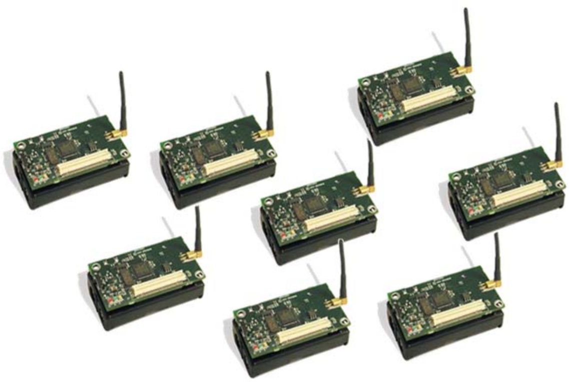
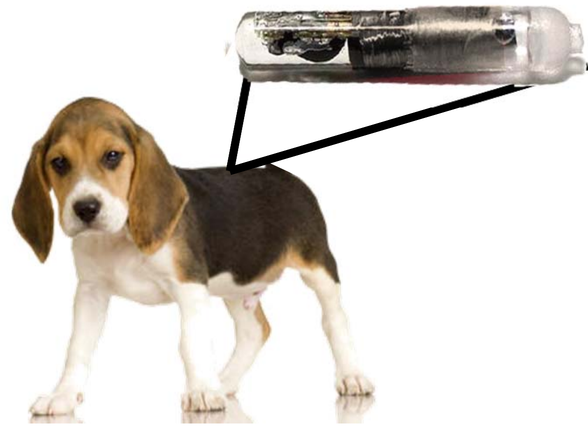
Faster GPS via the Sparse Fourier Transform

Haitham Hassanieh
Fadel Adib
Dina Katabi
Piotr Indyk

Massachusetts Institute of Technology

GPS Is Widely Used





Faster GPS benefits many applications

How Do We Improve GPS?

⬇

Need to Improve GPS Synchronization

GPS Synchronization

Synchronization is locking onto a satellite's signal

- Consumes 30%-75% of GPS receiver's power [ORG447X datasheet, Venus 6 datasheet]

GPS signals are very weak, less than -20dB SNR

100s of millions of multiplications

[Team, Kaplan]

Goal

Faster Synchronization Algorithm

Reduce number of operations

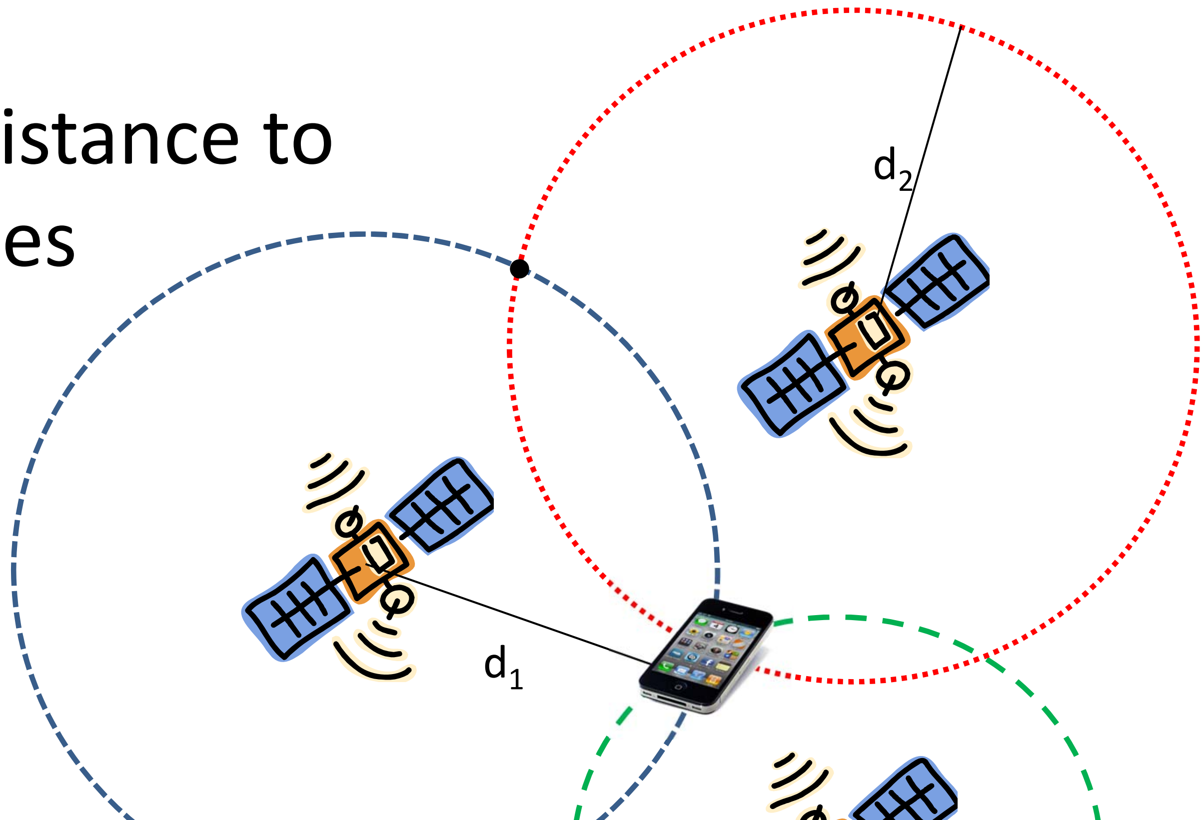
Reduction in power consumption and delay

Rest of this Talk

- GPS Primer
- Our GPS Synchronization Algorithm
- Empirical Results

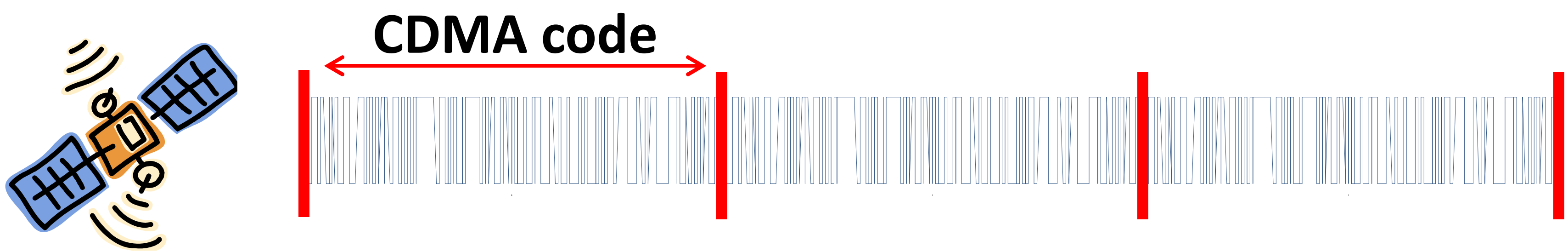
How Does GPS Work?

Compute the distance to the GPS satellites



distance = propagation delay × speed of light

How to Compute the Propagation Delay?



