Heterogeneous architectures and accelerators

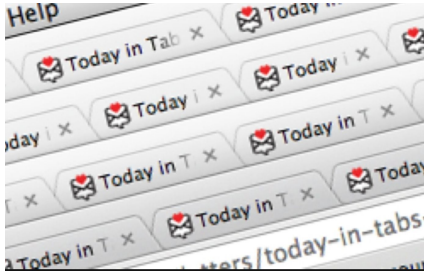
Marcelo Novaes e Rodrigo Oliveira

DCC819: Arquitetura de Computadores 2017.2

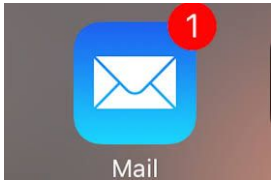
Motivation


EA SPORTS
FIFA 18
FIFA
OFFICIAL LICENSED PRODUCT
GET IT ON Google Play
Download on the App Store


SAMSUNG
SELECT
START


Help
Today in Tab
Today
Today in T
Today in T
Today in T
Today in T
Today in T
ters/today-in-tabs

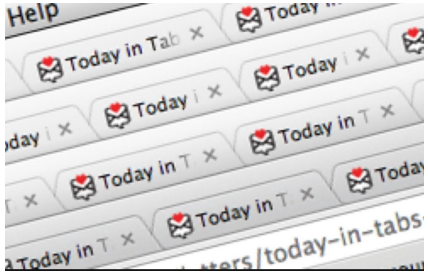

SMS

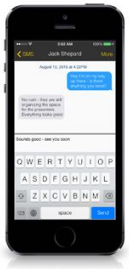

1
Mail

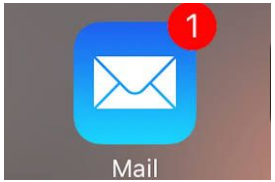
Motivation

Motivation

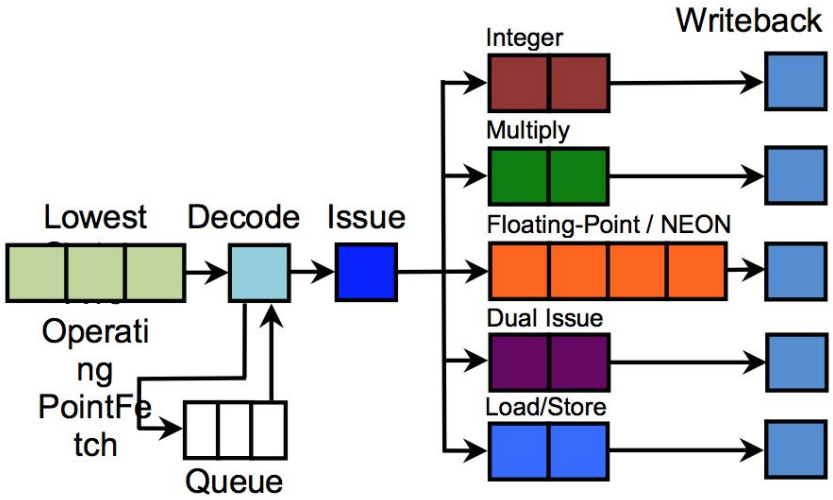
- Our use of mobile applications is heterogeneous.
  - At least two categories: High intensity tasks and Low intensity tasks.



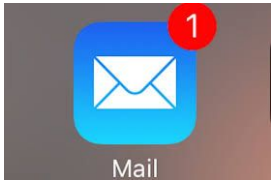




Motivation


Writeback
Integer
Multiply
Lowest
Decode
Issue
Floating-Point / NEON
Dual Issue
Operati
ng
PointFe
tch
Load/Store
Queue

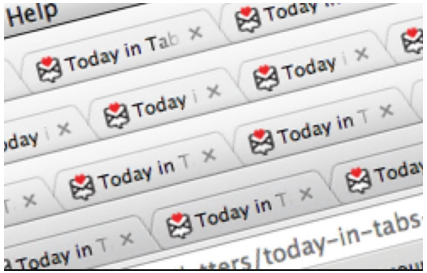

SMS

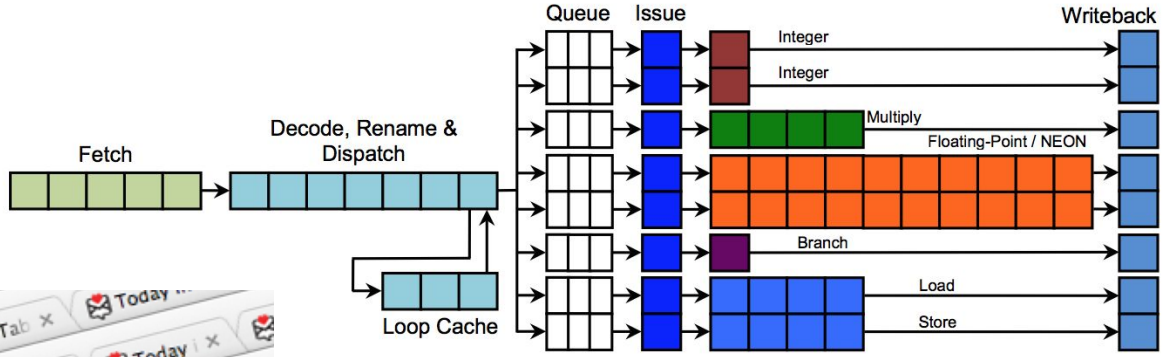

1
Mail

Motivation

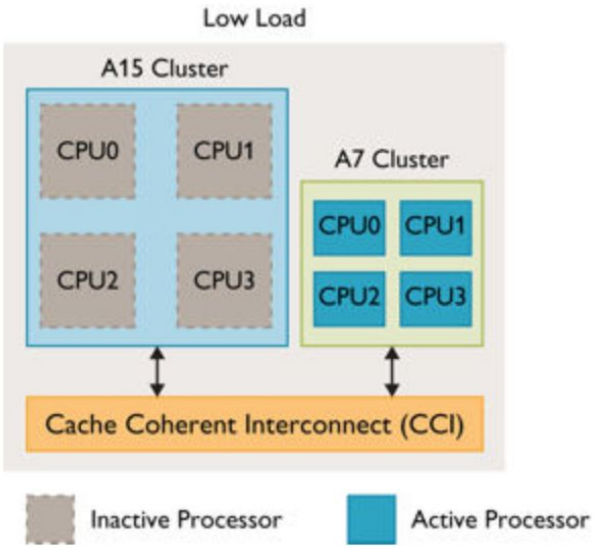

EA SPORTS
FIFA18
FIFA
OFFICIAL LICENSED PRODUCT
GET IT ON Google Play
Download on the App Store


SAMSUNG
SELECT
START

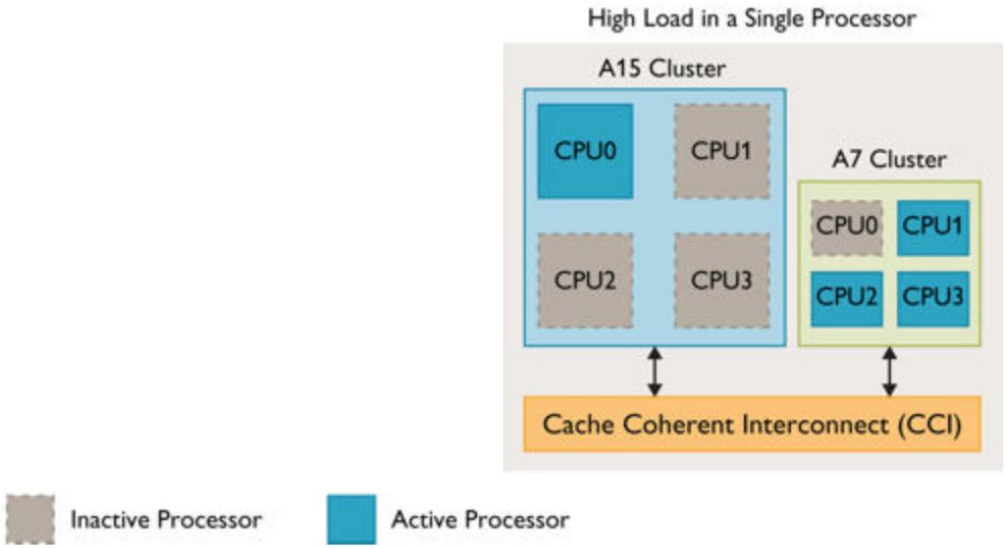

Help
Today in Tab
Today
Today in T
Today in T
Today in T
Today in T
Today
tters/today-in-tabs


Fetch
Decode, Rename & Dispatch
Loop Cache
Queue
Issue
Writeback
Integer
Integer
Multiply
Floating-Point / NEON
Branch
Load
Store

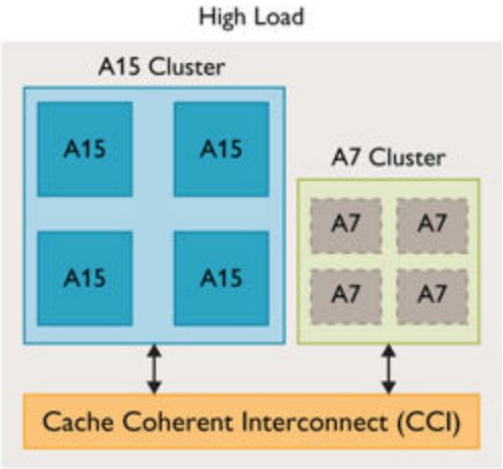
Modern Heterogêneous Multicore processors



Modern Heterogêneous Multicore processors



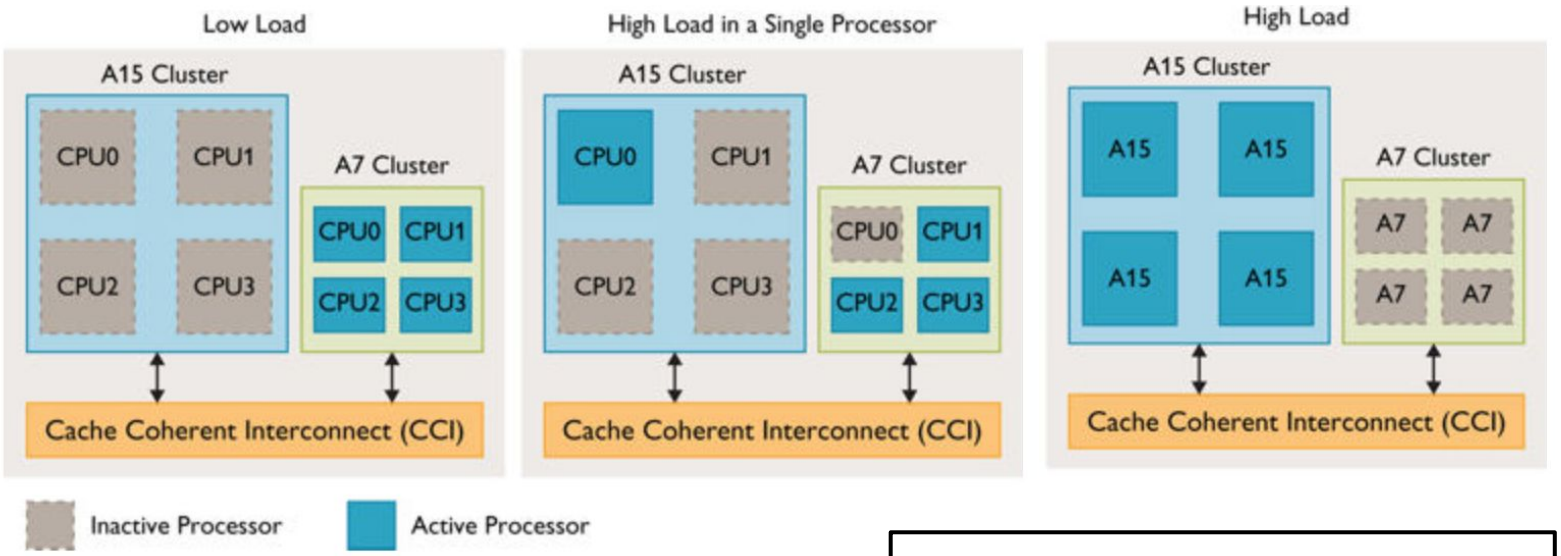
Modern Heterogêneous Multicore processors



Inactive Processor

Active Processor

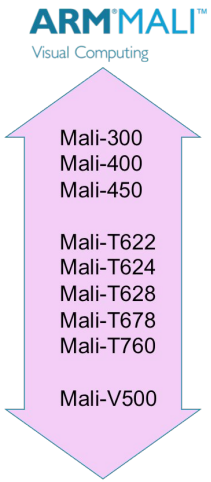
Modern Heterogêneous Multicore processors

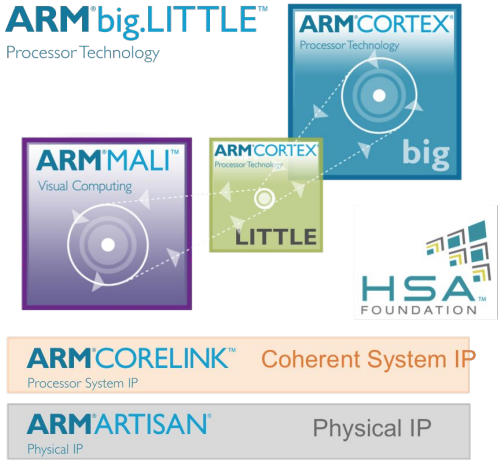


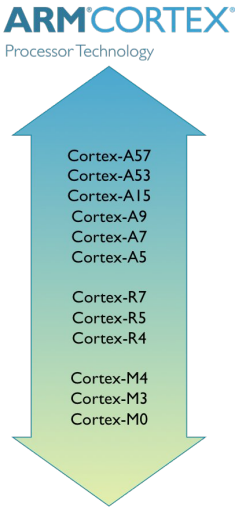
Single-ISA heterogeneous multicore

Existing software can be **transparently migrated** one from another.

