Vivado Simulator

In Vivado, simulation is handled by **XSim**, the built-in simulator fully integrated into the Vivado GUI. This allows you to run HDL simulations directly within the environment, without needing to launch external tools like ModelSim. While external simulators can still be used if preferred, XSim provides a convenient and efficient solution for most simulation tasks, seamlessly fitting into the Vivado design flow.

Vivado provides two main methods for simulating VHDL designs:

1. Simulation without a Testbench (Manual Signal Forcing)

This method is useful for very simple circuits and quick demonstrations. You write your VHDL module (e.g., an AND gate), then use Vivado's built-in simulation environment to manually set input signals and observe the output directly.

This is suitable when:

- You want to test individual logic gates or small modules
- You don't need automated test cases
- You're learning basic VHDL behavior

Example: Simulating a simple AND gate by manually forcing inputs a and b, and observing the output y.

2. Simulation with a Testbench

In this method, you write a separate VHDL file (testbench) that generates input signals and checks the output. You can also write test benches the same way as you did in ModelSim no syntax change This is the recommended approach for larger systems, where testing needs to be automated and reproducible**.**

This is used when:

- Your design includes multiple components
- You want to run multiple test cases quickly
- You need to validate logic and timing in a structured way

The testbench instantiates the design under test (DUT) and applies various input combinations over time.

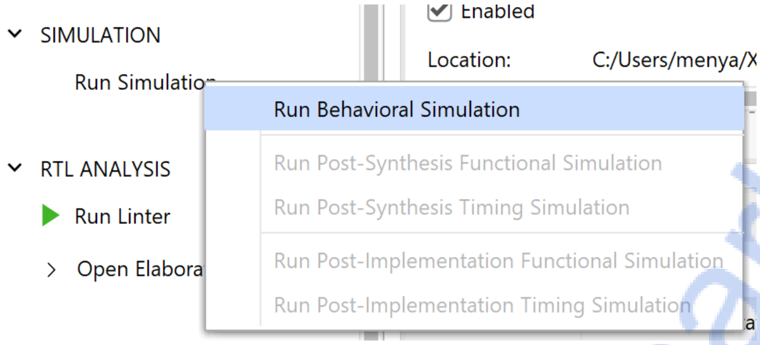
Other Simulation Tools (Optional)

Vivado comes with a built-in simulator, but it is also possible to use external simulators such as ModelSim. However, using ModelSim from within Vivado requires a separate license, and is not available by default in all Vivado installations.

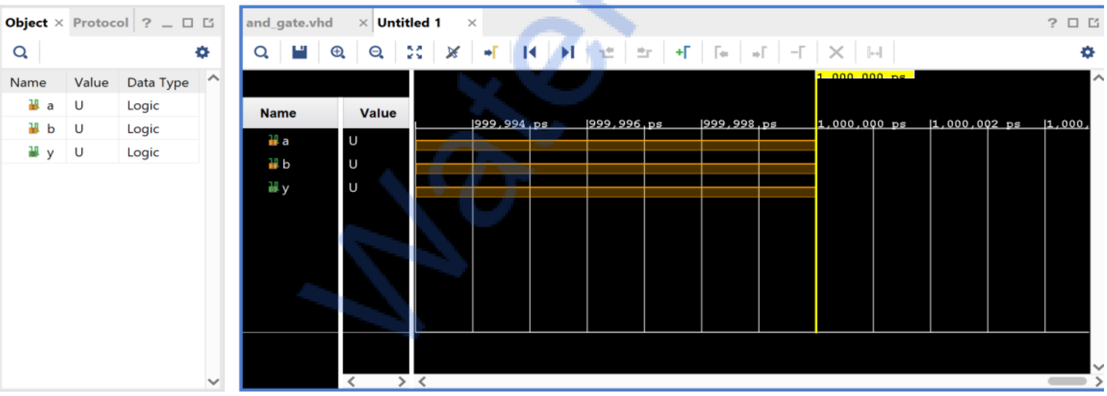
Now we will go through the two methods of simulation in Vivado.

The previous project (AND Gate) can also be used to simulate with manual constant forcing.

To start click on **Run Simulation** in the project manager menu and then **Run Behavioral Simulation**.


SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elabora
Enabled
Location: C:/Users/menya/X
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation

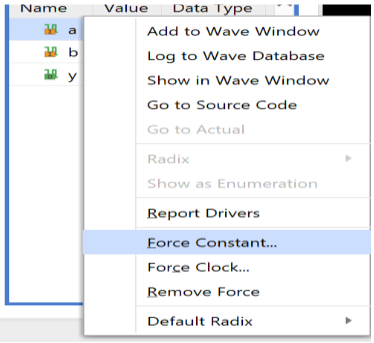
The waveform window of the simulated file now appears, showing undefined signals (U).



Name	Value	Data Type
a	U	Logic
b	U	Logic
y	U	Logic

Name	Value
a	U
b	U
y	U

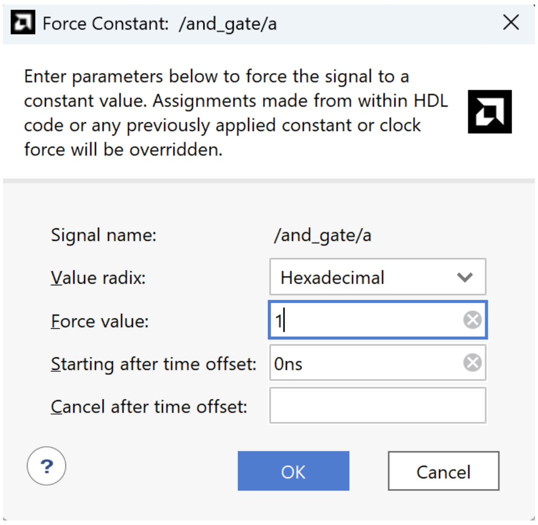
We need to define the signals, and we do that by right clicking on each signal and selecting **Force Constant and.**



Now a window will appear showing:

- **Signal name:** The signal to be forced (here: /and gate/a)
- **Value radix:** The number format (Hexadecimal, Binary, etc.)
- **Force value:** The value you want to set (1 or 0)
- **Starting after time offset:** When the value should start (e.g., 0ns)
- **Cancel after time offset:** (Optional) When to stop forcing the value

In this case for signal **(a)** the value will be **(1)**.



Force Constant: /and_gate/a

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name: /and_gate/a

Value radix: Hexadecimal

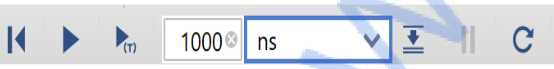
Force value: 1

Starting after time offset: 0ns

Cancel after time offset:

Click **OK** to apply.

And the same thing for the signal **(b)** because it and gate.

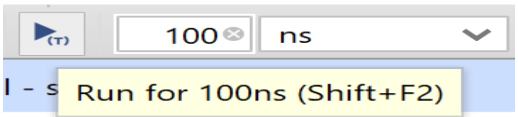


This image shows the simulation time control bar in Vivado's waveform viewer. Here's a simple explanation:

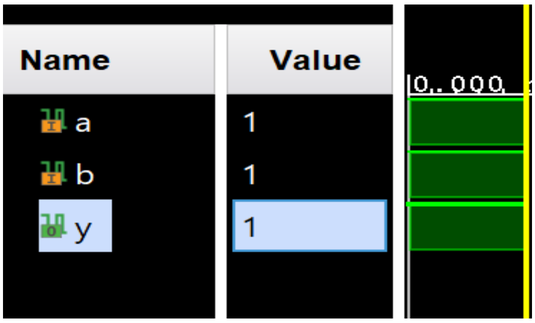
Simulation Time Control (Vivado)

- ▶ **Play buttons**: Used to run or step through the simulation.
- 🕓 **Time box (1000 ns)**: Specifies how much simulation time to run when stepping forward (here: 1000 nanoseconds), it will be 100 ns in this case .
- ⏱ **Time unit** Next to the time box here you can **change** the time unit (e.g., ns, µs, ms).
- 🔁 **Reload/Restart button**: Restarts the simulation from the beginning.

After changing the simulation time click on **Run** or **(Shift+F2)**.

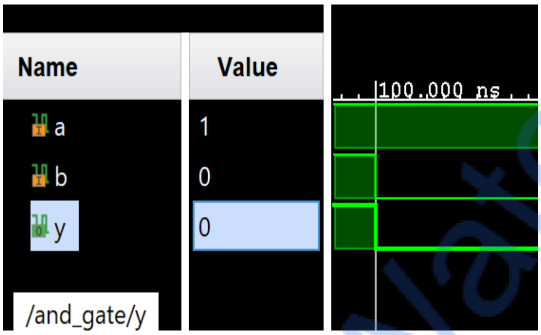


If the **a** and **b** have the value **(1)** then the output signal **y** will be **(1)** as shown in the image below.



