Seven-Segment Display

EE332: Digital System Design

Lab3

Objectives

The objectives of this laboratory are the following:

- To become familiar with the seven-segment display on the Nexys4 board
- To design a circuit using decoders and multiplexers that drives the seven-segment display on the Nexy4 DDR board
- To implement a 8-digit hex-to-7-segment decoder on the Nexys4 DDR FPGA prototyping board

Seven-segment displays are commonly used as alphanumeric displays by logic and computer systems. A seven segment display is an arrangement of 7 LEDs (See below) that can be used to show nay hex number between 0000 and 1111 by illuminating combinations of these LEDs. Seven-segment displays come in two flavors: common anode and common cathode. A common anode 7-segment display has all of the anodes tied together while a common cathode 7-segment display has all the cathodes tied together.

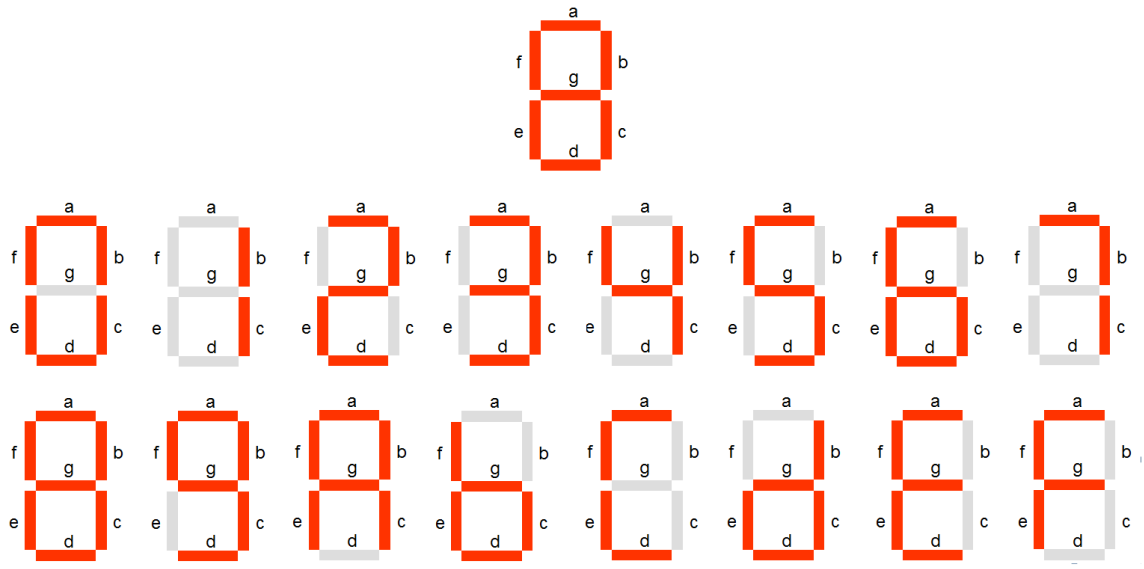

Fig. 1 A 7-segment display contains seven light emitting diodes (LEDs)

Seven-segment Displays on the Nexys4 DDR Board

The nexys DDR board has an eight-digit 7-segment display. Each seven-segment display consists of seven LED bars and a single LED round (for the decimal point), as shown in the figure below. You can reference the Digilent Nexys4 DDR Board Reference Manual for more information.

