# Towards Online Structural State-Estimation with Sub-millisecond Latency

**Austin R.J. Downey[1,2], Joud Satme[1], Ehsan Kabir[3], Daniel Coble[1], Jason D. Bakos[4], David Andrews[3], Miaoqing Huang[3], Adrine Moura[4], and Jacob Dodson[5]**

[1]University of South Carolina, Department of Mechanical Engineering
[2]University of South Carolina, Department of Civil and Environmental Engineering
[3]University of Arkansas, Department of Computer Science and Computer Engineering
[4]University of South Carolina, Department of Computer Science and Engineering
[5]Applied Research Associates, Emerald Coast Division
[6]Air Force Research Laboratory, Eglin Air Force Base




# Structures Experiencing High-Rate Dynamic Events

Applications:

1. Vehicle collision
2. Blast mitigation
3. Ballistic packages
4. Hypersonic vehicles
5. Hard Target Penetrating Weapons

Vehicle Collision


Active Blast Mitigation

Ballistics Packages

Hypersonic Vehicles

Hard Target Penetrating Weapons

# Formal Definition for High-Rate Dynamic Events

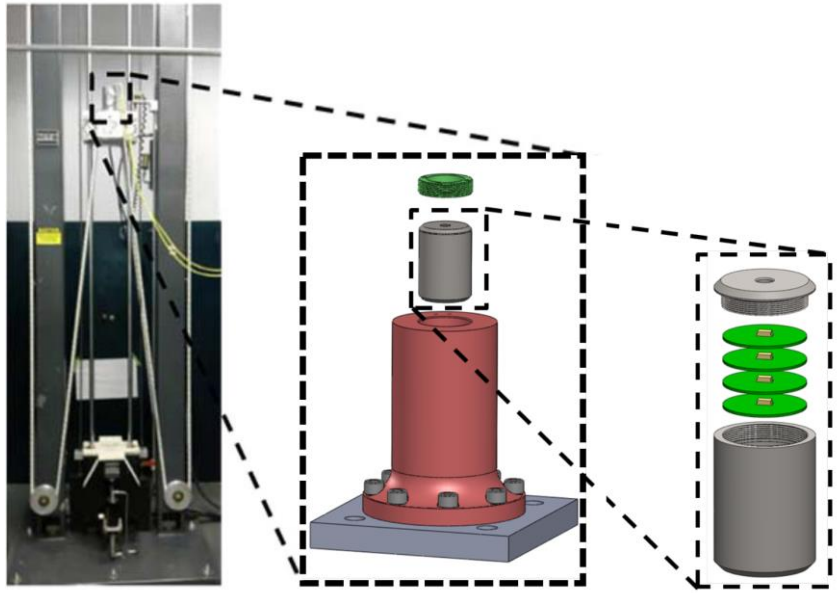
High-rate dynamics are described as a dynamic response from a high-rate (<100 ms) and high-amplitude (acceleration > 100 *g*) event such as a blast or impact.

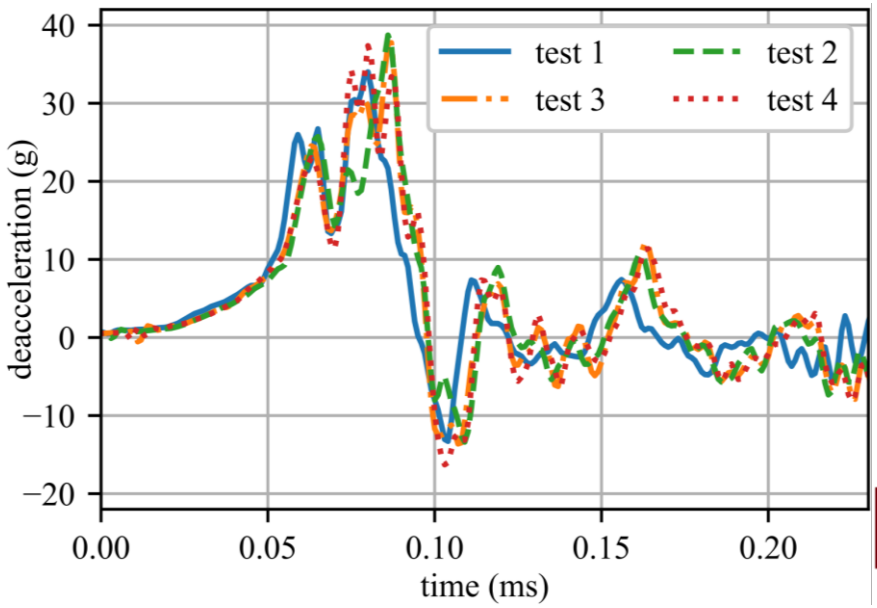
The high-rate problem contains many complexities that can be summarized as having:

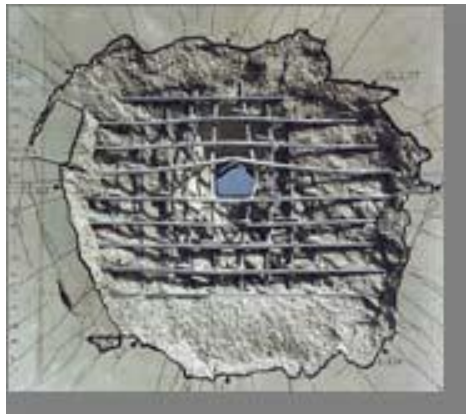
1) large uncertainties in the external loads;

2) high levels of non-stationarities and heavy disturbances; and

3) generated unmodeled dynamics from changes in system configuration.





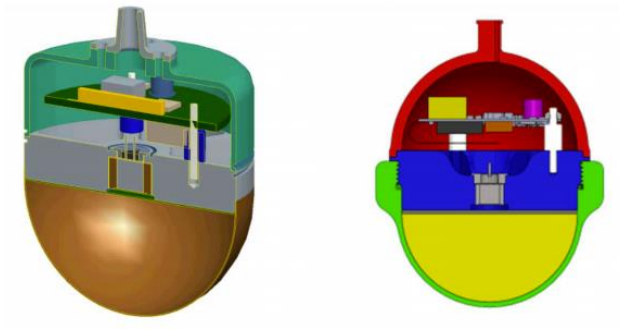



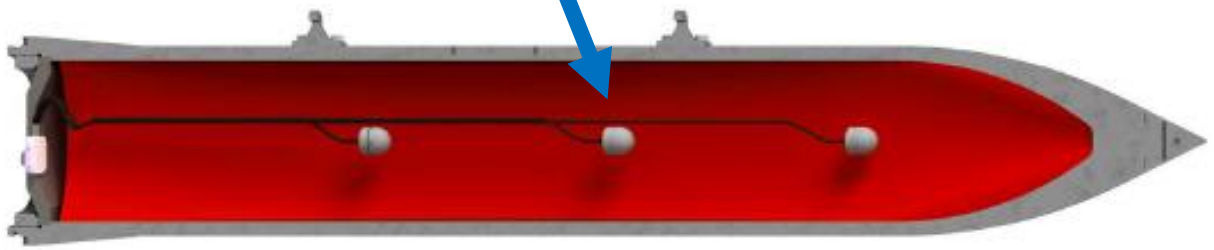
JUNGHANS Microtec

# Next Generation Fuzes

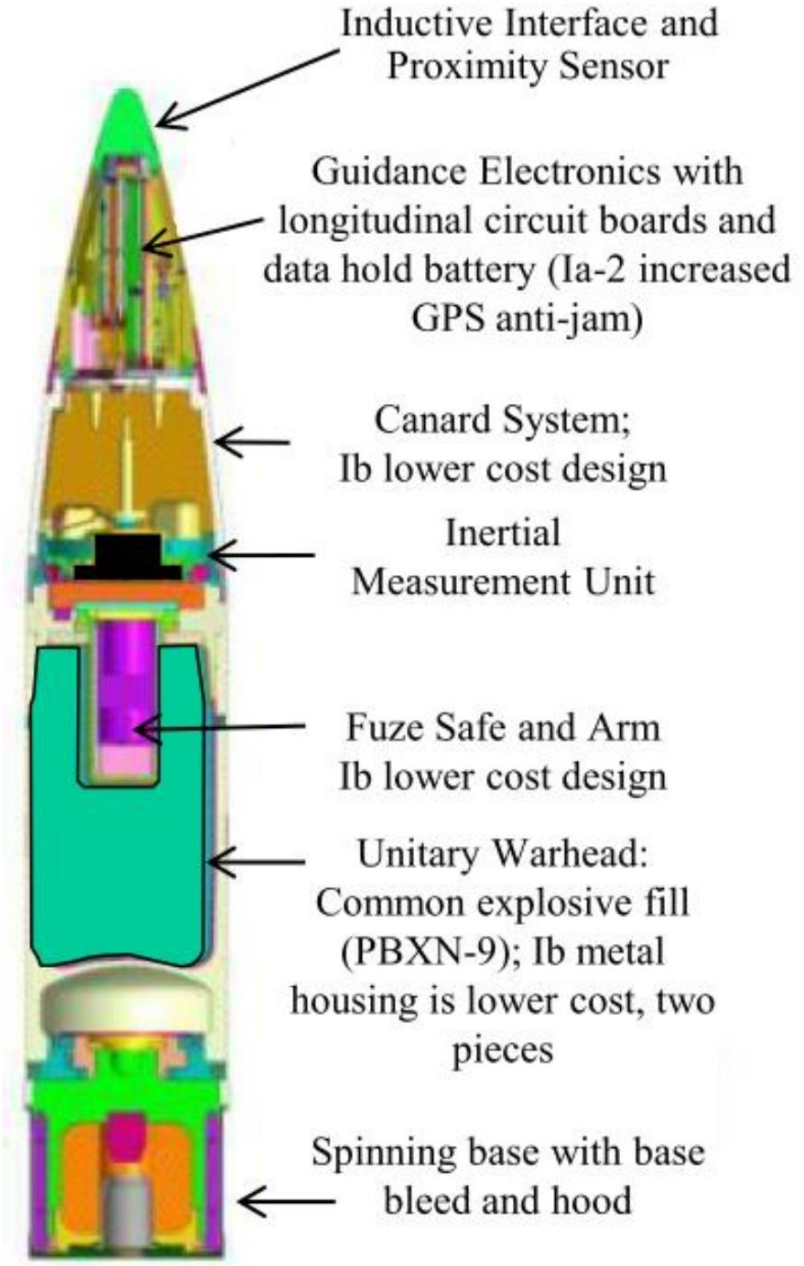
Thorough collaborations with the AFRL we are working on enabling technology for

