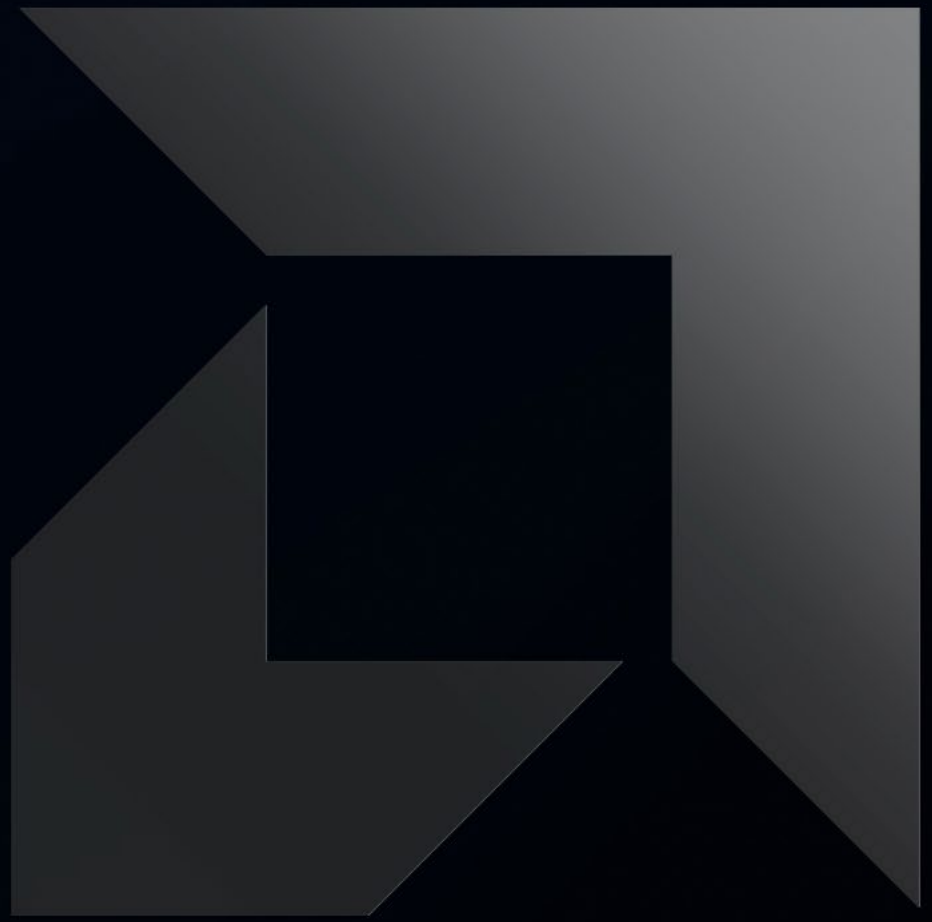


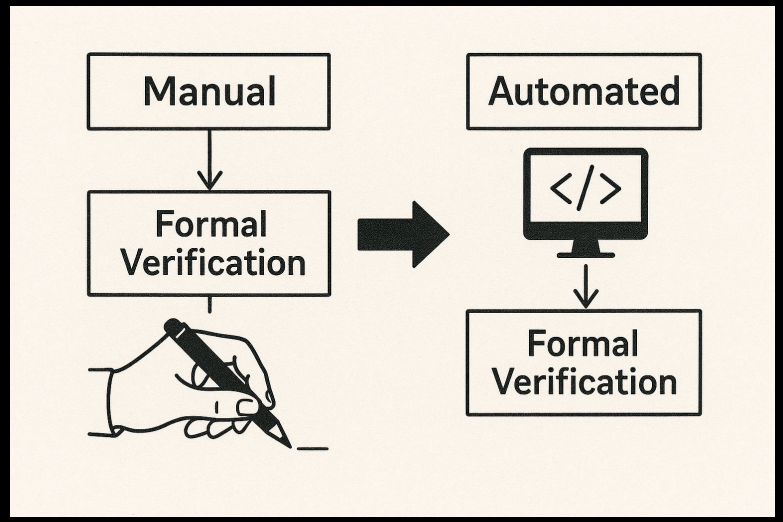
Application of AI to Accelerate Formal Verification Workflow

Security IP Team
Liya Liu
Parmar Jayesh Parmar
Namrata Teggi
Reddy Shilpa

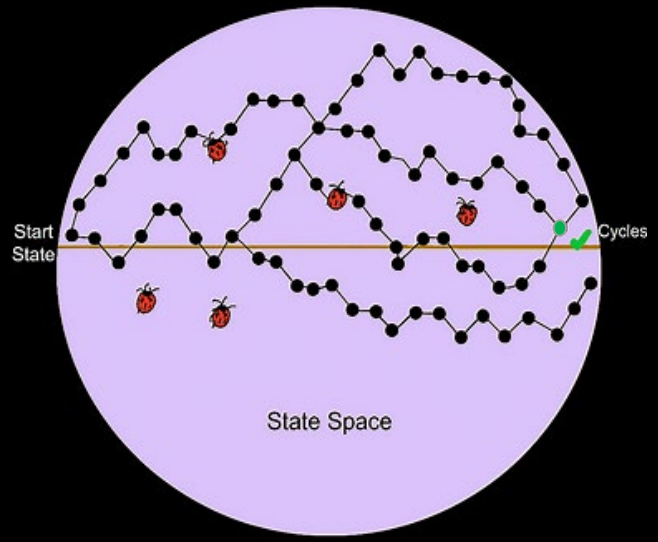
AMD
together we advance_

Outline

- Introduction
  - Formal Verification
  - Formal Verification Workflow
- Methodology
  - Automate FV Setup
  - Automate Property Implementation
  - Automate Report
- Conclusion