Satellite Transmits CDMA code

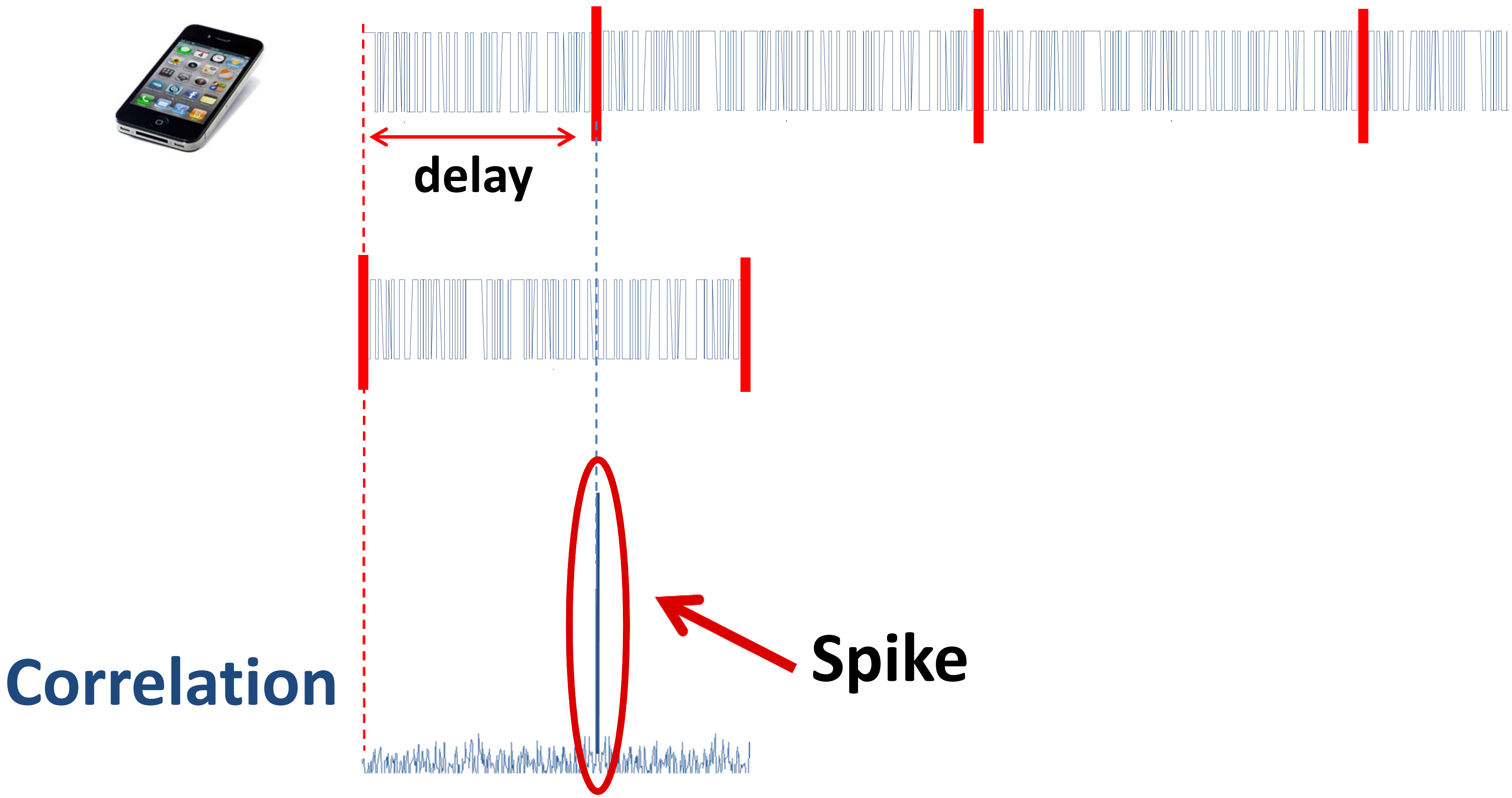
How to Compute the Propagation Delay?

Code arrives shifted by propagation delay

How to Compute the Propagation Delay?

Receiver knows the code and when the satellite starts transmitting

How to Compute the Propagation Delay?



Spike determines the delay

GPS Synchronization is a convolution with CDMA code

Convolution in Time ⟺ **Multiplication in Frequency**

$O(n^2)$ $O(n \log n)$

State of the art GPS synchronization algorithm: $O(n \log n)$

Rest of this Talk

- GPS Primer
- Our GPS Synchronization Algorithm
- Empirical Results

QuickSync

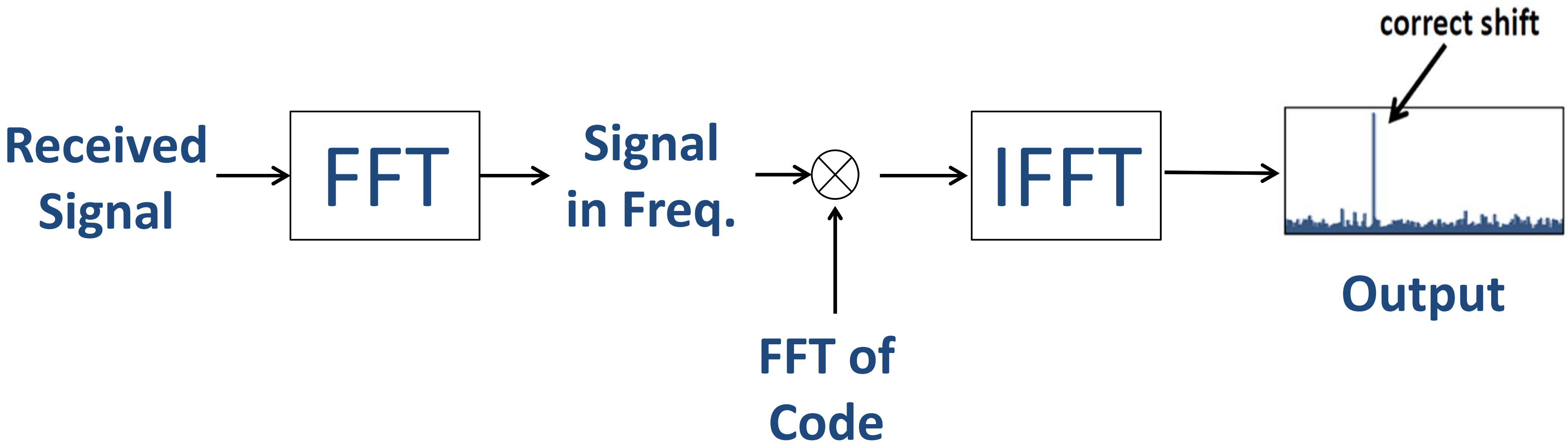
- Faster GPS synchronization algorithm

- Complexity:
  - $O\left(n\sqrt{\log n}\right)$ for any SNR
  - $O(n)$ when noise is bounded by $O(n/\log^2 n)$

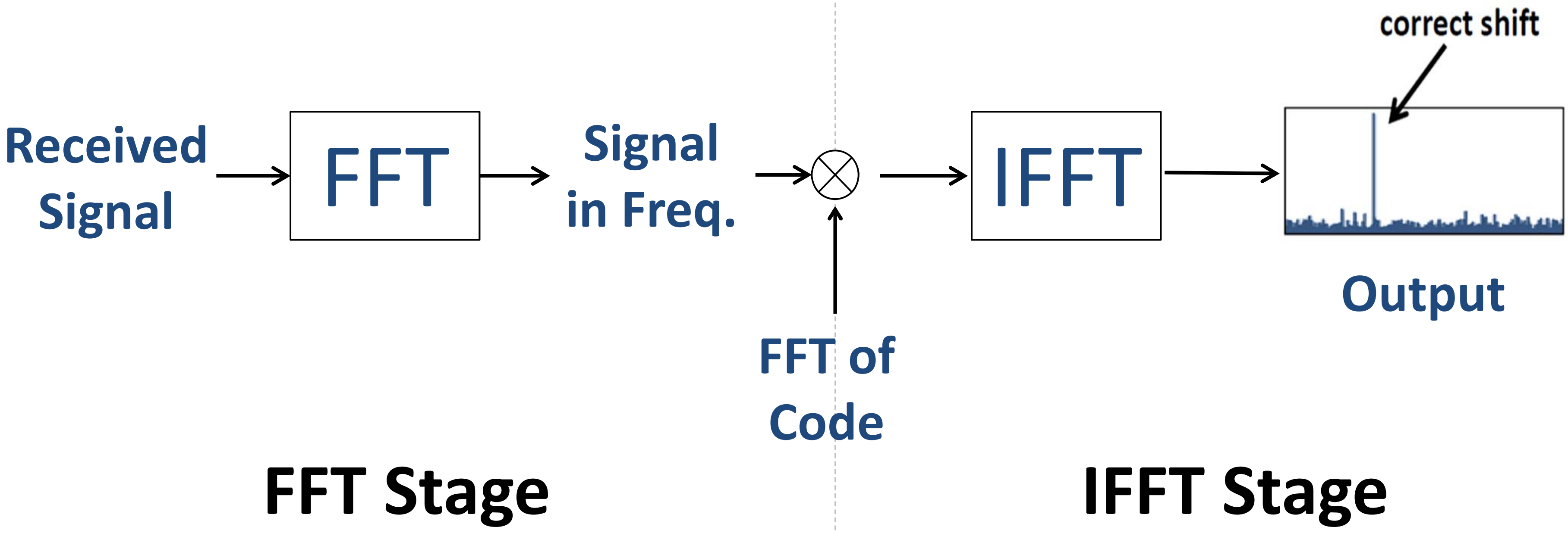
- Empirical Results:
  - Evaluated on real GPS signals
  - Improves performance by 2.2x

How can we make GPS synchronization faster than FFT-Based synchronization?

