

Accelerators

Accelerators

◆ Accelerated systems.

◆ System design:

- performance analysis;
- scheduling and allocation.

Accelerated systems

- ◆ Use additional computational unit dedicated to some functions?
 - Hardwired logic.
 - Extra CPU.
- ◆ **Hardware/software co-design**: joint design of hardware and software architectures.

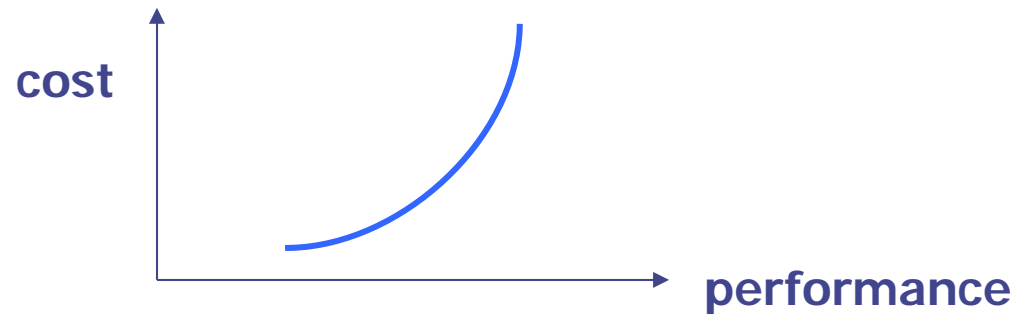
Accelerator vs. co-processor

- ◆ A co-processor executes instructions.
 - Instructions are dispatched by the CPU.
- ◆ An accelerator appears as a device on the bus.
 - The accelerator is controlled by registers.

Why accelerators?

◆ Better cost/performance.

- Custom logic may be able to perform operation faster than a CPU of equivalent cost.
- CPU cost is a non-linear function of performance.



Why accelerators? cont'd.

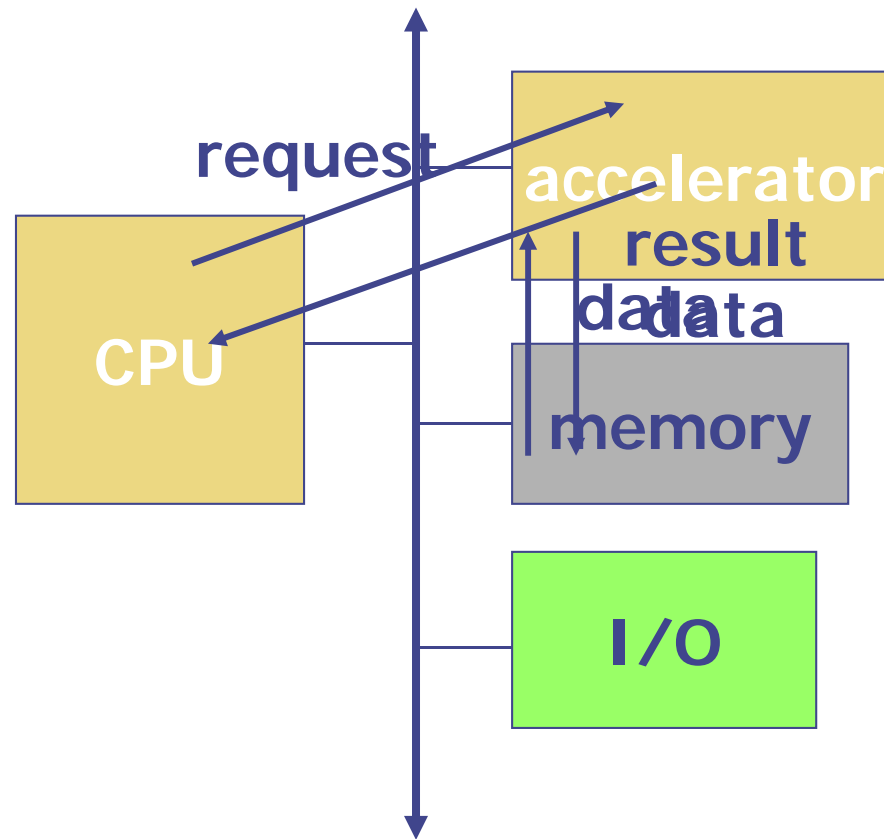
- ◆ Better real-time performance.
 - Put time-critical functions on less-loaded processing elements.
- ◆ Better Energy-Delay tradeoffs

Why accelerators? cont'd.

