**Embedded Systems Spring 2019**

**Lab 1**

**Jonathan Colella**

**February 28, 2019**

**Purpose**

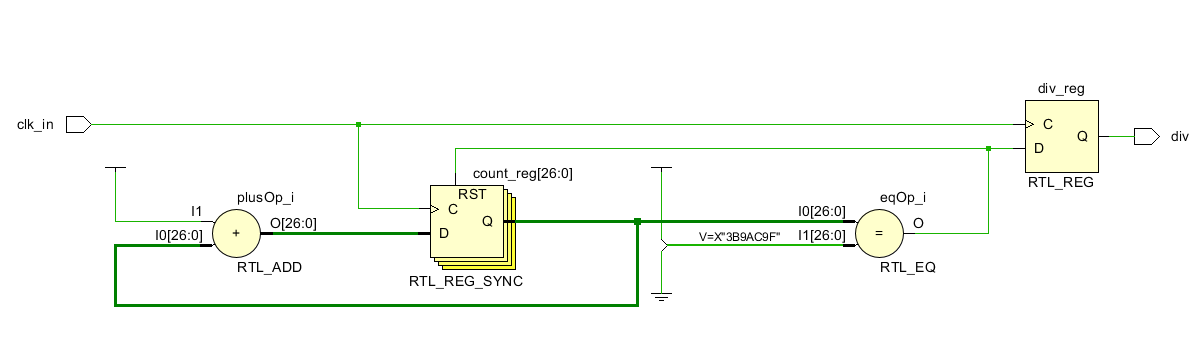
This lab introduces counters and debouncers in the context of VHDL modelling. I will create models for the various required components, then combine them into a single unit. The lab assignment will help develop my thinking when writing VHDL, and to gain experience in designing embedded systems. This will help serve as a foundation for later labs in this course.

**1a) Clock Divider**

**Theory of Operation**

This device takes in a 125MHz clock signal and outputs a 2Hz clock signal. It operates by counting every time a clock is input, and only outputs a clocked ‘1’ once for every 62,500,000 input clocks. It resets the counter each time this happens.

**Elaboration Schematic**



**Design**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL; --this library makes incrementing the vector easier

entity clock\_div is

Port ( clk\_in : in STD\_LOGIC;

div : out STD\_LOGIC);

end clock\_div;

--divides a 125MHZ signal down to a 2Hz signal

architecture Behavioral of clock\_div is

signal count: std\_logic\_vector (26 downto 0); --26 bits needed to count to 62,499,999

begin

process(clk\_in) begin

if (rising\_edge(clk\_in)) then

if (count = "11101110011010110010011111") then --binary for 62,499,999

count <= (others => '0'); --reset to 0 when we get to 62.5 million

div <= '1'; --output is high

else

count <= count + 1; --increment by one otherwise with help of unsigned

div <= '0'; --output is low

end if;

end if;

end process;

end Behavioral;

**Test Bench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity clock\_div\_tb is

end clock\_div\_tb;

architecture tb of clock\_div\_tb is

component clock\_div

port ( clk\_in : in std\_logic;

div : out std\_logic);

end component;

signal tb\_clk : std\_logic := '0';

signal tb\_div : std\_logic := '0';

begin

dut: clock\_div port map (clk\_in => tb\_clk, div => tb\_div);

process begin

tb\_clk <= '0'; --simulate 125MHz clock

wait for 4 ns;

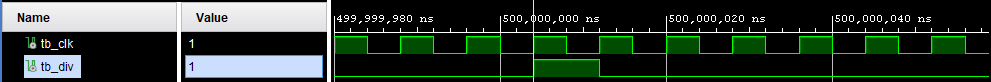
tb\_clk <= '1';

wait for 4 ns;

end process;

end tb;

**Simulation**

****

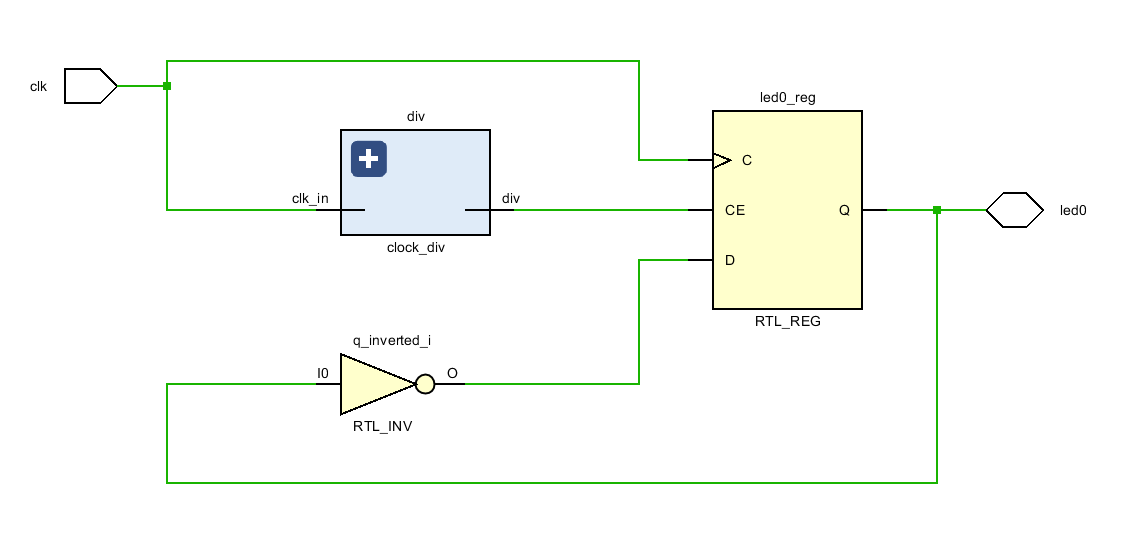
The output oscillates at 2Hz.

**1b) Top-Level Divider**

**Theory of Operation**

This device is the top-level RTL schematic for the clock divider. It divides the clock down to 2 Hz and stores the current clock value in a register. This register allows the clock value to be held at either 1 or 0 as needed, as opposed to the clock\_div module that only generates pulses.

**Elaboration Schematic**

****

**Design**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity divider\_top is

Port ( clk : in STD\_LOGIC;

led0 : inout STD\_LOGIC);

end divider\_top;

architecture rtl\_ckt of divider\_top is

--components

component clock\_div

port(clk\_in : in std\_logic;

div : out std\_logic);

end component;

--signals

signal div\_output : std\_logic;

signal q\_inverted : std\_logic;

Begin

div: clock\_div

port map ( clk\_in => clk,

div => div\_output);

led\_reg: process(clk)

Begin

if (rising\_edge(clk)) then

if (div\_output = '1') then

led0 <= q\_inverted;

end if;

end if;

end process;

q\_inverted <= NOT(led0);

end rtl\_ckt;

**Discussion**

**1.1**

**1.2**

**2.1**

**2.2**

**2.3**

**2.4**