Introduction to High Performance Computing

Dr. Tassadaq Hussain

RIPHAH INTERNATIONAL UNIVERSITY

Microsoft Research Centre

- **Personal Introduction**
- Problem Formulation
  - Importance of Information
  - Importance of Digital Electronics
  - Problems and Solution
- HPC For Real World Problems
  - Hardware Architecture
  - Programming Models
  - Design Approach
- Projects

Education

- International **PhD BarcelonaTech Barcelona** 2014
  - Computer Architecture
  - High Performance System Design

- **Masters ISEP Paris** 2009
  - Electronics and Communication
  - Electronics for System

- B.Sc. (Electrical Engineering), RIU Islamabad 2005

Microsoft Research

Infineon technologies

Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

Experience

- **Riphah International University as** Assistant Professor
  - January 2015 to till date
- **Microsoft Barcelona supercomputing center**
  August 2009 – December 2014 www.bscmsrc.es
  - Worked closely with High Level Synthesis designers at **Ylichron technologies** (**PLDA Italia**) to develop Three-dimensional memory organization for stencil computation www.ylichron.it
  - Designed Programmable Memory Controller for Vector System on Chip **Microsoft Research Cambridge research.microsoft.com**
- **Infineon Technology** digital design department
  July 2008 to 31st March 2009 www.infineon.com
- **Pakistan Broadcasting Corporation** as Senior Broadcasting Engineer
  August 2005 to September 2007 www.radio.gov.pk
- **Center for Advanced Research in Engineering**
  August 2004 to August 2005 www.carepvtltd.com

Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

Projects

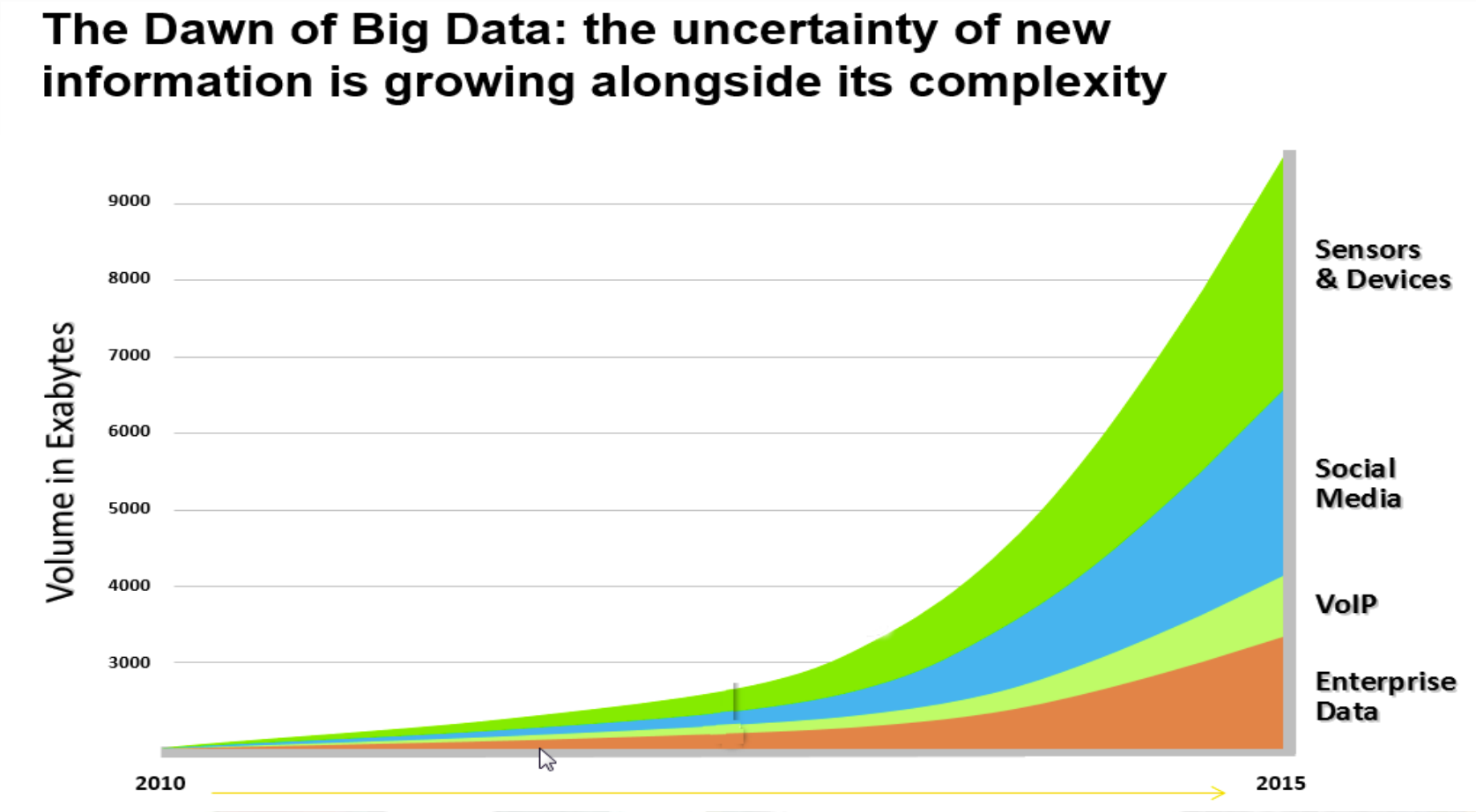
- Design Ultra Low Cost Display Camera Interface for Mobile Baseband XGold Chip at **Infineon Technologies France.**
- Implementation of Reverse Time Migration on FGPAs at **PLDA Italia** and **REPSOL BSC Research Center.**
- Programmable Memory Controller for Vector System on Chip **Microsoft Research Cambridge.**
- Programmable Vector Memory Controller for **European ParaDIME research group at BSC.**
- **Low Power Low Cost Supercomputer Architecture for Undeveloped Countries** at **RIU Pakistan and BSC Spain.**
- **ViPS: Visual Processing Toolkit** at **UCERD Pakistan and BSC Spain.**

Research

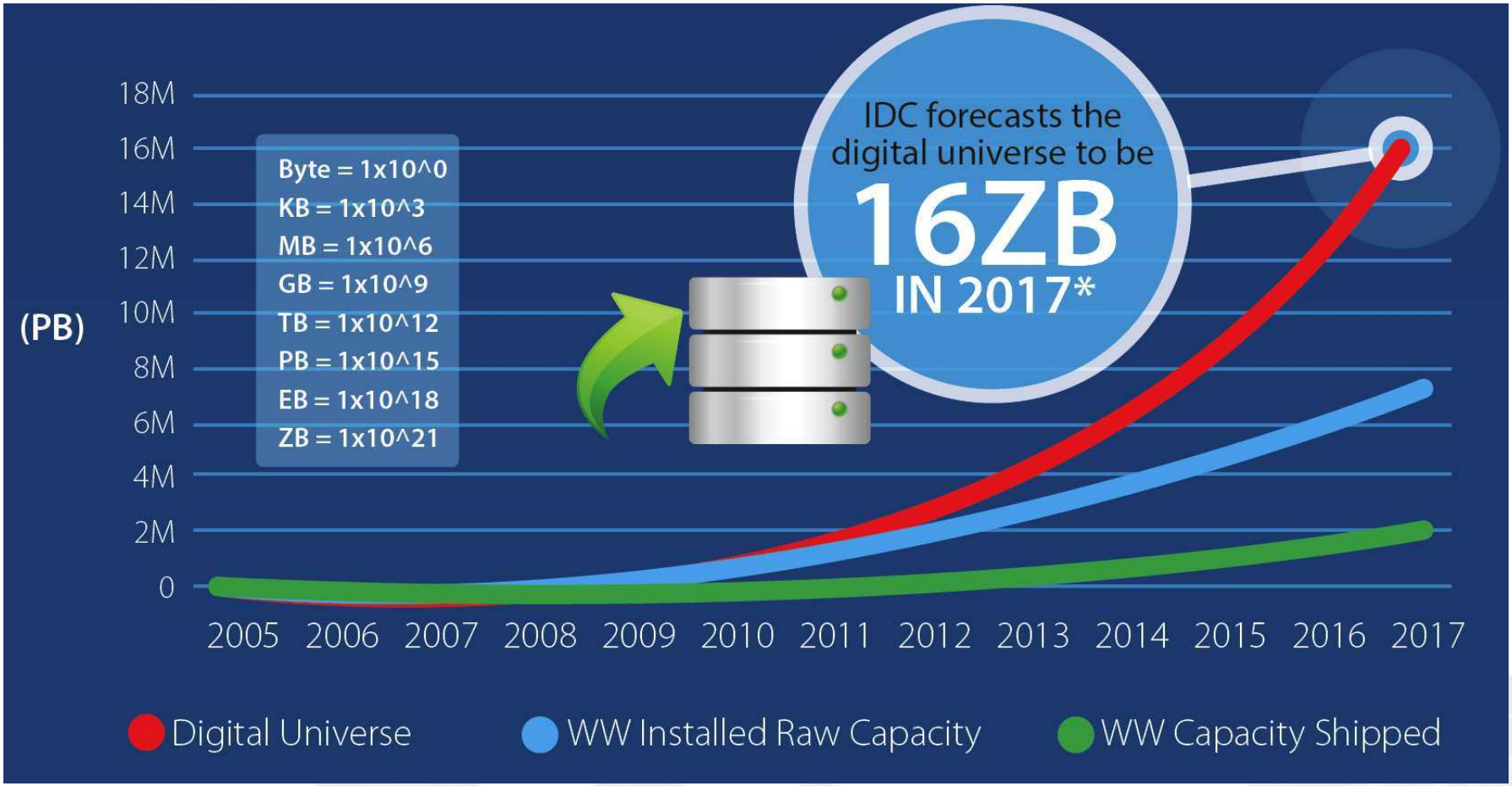
- Published 18 International Paper Publications
- 12 **Journals 4 Accepted** 8 Waiting for Reviews
- Completed **4 International** and **2 National Projects**
- **Working on 2 Patient**

- Personal Introduction
- **Problem Formulation**
  - Importance of Information
  - Importance of Digital Electronics
  - Problems and Solution
- HPC For Real World Problems
  - Hardware Architecture
  - Programming Models
  - Design Approach
- Projects

Importance of Information and Systems



Future of Information

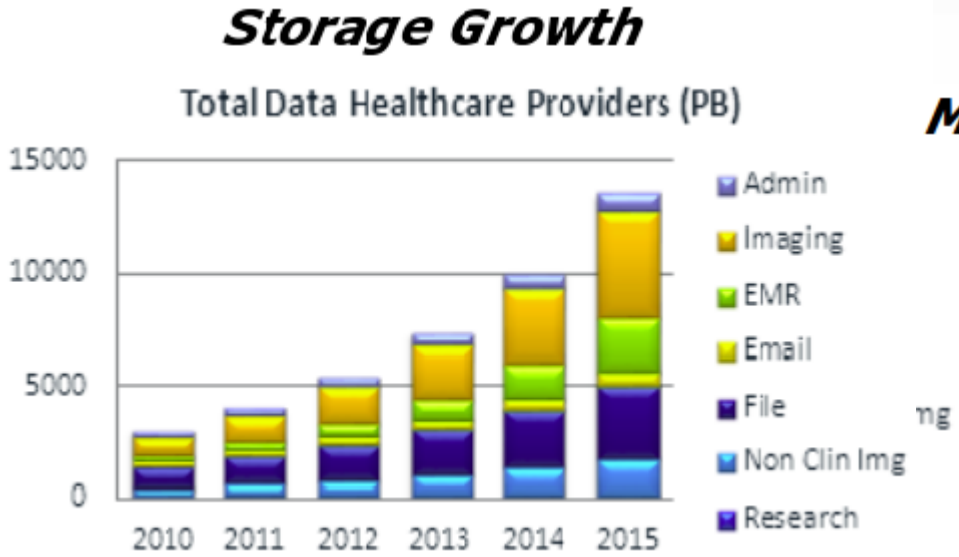


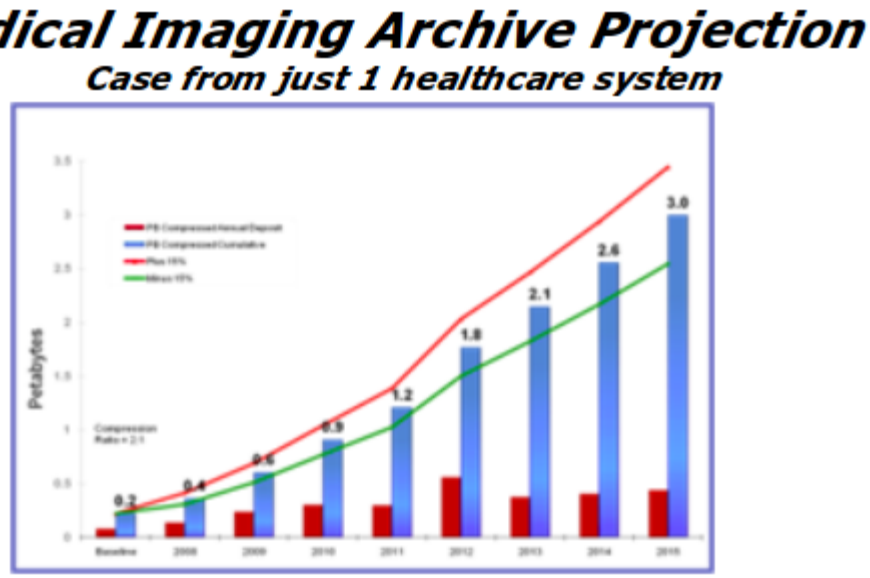
Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

Medical Data





Data Explosion Projected to reach 35 Zetabytes by 2020
(Mckinsey Global Institute Analysis)

Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

4 Vs of Big Data

Volume

Data Size

Variety

Data Structure rel

Velocity

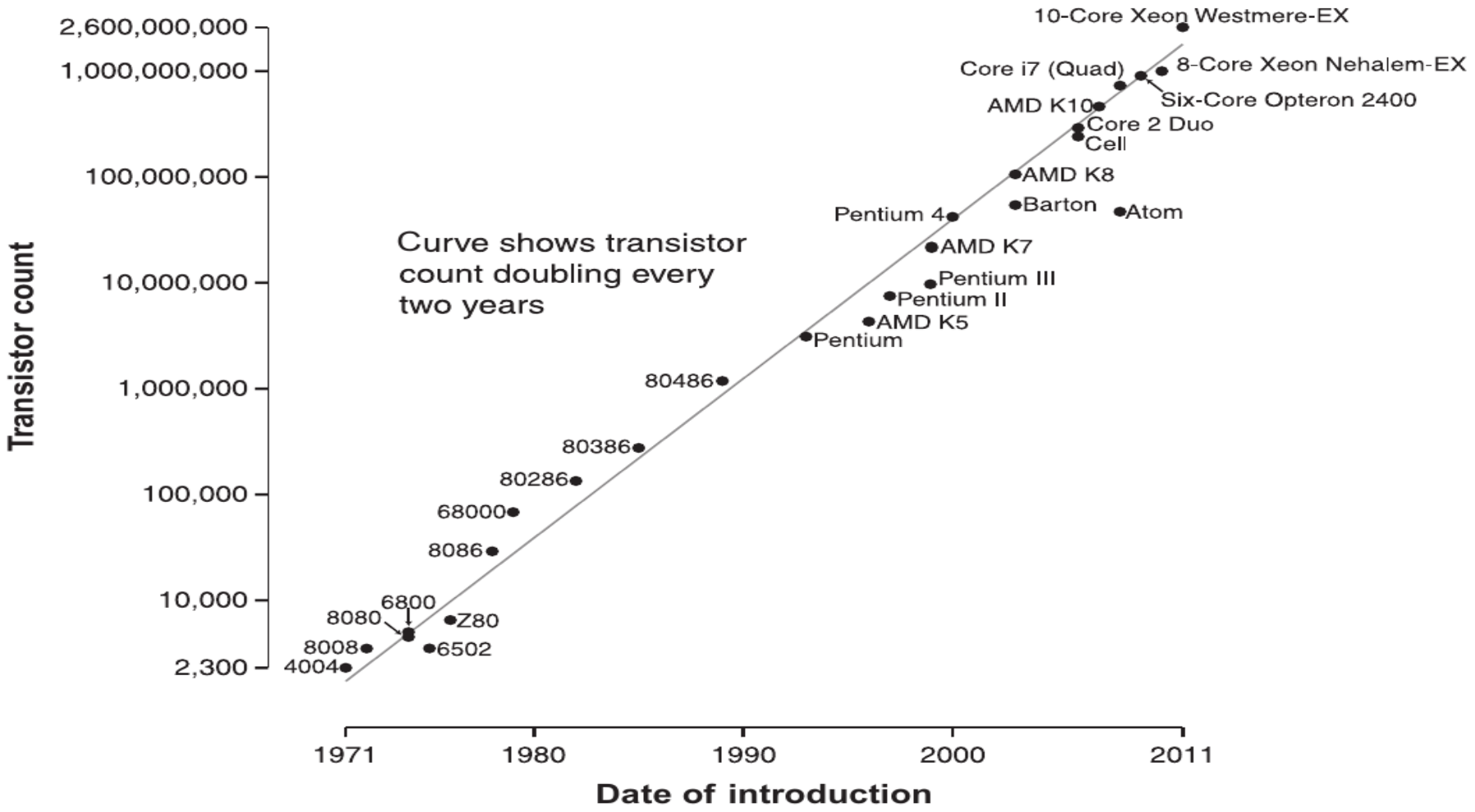
Real Time, Streaming

Value

Data for decision making

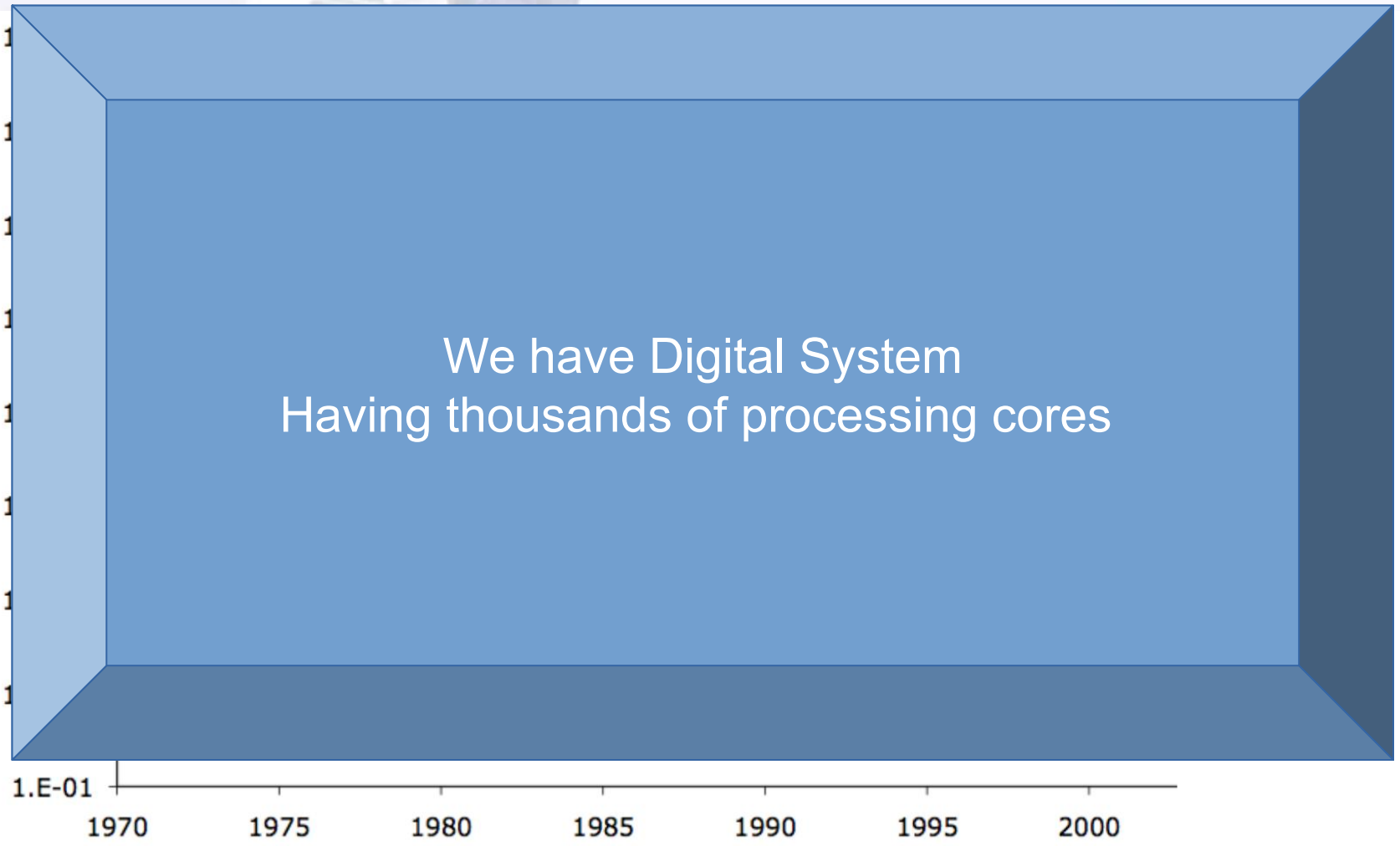
Past Present and Future of Digital Electronics

Moore's Law: microprocessor transistor counts doubling time of 18 months [1]



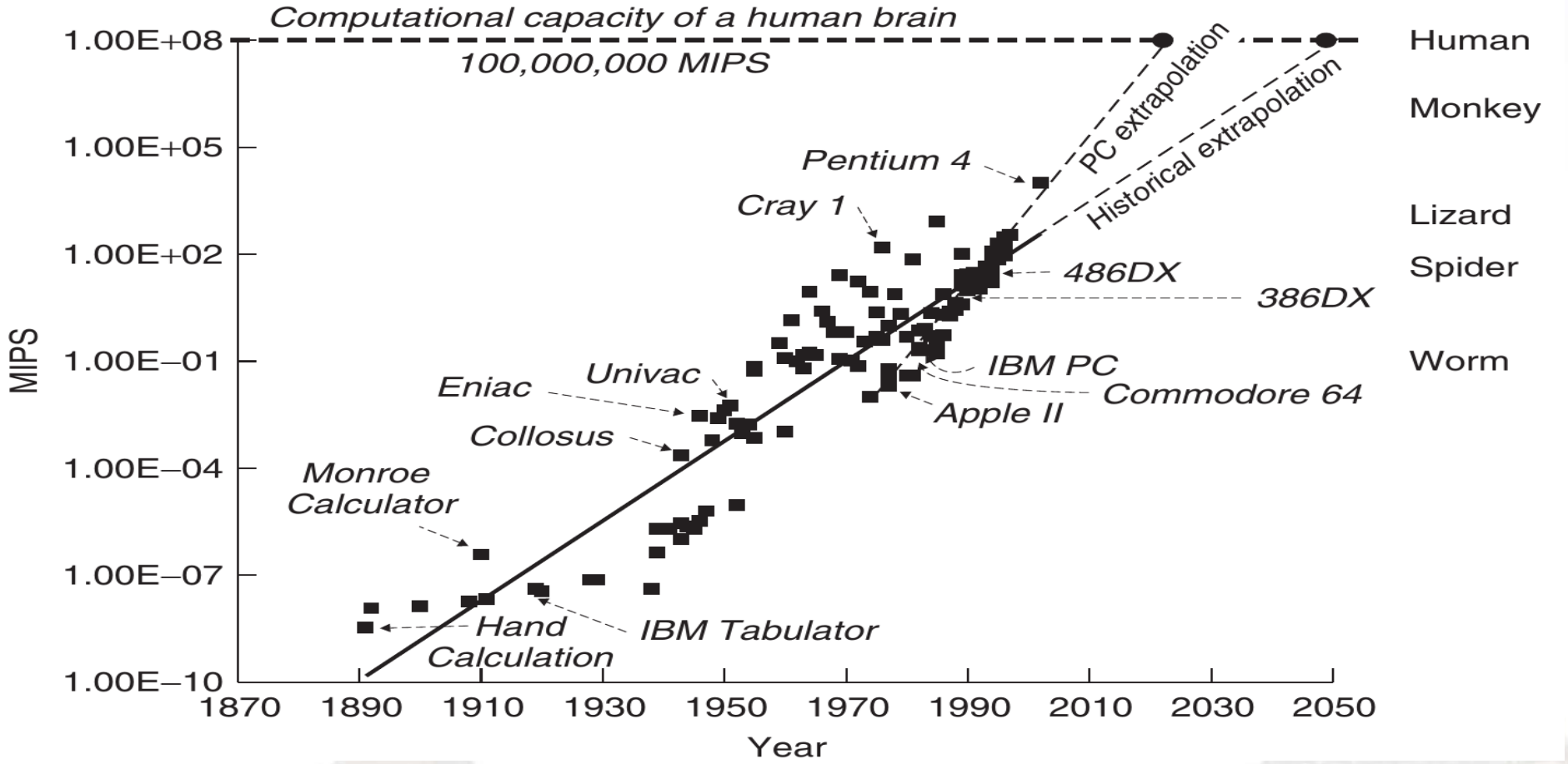
[1] G. E. Moore, "Cramming more components onto integrated circuits," Electronics, vol. 38, no. 8, April 1965.

Frequency and Power Issues



[1] G. E. Moore, "Cramming more components onto integrated circuits," Electronics, vol. 38, no. 8, April 1965.

Digital Systems: Future Trend



It is estimated that sometime between the years **2025 and 2050**, a **personal computers** will exceed the calculation power of a human brain.

Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

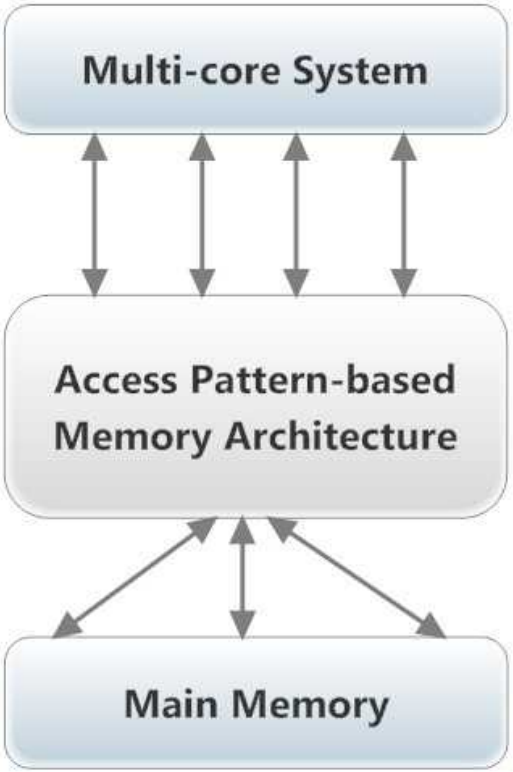
BSC Microsoft Research Centre

Problems of Digital Systems

- Power Wall
- Memory Wall
- Programming Wall
- Scalability
- Potability

Focus to Improve System Performance

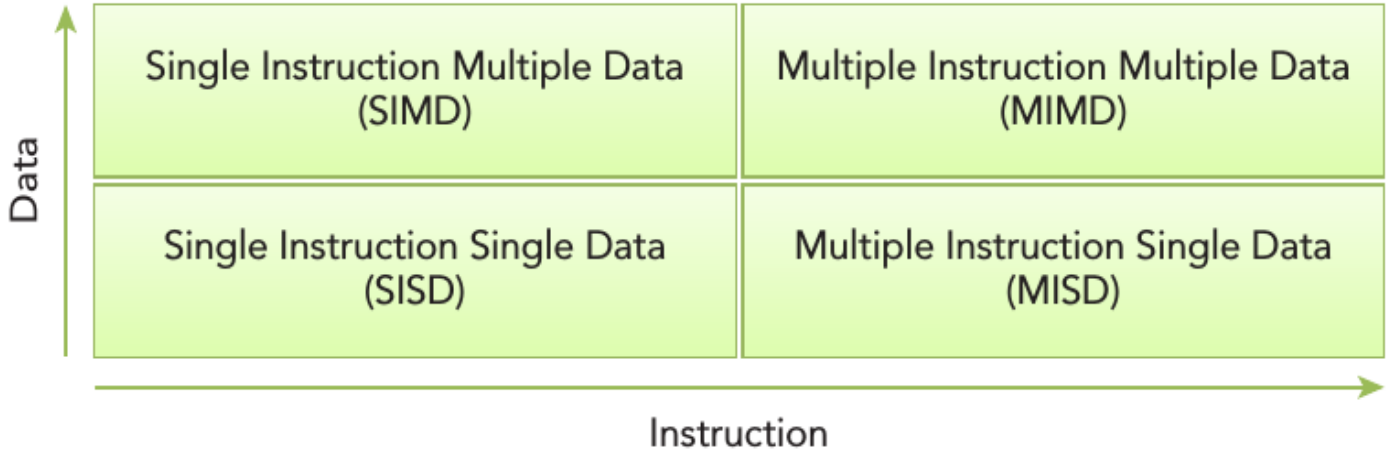
- Multi-core system
  - RISC
  - Vector & hardware accelerator cores
- Access Pattern-based Memory Architecture
  - Irregular/complex access patterns
- Programming Model



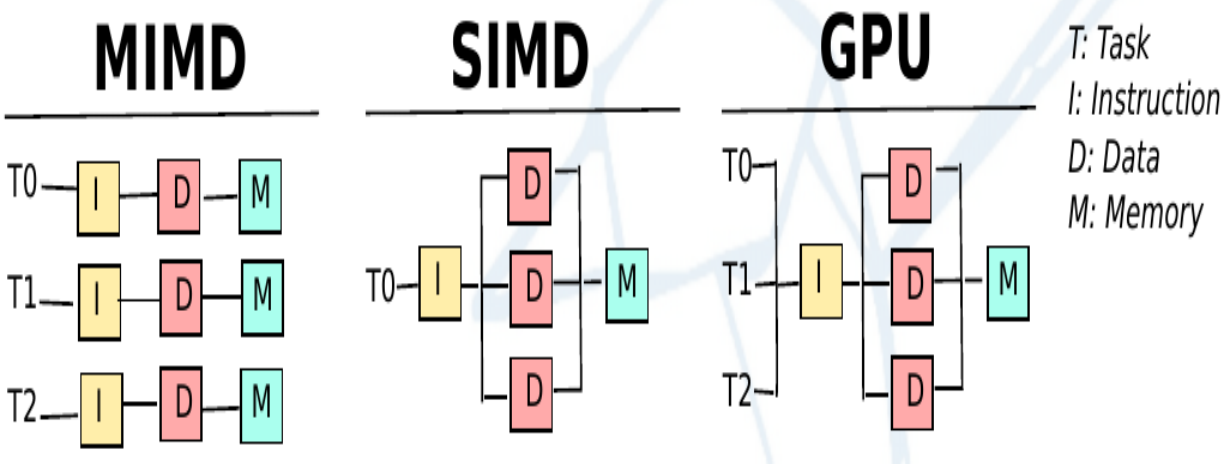
- Personal Introduction
- Problem Formulation
  - Importance of Information
  - Importance of Digital Electronics
  - Problems and Solution
- **HPC For Real World Problems**
  - Hardware Architecture
  - Programming Models
  - Design Approach
- Projects

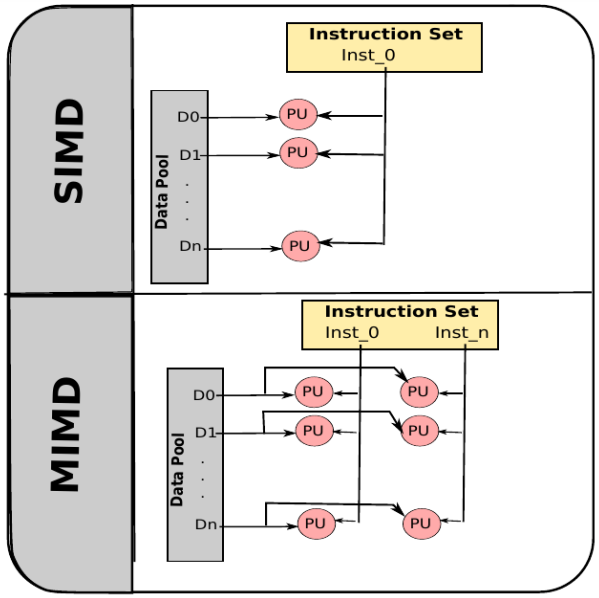
Processor Architectures

- Single Instruction Single Data (SISD)
- Single Instruction Multiple Data (SIMD)
- Multiple Instruction Single Data (MISD)
- Multiple Instruction Multiple Data (MIMD)

