CS 758: Advanced Topics in Computer Architecture

Lecture #5: Introduction to GPU Microarchitecture

Professor Matthew D. Sinclair

Some of these slides were developed by Tim Rogers at the Purdue University and Tor Aamodt at the University of British Columbia

Slides enhanced by Matt Sinclair

Announcements

- HW0 “due” today
  - Issues?
  - Questions?
- HW1 will be posted today
- Moving exam after Affiliates meeting -- poll

Objectives

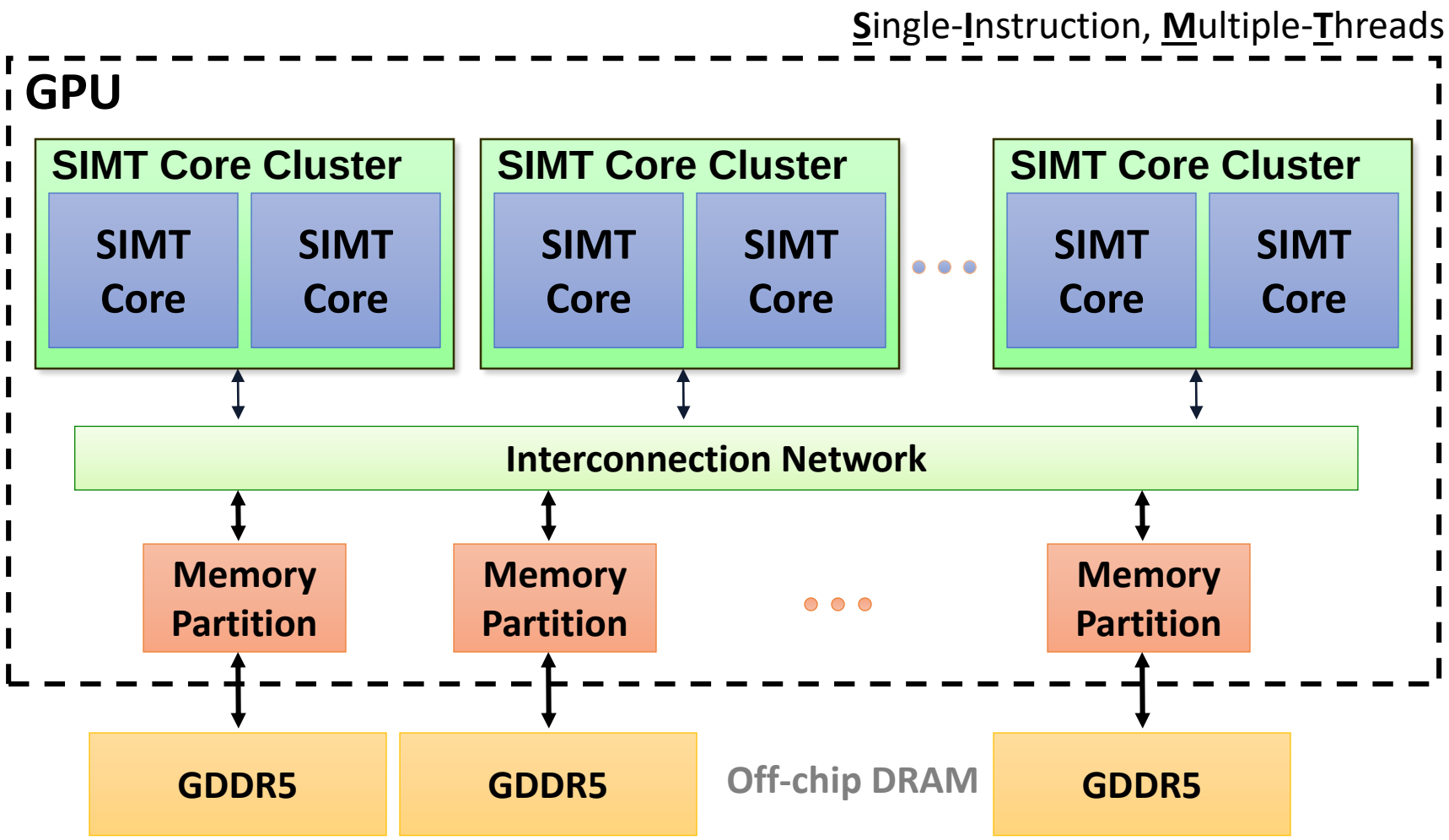
- A Simple GPU Pipeline
  - “One Scheduler Approximation”
- The SIMT Stack
- Control Flow Divergence Mitigation in Hardware

Extra resources

GPGPU-Sim 3.x Manual http://gpgpu-sim.org/manual/index.php/GPGPU-Sim_3.x_Manual

General Purpose GPU Architecture (Aamodt, Fung and Rogers)

