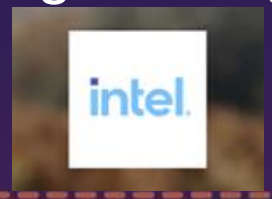
Formal Property Verification (FPV) Deployment on Xeon SoC owned IPs

Prakeerthi Jallipalli - Formal Verification Lead/Manager
Xeon Server Engineering(XSE) Group



62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

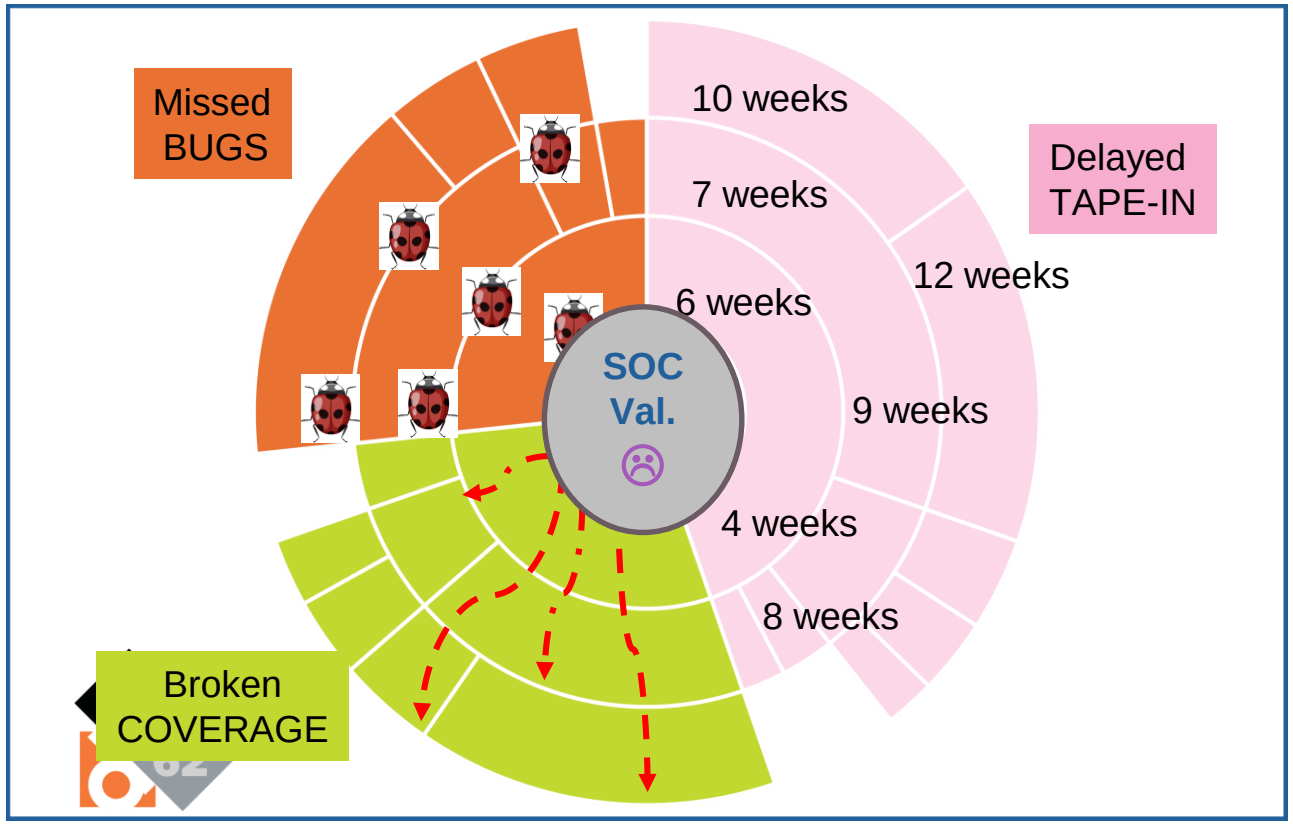
FPV Motivation on Xeon SoCs and it's IPs

Server/AI Accelerator

Integrate 190+ IPs

SoC Integration Team

Owns & Develops "26 IPs" in-house
- Baseline(60%) & Derivative(40%)
- Comprehensive validation is complex.



