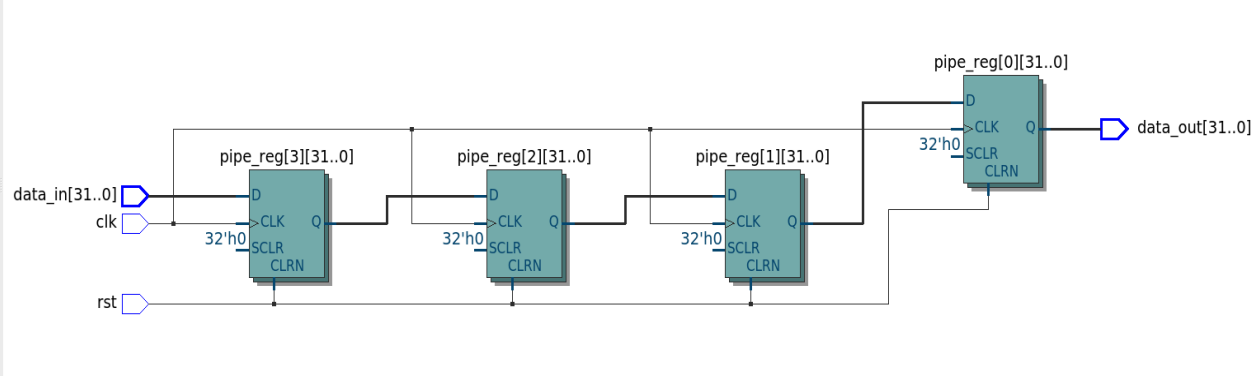
“1. Multiple Modules in a Single Testbench”

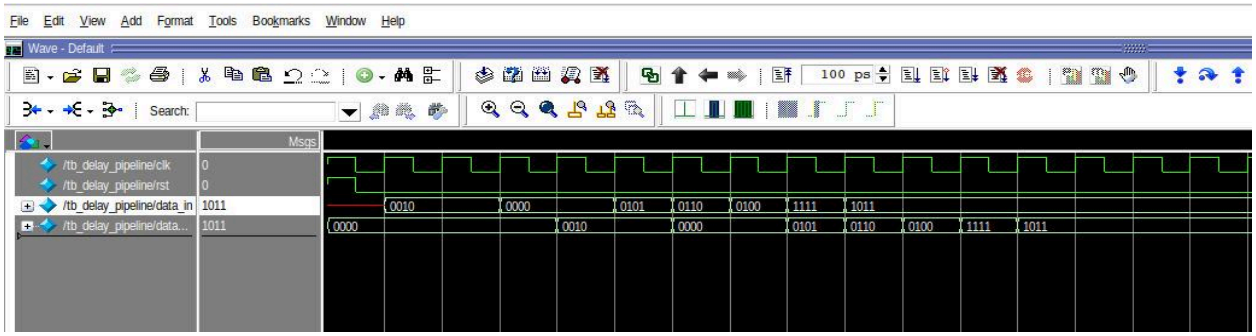
1. *N-Cycles Delay Pipeline:*

Design:



- The module implements an N-stage delay pipeline that shifts 32-bit input data across clock cycles.
- The user inputs the parameter N in order to delay the input for N cycles, so that output appears after certain delay.
- A synchronous reset clears all registers, and the output is taken from the first stage to achieve the intended delay.

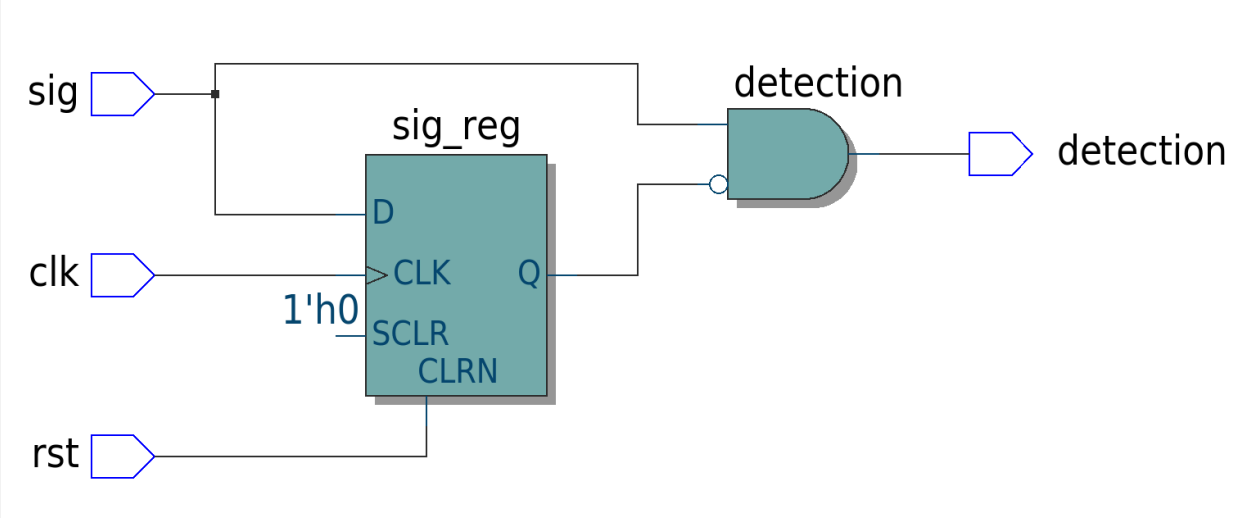
Verification:



- The testbench instantiates the “delay_pipeline” with four stages (delay = 4) and connects clock, reset, and data signals.
- A clock generator runs with a 10 ns period, while reset is applied briefly to clear the pipeline.
- Random values are driven into data_in using $urandom to test the pipeline with varied inputs, tb runs and simulation ends with $stop.

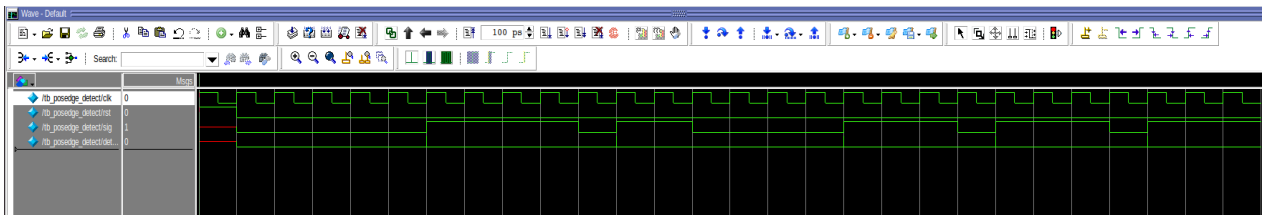
2. *Posedge Detector:*

Design:



- The module identifies rising edges of a 1-bit input signal and generates a single-cycle output pulse.
- A register stores the previous signal value, enabling comparison with the current input on each clock edge.
- Reset clears the register to zero, for proper initialization before edge detection starts.

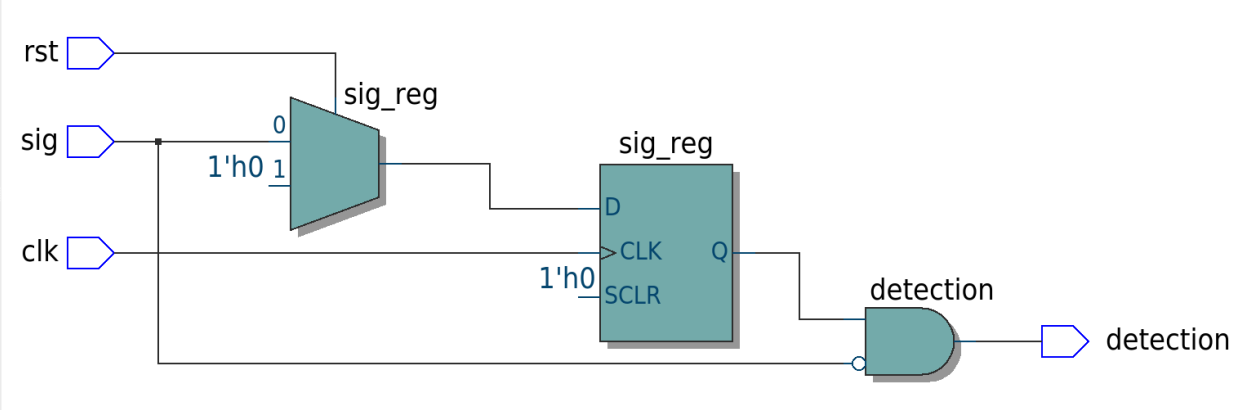
Verification:



- The testbench instantiates the posedge detect module and connects clock, reset, input signal, and output.
- A clock generator produces a 10 ns period waveform, while reset is asserted at the start to initialize the design.
- Random binary values are applied to the input signal using $urandom_range to create varied edge conditions, and runs till $stop.

