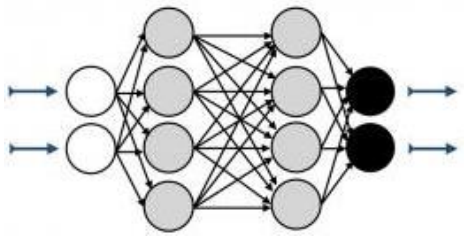
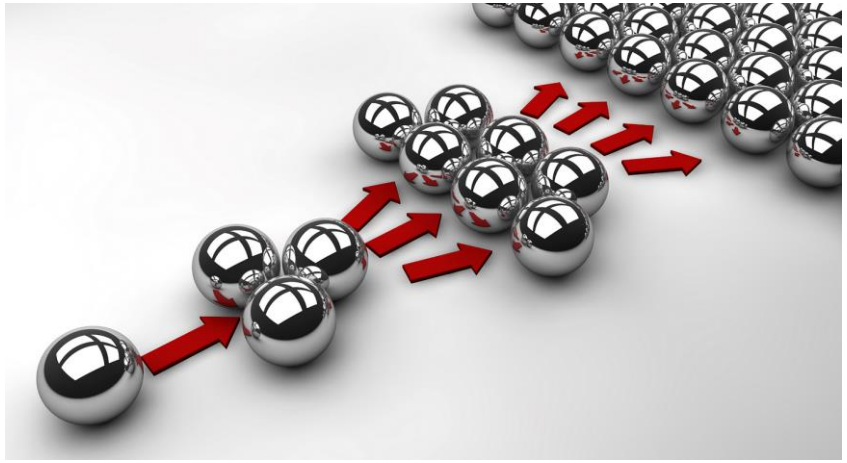
Brajmohan Sharma

Senior Staff verification engineer

Marvell Technology

Advanced formal verification and robust regression strategy for highly parameterized design





