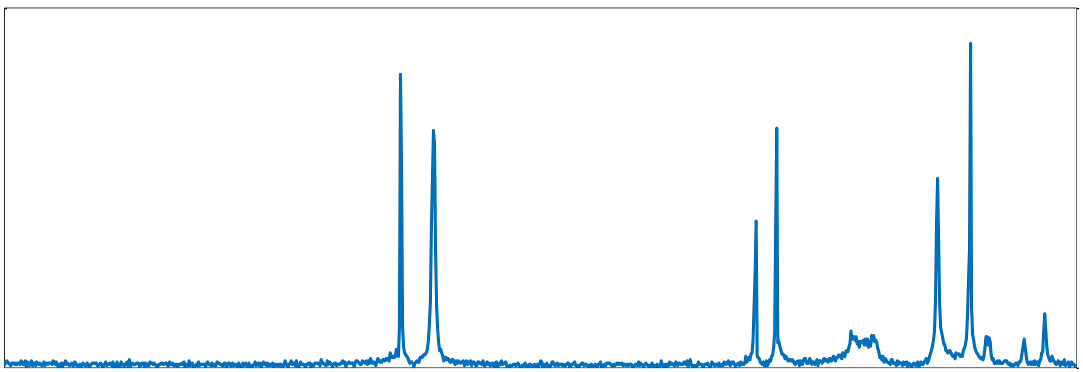
Recent Developments in the Sparse Fourier Transform



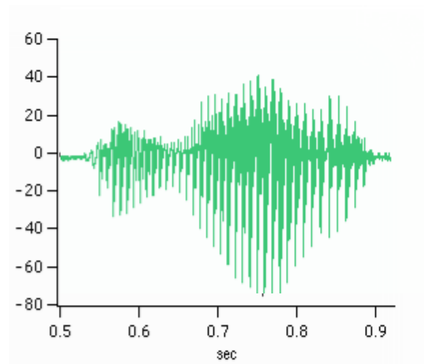
Piotr Indyk

MIT

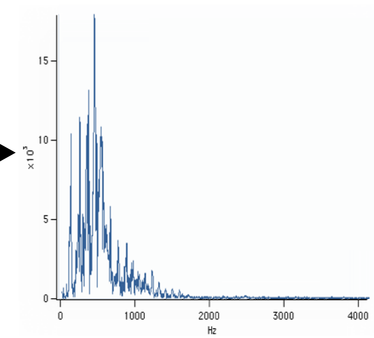
Fourier Transform

- Discrete Fourier Transform:
  - Given: a signal a[1…n]
  - Goal: compute the frequency vector â where

$$\hat{a}_f = \Sigma_t \, a_t \, e^{-2\pi i \, tf/n}$$

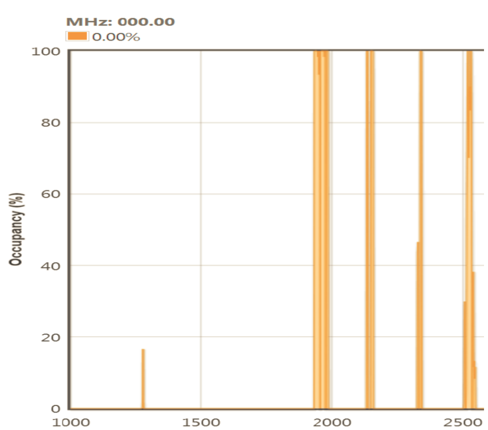


DFT

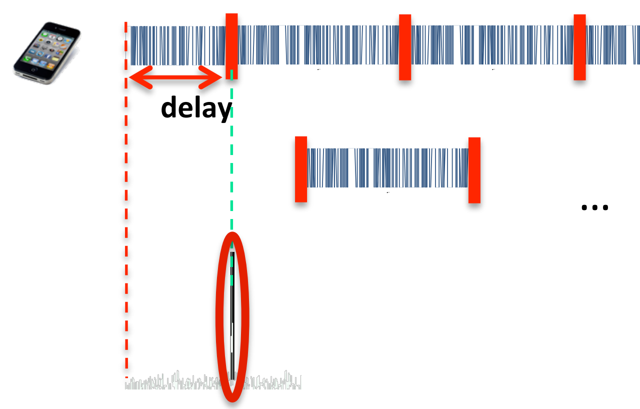


- Very useful tool

Video / Audio compression denoising



Communication

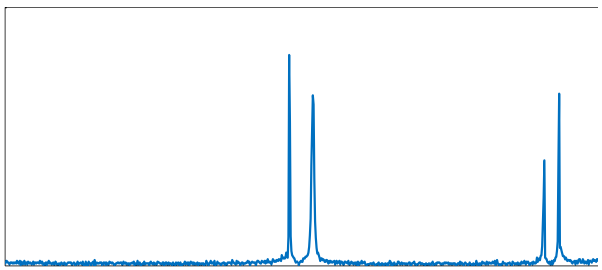


Convolutions

Known algorithms

- Fast Fourier Transform (FFT) computes the frequencies in time $O(n \log n)$
- Not known if the bound can be improved
- But, we can do better if we only care about small number k of "dominant frequencies"
  - E.g., recover assuming DFT is k-sparse (only k non-zero entries)
- Plenty of algorithms known:
  - Boolean cube (Hadamard Transform): [KM'91] (cf. [GL])
  - Complex FT: [Mansour'92, GGIMS'02, AGS'03, GMS'05, Iwen'10, Akavia'10]
- Best running time: $k \log^c n$ for some $c=O(1)$ [Gilbert-Muthukrishnan-Strauss'05]
  - Improve over FFT for $k \ll n/\log^{c-1} n$

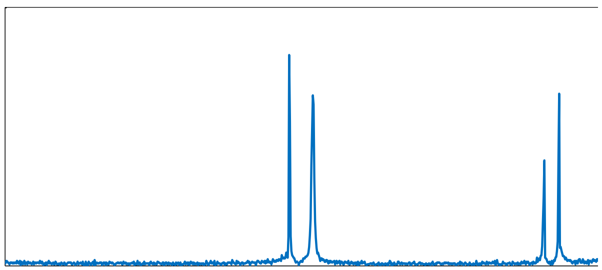
*Assuming entries of x are integers with $O(\log n)$ bits of precision.



Signal spectrum

Challenges

- Run-time: : $k \log^c n$ [GMS05]
- Problem:
  - $c \approx 4$
  - Need $k < 100$ to beat FFTW for $n=4{,}000{,}000$
- Recent line of research:
  - Theory: improve over FFT for all values of $k=o(n)$
  - Improve in practice, applications



Plan

- Introduction
- Results overview
- Techniques
- Applications
- Future directions/open problems

Guarantees

- All algorithms randomized, with constant probability of success, n is a power of 2
- Recovery guarantees:
  - Exactly k-sparse case: report exact answer*
  - Approximately k-sparse case: report k-sparse $\hat{c}$ that satisfies the l_2/l_2 guarantee:

$$||\hat{a}-\hat{c}||_2 \leq C \min_{k\text{-sparse } \hat{u}} ||\hat{a}-\hat{u}||_2$$

*Assuming entries of x are integers with $O(\log n)$ bits of precision.

Recent results

	Time	Sparsity	Comments	Samples
HIKP'12, HIKP'12b	k log n	Exact	Faster than FFTW if k<100,000* (n=4,000,000)	k log n
	k log n log(n/k)	Approximate	Faster than FFTW if k<2000	k log n log(n/k)
GHIKPS'13, PR'13	k log n	Exact	Average case,$k<n^{1-\delta}$**	k
	$k \log^2 n$	Approximate	Average case,$k<n^{1-\delta}$	k log n
IKP'14	$k \log^2 n$	Approximate		$k \log n * \log^c \log n$
IK'14	$n \log^c n$	Approximate		k log n

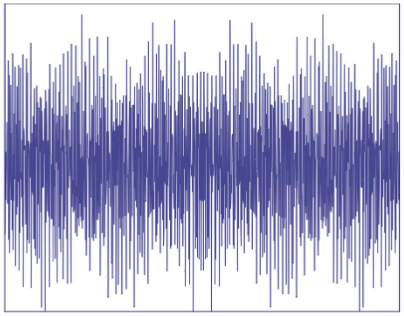
*Further efficiency improvement by 2-5x was achieved by Pueschel-Schumacher'13

**GHIKPS proved it for δ=1/2

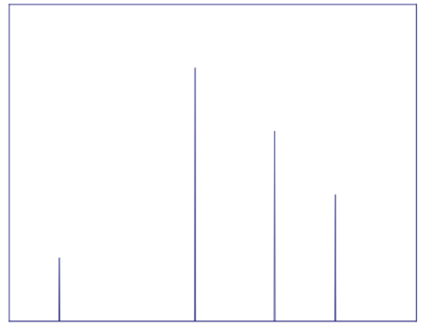
Exact Sparsity

	Time	Sparsity	Comments
HIKP'12b	k log n	Exact	Faster than FFTW if k<100,000

Intuition I: Signal Processing



Time Domain Signal

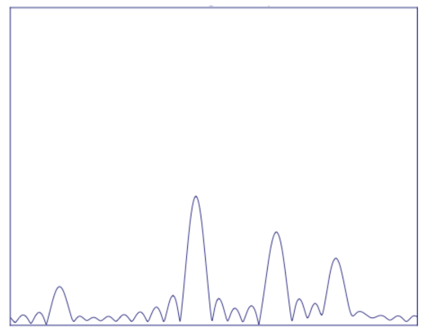


Frequency Domain

n-point DFT : $n\log(n)$

a ➡ â

Cut off Time signal

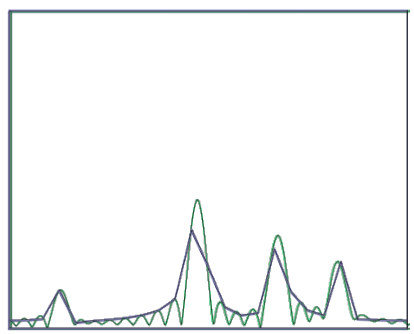


Frequency Domain

n-point DFT of first B terms : $n\log(n)$

Rect × a ➡ Sinc * â

First B samples



Frequency Domain

B-point DFT of first B terms: $B\log(B)$

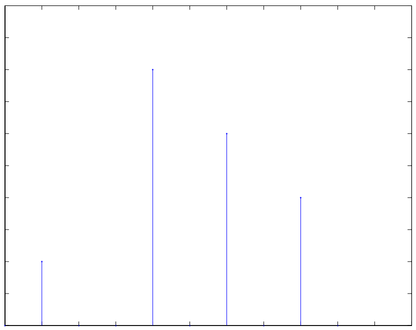
Alias(Rect × a)

