SIMD Re-convergence at Thread Frontiers

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Haicheng Wu, Sudhakar Yalamanchili,

Georgia Institute of Technology

December 7, 2011
Goals

• Understand the opportunities for earlier reconvergence than achieved by PDOM
Thread Frontiers

Simple, efficient support for unstructured control flow on Single-Instruction Multiple-Thread (SIMT) execution models

if ( (cond1() || cond2()) && (cond3() || cond4()) )
{
   ....
}

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr
Structured Control Flow

Hammock graphs: A subgraph with a single entry, and a single exit.

![Hammock graphs diagram](image)

- if-then-else
- for-loop/while-loop
- do-while-loop

**CFGs composed of these subgraphs are structured.**

Exit node is earliest opportunity to reconverge [1]

- This is the immediate post-dominator of the entry node

A subgraph with multiple entry points, or multiple exit points, is not a hammock. It is not structured. [1]

\[
\text{if ( (cond1() || cond2()) && (cond3() || cond4()) ) }
\{
\text{.....}
\}
\]

Unstructured Control Flow

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, SIMD Re-convergence at Thread Frontiers 8/50
Unstructured Control Flow

<table>
<thead>
<tr>
<th>Warp PC</th>
<th>Predicate Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2</td>
<td>T1  T2  T3</td>
</tr>
<tr>
<td>B3</td>
<td>T0</td>
</tr>
<tr>
<td>Exit</td>
<td>T0  T1  T2  T3</td>
</tr>
</tbody>
</table>

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

<table>
<thead>
<tr>
<th>Warp PC</th>
<th>Predicate Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3</td>
<td>T2 T3</td>
</tr>
<tr>
<td>PDOM</td>
<td>T1</td>
</tr>
<tr>
<td>B3</td>
<td>T0</td>
</tr>
<tr>
<td>Exit</td>
<td>T0 T1 T2 T3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
</tr>
<tr>
<td>B3</td>
<td>B3</td>
<td>B3</td>
<td>B3</td>
</tr>
</tbody>
</table>

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

Warp PC | Predicate Mask
---|---
B4 | T3
B5 | T2
PDOM | T1
B3 | T0
Exit | T0 T1 T2 T3

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
</tr>
<tr>
<td>B1</td>
<td>B1</td>
<td>B1</td>
<td>B1</td>
</tr>
<tr>
<td>B3</td>
<td></td>
<td>B3</td>
<td>B3</td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
<td>B4</td>
</tr>
</tbody>
</table>

cond1() -> B1
cond2() -> B2
cond3() -> B3
cond4() -> B4

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Sudhakar Yalamanchili, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr,
SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

<table>
<thead>
<tr>
<th>Warp PC</th>
<th>Predicate Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDOM</td>
<td>T2</td>
</tr>
<tr>
<td>PDOM</td>
<td>T1</td>
</tr>
<tr>
<td>B3</td>
<td>T0</td>
</tr>
<tr>
<td>Exit</td>
<td>T0 T1 T2 T3</td>
</tr>
</tbody>
</table>

\[ \text{PDOM} \quad \text{PDOM} \quad \text{PDOM} \quad \text{PDOM} \]

\[ \text{T0} \quad \text{T1} \quad \text{T2} \quad \text{T3} \]

\[ \text{B1} \quad \text{B2} \quad \text{B3} \quad \text{B4} \]

\[ \text{T0} \quad \text{T1} \quad \text{T2} \quad \text{T3} \]

\[ \text{Entry} \quad \text{T0} \quad \text{T1} \quad \text{T2} \quad \text{T3} \]

\[ \text{Exit} \quad \text{Entry} \quad \text{Entry} \quad \text{Entry} \]

\[ \text{B1} \quad \text{B2} \quad \text{B3} \quad \text{B4} \]

\[ \text{B1} \quad \text{B1} \quad \text{B1} \quad \text{B1} \]

\[ \text{B2} \quad \text{B2} \quad \text{B2} \quad \text{B2} \]

\[ \text{B3} \quad \text{B3} \quad \text{B3} \quad \text{B3} \]

\[ \text{B4} \quad \text{B4} \quad \text{B4} \quad \text{B4} \]

\[ \text{B5} \quad \text{B5} \quad \text{B5} \quad \text{B5} \]

\[ \text{PDOM - POP} \quad \text{PDOM - POP} \quad \text{PDOM - POP} \quad \text{PDOM - POP} \]

