Ben Cohen
abv-sva.org

VMMing a SystemVerilog Testbench by Example

Ben Cohen
Srinivasan Venkataramanan
Ajeetha Kumari

http://www.abv-sva.org/
ben@abv-sva.org

Objectives (This presentation)

Ben Cohen
abv-sva.org

- **Goals**
  - Experiences in creating a VMM compliant testbench
  - Demonstrate key applications of VMM features
- **Roadmap**
  - DUT is FIFO *(in SV with SVA)*
  - SystemVerilog Testbench with VMM for
    - Generation of random transactions
      With transactor, through channel
    - Consumption of transactions from channels
      With transactor
    - Creation of monitor transactions
      Through channel, for consumption by scoreboard
- **Not Addressed here**
  - Factories
  - Callbacks

Language Supported by a Methodology

Ben Cohen
abv-sva.org

- **RVM is proven methodology based on Vera**
  (RVM is Synopsys Reference Verification Methodology based on Vera)
  - **Framework for testbench software**
  - **Transaction-based & coverage-based**
  - **Supported by reusable library**
- **VMM is adaptation of RVM for SystemVerilog**
  - **Transaction-based & coverage-based**
  - **Supported by reusable library**
  - **Documented in books**
    - ***Verification Methodology Manual for SystemVerilog, 2005 Springer***
    - ***SystemVerilog For Verification, A Guide to Learning the Testbench Language Features,*** Chris Spear, 2006 Springer

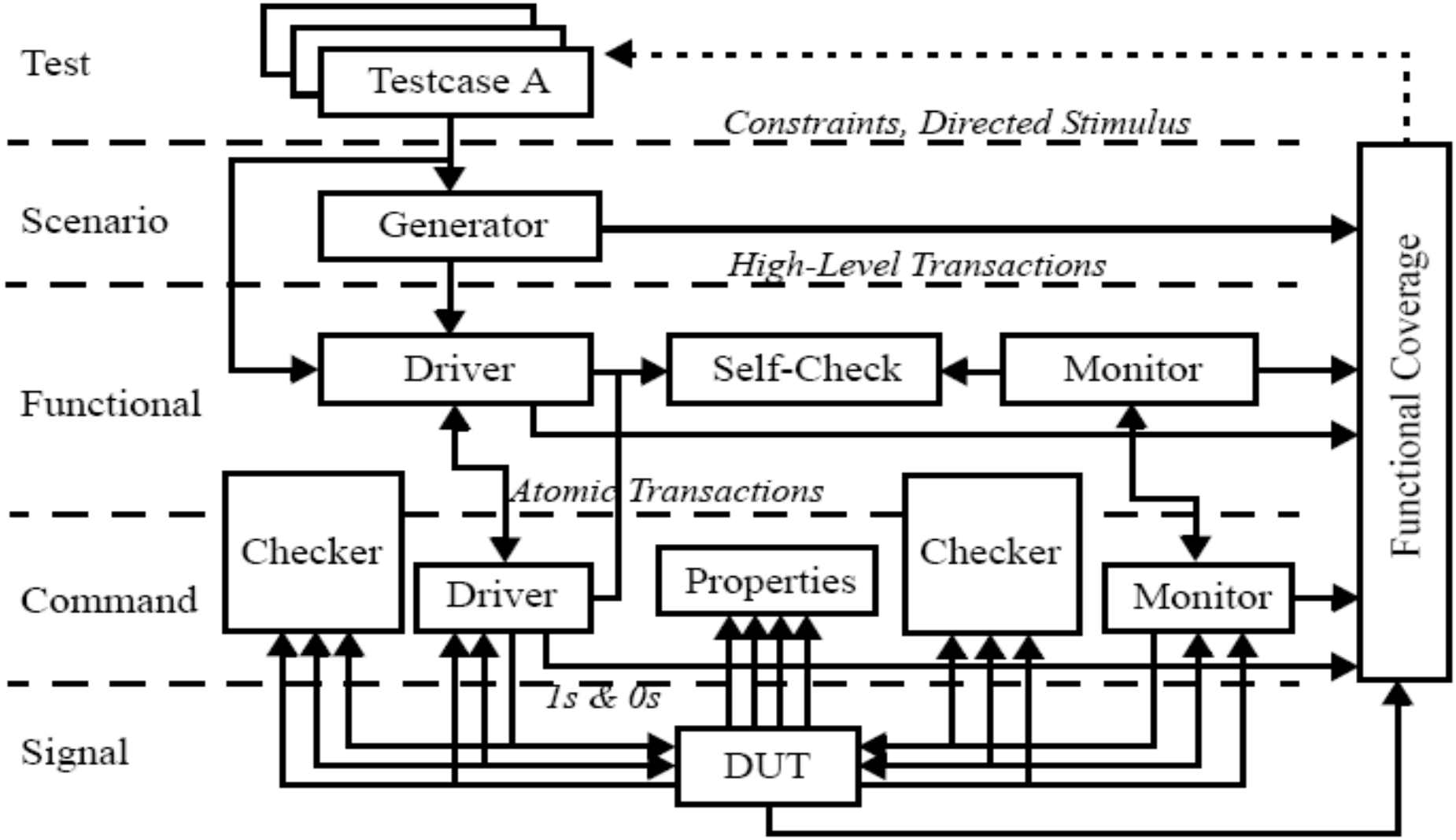
Goals of VMM

Ben Cohen
abv-sva.org

- **Verification productivity**
  - **Framework supports a consistent testbench flow**
    - **Structural aspect**
      - transaction / channel / transactor / environment
      - Monitor / scoreboard
      - Testbench construction
      - Automation in construction (macros)
    - **Run flow**
      - Execution sequence
      - Scenarios
      - Random and directed tests

