Multi-Level Computing Architectures (MLCA)

Faraydon Karim
ST US-Fellow

Emerging Architectures
Advanced System Technology

STMicroelectronics

Outline

- MLCA
  - What's the problem
  - Traditional solutions: VLIW, superscalar, multiscalar, Multiprocessor
- Vision, roadmap
  - What MLCA enables, in different fields
- Status
  - Tools we have
  - Current research and cooperations
  - Future work (short term)

“The Key To Success”

“*The key to success in parallel processing on a chip will be the ability to map computational algorithms efficiently on to an array of resources, and hide the complexity from both programmer and user. The company that can do that has a shot at being the next Intel*.”

-- The Economist, March 13th 2003

System-Level Design Needs

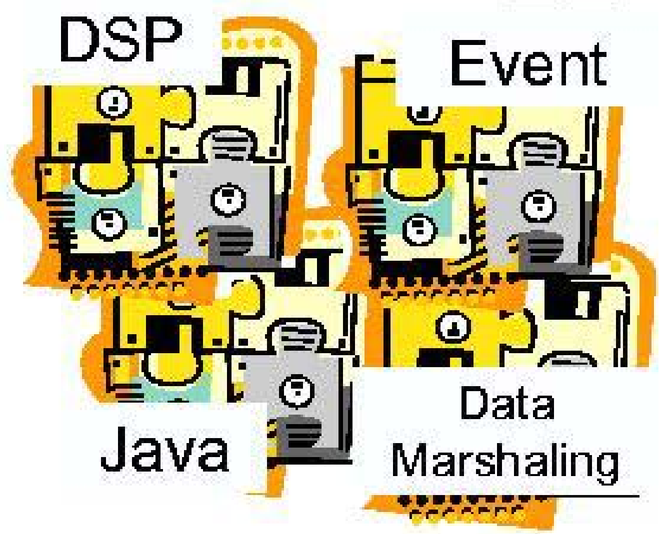
A Software/Systems Vision

TODAY:

- We don't know if a proposed solution will solve a customers problem until we build it (for infrastructure this takes us several years).

TOMMOROW:

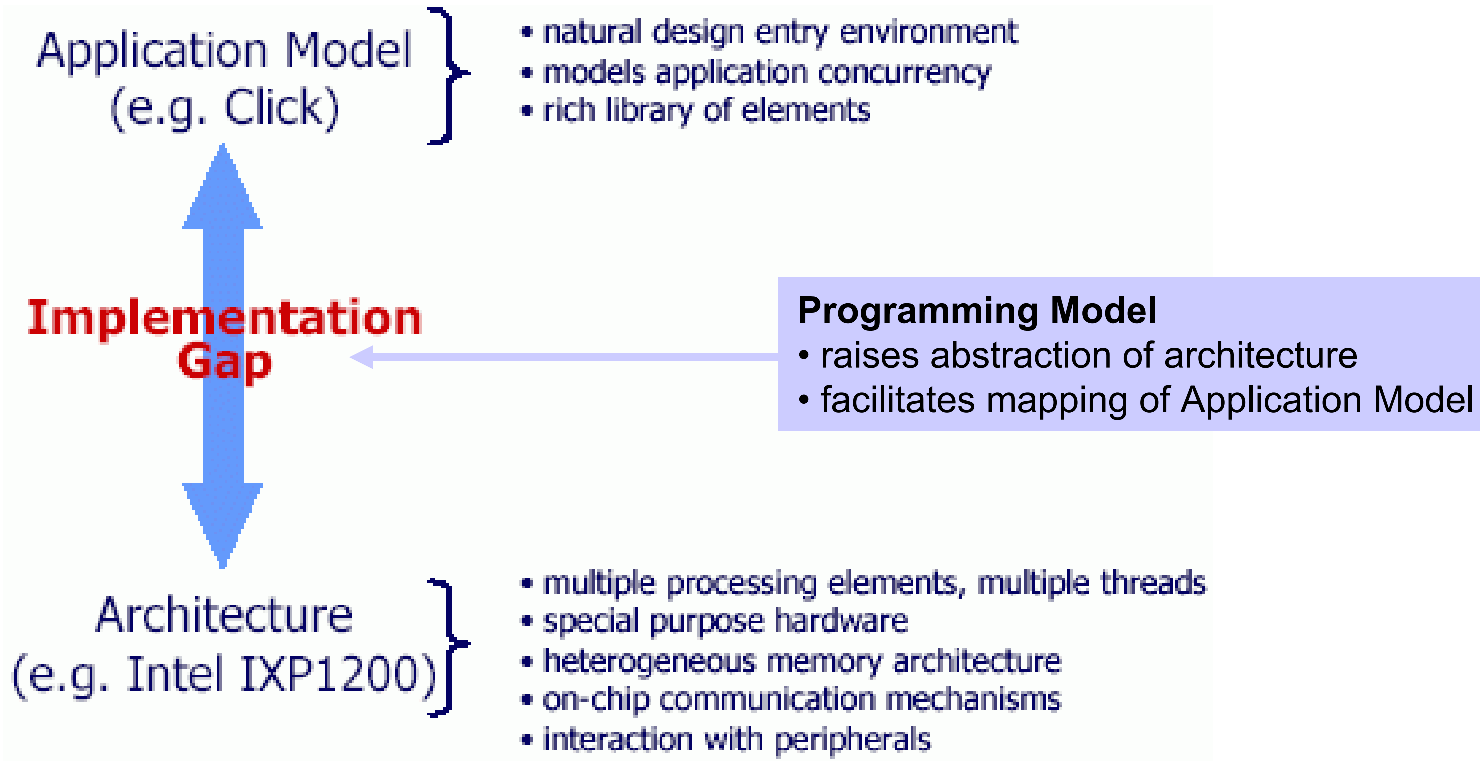
- An integrated software/system development environment that allows us to determine rapidly if an architecture meets the needs of the customer within a few weeks to a few months!



Jim Brodie, Manager, Performance Excellence, Motorola Global Software Group at CODES+ISSS'03 SLD tools panel

System-Level Programming Model

“*The mismatch of concerns between the application and the target architecture results in an implementation gap. To facilitate bridging this gap we introduce an intermediate layer called a "Programming Model," which presents an abstraction of the underlying architecture while still providing a natural way of describing applications*.”



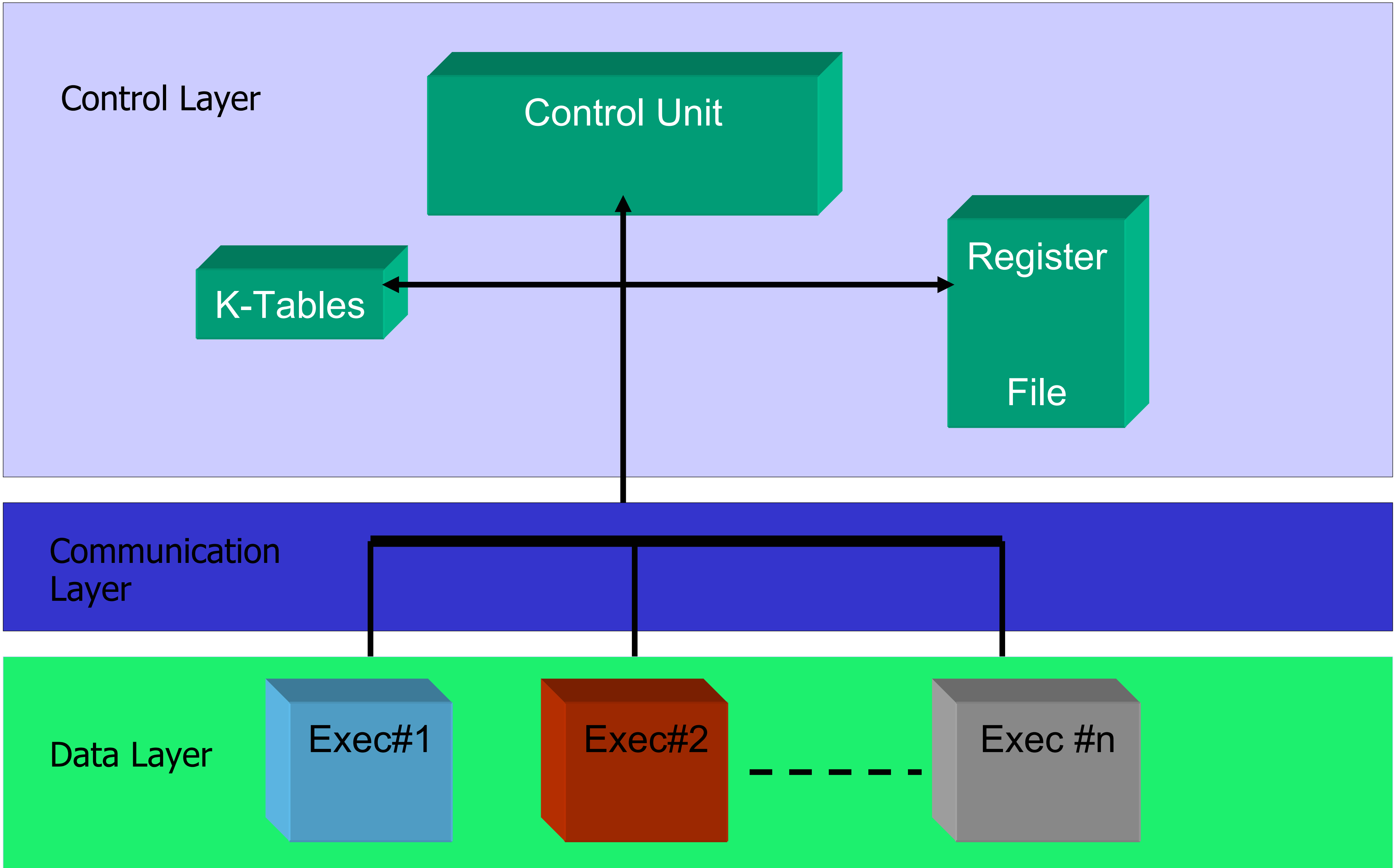
source: http://www.gigascale.org/mescal/progmodel.html
http://www.gigascale.org/pubs/356.html