SOLUTION

FPV • Formal Property Verification

Cadence • Jasper Tool

SVA • System Verilog Assertions

Early Bug Detection

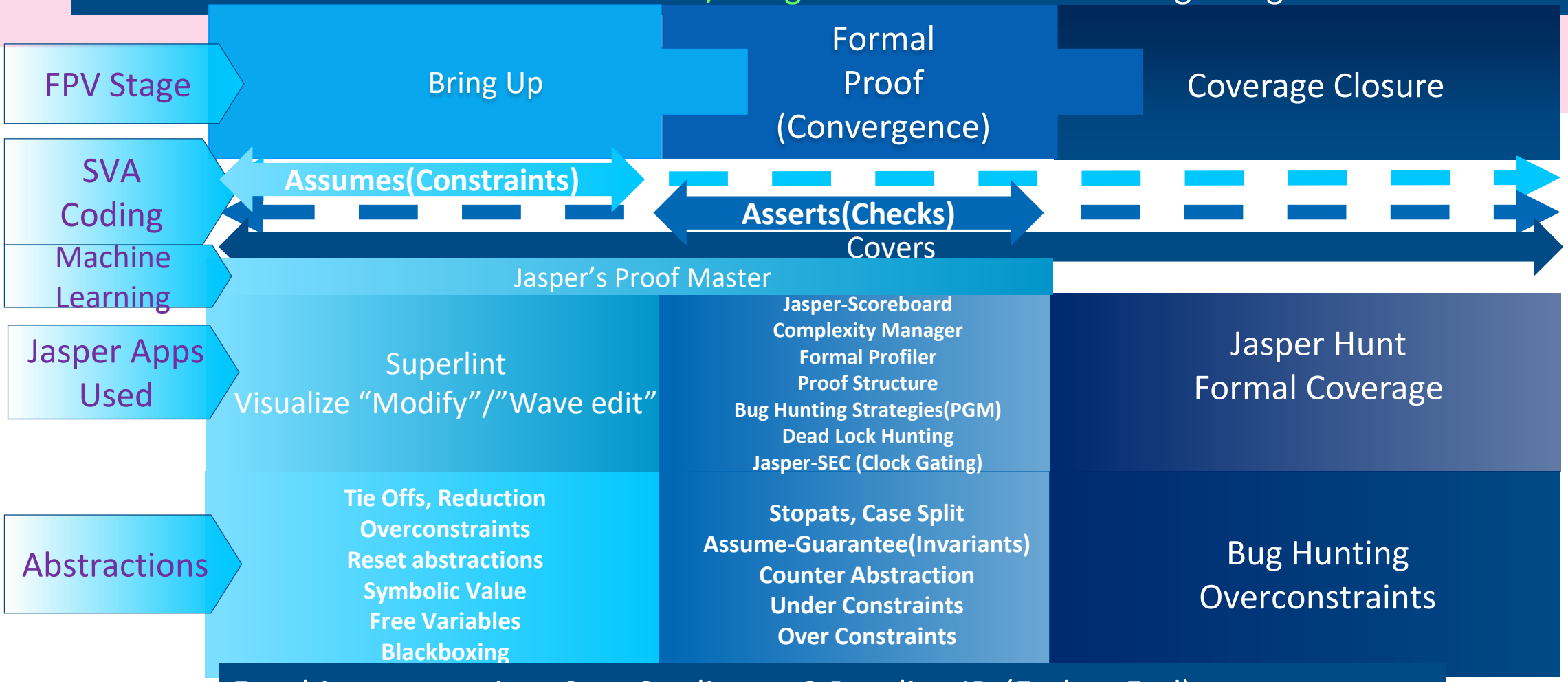
Exhaustive Validation

Critical areas of logic

Out of 26 IPs "17 IPs" are identified for FPV

FPV Idea & Methodology on Xeon SoC IPs

FPV across diverse Xeon SoC owned IPs(17 IPs from 16 different functional domains)
Cross-functional collaboration with Archs./Designers and Sim teams to align on goals and fill validation gaps.