- Fuzes with decision-making capabilities.
- Fuzes that can “adapt” to their condition.
- Fuzes that are resilient to impact (e.g. after an impact, the are just as strong as before).





Embedded fuze design are being designed for enhanced safety and reduced cost.






SAND2019-5293C

# Fuzes for Hard Targets

**The Challenge of Hard Target Fuze Design**

- Fuzes are a relatively delicate payload
- Fuzes break in harsh environments
- Too many failure modes for fly-fix-fly approach
- Full scale high-g testing is expensive
- Full scale high-g testing can't uncover all failure modes

**Desired Outcomes**

- Fuzes have one good outcome:
  - Initiation when intended
- They have two incorrect outcomes:
  - Initiation before or after intended
  - Failure to initiate

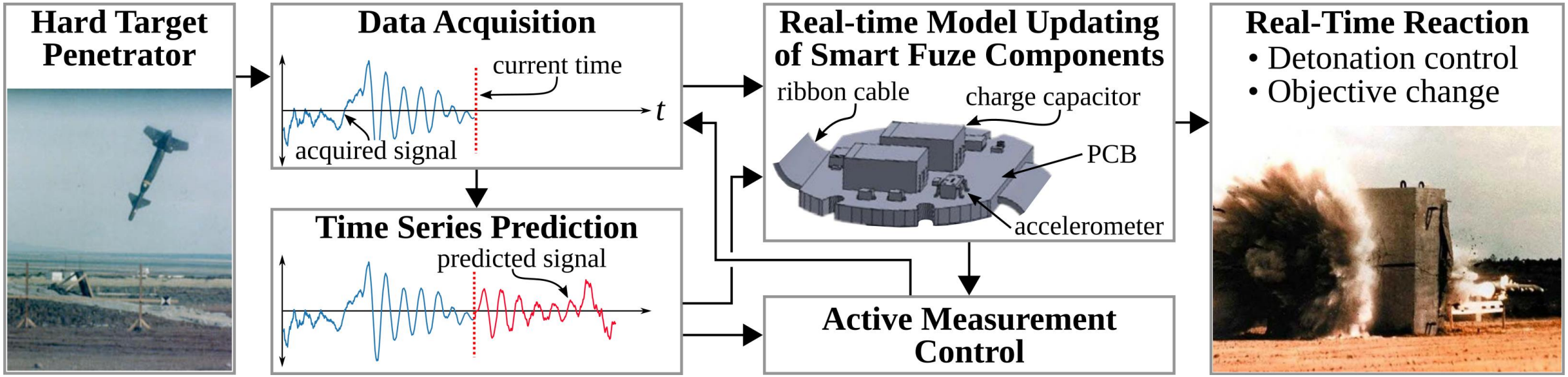

Harsh environment

delicate payload

# Project's Long-term Applied Goal

Develop a computationally efficient real-time decision-making framework for structures (i.e. fuzes) experiencing high-rate dynamics:

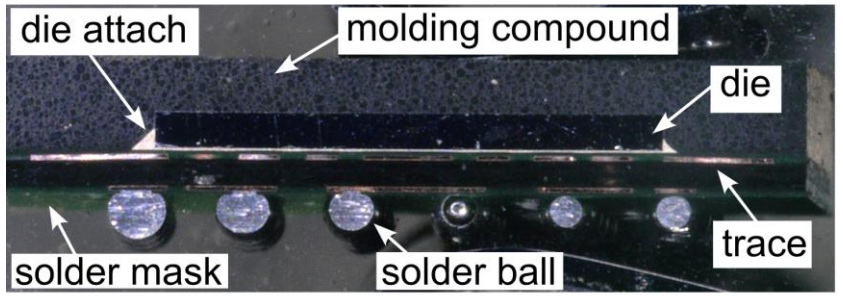
- The framework should be capable of being executed on edge-computing platforms with a timestep of 1 ms.
- Real-time state estimation of the structure is an important first step.

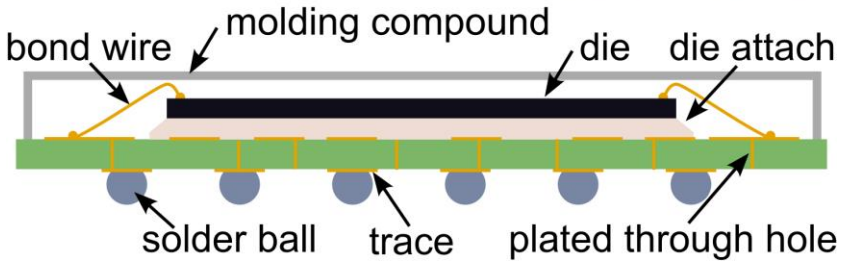


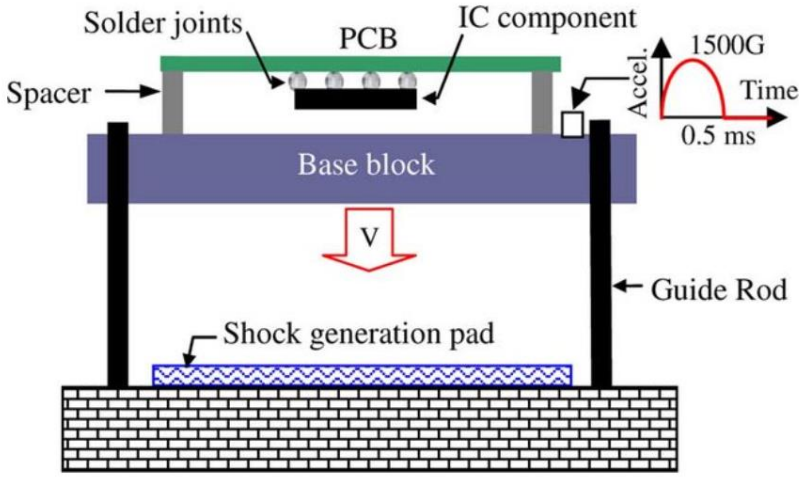
# Typical PCB Failure Mechanisms under Shock

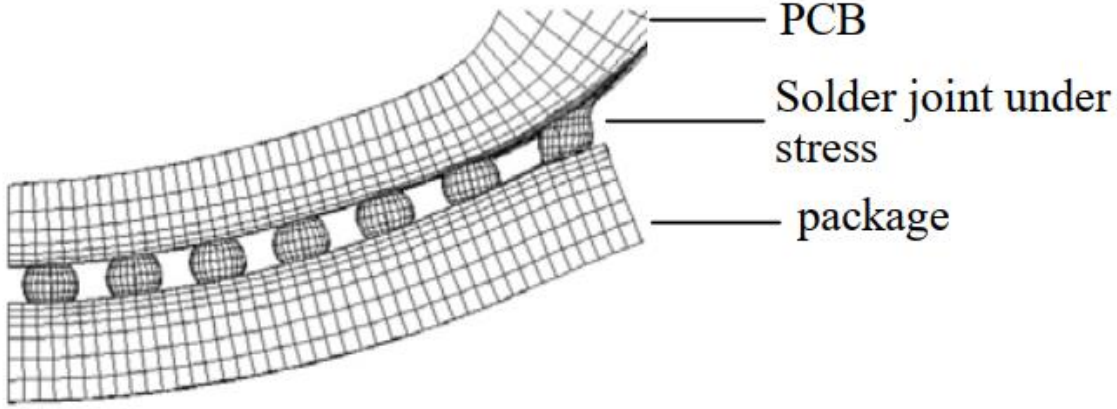
PCB failures under shock are caused by:

- Bending of the base PCB board, causing stresses to build up at the solder balls.
- Adhesion challenges of masses (components) accelerating away from the PCB.




Wong, E. H., Yiu-Wing Mai, and Matthew Woo. "Analytical solution for the damped-dynamics of printed circuit board and applied to study the effects of distorted half-sine support excitation." IEEE Transactions on advanced packaging 32.2 (2009): 536-545.

