Efficient Low-Latency Real-Time Convolution for Multi Core

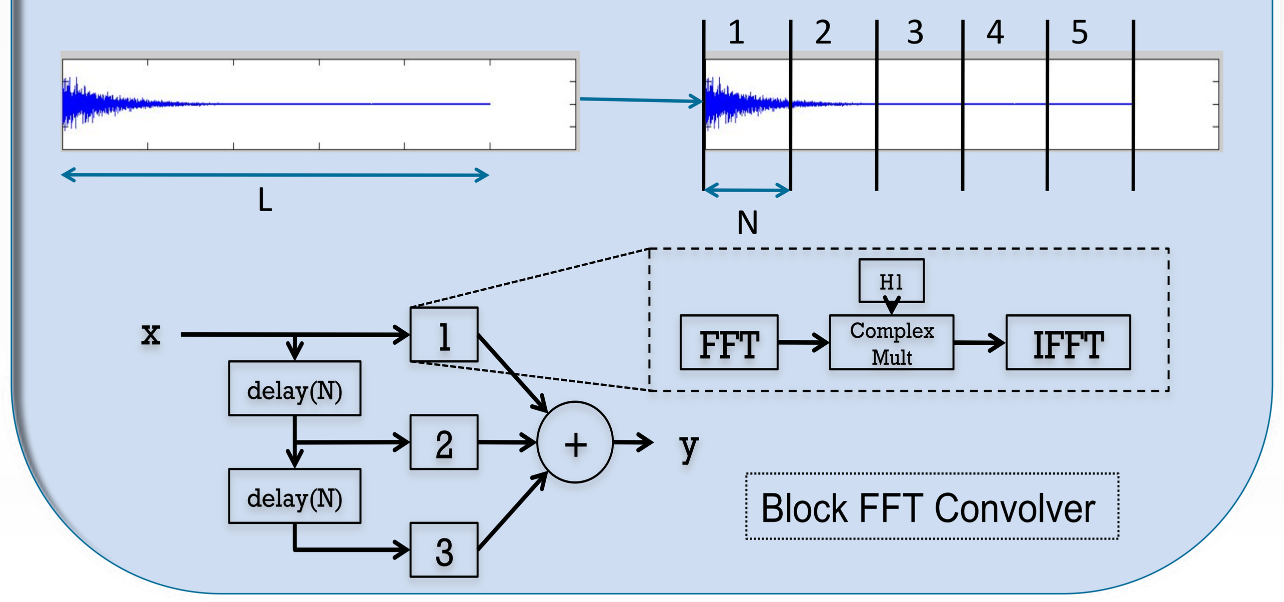
Eric Battenberg, David Wessel
ericb@eecs.berkeley.edu

The Application

- ***First real-time app in the Par Lab.***
- Partitioned Convolution – an efficient way to do low-latency audio filtering with a long impulse response.
- Used in convolution reverb for environment simulation, creative effect processing, and electronic instrument creation.

Partitioned Convolution

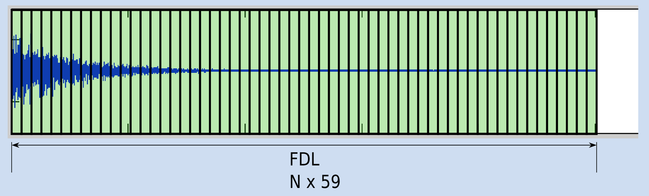
- Direct convolution is expensive.
- Block-FFT convolution has higher latency.
- Partitioning a filter into smaller Block-FFT filters allows us to reduce latency while preserving efficiency.