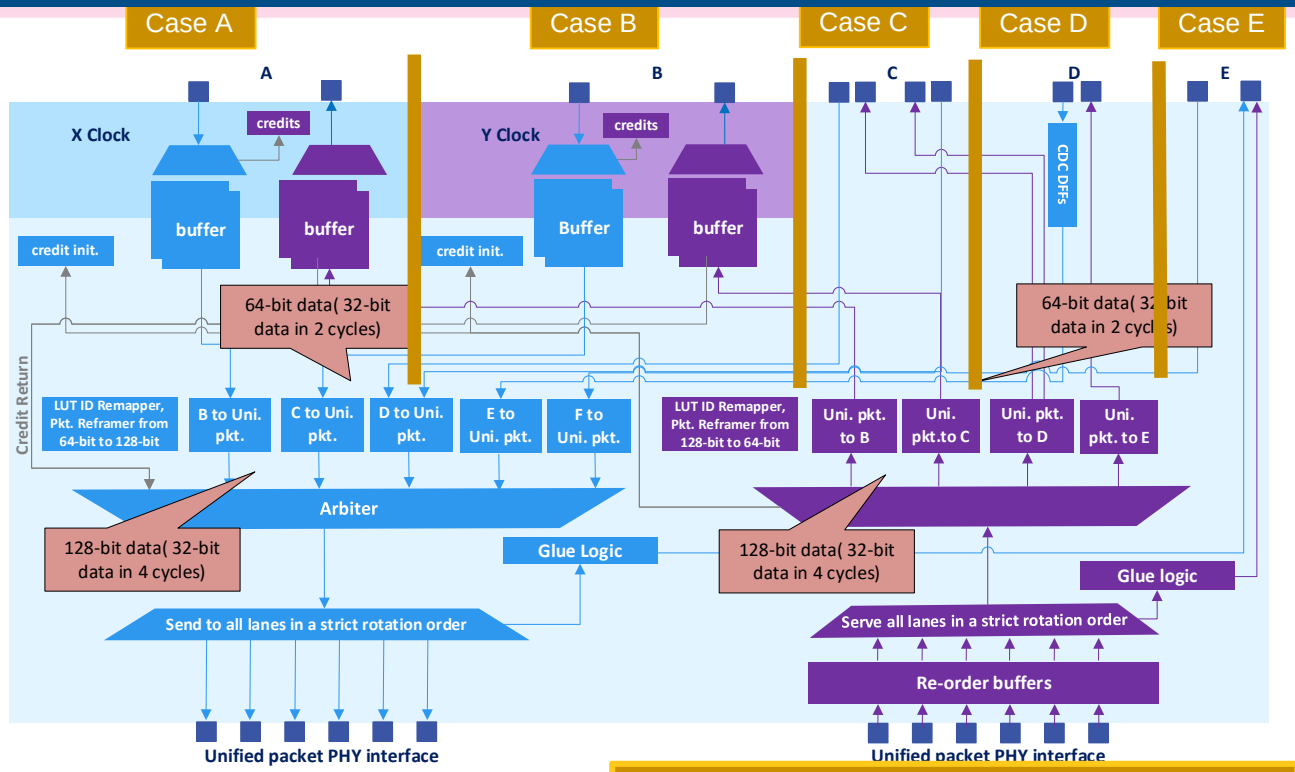


For this presentation, Case Studies on 3 Baseline IPs(End-to-End):

1. Remapper Bridge(45k flopcount)
2. SRAM Controller(8k flopcount SRAM logic embedded in 34k flopcount IP module)
3. Data Processing Unit(70k flopcount)

Case Study 1 - FPV on Remapper Bridge – Arbiter, Reachability Analysis

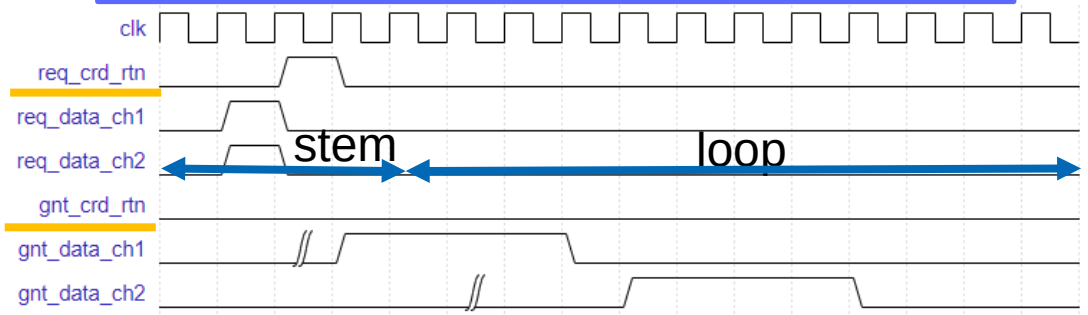
- **Goal**: Complement Simulation, achieve 100% design coverage and help validate critical areas of design logic
- **Techniques**: Symbolic abstraction, case-splitting, Assume-guarantee reasoning



Input on Egress -Verilog modelling for all 9 Virtual Channels(VCs)

Remapper Bridge Coverage Phase	Total SVA Properties		
	Assumes	Asserts	Covers
Before Reachability Analysis	472	613	2492
After Reachability Analysis	483	614	3087
Total New Covers Added			595

Jasper CEX : Liveness assertion on Mixed arbiter scheme(RR&FP) Arbiter



Arbiter **BUG – extreme corner case scenario**

- Effort of 4 weeks (well before Simulation team) (1 Engineer)
- **6 Bugs** Found
- And Significantly increasing Designer's Confidence to sign off on RTL

Case Study 1 - FPV on Remapper Bridge – Clock Gating verification

- **Goal**: To validate Clock Gating feature
- "Jasper – SEC" App – "RTL-to-RTL" comparison
- 'Chicken bit' enabled on SPEC side and disabled on the IMP side
- **Methodology** : "Hierarchical Clock Gating "
- Effort of 6 weeks with Complete Proof on each instance.
- Found **3 unique critical bugs** – "very expensive"
- Sim. team couldn't reproduce these bugs - Used Machine Learning(ML) to reproduce these which also found bug in their Egress BFM.
- Analysis on these bugs if found at :
  - SoC level – would take "~6 to 8 weeks" to root-cause, debug, fix and re-validate
  - Post-Si level – very expensive - millions to billions of $s depending upon what stage of Post-Si it would be discovered.

