

# Bendable non-silicon RISC-V microprocessor

<https://doi.org/10.1038/s41586-024-07976-y>

Received: 3 April 2024

Accepted: 21 August 2024

Published online: 25 September 2024

Open access

 Check for updates

Emre Ozer<sup>1,4</sup>✉, Jędrzej Kufel<sup>1,4</sup>, Shvetank Prakash<sup>1,2,4</sup>, Alireza Raisiardali<sup>1</sup>, Olof Kindgren<sup>3</sup>, Ronald Wong<sup>1</sup>, Nelson Ng<sup>1</sup>, Damien Jausseran<sup>1</sup>, Feras Alkhalil<sup>1</sup>, David Kong<sup>2</sup>, Gage Hills<sup>2</sup>, Richard Price<sup>1</sup> & Vijay Janapa Reddi<sup>2</sup>

Semiconductors have already had a very profound effect on society, accelerating scientific research and driving greater connectivity. Future semiconductor hardware will open up new possibilities in quantum computing, artificial intelligence and edge computing, for applications such as cybersecurity and personalized healthcare. By nature of its ethos, open hardware provides opportunities for even greater collaboration and innovations across education, academic research and industry. Here we present Flex-RV, a 32-bit microprocessor based on an open RISC-V (ref. 1) instruction set fabricated with indium gallium zinc oxide thin-film transistors<sup>2</sup> on a flexible polyimide substrate, enabling an ultralow-cost bendable microprocessor. Flex-RV also integrates a programmable machine learning (ML) hardware accelerator inside the microprocessor and demonstrates new instructions to extend the RISC-V instruction set to run ML workloads. It is implemented, fabricated and demonstrated to operate at 60 kHz consuming less than 6 mW power. Its functionality when assembled onto a flexible printed circuit board is validated while executing programs under flat and tight bending conditions, achieving no worse than 4.3% performance variation on average. Flex-RV pioneers an era of sub-dollar open standard non-silicon 32-bit microprocessors and will democratize access to computing and unlock emerging applications in wearables, healthcare devices and smart packaging.

We present Flex-RV, a 32-bit RISC-V microprocessor based on an open-source 32-bit RISC-V central processing unit (CPU) extended with machine learning (ML) features fabricated with indium gallium zinc oxide (IGZO) thin-film transistors (TFTs), enabling an ultralow-cost and conformable microprocessor for emerging applications.

Emerging applications such as fast-moving consumer goods (for example, smart labels and packaging), healthcare wearables (for example, smart patches and dressings), single-use healthcare implantables (for example, neural interfaces), and single-use healthcare test strips (for example, lateral flow tests, microfluidics) have not been embedded with microprocessors mainly because of cost and form factor. Cost is the determining factor in enabling these applications, in particular, in smart labels and packaging for goods. Moreover, form factor, in the context of physical flexibility and bendability, is important for healthcare wearables and implantables. The computational requirements of these emerging applications are not intensive in terms of speed and communication bandwidth. Data sampling rates required by the sensors in these applications are no higher than 200 Hz (ref. 3); in some cases, sampling rates can be as high as 1 kHz, so microprocessors operating at low clock frequencies (for example, <100 kHz) can meet the computational requirements of the applications.

Silicon (Si) has been the underlying semiconductor technology to develop microprocessors for over 50 years improving performance, area, power consumption and cost in its evolutionary path, mainly driven by Moore's law. Over this period, microprocessors have gone faster from kHz to GHz because the transistor geometry shrunk from a few  $\mu$ m to 2–3 nm today, and the unit cost up to a few dollars for low-end

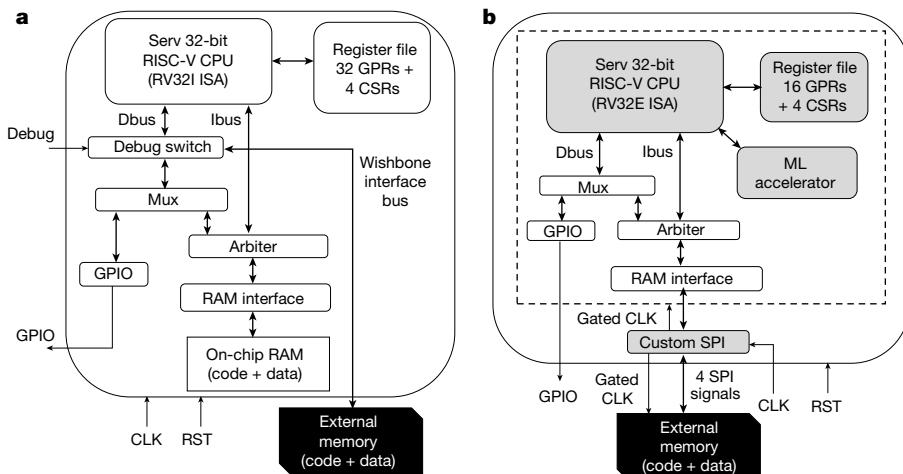
microcontrollers. However, the unit cost of a few dollars for a microprocessor is a non-starter for many of these emerging applications. There are three key reasons why the unit cost cannot be reduced to sub-dollar levels: (1) Si fab capital costs; (2) CPU instruction set, intellectual property (IP) licence and non-recurring engineering (NRE) costs; and (3) cost of microprocessor chip packaging.

1. Si fab capital: the state-of-the-art silicon fabrication plants require several millions in capital investment and focus primarily on high-end microprocessors for advanced nodes (for example, 2–3 nm). Lower-end microprocessors such as microcontrollers for embedded systems can be produced in older, legacy fabs (>65 nm) to minimize investment costs. However, factories invest comparatively less in legacy fabs, which leads to increased operational and compliance costs over time along with environmental concerns<sup>4</sup>.

2. IP licence and NRE costs: every microprocessor has a CPU that executes programs based on its specific instruction set architecture (ISA), which can be either proprietary (for example, x86 by Intel & AMD, ARMvX by Arm) or open source (that is, RISC-V; ref. 1). Proprietary ISAs offer a mature ecosystem but are costly to license and offer limited customization as adding new instructions is generally restricted. Alternatives include licensing an existing CPU that incurs upfront IP costs, or developing a new proprietary ISA, which entails CPU design NRE costs and software toolset development expenses, both of which increase unit cost.

3. Chip packaging: Si dies are sensitive to environmental conditions, including mechanical stress. They are brittle and need to be assembled onto a separate package (for example, plastic, ceramic) to protect

<sup>1</sup>Pragmatic Semiconductor, Cambridge, UK. <sup>2</sup>John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. <sup>3</sup>Qamcom, Karlstad, Sweden. <sup>4</sup>These authors contributed equally: Emre Ozer, Jędrzej Kufel, Shvetank Prakash. <sup>✉</sup>e-mail: eozer@pragmaticssemi.com



**Fig. 1 | System architecture.** **a**, Servant as the baseline system architecture for Flex-RV. **b**, Modified Servant and additional blocks added to Servant to create Flex-RV. GPRs, general-purpose registers; CSRs, control and status registers; CLK, clock; RST, reset; Dbus, data bus; Ibus, instruction bus; Mux, multiplexer.

them from the environment. The cost of die attachment, assembly, packaging material and packaged chip test increases the nominal unit cost of a microprocessor. Moreover, embedding a rigid Si microprocessor into one of these wearable or implantable devices is challenging because the chip package of the microprocessor will limit the abilities of the device in terms of its rigidity and thickness. The rigid chip packaging material must be eliminated to reduce the cost of the microprocessor and to allow its conformability in emerging applications.

