Servomotor Control

by

151220202061 Yunus Emre Selen

An Engineering Term Project Report

Supervisor, Advisor:

Asst. Prof. Erol Seke

Electrical-Electronics Engineering Department

ESKISEHIR OSMANGAZI UNIVERSITY

December 24, 2024

ABSTRACT

In this project, the Spartan-3E Starter Kit FPGA board was utilized to implement servo motor control and real-time data display on a VGA screen. The rotation angle, direction, and speed of the servo motors were controlled using a rotary encoder and switches, and this information was displayed on the VGA screen. A ROM module containing ASCII characters was designed, and these characters were rendered in a 10x10 pixel format on the VGA display. The design successfully demonstrated the FPGA-based system's capability to perform both control and visualization tasks effectively..

ÖZET

Bu projede, Spartan-3E Starter Kit FPGA kartı kullanılarak servo motor kontrolü ve VGA ekranı üzerinden bilgi gösterimi gerçekleştirilmiştir. Servo motorların dönüş açısı, yönü ve hızı rotary encoder ve switchler yardımıyla kontrol edilmiş, bu bilgiler gerçek zamanlı olarak VGA ekranına aktarılmıştır. ASCII karakterlerini içeren bir ROM modülü tasarlanmış ve bu karakterler 10x10 piksel formatında VGA ekranında görüntülenmiştir. Tasarım, FPGA tabanlı bir sistemin hem kontrol hem de görselleştirme işlevlerini başarıyla yerine getirdiğini göstermiştir.

TABLE OF CONTENTS

1. Background.....1

2. Introduction.....1

3.Debouncer and Clock Divider 2

4.Servo Motor Control 3

4.1. PWM Signal Generation.....3

5.VGA…….....4

5.1. Raster 5

5.2. Displaying Characters And ROM.....6

6.User Guide 8

7.Conclusion 12

8.References.....13

9.Appendices.....14

1. Background

In modern digital electronics design, FPGAs (Field Programmable Gate Arrays) play an important role. FPGAs enable hardware-level programming, providing flexible and powerful design options. Image processing and control applications are among the areas where FPGAs stand out due to their high-speed and parallel processing capabilities.

This project aims to generate PWM signals and control servo motors using the Spartan-3E FPGA development board. Additionally, ASCII characters and various visualization structures are displayed on the VGA screen to enhance clarity and visual appeal. The main components of the project include controlling servo motors as desired through PWM signals, generating correct VGA signals, designing a ROM (Read-Only Memory) to store ASCII characters, positioning the characters properly on the screen, and converting image data into pixels.

This project demonstrates how to design a real-time motor control and visualization system using an FPGA, aiming to improve hardware programming skills.

2. Introduction

Field Programmable Gate Arrays (FPGAs) have become indispensable in the field of digital design due to their versatility and parallel processing capabilities. Unlike traditional microcontrollers, FPGAs offer the ability to program at the hardware level, providing customized, high-performance solutions for a wide range of applications, from signal processing to real-time control systems.

The primary objective of this project is to generate PWM signals to control servo motors and demonstrate the power of FPGAs in real-time motor control. The project involves controlling the speed, direction, and position of servo motors using rotary encoders and switches. The real-time status and movement of the servo motors are visually represented on a VGA screen to provide feedback.

To expand the scope of the project and enhance visual appeal, ASCII characters and various visualization structures were integrated and displayed on the VGA screen. During this process, a system capable of generating ASCII characters was developed and displayed through VGA output using the Spartan-3E FPGA development board. The implementation includes the design of a custom ROM to store character bitmaps, VGA signal generation, and the correct positioning of characters on the screen.

This integration highlights the flexibility and power of FPGAs in both hardware-based tasks such as motor control and computation-based tasks like visualization. The project aims to demonstrate how to develop real-time control systems using FPGAs and improve hardware programming skills.

3. Debouncer and Clock Divider

In the project, a Clock Divider was used to reduce the 50 MHz system clock to a lower frequency, as the high frequency was excessively high for the servo motor to operate (50 Hz). The VGA display was configured to operate directly with the 50 MHz clock frequency.

The Debouncer was implemented to eliminate mechanical bounce effects from components such as switches and rotary encoders. When the Clock Divider was set to a value of 250,000, sufficient delay was provided to filter out bounce effects. A shift register (SR) was used to monitor input signals and produce a stable output.

