Warped Gates: Gating Aware Scheduling and Power Gating for GPGPUs

Mohammad Abdel-Majeed*    Daniel Wong*
Murali Annavaram

Ming Hsieh Department of Electrical Engineering
University of Southern California

* Equal Contribution
Problem Overview

- Execution unit accounts for majority of energy consumption in GPGPU, even more than Mem and Reg!

- Leakage energy is becoming a greater concern with technology scaling

---

Component Energy Breakdown for GTX480[^1]

SP accounts for 98% of Execution Unit Leakage Energy
Execution units account for 68% of total on chip area
Power Gating Overview

- Cuts off leakage current that flows through a circuit block
- Power gate at SP granularity
- Important Parameters:
  - **Wakeup Delay** – Time to return to Vdd (3 cycles)
  - **Breakeven Time** – # of consecutive power gated cycles required to compensate PG energy overhead (9-24 cycles)
  - **Idle Detect** - # of idle cycles before power gating

### Static Energy Chart

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Cumulative energy savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Eoverhead to Sleep</td>
</tr>
<tr>
<td>1</td>
<td>Overhead to sleep and Wakeup</td>
</tr>
<tr>
<td>2</td>
<td>Busy</td>
</tr>
<tr>
<td>3</td>
<td>Uncompensated</td>
</tr>
<tr>
<td>4</td>
<td>Overhead</td>
</tr>
</tbody>
</table>

### Breakeven Time

- **Wakeup**
- **Cycle 1**
- **Cycle 1+BET**
- **Compensated**
- **Cycles>BET time**
- **Cycles>wakeup_delay**
- **Cycles>idle_detect**

---

Power Gating Challenges in GPGPUs
Power Gating Challenges in GPGPUs

- Traditional microprocessors experience idle periods many 10s of cycles long\(^[3]\)
- Int. Unit Idle period length distribution for hotspot
  - Assume 5 idle detect, 14 BET

![Graph showing idle period length distribution]

- Frequency
- Lost Opportunity
- Energy Loss or Neutral
- Energy Savings

Power Gating Challenges in GPGPUs

- Traditional microprocessors experience idle periods many 10s of cycles long[^3]
- Int. Unit Idle period length distribution for hotspot
  - Assume 5 idle detect, 14 BET

![Diagram showing idle period length distribution](image.png)

- Lost Opportunity
- Energy Loss or Neutral
- Energy Savings

Need to increase idle period length

Warp Scheduler Effect on Power Gating

Need to coalesce warp issues by resource type

Idle periods interrupted by instructions that are greedily scheduled
GATES:

Gating Aware Two-level Scheduler

Issue warps based on execution unit resource type
Gating Aware Two-level Scheduler (GATES)

Ready Warps

Idle periods are coalesced

Idle Period
Gating Aware Two-level Scheduler (GATES)

- Per instruction type active warps subset
- Instruction Issue Priority
- Dynamic priority switching
  - Switch highest priority when it out of ready warps
Effect of GATES on Idle Period Length

Need to further stretch idle periods

- ~3x increase in positive power gating events
- ~2x increase in negative power gating events
Blackout Power Gating

Forced idleness of execution units to meet BET
Blackout Power Gating

- Force idleness until break even time has passed
  - Even when there are pending instructions
- Would this not cause performance loss?
  - No, because of GPGPU-specific large heterogeneity of execution units and good mix of instruction types
Blackout Power Gating

- ~2.4x increase in positive PG events over GATES
  (GATES ~3x w.r.t. baseline)
Blackout Policies

Naïve Blackout
- GATES and Blackout is independent

Can lead to overaggressive power gating
Blackout Policies

<table>
<thead>
<tr>
<th>Naïve Blackout</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATES and Blackout is independent</td>
</tr>
</tbody>
</table>

- Can lead to overaggressive power gating
Blackout Policies

<table>
<thead>
<tr>
<th>Naïve Blackout</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATES and Blackout is independent</td>
</tr>
</tbody>
</table>

