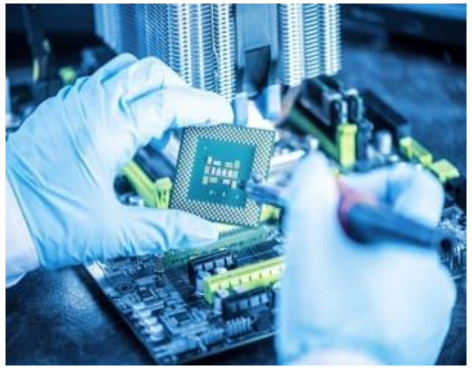
TESSOLVE

A HERO ELECTRONIX VENTURE

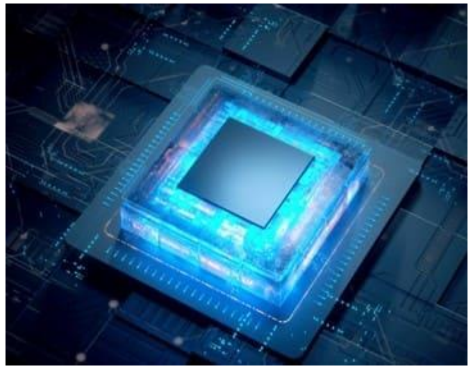
Automation/AI Efficiencies in Verification

