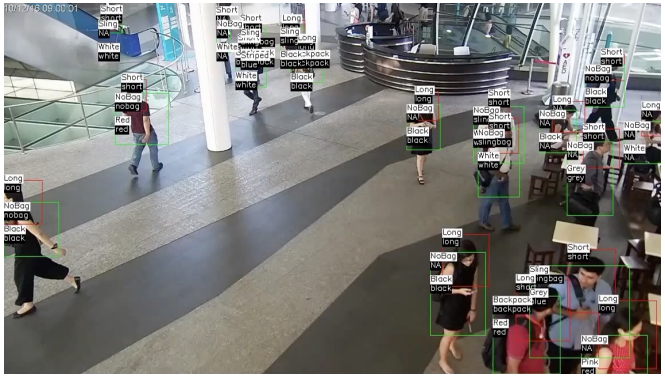
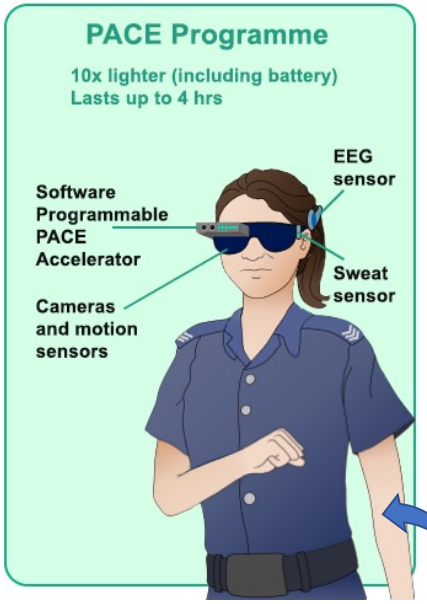
PACE: Enabling Multisensory Edge Analytics



Real-time event analytics for Singapore police

Institute for Infocomm Research



Circa: 2019

Circa: 2025

Mixed-reality event analytics for police
- Multimodal-multisensor interface
- Emotional, physiological state monitoring
- Orientation, position monitoring

Beyond deep neural networks
- Machine vision: scene understanding
- Navigation via visual-inertial odometry
- Continuous biometrics

PACE: 2 TOPS/Watt software programmable accelerator for intensive edge computing

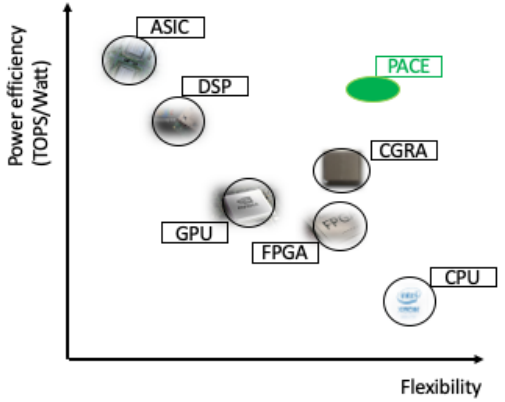
Edge Computing Hardware Landscape

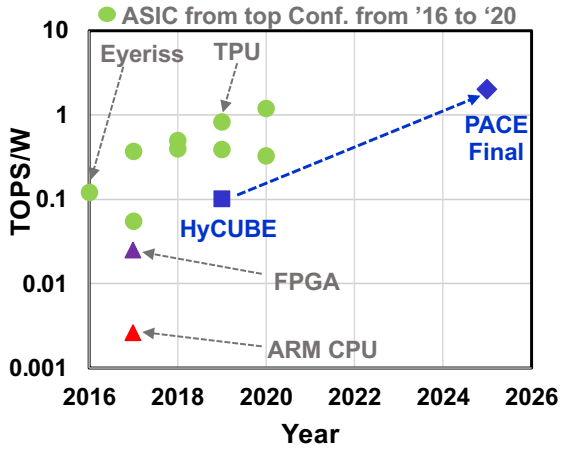
Existing solutions are either inefficient (CPU, GPU, FPGA) or specialized (Google TPU, MIT Eyeriss)
Multimodal-multisensory analytics on IoT edge device requires general-purpose efficient accelerator

	Performance	Power	Versatility	Programmability
ASIC Accelerator	👍	👍	👎	👎
CPU	👎	👎	👍	👍
GPU	👉	👉	👉	👉
FPGA	👉	👉	👍	👉
PACE	👍	👍	👍	👍

Gesture recognition requires < 8ms
- Tiny CPU: 577ms, 9 mW
- Smartphone CPU: 13ms, 469 mW
- NUS HyCUBE CGRA: 7ms, 100mW

PACE: Software-programmable accelerator to achieve the efficiency of today's specialized accelerators





PACE Innovation: Dataflow computing

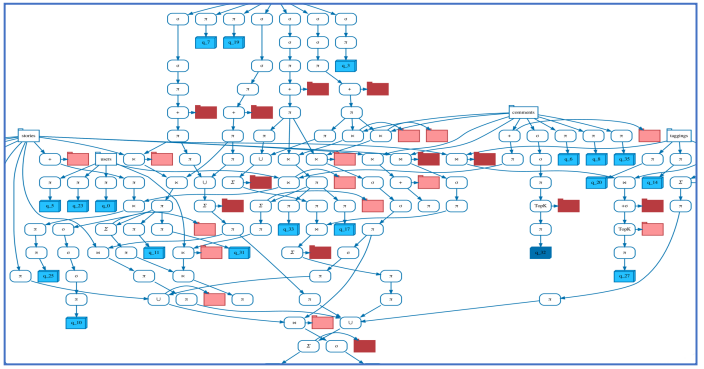
Move from sequential to parallel dataflow model

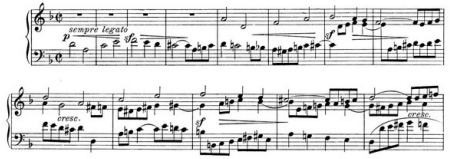
```
#include <stdlib.h>
int sub(int x, int y){
        return 2*x+y;
}

int main(int argc, char ** argv){
        int a;
        a = atoi(argv[1]);
        return sub(argc,a);
}
```

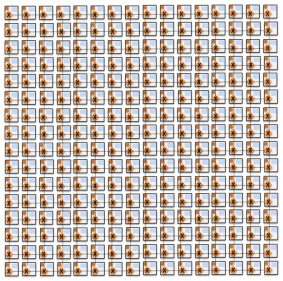
```
.text:00000000 _sub:      push   ebp
.text:00000001            mov    ebp, esp
.text:00000003            mov    eax, [ebp+8]
.text:00000006            mov    ecx, [ebp+0Ch]
.text:00000009            lea    eax, [ecx+eax*2]
.text:0000000C            pop    ebp
.text:0000000D            retn
.text:00000010 _main:     push   ebp
.text:00000011            mov    ebp, esp
.text:00000013            push   ecx
.text:00000014            mov    eax, [ebp+0Ch]
.text:00000017            mov    ecx, [eax+4]
.text:0000001A            push   ecx
.text:0000001B            call   dword ptr ds:__imp__atoi
.text:00000021            add    esp, 4
.text:00000024            mov    [ebp-4], eax
.text:00000027            mov    edx, [ebp-4]
.text:0000002A            push   edx
.text:0000002B            mov    eax, [ebp+8]
.text:0000002E            push   eax
.text:0000002F            call   _sub
.text:00000034            add    esp, 8
.text:00000037            mov    esp, ebp
.text:00000039            pop    ebp
.text:0000003A            retn
```

88





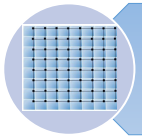
- Parallelism is fully exposed at software level
- Past dataflow architectures did not succeed as complex hardware determined when an instruction would execute



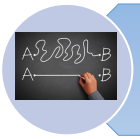
PACE: Dataflow computing with a twist

- Simple, parallel hardware: low power, high performance
- Smart software compiler spells out spatio-temporal schedule of instructions completely

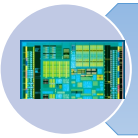
Pace Innovation: Let Software Define Hardware



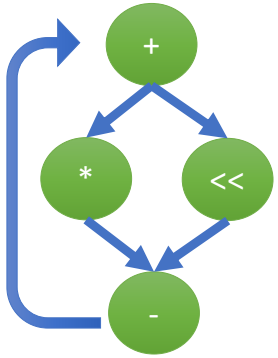
Embrace Parallelism

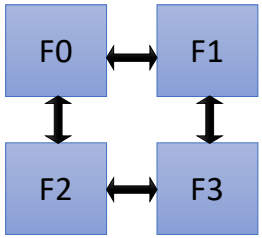


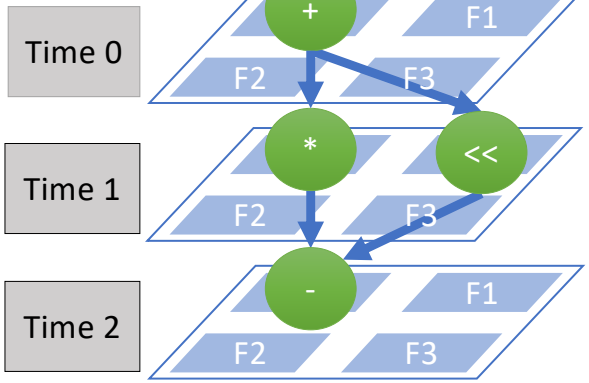
Simple Hardware



Expose to Software

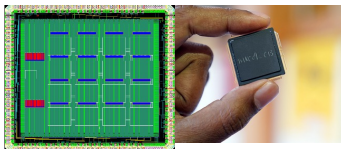


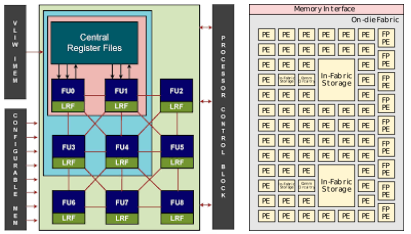




Coarse-Grained Reconfigurable Array (CGRA) at NUS

- Top international CGRA research group: PI Mitra, Peh
- Award-winning theoretically optimal CGRA Compiler
- Edge CGRA with highest compute-efficiency





