Mini Project Report: 4:1 MUX-based Data Routing System

# 1. Introduction

This project involves designing a digital system on an FPGA using Verilog HDL. The system utilizes a 4:1 multiplexer, decoder, latch, and a 2-bit counter to route data inputs based on the select lines generated by the counter. The project is implemented and tested on an FPGA development board with virtual input/output IP cores and a Xilinx IP core counter.

# 2. Objective

To implement a 4:1 MUX-based data routing system using behavioral and dataflow modeling techniques in Verilog HDL, and to visualize internal signals using the ChipScope IP Core.

# 3. Block Diagram

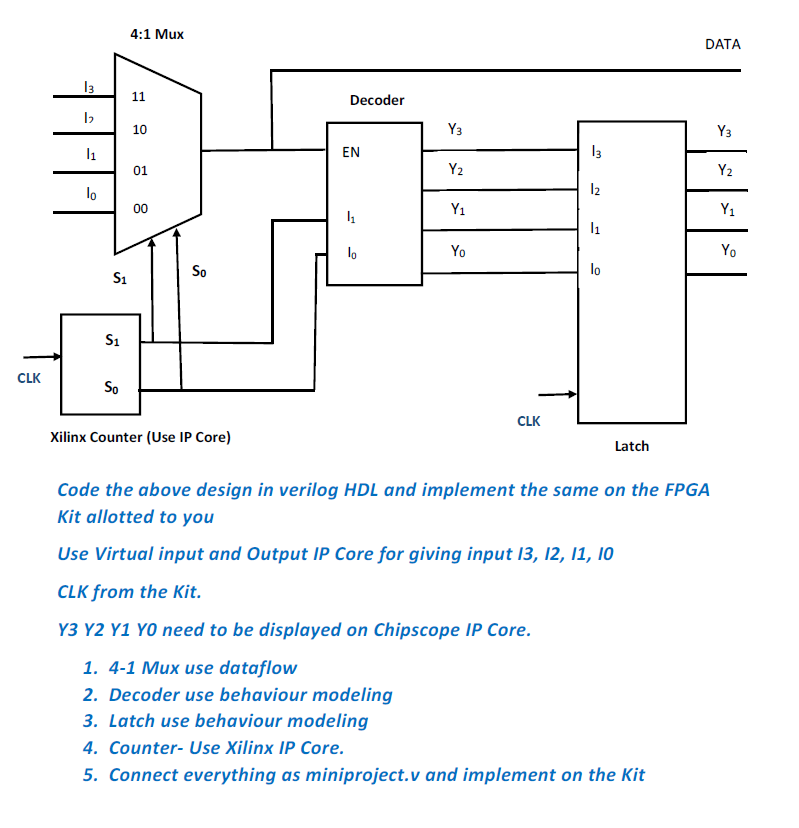


Figure 1: System Architecture

# 4. Methodology

The system is divided into four key modules:  
1. 4:1 Multiplexer - Designed using dataflow modeling.  
2. 2-to-4 Decoder - Designed using behavioral modeling.  
3. Latch - Designed using behavioral modeling to hold the selected data.  
4. Counter - Implemented using the Xilinx IP Core (2-bit counter).  
All modules are interconnected in a top-level module `miniproject.v`, with inputs from VIO IP cores and outputs monitored using the ChipScope IP Core.

# 5. Verilog Design Approach

Each module is implemented as follows:

* • Multiplexer: Dataflow modeling using assign statements.
* • Decoder: Behavioral modeling using case statements.
* • Latch: Behavioral modeling using always blocks triggered by the clock.
* • Counter: Xilinx IP core configured as a 2-bit synchronous counter.

# 6. FPGA Implementation

• Inputs (I0–I3) are provided via the VIO IP core.  
• Clock is sourced from the FPGA development board.  
• The counter generates select lines (S1, S0) for the MUX and decoder.  
• The latch captures the routed data based on the decoder’s output.  
• Output signals Y3–Y0 are monitored using the ChipScope IP Core.

# 7. Observations

The waveform captured via ChipScope shows correct selection and routing of input data lines based on the counter’s select signals. Each output Yx is asserted in sequence matching the expected behavior.

# 8. Conclusion

This mini project successfully demonstrates modular digital design using Verilog and FPGA tools. It reinforces concepts of data routing, behavioral/dataflow modeling, and IP core integration. The use of ChipScope enabled real-time signal analysis and verification.

