Pangaea: A Tightly-Coupled Heterogeneous IA32 Chip Multiprocessor

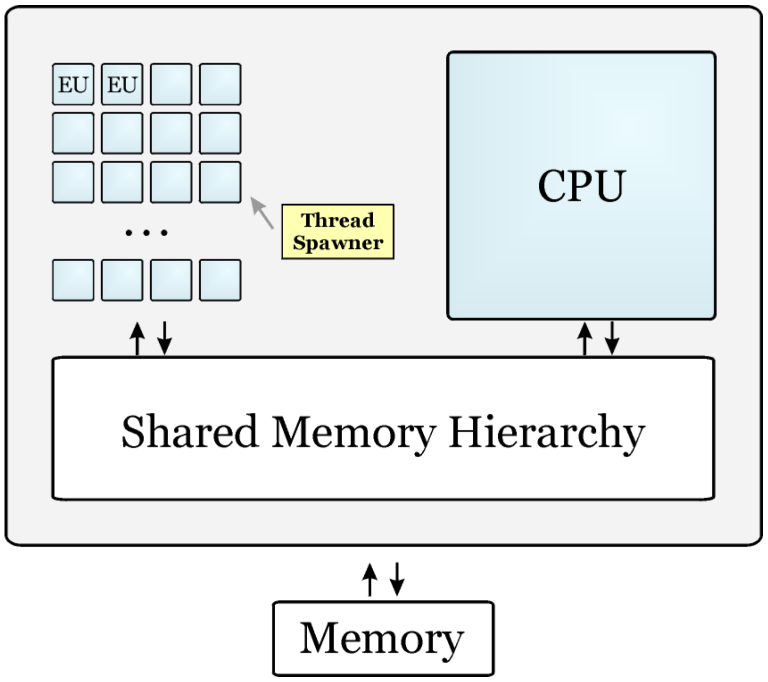
Henry Wong[1], Anne Bracy[2], Ethan Schuchman[2], Tor M. Aamodt[1], Jamison D. Collins[2], Perry H. Wang[2], Gautham Chinya[2], Ankur Khandelwal Groen[3], Hong Jiang[4], Hong Wang[2]
henry@stuffedcow.net, anne.c.bracy@intel.com

[1]Dept. Of Electrical and Computer Engineering, University of British Columbia
[2]Microarchitecture Research Lab, Microprocessor Technology Labs, Intel Corporation
[3]Digital Enterprise Group, Intel Corporation
[4]Graphics Architecture, Mobility Groups, Intel Corporation

Parallel Architectures and Compilation Techniques, October 27, 2008

Pangaea

- Integrates IA32 CPU with GPU cores
- Improved area/power efficiency
- Tighter integration
- Modular design



Motivation

- GPUs have low Energy Per Instruction
  - ~100x less EPI than CPU
  - Parallel performance too
  - Pangaea targets non-graphics computation for further efficiency gains
- Tightly-coupled
  - easier to program
  - lower communication latency
- Minimize changes to existing software (OS)

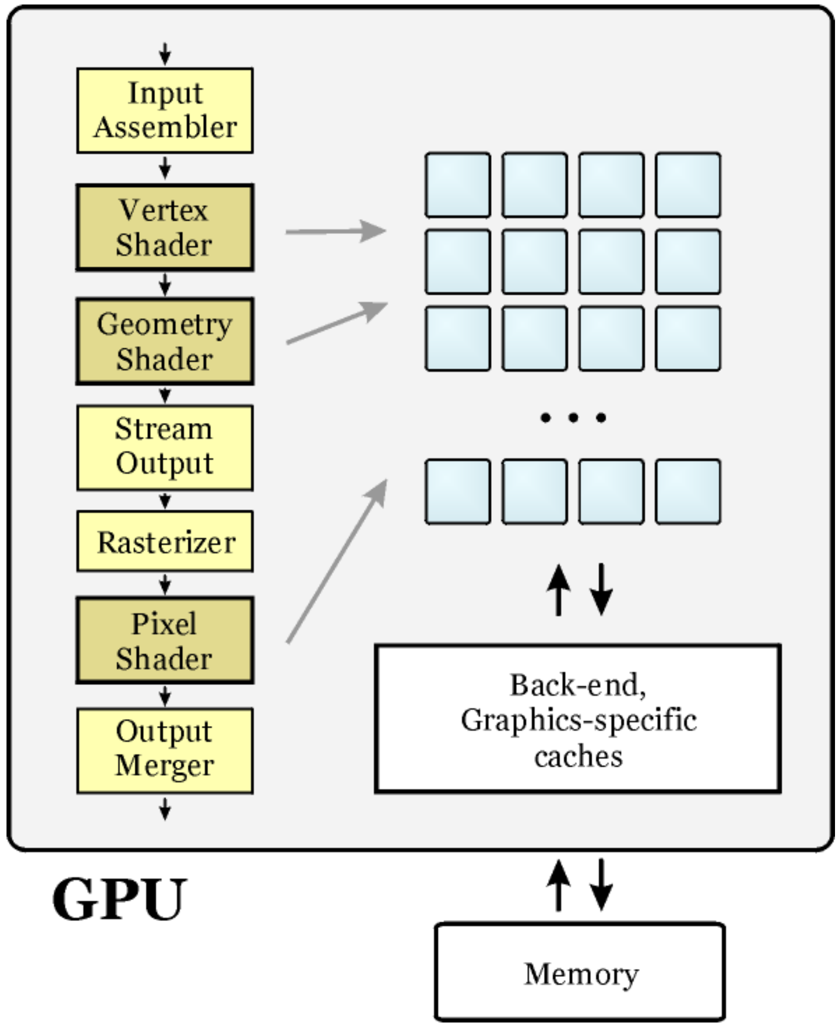
Overview

- Background on GPU Computation
- Pangaea: IA32-GPU chip multiprocessor
  - User-Level Interrupt mechanism
  - Architecture trade-offs
  - Prototype performance
- Conclusion

