Experiment No. 10
Digital to Analog Converters
ECE 446

Peter Chinetti

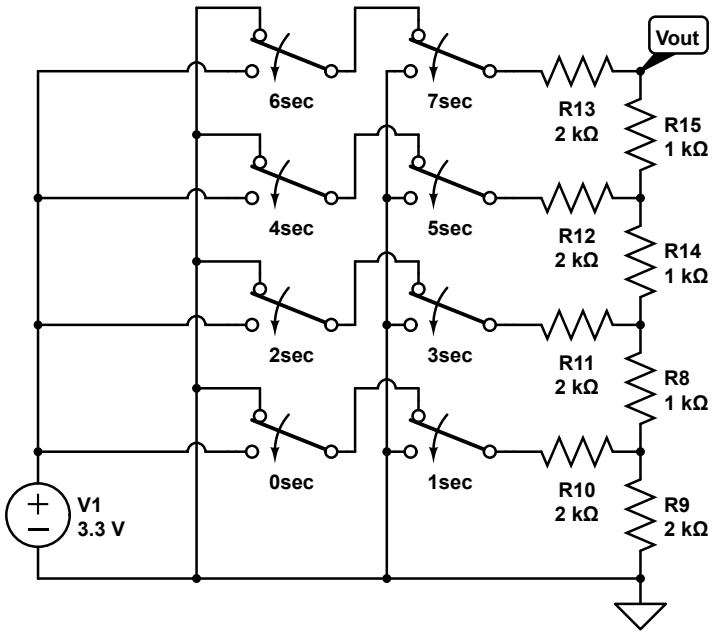
December 4, 2014

Instructor: Professor Shanechi

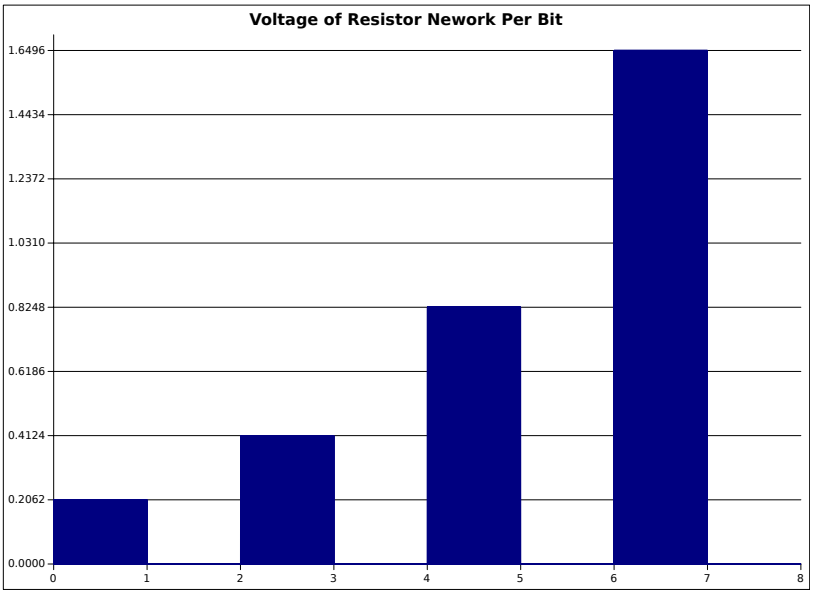
1 Introduction

Digital signals are represented as a high and ground voltage. When interacting with the physical world however, it is often useful to be able to produce a range of voltages. A digital to analog converter translates a digital vector into a range of voltages.

The translation is done with a resistor bridge as shown below:



Graphing Vout over time gives each bit's individual voltage contribution.



Where the LSB is on the left, and the MSB is on the right. To find any other combination of powered and grounded bits, simply add the voltages for the bits that are high (3.3v) to find the resultant Vout.

