A whistle-stop tour of some parallel architectures that are used in practice

David Gregg
Department of Computer Science
University of Dublin, Trinity College

Parallel Architectures

- Wide range of parallel architectures
  - Many in use
  - Many more have been proposed
- Same ideas emerge again and again
  - But Moore's law means that the components change that we build parallel computers from
  - So the trade-offs between approaches change

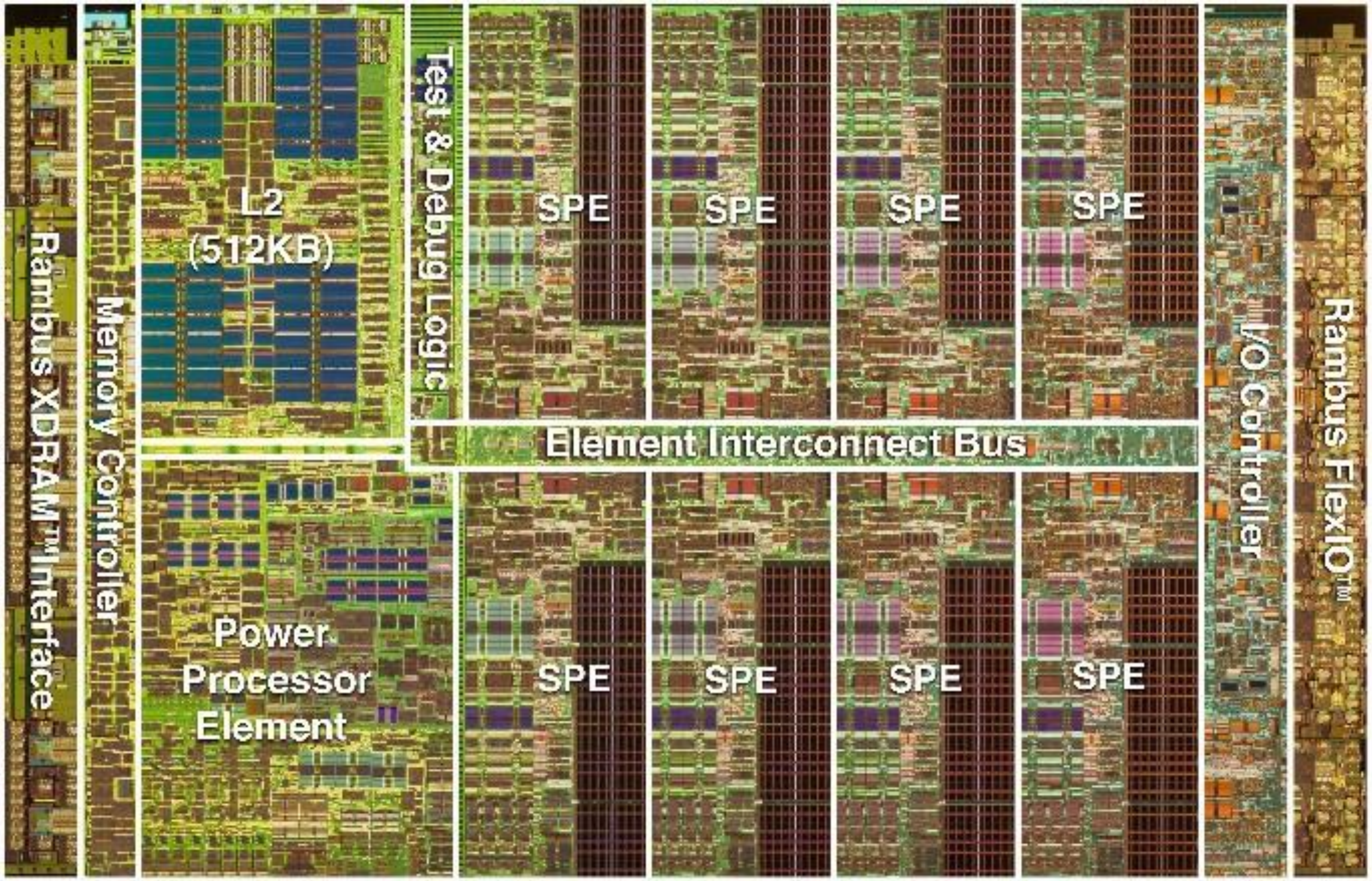
Levels of parallelism

- Different parallel architectures exploit parallelism at different levels
  - Coarse or fine grain
  - Different types of synchronization
- Trend towards architectures that exploit multiple types and levels of parallelism

