Control of DC Motor using FPGA in vivado using Verilog

Introduction

This FPGA-based motor control project implements precise control of a DC motor using PWM for speed regulation and directional control via a switch-based interface. It leverages real-time signal generation and feedback logic to achieve high efficiency and reliability. The system is ideal for industrial applications requiring dynamic motor control.

Objectives

- Implement Precise Motor Control
- Design and Generate PWM Signals
- Provide Switch-Based Control Interface
- Optimize for Real-Time Performance

Tools and Techniques

- Tools: FPGA development board Xilinx, Verilog HDL, simulation tools Vivado
- Techniques: PWM for speed control, finite state machines for motor state management, and real-time signal processing for precise control and timing.

Components Used:

- **FPGA Board (Nexys A7-100T)**: For implementing the motor control logic using Verilog and running the PWM, direction, and speed control.
- **L298N Motor Driver**: To drive the DC motor, enabling bidirectional operation and speed regulation using PWM signals.
- **DC Motor**: The load controlled by the FPGA-based system for testing speed and direction functionality.
- **Oscilloscope**: Used for verifying PWM signal characteristics, such as frequency and duty cycle, during testing.
- **Power Supply**: To provide the required voltage and current for the motor driver and FPGA board.

Verilog code

Constraints File :

Clock Signal

```
set_property PACKAGE_PIN E3 [get_ports {CLK}];

create_clock -name sysclk -period 10 -waveform {0 5} [get_ports {CLK}];

set_property IOSTANDARD LVCMOS33 [get_ports {CLK}];
```

controlling the motor Switch

```
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {SWITCH[0]}];

set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {SWITCH[1]}];

set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {SWITCH[2]}];
```

Motor Control Pins (L298N Motor Driver)

```
set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {MOTOR[0]}];

set_property -dict {PACKAGE_PIN H1 IOSTANDARD LVCMOS33} [get_ports {MOTOR[1]}];

set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {MOTOR[2]}];
```

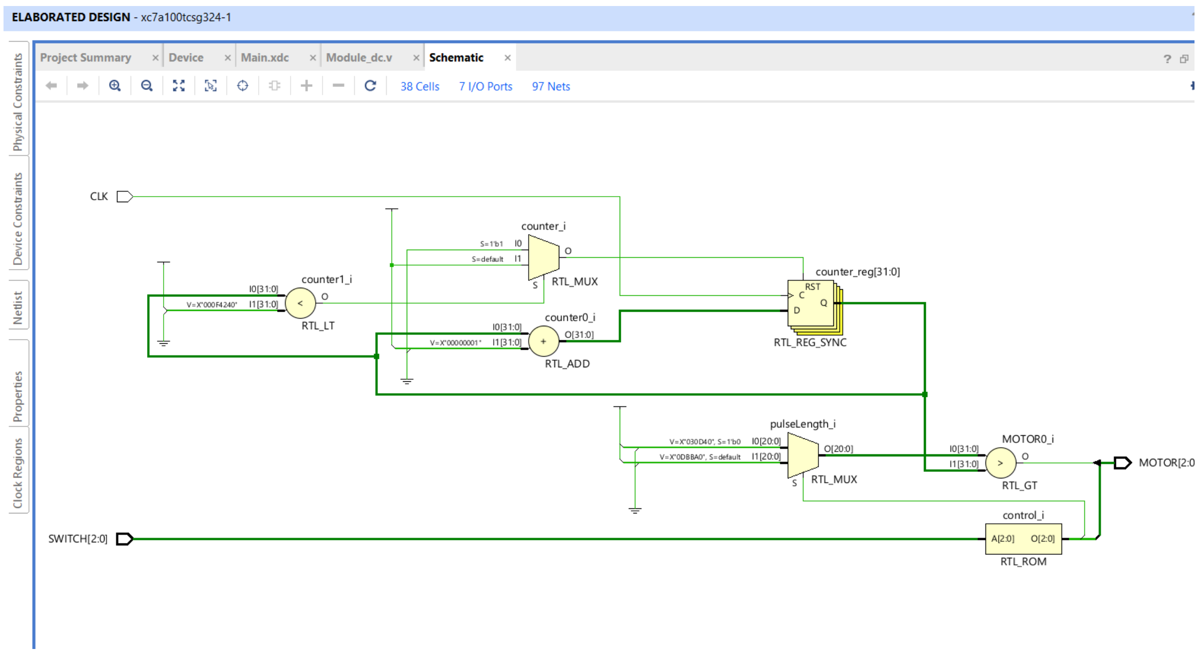
Verilog Module for Controlling motor:

```
`timescale 1ns / 1ps
module motor_controller (
    input wire CLK,                // System clock input
    input wire [2:0] SWITCH,       // 3-bit input switch for motor control
    output wire [2:0] MOTOR        // Motor control outputs
);

    // Internal signals
    reg [2:0] motor_control = 3'b000;  // Control signals for motor (Direction and Speed)

    // Parameters for PWM generation
    parameter integer PERIOD_LENGTH = 1000000;  // Total period of the PWM signal (clock cycles)
    parameter integer LOW_SPEED = 200000;       // PWM duty cycle for low speed
    parameter integer HIGH_SPEED = 900000;      // PWM duty cycle for high speed

    reg [19:0] pwm_counter = 0;                 // Counter for generating PWM signals
    integer pulse_width = 0;                    // Variable for adjusting duty cycle (speed)

    // Counter logic for PWM signal
    always @(posedge CLK) begin
        if (pwm_counter < PERIOD_LENGTH - 1)
            pwm_counter <= pwm_counter + 1;
        else
            pwm_counter <= 0;
    end

    // Switch-based motor control logic
    always @(*) begin
        case (SWITCH)
            3'b001: motor_control = 3'b001;   // Clockwise, low speed
            3'b101: motor_control = 3'b101;   // Clockwise, high speed
            3'b011: motor_control = 3'b010;   // Counterclockwise, low speed
            3'b111: motor_control = 3'b110;   // Counterclockwise, high speed
            default: motor_control = 3'b000;  // Motor OFF
        endcase

        // Adjust duty cycle based on speed selection
        if (motor_control[2] == 1'b0)
        pulse_width = LOW_SPEED;          // Low speed
    else
        pulse_width = HIGH_SPEED;         // High speed
  end

  // Output assignments
  assign MOTOR[0] = motor_control[0];      // Direction control: IN3
  assign MOTOR[1] = motor_control[1];      // Direction control: IN4
  assign MOTOR[2] = (pwm_counter < pulse_width) ? 1'b1 : 1'b0; // PWM for speed (ENB)

endmodule
```

Learnings from the Project:

1. Gained hands-on experience in **programming using Verilog HDL** to design and implement hardware logic for motor control.
2. Learned the operation and integration of an **L298N motor driver** to control DC motors using PWM and directional signals.
3. Acquired proficiency in using **Xilinx Vivado** for writing, synthesizing, simulating, and implementing FPGA designs.
4. Developed the ability to create and analyze **RTL designs**, ensuring logical correctness and efficient resource usage.
5. Mastered simulation and testing techniques to verify the functionality of the motor control system in a real-world hardware setup.

Elaborated Design:



Result of the Project:

The FPGA-based motor control system successfully demonstrated precise speed regulation and directional control of a DC motor. Using PWM signals with dynamically adjustable duty cycles, the motor operated smoothly in both clockwise and counterclockwise directions at low and high speeds.