Lab 9 – Datapath and Control for the 4-digit, 7-segment Displays

Introduction

In Lab 3 you designed and tested two decoders: a seven segment decoder and an anode decoder. However, they were implemented and tested on a single digit. Now that you understand how to design with sequential logic, you will add the time division multiplexing required for driving all four display digits.

In the Basys 3 FPGA Board Reference Manual, you recall that the 4-digits of the 7-segment display block share the cathodes, but have unique anodes for each digit (also illustrated in the Lab 3 write up). To drive the full four digit display, the cathode pattern for digit 0 must be on the cathode bus when the anode signal for digit 0 is active (low). The pattern for digit 1 must be on the cathode bus when the anode signal for digit 1 is active (low). And so on for digits 2 & 3. You can see how this should function from Figure 19 in the reference manual of the Basys 3 board, illustrated below.

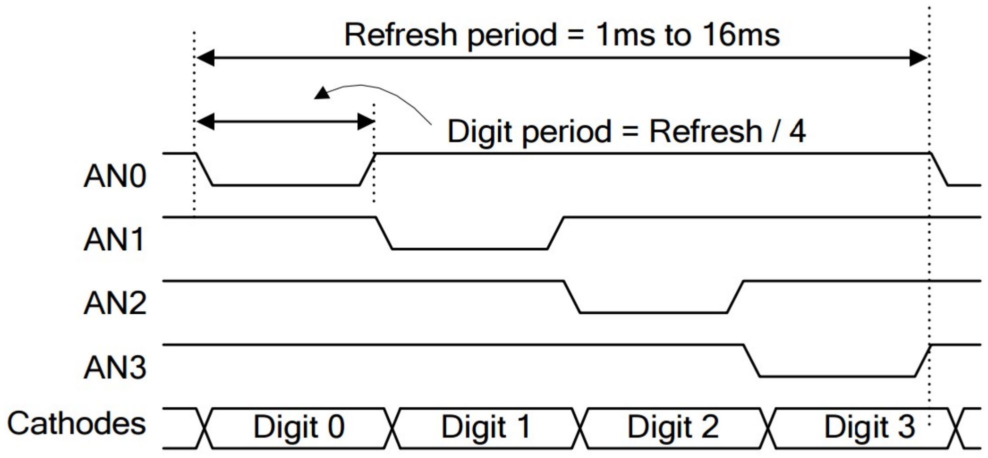


Figure 19. Four digit scanning display controller timing diagram.