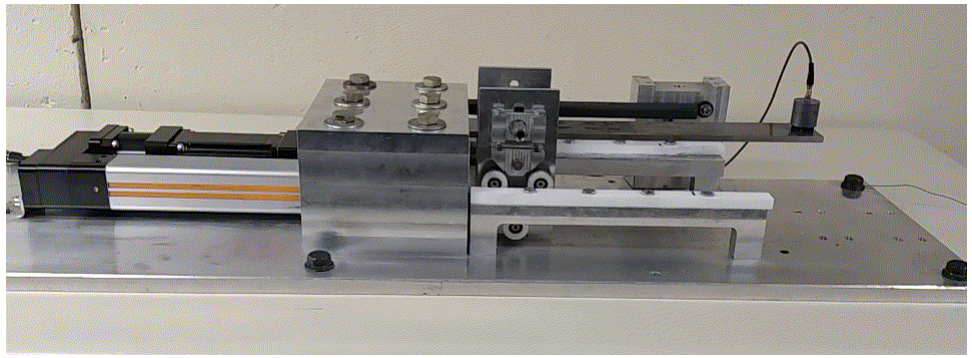
Seah, S. K. W., Wong, E. H., Ranjan, R., Lim, C. T., and Mai, Y. W., 2005, "Understanding and testing for drop impact failure," ASME Pacific Rim Technical Conference and Exhibition on Integration and Packaging of MEMS, NEMS, and Electronic Systems, pp. 1089-1094.

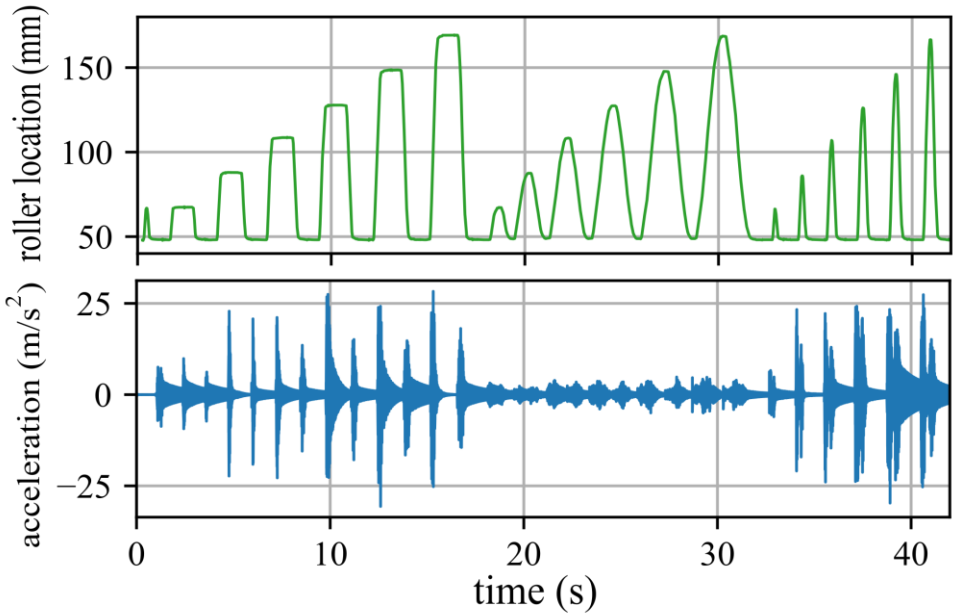
# Data-driven Modeling for Structural State-estimation

# Experimental System used for Validation

DROPBEAR experimental testbed:

- The Dynamic Reproduction of Projectiles in Ballistic Environments for Advanced Research (DROPBEAR) was used to generate the experimental data in this work.
- Cantilever beam with a controllable roller to alter the state.
- Acceleration and pin location are recorded.
- Dataset available on GitHub at: https://github.com/High-Rate-SHM-Working-Group/Dataset-2-DROPBEAR-Acceleration-vs-Roller-Displacement

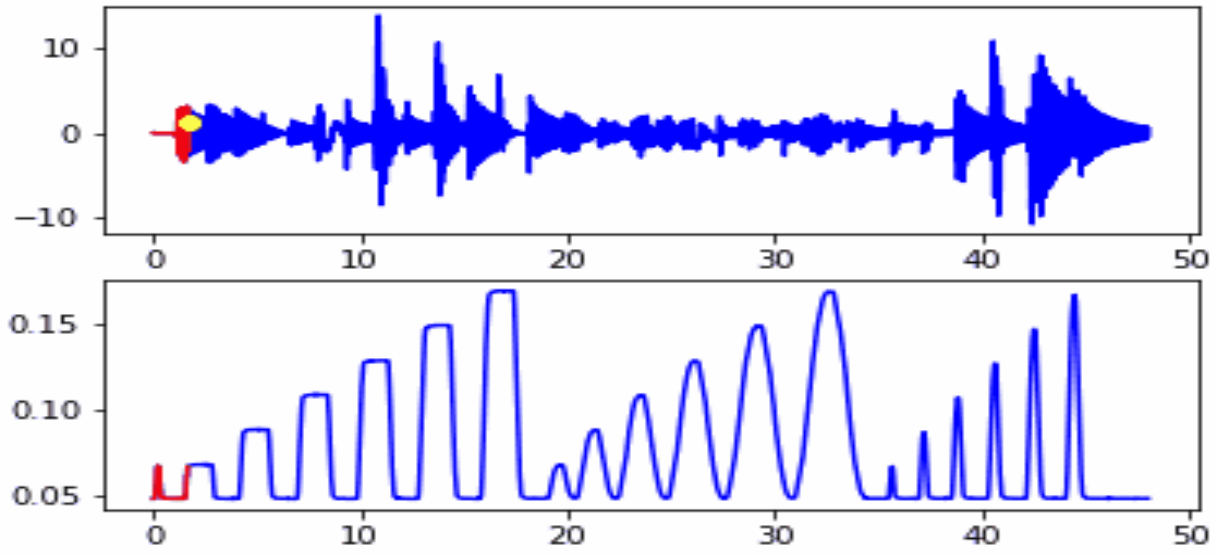


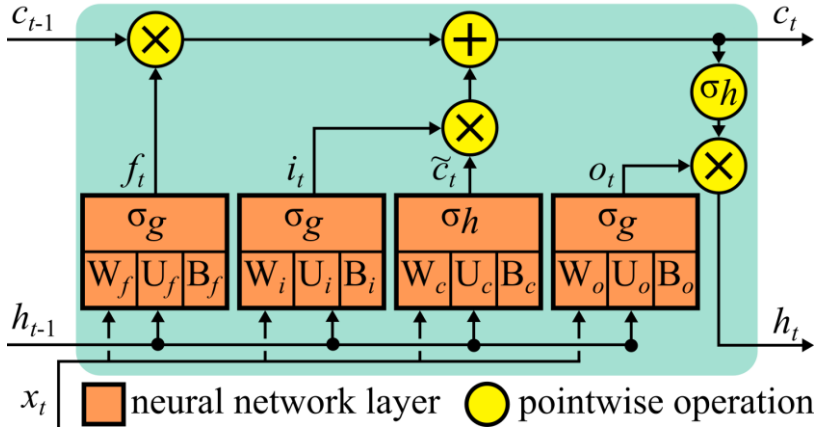


# LSTM-based Real-time State Estimation

In this work:

- Long short-term memory (LSTM) models are used for real-time state estimation.
- These data-driven models are trained offline on pre-recorded data.