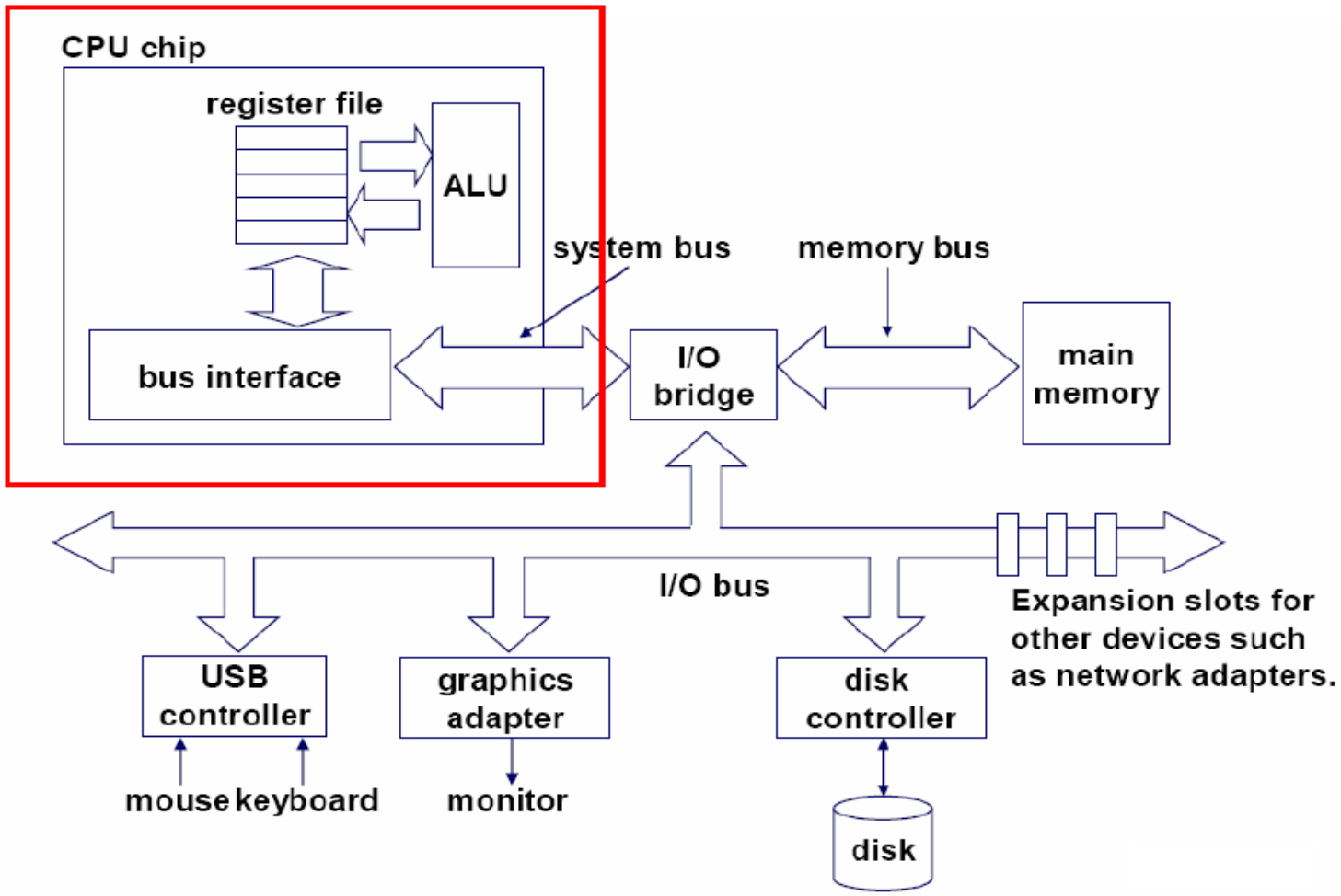


Basic Processor Architectures

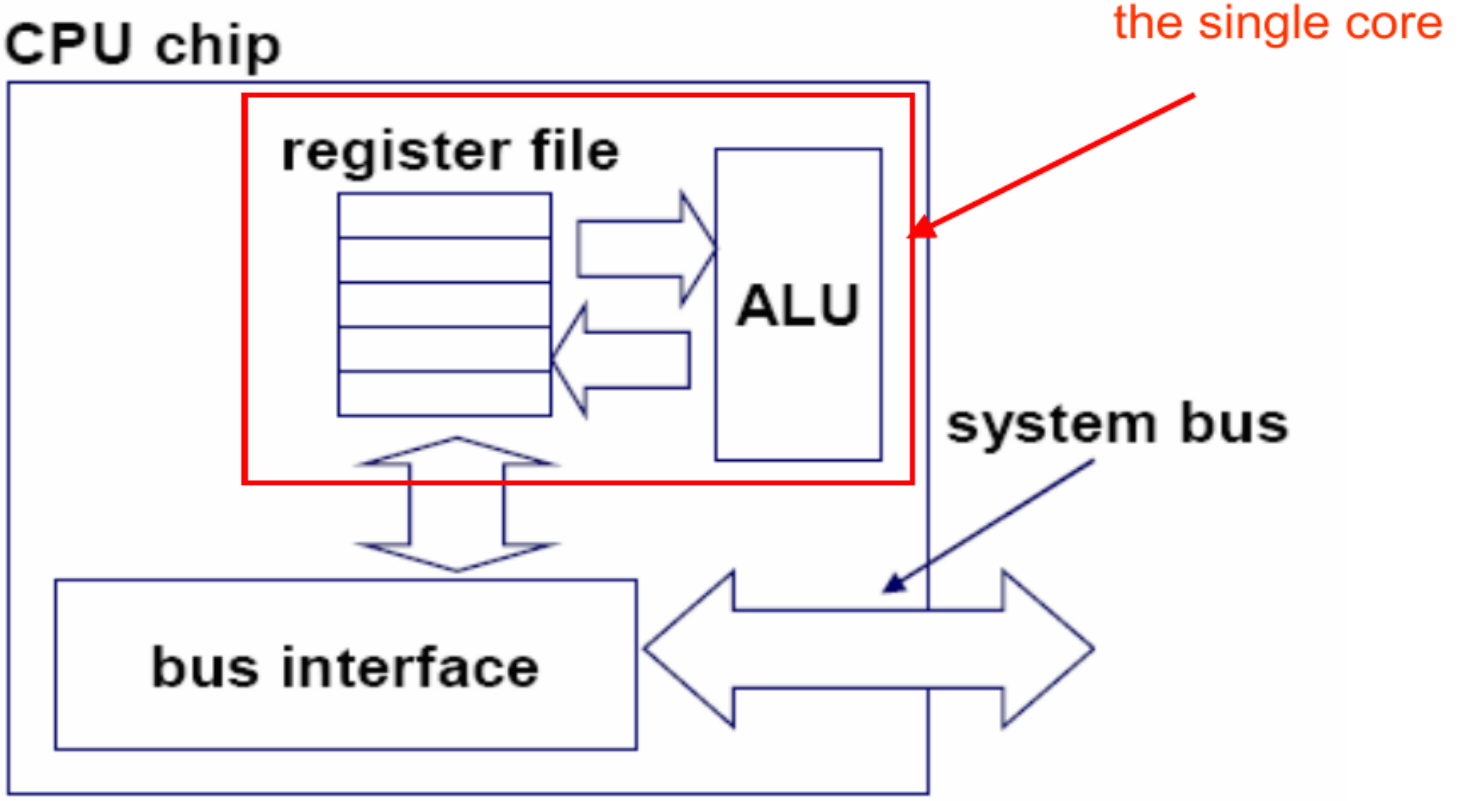




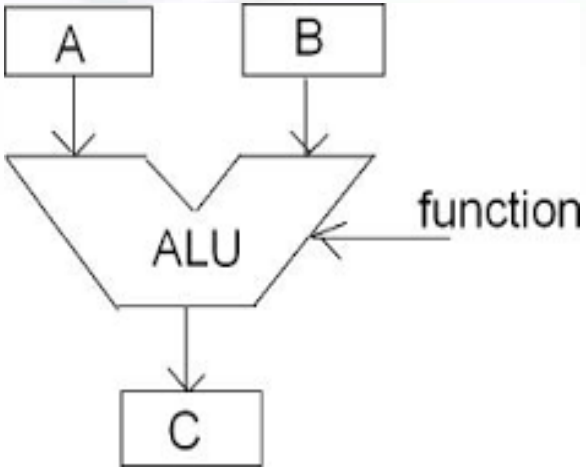
Uni Processor Architecture

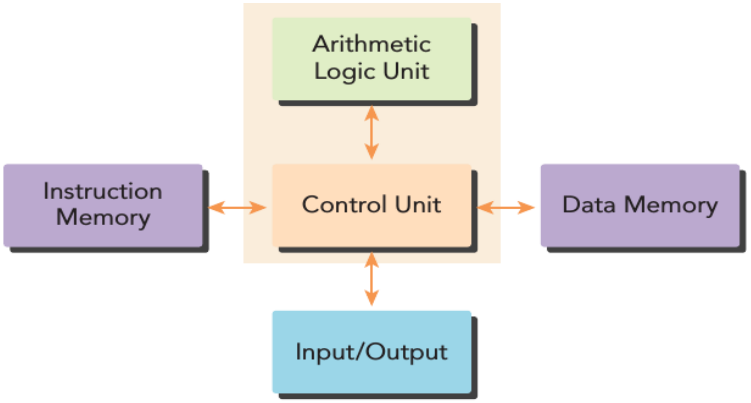


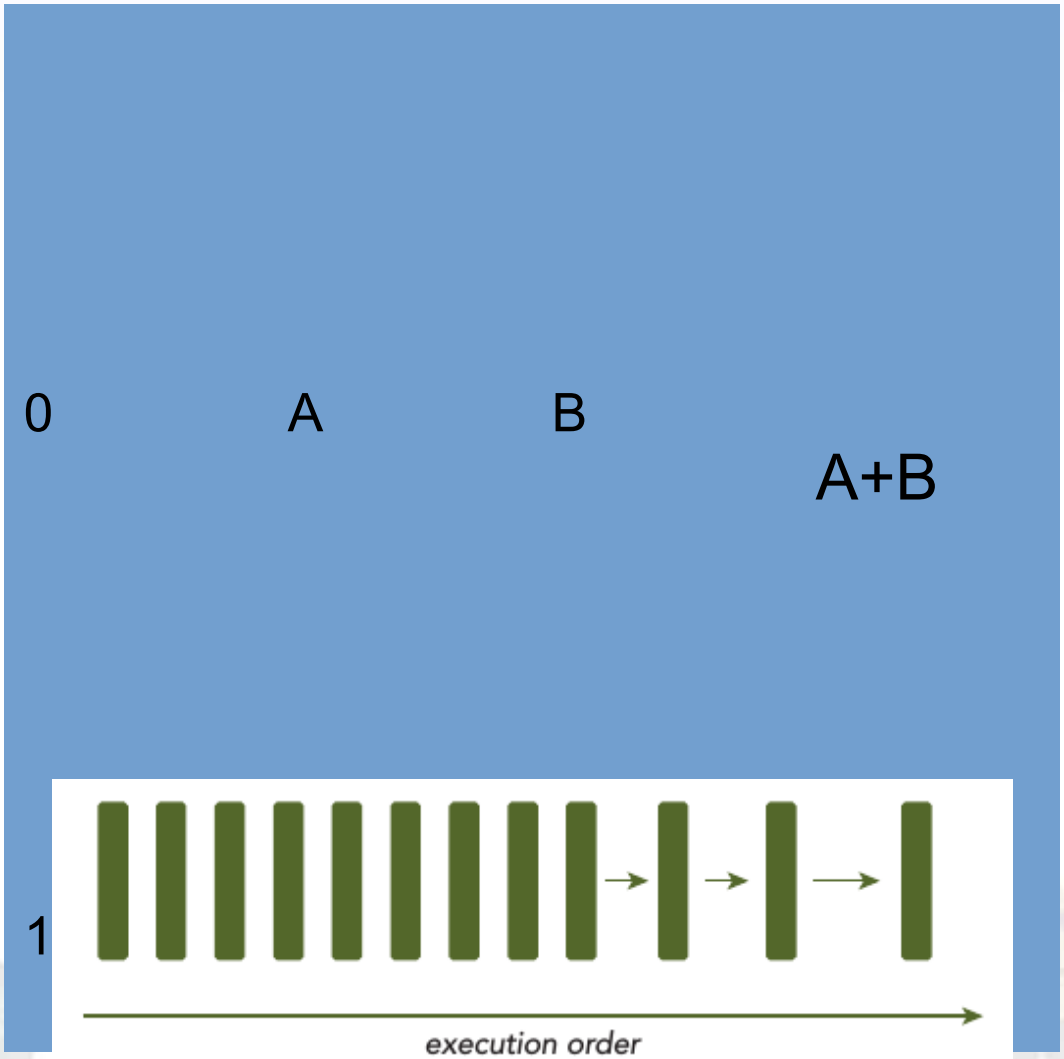
Uni Core



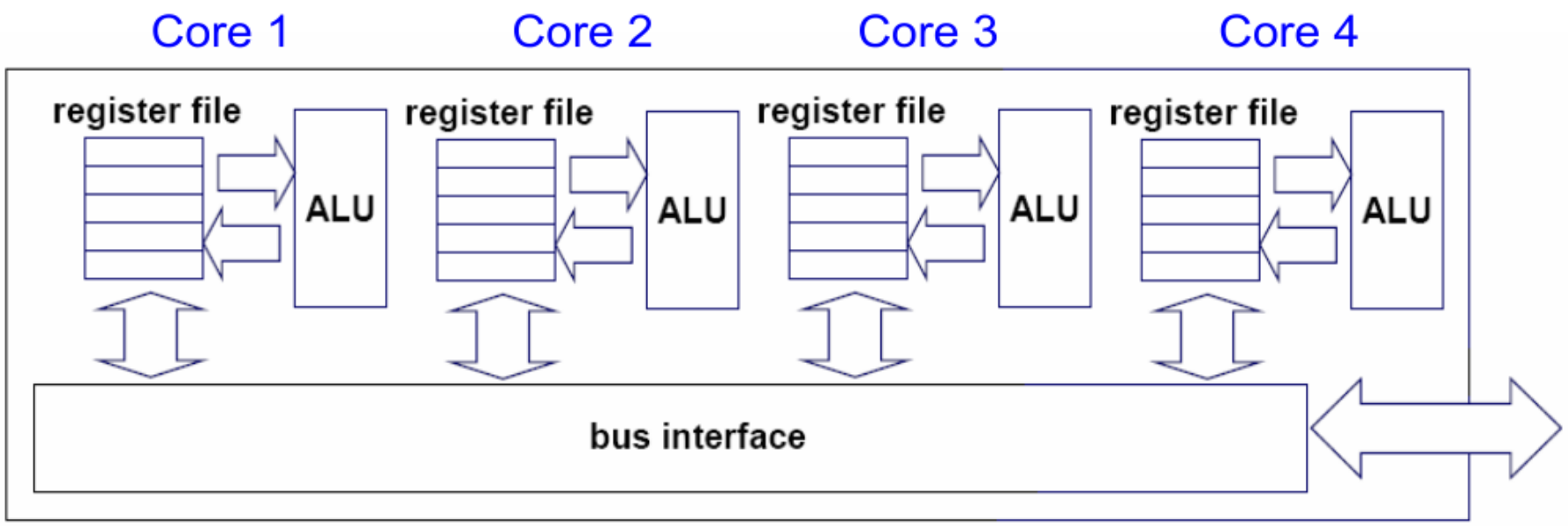
Basic introduction of Microprocessor







Multi-core Processor



Types of MIMD Architecture

Centralized shared-memory architectures or symmetric shared-memory multiprocrssors (SMP) or uniform memory access (UMA) architectures.

Distributed-memory multiprocessors.

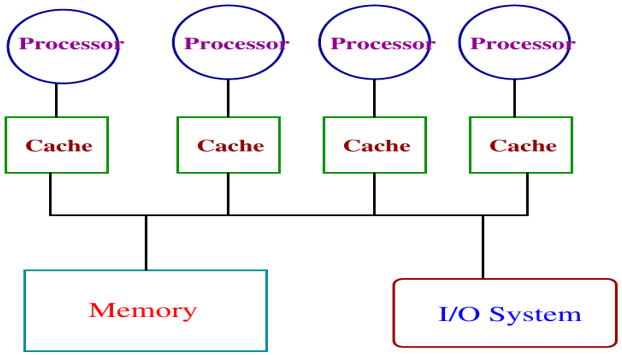
SMP: Shared Memory Processor

Small number of similar processors (at most a few dozen).

