

Design of 4-bit updown counter using nclaunch Simulator in Cadence software

Aim:

To write a verilog code for 4bit up/down counter and verify the functionality using Test bench.

Tools used for ASIC Flow:

Functional Simulation: nclaunch Simulator (nclaunch)

Design Information and Block Diagram:

- An up/down counter is a digital counter which can be set to count either from 0 to MAX_VALUE or MAX_VALUE to 0.
- The direction of the count(mode) is selected using a single bit input. The module has 3 inputs - clk, reset which is active high and a Up Or Down mode input. The output is Counter which is 4 bit in size.
- When Up mode is selected, counter counts from 0 to 15 and then again from 0 to 15.
- When Down mode is selected, counter counts from 15 to 0 and then again from 15 to 0.
- Changing mode doesn't reset the Count value to zero.
- You have to apply high value to reset, to reset the Counter output.

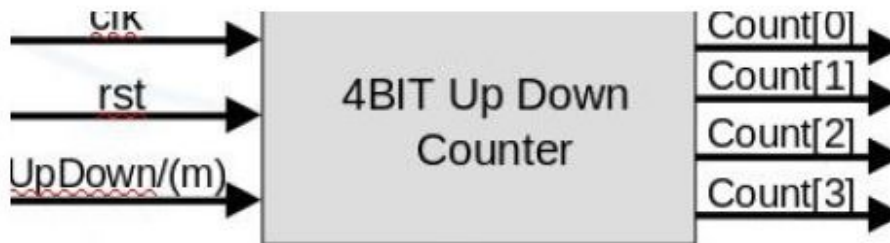
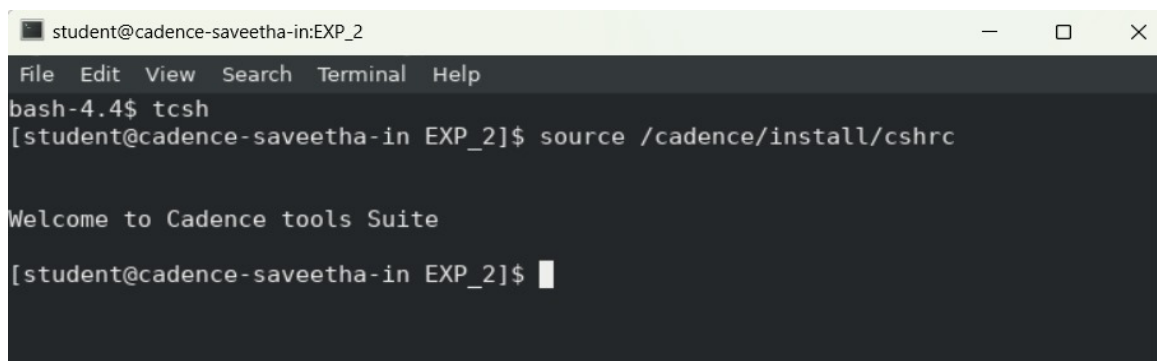


Fig 1: 4 Bit Up/Down Counter

Creating a Work space :

- Create a folder in your name (Note: Give folder name without any space) and Create a new sub-Directory name it as Exp2 or counter_design for the Design and open a terminal from the Sub-Directory. Functional Simulation:
- Invoke the cadence environment by type the below commands
- tcsh (Invokes C-Shell)
- source /cadence/install/cshrc (mention the path of the tools) (The path of cshrc could vary depending on the installation destination)
- After this you can see the window like below

A terminal window titled 'student@cadence-saveetha-in:EXP_2' with standard window controls. The terminal shows the following commands and output:

```
File Edit View Search Terminal Help
bash-4.4$ tcsh
[student@cadence-saveetha-in EXP_2]$ source /cadence/install/cshrc

Welcome to Cadence tools Suite

[student@cadence-saveetha-in EXP_2]$
```

Fig 2 : Invoke the Cadence Environment

Creating Source Code:

- In the Terminal, type `gedit .v` or `.vhd` depending on the HDL Language you are to use (ex: `4b_up_downCount.v`).
- A Blank Document opens up into which the following source code can be typed down.