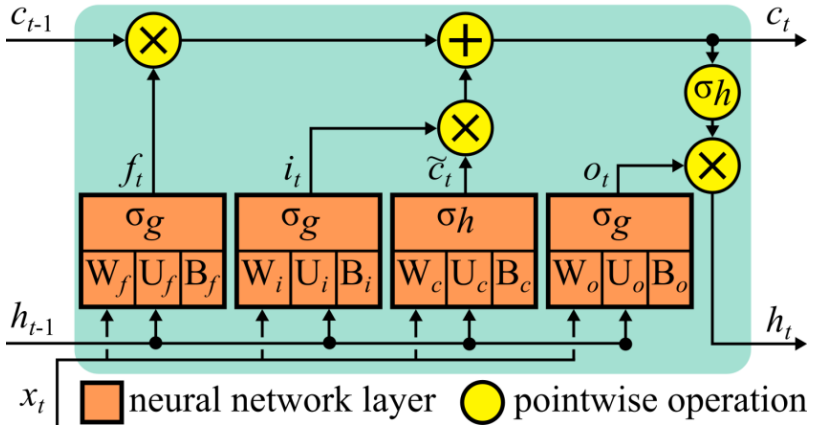


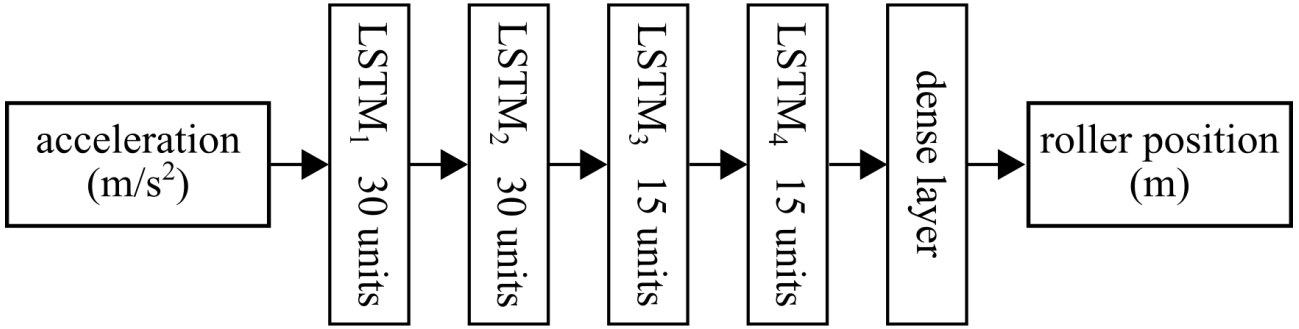



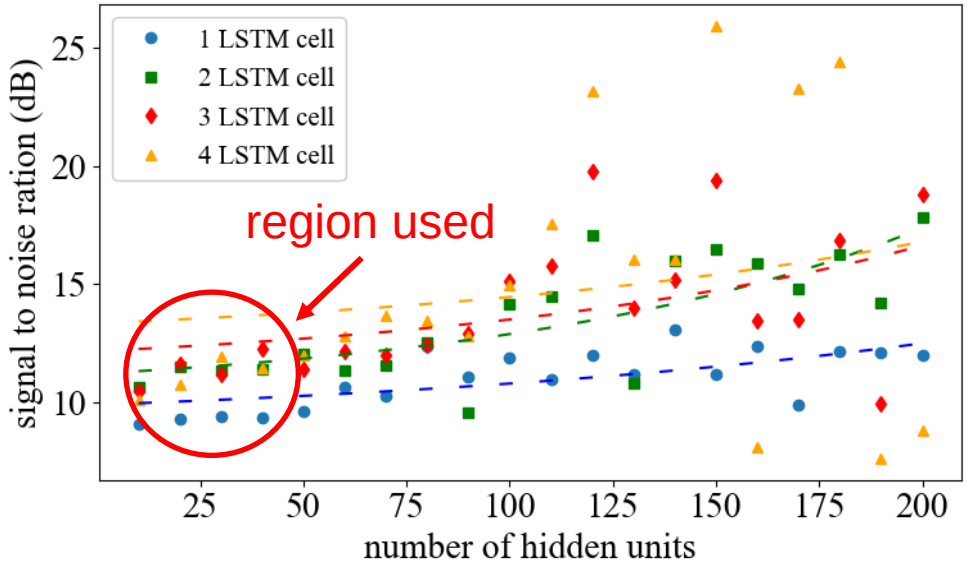
# Long Short-term Memory Model

LSTM features and development:

- LSTMs are a Recurrent Neural Network (RNN) that propagates through long- and short-term memory forms.
- Four stacked LSTM cells (30, 30, 15, 15 units) with a fully connected layer at the output.



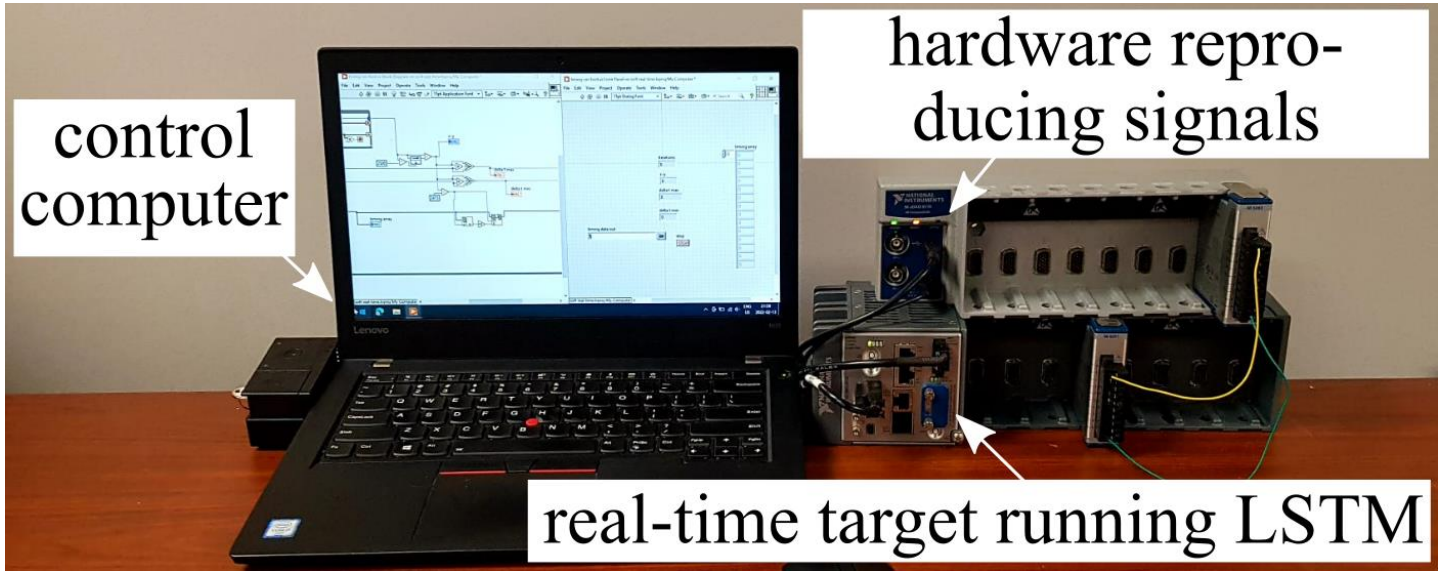



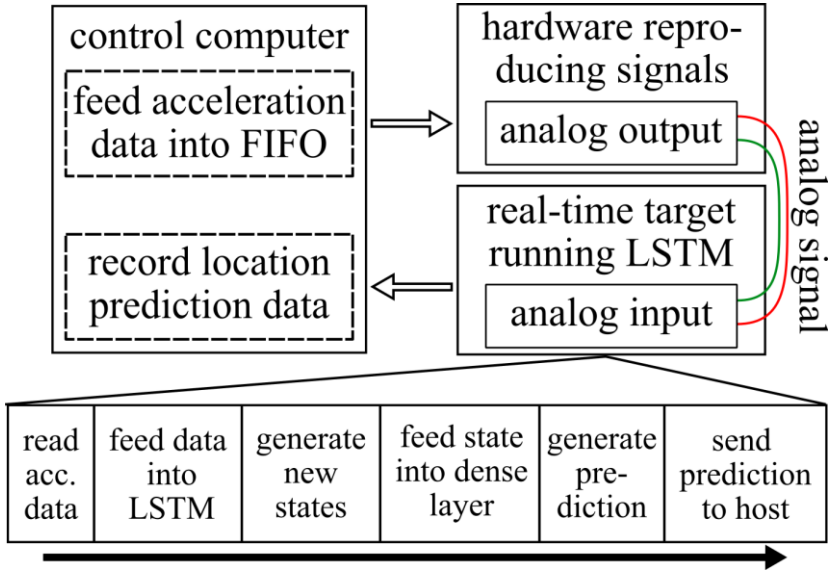
# Model Deployment on a Real-Time Operating System (RTOS)

# Model Deployment on RTOS

Real-time validation performed on an embedded system running:

- The experimental setup consisted of two subsystems:
  - **Hardware reproducing Signals** reproduces the DROPBEAR dataset using a digital to analog converter.
  - **Real-time Target** digitizes the analog voltage and feeds the input into the LSTM architecture (cRIO-9035).
- Data is sampled at 1250 S/s; therefore, a prediction is made every 800 µs.
- State predictions are returned via a first-in-first-out (FIFO) buffer to the host PC).

