EE2016-Microprocessors (Theory + Lab)
Experiment-3: Multiplier Design and LCD Interface to FPGA

1 Introduction

In experiments 1 and 2, we have designed some components of a microprocessor (and implemented on the FPGA board). In this experiment, we will design a multiplier (termed Wallace multiplier) in Verilog for a pair of four-bit unsigned numbers and perform a simulation. We will also show the results of Wallace multiplication on the LEDs on the FPGA board. In addition, we will learn how to write Verilog code to output a set of characters on the LCD (accessory to the FPGA board). An optional part for this experiment is to show the product of two unsigned numbers on the LCD.

2 Description of the Multiplier

The Wallace multiplier for a pair of 4-bit unsigned numbers is depicted in Figure 1. The elements in the partial product are shown in the upper half of the figure (labelled (a)). Note that the partial product elements are shown only to indicate what portions of the "multiplicand" are inputs to half-adders. The lower half (labelled (b)) shows which elements would become inputs to various full-adders (that add 3 bits at a time). The addition of (P3[3] S4 S3 S2 S1) to (C4 C3 C2 C1 C0) can be done by a 'dedicated' 5-bit adder or handled separately using one half-adder for S1 and C0 and 4 full-adders (each taking also the output carry bits from the stage on the 'right': for eg. a full-adder that takes S2 and C1 along with carry output of addition of S1 and C0).

