Lab Experiment No. 02:

Four Bit Ripple-Carry Adder/ Subtractor in VHDL

By: Alexander Lukens (A20417036)
Instructor: Dr. Jafar Saniie
ECE 441-001
Lab Completion Date: 06/15/2020

Acknowledgement: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature:____Alexander Lukens____

I. Introduction

A. Purpose

The main purpose of this experiment is to introduce students to in depth elements of the Xilinx Vivado VHDL design environment. Students will become familiar with utilizing VHDL modules and components to construct more complicated digital logic designs. Specifically, students will design a digital logic full-adder, and then use this to construct a 4-bit adder or subtractor in VHDL. This code will then be simulated in the Vivado software environment and optionally uploaded to a Xilinx Artix-7 based FPGA.

B. Background

Xilinx is a technology company based in California, USA that creates field programmable gate array (FPGA), system-on-chips (SoCs), and other semiconductor products. They compliment their hardware designs with a robust software environment for designing digital logic circuits using System Verilog and VHSIC-Hardware Description Language (VHDL). This allows silicon engineers to rapidly develop and test digital logic designs without physically designing them on a breadboard or waiting for dedicated, application specific silicon to be produced.

In order to perform addition operations on binary numbers, the numbers must be broken down into individual bits. A Full-Adder is a digital logic device that takes in two bits (one from each operand) and a carry-in bit, and provides two output bits, a sum bit and a carry-out bit. Subtraction can be performed by XOR'ing one operand with "1" and asserting the carry-in signal of the least significant bit (creating the 2's complement of one operand). These bit-size operations can be chained together in a "ripple-adder" to conduct arithmetic addition and subtraction on larger binary numbers.

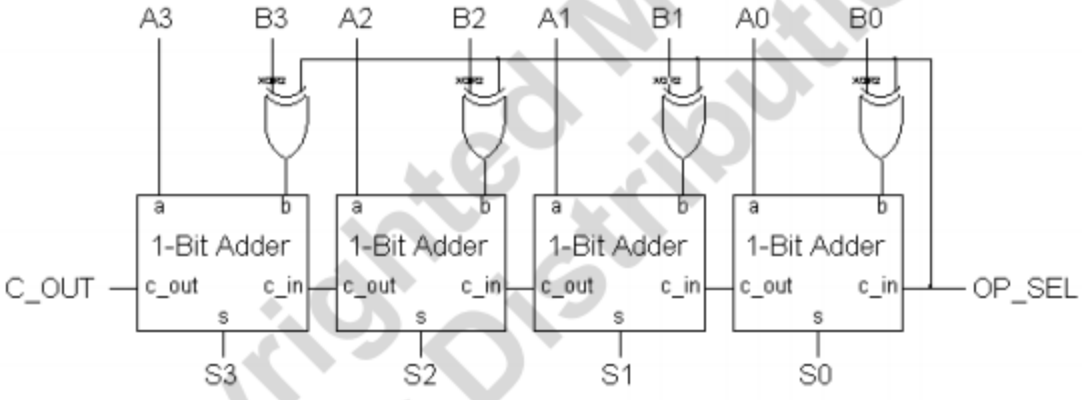


Figure 1 – The schematic diagram for the four-bit ripple-carry adder/subtractor circuit.

II. Lab Procedure and Equipment List

A. Equipment

Equipment required for this lab includes the following:

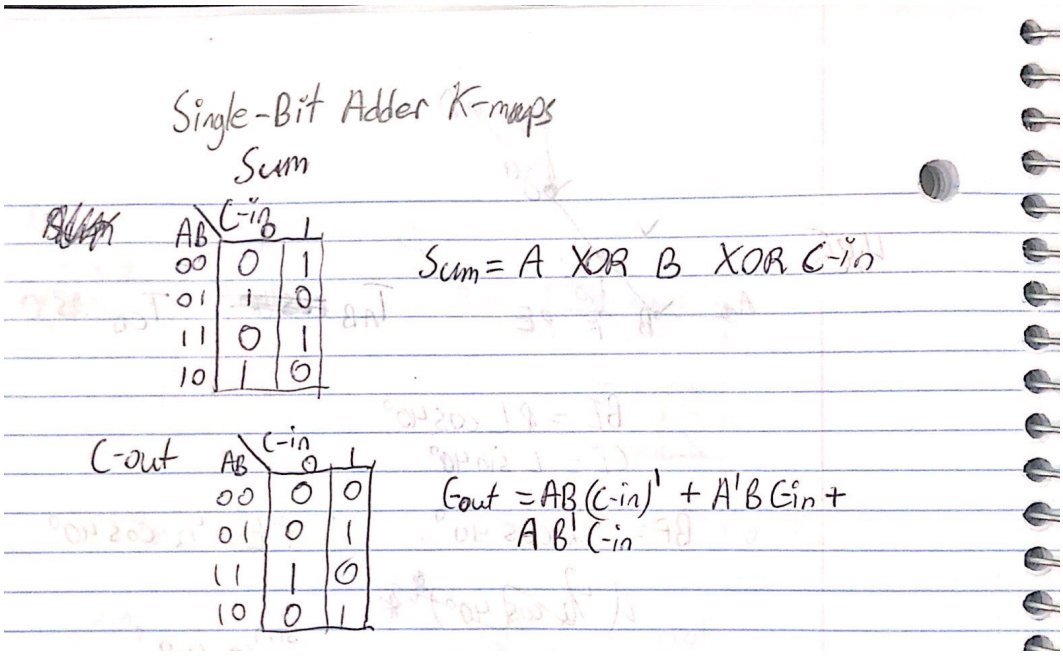
- Xilinx Artix-7 based FPGA
- PC with Xilinx Vivado software

B. Procedure

Students will follow the instructions outlined in the ECE 742 - FPGA Lab Manual for experiment 2. This includes designing and implementing a single-bit full adder and then using this VHDL component to create larger, 4-bit adders. Then, students will implement a "select" signal to allow users to either perform addition or subtraction on the two operands.

III. Results and Analysis

A. Discussion



Below is the code designed to implement a single bit adder (named SingleAdder), and subsequently used to design a 4-bit adder (named Adder4). This design uses 4 instances of the SingleAdder component linked together using internal signals to act as a single, 4-bit adder.

4-bit adder (no subtraction capabilities)

--Created by Alex Lukens for ECE 742 Summer 2020 class

-- this program will first design a single bit adder with carry capabilities and then use this component to

```
-- design a more complicated 4-bit adder that utilized 4 of the single-bit adders
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SingleAdder is
    Port ( A_in : in STD_LOGIC;
           B_in : in STD_LOGIC;
           Carry_in : in STD_LOGIC;
           Sum_out : out STD_LOGIC;
           Carry_out : out STD_LOGIC);
end SingleAdder;

architecture Behavioral of SingleAdder is

begin
    Sum_out <= A_in XOR B_in XOR Carry_in; --logic equation to find single bit 'sum' value
    Carry_out <= (A_in AND B_in AND (NOT Carry_in)) or ((NOT A_in) AND B_in AND Carry_in) or
(A_in AND (NOT B_in) AND Carry_in); --find carry bit

end Behavioral;


--Below is the 4-bit adder implementation that utilized 4 of the single-bit adders designed above
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Adder4 is
    Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
           B : in STD_LOGIC_VECTOR(3 downto 0);
           Cin : in std_logic;
           Sum : out STD_LOGIC_VECTOR(3 downto 0);
           Cout : out STD_LOGIC);
end Adder4;

architecture behavioral of Adder4 is

component SingleAdder is
Port ( A_in : in STD_LOGIC;
           B_in : in STD_LOGIC;
           Carry_in : in STD_LOGIC;
           Sum_out : out STD_LOGIC;
           Carry_out : out STD_LOGIC);
     end component;
```

```
signal temp0,temp1,temp2 : STD_LOGIC;

begin
Add0: SingleAdder port map(A(0),B(0),Cin,Sum(0),temp0);
Add1: SingleAdder port map(A(1),B(1),temp0,Sum(1),temp1);
Add2: SingleAdder port map(A(2),B(2),temp1,Sum(2),temp2);
Add3: SingleAdder port map(A(3),B(3),temp2,Sum(3),Cout);
end behavioral;
```

For the second part of the lab, a 4-bit adder with subtraction capabilities was required to be designed using the SingleAdder components. This required additional digital logic to create the 2's complement of one operand and add it to the other operand. To conduct this logic, I used a "select" signal (named 'sel') to determine if addition or subtraction was requested. This select signal is connected to the carry in bit of the LSB adder, and each bit of the "B" operand via 4 XOR gates. If sel = 1, the XOR gates will invert each individual bit of the "B" operand and add 1 to the LSB, resulting with the 2's complement. Then, when the adder operation is completed, the result will be given in 2's complement notation.