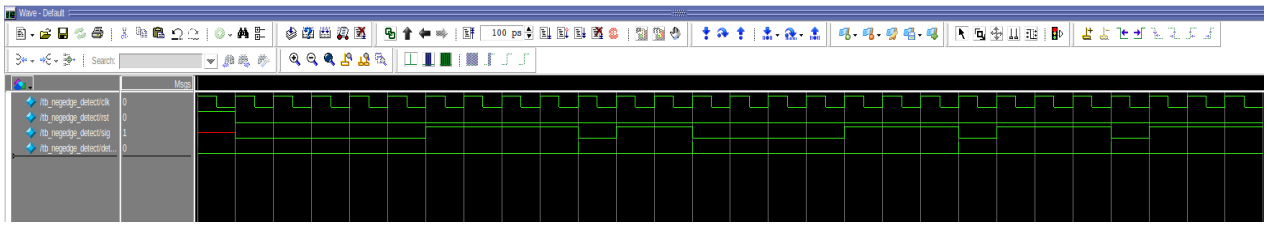
3. Negedge Detector:

Design:



- The module detects falling edges of a 1-bit input signal and generates a single-cycle output pulse.
- A register holds the previous signal value, enabling comparison with the current input on each clock edge.
- Reset clears the register to zero, for proper initialization before falling edge detection starts.

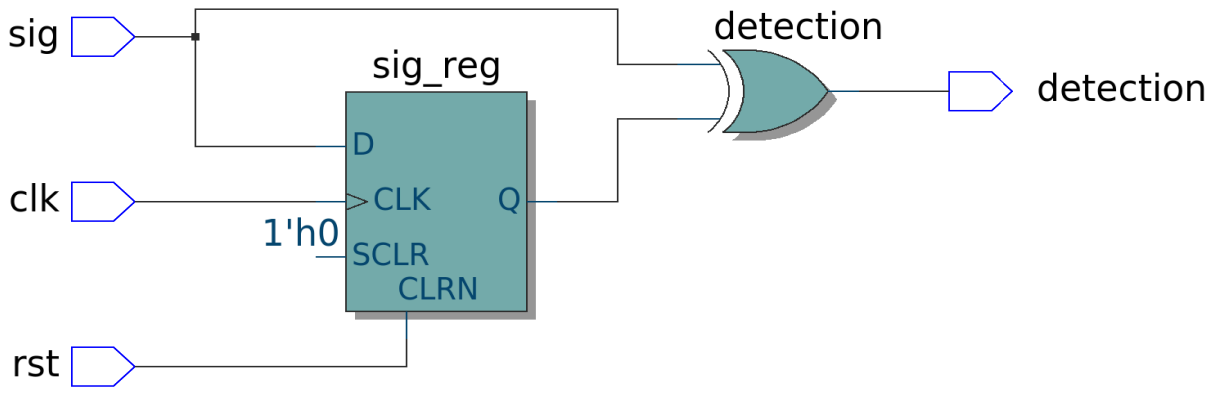
Verification:



- The testbench instantiates the negedge detect module and connects clock, reset, input signal, and output.
- A clock generator produces a 10 ns period waveform, while reset is asserted at the start to initialize the design.
- Random binary values are applied to the input signal using $urandom_range to create falling edge conditions, simulation continues until $stop.

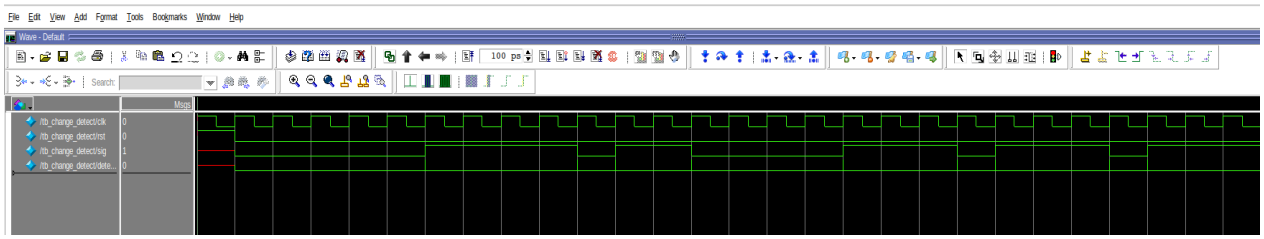
4. Posedge-Negedge Detector:

Design:



- The module detects any change in the input signal and generates a single cycle output pulse.
- A register stores the previous value of the signal on each clock edge for comparison.
- The detection logic uses an XOR operation between the stored and current values to identify transitions.
- A reset input clears the register to zero forproper initialization before change detection begins.

