# CS528
# High Performance Computing

A Sahu

Dept of CSE, IIT Guwahati

# Outline

- Quick looks
  - HPC Trends and study requirement
  - HPC Course : Books and Contents
- Multiprocessor Intro
- Serial Code Optimization

# Param Ishan HPC

- **HPC system : Data Center**
- **Many Racks, Many rack server in a rack**
- **Nodes are Multicore : One rack server**


**Node/RackServer**

# Param Ishan SC @IITG

- Login Node:
  - 2x CPU login nodes, 1x GPU login node , 1x MIC login node
- Head and Management:
  - 1 pair head node (in redundant mode), 1x Management Node
- Compute Node:
  - 126x compute nodes
  - 4x high memory compute nodes
  - 16x CPU-GPU hybrid compute nodes
  - 16x CPU-MIC hybrid compute nodes.
- Network:  FDR InfiniBand network
- Storage :
  - 150TB high throughput scratch space.
  - 100TB high throughput home area
  - 50TB archival for long term data storage.

# Trend of HPC

- **HPC system**
  - Multi Nodes/Computer/Blades
  - **Programming Model MPI**
- **Nodes are Multicore**
  - Node have accelerators
  - **Programming Model : OpenMP, OpenCL/Cuda**
- **Core**
  - Multi Threaded, With vector instructions
  - 4 issue OOO Pipelines, Multilevel Caches,
  - **Programming Model: gcc optimized, vectorized code, OpenMP**

# Need to study in HPC: User Prospects

- Single Processor
  - Architecture: Core Pipeline, Core Multithreading, Cache Hierarchy, SIMD
  - C/C++ Optimization Methods: gcc, OpenMP, Simidization, cache optimized code
- Multicore node
  - Multicore, Accelerator, Interconnections
  - OpenMP Model, Cuda Model, Accelerated Model
- HPC Server
  - Multiple Nodes/Blades, Interconnection, Storages
  - Programming Model : MPI

# HPC course

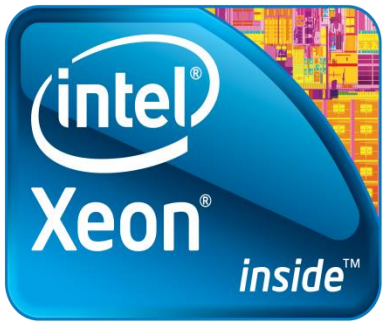
- 1st Half : Before Mid Sem
  - HPC, Architecture, Programming, Code Optimization
  - Scheduling, Energy Efficiency, Power Efficiency
- 2nd Half : After Mid Sem
  - Cloud Computing
  - Mobile Cloud, FOG, EGDE, IoT

