Parallel Computing with GPUs

Warp Level CUDA and Atomics Part 2 – Advanced Divergence

The University Of Sheffield.

Dr Paul Richmond

http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- Advanced Divergence
  - Compare divergence with predication
  - Present changes to instruction operation in Volta+ hardware
  - Explain the impact of Volta hardware changes on scheduling

Branching vs. Predication

- Predication is an optional guard that can be applied to machine instructions
  - A predicate is set in predicate registers (virtual registers)
  - Predicates are unique to each thread
- Depending on the predicate value the instruction can be conditionally executed
  - `NOP` otherwise
- How does this differ to branching?
  - No labels or change in program counter
  - Smaller more compact code
    - Less operations = better performance

Branching code

CUDA C

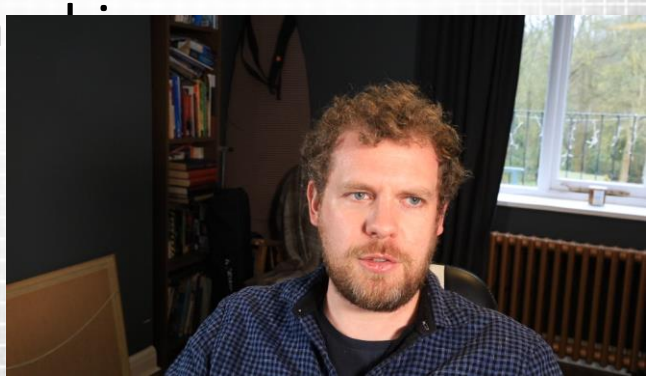
```
int a = 0;

if (i < n)
  a = 1;
else
  a = 2;
```

PTX ISA

```
        mov.s32 a, 0;           //a=0
        setp.lt.s32 p, i, n;    //p=(i<n)
@!p     bra A_FALSE;
A_TRUE:                         //if true
        mov.s32 a, 1;           //a=1
        bra A_END;
A_FALSE:                        //if false
        mov.s32 a, 2;           //a=2
A_END:
        ...
```

- Consider the following branching code…
- Code is PTX ISA
  - A low-level parallel thread execution virtual machine and instruction set architecture (ISA) for CUDA
  - Independent of NVIDIA GPU architecture
  - Used to generate native target architecture m… instructions



Branching code using predicate

CUDA C

```
int a = 0;

if (i < n)
  a = 1;
else
  a = 2;
```

PTX ISA (compiler optimised)

```
        mov.s32 a, 0;           //a=0
        setp.lt.s32 p, i, n;    //p=(i<n)
@p      mov.s32 a, 1;           //a=1
@!p     mov.s32 a, 2;           //a=2
```

CUDA C (improved)