Marmik Soni and Mike Bartley, Tessolve

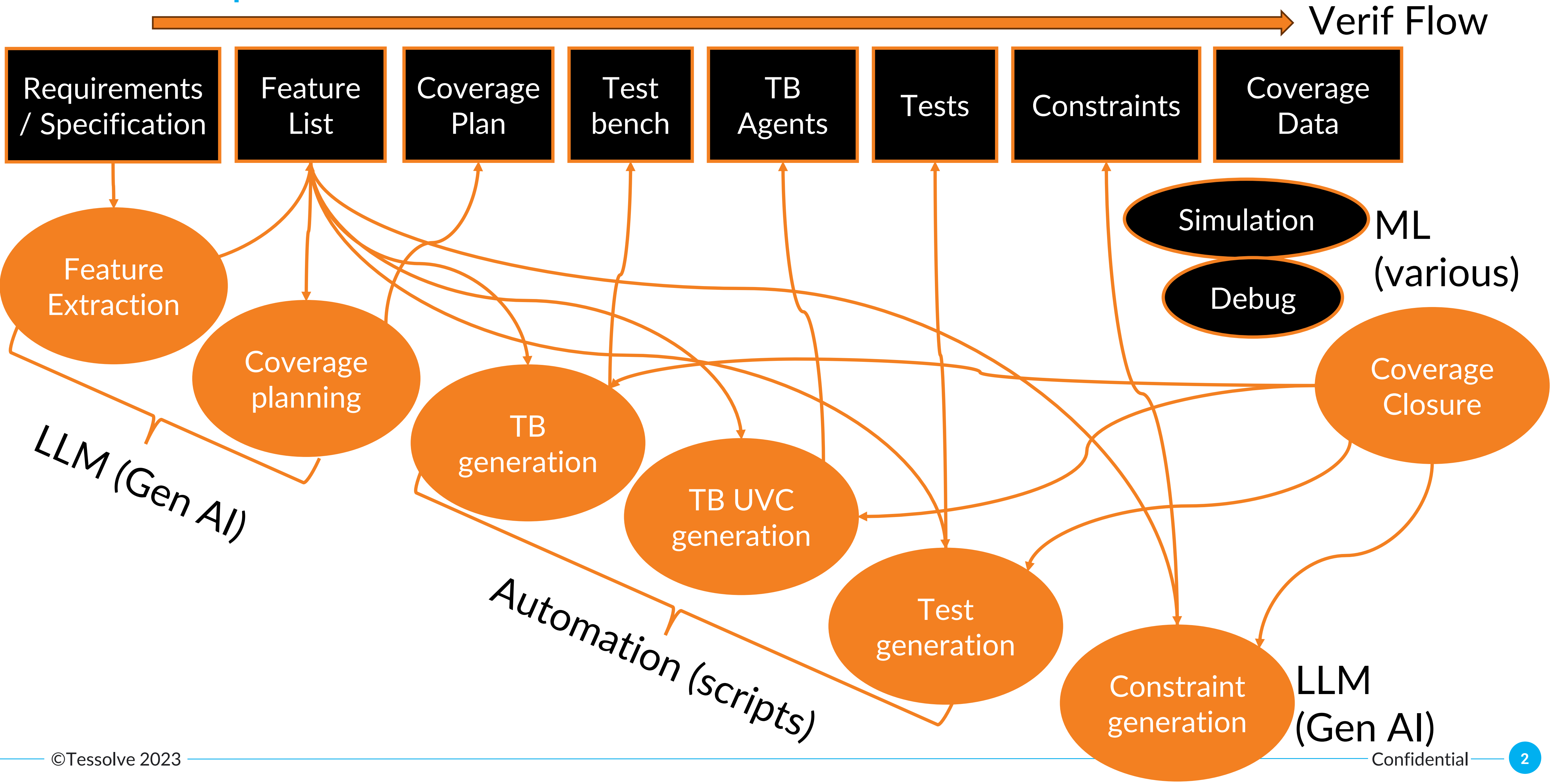
Chip Design


Test Engineering

Hardware Design


Embedded Systems

AI/ML Experiments in IP Verification



Ex: Xilinx Synchronous FIFO V3.0

Using LLM to Specification for Feature Extractio

Input = PDF Specification

output = table of I/O

output = list of features

Signal Name	I/O	Description
DIN[N:0]	Input	Data Input
WR_EN	Input	Write Enable (request)
WR_ACK	Output	Write Acknowledge: Handshake signal indicates successful write operation
WR_ERR	Output	Write Error: Handshake signal indicates unsuccessful write operation
RD_EN	Input	Read Enable (request)
RD_ACK	Output	Read Acknowledge: Handshake signal indicates successful read operation
RD_ERR	Output	Read Error: Handshake signal indicates unsuccessful read operation
CLK	Input	Clock for read and write operations
SINIT	Input	Synchronous initialization of all FIFO functions, flags, and pointers
DOUT[N:0]	Output	Data Output
FULL	Output	Full Flag: Indicates that no additional writes can be performed
EMPTY	Output	Empty Flag: Indicates that no data is present
DATA_COUNT[C:0]	Output	Data Count: Vector

Features to be Verified:

Data Read and Write Operations:

Verify that data presented at the data-input port (DIN) is correctly written into the FIFO on a rising clock edge when the write-enable input (WR_EN) is high.

