ECE 3300L.01 - Lab 5

BCD Up/Down Counter on 7-Segment Display

Professor Mohamed Aly
Group Q: Kevin Tang(015429622) &
Jared Mocling(015215057)

July 21th, 2025

Objective:

- Design a two-digit BCD up/down counter in Verilog-HDL.
- Implement a clock-divider with a 32-bit counter + 32×1 Mux and 5 SW speed select.
- Control counting direction via BTN1 (Up/Down) and reset via BTN0.
- Drive a dual-digit 7-segment display using multiplexing.
- Deploy on Nexys A7: simulate, synthesize, program, and test.

Design:

Clock divider:

```
module clock_divider(
    input clk,
    output reg [31:0] cnt = 0
);
    always @(posedge clk)
        cnt <= cnt + 1;
endmodule
```

Mux32x1:

```
module mux32x1(
    input [31:0] cnt,
    input [4:0] sel,
    output clk_out
);
    assign clk_out = cnt[sel];
endmodule
```

BCD Up/Down Counter:

```
module bcd_up_down_counter(
    input clk_out,
    input rst_n,
    input dir, // 1 for up, 0 for down
    output reg [3:0] digit1, // tens
    output reg [3:0] digit0  // units
);
    always @(posedge clk_out or negedge rst_n) begin
        if (!rst_n) begin
            digit0 <= 0;
            digit1 <= 0;
        end else if (dir) begin
            // Count Up
            if (digit0 == 9) begin
                digit0 <= 0;
                if (digit1 == 9)
                    digit1 <= 0;
                else
                    digit1 <= digit1 + 1;
            end else begin
                digit0 <= digit0 + 1;
            end
        end else begin
            // Count Down
            if (digit0 == 0) begin
                digit0 <= 9;
                if (digit1 == 0)
                    digit1 <= 9;
                else
                    digit1 <= digit1 - 1;
            end else begin
                digit0 <= digit0 - 1;
            end
        end
    end
endmodule
```

7 Segment Display:

```
module seg7_scan(
    input clk,
    input [3:0] digit1,
    input [3:0] digit0,
    output reg [7:0] AN,
    output reg [6:0] SEG
);

    reg [15:0] refresh_counter = 0;
    wire sel;
    reg [3:0] current_digit;

    always @(posedge clk) begin
        refresh_counter <= refresh_counter + 1;
    end

    assign sel = refresh_counter[15];

    always @(*) begin
        case (sel)
            1'b0: begin
                AN <= 8'b11111110; // Activate digit0
                SEG <= seg_decoder(digit0);
            end
            1'b1: begin
                AN <= 8'b11111101; // Activate digit1
                SEG <= seg_decoder(digit1);
            end
        endcase
    end

    function [6:0] seg_decoder;
        input [3:0] num;
        case (num)
            4'd0: seg_decoder=7'b0000001;
            4'd1: seg_decoder=7'b1001111;
            4'd2: seg_decoder=7'b0010010;
            4'd3: seg_decoder=7'b0000110;
            4'd4: seg_decoder=7'b1001100;
```

```
            4'd5: seg_decoder=7'b0100100;
            4'd6: seg_decoder=7'b0100000;
            4'd7: seg_decoder=7'b0001111;
            4'd8: seg_decoder=7'b0000000;
            4'd9: seg_decoder=7'b0001100;
            default: seg_decoder = 7'b1111111; // off
        endcase
    endfunction
endmodule
```