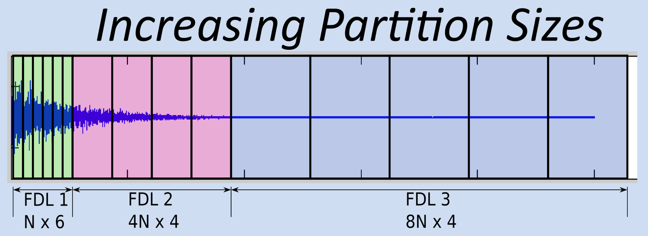
Multiple Partition Sizes

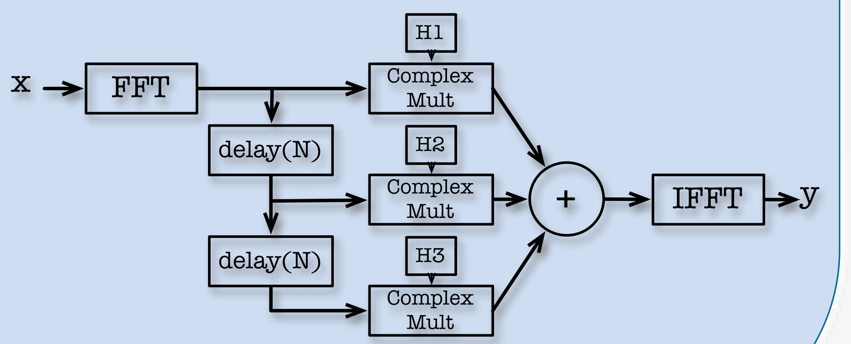
- To increase efficiency, we can increase the partition size as we progress through the filter.
- We can reuse FFT's amongst same-size partitions in a Frequency Delay Line (FDL)
- *So, what is the most efficient combination of FDL's for a particular filter length and latency?*

Single Partition Size



Increasing Partition Sizes





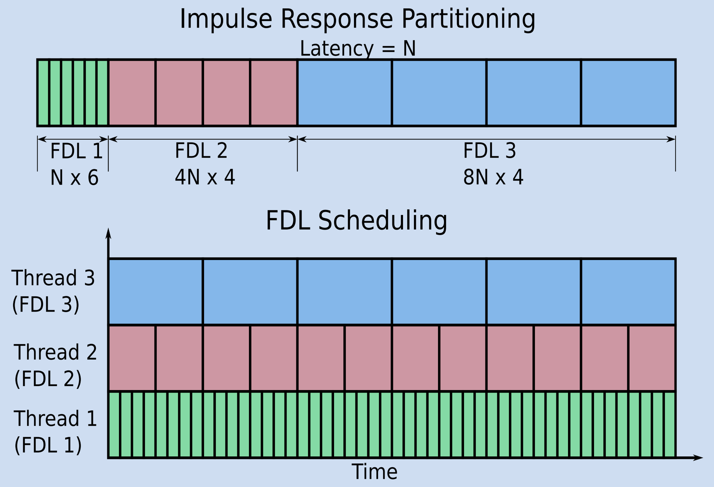
Frequency Delay Line

Auto-Tuning for Real-Time

- Each FDL gets its own thread which is preemptive with fixed-priority.
- Longer partition sizes are allowed a longer compute time to preserve uniform processor loading.
- We tune for Worst-Case Processor Load (WCPL) using an algorithm based on dynamic programming.

Processor Loads for 661500 sample (15sec) filter with latency of 64 (1.5ms)

Scheme	Partitioning (block-size x number of blocks)	Load
Uniform	64x10336	356%
Gardner (fastest FDL growth)	64x2, 128x2, 512x2, 1024x2, 2048x2, 4096x2, 8192x2, 16384x2, 32768x2, 65536x2, 131072x2	3.54%
Min WCPL	**64x2, 128x6, 512x6, 2048x6, 8192x6, 32768x19**	**2.55%**



Example scheduling of each FDL work unit

Optimizations

- SSE3 instructions for complex mult-adds.
- Synchronization between FDL's is done using Condition Variables (CV) and Atomic ops.
- To reduce system calls, synchronization is organized so that only a single CV is used in each callback to signal all worker threads.

Conclusions/Future Work

- **Tuning and optimizations allow us to process <u>100+ channels of audio</u> on current multi-core machines.**
- The multi-rate structure of a non-uniform partitioning provides an interesting scheduling problem when interfacing with our cooperative audio graph host.
- An explicitly cooperative implementation could provide more reliable performance at the expense of programmer effort.