Lab 1: Report on gen\_tick

**Table of Contents**

[Introduction 1](#__RefHeading___Toc410_1450623777)

[gen\_tick Module 1](#__RefHeading___Toc412_1450623777)

[Module Code 1](#__RefHeading___Toc422_1450623777)

[Explanation 2](#__RefHeading___Toc414_1450623777)

[Test Bench for gen\_tick Module 3](#__RefHeading___Toc416_1450623777)

[Code: 3](#__RefHeading___Toc424_1450623777)

[Explanation 7](#__RefHeading___Toc418_1450623777)

[Conclusion 7](#__RefHeading___Toc420_1450623777)

# Introduction

This report discusses the implementation and testing of the gen\_tick module in Verilog. The gen\_tick module generates a tick signal at a specified frequency based on a source clock input. The accompanying test bench verifies the functionality of the gen\_tick module by simulating different scenarios and configurations.

# gen\_tick Module

The gen\_tick module produces a tick signal at a given frequency (TICK\_FREQ) from an input source clock (SRC\_FREQ). The module includes parameters for the source clock frequency and the desired output tick frequency.

## Module Code

|  |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date: 06/06/2024 06:17:18 PM  // Design Name:  // Module Name: gen\_tick  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module gen\_tick #(  parameter SRC\_FREQ = 5000, // Source clock frequency in Hz  parameter TICK\_FREQ = 1 // Output tick frequency in Hz  )(  input src\_clk, // Source clock input  input enable, // Enable signal  output tick // Output tick signal  );  // Calculate the number of source clock cycles per tick  localparam integer TICK\_COUNT = SRC\_FREQ / TICK\_FREQ;  // Declare registers and wires here  reg [31:0] counter = 0;  reg tick\_reg = 0;  always @(posedge src\_clk) begin  if (!enable) begin  counter <= 0;  tick\_reg <= 0; // No tick if not enabled  end else begin  if (counter == TICK\_COUNT - 1) begin  counter <= 0;  tick\_reg <= ~tick\_reg; // Toggle the tick signal  end else begin  counter <= counter + 1;  end  end  end  // Assign the tick register to the output tick signal  assign tick = tick\_reg;  endmodule |

## Explanation

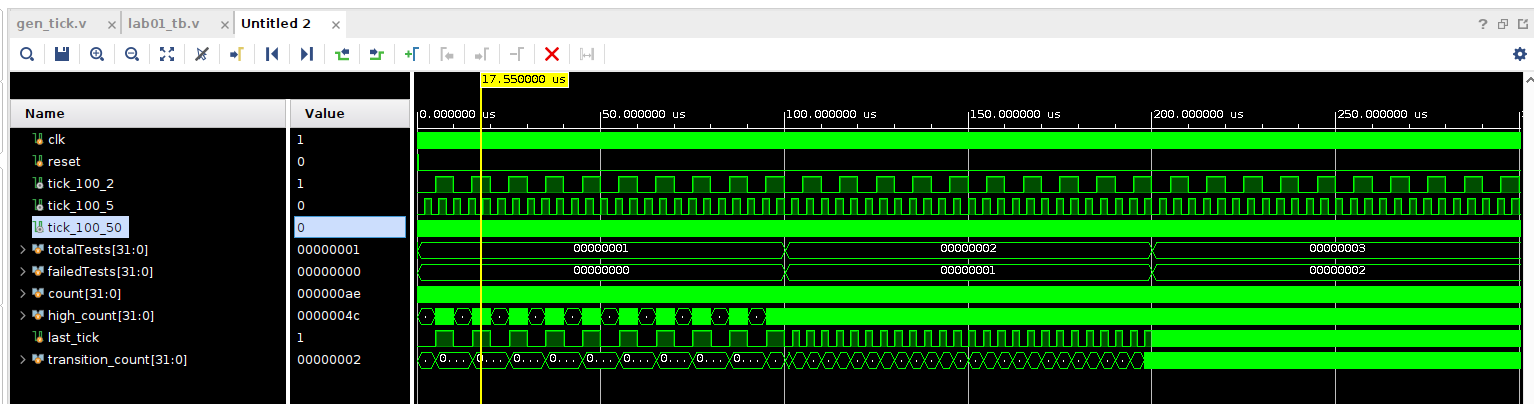
* Parameters:
  + SRC\_FREQ: Frequency of the input source clock in Hz.
  + TICK\_FREQ: Desired frequency of the output tick signal in Hz.
* Local Parameters:
  + TICK\_COUNT: Number of source clock cycles required for one tick period (SRC\_FREQ / TICK\_FREQ).
* Registers:
  + counter: A 32-bit register to count the source clock cycles.
  + tick\_reg: A register to hold the tick signal state.
* Always Block:
  + If enable is low, counter and tick\_reg are reset to 0.
  + If enable is high, counter increments on each rising edge of src\_clk.
  + When counter reaches TICK\_COUNT - 1, it resets to 0 and toggles the tick\_reg signal.
* Output Assignment:
  + The tick output is assigned the value of tick\_reg.

