Challenges & Opportunities in Heterogeneous Multi-Core Era

R. Govindarajan
Supercomputer Centre
Indian Institute of Science
Bangalore, India
govind@serc.iisc.ernet.in

Overview

- Introduction
- Programming Challenges
- Exploiting Data, Thread and Task Level Parallelisms
  - StreamIT on CPU and GPU cores
  - MATLAB on CPU and GPU cores
- Other Challenges and Opportunities
- Conclusions

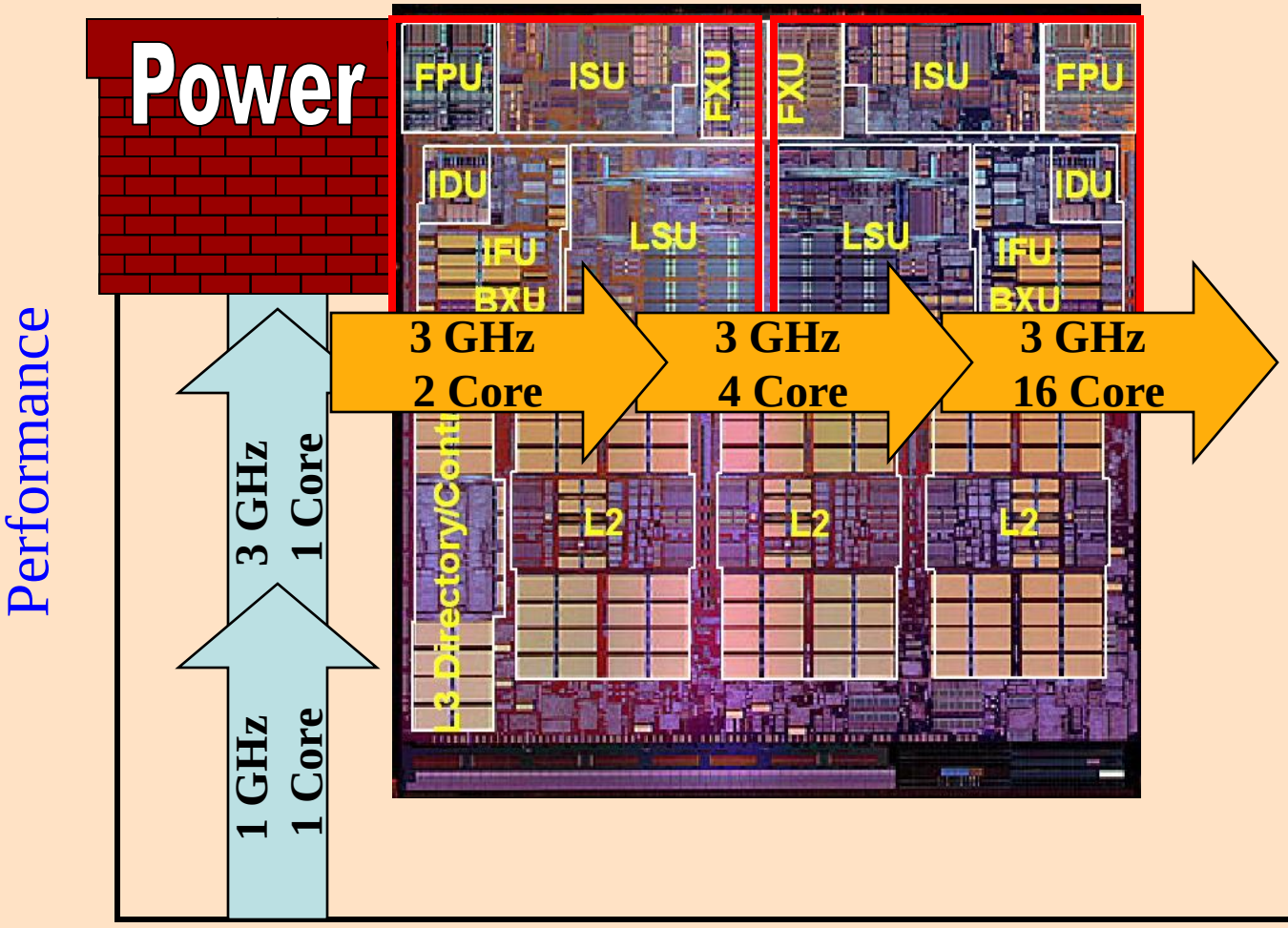
Moore's Law : Performance

Progress in Processor Architecture

- More transistors ⇒ New architecture innovations
  - Multiple Instruction Issue processors
    - VLIW
    - Superscalar
    - EPIC
  - More on-chip caches, multiple levels of cache hierarchy, speculative execution, …

Era of Instruction Level Parallelism

Multicores : The Right Turn



Progress in Processor Architecture

- More transistors ⇒ New architecture innovations
  - Multiple Instruction Issue processors
  - More on-chip caches
  - Multi cores
  - Heterogeneous cores and accelerators

    Graphics Processing Units (GPUs)