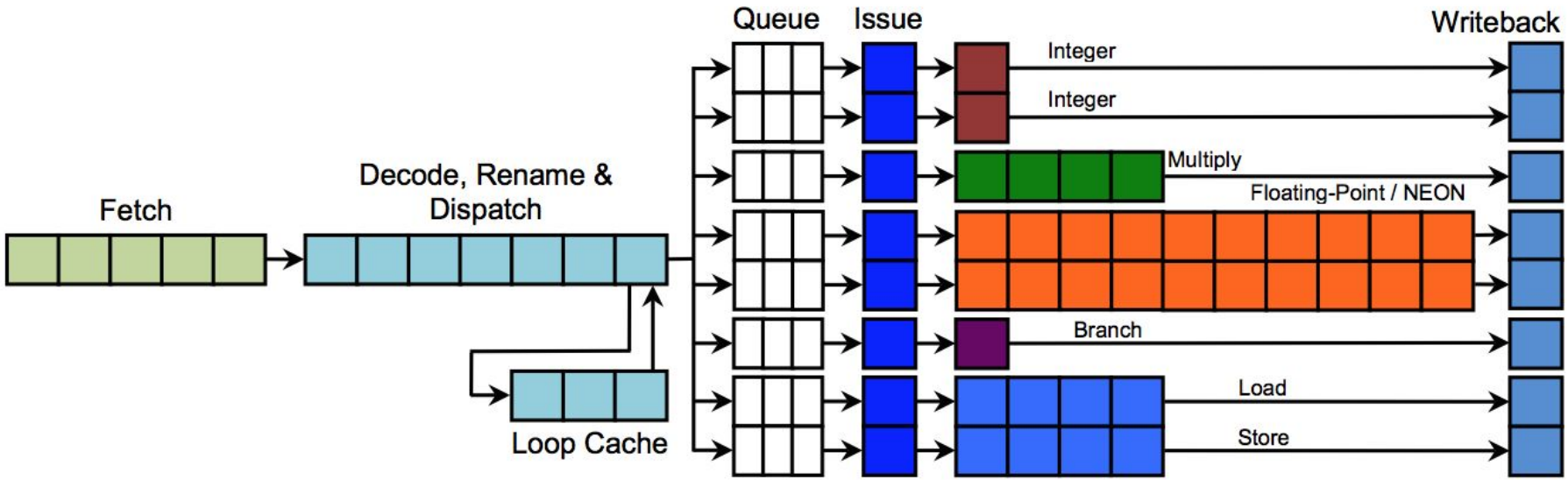
Big.LITTLE Architectures



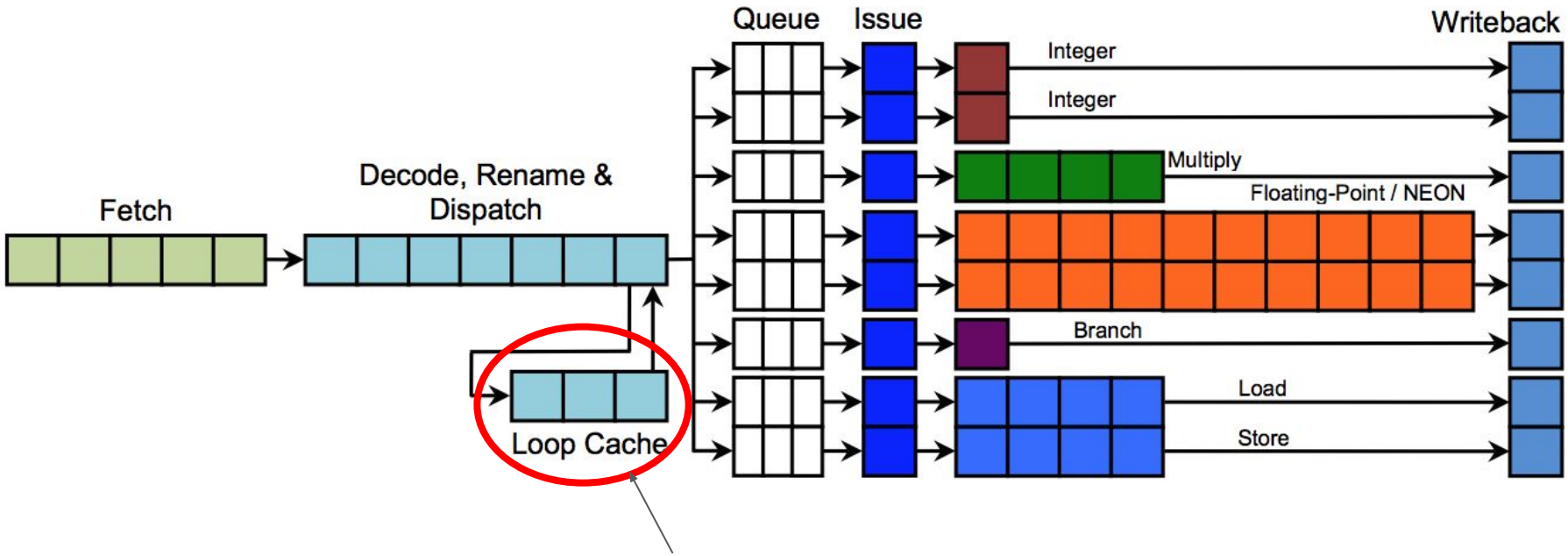




ARM Cortex-A15 (Big)

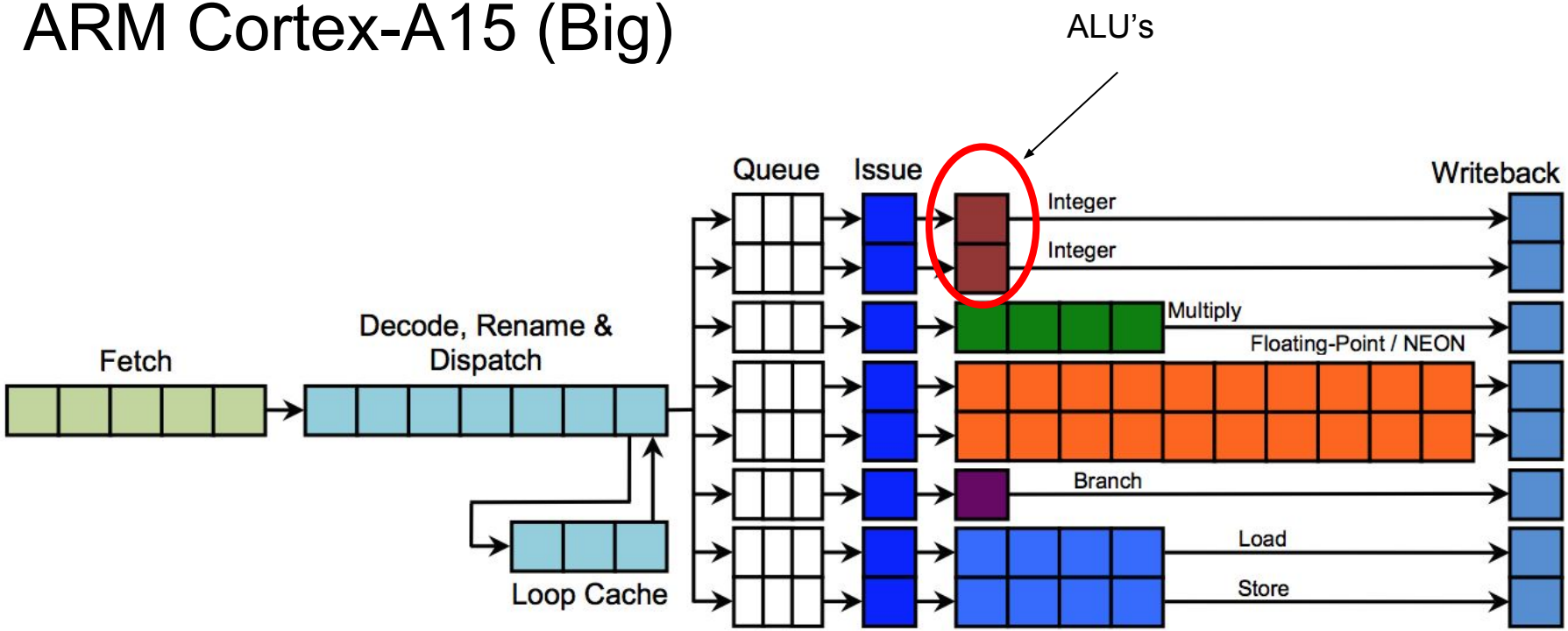


ARM Cortex-A15 (Big)

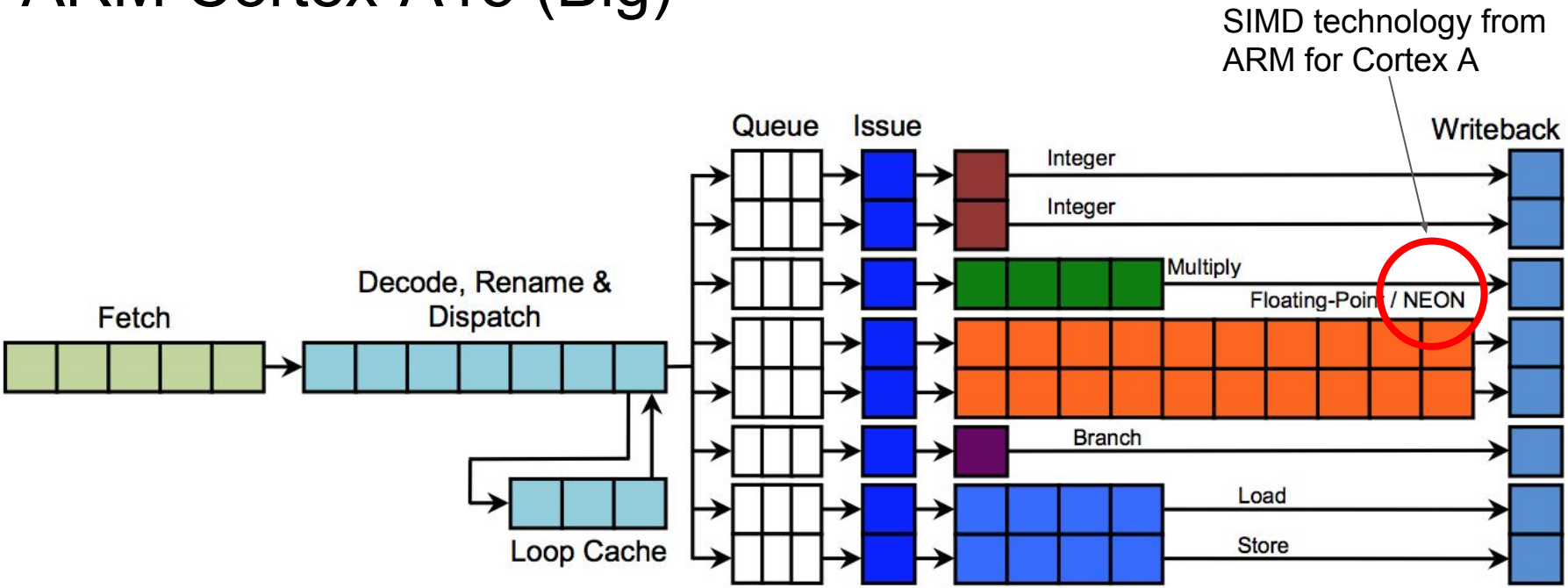


It holds instructions of small loops in order to do a Fetch faster than doing from L1 cache.

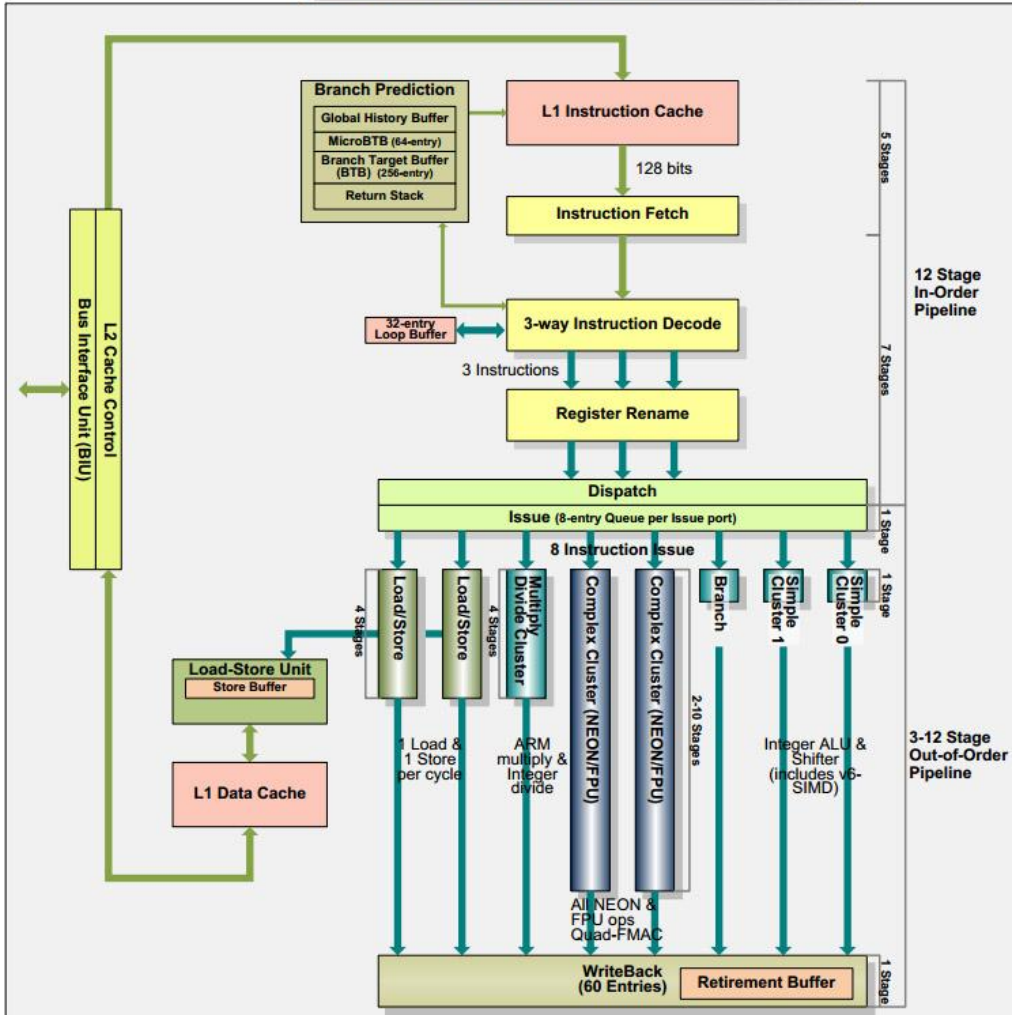
ARM Cortex-A15 (Big)



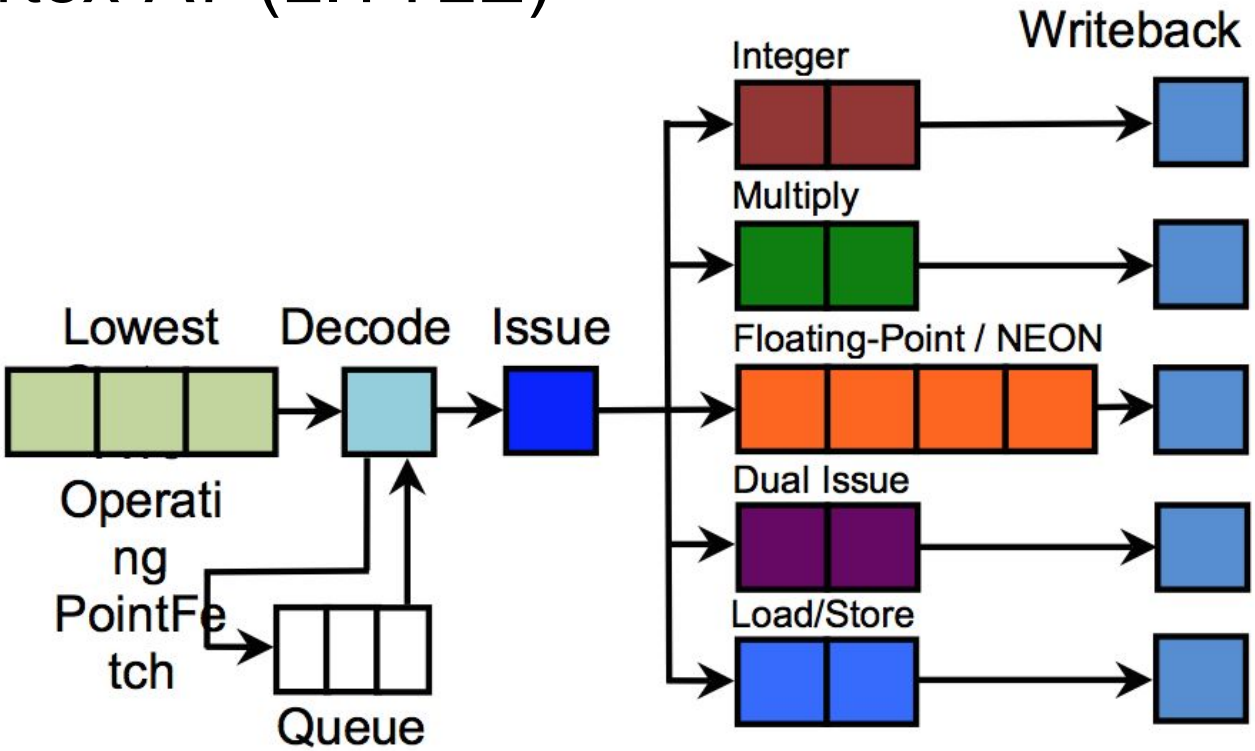
ARM Cortex-A15 (Big)



