EE345L – Lab 2: Performance Debugging
Antonius Keddis and Stephen To
02/14/07

1.0 OBJECTIVE
The lab aimed to familiarize students with dynamic and real-time performance debugging techniques with various degrees of intrusiveness. Oscilloscopes, logic analyzers, and software dumps were used to observe data.

Profiling also presented the detection and visualization of program activity. The program we profiled implemented a FIFO queue with interrupts enabled.

2.0 ANALYSIS AND DISCUSSION
2.1 Theoretical Data
By observing the cycle count in the listing file, we calculated the total time to call and execute RxFifo_Get with Equation 1.

\[
\text{total time} = \frac{50 \text{cycles}}{1} \times \frac{1 \text{second}}{24 	imes 10^6 \text{cycles}} = 2.08 \mu s \quad \text{(Equation 1)}
\]

2.2 Experimental Data
2.2.1 Observing the Debugging Profile
PT0 \(\rightarrow\) Channel 1: Interrupt
rising edge is start of interrupt
falling edge is end of interrupt

PT1 \(\rightarrow\) Channel 2: Foreground
rising edge is foreground processing
falling edge is foreground waiting

2.2.2 Instrumentation Measuring with Independent Counter, TCNT
We observed the cycle counts for three different versions of RxFifo_Get: without debugging (A), with debugging print (B), and with debugging dump (C). The hyperterminal output was collected into Table 1. The results for Version A confirm our theoretical prediction and indicate minimal intrusiveness (that the profiling software minimally affects the execution speed).

<table>
<thead>
<tr>
<th>Version</th>
<th>Cycles</th>
<th>Execution Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>50</td>
<td>2.08E-06</td>
</tr>
<tr>
<td>B</td>
<td>16584</td>
<td>6.91E-04</td>
</tr>
<tr>
<td>C</td>
<td>84</td>
<td>3.50E-06</td>
</tr>
</tbody>
</table>

Table 1: RxFifo_Get execution times
2.2.3 **Instrumentation Output Port**
Instructions were placed before and after the RxFifo_Get function to “turn on” and “turn off” PT0. Figure 2 depicts the output of PT0. However, the measurement instruction themselves are intrusive. Notice the execution time is slightly longer than the theoretical prediction.

![Figure 2: PT0 output](image)

2.2.4 **Profiling with a Software Dump**
Two arrays were added as a software dump: one to record the time of an activity (timeBuf[]), one to record the location of activity (placeBuf[]). Refer to Appendix A and B for the software dump output.

![Figure 3: Data flow graph of the system](image)

2.2.5 **Threading Profile using Hardware**
We observed the currently running function by associating a function with a single output port (ie PT3 for RxFifo_Put). Figure 4 illustrates the oscilloscope output.

![Figure 4: Hardware Thread Profile](image)
2.3 Conclusions

Therefore, if we expected the execution speed to vary (ie from 5 to 20ms), we would use a software dump to determine execution speed. We would be able to calculate minimum, maximum, and average speed by observing the execution pattern.

For large execution speeds (ie 20s), we would use hardware profiling. During 20s, the program would have executed 20*24*10^6 cycles, and TCNT would be unable to capture this quantity. Using hardware, the profiling instruction would be nearly negligible compared to the program itself: microseconds compared to seconds.
### Appendix A: `timeBuf[]`

```
```
```