ECE M216A

Lecture 20

Future SoCs

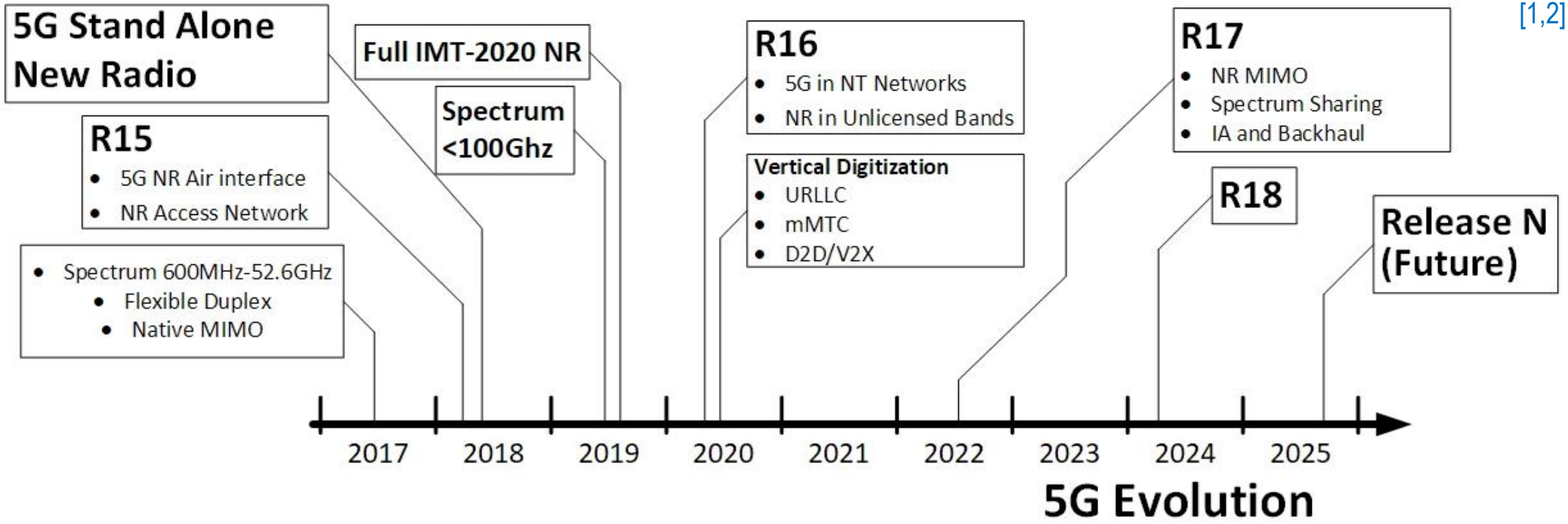
Prof. Dejan Marković

UCLA ECE Department

dejan@ucla.edu

Evolving Standards Demand *Flexibility and Efficiency*

Benefits: lower development cost and shorter time-to-market



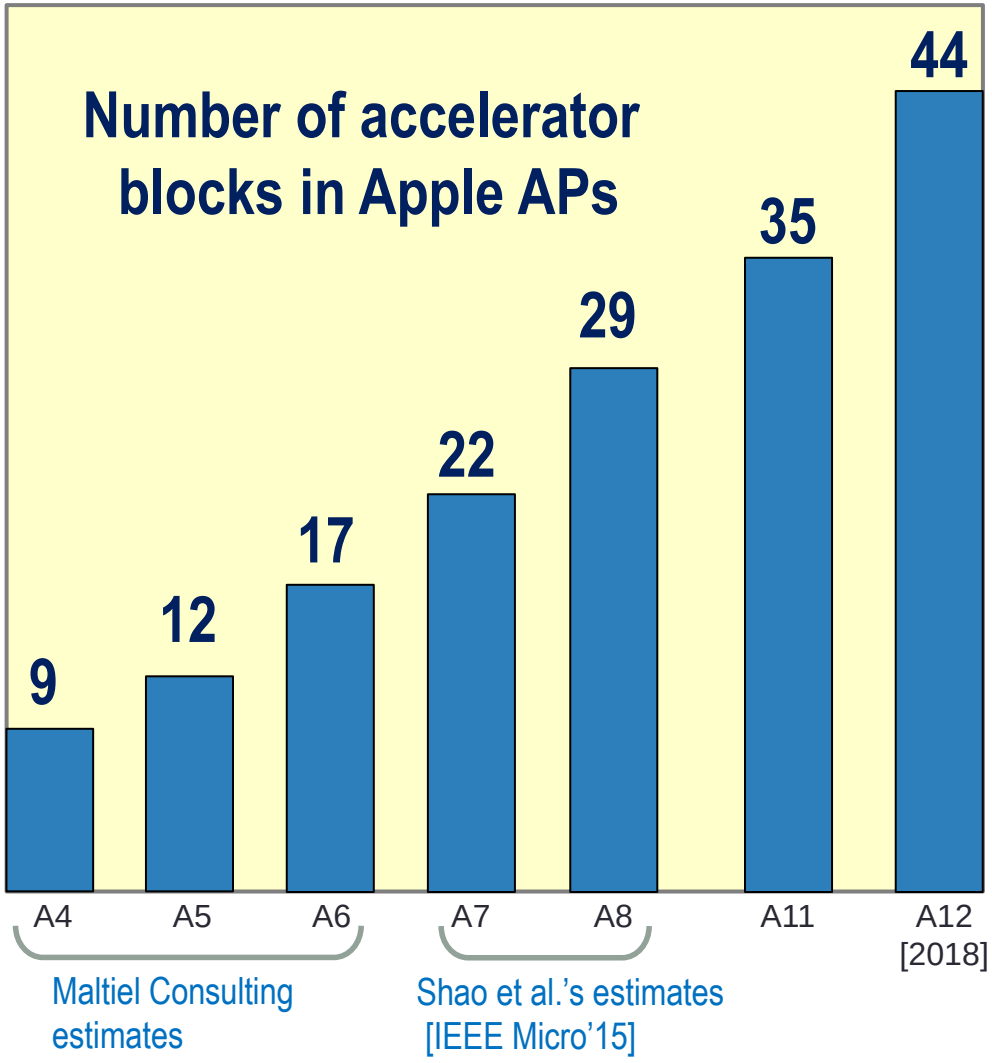
[1,2]

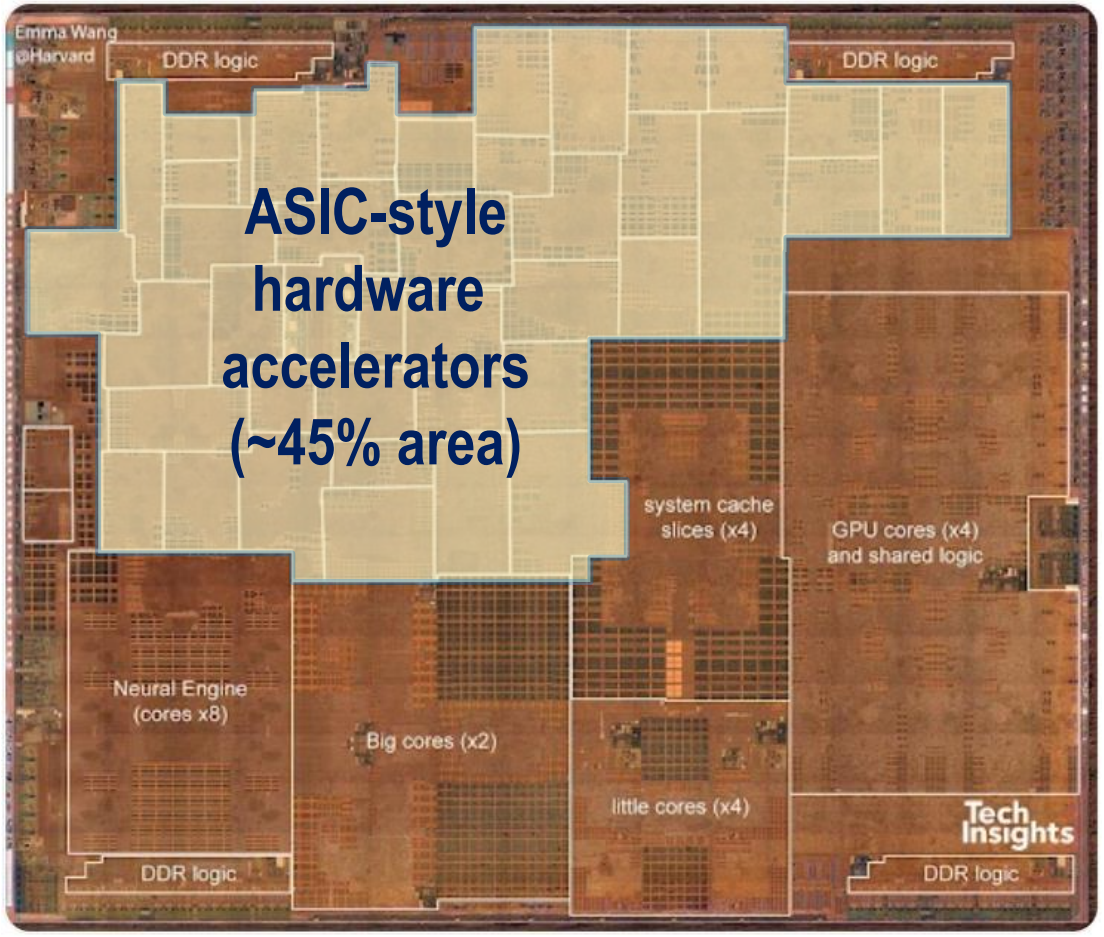
- SoC/ASIC revision at each iteration is **$$$** (~$106M in 16nm CMOS)
- Long design cycles (>1 year)[3] with increasing design complexity

[1] Xilinx Zynq RFSoC DFE; [2] 3GPP : 3GPP.org; [3] Jason Cong et al., IEEE design and test of computers, 2011

SoC (CPU + ASIC blocks) Today: The Apple Approach

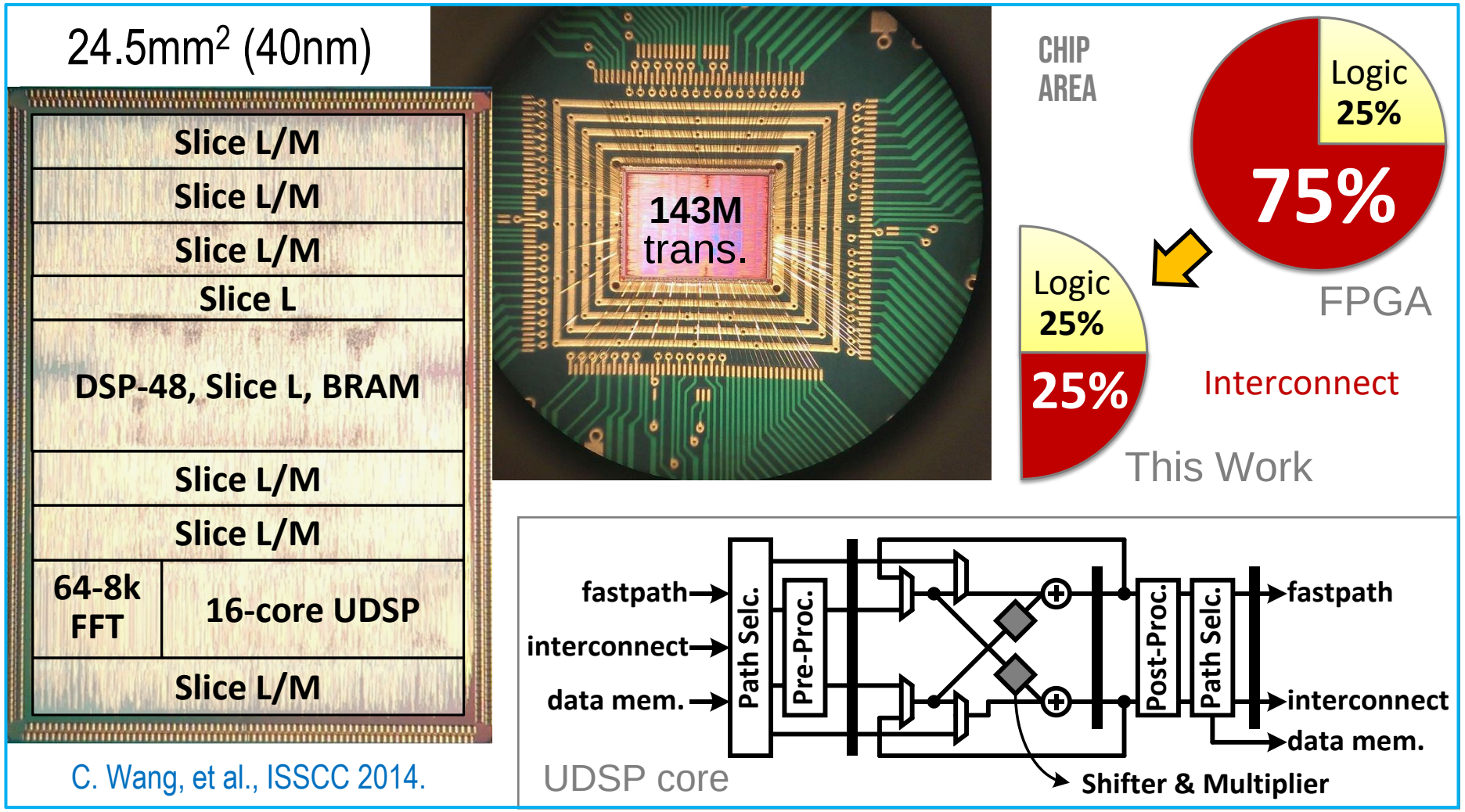
Increasing "dark silicon" area (A12: ~45%, A15: ~55%), <10% chip is active





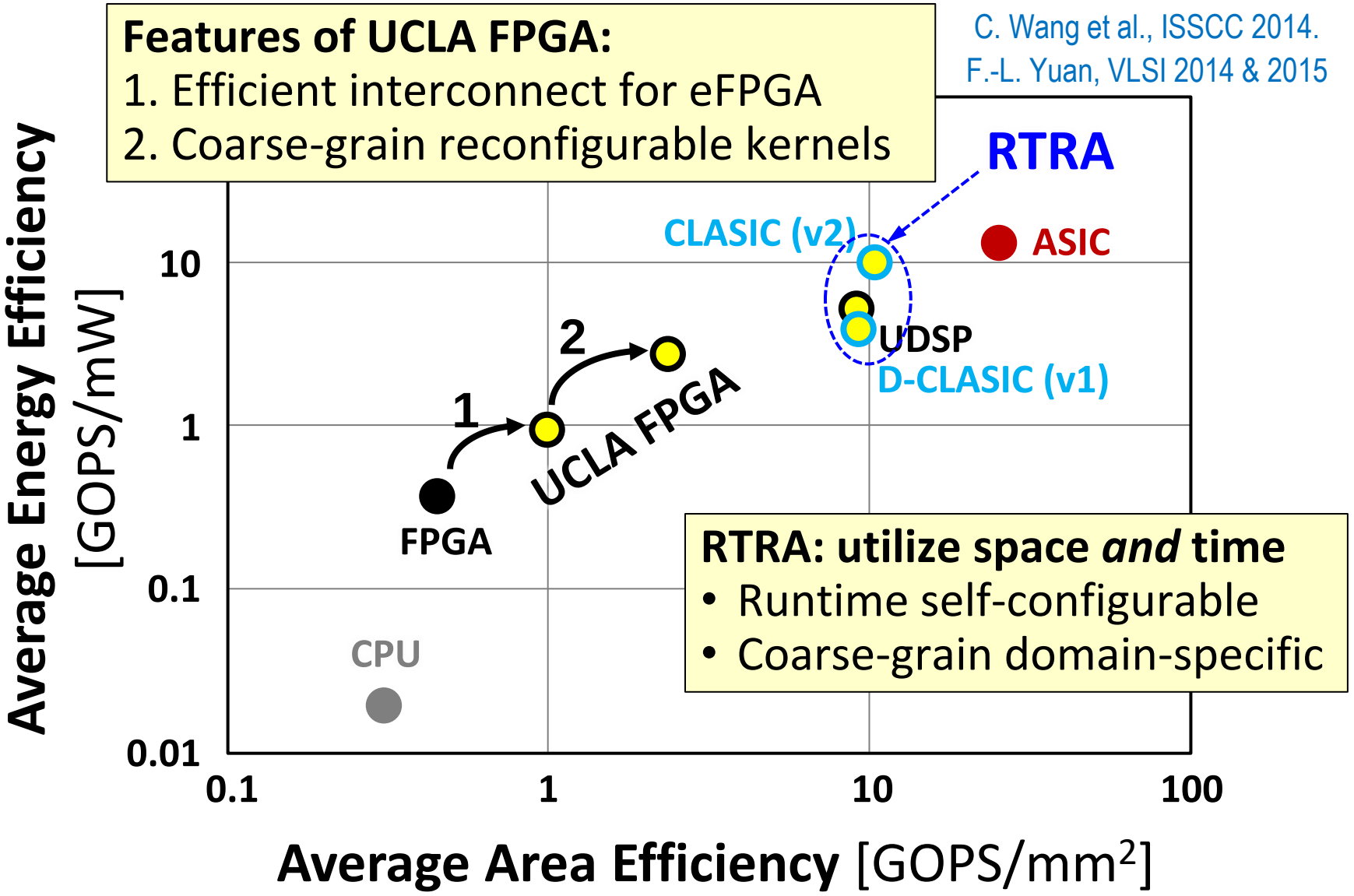
Apple A12 die photo

PoC for Embedded FPGA (Seeded Flex Logix, Inc.)

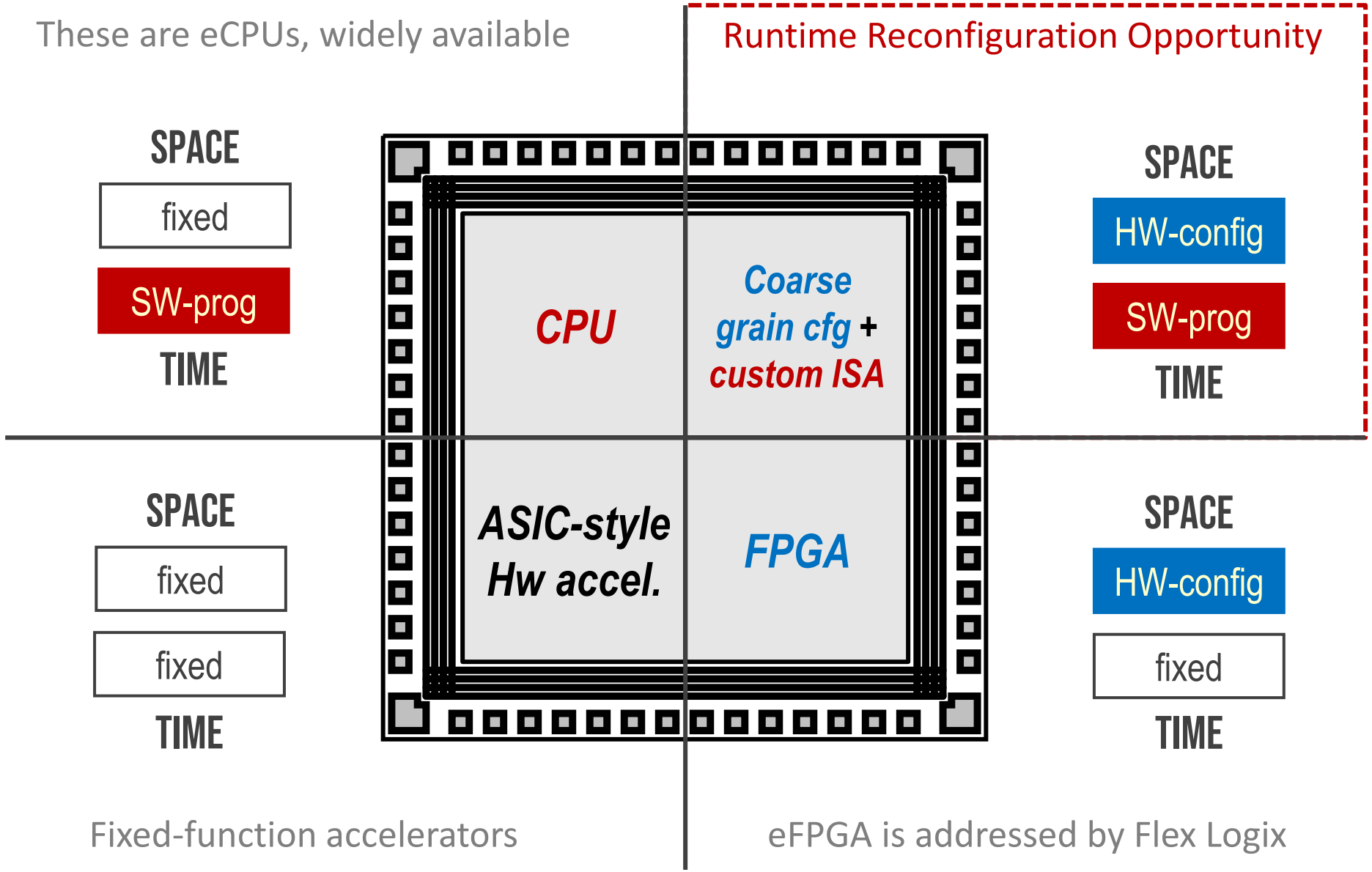


2014 ISSCC Lewis Winner Award for Outstanding Paper

Efficiency *and* Flexibility in Communications DSP



Space-Time Quadrants of Architecture “X”



UDSP @ ISSCC 2022

SESSION 2 Live Q&As **Monday February 21st, 8:30 AM PST**

Processors

Session Chair: Hugh Mair, *Mediatek, Fairview, TX*
Session Co-Chair: Shidhartha Das, *ARM, Cambridge, United Kingdom*

8:30 AM

2.1 Ponte Vecchio: A Multi-Tile 3D Stacked Processor for Exascale Computing
W. Gomes[1], A. Koker[2], P. Stover[3], D. Ingerly[1], S. Siers[2], S. Venkataraman[4], C. Pelto[1], T. Shah[5], A. Rao[2], F. O'Mahony[1], E. Karl[1], L. Cheney[2], I. Rajwani[2], H. Jain[4], R. Cortez[2], A. Chandrasekhar[4], B. Kanthi[4], R. Koduri[6], [1]Intel, Portland, OR; [2]Intel, Folsom, CA; [3]Intel, Chandler, AZ; [4]Intel, Bengaluru, India [5]Intel, Austin, TX; [6]Intel, Santa Clara, CA

8:40 AM

2.2 Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor
N. Nassif[1], A. O. Munch[1], C. L. Molnar[1], G. Pasdast[2], S. V. Iyer[2], Z. Yang[1], O. Mendoza[1], M. Huddart[1], S. Venkataraman[3], R. Marom[4], A. M. Kern[1], B. Bowhill[1], D. R. Mulvihill[5], S. Nimmagadda[3], V. Kalidindi[3], J. Krause[1], M. M. Haq[1], R. Sharma[1], K. Duda[5], [1]Intel, Hudson, MA; [2]Intel, Santa Clara, CA [3]Intel, Bangalore, India; [4]Intel, Haifa, Israel; [5]Intel, Fort Collins, CO

8:50 AM

2.3 IBM Telum: A 16-Core 5+ GHz DCM
O. Geva[1], C. Berry[1], R. Sonnelitter[1], D. Wolpert[1], A. Collura[1], T. Strach[2], D. Phan[1], C. Lichtenau[2], A. Buyuktosunoglu[3], H. Harrer[2], J. Zitz[1], C. Marquart[1], D. Malone[1], T. Webel[2], A. Jatkowski[1], J. Isakson[4], D. Hamid[1], M. Cichanowski[4], M. Romain[1], F. Hasan[4], K. Williams[1], J. Surprise[1], C. Cavitt[1], M. Cohen[1]
[1]IBM Systems and Technology, Poughkeepsie, NY
[2]IBM Systems and Technology, Boeblingen, Germany
[3]IBM Research, Yorktown Heights, NY; [4]IBM Systems and Technology, Austin, TX

9:00 AM

2.4 POWER10™: A 16-Core SMT8 Server Processor with 2TB/s Off-Chip Bandwidth in 7nm Technology
R. M. Rao[1], C. Gonzalez[2], E. Fluhr[3], A. Mathews[3], A. Bianchi[3], D. Dreps[3], D. Wolpert[4], E. Lai[3], G. Strevig[3], G. Wiedemeier[3], P. Salz[5], R. Kruse[3], [1]IBM, Bengaluru, India; [2]IBM, Yorktown Heights, NY [3]IBM, Austin, TX; [4]IBM, Poughkeepsie, NY; [5]IBM, Boblingen, Germany

