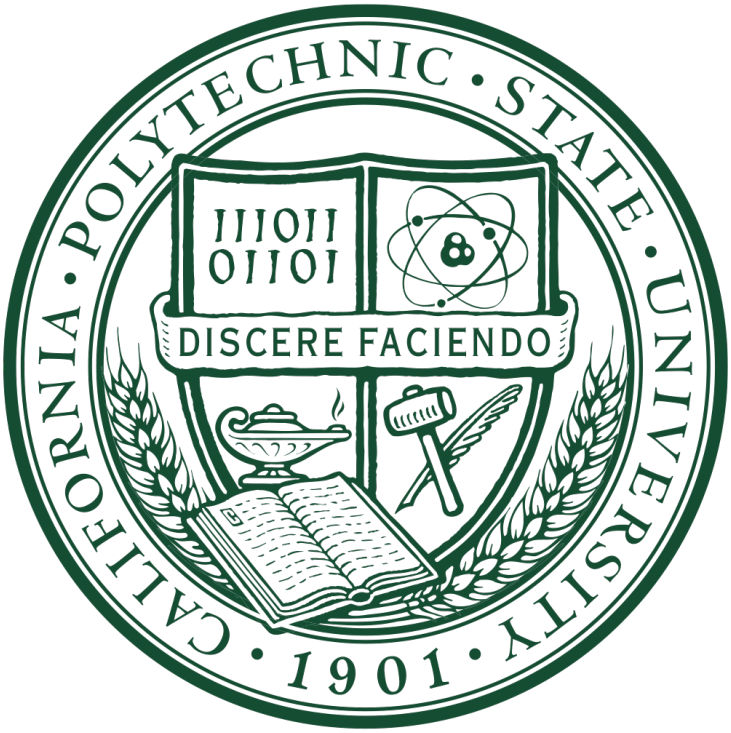
CPE 233: Computer Design and Assembly Language Programming

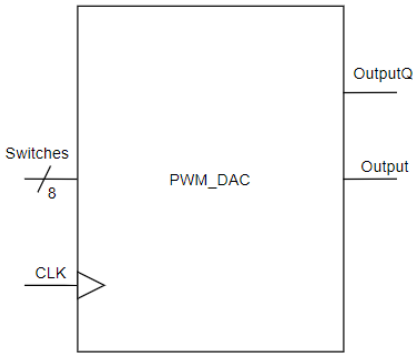
HW #1:
DAC using Pulse-Width Modulation with RC Filter