SAMSUNG

Samsung Reconfigurable Processor

intel

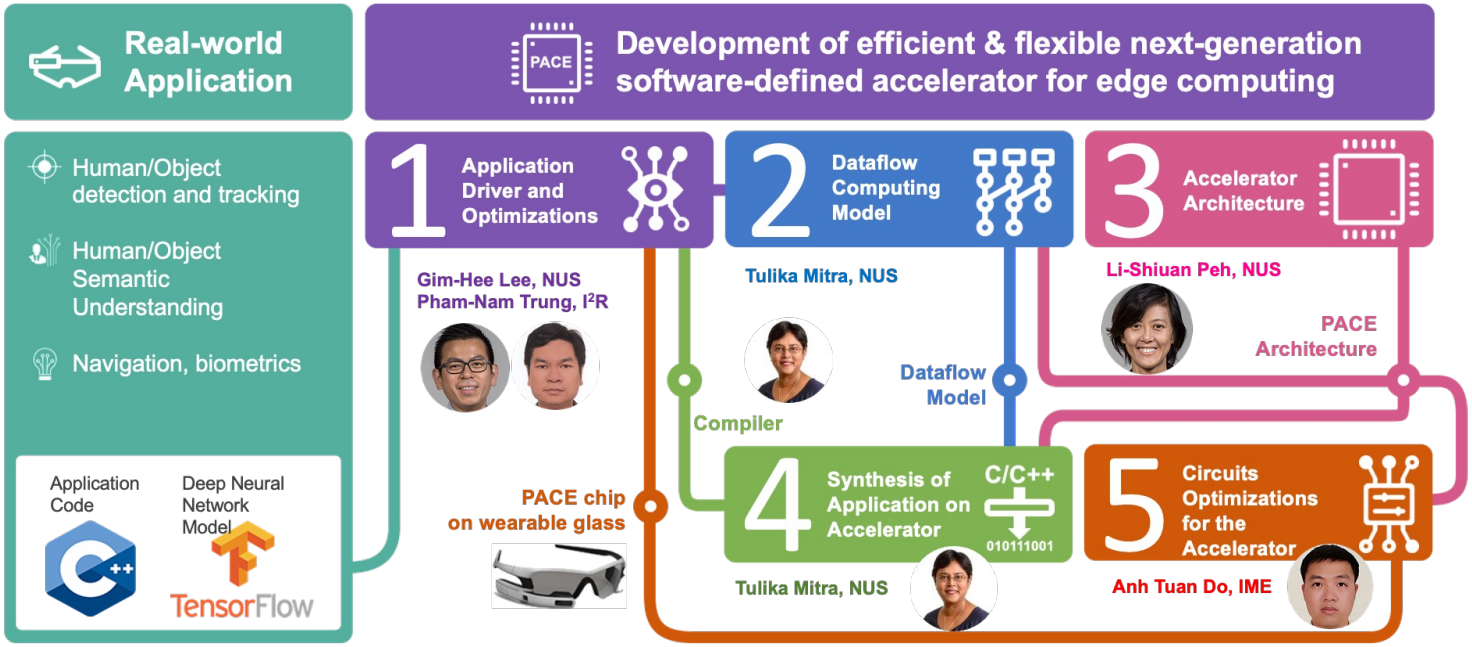
Configurable Spatial Accelerator

DARPA

Software-Defined Hardware Programme

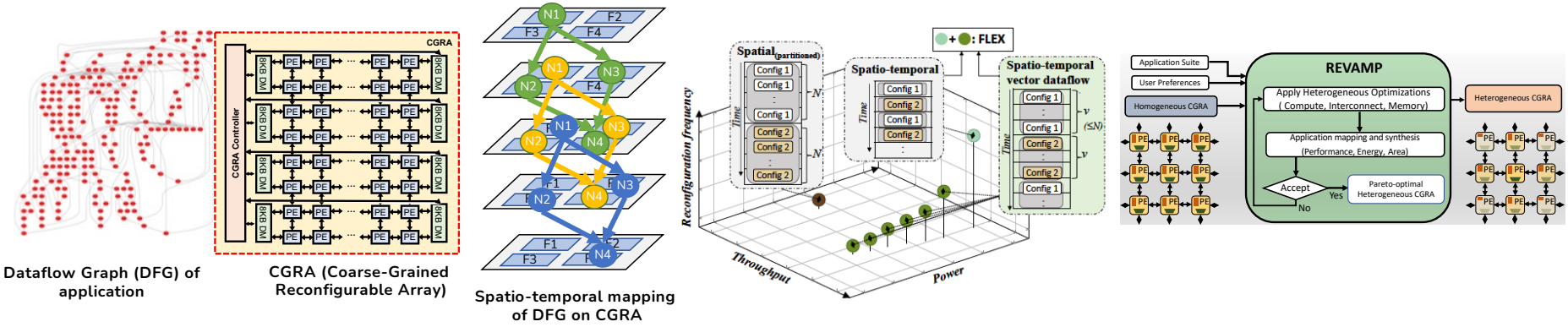
Poor compiler
Server vs. edge
Hardware vs. systems

PACE Programme: A synergistic Cross-Layer Approach



- Significant innovations in architecture, compiler, application optimizations to realize the software-defined edge accelerator vision
- Fully functional software-defined Coarse-Grained Reconfigurable Array (CGRA) accelerator chip with the highest energy efficiency
- World's first end-to-end software development framework for CGRAs with best quality mapping and shortest compilation time
- Demonstration of complete edge analytics applications (speech and gesture recognition) on PACE SoC development platform

PACE: Software-Programmable CGRA Accelerator
Dataflow Computing: Let Software Define Hardware



Dataflow Graph (DFG) of application

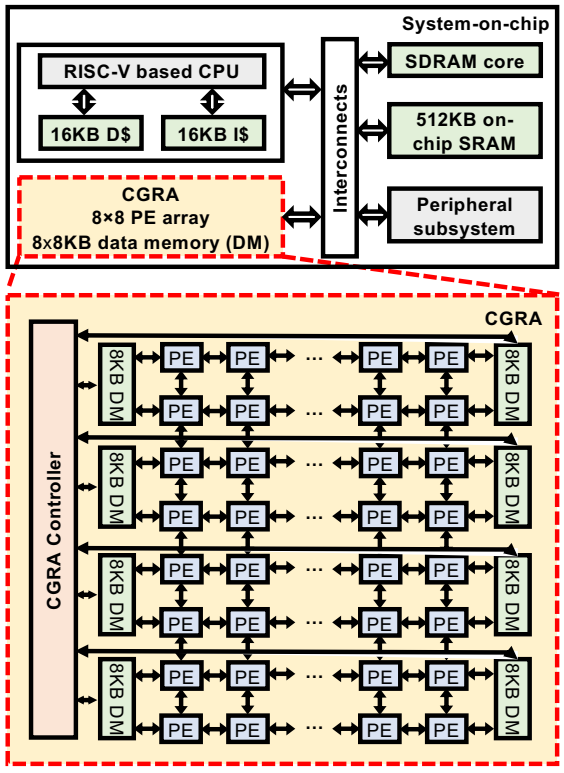
CGRA (Coarse-Grained Reconfigurable Array)

Spatio-temporal mapping of DFG on CGRA

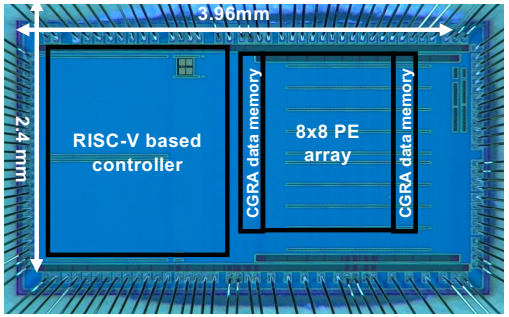
- PACE CGRA accelerator moves away from traditional sequential von Neumann model to inherently parallel dataflow computing model
- The hardware is a simple, low-power processing element (PE) array with small context memory to store limited reconfiguration information
- Compiler maps application dataflow graph (DFG) onto CGRA using per-cycle reconfiguration of compute and on-chip network
- PACE 1.0 accelerator chip achieves the highest energy efficiency among state-of-the-art CGRAs: 360 GOPS/W, 4.4mW at 0.6V, 40nm (measured) & 582 GOPS/W, 1.1mW at 0.45V (simulated, measurement on silicon in progress) through aggressive hardware-software codesign, power management of idle PEs, architectural innovations (single-cycle multi-hop network, software-defined memory hierarchy etc.)

Hardware: 64-PE CGRA SoC with RISC-V controller

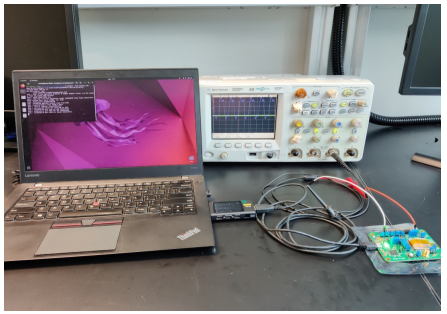
- We successfully designed, optimized and integrated our CGRA accelerator with a low-power RISC-V controller to create a complete system-on-chip (SoC) solution
- The SoC chip was designed and fabricated in 40nm CMOS process.



