Using Per-Loop CPU Clock Modulation for Energy Efficiency in OpenMP Applications

Wei Wang, University of Delaware
Allan Porterfield, RENaissance Computing Institute
John Cavazos, University of Delaware
Sridutt Bhalachandra, University of North Carolina at Chapel Hill

September 3, 2015
Table: Performance, power, and energy efficiency of top/green500 and exascale systems

<table>
<thead>
<tr>
<th>System Name</th>
<th>Performance (TFLOP/s)</th>
<th>Power (KW)</th>
<th>GFLOPS/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exascale System</td>
<td>1,000,000</td>
<td>20,000</td>
<td>50</td>
</tr>
<tr>
<td>MilkyWay-2</td>
<td>33,862.7</td>
<td>17,808</td>
<td>1.901</td>
</tr>
<tr>
<td>Titan</td>
<td>17,590.0</td>
<td>8,209</td>
<td>2.143</td>
</tr>
<tr>
<td>Sequoia</td>
<td>17,173.2</td>
<td>7,890</td>
<td>2.176</td>
</tr>
<tr>
<td>Shoubu</td>
<td>353.9</td>
<td>50.32</td>
<td>7.032</td>
</tr>
<tr>
<td>Suiren Blue</td>
<td>193.3</td>
<td>28.25</td>
<td>6.842</td>
</tr>
<tr>
<td>Suiren</td>
<td>202.6</td>
<td>32.59</td>
<td>6.217</td>
</tr>
</tbody>
</table>

Exascale computing requires more than $20 \times$ improvement in GFLOPS/Watts.
Motivation

(a) LULESH-MemoryIntensiveLoop: Energy reduced and EDP lowered with low performance impact

(b) LULESH-ComputeIntensiveLoop: Energy reduced with big performance slowdown and increased EDP
Motivation

(a) LULESH-MemoryIntensiveLoop: Energy reduced and EDP lowered with low performance impact

(b) LULESH-ComputeIntensiveLoop: Energy reduced with big performance slowdown and increased EDP

Loops of the same application prefer different frequencies
Issues

- DVFS mostly applied in coarse-grain cases
- Fine-grained (per-loop) energy control requires faster frequency transition techniques
- Could other power management technique (e.g. Clock Modulation/Duty Cycle Modulation) help?
CPU Clock Modulation

- Write Specific Value to IA32_CLOCK_MODULATION (0x19a) MSR
- Modify /dev/cpu/cpu{0:15}/msr with root privilege
- Invoke wrmsr inline assembly from applications using added System Call

## Available Frequencies

<table>
<thead>
<tr>
<th>Duty Cycle Level</th>
<th>Binary</th>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Effective Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001B</td>
<td>17</td>
<td>11H</td>
<td>6.25%</td>
</tr>
<tr>
<td>2</td>
<td>10010B</td>
<td>18</td>
<td>12H</td>
<td>12.5%</td>
</tr>
<tr>
<td>3</td>
<td>10011B</td>
<td>19</td>
<td>13H</td>
<td>18.75%</td>
</tr>
<tr>
<td>4</td>
<td>10100B</td>
<td>20</td>
<td>14H</td>
<td>25%</td>
</tr>
<tr>
<td>5</td>
<td>10101B</td>
<td>21</td>
<td>15H</td>
<td>31.25%</td>
</tr>
<tr>
<td>6</td>
<td>10110B</td>
<td>22</td>
<td>16H</td>
<td>37.5%</td>
</tr>
<tr>
<td>7</td>
<td>10011B</td>
<td>23</td>
<td>17H</td>
<td>43.75%</td>
</tr>
<tr>
<td>8</td>
<td>11000B</td>
<td>24</td>
<td>18H</td>
<td>50%</td>
</tr>
<tr>
<td>9</td>
<td>11001B</td>
<td>25</td>
<td>19H</td>
<td>56.25%</td>
</tr>
<tr>
<td>10</td>
<td>11010B</td>
<td>26</td>
<td>1AH</td>
<td>63.5%</td>
</tr>
<tr>
<td>11</td>
<td>11011B</td>
<td>27</td>
<td>1BH</td>
<td>69.75%</td>
</tr>
<tr>
<td>12</td>
<td>11100B</td>
<td>28</td>
<td>1CH</td>
<td>75%</td>
</tr>
<tr>
<td>13</td>
<td>11101B</td>
<td>29</td>
<td>1DH</td>
<td>81.25%</td>
</tr>
<tr>
<td>14</td>
<td>11110B</td>
<td>30</td>
<td>1EH</td>
<td>87.5%</td>
</tr>
<tr>
<td>15</td>
<td>11111B</td>
<td>31</td>
<td>1FH</td>
<td>93.75%</td>
</tr>
<tr>
<td>16</td>
<td>00000B</td>
<td>0</td>
<td>00H</td>
<td>100%</td>
</tr>
</tbody>
</table>
Multi-frequency Execution of Multi-loop Applications

