Characterization of Future Deep Space Computing Loads

Peter M. Kogge
Univ. of Notre Dame
kogge@cse.nd.edu

Jeffrey Namkung, Nazeeh Aranki,
N. Benny Toomarian
Jet Propulsion Laboratory
{Jeffrey.Namkung, Nazeeh.I.Aranki, Nikzad.Toomarian} @jpl.nasa.gov

Kanad Ghose
State Univ. of New York at Binghamton
ghose@cs.binghamton.edu

Abstract

This paper reports on the characteristics of a future deep space computing application. This includes the testbed used to make the measurements, the application suite itself, the scenarios measured, the detailed power and performance measurements taken, and the timing characteristics of the system.

1. Introduction

As part of the MORPH project [1] on inherently low power microprocessor architectures, a collaboration of the Univ. of Notre Dame, JPL, and the State University of New York at Binghamton have developed a testbed of a future deep space mission, DEEP IMPACT [2], and instrumented it to provide insight into very realistic processing loads and timing requirements under a variety of scenarios. Our original project goal was to use this data to baseline time and power data against which our proposed new architectures [5] would be, and have been [4], evaluated. However, the wealth of data from this testbed is useful for far more than just our project, since they represent a unique view into a type of application that will become of increasing importance, namely embedded systems involving multiple different tasks running under real-time constraints.

This paper will describe the testbed facility, an application suite that when running on the testbench mirrors closely the code expected to run on the real mission, and summarize the key characteristics of load patterns derived from this testbed.

2. An Instrumented TestBench

The testbench itself is modified single board computer (SBC) of the same type and performance as is projected to fly in several future missions. Modifications to the board include chip extenders to allow access to individual chips. This testbench was then connected to a testbed that provided a simulation of the rest of the spacecraft, in real time. A methodology was then developed that allowed instrumentation connected to this testbench to perform significant and detailed power and timing measurements, while application scenarios were being run.

2.1. The Modified Single Board Computer

We chose an SBC designed by WindRiver that housed a Motorola PowerPC 750 processor, 128 MB RAM, a serial port, and two PCI ports. It runs VxWorks^{tm} as its operating system, just as the real system. One PCI port contained a Network Interface Card, while the other PCI port contained a reflective shared memory card. This board was a prime choice due to its use of a PGA (Pin Grid Array) connection to attach the processor chip to the board. This allowed us to access all the pins on the processor, and thus directly measure current between the power and ground pins on the processor. Furthermore, access to all the I/O pins allows the use of a logic analyzer to accurately obtain a timing profile of the processor's interaction with peripheral devices, e.g. bridge controllers, memory, etc.

Modifications as seen in Figure 1 were made to the SBC to allow access to circuit connections designated for power to the processor. An oscilloscope with a current probe measured the current being drawn by the CPU and memory. A power distribution module provided easy access via a connector to the board housing

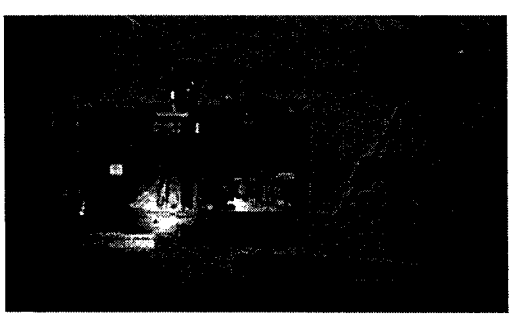


Figure 1. The single board computer

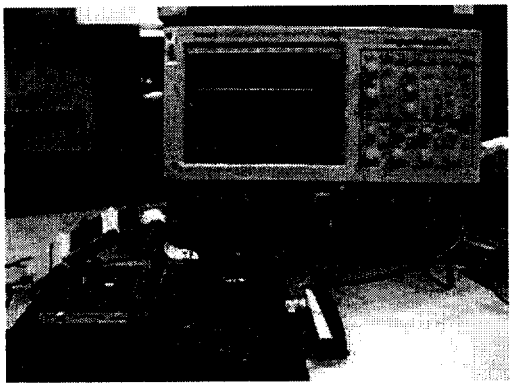


Figure 2. Current probe

the processor and memory. Calculating power from the current measurements was simply done by measuring the voltage at the CPU and memory and using the product of the two to determine the instantaneous power consumption. Figure 2 pictures an oscilloscope with the current probe attached to the SBC to measure power.

2.2. Monitoring Facilities

JPL supports an Autonomous Test Bench, known as Babybed, which had been previously been developed for mission software development and performance benchmarking, as shown in Figure 3. The system runs on two processors that share a common VME or PCI back plane. One processor runs a simulation of a virtual vehicle and the second processor runs the control software that drives the simulated vehicle, currently a three-axis stabilized, free-floating spacecraft. The simulation is a fully end to end real-time software simulation which allows actual mission fight software to communicate with virtual spacecraft devices like actuators and sensors, as well as power, telecom and science hardware. As commands are issued and executed by the virtual devices, the spacecraft dynamics are affected accordingly and the sensor models sense the results.

