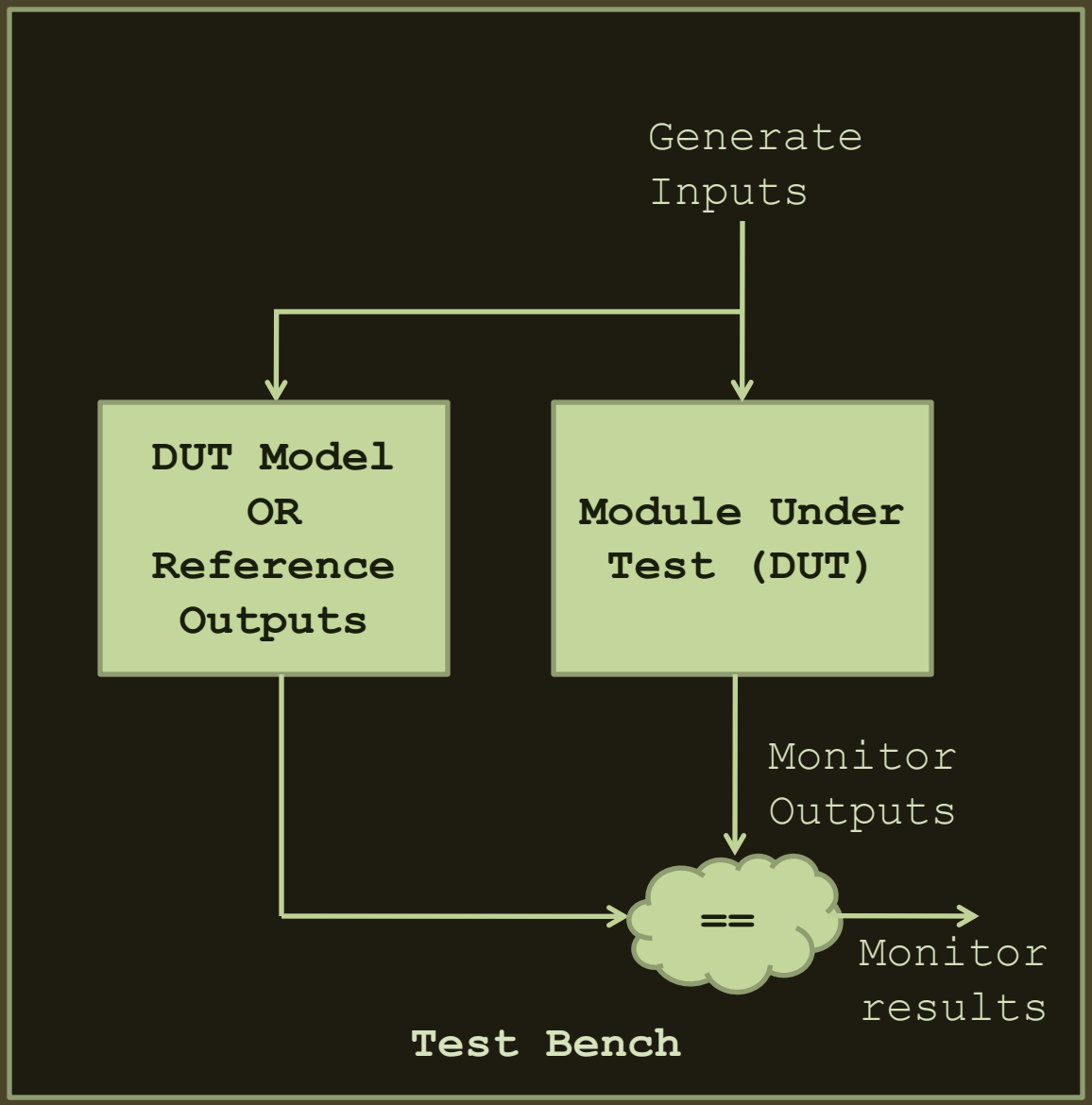
Verilog

Logic Verification

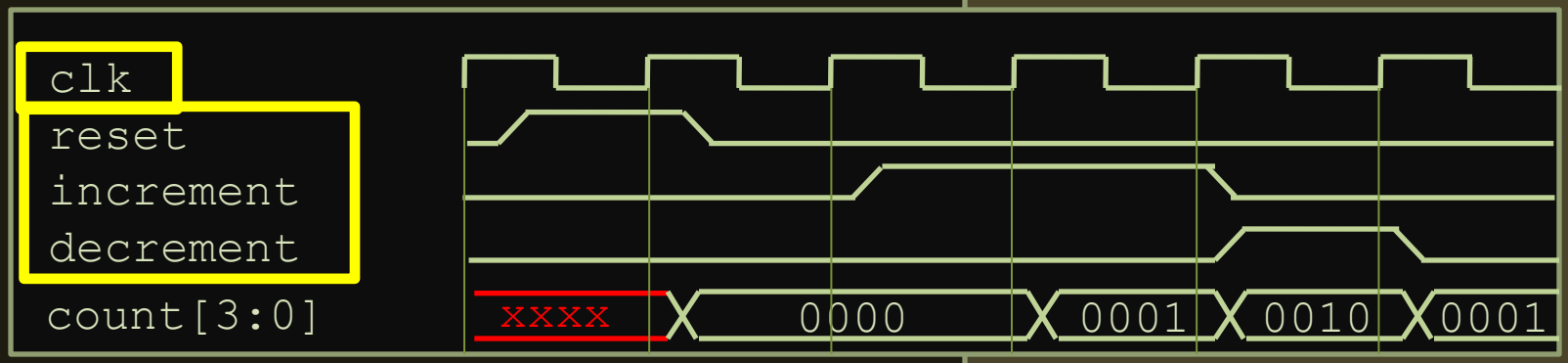

Generate
Inputs
DUT Model
OR
Reference
Outputs
Module Under
Test (DUT)
Monitor
Outputs
==
Monitor
results
Test Bench

Example: Test Bench for counter

Example: Test Bench for counter

```
module tb_counter;

 reg clk;
 reg reset;
 reg increment;
 reg decrement;
