Parallel Ultra Low-Power Processing (PULP) Systems

OPRECOMP SUMMER SCHOOL

19.07.2018

Davide Rossi
Frank K. Gürkaynak

davide.rossi@unibo.it

[1]Department of Electrical, Electronic
and Information Engineering

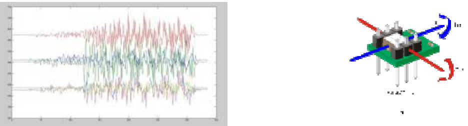
[2]Integrated Systems Laboratory

OPRECOMP
Open Transprecision Computing

Computing fot the Internet of Things

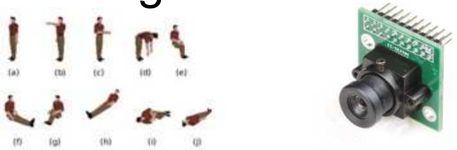
Sense

MEMS IMU

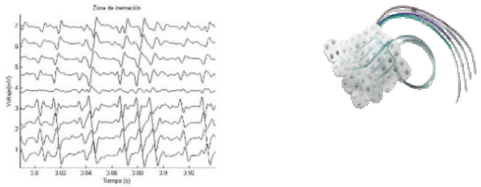


MEMS Microphone

ULP Imager



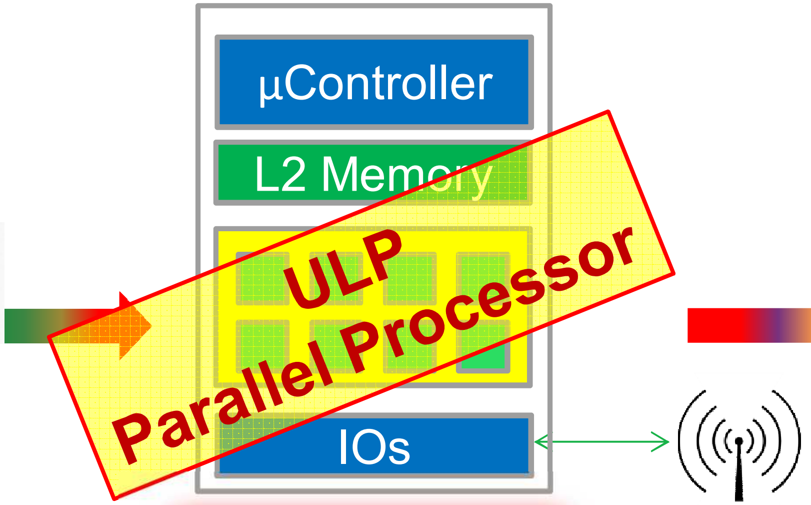
EMG/ECG/EIT



100 µW ÷ 2 mW

PULP

Analyze and Classify



1 ÷ 25 MOPS
1 ÷ 10 mW

Battery + Harvesting powered
→ a few mW power envelope

Transmit

Short range, medium BW

Low rate (periodic) data

SW update, commands

Long range, low BW

LoRa SEMTECH

Idle: ~1µW
Active: ~ 50mW

Near-Sensor Processing

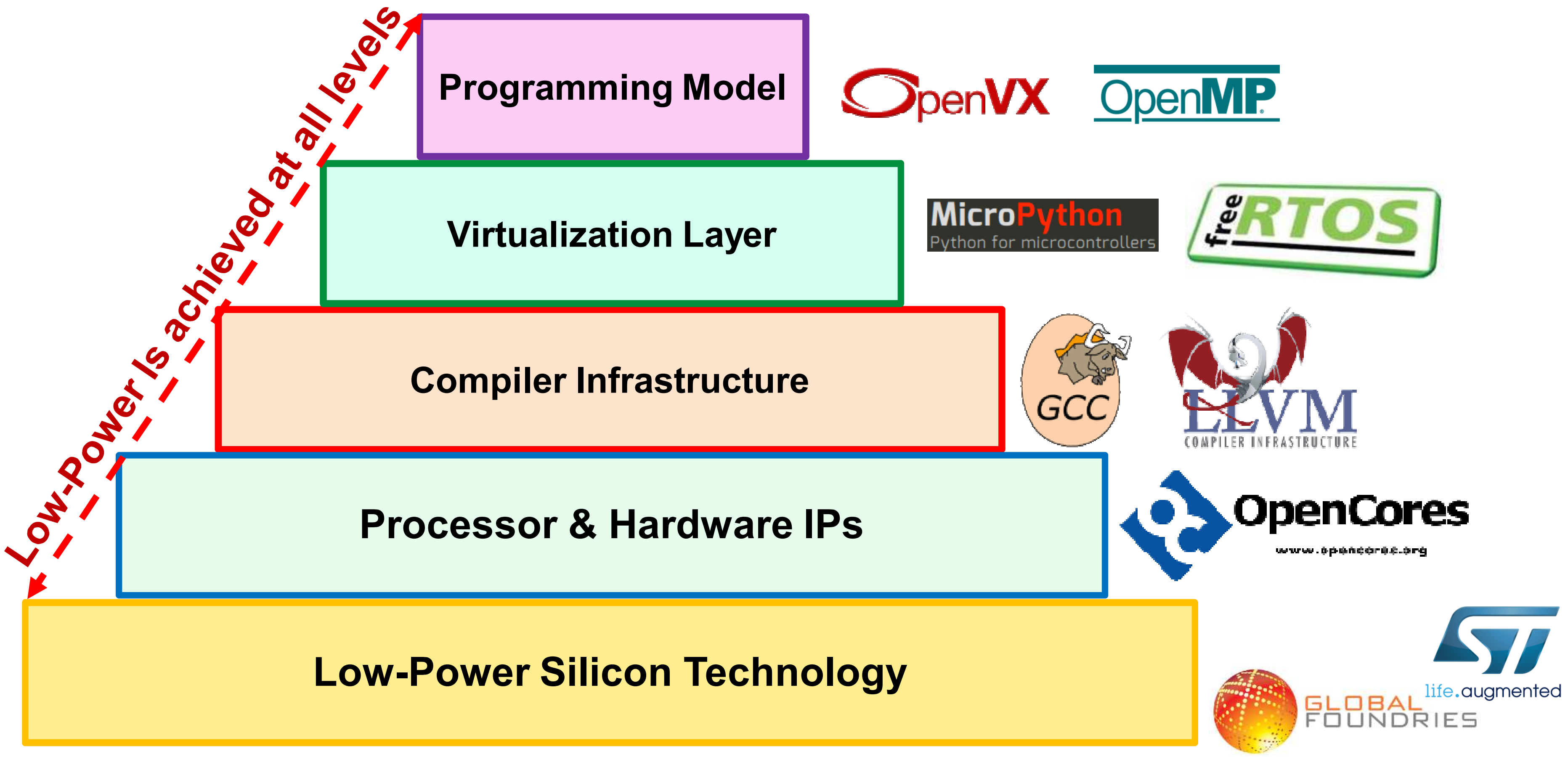
	INPUT BANDWIDTH	COMPUTATIONAL DEMAND	OUTPUT BANDWIDTH	COMPRESSION FACTOR
▪ Image				
Tracking: [*Lagroce2014]	80 Kbps	1.34 GOPS	0.16 Kbps	500x
▪ Voice/Sound				
Speech: [*VoiceControl]	256 Kbps	100 MOPS	0.02 Kbps	12800x
▪ Inertial				
Kalman: [*Nilsson2014]	2.4 Kbps	7.7 MOPS	0.02 Kbps	120x
▪ Biometrics				
SVM: [*Benatti2014]	16 Kbps	150 MOPS	0.08 Kbps	200x

➡ ***Extremely compact output (single index, alarm, signature)***

➡ ***Computational power of ULP µControllers is not enough***

➡ ***Parallel worloads***

PULP: pJ/op Parallel ULP computing



Programming Model

Virtualization Layer

Compiler Infrastructure

Processor & Hardware IPs

Low-Power Silicon Technology

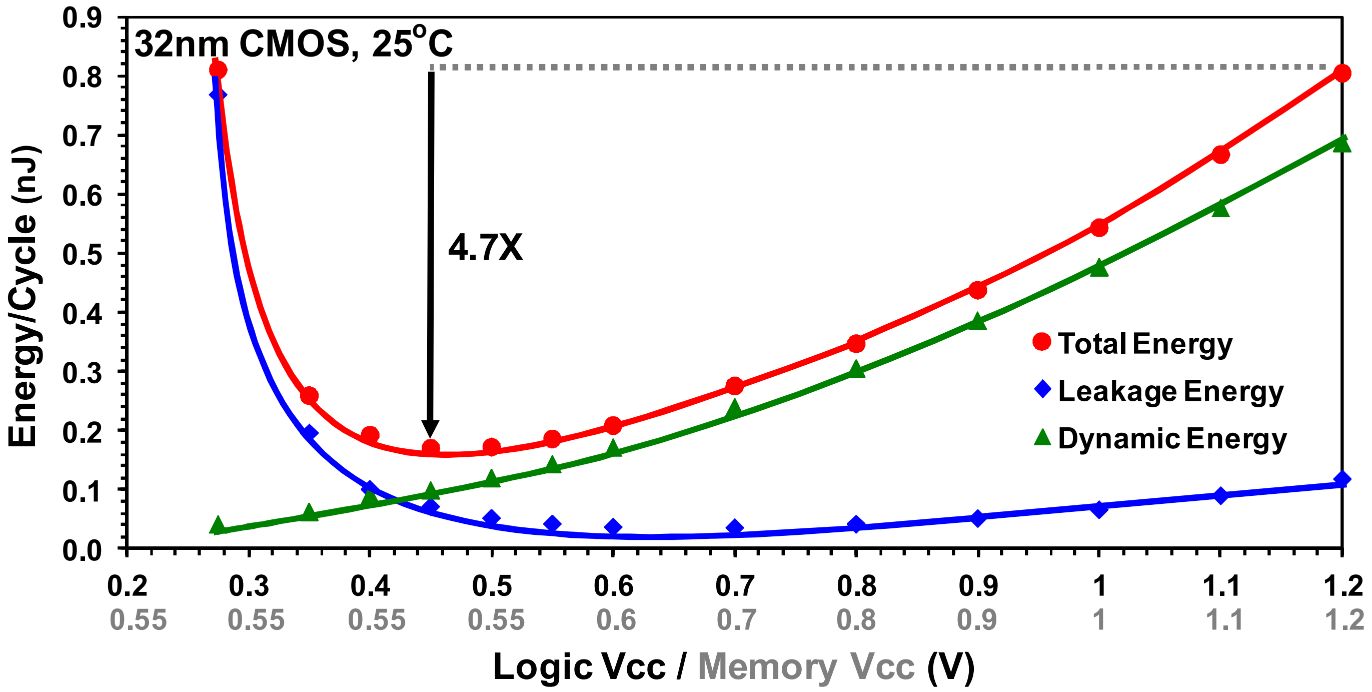
Low-Power Is achieved at all levels

Parallel + Programmable + Heterogeneous ULP computing

1mW-10mW active power

Parallel Ultra Low Power

Minimum Energy Operation

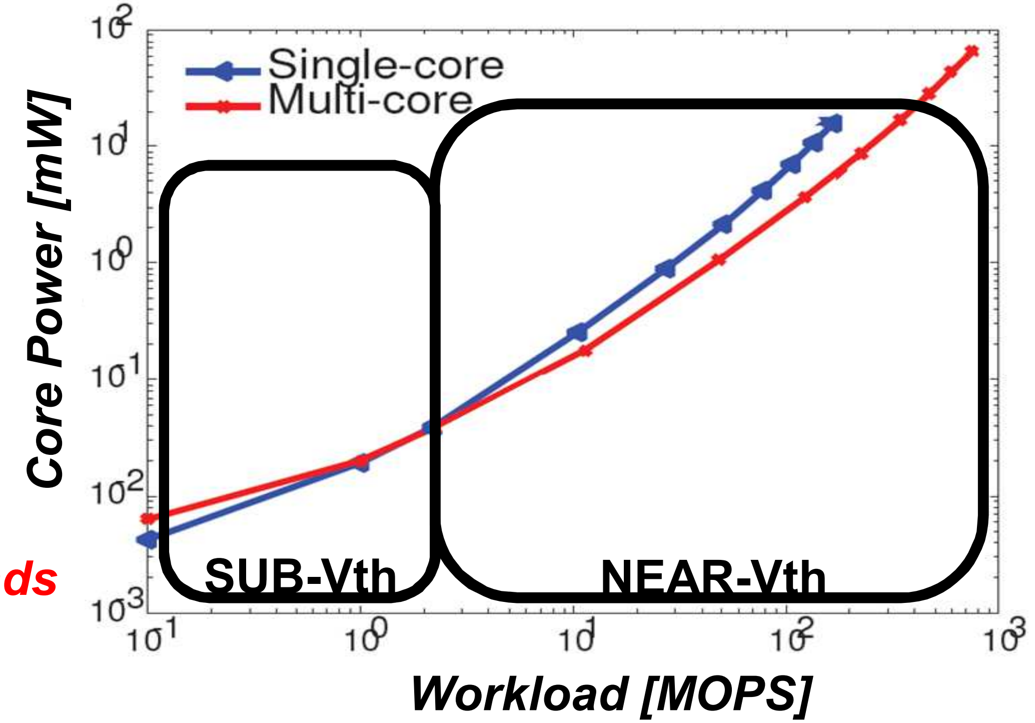


[VivekDeDATE2013]

Near-Threshold Computing (NTC):

- Don’t waste energy pushing devices in strong inversion
- Recover performance with parallel execution
- Aggressively manage idle power (switching, leakage)
- Manage Process and temperature variations in NT

Parallel NTC



High Workloads

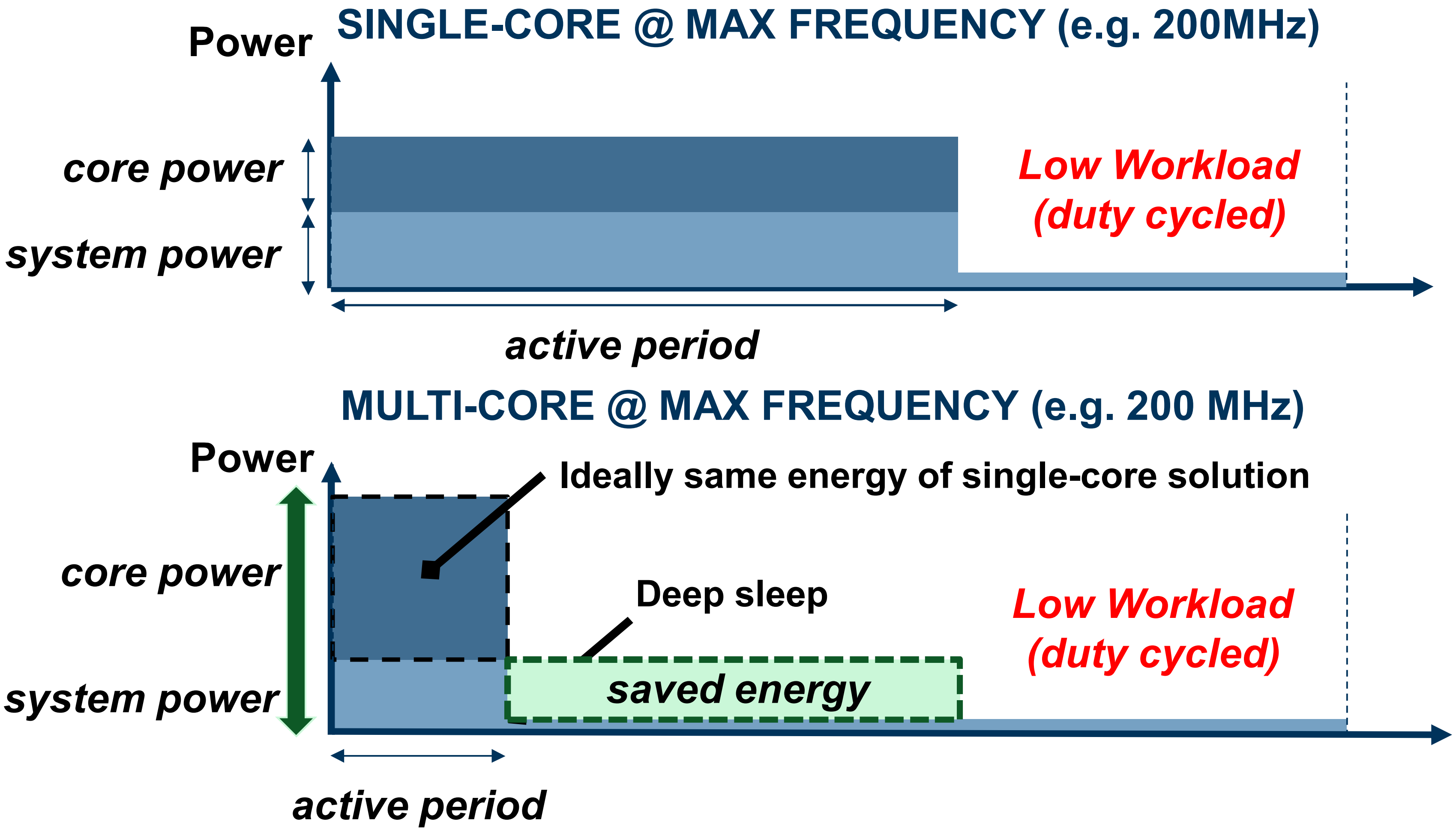
Low Workloads

[DoganICSDPTMO2011]