The autonomous test bench also includes two single-board computers communicating via shared memory. implemented by using two reflective memory cards and a fiber optical cable providing transparent communication between them. One of the SBCs is responsible for running the flight software. The other single-board

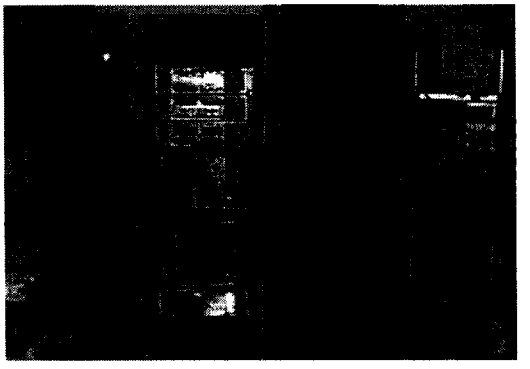


Figure 3. Babybed autonomous test bed

computer is responsible for running a space environment simulator as inputs to the flight software.

A performance profile was generated for the flight software in terms of cache misses, CPU utilization, and MIPS. These figures can be seen in Table 1..

The test bench described in the previous section was ported to the autonomous test bench by replacing the single-board computer running the flight software with the power measurements test bench. Because the power measurements test bench provided two PCI slots, a reflective memory card with a PCI interface was used to enable communication of the space environment simulator with the power measurement test bench.

2.3. Measurement Methodologies

Power profiling was performed at three levels of granularity. At the fine-grain level, power was measured at the instruction-level. The instruction set of the processor used for experiments, a Motorola PowerPC 750, was profiled comprehensively, i.e. a measurement for each instruction in the instruction-set was obtained. The basic idea for profiling was to place a single instruction into a loop, and repeat it until it reaches a steady state. At this point we could measure the power of that instruction. More details of this method is documented in [6], with a subset seen in Figure 4.

At the medium-grain level, power was measured at the event-level. Events at the micro-architecture level included data-forwarding, cache accesses, floating-point pipelining, instruction units, etc. A sequence of instructions was programmed with the understanding

Table 1. Basic processor characteristics

MIPS Rating for Selected Tests			
Test #	Test Name	MIPS (PPC750 @ 233MHz)	MIPS (PPC750 @ 133MHz)
2	pre launch idle	264	151
3	post detumble idle	260	149
5	dseu in scan mode	260	148
7	bursting dseu and ip	258	147
12	Image Compression	195	111
13	Orbit Determination	212	121

Test #	Test Name	I-Cache Miss %		D-Cache Miss %		PPC750 CPU %	PPC750 CPU IDLE %
		Measurement A	Measurement B	Measurement A	Measurement B		
1	Post launch detumble	0.013	0.014	0.024	0.026	3.4	96.6
2	Pre launch idle	0.009	no data	0.017	no data	1.2	98.8
3	Post detumble idle	0.012	no data	0.026	no data	3.3	96.7
12	Image Compression	0.039	0.044	1.14	1.188	60.2	39.8
13	Orbit Determination	0.104	no data	0.734	0.777	67.0	33.0
14	Manuever Planning	0.018	no data	0.034	0.041	4.9	95.1

Instruction	Execution units involved	Measured current (Amps)
subfze	**Both IU* units**	**1.75**
subf	**Both IU units**	**1.89**
subfc	**Both IU units**	**1.90**
subfe	**Both IU units**	**1.75**
subfic	**Both IU units**	**1.90**
subfme	**Both IU units**	**1.77**
sub	**Both IU units**	**1.88**
addic	**Both IU units**	**1.90**
addis	**Both IU units**	**1.89**
divw	**IU2**	**1.65**
mullw	**IU2**	**1.78**

Figure 4. Sample instruction power measurements

that certain events would take place. For example, sequencing an add instruction with dependencies on each previous add instruction would invoke the data-forwarding mechanism to avoid pipeline stalls. Further details on the methodology and analysis of these results can also be found in [6].

Lastly, coarse-grain measurements were performed at the operating system level by analyzing multiple running tasks and associated power profiles with the tasks. This paper focuses on these results.

To identify tasks that were executing during a test we invoked a WindRiver tool called WindView. This allowed us to generate a timeline of software activity. The start and stop time of each task when it took control of the processor was shown graphically and could also be stored in a database file. Combined with the power measurements, the relationship between the power profile of software and the tasks running were trivial.