# 9. Verilog HDL Code

## MUX (Dataflow Modeling)

module mux (

input wire I0, I1, I2, I3, // Data inputs

input wire S0, S1, // Select lines

output wire Y // Output

);

assign Y = (~S1 & ~S0 & I0) |

(~S1 & S0 & I1) |

( S1 & ~S0 & I2) |

( S1 & S0 & I3);

endmodule

## Decoder (Behavioral Modeling)

module decod (

input wire sl1,sl0, // 2-bit input

output reg [3:0] out // 4-bit output

);

always @(\*) begin

case ({sl1,sl0})

2'b00: out = 4'b0001;

2'b01: out = 4'b0010;

2'b10: out = 4'b0100;

2'b11: out = 4'b1000;

default: out = 4'b0000;

endcase

end

endmodule

## Latch (Behavioral Modeling)

module d\_latch (

input wire [3:0]I, // Data input

input wire En, // Enable (level-sensitive control)

output reg q1,q2,q3,q4

);

always @(\*) begin

if (En)

q1 = I[0];

q2 = I[1];

q3 = I[2];

q4 = I[3]; // Transparent when enabled

// else Q holds its value (implied)

end

endmodule

## Top Module (miniproject.v)

module system (input clk,input [3:0]N,output [3:0]Y,output data);

wire L1;

wire [1:0]W;

wire [3:0]a;

wire b1,b2,b3,b4;

mux uut1(.I0(N[0]),.I1(N[1]),.I2(N[2]),.I3(N[3]),.S0(W[0]),.S1(W[1]),.Y(L1));

decod uut2(.sl0(W[0]),.sl1(W[1]),.out(a));

d\_latch uut3(.I(a),.En(clk),.q1(b1),.q2(b2),.q3(b3),.q4(b4));

c\_counter\_binary\_0 uut4(.CLK(clk),.Q(W));

assign Y={b4,b3,b2,b1};

assign data=L1;

ila\_0 my\_ila (

.clk(clk), // Clock for ILA

.probe0(Y[0]) ,

.probe1(Y[1]),

.probe2(Y[2]),

.probe3(Y[3]) // Signal to be observed

);

endmodule

module top (input clk,output data);

wire [3:0] I;

system primary(.clk(clk),.N(I),.data(data))

vio\_0 control (.clk(clk),.probe\_out0(I));

endmodule

# 10. Simulation Waveforms

The following waveform was captured using ChipScope (or Vivado Simulator) during functional simulation. It shows correct propagation of input values through the MUX, decoding, and latching to output lines.

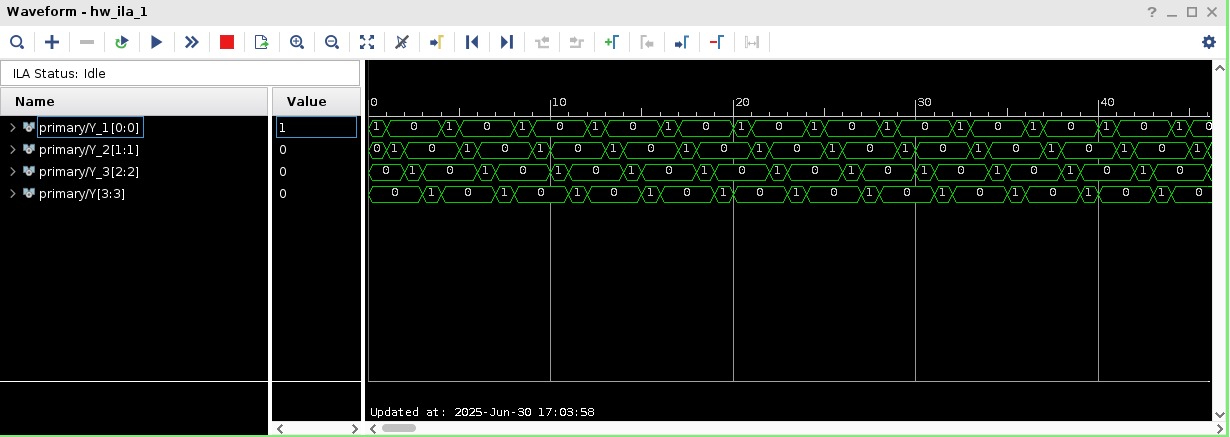
Waveforms should include signals: clk, sel (S1 S0), inputs (I3–I0), mux\_out, decoded[3:0], and out[3:0].

