ARTIFICIAL INTELLIGENCE

Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots

Songchen Ma[1]†, Jing Pei[1]†, Weihao Zhang[1]†, Guanrui Wang[1,2]†, Dahu Feng[1]†, Fangwen Yu[1], Chenhang Song[1], Huanyu Qu[1], Cheng Ma[1], Mingsheng Lu[1], Faqiang Liu[1], Wenhao Zhou[1], Yujie Wu[1], Yihan Lin[1], Hongyi Li[1], Taoyi Wang[1], Jiuru Song[1], Xue Liu[1], Guoqi Li[1], Rong Zhao[1], Luping Shi[1]*

Recent advances in artificial intelligence have enhanced the abilities of mobile robots in dealing with complex and dynamic scenarios. However, to enable computationally intensive algorithms to be executed locally in multi-task robots with low latency and high efficiency, innovations in computing hardware are required. Here, we report TianjicX, a neuromorphic computing hardware that can support true concurrent execution of multiple cross-computing-paradigm neural network (NN) models with various coordination manners for robotics. With spatiotemporal elasticity, TianjicX can support adaptive allocation of computing resources and scheduling of execution time for each task. Key to this approach is a high-level model, "Rivulet," which bridges the gap between robotic-level requirements and hardware implementations. It abstracts the execution of NN tasks through distribution of static data and streaming of dynamic data to form the basic activity context, adopts time and space slices to achieve elastic resource allocation for each activity, and performs configurable hybrid synchronous-asynchronous grouping. Thereby, Rivulet is capable of supporting independent and interactive execution. Building on Rivulet with hardware design for realizing spatiotemporal elasticity, a 28-nanometer TianjicX neuromorphic chip with event-driven, high parallelism, low latency, and low power was developed. Using a single TianjicX chip and a specially developed compiler stack, we built a multi-intelligent-tasking mobile robot, Tianjicat, to perform a cat-and-mouse game. Multiple tasks, including sound recognition and tracking, object recognition, obstacle avoidance, and decision-making, can be concurrently executed. Compared with NVIDIA Jetson TX2, latency is substantially reduced by 79.09 times, and dynamic power is reduced by 50.66%.

INTRODUCTION

The long-term goal for mobile robots is to be able to approximate human-level intelligence when dealing with complex and unknown environments. In recent years, with the rapid development of artificial intelligence (AI), various neural network (NN) algorithms have been widely used in robots to perform specific intelligent tasks with improved performance, such as convolutional NNs (CNNs) for image recognition and detection tasks (*1*, *2*); spiking NNs (SNNs) for high-speed tracking (*3*, *4*), event-driven sensing (*5*, *6*), and brain-computer interfaces (*7*, *8*); and recurrent NNs for sequence learning (*9*) in path planning and natural language processing (*10*). Moreover, the application of multiple models (*11*–*13*) and multi-tasking NNs (*14*–*17*) can further improve the ability and adaptability of robots. However, NN algorithms are usually computationally intensive. To efficiently implement various and multiple NNs on mobile robots, several major requirements need to be considered for computing hardware. First, it must support concurrent execution of multiple neural models locally with low latency, which is the key to improving the real-time processing capability. Second, it must endow the flexibility of deploying multiple models to achieve a balance between low latency, high efficiency, and high concurrency in dealing with dynamic scenarios. Third, it must support asynchronous executions and flexible interactions to acquire high hardware utilization and achieve adaptability in open environments.

Current computing hardware solutions face different difficulties in meeting these requirements. Modern general-purpose processors usually cannot provide massively parallel computing, resulting in low cost-effectiveness and high power when performing NNs in robotic systems. Graphic processing units (GPUs) have strong programmability and high parallelism. However, GPUs suffer from high power consumption and underutilization (*18*, *19*) because of frequent off-chip memory access and interaction with central processing units (CPUs). Recently, many kinds of AI computing hardware with greatly improved performance have emerged. Field-programmable gate array (FPGA)– and application-specific integrated circuit–based deep learning accelerators (*20*–*23*) can provide high efficiency via customized architecture optimization, especially for artificial NNs (ANNs). They usually provide massively uniform parallel computation based on single-instruction multiple-data architecture and try to increase the peak speed to the limit. These ANN accelerators are based on von Neumann architecture and take full advantage of the temporal complexity in the design of execution models. The processing element (PE) array is scheduled by the CPU to perform unified operations in the same execution period, which enables accelerators to work in time slicing. However, with the notable increase in the diversity of NNs and multitasks, hardware utilization and scheduling flexibility are facing great challenges (*24*–*26*). Frequent switching of NN contexts leads to redundant delays and energy costs, making it extremely difficult for accelerators to simultaneously implement multiple different algorithms with high performance in robots. Neuromorphic computing chips (*27*–*30*) are

[1]Center for Brain-Inspired Computing Research (CBICR), Beijing Innovation Center for Future Chip, Optical Memory National Engineering Research Center, Department of Precision Instrument, Tsinghua University, Beijing 100084, China. [2]Lynxi Technologies Co. Ltd, Beijing, China.
*Corresponding author. Email: lpshi@tsinghua.edu.cn
†These authors contributed equally to this work.

good candidates for simultaneously executing multiple NN models due to the decentralized non–von Neumann architecture, which can use the spatial and temporal complexity. However, the current neuromorphic chips usually preconfigure cores and process NNs in a pipelined way through space slicing. Each core repeatedly performs preconfigured operations in different execution cycles, causing inflexibility in resource allocation and eventually underutilization of resource. In summary, the inherent bottlenecks of the underlying architecture or execution models prevent existing computing hardware from implementing multiple intensive algorithms locally with low latency and high efficiency. Innovations in computing hardware with efficient computing for truly mobile multitasking robotics are highly desired.

In this article, we report a neuromorphic computing hardware, TianjicX, which has a spatiotemporal elasticity during the tasks' execution and cooperation. Spatiotemporal elasticity is defined as the ability of hardware to adaptively allocate computing resources and schedule execution time for a task. TianjicX can perform true concurrent execution of cross-computing-paradigm NN models, including ANNs, SNNs, and the hybrid of both, for multi-intelligent-tasking (MITs) robotics. When designing the computing hardware for MITs robotics, one key challenge is meeting the performance requirements of latency-concurrency-power (LCP), particularly for various NNs implementations. Another key challenge is maintaining the independent execution of each task without interference while providing support for intertask interactions. In overcoming these challenges, a set of designs are carried out from different levels, including architecture, chip, and model deployment, as illustrated in Fig. 1. At the architecture level, inspired by traditional computers, we first propose an execution model, Rivulet, to bridge the gap between robotic-level requirements and specific hardware implementation, laying the architecture foundation of TianjicX. A rivulet abstracts the basic NN execution activities and unifies ANNs and SNNs with "static data" and "dynamic data," which provides a concrete operable and describable entity for resource allocation and task scheduling. Then, a resource model adopting both time and spatial slice is developed to manage multi-rivulets. Furthermore, executions of rivulets are designed in a hybrid synchronization and