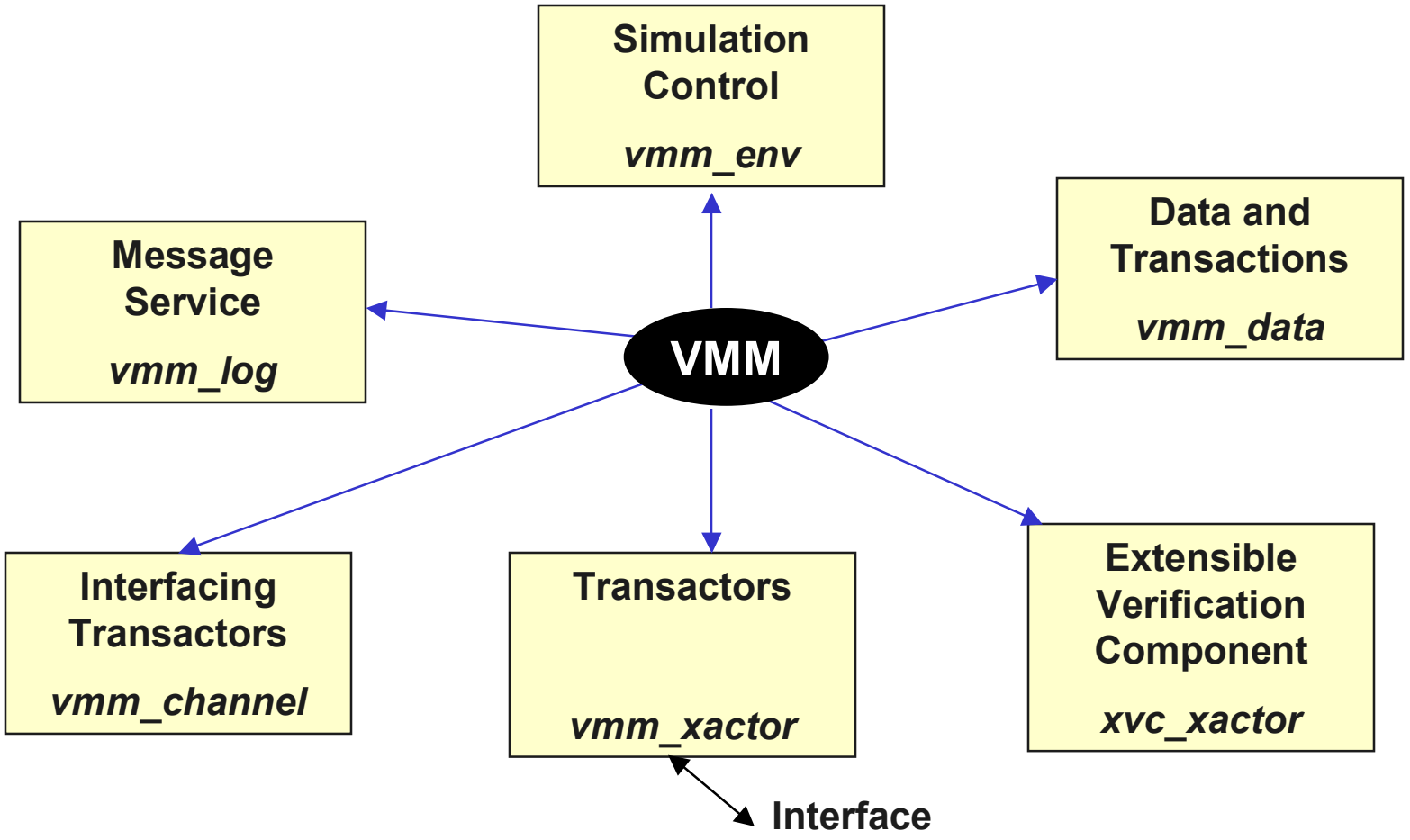
Layered Verification Environment Architecture. VMM Figure 4.2