Name	Value
a	1
b	1
y	1

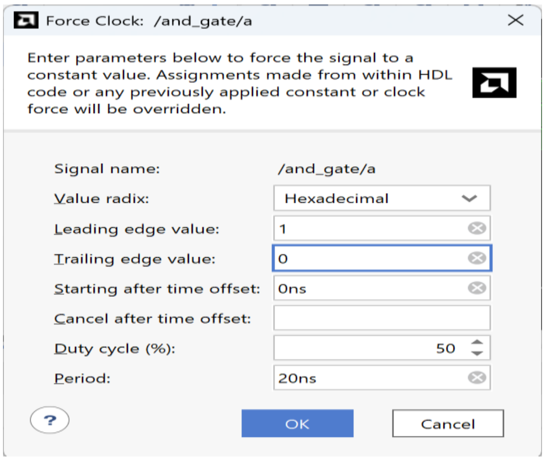
Now let's change the value of one of the inputs to (0).



Name	Value
a	1
b	0
y	0

The output **y** is currently **(0)**, which is correct.

To repeat this process of changing the value we will force the clock this time to do this choose a signal and lift click on it then select **Force Clock**.



Force Clock: /and_gate/a

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

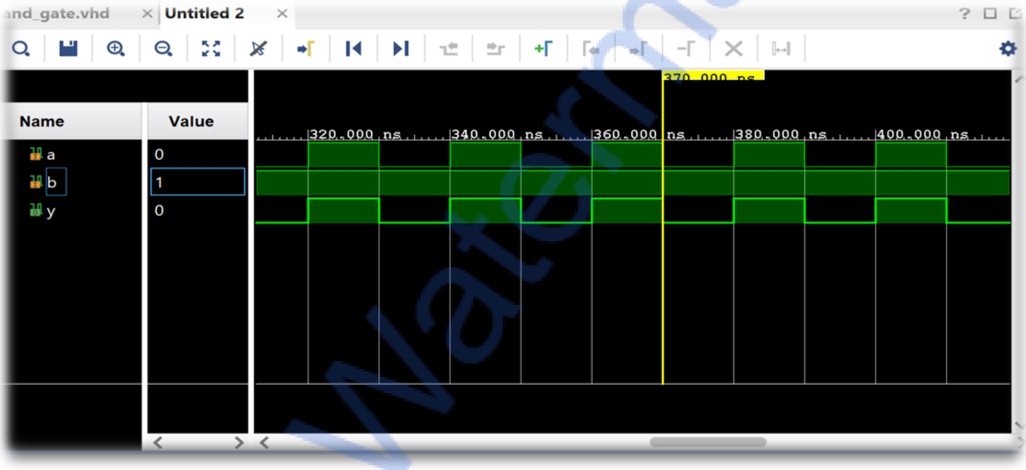
Signal name:	/and_gate/a
Value radix:	Hexadecimal
Leading edge value:	1
Trailing edge value:	0
Starting after time offset:	0ns
Cancel after time offset:	
Duty cycle (%):	50
Period:	20ns

This image shows the **"Force Clock"** dialog in a simulation environment (likely Vivado or ModelSim), where you can apply a periodic clock-like signal to a selected input signal in this case input **a**. You can change the period if you want to, I changed it to 20ns. Click when you are done.

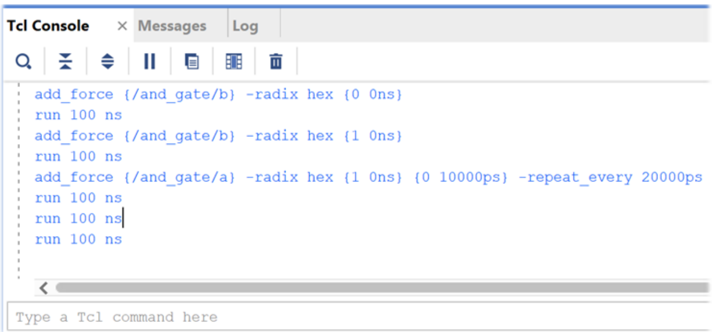
Simple explanation:

- **1** on the rising edge (start of pulse)
- **0** on the falling edge (end of pulse)
- **Start time:** 0 ns
- **Period:** 20 ns
- **Duty cycle:** 50% (equal time for 1 and 0)

Now the signal **a** will toggle between 1 and 0 every 20 ns, and the value of **y** will change accordingly as shown in the image below.