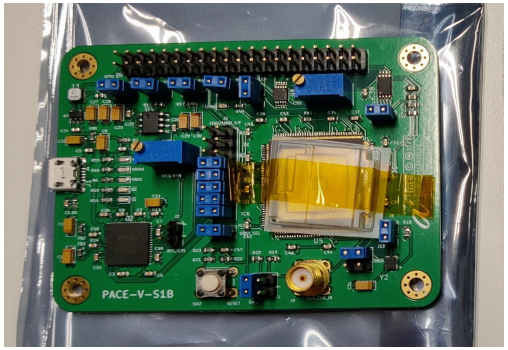
CGRA Integrated in SoC



Chip micrograph



Test set-up



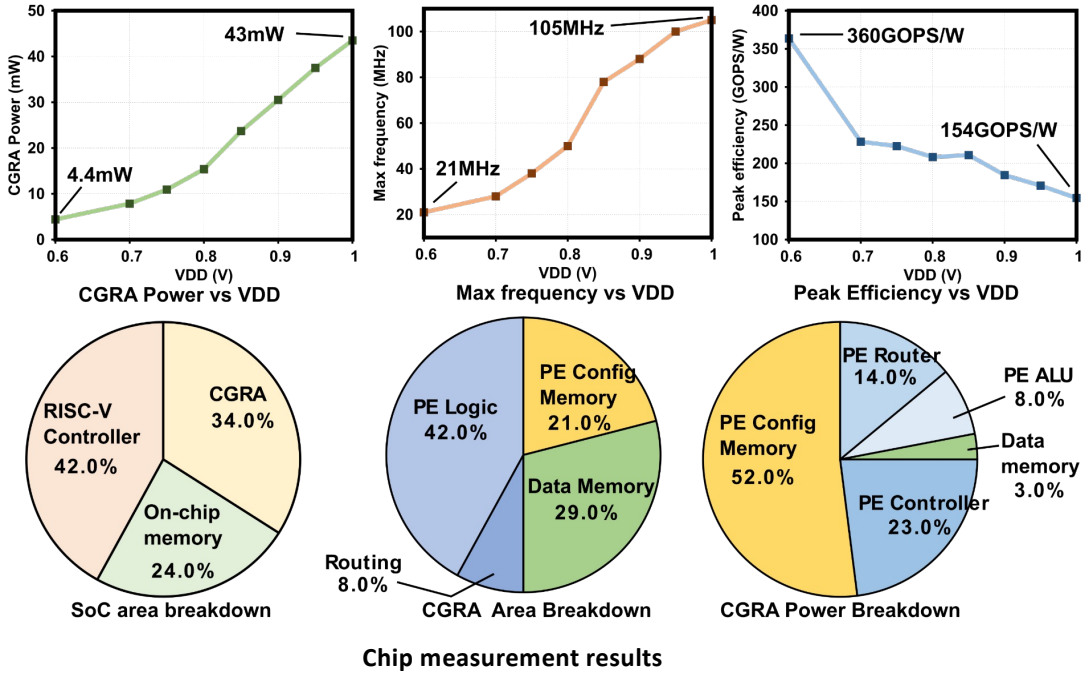
PACE assembled on PCB board

Design techniques

- Light-weight ALU (17 instructions)
- Configurable router supporting single-cycle multi-hop transfer.
- Static clock gating can be applied when inactive PEs are known beforehand, saving 70% power
- Dynamic clock gating by NOP instructions → 10.8% power reduction
- Block-based back-end implementation → improve timing closure, clock tree power

Hardware Evaluation: Measured 360 GOPS/W at 0.6 V, 21 MHz

- The whole SoC showed full functionalities when running micro-sized speech kernels with the supply voltage from 0.6V-1.2V.
- At 0.6V the CGRA accelerator consumes only 4.4 mW @ 21 MHz, offering an energy efficiency of 360 GOPS/W
- Work-in-progress to measure expected energy efficiency of 582 GOPS/W at 0.45V resulting in 1.1mW @ 10 MHz
- Breakdown of the power and area consumption of our fabricated chip are also shown below. Within the PE array, the configuration memory is the most power-hungry block because it is read-out every cycle to control the next operation of the ALU and the router.



Chip measurement results

Benchmarking against the state of the arts

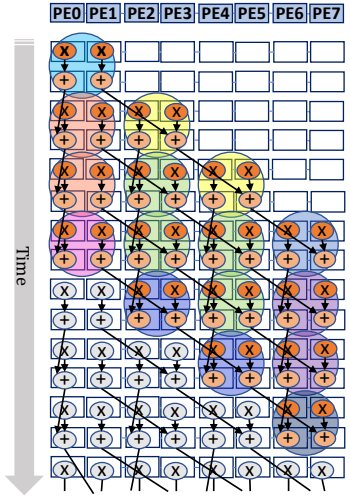
- The whole SoC showed full functionalities when running micro-sized speech kernels with the supply voltage from 0.6V-1.2V.
- PACE achieves 1.2x better efficiency as compared to the state-of-the-art
- At 0.6 V, 4.4mW @ 21 MHz, it offers the best measured peak energy efficiency of 360 GOPS/W. Our simulation shows that the chip can achieve 582 GOPS/W at 0.45V, 1.1mW @ 10MHz. Further test are still on-going towards this goal. This is equivalent to > 1 TOPS/W at 22nm or 28nm CMOS.
- In the next tape-out, we are planning to move to 28nm CMOS to better achieve the final goal of the program (i.e., 2 TOPS/W)

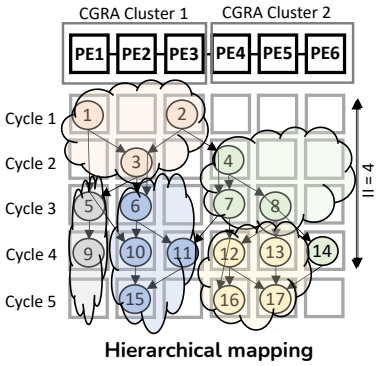
	Amber [1]	SSCL [2]	ISSCC [3]	TVLSI [4]	Hycube [5]	Snafu [6]	RipTide [7]	JSSC [8]	**This work (measured)**	**This work (simulated)**
Year	2022	2020	2019	2018	2019	2021	2022	2020	**2023**	**2023**
Tech (nm)	16	28	22	55	40	28	22	28	**40**	**40**
CGRA area (mm^2)	20.1	3.9	4.9	5.19	2.87	1.0	0.25	4.80	**3.02**	**3.02**
#PEs	384	120	15	30	16	36	36	64	**64**	**64**
Voltage (V)	1.29	0.9	0.8	--	1.1	N/A	N/A	0.9	**1.0/0.6**	**1.0/0.45**
Freq (MHz)	955	89	36	450	853	50	50	800	**100 21@0.6V**	**100 10@0.45V**
Throughput (GOPS)	367 (INT16)	14.1	145	77.4	6.48	N/A	62	0.88	**64**	**64**
Power (mW)	N/A	45.9	N/A	1526	72	0.32	0.24	537	**43@1V 4.4@0.6V**	**43@1V 1.1@0.45V**
Efficiency (GOPS/W)	538@ 1.29V (INT16)	307@ 0.9V	978@0.48V	50.8	90	305	254	196	**154@1V 360@0.6V**	**582@0.45V**
Memory	4500KB	234KB	690KB	54KB	7KB	256KB	256KB	320KB	**80KB**	**80KB**
Norm. area (mm^2)	50	5.5	3.2	3.67	2.87	2.0	1.0	6.86	**3.02**	**3.02**
Norm. area per PE (mm^2)	0.13	0.05	0.21	0.12	0.18	0.06	0.03	0.11	**0.05**	**0.05**
Norm. efficiency (GOPS/W)	86	150	296	96	90	151*	77*	96	**360**	**582**

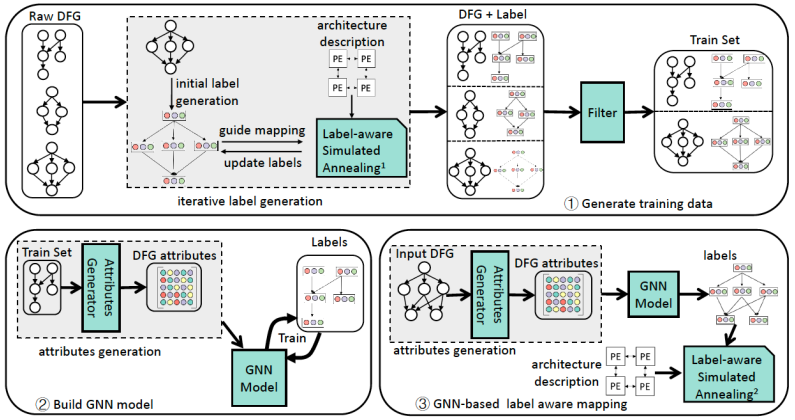
Norm. efficiency = efficiency $\times \left(\frac{\text{node}}{40\ nm}\right)^2$ Norm. area = area $\times \frac{40\ nm}{\text{node}}$ * simulated

High-Quality Dataflow Synthesis on CGRA

- Invented scalable and portable compiler techniques for high-quality dataflow synthesis
- HiMap (TCAD 2022), Panorama (DAC 2022): Scalable mapping of regular and irregular complex dataflow on larger CGRAs through hierarchical abstractions
  - Mapping quality: 17x improved performance, 5x improved energy-efficiency
  - Reduce compilation time from days to 15-mins
- LISA (HPCA 2022): Automated compiler generation on any new spatial accelerator architecture using graph neural network (GNN)
  - Better mapping quality than hand-crafted compiler, 17x reduction in compilation time

