Progress Report

Solved Issues:

Ravi:

1. Downloaded the clean OpenImpact sources and Fault Tolerant module.

2. Got OpenImpact to build on the x86
   Problems Encountered:
   a. Perl and Autoconf version conflict was solved by installing autoconf.
   b. Got Edgcpfe and inserted it into the source code.

3. Got OpenImpact clean sources to build on the Itanium machine. Compiled and executed a test "Hello World" program.

Muhammad:

4. Conducted rudimentary timing analysis on two possible methods to reduce instruction duplication for soft error detection in C.

   a. For both schemes used intels time stamp register to measure the execution times for the test runs

   b. Initially tried the improved optimization technique without turning off the compiler optimizations. The performance achieved was almost the same as that of EDDI.

   c. Turned the compiler optimizations off and retested the performance. There was a significant improvement in our schemes execution times.

   d. Manual inspected the generated assembly source code to confirm that the compiler had not removed duplicated code unnecessarily.

   e. Results showed that there was a 39% reduction in the amount of duplication overhead compared to EDDI for the "switch" construct and a 35% reduction for the "for" construct

Open Issues:

1. Get the Fault tolerant SWIFT version of OpenImpact to build.

2. Incorporating hardware abstractions in the timing framework.

Next Step:

1. Understand the OpenImpact code base and the SWIFT fault tolerance modules (Nov 9).

2. Understand the (duplicate) instruction generation for switch cases and elementary for loops (Nov 13).

3. Convert the SEU injecting framework from socket based to ptrace based (Nov 7).

4. Incorporating hardware abstractions in the timing framework (Nov 13).

5. Search for more constructs to reduce instruction duplication for soft error detection.