In general, every action you perform in the simulation GUI (such as forcing signals, running the simulation, or setting waveforms) automatically translated into **Tcl commands** behind the scenes. These commands appear in the **Tcl Console**.



```
add_force {/and_gate/b} -radix hex {0 0ns}
run 100 ns
add_force {/and_gate/b} -radix hex {1 0ns}
run 100 ns
add_force {/and_gate/a} -radix hex {1 0ns} {0 10000ps} -repeat_every 20000ps
run 100 ns
run 100 ns
run 100 ns
```

Type a Tcl command here

What does this mean?

- add_force: Sets a value to a signal.
- {1 0ns} : The signal is 1 at time 0 ns.
- {0 10000ps} : The signal becomes 0 at 10 ns.
- -repeat_every 20000ps: This pattern (1 → 0) repeats every 20 ns (i.e. a clock).
- run 100 ns: Run the simulation for 100 ns.

Why is this useful?

- You can **run simulations using only commands**, without using the GUI.
- It's great for **automation** or **repeating tests**.
- You can **copy commands** from the transcript and reuse them (I'll show you how in the demo video).

Even if you prefer using the GUI, always check the **Tcl transcript**. It shows you exactly what's happening behind the scenes and gives you full control when you need it.

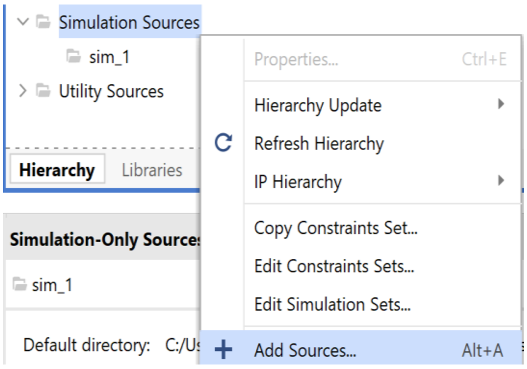
(More on Tcl will be in a separate document).

Now we are done with the first simulation method. Click on the link below to watch the first demo video for simulation in Vivado.

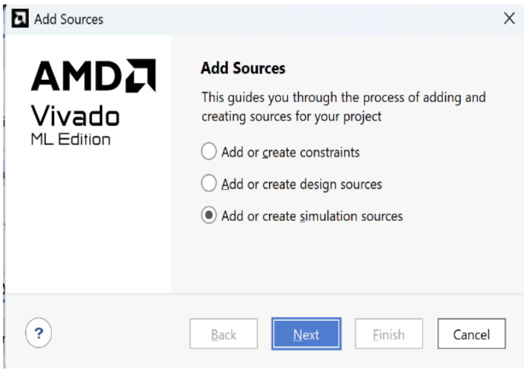
https://www.youtube.com/watch?v=99KIT_HBI74

Simulation with a Testbench

Right click on **Simulation Sources** then **Add Sources** To add or create a testbench to your design**.**



Now you choose **Add or create simulation sources** and click **Next.**



Add Sources

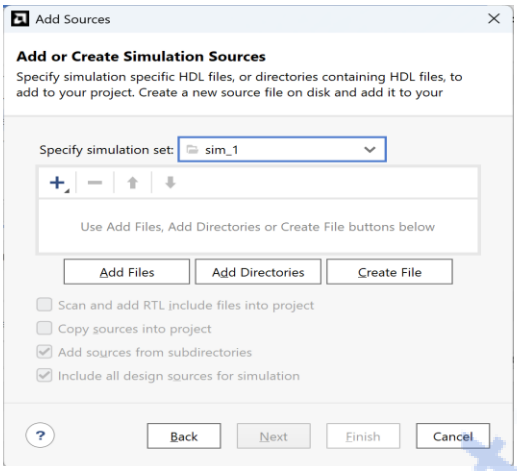
This guides you through the process of adding and creating sources for your project