FFT-Based GPS Synchronization



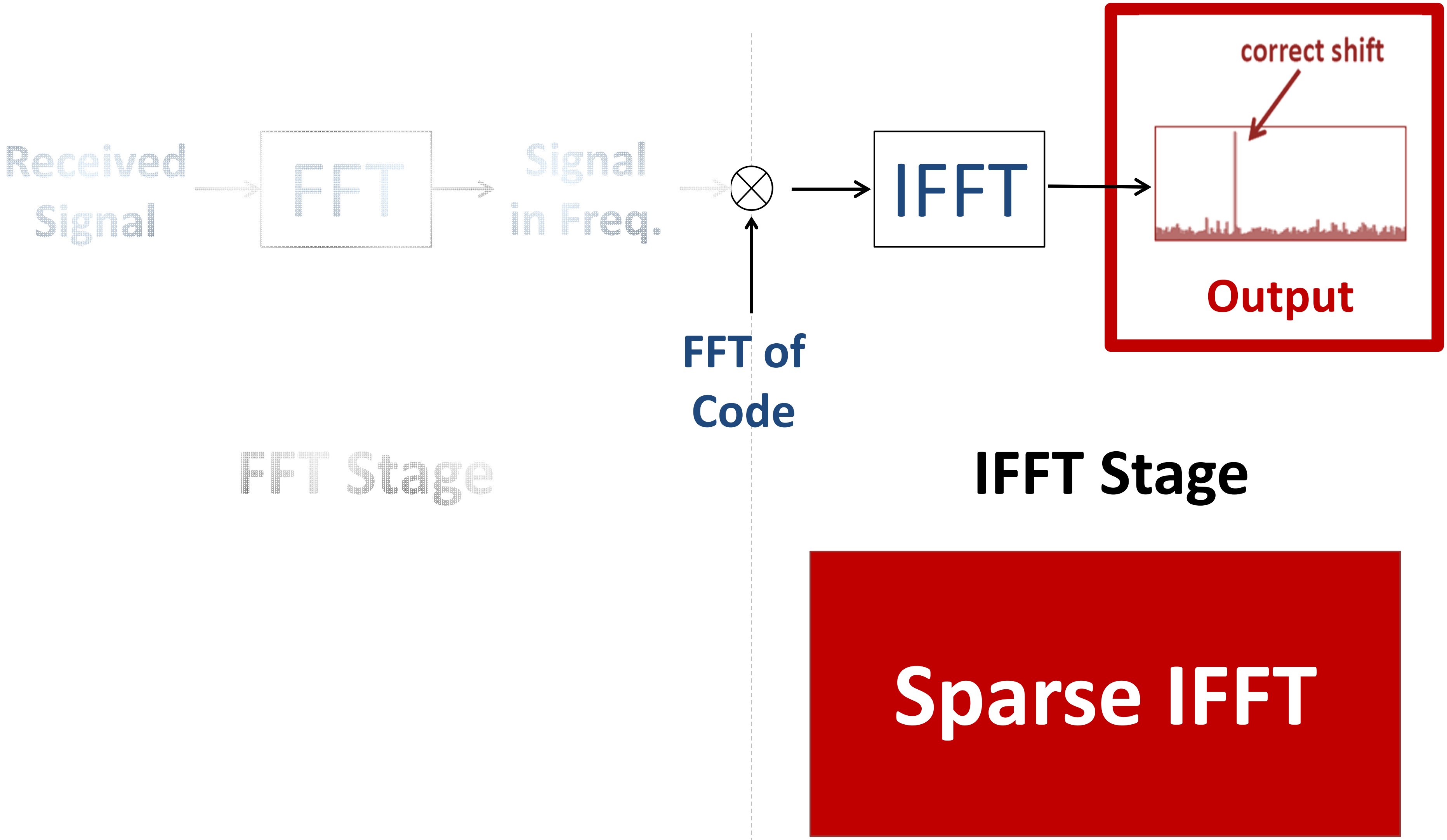
FFT-Based GPS Synchronization



Each stage takes $O(n \log n)$

→ need to reduce complexity of both stages

FFT-Based GPS Synchronization



QuickSync

A Sparse IFFT algorithm for GPS

- Exactly One Spike → Simpler algorithm
- Extends to the FFT-stage

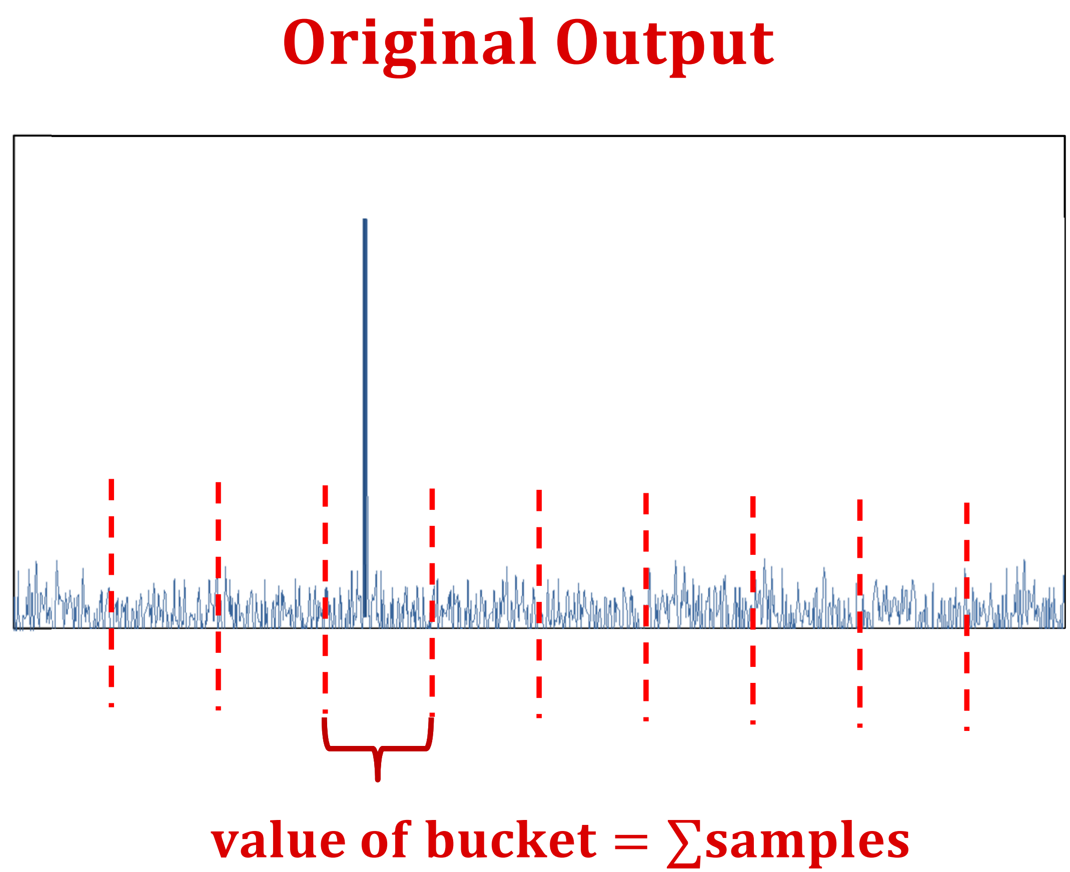
QuickSync's Sparse IFFT

