HyCUBE: A 0.9V 26.4 MOPS/mW, 290 pJ/op, Power Efficient Accelerator for IoT Applications

Bo Wang, Manupa Karunarathne, Aditi Kulkarni, Tulika Mitra, Li-Shiuan Peh
School of Computing, National University of Singapore

***Abstract*—IoT devices use ultra-low-power micro-controllers that cannot handle the performance demands of emerging compute-intensive applications. Accelerators can be added to improve system power-performance efficiency. We present HyCUBE, a Coarse-Grained Reconfigurable Array (CGRA) accelerator chip that realizes 127× improvement in power efficiency compared to TI Sensortag IoT platform. HyCUBE has a bufferless Network-on-Chip (NoC), enabling single-cycle data traversal to boost throughput and a software-scheduled architecture, automatically extracting application parallelism. Our 40nm test chip delivers peak efficiency of 26.4 MOPS/mW with 290 pJ/operation, realizing a power efficiency improvement of 28.6× and 26.5× compared to Xilinx Zynq FPGA and ARM Cortex-A7 core.**

Keywords*— *CGRA; accelerator; IoT; power efficient

I. Introduction

Internet of Things (IoT) place enormous performance-per-mW demands on the devices that have to run diverse IoT applications ranging from face recognition [1] to healthcare diagnostics [2]. Among IoT applications, high parallelism and loop-intensive processing are increasingly prominent. However, the ultra-low-power processors in IoT devices have difficulty keeping up with the applications' demands. For instance, Fig. 1 shows a heart rate monitoring application, TROIKA [3], running on a wearable platform, TI Sensortag with ARM Cortex-M3 core. As profiled, TROIKA is a loop intensive application where the innermost loops from the two kernels consume 37.9% and 37.5% of the total execution time, respectively. However our measurements show that the Sensortag can only support up to 48MHz frequency while consuming 350mW when running TROIKA, thus delivering a peak efficiency of 0.21 MOPS/mW, which do not suffice to support such a throughput critical application.

CGRAs are promising accelerators for IoT devices. Unlike FPGAs, where bit-level reconfiguration leads to high area and power overheads, CGRAs support much more efficient configuration at word level. Prior CGRA architectures have demonstrated good power efficiency. For instance, Samsung's SRP CGRA [4] achieves 22.56 MOPS/mW efficiency in post-layout simulation. The works in [5], [6], [7] rely on large number of PEs for acceleration, which is not practical for IoT devices which are area constrained. Besides, this leads to low PE utilization and poor power efficiency.

Enhancing PE utilization and throughput is critical for realizing power-efficient CGRAs. This paper presents the HyCUBE CGRA accelerator chip targeting IoT applications. It has two key features: (1) A low-power NoC that offloads communications from the ALUs, delivering data across the chip within a single cycle, improving PE utilization and throughput, (2) A software-scheduled architecture extracting parallelism and orchestrating the cycle-by-cycle schedule of the PEs and NoC, enabling lightweight hardware design. Our 40nm test chip shows a power efficiency of 26.4 MOPS/mW with energy consumption of 290 pJ/operation, which is on average 127.5×, 28.6× and 26.5× higher than Sensortag, Xilinx Zynq FPGA and ARM A7 core, respectively.

(a)

TROIKA kernel	Loop iteration*	Share of run time
MPInverse	8000K	37.9%
FOCUSS	16000K	37.8%

*Iteration is calculated based on the innermost loop

(b)

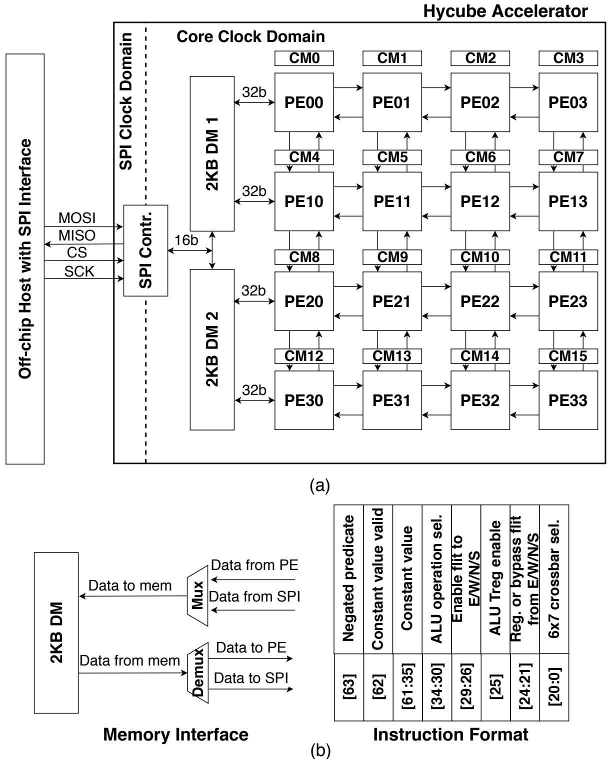
Fig. 1. (a) TI Sensortag IoT platform running TROIKA and (b) profiling of the innermost loops in two kernels.