MARVELL

Essential technology, done right™

DV club

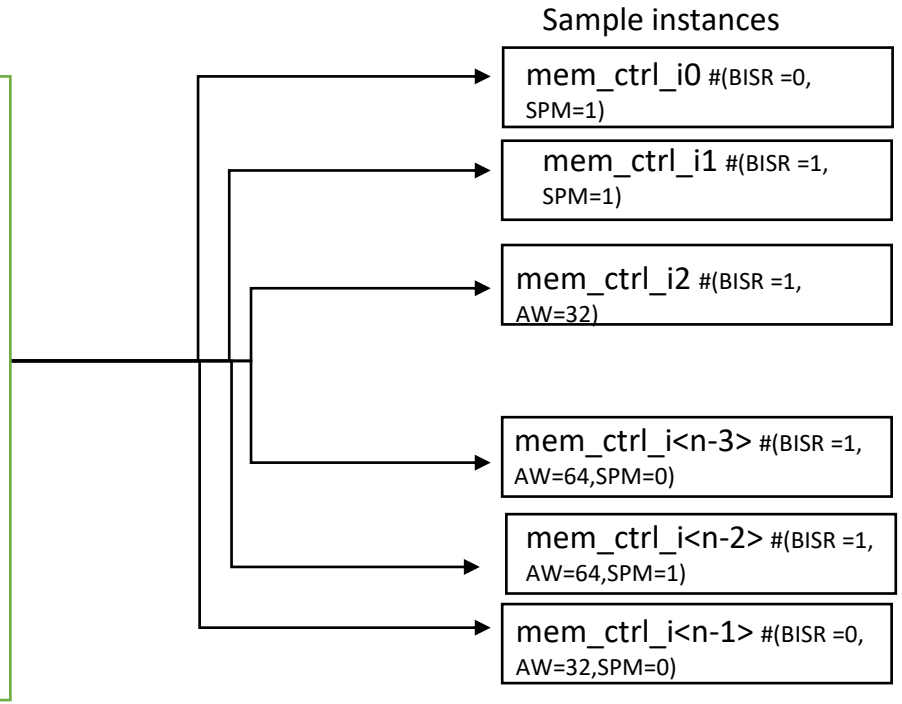
Agenda

- Introduction
- Problem statement
- Proposed Solution
- Result
- Key takeaways

Introduction

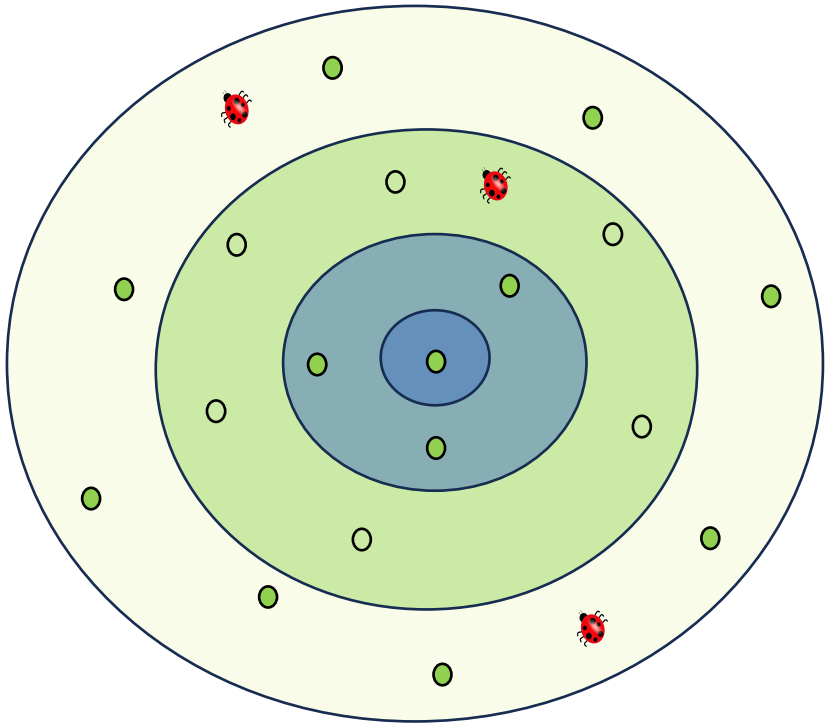
- Parameterized design supports diverse design variants for specific goals.
- It promotes scalability and expands the design space beyond traditional methods

```
module mem_ctrl #(
parameter BISR = 0,
parameter ROW = 16,
parameter AW = 10
parameter SPM = 1,
…..
)(I/O);
//-----*Design Logic *-----
endmodule
```



Introduction

- For a CPU event controller with 100s of parameters, there can be millions of design variants.
- It's crucial to verify all logic and design variants for a complete sign-off
- Formal verification
  - Formal verification uses mathematical proofs to validate correctness of design
  - Exhaustively verifies design functionality
  - Easy to find the corner cases
  - Provides higher confidence in design correctness in comparison to traditional methods



Problem statement

Problem statement

- Characteristics of a highly parameterized design
  - Multiple design variants
  - Example block mem_ctrl (16 parameters)
  - Total design variants 2^16 {unique and valid ~50}
- Challenges with Formal verification?

Challenge -1
▪ Creation of a parameterized Formal Testbench

Challenge-2
▪ Verification of all the design variants

Proposed Solution

Challenge-1

Creation of Formal TB

- Formal modeling:
  - Use of parameters inside auxiliary code
- Formal properties:
  - Use of parameters inside the properties
  - Enable/disable properties based on parameter

