micromachines

Article

Exploiting Pull-In/Pull-Out Hysteresis in Electrostatic MEMS Sensor Networks to Realize a Novel Sensing Continuous-Time Recurrent Neural Network

Mohammad H Hasan [1], Amin Abbasalipour [2], Hamed Nikfarjam [2], Siavash Pourkamali [2], Muhammad Emad-Ud-Din [3], Roozbeh Jafari [3,4,5] and Fadi Alsaleem [6,*]

[1] Department of Earth and Space Sciences, Columbus State University, Columbus, GA 31909, USA; hhasan_mohammad@columbusstate.edu
[2] Department of Electrical and Computer Engineering, University of Texas at Dallas, Dallas, TX 75080, USA; axa147730@utdallas.edu (A.A.); hamed.nikfarjam@utdallas.edu (H.N.); Siavash.Pourkamali@utdallas.edu (S.P.)
[3] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA; emaad22@tamu.edu (M.E.-U.-D.); rjafari@tamu.edu (R.J.)
[4] Department of Biomedical Engineering, Texas A&M University, College Station, TX 77843, USA
[5] Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA
[6] Durham School of Architectural Engineering and Construction, University of Nebraska—Lincoln, Omaha, NE 68182, USA
* Correspondence: falsaleem2@unl.edu; Tel.: +1-(402)-554-3283

Citation: H Hasan, M.; Abbasalipour, A.; Nikfarjam, H.; Pourkamali, S.; Emad-Ud-Din, M.; Jafari, R.; Alsaleem, F. Exploiting Pull-In/Pull-Out Hysteresis in Electrostatic MEMS Sensor Networks to Realize a Novel Sensing Continuous-Time Recurrent Neural Network. *Micromachines* **2021**, *12*, 268. https://doi.org/10.3390/mi12030268

Academic Editor: Yao-Feng Chang

Received: 2 February 2021
Accepted: 3 March 2021
Published: 5 March 2021

Abstract: The goal of this paper is to provide a novel computing approach that can be used to reduce the power consumption, size, and cost of wearable electronics. To achieve this goal, the use of microelectromechanical systems (MEMS) sensors for simultaneous sensing and computing is introduced. Specifically, by enabling sensing and computing locally at the MEMS sensor node and utilizing the usually unwanted pull in/out hysteresis, we may eliminate the need for cloud computing and reduce the use of analog-to-digital converters, sampling circuits, and digital processors. As a proof of concept, we show that a simulation model of a network of three commercially available MEMS accelerometers can classify a train of square and triangular acceleration signals inherently using pull-in and release hysteresis. Furthermore, we develop and fabricate a network with finger arrays of parallel plate actuators to facilitate coupling between MEMS devices in the network using actuating assemblies and biasing assemblies, thus bypassing the previously reported coupling challenge in MEMS neural networks.

Keywords: neuromorphic computing; MEMS; Sensor Network; CTRNN

1. Introduction

Wearable devices promise great improvement in human quality of life by enabling health monitoring and diagnostics via human activity recognition (HAR), which is essential for fitness tracking, productivity assessment, and comfort management. Wearable devices rely on biological data measured through sensors such as accelerometers. The data points are then processed through complex machine learning schemes to determine the biological state. However, as wearable electronics are limited in power and space, complex machine learning approaches cannot be efficiently implemented locally. Instead, biological data are typically sent to the cloud for processing, causing power loss through wireless communication, and posing security risks in such systems.

Neuromorphic computing, first introduced by Mead [1], is a viable solution to the challenge of local computing in wearable devices. Neuromorphic computing started as an idea of using transistors in the subthreshold regime to simulate the response of biological neurons. More recently, this has evolved into a set of computing schemes that utilize analog devices to perform computing [2]. Neuromorphic computing has shown great power-saving potential while maintaining substantial computational ability [2]. Spiking neural networks (SNNs) are considered the most well-known neuromorphic computing schemes. In such a scheme a network of analog devices is used to produce a spiking response, like that observed in biological neurons [3]. Neuromorphic sensors have also been introduced previously to simulate the behavior of sensory organs and provide visual sensing [4], audio sensing [5], and olfactory sensing [6]. Such devices produce asynchronous spiking outputs corresponding to changes in the measured signals. However, for both neuromorphic implementations (sensors and SNNs) additional components are needed to translate the spiking signals into digital signals for further processing, increasing the footprint of such devices. Moreover, neuromorphic sensors require neuromorphic, spike-based processors, adding to the system size and power requirements [3].

Inspired by the neural system of very tiny biological systems, such as some insects as shown in Figure 1 [7] an alternative means of computing that addresses such concerns is colocalized sensing and computing. In this approach, some of the measured signals are preprocessed at the sensor level. Sensory information is consequently produced or post-processed by a digital processor. Microelectromechanical Systems (MEMS) sensors have been previously considered for this type of computing process. Networks of MEMS oscillators were shown to be able to perform computing through oscillator synchronization [8,9]. However, phase comparison between MEMS oscillators and the need to maintain specific initial conditions of oscillators are challenges that require addressing. More recently, a single MEMS device has been shown to perform computing through reservoir computing, by utilizing time-multiplexing to create temporally coupled virtual nodes [10,11]. However, in this approach, the response of the MEMS device is required to be sampled at very high rates (tens or hundreds of kHz). Furthermore, delayed feedback is required, which further complicates the required electronics.

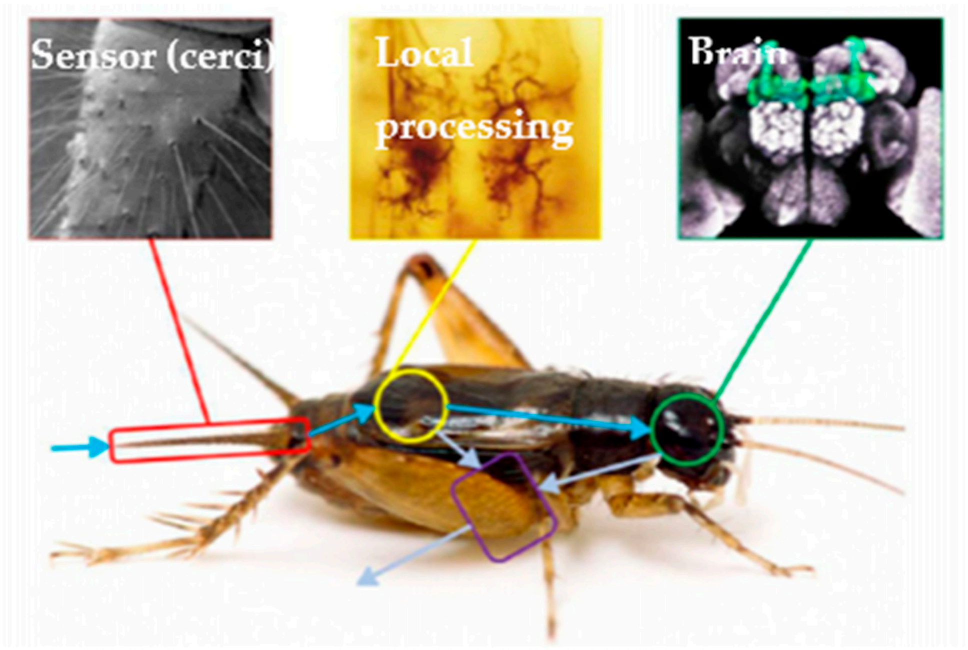

Figure 1. New findings in insect neural systems reveal that they have local integrated sensing and computing neurons to reduce computing demands at the central processing unit (the brain) [7].

In our previous work, we presented the novel use of MEMS electrostatic sensor dynamics with special geometric nonlinearities to naturally solve the continuous-time recurrent neural network (CTRNN) equations [12,13]. In that implementation, there is no need for a digital computer to solve the CTRNN equations. As an application, it was shown that the dynamics of eight coupled MEMS devices can be trained to perform a classification and object tracking of a mobile robot application [13]. In this paper, the concept of MEMS-based CTRNN is expanded to enable the use of almost any type of electrostatic MEMS to perform CTRNN computing. This new implementation relies on the hysteresis due to the Pull-in/Pull-out behavior that inherently exists in almost any parallel-plate electrostatic MEMS transducer. Furthermore, compared to previous works, the concept of colocalized sensing and computing eliminates the need for additional sensors at the network input layer and the need for capacitive measurement elements, thus reducing the network footprint.

The organization of this article is as follows: In Section 2 The theory for CTRNN and the use of MEMS Pull-in/out instability to perform CTRNN computation is introduced. In Section 3, we demonstrate through simulation model for a small network of off-the-shelf MEMS accelerometer sensors to perform a simple classification problem and we present the challenge for experimental implantation. In Section 4, we design and fabricate the first MEMS CTRNN to perform simultaneous sensing and computing. We also provide some preliminary results and motivate the need for more thorough parameter optimization for this novel network to perform classification problems experimentally. Finally, we provide conclusions and future work in Section 5.

2. Theory and Methodology

2.1. RNN vs. CTRNN

Recurrent neural networks (RNN), unlike traditional feed-forward neural networks (FFNN), utilize internal memory through self-feedback to preserve the sequences of input data during training [14,15]. Thus, the RNNs have shown great success in sensory applications such as image, video, and audio processing, as well as in optimization, associative memories, and controls [14]. A special, yet very complex form of RNN known as a CTRNN [16], uses differential equations to describe the activation level of the neurons (see Equation (1) below). To perform a certain classification problem, the self-coupling and cross-coupling weights between different neurons of a CTRNN are determined through the training performed during the design phase of the network.

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

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 ith neuron and the jth neuron, h is a bias term, I_i is the input to the ith neuron, and the dot operator represents the time derivative.