◆ Good for :

- I/O processing in real-time.
- Data streaming (audio, video, network traffic, real-time monitoring, etc.)
- Specific "complex" operations:
 - ◆ FFT, DCT, EXP, LOG, ...
- Specific "complex" algorithms:
 - ◆ Neuronal networks, ...

Accelerated system architecture



Accelerator implementations

- ◆ Application-specific Integrated Circuit (ASIC).
- ◆ Field-programmable gate array (FPGA).
- ◆ Standard component.
 - Example: graphics processor.

System design tasks

- ◆ Design a heterogeneous multiprocessor architecture.
 - Processing element (PE): CPU, accelerator, etc.
- ◆ Program the system.

Accelerated system design

- ◆ First, determine that the system really needs to be accelerated.
 - How much faster is the accelerator on the core function?
 - How much data transfer overhead?
- ◆ Design the accelerator itself.
- ◆ Design CPU interface to accelerator.

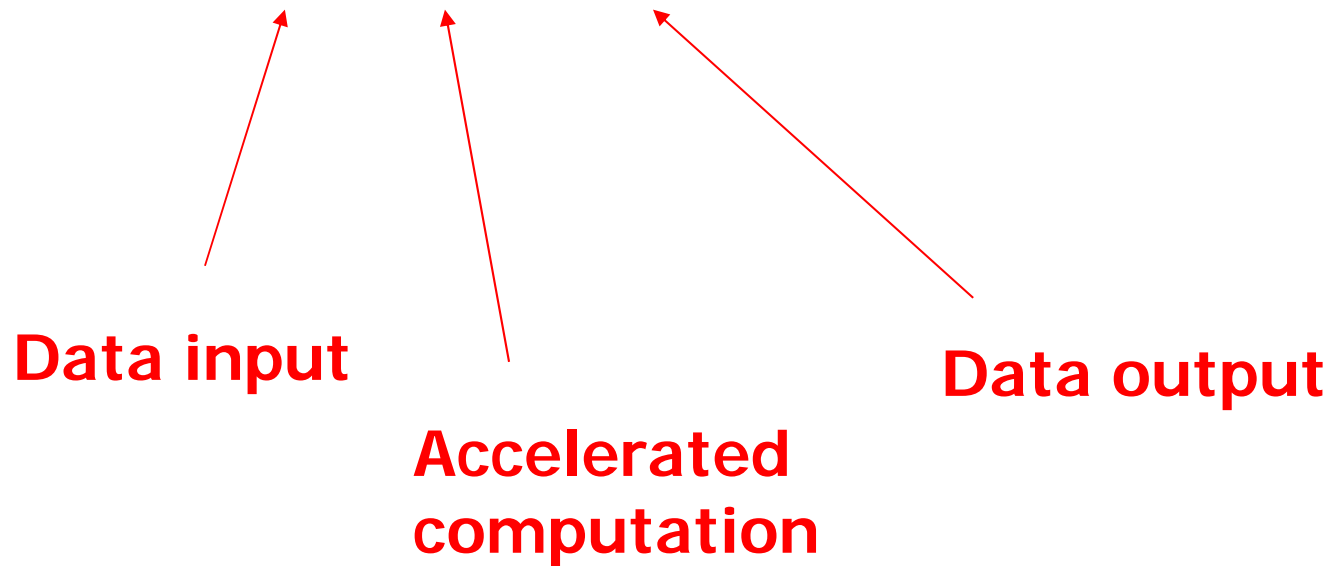
Performance analysis

- ◆ Critical parameter is **speedup**: how much faster is the system with the accelerator?
- ◆ Must take into account:
 - Accelerator execution time.
 - Data transfer time.
 - Synchronization with the master CPU.

Accelerator execution time

◆ Total accelerator execution time:

- $t_{\text{accel}} = t_{\text{in}} + t_x + t_{\text{out}}$



Data input/output times

◆ Bus transactions include:

- flushing register/cache values to main memory;
- time required for CPU to set up transaction;
- overhead of data transfers by bus packets, handshaking, etc.