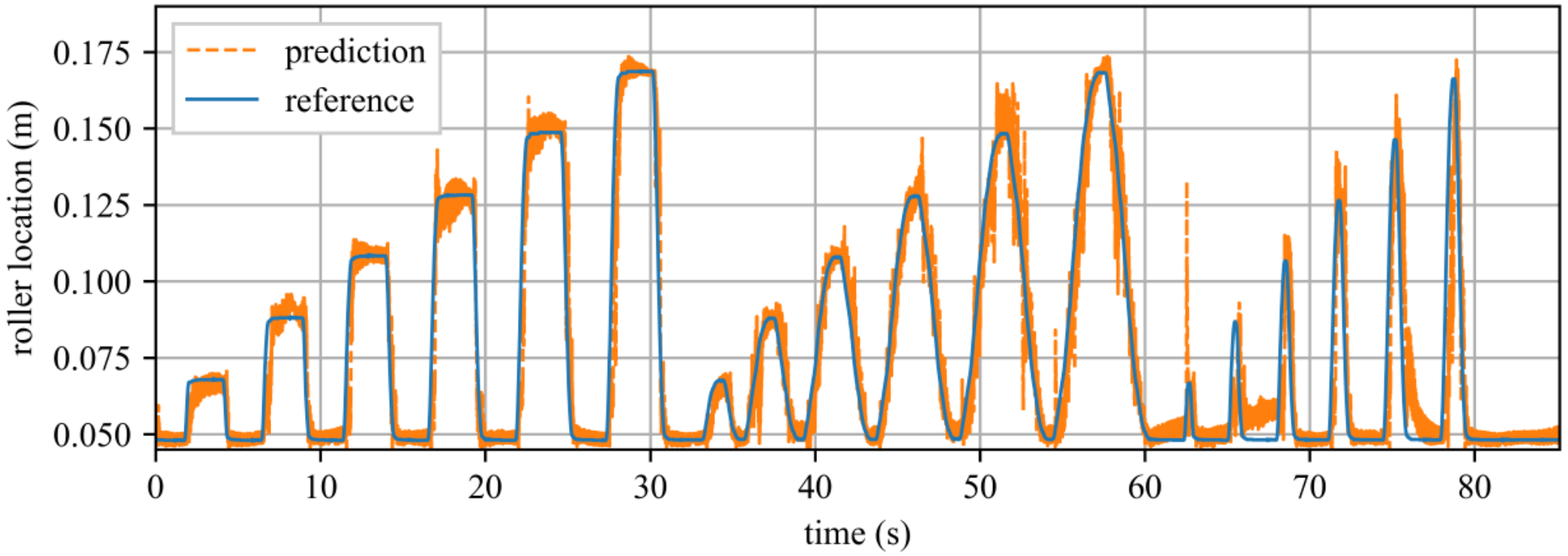


# Real-time LSTM Modeling Results

LSTM model performance results:

- $SNR_{dB}$ 17.4888 dB.
- RMSE of 11.471 m mm.
- LSTM traces reference pin location closely.



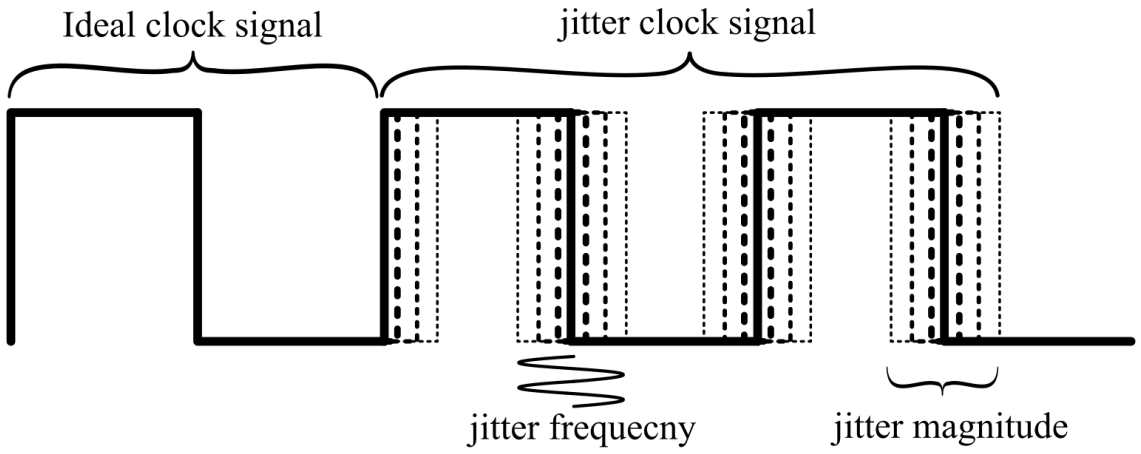
# Real-time LSTM Modeling Results

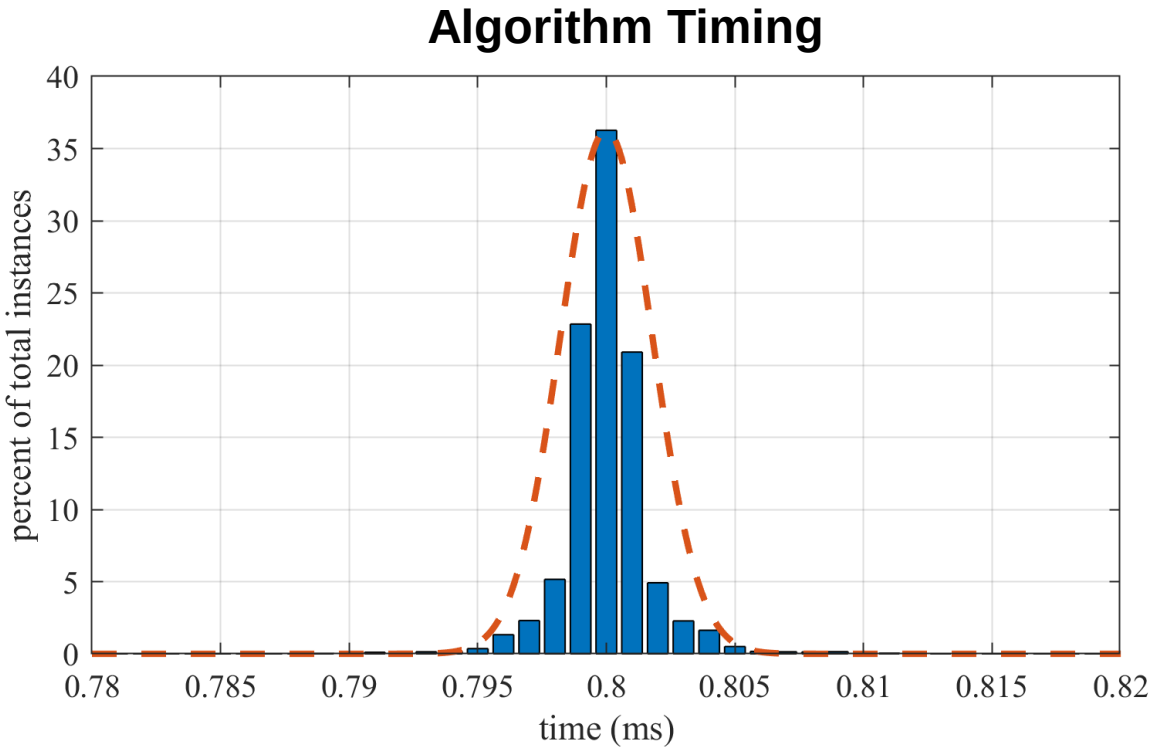
LSTM model timing results:

- Average: 800 µs.
- Standard deviation: 1.79 µs.
- Max overshoot: 26 µs.

Timing accuracy results:

- Execution-time jitter as expected.
- Timing follows a normal distribution.




Intel Atom® Processor E3825

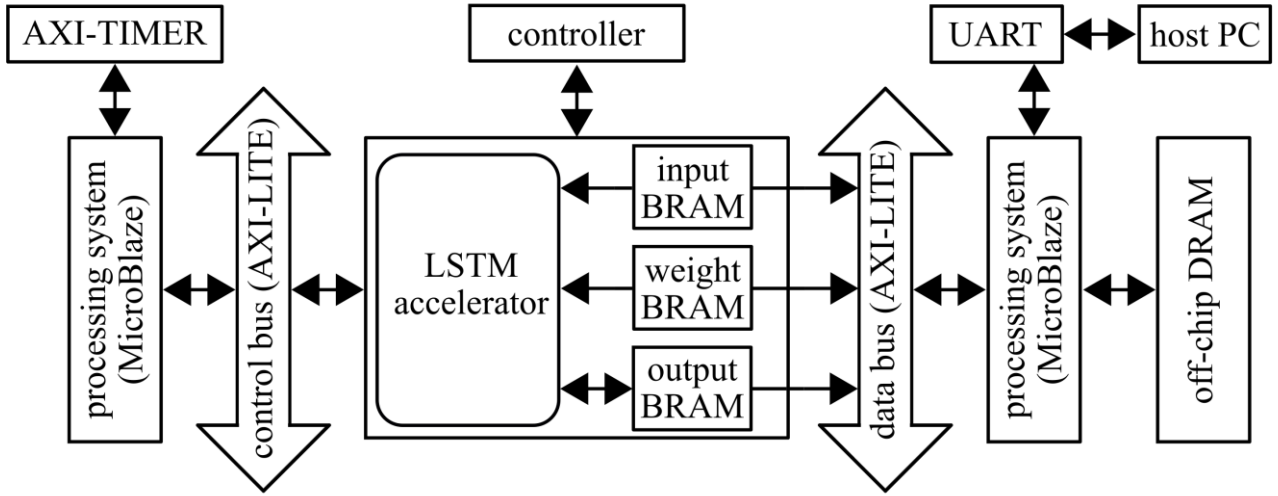
- Total Cores: 2 (2 threads)
- Processor Base Frequency: 1.33 GHz
- Cache: 1 MB L2 Cache
- Use Conditions: Automotive, Embedded

# Model Deployment on a Field Programable Gate Array (FPGA)

# Model Deployment on FPGA

LSTM model deployed on a Xilinx Virtex 7 (VC707) FPGA:

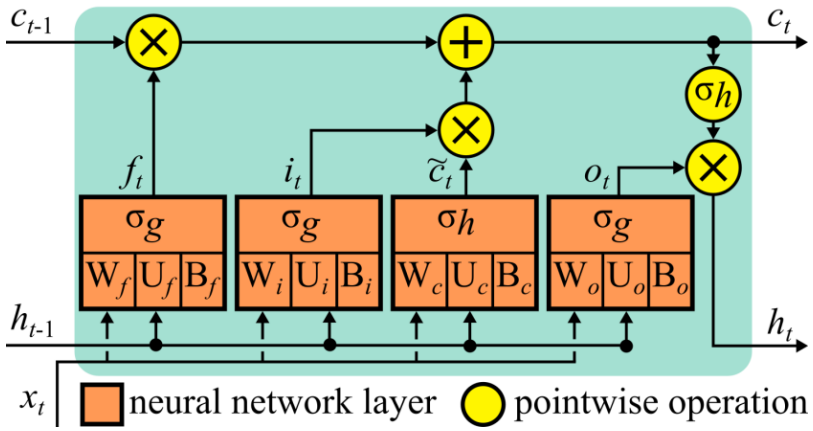
- Implemented in both 16-bit fixed point and a 32-bit float version.
- Developed an LSTM hardware accelerator where data in and out the FPGA is pre and post-processed with the MicroBlaze soft core processor.

