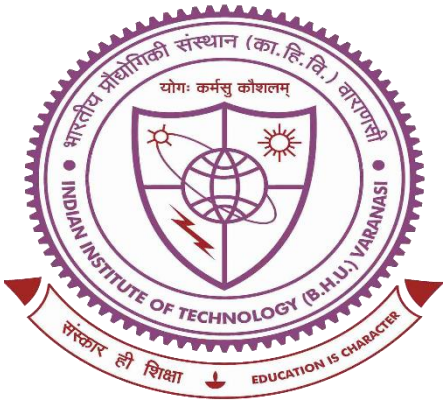
A Project Report
ON
DESIGN AND VERIFICATION OF DIGITAL SYSTEMS IN XILINX VIVADO USING VERILOG HDL



Submitted by

Akhilesh Kumar Meena

Under the guidance of

Dr. Avirup Maulik

DEPARTMENT OF ELECTRICAL ENGINEERING INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI, UTTAR PRADESH, INDIA-221005

20th MAY 2024 – 15th JULY 2024

ABSTRACT

This project focuses on the design and implementation of digital systems using Verilog Hardware Description Language (HDL) on the Digilent Basys 3 FPGA board. The project began with an in-depth study of digital design principles, specifically combinational and sequential circuits. Following this foundational knowledge, Verilog HDL was employed to describe various digital circuits, with an emphasis on precise and efficient design.

Multiple combinational and sequential circuits were developed, with their Register Transfer Level (RTL) design, synthesis, and implementation carried out using the Vivado design suite. The process included generating bitstreams and subsequently deploying them on the FPGA for real-time testing. The functionality of each design was rigorously verified on the Basys 3 board, ensuring that the circuits operated as intended.

This project demonstrates the practical application of Verilog HDL in digital design, showcasing the entire workflow from initial design through to hardware testing. The use of the FPGA board underscores the versatility and power of FPGA-based development, with applications in a wide range of digital systems.

INTRODUCTION

The ability to design and implement digital systems is crucial in today's technology-driven world. This project explores the use of Verilog HDL for designing various digital circuits and demonstrates their functionality on the Digilent Basys 3 FPGA board. By starting with fundamental digital

design principles, the project delves into creating and testing both combinational and sequential circuits.

Verilog HDL provides a robust framework for describing complex digital systems, and this project employs it to craft detailed circuit designs. Using the Vivado design suite, the project advances through the stages of RTL design, synthesis, and implementation, culminating in real-time testing on the FPGA. This hands-on approach highlights the practical application of Verilog and FPGA technology, illustrating their effectiveness in developing and verifying digital systems.

TYPES OF CIRCUIT DESIGNED:

Combinational Circuits: In the realm of digital electronics, combinational circuits are essential components used to perform various logical operations based solely on current input values. Unlike sequential circuits, which depend on past inputs and incorporate memory elements, combinational circuits deliver outputs that are directly related to the present combination of inputs. Combinational circuits process inputs to produce outputs through a series of logic gates, such as AND, OR, and NOT gates. These circuits do not have memory or feedback loops, meaning their operation is determined instantaneously by the input values.

Examples: In this project, combinational circuits such as adders, multiplexers, and decoders were implemented. These examples illustrate the practical applications of combinational logic in performing arithmetic operations, data selection, and encoding/decoding tasks.

Sequential Circuits: Sequential circuits are a class of digital circuits where the output depends not only on the current inputs but also on the history of past inputs. Unlike combinational circuits, which produce outputs solely based on the present input values, sequential circuits incorporate memory elements that allow them to store and use past information. This makes them essential for tasks that require state retention and time-based behavior. Sequential circuits use components like flip-flops, latches, and

registers to store data. These memory elements enable the circuit to maintain a state and respond to changes over time.

Types: Sequential circuits are categorized into two main types:

Synchronous Sequential Circuits: These circuits use a clock signal to synchronize changes in the state. The state transitions occur at specific clock edges, making the design and analysis more predictable. Examples include counters and shift registers.

Asynchronous Sequential Circuits: These circuits do not rely on a clock signal. Instead, they change states based on the input signals and their timing. They can be more complex to design and analyze due to their sensitivity to input changes.

Examples: In this project, sequential circuits such as counters and state machines were implemented. These examples demonstrate how sequential logic can be used for counting operations, state transitions, and managing sequential tasks in digital systems.

