A MEMS-BASED ANALOG COMPUTER FOR EDGE AI COMPUTING

David Lin[1], Johan M. Reimann[1], Dorin E. Calbaza[1], Robert J. MacDonald[1], Zhihui Yang[1], Abdallah K. Alzubi[2], Mohammad S. Megdadi[2], and Fadi M. Alsaleem[2]*
[1]GE Aerospace Research, Niskayuna, New York, USA
[2]University of Nebraska-Lincoln, Lincoln, Nebraska, USA

ABSTRACT

Artificial Intelligence computing at the edge is often limited by size, weight and power. Existing approaches, based on digital computers, are inefficient due to the analog-to-digital conversion burdens and other processing bottlenecks inherently present in the von Neumann architecture. To address these challenges, a novel MEMS-based analog computer was developed and shown to achieve 300x reduction in power and 100x improvement in speed compared to a typical digital computer solving the same machine learning signal de-noising problem. The MEMS analog computer is enabled by two key innovations: MEMS continuous-time recurrent neural network and in-situ supervised training.

KEYWORDS

MEMS, Continuous-Time Recurrent Neural Network, Analog Computer, Edge AI

INTRODUCTION

Artificial intelligence (AI) has impacted our society in many profound ways. This trend will continue to accelerate. There are many exciting opportunities and challenges at the same time. For example, AI computing can consume significant power. Training a large language model like OpenAI's GPT-3, is estimated to consume 1,287 megawatt-hours (MWh) of electricity, which is comparable to the annual consumption of about 130 US homes [1]. The power-intensive technologies used for training AI models in data centers is generally not be suitable for edge applications. Many of these applications require interpreting and reasoning over analog data. Existing approaches, based on digital computers, are highly inefficient due to the analog-to-digital conversion burdens and other processing bottlenecks inherently present in the von Neumann architecture. Instead of building better graphics processing units (GPUs) or smaller transistors, a new approach to designing AI hardware for edge computing is needed. To this aim, we revisit the concept of analog computer which predates the invention of digital computers to address some of the edge deployment challenges.

An analog computer uses the continuous variation aspect of physical phenomena (analog signals) to model the problem being solved. Mead estimated a factor of 10^4 in difference between making a digital operation out of AND and OR gates and using the physics of the device to do the operation [2]. While analog computers can perform computing tasks with orders-of-magnitude lower power, they can only solve specific problems with low accuracy due to the noise and variabilities inherent to analog hardware. These limitations have largely kept analog computers out of the mainstream application domains. However, this is changed by the introduction of new AI approaches. Al computing is typically done by repeatedly applying a set of specific operations such as multiply–accumulate (MAC) which has lower accuracy requirement in comparison with scientific computing. Consequently, Analog computing is an intriguing candidate to disrupt existing AI hardware modalities by combating the growing energy footprint of AI.

Analog computing is especially powerful for the edge AI applications where size, weight and power (SWaP) is critical and most of the data collected is produced by analog sensors. By performing AI computing in the analog domain, the need for two-way analog-to-digital conversion can be eliminated. Analog, continuous-time processing does not require constantly moving data in and out of the memory which solves one of the major bottlenecks for digital architecture. This enables fast (real time, no latency), ultra-low power (mW to μW) and low-cost AI computing at the edge. It must be emphasized the power consumption by edge AI computing can quickly add up when aggregating over all the edge devices. The impact of edge AI power consumption to the society should not be overlooked.

Another problem of particular interest is the in-situ training for the edge applications. This problem is largely unexplored by the analog computing community. The conventional stochastic gradient descent (SGD) learning approach relies on the error backpropagation through multiple layers of neural network, which is very difficult to implement in analog hardware. However, this is a critical problem that must be solved. Analog devices do vary from one to the other and their characteristics can drift over time. In-situ training enables each of the edge AI system to learn its parameters based on the individual edge hardware. It also allows the edge AI system to recalibrate itself through re-learning to mitigate the drift of analog hardware.