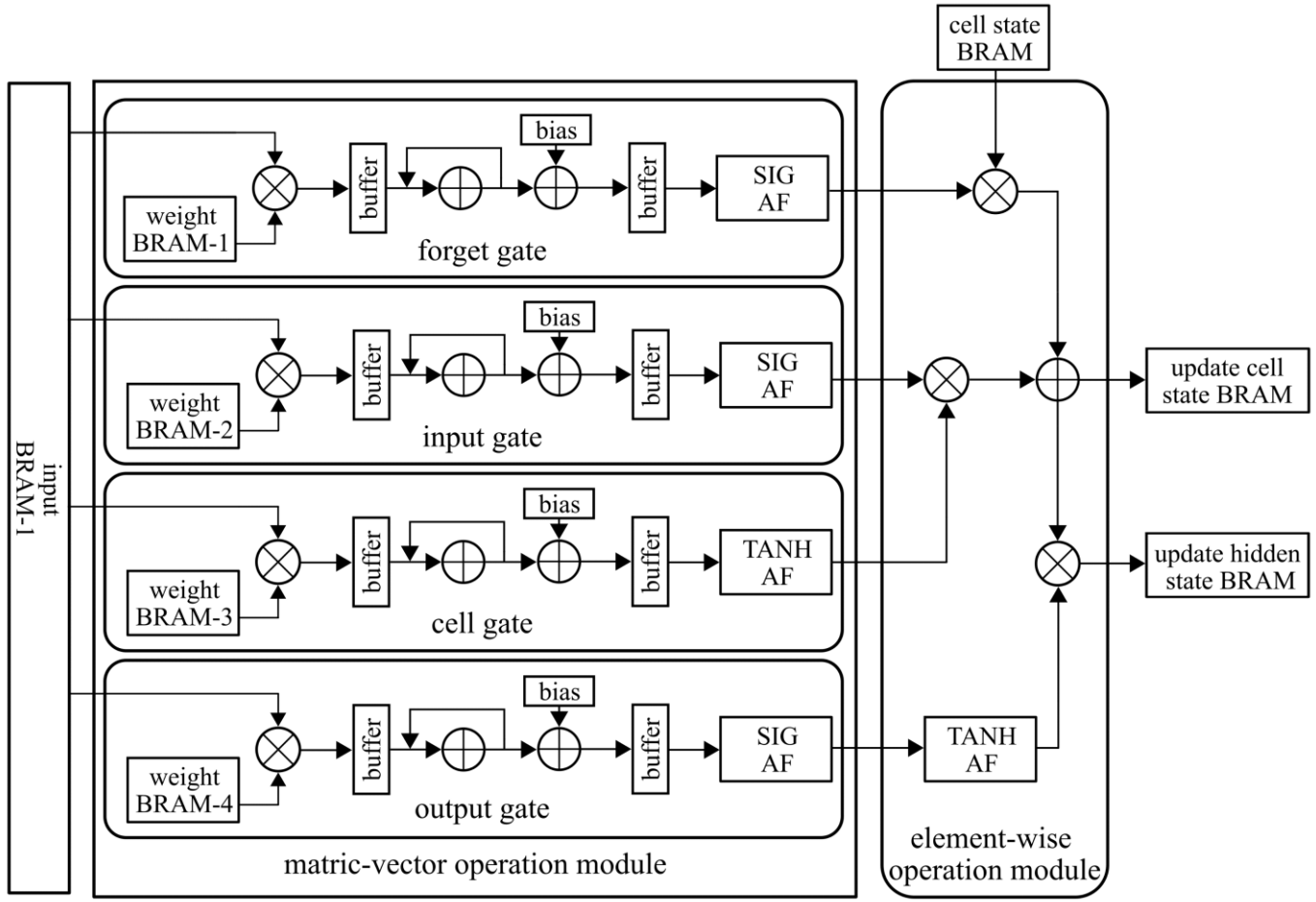



Xilinx Virtex 7 (VC707)

# LSTM deployment on an FPGA

The developed hardware accelerator is split up into the LSTM's gates for deployment.



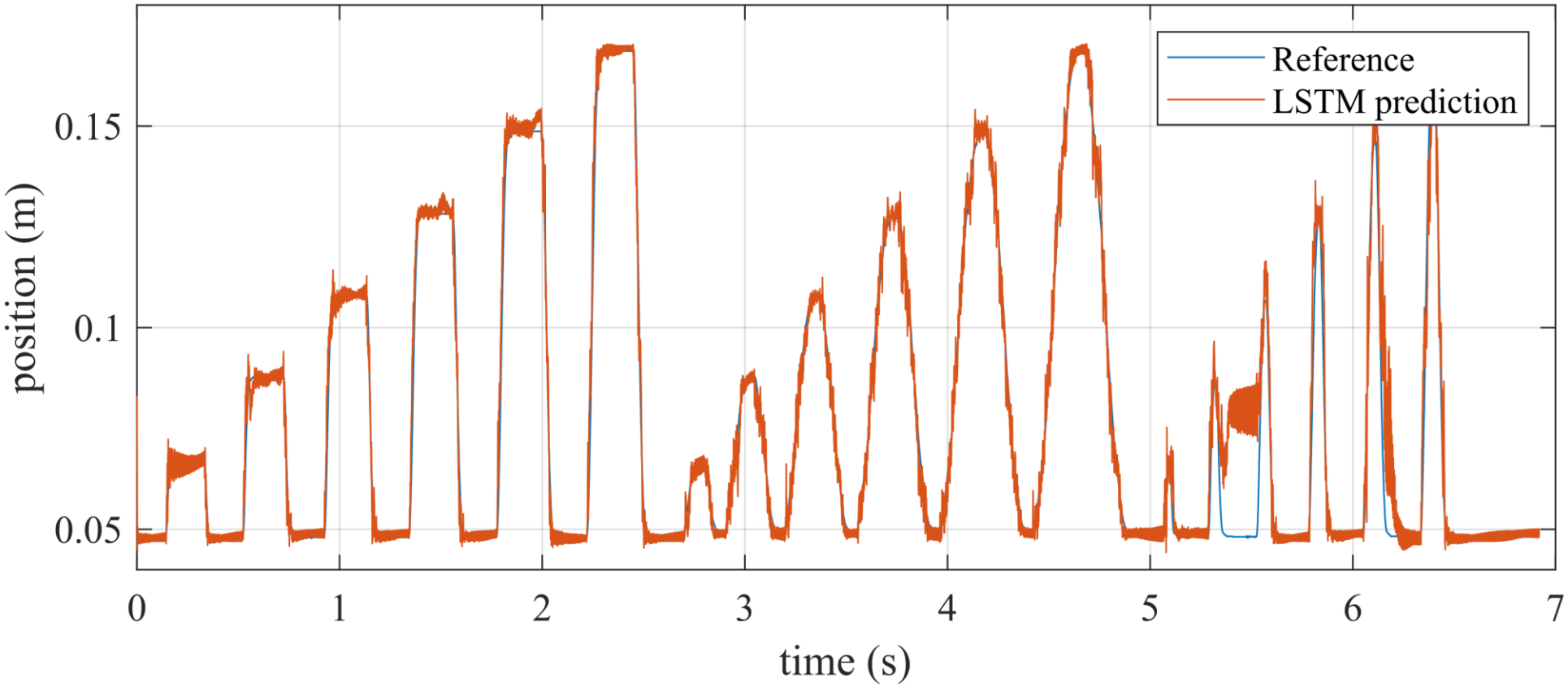
# Real-time LSTM Modeling Results

16-bit fixed point model performance:

- $SNR_{dB}$ of 19.54 dB.
- RMSE of 9.1 mm.

32-bit floating point model performance:

- $SNR_{dB}$ of 22.02 dB.
- RMSE of 6.8 mm.