Ben Cohen
abv-sva.org



VMM Adoption

Ben Cohen
abv-sva.org

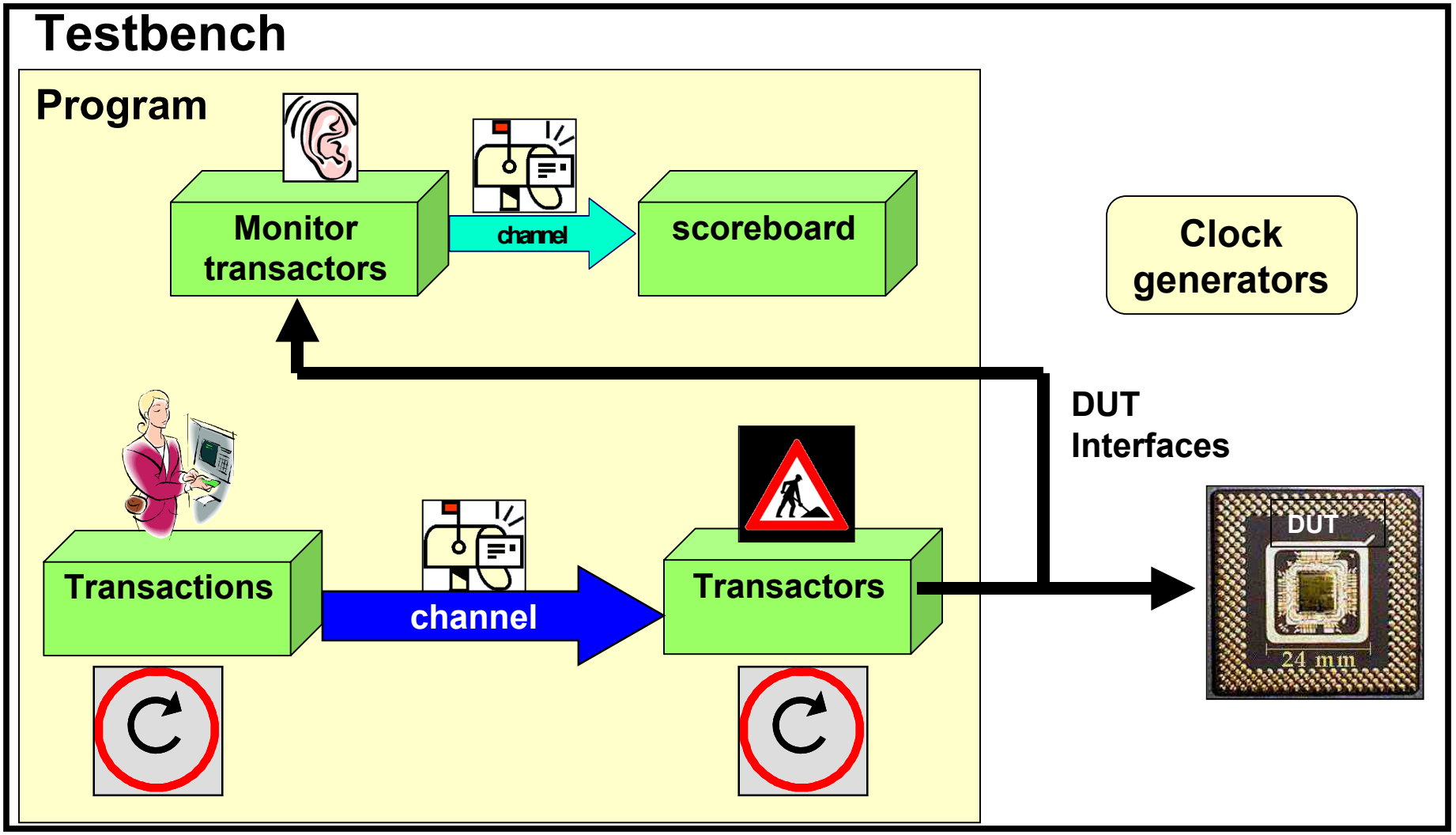


VMMing the FIFO testbench

Ben Cohen
abv-sva.org

Transaction-based verification Conceptual View

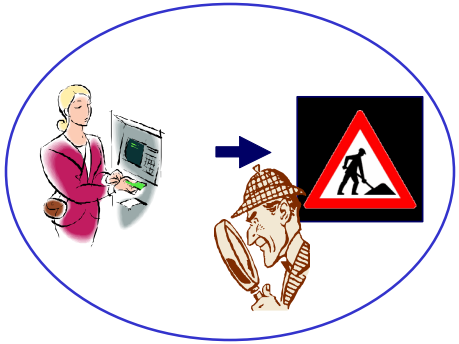
Ben Cohen
abv-sva.org

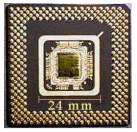


Testbench a la VMM

Ben Cohen
abv-sva.org

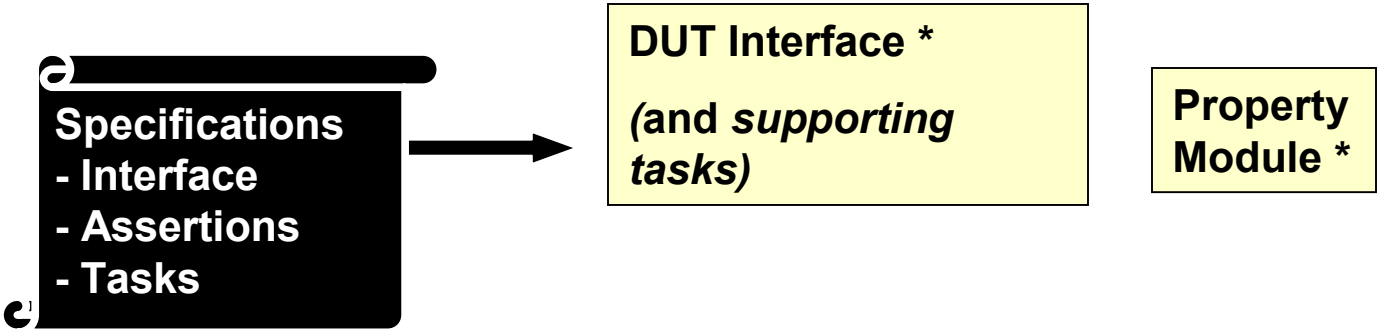
```
module fifo_tb;
  timeunit 1ns; timeprecision 100ps;
  logic clk = 1'b0;
  logic  reset_n;
  fifo_if f_if(.*);
  fifo_test_pgm utest_pgm (.fifo_if_0(f_if),
                           .reset_n(reset_n)
                            );
  fifo fifo_rtl_1(.f_if(f_if), .*);

  bind fifo fifo_props fifo_props_1
            (clk, reset_n, f_if);

   initial forever #50 clk = ~clk;
//..
endmodule : fifo_tb
```





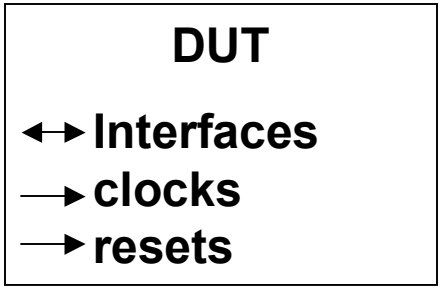
Design Process Supporting TBV Methodology with SystemVerilog