Figure 2 shows schematic diagrams comparing the structure of a single feedforward neuron (FFN), a recurrent neuron (RN), and a continuous-time recurrent neuron (CTRN). The schematics show that while having self-feedback is the main difference between CT/RN and the FFN, the differential equation is the main difference between the RN and the CTRN. The first-order differential equation with a time constant τ of the CTRN model acts as a low-pass filter. The function of τ_i is to produce a resistance to reject the input from other neurons and try to maintain the influence of previous inputs on the neuron. Larger τ_i means stronger resistance and a slower activation process. In other words, a neuron with a large time constant attempts to store the history information and needs a longer time to accept new inputs. The value of τ_i thus has a profound impact on the overall model learned by the CTRNN network. Moreover, it provides the CTRNN a learning capability comparable to the state-of-art advanced, yet complex, recurrent neural networks such as Long Short-Term Memory (LSTM) neural network. As such, CTRNNs have emerged as a very attractive machine learning option as they require fewer neurons for high-level learning. For example, a CTRNN made of only four CTRNs was needed to learn eight wrist trajectories from its acceleration measurements [17], where 128 RNs were needed to perform a similar task [18]. However, CTRNNs are computationally expensive for real-time implementation as they require simultaneous solutions of highly coupled multiple differential equations. This makes them unsuitable for many emerging applications such as wearable devices with limited memory and processing capabilities [19].

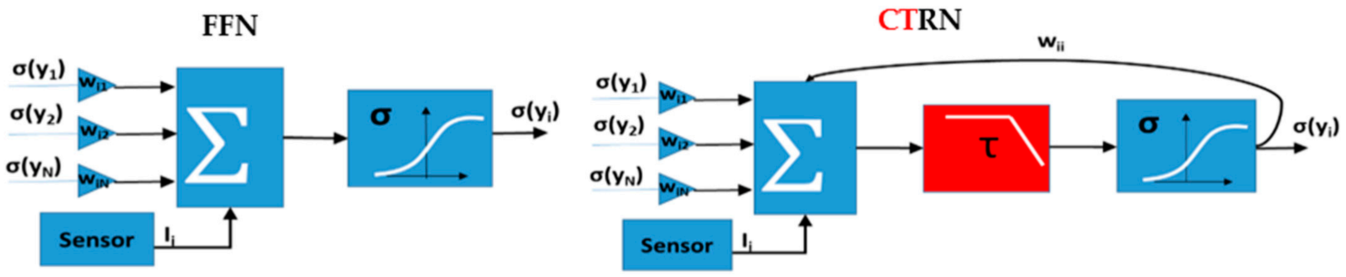

Figure 2. The differences between FFN and CT/RN. While CTRN and RN have internal memory through self-feedback, a CTRN approximates the response of a group of RNs by having a first-order differential equation.

2.2. MEMS-Based CTRNN Approach

To overcome the challenges, in previous work, we have identified nonlinearity and hysteresis as essential properties for CTRNNs to perform computing. Thus, we have shown that systems exhibiting these properties, such as a network of coupled bi-stable MEMS devices, are candidates for performing CTRNN computing in an analog fashion [12,13]. However, while that work demonstrated an efficient way to perform CTRNN computing using MEMS devices, it followed a typical machine learning structure that separates the input (sensor) layer from the computing layer (Figure 3a). As MEMS devices were originally designed to be sensors, in this work, we expand the MEMS novel computing concept to allow a MEMS bi-stable network to perform simultaneous sensing and computing (Figure 3b). Thus, eliminating the need for the complex sensor interfaces and signal conditioning circuits to perform similar computation.

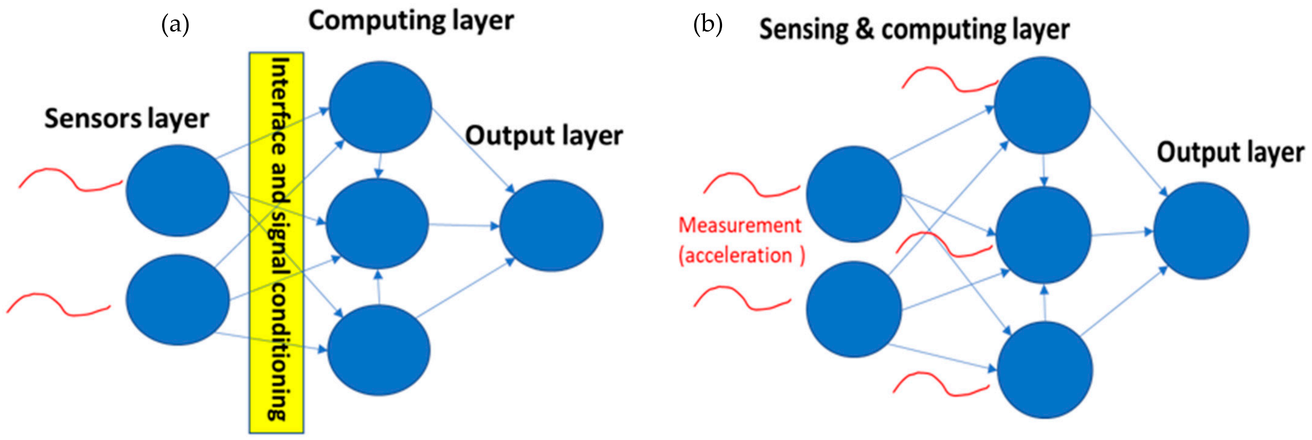
(a)
Computing layer
Sensors layer
Interface and signal conditioning
Output layer
(b)
Sensing & computing layer
Output layer
Measurement (acceleration)

Figure 3. (**a**) Our previous attempt for building a MEMS CTRNN follows a typical machine learning approach that separates sensing and computing. While it provides an efficient way to do computing, it still requires the complex sensor reading and interface between the input and output layers. (**b**) This paper's contribution is to investigate and demonstrate a MEMS CTRNN that no longer separates between the input and output layers.

2.3. Modeling

In the new MEMS sensing and computing implementation, we approximate the response of each MEMS device in the N-MEMS network as a single-degree of freedom spring-mass-damper system, shown in Figure 4 and governed by (1):

$$m_i\ddot{z}_i(t) + c_iz_i(t) + k_iz_i(t) = \frac{\varepsilon A\left(\sum_{j=1}^{N} w_{ij}^{*}V_jU\left(z_j(t) - d_j\right) + V_{bi}\right)^2}{2(d_i - z_i)^2} - m_i\ddot{y}(t) \quad (2)$$

where $z_i(t) = x_i(t) - y_i(t)$ is the relative deflection of the ith MEMS device at time t, computed as the difference between the absolute MEMS deflection $x_i(t)$ and the base (sensor casing) displacement $y_i(t)$. The ith MEMS device in the network has a mass m_i, damping constant c_i and stiffness k_i. The surface area of each MEMS electrode is A_i and the separation between the stationary and moving electrodes of the MEMS devices, when at rest, is d_i. Each MEMS device is electrostatically actuated using a signal $V_i = \sum\left(w_{ij}V_jU\left(z_j(t) - d_j\right)\right) + V_{bi}$ composed of a bias voltage V_{bi} and an external weighted signal from the other MEMS devices in the network $w_{ij}^{*}V_jU\left(x_j(t) - d_j\right)$, where $U\left(x_j(t) - d_j\right)$ is a unit step function that activates when $x_j(t) \geq d_j$, representing a switching action and w_{ij}^{*} is the connection weight from the jth MEMS device to the ith MEMS device. Each MEMS device in the network may experience pull-in instability if the total applied voltage (pull-in voltage) produces an electrostatic force that exceeds the mechanical stiffness of the structure. This leads to the collapse of the MEMS structure and the closing of an electrical circuit (ON state). However, due to internet hysteresis behavior, the voltage needs to be reduced to a value smaller than the pull-in voltage (release voltage) to release the proof mass (OFF state). To simulate the impact of pull in/out hysteresis in the MEMS network response in (2), we limit the MEMS deflection to a threshold value x_s, using mechanical stoppers, where $0.33d < x_s <$ d, where higher x_s values indicate more hysteresis. In real MEMS implementation, hysteresis can be controlled by the thickness of the thin intermediate dielectric layer on the substrate [20,21].

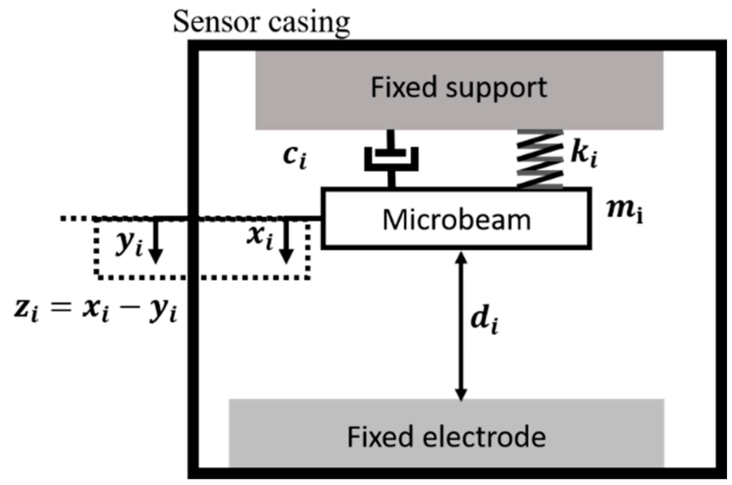

Figure 4. MEMS Schematic.

