EMBEDDED HETEROGENEOUS COMPUTING: A SOFTWARE PERSPECTIVE

Tulika Mitra
School of Computing
National University of Singapore

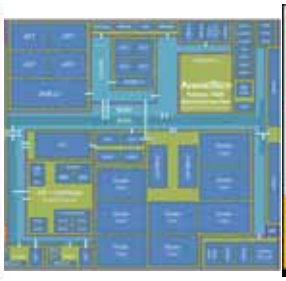
IEEE Silicon Valley Computer Chapter, October 2021

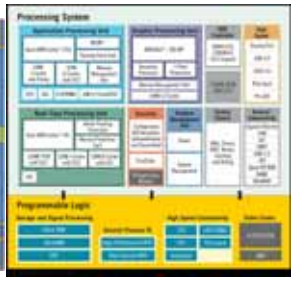
For future talks, or to view past videos, visit our website: **r6.ieee.org/scv-cs/**

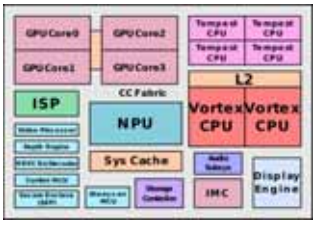
Embedded Heterogeneous Computing

- A System-on-a-Chip (SoC) comprising of heterogeneous computing elements, such as different CPU cores, GPU cores, DSP cores, and accelerators

Qualcomm Snapdragon 835


Samsung Exynos 7420


Xilinx Zynq Ultrascale+

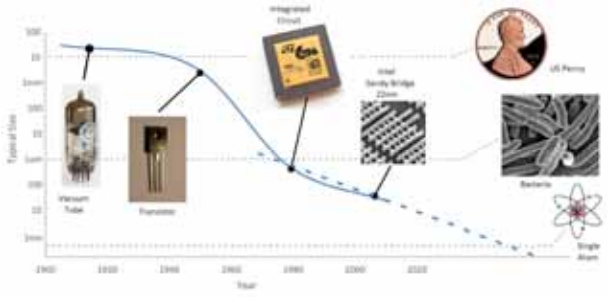


Apple A12 Bionic

Image credit: Qualcomm, AnandTech, Xilinx, Wikichip

Moore's Law

- Prediction by Intel co-founder Gordon Moore in 1965:
- Transistor density in an integrated circuit doubles every year or two
- Transistor's linear dimension shrinks by a factor of λ, area shrinks by a factor of λ^2
  Number of transistors increase by a factor of λ^2 per technology generation



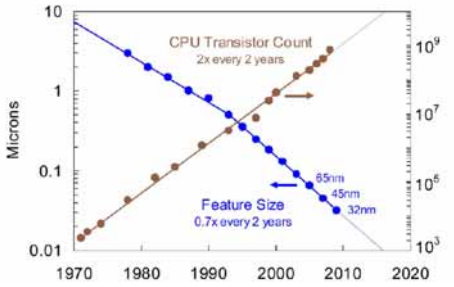


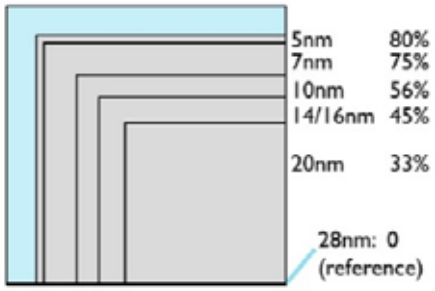
Image credit: Soham Chatterjee, KJ Kuhn

End of Dennard Scaling

- Complementary to Moore's Law formulated by Robert Dennard in 1974
- As transistors get smaller, their power density stays constant
- Transistor's linear dimension scaling factor: $\frac{1}{\lambda}$ (≈ 0.7x per technology generation)
- Transistor area scaling factor: $\frac{1}{\lambda^2}$ (≈ 0.5x per technology generation)
- Both current and voltage scaling factor: $\frac{1}{\lambda}$
- Transistor frequency scaling factor: λ
- Power dissipation scaling factor: $\frac{1}{\lambda^2}$
- Power density scaling factor: 1
- Dennard scaling broke down around 2004 with unscaled interconnect delay and our inability to scale down voltage and current due to reliability concerns

Creation of Dark Silicon

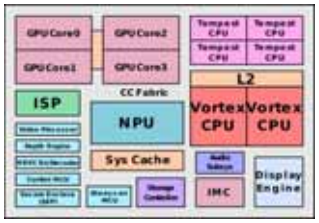
- We can have more transistors and cores due to Moore's Law
- We cannot power them all due to Thermal Design Power (TDP) constraint
- At 5nm, 80% of the chip will be dark



Source: Paul McLellan, Cadence

Heterogeneous cores towards rescue

- Turning a problem into an opportunity: Silicon area is cheaper relative to power
- Spend *area* to buy *power*: heterogeneous cores
- Power on only the most appropriate cores
- Energy-efficient, high-performance computation



Apple A12 Bionic

Image credit: Wikichip

Functional Heterogeneity

- Computational elements with different functionality and instruction-set architecture
- Different programming models (DSP, GPU, FPGA, CGRA) or non-programmable accelerators (video encoder, machine learning accelerator)

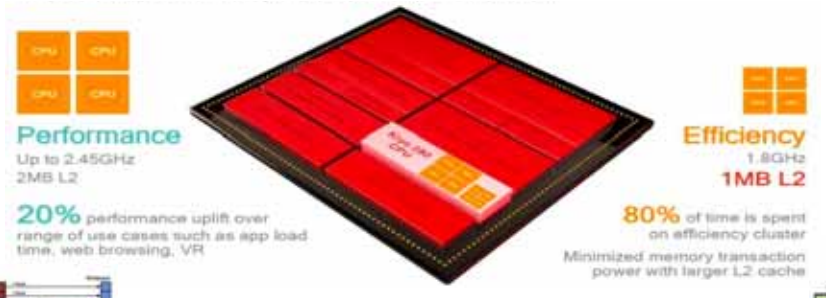
Qualcomm Snapdragon

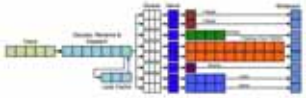
NVIDIA Tegra

Image credit: Qualcomm, NVIDIA

Performance Heterogeneity

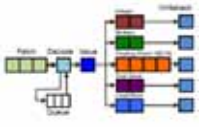
- Cores with the same instruction-set architecture but different underlying micro-architectures
- Different performance-power-area characteristics





ARM big core

Qualcomm KRYO 280 CPU in SNAPDRAGON 835



ARM LITTLE core

Image Credit: ARM, Qualcomm

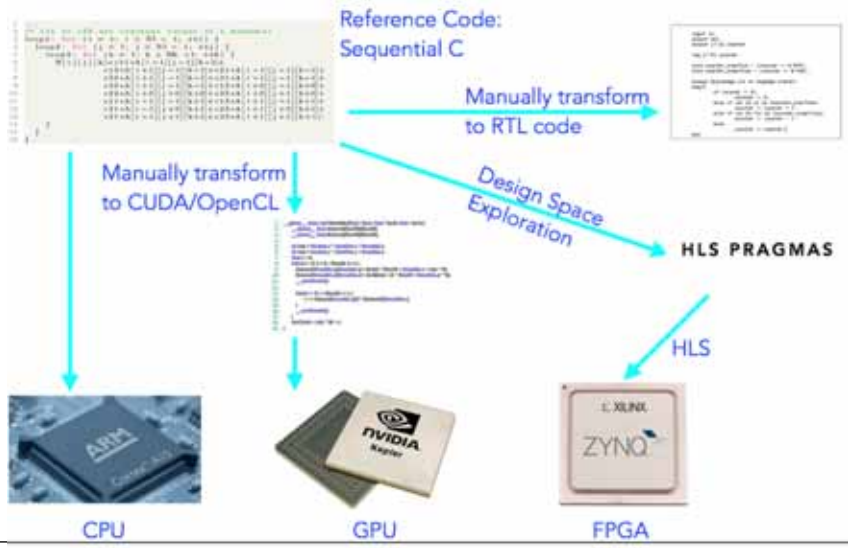
MISSING PIECE: SOFTWARE FOR HETEROGENEOUS COMPUTING

Image credit: Riosis, Shutterstock, Khoi Vinh

Best of Times… Worst of Times..