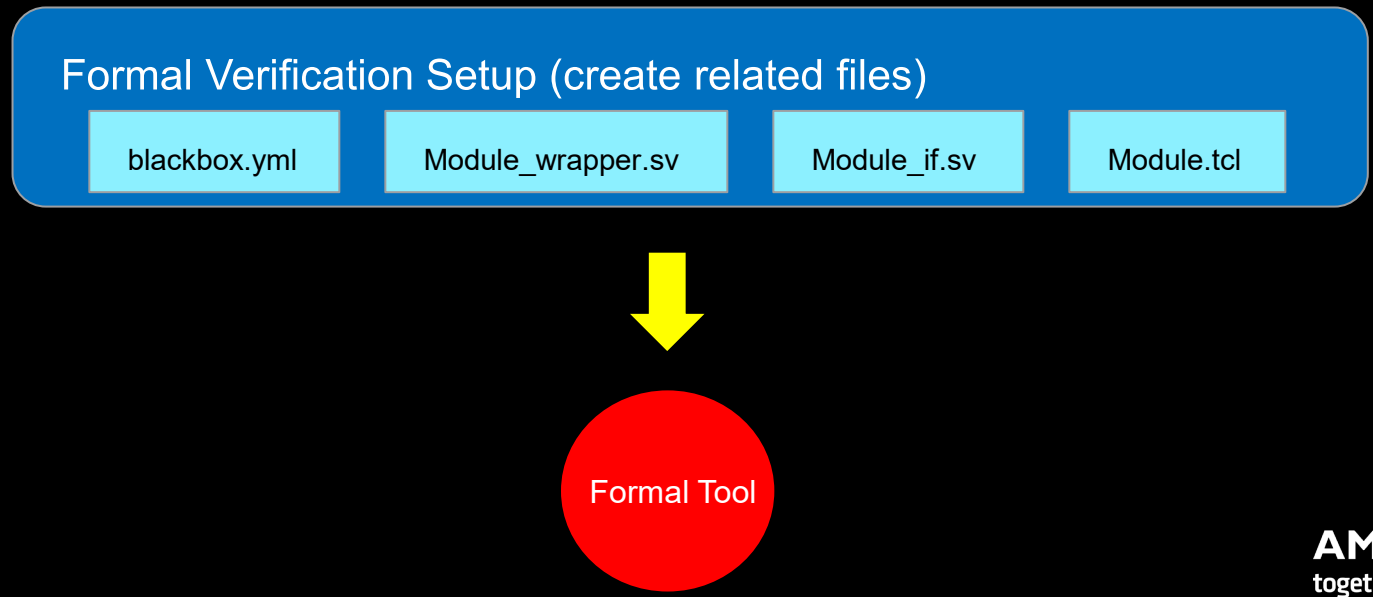
Formal Verification



- Formal verification uses mathematical techniques to exhaustively check all possible states of a design to ensure its correctness.
- For safe critic mission, exhaustive analysis is must.
- SystemVerilog properties are used to describe design behavior based on specification.
- These properties are bound to design interface.
- Formal verification tool checks the behavior described in properties match with design implementation or not.

Formal verification applications:

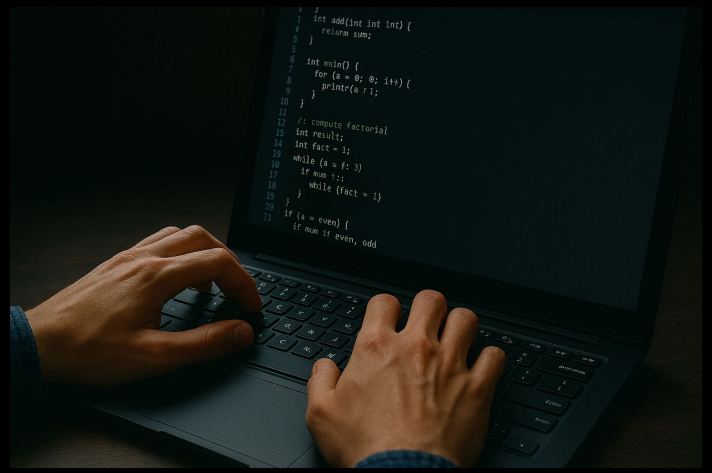
- AEP: Automatically Extract Properties
- FPV: Formal Property Verification
- DPV: Data Path Validation
- FXP: Formal X-Propagation
- FRV: Formal Register Verification
- ……

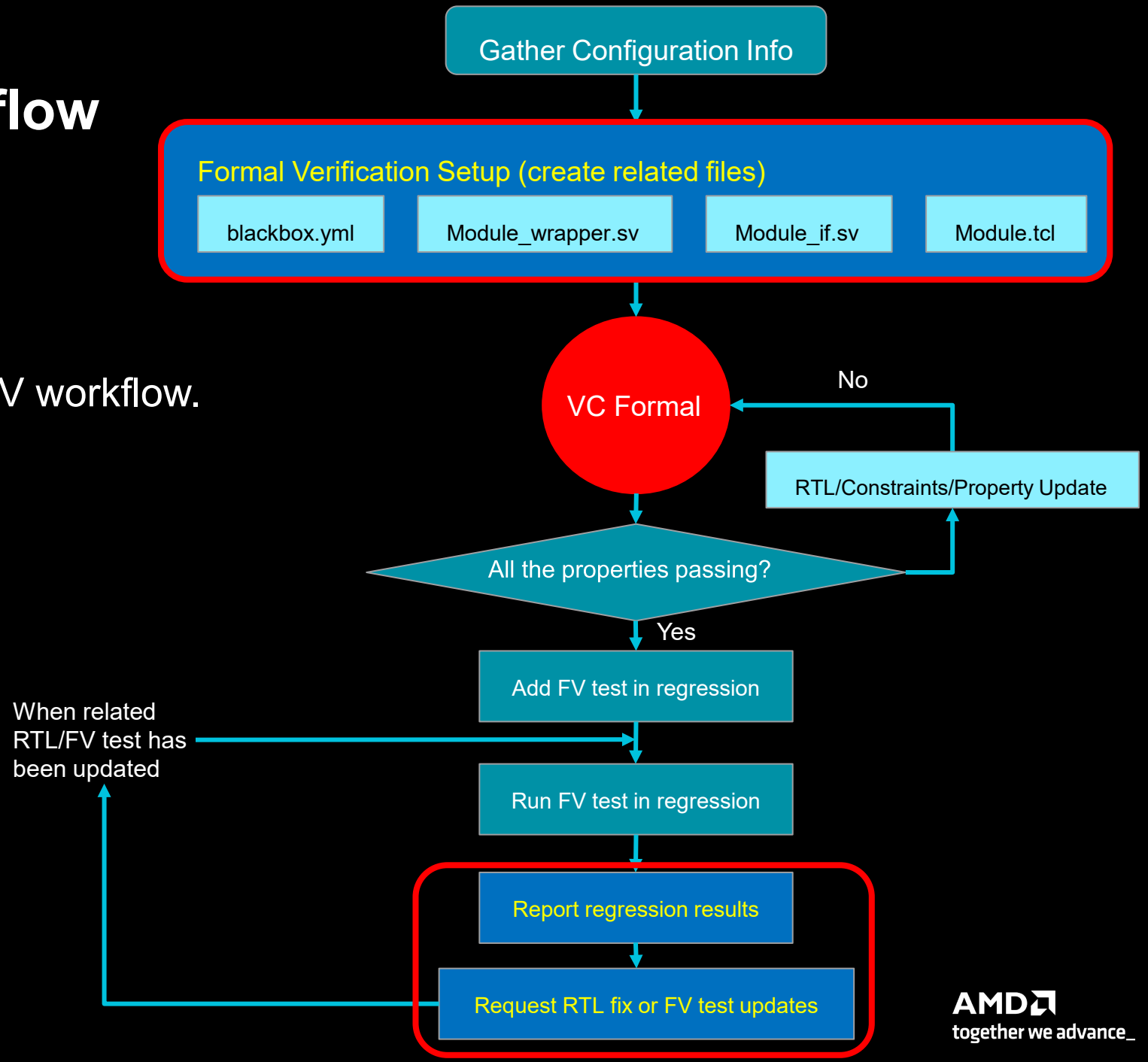


Formal Verification Workflow

Problematic

Lots of manual work involved in FV workflow.





