Alexandria University

Faculty of Engineering

Department of Electrical Communications and Electronics

8-bit Super Register

Digital Systems Design Lab Project

Project Team Members

Student Name	Student ID
Elsayed Ashraf Bakry	23010293
Mohamed Ayman Elsaygh	23010730
Ahmed Bassem AboKila	24011105
Omar Ibrahim Elsagan	23010600
Mohamed Yasser Samak	24011107

Course: Digital Systems Design

Academic Year: 2025 – 2026

Contents

1 Introduction **2**

2 Complete VHDL Code **3**

3 Store Operation (000) **4**

4 Parallel Load Operation (001) **5**

5 Shift Right Operation (010) **6**

6 Shift Left Operation (011) **7**

7 Rotate Right Operation (100) **8**

8 Rotate Left Operation (101) **9**

9 Count Up Operation (110) **10**

10 Count Down Operation (111) **11**

11 Control Signal Summary **12**

12 Flag Operation **12**

13 Conclusion **13**

1 Introduction

This report presents the design and implementation of an 8-bit Super Register using VHDL. The register is capable of performing eight different operations including storage, shifting, rotation, and counting. The operation is selected using a 3-bit control signal, and all operations are synchronized with the rising edge of the clock.

2 Complete VHDL Code

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity super_register is
port(
    clk         : in  std_logic;
    data_in     : in  std_logic_vector(7 downto 0);
    data_sh_r   : in  std_logic;
    data_sh_l   : in  std_logic;
    control     : in  std_logic_vector(2 downto 0);
    data_out    : out std_logic_vector(7 downto 0);
    flag        : out std_logic
);
end super_register;

architecture behavior of super_register is
signal Q : std_logic_vector(7 downto 0);
begin
process(clk)
begin
    if rising_edge(clk) then
        case control is
            when "000" => Q <= Q;
            when "001" => Q <= data_in;
            when "010" =>
                for i in 0 to 6 loop
                    Q(i) <= Q(i+1);
                end loop;
                Q(7) <= data_sh_r;
            when "011" =>
                for i in 0 to 6 loop
                    Q(7-i) <= Q(6-i);
                end loop;
                Q(0) <= data_sh_l;
            when "100" =>
                for i in 0 to 6 loop
                    Q(i) <= Q(i+1);
                end loop;
                Q(7) <= Q(0);
            when "101" =>
                for i in 0 to 6 loop
                    Q(i+1) <= Q(i);
                end loop;
                Q(0) <= Q(7);
            when "110" => Q <= std_logic_vector(unsigned(Q) + 1);
            when "111" => Q <= std_logic_vector(unsigned(Q) - 1);
            when others => Q <= Q;
        end case;
    end if;
end process;

data_out <= Q;
flag <= '1' when (Q = "00000000" or Q = "11111111") else '0';
end behavior;
```

3 Store Operation (000)

The register holds its current value without change.