- Tremendous progress in embedded SoC design with heterogeneous computing
- Software development remains as challenging (if not more)
  - Which core or accelerator to use for my app?
  - How to orchestrate application execution across multiple cores or accelerators?
  - How to remain within TDP (thermal design power) in the dark silicon era?
  - How to maximize battery life?

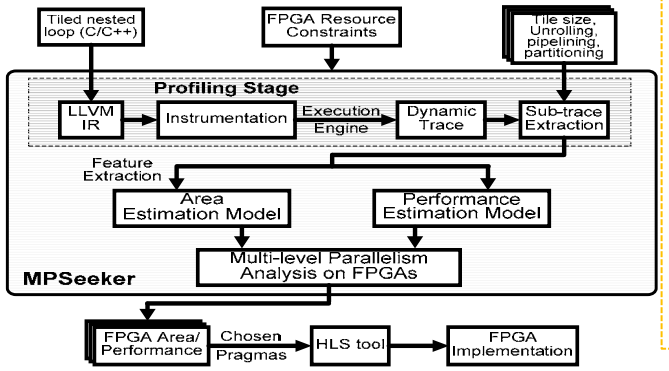
Compute Core Selection



Performance Prediction on Accelerators

- Lin-Analyzer: Performance prediction for FPGAs from sequential C code
- CGPredict: Performance prediction for GPUs from sequential C code
- Provide performance insights to the software developers

FPGA Performance Estimation and DSE from Sequential Code without HLS

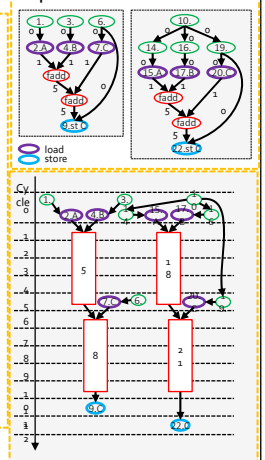


The trace (LLVM-IR):

```
... ...
%indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next, %for.body ]
%arrayidx = getelementptr inbounds float* %A, i64 %indvars.iv
%tmp = load float* %arrayidx, align 4
%arrayidx2 = getelementptr inbounds float* %B, i64 %indvars.iv
%tmp1 = load float* %arrayidx2, align 4
%add = fadd float %tmp, %tmp1
%arrayidx4 = getelementptr inbounds float* %C, i64 %indvars.iv
%tmp2 = load float* %arrayidx4, align 4
%add5 = fadd float %tmp2, %add
store float %add5, float* %arrayidx4, align 4
%indvars.iv.next = add nuw nsw i64 %indvars.iv, 1
%exitcond = icmp eq i64 %indvars.iv.next, 16
br i1 %exitcond, label %for.end, label %for.body

%indvars.iv = phi i64 [ 0, %entry ], [ %indvars.iv.next, %for.body ]
%arrayidx = getelementptr inbounds float* %A, i64 %indvars.iv
%tmp = load float* %arrayidx, align 4
%arrayidx2 = getelementptr inbounds float* %B, i64 %indvars.iv
%tmp1 = load float* %arrayidx2, align 4
%add = fadd float %tmp, %tmp1
%arrayidx4 = getelementptr inbounds float* %C, i64 %indvars.iv
%tmp2 = load float* %arrayidx4, align 4
%add5 = fadd float %tmp2, %add
store float %add5, float* %arrayidx4, align 4
%indvars.iv.next = add nuw nsw i64 %indvars.iv, 1
%exitcond = icmp eq i64 %indvars.iv.next, 16
br i1 %exitcond, label %for.end, label %for.body
... ...
```

Optimized DDDG



Design Space exploration of FPGA-based accelerators with multi-level parallelism DATE 2017
Lin-Analyzer: a high-level performance analysis tool for FPGA-based accelerators DAC 2016
https://github.com/zhguanw/lin-analyzer

Power of Lin-Analyzer

Input Size	Loop Pipelining	Loop Unrolling	Array Partitioning	Vivado HLS Runtime
32*32*32	Disabled	loop3 factor=30	A, cyclic, 2 B, cyclic, 2	44.25 seconds
	loop3, yes	loop3 factor=15	A, cyclic, 16 B, cyclic, 16	1.78 hours
	loop3, yes	loop3 factor=16	A, cyclic, 16 B, cyclic, 16	3.25 hours

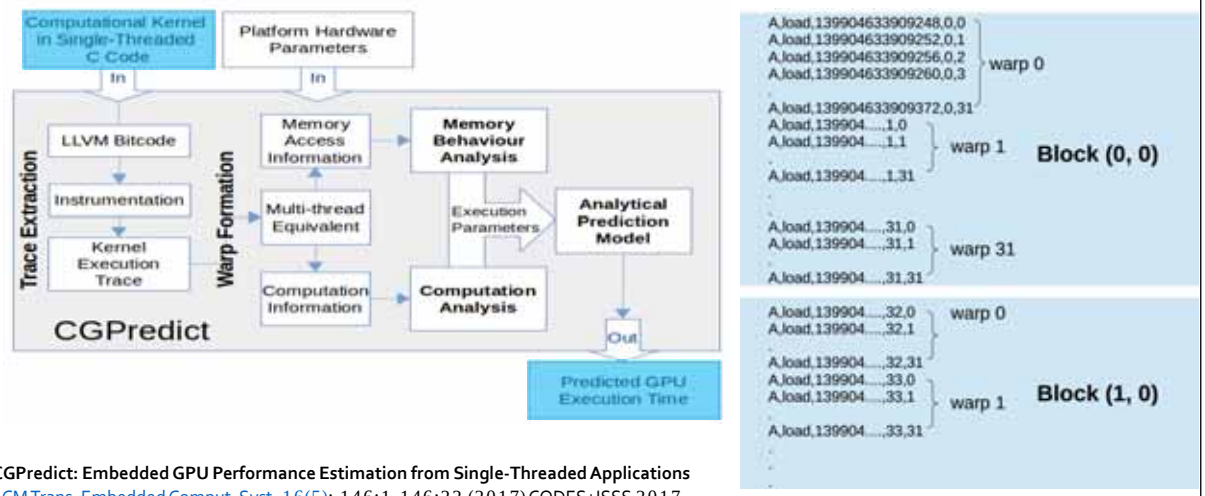
Input Size	Design Space	DSE Time	
		Exhaustive HLS-based	Lin-Analyzer
32*32*32	120	10 days*	29.30 seconds