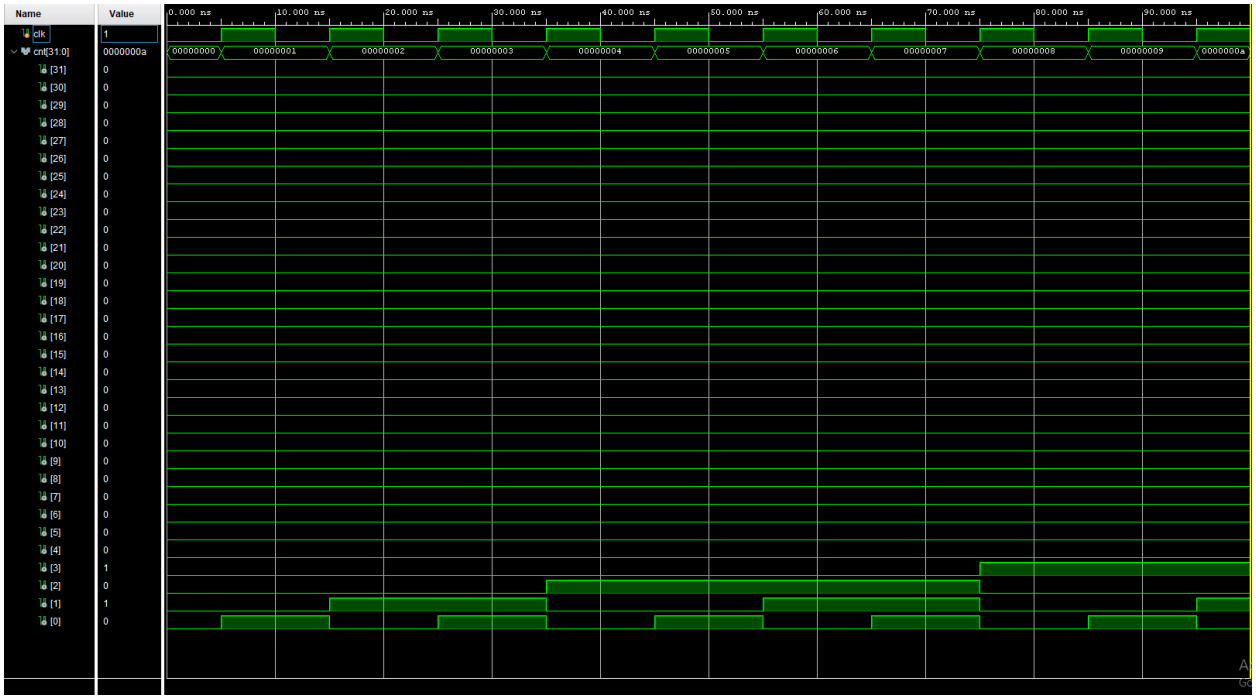
Top Module:

```
module top_lab5(
    input clk,
    input [4:0] SW,
    input [1:0] BTN,
    output [7:0] AN,
    output [6:0] SEG,
    output [12:0] LED
);

    wire [31:0] cnt;
    wire clk_out;
    wire [3:0] digit0, digit1;

    assign rst_n = ~BTN[0];

    //clock divider
    clock_divider cd (
        .clk(clk),
        .cnt(cnt)
    );

    // Mux
    mux32x1 mux (
        .cnt(cnt),
        .sel(SW),
        .clk_out(clk_out)
    );

    // BCD counter
```