Introduction to Verilog HDL:

Verilog Hardware Description Language (HDL) is a powerful and widely used language for modeling and designing digital systems. Developed in the 1980s, Verilog provides a comprehensive framework for describing both the structural and behavioral aspects of digital circuits, making it a key tool in electronic design automation.

Language Overview: Verilog HDL enables designers to describe hardware at various levels of abstraction, from high-level behavioral descriptions to low-level gate and transistor-level implementations. This flexibility allows for both simulation and synthesis of digital systems. Verilog HDL have three type of modelling level.

Behavioral Level: Describes the operation of a circuit in terms of its functionality and algorithm without detailing the specific hardware

implementation. This level is useful for early design stages and rapid prototyping.

Register Transfer Level (RTL): Focuses on the data transfer between registers and the operations performed on this data. It is a common abstraction level for synthesis and optimization.

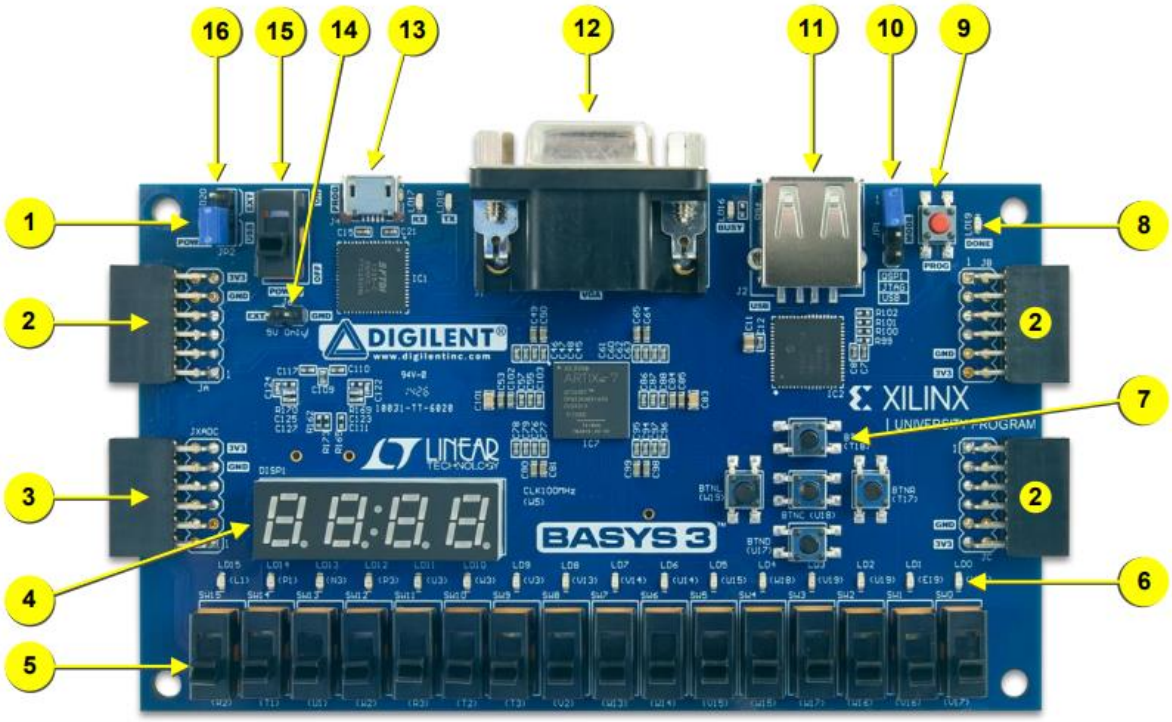
Gate Level: Describes the circuit in terms of logic gates and their interconnections. This level is used for detailed design and verification before physical implementation.

Tools and Hardware :

Xilinx vivado: Xilinx Vivado is a versatile software tool that plays a critical role in designing, synthesizing, and implementing digital circuits on FPGAs. It provides a user-friendly environment for writing and testing Verilog code, along with powerful tools for turning that code into functional hardware.Vivado excels at optimizing complex designs, ensuring they run efficiently on the FPGA. The process begins with synthesis, where the Verilog code is translated into a hardware representation. This is followed by implementation, where the design is mapped onto the specific architecture of the FPGA. Finally, Vivado generates a bitstream, which is the file that can be loaded onto the FPGA to configure it with the designed circuit.In this project, Vivado was essential for converting the Verilog HDL designs into working circuits on the Digilent Basys 3 FPGA board. It facilitated every step of the process, from coding and simulation to synthesis, implementation, and bitstream generation, enabling successful real-time testing and validation on the hardware.