- Adding energy control APIs around loops
- Fine-grain loop regions require fast machine power-state transition to avoid overhead

```java
while (condition) {
    ...
    setLowFrequency();
    for (i=0; i<N; i++) {
        ...
    }
    resetFrequency();

    for (j=0; j<N; j++) {
        ...
    }
}
```
DVFS vs. Clock Modulation (Entire LULESH App)

Multi-Frequency Versions

Normalized Metrics

Time, Energy, EDP

Figure: CPU Clock Modulation.
Polybench Loops

Normalized Metrics

Benchmarks

Time
Energy
EDP
Frequency

durbin
adi
fddt-2d
lu
cholesky
floyd-warshall
gemver
covariance
gramschmidt
doitgen
fdtd-apml
trisolv
atax
bicg
seidel-2d
reg-detect
sv2k
symm
Loops have different energy characteristics responding to frequency changes.
## Multi-frequency Execution: LULESH Results

### Table: Comparison of execution time, energy consumption, and EDP for LULESH

<table>
<thead>
<tr>
<th>Version</th>
<th>Duty Cycle Level</th>
<th>Time</th>
<th>Energy</th>
<th>EDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>minT</td>
<td>100%</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>minE</td>
<td>56.25%</td>
<td>1.542</td>
<td>0.743</td>
<td>1.145</td>
</tr>
<tr>
<td>minEDP</td>
<td>81.25%</td>
<td>1.157</td>
<td>0.816</td>
<td>0.943</td>
</tr>
<tr>
<td>MultiFreq 1</td>
<td>100% &amp; 50%</td>
<td>1.049</td>
<td>0.882</td>
<td>0.925</td>
</tr>
<tr>
<td>MultiFreq 2</td>
<td>100% &amp; 62.5%</td>
<td>1.020</td>
<td>0.897</td>
<td>0.914</td>
</tr>
<tr>
<td>MultiFreq 3</td>
<td>100% &amp; 68.75%</td>
<td>1.015</td>
<td>0.914</td>
<td>0.928</td>
</tr>
</tbody>
</table>
Multi-frequency Execution: miniFE Results

**Table:** Comparison of execution time, energy consumption, and EDP for miniFE

<table>
<thead>
<tr>
<th>Version</th>
<th>Duty Cycle Level</th>
<th>Time</th>
<th>Energy</th>
<th>EDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>minT</td>
<td>100%</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>minE</td>
<td>62.5%</td>
<td>1.351</td>
<td>0.763</td>
<td>1.031</td>
</tr>
<tr>
<td>minEDP</td>
<td>81.25%</td>
<td>1.153</td>
<td>0.819</td>
<td>0.945</td>
</tr>
<tr>
<td>MultiFreq 1</td>
<td>100% &amp; 81.25%</td>
<td>1.029</td>
<td>0.893</td>
<td>0.919</td>
</tr>
<tr>
<td>MultiFreq 2</td>
<td>100% &amp; 87.5%</td>
<td>1.023</td>
<td>0.923</td>
<td>0.944</td>
</tr>
<tr>
<td>MultiFreq 3</td>
<td>100% &amp; 93.75%</td>
<td>1.000</td>
<td>0.954</td>
<td>0.954</td>
</tr>
</tbody>
</table>
Clock Modulation with Concurrency Throttling

- Concurrency Throttling mitigates resource contention
- Clock Modulation reduces idle state power

(a) Energy with concurrency throttling and clock modulation. Minimum occurs at (75%, 4)