Accelerator speedup

- ◆ Assume loop is executed n times.
- ◆ Compare accelerated system to non-accelerated system:
 - $S = n(t_{\text{CPU}} - t_{\text{accel}})$
 - $= n[t_{\text{CPU}} - (t_{\text{in}} + t_x + t_{\text{out}})]$

**Execution time on CPU**

Single- vs. multi-threaded

- ◆ One critical factor is available parallelism:
 - **single-threaded/blocking**: CPU waits for accelerator;
 - **multithreaded/non-blocking**: CPU continues to execute along with accelerator.
- ◆ To multithread, CPU must have useful work to do.
 - Software must also support multithreading.

Sources of parallelism

◆ Overlap I/O and accelerator computation.

- Perform operations in batches, read in second batch of data while computing on first batch.

◆ Find other work to do on the CPU.

- May reschedule operations to move work after accelerator initiation.

Accelerator/CPU interface

- ◆ Accelerator registers provide control registers for CPU.
- ◆ Data registers can be used for small data objects.
- ◆ Accelerator may include special-purpose read/write logic.
 - Especially valuable for large data transfers.

Accelerator “usual” problems

- ◆ Memory consistency and coherency (specially if the CPU has caches)
- ◆ Partitioning the source code into accelerated chunks.
- ◆ Scheduling of the code chunks.
- ◆ Allocation to accelerators (if many)

Accelerated systems

- ◆ Several off-the-shelf boards are available for acceleration in PCs:
 - FPGA-based core;
 - PCIe bus interface.

Natural Markets

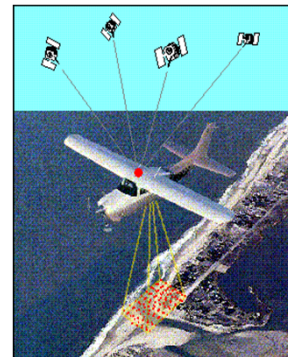
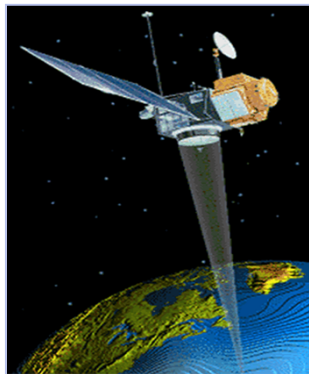
◆ Embedded Systems

- FPGAs appearing in set-top boxes, routers, audio equipment, etc.
- Advantages
 - ◆ Performance close to ASIC, sometimes at much lower cost
 - Many other embedded systems still use ASIC due to high volume
 - Cell phones, iPod, game consoles, etc.
 - ◆ Reconfigurable!
 - If standards change, architecture is not fixed
 - Can add new features after production



Natural Markets

- ◆ High-performance embedded computing (HPEC)
 - High-performance/super computing with special needs (low power, low size/weight, etc.)
 - ◆ Satellite image processing
 - ◆ Target recognition in a UAV
 - Advantages
 - ◆ Much smaller/lower power than a supercomputer
 - ◆ Fault tolerance



Natural Markets

◆ High-performance computing (HPC)

- Cray, SGI, DRC, GiDEL, Nallatech, XtremeData
 - ◆ Combine high-performance microprocessors with FPGA accelerators
- Novo-G
 - ◆ 192 Altera Stratix III FPGAs integrated with 24 quad-core microprocessors



◆ Advantages

- HPC used for many scientific apps
 - ◆ Low volume, ASIC rarely feasible, microprocessor too slow
- Lower power consumption
 - ◆ Increasingly important
 - ◆ Cooling and energy costs are dominant factor in total cost of ownership



Natural Markets

◆ General-purpose computing???

