# Assignment 7

# Assertion Based Verification

## 1. Immediate Assertion

Code :

```
/*
Date : 14/12/2023
module : assert_immediate.sv
Descrpition : System verilog - Immediate assertion -
*/

module assert_immediate();

reg  clk, grant, request;
time current_time;

initial begin
  clk = 0;
  grant   = 0;
  request = 0;
  #7 request = 1;
  #7 grant = 1;
  #7 request = 0;
  #7 $finish;
end

always #1 clk = ~clk;

always @ (posedge clk)
begin
  if (grant == 1) begin
     CHECK_REQ_WHEN_GNT : assert (grant && request) begin
        $display ("Seems to be working as expected");
     end else begin
        current_time = $time;
        #1 $error("assert failed at time %0t", current_time);
     end
  end
end

endmodule
```

**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own

testbench :

```
// Code your testbench here
// or browse Examples
module tb();
 initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule


/*output :

assert failed at time 21
time 22ns
assert failed at time 23
timw 24ns
assert failed at time 25
time 26ns
*/
```

## 2. Concurrent Assertion :

Code :

```
/*
Date : 14/12/2023
module : concurren_assertion
Descrpition : System verilog - concurrent assertion - behaviour evaluated over a
time period.The keyword "property" distinguishes a concurrent assertion from an
immediate assertion.
*/
module concurrent_assertion(
  input wire clk,req,reset,
  output reg gnt);

sequence req_gnt_seq;
  (~req & gnt) ##1 (~req & ~gnt);
endsequence

property req_gnt_prop;
  @ (posedge clk)
    disable iff (reset)
      req |-> req_gnt_seq;
```

```
endproperty

req_gnt_assert : assert property (req_gnt_prop)
                 else
                 $display("@%0dns Assertion Failed", $time);

always @ (posedge clk)
  gnt <= req;
endmodule
```

Testbench :

```
/*
Date : 14/12/2023
module : concurren_assertion_tb
*/
`timescale 1us/100ns
module concurrent_assertion_tb();

reg clk = 0;
reg reset, req = 0;
wire gnt;

always #3 clk ++;

initial begin
  reset <= 1;
  #20 reset <= 0;
  // Make the assertion pass
  #100 @ (posedge clk) req  <= 1;
  @ (posedge clk) req <= 0;
  // Make the assertion fail
  #100 @ (posedge clk) req  <= 1;
  repeat (5) @ (posedge clk);
  req <= 0;
  #10 $finish;
end
concurrent_assertion dut (clk,req,reset,gnt);
  // to display waveform.
   initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```

**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own

```
/*
ouptut :
@129000ns Assertion Failed
@237000ns Assertion Failed
@243000ns Assertion Failed
@249000ns Assertion Failed
@255000ns Assertion Failed
@261000ns Assertion Failed
*/
```

## 3. SVA built in methods : $rose ; $fell ; $Stable

Code :

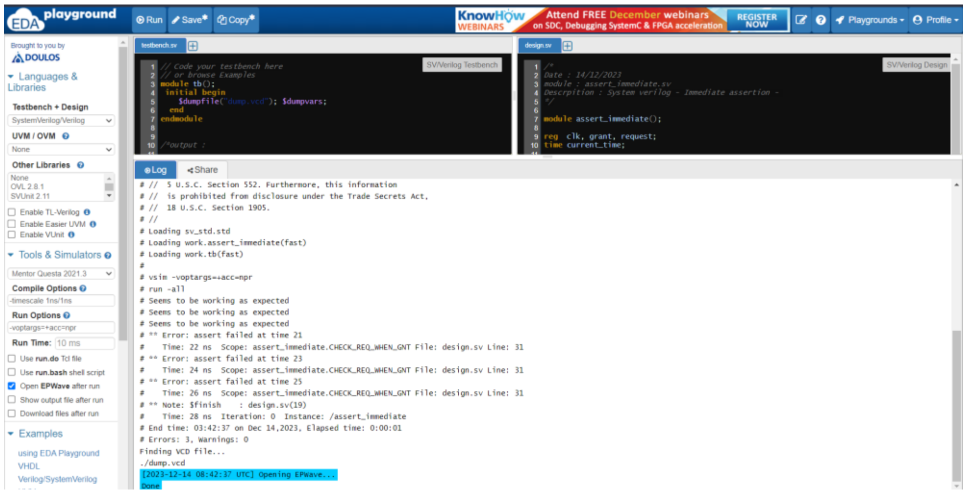
```
/*
Date : 14/12/2023
Descrpition : System verilog - $rose ; $fell ; $stable

$rose - a system task which can be used to detect a positive edge of the given
signal.