```
1 when "000" => Q <= Q;
```

Listing 1: Hold Operation VHDL Code

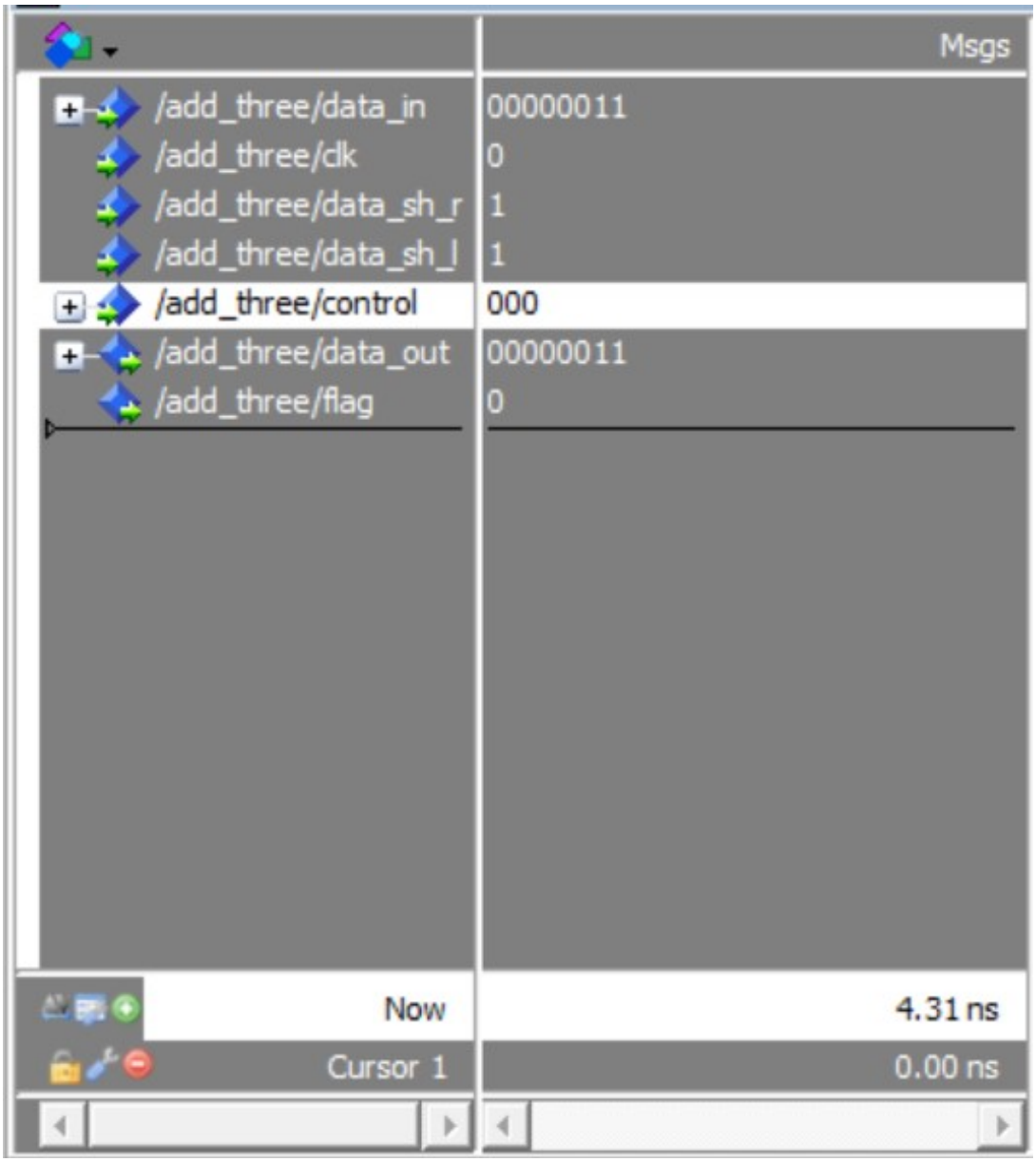

Figure 1: Simulation of Parallel Load Operation

Note: The hold operation maintains the current state of the register without any modification.

4 Parallel Load Operation (001)

Loads the input data into the register.

```
when "001" => Q <= data_in;
```

Listing 2: Parallel Load Operation VHDL Code

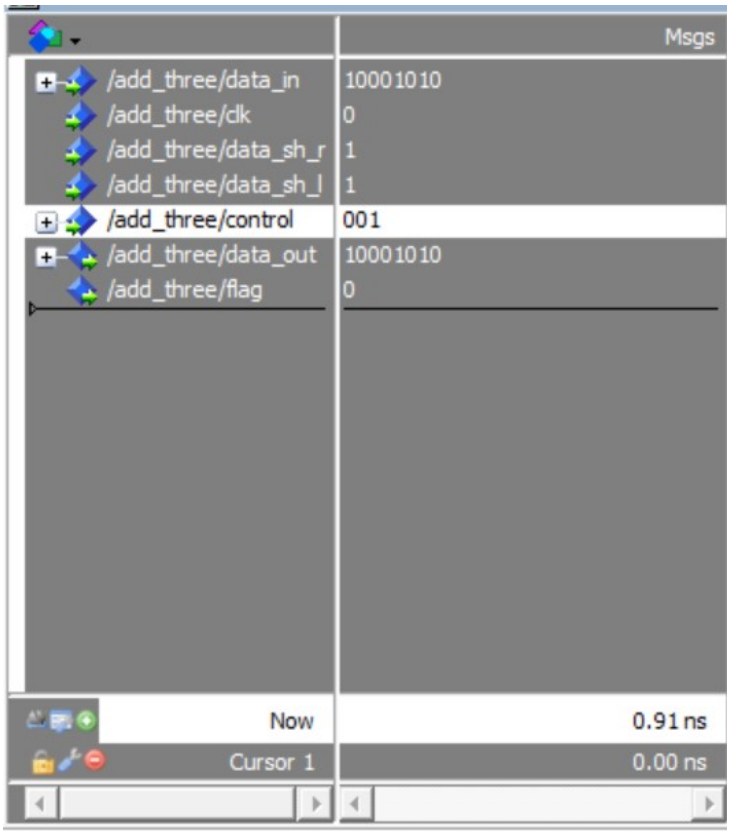

Figure 2: Simulation of Parallel Load Operation

Explanation: When control = "001", the register loads the 8-bit value from `data_in` on the rising edge of the clock.

5 Shift Right Operation (010)

Shifts all bits one position to the right. The most significant bit (MSB) receives the value from `data_sh_r`.