2.4. Power Measurements

Power profiles were taken from a subset of the entire software suite. The modules we chose to profile included image compression, orbit determination, maneuver planning, and several operating modes. The first three mentioned are of importance because they show dynamic profiles that reflect their computational complexity. Figure 5 shows power profiles for these.

2.5. Timing Measurements

The timing profiles produced by WindView became very useful in determining which task was consuming power at different times. Graphically, the timing profiles for each of the three modules mentioned above can be seen in Figure 6. As can be seen, there is a close correlation between power profiles and timing profiles.

3. The Application Suite

The application suite used in Morph is derived from the Deep Space 1 (DS1) mission, and modified to reflect potential use on the Deep Impact mission. Deep Impact involves a comet orbiter that includes an instrumented spike-like probe. After release from the orbiter, the probe will impact the comet. Visual analysis of the dispersal from the orbiter should yield significant insight into the interior nature of the comet.

The DS1 flight software is comprised of some 60 tasks that are initiated and initialized at startup, and run forever. The tasks vary in priority. Some tasks wake up and execute in response to interrupts, and others run when the scheduler activates them. The software operates loosely on a 1 second cycle. Nominally all tasks run at least once every second. Some tasks run more frequently (4Hz and 8Hz rates). The general procedure is for a task to wake up, run and go back to sleep (pending state).

When a task wakes up, it performs some basic duties and/or check its inter-process communications (IPC) queue(s). It will attempt to complete its duties and/or process all the data in its queue(s) before it is suspended

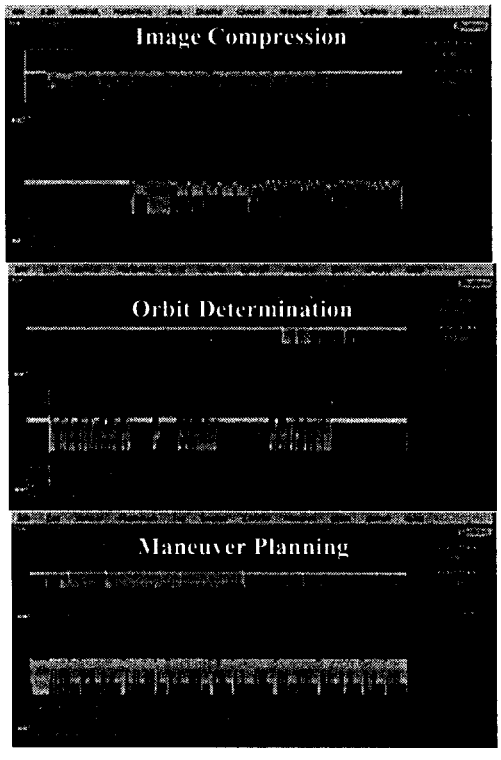


Figure 5. Sample power profiles

by the operating system. Any unfinished business is picked up the next time the task awakes. Naturally, tasks with high priority have more opportunity to execute each second than lower priority tasks. As a result, busy high priority tasks may prevent lower priority tasks from ever waking up within some second.

When there is not much activity and all tasks finish early within the 1 second interval, the tIdleTask uses the remaining time. As system activity increases, the amount of time the tIdleTask runs becomes less and less. Further details can be found in [3].

Due to ITAR restrictions, the DS1 flight software is not publicly available to the research community. As an alternative, a software module from the suite was modified so that it could be made publicly available. The module chosen, MICAS, is an image compression algorithm. The computational complexity coupled with the need for generous amounts of memory I/O made this a good choice for power profiling and optimization.

In addition to providing the MICAS source, JPL performed both power and performance characterization of the software without any optimizations. A general power profile of MICAS can be seen in Figure 7(a).