Problems: FV Files Generation

Formal Verification Setup (create related files)

- blackbox.yml
- Module_wrapper.sv
- Module_if.sv
- Module.tcl

```
/*****************************************************************************************************************
This interface file is auto-generated from rtl_top.v from build $OUT_HOME
*****************************************************************************************************************/

interface rtl_top_if #(
    parameter DATA_WIDTH = 8,
    parameter ADDR_WIDTH = 4
)(
    input                   clk,
    input                   reset,
    input                   wr_en,
    input                   rd_en,
    input [DATA_WIDTH-1:0]  wr_data,
    input [DATA_WIDTH-1:0]  rd_data,
    input                   full,
    input                   empty,
    input [ADDR_WIDTH:0]    fifo_count
);
    default clocking props_clocking @(posedge clk); endclocking
    default disable iff(reset);

    //Paramater updated from RTL
    localparam [1:0] //FSM states
        ST_A  = 2'b00,
        ST_B  = 2'b01,
        ST_C  = 2'b10;

endinterface

bind rtl_top  rtl_top_if #(.DATA_WIDTH(PARAM_DATA_WIDTH),
                           .ADDR_WIDTH(PARAM_ADDR_WIDTH))
                           rtl_top_if_inst(.*);
```

- Parameters
- Input/output ports

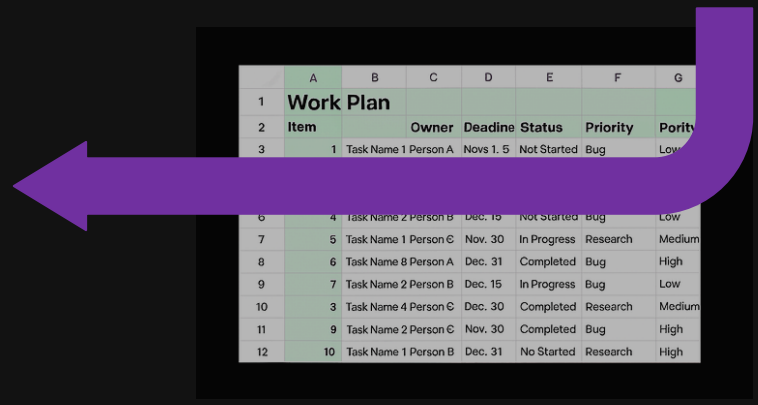
Manual work

Problem: Property Implementation

Formal Verification Setup (create related files)

- Blackbox.yml
- Module_wrapper.sv
- Module_if.sv
- Module.tcl

```
// Basic state transition
property valid_state_transition;
  @(posedge clk) disable iff (!rst_n)
  (current_state == IDLE && start_signal) |=> (current_state == ACTIVE);
endproperty

// Conditional transitions
property conditional_transition;
  @(posedge clk) disable iff (!rst_n)
  (current_state == ACTIVE && (counter >= threshold)) |=> (current_state == DONE);
endproperty

// Multi-cycle transitions
property multi_cycle_transition;
  @(posedge clk) disable iff (!rst_n)
  (current_state == PROCESSING) |-> ##[1:5] (current_state == COMPLETE);
endproperty

// Output invariants in specific states
property idle_state_outputs;
  @(posedge clk) disable iff (!rst_n)
  (current_state == IDLE) |-> (busy == 1'b0 && valid_out == 1'b0);
endproperty
```



	A	B	C	D	E	F	G
1	Work Plan						
2	Item		Owner	Deadine	Status	Priority	Porit
3	1	Task Name 1	Person A	Novs 1. 5	Not Started	Bug	Low
7	5	Task Name 1	Person C	Nov. 30	In Progress	Research	Medium
8	6	Task Name 8	Person A	Dec. 31	Completed	Bug	High
9	7	Task Name 2	Person B	Dec. 15	In Progress	Bug	Low
10	3	Task Name 4	Person C	Dec. 30	Completed	Research	Medium
11	9	Task Name 2	Person C	Nov. 30	Completed	Bug	High
12	10	Task Name 1	Person B	Dec. 31	No Started	Research	High

Problems: Regression Result Report

Emails sent out manually

Report regression results

Summary Results
Property Summary: FPV
> Assertion
- # found : 6
- # proven : 6
> Vacuity
- # found : 8
- # non_vacuous : 8
> Witness
- # found : 9
- # covered : 9
> Cover
- # found : 3
- # covered : 3
> Constraint
- # found : 4

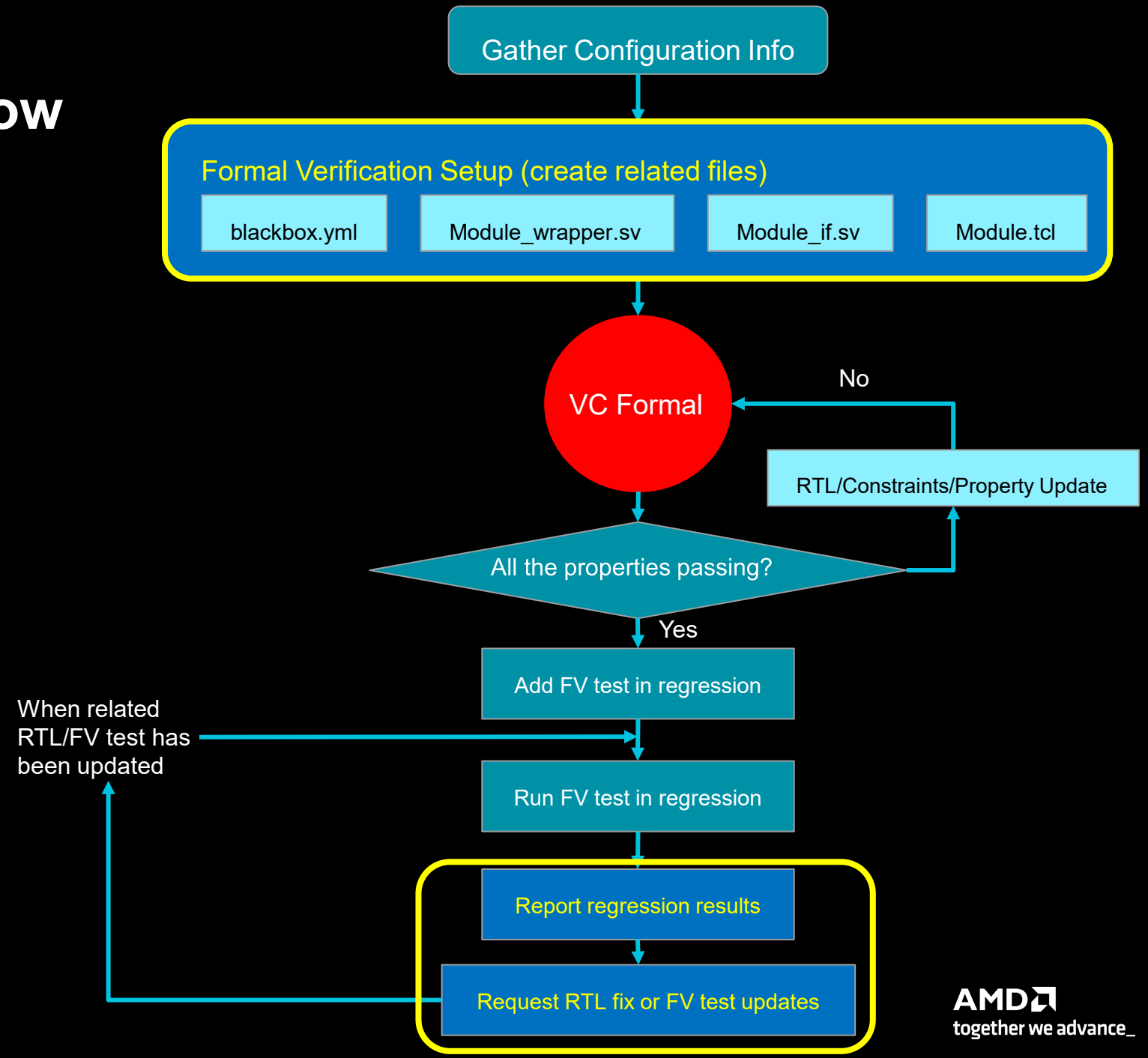
Summary Results
Property Summary: AEP
> Constraint
- # found : 8
AEP Summary: AEP
> Assertion
- # found : 37
- # proven : 37