Reference Timing Diagram from Basys 3 FPGA Board Reference Manual

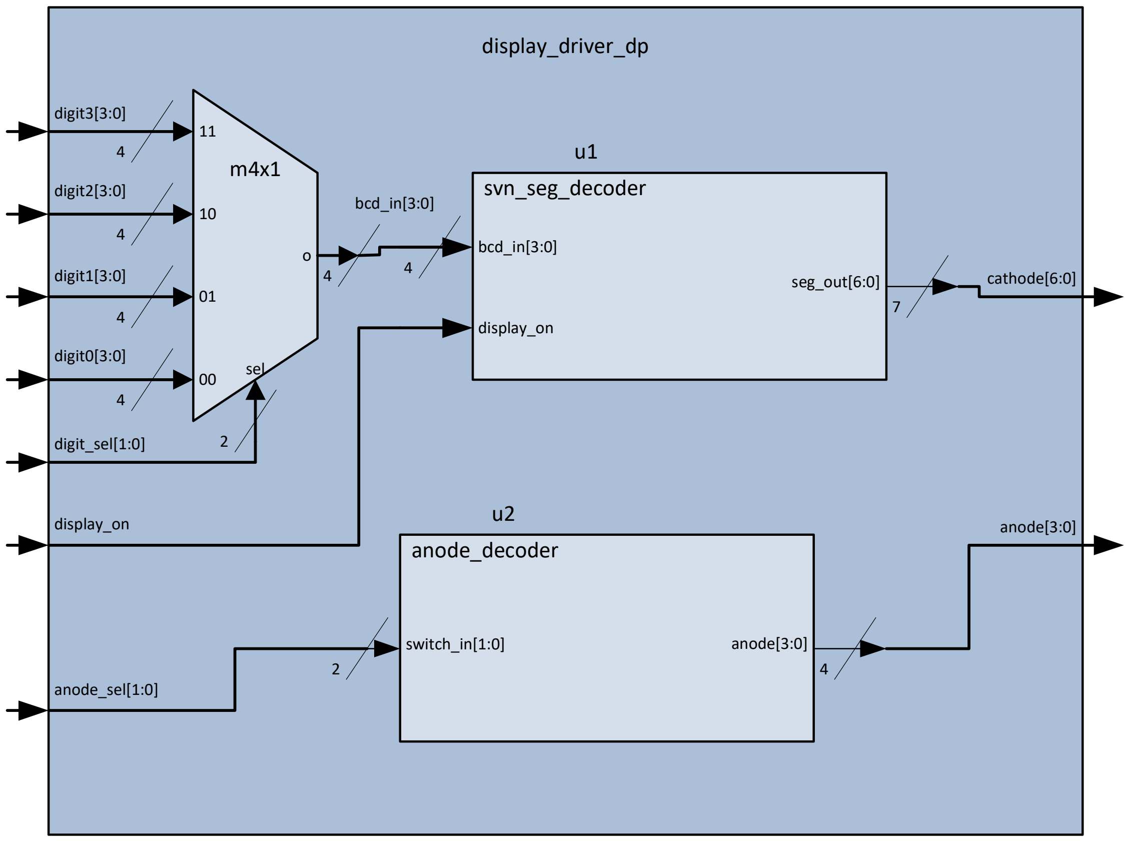
For this design, we will use a 1 KHz switching frequency (1 msec), for a total refresh period of 4 msec. The design will be demonstrated with a fixed, unique pattern displayed on each of the digits. You will also verify the time division multiplexing timing with an oscilloscope.

Detailed Specification:

A block diagram for each module is illustrated below. The design has several levels of hierarchy, as illustrated from the diagrams below. They are basically split into ***datapath*** and ***control*** blocks, as illustrated in the final block diagram.