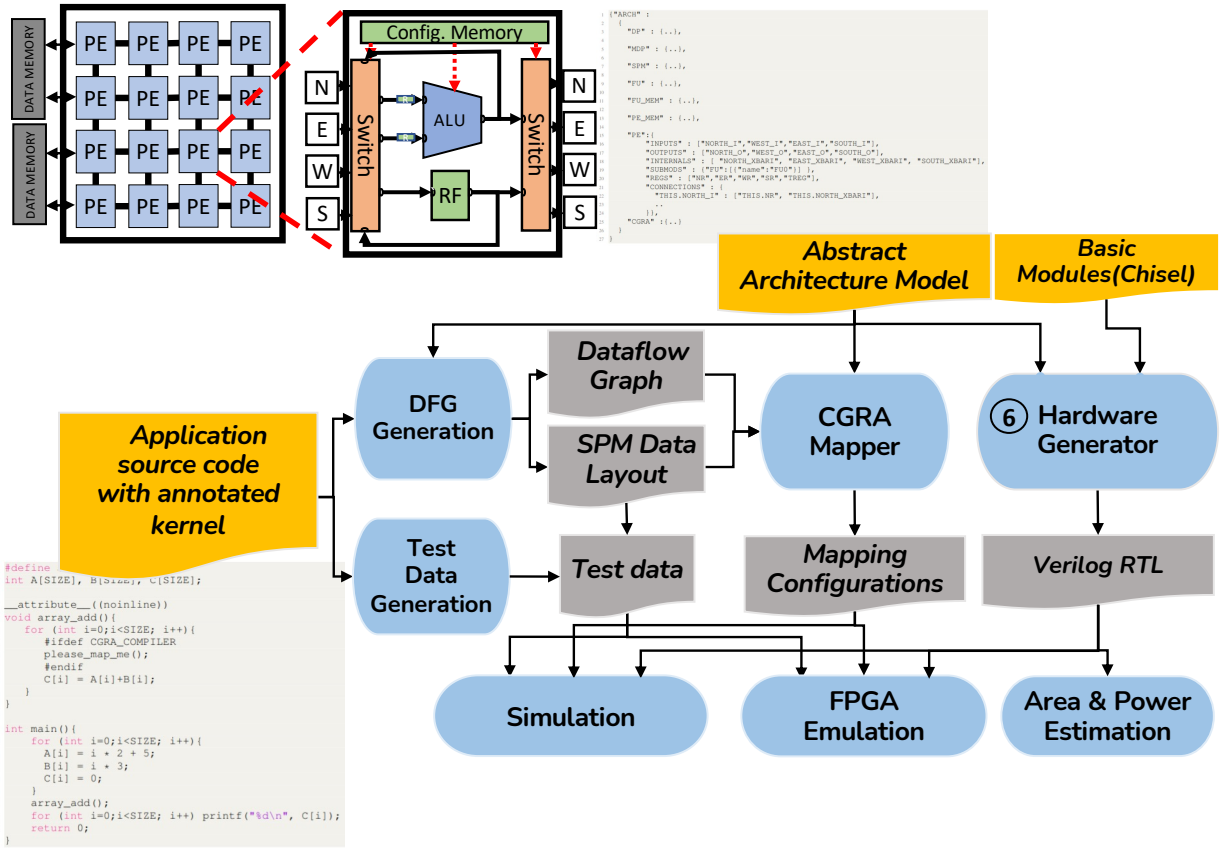






Morpher Open-Source CGRA Toolchain

SCAN ME

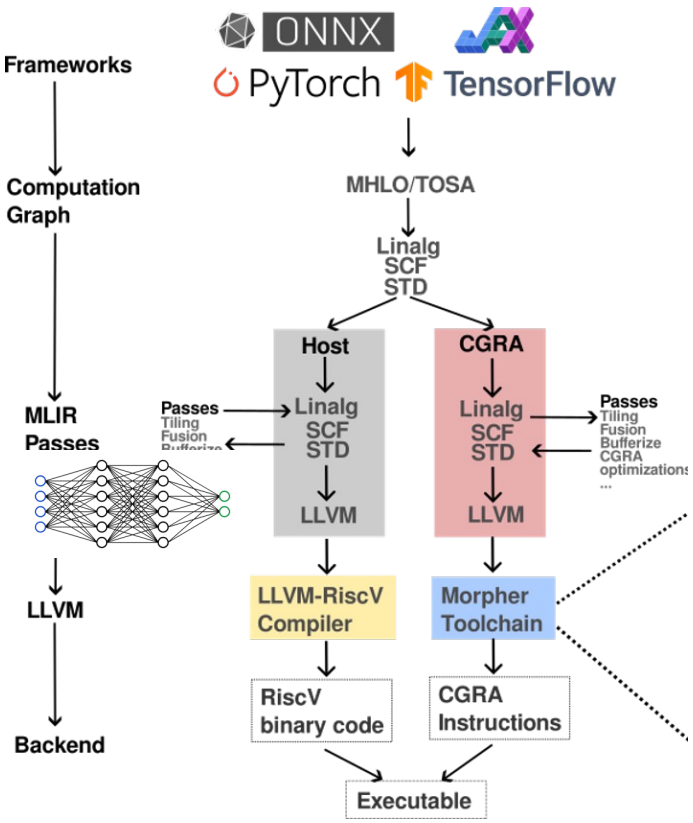


- Only comprehensive end-to-end software development framework for CGRAs with compiler, testing and validation capability, simulation, emulation, and hardware generation.
- Portable, scalable compiler providing best quality mapping & reducing compilation time from days to minutes.

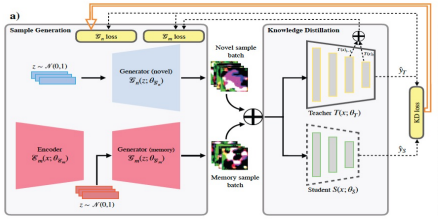
Features		CGRA-ME	Pillars	OpenCGRA	CCF	Morpher
DFG Generation	Models control divergence	✘	✘	✔	✔	✔
	Recurrence edges	✘	✘	✔	✔	✔
Architecture Modeling	Adapt user defined architectures	✔	✔	✔	✘	✔
	Multi-hop connections	✘	✘	✘	✘	✔
	Different memory organizations	✘	✘	✔	✘	✔
P&R Mapper	Architecture adaptive mapping	✔	✔	✘	✘	✔
	Data layout aware mapping	✘	✘	✘	✘	✔
	Recurrence aware mapping	✘	✘	✔	✔	✔
Simulation & validation	Cycle accurate simulation	✘	✔	✔	✔	✔
	Test data generation	✘	✘	✘	✘	✔
	Validation against test data	✘	✘	✘	✘	✔

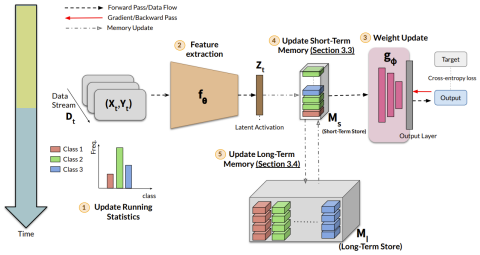
https://github.com/ecolab-nus/morpher

Edge AI Acceleration

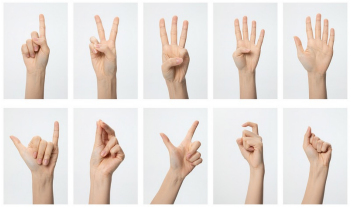


Compress large ML models (deep neural networks) to smaller models for inference on edge devices through robust data-free knowledge distillation (AAAI 2022), class-aware pruning and on-device learning (DATE 2023), as well as quantization

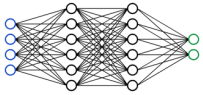




End2End framework that connects ML frameworks with Morpher toolchain to map complex applications on heterogeneous system (RISC-V + CGRA)

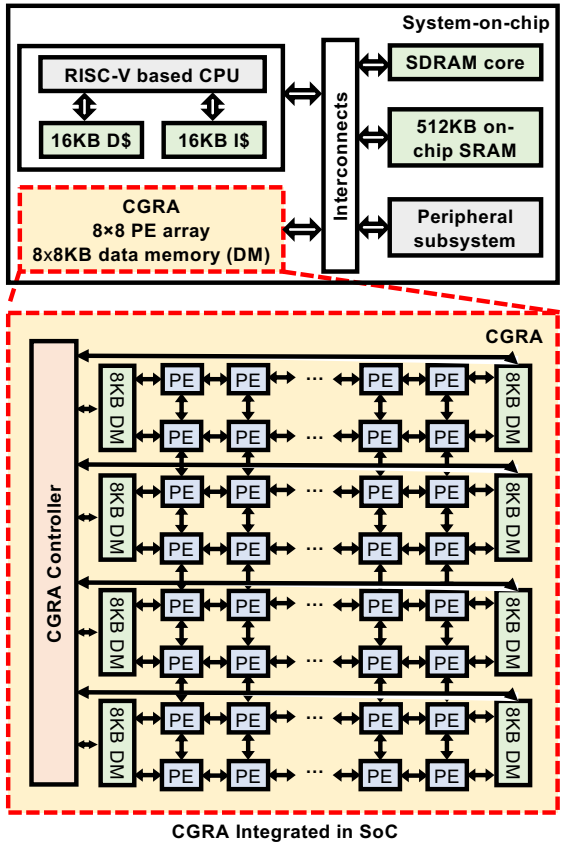


Gesture recognition application



Automated mapping and execution on PACE accelerator (RISC-V + CGRA)

PACE Hardware Design



CGRA Integrated in SoC

Objective:

- Design & implement coarse-grained reconfigurable array (CGRA) architecture for IoT computation workloads

Project Deliverables:

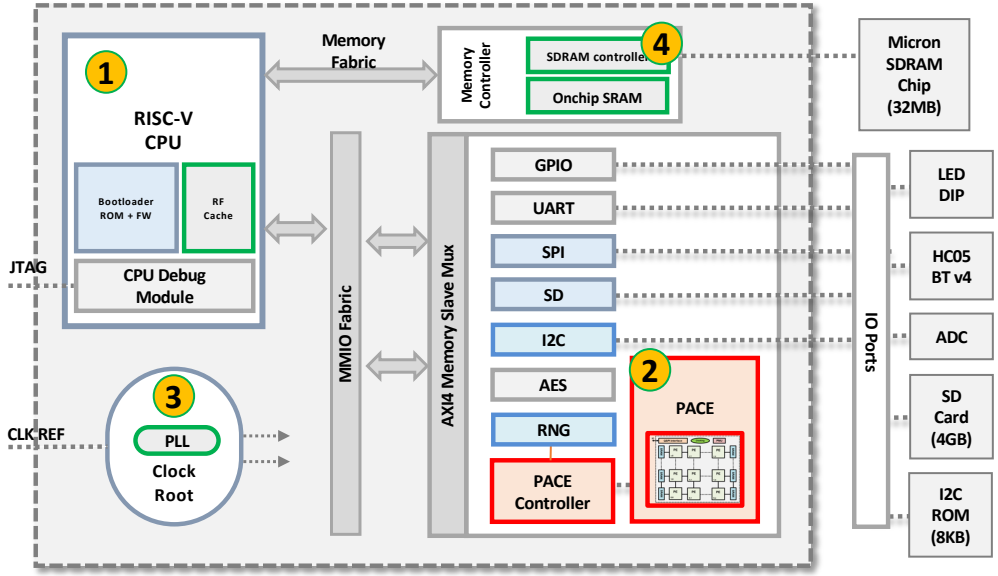
- FPGA-based demonstration/Verification of RTL design
- 2 tape-outs: TC1 (Homogeneous PE array for neural network & loops) & TC2 (Heterogeneous PE array with customized circuits for ULP mixed reality analytics)

PACE CGRA specifications:

- All data paths are 16-bit wide.
- Consists of an 8×8 array of PEs
- A total of 64KB CGRA data memories, each 8KB dual port SRAMs.
- Integrated within a RISC-V SoC for seamless data and instructions transfer

PACE-V SoC Architecture

- PACE-V SoC is a RISC-V based IoT processor with embedded CGRA for accelerating computing workload on Edge devices.
- The SoC is built in 40nm ULP for, aiming for high energy efficiency with rich communication interface (see on the right) and large on-chip SRAM to avoid off-chip memory access.
- The system clock can be supplied by either an integrated PLL (upto 400 MHz) or a 100 MHz off-chip crystal.
- We'll discuss the details of 4 main blocks in the next few slides