1- Bucketize

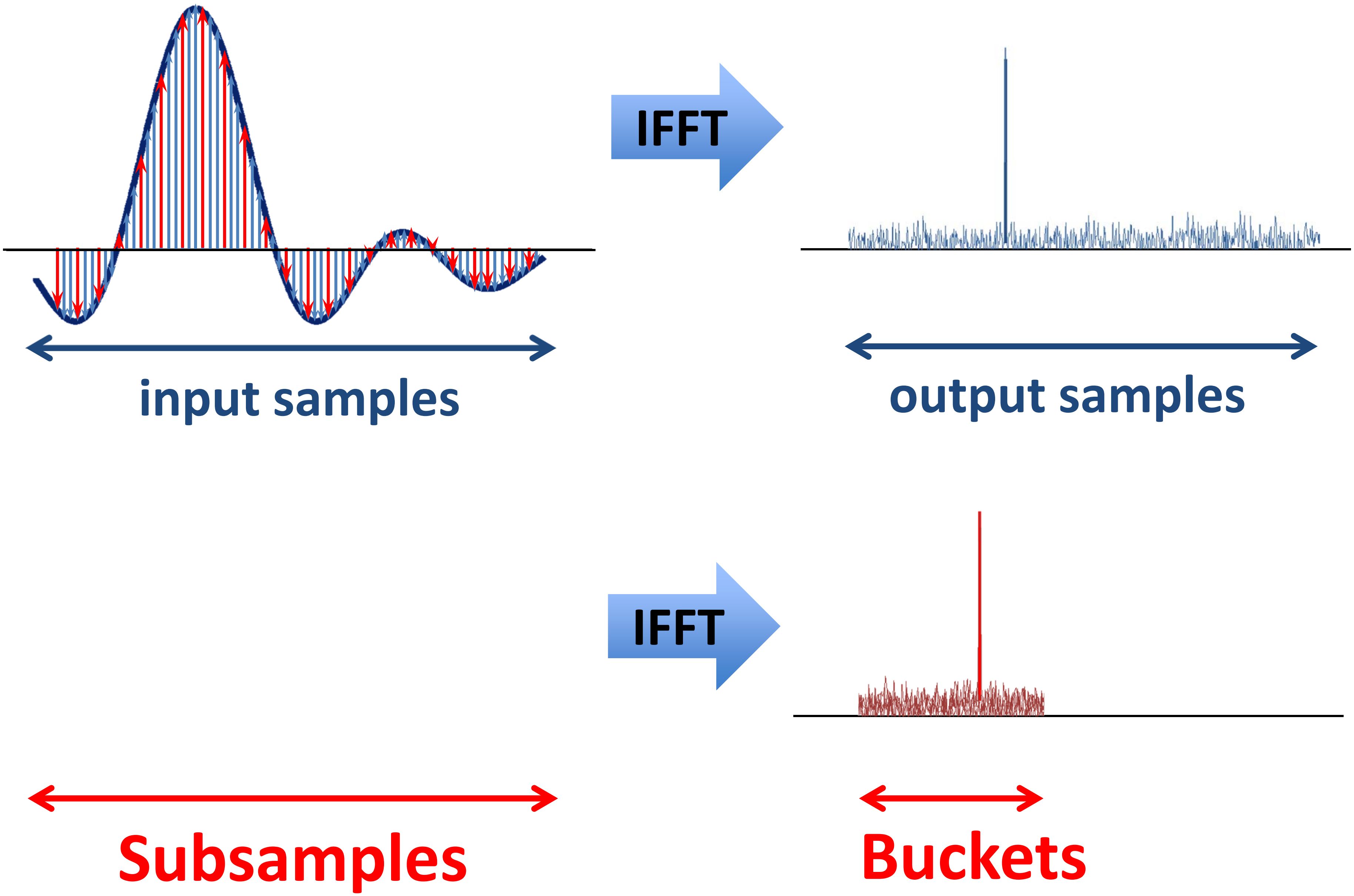
Divide output into a few buckets

2- Estimate

Estimate the largest coefficient in the largest bucket



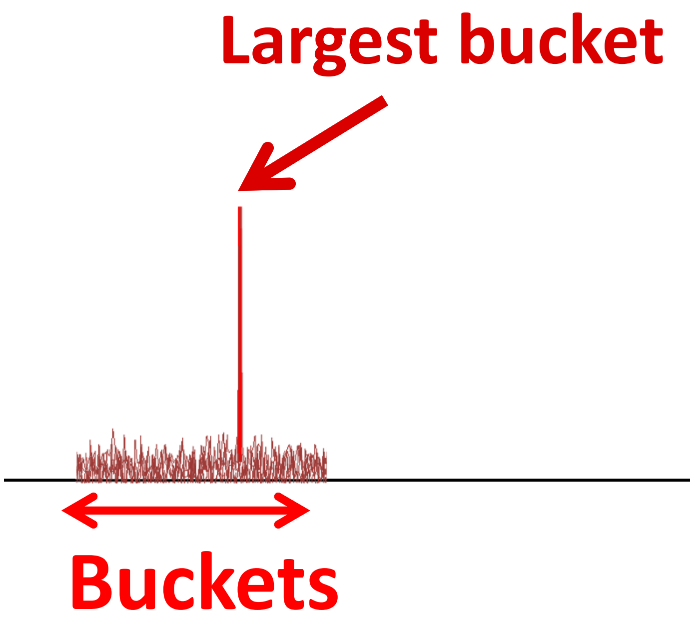
How to Bucketize Efficiently?



How to Estimate Efficiently?

- Keep largest bucket; ignore all the rest
- Out of the samples in the large bucket, which one is the spike?