To rewrite (2) in a similar form to the CTRNN equation in (1), we first non-dimensionalize (2) using the nondimensional parameters in (3):

$$\hat{z}_i = \frac{z_i}{d_i}, \hat{t} = \frac{t}{T_i}, T_i = \frac{1}{\omega_{n_i}}, \zeta_i = \frac{c_i}{2m_i\omega_{n_i}} \quad (3)$$

Applying the substitutions in (3) to (2) yields (4):

$$\ddot{\hat{z}}_i + 2\zeta_i\dot{\hat{z}}_i + \hat{z}_i = \frac{\varepsilon A\left(\sum_{j=1}^{N} w_{ij}^{*}V_jU\left(\hat{z}_j(t) - 1\right) + V_{bi}\right)^2}{2d_i^3k_i(1 - \hat{z}_i)^2} - \frac{m_i}{\omega_{ni}^2 d_i}\ddot{y}(t) \quad (4)$$

where $\omega_{ni} = \sqrt{k_i/m_i}$ is the natural resonance frequency of the ith MEMS device, and ζ_i is its damping ratio. One can show through dimensional analysis that, the first term in (4) can be dropped if the MEMS resonance frequency is sufficiently high, for a given damping ratio. This condition is easy to satisfy when the MEMS device is operated under atmospheric pressure due to the prevalence of squeeze-film damping [12]. Therefore, (4) can be rewritten in a form like the CTRNN equation, as follows:

$$\tau_i \dot{\hat{z}}_i(t) = -\hat{z}_i(t) + \sigma_M\left(\sum_{j=1}^{N} w_{ij} V_j U\left(\hat{z}_j(t) - 1\right) + \theta_i, \hat{z}_i(t)\right) + I_i(t) \tag{5}$$

where $\tau_i = 2\zeta_i$ is the MEMS time constant, $I_i(t) = \frac{m_i}{\omega_{ni}^2 d_i}\ddot{y}(t)$ is the acceleration input to the MEMS device, $w_{ij} = \frac{\varepsilon A w_{ij}^*}{d_i\sqrt{2k_i}}$ is the effective connection weight between the MEMS devices, $\theta_i = \frac{\varepsilon A V_{b_i}}{d_i\sqrt{2k_i}}$ is the bias signal and $\sigma_M(\alpha, \beta) = \alpha^2/(1-\beta)^2$ is a nonlinear transformation corresponding to the nonlinear electrostatic forcing on the MEMS device. The parameters to be optimized in the MEMS CTRNN to achieve a certain functionality are ω_{n_i}, k_i, d_i V_{b_i}, and $w_{i,j}^*$.

2.4. Weight Implementation

To achieve coupling with adjustable weights between the MEMS devices in a computing network, we have used operational amplifiers.[13] However, this approach requires extra electronics and power and scales poorly as the network size increases. Moreover, it cannot be used to represent negative weights as while an operational amplifier can invert the input voltage polarity, a MEMS according to (4) only responds to the square of the voltage. To address this challenge, we have adopted the electrostatic mechanical coupling mechanism. This approach has been already used by our team to realize a digital mechanical MEMS accelerometer [22].

In this approach, electrostatic parallel plate finger arrays will be adopted to realize the coupling between the neurons of the MEMS-based CTRNN. For example, in the sensing and computing layer, a proof mass i is coupled to other proof masses by different sets of fingers as shown in Figure 5. The activation voltage of each j set of fingers acting on proof mass i is controlled by the corresponding proof mass j status. Thus, the total electrostatic forces acting on a proof mass i can be described by:

$$F_{ic} = \frac{\varepsilon A_1\left(\sum_{j=1}^{N} \mp n_j V_{out,j}^2\right)}{2d_1^2} \tag{6}$$

where $V_{out,j}(u_j)$ is the output voltage from the jth proof mass, and n_j, A_1, and d_i are the number of parallel fingers controlled by the proof mass j, the overlapping area of the parallel fingers, and the nominal separation between the fingers, respectively.

The coupling effect may be positive or negative depending on the relative position between the stationary fingers and the moving fingers (attached to proof mass). For example, for the proof mass$_i$ shown in the figure, the voltage signal for fingers activated by proof mass$_{j=2}$ are associated with a positive effect because they produce a force that moves mass$_i$ toward its fixed electrode. On the other hand, the fingers activated by mass$_{j=1}$ are associated with a negative effect as it produces a force in the opposite direction. The former operation is demonstrated in the top schematic of Figure 5, where the mass$_{j=1}$ that is oriented along the y-direction may receive enough acceleration to bring it into contact with its fixed substrate. This in turn activates the applied voltage V_1 on its corresponding fingers acting on mass$_i$ to pull it away from its fixed substrate.

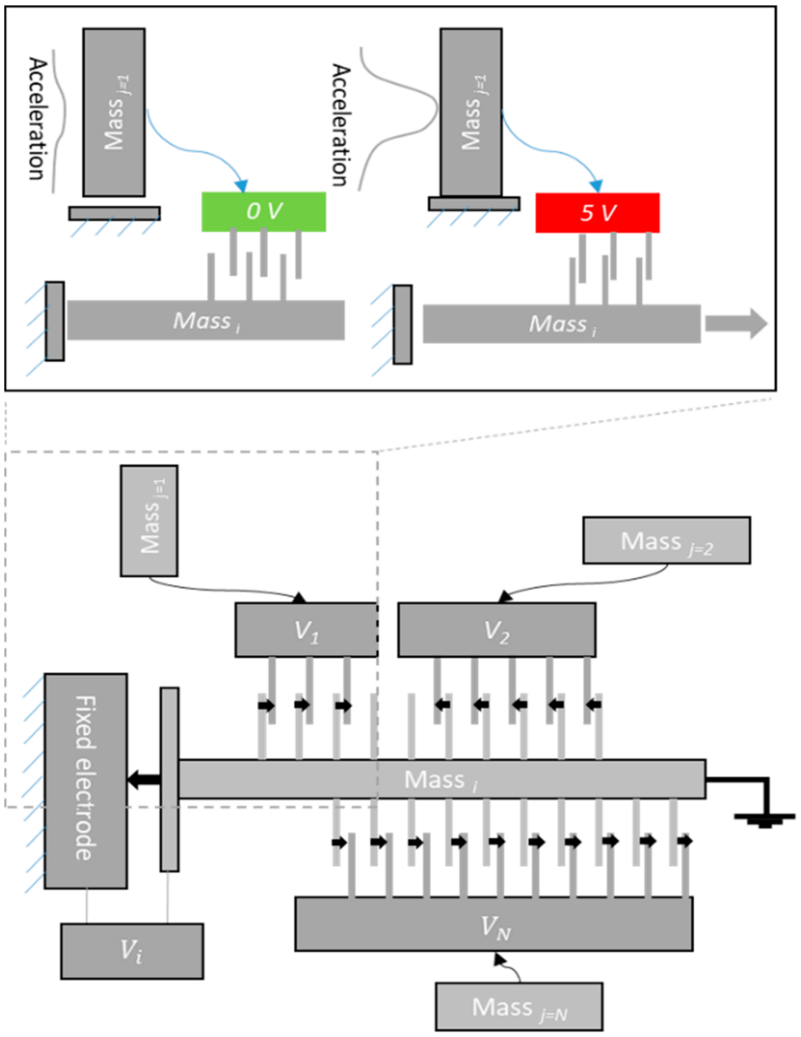

Figure 5. The finger arrays approach to realize electrostatic coupling between the MEMS CTRNs.

3. Waveform Classification Using a Commercial Off-the-Shelf Accelerometer

Classification is one of the most popular tasks in the machine learning literature. For this work, we consider a simple classification task as a test for the computational potential of a network of MEMS devices. The task here involves the non-trivial problem in the literature [23,24] to classify an input waveform into either 'Square' signal or 'Triangular' signal, as shown in Figure 6. The input waveforms are supplied as acceleration waveforms. We note here that, unlike other physical implementations of neural networks where inputs are electrical signals, the MEMS network simultaneously performs sensing and computing. For the MEMS CTRNN to perform the computational task properly, the size of the network and the connection weights between the MEMS devices are optimized. Optimization was performed manually by starting from a ladder diagram optimization scheme, assuming each MEMS device is a relay switch with no memory. Under that assumption, five MEMS devices are required to perform the computational task. The number of MEMS devices required is reduced to three by taking advantage of the dynamics of MEMS devices, namely inertia and pull in/out hysteresis.