II. HyCUBE Accelerator Design

A. Execution Model

Fig. 2(a) shows the architecture of HyCUBE, which consists of a 4 × 4 PE array, 2 dual-port data memories (DMs), 16 configuration memories (CMs) and an SPI interface for communication with host. HyCUBE is statically scheduled, thus the compiler [8] determines the PE and the cycle where each operation is scheduled. The instructions are encoded with routing information that can be used for communication, concurrently to the computations. Inter-PE communication is realized by crossbar switch circuits and links between the PEs, which forms a bufferless NoC. Moreover, HyCUBE's unique ability to communicate to distant PEs in a single cycle is enabled through the NoC that can bypass intermediate nodes in multi-hop path. The compiler thus exploits such connections to increase instruction-level parallelism (ILP) in kernels.

Initially, the compiler analyzes the control and data dependencies of the kernels and constructs a control-data dependency graph (CDFG). Then compilation models architecture



(a)

(b)

Fig. 2. (a) Architecture of HyCUBE accelerator and (b) its memory interface with instruction format

as a time-extended resource graph – MRRG[9], that includes single-cycle multi-hop connectivity. Finally, the compiler maps the CDFG to the MRRG exploiting higher degree of ILP and generates instruction streams for configuration.

Once the instruction and data are ready, they are loaded via SPI to memories. Thereafter, each PE will read its CM and execute the instruction. After the execution, the host reads back the computed data, again using the SPI.

B. Detailed Architecture of HyCUBE Accelerator

As Fig. 2(b) shows, the memory interfaces with both SPI and PEs so initial data and configuration loaded from SPI can be propagated to all PEs. HyCUBE has a custom 64b instruction format where crossbar selection, bypass enabling and ALU operations can be configured (Fig. 2(b)). The accelerator has two clock domains, a core clock domain including PEs and memory, and an SPI clock domain synchronized by the host. When crossing the domains, the 16b data from memory is buffered in a FIFO before being accessed by SPI and vice versa. The 2KB DM bank is organized as 512 × 32 where each 32b word can be partially updated by an 8b data chunk to accommodate 1B/2B/4B data sizes. Each 192B CM is able to store 24 instructions to orchestrate PE execution. A PE either executes computation or helps to bypass the received data to a distant PE. When computation completes, the DMs can be updated by the leftmost PEs where all other PEs have to transfer the output to one of them at first.

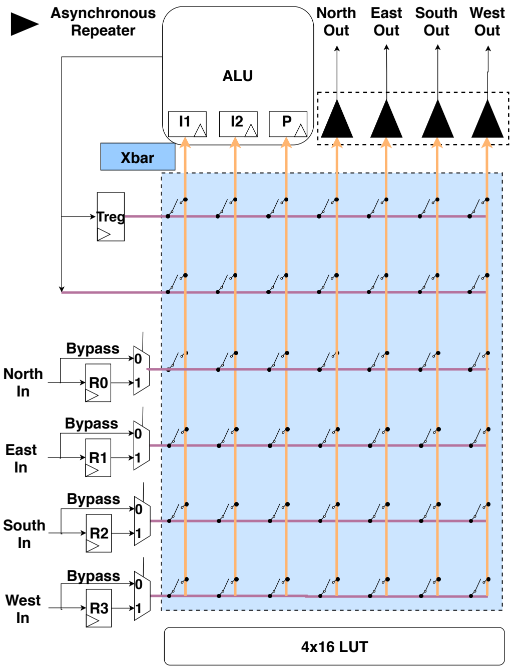


Fig. 3. Circuit diagram of HyCUBE tile

C. Tile design

To minimize critical path, each PE is grouped with its NoC crossbar switch to form a tile. A tile propagates flits to its neighbours for communication. A flit word incorporates 32b data and a predication bit P for control divergence. The circuit diagram of tile is depicted in Fig. 3. It comprises 4 input registers (R0∼R3), a 6×7 crossbar circuit (Xbar), an ALU and a lookup table (LUT).

