PARALLEL EVENT DETECTION IN SENSOR DATA

Marc P. Greenbaum

CSE 633

Spring 2017

University at Buffalo The State University of New York

University at Buffalo The State University of New York

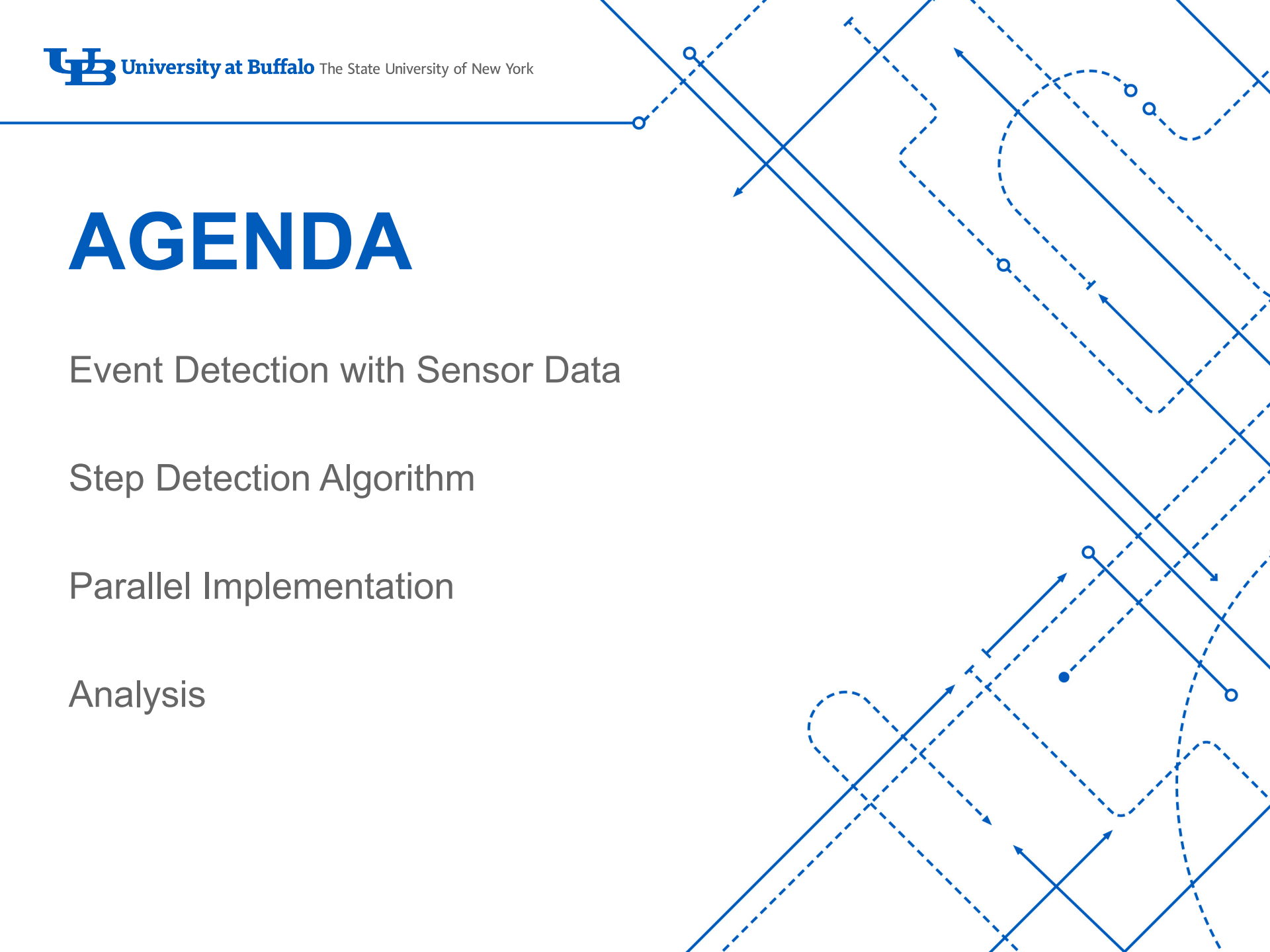
AGENDA

Event Detection with Sensor Data

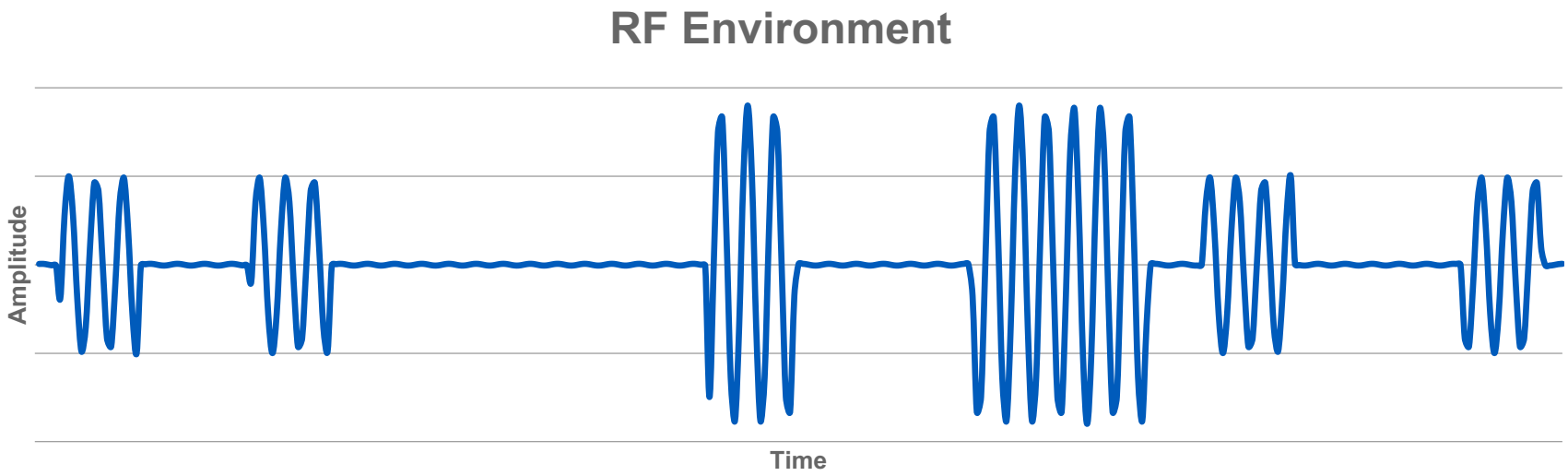
Step Detection Algorithm

Parallel Implementation

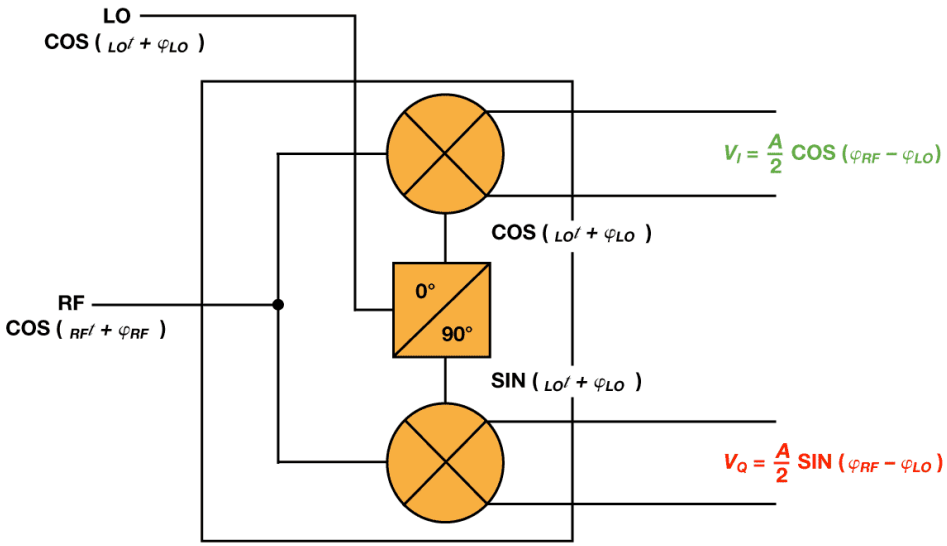
