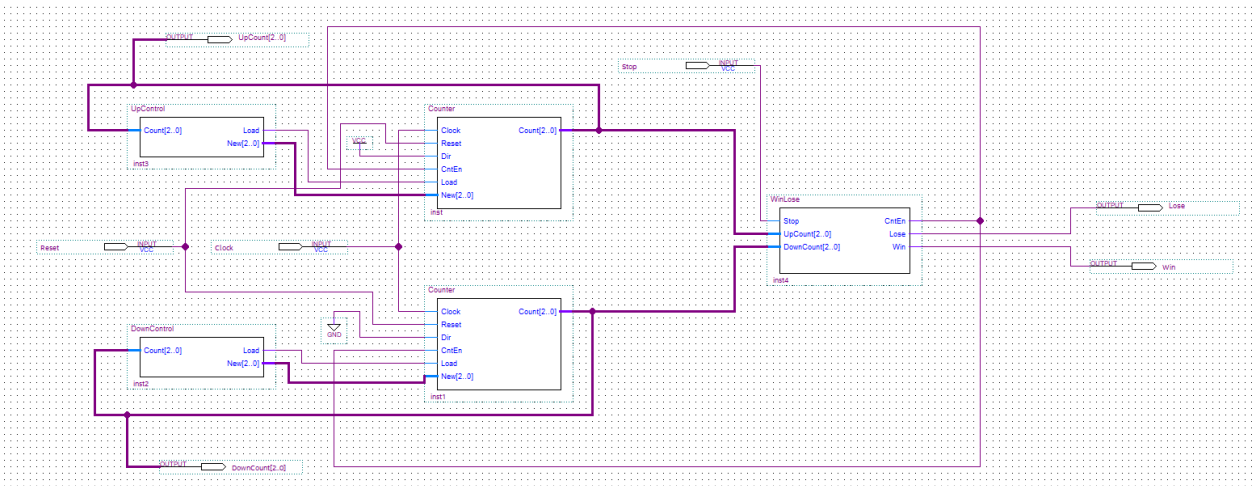
Tushar Shrivastav Luca Chierotti
COEN21L
2/25/22

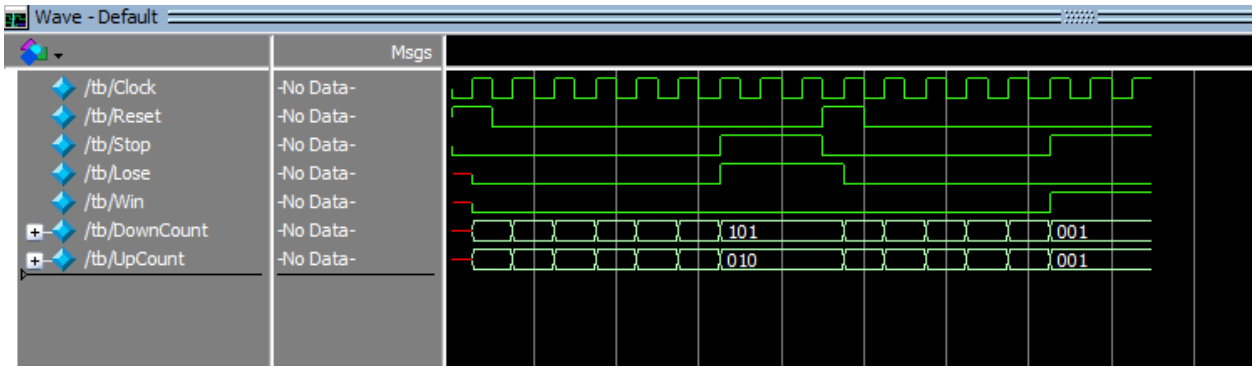
Lab 7 Report

1. Introduction

In this lab we are designing a game that consists of two counters, one that counts up (1 to 5) and one that counts down (5 to 1). The users are tasked to press a button and to win you must release the button at the same number, otherwise you lose. We wrote the logic for the game in Verilog in our Pre-Lab (Code is Below), and the final Circuit for the game is below too.