Basys3 fpga board: The Basys 3 board is a versatile digital circuit development platform powered by the latest Artix-7 FPGA from Xilinx (specifically, the XC7A35T-1CPG236C model). It offers a robust solution for a wide range of projects, from simple combinational circuits to more intricate sequential systems such as embedded processors and controllers. With its combination of a high-capacity

FPGA, cost-effective design, and a variety of ports like USB and VGA, the Basys 3 board is ideal for both beginners and advanced users.



The board is equipped with an array of input/output devices, including 16 user switches, 16 user LEDs, and 5 pushbuttons, along with a 4-digit 7-segment display. It also features multiple Pmod ports for expansion, a 12-bit VGA output for display purposes, a USB-UART bridge for serial communication, and a USB-JTAG port for FPGA programming and debugging. Additionally, the board supports USB HID devices like mice, keyboards, and memory sticks, offering flexibility for various applications without the need for additional hardware.

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

The Artix-7 FPGA on the Basys 3 is designed for high-performance logic operations, providing greater capacity, enhanced speed, and more resources compared to previous generations. This makes it well-suited for demanding digital designs.

Design and Verification of Systems:

The design and modelling of circuit done by writing Verilog source code in Xilinx vivado tool and for input-output planning a constraint file written and after that bit generation process done and verification done by using basys 3 FPGA board.

Design of combinational circuit: In this case different type of combinational circuite design and verifiacation validated such as adder multiplexer and decoder.

Full adder:

```
`timescale 1ns / 1ps

module fulladder(a,b,c_in,sum,c_out);

input a,b,c_in;

output c_out;

output sum;

assign c_out=(a&b)|(b&c_in)|(a&c_in);

assign sum=a^b^c_in;

endmodule

module fulladderFA(a,b,c_in,sum,c_out);

input [3:0]a,b;

input c_in;

output[3:0] c_out;

output [3:0]sum;

 fulladder FA0(a[0],b[0],c_in,c_out[0],sum[0]);

 fulladder FA1(a[1],b[1],c_out[0],c_out[1],sum[1]);

 fulladder FA2(a[2],b[2],c_out[1],c_out[2],sum[2]);

 fulladder FA3(a[3],b[3],c_out[2],c_out[3],sum[3]);
```

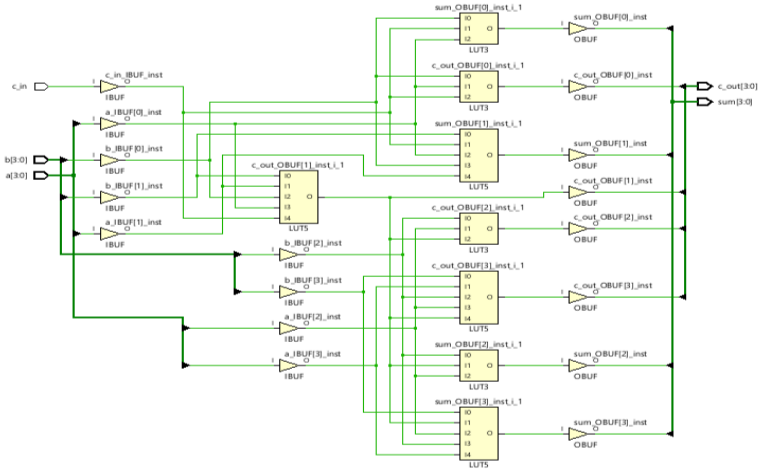


figure: Synthesised schematic.

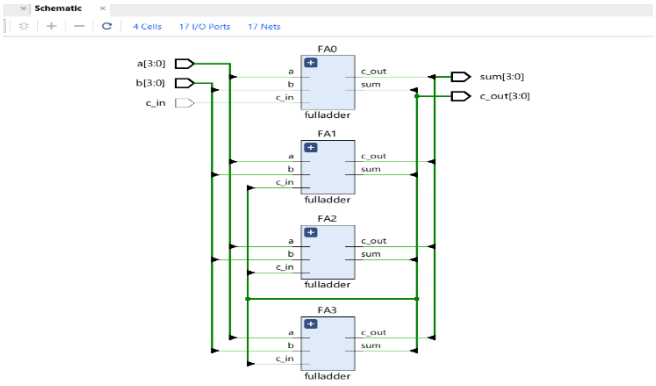


figure: RTL design schematic.

