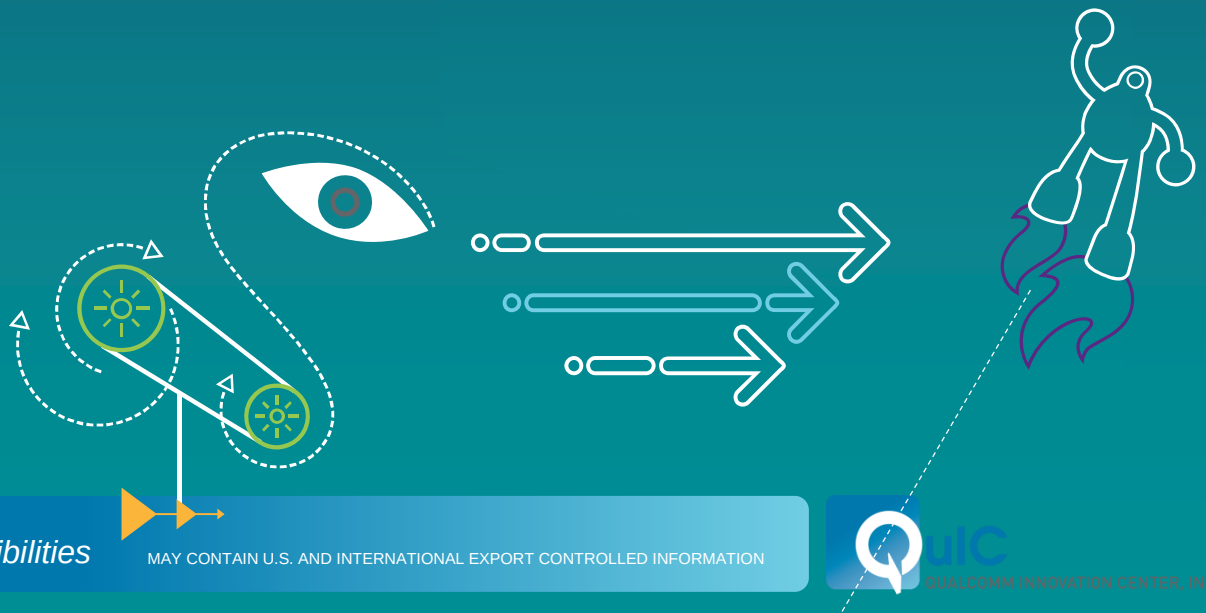
Brendon Cahoon, Qualcomm Innovation Center
Ikhlas Ajbar, Qualcomm Innovation Center

# An Implementation of Swing Modulo Scheduling in a Production Compiler

QUALCOMM®

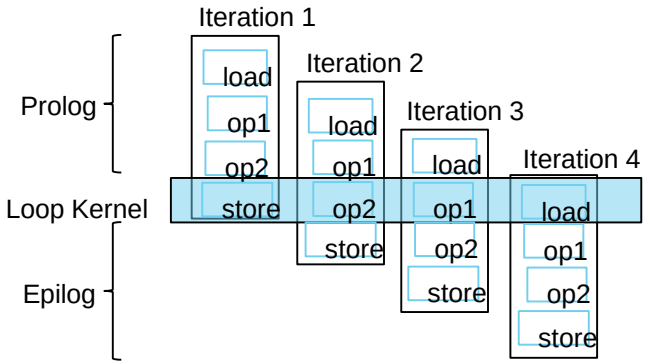




Open *Source* | Open *Possibilities*

# Swing Modulo Scheduling Algorithm

- Improves ILP in loops by overlapping iterations.




1. Calculate the minimum initiation interval
2. Analysis of data dependence graph
3. Order the nodes by priority.
4. Schedule the nodes in order.
   - If schedule fails, increase II by 1 and try again
5. Generate prolog(s), epilog(s), and new kernel.

Original code

```
    loop0(.LBB0_2, r2)
.LBB0_2:
    {
      r2 = memw(r1++#4)
      r4 = memw(r3++#4)
    }
    {
      r0 += mpyi(r2, r4)
      nop
    } :endloop0
```

Pipelined Kernel

```
.LBB0_2:
    {
      r5 += mpyi(r4, r2)
      r2 = memw(r3++#4)
      r4 = memw(r1++#4)
    } :endloop0
```