4-bit adder with subtraction capabilities

```
--Designed by Alex Lukens for ECE 742 Summer 2020 Class
--4 bit adder with subtraction capabilities (using 4x SingleAdder single-bit adder components from previous design)
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AdderSubtracter is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           sel : in STD_LOGIC;
           Sum : out STD_LOGIC_VECTOR (3 downto 0);
           Cout : out STD_LOGIC);
end AdderSubtracter;

architecture Behavioral of AdderSubtracter is

component SingleAdder is
Port ( A_in : in STD_LOGIC;
           B_in : in STD_LOGIC;
           Carry_in : in STD_LOGIC;
           Sum_out : out STD_LOGIC;
           Carry_out : out STD_LOGIC);
```

```
    end component;

signal temp0,temp1,temp2 : STD_LOGIC;
signal B_New : STD_LOGIC_VECTOR(3 downto 0);

begin
B_New(0) <= B(0) XOR sel; --use select signal to create 1's complement of 'B' operand
B_New(1) <= B(1) XOR sel;
B_New(2) <= B(2) XOR sel;
B_New(3) <= B(3) XOR sel;

Add0: SingleAdder port map(A(0),B_New(0),sel,Sum(0),temp0); --add 1 to LSB using sel to create 2's
complement
Add1: SingleAdder port map(A(1),B_New(1),temp0,Sum(1),temp1);
Add2: SingleAdder port map(A(2),B_New(2),temp1,Sum(2),temp2);
Add3: SingleAdder port map(A(3),B_New(3),temp2,Sum(3),Cout);
end Behavioral;
```

The Digilent Arty A7 FPGA only has 4 switches available, which is not enough to individually represent each bit of the operands. To overcome this obstacle, I decided to instantiate the first operand "A" on the rising_edge of button #3 (when it is initially pushed down) and the second operand "B" on the falling_edge of button #3 (when the button is released). This allows the user to individually program both operands while using only 4 switches. The operands are then stored, and subtraction can be implemented on the operands by holding down button #0. The results are shown on the LEDs available on the FPGA (carry out bit represented by RGB_LED_0).

FPGA Implementation of 4-bit Adder

```
--Designed by Alex Lukens for ECE 742 Summer 2020 Class
--This FPGA implementation of a 4-bit adder takes the first operand from the 4 switches
--available on the Digilent Arty A7 when btn3 is on the RISING_EDGE
--and the second operand on the FALLING_EDGE
--Subtraction is implemented by holding down btn(0)
--result shown on LEDs and carry out shown on rgb_led0_blue
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity FPGAatt2 is
   Port ( sw : in STD_LOGIC_VECTOR (3 downto 0);
          btn : in STD_LOGIC_VECTOR (3 downto 0);
          led0_b : out STD_LOGIC;
```

```
        led : out STD_LOGIC_VECTOR (3 downto 0));
end FPGAatt2;

architecture Behavioral of FPGAatt2 is

component AdderSubtracter is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           sel : in STD_LOGIC;
           Sum : out STD_LOGIC_VECTOR (3 downto 0);
           Cout : out STD_LOGIC);
    end component;
signal op1,op2 : STD_LOGIC_VECTOR(3 downto 0);
signal carry1 : STD_LOGIC :='0'; --carry in is static 1 to disable subtraction mode
begin
process(btn(3))
    begin
    if(rising_edge(btn(3)))
    then op1<=sw; --set first operand on rising edge
    end if;
    if(falling_edge(btn(3)))
    then op2<=sw; --set second operand on falling edge
    end if;
end process;

A1 : AdderSubtracter port map(op1,op2,btn(0),led,carry1); --4-bit adder instance
led0_b<=carry1; --led0_blue designates carry out from adder
end Behavioral;
```

IV. Conclusion

This lab should be considered a complete success. Students were successfully able to implement a single-bit full adder in VHDL, and then utilize this component to design a more complicated 4-bit ripple adder circuit. Students then expanded on this design by enabling subtraction capabilities and conforming the 4-bit adder to be usable on the Digilent Arty A7 FPGA. Students will utilize the skills used in this laboratory to develop more complex circuits in VHDL and to increase their comprehension of digital logic circuits.

References

[1] Wikipedia contributors. (2020, June 14). Xilinx. In *Wikipedia, The Free Encyclopedia*. Retrieved 02:24, June 16, 2020, from https://en.wikipedia.org/w/index.php?title=Xilinx&oldid=962581047

[2] FPGA Lab Manuals - ECE 742