8-bit Up/Down Counter Report

Bonus Assignment

CP319

Talei Ibhanesebhor, 169020238

December 20th, 2024

Introduction

This report details the design and implementation of an 8-bit Up/Down Counter using VHDL and Falstad Circuit Simulator. The counter performs increment and decrement operations based on a control signal (UP/DOWN switch). The VHDL version was simulated using EDA Playground, while the Falstad simulation provides a visual logic-based implementation for educational purposes. The project emphasizes synchronous clocking, modular design, and behavioral modeling in VHDL, alongside a practical circuit design in Falstad.

Design Process

1. VHDL Implementation

Modules in VHDL:

- **Main Counter Module:** Implements the counting logic.
- **Testbench Module:** Simulates input signals and validates functionality.

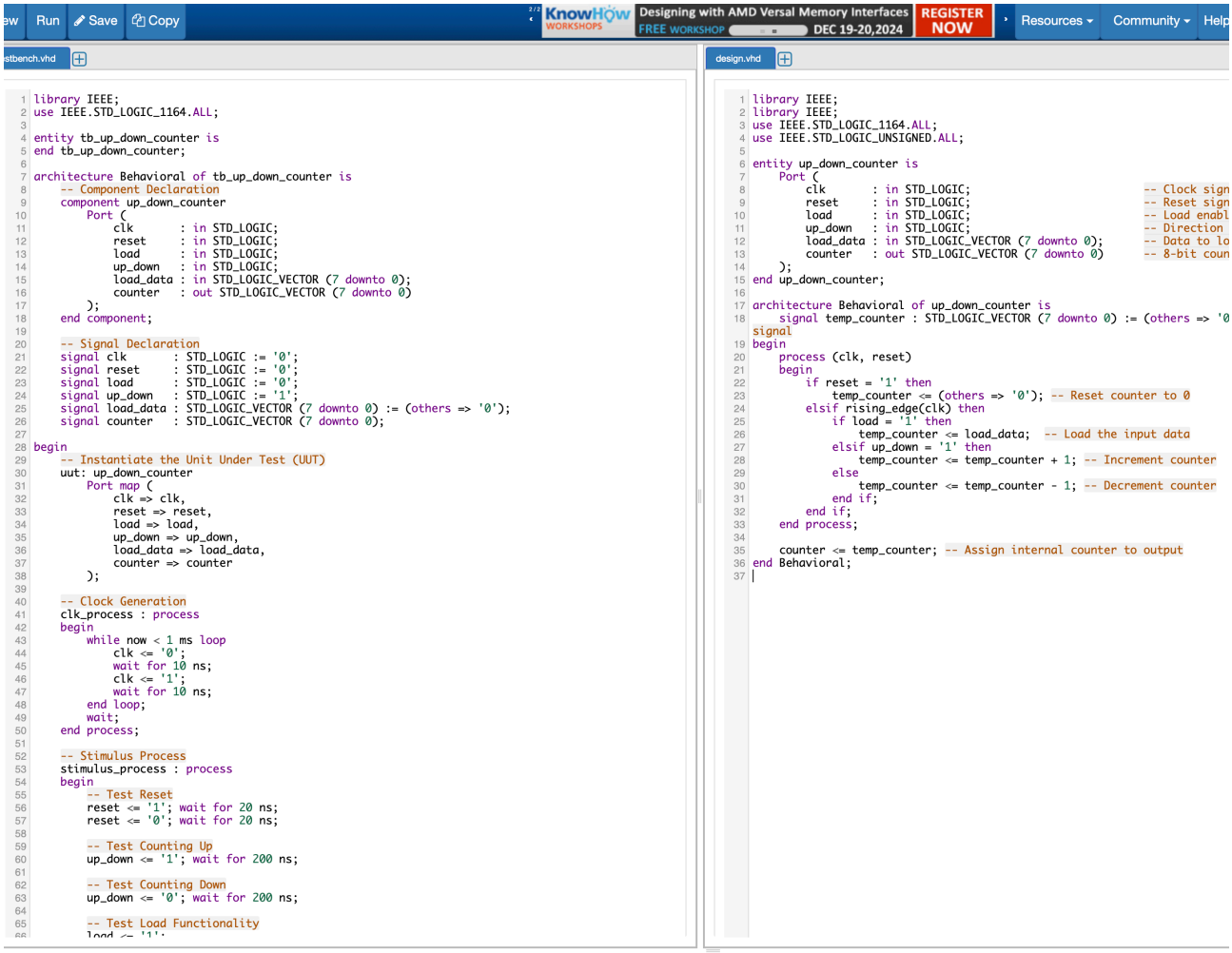
Logic Description:

- **UP Mode:** When the UP/DOWN signal is high (1), the counter increments.
- **DOWN Mode:** When the UP/DOWN signal is low (0), the counter decrements.
- **Clock Signal:** Drives all flip-flops synchronously.

Key Features:

- Synchronous design using a single clock source.
- Behavioral VHDL for describing counter functionality.
- Extensibility for counters larger than 8 bits.

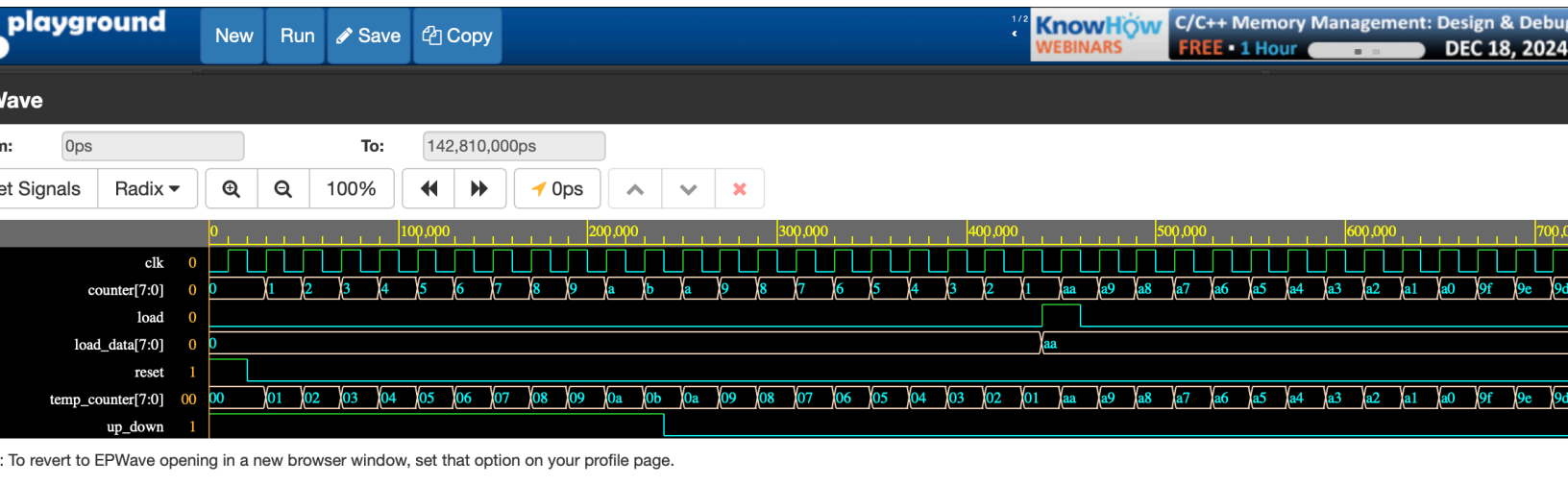
VHDL Code



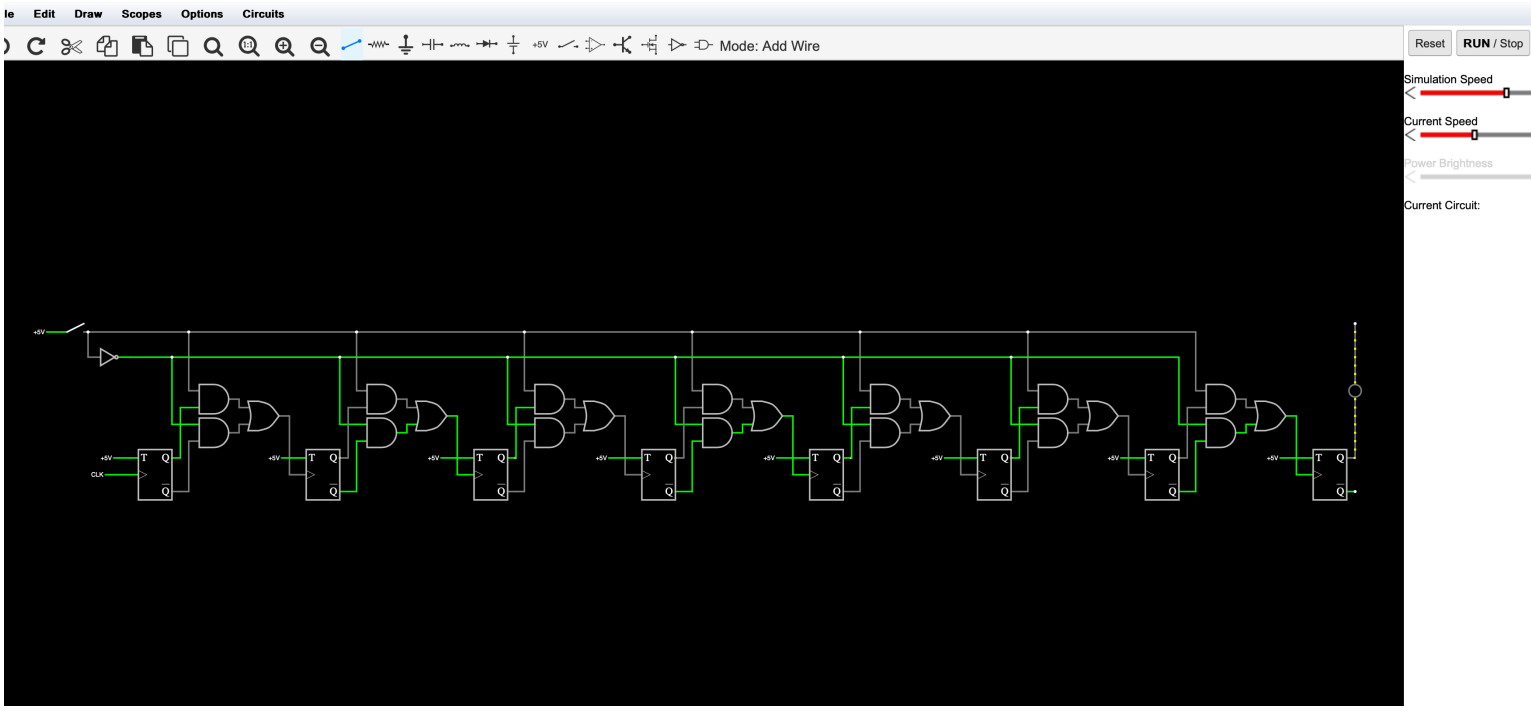
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_up_down_counter is
end tb_up_down_counter;

architecture Behavioral of tb_up_down_counter is
    -- Component Declaration
    component up_down_counter
        Port (
            clk       : in STD_LOGIC;
            reset     : in STD_LOGIC;
            load      : in STD_LOGIC;
            up_down   : in STD_LOGIC;
            load_data : in STD_LOGIC_VECTOR (7 downto 0);
            counter   : out STD_LOGIC_VECTOR (7 downto 0)
        );
    end component;

    -- Signal Declaration
    signal clk       : STD_LOGIC := '0';
    signal reset     : STD_LOGIC := '0';
    signal load      : STD_LOGIC := '0';
    signal up_down   : STD_LOGIC := '1';
    signal load_data : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
    signal counter   : STD_LOGIC_VECTOR (7 downto 0);

begin
    -- Instantiate the Unit Under Test (UUT)
    uut: up_down_counter
        Port map (
            clk => clk,
            reset => reset,
            load => load,
            up_down => up_down,
            load_data => load_data,
            counter => counter
        );

    -- Clock Generation
    clk_process : process
    begin
        while now < 1 ms loop
            clk <= '0';
            wait for 10 ns;
            clk <= '1';
            wait for 10 ns;
        end loop;
        wait;
    end process;