⬇

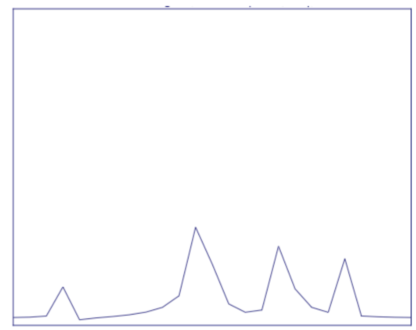
Subsample(Sinc * â)

Intuition II: Computer Science

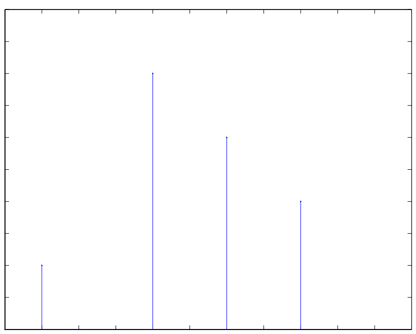
- Make this:



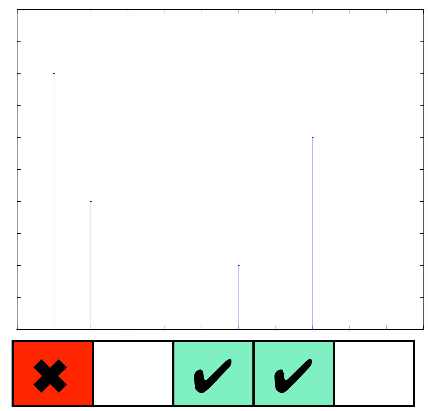
previous slide



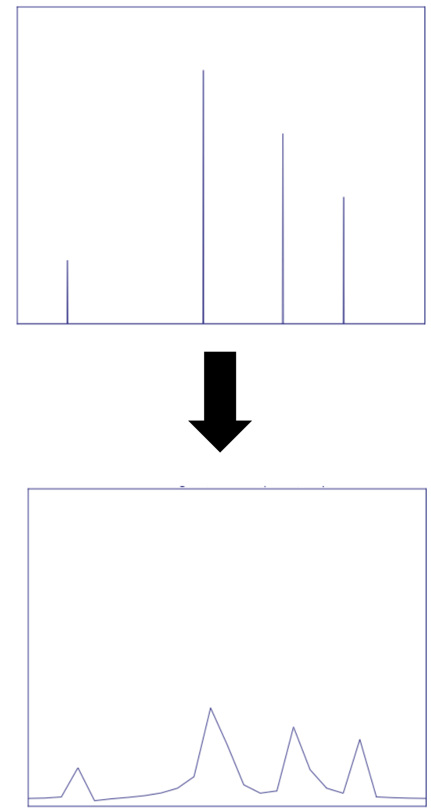
… more like this:



balls into random bins
(hashing)



Issues



- Issues:
  - Where is hashing ? Need some random rearrangement

    Show how to permute the spectrum pseudo-random by permuting the signal
    (or just assume randomness)
  - Leakage

    Replace Boxcar filter by a nicer function
  - Identify the isolated coefficients

    Recover the index from the phase
    - “OFDM trick” (see also [A. K. Paul’72])
    - Special case of Matrix Pencil, Prony method [Chiu-Demanet’13, Potts-Kunis-Heider-Veit’13]

Spectral hashing

Pseudo-random Spectrum Permutation

[Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02, Gilbert-Muthukrishnan-Strauss'05]

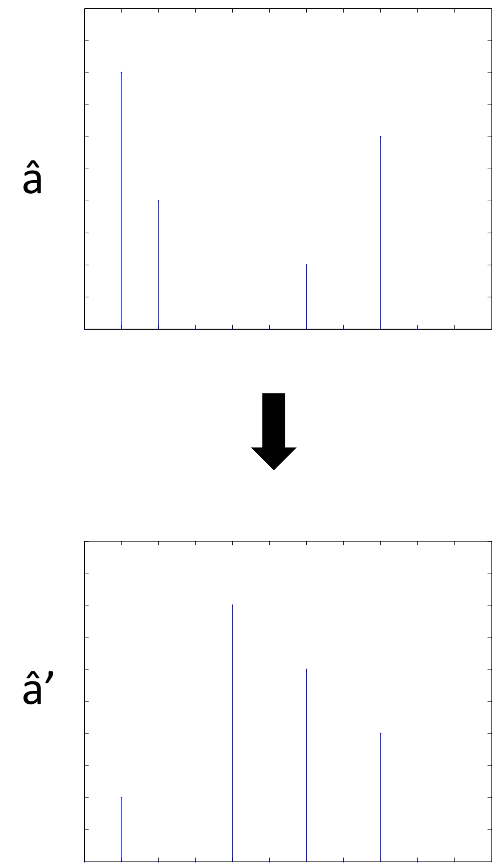
- Permute time domain signal → permute frequency domain
- Let

$$a'_t = a_{\sigma t}\, e^{-2\pi i\, t\, \beta / n}$$

- If σ is invertible mod n

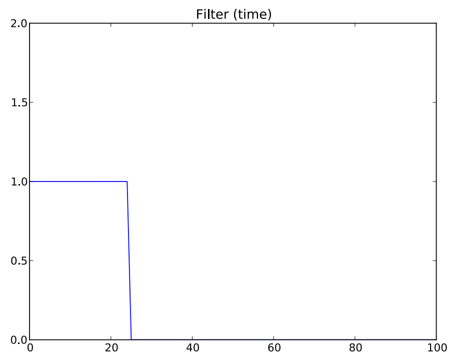
$$\hat{a}'_f = \hat{a}_{1/\sigma\, f + \beta}$$

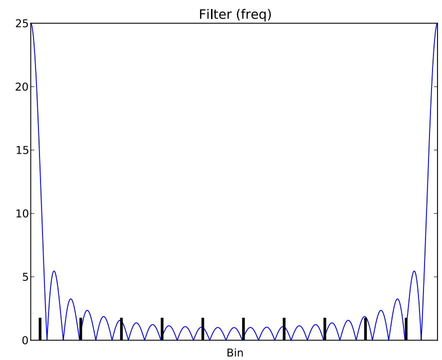
  - If n is a power of 2, any odd σ is OK
- Pseudo-random permutation: select
  - β uniformly at random from {0…n-1}
  - σ uniformly at random from odd numbers in {0…n-1}
- Each access to a coordinate of a'_t can be simulated by accessing $a_{\sigma t}$ and multiplication



Reducing leakage

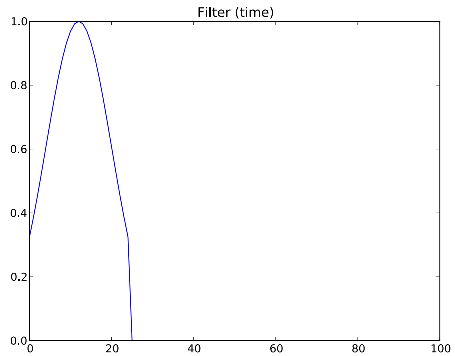
Filters: boxcar filter (used in[GGIMS02,GMS05])

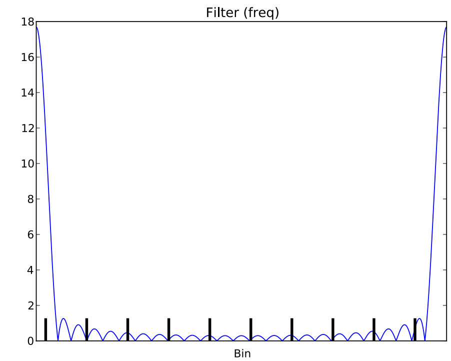




- Rect -> Sinc
  - Polynomial decay
  - Leaking to many buckets

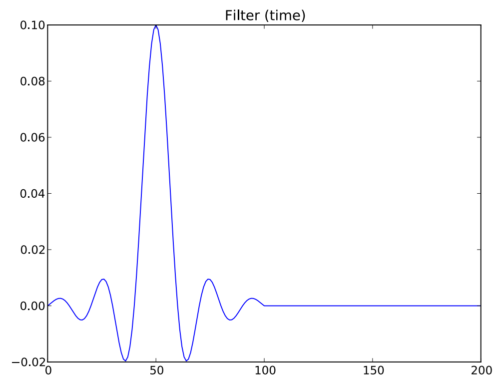
Filters: Gaussian

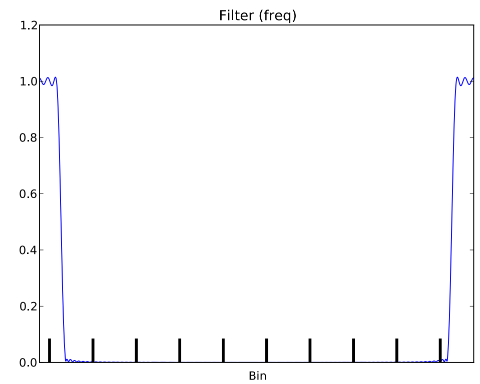




- Gaussian -> Gaussian
  - Exponential decay
  - Leaking to (log n)$^{1/2}$ buckets