Accuracy

Benchmark	ATAX	BICG	CONV2D	CONV3D	GEMM
Difference (%)	2.68	1.24	1.63	3.75	3.25
Benchmark	GESUMMV	MM	MVT	SYR2K	SYRK
Difference (%)	5.15	2.69	2.05	1.32	2.78

Average design space exploration time down from 28.1 hours (using HLS) to 1.6 mins

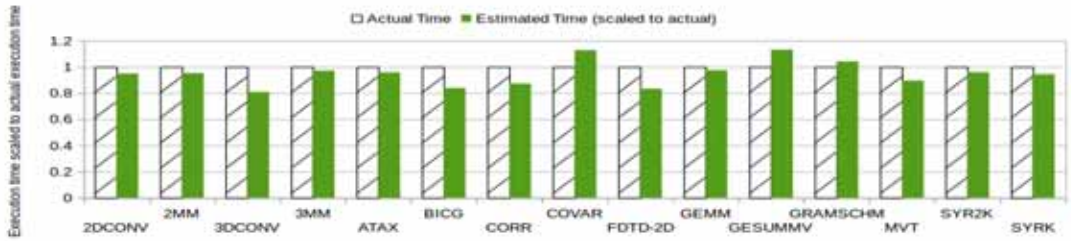
Sequential C code to GPU Performance



CGPredict: Embedded GPU Performance Estimation from Single-Threaded Applications
ACM Trans. Embedded Comput. Syst. 16(5): 146:1-146:22 (2017) CODES+ISSS 2017

Accurate Performance Estimates

- Estimation Error: As low as 2.66%, avg 9%

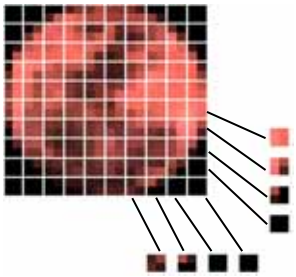


Choice between GPU and FPGA via Analysis of C programs

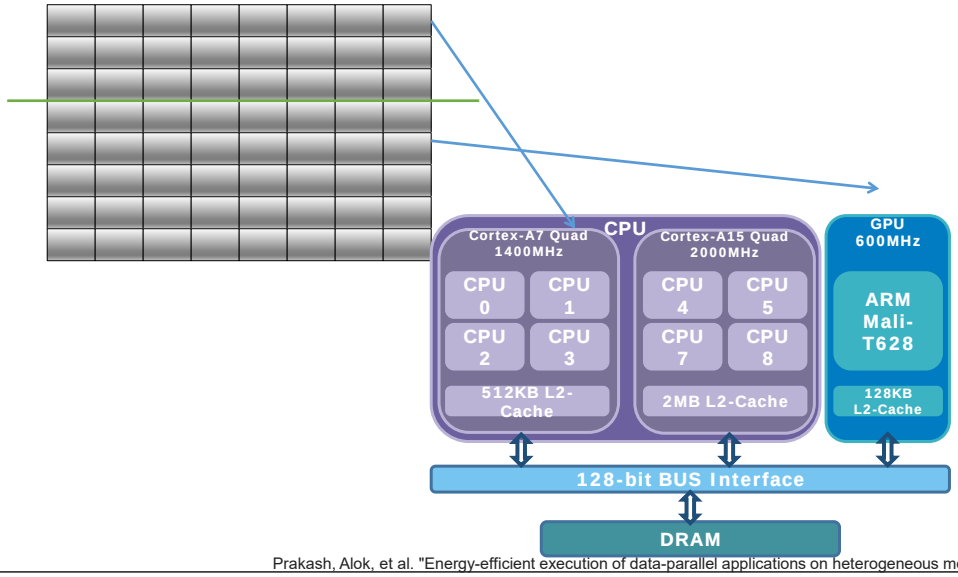
Benchmark Name	Input Size	Estimated Time (ms)		Actual Time (ms)		Choice of Platform
		GPU	FPGA	GPU	FPGA	
MM	1024	**242.51**	1180	**250.27**	1450	GPU
MVT	2048	48.31	**9.09**	42.371	**10.41**	FPGA
GEMVER1	2048	**2.61**	16.55	**4.57**	19.81	GPU
DERICHE1	1024	**0.95**	2.99	**1.53**	3.37	GPU
DCT1D	1024	2697.75	**636.47**	2685.362	**650.8**	FPGA

OpenCL to Support Heterogeneity

- Free, open standard for cross-platform, parallel programming on CPU, GPU, FPGAs
- Fine grained parallelism to allow thousands of active threads

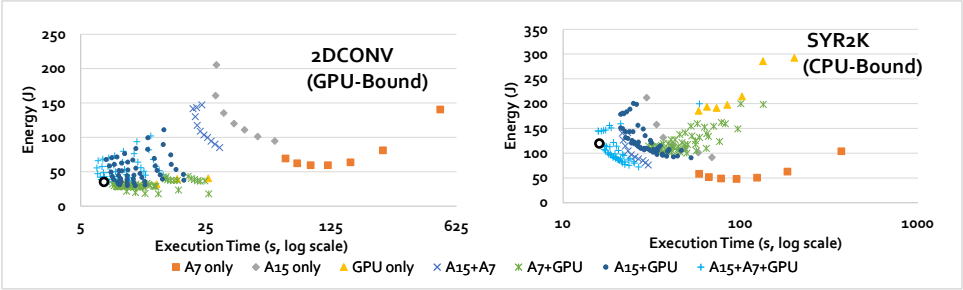


Workload Partition: CPU + GPU



Prakash, Alok, et al. "Energy-efficient execution of data-parallel applications on heterogeneous mobile platforms." *ICCD 2015*

Design Space: Energy-Delay

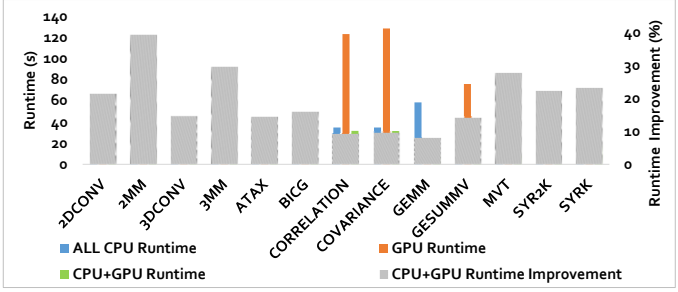


- 2DCONV: Works well with Functional Heterogeneity (A7+GPU)
- SYR2K: Works well with Performance Heterogeneity (A7+A15)
- Main Factors: DVFS, Workload Partition & Memory Contention