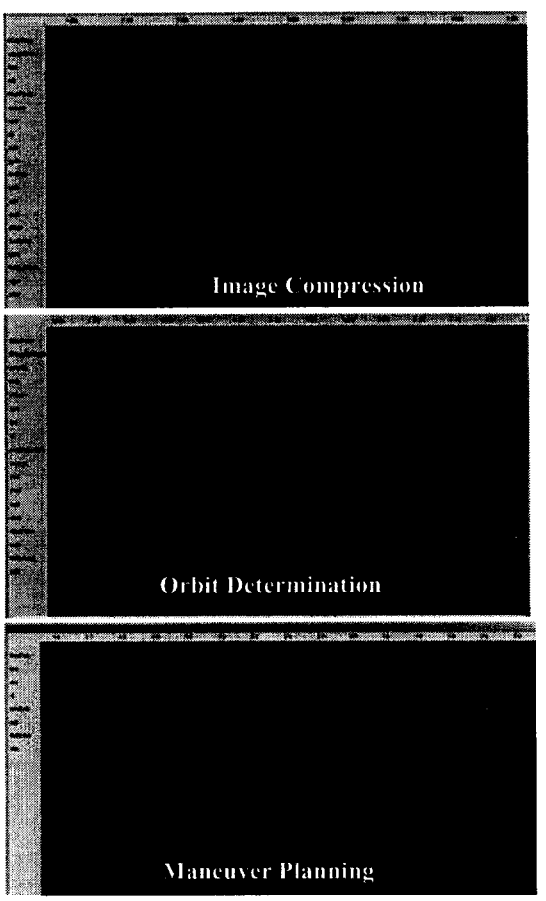


Figure 6. Timing Profiles

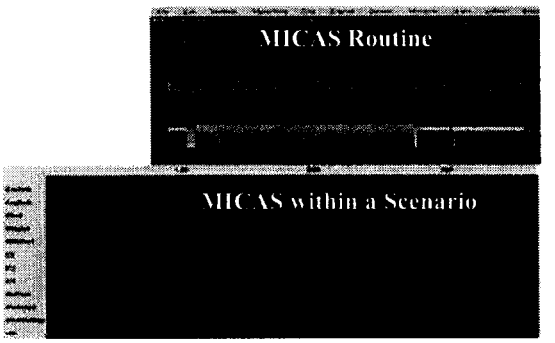


Figure 7. MICAS power details

WindView provided a timing profile of the tasks running. The results from WindView can be seen in Figure 7(b). As can be seen there is a strong correlation between the power and timing profiles.

Each task trace shown graphically was exported to an Excel worksheet displaying the timestamps of each task's start and stop time, as demonstrated in Figure 8.

3.1. Simulated Flight Scenarios

Seven different flight scenarios were profiled for power consumption and time traces. The first scenario (prelaunch) simulated the flight computer while in an idle state prior to vehicle launch. The second scenario was to profile the flight after a vehicle launch. The third and fourth scenarios (DSEU_Scan and DSEU_Burst) involved placing the flight computer into different modes corresponding to their DSEU operation. The remaining scenarios involved image compression, orbit determination, and maneuver planning.

The Motorola PowerPC 750 can be configured to enable access to an L2 cache or to bypass the L2 and go directly to memory. In addition, a dynamic power management (DPM) feature, based on clock gating, can be enabled or disabled. Most of the seven scenarios mentioned above were run in three modes: L2 enabled with DPM on, L2 enabled with DPM off, and L2 accesses disabled with DPM off.

0.0128892	tLogTask
0.0128892	tPortmapd
0.0128892	tRlogind
0.0128892	tShell
0.0128892	tWvRBuffMgr
0.0128938	tNetTask
0.0128938	tWdbTask
0.012913	idle
0.012913	tNetTask
0.0166696	idle
0.0185665	idle
0.0185665	tNetTask
0.0186709	idle
0.0186709	tNetTask
0.0333347	idle

Figure 8. Sample Task Trace

4. Trace Processing

The traces as gathered from the testbench formed the basis for the load analysis of the Deep Impact code done for the Morph project. This analysis was done in three steps - a preprocessing step, an interval identification and analysis step, and a power modeling step. The last is discussed more fully in [4].

4.1. Preprocessing

The trace logs consisted of literally thousands of entries, each giving start time and task number. These were first inspected to eliminate recording errors, and then preprocessed to remove effects of the task tracing hooks by replacing all routines known to be part of the testbench system, and not to be found in the expected flight software, by idle. Multiple neighboring entries to the same task (primarily idle) were then collapsed into one, and each remaining trace entry augmented from outside information with the task priority (in the current state of the flight software, priority assignments are static, and range from a highest level of "0", to a low idle level of "255"). A variety of global statistics were then computed for each trace, such as total time, tasks started per second, total busy and idle time, etc., along with statistics on each task, such as the number of times it was run, the minimum, maximum, and average execution times, and the standard deviation in this time.

4.2. Interval Analysis

From each preprocessed trace, sequences of processing "intervals" were identified, where one interval starts at the end of an idle period, after which a string of one or more real tasks were executed, and terminated with some other idle period. For each such interval, the overall busy and idle periods were computed. Histograms were then developed from this data.

A key statistic taken from each such interval for use in the power modeling analysis was the ratio of the total interval time to the processing time. This ratio indicates by how much a factor the CPU's performance could be slowed down during the processing period to eliminate the idle period, and run in a lower power mode. Again, this is discussed more fully in [4].

5. Application Characterization

Table 2 summarizes the overall characteristics of each of the 15 scenarios investigated, including:

- the scenario name (with "oo", "of", & "ff" corresponding to L2 and DPM on and off),
- the overall power configuration (whether or not the L2 cache and/or the DPM facility was active),
- the total time of the simulated mission phase,
- the total number of original traces from the trace log, expressed as an average per second,
- the equivalent number of scheduling "events" per second after eliminating the non-operational tasks and combining idle tasks,
- the average number of intervals per second that would be observed in the mission phases,
- the percent of the total scenario time that the microprocessor was idle,
- the percent of the time that the processor was busy executing code (100% minus the prior number).