The registers cache an incoming flit from N/S/E/W direction if it is locally consumed. Otherwise the flit bypasses the register and is directly routed to a neighbour PE without buffering. The crossbar is able to select incoming data from neighbours and eject it to N/S/E/W/PE direction. It can eject 2 operands ($I1$, $I2$) and the predication bit (P) to ALU for computation. The ALU then sends the result to Xbar or a single register $Treg$. A tiny LUT maintains the start and end locations of loops so the loop pointer can access easily for innermost loop acceleration. The output channel is connected with asynchronous repeaters (buffers). Conventional NoCs employ output registers so that flits can be buffered for each hop. As HyCUBE offloads flow control to the compiler, buffers and output registers are no longer necessary. Hence, the data path from ALU to output is no longer re-timed and the propagation is not terminated until the flit is latched by a tile. This facilitates traversal across multiple hops within a single cycle [10], which will be elaborated in Section D.

Fig. 4 plots the simulated power breakdown of the tile circuit. The result shows the NoC crossbar and the ALU dissipating 36.3% and 14.7% of total power, respectively.

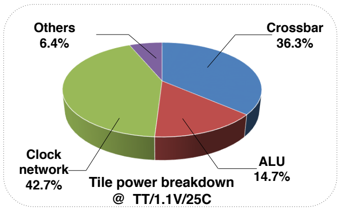


Fig. 4. Power breakdown of a tile circuit

D. Software-scheduled NoC for single cycle traversal

The statically-scheduled HyCUBE chip comes with a compiler tool chain which takes C programs as input, runs profiling to identify frequently occurring kernels, then automatically extracts parallelism as CDFGs and schedules them onto the chip, storing the cycle-by-cycle scheduling into the 16 CMs. The NoC is then orchestrated cycle-by-cycle by the CMs. Each cycle, HyCUBE PE reads a 64b instruction which encodes the configuration for ALU and crossbar. This static scheduling obviates the power and area overheads consumed by scheduling, route computations and flow control.

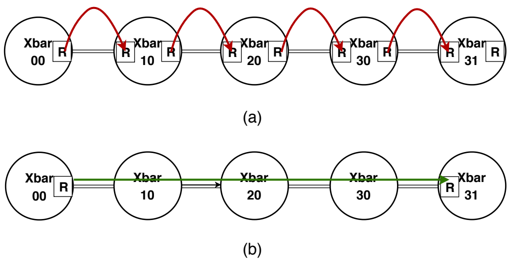


Fig. 5. 4-hop paths in (a) conventional NoCs and (b) HyCUBE NoC

In conventional CGRAs, it takes multiple cycles for data to travel between two distant PEs as data is registered every time when reaching a PE. Fig. 5(a) shows a 4-hop data path between PE00 and PE31. The intermediate nodes, PE10∼PE30 are merely used for communication rather than computing, leading to low PE utilization and throughput. Fig. 5(b) shows the output from PE00 is directly sent to PE31 in a single cycle in HyCUBE, forming a critical path through the intermediate hops. As HyCUBE NoC cancels data re-timing along PE10∼PE30, it can use all the nodes for compiler mapping after one cycle. Fig. 6 compares the critical path (4-hop latency) in HyCUBE against that in conventional NoCs. While it prolongs the cycle time, it shortens the 4-hop latency by 47.5%. As 4-hop is able to cover the chip edge, HyCUBE constrains the maximum number of hops to 4 as trade-off between throughput and frequency. The timing loops occurring at place-and-route can be broken up with appropriate configuration of backend tools. As prior CGRAs [4][5][6][7][11] only published simulation results, we compare them with HyCUBE's post-layout results at nominal voltages based on the FFT benchmark in Table I. HyCUBE throughput (MOPS) is calculated by multiplying operations per cycle by frequency, while the efficiency (MOPS/mW) is obtained by dividing throughput by power. HyCUBE achieves a throughput of 6482.8MOPS and power efficiency of 90 MOPS/mW at 1.1V, 4× and 1.4× as high as [4] and [11], respectively. Fig. 7 shows the simulated power breakdown of the CGRA circuit. The PE array and memory consume most of the power (64% and 23.6%, respectively) while the link (sans crossbar) only dissipates 0.4% of the total power.

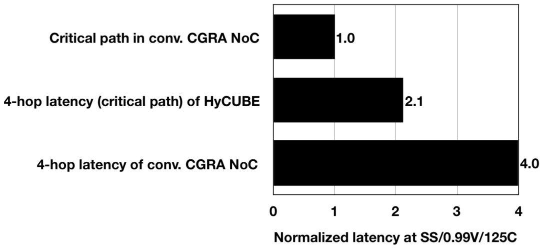


Fig. 6. Comparison of 4-hop latency in conventional NoC and HyCUBE.