There is a need for a microprocessor that is ultralow cost and conformable providing acceptable performance to meet the requirements of the emerging applications. To realize a sub-dollar and physically flexible microprocessor, two enabling technologies can help us overcome the previous limitations:

1. **RISC-V ISA:** it is essential that the NRE costs of a CPU inside the sub-dollar microprocessor are minimized (for example, ISA/IP licensing). RISC-V (ref. 1) is an open and free ISA standard that allows anyone to develop a CPU implementing the ISA, which eliminates the ISA licence fees, and therefore the NRE costs of microprocessor development. Moreover, the RISC-V ecosystem encourages innovation and does not restrict modifications or extensions (for example, new instructions), enabling designers to tailor the microprocessor to meet bespoke or application needs without the burden of additional costs or restrictive licences.
2. **IGZO TFTs:** a sub-dollar microprocessor must be fabricated in a semiconductor technology alternative to Si in a fab that is orders of magnitude cheaper with a reduced carbon footprint than Si fabs (including legacy fabs)<sup>5</sup> in which carbon footprint in this context refers to the CO<sub>2</sub> emissions during the processes of chip design and fabrication. IGZO is a metal-oxide semiconductor material that can be used to develop TFTs<sup>2</sup> on a flexible polyimide substrate using a low-temperature lithography process. The TFTs are manufactured by deposition of semiconductor, dielectric and electrodes on an insulator substrate as opposed to developing MOSFETs (metal-oxide semiconductor field effect transistors) using doped Si on a rigid silicon wafer. *FlexLogic* fab of Pragmatic can manufacture flexible chips or FlexICs made of IGZO TFTs, which have orders of magnitude lower carbon footprint per FlexIC than an equivalent Si fab<sup>6</sup>. Also, FlexICs do not need additional chip packaging (for example, plastic, ceramic) like Si dies to protect them from mechanical stress because they are not brittle and can resist mechanical stress such as bending. This reduces the microprocessor unit cost further into sub-dollar levels at high volumes while also enabling flexion capabilities.

There have been previous non-silicon microprocessor prototypes on flexible substrates. Early works were based on 8-bit CPUs that use low-temperature poly-silicon TFTs<sup>7–10</sup> and organic, hybrid-oxide and metal-oxide TFTs<sup>11–13</sup>. In ref. 14, the authors presented PlasticARM—the first 32-bit microprocessor based on the proprietary ARM ISA that uses IGZO-based TFTs. PlasticARM was not a programmable microprocessor running only three hardwired programs in its on-chip read-only memory. Although not on a flexible substrate, ref. 15 demonstrated a 16-bit CPU derived from the RISC-V ISA built with complementary carbon nanotube transistors on a conventional Si-wafer. Apart from non-silicon microprocessors, previous studies<sup>16–20</sup> have demonstrated several ML ASICs (application-specific integrated circuits) designed and implemented using IGZO TFTs fabricated on flexible substrates.

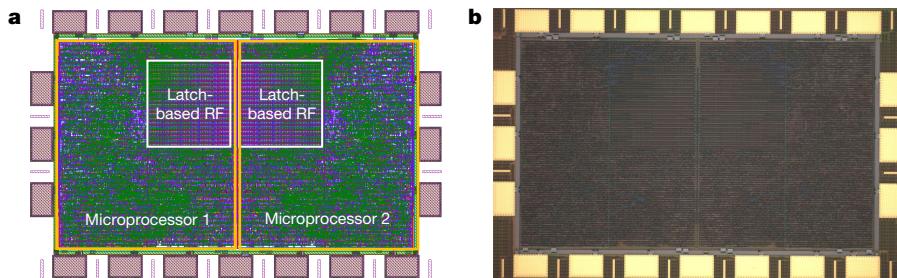
Flex-RV is different from previous non-silicon microprocessors in three aspects: (1) The 32-bit microprocessor is programmable and can run arbitrary compiled programs written in high-level languages (for example, C). (2) The open-source nature of RISC-V allows extending the ISA with new instructions, so Flex-RV also features a programmable ML accelerator tightly coupled to the CPU with custom RISC-V instructions added. (3) We demonstrate the functionality of the microprocessor beyond wafer-level testing by assembling the microprocessor bare die onto a flexible printed circuit board (FlexPCB) using an innovative technique developed for this work called over-edge printing or OEP, whereas all other previous works were demonstrated at wafer level without being released from the glass carrier and diced.

This allows us to conduct physical bending tests on Flex-RV while running the programs to validate the functionality under mechanical stress. We demonstrate the functionality of the Flex-RV assembled onto the FlexPCB, and quantify the performance across several test programs. Our results show that Flex-RV assembled on a FlexPCB can run as fast as 60 kHz while consuming less than 6 mW and can be bent to a radius of curvature below 5 mm while still executing programs correctly.

This work unlocks the next generation of intelligent, ubiquitous computing in which the integration of flexible, programmable processors with ML capabilities embeds smart sensing into everyday objects.

## Flex-RV system architecture

We use the open-source Serv RISC-V CPU<sup>21</sup> to implement the microprocessor, which is one of the smallest open-source RISC-V CPUs developed so far. Serv is a bit-serial CPU in which 32-bit computations are performed bit by bit rather than in parallel as a more typical CPU would do. This reduces the design complexity of a 32-bit RISC-V processor at



**Fig. 2 | Flex-RV chip.** **a**, Layout of the 9 mm × 6 mm test chip containing two Flex-RV microprocessors. **b**, Die photo of the test chip taken under a microscope.

the expense of performance, and its low design complexity is the main reason why Serv is chosen as the CPU for Flex-RV. We design a simple system-on-a-chip (SOC) around the Serv CPU to develop the microprocessor that can communicate with the external world.

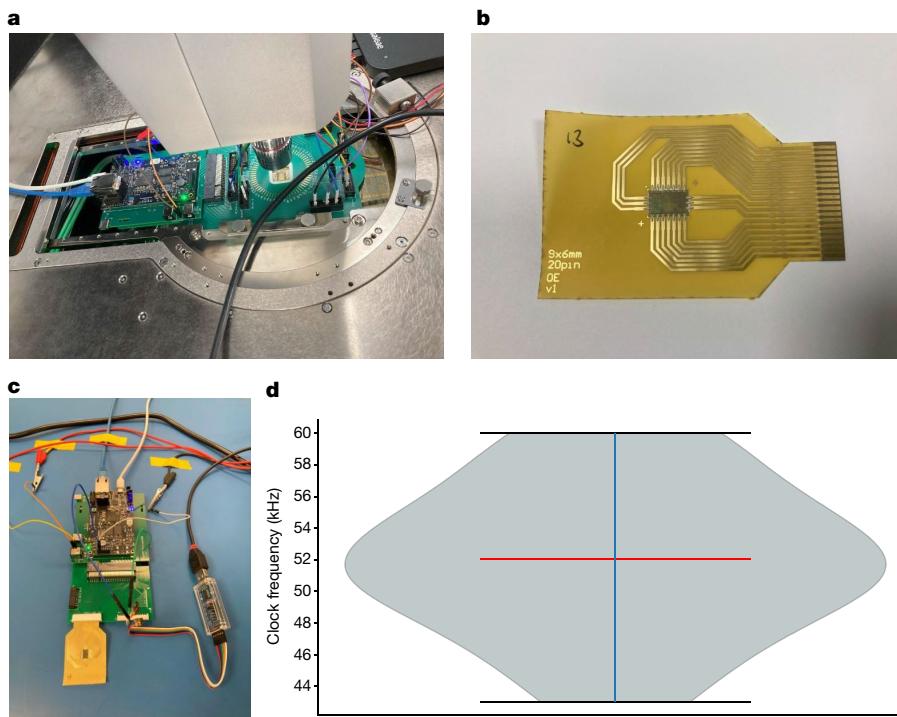
Flex-RV is designed to be a simple SOC consisting of a Serv RISC-V CPU and additional peripherals. The Servant SOC<sup>22</sup> that was developed as a reference platform for Serv is used as a starting point for Flex-RV. Servant as shown in Fig. 1a contains the Serv CPU, a register file (RF), a debug switch, a multiplexor switch, an arbiter, a Random Access Memory (RAM) interface, an on-chip RAM block and a General-purpose Input/Output (GPIO) interface. Serv supports the RISC-V RV32I ISA (ref. 1) that has 32 general-purpose registers, and four extra control and status registers. The code and data are loaded into the on-chip RAM through the debug switch from an external memory. The GPIO interface can be used to communicate with the external world (for example, display).