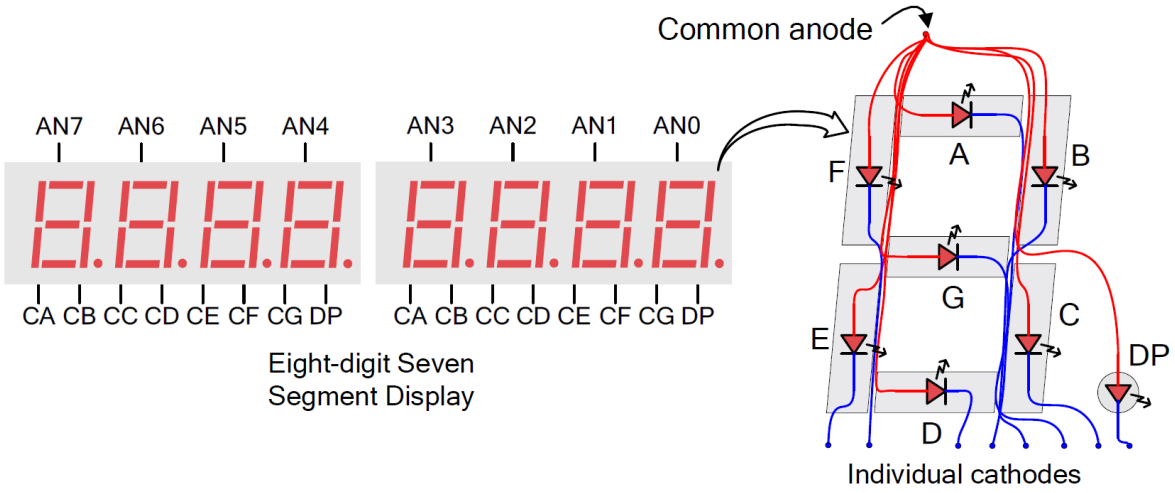

Fig. 2 Common anode circuit node

The Nexys4 DDR board uses the common anode method for its displays. This means that all the anodes are tied together and connected through a PNP transistor to +3.3V, as shown in the above figure. A different FPGA output pin is connected through a 100Ω current-limiting resistor to each of the cathodes, a – g, plus the decimal point. A control signal of 0 to a cathode will turn on the LED segment and a signal of 1 will turn it off. A hex-to-7 segment decoder takes a 4-bit input (a Hex digit) and generates the corresponding 8-bit pattern (including the decimal point) to light the appropriate LED segments in the display.

The table below shows output cathode values for each segment a – g needed to display all hex values form 0 – F.

x	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

1 = off

0 = on

Hex-to-7-Segment Decoder

To display hexadecimal digits on a 7-segment display, we need to design a hex-to-7-segment decoder (called hex7seg), whose input is a 4-bit number (x[3:0]), and outputs are the 7-segment values a – g (LED[7:0]) given by the truth table above. You may use, for example, a **case** statement in a process to implement the decoder.

```
case x[3:0] is
    when “0000” =>
        LED <= “00000011” -- LED[0] is for the decimal point
    when “0001” =>
        LED <= “10011111”
……
end case;
```

Multiplexing 4 Hex-to-7-Segment Displays

As described in the Nexys4 DDR User’s Guide, the Nexys4 DDR board designers saved FPGA pins by wiring the eight seven-segment displays to the same set of (cathode) control lines. The user can display eight separate characters “simultaneously” by time-multiplexing the seven-segment display control lines at a fast enough rate so that the human eye views all four of the displays as ON and displaying the correct value. Each digit is illuminated just one-eighth of the time, but because the eye cannot perceive the darkening of a digit before it is illuminated again, the digit appears continuously illuminated. If the “refresh” rate is slowed to about 45 hertz, the display will start flickering.

For each of the eight digits to appear bright and continuously illuminated, all eight digits should be driven at least once every 1 to 16 ms, for a refresh frequency of about 1 KHz to 60Hz. For example, in a 62.5Hz refresh scheme, the entire display would be refreshed once every 16ms, and each digit would be illuminated for 1/8 of the refresh cycle, or 2ms. The controller must drive low the cathodes with the correct pattern when the corresponding anode signal is driven high.

To illustrate the process, if AN0 is asserted while CB and CC are asserted, then a “1” will be displayed in digit position 1. Then, if AN1 is asserted while CA, CB, and CC are asserted, a “7” will be displayed in digit position 2. If AN0, CB, and CC are driven for 2ms, and then AN1, CA, CB, and CC are driven for 2ms in an endless succession, the display will show “71” in the first two digits. An example timing diagram for a four-digit controller is shown below.

The figure below shows an example timing diagram for a four-digit controller.