Ben Cohen
abv-sva.org

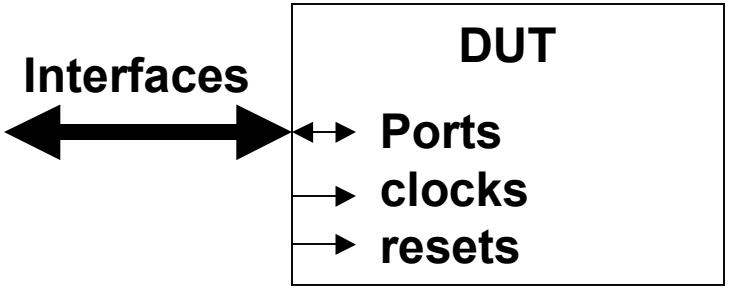


* Recommended

Interfaces and Ports
SystemVerilog Style



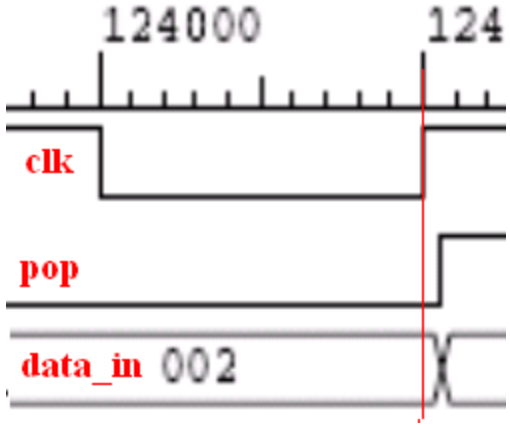
Ports only
Verilog Style



FIFO Interface and modports and clocking block

Ben Cohen
abv-sva.org

```
interface fifo_if(input wire clk,
                  input wire reset_n);
   timeunit 1ns;
   timeprecision 100ps;
//   import fifo_pkg::*;
  logic  push;
  logic  pop;
  wire  full;
  wire  empty;
  logic  error;
  word_t data_in;
  word_t data_out;
  parameter hold_time=3;
  parameter setup_time = 5;
```

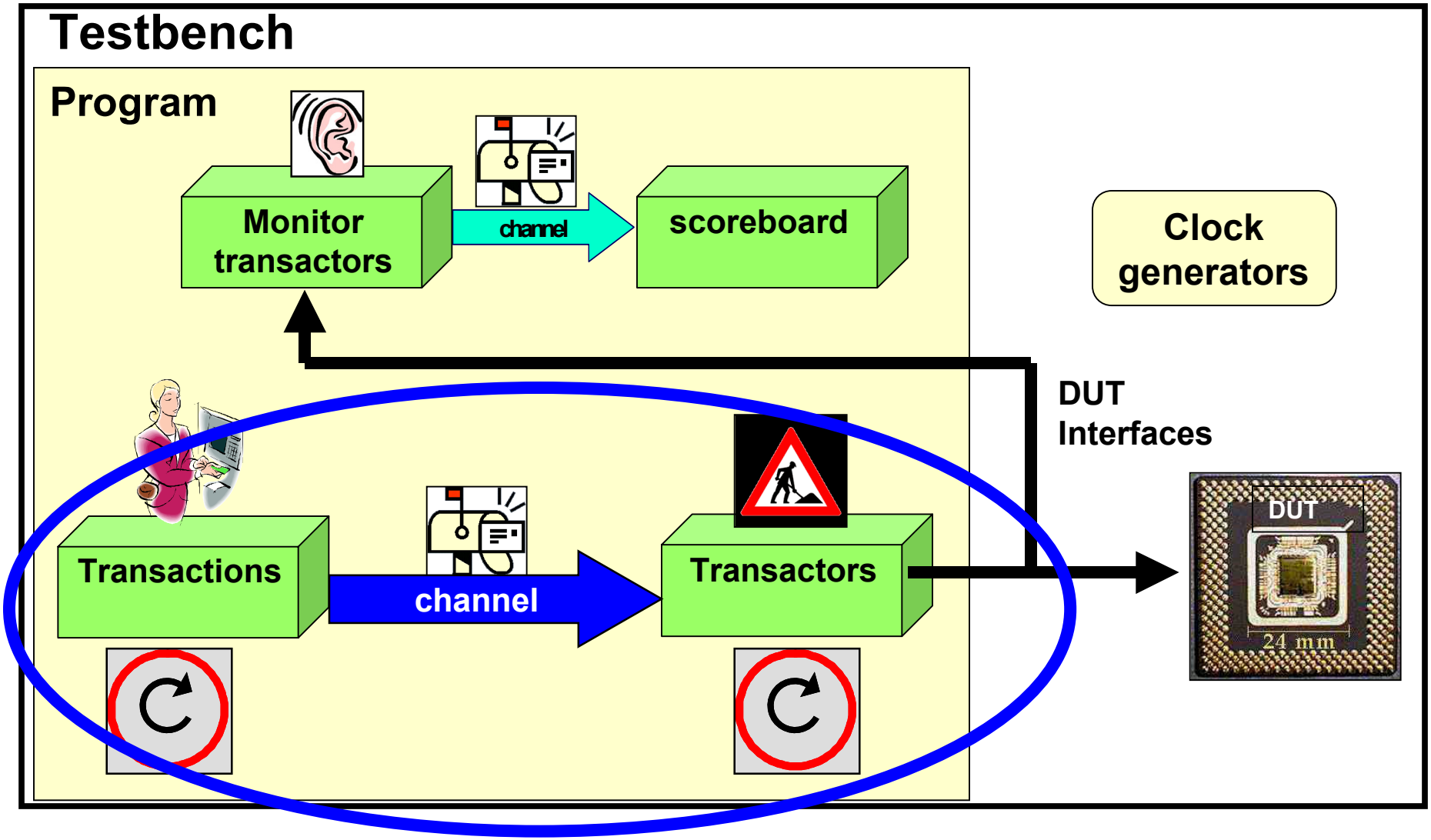


Identifies sampling and delays

```
clocking driver_cb @ (posedge clk);
   default input #setup_time
                 output #hold_time;
     input  empty, full, data_out, error;
     output data_in, push, pop;
  endclocking : driver_cb

 modport fdrvr_if_mp (clocking driver_cb);
```

