EENG 2131 - Lab 3

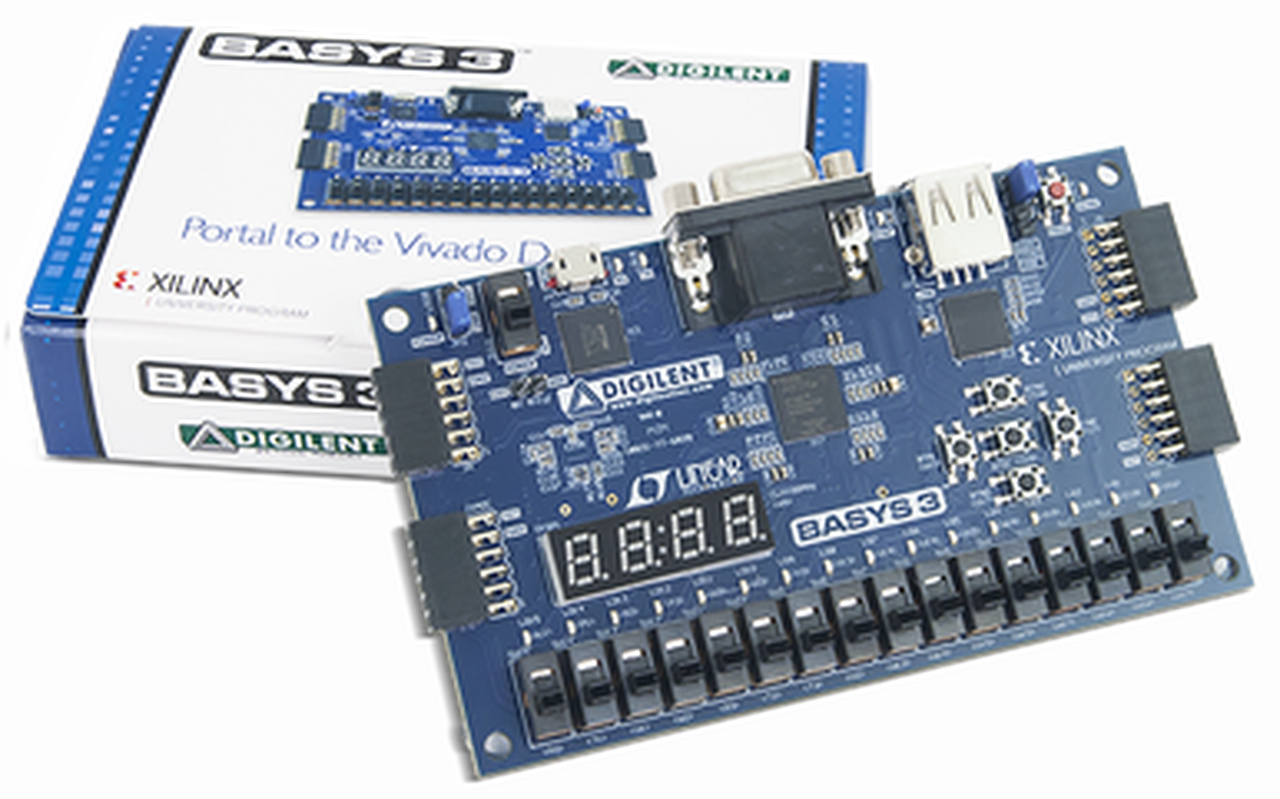
Verilog Synthesis and FPGA dev boards

Antonio Veguilla Hernandez

Student ID: P000166790

# Part 0: FPGA check out

The instructor will pass out copies of the Basys3 FPGA dev board. **Confirm that the instructor has recorded the serial number of your board in the tracking spreadsheet.** You’ll need to return the FPGA dev board at the end of the semester, but until then, keep it safe and feel free to use it outside of lab time to work on lab work, or your own experiments and explorations.



# Part 1: Get to blinky

For this part, we will get the FPGA board connected to your Vivado software, and confirm that everything is configured properly to blink an LED at a configurable rate

Follow the instructions here: <https://digilent.com/reference/vivado/getting_started/2018.2>

We are using the Basys3 FPGA dev board.

**When you have made modifications to the constraint file, have the instructor double-check your changes.**

Once you have loaded the bitstream into your Basys3 board, confirm that the LED is flashing as expected. What rate is it supposed to be? How might you adjust the rate of flashing?

**Once you have your Basys3 flashing the LED, show it to your instructor before you move on.**

# Part 2 – Multi clock outputs

For this part, modify your part 1 code to product four different clock outputs, which must be visible on the first four LEDs. You will likely need to modify your constraint file to connect the LEDs to your top-level module. Try to think of way to use the 100 MHz clock input and toggle the LED outputs at the requested rates:

LED 0 must output at 1 HZ.

LED 1 must output at 4 HZ.

LED 2 must output at 5 HZ.

LED 3 must output at 27 HZ.