	PACE SoC Specifications	
Core and peripheral	Tech. Node	UMC 40nm ULP
	Supply Voltage	1V/3.3V
	Processor	RISC-V CPU (5 stage Rocket Core)
	Primary memory	Micron MT48LC64M4A2 SDRAM (32MB)
	Supported Comm.	GPIO×16, SD card, SPI×4 & UART
	Debug	JTAG
	On-chip SRAM	512 KB
	System clock freq.	400/200/100 MHz
	Core power	14mW @ 1.0V
	iCache/dCache	16KB / 16KB
	Custom bootloader	Yes, external EEPROM
	Operating Temp.	0^0C - 85^0C
PACE HW	Number of PEs	8x8
	PACE max clock freq.	T_{SIP}, 100 MHz
	Data memory	8 x 8KB dual-port SRAM
	Instruction set	17, including NoP
	Config memory	64 x 0.25KB single-port SRAM
	Interface	AXI MIMO (32 bits)

PACE CGRA Architecture

- 8×8 PE with 8 dedicated data memories. Each data memory will serve a group of 2×4 PEs.
- Each PE supports 17 instructions with 16-bit data (including NOP).
- All work at 100 MHz at nominal operation voltage of 1V.
- Individual PE can be statically turned off via the PACE controller or dynamically turned off using NOP opcode
- Equivalent to 166 TOPS/W (assuming 1 OPS & 1V supply)

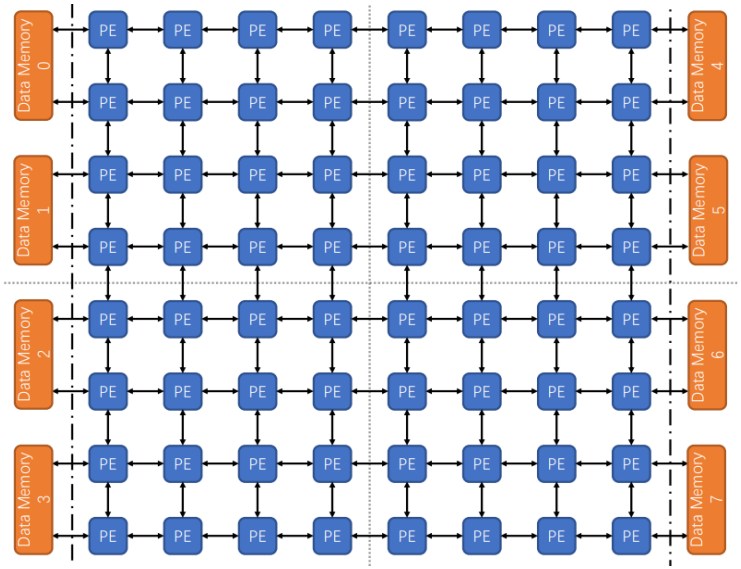
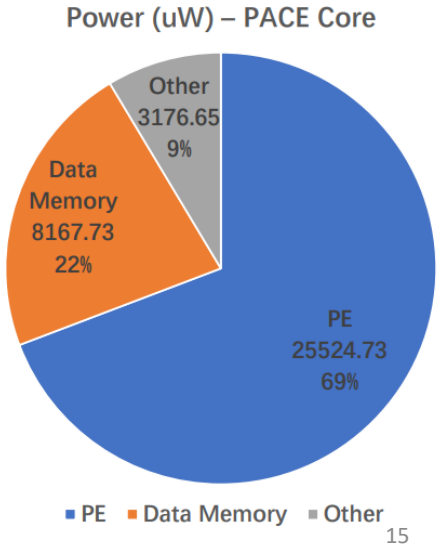


Table 3.3: Operations of the ALU module

Operation	Opcode	Description
NOP	5'b00000	No operation
ADD	5'b00001	Unsigned addition
SUB	5'b00010	Unsigned subtraction
MULT	5'b00011	Signed multiplication
LS	5'b01000	Left shifter
RS	5'b01001	Right shifter
ARS	5'b01010	Algorithm shifter
AND	5'b01011	Bitwise and
OR	5'b01100	Bitwise or
XOR	5'b01101	Bitwise xor
SEL	5'b10000	Operand selection
CMERGE	5'b10001	Operand selection
CMP	5'b10010	Equal-to comparison
CLT	5'b10011	Less-than comparison
BR	5'b10100	Multiple bitwise or
CGT	5'b10101	Greater-than comparison
MOVC	5'b11111	Operand move



Rocket CPU

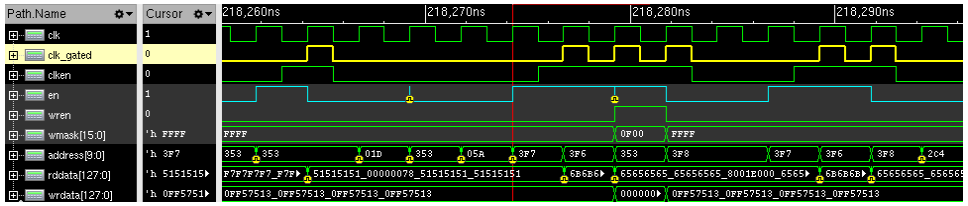
- RISC-V CPU parameters:

- Max frequency: **400MHz**
- **Single core** CPU (5 Stages, **RV32IMAFC** variant with **FPU**).
- **Single memory bus** cached interface (AXI4 - SDRAM/SRAM).
- **Single periphery bus** interface (AXI4 - Devices).
- Instruction Cache: **16KB** (1/3 of CPU area)
- Data Cache: **16KB** (1/3 of CPU area)

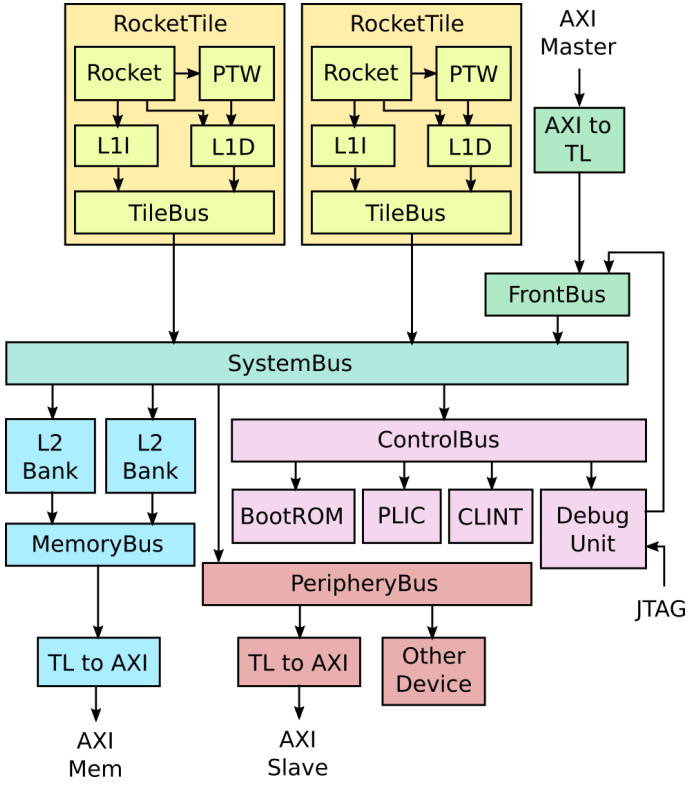
- RISC-V CPU extended customizations:

- Customized **low hardware footprint boot ROM (269 bytes)**.
- **Clock gating mechanism for cache** memories **(2-3mW reduction)**.
- Attached 1ms interrupt core for **RTOS support**.
- Updated interrupt controller to support **6 IRQ lines** (for SPI, I2C, etc)
- **High speed JTAG debug** support (>30MHz) programmable IO pin drive strength coupled with onboard FT2232 chip.

Customized CPU Cache Clock Gating Circuitry



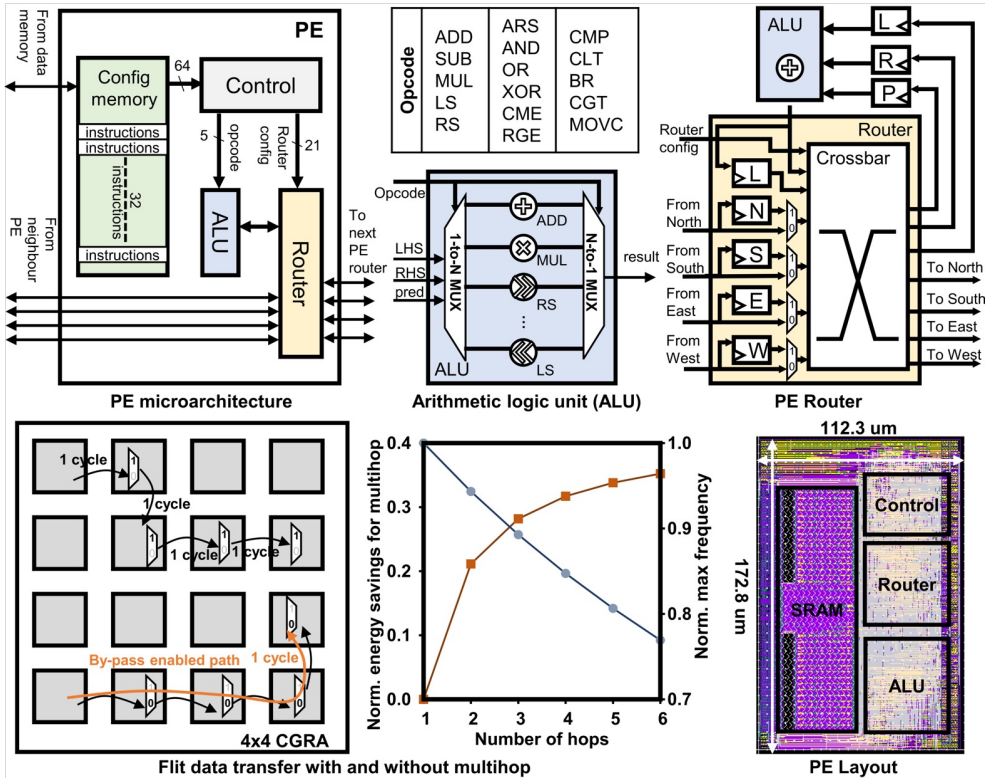
Rocket Core System Architecture



PACE CGRA PE Microarchitecture

- The PEs consists of:
  - Can hold 32 x 64-bit instructions
  - An arithmetic logic unit (ALU)
  - A register-file based configuration memory
  - A configurable router with a bypass-able registered interface to support multihops

- The ALU supports 17 operations
  - Has arithmetic, shift and Boolean instructions.
  - Supports control divergence and recurrence edges using comparator (CMP) and branching (BR) opcodes.
  - NOP instruction added for better power savings

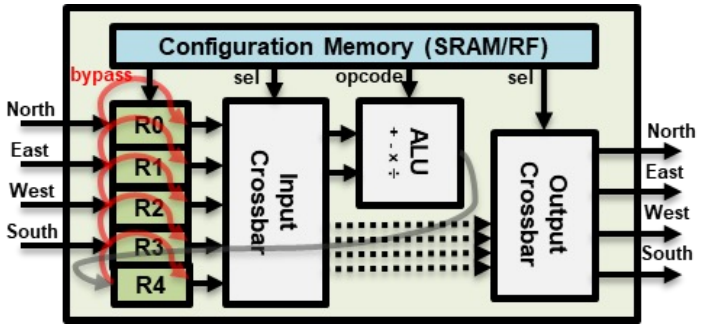


PACE Idle State Implementation (Tile)

Idle state implementation:

- **NOP** (no-operation) is opcode 0.
- Opcode operand should include **how many** idle/NOP cycles is needed, to allow for accurate clock gating counter logic.