Transaction-based verification Conceptual View

Ben Cohen
abv-sva.org

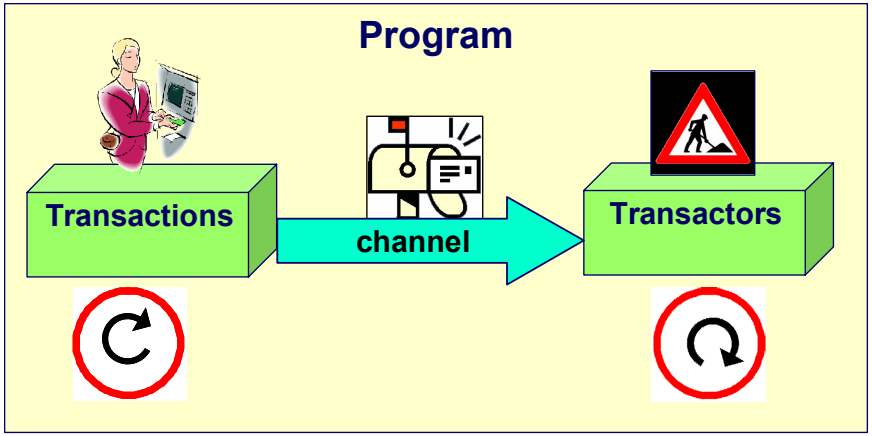


A Transaction

Ben Cohen
abv-sva.org

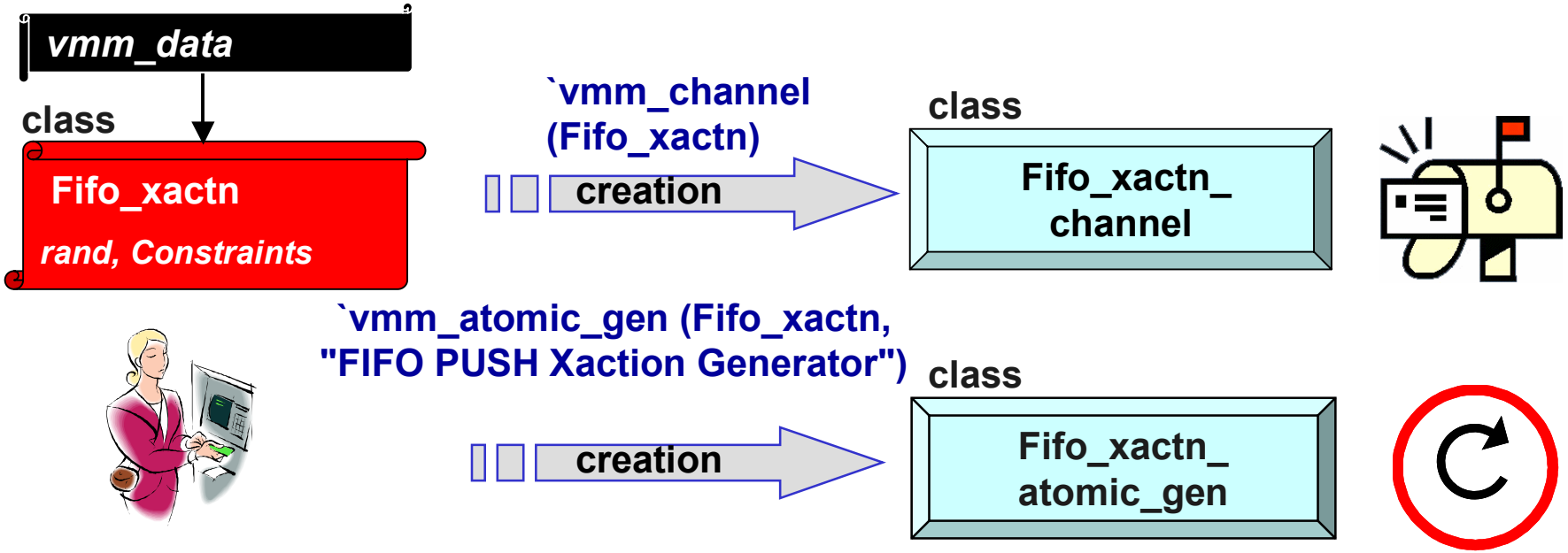
```
class Fifo_xactn extends vmm_data;
  rand  fifo_scen_t kind;
  rand logic [BIT_DEPTH-1:0]  data;
  rand int idle_cycles;
  time xactn_time;
  ..
  constraint cst_xact_kind {
    kind dist {
      PUSH := 5,
      POP := 0,
      IDLE := 8,
      RESET := 0
    };
  }
 ..
endclass:Fifo_xactn
```

extern virtual function Fifo_xactn
copy(vmm_data cpy = null);



Creation of Transaction Channel and Generator Classes

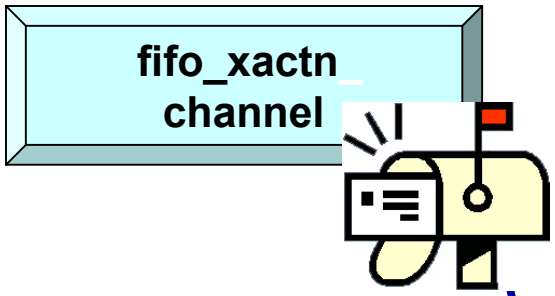
Ben Cohen
abv-sva.org



Consumption of Transactions from Channel

Ben Cohen
abv-sva.org

```
class Fifo_cmd_xactor extends vmm_xactor;
  virtual fifo_if.fdrvr_if_mp f_if;
  Fifo_xactn_channel   in_chan;

function new( ..
            virtual fifo_if.fdrvr_if_mp new_vir_if,
            Fifo_xactn_channel     new_in_chan);
      ..  this.f_if = new_vir_if;
      this.in_chan = new_in_chan;
endfunction : new

 task main();
    …
    forever
    begin : main_loop
      fifo_xactn_push  push_xaction;
     this.in_chan.get(push_xaction);
     case (push_xaction.kind)
      PUSH :  this.push_task(push_xaction.data);
```



Started by fifo_env::start()

Push task in transactor

Ben Cohen
abv-sva.org

```
task push_task (logic [BIT_DEPTH-1:0] data);
    begin
      $display ("%0t %m Push data %0h ", $time, data);
      f_if.driver_cb.data_in <= data;  // using clocking block
      f_if.driver_cb.push <= 1'b1;
      f_if.driver_cb.pop  <= 1'b0;
      @ ( f_if.driver_cb);
      f_if.driver_cb.push <= 1'b0;
    end
 endtask : push_task
```

```
// in fifo_if
 clocking driver_cb @ (posedge clk);
   default input #setup_time
                 output #hold_time;
     input  empty, full, data_out, error;
     output data_in, push, pop;
  endclocking : driver_cb
```

Class Relationships (in UML)