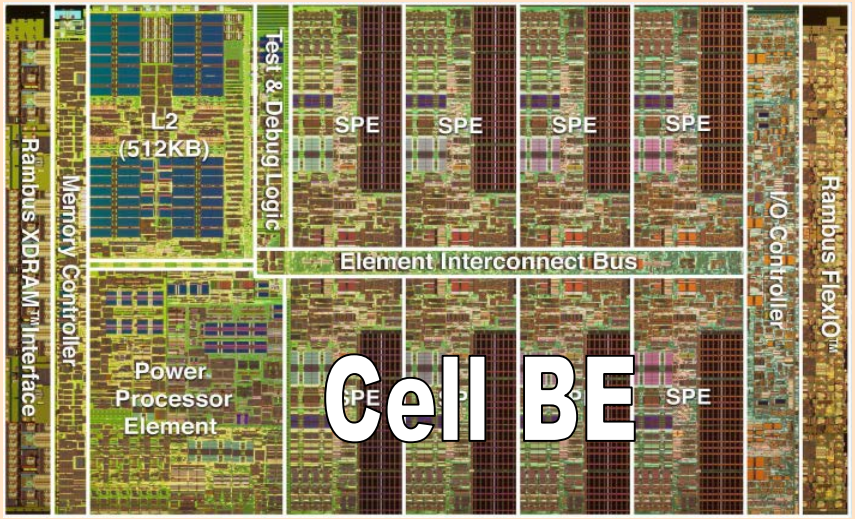
    Cell BE

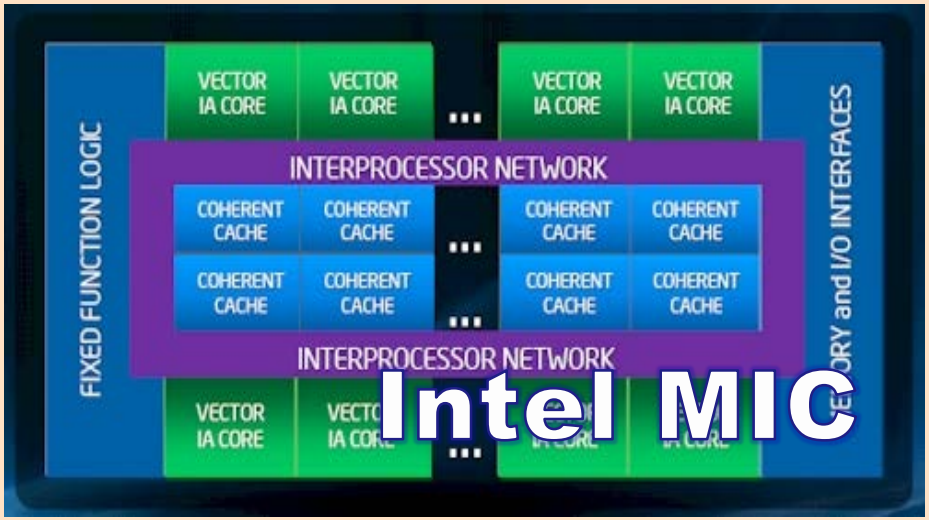
    Many Integrated Cores (MIC)