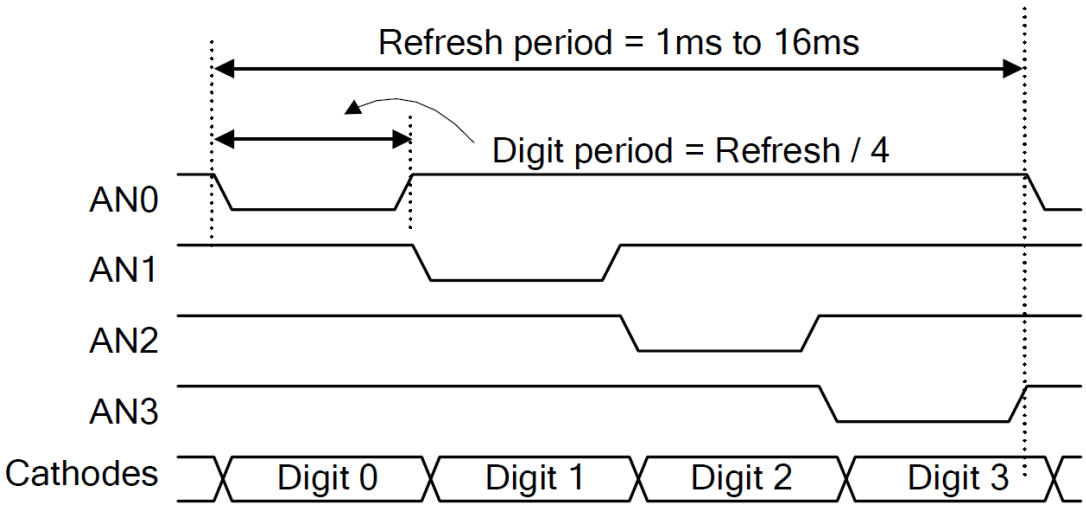

Fig. 4 Four digit scanning display controller timing diagram.

Overall structure

Our 8-digit seven-segment controller will take a clock and eight characters (4-bit each) as inputs, and will write the seven segment control signals (CA to CG) as well as the eight anode signals (AN0 to AN7) to display all eight characters simultaneously. The figure bellow shows a block diagram of a possible implementation of this controller.