The output changes like 0001 → 0010 → 0100 → 1000 because:

* The **2-bit counter** cycles through 00 to 11
* The **decoder** converts that into **one-hot output**: Y0 to Y3
* That’s why only one bit is high at a time in a round-robin fashion
* The initial 0000 is just the power-on/reset state

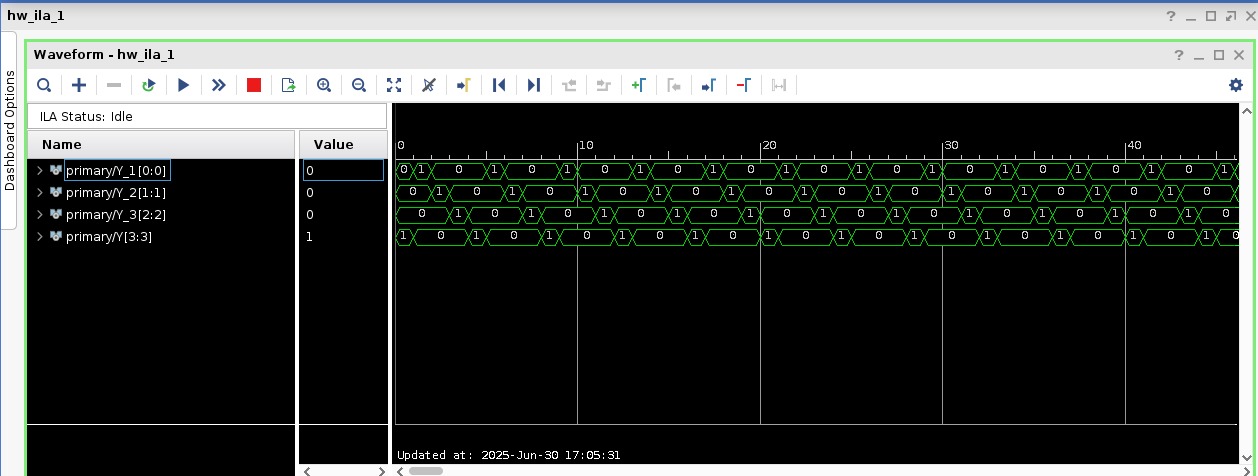
# 11. Simulation Waveform Results for Various Inputs

## With input I = 0000



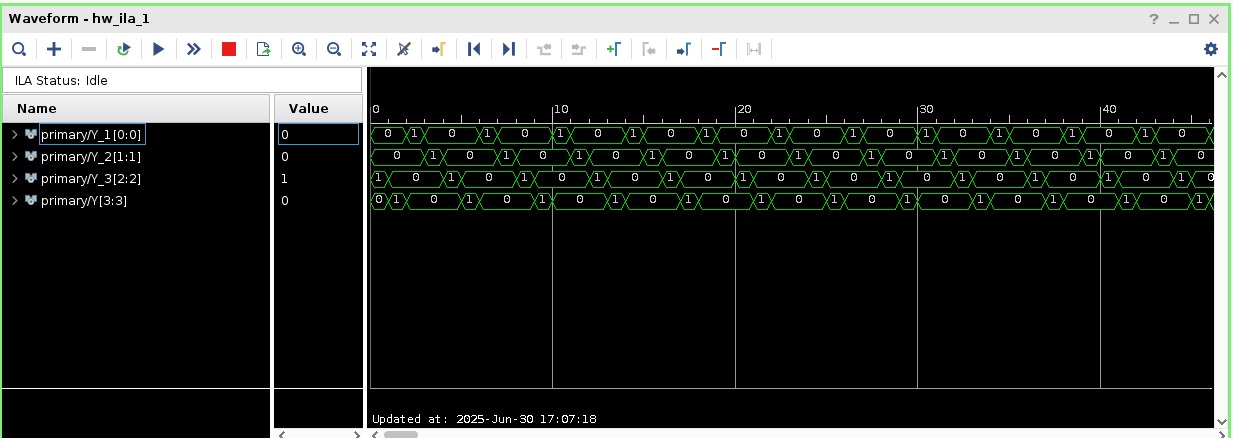
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 0001



