Abstract

The goal of this lab was to gain experience with HDL programming by designing a 4-bit multiplier and then programming it in VHDL. The multiplier would take two 4-bit numbers and multiply them through a series of full adders. The inputs for the first 4-bit number are switches 3 through 0 and the second number's inputs are switches 7 through 4. The output of the multiplication is LED 7 though 0. The lab ended with success in that the outputs displayed on the LED was the expected result of the multiplication of the inputs. We used a series of logic gates and full adders (see figure) to achieve the desired results.

Summary

The lab was first spent trying to implement the recommended simple circuit to gain an understanding of VHDL. This involved programming a circuit where the input would be inverted and displayed on an LED. We used this to answer some questions we had about inputs and outputs in VHDL and on architectures of the entity declaration. With the experience we gained, we were able to (relatively) smoothly implement the task, with the help of a framework provided by Dr. Aamodt.

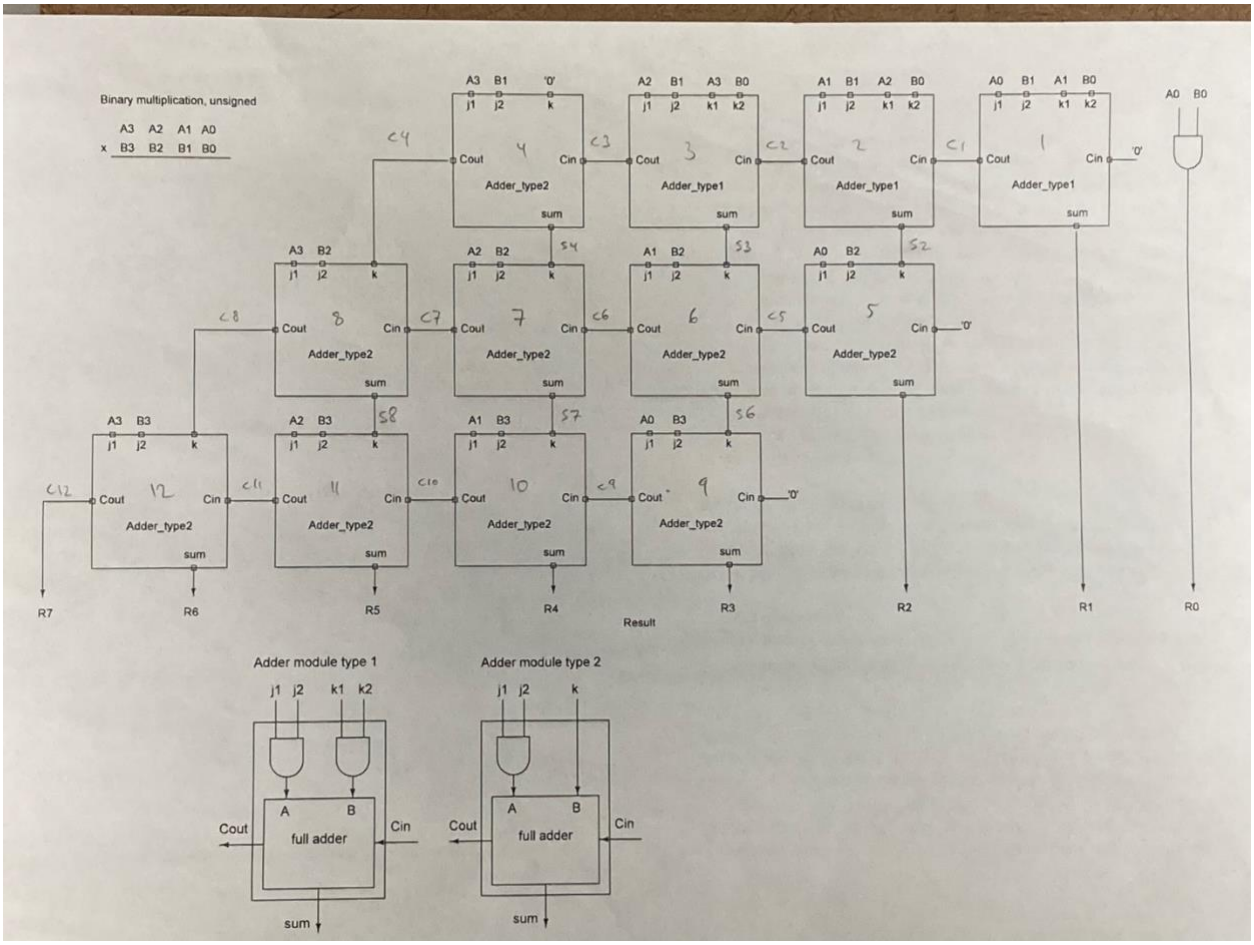

Figure 1

```
----------------------------------------------------------------------------------
-- Company: Walla Walla University
-- Engineer: Eric Walsh & Nicholas Zimmmeran
--
-- Create Date:    16:37:18 10/19/2021
-- Design Name:
-- Module Name:    Multiplier - Behavioral
-- Project Name:   Digital Design Lab 4
-- Target Devices: Artix-7 (Xilinx ISE)
-- Tool versions:
-- Description: Multiplies two 4-bit inputs and produces an 8-bit output. Switches are inputs and LEDs are outputs.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type1 is
    port(j1,j2,k1,k2,Cin : in std_logic;
         Cout, sum: out std_logic);
end adder_type1;

architecture Behavioral of adder_type1 is
    signal a, b, aXORb : std_logic;
begin
    a <= j1 and j2;
    b <= k1 and k2;

    --Full-adder
    aXORb <= a xor b;
    sum <= aXORb xor Cin;