Analysis

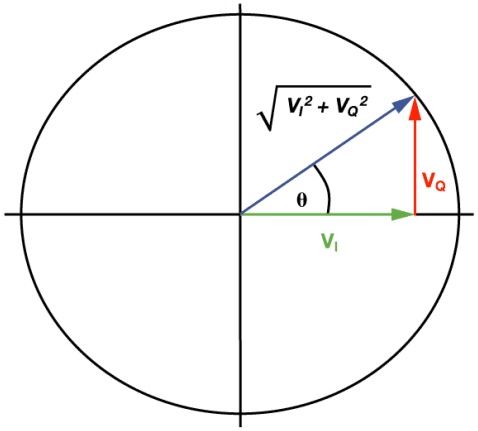


Radio Frequency Signals



In-phase and Quadrature Sampling





$$I = A\cos(\omega_{RF}t + \varphi_{RF}) \times \cos(\omega_{LO}t + \varphi_{LO}) = \frac{A}{2}\left[\cos(\omega_{RF}t - \omega_{LO}t + \varphi_{RF} - \varphi_{LO}) + \cos(\omega_{RF}t + \omega_{LO}t + \varphi_{RF} + \varphi_{LO}\right]$$

$$V_I = \frac{A}{2}\left[\cos(\varphi_{RF} - \varphi_{LO})\right]$$

Let $\omega_{RF} = \omega_{LO}$ difference term at dc

Sum term gets filtered

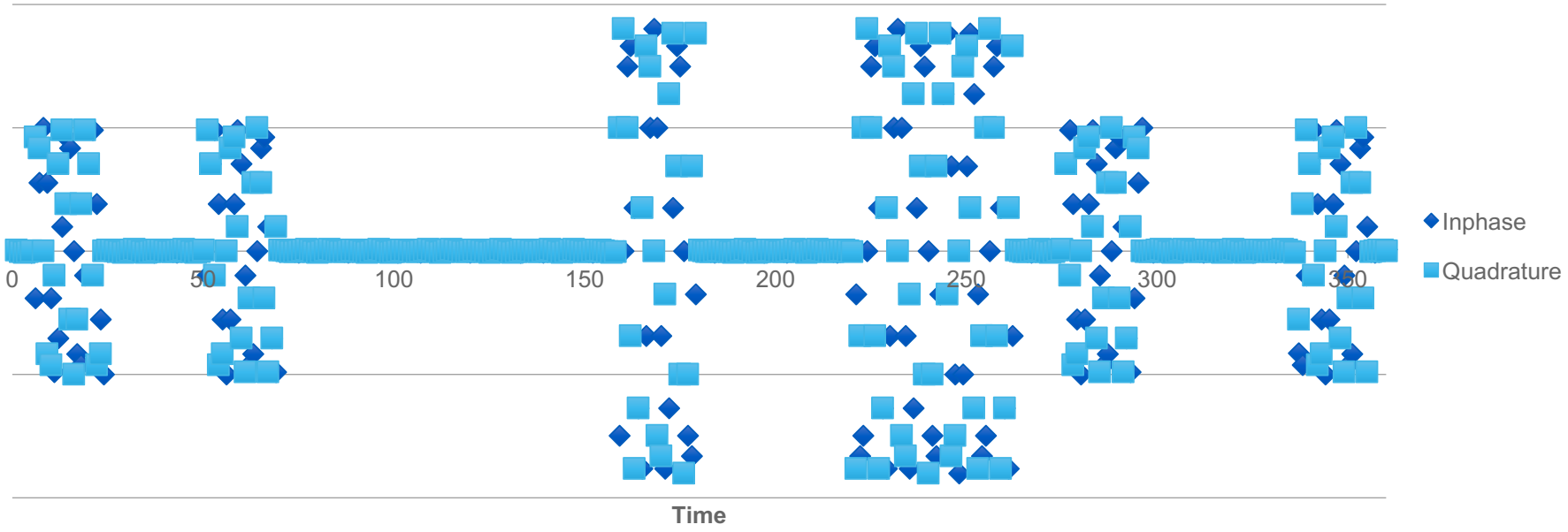
$$Q = A\cos(\omega_{RF}t + \varphi_{RF}) \times \sin(\omega_{LO}t + \varphi_{LO}) = \frac{A}{2}\left[\sin(\omega_{RF}t - \omega_{LO}t + \varphi_{RF} - \varphi_{LO}) + \sin(\omega_{RF}t + \omega_{LO}t + \varphi_{RF} + \varphi_{LO}\right]$$