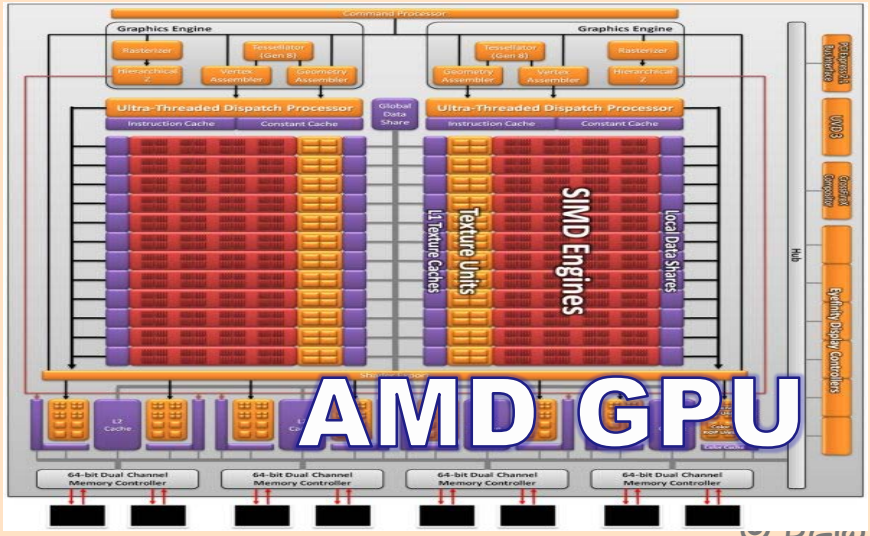
    Reconfigurable accelerators …

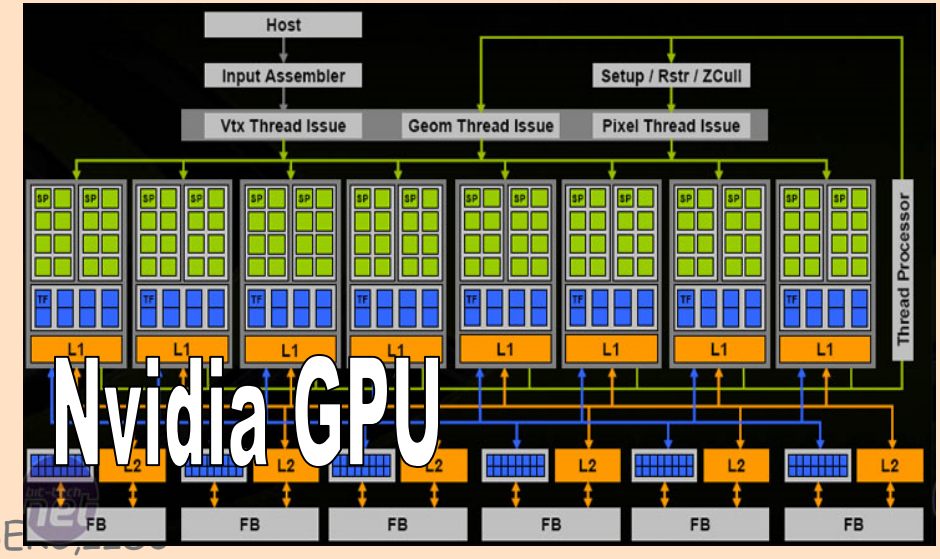
Era of Heterogeneous Accelerators

Accelerators





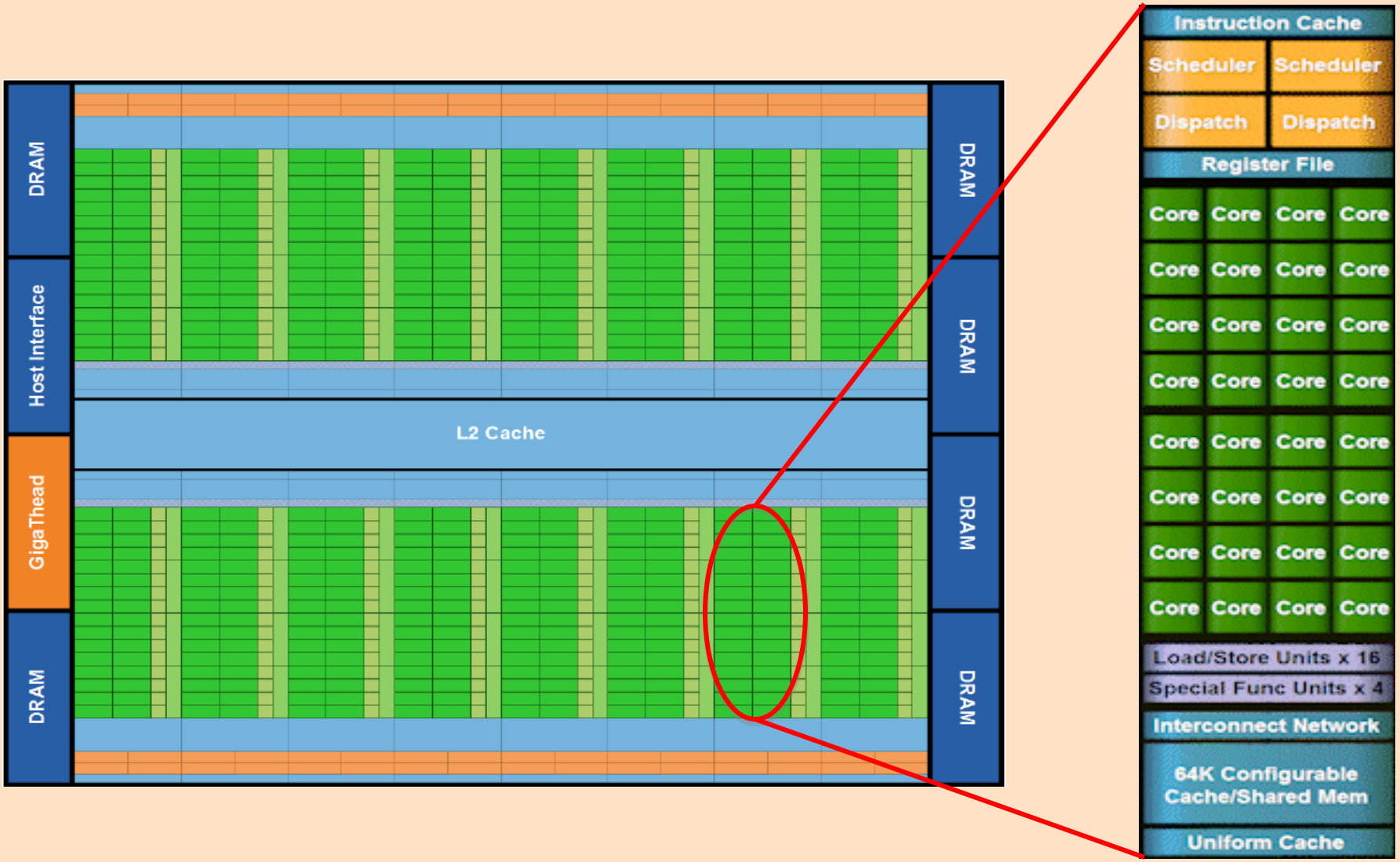




Accelerators: Hype or Reality?

Some Top500 Systems (Nov. 2011 List)				
Rank	System	Description	# Procs.	R_max (TFLOPS)
2	Tianhe	Xeon + Nvidia C2050 GPUs	186368	2,566
4	Nebulae-Dawning	Intel X5650, Nvidia C2050 GPU	55,680 + 64,960	1,271
5	Tsubame	Xeon + Nvidia GPU	73278	1,192
10	Roadrunner	Opteron + CellBE	6480 +12960	1,105

Accelerator – Fermi S2050



Handling the Multi-Core Challenge

- Shared and Distributed Memory Programming Languages
  - OpenMP
  - MPI
- Other Parallel Languages (partitioned global address space languages)
  - X10, UPC, Chapel, …
- Emergence of Programming Languages for GPU
  - CUDA
  - OpenCL