○ Add or create constraints

○ Add or create design sources

● Add or create simulation sources

If you already have a testbench you want to use, click on **Add Files.**



Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your

Specify simulation set: sim_1

Use Add Files, Add Directories or Create File buttons below

□ Scan and add RTL include files into project

□ Copy sources into project

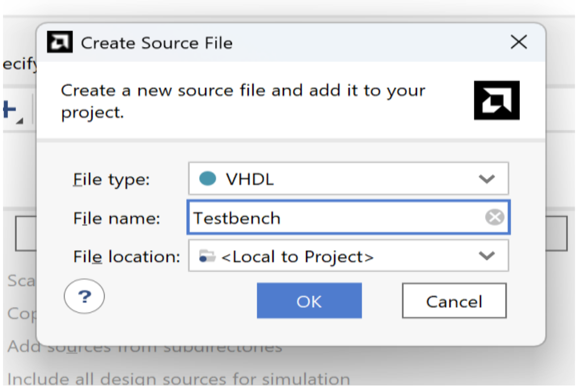
☑ Add sources from subdirectories

☑ Include all design sources for simulation

Specify simulation set: The currently active simulation fileset is the default fileset to add simulation files to. However, you can also select another existing sim_set or create a new simulation fileset for the project. Selecting Create Simulation Set will open the Create Simulation Set dialog box letting you specify the name for the new simulation fileset.

In this case we will not add or change the simulation set.

Click on **Create File** to create a testbench.



Create a new source file and add it to your project.

File type: VHDL

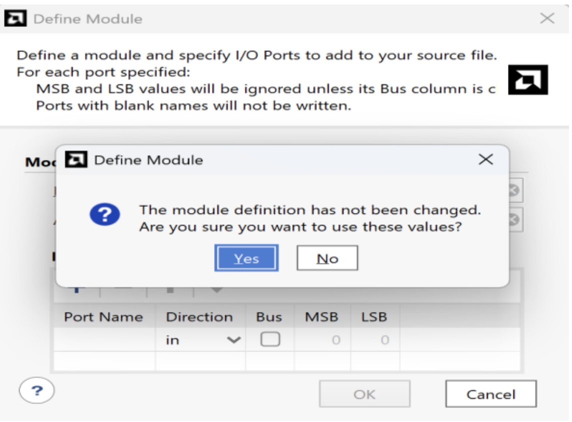
File name: Testbench

File location: <Local to Project>

?
OK
Cancel
Add sources from subdirectories
Include all design sources for simulation

Change the language and give a name for the file, click on **OK** and **Finish.**

The same window that appears when creating any file will now appear to help you create a ready-made template. However, this time you will skip it because no ports need to be defined in a testbench.



Define Module

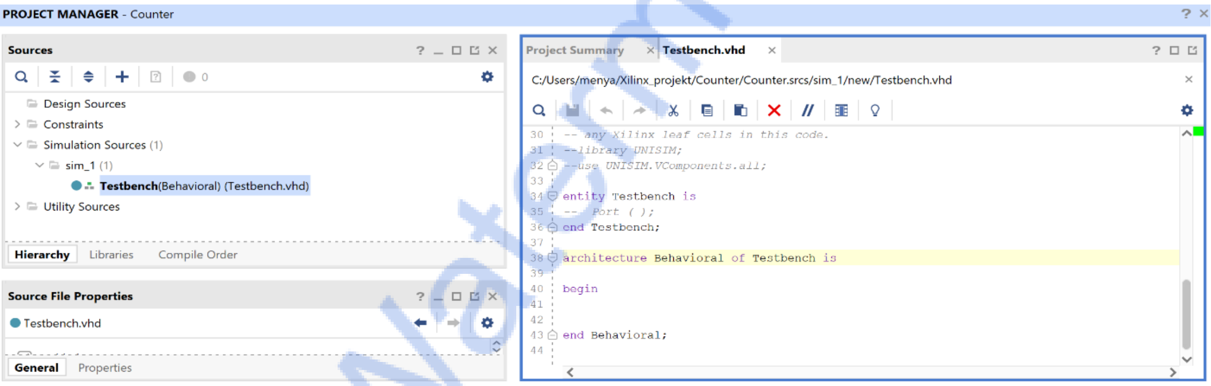
Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is c
Ports with blank names will not be written.

Define Module

The module definition has not been changed.
Are you sure you want to use these values?

Click on **OK** and **Yes.**

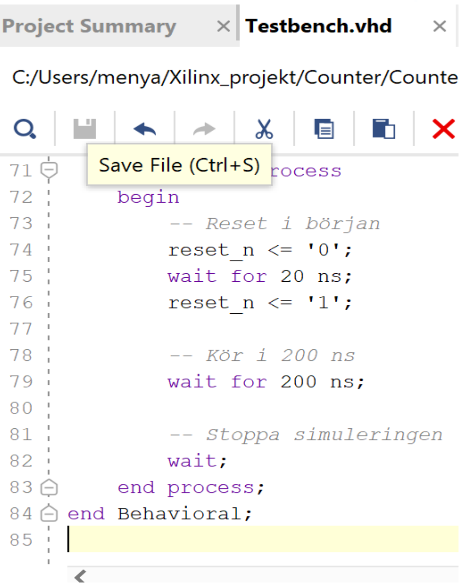
Now your empty testbench has been added to the simulation sources and you can start writing your test cases in it.



