CSC 714 Project Report 1

Nachiket S Deshpande
nsdeshpa@ncsu.edu

"Study of memory leakage on an IBM PowerPC 405LP Embedded Processor and ways to reduce energy consumption, by combining sleep and low-power modes."

Project URL: http://www4.ncsu.edu/~nsdeshpa/project.html

Solved Issues

- Studied the leakage-aware DVS fundamentals from papers and got to know the exact scope of the project.
- Studied the fundamentals of the system from seniors – got background knowledge of the IBM PowerPC 405LP, the Comedi data acquisition system, and the Beech program for monitoring the voltage and current values of the processor.
- Finished preliminary testing of different processor sleep modes.
- Measured voltage and current levels in each mode during sleep, wakeup and after wakeup conditions, and the approx time taken to power back up.
- Produced a comparative chart to compare and contrast between the modes and find out the useful ones.
- Started writing code to automate the manual sleep measurement process – the aim was to initially read the system clock, add 1 minute to it and then echo this value into pm_alarm, so that the processor will wake up at this time.
- Started looking for methods to write into the NFS mounted filespace, so that written code could be ported over to the board.
- Found a simple way to port files across to the board – the file to be sent was just copied to the following path: /opt/hardhat/previewkit/ppc/405/target
- Finished writing code that was able to write a given hh:mm:ss string into pm_alarm – the user was able to enter a string, which got echoed into pm_alarm, thereby arming the timer. The status of pm_alarm also could be read back.
- Started exploring different functions that could read the current system time, like gettimeofday() and localtime().

- Used localtime() to read the current system time into a structure, add 1 minute to the corresponding field and then writing the string into pm_alarm.
- Started using functions like setitimer() to setup a timer that will fire periodically, at which the system status can be checked, and signal handling functions like signal() and sigaction() to setup a handler for the interrupts generated.
- Was able to successfully measure the time delay between successive interrupts in terms of processor clock cycles using the clock() function. Tested this for granularity in the order of seconds. Refined the program by replacing clock() with gettimeofday(), as it can measure delays of the order of microseconds.

Open Issues

- Refining the measurement granularity to 10 ms (currently, there is a problem that the system can report microsecond delays accurately only if the interval between successive interrupts is 30ms or greater. (Could be resolved by using sigaction() instead of signal(), to be tested.)
- Once a granularity of 10 ms is reached, reading the actual system signals (interrupts from the APM) so that the sleep overhead for the suspend mode can be accurately determined.
- Putting this functionality into a system call and seeing which approach would be better (programming the /proc interface VS doing it with a system call)
- Integration with DVS and further experimentation

Next Steps

- Getting a 10ms granularity (by Nov 4)
- Porting the code to the board, and actual sleep mode testing (reading the board signals) and finding the overhead (by Nov 11)
- System call implementation (by Nov 18)
- DVS integration (by Nov 25)