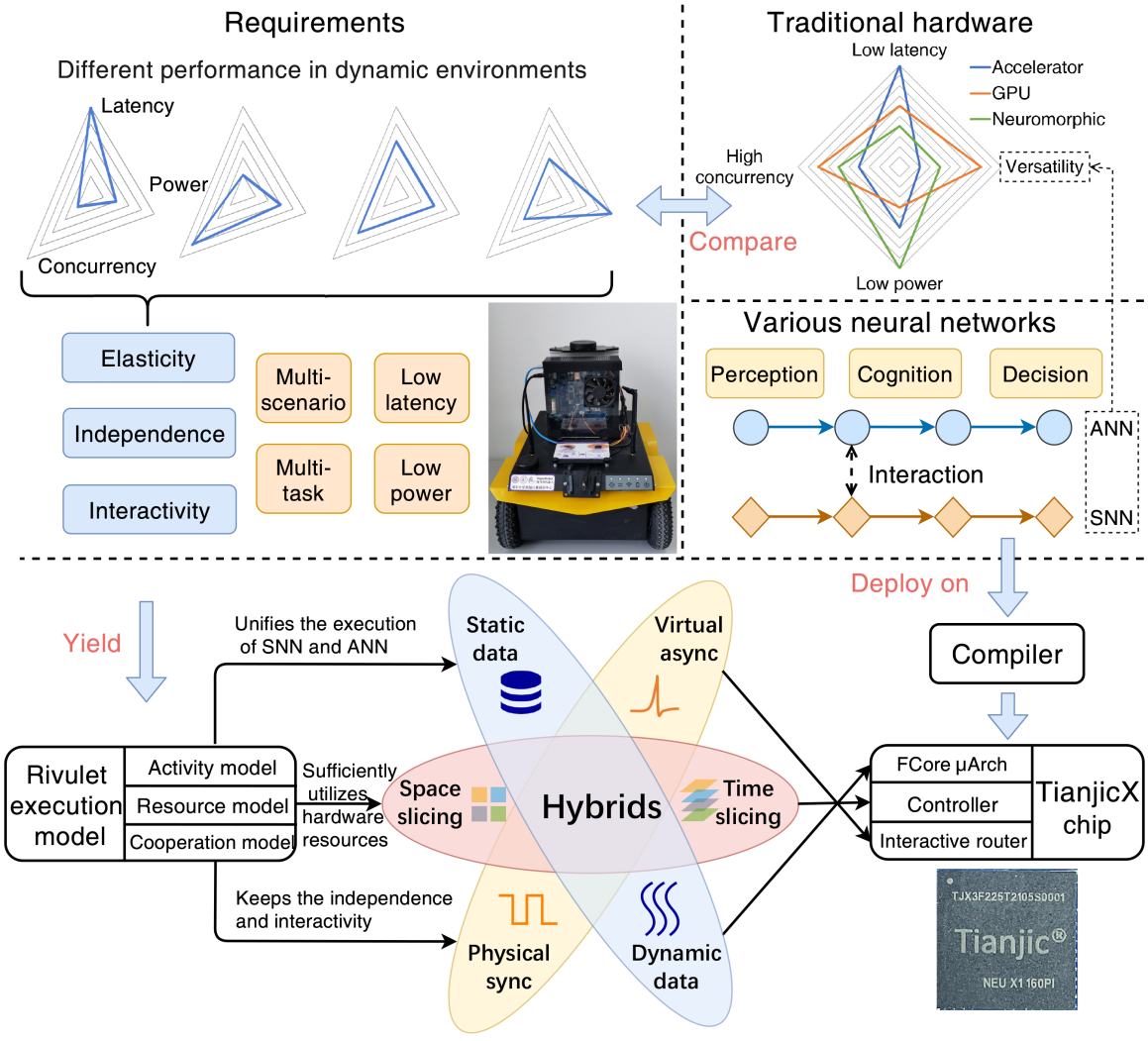

Fig. 1. A neuromorphic computing platform for multi-intelligent-tasking robotics. The key designs of the platform include three levels: a Rivulet execution model, a TianjicX chip with a specially designed compiler, and a robotic system based on the TianjicX.

asynchronization manner through virtual grouping to enable multiple independent or interacting rivulets.

On the basis of Rivulet-based architecture, we design and fabricate a TianjicX chip based on 28-nm complementary metal-oxide semiconductor (CMOS). It adopts the many-core decentralized architecture that is widely used for neuromorphic chips and integrates 160 configurable cross-computing paradigm cores with massive parallel computing units, abundant on-chip memory, and an arbitrarily configurable primitive sequence for each core. A unified primitive instruction set is built to support the high programmability and versatility of ANNs, SNNs, and cross-modeling. In the core, a dedicated controller that balances computation and scheduling manages each core locally to promote versatility and efficiency. Furthermore, through event-driven design and multilevel grouping of cores, different NNs can be executed asynchronously according to the dynamic environmental changes and perform global interactions. A compiler stack adopting a spatiotemporal mapping method is further developed for model deployment to make full use of the elasticity of TianjicX, which can flexibly colocate multiple tasks according to the actual needs of different scenarios. The TianjicX chip exhibits a peak dynamic energy efficiency of 3.2 tera operations per second (TOPS)/W and on-chip memory bandwidth of 5.12 terabytes/s. Because of the optimization of on-chip memory utilization, it achieves a computing power per unit area as high as 0.2 TOPS/mm^2.

Using a single TianjicX chip, we built a mobile robot, namely, Tianjicat, and designed a cat-and-mouse game in which Tianjicat plays the role of a cat. We demonstrate that the robot can perform multiple tasks in real time, such as sound recognition, sound source localization, objection detection and recognition, obstacle avoidance, and decision-making, which are realized by implementing different ANN and SNN models. The cooperation and data transmission between tasks do not need to be scheduled through other hardware because of the instant primitive mechanism. The response delay of multiple networks on TianjicX is reduced by about 80 times compared with NVIDIA Jetson TX2. The entire robot works in an event-driven manner with high parallelism. Each NN only executes when the corresponding input from the external sensor is updated, leading to low power consumption. For the cat-and-mouse game, 128 cores within one TianjicX chip are used, and the total dynamic power is 0.6 W.

Our TianjicX platform with high spatiotemporal elasticity architecture can improve the ability of robots to deal with versatile and multiple intelligent tasks in complex and dynamic environments. Specifically, the elastic resource allocation can improve the hardware utilization and meet different performance requirements of robots. Independent execution contexts enable robots to perform multiple tasks concurrently and asynchronously; the support for interactivity between tasks ensures that multiple modules of the robot (i.e., multimodal sensing and coordination of multimotion modules) can cooperate smoothly toward a common and challenging goal. TianjicX opens a path for the development of computing hardware for mobile robotics to perform intensive and complex tasks locally with low latency and low power.

RESULTS

Rivulet execution model for elastic MITs

The Rivulet execution model is an abstraction layer that decomposes high-level requirements into architecture design guidelines (*31*, *32*). It also summarizes important design features that build a general architectural basis and solution for mobile robots to achieve MITs. It includes three parts: an activity model that builds the basic contexts of MITs on neuromorphic architecture, a resource model that describes the basic conditions of MITs with high efficiency and elasticity, and a cooperation model that clarifies the relationships between tasks.

Activity model

The activity model abstracts the execution of an NN task. Inspired by the concept of the process in the traditional CPU, an activity abstraction can provide a concrete, operable, and describable entity for resource allocation and scheduling in multitasking.

Neuromorphic chips usually adopt a highly parallel architecture with many-core design. For a more general description, a core is called a space unit. The elapsed time for one operation execution or basic schedule window is called a time unit. A time unit at a space unit is called an execution unit. Thus, the neuromorphic architecture can be abstracted as a collection of execution units: $\{e_{s,t} \mid s \in \text{Space}, t \in \text{Time}\}$, where $e_{s,t}$ is the specific execution unit in space s and time t.

An intelligent task (NN-paradigm model) can be completed by the execution of a series of execution units, namely, a rivulet as shown in Fig. 2A. We define a rivulet as a five-tuple: $<E, S, D, S \rightarrow E, D \rightarrow EE>$. E represents the execution units that the rivulet occupies. S is the static data of this rivulet, which usually contains the weight and operation code of the NN. D is the dynamic data of this rivulet, which usually represents the activation values or the intermediate states of the NNs. $S \rightarrow E$ is the mapping from the static data to execution units. The static data will be placed and kept at specific execution units during the execution. $D \rightarrow EE$ is the mapping from the dynamic data to the tuple of source execution units and destination execution units: $<e_{s1,t1}, e_{s2,t2}>$ with $t2 \geq t1$, indicating that the dynamic data is sent from $e_{s1,t1}$ to $e_{s2,t2}$. With rivulet activities, supporting MITs execution means supporting the execution of multi-rivulet as shown in Fig. 2B.

Resource model

NN executions are both computationally and storage intensive. Hence, resource distribution is crucial for rivulet, especially multi-rivulet execution. To meet the flexible and maximized utilization of resources for robotics in various scenarios, our Rivulet model inherits the decentralized philosophy of neuromorphic architecture to increase locality and prevent the potential bottlenecks caused by the centralized resource access. Each space unit has its memory and controller for computation. During the execution of rivulet, the static data are fixed in the corresponding memory and the dynamic data usually flow between adjacent execution units.

In addition, the performance requirements of tasks in different scenarios are also different. Latency, concurrency, and power are three important performance indicators, which put forward high demand for intelligent balance in different cases. We call this "LCP elasticity," which requires independent control of execution units so that the hardware can reach a balance point in the LCP triangle according to various needs brought by complex environments, as shown in Fig. 2C.

Accelerators using time slicing and traditional neuromorphic chips using space slicing are difficult to achieve LCP elasticity. Specifically, LCP elasticity can bring the following three capabilities [$\text{Op}(e_{s,t})$ means the operation of $e_{s,t}$]: (i) space slicing, $\exists s_1, s_2 \in \text{Space}\ \text{Op}(e_{s1,t}) \neq \text{Op}(e_{s2,t})$; (ii) time slicing, $\exists t_1, t_2 \in \text{Time}\ \text{Op}(e_{s,t1}) \neq$