Case Study 2 – FPV on SRAM Controller

Goal: Validate Controller & entire 4K memory(four 1k SRAMs) without blackboxing.

Immediate Question: Four 1k Memories – can FPV handle it ?

Immediate answer: NO (simple and easy solution! ☺). Still went ahead & validated.

Methodology: Start small and gradually scale it to see if Jasper tool can converge the entire 4k memories

```
//Symbolic value abstraction on 'addr' & 'data'
    asm_addr: assume property (##1 $stable(addr));
    asm_data: assume property (##1 $stable(data) );//Use  either this or "asm_data_bb" (below) to get faster convergence(not both)

//Constraint 'data'  using Bit-blasting technique
    generate (n=0;  n<DATA_WIDTH; n++)   //DATA_WIDTH is 68-bits
        asm_data_bb: assume property(##1 $stable(data[n]) );

//Check data-integrity from SRAM read_data on 'read' command using Bit-blasting
    generate (n=0;  n<DATA_WIDTH; n++)
        ast_data_bb: assert property(SRAM_rd_data[n] == data[n]) ;
```

- By choosing the right "abstraction technique and the methodology", full convergence on entire 4k Memories without Blackboxing any of them.

Proof Time on 'Data Integrity' Checks for various sizes of SRAM Banks		
addr[1:0] is used to select SRAM bank i.e., 2'b00 - **SRAM0**, 2'b01 - **SRAM1**, 2'b10 - **SRAM2**, 2'b11 - **SRAM3**		
Num. of Locations * 4 SRAMs	**Constraint on Address[11:0]**	**Wall Clock Time For Full FPV Proof**
16 *4	addr[11:6]==6'b000000	~1.25 Hrs
256 * 4	addr[11:10]==2'b00	~2.5 Hrs
1024 * 4	NO constraint on addr[11:0]	~4.25 Hrs

- Effort of 7 weeks (1 Engineer)
- 3 Bugs Found

Case Study 3 – FPV on Data Processing Unit(DPU)

Goal : Enhanced path exploration (where Simulation is unable to hit) using unique SVA covers.

Methodology : 'Divide and Conquer' method (on each RTL parameter settings – 3 in total)

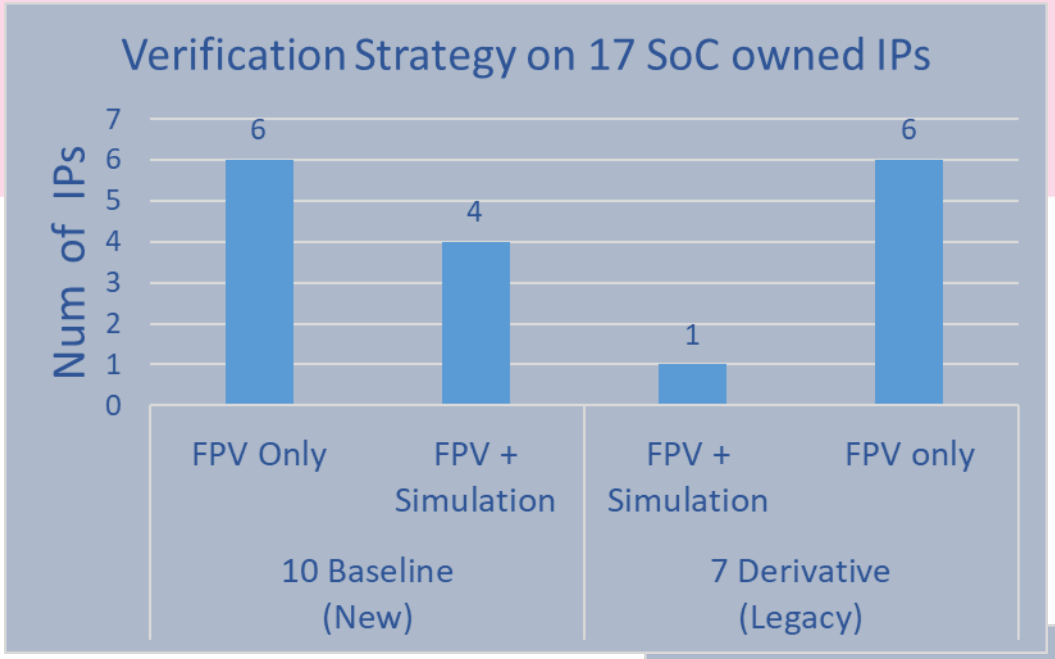
Note : this is NOT a 'Case-splitting' scenario since each RTL slice needs different RTL parameter value setting.