- Ideal situation: desktop machine/OS uses a programmable accelerator to speed up all applications (similar to GPU trend)
- Problems
 - ◆ The accelerator can be very fast, but not for all applications
 - Generally requires parallel algorithms
 - ◆ Coding constructs used in many applications not appropriate for hardware
- Subject of tremendous amount of past and likely future research

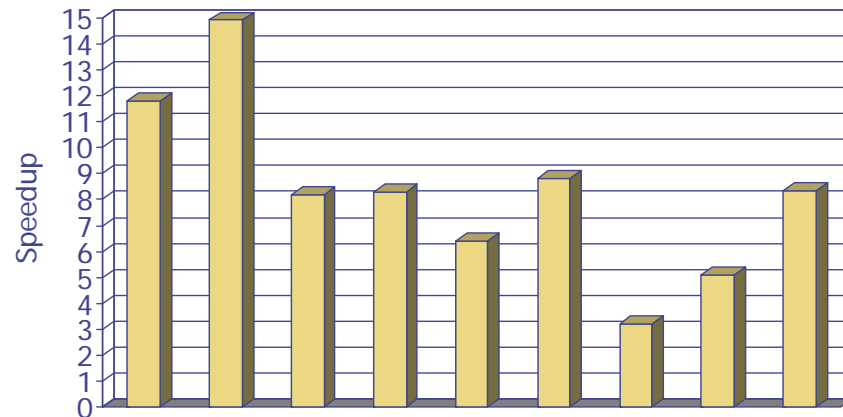
◆ How to use extra transistors on general purpose CPUs?

- More cache
- More microprocessor cores
- GPU
- FPGA?
- Something else?

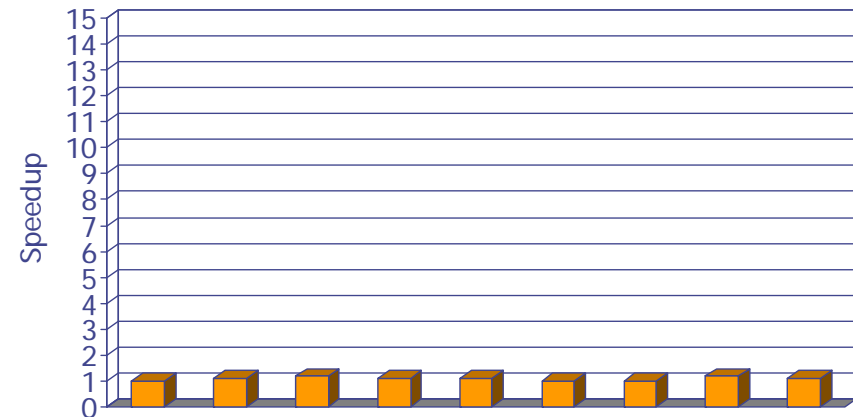
Limitations of FPGA acceleration

◆ 1) Not all applications can be improved

Embedded Applications – Large Speedups



Desktop Applications – *No Speedup*



◆ 2) Tools need serious improvement!

◆ 3) Design strategies are often ad hoc

◆ 4) Floating point?

- Requires a lot of area, but performance is becoming competitive with other devices
 - ◆ Already superior in terms of energy

Natural Markets

◆ General-purpose computing???

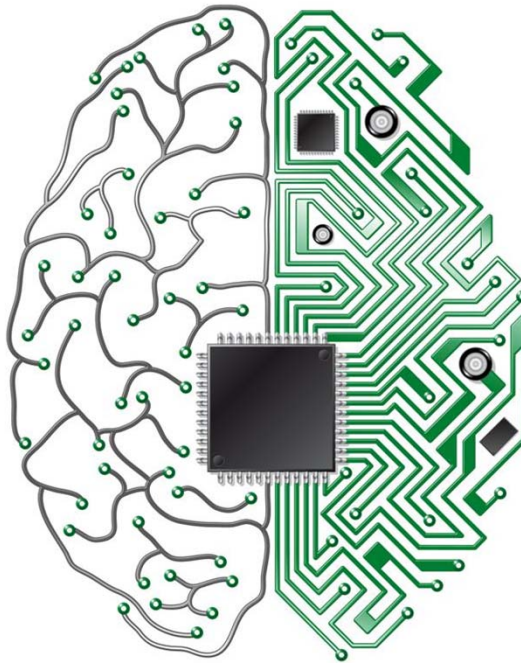
- Ideal situation: desktop machine/OS uses a programmable accelerator to speed up all applications (similar to GPU trend)
- Problems
 - ◆ The accelerator can be very fast, but not for all applications
 - Generally requires parallel algorithms
 - ◆ Coding constructs used in many applications not appropriate for hardware
- Subject of tremendous amount of past and likely future research

◆ How to use extra transistors on general purpose CPUs?