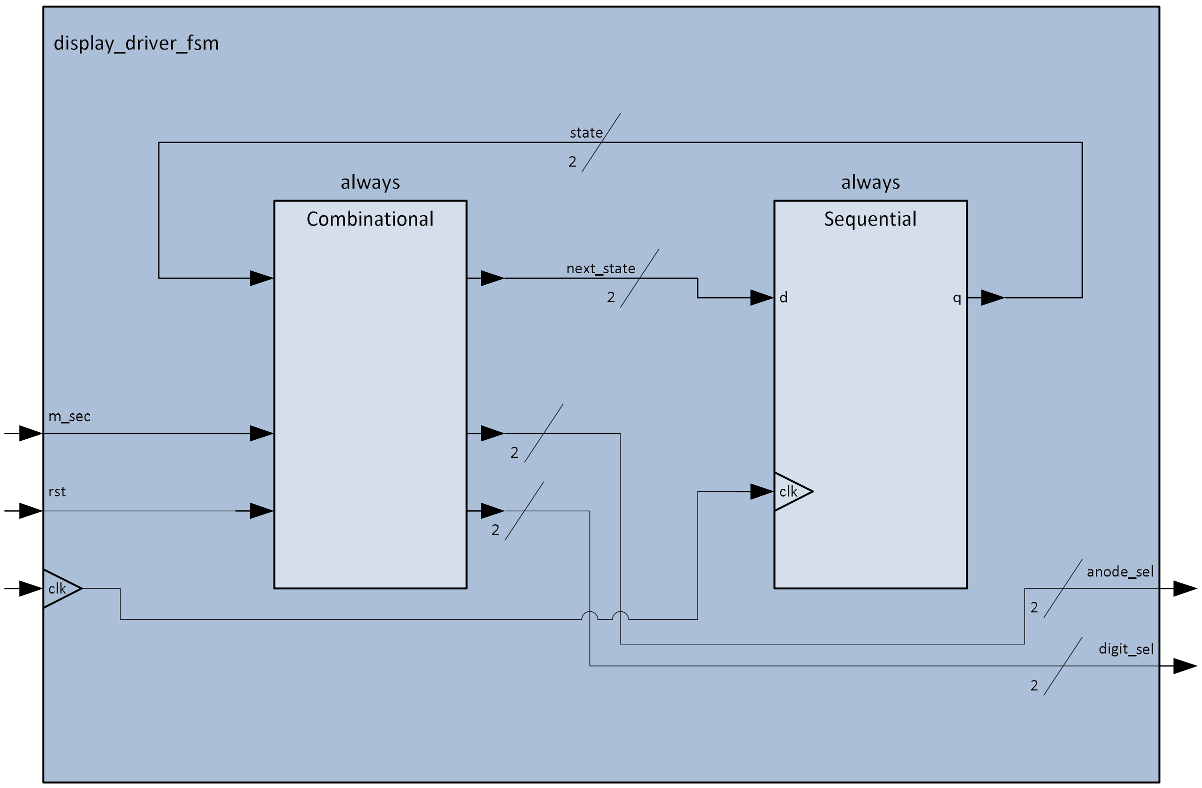
Datapath

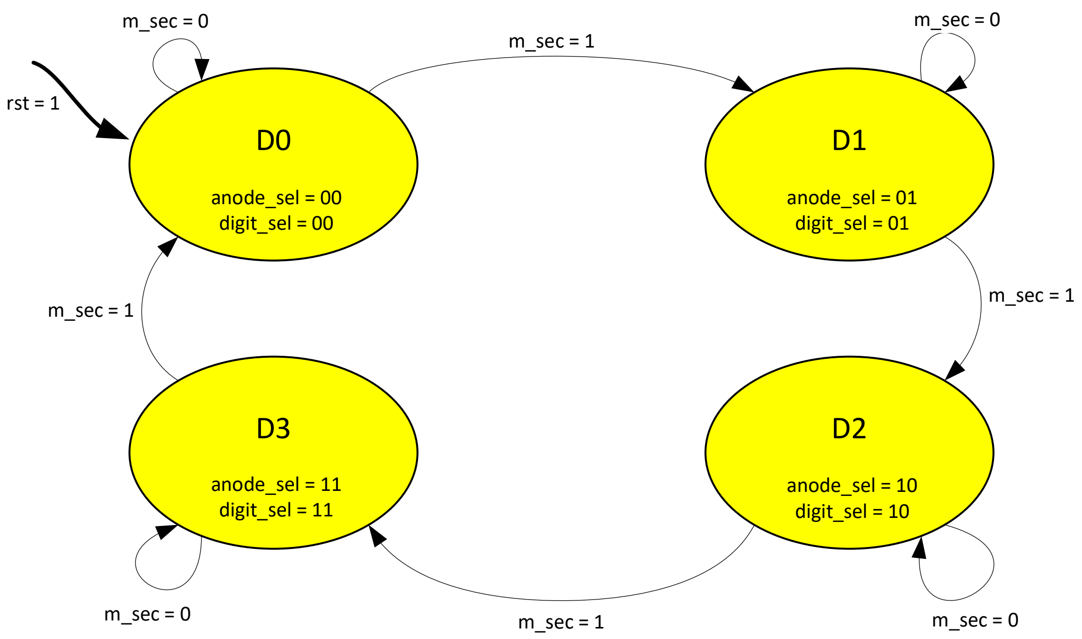
The datapath block diagram is shown below. You will use your seven segment decoder and anode decoder modules from Lab 3 to this project. You will instantiate these (***u1*** & ***u2***) and add a 4 to 1 multiplexer for the 4-bit digits given as inputs to the module.



Finite State Machine (FSM)

The FSM module, ***display_driver_fsm***, has three inputs: the 100 MHz clock (***clk***), a timed control signal that defines when to change states – this is 10 nsec wide (1/100 MHz), one pulse every millisecond (***m_sec***), a synchronous reset signal (***rst***) that initializes the state machine to state ***D0***. Thus, the FSM transitions to a different state every millisecond. Four states are defined that will create the ***digit_sel*** and ***anode_sel*** output control signals. These specifications are illustrated in the block diagram and state diagrams below.