(Note : File name should be with HDL Extension)

Program:

Verilog code for 4-Bit Up-Down Counter:

```
`timescale 1ns / 1 ns
module counter(clk,m,rst,count);
input clk,m,rst;
output reg [3:0] count;
always@(posedge clk or negedge rst)
begin
if (!rst)
count=0;
else if(m)
count=count+1;
else
count=count-1;
end
endmodule
```

Use Save option or Ctrl+S to save the code or click on the save option from the top most right corner and close the text file.

Creating Test bench:

Similarly, create your test bench using `gedit <filename_tb>.v` or `<filename_tb>.vhd` to open a new blank document (`4bitup_down_count_tb.v`).

Test-bench code for 4-Bit Up-Down Counter:

```
`timescale 1ns / 1ns
module counter_test;
reg clk,rst,m;
wire [3:0] count;
initial
begin
clk=0;
rst=0;#5;
rst=1;
end
initial
begin
m=1;
#160 m=0;
end

counter counter1 (clk,m,rst, count);
always #5 clk=~clk;
initial $monitor("Time=%t rst=%b clk=%b count=%b" , $time,rst,clk,count);
initial
#320 $finish;
endmodule
```

To Launch Simulation tool

- linux:/> nclaunch -new& // “-new” option is used for invoking NCVERILOG for the first time for any design
- linux:/> nclaunch& // On subsequent calls to NCVERILOG

It will invoke the nclaunch window for functional simulation we can compile,elaborate and simulate it using Multiple step



Fig 3 : Setting Multi-step simulation

- Select Multiple Step and then select “Create cds.lib File” as shown in below figure
- Click the cds.lib file and save the file by clicking on Save option

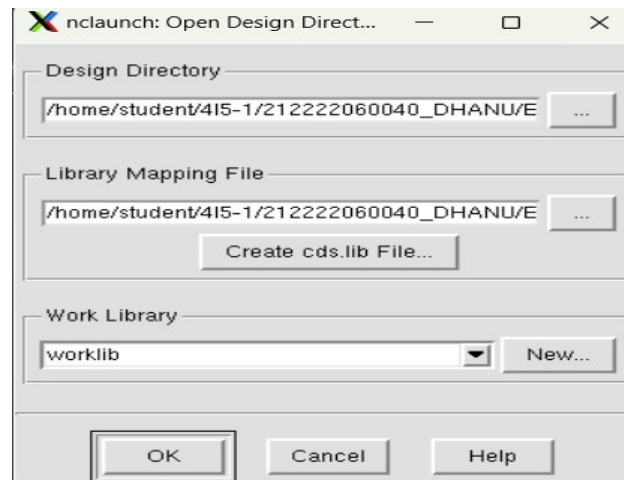


Fig 4 : cds.lib file Creation

- Save cds.lib file and select the correct option for cds.lib file format based on the HDL Language and Libraries used.
- Select “Don’t include any libraries (verilog design)” from “New cds.lib file” and click on “OK” as in below figure
- We are simulating verilog design without using any libraries

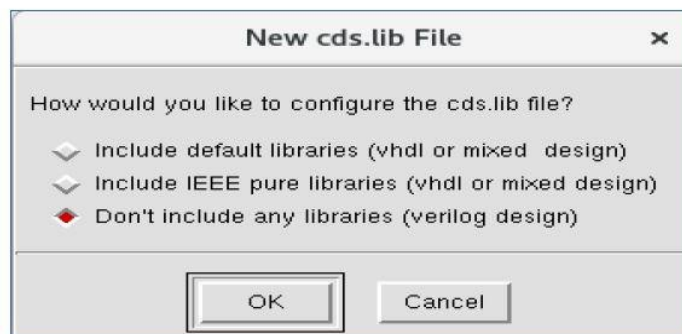


Fig 5 : Selection of Don’t include any libraries

- A Click “OK” in the “nclaunch: Open Design Directory” window
- A ‘NCLaunch window’ appears as shown in figure below
- Left side you can see the HDL files. Right side of the window has worklib and snapshots directories listed.
- Worklib is the directory where all the compiled codes are stored while Snapshot will have output of elaboration which in turn goes for simulation

To perform the function simulation, the following three steps are involved Compilation, Elaboration and Simulation.

Step 1:

Compilation: Process to check the correct Verilog language syntax and usage

- Inputs: Supplied are Verilog design and test bench codes
- Outputs: Compiled database created in mapped library if successful, generates report else error reported in log file

Steps for compilation:

1. Create work/library directory (most of the latest simulation tools creates automatically)
2. Map the work to library created (most of the latest simulation tools creates automatically)
3. Run the compile command with compile options

(i.e) Cadence IES command for compile: `ncverilog +access+rwc -compile fa.v`

-- Left side select the file and in Tools : launch verilog compiler with current selection will get enable.