The spike is the sample that has the maximum correlation



QuickSync's Sparse IFFT

- n is number of samples
- B buckets → n/B samples per buckets

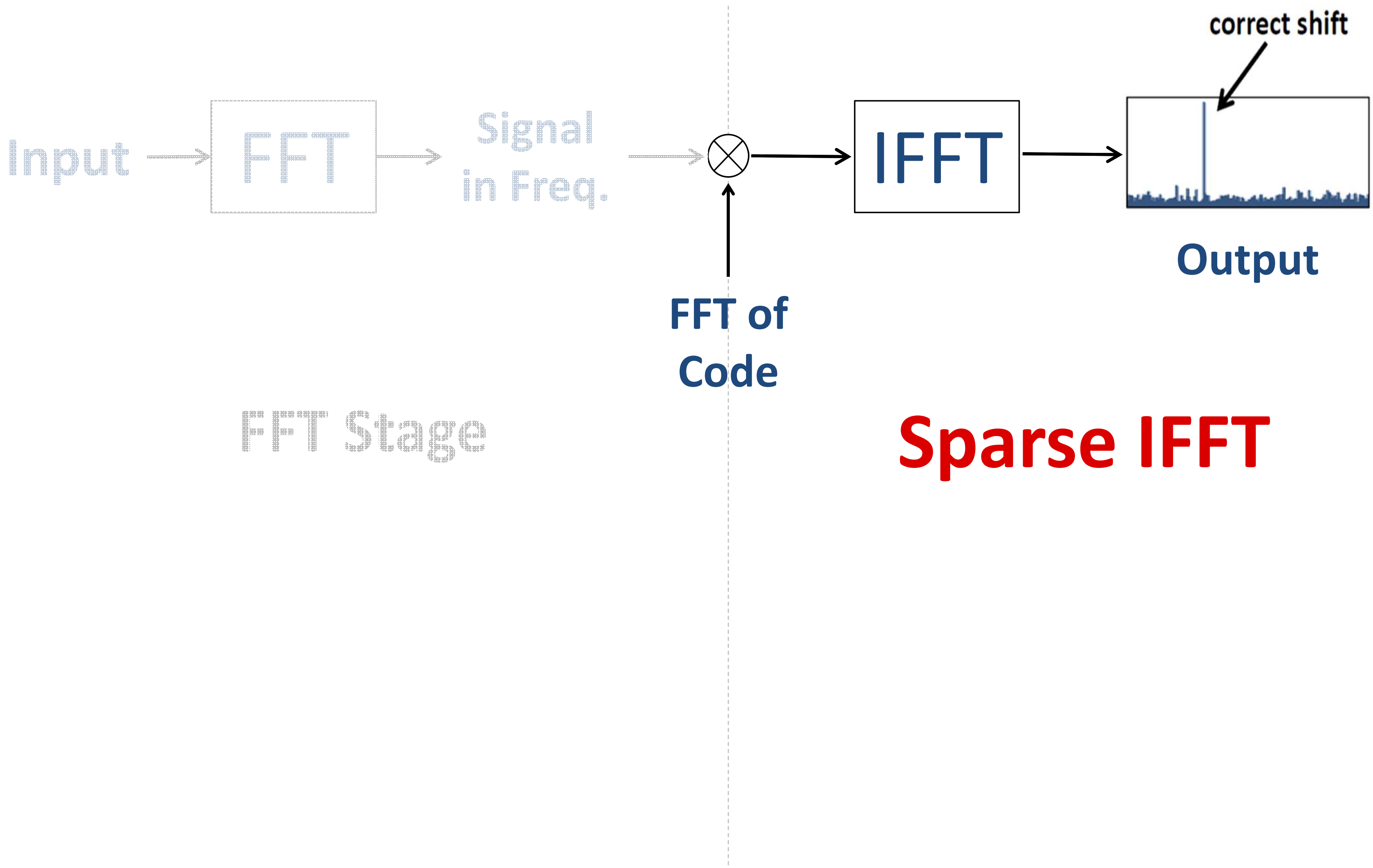
Bucketization: $B \log B$

Estimation: $n/B \times B$

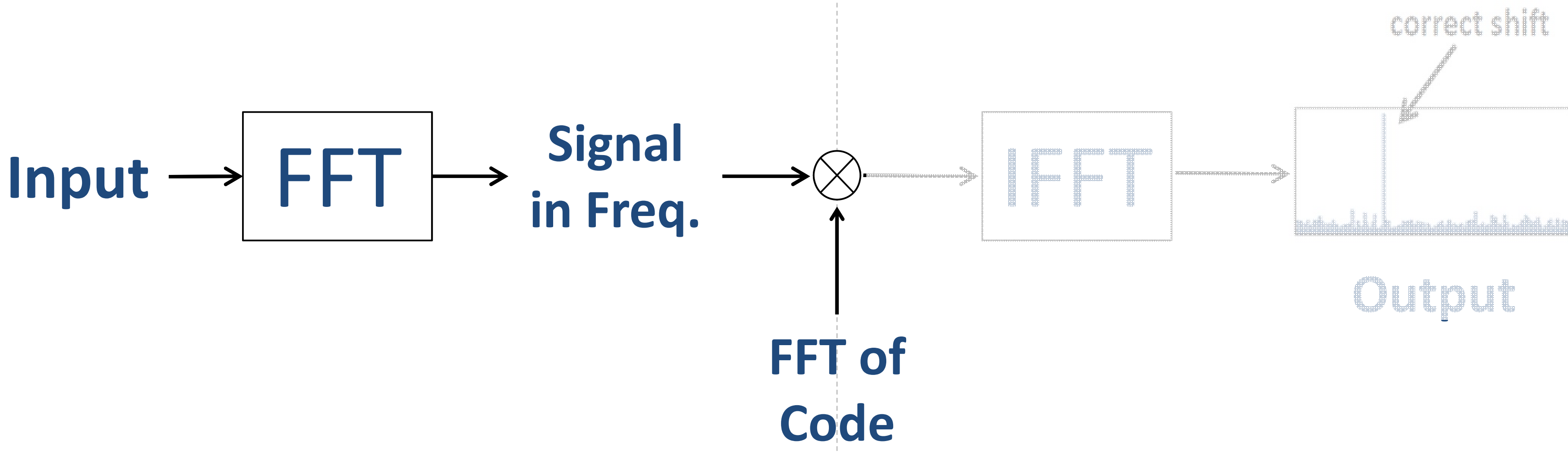
$B = n/\log n$ ⇒ $O(n)$

$B = n/\sqrt{\log n}$ ⇒ $O(n\sqrt{\log n})$

QuickSync Synchronization



QuickSync Synchronization



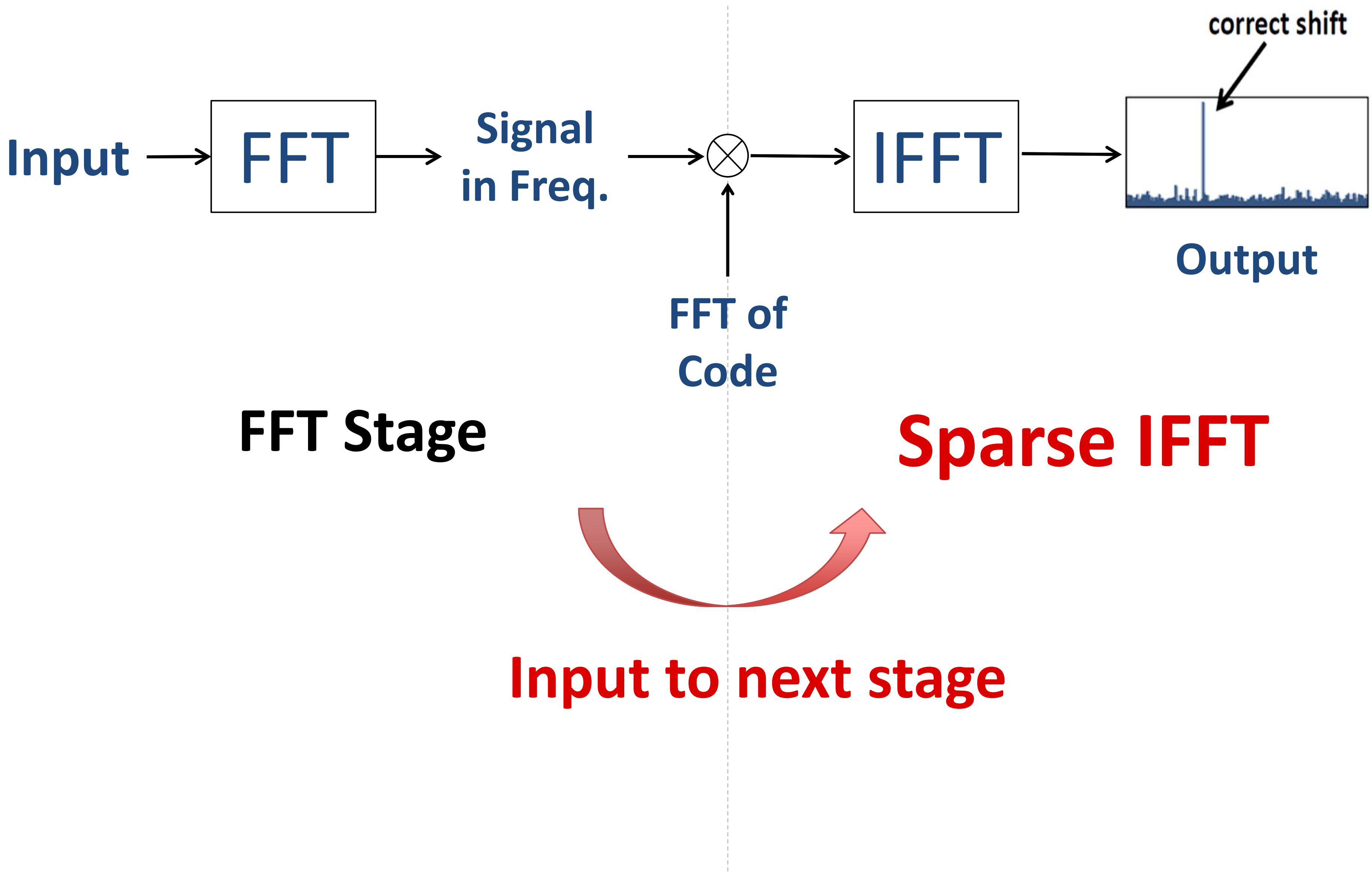
FFT Stage

Sparse IFFT

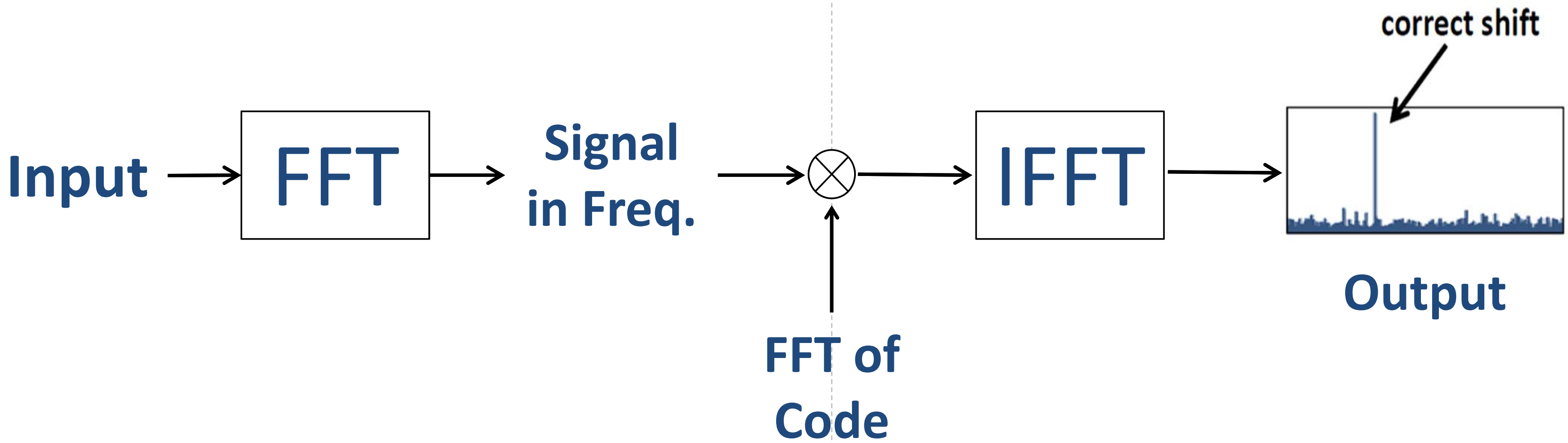
Output is not sparse

Cannot Use Sparse FFT

QuickSync Synchronization



QuickSync Synchronization



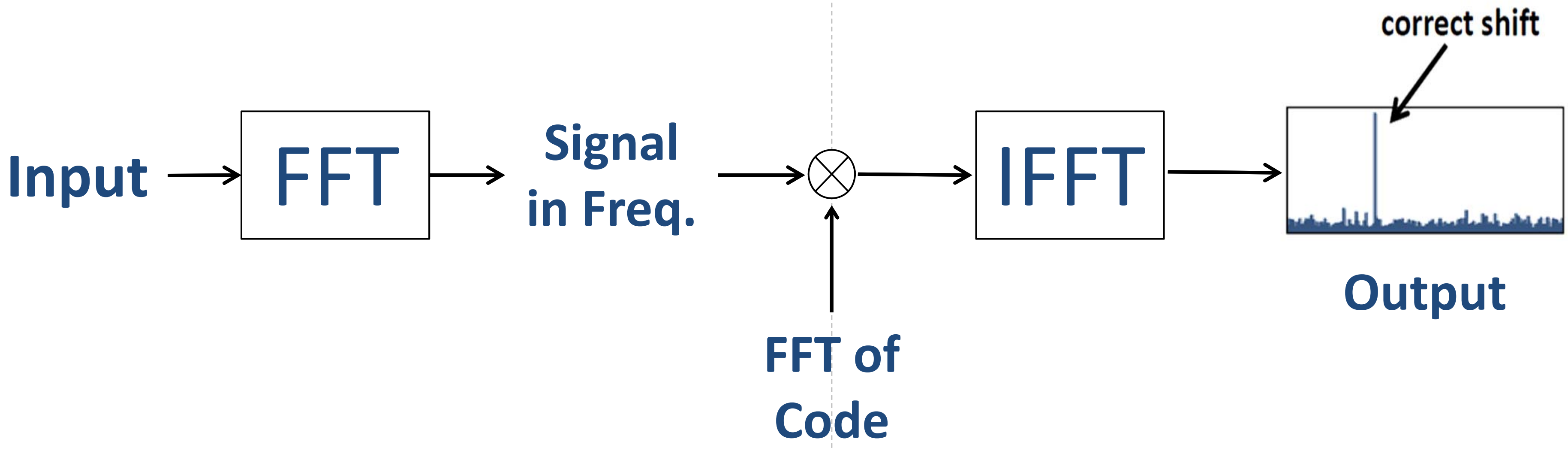
Subsampled FFT

Need only few samples of FFT output

Sparse IFFT

IFFT samples its input

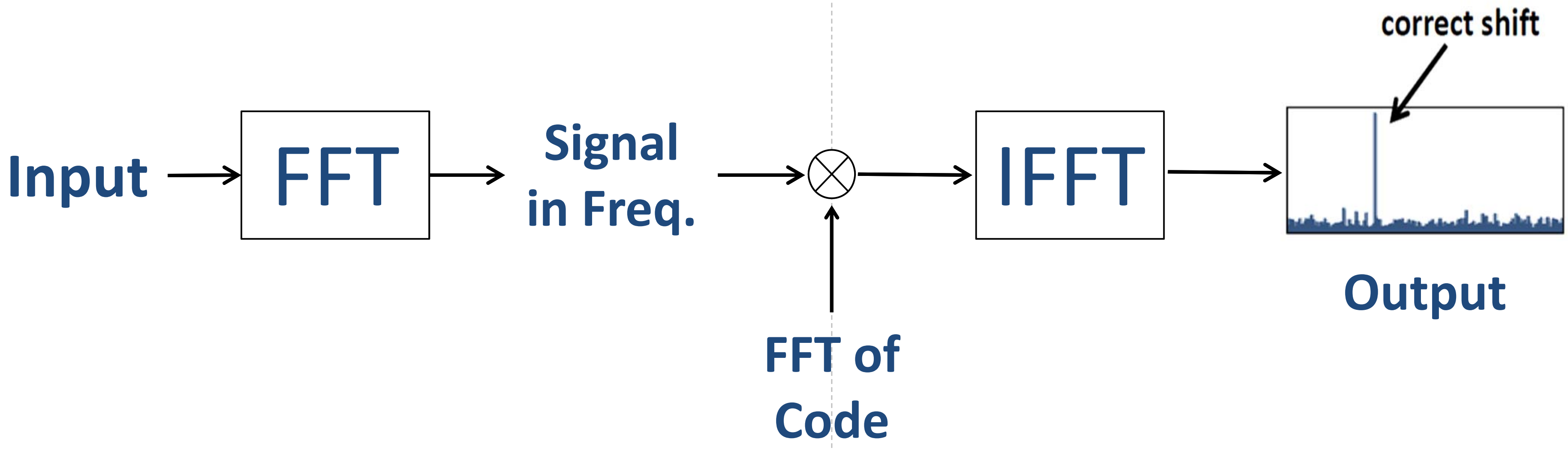
QuickSync Synchronization