Decimal Input Value	$OutputVoltage$
0	0.00000
1	0.20625
2	0.41250
3	0.61875
4	0.82500
5	1.03125
6	1.23750
7	1.44375
8	1.65000
9	1.85625
10	2.06250
11	2.26875
12	2.47500
13	2.68125
14	2.88750
15	3.09375

2 Procedure

a. Write VHDL to implement counter

b. Assign pins to ports

c. Simulate

d. Program and Test

3 Equipment

- PC
- Spartan-3E development board

4 Code

4.1 Counter Module

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

entity Counter is
    Port ( clk_in : in  STD_LOGIC;
           o : out STD_LOGIC_VECTOR(3 downto 0) :="0000";
        tc : out STD_LOGIC := '0';
           reset : in  STD_LOGIC);
end Counter;

architecture Behavioral of Counter is
  signal counter: unsigned(3 downto 0) :="0000";
  signal cnt_clk: STD_LOGIC;

  component selectable_clock is
  Port ( clk : in std_logic;
         s0 : in std_logic;
         s1 : in std_logic;
         out_clk : out std_logic);
  end component;
begin
  clk_div : selectable_clock
  port map(
    clk => clk_in,
    s0 => '1',
    s1 => '1',
    out_clk => cnt_clk);

  o <= std_logic_vector(counter);
  process(cnt_clk, reset)
  begin
    if rising_edge(cnt_clk) then
```

```
      if reset = '1' then
        counter <= "0000";
      elsif counter <= "1111" then
        counter <= counter + "0001";
      elsif counter = "1111" then
        counter <= "0000";
      end if;
    end if;
  end process;
  process(counter)
  begin
    if counter = "1111" then
      tc <= '1';
    else
      tc <= '0';
    end if;
  end process;
end Behavioral;
```

Counter.vhd

4.2 Counter Test Module

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;


ENTITY Counter_test IS
END Counter_test;

ARCHITECTURE behavior OF Counter_test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Counter
    PORT(
         clk_in : IN  std_logic;
         o : OUT  std_logic_vector(3 downto 0);
         tc : OUT  std_logic;
         reset : IN  std_logic
        );
    END COMPONENT;


   --Inputs
   signal clk_in : std_logic := '0';
   signal reset : std_logic := '0';

  --Outputs
   signal o : std_logic_vector(3 downto 0);
   signal tc : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

```

```
BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: Counter PORT MAP (
          clk_in => clk_in,
          o => o,
          tc => tc,
          reset => reset
        );

   -- Clock process definitions
   clk_process :process
   begin
    clk_in <= '0';
    wait for clk_period/2;
    clk_in <= '1';
    wait for clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for clk_period*10;

      -- insert stimulus here

      wait;
   end process;

END;
```