9:10 AM

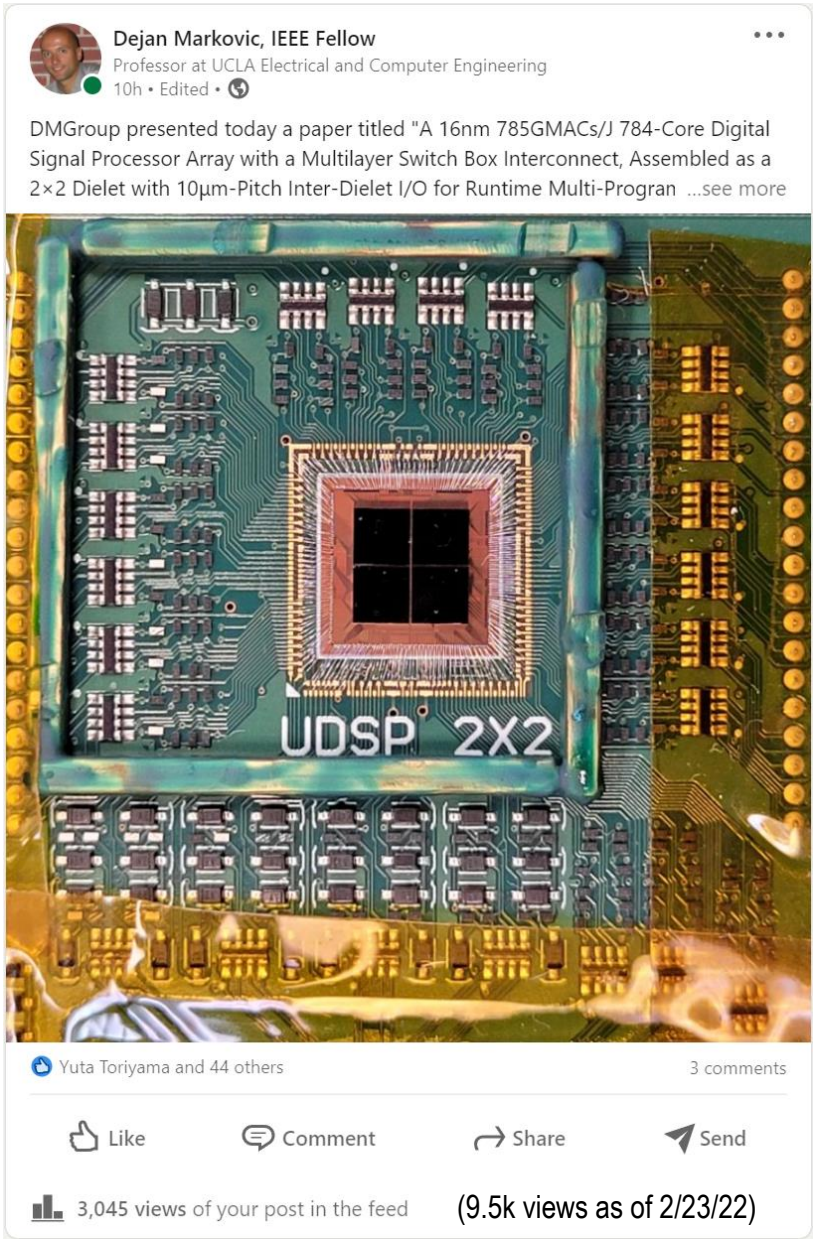
2.5 A 5nm 3.4GHz Tri-Gear ARMv9 CPU Subsystem in a Fully Integrated 5G Flagship Mobile SoC
A. Nayak[1], H. Chen[1], H. Mair[1], R. Lagerquist[1], T. Chen[1], A. Rajagopalan[1], G. Gammie[1], R. Madhavaram[1], M. Jagota[1], C. Chung[1], J. Wiedemeier[1], B. Meera[1], C-Y. Yeh[2], M. Lin[2], C. Lin[2], V. Lin[2], J. Lin[2], Y. Chen[2], B. Chen[2], C-Y. Wu[2], R. ChangChien[2], R. Tzeng[2], K. Yang[2], A. Thippana[1], E. Wang[2], S. Hwang[2]
[1]MediaTek, Austin, TX; [2]MediaTek, Hsinchu, Taiwan

9:20 AM

2.6 A 16nm 785GMACs/J 784-Core Digital Signal Processor Array with a Multilayer Switch Box Interconnect, Assembled as a 2×2 Dielet with 10µm-Pitch Inter-Dielet I/O for Runtime Multi-Program Reconfiguration
U. Rathore, S. S. Nagi*, S. Iyer, D. Marković,* *Equally-Credited Authors (ECAs)
University of California, Los Angeles, CA

9:30 AM

2.7 Zen3: The AMD 2nd-Generation 7nm x86-64 Microprocessor Core
DS2 *T. Burd[1], W. Li[1], J. Pistole[1], S. Venkataraman[1], M. McCabe[1], T. Johnson[1], J. Vinh[1], T. Yiu[1], M. Wasio[1], H-H. Wong[1], D. Lieu[1], J. White[2], B. Munger[2], J. Lindner[2], J. Olson[2], S. Bakke[2], J. Sniderman[2], C. Henrion[3], R. Schreiber[4], E. Busta[3], B. Johnson[3], T. Jackson[3], A. Miller[3], R. Miller[3], M. Pickett[3], A. Horiuchi[3], J. Dvorak[3], S. Balagangadharan[5], S. Ammikkallingal[5], P. Kumar[5]*
[1]AMD, Santa Clara, CA; [2]AMD, Boxborough, MA; [3]AMD, Fort Collins, CO; [4]AMD, Austin, TX
[5]AMD, Bangalore, India



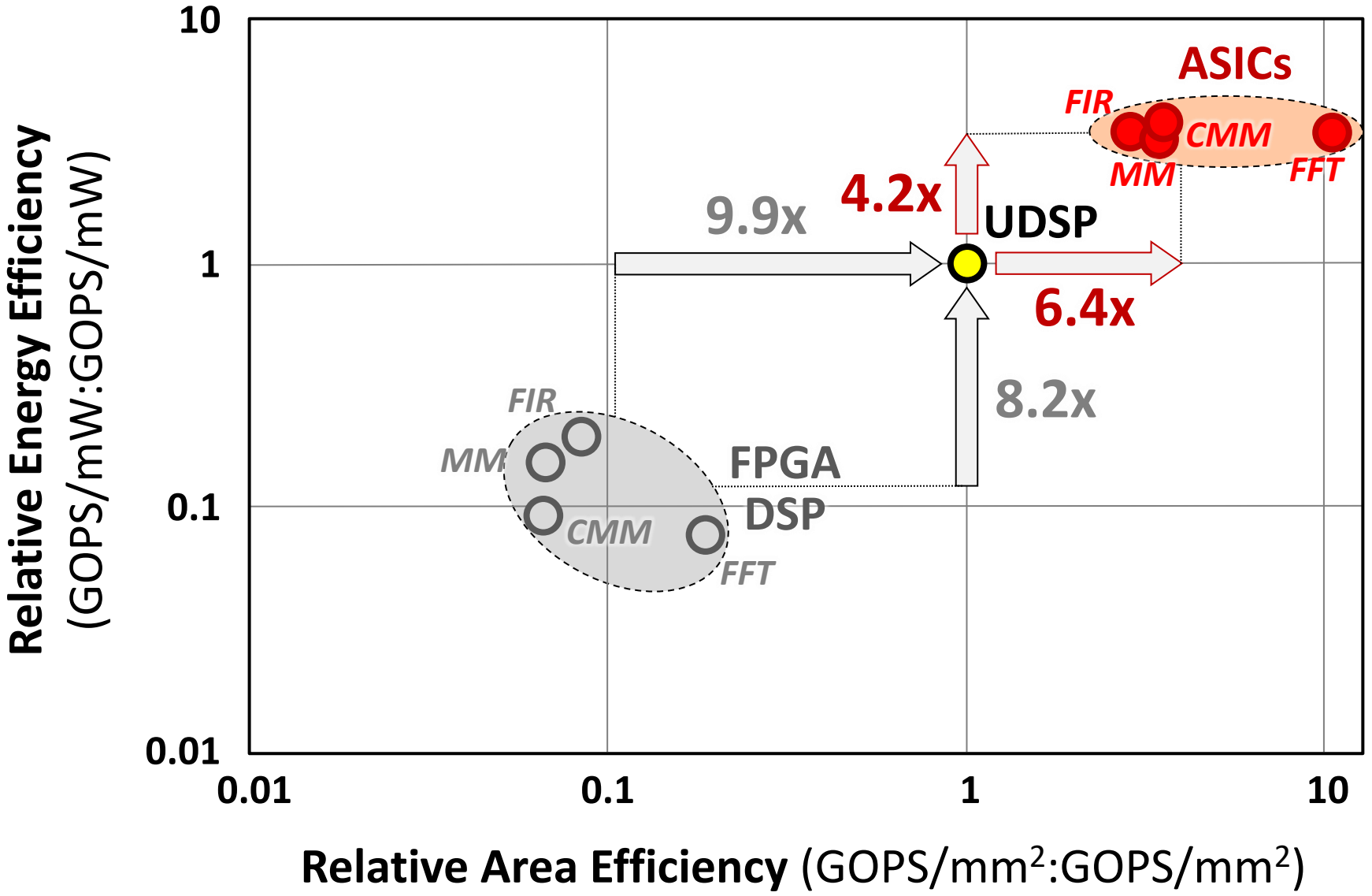
(9.5k views as of 2/23/22)

The only university paper in industry driven session

3,045 views 1st 10 hrs
123 Intel
83 Apple
42 Qualcomm
22 NVIDIA
21 Google
20 Broadcom
20 Cadence

355 SF Bay Area
72 Portland, OR
48 Austin, TX
45 NYC
42 San Diego

Chip Measurements: FPGA vs. UDSP vs. ASIC



1.1 GHz UDSP

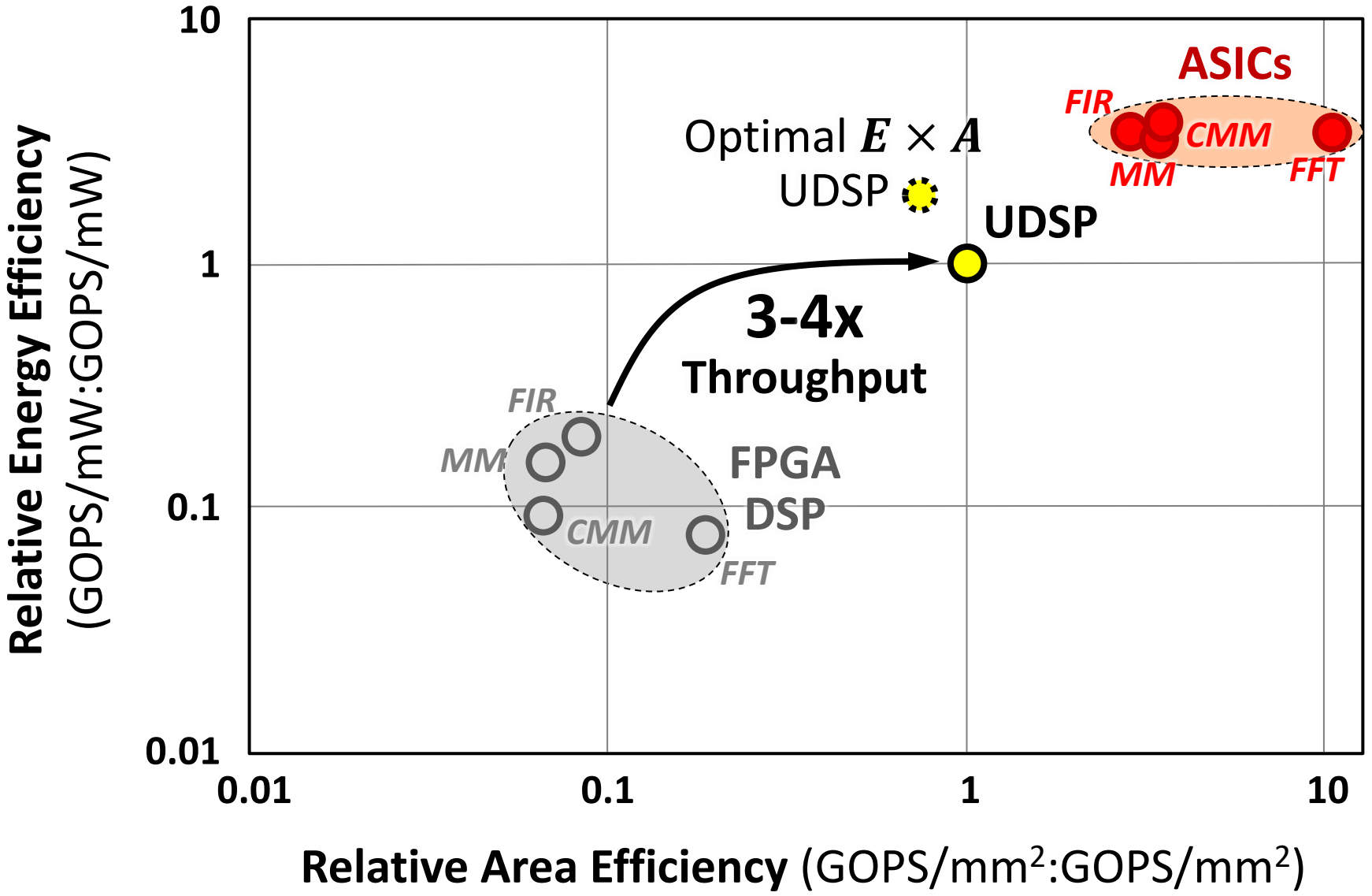
~10× better energy and area efficiency than FPGA

~5× away from ASIC

x/y-axes relative to
~70 GMAC/s/mm²
~310 GMAC/s/W

U. Rathore, et al., ISSCC 2022 (Paper 2.6)

Chip Measurements: FPGA vs. UDSP vs. ASIC



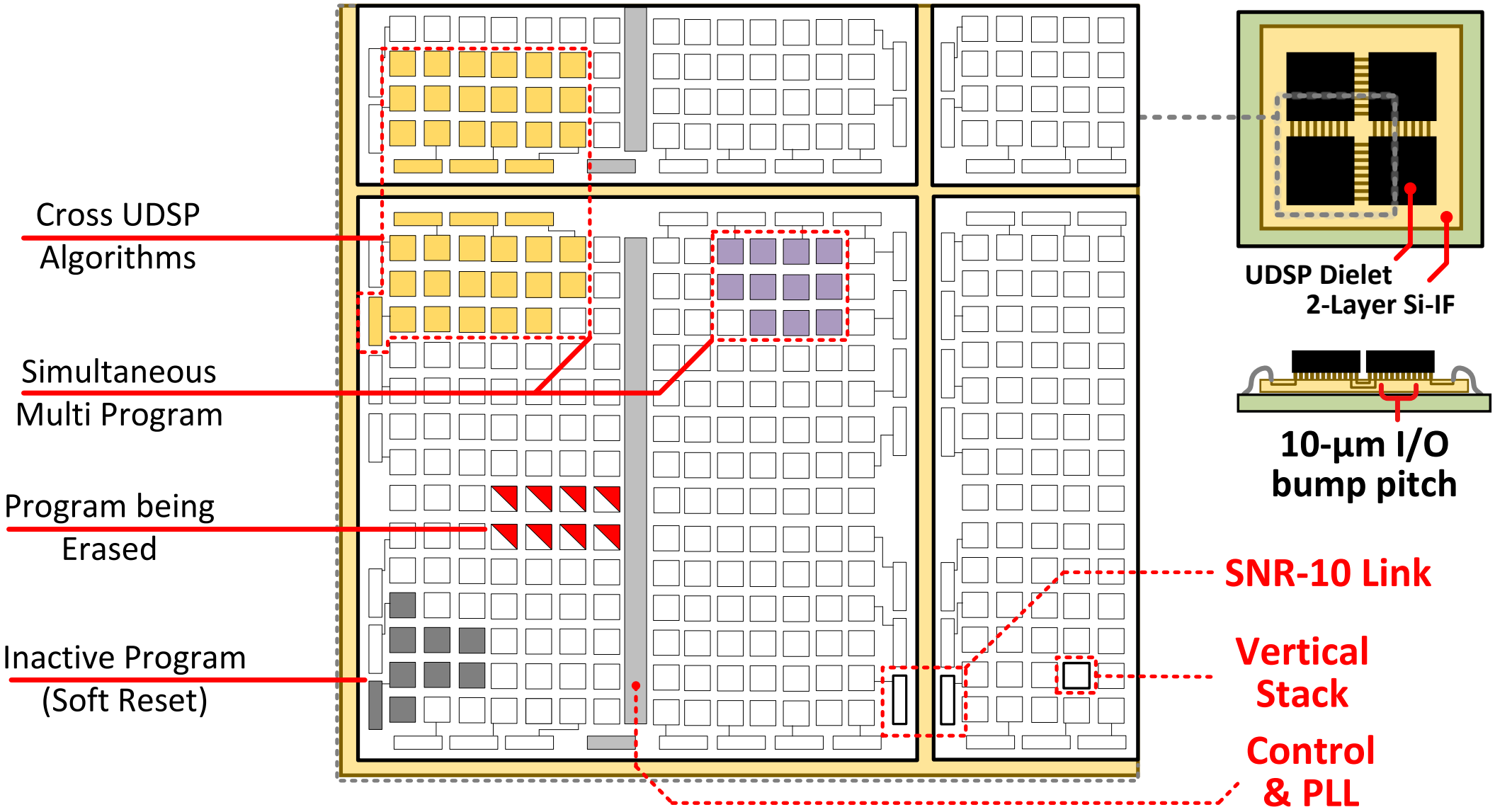
At 1.1 GHz, UDSP is 3-4× faster than FPGA

At min E×A product ~2× higher energy efficiency, at **~30%** lower throughput

x/y-axes relative to
~70 GMAC/s/mm²
~310 GMAC/s/W

U. Rathore, et al., ISSCC 2022 (Paper 2.6)

UDSP Multi-Chip and Multi-Program Compile

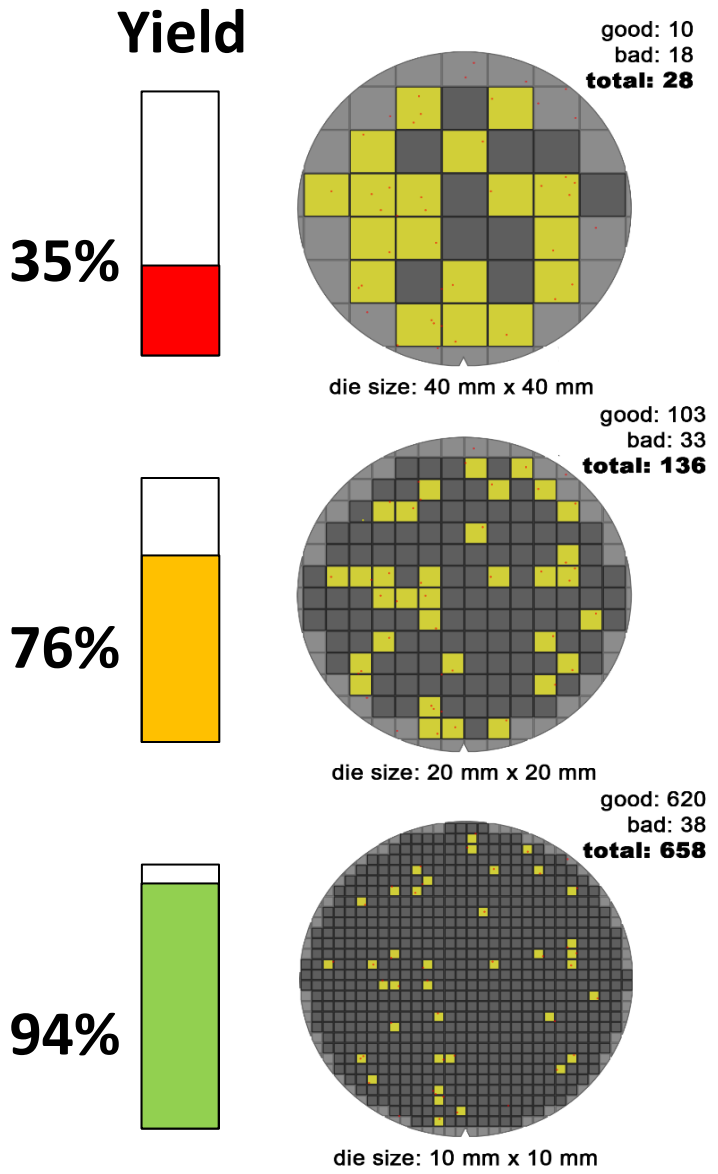


Efficient Multi-Chip Module (MCM) Scaling

- **Large SoCs incur higher cost:**
  - Lower yield of larger chips
  - Delayed time to market

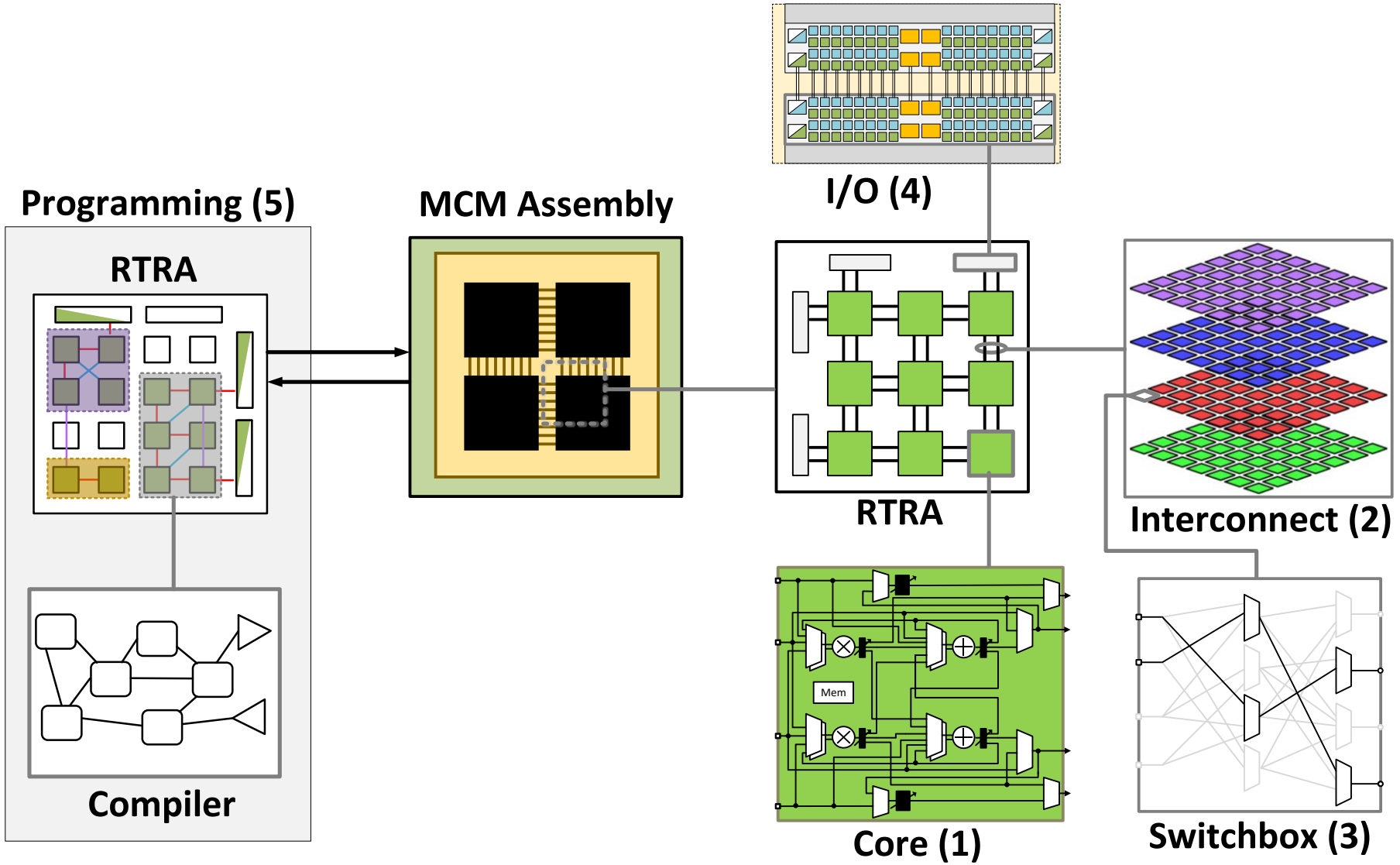
- **Cost benefits of MCM scaling:**
  - Smaller chips give better $\textbf{Yield } \alpha \frac{1}{1+D*Area^{1\sim3}}$
  - AMD 2990X 4x213 mm² → $1.8K
  - Nvidia V100 815 mm² → $8.2K

- **Challenges with MCM design:**
  - High bandwidth density, low link latency
  - High transfer energy and area efficiency