Programmable GPU

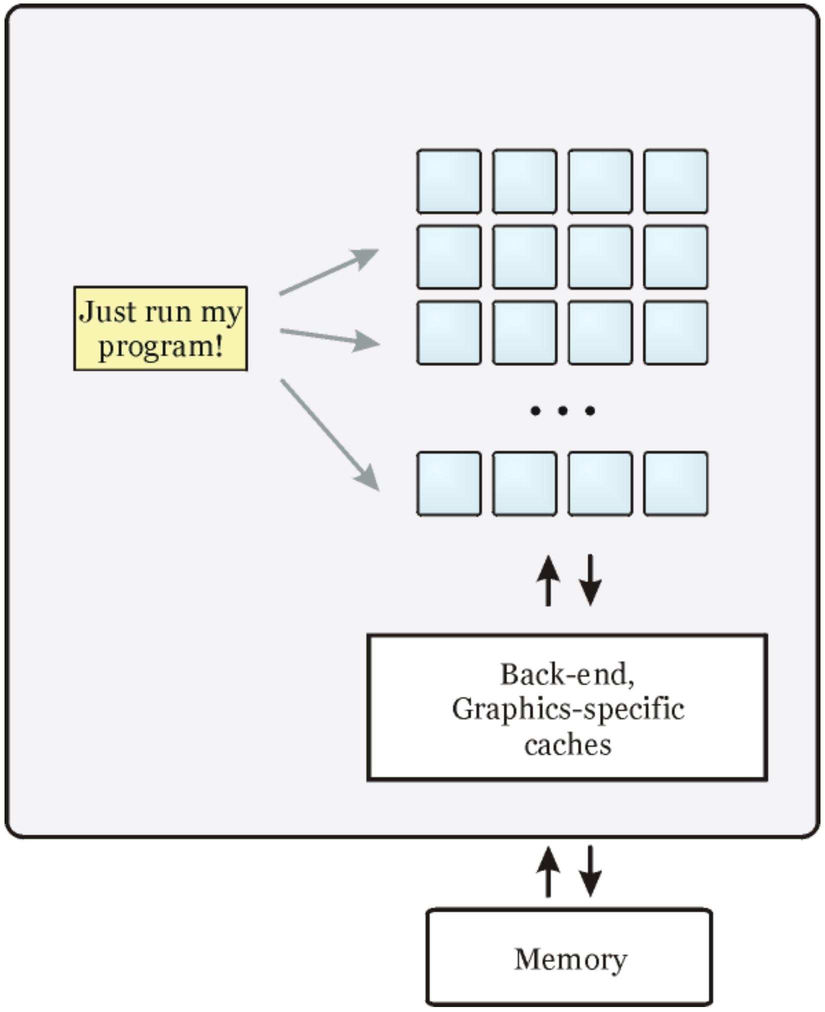
- Rendering pipeline
  - Polygons go in
  - Pixels come out
- DX10 has 3 programmable stages

☐	Fixed Graphics Pipeline Stage
▨	Programmable Graphics Pipeline Stage



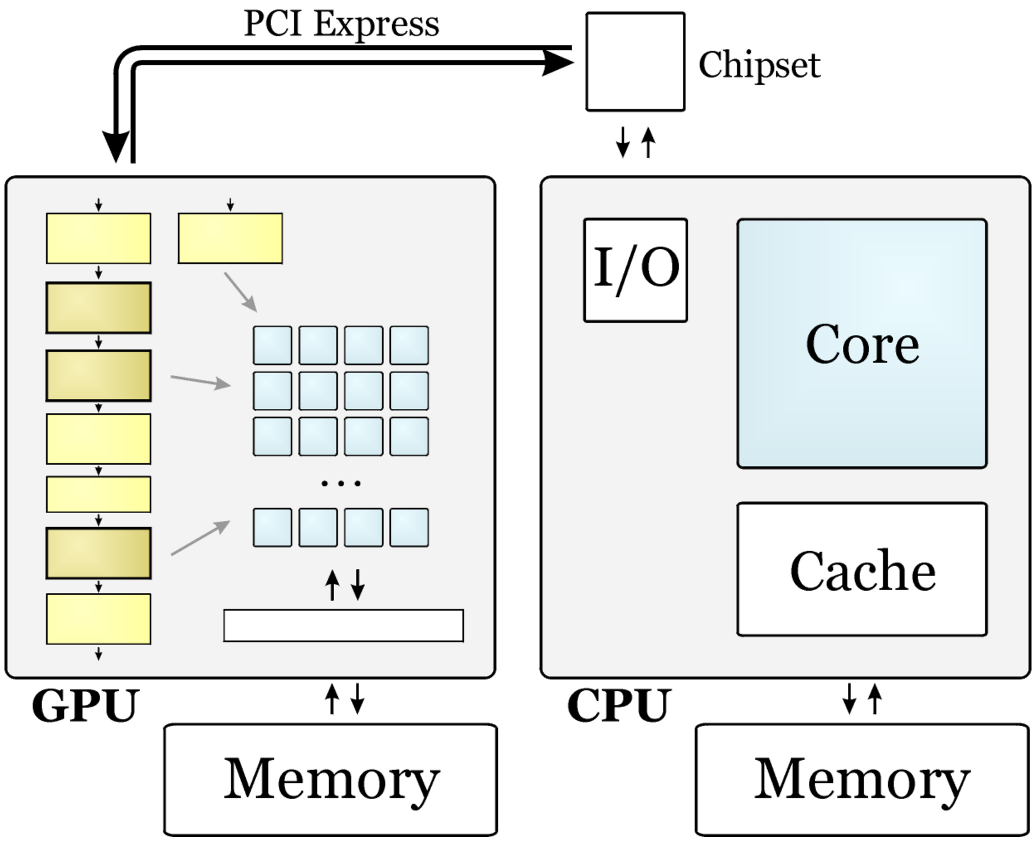
Nvidia CUDA, AMD Stream

- Use shader processors without graphics API
- C-like high-level language for convenience