- More cache
- More microprocessor cores
- GPU
- FPGA?
- Something else?

Something else...

- ◆ Brain-inspired accelerators



Neuromorphic Architectures

Historical Highlights

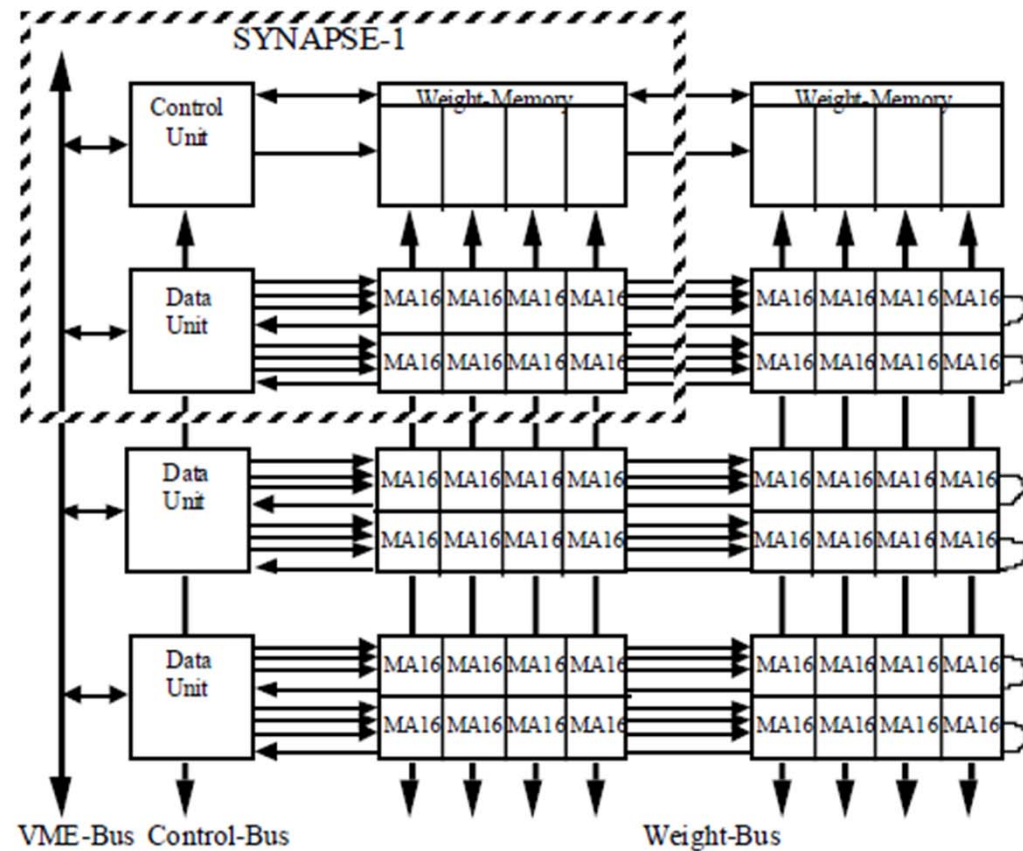
- ◆ Neuromorphic: mimic neuro-biological architectures present in the nervous system
- ◆ Coined by Carver Mead (CalTech) in the 80's
- ◆ Took-off in the 90's.
- ◆ Consolidated in the late 00's
 - 2009 Stanford (Neurogrid)
 - 2011 MIT (Brain chip)
 - 2012 IBM (Neurosynaptic chip)
 - 2013 HP

Artificial Neural Network Chips

- ◆ Early neuromorphic architectures were artificial neural network chips
- ◆ Examples:
 - ETANN : (1989) Entirely analog chip that was designed for feed forward artificial neural network operation.
 - Ni1000 : (1996) Significantly more powerful than ETANN, however has narrower functionality

SYNAPSE-1 System Architecture

SYNAPSE-1 is a modular system arranged as a 2D array of MA16s, weight memories, data units, and a control unit



