CSC 714
Report 2
Implementing Frequency Scaling on the iPAQ H3975

Group 2
Ramrakhyani, Prakash
Wah, Mark
Welch, Ben

Accomplishments

Week (11/04/02 – 11/10/02)

- Researched methods to change the frequency and resolved to go with assembly language programming option. Used assembly language to write to the CCLKCFG register on Co-Processor 14 of the Intel PXA250
- Ability to write to the memory-mapped register (Core Clock Configuration Register)
- Intel PXA250's manual showed that we need both changes mentioned above to achieve frequency scaling.

Week (11/11/02 – 11/17/02)

- Investigated third party benchmark software (VOBenchmark) to confirm frequency scaling.
- Researched threads functionality and Sleep() functionality in Windows CE. Results are not favorable since we discovered that the OEM (HP/Compaq) might have changed the functionality.
- Studied the algorithm (Energy-Conserving Feedback Scheduling) that we are using for this project.

Contributions (11/04/02 – 11/17/02)

Mark: Research on assembly language option, the appropriate assembler and technique in eVC++.

Prakash: Research on register details from the Intel PXA250 manual, calculating the supported frequencies.

Ben: Research on Threads, Sleep(), Posix threads and alternatives for the scheduler.

All: Programmed the registers successfully using a combination of ARM assembly language and memory-mapped register (using VirutalAlloc an VirtualCopy).

All: Studied on the algorithm in detail. Found follow-up on the paper and writing pseudo code for the implementation.

Setbacks

One of the major issues we've encountered for our scheduling program is how to put the scheduler to sleep for a specified amount of time. The most apparent option was to use Posix threads. However, Windows CE is not Posix compliant [3]. Another option was to use the Sleep() function provided in the Windows CE API. The Sleep() method is defined to allow a thread to sleep for a certain amount of time, specified in milliseconds. However this function does not seem to operate properly under the Pocket PC 2002 SDK. This could be due to the OEM redefining the functionality.

We are considering the use of a third party thread library. Red Hat has a p-thread library for use in the windows 32 environment [4]. Another alternative would be to have the scheduler perform computations during each "jobs" time-slice, to simulate a load on the processor. This would be a worst-case solution to the problem of putting the scheduler to sleep since we would never have to make the scheduler "sleep".

Currently we have decided to place more emphasis on understanding an implementing the DVS algorithm. We plan to further research the threading issue when we are nearing the completion of the DVS scheduler.

Detail

Investigated Power Management APIs but did not find proper APIs to set frequency levels. Found API to get the battery status. Opted to use assembly language support for frequency scaling.

Researched assembly language usage and discovered that inline support is weak and there is some pessimism in finding the right assembler for the Intel PXA250. More research shows that the ARM assembler that comes with eVC++ 3.0 is compatible with the Intel PXA250.

Using assembly language requires some understanding in the ARM assembly code. The easiest way to do so is to generate an assembly output from a C file. This requires some Project settings in eVC++. The code generated is placed in the output portion (ARMDbg or ARMRel) of the project workspace and we need to move it to the source code space. We also need to specify the build instructions for this .asm file since we are using the ARM assembler (armasm.exe). This is done successfully.

The PXA250 has 4 frequency settings viz, 100, 200, 300 and 400 Mhz.

To change the frequency on the PXA250, we need to program two specific registers. These are:

CCCR or the Core Clock Configuration Register: This is a memory mapped register, at address 0x41300000. This register specifies 3 frequency multipliers that are used to determine the frequency of the processor. Detailed information on specific fields of this register can be obtained from chapter 3 in the Developer's manual for PXA250 [5].

CCLKCFG register of the Power Manager: This register 6 of Coprocessor 14(Power Manager). This register indicates if the processor is operating in Turbo Mode or Run

Mode. The FCS bit in this register when set, causes the processor to enter the Frequency Change Sequence.

The values for these registers for the four frequency settings are detailed in the following table:

	100 MHz	200 MHz	300 MHz	400 MHz
CCCR	289	321	449	577
CCLKCFG	2	2	3	3

The table below summarizes the typical voltage settings for the four frequency settings. (Source: Electrical, Mechanical, and Thermal Specification Datasheet for PXA250 [6])

Frequency(MHz)	Voltage (V)
100	0.85
200	1.00
300	1.10
400	1.30

Next Steps

Complete implementation on feedback DVS-EDF algorithm in EVC++

Breakdown of task:

Prakash: Preemption handling, slack generation

Mark: Task completion, scaling factor, idle task

Ben: Initialization, Task Activation, Scheduler design in general

All: Integration of separate portions, experiments and testing.

References

[1] A. Dudani, F. Mueller, Y. Zhu "Energy-Conserving Feedback EDF Scheduling for Embedded Systems with Real-Time Constraints."

[2] F. Mueller, Y. Zhu "Preemption Handling and Scalability of Feedback DVS-EDF."

[3] (Almost) No POSIX OS Is An Island (http://www.embedded.com/story/OEG20020111S0071)

[4] Open Source POSIX Threads for Win32 (http://sources.redhat.com/pthreads-win32/)

[5] Intel PXA 250 Application Processors Developer Manual (ftp://download.intel.com/design/pca/applicationsprocessors/manuals/278522-001.pdf)

[6] Electrical, Mechanical, and Thermal Specification Datasheet for PXA250 (ftp://download.intel.com/design/pca/applicationsprocessors/manuals/278524-001.pdf)