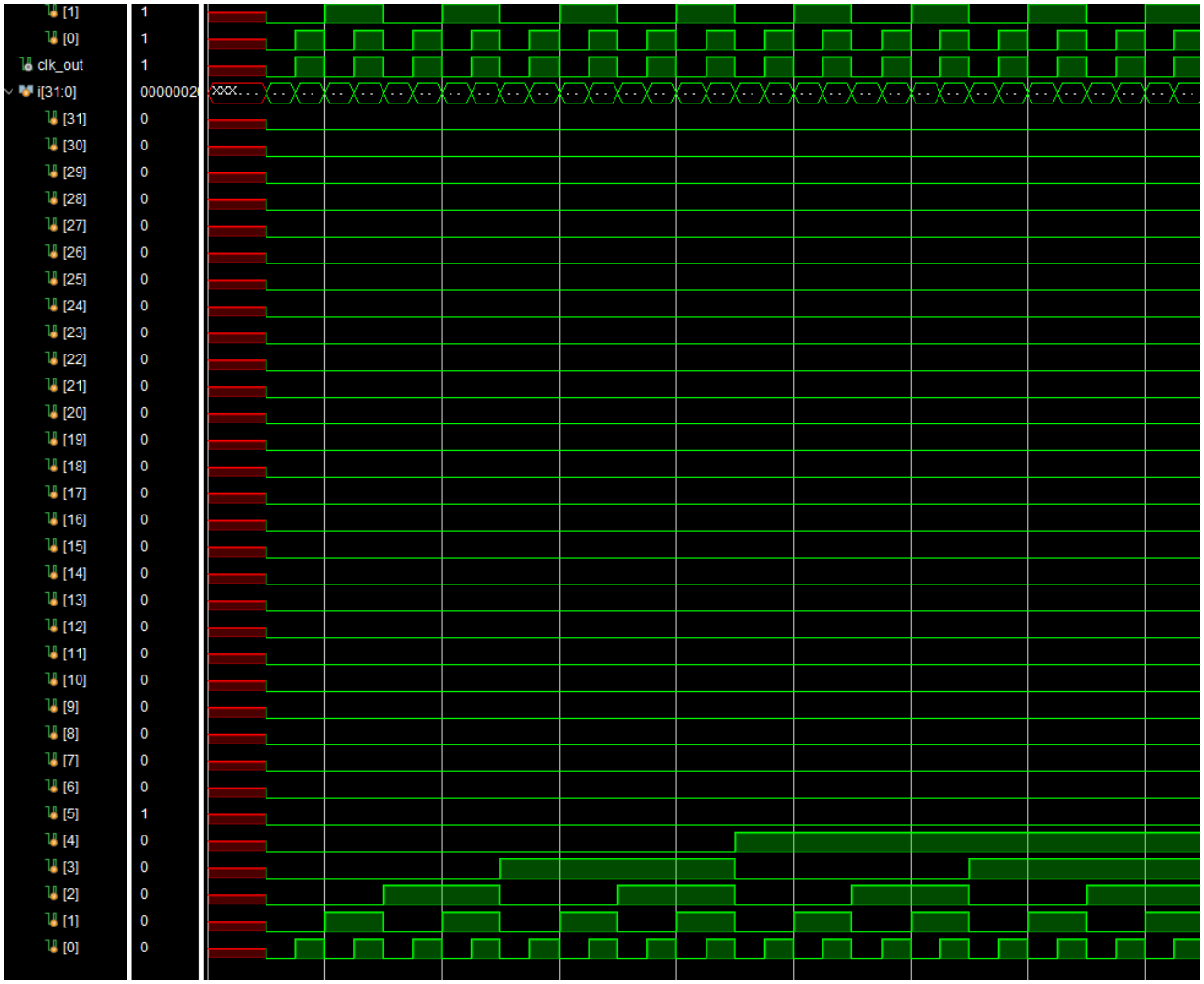
```
    bcd_up_down_counter bcd (
        .clk_out(clk_out),
        .rst_n(rst_n),
        .dir(BTN[1]),
        .digit0(digit0),
        .digit1(digit1)
    );

    // 7-segment scan
    seg7_scan scan (
        .clk(clk),
        .digit0(digit0),
        .digit1(digit1),
        .AN(AN),
        .SEG(SEG)
    );

    assign LED[4:0] = SW;
    assign LED[8:5] = digit0;
    assign LED[12:9] = digit1;

endmodule
```

Testbench Waveform:

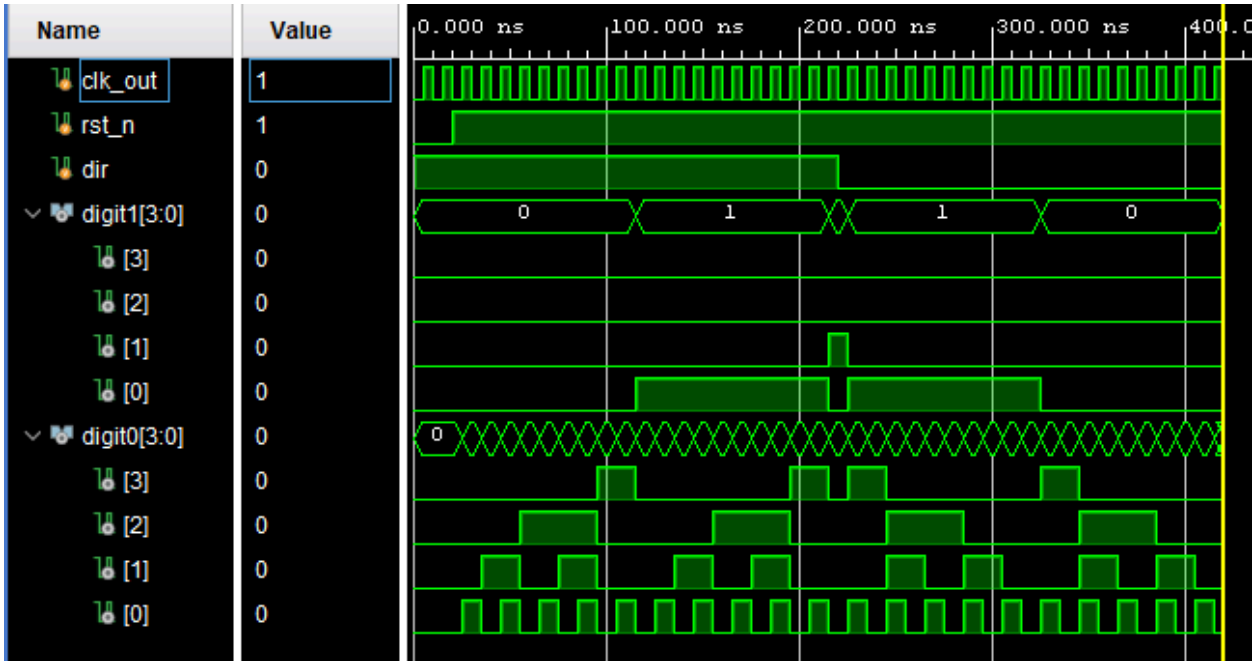
Clock Divider



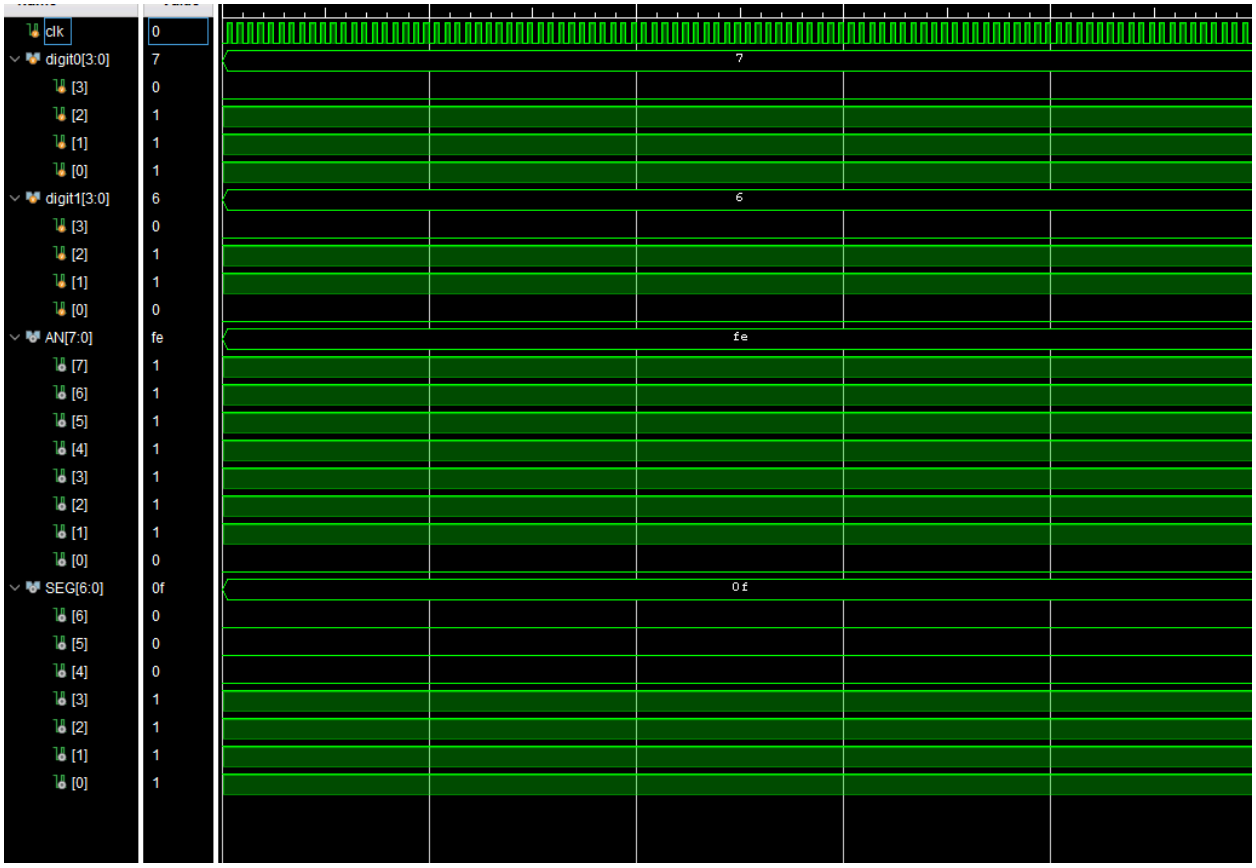
Mux 32x1


[1]
[0]
clk_out
i[31:0]
[31]
[30]
[29]
[28]
[27]
[26]
[25]
[24]
[23]
[22]
[21]
[20]
[19]
[18]
[17]
[16]
[15]
[14]
[13]
[12]
[11]
[10]
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]