# Implementation in LLVM

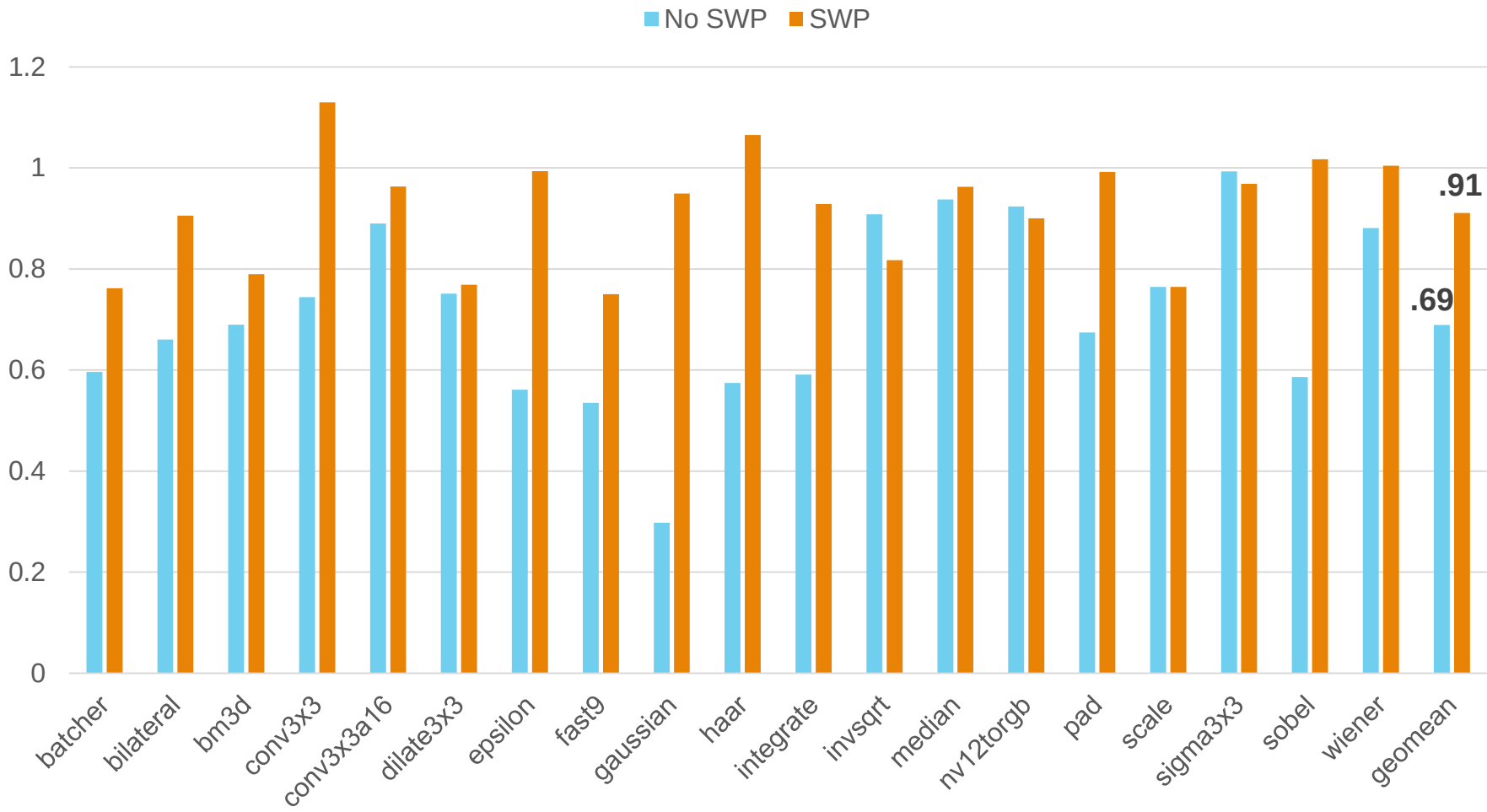
- A target independent back-end pass.
  - lib/CodeGen/MachinePipeliner.cpp
  - Added a few target hooks:
    - ReduceLoopCount(), AnalyzeLoop(), getIncrementValue()
- Use ScheduleDAGInstrs to build dependence graph.
  - Use chain edges to represent loop-carried dependences.
  - Significant post-processing of DAG.
- Called while in SSA form, prior to register allocation.
- DFAPacketizer checks resources and models parallelism.
- Works on a loop with a single basic block.
- Replace original loop with prolog(s), epilog(s), new kernel.
- Implementation for Hexagon
  - Enabled at –O2 and above
  - Largest MII is 27, Max number of overlapping iterations is 4

# Additions to original algorithm

- Model register pressure when calculating node order
  - Increase priority of node-sets that exceed register pressure
- Prioritize node-sets with common set of successors
- Additional heuristics for computing node order based upon previously scheduled instructions
- Attempt to order all instructions together
  - For loops with large MII and a large DAG depth
- Final ordering of instructions
  - Pipeliner models parallelism internally, but generates linear list of instructions

# Performance SWP vs. Hand-coded Assembly

Higher is better  Normalized, hand-coded assembly performance = 1



# Thank you

Follow us on:

For more information on Qualcomm, visit us at:
www.qualcomm.com & www.qualcomm.com/blog