Verify that data can be read out of the FIFO via the data-output port (DOUT) in the order it was written by asserting read-enable (RD_EN) prior to a rising clock edge.

Status Flags:

Verify the behavior of the memory-full status output (FULL) to indicate when no more empty locations are available in the internal memory.

Verify the behavior of the memory-empty status output (EMPTY) to indicate when no more data resides in the internal memory.

Invalid Request Handling:

Verify that the FIFO status is not corrupted by invalid read or write requests.

……………………………

FIFO – Gen AI TB

TESSOLVE
A HERO ELECTRONIX VENTURE

Input = PDF Specification

Output = Simple directed TB

Output = Coverage

```
module tb_fifo;
// Declare signals and variables for the testbench
 logic clock; logic reset;
 logic wr_en; logic rd_en;
 logic [DATA_WIDTH-1:0] data_in;
 logic [DATA_WIDTH-1:0] data_out;
 // Instantiate the FIFO module
 fifo #(DATA_WIDTH, FIFO_DEPTH) dut ( .clock(clock), .....)
 always #5 clock = ~clock; // Initialize signals and variables
  initial begin clock = 0;
  reset = 1;
  wr_en = 0;
  rd_en = 0;
  data_in = 0; // Reset the FIFO
  #10 reset = 0; // Write data into the FIFO
  #20; wr_en = 1;
  data_in = 8'hFF; // Example data value
  #10; wr_en = 0; // Read data from the FIFO
  #30; rd_en = 1;
  #10; rd_en = 0;
  #10; $finish;
 end // Display the output data
 always @(posedge clock)
 begin
  $display("Data Out: %h", data_out);
end
endmodule
```

