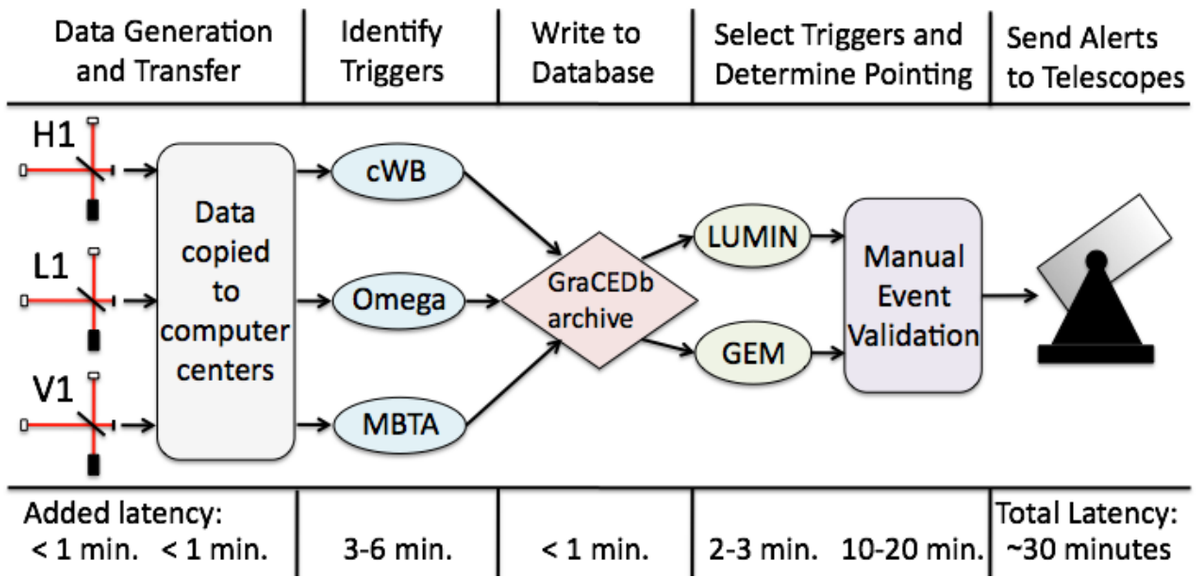
Resampler in OpenCL

Edwin J. Son (on behalf of NIMS)
[in collaboration with UWA]

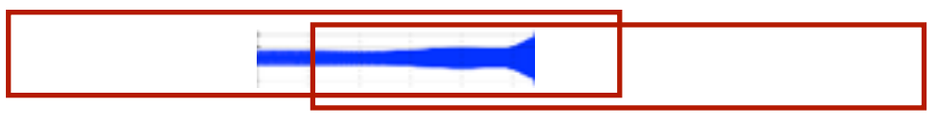
Introduction

Latency of matched-filter pipelines

[Low-latency optical followup]



[Time segments for matched filter]



This time segments set minimum latency

- MBTA (multiband template analysis, VIRGO): over two frequency bands
  < 3 min. of latency

- LLOID (low-latency on-line inspiral data, LIGO): down-sampling & multiple streams

- lower frequency bound will be dropped from 40 to 10 Hz in advanced era

- EM counterpart of a GRB event will be significantly faded

Summed Parallel IIR Filter

- SPIIR method shows a good performance in that it can retrieve SNR to greater than 99% of that produced from the optimal matched filter. [Hooper et al., PRD86, 024012 (2012)]
- GPU accelerated SPIIR code is around 2x faster than CPU SPIIR code.
- Linqing reported that there is a bottleneck at the upsampling process in the Beijing meeting (Jul. 2013).
- We are implementing parallelized resampler on GPU.

[Courtesy: Hooper et al.]

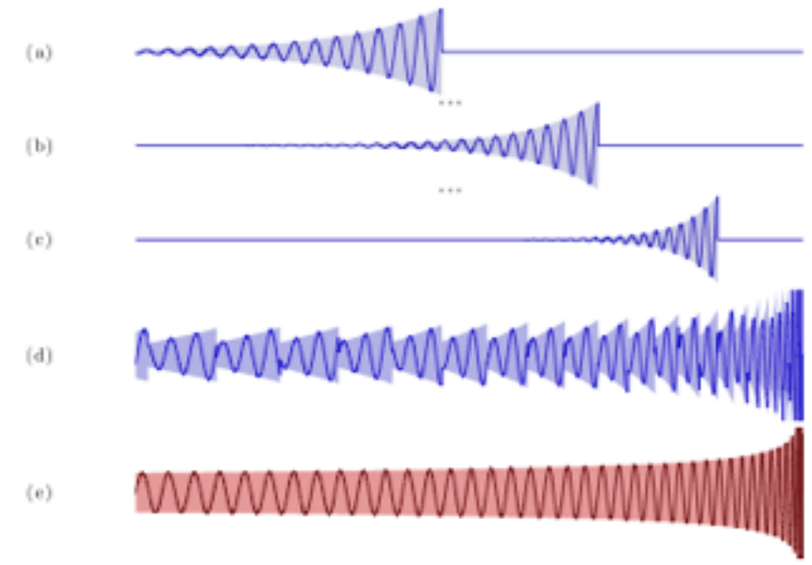


FIG. 3 (color online). An illustrative diagram demonstrating the ability to linearly sum exponentially increasing constant-frequency sinusoids to approximate an inspiral-like waveform. The top three panels (a–c) show three example sinusoids with different damping, frequency and cutoff time factors. Panel (d) shows the linear addition of all the sinusoids (at different scales). Panel (e) shows the exact inspiral-like waveform. Note that this figure is only for illustrative purposes.