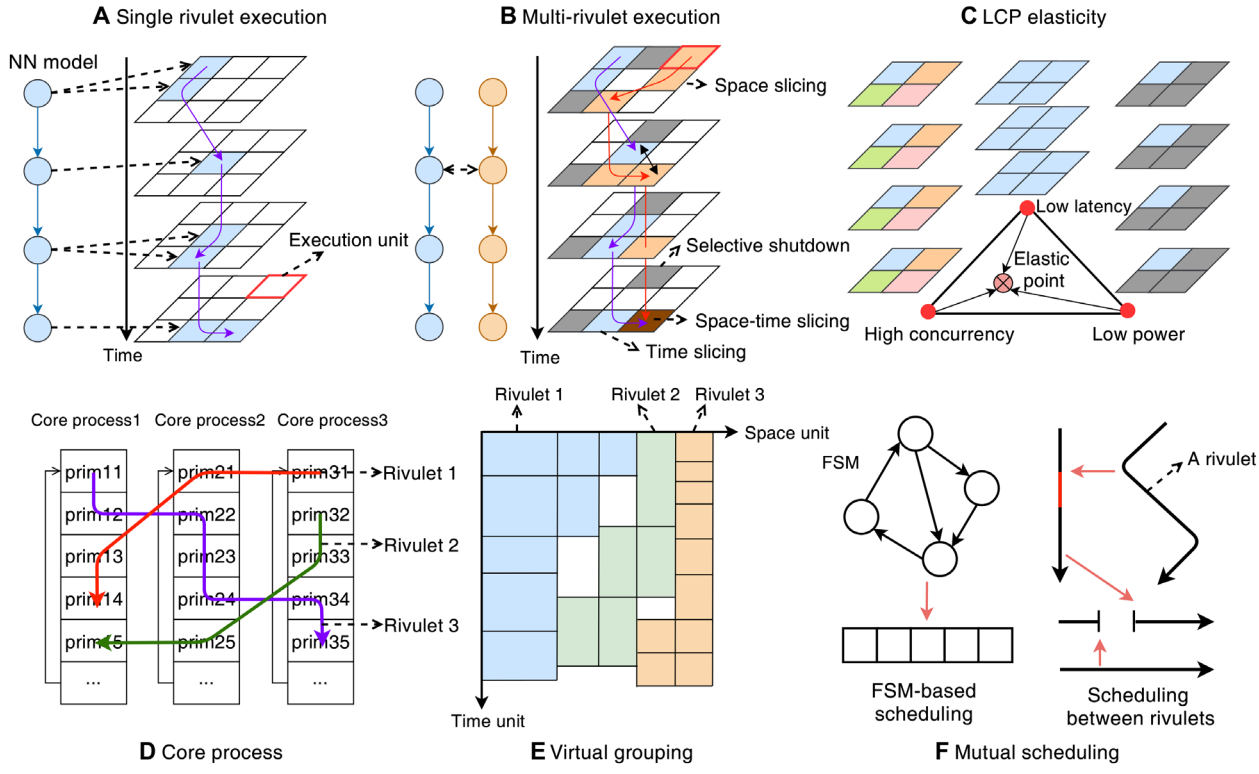

Fig. 2. Illustration of the Rivulet execution model. (**A**) A schematic diagram of the process of deploying NN models on the space and time units of hardware. The deployment generates a rivulet execution flow. (**B**) Multi-rivulet execution that supports time, space, or time-space resource sharing. (**C**) The rivulet execution provides the elasticity between low latency, high concurrency, and low power. The deployment can choose an elastic point in the LCP triangle to adapt to the environment. (**D**) The rivulet is formed by crossing multiple core processes. (**E**) The virtual grouping mechanism. Each rivulet can have its own execution pace and resource context. (**F**) The traditional scheduling method based on FSM and the mutual scheduling between rivulets. The red arrow means a scheduling operation.

$\mathrm{Op}(e_{s,t2})$; and (iii) space-time slicing, for rivulets, R_1, R_2, $E_1 \cap E_2 \neq \phi$. The space-time sharing also opens the possibility for the operation fusion across rivulets (*24*, *33*).

To endow the above capabilities, each space unit executes an independent primitive sequence. The sequence with corresponding resources can be viewed as a process in the traditional computing system. We define this as a core process. A rivulet is formed by crossing multiple core processes, as shown in Fig. 2D.

However, the general program control for each space unit will squeeze the computation and memory area, and increased logic may cause area and power overheads. Thus, we limit the flexibility of the core process to periodically execute a sequence of primitives

$$\forall t_1 \in \mathrm{Time}, \mathrm{Op}(e_{s,t1}) = \mathrm{Op}(e_{s,t1+k\times\mathrm{Period}}), k \in N$$

Cooperation model

In multitasking scenarios, the robot achieves a common goal by completing several tasks collaboratively. Each task requires completing its own work and interacting with others. This indicates that it is essential to maintain sufficient independence of each rivulet execution and provide various interactive flexibilities between rivulets.

Considering that each rivulet has its own execution pace due to different input frequencies of tasks, we adopt synchronized execution within rivulets and asynchronized execution between rivulets. The execution pace means the number of input samples processed by the hardware per unit of processing time. Elimination of inter-rivulet synchronization also allows different rivulets to make better use of the hardware resources. To achieve this, the execution units are divided into synchronized and asynchronized groups based on the rivulet mapping. Synchronous execution is adopted within groups, and asynchronous execution is adopted between groups, which we refer to as virtual grouping, as shown in Fig. 2E.

Interactions between rivulets can be divided into three categories: (i) The transmission between rivulets. One rivulet sends dynamic data to another rivulet. The most common case is that one NN receives the result of another NN. (ii) The modulation between rivulets. One rivulet changes the static data of another rivulet. The modulation is common in neuron systems (*34*). (iii) The mutual scheduling between rivulets. One rivulet controls the execution or resource allocation of another rivulet. For example, a rivulet suspends, starts another rivulet's execution, or issues a rivulet to execute a specific operation. The traditional computing hardware, similar to CPUs, usually adopts a finite-state machine (FSM) to schedule instructions or uses an implicit FSM in the operating system to schedule processes, as shown in Fig. 2F. However, our mutual scheduling explores a decentralized scheduling pattern with naturally concurrent, nonsymbolic, and learnable features.

In summary, the Rivulet execution model abstracts the key features of MITs computing architecture and places the following requirements for computing hardware: the support for efficient multi-rivulet execution; the support for the control of core process;

the support for virtual grouping; and the support for communication, modulation, and mutual scheduling between rivulets.

TianjicX neuromorphic computing chip

A neuromorphic chip, TianjicX, is developed to fulfill the aforementioned requirements and to efficiently implement the Rivulet model. The key challenge of the TianjicX design is to balance the efficiency of cross-paradigm NN computation performed (*35*) by rivulet execution and the flexibility of control, timing, and interaction of multiple rivulets. Here, we adopt a decentralized many-core architecture as the basis and develop a spatiotemporal hierarchical structure, including a core microarchitecture based on NN-paradigm primitives, a hybrid synchronous-asynchronous execution mechanism, and an interactive router that supports asynchronous communication. Compared with conventional neuromorphic chips, TianjicX can make full use of the data locality of intelligent algorithms, improve memory utilization, support various data movement patterns, and enhance programmability.

Microarchitecture of FCore

To support efficient cross-paradigm computing with flexible programmability and scheduling capability, we designed a microarchitecture of the unified functional core (FCore). Most NN layers can be decomposed into three stages: matrix/tensor data rearrangement, linear computation, and nonlinear computation. By extracting the common and special operations of ANNs and SNNs, we design a neuromorphic complete primitive set (*36*) with multiprecision computation to support a broader range of NN models, as shown in table S1. The primitives are further classified into different hardware modules according to the above three computation stages to ensure maximized share of hardware resources.

The FCore is inspired by biological neurons but seeks to extend the function of neurons. The axon is responsible for data arrangement of more general tensor operations instead of transmitting activations or spikes only. Integration is directly done by a dendrite module with 128 multiply accumulators, and our dendrite module also supports linear operations such as element-wise addition and multiplication. Nonlinear computation and memory data management operations are the soma's responsibility. The router not only performs data transfers between FCores but also realizes communication, modulation, and mutual scheduling between different rivulets. The axon, dendrite, soma, and router modules are fully decoupled and scheduled by the controller. To improve the flexibility and adaptability of SNNs, FCore supports the classic leaky integrate-and-firing (LIF) model of SNNs with a configurable integration time window by a separate primitive (named LIF^+) of soma. The LIF^+ primitive also retains various variants to allow more complex SNN models or neuromorphic algorithms (see table S2 for details).

TianjicX adopts a unified memory addressing and access, in which memory resources can be flexibly shared and allocated by multiple execution modules. To reduce memory access operations and simplify the data fetching logic, contiguous regions of memory are accessed during the primitive execution. In particular, data with different functions, such as input data, weight data, etc., are allocated into different data areas and continuously stored in the memory. Therefore, the memory access time overlaps with the calculation time to improve the efficiency. In other words, the core process of FCore in a time unit can be briefly described as the following data flow: reads data from the core memory, performs a linear operation by an axon and a dendrite, performs a nonlinear operation by soma, and writes the final result back to the core memory or sends it to another FCore through a router. (Fig. 3, A and B).

The controller is designed to schedule the execution of core process and enables the independent program execution capability of the FCore. In particular, the controller maintains a primitive sequence and a pointer for current execution. At each time unit, the execution modules can be configured independently on the basis of the current primitives. There are two execution periods, namely, time phase and time step. The time phase is the basic computational period for performing a primitive group. To reduce the complexity of scheduling and the burden of hardware design, the controller executes the preconfigured primitive sequence periodically in storage order. The cyclic execution period is called time step. The independently configurable primitive sequence is the foundation of the elasticity of Rivulet model, which allows FCore to switch tasks through time. For more details of primitive execution, please see fig. S1. In each time step, several modules (axon-dendrite, soma, and router) are parallel or pipelined during the execution. As shown in Fig. 3B, the massive parallel calculations are performed in the dendrite, whereas the data for the dendrite are arranged in the axon at the same time. Similarly, soma and router perform their tasks simultaneously. If there are data dependencies between the modules, they can execute in a pipelined manner. As long as one module outputs a block of data, the next module can perform the corresponding calculation. Such parallel and pipelined hybrid design highly improves the hardware throughput and reduce intermediate storage.

The hierarchical timing mechanism for virtual grouping

Under multi-rivulet execution, asynchronous execution between different rivulets is the key to improving the hardware utilization and independence of each rivulet. Virtual grouping is adopted to achieve the goal. We further treat MITs with different granularities. For example, a rivulet can perform a complete NN task, whereas one layer of NN can also be considered as a fine-grained task. A multilevel trigger timing mechanism is designed to endow TianjicX with a spatiotemporal hierarchical architecture for hardware optimization at different granularities.