```
when "010" =>
    for i in 0 to 6 loop
        Q(i) <= Q(i+1);
    end loop;
    Q(7) <= data_sh_r;
```

Listing 3: Shift Right Operation VHDL Code

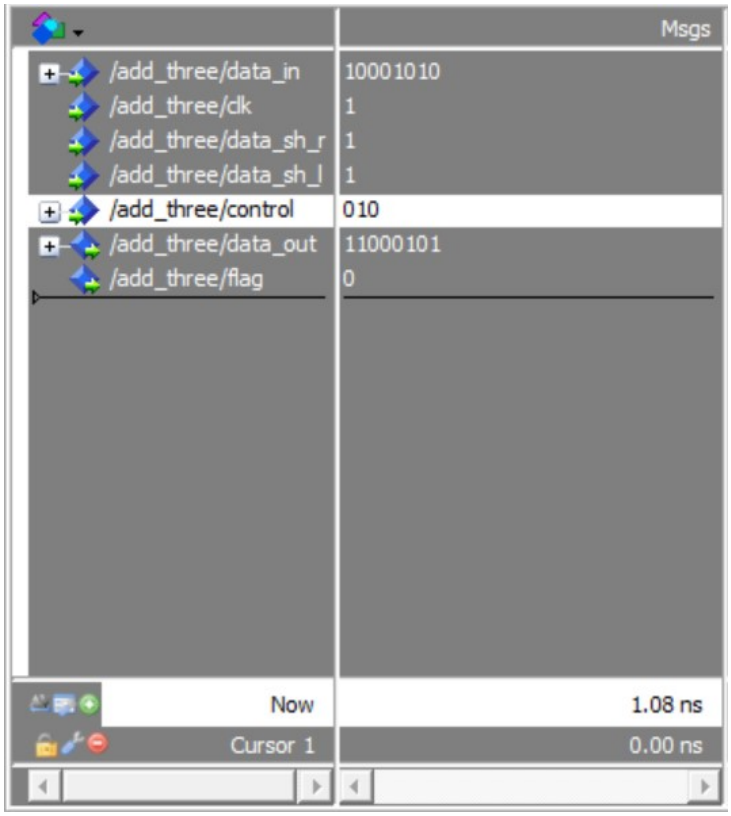

Figure 3: Simulation of Shift Right Operation

Operation:

- Bit 0 ← Bit 1
- Bit 1 ← Bit 2
- ...
- Bit 6 ← Bit 7
- Bit 7 ← `data_sh_r`

6 Shift Left Operation (011)

Shifts all bits one position to the left. The least significant bit (LSB) receives the value from `data_sh_l`.