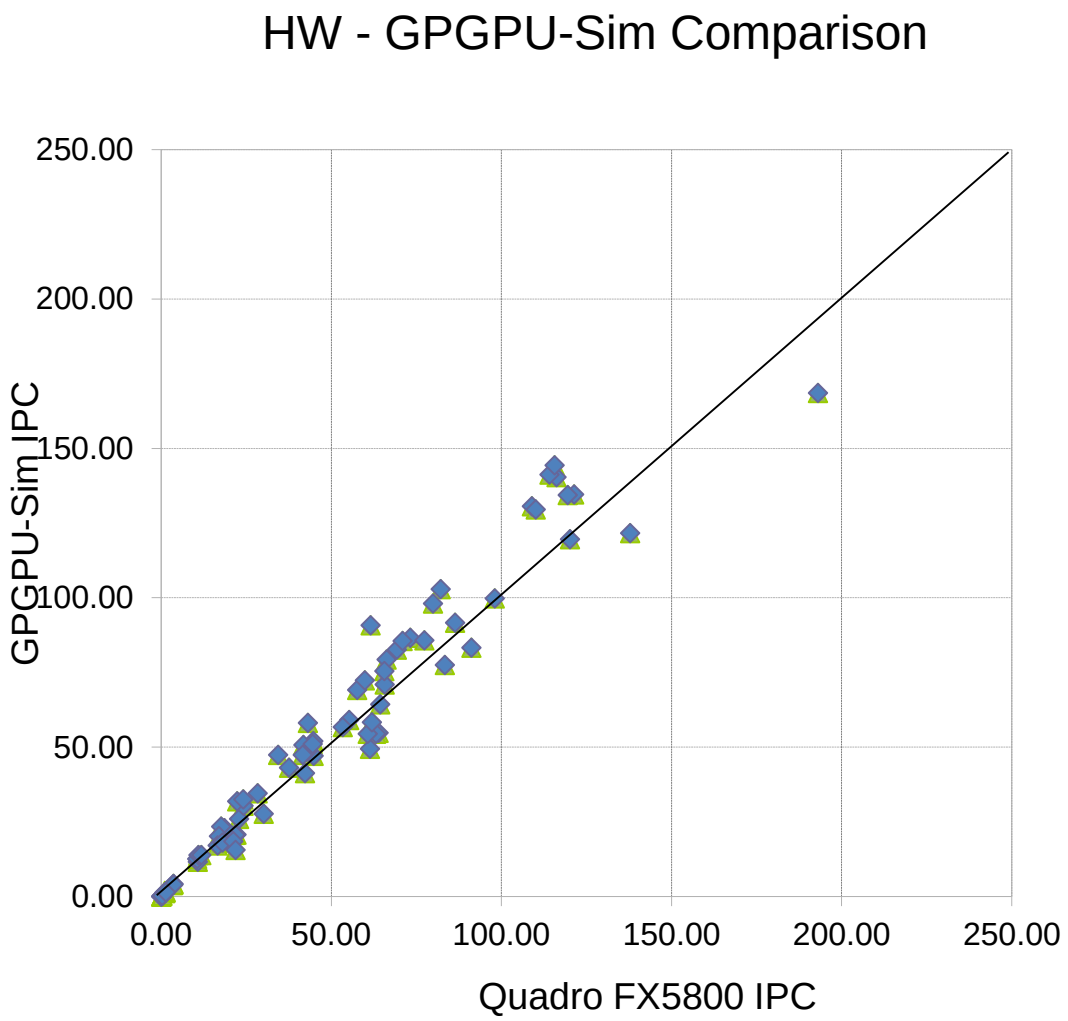
GPU Microarchitecture Overview



GPU Microarchitecture

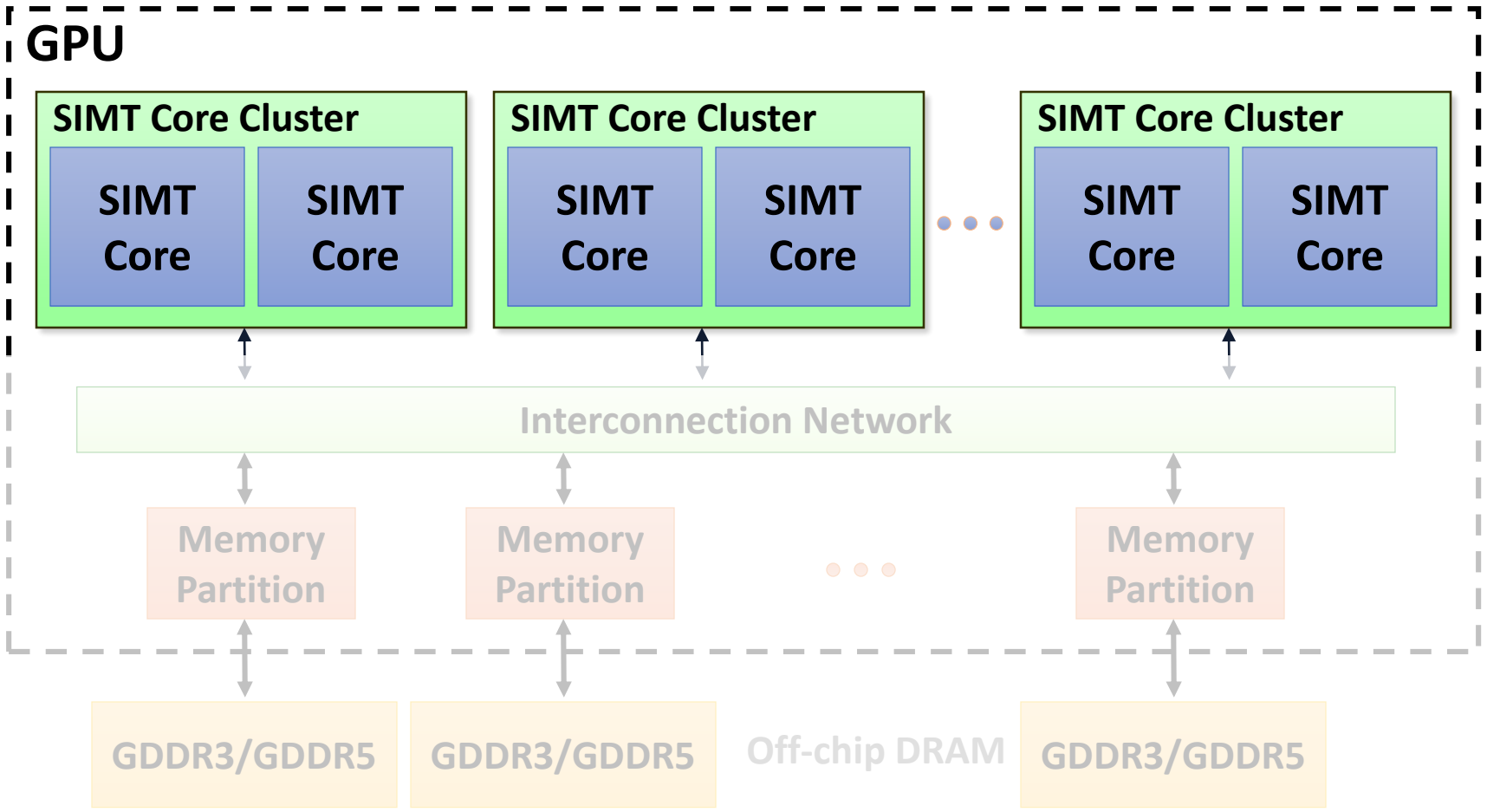
- Companies tight lipped about details of GPU microarchitecture.
- Several reasons:
  - Competitive advantage
  - Fear of being sued by “non-practicing entities”
  - The people that know the details too busy building the next chip
- Model described next, embodied in GPGPU-Sim, developed from: white papers, programming manuals, IEEE Micro articles, patents.