Circuit Diagram





Verilog

Prelab

```
module Counter(
        input Clock,
        input Reset,
```

```
		input Dir,
		input CntEn,
		input Load,
		input [2:0] New,
		output reg [2:0] Count);

		always @(posedge Clock) begin
			if(Reset == 1)
				Count <= 3'b001;
			else begin
				if(CntEn == 1) begin
					if(Load == 1)
						Count <= New;
					else begin
						if (Dir == 1)
							Count <= Count + 1;
						else
							Count <= Count - 1;
					end
				end
			end
		end
	endmodule

	module UpControl(
		input [2:0] Count,
		output Load,
		output [2:0] New);

		/* While all other "?" symbols in this file need to be replaced with something,
			the one in the below "assign Load" line should stay, as it's part of the
syntax
			of the completed line. The statement can be interpreted like this: assign
binary
			1 to Load if Count is binary 101, otherwise assign binary 0 to Load. */
		assign Load = (Count == 3'b101) ? 1'b1 : 1'b0;
		assign New = (Count == 3'b001);
	endmodule

	module DownControl(
	  input [2:0] Count,
	  output Load,
	  output [2:0] New);
```

```
  assign Load = (Count == 3'b001) ? 1'b1 : 1'b0;
  assign New = (Count == 3'b101);
endmodule

module WinLose(
        input Stop,
        input [2:0] UpCount,
        input [2:0] DownCount,
        output reg CntEn,
        output reg Lose,
        output reg Win);
        // There should be more than one line in this block (I had 11 for reference).
        always @(UpCount or DownCount) begin
                //Set Win and Lose to 0
                Win <= 0;
                Lose<= 0;

                if(Stop) //assert stop
                        begin
                                CntEn = 0; //stop counting
                                if(UpCount == DownCount) //win case
                                        Win<=1;
                                else
                                        Lose<=1; //lose case
                        end
                else
                        CntEn = 1; //continue counting
        end
endmodule
```

Final Lab Verilog

```
module Counter(
        input Clock,
        input Reset,
        input Dir,
        input CntEn,
        input Load,
        input [2:0] New,
        output reg [2:0] Count);

        always @(posedge Clock) begin
                if(Reset == 1)
                        Count <= 3'b001;
```

```
                else begin
                        if(CntEn == 1) begin
                                if(Load == 1)
                                        Count <= New;
                                else begin
                                        if (Dir == 1)
                                                Count <= Count + 1;
                                        else
                                                Count <= Count - 1;
                                end
                        end
                end
        end
endmodule

module UpControl(
        input [2:0] Count,
        output Load,
        output [2:0] New);

        /* While all other "?" symbols in this file need to be replaced with something,
                the one in the below "assign Load" line should stay, as it's part of the
syntax
                of the completed line. The statement can be interpreted like this: assign
binary
                1 to Load if Count is binary 101, otherwise assign binary 0 to Load. */
        assign Load = (Count == 3'b101) ? 1'b1 : 1'b0;
        assign New = 3'b001;
endmodule

module DownControl(
  input [2:0] Count,
  output Load,
  output [2:0] New);

  assign Load = (Count == 3'b001) ? 1'b1 : 1'b0;
  assign New = 3'b101;
endmodule

module WinLose(
        input Stop,
        input [2:0] UpCount,
        input [2:0] DownCount,
        output reg CntEn,
```

```
        output reg Lose,
        output reg Win);
        // There should be more than one line in this block (I had 11 for reference).
        always @(UpCount or DownCount) begin
                //Set Win and Lose to 0
                Win <= 0;
                Lose<= 0;

                if(Stop) //assert stop
                        begin
                                CntEn = 0; //stop counting
                                if(UpCount == DownCount) //win case
                                        Win<=1;
                                else
                                        Lose<=1; //lose case
                        end
                else
                        CntEn = 1; //continue counting
        end
endmodule
```

2. If you were to change your design to have the counters count from 2 to 6 (or 6 to 2) instead of only 1 to 5, list all the things you would need to change.

Firstly the bus dimensions would not change since 6 = 110 (which is still a three bit number).

In UpControl we would need to make the following changes
assign Load = (Count == 3'110) ? 1'b1 : 1'b0;
assign New = 3'010;

In DownControl
assign Load = (Count == 3'010) ? 1'b1 : 1'b0;
assign New = 3'110;

In the Counter line 12, when reset = 1
Change to Count <= 3'b010;