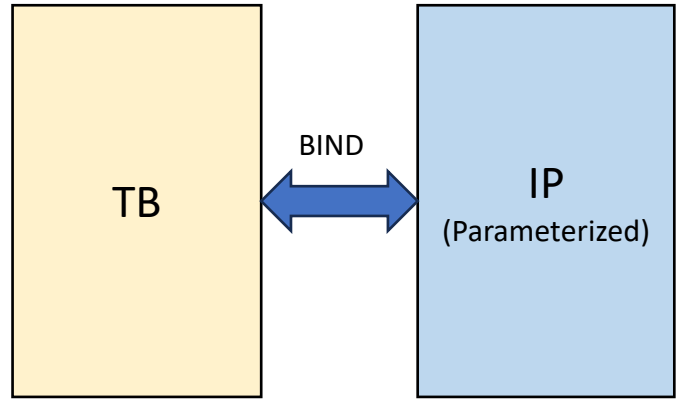
```
property ctrl_wr_ram_check(ram_we);
    @(posedge  wr_clk) disable iff (!wr_rst_n)
      (!SECURE_MEM && !CTRL_WR_DIS) && ram_we |-> ##WR_LATENCY
(ctrl_we);
endproperty
generate
for( bank = 0; bank < NUM_BANK; bank = bank+1)
if(! CTRL_WR_DIS)
   as_ ctrl_wr_ram_check : assert property(ctrl_wr_ram_check(ram_we[bank]));
```

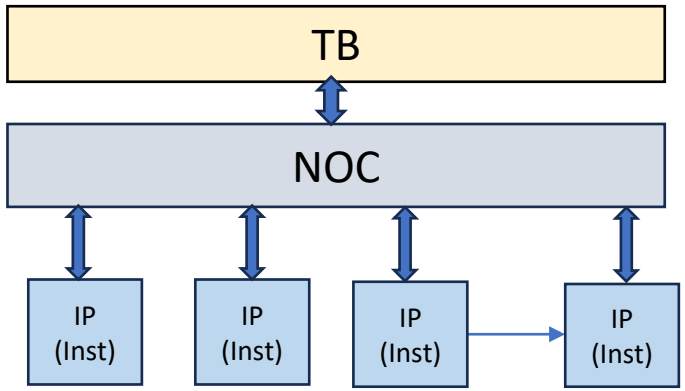
```
//write control- formal tracker
 genvar bank;
 generate
 for( bank = 0; bank < NUM_BANK; bank = bank+1)
  always_ff @(posedge wr_clk) begin
   if (!wr_rst_n) begin
     ctrl_wdata_sampled[bank] = 0;
     ctrl_bwe_sampled[bank]  = 0;
     ram_wb_predict[bank]    = 0;
     ctrl_wr_ongoing[bank] = 0;
   end
    else begin
   if (ctrl_we[bank] && (!SECURE_MEM && !CTRL_WR_DIS))
   begin
      ctrl_wdata_sampled[bank] = ctrl_wdat[bank];
      ctrl_bwe_sampled[bank]  = ctrl_bwe[bank];
      ram_wb_predict[bank]  = {(DW+1)/2{ctrl_bwe[bank]}};
      ctrl_wr_ongoing[bank] = 1;
   end
    if (ctrl_wr_ongoing[bank]  && ctrl_wr_open && (SPM == 1))
    begin
      ctrl_wr_ongoing[bank] = 0;
```


