COL215: Assignment 2
Hardware Assignment

Deadline: 8th September 2024

1 Introduction

Design and implement a circuit that takes a 4-digit decimal/hexadecimal number (so each number is 4-bit) from switches in the Basys3 board and displays it on the 4-seven segment displays on the board.

2 Problem Description

The assignment requires you to design the following:

1. Design a combinational circuit that takes a single 4-bit hexadecimal or decimal digit input from the switches and produces a 7-bit output for the seven-segment display of Basys 3 FPGA board.
2. Extend the design to create a circuit that drives all 4 displays for displaying 4 digits together.

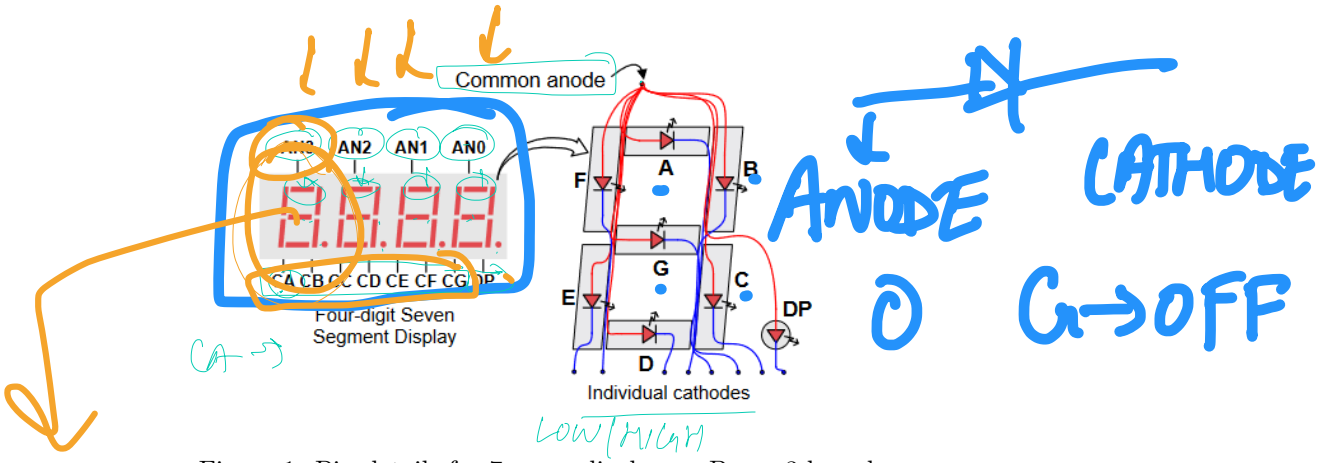

Figure 1: Pin details for 7 seven display on Basys 3 board

2.1 Seven segment decoder

The first module to be implemented is for displaying a 4-bit number on a 7-segment display. Figure 1 shows the pin-out details for the one display. It has 7 cathode pins, 1 anode pin (connected to all 7 LEDs) and 1 pin for decimal point.

- To switch an **LED** ON, the anode should be driven **HIGH** and cathode **LOW**. In general, the LED display is activated via driving anode **HIGH** and the corresponding digit via varying cathode signals.
- If the input number is "0000", then all LEDs except G should be switched ON. Refer Figure 2.
- Similarly, for "0011" LEDs F and E will be OFF, and the rest will be ON. Refer Figure 3.

You have to take the input from the switches present on the board (4-digits use 4-switches). Note that **on the Basys 3 board anode/cathode are ACTIVE LOW pins** (i.e., LOW = ACTIVE, HIGH = INACTIVE). For details, refer to the Basys 3 reference manual.

Design a combinational circuit for which the inputs are the four bits of the given decimal number and outputs correspond to the 7 cathode pins. The following need to be completed:

- Create a Truth Table with 4 inputs and 7 outputs. Using this, find out the minimized combinational logic to drive the 7 output signals.
- Implement a decoder module in VHDL, with input as a 4-bit number and output as a 7-bit cathode signal and a 4-bit Anode signal. To test your implementation on the board, you can use the slider switch for entering the number.

NOTE: use of VHDL case/if-else for this module is not allowed. Use elementary operations (AND, OR, etc.)

