Operation of the Basic SM Pipeline

Objectives

- Cycle-level examination of the operation of major pipeline stages in a stream multiprocessor
- A breadth first look at a basic pipeline
- Understand the type of information necessary for each stage of operation
- Identification of performance bottlenecks
  - Detailed implementations are addressed in subsequent modules
Objectives

Reading

- Documentation for the GPGPUSim simulator
  - Good source of information about the general organization and operation of a stream multiprocessor

- Operation of a Scoreboard

- General Purpose Graphics Architectures, T. Aamodt, W. Fung, and T. Rogers, Chapter 2.2
NVIDIA GK110 (Kepler)

Hierarchy of schedulers: kernel, TB, warp, memory transactions

Image from http://mandetech.com/2012/05/20/nvidia-new-gpu-and-visualization/

SMX Organization: GK 110

Multiple Warp Schedulers

64K 32-bit registers

192 cores – 6 clusters of 32 cores each

What are the main stages of a generic SMX pipeline?

Image from http://mandetech.com/2012/05/20/nvidia-new-gpu-and-visualization/
A Generic SM Pipeline

Scalar Fetch & Decode
Instruction Issue & Warp Scheduler
Predicate & GP Register Files
Scalar Pipelines
Data Memory Access
Writeback/Commit

Front-end
Scalar Cores
Back-end

Single Warp Execution

PTX (Assembly):

setp lt.s32 %p, %r5, %rd4;  //r5 = index, rd4 = N
%p bra L1;
bra L2;
L1:
ld.global.f32 %f1, [%r6];  //r6 = &a[index]
ld.global.f32 %f2, [%r7];  //r7 = &b[index]
add.f32 %f3, %f1, %f2;
st.global.f32 [%r8], %f3;  //r8 = &c[index]
L2:
ret;
Instruction Fetch & Decode

Examples from Harmonica2 GPU

Instruction Buffer

Example: buffer 2 instructions/warp

- Buffer a fixed number of instructions per warp
- Coordinated with instruction fetch
  - Need an empty I-buffer for the warp
- V: valid instruction in the buffer
- R: instruction ready to be issued
  - Set using the scoreboard logic

Instruction Buffer (2)

- Scoreboard enforces WAW and RAW hazards
  - Indexed by Warp ID
  - Each entry hosts required registers,
    - **Destination registers** are reserved at issue
  - **Reserved registers** released at writeback
- Enables multiple instructions to be in execution from a single warp


Instruction Buffer (3)

**Generic Scoreboard**

- Next: Modified scoreboard design to address
  - Have multiple instructions in transit
  - Excessive demand for register file ports

Instruction Issue

Instruction Issue (2)

• Barriers – warps wait here for barrier synchronization
  - All threads in the thread block must reach the barrier


(13)

(14)
Instruction Issue (3)

- Register Dependencies - track through the scoreboard


Instruction Issue (4)

- Control Divergence - per warp stack
- Create execution mask that is read with operands

Instruction Issue (5)

- Scheduler can issue multiple instructions from a warp
- Issue conditions
  - Has valid instructions
  - Not waiting at a barrier
  - Scoreboard check
  - Pipeline line is not stalled: operand access stage (will get to it later)
- Reserve destination registers
- Instructions may issue to memory, SP or SFU pipelines
- Warp scheduling disciplines $\Rightarrow$ more later in the course

Register File Access

Banks 0-15

Single ported Register File Banks

Arbiter

Operand Collectors (OC)

Dispatch Units (DU)

Xbar

ALUs

L/S

SFU

Banks 0-15
Scalar Pipeline

- Functional units are pipelined
- Designs with multiple issue

Shared Memory Access

- Multiple bank organization
- Data is interleaved across banks
- Bank conflicts extend access times
Memory Request Coalescing

- Pending Request Table (PRT) is filled whenever a memory request is issued.
- Generate a set of address masks → one for each memory transaction.
- Issue transactions.

From J. Leng et al., “GPUWattch: Enabling Energy Optimizations in GPGPUs,” ISCA 2013 (21)

Memory Hierarchy

- Configurable cache/shared memory configuration for L1.
- Read-only cache for compiler or developer (intrinsics) use.
- Shared L2 across all SMXs.
- ECC coverage across the hierarchy.
  - Performance impact.

From GK110: NVIDIA white paper (22)
Summary

- Synchronous progress of a warp through the SM pipelines
- Warp progress in a thread block can diverge for many reasons
  - Barriers
  - Control divergence
  - Memory divergence
- How is the execution optimized? Next →