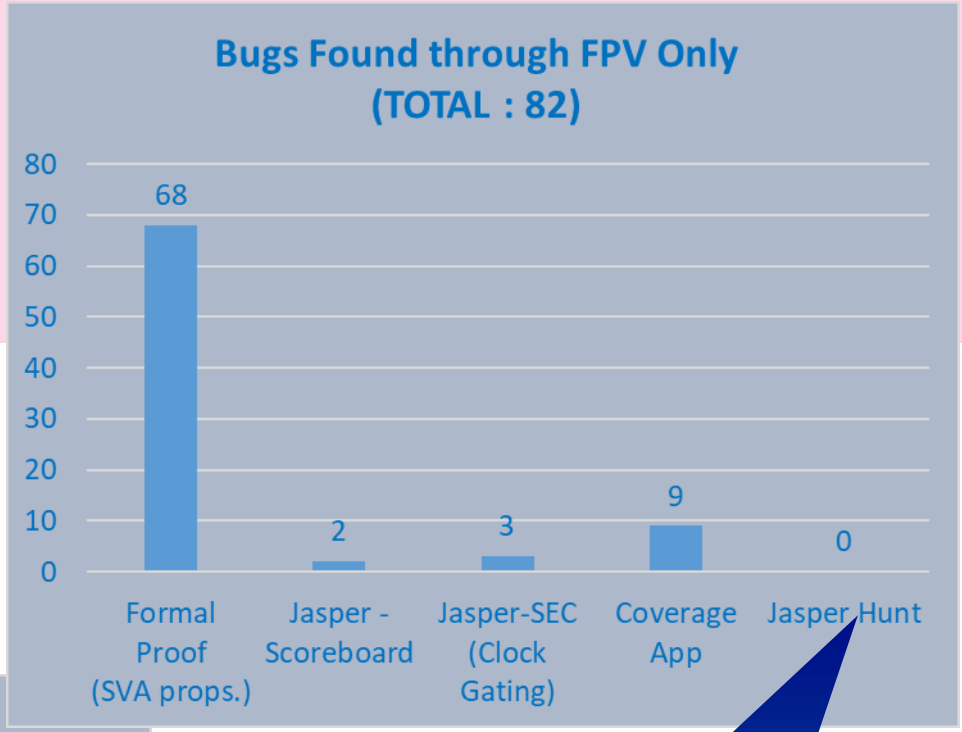
Abstractions : Blackboxing, Tie-Offs & Overconstraints

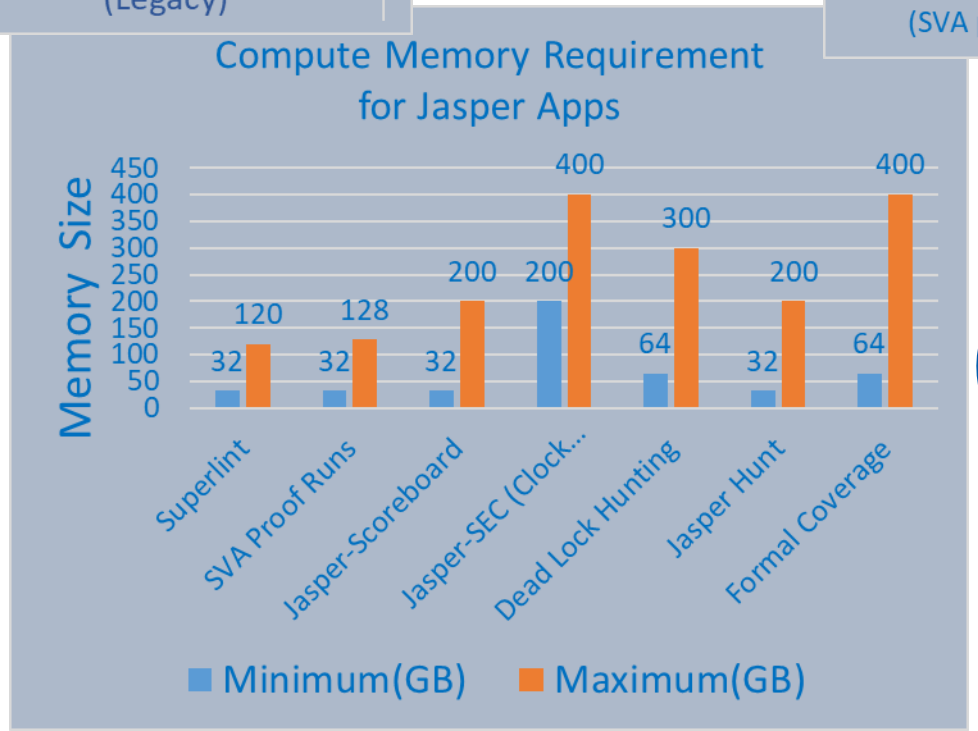
- Unique Covers added in FPV for each RTL path which remain 'unexplored' in Simulation.
- Total of "220 Unique SVA covers" in FPV added to explore all unexplored paths for all 3 slices combined.
- 3 unique bugs found
  - 1 Bug – One of the cover was unreachable in Jasper FPV which was expected to be 'covered'
  - 2 bugs – Negative covers written(expected to be unreachable) but resulted as 'covered'
- Entire effort was done in 6 weeks.
- Analysis on these bugs if found at :

  "**SoC level**" – to root-cause, debug, fix and re-validate would take ~5 to 7 weeks per Bug.