Our Goal

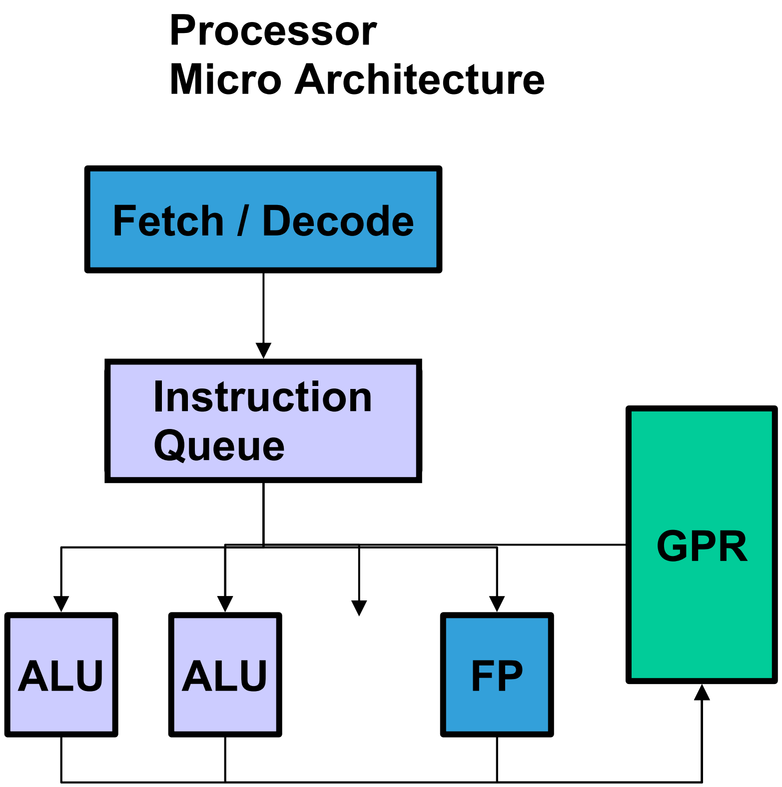
- Streamline the design of modern heterogeneous multi-processor systems by providing
  - An Architectural Framework,
  - A Programming Model,
  - Mapping Methodology and Tools

Multi-Level Computing Architectures

MLCA in general

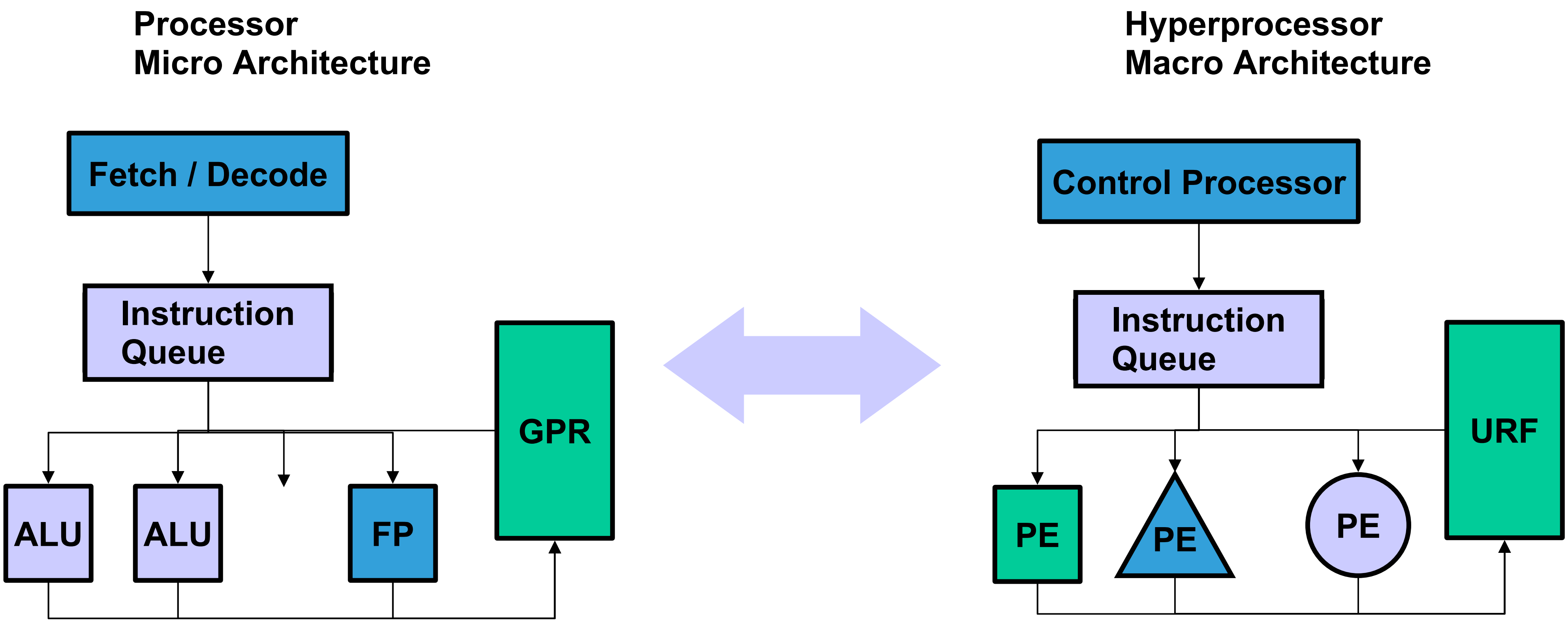


Micro- and Macro-Architecture Analogy



ILP
(Instruction-Level Parallelism)

Micro- and Macro-Architecture Analogy

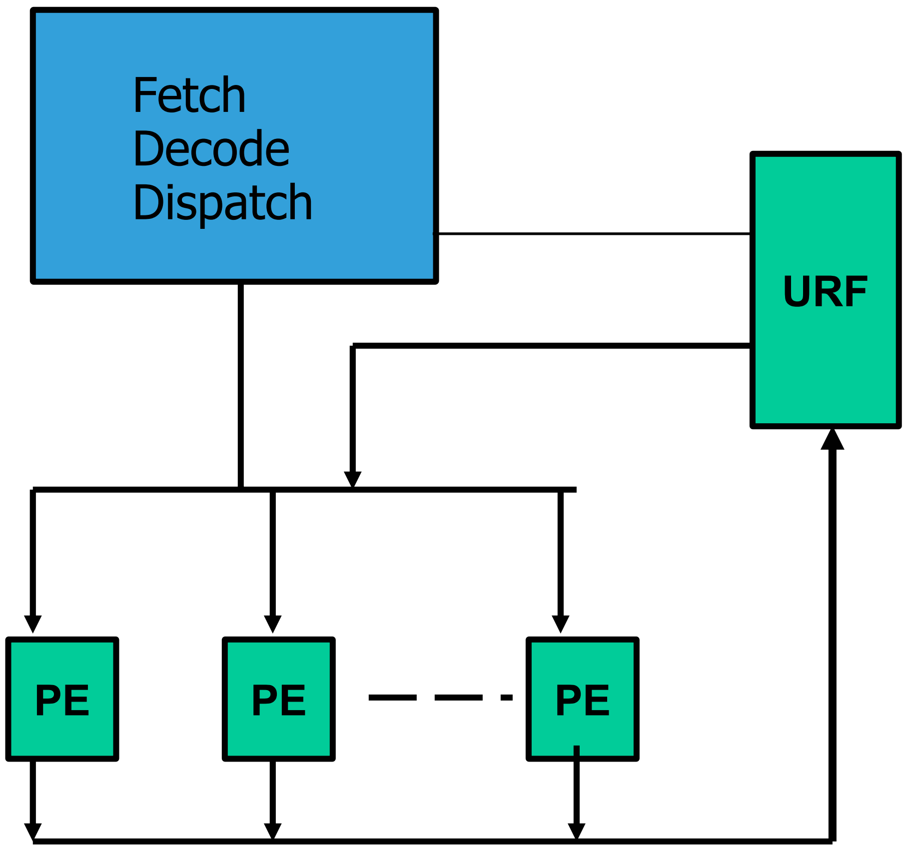


ILP
(Instruction-Level Parallelism)

TLP
(Task-Level Parallelism)

