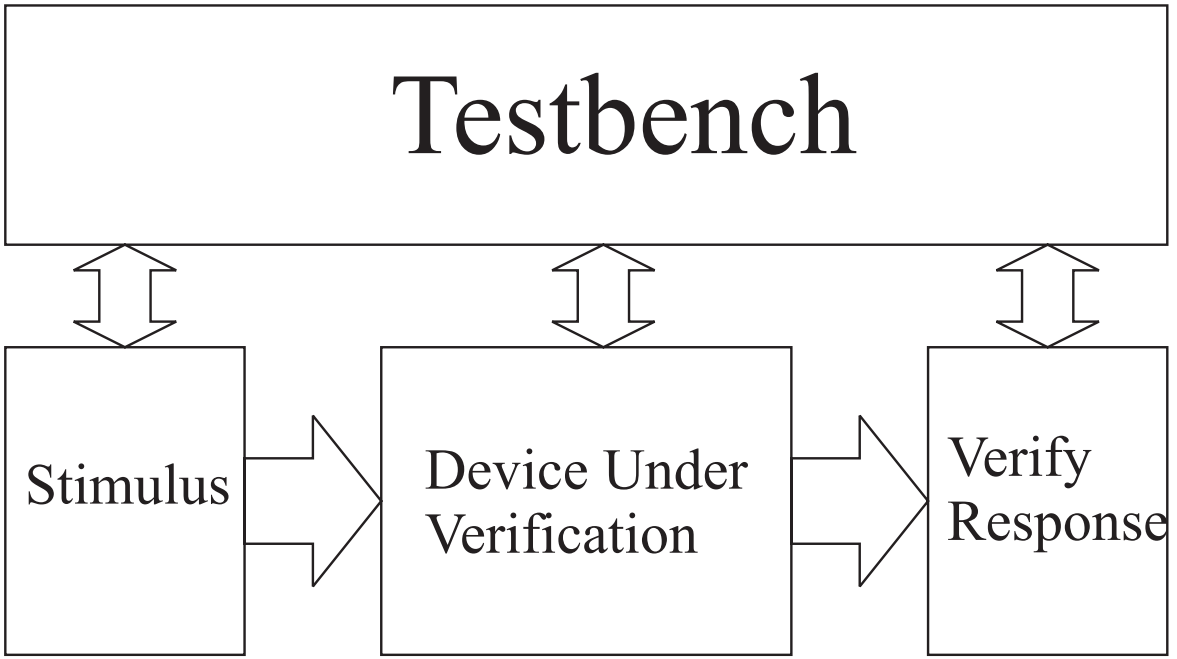
Modular testbench example

- Simple example to illustrate modularity



- Verify a sequential Booth multiplier (DUV)

Top Level

```
module testbooth;

parameter N = 16;

logic signed [N-1:0] ain, bin;
logic signed [2*N-1:0] qout;
logic clk, load, n_reset;
logic ready;

clock_gen #(10.0, 10.0) c0 (.*);
stimulus s0 (.*);
booth #(N, N, 2*N) b0 (.*);  //DUV
verify v0 (.*);

endmodule
```

Clock generator

```
module clock_gen #(parameter ClockFreq_MHz = 100.0,  ResetWidth=10.0)
      (output logic clk, n_reset);
timeunit 1ns;
timeprecision 100ps;

parameter ClockHigh = (500.0)/ClockFreq_MHz;

initial
  begin
       n_reset = '1;
       clk = '0;
  #ResetWidth n_reset = '0;
  #ResetWidth n_reset = '1;
  forever #ClockHigh clk = ~clk;
  end

endmodule
```

Note that this is a module

Stimulus

```
program stimulus #(parameter NA = 16, NB = 16)
       (output logic signed [NA-1:0] ain,
        output logic signed [NB-1:0] bin,
        output logic load);

initial
  begin
  #30ns ain = -10;
        bin = 10;
  #10ns load = '1;
  #10ns load = '0;
  end

endprogram
```

Program – see notes on Simulation

Executed in different region Of event queue to DUV.

Program may only contain initial blocks.

Assertions

```
module verify #(parameter AL = 8, BL = 8, QL = AL+BL)  (input logic signed [QL-1:0] qout,
  input logic ready,  input logic signed [AL-1:0]ain, input logic signed [BL-1:0] bin, input logic clk, load);

default clocking clock_block
 @(posedge clk);
endclocking

property LoadReady;
  load |=> ##AL ready;
endproperty

property Product;
  load |=> ##AL (qout == ain*bin);
endproperty

assert property (LoadReady);

assert property (Product)
else $error("%d * %d gives %d", ain, bin, qout);

endmodule
```