A 65-nm 8-to-3-b 1.0–0.36-V 9.1–1.1-TOPS/W Hybrid-Digital-Mixed-Signal Computing Platform for Accelerating Swarm Robotics

Ningyuan Cao, *Member, IEEE*, Muya Chang, and Arijit Raychowdhury, *Senior Member, IEEE*

***Abstract*— Low-power edge-intelligence is leading to spectacular advances in smart sensors, actuators, and human–machine interfaces. In particular, energy efficiency is driving key advances in robotics, where low-power computation is augmented with smart control and mechanical systems to enable small-sized and intelligent drones, unmanned aerial vehicles (UAVs), micro-sized cars, and so on with applications in surveillance, disaster relief, and reconnaissance. Furthermore, for a variety of tasks, swarms of robots are often used as opposed to the individual robots. This article presents an energy-efficient computing platform that can enable a sample class of algorithms for swarm robotics. We demonstrate that both physical-model-based algorithms as well as learning-based algorithms can be supported on the same computing platform. We also demonstrate that with changing swarm sizes, the number of bits required to compute also scales. We take advantage of this observation to propose a hybrid-digital-mixed-signal computing platform, whose energy efficiency scales with the resolution of the data path and hence the swarm size. Measurements on a 65-nm CMOS test-chip demonstrate a peak energy efficiency of 9.1 TOPS/W at a 3-b resolution, and it scales down to 1.1 TOPS/W at an 8-b resolution.**

***Index Terms*— Machine learning, mixed signal, robotics, swarm intelligence.**

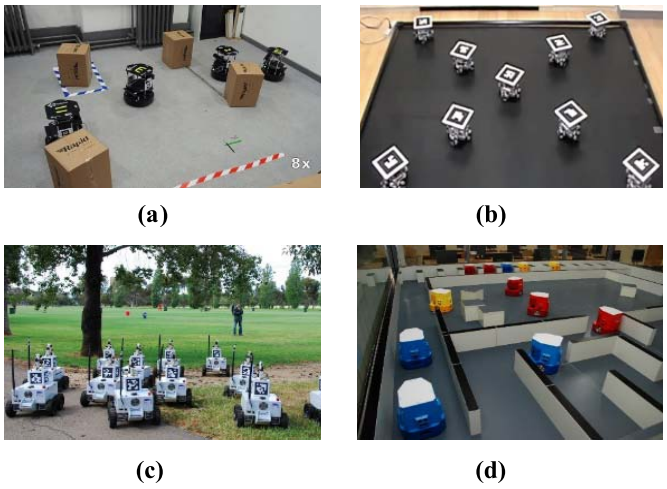


Fig. 1. Swarm algorithms that can successfully accomplish (a) collaborative path planning, (b) pattern formation, (c) multi-agent patrolling, and (d) multi-agent predator-prey.

I. Introduction

INSPIRED by the collective intelligence of biological systems, swarm robotics is an emerging area where multiple robots work together to enable complex swarm behavior. The problem-solving capability enabled through simple interactions among the agents enables novel applications [1]–[5]. In swarm robotics, multiple small and distributed robots coordinate and gather data to enable intelligent decision-making as a group (shown in Fig. 1). These have been used in applications, such as exploration, reconnaissance, and disaster relief [6]. The fact that distributed and swarm robotics are resilient to component-level failures further motivates the use of swarms. In swarm robotics, multiple robots often coordinate in real time to solve diverse problems, such as pattern formation, cooperative reinforcement learning (RL), and path planning. Some of these algorithms use learning-based methods and have gained increasing importance with the success of deep neural networks and neuromorphic computing. Although certain swarm algorithms rely on real-time learning (e.g., cooperative RL) representing a model-free approach, many powerful algorithms that have been developed over the past two decades (e.g., pattern formation) rely on a mathematical structure and represent a more traditional physical-model-based approach. The next generation of swarm hardware needs to support both of these approaches; and hence, it is important to identify the common computational kernels that need to be supported in hardware. However, hardware designs that can support computation in swarms are computationally challenging; especially from an energy-perspective. This is discussed in [7]: the main processor in a coin-size swarm robot consumes 4× energy than a micro-controller, and this energy is compared (more than 80%) with motors and camera-based sensors [7]. As swarm robots are expected to enable the so-called "intelligence" in reduced form factors, energy-efficient hardware design continues to be an active area of research. In this article, we identify the commonalities and shared compute primitives across a variety of model-based and model-freeswarm algorithms and present a unified, fully programmable, energy-efficient, and scalable platform capable of real-time swarm intelligence. Although we demonstrate how

Manuscript received May 2, 2019; revised July 15, 2019; accepted August 9, 2019. Date of publication August 29, 2019; date of current version December 27, 2019. This article was approved by Guest Editor Mingoo Seok. This work was supported by the Semiconductor Research Corporation under Grant JUMP CBRIC 2777.006. *(Corresponding author: Ningyuan Cao.)*

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: nycao@gatech.edu; mchang87@gatech.edu; arijit.raychowdhury@ece.gatech.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2019.2935533

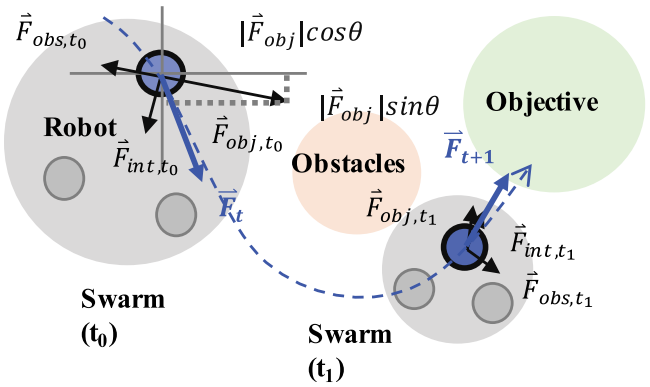


Fig. 2. Schematic map showing APF-based path planning and formation.

to support some sample algorithms here, the design principles are scalable and can be applied to larger swarms, enabling more advanced algorithms.

To enable a unified energy-efficient computing platform for swarm robotics, we demonstrate a hybrid-mixed-signal and digital design. In [8], we demonstrated a purely time-based mixed-signal neural network for RL on edge devices. However, a purely mixed-signal solution shows superior energy efficiency for low bits of resolution. As the number of bits on the data path increases, mixed-signal solutions tend to be less efficient than purely digital counterparts. In swarm robotics, the size of the swarm determines the size of vectors that need to be computed, and hence, the bit-width required for high accuracy also scales with the swarm size. Hence, the mixed-signal solutions are efficient for small swarms, while the digital solutions tend to outperform in larger swarms. To enable such scalability, we demonstrate a hybrid-digital-mixed-signal (HDMS) solution where a time-domain mixed-signal (TDMS) kernel computes on 3–5-b data. A digital wrapper around the mixed-signal kernel further scales the computing platform to 6–8 b. This allows high energy efficiency for low precision along with the excellent energy scalability of digital computing for larger bit-widths.

The test chip has been fabricated in a 65-nm CMOS process. We demonstrate 9.1-TOPS/W peak energy efficiency at 3-b resolution. The energy efficiency decreases to 1.1 TOPS/W for 8-b resolution. The test chip interfaces with a Raspberry Pi platform consisting of integrated sensors (inertial sensors and ultrasonic distance sensors) and long-range (LoRa) radios for decentralized, peer-to-peer communication among mobile robotic vehicles in a swarm. The rest of this article is divided as follows. Section II provides an overview of the swarm algorithms. Sections III and IV describe the scalability of the computing platform with the swarm size and the HDMS design. The system overview is described in Section V, and the measurement results are shown in Section VI. Finally, an outlook of potential future works is discussed in Section VII, and the conclusion is drawn in Section VIII.