E.g. 1: STI Cell B/E

- Cell Broadband Engine is a processor developed by Sony, Toshiba and IBM
- Most celebrated use is in Playstation 3
  - Good graphics, physics, etc.
- Also used in IBM Cell clusters for HPC
  - Computing values of options
- Also used in Toshiba consumer electronics

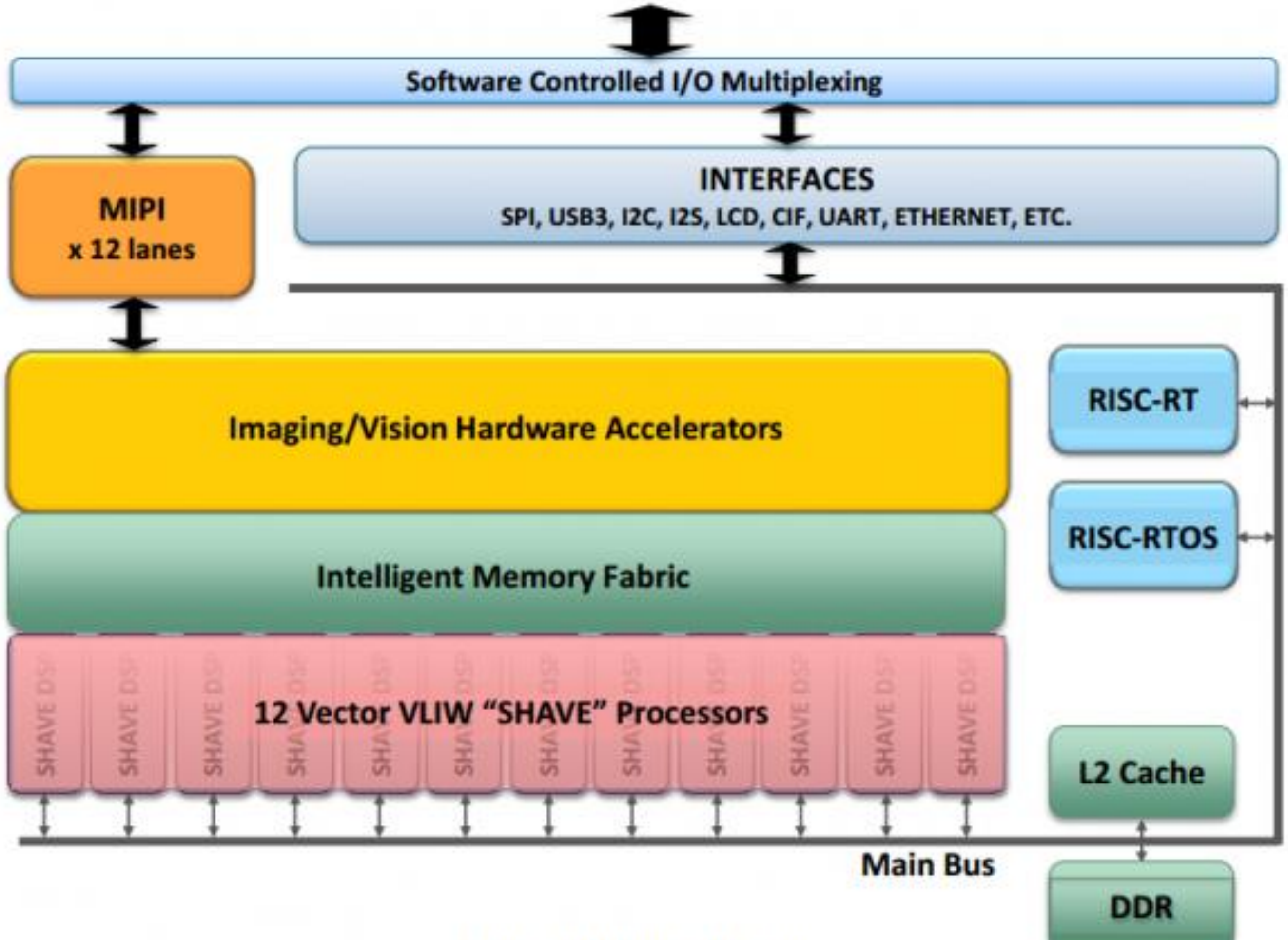
Cell B/E


Rambus XDRAM™ Interface
Memory Controller
L2 (512KB)
Test & Debug Logic
Power Processor Element
SPE
SPE
SPE
SPE
Element Interconnect Bus
SPE
SPE
SPE
SPE
I/O Controller
Rambus FlexIO™