Universal Digital Signal Processor (UDSP) Overview

Co-design of compute, interconnect, I/O network and channel, compilers, package

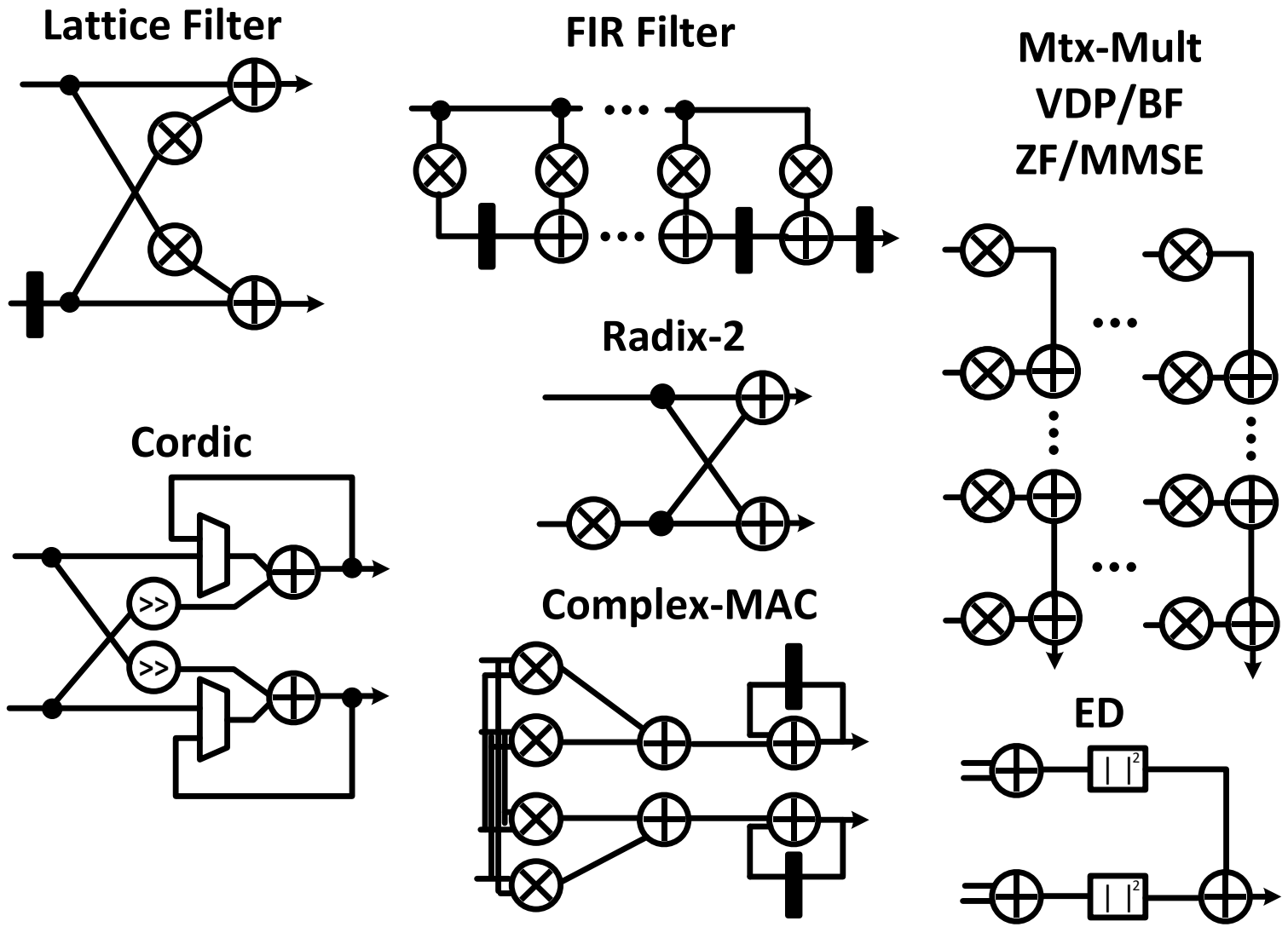


(1) Study of Algorithm Ontology: Example DSP Kernels

Example Kernels derived from common DSP algorithms:

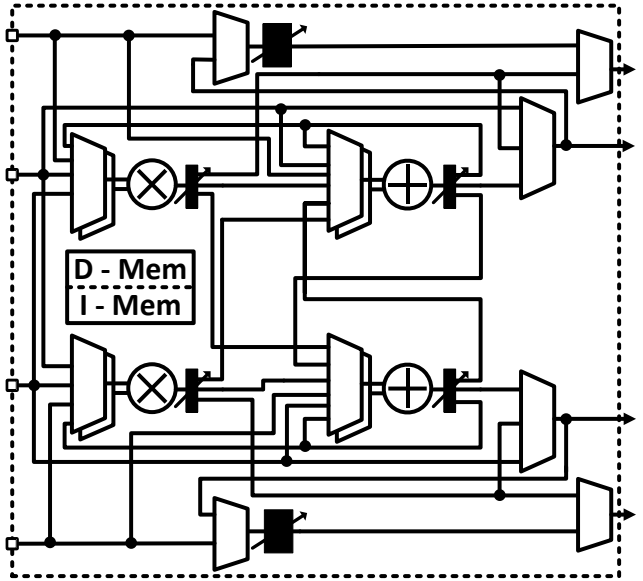
- DUC/DDC
- MIMO
- Adaptive Filtering
- IFFT/FFT
- NN Inference
- Spectrum Sensing

DUC / DDC : Digital Up/Down Conversion
MIMO : Multiple Input Multiple Output
IFFT / FFT: Inverse/ Fast Fourier Transform
NN : Neural Network
ZF : Zero Forcing
MMSE : Minimum Mean Squared Estimation
VDP : Vector Dot Product
MAC : Multiply Accumulate
FIR : Finite Impulse Response
ED : Euclidian Distance



(1) Iterative Process of Core Design

UDSP Core v4.2



16-bit fixed-point data paths

1.1 GHz clock frequency

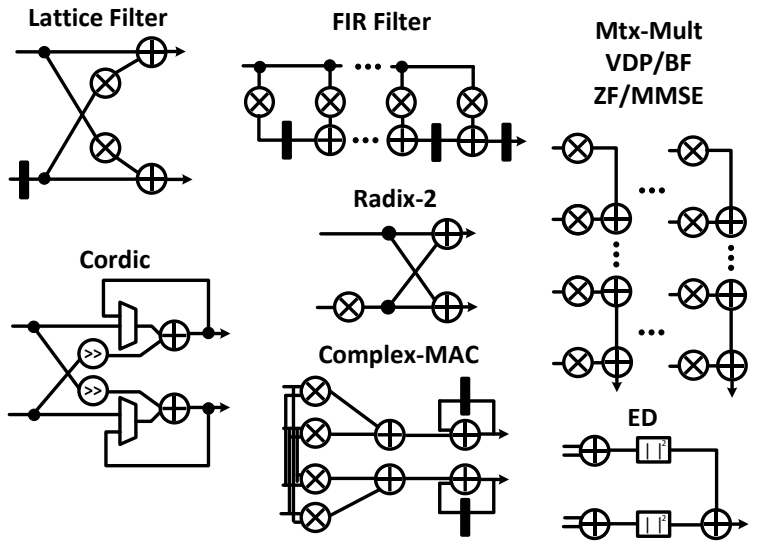
256b D-Mem, 384b I-Mem/I

4 Inputs, 4 Outputs

Connection Adjust

Mapping

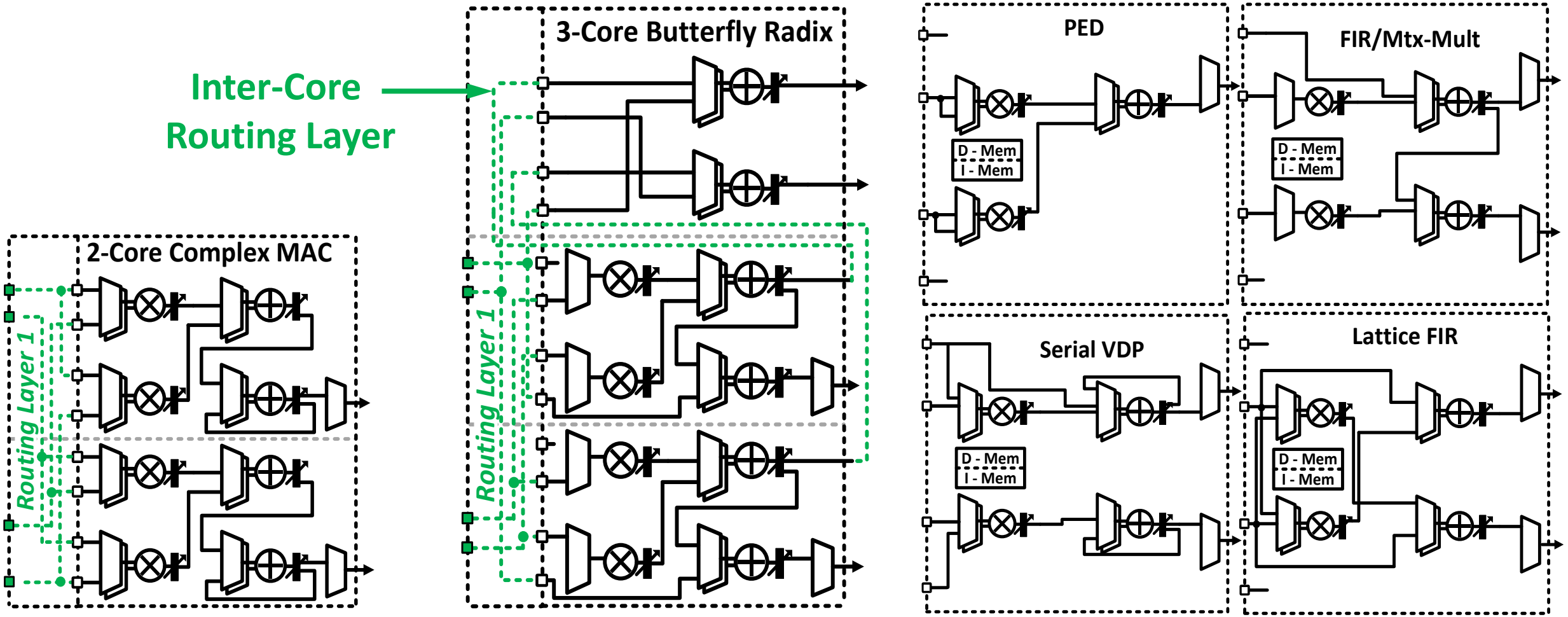
DSP Kernels



Balancing *core granularity* and *core utilization* to maximize efficiency

(1) Example Common Core Configurations

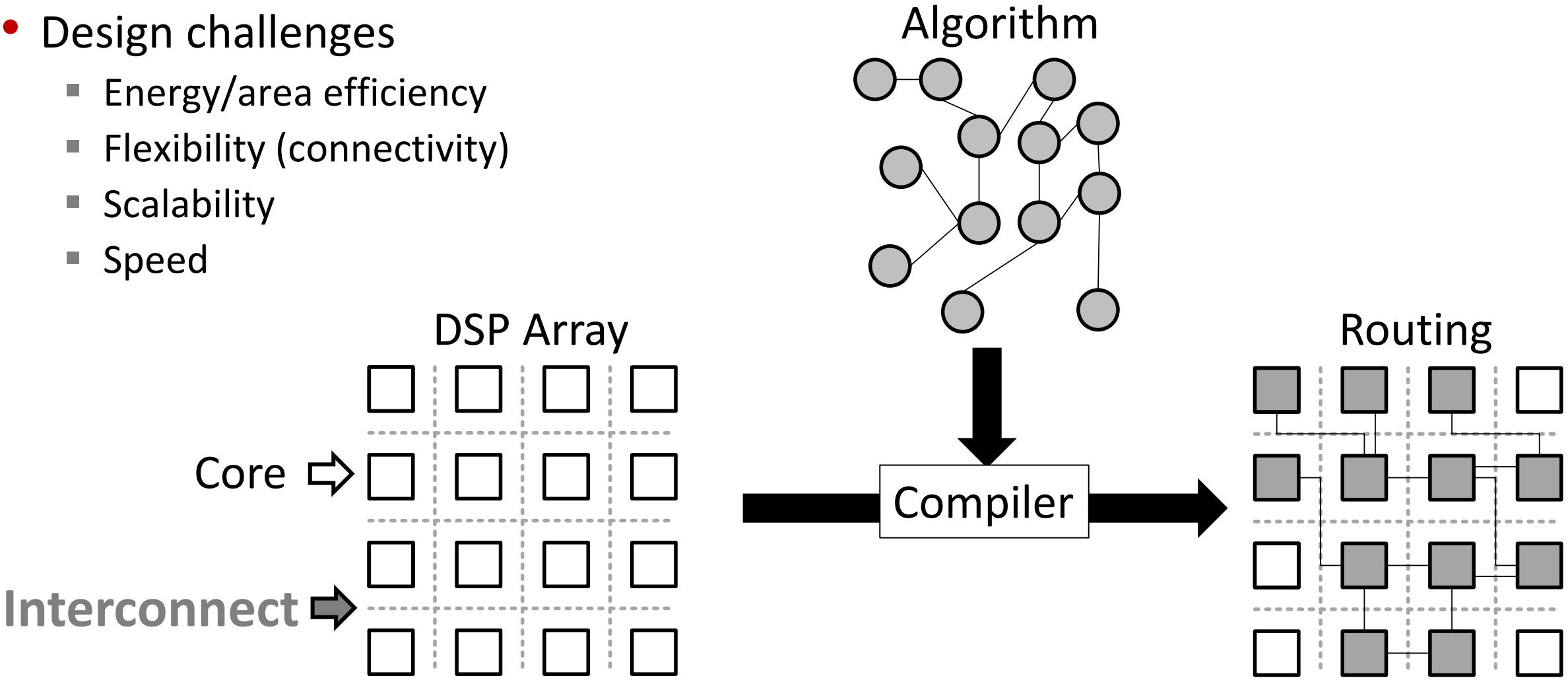
Preserve common sub-structure of DSP graphs, concatenate cores for larger kernels



(2) Interconnects: An Exercise in Co-design

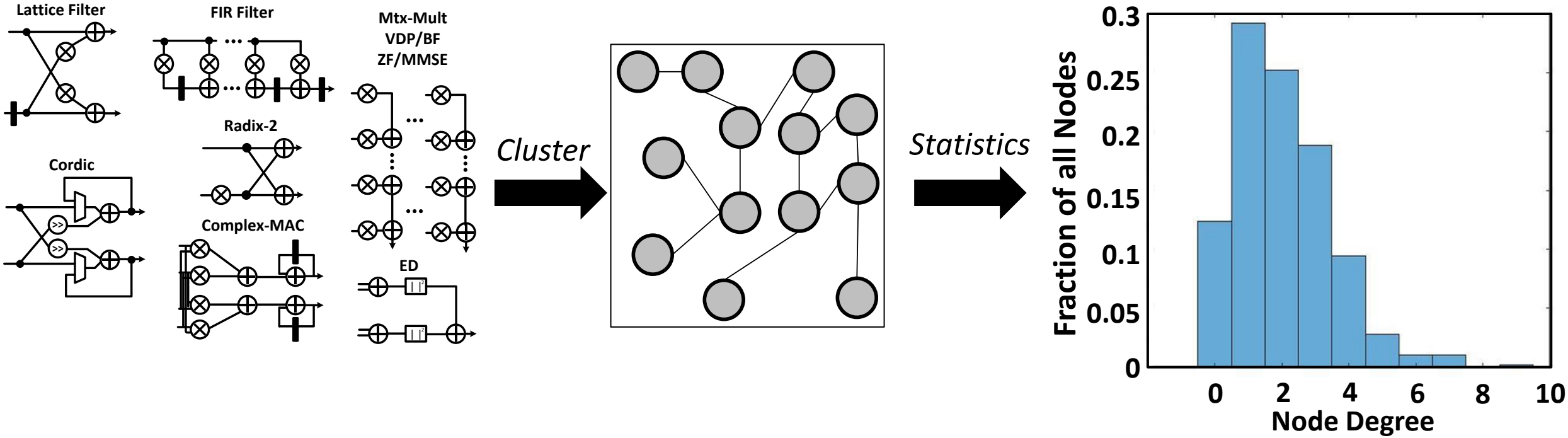
Interconnect design is linked to algorithm and routing

- Design challenges
  - Energy/area efficiency
  - Flexibility (connectivity)
  - Scalability
  - Speed



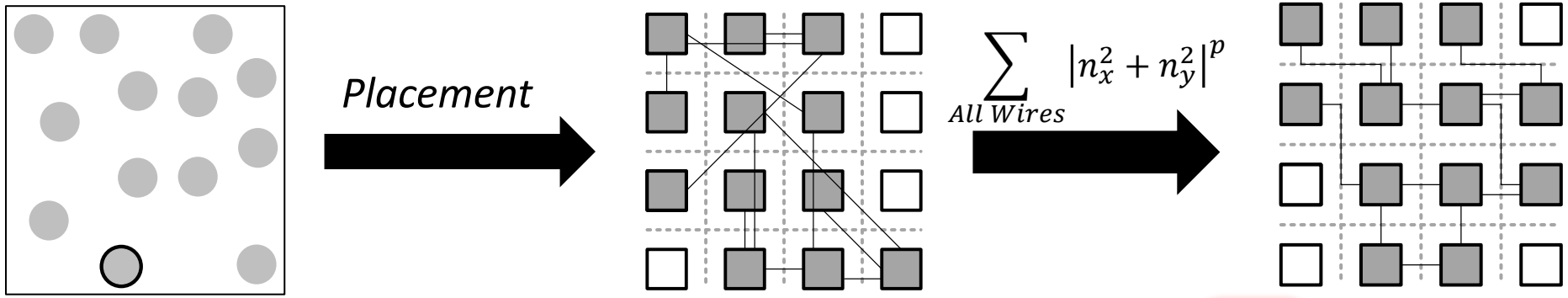
(2) Core Cluster Statistics

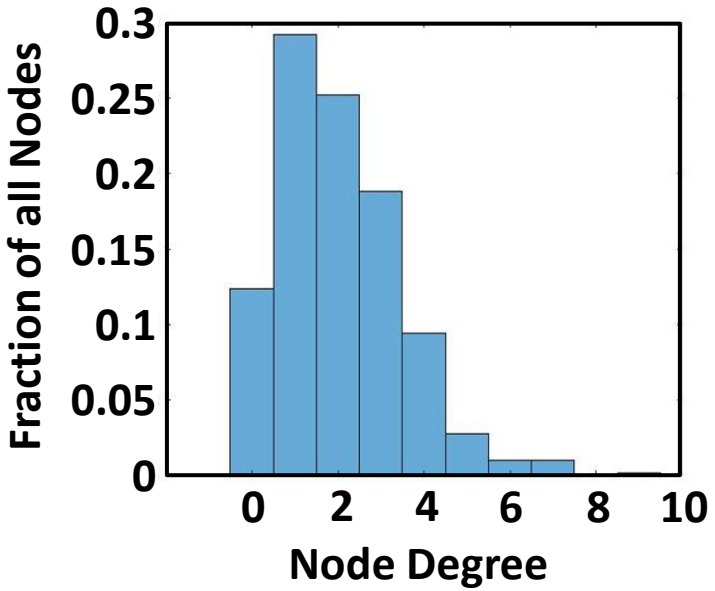
- **(Graph Space/Time) Correlation in DSP Algorithms**
  - Limited dependency (spatial and temporal) of nodes
  - Limited degree of nodes ~2-3

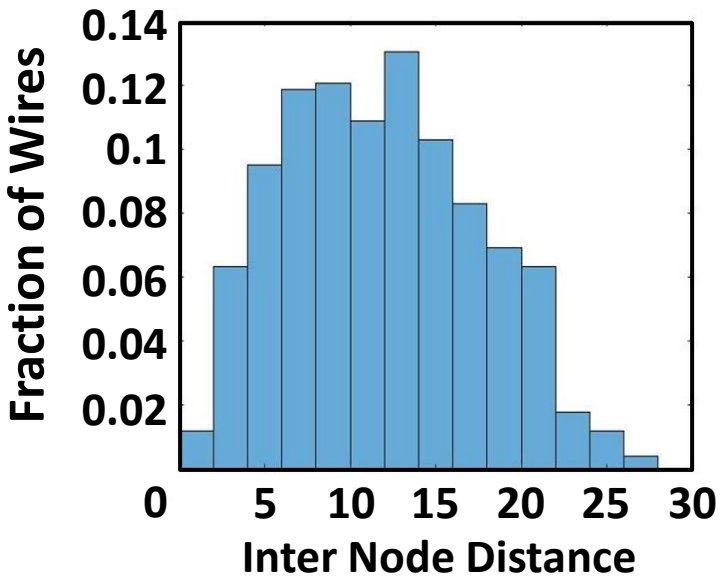


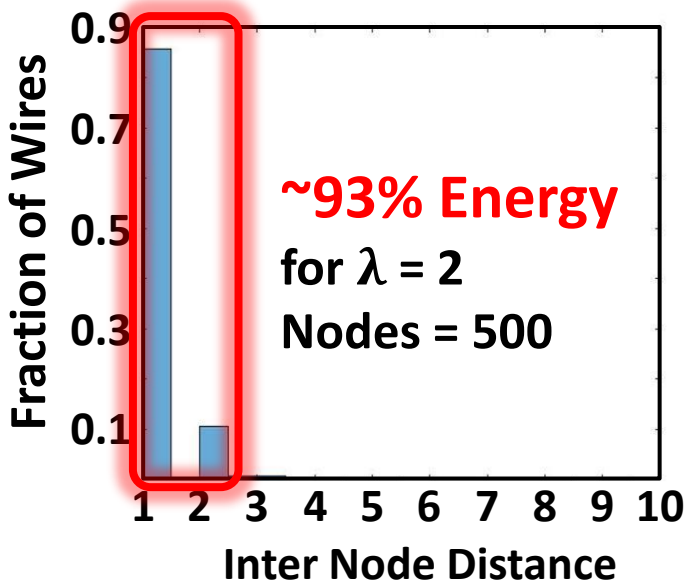
(2) Modeling Connection Statistics using BRG

- Bernoulli random graph model → Poisson degree distribution (mean λ)
- Manhattan Sorted → Exponential[2] (mean μ)





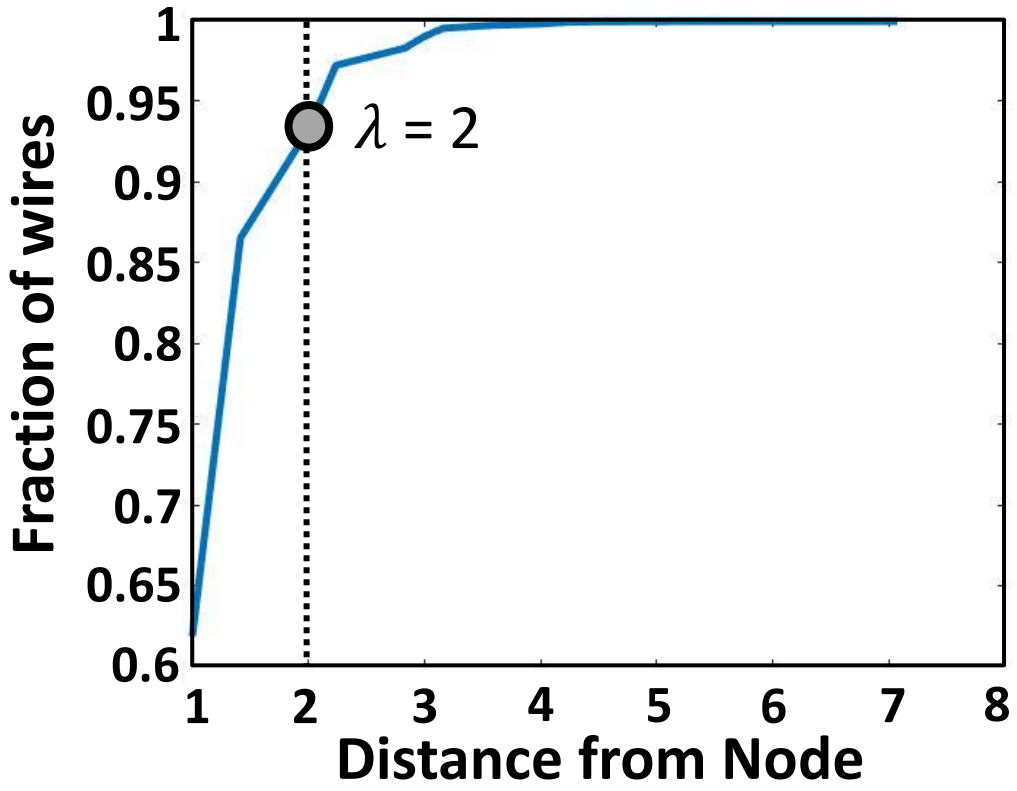




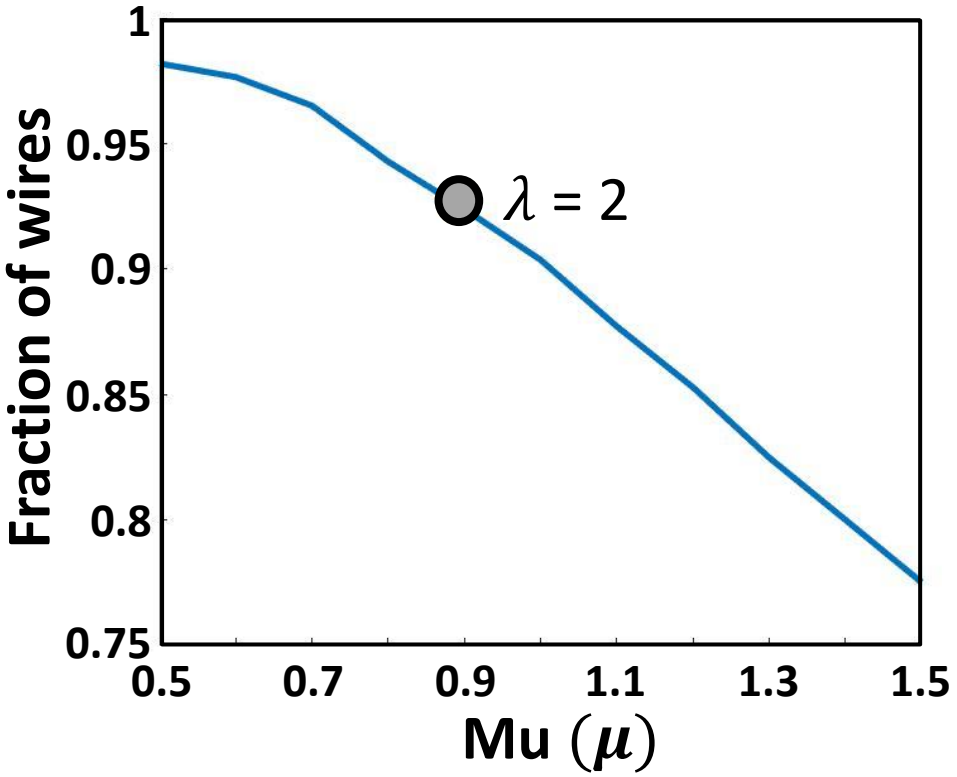
(2) Wire Length Coverage Determines Node Distance