    -- Stimulus Process
    stimulus_process : process
    begin
        -- Test Reset
        reset <= '1'; wait for 20 ns;
        reset <= '0'; wait for 20 ns;

        -- Test Counting Up
        up_down <= '1'; wait for 200 ns;

        -- Test Counting Down
        up_down <= '0'; wait for 200 ns;

        -- Test Load Functionality
```

```vhdl
library IEEE;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity up_down_counter is
    Port (
        clk       : in STD_LOGIC;                       -- Clock sign
        reset     : in STD_LOGIC;                       -- Reset sign
        load      : in STD_LOGIC;                       -- Load enabl
        up_down   : in STD_LOGIC;                       -- Direction
        load_data : in STD_LOGIC_VECTOR (7 downto 0);   -- Data to lo
        counter   : out STD_LOGIC_VECTOR (7 downto 0)   -- 8-bit coun
    );
end up_down_counter;

architecture Behavioral of up_down_counter is
    signal temp_counter : STD_LOGIC_VECTOR (7 downto 0) := (others => '0
signal
begin
    process (clk, reset)
    begin
        if reset = '1' then
            temp_counter <= (others => '0'); -- Reset counter to 0
        elsif rising_edge(clk) then
            if load = '1' then
                temp_counter <= load_data;  -- Load the input data
            elsif up_down = '1' then
                temp_counter <= temp_counter + 1; -- Increment counter
            else
                temp_counter <= temp_counter - 1; -- Decrement counter
            end if;
        end if;
    end process;

    counter <= temp_counter; -- Assign internal counter to output
end Behavioral;
```

EP Wave diagram



2. Falstad Circuit Design



Circuit Description:

1. T Inputs: All T inputs of flip-flops are set to `1`.

2. Clock Propagation:

- The CLK input of Q0 connects directly to the clock source.

- For Q1 to Q7, the CLK input is derived from the OR gate output of the preceding flip-flop.

3. UP/DOWN Logic:

- AND Gate 1: Inputs = UP/DOWN signal, Q output of the current flip-flop.

- AND Gate 2: Inputs = NOT(UP/DOWN), Q' (NOT Q) output of the current flip-flop.

- OR Gate: Combines outputs of the two AND gates and feeds the CLK input of the next flip-flop.

Simulation Results

1. **UP** Mode**:** In VHDL, output transitions from 00000000 to 111111111 on every clock cycle when UP/DOWN = ‘1'. In Falstad, Q7 LED lights up sequentially to display the binary counting state.
2. **DOWN** Mode**:** In VHDL: Output transitions from 11111111 to 00000000 on every clock cycle when UP/DOWN = ‘0’. In Falstad, Q7 LED lights down sequentially, showing downward counting.
3. **RESET (VHDL Only):** In VHDL, when RESET = '1', the counter resets to 00000000.

Challenges Faced and Fixes

1. **Writing and Debugging VHDL Code:** Ensuring the behavioral VHDL implementation worked as expected required thorough testing of edge cases such as RESET and transitions between UP and DOWN modes. Debugging the interactions between signals and verifying the correct operation of synchronous clock edges took multiple iterations.
2. **Clock Propagation Issues in Falstad:** Initially, the CLK inputs of Q1 to Q7 were not correctly connected, resulting in only Q0 toggling. This was resolved by ensuring that the OR gate output from the preceding flip-flop drives the CLK input of the next flip-flop.
3. **UP/DOWN Logic Configuration:** The AND and OR gates for UP/DOWN control required careful debugging to ensure that Q and Q' outputs were feeding correctly into the logic. Testing individual logic paths and gates in isolation was critical to identify and fix incorrect connections.

Conclusion

The 8-bit Up/Down Counter was successfully implemented and simulated using both VHDL and Falstad. While the VHDL implementation includes RESET functionality, the Falstad version provides a practical, logic-based visualization without RESET. Both designs demonstrate accurate and reliable counting behavior with smooth transitions between UP and DOWN modes.

References

1. EDA Playground.
2. Falstad Circuit Simulator.
3. Course materials on flip-flops and synchronous counters.