Target Workload [MOPS]	1-Core Energy Efficiency (ideal) [MOPS/mW]	4-Cores Energy Efficiency (ideal) [MOPS/mW]	Ratio
100	43	55	1.3x
200	33	50	1.5x
400	18	43	2.4x

*Measured on our first prototype

Parallel NTC + Race to Halt



- Going faster allows to integrate system power over a smaller period
- The main constraint here is the power envelope

Building PULP

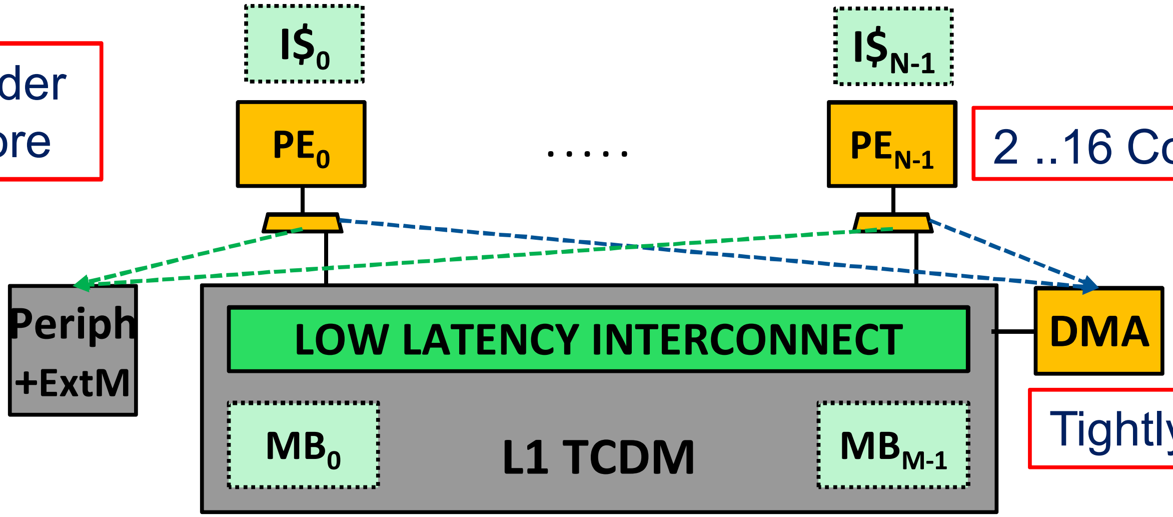
Building PULP

SIMD + MIMD + sequential

Private per-core instruction cache

4-stage, in-order OpenRISC core

DEMUX



2 ..16 Cores

Double buffering

Tightly Coupled DMA

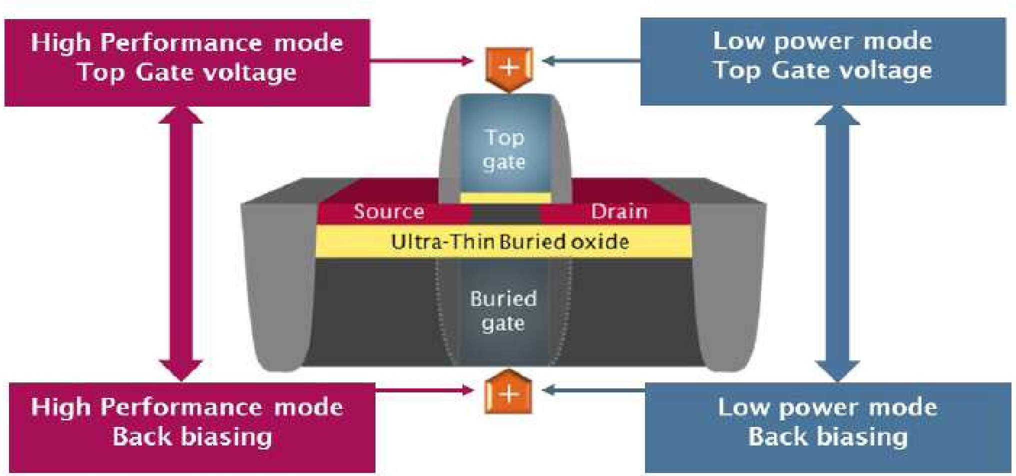
1 Cycle Shared Multi-Banked L1 Data Memory + Low Latency Interconnect

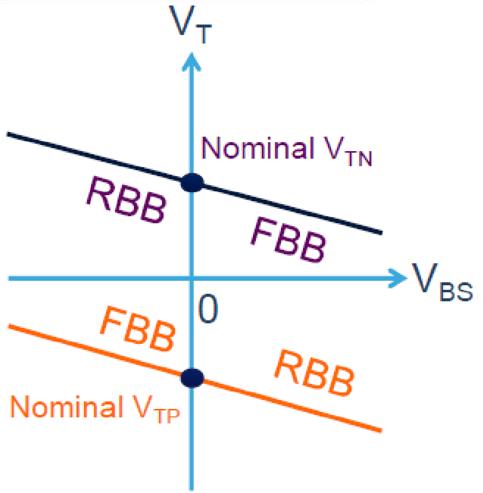
"GPU like" shared memory → low overhead data sharing

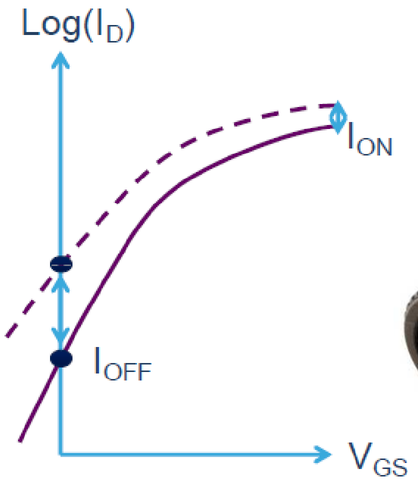
- Near Threshold but parallel → Maximum Energy efficiency when Active
- + strong power management for (partial) idleness

D. Rossi *et al.*, «PULP: A Parallel Ultra Low Power Platform for Next Generation IoT Applications," in *HOT CHIPS* 2015.

Near threshold FDSOI technology







NXP

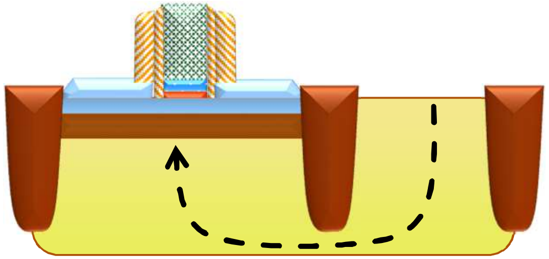
SONY

freescale™

SAMSUNG

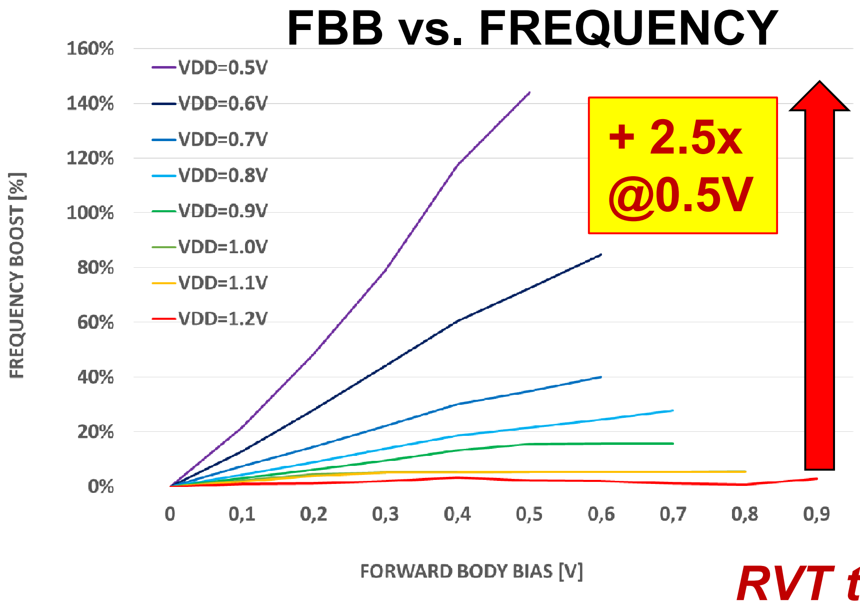
life.augmented

GLOBAL FOUNDRIES

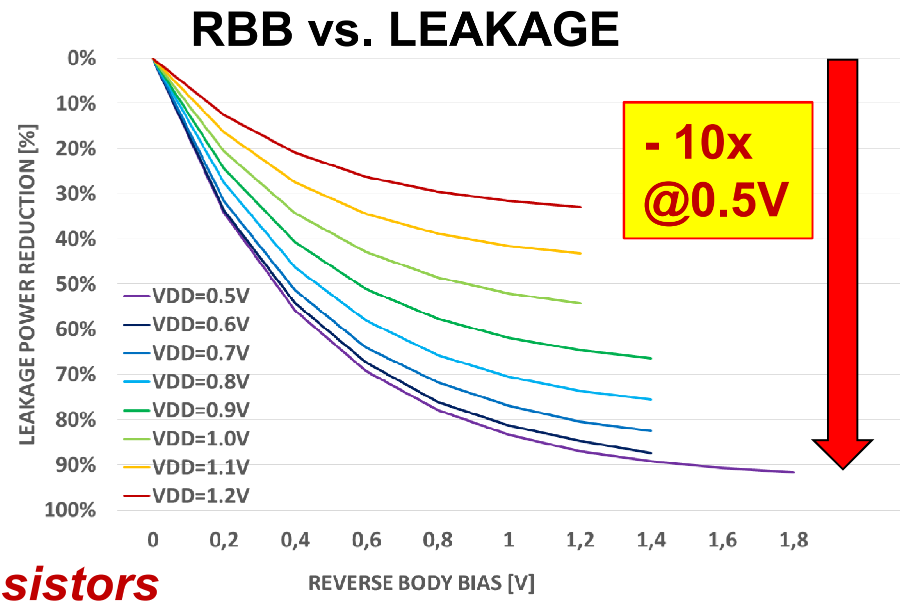


Body bias: Highly effective knob for power & variability management!

Near Threshold + Body Biasing Combined



+ 2.5x @0.5V



- 10x @0.5V

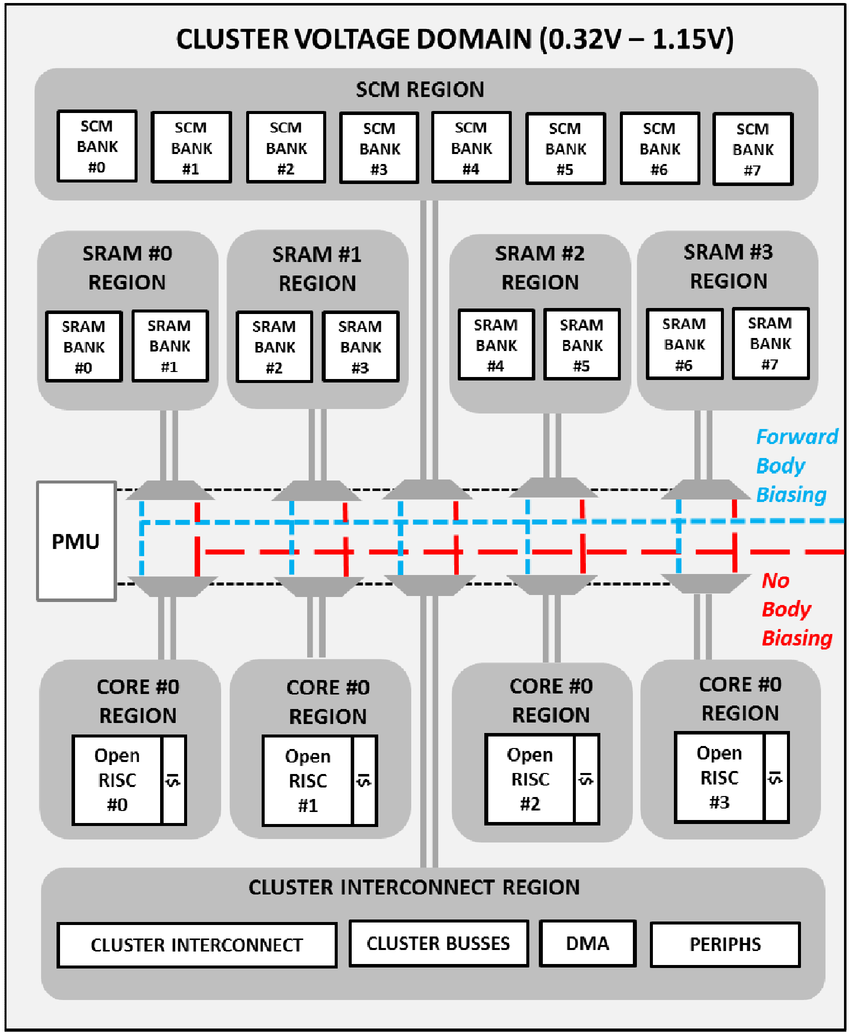
RVT transistors

- State retentive (no state retentive registers and memories)
- Ultra-fast transitions (tens of ns depending on n-well area to bias)
- Low area overhead for isolation (3µm spacing for deep n-well isolation)
- Thin grids for voltage distribution (small transient current for wells polarization)
- Simple circuits for on-chip VBB generation (e.g. charge pump)

But even with aggressive RBB leakage is not zero!

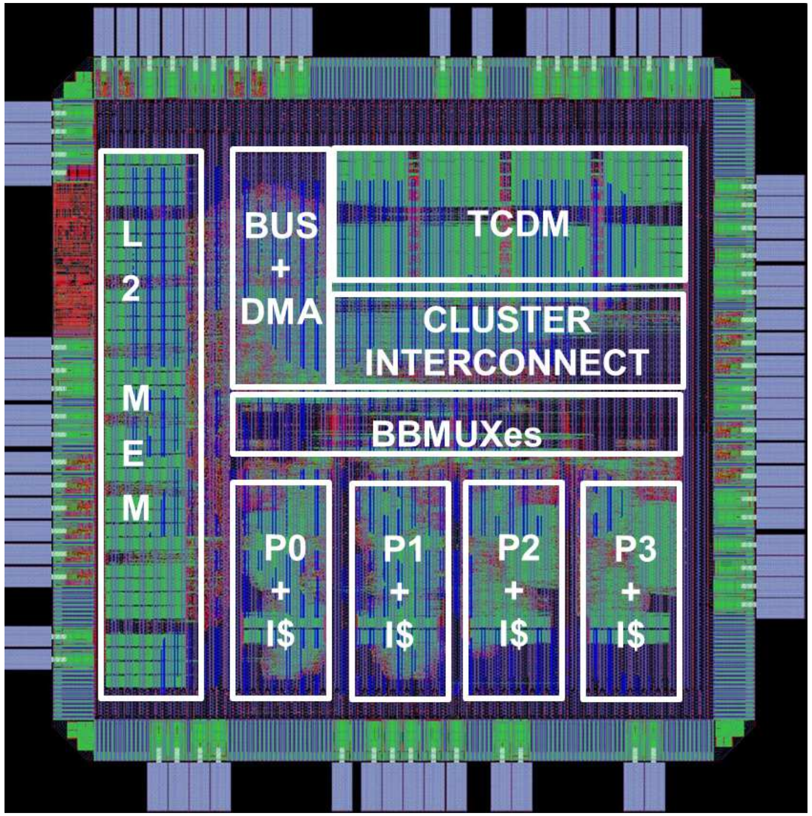
Selective, Fine Grained Body Biasing

- The cluster is partitioned in separate **clock gating and body bias regions**
- **Body bias multiplexers (BBMUXes)** control the well voltages of each region
- A **Power Management Unit (PMU)** automatically manages transitions between the operating modes
- Power modes of each region:
  - **Boost mode**: active + FBB
  - **Normal mode**: active + NO BB
  - **Idle mode**: clock gated + NO BB (in LVT) RBB (in RVT)



D. Rossi *et. al.*, «A 60 GOPS/W, −1.8V to 0.9V body bias ULP cluster in 28nm UTBB FD-SOI technology», in Solid-State Electronics, 2016.