Click it to compile the code

-- Worklib is the directory where all the compiled codes are stored while Snapshot will have output of elaboration which in turn goes for simulation

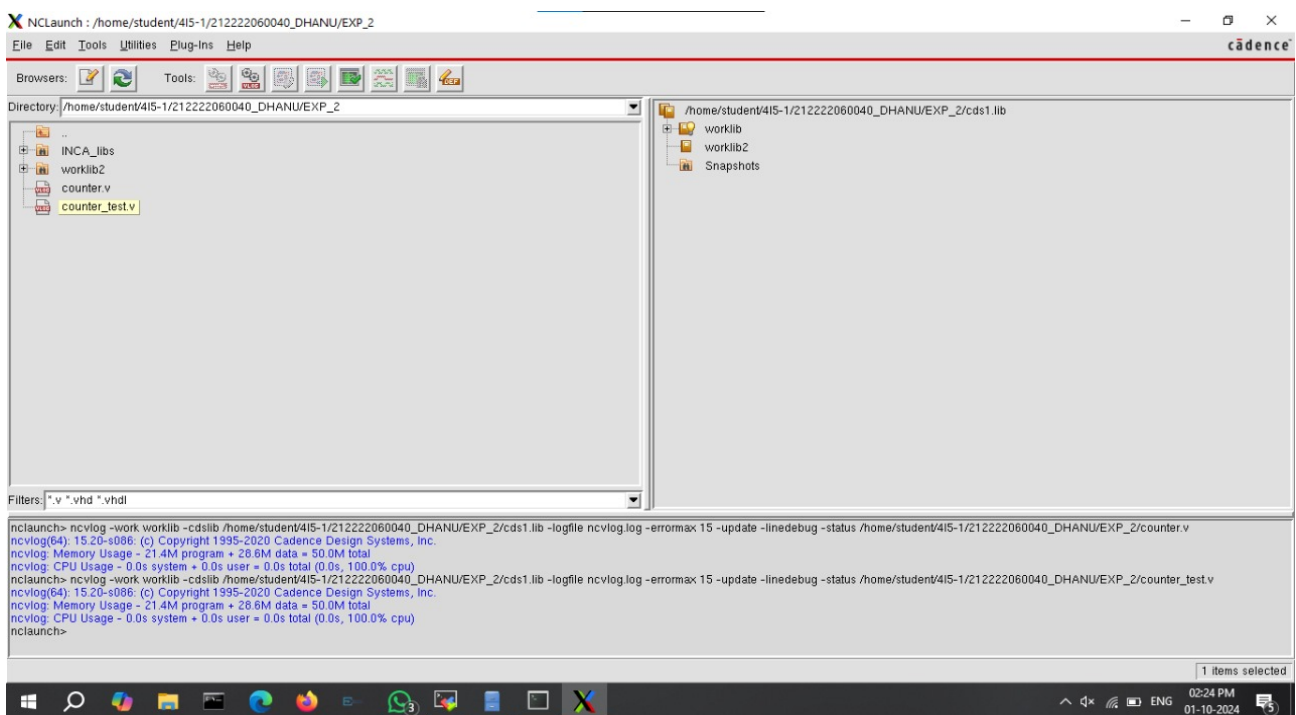


Fig 6 : Compilation database in worklib

- After compilation it will come under worklib you can see in right side window
- Select the test bench and compile it. It will come under worklib. Under Worklib you can see the module and test-bench.
- The cds.lib file is an ASCII text file. It defines which libraries are accessible and where they are located. It contains statements that map logical library names to their physical directory paths. For this Design, you will define a library called “worklib”

Step 2:

Elaboration: To check the port connections in hierarchical design

- Inputs: Top level design / test bench Verilog codes
- Outputs: Elaborate database updated in mapped library if successful, generates report else error reported in log file
- Steps for elaboration – Run the elaboration command with elaborate options
 1. It builds the module hierarchy
 2. Binds modules to module instances
 3. Computes parameter values
 4. Checks for hierarchical names conflicts
 5. It also establishes net connectivity and prepares all of this for simulation

