MERLOT: Architectural Support for Energy-Efficient Real-time Processing in GPUs

Muhammad Santriaji
santriaji@uchicago.edu

Henry Hoffmann
hankhoffmann@cs.uchicago.edu
GPU
Importance
Problem:

High power

Safety critical application needs Hard Real Time Guarantee
Challenges in GPU:

1. Reduce Energy.
2. Hard real time.
Slack Transfer

Kernel 1 finished early

Kernel 2 runs with less resource
Slack Transfer Problem in Software

- Checkpoints between kernels.
- GPU Application characteristics:
  - Long kernel length.
  - Not enough kernel.
- To reduce memory copy overhead.
GPU Application

• GPU Applications consist of kernels.
• Kernels consists of CTAs.
• CTA consists of threads.
GPU Application

• CTA is queued because lacks of resources.
• CTA can be used as checkpoints.
Uniqueness in GPU

• Hard real time needs checkpoints.
• CTA can be used as checkpoints.
  – A lot of them.
  – Without editing the GPU code.
Scenario

• User gives information about
  – Number of CTA.
  – WCET corresponds to the time to finish the CTA.

• MERLOT configure the resources such that:
  – Energy is decreased
  – Hard real time is guaranteed
Example
MERLOT Design
Hardware extension needed

Every time CTA is finished, counter is increased.
Approach

- Slack transfer in every CTA checkpoint.

\[ \eta = \text{slowness from slack transfer} \]
\[ \lambda = \text{frequency derived from slowdown} \]

3: procedure CHECKPOINT HANDLER
4: if \( nCTA_j = CTA_{\text{current}} \) then
5: \[ t_{\text{ahead}} = wCTA_j - t_{\text{current}} \]
6: \[ t_{\text{remaining}} = wCTA_{j+1} - wCTA_j \]
7: \[ \eta = \frac{t_{\text{remaining}} + o}{t_{\text{remaining}} + t_{\text{ahead}} + o} \]
8: \[ \lambda = \lceil f_{\text{max}} \cdot \eta \rceil \]
9: \[ j++ \]
Approach

• Limiting the slowdown (f critical)

```plaintext
11: if \( \lambda < f_{crit} \) then
12: \hspace{1em} if ALU utilization < \( th_{ALU} \) then
13: \hspace{2em} f_{SM} \leftarrow f_{crit} - 1
14: \hspace{2em} f_{mem} \leftarrow f_{crit}
15: \hspace{1em} else
16: \hspace{2em} f_{SM} \leftarrow f_{crit}
17: \hspace{2em} f_{mem} \leftarrow f_{crit} - 1
18: \hspace{1em} else
19: \hspace{2em} if ALU utilization < \( th_{ALU} \) then
20: \hspace{3em} f_{SM} \leftarrow \lambda
21: \hspace{3em} f_{mem} \leftarrow \lambda + 1
22: \hspace{2em} else
23: \hspace{3em} f_{SM} \leftarrow \lambda
24: \hspace{3em} f_{mem} \leftarrow \lambda
```
Approach

• Monitoring the workload (ALU utilization)

```
11: if λ < f_{crit} then
12:   if ALU utilization < th_{ALU} then
13:     f_{SM} ← f_{crit} - 1
14:     f_{mem} ← f_{crit}
15:   else
16:     f_{SM} ← f_{crit}
17:     f_{mem} ← f_{crit} - 1
18: else
19:   if ALU utilization < th_{ALU} then
20:     f_{SM} ← λ
21:     f_{mem} ← λ + 1
22: else
23:     f_{SM} ← λ
24:     f_{mem} ← λ
```

Decreased fcrit based on workload.
Compute workload : lower memory frequency
Memory workload : lower SM frequency

Guarantee the hard real time because frequency always equal or above λ
Approach

- Cutoff unused resources

```plaintext
25: \( nSM = 0 \)
26: \textbf{for all} sm \textbf{in the GPU do}
27: \textbf{if} sm == active \textbf{then}
28: \( nSM = nSM + 1 \)
```
Implementation

• GPGPU-Sim
  – GPU simulator.
  – GTX480.
• GPU-Wattch
  – Acquire power
• FPGA
  – Show it is feasible
  – Get overhead
    • 0.151 microseconds.
    • 0.299 Watts.
Simulation Setup

- **Benchmark**
  - Rodinia Benchmark
  - Parboil Benchmark

- **WCET**
  - Measure the WCET time by MERLOT (dynamic).
  - Set the deadline = WCET.
    - Varying the WCET from default to 1.5X, 2X, 5X, 10X, 15X.

- **DVFS with 23.3 W idle power.**

<table>
<thead>
<tr>
<th>SM frequency</th>
<th>DRAM frequency</th>
<th>Normalized Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{SM}$</td>
<td>$f_{mem}$</td>
<td>frequency</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>700 MHz</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>600 MHz</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>500 MHz</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>400 MHz</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>300 MHz</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>200 MHz</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>100 MHz</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>924 MHz</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>792 MHz</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>660 MHz</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>528 MHz</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>396 MHz</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>264 MHz</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>132 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.55</td>
</tr>
</tbody>
</table>
Results

• Timing, never miss deadline.
• Left 1.5X WCET, Right 2X WCET
Results 1.5X WCET

- Race to Idle (default speed)  
  - 34% faster.
- Software (only reconfigure between kernels)  
  - 24% faster.
- MERLOT (slack transfer in CTA)  
  - 22% faster
Results 2X WCET

- Race to Idle (default speed)
  - 50% faster.
- Software (only reconfigure between kernels)
  - 43% faster.
- MERLOT (slack transfer in CTA)
  - 5% faster
Energy Results (1.5X)

- Software (only reconfigure between kernels)
  - 6.38% energy saving.
- MERLOT (slack transfer in CTA)
  - 16.43% energy saving.
Energy Results (2X)

- Software (only reconfigure between kernels)
  - 6.67% energy saving.
- MERLOT (slack transfer in CTA)
  - 17.03% energy saving.
Analysis

- One application one kernel (Hotspot) in 1.5X.
Analysis

• Too much slowdown in software (BFS).
• For (256){
  – BFS()}

![Graph showing normalized runtime and energy](image)
MERLOT sensitivity in huge WCET

- 5X saves 10.74%
- 10X saves 6.91%
- 15X saves 5.19%
MERLOT sensitivity in huge WCET

- Decreased because slowdown is limited by frequency critical.

- Energy saving = 1 - \( \frac{\text{MERLOT energy} + \text{idle time} \times \text{idle power}}{\text{default energy} + \text{idle time} \times \text{idle power}} \)
Sensitivity to new technology

• Idle power is decreased to 6.8W.
• Better power and performance tradeoff.
• Higher static/leakage power.
Results

• New technology

<table>
<thead>
<tr>
<th>Deadline</th>
<th>GTX 580 41.9W</th>
<th>GTX 580 83.8W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>33.93 %</td>
<td>29.76 %</td>
</tr>
<tr>
<td>2</td>
<td>34.92 %</td>
<td>30.61 %</td>
</tr>
<tr>
<td>5</td>
<td>30.09 %</td>
<td>26.98 %</td>
</tr>
<tr>
<td>10</td>
<td>24.54 %</td>
<td>23.08 %</td>
</tr>
<tr>
<td>15</td>
<td>20.89 %</td>
<td>20.31 %</td>
</tr>
</tbody>
</table>
MERLOT

- Slack transferring in CTA level.
- Reduce Energy with hard real time guarantee.
- Light overhead without editing GPU code.
- Check the code at https://github.com/santriaji/MERLOT.