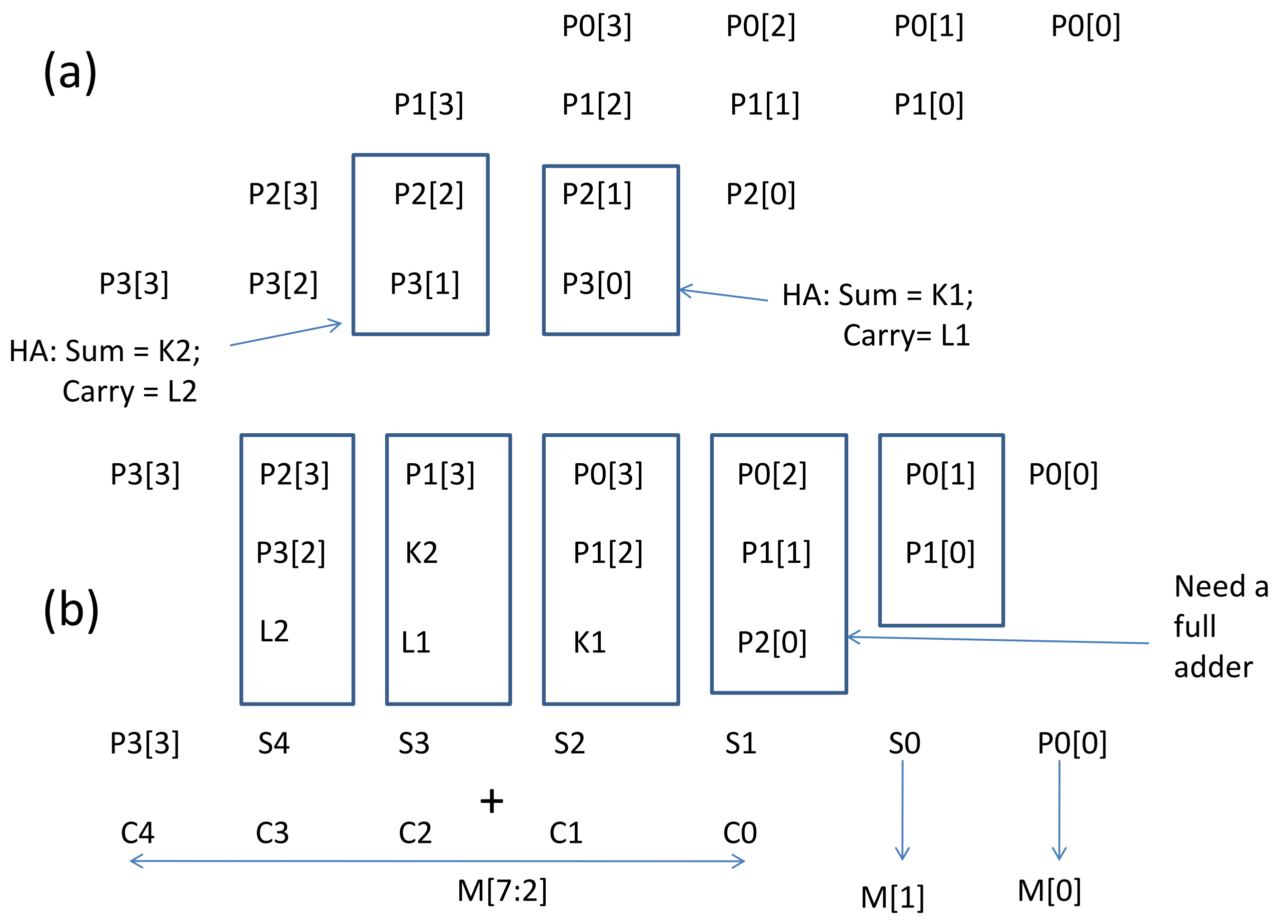


Figure 1: The Wallace Multiplier

3 Interfacing of LCD with the Artix FPGA Board

The LCD is interfaced with the FPGA chip as shown in Figure 2. The figure shows the I/O pins involved (which will be used in your Verilog module as well as in the design constraints file). In particular, the enable, register select (RS) and data pins need to be used. Register select (RS) setting helps to switch from command register (RS=0) to data register (RS=1). Note that the command register stores instructions (such as clear the display, move the cursor to a certain location etc.) given to the LCD. (An R/W line is additionally available and one may use it, in general, to specify read (from) or write (to) LCD. In our case, we will restrict to write and assume a default setting (of 0) for write.)

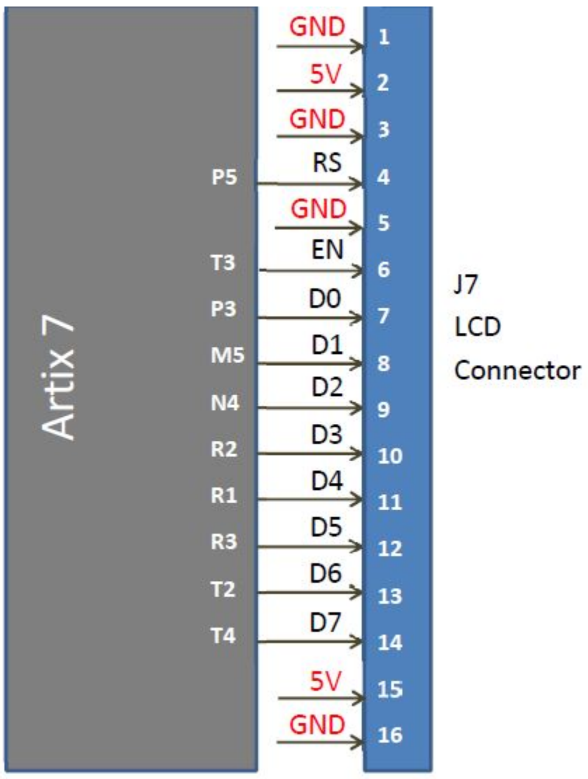


Figure 2: Interfacing the LCD

A few "control" commands need to be issued to the LCD before it can start printing any string. The commands are listed in Figure 3.