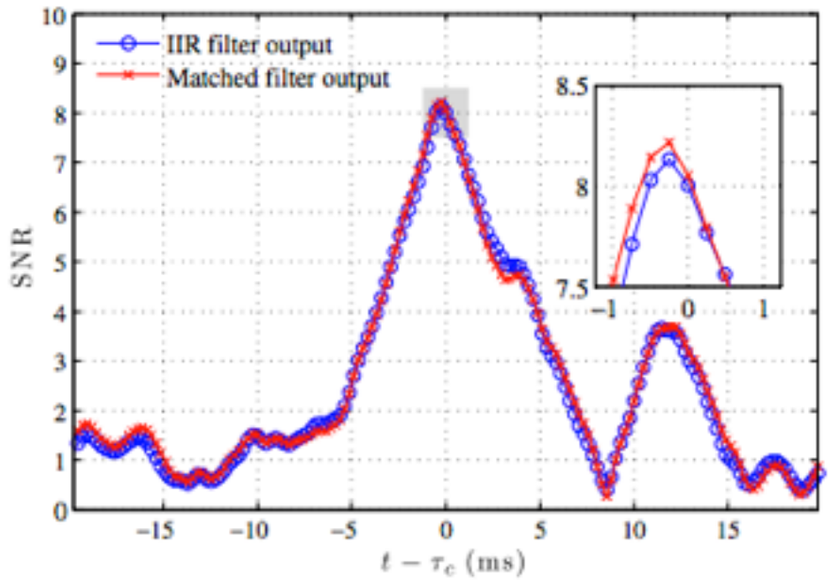
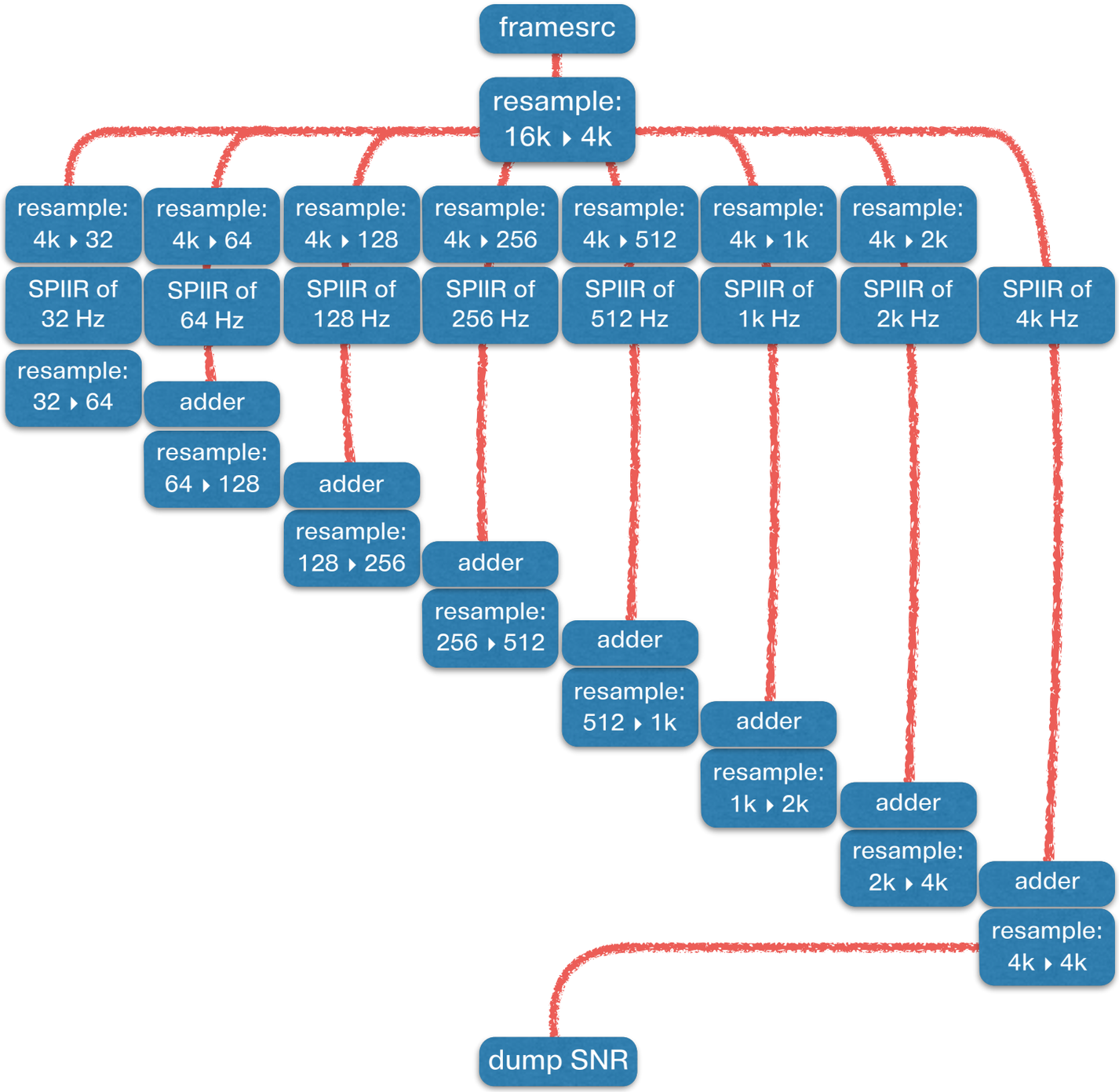
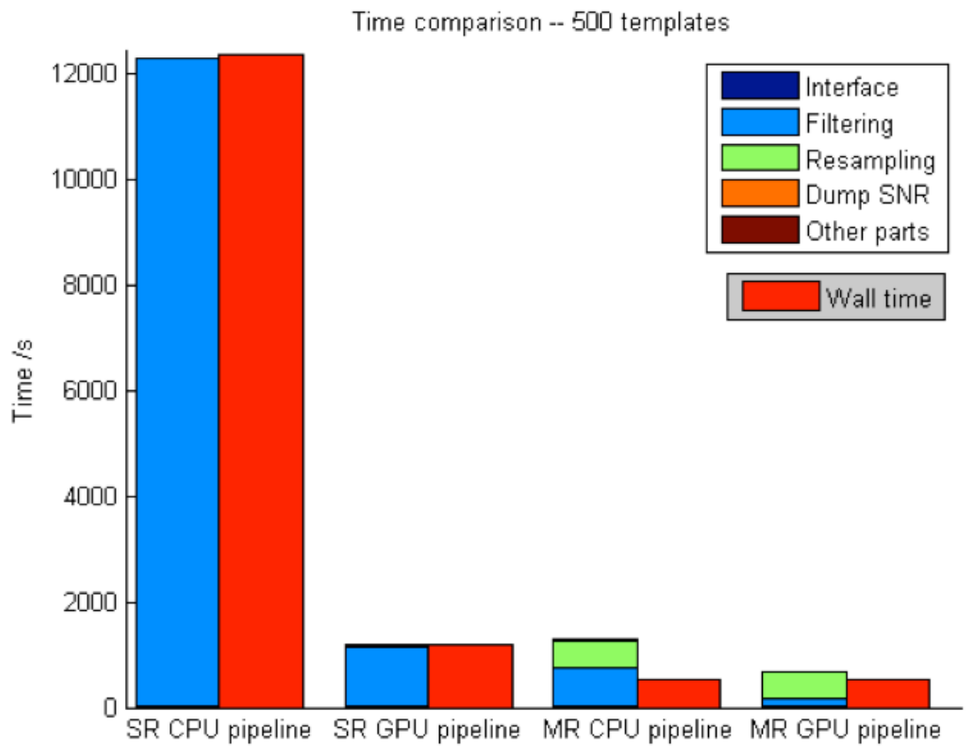