Results and FPV Stats

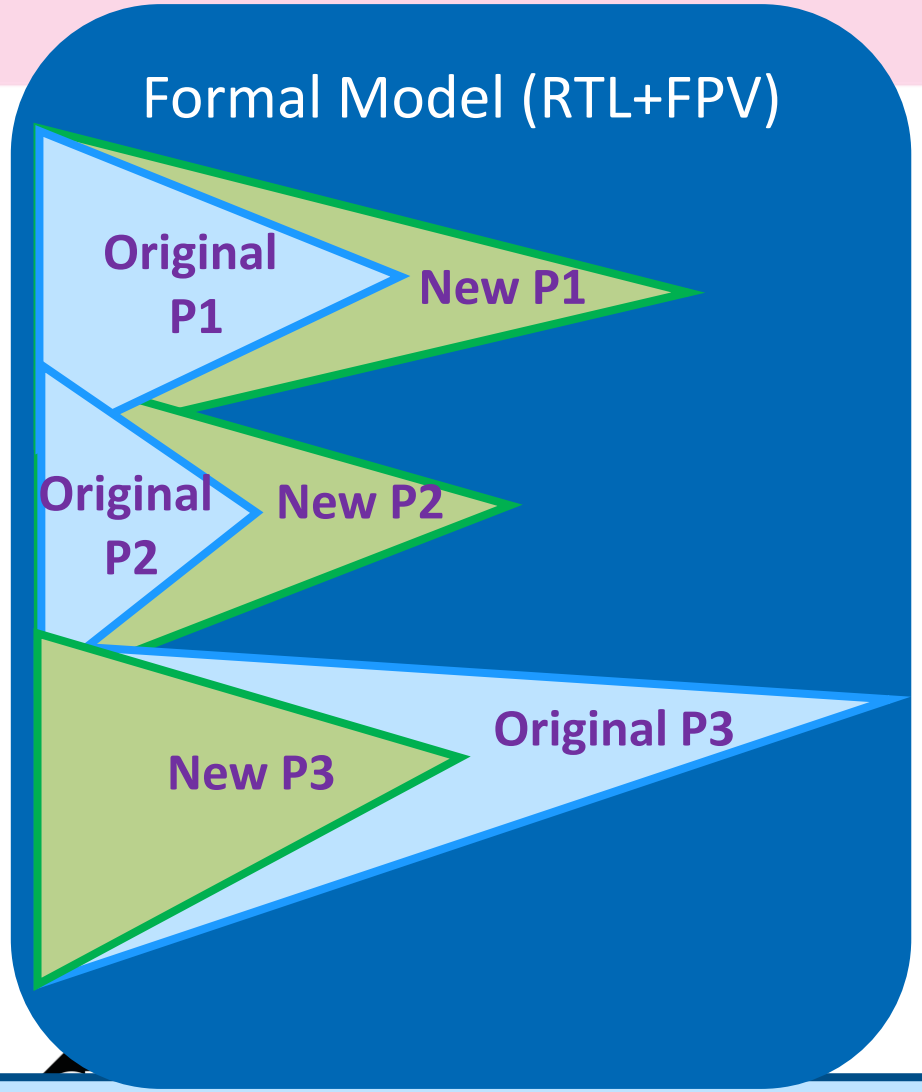




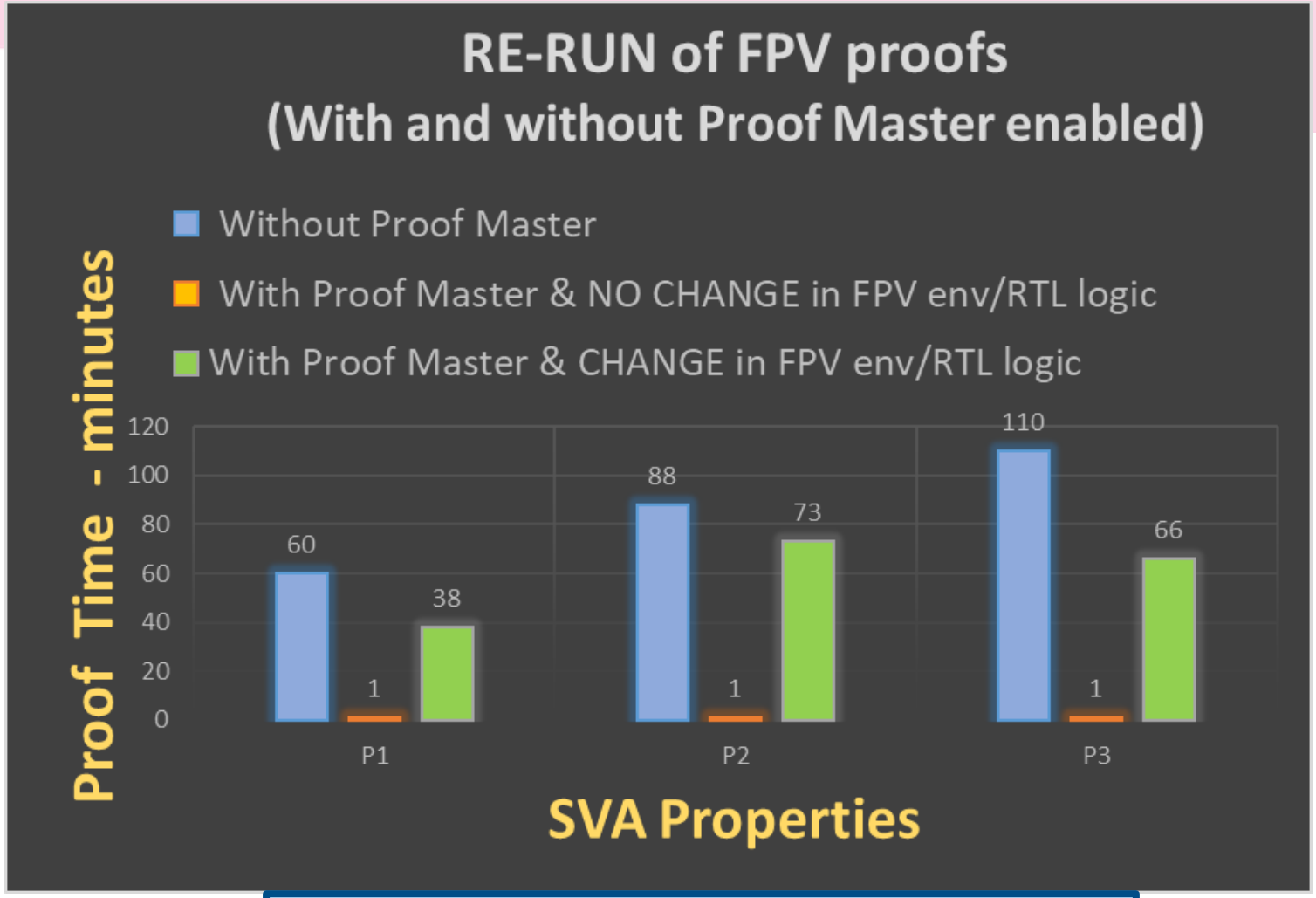


Learning: We started 'Jasper Hunt' very late in our Validation Cycle due to other priorities and challenges!

Machine Learning usage in FPV and it's ROI on Proof time



Cone of Influence(COI) of SVA properties P1, P2 & P3



Proof Time Improvement with Proof Master

Evidence of FPV deployment and it's ROI on Xeon IPs