GPGPU-Sim v3.x w/ SASS

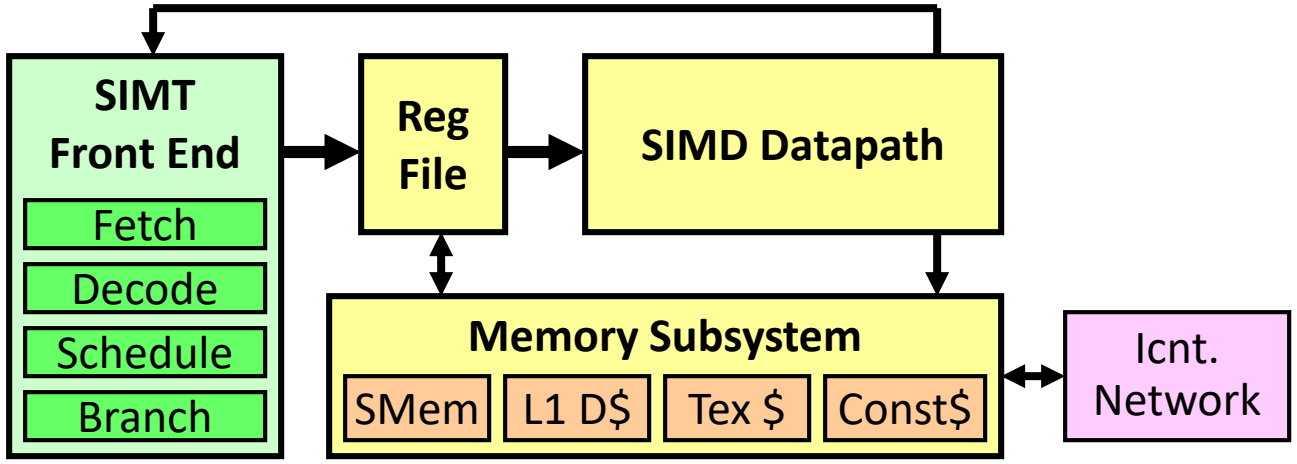


Correlation
~0.976

GPU Microarchitecture Overview

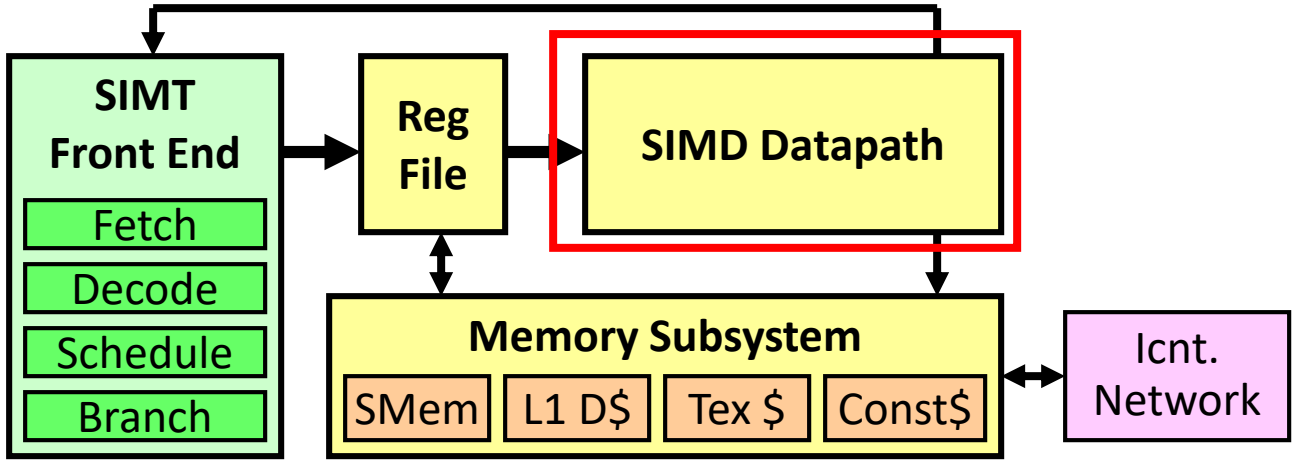


Inside a SIMT Core



- SIMT front end / SIMD backend
- Fine-grained multithreading
  - Interleave warp execution to hide latency
  - Register values of all threads stays in core

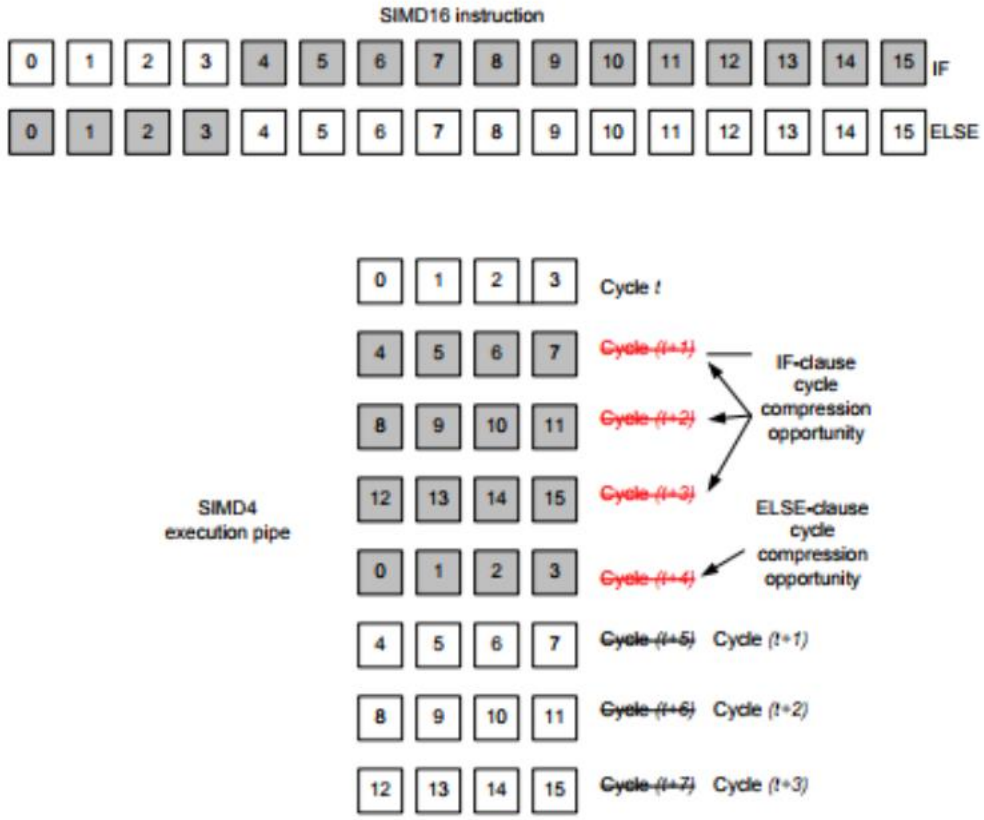
SIMD width and warp size



- The SIMD width may be smaller than the warp size
- If the SIMD is smaller, it can be run at a higher clock (potentially deeper pipeline).
  - E.g. Fermi was SIMD 16, Volta is SIMD 32.
- Research has exploited this difference in SIMD width and warp size to help mitigate control flow divergence