This ensured the correct selection of the servo motor through switches and reliable detection of speed adjustments from the rotary encoder. As a result, data was seamlessly transmitted to the VGA screen, and the servo motors were accurately controlled.

Clock Input	FPGA Pin	Global Buffer	Associated DCM
CLK_50MHZ	C9	GCLK10	DCM_X0Y1
CLK_AUX	B8	GCLK8	DCM_X0Y1
CLK_SMA	A10	GCLK7	DCM_X1Y1

Figure 1. (Clock Connections)

4. Servo Motor Control

In this project, servo motor control was implemented to achieve precise movement and speed adjustments. The servo motors used were capable of rotating within a range of 0 to 180 degrees. The speed of the servos was determined using a rotary encoder, while the selection of the servo motor to be controlled was achieved through the use of switches. A Clock Divider was employed to generate a clock signal appropriate for controlling the timing of the servo signals, ensuring accurate pulse-width modulation (PWM) for position and speed adjustments. Additionally, the Debouncer circuit was utilized to stabilize the signals received from the rotary encoder and switches, eliminating potential inaccuracies caused by signal noise or mechanical bounce.

The VGA screen was integrated into the system to display real-time information about the servo motor's speed, rotation angle, and direction. This visual feedback enhanced the user interface, making it easier to monitor and adjust the motor's operation. The system achieved the desired results, demonstrating smooth and reliable control of the servo motors with a clear representation of their operating parameters on the VGA display.

4.1 PWM Signal Generation

The PWM (Pulse Width Modulation) generator is essential for controlling the servo motor by producing a pulse signal with a variable duty cycle that determines the motor's rotation angle and speed. In this project, the PWM generator uses a 50 MHz system clock to create a signal with a frequency of 50 Hz. The duty cycle, which adjusts dynamically, defines the high-time (`high_time`) of the signal, calculated as a fraction of the total period based on the `MAXV` value (1,000,000). When the counter value (`cntr`) is less than `high_time`, the PWM output (`pwm_o`) is set high ('1'), and otherwise, it is set low ('0'). The servo motor interprets the duty cycle to determine its position, where, for example, a 5% duty cycle might correspond to 0 degrees and 15% to 180 degrees. In this project, the PWM generator provided precise control over the motor's position and speed, dynamically adjusting the duty cycle based on inputs from the rotary encoder and switches. The encoder controlled the speed or angular increment, while the switches selected the motor to control. This setup enabled smooth and accurate motor operation, with real-time feedback displayed on the VGA screen, showcasing parameters like speed, direction, and angle.

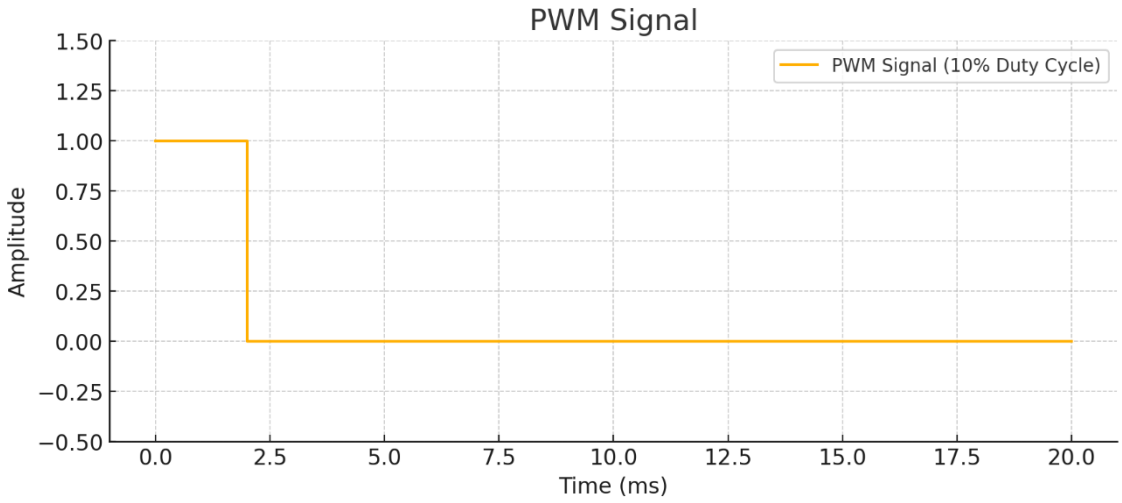

Figure 2. (PWM Signal)

5. VGA

In our VGA project, we implemented a VGA controller to display information on a screen using the Spartan 3E FPGA board. The project involved generating a raster scan to synchronize the horizontal and vertical signals, enabling the display of characters and graphical data on the VGA monitor. The VGA controller consisted of a clock signal, horizontal sync (HS), and vertical sync (VS) signals, which were generated using specific timing parameters adhering to VGA standards.