GPU Programming: Good News

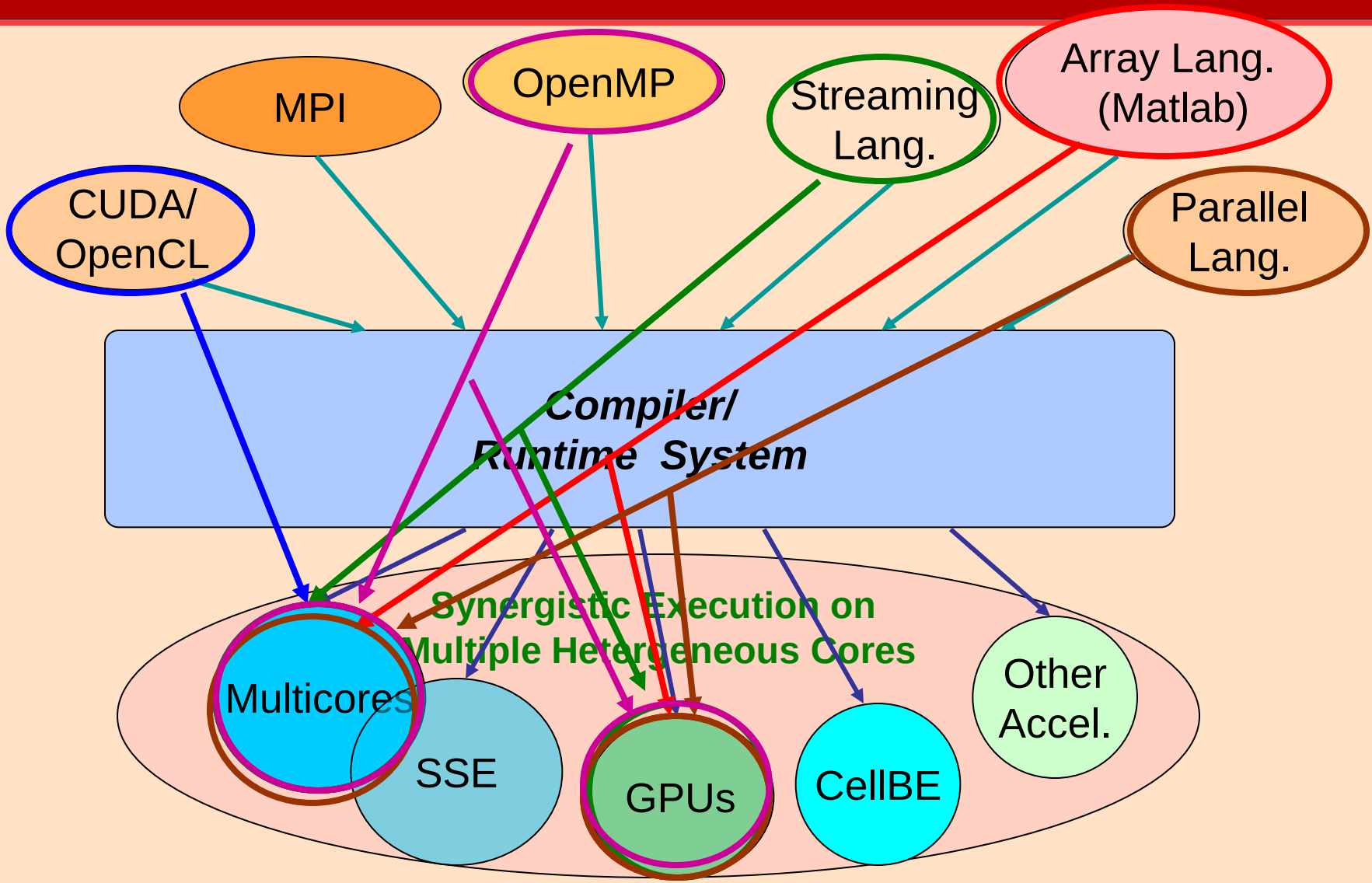
- Emergence of Programming Languages for GPU
  - CUDA
  - OpenCL – Open Standards
- Growing collection of code base
  - CUDAzone
  - Packages supporting GPUs by ISV
- Impressive performance
  - Yes!
- What about Programmer Productivity?

GPU Programming: Boon or Bane

- Challenges in GPU programming
  - Managing parallelism across SMs and SPMD cores
  - Transfer of data between CPU and GPU
  - Managing CPU-GPU memory bandwidth efficiently
  - Efficient use of different level of memory (GPU memory, Shared Memory, Constant and Texture Memory, …
  - Efficient buffer layout scheme to ensure all accesses to GPU memory are coalesced.
  - Identifying appropriate execution configuration for efficient execution
  - Synchronization across multiple SMs

What Parallelism(s) to Exploit?

Our Apprach

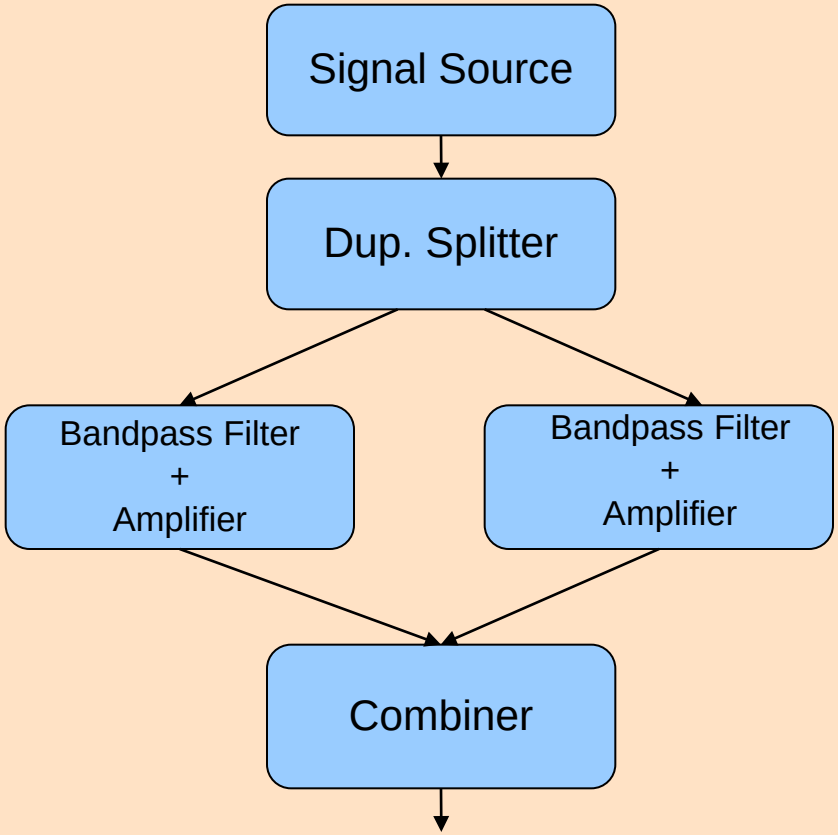

MPI
OpenMP
Streaming Lang.
Array Lang. (Matlab)
CUDA/ OpenCL
Parallel Lang.
Compiler/ Runtime System
Synergistic Execution on Multiple Heterogeneous Cores
Multicores
SSE
GPUs
CellBE
Other Accel.