```
when "011" =>
    for i in 0 to 6 loop
        Q(7-i) <= Q(6-i);
    end loop;
    Q(0) <= data_sh_l;
```

Listing 4: Shift Left Operation VHDL Code

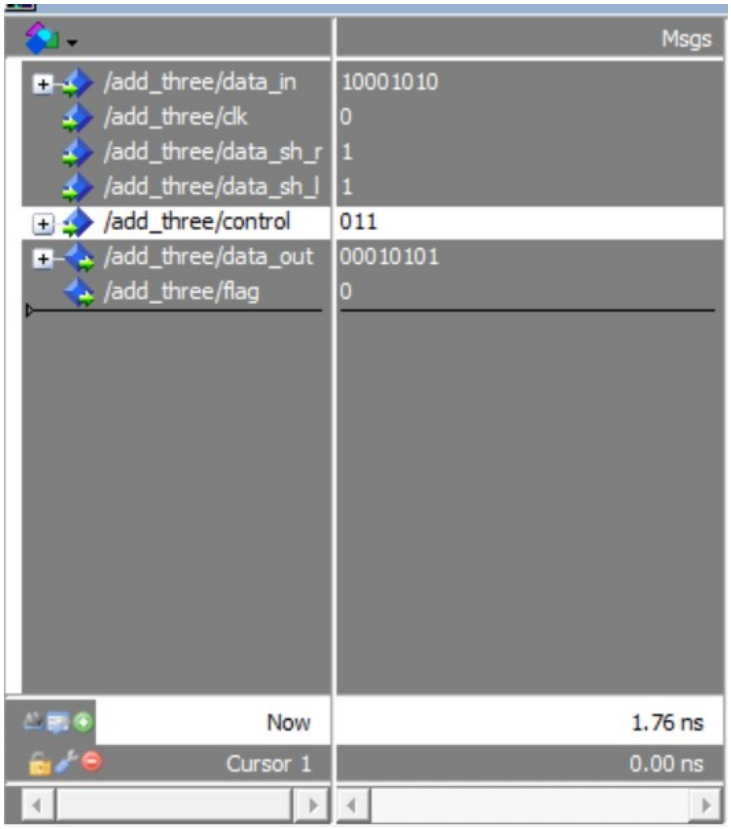

Figure 4: Simulation of Shift Left Operation

Operation:

- Bit 7 ← Bit 6
- Bit 6 ← Bit 5
- ...
- Bit 1 ← Bit 0
- Bit 0 ← `data_sh_l`

7 Rotate Right Operation (100)

Performs circular right rotation. The least significant bit wraps around to become the most significant bit.

```
when "100" =>
    for i in 0 to 6 loop
        Q(i) <= Q(i+1);
    end loop;
    Q(7) <= Q(0);
```

Listing 5: Rotate Right Operation VHDL Code

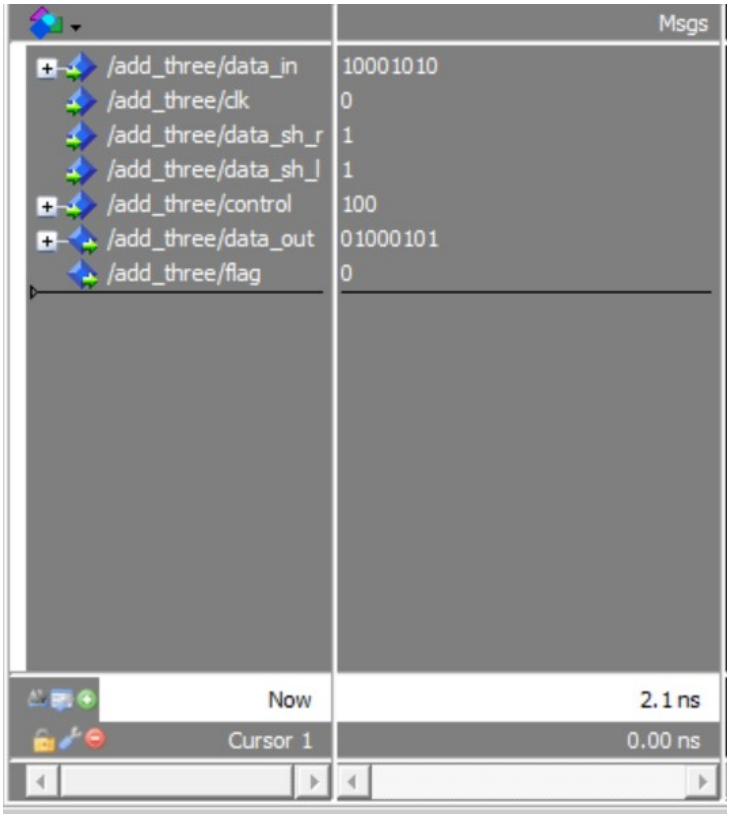

Figure 5: Simulation of Rotate Right Operation

Operation:

- Bit 0 ← Bit 1
- Bit 1 ← Bit 2
- ...
- Bit 6 ← Bit 7
- Bit 7 ← Original Bit 0

8 Rotate Left Operation (101)

Performs circular left rotation. The most significant bit wraps around to become the least significant bit.