 wire [3:0] count;

counter counter_inst(
.clk          ( clk         ),
.reset        ( reset       ),
.increment    ( increment   ),
.decrement    ( decrement   ),
.count        ( count       )
);

endmodule
```

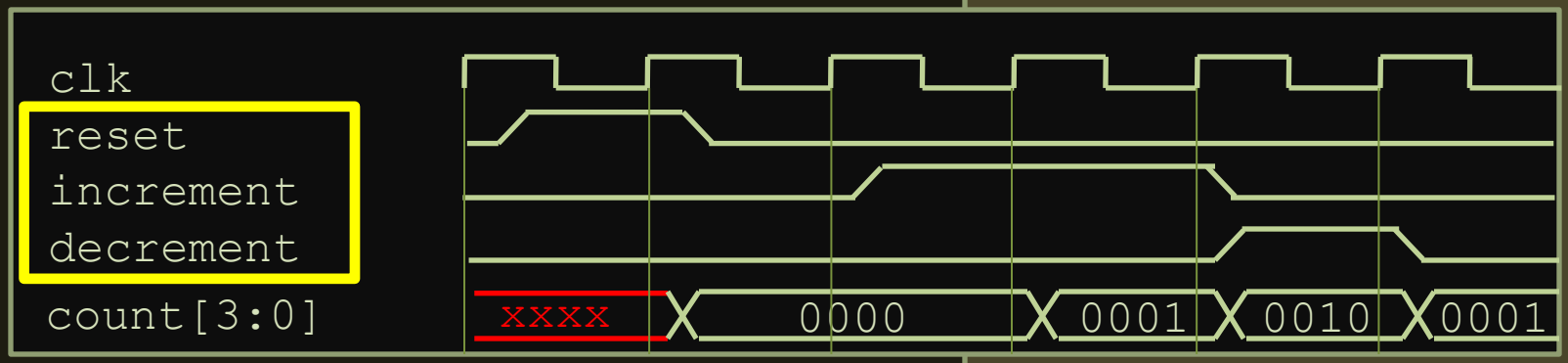
```
module tb_counter;
```




```
initial
begin
   clk = 0;
   forever
       #5 clk = ~clk;
end



endmodule
```