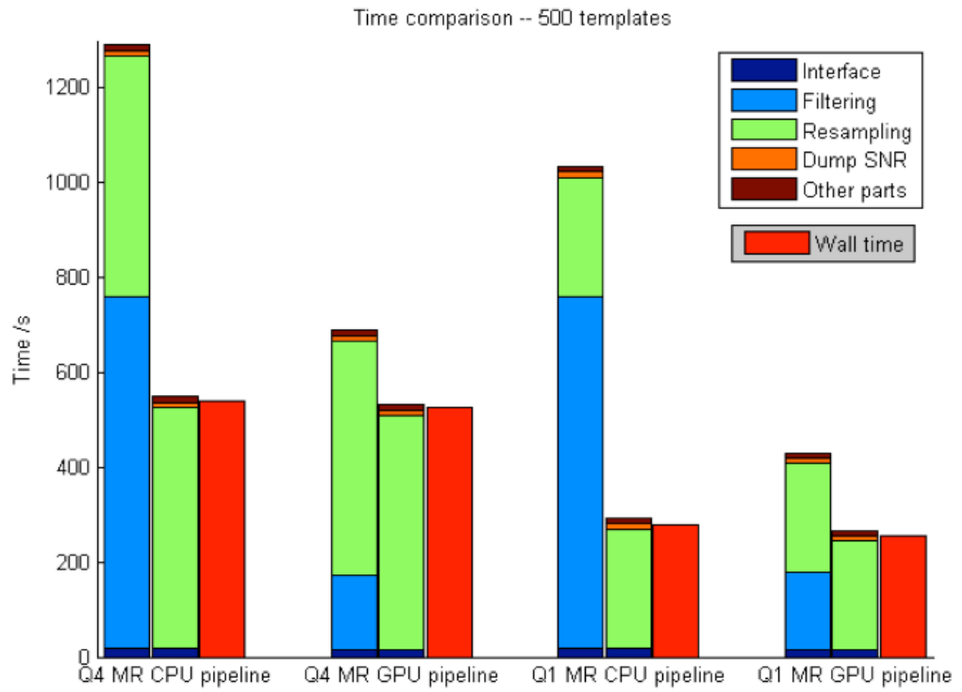
FIG. 5 (color online). The SNR output of both the SPIIR method and a traditional matched filter method. The plot is centered on $t - \tau_c$ where τ_c is the time at which the injection ends. From the two curves, it is clear that the SPIIR method can return a very similar SNR to that from the optimal filter.

SPIIR pipeline



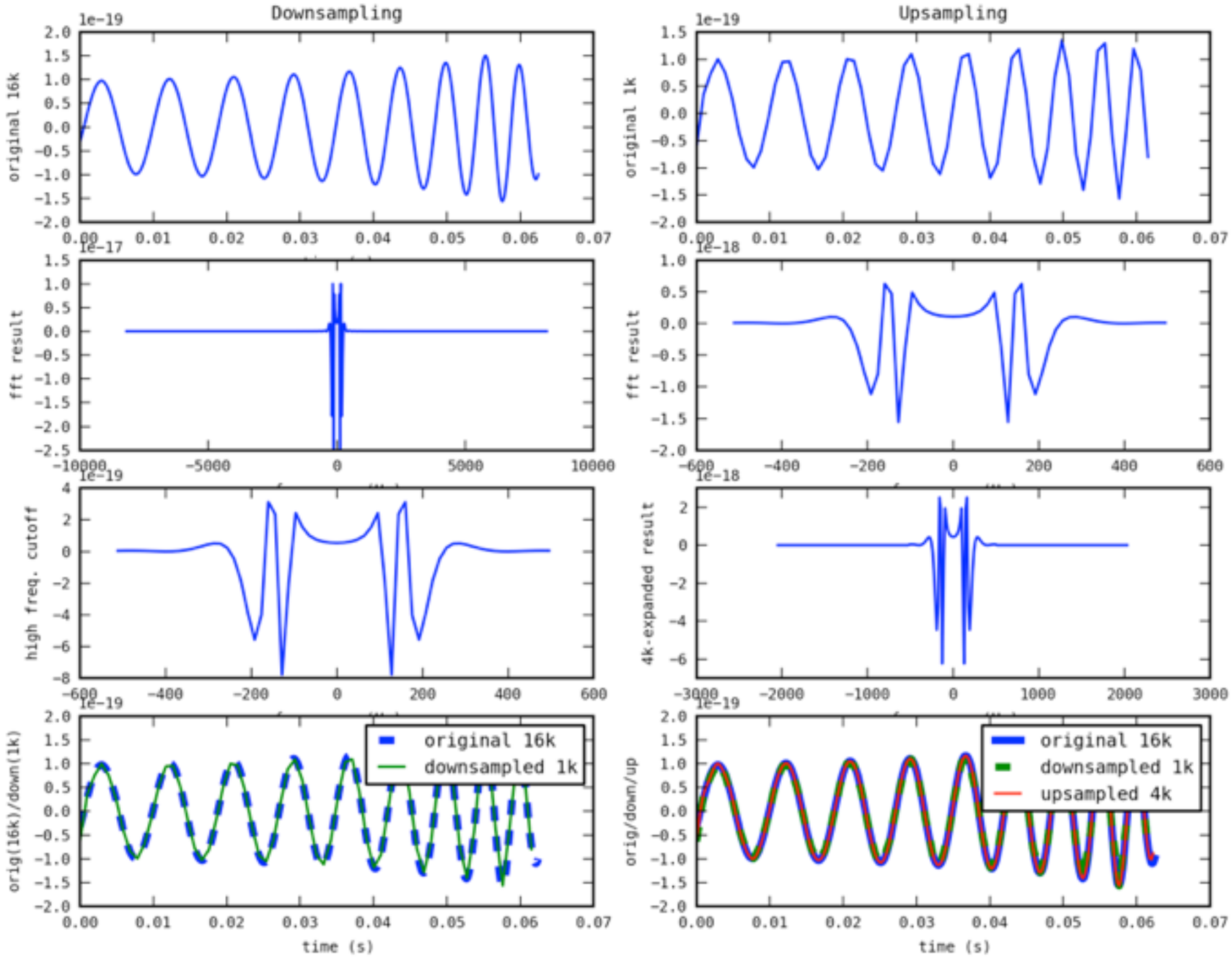
Bottleneck in SPIIR





[Courtesy: Linqing Wen]

Resampling


Downsampling
Upsampling
original 16k
original 1k
fft result
high freq. cutoff
4k-expanded result
orig(16k)/down(1k)
orig/down/up
time (s)
original 16k
downsampled 1k
upsampled 4k

Resample with sinc

- Continuous time signal x(t) can be reproduced from the sampled data x_n up to the Nyquist frequency f_N.
- Using this D-to-C converter, we can implement an resampler which can be used in a streaming pipeline.