```
when "101" =>
    for i in 0 to 6 loop
        Q(i+1) <= Q(i);
    end loop;
    Q(0) <= Q(7);
```

Listing 6: Rotate Left Operation VHDL Code

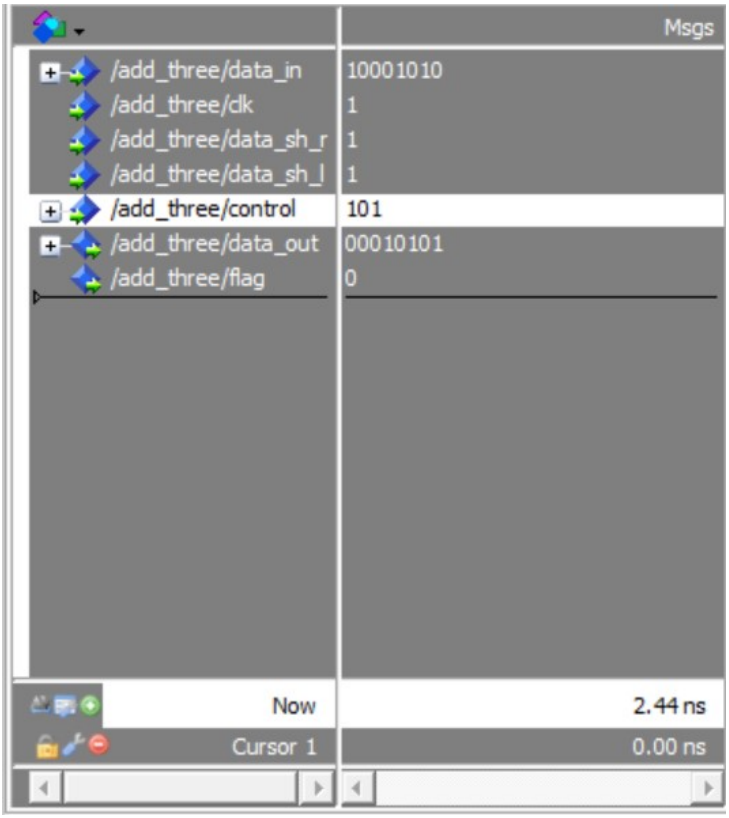

Figure 6: Simulation of Rotate Left Operation

Operation:

- Bit 7 ← Bit 6
- Bit 6 ← Bit 5
- ...
- Bit 1 ← Bit 0
- Bit 0 ← Original Bit 7

9 Count Up Operation (110)

Increments the register value by one using unsigned arithmetic.

```
1 when "110" => Q <= std_logic_vector(unsigned(Q) + 1);
```

Listing 7: Count Up Operation VHDL Code

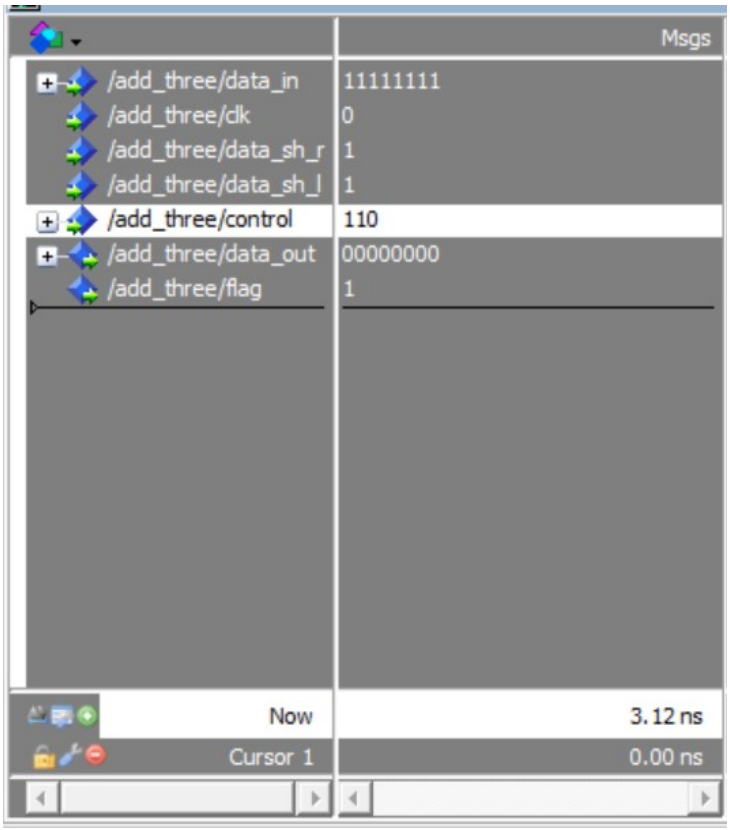

Figure 7: Simulation of Count Up Operation (with Flag Assertion)

Note: The flag signal becomes '1' when the register value reaches "11111111" and wraps to "00000000".

10 Count Down Operation (111)

Decrements the register value by one using unsigned arithmetic.