Table 2. Overview Scenario Characteristics

Scenario	Mission Phase	L2	DPM	Total Time	Traces/sec	Events/sec	Intervals/sec	% Idle	% Busy
PLoo	Prelaunch	On	On	7.635	1027.652	284.353	15.455	96.907	3.093
PLof	Prelaunch	On	Off	8.398	1219.915	270.973	18.105	84.737	15.263
PLff	Prelaunch	Off	Off	9.991	1834.617	343.298	25.122	94.442	5.558
DSoo	DSEU Scan	On	On	9.610	1284.293	298.753	16.441	96.049	3.951
DSof	DSEU Scan	On	Off	6.567	1207.951	308.802	17.054	95.158	4.842
DSff	DSEU Scan	Off	Off	7.187	1351.378	302.640	18.785	88.594	11.406
MPoo	Maneuver Planning	On	On	3.730	1365.427	665.689	82.574	62.510	37.490
MPof	Maneuver Planning	On	Off	3.656	1892.025	689.784	129.908	59.261	40.739
MPff	Maneuver Planning	Off	Off	4.707	1584.870	567.876	96.452	48.067	51.933
ODoo	Orbit Determination	On	On	2.657	1136.030	523.221	47.429	62.206	37.794
ODof	Orbit Determination	On	Off	2.111	1181.248	567.308	75.782	66.443	33.557
ODff	Orbit Determination	Off	Off	3.148	971.855	415.238	46.385	54.310	45.690
IPoo	Image Processing	On	On	9.984	1673.411	424.462	9.415	40.025	59.975
IPof	Image Processing	On	Off	9.984	2300.885	541.644	10.416	60.125	39.875
IPff	Image Processing	Off	Off	9.985	2331.782	557.657	10.917	42.400	57.600
DBoo	DSEU Burst	On	On	7.156	1349.682	293.610	1.118	89.945	10.055
DBof	DSEU Burst	On	Off	6.616	1680.678	310.313	18.193	84.119	15.881
MIoo	MICAS	On	On	3.288	1313.537	355.632	43.795	37.581	62.419

Figure 9 plots three of the more important of these characteristics grouped by mission phase, and ordered in rough order of load on the CPU (% Busy).

A key observation is the broad range of busy times, from about 3% to 60%, for a 20 to 1 variation. This verified a key premise for the Morph program as a whole that processing loads were highly variable during missions, and that knowing that fact, and having control over power/performance of the CPU could lead to dramatic overall mission power and energy savings.

One interesting observation that the data doesn't always play out as one would expect is variations as the settings for L2 and DPM are changed. For example, one would expect the most busy configuration to be the ones where L2 if off, but this is not true for Prelaunch. A similar seeming inconsistency can be seen in the "Intervals per second," where variations are strongest for the middle phases when the DPM is turned off.

A more general observation is that for the most part, the number of tasks scheduled per second changes dramatically versus mission phase (as expected), but does not change much versus configuration. This makes sense, since in each phase one would expect the same mix of tasks to be executed at the same rates, regardless of the loading on the microprocessor.

	PLoo	PLof	PLff	DSoo	DSof	DSff	DBoo	DBof	MPoo	MPof	MPff	ODoo	ODof	ODff	IPoo	IPof	IPff	MIoo	
% Busy	3.09	15.26	5.56	3.95	4.84	11.41	10.06	15.88	37.49	40.74	51.93	37.79	33.56	45.69	59.98	39.87	57.60	62.42	
Ratio	927.6	797.6	718.0	786.2	818.8	791.8	657.0	614.5	489.2	474.9	1.0	1.0	1.0	1.0	478.4	613.2	337.4		Ratio
tidleTask	97%	85%	95%	96%	95%	89%	90%	84%	63%	59%	48%	62%	66%	54%	40%	60%	42%	38%	2.6
strokeWDT																			0.0
health																			0.0
bus	0.4%	0.4%	0.4%	0.5%	0.5%	0.6%	0.7%	0.7%	0.4%	0.4%	0.5%	0.4%	0.4%	0.5%	0.5%	0.4%	0.5%		1.7
dfi	0.1%	0.1%	0.2%	0.1%	0.1%	0.2%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.2%	0.2%	0.2%	0.2%		1.7
eep_sec_hi																			0.0
eep_file																			0.0
eep_file_small																			0.0
eep_sec_lo																			0.0
micas															9.3%	6.3%	10%		1.6
acs	0.3%	0.3%	0.4%	0.3%	0.3%	0.4%	0.3%	0.4%	0.4%	0.4%	0.4%	0.4%	0.3%	0.5%	0.4%	0.4%	0.4%		1.5
fpr																			0.0
t1																			0.0
t2	2%	13%	4%	2%	3%	9%	7%	13%	28%	33%	44%	33%	27%	40%	47%	31%	44%		23.4

5.1. Task Scheduling Mix Characteristics

Table 3 summarizes the dynamic scheduling characteristics of the different scenarios. Scenario names are listed across the top. The second row labeled "% Busy" is just the average percent of time that the CPU is busy during that scenario. "Ratio" is the ratio over that scenario between the highest task scheduling rate and the lowest non-zero scheduling rate. Next, "% of all Tasks" is the percent of all the listed tasks that are in fact invoked at one time or another during the scenario.