$$V_Q = \frac{A}{2}\left[\sin(\varphi_{RF} - \varphi_{LO})\right]$$

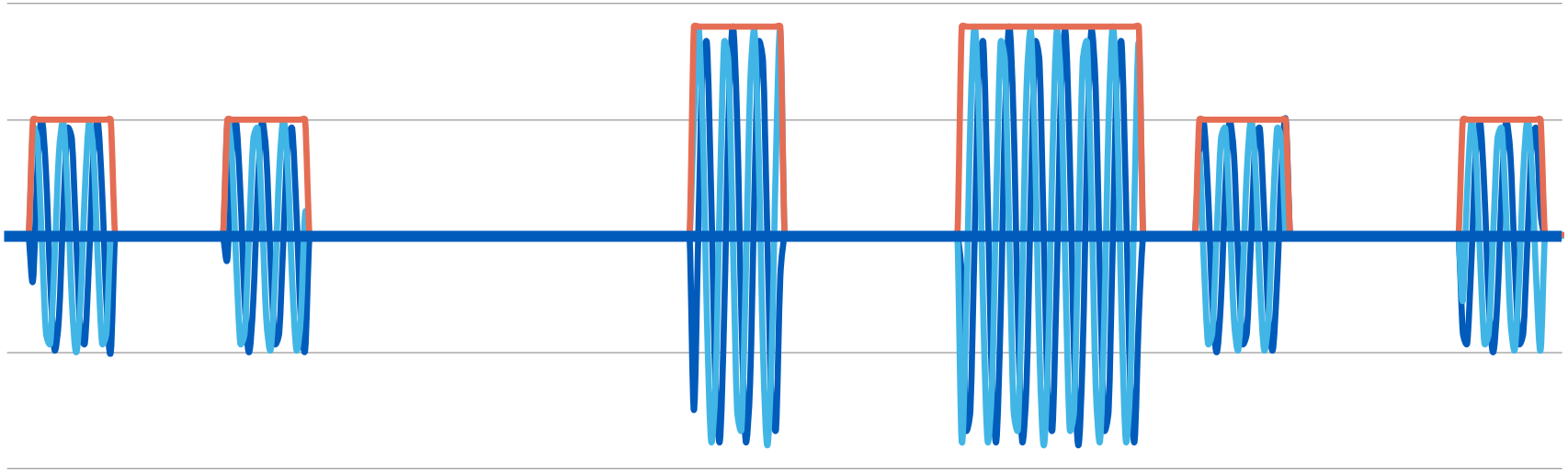
Let $\omega_{RF} = \omega_{LO}$ difference term at dc

Sum term gets filtered

Digitized Environmental Data



In-phase and Quadrature Analysis



Serial Step Detection Algorithm

Input: I_Data<NUM>, Q_Data<NMU>, Threshold
Output: Start_Points<INDEX>, End_Points<INDEX>

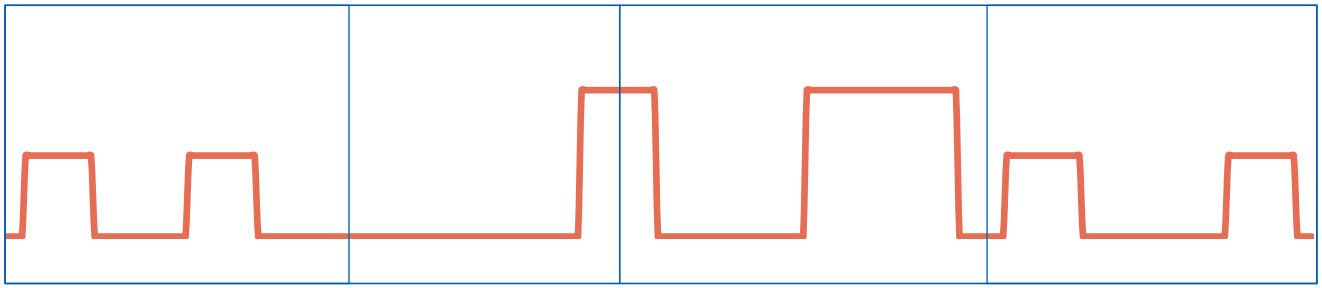
Boolean above = false

For index[start:stop]
    amplitude_sq = I_DATA[index]^2 + Q_DATA[INDEX]^2
    if (amplitude_sq > threshold^2 && !above) {
        Start_Points.pushback(index)
        above = true
    }
    else if(amplitude_sq < threshold^2 && above) {
        Stop_Points.pushback(index)
        above = false
    }