Can lead to overaggressive power gating
Blackout Policies

<table>
<thead>
<tr>
<th>Naïve Blackout</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATES and Blackout is independent</td>
</tr>
</tbody>
</table>

Can lead to overaggressive power gating
Blackout Policies

| Coordinated Blackout

Warp Scheduler (GATES)

Dynamic priority switching is Blackout aware

Idle Detect

SP0  SP1
Blackout Policies

Coordinated Blackout

Warp Scheduler (GATES)

Dynamic priority switching is Blackout aware

PG only when active warps count = 0
Blackout Policies

- Coordinated Blackout

  - Warp Scheduler (GATES)
  - Dynamic priority switching is Blackout aware
  - INT
  - Active Warp Count Based
  - PG only when active warps count = 0
Impact of Blackout

- Some benchmarks still show poor performance
  - Not enough active warps to hide forced idleness
- Goal is as close to 0% overhead as possible
Adaptive Idle Detect

Reducing Worst Case Blackout Impact
Adaptive Idle Detect

- Dynamically change idle detect to avoid aggressive PG
- Infer performance loss due to Blackout
  - "Critical Wakeup" – Wakeup that occur the moment blackout period ends

![Graph showing normalized runtime vs critical wakeups per 1000 cycles for various benchmarks.](image)

- heartwall (0.99)
- NN (0.99)
- backprop (0.99)
- hotspot (0.99)
- nw (0.99)
- btree (0.99)
- gaussian (0.99)
- bfs (0.98)
- srad (0.97)
- ibm (0.96)
- cutcp (0.90)
- LIB (0.60)
- kmeans (-0.30)
- MUM (-0.28)
- lavaMD (-0.24)

High Correlation vs Runtime
Adaptive Idle Detect

- Independent idle detect values for INT and FP pipelines
- Break execution time into epoch (1000 cycles)
- If critical wakeup > threshold, idleDetect++
- Conservatively decrement idleDetect every 4 epochs
- Bound idle detect between 5 – 10 cycles
Architectural Support

Conv_PG  Gating Aware Scheduler

Scoreboard

I_Buffer

2-bit type indicator

V Dec_INST

INT

FP

SFU

LD

Priority Logic

INT_ACTV

FP_ACTV

Arbiter

Issued Instruction

INT_RDY

FP_RDY

SFU_RDY

LDST_RDY

PG_Signals

BET_counter

Critical wakeuP trigger

Idle_detect value

Idle detect logic

Execution Units

PG_Status

2 counters keep track of number of INT/FP instr in active subset. Used to determine dynamic priority

Architectural Support | 27
University of Southern California
Evaluation
Evaluation Methodology

- GPGPU-Sim v3.0.2
  - Nvidia GTX480
- GPUWattch and McPAT for Energy and Area estimation
- 18 Benchmarks from ISPASS, Rodinia, Parboil
- Power Gating parameters
  - Wakeup delay – 3 cycles
  - Breakeven time – 14 cycles
  - Idle detect – 5 cycles
Coalescing idle periods – fewer, but longer, idle periods

Blackout reduces PG overhead by 26%

Warped Gates reduces PG overhead by 46%
Blackout/Warped Gates is able to save energy when ConvPG cannot

Warped Gates saves ~1.5x static energy w.r.t. ConvPG
Warped Gates save \(~1.5x\) static energy w.r.t. ConvPG

(ignores Integer only benchmarks)
Performance Impact

- Naïve Blackout has high overhead due to aggressive PG
- Both ConvPG and Warped Gates has ~1% overhead
Conclusion

- Execution units – largest energy usage in GPGPUs
- Static energy becoming increasingly important
- Traditional microprocessor power gating techniques ineffective in GPGPUs due to short idle periods
- GATES – Scheduler level technique to increase idle periods by coalescing instruction type issues
- Blackout – Forced idleness of execution unit to avoid negative power gating events
- Adaptive Idle Detect – Limit performance impact
- Warped Gates able to save 1.5x more static power than traditional microprocessor techniques, with negligible performance loss
Thank you!

Questions?