There is one row below these for each task that is part of the operational mix (not all rows shown here). The column next to the task name is the percent of all the scenarios studied where that task was invoked at least once. The numbers inside the table are the average number of invocations of that task per second, per scenario. The column on the right is a ratio between the highest and lowest scheduling rate for that task. The row labeled "tidleTask" indicates the number of times a second the CPU switches into idle.

Several observations come out of this table. With the exception of MICAS, every scenario includes some executions of virtually every task, but with a significant variation in how often each task is invoked. Within a scenario, there is an even bigger variation in scheduling rates of the different tasks.

5.2. Execution Breakdown by Task

Table 4 is similar in structure to the previous table, but with the main table entries indicating what percent each task's execution takes up out of a second of execution of the specified scenario. A blank entry indicates that the task is executed for less than 0.1% of a second. This table thus indicates how a CPU spends its time over the average second, during each scenario. The "tidleTask" is again the time the CPU is idle.

The "Ratio" row is the ratio of the largest entry over the smallest non-zero entry, as evaluated over each scenario. The "Ratio" column on the right gives an equivalent ratio over each task.

The tasks that had a significant amount of processing divided into three groups. The first are relatively low level processing that is constant across all scenarios. The second are also relatively low, but much more scenario dependent. Finally, are heavy tasks that are scenario dependent.

Table 3. Task scheduling characteristics

	%M	PLoo	PLof	PLff	DSoo	DSof	DSff	DBoo	DBof	MPoo	MPof	MPff	ODoo	ODof	ODff	IPoo	IPof	IPff	MIoo	
% Busy		3.1	15.3	5.6	4.0	4.8	11.4	10.1	15.9	37.5	40.7	51.9	37.8	33.6	45.7	60.0	39.9	57.6	62.4	
Ratio		796	808	1274	1019	699	752	716	731	941	924	950	546	503	469	1162	1554	1573		
% of all Tasks		89%	89%	95%	91%	91%	91%	91%	91%	91%	91%	93%	91%	91%	95%	89%	89%	95%	4%	Ratio
tidleTask	100%	103	95.5	128	104	106	102	99.2	107	252	253	202	206	238	149	110	156	151	178	2.6
strokeWDT	17%	0.0	0.0	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.9	0.0	0.0	2.2	0.0	1.2
health	94%	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.3	0.3	0.2	0.4	0.5	0.3	0.1	0.1	0.1	0.0	4.7
bus	94%	25.7	25.3	29.5	24.6	25.6	26.3	24.2	24.9	25.7	24.6	27.0	26.7	23.7	26.4	27.4	36.6	34.6	0.0	1.5
dfi	94%	6.3	6.6	7.7	6.6	6.4	6.8	6.3	6.7	6.7	6.8	6.8	6.4	7.1	7.0	7.9	11.3	11.0	0.0	1.8
eep_sec_hi	94%	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.3	0.3	0.2	0.4	0.5	0.3	0.1	0.1	0.1	0.0	4.7
eep_file	94%	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.3	0.3	0.2	0.4	0.5	0.3	0.1	0.1	0.1	0.0	4.7
eep_file_small	94%	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.3	0.3	0.2	0.4	0.5	0.3	0.1	0.1	0.1	0.0	4.7
eep_sec_lo	94%	0.1	0.1	0.1	0.1	3.3	0.1	0.1	0.1	0.3	0.3	0.2	0.4	0.5	0.3	0.1	0.1	0.1	0.0	33.5
micas	94%	0.3	0.8	0.8	0.2	0.3	1.1	1.0	0.7	0.5	0.3	0.4	0.8	0.5	0.6	72.4	76.9	87.3	0.0	419.6
acs	94%	19.6	20.0	23.9	19.4	19.6	20.0	19.1	28.6	20.9	21.6	21.0	19.6	19.4	20.7	21.6	32.3	30.7	0.0	1.7
fpr	94%	1.0	1.1	1.4	0.9	1.1	1.1	1.0	1.0	1.1	1.1	0.8	1.1	0.9	1.3	1.3	3.0	2.6	0.0	3.5
t1	11%	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	1.0
t2	94%	104.3	96.2	121.7	106.0	105.4	104.6	100.1	98.9	106.2	104.7	97.9	93.7	96.1	82.9	116.4	145.1	157.5	0.0	1.9
sdst	94%	0.4	0.7	0.8	3.1	2.7	2.9	2.9	2.6	0.8	0.8	0.6	0.4	0.5	0.3	0.6	0.7	0.8	0.0	9.8