MEMS analog computer is a special class of analog edge computer that uses signals from multiple physics domains to perform computing tasks. MEMS as sensors have gained popularity in various applications, from cell phones to medical equipment and health care. However, the concept of using MEMS for computing is relatively new. One of the earliest works proposed using MEMS as a computing unit [3] employs a network of MEMS oscillators with autocorrelative associative memory. Later, other work proposed using a network of coupled MEMS as a neural computing unit that mimics a continuous-time recurrent neural network (CTRNN) [4]. Another alternative was using a single MEMS for a delay-based reservoir computing (RC) unit [5-6]. Finally, the MEMS capacity to sense has inspired the new concept of simultaneous sensing and computing units [7]. Despite these promising findings, using MEMS for computing is still in its infancy. Scaling and physical training of the MEMS computing units are still open challenges. Toward these challenges, this paper presents a novel MEMS-based analog computer enabled by two key innovations: (1). A MEMS Continuous-Time Recurrent Neural Network (CTRNN) which leverages electrostatic MEMS neurons to achieve network computation in continuous time and analog domain; (2). In-situ supervised training of CTRNNs which allows learning directly on the edge hardware using received analog information with self-correction for the variation and drifting of the analog hardware.

METHODOLOGY

Computing Architecture

The heart of our work is the implementation of CTRNN using a network of MEMS devices. CTRNN is a well-known type of neural network with two key features: continuous-time analog signals and recurrent node states. A CTRNN has internal memory through self-feedback making it suitable for applications where salient information is in temporal sequences. Shown in Eq.1, the state value of each CTRNN node (neuron) is defined by an ordinary differential equation (ODE) that gives the network memory and

978-1-940470-05-4/HH2024/$25©2024TRF
DOI 10.31438/TRF.HH2024.8

enables it to model dynamic and oscillatory behavior [8].

$$\dot{y}_i = f_i(y_1, \dots, y_N) = \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{N} w_{ij}\sigma(y_j) + h_i + I_i\right), i = 1,2, \dots, N \quad (1)$$

where σ is an activation function, τ_i and y_i are the time constant and activation level of neuron i, respectively, w_{ij} is the connection strength between the i[th] neuron and the j[th] neuron, h is a bias term, I_i is the input to the i[th] neuron, and the dot operator represents the time derivative.

In a MEMS-CTRNN implementation, each neuron is a single MEMS device. The dynamic behavior of the MEMS device can be configured to mimic Eq.1 required to implement a CTRNN node. Typical MEMS second order dynamics equation can be written as Eq. 2, where Z is the MEMS displacement, ζ is the damping ration, ω_0 is the natural frequency, V_b, V_i and V_t are the control voltages and m, d, A are the MEMS design parameters.

$$\ddot{Z} + 2\zeta\omega_0\dot{Z} + \omega_0^2 Z = \frac{2}{m}\frac{\varepsilon A}{d^2}V_b V_i + \frac{\varepsilon A_t}{md^3}V_t^2 \mathrm{Z} \quad (2)$$

Assuming the 2[nd] order term is small, Eq 2. can be mapped to Eq 1. by rearranging it into Eq. 3

$$\dot{Z} = -\frac{Z}{2\zeta\omega_0}(\omega_0^2 - \frac{\varepsilon A_t}{md^3}V_t^2) + \frac{1}{2\zeta\omega_0}\frac{2}{m}\frac{\varepsilon A}{d^2}V_b V_i \quad (3)$$

Eq. 3 shows how we may design MEMS properly to achieve self-feedback (V_t term), time constant ($2\zeta\omega_0$) and coupling from the other MEMS nodes (V_i term). MEMS CTRNN has several advantages. First, the intrinsic nonlinearities of electrostatic MEMS can be harnessed to tune the device parameters, including effective stiffness, damping, and resonant frequencies. This leads to tunable and flexible CTRNN implementations. Second, the capacitive MEMS interface allows for significantly lower signal noise and power consumption compared to current-based computing architectures. Finally, MEMS has rich dynamics, including multiple resonant modes, which can be programmed and controlled separately to enable hyper-spectral computing.

