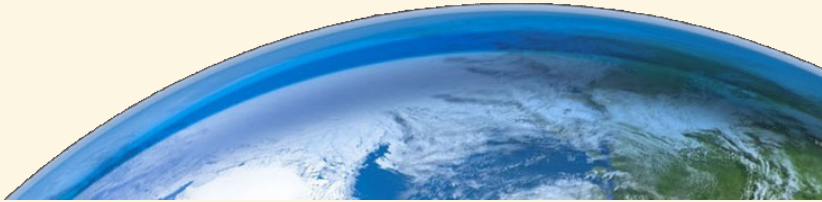
Coprocessor **A**ccelerated **F**ilterbank Extension Library

Mummy, are we there yet

Jan Krämer

DLR Institute of Communication and Navigation (IKN)

04.02.2018



German Aerospace Center

Who am I?

Jan Krämer
Software Defined Radio Imposter at German Aerospace Centre Oberpfaffenhofen
General interest in making stuff a bit faster

I fought my own officemate for rights to that name...

CAFE is the **C**oprocessor **A**ccelerated **F**ilterbank **E**xtensions Library
Realtime Polyphase Filterbank Channelizer (PFB-C)

- 45 channels
- 1550 tap filter
- 4 MSamples/s needed

Optimized CPU Version: 1-2 MSamples/s

Regular ordinary frametitle, no memes here

GPGPU TO THE RESCUE!!!

Yo check me out, I'm awesome

- Channelizer presented already last year[1]
- Oversamples the output to all factors that are integer divisions of the channel number (e.g. 3x oversampled = 45 channels/15)
- Able to achieve 110 MSamples/s (45 Channels, 1550 tap protoype filter)
- Now does CuFFT output reshuffle → additional performance gains are expected

Who wrote those specs...

- Timing sync needs 4x oversampling factor
- PFB-C gets to 4.2666x oversampling factor
- Arbitrary resampler needed

Bloody Resamplers, how do they work?

- Use PFB to "upsample" the signal
- Downsample by skipping the right filters in the bank
- Filter the signal with normal filter and a differential filter in parallel
- Interpolate between the 2 outcomes of the filter
- Profit

I wish I had a mouse to draw this...

Start with normal vector of taps

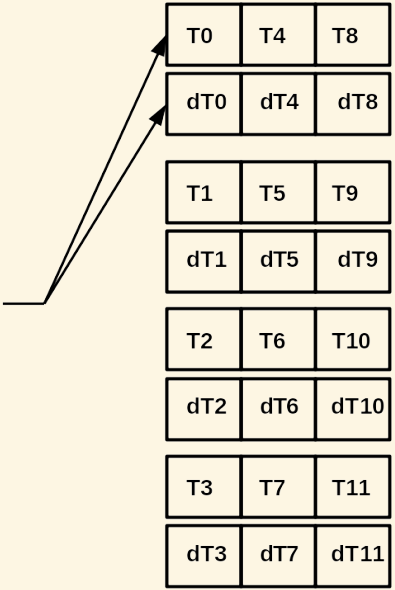
Halp...this is LibreOffice Draw

Add the differential tap vector

$$diff\_tap[i] = tap[i+1] - tap[i] \tag{1}$$

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
T1-T0	T2-T1	T3-T2	T4-T3	T5-T4	T6-T5	T7-T6	T8-T7	T9-T8	T10-T9	T11-T10	0

Usual partitioning is applied...Oh god I suck at graphics



Breakdown of operations

- interpolation_rate = How much to upsample
- decimation_rate = How much to downsample
- floating_rate = Difference between the integer downsampling and the actual needed downsampling factor
- accumulated_rate = Accumulated difference between the integer filter skips and needed filter skips

Did you notice the last 2 frametitles made sense?