endmodule

4:1 Multiplexer:

```
`timescale 1ns / 1ps
module new2(a,b,c,d,s1,s2,z);
input [2:0]a,b,c,d;
input s1,s2;
output [2:0]z;
reg[2:0] z;
always(*)
begin
if(s1==0&s2==0)
z=a;
else if(s1==0&s2==1)
z=b;
else if(s1==1&s2==0)
z=c;
else if(s1==1&s2==1)
z=d;
end
endmodule
```

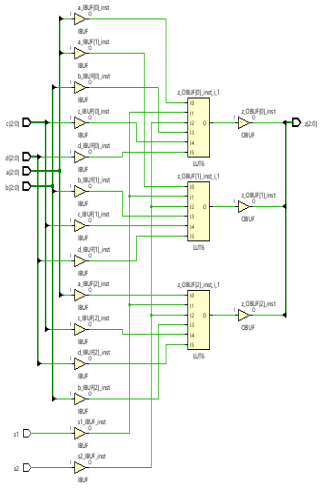


figure: synthesised schematic of 4:1 mux

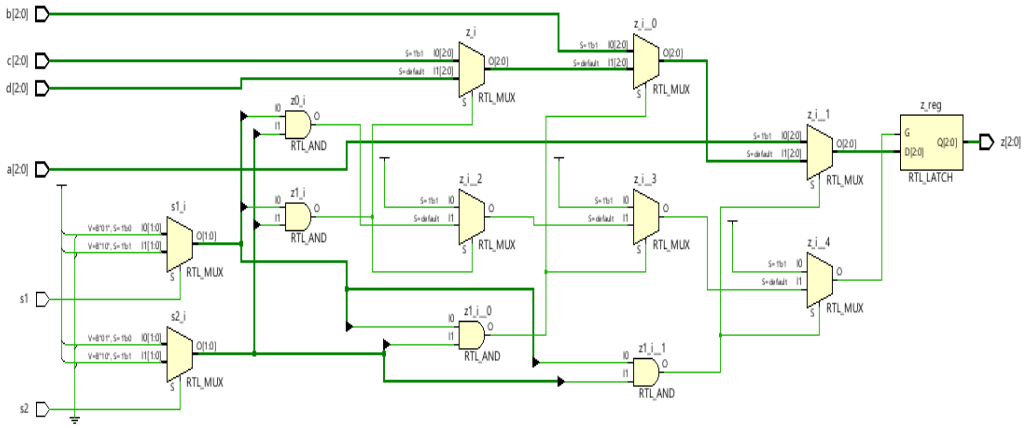


figure : RTL schematic of 4:1 mux

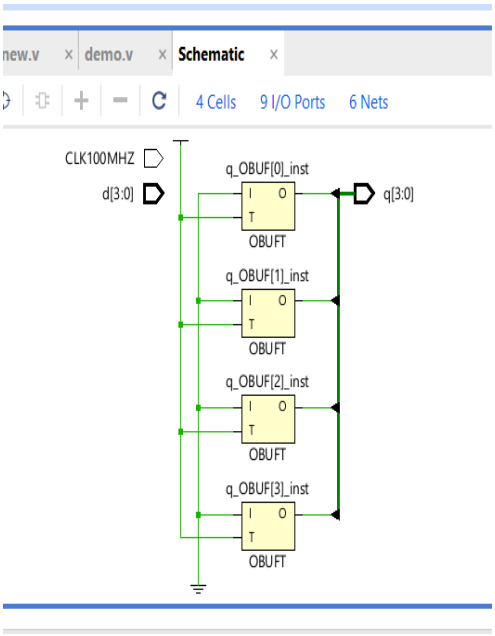
Sequential circuit:

counter counting upto 99and display on 7 segment display:

```
`timescale 1ns / 1ps
module new2(rst,CLK100MHZ,push_in,seg,an );
input rst; input CLK100MHZ;
    reg Q1, Q2;
    wire push_out;
    reg select;
    wire refresh_cycle;
    input push_in;
```

```
 reg [23:0]count;
    reg [3:0]q0;
    reg [3:0]q1;
    wire [3:0]q;
    //reg [3:0]clk_slow;
    output reg [0:6]seg;
    output [3:0]an;
    wire [3:0]an;
// 24-bit counter
    always @(posedge CLK100MHZ)
    begin
        count = count + 1'b1;
```

