Lab 5
3300L.E01

BCD Up/Down Counter on 7-Segment Display

Group U
Justin Herrera - Pandarb004
Nathan Rahbar - 9athan

Design

Constraints

For Lab 4, our team implemented the 7 segment displays on the Nexys A7, which were controlled through switches. We had a constraints file in which we initialized our switches, buttons, LEDs, and 7 segment displays.

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK}];

## Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }];
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }];
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }];
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }];
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }];

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }];
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }];
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { LED[2] }];
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { LED[3] }];
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }];
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }];
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }];
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }];
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { LED[8] }];
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { LED[9] }];
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { LED[10] }];
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { LED[11] }];
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { LED[12] }];
```

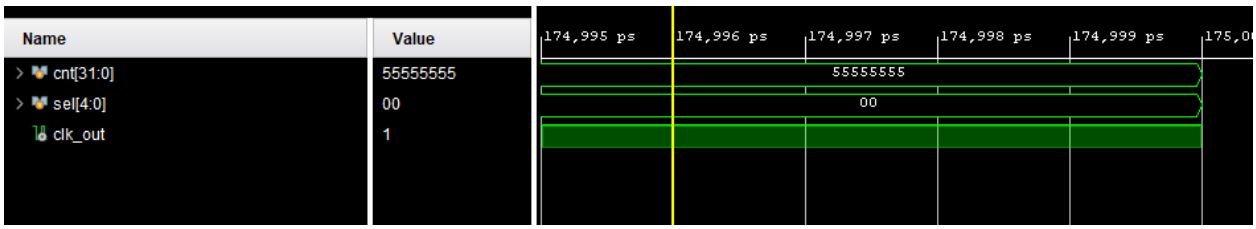
```
## 7-Segment Display (Critical Updates)
# Segment lines
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { SEG[0] }];
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { SEG[1] }];
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { SEG[2] }];
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { SEG[3] }];
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { SEG[4] }];
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { SEG[5] }];
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { SEG[6] }];

# Anode lines - NOW CONTROLLING ALL DIGITS
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; # Rightmost digit
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }];
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }];
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }];
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }];
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }];
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }];
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; # Leftmost digit

## Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTN0 }];
set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTN1 }];
```

Driver Module

```
`timescale 1ns / 1ps

module top_lab5 (
    input           CLK,
    input           BTN0,
    input           BTN1,
    input  [4:0]    SW,
    output [12:0]   LED,
    output [1:0]    AN,
    output [6:0]    SEG
);
    wire [31:0]     cnt;
    wire            clk_out;
    wire [3:0]      lsb0;
    wire [3:0]      lsb1;

    clock_divider u_div (
        .clk (CLK),
        .rst_n (!BTN0),
        .cnt (cnt)
    );

    mux32x1 u_mux (
        .cnt (cnt),
        .sel (SW),
        .clk_out (clk_out)
    );

    bcd_up_down_counter u_bcd (
        .clk (clk_out),
        .rst_n (!BTN0),
        .dir (BTN1),
        .lsb0 (lsb0),
        .lsb1 (lsb1)
    );

    seg7_scan u_scan (
        .clk (CLK),
        .rst_n (!BTN0),
        .lsb0 (lsb0),
```

```
seg7_scan u_scan (
    .clk (CLK),
    .rst_n (!BTN0),
    .lsb0 (lsb0),
    .lsb1 (lsb1),
    .seg (SEG),
    .an (AN)
);



assign LED[4:0]     =   SW;
assign LED[8:5]     =   lsb0;
assign LED[12:9]    =   lsb1;
```

Testbench

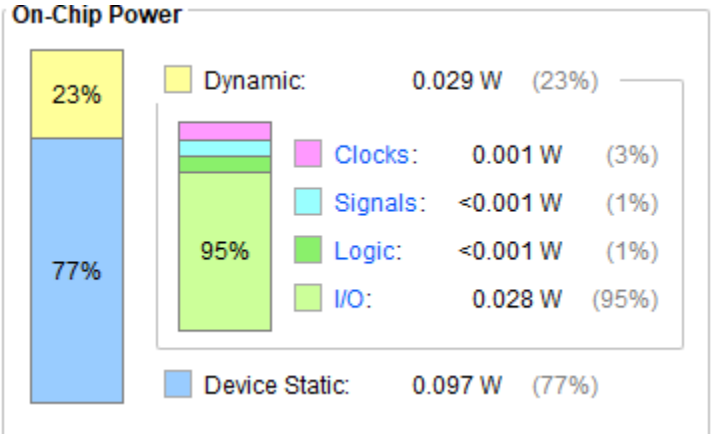


Test Bench Code

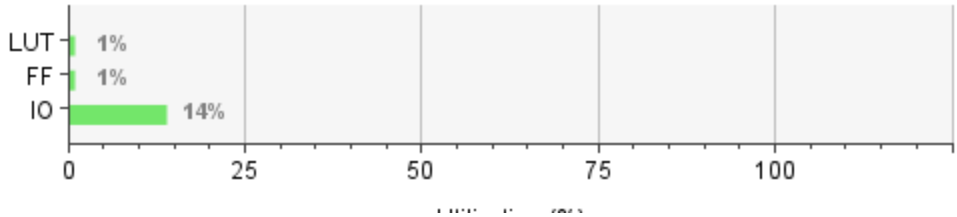
```
`timescale 1ns / 1ps

module bcd_up_down_counter_tb();
    reg clk = 0;
    reg rst_n;
    reg dir;
    wire [3:0] lsb0, lsb1;

    bcd_up_down_counter tb (
        .clk(clk),
        .rst_n(rst_n),
        .dir(dir),
        .lsb0(lsb0),
        .lsb1(lsb1)
    );

    always #5 clk = ~clk;

    initial begin
        rst_n = 0; dir = 1; #20;
        rst_n = 1; #100;

        dir = 0; #100;
        $finish;
    end
endmodule
```

Our test bench is used to test the driver module.

Implementation



Resource	Utilization	Available	Utilization %
LUT	25	63400	0.04
FF	56	126800	0.04
IO	30	210	14.29



In our implementation, we were able to analyze utilization and see our Flip Flop and LUT counts. This is a simple design, so utilization was not particularly high. We do see 95% of power utilization comes from I/O, which makes sense because we use all switches.

Contributions

Justin - Testbench, Report. - 50%
Nathan - Driver Module, Constraints. - 50%

Demo

https://youtu.be/rcLzSD2aFzM