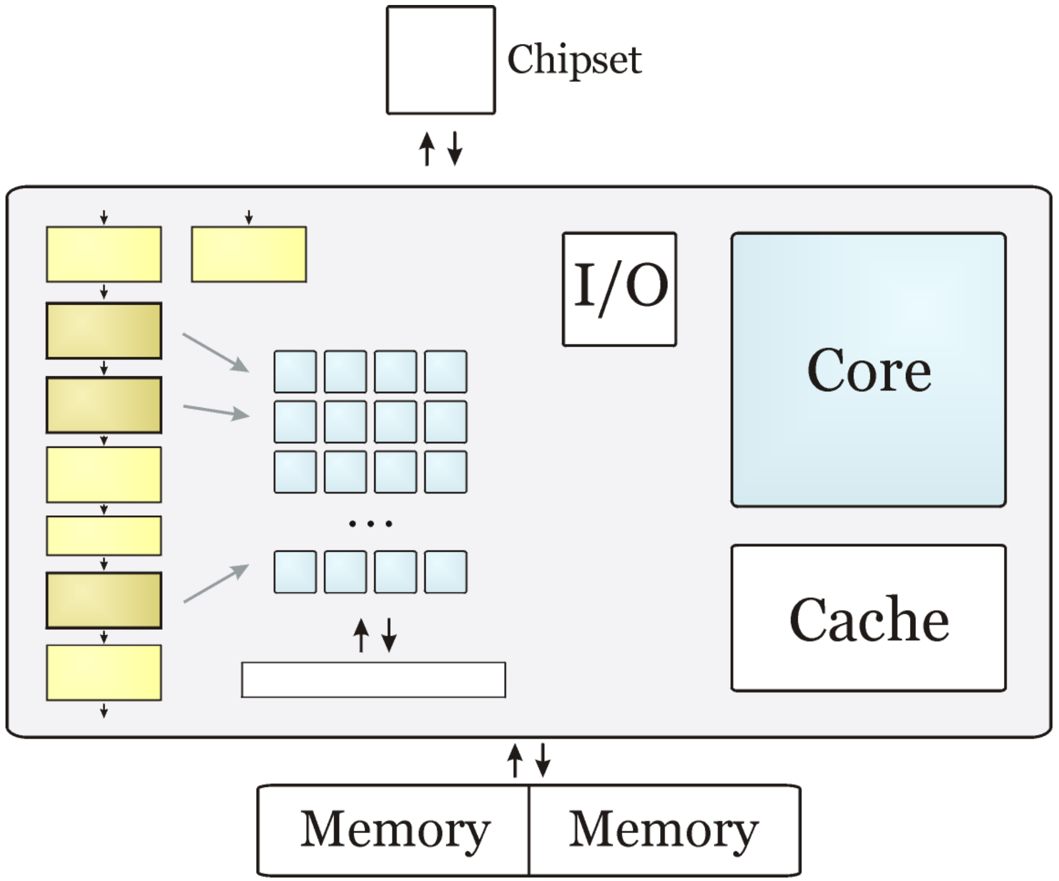
GPU + CPU

- Loosely-coupled to the CPU
  - Off-chip latency
  - Explicit data copy between memory spaces
  - Cooperation?



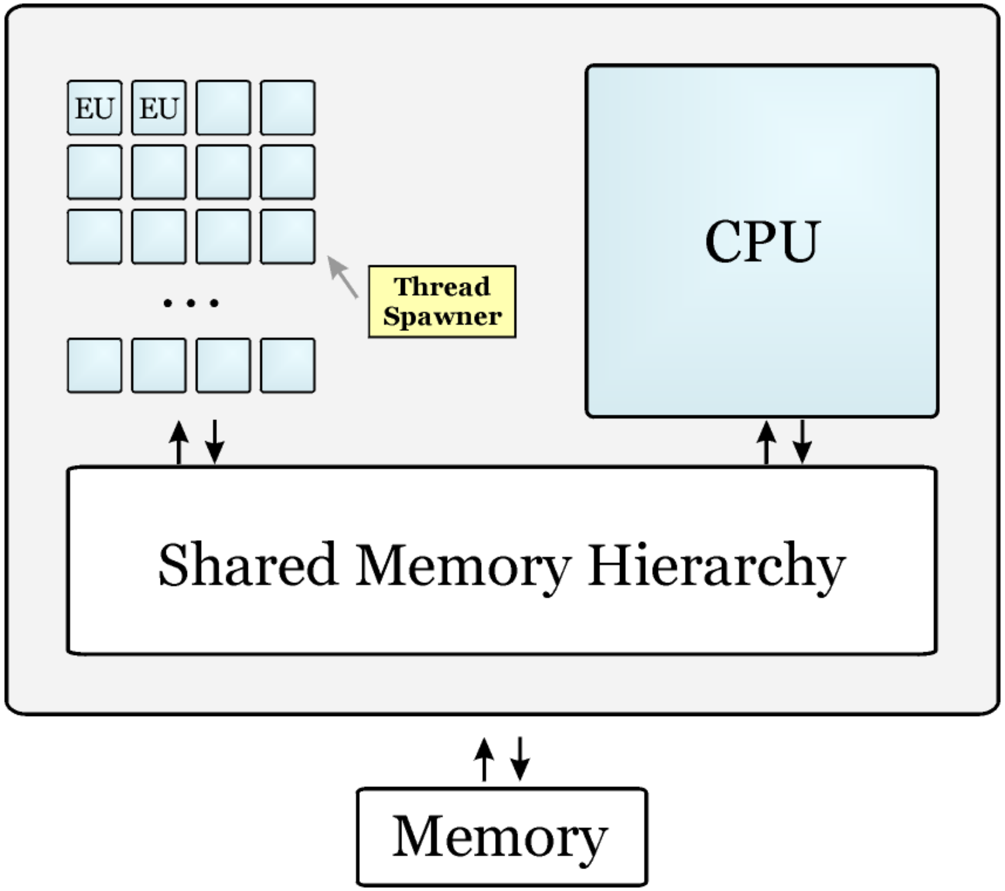
GPU Integration

- Put them on the same chip
  - ~~Off-chip latency~~
  - Explicit data copy between memory spaces
  - Cooperation??