Ben Cohen
abv-sva.org

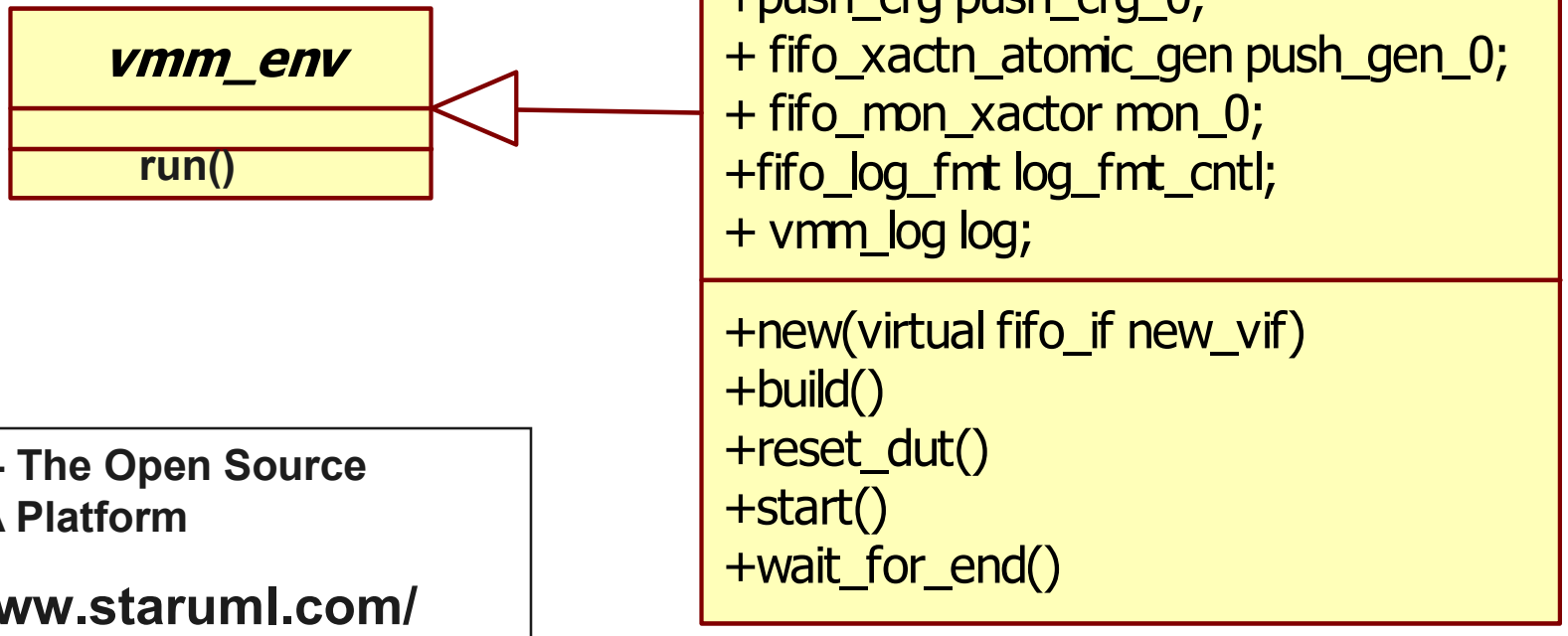
The Environment

Ben Cohen
abv-sva.org

- Instantiates classes
- Build connections
- Proper links including redirections
- Reset
- Configurations
- Start
- wait for end
- Stop
- Cleanup
- reports