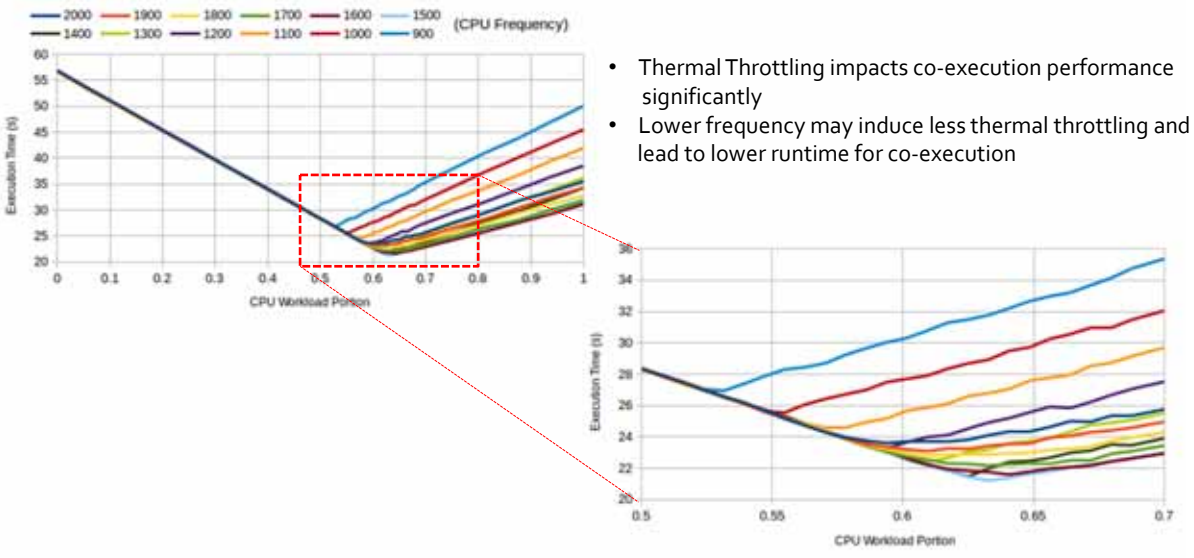
Finding optimal configuration: Models

Gain with Heterogeneity

- Optimal partitioning for CPU and GPU + DVFS
- Average 19% runtime improvement



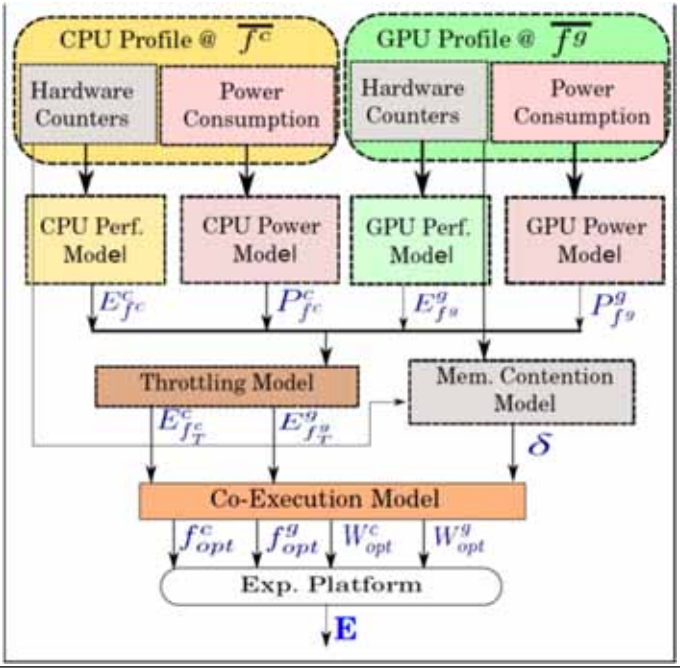
Thermal Constraint



- Thermal Throttling impacts co-execution performance significantly
- Lower frequency may induce less thermal throttling and lead to lower runtime for co-execution

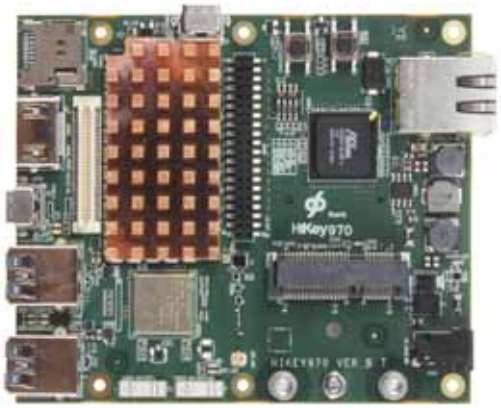
OPTiC

- A framework to select CPU-GPU workload partition and corresponding frequency to minimize thermal throttling and maximize performance



Wang, Siqi et al. OPTiC: Optimizing Collaborative CPU-GPU Computing on Mobile Devices with Thermal Constraints. IEEE TCAD 2019

Deep Learning on Embedded Heterogeneous SoC



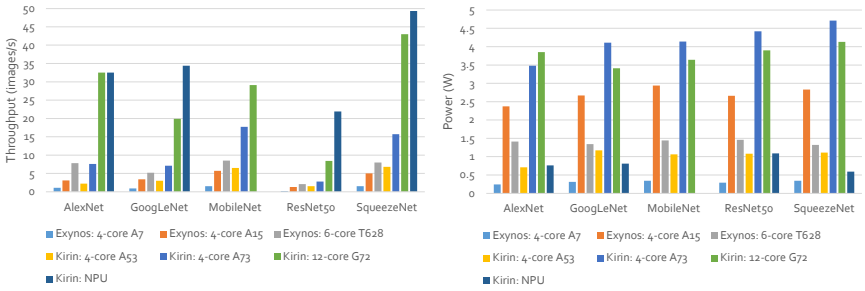
Huawei HiKey 970 Mobile Platform

HiSilicon Kirin 970
big.LITTLE CPU, GPU, NPU, ISP, Security Engine

Image Credit: Huawei

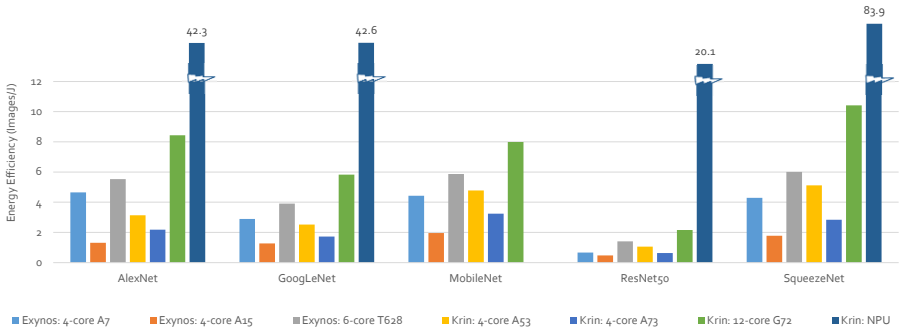
Performance and Power Characteristics

- GPU is comparable to NPU
- Performance gap between GPU and CPU clusters is about 2.5X
- High-end GPU comsumes high power comparable to big CPU cluster
- Low-end SoCs have no single component sustaining ML workloads



Energy-Efficiency Characterization

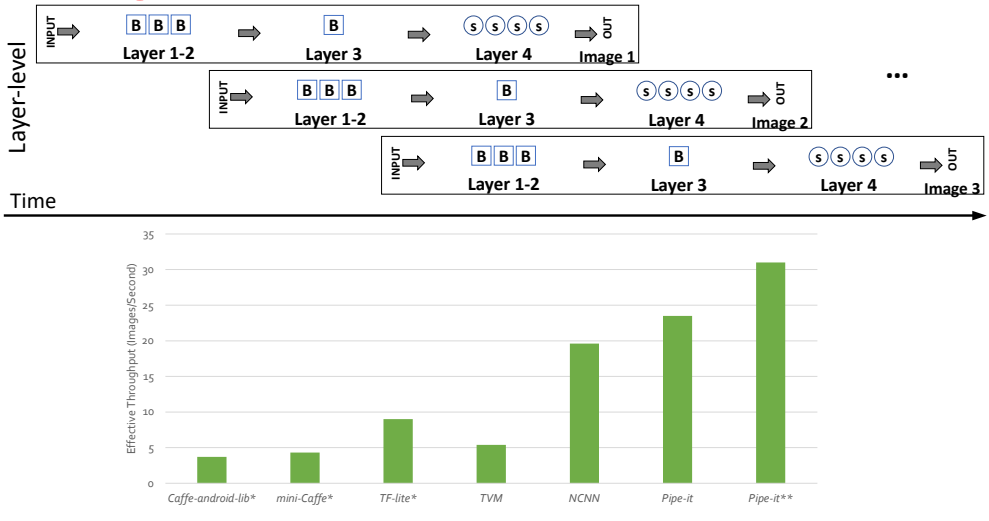
- Energy efficiency is defined as Images/Joule
- NPU is superior in both performance and energy efficiency