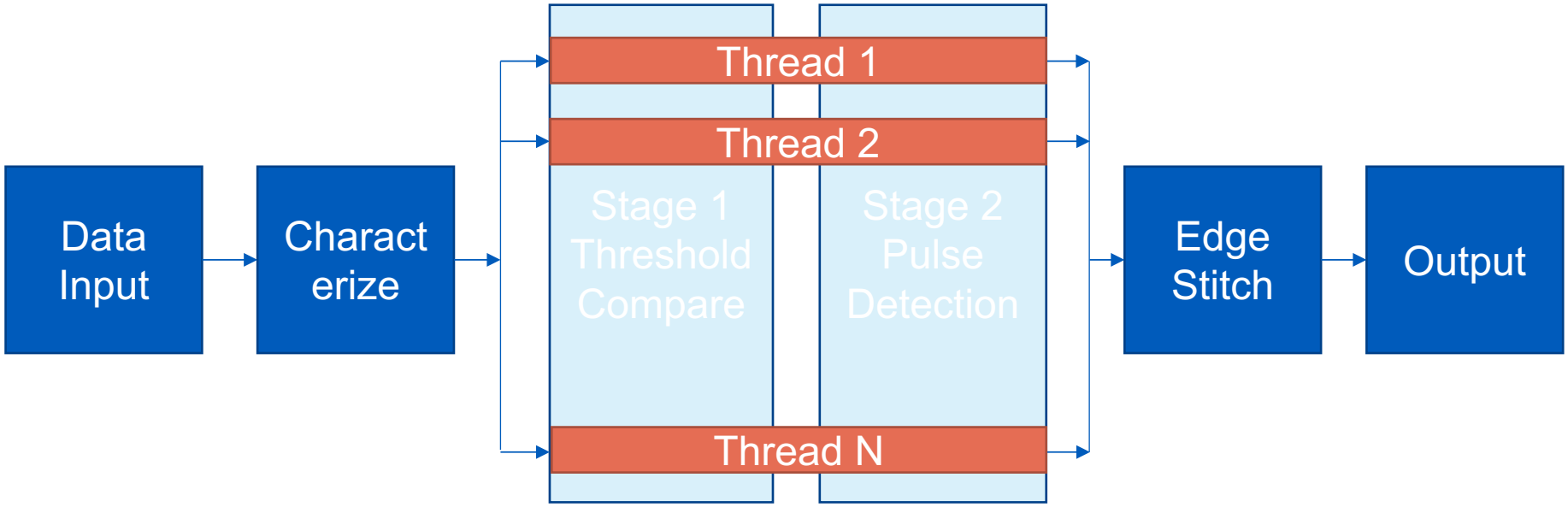
Parallel Algorithm

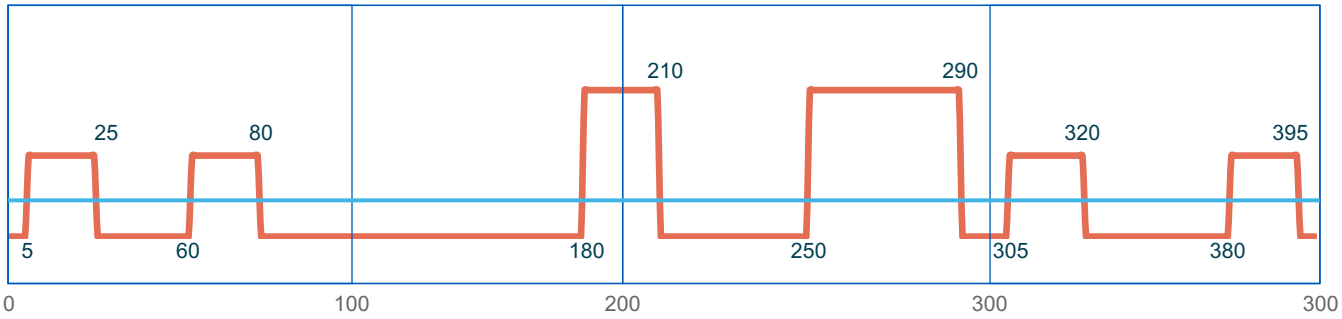
Divide and Conquer

- Easy split
  - Divide data amongst N processors
- Stage 1 in Parallel
  - Determine if amplitude is above threshold
- Stage 2 in Parallel
  - Identify edges
- Combine
  - Merge vectors
  - Handle pulse overlap