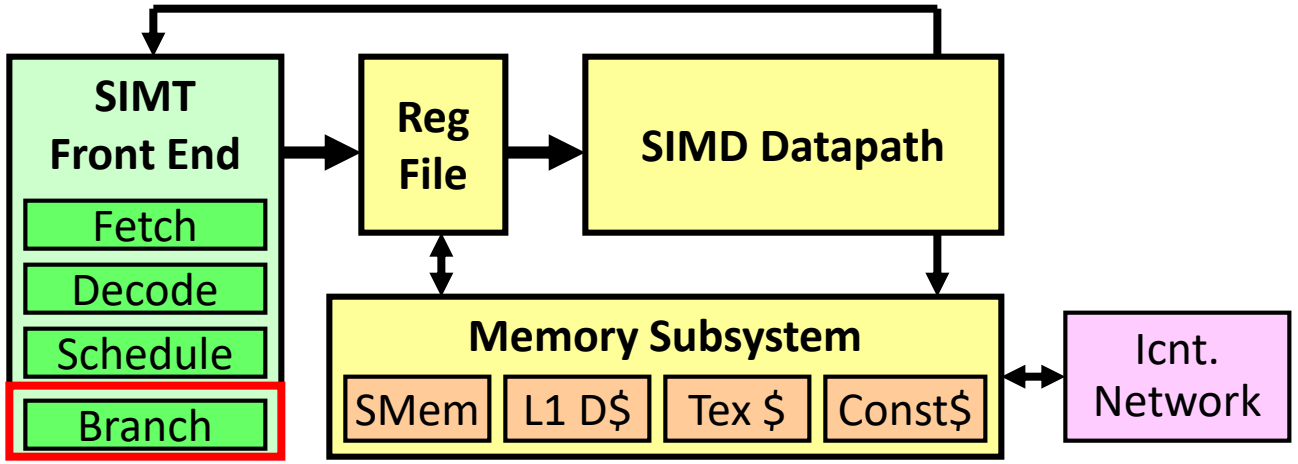
Research Aside: Exploiting SIMD width to mitigate CF divergence

- SIMD Divergence Optimization through Intra-Warp Compaction [ISCA 2013]
- Key Idea: Skip cycles where the whole SIMD is masked off



A ”One loop” Approximation

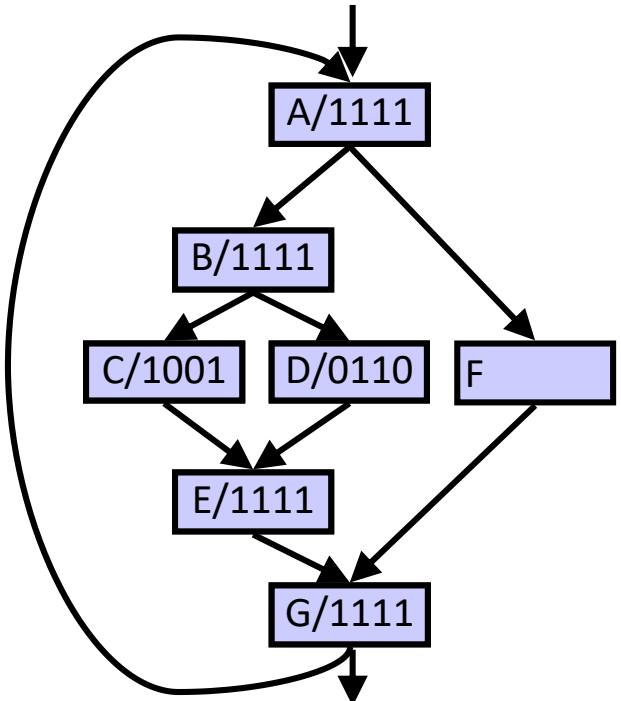
- Simple model for the GPU.
- In the beginning, all warps are eligible for fetch
- One warp is selected, enters the pipe.
  - While that warp is processing, another warp is selected and enters pipe
  - Warps do not become eligible for fetch again until current instruction completes. Only one instruction/warp in pipeline



SIMT Using a Hardware Stack

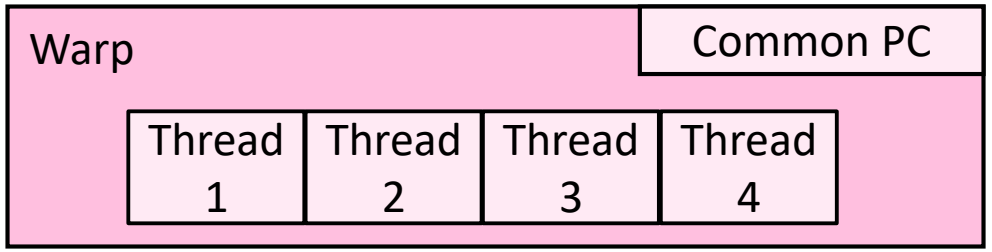
Stack approach invented in early 1980's

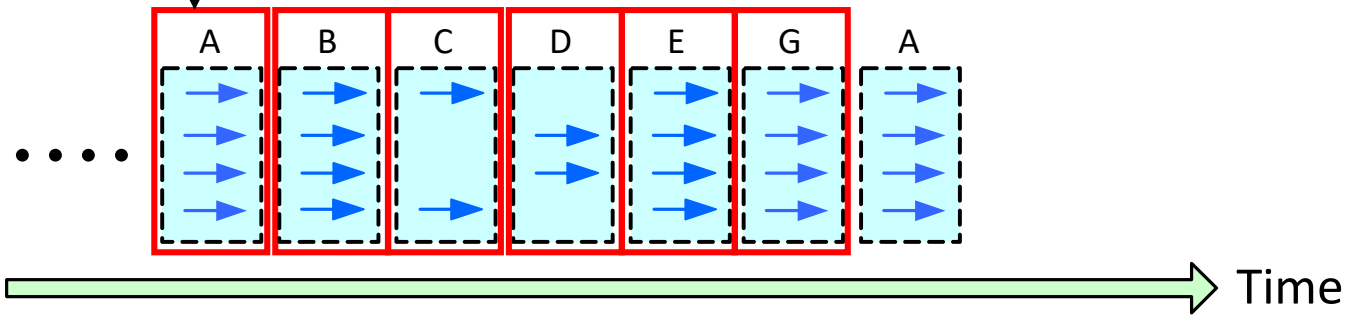
Version here from [Fung et al., MICRO 2007]