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Sudhakar Yalamanchili, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

Warp PC | Predicate Mask
---|---
PDOM | T1
B3 | T0
Exit | T0 T1 T2 T3

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
<td>Entry</td>
</tr>
</tbody>
</table>

| B1 | B1 | B1 | B1 |
| B3 | B3 | B3 | B3 |
| B4 | | | B4 |
| B5 | | | B5 |

PDOM - POP
PDOM - POP
PDOM - POP

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Sudhakar Yalamanchili, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

Warp PC  Predicate Mask

<table>
<thead>
<tr>
<th></th>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B3</strong></td>
<td>B3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exit</strong></td>
<td>T0</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
</tr>
</tbody>
</table>

```
bra cond1()
B4
B5.........
cond4()
cond2()
B2
cond3()
B3
 Entry
cond1()
T0  T1  T2  T3
Exit
T0      T1    ...                     B5B5
  B3B3
PDOM
T0B3
Warp PC Predicate Mask
PDOM - POP
PDOM - POP
PDOM - POP
T0 T1 T2 T3Exit
```
Unstructured Control Flow

Warp PC  Predicate Mask

<table>
<thead>
<tr>
<th>B4</th>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>T0</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
</tr>
</tbody>
</table>

**T0**

bra cond1()

**T1**

cond2()

**T2**

cond3()

**T3**

Entry

cond1()

**T0**

Entry

**T1**

Entry

**T2**

Entry

**T3**

Entry

<table>
<thead>
<tr>
<th>B1</th>
<th>B1</th>
<th>B1</th>
<th>B1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3</td>
<td>B3</td>
<td>B3</td>
<td>B3</td>
</tr>
<tr>
<td>B4</td>
<td>B4</td>
<td>B4</td>
<td>B4</td>
</tr>
<tr>
<td>B5</td>
<td>B5</td>
<td>B5</td>
<td>B5</td>
</tr>
<tr>
<td>B3</td>
<td>B3</td>
<td>B3</td>
<td>B3</td>
</tr>
<tr>
<td>B4</td>
<td>B4</td>
<td>B4</td>
<td>B4</td>
</tr>
<tr>
<td>PDOM</td>
<td>PDOM - POP</td>
<td>PDOM - POP</td>
<td>PDOM - POP</td>
</tr>
</tbody>
</table>

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Sudhakar Yalamanchili, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

Warp PC | Predicate Mask
---|---
B5 | T0 T1 T2 T3
Exit | T0 T1 T2 T3

---

T0 T1 T2 T3
Entry

B1
cond1() B4
B5

cond2()
B2
cond3()
B3
Entry
cond1()
T0 T1 T2 T3
Exit

T0 T1 T2 T3
Entry

B1 B1 B1 B1


B3 B3 B3 B3

B4 B4 B4 B4

B5 B5 B5 B5

PDOM - POP

PDOM - POP

PDOM - POP

PDOM - POP

T0 T1 T2 T3
Exit

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, SIMD Re-convergence at Thread Frontiers
Unstructured Control Flow

<table>
<thead>
<tr>
<th>Warp PC</th>
<th>Predicate Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDOM</strong></td>
<td>T0  T1  T2  T3</td>
</tr>
<tr>
<td><strong>Exit</strong></td>
<td>T0  T1  T2  T3</td>
</tr>
</tbody>
</table>

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr,
SIMD Re-convergence at Thread Frontiers
Unstructured control flow delays PDOM reconvergence.
Prevalance of Unstructured Control Flow