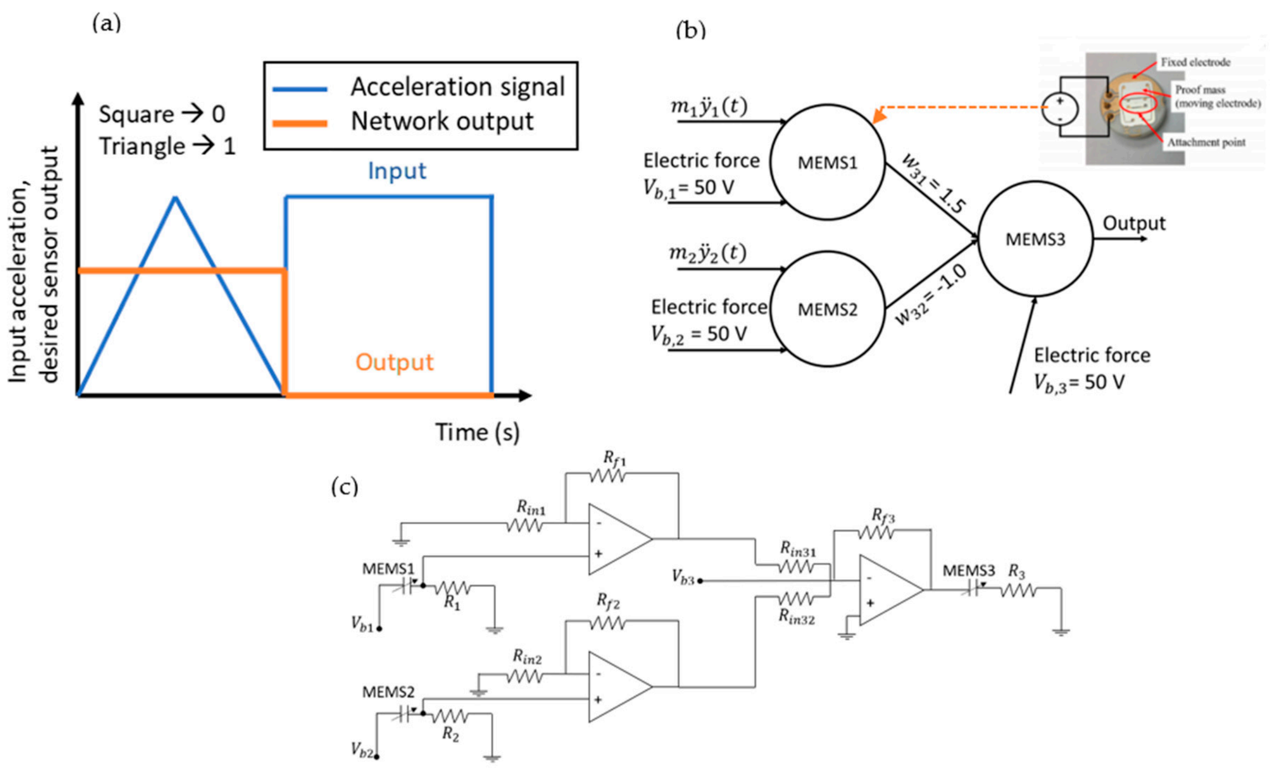

Figure 6. Classification task considered in this work. (**a**) Visualization of the binary classification problem. (**b**) MEMS network used for classification. The network is composed of three identical devices. Two devices receive an input acceler-ation signal and one device performs classification. (**c**) A connection circuit for the MEMS network.

The bias voltages were chosen such that $V_{b,1} > V_{b,2}$ to force MEMS1 to pull-in ahead of MEMS2 when supplied by a ramped signal. MEMS1 and MEMS2 pull-in nearly simultaneously when a square acceleration signal is applied to the CTRNN. The connection weights between the MEMS devices in the network are also optimized manually by taking advantage of the 'selection properties' of a CTRNN [12,16]. Due to selection, the influence of input signals depends on the amplitude of the input signals as well as their temporal order. We note here that, due to our chosen method of weight optimization, the MEMS CTRNN will be able to classify any quasi-static acceleration signal. However, at acceleration frequencies close to the natural frequencies of MEMS1 and MEMS2, this method fails. Other optimization methods would be required to enable the classification of such signals.

For our task, a model for a network of identical commercial off-the-shelf accelerometer doubly cantilever MEMS accelerometer devices fabricated by Sensata technologies was used. The accelerometer is designed to measure low g acceleration, but if a high bias voltage is applied, it can pull-in and acts as a switch. The network is assumed to be coupled using operational amplifiers [12], in a fashion similar to that shown in Figure 6c. Here, the resistor values would be chosen for two purposes: reducing the current follow at pull-in; and tuning the connection weights between the MEMS devices. The parameters of the MEMS devices are presented in Table 1. This MEMS device is shown in the insert of Figure 6b. The MEMS devices in this circuit can be connected in series to large resistor to reduce the current following in the circuit at pull-in, which would otherwise burn the MEMS circuit. Additional information about the sensor and its model can be found in [12]. Here, it is assumed that MEMS1 and MEMS2 are input neurons, directly influenced by the acceleration signal. MEMS3, however, to simplify the calculation, is designed to be oblivious to the acceleration signal. This can be achieved by rotating MEMS3 such that the acceleration signal is perpendicular to the MEMS motion.

Table 1. MEMS parameters.

MEMS Parameter	Value
Length *(l)*	9 mm
Width *(b)*	5.32 mm
ε	8.85×10^{-12} F/m
Gap *(d)*	42 µm
Stiffness *(k)*	215 N/m
Mass (m)	143 mg
Dampign cofficient (c)	0.351 N. s/m
Bias MEMS1 $(V_{b,1})$	50 V
Bias MEMS2 $(V_{b,2})$	50 V
Bias MEMS3 $(V_{b,3})$	50 V
Weight MEMS3$\rightarrow$ 1 (w_{31})	1.5
Weight MEMS3 $\rightarrow$ 2 w_{32}	-1
Threshold deflection (x_s)	30 µm

As a demonstration, the MEMS CTRNN is subjected to a sequence of a square and triangle signal with an amplitude $\ddot{y} = -5g$. The results of the MEMS CTRNN are shown in Figure 7. This figure is produced using a Matlab code, assuming that each MEMS device acts as a perfect switch with output $V_{out,i} = V_i U(x - x_s)$. The simulated shock signal excites both MEMS1 and MEMS2 (Figure 7a,b, respectively). Initially, when a triangle signal is observed, MEMS1 pulls-in (at around -2 g) first, ahead of MEMS2, due to its higher bias voltage. Consequently, MEMS3 pulls-in. When the acceleration signal ramps to -3 g, MEMS2 pull-in. Since MEMS2 has a negative connection weight, it reduces $V_3(t)$ to a value below the MEMS3 pull-in voltage. However, this reduction is insufficient to release MEMS3, due to the hysteresis at pull-out. Thus, MEMS3 remains pulled-in until the acceleration amplitude is reduced to below -2 g. Hence, despite MEMS1 and MEMS2 eventually pulling-in when they experience a triangle-shaped acceleration signal, the difference in pull-in timing ultimately results in triangle classification.

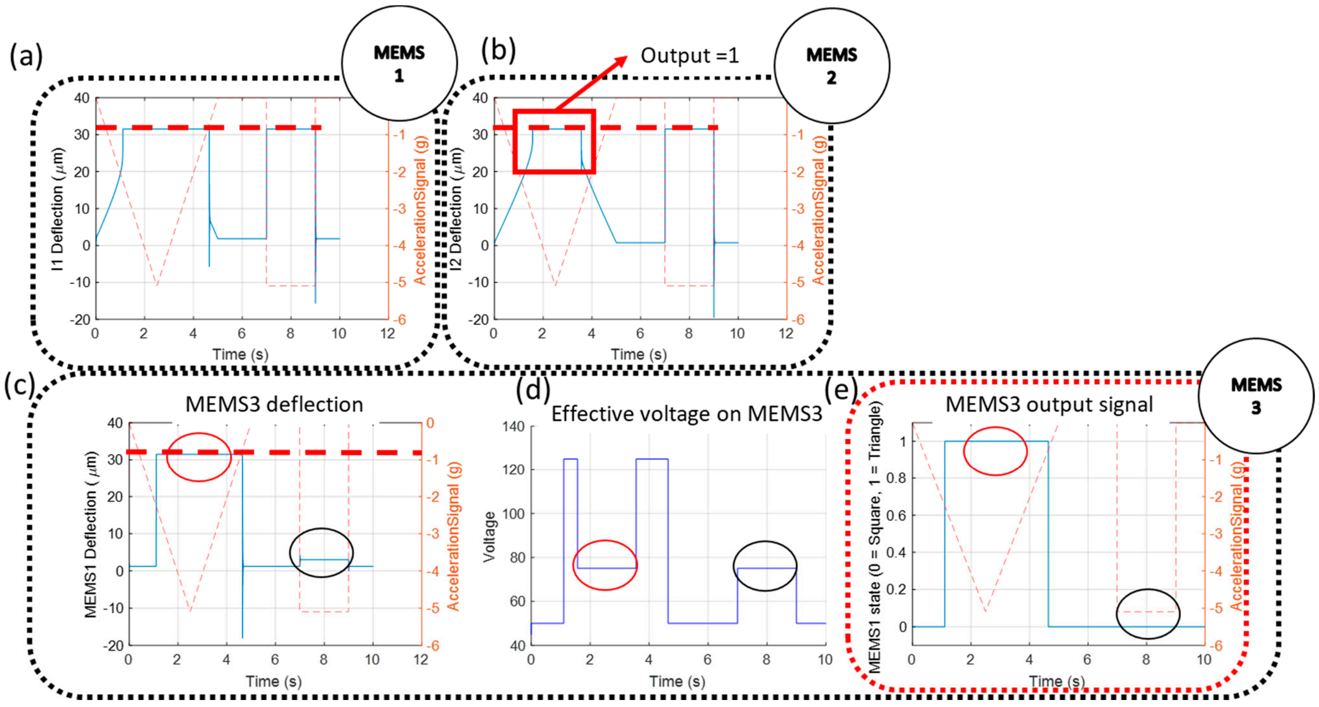

Figure 7. Classification test results showing the response of MEMS1 (**a**), MEMS2 (**b**) and MEMS3 (**c**). (**d**) The effective voltage acting on MEMS3 $V_3(t)$. (**e**) The state of MEMS3 when subject to a triangle or a square signal.

Alternatively, when a square signal is encountered, MEMS1 and MEMS2 experience a sudden and immediate change in amplitude, which results in them pulling-in (nearly) simultaneously. In this case, the voltage acting on MEMS3 is immediately equal to $w_{31}V_{b,1} + w_{3,2}V_{b,2} + V_{b,3}$ (noting that $w_{31} > 0$, $w_{32} < 0$). By design, this voltage is insufficient to pull-in MEMS3. Therefore, the output of MEMS3 remains low and square classification is performed. Interestingly, MEMS inertia is beneficial in this computing scheme as inertia prevents MEMS3 from pulling-in if MEMS1 is pulled in momentarily before MEMS2. Moreover, inertia allows this scheme to be performed to classify imperfect square signals, such as signals generated from a shaker, which tend to be trapezoidal, assuming the signal ramp is sufficiently steep since the MEMS devices will slightly lag the input signal.