- **Fraction of wire distances less than equal to a certain distance**
  - Concentrated energy in $distance \leq 2$

CDF (Wire Distance)

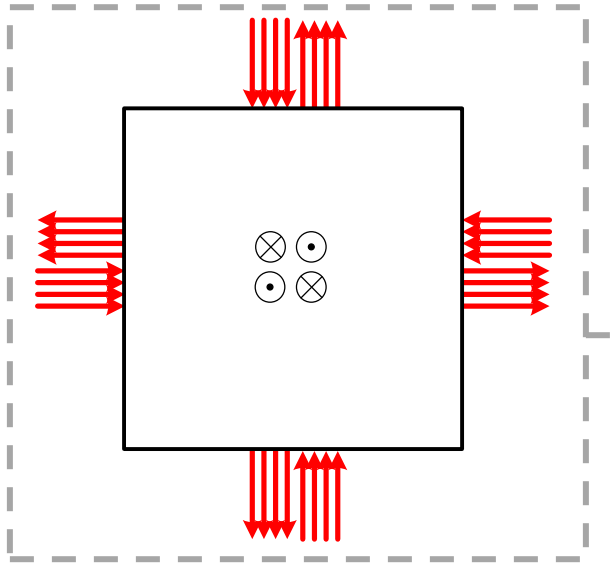


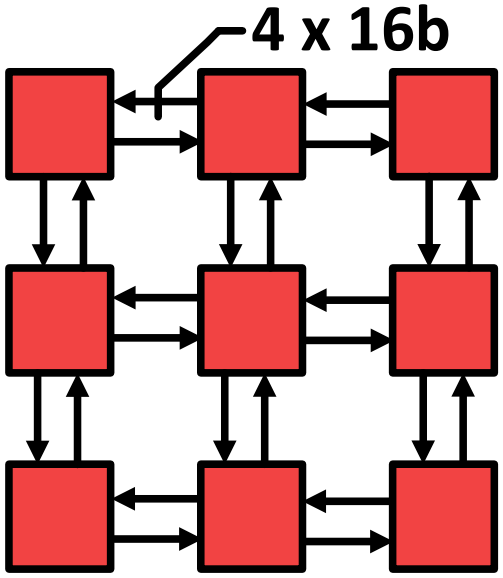
Probability of coverage for distance $1,\ \sqrt{2},\ 2$



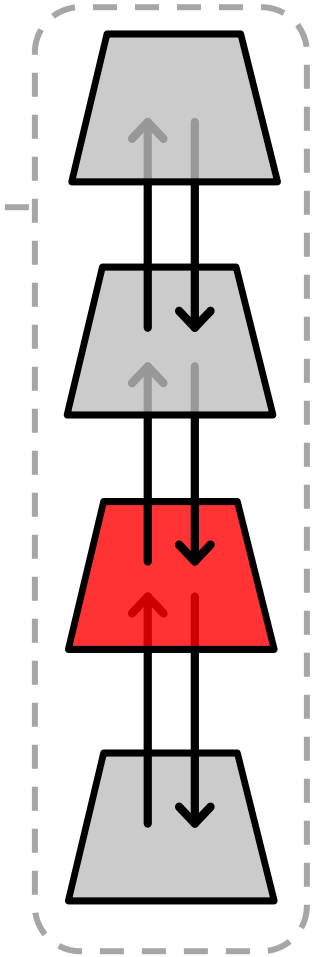
U. Rathore, et al., ISSCC 2022 (Paper 2.6)