4 Tasks to Perform

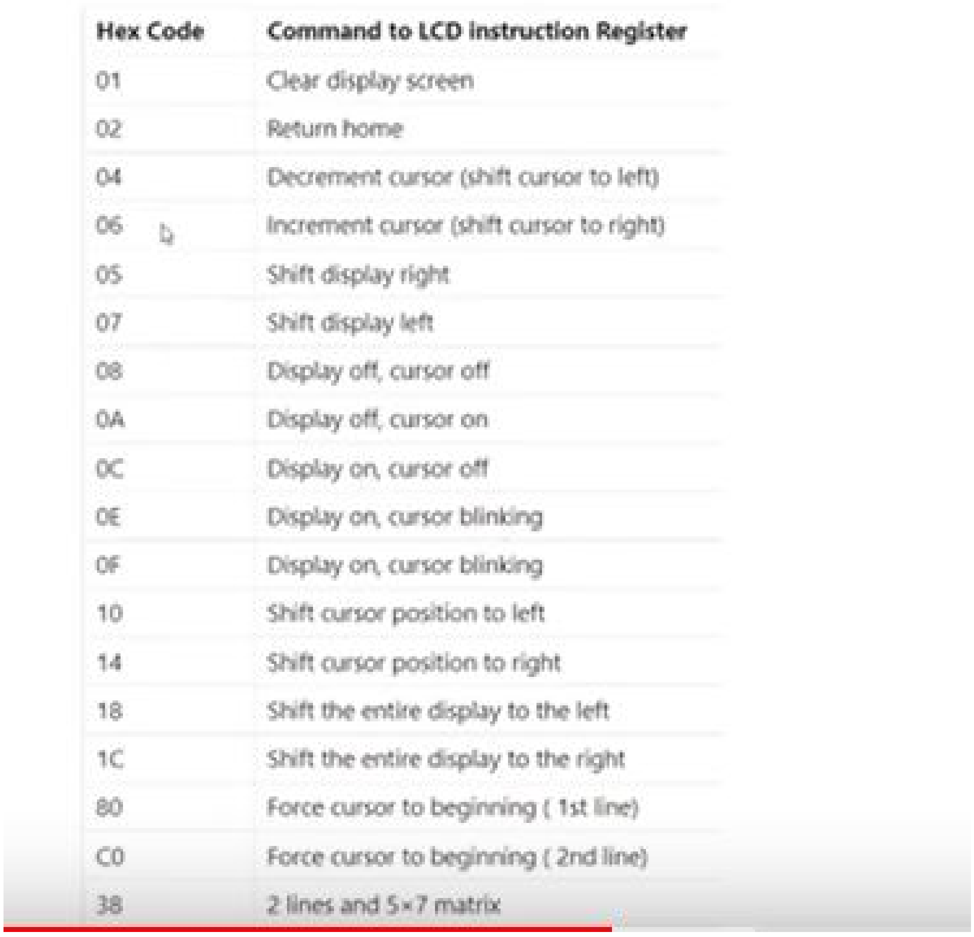
Task 1: Complete the Verilog module below to describe the functionality of the Wallace multiplier.

```
module unsigned_mult( m, a, b );
input [3:0]a,b;
output [7:0]m ;

wire [3:0]p0, p1, p2, p3;
wire s5, s4,s3,s2,s1,s0;
wire c5, c4,c3,c2,c1,c0;



// Write code for various blocks in the circuit
................................
```



Hex Code	Command to LCD Instruction Register
01	Clear display screen
02	Return home
04	Decrement cursor (shift cursor to left)
06	Increment cursor (shift cursor to right)
05	Shift display right
07	Shift display left
08	Display off, cursor off
0A	Display off, cursor on
0C	Display on, cursor off
0E	Display on, cursor blinking
0F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning (1st line)
C0	Force cursor to beginning (2nd line)
38	2 lines and 5×7 matrix

Figure 3: Commands to LCD instruction register

```
// Write code using assign (or other constructs) for getting outputs M[0] to M[7]


endmodule
```

Task 2: Write a Verilog testbench and simulate the Wallace multiplier.

Task 3: Hard code a pair of 4-bit numbers and show the results of Wallace multiplication on LEDs on the FPGA board. Write a suitable Xilinx design constraints file.