The results from Figure 7 also clearly demonstrate the importance of hysteresis in a MEMS CTRNN as inputs of equal amplitudes may lead to significantly different behaviors depending on past information. (see the areas marked by the red circle and black dashed circle in Figure 7a–d, in which MEMS1 and MEMS2 are simultaneously pulled-in, yet MEMS3 can assume two different configurations).

It is worth mentioning that while the above simulated results provide great insight into the possibility of realizing a MEMS sensing and computing CTRNN and its working principle, a physical implementation of such a network using the commercial MEMS accelerometers network is not warranted. The commercial MEMS accelerometers are fabricated and packaged individually without any sort of mechanical coupling. Thus with this configuration it is hard to implement the negative weight (w_{32} in Table 1). The limitation of the commercial MEMS accelerometers has motivated the need to fabricate a customized design that utilizes mechanical coupling to achieve negative weight.

4. Waveform Classification Using a Customized MEMS Network

In this section, a novel design of a MEMS network to perform sensing and computing is presented. A schematic for the full network is presented in Figure 8 and a detailed schematic with dimensions of each MEMS in the network is in Figures A1 and A2 in the Appendices A and B. Figure 8 shows the network is made of two input MEMS sensors, each biased with a different voltage, to enable a different response to the applied acceleration. If the applied acceleration exceeds a threshold value, the MEMS will act as an ON switch that will activate a set of fingers on the computing MEMS (MEMS3). The bias voltages for each MEMS device and the number of fingers were manually tuned so that the MEMS3 will be pulled in (ON switch) when the applied acceleration is a triangular signal. Otherwise, it will be off. We note here that the output terminals have been designed at the proof mass contact point (represented by the triangular edges in Figure 8) to reduce the contact gap and minimize the risk of stiction. These contact tips additionally serve as stoppers to limit the distance between the electrodes in the moving and stationary assemblies upon the MEMS motion.

Like the commercial accelerometer network, a new model was developed for the customized network that accounts for electrostatic finger array coupling. Figure 9 shows the high accuracy of the network, using the tuned parameters, to distinguish a square signal from a triangular one. The next step was to fabricate the optimized network. Figure 10 shows the 2-mask micromachining process flow. In this approach, the devices are comprised of a 20–40 μm thick single-crystalline silicon device layer of a Silicon on Insulator (SOI) substrate with a thin coating of Ruthenium. The coating of Ruthenium is for mechanical robustness and low electrical contact resistance at output electrode contacts. The thickness of the substrate's device layer was chosen to be 30 μm with buried oxide (BOX) layer thickness of 2 μm. First, a 50 nm layer of aluminum oxide (Al2O3) was deposited on the SOI device layer via atomic layer deposition (ALD). The deposited thin film, patterned using the first lithography mask (Figure 10a), is to serve as a hard mask for the following device layer silicon etch. The device silicon skeletons were then carved into the device layer via deep reactive ion etching (DRIE) (Figure 10b). The second mask was used for backside lithography, which was followed by a long DRIE to remove the handle layer underneath the movable parts of the devices (Figure 10c). This is to avoid any potential stiction issues for the large proof masses. The buried oxide layer was wet etched from the backside by a 6-min dip in 49% hydrofluoric acid (HF) solution. The remaining Al2O3 is also removed during this step and the wire bonding pads are partially undercut due to the partial removal of the BOX underneath. Since the ruthenium deposition step is a maskless process, the undercut helps avoid shorts after the metal deposition. Finally, a thin layer of ruthenium (~400 nm thick) was sputtered on the fabricated devices (Figure 10d). The metal coating slightly covers the sidewalls contributing to a high-quality metal to metal electrical contact between the tip of the proof mass and the output electrode. Ruthenium was chosen due to its very high mechanical hardness and excellent wear resistance. A SEM view of a sample of the fabricated MEMS CTRNN is shown in Figure 11.

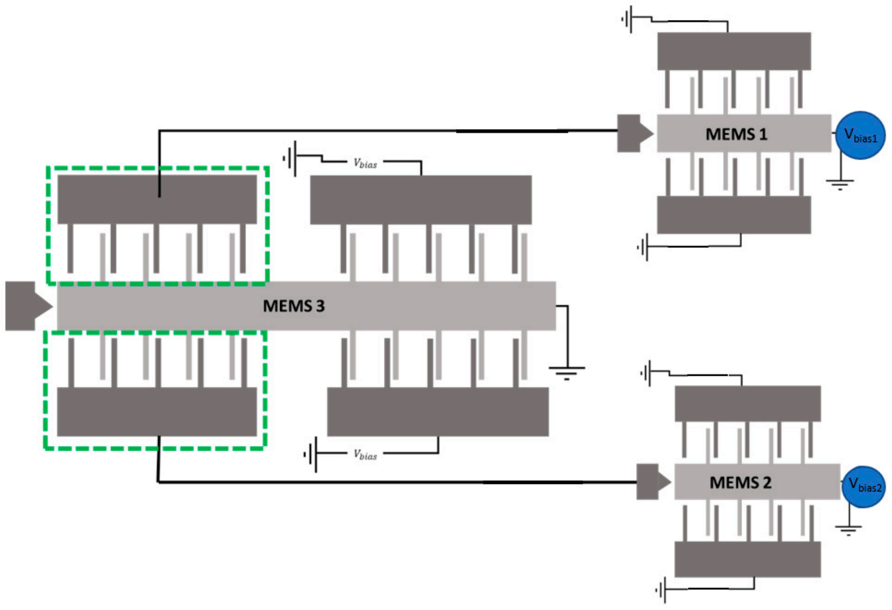

Figure 8. A schematic for the customized MEMS CTRNN to perform waveform classification. The schematic highlights the coupling through fingers between the three MEMS.

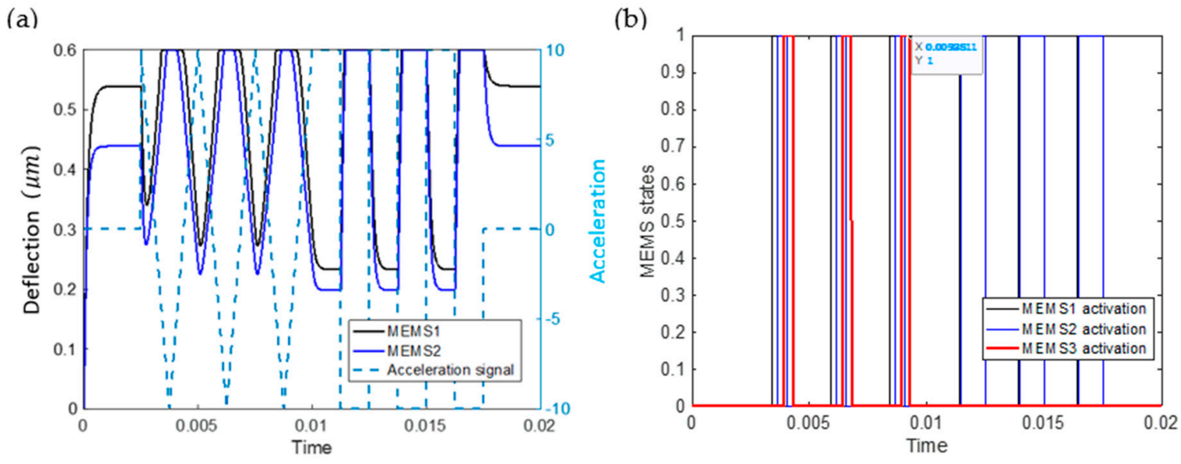

Figure 9. Simulation results for the customized MEMS CTRNN to classify square signal from triangle signal. (**a**) The deflection of the two input MEMS neuron and input acceleration signal. (**b**) the status of the three MEMS neurons, where the status of MEMS3 is considered the network output. The state of the MEMS device is considered to be 1 if pulled-in and 0 otherwise. By the end of each waveform cycle, MEMS3 is correctly on when the input signal is a triangle and is Off when the input signal is a square.

