# MICROPROCESSOR LAB EXPERIMENT 2

Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

18 August 2024

## Introduction

FPGA board : Edge Artix 7 This experiment involves

- Simulating a half-adder using Xilinx Vivado and implementing on the FPGA board.
- Extending the half-adder design to a full-adder, simulating it and implementing on the FPGA board.
- Designing a 4-bit ripple-carry adder and implementing on the FPGA board.

## Xilinx Vivado

- We were introduced to Xilinx vivado , a software that is used to synthesize and analyse the hardware description designs.
- The procedures that we followed are ,
  - Create a project with source file as our Verilog code.
  - Add constraints to the ports that we have defined in the Verilog code.
  - Run the synthesis to check whether our Verilog code has any error in it.
  - We can open the Schematics to see the design that we have coded in Verilog.
  - After running synthesis , we can run simulation using our testbench and check for logical errors.
  - Once we confirm there is no logical/syntax error , we can run the implementation which basically implements our hardware description in the board which we have chosen while creating the project.
  - After the implementation is done , we can generate bitstream which is basically a file that contains the configuration information for an FPGA.
  - Once we successfully generate the bitstream , we can connect the target source and program it with the generated bitstream.
- In this report , we have included
  - Verilog code for module instantiation.
  - We have included both data flow as well as gate level modelling here.
  - The Schematics generated from Xilinx Vivado.
  - The Constraints file generated for FPGA.
  - The testbench and simulation generated
  - The analysis of reports obtained from Xilinx Vivado.

# Johnson Counter

## Data flow model

```
module decoder(input[2:0] cntr,output [7:0] led);
        reg [6:0] val;
        assign led={1'b1,~(val)};

        always@(cntr)
        begin
                case(cntr)
                3'd0 : val=7'b0111111;
                3'd1 : val=7'b0000110;
                3'd2 : val=7'b1011011;
                3'd3 : val=7'b1001111;
                3'd4 : val=7'b1100110;
                3'd5 : val=7'b1101101;
                3'd6 : val=7'b1111101;
                3'd7 : val=7'b0000111;
                endcase
        end
endmodule

module Johnson_count(input clk_in,reset,input [3:0] digit,output [7:0] led);
        wire [2:0] cntr;
        wire q0,q1,q2;
        wire q0_bar,q1_bar,q2_bar;
        wire clk_out;

        assign digit=4'b0001;

        clk_gen c1(clk_in,reset,clk_out);

        D_FF d1(q0_bar,clk_out,reset,q2,q2_bar);
        D_FF d2(q2,clk_out,reset,q1,q1_bar);
        D_FF d3(q1,clk_out,reset,q0,q0_bar);

        assign cntr={q2,q1,q0};

        decoder de1(cntr,led);

endmodule
```

**Schematics :**

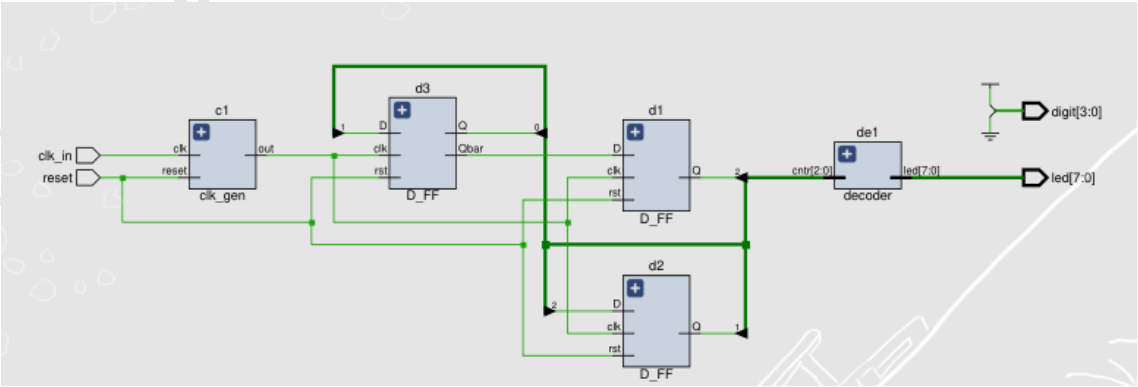
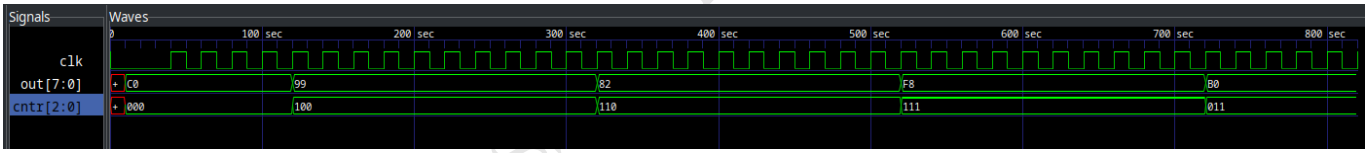



Figure 1: Schematics of the Data flow modelling

## Constraints on ports of FPGA

```
set_property IOSTANDARD LVCMOS33 [get_ports {digit[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {digit[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {digit[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {digit[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk_in]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN F2 [get_ports {digit[0]}]
set_property PACKAGE_PIN E1 [get_ports {digit[1]}]
set_property PACKAGE_PIN G5 [get_ports {digit[2]}]
set_property PACKAGE_PIN G4 [get_ports {digit[3]}]
set_property PACKAGE_PIN H1 [get_ports {led[7]}]
set_property PACKAGE_PIN H2 [get_ports {led[6]}]
set_property PACKAGE_PIN J4 [get_ports {led[5]}]
set_property PACKAGE_PIN J5 [get_ports {led[4]}]
set_property PACKAGE_PIN H4 [get_ports {led[3]}]
set_property PACKAGE_PIN H5 [get_ports {led[2]}]
set_property PACKAGE_PIN G1 [get_ports {led[1]}]
set_property PACKAGE_PIN G2 [get_ports {led[0]}]
set_property PACKAGE_PIN N11 [get_ports clk_in]
set_property PACKAGE_PIN L5 [get_ports reset]
```

## Simulation :




## Reports :

**Resources utilized :**

- 47 **SLICE LUT** were used, where LUT is used as a logic and not memory.
- 37 Slice Registers were used as Flip Flop.

**Clock :**

- only 1 Global Clock was utilized.

| *Clock Region Name* | *Global Clock (Used)* | **FF (Used)** | **LUTM (Used)** |
|---|---|---|---|
| *X0Y0* | *0* | *0* | *0* |
| *X1Y0* | *0* | *0* | *0* |
| *X0Y1* | *0* | *0* | *0* |
| *X1Y1* | *1* | *37* | *6* |
| *X0Y2* | *0* | *0* | *0* |
| *X1Y2* | *0* | *0* | *0* |

**Time delays**

| *Type of Path* | *Path taken* | **Time delay (ns)** |
|---|---|---|
| *Max Delay path* | → | |
| | → | |
| | → | |
| | → | |
| | → | |
| *Min Delay path* | → | |
| | → | |
| | → | |
| | → | |
| | → | |

**Power consumed**

- The power consumed by FPGA is **0.325**