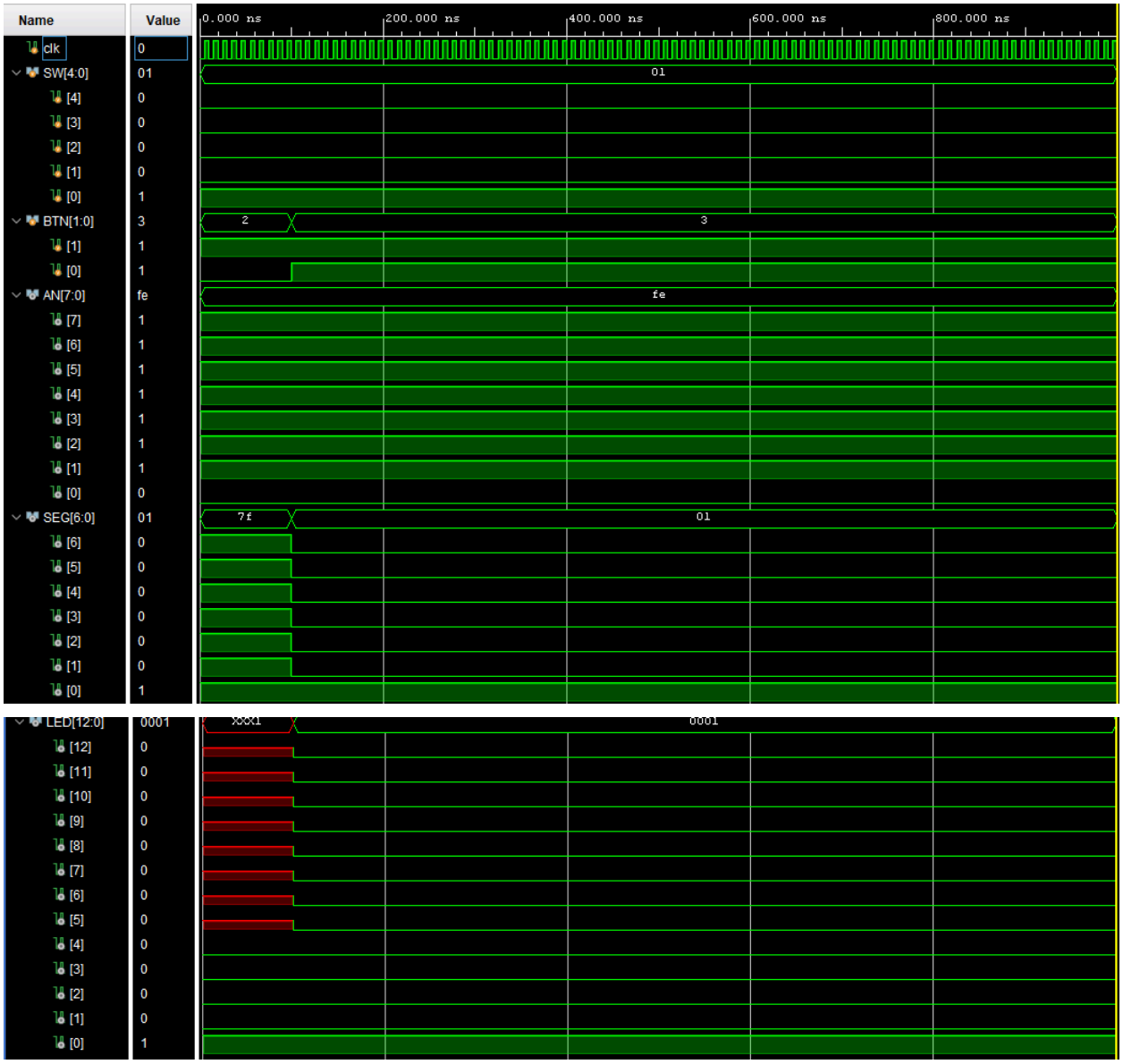
BCD Counter



7 Segment Display

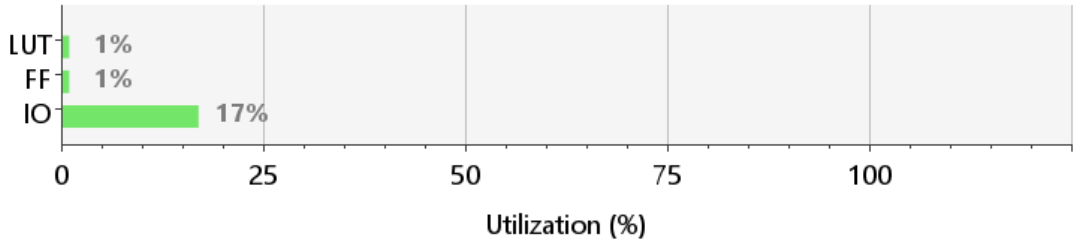


Top module


Name
Value
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
clk
0
SW[4:0]
01
[4]
0
[3]
0
[2]
0
[1]
0
[0]
1
BTN[1:0]
3
2
3
[1]
1
[0]
1
AN[7:0]
fe
fe
[7]
1
[6]
1
[5]
1
[4]
1
[3]
1
[2]
1
[1]
1
[0]
0
SEG[6:0]
01
7f
01
[6]
0
[5]
0
[4]
0
[3]
0
[2]
0
[1]
0
[0]
1
LED[12:0]
0001
XXX1
0001
[12]
0
[11]
0
[10]
0
[9]
0
[8]
0
[7]
0
[6]
0
[5]
0
[4]
0
[3]
0
[2]
0
[1]
0
[0]
1

Utilization/Implementation:

Resource	Utilization	Available	Utilization %
LUT	28	63400	0.04
FF	56	126800	0.04
IO	36	210	17.14