Proposed Solution
Challenge-2

Recap Challenge-2

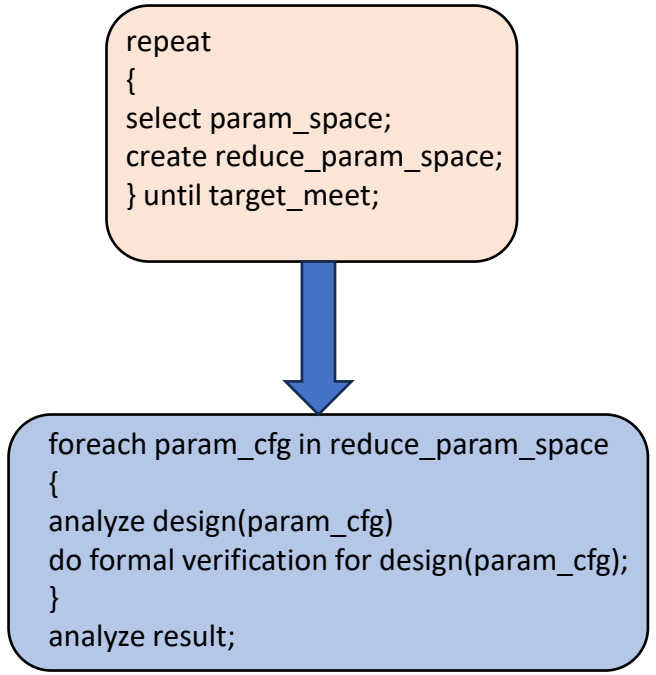
- Verification Goal – To find bug during early phase of design development
- Block level simulation: Different database with multiple parameter combination
- Formal verification (block level) : Also necessitates different databases
- Block level : Verifying all design variants poses a challenge
- Full chip simulation : Covers all variants; not exhaustive

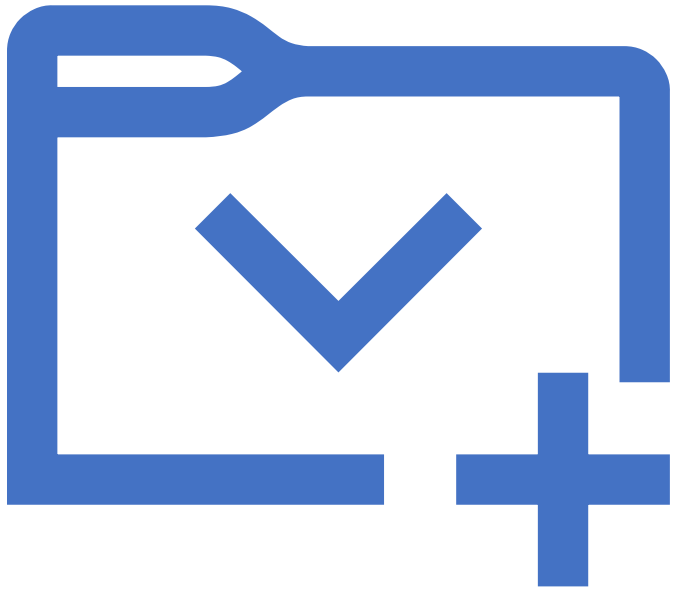




Proposed solution-Verification strategy

- Proposing Formal parametric regression
- Running a formal regression with all supported parameter