We modified some of the features in Servant to develop Flex-RV as shown in Fig. 1b. For example, Serv is modified to support the RV32E ISA (ref. 1), which supports the same instruction set as RV32I but uses only the first 16 registers. We removed the debug switch and on-chip RAM. As the current FlexIC technology does not allow us to put a sizable

on-chip memory (for example, >1kB SRAM) for area/power reasons, we opt to use an off-chip memory to access the code and data, and therefore, added a custom Serial Peripheral Interface (SPI) or C-SPI block to communicate with the external memory. The C-SPI block fetches 32-bit RISC-V instructions from the external memory as well as performing 32-bit data load/store operations from/to the memory. The main reason for using the C-SPI to communicate with the external memory rather than accessing it using parallel data, address and control buses is to reduce the number of off-chip pads to simplify the assembly process of Flex-RV onto a FlexPCB. The details of the C-SPI block, including the memory transactions, can be found in the Methods.

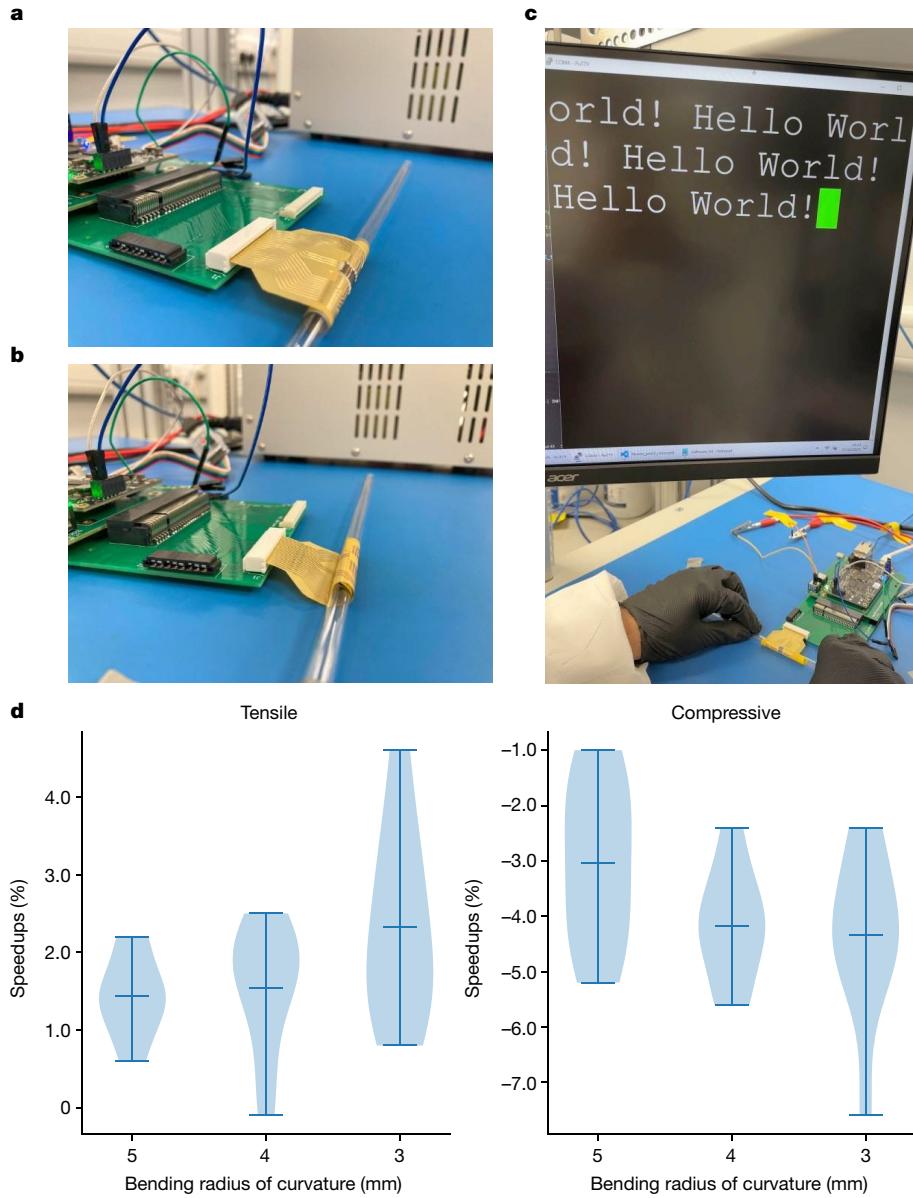
We also designed a programmable ML accelerator that is an SIMD (single instruction multiple data) engine for speeding up matrix multiplication and post-processing operations commonly used in contemporary ML algorithms. The ML accelerator is tightly coupled to Serv as a custom functional unit (CFU)<sup>23</sup>, and four new instructions were added to the RV32E ISA so that the accelerator can be programmed. The details of the ML model, accelerator architecture, four new instructions and programming interface are described in the Methods.

Flex-RV has been validated using the RISC-V architectural compliance suite<sup>24</sup> for RV32E ISA. Several test benchmarks were also developed



**Fig. 3 | Test infrastructure.** **a**, Wafer-on-glass test infrastructure using an FPGA board connected to the probe card attached to the semi-automatic wafer probe station. **b**, The FlexPCB on which the die is assembled. **c**, The FlexPCB is connected to the FPGA board. The GPIO signal from the test chip is connected

back to the FPGA, encoded and connected to UART2USB to display the output of the program on a laptop. **d**, The distribution of the highest achievable clock frequencies in kHz for all test benchmarks per Flex-RV in a violin plot is shown for Flex-RVs assembled onto 13 FlexPCBs.



**Fig. 4 | Bendability tests of Flex-RV on a FlexPCB while executing test programs.** The FlexPCB is rolled around a non-conductive cylinder along an axis parallel to the connector of the FlexPCB and, therefore, along the length of the Flex-RV chip until it is bent to a radius of curvature of the cylinder. **a**, Tensile mode. **b**, Compressive mode. **c**, A Flex-RV microprocessor is running the 'Hello World' displayed on the screen while bent in tensile mode (see also the Supplementary Video for the microprocessor operating in the entire bending

cycle). **d**, The distributions of relative speedups in tensile and compressive modes with respect to the flat position for 3 mm, 4 mm and 5 mm bending radii of curvature in violin plots are shown. The relative speedup is measured as the percentage increase in the clock frequency of a Flex-RV chip in a bent mode with respect to the clock frequency of the chip at the flat position. Negative speedup implies slowdown.

to verify its functionality, which were written in C (with some inline assembly code), and compiled using the RISC-V GNU compiler tools. The details of the test benchmarks and the compilation environment can be found in the Methods.

We implement Flex-RV using 0.6  $\mu\text{m}$  FlexIC technology of Pragmatic that uses a unipolar logic consisting of an n-type IGZO TFT and a resistive pull-up. The chip implementation (that is, synthesis, place and route, static timing analysis, post-place-and-route simulation and sign-off) is performed using commercial Electronic Design Automation (EDA) tools. The RF is a dual-ported array having one read and one write port with an array size of  $20 \times 32$  bits (or 80 bytes). It is implemented with latches using a layout abutment method to increase the density of the array (see the Methods for the details). The layout and die photo of the chip are shown in Fig. 2. The test chip has a  $9 \text{ mm} \times 6 \text{ mm}$  die size and 20 pins, and two Flex-RV microprocessors are fitted into

the chip. Each Flex-RV microprocessor has a core area of  $17.5 \text{ mm}^2$ , a NAND2-equivalent gate count of 12,596, and a power consumption of 5.8 mW at 3 V. The 5.8 mW power consumption is predominantly static (99%) because of the resistive pull-up logic.

Test chips were fabricated in FlexLogic fab of Pragmatic on 200 mm polyimide wafers with a thickness of 30  $\mu\text{m}$  using the thin-film deposition process that creates patterned layers of metal-oxide TFTs and resistors, with four routable metal layers and an additional RDL (redistribution layer) to re-route the core pads of a chip to the periphery pads that will be the interface to the external world.

## Test infrastructure and results

A test infrastructure consists of two integral phases: (1) wafer-on-glass testing and (2) FlexPCB testing.