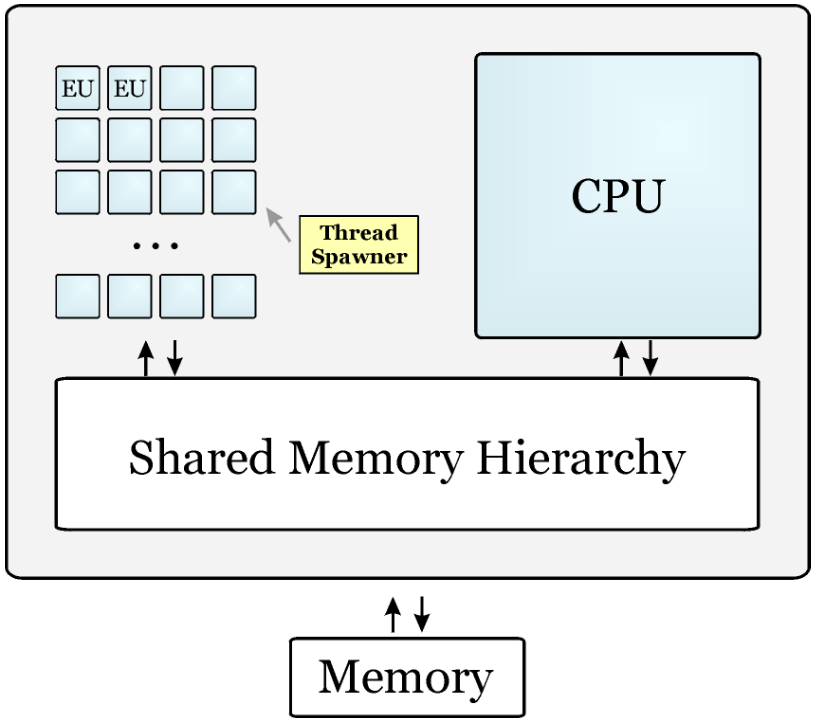
Pangaea

- Single-chip, tightly-coupled
  - ~~Off-chip latency~~
  - Shared memory address space: **Share**, not copy
  - Cooperation!



Pangaea Architecture

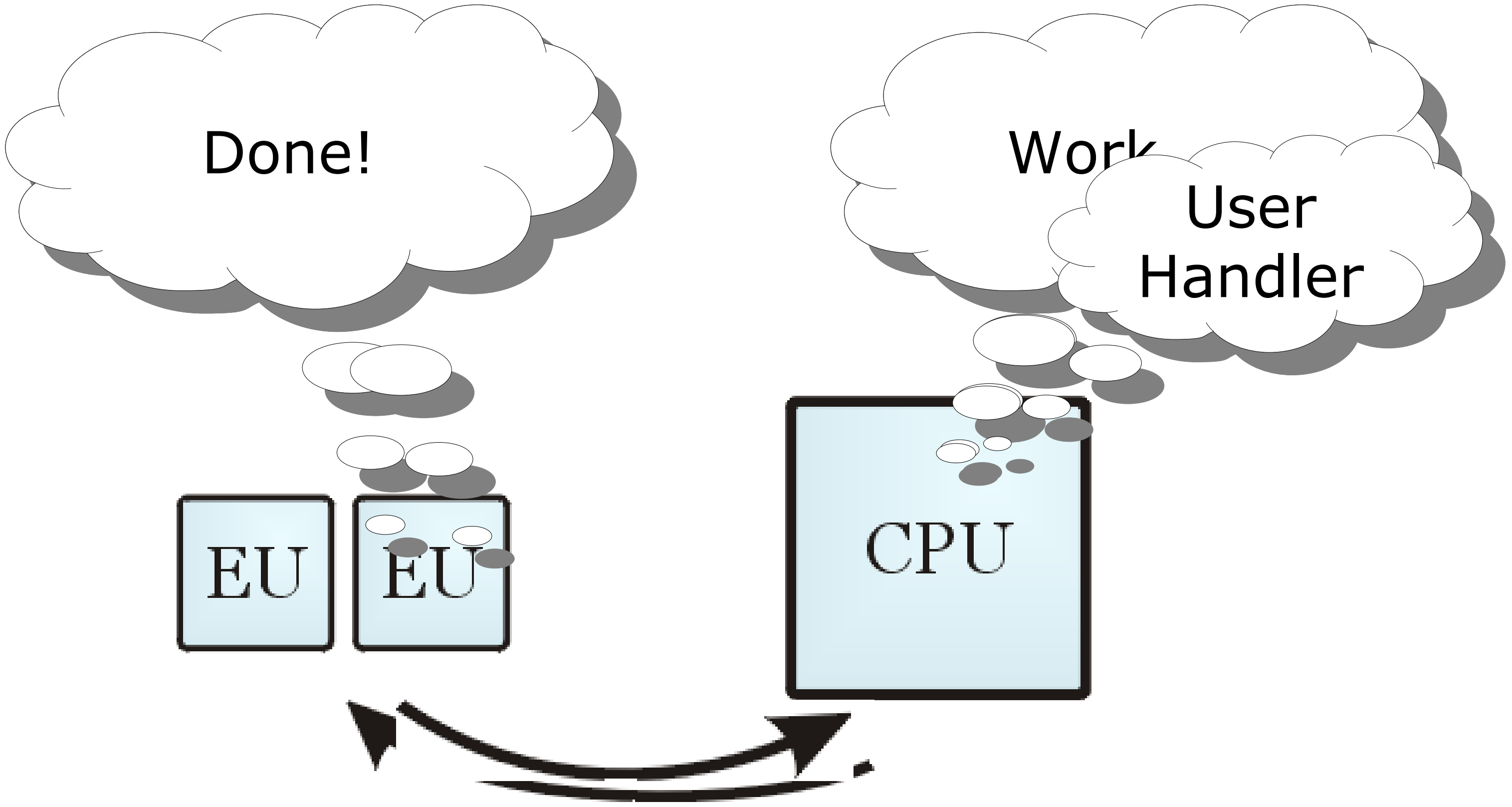
- Tightly-integrated
  - User-level interrupts (ULI) for communication
  - Shared memory and cache
- Use GPU cores for compute
  - "Execution Unit" (EU)



Overview

- Background on GPU Computation
- Pangaea: IA32-GPU chip multiprocessor
  - **User-Level Interrupt mechanism**
  - Architecture trade-offs
  - Prototype performance
- Conclusion

EU Thread Life Cycle



User-Level Interrupts (ULI)

- EMONITOR
  - Watches for an address invalidation
  - Calls user interrupt handler in response
- ERETURN
  - Returns from user-level interrupt handler
- SIGNAL
  - Tells Thread Spawner to start new thread.