Summary Results
Property Summary: FPV
> Assertion
- # found : 21
- # proven : 5
- # falsified : 16
> Vacuity
- # found : 24
- # non_vacuous : 24
> Witness
- # found : 25
- # covered : 25
> Cover
- # found : 3
- # covered : 3
> Constraint
- # found : 5

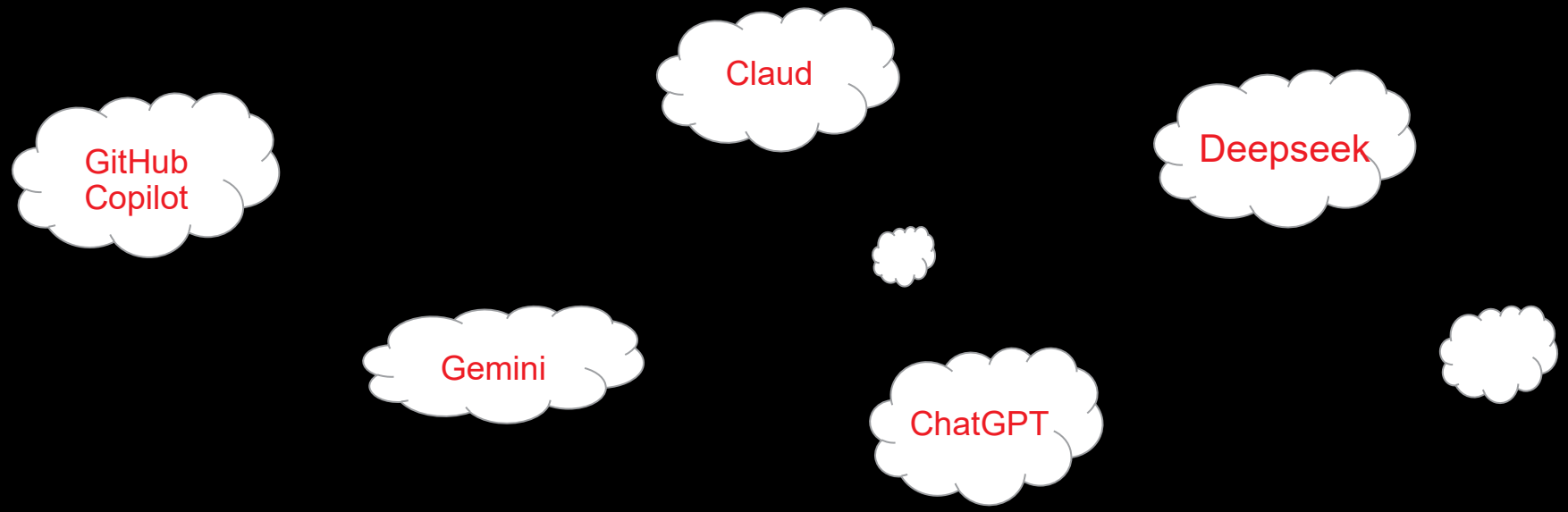
Manually check regression results

Report

Automation in FV Workflow



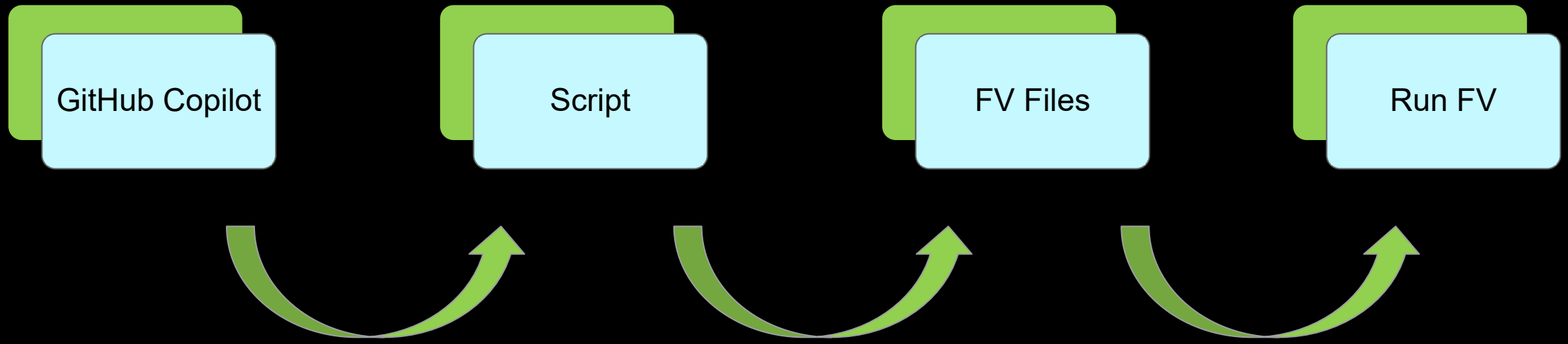
Artifical Intellegence Tools



GitHub Copilot

Code suggestions and completions inside IDEs.