Filters: Sinc × Gaussian

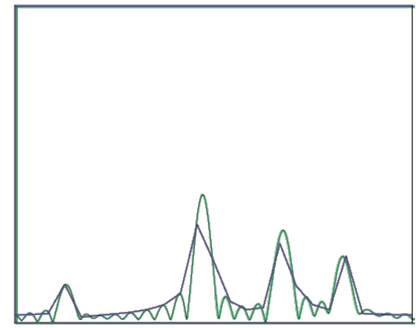




- Sinc Gaussian -> Rect*Gaussian
  - Still exponential decay
  - Leaking to <1 buckets
  - Sufficient contribution to the correct bucket
- Actually we use Dolph-Chebyshev filters

Finding the support

Finding the support



- $\hat{c}$= B-point DFT $(a \times G)$
  = Subsample($\hat{a} * \hat{G}$)
- Assume no collisions:
  - At most one large frequency hashes into each bucket.
  - Large frequency f_1 hashes to bucket b_1

$$\hat{c}_{b1} = \hat{a}_{f1} \hat{G}_{\Delta} + \text{lea}\cancel{\text{kage}}$$

- Let a^{τ} be the signal time-shifted by τ, i.e. $a^{\tau}_t = a_{t-\tau}$
- Recall $\mathrm{DFT}(a^{\tau})_f = \hat{a}_f \, e^{-2\pi i \tau f/n}$
- $\hat{c}^{\tau}$ = B-point DFT $(a^{\tau} \times G)$

$$\hat{c}^{\tau}_{b1} = \hat{a}_{f1} e^{-2\pi i \tau f1/n} \hat{G}_{\Delta} + \text{lea}\cancel{\text{kage}}$$

Finding the support, ctd

- At most one non-zero frequency f_1 per bucket b_1
- We have

$$\hat{c}_{b1}=\hat{a}_{f1}\hat{G}_{\Delta}$$

and

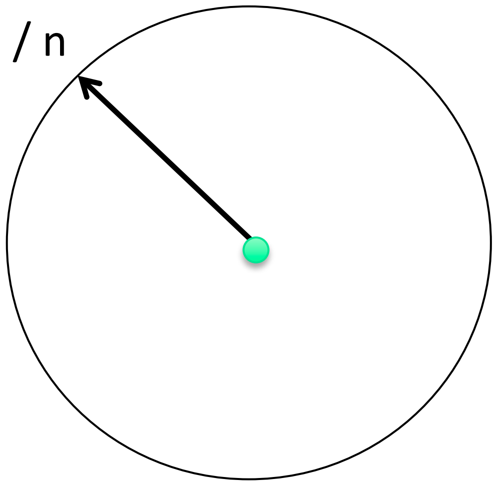
$$\hat{c}^{\tau}_{b1}=\hat{a}_{f1}e^{-2\pi i\,\tau\, f1/n}\,\hat{G}_{\Delta}$$

- So, for $\tau=1$ we have

$$\hat{c}_{b1}/\hat{c}^{1}_{b1} = e^{-2\pi i\, f1\,/n}$$

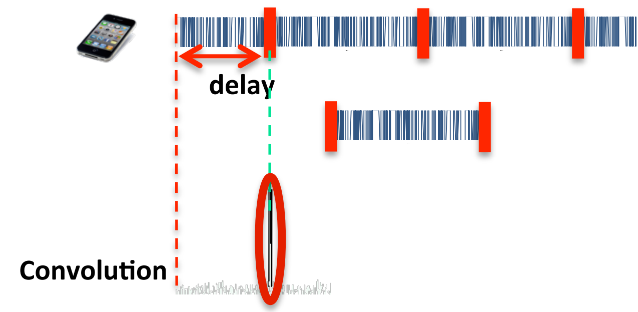
- Can get f1 from the phase
- Digression:
  - Cannot do this when the noise too large (approximately k-sparse case)
  - Instead, read bit by bit, multiply the runtime and sample complexity

    by log(n/k)

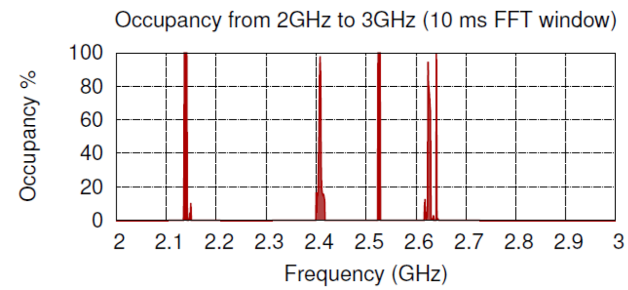


Applications

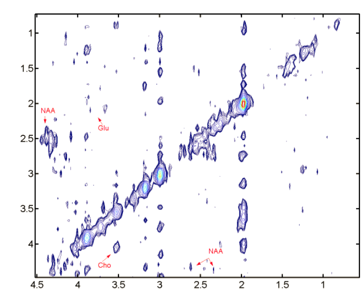
Applications



- GPS synchronization [Hassanieh-Adib-Katabi-Indyk, MOBICOM'12]



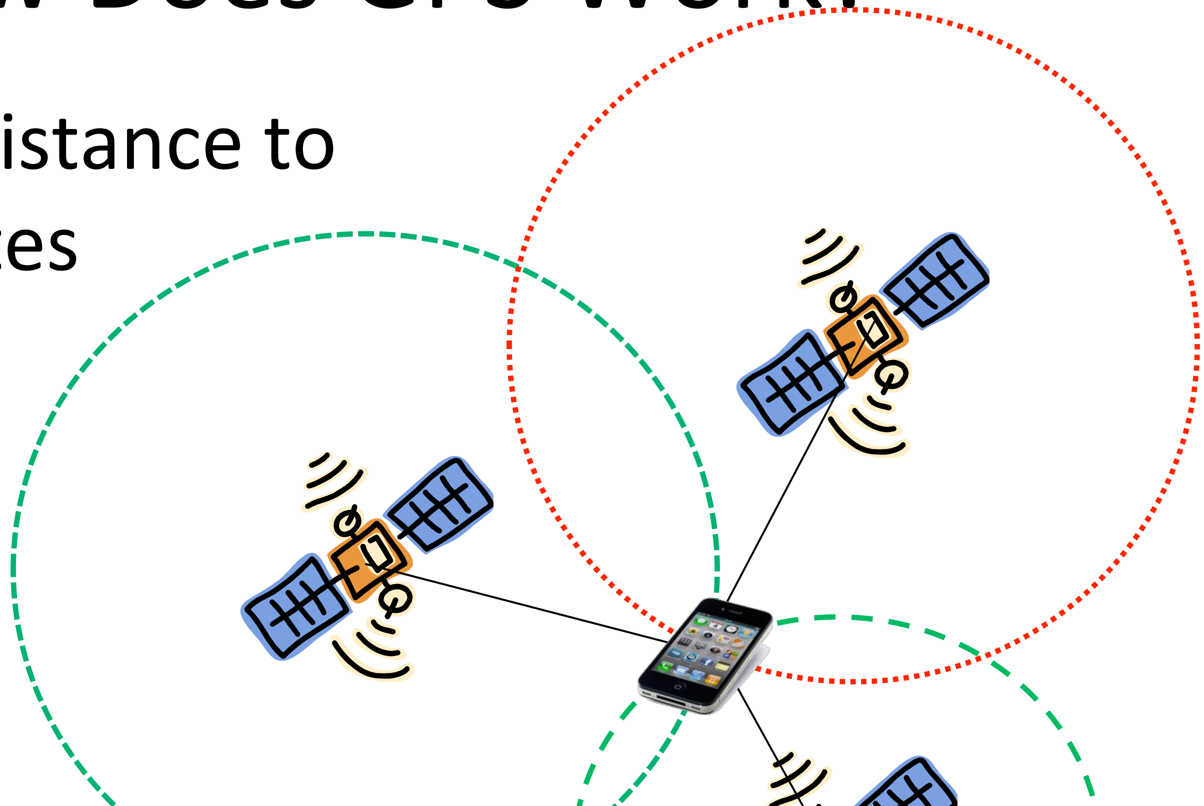
- Spectrum sensing [Hassanieh-Shi-Abari-Hamed-Katabi, INFOCOM'14]



- Magnetic Resonance Spectroscopy [Shi-Andronesi-Hassanieh-Ghazi-Katabi-Adalsteinsson' ISMRM'13]
- Exploiting Sparseness in Speech for Fast Acoustic Feature Extraction [Nirjon-Dickerson-Stankovic-Shen-Jiang, Workshop on Mobile Computing Systems and Applications'13]
- …

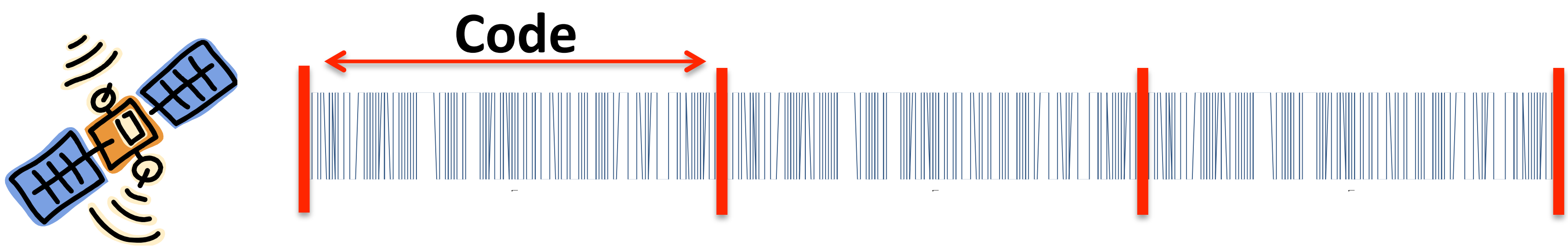
How Does GPS Work?