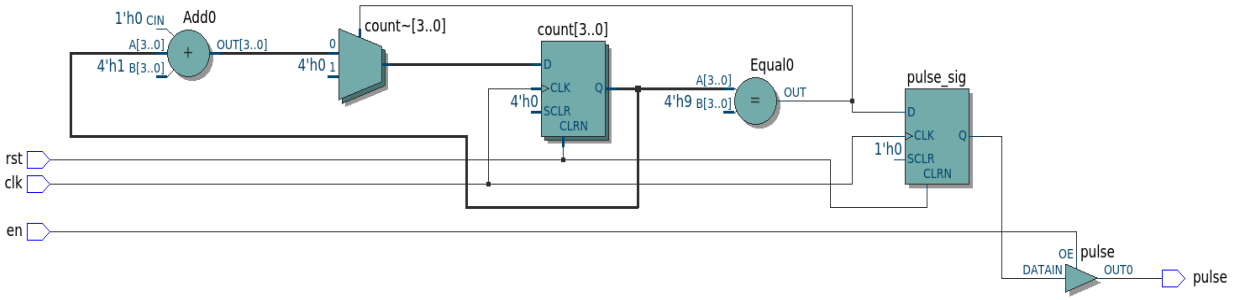
Verification:



- The testbench instantiates the change detect module and connects clock, reset, input signal, and output.
- A clock generator produces a 10 ns period waveform, while reset is applied at the start to initialize the design.
- Random binary values are driven into the input signal using $urandom_range to create change conditions and continue till $stop;

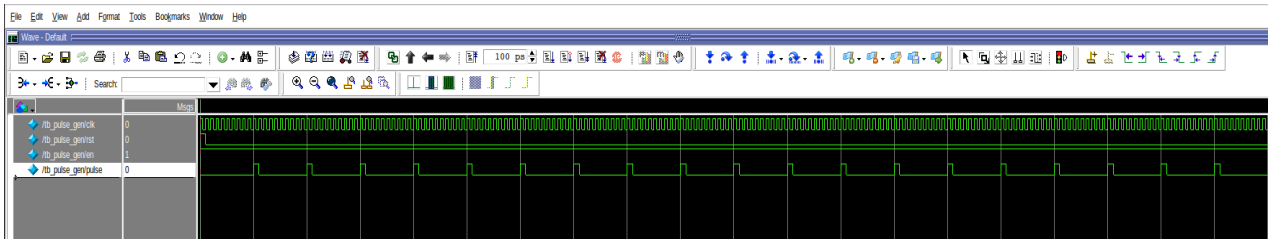
5. Pulse Generator:

Design:



- The module generates a single pulse every 100 ns by counting clock cycles and resetting after ten counts.
- A counter and control signal are used, with the pulse asserted for one cycle when the count reaches nine.
- The output is enabled through the en input, otherwise it remains in high impedance, and reset clears all states.