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \operatorname{sinc}\left[\pi f_s \left(t - \frac{n}{f_s}\right)\right],$$

$$\text{where } \operatorname{sinc}(z) = \frac{\sin z}{z},$$

$$f_s = 2f_N = \text{sampling rate}$$

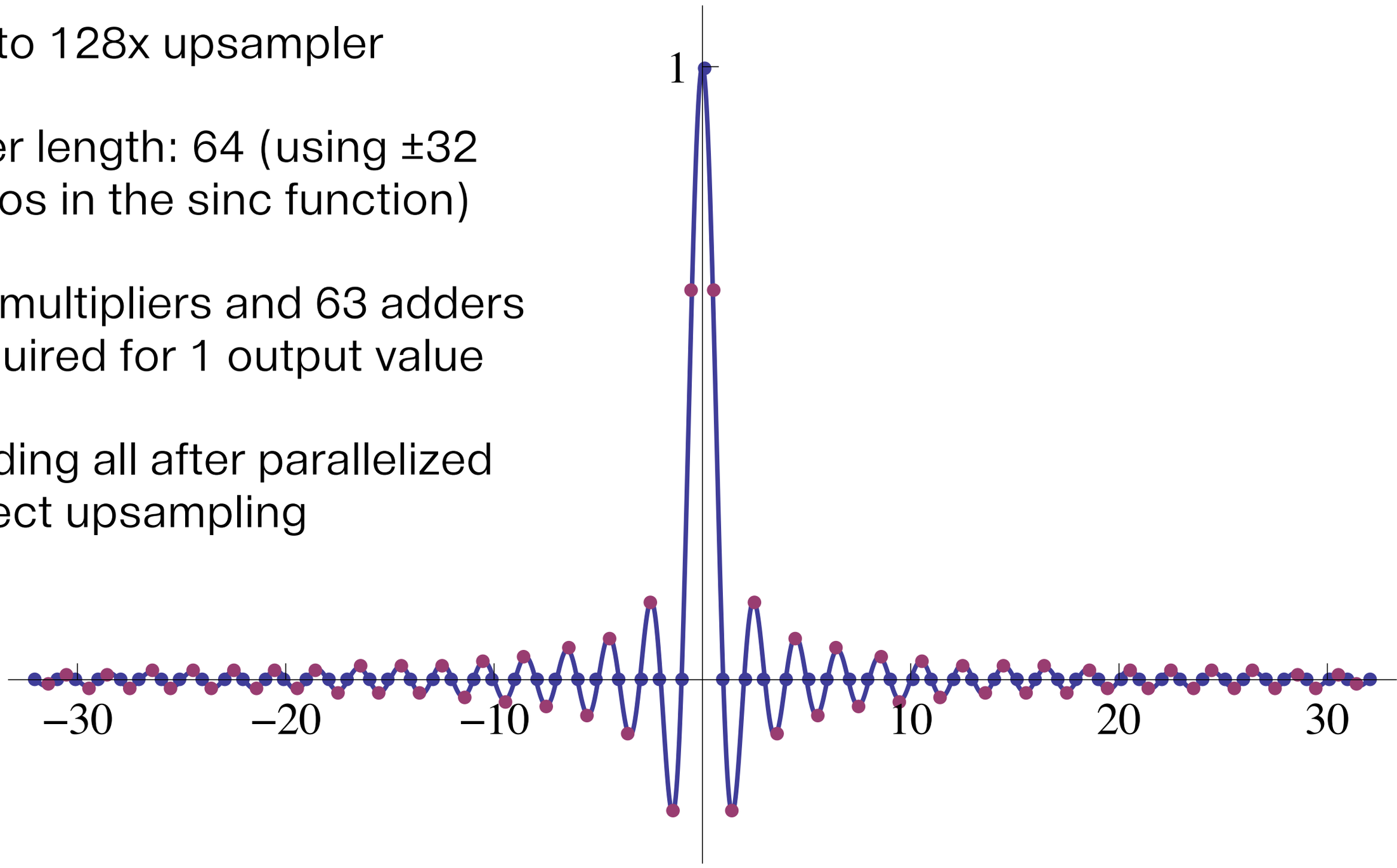
$$x_k^{\text{new}} = \frac{\min(f_s, f_s^{\text{new}})}{f_s} \sum_{n=-\infty}^{\infty} x_n \operatorname{sinc}\left[\pi \min(f_s, f_s^{\text{new}}) \left(\frac{k}{f_s^{\text{new}}} - \frac{n}{f_s}\right)\right],$$

$$x_k^{(m)} = \frac{\min(f_s, f_s^{\text{new}})}{f_s} \sum_{n \in [\tilde{k}-m, \tilde{k}+m]} x_n \operatorname{sinc}\left[\pi \min(f_s, f_s^{\text{new}}) \left(\frac{k}{f_s^{\text{new}}} - \frac{n}{f_s}\right)\right],$$

$$\text{where } \tilde{k} = \frac{f_s k}{f_s^{\text{new}}}$$

Strategy

- 2x to 128x upsampler
- filter length: 64 (using ±32 zeros in the sinc function)
- 64 multipliers and 63 adders required for 1 output value
- Adding all after parallelized direct upsampling

