Contents

Lab 2

Description

This VHDL code defines a digital system entity named "top" with configurable parameters and several processes to handle signal manipulation and detection logic. Here is a detailed description of the code.

Entity Declaration

- **Entity Name**: top
- **Generic Parameters**:
  - n: A positive integer (default 8), defining the bit-width of the signal x.
  - m: A positive integer (default 7), used for detection logic.
  - k: A positive integer (default 3), another parameter (not used in the provided code).
- **Ports**:
  - rst, ena, clk: Input signals of type std_logic.
  - x: An input signal of type std_logic_vector with a width of n bits.
  - DetectionCode: An input signal of type integer ranging from 0 to 3.
  - detector: An output signal of type std_logic.

Architecture

- **Architecture Name**: arc_sys
- **Internal Signals**:
  - x_int: A signal to store the input x.
  - out1_int, out2_int: Signals to store previous values of x.
  - valid_int: A signal indicating the validity of a detection.
  - adder_in: A signal used for addition operations.
  - sum_2_DetCode: A signal to store the sum of out2_int and adder_in.

- carry: A signal for the carry-out of the addition.

Processes

process 1:

Purpose: Process 1 is responsible for capturing and storing the current and previous values of the input signal x on the rising edge of the clock (clk), provided that the enable signal (ena) is active. It also handles reset (rst) conditions.

Signals and Variables:

- x_int: Captures the input x signal.
- out1_int, out2_int: Signals to store previous values of x.
- prev_x, prev_prev_x: Variables used within the process to temporarily hold previous values of x.

Behavior:

1. **Initialization**:
    - The signal x_int is assigned the value of x.
2. **Sensitivity List**:
    - The process is sensitive to changes in clk, rst, ena, and x_int.
3. **Reset Condition**:
    - If rst is high (rst = '1'):
        - out1_int and out2_int are set to zero (all bits are '0').
        - prev_x and prev_prev_x are also set to zero.
4. **Clock Edge Detection**:
    - If a rising edge of clk is detected (rising_edge(clk)):
        - **Enable Check**:
            - If ena is high (ena = '1'):
                - prev_prev_x is updated with the value of prev_x.

        - prev_x is updated with the current value of x_int.
        - out2_int is updated with prev_prev_x.
        - out1_int is updated with prev_x.
    - If ena is not high (ena = '0'):
        - out1_int and out2_int are set to zero.

Process 2

Purpose: This process is responsible for converting the DetectionCode input, which is an integer ranging from 0 to 3, into a std_logic_vector that will be used for addition operations in the system.

Behavior:

- **Sensitivity List**:
    - The process is sensitive to changes in the DetectionCode signal.
- **Operation**:
    - It uses a case statement to handle different values of DetectionCode.
    - Depending on the value of DetectionCode, it sets the adder_in signal to a specific std_logic_vector value.
        - When DetectionCode = 0, adder_in is set to a vector with all zeros except the least significant bit (LSB) which is '1'.
        - When DetectionCode = 1, adder_in is set to a vector with all zeros except the two least significant bits which are "10".
        - When DetectionCode = 2, adder_in is set to a vector with all zeros except the two least significant bits which are "11".
        - When DetectionCode = 3, adder_in is set to a vector with all zeros except the three least significant bits which are "100".
        - For any other value, adder_in is set to a vector of all zeros.

Process 3

Purpose: This process handles the final detection logic based on the valid_int signal. It counts the number of consecutive cycles where valid_int is '1' and sets the detector output signal accordingly.

Behavior:

- **Variables**:
    - counter: An integer variable used to count the number of consecutive valid cycles.
- **Sensitivity List**:
    - The process is sensitive to changes in clk and rst.
- **Reset Condition**:
    - If rst is high (rst = '1'):
        - detector is set to '0'.
        - counter is reset to 0.
- **Clock Edge Detection**:
    - If a rising edge of clk is detected (rising_edge(clk)):
        - **Enable Check**:
            - If ena is high (ena = '1'):
                - If valid_int is '1':
                    - Increment the counter by 1.
                - If valid_int is '0':
                    - Reset the counter to 0.
                - **Detector Output Logic**:
                    - If counter is greater than or equal to m:
                        - Set detector to '1'.
                    - If counter is less than m:
                        - Set detector to '0'.

        - If ena is not high (ena = '0'):
            - Set detector to '0'.
            - Reset the counter to 0.

Reset Condition

- Resets the detector signal to '0' and the counter to 0 when rst is high.

Clock Edge Handling:

- On the rising edge of clk and if ena is high:
    - If valid_int is '1', increment the counter.
    - If valid_int is '0', reset the counter.
    - Set detector to '1' if the counter is greater than or equal to m.
    - Set detector to '0' if the counter is less than m.

Enable Check:

- If ena is not high, reset detector to '0' and the counter to 0.

Testing

Stimulus Processes:

1. **Clock Generation** (gen_clk): A process that toggles the clock signal every 50 ns to create a 100 ns period clock.
2. **Input Signal Generation** (gen_x): A process that initializes x to zero and DetectionCode to a variable j. It then updates x in a loop every 100 ns by adding j + 1 to the current value of x.
3. **Reset Signal Generation** (gen_rst): A process that sets the reset signal (rst) high initially and then low after 1500 ns.
4. **Enable Signal Generation** (gen_ena): A process that sets the enable signal (ena) low initially and then high after 2000 ns.

Purpose:

The testbench simulates the top entity by providing clock, reset, enable, and input signals, and observes the detector output to verify the behavior and functionality of the top entity under various conditions.

This setup helps in verifying that the top entity works correctly by simulating its operation over time and checking the outputs against expected results.

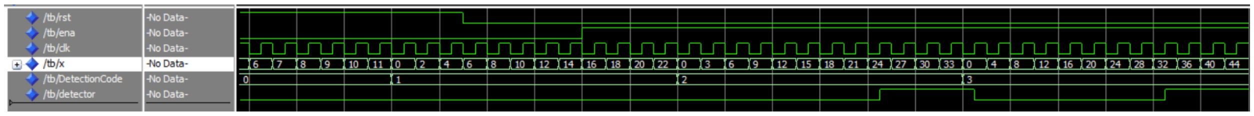

Figure 1 - screenshot of the testbench