Synergistic Coarse-grained Co-execution

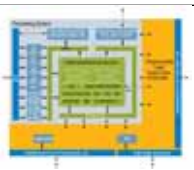
Siqi Wang et al. "Neural Network Inference on Mobile SoC" IEEE Design & Test 2020

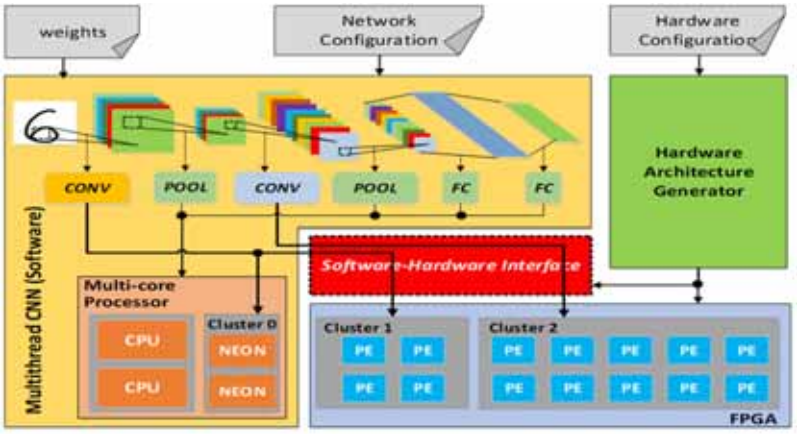
Pipe-It: Synergistic Pipelined Execution on ARM big.LITTLE



High-Throughput CNN Inference on Embedded ARM big. LITTLE Multi-Core Processors. TCAD 2020

Co-operative CNN on CPU, Neon, FPGA

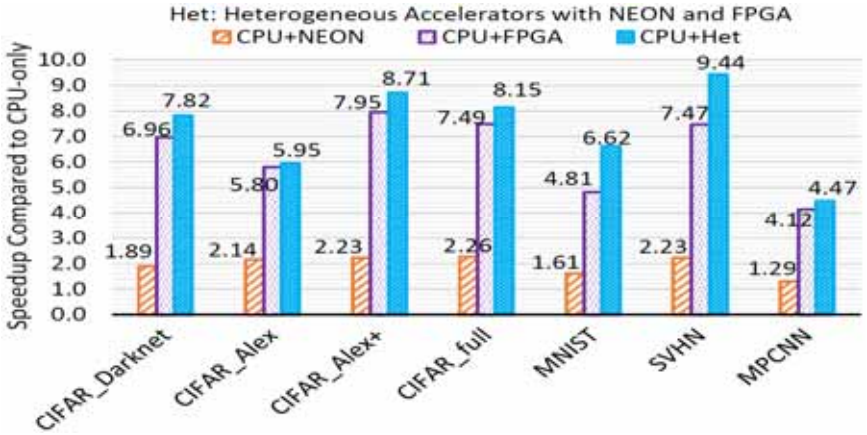




- Xilinx ZedBoard Zynq XC7Z020 low-cost SoC
- 63.5 frames/sec (CIFAR)
- 1.67 GOPS 32-bit floating-point
- 54 mJ/frame
- Highest throughput, energy-efficiency at low-cost

Synergy: A HW/SW Framework for High Throughput CNNs on Embedded Heterogeneous SoC. ACM TECS 18(2), 2019

Benefit of Co-operative Heterogeneity



Dark Silicon in Practice: Samsung Exynos 7420 SoC

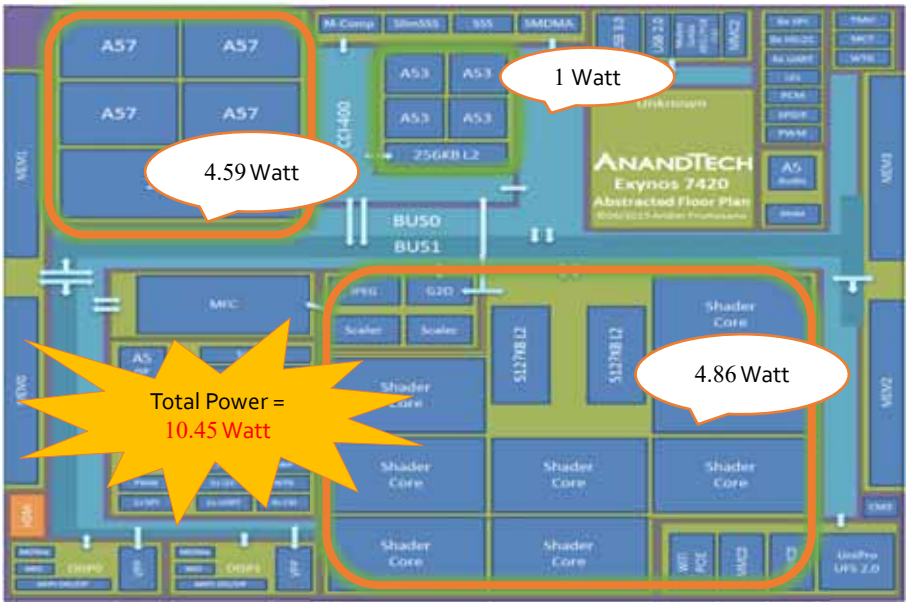


Image credit: AnandTech

Heterogeneity-aware Runtime Management

- Maintain total chip power below Thermal Design Power (TDP)
- Knobs: Orchestrate in synergistic fashion
  - Task to core mapping and migration
  - DVFS and power states per core/cluster
- Goal: Provide best user experience while minimizing energy

User experience in Interactive applications

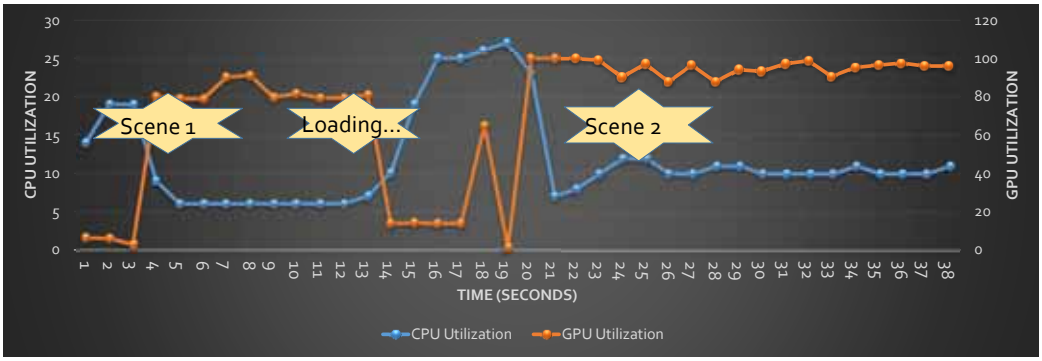
- Quality of Service (QoS) is the key metric
  - Example: Frames per second in games, video/audio encoding/decoding
- QoS should be maintained within a range for each task
- Going beyond the maximum QoS bound is wasteful from power/energy perspective

Android Power/Thermal Management

CPU DVFS (Linux Governor)	GPU DVFS (Firmware-based Governor)

No synergy between the two DVFS managers!