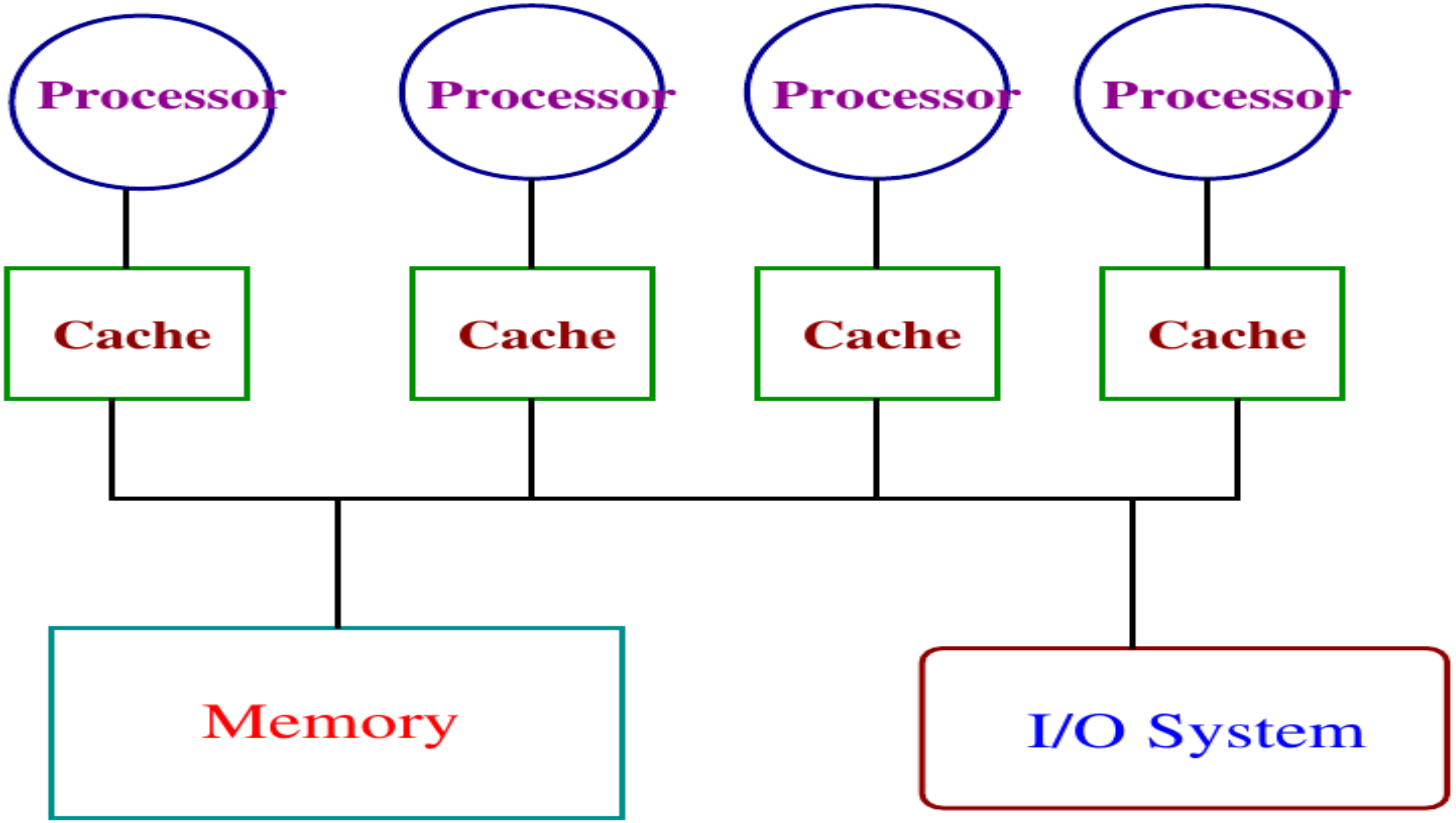
Each processor has a large cache.

A centralized memory (multiple banks) is shared through a memory bus.

Each memory location has identical access time from each processor.



SMP

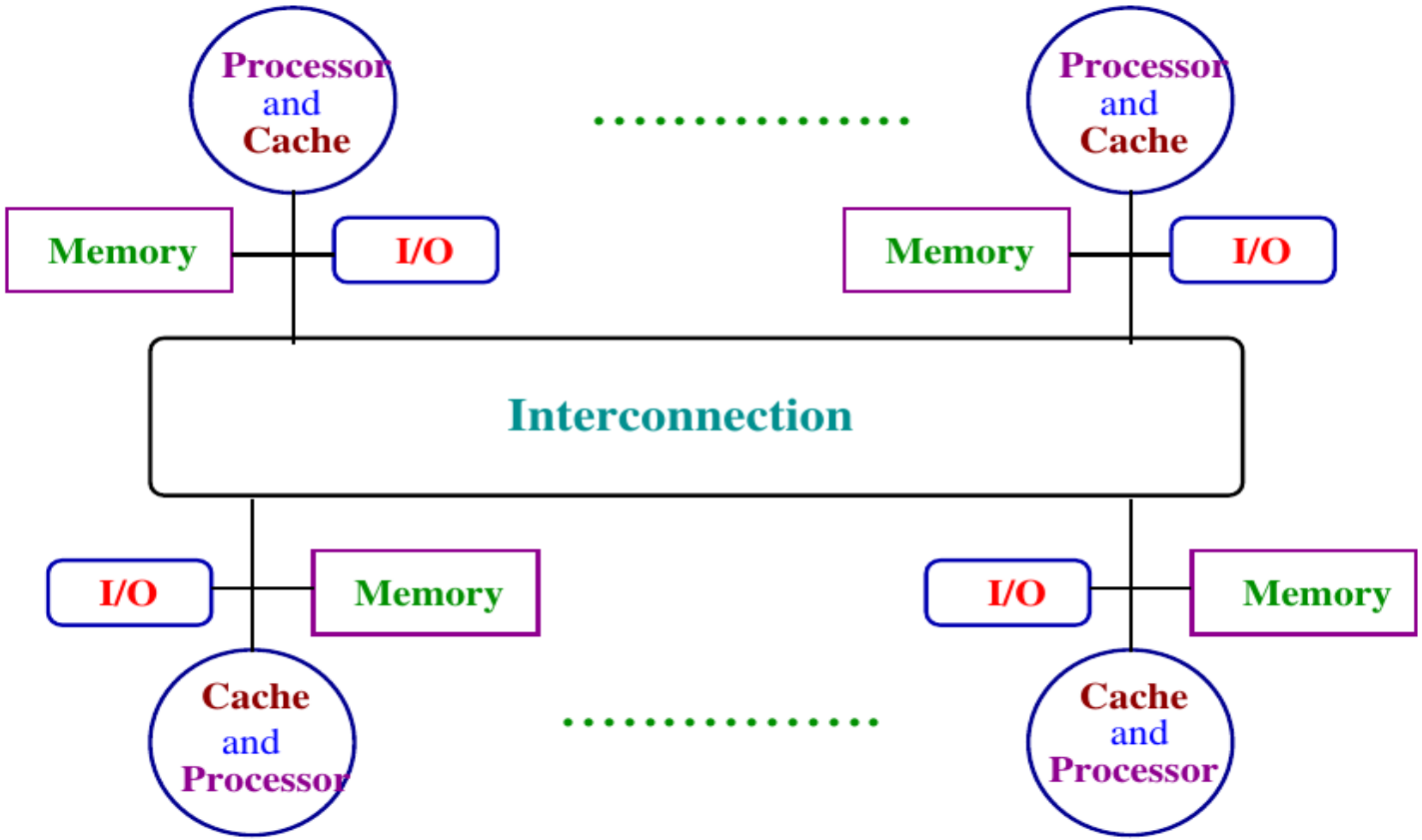

Processor
Processor
Processor
Processor
Cache
Cache
Cache
Cache
Memory
I/O System

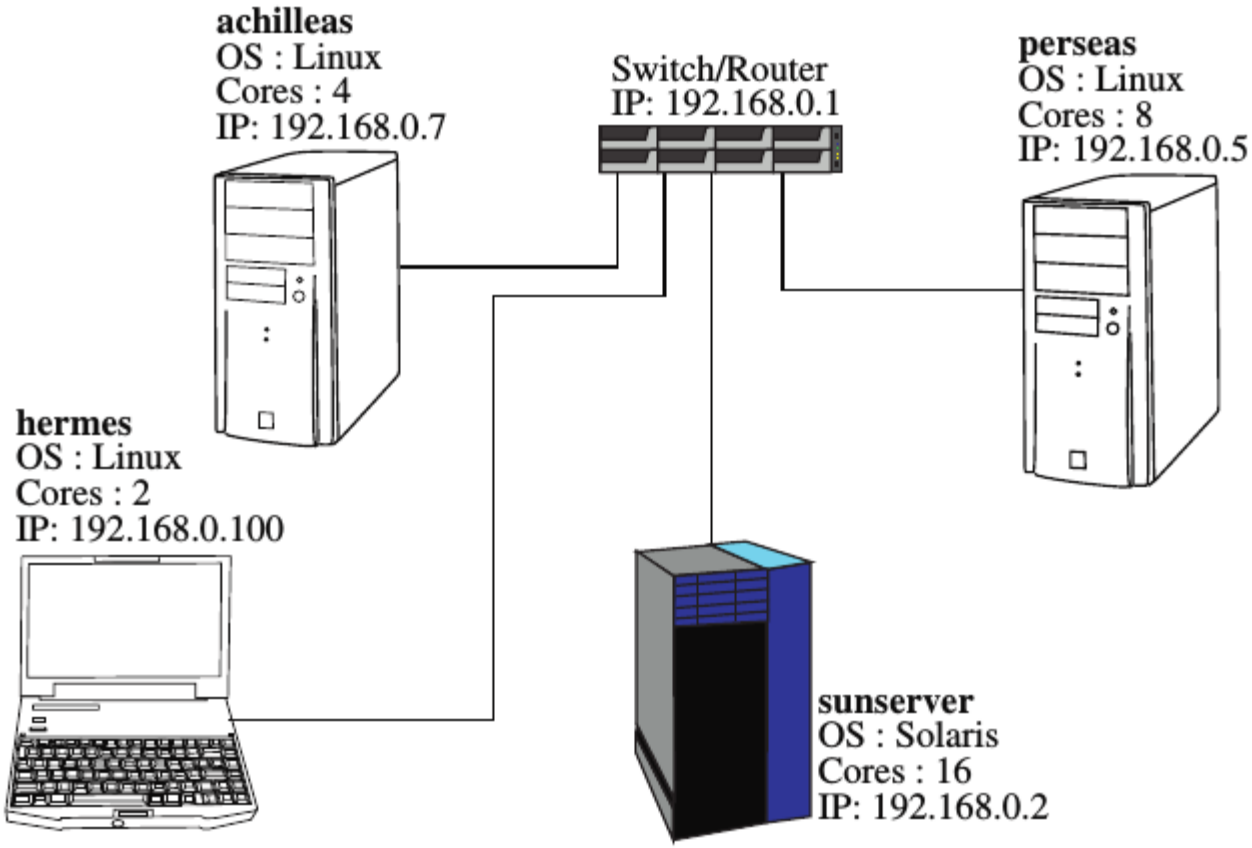
Distributed Memory

Larger processor count.

Memory is physically distributed among the processors for better bandwidth.

Connected through high-speed interconnection e.g. switches.





Distributed Memory

The bandwidth for the local memory is high and the latency is low.

But access to data present in the local memory of some other processor is complex and of high latency.

Memory Architecture and Programming

Large-scale multiprocessors have physically distributed memory with the processors.

There are essentially two different models of memory architectures and the corresponding models of communication.

Memory Arch..

- Memory is distributed with different processors to support higher bandwidth demand of larger number of processors.
- Any processor can access a location of physically distributed memory (with proper access permission).
- This is called distributed shared-memory architecture (DSM) also known as NUMA (nonuniform memory access).

Shared Memory Architecture

The same physical address on two processors refers to the same location in memory.

The communication is through the shared address space.

The other alternative is that every processor has its private address space.

Each processor is essentially a separate computer - this is called a multicomputer model or clusters.

The communication is by message-passing.

Such cluster of processors use standardized or customized interconnect for communication

Message Passing: Memory Sharing

If a processor wants to access (or process) some data in a remote memory, it sends a message (similar to remote procedure call (RPC)).

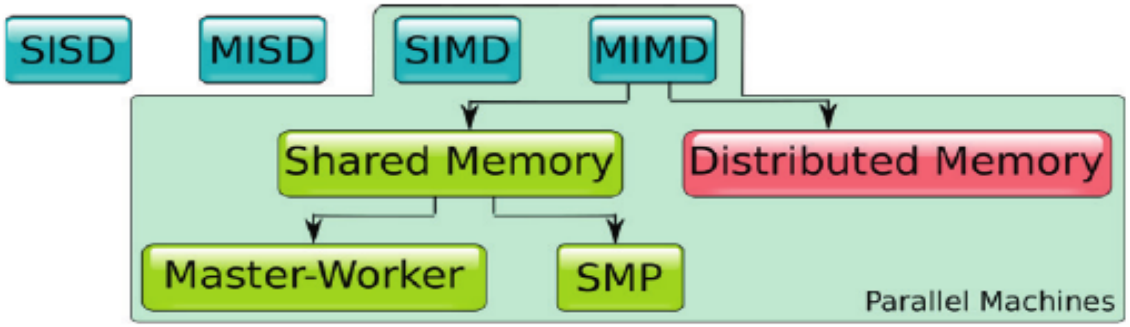
The destination processor receives it (polling or interrupt), performs the operation, and returns the result through a reply message.

The message passing is synchronous - the initiating processor after sending the requests waits for the reply.

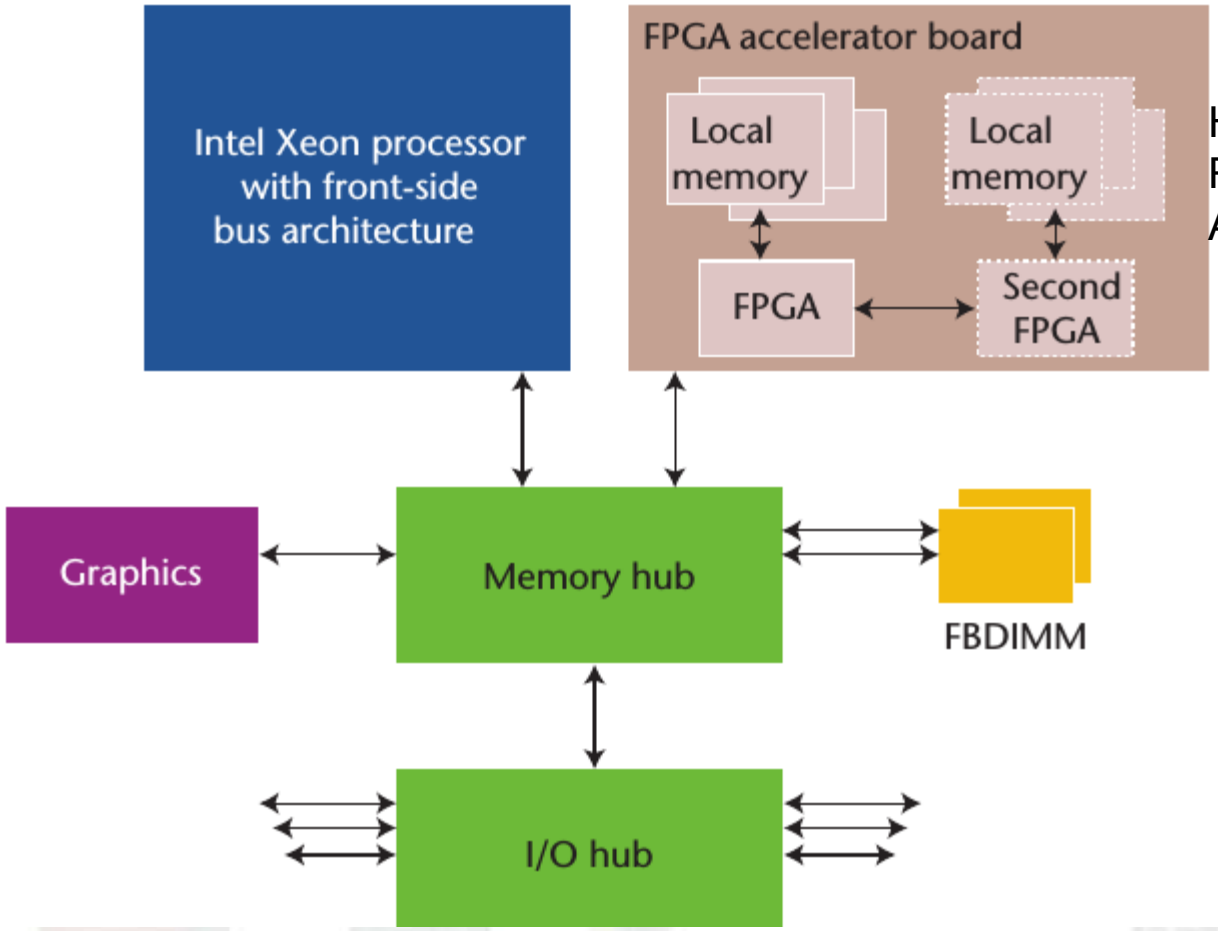
Available Computer Architectures

There are currently two trends in utilizing the increased transistor count afforded by miniaturization and advancements in semiconductor materials:

- Increase the **on-chip core count**,
  - Combined with augmented specialized SIMD instruction sets (e.g., SSE and its subsequent versions, MMX, AESNI, etc.) and larger caches.
  - This is best exemplified by Intel's x86 line of CPUs and the Intel Xeon Phi coprocessor.
- **Combine heterogeneous cores** in the same package,
  - Typically CPU and GPU ones, each optimized for a different type of task.
  - This is best exemplified by AMD's line of Accelerated Processing Unit (APU) chips. Intel is also offering OpenCL-based computing on its line of CPUs with integrated graphics chips.