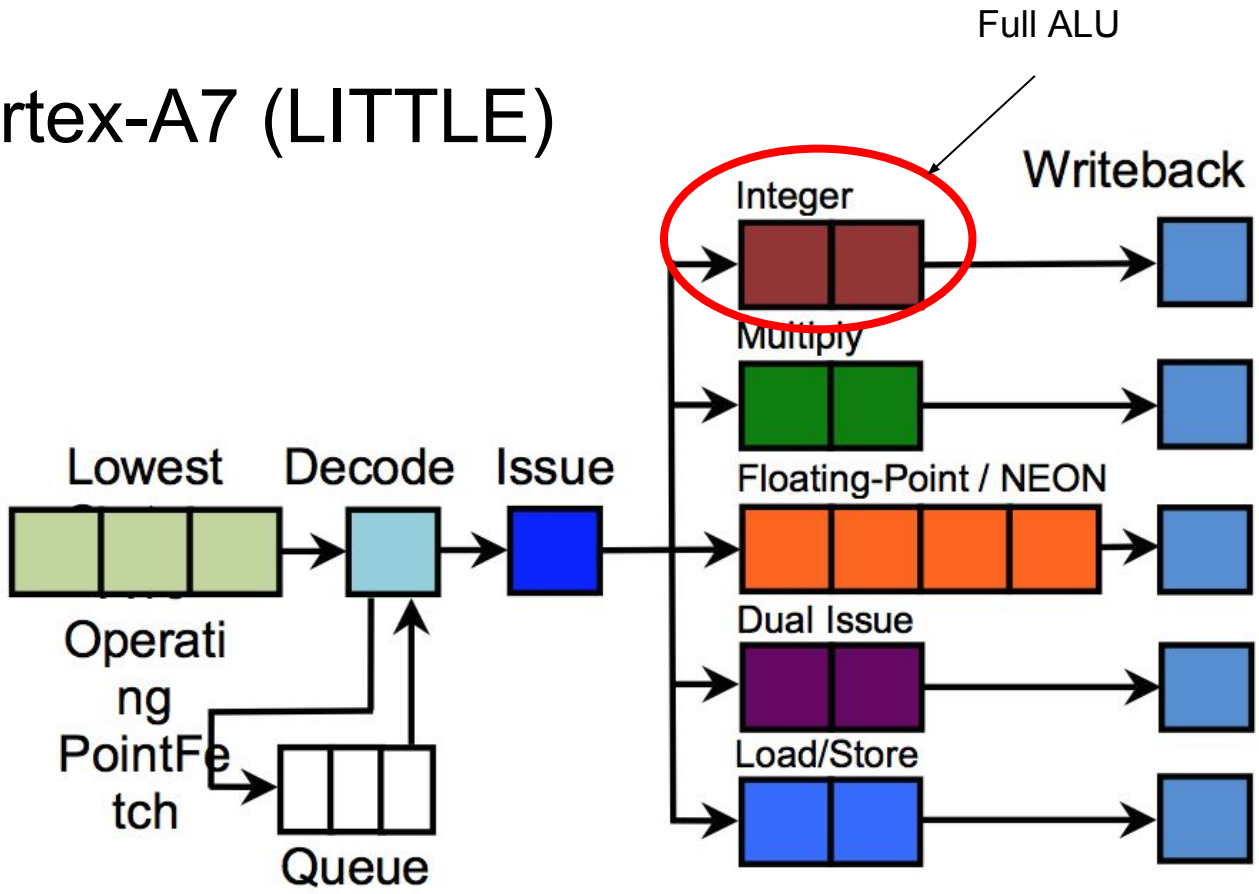
ARM Cortex-A15 Block Diagram



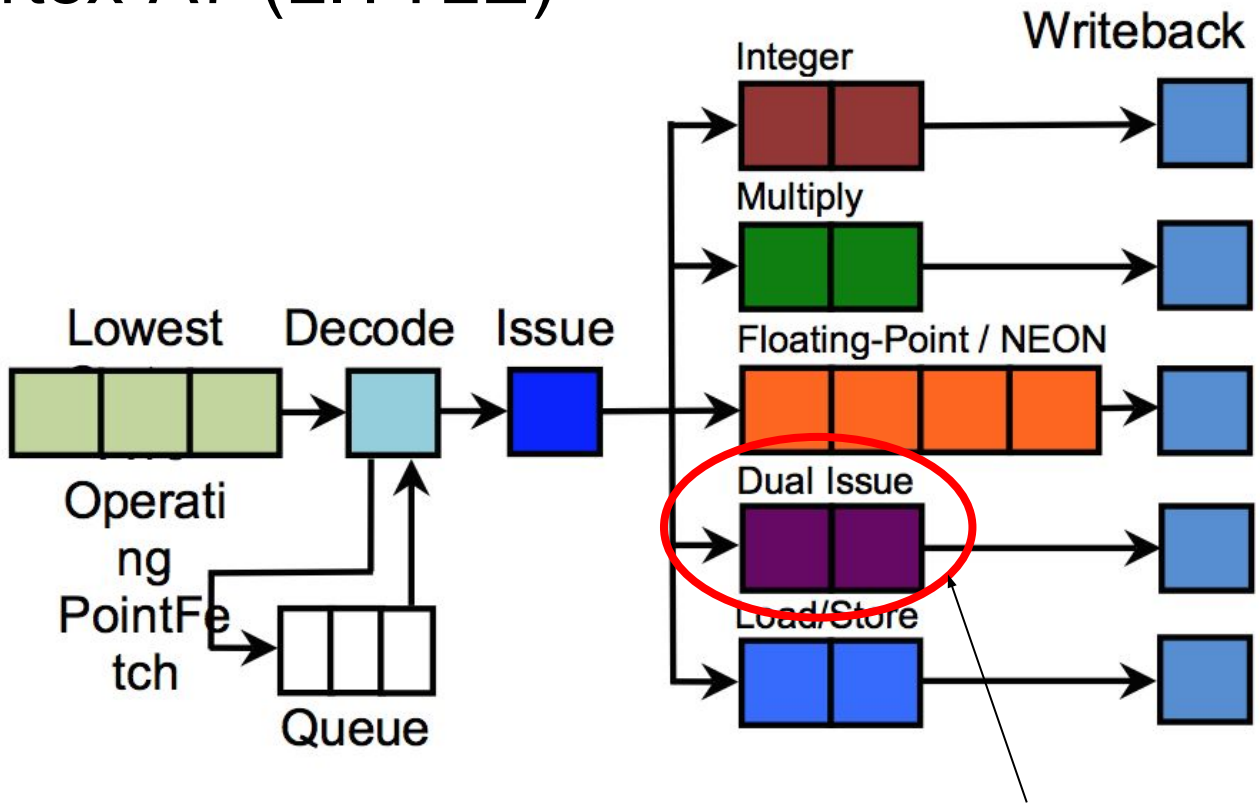
ARM Cortex-A7 (LITTLE)



ARM Cortex-A7 (LITTLE)



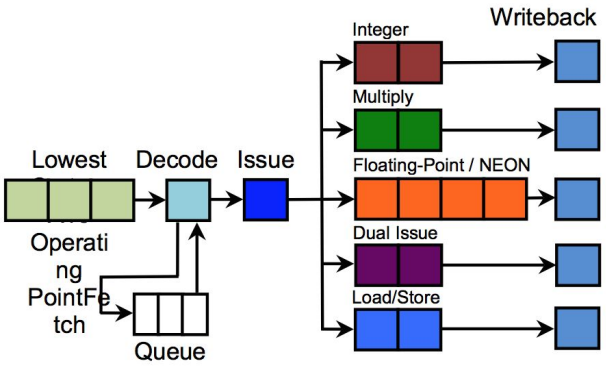
ARM Cortex-A7 (LITTLE)



“Partial” ALU (does not do all integers operations)

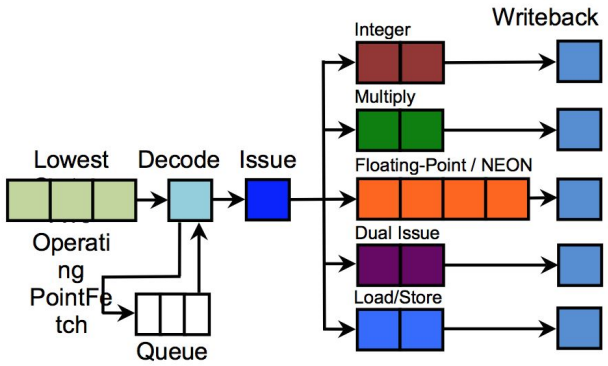
ARM Cortex-A7 (LITTLE) vs Cortex-A15 (Big)

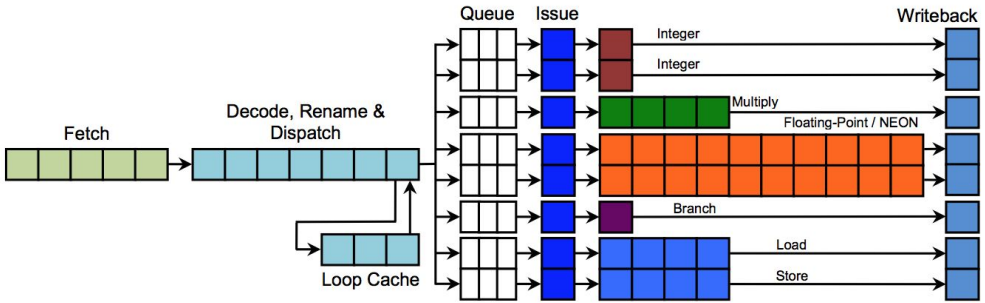
- In-order
- Non-symmetric Dual Issue
- Processor pipeline between 8-10 stages



ARM Cortex-A7 (LITTLE) vs Cortex-A15 (Big)

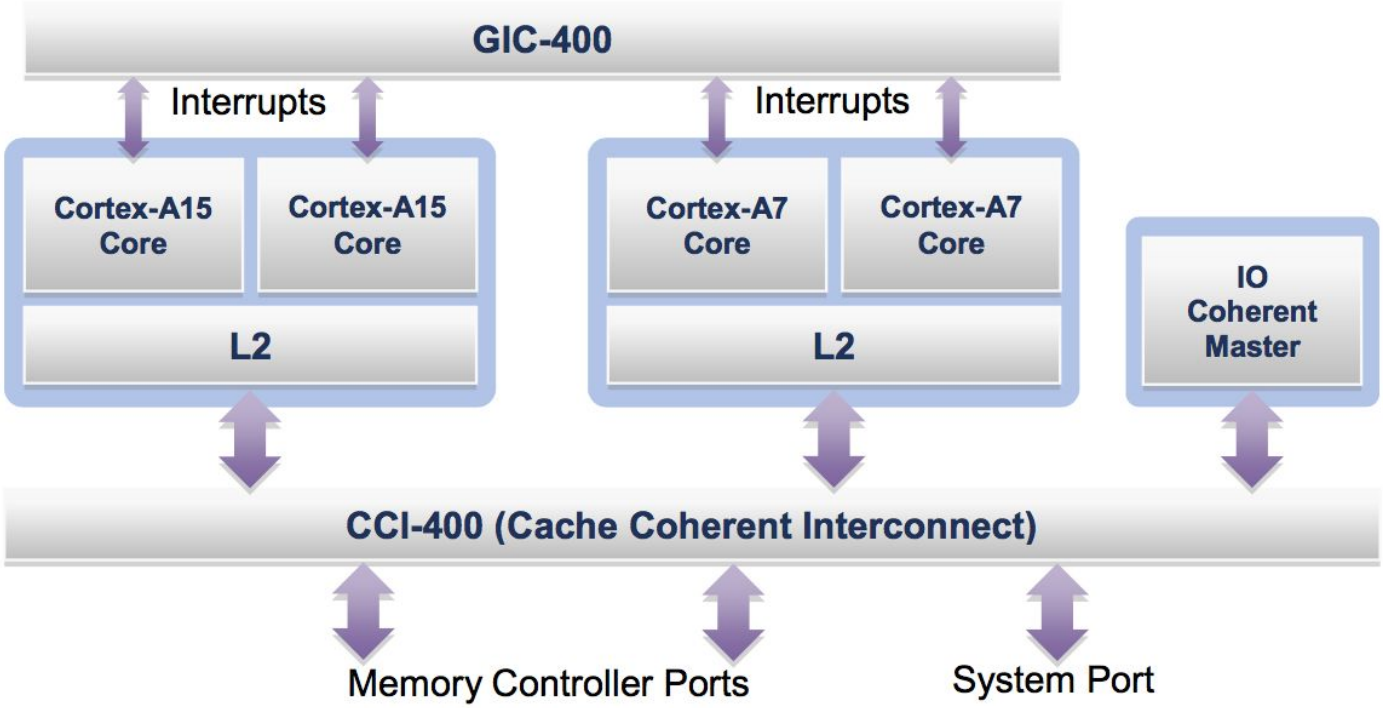
- In-order
- Non-symmetric Dual Issue
- Processor pipeline between 8-10 stages