We use two configurable trigger groups to control the timing for two-level execution periods and adopt a multilevel arbitrary grouping mechanism. From small to large scale, the spatial hierarchical structures are FCore, phase group, step group, and chip. As shown in Fig. 3C, through configuring each FCore's registers, the FCores can be virtually grouped into multiple phase groups. In the same phase group, the FCores have the same time phase execution cycle. The execution of time phase is asynchronous among different phase groups. Composed of multiple phase groups, step groups have various activation and execution paces. The FCores have the same time step trigger cycle in the same step group, whereas they are asynchronous in different step groups (Fig. 3D). In principle, a single rivulet can be partitioned into multiple phase groups to optimize load balance. For parallel execution of multiple rivulets, FCores are grouped into different step groups to achieve asynchronous execution of multiple tasks and independent optimization. Compared with the global synchronous execution, this can reduce the synchronization overheads between FCores performing different tasks. As shown in Fig. 3D, we also design configurable external trigger signals for each step group to realize an event-driven execution of each rivulet, which can greatly reduce the number of unnecessary computations and improve energy efficiency.

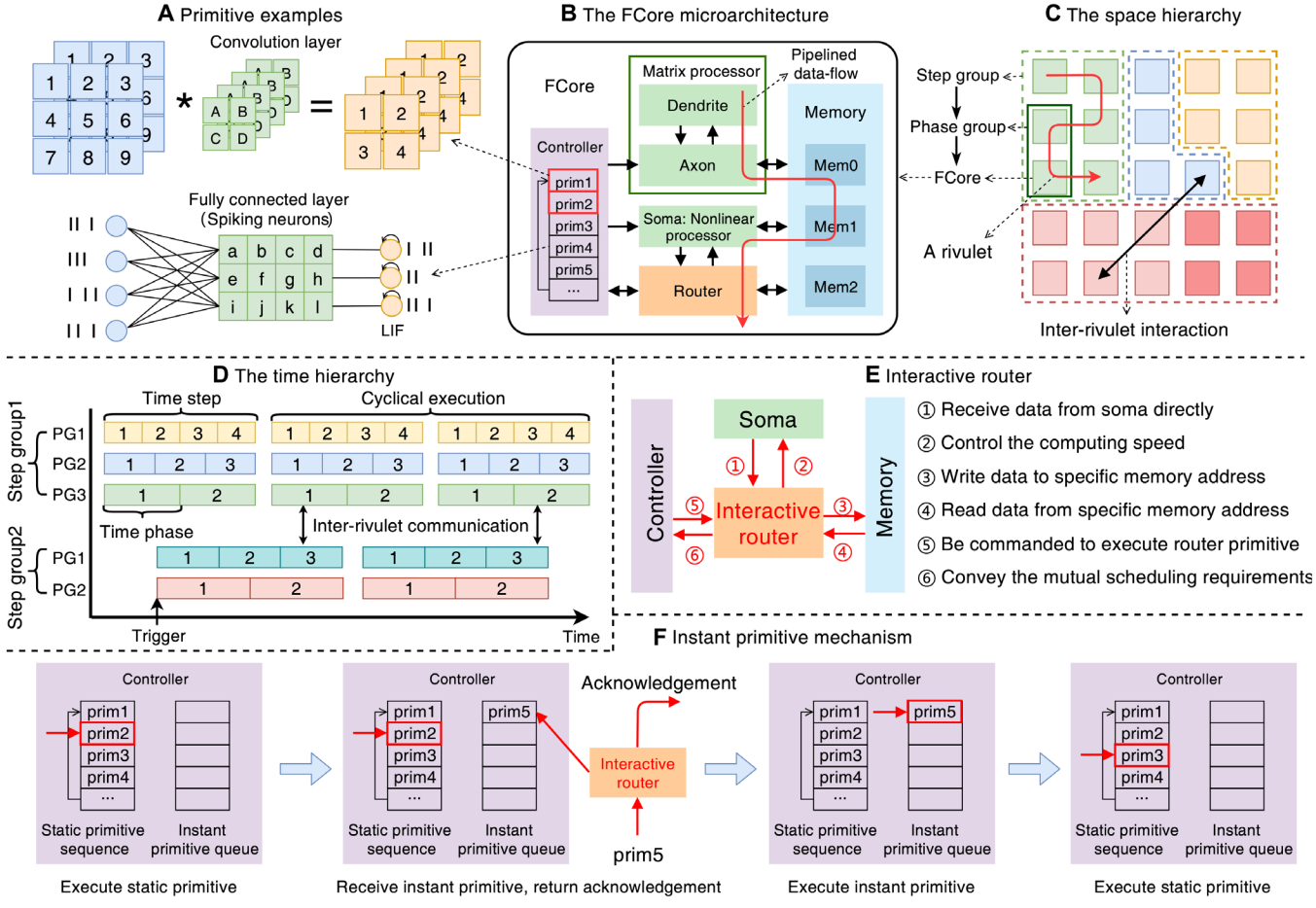

Fig. 3. The hardware architecture of TianjicX. (**A**) Primitive examples and (**B**) diagram of the unified functional core (FCore) in TianjicX. In an FCore, the controller schedules and delivers primitives to execution modules (dendrite-axon, soma, or router). (**C**) The space hierarchy of TianjicX. TianjicX has two levels of space clusters: step group (SG) and phase group (PG). (**D**) The time hierarchy of TianjicX. FCores in the same PG execute in synchronized timing. Different PGs or SGs are in asynchronized timing. (**E**) The interactive router that coordinates with soma, memory, and controller. (**F**) The instant primitive mechanism that supports asynchronous interactions among rivulets through an interactive router.

Interactive router with instant primitive

For the interactions between different rivulets, the data movement infrastructure of TianjicX provides a synchronous routing within a phase group and asynchronous communication among phase groups by a unified router design, called an interactive router as shown in Fig. 3E. It coordinates the state of the core with the information outside the core and reads and writes data from memory for data transmission and modulation. The result of soma can also be sent by the router directly. The router can interfere with the controller to achieve mutual scheduling. Specifically, the instant primitive is used for one rivulet to trigger another rivulet to execute a specific operation at any time. Normally, the core periodically executes a sequence of static primitives. The instant primitives are the execution requirements sent from other rivulets. When the router receives an instant primitive request, it will put the primitive index carried by the "request" into the instant primitive queue and return an instant primitive acknowledgment to the sender. When the instant primitive queue is not empty, the controller will take out the first primitive and execute it instantly. The procedure is illustrated in Fig. 3F.

Compiler stack for multitasking deployment

Although the Rivulet model standardizes hardware execution and adopts inter-rivulet synchronization to facilitate the compilation, there are still a vast number of valid deployment strategies because of independent configurable execution units. This brings challenges to the rapid and automatic generation of appropriate strategies, especially in the case of complex hardware implementations. To solve this challenge, we abstract the hardware details and unify the diversity of tasks at different levels. A compiler stack is designed with three layers (Fig. 4A): transformer, mapper, and code generator.

The transformer unifies various NNs to meet hardware functional constraints, including primitive transformation and quantization. The mapper allocates tasks in space and time and generates rivulet activities. The code generator generates machine codes, including memory address generation, register management, routing optimization, etc. We also design a hierarchical intermediate representation between layers, as shown in Fig. 4B, which decouples software and hardware to build a versatile programming interface.

The mapper is the most important layer, with three design challenges. First, the diversity of tasks makes it difficult to find a unified