Using ULI – CPU Code

```
{
  task_complete = false;
  EMONITOR(&task_complete, &handler);
  SIGNAL(&eu_routine, &eu_data);

  { Do some work }
}
```

Using ULI – EU Code

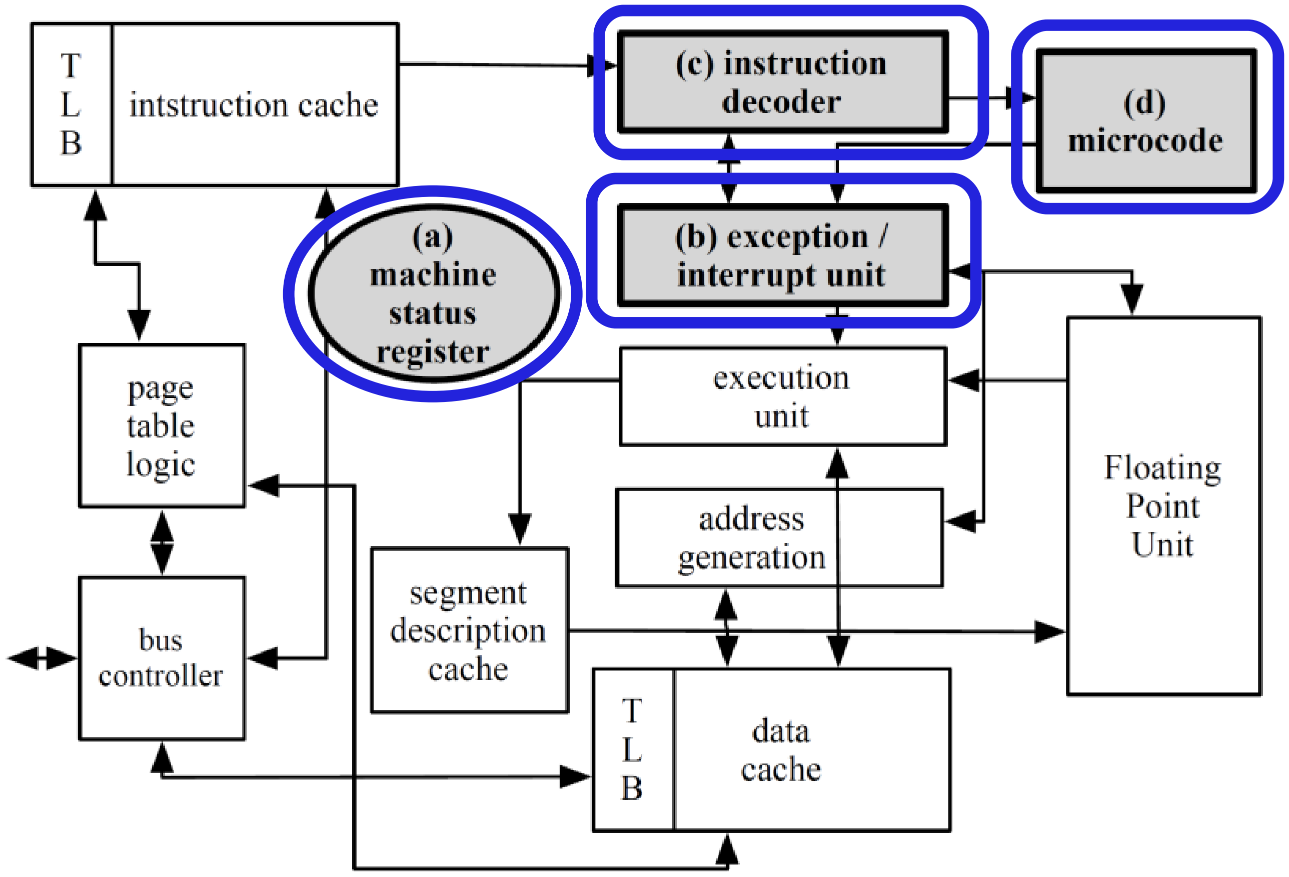
```
{
  Do some work;

  task_complete = true;
}
```

Using ULI – User Handler

```
handler() {
  if (task_complete) {
    Use EU result or start EU task
  }
  ERETURN();
}
```

ULI Pipeline Modifications



[illegible]

[illegible] instructions

Overview

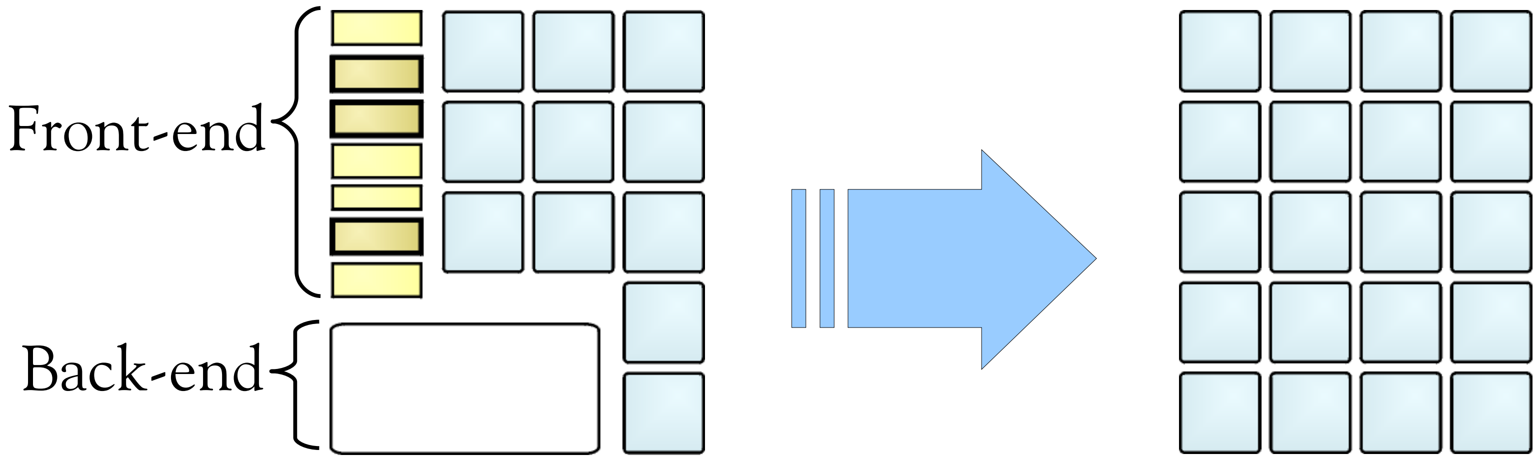
- Background on GPU Computation
- Pangaea: IA32-GPU chip multiprocessor
  - User-Level Interrupt mechanism
  - **Architecture trade-offs**
  - Prototype performance
- Conclusion