Automation in workflow

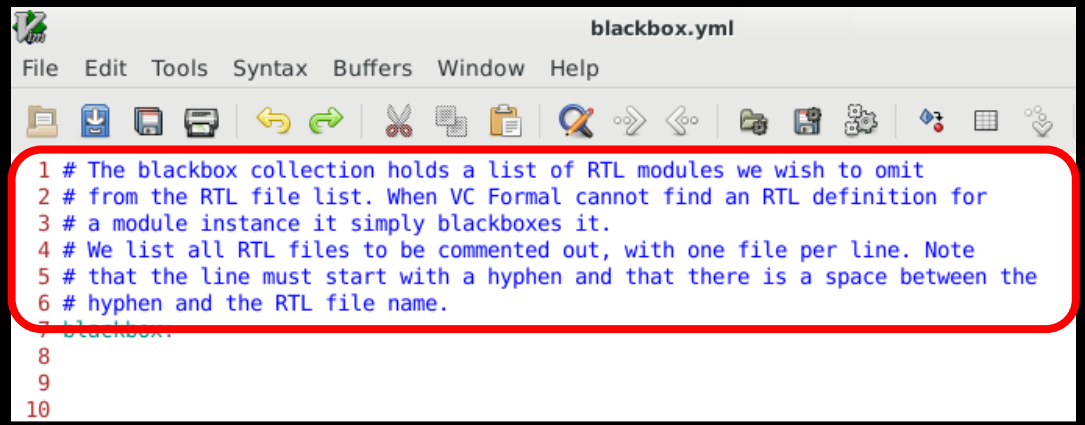


Prompt
Generate Python Code

write a python function to generate a blackbox.yml file in which involves following lines:
"# The blackbox collection holds a list of RTL modules we wish to omit
from the RTL file list. When VC Formal cannot find an RTL definition for
a module instance it simply blackboxes it.
We list all RTL files to be commented out, with one file per line. Note
that the line must start with a hyphen and that there is a space between the
hyphen and the RTL file name.
#blackbox: "

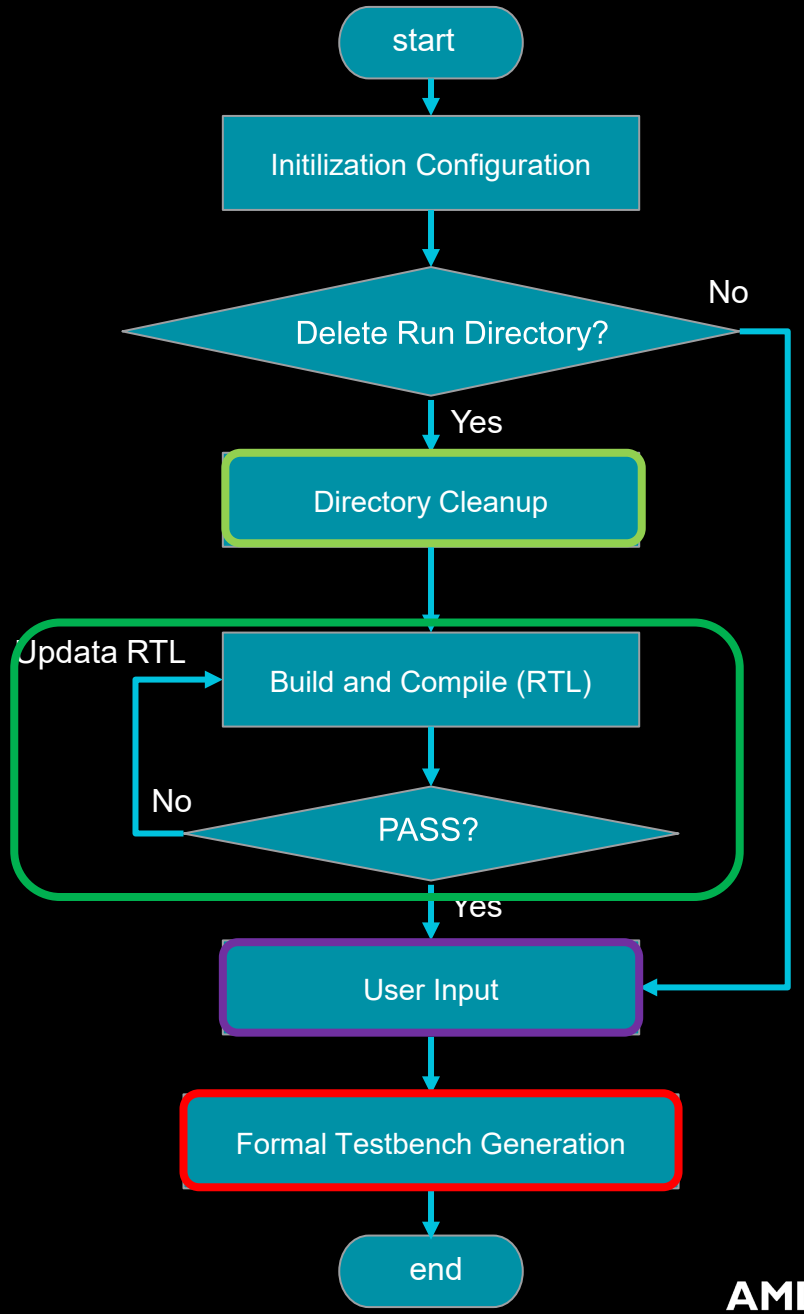
GitHub Copilot

```python
def generate_blackbox(self):
    tcl_cfg = self.get_tcl_cfg_str()
    module_dir = self.ip_inst.lower()
    self.write_str(self.tb_dirname,
                   self.tcl_fn,
                   tcl_cfg)
    # Skip writing the generic interface file if we've already generated a module-specific one
    module_specific_if_path = os.path.join(self.tb_dirname, "src/verif/vcf", self.ip_inst.lower(), f"{self.ip_inst}_if.sv")
    if not os.path.exists(module_specific_if_path):
        print(f"No module-specific interface file found at {module_specific_if_path}")
        print(f"Creating generic interface file {self.get_ifname()}.sv")
        self.write_str(self.tb_dirname,
               self.get_ifname() + ".sv",
               self.get_sv_if_str())
    else:
        print(f"Creating {module_specific_if_path}")
    if self.has_wrapper:
        self.write_str(self.tb_dirname,
                       self.get_wrapper_name() + ".sv",
                       self.get_wrapper_str())
    self.write_str(self.tb_dirname,
                   "cc_checklist.csv",
                   "ENABLE,SOURCE,DESTINATION,NAME\n")
    self.write_str(self.tb_dirname,
                   "blackbox.yml",
                   "# The blackbox collection holds a list of RTL modules we wish to omit\n"
                   "# from the RTL file list. When VC Formal cannot find an RTL definition for\n"
                   "# a module instance it simply blackboxes it.\n"
                   "# We list all RTL files to be commented out, with one file per line. Note\n"
                   "# that the line must start with a hyphen and that there is a space between the\n"
                   "# hyphen and the RTL file name.\n"
                   "#blackbox:"
                   )
    return
```