PULPv1



CHIP FEATURES	
Technology	28nm FDSOI (RVT)
Chip Area	3mm^2
# Cores	4xOpenRISC
I$	4x1kbyte (private)
TCDM	16 kbyte
L2	16 kbyte
BB regions	6
VDD range	0.45V-1.2V (SRAM: 0.55V-1.2V)
VBB range	-1.8V - +0.9V
Perf. Range	1 MOPS-1.9GOPS
Power Range	100 µW-127 mW
Peak Efficiency	60 GOPS/W@20 MOPS, 0.55V

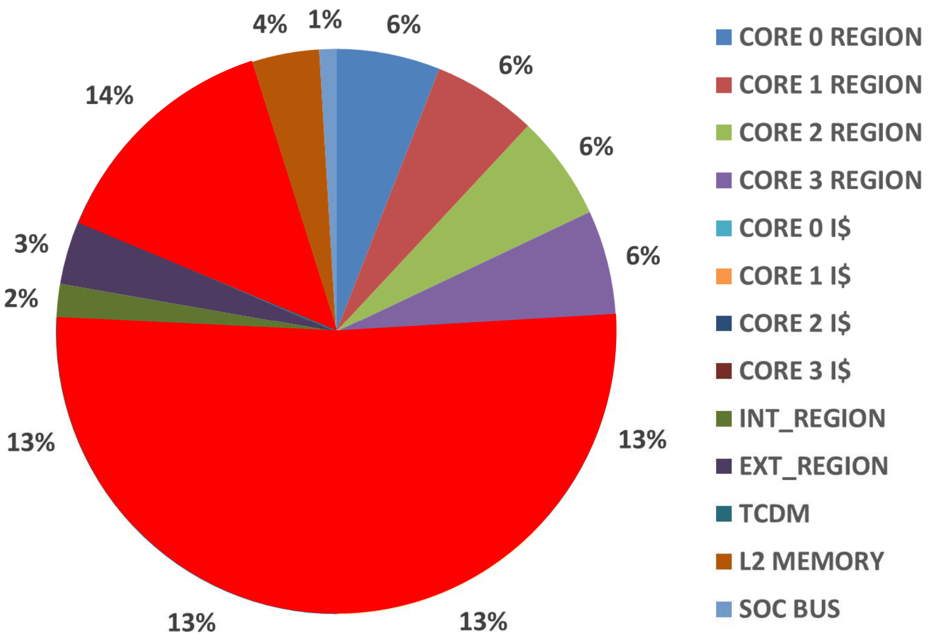
PULPv1 issues:

1) World record energy efficiency (60 MOPS/mW), but @ too small performance (20 MOPS)
2) SRAM limits voltage scalability (very well known problem…)

D. Rossi *et. al.*, «A 60 GOPS/W, −1.8V to 0.9V body bias ULP cluster in 28nm UTBB FD-SOI technology», in Solid-State Electronics, 2016.

The Memory bottleneck

PULPv1 POWER BREAKDOWN @ BEST ENERGY POINT:



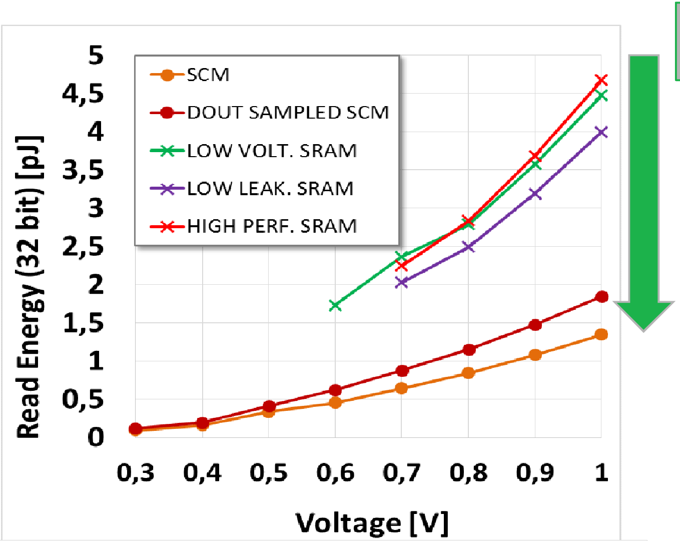
PULPv1 issues:

1) World record energy efficiency (60 MOPS/mW), but @ too small performance (20 MOPS)
2) SRAM limits voltage scalability (very well known problem…)
3) SRAM forms a huge bottleneck for energy efficiency (>60% of total power)

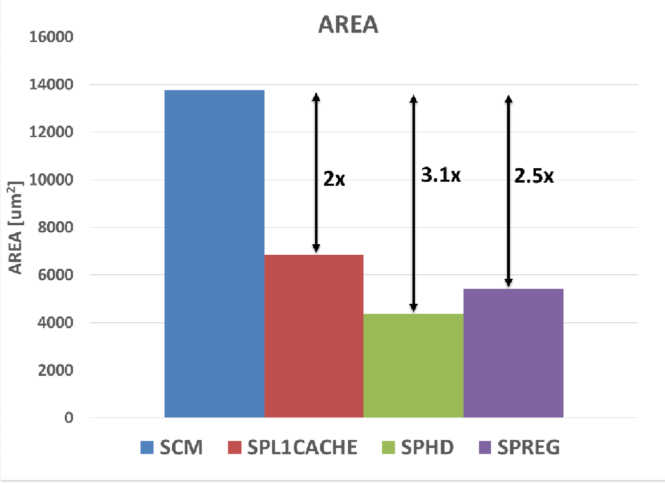
ULP (NT) Bottleneck: Memory

- “Standard” 6T SRAMs:
  - High VDDMIN
  - Bottleneck for energy efficiency
    - >50% of energy can go here!!!
- Near-Threshold SRAMs (8T)
  - Lower VDDMIN
  - Area/timing overhead (25%-50%)
  - High active energy
  - Low technology portability
- Standard Cell Memories:
  - Wide supply voltage range
  - Lower read/write energy (2x - 4x)
  - High technology portability
  - Major area overhead 4x → 2.7x with controlled placement

256x32 6T SRAMS vs. SCM

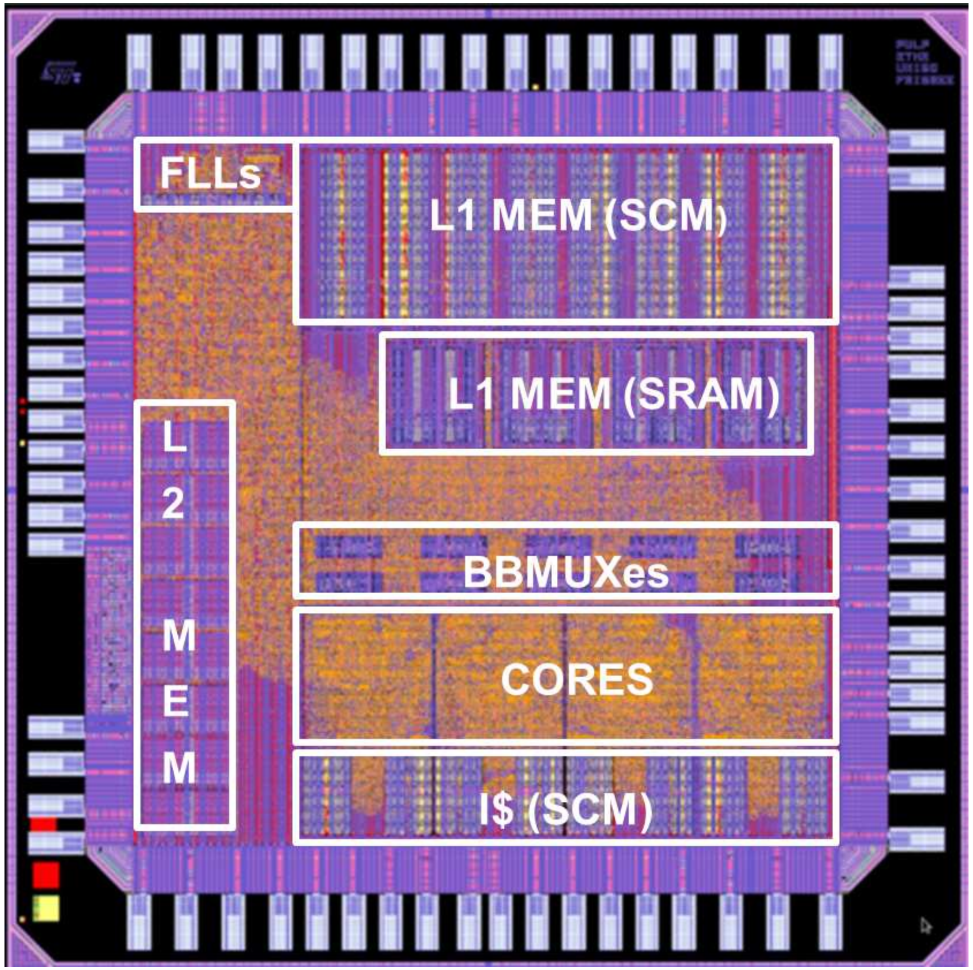


2x-4x



A. Teman *et.al.*, ‘Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement’, in ACM TDAES, May 2016

PULPv2



CHIP FEATURES	
Technology	28nm FDSOI (LVT)
Chip Area	3mm^2
# Cores	4xOpenRISC
I$ (SCM)	4x1kbyte (private)
TCDM	32 + 8 Kbyte
L2	64 kbyte
BB regions	10
VDD range	0.3-1.2V (0.5-1.2V)
VBB range	0V-2V
Perf. Range	1 MOPS - 4 GOPS
Power Range	10µW - 300 mW
Peak Efficiency	192 GOPS/W@0.5V

D. Rossi *et al.*, "Energy-Efficient Near-Threshold Parallel Computing: The PULPv2 Cluster," in *IEEE Micro*, Sep./Oct. 2017.

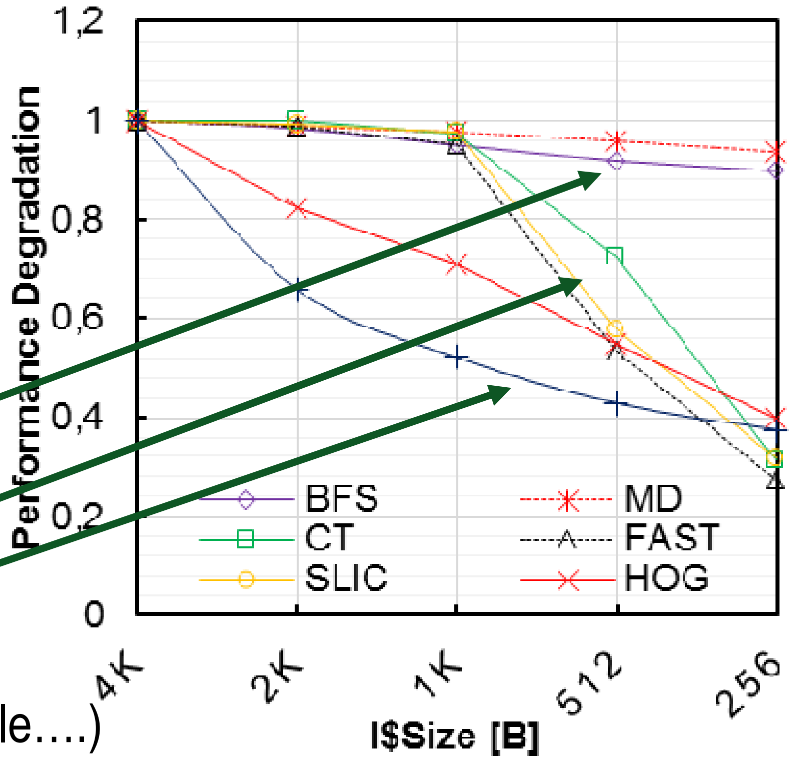
I$: a Look Into ‘Real Life’ Applications

SCM-BASED I$ IMPROVES EFFICIENCY BY ~2X ON SMALL BENCHMARKS, BUT…

Survey of State of The Art

Exixting ULP processors	Latch based I$
REISC (ESSCIRC2011)	64b
Sleepwalker (ISSCC 2012)	128b
Bellevue (ISCAS 2014)	128b

Applications on PULP



SHORT JUMP LOOP BASED APPLICATIONS

LONG JUMP APPLICATIONS

LIBRARY BASED

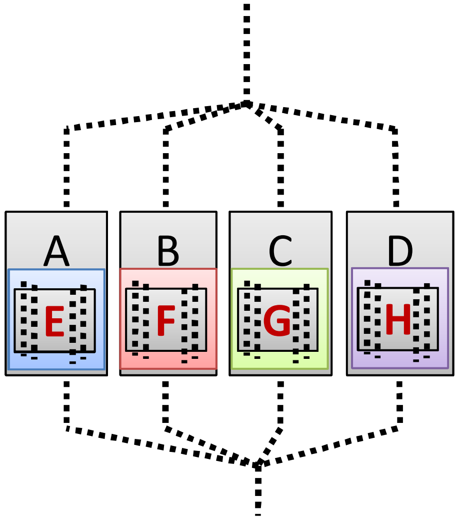
Issues:

1) Area Overhead of SCMs (4Kb/core not affordable….)
2) Capacity miss (with small caches)
3) Jumps due to runtime (e.g. OpenMP, OpenCL) and other function calls

OpenMP for PULP

- OpenMP on PULP:
  - Lightweight implementation on top of a bare-metal runtime (custom GCC libgomp)
  - A subset of OpenMP 3.0 supported (e.g. no tasking)
  - Power management embedded in the runtime (transparent to the end-user)
- Architectural Implications
  - #pragmas are translated into runtime function calls
  - **Function calls cause I$ cache pollution**
  - **Call to OpenMP functions feature intrinsic overhead**

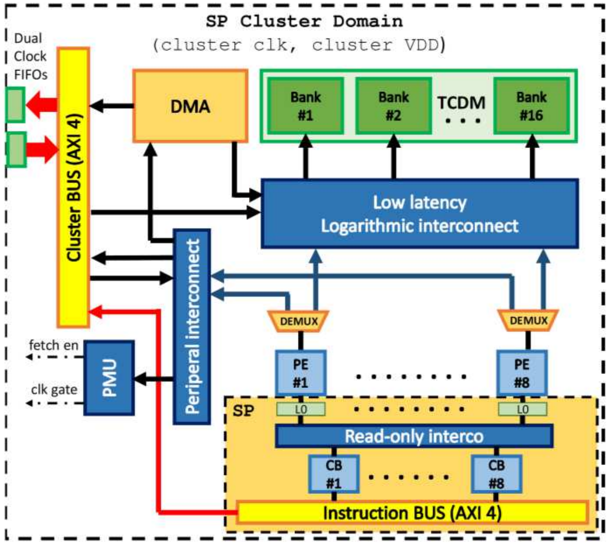
```
#pragma omp parallel
{
    #pragma omp sections
    {
        #pragma omp section
        {
A.          #pragma omp parallel
            Task_A ();
        }
        #pragma omp section
        {
B.          #pragma omp parallel
            Task_B ();
        }
        #pragma omp section
        {
C.          #pragma omp parallel
            Task_C ();
        }
        #pragma omp section
        {
D.          #pragma omp parallel
            Task_D ();
        }
    }
}
```

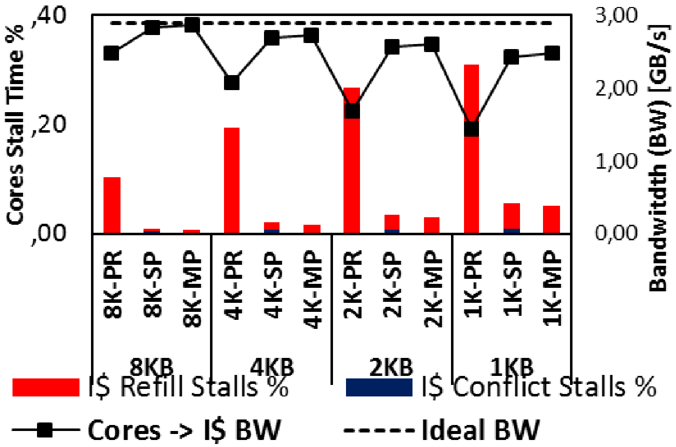


A. Marongiu et. al., "An OpenMP Compiler for Efficient Use of Distributed Scratchpad Memory in MPSoCs," in IEEE Transactions on Computers, Feb. 2012.