Shared Memory Hierarchy

- Shared address space
  - *Address Translation Remapping*: CPU handles memory translation when EU TLB misses
  - See *Perry Wang, et al.,* EXOCHI: Architecture and Programming Environment for a Heterogeneous Multi-core Multithreaded System
- Shared memory hierarchy
  - Share a cache with the CPU
  - Helps collaborative multithreading
  - Avoids copying data between CPU and GPU

Area/Power Efficiency

- Graphics pipeline area is 9.5 cores
  - 65 nm synthesis of Intel GMA X4500
- Power is 4.9 cores
- Replace graphics pipeline with Thread Spawner
  - Thread Spawner is tiny: 1% of core



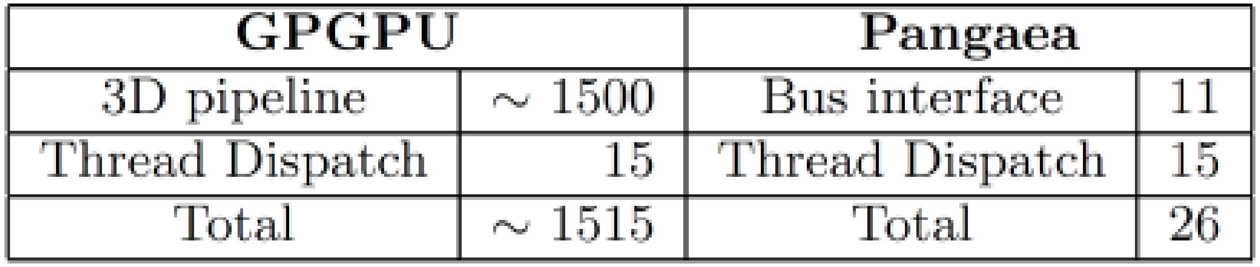
Overview

- Background on GPU Computation
- Pangaea: IA32-GPU chip multiprocessor
  - User-Level Interrupt mechanism
  - Architecture trade-offs
  - **Prototype performance**
- Conclusion

Pangaea Prototype

- Synthesis of production-quality RTL code
  - 2-issue, in-order IA32 CPU (37% of design)
  - 2 EUs from Intel GMA X4500 (31% x 2)
- Virtex 5 LX330, 136772 LUTs, 17 MHz
  - 66% of LX330
- Boots Linux, Windows, DOS, …

Thread Spawn Latency

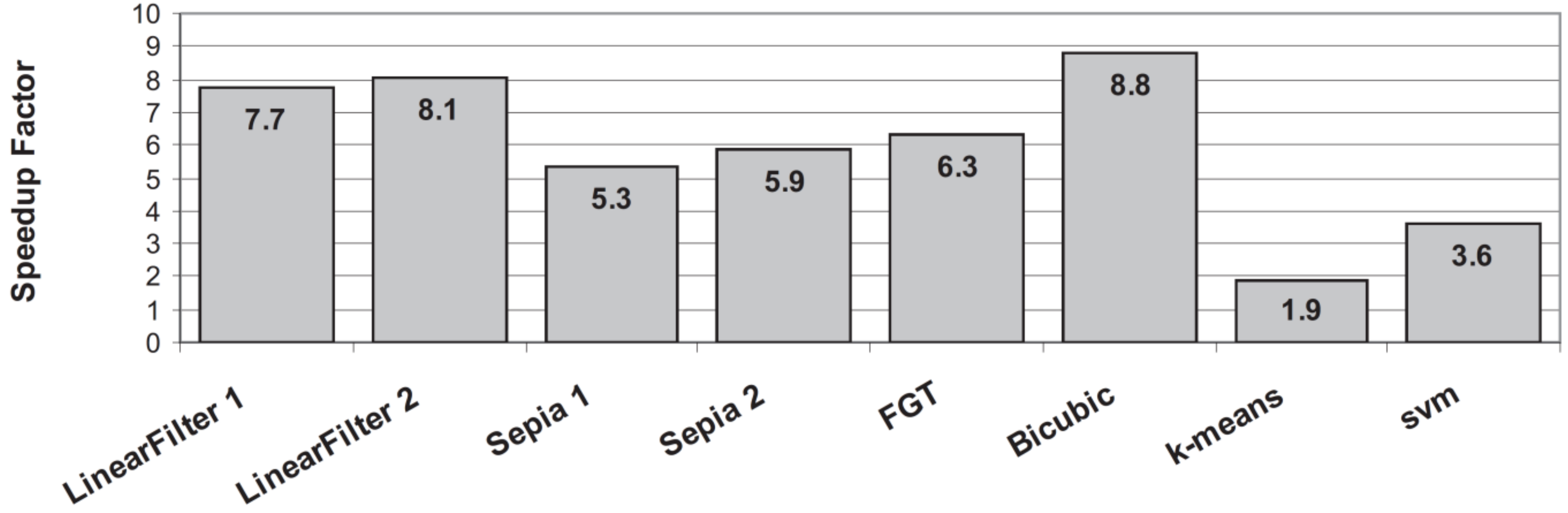


GPGPU		Pangaea	
3D pipeline	~ 1500	Bus interface	11
Thread Dispatch	15	Thread Dispatch	15
Total	~ 1515	Total	26