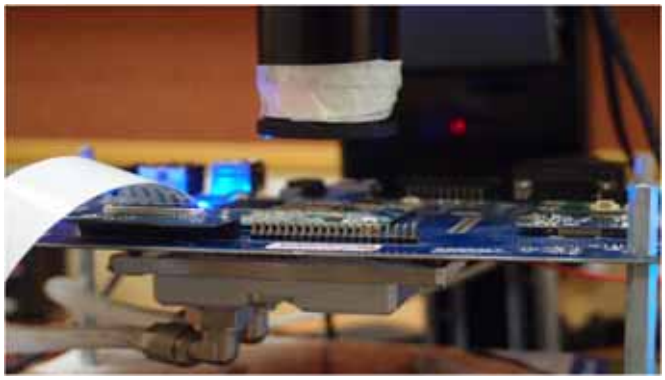
Co-Operative Power Management

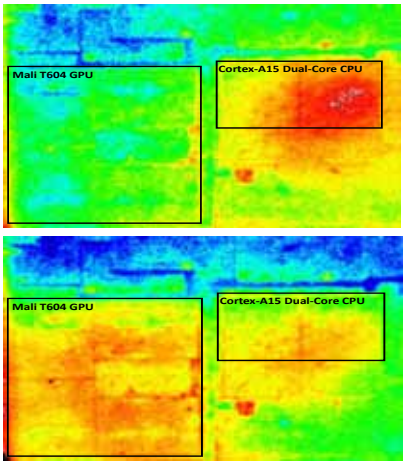


- Average 20% FPS per Watt improvement through synchronized, QoS-aware power management

Power-Performance Modelling of Mobile Gaming Workloads on Heterogeneous MPSoCs DAC 2015
Integrated CPU-GPU Power Management for 3D Mobile Games DAC 2014

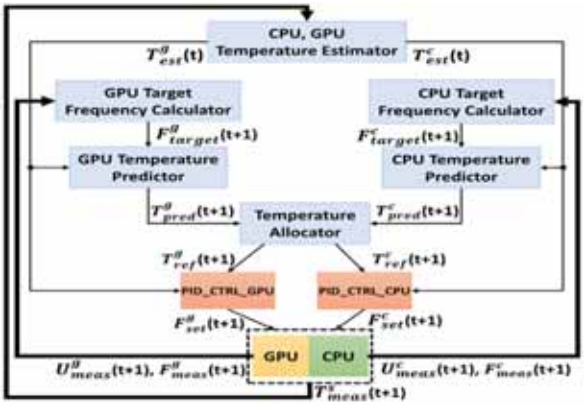
CPU-GPU Thermal Hotspot switching





Collaboration with Chair for Embedded Systems , KIT

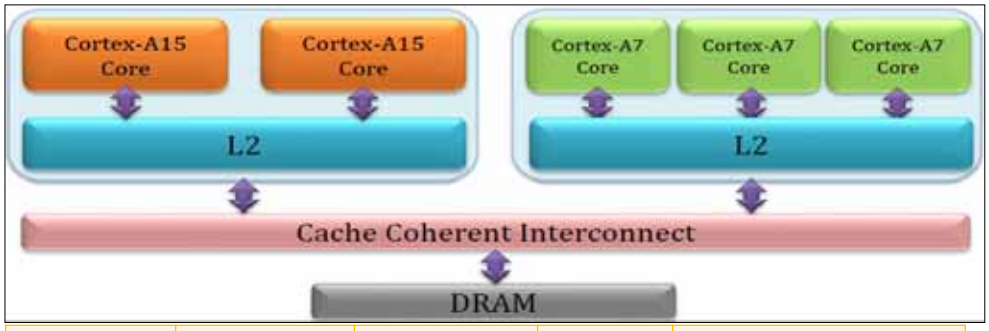
Coordinated Thermal Management



- Average 19% improvement in FPS over Linux
- Peak temperature reduced from 83°C to 77°C
- Minimal temperature variance compared to 10°C variance for Linux
- Similar average temperature

Improving mobile gaming performance through cooperative CPU-GPU thermal management DAC 2016

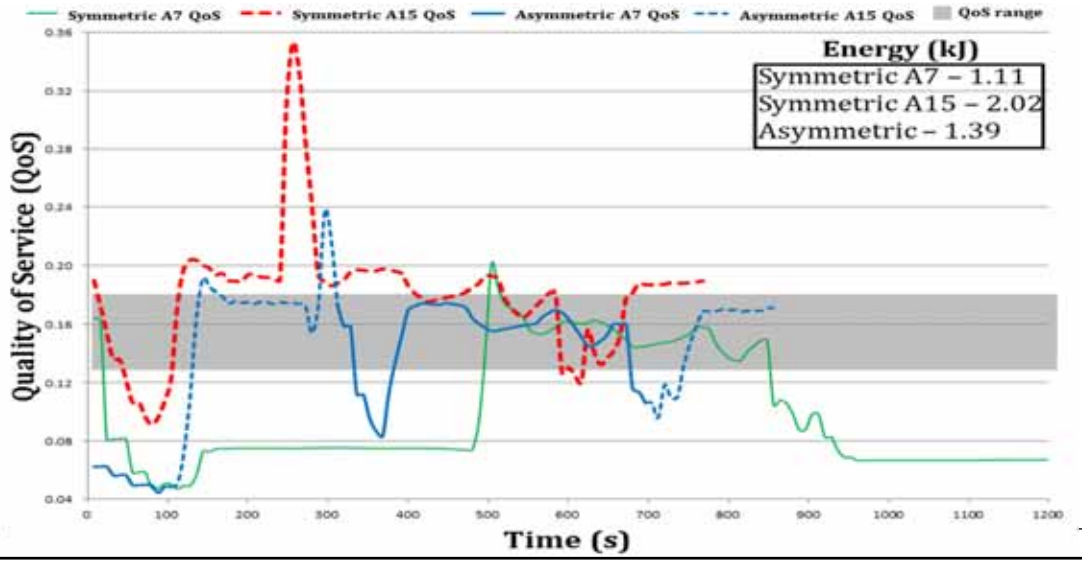
ARM big.LITTLE Architecture



CPU	Pipeline	Issue Width	L2$ size	Frequency Range
small	In-order	2	512 KB	350 – 1000 MHz
big	Out-of-order	3	1 MB	500 – 1200 MHz