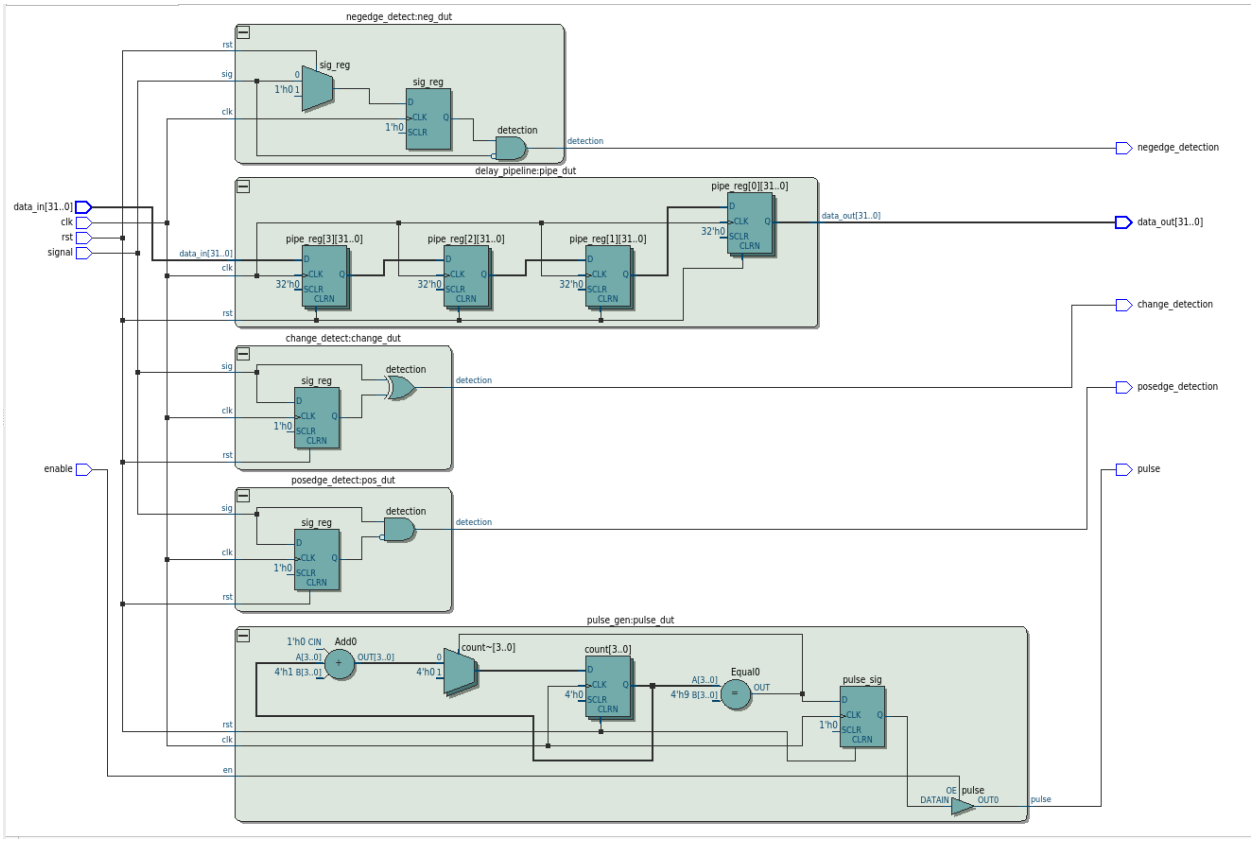
Verification:



- The testbench instantiates the pulse generator module and connects clock, reset, enable, and output signals.
- A clock generator produces a 10 ns period waveform, while reset is applied at the start to initialize the design.
- The enable signal is set high and the simulation runs for 200 ns, allowing observation of periodic output pulses.

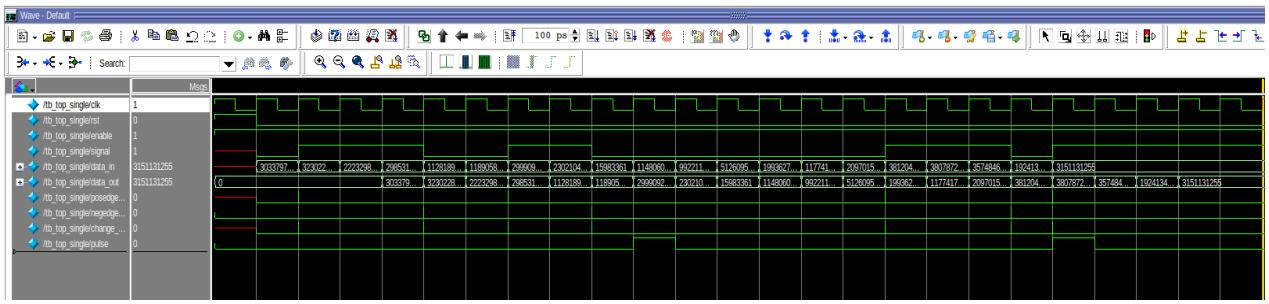
6. Top Level Unified Design:

Design:



- The top module integrates all submodules into a single design, connecting clock, reset, and inputs.
- It combines edge detection, change detection, pulse generation, and data delay functions in one place.
- Outputs from each submodule are exposed, providing a unified interface for signal monitoring and data processing.

Verification:



- The testbench instantiates the top module and connects all signals including edge detectors, pulse generator, and pipeline.
- Random stimulus is applied to signal and data inputs while results are stored in a queue for pipeline checking.
- Tasks verify unit outputs and delayed data on each clock edge, printing pass or fail messages during simulation.

```
# Posedge detected: PASS
# Change detected: PASS
# Negedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Posedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Negedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Pulse detected: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Posedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Delay Pipeline: PASS
# Negedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Posedge detected: PASS
# Change detected: PASS
# Delay Pipeline: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
# Pulse detected: PASS
```