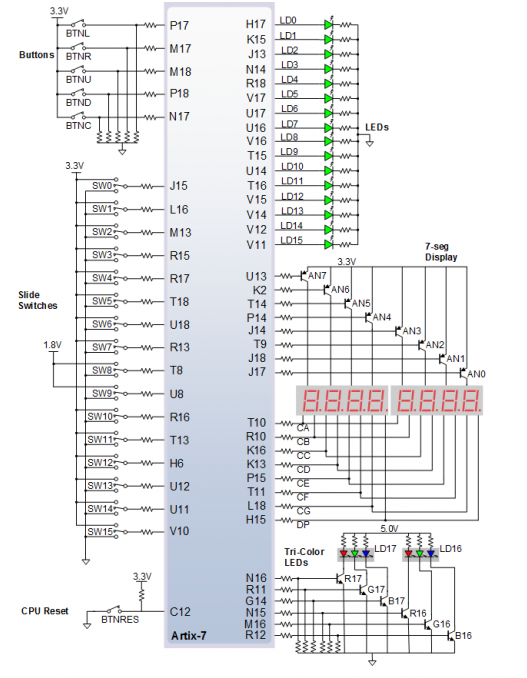
Marcin Wisniowski

Lab 1, Project 1 – Seven-Segment Decoder:

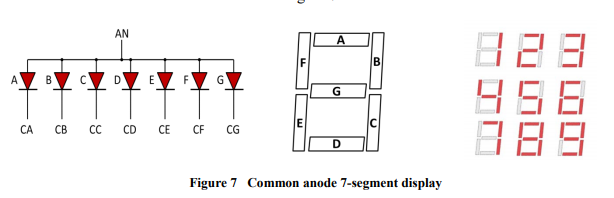
The following lab goes through teaching how to use Xilinx Vivado 2019.2 and programming the Nexys A7 board with some simple code that utilizes the 7 segment display.

The Nexys A7 board connects to a host computer via a USB interface that allows for code to be loaded onto the FPGA board and allows for reprogramming. In this lab, the user is dependent on the 7-segment display as well as the switches at the bottom of the board to display values across the LED 7-segment display.

In order to effectively use the board, the coder must draw parallels in writing the system architecture in VHDL with the actual circuit components on the board itself. In order to do this most effectively, the reference manual can be used to draw these parallel connections:



The 7-segment display is so called that because it uses 7 individual lines that can be high or low to create the digits and letters that we specify.



In the code, this relates to the CA through CG lines which are further explained as the L18, F18, D17, etc. lines seen on the bottom right of the Spartan A7 FPGA board schematic above. We will use these correlations in the code to create the specific design we want. Furthermore, a final DP line is used to illuminate the display in the end to see the specified design, allowing for an empty field to also be present.

The Nexys2 reference manual states that for the digits to appear bright, they should be refreshed every 1 to 16 ms. This allows all the digits to appear to be on at the same time, when they are actually flipping on and off in a pattern.

In the VHDL code, a large select statement is made with the following variables used and the reasons why:

variable dig – dig is two digit bit (00, 01, 10, 11) that is mapped to R17 and T18 in the FPGA I/O constraints file. This relates to two switches that chooses which digit is illuminated by relating to the anode segment in the switch statement.

variable data – data is a 4 digit bit that is mapped to G18,H18,K18,K17 which are the far right switches. This allows us to decide what digits are being displayed through the code that maps the 7 digit (segment) vector seg to the data chosen.

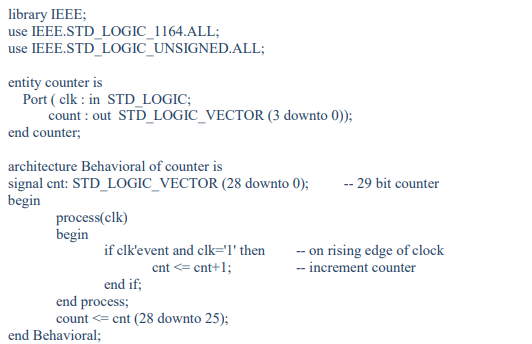
variable anode – in the nexys2 manual, all the segments of the display are mapped to a single common anode that can be used to power on and off the displays. Dig chooses these and when set to one, they open a transistor switch which allows for the LEDs to be powered on. To illuminate a segment, the anode should be driven high, while the cathode is driven low, however because of transistors being used, this is inverted. Therefore, in the code, when an anode segment is set to 0, it will be active. Because the anode is written in Big-Endian, this also appears to be backwards, as the last digit in the code being 0, i.e. “1110” relates to the first digit being illuminated in the displays.

variable segment – the variable segment holds the data for which segments should be powered on to create the specific displays. This is paralleled with the data input to make sure that when input is 5, the correct segments are lit up to create the number 5. These use the CA 🡪 CG connections.

Lab 1, Project 2- Hex Counter:

The leddec, LED decoder file is still used from the last project but now, a new counter is built on top of that file.

A new counter entity is made to the specifications below:

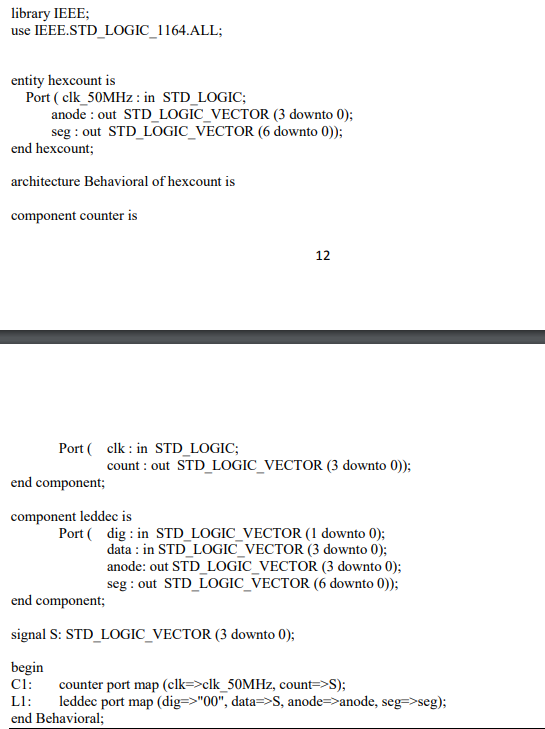


The clk input is mapped to the E3 port, which is a 100 MHz crystal oscillator on the Nexys A7 board. This is set to 50 MHz for this project in order to best see the changes that are occurring on the board.

In the code, the counter architecture is specified to increment a cnt variable on a rising edge of the clock. This is done by specifying that the clk needs to have an event, clk’event and that the clock also needs to be clk=’1’ at that time, therefore the clock needs to have an event where it ends as a positive value.

Interestingly, the clock is running at 50 MHz and so this counter variable is incrementing 50 million times a second. In order to fully see the changes, a second variable count is made to only keep track of the 4 most significant bits of the cnt variable (bits 28,27,26,25). Because we now need to wait until the 25th bit changes, we can divide 50 Mhz / 2^25 and get that the counter will update every 1.5 Hz, or once every 0.66 seconds.

At the moment, however, only a count variable is being changed, but there is no reference of this to any 7-segment displays. Therefore, another entity must be made.



The hexcount file takes in the components previously made in the other two files and maps them to the appropriate areas allowing for the counter to affect the LED outputs.