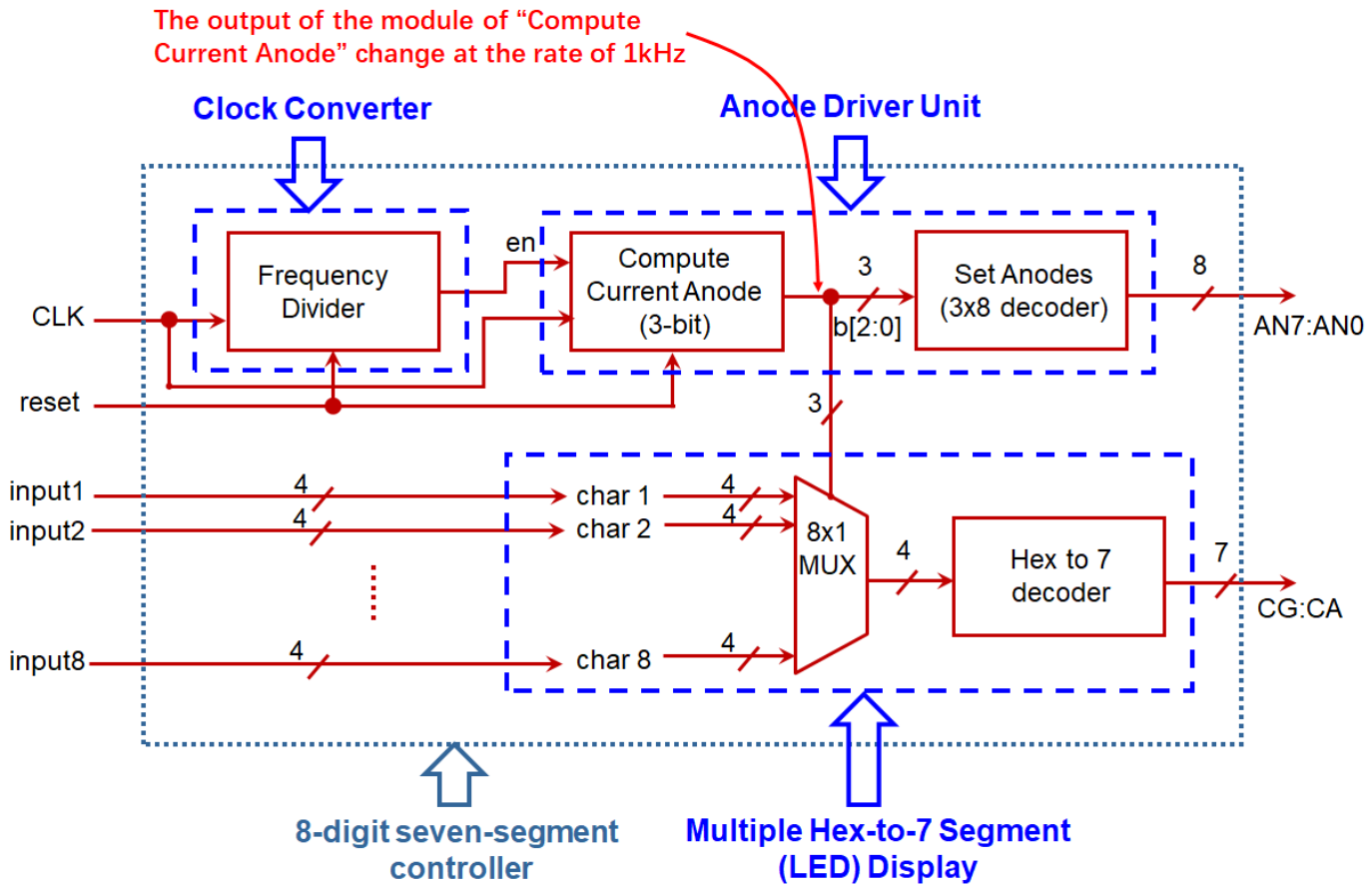

Fig. 5 A block diagram of a possible implementation of the seg7-driver

Anode Driver Unit

As shown in Fig. 4, the output (AN7:AN0) is activated one at each digital period and **b[2:0]** in Fig. 5 corresponds to the active signal 000, 001, 010, …, or 111. To have a refresh frequency of 1 KHz, the sequence **b[2:0]** can be generated by a 3-bit counter that increases 1 for every 10^5 clock tics, as the input clock have a frequency 100MHz. Thus, the output of a 3-bit counter cycles from 000 to 111, incrementing the count with each 10^5 clock tics. After **b[2:0]** = 111 is generated on the eighth clock cycle, the count will "rollover" back to 000 and continue. To generate the individual anode control signals AN0-AN7, you can use a 3-to-8 decoder with **b[2:0]** as the inputs. However, remember that AN0-AN7 need to be active LOW outputs.

Pre-Lab Preparation

Part 1: Reading

Read pages 19-20 in the Nexys4 DDR Board Reference Manual.

Part 2: Hex-to-7-Segment Decoder Using Case Statement

1. Design a hex-to-7-segment decoder using the case statement.
2. Perform a functional simulation, and a post synthesis simulation of the circuit. Include the results in your prelab report.
3. Create a symbol for the hex7seg.
4. Bring your VHDL codes.

Part 3: Multiple Digits Display

5. Design an anode driver unit that generates the anode driver signals.
6. Design an 8-digit seven-segment controller.

Pre-Lab Report

In your prelab report, include the following:

• Truth Table, VHDL program, and simulation results for the hex-to-7-segment decoder with case statement, with all possible values of inputs x[3-0].

• RTL block diagram, VHDL program, and simulation results for the anode signals. Incorrect or incomplete designs and VHDL programs will not receive full credit.

• The block diagram of your hierarchical design of the overall structure.

If you have any problems with VHDL syntax and other pre-lab related issues, please resolve them before coming to the lab. Your TA may not be able to help you with these issues during the lab session.

In-Lab Procedure

Bring flash drives to store your data.

Ask questions regarding any procedures about which you are uncertain. Implement the complete design (synthesize, and Place & Route) of your two designs using the Xilinx vivado tools. Program the FPGA using the bit-stream file which is generated in the process. The eight 4-bit number to be displayed on the eight 7-segment displays on the Nexsys4 DDR board are the following:

1. The first 4 of the 16 switches on the Nexys4 DDR board, denoted as A
2. The second 4 of the 16 switches on the Nexys4 DDR board, denoted as B
3. The third 4 of the 16 switches on the Nexys4 DDR board, denoted as C
4. The fourth 4 of the 16 switches on the Nexys4 DDR board, denoted as D
5. A + B (ignore the effects of overflow)
6. A – B (ignore sign & underflow)
7. C + D (ignore the effects of overflow)
8. C – D (ignore sign & underflow)

For checkoff, you will show the following:

1. Show the design working on the board.
2. Demonstrate the input patterns you have used in the test bench. Show that your simulation results above match the observed waveforms.
3. Demonstrate your 8-digit display with the clock at 1 KHz.

Post-Lab Report

Write up your code, schematics, and lab procedures. Demonstrate the correctness of your designs through your pre-lab simulations and note any differences between what you simulated and how the circuits behaved in the lab.

Extra Design

Making use of the seven-segment display drive as a component, design a digital clock. You may design all the modes, functions, interfaces.