Thread Spawn latency reduced by 60x when bypassing graphics pipeline

– GPGPU driver software overhead not included

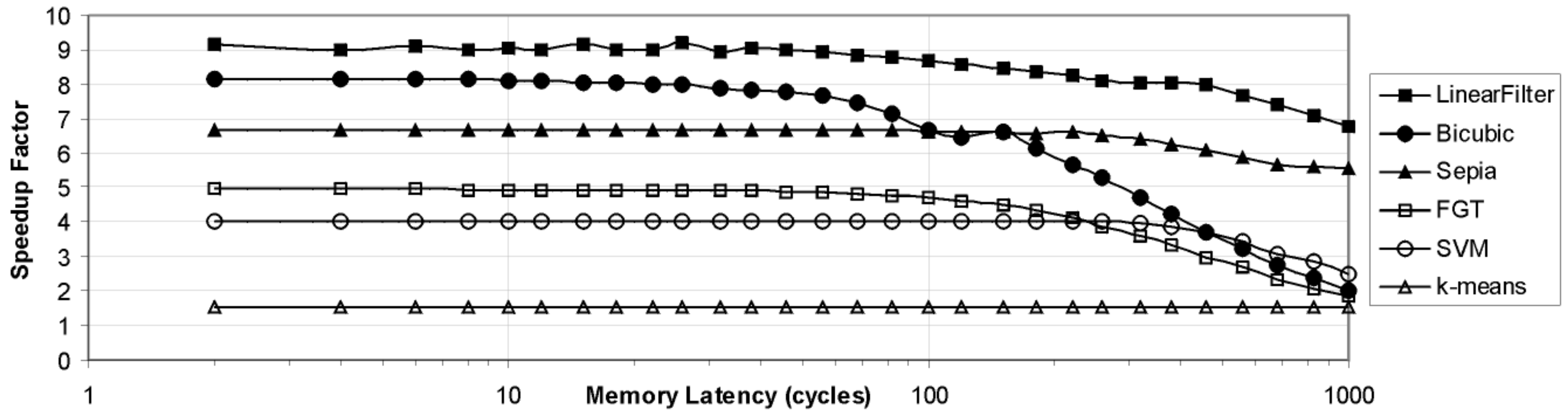
Throughput Performance



2 EUs vs. 1 CPU

- k-means and svm collaborate with CPU
- k-means is CPU-bound

Latency Sensitivity



- Bicubic and FGT code larger than 4KB i-cache
- k-means is CPU-bound
- Insensitive to memory latency < ~60 cycles
  - Can trade off level of memory hierarchy to share

Conclusions

- Added ULI communication to IA32, built on cache coherency mechanisms
  - Modularity allows scalable design
- Shared memory and cache is good for ease of programming and collaboration
  - Highest-performance implementation not critical
- Legacy graphics takes up 9.5 EUs of area, 4.9 EUs of power. Remove if not necessary.
  - Prototype shows it is ok to remove

Conclusions

- IA32 ULI built on cache coherency mechanisms enables scalable, modular design
- Shared memory and cache is good for ease of programming and collaboration
- Legacy graphics fixed functions have high overhead

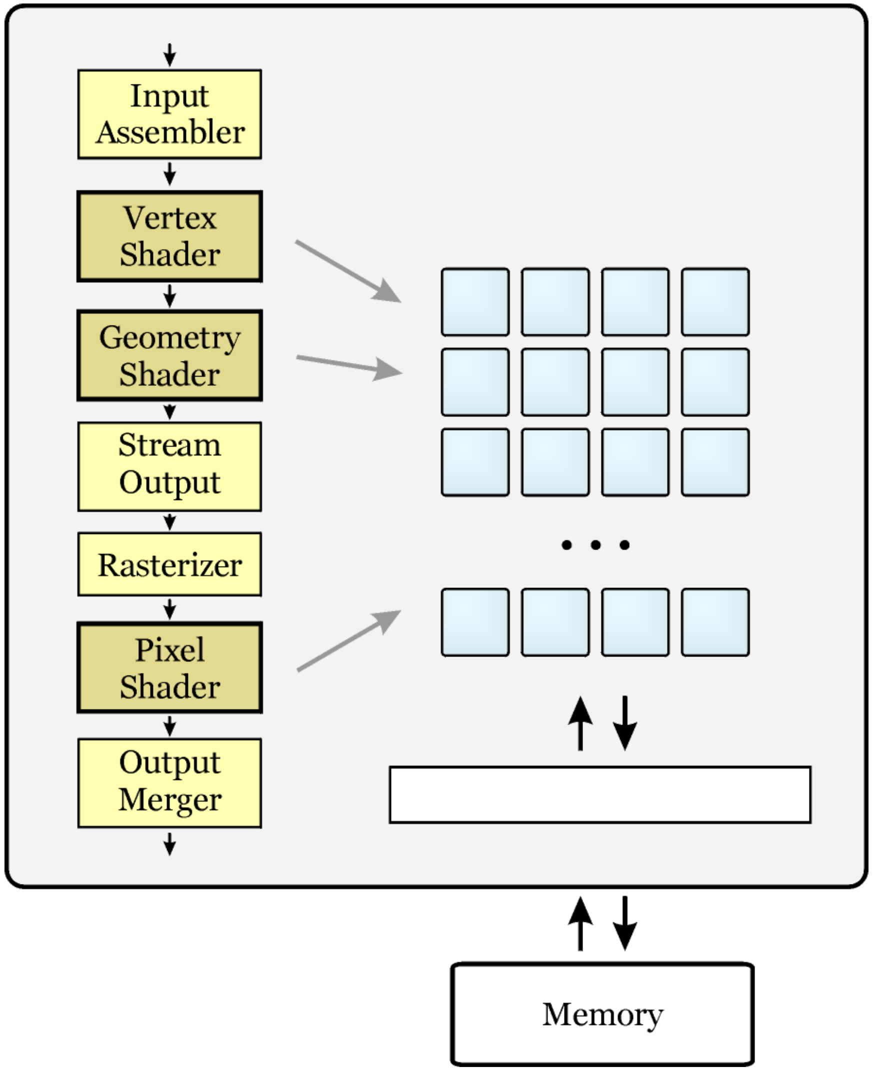
Questions?

EU vs. CPU Peak Throughput

- 2 EUs have 2x peak performance vs. CPU
- TLP increases utilization (92% vs. 65%, linear)
- Large register file (57% vs. 7.4% memory, bicubic)
- Multiply-accumulate (55% of bicubic)
- SIMD-8/16 instructions lowers instruction count