Stack

	Reconv. PC	Next PC	Active Mask
TOS →	-	E	1111
TOS →	E	D	0110
TOS →	E	E	1001





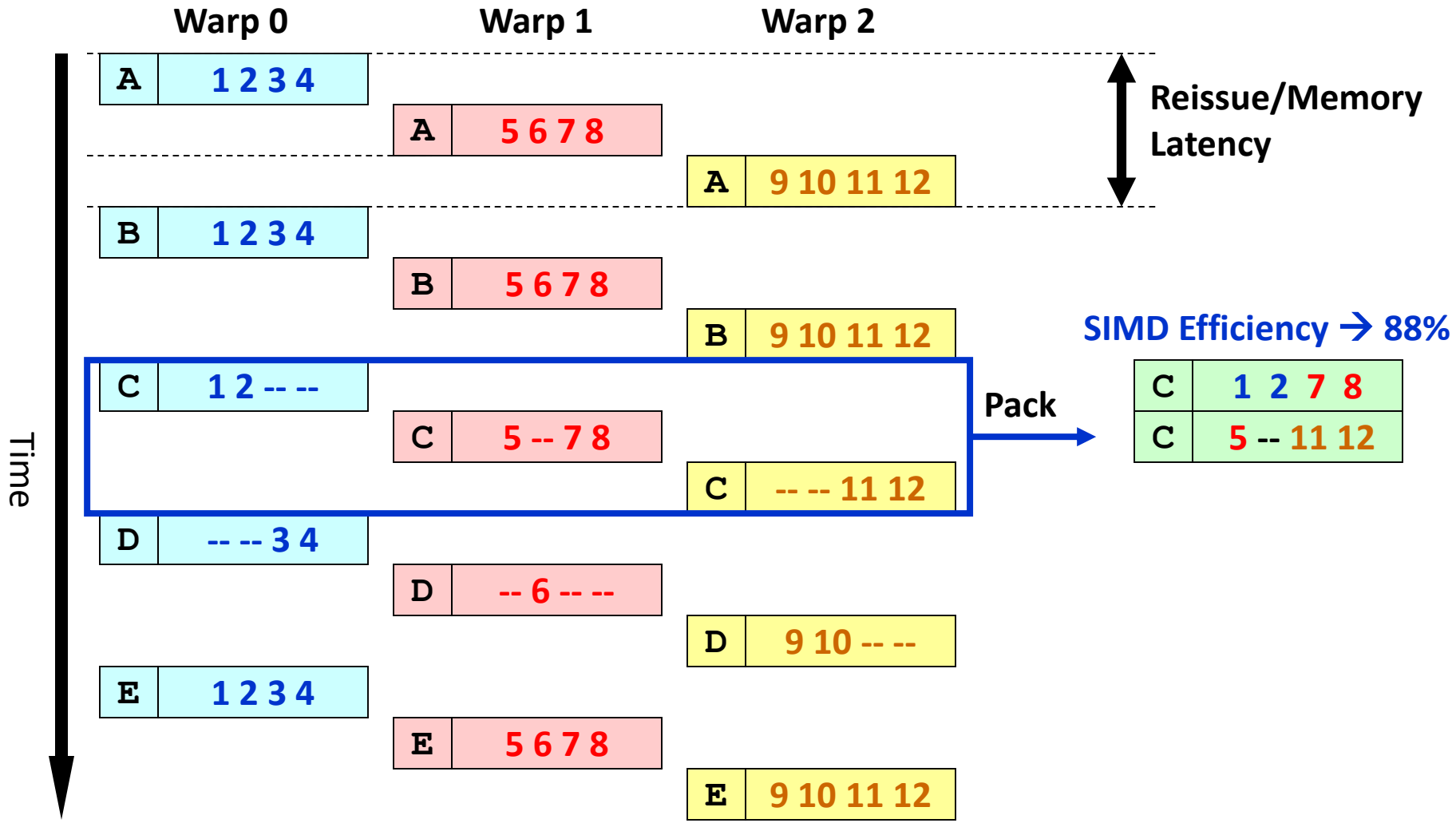
SIMT = SIMD Execution of Scalar Threads

SIMT Notes

- Execution mask stack implemented with special instructions to push/pop. Descriptions can be found in AMD ISA manual and NVIDIA patents.
- In practice augment stack with predication (lower overhead).