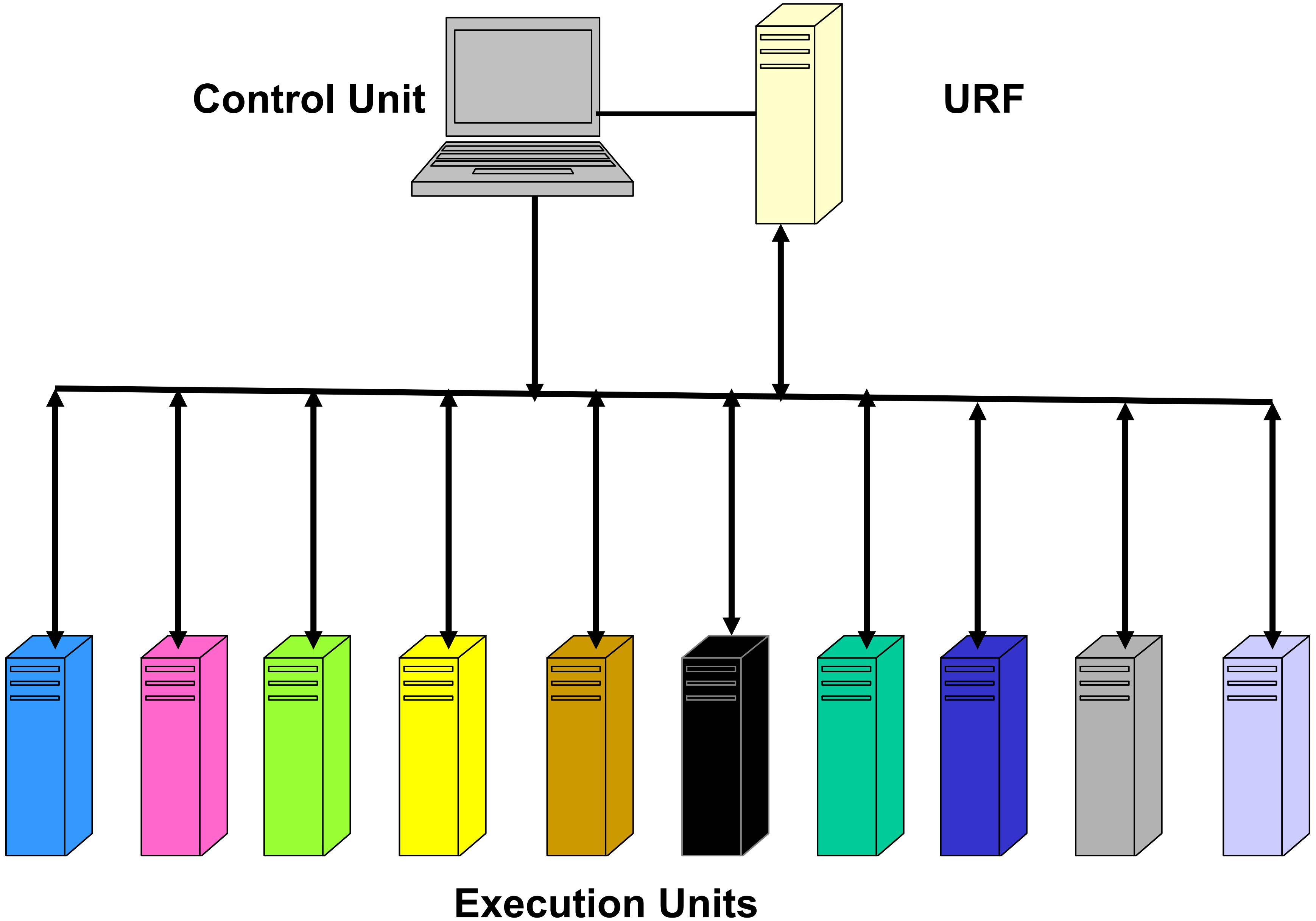
Hyperprocessor

Control Processor



Processing Elements

PE = Processor,
FPGA
Hardware

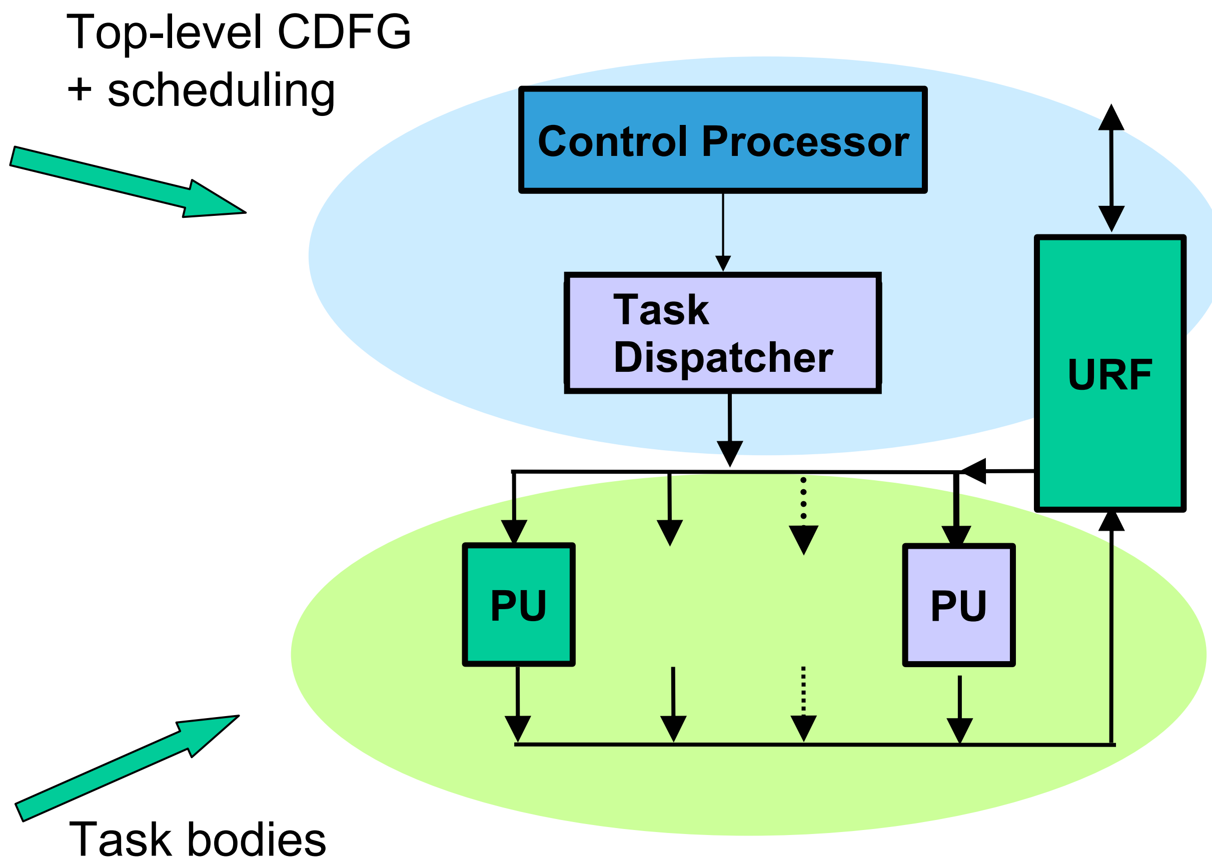


(Hypercomputer)

Layered Programming

```
main()
{
  …
  while (GetFrame(buf1)) {
    QAM(buf1, buf2, buf3);
    CIC_I(buf2, buf4);
    CIC_Q(buf3, buf5);
  …
}
```

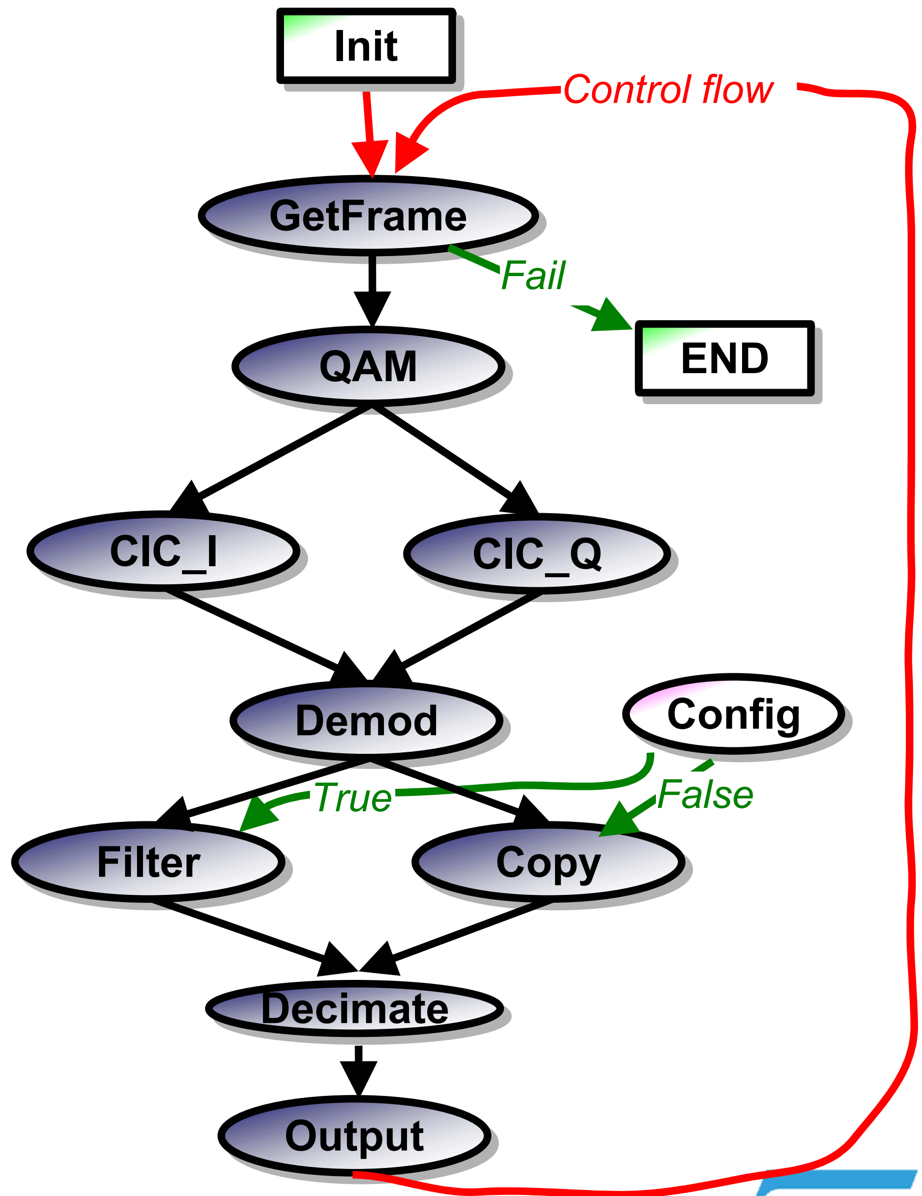
```
void
QAM(int *ptr1, …)
{
  …
}
void
CIC_I(int *ptr1, …)
{
  …
}
```