Compute the distance to the GPS satellites



distance = propagation delay × speed of light

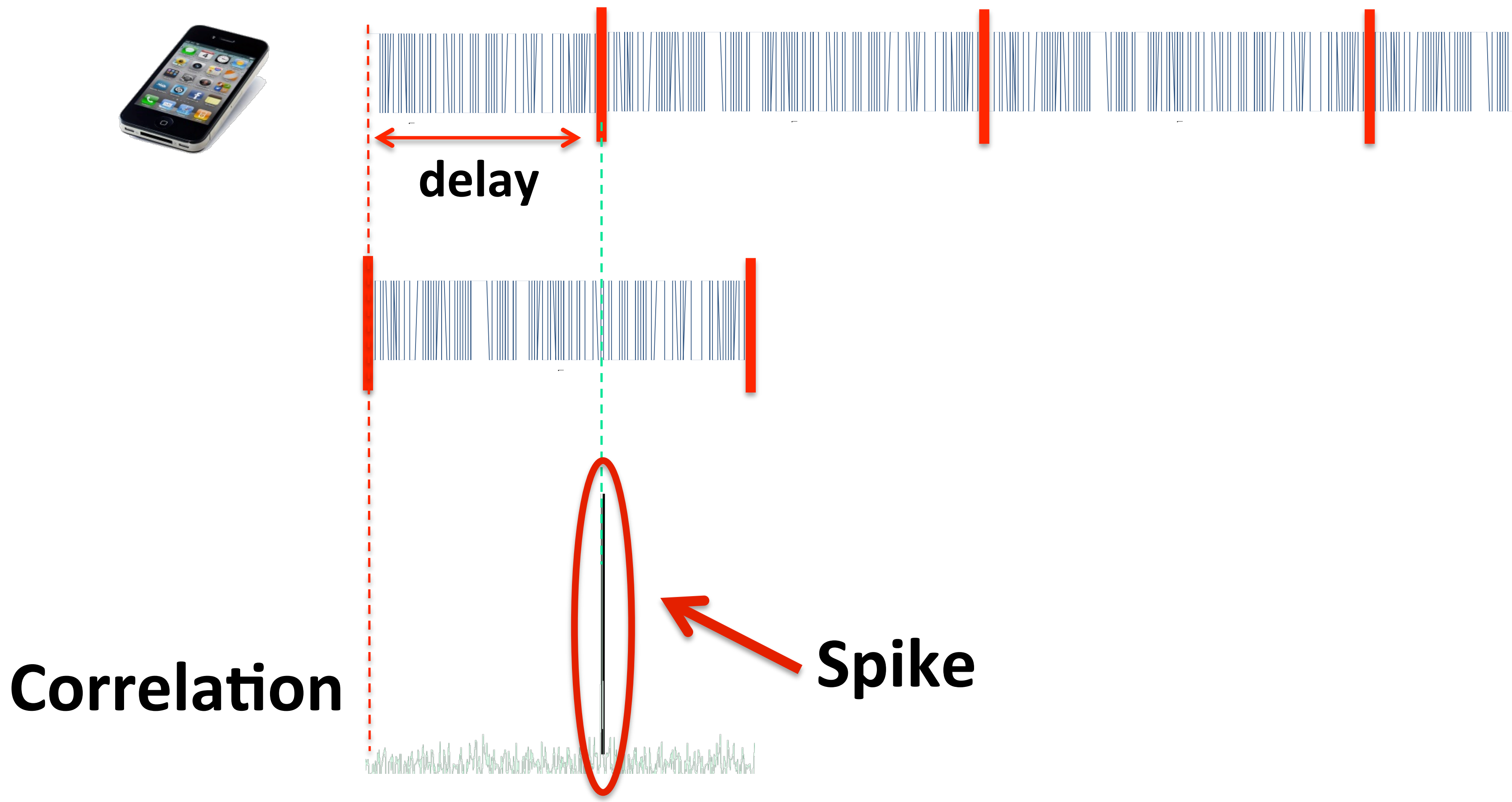
How to Compute the Propagation Delay?



How to Compute the Propagation Delay?

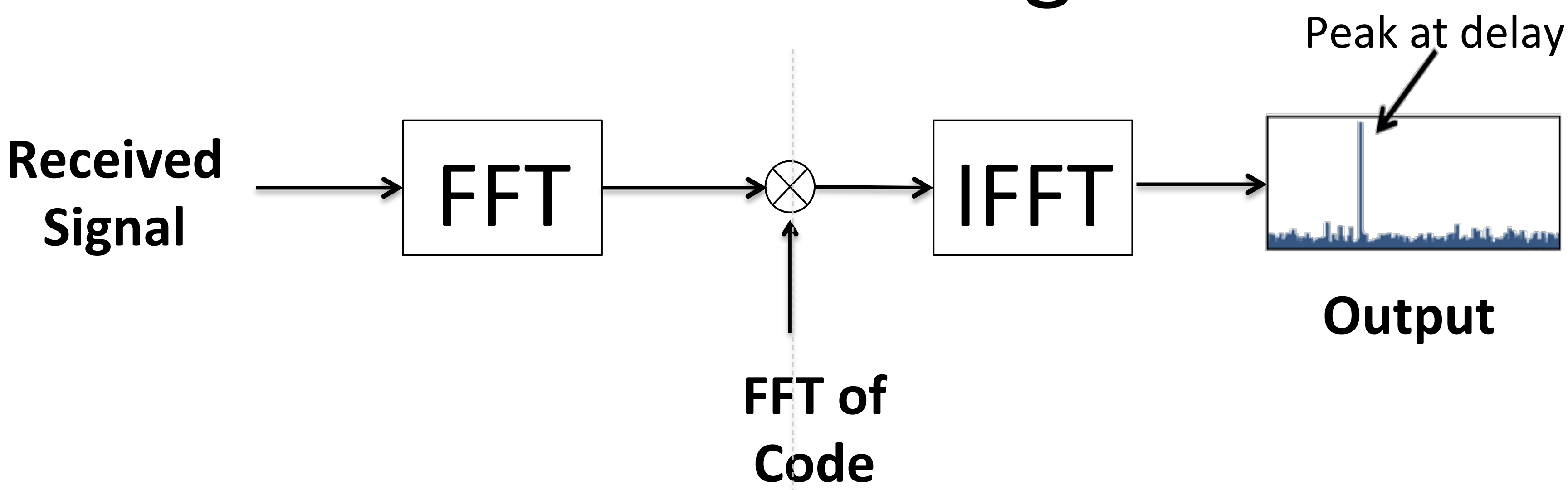
Code arrives shifted by propagation delay

How to Compute the Propagation Delay?