A MicroZed Zynq-7000 field-programmable gate array (FPGA) board<sup>25</sup> is used to test the Flex-RVs. The FPGA chip is programmed to emulate the external memory and communicates with the Flex-RVs on the wafer through an SPI block implemented also on the FPGA. The binaries of the compiled test benchmarks are loaded into the memory on the FPGA, and the FPGA board resets each Flex-RV in a test chip after which they start executing each kernel. The results of the test benchmarks are transmitted through the GPIO pin of each Flex-RV, which is connected through FPGA to the universal asynchronous receiver-transmitter (UART) interface of a personal computer (PC) to display the results on the screen. The FPGA board also supplies each test chip with power, clock and reset signals.

In the first phase, the test chips are functionally tested on wafer using a semi-automatic wafer probe station (Fig. 3a) to identify functional Flex-RVs. In the second phase, the functional Flex-RVs identified in the first phase are assembled onto FlexPCBs using OEP—a new assembly method developed for this work (Fig. 3b). The assembled FlexPCB is then connected to the FPGA board through an FPC connector (Fig. 3c). The details of the FPGA-based test infrastructure and the OEP assembly process can be found in the Methods.

Figure 3d shows the distribution of the highest achievable clock frequencies in kilohertz (kHz) at which a Flex-RV microprocessor in a test chip can run all test benchmarks across 13 assembled FlexPCBs. The average and maximum clock frequencies across 13 FlexPCBs are 52 kHz and 60 kHz, respectively.

Finally, we validate the functionality of the Flex-RVs assembled FlexPCBs under mechanical stress, and measure the variations in the highest achievable clock frequencies. The mechanical stress is achieved by rolling each FlexPCB around a non-conductive cylinder along an axis parallel to the connector of the FlexPCB (also along the length of the Flex-RV) until the test chip is bent to a radius of curvature of the cylinder. IGZO TFTs are known to be bent to a radius of curvature of 3 mm without damage, and device parameters such as mobility and threshold voltage change under different strain scenarios<sup>26</sup>. However, these tests aimed at individual IGZO TFT devices while we study primarily the bendability tolerance at the FlexIC level.

There has been no study to demonstrate the bendability of a complex FlexIC like Flex-RV alone or assembled on a FlexPCB while it is powered up and running. We demonstrate the bendability of Flex-RV assembled a FlexPCB using three different cylinders having 3 mm, 4 mm and 5 mm radii and test them in tensile (that is, the curvature of the chip is outwards) and compressive (that is, the curvature of the chip is inwards) modes as shown in Fig. 4a,b, respectively. A total of seven assembled FlexPCBs comprising nine Flex-RV microprocessors are tested for bendability analysis. This is a dynamic bendability test as rolling the FlexPCB and bending the Flex-RV chip occur while both Flex-RV microprocessors in the chip are running the test benchmarks—an example shown in Fig. 4c. The test benchmarks start running on a microprocessor when a FlexPCB is flat. While the programs are running, the FlexPCB/Flex-RV is bent to the tensile mode around the cylinder and then brought back to flat. Next, they are bent to the compressive mode and then brought back to flat again. This is repeated twice per Flex-RV microprocessor per FlexPCB.

Figure 4d shows the speedup results of the bending experiments across three different bending radii. Test benchmarks are run to completion on each Flex-RV microprocessor when the chip is in a tensile or compressive mode during the dynamic bending test. We measure the highest achievable clock frequency of a Flex-RV microprocessor across all test benchmarks when in a tensile or compressive mode, and compare it to the highest achievable clock frequency when it is flat, which is the speedup in clock frequency. Our results show that Flex-RV microprocessors demonstrate flexibility as each microprocessor goes back to its highest achievable clock frequency, that is, between two flat positions, two tensile and compressive modes. We also observe two trends: (1) Flex-RV gets faster in tensile mode but

runs slower in compressive mode, which is attributed to the change in the device (that is, TFTs and resistors) and wire parameters (for example, resistance and capacitance). (2) As the radius of curvature decreases, its performance increases in tensile mode. This is because as the chip gets more tensile, the device and wire parameters change proportionally to improve the performance of Flex-RV, whereas the opposite is observed in compressive mode. The bending experiments demonstrate that Flex-RV can function correctly at a bending radius of 3 mm, and on average, Flex-RV can operate 2.3% faster in tensile mode and 4.3% slower in compressive mode with respect to the flat position.

## Conclusions

We have developed a sub-dollar, bendable and flexible microprocessor, Flex-RV, using a non-silicon technology based on 0.6  $\mu\text{m}$  IGZO TFT technology and supporting the open RISC-V instruction set. Moreover, we have incorporated a programmable hardware accelerator into Flex-RV to enable ML applications. Flex-RV has been fabricated on a polyimide substrate having a thickness of 30  $\mu\text{m}$  in the low-cost and low-environmental footprint FlexLogic fab and then assembled onto a FlexPCB with a thickness of 45  $\mu\text{m}$  to build an ultrathin computing system. Our experimental results have shown that Flex-RV can run as fast as 60 kHz while consuming less than a total power of 6 mW and can operate correctly down to a 3-mm bending radius of curvature with performance variation from 2.3% speedup to 4.3% slowdown on average while running programs. Thus, it can meet the requirements of many emerging applications in fast-moving consumer goods, wearables and healthcare devices.

Flex-RV is an important milestone in building an ultralow-cost bendable computer that does not rely on any conventional electronic components.

## Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at <https://doi.org/10.1038/s41586-024-07976-y>.

1. Waterman, A., Lee, L., Patterson, D. A. & Krste A. *The RISC-V instruction set manual, Volume I: Base user-level ISA*. Report UCB/EECS-2011-62 (EECS Department, University of California, Berkeley, 2011).
2. Weimer, P. K. The TFT: a new thin-film transistor. *Proc. IRE* **50**, 1462–1469 (1962).
3. Bleier, N. et al. Exploiting short application lifetimes for low cost hardware encryption in flexible electronics. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–6 (Institute of Electrical and Electronics Engineers, 2023).
4. Chen, S., Gautam, A. & Weig, F. *Bringing Energy Efficiency to the Fab* (McKinsey, 2013).
5. Ahamed, A., Anand, C. & Young, J. *Intelligent Low-Carbon Semiconductor Manufacturing* White Paper (Pragmatic, 2023).
6. Ahamed, A., Anand, C. & Young, J. *Life Cycle Analysis of FlexICs* White Paper (Pragmatic, 2023).
7. Takayama, T. et al. A CPU on a plastic film substrate. In *Symposium on VLSI Technology* 230–231 (IEEE, 2004).
8. Dembo, H. et al. RFCPUs on glass and plastic substrates fabricated by TFT transfer technology. *IEEE International Electron Devices Meeting (IEDM)* 125–127 (IEEE, 2005).
9. Karaki, N. et al. A flexible 8b asynchronous microprocessor based on low-temperature poly-silicon TFT technology. *IEEE International Solid-State Circuits Conference (ISSCC)* 272–273 (IEEE, 2005).
10. Kurokawa, Y. et al. UHF RFCPUs on flexible and glass substrates for secure RFID systems. *IEEE J. Solid-State Circuits* **43**, 292–299 (2008).
11. Myny, K., van Veenendaal, E., Gelinck, G. H., Genoe, J. & Dehaene, W. An 8-bit, 40-instructions-per-second organic microprocessor on plastic foil. *IEEE J. Solid-State Circuits* **47**, 284–291 (2012).
12. Myny, K. et al. 30.1 8b Thin-film microprocessor using a hybrid oxide-organic complementary technology with inkjet-printed  $\text{P}^2\text{ROM}$  memory. In *IEEE International Solid-State Circuits Conference (ISSCC)* 486–487 (IEEE, 2014).
13. Çeliker, H. et al. Flex6502: a flexible 8b microprocessor in 0.8  $\mu\text{m}$  metal-oxide thin-film transistor technology implemented with a complete digital design flow running complex assembly code. In *IEEE International Solid-State Circuits Conference (ISSCC)*, 272–274 (IEEE, 2022).