II. Overview of Swarm Algorithms

Swarm algorithms can be broadly classified into two categories: the ones based on the physical and mathematical models and the ones based on learning. In Sections II-A and II-B, we provide an overview of the types of algorithms that are supported by the common unified platform.

A. Algorithms Based on Physical Models

Over the past decades, there has been a significant development in swarm control algorithms inspired by the physical and mathematical models. Among these mathematical models, artificial potential field (APF) is a popular and practically useful computational approach. In APF, we assume that the robots and the objects (goal, obstacles, and teammates) are similar to "electrical charge" that produces artificial attractive and repulsive potential fields whose potential functions are to be leveraged by the system designer for optimal robotic control and system performance. By aggregating the potential fields (i.e., forces), the motion vector can be obtained at each evaluation step. In general, the APF algorithm has the following format [9]–[13]:

$$m_i \frac{d\vec{v}_i}{dt} = \vec{F}_{\text{pro},i} + \vec{F}_{\text{int},i} + \vec{F}_{\text{esp},i} + \vec{F}_{\text{est},i}. \tag{1}$$

This is based on Newton's second law to describe the ith robot's velocity v_i change determined by propulsion $\vec{F_{\text{pro},i}}$, interaction $\vec{F_{\text{int},i}}$, objective escape $\vec{F_{\text{esp},i}}$, and stochastic forces $\vec{F_{\text{est},i}}$ and mass m_i. By properly choosing the potential function that generates each term, we are able to design a cooperative control algorithm that can implement applications, such as collaborative path planning and co-coordinated formation. A typical example is shown in Fig. 2.

For example, for path-planning applications as shown in Fig. 1(a), the positional information of objectives and obstacles is required in determining the motion vectors. In this design, we consider the standard parabolic potential U_{obj} for the object and an exponential potential barrier for the obstacles U_{obs} from [9]

$$U_{\text{obj}}(\vec{r}) = k_{\text{obj}}\text{dis}(\vec{r}, \vec{r}_{\text{obj}})^2 \tag{2}$$

$$U_{\text{obs}}(\vec{r}) = k_{\text{obs}}\text{dis}(\vec{r}, \vec{r}_{\text{obs}})^{-1} \tag{3}$$

where $\vec{r}_{\text{obj}}$ and $\vec{r}_{\text{obs}}$ are the positions of the objective and obstacles, respectively. The force vectors created by these potential functions in the 2-D plane are of the form

$$\vec{F_{\text{pro}}} = -k_f \nabla_i \left(U_{\text{obj}}(\vec{r}) + \sum_{m=1}^{M} U_{\text{obs}}(\vec{r}) \right) \tag{4}$$

$$F_{\text{pro},x} = \alpha|\vec{r} - \vec{r}_{\text{obj}}|\cos\theta_{\text{obj}} + \sum_{m=1}^{M} \beta_m |\vec{r} - \vec{r}_{\text{obs},m}|^{-2} cos\theta_{\text{obs},m} \tag{5}$$

$$F_{\text{pro},y} = \alpha|\vec{r} - \vec{r}_{\text{obj}}|\sin\theta_{\text{obj}} + \sum_{m=1}^{M} \beta_m |\vec{r} - \vec{r}_{\text{obs},m}|^{-2} sin\theta_{\text{obs},m}. \tag{6}$$

For formation applications as shown in Fig. 1(b), the potential function uses a logarithm-cosine-hyperbolic function

$$U_{\text{int}}(\vec{r}) = \beta ln(\cos h|\vec{r} - \vec{R}|) \tag{7}$$

where $\vec{r}$ is the interaction vector, while $\vec{R}$ is the target vector. Enabling each interaction with a dedicated target vector allows

fine-tuning of the shape of the formation. The resulting force equations in the 2-D plane can be expressed as

$$\vec{F_{\text{int}}} = -\nabla_i(U_{\text{int}}(\vec{r}_i, \vec{r}_i) \tag{8}$$

$$F_{int,x} = \sum_{m=1}^{M} \alpha_i[\tanh(|\vec{r}_j - \vec{R}_j|)\cos\theta_j] \tag{9}$$

$$F_{int,y} = \sum_{m=1}^{M} \alpha_i[\tanh(|\vec{r}_j - \vec{R}_j|)\sin\theta_j]. \tag{10}$$

To solve swarm problems, we need to compute (1) with the correct parametric representations of the functions and parameters as obtained from (4) to (6) and (8) to (10). These parameters are obtained from system-level simulations before deployment.

B. Learning-Based Algorithms

With the rapid development of hardware systems to support machine learning and artificial intelligence [8], [14]–[16], advanced learning-based techniques are becoming popular for applications, such as multi-robot predator-prey and multi-agent patrolling, as shown in Fig. 1(c) and (d); learning-based algorithms have now become competitive in a variety of problems where the pre-defined models may not exist or may be incomplete. The motivation for the learning-based approach is to allow each robot to learn continuously without human intervention and establish a control model with real-world knowledge. Among all the approaches, an RL-based cooperative Q-learning [2], [17]–[19] algorithm has shown great promise.

Single-agent Q-learning [20], [21] is based on the iterative update of the Q value, as a robot navigates through a series of (state, action, and reward) tuples. This iterative scheme is derived from the Bellman equation [22] for optimal control. The iterative algorithm can be summarized as

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha(R_t + \gamma \max Q_t(S_{t+1}, A_t) - Q_t(S_t, A_t)) \tag{11}$$

$$R_t = f(S_t) \tag{12}$$

where γ and α are the discount factor and the learning rate to aggregate the distant rewards and update Q-tables, respectively. By taking a series of actions A (moving forward and backward) in the state space S (positions and obstacle vectors), the robot calculates the reward for each action and updates the Q-table, thus creating a robust functional mapping from the state space to the action space. The reward is based on a single robot's current state. A hardware implementation of Q-learning for autonomous navigation has been presented in [8] and [23], and interested readers are pointed to the references for more details.

In cooperative Q learning, global, instead of local, states and rewards are utilized to facilitate multi-agent collaboration. As opposed to the baseline Q-learning where a single agent's local state is used, in a swarm, the local states are broadcasted to all the teammates. This forms a global state, which incorporates the knowledge of all teammates. The Q value of the swarm is now evaluated as

$$Q_{t+1}(S_{t,\text{global}}, A_t) = Q_t(S_{t,\text{global}}, A_t) + \alpha(R_{t,\text{global}} + \gamma \max Q_t(S_{t+1,\text{global}}, A_t) - Q_t(S_{t+1,\text{global}}, A_t)) \tag{13}$$

$$S_{t,\text{global}} = [S_{t,1}, S_{t,2} \cdots S_{t,N}]. \tag{14}$$

As described in [23], each robot will now take an action based on the best Q value of the current global state. A global reward is evaluated based on the team's performance, for example, whether one of the targets has been reached by one of the team members. It is worth noting that we incorporate the task completion time as a reward function, as it improves the swarm's performance and facilitates convergence by encouraging all robots to take continuous actions