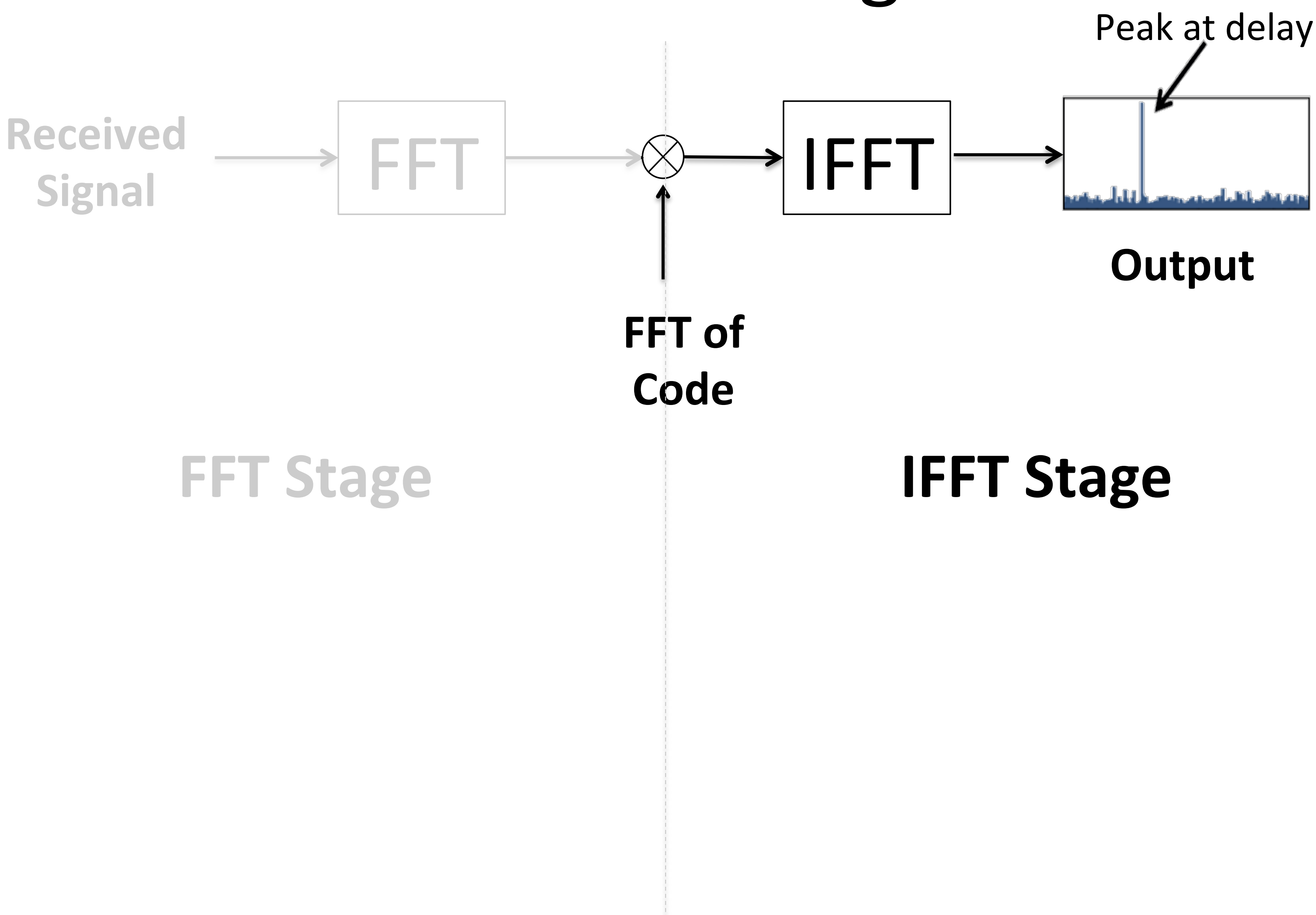
Spike determines the delay

Our GPS Design

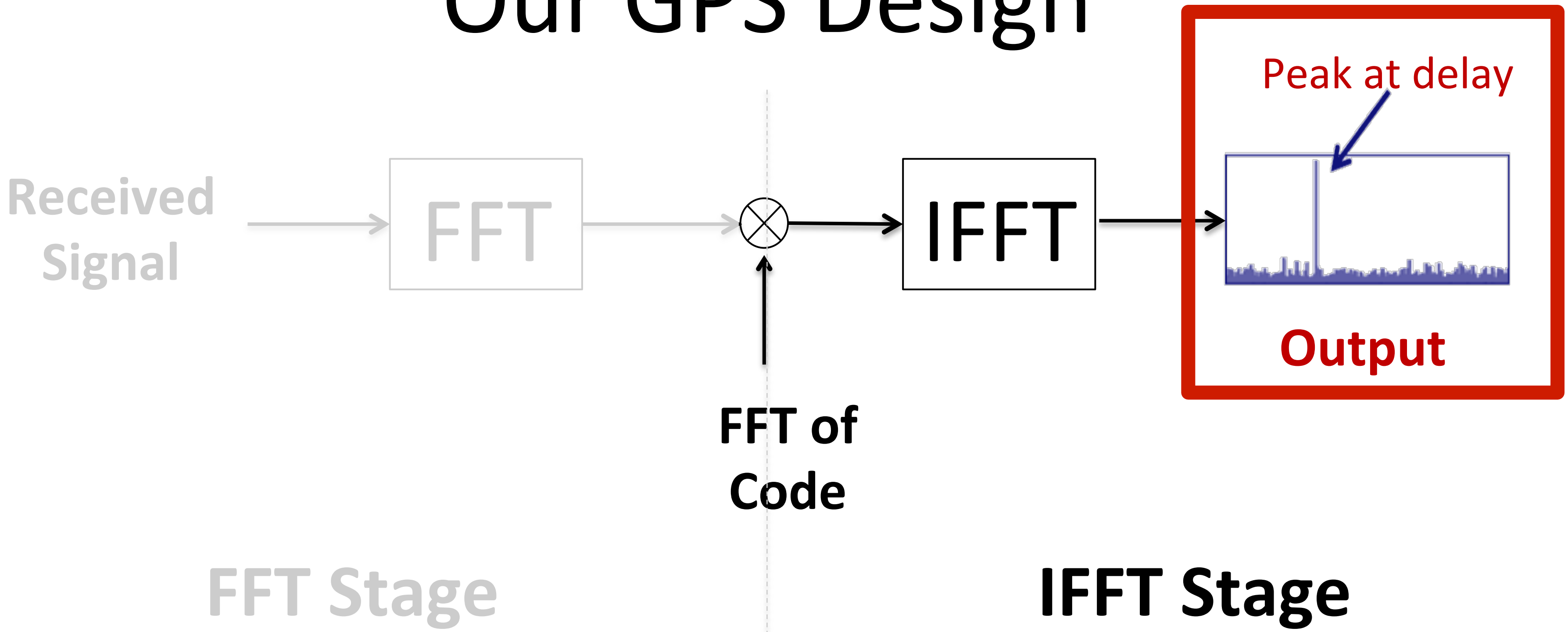


FFT Stage **IFFT Stage**

Our GPS Design

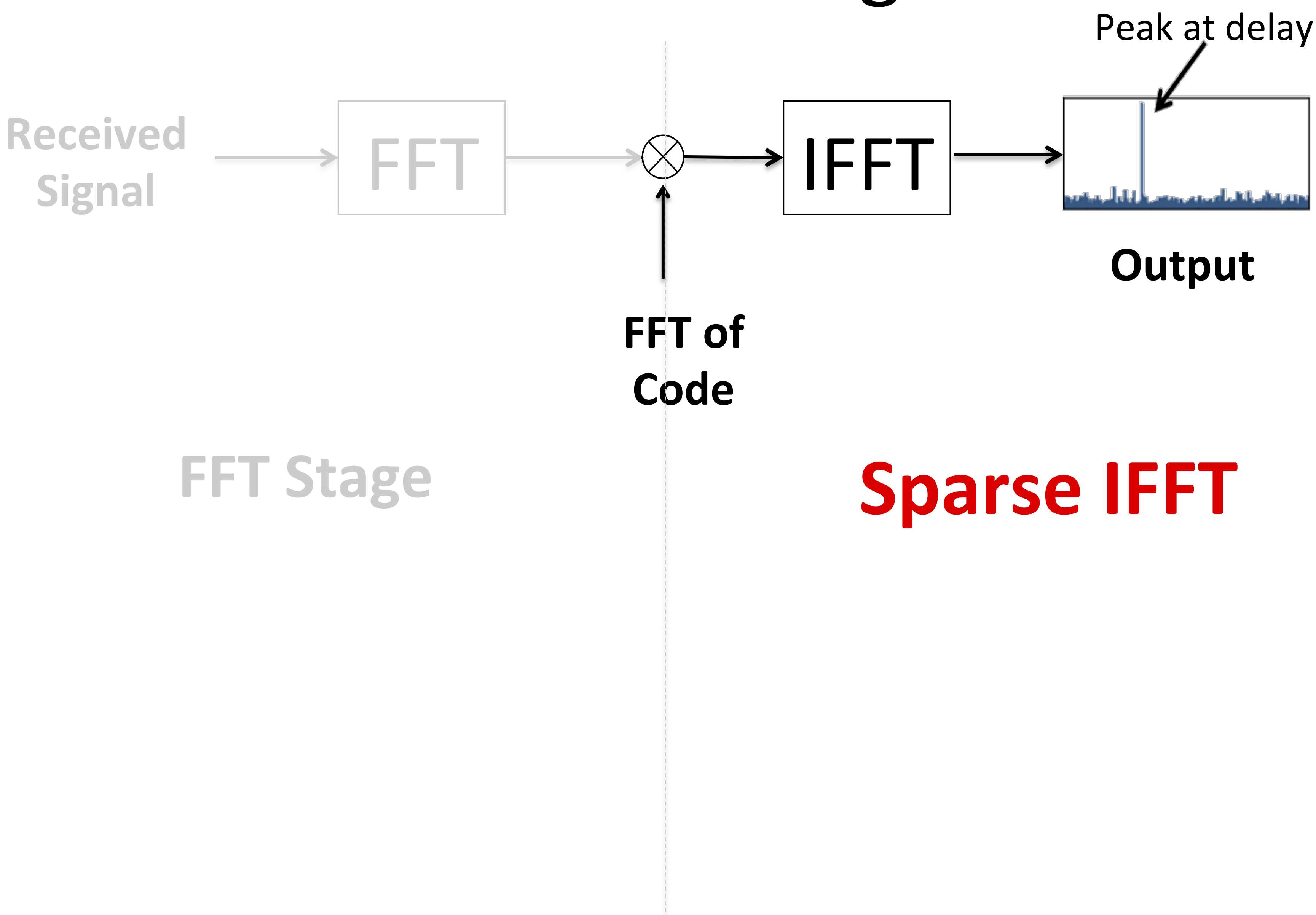


Our GPS Design

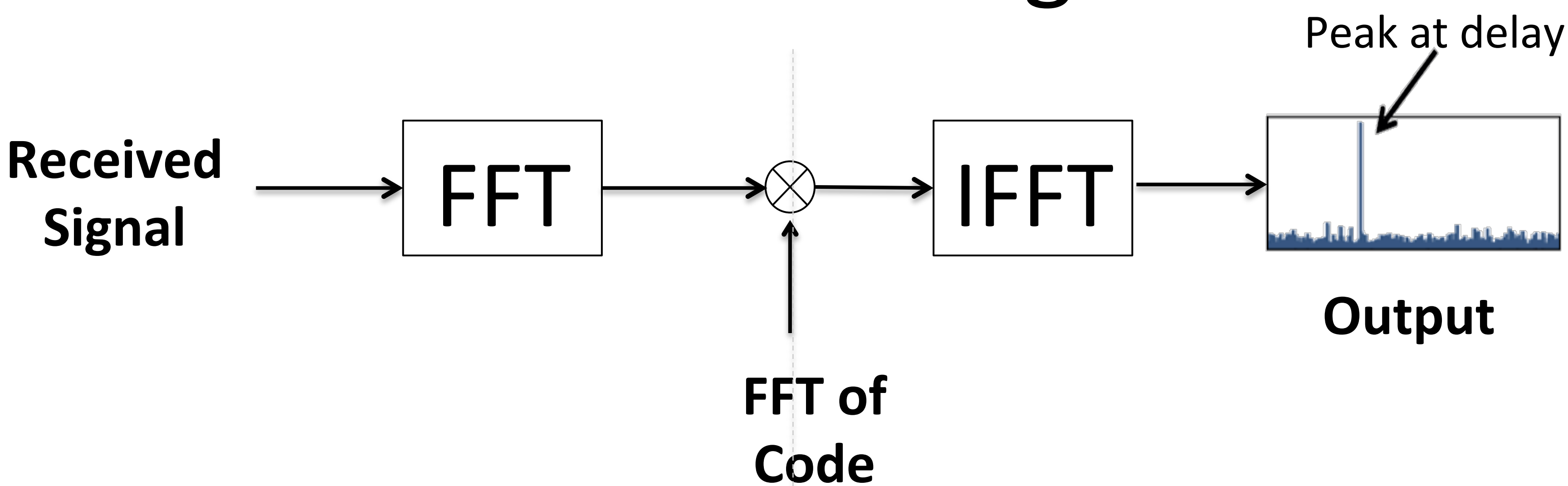


Sparse Fourier Transform

Our GPS Design



Our GPS Design



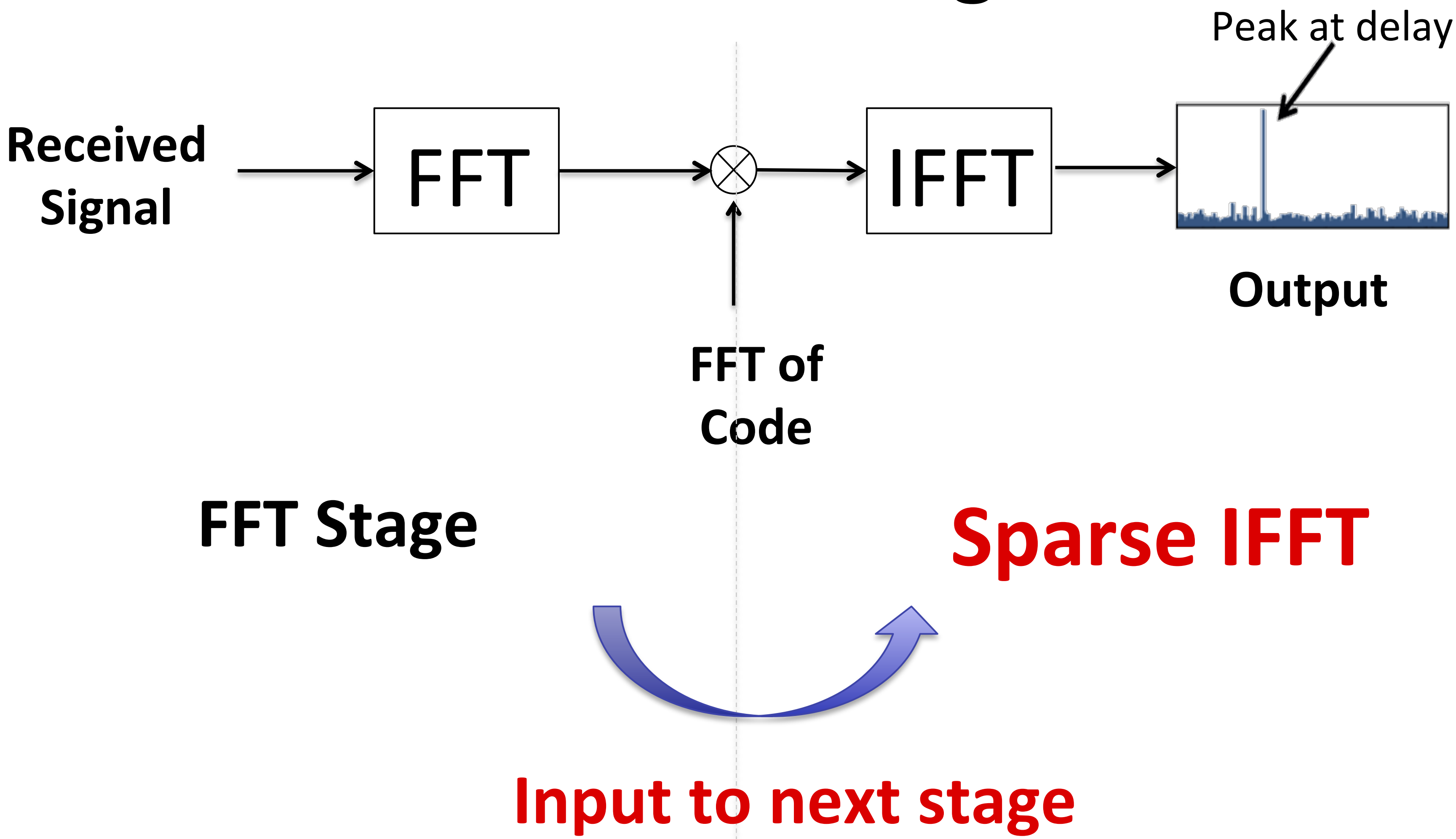