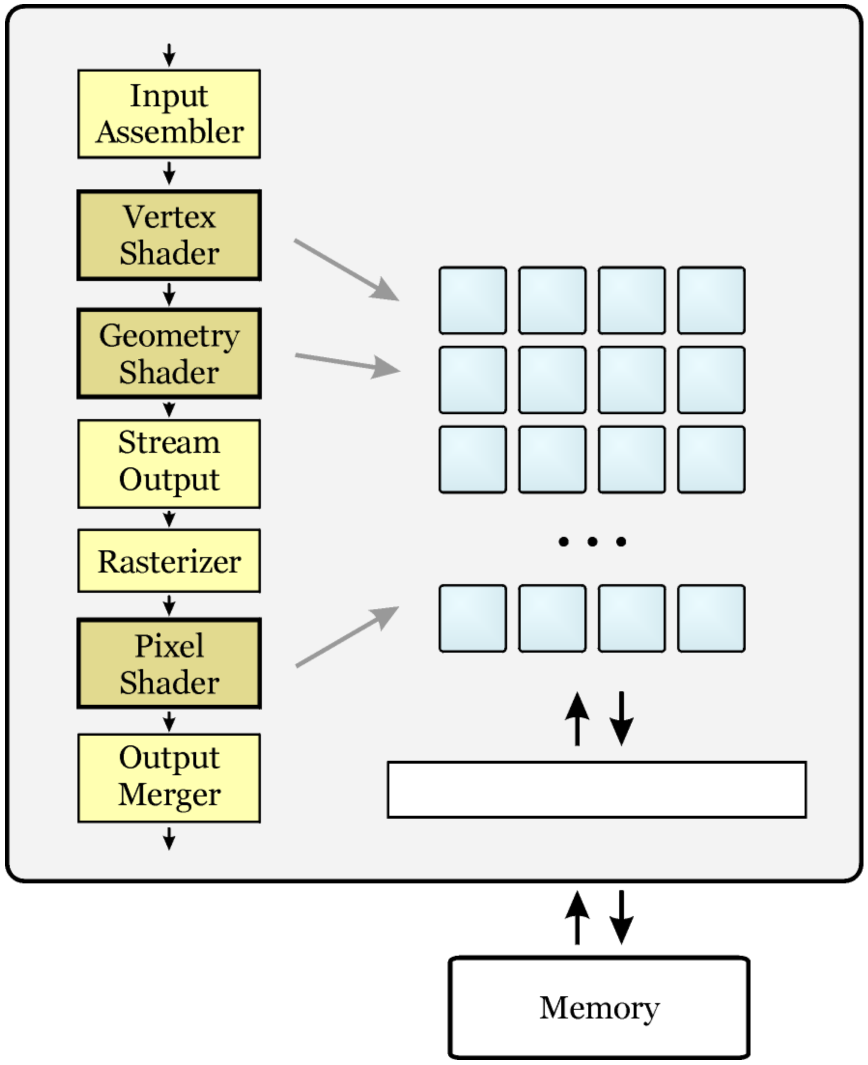
Shaders

- For each vertex, run a program.
  - ...or each pixel
- Program instances mutually independent
- Shaders designed to run many independent instances of the same short program



GPGPU

- For each ______, run a program.
- Write shader programs to do something non-graphics
- Sparse matrix solvers, linear algebra, sorting algorithms...
- Brook for GPUs

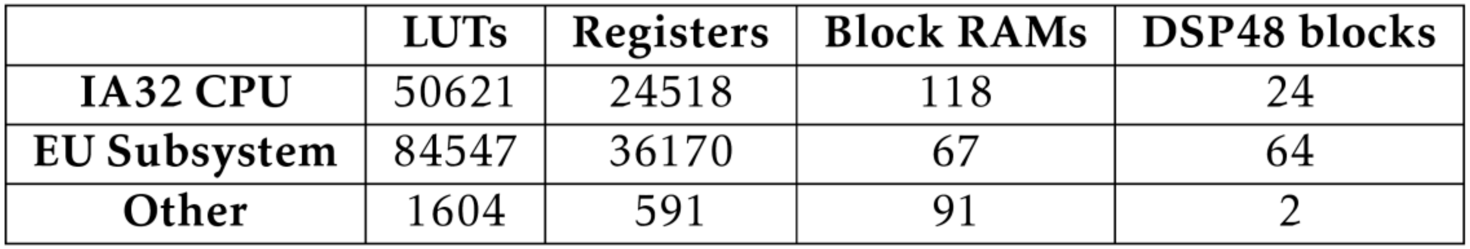


Other Stuff from Intel MRL

- Papers
  - Multiple Instruction Stream Processor
  - EXOCHI: Architecture and Programming Environment for a Heterogeneous Multi-core Multithreaded System
- Ideas
  - User-level “sequencer” so OS isn't modified
  - Let CPU handle exceptions on behalf of “sequencer”
  - Shared memory space for ease of programming
- Pangaea can be thought of as extension of Exo

Pangaea Resource Usage

- 2-issue, in-order IA32 CPU
- 2 EUs from Intel GMA X4500
- Virtex 5 LX330, 136772 LUTs, 17 MHz

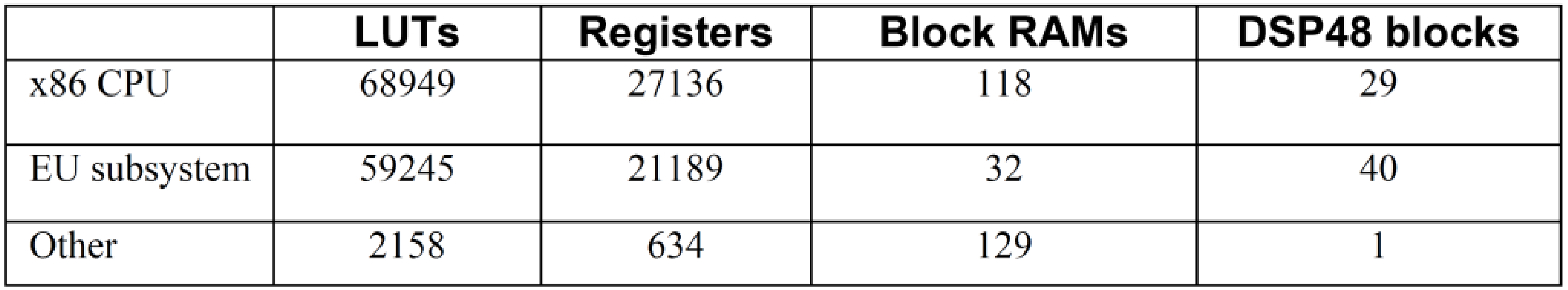


	LUTs	Registers	Block RAMs	DSP48 blocks
IA32 CPU	50621	24518	118	24
EU Subsystem	84547	36170	67	64
Other	1604	591	91	2

Table 2.2: Virtex-5 FPGA Resource Usage for the Pangaea configuration in Table 2.1.

Earlier Pangaea Prototype

- 1 CPU, 1 EU, 256 kB memory
- Virtex 4 LX200, 130352 4-LUTs, 17.5 MHz



	LUTs	Registers	Block RAMs	DSP48 blocks
x86 CPU	68949	27136	118	29
EU subsystem	59245	21189	32	40
Other	2158	634	129	1