Sample C Program

C (partitioned manually)

```
#define N 1024
bool doFilter;
int32 buf1[N], buf2[N], … ,buf8[N];
doFilter = Config();
while (GetFrame(buf1)) {
  QAM(buf1, buf2, buf3);
  CIC_I(buf2, buf4);
  CIC_Q(buf3, buf5);
  Demod(buf4, buf5, buf6);
  if (doFilter) {
    Filter(buf6, buf7);
  } else {
    Copy(buf6, buf7);
  }
  Decimate(buf7, buf8);
  Output(buf8);
}
```



Sample HISA Program

C (partitioned manually)

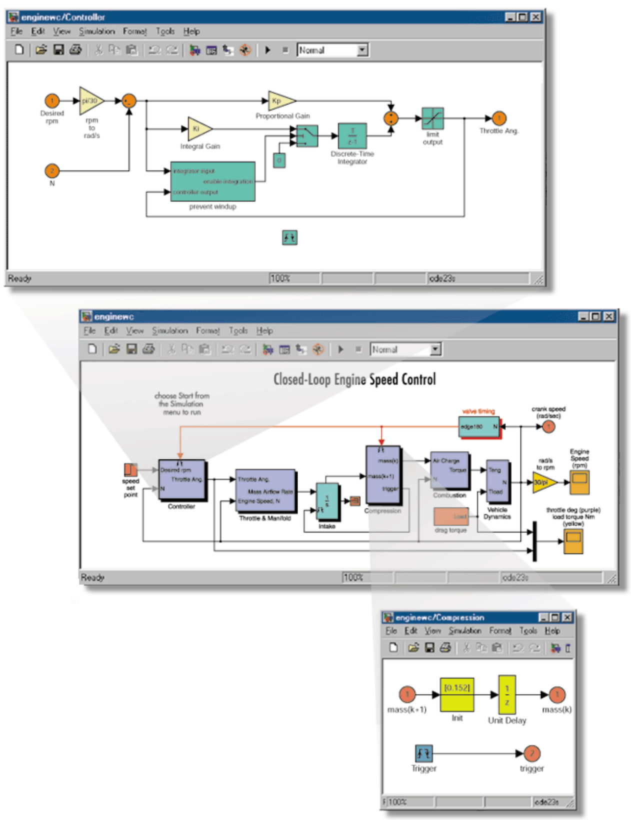
```
#define N 1024
bool doFilter;
int32 buf1[N], buf2[N], … ,buf8[N];
doFilter = Config();
while (GetFrame(buf1)) {
  QAM(buf1, buf2, buf3);
  CIC_I(buf2, buf4);
  CIC_Q(buf3, buf5);
  Demod(buf4, buf5, buf6);
  if (doFilter) {
    Filter(buf6, buf7);
  } else {
    Copy(buf6, buf7);
  }
  Decimate(buf7, buf8);
  Output(buf8);
}
```

Hyperprocessor ASM

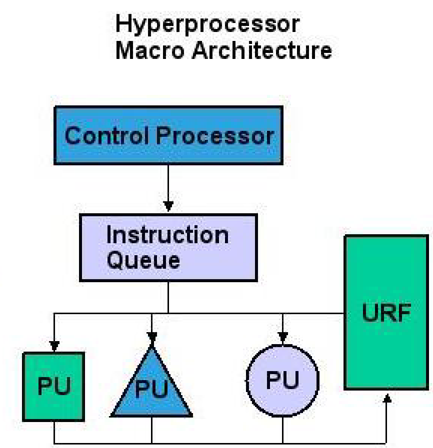
```
task Init
task Config, CR2
L_TOP:
task GetFrame, CR3, R1:w
if true (CR3 != fail) jmpa L_End
task QAM, R1:r, R2:w, R3:w
task CIC_I, R2:r, R4:w
task CIC_Q, R3:r, R5:w
task Demod, R4:r, R5:r, R6:w
if (CR2 == true)
      task Filter, R6:r, R7:w
if (CR2 == false)
      task Copy, R6:r, R7:w
task Decimate, R7:r, R8:w
task Output, R8:r
jmpa L_TOP
L_END:
```

Enabler for System-Level Compiler

“The System is the CPU”




```
task Init
task Config, CR2
L_TOP:
task GetFrame, CR3, R1:w
if true (CR3 != fail) jmpa L_End
task QAM, R1:r, R2:w, R3:w
task CIC_I, R2:r, R4:w
task CIC_Q, R3:r, R5:w
task Demod, R4:r, R5:r, R6:w
if (CR2 == true)
      task Filter, R6:r, R7:w
if (CR2 == false)
          task Copy, R6:r, R7:w
task Decimate, R7:r, R8:w
task Output, R8:r
jmpa L_TOP
L_END:
```



TLP
(Task-Level Parallelism)

HyperCompiler

- Develop and implement compiler technology to port applications to the Hyperprocessor.

- Reduces port time
- Enhances portability and reduces errors
- Reduces cost

Uniform Architecture from Smallest Processing Element to Giant Compute Farm

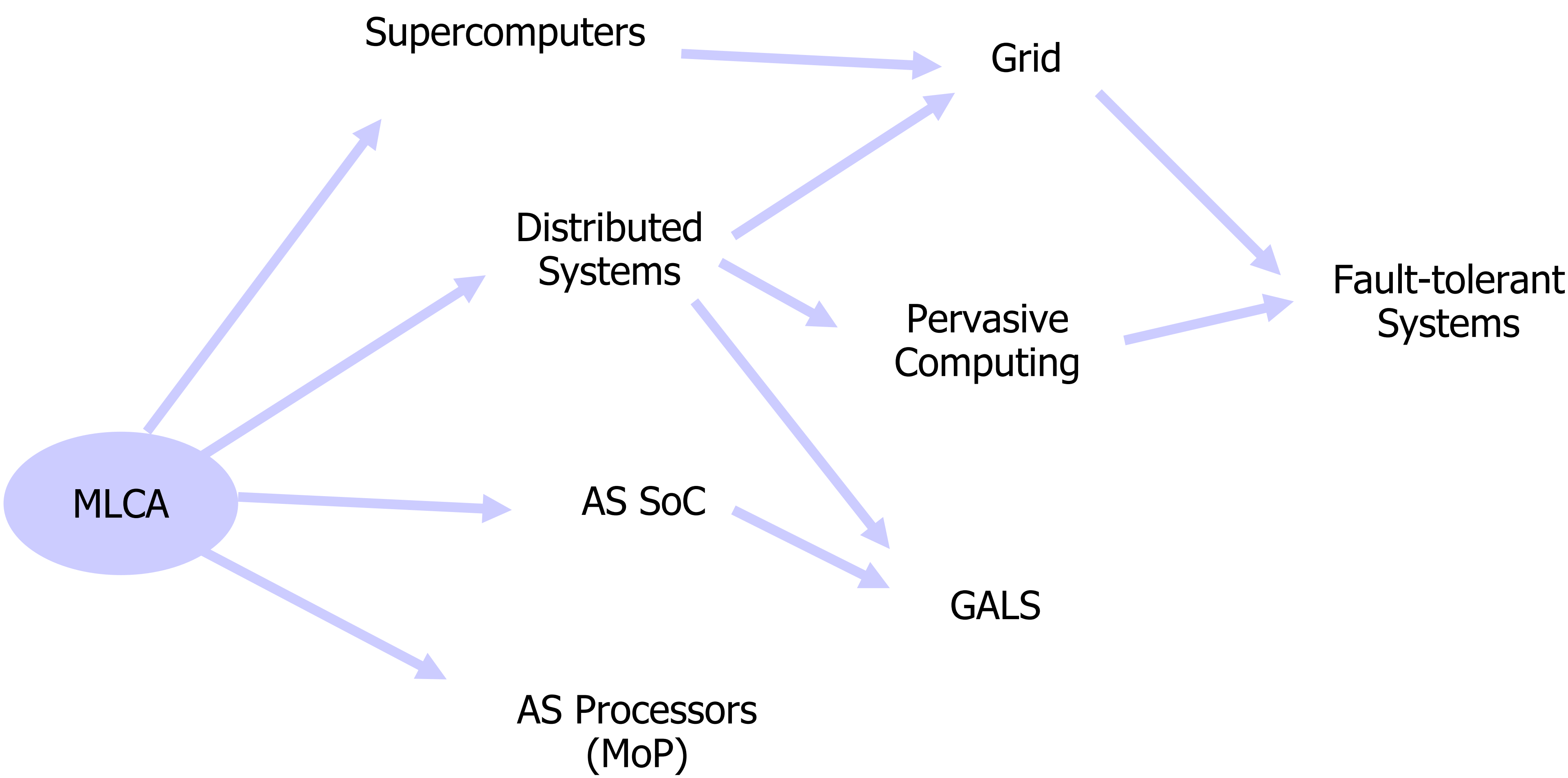
Parts of MLC	Microprocessor	Hyperprocessor	Hypercomputer
Control Unit	Instruction decode and K-table Dispatch	Task decode and K-table Dispatch	Task decode and K-table Dispatch
URF	USE GPR +dirt	URF + dirty bits	Computer to do URF
Execution Units	Multiple ALUs	Many Microprocessors	Many computers

MLCA is a unique Architecture

Derived from Basic idea of a Microprocessor, namely the Superscalar and Multiscalar architectures to formulate a unique computing Machine and solve today's complex problems.

