Project Progress report:

Feb - April 2013

# Previous work

My most recent Project Proposal (Jan 2013) outlined the following work that had been carried out on the project:

* Re-worked the CAN analysis script above to simulate the filtering system in my most recent proposal, utilising multiple acceptance filters. As the single-ID filter could not cope well with the varying sequence on the CAN bus, it was hoped that allowing a block of different ID’s at any one time, and replacing those ID’s with new ones as they appeared on the bus, would have a higher success rate.
* Presented results from preliminary tests, highlighting unexpected behaviour.

# Work this since previous report

Since the previous report, I have carried out further tests, changing the number of ID’s in the sequence as well as the filter. These tests showed similar hit/miss rate patterns to the previous tests, even when lower frequency ID’s were eliminated and all logged ID’s were arriving with the same cycle time.

Referring back to the original tests I did, however, find that eliminating the less sequential ID’s from the sequence flattened out the upper end of the curve.

I have, also started to read around the subject of caching behaviour. From my reading, it seems that there are some parallels to be drawn between this behaviour and what I’m trying to achieve, although the replacement rules are different; Caches tend to keep the most used addresses, and replace the least-used ones. With my system, I need to replace the addresses as they are accessed, in order to leave the filter open to other addresses in the sequence. I did see a graph in one book that was a very similar curve to the one I’m seeing, however it didn’t go into too much detail about what caused it – simply that sometimes having a larger cache can increase the miss rate.

# Short-term plan

* Further reading around caching and related behaviour.
* Modify the test C code to associate counters and timeouts with various elements of the filtering mechanism. Investigate the effects these have on the hit/miss behaviour.
* Begin to write embedded software – choose hardware platform (probably TI C2000 as I am gaining experience on this platform with other work at Smith.

# For Discussion

* Attending the B1 module on FPGA’s has suggested that some of my project could be implemented in hardware
  + I would like some advice on whether this is possible / feasible in the timeframe I have, and also would need to investigate where the hardware / software divide would need to be.