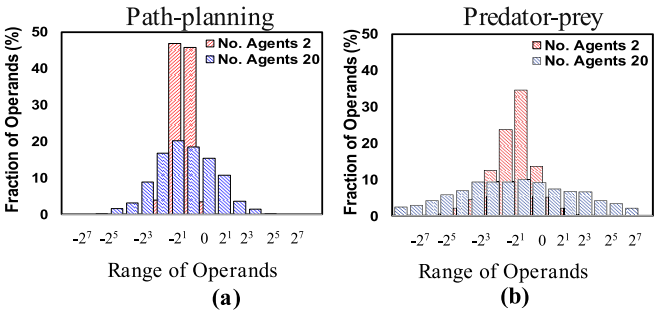
$$R_t = g(S_{t,\text{global}}, t). \tag{15}$$

When the environment is complex and the swarm size is large, the global state can also be significantly large. It is difficult to store all the Q values in a table, especially in memory-constrained design. Therefore, the Q value is typically approximated as a neural network output. The states ($S_{t,\text{global}}$) (sensor values and current positions) act as inputs to the neural network. Then, every neural network propagates the states through an embedded neural network and produces Q values of each action. A hard-max function at the end of the neural network establishes the best action to be taken. We use ϵ-greedy as means to perform exploration. The details of cooperative Q-learning for multi-robot action are a rich and evolving area of algorithmic research. For more details on cooperative Q-learning, interested readers are directed to [2].

III. Common Computing Platform

As we mentioned earlier, future computing platforms that can support swarm algorithms need to support both mathematical algorithms as well as learning-based algorithms. Interestingly, we observe that both these two algorithms have a basic mathematical structure. As computational problems, they both feature as follows.

1) *Linear Processing Unit (LPU):* Both types of algorithms work on vectors and matrices, and hence, linear processing is a critical component of computation. In APF-based algorithms, linear operations are performed on trigonometric transformations of motion vectors [see (1)]. In neural networks, the linear units allow the synaptic weights to be summed up at the input of a neuron. Fundamentally, the computational platform needs to support multiplications and additions [through multiply-and-accumulate, multiplication and accumulation (MAC) units].
2) *Non-Linear Processing Unit:* Apart from linear vector processing, both algorithms require non-linear transformations. In APF algorithms, these transformations are mostly trigonometric [see (4)–(6) and (8)–(10)], whereas in neural networks, these transformations are the activation functions (sigmoid and rectified linear unit). In APF, the linear processing is done on non-linearly transformed



Swarm size	Algorithms			
	Path-planning	Formation	Predator-prey	Cooperative exploration
2	3	3	5	4
5	4	4	7	4
10	5	5	7	5
15	5	6	8	5
20	6	7	8	6

(c)

Fig. 3. Algorithmic simulations demonstrate how the required bit precision scales with the swarm size for two template problems (a) collaborative path planning and (b) multi-agent predator-prey. (c) Number of bits required to accurately compute different template algorithms for varying swarm sizes.

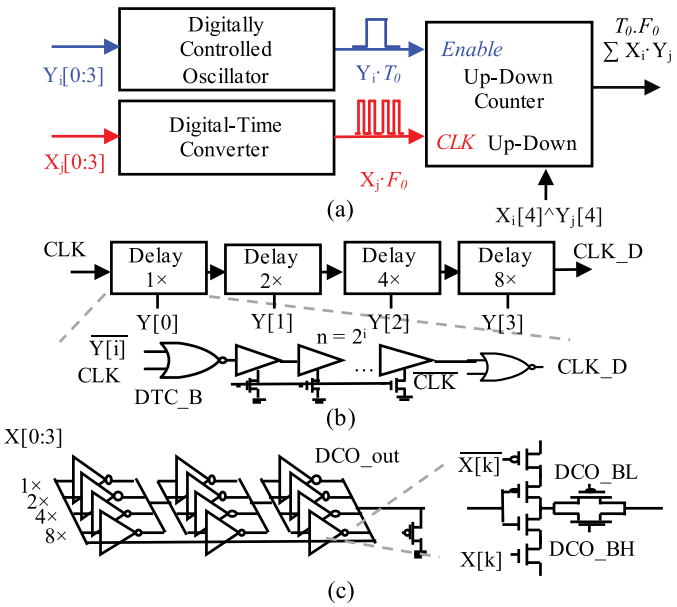


Fig. 4. Circuit schematic illustrating (a) TDMS MAC circuit, (b) DPC, and (c) DCO.

motion and position vectors; hence, we perform non-linear processing followed by linear processing. On the other hand, in neural networks, we perform MACs first, followed by non-linear activation functions.

Since linear/non-linear operations are the major workloads in robotic algorithms, this unified compute platform is designed to provide a unified solution to accelerate both types of computation. With a dedicated non-linear processing unit and an LPU, we achieve high energy efficiency as will be described in Section VII.

The order of linear and non-linear processing is different in the two algorithms, but in a memory-centric system, this amounts to simply changing the order of instructions to support both the classes of algorithms. This shows that a unified computing platform comprising of: 1) an LPU; 2) a programmable non-linear processing unit; 3) a data cache; and 4) an instruction cache will be able to support both the model-based and learning-based algorithms. In the proposed ASIC, we demonstrate support for both types of algorithms with a non-linear processing unit, which is composed of a lookup table (LUT)-based piecewise approximation of the non-linear function. The LPU is composed of an MAC array and data cache and instruction cache with standard 6T SRAM cells.

IV. Scalability With Swarm Size

The number of agents in a swarm, also called the swarm size, is a major design parameter for providing optimal performance and robustness at a minimal system cost. For example, in disaster relief, to ensure the largest area coverage and the fastest convergence rate, a relatively large number of agents are often preferred. However, for indoor exploration, a small group of robots is likely to be sufficient, given the reduced problem complexity and increased environmental clutter. As a consequence, future computing platforms that can support multiple swarm algorithms also need to be able to support multiple swarm sizes. To prevent over-design, the computing platforms need to perform at optimal energy efficiency for a large scale of swarm sizes.

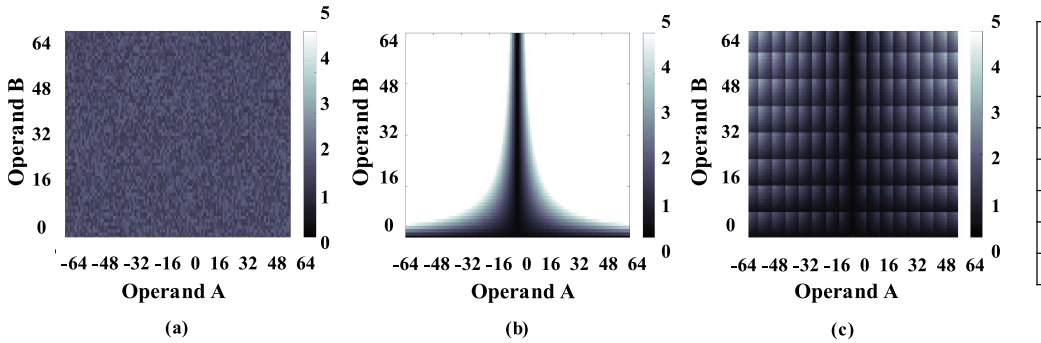
To better understand the computation requirement for varying swarm sizes, we analyze both the model-based and learning-based algorithms as a function of the swarm size. In model-based APF swarm control, the mathematical structure of the problem follows a general form:

$$\vec{F} = \sum_{m=1}^{M} NL_m(\vec{d}_m) \tag{16}$$

where $\vec{F}$, NL_m, and $\vec{d}_m$ represent the aggregated potential field force vector, the mth nonlinear function, and the mth distance vector, respectively, while M is the total number of vectors. On the other hand, for learning-based cooperative RL algorithms, as the Q-table is approximated by the neural network, the general computation paradigm is the same as computing each neuron's output

$$y_j = a\left(\sum_{i=1}^{N} w_{i,j}(x_i)\right) \tag{17}$$

where x, w, y, and a are the inputs, weights, neuron outputs, and nonlinear activation functions, respectively, while N is the number of pre-synaptic neurons. It is easy to understand that M will scale with swarm size, especially in applications such as pattern formation. Similarly, N is determined by the dimension of the global states of the system, which scales with the swarm size. As a result, a larger swarm will require a wider range of operands, thus requiring a higher bit precision to correctly process APF algorithms as well as cooperative RL. Fig. 3(a) and (b) shows the simulation results of representing