The Solution: Shared I$

- Share instruction cache
  - OK for data parallel execution model
  - Not OK for task parallel execution model, or very divergent parallel threads
- Architectures
  - SP: single-port banks connected through a read-only interconnect
    - Pros: Low area overhead
    - Cons: Timing pressure, contention
  - MP: Multi-ported banks
    - Pros: High efficiency
    - Cons: Area overhead (several ports)
- Results
  - Up to 40% better performance than private I$
  - Up to 30% better energy efficiency
  - Up to 20% better energy*area efficiency



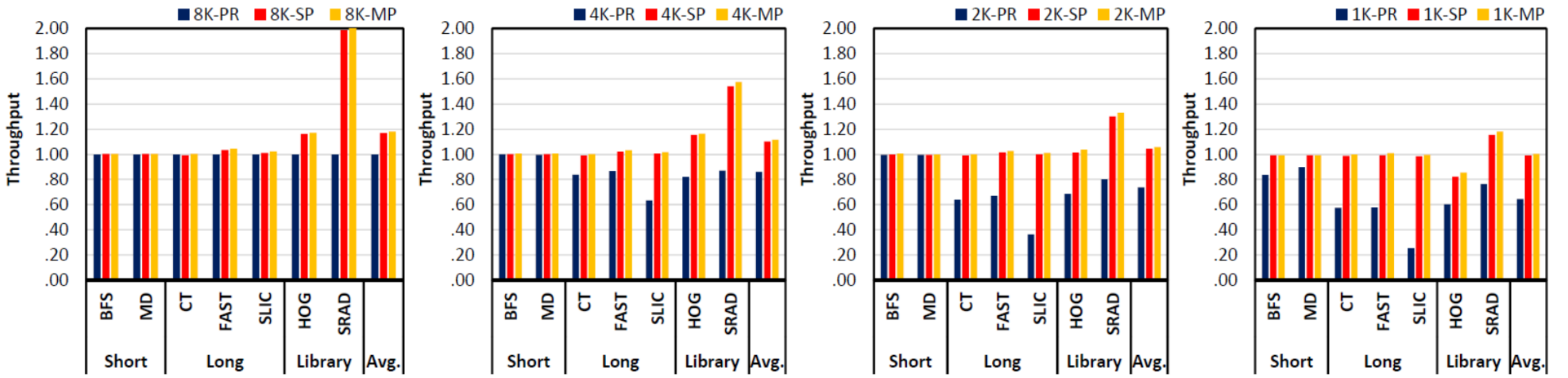


I. Loi, *et.al.*, "The Quest for Energy-Efficient I$ Design in Ultra-Low-Power Clustered Many-Cores," in IEEE Transactions on Multi-Scale Computing Systems, In Press.

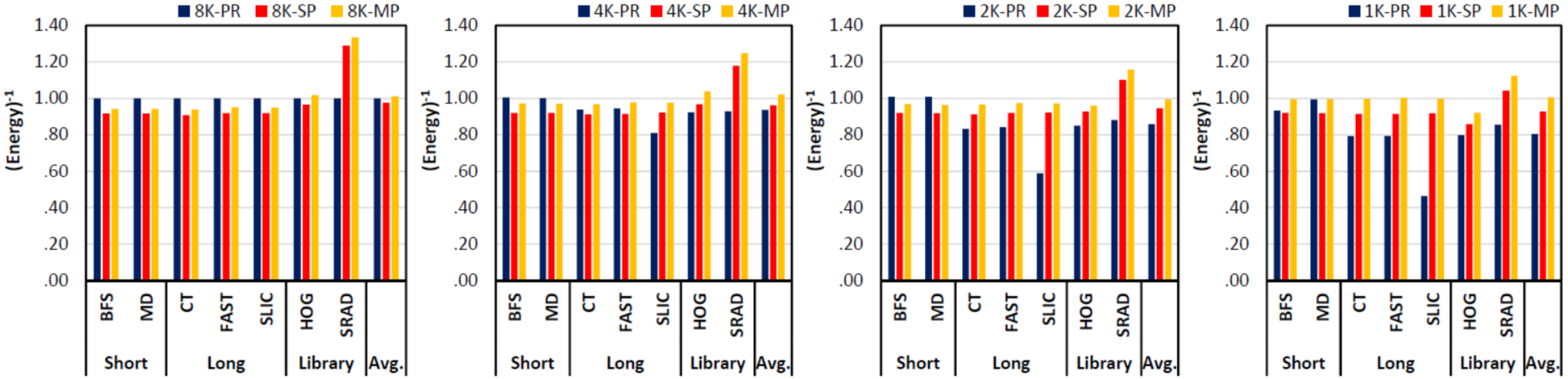
Shared Cache Benchmarking

EQUI-AREA ANALYSIS FOR AN 8-CORE CONFIGURATION

+40%

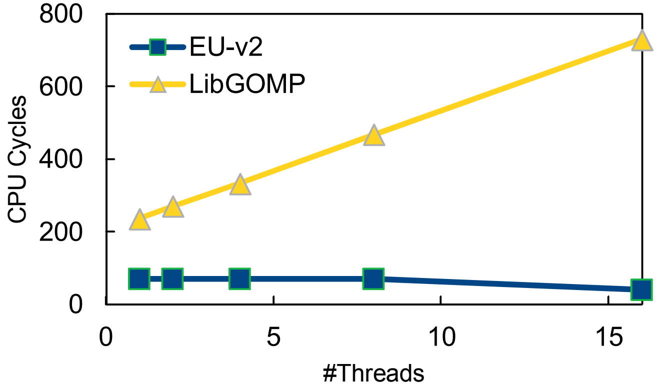


-20%

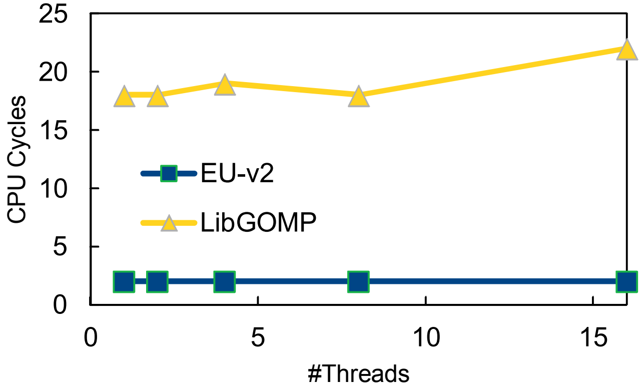


HW Synchronizer: Impact on OpenMP primitives

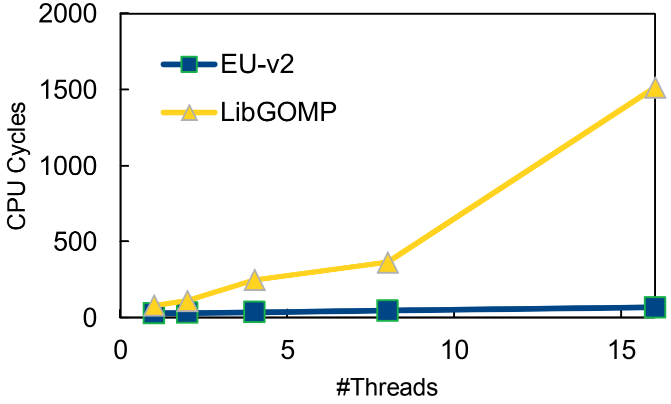
#Parallel



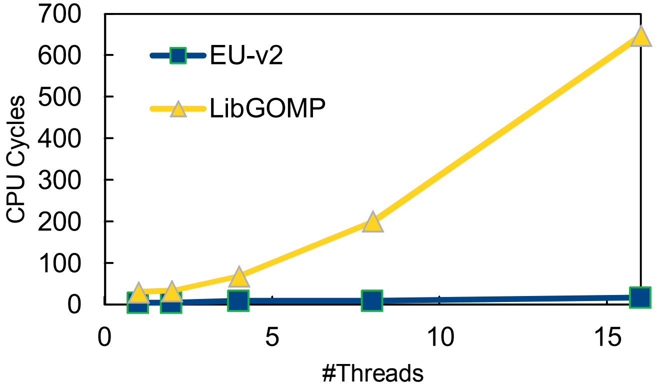
#Barrier



#Sections



#Critical

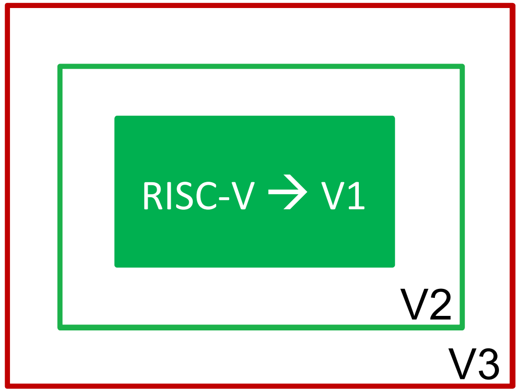


- Cost of OpenMP runtime reduced by more than one order of magnitude
- Better scalability with number of cores

Extending RISC-V for NSP

<32-bit precision → **SIMD2/4 opportunity**

1. HW loops and Post modified LD/ST
2. Bit manipulations
3. Packed-SIMD ALU operations with dot product
4. Rounding and Normalizazion
5. Shuffle operations for vectors



V1 Baseline RISC-V RV32IMC

V2 HW loops
Post modified Load/Store
Mac

V3 SIMD 2/4 + DotProduct + Shuffling
Bit manipulation unit
Lightweight fixed point

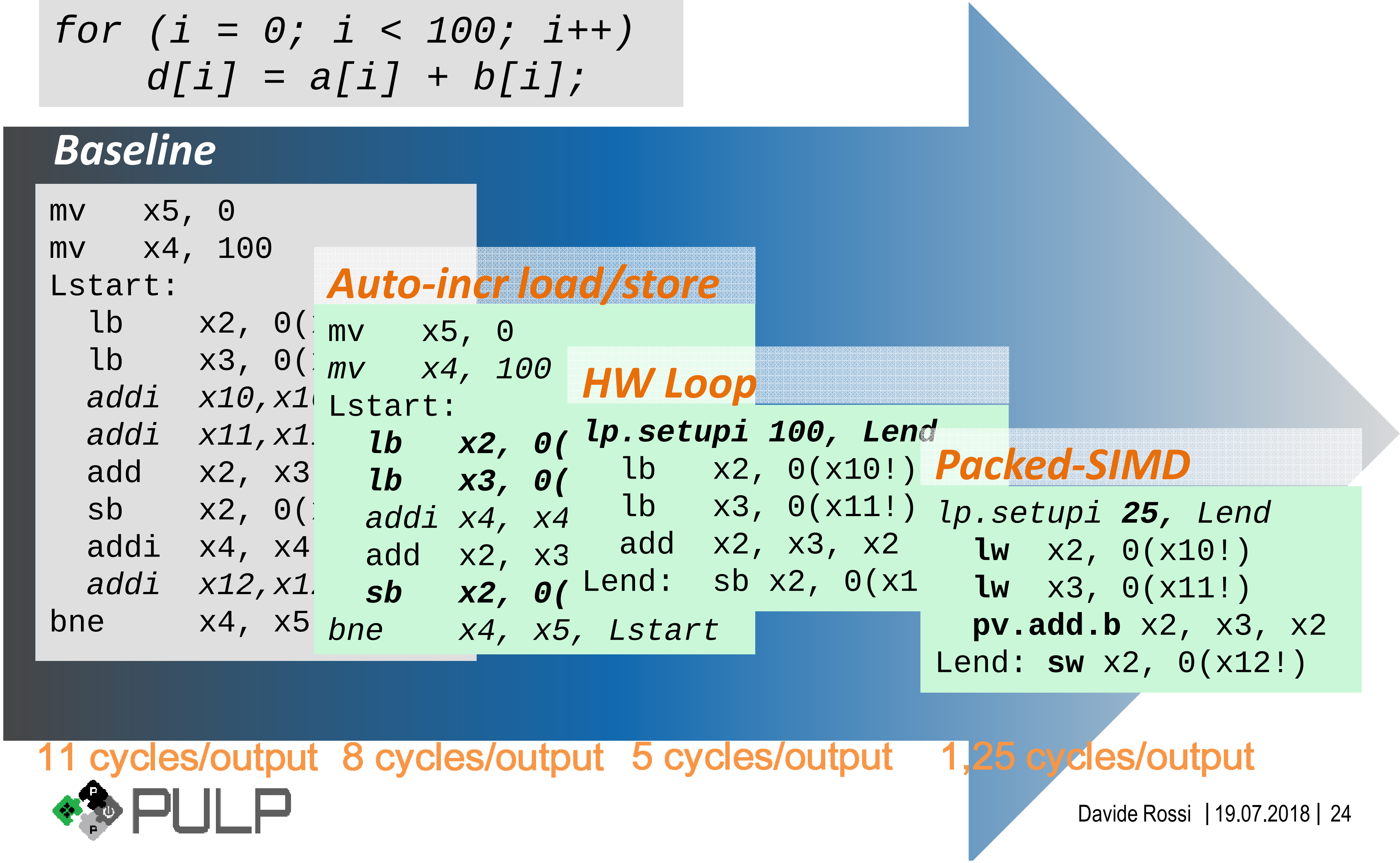
Small Power and Area overhead

PULP

M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," in IEEE TVLSI, Oct. 2017.

ISA Extensions improve performance

```
for (i = 0; i < 100; i++)
    d[i] = a[i] + b[i];
```



Baseline

```
mv    x5, 0
mv    x4, 100
Lstart:
   lb    x2, 0(
   lb    x3, 0(
   addi  x10,x1
   addi  x11,x1
   add   x2, x3
   sb    x2, 0(
   addi  x4, x4
   addi  x12,x1
bne   x4, x5
```

Auto-incr load/store

```
mv    x5, 0
mv    x4, 100
Lstart:
   lb    x2, 0(
   lb    x3, 0(
   addi  x4, x4
   add   x2, x3
   sb    x2, 0(
bne   x4, x5, Lstart
```

HW Loop

```
lp.setupi 100, Lend
   lb    x2, 0(x10!)
   lb    x3, 0(x11!)
   add   x2, x3, x2
Lend:  sb x2, 0(x1
```

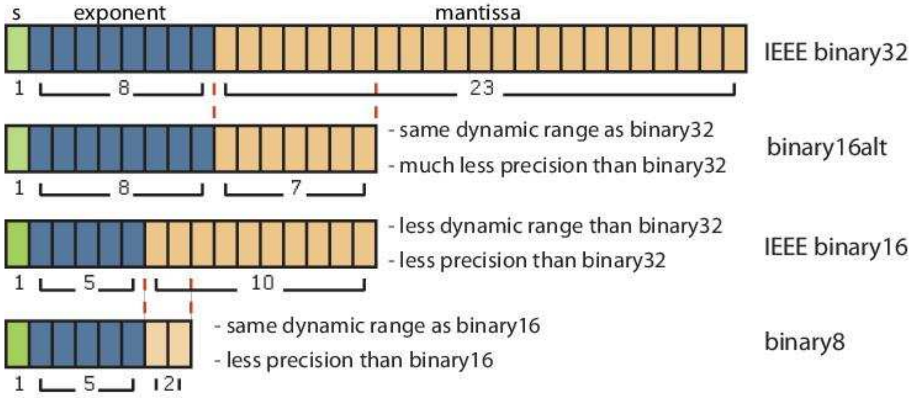
Packed-SIMD

```
lp.setupi 25, Lend
   lw   x2, 0(x10!)
   lw   x3, 0(x11!)
   pv.add.b x2, x3, x2
Lend: sw x2, 0(x12!)
```

11 cycles/output 8 cycles/output 5 cycles/output 1,25 cycles/output

What About FP?