Class Relationships
fifo_env

Ben Cohen
abv-sva.org



StarUML - The Open Source UML/MDA Platform

http://www.staruml.com/

Getting the Ball Rolling

Ben Cohen
abv-sva.org

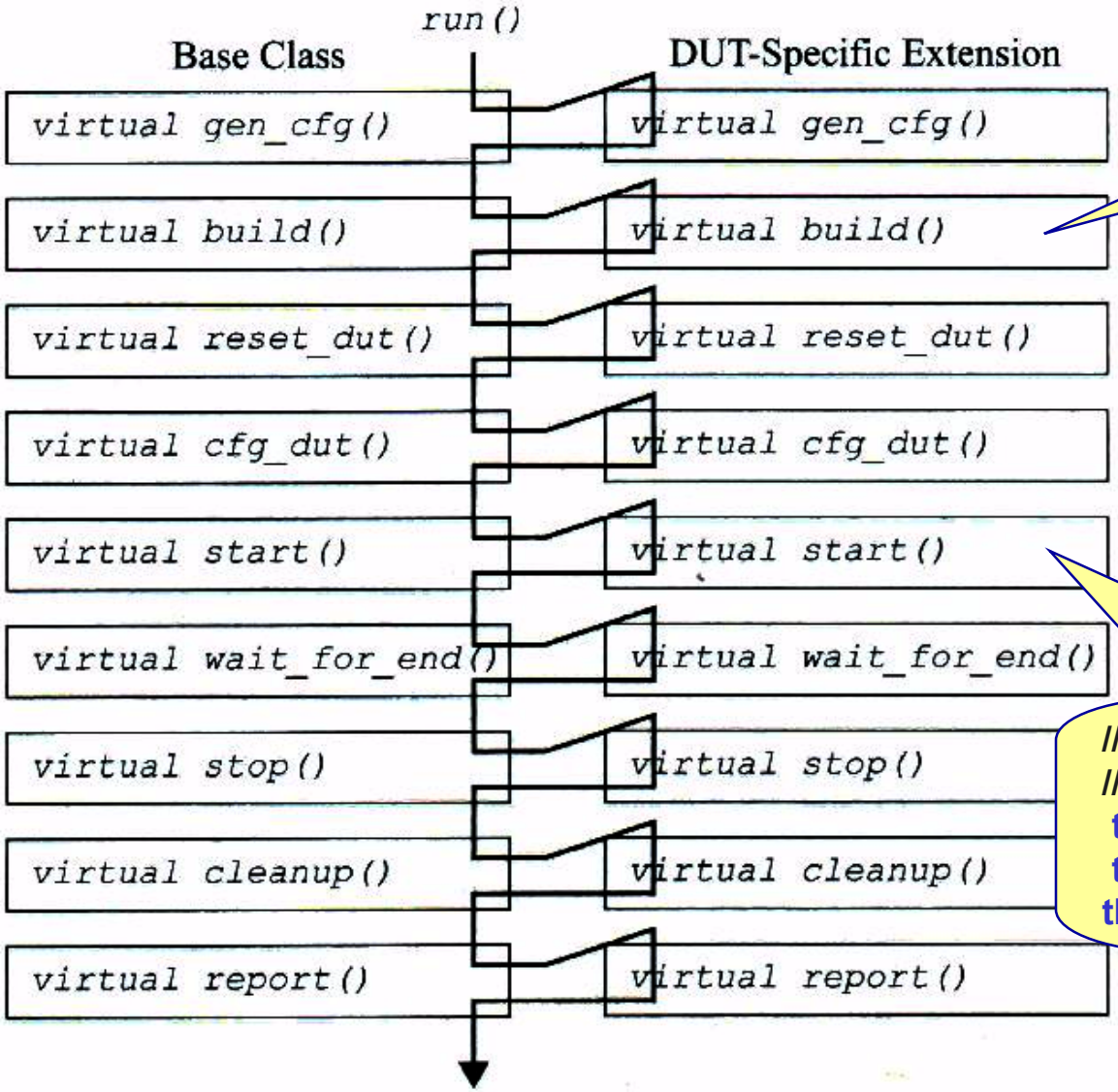
```
program automatic fifo_test_pgm (        fifo_if  fifo_if_0,
                         output logic    reset_n);
   timeunit 1ns; timeprecision 100ps;
  `include "vmm.sv"`include "fifo_common_include.sv"
  `include "fifo_xactn.sv"  `include "fifo_env.sv"
   vmm_log log;
   Fifo_env fifo_env_0;

  initial
  begin
    log = new("Pgm_Logger",0);
    fifo_env_0 = new(fifo_if_0);
    `vmm_note(log,"Started");
    fork : f1
      fifo_env_0.run();
    join_none
    #1000000;
  end
endprogram : fifo_test_pgm
```

```
`include "fifo_log_fmt.sv"
`include "fifo_cmd_xactor.sv"
`include "fifo_gen_xactor.sv"
`include "fifo_mon_xactor.sv"
```

automatic -> separate storage for each task call

Execution Sequence in *vmm_env*

Ben Cohen
abv-sva.org



// in *fifo_env*
Builds the connections

// in *fifo_env*
// starts the generator and transactor
this.push_gen_0.start_xactor();
this.fifo_cmd_xactor_0.start_xactor();
this.mon_0.start_xactor();

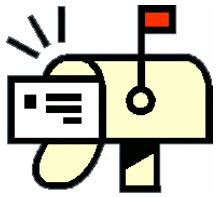
Creation and Consumption
Putting in together (build)

Ben Cohen
abv-sva.org

vmm_env

class

fifo_xactn_
channel



fifo_push_channel

```
function void build();
  this.fifo_push_channel =
            new("push_chan","0");

  this.push_gen_0 =
      new ("Push Xaction generator", 0);

this.fifo_cmd_xactor_0 =
    new("PUSH_XACTOR",
            0,
            vir_if,
            fifo_push_channel
            );

this.push_gen_0.out_chan =
            this.fifo_push_channel;
```

class

fifo_xactn_
atomic_gen

push_gen_0;