Attributes	Multiprocessor	Multiscalar	MLCA
Task Determination	Static	Static and Dynamic	Static and Dynamic
Static guarantee of inter-task control independence	Required	Not required	Not required
Static guarantee of inter-task data independence	Required	Not required	Not required
PE types	Homogenous or Heterogeneous	Homogenous	Homogenous or Heterogeneous
PE organization	Tightly, loosely, or distributed	Tightly in Circular fashion	Tightly, Snugly, or distributed
Medium for inter-task communication	Memory, or message passing	Register and memory	Universal Register file, local register files, and storage
Register space	Distinct	Common	Distinct and Common
Memory space	Common or distinct	Common	Common or distinct ??
Speculative	No	Yes	Yes

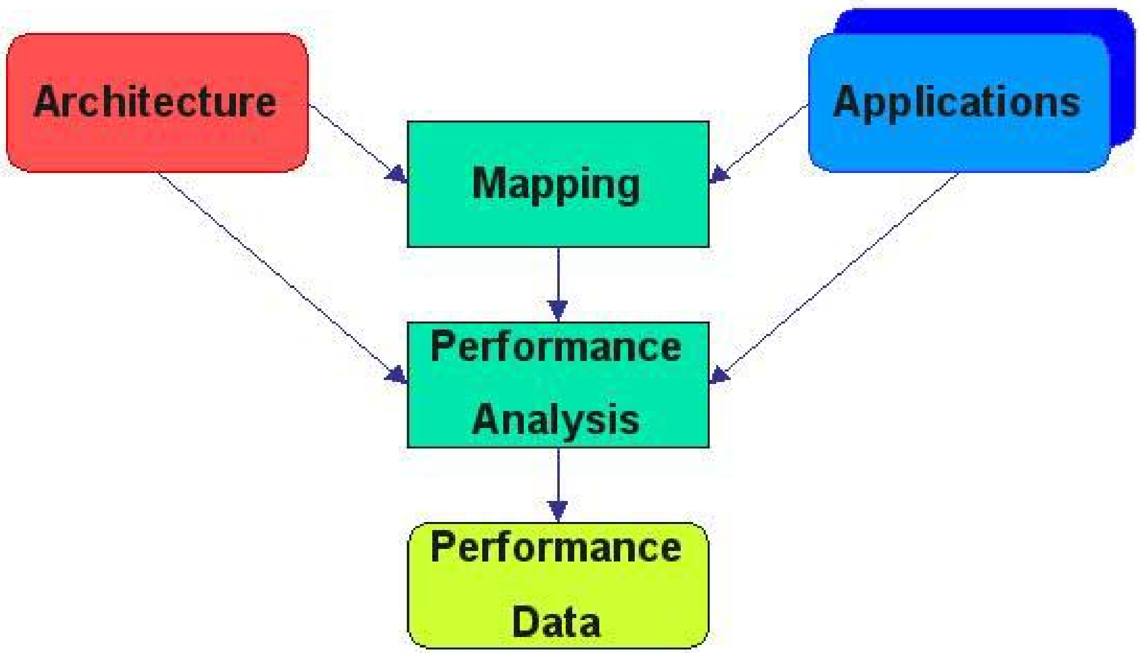
A World of Opportunities



Design Methodology

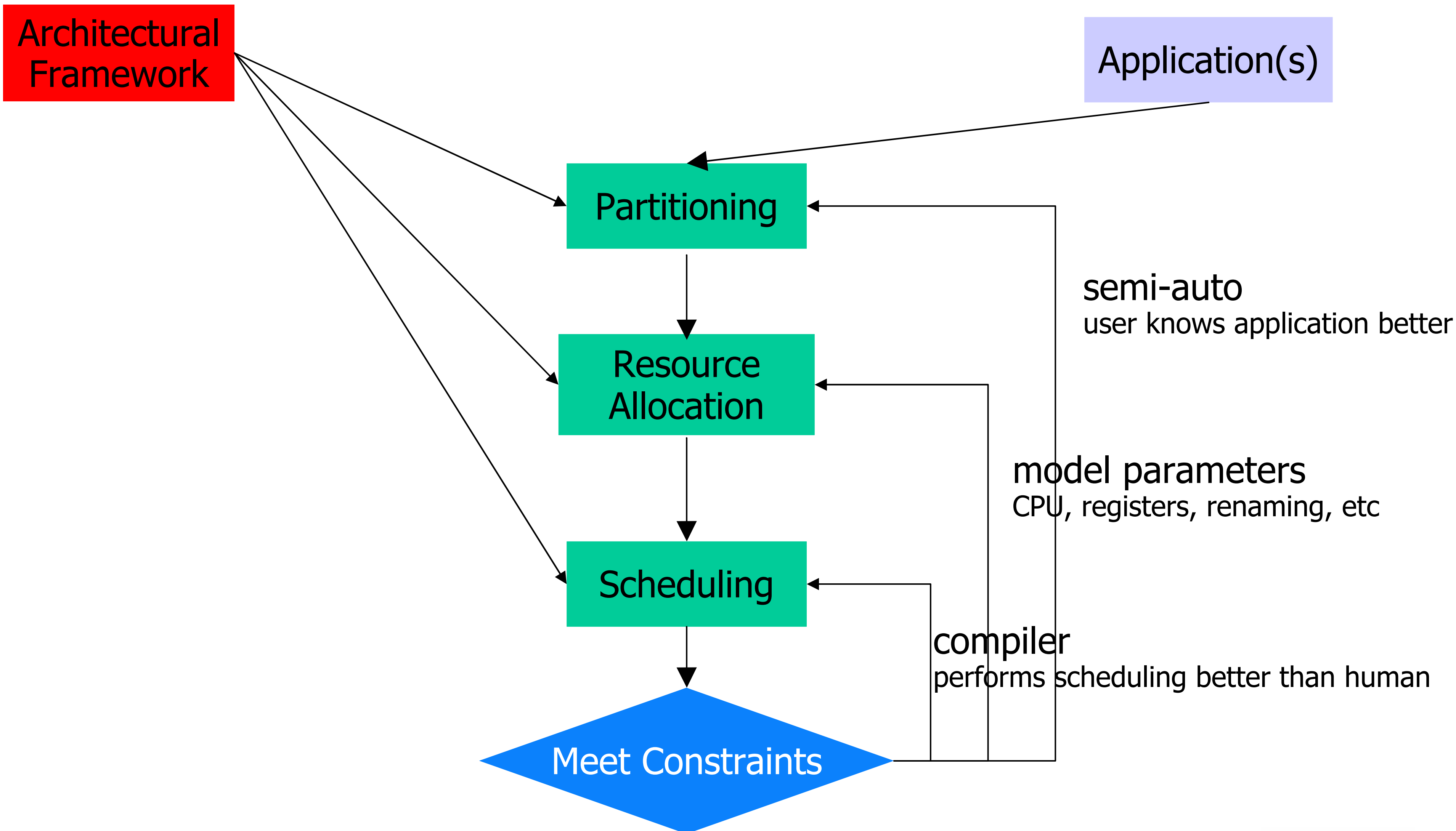
Architecture Design: Y - Chart

Benchmark driven, based on a Set of Applications



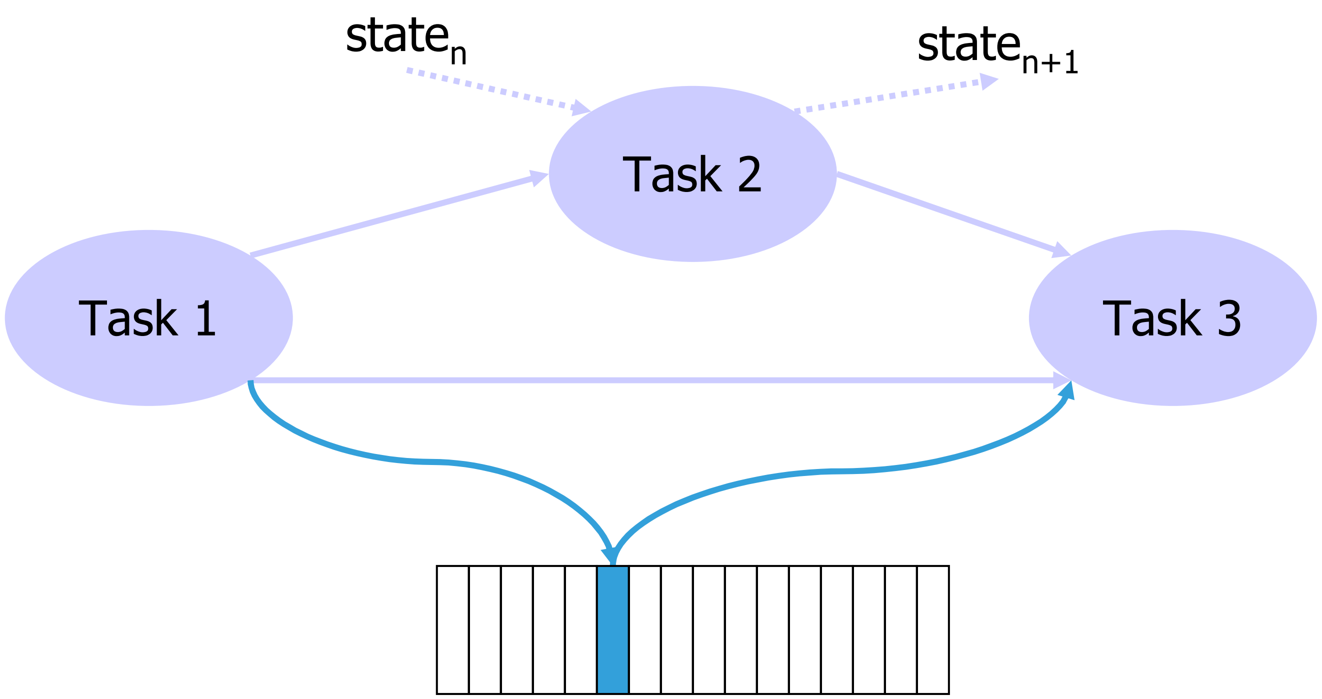
ST, October 27 2003, La Jolla CA, K. Vissers 2-30