14. Biggs, J. et al. A natively flexible 32-bit Arm microprocessor. *Nature* **595**, 532–536 (2021).
15. Hills, G. et al. Modern microprocessor built from complementary carbon nanotube transistors. *Nature* **572**, 595–602 (2019).
16. Ozer, E. et al. Bespoke machine learning processor development framework on flexible substrates. In *IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, 1–3 (IEEE, 2019).
17. Ozer, E. et al. A hardwired machine learning processing engine fabricated with submicron metal-oxide thin-film transistors on a flexible substrate. *Nat. Electron.* **3**, 419–425 (2020).
18. Ozer, E. et al. Binary neural network as a flexible integrated circuit for odour classification. In *IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, 1–4 (IEEE, 2020).
19. Ozer, E. et al. Malodour classification with low-cost flexible electronics. *Nat. Commun.* **14**, 777 (2023).
20. Iordanou, K. et al. Low-cost and efficient prediction hardware for tabular data using tiny classifier circuits. *Nat. Electron.* **7**, 405–413 (2024).
21. SERV - The SERIAL RISC-V CPU. GitHub [github.com/olofk/serv](https://github.com/olofk/serv) (Olof Kindgren, 2020).
22. Servant: FPGA reference platform. SERV [serv.readthedocs.io/en/latest/servant.html](https://serv.readthedocs.io/en/latest/servant.html) (Olof Kindgren, 2020).
23. Prakash, S. et al. CFU playground: full-stack open-source framework for Tiny Machine Learning (TinyML) acceleration on FPGAs. In *IEEE International Symposium on Performance Analysis of Systems and Software* 157–167 (IEEE, 2023).
24. RISC-V Architecture Test. GitHub <https://github.com/riscv-non-isa/riscv-arch-test> (2024).
25. Avnet. MicroZed™ FPGA board with Zynq®-7000. Product Brief (2022).
26. Jang, H.-W., Kim, G.-H. & Yoon, S.-M. Analysis of mechanical and electrical origins of degradations in device durability of flexible InGaZnO thin-film transistors. *ACS Appl. Electron. Mater.* **2**, 2113–2122 (2020).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024

## Methods

In this section, we describe the microarchitecture of the C-SPIIP block, the ML hardware accelerator, the new instructions to program it, the description of the test benchmarks, the compilation environment, the implementation details of the latch-based RF, the details of the FPGA-based test infrastructure and the OEP assembly method.

### Architecture of the C-SPI

To reduce the complexity of assembling Flex-RV onto a FlexPCB in terms of the number of pads, we have developed C-SPI to communicate with external memory by interfacing the 79 RAM ports to 4 SPI ports rather than exposing the 79 ports to the pads of the chip. This results in slower memory access but facilitates the assembly of Flex-RV onto a FlexPCB.

The internal and external ports of the C-SPI are shown in Extended Data Fig. 1. The number of address bits is set to 10 and allows the processor to access 4 kB of external memory, which is sufficient to store the code and data for the largest test bench.

As Serv requires a single-cycle memory access, C-SPI implements a top-level clock gate that halts the rest of the Flex-RV microprocessor when any external memory transaction occurs using a gated clock signal. The gated clock signal is also sent as an output to the external controller such as an FPGA.

There are two internal registers dedicated to transmission in the C-SPI. The transfer register consists of a 46-bit transmit register and a 32-bit receive register. The transmit register serves the purpose of writing to the memory (that is, store operation) and transmitting the designated address for reading, whereas the receive register is used to capture the desired data (that is, load operation). Each write transfer requires a total of 47 clock cycles. The initial clock cycle designates the write mode, followed by 10 clock cycles for sending the memory address, and subsequently, 32 clock cycles for transmitting the write data. The final four cycles are allocated for the write mode type determined by the 4-bit 'Write En'. The commands are provided in Extended Data Table 1.

Every read transfer involves a total of 46 clock cycles. The initial cycle is dedicated to specifying the reading mode, followed by 10 clock cycles for transmitting the memory address. Subsequently, three cycles are allocated for reading data from the memory and preparing for transmission. Finally, 32 clock cycles are necessary to receive the 32-bit data. The state machine in Extended Data Fig. 2 shows the write and read modes broken down into states.

### Microarchitecture of the ML hardware accelerator

The ML hardware accelerator is a tightly coupled CFU<sup>23</sup> extending the data path of the CPU. It follows the RISC-V instruction R-format in which it receives two operands from the RF and writes one result back. Extended Data Fig. 3 shows the microarchitecture of the ML accelerator in relation to the rest of the CPU. The boundary between the CPU and CFU is strictly logical. The current implementation flattens the design and optimizes, places and routes it all together.

The accelerator features a SIMD multiply–accumulate processing engine composed of two  $8 \times 4$  multipliers and two  $4 \times 4$  multipliers as prior art<sup>27</sup> has shown sufficient accuracy using 4-bit integer quantization for weights of embedded ML models with 8-bit inputs. The accelerator also has specialized hardware for handling post-processing of activations such as bias addition, applying non-linearity (for example, ReLU) and quantization rescaling. These architectural features can be invoked in software using four custom instructions: (1) use the two  $8 \times 4$  multipliers alone; (2) use all four multipliers with  $4 \times 4$  precision; (3) apply bias addition and non-linearity to activations; and (4) rescale activations for quantization.

To test our accelerator, we developed a Tiny Machine Learning (TinyML) model using Tensorflow<sup>28</sup> to perform ECG anomaly detection using the ECG5000 time series classification dataset<sup>29</sup>. Our tiny neural

network consists of a one-dimensional convolution followed by a fully connected layer, an approach commonly used in simple time series classification. We use linear symmetric affine quantization instead of asymmetric quantization to avoid the additional computation of subtracting offsets for each loop iteration.

Using post-training quantization, we quantize our model down to 4-bit precision for weights and activations with minimal accuracy loss. However, the bias must be kept at 32-bit precision to avoid large degradations in accuracy. Finally, we implemented the power of two scaling for our step size so that multiplication during post-processing of a layer could be done using bit shifts. These model optimizations coupled with our hardware co-design enable us to run ML algorithms even in these deeply embedded application settings in which computational resources are limited and constrained.

### Test benchmarks and compilation environment

We have written several test benchmarks in C language with some inline RISC-V RV32E assembly code to use in functional testing of the wafers and assembled FlexPCBs. The test benchmarks, as shown in Extended Data Table 2, were developed to exercise many hardware blocks in Flex-RV such as the functional units inside the Serv CPU such as the arithmetic logic unit, RF, ML accelerator and also outside the CPU such as the C-SPI (through memory reads and writes). The core of the test benchmarks is written in C, whereas the GPIO communication code is written in RV32E assembly. At the end of each test benchmark, the results from the test benchmark are sent through the GPIO to the external world. In our test infrastructure, the external world is the FPGA board that reads the data coming out of the GPIO output, and the FPGA board relays the data to a PC through UART to display the results on the PC screen.

The test benchmarks were compiled for bare metal execution using the riscv64-unknown-elf-gcc compiler with the flags of 'rv32e', 'ilp32e', 'nostdlib' and 'nostartfiles' to generate an elf file, which was then translated into a hex file using riscv64-unknown-elf-objcopy with the flag '-O verilog'. The hex files can be used to test Flex-RV in Verilog simulation environments as well as in wafer and assembled FlexPCB tests when the FPGA memory is loaded with the hex file of each test benchmark.

To invoke and compile the custom instructions for ML workloads, the 'asm' directive for inlining assembly is used within the C kernel to directly insert assembly language instructions. This approach bypasses the need for the compiler to understand the custom instruction by manually coding it into the program. The custom instruction is defined using the 'word' directive followed by the opcode and operand specifiers, which are filled in based on the provided C variables holding the operands to be used. When the processor sees the dedicated opcode for the custom instruction, the operands are passed to the accelerator unit.

### Implementation of the RF circuitry

The block diagram of the dual-port latch-based memory used as an RF is shown in Extended Data Fig. 4 allowing one register read and one write at the same cycle. Write and read address decoders are synthesized using standard cells. Logic gates in the highlighted region instantiate special standard cells that allow signal connection through the abutment. During implementation, the placement of these cells is controlled through relative cell placement. This achieves much higher cell density and reduces the amount of routing that would otherwise be performed by the EDA tool. All latches use an additional enable signal shown as RWL (read word line), which enables the output RBL (read bit line). To reduce the static power consumption, latches in the bottom word line implement a pull-up resistor on the RBL output, whereas latches in other word lines do not implement such a resistor. Hence, there is only one pull-up resistor per bit column.