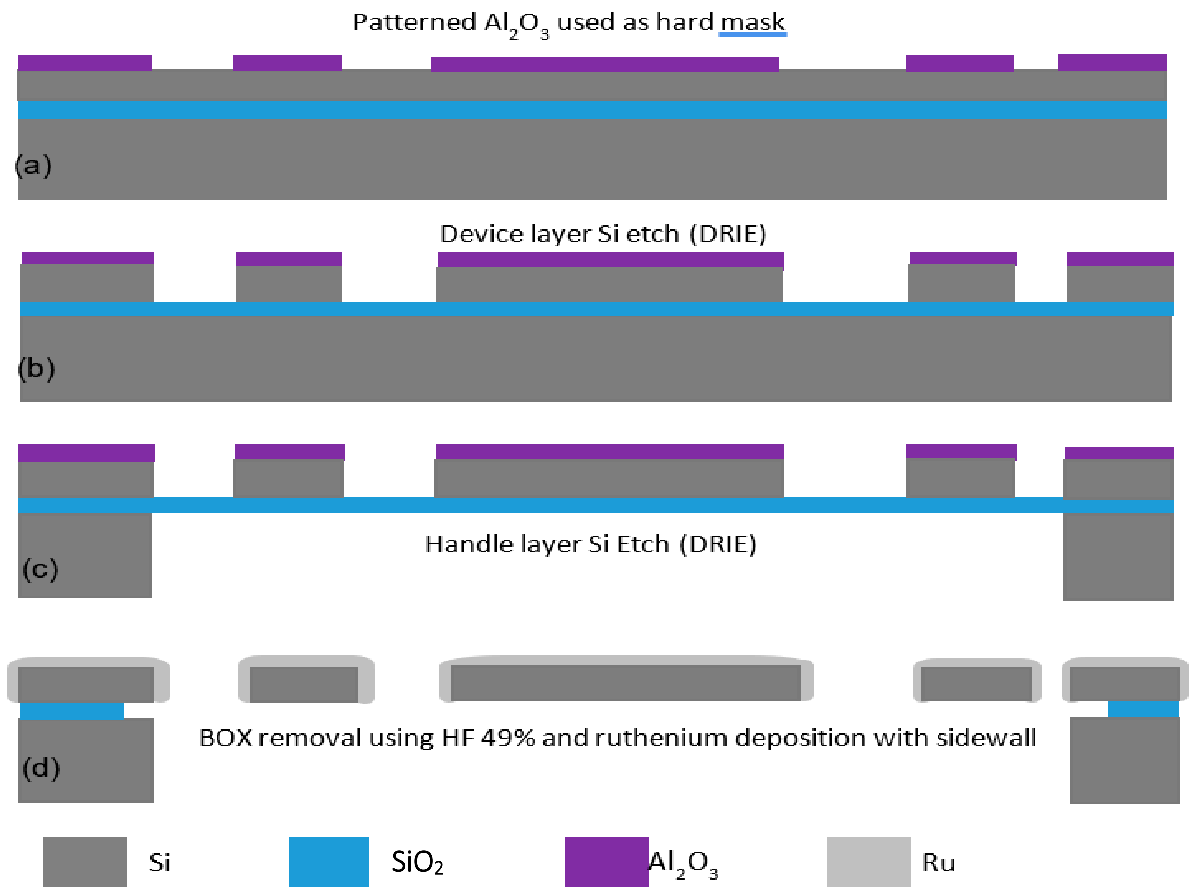
Patterned Al_2O_3 used as hard mask
(a)
Device layer Si etch (DRIE)
(b)
Handle layer Si Etch (DRIE)
(c)
BOX removal using HF 49% and ruthenium deposition with sidewall
(d)
Si
SiO2
Al_2O_3
Ru

Figure 10. Schematic side view showing the microfabrication process flow: (**a**) Patterned Al2O3 used as hard mask, (**b**) Device layer etch (DRIE) (**c**) Backside etch (DRIE) (**d**) Oxide etch in HF followed by ruthenium deposition with sidewall coverage.

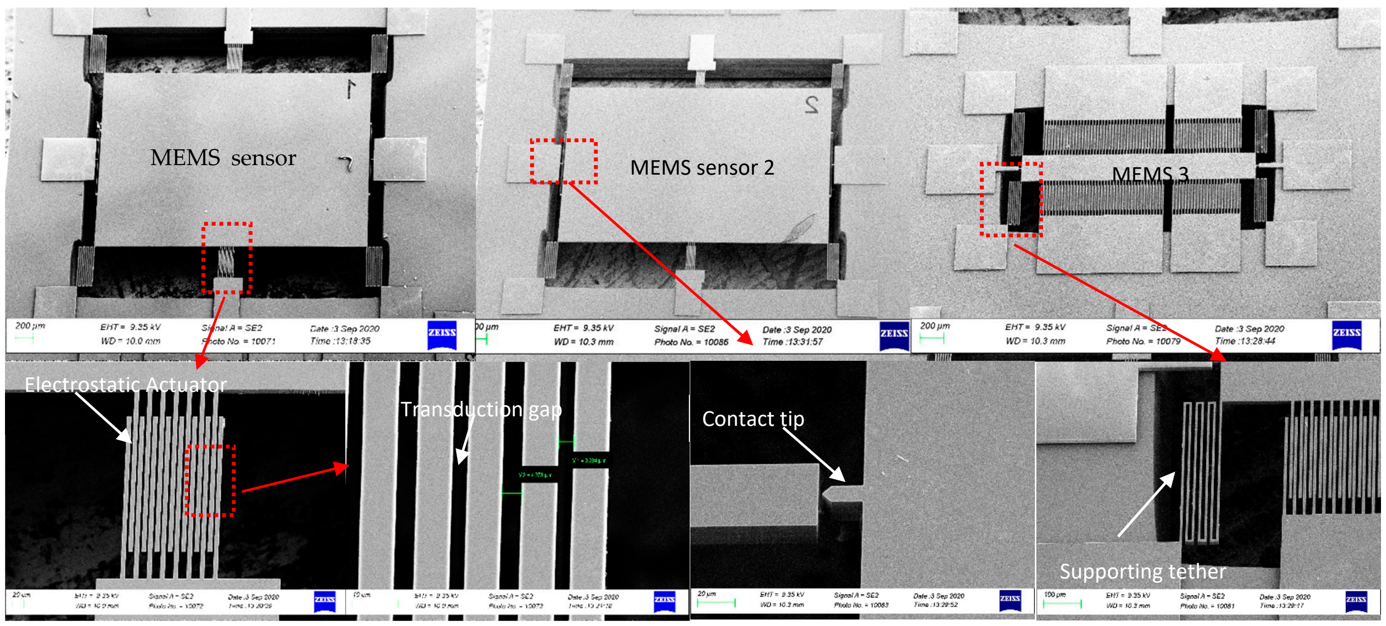

Figure 11. Scanning electron microscope (SEM) views of a fabricated MEMS CTRNN. This novel MEMS network can perform intelligent computing using only bias voltages.

A complete experimental set up shown in Figure 12 was designed to test the fabricated MEMS networks. In this setup, the MEMS device is fixed on a shaker. The MEMS response is measured as the difference between the microbeam and substrate base deflections. The shaker is controlled through a dedicated adaptive controller to produce the required signal as shown in Figure 12b. The vibrometer here is used to measure the motion of the entire MEMS structure. However, the actual proof mass deflection cannot be recorded using the vibrometer as the MEMS structure is in-plane. The MEMS response is instead probed electrically at pull-in. The actual MEMS deflection can be found by analyzing images from a digital holographic microscope using edge detection (Figure 13). However, while the computing MEMS3 seems to work as expected as shown in Figure 13, there were issues with MEMS1 & MEMS2 during operation. Specifically, the manual tuning for the design parameters for MEMS1 & MEMS2 devices resulted in having a large proof mass along with very low stiffness tethers for the devices. Thus, they were very vulnerable to shock and vibration, even those happening during handling and mounting the chips. This resulted in multiple supporting tethers breaking. A sample hysteresis plot of MEMS3 is shown in Figure 13c, showing pull-in near 22 V and pull-out near 16 V, providing a wide regime of hysteresis in-between. Here, the MEMS circuit for measuring the output voltage includes a MEMS device, a DC output voltage supply of 5 V and a 200 kΩ resistor. Most voltage drop is across the MEMS device when the device is not pulled in. Once pull-in occurs, the MEMS device acts as an element with low resistance (around 1 kΩ), thus most voltage drop is reported across the external resistor, resulting in the voltage drop across the MEMS device reported in Figure 13c. The reported voltage of 0.3 V at pull-in is a result of the reading being reported using a 1 MΩ input impedance oscilloscope for measurement. Figure correction is attainable by shifting the entire figure by ≈ 0.3 V.

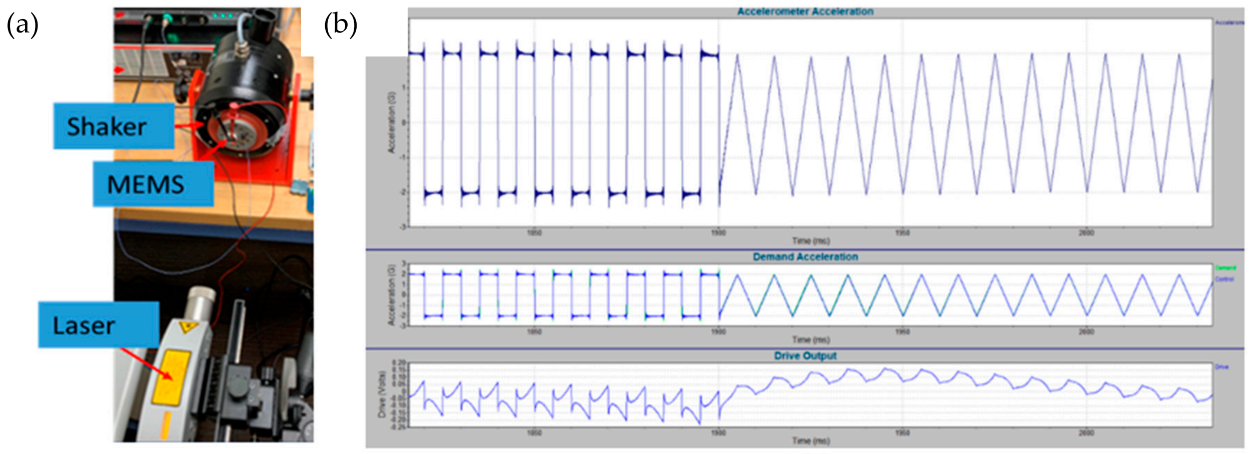

Figure 12. (**a**) The experimental set up to test the MEMS CTRNN, (**b**) samples of the triangle and square acceleration profiles generated by the mechanical shaker.