(2) UDSP Interconnect Architecture – Layer 1

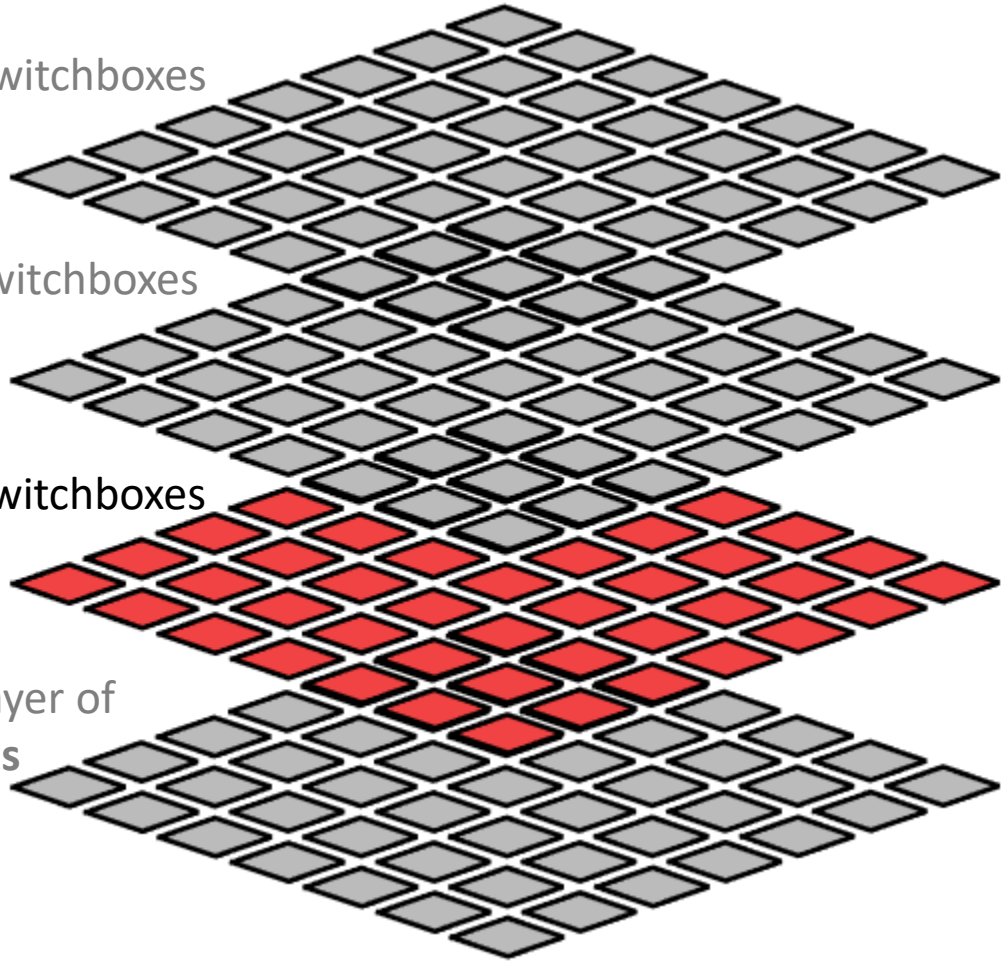




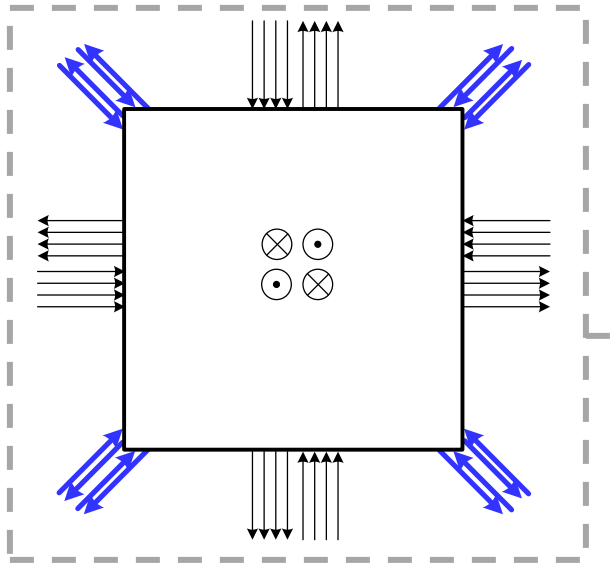
Vertical Stack

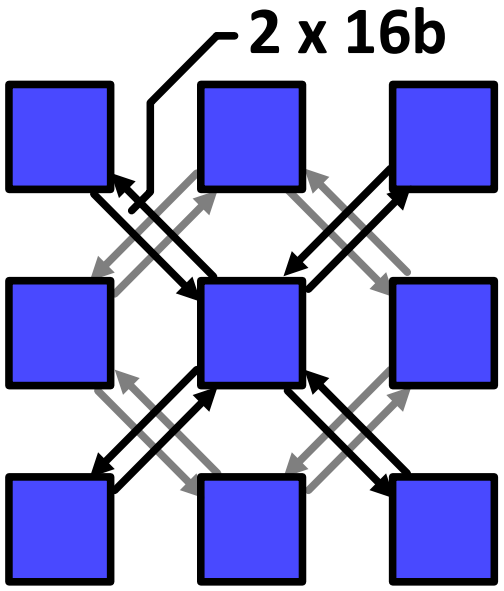


Distance = 1

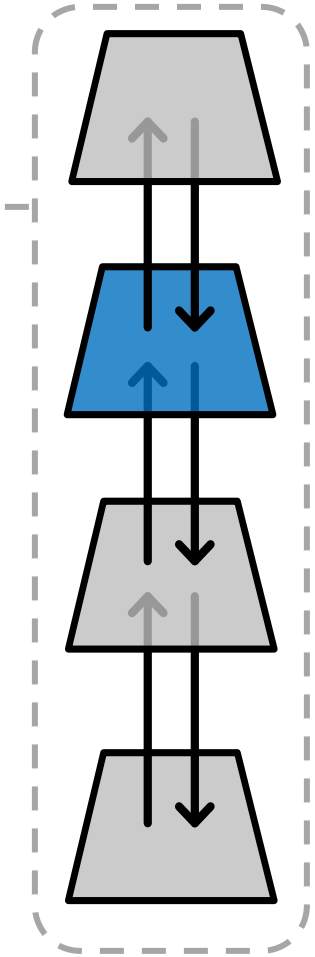


(2) UDSP Interconnect Architecture – Layer 2

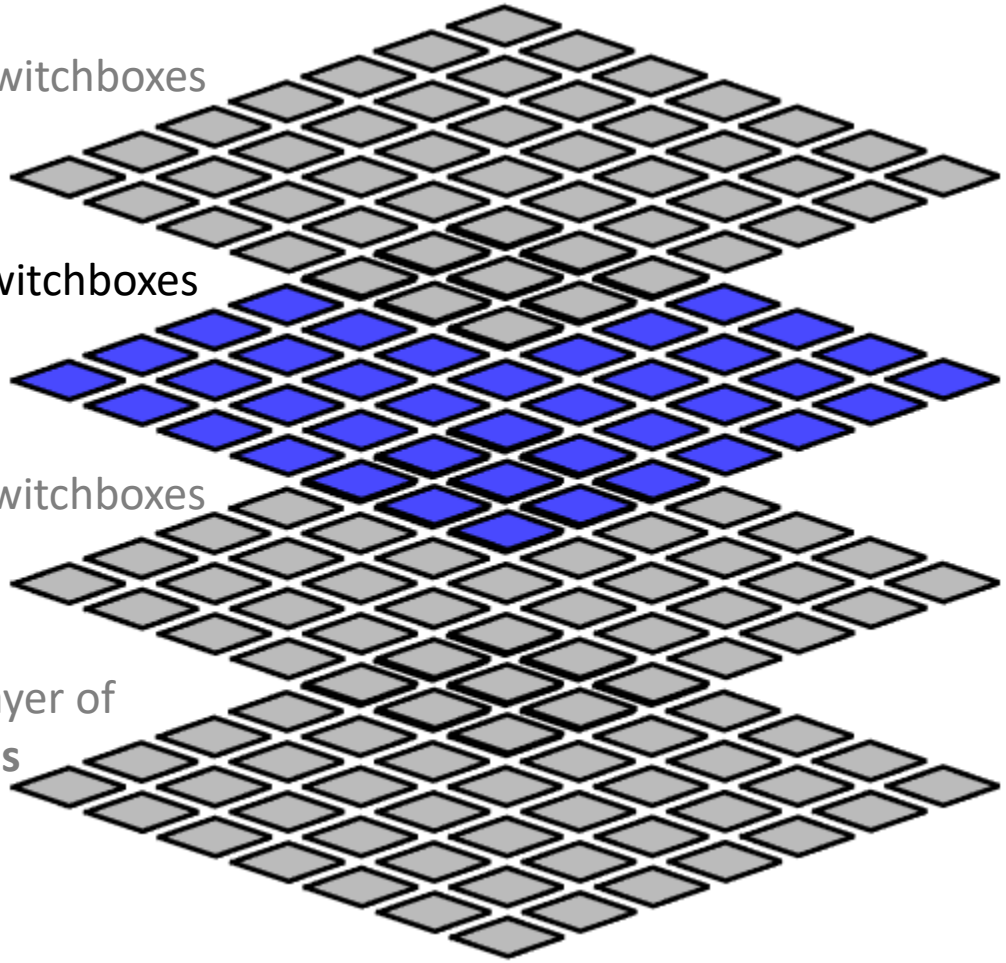




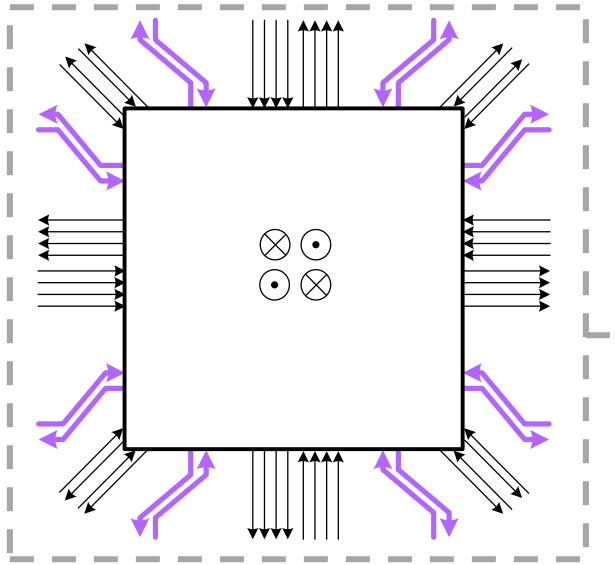
Vertical Stack

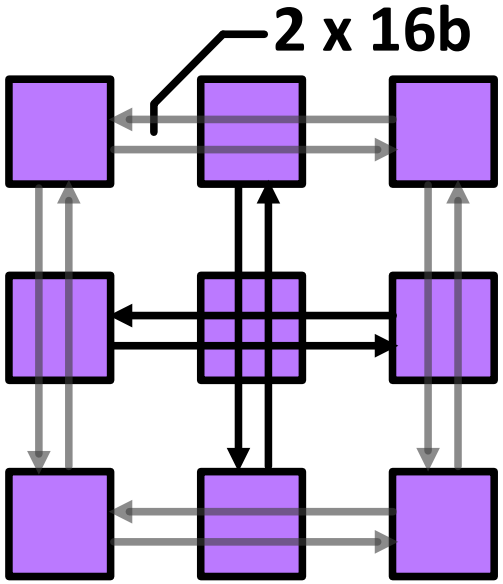


Distance = $\sqrt{2}$

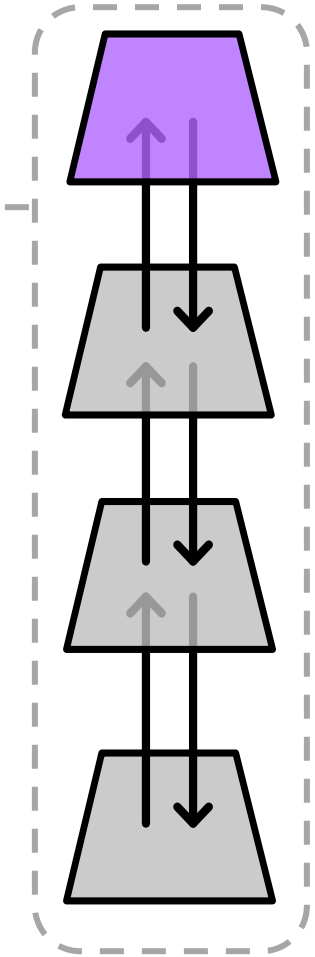


(2) UDSP Interconnect Architecture – Layer 3

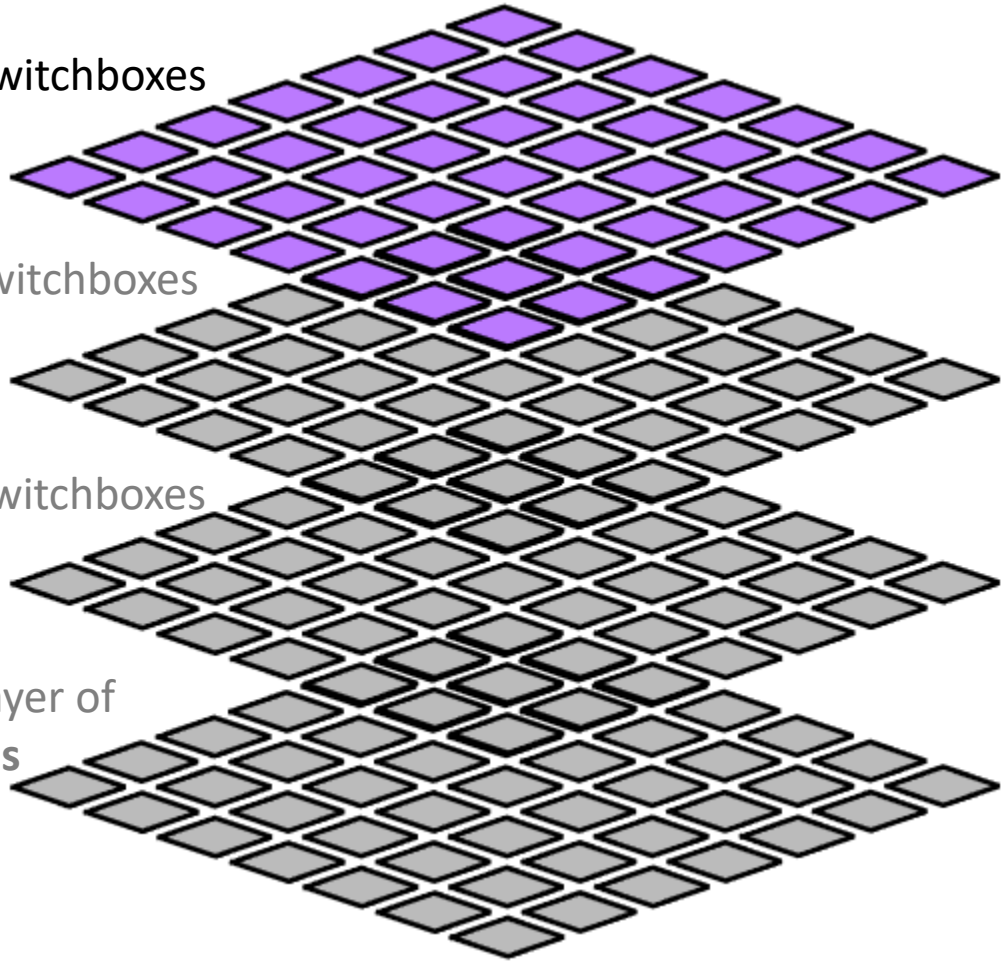




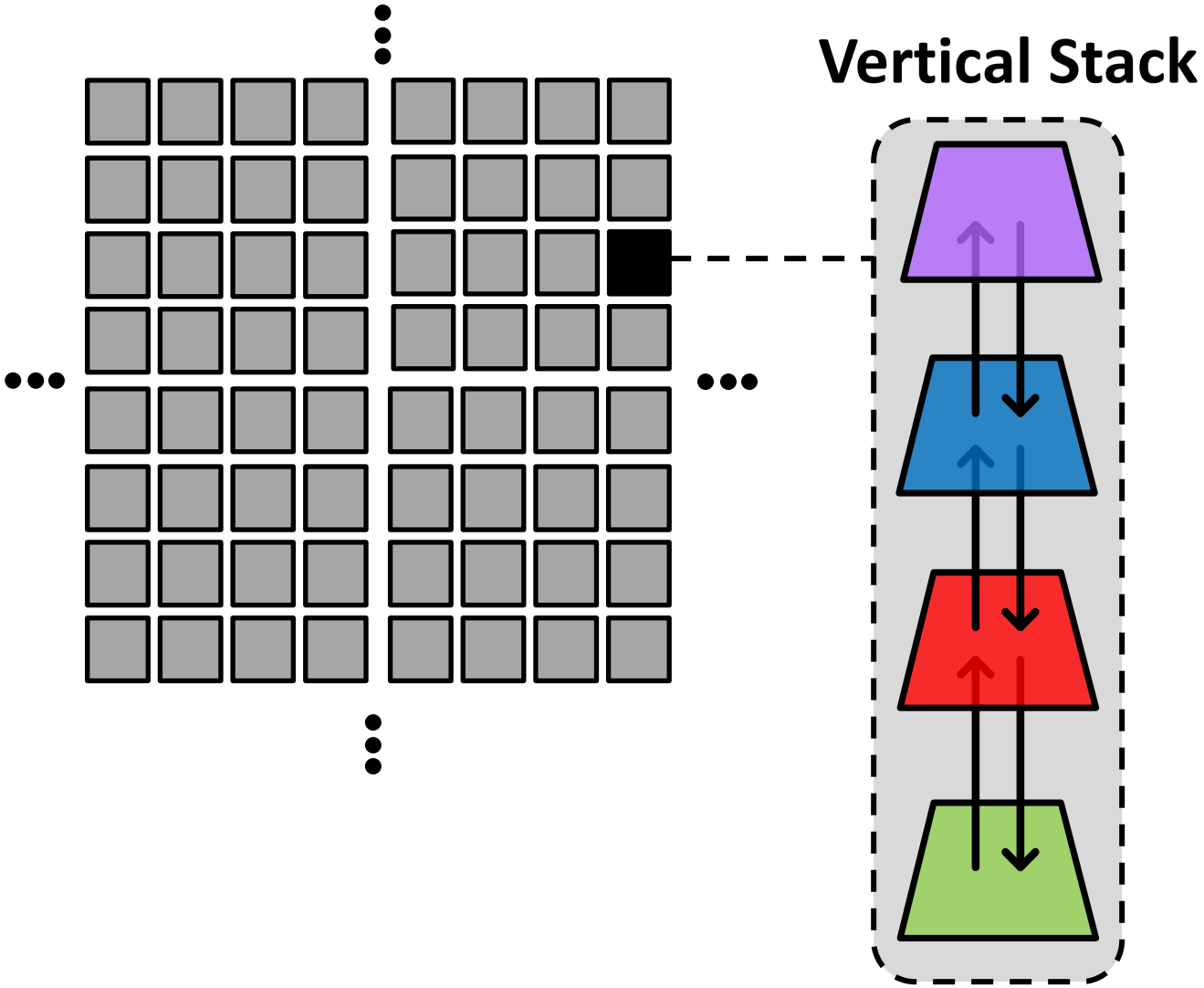
Vertical Stack

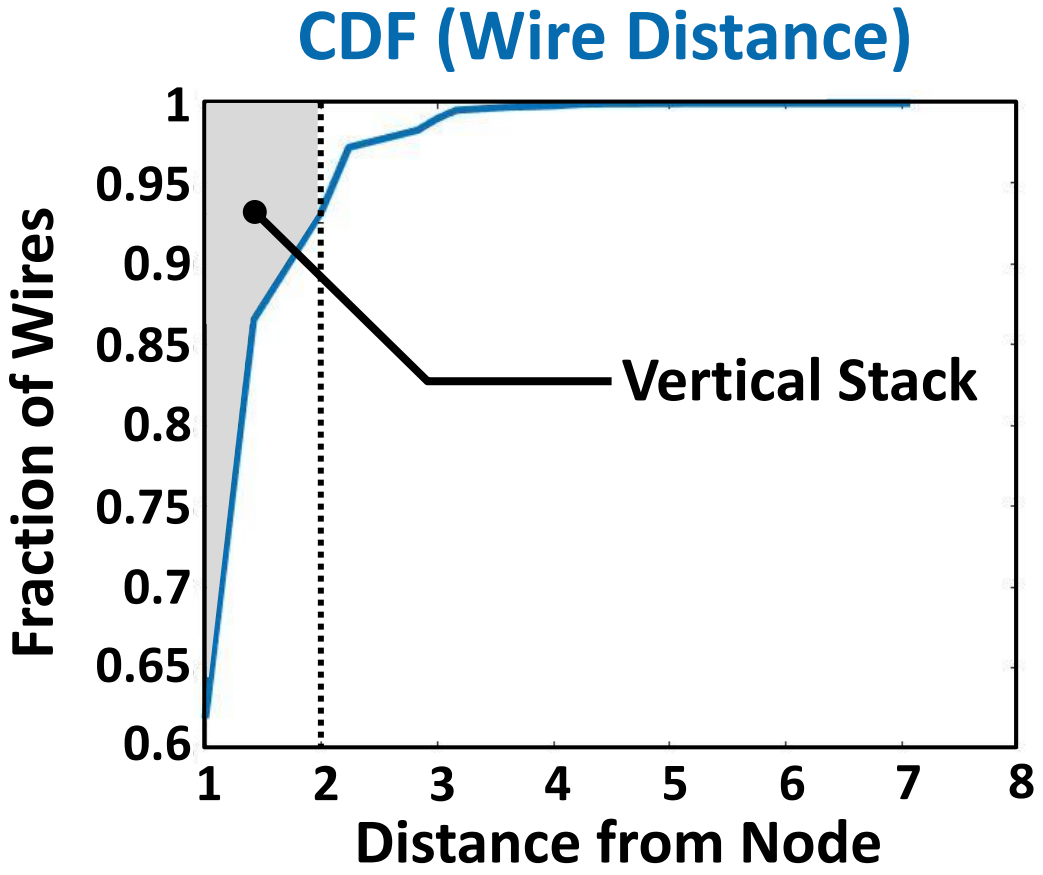


Distance = 2

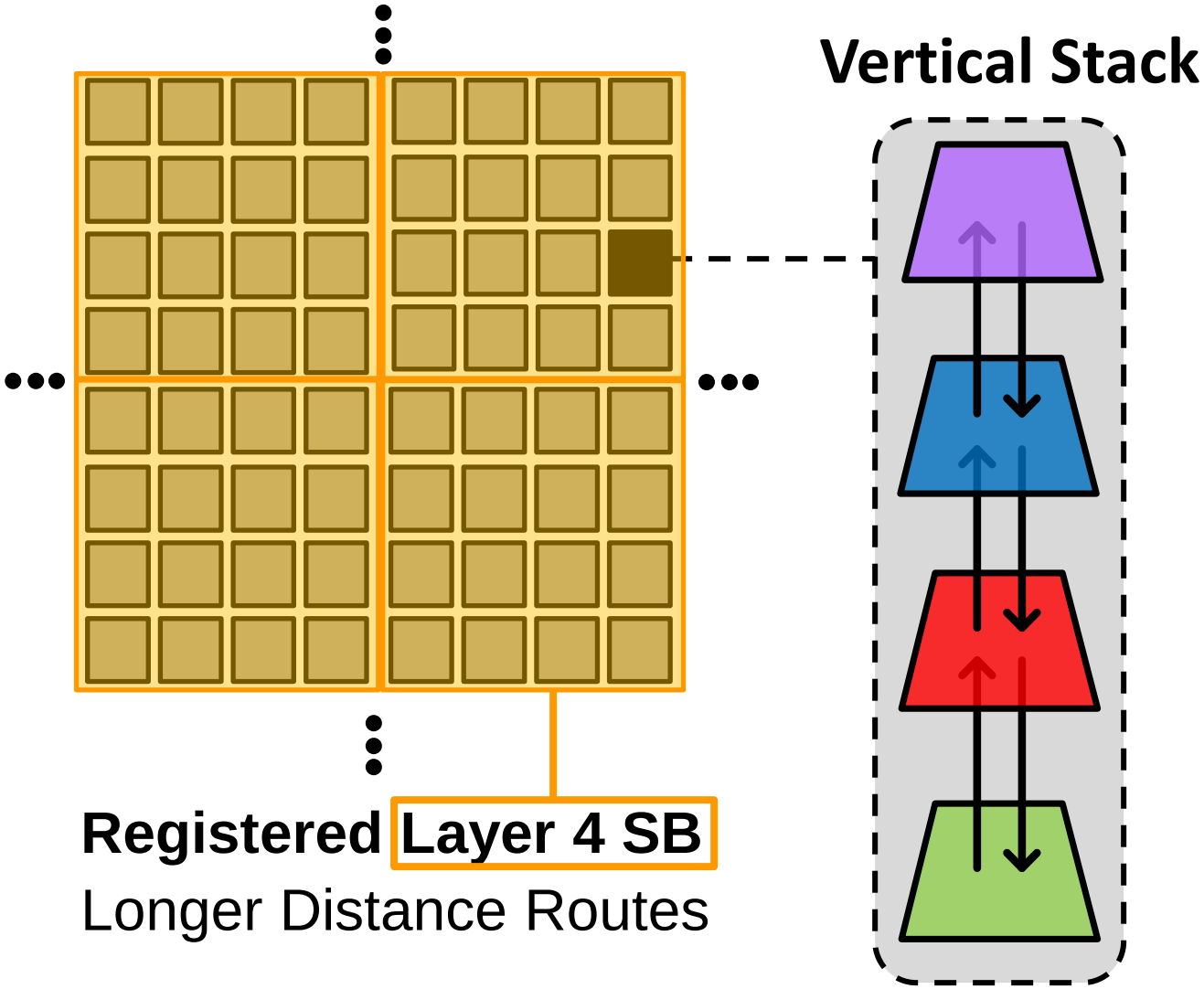


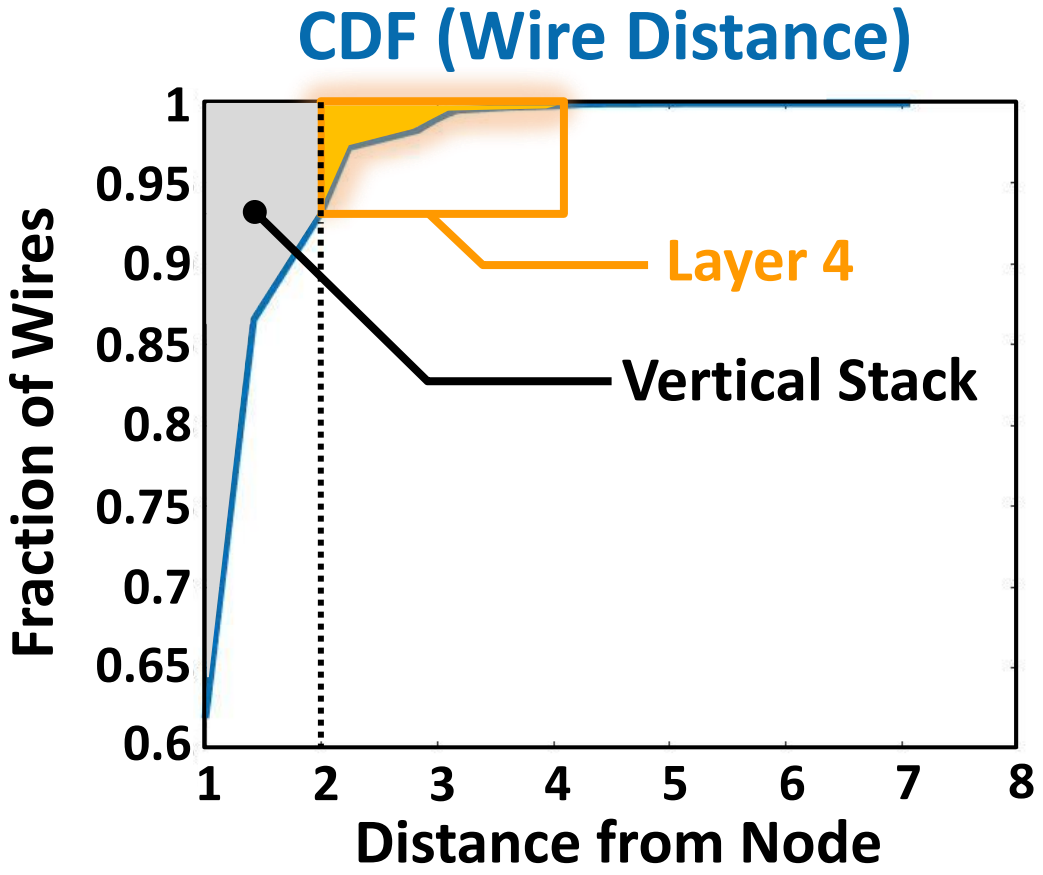
(2) UDSP Interconnect Architecture – Layer 4





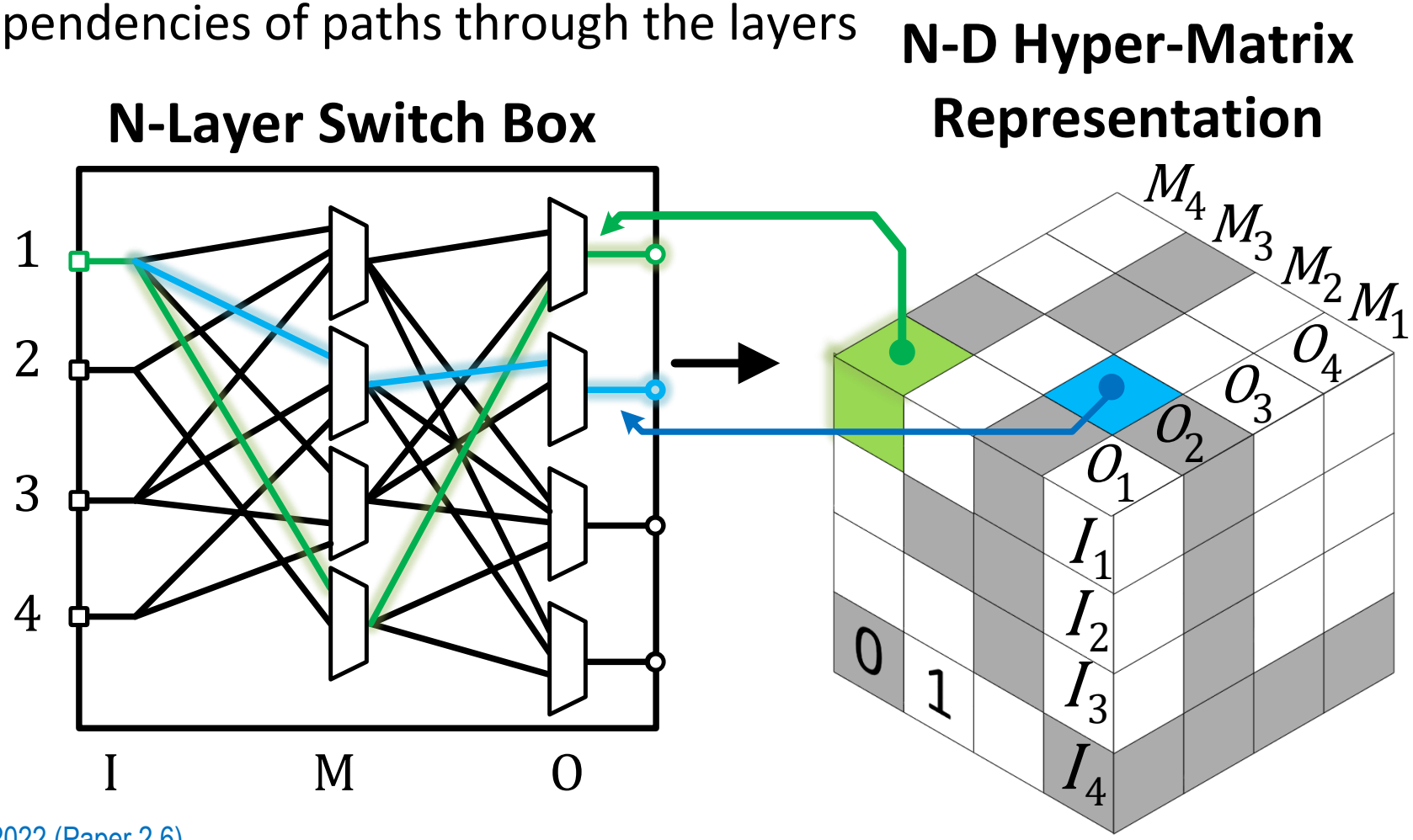
(2) UDSP Interconnect Architecture – Layer 4





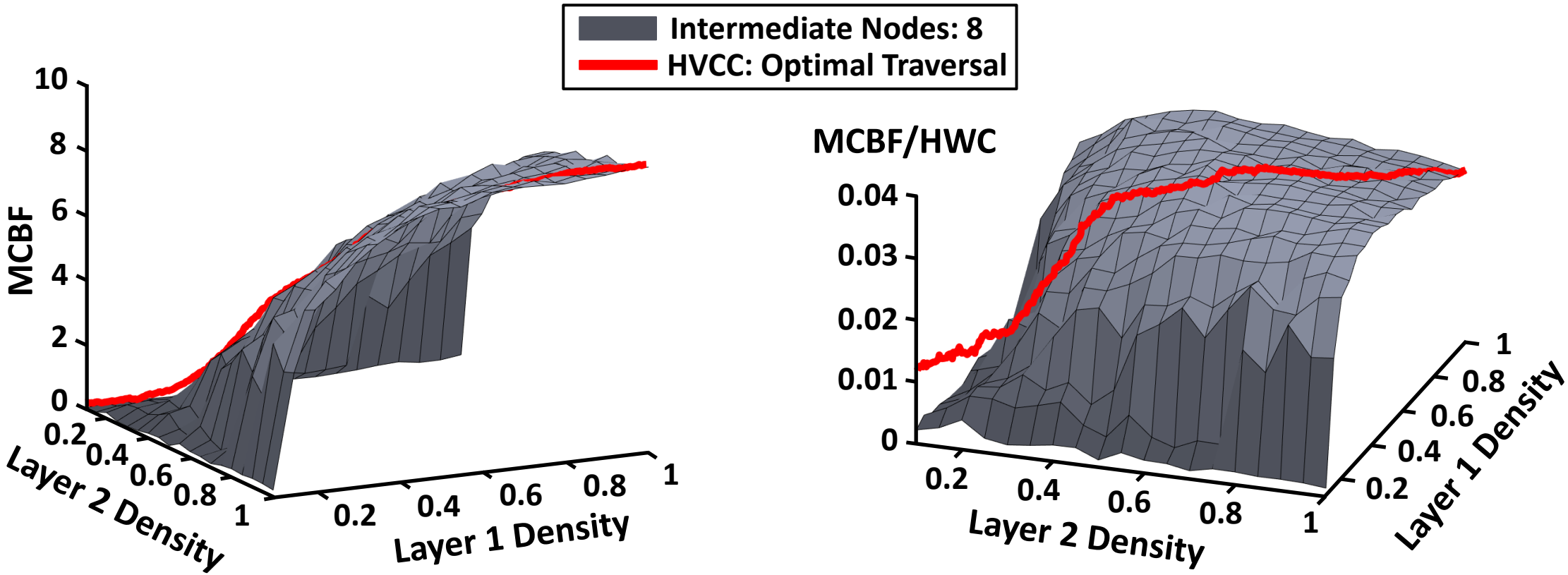
(3) N-Layer SB: Hyper-Matrix Representation

- Calculate hyper-vector cross-correlation (HVCC) in each dimension (layer)
  - HVCC (a dot product of the combination of vector pairs) for a layer measures global inter-dependencies of paths through the layers



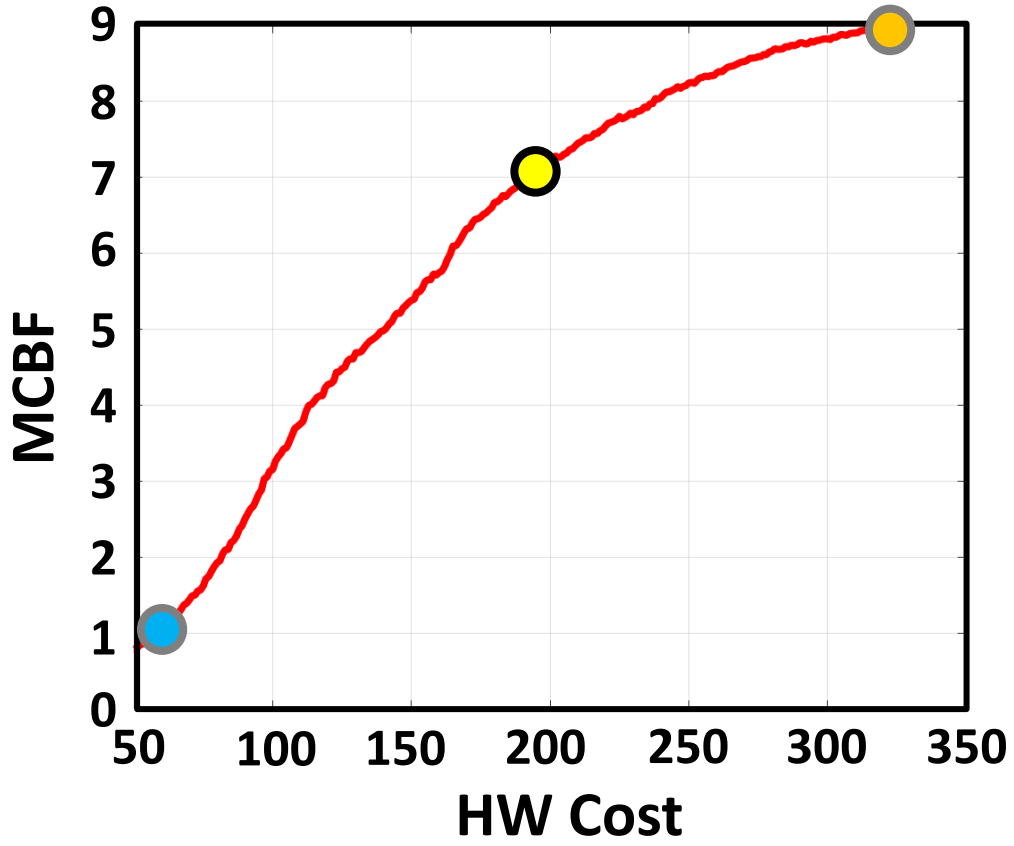
(3) Design Space Exploration: HVCC vs. Baseline Brute Force

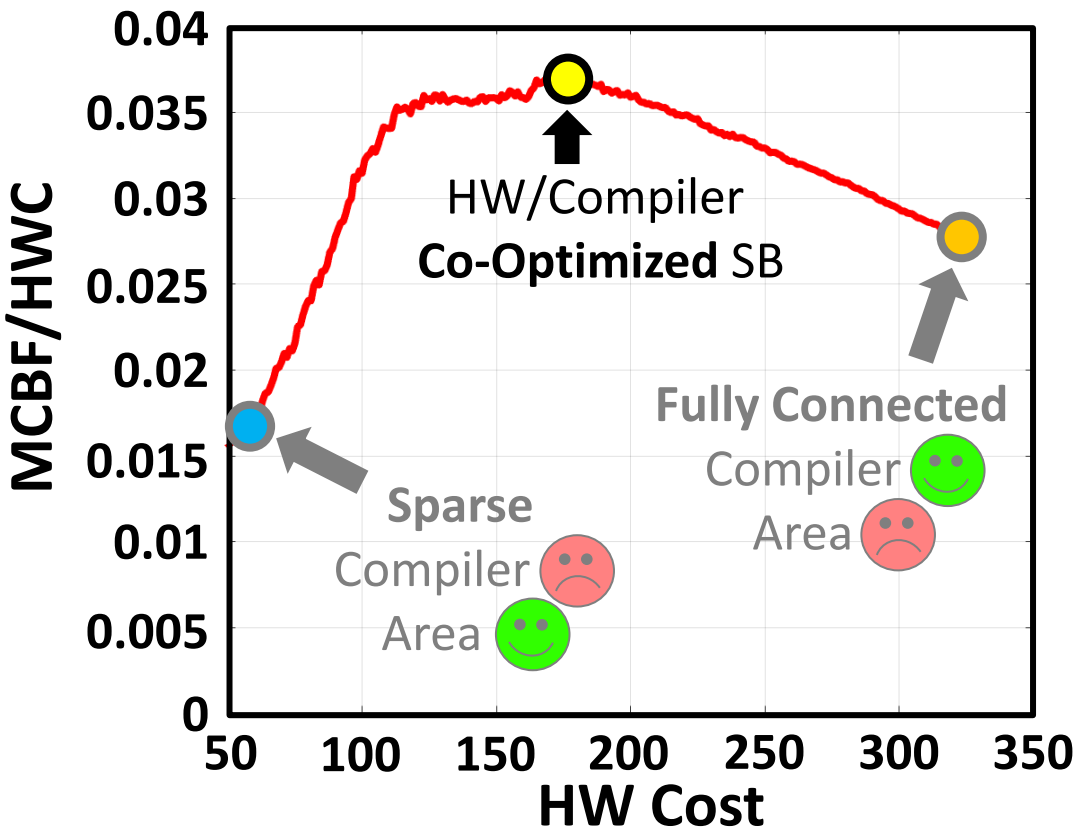
- **HVCC optimized Switchbox**
  - Plotted against layer density for 3-layer SB (HWC = Hardware Cost, No of Muxes)
  - Switchbox configuration example: 22-8-22



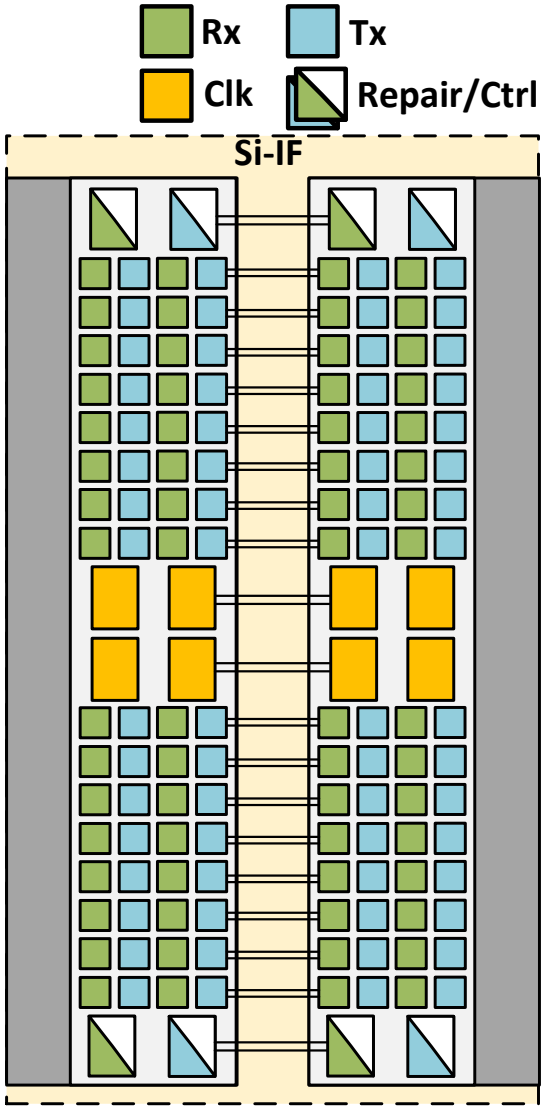
(3) DSE: Maximizing Silicon Efficiency

- Switchboxes are designed at the peak of the MCBF/HWC for Sw/Hw balance
  - Switchbox configuration example: 22-8-22





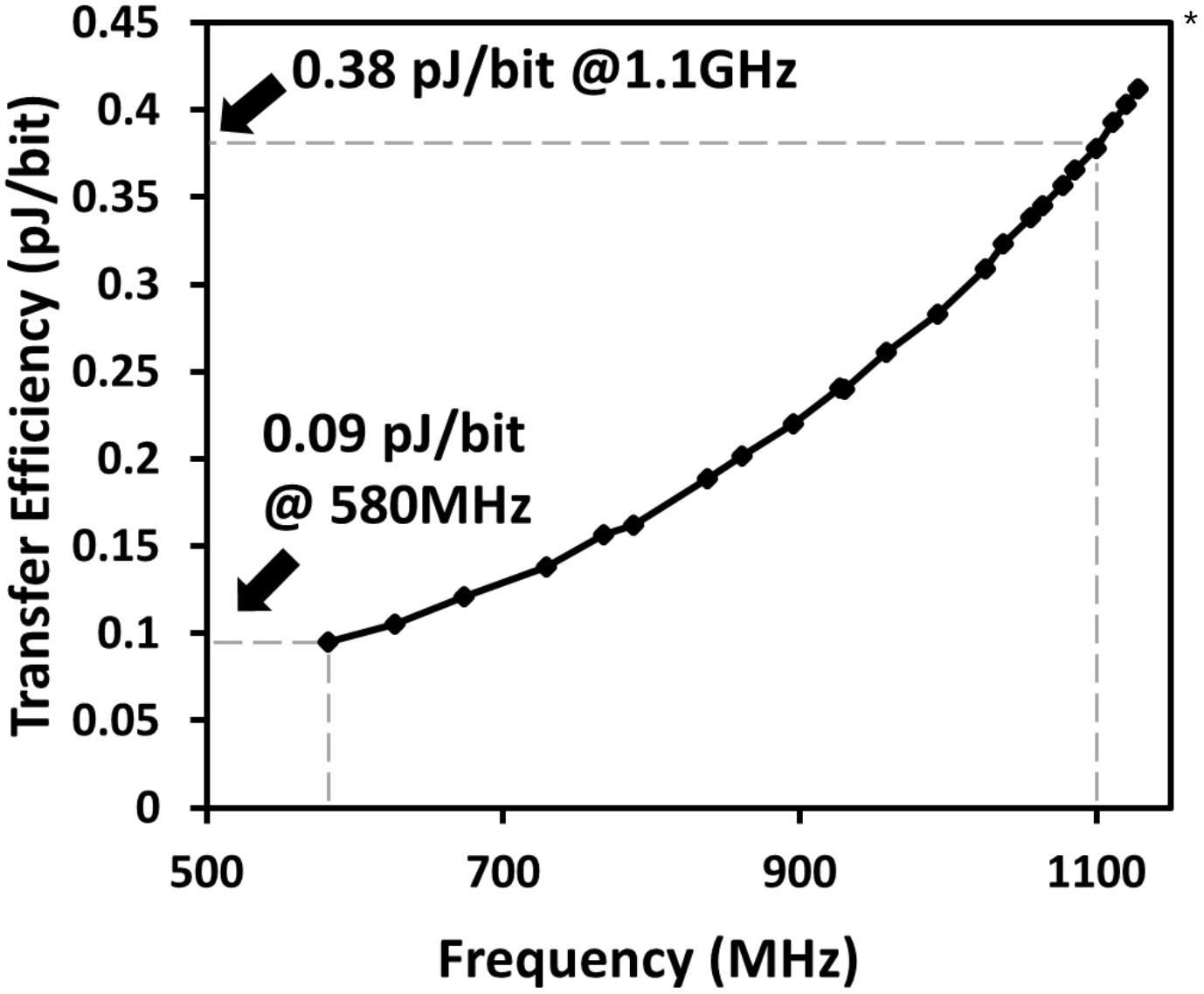
(4) UCLA SNR-10 Channel and Comparison to SoA



