Experiment No. 2
Four-Bit Ripple-Carry Adder/Subtractor
ECE 446

Peter Chinetti

September 16, 2014

Date Performed:	September 9, 2014
Instructor:	Professor Shanechi

1 Introduction

Ripple adders are a type of adder circuit that uses chained single bit adders to create a wider adder unit. They were chosen for this lab to demonstrate how VHDL allows for modular circuit construction. This modular construction lays at the core of why hardware programming languages have become so popular: they allow design reuse.

2 Background

2.1 Karnaugh Maps

2.1.1 Sum

		A	B		
		00	01	11	10
Cin	0	0	1	0	1
	1	1	0	1	0

2.1.2 C_{out}

		A	B		
		00	01	11	10
Cin	0	0	0	$\boxed{1}$	0
	1	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$

2.2 Minimized Equations

$Sum = A \oplus B \oplus C_{in} \oplus op\_sel$
$C_{out} = C_{in}\overline{A}\overline{(B \oplus op\_sel)} + \overline{C_{in}}A(B \oplus op\_sel) + C_{in}A(B \oplus op\_sel) + C_{in}A\overline{(B \oplus op\_sel)} +$
$A(B \oplus op\_sel) + C_{in}A\overline{(B \oplus op\_sel)}$

3 Procedure

a. Generate minimized equations for adder.

b. Write VHDL to implement logic.

c. Assign pins to ports

d. Simulate

e. Program and Test

4 Equipment

- PC
- Spartan-3E development board

5 Code

5.1 Top-level Module

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

entity Adder is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : in  STD_LOGIC_VECTOR (3 downto 0);
           Co : out  STD_LOGIC;
           S : out  STD_LOGIC_VECTOR (3 downto 0);
        OP_sel : in STD_LOGIC);
end Adder;

architecture Behavioral of Adder is

  signal c_0, c_1, c_2 : STD_LOGIC;

  component adder_block
    Port( a, b, ci, op_sel : in STD_LOGIC;
        s, co : out STD_LOGIC);

  end component;
begin
```

```
  adder_0: adder_block
    port map (
      a => A(0),
      b => B(0),
      op_sel => OP_sel,
      ci => OP_sel,
      co => c_0,
      s => S(0)
    );
  adder_1: adder_block
    port map (
      a => A(1),
      b => B(1),
      op_sel => OP_sel,
      ci => c_0,
      co => c_1,
      s => S(1)
    );
  adder_2: adder_block
    port map (
      a => A(2),
      b => B(2),
      op_sel => OP_sel,
      ci => c_1,
      co => c_2,
      s => S(2)
    );
  adder_3: adder_block
    port map (
      a => A(3),
      b => B(3),
      op_sel => OP_sel,
      ci => c_2,
      co => Co,
      s => S(3)
    );

end Behavioral;
```

Adder.vhd

5.2 1 Bit Adder Module

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

entity adder is
  Port ( A : in  STD_LOGIC;
         B : in  STD_LOGIC;
         Cin : in STD_LOGIC;
         sum : in STD_LOGIC;
         Cout : in STD_LOGIC;
         op_sel : in STD_LOGIC);
end adder;

```

```
architecture add_arch of adder is
begin

  sum <= (Cin and (not A) and (not (B xor op_sel))) or ((not Cin)
    and (not A) and (B xor op_sel)) or (Cin and A and (B xor op_sel
    )) or ((not Cin) and A and (not (B xor op_sel)));
  Cout <= (A and (B xor op_sel)) or (Cin and A) or (Cin and (B xor
    op_sel));
end add_arch;
```

short.vhd

5.3 Test Module

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


ENTITY small_adder IS
END small_adder;

ARCHITECTURE behavior OF small_adder IS

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

    COMPONENT adder_block
    PORT(
          a : IN  std_logic;
          b : IN  std_logic;
          ci : IN  std_logic;
          s : OUT  std_logic;
          co : OUT  std_logic;
          op_sel : IN  std_logic
         );
    END COMPONENT;


   --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal ci : std_logic := '0';
   signal op_sel : std_logic := '0';

  --Outputs
   signal s : std_logic;
   signal co : std_logic;


BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: adder_block PORT MAP (
          a => a,
          b => b,
          ci => ci,
          s => s,
          co => co,
          op_sel => op_sel
        );


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

     a <= '1';

     wait for 10 ns;

     a <= '0';
     b <= '1';

     wait for 10 ns;

     a <= '1';

     wait for 10 ns;

     a <= '0';
     b <= '0';

     wait for 10 ns;

     op_sel <= '1';
     ci <= '1';

     wait for 10 ns;

     a <= '1';

     wait for 10 ns;

     a <= '0';
     b <= '1';

     wait for 10 ns;

     a <= '1';

      wait;
   end process;

END;
```

test.vhd

6 Conclusions

The purpose of this lab was achieved. A ripple adder was built and tested. Additionally, the module functions of VHDL were demonstrated through the division of the single bit adder and full module.