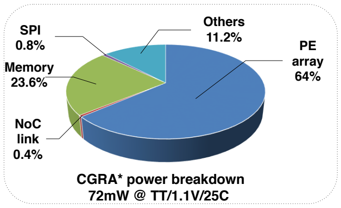


Fig. 7. Power breakdown of HyCUBE CGRA circuit

TABLE I
COMPARISON OF SIMULATED POWER EFFICIENCY ON VARIOUS CGRAS

	SRP[4]	REMUS[5]	RAA[6]	BilRC[7]	DRRA[11]	This work
Technology	40nm	65nm	180nm	90nm	65nm	40nm
VDD (V)	1.1	1.2	1.8	1.0/1.2	1.2	1.1
PE count	9	256	64	39	12	16
FFT benchmark	256-pt	256-pt	256-pt	1024-pt	2048-pt	256-pt
Freq. (MHz)	100	200	70	492	450	853
Throughput (MOPS)	467[1]	NA	NA	8757.6	~5395	6482.8
Power (mW)	20.7	NA	~185	NA	~135.2	72
Power efficiency (MOPS/mW)	22.56	NA	NA	NA	39.9	90
Norm. efficiency[2] (MOPS/mW)	22.56	NA	NA	NA	64.8	90
Core area (mm^2)	NA	21.6	NA	3.67	NA	2.87

1. 467 MOPS in SRP work is estimated from published 467 MIPS. Real MOPS can be less
2. Norm. efficiency is calculated as performance scaling up to 40nm

III. MEASUREMENT RESULTS

The HyCUBE test chip was fabricated in 40nm CMOS technology with core area of 2.86 mm^2. Fig. 8 shows the

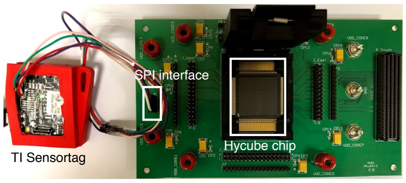
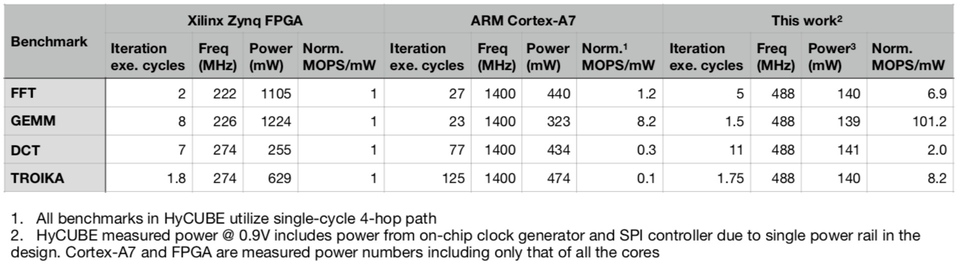


Fig. 8. Prototype of HyCUBE chip interfacing with TI Sensortag

TABLE II

CHIP MEASUREMENT COMPARISON OF PERFORMANCE-PER-MW



Benchmark	Xilinx Zynq FPGA				ARM Cortex-A7				This work[2]			
	Iteration exe. cycles	Freq (MHz)	Power (mW)	Norm. MOPS/mW	Iteration exe. cycles	Freq (MHz)	Power (mW)	Norm.[1] MOPS/mW	Iteration exe. cycles	Freq (MHz)	Power[3] (mW)	Norm. MOPS/mW
FFT	2	222	1105	1	27	1400	440	1.2	5	488	140	6.9
GEMM	8	226	1224	1	23	1400	323	8.2	1.5	488	139	101.2
DCT	7	274	255	1	77	1400	434	0.3	11	488	141	2.0
TROIKA	1.8	274	629	1	125	1400	474	0.1	1.75	488	140	8.2

1. All benchmarks in HyCUBE utilize single-cycle 4-hop path
2. HyCUBE measured power @ 0.9V includes power from on-chip clock generator and SPI controller due to single power rail in the design. Cortex-A7 and FPGA are measured power numbers including only that of all the cores

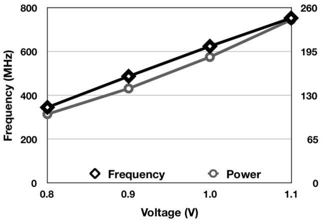


Fig. 9. Measured frequency and power of HyCUBE chip

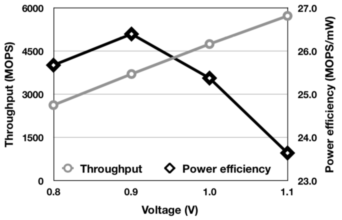


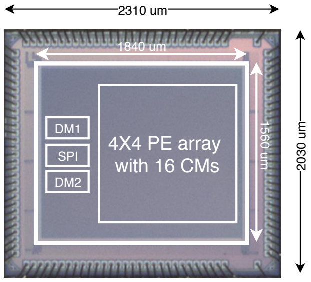
Fig. 10. Measured throughput of HyCUBE chip

prototype of the HyCUBE accelerator interfacing with the host, Sensortag via SPI.