Spec	[This work]	CICC'21	JSSC'20	ISSCC'18
Techology	16nm	16nm	7nm	16nm
Package Substrate	**2-Layer Si-IF**	4-Layer EMIB	15-Layer CoWoS	MCM/ PCB
Bump Pitch (µm)	**10**	55	40	150
Reach (µm)	350	3,000	500	80,000
Data Rate (Gbps/pin)	1.1	2	8	25
Voltage (V)	0.8	0.9	0.3	0.95
Energy Effc. (pJ/bit)	**0.38**	0.83	0.56	1.17
I/O Area Density (μm^2/bit)	**137**	203	500	10,175
Peak Shoreline BW Density (Gbps/mm)	297	256	1600	292
Layer BW Density (Gbps/mm/layer)	**149**	64	107	25

U. Rathore, et al., ISSCC 2022 (Paper 2.6)

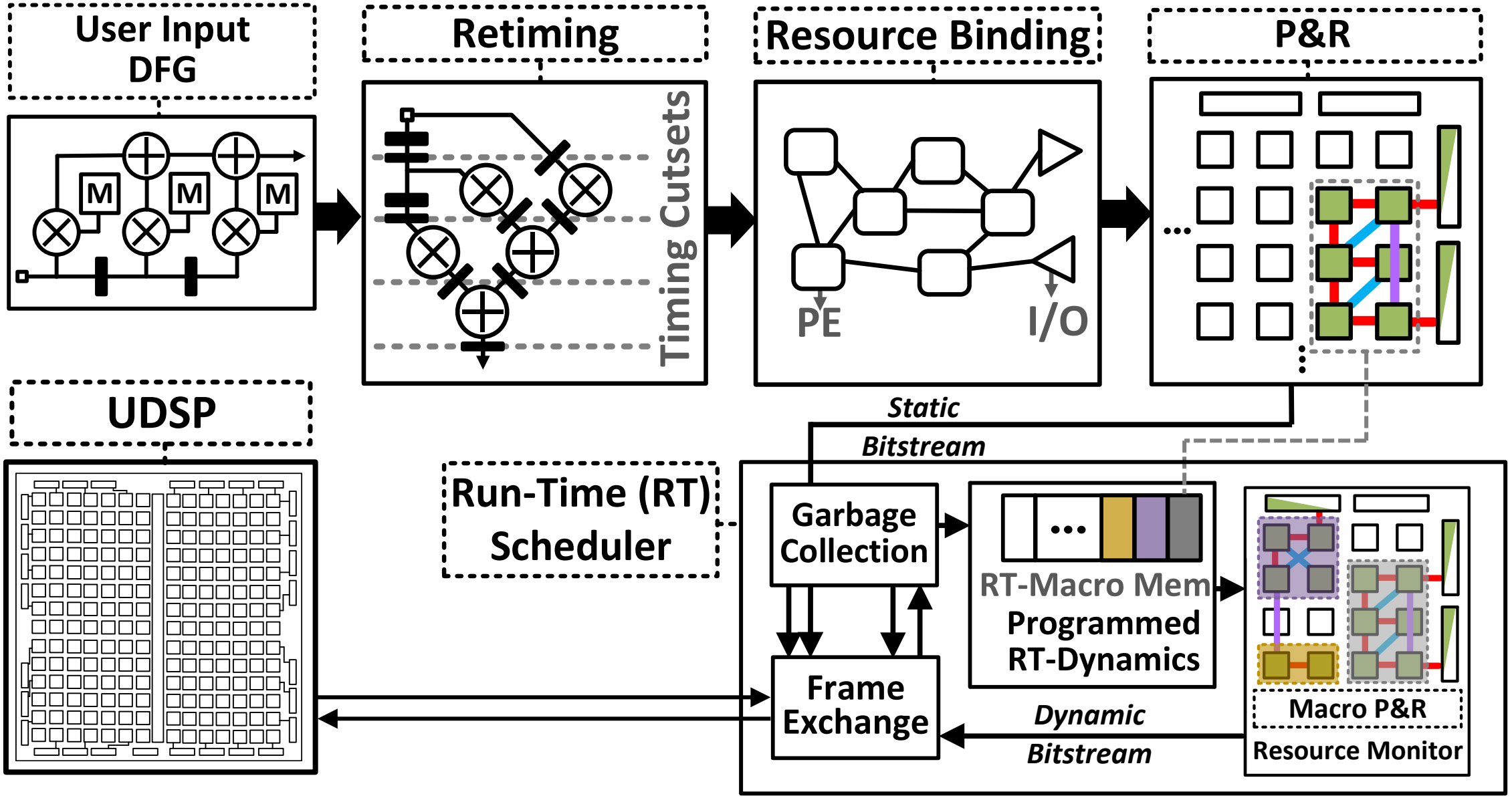
(4) SNR-10 Transfer Efficiency vs. Voltage



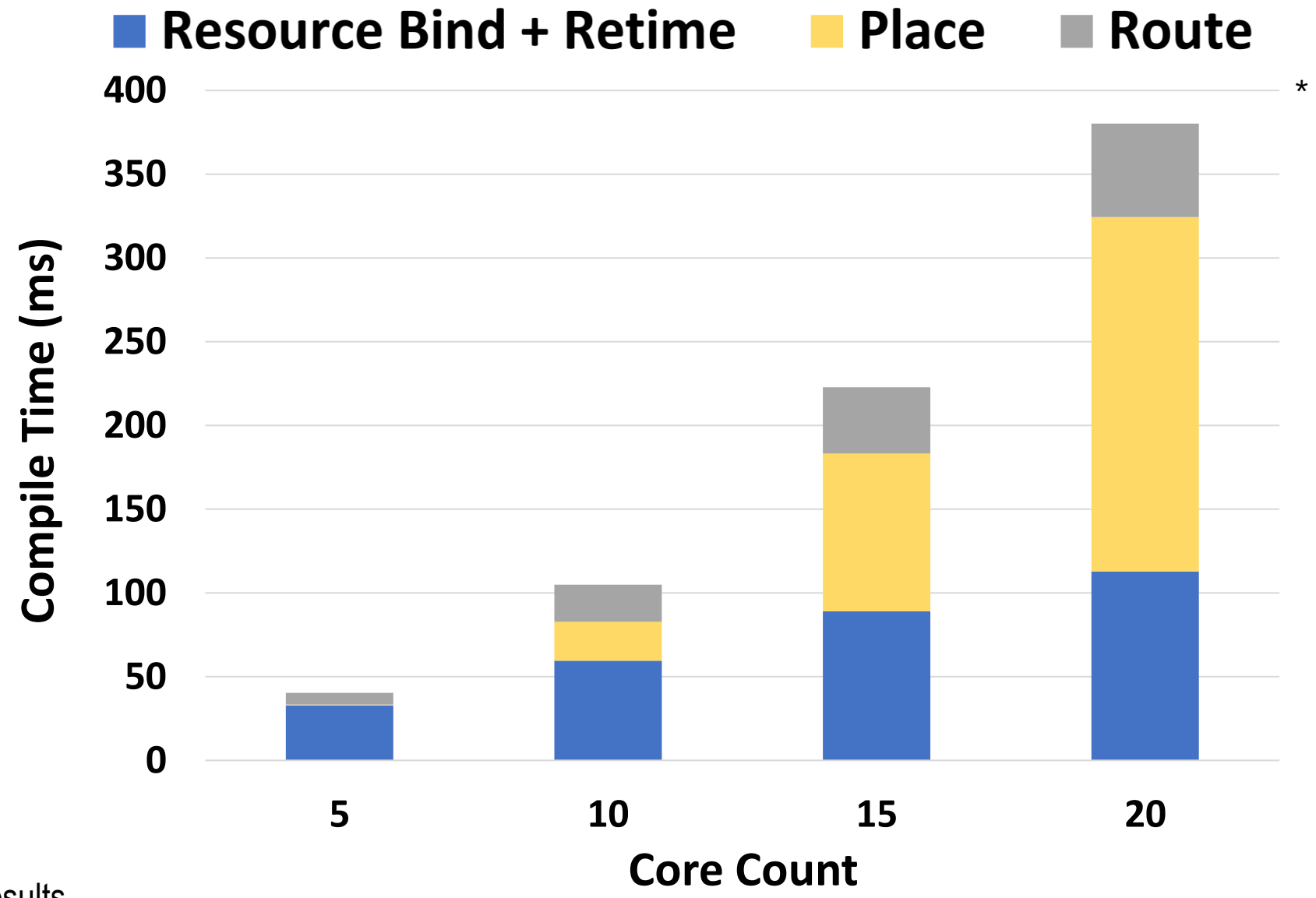
U. Rathore, et al., ISSCC 2022 (Paper 2.6)

*100% toggle rate

(5) UDSP Programming Flow



(5) UDSP Compile Time Breakdown

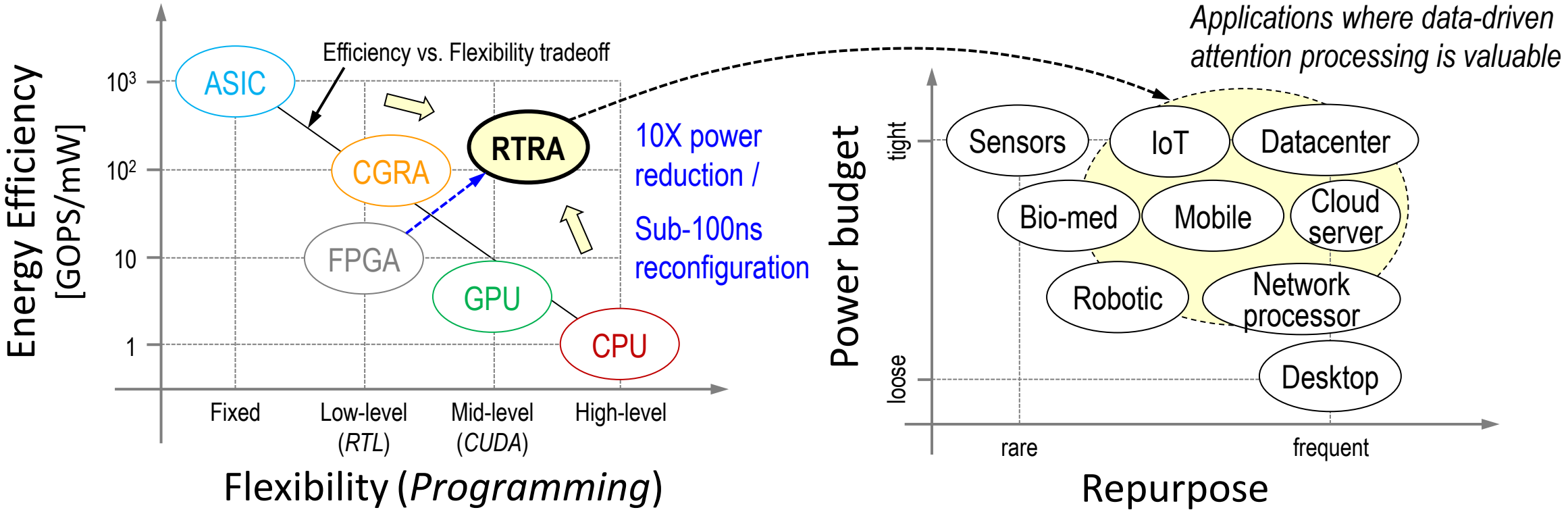


*Average over 50 results

Courtesy: Tim Ling (UCLA)

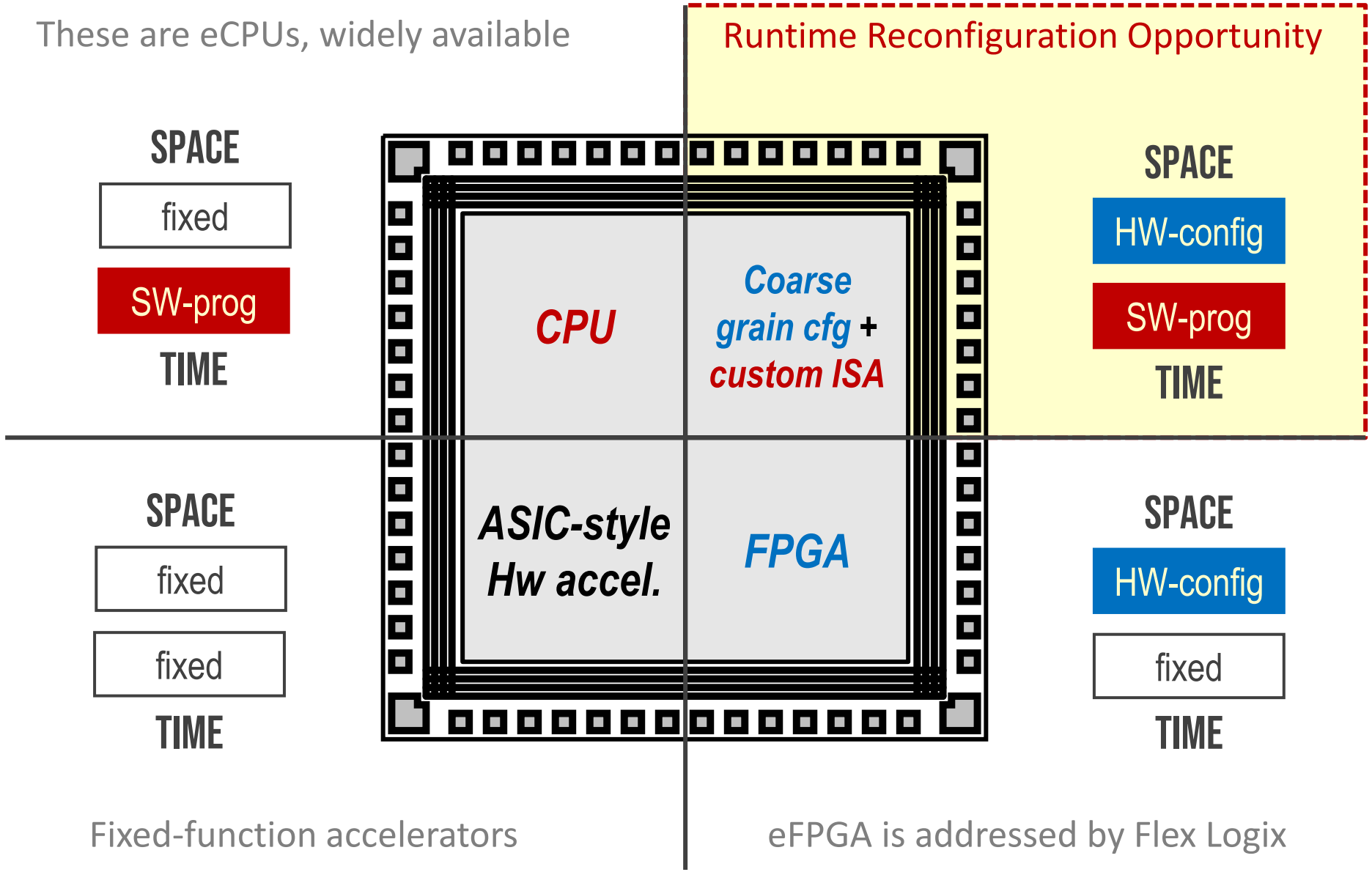
Runtime Reconfiguration for Data-Driven Processing

RTRA breaks standard efficiency vs. flexibility tradeoff



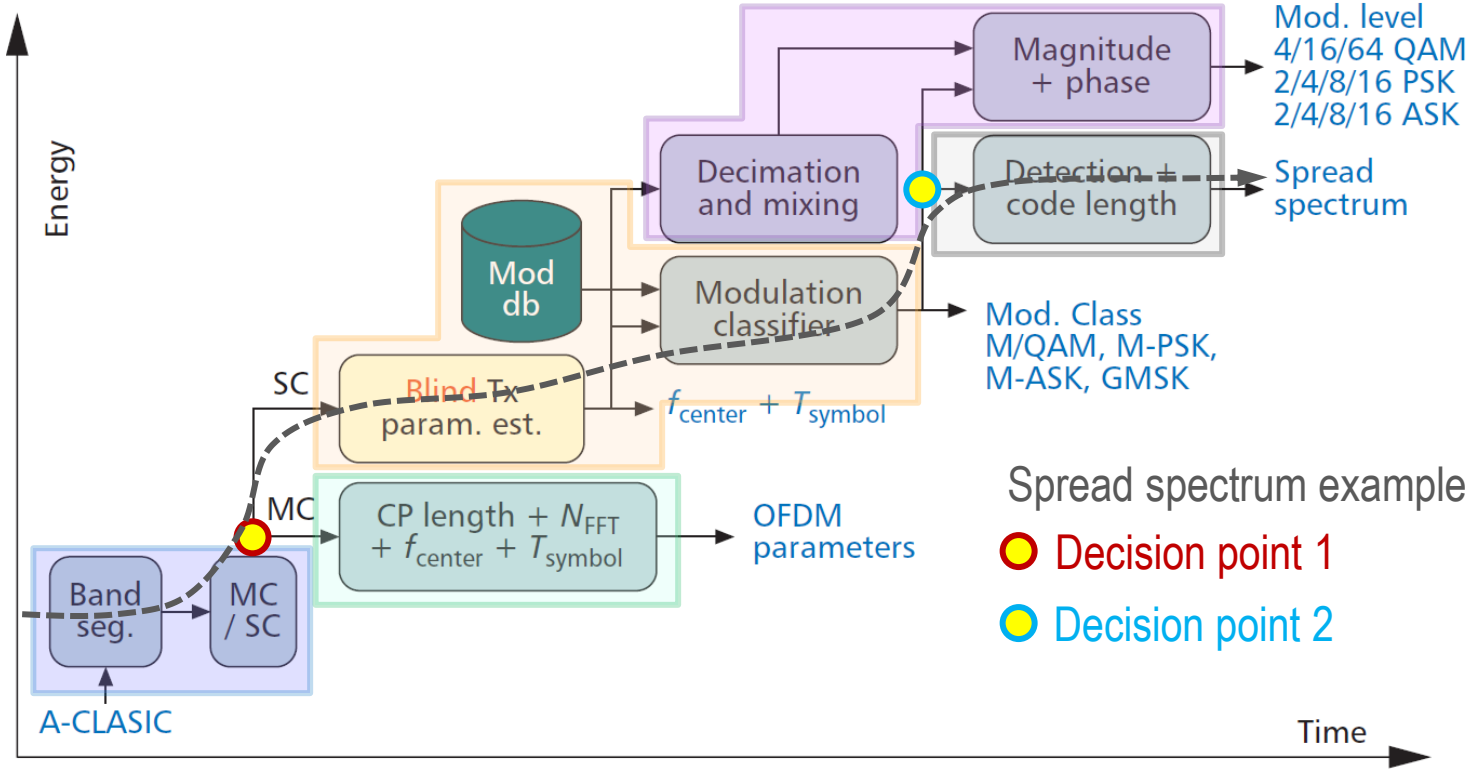
- Initial use case: software radio (dynamic spectrum access, networks, etc.)
  - Signal classification relies on data-driven outcomes-based processing pipelines
  - Future extension to network-scale adaptation involving learning-based methods

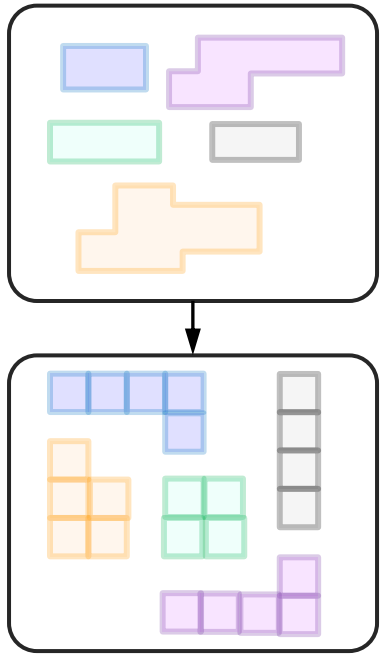
RTRA is the Missing Element of a Canonical SoC



Blind Spectrum Sensing Application

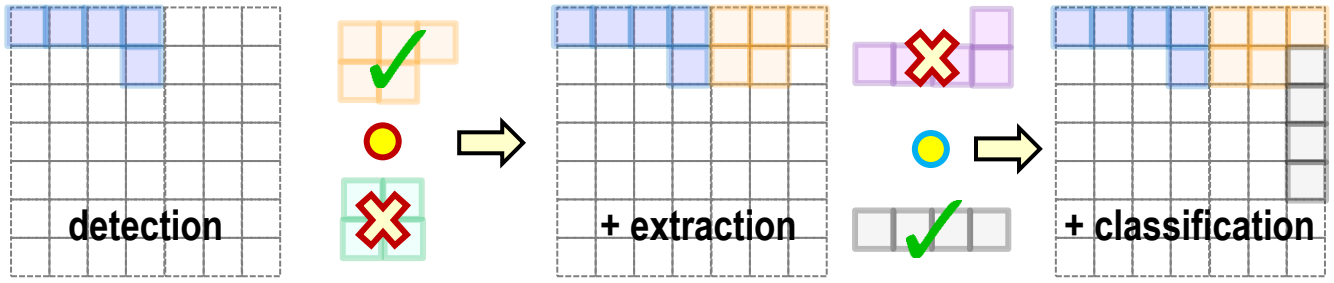
Data-driven outcomes-based detection, extraction, and classification





Colored shapes have corresponding dependency graphs

Compiled soft-maps: scheduler dynamically allocates them to vacant polygons



Spread spectrum example showing execution path for a spread spectrum signal; middle plot assumes detection is always-on looking for new signals; right plot could re-purpose orange block if OFDM signal entered middle pipeline after SS completed

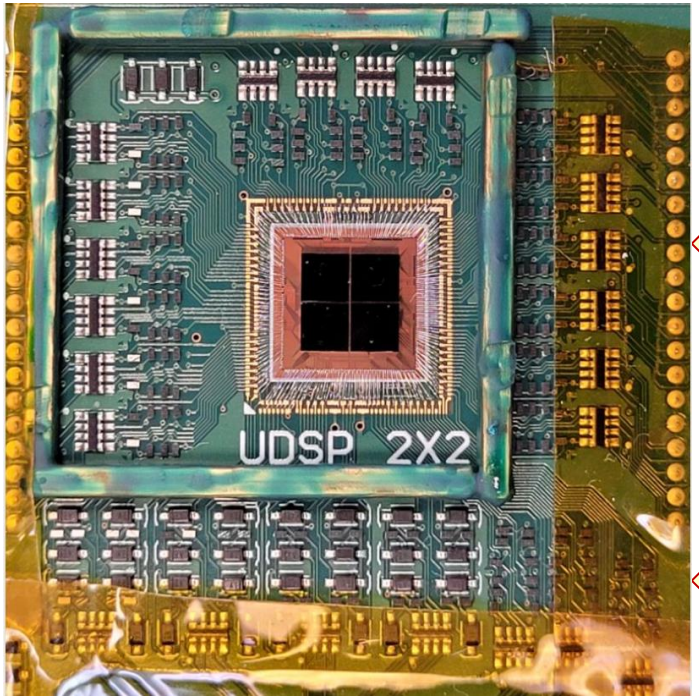
Algorithms from: R. Harjani, D. Cabric, D. Markovic, et al., "Wideband Signal Classification on a Battery Budget," IEEE Comm. Magazine, Oct. 2015.