No. Bits	TD-MS		HDMS	
	Ave	Worst	Ave	Worst
3	0.10	0.49	0.19	0.52
4	0.14	0.56	0.16	0.61
5	0.28	0.72	0.29	0.74
6	0.64	1.74	0.69	0.94
7	2.21	3.86	0.70	1.02
8	5.82	9.32	0.69	1.27

(d)

Fig. 5. Energy map versus an operand range in pJ for (a) digital, (b) TDMS, and (c) HDMS MAC implementations. (d) Energy/MAC (normalized to a digital implementation) for TDMS and HDMS implementations. We see that HDMS outperforms TDMS (average and worst cases) and digital (average case) for large swarm sizes.

the required range of the operands for different swarm sizes in both physical model-based (coordinated path planning) and learning-based (multiple predator-prey) template algorithms. We note that as the swarm size increases, the bit precision required to correctly compute also increases. The simulation results can be summarized in Fig. 3(c), where the template algorithms that can be supported require a bit-width of 3 b to a maximum of 8 b. In these applications, the sensor data are assumed to have a bit-width of 8 b or less, and obstacle avoidance is performed using ultrasonic sensors.

V. Hybrid-Digital-Mixed-Signal Computing

The advantage of using analog and mixed-signal design principles for energy-efficient computing has been demonstrated in [8], [23], and [24]. More recently, there has been an increasing interest in time-based mixed-signal computing. Here, information is represented in a phase or frequency domain, and hence, the effective number of bits is not limited by the voltage scalability of the design. However, since the data are processed in the time domain, the system throughput is lower than the corresponding digital systems. For many problems of practical interest, in particular, for control and robotics on small form factors where the data-processing speed is relatively low, this is a favorable tradeoff. It has been demonstrated successfully in RL problems [8], [23] as well as in convolutional neural networks [25], decoders [26], and pipelines circuits [27]. In spite of its superior energy efficiency at low bit-widths (typically less than 5 or 6 b depending on the process), it is well understood that as the bit precision scales to high values, the energy efficiency of digital circuits takes over. Hence, for the problem at hand, where an increasing swarm size should be supported with a higher bit-width, an ideal system should scale seamlessly between a mixed-signal (time-based) to a digital design, such that high energy efficiency is obtained, as the system specifications scale.

A. Time-Domain Multiplication and Accumulation

The details of time-domain MAC have been described in [8] and will be summarized here for completeness. Fig. 4(a)–(c) shows the time-based MAC circuit. The time-based circuit operates on the 5-b data, representing both positive and negative numbers. It has a pulse input (T_p) used as the "enable" signal to an up-down counter. Signed operation is handled by XOR operation of sign bit, as shown in Fig. 4(a). One of the operands (X[0:4]) is encoded in the pulsewidth of T_p using a digital-to-pulse-converter (DPC) with X[4] as a sign bit. For the ith input X_i, we obtain

$$T_{p_i} = X_i * T_0 \tag{18}$$

where T_0 is the unit time-constant for the DPC. The other input (Y[0:4]) is encoded in the signed magnitude format and controls a digitally controlled oscillator (DCO). Y[4] represents the sign bit, and Y[0:3] represents the magnitude of the second operand. The three-stage DCO converts the digital value to a frequency proportional to Y[0:3]. Each stage of the DCO consists of a bank of parallel binary-sized inverters controlled by the digital value (Y[0:3]), as shown in Fig. 4(b) and (c). The frequency of the DCO for the jth word (Y_j) is F_j, and ignoring the second-order effects such as non-linearity is given by

$$F_j = Y_j * F_0 \tag{19}$$

where F_0 is the unit frequency of the DCO corresponding to a *code* 1 when $W = 00001$. The clock to the counter is driven by the DCO, and the enable signal is controlled by the pulsewidth (T_{p_i}). Hence, the counter output is given by

$$DT_{ij} = T_{p_i} * F_j = X_i * Y_j * F_0 * T_0. \tag{20}$$

From (20), we can observe that the counter output is proportional to the product of the two operands. As shown in Fig. 4(a), the polarity of MAC is taken care of through up-down knob of the counter controlled by an XOR of X[4] and Y[4]. The constants, F_0 and T_0, represent the overall system throughput and designed to maintain correct functionality amidst non-linearities. The scalability of this design to a large number of vector-parameters has been discussed in [23] and [24].

B. Hybrid-Digital-Mixed-Signal Computing Platform

It is worth noting that the time-domain MAC shows high energy efficiency for low bit-widths only. Fig. 5(a) and (b) shows the simulation results of a 65-nm CMOS GP process,

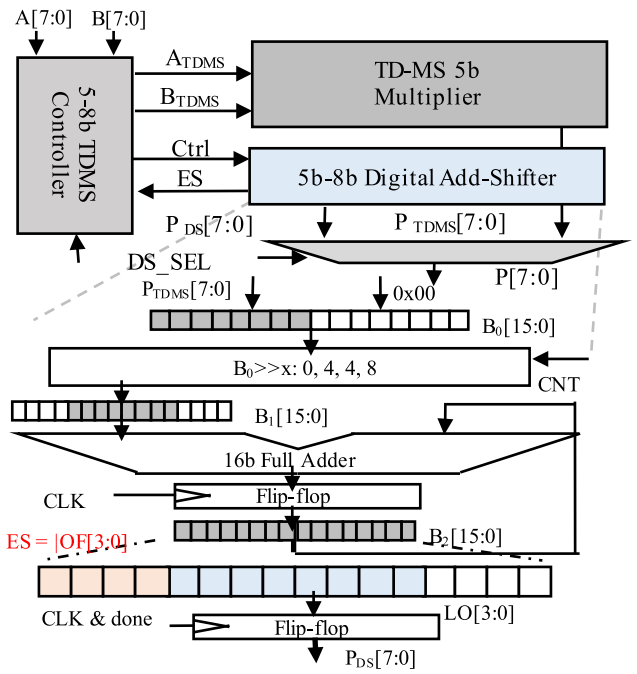


Fig. 6. Circuit schematic of the HDMS circuit illustrating the 5-b TDMS kernel and the digital peripherals to enable efficient scaling to 8 b.

