Australian National University

Warp Synchronization

Dr Eric McCreath

Research School of Computer Science

The Australian National University

Introduction

Since Cuda 9 you have been able to synchronize across blocks within a grid. This is done via "Cooperative Groups". These cooperative groups also cover synchronization at the block and warp level. This inter thread warp communication and synchronization is supported by warp-level primitives. These include:

```
__all_sync - checks if a predicate is true for all the threads in the warp's mask
__any_sync - checks if a predicate is true for any of the threads in the warp
__ballot_sync - evaluates the predicate for all participating threads creating of b
__shfl_sync - this enables exchange of information between threads without using sh
__shfl_up_sync, __shfl_down_sync - copy values from other lanes in the warp
__shfl_xor_sync - also copies values between lanes (butterfly pattern)
__match_any_sync - returns a mask of all the threads with the same "value"
__match_all_sync - returns mask if all threads have the same value, also sets a pre
```

__ballot_sync

The __ballot_sync enables a thread to state a predicate's value (true/false), and in this one instruction see the predicate value of all the other threads.

```
unsigned __ballot_sync(unsigned mask, int predicate);
```

Say you had any array of ints called data={10, 20, 5, 4, 17, 21, 2, 20 ...} and executed :

```
int lane = threadIdx.x % 32;
int bitPattern = __ballot_sync(0xFFFFFFFF,data[lane]<10);
```

After this all the threads in the warp would have:

```
bitPattern == 00110010....
```

This provides a very quick way of all threads in the wrap knowing the "involvement" in an activity of all the other threads. This can be used for creating masks for other warp level operations.

__shfl_sync

The __shfl_sync provides a way of moving a value from one thread to other threads in the warp in one instruction.

```
T __shfl_sync(unsigned mask, T var, int srcLane);
```

Say you executed:

```
value =  __shfl_sync(0xFFFFFFFF, value, 7);
```

this would broadcast the "value" that the thread in lane 7 has to all the other threads in the warp.

__shfl_down_sync

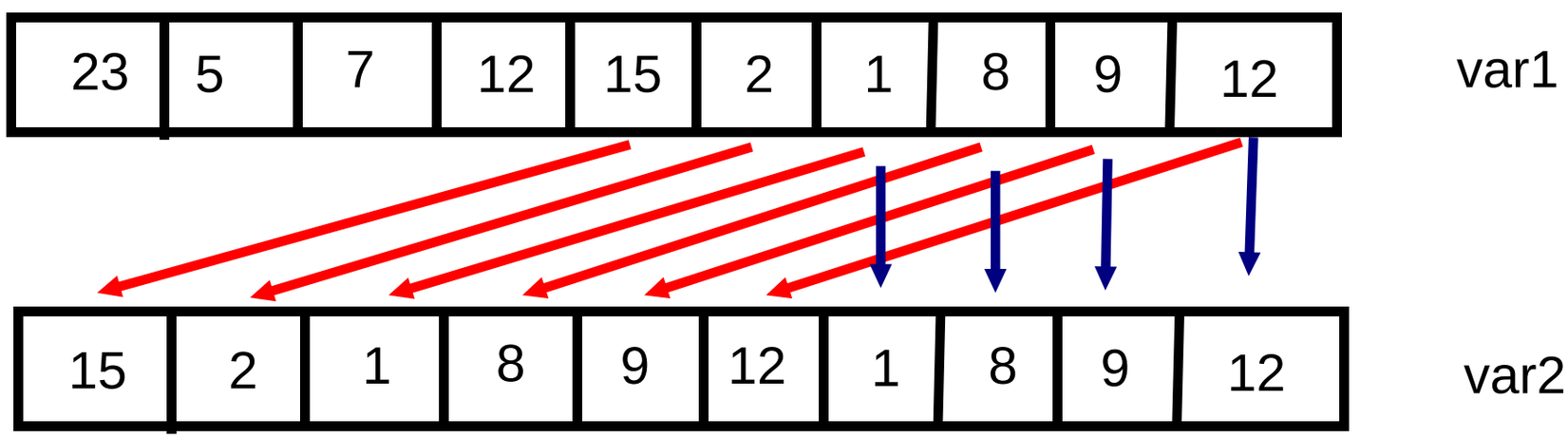
The shuffle wrap operations enable you so shift data within a warp in one instruction. So basically the "var" is shiffted down "delta" lanes.

```
T __shfl_down_sync(unsigned mask, T var, unsigned int delta);
```

If say we executed:

```
int var2 = __shfl_down_sync(0xFFFFFFFF, var1, 4);
```

We would have (my diagram just has 10 elements although in reality there is 32 in a warp):



Reduction

The below code does a tree based reduction within the warp. This will be a lot lot faster than using __syncthreads() and shared memory within a block.

```
unsigned mask = __ballot_sync(FULL_MASK, threadIdx.x < NUM_ELEMENTS);
if (threadIdx.x < NUM_ELEMENTS) {
    val = input[threadIdx.x];
    for (int offset = 16; offset > 0; offset /= 2)
        val += __shfl_down_sync(mask, val, offset);
    ...
}
```

The above code is from "Using CUDA Warp-Level Primitives"

By Yuan Lin and Vinod Grover,
https://devblogs.nvidia.com/using-cuda-warp-level-primitives/

References

- Timcheck, Stephen W., "Efficient Implementation of Reductions on GPU Architectures" (2017). Honors Research Projects. . http://ideaexchange.uakron.edu/honors_research_projects/479
- Mark Harris, Optimizing Parallel Reduction in CUDA

https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf
- Mark Harris and Kyrylo Perelygin , Cooperative Groups: Flexible CUDA Thread Programming, 2017 https://devblogs.nvidia.com/cooperative-groups/
- Cuda c Programming Guide

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html
- Lecture 4: warp shuffles,and reduction / scan operationsProf. Mike Giles, https://people.maths.ox.ac.uk/gilesm/cuda/lecs/lec4.pdf