Assertion for FIFO empty and full signals

```
assert property (fifo_inst.EMPTY |-> !fifo_inst.FULL);
```

```
covergroup fifo_coverage @(posedge clk);
 // Declare coverpoints for different events
 coverpoint data_count {
  bins empty = {0};
  bins non_empty = [1:$];
 }
 coverpoint write_enable;
 coverpoint read_enable;
 coverpoint full;
 coverpoint almost_full;
 coverpoint almost_empty;
 // Declare cross coverage
 cross data_count, write_enable;
 cross data_count, read_enable;
 cross full, write_enable;
 cross almost_full, write_enable;
 cross almost_empty, read_enable;
endgroup
```

CPU Register Extraction -Gen AI

TESSOLVE
A HERO ELECTRONIX VENTURE

Input = PDF Specification

Output = Register Details

Sr. No.	Register Name	Address	Reg Size	Access	Importance	Functionality
1	[redacted]X	0x3XX	32 bits	R/W	High	Stores the 32-bit secret key value used for secure operations.
2	[redacted]	0x3XX	32 bits	R/W	High	Stores the 32-bit decryption key value used for secure operations.
3	[redacted] Register	0x3XX	32 bits	R/W	Medium	Contains interrupt and debug-related information for host communication.
4	[redacted]	0x3XX	1 bit	R	Medium	Indicates the presence of an interrupt condition.
5	I[redacted]	0x7XX	1 bit	R/W	Medium	Enables or disables interrupts.
6	[redacted]	0x8XX	1 bit	R/W	Medium	Selects the key size (128-bit or 256-bit).
7	D[redacted]anted	0x7XX	1 bit	R	Medium	Indicates whether debug access is granted or not.
8	H[redacted]	0x3XX	1 bit	R/W	Medium	Clears the host interrupt.
9	[redacted]	0x5XX	32 bits	R	Low	Connected to the chip top for external status information.
10	[redacted]nal Status Register 1	0x5XX	32 bits	R	Low	Connected to the chip top for external status information.
11	E[redacted]s Register 2	0x5XX	32 bits	R	Low	Connected to the chip top for external status information.
12	Lifecycle Register	0x5XX	32 bits	R	Low	Contains the lifecycle word read from OTP memory.
13	[redacted]	0x6XX	32 bits	R/W	Low	Controls the reset and enable signals for the subsystem.
14	[redacted]/Status Register	0x6XX	32 bits	R/W	Low	Control and status register with undefined usage.

UVM Automation – list of files generated + sample code

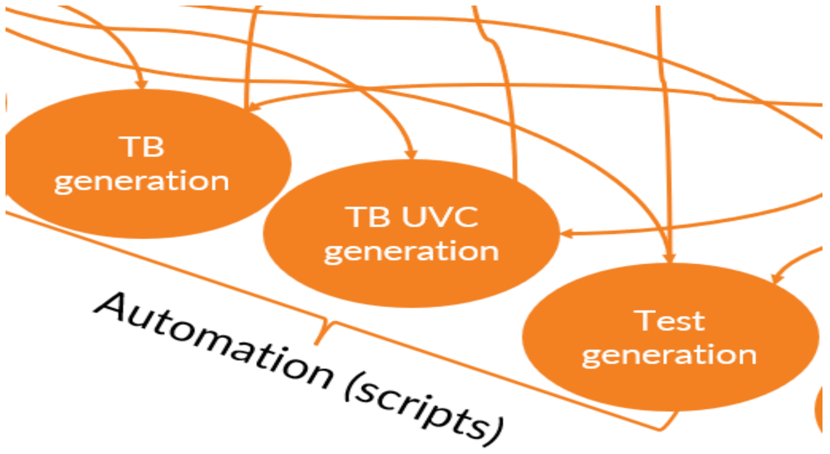
Input = DUT Details including:

- DUT interface
  - Control
  - Data
- Interface protocols

Output

Py Generated Files
1. fifo_agent_c
2. fifo_coverage
3. fifo_driver_c
4. fifo_env
5. fifo_monitor
6. fifo_pkg
7. fifo_scoreboard
8. fifo_sequence_base_c
9. fifo_sequence_item_c
10. fifo_sequence_read_c
11. fifo_sequencer_c
12. fifo_test
13. Interf
14. testbench

80% Effort Reduction



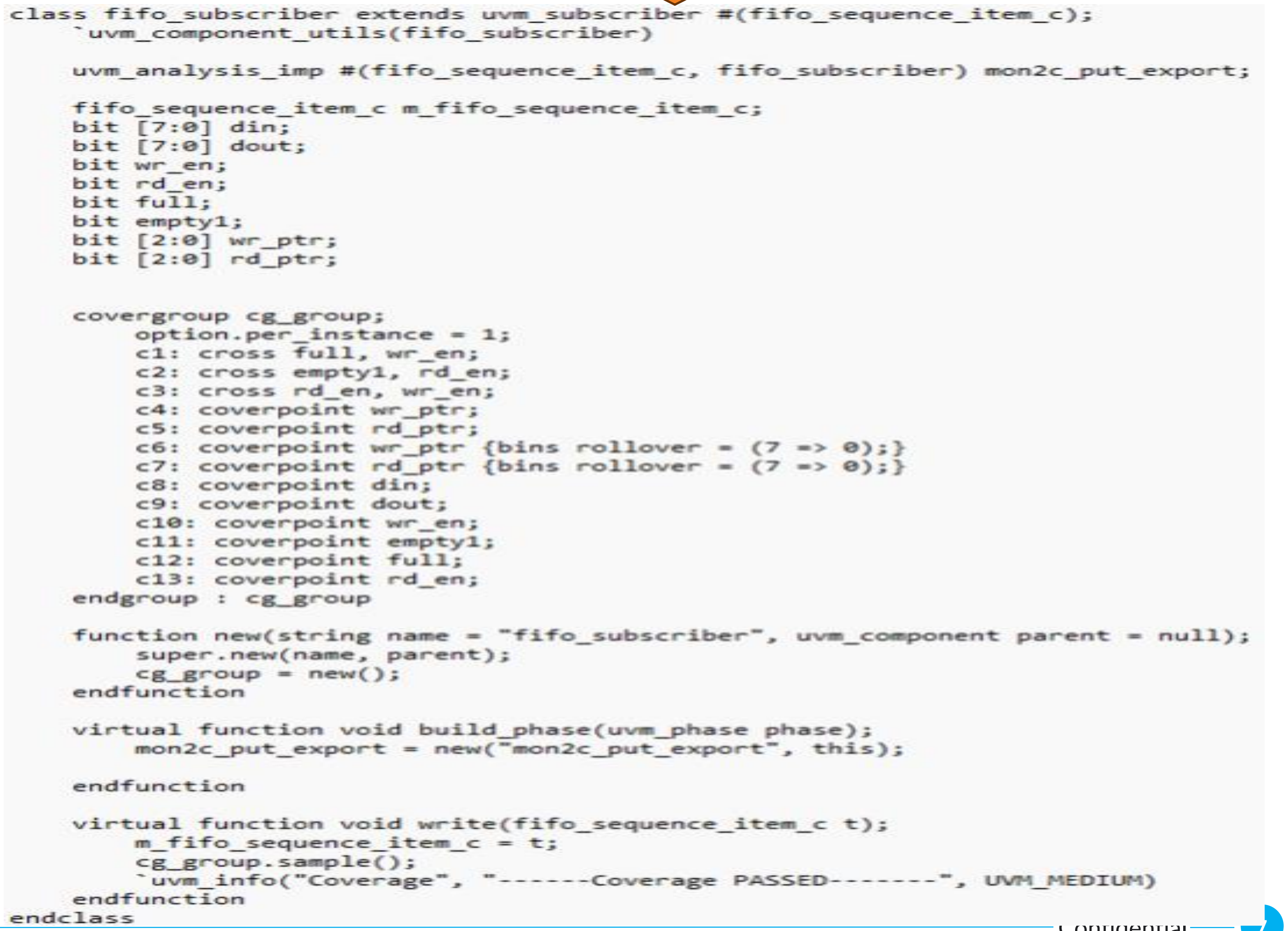
UVM Automation- sample code

TESSOLVE
A HERO ELECTRONIX VENTURE

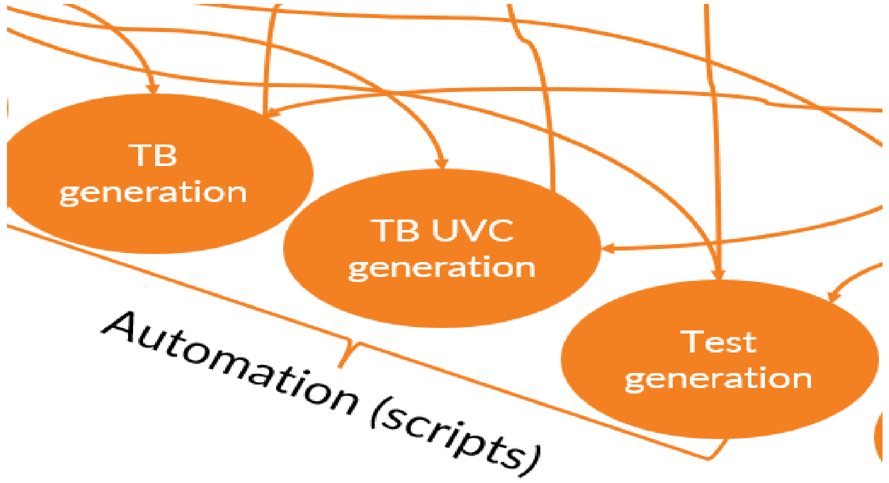
Input = DUT Details including:

- DUT interface
  - Control
  - Data
- Interface protocols

Output = Coverage

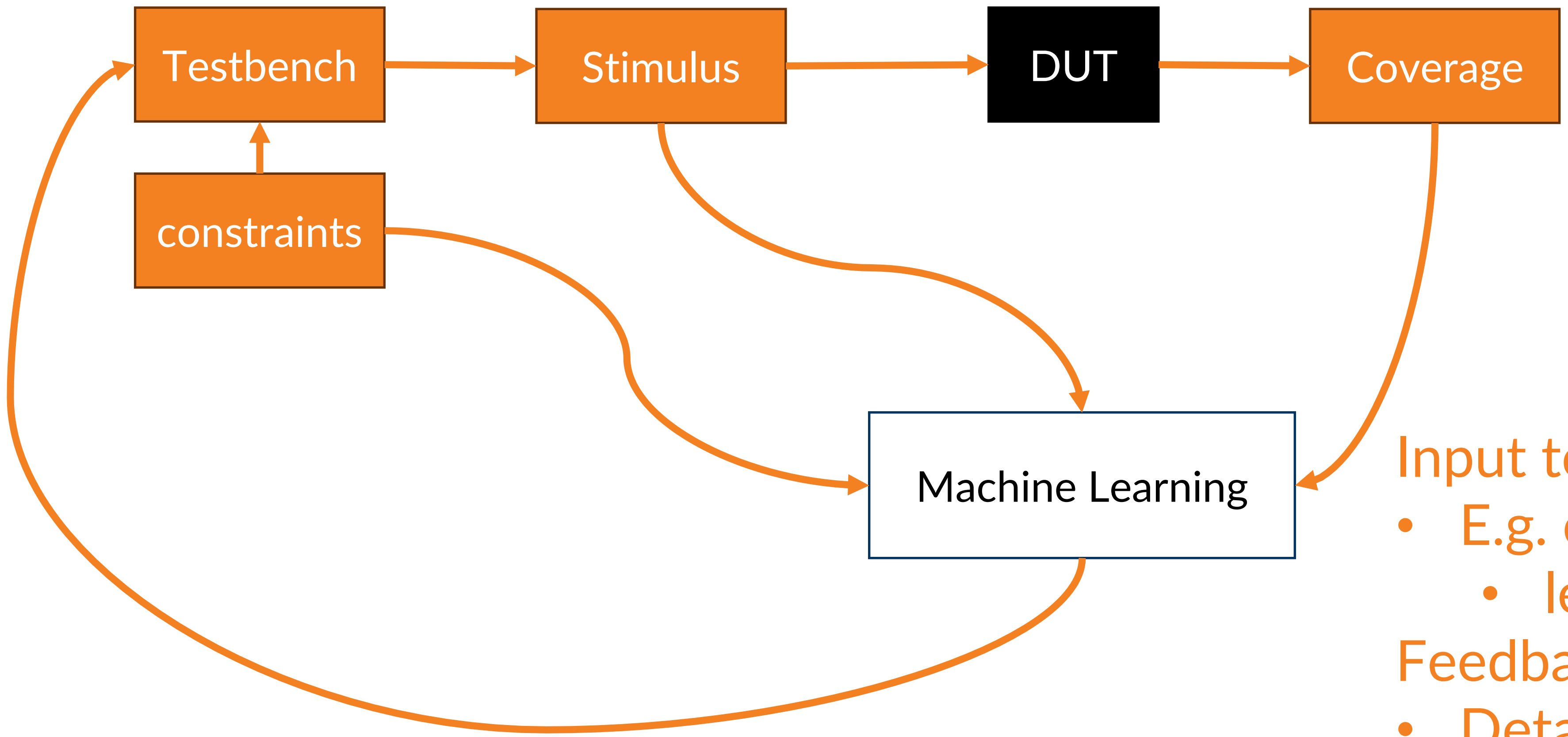


```
class fifo_subscriber extends uvm_subscriber #(fifo_sequence_item_c);
    `uvm_component_utils(fifo_subscriber)

    uvm_analysis_imp #(fifo_sequence_item_c, fifo_subscriber) mon2c_put_export;

    fifo_sequence_item_c m_fifo_sequence_item_c;
    bit [7:0] din;
    bit [7:0] dout;
    bit wr_en;
    bit rd_en;
    bit full;
    bit empty1;
    bit [2:0] wr_ptr;
    bit [2:0] rd_ptr;


    covergroup cg_group;
        option.per_instance = 1;
        c1: cross full, wr_en;
        c2: cross empty1, rd_en;
        c3: cross rd_en, wr_en;
        c4: coverpoint wr_ptr;
        c5: coverpoint rd_ptr;
        c6: coverpoint wr_ptr {bins rollover = (7 => 0);}
        c7: coverpoint rd_ptr {bins rollover = (7 => 0);}
        c8: coverpoint din;
        c9: coverpoint dout;
        c10: coverpoint wr_en;
        c11: coverpoint empty1;
        c12: coverpoint full;
        c13: coverpoint rd_en;
    endgroup : cg_group

    function new(string name = "fifo_subscriber", uvm_component parent = null);
        super.new(name, parent);
        cg_group = new();
    endfunction

    virtual function void build_phase(uvm_phase phase);
        mon2c_put_export = new("mon2c_put_export", this);

    endfunction

    virtual function void write(fifo_sequence_item_c t);
        m_fifo_sequence_item_c = t;
        cg_group.sample();
        `uvm_info("Coverage", "------Coverage PASSED-------", UVM_MEDIUM)
    endfunction
endclass
```



