			
- 🔄 cp_queue_map_size			100.00%	30	0	30	0 100%	1	1		1	0			58	bins bi_mex = (37'h10_0000_0000);		- w bi size[780000000 7bffffff] User 1 1 1	
C cp_size			95.80%	64	s	62	0 2005	1	2		1	0			60	endgroup : cy_data;		- vi bi_size[7c000000.7ffffff] User 1 1	
- Ge_start		_	100.00%	64	0	64	0 100%	1	2		1	0			61			- V bi_size[800000000:83fffffff] User 1 1	
Cp_start_p_size			100.00%	65		65	0 100%	1	1		-	0			62	covergroup cg_queue_data; cp_tgt_addr: coverpoint this.queue_data.tgt_addr 1		bi size[84000000 87////// User 1 1 1	
Ce Sunit mmap_data_cov cg_queue_data			100.00%	9557		9557	0 100%	1	1	0		64 6			64	bins bi_tgt_addr[256] = cp_tgt_addr;		b sizelectodobo esminin oser 1 1	
Ge \$unit:mmap_data_cov.cg_r_cur_map			100.00%	128	0	128	0 100%	1	1	0		64 6			65	1			
Sunt mmep_deta_cov cg_r_cur_offs			300.00%	16	0	16	0 100%	7	1	0		64 6			67	<pre>cp_map_mize: coverpoint this.queue_data.map_mize { bins bi_one = {@'h011;; }</pre>		- 25 bi_size[940000000.97fffffff] User 1 1	
Sunit mmap_data_cox cg_r_hang_add			100.00%	64	0	64	0 100%	1	1	0		64 6 64 6			68 69	bins bi_mep_size[] = ([8,05:8,011]);			
															73	cp_cur_offs: coverpoint this.queue_data.cur_offs_add this.data.start [ldr -	bi_size[ac0000000:afffffff] User 1 1 bi_size[b0000000:b3fffffff] User 1 1	
															74 75	<pre>bins bi_cur_offs[256] = cp_cur_offs; }</pre>		ber bissebetaan bissebeta	
															75 76] cp_trans_total: coverpoint this.queue_data.trans_total		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	
														Covier	75 76] cp_trans_total: coverpoint this.queue_data.trans_total		bi_size(b85000000 bbffffff) User 1 1	2
Ion Manager													8	Covision	75 76] cp_trans_total: coverpoint this.queue_data.trans_total		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	# 2 - 0
ien Manager.														Covision	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	±8-0
	Details A	nnetation	Signature 1	This									•	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	≠8-0
	Details A	nnetation	Signature	the									•	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	*8-0
lame 0 Sunit.e_master_timing Cover.Group: and n	ready_before	nnetation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	18-0
ame 0 Sunit e_master_timing Cover Group an n Cover Group : an y		nnstation												1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(b000000 bHfffff) User 1 1 b_size(b000000 bHfffff) User 1 1	* <i>8</i> - 0
ame 0 Sunit e_master_timing Cover Group: am n Cover Group: am v Ax_mbf	ready_before	nnetation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	* 8- 0 1
ame 0 ■ Sunit e_master_timing Cover Group: an n Cover Group: an v an, abdf Sax, ad immap	ready_before	nnekation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	* 8° - 0
ame 0 Cover Group as n Cover Group as n Cover Group as n Sax, add Sax, ad, immap Sax, ad, immap	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	* 8 1
ame 0 Cover Group: an Cover Gr	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	48-a
ame 0 Cover Group: an Cover Gr	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	*@
ame 0 South a master Sming Cover Group, and Cover Group, and Cover Group, and Cover Group, and Sac, and mmang, mapper Sac, and mmang, front Sac, and front Sac, and front Sac, and front Sac, and front Sac, and front Sac, and	ready_before	nnatation .	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	#8- 0 1
ame 0 Sports _master_Sming Cover Group as n Cover Group as n Cover Group as n Cover Group as n So, and prima So, and pr	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	4 E - 0
ama 0 0 Court Group as n Court Group as n Cou	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	48- 0 1
And Control of the second of t	ready_before	motation	Sunit e_master_timing	/mm-ap/tb/t									-	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	*8- 0 1
Anno. Construction of the second seco	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	*Ø
South a preserve forming South a preserve forming Concer Orace (and Concer Orace (andd Co	ready_before	nnobation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	*8- 0 1
By loss a, name some D By loss a, name some n Cover fores : a n By loss a, name B	ready_before	matation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	*@
۱۹۹۰ <	ready_before	nnatation	Sunit e_master_timing	/mm-ap/tb/t									-	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	* <i>8</i> - 0
Construction C	ready_before	notation	Sunit e_master_timing	/mm-ap/tb/t									2	1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	* <i>8</i> - 0
الله الله الله الله الله الله الله الله	ready_before	nnotation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	18-0
هود می میده به دسم این از	ready_before vald_wth_ready	matation	Sunit e_master_timing	/mm-ap/tb/t										1	75 76 c 1 Hvs) cp_trans_total: coverpoint this.queue_data.trans_total 2		b_size(bs000000 bbffffff) User 1 1 b_size(bc000000 bffffff) User 1 1	18.





Closing Coverage





Universal Verification Methodology

- > Framework for building coverage driven constrained random test benches
- > Consists of a class library
- > Std IEEE 1800.2
- > Amalgamation of several previous competing frameworks







Code Reuse

- > Test benches are complex software systems
- > Large amount of code
- > Many test benches for a chip
- > Code reuse is essential
 - Across similar test benches Horizontal
 - Across hierarchy levels Vertical
 - Across iterations of the chip







Formal Verification





Formal Verification

- > Use of tools that analyze the space of all possible behaviors of a design
- > Excellent complement to simulation for selected problems
- Complexity often an issue for real world designs







Summary

- > Verification is an essential part of developing integrated circuits
- > Active area
 - New techniques
 - New tools
- > Requires both hardware and software knowledge





THANK YOU

We have more than 50 thesis proposals within:

AI Machine Learning Image Quality Audio Quality Supply Chain Management **UX Design** Electronics ASIC Testing Mechanics

Software Algorithms Radar Lidar Audio Firmware Deep Learning Image Processing Analytics Computer Vision