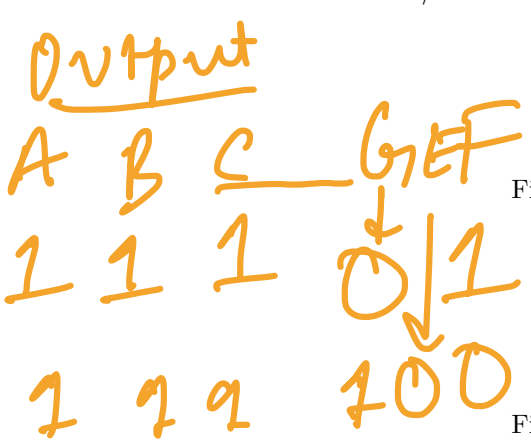

Figure 2: Output for input number "0000"

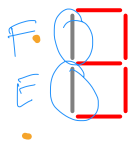

Figure 3: Output for input number "0011"

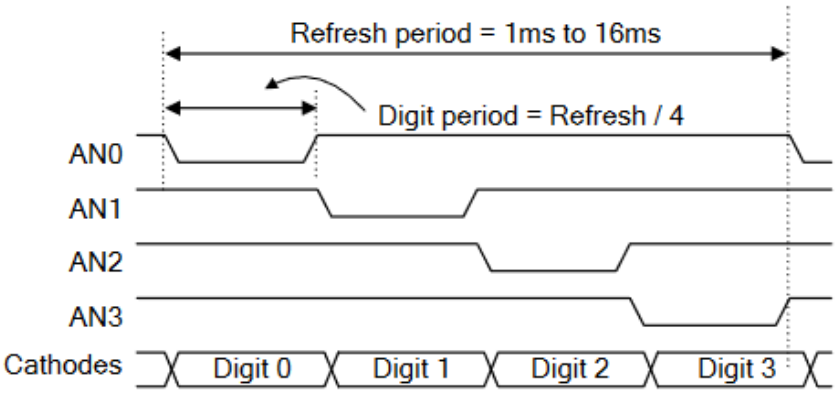

Figure 4: Timing details for 7 seven segment displays

2.2 Driving all four LED displays

As shown in Figure 1, 7 cathode pins are common for each LED display. This means only one display can be activated at a given time. To display a separate digit on each LED display, the corresponding anode signal needs to be activated in a cyclic manner. To avoid flickering, refresh rate should vary between 1kHz - 60Hz (1-16 ms period). Figure 4 shows the timing details and the signal waveforms. For more details, refer to the Basys 3 reference manual.

Using the module designed in section 2.1, create a 4:1 multiplexer module and timing circuit to drive the LED displays. The following need to be completed (Figure 5):

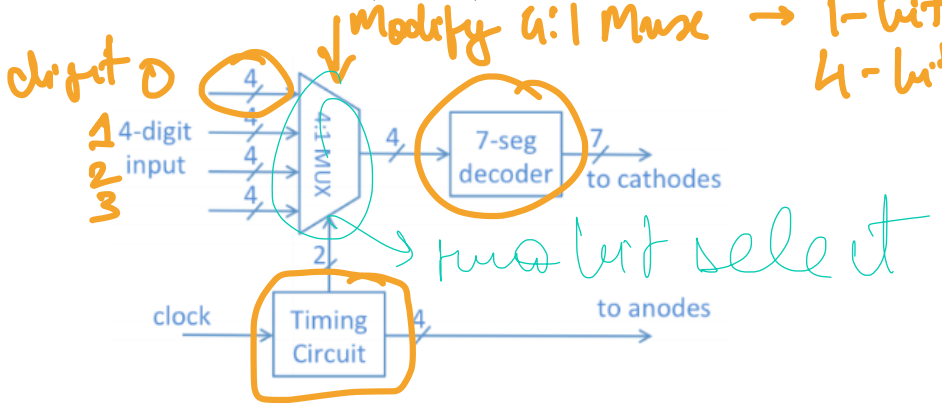

Figure 5: Block diagram for driving four 7 seven segment displays

- Multiplexer module with four 4-bit inputs from slider switches and output going to 7-segment module.
- Timing circuit to drive Anode signals and select signal of multiplexer.

You need to reduce the frequency of the onboard 100 MHz clock to the suitable display frequency.

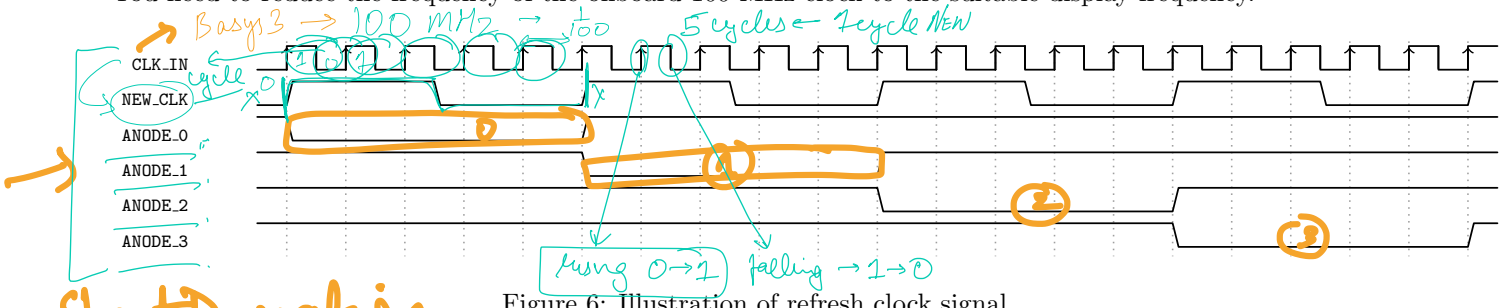

Figure 6: Illustration of refresh clock signal

A CLOCK signal is a periodic square (signal CLK_IN in the waveform of Figure 6) that is useful in digital circuits – actions can be synchronised with the *rising edge* (signal going from 0 to 1) or *falling edge* of a clock signal. The Basys 3 board has an onboard 100 MHz clock. You need to reduce the frequency of this clock signal so that it is usable in the display circuit.