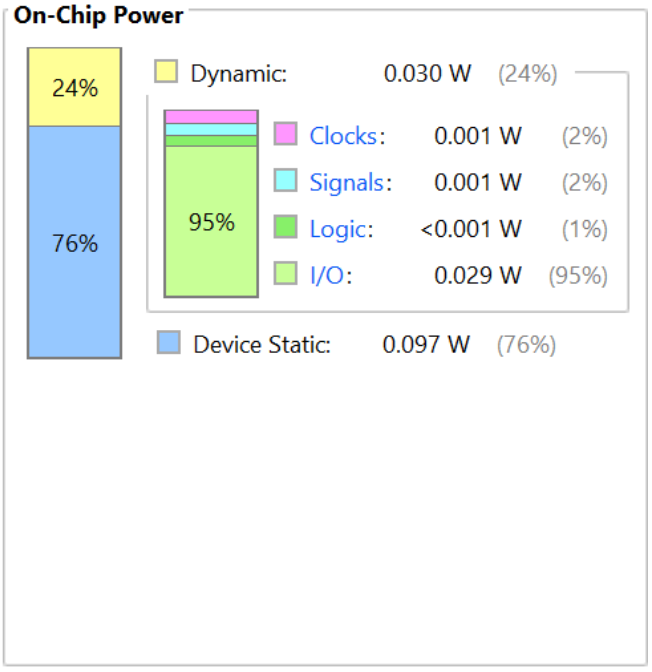


Power

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.127 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.6°C**
Thermal Margin:	59.4°C (12.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Time Summary:

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.974 ns	Worst Hold Slack (WHS):	0.324 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	48	Total Number of Endpoints:	48	Total Number of Endpoints:	49

All user specified timing constraints are met.

Contributions:

Jared Mocling (50%) - board demo, compiled code, report.
Kevin Tang (50%)- compiled code, simulation code, Synthesis reports, report