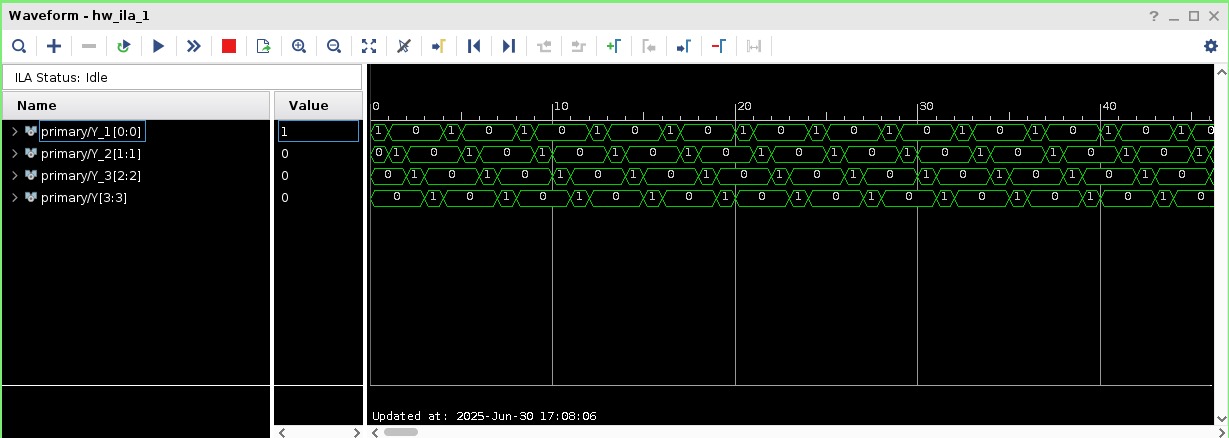
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 0010



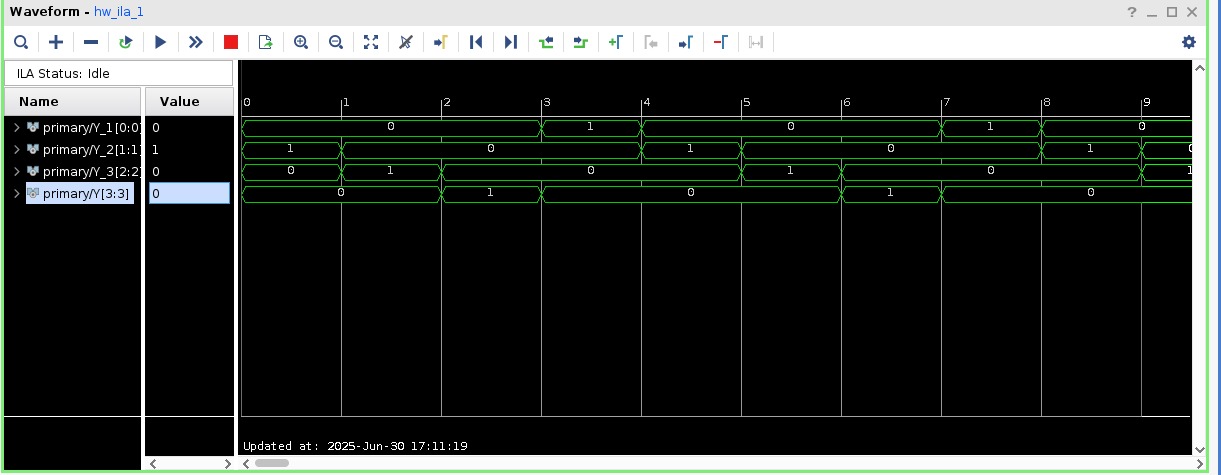
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 0011



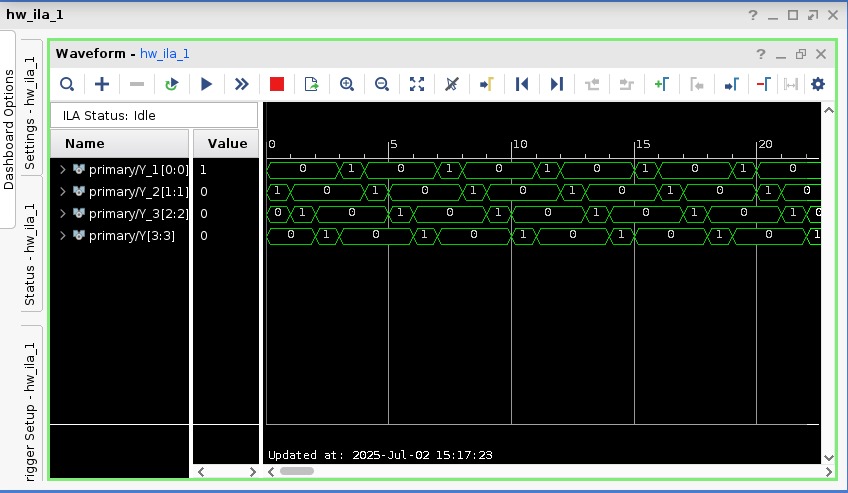
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 0100