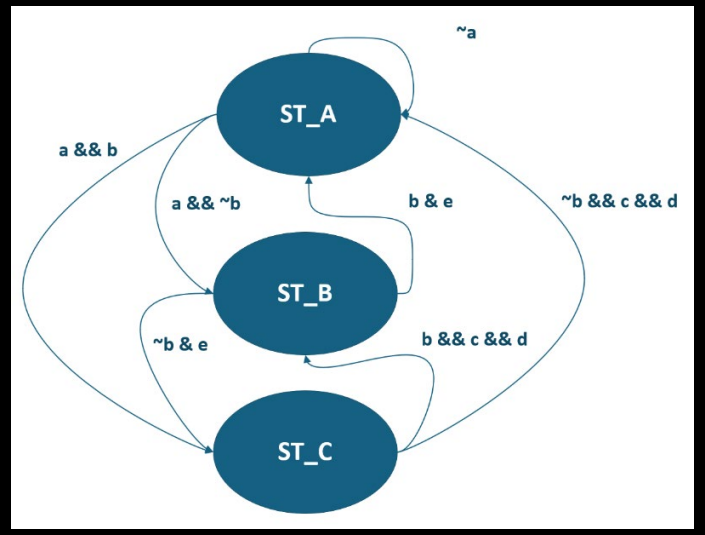
Automate FV Setup

```
python3 src/verif/vcf/scripts/vcf_formal_tb_builder.py
Do you want to delete $SIM directory? (y/n): y
Do you want to delete vcf_runs directory? (y/n): y
✅ Successfully cleaned up $SIM: /proj/version_1/sim
✅ Successfully cleaned up $SIM: /proj/version_1/vcf_run directory
Enter BUILD target: top component
================================================================================
                BUILD IS COMPLETED : FORMAL SETUP STARTS
================================================================================
Top level RTL module to verify (e.g. mpART): sync_fifo
Has wrapper (Y/N): Y
The BUILD test configuration used for the build command (e.g. from 'make quick'): modeA_sync_fifo
Directory to hold TCL config file (default is moudle_top) -- hit enter to skip:

Enter as many clocks as necessary, starting with the main clock.
The period doesn't have to match reality but the relative period ratios between clocks must be respected.
Clock name (hit enter to skip only if have at least one clock already): clk
Period in ns (number only, no units): 10
Clock name (hit enter to skip only if have at least one clock already):

Enter as many resets as necessary, starting with the main reset. We need at least one.
Reset name (hit enter to skip only if have at least one reset already): reset
Which edge are we using [low/high] ? : high
Reset name (hit enter to skip only if have at least one reset already):
Name of TCL configuration file (default is module_top.tcl)     hit enter to skip :
✅ Found module file: /proj/version_1/src/rtl/design_1_0/fifos/sync_fifo.v

Generating interface from context 'sync_fifo'
================================================================================
✅ Generated: src/verif/vcf/sync_fifo_top_if.sv
================================================================================
Creating /proj/version_1/src/verif/vcf/sync_fifo/sync_fifo.tcl
Creating /proj/version_1/src/verif/vcf/sync_fifo/sync_fifo_if.sv
Creating /proj/version_1/src/verif/vcf/sync_fifo/cc_connectivity.csv
Creating /proj/version_1/src/verif/vcf/sync_fifo/sync_fifo_wrapper.sv
Creating /proj/version_1/src/verif/vcf/sync_fifo/blackbox.yml
```



Properties Generation with Diagram

Generate the SystemVerilog properties for formally verifying the FSM shown in the attached diagram with the state defined as “reg [1:0] stateCur”.



Copilot

Assuming the states are encoded as:

Systemverilog ^ Copy

```
localparam ST_A = 2'b00;
localparam ST_B = 2'b01;
localparam ST_C = 2'b10;

property p_A_to_B;
  @(posedge clk) stateCur == ST_A && a && !b |=> stateCur == ST_B;
endproperty

property p_A_to_C;
  @(posedge clk) stateCur == ST_A && b && e |=> stateCur == ST_C;
endproperty

property p_A_to_A;
  @(posedge clk) stateCur == ST_A && !a |=> stateCur == ST_A;
endproperty

property p_B_to_C;
  @(posedge clk) stateCur == ST_B && b && c && d |=> stateCur == ST_C;
endproperty

property p_B_to_A;
  @(posedge clk) stateCur == ST_B && a && b |=> stateCur == ST_A;
endproperty

property p_B_to_B;
  @(posedge clk) stateCur == ST_B && !b && e |=> stateCur == ST_B;
endproperty

property p_C_to_A;
  @(posedge clk) stateCur == ST_C && !b && c && d |=> stateCur == ST_A;
endproperty

property p_C_to_B;
  @(posedge clk) stateCur == ST_C && !b && e |=> stateCur == ST_B;
endproperty

property p_C_to_C;
  @(posedge clk) stateCur == ST_C && b && c && d |=> stateCur == ST_C;
endproperty
```