Fig. 1 shows the IR image of the fabricated MEMS computing neuron. The MEMS neuron is excited electrostatically by feeding the input voltage signal to the excitation electrodes. The displacement of the MEMS neuron is measured for capacitive output. Detection electrodes are used to capture the differential capacitance of the MEMS neuron under dynamic excitation. Tuning electrodes are used to tune the MEMS internal dynamics and self-weight. The MEMS neurons were fabricated in the GE Microfabrication cleanroom using the GE Polaris process [9] which features 100um thick silicon SOI, wafer level encapsulation with through silicon via (TSV). The MEMS neuron has a size of 1500um x 400um. It has a mechanical resonant frequency of 4.5kHz and damping ratio of 1.61. The MEMS neuron is actuated by a DC bias of 10V (V_b) and a time-varying voltage signal (V_i). Fig. 2 shows the result of the displacement of the MEMS neuron by finite element simulation. The maximum displacement is 1.43um with V_b=10V and V_i=10V, which produces a differential capacitive output of 125fF.

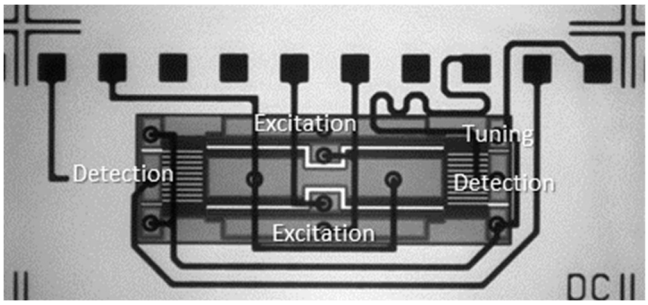

Figure 1: IR image of the MEMS computing neuron.

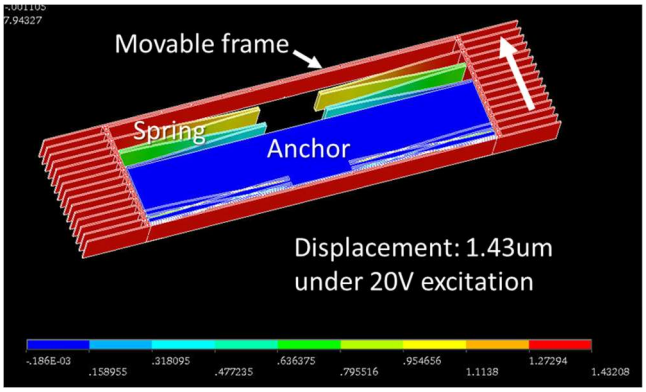

Figure 2: Simulated displacement of the MEMS neuron under 20V excitation by finite element analysis.

The MEMS computing neurons are the building blocks for a multi-layer hierarchical CTRNN shown in Fig. 3. The MEMS CTRNN has three layers: five MEMS neurons in the first layer, five forward neurons in the second layer, and a single neuron in the last output layer. The first layer is configured as five isolated MEMS devices (shown in Fig.1), followed by a dense layer that has five cells with RELU activation function. The last layer is the linear activation cell. In this architecture, each MEMS deflects by the applied input voltage. The measured capacitance is converted to voltage, multiplied by a weight, and then passed to the following RELU neuron layers as input. Finally, the output sum of the RELU neurons is entered into the final linear layer, providing the output.

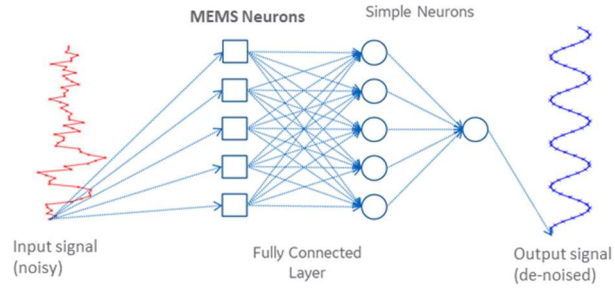

Figure 3: MEMS CTRNN architecture

In-Situ Supervised Training

The use of a CTRNN with continuous-time inputs and outputs introduces a special neural network training challenge that in the end becomes an important MEMS-CTRNN system feature. Conventional discrete-time neural networks use batch training. Their training data set is repeatedly retrieved and utilized in any order desired. Nearly all of machine learning works in this way today. It is difficult to do batch training with continuous-time data, unless the waveforms are sampled, digitized, stored, retrieved, converted back to analog, and reconstructed in time. That is undesirable and defeats our low-SWaP goal.