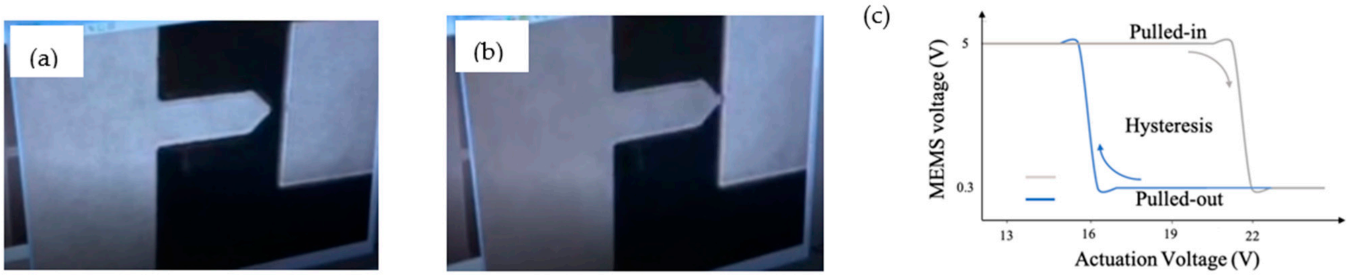

Figure 13. Pull out (**a**) and pull in (**b**) images for MEMS 3. (**c**) A sample hysteresis plot.

The reliability of the MEMS devices based on the number of contacts prior to failure has been characterized. In order to perform the reliability test, the ohmic resistivity between metallic tip of the proof mass and the output electrode (coated with ruthenium thin film) has been monitored for a long-time operation of the device. In this manner, electrostatic actuator of a sample MEMS device was fed by pulse signal with predetermined frequency of 50 Hz and an amplitude that assures pull-in, while the output electrode was biased with a DC voltage of 5 V through a very large resistivity of 100 kΩ. Similar to electrical configuration of the device during the acceleration measurement operation, proof mass was electrically grounded. This operation simulates operating the MEMS device as a switch, which is continuously turned on and off. Contact between the tip of the proof mass and output electrode closes the electrical circuit and results in a DC current through the contact point. In this manner, the ohmic resistivity of metal-metal contact can be simply measured using ohm's law. An ohmic resistivity of 1.2 kΩ has been measured for the very early cycles of operation. Damage of the thin film metal deposited on the silicon skeleton increases the ohmic resistivity of the contact suddenly at around the 17.5 million cycle mark, which occurred after 4 days of continuous pull-in and pull-off operations. An ohmic resistivity of around 1 MΩ has been measured after the damage of the metal film. Figure 14 shows SEM zoomed-in view of both sides of the contact point after 17.5 million cycles of operation.

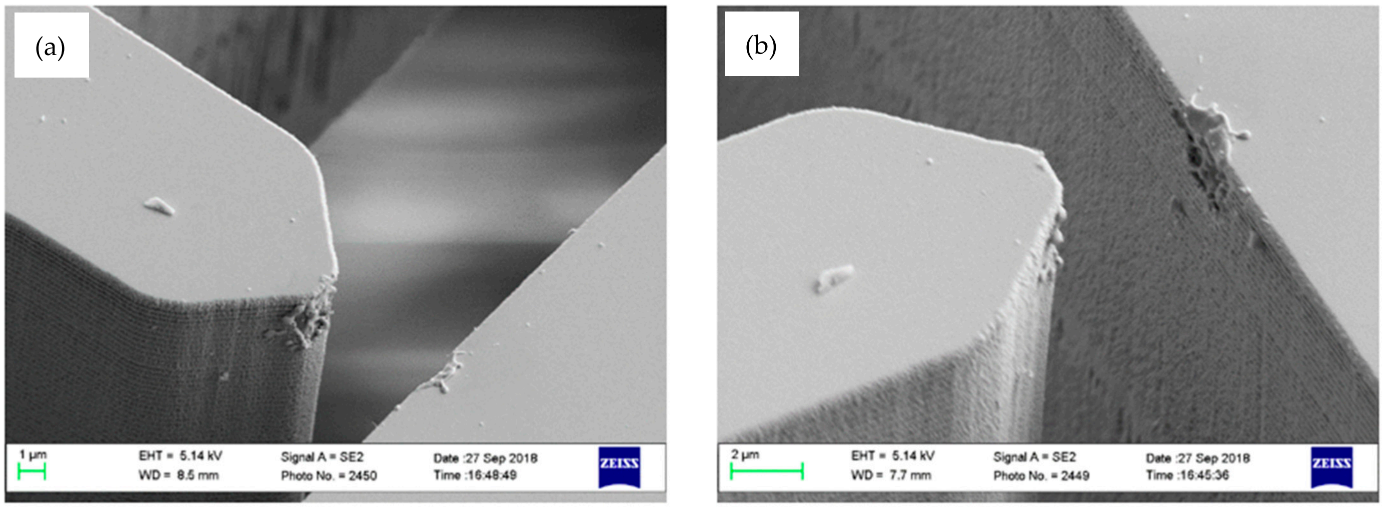

Figure 14. (**a**) SEM view of the contact tip after the long-term operation causing damage to the metal film. (**b**) SEM view of the damaged metal film on output electrode.

5. Conclusions

The concept of performing sensing and computing using MEMS devices has great potential for advancing computing in many applications such as wearable devices, however, poses new challenging problems that require new ways of thinking to solve. For example, this novel concept requires optimizing the MEMS design parameters to afford simultaneous sensing and computing. In this paper, however, we adopted a manual training technique that solves intuitively the computing behavior, while ignoring the sensing limitation. While our simulation shows a great response, the real implementation and fabrication of this MEMS CTRNN network revealed that its sensing mechanical parameters (i.e., mass and stiffness) while accommodating the required computing aspect, were too sensitive to shocks resulting in their mechanical failure. Our ongoing approach involves using common machine learning techniques such as genetic algorithm and Back Propagation Through Time (BPTT) among other methods to optimize the MEMS parameters to satisfy both the computing and sensing requirements of the MEMS CTRNN network. We also plan to investigate the capability of MEMS CTRNN in more complex classification applications with relatively long-term time-series patterns such as those that occur in motion sensor data involving human activities.

Author Contributions: Conceptualization, F.A. and S.P.; methodology, F.A., M.H.H., A.A. and H.N.; software, M.H.H. and M.E.-U.-D.; validation, M.H.H.; formal analysis, M.H.H.; investigation, F.A.; resources, S.P.; data curation, M.H.H.; writing—original draft preparation, F.A. and M.H.H.; writing—review and editing, S.P.; visualization, M.H.H.; supervision, F.A., S.P. and R.J.; project administration, F.A.; funding acquisition, F.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation, grant number (#1935598, #1935641).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Schematics and dimensions of the computing MEMS, MEMS3, fabricated in Section 4.

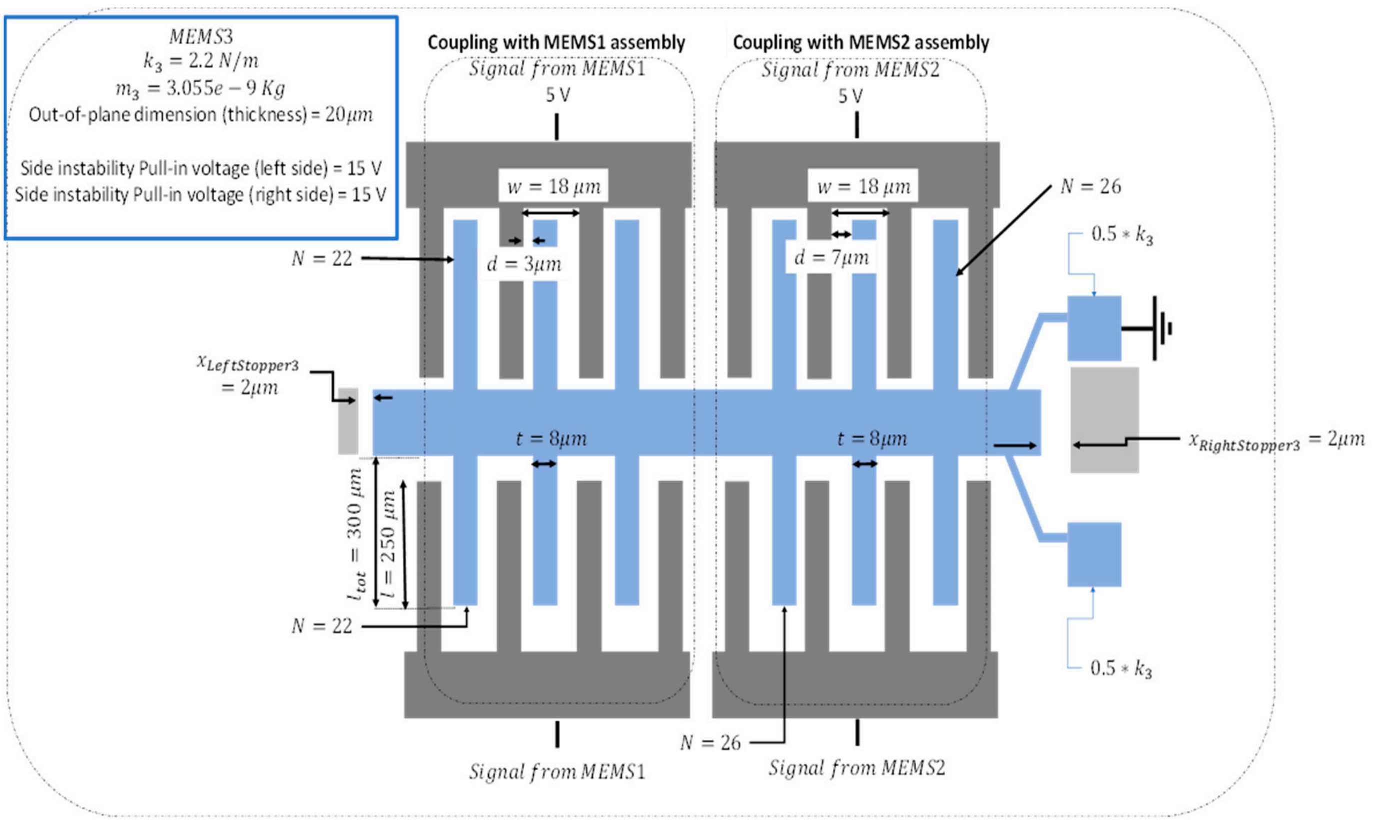

Figure A1. Design of the computing MMES device (labeled MEMS3 in Section 4), including information about the total stiffness of the MEMS device and the pull-in voltage.