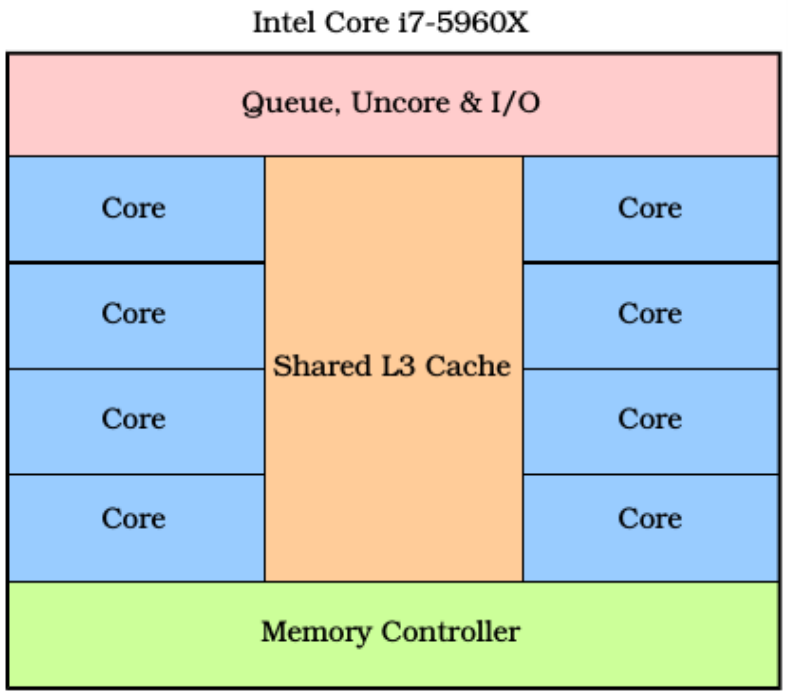
Reconfigurable Accelerators



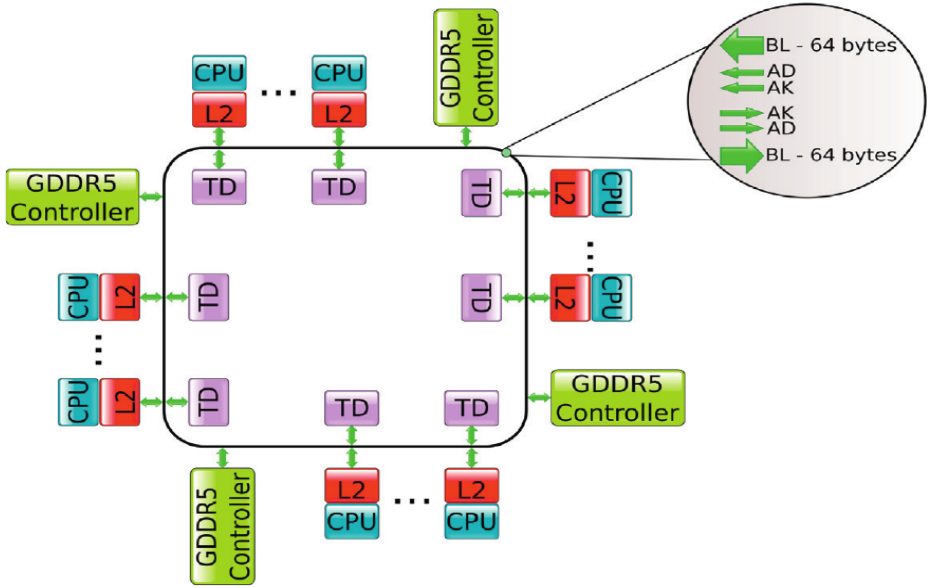
HPC Reconfigurable Architecture

CPU: Intel Processor

CPUs employ large on-chip (and sometimes multiple) memory caches, few complex (e.g., pipelined) arithmetic and logical processing units (ALUs), and complex instruction decoding and prediction hardware to avoid stalling while waiting for data to arrive from the main memory.



Intel Xeon Phi



- A Super-scalar Architecture
- Xeon Phi comes equipped upto 72 x86 cores that are heavily customized Pentium cores.
- The customizations include the ability to handle four threads at the same time.
- The coherency is managed by distributed tag directories (TDs)

Intel Super Scalar: A Many Core Architecture

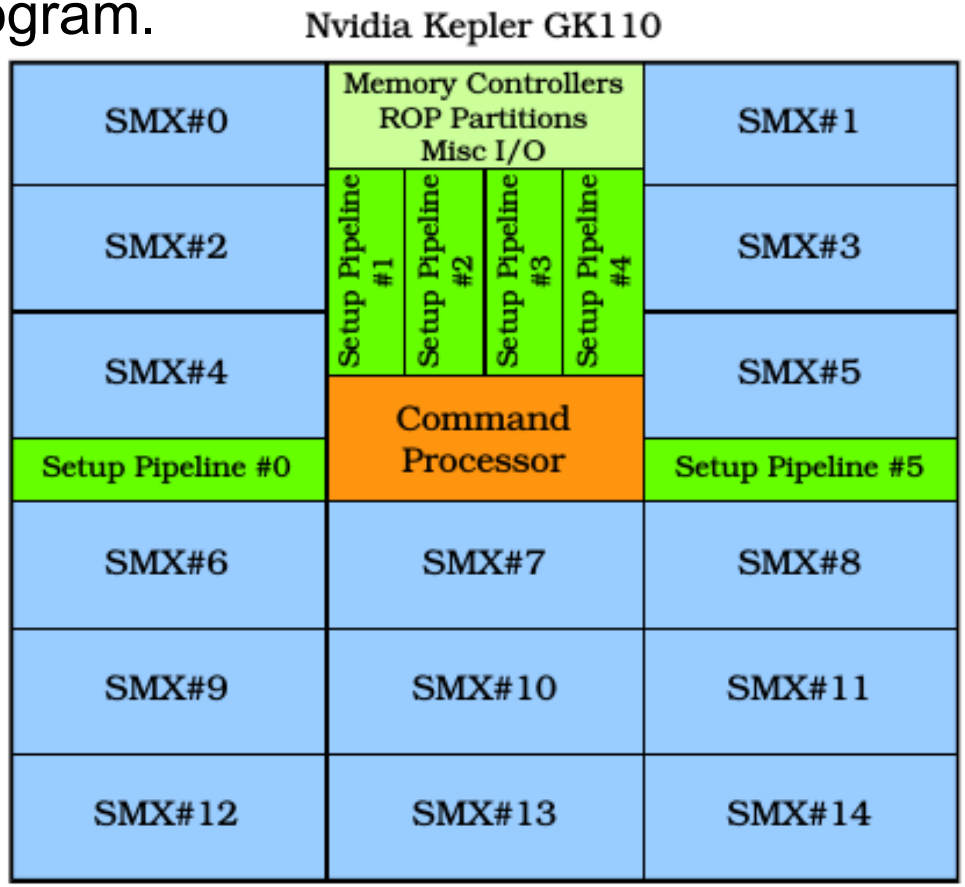
Processor Number	Availability	# of Cores/# of Threads	Clock Speed	Max TDP/Power	Memory Types	Fabric	L2 Cache
Intel® Xeon Phi™ Processor 7250 (16GB, 1.40 GHz, 68 core)	Now	68/272	1.4 GHz	215 W	DDR4-2400	No	34 MB
Intel® Xeon Phi™ Processor 7230 (16GB, 1.30 GHz, 64 core)	Now	64/256	1.3 GHz	215 W	DDR4-2400	No	32 MB
Intel® Xeon Phi™ Processor 7210 (16GB, 1.30 GHz, 64 core)	Now	64/256	1.3 GHz	215 W	DDR4-2133	No	32 MB
Intel® Xeon Phi™ Processor 7290 (16GB, 1.50 GHz, 72 core)	Sept. 2016	72/288	1.5 GHz	245 W	DDR4-2400	No	36 MB
Intel® Xeon Phi™ Processor 7290F (16GB, 1.50 GHz, 72 core)	Oct. 2016	72/288	1.5 GHz	260 W	DDR4-2400	Yes	36 MB
Intel® Xeon Phi™ Processor 7250F (16GB, 1.40 GHz, 68 core)	Oct. 2016	68/272	1.4 GHz	230 W	DDR4-2400	Yes	34 MB
Intel® Xeon Phi™ Processor 7230F (16GB, 1.30 GHz, 64 core)	Oct. 2016	64/256	1.3 GHz	230 W	DDR4-2400	Yes	32 MB
Intel® Xeon Phi™ Processor 7210F (16GB, 1.30 GHz, 64 core)	Oct. 2016	64/256	1.3 GHz	230 W	DDR4-2133	Yes	32 MB

Graphics Processing Unit (GPU) and CPU

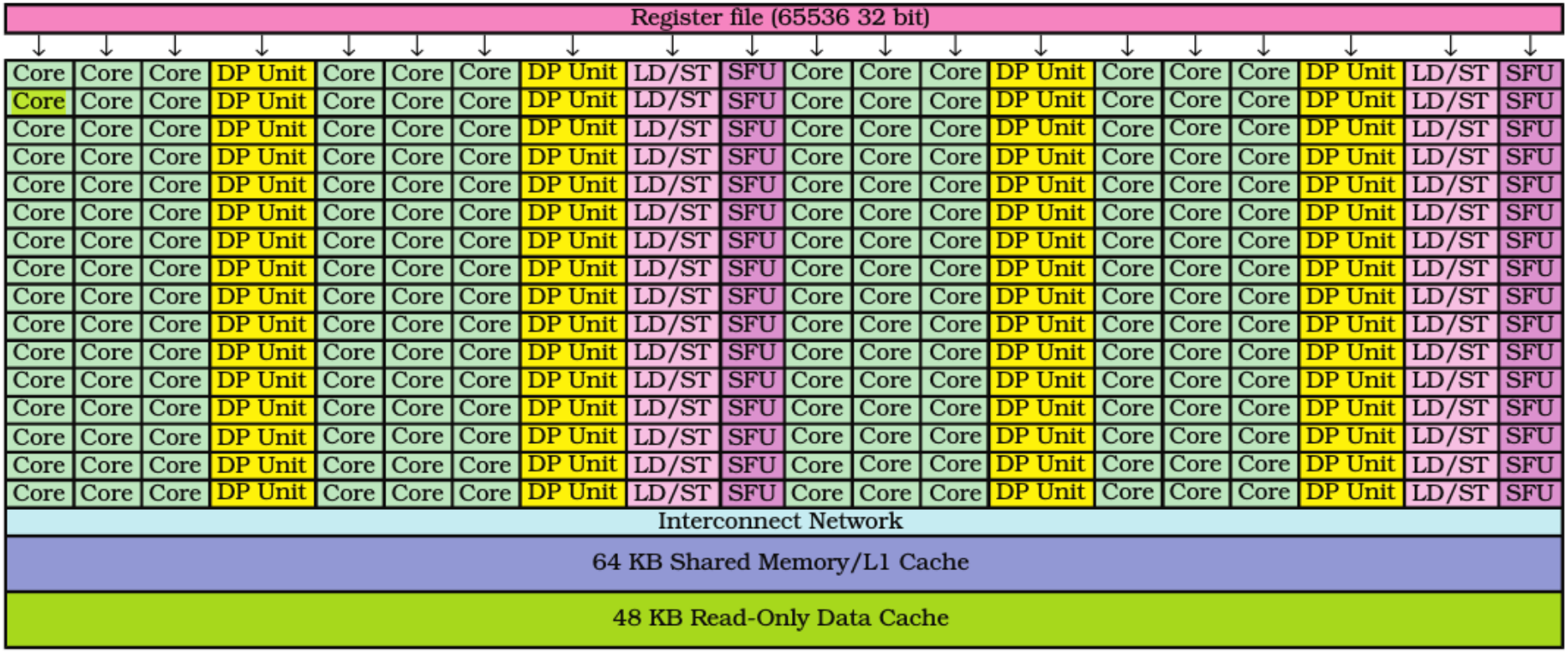
- **GPUs** have been developed as a means of processing massive amount of graphics data very quickly, before they are placed in the card's display buffer.
- Their design envelope dictated a layout that departed from the one traditionally used by conventional CPUs.
- **GPU** uses small on-chip caches with a big collection of simple ALUs capable of parallel operation, since data reuse is typically small for graphics processing and programs are relatively simple. In order to feed the multiple cores on a GPU, designers also dedicated very wide, fast memory buses for fetching data from the GPU's main memory.

Nvidia Graphics Processing Unit (GPU)

- SM, SMM SMX (Streaming Multiprocessors): Single SMX contains 192 cores executes in SIMD fashion
- Each SMX can run its own program.
- CUDA and OpenACC Programming Models



GPU SMX Internal Architecture



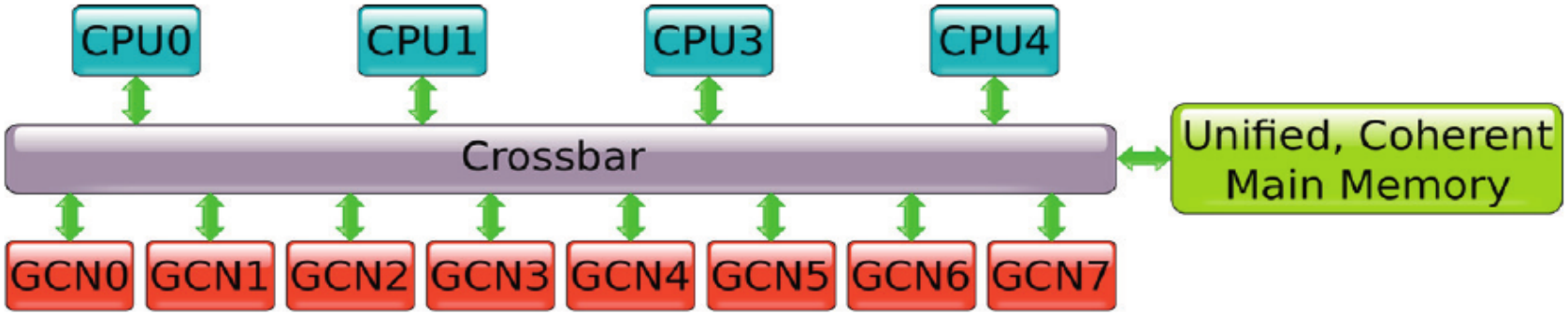
192 **Core** : single-precision cores
64 **DP Unit** : double -precision cores
32 **LD/ST** : load/store units
32 **SFU** : Special Function Units

High Performance Accelerators

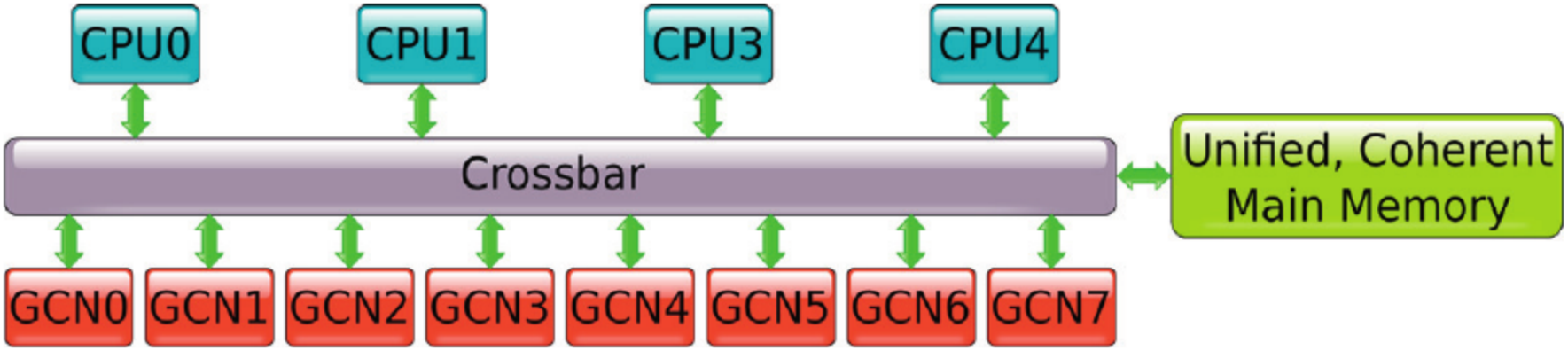
NVIDIA Tesla Family Specification Comparison				
	Tesla P100	Tesla K80	Tesla K40	Tesla M40
Stream Processors	3584	2 x 2496	2880	3072
Core Clock	1328MHz	562MHz	745MHz	948MHz
Boost Clock(s)	1480MHz	875MHz	810MHz, 875MHz	1114MHz
Memory Clock	1.4Gbps HBM2	5Gbps GDDR5	6Gbps GDDR5	6Gbps GDDR5
Memory Bus Width	4096-bit	2 x 384-bit	384-bit	384-bit
Memory Bandwidth	720GB/sec	2 x 240GB/sec	288GB/sec	288GB/sec
VRAM	16GB	2 x 12GB	12GB	12GB
Half Precision	21.2 TFLOPS	8.74 TFLOPS	4.29 TFLOPS	6.8 TFLOPS
Single Precision	10.6 TFLOPS	8.74 TFLOPS	4.29 TFLOPS	6.8 TFLOPS
Double Precision	5.3 TFLOPS (1/2 rate)	2.91 TFLOPS (1/3 rate)	1.43 TFLOPS (1/3 rate)	213 GFLOPS (1/32 rate)
GPU	GP100 (610mm2)	GK210	GK110B	GM200
Transistor Count	15.3B	2 x 7.1B(?)	7.1B	8B
TDP	300W	300W	235W	250W
Cooling	N/A	Passive	Active/Passive	Passive