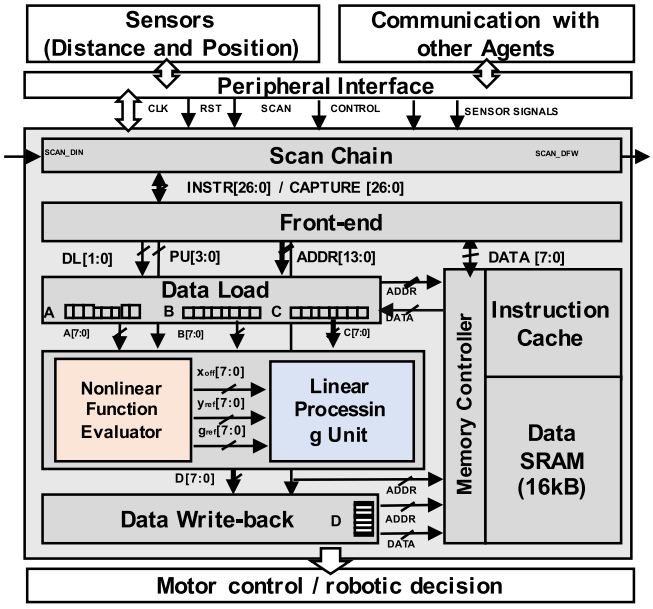
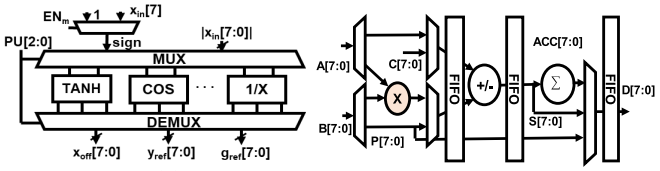


Fig. 7. Overall system architecture of the unified computing platform.



PU	FUNC	EN	seg	#comp
000	Tanh	1	10	5
001	Cosine	0	12	6
010	Sqrt	1	4	4
011	step	1	2	1
100	ReLU	1	2	1
101	X^2	0	12	6
110	1/X	1	10	5

(b)

INS	OP	#Bytes	#Clk
000	A+B	2	1
001	A-B	2	1
010	AB+C	2 / 3	2 / 6
011	∑ A	N	N+1
100	∑ AB	N / 2N	N+2/5
101	∑ AB+C	2 / 3N	N+3/6

(d)

Fig. 8. (a) Circuit schematic and (b) corresponding control bits for the NFE. (c) Circuit schematic and (d) instructions for the LPU.

and it reveals that the energy consumed for an MAC operation scales faster than a digital system. This can be intuitively understood from the fact that the number of switching events (in the worst case) for a time-domain phase–frequency-based design scales as 2^N for N-bit operands. This results in an interesting artifact, where important computation (where operands have higher magnitudes) consumes more energy than less important computation (where operands have lower magnitudes). The 2-D energy bar shown in Fig. 5 shows how a time-domain system displays high energy efficiency for bit-width less than 5, but it increases dramatically as the bit-width increases. To maintain high efficiency across the entire operating range, we propose an HDMS MAC kernel, as shown in Fig. 6.

The HDMS MAC kernel consists of a conventional TDMS multiplier, a 5–8-b digital adder and shifter, and a 5–8-b TDMS controller. For bit precisions less than 5 b, the circuit is operated completely in the time domain. The idea is to compute an 8-bit multiplication via shift-and-add. At the core, we have an energy-efficient time-domain 5-b multiplier. Around that, we have peripheral circuits (add-shifter and controller) to reconfigure the multiplier to higher bit precision, as needed by allowing seamless shift and add operations. The 5–8-b digital add-shifter circuit diagram is shown in Fig. 6. Fig. 6 shows the HDMS circuit: a shifter shifts the TDMS products by 0, 2, 4, and 8 bit each time and accumulates through time with a 16-b full adder. The computation starts from the most significant bit and proceeds to the least significant bit (LSB). This helps us to save unnecessary switching by stopping the computation as soon as any overflow is detected through the embedded overflow detection. By driving a digital select signal (DS_SEL) active, the 5–8-b TDMS controller splits 8-bit input operands A and B into 4-bit components, passes them to the TDMS multiplier in pairs, and controls the add-shifter to produce high-precision output. With the proposed kernel, we are able to preserve the energy efficiency of the time-domain computation for lower bit precision while leveraging the efficiency of digital computation for higher bits of precision. The energy map of HDMS is shown in Fig. 5(c), and TDMS/HDMS energy normalized to a digital circuit with the same bit precision is shown in the table in Fig. 5(d). It should also be noted that the proposed scheme is scalable to handle more than 8-b operations.

We should also note that, although HDMS requires additional clock cycles (4×) than TDMS, it still shows higher throughput; owing to the fact that HDMS avoids long clock periods, typical of 8-b TDMS (16×). With both energy and throughput advantages, the major tradeoff is the additional area required for the digital peripherals. However, it should

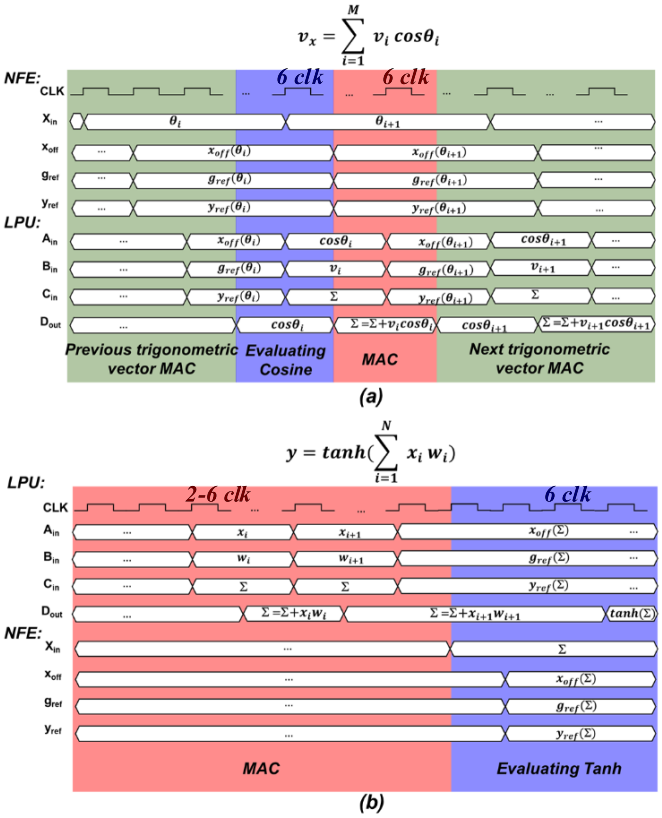
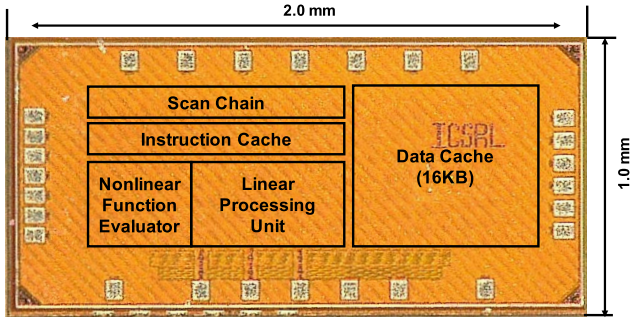


Fig. 9. Clock diagram for examples in (a) APF and (b) cooperative RL.



Chip Characteristics	
Technology	65nm 1P9M CMOS
Die area	1mm*2mm
Testing interface	QFN package
Pin Count	28

Fig. 10. Die photograph and chip characteristics.

be noted that HDMS achieves lower throughput than high-speed digital. In the current application, the throughput that we achieve is more than sufficient to support the data rate for the sensors and actuators.