Design Methodology – 2



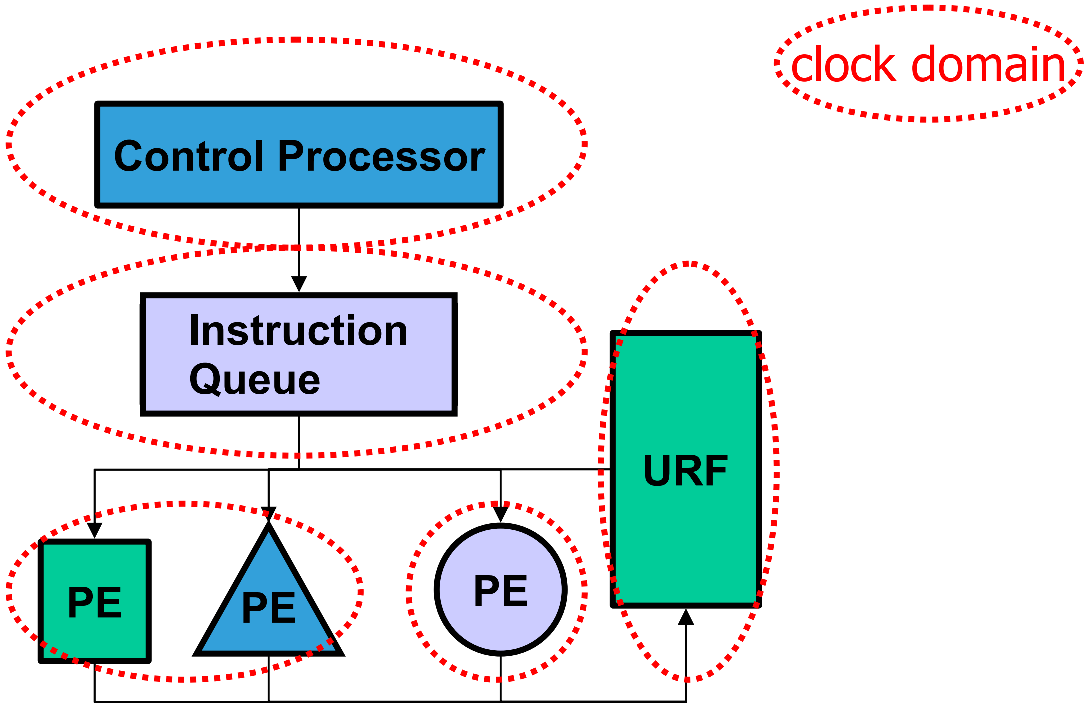
Programming Model

- Similarities with coarse-grain dataflow
  - Actors ⇒ task (copy-in, copy-out)
- But more flexible communication
  - Arc ⇒ URF register
  - Renaming will allow speedup



GALS

Globally Asynchronous Locally Synchronous



- Programming Model does not assume synchronous communications
- Allows for software control of Voltage/frequency

What's MLC?

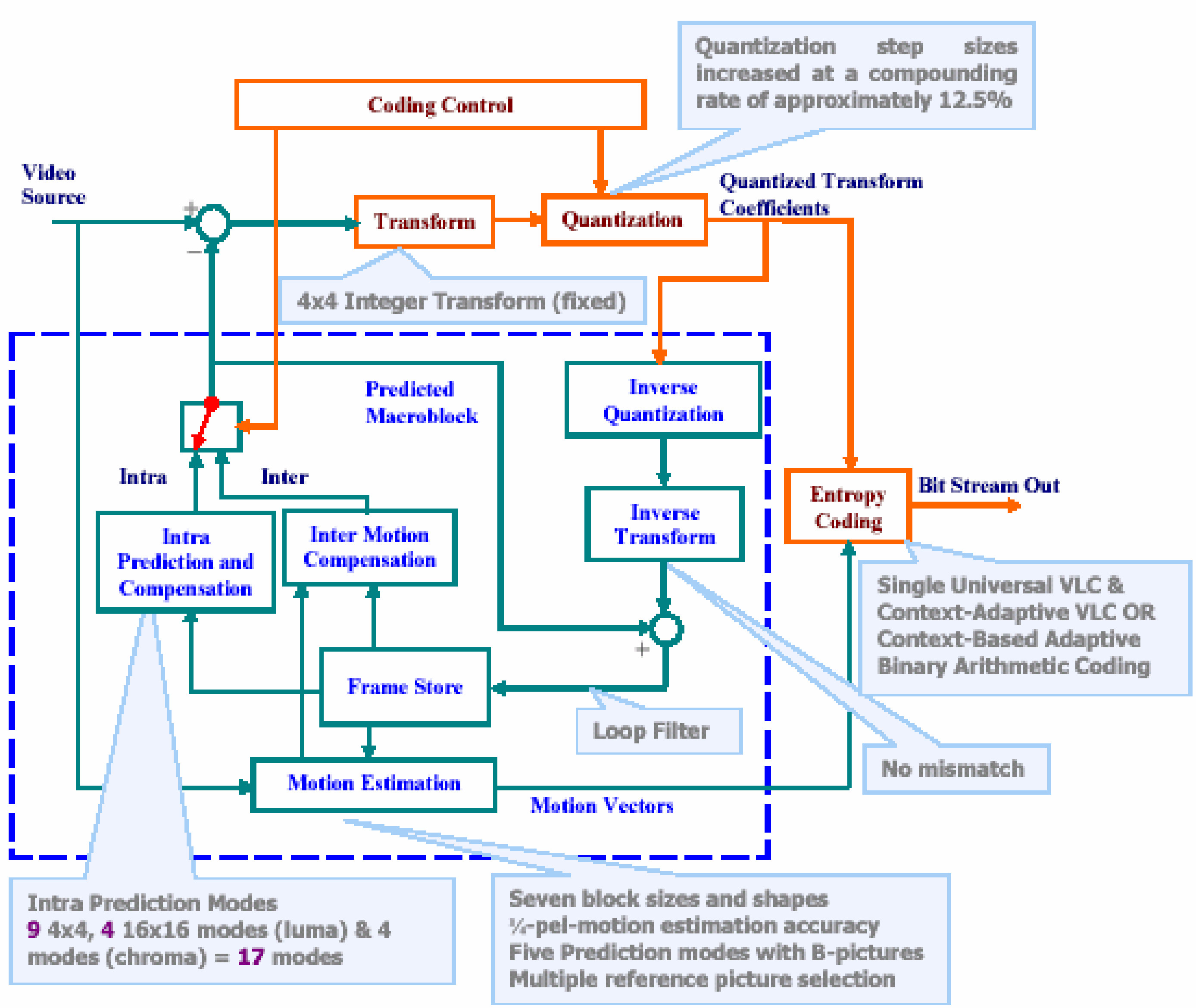
What do we have?

- A powerful abstraction ("programming model")
  - pseudo-sequential programming style
  - usable by compilers (see work with UofT)
  - Independent of memory architecture
    - memory architecture is dictated by application
    - does not rely on memory coherency
  - based on coarse-grain dataflow
  - fosters modular programming
- An architecture framework that matches the programming model
  - natural mapping
  - implementation (HW/SW) depends on application domain
- A tool for exploration: functional model
  - profile applications
  - explore design space
    - computation, storage, communication, scheduling

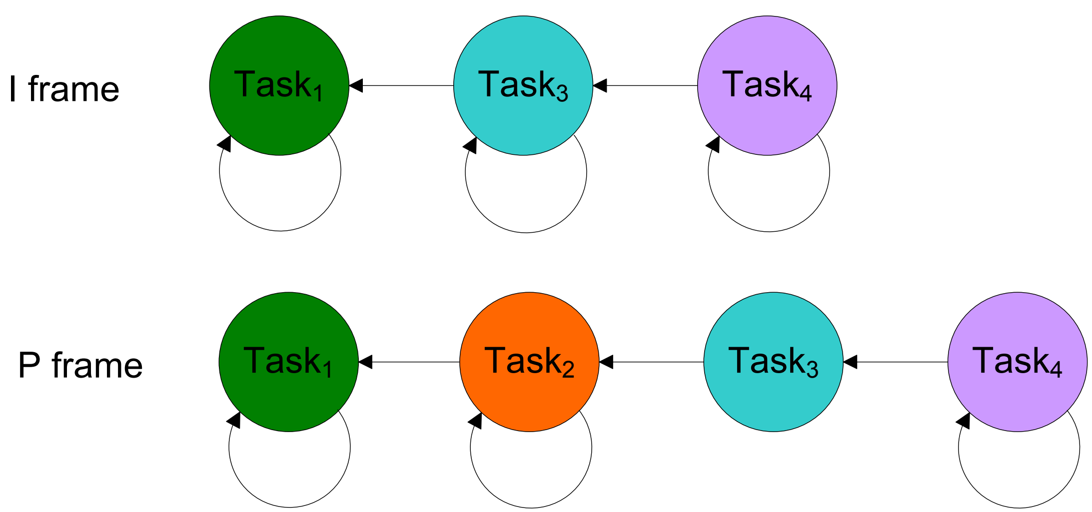
An Example: H.264

- Many ways to be H.264 compliant (45 levels so far)
- Implementation of highest levels is still unsolved
- Imagine an objective: develop an H.264 application
- Steps to take:
  - Develop an architecture for heterogeneous processors – check
  - Develop programming model for processor architecture – check
  - Construct block model of application
  - Partition and profile application
  - Scheduling
  - Test