- *interpolation_rate* = number of filter (2)
- *decimation_rate* = *floor*(*interpolation_rate*/*rate*) (3)
- *floating_rate* = (*interpolation_rate*/*rate*) − *decimation_rate* (4)
- accumulated_rate in 2 steps:
  - *accumulated_rate* += floating_rate (5)
  - *accumulated_rate* = *accumulated_rate* % 1.0 (6)

I hope you rembered those equation numbers!

Filterskips and interpolation

- Calculate ouput_normal and output_diff of both filters at filter_index
- $result = output\_normal + accumulated\_rate * output\_diff$ (7) (Interpolation)
- Update accumulated_rate according to [5]
- Update filter_index += decimation_rate + floor(accumulated_rate) (8)
- Update accumulated_rate according to [6]
- Update input = input + filter_index/interpolation_rate (9)

You hear the music, don't you?

One slide, sure...

CUDA in one slide:

- Used to launch operations in massively parallel fashion on the GPU
- Closely related to NVidia GPU architecture
  - Several multiprocessors each with local on-chip memory and cache (fast)
  - Several CUDA Cores/ALUs per multiprocessor
  - Large (but slow) Global memory

Told you it won't work

CUDA in ~~one~~ several slides:

- CUDA divides operations into a grid of blocks
- Maps:
  - *Grid* $\Rightarrow$ *GPU*
  - *Block* $\Rightarrow$ *Multiprocessor*
  - *Thread* $\Rightarrow$ *ALU*
- Threads are scheduled in groups of 32 $\Rightarrow$ Warps
- All Threads in a block can use shared, fast on-chip memory

As it is written in the sacred NVIDIA optimization guide

CUDA rules of thumb

- More threads than your Multiprocessor has ALUs $\Rightarrow$ keeps huge pipeline busy
- On-Chip memory waaaay faster than Global memory
- Loads from both memories are done with a huge cacheline

  $\Rightarrow$ have adjacent threads in a warp use adjacent memory entries $\Rightarrow$ minimizes memory loads

Where have I heard this before...

- Target outputs of the PFB Channelizer ⇒ Maximum use of the available cores
  - One channel mapped to one CUDA block
- Each thread computes one resampler output
- Each thread computes both filter results and interpolation
- Concurrency only through processing of multiple samples ⇒ minimal synchronization needed
- Same division as the PFB Channelizer

Prayers to the floating point god

Filter calculations

- All filter updates calculated on the GPU
- Filter processes all samples in its input
  - Uncertainty in produced outputsamples
  - Precalculate the number of operations on the CPU
  - Transfer expected end filter and number of ops to the GPU before every run
  - Dummy calculations might be done by a Warp $\Rightarrow$ take care of it when copying data back from the GPU

Just imagine a fancy graphic

Results look promising for our use case

- Software runs on Intel i7-6800k with NVidia GTX970 GPU
- Benchmarked the full chain PFB Channelizer + PFB Resampler
- 45 Channels + 1550 taps protoype filter used
- 768 samples per channel processed in parallel
- Result $\Rightarrow$ 25 MSamples/s average throughput

Call me Don Quijote

Harti (awesome colleague) and I battling since september to get it open sourced
Established an open sourcing process at IKN with me as the lab rat

- Check licenses
- Check export control
- Check with project partners and project sponsor/coordinator
- Establish CLA

What an excuse for this subpar presentation

- Still had to convince the institute management
- Several presentations on how open source benefits everyone (DLR and you gals and guys)
- Several written documents basically claiming the same as the presentations
- The whole project (and this talk) was in jeopardy

Finally on monday we got the greenlight
1 hour before I went on vacation...

Thanks Obama

Special thanks to these people at IKN

Gianluigi Liva group leader for the information transmission group at DLR Institute of Communication and Navigation (DLR IKN)
Hartmut "Harti" Brandt lead developer at the satellite communication group at DLR IKN

Thanks Obama

Even more special thanks to

Joni
Gerald

For all the Kung Fury inspiration!!