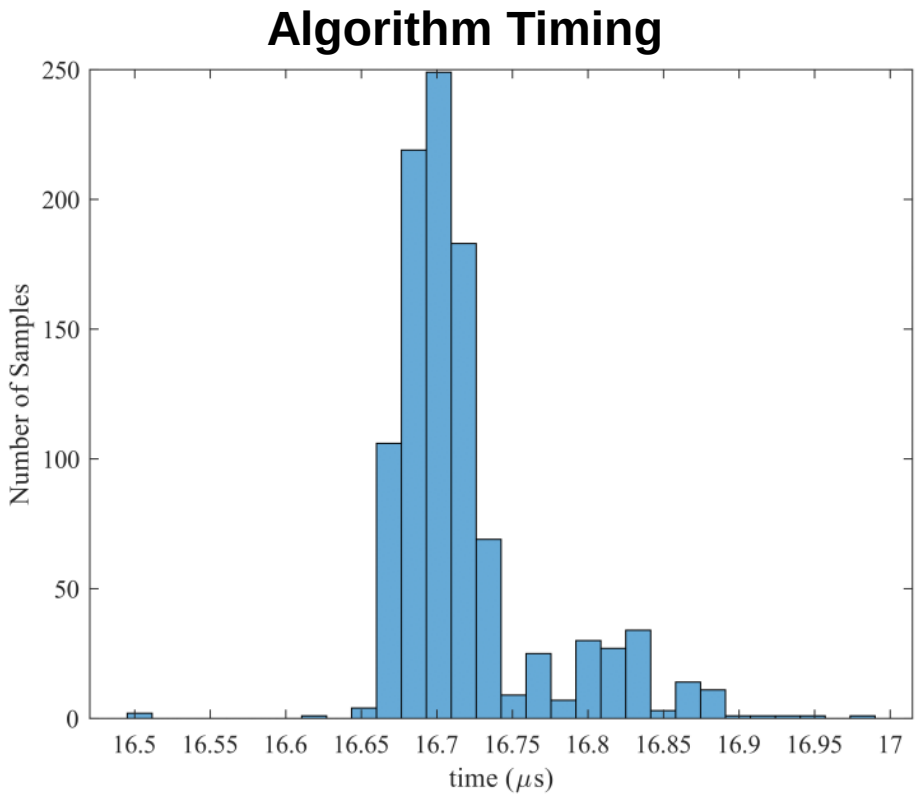
# Real-time LSTM Timing Results

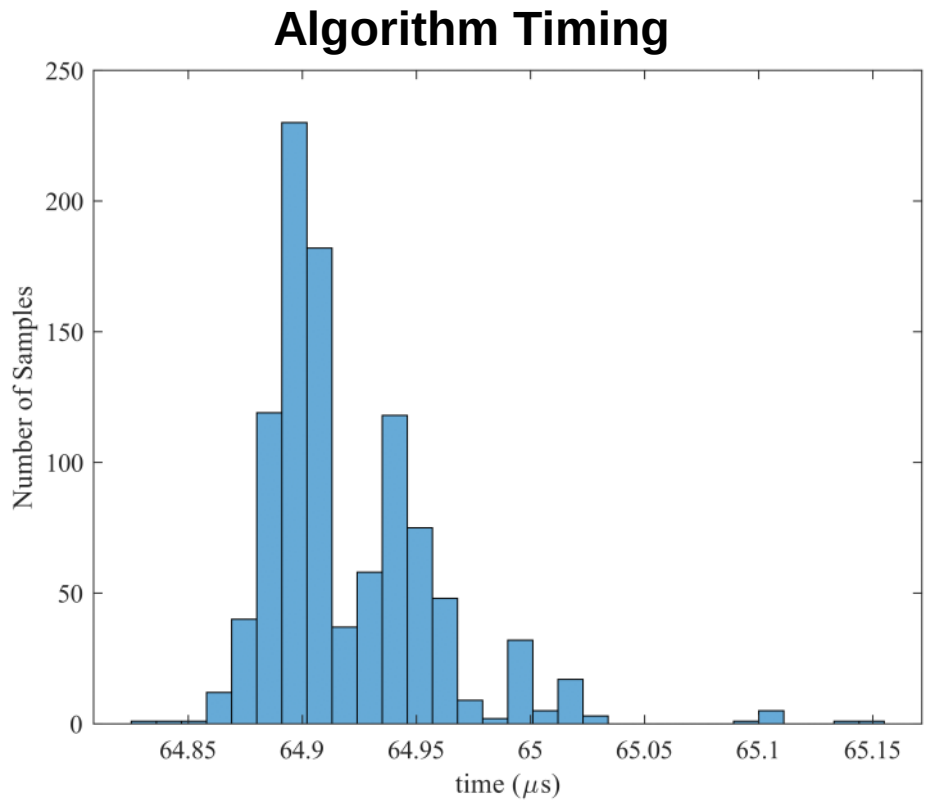
16-bit fixed point model performance:

- Time step of 16.7 µs.
- Standard deviation: 0.0509 µs.
- 50X speed up over RTOS.

32-bit floating point model performance:

- Time step of 64.9 µs.
- Standard deviation: 0.0379 µs.
- 12X speed up over RTOS.



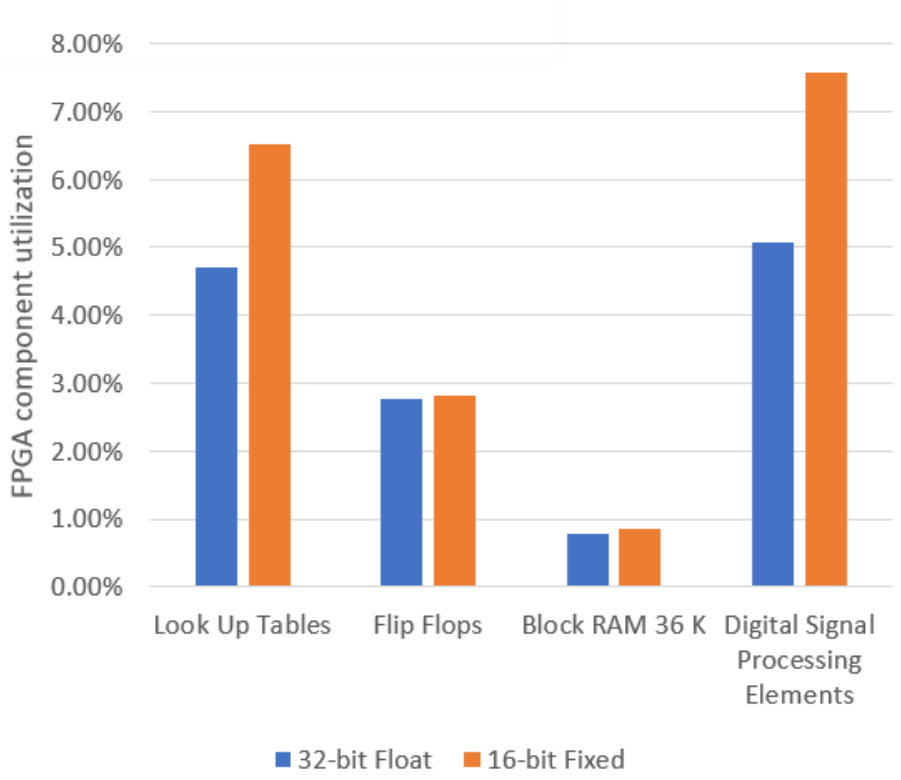



# FPGA Resource Utilization Results

Synthesized tow models for the Xilinx Virtex 7 (VC707):

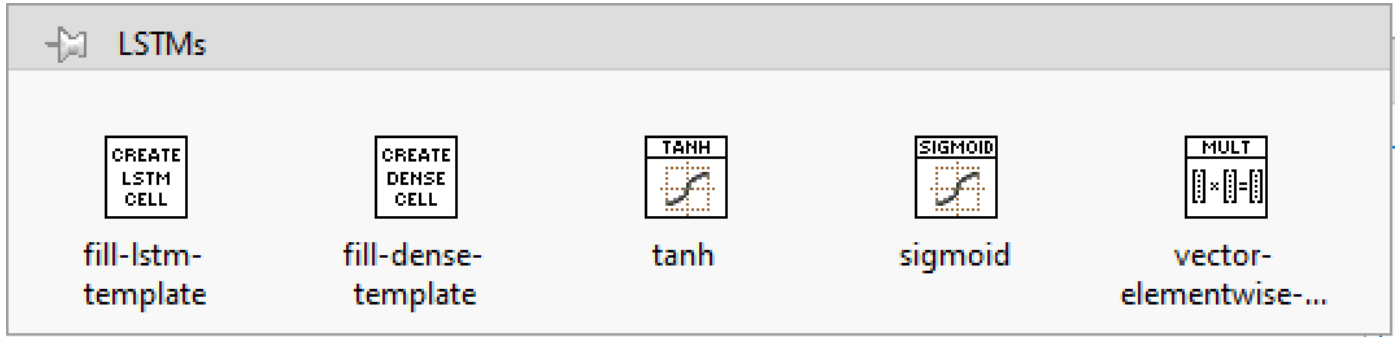
- 16-bit fixed point implementations is significancy faster than the 32-bit float version.
- Both implementations consume less the 10% of FPGA resources.