Subsampled FFT **Sparse IFFT**

FFT and IFFT are dual of each other

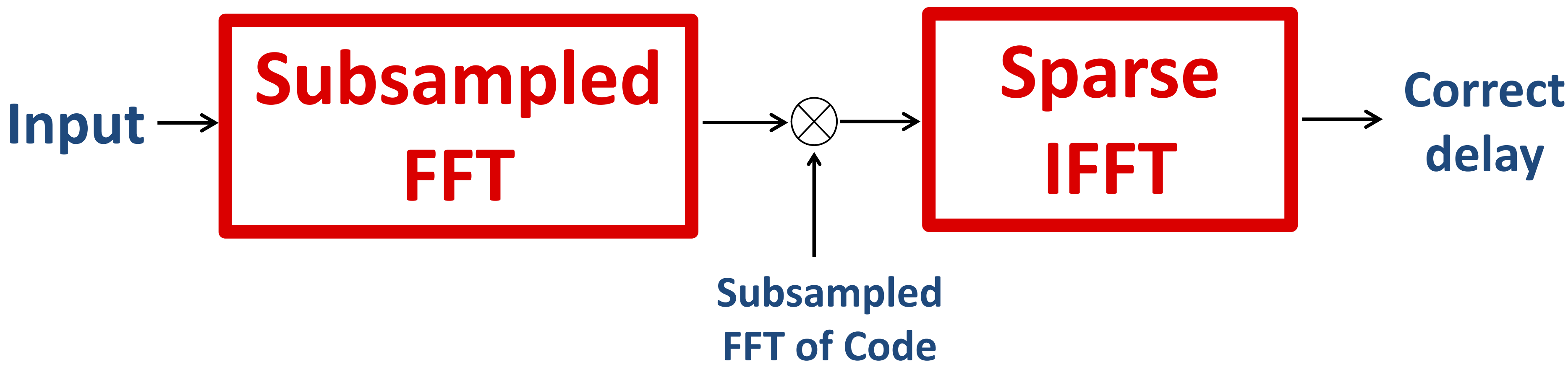
QuickSync Synchronization



Subsampled FFT | **Sparse IFFT**

Bucketization **FFT** Subsampling **IFFT** Bucketization

QuickSync Synchronization



Theorem:

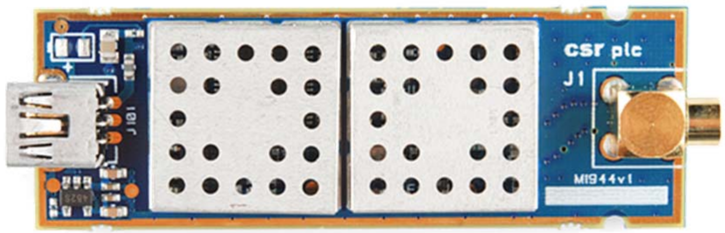
For any SNR QuickSync achieves the same accuracy as FFT-Based synchronization and has a complexity of $O(n\sqrt{\log n})$ *where* n *is the number of samples in the code*

When noise is bounded by $O(n/\log^2 n)$*, QuickSync has* $O(n)$ *complexity.*

Rest of this Talk

- GPS Primer
- Our GPS Synchronization Algorithm
- Empirical Results

Setup



SciGe GN3S Sampler

USRP Software radios

- Traces are collected both US and Europe
- Different locations: urban – suburban
- Different weather conditions: cloudy – clear