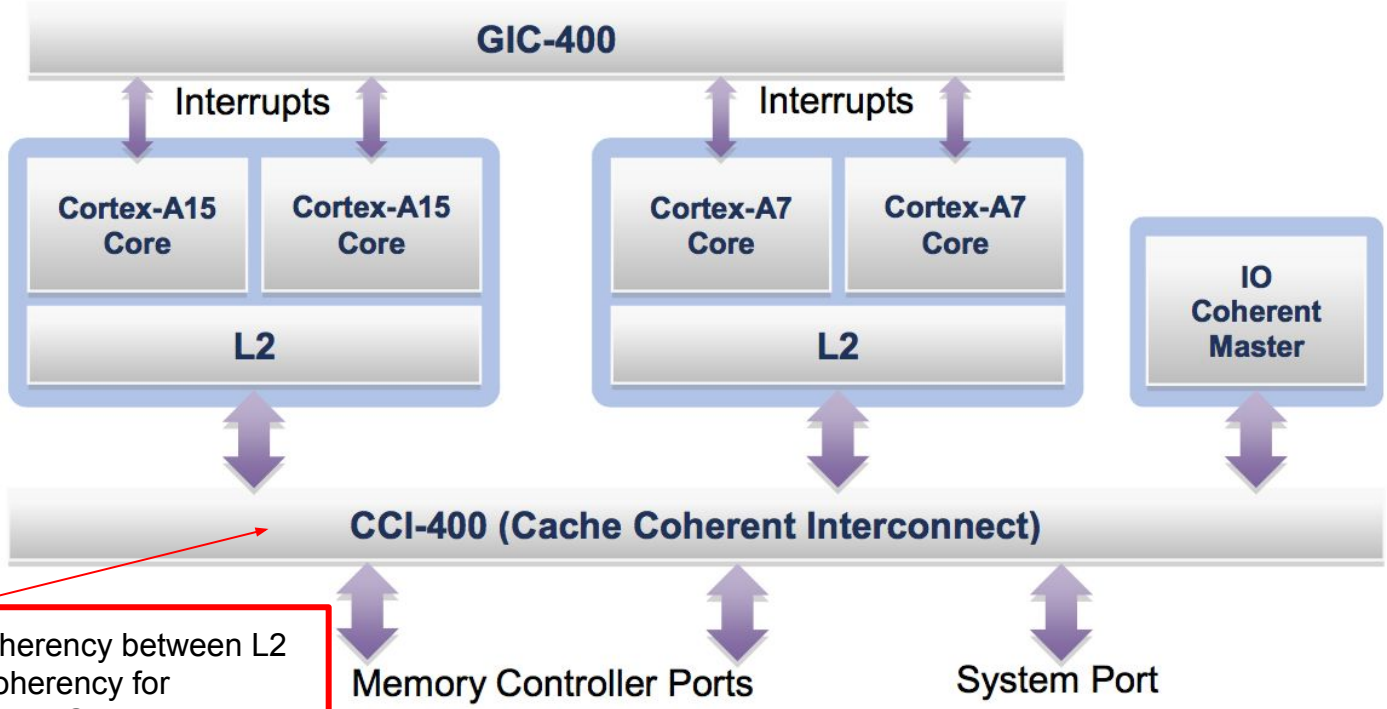


- Out-of-order
- Sustained Triple-issue
- Pipeline between 15-24 sages

Interconnection



Interconnection

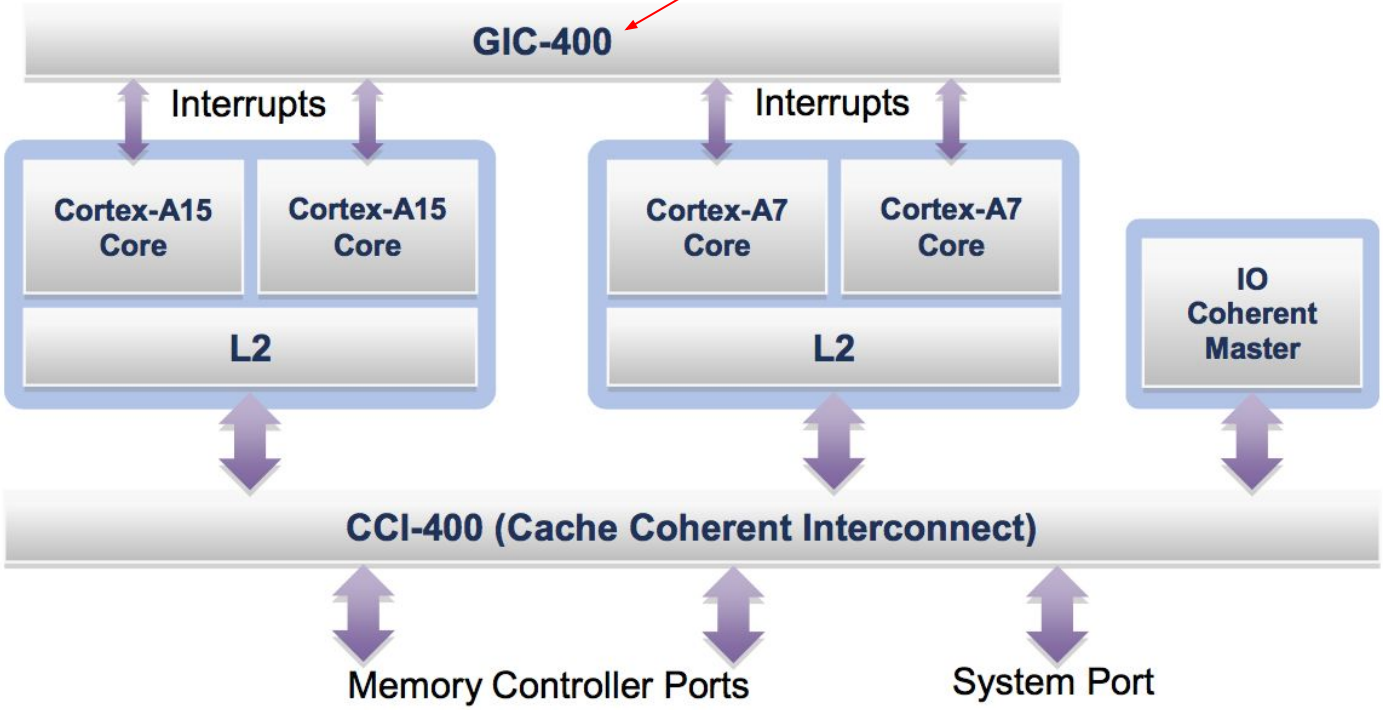


Facilitates full-coherency between L2 caches and IO coherency for components such as GPU.

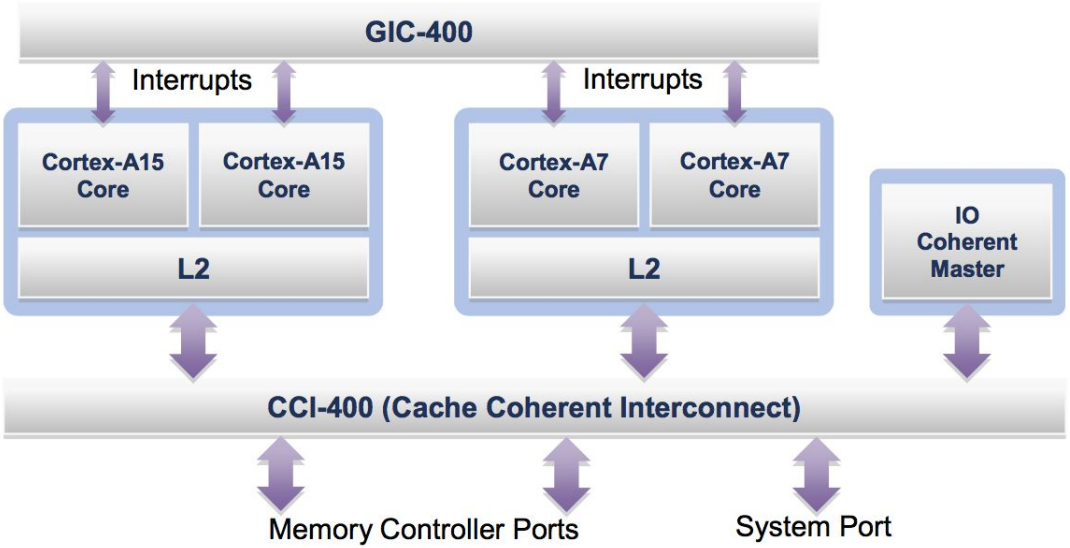
Interconnection

Distribute up to 480-interrupts.

GIC-400 allows interruptions to be migrated between any cores in the Cortex-A15 cluster or Cortex-A7 cluster.



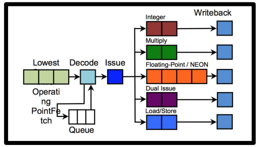
Interconnection

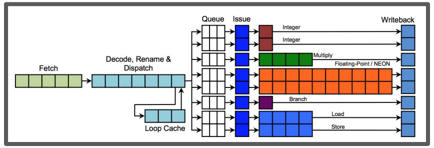


From ARM White Paper:

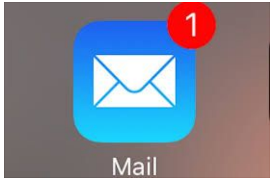
- Any arrangements of Big.LITTLE is possible.
- There is no feature or optimization for a specific configuration of Big.LITTLE prior from ARM.
- The recommendation is # bigs = # littles (in order to reduce **software complexity** of the task scheduler).

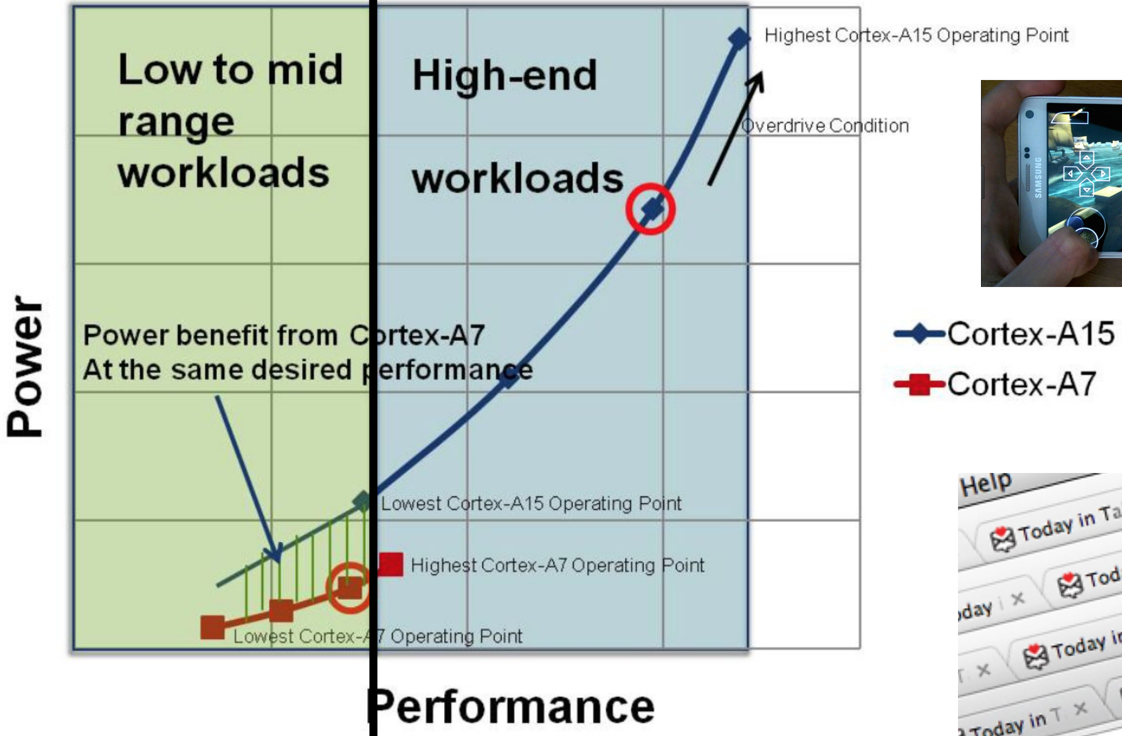
Usage


Writeback
Integer
Multiply
Lowest
Decode
Issue
Floating-Point / NEON
Operati
ng
PointFe
tch
Dual Issue
Load/Store
Queue


Queue
Issue
Writeback
Integer
Integer
Multiply
Floating-Point / NEON
Fetch
Decode, Rename &
Dispatch
Branch
Load
Store
Loop Cache

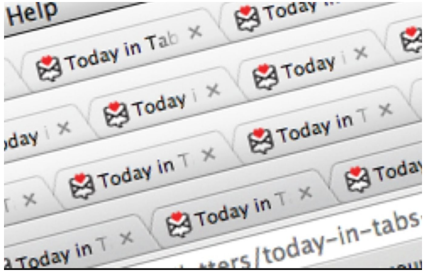

SMS


1
Mail


Low to mid range workloads
High-end workloads
Highest Cortex-A15 Operating Point
Overdrive Condition
Power benefit from Cortex-A7
At the same desired performance
Lowest Cortex-A15 Operating Point
Highest Cortex-A7 Operating Point
Lowest Cortex-A7 Operating Point
Power
Performance
Cortex-A15
Cortex-A7

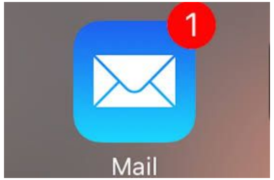

EA SPORTS
FIFA 18
FIFA
Fly Emirates
GET IT ON Google Play
Download on the App Store

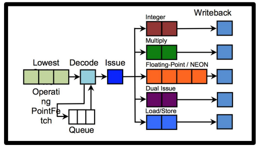

SAMSUNG
SELECT
START

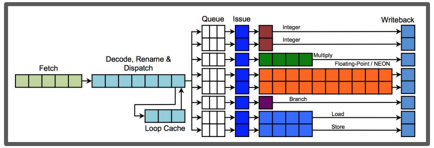

Help
Today in Tab
Today
Today in T
today-in-tabs

Usage

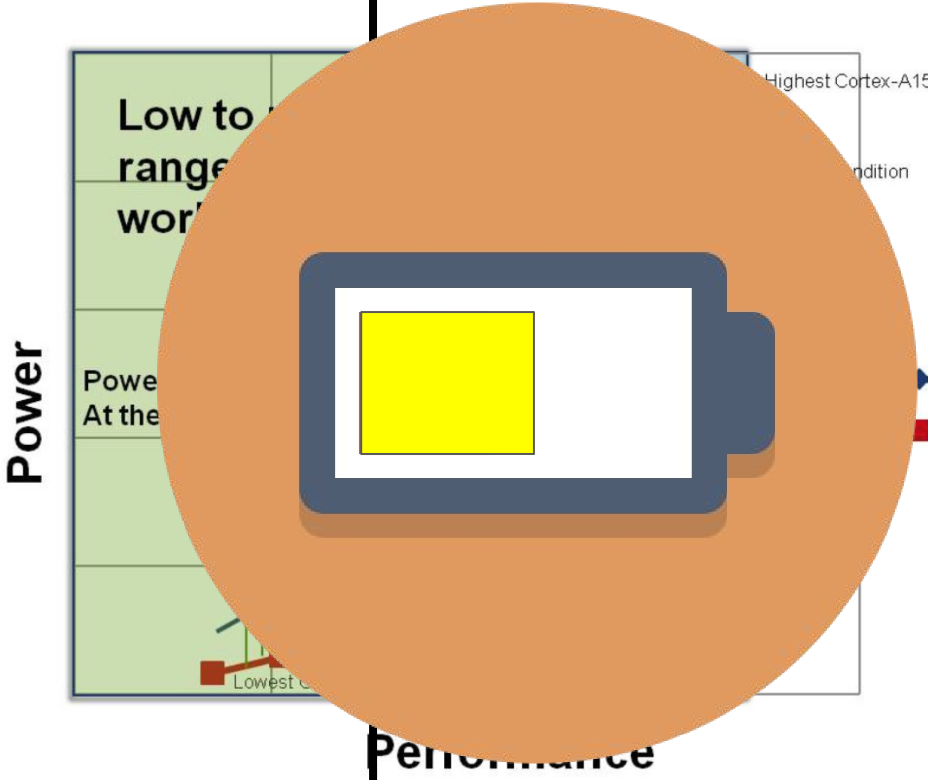

SMS


1
Mail

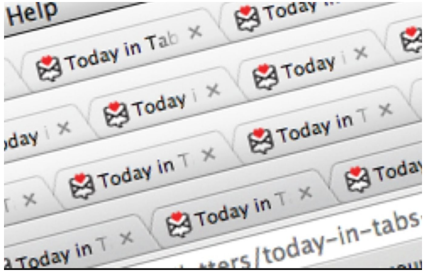

Writeback
Integer
Multiply
Lowest
Decode
Issue
Floating-Point / NEON
Operating PointFetch
Dual Issue
Load/Store
Queue


Queue
Issue
Writeback
Integer
Integer
Fetch
Decode, Rename & Dispatch
Multiply
Floating-Point / NEON
Branch
Load
Loop Cache
Store


EA SPORTS
FIFA18
FIFA
Fly Emirates
GET IT ON Google Play
Download on the App Store


Highest Cortex-A15 Operating Point
Low to
range
wor
ndition
Power
Powe
At the
Lowest
Performance
Cortex-A15
Cortex-A7


SAMSUNG
SELECT
START


Help
Today in Tab
Today
Today
Today
Today in T
Today in T
Today in T
Today in T
Today
ters/today-in-tabs

Task Migration

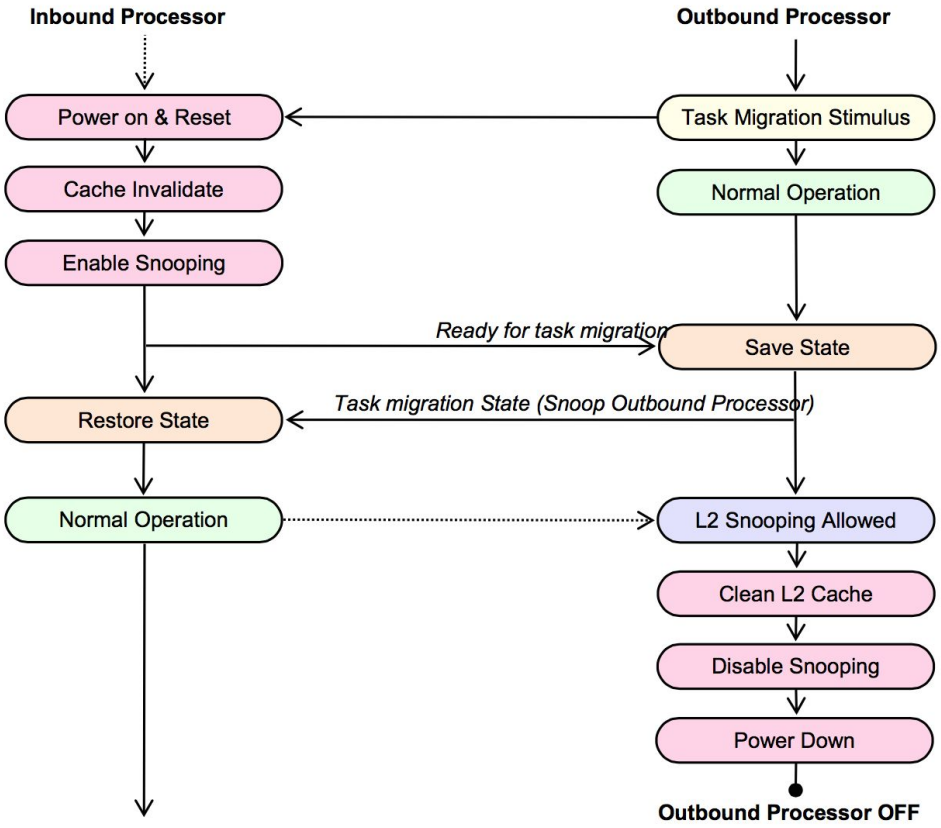
Migration between a little to a big core or vice-versa takes less than **20.000 cycles**.