Stream Programming Model

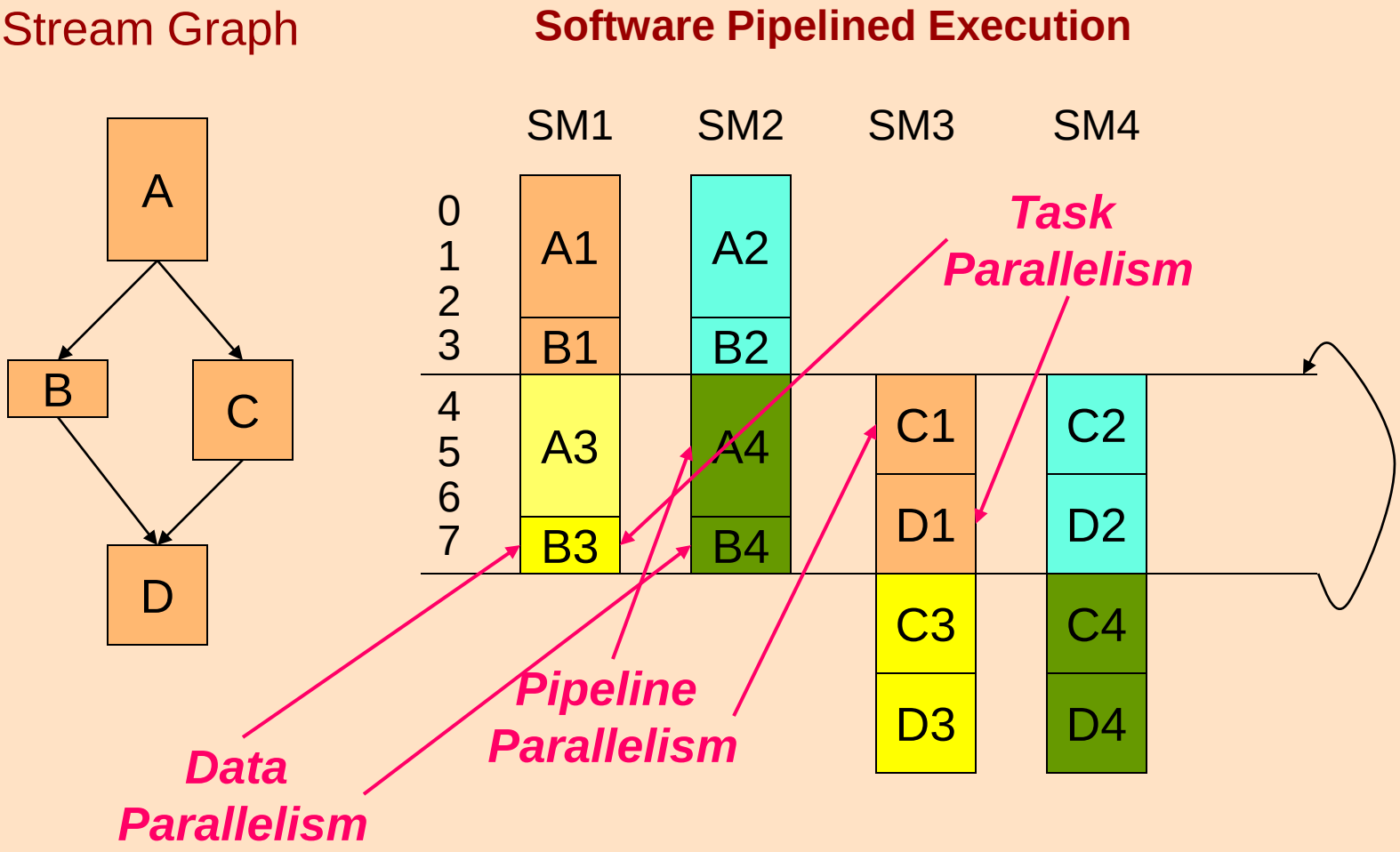
- Higher level programming model where nodes represent computation and channels communication (producer/consumer relation) between them.
- Exposes Pipelined parallelism and Task-level parallelism
- Synchronous Data Flow (SDF), Stream Flow Graph, StreamIT, Brook, …
- Compiling techniques for achieving rate-optimal, buffer-optimal, software-pipelined schedules
- Mapping applications to Accelerators such as GPUs and Cell BE.