VI. System Overview

The system architecture of the proposed computation platform is shown in Fig. 7. As mentioned in Section IV, we have noted that APF and cooperative RL are essential combinations of nonlinear evaluations and linear operations. This has inspired us to design a dedicated accelerator for nonlinear and linear computations, which are called the nonlinear function evaluator (NFE) and the LPU, respectively. NFE implements the non-linear function using the piecewise linear approximation of the nonlinear functions. We embed a number of widely used nonlinear functions in the NFE. By choosing the function to evaluate and provide the input parameter, NFE generates an offset (x_{off}), a reference gradient (g_{ref}), and a reference offset (y_{ref}) in one clock cycle. The corresponding evaluation result is generated by multiplication/addition of x_{off}, g_{ref}, and y_{ref} in the LPU. The number of clock cycle depends on the bit precision selected. We observe that many of the required functions show symmetry or periodicity, and we take advantage of that to implement a mapping mechanism to reduce the number of comparisons and computations. This saves active die-area as well as computational energy. The reference parameters are stored in an LUT. By storing only the important parameters, determined from the range of the inputs and by interpolating in the LPU, NFE is achieving target accuracy for the entire range of the data. As opposed to using an LUT for the complete range of inputs, the proposed design allows a compact implementation with a reduced memory footprint. On the other hand, the LPU supports all the linear operations (addition and multiplication). Most operations are implemented in the digital domain except for MAC. Circuit and control details of NFE and LPU are shown in Fig. 8(a)–(d).

We provide the bi-directional local data path between LPU and NFE for computations. Data can move between the LPU and the NFE seamlessly to preserve the data locality.

A 16-kB on-chip SRAM is embedded together with an instruction cache, a data loader, and write-back controllers. A front-end controller is also provided, and the design is full-scan. It should be noted that, either in model-based or learning-based applications, required information storage will scale with the swarm size and the complexity of the environment. The current design is a prototype with the 16-kB on-chip memory. For more complex "experience maps," off-chip storage is required. This is not supported in the current test chip. With the embedded computation/storage capability, the chip is able to interface with the sensors and communication components for swarm robotics. The sensors and actuators interface through a Raspberry Pi, which acts simply as an interface. All the sensors produce digital outputs. Ultrasonic sensors are used for depth measurements. Inertial measurement units are used to estimate position, by integrating into the Raspberry Pi. In future work, the system may be scaled to enable more complex mapping and localization algorithms. With limited on-chip resources, this test chip is intended to work as a co-processor to support key algorithms and applications. Sample timing diagrams for two tasks, one for APF algorithm and one for the cooperative RL, are shown in Fig. 9.

VII. Measurements

The proposed computational platform is implemented and taped-out in a 65-nm GP CMOS process. It occupies a total

TABLE I

BENCHMARKING TABLE SHOWING COMPETITIVE FIGURES-OF-MERIT COMPARED WITH SIMILAR HARDWARE ACCELERATORS

	This work	[8]	[28]	[29]	[30]	[16]	[25]	[27]
Application	Swarm Learning	Autonomous micro-robotics	CNN Inference	DNN Inference	CNN Inference	CNN Inference	CNN Inference	DTW
Optimization algorithm	Cooperative RL/potential field	Reinforcement Learning	none	none	none	none	none	Time-series Classification
Learning/Training	Online real-time	Online real-time	offline	offline	none	offline	offline	none
Technology	65nm	55nm	180nm	65nm	65nm	65nm	40nm	65nm
Aream	2mm^2	3.4mm^2	3.3mm^2	16mm^2	16mm^2	16mm^2	0.124mm^2	1.67mm^2
On-die SRAM	16KB	200B	144KB	36KB	490.5KB	181.5KB	/	/
Resolution	5-8b	6b	4b-16b	16b	16b	16b	8/1b	4/10b
Power	0.3-3.4μW	650μW	7.5-300mW	45mW	6.57mW	278mW	28.67μW	35-136mW
Frequency	1KHz-1.5MHz	67.5MHz	200MHz	125MHz	10-100MHz	200MHz	24MHz	110MHz
Supply voltage	0.4-1V	0.4-1V	1V	1.2V	0.7-1.2V	0.82-1.17V	0.375-1.1V	0.7V
Performance (TOPS/W)	1.1-9.1	3.12	0.26-10.0	1.42	/	0.21	4.65-12.08	/
Norm Performance (TOPS/W·Byte)	1.1-3.4	2.34	0.52-5.0	2.84	/	0.42	1.51-4.65	/

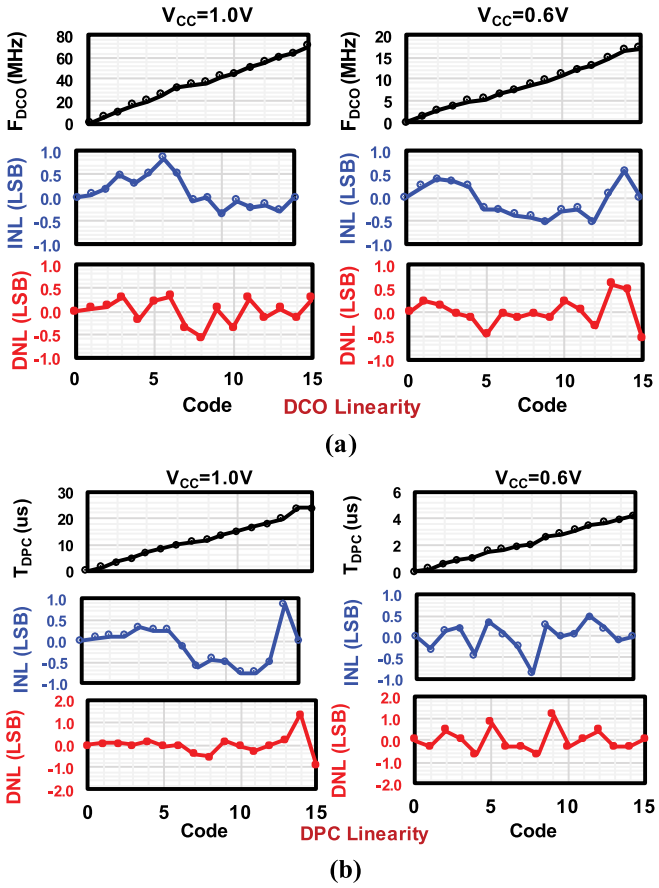


Fig. 11. Measured linearity of (a) DCO and (b) DPC.

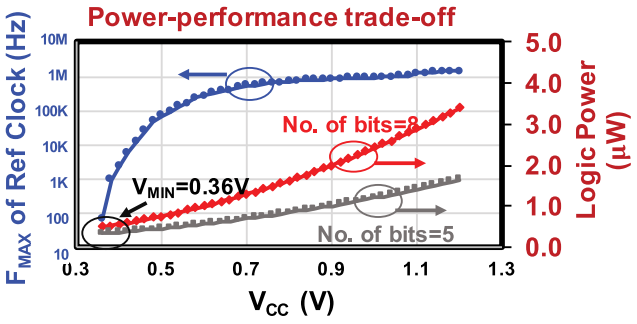


Fig. 12. Measured power-performance tradeoff.