We used a "raster" module to manage horizontal and vertical counters for pixel scanning and synchronization. These counters ensured that the correct pixels were addressed, and the horizontal and vertical sync pulses were generated at the appropriate times. Additionally, we incorporated ASCII character generation, allowing alphanumeric characters to be displayed on the VGA screen. The character bitmaps were stored in ROM and fetched based on the pixel coordinates.

To enhance the display, we added functionality to show servo motor parameters, such as speed, angle, and direction, on the VGA screen. This was achieved by interfacing the motor control module with the VGA display module and formatting the data into a readable graphical output.

The overall implementation demonstrated the capability of the FPGA to handle real-time video signal generation and processing, showcasing the versatility of programmable hardware in embedded display systems.

Figure 3. (VGA Display)

5.1. Raster

A raster is a technique used in display systems where an image is represented as a grid of horizontal and vertical lines (pixels). In the context of VGA systems, the raster is responsible for generating the scanning process needed to display images on the screen. This involves creating horizontal and vertical synchronization signals (HS and VS) to control the timing and placement of pixels, ensuring proper image rendering on the display. The raster scans each line of pixels sequentially, starting from the top-left corner and moving to the bottom-right, before repeating the process to refresh the display.

In our project, we utilized the raster module to manage the generation of horizontal and vertical counters, which were essential for creating the synchronization signals required

by the VGA monitor. The counters determined the exact pixel being addressed at any given time and ensured that the synchronization signals adhered to VGA timing standards. This allowed us to properly render images and characters on the screen.

Specifically, in our project, the raster was crucial for displaying servo motor information, such as speed, angle, and direction, on the VGA screen. The horizontal and vertical counters were used to fetch pixel data from a ROM containing ASCII character bitmaps. This data was then displayed at the appropriate location on the screen, ensuring clear and organized visualization. By integrating the raster module, we were able to synchronize the display of both static and dynamic information, effectively demonstrating the servo motor's operation visually. This implementation highlighted the raster's role in enabling precise image rendering and real-time data display in FPGA-based systems.

5.2 Displaying Characters And ROM

First, a submodule was designed as a ROM structure, and an address structure was implemented within it. Initially, this address structure was planned as an 8x8 matrix. This allowed each character to be drawn within an 8x8 matrix. The address bit width was expanded to 9 bits (512 lines) to accommodate all desired characters, including alphabets, numbers, and some special symbols, resulting in approximately 400 lines. A data output was created to retrieve each line for the relevant module. Initially, this data output was designed to be 8 bits wide.

Later, all alphabets, numbers, and symbols were tested by displaying them on the screen through the VGA_Display module. During this process, distortions were observed in some letters. Upon investigating and performing solution-oriented experiments, it was discovered that the distortions were caused by overlapping in the 8x8 matrix structures. This overlap occurred because the characters fully utilized the 8x8 boundaries, causing conflicts with residual data from previous addresses.

To resolve this issue, the 8x8 structure used to store characters was updated to a 10x10 matrix. Both the first and last rows, as well as the first and last columns, were framed with '0' bits to create a border.

Example of the new border structure:

"0000000000"
"0.....0"
"0.....0"
"0.....0"
"0.....0"
"0.....0"
"0.....0"
"0.....0"
"0.....0"
"0000000000"

This framing effectively resolved the display issues.

Retrieving and Displaying Characters from ROM on Screen:

When the desired position is reached on the screen, represented by x and y coordinates, the address input of the ROM module receives the address of the first row of the 10x10 matrix structure for the character to be displayed. The bit width of the data at that address is also provided. The bits of the data retrieved from the address are then scanned along the x-axis, and any bits set to '1' are drawn in white on a black screen. This process ensures that each row and column of bits retrieved from the address is scanned and displayed on the screen.