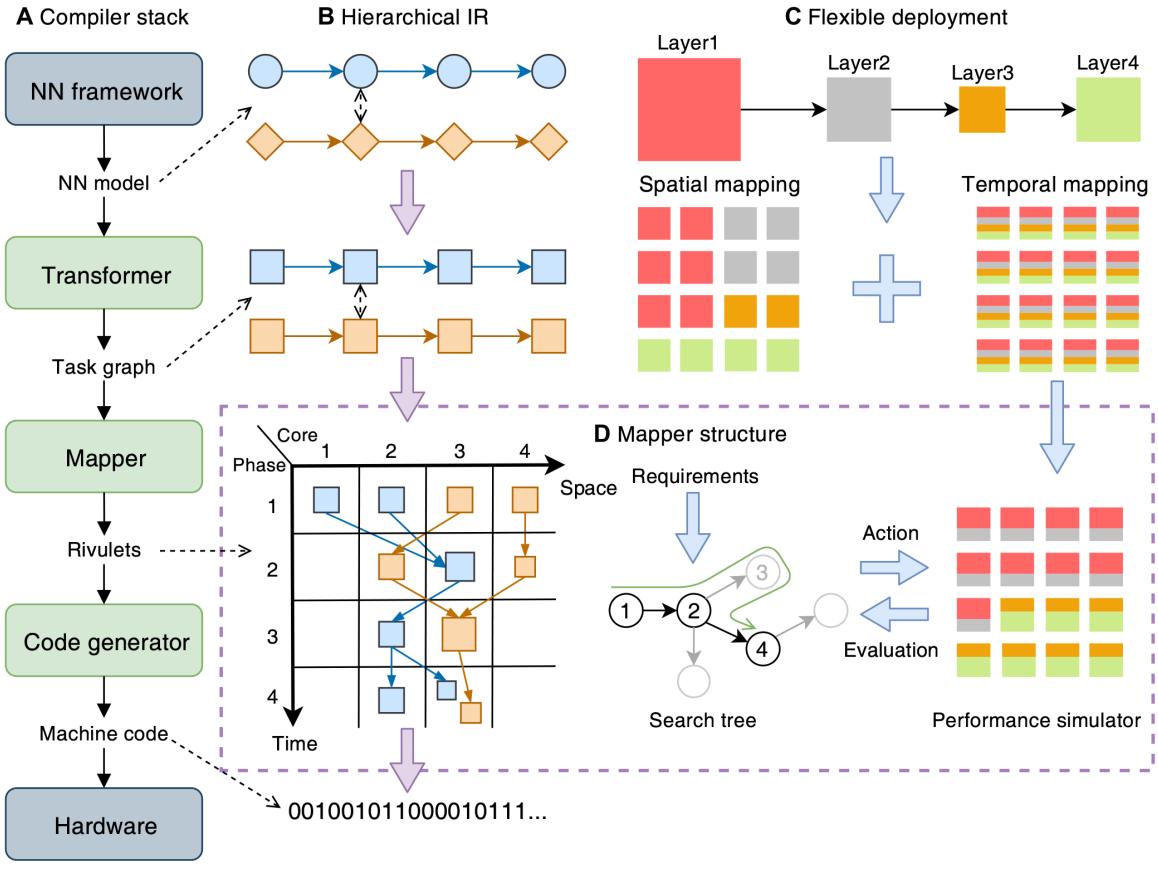

Fig. 4. The compiler stack. (**A**) The compiler stack contains a transformer, mapper, and code generator. (**B**) The hierarchical intermediate representation (IR) for abstracting the hardware and unifying tasks via a software-hardware co-design. (**C**) Demonstration of flexible deployment on TianjicX. Using a four-layer NN as an example, the compiler can combine spatial mapping and temporal mapping. Different from neuromorphic chips that are typical of using spatial mapping and accelerators that usually adopt temporal mapping, our mapper achieves efficient multitasking and elastic deployment by combing both mappings. (**D**) The basic structure of the mapper, including the search tree and the performance simulator.

mapping strategy. Second, the complex hardware design makes execution performance and resource utilization unpredictable. Normally, neuromorphic chips use spatial mapping, and accelerators adopt temporal mapping. In this work, our mapper combines these two mappings together so that it can identify a deployment strategy to largely meet the allocation constraints, achieve load balance, and meet the unique demand of the task as shown in Fig. 4C. It, therefore, endows TianjicX with the flexibility to achieve efficient multitasking and elastic execution.

Furthermore, our mapper combines general search and fast performance evaluation, as illustrated in Fig. 4D. When the performance requirements are input to the mapper, the search part will find a specific mapping strategy step by step. For each step, the search tree will send action directives to the performance simulator to quickly perform the evaluation, thereby realizing the mapping for different environmental demands.

Chip evaluation

TianjicX chip was fabricated in UMC 28-nm High Performance Compact Plus (HPC$^+$) CMOS process and assembled in an FBGA-225 package. Table 1 lists the basic characteristics and parameters. The chip includes 160 FCores and a high-speed serializer/deserializer (SerDes) interface for interchip communication. The physical layout of the chip and the layout and area distribution of each module are shown in Fig. 5 (A to C, respectively). The controller only occupies about 1% of the FCore area but improves the flexibility and efficiency of task execution and interactions markedly. The core memory module is composed of five static random-access memory (SRAM) blocks with a total capacity of 144 kilobytes. Through the high bit-width parallel read-write access interface, the entire chip can have a memory access bandwidth of up to 5.12 terabytes/s at 400 MHz.

We experimentally measure the performance of the TianjicX chip with a focus on power, latency, and throughput. Two mainstream ANN models, MobileNet (*37*) and ResNet50 (*38*), are implemented using different mapping strategies. Figure 5D plots the processing speed against power consumption for TianjicX chip. For comparison, we also plot the reported performance for different types of GPUs, CPU, and deep learning accelerators. From Fig. 5D, we can observe that TianjicX demonstrates the high speed of image processing in a moderate power consumption for MobileNet, indicating that it is competitive in edge applications. TianjicX can also handle ResNet50 with low power, suggesting the great potential for large-scale ANNs. In addition to ANNs, we also demonstrate that TianjicX is able to support SNNs, and hybrid spiking/non-spiking models. The

performance and power consumption for representative SNNs and hybrid models (*39*) are listed in Table 2. Details of the experiments are provided in Materials and Methods. It is notable that the performance can be further improved through optimizing mapping strategies for TianjicX. These results show that TianjicX has a capability to process single NN models in multiple scales and cross-paradigm, which is the key to supporting multitasking of these NN models.

Table 1. The key characteristics and performance of TianjicX chip.

Features	TianjicX	Features	TianjicX
Technology	28 nm	Computing power per unit area	0.2 TOPS/mm^2
Supply voltage	0.82 to 0.95 V	Peak energy efficiency	3.2 TOPS/W @0.84 V
Clock	400 MHz		1.6 TSyOPS/W @0.84 V
Chip area	81.8272 mm^2	Intrachip bandwidth	5.12 terabytes/s
On-chip memory	22.5 megabytes	Interchip bandwidth	64 gigabits/s
Peak throughput	16 TOP/s	Core area	0.411 mm^2
Neurons	160K	Core SRAM	144 kilobytes
Synapses	20M	Core MACs	128

To illustrate the LCP elasticity of TianjicX, a two-layer multi-layer perception (MLP) and a two-layer CNN are used in our experiments. In the MLP experiments, the number of cores used varies from 1 to 8, and the number of MLPs executed on the selected cores varies from 1 to 8. In the CNN experiments, the number of cores varies from 1 to 128, and the number of CNNs varies from 1 to 128. For each type of network, we conduct seven experiments to generate seven points on the LCP triangle as shown in Fig. 5E. Various balanced points among power, concurrency, and latency are achieved by different mapping configurations of TianjicX. Please see Materials and Methods for details of all the experiments.

Tianjicat: A multitask intelligent robot

To demonstrate the capability of TianjicX for MITs applications, a mobile robotic development platform, Tianjicat, is built on the basis of a modified mobile car and equipped with a TianjicX chip array and multimodal sensors as shown in Fig. 6A. The development board consists of four TianjicX chips in a 4 × 1 array. It is worth noting that these chips can be activated individually. In the following experiment, only one TianjicX chip is activated to realize various NNs, whereas the other three chips are deactivated.

We further design a challenging cat-and-mouse game in a complex dynamic environment (Fig. 6B). The Tianjicat plays the role of a cat and tries to catch a randomly running electronic mouse (see movie S1 for the cat-and-mouse game demonstration). Various obstacles are randomly and dynamically placed in different locations.

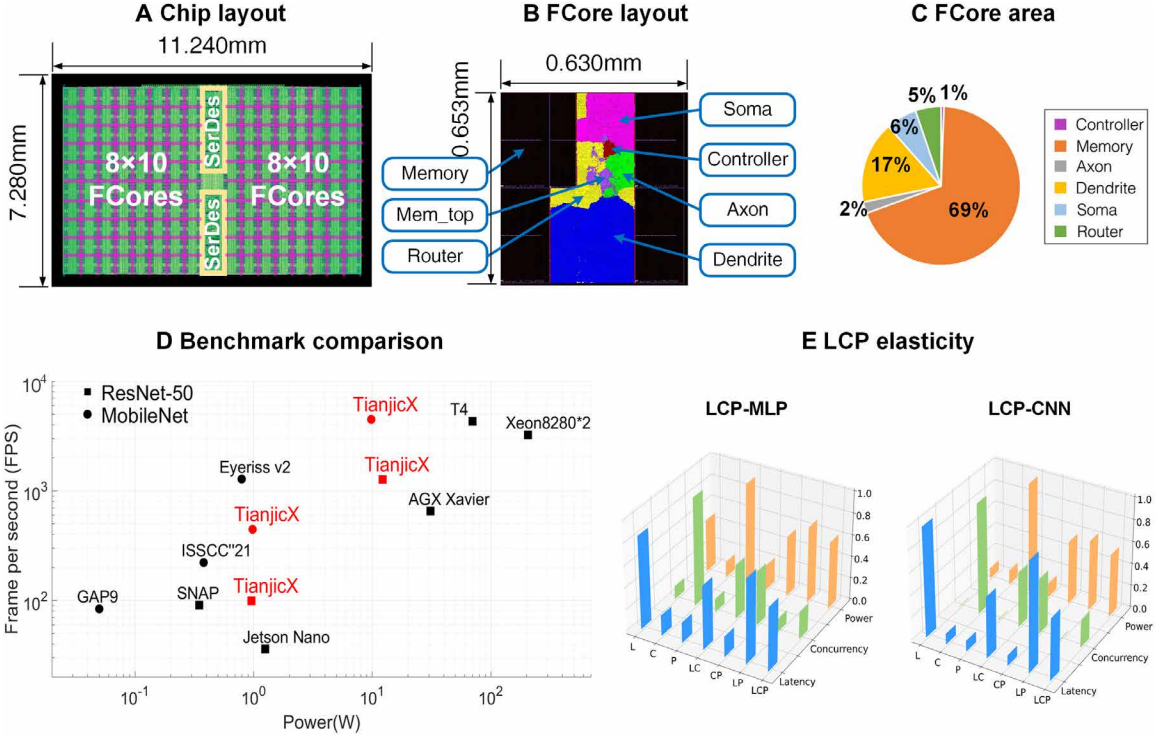

Fig. 5. Summary of chip evaluation. (**A**) Chip layout. Each TianjicX chip has 160 FCores arranged in a 16 × 10 two-dimensional mesh array. (**B**) FCore layout, (**C**) FCore core area partition. (**D**) State-of-the-art measured MobileNet (*37*) and ResNet50 (*38*) performance versus power consumption. Measurements of this paper show two scenarios (low power and high performance) for each NN. Also shown are GPUs, a CPU, and NN accelerators (*20*, *43*, *44*). (**E**) LCP elasticity evaluation based on CNN and MLP. On the *x* axis, "L," "C," "P," "LC," "CP," "LP," and "LCP" represent seven balanced points achieved in our LCP experiments, which are explained in Materials and Methods. The *y* axis signifies three requirements in LCP elasticity. The *z* axis shows the normalized values of latency, power, and concurrency, where a greater value always means a better indicator, i.e., lower latency, higher concurrency, or lower power.