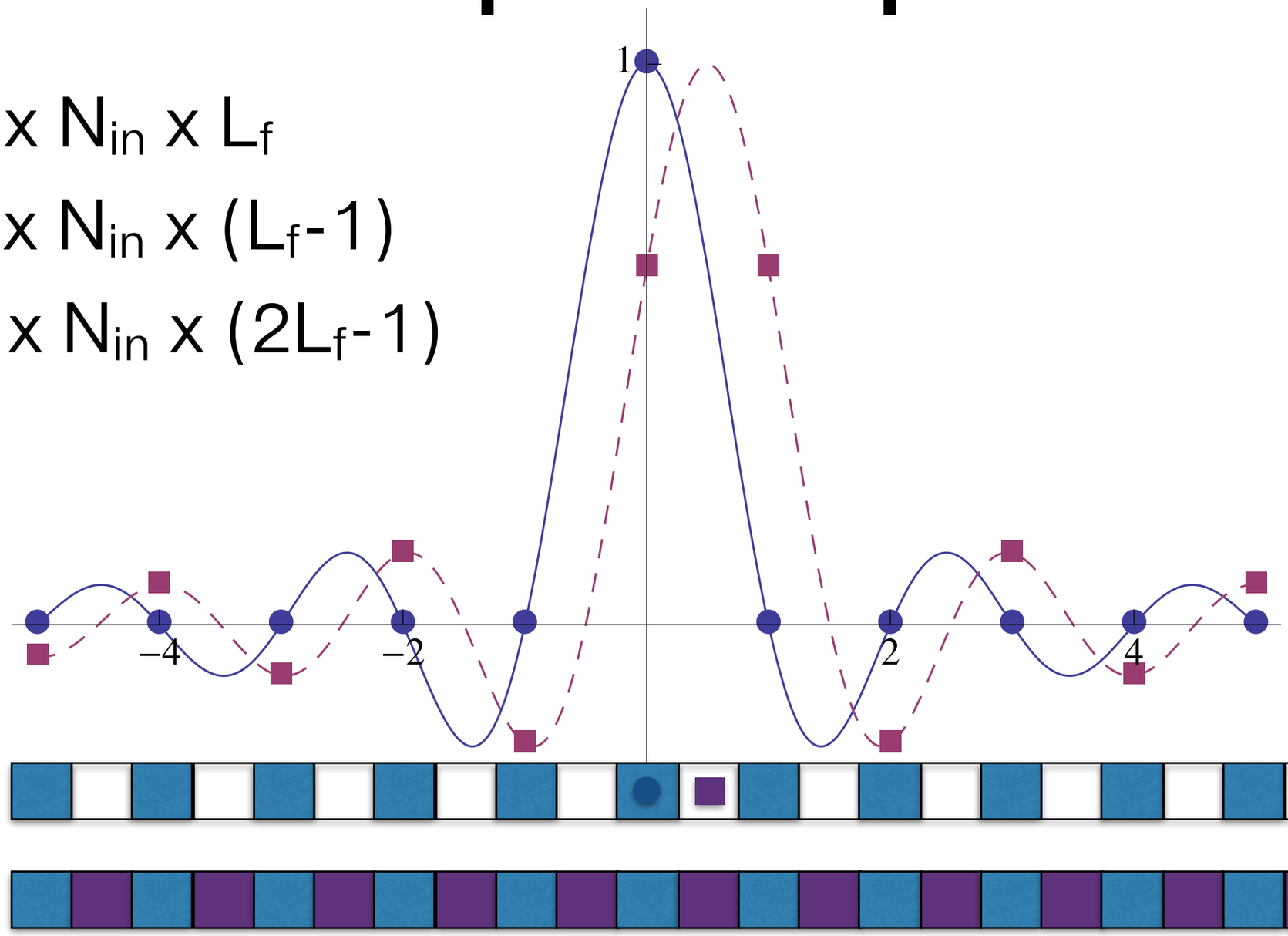


2x Upsampler

$$N_x = 2 \times N_{in} \times L_f$$

$$N_+ = 2 \times N_{in} \times (L_f - 1)$$

$$N_{flops} = 2 \times N_{in} \times (2L_f - 1)$$

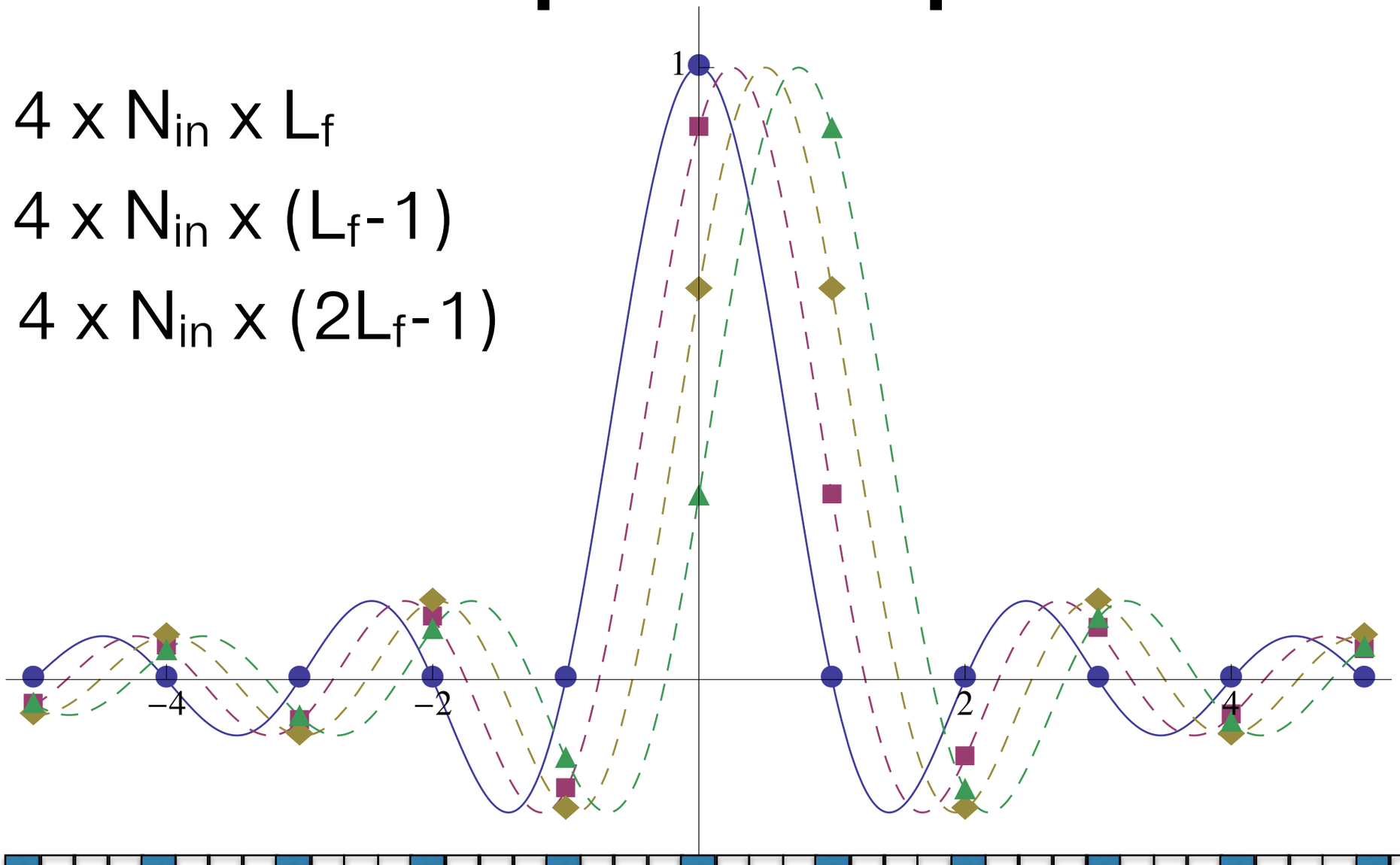


4x Upsampler

$$N_x = 4 \times N_{in} \times L_f$$

$$N_+ = 4 \times N_{in} \times (L_f - 1)$$

$$N_{flops} = 4 \times N_{in} \times (2L_f - 1)$$



Input:

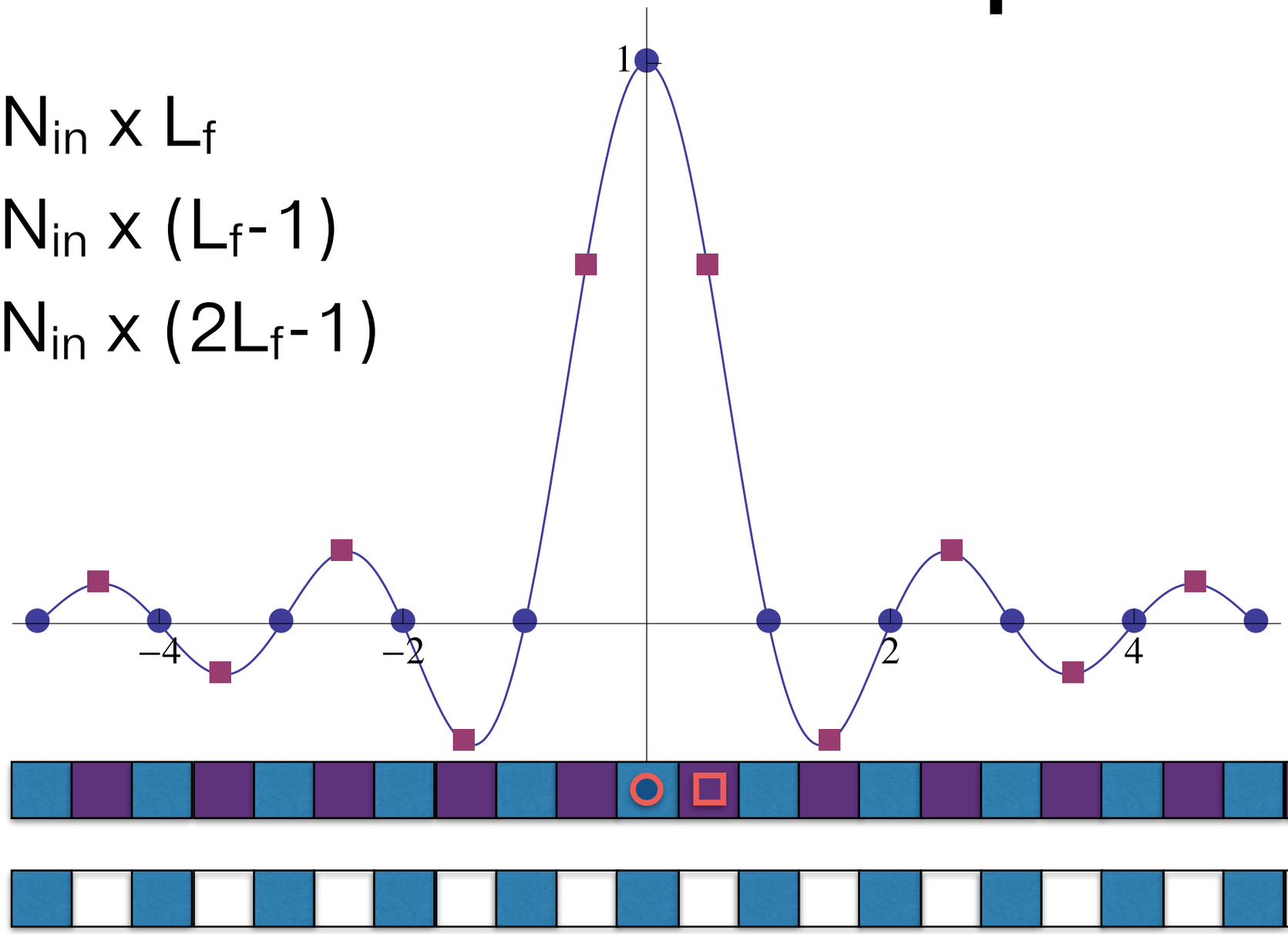
Output:

2x Downsampler

$$N_x = N_{in} \times L_f$$

$$N_+ = N_{in} \times (L_f - 1)$$

$$N_{flops} = N_{in} \times (2L_f - 1)$$

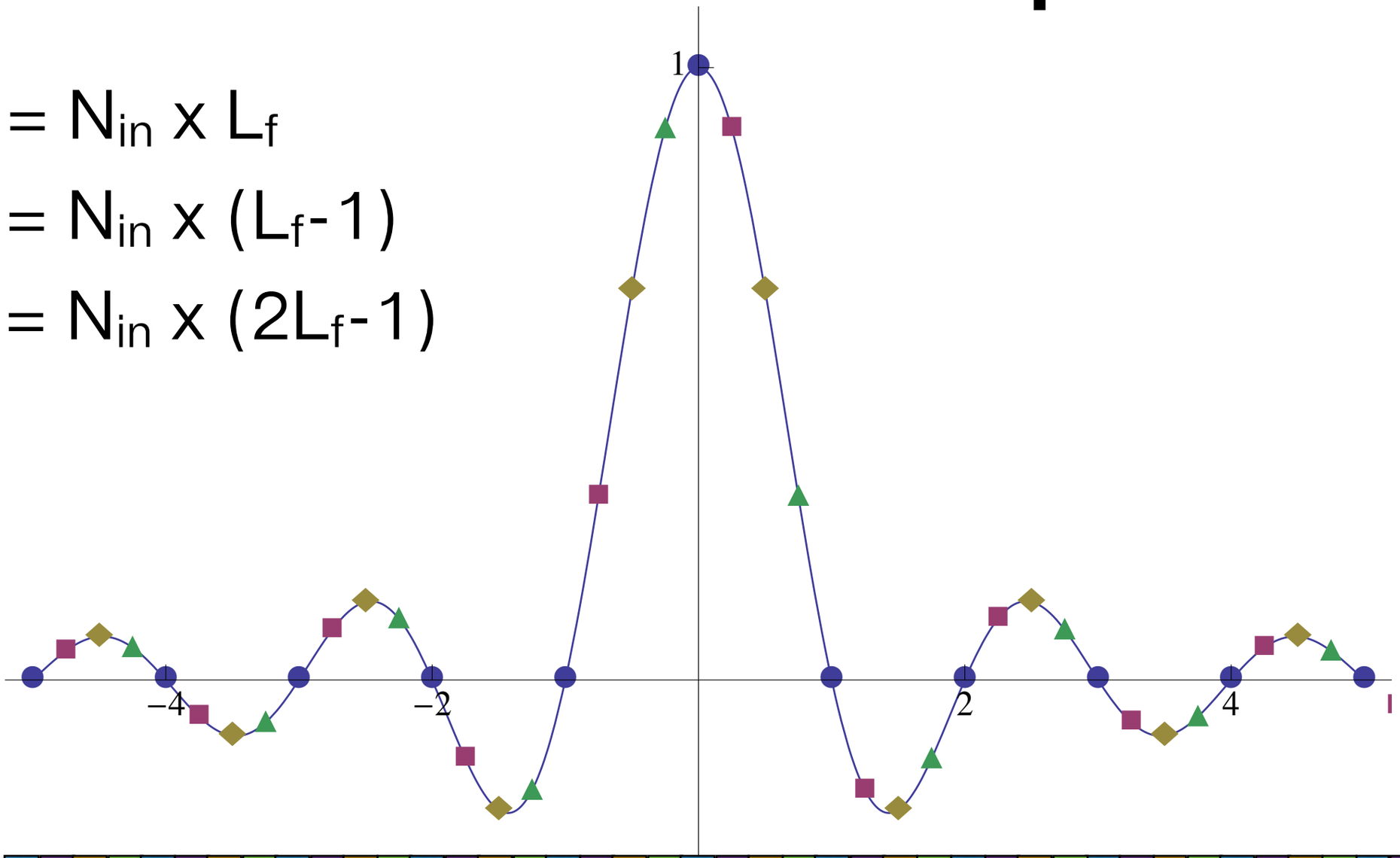


4x Downsampler

$$N_x = N_{in} \times L_f$$

$$N_+ = N_{in} \times (L_f - 1)$$

$$N_{flops} = N_{in} \times (2L_f - 1)$$

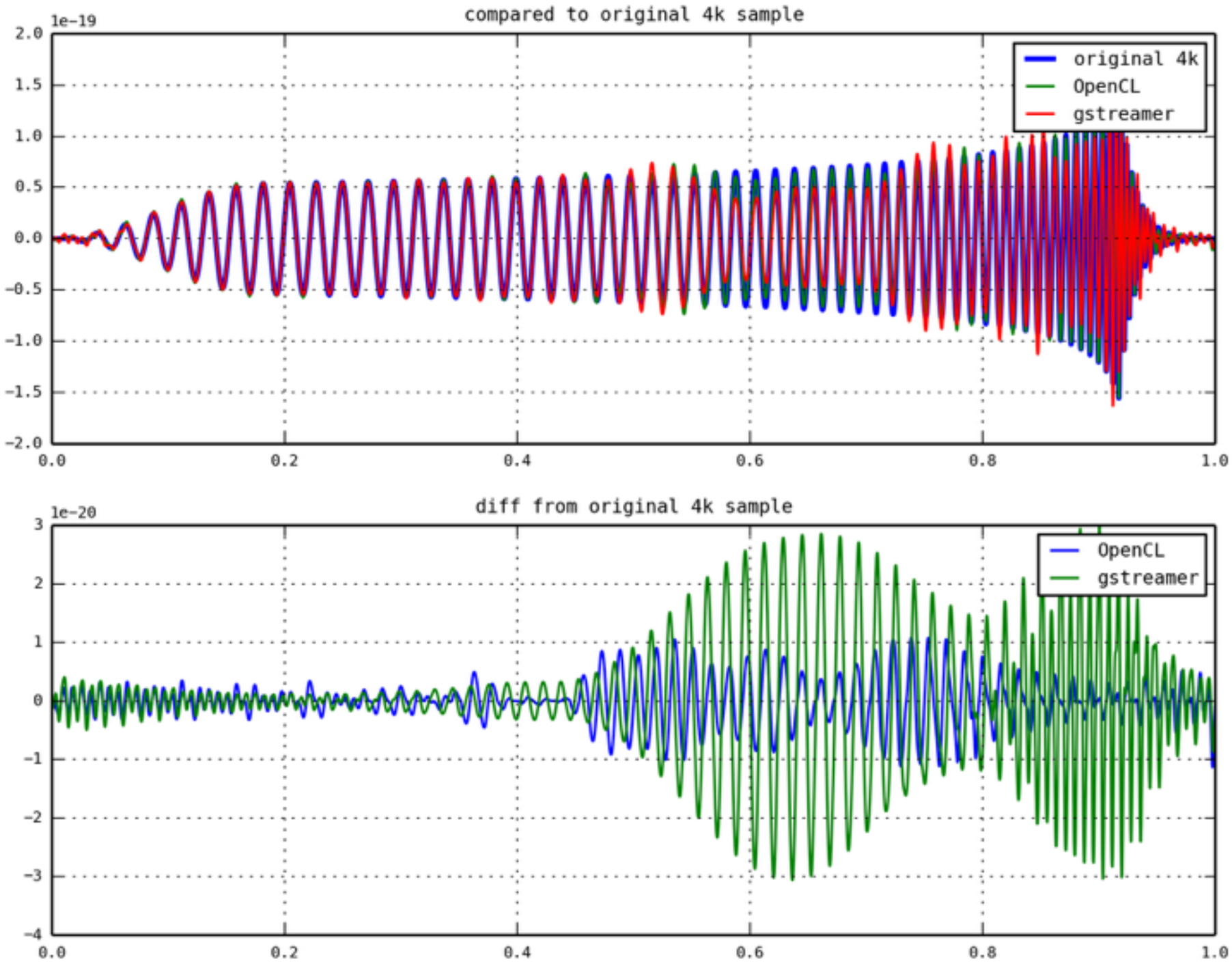


Input:

Output:

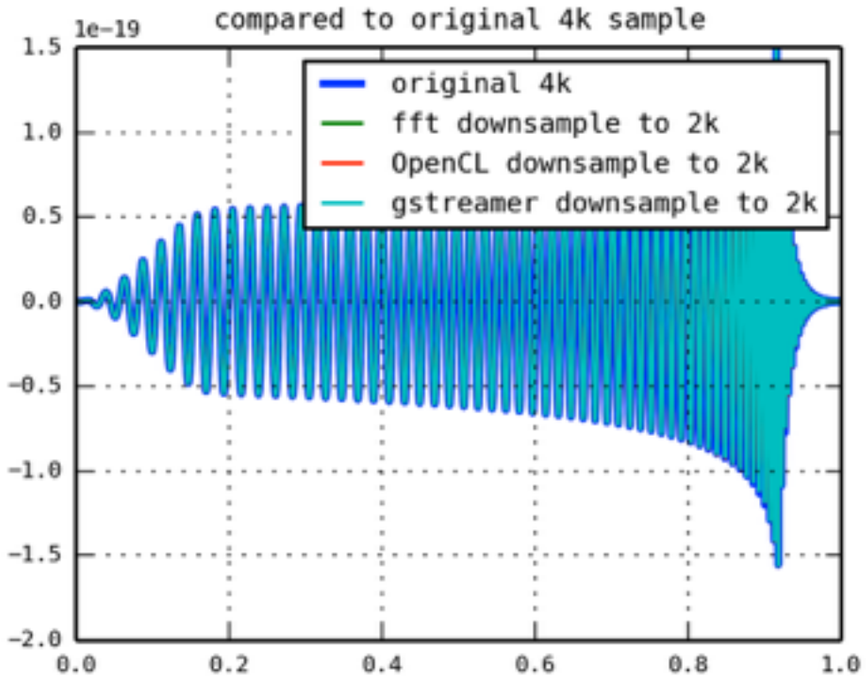
Comparison w/ gstreamer

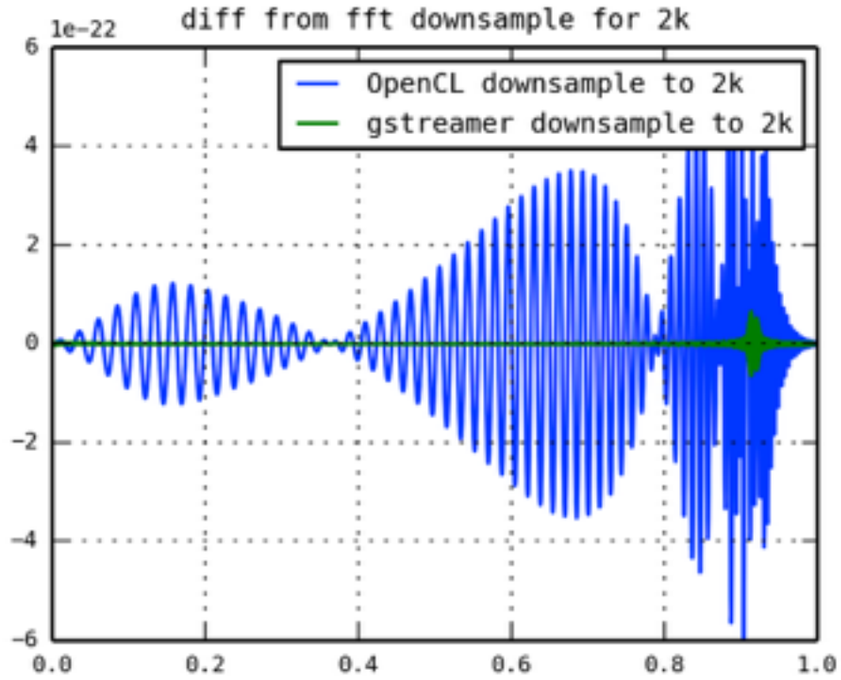
(Upsampler)

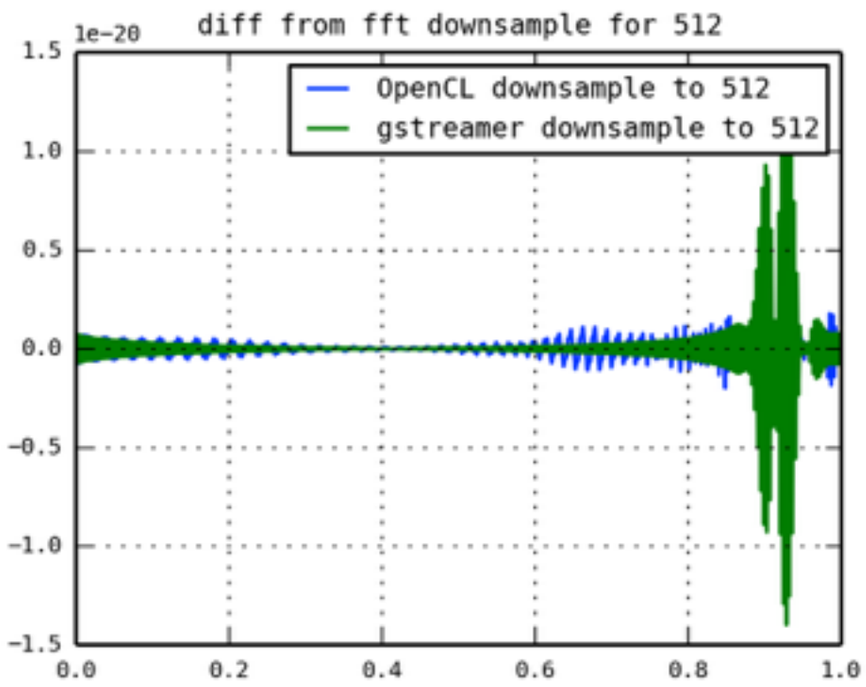


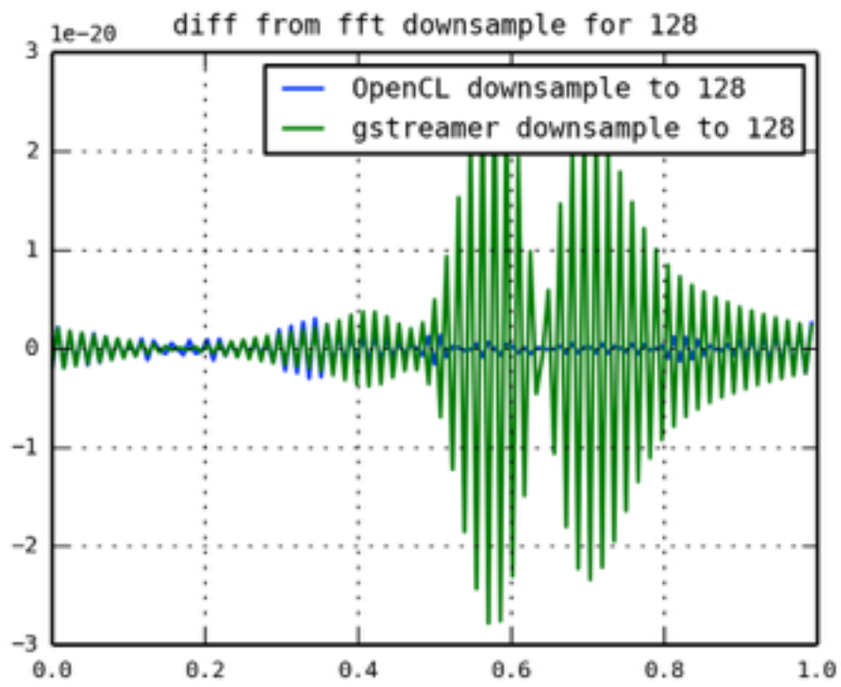
Comparison w/ gstreamer

(Downsampler)









Result & Discussion

- The difference between the result and the original data is less than around 10 percent, which shows **better accuracy** than the gstreamer.

- It is reported that the audioresample in gstreamer used in SPIIR pipeline takes about **0.46ms in a single CPU core**.
  - sequential up sampling

- Sequential upsampling in OpenCL takes about **0.1ms in NVIDIA GPU** and 0.5ms in an octa-core CPU.

- The downsampling test shows **similar speedup**.

- If we consider an octa-core CPU and 4 GPU devices in a single node, **the expected speedup is twice**.

Thank you!