FFT Stage

Output is not sparse

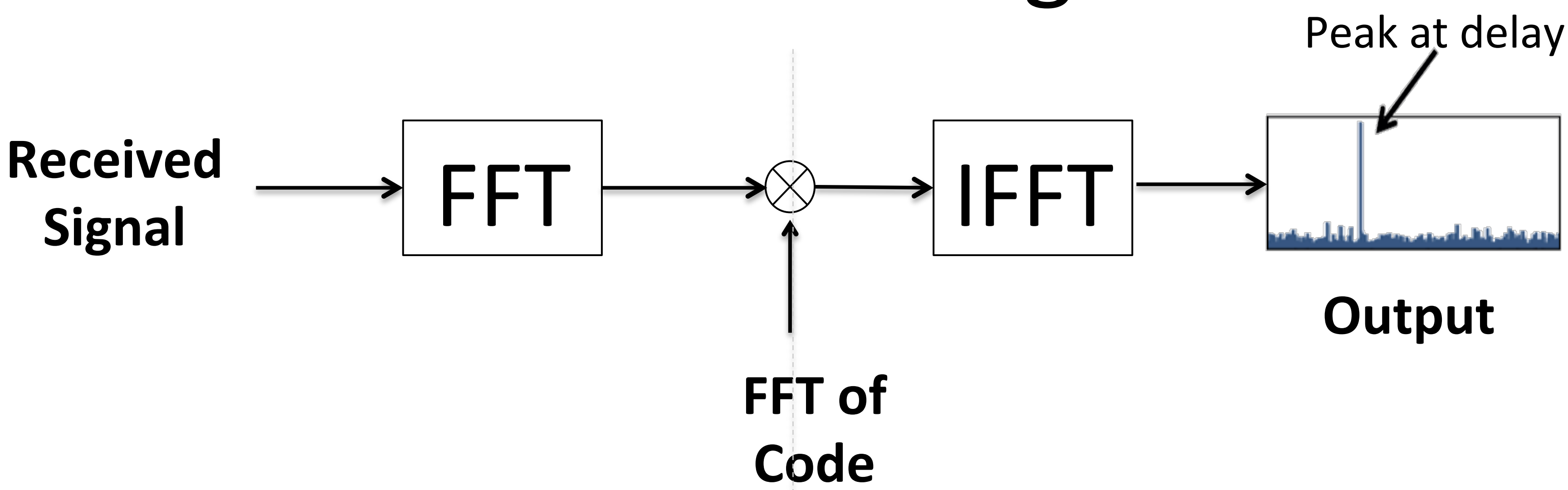
Cannot Use the Sparse Fourier Transform

Sparse IFFT

Our GPS Design



Our GPS Design



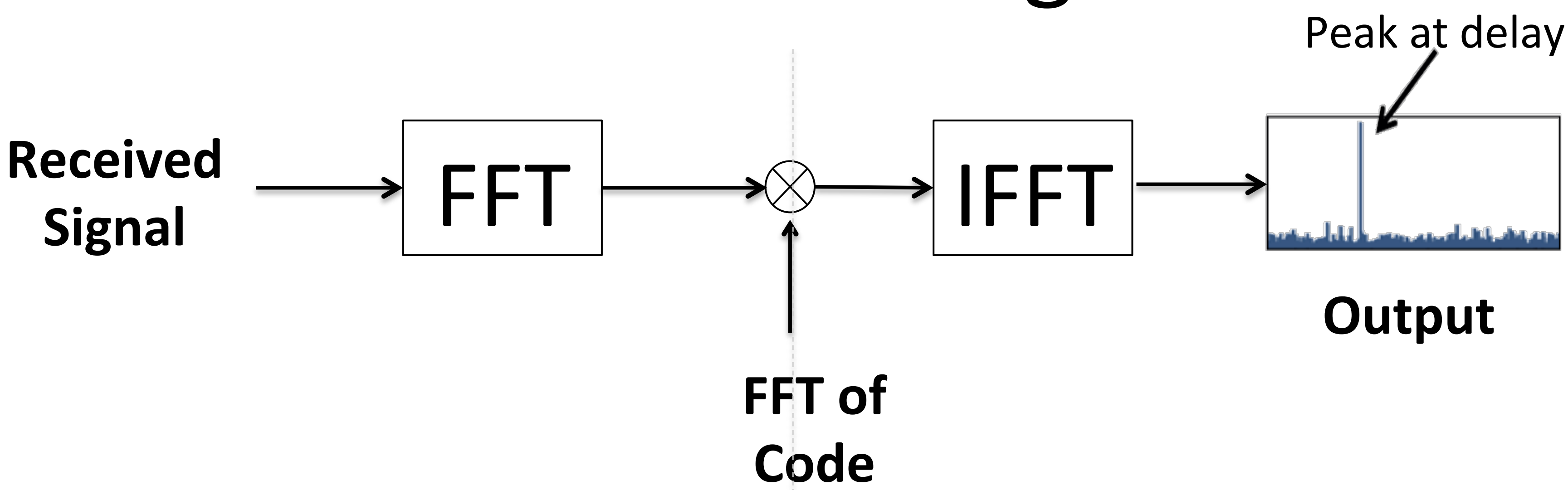
Subsampled FFT

Sparse IFFT

Need only few samples of FFT output

Sub-samples its input

Our GPS Design



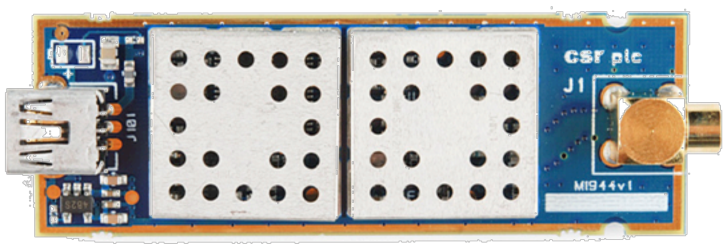