$fell - Similar to $rose but, this is to detect the negative edge of the given
signal.

$stable - returns true if the value of the expression did not change. Otherwise,
it returns false.
*/
module tb;
  bit a ;
  bit clk;

  sequence s_a;
    @(posedge clk) $rose(a); // sequence checks for "a" value at every +ve edge
if it doesn't occur it returns false. thus the assertion fails.
    // replace with $stable(a) & $fell(a) when needed.
  endsequence

  assert property(s_a);

   always #10 clk = ~clk;
    initial begin
      for(int i = 0; i<10; i++)begin
        a = $random;
        @(posedge clk);
```

**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own

```
        $display("[%0t] a = %0d", $time,a);
    end
      #20 $finish;
    end
  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```

## Outputs :

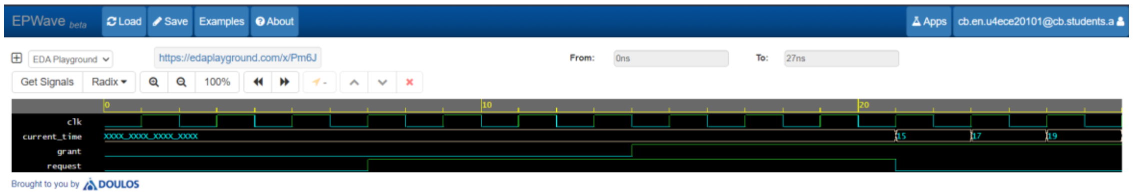
### Immediate Assertion output :



```
// Code your testbench here
// or browse Examples
module tb();
 initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule


/*output :
```

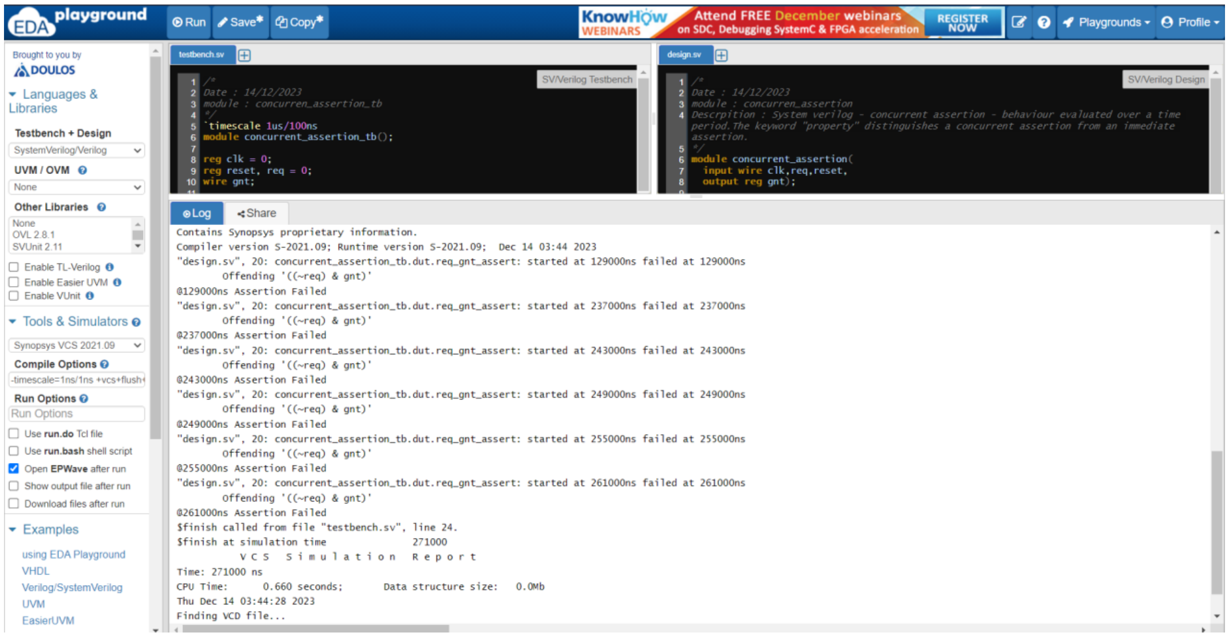
```
/*
Date : 14/12/2023
module : assert_immediate.sv
Descrpition : System verilog - Immediate assertion -
*/

module assert_immediate();

reg  clk, grant, request;
time current_time;
```

```
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
# //  18 U.S.C. Section 1905.
# //
# Loading sv_std.std
# Loading work.assert_immediate(fast)
# Loading work.tb(fast)
#
# vsim -voptargs=+acc=npr
# run -all
# Seems to be working as expected
# Seems to be working as expected
# Seems to be working as expected
# ** Error: assert failed at time 21
#    Time: 22 ns  Scope: assert_immediate.CHECK_REQ_WHEN_GNT File: design.sv Line: 31
# ** Error: assert failed at time 23
#    Time: 24 ns  Scope: assert_immediate.CHECK_REQ_WHEN_GNT File: design.sv Line: 31
# ** Error: assert failed at time 25
#    Time: 26 ns  Scope: assert_immediate.CHECK_REQ_WHEN_GNT File: design.sv Line: 31
# ** Note: $finish    : design.sv(19)
#    Time: 28 ns  Iteration: 0  Instance: /assert_immediate
# End time: 03:42:37 on Dec 14,2023, Elapsed time: 0:00:01
# Errors: 3, Warnings: 0
Finding VCD file...
./dump.vcd
[2023-12-14 08:42:37 UTC] Opening EPWave...
Done
```