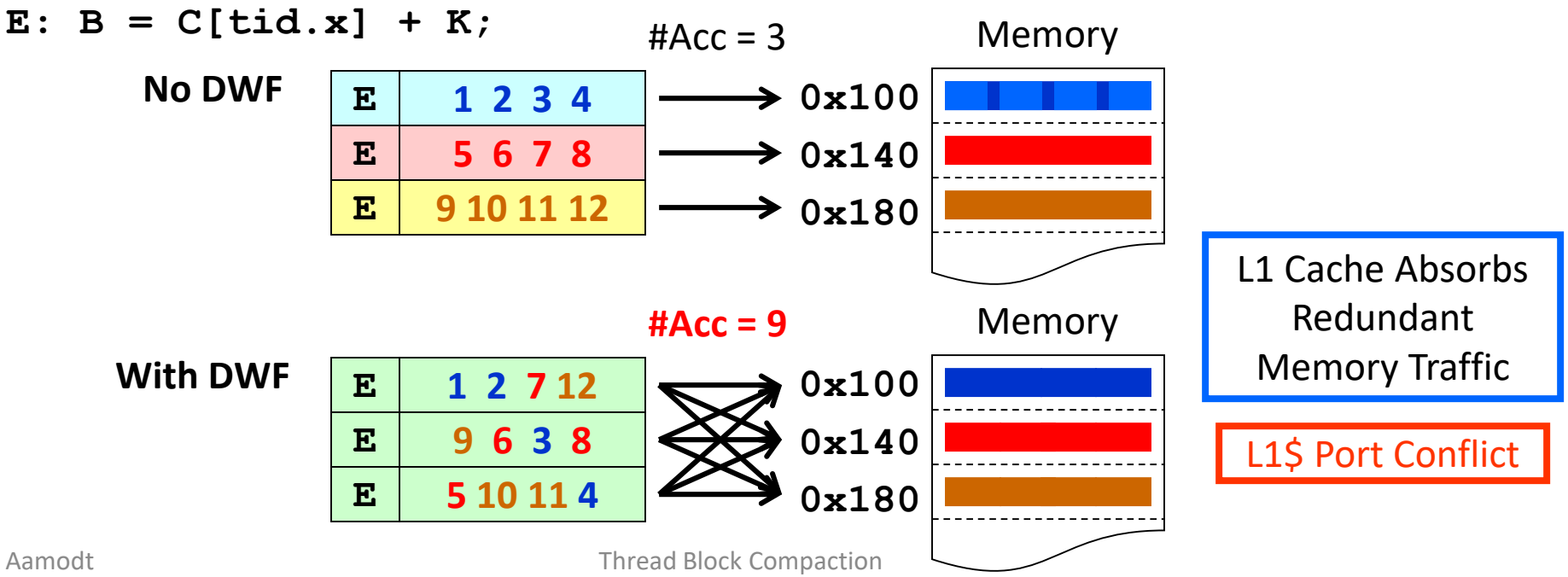
Dynamic Warp Formation

(Fung MICRO'07)



DWF Pathologies: Extra Uncoalesced Accesses

- Coalesced Memory Access = Memory SIMD
  - 1st Order CUDA Programmer Optimization
- Not preserved by DWF



DWF Pathologies: Implicit Warp Sync.

- Some CUDA applications depend on the lockstep execution of “static warps”

Warp 0	Thread 0 ... 31
Warp 1	Thread 32 ... 63
Warp 2	Thread 64 ... 95

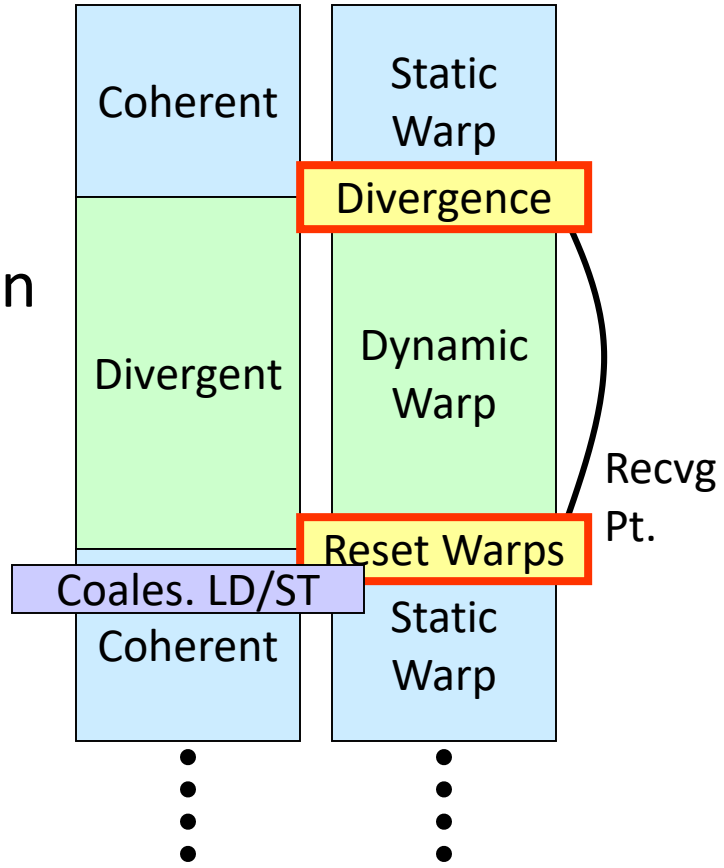
  - E.g. Task Queue in Ray Tracing

```
int wid = tid.x / 32;
if (tid.x % 32 == 0) {
  sharedTaskID[wid] = atomicAdd(g_TaskID, 32);
}
my_TaskID = sharedTaskID[wid] + tid.x % 32;
ProcessTask(my_TaskID);
```

Implicit Warp Sync.