<table>
<thead>
<tr>
<th>Suite</th>
<th>Number of Benchmarks</th>
<th>Number of benchmarks with Unstructured Control Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA SDK</td>
<td>56</td>
<td>4</td>
</tr>
<tr>
<td>Parboil</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>Rodinia</td>
<td>20</td>
<td>9</td>
</tr>
<tr>
<td>Optix</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>Total</td>
<td>113</td>
<td>27</td>
</tr>
</tbody>
</table>

Sources of unstructured control flow [1]

- Exceptions
- Short-circuit conditionals
- goto statements
- State machines
- Compiler optimizations

SIMD Reconvergence at Thread Frontiers
Thread frontier: the set of other basic blocks where divergent threads may be waiting to execute.

Thread Frontier is defined for each basic block.

If a warp is executing BB2, all divergent threads from that warp will be found in the thread frontier of BB2.

An algorithm for computing thread frontiers statically is presented in paper.
Compiler Support

• Requirements
  ❖ #1: If a thread is disabled, it is waiting in the thread frontier of the basic block being executed
  ❖ #2: If a partially enabled warp (includes divergent threads) enters a basic block in its thread frontier, a check is made for reconvergence

Note that thread frontier lies between the IPDOM and the divergent branch
Conservative Branches

- T0 execution path: BB0, BB1, BB2, BB4, BB7
- T1 execution path: BB0, BB3, BB5, BB7
Scheduling for Early Reconvergence

Strategy:
- Prioritize execution of basic blocks
- Execute earlier blocks before later blocks

Technique:
- Define priority mapping of basic blocks
- Identify region where waiting threads may be ready to execute
- Jump to highest priority block $\in \{\text{active context}\} \cup \{\text{thread frontier}\}$
- Reconverge warps waiting at same program counter
Priority Execution

Branch to highest priority block
- **reverse**: jump to branch target
- **forward**: jump to thread frontier

**Figure:**
- **Reverse branch** (low to high priority)
- **Forward branch** (high to low priority)
Requirements for Hardware Support

• Compiler has determined priorities and TF for each basic block

• TF
  ❖ Establish priorities for Basic Blocks
  ❖ Priority based scheduling of threads/warps (basic blocks)
    o Use branch directions (Intel Sandy Bridge implementation)
  ❖ Check for stalled threads in the thread frontier
    o Hardware check
    o Conservative branches

• Scheduler – on a divergent branch
  ❖ Branches to higher priority blocks proceed normally
  ❖ Branches to lower priority blocks instead branch to the frontier
Hardware Support
(priority sets)
Conservative branch:

- No threads waiting in the Thread-Frontier during forward branch
- HW executes NO-OPs
- Reconverge with warps waiting in lower-priority basic block
Conservative Branches

(1) active warp PC

PC: BB4

per-thread program counters

<table>
<thead>
<tr>
<th>BB4</th>
<th>Exit</th>
<th>Exit</th>
<th>BB4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td></td>
<td></td>
<td>T3</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td>T2</td>
</tr>
</tbody>
</table>

Thread Frontier

{-BB5, Exit}

BB4

{Exit}

BB5

Exit

{ }
Conservative Branches

(2) PC: BB5

- Per thread program counters
- Layout where PC = priority
- if and continue branch instructions
Conservative Branches

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr

SIMD Re-convergence at Thread Frontiers
**Hardware Realization: Sorted Predicate Stack**

Dynamically determine thread frontiers
- Additional hardware: **sorted predicate stack**
- Execute highest-priority warp context

Branches result in new contexts
- Merge with existing contexts at same PC
- Insert into stack if unique PC
- Sorted by PC priority
Hardware Realization: Sorted Predicate Stack