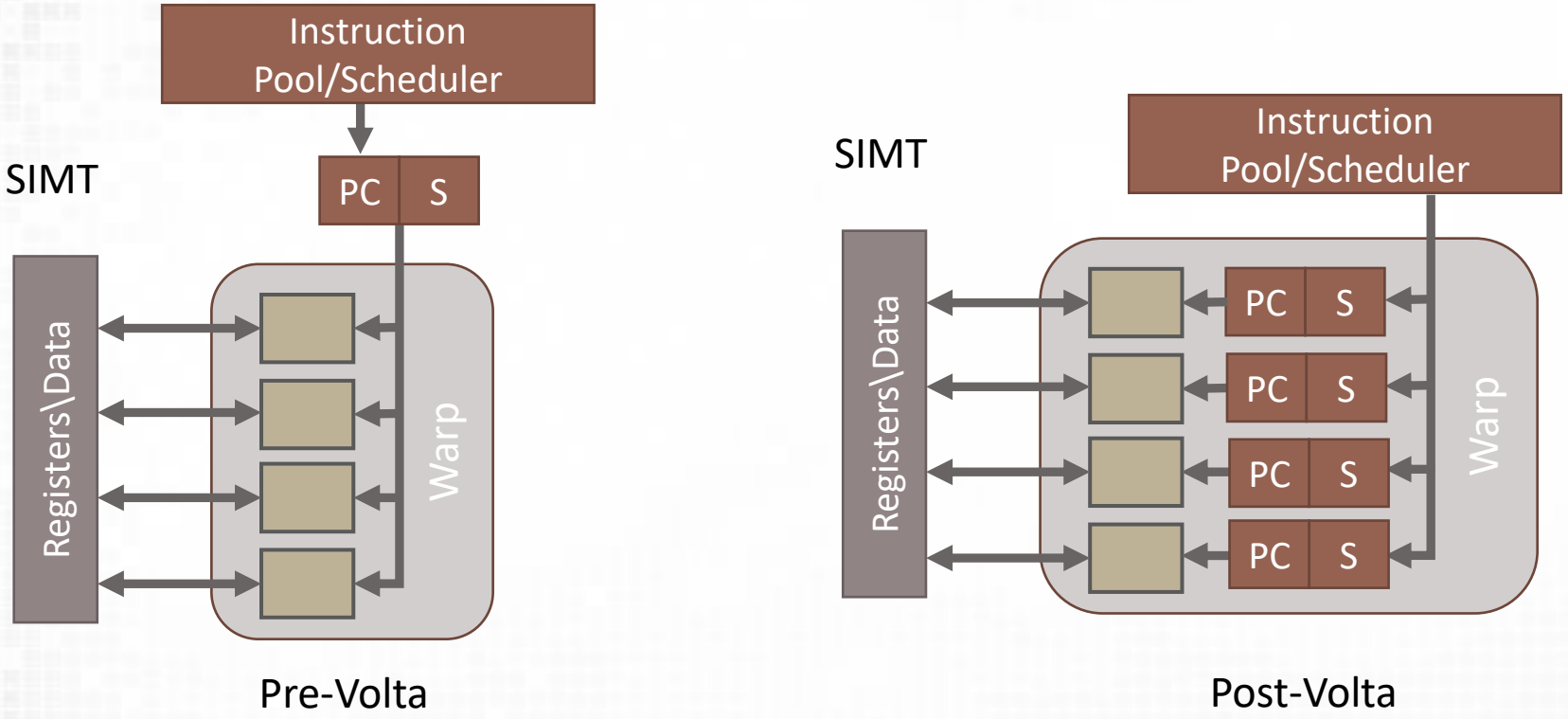
```
int a = 0;
a = (i < n)? 1: 2;
```

PTX ISA (improved)

```
        mov.s32 a, 0;           //a=0
        setp.lt.s32 p, i, n;    //p=(i<n)
        selp a, 1, 2, p         //a=(p)?1:2
```

- Consider the following branching code…
- In this case the predicate can be used to reduce the number of instructions
- The compiler is good at balancing branching and predication
- Can hint to the compiler by using ternary op

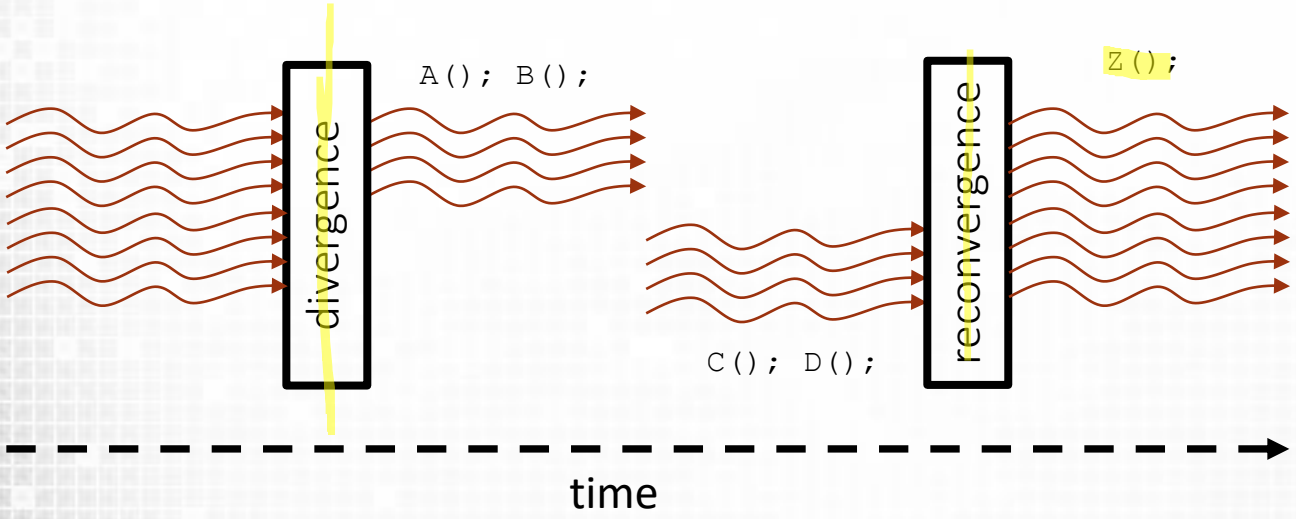
Volta+ hardware changes



- Volta hardware gives each thread its own Program Counter Counter (S)
- Warps still only able to **execute** a single instruction at one t