    Cout <= (a and b) or (aXORb and Cin);
```

```
end Behavioral;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type2 is
    port(j1,j2,k,Cin : in std_logic;
         Cout, sum: out std_logic);
end adder_type2;

architecture Behavioral of adder_type2 is
    signal a, aXORk : std_logic;
begin
    a <= j1 and j2;

    --Full-adder
    aXORk <= a xor k;
    sum <= aXORk xor Cin;
    Cout <= (a and k) or (aXORk and Cin);

end Behavioral;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab4_solution_top is
    port( sw : in std_logic_vector (7 downto 0);
          led : out std_logic_vector (7 downto 0));
end lab4_solution_top;

architecture Behavioral of lab4_solution_top is

    component adder_type1
        port(j1,j2,k1,k2,Cin : in std_logic;
          Cout, sum: out std_logic);
    end component;
```

```
    component adder_type2
        port(j1,j2,k, Cin : in std_logic;
            Cout, sum: out std_logic);
    end component;

    signal A, B: std_logic_vector (3 downto 0);
    signal C: std_logic_vector (12 downto 1);
    signal S2, S3, S4, S6, S7, S8: std_logic;
begin
    A(3 downto 0) <= sw(7 downto 4);
    B(3 downto 0) <= sw(3 downto 0);
    led(0) <= A(0) and B(0);
    add1: adder_type1 port map(A(0),B(1),A(1),B(0),'0',C(1),led(1));
    add2: adder_type1 port map(A(1),B(1),A(2),B(0),C(1),C(2),S2);
    add3: adder_type1 port map(A(2),B(1),A(3),B(0),C(2),C(3),S3);
    add4: adder_type2 port map(A(3),B(1),'0',C(3),C(4),S4);
    add5: adder_type2 port map(A(0),B(2),S2,'0',C(5),led(2));
    add6: adder_type2 port map(A(1),B(2),S3,C(5),C(6),S6);
    add7: adder_type2 port map(A(2),B(2),S4,C(6),C(7),S7);
    add8: adder_type2 port map(A(3),B(2),C(4),C(7),C(8),S8);
    add9: adder_type2 port map(A(0),B(3),S6,'0',C(9),led(3));
    add10: adder_type2 port map(A(1),B(3),S7,C(9),C(10),led(4));
    add11: adder_type2 port map(A(2),B(3),S8,C(10),C(11),led(5));
    add12: adder_type2 port map(A(3),B(3),C(8),C(11),led(7),led(6));
end Behavioral;
```