Parallel Algorithm





Parallel Implementation

C++ Application

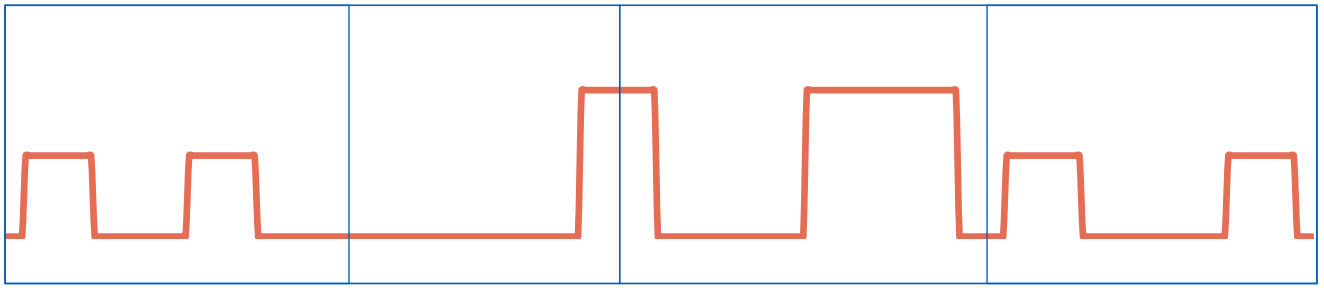
- Data generation utility
- Framework for stage based signal processing
- Processing algorithms in discrete kernels

OpenMP

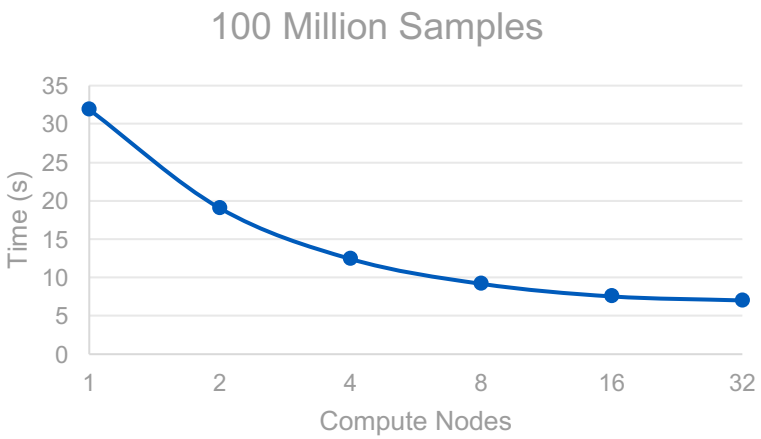
- omp parallel for
- omp_set_num_threads()