In a 1Ghz processors, it would take **20 us (micro-seconds)**.

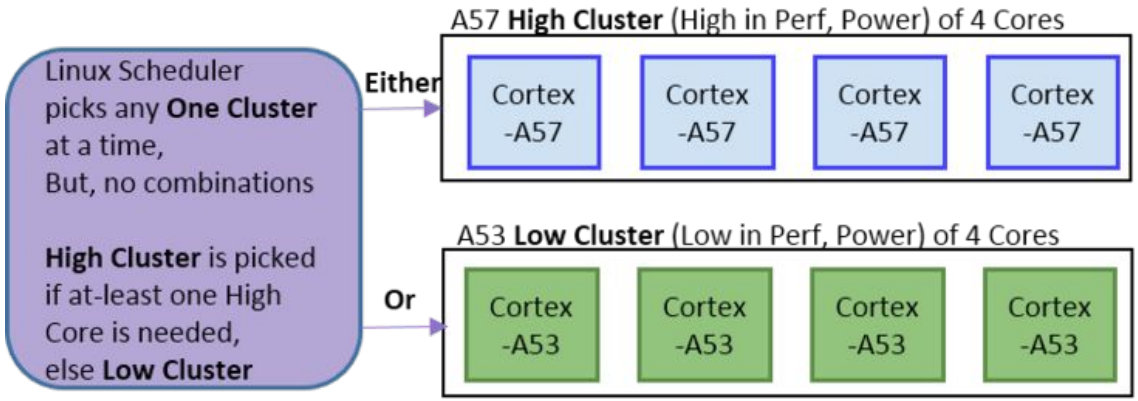
In practice, it goes around 30us

This is low:

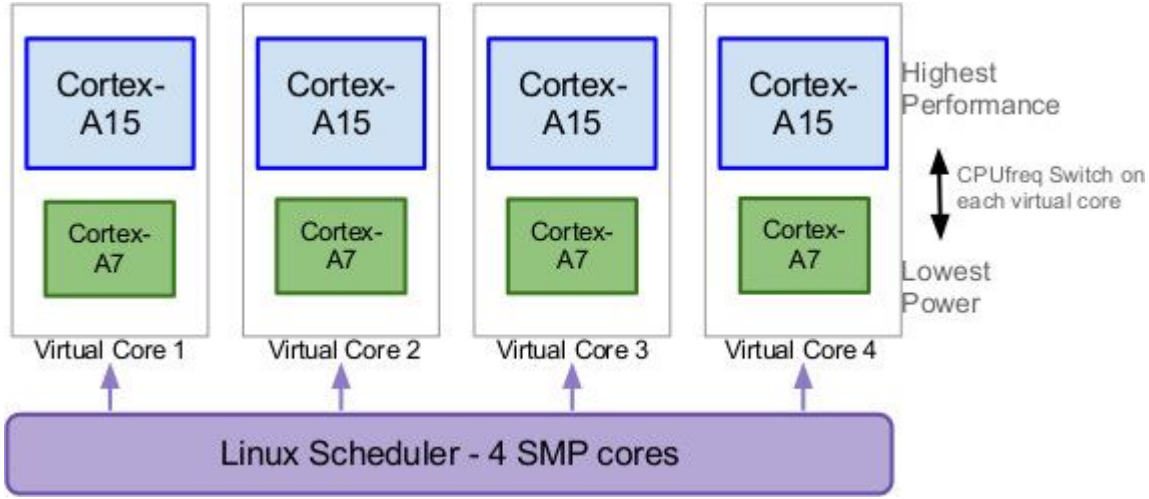
- DVFS I would wait around 100/200 microseconds.
- Any consistent load monitoring takes at least 1 millisecond.



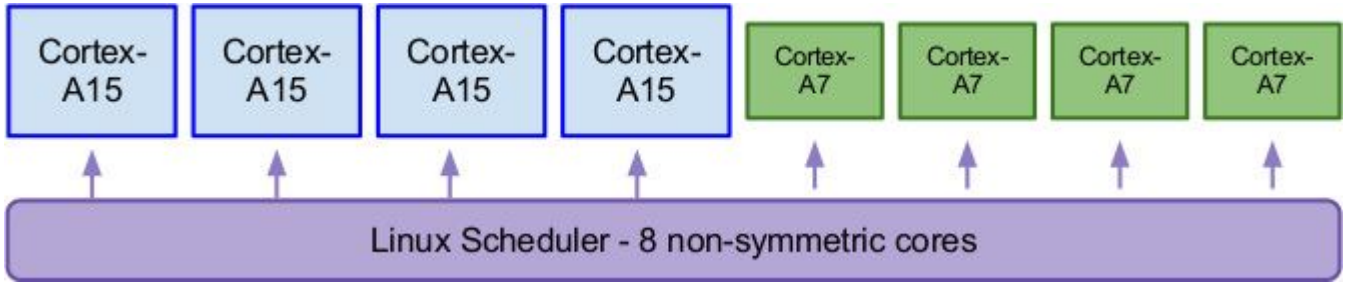
Big.LITTLE arrangement: clustered switching



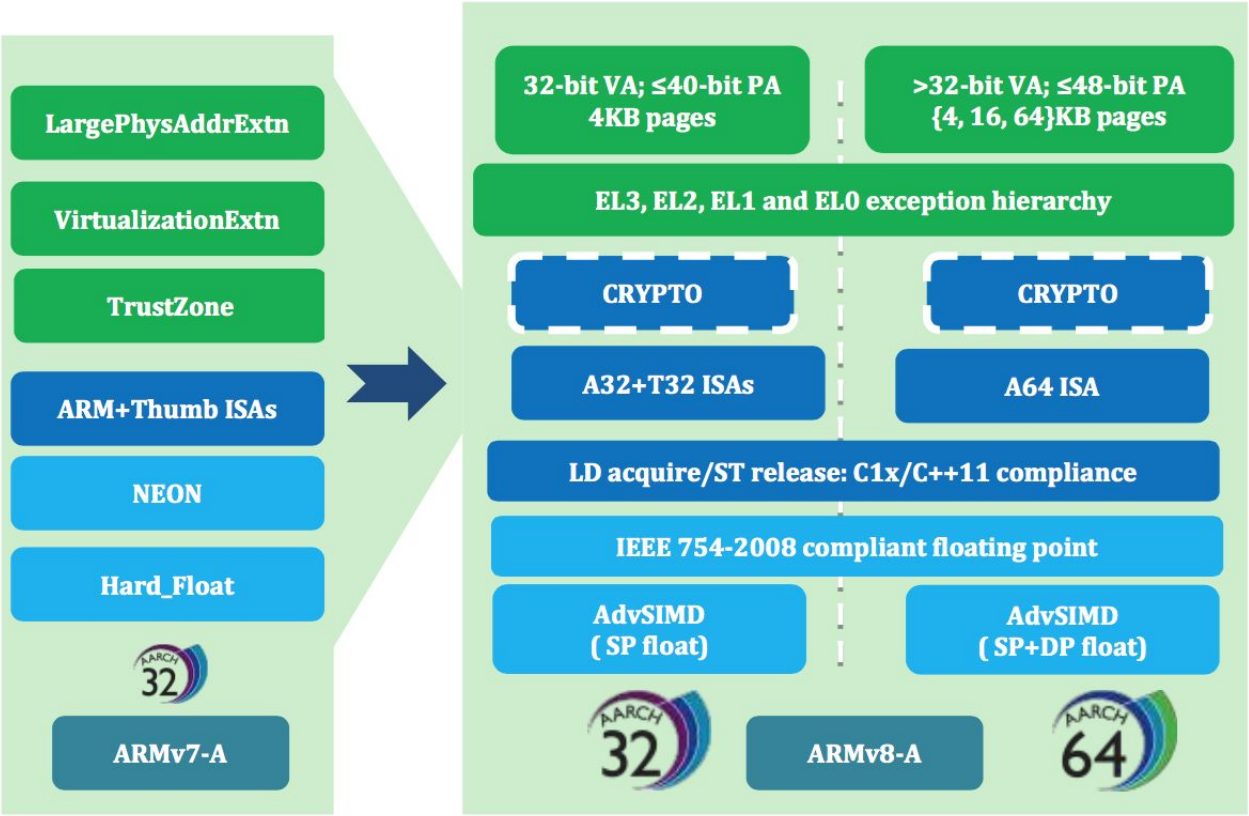
Big.LITTLE arrangement: in-kernel switcher



Big.LITTLE arrangement: heterogeneous multi-processing

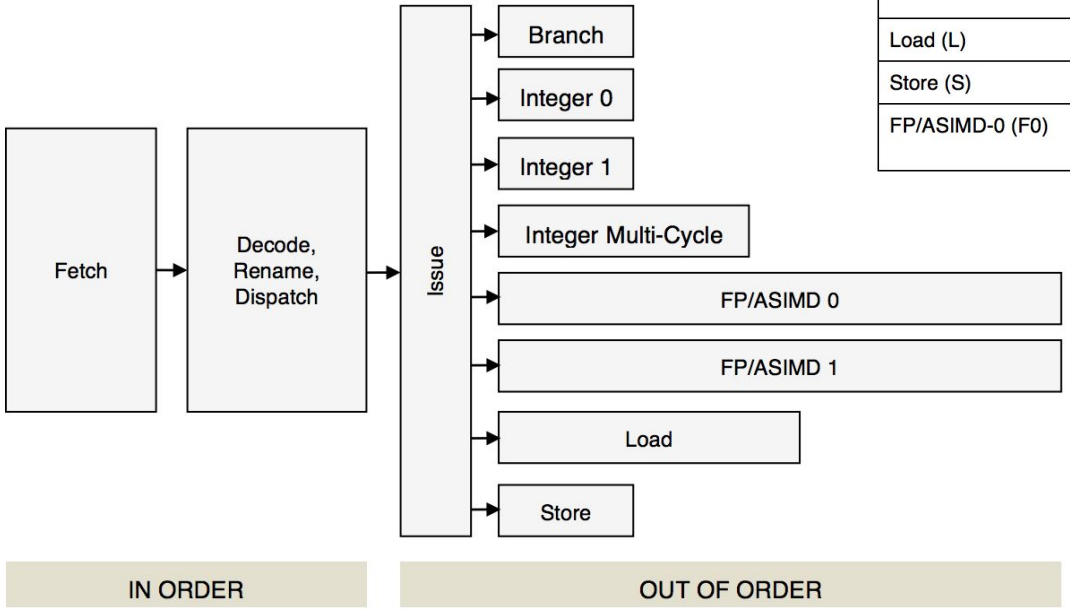


Big.LITTLE on armv8



Big.LITTLE on armv8

A-57



Pipeline (mnemonic)	Supported functionality
Branch (B)	Branch μops
Integer 0/1 (I)	Integer ALU μops
Multi-cycle (M)	Integer shift-ALU, multiply, divide, CRC and sum-of-absolute-differences μopμops
Load (L)	Load and register transfer μops
Store (S)	Store and special memory μops
FP/ASIMD-0 (F0)	ASIMD ALU, ASIMD misc, FP misc, FP add, FP multiply, ASIMD integer multiply, FP divide μops, crypto μops

Big.LITTLE on armv8

A-53

- Versatile, can be paired with any Armv8.0 core in a big.LITTLE configuration, including Cortex-A57, Cortex-A72, other Cortex-A53, and Cortex-A35 processors.

  This processor can also be implemented in an Arm big.LITTLE configuration.

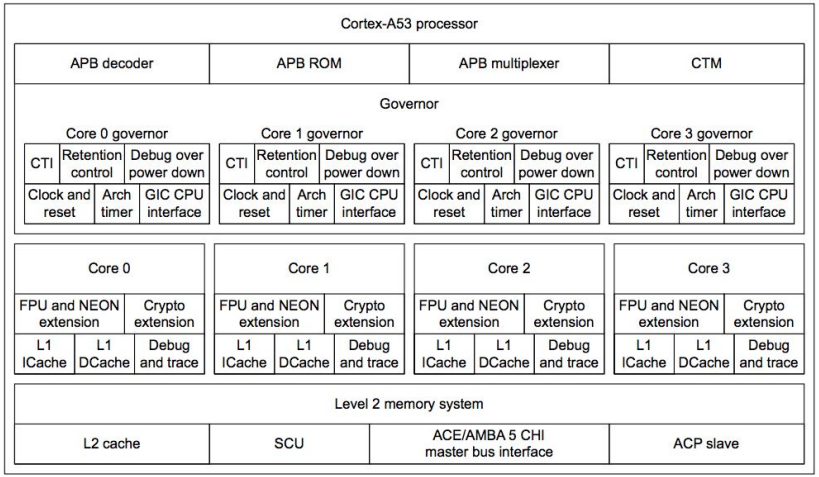
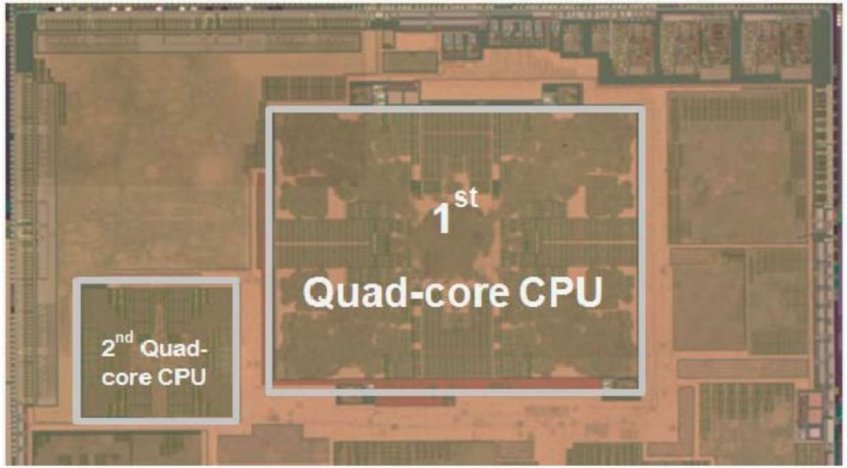


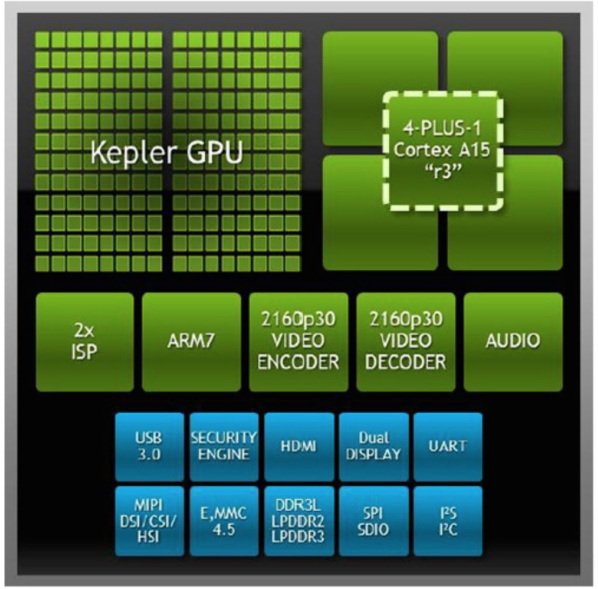
Figure 2-1 Cortex-A53 processor block diagram

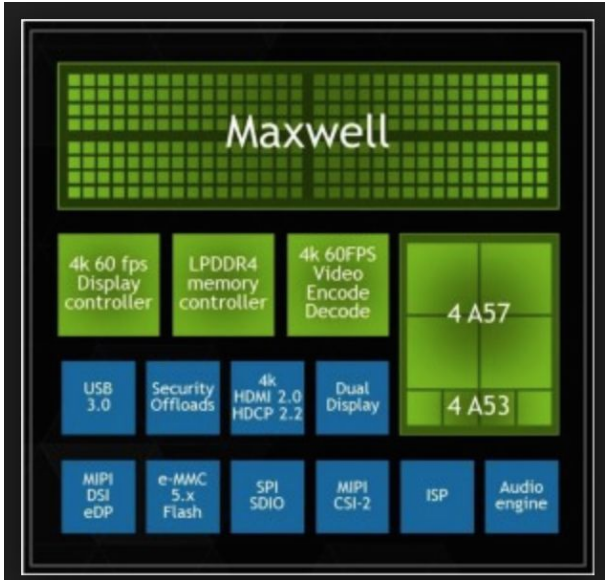
Applications

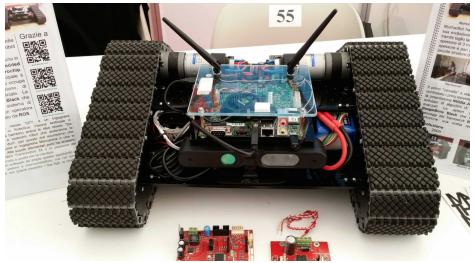


Inside the first Samsumg Exynos Octacore chip

Applications



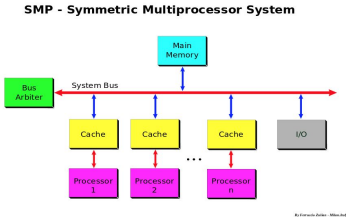




NVIDIA Tegra TK1 - 4 Plus 1

NVIDIA Tegra X1 - 4-4

A bit of History



Homogeneous computing: adequate performance for many applications in the past.