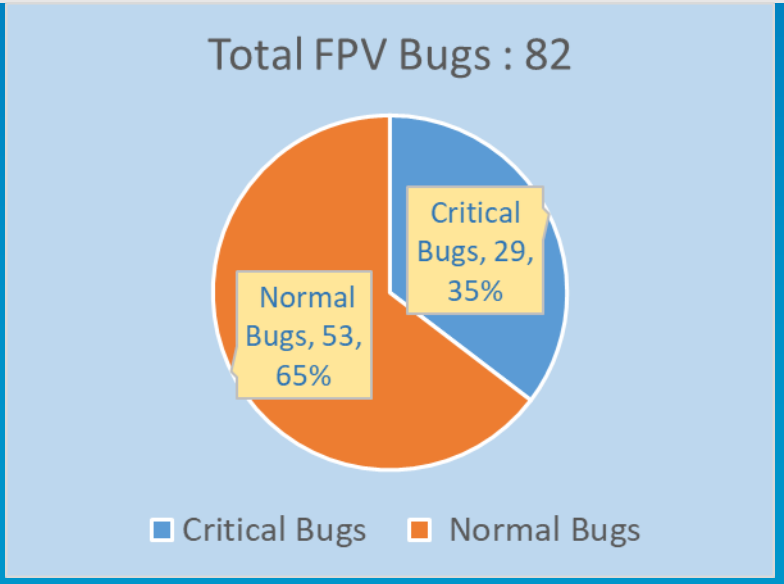
Quantitative Outcomes

- Remapper Bridge:
  - 3 bugs found in Arbiter logic
  - 6 bugs discovered during 'Reachability Analysis' after 90% simulation coverage
  - 3 critical Clock Gating bugs identified
- SRAM Controller:
  - Full Convergence on entire 4k memories
  - 3 unique bugs found in 7 weeks
- Data Processing Unit:
  - Identified 3 critical bugs within 6 weeks