Opcode (4-bits)	RHS Operand (16-bits)	LHS Operand (16-bits)
0000 (**NOP**)	**Number of idle cycles** (previously unused)	Unused



Clocked units:

- **Router/Crossbar** (if bypass not enabled)
- **Various registers**: start_exec_shifted, prev_cycle_p_i2_i1, operands, etc.

Notes:

- Logic that translates the address into the PE mem must not be gated at the n-1 cycle of the idle state.
- Turning the clock off for some registers may lead to unexpected behavior, that may require debug & fine tuning.

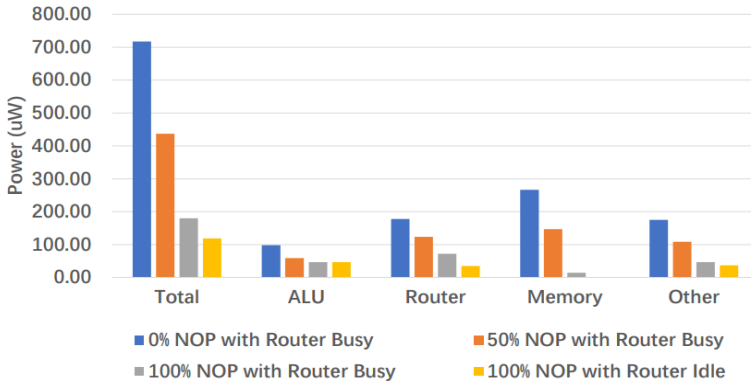


Figure 5.5: Power analysis of PE sub-modules.

PACE Idle State Implementation (Tile)

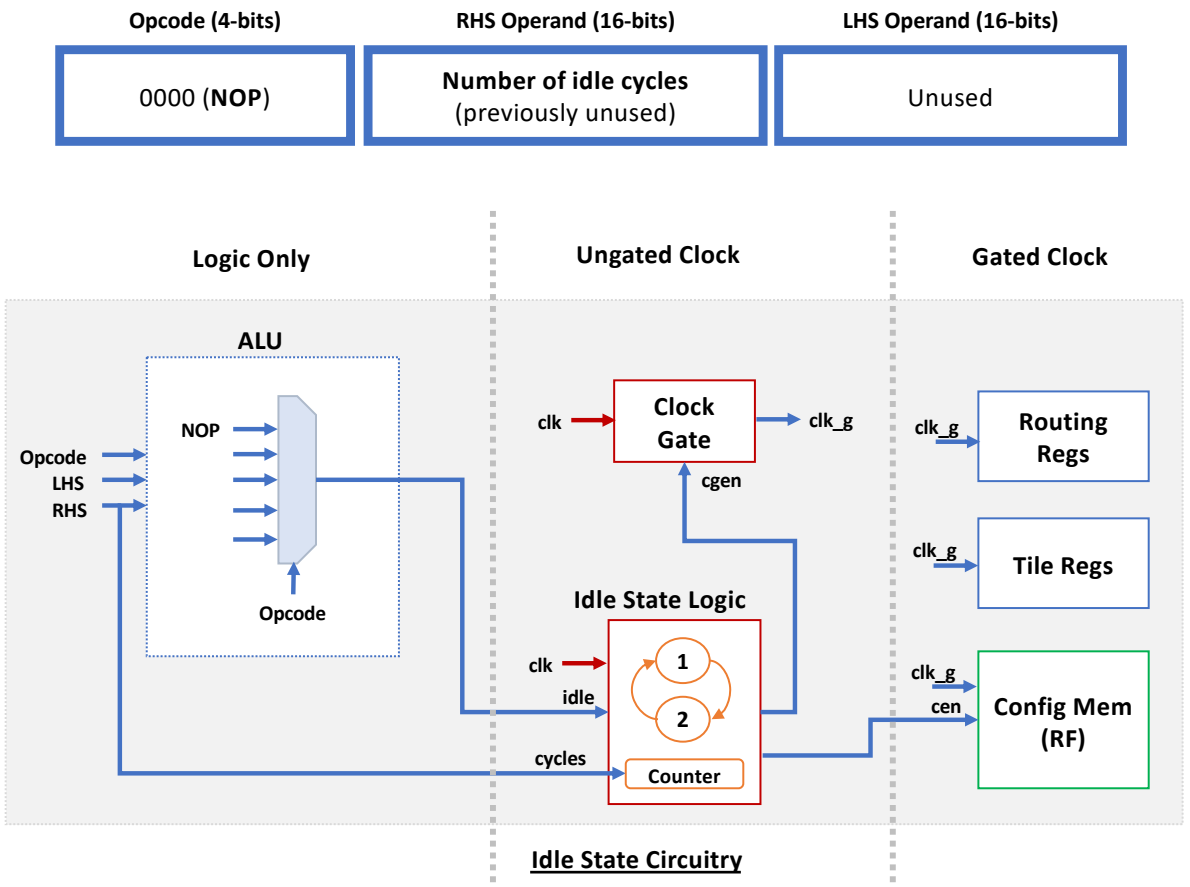
Functional steps:

1. The idle state logic unit waits for the tile to receive a NOP instruction.
2. The RHS operand is decoded as the number of idle cycles.
3. If RHS > 0, then the clock is immediately turned of via the clock gate.
4. Once the counter has reached the RHS value, the counter is reset and the clock is resumed.

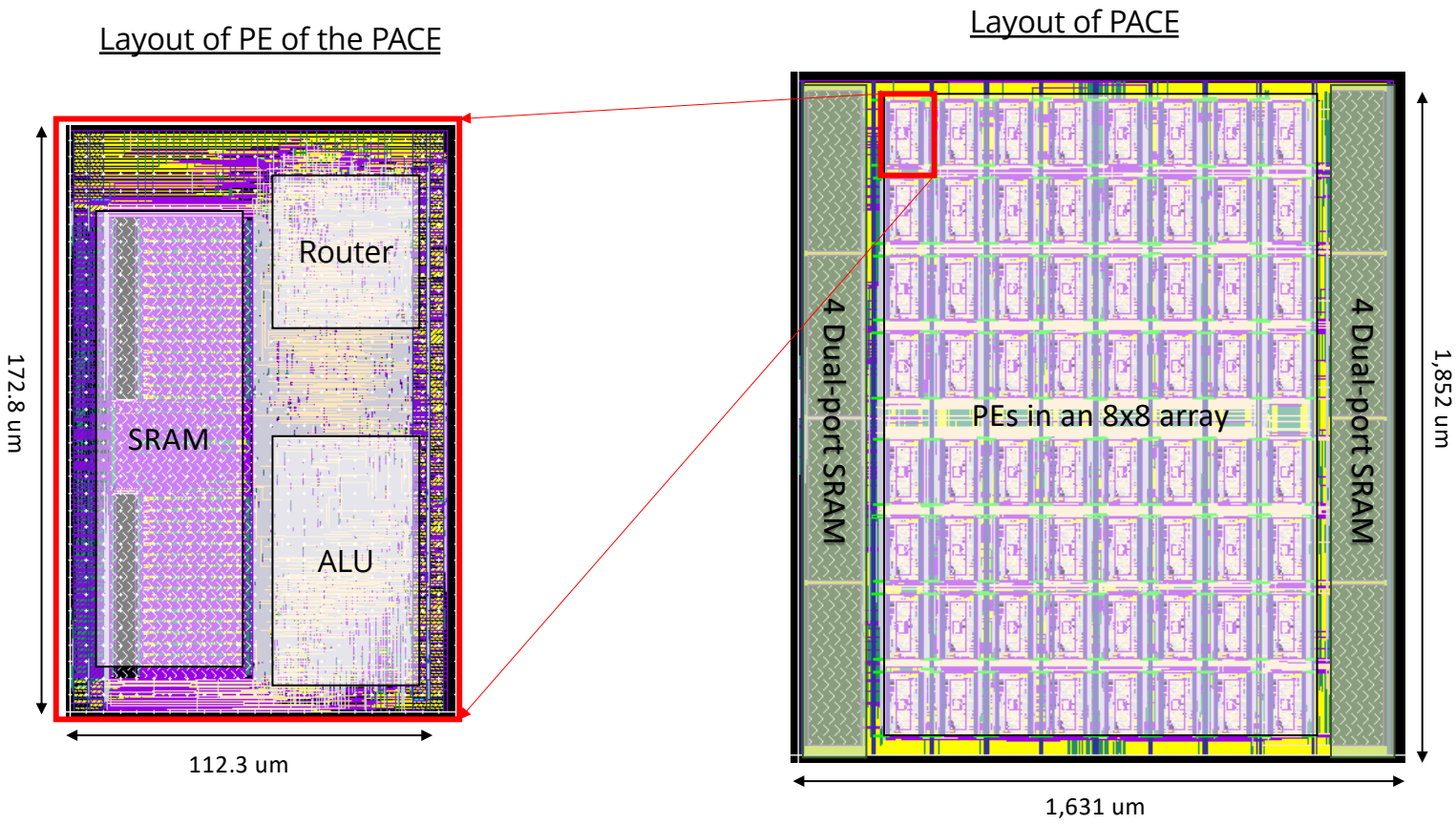
Note: This feature can be disabled via NOP with operand 0.