Example Character and Displaying It on Screen:

The 10x10 'A' character in ROM:

```
0   => "0000000000", -- 'A'
1   => "0011111100",
2   => "0100000010",
3   => "0100000010",
4   => "0111111110",
5   => "0100000010",
6   => "0100000010",
7   => "0100000010",
8   => "0100000010",
9   => "0000000000",
```

Figure 4. 10x10 'A' character's bitmap

Displaying the 'A' character at coordinates x: 690-700, y: 500-510

```
--A
if (x >= 690 and x < 700) and (y >= 500  and y < 510) then
    rom_address <= std_logic_vector(to_unsigned(y - 500 +0, 9));

    if rom_data(x - 690) = '1' then
 RGB  <= "111";
    end if;

end if;
```

Figure 5. Displaying 'A' character

This code ensures that the 'A' character is displayed correctly within the specified coordinates without any distortions.

6. User Guide

Servo Motor Control System User Guide

This document explains how to use the switches and buttons in the VHDL-based servo motor control project and describes the behavior of the motors. The goal is to make the system easy to understand and use for users.

- **Function of Switches**

Switch	Functionality
`Switch2`	Determines the active motor group:
	- `0` : Right-Left (RL) and Up-Down (UD)
	- `1` : Front-Back (FB) and Open-Close (OC)
`Switch3`	Selects which motor's speed configuration to apply:
	- `0` : RL or FB (based on `Switch2`)
	- `1` : UD or OC (based on `Switch2`)
`Switch4`	Determines the speed mode:
	- `0` : Slow modes are active
	- `1` : Fast modes are active

Table 1. Switches and Functionality

Switches determine which servo motor is controlled and how the speed configurations are applied. The table below shows the functionality of each switch:

By combining these switches, the user can control both the active motor group and the speed configurations.

- **Function of Buttons**

Buttons control the movement of motors based on the configuration of `Switch2`. Their functions are outlined below:

Button	Action (When `Switch2 = 0`)	Action (When `Switch2 = 1`)
`East Button`	Move Right (RL)	Move Forward (FB)
`West Button`	Move Left (RL)	Move Backward (FB)
`North Button`	Move Up (UD)	Open (OC)
`South Button`	Move Down (UD)	Close (OC)
`Rotary Button`	Switch Modes (Fast/Slow)	Switch Modes (Fast/Slow)

Table 2. Buttons and Actions

- **Duty Cycle Limits**

Duty Cycle and Angular Positions:

The duty cycle of each motor is maintained within a specific range to prevent overloading or malfunction. The table below outlines the relationship between duty cycle values and corresponding angular positions of the servo motor:

Duty Cycle Value	Angular Position
`25001`	-90°
`75000`	0°
`125000`	+90°

Table 3. Duty Cycle Limits

- **Modes and Speeds**

Each servo motor has speed modes. The modes and their corresponding duty cycle increment steps are as follows:

Mode	Duty Cycle Increment Step
`ModeF4` (Fast)	±48
`ModeF3`	±36
`ModeF2`	±24
`ModeS4` (Slow)	±1
`ModeS3`	±4
`ModeS2`	±6

Table 4. Servo Modes

Modes are automatically adjusted based on rotary button and switch combinations.

- **Switching Between Modes**

Motor modes (Fast/Slow) can be changed using the rotary button:

1. **Pressing the Rotary Button**:
   - Each press of the rotary button cycles the active motor to the next mode.
   - Modes cycle as ModeS4 -> ModeS3 -> ModeS2 -> ModeF2 -> ModeF3 -> ModeF4.
2. **Effect of Switches**:
   - Change the switch configuration to select the active motor.
   - The rotary button cycles the modes of the motor selected by Switch2.
3. **Verifying the Mode**:
   - After a mode change, check the speed of duty cycle increments or decrements to confirm the active mode.
4. **Importance of Switch3 and Switch4 in Speed Configuration**:
   - **Switch3** selects which motor group's speed configuration to adjust:
     - Switch3 = 0: RL or FB (depending on Switch2)
     - Switch3 = 1: UD or OC (depending on Switch2)
   - **Switch4** determines whether slow or fast modes are active:
     - Switch4 = 0: Slow modes (ModeS2, ModeS3, ModeS4) are active.
     - Switch4 = 1: Fast modes (ModeF2, ModeF3, ModeF4) are active.