AMD GPU

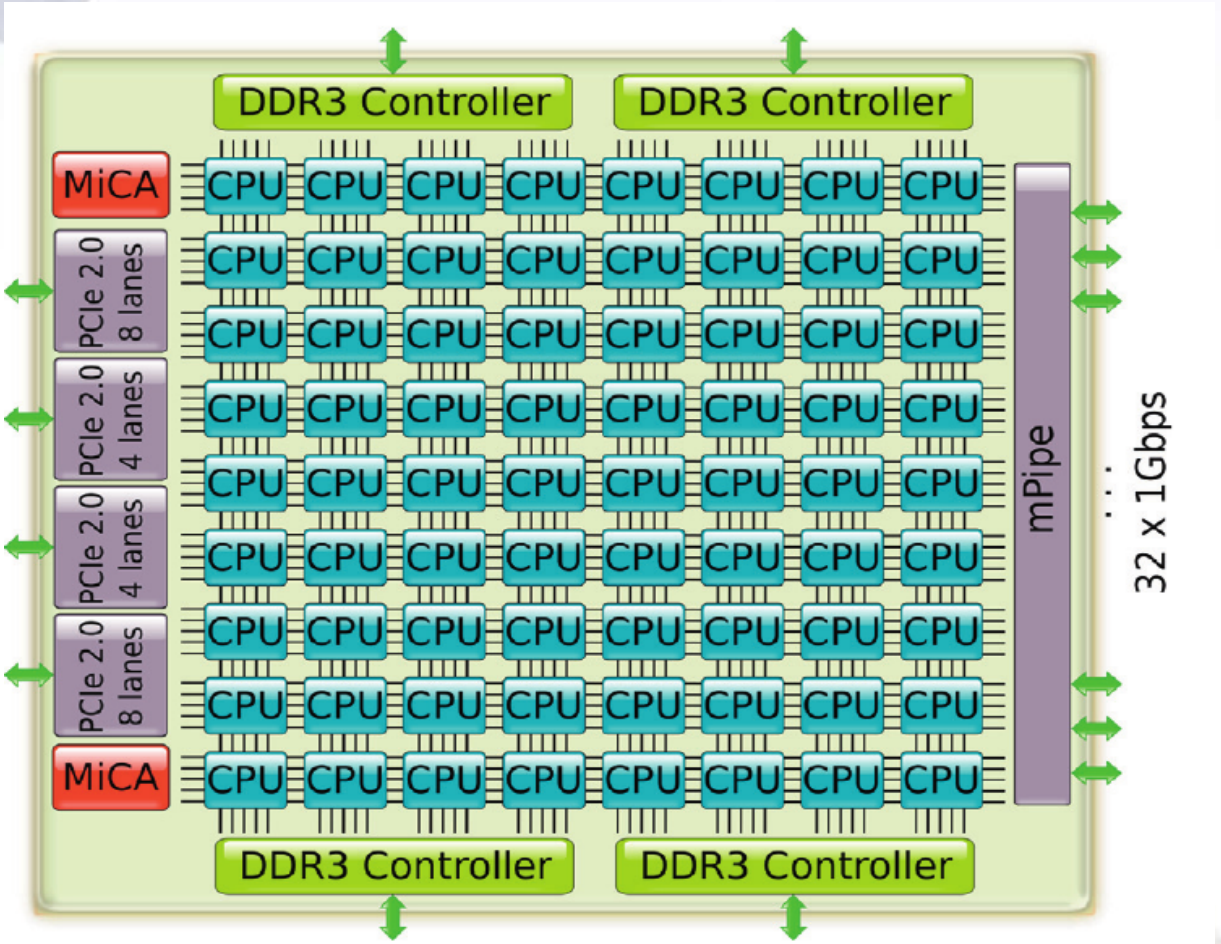
- AMD's APU chips implement the Heterogeneous System Architecture (HSA).
- The significant of AMD GPU is the unification of the memory spaces of the CPU and GPU cores. This means that there is no communication overhead associated with assigning workload to the GPU cores, nor any delay in getting the results back.
- This also removes one of the major hassles in GPU programming, which is the explicit (or implicit, based on the middleware available) data transfers that need to take place.



- The HSA architecture identifies two core types:
- The Latency Compute Unit (LCU), which is a generalization of a CPU. A LCU supports both its native CPU instruction set and the HSA intermediate language (HSAIL) instruction set.
- The Throughput Compute Unit (TCU), which is a generalization of a GPU. A TCU supports only the HSAIL instruction set. TCUs target efficient parallel execution.

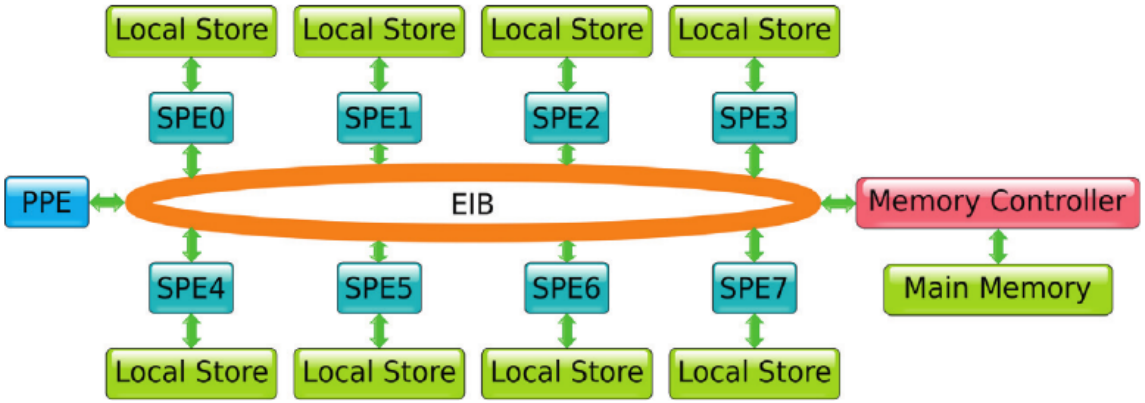


TILERA'S TILE-GX8072



Power PC

- Master Core: 64-bit PowerPC core also called the Power Processing Element.
- Worker Core: Synergistic Processing Element SPE having 128-bit vector processors.
- Own SIMD instruction set.



Programming Model

- C/C++
- OpenMP
- OpenACC
- OpenCL
- HLS

Parallel Programming Paradigms --Various Methods

- There are many methods of programming parallel computers. Two of the most common are message passing and data parallel.
  - Message Passing - the user makes calls to libraries to explicitly share information between processors.
  - Data Parallel - data partitioning determines parallelism
  - Shared Memory - multiple processes sharing common memory space
  - Remote Memory Operation - set of processes in which a process can access the memory of another process without its participation
  - Threads - a single process having multiple (concurrent) execution paths
  - Combined Models - composed of two or more of the above.

Note: these models are machine/architecture independent, any of the models can be implemented on any hardware given appropriate operating system support. An effective implementation is one which closely matches its target hardware and provides the user ease in programming.

Parallel Programming Paradigms: Message Passing

- The message passing model is defined as:
  - set of processes using only local memory
  - processes communicate by sending and receiving messages
  - data transfer requires cooperative operations to be performed by each process (a send operation must have a matching receive)
- Programming with message passing is done by linking with and making calls to libraries which manage the data exchange between processors. Message passing libraries are available for most modern programming languages.

Parallel Programming Paradigms: Data Parallel

- The data parallel model is defined as:
  - Each process works on a different part of the same data structure
  - Commonly a Single Program Multiple Data (SPMD) approach
  - Data is distributed across processors
  - All message passing is done invisibly to the programmer
  - Commonly built "on top of" one of the common message passing libraries
- Programming with data parallel model is accomplished by writing a program with data parallel constructs and compiling it with a data parallel compiler.
- The compiler converts the program into standard code and calls to a message passing library to distribute the data to all the processes.

Architectures Programming

- Single Instruction Multiple Data (SIMD) architectures can exploit significant data-level parallelism (matrix, images, signal processing), fetching one instruction per data operation.
- SIMD allows programmer to think sequentially and achieve parallel speed-ups
- Multiple Instruction Multiple Data (MIMD) architecture relies on a number of independent processors that can operate upon separate data concurrently and asynchronously.
- Hence each processor has its own program memory or has access to program memory.
- GPU exploits task-level parallelism, using multi-threading to hide memory latency.
- It has many functional units, as opposed to a few deeply pipelined units like a vector processor.

Steps for Creating a Parallel Program

1. If you are starting with an existing serial program, debug the serial code completely
2. Identify the parts of the program that can be executed concurrently:
   - Requires a thorough understanding of the algorithm
   - Exploit any inherent parallelism which may exist.
   - May require restructuring of the program and/or algorithm. May require an entirely new algorithm.
3. Decompose the program:
   - Functional Parallelism
   - Data Parallelism
   - Combination of both
4. Code development
   - Code may be influenced/determined by machine architecture
   - Choose a programming paradigm
   - Determine communication
   - Add code to accomplish task control and communications
5. Compile, Test, Debug
6. Optimization
   1. Measure Performance
   2. Locate Problem Areas
   - Improve them

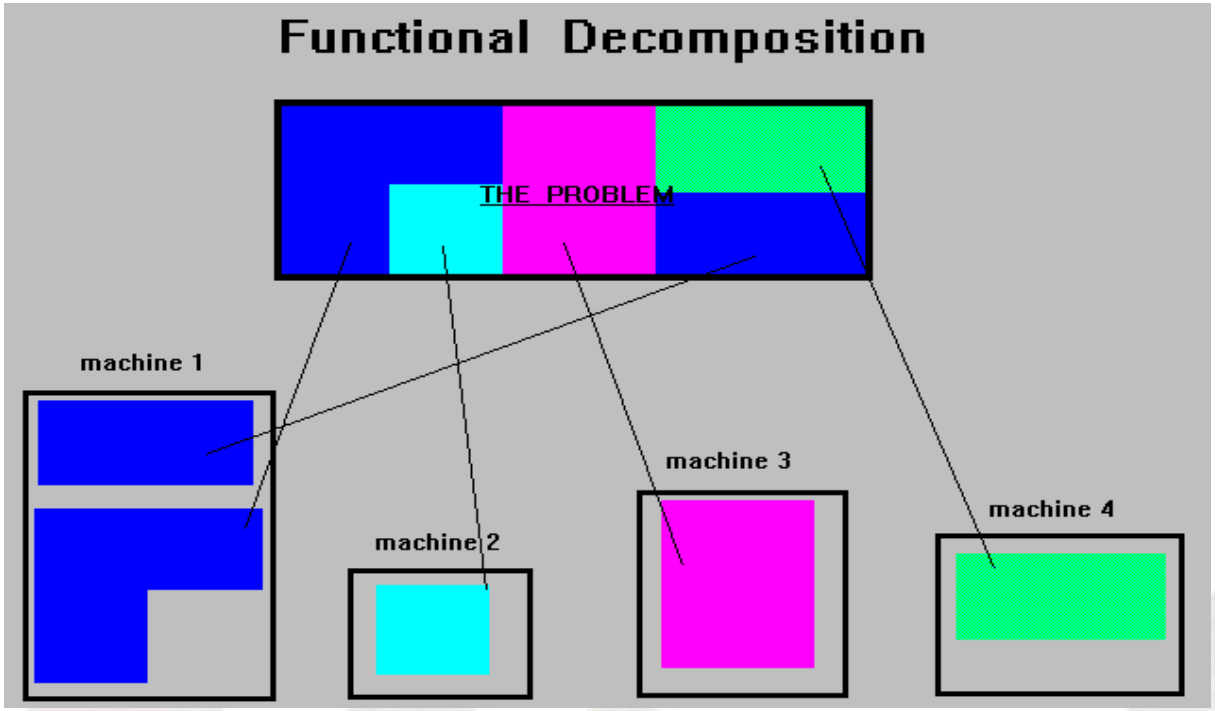
Decomposing the Program

There are three methods for decomposing a problem into smaller tasks to be performed in parallel: Functional Decomposition, Domain Decomposition, or a combination of both

- Functional Decomposition (Functional Parallelism)
  - Decomposing the problem into different tasks which can be distributed to multiple processors for simultaneous execution
  - Good to use when there is not static structure or fixed determination of number of calculations to be performed
- Domain Decomposition (Data Parallelism)
  - Partitioning the problem's data domain and distributing portions to multiple processors for simultaneous execution
  - Good to use for problems where:
    - data is static (factoring and solving large matrix or finite difference calculations)
    - dynamic data structure tied to single entity where entity can be subsetted (large multi-body problems)
    - domain is fixed but computation within various regions of the domain is dynamic (fluid vortices models)
  - There are many ways to decompose data into partitions to be distributed:
    - One Dimensional Data Distribution
      - Block Distribution
      - Cyclic Distribution
    - Two Dimensional Data Distribution
      - Block Block Distribution
      - Block Cyclic Distribution
      - Cyclic Block Distribution

Functional Decomposing of a Program

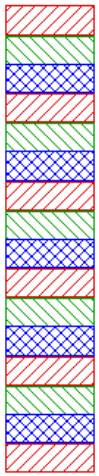
- Decomposing the problem into different tasks which can be distributed to multiple processors for simultaneous execution
- Good to use when there is not static structure or fixed determination of number of calculations to be performed



Domain Decomposition (Data Parallelism)

- Partitioning the problem's data domain and distributing portions to multiple processors for simultaneous execution
- There are many ways to decompose data into partitions to be distributed:

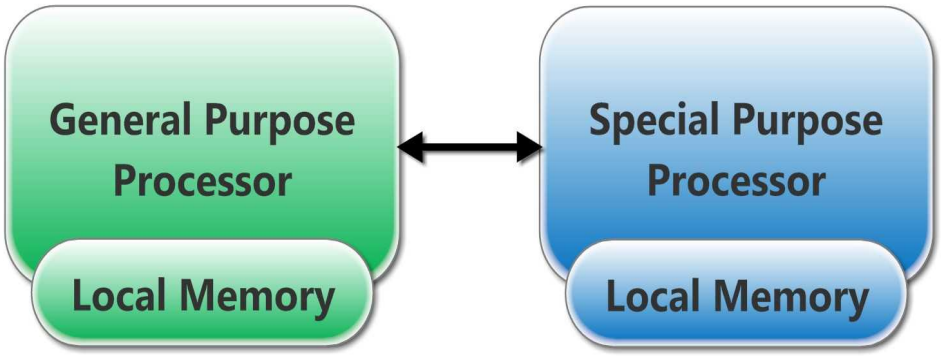
Cyclic Distribution of A



Block Distribution of A

Generic: HPC System

```
void main()
{
variable initialize
Memory Data Sets
Memory Read(Local Memory)
Computations (Memory)
Transfer Data
for ( …
   for ( …
      for ( …
Memory Write(Local Memory)
}
```



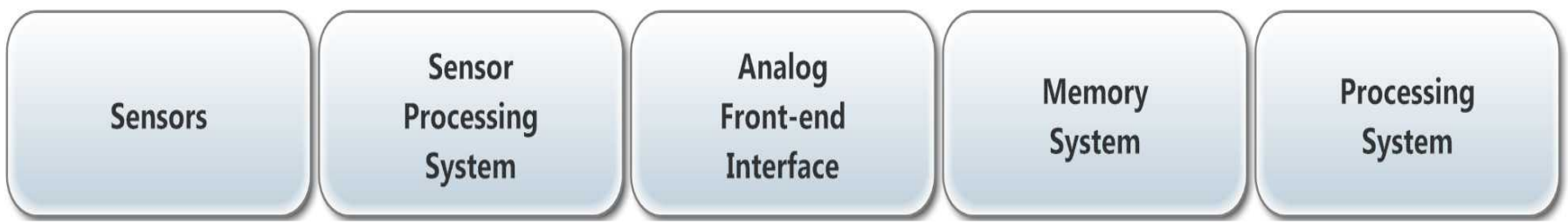
Irregular/Complex Applications e.g. Sorting, Search.

Dense Applications e.g. Matrix Multiplication, Tile Computation.