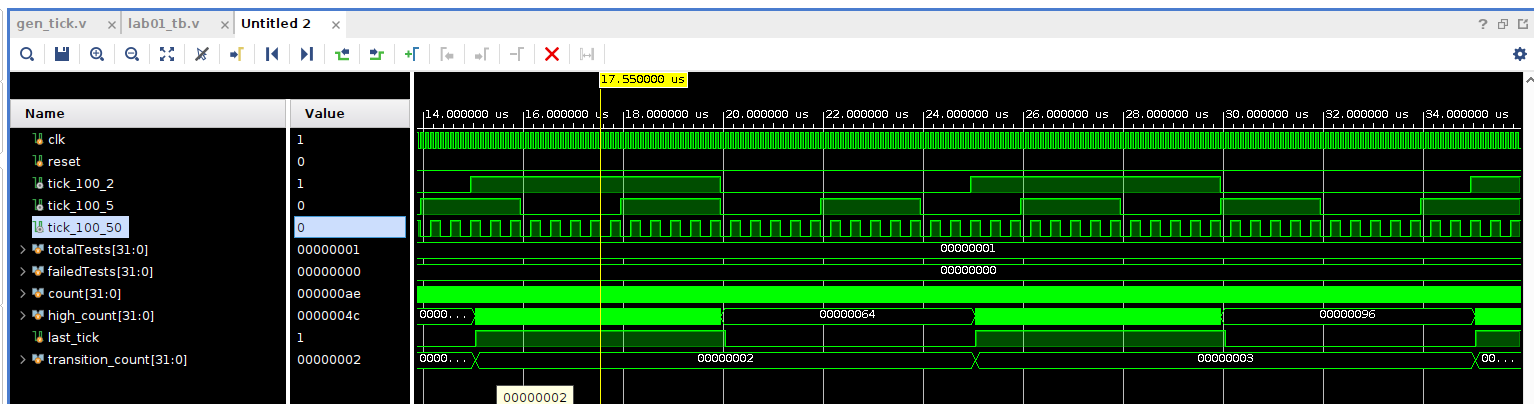
# Test Bench for gen\_tick Module

The test bench lab01\_tb is designed to verify the functionality of the gen\_tick module. It instantiates three instances of gen\_tick with different configurations and monitors their outputs.