Fig. 9 depicts the operating frequency with voltage scaling. The frequency varies from 753MHz to 346MHz when the supply goes from 1.1V to 0.8V. Fig. 10 plots the measured throughput and power efficiency. The HyCUBE chip achieves the maximum throughput of 5380MOPS at 1.1V and the minimum of 2630MOPS at 0.8V. The peak power efficiency is observed at 0.9V, as high as 26.4MOPS/mW with an energy of 290 pJ/op based on the FFT benchmark. For the TROIKA application, the chip (24.8MOPS/mW at 0.9V) outperforms 117×, 7.2× and 81.7× compared to TI Sensortag (0.21MOPS/mW), Xilinx Zynq FPGA (3.02MOPS/mW) and ARM Cortex-A7 core (0.3MOPS/mW), respectively. We choose ARM Cortex-A7 core for comparison as it was developed for wearable and IoT devices. Further peak power efficiency comparison (Table II) is made among 4 benchmarks. From the table, HyCUBE improves power efficiency by 28.6× and 26.5× on average compared to Zynq FPGA and Cortex-A7 core, respectively. Note that the power of FPGA and Cortex-A7 are underestimated as they merely include core power whereas HyCUBE includes entire on-chip power. Fig. 11 shows the die photo with summary.

IV. CONCLUSION

We propose HyCUBE, a novel CGRA chip with 16 PEs, where communication is handled by a bufferless NoC that is scheduled and controlled by the accompanying compiler, leading to high PE utilization at low power. Chip measurements show the HyCUBE chip has power efficiency improvement of 127.5×, 28.6× and 26.5× compared to the commercial IoT platform TI Sensortag, Xilinx Zynq FPGA and ARM A7 core, respectively.



Technology	40nm LP 1P10M
Core area	1.84mm x 1.56mm
Total DM size	4KB
Total CM size	3KB
No. of PE	16
Data width	32b fixed-point
Supply Voltage	0.8V ~ 1.1V
Core frequency	346MHz ~ 753MHz
Throughput	2.63GOPS ~ 5.38GOPS
Power	102mW ~ 242mW

Fig. 11. Die photo of the HyCUBE chip with summary

REFERENCES

[1] S. Kodali *et al.*, “Applications of deep neural networks for ultra low power IoT,” in *ICCD*, pp. 589–592, Nov 2017.

[2] A. Limaye *et al.*, “HERMIT: A benchmark suite for the internet of medical things,” *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4212–4222, 2018.

[3] Z. Zhang *et al.*, “TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise,” *IEEE Trans. on Bio. Engineering*, vol. 62, no. 2, pp. 522–531, 2015.

[4] C. Kim *et al.*, “ULP-SRP: Ultra low power Samsung Reconfigurable Processor for biomedical applications,” in *FPT*, IEEE, 2012.

[5] J. Wei *et al.*, “An efficient implementation of FFT based on CGRA,” *ICCSNT*, vol. 2018-Janua, pp. 493–497, 2018.

[6] Y. Kim *et al.*, “Hierarchical reconfigurable computing arrays for efficient CGRA-based embedded systems,” *DAC*, 2009.

[7] O. Atak *et al.*, “BilRC: An execution triggered coarse grained reconfigurable architecture,” *IEEE Trans. on VLSI Sys.*, vol. 21 (7), 2013.

[8] M. Karunaratne *et al.*, “Hycube: A CGRA with reconfigurable single-cycle multi-hop interconnect,” in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2017.

[9] B. Mei *et al.*, “DRESC: A retargetable compiler for coarse-grained reconfigurable architectures,” in *2002 IEEE International Conference on Field-Programmable Technology, 2002.(FPT). Proceedings.*, pp. 166–173, IEEE, 2002.

[10] T. Krishna *et al.*, “Breaking the on-chip latency barrier using SMART,” in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 378–389, Feb 2013.

[11] N. Farahini *et al.*, “39.9 GOPs/watt multi-mode cgra accelerator for a multi-standard basestation,” in *ISCAS*, pp. 1448–1451, IEEE, 2013.