Figure 6 illustrates an example showing the number of clock cycles in **CLK_IN** that are required to make new signal **NEW_CLK**. From the illustration, we have 5 cycles for 1 cycle in **NEW_CLK**. For that 1 clock cycle period, the **ANODE** signal for digit 0 will be LOW. Similarly, **ANODE** signal for other digits will be 0 in a cyclic manner. Your task is to:

- Select appropriate display/refresh clock
- Determine the number of clock cycles in 100 Mhz for the refresh clock

To implement the above logic, you will need to use the VHDL process construct. A process allows you to use IF-ELSE and CASE statement within it. Refer to *sample_process* below; it is triggered whenever signal **CLK_IN** changes. **rising_edge** is used to detect a change of signal from LOW to HIGH. Whenever a rising edge is detected, the **count** signal is updated; this is reflected on the **Output** port signal. Figure 7 shows the timing diagram for the *example process*. **Output** is a 2-bit standard logic vector, which is incremented on every rising edge of the **CLK_IN**, wrapping around from 11 to 00.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity P_block is
    Port (
       clk_in  : in STD_LOGIC;
       Output : out STD_LOGIC_VECTOR (1 downto 0);
    );
end P_block;

architecture Behavioral of P_block is
    signal count : STD_LOGIC_VECTOR (1 downto 0) := "00";
begin
    --Example Process
    Sample_process: process(clk_in)
    begin
      if rising_edge(clk_in) then
         count <= count + 1;
      endif
    end process;
    Output <= count;
end Behavioral;
```

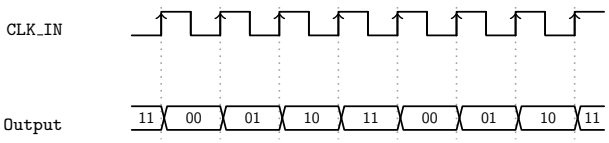

Figure 7: Timing diagram for the example process

You are provided with an initial template for the timing circuit module.

- Decide on the value of constant integer **N**.
- Add code for the process blocks. You can add or remove a process blocks if necessary.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Timing_block is
    Port (
        clk_in  : in  STD_LOGIC;  -- 100 MHz input clock
        reset   : in  STD_LOGIC;  -- Reset signal
        mux_select : out STD_LOGIC_VECTOR (1 downto 0);   -- Signal for the mux
        anodes : out STD_LOGIC_VECTOR (3 downto 0) -- Anodes signal for display
    );
end Timing_block;

architecture Behavioral of Timing_block is
    constant N : integer := ;-- <need to select correct value>
    signal counter: integer := 0;
    signal new_clk : STD_LOGIC := '0';
begin
    --Process 1 for dividing the clock from 100 Mhz to 1Khz - 60hz
    NEW_CLK: process(clk_in, reset)
    begin

    end process;
    --Process 2 for mux select signal
    MUX_select: process(new_clk)
    begin

    end process;
    --Process 3 for anode signal
    ANODE_select: process(mux_select)
    begin

    end process;
end Behavioral;
```

3 Assignment Submission Instructions

General assignment instructions that need to be followed for all assignments: only one partner needs to submit. Mention all team member names and entry IDs during the submission.

1. Name the submission file as entryNumber1_entryNumber2.zip or entryNumber1.zip
2. Go to Gradescope via moodle and upload the file under Hardware Assignment 2.
3. Only one submission per group is required. Gradescope will allow you to select the group partner.
4. The following files should be part of the zip folder:

- Source files (.vhd) including test bench files.
- Constraint Files (.xdc) for the final module.
- Bit files (.bit) for the final module.
- Report as a .pdf file (handwritten report will be rejected). The report needs to state:
  - your design decisions (if any)
  - truth table and logic minimization
  - simulation snapshots
  - schematic snapshot
  - resource utilization table (including resource counts: Flip-flops, LUTs, BRAMs, and DSPs)

4 Resources

- IEEE VHDL Reference Manual : `https://ieeexplore.ieee.org/document/8938196`
- Basys 3 board reference manual: `https://digilent.com/reference/_media/basys3:basys3_rm.pdf`
- Online VHDL simulator: `https://www.edaplayground.com/x/A4`