**Once you have your Basys3 flashing the LED, show it to your instructor before you move on, and also explain how your module works.**

**Part1 & Part 2 code**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company: Dunwoody College of Technology

// Engineer: Antonio Veguilla Hernandez

//

// Create Date: 09/15/2021 12:16:47 PM

// Design Name:

// Module Name: Blinky

// Project Name: Lab3\_Part1 & Part2

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module Blinky(

input clk,

output led0, //declaring led0 from the BASYS 3 board to be used as an output

output led1, //declaring led1 from the BASYS 3 board to be used as an output

output led2, //declaring led2 from the BASYS 3 board to be used as an output

output led3 //declaring led3 from the BASYS 3 board to be used as an output

);

reg [25:0] count0 = 0; //creating count 0 with a bit count of 0-25 = 2^26 = 67108864

reg [24:0] count1 = 0; //creating count 1 with a bit count of 0-24 = 2^25 = 33554432

reg [24:0] count2 = 0; //creating count 2 with a bit count of 0-24 = 2^25 = 33554432

reg [24:0] count3 = 0; //creating count 3 with a bit count of 0-24 = 2^25 = 33554432

reg led0reg = 0; //creating led0reg and initializing it by setting to 0.

reg led1reg = 0; //creating led1reg and initializing it by setting to 0.

reg led2reg = 0; //creating led2reg and initializing it by setting to 0.

reg led3reg = 0; //creating led3reg and initializing it by setting to 0.

assign led0 = led0reg; //This assign the vlaue at ledreg0 into led0;

assign led1 = led1reg; //This assign the vlaue at ledreg1 into led1;

assign led2 = led2reg; //This assign the vlaue at ledreg2 into led2;

assign led3 = led3reg; //This assign the vlaue at ledreg3 into led3;

always @ (posedge(clk))

begin

if(count0 == 50000000) //1Hz blink

begin

count0 <= 0; //This will reset the variable count0 to 0.

led0reg <= !led0reg; //This will toggle the value located at led0reg.

end

else

begin

count0 <= count0 + 1; //This keep the counter counting

end

if(count1 == 12500000) //4Hz Blink

begin

count1 <= 0;//This will reset the variable count1 to 0.

led1reg <= !led1reg; //This will toggle the value located at led1reg.

end

else

begin

count1 <= count1 + 1; //This keep the counter counting

end

if(count2 == 10000000) //5Hz Blink

begin

count2 <= 0;//This will reset the variable count2 to 0.

led2reg <= !led2reg; //This will toggle the value located at led2reg.

end

else

begin

count2 <= count2 + 1; //This keep the counter counting

end

if(count3 == 7142857) //7Hz Blink

begin

count3 <= 0;//This will reset the variable count3 to 0.

led3reg <= !led3reg; //This will toggle the value located at led3reg.

end

else

begin

count3 <= count3 + 1; //This keep the counter counting

end

end

endmodule

# Part 3 – 4-bit ripple carry adder

This part requires you to use the Basys3 to implement a 4-bit ripple carry adder, similar to the final part of the previous lab. In fact, you can use your code from last week as a starting point here.

The inputs and outputs of the 4-bit adder must be connected as follows:

Carry input: Switch 0  
First number input: Switches 4:1  
Second number input: Switches 8:5  
Sum output: LEDs 14:11  
Carry output: LED 15

Additionally, the current state of the slide switch must be reflected on the corresponding LED above each switch.

**Once you have your Basys3 loaded with your working 4-bit ripple carry adder, show it to your instructor before you move on, and also explain how your module works.**

**Part 3\_4 code**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company: Dunwoody College of Technology

// Engineer: Antonio Veguilla Hernandez

//

// Create Date: 09/19/2021 06:58:45 PM

// Design Name:

// Module Name: Lab3\_Part3\_4

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module Board\_interface(

input sw0,

input sw1,

input sw2,

input sw3,

input sw4,

input sw5,

input sw6,

input sw7,

input sw8,

output led0,

output led1,

output led2,

output led3,

output led4,

output led5,

output led6,

output led7,

output led8,

output led11,

output led12,

output led13,

output led14,

output led15

);

Lab3\_Part3\_4 dut(sum, cout, {sw4,sw3,sw2,sw1}, {sw8,sw7,sw6,sw5}, {sw0});

wire [3:0]sum;

wire cout;

assign led0 = sw0; //Carry input

assign led1 = sw1; //X LSB 1st bit

assign led2 = sw2; //X 2nd bit

assign led3 = sw3; //X 3rd bit

assign led4 = sw4; //x 4th bit

assign led5 = sw5; //Y LSB 1st bit

assign led6 = sw6; //Y 2nd bit

assign led7 = sw7; //Y 3rd bit

assign led8 = sw8; //Y 4th bit

assign led11 = sum[0];

assign led12 = sum[1];

assign led13 = sum[2];

assign led14 = sum[3];

assign led15 = cout;

endmodule

module FullAdder(sum, cout, x, y, cin);

input x;

input y;

input cin;

output cout;

output sum;

wire e, f, g;

xor #10 (e, x, y);

xor #10 (sum, e, cin);

and #10 (f, e, cin);

and #10 (g, x, y);

or #10 (cout, f, g);

endmodule

module Lab3\_Part3\_4 (sum, cout, x, y, cin);

output [3:0] sum;

output cout;

input [3:0] x, y;

input cin;

wire [3:1] c;

FullAdder FA0 (sum[0], c[1], x[0], y[0], cin); //sum, cout, x, y, cin

FullAdder FA1 (sum[1], c[2], x[1], y[1], c[1]);

FullAdder FA2 (sum[2], c[3], x[2], y[2], c[2]);

FullAdder FA3 (sum[3], cout, x[3], y[3], c[3]);

endmodule