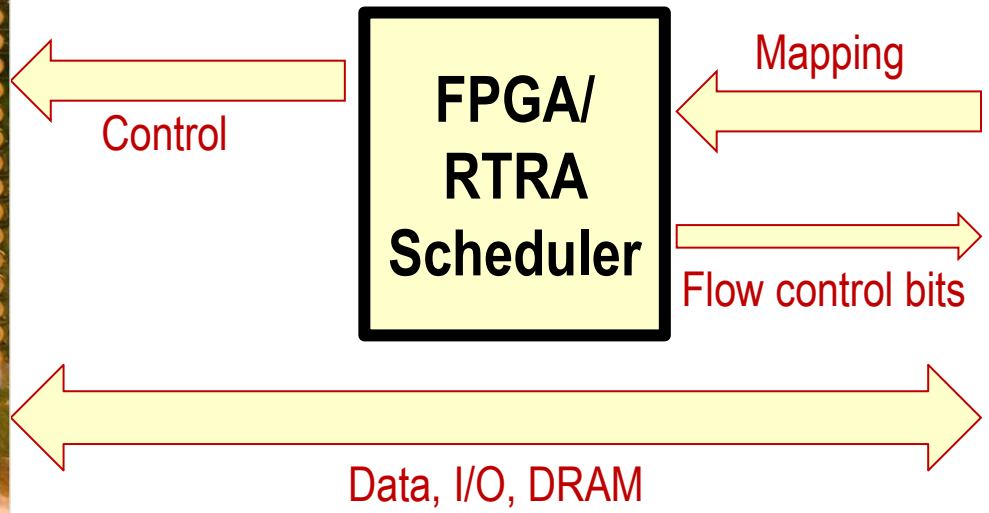
Hardware-in-the-Loop Characterization

Baseline work-set
** FFT, FIR, IIR, C-Mtx algebra*
Application work-set
** detection, estimation, classification*

Goals of the study:

- Characterize utilization (compute and network), time to repurpose, compute time (programming + execution) under multi-program, multi-size, priority mappings
- Compare with statically configured UDSP and FPGA





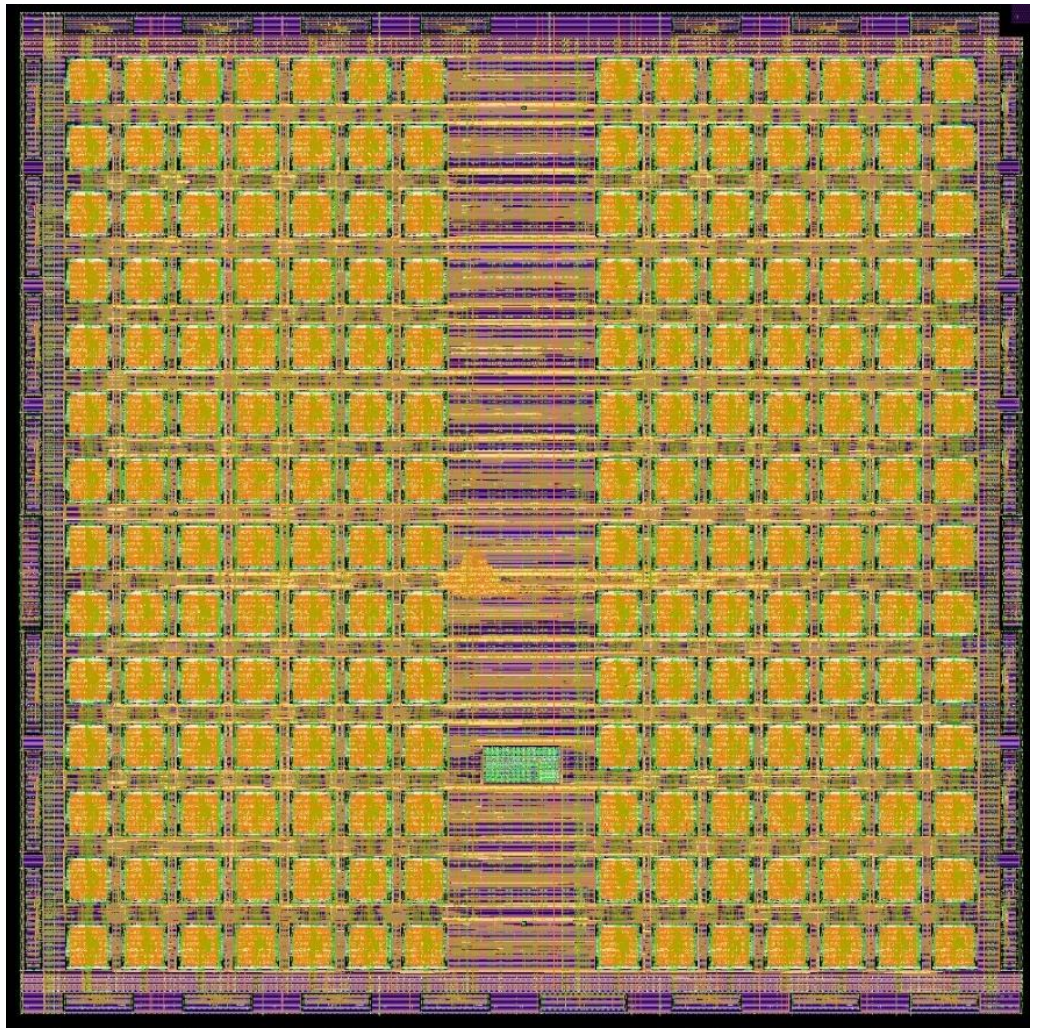
Comparison between
** RTRA (emulated)*
** UDSP*
** FPGA*

Post Synthesis Area and Power Estimates (TSMC 16nm)

Scheduler finds feasible polygon mapping **within 20 clock cycles**

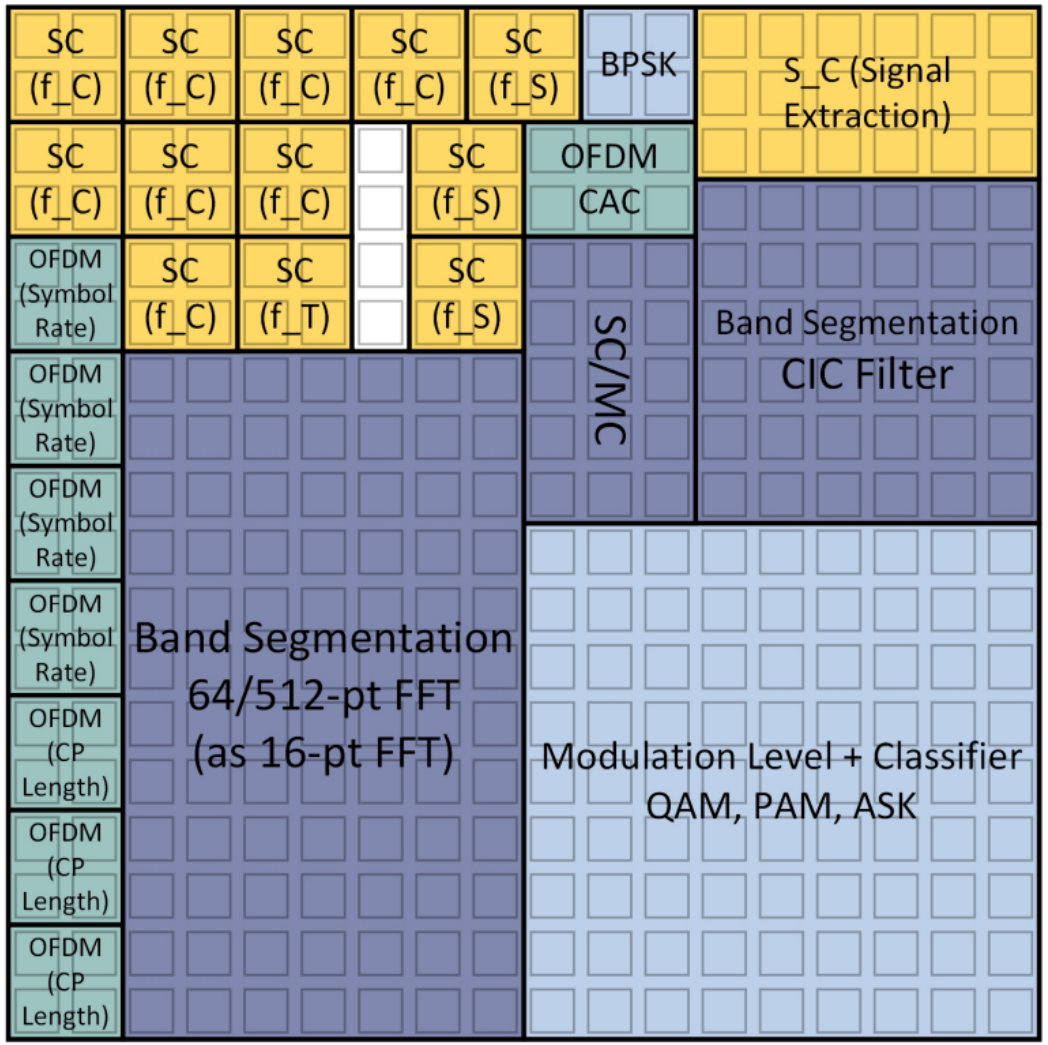
Approximate size of scheduler with 2x layout factor

	Scheduler (Post synthesis	UDSP (Silicon Area)	Fraction
Area	0.08 mm^2	5.29 mm^2	1.5%
Power	40 mW @1GHz	1400mW @1GHz	3%

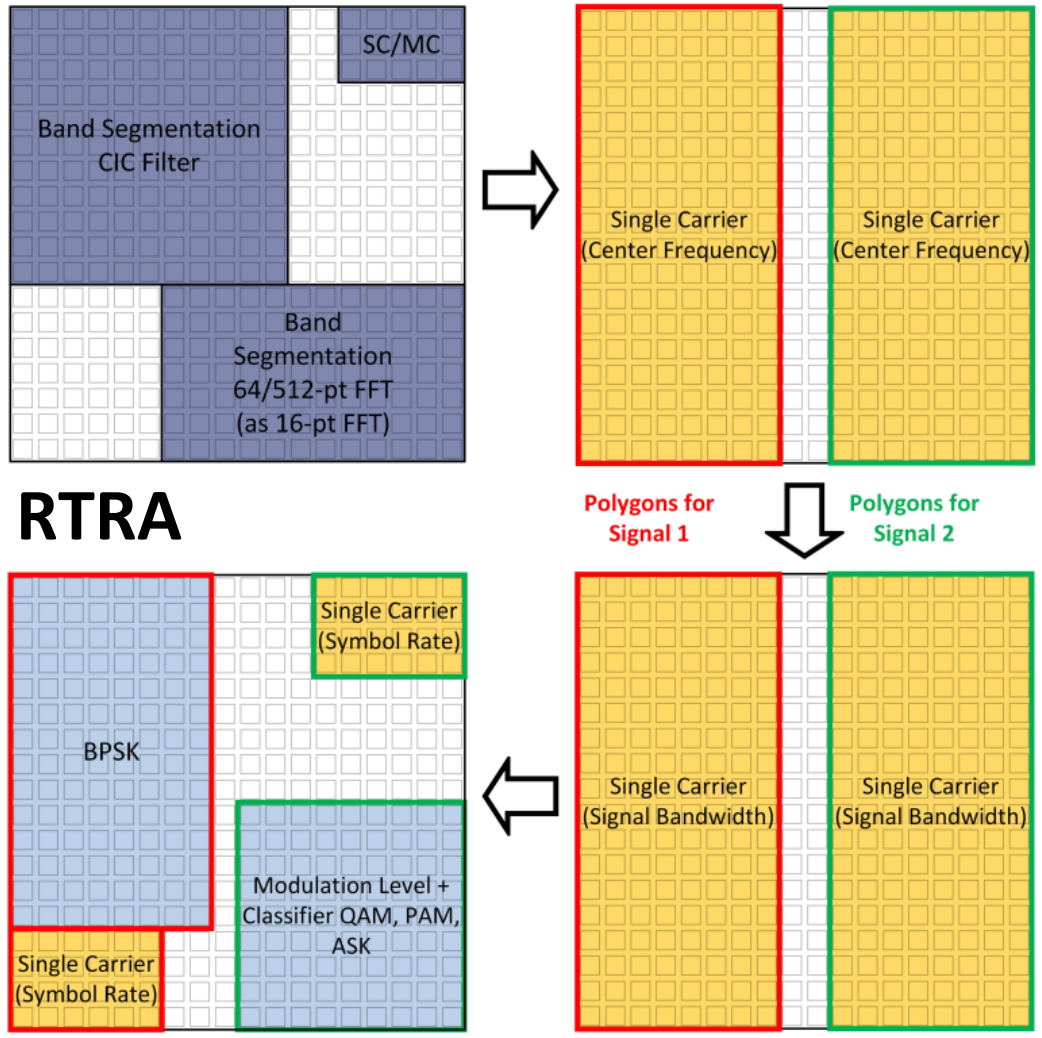


UDSP layout for reference

RTRA Dynamically Configures Hardware Pipeline



UDSP array size: 18x18

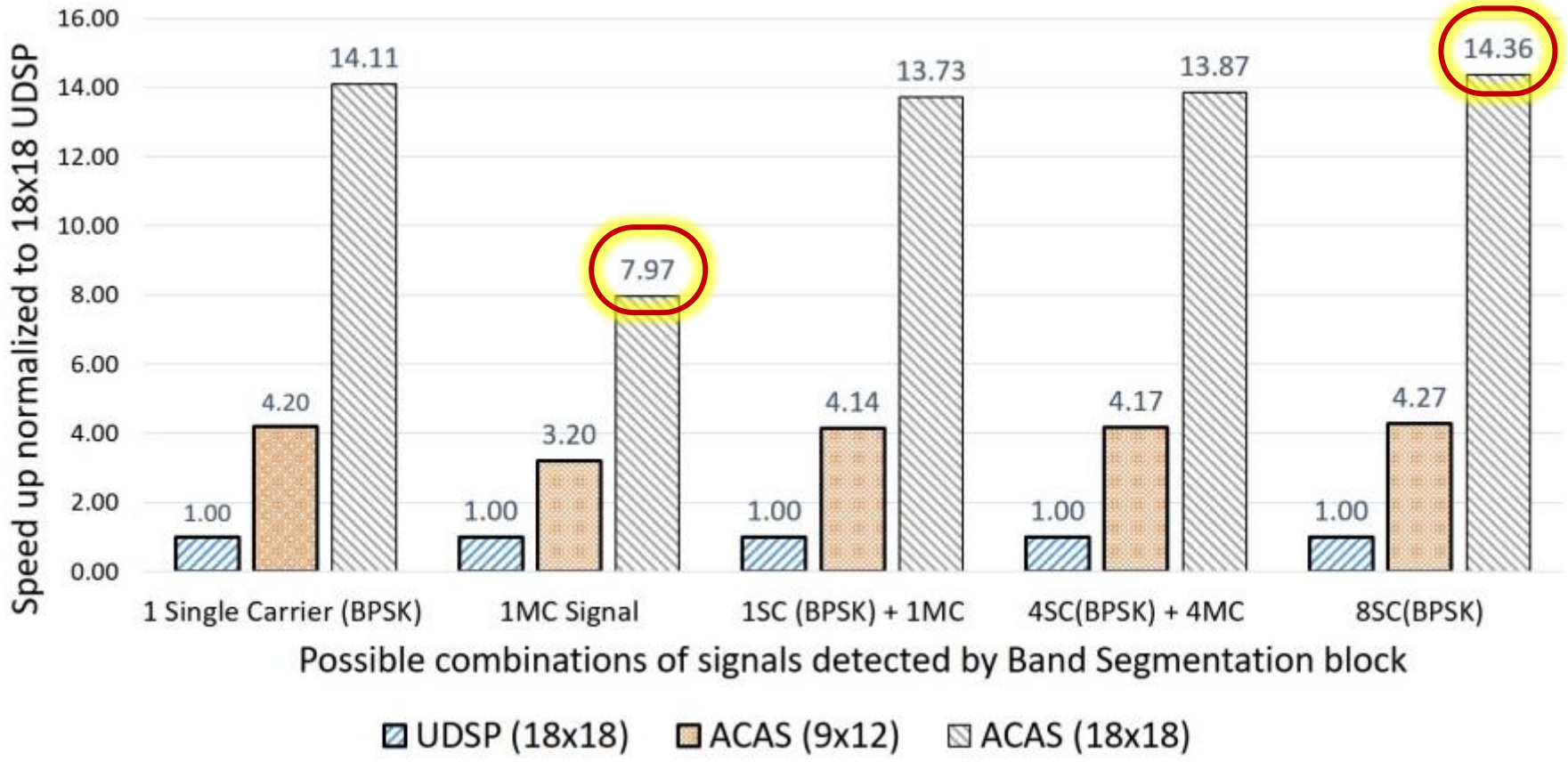


RTRA array size: 18x18

Courtesy: Sumeet Nagi (UCLA), manuscript in preparation (MICRO'22)

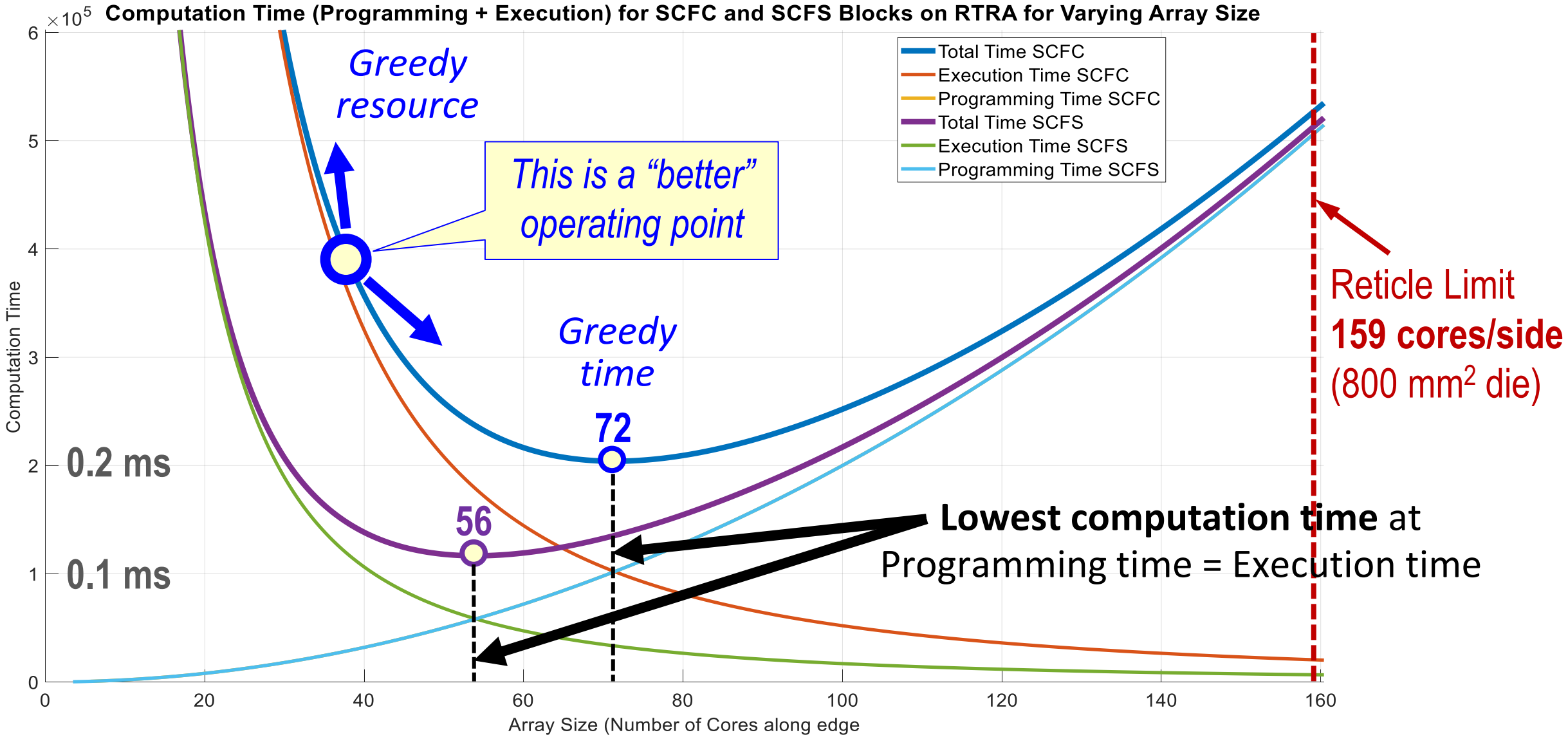
RTRA Maximizes Active Area Utilization

RTRA (ACAS) achieves **8-14x speed up** compared to statically-configured UDSP



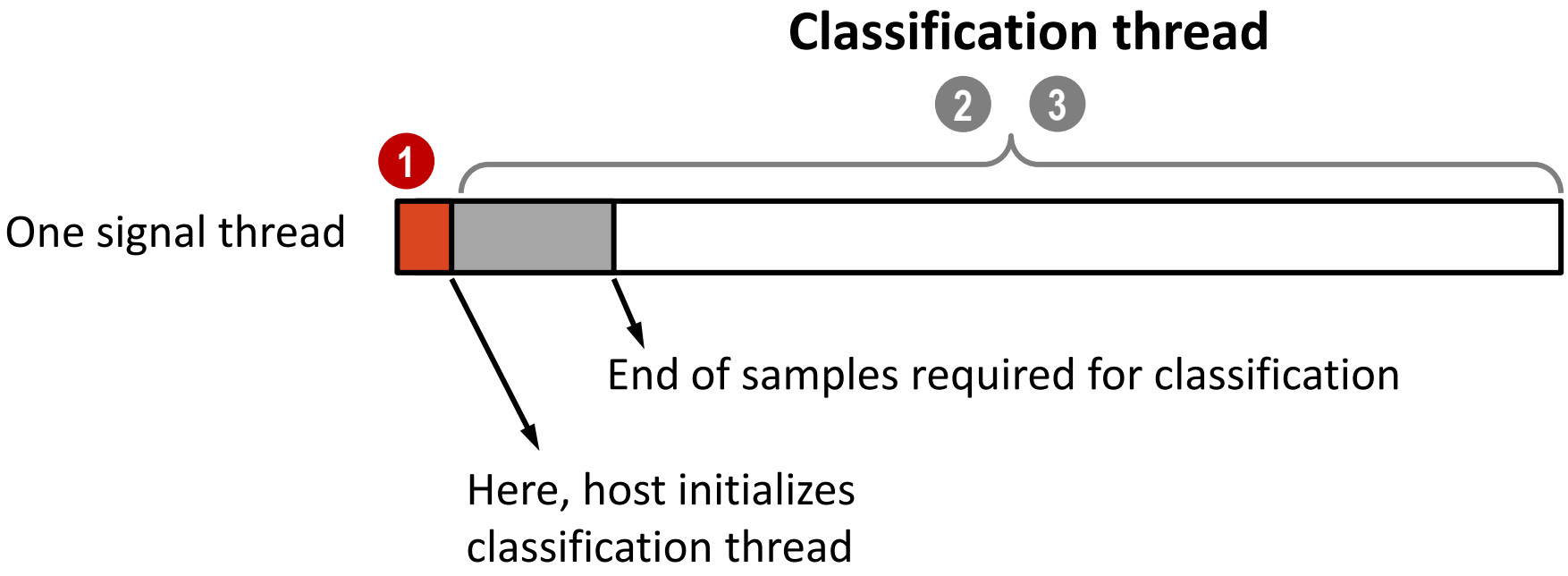
Courtesy: Sumeet Nagi (UCLA), manuscript in preparation (MICRO'22)

How to Balance Compute Time and Hardware Resources?

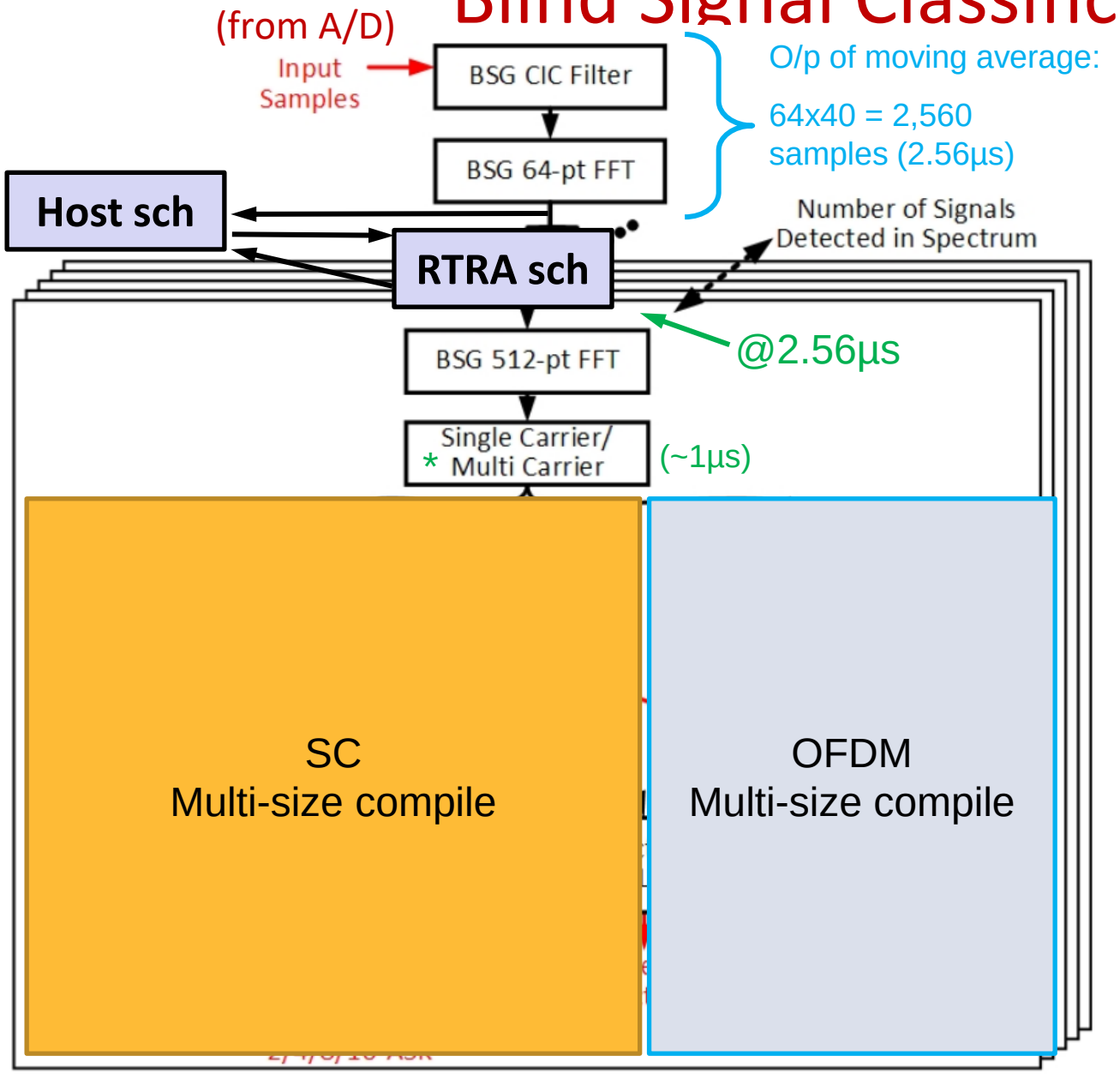


Signal Pipeline Model (Simplified)

1. Always-on coarse detection (part of band segmentation)
   - Once signal is detected, host will initialize **classification thread** and send program to RTRA
2. RTRA has internal scheduler and will map the program to the array
3. RTRA communicates branching dependencies with the host
   - RTRA executes all those data-driven programs and communicates completion to the host