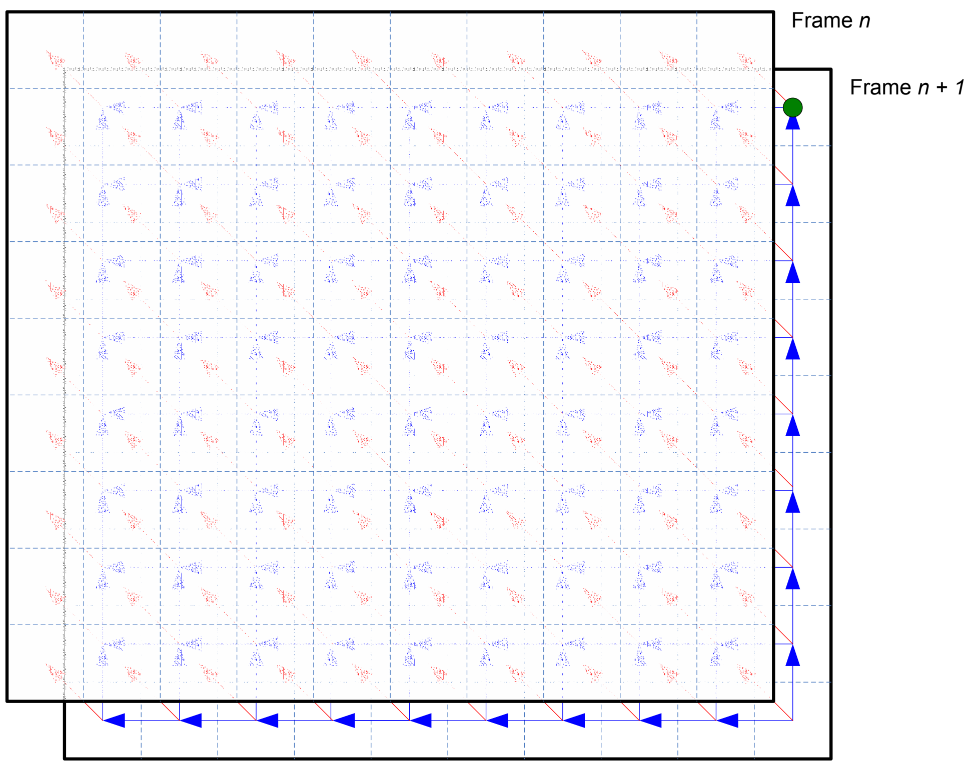
H.264 Block Diagram



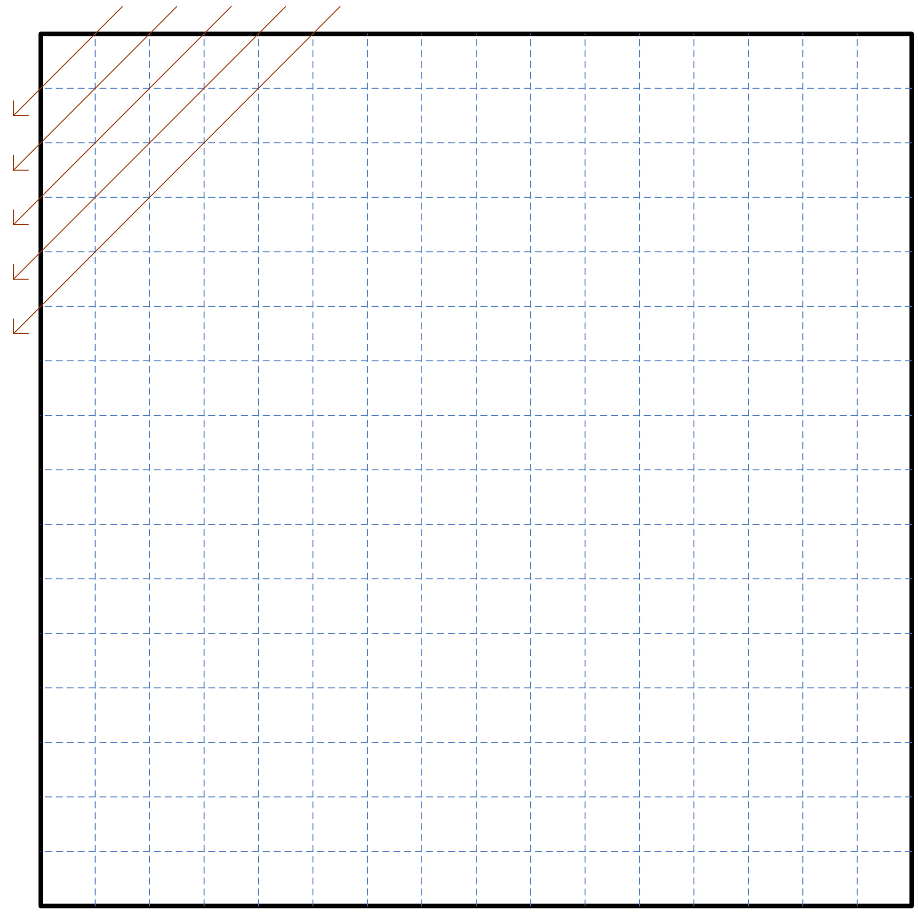
Example Profile Result



Examine Task Dependency (for scheduling)