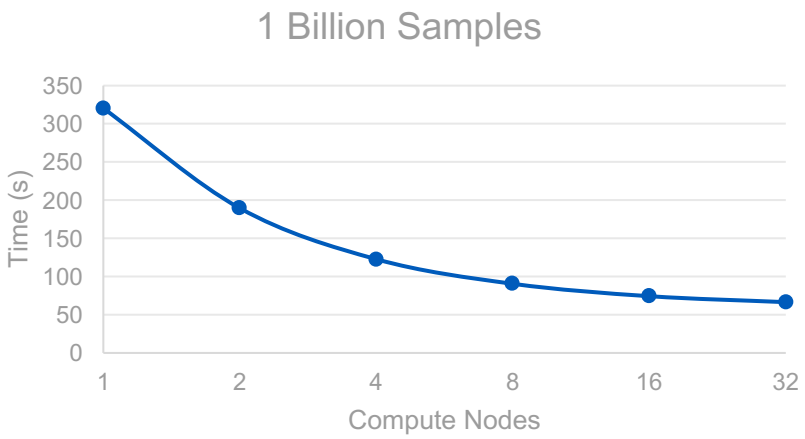
Large Memory Cluster

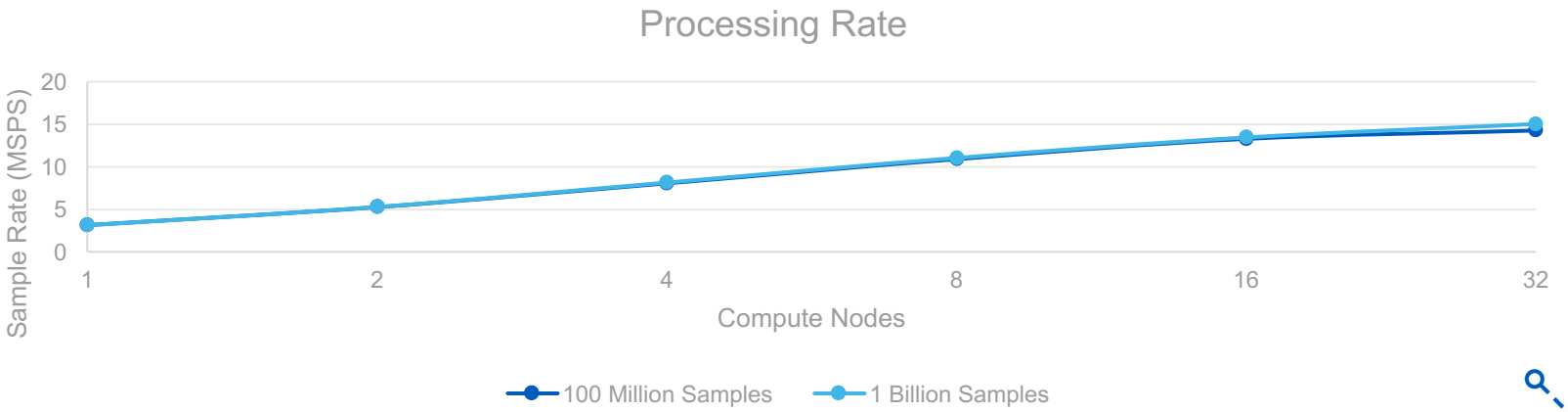
- 32 Core Machine
- 256+ GB RAM



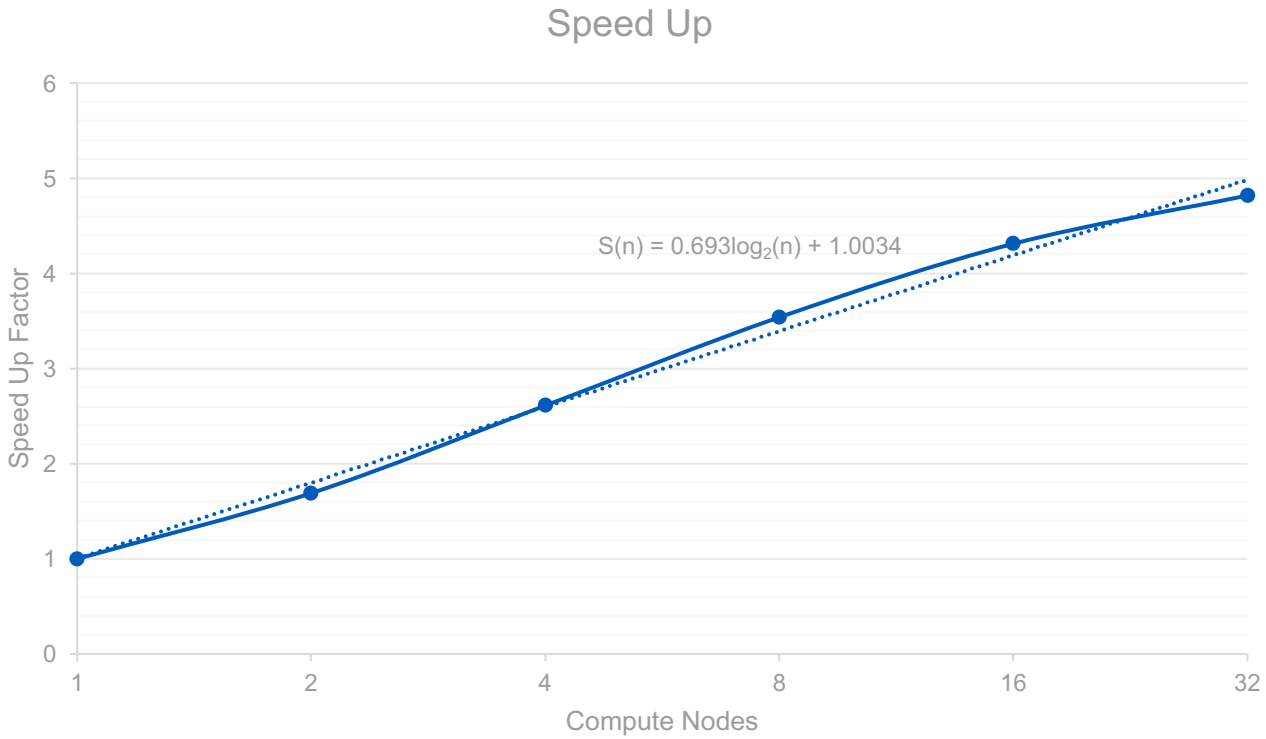
Runtime Analysis







Runtime Analysis



Conclusions

Successes

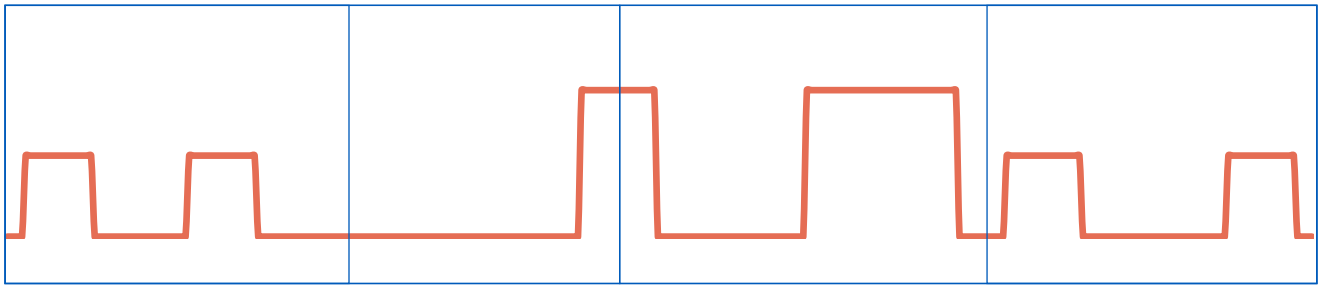
- Linear Speed Up achieved
  - 4x speed up on 8 core machine
- Reduce user facing processing time by utilizing idle cores
- Creation of parallelizable Digital Environment Processing Framework

Room for improvement

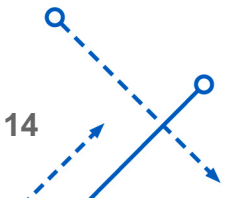
- Optimize processing at stage level

Future Work

- Add more DSP stages to processing framework
- Implement GPGPU kernels



Questions?



References

IQ Receiver Diagram
http://www.analog.com/en/analog-dialogue/articles/rf-to-bits-solution.html

IQ Sampling
http://www.ieee.li/pdf/essay/quadrature_signals.pdf

OpenMP and Slurm Support
https://ubccr.freshdesk.com/support/solutions/articles/13000026245-tutorials-and-training-documents

Runtime Data

Samples/Cores	1	2	4	8	16	32
1.00E+07	2.956138	1.839053	1.245545	0.953884	0.825309	0.841648
1.00E+08	31.872928	19.02572	12.410948	9.170127	7.524059	7.004921
1.00E+09	320.359305	189.516166	122.697726	90.522017	74.30246	66.487053