FIGURE: synthesised schematic



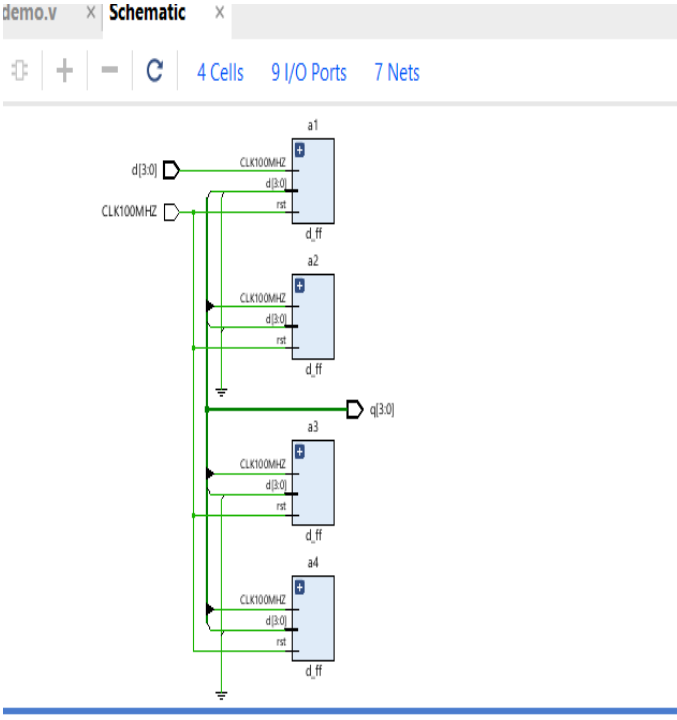


figure: RTL Schematic

```
    end
// D Flip Flop 1
    always @(posedge count)
    begin
    Q1 = push_in;
    end
// D Flip Flop 2
    always @(posedge count)
    begin
    Q2 = Q1;
    end
// AND gate
    and a1(push_out, Q1, ~Q2);
// REFRESH CYCLE ////////////////////////////////////////////
 assign refresh_cycle = count[18];
// Counters for unit digit and ones digit
    always @(posedge push_out, posedge rst)
    begin
```

```
    if (rst | (q1[3]& ~q1[2]&q1[1]&~q1[0]))
    begin
      q0 = 4'b0;
      q1 = 4'b0;
    end
    else
      begin
        if (q0[3]&~q0[2]&q0[1]&~q0[0])
         begin
         q0 = 4'd0;
         q1 = q1 + 1'b1;
        end
        else
         q0 = q0 + 1'b1;
      end
    end

// Selcting the display
    always @(refresh_cycle)
    begin
    if (refresh_cycle)
    select = 1'b1;
    else
    select = 1'b0;
    end
    assign an = select? (4'b1110):(4'b1101);
    assign q = select? (q0):(q1);
    always @(refresh_cycle)
       begin
       case (q)
          4'd0 : seg = 7'b0000001;
```

```
        4'd1 : seg = 7'b1001111;
        4'd2 : seg = 7'b0010010;
        4'd3 : seg = 7'b0000110;
        4'd4 : seg = 7'b1001100;
        4'd5 : seg = 7'b0100100;
        4'd6 : seg = 7'b0100000;
        4'd7 : seg = 7'b0001111;
        4'd8 : seg = 7'b0000000;
        4'd9 : seg = 7'b0000100;
        endcase
    end
endmodule
```

Constraints Code For Varification:

```
##############################
##two_digit_segment



#set_property PACKAGE_PIN W5 [get_ports CLK100MHZ]

#       set_property IOSTANDARD LVCMOS33 [get_ports CLK100MHZ]
#       create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLK100MHZ]

#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports pu]

#set_property PACKAGE_PIN U18 [get_ports rst]
```

```
#       set_property IOSTANDARD LVCMOS33 [get_ports rst]


## 7 segment
#set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
#set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
#set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
#set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
#set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
#set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
#set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]


## Anodes
#set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
#set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
#set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
#set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {an[3]}]
```

REMARK: HERE I AM GIVING GOOGLE DRIVE FILE FOR OTHER CODES AND ITS CONSTRAINTS CODE-

https://drive.google.com/drive/folders/1FStSOLGKq0hwTyx3Gwae8BWxC5vvg44F?usp=drive_link

CONCLUSION:

In this project, we successfully designed and tested digital systems using Verilog HDL on the Basys 3 FPGA board. By working with both combinational and sequential circuits, we gained practical experience in digital design. The Vivado tool was used to take our designs from concept to real hardware testing.

The FPGA board proved to be a powerful tool for creating and verifying digital circuits. The project helped us understand how to design, implement, and test digital systems effectively. The skills learned will be useful for future projects in digital design.