Report

Inputs:

Input: inCount: IN std_logic_vector(6 downto 0);

Explanation: The starting total number of food bars, entered by user using the left most 7 switches in the FPGA.

Input: bars_per_time: IN std_logic_vector(1 downto 0):="00";

Explanation: The number of bars dispensed per time, entered by user using the right most 2 switches in the FPGA.

Input: clk: IN std_logic;

Explanation: The clock which is generated from the FPGA.

Input: sensor: IN std_logic;

Explanation: The infra-red sensor used.

Input: inCount: IN std_logic_vector(6 downto 0);

Explanation: The starting total number of food bars

Outputs:

Output: outDisplayLeftMost: OUT std_logic_vector(6 downto 0):="1111111"; --T2

Explanation: The 4th from the right 7 segment display.

Output: outDisplayLeftMiddle: OUT std_logic_vector(6 downto 0):="1111111";

Explanation: The 3rd from the right 7 segment display.

Output: outDisplayRightMiddle: OUT std_logic_vector(6 downto 0);

Explanation: The 2nd from the right 7 segment display.

Output: outDisplayRightMost: OUT std_logic_vector(6 downto 0);

Explanation: The 1st from the right 7 segment display.

Output: pwm: OUT std_logic);

Explanation: The signal controlling starting and stopping of the motor.

Pin Assignments:

outDisplayLeftMost[4]	Output	PIN_C20
outDisplayLeftMost[3]	Output	PIN_C19
outDisplayLeftMost[2]	Output	PIN_E21
outDisplayLeftMost[1]	Output	PIN_E22
outDisplayLeftMost[0]	Output	PIN_F21
outDisplay...tMiddle[6]	Output	PIN_B17
outDisplay...tMiddle[5]	Output	PIN_A18
outDisplay...tMiddle[4]	Output	PIN_A17
outDisplay...tMiddle[3]	Output	PIN_B16
outDisplay...tMiddle[2]	Output	PIN_E18
outDisplay...tMiddle[1]	Output	PIN_D18
outDisplay...tMiddle[0]	Output	PIN_C18
outDisplay...ghtMost[6]	Output	PIN_C17
outDisplay...ghtMost[5]	Output	PIN_D17
outDisplay...ghtMost[4]	Output	PIN_E16
outDisplay...ghtMost[3]	Output	PIN_C16
outDisplay...ghtMost[2]	Output	PIN_C15
outDisplay...ghtMost[1]	Output	PIN_E15
outDisplay...ghtMost[0]	Output	PIN_C14
pwm	Output	PIN_V10
sensor	Input	PIN_V9

Node Name	Direction	Location
bars_per_time[1]	Input	PIN_C11
bars_per_time[0]	Input	PIN_C10
clk	Input	PIN_N5
inCount[6]	Input	PIN_F15
inCount[5]	Input	PIN_B14
inCount[4]	Input	PIN_A14
inCount[3]	Input	PIN_A13
inCount[2]	Input	PIN_B12
inCount[1]	Input	PIN_A12
inCount[0]	Input	PIN_C12
outDisplay...tMiddle[6]	Output	PIN_B22
outDisplay...tMiddle[5]	Output	PIN_C22
outDisplay...tMiddle[4]	Output	PIN_B21
outDisplay...tMiddle[3]	Output	PIN_A21
outDisplay...tMiddle[2]	Output	PIN_B19
outDisplay...tMiddle[1]	Output	PIN_A20
outDisplay...tMiddle[0]	Output	PIN_B20
outDisplayLeftMost[6]	Output	PIN_E17
outDisplayLeftMost[5]	Output	PIN_D19
outDisplayLeftMost[4]	Output	PIN_C20
outDisplayLeftMost[3]	Output	PIN_C19
outDisplayLeftMost[2]	Output	PIN_E21
outDisplayLeftMost[1]	Output	PIN_E22
outDisplayLeftMost[0]	Output	PIN_F21
outDisplay...tMiddle[6]	Output	PIN_B17
outDisplay...tMiddle[5]	Output	PIN_A18

Code Description:

CODE: if rising_edge(clk) then

```
if flag='0' then
DIFF:=to_integer(unsigned(bars_per_time));
COUNT:=to_integer(unsigned(inCount));
outDisplayLeftMost<="1111111";
outDisplayLeftMiddle<="1111111";
flag<='1';
end if;
q <= q+1;
if q =500000 then
--motorSignal<="00";
q <=0;
IF  sensor='0'  AND F='0'THEN
COUNT := COUNT - DIFF;
F:='1';
```

Description: At clk's rising edge we set the initial values of DIFF,COUNT, outDisplayLeftMost, outDisplayLeftMiddle and flag.

Using q we add a delay along with flag F we subtract the difference set by user to accumulative value "COUNT" because if q and F are not added the sensor will sense the object too fast and count doesn't register the subtraction of difference on the segment display.