(b) Time with concurrency throttling and clock modulation. Minimum occurs at (100%, 6)

Figure: fdtd-2d Polybench
(a) Energy results. Minimum occurs at (75%, 6)

(b) Time results. Minimum occurs at (100%, 8)

<table>
<thead>
<tr>
<th>Version</th>
<th># of Threads</th>
<th>Duty Cycle Level</th>
<th>Time</th>
<th>Energy</th>
<th>EDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>16</td>
<td>100%</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CT</td>
<td>6</td>
<td>100%</td>
<td>0.92</td>
<td>0.94</td>
<td>0.87</td>
</tr>
<tr>
<td>CT+CM</td>
<td>6</td>
<td>75%</td>
<td>0.95</td>
<td>0.87</td>
<td>0.83</td>
</tr>
</tbody>
</table>
(a) Energy results. Minimum occurs at (75%, 8)

(b) Time results. Minimum occurs at (100%, 14)

<table>
<thead>
<tr>
<th>Version</th>
<th># of Threads</th>
<th>Duty Cycle Level</th>
<th>Time</th>
<th>Energy</th>
<th>EDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>16</td>
<td>100%</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CT</td>
<td>10</td>
<td>100%</td>
<td>1.02</td>
<td>0.86</td>
<td>0.88</td>
</tr>
<tr>
<td>CT+CM</td>
<td>10</td>
<td>81.25%</td>
<td>1.06</td>
<td>0.79</td>
<td>0.84</td>
</tr>
</tbody>
</table>
When Concurrency Throttling is Not Beneficial

(a) Energy results. Minimum occurs at (75%, 16)

(b) Time results. Minimum occurs at (100%, 16)

Figure: brdr2d results when applying both concurrency throttling and clock modulation.
Memory Access Density vs. Three Types of Loops

Benchmarks

durbin  adi  jacobi-2d-imper  fddt-2d  lu  cholesky  floyd-warshall  gesummv  covariance  gramschmidt  ludcmp  dynprog  2mm  3mm  syrk  trmm  correation  dojgen  fddt-apml  trisolv  ataix  bicg  seidel-det  2k  syymm

Normalized EDP

MAD Value

EDP  MAD

0  0.75  0.8  0.85  0.9  0.95  1  1.05  1.1

0  10  20  30  40  50

0  0.7  0.75  0.8  0.85  0.9  0.95  1  1.05  1.1
Memory Access Density could be used as loop type indicator.
Concurrency Throttling and Memory Access Density

Normalized Time/EDP vs. Number of Threads for various benchmarks.
Loops with high MAD value tend to benefit from concurrency throttling.
Conclusion

1. Multi-frequency execution of OpenMP loops with Clock Modulation can achieve better energy efficiency.
2. Concurrency throttling can be combined with Clock Modulation to save more energy.
Acknowledgment

1. U.S. Department of Defense: ATPER project
2. National Science Foundation: Award Number 1218734
Thanks!
backup slides: dvfs vs. dcm

measure and compare the execution time and power of loop1 and loop2 with and without energy control apis.

```c
while (condition)
{
    ...
    //Loop1
    MemLoop();
    //Loop2
    CompLoop();
    OtherLoops();
}
```

vs.

```c
while (condition)
{
    ...
    setFrequency();
    MemLoop();
    resetFrequency();
    CompLoop();
    OtherLoops();
}
```
DVFS vs. Clock Modulation (Loop Analysis)

(a) LULESH-MemoryIntensiveLoop

(b) LULESH-ComputeIntensiveLoop
DVFS vs. Clock Modulation (Loop Analysis)

Clock Modulation Performs Slightly Better Than DVFS
Benchmarks and Experimental Setup

1. Benchmarks
   - LULESH: Hydrodynamics
   - miniFE from Mantevo Project: implicit finite-element application
   - brdr2d: 2D Cardiac Wave Propagation Simulation
   - Polybench: 30 Computational Kernels

2. Hardware/Software Setup
   - Intel Xeon E5-2680 (Dual Socket, 8-core processor with 20MB LLC, 2.7GHz)
   - Linux 2.6.32 with ACPI and MSR modules
   - Intel ICC v14.0.2 with -O3