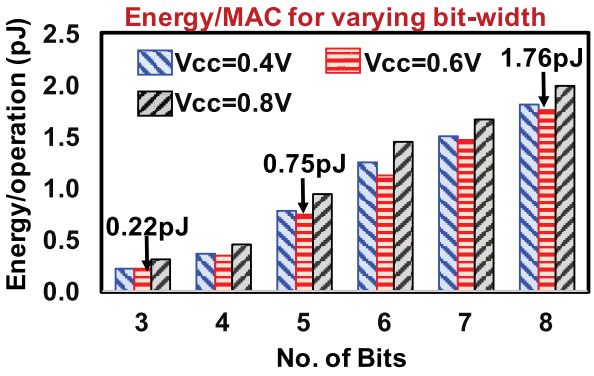


Fig. 13. Measured energy per MAC across for different bit-widths at V_{CC} = 0.4, 0.6, and 0.8 V.

area of 2 mm^2 and is packaged in a chip-size quad flat no-leads package. The chip die photograph and characteristics are shown in Fig. 10. Since the TDMS circuits use mixed-signal DCO and DPCs, we characterize their non-linearities at two different voltages ($V_{CC} = 1.0$ V and $V_{CC} = 0.6$ V). The worst case integral nonlinearity and differential nonlinearity range from -1.0 to 1.1 LSB, as shown in Fig. 11. The measured power-performance tradeoff is shown in Fig. 12. We note a measured peak F_{MAX} of 1.5 MHz and correct functionality down to V_{MIN} of 0.36 V, below which the embedded SRAM arrays cease to function. The processing throughput scales with supply voltage and thus clock frequency. We measure a logic-power dissipation of 3.2 μW (1.9 μW) for 8-b (5-b) operations. The measured energy/operation (in Fig. 13) shows high scalability with the bit resolution, illustrating a peak of energy efficiency of 0.22 (at 3 b) and 1.76 pJ/MAC (at 8 b). We note that at low bit-widths, the TDMS circuit cores show superior energy efficiency, while the digital peripherals allow almost linear energy-scaling for 5–8 b. We also measure the average arithmetic energy efficiency as a function of the supply

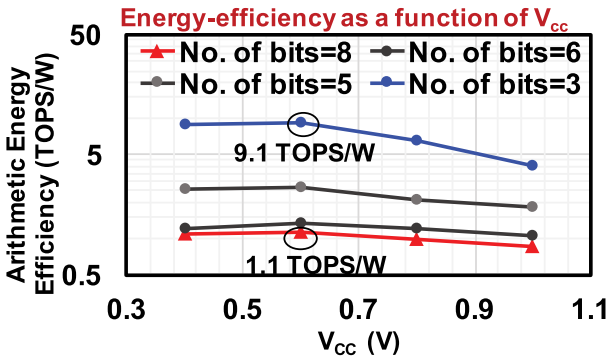


Fig. 14. Measured arithmetic energy efficiency as a function of the operating voltage for different bit-widths.

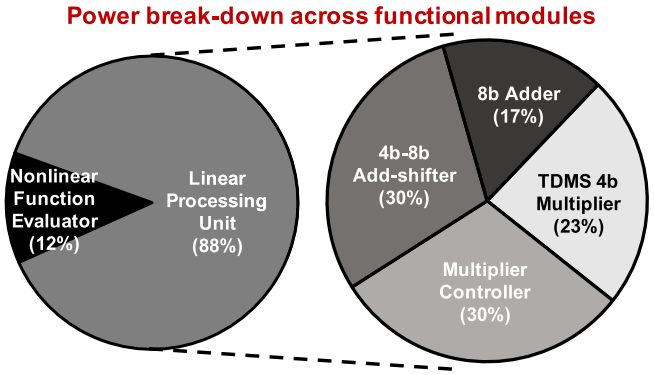


Fig. 15. Measured power breakdown among different computational blocks.

voltage and record a 9.1 TOPS/W (for 3-b operations), and it decreases to 1.1 TOPS/W (for 8-b operations) as is shown in Fig. 14. This shows how the bit-resolution scalability allows efficient operations for multiple bit-widths and hence swarm sizes. We plot the energy breakdown of the computation unit in Fig. 15 and show that the LPU and the NPE consume 88% and 12% of the logic power, respectively. The power distribution across various blocks of the LPU is further shown, and all the components contribute equally in the power dissipation.

The test chip is integrated and mounted on an application platform. It is used as a controller for a robotic car as shown in Fig. 16(a) and (b) and interfaces with a Raspberry Pi, motor controllers, sensors, and LoRA radios. The convergence of cooperative RL is shown in Fig. 16(c). The neural-network Q-approximator has two layers, and each layer has 100 neurons. Through hyper-parameter tuning, this setting results in the best performance under the constraints of the limited on-chip memory. Here, the inference is implemented in 5-b TDMS and learning in 8-b HDMS. In either mode, the non-linearity of the DPC and DCO (post-calibration) does not affect the accuracy of the algorithms. Particularly during learning, the digital peripheral circuits for HDMS reduce the impact of non-ideality and can successfully train the network. Furthermore, for applications requiring higher bit precision, the proposed HDMS can be scaled to 12–16 b. A video demonstration of this can be found in `https://www.youtube.com/watch?v=_NqdJabFJKo`. In this video,

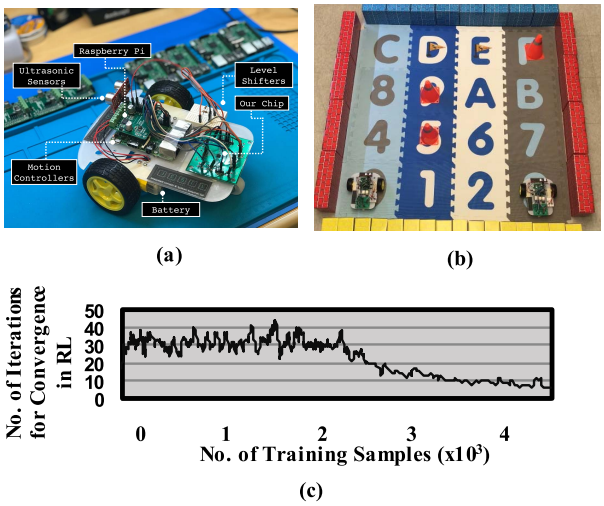


Fig. 16. (a) Test chip mounted on a robotic car with peripheral circuits. (b) Experimental setup. (c) Number of iterations required for convergence in cooperative RL.

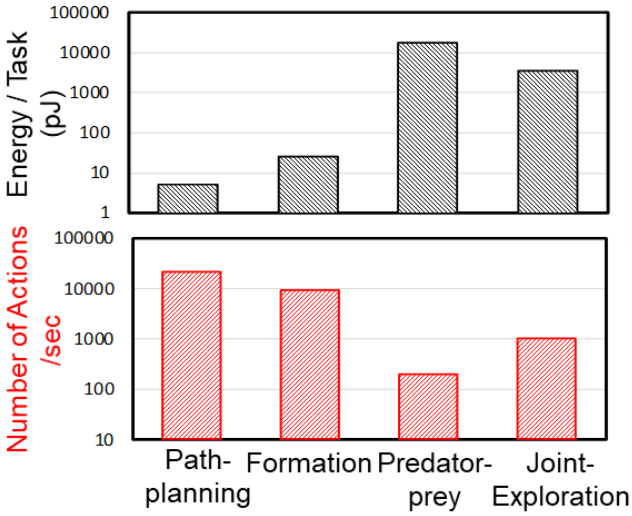


Fig. 17. Application-level benchmarking demonstrating measured energy/performance for different template algorithms.

we demonstrate a two-robot cooperative learning task for predator-prey applications. The swarm size can also be scaled in the future for more complex demonstrations. However, the current design is limited in the number of sensor interfaces that it can handle; and further modifications are required to develop real-time demonstrations of larger swarms. We implement four template swarm algorithms, namely, path planning, pattern formation, predator-prey, and joint exploration. The first two are based on the physical and mathematical models, and the last two are based on the learning algorithms. We measure the total energy as well as the number of actions taken per second for each of these tasks in sample environments. These are plotted in Fig. 17, and we note a large variation in both the energy cost and the number of actions per second for these template problems. This also illustrates the wide variety of algorithms [simultaneous localization and mapping (SLAM)

and vision-based path planning] that need to be supported in the future robotic controllers, as the complexities of the environments and the cost functions can dramatically change.