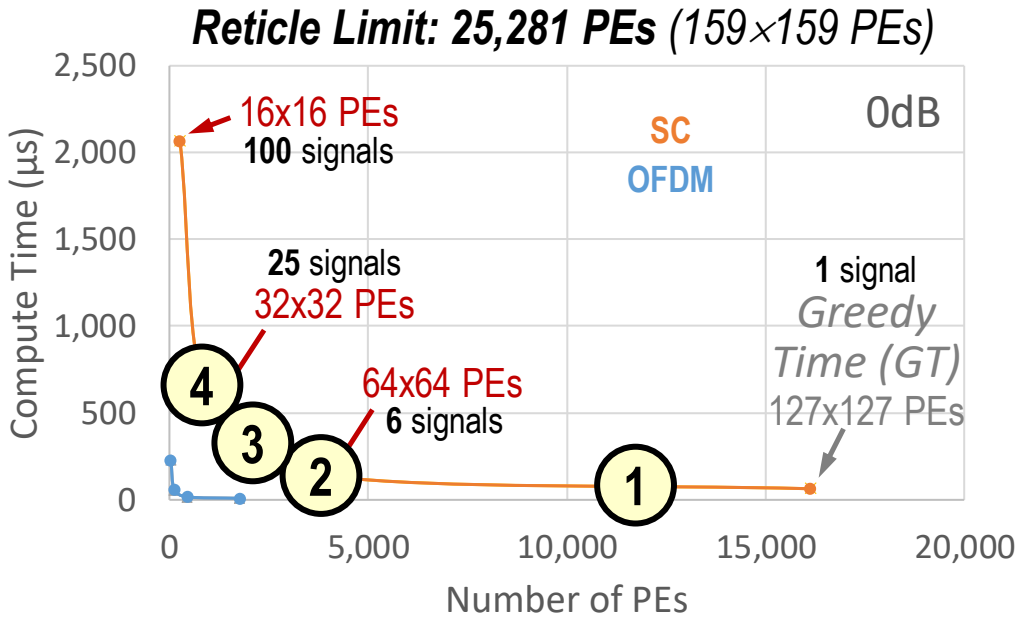
Blind Signal Classification Pipeline



- Always-on front-end (from A/D)
  - Takes **2.56µs** initially to produce first signal thread(s)
  - Scans for signal thread(s) every 64 samples thereafter (**64ns**)
- Samples required for classification
  - 128k samples (**128µs**)
  - @2.56µs classification starts
- Then goes into multi-size compile
  - Host decides on multi-size compile (scheduling is across host / RTRA)

Multi-Signal Service Time Balancing (Same Service/End Time)

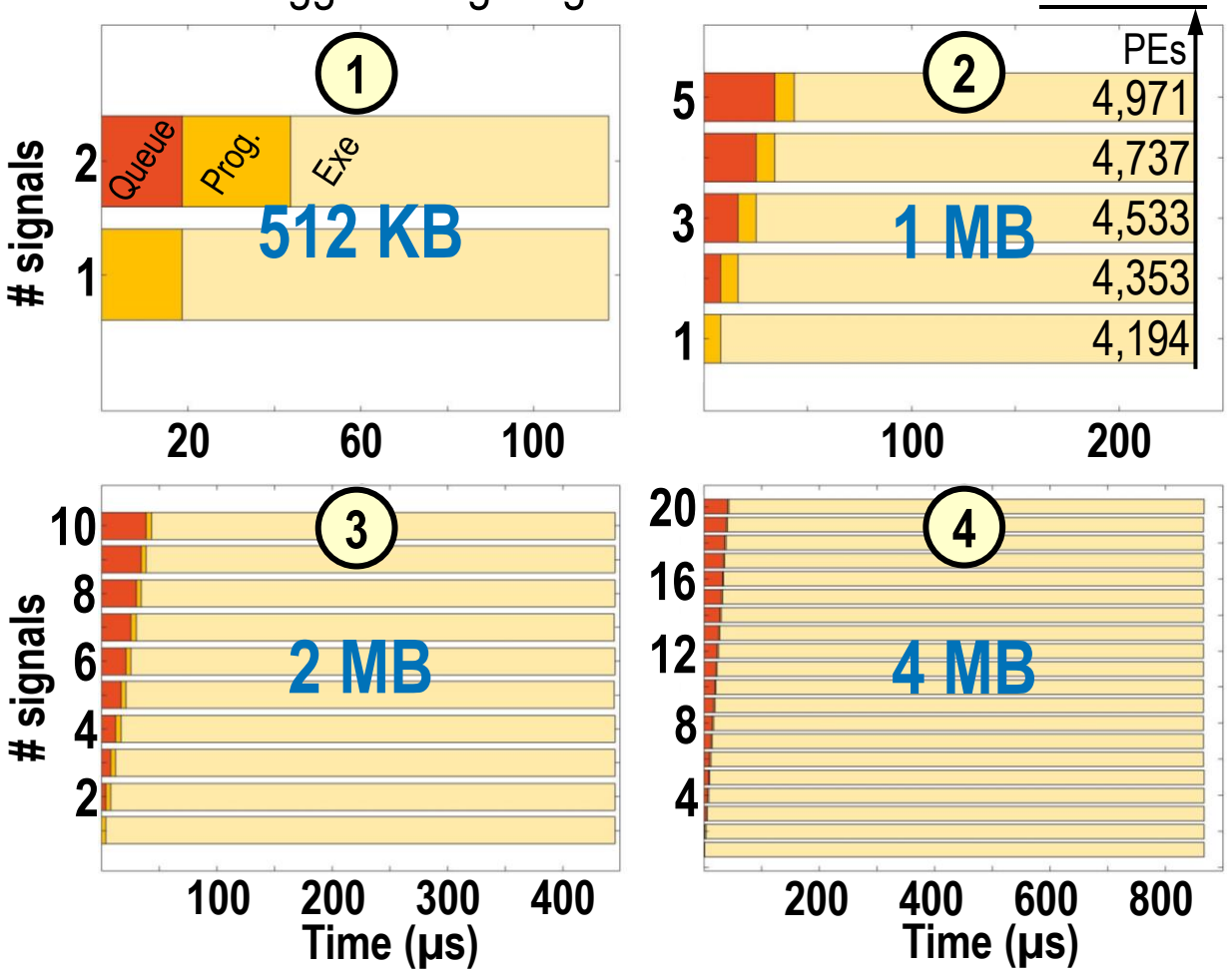
Multi-size compile is driven by # signals that need to be classified simultaneously



PT: Prog. Time | ET: Exe. Time | SC: Single Carrier

PT/PE	SNR	# PEs (Cost function: CT^a*PE^b)			
2ns	0dB	127x127	64x64	32x32	16x16
SC	PT (µs)	32.3	8.1	2.0	0.5
	ET (µs)	32.3	129.0	516.1	2064.5

Each staggered signal gets more PEs to match end time



TSMC 16nm (5.9mm^2/MB)

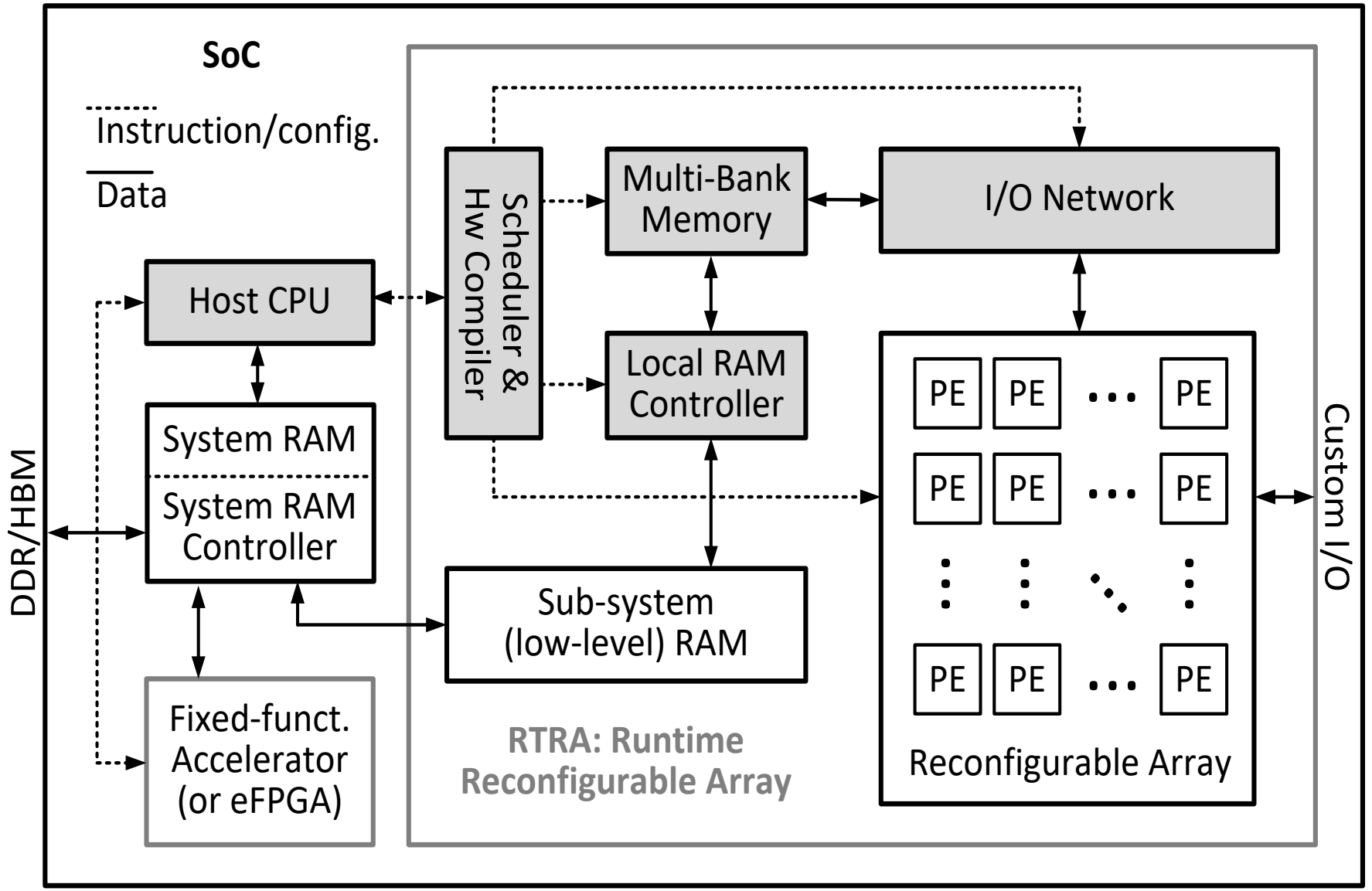
Line buffer size: Max-Time × 4 (KB) (×4 because of 16-b I/Q = 4B) (1ns Clk)

Architecture of RTRA

Hw/Sw balance

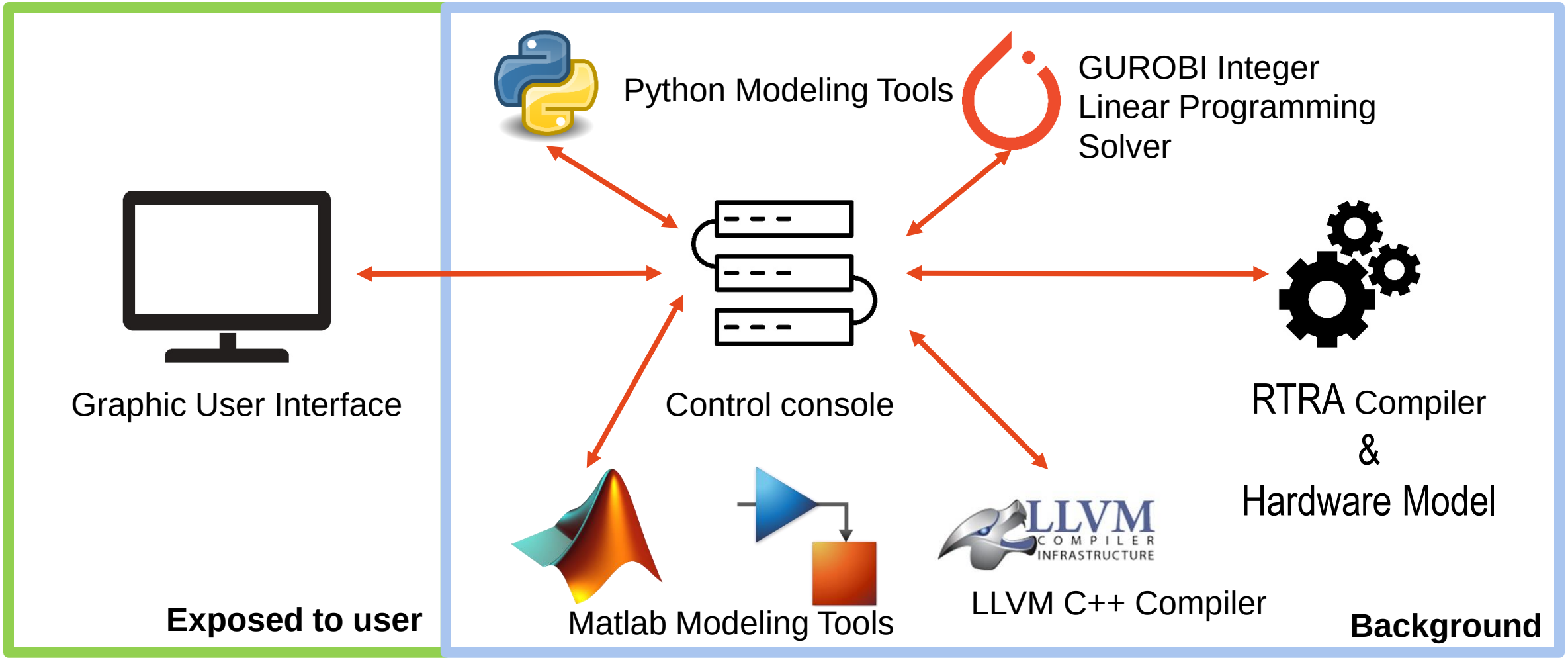
Hardware
- Cores
- Memory
- Interconnect
- I/O network

Software
- C++ to soft map
- Dynamic hard map



Software: RTRA Workstation

Unified user interface integrates multiple tool chains



Compiler Frontend

.CPP Source File

↓

Clang/LLVM Compiler

↓

Assembly Parser / DFG Generator

↓

DFG Optimizer

↓

Pre-processing

↓

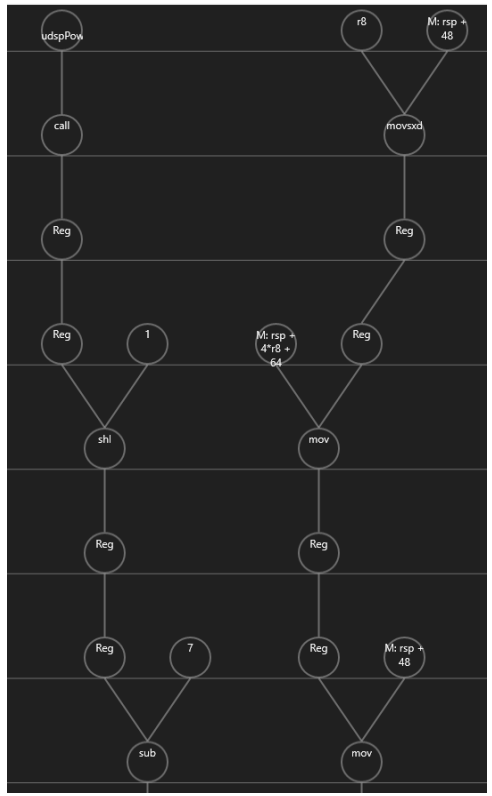
.CPP Source File

```cpp
int main()
{
    int c[10];
    int a = 3;
    int b = 5;
    int i = 0;
    int d = 6;
    while (i <= 10)
    {
        b += 9 + d;
        a +=  4 * udspOperator(b, 3);
        c[i] = 3 * udspOperator( a , 3) - 7;
        i++;
    }
}
```

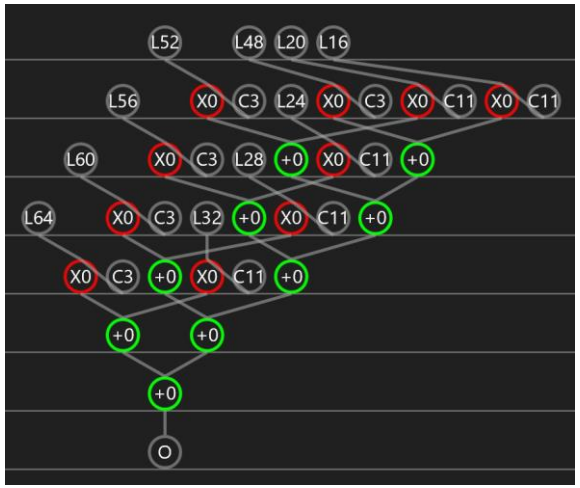
Clang/LLVM Compiler

```
.LBB0_1:                                # =>This Inner Loop Header: Depth=1
    .cv_loc 0 1 20 0                    # LLVM_Test.cpp:20:0
    cmp dword ptr [rsp + 48], 10
    jg  .LBB0_3
# %bb.2:                                #   in Loop: Header=BB0_1 Depth=1
.Ltmp1:
    .cv_loc 0 1 22 0                    # LLVM_Test.cpp:22:0
    mov eax, dword ptr [rsp + 44]
    add eax, 9
    add eax, dword ptr [rsp + 52]
    mov dword ptr [rsp + 52], eax
    .cv_loc 0 1 23 0                    # LLVM_Test.cpp:23:0
    mov ecx, dword ptr [rsp + 52]
    mov edx, 3
    call    "?udspOperator@@YAHHH@Z"
    shl eax, 2
    add eax, dword ptr [rsp + 56]
    mov dword ptr [rsp + 56], eax
    .cv_loc 0 1 24 0                    # LLVM_Test.cpp:24:0
    mov ecx, dword ptr [rsp + 56]
    mov edx, 3
    call    "?udspOperator@@YAHHH@Z"
    imul    eax, eax, 3
```

DFG Generator



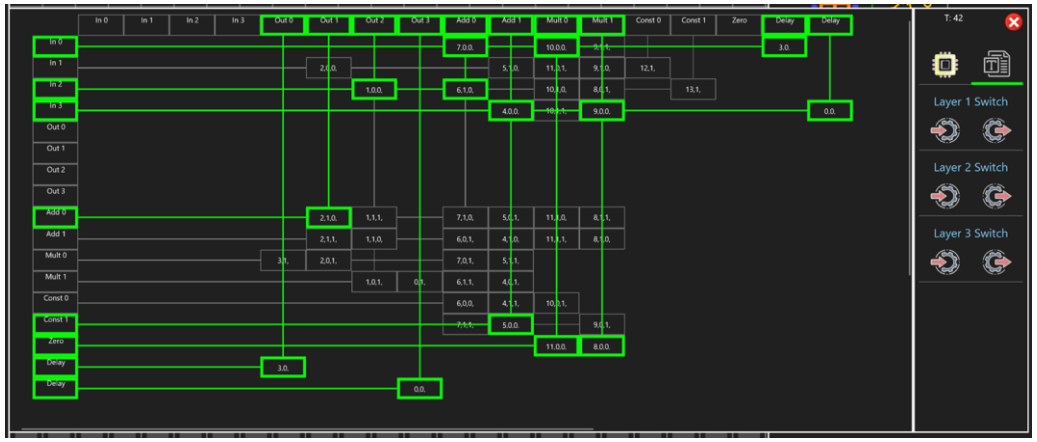
Optimizer + Pre-processing



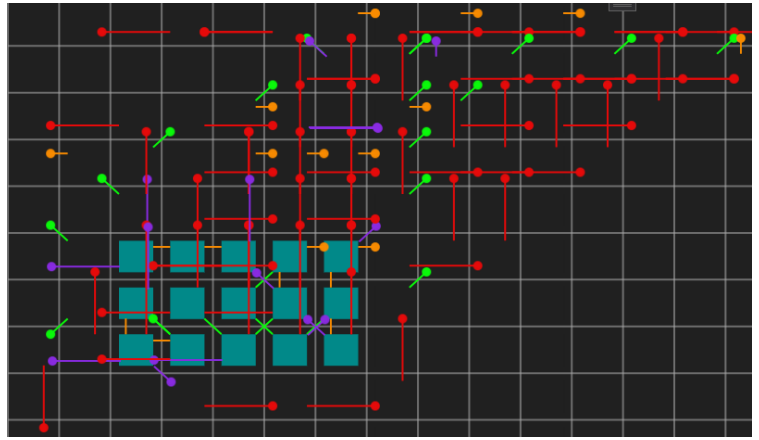
Compiler Backend

- Resource Binding
- Retiming
- Placement
- Local Route
- Global Route
- Bitstream & Simulation

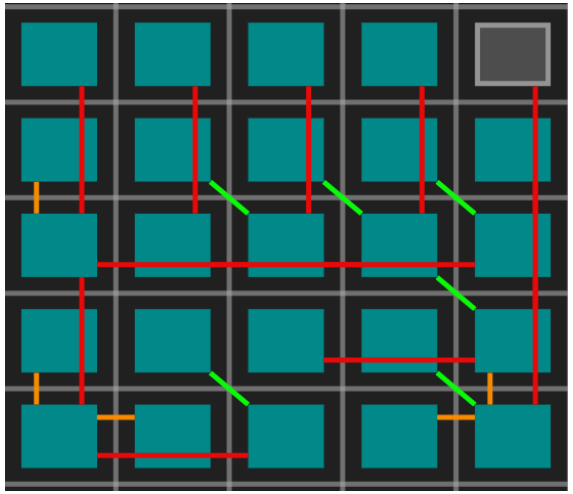
Resource Binding



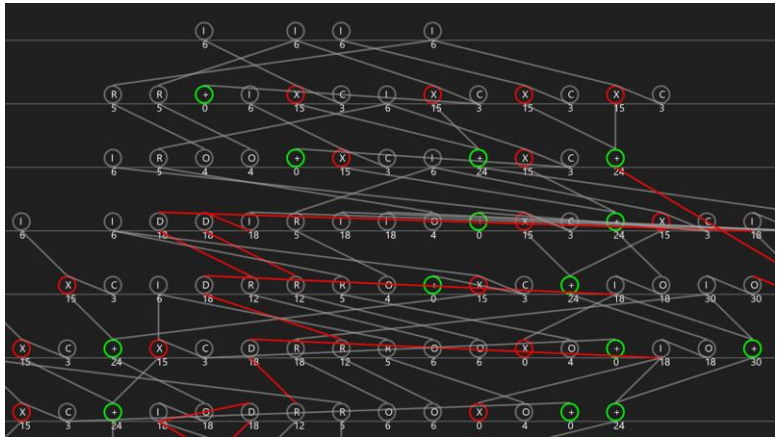
Global Route



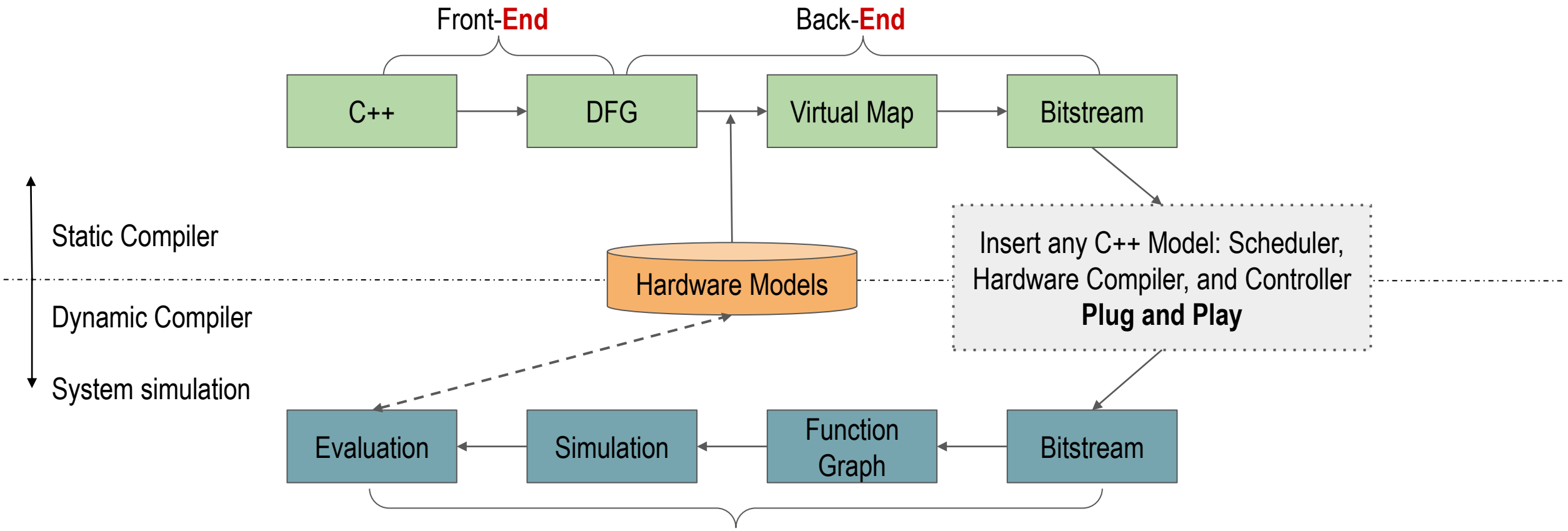
Placement & Local Route



Simulation



An End-to-End Software Stack



Future Datacenter: ACcelerator As a Service (ACAS) Model

- Dynamically compose and relocate accelerators on-the-fly and handle multiple accelerator requests simultaneously