StreamIT Example Program

2 – Band Equalizer



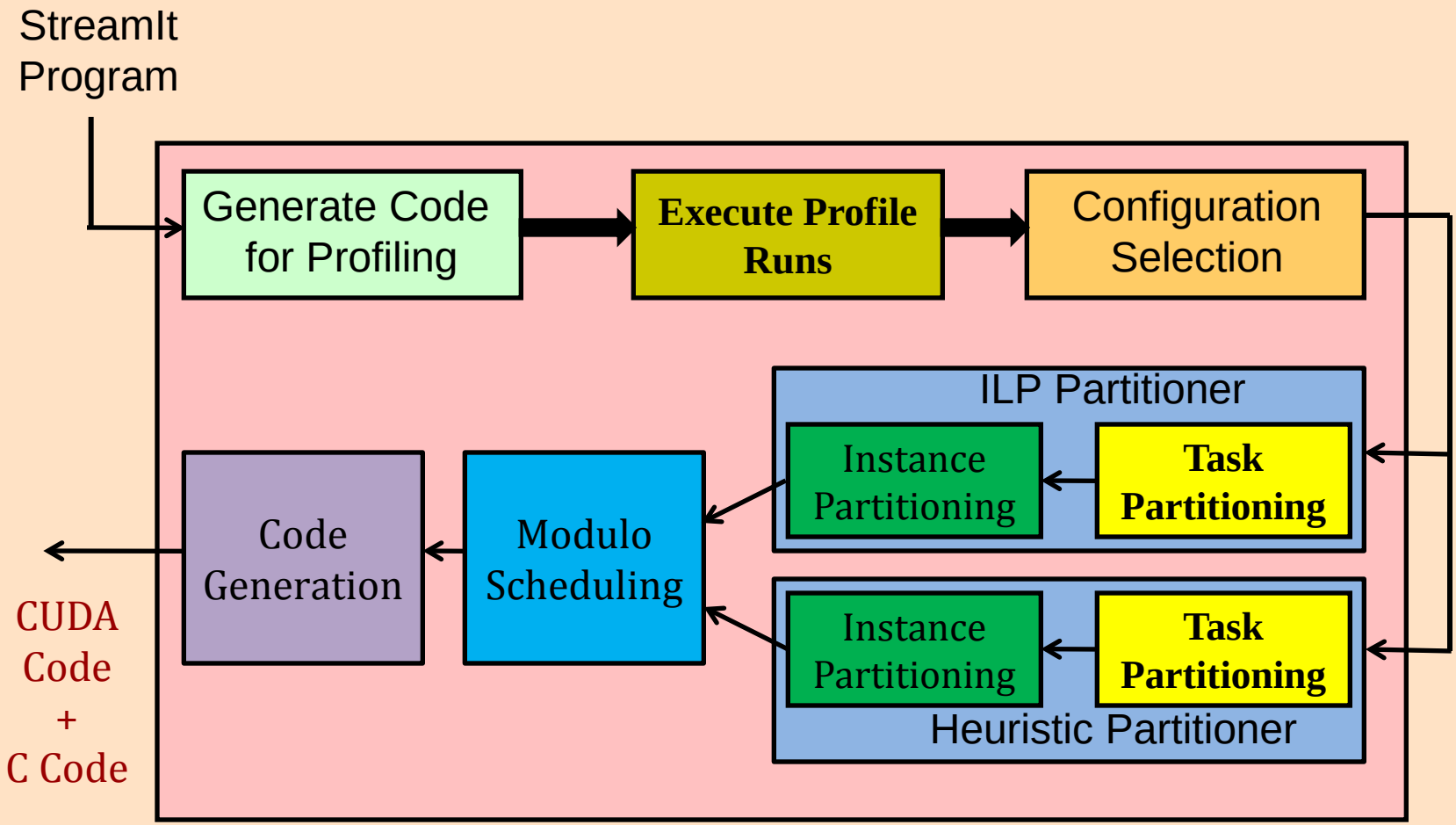
Stream Graph Execution



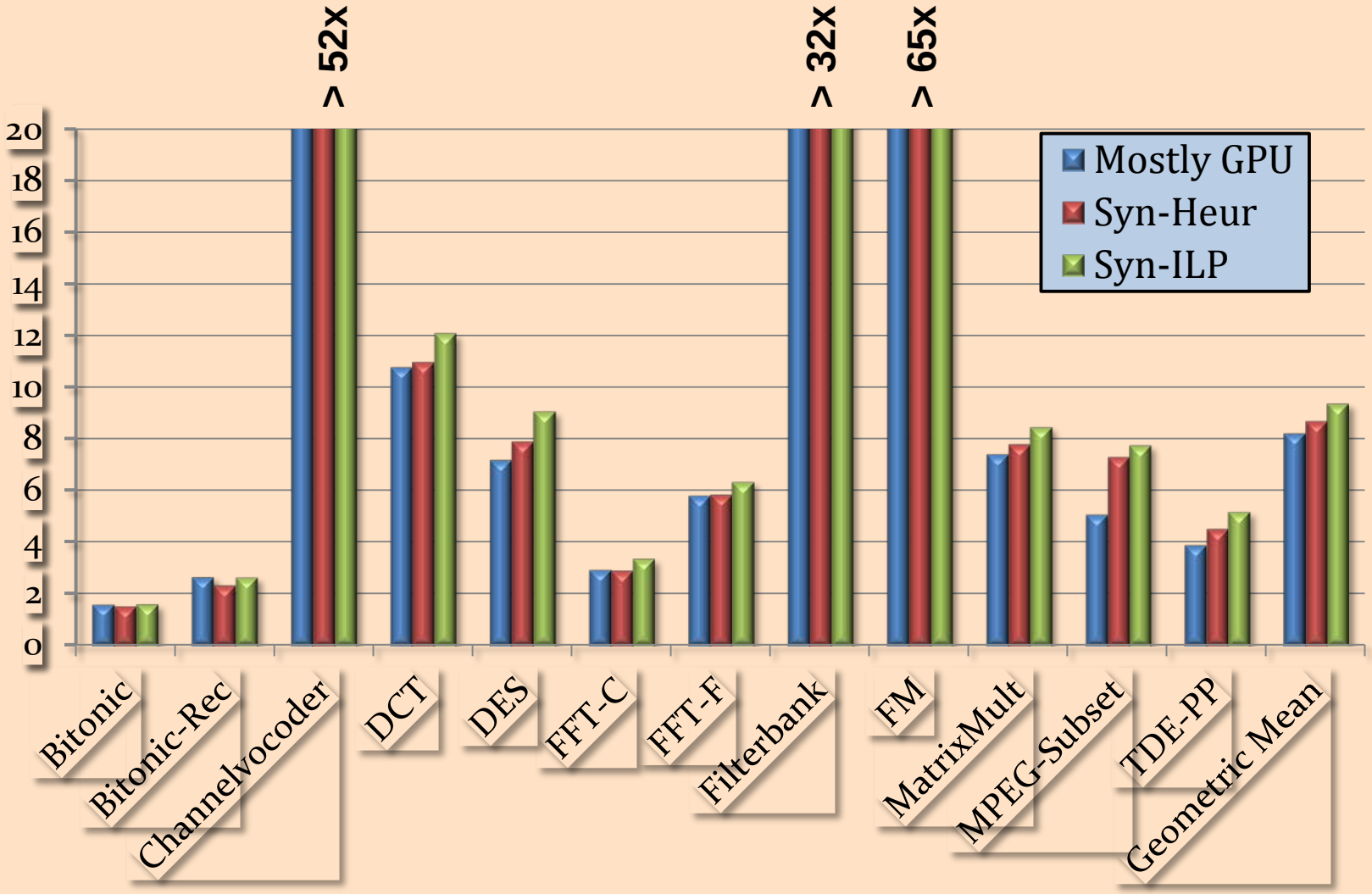
Our Approach

- Multithreading
  - Identify good execution configuration to exploit the right amount of data parallelism
- Memory
  - Efficient buffer layout scheme to ensure all accesses to GPU memory are coalesced.
- Task Partition between GPU and CPU cores
- Work scheduling and processor (SM) assignment problem.
  - Takes into account communication bandwidth restrictions

