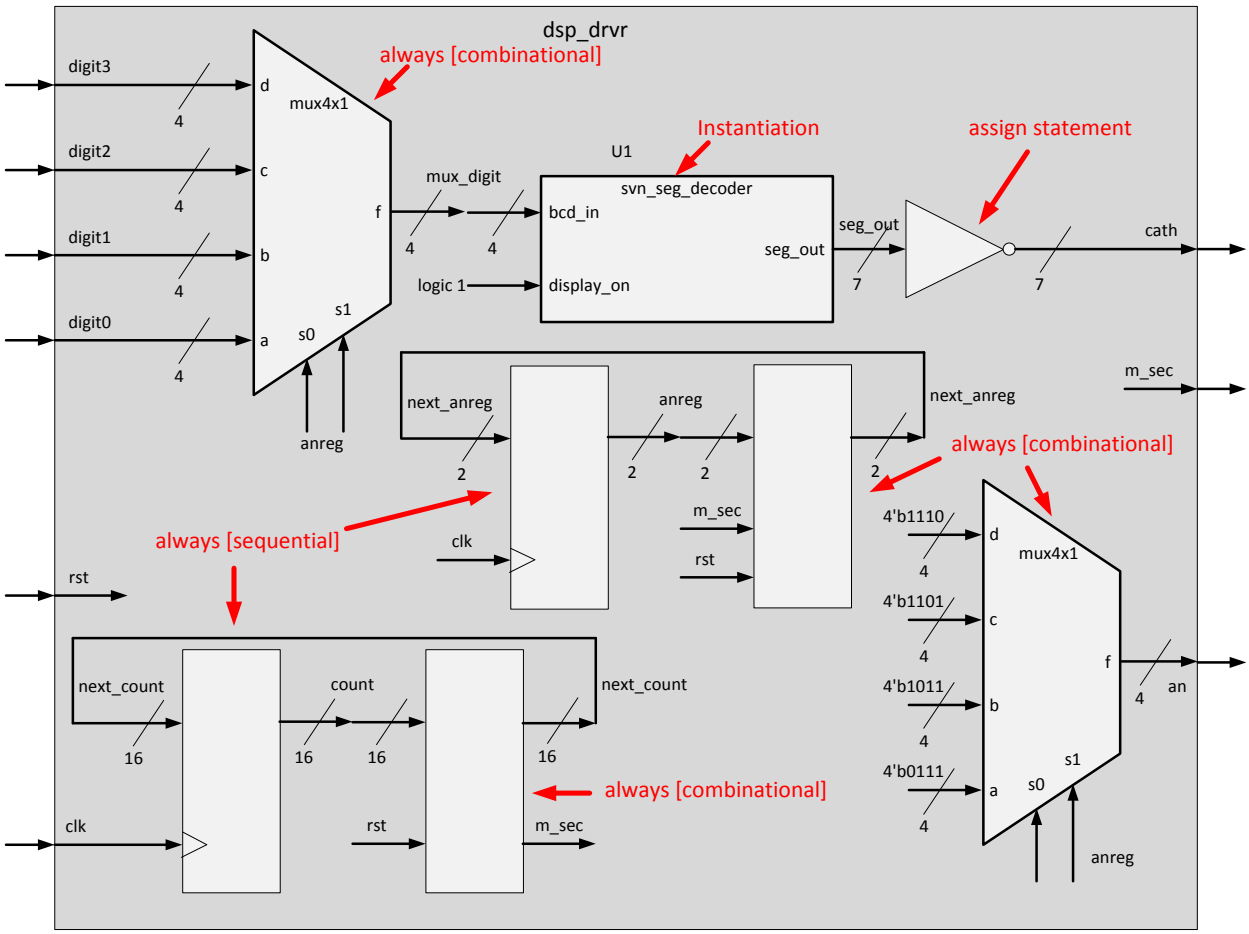
Lab 7 – Display Driver Integration and Timing Measurement