	Old Tile	New Tile
Area	11346.962	11453.225 (+2%)
Cell Count	2080	2121 (+2%)

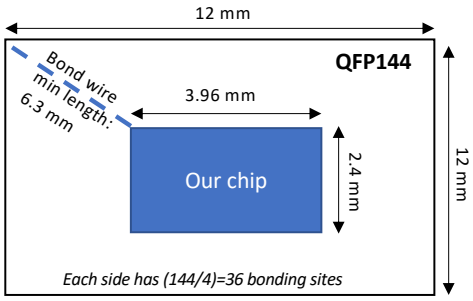
Area Difference



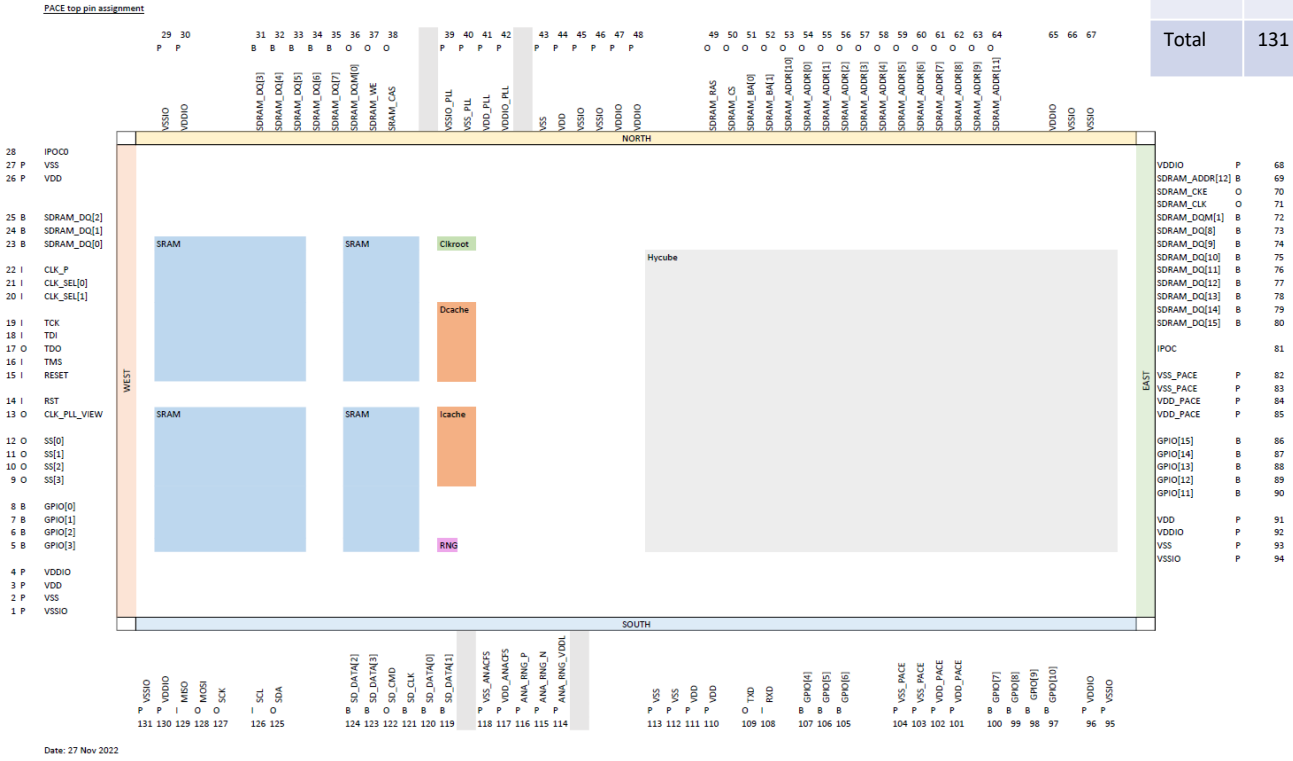
PACE Accelerator Layout



Updated IO planning



Side	#pads	#signals	#power pads	Power
North	39	24	15	Core, IO, PLL
South	37	20	17	Core, IO, RNG, PACE
East	27	18	9	Core, IO, PACE
West	28	22	6	Core, IO
Total	131	84	47	



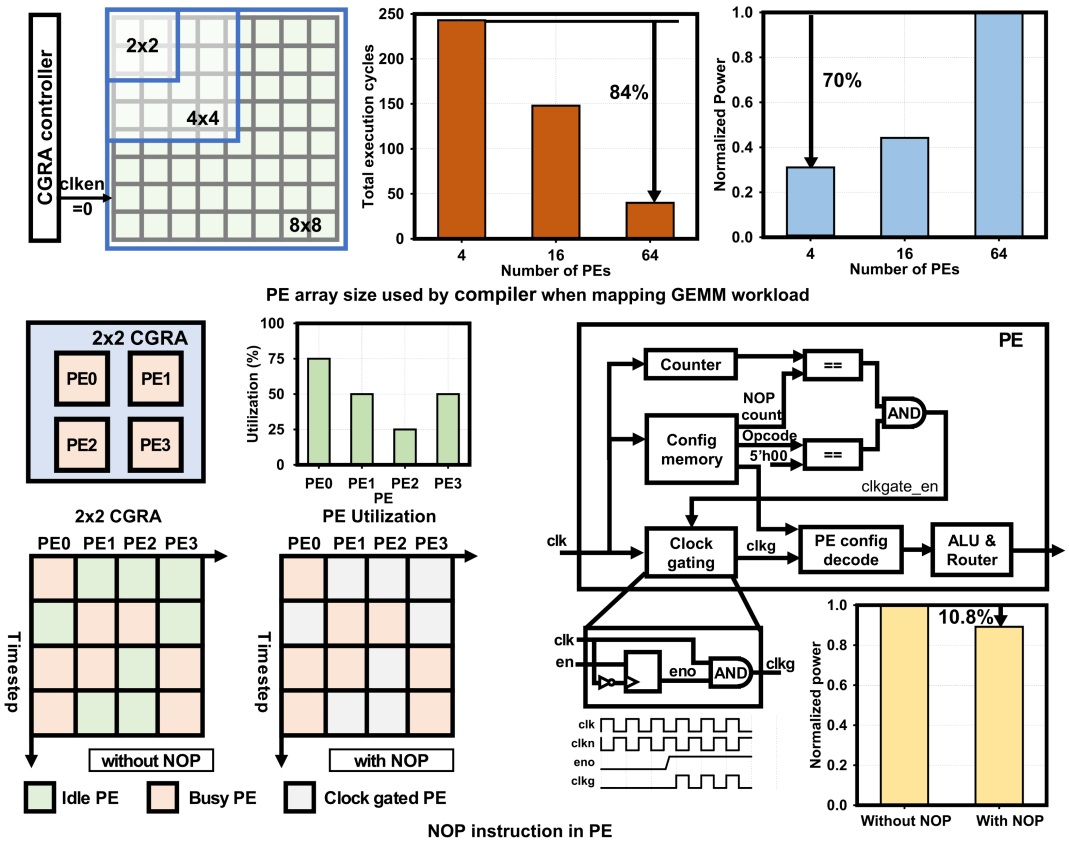
The pad arrangement takes into account that:

- Pin sequence of external drivers such as SD and SDRAM
- Maximum #bonding site at all sides
- Most power pads are placed at the edges of the die
- #VDDIO/VSSIO pairs to provide enough power for driving external devices

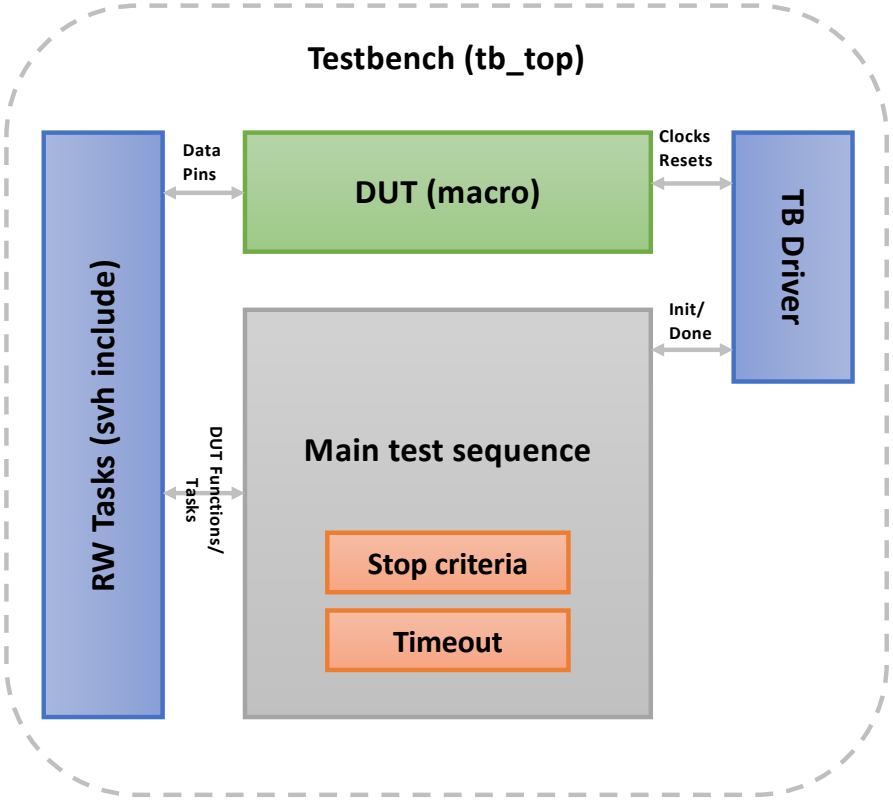
In total 131 pads, hence QFP144 package is planned

Static and dynamic clock gating

- Static clock gating can be applied when inactive PEs are known beforehand
  - The power savings can be reduced by 70%
- Dynamic clock gating is enabled by the NOP instructions
  - A PE can save 10.8% of powers when executing NOP