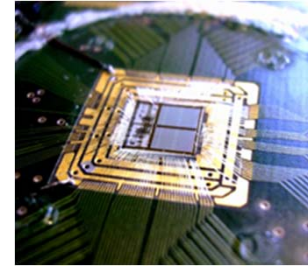
Siemens, 1995

Modern Architectures: Custom Circuits

Neurogrid

- ◆ (2005) Neurogrid is a multi-chip system developed by Kwabena Boahen and his group at Stanford University [9]
 - Objective is to emulate neurons
 - Composed of a 4x4 array of Neurocores
 - Each Neurocore contains a 256x256 array of neuron circuits with up to 6,000 synapse connections

The FACETS Project



- ◆ (2005) Fast Analog Computing with Emergent Transient States (FACETS)
 - A project designed by an international collective of scientists and engineers funded by the European Union
 - Developed a chip containing 200,000 neuron circuits connected by 50 million synapses.
- ◆ Now continues under the Brain Scale S project

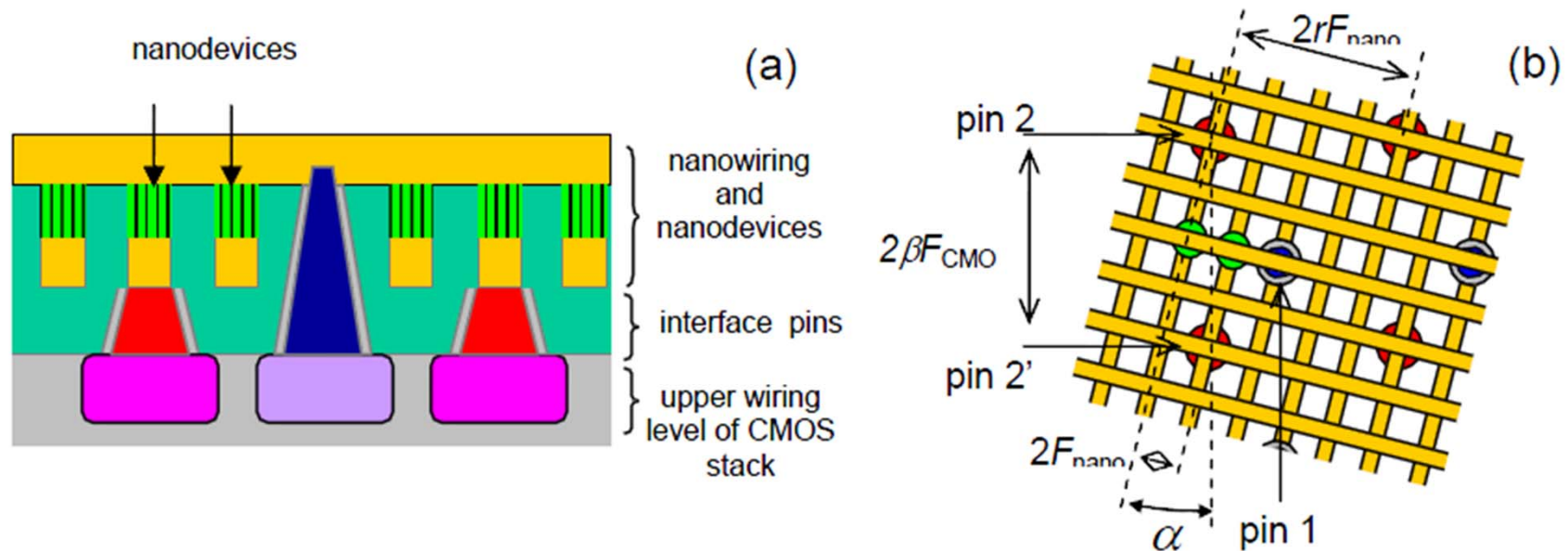
Modern Architectures: Custom Circuits

FPGA Model

- ◆ Torres-Huitzil et. al (INRIA, 2005) designed an hardware architecture for a bio-inspired neural model for motion estimation.
 - Architecture has 3 basic components which perform spatial, temporal, and excitatory-inhibitory connectionist processing.
 - Observed approximately 100 x speedup over Pentium 4 processor implementation for 128x128 images