E.g. 2: Myriad X

- Myriad X is an embedded accelerator processor from Movidius-Intel
- Used in drones, headsets, phones
  - Image processing, deep neural networks
- Two simple controller cores
- 12 VLIW vector cores
  - Shared on-chip memory
- Dedicated hardware accelerators
  - Image processing, AI

Movidius Myriad X


Software Controlled I/O Multiplexing
MIPI
x 12 lanes
INTERFACES
SPI, USB3, I2C, I2S, LCD, CIF, UART, ETHERNET, ETC.
Imaging/Vision Hardware Accelerators
RISC-RT
RISC-RTOS
Intelligent Memory Fabric
12 Vector VLIW "SHAVE" Processors
SHAVE DSP
L2 Cache
Main Bus
DDR

Myriad 2 Block Diagram

Flynn's Taxonomy

- Prof. Mike Flynn's famous taxonomy of parallel computers
  - Single instruction stream, single data stream
    - E.g. VLIW, found in Cell BE
  - Single instruction stream, multiple data streams
    - E.g. Vector SIMD, found in Cell BE
  - Multiple instruction streams, multiple data streams
    - E.g. Multiple parallel cores, found in Cell BE
  - Multiple instruction streams, single data stream
    - Unusual, sometimes found in safety-critical systems
    - Texas Instruments Hercules processors

Flynn’s Taxonomy

- It is quite difficult to fit many parallel architectures into Flynn’s taxonomy
  - It attempted to describe different types of parallel architectures in 1966
  - It’s not entirely clear where some architectures fit
    - E.g. instruction level parallel
    - E.g. fine-grain speculative multithreading
- Most important distinction is between SIMD and MIMD
  - These terms are used a lot

Instruction level parallelism

- Attempt to execute multiple machine instructions from the same instruction stream in parallel
- Two major approaches
  - Deep pipelining (super-pipelining)
    - Production line of instructions
    - Overhead at each pipeline stage so very deep pipelining has lots of overhead
  - Multiple instruction pipelines
    - "Superscalar"
    - Complex hardware or compiler

Instruction level parallelism

- Finding parallel instructions, either…
  - Get the compiler to find the parallelism
    - VLIW, in-order superscalar
  - Get the hardware to find independent instructions in the running program
    - Out-of-order superscalar
- Cell B/E
  - Main processor core is superscalar
  - Accelerator cores are VLIW
- Myriad X
  - Accelerator cores are VLIW

Instruction level parallelism

- Cell B/E
  - Main processor core is superscalar
  - Accelerator cores are VLIW
- Myriad X

Vector Computers

- Most machine instructions operate on a single value at a time
- Vector computers operate on an array of data with a single instruction
- Many parallel computing problems operate on large arrays of data
- Vector computers were some of the earliest and most successful parallel computers
  - E.g. many supercomputers designed by Seymour Cray

Vector Computers

- Cray 1
    - glass panels allow viewer to “see more” Cray

Vector Computers

- Vector instructions are a form of SIMD
  - very popular on modern computers
  - Accelerate multimedia, games, scientific, graphics apps
- Vector sizes are smallish
  - E.g. 16 bytes, four floats – SSE, Neon
  - Intel AVX uses 32 bytes
  - Intel AVX-512 uses 64 bytes
- Vector SIMD used in Cell BE, where all SPE instructions are SIMD

Multithreaded Computers

- Pipelined computers spend a lot of their time stalled
  - Cache misses, branch mispredictions, long-latency instructions
  - Multi-threaded computers attempt to cover the cost of stalls by running multiple threads on the same processor core
  - Require very fast switching between threads
    - Separate register set for each thread

Multithreaded Computers

- Three major approaches
  - Switch to a different thread every cycle
  - Switch thread on a major stall
  - Use OOO superscalar hardware to mix instructions from different threads together (aka simultaneous multithreading)
- Big problem is software
  - Programmer must divide code into threads
- Multithreading is used in Cell main core

Multi-core computers

- Power wall, ILP wall and memory wall pushed architecture designers to put multiple cores on a single chip
- Two major approaches
  - Shared memory multi-core
  - Distributed memory multi-core
- Shared memory is usually easier to program, but
  - hardware does not scale well
  - shared memory needs cache consistency
- Big problem is dividing software into threads
- Cell BE has distributed memory SPEs, but SPEs can also access global memory shared by PPE