PACE SoC Verification Architecture

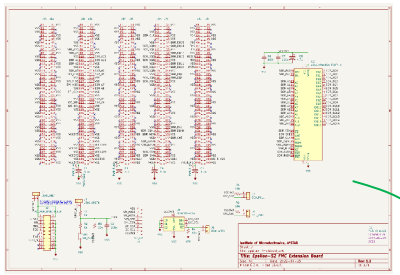


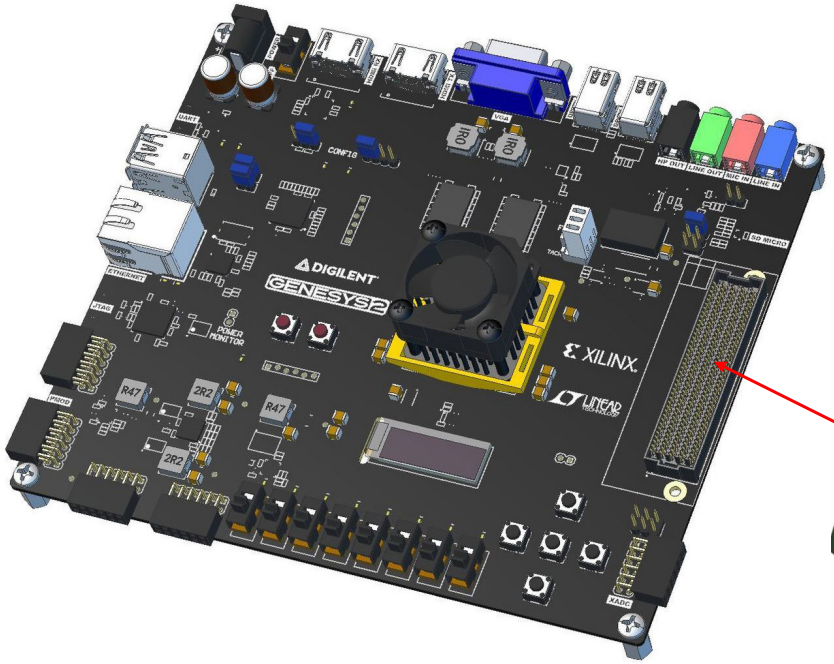
Standardized TB architecture:

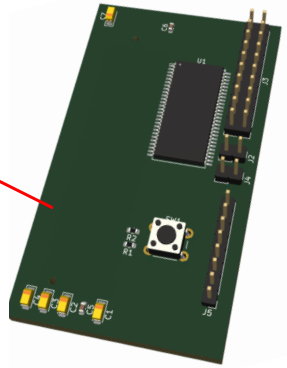
- **Summary**: Minimal effort TB sim development work.Tests will just load the compiler generated traces into the memories and simulate the chip.
- Predefined DUT macro and connections to avoid mistakes.
- Uniform TB driver sequences.
- RW tasks for data injection, eg. QSPI transacions, config/data memory loading sequences.
- Main test sequence is test dependent.
- All tests can be power profiled.

FPGA-based Verification Platform

Customized Epsilon-S2 extension PCB to closely emulate final chip







Digilent Genesys 2 features:

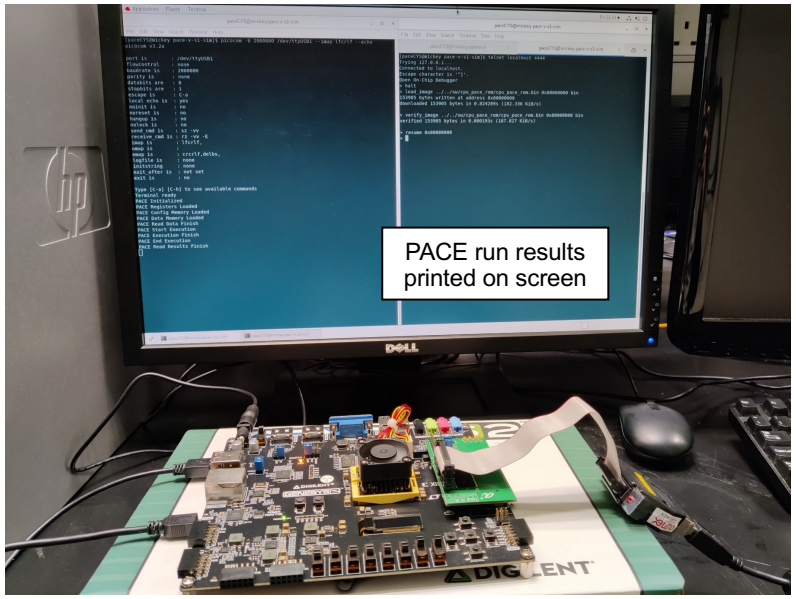
- On-board FPGA: Xilinx Kintex-7™ FPGA (XC7K325T-2FFG900C)
- Selectable IO voltage (1.8V-**3.3V**)
- Micro SD card connector
- Fully-populated 400-pin FMC HPC connector w/ ten GTX lanes
- USB-UART Bridge
- 128x32 pixel OLED
- 16-bit VGA connector
- 10/100/1000 Ethernet PHY
- 1GB 1800Mbps on-board DDR3
- Dedicated USB port for JTAG programming and data transfers

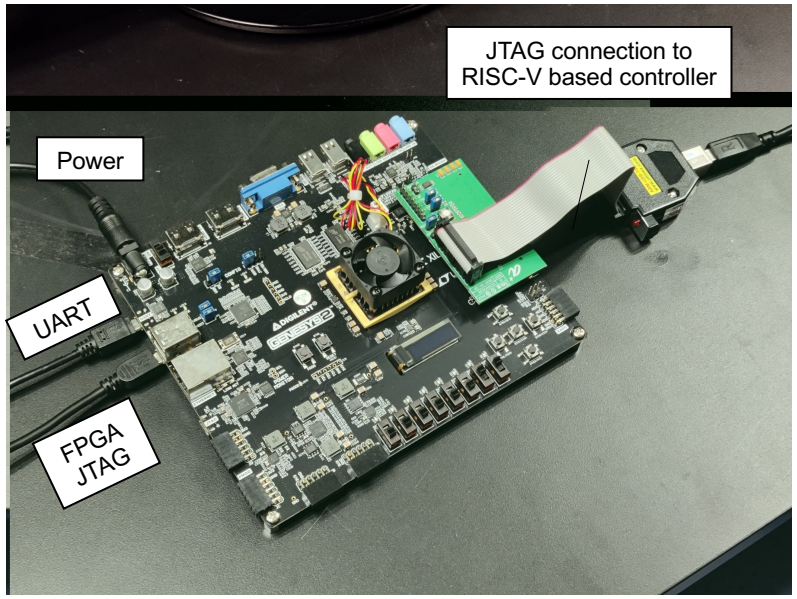
Epsilon S2 extension PCB:

- 400-pin FMC connector
- SDRAM chip
- I2C EEPROM chip
- Reset pushbutton
- JTAG headers
- I2C/SPI/QSPI headers

FPGA emulation setup

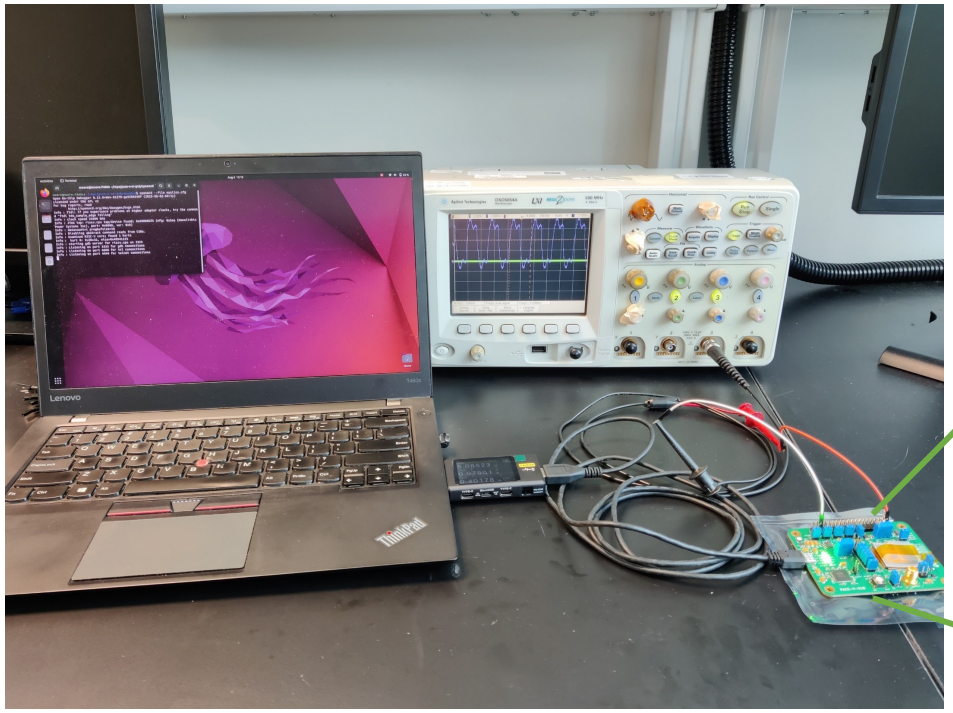
- The PACE design coupled with a RISC-V CPU is synthesized and run on a FPGA board successfully
- The PACE CGRA can run computation successfully



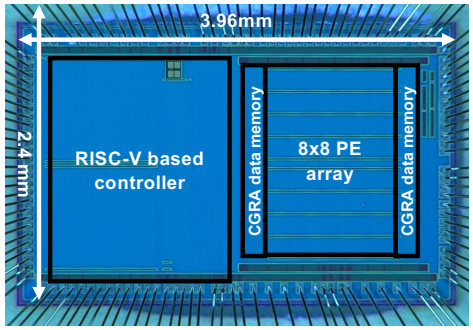


Test chip assembled on PCB for testing

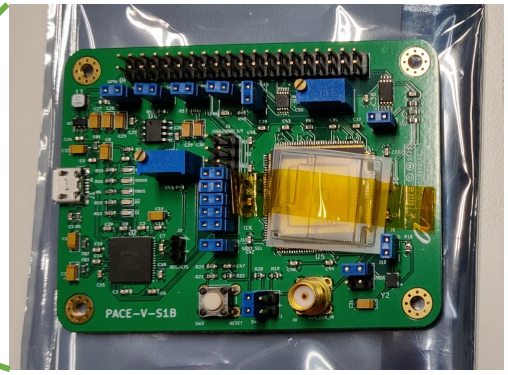
- The PACE SoC is taped out in UMC40nm process node and is assembled on PCB



Test laptop, oscilloscope and the PCB board



Chip micrograph



PACE assembled on PCB board