We have created a novel training process to train the MEMS-CTRNN *in situ* with ground truth continuous-time analog data presented only as it occurs and not stored or revisited. This will support training to learn a continuous output (estimation) and training to learn a discrete or probabilistic output (classification /detection). This continuous online incremental (at the edge) training will be an important feature of MEMS-CTRNN for many applications. One can think of this built-in hardware training feature as taking the place of TensorFlow or PyTorch, tools used to batch train many conventional neural networks.

A central challenge was to determine how to train the MEMS hardware to effectively learn temporal behaviors of the analog signals presented to the system. When implementing in-situ learning in analog hardware, the problem of flowing information backwards through the hardware is non-trivial. Consequently, we developed a target-propagation algorithm to solve the credit assignment problem and optimize the network weights. As opposed to the standard backpropagation algorithm where the weight gradient is identified by propagating the global loss error backwards though the network, the target-propagation uses an autoencoder type of architecture to establish intermediate targets. This approach has two main advantages: 1) The information needed at the output of each layer of the neural network can be obtained using a simple forward pass through the network and 2) the update equations are local to each layer meaning that to update the weight of layer i, only local information is needed.

Assuming the inputs to the MEMS neuron stay constant in a very small interval [t_0, t_0+Δt]), we can rewrite Eq.3 into a simplified ODE shown in Eq 4. Solving Eq. 4 leads to the update rules shown in Eq. 5 and Eq. 6.

$$\frac{dz_i}{d\hat{t}} = -\frac{z_i}{\tau_i} + f\left(\sum w_{ij}x_j\right), \hat{t}\sim[t_0, t_0+\Delta t] \quad (4)$$

Where $\tau_i = \frac{2\zeta\omega_0}{(\omega_0^2 - \frac{\varepsilon A_t}{md^3}V_t^2)}$, $V_i = \sum w_{ij}x_j$.

$$\frac{dz_i}{d\tau} = -\Delta t \cdot \frac{e^{\frac{-\Delta t}{\tau}}}{\tau^2} + wx \quad (5)$$

$$\frac{dz_i}{dw} = \tau \cdot x \quad (6)$$

The in-situ training was performed with the MEMS-CTRNN hardware in the loop. The target-propagation algorithm was implemented in MATLAB which allows for quick debugging and iterative development. The system is illustrated in Fig. 5. In this block diagram the area highlighted in pink shade depicts the hardware parts of the network, while the yellow shaded describes the part of the neural network ran on a MATLAB algorithm.

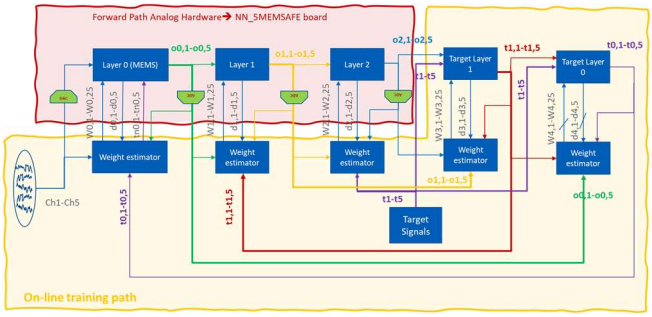

Figure 4: MEMS CTRNN in-situ training system includes the hardware-in-the-loop and the MATLAB real-time control algorithm.

The MEMS-CTRNN hardware's input analog signals are numerically computed in MATLAB, then sent through an USB interface to a microcontroller. The samples are then provided to 5 DAC channels which create the analog inputs processed by the neural network. The hardware portion of the neural network contains 3 layers of 5 neurons, the first layer (Layer 0) has the neuron activation function implemented using MEMS devices. The subsequent Layer 1 and Layer 2 are built using electronic means to implement the RELU activation function.

The Layer 0 of the neural network receives the input analog signals, combines them and provides them to the 5 MEMS based neurons for further processing. The outputs of the MEMS based neurons from Layer 0 are then combined by the neurons in Layer 1, and then the analog processing concludes with the Layer 2 of neurons. The outputs of each layer are sampled by ADC and then send by microcontrollers through multiple USB interfaces for further processing by the MATLAB algorithm.

In the software portion of the Neural Network, two layers of 5 neurons (Target Layer 1, and 2) compute the local targets, which are used by weight estimators to compute the weights necessary for each neuron present in the network, including the neurons implemented in hardware. All communication between the software portion and the hardware portion of the neural network passes through 6 USB interfaces implemented on Teensy devices present on the hardware section.

