EE 445L - Lab 2: Performance Debugging

Joshua Bryant
jmb6357

James Morris
jsm3288

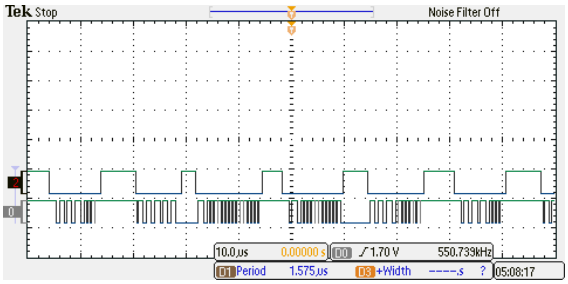
September 19, 2014

1 Objective

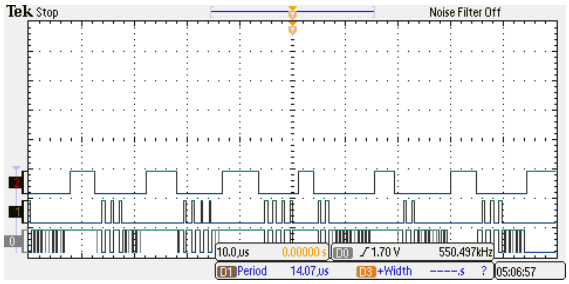
The five main objectives for this lab are: (1) developing software debugging techniques (2) passing data using a FIFO queue (3) learning how to use the oscilloscope and logic analyzer (4) observing critical sections, and (5) getting an early start on Lab 3 by writing a line drawing function. The specific software debugging techniques used for this lab include performance debugging in real time and profiling program activity.

2 Measurement Data

2.1 Part B and C



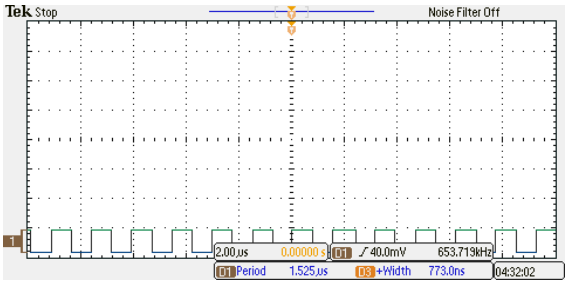
(a) Part B screen shot



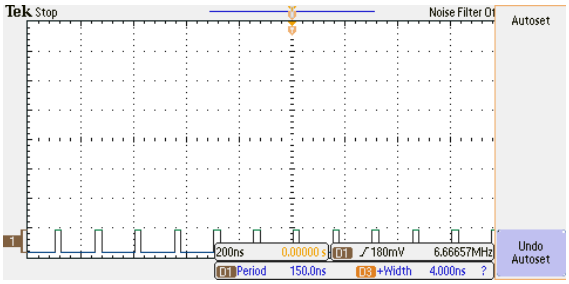
(b) Part C screen shot

Figure 1: In both images, line 2 shows the interrupt occurring with two successive interrupts

2.2 Part E



(a) Execution time of RxFifo_Get



(b) Execution time of toggling pins

The SysTick technique had the benefit of providing not only the execution time of each cycle of RxFifo_Get but also additional data that can be useful when trying to debug issues with code. This comes at the

expense of being, at best, minimally intrusive. The scope technique has the advantage of being less intrusive to the program than the SysTick techniques but its downside is not providing as much data for use in debugging code.

2.3 Part H

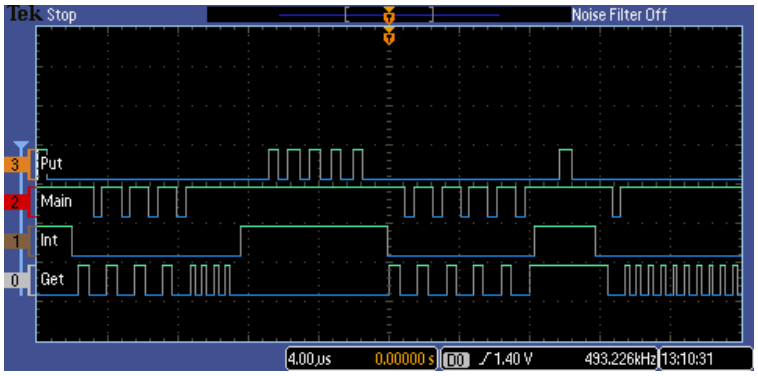


Figure 3: Screen shot of oscilloscope showing two interrupts

3 Analysis and Discussion

1. We did not measure the same result for execution speed of RxFifo_Get for all three methods. The first method of printing out the cycle count after returning from RxFifo_Get was the least intrusive but only stored one value from FIFO at a time and wrote over it every execution. Using printf inside the RxFifo_Get function was the most intrusive but had the benefit of providing not only the execution time but also the data received from the FIFO and the current FIFO pointer for ever execution cycle. The memory dump provided only the the first 10 data values recieved from the FIFO but was less intrusive than inserting the printf function into RxFifo_Get leading to a value generally between the values measured by the original FIFO function and the FIFO function with printf inside.

2. If you expected the execution speed to vary a lot, you should use the data dump technique. This technique is less intrusive than printing data to a screen and the amount of time needed to dump data each execution cycle should remain almost constant. This technique also allows for storing a large number of data entries for review later to allow averaging over hundreds of samples for average execution time as well as finding the most likely minimum and maximum execution times.

3. If the expected execution speed is very large, using printf output is appropriate provided small strings were used for output. Although printf can be very intrusive in faster programs, the long execution period means that outputing a few characters won't significantly slow down the program, as discussed in class.

4. Minimally intrusive debugging techniques are defined to have a negligible effect on the system being debugged.

5. The two necessary components collected during a "profile" are the timing characteristics and the execution patterns of a program.

6. The critical sections in the bad FIFO were at the increment and decrement since they happen after the value being changed is called from memory but before it is being stored. The way to fix this problem on ARM would be to include the post-increment and post-decrement inside the functions using these values as parameters. This is clear after viewing both the standard FIFO and bad FIFO code side by side.