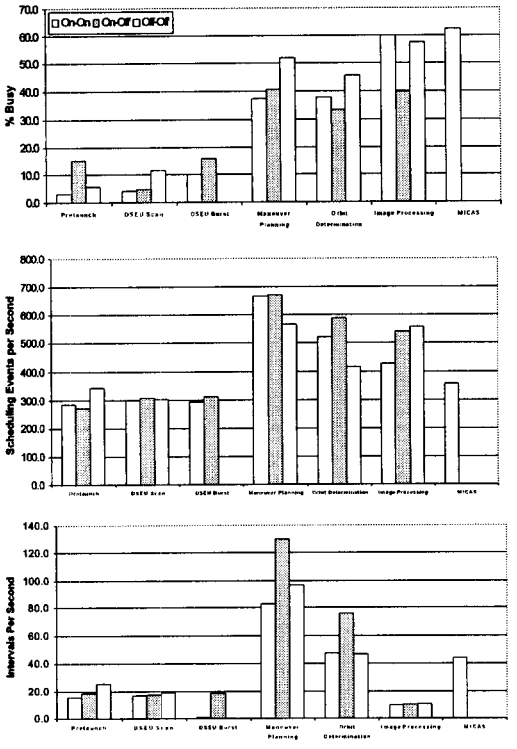
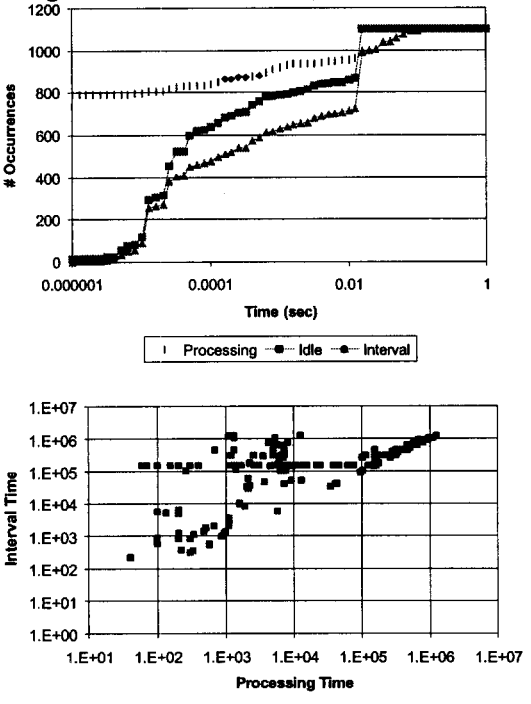


Figure 9. Characteristics by scenario



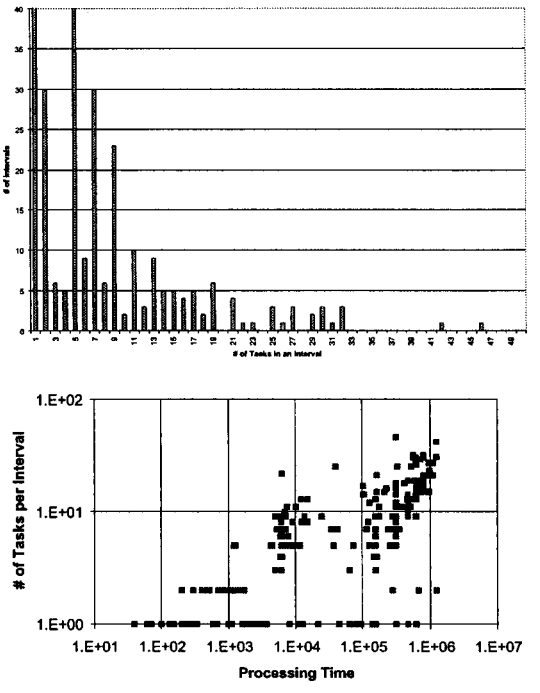


Figure 10. Interval Statistics

5.3. Intervals

An interval is a period of time that starts at the end of an idle period when a string of tasks are executed back to back, followed by another idle period. Figure 10(a) gives a cumulative distribution of both the processing, idle, and total times of all intervals in the Image Processing scenario (both L2 and DPM on). The 50% point for intervals as a whole is at about 300 microseconds, while the equivalent 50% point for idle periods is at about 70 microseconds. On the other hand, more than 50% of all intervals are less than 10 usec.