Limited by the current MEMS mechanical frequency, the analog signals are bandwidth limited at 5KHz for this implementation with a numerical sampling rate of 25KHz. The MATLAB algorithm receives all samples in real time and computes the weights necessary to adjust the neurons responses. The weights update takes place at a lower rate, with an epoch update rate faster than 10Hz.

Fig. 5 shows the hardware implementation of the MEMS CTRNN using MEMS neurons and programmable analog arrays. The hardware is implemented as a Motherboard in which the MEMS neurons are inserted as Daughterboards. These MEMS daughterboards are implementing the electronic means necessary to convert the capacitive response of each MEMS to an analog signal to be provided to the Layer 1 of the neural network. Each daughter board contains a driver circuit that translates the input signal to the range the MEMS prefers. The daughterboard also contains oscillator, demodulator, and amplifier circuits implementing a lock-in amplifier necessary to produce the desired output signal.

The motherboard contains the Teensy devices used for communication with a host PC and to sample the signals produced by each neural network layer. The motherboard also contains the DACs needed to produce 5 analog input signals and multiple DC bias signals. In addition, 5x3 field programmable analog array (FPAA) that combine multiple analog inputs to provide them to the activations functions specific to each layer are also implemented in the motherboard.

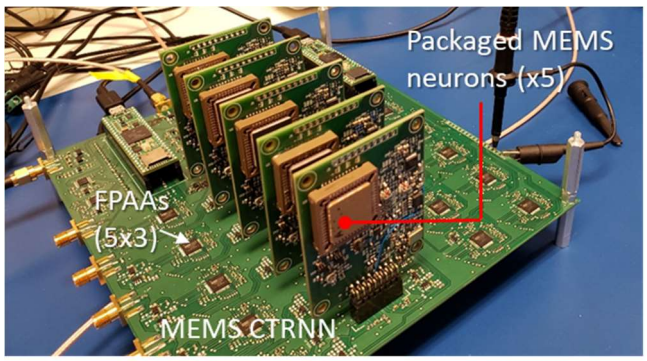

Figure 5: MEMS CTRNN hardware implemented using MEMS neurons and programmable analog arrays.

RESULTS

A machine learning problem of signal de-noising was used to evaluate the MEMS-CTRNN hardware prototype and its in-situ training performance. This problem aims to recover the original ground truth signal embedded in a very noisy signal. The input noised signal x_t is a sine wave of frequencies ranging between 300Hz -600Hz superimposed by multiple noise signals of frequencies between 500-3750 Hz. Fig. 6 shows the inference path testing results from the MEMS-CTRNN hardware prototype. The MEMS network weights were trained offline using a simulation

model for the network. It can be observed the neural network reduced the amplitude of the unwanted noise components present on the input signal and produced a cleaner sinusoidal signal at the output according to model prediction.

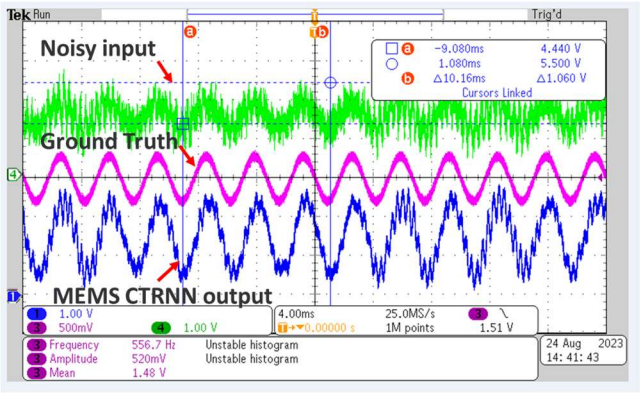

Figure 6: Validation of MEMS-CTRNN inference for solving a signal denoising machine learning problem.

Fig. 7 shows the validation of MEMS CTNN in-situ training for the signal-denoising problem. By feeding the training signal and ground truth into the MEMS CTRNN hardware, the MEMS analog computer was able to learn to adaptively filter a noisy signal to recover the clean signal of ground-truth and compute the weights *in situ*. These weights were later used in a test run to validate the accuracy of in-situ training (Fig. 7b).