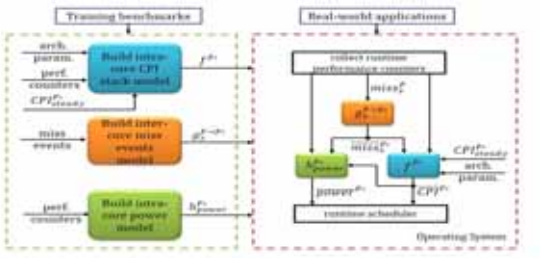
* Test chip at 45nm technology

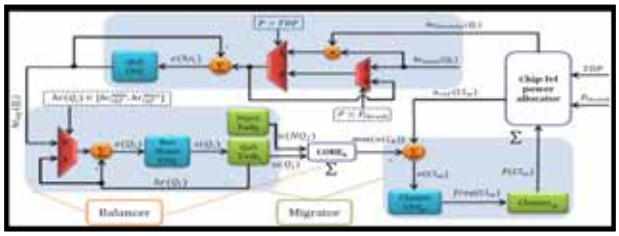
Co-operative Task Migration and DVFS



Average Power Comparison

- Default Linux governor: 5.99 W
- Reactive (Control theory): 3.43 W
- Proactive (Price theory): 2.96 W

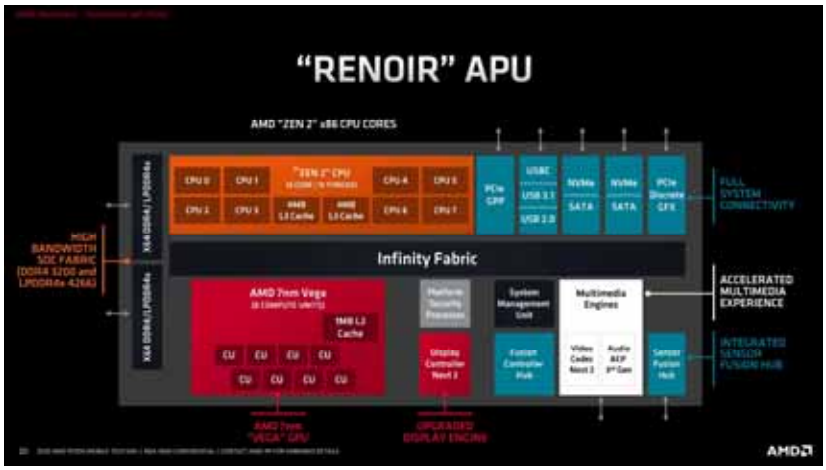




Hierarchical power management for asymmetric multi-core in dark silicon era DAC 2013
Power-performance modeling on asymmetric multi-cores CASES 2013
Price theory based power management for heterogeneous multi-cores ASPLOS 2014

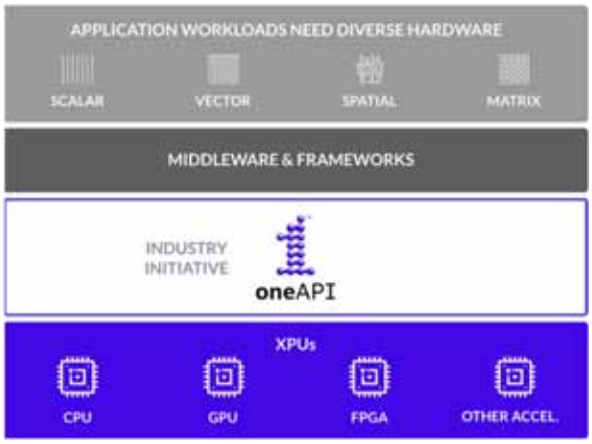
Heterogeneous Server Chip: Intel Alder Lake Hybrid CPU

AMD APU



oneAPI

- Cross-industry, open, standards-based unified programming model that delivers a common developer experience across accelerator architectures



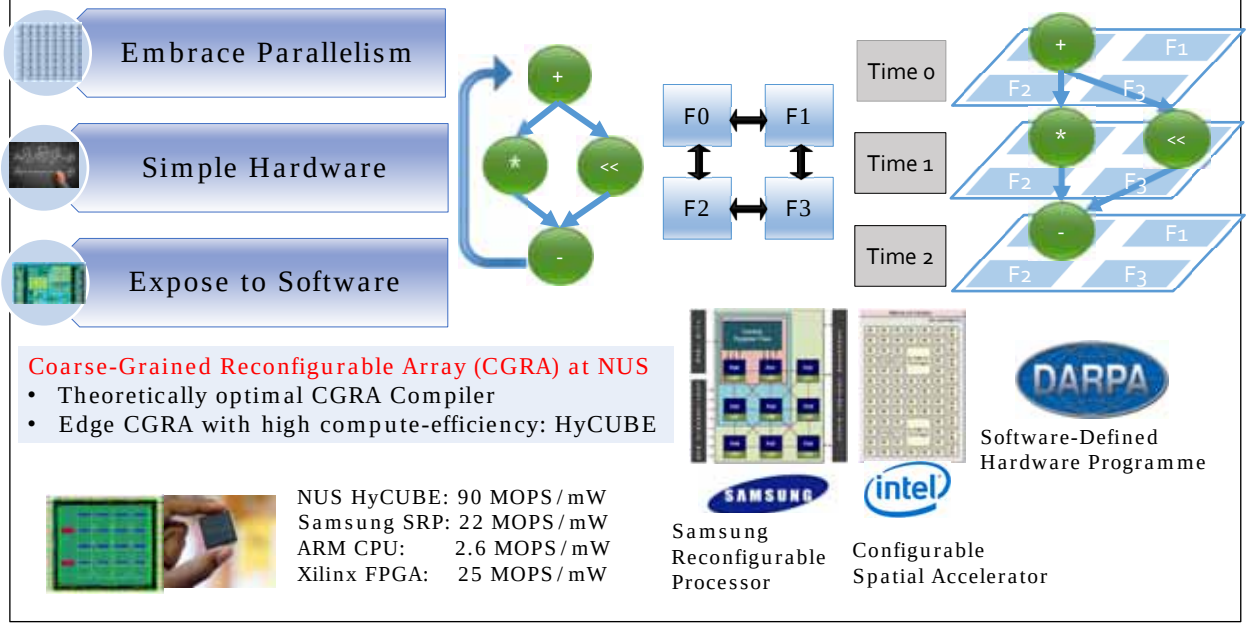
High-Performance Heterogeneous Computing

