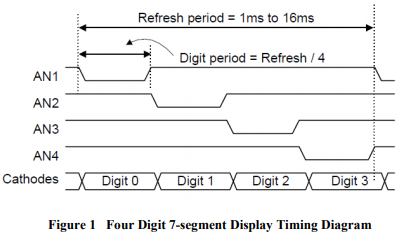
Marcin Wisniowski

Lab 2, Project 1 – Four Digit Hex Counter

In the first lab, the FPGA board used the 7-segment display to show one value on all of the values within it. This time, the output will be split and allowed to count independently of each other.

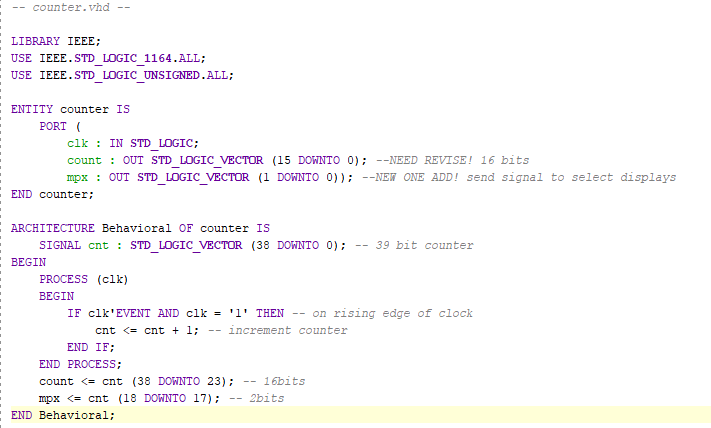
In the 7-segment display, four digits are combined together to share the same cathode lines CA~CG. In order to therefore display on all of the displays at the same time, the output must be drove across each one independently and shifted across each relatively quickly in time. This means, that in actuality, only one display is on at a time, but it is quickly multiplexed among all of them to appear simultaneous. Each digit is only illuminated for a couple seconds.



In order to accurately display the values, we need to create a signal to read through each of the values and multiplex the values. The sequence will repeat at a frequency which is fast enough to eliminate any visual flicker in the 7-segment displays.

There exist three important files in order to have this system work effectively. The counter, hexcount, and leddec files.

Counter File:



This counter file creates an entity with a clk, count, and mpx values.

clock input keeps track in accordance with the system clock as an event driver for system changes to occur.

During these clock frames, we check for a rising edge and increment the counter allowing it to loop through the cnt variable. While this value is 39 bits long, it is later set to only display the 16 most significant bits in another variable count. Mpx is also set at a specific timeframe of changing, in this case the 17th and 18th most significant bits in order to have a constantly looping value of 00, 01, 10, 11 but at a set increment of time between each.

Using this mpx value, we could then multiplex to choose which value should be chosen into which 7-segment display. This will loop through all the different displays to make all of them count separately.

In the hex counter file, a higher level architecture is built using the LED decoder file and the counter file in order to create a hex counter file which accurately displays the numbers given the correct mpx and S (count) value that are previously received in the counter file. This in turn helps to refresh each digit specifically and show different numbers on each of the four displays.

Finally, a constraints file is used to port specific variables to input pins in order to allow for control of the system.

In order to get an FPGA to be more effective, it is also important to be able to create memory configuration files and allow the device to store the program on itself in order to boot up without needing to be connected to a PC. In order to do this, the program was normally generated with the bitstream generation.

Afterwards, the program device memory part was chosen from a list of components and a .mcs or memory configuration file was created to be placed onto it. The Vivado program allowed for the .mcs file to be uploaded into the memory part by programming that part specifically. Once that was done, the hardware manager could be closed and the program was able to run off the device.

A jumper had to be moved from JTTAG to QSPI in order to get this functionally to work from memory.