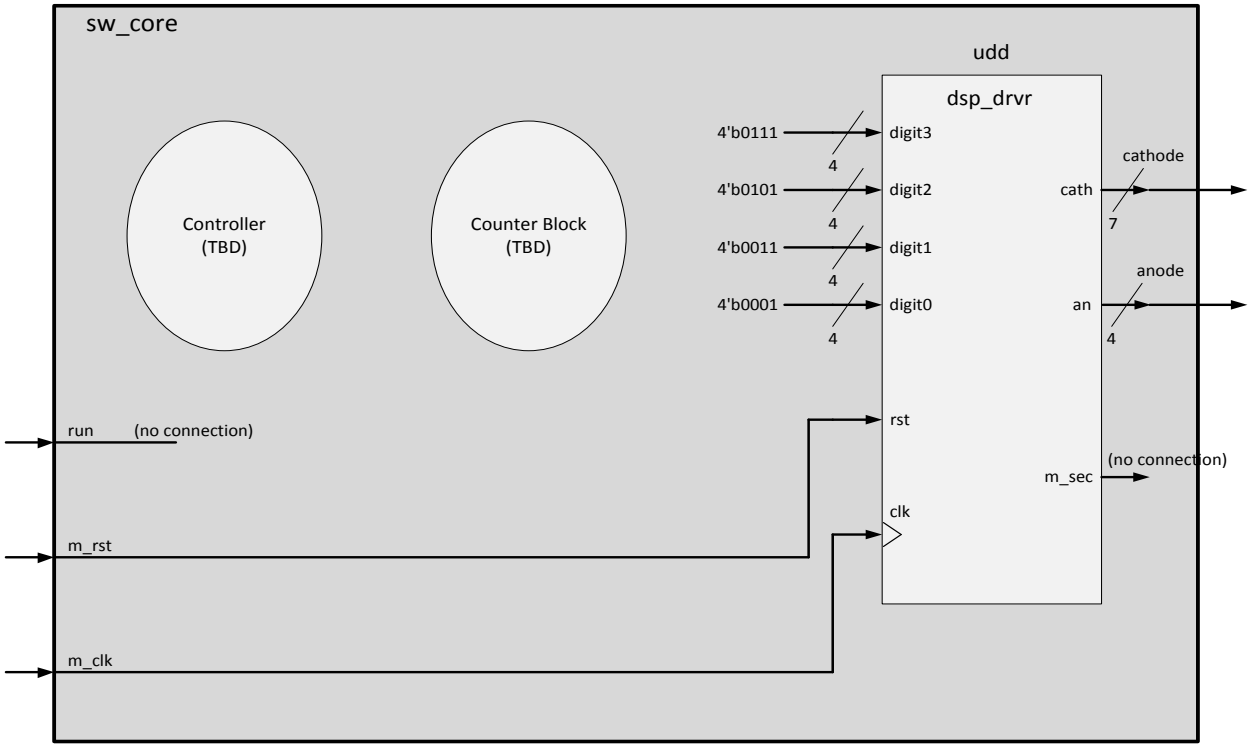
In this lab you will integrate your seven segment decoder from either Lab 3 or 4 (your preferred implementation) into the display driver module that was developed in the class lectures. You will also make some actual timing measurements using a dual trace oscilloscope, verifying that your design meets the intended specification.

The block diagram below illustrates the display driver module (*dsp_drvr*) that was developed in the class lectures (review your class lecture notes for more information). The bulk of this module will be provided to you, except for your seven segment decoder design.



You will also build a stopwatch core module called *sw_core*. This will be the high level module that will eventually contain all of the stopwatch design. For Lab 7 you will be only integrating the display driver module into it (via instantiation). In Lab 8 you will be building and adding a counter block to this module.

A block diagram of the *sw_core* module is shown below.



Take note of this module's i/o's. Also, the input digits (0-3) are hardwired to 1, 3, 5 and 7, respectively. Your final hardware implementation for this lab should display these numbers (or any other pattern you select). The *m_clk* signal is the master clock and will be connected to the board level 50 MHz clock signal. The *m_rst* signal is the master reset and will be connected to switch 0. The run signal is not connected to any internal circuitry at this time, but is connected to switch 1 at the top level.