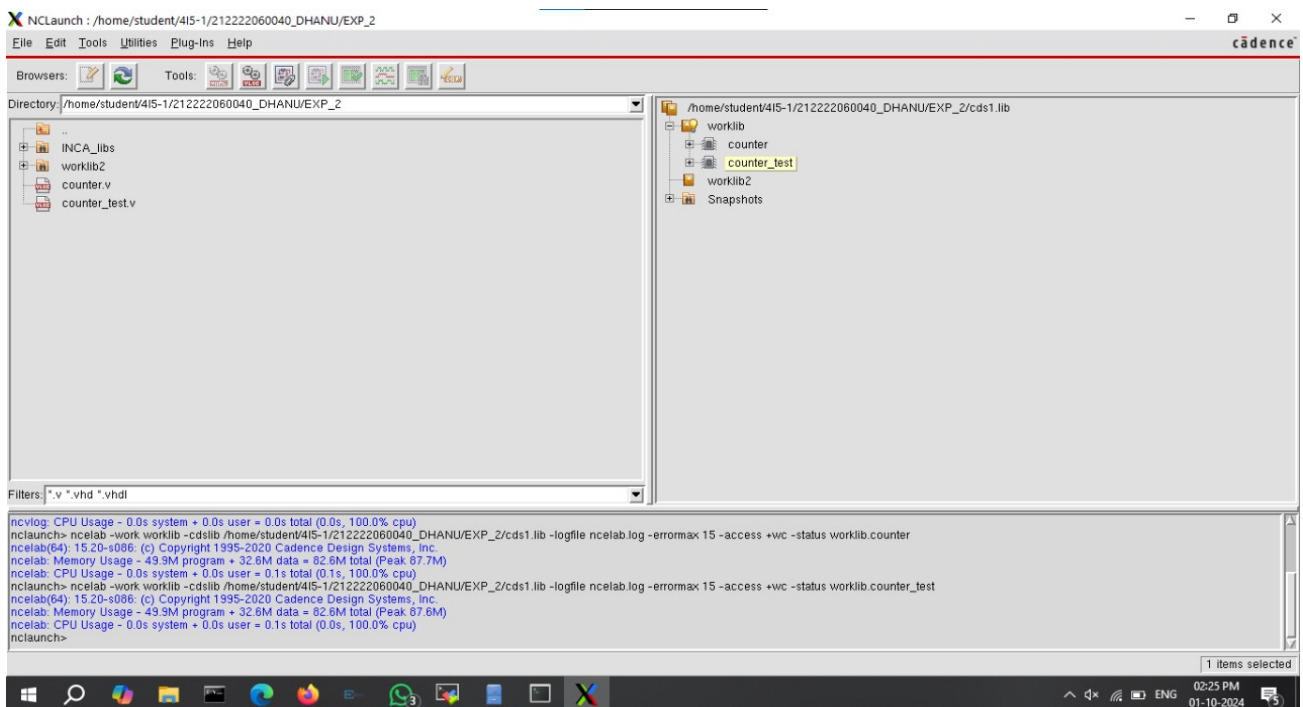


Fig 7 : Elaboration database in worklib

- After elaboration the file will come under snapshot. Select the test bench and simulate it.

Step 3:

Simulation: Simulate with the given test vectors over a period of time to observe the output behaviour.

- Inputs: Compiled and Elaborated top level module name
- Outputs: Simulation log file, waveforms for debugging
- Simulation allow to dump design and test bench signals into a waveform
- Steps for simulation – Run the simulation command with simulator options