- More parallelization
- Heterogeneous consumption of resources from applications
- Opportunity: initially for thinking on performance

Heterogeneous Computing

“is the well-orchestrated and coordinated effective use of a **suite of diverse** high-performance **machines** (including parallel machines) to provide superspeed processing for **computationally demanding tasks with diverse computing needs**. An HC system includes heterogeneous machines, high-speed networks, interface.” (1993)

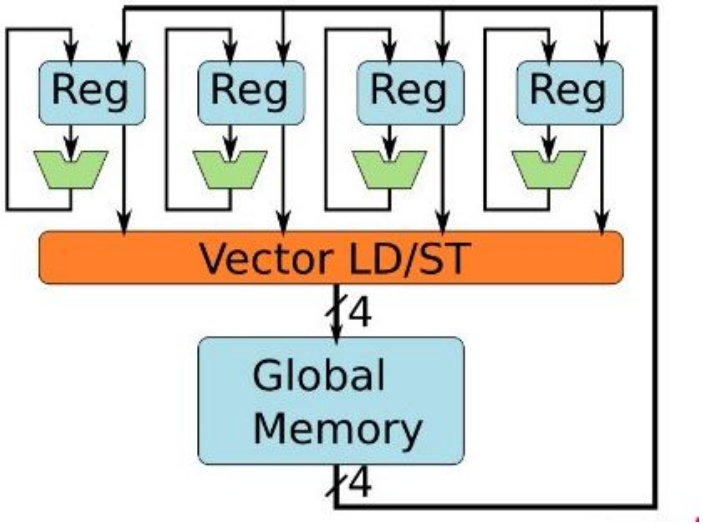
“refers to the use of **different processing cores** to maximize performance” (2010)

Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs

Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs
  - FPGAs



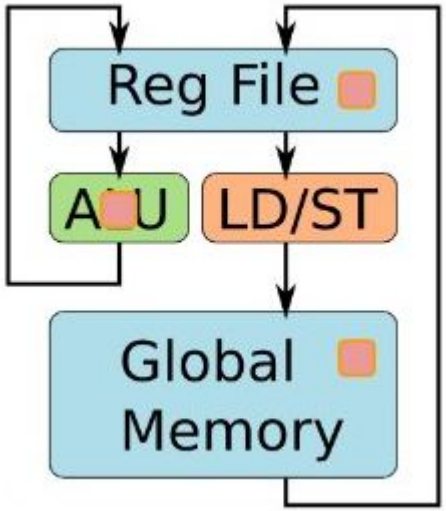
Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs
  - FPGAs
- CPU example

```
int A[2][4];
for(i=0;i<2;i++){
    for(j=0;j<4;j++){
        A[i][j]++;
    }
}
```

Assembly code of inner-loop

```
lw r0, 4(r1)
addi r0, r0, 1
sw r0, 4(r1)
```



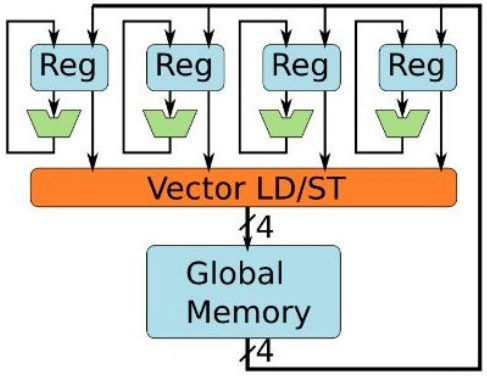
Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs
  - FPGAs
- SSE example - 4 ALUs

```
int A[2][4];
for(i=0;i<2;i++){
    for(j=0;j<4;j++){
        A[i][j]++;
    }
}
```

→

```
int A[2][4];
for(i=0;i<2;i++){
    movups xmm0, [ &A[i][0] ] // load
    addps xmm0, xmm1  // add 1
    movups [ &A[i][0] ], xmm0 // store
}
```



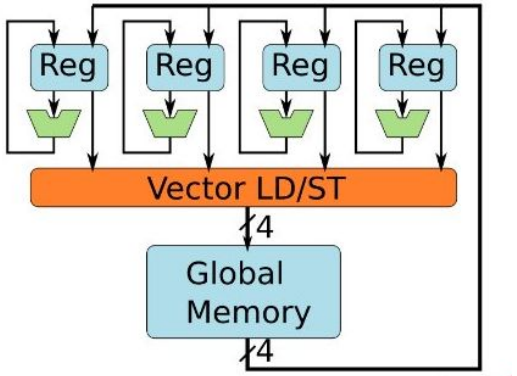
Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs
  - FPGAs
- SSE
  - X86 Pentiums - AMD Atlhon XP

```
int A[2][4];
for(i=0;i<2;i++){
    for(j=0;j<4;j++){
        A[i][j]++;
    }
}
```

→

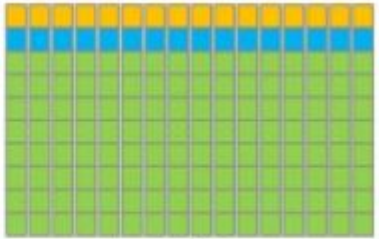
```
int A[2][4];
for(i=0;i<2;i++){
    movups xmm0, [ &A[i][0] ] // load
    addps xmm0, xmm1  // add 1
    movups [ &A[i][0] ], xmm0 // store
}
```



Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs

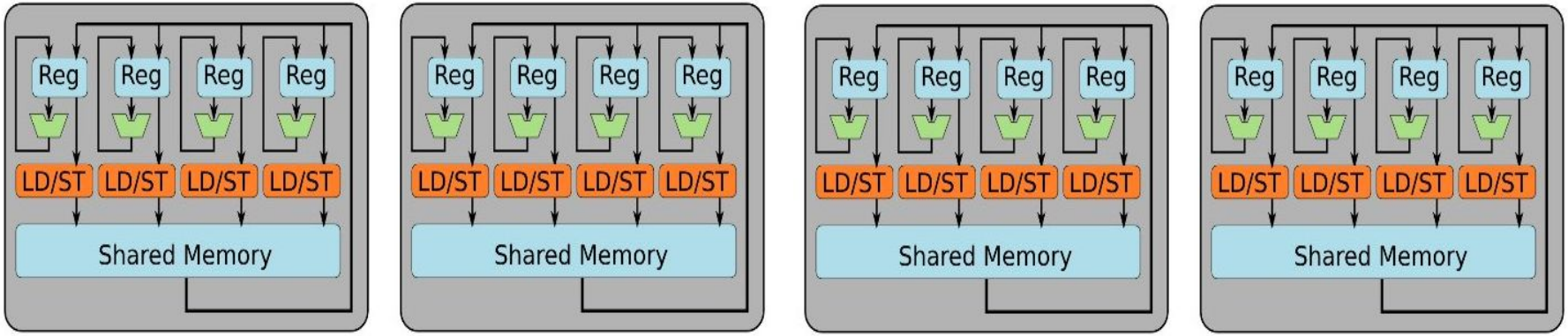
CPU



GPU

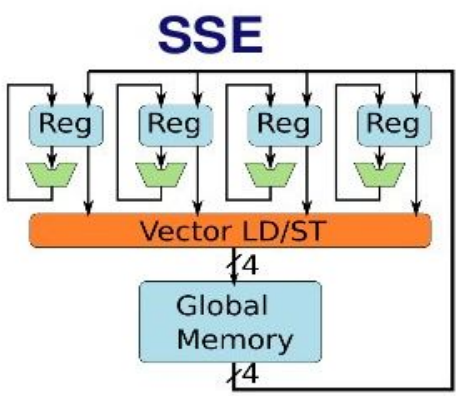
Accelerators

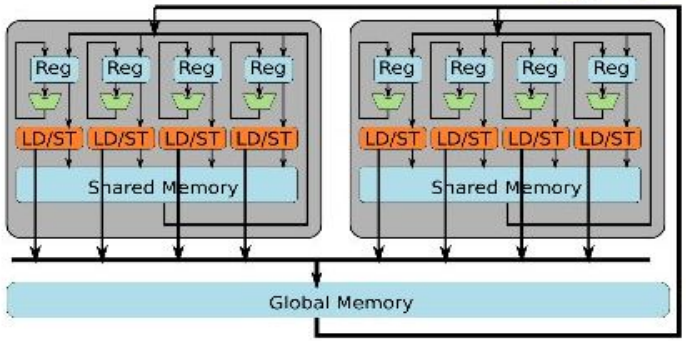
- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- GPU contains multiple SIMD cores.



Accelerators

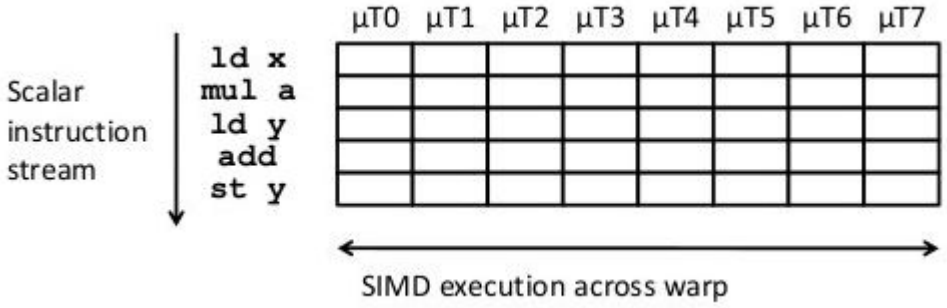
- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- Shared memory





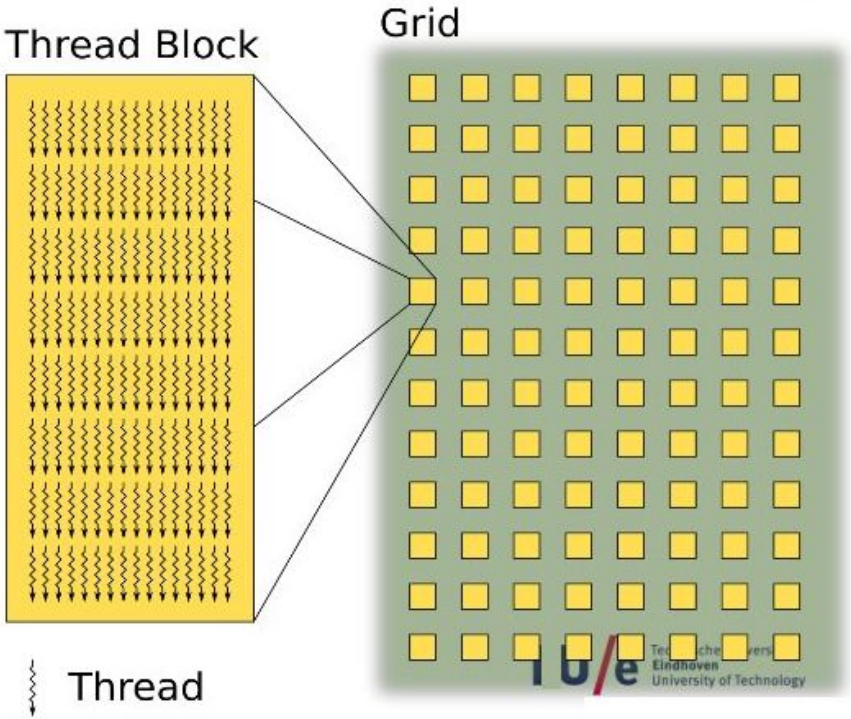
Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- SIMT (Single Instruction Multiple Threads)



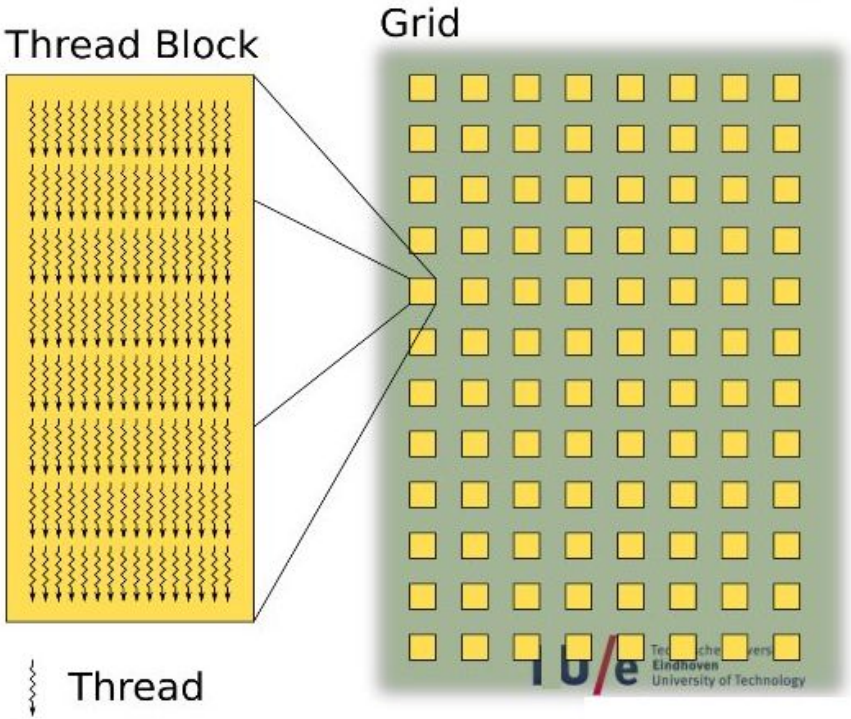
Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- Thread Hierarchy
  - Grid contains Blocks
  - Blocks contains Threads



Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- Thread Hierarchy
  - Grid contains Blocks
  - Blocks contains Threads
  - NVIDIA
    - 32 threads = *warp*

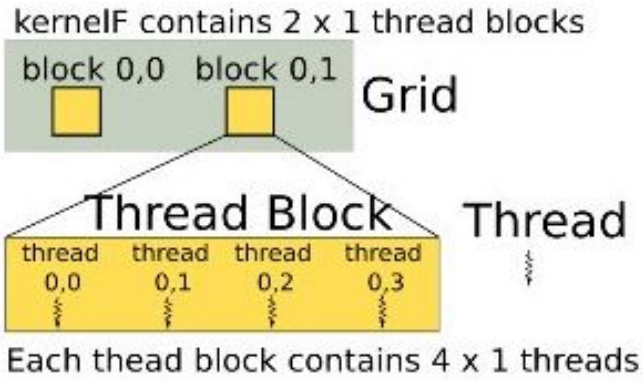


Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- Thread Hierarchy
  - Grid contains Blocks
  - Blocks contains Threads
  - NVIDIA
    - 32 threads = *warp*

```
int A[2][4];
kernelF<<<(2,1),(4,1)>>>(A);   // define threads

__device__    kernelF(A){   // all threads run same kernel
    i = blockIdx.x;         // each thread block has its id
    j = threadIdx.x;        // each thread has its id
    A[i][j]++;              // each thread has a different i and j
}
```