## Code:

|  |
| --- |
| `timescale 1ns / 1ps  module lab01\_tb;  // Inputs  reg clk;  reg reset;  // Outputs  // -------------------------------------------------------  // Setup output file for possible debugging uses  // -------------------------------------------------------  initial  begin  $dumpfile("lab01.vcd");  $dumpvars(0);  end  // Declare wires for each unit under test  wire tick\_100\_2;  wire tick\_100\_5;  wire tick\_100\_50;  // -------------------------------------------------------  // Instantiate at least 3 Units Under Test (UUTs)  // -------------------------------------------------------  gen\_tick #(.SRC\_FREQ(100), .TICK\_FREQ(2)) uut\_100\_2 (  .src\_clk(clk),  .enable(1'b1),  .tick(tick\_100\_2)  );  gen\_tick #(.SRC\_FREQ(100), .TICK\_FREQ(5)) uut\_100\_5 (  .src\_clk(clk),  .enable(1'b1),  .tick(tick\_100\_5)  );  gen\_tick #(.SRC\_FREQ(100), .TICK\_FREQ(50)) uut\_100\_50 (  .src\_clk(clk),  .enable(1'b1),  .tick(tick\_100\_50)  );  initial begin  clk = 0; reset = 1; #50;  clk = 1; reset = 1; #50;  clk = 0; reset = 0; #50;  clk = 1; reset = 0; #50;    forever begin  clk = ~clk; #50;  end  end    integer totalTests = 0;  integer failedTests = 0;  integer count = 0;  integer high\_count = 0;  reg last\_tick = 0;  integer transition\_count = 0;  initial begin // Test suite  // Reset  @(negedge reset); // Wait for reset to be released (from another initial block)  @(posedge clk); // Wait for first clock out of reset  #10; // Wait 10 cycles  // ---------------------------------------------  // Testing Source clock 100Hz, Tick 2Hz  // ---------------------------------------------  $write("Test Source clock 100Hz, Tick 2Hz ... ");  totalTests = totalTests + 1;  while(count < 1000) begin  @(posedge clk);  if (last\_tick == 0 & tick\_100\_2 != last\_tick) begin  transition\_count = transition\_count + 1;  end  count = count + 1;  if (tick\_100\_2 == 1) begin  high\_count = high\_count + 1;  end  last\_tick = tick\_100\_2;  end  if (high\_count == 500 & transition\_count == 20) begin  $display("PASSED");  end else begin  $display("FAILED");  failedTests = failedTests + 1;  end  $display("Load (%d/%d): %0.2f", high\_count, count, 1.0 \* high\_count / count);  $display("Transition count: %d", transition\_count);  // Re-initialize counters for each test  last\_tick = 0;  transition\_count = 0;  count = 0;  high\_count = 0;  // ---------------------------------------------  // Testing Source clock 100Hz, Tick 5Hz  // ---------------------------------------------  $write("Test Source clock 100Hz, Tick 5Hz ... ");  totalTests = totalTests + 1;  while(count < 1000) begin  @(posedge clk);  if (last\_tick == 0 & tick\_100\_5 != last\_tick) begin  transition\_count = transition\_count + 1;  end  count = count + 1;  if (tick\_100\_5 == 1) begin  high\_count = high\_count + 1;  end  last\_tick = tick\_100\_5;  end  if (high\_count == 1000 & transition\_count == 50) begin  $display("PASSED");  end else begin  $display("FAILED");  failedTests = failedTests + 1;  end  $display("Load (%d/%d): %0.2f", high\_count, count, 1.0 \* high\_count / count);  $display("Transition count: %d", transition\_count);  // Re-initialize counters for each test  last\_tick = 0;  transition\_count = 0;  count = 0;  high\_count = 0;  // ---------------------------------------------  // Testing Source clock 100Hz, Tick 50Hz  // ---------------------------------------------  $write("Test Source clock 100Hz, Tick 50Hz ... ");  totalTests = totalTests + 1;  while(count < 1000) begin  @(posedge clk);  if (last\_tick == 0 & tick\_100\_50 != last\_tick) begin  transition\_count = transition\_count + 1;  end  count = count + 1;  if (tick\_100\_50 == 1) begin  high\_count = high\_count + 1;  end  last\_tick = tick\_100\_50;  end  if (high\_count == 500 & transition\_count == 100) begin  $display("PASSED");  end else begin  $display("FAILED");  failedTests = failedTests + 1;  end  $display("Load (%d/%d): %0.2f", high\_count, count, 1.0 \* high\_count / count);  $display("Transition count: %d", transition\_count);  $finish;  end  endmodule |





## Explanation

* Inputs:
  + clk: Clock signal for the test bench.
  + reset: Reset signal for the test bench.
* Initialization:
  + $dumpfile and $dumpvars are used for waveform generation.
  + Three instances of the gen\_tick module are instantiated with different configurations:
    - uut\_100\_2 with SRC\_FREQ = 100 Hz and TICK\_FREQ = 2 Hz.
    - uut\_100\_5 with SRC\_FREQ = 100 Hz and TICK\_FREQ = 5 Hz.
    - uut\_100\_50 with SRC\_FREQ = 100 Hz and TICK\_FREQ = 50 Hz.
* Clock Generation:
  + The clock signal (clk) is toggled every 50 time units to simulate a 10 MHz clock.
* Test Scenarios:
  + Each test case verifies the tick generation for different configurations by counting the high ticks and transitions over 1000 clock cycles.
  + Test Case 1: SRC\_FREQ = 100 Hz, TICK\_FREQ = 2 Hz.
    - The expected number of high ticks is 500, and the number of transitions should be 20.
  + Test Case 2: SRC\_FREQ = 100 Hz, TICK\_FREQ = 5 Hz.
    - The expected number of high ticks is 1000, and the number of transitions should be 50.
  + Test Case 3: SRC\_FREQ = 100 Hz, TICK\_FREQ = 50 Hz.
    - The expected number of high ticks is 500, and the number of transitions should be 100.
* Output Monitoring:
  + The $display task is used to print the test results, indicating whether each test passed or failed based on the expected output.

# Conclusion

The gen\_tick module successfully generates a tick signal at a specified frequency derived from a source clock. The test bench verifies the functionality of the gen\_tick module by simulating various configurations and observing the output behavior. The simulation results confirm that the module operates correctly when tested with different source clock and tick frequencies.