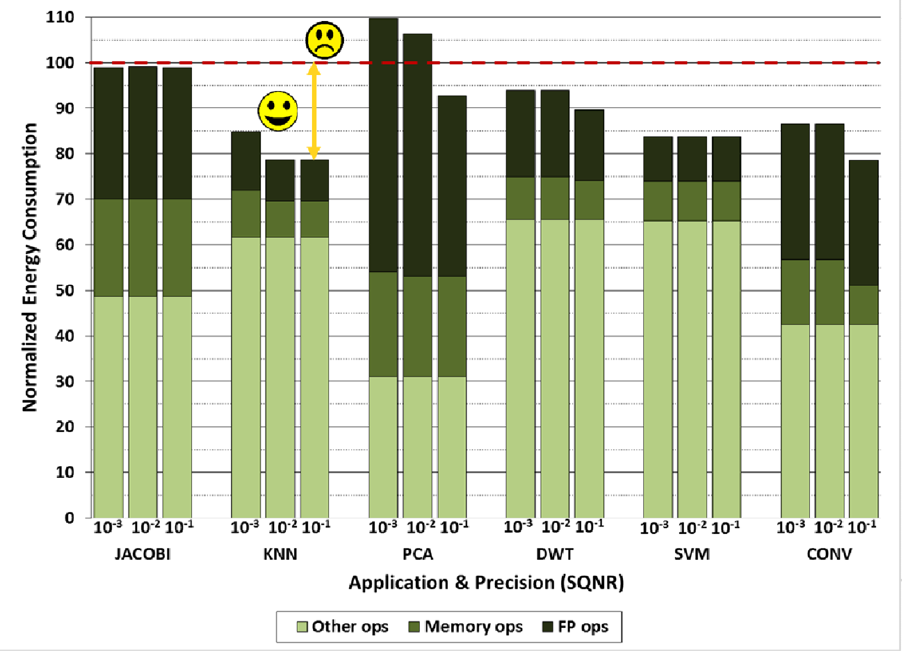
- The adoption of floating point formats requiring a lower number of bits (***smallFloats***) can reduce execution time and energy consumption
  - **Simpler Logic** (smaller pj/op)
  - **Vectorization** (smaller execution time)
- SW support:
  - ***flexFloat*** library for emulation of smallFloat format
  - Automatic exploration tool to tune precision of ***individual FP operations for the required application accuracy***
- Hardware support:
  - smallFloat ***scalar*** and ***vector*** operations
  - **Casting** operations to move data through different FP types



- same dynamic range as binary32
- much less precision than binary32

- less dynamic range than binary32
- less precision than binary32

- same dynamic range as binary16
- less precision than binary16

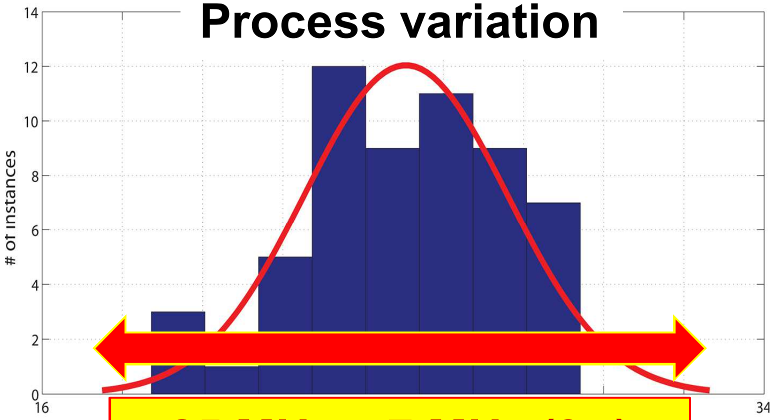


Tagliavini et. A.l, «A Transprecision Floating-Point Platform for Ultra-Low Power Computing», DATE 2018

Process & Temperature Variations in NTC

Normalized Frequency

Process variation



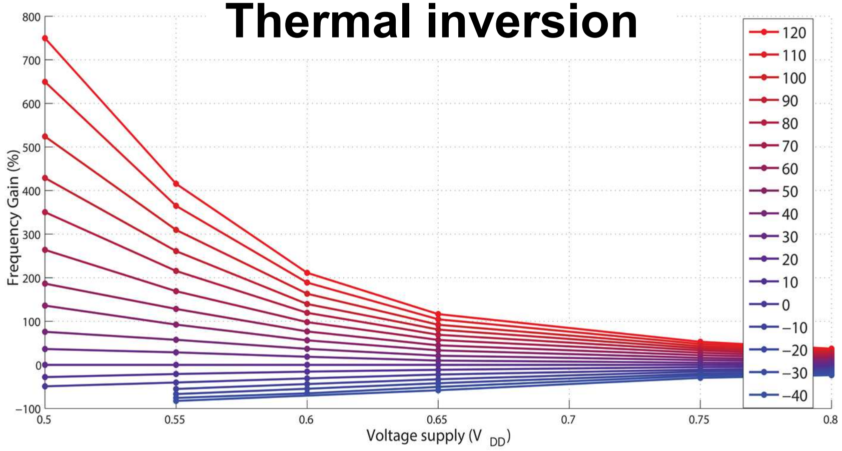
25 MHz ± 7 MHz (3σ)

120°C

100x @0.5V

-40°C

Thermal inversion



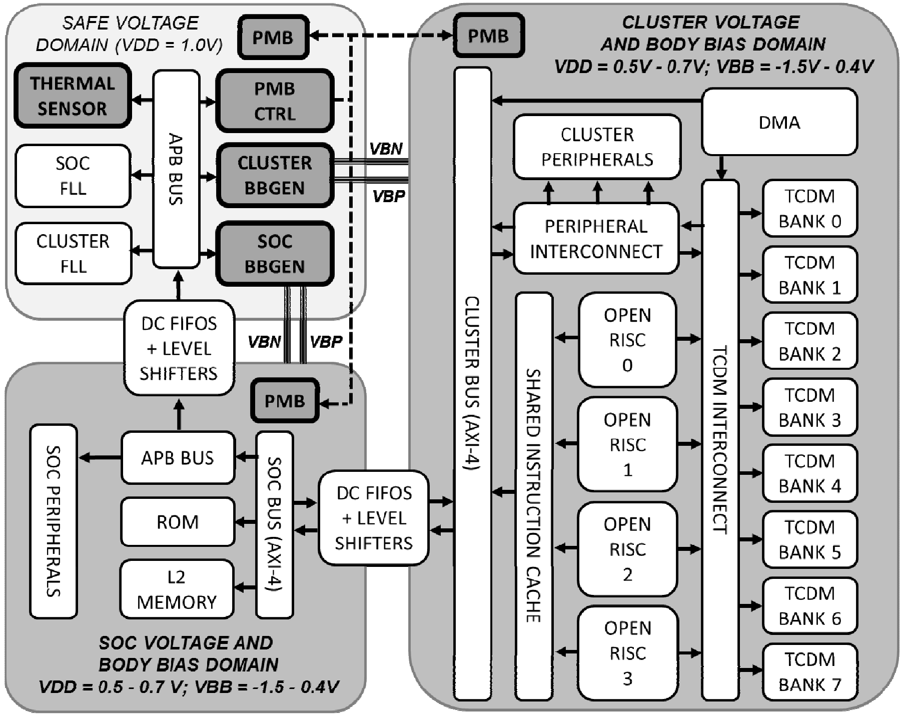
0.5V

0.8V

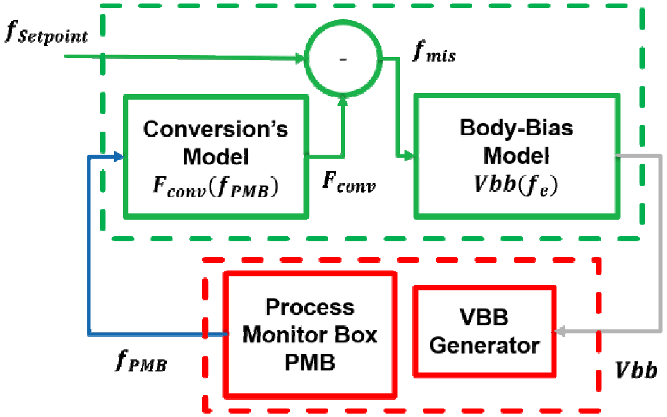
Voltage

Process variation over a distribution of 60 chips @ 0.6V

Body Bias-Based Performance Monitoring and Control



→ Mixed Hardware/Software control loop exploiting on-chip frequency measurement (PMB)



- 🟩 Software functions
- 🟥 Hardware components
- ▬ Memory mapped status register reading
- ▬ Memory mapped control register writing

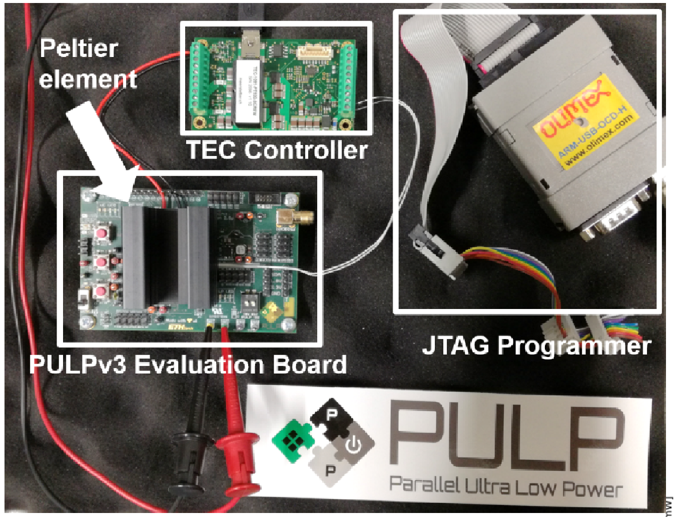
Goal:

- Compensate the effects of external factors like ambient temperature
- Reduce PVT Margings at design time
- Improve Energy Efficiency

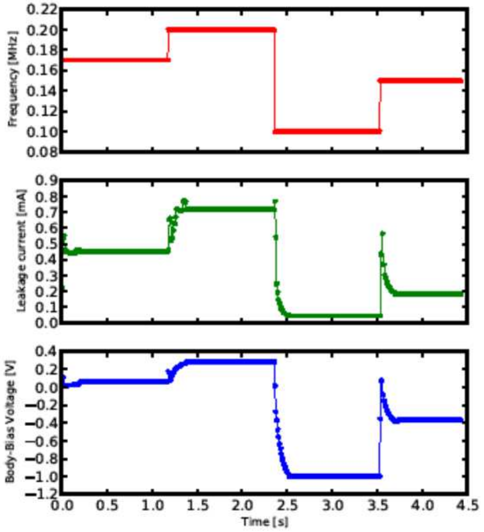
D. Rossi et al., "A Self-Aware Architecture for PVT Compensation and Power Nap in Near Threshold Processors," in IEEE Design & Test, Dec. 2017.

Dynamic PVT Compensation

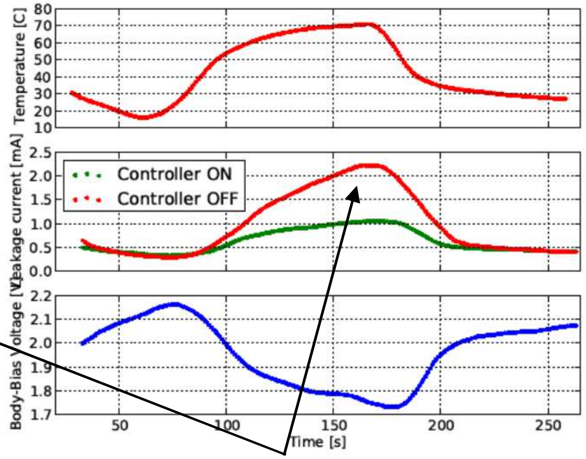
→PULPv3 board featuring voltage regulator (down to 0.5V)
→Peltier element to heat-up / cool the samples
→TEC controller to drive the peltier element
→Embecosm MAGEEC Energy Monitoring Shield for power measurements
→JTAG Programmer for loading code on PULPv3
→Host laptop to show plots



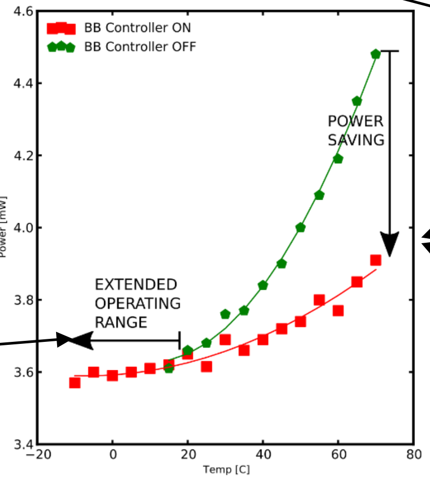
Frequency Tracking



Temperature Tracking



Results



Extended operating range for zero-margin design (i.e. signoff in typical corner)
→30% energy reduction

>2x leakage reduction @ 70°C

A. Di Mauro, et. al., "Temperature and process-aware performance monitoring and compensation for an ULP multi-core cluster in 28nm UTBB FD-SOI technology," PATMOS, 2017.

The Evolution of the 'Species'

	PULPv1	PULPv2	PULPv3
# of cores	4	4	4
L2 memory	16 kB	64 kB	128 kB
TCDM	16kB SRAM	32kB SRAM 8kB SCM	32kB SRAM 16kB SCM
DVFS	no	yes	yes
I$	4kB SRAM private	4kB SCM private	4kB SCM shared
DSP Extensions	no	no	yes
HW Synchronizer	no	no	yes