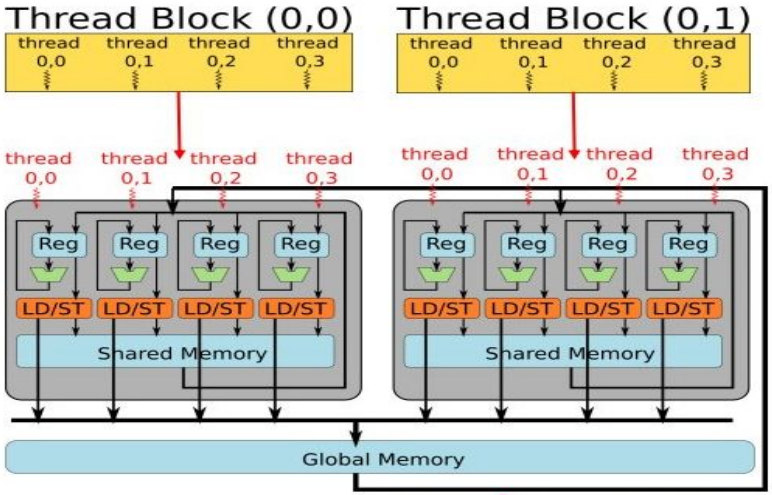


Accelerators

- CPU sometimes is not good enough!
  - SSEs (Stream SIMD Extensions)
  - GPUs (Graphics Processing Unit)
  - FPGAs
- Thread Hierarchy
  - Grid contains Blocks
  - Blocks contains Threads
  - NVIDIA
    - 32 threads = *warp*

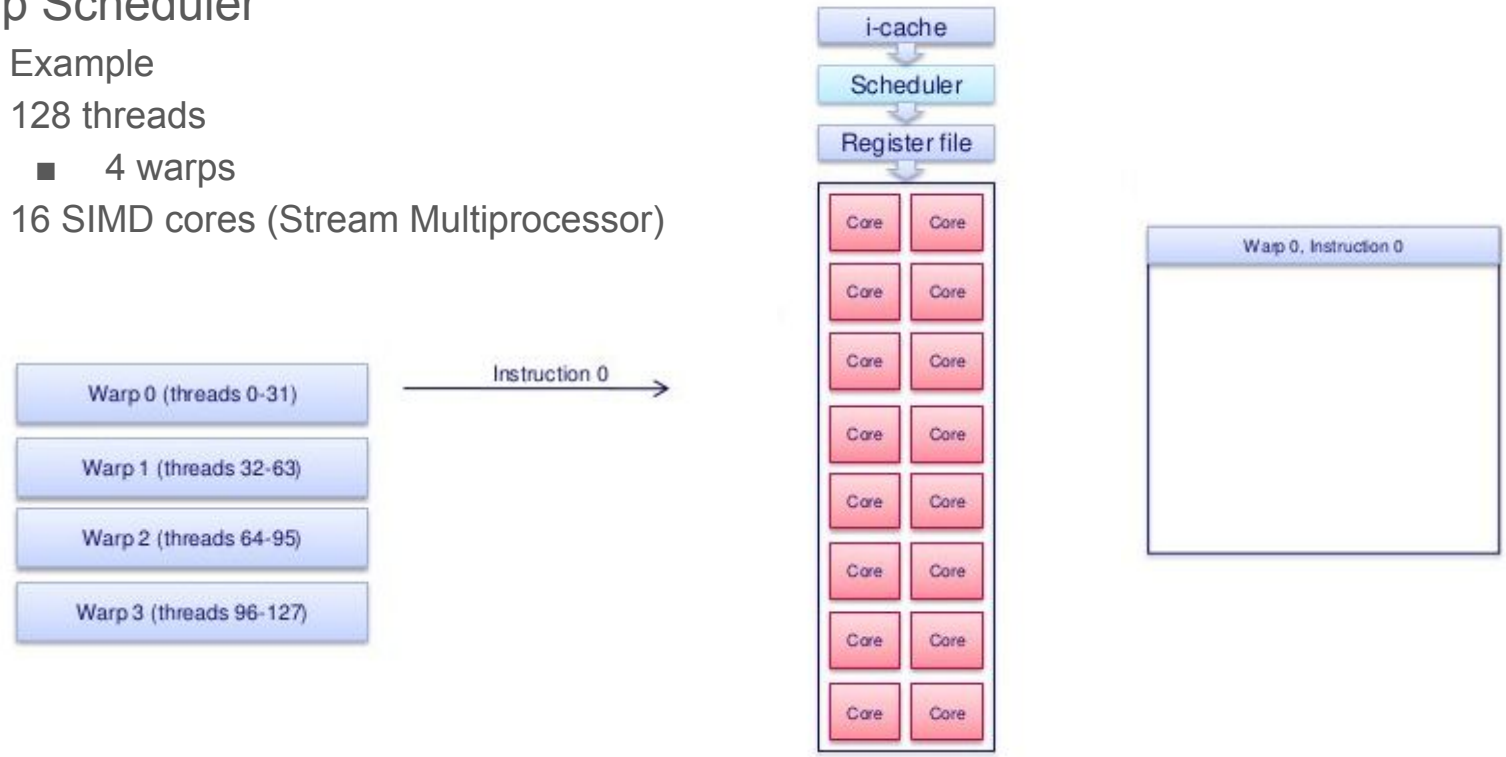
```
int A[2][4];
kernelF<<<(2,1),(4,1)>>>(A);   // define threads

__device__    kernelF(A){   // all threads run same kernel
    i = blockIdx.x;         // each thread block has its id
    j = threadIdx.x;        // each thread has its id
    A[i][j]++;              // each thread has a different i and j
}
```



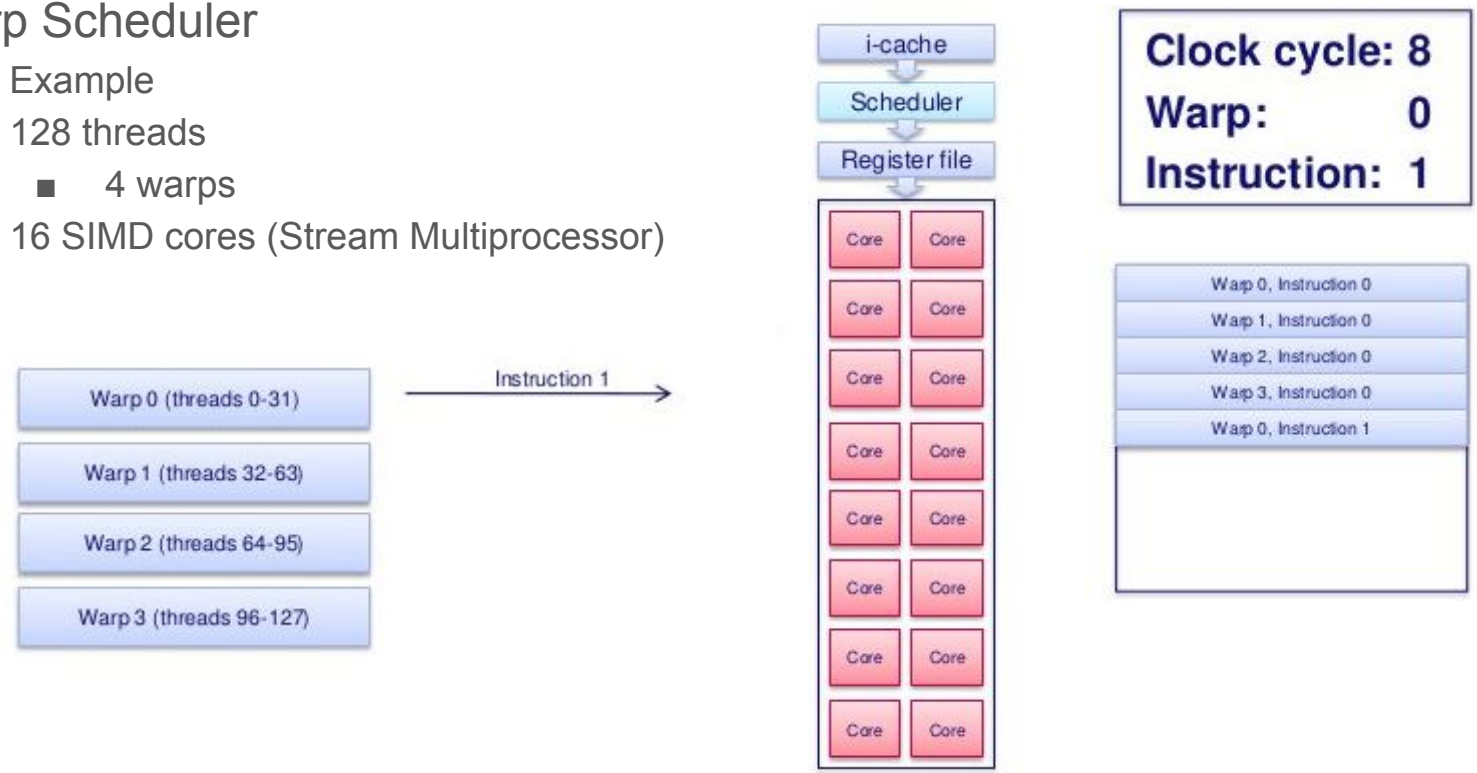
Accelerators

- Warp Scheduler
  - Example
  - 128 threads
    - 4 warps
  - 16 SIMD cores (Stream Multiprocessor)



Accelerators

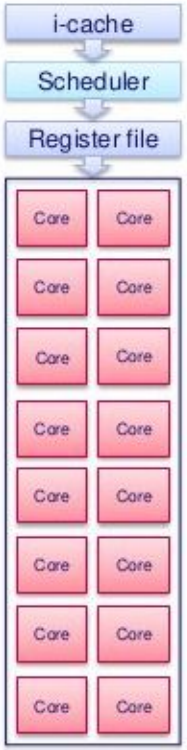
- Warp Scheduler
  - Example
  - 128 threads
    - 4 warps
  - 16 SIMD cores (Stream Multiprocessor)



Accelerators

- Warp Scheduler
  - Example
  - 128 threads
    - 4 warps
  - 16 SIMD cores (Stream Multiprocessor)

```
__device__    kernelF(A){
    i = blockIdx.x;
    j = threadIdx.x;
    if(j%2 == 1)
        A[i][j]++;
    else
        A[i][j]--;
}
```



Accelerators

- Warp Scheduler

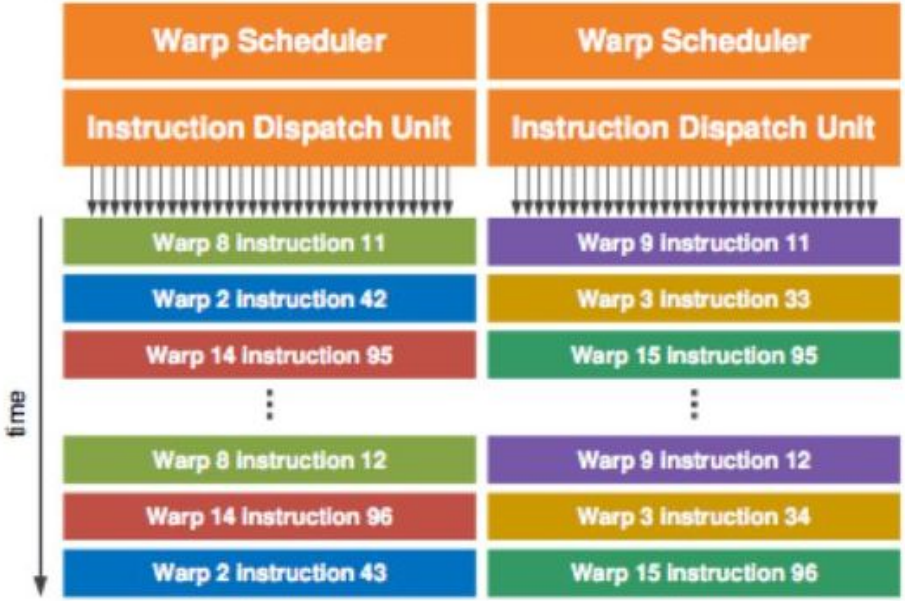
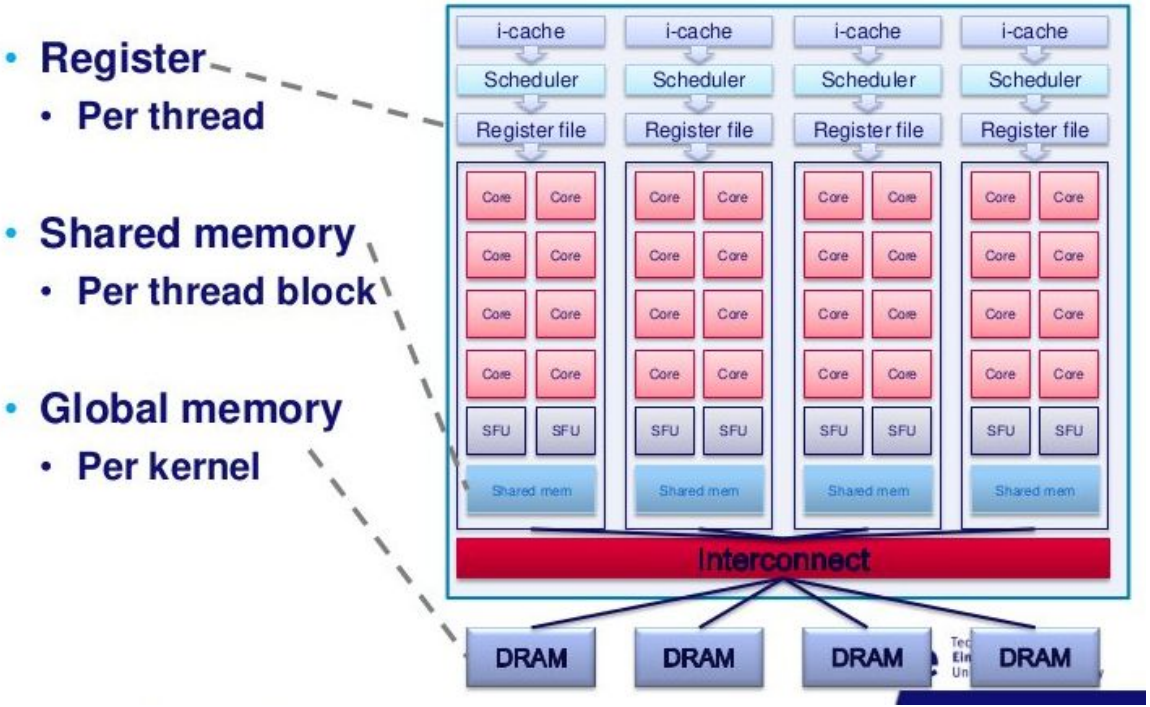