Lab steps:

1. Open the *lab7* project in directory lab7. It will contain three design files: a) the display driver module, *dsp_drvr.v*, b) a testbench module for *sw_core*, *tb_sw_core.v*, and c) a top level wrapper for hardware implementation, *lab7_top_io_wrapper.v*.
2. Open *dsp_drvr.v* and instantiate your seven segment decoder, connecting it as defined in the display driver block diagram on page 1.
3. Add your preferred implementation of the seven segment decoder from your Lab 3 or 4 (*seven_seg_decoder.v*). Use: *Project->Add Copy of Source…*. Do **NOT** use: *Add Source*… – grab a copy of it rather than referencing it, in case you have to do some edits.
4. Create a new *sw_core* module using the New Source Wizard (*Project->New Source… Verilog Module*, etc.). Use the *sw_core* block diagram above to guide your design: a) instantiation of *dsp_drvr*, b) connecting it to the top level i/o's and c) hardwiring the digits. You must define the i/o's **exactly** as illustrated in the block diagram – otherwise it

will not correctly plug into the top level wrappers. When you are finished note in the hierarchy window how you now have four (now connected) levels of hierarchy, from the top level wrapper down to your seven segment decoder.

5. Simulate your design using the testbench wrapper: *tb_sw_core.v*. The text output from the simulation should represent your hardwired digits (seg_out) as illustrated below:

```
…
Finished circuit initialization process.
Anodes start - time = 20 ns
anodes: 0111 seg_out: 1001111

# run all
Anodes changed - time = 1000010 ns
anodes: 1011 seg_out: 1001111
Anodes changed - time = 2000010 ns
anodes: 1101 seg_out: 0000110
Anodes changed - time = 3000010 ns
anodes: 1110 seg_out: 0010010
Anodes changed - time = 4000010 ns
anodes: 0111 seg_out: 1000111
Anodes changed - time = 5000010 ns
anodes: 1011 seg_out: 1001111
Simulation complete!!!
Stopped at time : 5000010 ns …
```

   If you changed your 1, 3, 5 and 7 to something else, the seg_out's should change appropriately (remember, the seg_out pattern is inverted now). Look at the timing diagram of your simulation. Zoom to the full view. The testbench has run through 5 milliseconds of simulation – which is 5 cycles of the anode signals cycling. Open up the *anodes* bus and observe the individual signals. These should match our design specification as shown on page 15 in the *Digilent Nexys2 Board Reference Manual*.
6. After you are satisfied that your design passes the simulation test, implement the design in hardware. Your module *sw_core* should have automatically plugged into the top level wrapper, *top_io_wrapper*. Synthesize, then download the bit file and test the operation. You will get some warnings from the unconnected *run* signal, but this will be used in Lab 8. Remember switch 0 is connected to the master reset signal. When not reset, your hardware should display the static numbers, 1, 3, 5, and 7. Explain what happens when you activate the reset switch. Is what is displayed what you expected?
7. Oscilloscope measurement: The top level anode signals have been routed to external signals on connector JA (two-row 6-pin PMOD connectors – see page 15 in the *Digilent Nexys2 Board Reference Manual*). In particular, pins JA 1, 2, 3 and 4 have been connected to AN1, AN2, AN3 and AN4 as illustrated in Figure 10 in the *Digilent Nexys2 Board Reference Manual*. [Note the ground signals on pins 5 and 11 for connecting the

oscilloscope probe's ground. The TA will supply you with appropriate connectors for probing these signals and connecting to ground, etc.] You will be using a two channel oscilloscope to observe the four signals, but you should be able to see that they meet the desired design specification, as well as match what you observed in your simulation timing diagram. Measure the width of these signals. Do they match the design specification?

8. What happens to the signal when the reset switch is activated?

Extras:

- How would you change the design to slow the signals down by 10%? You can try a new design and verify it by observing/measuring the signals with an oscilloscope. You will not notice any difference in the LED's display output.