## FPGA-based test infrastructure

The test infrastructure uses MicroZed FPGA board with Zynq-7000 SOC (ref. 25). Our testing methodology involves two distinct phases: wafer-on-glass testing and FlexPCB testing. These phases serve different purposes and are integral parts of our test infrastructure.

1. Wafer-on-glass testing validates the functionality of each Flex-RV on the wafer (or more specifically wafer on glass) using a semi-automatic wafer probe station. The goal of this phase is to identify functional Flex-RVs that are suitable for FlexPCB assembly and testing. The test probe card of the wafer probe station is connected to the FPGA board that sends each test benchmark as an input to a Flex-RV on the wafer, and the output of the executed benchmark from the Flex-RV is sent through the probe card to the FPGA for verification. We also connect a logic analyser to verify that all input and output signals of the Flex-RVs are within specifications. The logic analyser has an input capacitance of 10 pF that has an effect on the performance of the IO drivers inside Flex-RVs. This leads to achieving lower clock frequencies for Flex-RVs.

2. The second phase is FlexPCB testing in which the wafers are released from the glass carrier, and the dies are diced. The goal of FlexPCB testing is to evaluate the performance of functional Flex-RVs or known good dies (KGDs) in their intended flexible configuration. The KGDs identified in the first phase are separated from the faulty dies and assembled onto FlexPCBs. The FlexPCB is a polyimide substrate with a thickness of 45  $\mu\text{m}$  onto which a KGD is assembled. The KGD is attached to the FlexPCB using epoxy, whereas the die pads are aligned with the PCB pads. A silver paste is printed onto the aligned pads to bond the pads electrically and mechanically through the OEP assembly method as described below. The PCB traces connect the die pads to an FPC connector through which the FlexPCB is connected directly with the same FPGA board. No logic analyser is used in this phase as functional correctness was already verified in the previous phase. Once the functionality is confirmed, the FlexPCB tests can be conducted without the logic analyser.

Our experiments showed that the highest achievable clock frequencies in wafer-on-glass testing are 25–30% lower than the FlexPCB testing because of high capacitive loading from the logic analyser on the IO drivers of Flex-RVs. The total resistance and capacitive of PCB traces between the test chip and the FPGA board have much less effect compared with the effect of the logic analyser. The total resistance and capacitance from test chips to the FPGA board in wafer-on-glass test setup are  $2\Omega$  and about 2.5 pF, whereas these values are about  $1\Omega$  and 2.5 pF on the FlexPCB test setup. The additional 10 pF capacitance from the logic analyser is the reason for lower clock frequencies in the wafer-on-glass test setup.

The block diagram of FPGA implementation is shown in Extended Data Fig. 5. The Python script running on the processing system controls the test. Flex-RV is held in reset during the initial setup stage, whereas the 4 kB memory is programmed with benchmark binary and the data flow is configured. Although GPIO from Flex-RV could be connected directly to UART Tx pin, as the clock to Serv is halted with every memory transaction, the bitstream is corrupted. Therefore, the GPIO data are captured and re-encoded into the correct UART bitstream using the gated clock signal. Moreover, UART transactions are stored in a separate 1 kB memory and later compared with the expected response. This allows a frequency sweep to be run for each benchmark.

## OEP assembly method

With conventional flip-chip assembly approaches—for example, anisotropic conductive paste and anisotropic conductive film, with applying relatively high thermo-compression force—there is a high chance of circuitry damage by conductive particles or thickness difference of the area on which the chip is bonded. An alternative is to

attach the flexible chip face-up (non-flip chip) with thermo-free assembly process named OEP assembly method developed by Pragmatic. OEP has the following steps as shown in Extended Data Fig. 6:

1. Underfill adhesive application: non-conductive adhesive is applied on the FlexPCB as underfill adhesive to hold the FlexIC. The volume and position of the underfill adhesive must be well controlled for the coverage of the four edges to prevent undesired conductive path formed by infiltration underneath the FlexIC. At the same time, the underfill adhesive must not overflow on top of the FlexIC during the die attachment process, because that could hinder the conductive path formation above the contact pads.
2. Die attachment: FlexIC is bonded with the RDL layer facing up. Specially designed ejectors release singulated chips from release tape on the frame, preventing damage to the FlexIC circuitry. The levelling and bond force control must be optimized to achieve 100% coverage of the four edges but no underfill adhesive overflowing on top of the FlexIC.
3. Interconnection formation: stretchable isotropic conductive adhesive is applied by a high-precision time-pressure controlled dispensing machine with camera alignment functionality with high placement accuracy. Dispense height measurement is essential to control the isotropic conductive adhesive thickness and its uniformity. The isotropic conductive adhesive material is specially chosen to be stretchable, which can maintain low-resistance electrical connectivity under bending conditions. The coefficient of thermal expansion and elongation should match with the materials such as FlexIC and FlexPCB to keep the devices functional and reliable during and after bending conditions.

## Data availability

Source data are provided with this paper.

## Code availability

Serv is an open-source CPU, which is freely available at GitHub (<https://github.com/olofk/serv>). The source code of the test benchmarks, the changes made in the Serv CPU Verilog code, and the Verilog code of the ML hardware accelerator are available from the corresponding author upon request.

27. Banner, R., Nahshan, Y. & Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Proc. 33rd International Conference on Neural Information Processing Systems* Vol. 714, 7950–7958 (ACM, 2019).
28. Abadi, M. et al. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* Vol. 16 (eds Keeton, K. & Roscoe, T.) 265–283 (Association for Computer Machinery, 2016).
29. Chen, Y. & Keogh, E. ECG5000 dataset <https://www.timeseriesclassification.com/description.php?Dataset=ECG5000> (2024).

**Author contributions** O.K. developed the open-source RV32I-based Serv and Servant. E.O. and O.K. modified Serv to support RV32E ISA. E.O. architected the Flex-RV system architecture. A.R. developed the C-SPI block. S.P., D.K., G.H. and V.J.R. developed the programmable ML hardware accelerator and integrated it into Flex-RV. E.O., J.K., S.P. and A.R. did Flex-RV SOC integration and simulation. J.K. developed the FPGA test setup and infrastructure. E.O. and S.P. developed the test benchmarks and prepared the compilation environment. D.J. and J.K. implemented the RF. J.K. did the physical chip implementation of Flex-RV. E.O., J.K. and R.P. were involved in the facilitation of fabricating Flex-RV. E.O. and J.K. did the wafer-on-glass functional testing. N.N., R.W. and F.A. assembled the KGDs onto FlexPCB. E.O., J.K. and S.P. did the functional testing of the assembled FlexPCB, including the bending tests. E.O., J.K., S.P., A.R., R.W., V.J.R. and R.P. wrote the paper.

**Competing interests** The authors declare no competing interests.

## Additional information

**Supplementary information** The online version contains supplementary material available at <https://doi.org/10.1038/s41586-024-07976-y>.

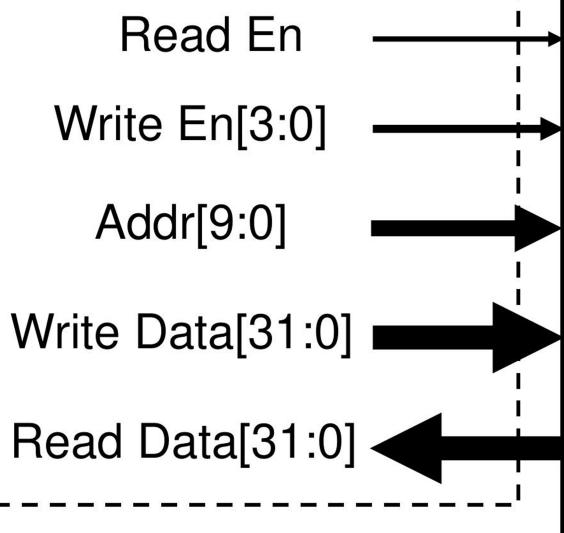
**Correspondence and requests for materials** should be addressed to Emre Ozer.