C:/Users/menya/Xilinx_projekt/Counter/Counter.srcs/sim_1/new/Testbench.vhd

```
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Testbench is
--  Port ( );
end Testbench;

architecture Behavioral of Testbench is

begin


end Behavioral;

```

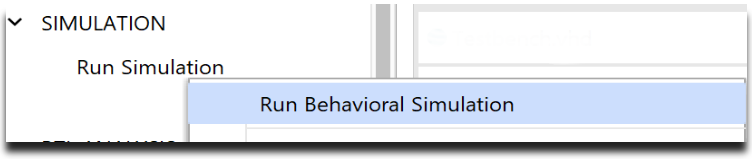
Click on **Save** when you are done



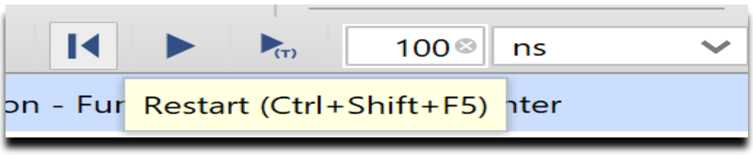
C:/Users/menya/Xilinx_projekt/Counter/Counte

```
                           rocess
    begin
        -- Reset i början
        reset_n <= '0';
        wait for 20 ns;
        reset_n <= '1';

        -- Kör i 200 ns
        wait for 200 ns;

        -- Stoppa simuleringen
        wait;
    end process;
end Behavioral;

```

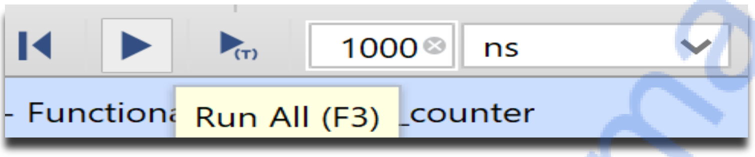
Everything is ready now to simulate the design, click on **Run Simulation** and **Run Behavioral Simulation**.



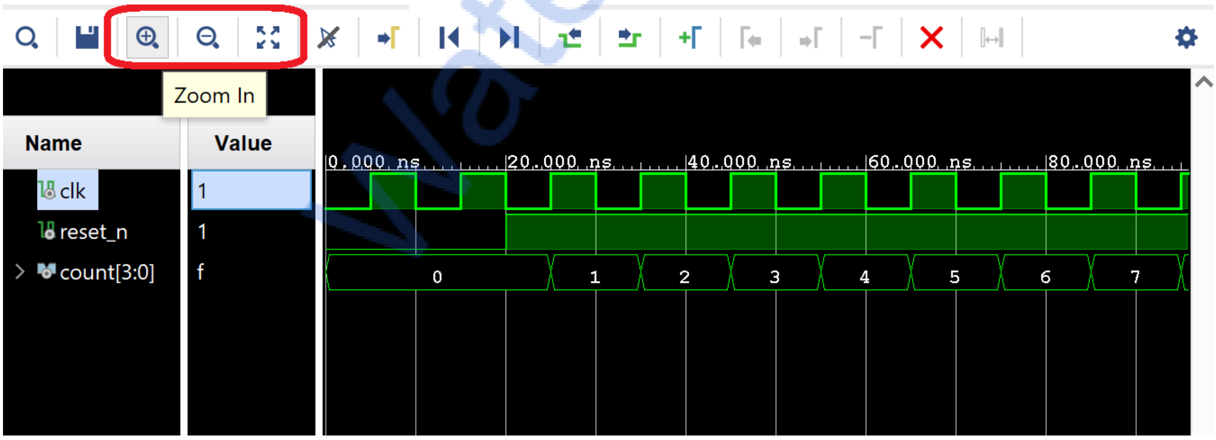
Click on the Restart icon and change the simulation time if you want to.



Click on the **Start** button to run the simulation.



In the toolbar of the Wave window, you can zoom in and out as you like.



Click on the link below to watch the demo video for this part.

https://www.youtube.com/watch?v=1Ur5L09UKSU&t=2s

Congratulations! Now you can simulate in Vivado using two methods: Manual Signal Forcing and Simulation with testbench.