Experiment No. 12
Successive Approximation A/D Converter
ECE 446

Peter Chinetti

December 4, 2014

Instructor: Professor Shanechi

1 Introduction

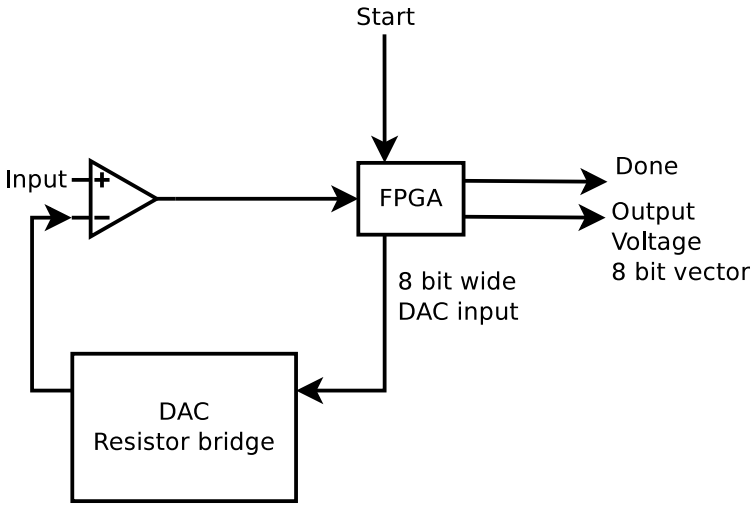
When working with real world signals, it is useful to be able to sample voltage levels. In this lab, the design of the Analog to Digital Converter is improved and implemented.

The translation is done with a DAC and a comparator. The sample voltage is held to one end of the comparator, and the DAC is tied into the other side. Each bit (starting with the MSB) in the DAC is set, the comparator output is tested, and unset if the signal goes too high.

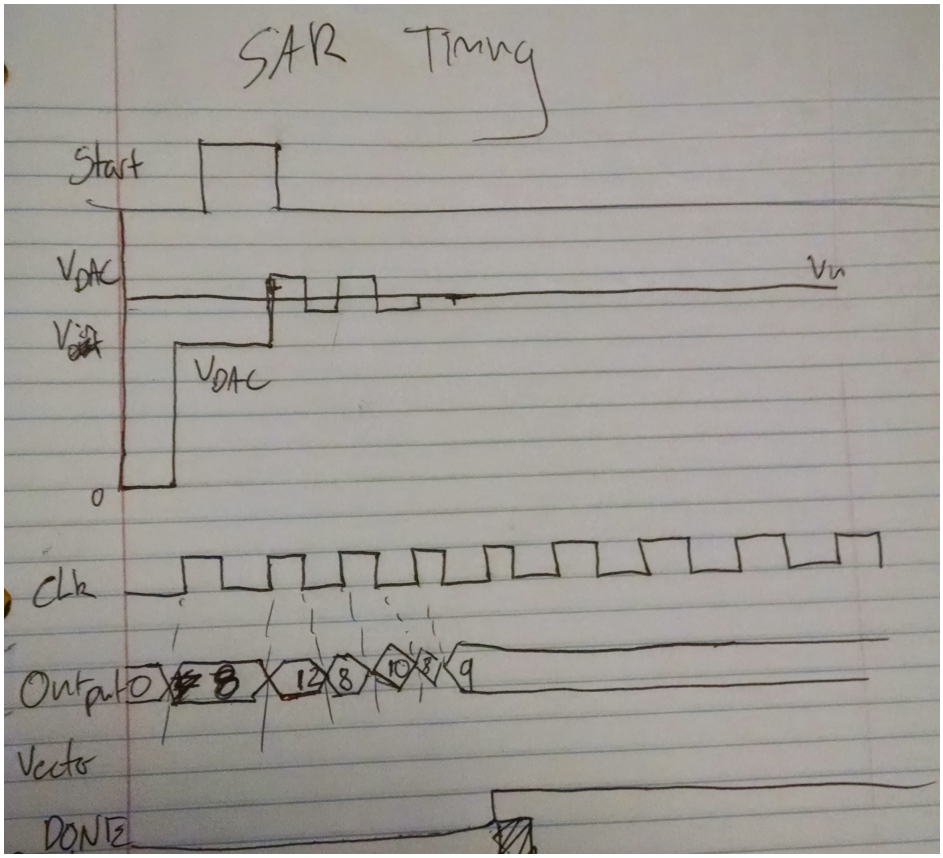
This algorithm improves the two problems with the previous design. It takes time proportional to the logartithm of the number of output state, and it takes constant time to preform for all inputs. The only situation where this design performs more poorly than the original design is when the input voltage is lower than 4 DAC divisions.