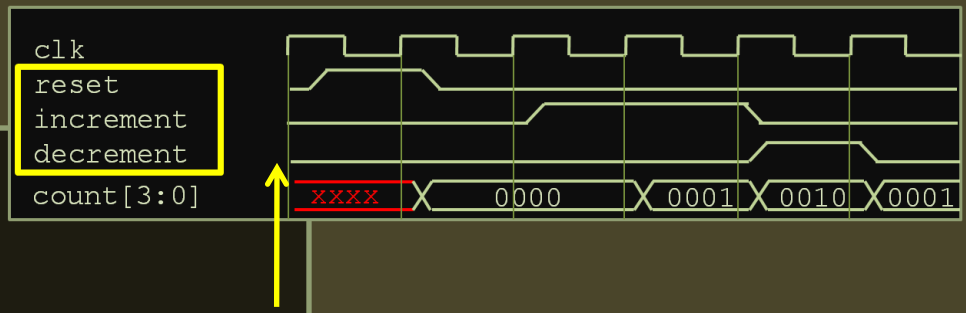
C style block executing statements sequentially unless we move forward in time axis using # ,@(something), wait etc.

```
module tb_counter;

endmodule
```



C style block executing statements sequentially unless we move forward in time axis using # ,@(something), wait etc.

```
initial
begin
    increment = 0;
    decrement = 0;
    reset = 0;
    @(posedge clk)
        reset <= #1 1;
    @(posedge clk)
        reset <= #1 0;
    @(posedge clk);
        #1 increment = 1;
    repeat(2) @(posedge clk);
        #1 increment = 0;
        #1 decrement = 1;
    @(posedge clk);
        #1 decrement = 0;
    repeat(20)@(posedge clk);
    $stop;


end
```



C style block executing statements sequentially unless we move forward in time axis using # ,@(something), wait etc.

```
module tb_counter;
// Variables Declaration
reg clk;
reg reset;
reg increment;
reg decrement;
wire [3:0] count;
// DUT instantiation
counter counter_inst(
.clk        ( clk                   ),
.reset      ( reset                 ),
.increment  ( increment             ),
.decrement  ( decrement             ),
.count      ( count                 )
);
//-------------------------------------------------
// Clock Generation
initial
begin
   clk = 0;
   forever
     #5 clk = ~clk;
end
//-------------------------------------------------
// Main Simulation
initial
begin
    increment = 0;
    decrement = 0;
    reset = 0;
   @(posedge clk)
      reset <= #1 1;
   @(posedge clk)
      reset <= #1 0;
   @(posedge clk);
      #1 increment = 1;
   repeat(2) @(posedge clk);
      #1 increment = 0;
      #1 decrement = 1;
   @(posedge clk);
       #1 decrement = 0;
   repeat(20)@(posedge clk);
   $stop;
end
endmodule
```

DUT Instantiation and Variable Declarations

Clock Generation

Monitor Results

Main simulation generating inputs in C style code

```
module tb_counter;
// Variables Declaration
reg clk;
reg reset;
reg increment;
reg decrement;
wire [3:0] count;
// DUT instantiation
counter counter_inst(
.clk        ( clk                   ),
.reset      ( reset                 ),
.increment  ( increment             ),
.decrement  ( decrement             ),
.count      ( count                 )
);
//-------------------------------------------------
// Clock Generation
initial
begin
   clk = 0;
   forever
     #5 clk = ~clk;
end
//-------------------------------------------------
// Main Simulation
initial
begin
    incremen
    decremen
    reset =
   @(posedge
      reset
   @(posedge
      reset
   @(posedge
      #1 inc
   repeat(2) @(posedge clk);
      #1 increment = 0;
      #1 decrement = 1;
   @(posedge clk);
       #1 decrement = 0;
   repeat(20)@(posedge clk);
   $stop;
end
endmodule
```

DUT Instantiation and Variable Declarations

Clock Generation

Monitor Results