Experiment No. 5
Barrel Shifter
ECE 446

Peter Chinetti

September 29, 2014

Date Performed:	September 21, 2014
Instructor:	Professor Shanechi

1 Introduction

Data shifting circuits are of critical importance in CPU design. They are useful for bit-mask manipulation and various other operations that would be cumbersome using other mathematical functions. When designing and building a barrel shifter, there are several functional specifications that must be considered. For instance, will the shifter move data to the left or to the right? More often than not, in order to allow the shifter circuit to be general purpose, it should support both left and right shifts of the data with which it is supplied, based upon some selection input setting. Another issue that must be considered when designing a shifter is what type of shift operation will be performed. When a logical shift circuit moves data left or right, the data shifted out of the range of the data storage element is dropped. In addition, the empty space created in the storage element with each bit shift is filled with a pre-determined, or runtime specified, bit value that is typically zero. This type of operation is most often useful for bit-mask manipulations.

A circular shift circuit behaves similarly to the logical shift circuit; however, bits that are shifted out of one end of the storage element are fed back into the other end as inputs. This allows all of the original data to be kept, even though it is moved around. Circular shift functionality is useful for certain bit manipulations that may or may not use masks. Finally, arithmetic shift functionality is set up to achieve very low cost multiplications or divisions by powers of two. To achieve this, a left shift operation will input zeros into the newly vacated LSBs of the data storage element. A right shift operation, on the other hand, will replicate the sign bit of the original data into the MSBs of the data storage element that are emptied during the shift operation. This is the type of shifting circuit that will be designed and implemented in this laboratory.

A final consideration when designing shifting circuits is the amount of shift the circuit will support. A single bit shifting circuit, while simple to design, will not be terribly useful, as multi-bit shift operations will require the data to be fed through the shifter several times. A shifter that will handle a variety of different shift amounts will be more complicated to design, but will ultimately be more useful.

2 Procedure

a. Write VHDL to implement encoder/decoder logic.

b. Assign pins to ports

c. Simulate

d. Program and Test

3 Equipment

- PC
- Spartan-3E development board

4 Code

4.1 Top-level Module

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

entity Shifter is
  Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);
         os : out  STD_LOGIC_VECTOR (7 downto 0);
         sh : in  STD_LOGIC_VECTOR (2 downto 0);
         d : in  STD_LOGIC);
end Shifter;

architecture Behavioral of Shifter is

  signal l_0, l_1, l_2, r_0, r_1, r_2 : STD_LOGIC_VECTOR (7 downto
    0);
  signal z : STD_LOGIC;

  component Mux
    Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
           b : in STD_LOGIC_VECTOR (7 downto 0);
           s : in STD_LOGIC;
           o : out STD_LOGIC_VECTOR (7 downto 0));
  end component;
```

```
begin

  z <= '0';

  --shift left by 1

  mux_l_1: Mux
  port map(
  a(7 downto 1) => i(6 downto 0),
  a(0) => '0',
  b => i,
  s => sh(0),
  o => l_0
);

mux_l_2: Mux
port map(
a(7 downto 2) => l_0(5 downto 0),
a(1) => '0',
a(0) => '0',
b => l_0,
s => sh(1),
o => l_1
  );

  mux_l_4: Mux
  port map(
  a(7 downto 4) => l_1(3 downto 0),
  a(3 downto 0) => "0000",
  b => l_1,
  s => sh(2),
  o => l_2
);

mux_r_1: Mux
port map(
a(6 downto 0) => i(7 downto 1),
a(7) => i(7),
b => i,
s => sh(0),
o => r_0
  );

  mux_r_2: Mux
  port map(
  a(5 downto 0) => r_0(7 downto 2),
  a(7) => r_0(7),
  a(6) => r_0(7),
  b => r_0,
  s => sh(1),
  o => r_1
);

mux_r_4: Mux
port map(
a(3 downto 0) => r_1(7 downto 4),
a(7) => r_1(7),
a(6) => r_1(7),
a(5) => r_1(7),
a(4) => r_1(7),
b => r_1,
s => sh(2),
o => r_2
  );

  mux_d: Mux
  port map(
      a => r_2,
      b => l_2,
      s => d,
      o => os
    );

end Behavioral;
```

Barrel_Shifter_better/Shifter.vhd

4.2 Mux

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

entity Mux is
  Port ( A : in  STD_LOGIC_VECTOR(7 downto 0);
         B : in  STD_LOGIC_VECTOR(7 downto 0);
         S : in  STD_LOGIC;
         O : out  STD_LOGIC_VECTOR(7 downto 0) );
end Mux;

architecture arch_mux of Mux is

begin

  process (S,A,B)
  begin

    if S = '0' then
      O <= A;
    elsif S = '1' then
      O <= B;
    end if;
  end process;
end arch_mux;
```

Barrel_Shifter_better/Mux.vhd

4.3 Shifter Test

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

```

ENTITY shifter_test IS
  END shifter_test;

ARCHITECTURE behavior OF shifter_test IS

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

  COMPONENT Shifter
    PORT(
          i : IN  std_logic_vector(7 downto 0);
          os : OUT  std_logic_vector(7 downto 0);
          sh : IN  std_logic_vector(2 downto 0);
          d : IN  std_logic
        );
  END COMPONENT;


   --Inputs
   signal i : std_logic_vector(7 downto 0) := (others => '0');
   signal sh : std_logic_vector(2 downto 0) := (others => '0');
   signal d : std_logic := '0';

   --Outputs
   signal os : std_logic_vector(7 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: Shifter PORT MAP (
               i => i,
               os => os,
               sh => sh,
               d => d
             );


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
     wait for 10 ns;

     d <= '1';
     sh <= "111";
     i <= "11111111";

     wait for 10 ns;

     sh <= "110";

     wait for 10 ns;

     sh <= "101";

     wait for 10 ns;

     sh <= "011";

     wait for 10 ns;

     sh <= "111";
     i <= "01111111";
     d <= '0';

     wait for 10 ns;

     sh <= "110";

     wait for 10 ns;

     sh <= "101";

     wait for 10 ns;

     sh <= "011";


     wait;
   end process;

END;
```

Barrel_Shifter_better/shifter_test.vhd

4.4 Mux Test

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

ENTITY mux_test IS
  END mux_test;

ARCHITECTURE behavior OF mux_test IS

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

  COMPONENT Mux
    PORT(
          A : IN  std_logic;
          B : IN  std_logic;
          S : IN  std_logic;
          O : OUT  std_logic
        );
  END COMPONENT;


   --Inputs
  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal S : std_logic := '0';
```

```
   --Outputs
  signal O : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
  uut: Mux PORT MAP (
          A => A,
          B => B,
          S => S,
          O => O
        );

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

    A <= '1';

    wait for 100 ns;

    S <= '1';

    wait for 100 ns;

    B <= '1';

    wait for 100 ns;

    A <= '0';

    wait for 100 ns;

    A <= '1';
    B <= '0';

    wait for 100 ns;

    S <= '0';

      -- insert stimulus here

    wait;
  end process;

END;
```

Barrel_Shifter_better/mux_test.vhd

5 Conclusions

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