Subsampled FFT

Sparse IFFT

Aliasing input → FFT → Subsampling → IFFT → Aliasing output

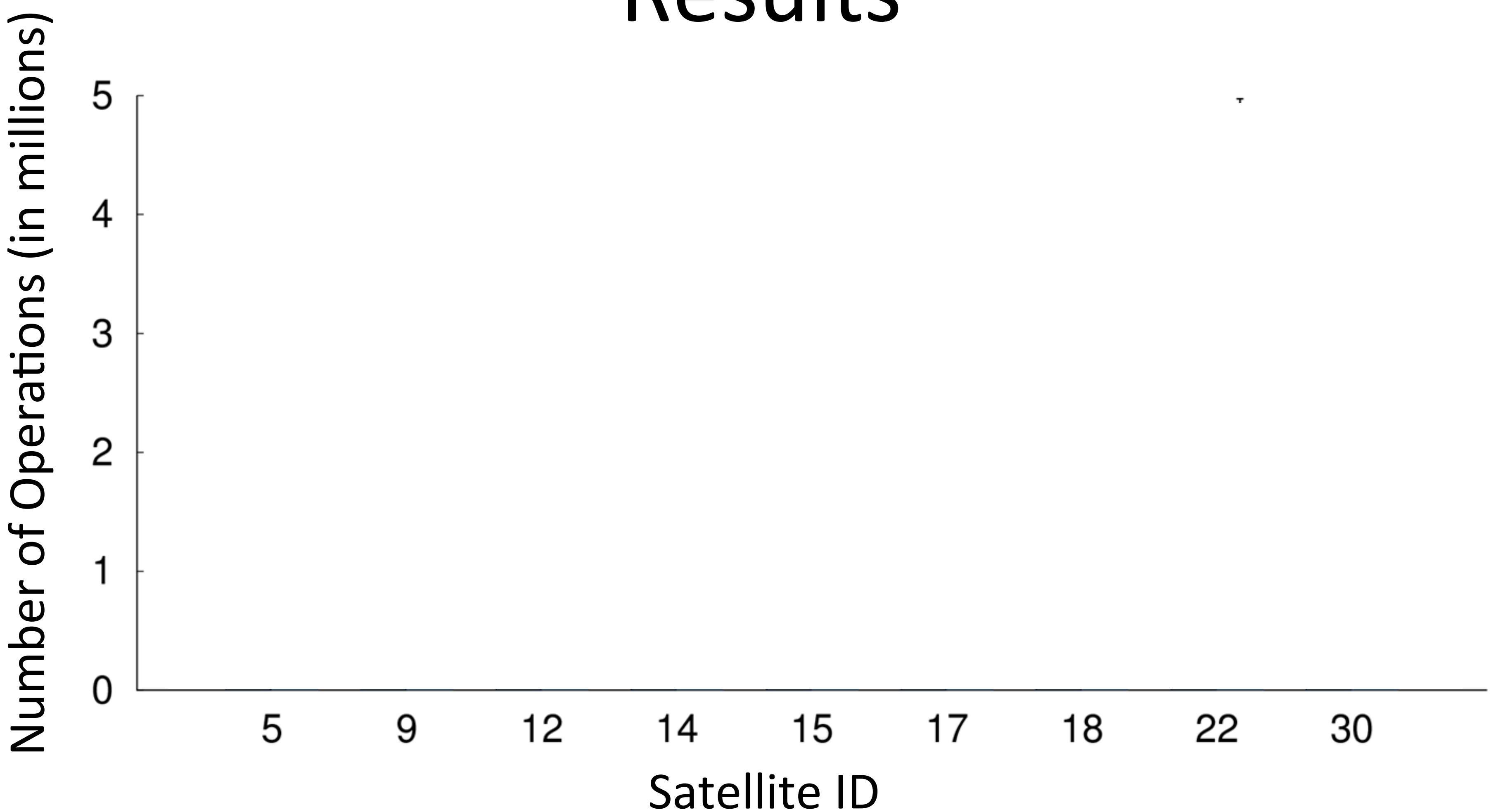
Lowest complexity GPS algorithm that maintains performance guarantees

Experiments



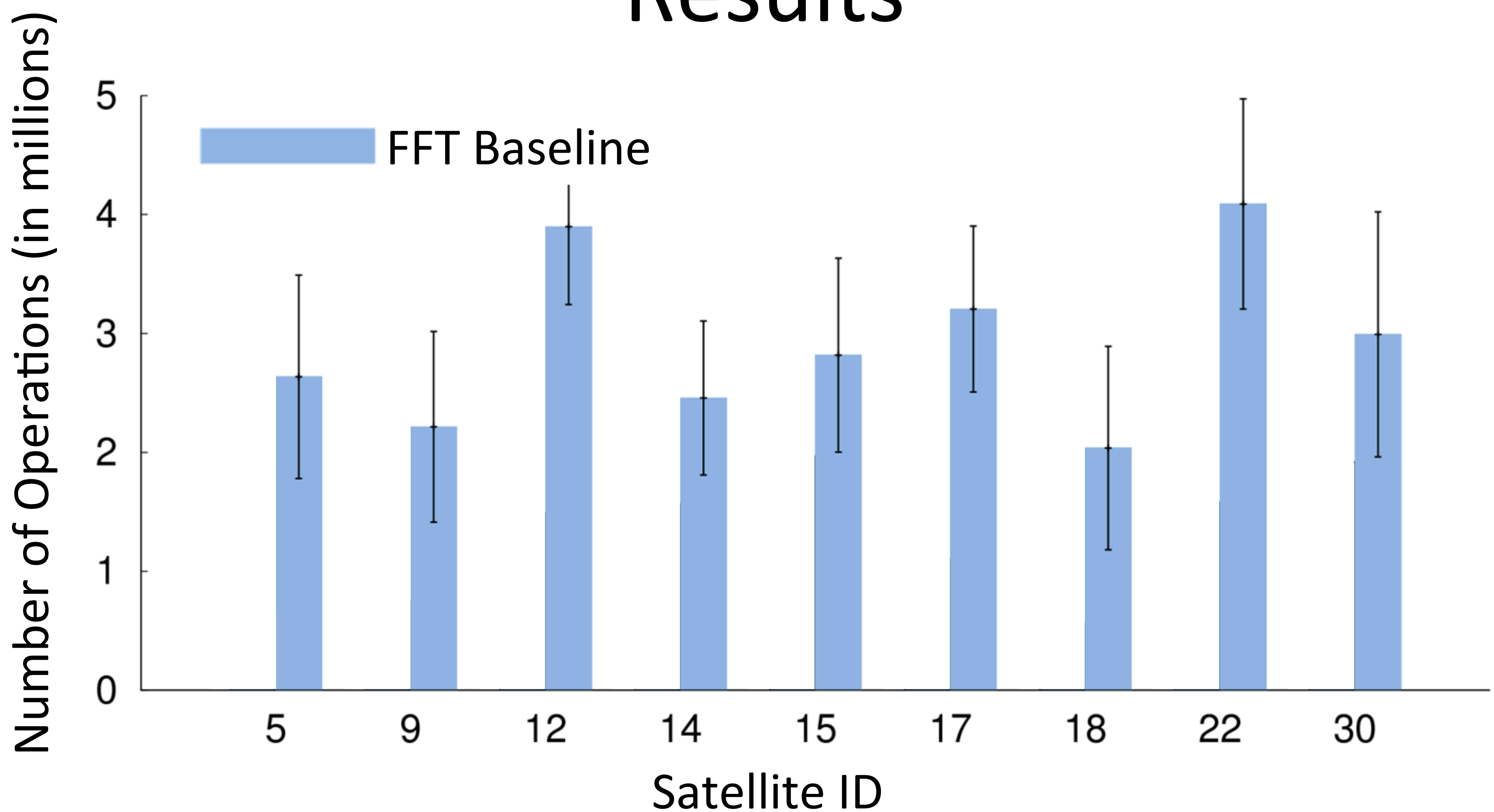
- Run over real satellite signals from various locations in the Boston area.