Compiler Framework



Experimental Results on Tesla

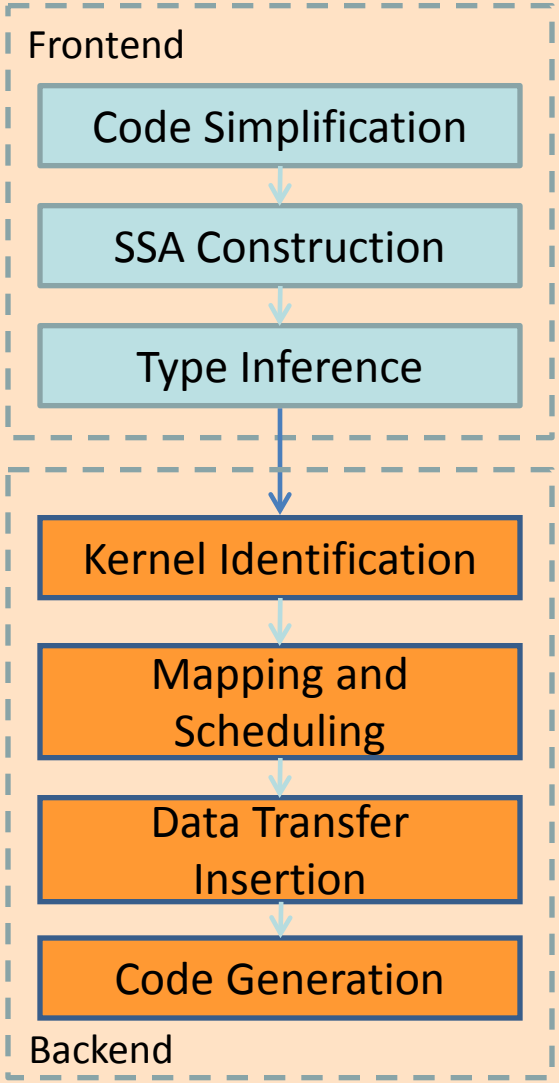


Compiling MATLAB to Heterogeneous Machines

- MATLAB is an array language extensively used for scientific computation
- Expresses data parallelism
  - Well suited for acceleration on GPUs
- Current solutions (Jacket, GPUmat) require user annotation to identify “GPU friendly” regions
- Our compiler, **MEGHA** (**M**ATLAB **E**xecution on **G**PU-based **H**eterogeneous **A**rchitectures), is fully automatic

Compiler Overview

- Frontend constructs an SSA intermediate representation (IR) from the input MATLAB code
- Type inference is performed on the SSA IR
  - Needed because MATLAB is dynamically typed
- Backend identifies “GPU friendly” kernels, decides where to run them and inserts reqd. data transfers