- **Usage Flow of Switches and Buttons**

Motor controls are performed with the following steps:

1. **Switch Selection**:
    - Set Switch2 to select the motor to be controlled.
    - For RL and UD motors, set Switch2 = 0; for FB and OC motors, set Switch2 = 1.
2. **Button Movements**:
    - Press the relevant buttons (East, West, North, South) to select the direction.
3. **Mode Adjustment**:
    - Use Switch3 and Switch4 to configure speed-related settings.
    - Press the rotary button to switch between speed modes (Fast/Slow).

7. Conclusion

In this project, we successfully implemented a complete system on an FPGA platform that integrates servo motor control with a VGA display interface. By leveraging key components such as a PWM generator, a raster module, and a ROM for ASCII character storage, we demonstrated precise motor control while providing real-time visual feedback on the VGA display. The clock divider and debouncer modules ensured smooth and accurate signal processing, further enhancing system reliability.

The servo motor control was achieved using a PWM signal that allowed us to adjust the motor's speed, angle, and direction effectively. The VGA display was used to present critical information, including motor parameters and ASCII characters, making the system both interactive and user-friendly. The raster module and ROM worked together to render text and graphics accurately on the screen, demonstrating the effectiveness of integrating digital design techniques in real-time applications.

Overall, this project highlights the versatility and capabilities of FPGA-based systems in combining real-time control with visual feedback. The successful implementation of the system showcases our understanding of digital design principles, as well as our ability to apply them to practical and functional use cases. The outcomes of this project not only meet our objectives but also pave the way for further improvements and applications in embedded systems design.

8. References

[1] Spartan-3E Starter Kit User Guide

[2] IEEE Standard for VHDL Language Reference Manual

[3] FPGA Prototyping by VHDL Examples by Pong P. Chu

[4] Digital Design: Principles and Practices by John F. Wakerly

9. Appendices

Appendix 1

BLOCK DIAGRAM OF DESIGN

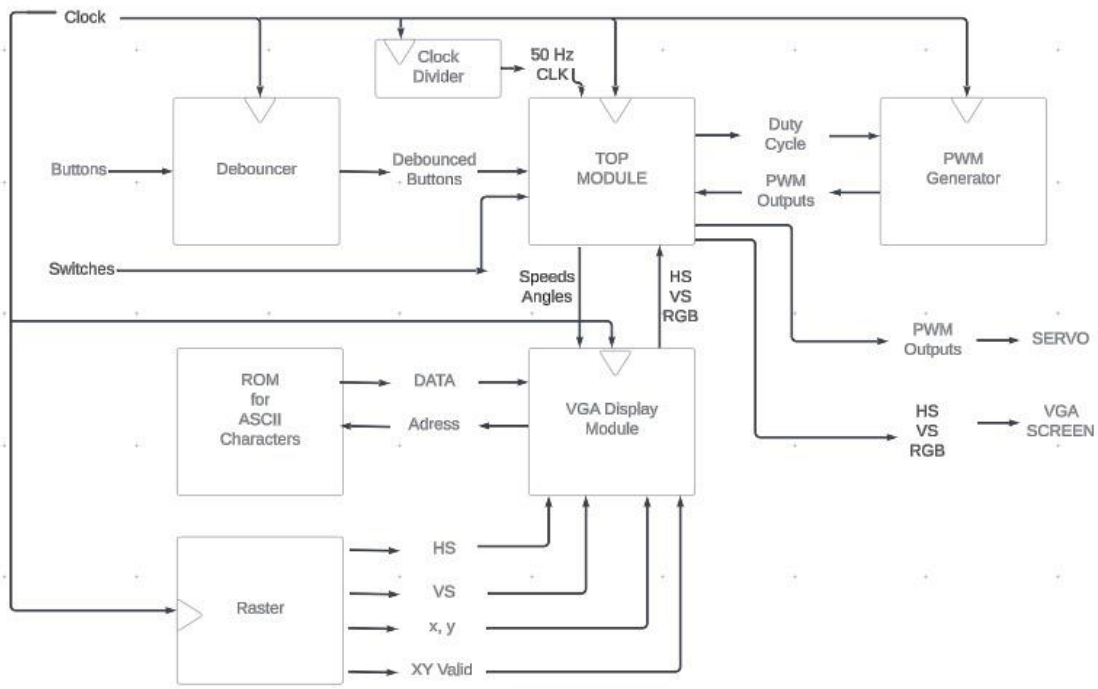

Figure 6. (Block Diagram of the Whole System)