CODE:

```
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01000" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01001";
                WHEN "10" =>    motorSignal <= "01010";
                WHEN "11" =>    motorSignal <= "01011";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01001" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01010";
                WHEN "10" =>    motorSignal <= "01011";
                WHEN "11" =>    motorSignal <= "01100";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01010" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01011";
                WHEN "10" =>    motorSignal <= "01100";
                WHEN "11" =>    motorSignal <= "01101";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01011" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01100";
                WHEN "10" =>    motorSignal <= "01101";
                WHEN "11" =>    motorSignal <= "01110";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01100" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01101";
                WHEN "10" =>    motorSignal <= "01110";
                WHEN "11" =>    motorSignal <= "01111";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01101" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
                WHEN "01" =>    motorSignal <= "01110";
                WHEN "10" =>    motorSignal <= "01111";
                WHEN "11" =>    motorSignal <= "10000";
                WHEN OTHERS => motorSignal <= "00000";
            END CASE;
WHEN "01110" => CASE bars_per_time IS
                WHEN "00" =>    motorSignal <= "00000";
```

```
IF COUNT>=0 THEN
    CASE motorSignal IS
        WHEN "00000" =>
                    CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "00001";
                        WHEN "10" =>    motorSignal <= "00010";
                        WHEN "11" =>    motorSignal <= "00011";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00001" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00001";
                        WHEN "01" =>    motorSignal <= "00010";
                        WHEN "10" =>    motorSignal <= "00011";
                        WHEN "11" =>    motorSignal <= "00100";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00010" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00010";
                        WHEN "01" =>    motorSignal <= "00011";
                        WHEN "10" =>    motorSignal <= "00100";
                        WHEN "11" =>    motorSignal <= "00011";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00011" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "00100";
                        WHEN "10" =>    motorSignal <= "00101";
                        WHEN "11" =>    motorSignal <= "00110";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00100" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "00101";
                        WHEN "10" =>    motorSignal <= "00110";
                        WHEN "11" =>    motorSignal <= "00111";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00101" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "00110";
                        WHEN "10" =>    motorSignal <= "00111";
                        WHEN "11" =>    motorSignal <= "01000";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00110" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "00111";
                        WHEN "10" =>    motorSignal <= "01000";
                        WHEN "11" =>    motorSignal <= "01001";
                        WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
        WHEN "00111" => CASE bars_per_time IS
                        WHEN "00" =>    motorSignal <= "00000";
                        WHEN "01" =>    motorSignal <= "01000";
```

```
                    END CASE;
                WHEN "10000" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10001";
                                WHEN "10" =>    motorSignal <= "10010";
                                WHEN "11" =>    motorSignal <= "10011";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN "10001" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10010";
                                WHEN "10" =>    motorSignal <= "10011";
                                WHEN "11" =>    motorSignal <= "10100";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN "10010" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10011";
                                WHEN "10" =>    motorSignal <= "10100";
                                WHEN "11" =>    motorSignal <= "10101";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN "10011" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10100";
                                WHEN "10" =>    motorSignal <= "10101";
                                WHEN "11" =>    motorSignal <= "10110";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN "10100" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10101";
                                WHEN "10" =>    motorSignal <= "10110";
                                WHEN "11" =>    motorSignal <= "10111";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN "10101" => CASE bars_per_time IS
                                WHEN "00" =>    motorSignal <= "00000";
                                WHEN "01" =>    motorSignal <= "10110";
                                WHEN "10" =>    motorSignal <= "10111";
                                WHEN "11" =>    motorSignal <= "11000";
                                WHEN OTHERS => motorSignal <= "00000";
                    END CASE;
                WHEN OTHERS => motorSignal <= "00000";
                END CASE;
            END IF;
        END IF;
    END IF;
END IF;
```

Description:

We encountered trouble with multiplication of std_logic_vectors and the conversion of integer to std_logic_vector so we decided due to lack of time to write this inefficient code that according to previous “motorSignal” and entered “bars_per_time” we set motor signal according to that so in general: if “bar_per_time” is larger then difference between previous and current motorSignal will be larger for a larger angle of rotation for larger number of bars to be shown to pet.

CODE:

```
                    IF sensor='1' THEN
F:='0';
END IF;
if(COUNT<0) then
outDisplayRightMiddle<="0101011";
outDisplayRightMost<="0001100";
outDisplayLeftMost<="0000110";
outDisplayLeftMiddle<="0101011";
--motorSignal<="00000";
else
CASE (COUNT MOD 10) IS
WHEN 0 => outDisplayRightMost <=    "1000000";
WHEN 1=> outDisplayRightMost <=    "1111001";
WHEN 2=> outDisplayRightMost <=    "0100100";
WHEN 3 => outDisplayRightMost <=    "0110000";
WHEN 4 => outDisplayRightMost <=    "0011001";
WHEN 5 => outDisplayRightMost <=    "0010010";
WHEN 6 => outDisplayRightMost <=    "0000010";
```

```
WHEN 7 => outDisplayRightMost <=    "1111000";
WHEN 8 => outDisplayRightMost <=    "0000000" ;
WHEN 9 => outDisplayRightMost <=    "0010000";
WHEN OTHERS => outDisplayRightMost <= "-------";
END CASE;
CASE (COUNT/10) IS
WHEN 0 => outDisplayRightMiddle <=    "1000000";
WHEN 1=> outDisplayRightMiddle <=     "1111001";
WHEN 2=> outDisplayRightMiddle <=     "0100100";
WHEN 3 => outDisplayRightMiddle <=    "0110000";
WHEN 4 => outDisplayRightMiddle <=    "0011001";
WHEN 5 => outDisplayRightMiddle <=    "0010010";
WHEN 6 => outDisplayRightMiddle <=    "0000010";
WHEN 7 => outDisplayRightMiddle <=    "1111000";
WHEN 8 => outDisplayRightMiddle <=    "0000000" ;
WHEN 9 => outDisplayRightMiddle <=    "0010000";
WHEN OTHERS => outDisplayRightMiddle <= "-------";
END CASE;
END iF;
END PROCESS;
stageOpenFirst: gate PORT MAP(clk,motorSignal,pwm);
```

Description:

If the sensor is not sensing anything change F to 0 used when subtracting difference and motor.

Then display current count on the 2 right 7 segment display by separating the count to 2 digits the ones and the tens.

Then use the gate component to input clk and motorSignal (for specific rotaion) and outputs pwm that controls the motor.