Schedule Tasks

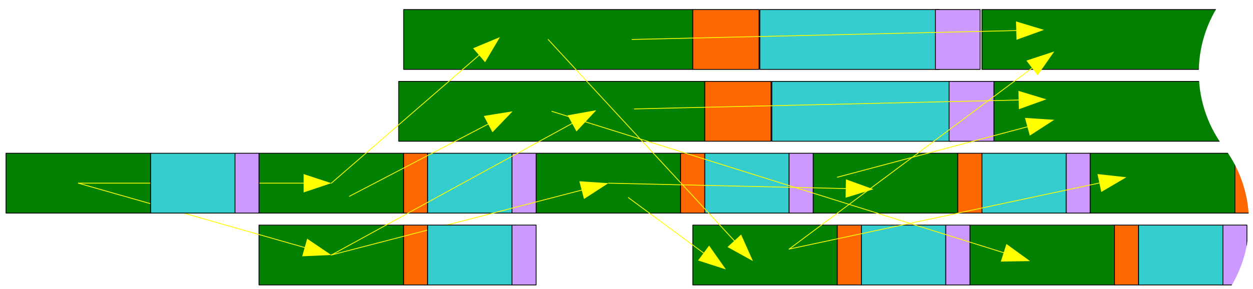


a_{11}

a_{21}

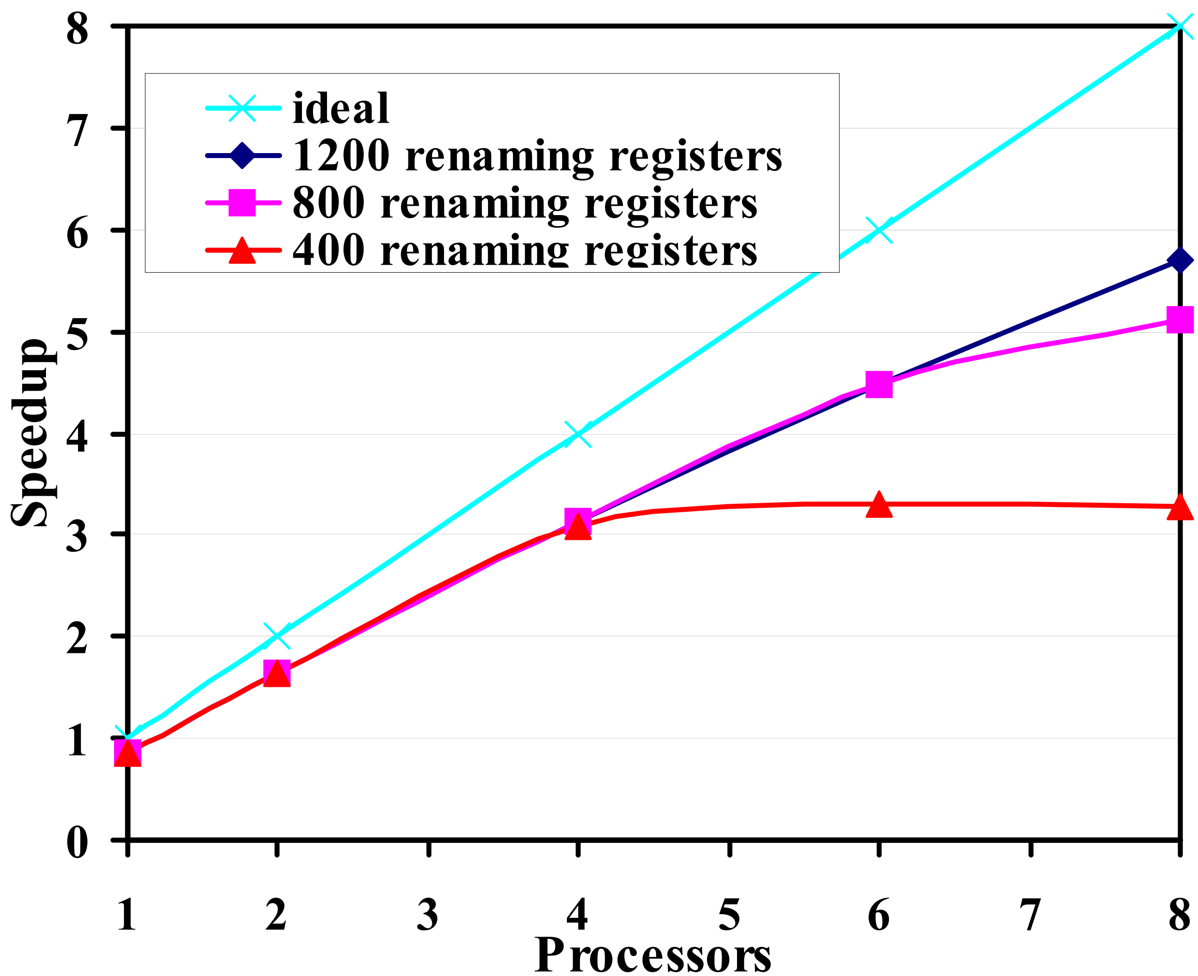
a_{31}

Prototype Machine



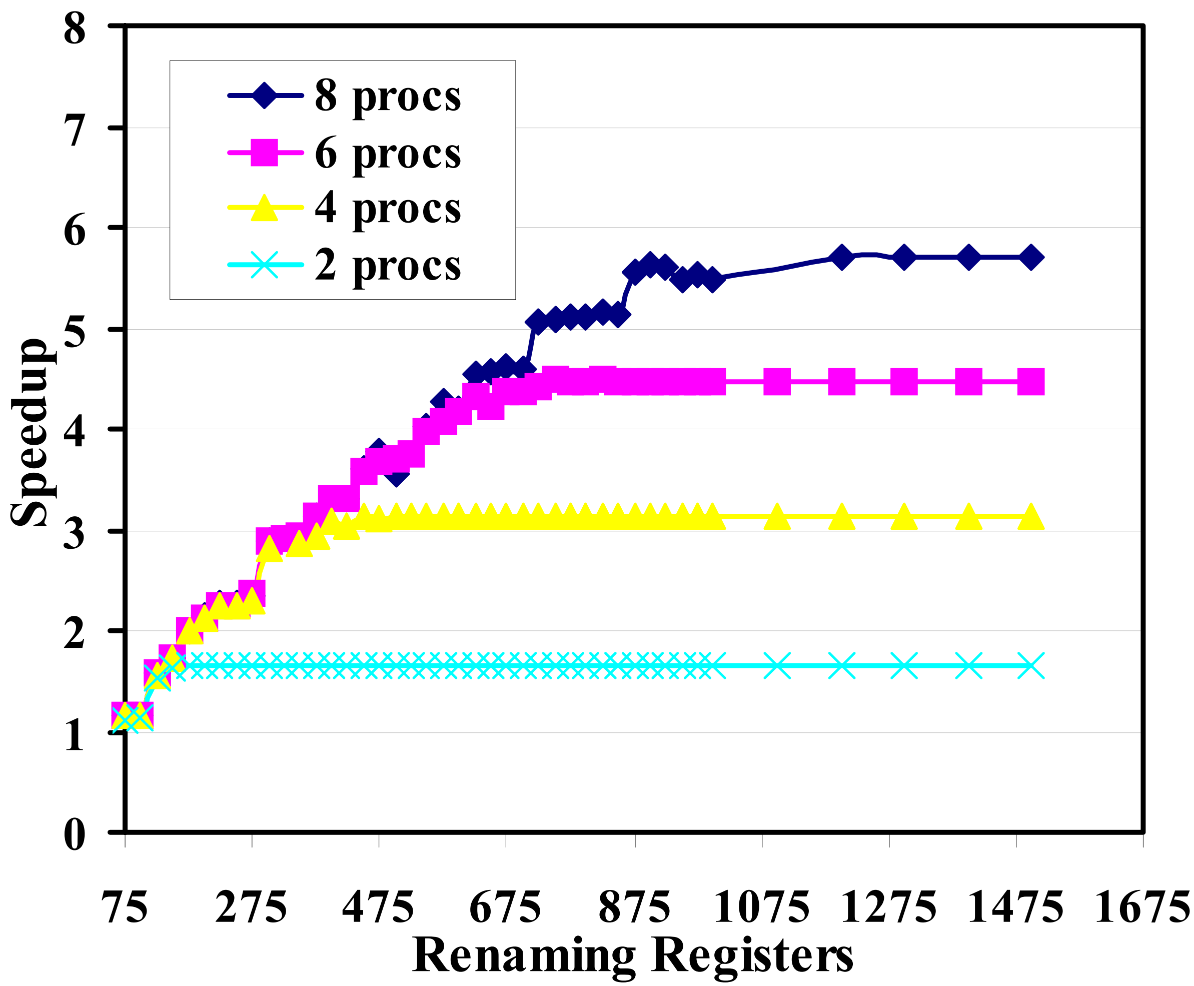
- Assume 4-processor machine
  - 2 fast (DSPs) and 2 slow (programmable)
- Machine performance is easily estimated
  - If performance is not satisfactory, add/upgrade processors
  - Else, remove/downgrade processors

MP3 Decoder: Parallel Speedup



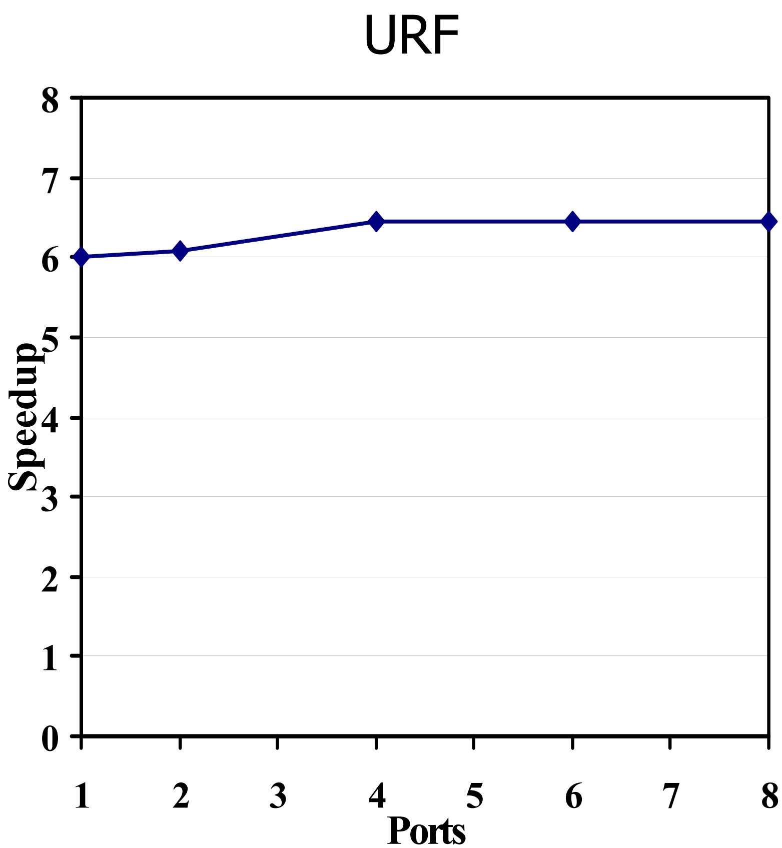
- Scaling performance particularly with a large number of renaming registers.

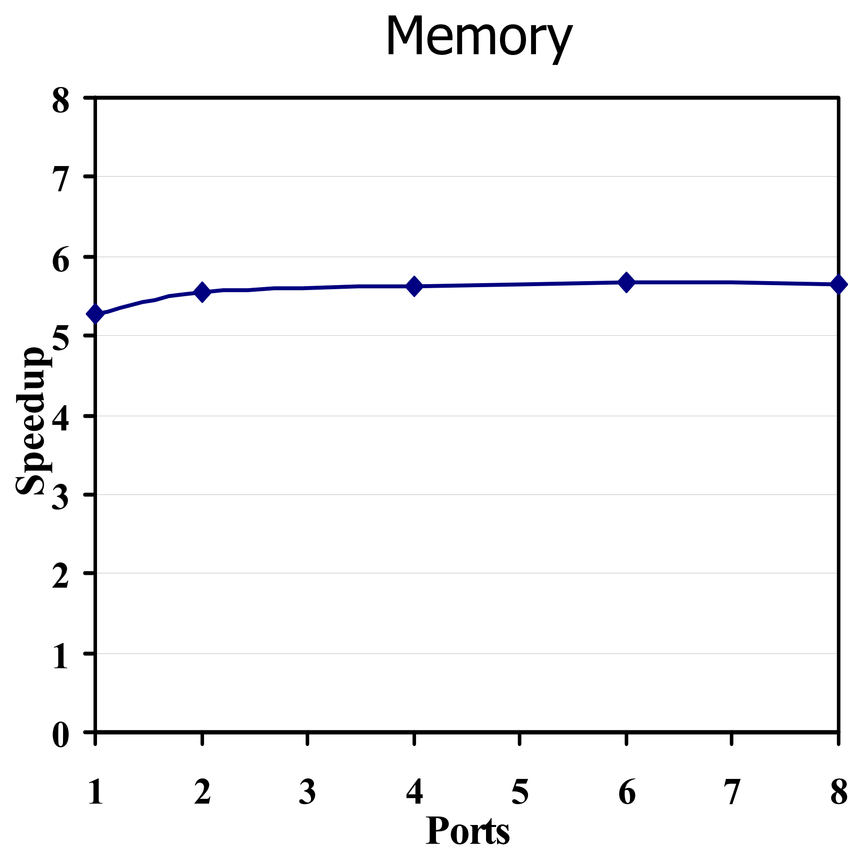
MP3 Decoder: Renaming Registers



- Performance increases up to a “breakpoint”.
- Performance may decrease with more renaming registers!

MP3 Decoder: Resource Contention





- Little if any contention over URF registers and memory.

Research Challenges

- How to form the control program and the tasks.
  - *The task formation problem*
- How to improve performance.
  - *The task optimization problem & Resource assignment*
- How to schedule tasks to improve performance and reduce power.
  - *The task scheduling problem*
- How to synthesize a Hyperprocessor instance for a given set of applications.
  - *The synthesis problem*

On Going research activities

- *HyperCompiler*
  - *University of Toronto*
- *Applications*
  - *UCLA, PolitecMilano, UCSD, and ST*
- *Fault-tolerance*
  - *UCSD*
- *High Level Modeling*
  - *CMU*
- *HyperComputer*
  - *AST lab*
- Future Work
  - *Mobile Supercomputing*
  - *Hyper OS*
  - *More industry standard Applications*
  - *Development Methodology*

Conclusion

- MLCA is natural evolution of microprocessor and system design to merge into a unified architecture.
- It gives the system image of single processor
  - Simplifies design
  - Simplifies programming
  - Makes multiprocessing easy
- Provides the system houses with top to bottom design methodology.
- Speeds up system design and TTM
- RAS Issues

Thank You