Observation

- Compute kernels usually contain divergent and non-divergent (coherent) code segments
- Coalesced memory access usually in coherent code segments
  - DWF no benefit there



Thread Block Compaction

- Run a thread block like a warp
  - Whole block move between coherent/divergent code
  - Block-wide stack to track exec. paths reconvg.
- Barrier @ Branch/reconverge pt.
  - All avail. threads arrive at branch
  - Insensitive to warp scheduling
- Warp compaction
  - Regrouping with all avail. threads
  - If no divergence, gives static warp arrangement

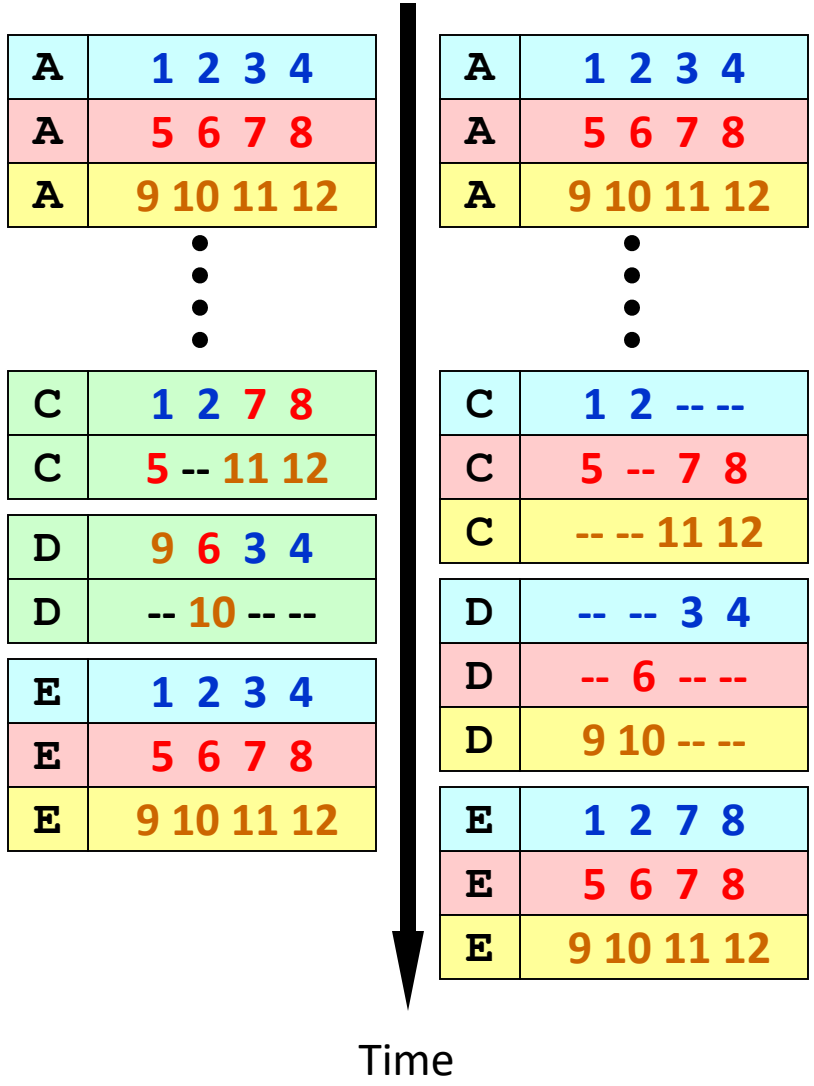
✓ **Implicit Warp Sync.**

~~**Extra Uncoalesced Memory Access**~~

Thread Block Compaction

PC	RPC	Active Threads											
E	-	1	2	3	4	5	6	7	8	9	10	11	12
D	E	--	--	--	--	--	--	--	--	--	--	--	--
C	E	--	--	--	--	--	--	--	--	--	--	--	--

```
A: K = A[tid.x];
B: if (K > 10)
C:     K = 10;
   else
D:     K = 0;
E: B = C[tid.x] + K;
```



Recent work on warp divergence

- Intel [MICRO 2011]: Thread Frontiers – early reconvergence for unstructured control flow.
- UT-Austin/NVIDIA [MICRO 2011]: Large Warps – similar to TBC except decouple size of thread stack from thread block size.
- NVIDIA [ISCA 2012]: Simultaneous branch and warp interweaving. Enable SIMD to execute two paths at once.
- NVIDIA: Temporal SIMT [described briefly in IEEE Micro article and in more detail in CGO 2013 paper]
- UT Austin [HPCA 2013]: The dual-path execution model for efficient GPU control flow
- NVIDIA [ISCA 2015]: Variable Warp-Size Architecture – merge small warps (4 threads) into “gangs”.

SIMT outside of GPUs?

- ARM Research looking at SIMT-ized ARM ISA.
- Intel MIC implements SIMT on top of vector hardware via compiler (ISPC)
- Possibly other industry players in future