Figure 10(b) gives a different view of intervals for the same scenario- a histogram of the number of tasks in the processing part of an interval. This is an exponential drop-off, with a long tail - a significant number of intervals have thirty of more tasks present in them.

Next, Figure 10(c) gives a scatter diagram where each point comes from an interval in the Image Processing scenario, L2 and DPM on. Time is in microcesonds. There are two strong lines that show up here. First is a line at 0.125 seconds which must represent the 8 Hz period. Second is a 45 degree line representing intervals with very short idle times.

Finally, Figure 10(d) gives a scatter plot of time versus number of tasks in an interval. A strong horizontal line at 1 task indicates very simple intervals whose pro-

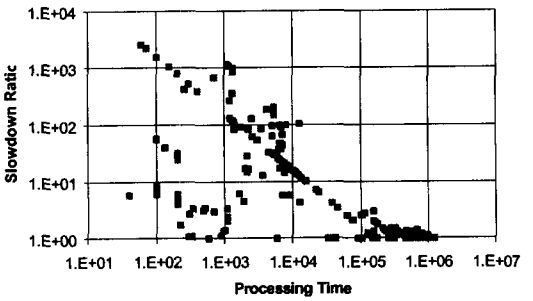


Figure 11. Slowdown Ratio vs. Processing Time

cessing times can vary significantly. Second is a band that grows with the processing time. More tasks here translates into increased processing time.

A key part of the power modeling work done on Morph was based on slowdowns that could be imposed on the processing parts of intervals. Figure 11 gives yet another scatter plot on the same scenario, this time with the slowdown ratio of total interval time to processing time. What is key is the large number of intervals with ratios in the tens, hundreds, and thousands. Also apparent is the diagonal line downward. This probably corresponds to the horizontal line in the prior chart, where interval time was independent of processing time.

6. Summary and Conclusions

This paper has discussed a very detailed analysis of a complex suite of real-time applications running together on a single processor under realistic data. Our analysis of the trace data for the Deep Impact application reveals the presence of a large dynamic range of the processing activities across the various phases of this application. CPU utilizations in-between idle periods can vary dramatically; the number of scheduling events within each phase as well as the number of intervals per second within a phase can both vary significantly from one phase to another. Understanding the dynamics of these statistics will be invaluable in understanding how best to craft efficient real-time embedded processing systems, especially for space applications.

As a particular example, the project originating this work, MORPH, has as its core development of a processor architecture and its associated run-time system that is designed to cope with such processing dynamics and actually exploit the dynamics to conserve the energy required for such processing. Rather than using a conventional microarchitecture that always dedicates a fixed set of resources for the processing, regardless of the dynamics of the application, the MORPH processor allocates data path resources such as register file segments, issue queue and reorder buffer partitions [7] and clusters of function units, register files and cache partitions on the basis of the dynamic demands of the application [4]. The main idea is to conserve the energy requirements of the application by allocating just the right amount of resources to meet the instantaneous performance needs of the computation. Additionally, further energy conservation is achieved by deactivating byte-slices (within the data path's interconnections, storage artifacts and function units) that do not contribute to the results produced [8]. The MORPH run-time system incorporates functions for explicitly controlling data placement and its movement within the memory hierarchy, for implementing power-aware real-time scheduling algorithms (including facilities for traditional voltage and frequency scaling) and for selective activation and deactivation of partitions of an inherently low-power cache system. Our planned future activities include the incorporation of these primitives into VxWorks and an evaluation of the efficacy of our solutions using the testbed described in this paper.

7. Acknowledgements

This work was funded in part by the DARPA Power Aware Computing and Communication (PAC/C) program under contract F30602-00-2-0525.

8. References

[1] http://www.cse.nd.edu/~cse_proj/morph

[2] http://www.ss.astro.umd.edu/deepimpact/

[3] "DI PPC750 Evaluation", JPL Internal Document 2001

[4] Peter M. Kogge, et al, 'A Comparative Analysis of Power and Energy Management Techniques in Real Embedded Applications," *IWIA'03*, IEEE, Kauai, HI, Jan. 24, 2003.

[5] Peter Kogge, et al, "Morph: Adding an Energy Gear to a High Performance Microarchitecture for Embedded Applications," *Kool Chips Workshop, MICRO-33*, IEEE Monterey, CA, Dec. 10, 2000.

[6] Jeffrey Namkung, "An Event-Level Power Measurement Methodology and Analysis," Master's Thesis, May 2001.

[7] Dmitry Ponomarev, G. Kucuk, K. Ghose, "Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources," MICRO-34, Dallas, TX, Dec. 2001, pp.90-101

[8] Dmitry Ponomarev, G. Kucuk, K. Ghose, " Dynamic Allocation of Datapath Resources for Low Power", Workshop on Complexity-Effective Design (WCED'01), ISCA-28, Goteborg, Sweden, June 2001.