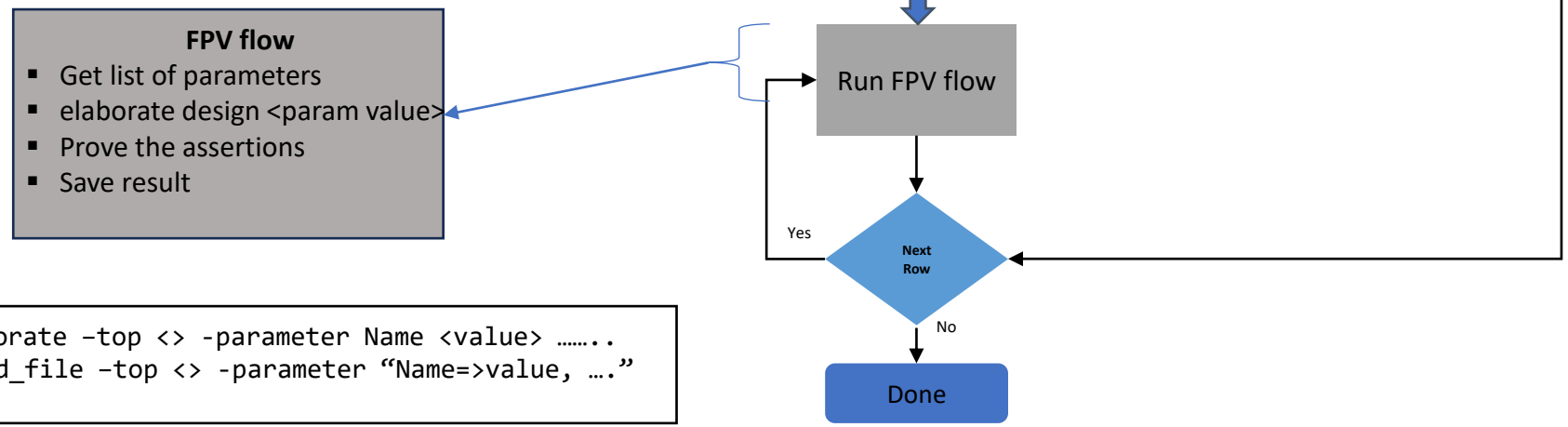


Formal Parametric regression

Parametric regression

- Elaboration – Parameter can be passed
- Bind TB and DUT along with parameters
- Create List of all valid parameter combination

No.	BISR	ROW	CTRL_WR_DIS	SPM	AW	SECURE_MEM	WR_LATENCY
1	0	16	1	1	6	0	2
2	0	32	1	0	6	0	1
3	0	32	1	1	8	1	0
4	0	64	0	0	10	1	2
5	1	64	1	1	8	0	1

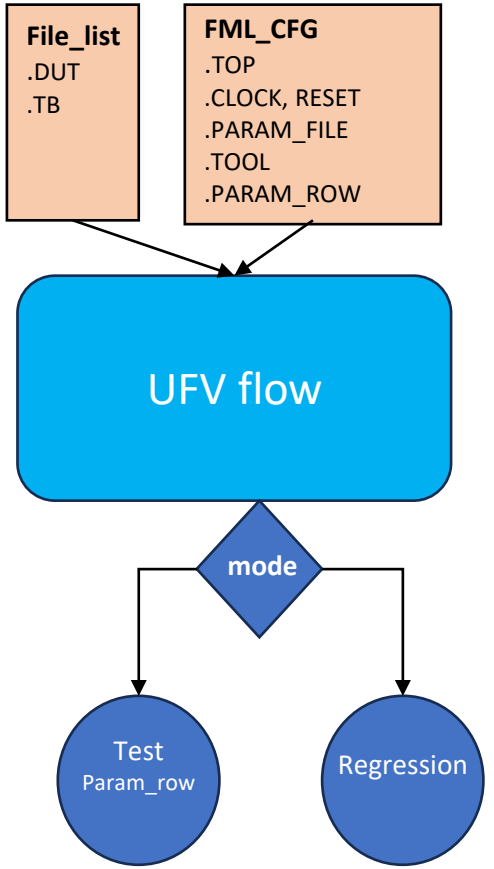



```
JG> elaborate –top <> -parameter Name <value> ……..
VCF> read_file –top <> -parameter “Name=>value, ….”
```