**Peer review information** *Nature* thanks Subho Dasgupta, Gabriel Marques and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

**Reprints and permissions information** is available at <http://www.nature.com/reprints>.

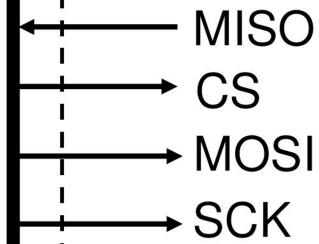
## Internal Ports

### RAM Ports



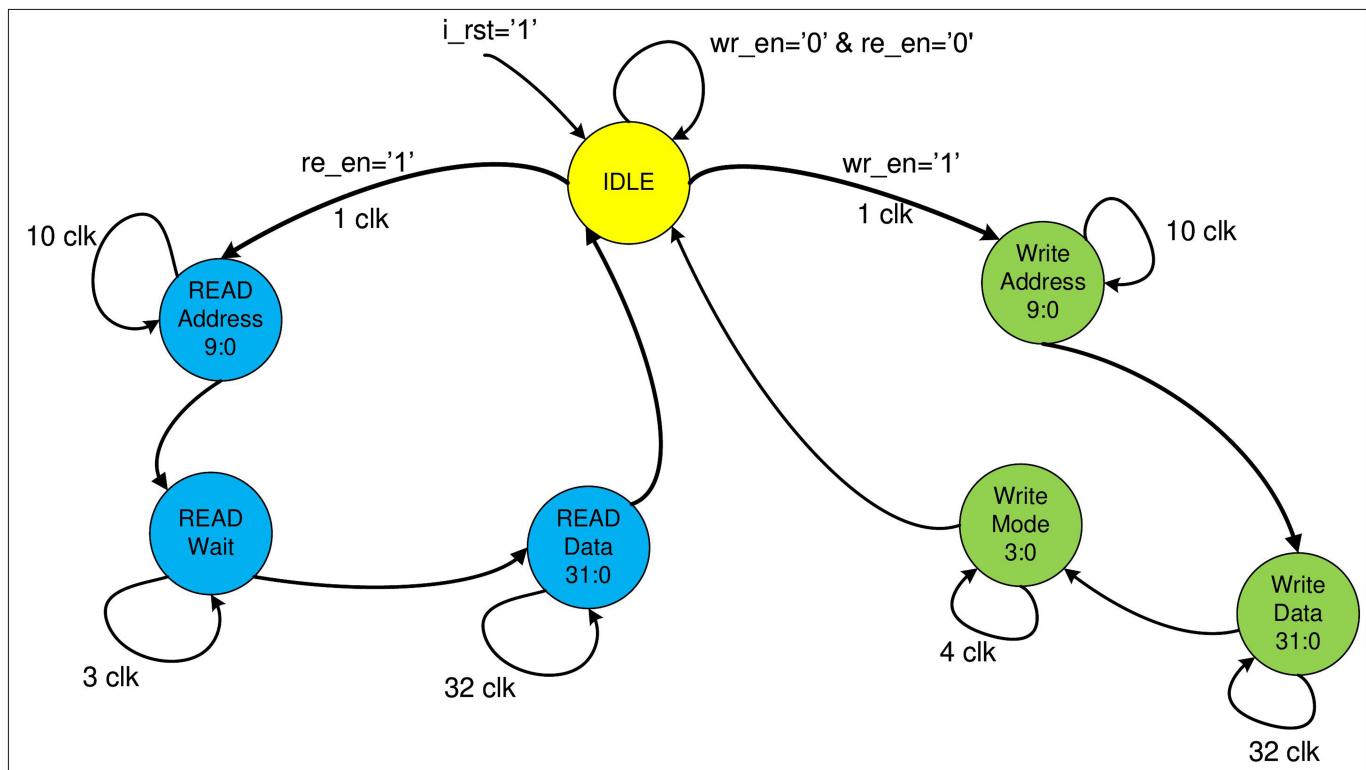
## External Pads

### SPI Ports

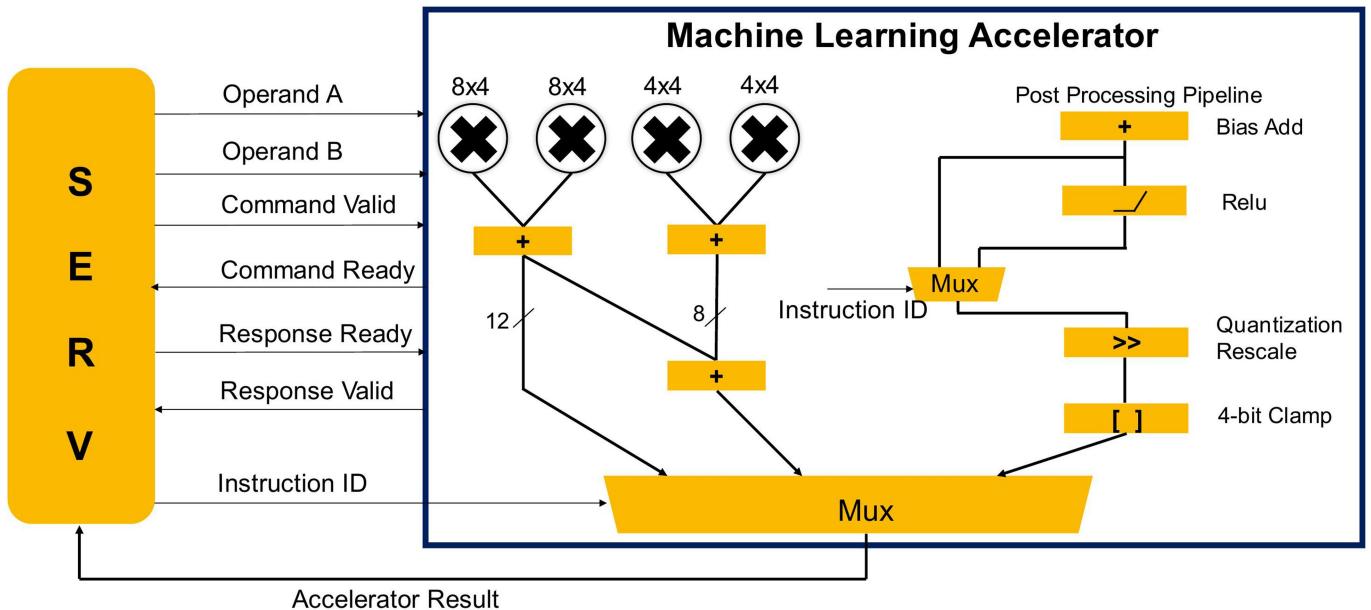


**Extended Data Fig. 1 | C-SPI.** It shows the diagram of the C-SPI block that provides the interfacing of the 79 RAM internal ports to 4 SPI ports. C-SPI also generates two gated clocks: one to stall the entire Flex-RV microprocessor, and another one to signal an external controller (e.g., an FPGA board).

## Article

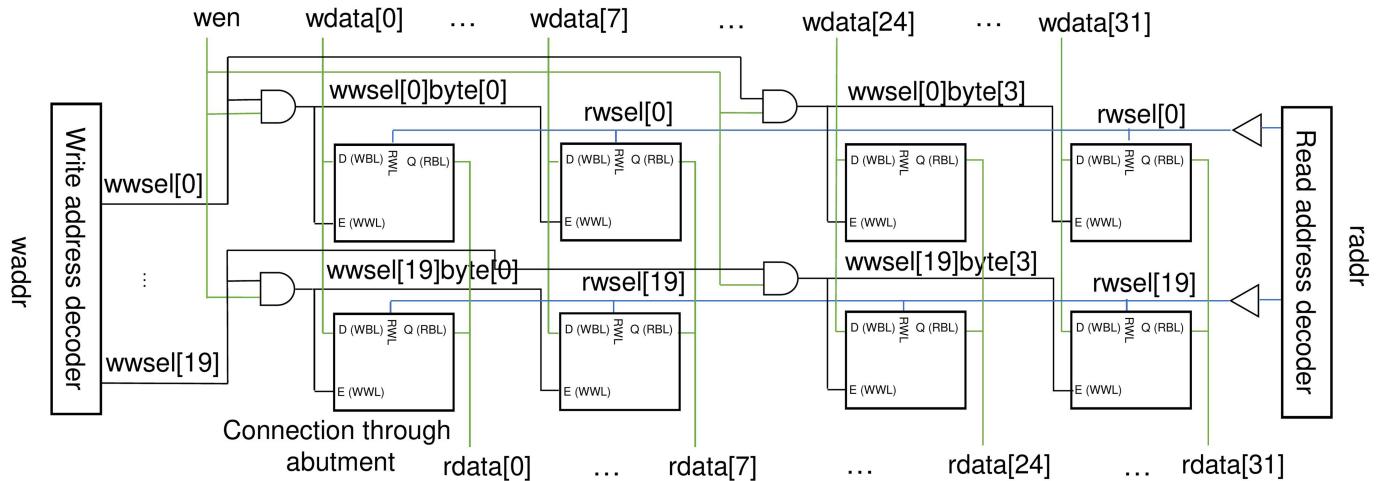


**Extended Data Fig. 2 | C-SPI State Machine.** Resetting to Idle state and progressing through sequential states upon receipt of R/W request.