Property Generation with Reference Code

Example

Comments

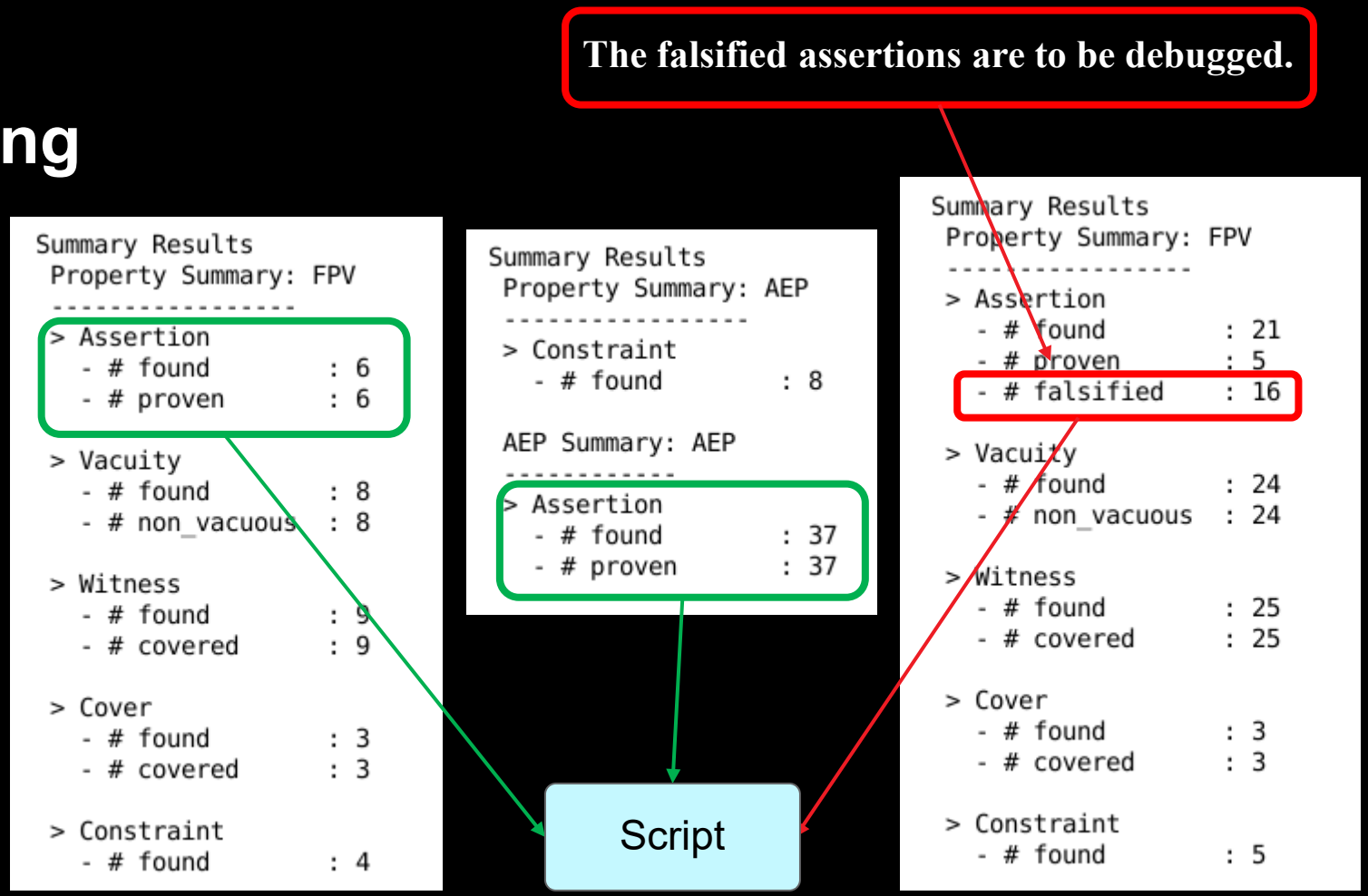
Refer to the property req_neg_id_prop in test_if.sv to generate a property req_size_len_prop to check in a read request when size is less than 2 and read 2 data, the response of this request should have RRESP being SLV_ERR.

```
// ==================    req_neg_id property     ==================
// The goal of this property is the AXI read access security check based on Read Address ID.
// The specification requires that ARID[2:0] must be 2, ARID[10:4] must be the defined INIT_ID, and
// ARID[20:11] must be the defined UNIT_ID.
// ======================================================================

property req_neg_id_prop (AxADDR, AxREADY, AxVALID, AxID, AxUSER, AxLEN, min_addr, max_addr,
                          xREADY, xVALID, xID, xRESP);
   int my_txn_id;
     ((AxREADY && AxVALID &&
       ((AxID[2:0]!= 3'h2) || (AxID[20:11] != INIT_ID)) &&
       (AxADDR inside {[min_addr:max_addr]})), my_txn_id = AxID)
         |=> !((xID == my_txn_id) && xVALID && xREADY)[*0:$] ##1 ((xID == my_txn_id) && xVALID && xREADY) && (xRESP == SLVERR);
endproperty // req_neg_id_prop

 // ======== Place your checker instantiations here ======
req_neg_id : assert property (req_neg_id_prop (
                                    .AxADDR  (XBAR__s_axi_araddr  ),
                                    .AxREADY (s_XBAR_axi_arready ),
                                    .AxVALID (XBAR__s_axi_arvalid ),
                                    .AxID    (XBAR__s_axi_arid),
                                    .AxUSER  (XBAR__s_axi_aruser),
                                    .AxLEN   (XBAR__s_axi_arlen   ),
                                    .min_addr(UNIT_ID_START),
                                    .max_addr(UNIT_ID_END),
                                    .xREADY  (XBAR__s_axi_rready  ),
                                    .xVALID  (s_XBAR_axi_rvalid  ),
                                    .xID     (s_XBAR_axi_rid     ),
                                    .xRESP   (s_XBAR_axi_rresp   )
                                                                  ));
```

Copilot

```
// Property: req_size_len
// Checks for size < 2 and reading 2 keys (AxLEN == 15), response should be SLVERR
property req_size_len (AxADDR, AxREADY, AxVALID, AxID, AxUSER, AxLEN, min_addr, max_addr,
                       xREADY, xVALID, xID, xRESP);
   int my_txn_id;
     ((AxREADY && AxVALID && (AxUSER < 2) && (AxLEN == 15) && (AxADDR inside {[min_addr:max_addr]})), my_txn_id = AxID)
         |=> !((xID == my_txn_id) && xVALID && xREADY)[*0:$] ##1 ((xID == my_txn_id) && xVALID && xREADY) && (xRESP == SLVERR);
endproperty // req_size_len

req_size_len_check : assert property (req_size_len (
                                    .AxADDR  (XBAR_s_axi_araddr  ),
                                    .AxREADY (s_XBAR_axi_arready ),
                                    .AxVALID (XBAR_axi_arvalid ),
                                    .AxID    (XBAR_axi_arid),
                                    .AxUSER  (XBAR_axi_aruser),
                                    .AxLEN   (XBAR_s_axi_arlen   ),
                                    .min_addr(REG_ADDR_START),
                                    .max_addr(REG_ADDR_END),
                                    .xREADY  (s_XBAR_axi_rready  ),
                                    .xVALID  (s_XBAR_axi_rvalid  ),
                                    .xID     (s_XBAR_axi_rid     ),
                                    .xRESP   (s_XBAR_axi_rresp   )
                                                                  ));
```

Automate Reporting



Extract Info from regression results

Report

Emails sent out automatically

Results

Applied the method in formal verification of 5 modules

RTL Block	Number of signals on i/f	Number of properties	Approach	FV setup	Property Development	Estimated Regression Status Report
Module A	15	38	Before	2 hours	2 hours	10 minutes
			After	5 minutes	5 minutes	2 minutes
Module B	27	11	Before	2.5 hours	3.5 hours	10 minutes
			After	5 minutes	1.5 hours	2 minutes
Module C	40	22	Before	3 hours	3.5 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes
Module D	29	13	Before	2 hours	4 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes
Module E	298	22	Before	4 hours	9 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes

Before: Manual work
After: Automated

FV setup

1. Create folders/files.
2. Edit files including the interface signals added/edited.

Property Development

Type SystemVerilog code referring to the property plan based on RTL specification.