Appendix B

Schematics and dimensions of the input-layer MEMS devices: MEMS1 and MEMS2, fabricated in Section 4. The two MEMS devices are shown below, respectively.

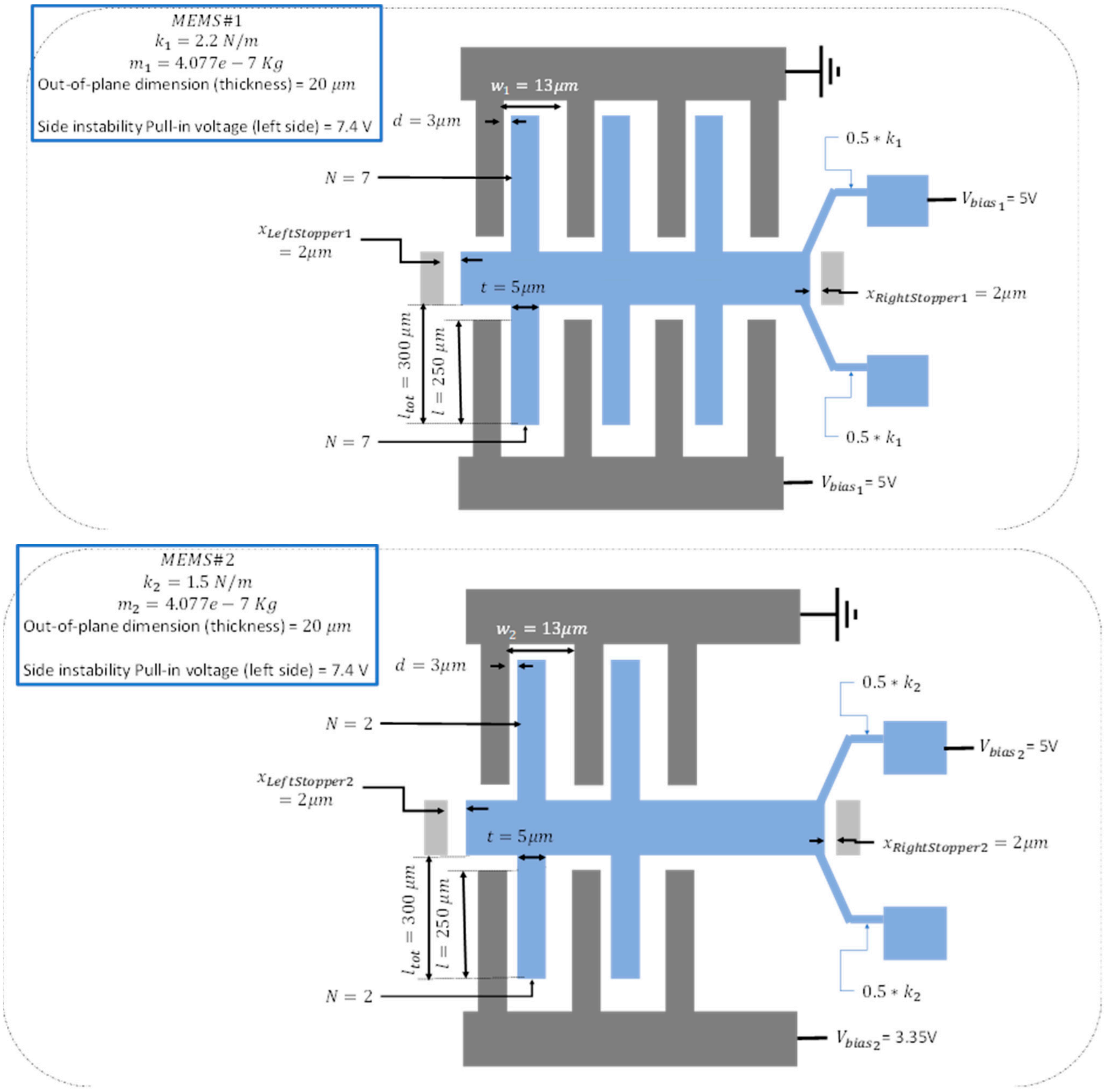

Figure A2. Design of the input layer MMES device (labeled MEMS1 and MEMS2 in Section 4), including information about the total stiffness of the MEMS devices and the pull-in voltages. Top figure: MEMS1, bottom figure: MEMS2.

References

1. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **1990**, *78*, 1629–1636. [CrossRef]
2. Schuman, C.D.; Potok, T.E.; Patton, R.M.; Birdwell, J.D.; Dean, M.E.; Rose, G.S.; Plank, J.S. A survey of neuromorphic compu-ting and neural networks in hardware. *arXiv* **2017**, arXiv:1705.06963.
3. Qiao, N.; Mostafa, H.; Corradi, F.; Osswald, M.; Stefanini, F.; Sumislawska, D.; Indiveri, G. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front. Neurosci.* **2015**, *9*, 141. [CrossRef] [PubMed]
4. Delbrück, T.; Mead, C.A. Adaptive photoreceptor with wide dynamic range. In Proceedings of the IEEE International Symposium on Circuits and Systems—ISCAS '94, London, UK, 30 May–2 June 1994; Volume 4, pp. 339–342.
5. Lyon, R.; Mead, C. An analog electronic cochlea. *IEEE Trans. Acoust. Speech Signal. Process.* **1988**, *36*, 1119–1134. [CrossRef]
6. Ng, K.T.; Boussaid, F.; Bermak, A. A CMOS Single-Chip Gas Recognition Circuit for Metal Oxide Gas Sensor Arrays. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2011**, *58*, 1569–1580. [CrossRef]

7. Dalgaty, T.; Vianello, E.; De Salvo, B.; Casas, J. Insect-inspired neuromorphic computing. *Curr. Opin. Insect Sci.* **2018**, *30*, 59–66. [CrossRef] [PubMed]
8. Kumar, P. Mohanty, Autoassociative memory and pattern recognition in micromechanical oscillator network. *Sci. Rep.* **2017**, *7*, 1–9.
9. Hoppensteadt, F.; Izhikevich, E. Synchronization of MEMS resonators and mechanical neurocomputing. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2001**, *48*, 133–138. [CrossRef]
10. Dion, G.; Mejaouri, S.; Sylvestre, J. Reservoir computing with a single delay-coupled non-linear mechanical oscillator. *J. Appl. Phys.* **2018**, *124*, 152132. [CrossRef]
11. Hasan, M.H.; Al-Ramini, A.; Abdel-Rahman, E.; Jafari, R.; Alsaleem, F. Colocalized Sensing and Intelligent Computing in Micro-Sensors. *Sensors* **2020**, *20*, 6346. [CrossRef] [PubMed]
12. Alsaleem, F.M.; Hasan, M.H.H.; Tesfay, M.K. A MEMS Nonlinear Dynamic Approach for Neural Computing. *J. Microelectromechanical Syst.* **2018**, *27*, 780–789. [CrossRef]
13. Rafaie, M.; Hasan, M.H.; Alsaleem, F.M. Neuromorphic MEMS sensor network. *Appl. Phys. Lett.* **2019**, *114*, 163501. [CrossRef]
14. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv* **2015**, arXiv:1506.00019.
15. Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015; Volume 25.
16. Beer, R.D. The Dynamics of Active Categorical Perception in an Evolved Model Agent. *Adapt. Behav.* **2003**, *11*, 209–243. [CrossRef]
17. Bailador, G.; Roggen, D.; Troster, G.; Trivino, G. Real time gesture recognition using Continuous Time Recurrent Neural Networks. In Proceedings of the ICST 2nd International Conference on Body Area Networks, Florence, Italy, 11–13 June 2017; p. 15.
18. Shin, S.; Sung, W. Dynamic Hand Gesture Recognition for Wearable Devices with Low Complexity Recurrent Neural Networks. In Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 22–25 May 2016.
19. Kalantarian, H.; Sideris, C.; Mortazavi, B.; Alshurafa, N.; Sarrafzadeh, M. Dynamic Computation Offloading for Low-Power Wearable Health Monitoring Systems. *IEEE Trans. Biomed. Eng.* **2016**, *64*, 621–628. [CrossRef] [PubMed]
20. Gorthi, S.; Mohanty, A.; Chatterjee, A. Cantilever beam electrostatic MEMS actuators beyond pull-in. *J. Micromech. Microeng.* **2006**, *16*, 1800–1810. [CrossRef]
21. Ramini, A.; Bellaredj, M.L.F.; Al Hafiz, A.; Younis, M.I. Experimental investigation of snap-through motion of in-plane MEMS shallow arches under electrostatic excitation. *J. Micromech. Microeng.* **2016**, *26*, 015012. [CrossRef]
22. Abbasalipour, A.; Nikfarjam, H.; Pourkamali, S. An 8-Bit Digitally Operated Micromachined Accelerometer. *J. Microelectromechanical Syst.* **2020**, *29*, 1132–1136. [CrossRef]
23. Paquot, Y.; Duport, F.; Smerieri, A.; Dambre, J.; Schrauwen, B.; Haelterman, M.; Massar, S. Optoelectronic Reservoir Computing. *Sci. Rep.* **2012**, *2*, 1–6. [CrossRef] [PubMed]
24. Vandoorne, K.; Dierckx, W.; Schrauwen, B.; Verstraeten, D.; Baets, R.; Bienstman, P.; Van Campenhout, J. Toward optical signal processing using Photonic Reservoir Computing. *Opt. Express* **2008**, *16*, 11182–11192. [CrossRef] [PubMed]