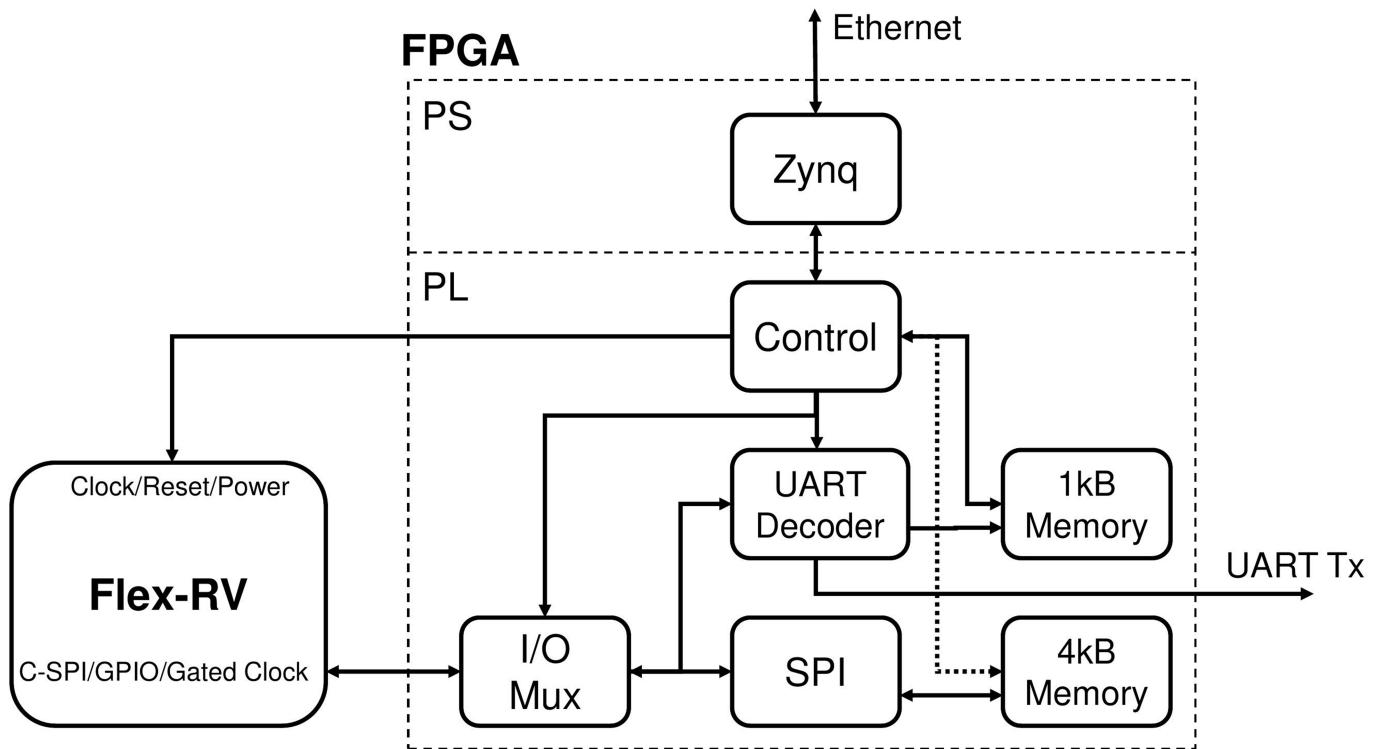


**Extended Data Fig. 3 | ML Accelerator Microarchitecture.** The ML accelerator is tightly coupled to *Serv*'s datapath to behave as a custom functional unit of *Serv*. *Serv* sends the custom instructions to run on the accelerator, which returns the result back to *Serv*.

# Article

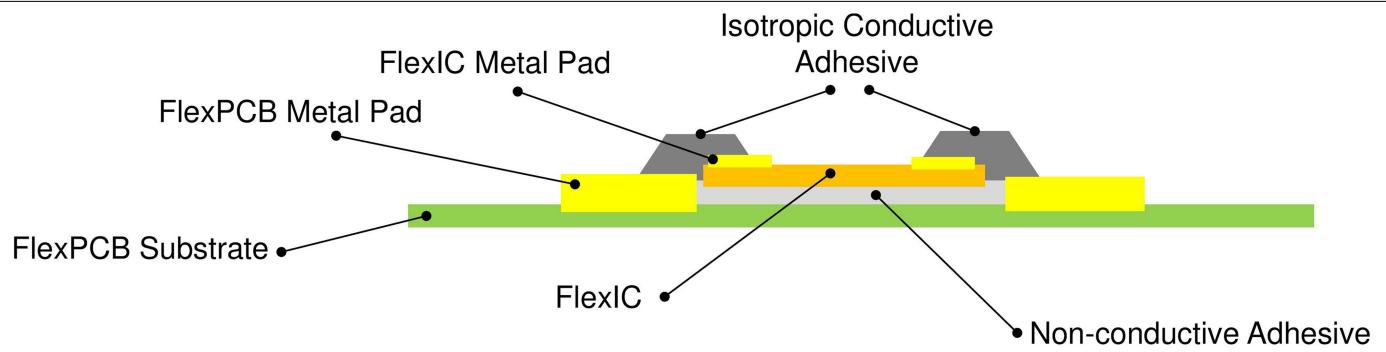


**Extended Data Fig. 4 | Register file inside the microprocessor.** The dual-port (one read and one write) latch-based register file is built by the abutment method.



**Extended Data Fig. 5 | FPGA set-up.** The block diagram of the FPGA-based test infrastructure is shown. FPGA controls all the test protocol to test Flex-RVs.

## Article



**Extended Data Fig. 6 | OEP assembly.** The cross-section of the OEP assembly process of assembling Flex-RV as a FlexIC onto a FlexPCB is shown.

---

**Extended Data Table 1 | C-SPI commands**

Write Mode	Data Portion in Memory Word
0001	Byte0
0010	Byte1
0100	Byte2
1000	Byte3
1111	All four bytes

Write modes and corresponding data portion written into the 32-bit memory word are shown.  
Either a byte or a word can be written.

# Article

## Extended Data Table 2 | Test benchmarks used to test Flex-RV

Test Benchmark Name	Description	Execution Time in seconds at 60kHz (FlexPCB Max Frequency)
hello_world	Reads from memory the characters "Hello World"	6.7
sum	Sums the numbers from 1 to 99	11
fibonacci	Computes 7 <sup>th</sup> Fibonacci number	2
find_max	Finds the maximum number in an array size of 16 numbers	3.3
find_min	Finds the minimum number in an array size of 16 numbers	2
find_mean	Finds the average of an array size of 16 numbers	2.7
intdiv	Divides two integer numbers	3.7
intmul	Multiplies two integer numbers	4.7
simd_matmul	Computes the dot product of two vectors (uses ML accelerator)	2.7
nn_inference_large_cfu_abnormal	Performs 1D convolution followed by a fully connected network on pre-processed ECG data signal using tiny neural network weights to detect abnormal rhythm (uses ML accelerator)	2.3
nn_inference_large_cfu_normal	Performs 1D convolution followed by a fully connected network on pre-processed ECG data signal using tiny neural network weights to detect normal rhythm (uses ML accelerator)	2.3

At the end of the run, each test benchmark sends the results through the GPIO pin.