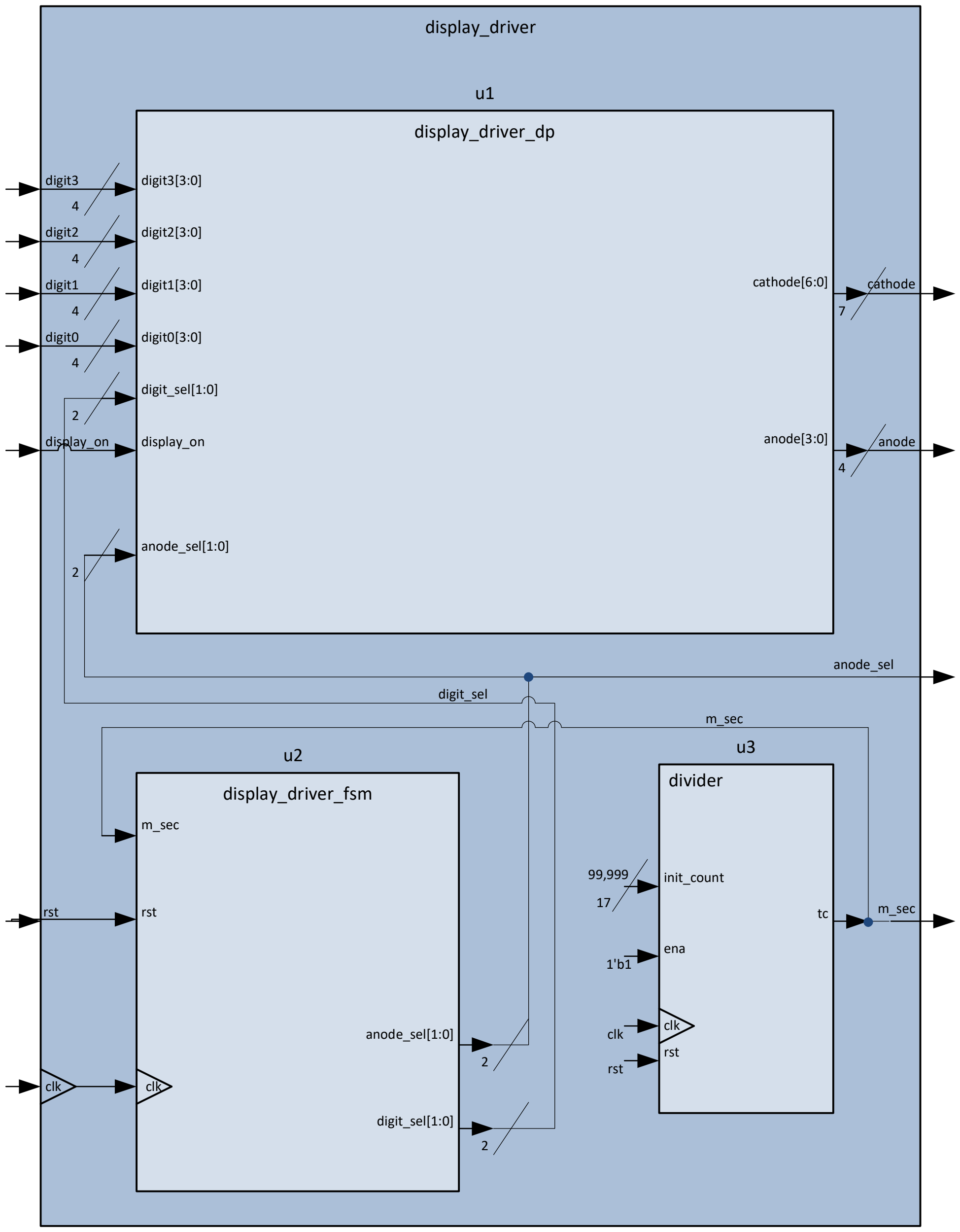


Integration

The integrated display driver combines the ***datapath*** and ***control*** blocks by connecting the control signals as illustrated in the block diagram below. In addition you will reuse the divider module from your Lab 7 design and instantiate it to create the ***m_sec*** signal. This will generate a

10 nanosecond wide pulse that repeats with a frequency of 1 KHz. So this divider will divide 100 MHz by 100,000 to get 1 KHz.