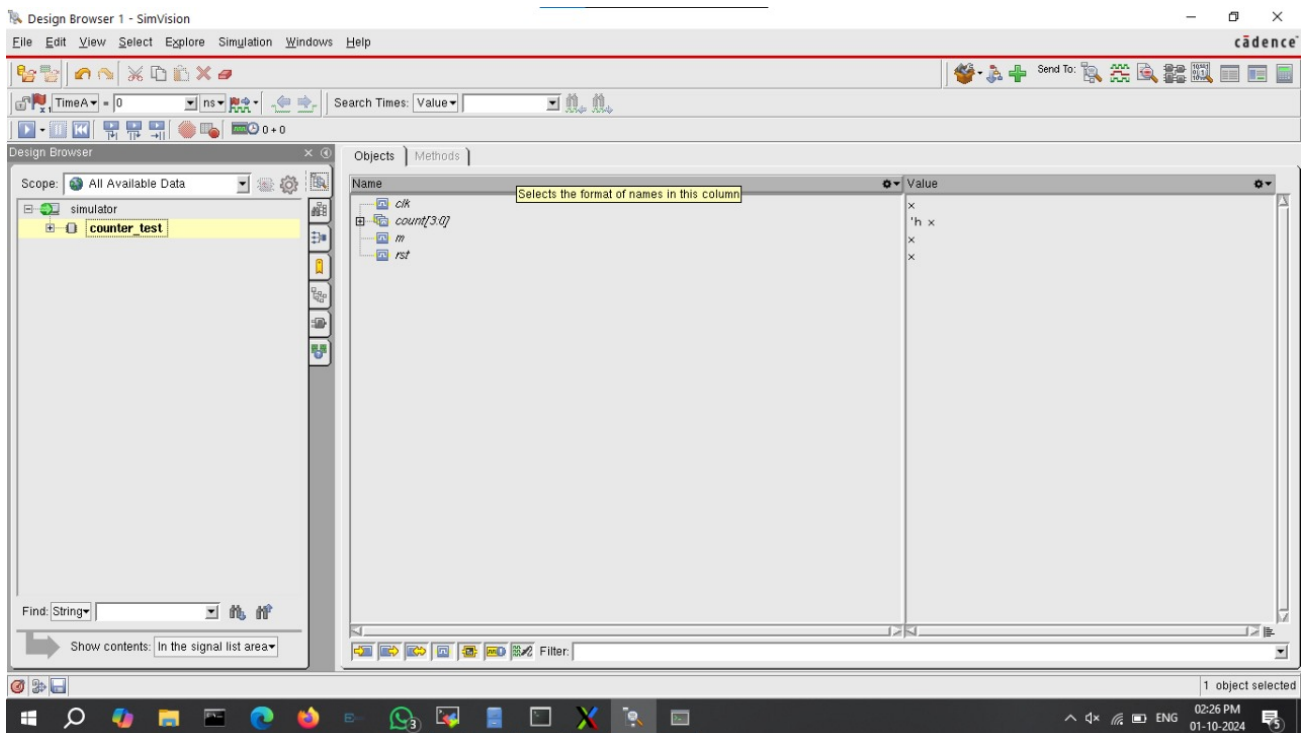


Fig 8 : Design Browser window for simulation

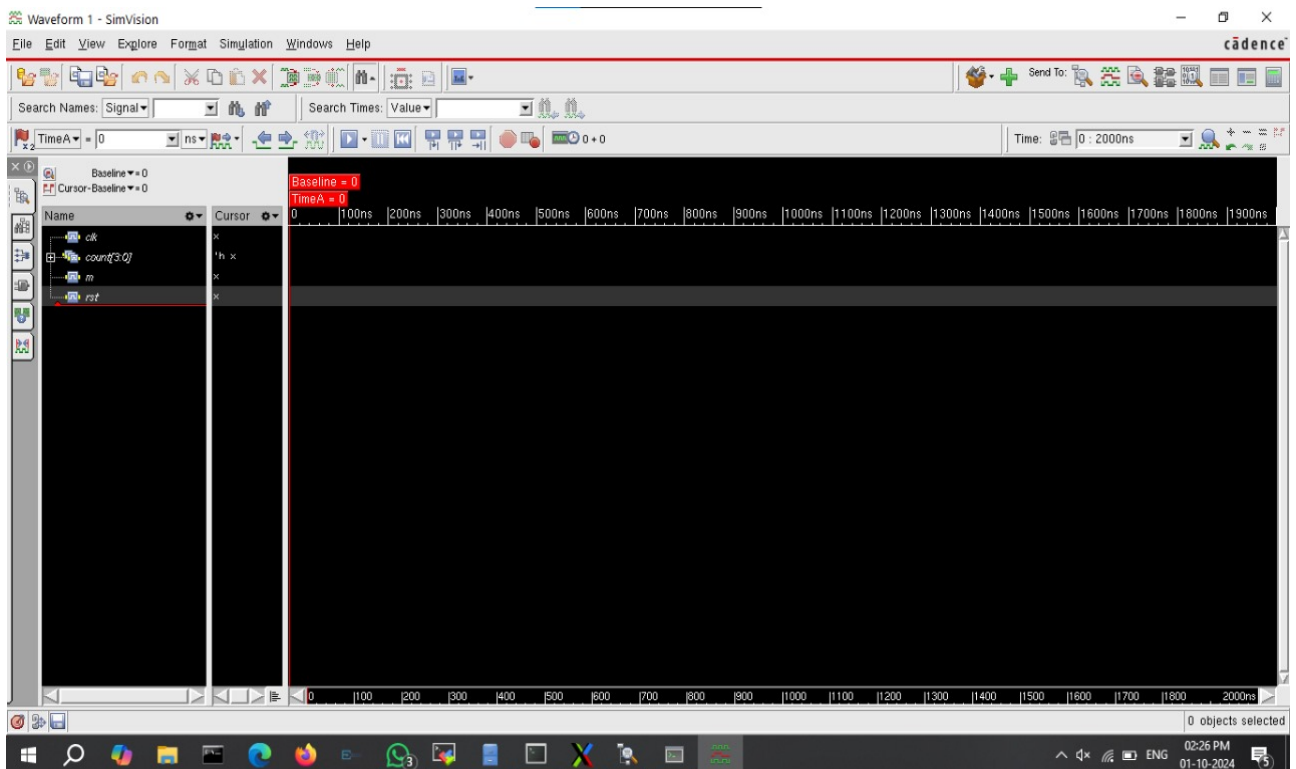


Fig 9 : Simulation Waveform Window