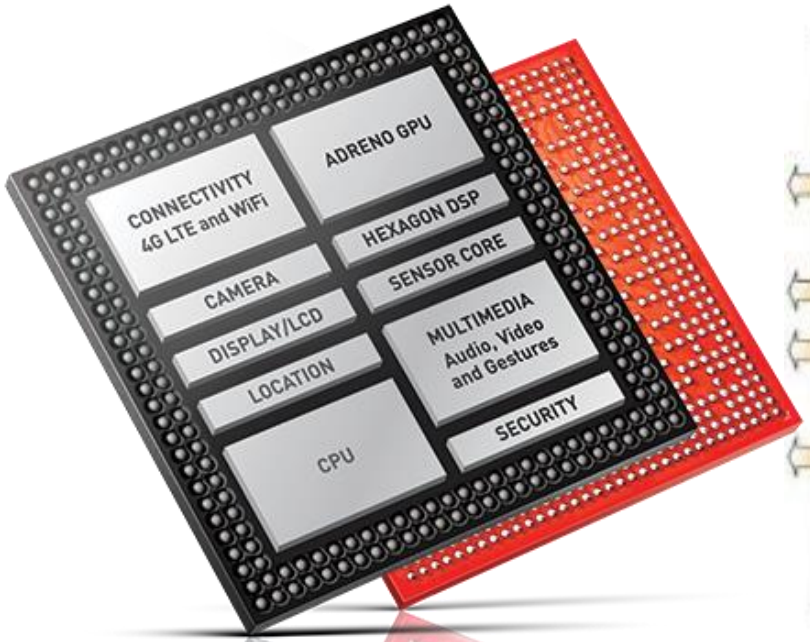
# Books : Text

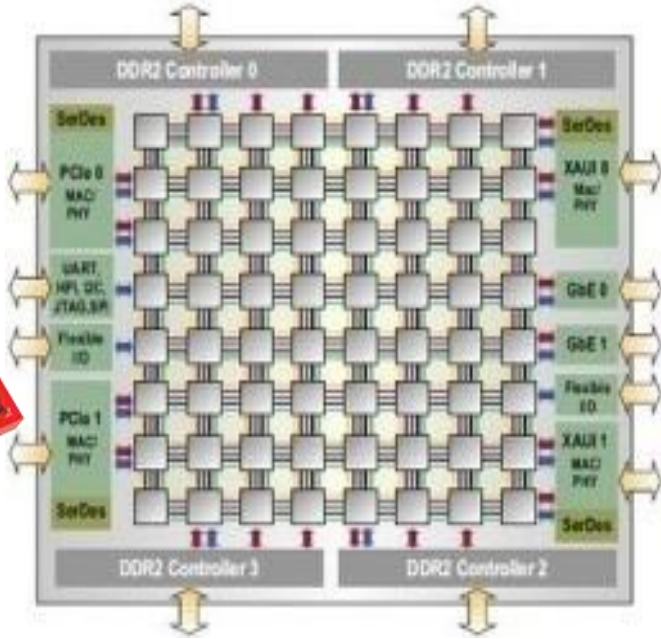
- Hager G and Wellein G . ***Introduction to High Performance Computing for Scientists and Engineers*** (1st ed.). CRC Press,, India, 2010.
- **Some user manuals**
- **Some recent papers**

# Multiprocessor
# (Will revisit again)

# Example of Multicore

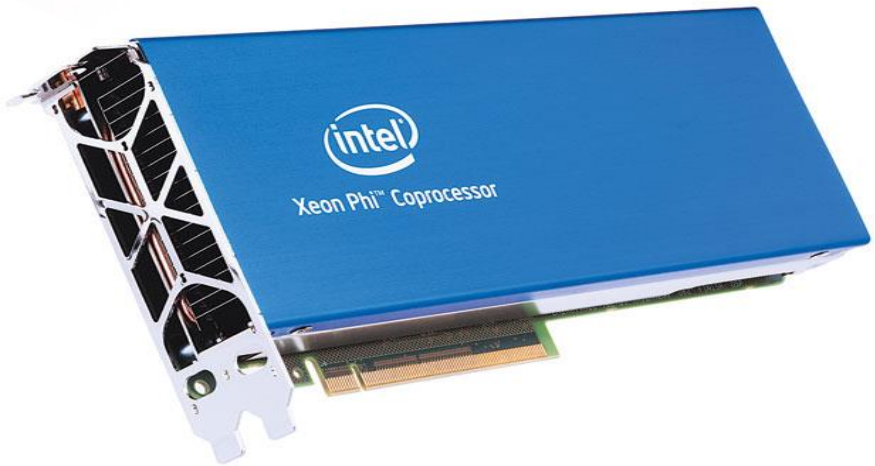



Tilera64





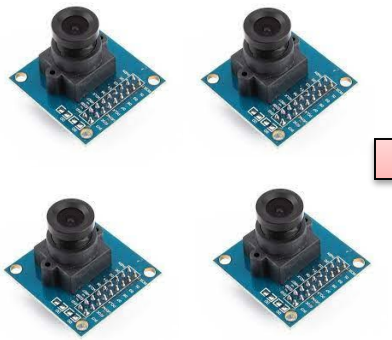
# Mobile SoC Example

- Heterogeneous
- Diff H/W for different purpose
- Efficiency in terms
  - Perf.
  - Energy
- All in one Chip

# Mobile SoC + Peripherals

**Similar to motherboard and components assembly**

**For every components we get dozens of variety to choose**



Mobile SoC


antenna

# Example of Multicore Arm SoC

- Apple: A15, M1, M1X, M2
  - 2x 3.23GHz (Firestorm) + 4x 2GHz (Icestorm) or 8 core, Neural engine, GPU
- **Qualcomm: SD888, SD870**
  - y. It has 1 KryoX1@2.8, 3 A78@2.4, 4 A55@1.8, AI, 5G, GPU
- Samsung: Exynos 9611
  - 4 A73@2.3Ghz, 4 A53@1.7Ghz, Mali G72, 5G, Codecs
- Huwai : hisilicon9000
  - 1 A77@3.13, 3 A77@2.54, 4 A55@2.0, Mali MP24, AI, 5G, neural
- **Mediatek : Dimensity 1200**
  - 1 A78@3.0, 3 A78@2.6, 4 A55@2.0, Mali MP24, 5G, AI,
- **Benchmarking: Antutu9, Geekbench 5, 3D Mark, Gaming FPS**