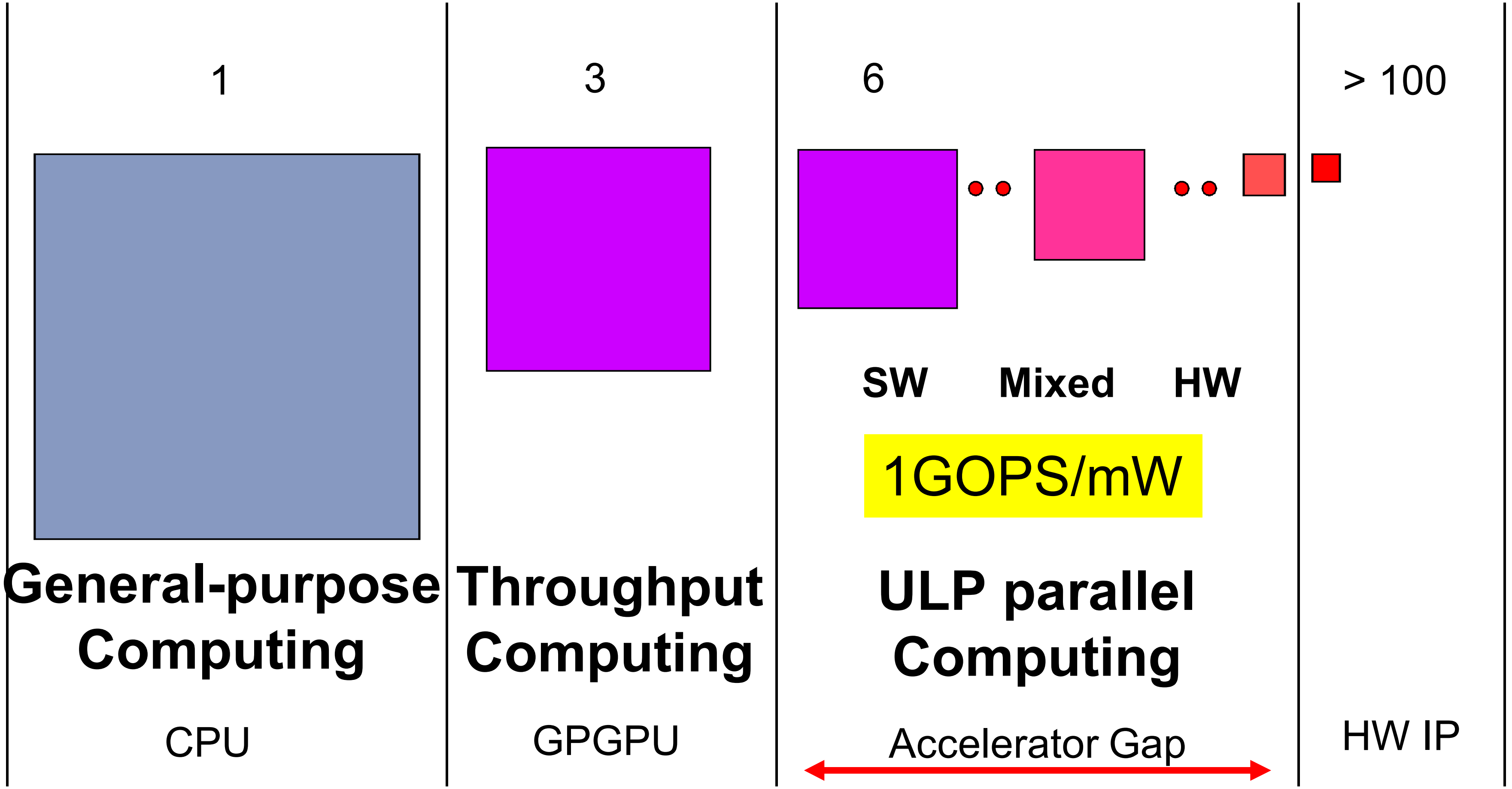
	PULPv1	PULPv2	PULPv3
Status	silicon proven	Silicon proven	post tape out
Technology	FD-SOI 28nm conventional-well	FD-SOI 28nm flip-well	FD-SOI 28nm conventional-well
Voltage range	0.45V - 1.2V	0.3V - 1.2V	0.5V - 0.7V
BB range	-1.8V - 0.9V	0.0V - 1.8V	-1.8V - 0.9V
Max freq.	475 MHz	1 GHz	200 MHz
Max perf.	1.9 GOPS	4 GOPS	1.8 GOPS
Peak en. eff.	60 GOPS/W	135 GOPS/W	385 GOPS/W

PULP

Recovering Efficiency Through Flexible Customization

Recovering More Silicon Efficiency

GOPS/W



1

3

6

> 100

SW **Mixed** **HW**

1GOPS/mW

General-purpose Computing

Throughput Computing

ULP parallel Computing

CPU

GPGPU

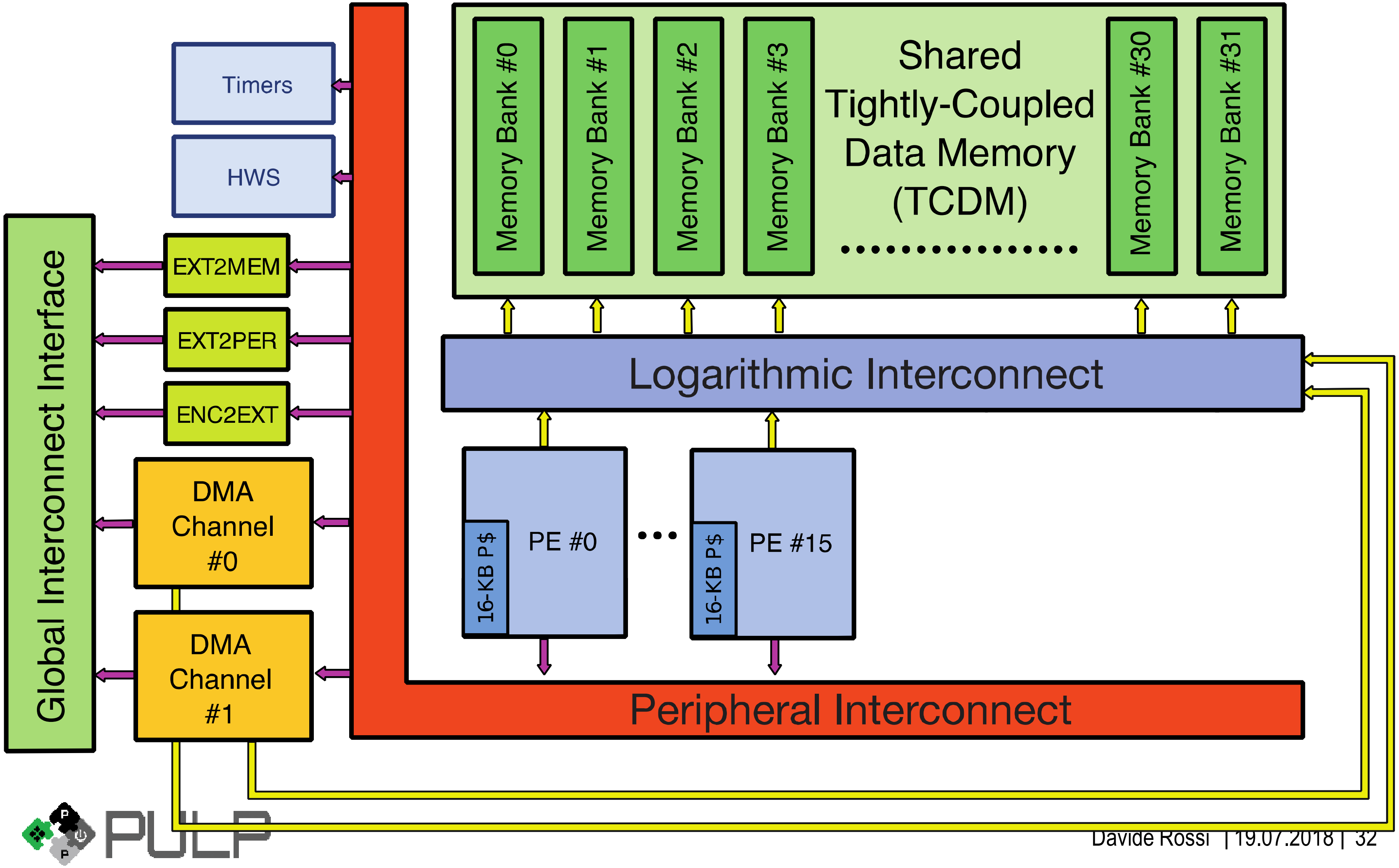
Accelerator Gap

HW IP

Closing The Accelerator Efficiency Gap with Agile Customization

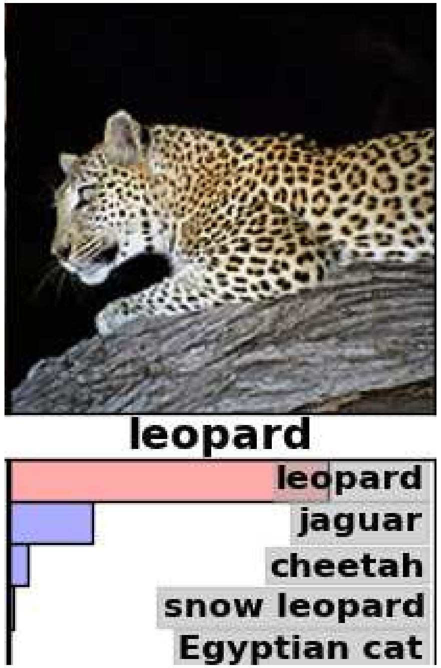
Recovering Even More Efficiency

Fixed function accelerators have limited reuse… how to limit proliferation?



Learn to Accelerate

- Brain-inspired (**deep convolutional networks**) systems are high performers in many tasks over *many domains*



Human:
85% (untrained),
94.9% (trained)

CNN:
93.4% accuracy

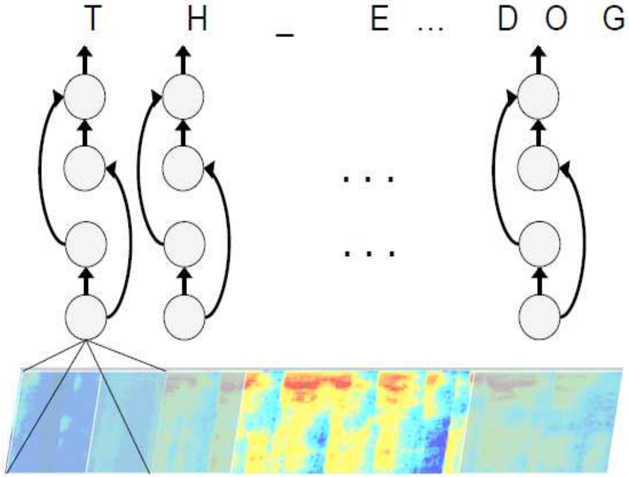


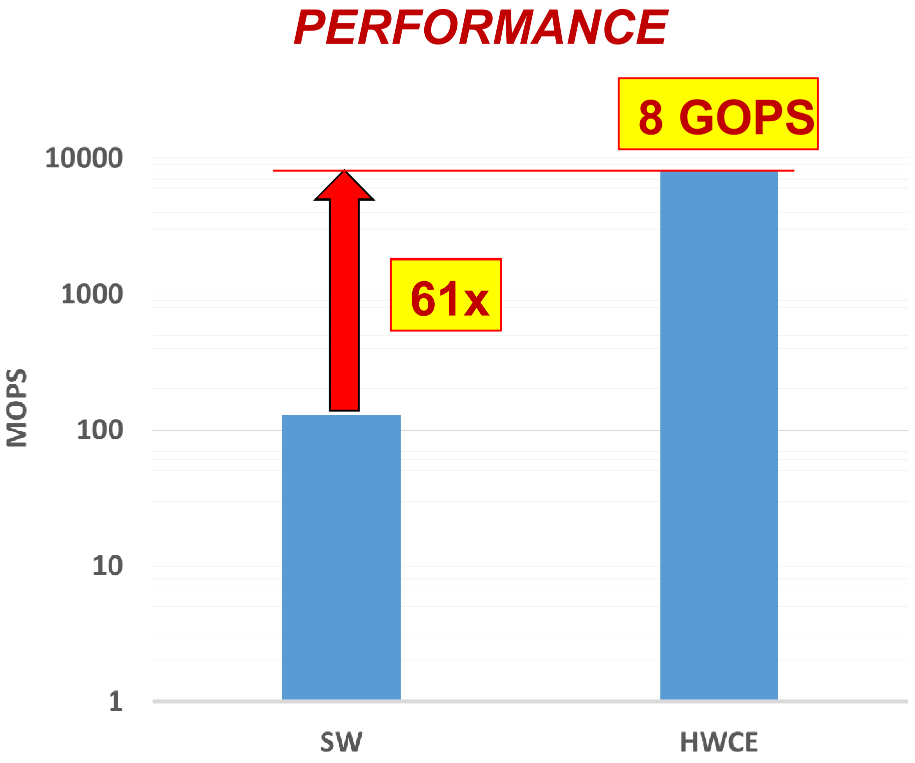
Image recognition
[RussakovskyIMAGENET2014]

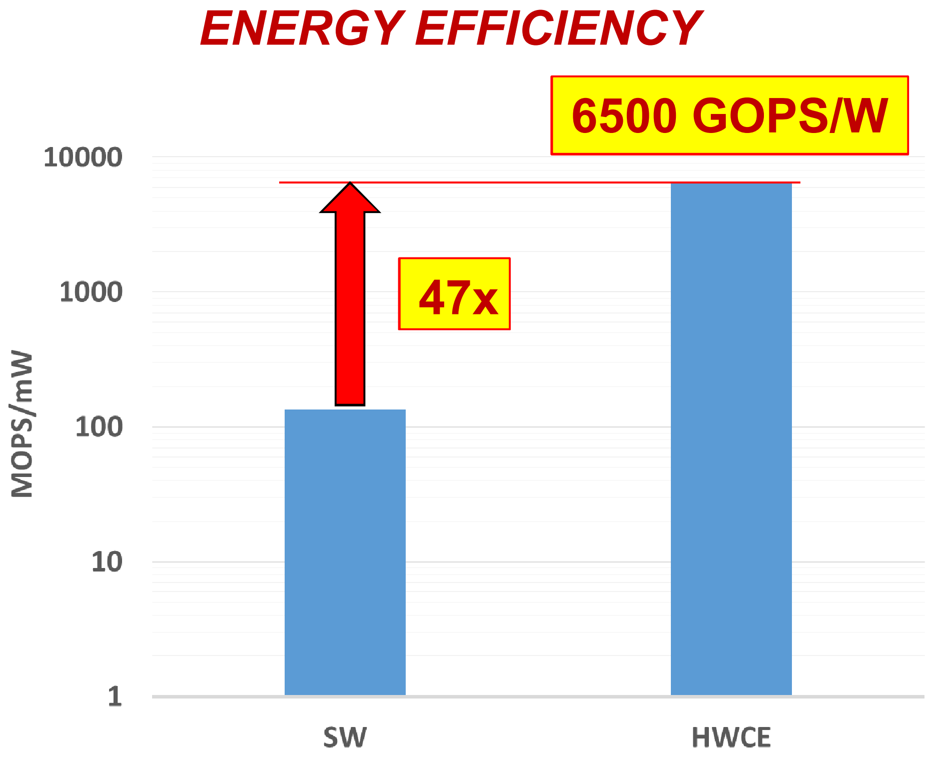
Speech recognition
[HannunARXIV2014]

Flexible acceleration: learned CNN weights are “the program”

PULP CNN Performance

Average performance and energy efficiency on a 32x16 CNN frame





PULPv3 ARCHITECTURE, CORNER: tt28, 25°C, VDD= 0.5V, FBB = 0.5V

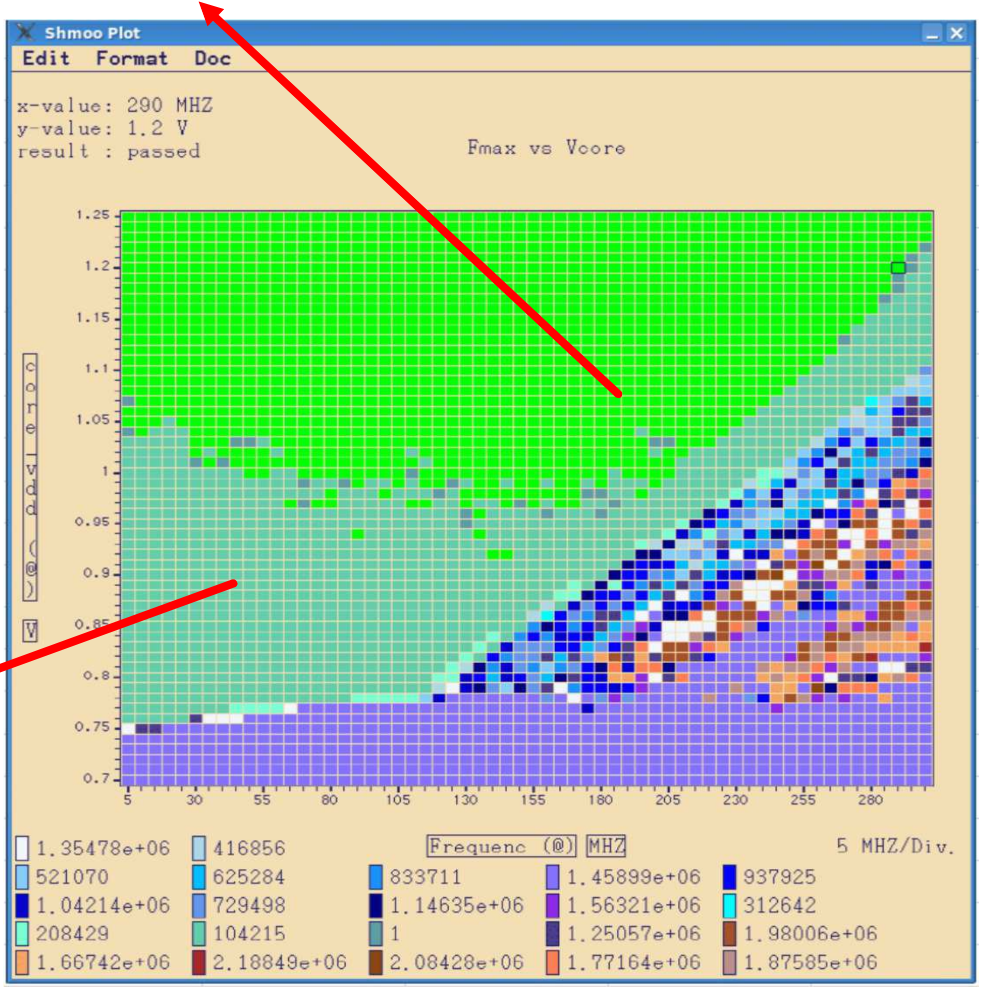
Sub pJ/OP? Approximate Computing

437 GOPS/W @1.2V

803 GOPS/W @0.8V

CNNs are intrinsically resilient to (a small amount) of soft errors

0% bit flips



x-value: 290 MHZ
y-value: 1.2 V
result : passed

1% bit flips

1.84x energy improvement

1.2pJ/OP

From Approximate to Transprecision: YodaNN[1]