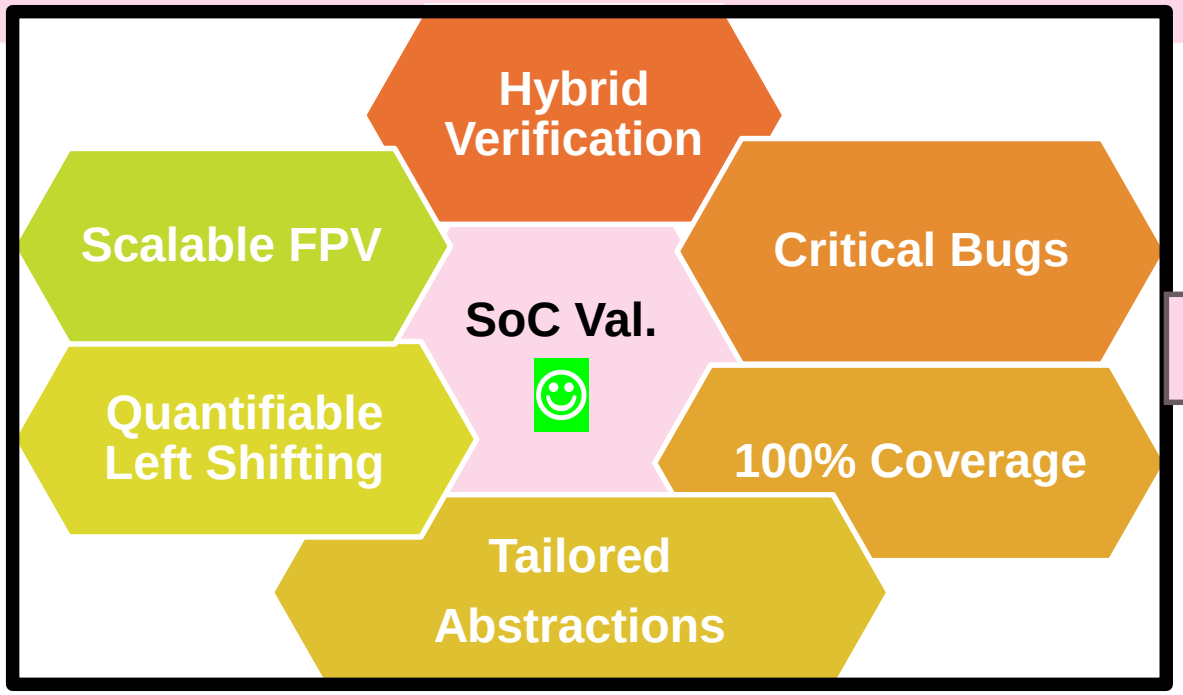
	Simulation Only		With FPV		ROI (Savings)	
Case Studies (3 Xeon IPs)	Resources (Engineers)	Time (weeks)	Resources (Engineers)	Time (weeks)	Resources (Engineers)	Time (weeks)
Remapper Bridge	2	16	2	8	0	8
SRAM Controller	1	12	1	7	0	5
Data Processing Unit	2	15	1	6	1	9
				Total Savings (3 Xeon IPs)	1	22

ROI Metrics on all 17 IPs:

- **29 Critical Bugs found through FPV**
  -> difficult to find through Sim/Emu methods
- **Significant decrease in Post-Silicon Debug time**
  -> 6 to 8 weeks saved per each of the 29 bugs.
- **Significant Left-shift i.e., 58 weeks(or 4 quarters) saved**
  -> compared to Simulation-only approach
- **FPV is now Critical for A0 PRQ Validation Strategy**
  -> Saving millions of $s for any Xeon product line



Summary - Contributions, Impact, Future Work



IMPACT

FPV as Critical Val Strategy for
- All future Xeon SoC product lines
- And their IPs for A0 PRQ

Time-to-Market(TTM) Improvements
- By saving several quarters of SoC debug time

Significant Cost Savings
- By preventing SoC level bugs &
- Costly Post-Si bugs

Future Work - Advancing the usage of Machine Learning (ML) to address additional FPV challenges:

1. Improve Efficiency - All combinations of knobs are tested to catch failures in already proven env
2. Reduce Compute - Generate unique CEXs with lowest bound
3. Reduce Debug – Avoid duplication of failing signatures
4. Reproduce Post-Si failures in FPV env with minimal effort