2 Pre-Lab Questions

2.1



2.2



2.3

Maximum error occurs when the input is very close to, but slightly less than, one of the DAC's output voltage. In that case, the voltage gets rounded down to the voltage below. This corresponds to (at the limit) an error of $\frac{V_r}{2^n}$. For a $V_r = 5V$ and $n = 4$, the maximum error is 0.3125 volts.

3 Procedure

a. Build resistor net and comparator circuit

b. Write VHDL to implement ADC

c. Assign pins to ports

d. Simulate

e. Program and Test

4 Equipment

- PC
- Spartan-3E development board
- Op-Amp
- Assorted resistors and diodes
- Breadboard

5 Code

5.1 Top-Level Module

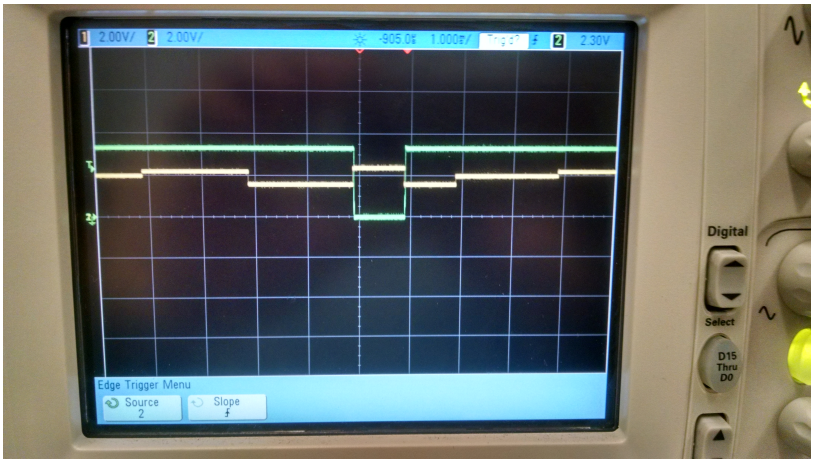
```
1
library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;

5 entity sar_fsm is
    Port ( start : in  STD_LOGIC;
7          en : in  STD_LOGIC;
           dac_out : out  STD_LOGIC_VECTOR (3 downto 0);
9          result : out  STD_LOGIC_VECTOR (3 downto 0);
           done_out : out  STD_LOGIC;
11         clk_in : in  STD_LOGIC);
end sar_fsm;
13
architecture Behavioral of sar_fsm is
15  type state is (DONE, COMP_B3, SET_B2, COMP_B2, SET_B1, COMP_B1,
    SET_B0, COMP_B0);
  signal p, f : state;
17  signal d, q : STD_LOGIC_VECTOR (3 downto 0);
  signal clk_i : STD_LOGIC;
19
  component selectable_clock is
21  Port ( clk : in std_logic;
         s0 : in std_logic;
23       s1 : in std_logic;
         out_clk : out std_logic);
25  end component;
begin
27  div : selectable_clock
  port map(
29    clk => clk_in,
    s0 => '1',
31    s1 => '1',
    out_clk => clk_i);
33
  dac_out <= q;
```

```
35    result <= q;

37    process (clk_i)
      begin
39      if rising_edge(clk_i) then
          p <= f;
41        q <= d;
        end if;
43    end process;

45    process (p, en, start, q)
      begin
47      case p is
          when DONE =>
49          if start = '0' then
              d <= q;
51            f <= DONE;
              done_out <= '1';
53          else
              f <= COMP_B3;
55            d <= "1000";
              done_out <= '0';
57          end if;
          when COMP_B3 =>
59          f <= SET_B2;
            done_out <= '0';
61          if en = '0' then
              --d(3) <= '0';
63            d <= q and "0111";
            else
65            d <= q;
            end if;
67        when SET_B2 =>
            f <= COMP_B2;
69          done_out <= '0';
            --d(2) <= '1';
71          d <= q or "0100";
          when COMP_B2 =>
73          f <= SET_B1;
            done_out <= '0';
75          if en = '0' then
              --d(2) <= '0';
77            d <= q and "1011";
            else
79            d <= q;
            end if;
81        when SET_B1 =>
            f <= COMP_B1;
83          done_out <= '0';
            --d(1) <= '1';
85          d <= q or "0010";
          when COMP_B1 =>
87          f <= SET_B0;
            done_out <= '0';
89          if en = '0' then
              --d(1) <= '0';
91            d <= q and "1101";
```

```
          else
93          d <= q;
          end if;
95      when SET_B0 =>
          f <= COMP_B0;
97        done_out <= '0';
          --d(0) <= '1';
99        d <= q or "0001";
        when COMP_B0 =>
101       f <= DONE;
          done_out <= '0';
103       if en = '0' then
            --d(0) <= '0';
105         d <= q and "1110";
          else
107         d <= q;
          end if;
109     end case;
    end process;
111 end Behavioral;
```

sar_fsm.vhd

6 Scope Trace



7 Conclusions

The purpose of this lab was achieved. An improved ADC was built and tested. Operation was verified through simulation and physical implementation.