**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own

## Concurrent Assertion :



Testbench + Design: SystemVerilog/Verilog
UVM / OVM: None
Tools & Simulators: Synopsys VCS 2021.09
Compile Options: -timescale=1ns/1ns +vcs+flush+
Open EPWave after run ✓

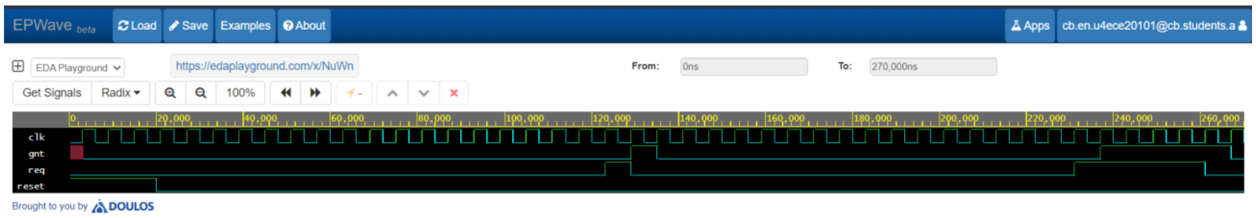
testbench.sv

```
/*
Date : 14/12/2023
module : concurren_assertion_tb
*/
`timescale 1us/100ns
module concurrent_assertion_tb();

reg clk = 0;
reg reset, req = 0;
wire gnt;
```