Note: Implemented Unified formal verification flow to automate this

Unified Formal verification flow

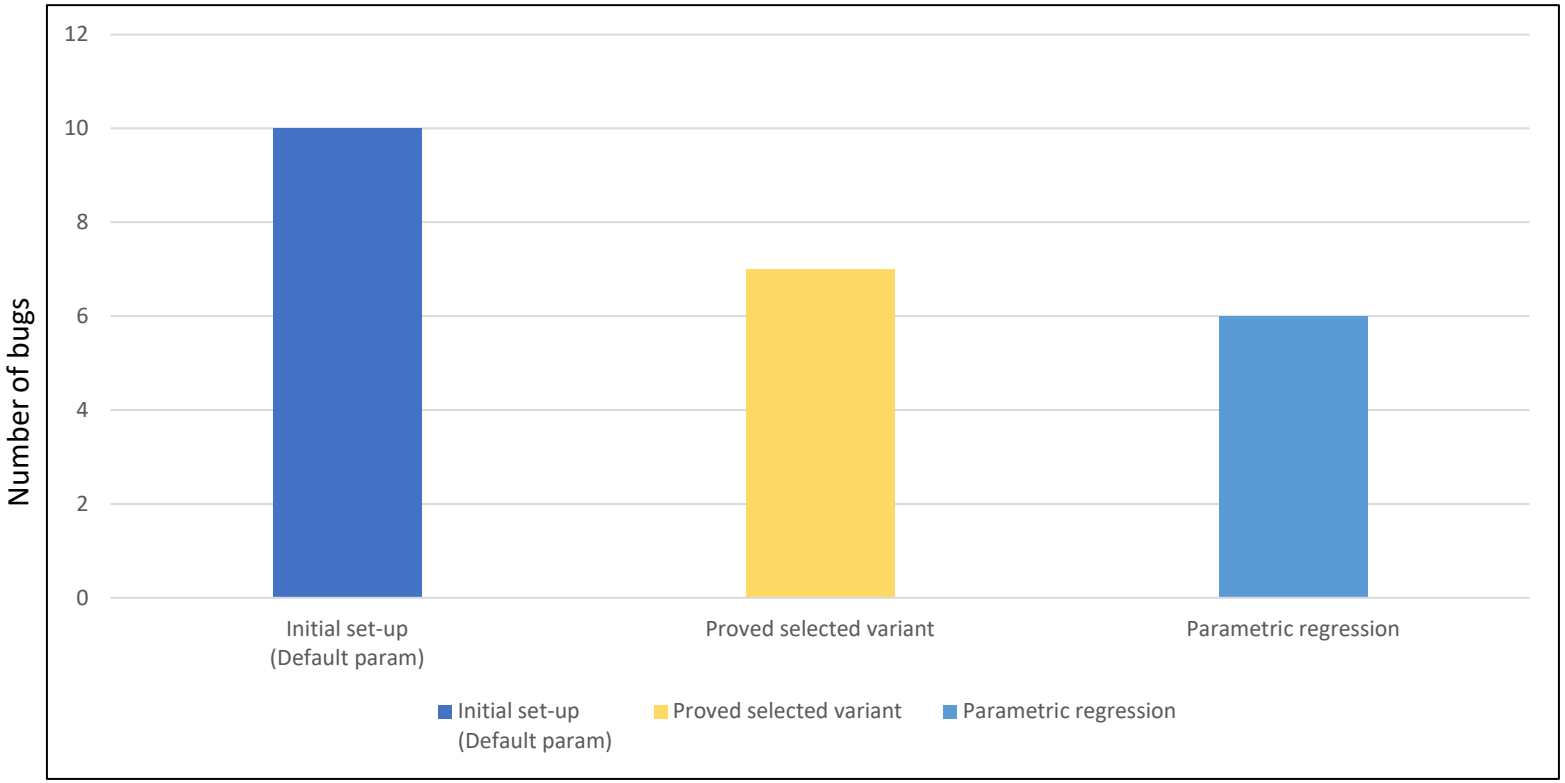
- UFV flow:
  - Flow to set-up formal TB in unified way
- Formal CFG (attributes)
  - TOP: design top
  - CLOCK, RESET
  - PARAM_FILE: specify param file
  - TOOL: JG/VCF
  - PARAM_ROW: debug specific variant
  - Mode: regress/debug
- Avoid scripting
- Periodic regression: Tracking future RTL changes