Sorted by priority, descending

BB4
BB5
Exit T1
T0
T2
T3

active
highest priority

Thread Frontier

{ BB5, Exit }

per-thread program counters

PC: BB4

active warp PC

sorted by priority, descending

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr,

SIMD Re-convergence at Thread Frontiers
Hardware Realization: Sorted Predicate Stack

active warp PC

PC: BB5

per-thread program counters

BB5 Exit BB5 Exit

T0 T1 T2 T3

active

highest priority

BB5

T0 T2

Exit

T1 T3

sorted by priority, descending

Thread Frontier

BB4

BB5 { Exit }

Exit

sorted by priority, descending

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr, Sudoku Yalamanchili, SIMD Re-convergence at Thread Frontiers
Hardware Realization: Sorted Predicate Stack

active warp PC

per-thread program counters

sorted by priority, descending

Thread Frontier

sorted by priority, descending

Gregory Diamos, Benjamin Ashbaugh, Subramaniam Maiyuran, Andrew Kerr,
Performance Comparisons
Ocelot PTX Emulator [1]
- 32-thread warp
- simulated PTX instructions, in-order execution
- real-world applications with unstructured control flow

Reconvergence mechanisms under test
- **struct**: structural transformations then PDOM reconvergence
- **pdom**: immediate post-dominator reconvergence, no structural transformations
- **tf-stack**: thread frontiers with sorted predicate stack
- **tf-sandy**: thread frontiers modeled after Intel Sandybridge/GEN6

## Benchmark Applications

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background-subtract video analysis and object partitioning</td>
<td>visualizations of the Mandelbrot set (CUDA SDK)</td>
</tr>
<tr>
<td>Mandelbrot visualization of the Mandelbrot set (CUDA SDK)</td>
<td></td>
</tr>
<tr>
<td>MCX medical imaging RNG benchmark</td>
<td></td>
</tr>
<tr>
<td>Mummer GPU DNA sequence alignment using suffix tree searches</td>
<td></td>
</tr>
<tr>
<td>OptiX NVIDIA's graph traversal and ray tracing framework</td>
<td></td>
</tr>
<tr>
<td>Pathfinding multi-agent path planning</td>
<td></td>
</tr>
<tr>
<td>Photon-Transport stochastic simulation of light transport</td>
<td></td>
</tr>
<tr>
<td>CUDA Renderer ray tracing implementation in CUDA</td>
<td></td>
</tr>
<tr>
<td>Parboil Suite CUDA benchmark applications</td>
<td></td>
</tr>
<tr>
<td>except contains exceptions that are never thrown</td>
<td></td>
</tr>
<tr>
<td>short demonstrates short-circuit conditional evaluation</td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Instruction Counts

2.2%-633% reductions in dynamic instruction counts

- Compared to PDOM reconvergence
Thread frontiers reconvergence improves SIMD utilization and memory efficiency

- tf-sorted maximizes activity factor
- SIMT execution of threads enables coalescing memory transactions
Sorted stack typically has 1-2 entries

- Bounded by nesting depth
  - No more than 6 entries for these applications
- Maximum thread frontier size: 5 blocks
Conclusions

SIMD Re-convergence at thread frontiers
- Priority-based scheduling promotes early reconvergence
- Realizable in Intel Sandybridge/GEN6 GPU microarchitecture
- More efficient with sorted predicate stack

Advantages:
- Supports unstructured control flow
  - Exceptions, function calls, state machines, graph traversal
  - Incorrect analysis impacts performance, not correctness

Tradeoffs:
- Microarchitecture support required, replaces predicate stack
- Priority assignment requires whole-program analysis
Questions?

Contact Us:

gregory.diamos@gatech.edu
bashbaugh@intel.com,
subramaniam.maiyuran@intel.com,
arkerr@gatech.edu,
hwu36@gatech.edu,
sudha@ece.gatech.edu

Acknowledgements:

Download Ocelot:

GPU Ocelot: http://code.google.com/p/gpuocelot/
The authors gratefully acknowledge the generous support of this work by LogicBlox Inc., IBM Corp., and NVIDIA Corp. both through research grants, fellowships, as well as technical interactions, and equipment grants from Intel Corp. and NVIDIA Corp.