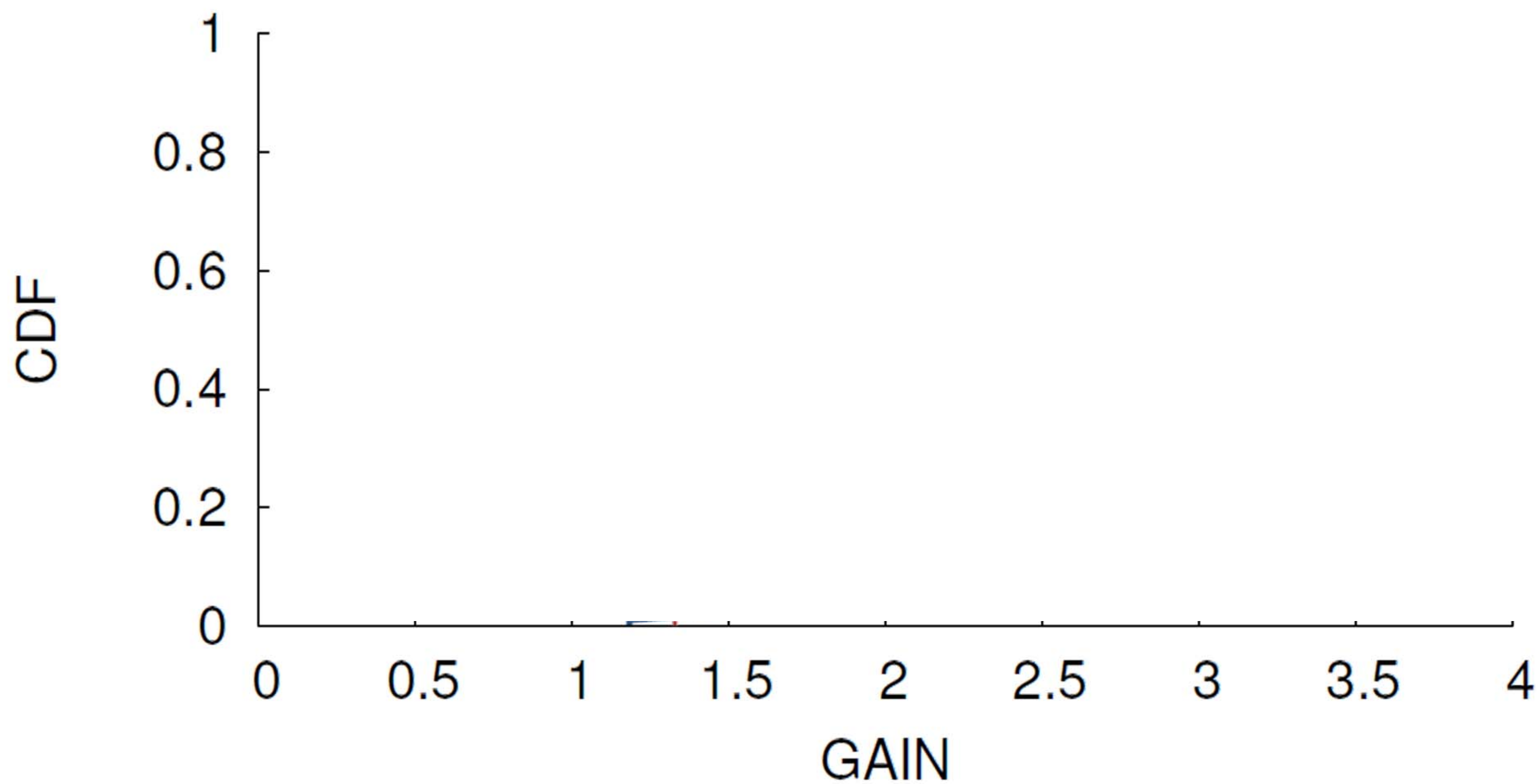
Compared Schemes

- QuickSync Synchronization
- FFT-Based Synchronization

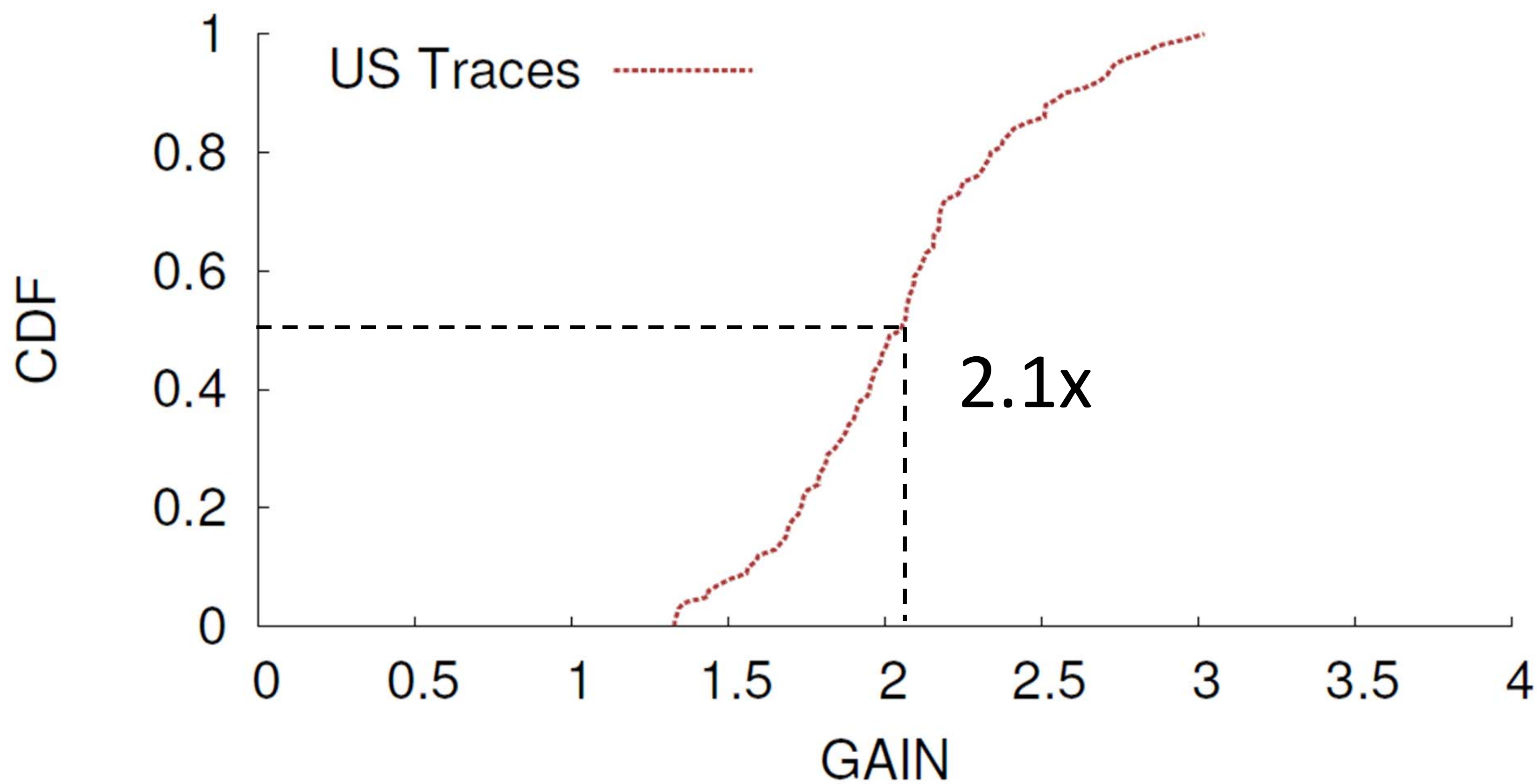
Metrics

- $$\text{Multiplication Gain} = \frac{\text{Multiplications of baseline}}{\text{Multiplications of QuickSync}}$$
- $$\text{FLOPS Gain} = \frac{\text{FLOPS of baseline}}{\text{FLOPS of QuickSync}}$$