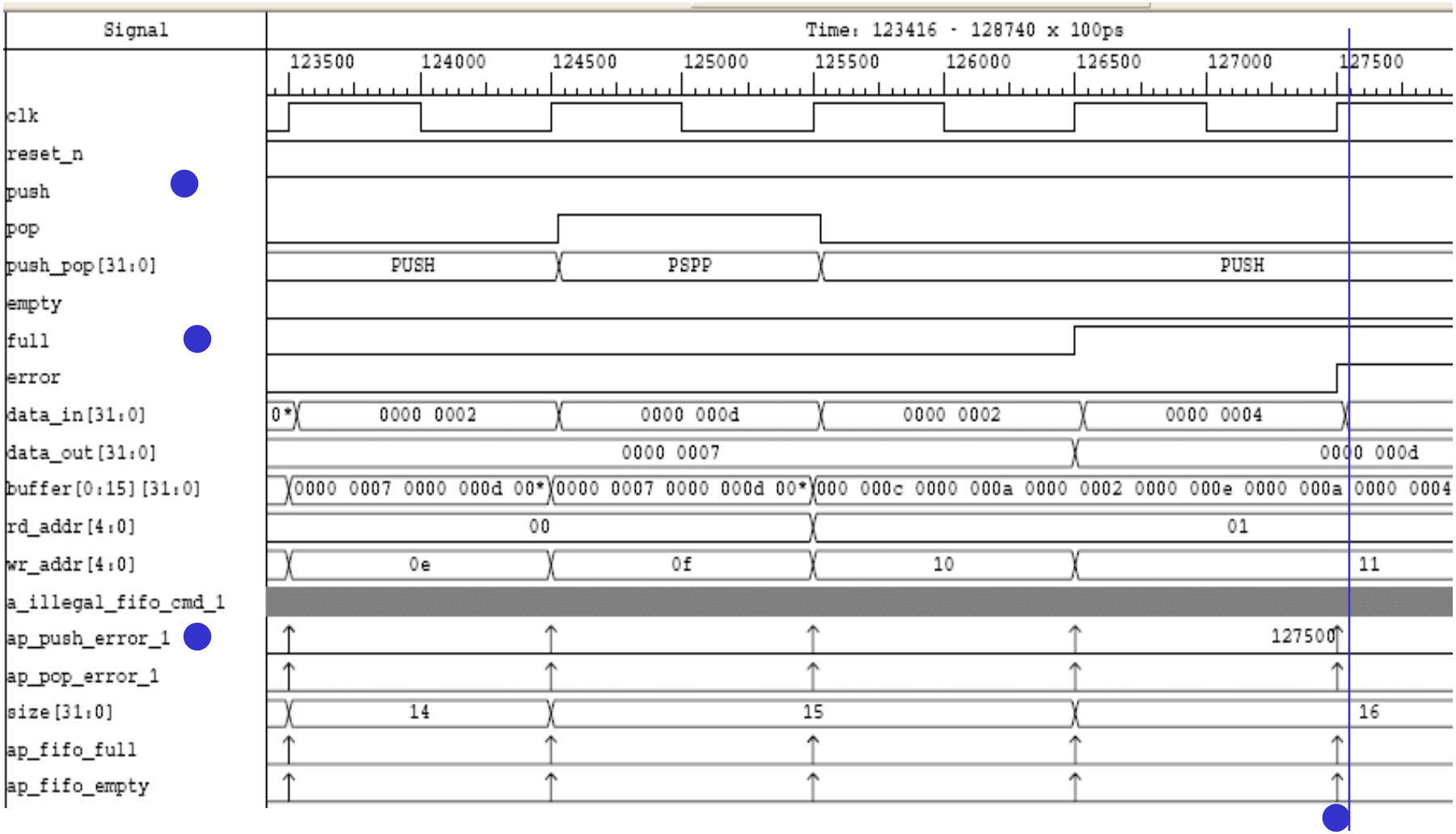
class

fifo_cmd_xactor

fifo_cmd_xactor_0;

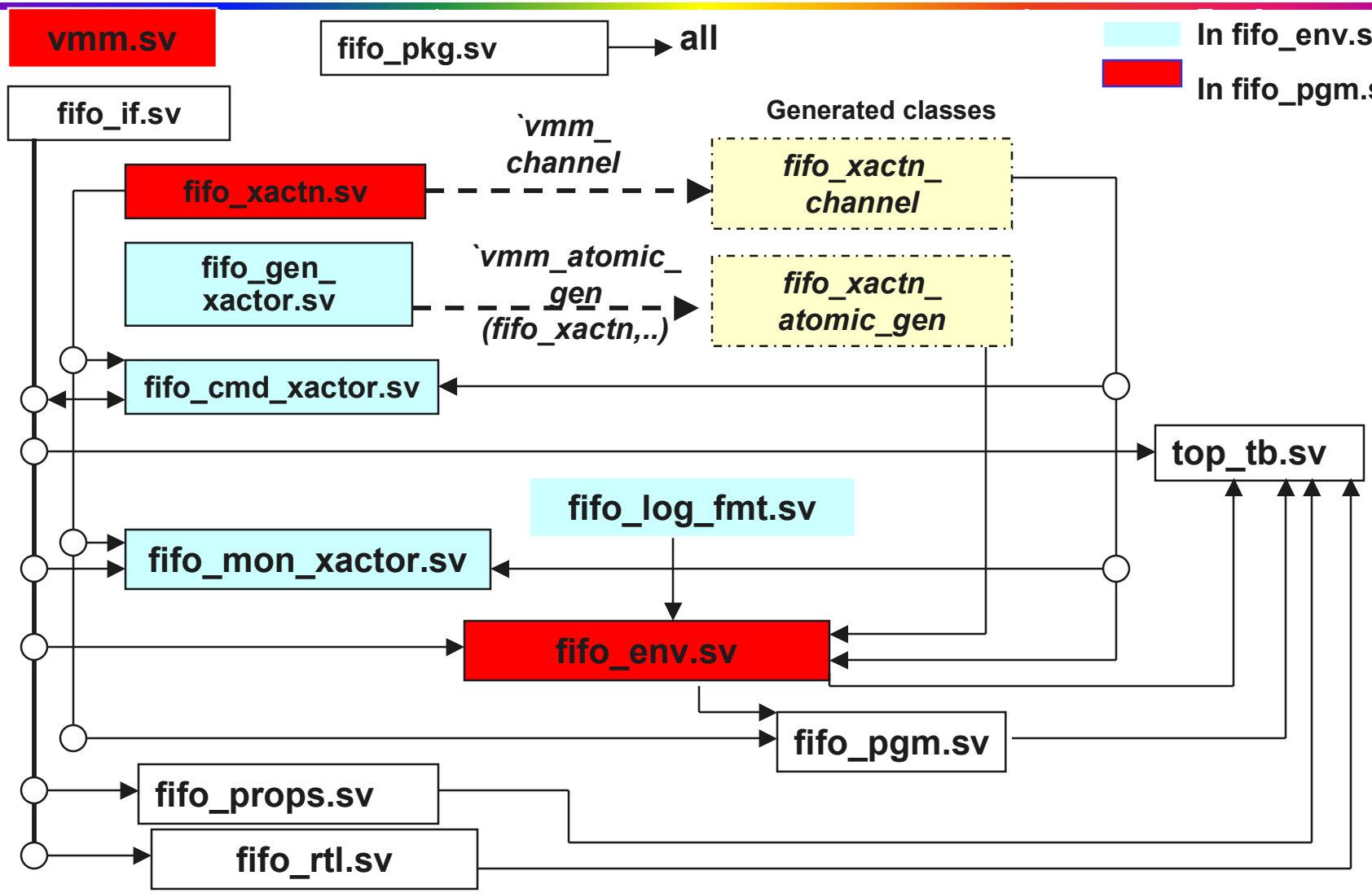
Push on FULL

Ben Cohen
abv-sva.org



File Structure

Ben Cohen
abv-sva.org



Conclusion

Ben Cohen
abv-sva.org

- VMM provides a consistent methodology
  - Layered testbench construction
  - Supported by a standard library
    - Base and utility classes
  - Unification in style and construction of testbench
  - Quick build of a layered and reusable testbench
  - Access to high-level tests
    - Constrained-random stimulus
    - Functional coverage
    - Directed tests
- Assertions and coverage do help in verification

Questions?

Ben Cohen
abv-sva.org

- Ease of use / learning / Debug ?
- Reuse?
- Transaction / transactor / generators
- Callback?
- Factory?
- Other TLM methodologies?