- Approximation at the algorithmic side →Binary weights
- BinaryConnect [Courbariaux, NIPS15]
  - Reduce weights from 12-bit to a binary value -1/+1
  - Stochastic Gradient Descent with Binarization in the Forward Path

$$w_{b,stoch} = \begin{cases} -1 & p_{-1} = \sigma(w) \\ 1 & p_1 = 1 - p_{-1} \end{cases} \qquad w_{b,det} = \begin{cases} -1 & w < 0 \\ 1 & w > 0 \end{cases}$$

  - Learning large networks is still an issue with binary connect…

- Ultra-optimized HW is possible!
  - Power reduction because of arithmetic simplification (multipliers →Two's complement + muxes)
  - Major arithmetic density improvements
    - **Area can be used for more energy-efficient weight storage**
  - SCM memories for lower voltage → E goes with 1/V^2

[1] After the Yedi Master from Star Wars - "Small in size but wise and powerful" cit. www.starwars.com

Comparison sith SoA

Publication	Throughput [GOPS]	En.Eff. [GOPS/W]	Supply [V]	Area Effic. [GOPS/MGE]
Neuflow	320	490	1.0	17
Leuven	102	2600	0.5 - 1.1	64
Eyeriss	84	160	0.8 - 1.2	46
NINEX	569	1800	1.2	51
k-Brain	411	1930	1.2	109
Origami	**196**/74	437/**803**	1.2/0.8	**90**/34
This Work*	**1510**/55	9800/**61200**	1.2/0.6	**1135**/41

2.7x

23x

10x

***UMC 65nm Technology, post place & route, 25°C, tt, 1.2V/0.6V**

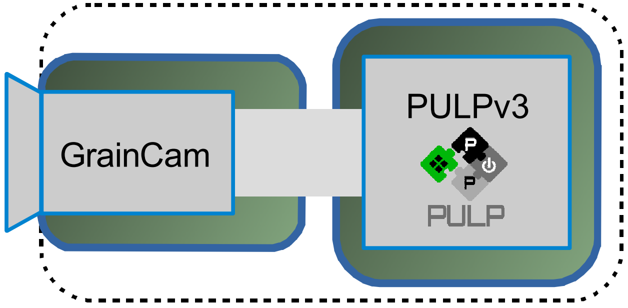
Breakthrough for ultra-low-power CNN ASIC implementation

fJ/op in sight: manufacturing in Globalfoundries 22FDX

From Frame-based to Event-based

Back to System-Level

Smart Visual Sensor→ idle most of the time (nothing interesting to see)



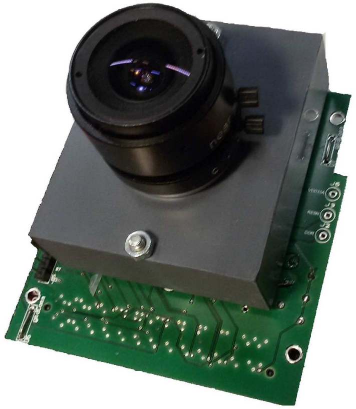
Mixed-Signal Event-based Imager

Digital Parallel Processor

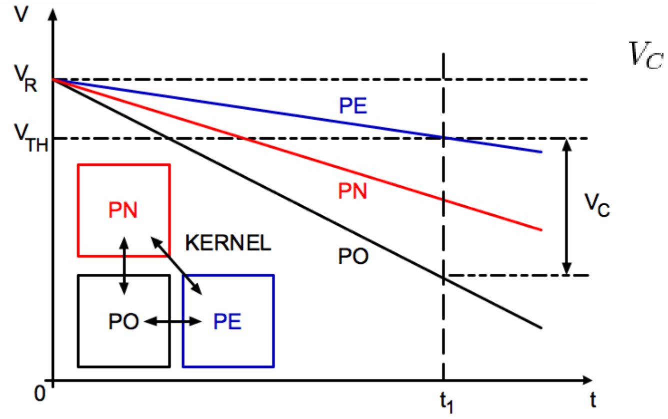
- **Event-Driven Computation**, which occurs only when relevant events are detected by the sensor
- **Event-based sensor interface** to minimize IO energy (vs. Frame-based inteface
- **Mixed-signal event triggering** with an ULP imager with internal processing AMS capability

A Neuromorphic Approach for doing *nothing* VERY well

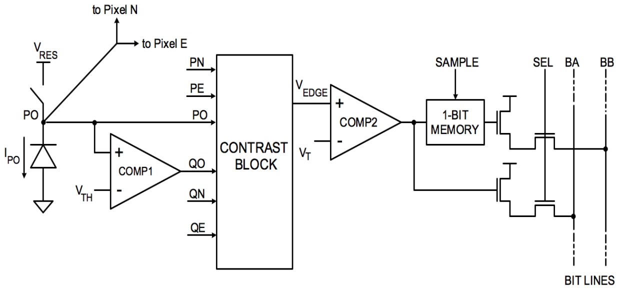
GrainCam Imager (FBK)



Pixel-level spatial-contrast extraction

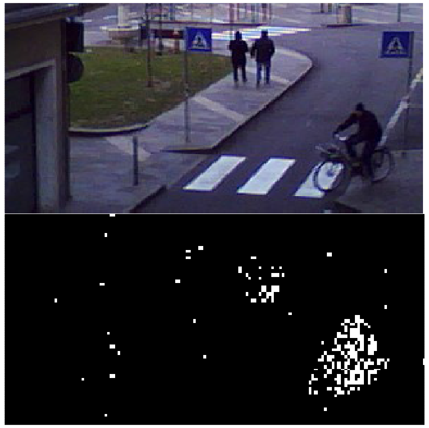


$$V_C = V_{PE}(t_1) - V_{PO}(t_1) = (V_R - V_{TH}) \left(\frac{I_{PO} - I_{PE}}{I_{PE}} \right)$$



Analog internal image processing

- Contrast Extraction
- Motion Extraction, differencing two successive frames
- Background Subtraction, differencing the reference image, stored in the memory, with the current frame

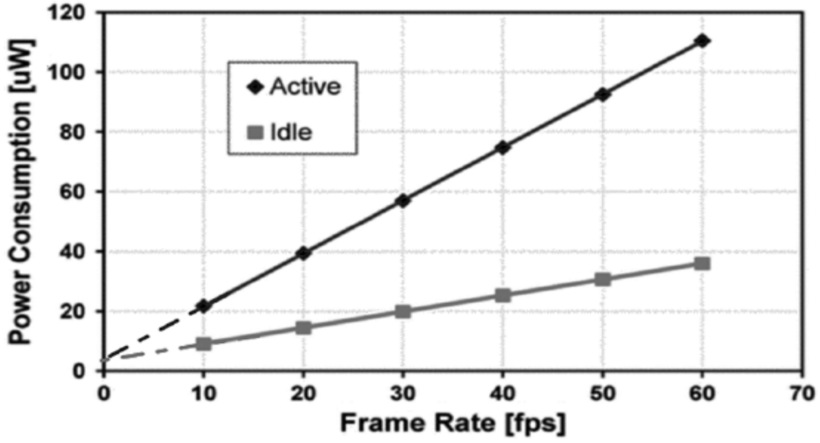


GrainCam Readout

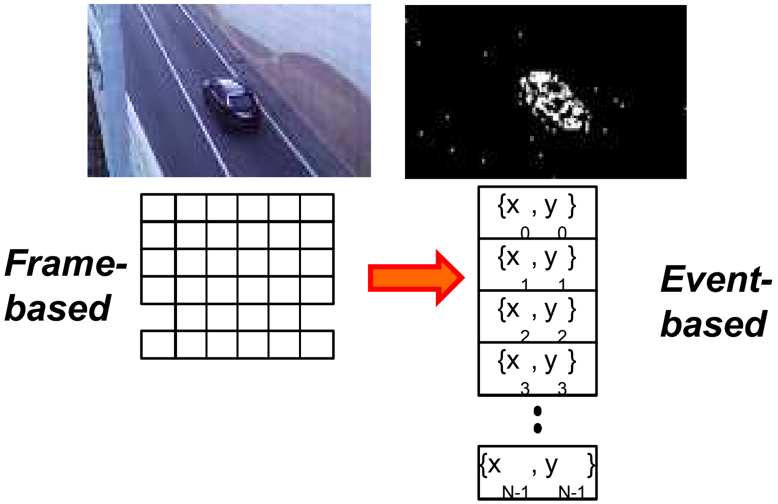
Readout modes:

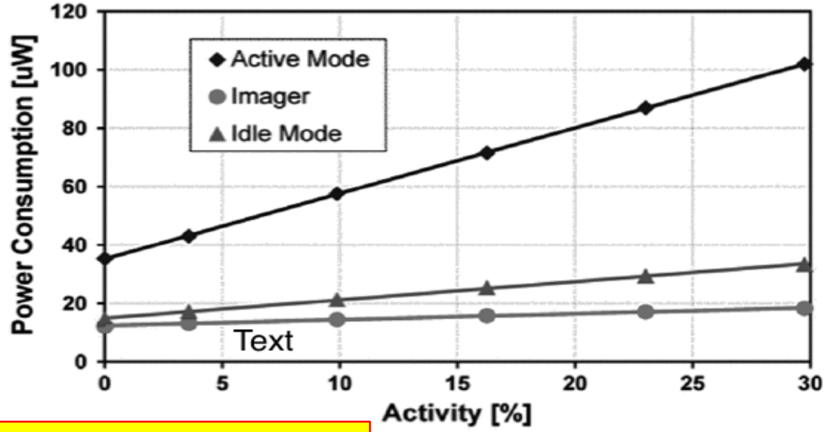
- IDLE: readout the counter of asserted pixels
- ACTIVE: sending out the addresses of asserted pixels (address-coded representation), according raster scan order

Event-based sensing: output frame data bandwidth depends on the external context-activity



(a)

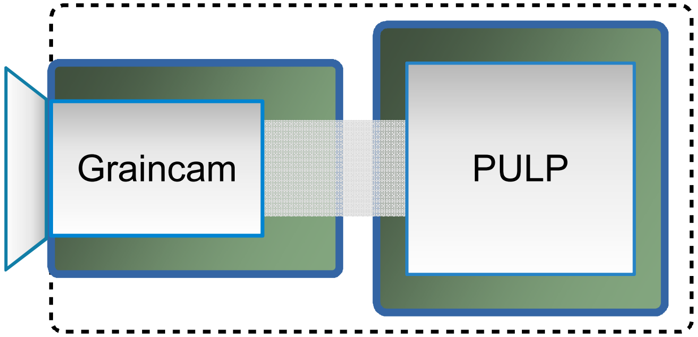


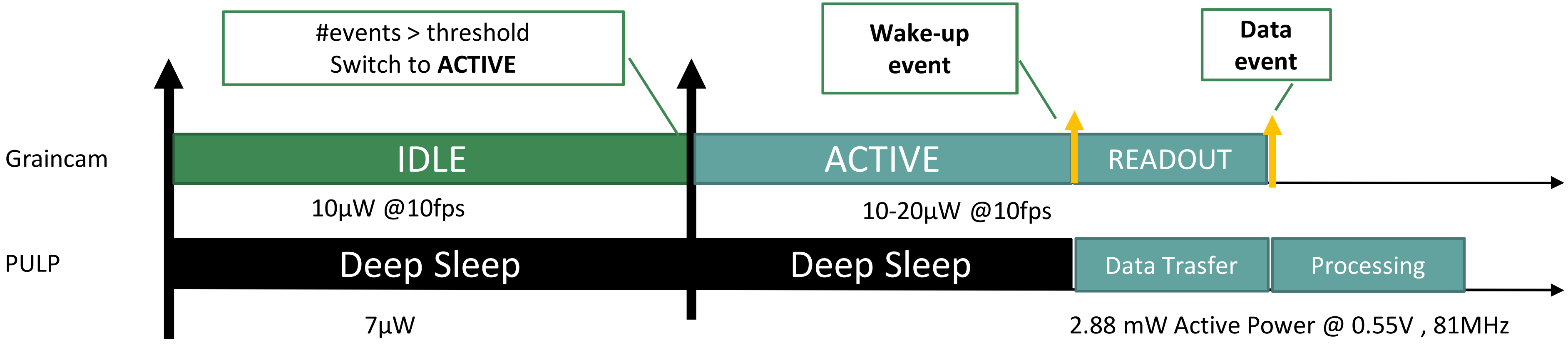


(b)

Ultra Low Power Consumption e.g. 10-20uW @10fps

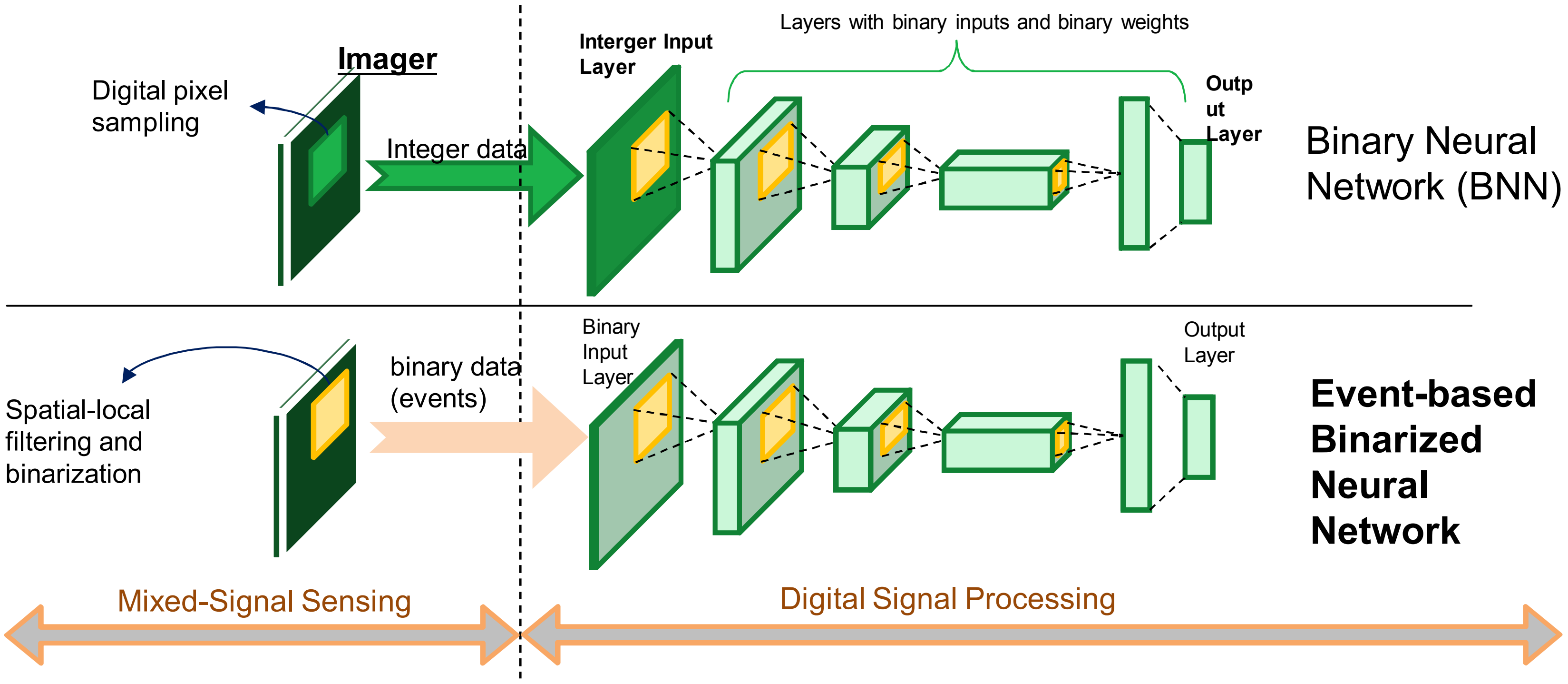
Power Management



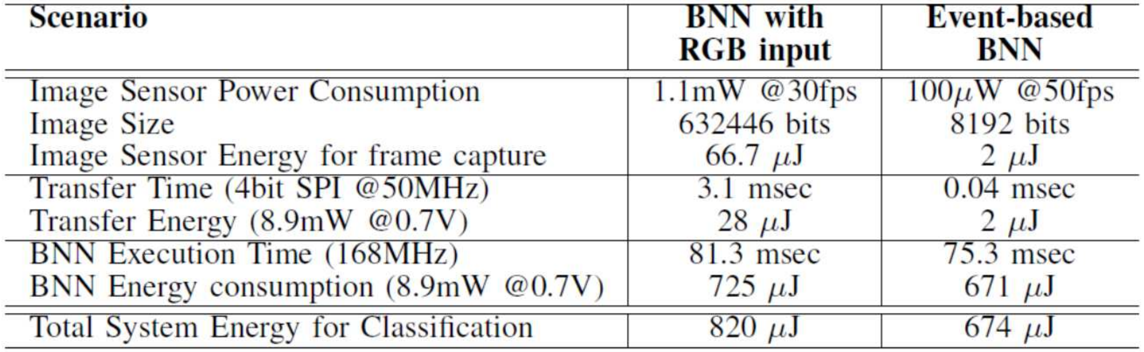


M. Rusci, et. al., "A Sub-mW IoT-Endnode for Always-On Visual Monitoring and Smart Triggering," in IEEE IoT Journal, 2017.