Table 2. The performance of TianjicX chip on different SNNs and hybrid models.

Model		Benchmark NNs	Performance (FPS)		Power (W)		Power efficiency (FPS/W)	
			2080Ti	TianjicX	2080Ti	TianjicX	2080Ti	TianjicX
SNN		NMNIST-MLP	41,891.1	16,056.5	102.6	1.4185	408.3	11,319.4
		MNIST-MLP	86,899.4	13,260.8	147.3	0.5296	589.9	25,039
		LeNet	4,460.6	1,962.8	90.9	0.1353	49.1	14,509.2
Hybrid	ANN to SNN	LeNet encoding	35,783.2	1,962.8	126.9	0.1354	282.1	14,492.9
		LeNet sampling	23,013.3	1,962.8	125.5	0.1404	183.5	13,982.8
	SNN to ANN	LeNet accumulate	35,013	1,957.1	123.5	0.1523	283.5	12,852.4
		LeNet expand	23,157.9	1,962.8	122.5	0.1527	189	12,852.4

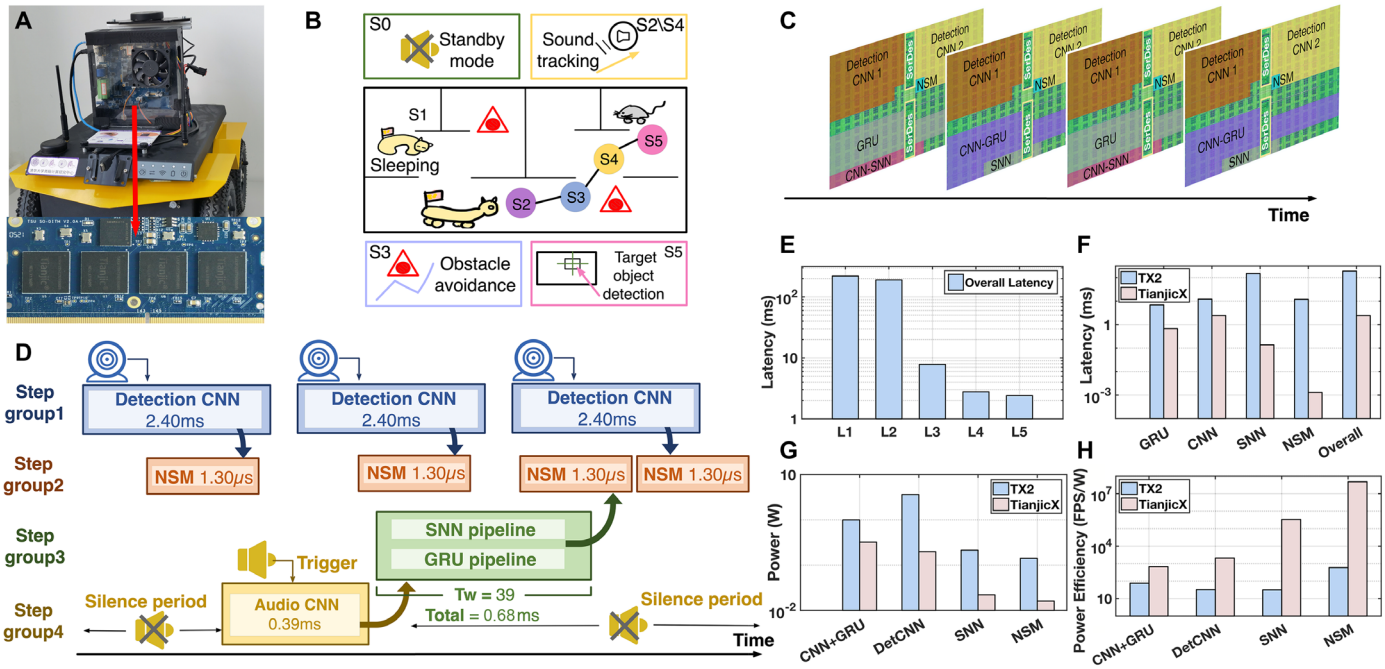

Fig. 6. The overall robot system and design for Tianjicat. (**A**) The physical picture of the Tianjicat robot, the development board, and the TianjicX chip array. The development board can control the enabling of one or more chips as required. (**B**) The schematic diagram of the robot scene. The Tianjicat has five main states: sleeping, or standby mode sound source localization, obstacle avoidance, and object detection. The state transition is determined by NSM (*41*). (**C**) The dynamic space slicing on the TianjicX. All these networks are mapped with flexible resource sharing on one TianjicX chip. (**D**) Diagram of external event–driven and asynchronous parallel execution of multitasking on TianjicX. (**E**) Measurement of overall latency under five scenarios. L1 and L2 represent TX2 with and without loading data, respectively. L3 and L4 represent TianjicX loading dynamic data under 4 and 16 gigabits/s, respectively. L5 represents TianjicX without loading data. (**F**) The comparison of latency of each NN model between TianjicX and TX2. (**G**) The comparison of power consumption of each NN model between TianjicX and TX2. (**H**) The comparison of power efficiency of each NN model between TianjicX and TX2.

The Tianjicat needs to track the mouse through visual recognition, sound tracking, or a combination of the two, then move toward the mouse without colliding with obstacles, and finally catch up with the mouse. In this process, a variety of NN algorithms are implemented to accomplish voice recognition, sound source localization, objection detection, obstacle avoidance, and decision-making in real-time scenes (see fig. S2 for details). Hence, it is key for these NN algorithms to be processed cooperatively and concurrently. In this work, lightweight detection CNNs are used for end-to-end multi-object detection. An SNN acts as an event-driven switch for the following sound processing NNs. A CNN-GRU (gated recurrent unit) hybrid network (*40*) is used to estimate the location of a sound source, and an SNN-based neural state machine (NSM) (*41*) is used for multinetwork scheduling and policy decisions. SNNs can memorize temporal information via intrinsic neuronal dynamics and encode the information into binary spike trains. The threshold-based mechanism of SNNs is also naturally similar to a switch that determines whether the sound localization network should be activated. Hence, an SNN model is selected as the switch NN in the tasks. The SNN and GRU use the same CNN-based feature (called audio-CNN) extractor for voice preprocessing. The details about the networks and their performance are provided in table S4 and Materials and Methods.

The algorithms are deployed on a single TianjicX chip asynchronously and in parallel. According to the computing performance

requirements of different networks, the limited hardware resource of one TianjicX chip is optimally allocated through a hybrid spatiotemporal mapping method. The resource occupation of these multiple tasks is shown in Fig. 6C. The whole system occupies 128 FCores, which are allocated to four step groups and executed in an event-driven manner as shown in Fig. 6D. In particular, the features extracted by the audio-CNN are shared by the GRU and the SNN. Therefore, through the space and time slicing of rivulets, these two networks can reuse the same cluster of FCores (four in total). Compared with two independent clusters of FCores executed without optimization, storage consumption is reduced by 8.2% without an increase in processing time.

Figure 6 (E and F) compares the response delays of multiple tasks performed on the TianjicX and NVIDIA Jetson TX2 platforms. TianjicX shows that the processing speed of the CNN and GRU tasks is increased by 5.06× and 10.2×, respectively, and the processing of SNN tasks is increased by about a thousand times. On the whole, the computation latency of TianjicX is reduced by 79.09 times compared with the TX2. Compared with the serial execution of multiple NNs in the GPU, our entire system can flexibly work in an event-driven way. Each task is executed in response only when its input is updated. In addition, the effect of multitasking with efficient asynchronous parallel execution and interactive features is negligible on the response of individual task to its input. The power consumption and power efficiency of each NN model are shown in Fig. 6 (G and H, respectively). Furthermore, when five NNs are processed simultaneously, the dynamic power consumption of the chip is about 0.6 W. Compared with the TX2, the dynamic power of TianjicX is reduced by 50.66%. Collectively, these results demonstrate that TianjicX has a good real-time performance with low latency and high energy efficiency for MITs.

DISCUSSION

In conclusion, this paper provides a systematic scheme to achieve elastic and parallel execution of multiple NNs with low latency and low power for mobile intelligent robots. The Rivulet execution model is first developed to solve the key conflict among efficiency, elasticity, and adaptability through a configurable primitive sequence and a hybrid synchronous-asynchronous execution mechanism. Then, a neuromorphic chip, TianjicX, is developed with a non–von Neumann decentralized architecture that integrates many cross-paradigm cores and massive-parallel computing units to meet the requirements of Rivulet operation. Equipped with TianjicX and multimodal sensors, our Tianjicat robot is able to process multisensory information at the same time and coordinate multiple tasks to achieve a common and challenging goal.

It is believed that rapid development of mobile robots brings opportunities to design alternative computing hardware based on their unique requirements. The neuromorphic architecture not only can be used to improve the level of intelligence but also can provide ideas toward alternative computing architecture design methodologies. We summarize these possible ideas as follows.

First, to provide collocated resources with decentralized distribution. From the collocated memory and computation to collocated all the resources, including the controlling, routing, etc., decentralized resources can further avoid the bottleneck caused by centralized access to a specific resource by increasing locality.

