SIMD Re-Convergence At Thread Frontiers

Gregory Diamos  
Georgia Institute of Technology  
School of ECE  
Atlanta, GA, USA  
gregory.diamos@gatech.edu

Benjamin Ashbaugh  
Intel  
Visual Computing Group  
Folsom, CA  
bashbaugh@intel.com

Subramaniam Maiyuran  
Intel  
Visual Computing Group  
Folsom, CA  
subramaniam.maiyuran@intel.com

Andrew Kerr  
Georgia Institute of Technology  
School of ECE  
Atlanta, GA, USA  
arkerr@gatech.edu

Haicheng Wu  
Georgia Institute of Technology  
School of ECE  
Atlanta, GA, USA  
hwu36@gatech.edu

Sudhakar Yalamanchili  
Georgia Institute of Technology  
School of ECE  
Atlanta, GA, USA  
sudha@ece.gatech.edu

Background

• Unstructured control flow occurs frequently in applications, often causing thread divergence.
• SIMD processors such as the Intel Sandybridge GPU use Immediate Post-Dominator Re-convergence (PDOM).
• Per-Thread Program Counters (PTPCs) allow disabling of threads which don’t match the program’s PC.
Unstructured Control Flow

- Unstructured control flow occurs when programs have multiple entries and exits.
- Can often be caused by short-circuiting, breaks or continues, and exceptions.
Thread Frontiers

- Thread Frontier - a list of all blocks that might simultaneously contain threads from the same warp.

- Using thread frontiers, we can re-converge earlier than with post dominators.
Thread Frontier Re-convergence

- Each basic block is assigned a priority.
- The hardware executes them in priority order.
- The compiler inserts re-convergence checks wherever a partial-warp enters a block in its thread frontier.
- The hardware performs the specified checks.

```
Input: Set of basic blocks sorted from high to low priority
Output: Mapping from basic block to thread frontier
1   tset := {};
2   foreach basic block b in sorted set do
3     if b is in tset then
4       remove b from tset;
5     frontier(b) := {tset};
6     if b ends with a divergent branch then
7       foreach target t of this branch do
8         if priority(t) < priority(b) then
9           add t to tset;
```

Algorithm 1: Computes the thread frontier of each BB.
Barriers

- A divergent barrier can cause deadlocks when using PDOM re-convergence.
- Thread frontiers can often avoid this issue.
• This is essentially a workaround in the event that there is no hardware support for TF
Hardware Support (Scheduling)

• When taking a divergent branch to a block with lower priority, schedule the highest priority block inside the thread frontier instead.

• This ensures that the warp always executes the block of highest priority.
Sorted Stack

- Problem: A straightforward approach to finding the highest priority block is expensive.
- Instead, keep blocks with waiting threads inside a sorted stack, based on priority order.
- Re-convergence only occurs at the top of the stack, requiring only a single comparison.
- Claim: In real workloads, the sorted stack never has more than three unique items.
Sorted Stack (example)

Thread Frontiers-Sorted-Stack
sorted by program counter, ascending

(1) T3 branches forward from BB4 to Exit. T3 is added to the existing entry at Exit containing T1. T0 branches forward from BB4 to BB5, it merges with the existing context containing T2.

(2) BB5

(3) BB5

Highest priority
Experimental Evaluation

• The Ocelot open source compiler infrastructure was used to simulate an Intel Sandybridge GPU.
• STRUCT refers to altering the unstructured program to make it structured, using preexisting techniques.
• PDOM refers to using immediate post-dominators for re-convergence.
• TF-STACK refers to using TF and a sorted stack.
• TF-SANDY refers to using TF and conservative branches.
Results: Dynamic Instruction Count

- TF techniques, especially TF-STACK, proved much more effective on average than the other methods.
Results: Activity Factor

- TF techniques allowed higher active threads per warp on average.
Results: Memory Efficiency

- TF techniques gave the simulated GPU more opportunities, on average, to combine threads’ memory requests into single transactions.
Advantages, Disadvantages

- **STRUCT**
  - Can be implemented without hardware support.
  - Can severely inflate the size of the program.
- **PDOM**
  - Is already supported by commodity hardware.
  - Re-converges later than optimal; easily deadlocks on barriers.
- **TF-STACK**
  - Has the most effective re-convergence.
  - Requires hardware support to maintain a sorted stack.
- **TF-SANDY**
  - Has effective re-convergence, can be implemented without hardware support.
  - Will sometimes take paths that have no threads to execute, adding excessive overhead.
More Potential

• If different parts of the code call the same function, TF-STACK can allow the two instances of this function to run in lockstep. This aspect will become more important as the GPU adds more support for commonly used library functions.

• Unlike PDOM, TF can allow for efficient exception catching.