The block diagram for this module is shown below. In the implementation mode, this is instantiated by an i/o wrapper so it will be implemented in hardware.



The *anode_sel* signals are brought out as outputs. The top level module connects them to *PMOD connector A* so you can monitor them with an oscilloscope.

Design/Module

The functionalities for all of the modules for this lab are defined in the Detailed Specification section above. A testbench file is provided.

For hardware implementation, the ***display_driver*** module is instantiated in the top level i/o wrapper, ***lab9_top_io_wrapper.sv***. It is also instantiated in the top level test bench, ***tb_display_driver.sv***, for simulation/verification.

The test vectors for this design are contained in ***tb_display_driver.txt***, with only 10 input/output combinations tested. Each vector tests represents the values between 1 millisecond pulses (***m_sec*** signal). So the simulation time is 10 milliseconds.

As usual, the user constraint file (***lab9.xdc***) contains the mapping and signal naming between the pins on the FPGA and the signal names used at the top level i/o wrapper (***lab9_top_io_wrapper.sv***). Do not modify these files for this lab.

Design steps:

a) Complete the design for the display driver datapath, ***display_driver_dp.sv***, based on the above specification.
b) Complete the design for the display driver FSM, ***display_driver_fsm.sv***, based on the above specification.
c) Integrate these into the display driver by completing the module, ***display_driver.sv***, based on the above specification.

Simulation/Verification

There is one simulation command file (right click the ***display_driver.sim*** file and Run). It is used to simulate your *display_driver* module. It uses the testbench, ***tb_display_driver.sv***, to simulate your design at the top level. It will read the text file, ***tb_display_driver.txt***. None of these files should be edited.

You should look carefully at the timing simulation and observe that the control signals are indeed toggling every millisecond. I have not put together test benches for each module. I'm assuming that your seven segment decoder, anode decoder and divider modules were properly designed and tested in previous labs. If you have some mismatches, double check your definitions in the anode decoder. [This design assumes that control signal (2'b00) will activate the least significant digit, and 2'b11 activates the most significant.]

This is not a full test bench. It is only testing a subset of possibilities that will validate your design, as three of the components you are using have already been verified. The test bench compares the outputs and sends a message to the display every 1 millisecond. A successful simulation will have the output:

```
## run 10100us
Time: 1000000 ns
Time: 2000000 ns
Time: 3000000 ns
Time: 4000000 ns
Time: 5000000 ns
Time: 6000000 ns
Time: 7000000 ns
Time: 8000000 ns
Time: 9000000 ns
Time: 10000000 ns
Simulation complete - no mismatches!!!
```

Note how each line is separated by exactly 1 millisecond (1,000,000 nanoseconds). If this is not correct, your divider module is likely the problem.

Synthesis

Generate a bit file as you did in past labs (right click on: *lab9_top_io_wrapper.tcl* and Run). For this lab it will be called: ***lab9_top_io_wrapper.bit***. Copy it to your flash drive to load and run on your Basys3 board.

The wrapper has connected the ***display_on*** signal connect to logic 1 so the on/off part of the display is wired *always on*. The ***rst*** signal is connected to the center pushbutton. The seven segment display should display the settings of the 16 switches, 4 at a time, in hexadecimal.

Use the *PMOD interface* and breadboard to connect and observe the signals from pins 1 & 2 on the oscilloscope. These are your ***anode_sel*** signals. They should be toggling at a 1 KHz frequency. If you're not sure how these should look, observe them in the simulation timing diagram. The ***m_sec*** signal is connected to pin 3.

What happens when you press and hold button 0 (***rst*** signal)? Is this the behavior you expect based on your design?