Supercomputer System Architectures

Rank	Site	System	Cores	Rpeak (TFlops)	Power (kW)	Tflop/KW
6	Swiss National Supercomputing Centre (CSCS), Switzerland	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , **NVIDIA K20x** - Cray Inc.	115984	6271	2325	2.697204
9	DOE/NNSA/LLNL, United States	Vulcan - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect - IBM	393216	4293.3	1972	2.17713
8	Forschungszentrum Juelich (FZJ), Germany	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect - IBM	458752	5008.9	2301	2.176836
3	DOE/NNSA/LLNL, United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom - IBM	1572864	17173.2	7890	2.176578
5	DOE/SC/Argonne National Laboratory, United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom - IBM	786432	8586.6	3945	2.176578
2	DOE/SC/Oak Ridge National Laboratory, United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, **NVIDIA K20x** - Cray Inc.	560640	17590	8209	2.14277
1	National Super Computer Center in Guangzhou, China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P - NUDT	3120000	33862.7	17808	1.901544
7	Texas Advanced Computing Center/Univ. of Texas, United States	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P - Dell	462462	5168.1	4510	1.14592
4	RIKEN Advanced Institute for Computational Science (AICS), Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect - Fujitsu	705024	10510	12660	0.830174

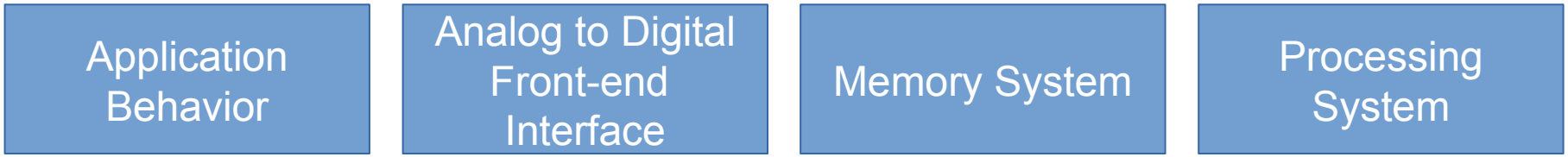
System Design Approach



- Design Specialized Hardware System for Life and Earth Science Applications.
  - Sensor System
  - Analog Front-End Interface
  - Memory System
  - Processing System

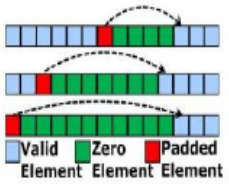
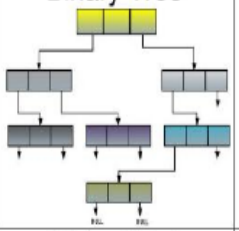
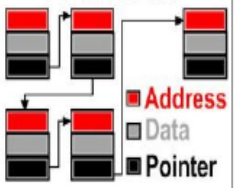
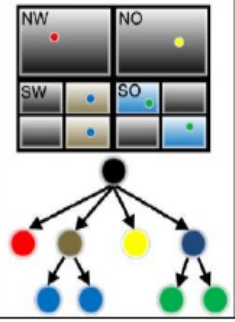
System Design Steps : Executing a Problem

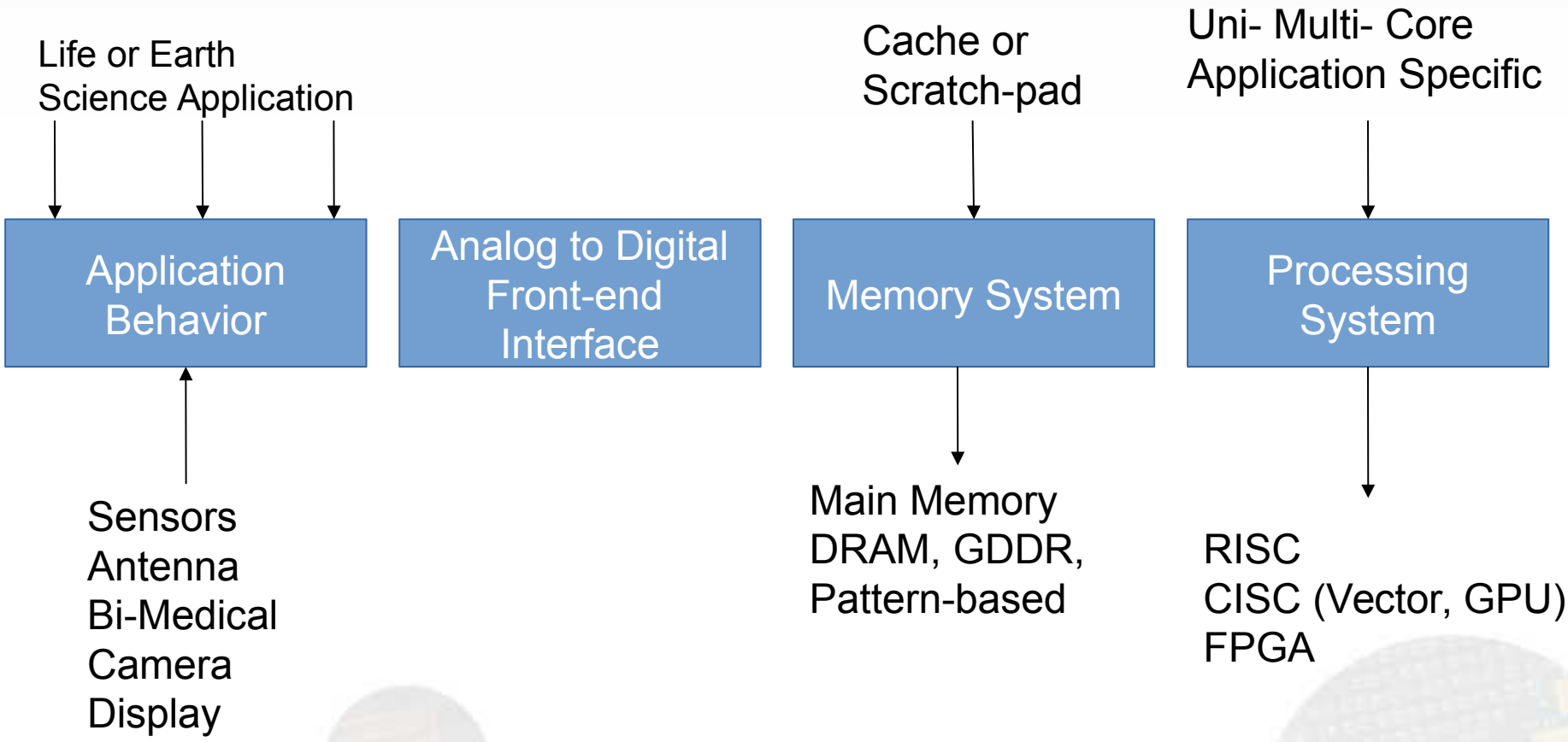
- Access Pattern
- Front-end Interface
- Memory System
- Processing System



Basic types of memory access patterns

- Regular access
  - Fixed stride
  - Predictable
  - Parallel
- Irregular access
  - Variable strides
  - Known
    - » Predictable at compile-time
  - Unknown
    - » Independent
    - » Dependent

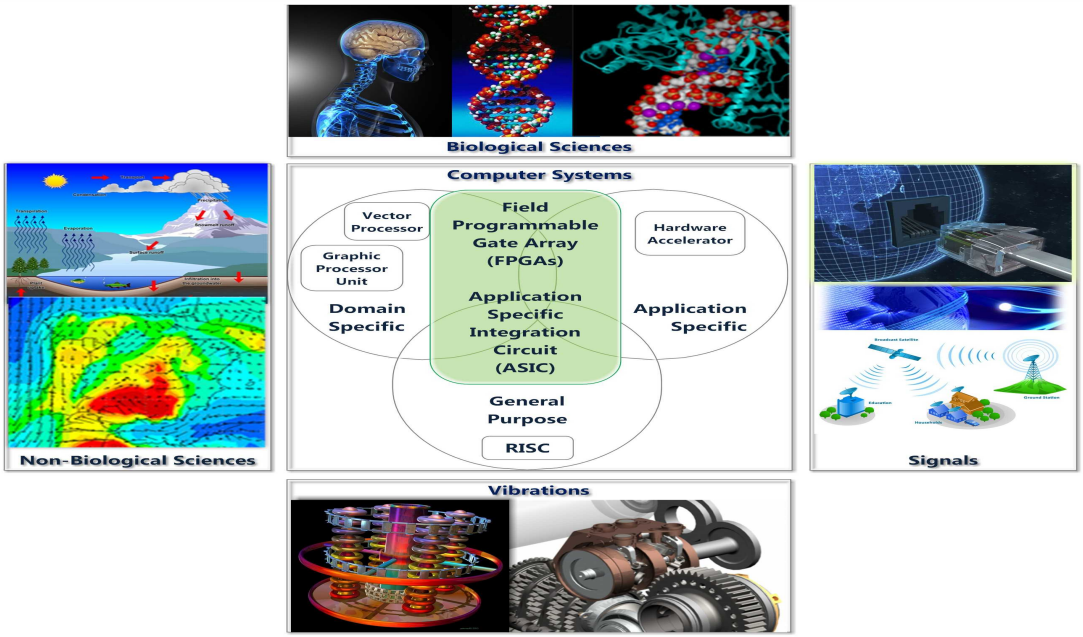
Kernel	Description	Access Pattern	Regular %	Irregular % known	Irregular % unknown
CRG	A compression algorithm, hides zero in a descriptor block	Pointer 		28	72
Huffman	Huffman is an entropy coding technique. Allocate codes to symbols, using frequency of occurrence for each symbol.	Binary Tree 		25	75
In_Rem	A Linked List Buffer	Linked List 	5	55	40
N-Body	The 3D-Hermite algorithm used to compute movement of bodies using the newtonian gravitational force.	Tree 	20	40	40


Life or Earth
Science Application
Application
Behavior
Sensors
Antenna
Bi-Medical
Camera
Display
Analog to Digital
Front-end
Interface
Cache or
Scratch-pad
Memory System
Main Memory
DRAM, GDDR,
Pattern-based
Uni- Multi- Core
Application Specific
Processing
System
RISC
CISC (Vector, GPU)
FPGA

- Personal Introduction
- Problem Formulation
  - Importance of Information
  - Importance of Digital Electronics
  - Problems and Solution
- HPC For Real World Problems
  - Hardware Architecture
  - Programming Models
  - Design Approach
- **Projects**

Target Applications Area

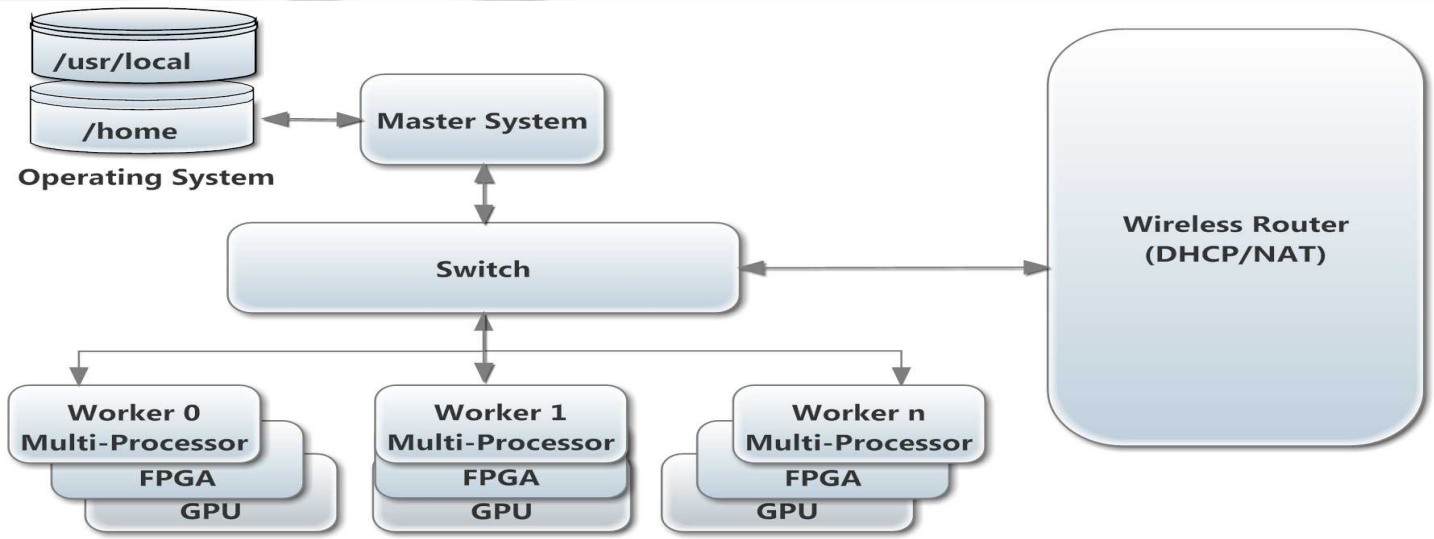
- Life Sciences
  - Biomedical Applications
  - Imaging Applications
  - Communication
  - Defense
- Earth Sciences
  - Interferometric Sensors
  - Oil Search



On Going Projects

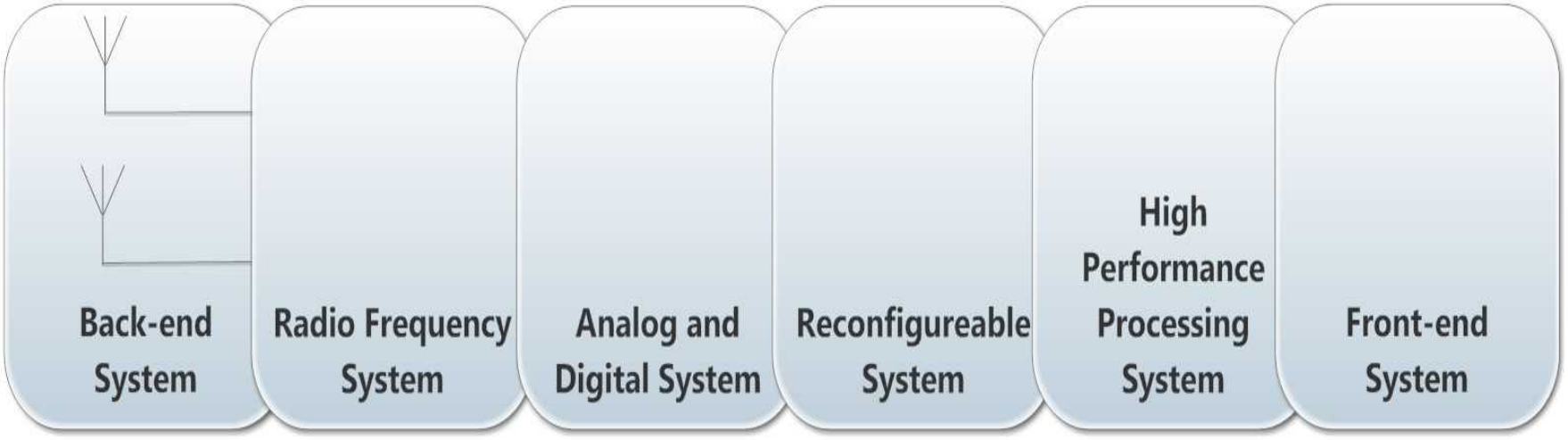
- Low Power Low Cost Supercomputer System
- Software Defined Radio based Heterogeneous System
- Medical Application Specific Processor
- Visual Processing System
- Analog & Digital Data Acquisition System
- High Performance Single Computer Board for Military and Industrial Applications

Low Power Low Cost Supercomputer System

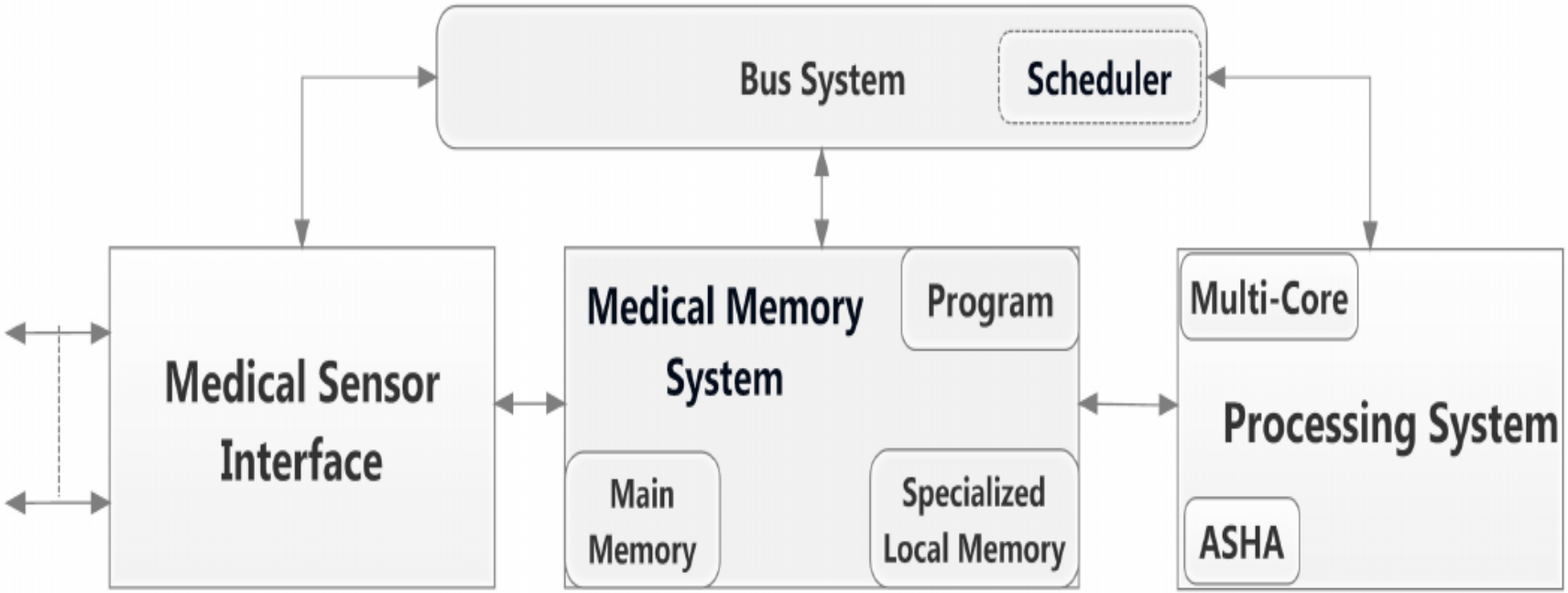


Institute	Peak Performance	Power Consumption
NUST, Islamabad	132 TFLOPS	N/A
(Our Proposal)	2.9 TFLOPS	420 Watts
KUST, Kohat	0.416 TFLOPS	N/A
COMSATS	0.158 TFLOPS	N/A
CIIT, Islamabad	0.05 TFLOPS	N/A
GIK Institute	N/A	N/A
KRL	N/A	N/A
UET Lahore	N/A	N/A
NED Karachi	N/A	N/A