The test chip has been benchmarked against similar designs and shows competitive figures-of-merit, as shown in Table I. This is also the first demonstration of a unified and programmable platform that can accelerate a large class of algorithms for swarm robotics with efficient scalability in terms of swarm sizes and application.

VIII. Outlook

Swarm robotics is computationally challenging, and the proposed test chip is a prototype to demonstrate some key enabling features. The first challenge is scalability. The current design is limited by the on-chip memory and the number of interfaces. Future platforms can extend the design by incorporating off-chip memory to store complex "experience maps." For higher bit precision, the HDMS circuits need to be evaluated, in particular, when support for more complex algorithms is required. Furthermore, to support advanced applications, such as SLAM and vision-based navigation, the current design needs to enable with higher throughput and near-/in-memory computing. Finally, higher throughput can be supported through an array of LPUs and NPEs, which can parallelize the algorithms.

IX. Conclusion

This article presents a 65-nm CMOS platform that supports both model-based and learning-based algorithms for swarm robotics. The proposed HDMS computational unit provides excellent scalability with swarm sizes. We measure a peak energy efficiency of 9.1 TOPS/W. The test chip is integrated with the peripheral controllers and sensors and mounted on a robotic car. Sample algorithms have been executed and benchmarked.

References

[1] Z. Shi, J. Tu, Q. Zhang, L. Liu, and J. Wei, "A survey of swarm robotics system," in *Advances in Swarm Intelligence* (Lecture Notes in Computer Science), Y. Tan, Y. Shi, and Z. Ji, Eds. Berlin, Germany: Springer, 2012, pp. 564–572.

[2] H. M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 52–63, Jan. 2015.

[3] R. Gross, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1115–1130, Dec. 2006.

[4] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 927–937, Aug. 2009.

[5] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, Apr. 2007, pp. 332–339.

[6] D. P. Stormont, "Autonomous rescue robot swarms for first responders," in *Proc. IEEE Int. Conf. Comput. Intell. Homeland Secur. Pers. Saf. (CIHSPS)*, Mar./Apr. 2005, pp. 151–157.

[7] S. Taranovich. *Efficient Powering of a Robot Swarm | EDN*. Accessed: Aug. 2016. [Online]. Available: https://www.edn.com/design/power-management/4442493/Efficient-powering-of-a-robot-swarm

[8] A. Amravati, S. B. Nasir, S. Thangadurai, I. Yoon, and A. Raychowdhury, "A 55nm time-domain mixed-signal neuromorphic accelerator with stochastic synapses and embedded reinforcement learning for autonomous micro-robots," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 124–126.

[9] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, May 1990, pp. 500–505.

[10] M. C. Lee and M. G. Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, vol. 2, Jul. 2003, pp. 735–740.

[11] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 2, pp. 224–241, Mar./Apr. 1992.

[12] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proc. Congr. Evol. Comput.*, vol. 1, Jul. 2000, pp. 256–263.

[13] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, vol. 2, Oct. 2006, pp. 622–627.

[14] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 242–243.

[15] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018.

[16] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 367–379.

[17] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[18] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 79–86.

[19] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.

[20] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Mach. Learn.*, vol. 23, pp. 279–303, May 1996.

[21] M. A. Anwar and A. Raychowdhury, "NavREn-Rl: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images," in *Proc. 25th Int. Conf. Mechatronics Mach. Vis. Pract. (M2VIP)*, Nov. 2018, pp. 1–6.

[22] S. Peng, "A generalized dynamic programming principle and Hamilton-Jacobi-Bellman equation," *Stochastics Stochastic Rep.*, vol. 38, no. 2, pp. 119–134, Feb. 1992.

[23] A. Amaravati, S. B. Nasir, J. Ting, I. Yoon, and A. Raychowdhury, "A 55-nm, 1.0–0.4V, 1.25-pJ/MAC time-domain mixed-signal neuromorphic accelerator with stochastic synapses for reinforcement learning in autonomous mobile robots," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 75–87, Jan. 2019.

[24] N. Cao, M. Chang, and A. Raychowdhury, "14.1 A 65 nm 1.1-to-9.1tops/W hybrid-digital-mixed-signal computing platform for accelerating model-based and model-free swarm robotics," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 222–224.

[25] A. Sayal, S. Fathima, S. S. T. Nibhanupudi, and J. P. Kulkarni, "All-digital time-domain CNN engine using bidirectional memory delay lines for energy-efficient edge computing," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 228–230.

[26] D. Miyashita *et al.*, "An LDPC decoder with time-domain analog and digital mixed-signal processing," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 73–83, Jan. 2014.

[27] Z. Chen and J. Gu, "A scalable pipelined time-domain DTW engine for time-series classification using multibit time flip-flops with 140Giga-cell-updates/s throughput," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 324–326.

[28] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[29] J. Sim, J.-S. Park, M. Kim, D. Bae, Y. Choi, and L.-S. Kim, "A 1.42tops/W deep convolutional neural network recognition processor for intelligent IoE systems," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Jan./Feb. 2016, pp. 264–265.

[30] S. Choi, J. Lee, K. Lee, and H.-J. Yoo, "A 9.02mw CNN-stereo-based real-time 3d hand-gesture recognition processor for smart mobile devices," in *IEEE Int. Solid-State Circuits Conf. ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 220–222.

Ningyuan Cao (M'17) received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2013, and the M.S. degree in electrical engineering from Columbia University, New York, NY, USA, in 2015. He is currently pursuing the Ph.D. degree with the Georgia Institute of Technology, Atlanta, GA, USA.

He has been with the Integrated Circuit and System Research Laboratory (ICSRL), Georgia Institute of Technology, since 2015. His research interests include low-power machine learning ASIC design, wireless sensor design, power management, and energy harvesting circuit design.

Muya Chang is currently pursuing the dual M.S. degree in computer science and the Ph.D. degree in electrical and computer engineering (ECE) with the Georgia Institute of Technology, Atlanta, GA, USA.

He is also a member of the Integrated Circuits and Systems Research Laboratory, Georgia Institute of Technology, and is advised by an ECE Associate Professor A. Raychowdhury. His research interests include energy-efficient hardware design for distributed optimizations.

Arijit Raychowdhury (M'07–SM'13) received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2007.

His industry experience includes five years as a Staff Scientist with the Circuits Research Laboratory, Intel Corporation, Portland, OR, USA, and a year as an Analog Circuit Researcher with Texas Instruments Inc., Bengaluru, India. He joined the Georgia Institute of Technology, Atlanta, GA, USA, in 2013, where he is currently an Associate Professor with the School of Electrical and Computer Engineering and also holds an ON Semiconductor Junior Professorship. He holds more than 25 U.S. and international patents and has published over 100 articles in journals and refereed conferences. His research interests include low-power digital- and mixed-signal circuit design, device–circuit interactions, and novel computing models and hardware realizations.

Dr. Raychowdhury was a recipient of the Intel Early Faculty Award in 2015, the NSF CISE Research Initiation Initiative Award (CRII) in 2015, the Intel Labs Technical Contribution Award in 2011, the Dimitris N. Chorafas Award for outstanding doctoral research in 2007, the Best Thesis Award from the College of Engineering, Purdue University, in 2007, and multiple best paper awards and fellowships.