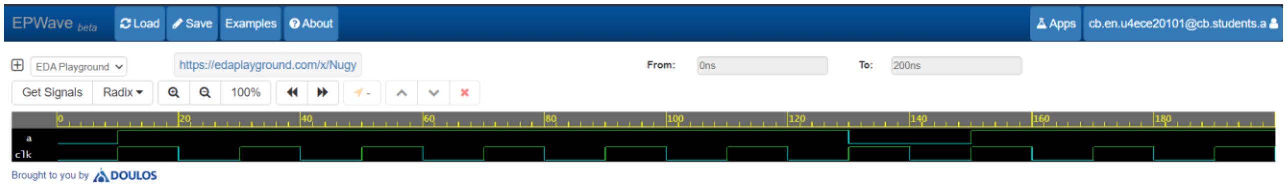
design.sv

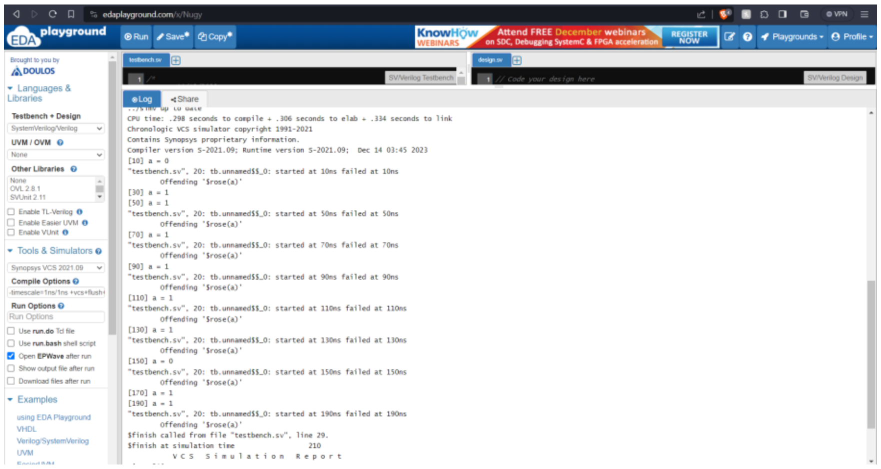
```
/*
Date : 14/12/2023
module : concurren_assertion
Descrpition : System verilog - concurrent assertion - behaviour evaluated over a time
period.The keyword "property" distinguishes a concurrent assertion from an immediate
assertion.
*/
module concurrent_assertion(
  input wire clk,req,reset,
  output reg gnt);
```

```
Contains Synopsys proprietary information.
Compiler version S-2021.09; Runtime version S-2021.09;  Dec 14 03:44 2023
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 129000ns failed at 129000ns
        Offending '((~req) & gnt)'
@129000ns Assertion Failed
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 237000ns failed at 237000ns
        Offending '((~req) & gnt)'
@237000ns Assertion Failed
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 243000ns failed at 243000ns
        Offending '((~req) & gnt)'
@243000ns Assertion Failed
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 249000ns failed at 249000ns
        Offending '((~req) & gnt)'
@249000ns Assertion Failed
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 255000ns failed at 255000ns
        Offending '((~req) & gnt)'
@255000ns Assertion Failed
"design.sv", 20: concurrent_assertion_tb.dut.req_gnt_assert: started at 261000ns failed at 261000ns
        Offending '((~req) & gnt)'
@261000ns Assertion Failed
$finish called from file "testbench.sv", line 24.
$finish at simulation time                271000
           V C S   S i m u l a t i o n   R e p o r t
Time: 271000 ns
CPU Time:      0.660 seconds;       Data structure size:   0.0Mb
Thu Dec 14 03:44:28 2023
Finding VCD file...
```



## SVA built in methods- $rose $fell $stable





```
CPU time: .298 seconds to compile + .306 seconds to elab + .334 seconds to link
Chronologic VCS simulator copyright 1991-2021
Contains Synopsys proprietary information.
Compiler version S-2021.09; Runtime version S-2021.09;  Dec 14 03:45 2023
[10] a = 0
"testbench.sv", 20: tb.unnamed$$_0: started at 10ns failed at 10ns
        Offending '$rose(a)'
[30] a = 1
[50] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 50ns failed at 50ns
        Offending '$rose(a)'
[70] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 70ns failed at 70ns
        Offending '$rose(a)'
[90] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 90ns failed at 90ns
        Offending '$rose(a)'
[110] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 110ns failed at 110ns
        Offending '$rose(a)'
[130] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 130ns failed at 130ns
        Offending '$rose(a)'
[150] a = 0
"testbench.sv", 20: tb.unnamed$$_0: started at 150ns failed at 150ns
        Offending '$rose(a)'
[170] a = 1
[190] a = 1
"testbench.sv", 20: tb.unnamed$$_0: started at 190ns failed at 190ns
        Offending '$rose(a)'
$finish called from file "testbench.sv", line 29.
$finish at simulation time                210
           V C S   S i m u l a t i o n   R e p o r t
```

**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own

**Inference :**

Thus, from the above experiments we can understand the basic concepts of immediate and concurrent assertions and how the both differ and, we were able to learn about SVA built in methods $rose, $fell, $stable.

- Immediate Assertion : https://edaplayground.com/x/Pm6
- Concurrent Assertion : https://edaplayground.com/x/NuWn
- SVA built in methods - $rose $fell $stable : https://edaplayground.com/x/Nugy
- Output Screenshots – Assertion based verification

**Honour Pledge**: I affirm that I will not give or receive any unauthorized help on this lab, and that all work will be my own