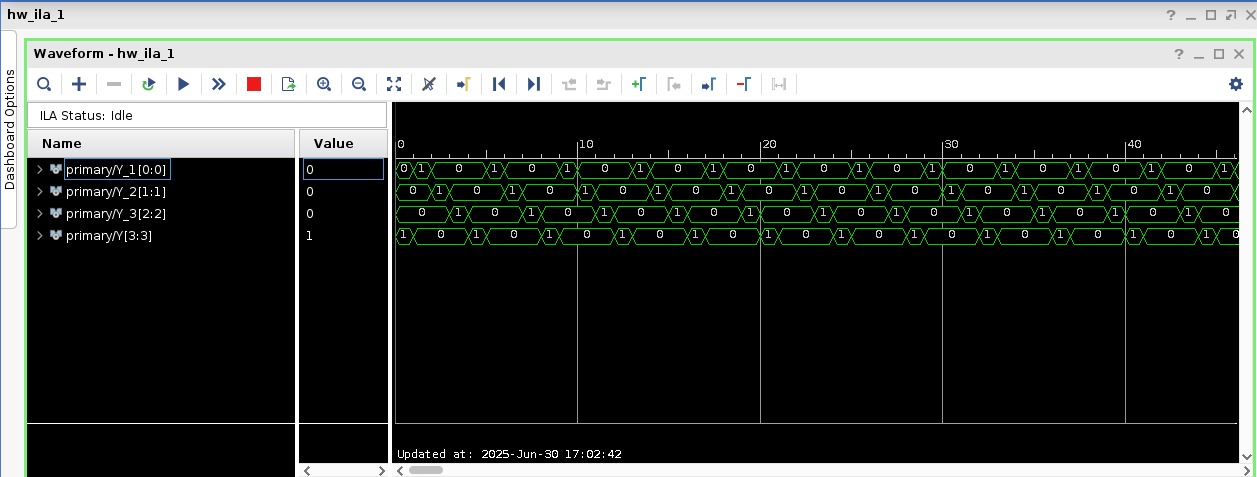
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 0101



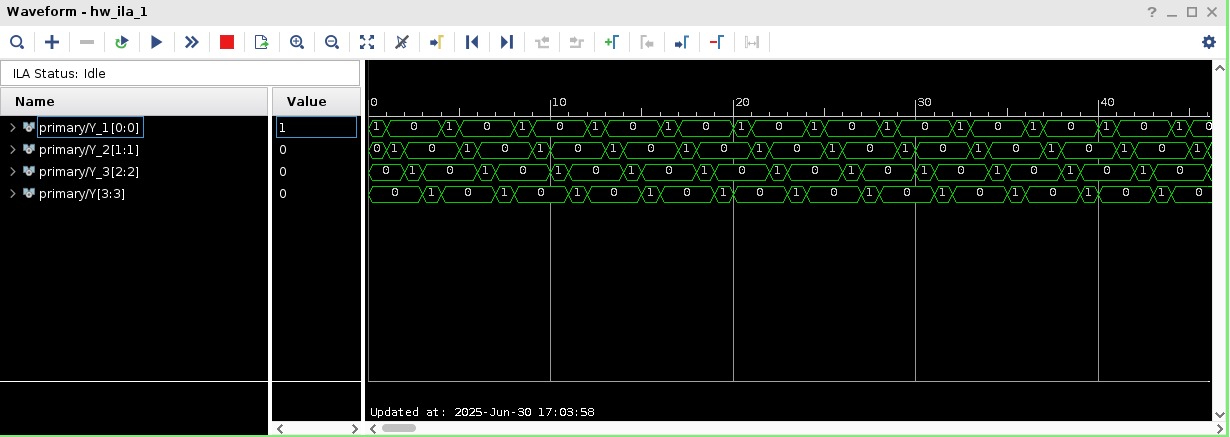
This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## With input I = 1111



This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.

## Miscellaneous waveform snapshot



This waveform shows the output signals Y0 to Y3 for the given input vector. The ChipScope trace confirms expected one-hot output behavior based on the decoded counter value.