Results

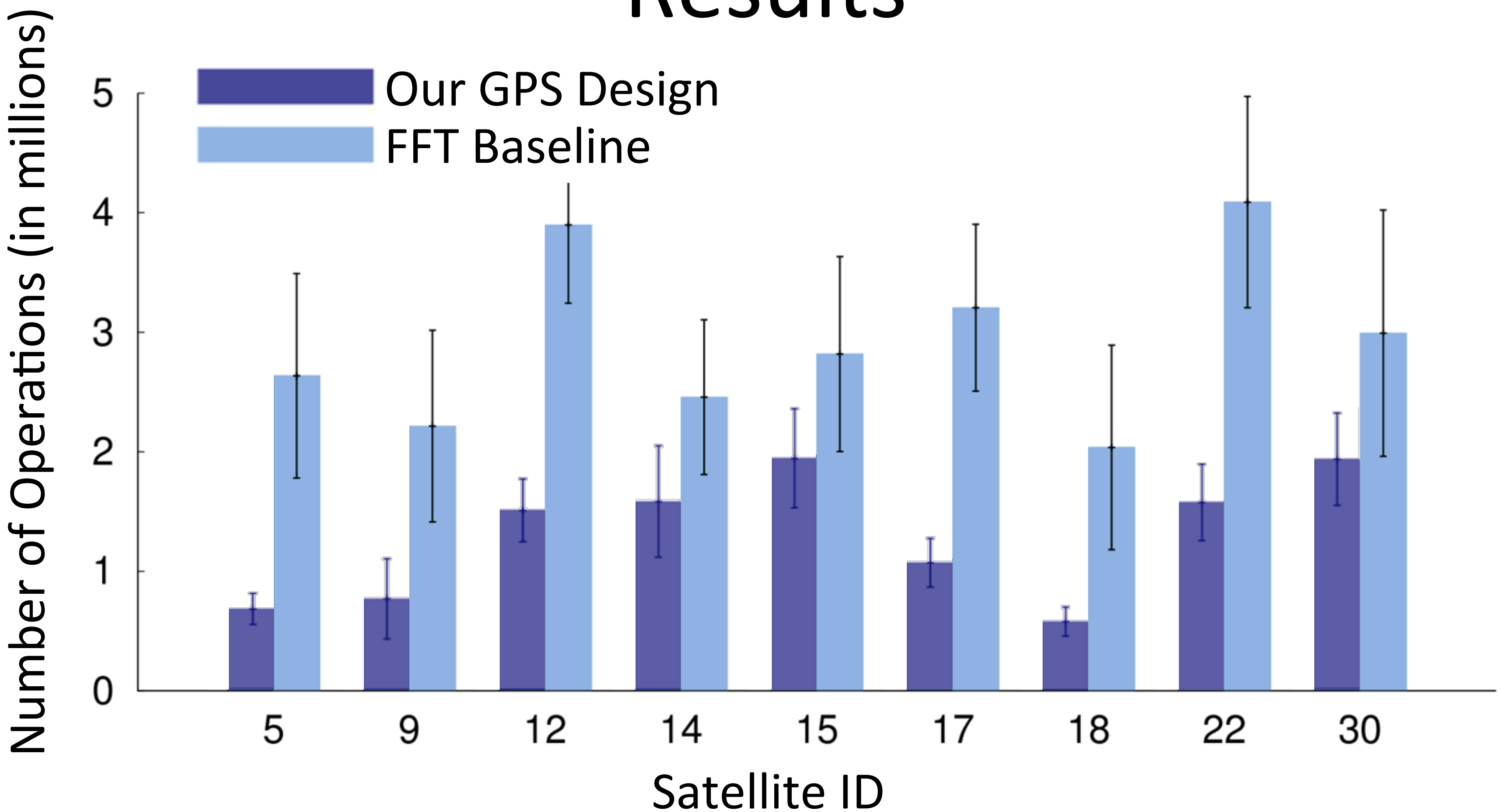


Number of Operations (in millions)
5
4
3
2
1
0
5
9
12
14
15
17
18
22
30
Satellite ID

Results


FFT Baseline
Number of Operations (in millions)
0
1
2
3
4
5
5
9
12
14
15
17
18
22
30
Satellite ID

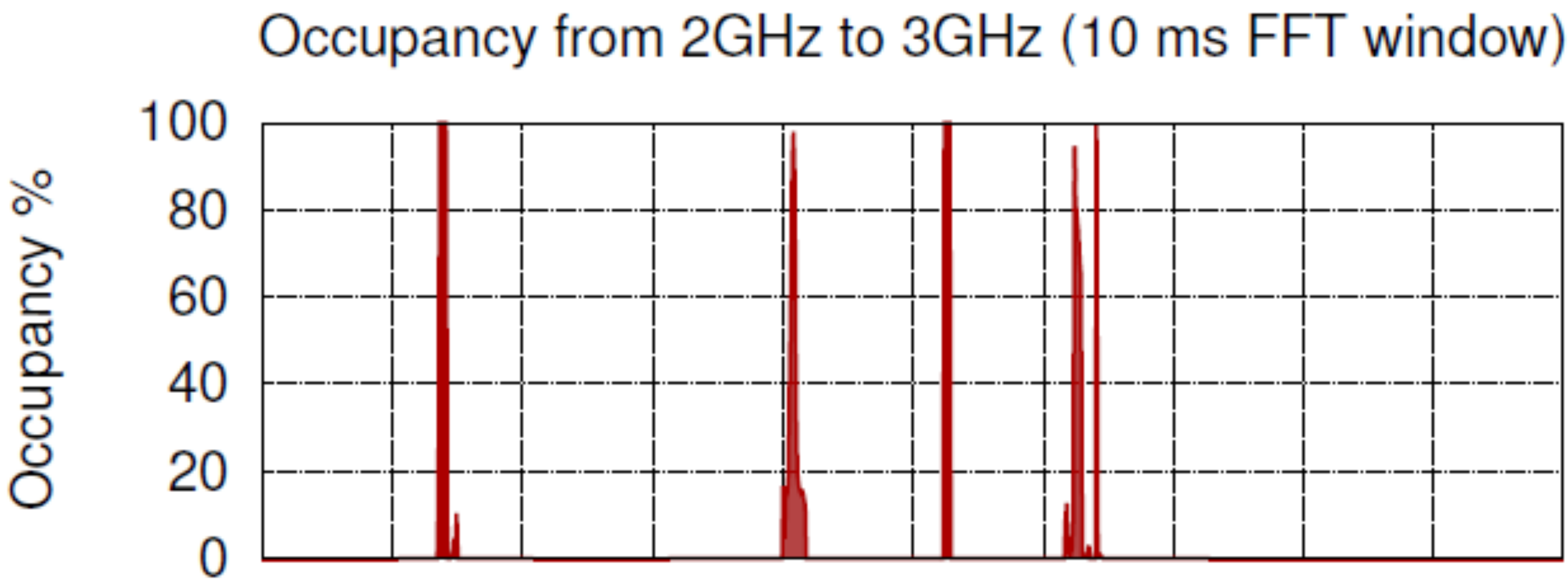
Results



- Reduce computations for all the satellites
- Min reduction is 35% and can go all the way to 75%

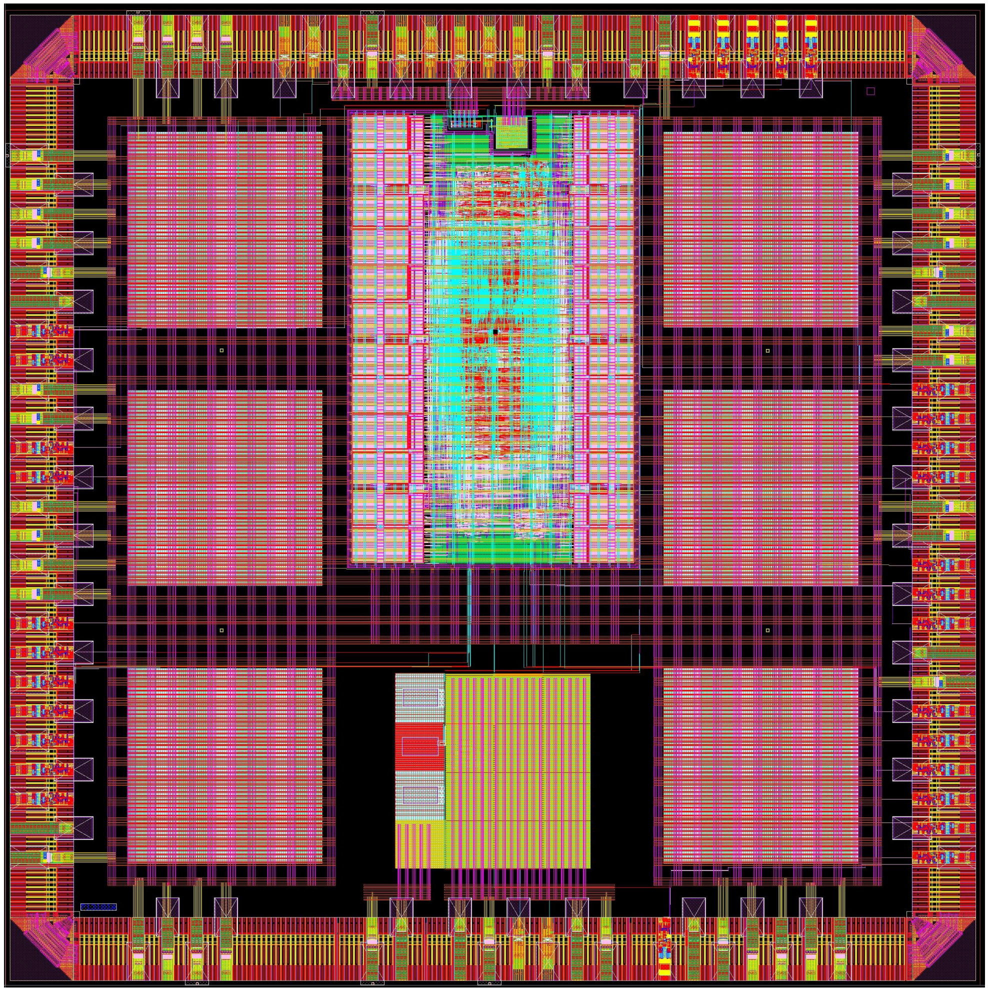
Realtime GHz Spectrum Sensing

Cambridge, MA January 15 2013



3 ADCs with a combined digital Bandwidth of 150 MHz can acquire a GHz

SFFT chip [O. Abari, E. Hamed, H. Hassanieh, A. Agarwal, D. Katabi, A. P. Chandrakasan, and V. Stojanovic, ISSCC'14]



Handles $n=3^6 2^{10} = 746946$ and sparsity up to 750

Conclusions

- O(k log n) times achievable for the k-sparse case
- O(k log n log(n/k)) achievable for the L2/L2 guarantee
- Better sample bounds, especially for average case
- Applications

Further directions

- Higher dimensions
  - Algorithms extend to higher dimensions, but
    - HIKP'12 has $(\log n)^d$ term [GHIKPS'13, Rauh-Arce'13)
    - IKP'14 has d^d term
  - Better dependence on the dimension ?
- Uniform (as opposed to randomized) guarantee
  - Possible in compressive sensing (RIP property)
  - Analogs for Sparse Fourier Transform ?
    - Best known result: $O(k^2 \log^c n)$ [Iwen'10]
- Model-based. E.g., what if coefficients cluster in blocks ?
  - In compressive sensing one can reduce number of measurements [Eldar-Mishali'09, Baraniuk-Cevher-Duarte'Hegde'09]
  - Improving Sparse Fourier Transform ?
    - One block case: [Plonka-Wannenwetsch'15]
- Off-grid frequencies
  - ~ $k \log^3 k$ [Boufounos-Cevher-Gilbert-Li-Strauss'12]

References

- Bibliography:
  - http://groups.csail.mit.edu/netmit/sFFT/paper.html
- Course: Algorithms and Signal Processing, Lectures 1..6

https://stellar.mit.edu/S/course/6/fa14/6.893/materials.html

- Survey: Recent developments in the sparse Fourier transform: A compressed Fourier transform for big data, Signal Processing Magazine, 2014.