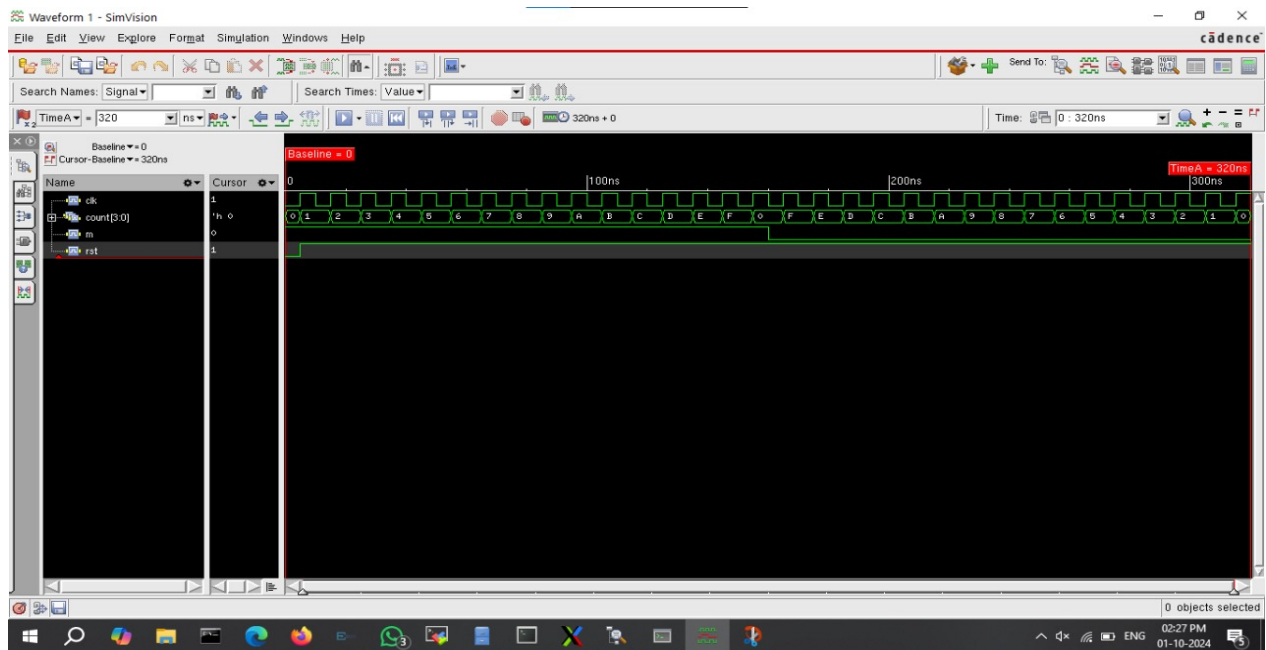


Fig 10 : Simulation Waveform Window

Github link : https://github.com/dhanu0503/4-bit_up-down_counter