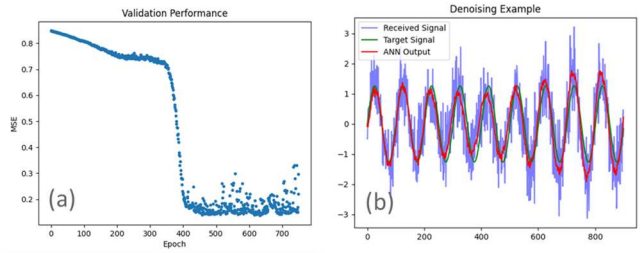

Figure 7: In-situ training using the MEM-CTRNN, (a). progress of Mean Squared Error over the computing iterations during the training; (b) validation run to compare received signal vs recovered signal (CTRNN output) vs ground truth.

Table 1 summarizes the benchmarking study to compare the MEMS analog computer and a digital workstation in solving the same signal denoising training problem. Our MEMS-CTRNN prototype solves the signal denoising problem in <10 minutes with 2Watt-hours energy consumed while the digital computer takes 15 hours and 600 Watt-hours to do the same. Even at this initial stage of development, the MEMS analog computer has already demonstrated 300x reduction in power consumption and 100x improvement in speed over a digital computer.

CONCLUSIONS

In this paper, we demonstrate a high-speed, energy-efficient MEMS analog computer which can perform inference and in-situ training computation for machine learning at the edge. Our hardware prototype has been validated by using a signal denoising problem and shown a 300x reduction in power consumption and a 100x increase in computational speed compared to a typical digital computer. The MEMS analog computer can provide a completely new way to overcome the SWaP challenges for many edge AI applications. Future work will include: 1) proving reliability, 2) expanding applications, 3) improving integration and scalability.

Table 1: Computational performance benchmarking: MEMS analog computer vs digital computer.

	Baseline	**MEMS CTRNN**
Architecture	Xeon® Gold 6148 Processor, 20 cores	MEMS analog computer
size	~20x 20 in^2	1.5 x 5mm^2 (MEMS die)
Speed	~15 hours	~10 minutes
Energy consumed	600 Watt-hours	2 Watt-hours

Based on measurement data of solving the same signal denoising learning problem using a digital computer and the MEMS analog computer

ACKNOWLEDGEMENTS

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via [2022-21102100011]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U. S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

REFERENCES

[1] A. de Vries, "The Growing Energy Footprint of Artificial Intelligence", *Joule*, 7 1-4, (2023).

[2] C. Mead, "Neuromorphic Electronic Systems", Proceedings of the IEEE, *Vol. 78, No. 10,* (1990).

[3] A. Kumar and P. Mohanty, "Autoassociative Memory and Pattern Recognition in Micromechanical Oscillator Network", *Sci Rep*, 7(1):411, (2017).

[4] F. M. Alsaleem, M. H Hasan, and M. K Tesfay, "A MEMS nonlinear dynamic approach for neural computing", *Journal of Microelectromechanical Systems*, 27(5):780-789, (2018).

[5] M. H. Hasan, A. Al-Ramini, E. Abdel-Rahman, R. Jafari, and F. M. Alsaleem, "Colocalized sensing and intelligent computing in micro-sensors", *Sensors*, 20(21):6346, (2020).

[6] T. Zheng, W. Yang, J. Sun, X. Xiong, Z. Wang, Z. Li, and X. Zou, "Enhancing performance of reservoir computing system based on coupled MEMS resonators", *Sensors*, 21(9):2961, (2021).

[7] H. Nikfarjam, M. Megdadi, M. Okour, S. Pourkamali, and F. Alsaleem, "Energy efficient integrated MEMS neural network for simultaneous sensing and computing", *Communications Engineering*, 2(19), (2023).

[8] B. Chandrasekaran, "Intelligence as adaptive behavior: An experiment in computational neuroethology" (Vol. 6). Academic press, (2013).

[9] D. Lin, R. MacDonald, D. Calbaza, J. Popp, T. Johnson, E. Andarawis and M. Aimi, "Polaris - A Low-Cost MEMS Fabrication Platform for Navigation-Grade Inertial Sensors", IEEE/INERTIAL, March 2021, (2021).

CONTACT

*D. Lin, tel: +1-518-387-6096; linyiz@ge.com