CMOL based design

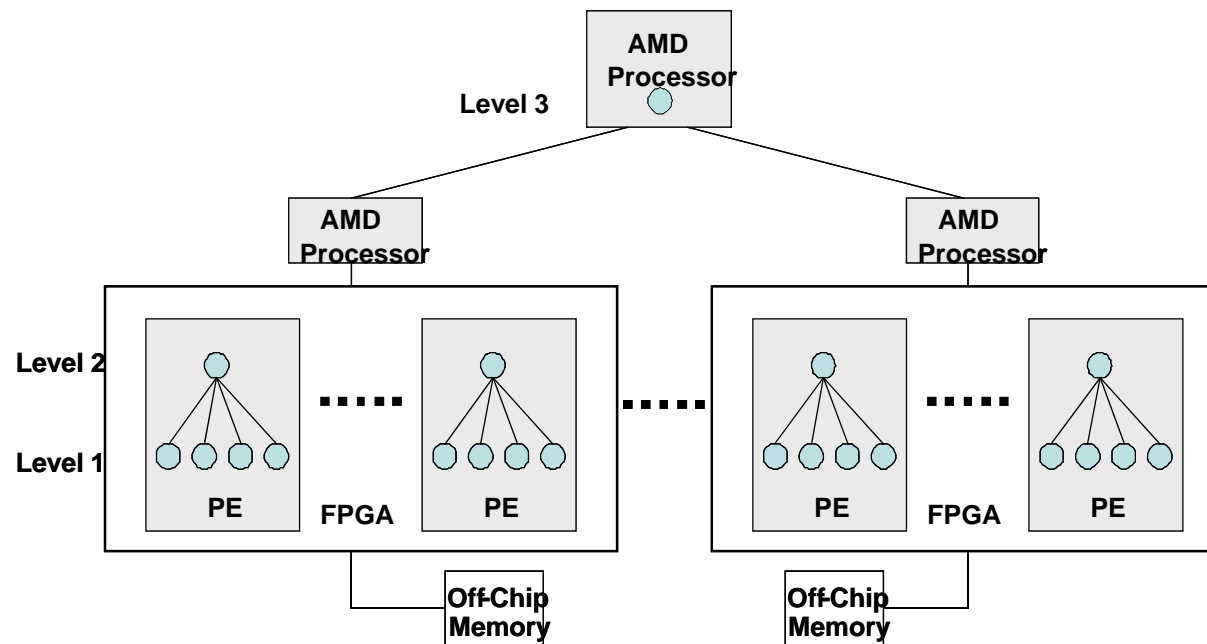
- ◆ CMOL = Cmos + MOlecular (2003)
- ◆ Konstantin Likharev (State University of New York at Stony Brook)
- ◆ Dan Hammerstrom (Portland State University / DARPA)