Second, to adopt the event-driven execution and scheduling. Event-driven mechanism helps improve the performance of scheduling optimization, hardware utilization, and power consumption.

Third, to achieve approximate computing through NN-like activities. Approximate computing helps the system break through the logic barrier, manage complexity spontaneously like NN models, reinforce robustness and provide the ability to coevolve with the environment.

Fourth, to use specialized hardware architecture to realize the general-purpose system. The execution of the specialized NN paradigm promises efficiency, and the compiler converts various procedures into the NN paradigm, making the system closer to general purpose.

The Rivulet execution model and the TianjicX chip implementation are designed on the basis of these ideas. It should be emphasized that in designing the TianjicX computing hardware, several trade-offs are taken to achieve high spatiotemporal elasticity in task execution, resource allocation, and task cooperation. For example, we build a basic primitive set that can support the cross-paradigm computing. However, there are many ANN and SNN models, and each has different operations and structures. It is unlikely to cover every operation and structure. Hence, we pay more attention to the commonly adopted algorithms, especially those widely applied in mobile robots. There is also a trade-off between high flexibility and high efficiency. To support spatiotemporal elasticity, a microcontroller is specially designed for the FCore to support independent configurable primitive sequences without too much design burden. In the future, we will continue to study the combination of neuromorphic hardware and robot's computing and to explore more possibilities for unmanned robots.

MATERIALS AND METHODS

Performance analysis of the TianjicX chip

To evaluate the performance of chips, we choose different types of NNs as a benchmark for tests, and all of the network structures are shown in table S3. For SNN (*42*), we use two fully connected networks. The input data's spatial resolution is $34 \times 34 = 1156$ with two channels and $28 \times 28 = 784$ with one channel. For ANN models, we use CNNs such as ResNet50 and MobileNet. In addition, TianjicX is also a hardware platform for the end-to-end implementation of hybrid spiking and nonspiking models, so we also decide to test multiple types of hybrid LeNet. We adopt the same NN structures proposed in the previous paper (*39*) and evaluate the four introduced signal conversion methods in the hybrid models.

We choose NVIDIA 2080Ti GPU and other neuromorphic/deep learning accelerators for comparison. To get the GPU power consumption benchmarks, we write a tool based on the API provided by NVIDIA to monitor the power of the GPU. For all of these models, we set the batch size to 1 to simulate the real-time deployment situation on TianjicX. Because GPUs are more efficient to deal with large-scale NN models, we test the same models with the maximum batch size for fair comparisons. First, we record the static power (maybe consumed by some other processes or hardware requirements such as fans) for a long time, and after obtaining the static power, we record the runtime power and calculate the average value. The performance of different kinds of networks is compared in Table 2 and Fig. 5D. It shows that TianjicX can efficiently support ANN, SNN, and hybrid spiking and nonspiking models.

LCP experiment

The MLP has the structure of 16-2048-32 neurons. For the two-layer convolutional network with 1×1 kernels, the shape (channels,

height, and width) of its input feature map is (16, 32, and 32) and the channel numbers are 16-32-32. For each type of network, we conduct seven experiments, producing seven points on the LCP triangle, three of which are the vertex of LCP triangle, three of which are on the edges, and the last one is an elastic point inside the triangle. At the P-point, where only one core is used to execute one task in our experiments, the hardware runs at the most power-efficient mode, which will cause the lowest concurrency and longest latency. At the L-point, all possible cores are used to execute as few tasks as possible to maximize the computation speed, which causes the lowest concurrency and the highest power. At the C-point, all possible cores are used to execute as many tasks as possible to maximize the concurrency which will cause the highest power consumption and the highest latency. For the points on an edge of the LCP triangle, the hardware execution will balance two of three LCP features and ignore the other one. For example, for a point on the LC edge, all possible cores will be used, which leads to maximal power consumption, but we try to achieve a balance point between high concurrency and low latency. A total of 128 cores are used to execute 64 CNNs in our experiments, which tolerate a lower concurrency while acquiring a lower latency, compared with the experiment of C-point where 128 cores are used to execute 128 CNNs. If the hardware execution point is inside the LCP triangle, then moderate power, concurrency, and latency are adopted.

Robotic system setup

The embedded computing platform of unmanned ground vehicle includes TianjicX chips and an Intel Arria 10 FPGA for acquiring data from external sensors. A RealSense RGB-D camera D435 and a ReSpeaker USB Mic Array are used for sensing audio and video information from environments. FPGA collects and preprocesses these data via an internet interface with socket communication and then transfers them to the TianjicX chip through a high-speed SerDes interface. TianjicX is integrated as a main computing hardware for processing the multitasking NN algorithms in a single chip through spatial and temporal mapping. After TianjicX completes the computation, it will control the peripheral equipment to make relevant feedback.

Algorithms for multiple tasks

Dataset

A multimodal dataset is collected for this task, including four-channel mouse voice samples recorded by a microphone array with direction angles labeled by indoor location system and RGB images with the obstacles and target mice labeled manually. The image data include five kinds of obstacles and three kinds of movable objects, and only the mouse is the target. The resolution of images is 480 × 360, and we cut five 360 × 360 patches from one training sample for data augmentation. For the audio data, we deal with the wave file with a fast Fourier transformation and handle processed data with the audio-CNN.

For target detection

We design a lightweight detection-CNN with 0.43M parameters. To increase the robustness of the algorithm, we add a voting mechanism when applying the detection-CNN in Tianjicat, where we use the 360 × 360 crop results at four corners and the middle of the original images, resize the five patches to 256 × 256 as the inputs of detection CNN, and then vote for final prediction. This method makes up for the deficiency of the visual field and reduces the influence of false detection results. To improve the detection performance of objects, we use different sets of parameters for obstacle detection and mouse detection, in which the mice are always much smaller than the obstacles in the environment.

For sound source localization and sound discrimination

We learn the local shift-invariant features in the spectrogram of the audio samples using a convolutional layer followed by a maximum pooling layer (audio-CNN). The GRU with 128 hidden units and the fully connected layer with two neurons use these features to predict the direction of the sound source. The features are also used for sound discrimination by an SNN acting as a voice wake-up function. The spiking signals are propagated layer by layer through neuronal dynamics and determine the output by counting the spiking firing rate in the output layer.

For the multiple network coordination and high-level decision-making

We develop a learnable NSM based on the model in (*41*). The state machine is composed of five states and four trigger signals. The state translation priority is considered to deal with complex decision-making. The state machine yields the following equation

$$S[k+1] = \mathrm{step}(W_{ss}S[k] + W_{ts}(W_{tt}T[k+1] \cdot W_{st}[k]s[k])) \quad (1)$$

where $s[k]$ is the one-hot vector to represent the state at step k and T[k] is a binary vector to represent the trigger signals. These trigger signals are obtained through voice wake-up SNN and threshold judgment networks. In particular, we set W_{ss}, W_{ts}, W_{tt}, and W_{st} as adjustable weights to realize desired translation strategy and provide better generalization ability. The NSM contains 100 hidden neurons, is trained in an end-to-end manner with an L1 loss function, and optimized by a stochastic gradient descent algorithm. Table S4 shows the structures and parameters of the networks used in Tianjicat demo.

Accuracy

We train those NNs in the GPU platform using FP32 (32-bit floating-point) precision and then apply the quantization-aware training to get the integer counterparts. We consider that the absolute angle error within 30° is a correct prediction for sound source localization. The FP32 CNN-GRU model provides 83.3% accuracy, and the CNN-SNN model obtains 100% sound discrimination accuracy, whereas the quantized model (which will run on TianjicX) still provides 79.36% direction accuracy and 99.78% sound discrimination accuracy. For the detection tasks, we use the F1 score for validation, which is the harmonic mean of precision and recall of the detection model and is a critical indicator in detection applications. The obstacle detection network achieves F1 = 0.988 in FP32 precision. After being quantized, the F1 score is 0.924. The mouse detection CNN has F1 score of 0.933 with a precision equal to 96.55%. The precision of the target detection is a more critical indicator in this task, which determines the safety of the robot and the smoothness of control.

SUPPLEMENTARY MATERIALS

www.science.org/doi/10.1126/scirobotics.abk2948
Supplementary Text
Figs. S1 and S2
Tables S1 to S4
Movie S1
References (*45*, *46*)

View/request a protocol for this paper from *Bio-protocol*.