Image Credit: NVIDIA

http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

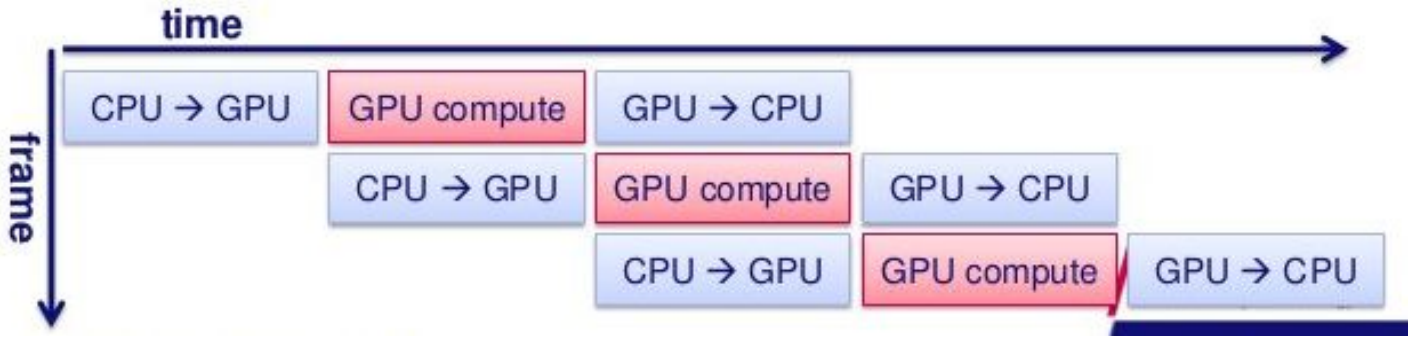
Accelerators

- Memory Hierarchy



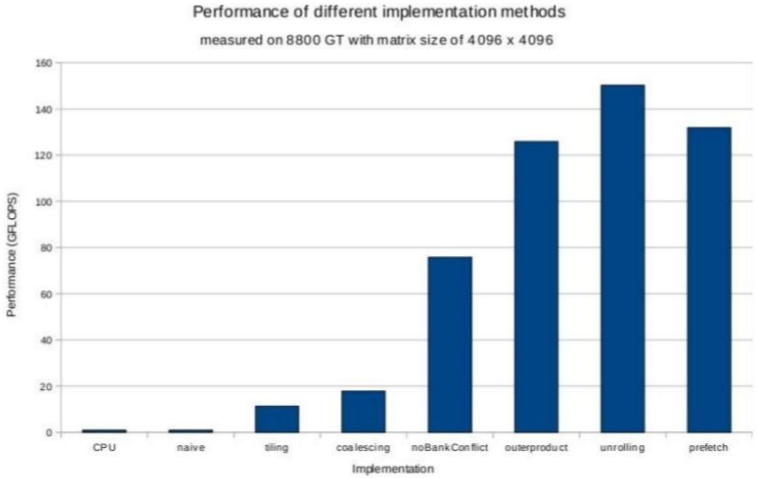
Accelerators

- Memory Hierarchy
- Takes long time to transmit data from CPUs to GPGPUs



Accelerators

- Memory Hierarchy
- Takes long time to transmit data from CPUs to GPGPUs
- Matrix Multiplication



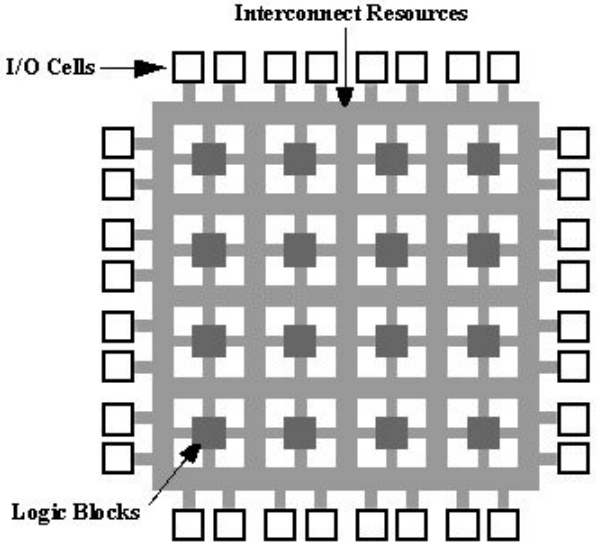
Accelerators

Nvidia Fermi GF100 GPU

Image Credit: NVIDIA [Wittenbrink, Kilgariff, and Prabhu, Hot Chips 2010]

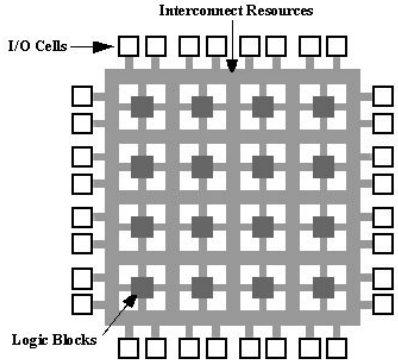
Accelerators

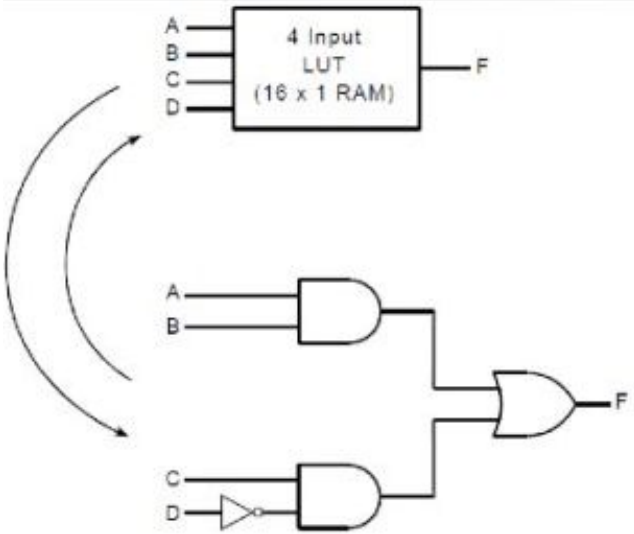
- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs (Field-Programmable Gate Array)



Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
- Logic as Memories
  - LUT (LookUp Table)

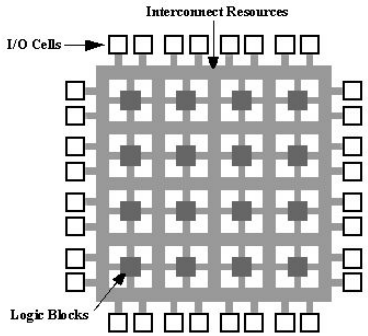


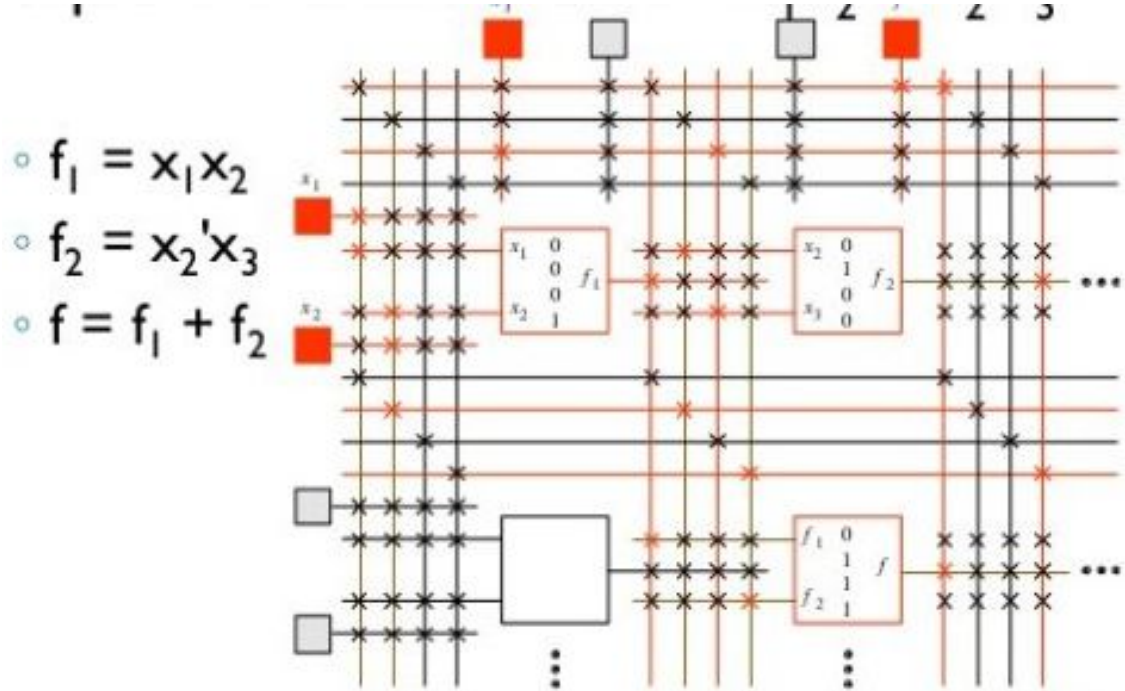


RAM Contents				
Address				Data
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

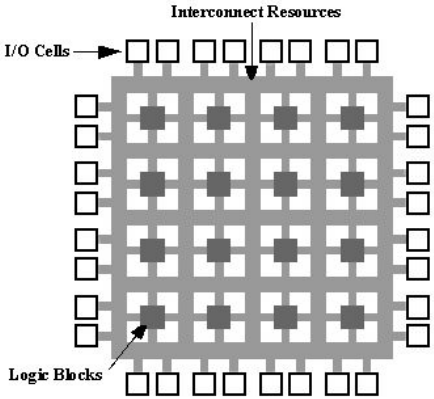
Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
- Interconnection

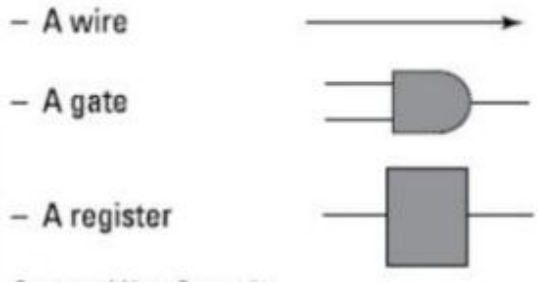




Accelerators



- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
- Logical Blocks



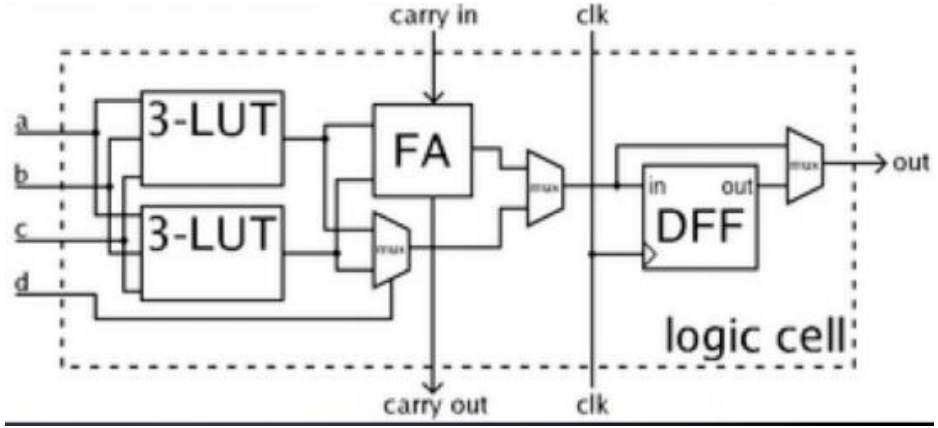
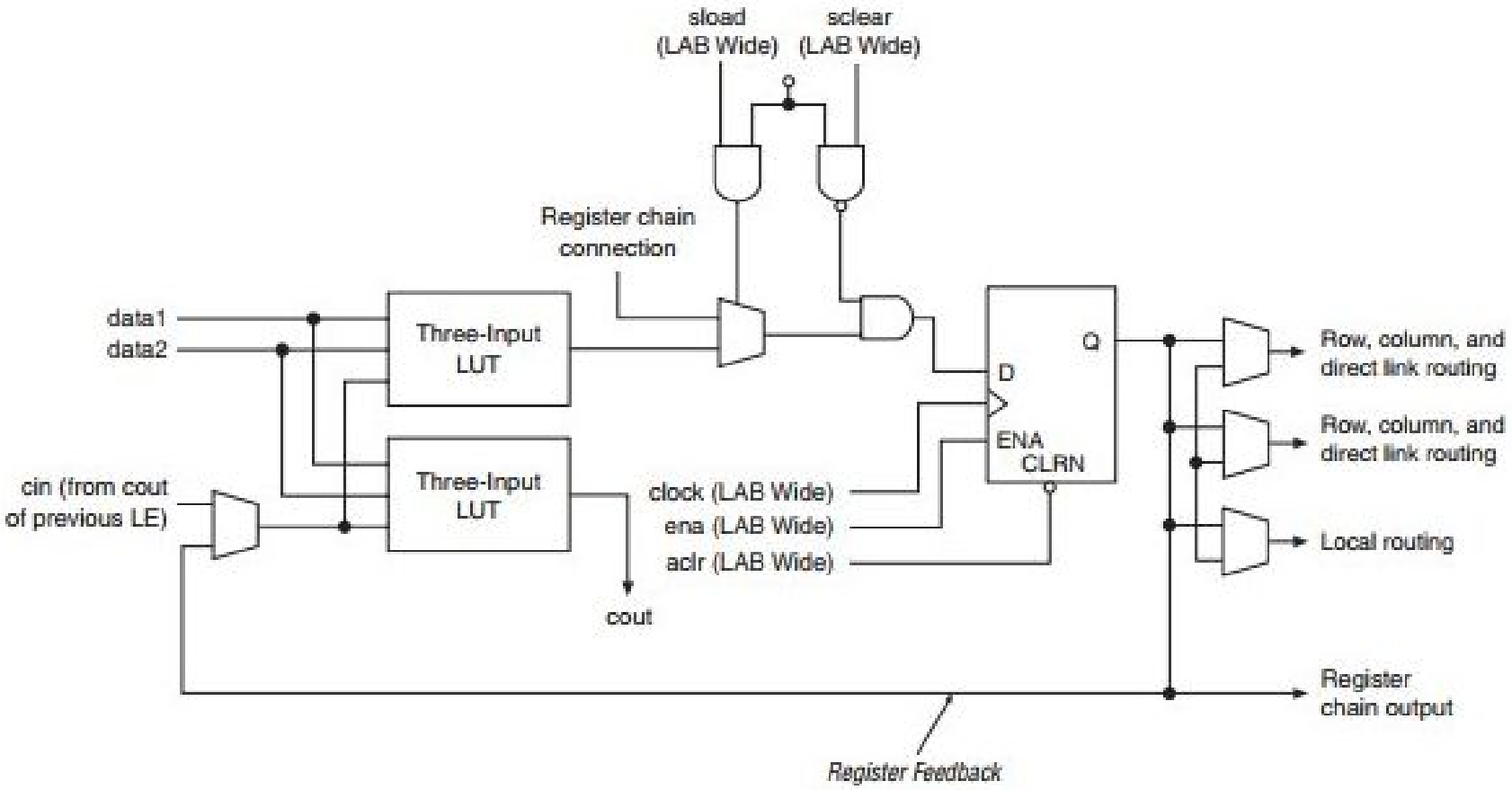


Figure 2–4. LE in Arithmetic Mode

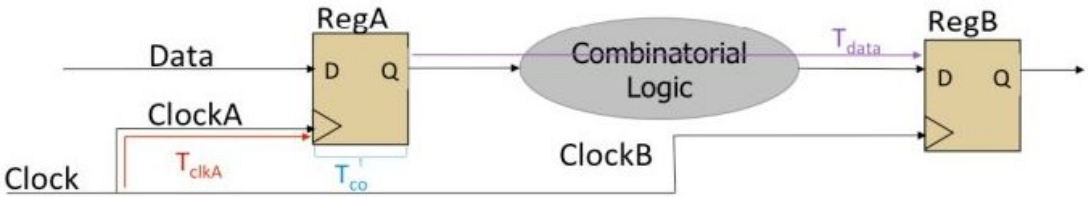


Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
    - Programming
      - Analysis & Synthesis (From HDL)
      - Fitter
      - Assembler
      - Timing Analysis

Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
    - Programming
      - Analysis & Synthesis (From HDL)
      - Fitter
      - Assembler
      - Timing Analysis



Accelerators

- CPU sometimes is not good enough!
  - SSEs
  - GPUs
  - FPGAs
    - Custom Blocks
      - DSPs
      - Multipliers
      - Embedded Memories
    - Applications
      - Image Processing
      - DNN (!!)
      - Prototyping
      - SoC FPGAs (CPU + FPGA)

References

Khokhar, Ashfaq A., et al. "Heterogeneous computing: Challenges and opportunities." Computer 26.6 (1993): 18-27.

Brodtkorb, Andre R., et al. "State-of-the-art in heterogeneous computing." Scientific Programming 18.1 (2010): 1-33.

Coursera: Computer Architecture (Princeton - David Wentzlaff); Introduction to FPGA Design for Embedded Systems (University of Colorado Boulder)

Henk Corporaal, Gert-Jan van den Braak (Introduction to GPGPU and GPU-architectures)