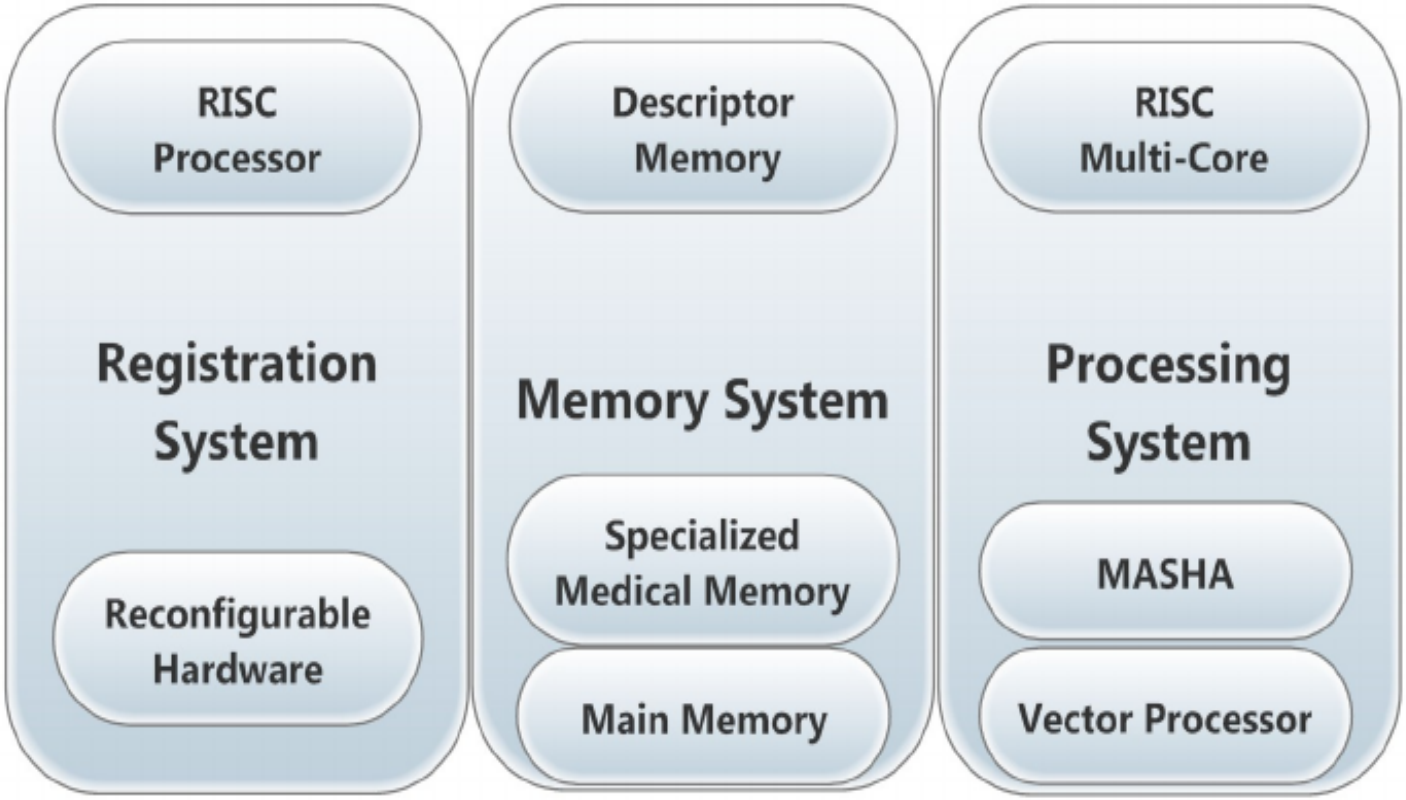
Software Defined Radio based Heterogeneous System



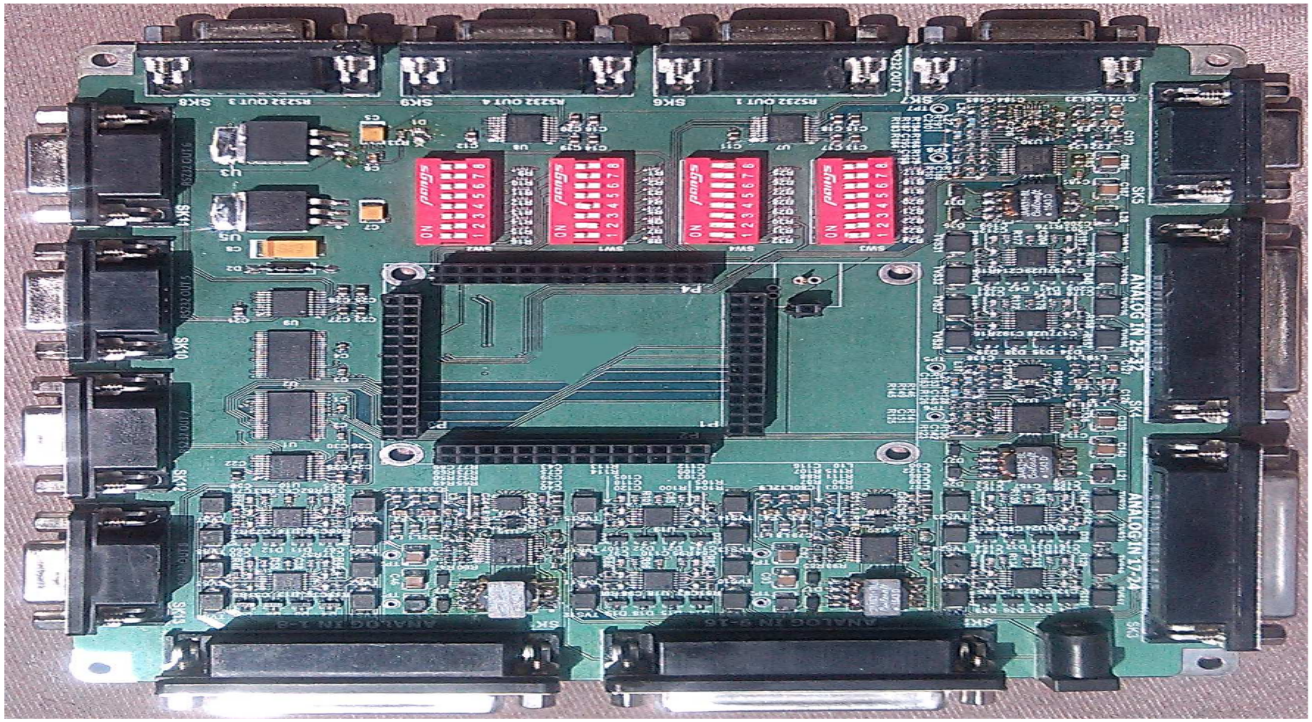
Medical Application Specific Processor



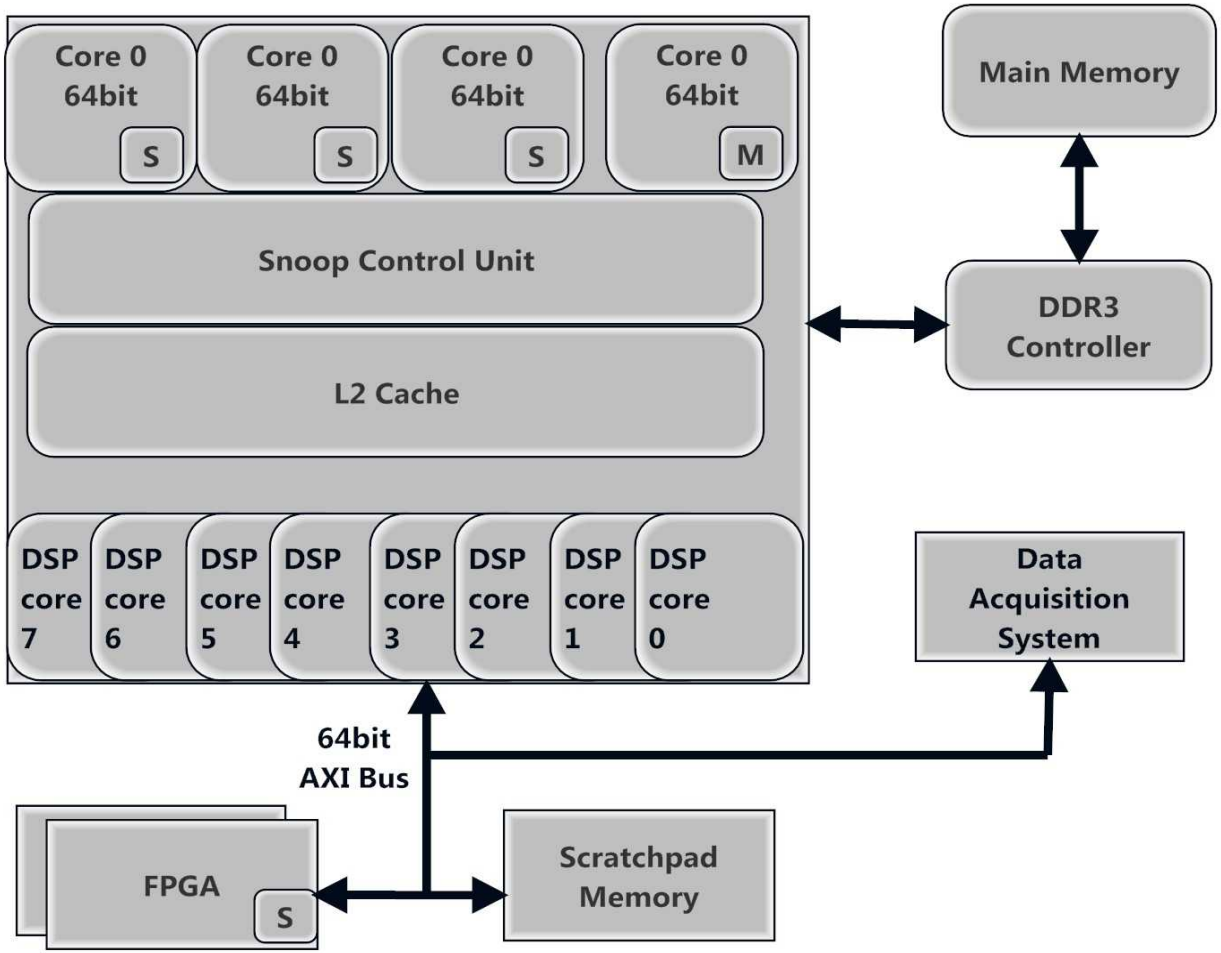
Visual Processing System

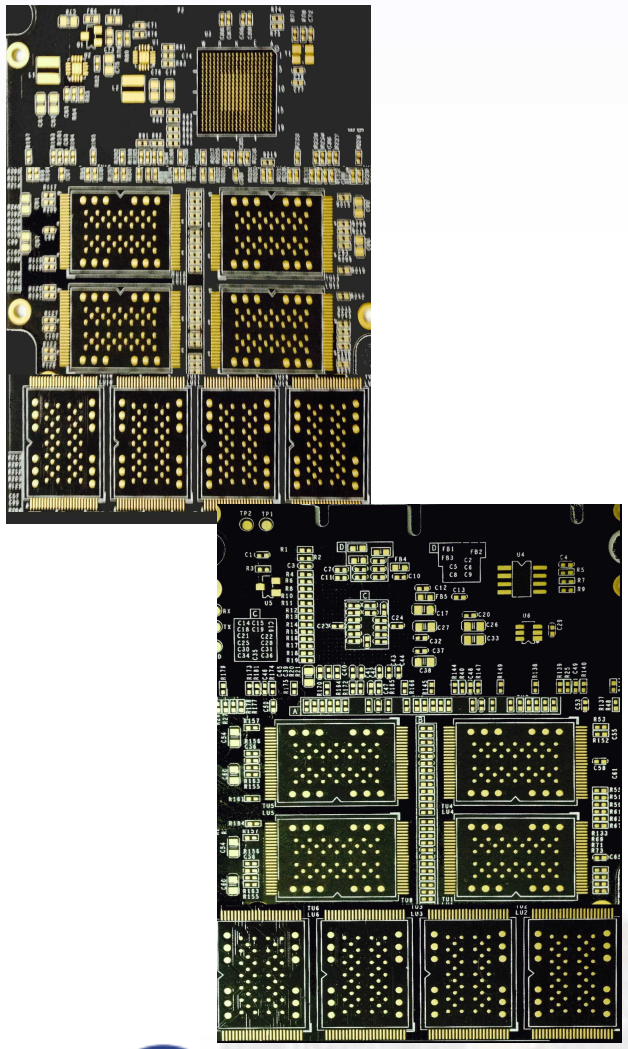


Analog and Digital Data Acquisition System



High Performance Single Computer Board for Military and Industrial Applications





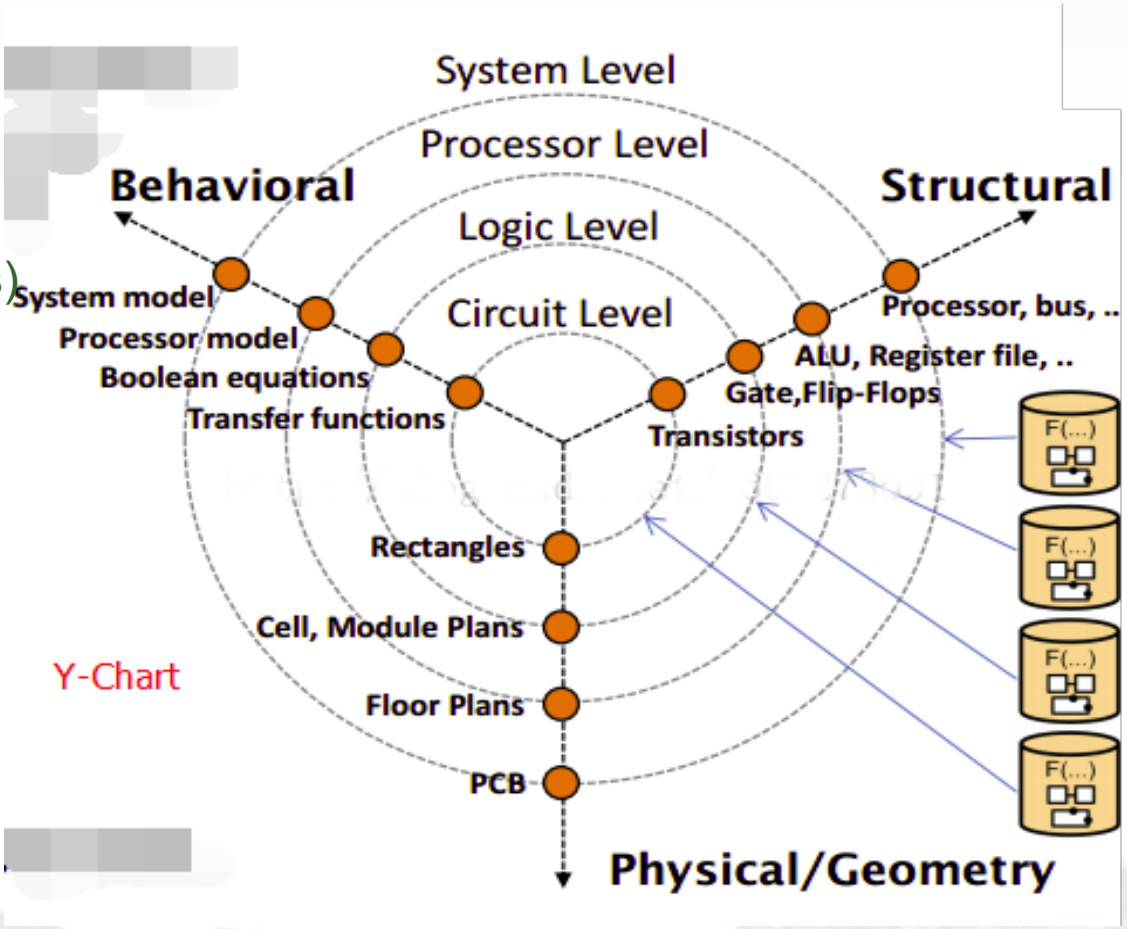
Unal Center of Education Research and Development
www.ucerd.com

RIPHAH INTERNATIONAL UNIVERSITY

BSC Microsoft Research Centre

System Design Approach to Solve Problem

- ,Multi-Core Systems
- GPU
- RISC (Intel, ARM, etc)
- ASIP (Hardware Accelerators)
- Architectures
- FPGA
  - HDL and HLS
- ASIC
  - DC
- Programming Models
- OpenMP
- OpenACC



Thanks