REFERENCES AND NOTES

1. D. Pena, A. Forembski, X. Xu, D. Moloney, Benchmarking of CNNs for low-cost, low-power robotics applic ations, in *Proceedings of the RSS 2017 Workshop: New Frontier for Deep Learning in Robotics* (RSS, 2017), pp. 1–5.
2. N. K. N. Aznan, J. D. Connolly, N. Al Moubayed, T. P. Breckon, Using variable natural environment brain-computer interface stimuli for real-time humanoid robot navigation, in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 4889–4895.
3. J. Li, Y. Fu, S. Dong, Z. Yu, T. Huang, Y. Tian, Asynchronous Spatiotemporal Spike Metric for Event Cameras, in *Proceedings of the IEEE Transactions on Neural Networks and Learning Systems* (IEEE, 2021), pp. 1–12.
4. Z. Yang, Y. Wu, G. Wang, Y. Yang, G. Li, L. Deng, J. Zhu, L. Shi, DashNet: A hybrid artificial and spiking neural network for high-speed object tracking. arxiv:1909.12942 [cs.CV] (15 September 2019).
5. G. Orchard, X. Lagorce, C. Posch, S. B. Furber, R. Benosman, F. Galluppi, Real-time event-driven spiking neural network object recognition on the SpiNNaker platform, in *Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2015), pp. 2413–2416.
6. T. Taunyazov, W. Sng, H. H. See, B. Lim, J. Kuan, A. F. Ansari, B. C. Tee, H. Soh, Event-driven visual-tactile sensing and learning for robots. arxiv:2009.07083 [cs.RO] (15 September 2020).
7. M. Li, H. Ruan, Y. Qi, T. Guo, P. Wang, G. Pan, Odor recognition with a spiking neural network for bioelectronic nose. *Sensors* **19**, 993 (2019).
8. J. Wang, A framework for the integration of the Emotiv EEG System into the NeuCube spiking neural network environment for robotics control (Auckland University of Technology, 2014); www.semanticscholar.org/paper/A-framework-for-the-integration-of-the-Emotiv-EEG-Wang/9a2f165e28bf688d21f7076cd511275b9e7e0512.
9. M. Inoue, T. Yamashita, T. Nishida, Robot path planning by LSTM network under changing environment, in *Advances in Computer Communication and Computational Sciences* (Springer, 2019), pp. 317–329.
10. W. Huang, J. Zhong, A. Cangelosi, Multiple Timescale and Gated Mechanisms for Action and Language Learning in Robotics, in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2020), pp. 1–7.
11. J.-F. Tremblay, T. Manderson, A. Noca, G. Dudek, D. Meger, Multimodal dynamics modeling for off-road autonomous vehicles. arxiv:2011.11751 [cs.RO] (23 November 2020).
12. H.-k. Chiu, J. Li, R. Ambrus, J. Bohg, Probabilistic 3D Multi-Modal, Multi-Object Tracking for Autonomous Driving. arxiv:2012.13755 [cs.CV] (26 December 2020).
13. Z. Yang, Y. Zhang, J. Yu, J. Cai, J. Luo, End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions, in *Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR)* (IEEE, 2018), pp. 2289–2294.
14. V. R. Kumar, S. Yogamani, H. Rashed, G. Sitsu, C. Witt, I. Leang, S. Milz, P. Mäder, Omnidet: Surround view cameras based multi-task visual perception network for autonomous driving. *IEEE Robot. Autom. Lett.* **6**, 2830–2837 (2021).
15. D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, K. Hausman, MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale. arxiv:2104.08212 [cs.ro] (16 April 2021).
16. J. Xu, Q. Liu, H. Guo, A. Kageza, S. AlQarni, S. Wu, Shared multi-task imitation learning for indoor self-navigation, in *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)* (IEEE, 2018), pp. 1–7.
17. M. Islam, V. Vibashan, H. Ren, Ap-mtl: Attention pruned multi-task learning model for real-time instrument detection and segmentation in robot-assisted surgery, in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020), pp. 8433–8439.
18. W. Zhang, Q. Chen, N. Zheng, W. Cui, K. Fu, M. Guo, Toward QoS-awareness and improved utilization of spatial multitasking GPUs. *IEEE Trans. Comput.* **71**, 866–879 (2021).
19. J. Kim, J. Kim, Y. Park, Navigator: dynamic multi-kernel scheduling to improve GPU performance, in *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2020), pp. 1–6.
20. Y.-H. Chen, T.-J. Yang, J. Emer, V. Sze, Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE J. Emerg. Select. Topics Circuits Syst.* **9**, 292–308 (2019).
21. B. Moons, M. Verhelst, A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets, in *Proceedings of the 2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)* (IEEE, 2016), pp. 1–2.
22. A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, W. J. Dally, SCNN: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Comput. Architect. News* **45**, 27–40 (2017).
23. Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury, S. Liu, A survey of fpga-based robotic computing. *IEEE Circuits Syst. Mag.* **21**, 48–74 (2021).
24. J. Lee, J. Choi, J. Kim, J. Lee, Y. Kim, Dataflow Mirroring: Architectural Support for Highly Efficient Fine-Grained Spatial Multitasking on Systolic-Array NPUs, in *Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2021).
25. E. Baek, D. Kwon, J. Kim, A multi-neural network acceleration architecture, in *Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (IEEE, 2020), pp. 940–953.
26. S. Ghodrati, B. H. Ahn, J. K. Kim, S. Kinzer, B. R. Yatham, N. Alla, H. Sharma, M. Alian, E. Ebrahimi, N. S. Kim, Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks, in *Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2020), pp. 681–697.
27. M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
28. D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, G. Pan, Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *J. Syst. Architect.* **77**, 43–51 (2017).
29. B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, K. Boahen, Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* **102**, 699–716 (2014).
30. P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
31. J. B. Dennis, A parallel program execution model supporting modular software construction, in *Proceedings of the Third Working Conference on Massively Parallel Programming Models (Cat. No. 97TB100228)* (IEEE, 1997), pp. 50–60.
32. P. Qu, J. Yan, Y. H. Zhang, G. R. Gao, Parallel turing machine, a proposal. *J. Comput. Sci. Technol.* **32**, 269–285 (2017).
33. G. Shomron, U. Weiser, Non-blocking simultaneous multithreading: Embracing the resiliency of deep neural networks, in *Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (IEEE, 2020), pp. 256–269.
34. K. Zipser, V. A. Lamme, P. H. Schiller, Contextual modulation in primary visual cortex. *J. Neurosci.* **16**, 7376–7389 (1996).
35. J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* **572**, 106–111 (2019).
36. Y. Zhang, P. Qu, Y. Ji, W. Zhang, G. Gao, G. Wang, S. Song, G. Li, W. Chen, W. Zheng, A system hierarchy for brain-inspired computing. *Nature* **586**, 378–384 (2020).
37. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications. arxiv:1704.04861 [cs.CV] (17 April 2017).
38. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.
39. G. Wang, S. Ma, Y. Wu, J. Pei, R. Zhao, L. Shi, End-to-end implementation of various hybrid neural networks on a cross-paradigm neuromorphic chip. *Front. Neurosci.* **15**, 615279 (2021).
40. R. Dey, F. M. Salem, Gate-variants of gated recurrent unit (GRU) neural networks, in *Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (IEEE, 2017), pp. 1597–1600.
41. L. Tian, Z. Wu, S. Wu, L. Shi, Hybrid neural state machine for neural network. *Sci. China Inform. Sci.* **64**, 132202 (2021).
42. Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, L. Shi, Direct training for spiking neural networks: Faster, larger, better, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 1311–1318.
43. R. Eki, S. Yamada, H. Ozawa, H. Kai, K. Okuike, H. Gowtham, H. Nakanishi, E. Almog, Y. Livne, G. Yuval, 9.6 A 1/2.3 inch 12.3 Mpixel with on-chip 4.97 TOPS/W CNN processor back-illuminated stacked CMOS image sensor, in *Proceedings of the 2021 IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, 2021), vol. 64, pp. 154–156.
44. J.-F. Zhang, C.-E. Lee, C. Liu, Y. S. Shao, S. W. Keckler, Z. Zhang, Snap: An efficient sparse neural acceleration processor for unstructured sparse deep neural network inference. *IEEE J. Solid State Circuits* **56**, 636–647 (2021).
45. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in *Proceedings of the 2009 IEEE Conference On Computer Vision and Pattern Recognition* (IEEE, 2009), pp. 248–255.
46. Y. LeCun, The MNIST database of handwritten digits; http://yann.lecun.com/exdb/mnist/.

Funding: This work was partly supported by the National Key Research and Development Program of China (grant no. 2021ZD0200300), National Nature Science Foundation of China (nos. 62088102 and 61836004), National Key R&D Program of China 2018YFE0200200, CETC Haikang Group–Brain Inspired Computing Joint Research Center, IDG/McGovern Institute for Brain Research at Tsinghua University, Beijing Advanced Innovation Center for Integrated Circuits, and Beijing Mlink Technology Inc. **Author contributions:** S.M., J.P., W. Zhang, and G.W. were in charge of the multitasking neuromorphic computing platform, the principles of chip

design, the execution model and software toolchain, and chip implementation and testing, respectively. S.M. and W. Zhang proposed the computing platform and execution model. J.P., G.W., S.M., C.S., C.M., and M.L. carried out the chip development. W. Zhang, S.M., C.S., and H.Q. developed the software stack. D.F., F.Y., R.Z., J.S, H.L, S.M., X.L., and G.W. worked on the multitasking robot experiment. D.F., F.L., W. Zhou, Y.W., Y.L and G.L. developed the algorithm. S.M., R.Z., W. Zhang, G.W., D.F., F.L., H.L, T.W., J.P., and L.S. contributed to the analysis and interpretation of results. All of the authors contributed to discussion of chip and robot design and experiment. S.M., R.Z., W. Zhang, G.W., D.F., and L.S. wrote the manuscript with input from all authors. L.S. proposed the concept of MITs architecture and supervised the whole project. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data are available in the main text or the Supplementary Materials.

Submitted 5 July 2021
Accepted 22 May 2022
Published 15 June 2022
10.1126/scirobotics.abk2948

Science Robotics

Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots

Songchen Ma, Jing Pei, Weihao Zhang, Guanrui Wang, Dahu Feng, Fangwen Yu, Chenhang Song, Huanyu Qu, Cheng Ma, Mingsheng Lu, Faqiang Liu, Wenhao Zhou, Yujie Wu, Yihan Lin, Hongyi Li, Taoyi Wang, Jiuru Song, Xue Liu, Guoqi Li, Rong Zhao, and Luping Shi

Sci. Robot. **7** (67), eabk2948. DOI: 10.1126/scirobotics.abk2948

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.