Event-Driven Binary Deep Network

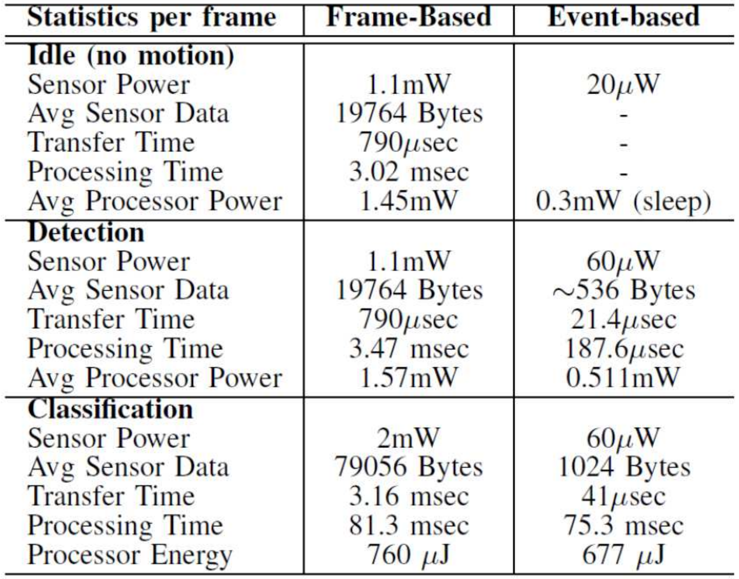

Digital pixel sampling
Imager
Integer data
Interger Input Layer
Layers with binary inputs and binary weights
Outp ut Layer
Binary Neural Network (BNN)
Spatial-local filtering and binarization
binary data (events)
Binary Input Layer
Output Layer
Event-based Binarized Neural Network
Mixed-Signal Sensing
Digital Signal Processing

Results

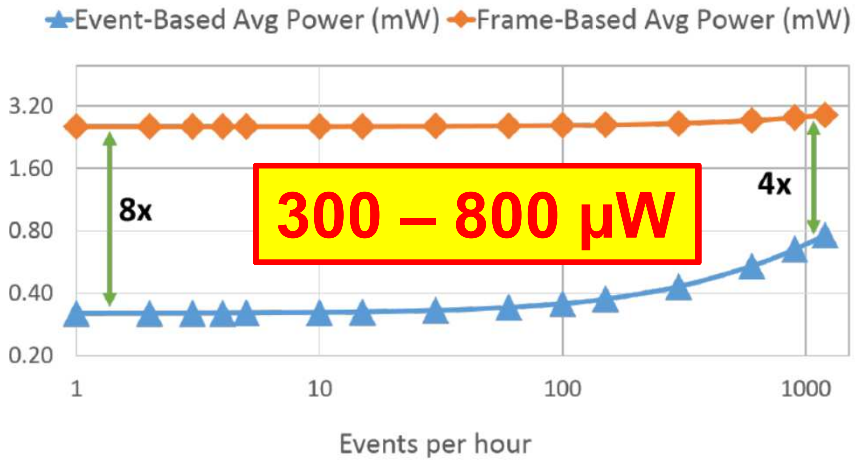


Scenario	BNN with RGB input	Event-based BNN
Image Sensor Power Consumption	1.1mW @30fps	100μW @50fps
Image Size	632446 bits	8192 bits
Image Sensor Energy for frame capture	66.7 μJ	2 μJ
Transfer Time (4bit SPI @50MHz)	3.1 msec	0.04 msec
Transfer Energy (8.9mW @0.7V)	28 μJ	2 μJ
BNN Execution Time (168MHz)	81.3 msec	75.3 msec
BNN Energy consumption (8.9mW @0.7V)	725 μJ	671 μJ
Total System Energy for Classification	820 μJ	674 μJ

- 3 Classes:
  - Pedestrians
  - Bikes
  - Cars
- 84.6% vs. 81.6% Accuracy



Statistics per frame	Frame-Based	Event-based
Idle (no motion)		
Sensor Power	1.1mW	20μW
Avg Sensor Data	19764 Bytes	-
Transfer Time	790μsec	-
Processing Time	3.02 msec	-
Avg Processor Power	1.45mW	0.3mW (sleep)
Detection		
Sensor Power	1.1mW	60μW
Avg Sensor Data	19764 Bytes	~536 Bytes
Transfer Time	790μsec	21.4μsec
Processing Time	3.47 msec	187.6μsec
Avg Processor Power	1.57mW	0.511mW
Classification		
Sensor Power	2mW	60μW
Avg Sensor Data	79056 Bytes	1024 Bytes
Transfer Time	3.16 msec	41μsec
Processing Time	81.3 msec	75.3 msec
Processor Energy	760 μJ	677 μJ



300 – 800 µW

Outlook and Conclusion

Conclusion

- Near-sensor processing → Energy efficiency requirements: pJ/OP and below
  - Technology scaling alone is not doing the job for us
  - Ultra-low power architecture and circuits are needed
- CNNs-based visual functions can be squeezed into mW envelope
  - Non-von-Neumann acceleration
  - Very robust to trans-precision computations (deterministic and statistical)
  - fJ/OP is in sight!
- More than CNN is needed (e.g. linear algebra, online optimizazion)
- Open Source HW & SW approach →innovation ecosystem

Thanks for your attention!!!

Parallel Ultra Low Power

www.pulp-platform.org
www-micrel.deis.unibo.it/pulp-project
iis-projects.ee.ethz.ch/index.php/PULP

The fun is just at the beginning...

A Very Short Review on CMOS power (and power minimization)

Summary of Power Dissipation Sources

$$P \sim \alpha \cdot \left(C_L\right) \cdot V_{swing} \cdot V_{DD} \cdot f + \left(I_{DC} + I_{Leak}\right) \cdot V_{DD}$$

- α – switching activity
- C_L – load capacitance
- V_{swing} – voltage swing
- f – frequency

- I_{DC} – static current
- I_{leak} – leakage current

$$\boxed{P = \frac{energy}{operation} \times rate + static\ \ power}$$

The Traditional Design Philosophy

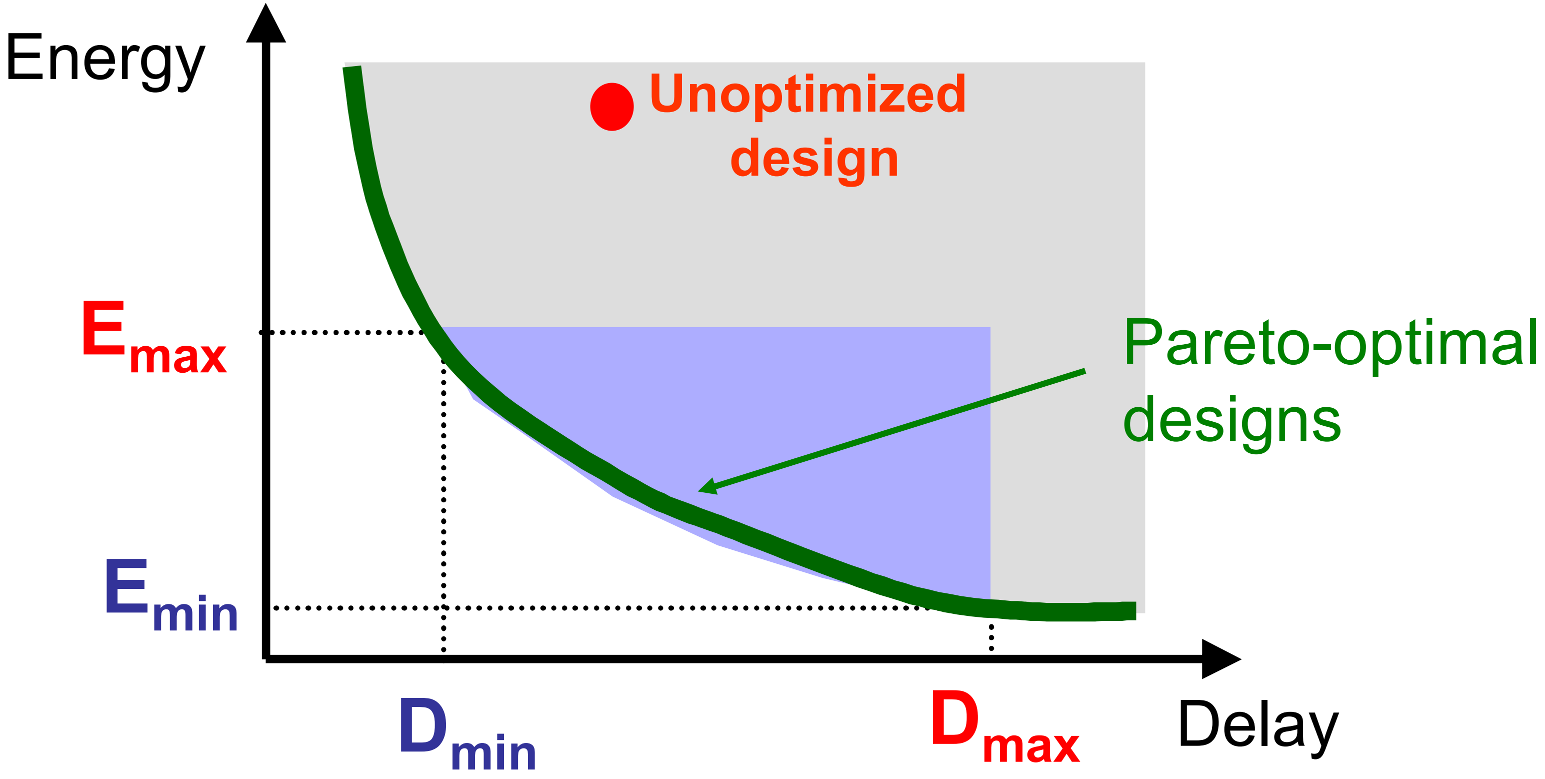
- Maximum performance is primary goal
  - Minimum delay at circuit level
- Architecture implements the required function with target throughput, latency
- Performance achieved through optimum sizing, logic mapping, architectural transformations.
- Supplies, thresholds set to achieve maximum performance, subject to reliability constraints

The New Design Philosophy

- Maximum performance (in terms of propagation delay) is too power-hungry, and/or not even practically achievable
- Many (if not most) applications either can tolerate larger latency, or can live with lower than maximum clock-speeds
- Excess performance (as offered by technology) to be used for energy/power reduction

Trading off speed for power

Exploring the Energy-Delay Space



In energy-constrained world, design is trade-off process

- ♦ Minimize energy for a given performance requirement
- ♦ Maximize performance for given energy budget

Reducing power @ all design levels

- Algorithmic level
- Compiler level
- Architecture level
- Micro-Architecture
- Circuit level
- Silicon level

- Important concepts:
  - Lower Vdd and freq. (even if errors occur) / dynamically adapt Vdd and freq.
  - Reduce circuit
  - Exploit locality
  - Reduce switching activity, glitches, etc.

Algorithmic level

- The best indicator for energy is …..
  …. the number of cycles

- Try alternative algorithms with lower complexity
  - E.g. quick-sort, O(n log n) ⇔ bubble-sort, O (n^2)
  - … but be aware of the 'constant' : O(n log n) ⇒ c*(n log n)

- Heuristic approach
  - Go for a good solution, not the best !!

Biggest gains at this level !!

Compiler level

- Source-to-Source transformations
  - loop trafo's to improve locality
- Strength reduction
  - E.g. replace Const * A with Add's and Shift's
  - Replace Floating point with Fixed point
- Reduce register pressure / number of accesses to register file
  - Use software bypassing
- Scenarios: current workloads are highly dynamic
  - Determine and predict execution modes
  - Group execution modes into scenarios
  - Perform special optimizations per scenario
    - DFVS: Dynamic Voltage and Frequency Scaling
    - More advanced loop optimizations
- Reorder instructions to reduce bit-transistions

Architecture level

- Going parallel
- Going heterogeneous
  - tune your architecture, exploit SFUs (special function units)
  - trade-off between flexibility / programmability / genericity and efficiency
- Add local memories
  - prefer scratchpad i.s.o. cache
- Cluster FUs and register files (see next slide)
- Reduce bit-width
  - sub-word parallelism (SIMD)

Organization (micro-arch.) level

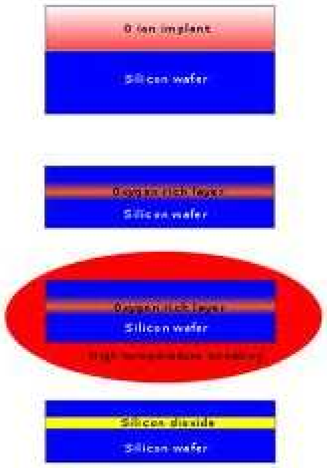
- Enabling Vdd reduction
  - Pipelining
    - cheap way of parallelism
  - Enabling lower freq. ⇒ lower V_{dd}
  - Note 1: don't pipeline if you don't need the performance
  - Note 2: don't exaggerate (like the 31-stage Pentium 4)
- Reduce register traffic
  - avoid unnecessary reads and write
  - make bypass registers visible

Circuit level

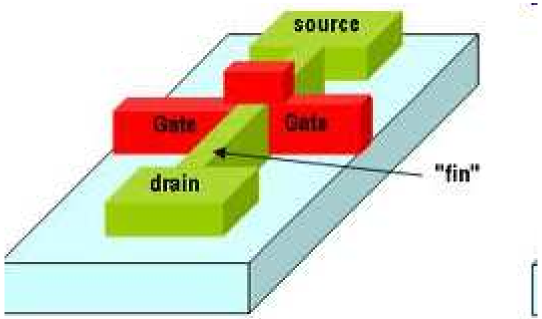
- Clock gating
- Power gating
- Multiple Vdd modes
- Reduce glitches: balancing digital path's
- Exploit Zeros
- Special SRAM cells
  - normal SRAM can not scale below Vdd = 0.7 - 0.8 Volt
- Razor method; replay
- Allow errors and add redundancy to architectural invisible structures
  - branch predictor
  - caches
- .. and many more ..

Silicon level

- Higher V_t (V_threshold)
- Back Biasing control
  - see thesis Maurice Meijer (2011)
- SOI (Silicon on Insulator)
  - silicon junction is above an electr. insulator (silicon dioxide)
  - lowers parasitic device capacitance



- Better transistors: Finfet
  - multi-gate
  - reduce leakage (off-state curent)



- .. and many more