# Why Multicore ?

- **Saturation of single processor performance**
- Speed limit not to crosses : 4GHz
  - The ultimate point
  - Power consumption is proportional to cube of frequency

$$P = \frac{1}{2}.C.V^2.f = k.f^3$$

*as V is proportional to f*

- Single-processor
  - Branch prediction accuracy gone upto 95%
  - L1 Cache hits gone upto 80%
  - **ILP exploited by uniprocessor is upto 8 (mostly)**
  - **Thread/Data level parallelism needs to exploit**

# Power Aware Scheduling

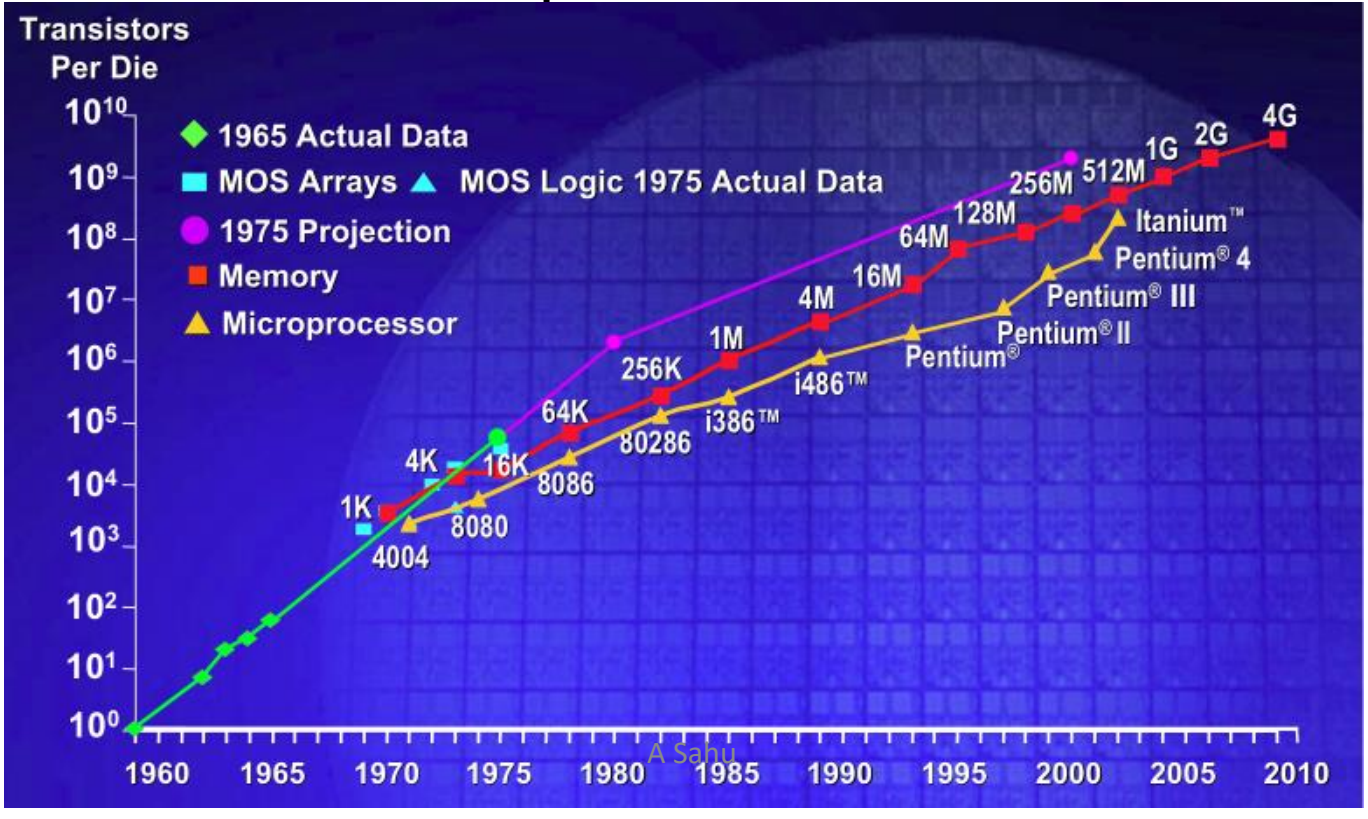
- $P = 1/2\ C\ V^2\ F = kF^3$ ***As V-F Pairs, V α F***
  - Running Processor at 3 Ghz will consumes 27 times higher Power as compared to running at 1Ghz
- $E = kF^3 * T$
- Running a task at F and F/3 :Assume 3Ghz and 1Ghz
  - $E_f = k\ F^3\ T$
  - $E_{f/3} = k\ (F/3)^3\ T*3 = k\ F^3\ T\ /9 = E_f/9$
  - 2 times slower but 4 times energy efficient
  - 3 times slower but 9 times energy efficient
- If time permit reduce the frequency
- If task have enough slack before deadline reduce frequency