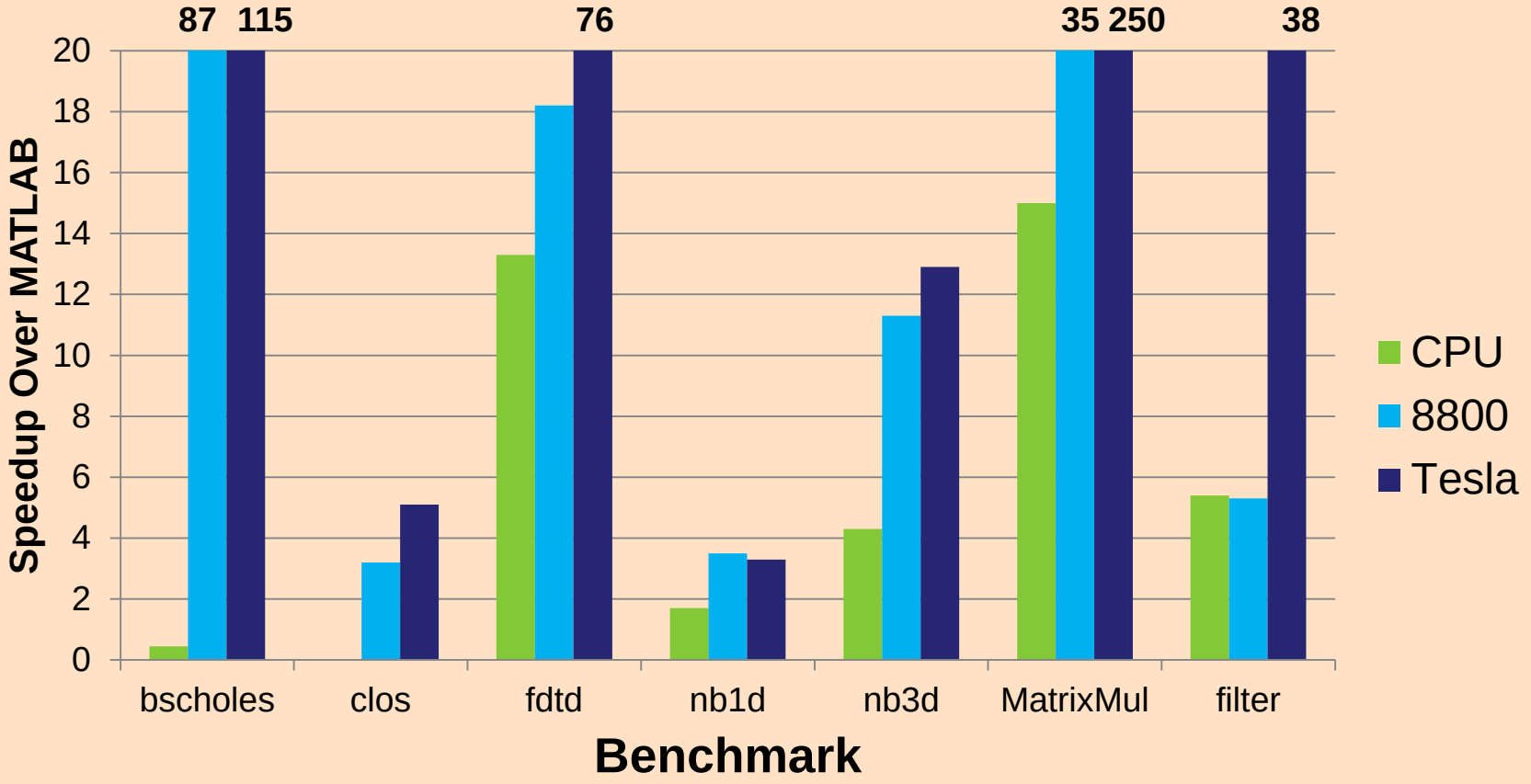
Backend : Kernel Identification

- ***Kernel identification*** identifies sets of IR statements (*kernels*) on which mapping and scheduling decisions are made
- Modeled as a ***graph clustering problem***
- Takes into account several costs and benefits while forming kernels
  - Register utilization (Cost)
  - Memory utilization (Cost)
  - Intermediate array elimination (Benefit)
  - Kernel invocation overhead reduction (Benefit)

Backend : Scheduling and Transfer Insertion

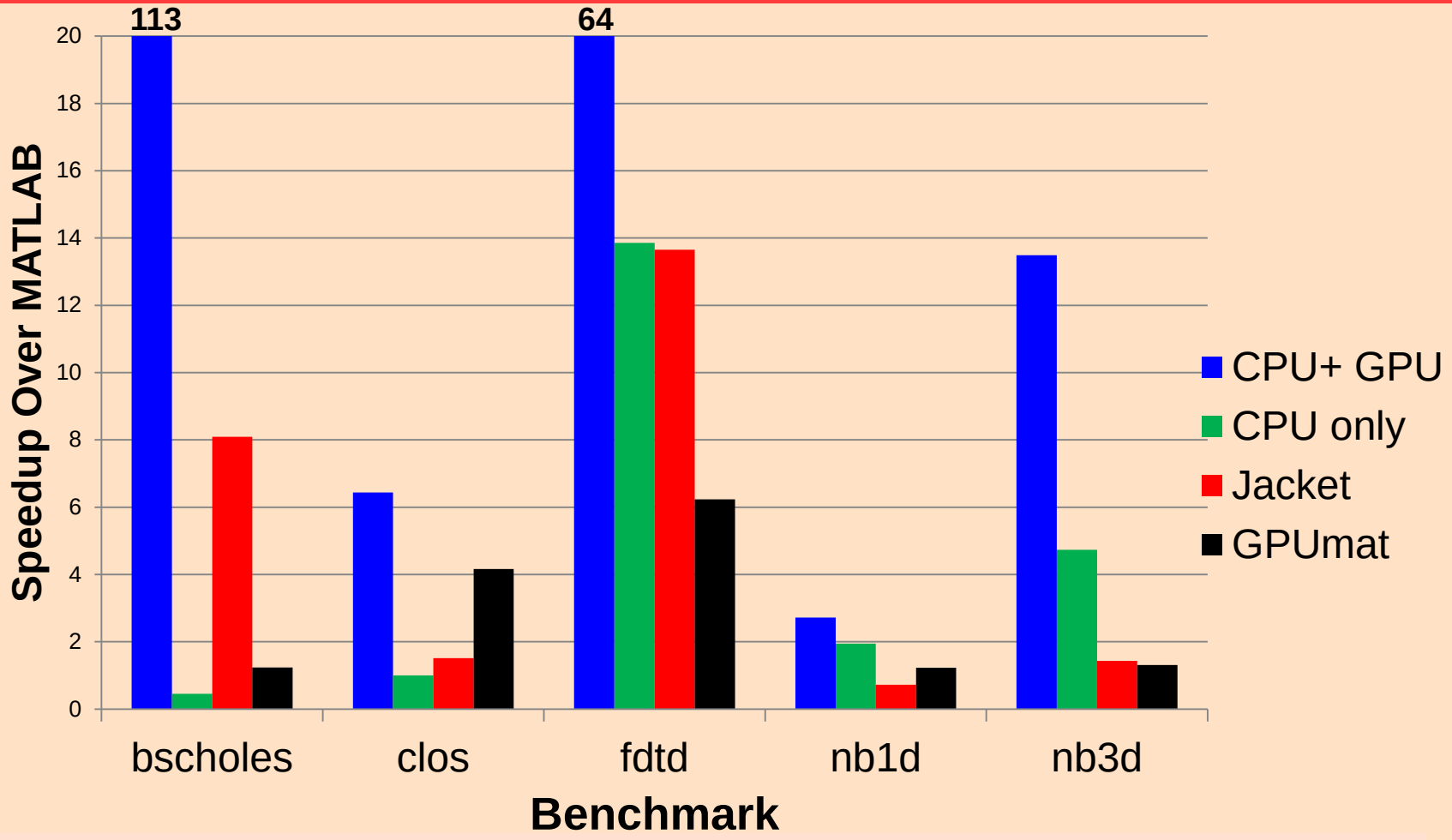
- Assignment and scheduling is performed using a heuristic algorithm
  - Assigns kernels to the processor which minimizes its completion time
  - Uses a variant of ***list scheduling***
- Data transfers for dependencies within a basic block are inserted during scheduling
- Transfers for inter basic block dependencies are inserted using a combination of data flow analysis and edge splitting

Results : Data Parallel Programs



- **Geometric mean speed up of 12X on 8800 and 29.2X on the Tesla**

Results : Data Parallel Programs



- **Performs better than GPUmat and Jacket**

Other Challenges & Opportunities

- Other accelerators
  - OMAP, MIC, FPGAs, …
- Need Programming model/languages for exploiting various types of parallelisms, still obtaining high level of Performance, Productivity and Portability
- Being able to efficiently map applications onto given platform.

Other Challenges & Opportunities

- In the context of embedded systems, the issues get even more complicated
  - Power, small form factor, multiple concurrent applications, response time, …
- Domain Specific Languages and Platforms

Thank You !!