Power-Aware DVFS on IBM PowerPC 405LP: Front Bus Scaling

Intermediate Project Report

Mohamed Nishar Kamaruddin (mkamaru@ncsu.edu)
Santhosh Selvaraj (sselvar@ncsu.edu)
Instructor: Dr. Frank Mueller

CSC714
REAL-TIME COMPUTER SYSTEMS
NORTH CAROLINA STATE UNIVERSITY

Completed work:

1. We got ourselves acquainted with the 405LP board and the development environment.
2. We learnt to execute the *example* program, which uses the EDF-DVS based thread library to schedule threads, in the target and collect processor core voltage readings from the data acquisition board via the *beech* program.
3. From the Dynamic Power Management documentation, we got to know about the various components of the PowerPC 405LP operating points under Linux such as Core Voltage, System Clock source, PLL Multiplier and Divider, CPU/PLB divider, etc. Some of these components are used by the existing implementation in defining various operating points. In our case, we needed to look at the component which scales the PLB frequency. The PLB frequency is derived from the component CPU/PLB divider, which is also the SDRAM frequency.
4. We had to understand the DPM module to invoke appropriate policies in *example.c* which helped us collect values for the study.
5. We learnt about the Comedi driver, its user-space library interface and its device hierarchy (devices, sub-devices and channels). We also familiarized ourselves with the acquisition terminology and the typical acquisition sequence which consists of identically repeated scans. These scans have number of parameters which mentions its start time, end time, etc.
6. With the knowledge of the Comedi driver and its interfaces, we modified *beech.c* to collect the voltage readings for the I/O subsystem in addition to the existing processor voltage calculations. However, initially we didn't exactly know to which channel the I/O subsystem raw voltage values are input. We experimented with first eight channels. After analyzing the voltage values from different channels, channel 3 had the right voltage value of ~3.3V. I/O subsystem has a persistent voltage source of 3.3V. So we modified the *beech.c* to process the results from channel 3 and channel 4.
7. To calculate the energy consumed by the memory, *avg.c* was modified to extend the processor energy calculations to the memory subsystem energy calculations as well.

Our initial approach for measuring the energy benefits that can be attained by the scaling the frequency of front side bus, is to collect the voltage/current values of the memory subsystem under different workloads (with or without memory intensive operations) with different PLB frequencies. We need to mention that voltage can't be scaled for the I/O subsystem since it has a persistent voltage source of 3.3V. For this purpose, we introduced the following two operating points in the file dpm405lp.c:

```
{ "200/50/24",     {1500, 1, 24,  4, 4, 32, 32, 4, 4,  4, 0, 0, -1, -1} }
{ "200/25/24",     {1500, 1, 24,  4, 8, 32, 32, 4, 4,  4, 0, 0, -1, -1} }
```

The red values denote the CPU/PLB divider. Here we reduce the PLB frequency to 50 MHz and 25 MHz.

For the workload with and without memory intensive operations, we developed the following applications:

- A memory intensive application which reads a huge file and also updates and prints a volatile variable
- A very less memory referencing application implemented by an infinite loop.

Results:

We got the following results after executing the applications under different frequencies. These are the first ten readings for each experiment.

A: Channel 1 reading
B: Voltage for CPU
C: Channel 2 reading
D: Converted voltage reading from channel 2
E: Current for CPU
F: Channel 3 reading
G: Voltage for I/O subsystem
H: Channel 4 reading
I: Converted voltage reading from channel 4
J: Current for I/O subsystem

File Reading application @ PLB 100 MHz:

Time	A	B	C	D	E	F	G	H	I	J
0.000000	2365	1.55067	2079	0.15385	0.05554	2720	3.28449	2086	0.18803	0.06788
0.008260	2368	1.56532	2088	0.19780	0.07141	2720	3.28449	2086	0.18803	0.06788
0.016520	2364	1.54579	2114	0.32479	0.11725	2719	3.27961	2086	0.18803	0.06788
0.024780	2362	1.53602	2083	0.17338	0.06259	2719	3.27961	2086	0.18803	0.06788
0.033040	2367	1.56044	2088	0.19780	0.07141	2719	3.27961	2089	0.20269	0.07317
0.041300	2365	1.55067	2137	0.43712	0.15780	2719	3.27961	2087	0.19292	0.06965
0.049560	2366	1.55556	2077	0.14408	0.05201	2719	3.27961	2086	0.18803	0.06788
0.057820	2365	1.55067	2089	0.20269	0.07317	2719	3.27961	2086	0.18803	0.06788
0.066080	2365	1.55067	2161	0.55433	0.20012	2720	3.28449	2088	0.19780	0.07141
0.074340	2365	1.55067	2092	0.21734	0.07846	2719	3.27961	2087	0.19292	0.06965

File Reading application @ PLB 50 MHz:

Time	A	B	C	D	E	F	G	H	I	J
0.000000	2371	1.57998	2070	0.10989	0.03967	2720	3.28449	2091	0.21245	0.07670
0.008196	2371	1.57998	2077	0.14408	0.05201	2720	3.28449	2079	0.15385	0.05554
0.016392	2372	1.58486	2072	0.11966	0.04320	2720	3.28449	2080	0.15873	0.05730
0.024588	2371	1.57998	2089	0.20269	0.07317	2720	3.28449	2079	0.15385	0.05554
0.032784	2371	1.57998	2088	0.19780	0.07141	2720	3.28449	2079	0.15385	0.05554
0.040980	2371	1.57998	2068	0.10012	0.03615	2720	3.28449	2092	0.21734	0.07846
0.049176	2371	1.57998	2069	0.10501	0.03791	2719	3.27961	2079	0.15385	0.05554
0.057372	2371	1.57998	2071	0.11477	0.04143	2720	3.28449	2080	0.15873	0.05730
0.065568	2372	1.58486	2085	0.18315	0.06612	2720	3.28449	2081	0.16361	0.05907
0.073764	2373	1.58974	2070	0.10989	0.03967	2720	3.28449	2079	0.15385	0.05554

Infinite Loop @ PLB 100 MHz:

Time	A	B	C	D	E	F	G	H	I	J
0.000000	2367	1.56044	2175	0.62271	0.22481	2720	3.28449	2087	0.19292	0.06965
0.008194	2365	1.55067	2239	0.93529	0.33765	2719	3.27961	2087	0.19292	0.06965
0.016388	2366	1.55556	2160	0.54945	0.19836	2720	3.28449	2086	0.18803	0.06788

```
0.024582 2365 1.55067 2179 0.64225 0.23186 2720 3.28449 2088 0.19780 0.07141
0.032776 2365 1.55067 2246 0.96947 0.34999 2720 3.28449 2086 0.18803 0.06788
0.040970 2366 1.55556 2161 0.55433 0.20012 2719 3.27961 2087 0.19292 0.06965
0.049164 2367 1.56044 2181 0.65201 0.23538 2720 3.28449 2086 0.18803 0.06788
0.057358 2365 1.55067 2225 0.86691 0.31296 2720 3.28449 2086 0.18803 0.06788
0.065552 2366 1.55556 2163 0.56410 0.20365 2719 3.27961 2087 0.19292 0.06965
0.073746 2365 1.55067 2177 0.63248 0.22833 2720 3.28449 2087 0.19292 0.06965
```

Infinite Loop @ PLB 50 MHz:

```
Time       A    B       C    D       E       F    G       H    I       J
0.000000 2369 1.57021 2137 0.43712 0.15780 2720 3.28449 2080 0.15873 0.05730
0.008194 2370 1.57509 2143 0.46642 0.16838 2721 3.28938 2079 0.15385 0.05554
0.016388 2371 1.57998 2151 0.50549 0.18249 2720 3.28449 2080 0.15873 0.05730
0.024582 2372 1.58486 2162 0.55922 0.20188 2720 3.28449 2079 0.15385 0.05554
0.032776 2373 1.58974 2190 0.69597 0.25125 2720 3.28449 2078 0.14896 0.05378
0.040970 2368 1.56532 2246 0.96947 0.34999 2720 3.28449 2079 0.15385 0.05554
0.049164 2369 1.57021 2138 0.44200 0.15957 2720 3.28449 2079 0.15385 0.05554
0.057358 2370 1.57509 2144 0.47131 0.17015 2722 3.29426 2079 0.15385 0.05554
0.065552 2371 1.57998 2152 0.51038 0.18425 2720 3.28449 2080 0.15873 0.05730
0.073746 2374 1.59463 2165 0.57387 0.20717 2720 3.28449 2080 0.15873 0.05730
```

Following are energy measurements for memory subsystem from the above data:

Infinite Loop @ PLB 50 MHz:
Power 149.464111 mWatt over 8.177612 secs = Energy **0.339517 mWatt-hours**

Infinite Loop @ PLB 100 MHz:
Power 226.028000 mWatt over 8.177612 secs = Energy 0.513436 mWatt-hours

File Reading application @ PLB 50 MHz:
Power 147.914917 mWatt over 8.179608 secs = Energy **0.336079 mWatt-hours**

File Reading application @ PLB 100 MHz:
Power 234.820145 mWatt over 8.243480 secs = Energy 0.537704 mWatt-hours

From the above results, it is observed that scaling the front bus frequency could produce significant energy savings.

Open Issues:

- We need to walkthrough the feedback control scheduling code and understand how the processor frequency assignment/scheduling are done for each subtask.
- Memory related parameters that are to be monitored by the feedback controller have to be identified, and also how it could be used to generate feedback for the memory related feedback controller has to be analyzed.
- We need to figure out how the existing feedback for processor voltage/frequency and the memory feedback mechanism would work together. This means implementing the memory feedback controller and integrating it with the existing PID feedback scheduler so that for

each invocation of a subtask an appropriate operating point is generated. This would include the processor frequency/voltage point and FSB frequency to be used.

Milestones:

Tasks	Due Date
Code Walkthrough of existing PID controller and the EDF-DVS scheduler	08th April 2009
Memory parameters for PID mechanism	12th April 2009
Integration with the existing scheduler	20th April 2009

Individual Contributions:

Tasks accomplished so far were contributed by both the team members. For next coming weeks we plan to accomplish the work in following manner:

Mohamed Nishar - Code Walkthrough of existing PID controller and EDF-DVS scheduler.
Santhosh Selvaraj - Memory parameters for PID and code walkthrough of the scheduler.

And the final integration work will be contributed by both.

References:

[1] Beech Board for NCSU – Beech Notes
[2] Dynamic Power Management for Embedded Systems *IBM and MontaVista Software Version 1.1a, November 11, 2002*
[3] Comedi Documentation: http://www.comedi.org/doc/index.html