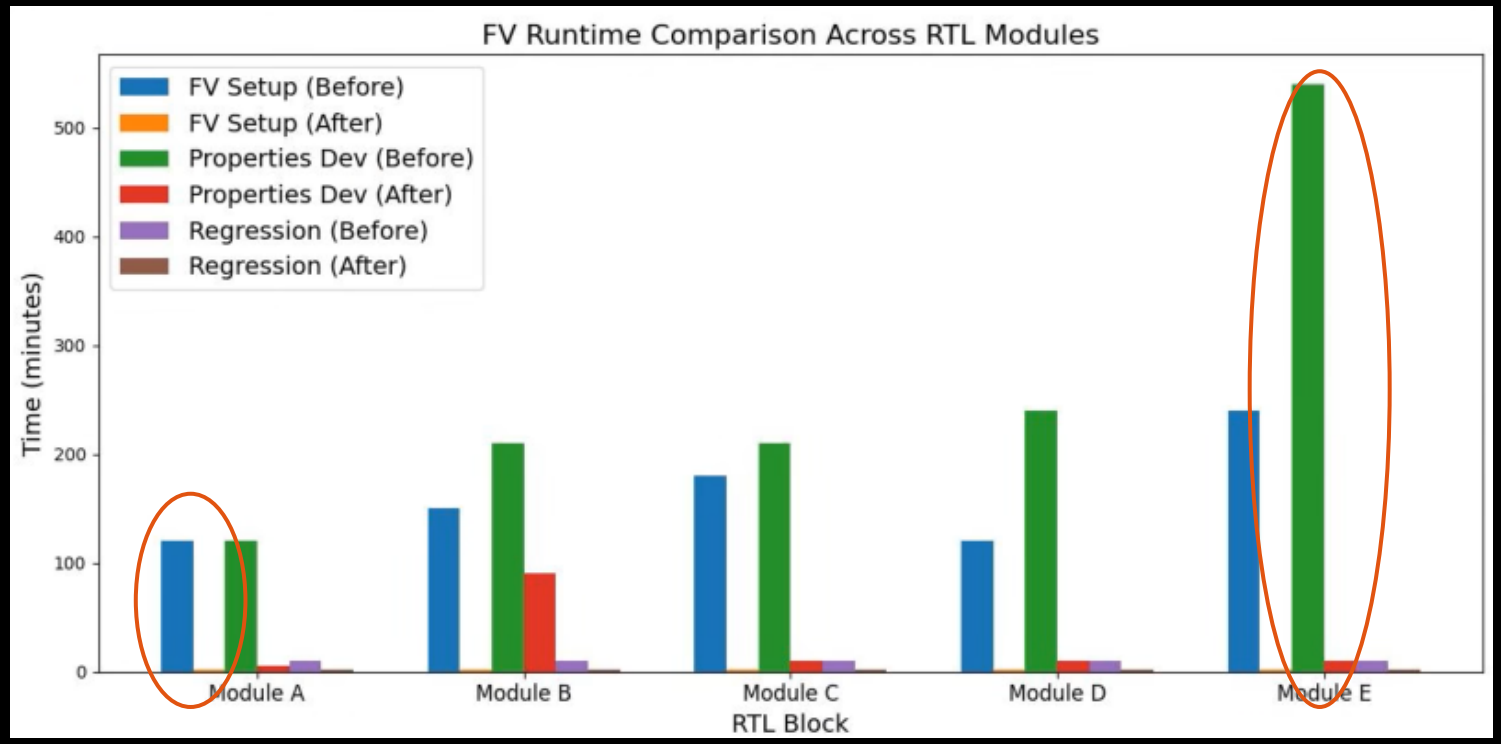
Regression Report

1. Open vcf.log files and search for summary.
2. Copy/paste property proof status into an excel file.
3. Write an email to send to all the owners/managers.

Results

Applied the method in formal verification of 5 modules

RTL Block	Number of signals on i/f	Number of properties	Approach	FV setup	Property Development	Estimated Regression Status Report
Module A	15	38	Before	2 hours	2 hours	10 minutes
			After	5 minutes	5 minutes	2 minutes
Module B	27	11	Before	2.5 hours	3.5 hours	10 minutes
			After	5 minutes	1.5 hours	2 minutes
Module C	40	22	Before	3 hours	3.5 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes
Module D	29	13	Before	2 hours	4 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes
Module E	298	22	Before	4 hours	9 hours	10 minutes
			After	5 minutes	10 minutes	2 minutes

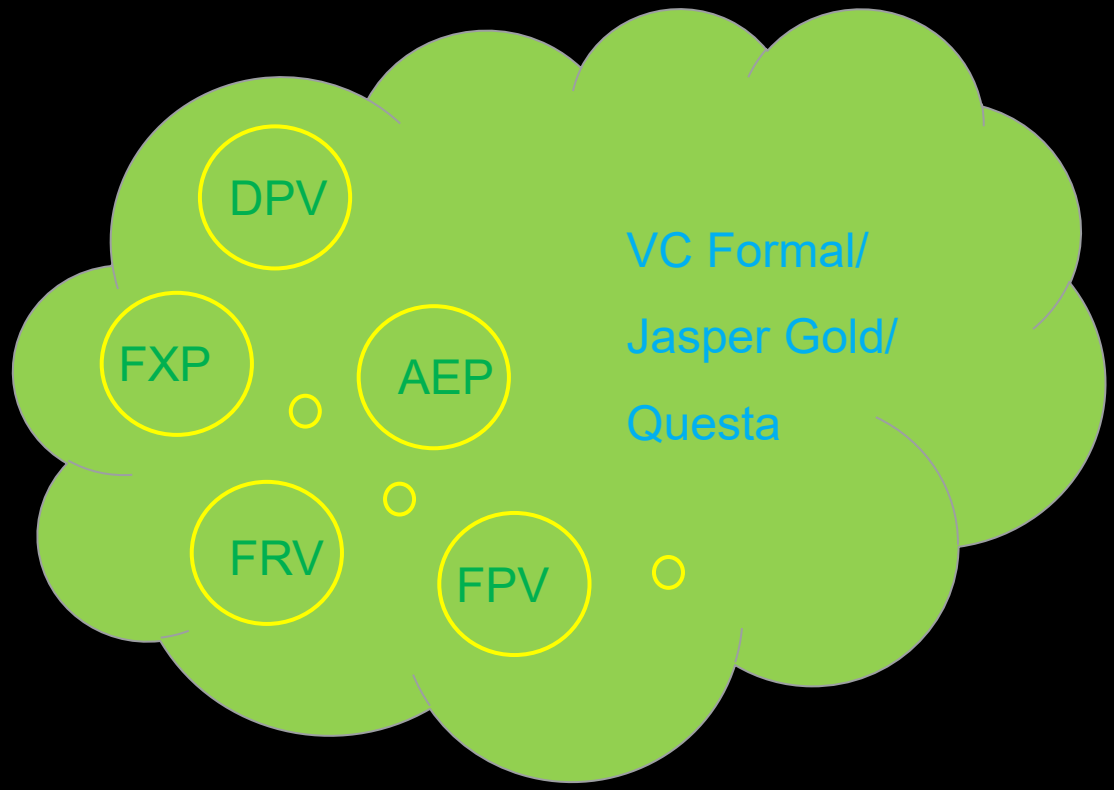


Conclusion

- ✓ Automated FV Files Generation
- ✓ Automated Properties Implementation
- ✓ Automated Regression Report

extend

Improved Work Efficiency!



Thank You


AMD