Task 4: We want to display the characters

1 2 3 4 5 6 7 8 9 A B C D E F G

on the LCD. Complete the following Verilog module.

```
 module lcd(in_Clk, lcd_rs, lcd_e, data);
    input in_Clk;
    output reg [7:0] data;
    output reg lcd_rs;
    output lcd_e;

    wire [7:0] command [0:4];
    reg [31:0] count=0;
    wire out_Clk;
    assign command [0] = ....;  // control signal to display on two lines
    assign command [1] = ....;   //  keep display on but cursor off
    assign command [2] = ....;  // increment the cursor
    assign command [3] = ....; // clear the display
    assign command [4] = ....;  // choose the second line

    clk_divider c0 (in_Clk, out_Clk);
    assign lcd_e = out_Clk;

    always@(posedge lcd_e) begin
       // Increment count

        case(count)
            1: begin lcd_rs = 0; data = ........; end  // fill in suitably
            2: begin lcd_rs = 0; data = ........; end
            3: begin lcd_rs = 0; data = ........; end
            4: begin lcd_rs = 0; data = ........; end
            5: begin lcd_rs = 0; data = ........; end
            6: begin lcd_rs = 1; data = ........; end // fill in hex corresp to ASCII for 1
            7: begin lcd_rs = 1; data = ........; end // 2
            8: begin lcd_rs = 1; data = ........; end  // 3
            9: begin lcd_rs = 1; data = ........; end // 4
            10: begin lcd_rs = 1; data = .......; end
            11: begin lcd_rs = 1; data = .......; end
            12: begin lcd_rs = 1; data = .......; end
            13: begin lcd_rs = 1; data = .......; end
            14: begin lcd_rs = 1; data = .......; end
            15: begin lcd_rs = 1; data = .......; end
            16: begin lcd_rs = 1; data = .......; end
            17: begin lcd_rs = 1; data = .......; end
            18: begin lcd_rs = 1; data = .......; end
            19: begin lcd_rs = 1; data = .......; end  // E
            20: begin lcd_rs = 1; data = .......; end  //  F
            21: begin lcd_rs = 1; data = .......; end // hex corresp to ASCII for  G

            default: begin lcd_rs = 0; data = .....; end
                         // fill in hex value  to return cursor to initial position
        endcase
    end
endmodule

module clk_divider (in_Clk,out_Clk);
 // Complete the clock divider module

endmodule
```

Task 5: Complete the following Xilinx design constraints (.xdc) file. Use Figure 3 to determine the entries in the .xdc file. Then use the (completed) Verilog LCD module along with the .xdc file to display the characters (listed in Task 4) on the LCD.

```
set_property -dict { PACKAGE_PIN P3 IOSTANDARD LVCMOS33 } [get_ports {data[7]}];
```

```
# fill in six more lines - identify the appropriate package pins


# fill in the package pins for Enable, RS and input clk
set_property -dict { PACKAGE_PIN .... IOSTANDARD LVCMOS33 } [get_ports {lcd_e}];
set_property -dict { PACKAGE_PIN .... IOSTANDARD LVCMOS33 } [get_ports {lcd_rs}];
set_property -dict { PACKAGE_PIN .... IOSTANDARD LVCMOS33 } [get_ports { in_Clk }];
```

Task 6 (Bonus Marks: 2; this task is optional): Show the operation of the Wallace multiplier using the LCD for various pairs of 4-bit numbers (input through the sliding switches). Try to print "Product is =" on the first line and the actual product on the second line of the LCD.

Please Note: The LCD unit is provided as an accessory to the FPGA board. Please connect the LCD unit gently to the FPGA board (and detach carefully once the experiment is done).

5 Report

Submit a report on the experiment on Moodle (within a week of this experiment). One report per group (with the names of the group members) is sufficient. The report should contain details of the solution (including the code), snapshots of simulation and your observations and experience (in programming, debugging etc.). Please note that reports that closely match those of other groups will be penalized.