```vhdl
when "111" => Q <= std_logic_vector(unsigned(Q) - 1);
```

Listing 8: Count Down Operation VHDL Code

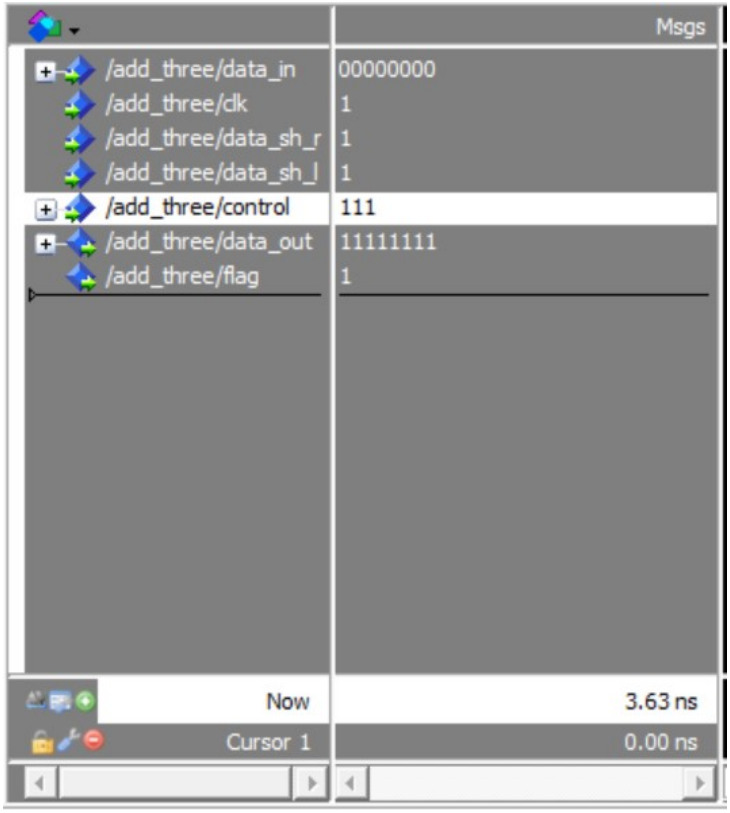

Figure 8: Simulation of Count Down Operation (with Flag Assertion)

Note: The flag signal becomes '1' when the register value reaches "00000000" and wraps to "11111111".

11 Control Signal Summary

Control	Operation	Description
000	Hold	Maintains current register value
001	Parallel Load	Loads `data_in` into register
010	Shift Right	Right shift with `data_sh_r` input
011	Shift Left	Left shift with `data_sh_l` input
100	Rotate Right	Circular right rotation
101	Rotate Left	Circular left rotation
110	Count Up	Increment register by one
111	Count Down	Decrement register by one

Table 1: Super Register Control Operations

12 Flag Operation

The flag signal indicates special conditions in the register. It is computed as follows:

```
flag <= '1' when (Q = "00000000" or Q = "11111111") else '0';
```

Listing 9: Flag Logic Implementation

Flag Conditions:

- `flag = '1'` when register contains all zeros (00000000)
- `flag = '1'` when register contains all ones (11111111)
- `flag = '0'` for all other values

Applications:

- Detecting overflow/underflow in counters
- Signaling end-of-sequence conditions
- Boundary detection in control algorithms

13 Conclusion

The 8-bit Super Register was successfully designed and implemented using VHDL. The design integrates multiple operations in a single synchronous circuit and was verified through simulation. The register can be easily extended or integrated into larger digital systems.

Key Features:

- 8 different operations controlled by 3-bit control signal
- Synchronous operation on rising clock edge
- Flag output for boundary condition detection
- Support for serial data input during shifts
- Built-in up/down counter functionality

Verification: All operations were verified through simulation as shown in the included figures, confirming correct functionality for each control code.