Counter_test.vhd

4.3 Clock Divider Module

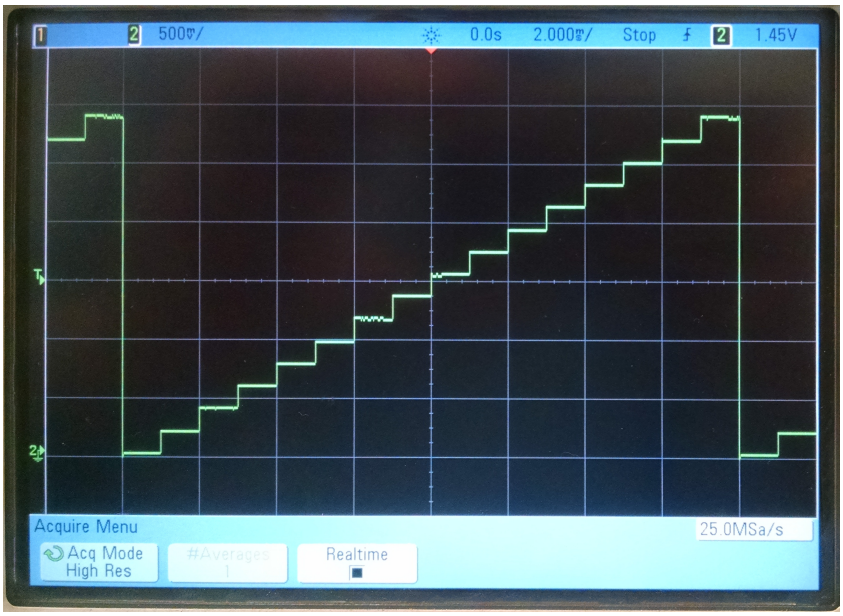
```
-- Selectable output frequency clock divider code.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity selectable_clock is
  Port ( clk : in std_logic;
         s0 : in std_logic;
         s1 : in std_logic;
         out_clk : out std_logic);
end selectable_clock;
--If s1 and s0 are both low, the output clock rate is 1/10 Hz.
--If s1 is low and s0 is high, the output clock rate is 1 Hz.
--If s1 is high and s0 is low, the output clock rate is 10 Hz.
--If s1 and s0 are both high, the output clock rate is 1 KHz.
architecture Behavioral of selectable_clock is
```

```
begin
  process (clk, s0, s1)
    variable count : integer := 0;
  begin
    if clk = '1' and clk'event then
      count := count + 1;
      -- Start a process.
      -- Variable declaration.
      -- Rising edge detection.
      -- Code to create the 1/10 Hz clock.
      if s0 = '0' and s1 = '0' then
        if count >= 500000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        if count >= 0 and count <= 250000000 then
          out_clk <= '1';
        -- High portion of 1/10 HZ clock.
        else
          out_clk <= '0';
        -- Low portion of 1/10 HZ clock.
        end if;
      end if;
      -- Code to create the 1 Hz clock.
      if s0 = '1' and s1 = '0' then
        if count >= 50000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        if count >= 0 and count <= 25000000 then
          out_clk <= '1';
        -- High portion of 1 HZ clock.
        else
          out_clk <= '0';
        -- Low portion of 1 HZ clock.
        end if;
      end if;
      -- Code to create the 10 Hz clock.
      if s0 = '0' and s1 = '1' then
        if count >= 5000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        if count >= 0 and count <= 2500000 then
          out_clk <= '1';
        -- High portion of 10 HZ clock.
        else
          out_clk <= '0';
        -- Low portion of 10 HZ clock.
        end if;
      end if;
      -- Code to create the 1 KHz clock.
      if s0 = '1' and s1 = '1' then
        if count >= 50000 then
```

```
74            -- Taken off a 50MHz clock.
              count := 0;
76          -- Reset count for next cycle.
            end if;
78          if count >= 0 and count <= 25000 then
              out_clk <= '1';
80          -- High portion of 1 KHz clock.
            else
82            out_clk <= '0';
            -- Low portion of 1 KHz clock.
84          end if;
        end if;
86    end if;
  end process;
88 end Behavioral;
```

clk_div.vhd

5 Scope Trace



6 Tabular Measurements

Decimal Input Value	*MeasuredOutput*	Calculated Output	Percent Error
0	0.00000	0.00000	0.00%
1	0.18700	0.20625	−9.33%

2	0.37500	0.41250	−9.09%
3	0.56250	0.61875	−9.09%
4	0.74375	0.82500	−9.85%
5	0.92500	1.03125	−10.30%
6	1.13125	1.23750	−8.59%
7	1.32500	1.44375	−8.23%
8	1.50000	1.65000	−9.09%
9	1.70000	1.85625	−8.42%
10	1.86750	2.06250	−9.45%
11	2.06875	2.26875	−8.82%
12	2.25625	2.47500	−8.84%
13	2.45000	2.68125	−8.62%
14	2.65000	2.88750	−8.23%
15	2.83750	3.09375	−8.28%

7 Conclusions

The purpose of this lab was achieved. A DAC was built and tested. Operation was verified through simulation and physical implementation.