Applying ML to coverage closure

TESSOLVE
A HERO ELECTRONIX VENTURE



Input to ML:
- E.g. constraints vs. stimulus
  - level of abstraction

Feedback from simulation
- Detailed coverage
  - Code and functional

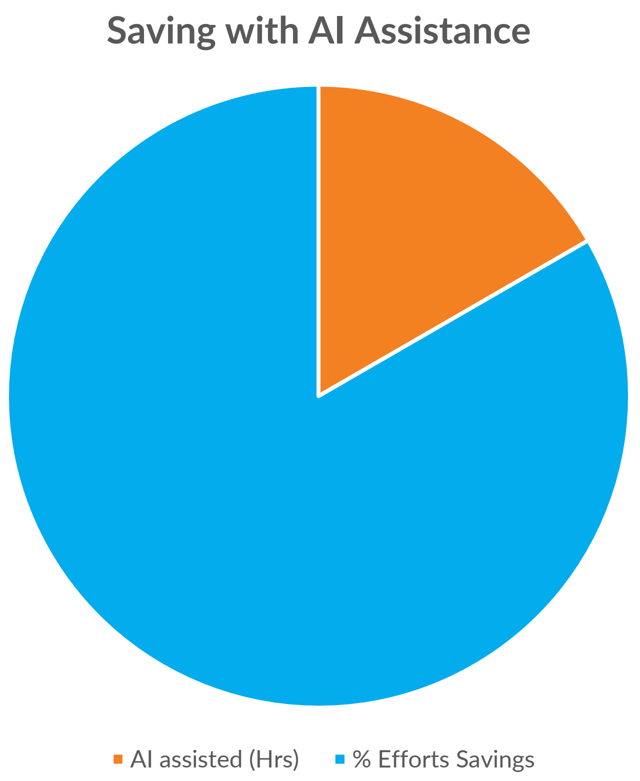
ML algorithm
- Starting with GA

Applying Gen AI on Real Project

Conventional (Hrs)	AI assisted (Hrs)	% Efforts Savings
102	17	85

TESSOLVE
A HERO ELECTRONIX VENTURE

DV Flow	Conventional Efforts	AI Assisted Efforts	Saving		
			Engineering Efforts Saved (Hrs)	% Efforts Savings	Speedup (times)
Specification Analysis	1 Week, 1 Engineer	2 Hrs, 1 Engineer	38	95	20
Feature Extraction & Test Plan Gen.	2 Days, 1 Engineer	1 Days, 1 Engineer	8	50	2
Constraint Generation	1 Day, 1 Engineer	2 Hours, 1 Engineer	6	75	4
Coverage Generation	6 Hours, 1 Engineer	1 Hour, 1 Engineer	5	83	6
Assertion Generation	1 Day, 1 Engineer	2 Hours, 1 Engineer	6	75	4
UVM Template Generation	3 Days, 1 Engineer	2 Hours, 1 Engineer	22	92	12



Practicalities of applying AI/ML in verification

- Use of LLM
    - Knowledge and expertise needed for
        - The DUT
        - On using LLM
- Use of ML in coverage closure
    - Knowledge and expertise needed for
        - The DUT
        - On using ML
- Data Set size
    - Large data sets (of clean, unbiased data) are needed
- Security
    - LLMs use the cloud
    - Documents are not saved – but data is sent and later deleted

Conclusions

- AI/ML will change the way we do verification
    - It is usable now
    - The results are already impressive and improving
- Chatbots are a good way to perform some basic tasks
    - Spec analysis, Feature extraction, test bench infrastructure
    - The quality is reasonable
- Applying ML to more complex tasks (e.g. coverage closure)
    - It is being applied on a point-wise problem basis
    - Applying it more widely with generalised model is more difficult
        - But is coming
- EDA are adding AI/ML under the hood
    - Useful but not differentiating
- Applying AI to debug can be done now
    - Using EDA or build-your-own