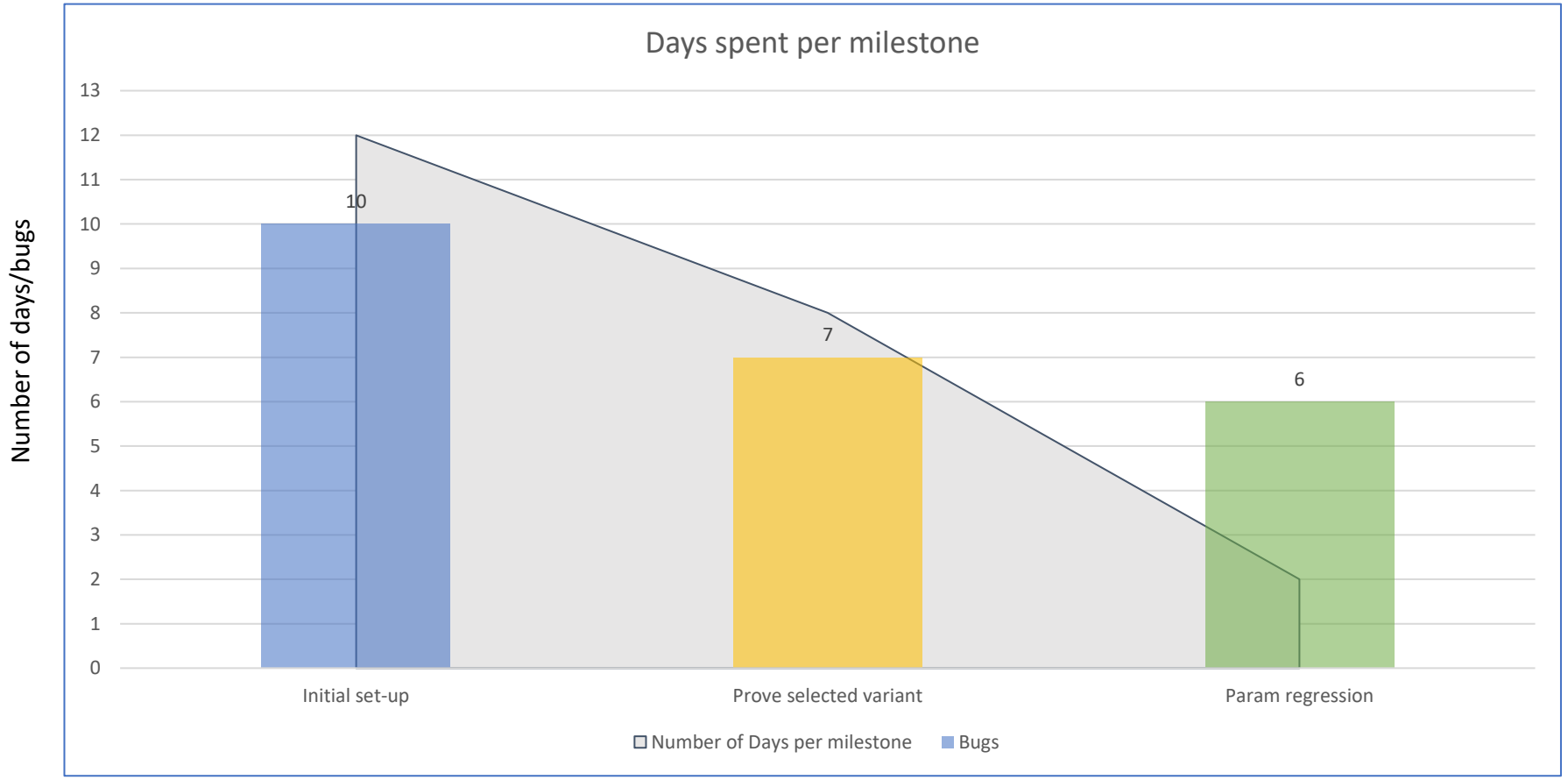
Result

Result: Bug report summary

- mem_ctrl block – 50 different design variants



Result: Verification effort



3 key takeaways

1. Exhaustive verification of parameterized IP through parametric regression
2. Simplified debugging for selected design variants
3. Automated parametric regression for tracking

Future work in progress

- Automatic generation of parameter lists
  - Python-based automation for parameter constraint
  - Constraint optimization
- Creation of configuration coverage
- Parallel regression flow

Q&A

Thank you