By: Roee Landesman and Amir Hashemizad



https://www.youtube.com/watch?v=1kDXINdvWHk&feature=youtu.be

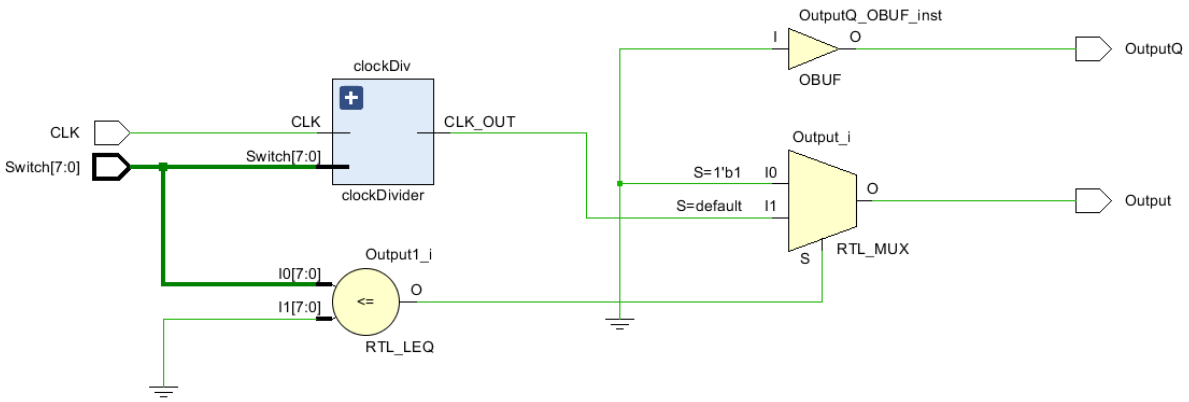
Black Box Block Diagram



Behavior Description

A pulse-width modulator controls the average output voltage of a microcontroller using a clock-divider. The component essentially determines the width of each pulse by translating a switch's value into binary and keeping the pulse high for that binary value. There is a simple counter in the clock-divider that increments on each rising edge. This entire PWM was then connected to a low-filter which allowed us to translate the sinusoidal wave into a square wave, effectively making a digital-to-analog convertor (DAC).

Structural Design



Specification

According the schematic that we drew, the component should be able to work on any sized clock speed. We created an 8-bit DAC which means we have up to 256 unique voltage partitions; depending on the voltage of the hardware itself, determines the resolution of the converted analog signal. Additionally, since we are using the Basys3 FPGA, the PWM DAC is limited to a clock speed of 100 mH, which means that the rising edge comes every 10 nanoseconds. The RC low-pass filter was build using a 1.6kOhm resistor and 100 nF capacitor to provide adequate time for the RC circuit to respond to the incoming rising edges.

VHDL Source Code

Main:

```
----------------------------------------------------------------------
-- Engineer: Amir Hashemizad and Roee Landesman
-- Create Date: 01/21/2018
-- Module Name: HW1project_module - Behavioral
-- Project Name: HW1
-- Target Devices: Basys3 Board
-- Revision 0.01 - File Created
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity project_module is
    Port ( Switch : in STD_LOGIC_VECTOR(7 downto 0);
           CLK    : in STD_LOGIC;
           OutputQ : out STD_LOGIC;
           Output : out STD_LOGIC);
end project_module;

architecture Behavioral of project_module is

    component clockDivider
        port(
            CLK : in std_logic;
            Switch : in std_logic_vector(7 downto 0);
            CLK_OUT : out std_logic);
    end component;

    signal OUT_TEMP : std_logic;

begin

    clockDiv: clockDivider port map(
        CLK => CLK, Switch => Switch, CLK_OUT => OUT_TEMP);
```

```
    process
    Begin
        if (Switch <= "00000000") then
            Output <= '0';
        else
            Output <= OUT_TEMP;
        end if;
        OutputQ <= '0';
    end process;

end Behavioral;
```

Clock Divider:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.all;

entity clockDivider is
    Port ( CLK : in STD_LOGIC;
           Switch   : in STD_LOGIC_VECTOR(7 downto 0);
           CLK_OUT : out STD_LOGIC);
end clockDivider;

architecture Behavioral of clockDivider is

    signal TEMP_CLK : std_logic := '1';

begin

    process(CLK)
        variable count : integer := 0;
    begin
        if (rising_edge(CLK)) then
            if (count = to_integer(unsigned(Switch))) then
                TEMP_CLK <= '0';
            end if;
            if (count = 255) then
                TEMP_CLK <= '1';
                count := 0;
            end if;
            count := count + 1;
        end if;
    end process;

    CLK_OUT <= TEMP_CLK;

end Behavioral;
```

Example Use Code

The following RAT Assembly code utilizes the PWM DAC with RC Low filter that we built in order to create a triangle wave. This wave starts with a Vrms of 1.65 Volts and then increases and decreases from there up to 255.

```
.EQU PWM_PORT = 0x42

.CSEG
.ORG 0x20

MAIN: MOV   R0, 0x7F
UP:   OUT   R0, PWM_PORT
      ADD   R0, 0x01
      CMP   R0, 0xFF
      BREQ  DOWN
      BRN   UP
DOWN: OUT   R0, PWM_PORT
      SUB   R0, 0x01
      CMP   R0, 0x7F
      BREQ  UP
      BRN   DOWN
```