Shared memory multiprocessors

- Shared memory multiprocessors are parallel computers where there are multiple processors sharing a single memory
- Main approaches
  - Symmetric multiprocessors
  - Non-uniform memory access (NUMA)
    - Usually a distributed shared memory
- Big problem is software
- IBM Cell Blades allow two Cell processors on the same board to share memory

Distributed memory multicomputers

- A system of communicating processors, each with their own memory
- Distributed memory systems scale extremely well
  - The hardware is simple
- Huge variety of configurations
  - ring, star, tree, hypercube, mesh
- Programming is arguably even more difficult than for shared memory machines
  - Always need to move data explicitly to where it is needed
- The Cell processor provides 8 SPEs, which each have their own local memory, and are connected in a ring topology

Clusters

- A group of loosely coupled computers that work closely and can be viewed as a single machine
- Multiple stand-alone machines connected by a network
- Clusters are a form of distributed memory multicomputers
  - But may support a virtual shared memory using software
  - E.g PVM (parallel virtual machine)
- Classic example is the Beowulf Cluster, made from standard PCs, running Linux or Free BSD, connected by ethernet

Clusters

- Most big supercomputers these days are clusters
  - It’s usually cheaper to build large machines out of thousands of cheap PCs
- Cell BE is used in IBM Cell clusters
- Clusters of Playstation 3 machines running Linux are also being used as cheap supercomputers

Graphics Processing Units (GPU)

- Emergence of GPUs is a major recent development in computer architecture
- GPUs have very large amounts of parallelism on a single chip
  - vector processing units
  - multithreaded processing units
  - multiple cores
- GPUs originally aimed at graphics rendering
  - Very large memory bandwidth
    - But high memory latency
  - Lots of low-precision floating point units
- GPUs implement graphics pipeline directly
  - Generality was once very limited, but is getting better with each new generation of GPU

Graphics Processing Units (GPU)

- GPUs are increasingly programmable
- Trend towards using GPUs for more "general purpose" computation (GPGPU)
  - But mapping general purpose applications to GPU architecture is challenging
  - Only certain types of apps run well on GPU
- Special languages are popular for GPGPU
  - E.g. CUDA, OpenCL

Field Programmable Gate Arrays

- Gate array
  - There is an array of gates that can be used to implement hardware circuits
- Field programmable
  - The array can be configured (programmed) at any time "in the field".
- FPGAs can exploit parallelism at many levels
  - Special purpose processing units
    - E.g. bit twiddling operations
  - Fine grain parallelism
    - Circuit is clocked, so very fine grain synchronization is possible
  - Coarse grain parallelism
    - FPGA can implement any parallel architecture that fits on the chip

Field Programmable Gate Arrays

- FPGAs are commonly used to implement special-purpose parallel architectures
  - E.g. Systolic arrays

Low Power Parallel Processors

- Several big changes in computing
  - (Exact categories and times very debatable)
  - Giant computer room with single user – 1950s
  - Mainframe computers with hundreds of users – 1960s
  - Minicomputer with dozens of users – 1970s
  - Microcomputer with single user – 1980s
  - Networks and microcomputer servers – 1990s
  - Continuous network connection – 2000s
  - Mobile internet devices – 2010s

Low Power Parallel Processors

- Power and energy are extremely important for mobile computing
  - Trend towards mobile devices, mobile phones, wearable computing, internet of things
  - Battery life depends directly on energy consumption
  - Devices that consume very little energy can really add up if you have dozens or hundreds of them
- Parallelism is alternative to faster clock
  - Dynamic power is Capacitance * Frequency * Voltage^2
  - Significant increase in clock normally requires increase in voltage

Low Power Parallel Processors

- Low power multicore processors
  - Often trade software complexity for hardware complexity
    - Many are VLIW processors
    - Instruction-level parallelism under software control
    - Complicated compilers
  - On-chip scratchpad memory instead of caches
    - Caches need logic to find cache line
    - Cache line tags, comparators, replacement logic, etc.
    - Scratch-pad memory is fully under software control
    - Data transfers using so-called directed memory access (DMA)
    - More complicated compilers, software
  - Cores do not share on-chip memory
    - Programmer must manage each core’s scratchpad memory
    - Explicit movement of data between cores