# Why Multicore ?

- Application specific IC (ASIC)
  - High performance, low power than **Processor**
  - **But complexity of ASIC design is very High**
  - Example: 50MP+UHDVideo, GPS Camera in side mobile handset
  - **It is fixed for an application**

# Why Multicore ?

- **VLSI technology offering high integration density**
- Moore's Law (In 1965, Gordon Moore Prediction)
  - Exponential growth of the number of transistors on an IC
  - Doubled every 26 months for the past three decades
- Why more transistors per IC? Smaller transistors, Larger dice



# Why Multicore ?

- Many applications are highly parallel
  - Take benefit of all parallelism (instruction, data and thread)
- Multiprocessors
    - Flexible, programmable, high performance
    - Take benefit of all parallelism (instruction, data and thread)
    - Likely to be cost/power effective  solutions

# Why Multicore ?

- Multiprocessors are
  - Flexible, programmable, high performance
    - Processor are programmable as compared to ASIC
    - Flexible in terms of portability as compared to ASIC
    - Higher Performance than single processor

# Why Multicore ?

- Multiprocessors are likely to be cost/power effective solutions
  - **Share lots of resources**
    - ***Personal room is costlier than dormitory***
    - ***You cannot allocate a Bungalow to each student: it will too costly***
      - ***Hostel room with shared facility is sufficient***
  - Need not require very high frequency to run
  - Lots of replication makes easy to manage and cost effective in design

# Multicore Difficulties I

- Multiprocessors are likely to be cost/power effective solutions
  - Because it share lots of resources
    - ***Personal room is costlier than dormitory***
  - Sharing resource arise many other problems
    - Critical Sections
      - Lock and Barrier Design
    - Coherence
      - Shared data at all placed should be same
    - Consistency
      - Order should be similar to serial (ROB)
    - One processor Interference others
      - Share efficiently using some policy

# Multicore Difficulties II

- Many applications are highly parallel
  - Take benefit of all parallelism (instruction, data and thread)
  - Most of the coder write sequential code
  - Who will extract parallelism from applications ?
  - There is no successful auto-parallelisation tool till date
    - » Attempts: Cetus, SUIF, SolarisCC
- ***Good news: CNN/DNN python parallel library is quite successful in GPU domain***

# Multicore Difficulties III

- Task scheduling in multiprocessors
  - Deterministic task scheduling on multiprocessor with more than 2 processor is NP-Complete problem
  - 4 Tasks (A,B, C and D), 3 Processors
    - {A,B,C,D}{}{}, {A,B,C}{D}{}, .....Exponential Number of Solutions
- Simple Example
  - 8 tasks with execution 2, 4, 8, 5, 6, 4, 3, 20
  - Need to executed non-pre-emptively on two processor P1 and P2
  - So that overall execution time is minimized
  - **Solution : Divide 8 tasks in to two subsets, with difference of their sum is minimized ;Subset Sum Problem**

# Fundamental
# of
# Serial Code Optimization

# Code Optimization & Performance

- Machine-independent opt
  - Code motion,  Reduction in strength, Common sub-expression Elimination
- Machine-dependent opt
  - Pointer code, Loop unrolling,  Enabling instruction-level parallelism
- Tuning: Identifying perf bottlenecks
- Understanding processor Opt
  - Translation of INS into Ops,  OOO
- Branches
- Caches and Blocking
- Advice

# Speed and optimization

- Programmer
  - Choice of algorithm, Intelligent coding
- Compiler
  - Choice of instructions, Moving code, Reordering code, Strength reduction, *Must be faithful to original program*
- Processor
  - Pipelining, Multiple FU, Memory accesses, Branches, Caches
- Rest of system : Uncontrollable, OS, Load