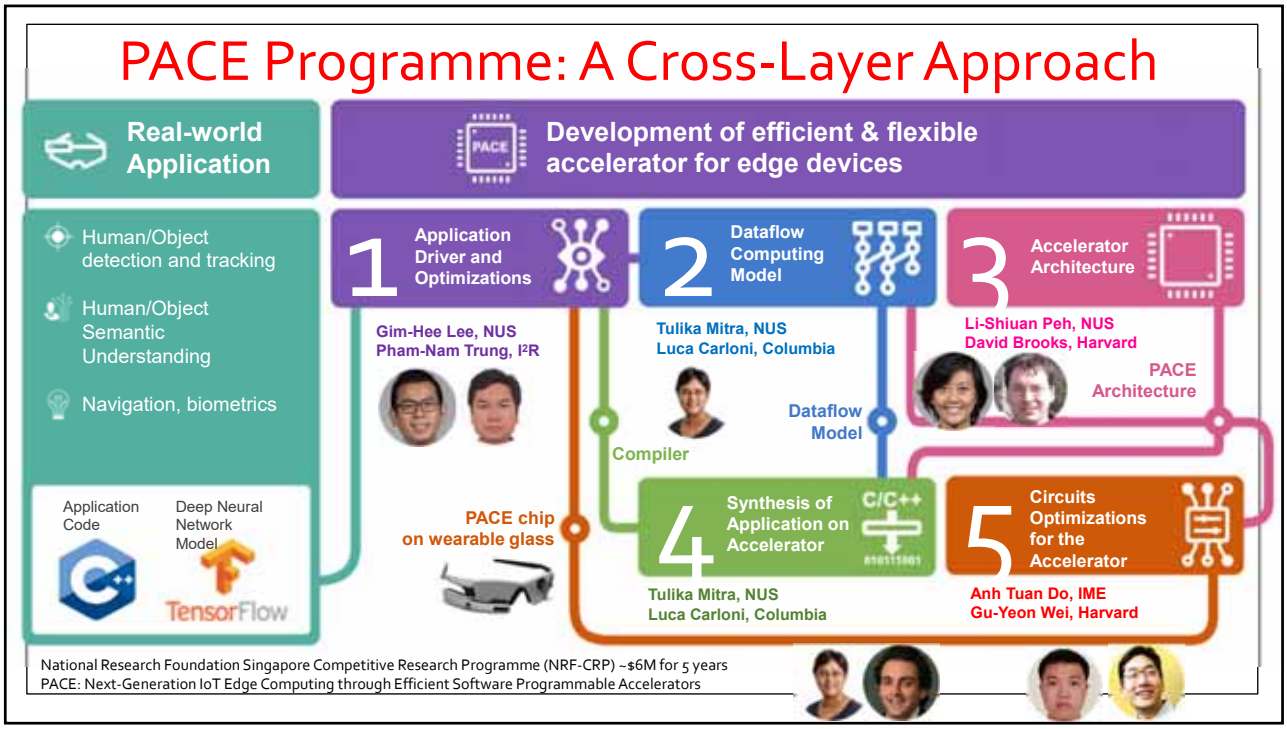
- Heterogeneous computing is becoming ubiquitous in high-performance computing
  - Laptop, desktop, servers
  - Single-chip, chiplets, multi-socket, data-center
- Software development for heterogeneous computing
  - Which core or accelerator to use for my application?
  - How to orchestrate application execution across multiple cores or accelerators?
  - How to run legacy code on heterogeneous computing platforms?
  - How to remain within TDP (thermal design power)?
  - How to reduce energy consumption?

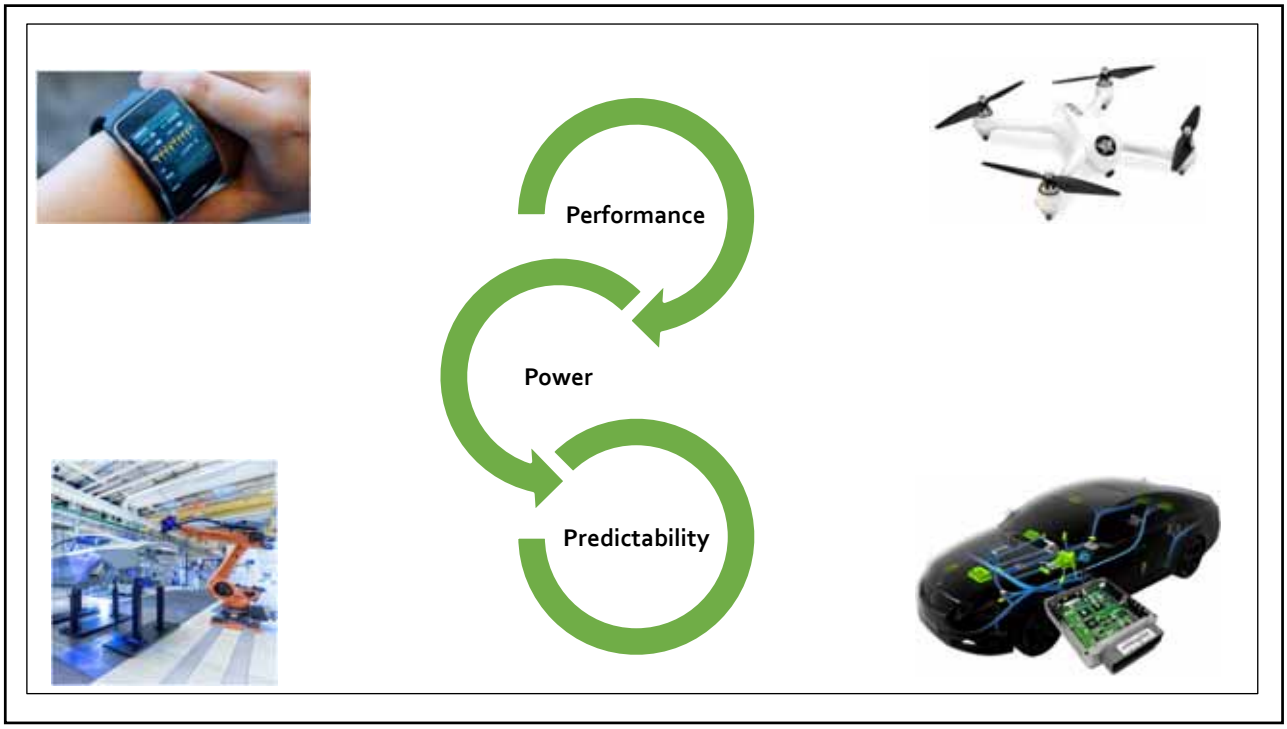
Domain-Specific to Domain-Agnostic Accelerator

- Universal accelerator with ASIC-like efficiency
- Instantiate domain-specific functionality through software

PACE: Let Software Define Hardware




PACE Programme: A Cross-Layer Approach
Real-world Application
Development of efficient & flexible accelerator for edge devices
Human/Object detection and tracking
Human/Object Semantic Understanding
Navigation, biometrics
Application Code
Deep Neural Network Model
TensorFlow
1 Application Driver and Optimizations
2 Dataflow Computing Model
3 Accelerator Architecture
Gim-Hee Lee, NUS
Pham-Nam Trung, I²R
Tulika Mitra, NUS
Luca Carloni, Columbia
Li-Shiuan Peh, NUS
David Brooks, Harvard
PACE Architecture
Dataflow Model
Compiler
PACE chip on wearable glass
4 Synthesis of Application on Accelerator
C/C++
5 Circuits Optimizations for the Accelerator
Tulika Mitra, NUS
Luca Carloni, Columbia
Anh Tuan Do, IME
Gu-Yeon Wei, Harvard
National Research Foundation Singapore Competitive Research Programme (NRF-CRP) ~$6M for 5 years
PACE: Next-Generation IoT Edge Computing through Efficient Software Programmable Accelerators


Performance
Power
Predictability

Acknowledgments

- Li-Shiuan Peh, NUS
- Jörg Henkel, KIT
- Sanjay Vishin, CSR/Qualcomm
- Thannirmalai Somu Muthukaruppan, NUS
- Mihai Pricopi, NUS
- Vanchinathan Venkataramani, NUS
- Anuj Pathania, NUS/KIT
- Alok Prakash, NUS
- Siqi Wang, NUS
- Manupa Karunaratne, NUS
- Alexandru Eugen Irimiea, NUS
- Tan Cheng, NUS
- Aditi Kulkarni, NUS
- Wang Bo, NUS
- Muhammad Shafique, KIT
- Hussam Amrouch, KIT

- Ministry of Education Singapore
- National Research Foundation
- Cambridge Silicon Radio (CSR)
- ARM
- Facebook Research
- Huawei Research
- Xilinx Research

Thank you!

Image credit: Madam Studio, WoodsBagot