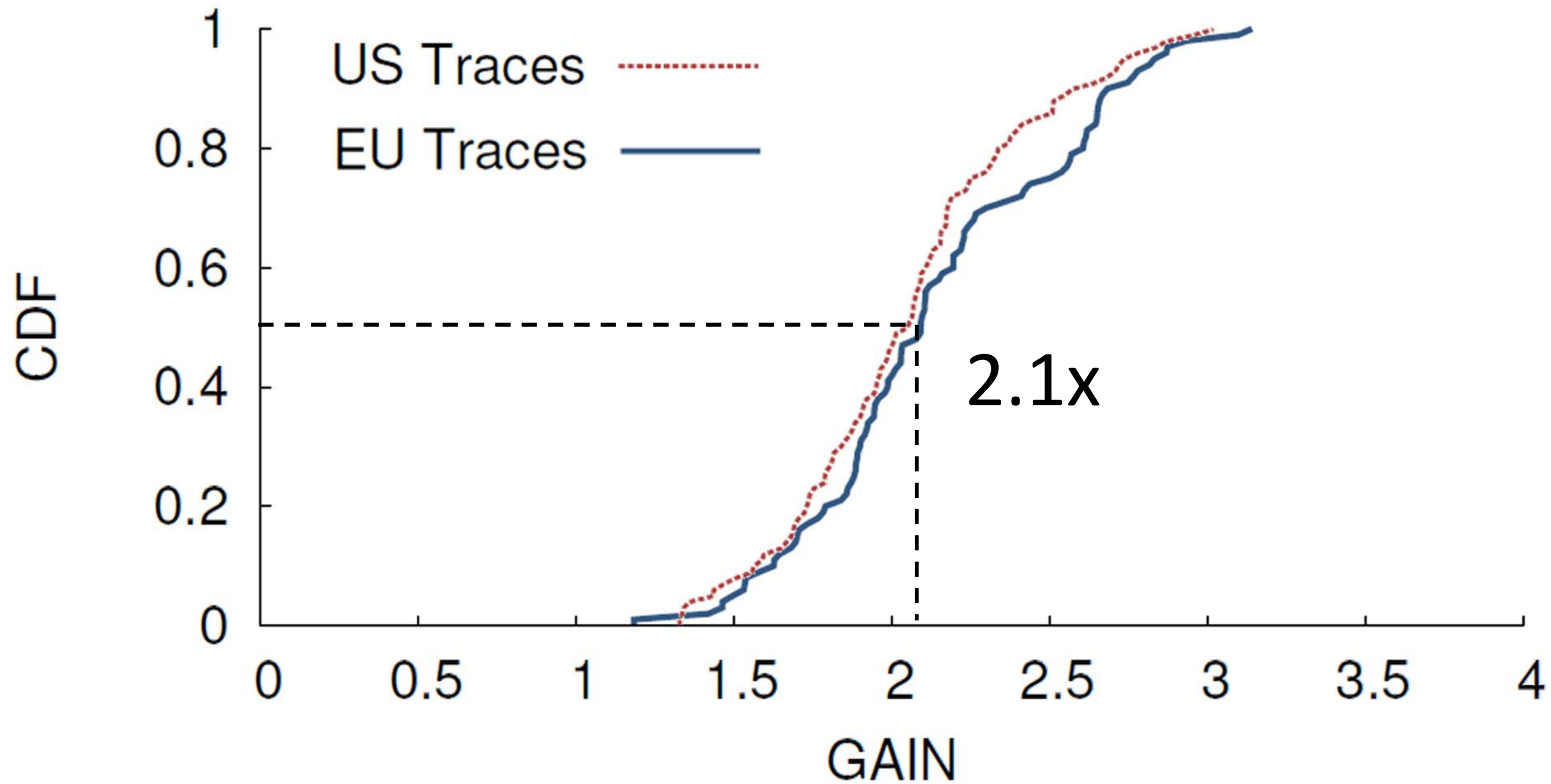
Multiplication Gain



Multiplication Gain

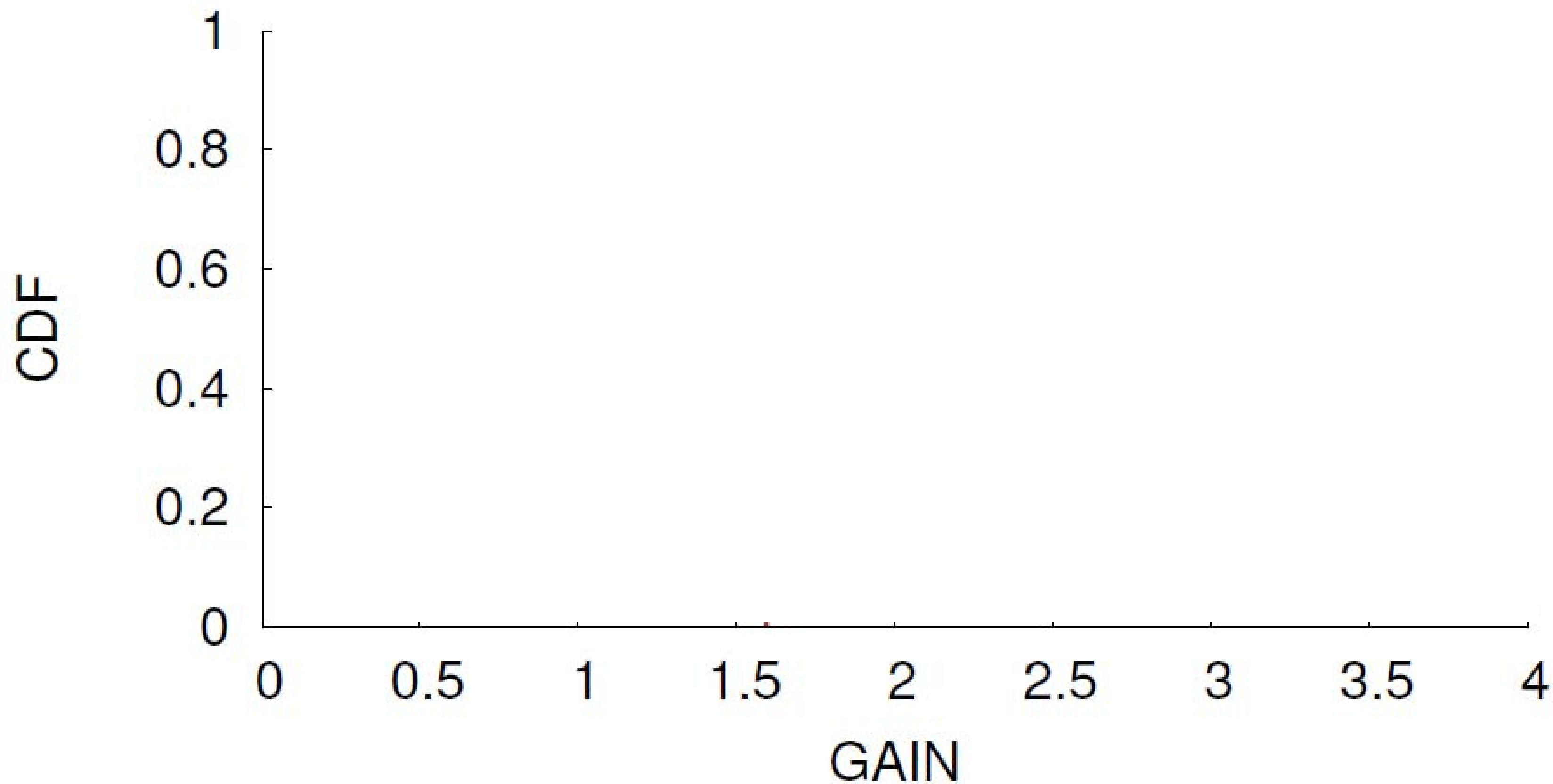


Multiplication Gain

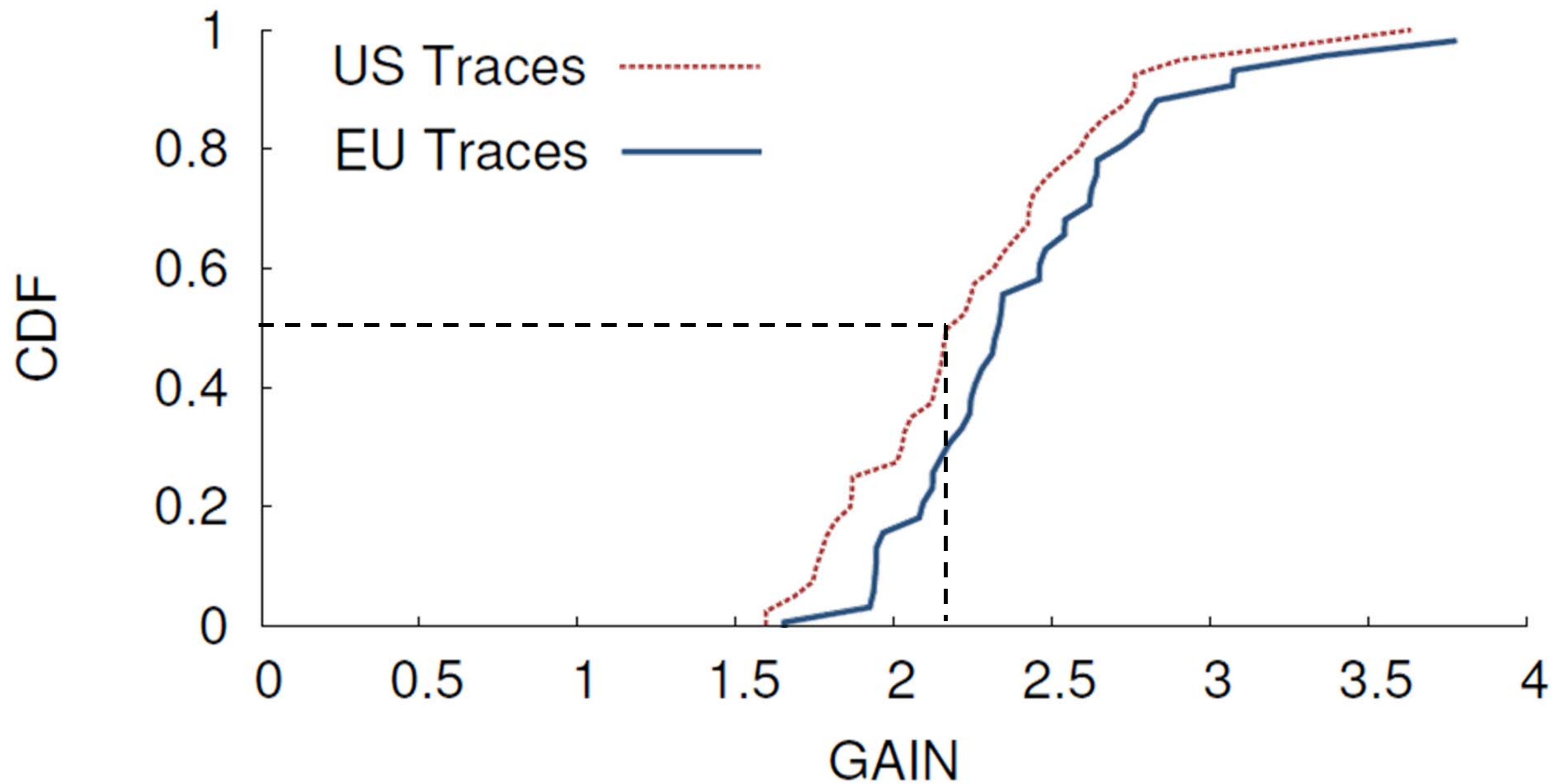


QuickSync provides an average gain of 2.1x

FLOPS Gain



FLOPS Gain



QuickSync provides an average gain of 2.2 ×

Related Work

- Past work on GPS [NC91, SA08, RZL11]
  - QuickSync presents the faster algorithm

- Sparse FFT Algorithms [Man02, GMS05, HKIP12a, HKIP12b]
  - QuickSync's bucketization leverages duality
    → reduces the complexity of both stages in GPS

Conclusion

- Fastest GPS synchronization algorithm
  - $O\left(n\sqrt{\log n}\right)$ for any SNR
  - $O(n)$ for moderately low SNR
- Empirical results show an average 2x gain
- Can we do better?
  - $O\left(n^{2/3}\right)$ for constant noise