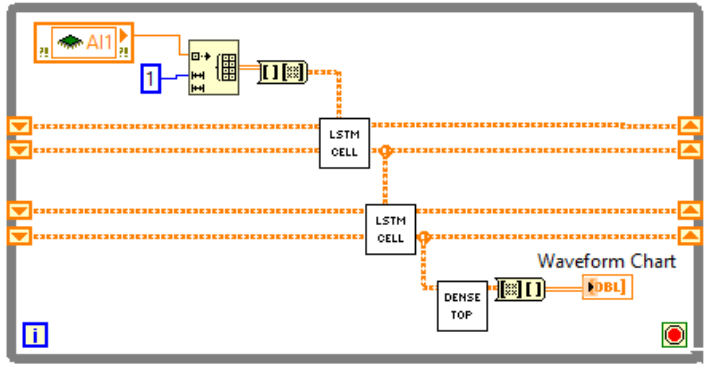
| Model | Freq. (MHz) | Data Precision | LUT | FF | BRAM 36 k | DSP | Hardware Execution Time (uS) |
|---|---|---|---|---|---|---|---|
| 16, 15, 15, 15, 1 | 200 | 32bit Float | 91611 | 107964 | 211 | 142 | 64.5 |
| 16, 15, 15, 15, 1 | 200 | 16bit Fixed | 126633 | 109186 | 229 | 212 | 16.155 |



# Open-Source Codes and Data Sets

- Open-Source library for Deploying LSTMs to the NI Linux Real-time Operating System at: https://github.com/ARTS-Laboratory/LabVIEW-LSTM
- Code for ASME-IDETC conference paper at: https://github.com/ARTS-Laboratory/Paper-Progress-towards-data-driven-high-rate-structural-state-estimation-on-edge-computing-devices
- Dataset available on GitHub at: https://github.com/High-Rate-SHM-Working-Group/Dataset-2-DROPBEAR-Acceleration-vs-Roller-Displacement



# Funding Acknowledgment

This material is based upon work partially supported by the Air Force Office of Scientific Research (AFOSR) through award no. FA9550-21-1-0083. This work is also partly supported by the National Science Foundation Grant numbers 1850012, 1937535, and 1956071. The support of these agencies is gratefully acknowledged. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Air Force.

South Carolina

# DISCUSSION

Research Group 2022 Spring Picnic

Contact Information: Austin Downey
Email: austindowney@sc.edu
Github: https://github.com/austindowney
Github-Lab: https://github.com/Arts-laboratory/

University of South Carolina

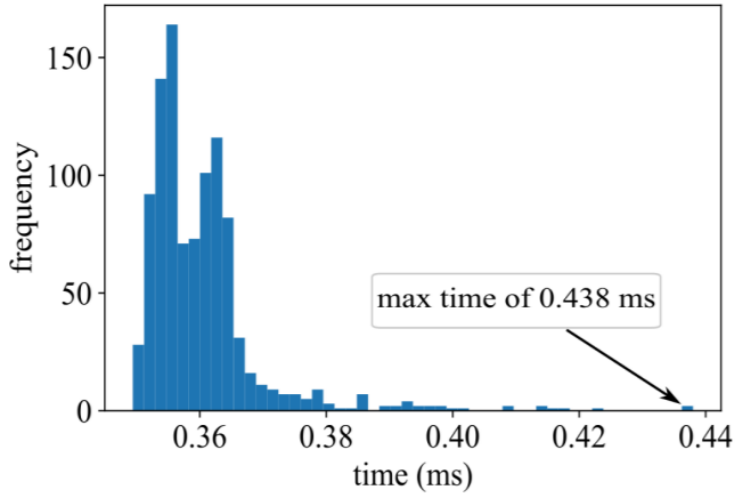
# Backup Slides on Hardware-software Co-design

# Selecting Proper Hardware for a Heterogenous System

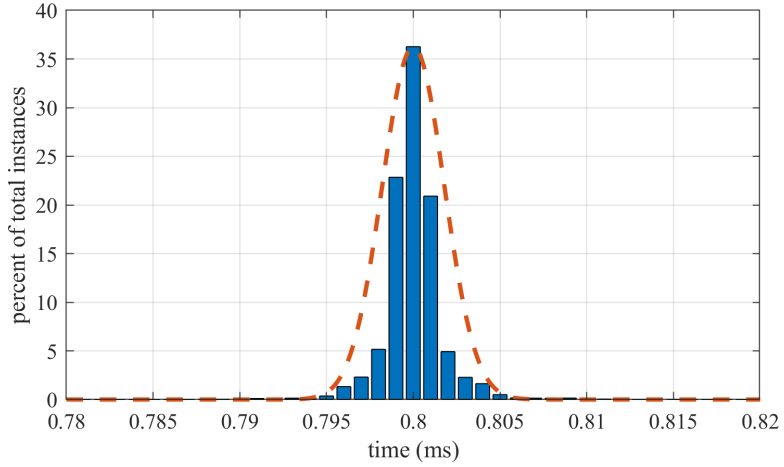
A few hardware considerations:

- General Purpose Operating System (GPOS):
  - Easy to program.
  - No guarantees on timing, no consideration of timing deadlines.
- Real-time Operating System (RTOS):
  - Still relatively easy to program.
  - Not faster; but provides decent bounding on jitter.
- Field Programmable Gate Array (FPGA):
  - Not simple to program.
  - Perfectly timing deterministic.

General Purpose Operating System (GPOS)



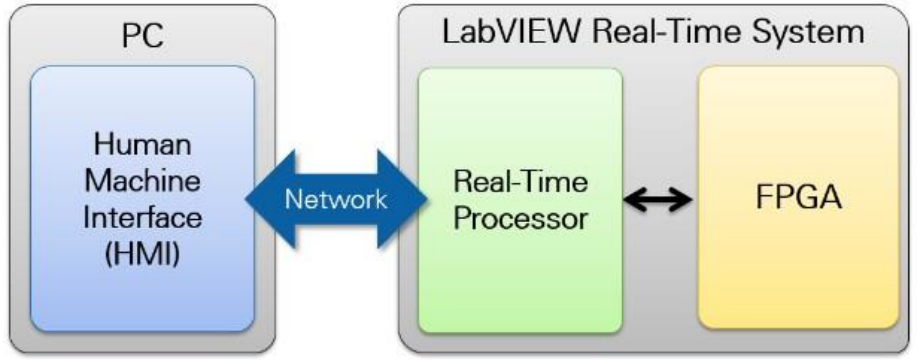

Real-time Operating System (RTOS)

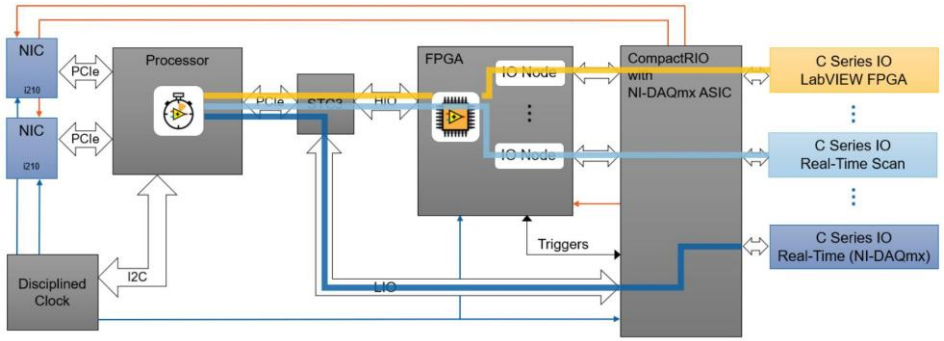


# Consequences of Missing Information at Strict Latencies

The exchange of data across processing systems has several key challenges for challenges:

- To use updated models, there needs to be some sort of AI or decision-maker one level up.
- To maintain high-rate transfers (to and from an FPGA), an RTOS will be needed between an GPOS and an FPGA.
- Information for decision making is typically done though a “publish and subscribe” transfer protocol, but this will introduce large uncertainties in timing.
- FPGAs are well suited for doing repetitive tasks, and as such should be considered for data fusion and filtration of sensor signals.



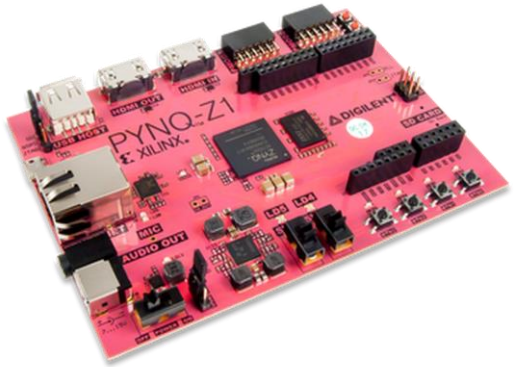

Embedded System Components. https://www.ni.com/en-us/support/documentation/supplemental/16/understanding-communication-options-between-the-windows-hmi--rt-.html

# Prototyping Ecosystems, and Future Deployments

- Different tools are being developed across a range of systems, each with their strengths and weaknesses.
- Long term, all modules will be turned into C/C++ and deployed onto a System on Chip (SoC) architecture.
- The FPGAs will be programmed in a hardware description language (HDL), that may come from the High-level synthesis (HLS) of C/C++ or converted down from high-level languages (MATLAB, Python, LabVIEW).

NI systems for programming FPGA in LabVIEW

Xilinx PYNQ SoCs for programming FPGAs in Python




Speedgoat Systems for programming FPGAs in MATLAB/Simulink