Pre-Volta Execution

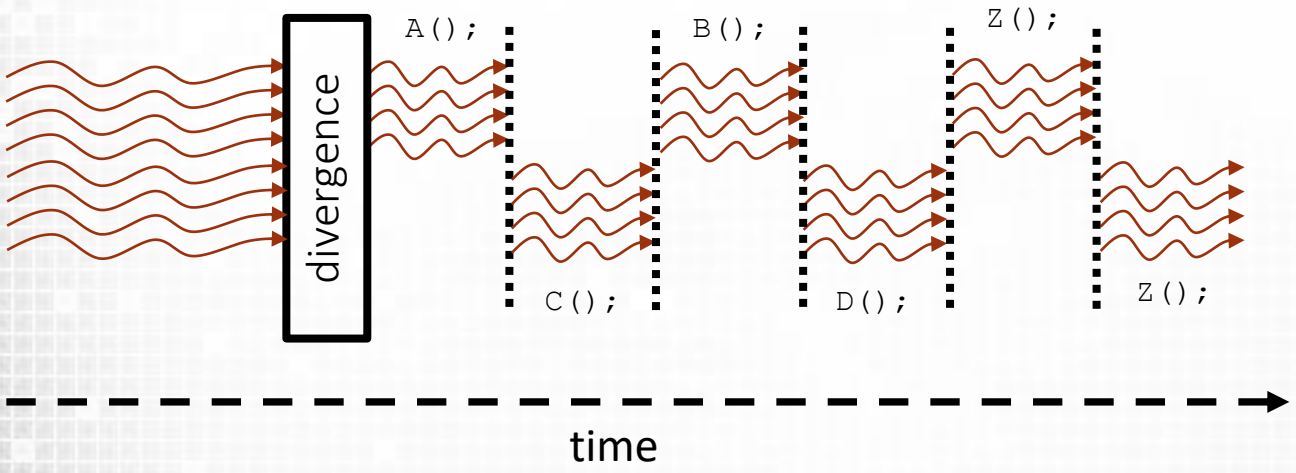
```
if (threadIdx.x < 4){
  A();
  B();
}else{
  C();
  D();
}
Z();
```



- Divergent threads only reconverge after branch evaluation
- Latency in A or B must be masked by *other* warps

Volta Execution

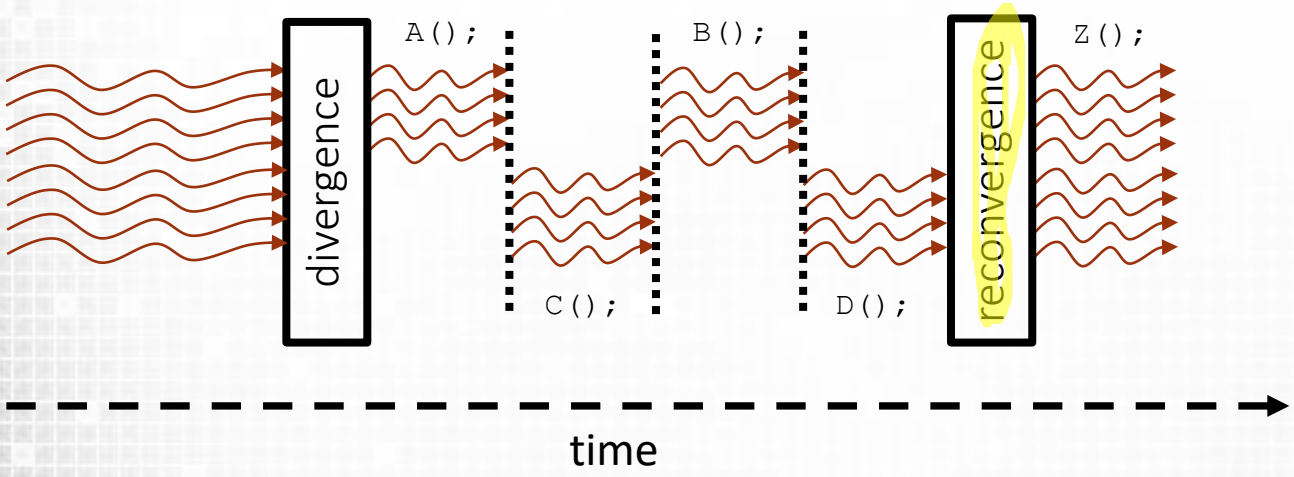
```
if (threadIdx.x < 4){
  A();
  B();
}else{
  C();
  D();
}
Z();
```



- Scheduler can yield execution to any thread at sub warp level
- Latency in functions can be masked by switching to active threads in other parts of the warp.

Warp Synchronisation

```
if (threadIdx.x < 4){
  A();
  B();
}else{
  C();
  D();
}
__syncwarp();
Z();
```



- The compiler will converge warps automatically
- Warps can be forced to reconverge
- Helpful to ensure greater effeciancy

Summary

- Advanced Divergence
  - Compare divergence with predication
  - Present changes to instruction operation in Volta+ hardware
  - Explain the impact of Volta hardware changes on scheduling

- Next Lecture: Atomics and Warp Operations