- ◆ Similar approach as HP with Memristors

HTM on FPGAs

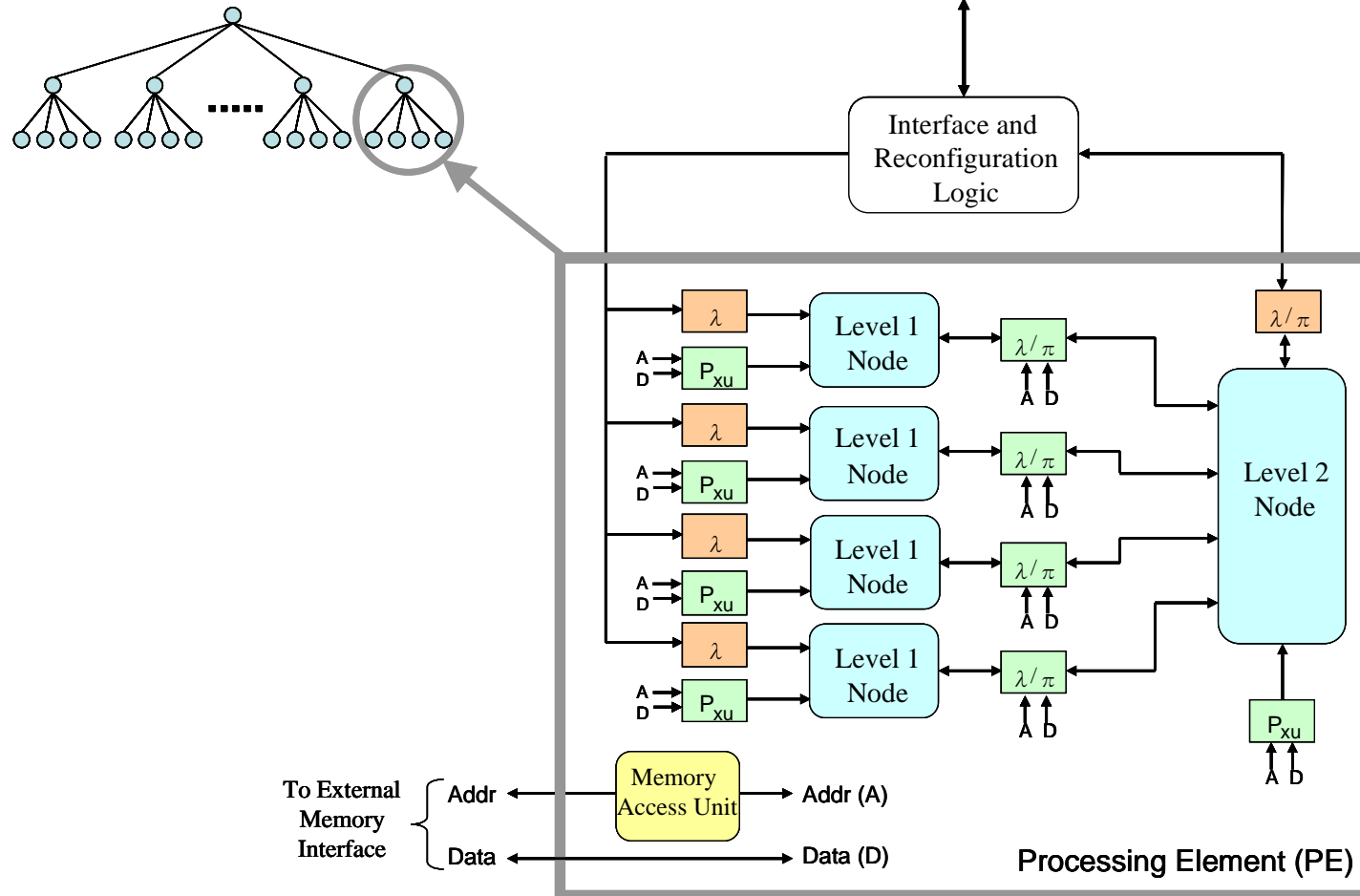
◆ Implemented on a Cray XD1



- ◆ Hierarchical temporal memory (HTM) is a machine learning model developed by Jeff Hawkins and Dileep George of Numenta, Inc. that models some of the structural and algorithmic properties of the neocortex.

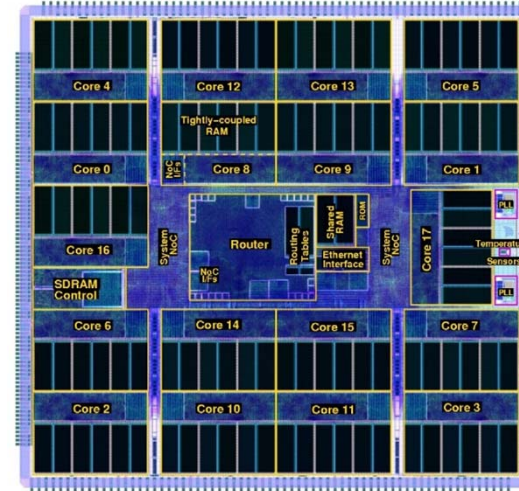
PEs on FPGA

- ◆ In an artificial neural network, neurons can take many forms and are typically referred to as Processing Elements (PE) to differentiate them from the biological equivalents



Other examples

- ◆ 18 ARM9 cores simulating the brain



- ◆ Accelerating specific applications

- Typically machine learning problems -> Hardware neuronal networks
- Pattern recognition
- Filtering, etc.
- Check the works of Olivier Temam (INRIA)

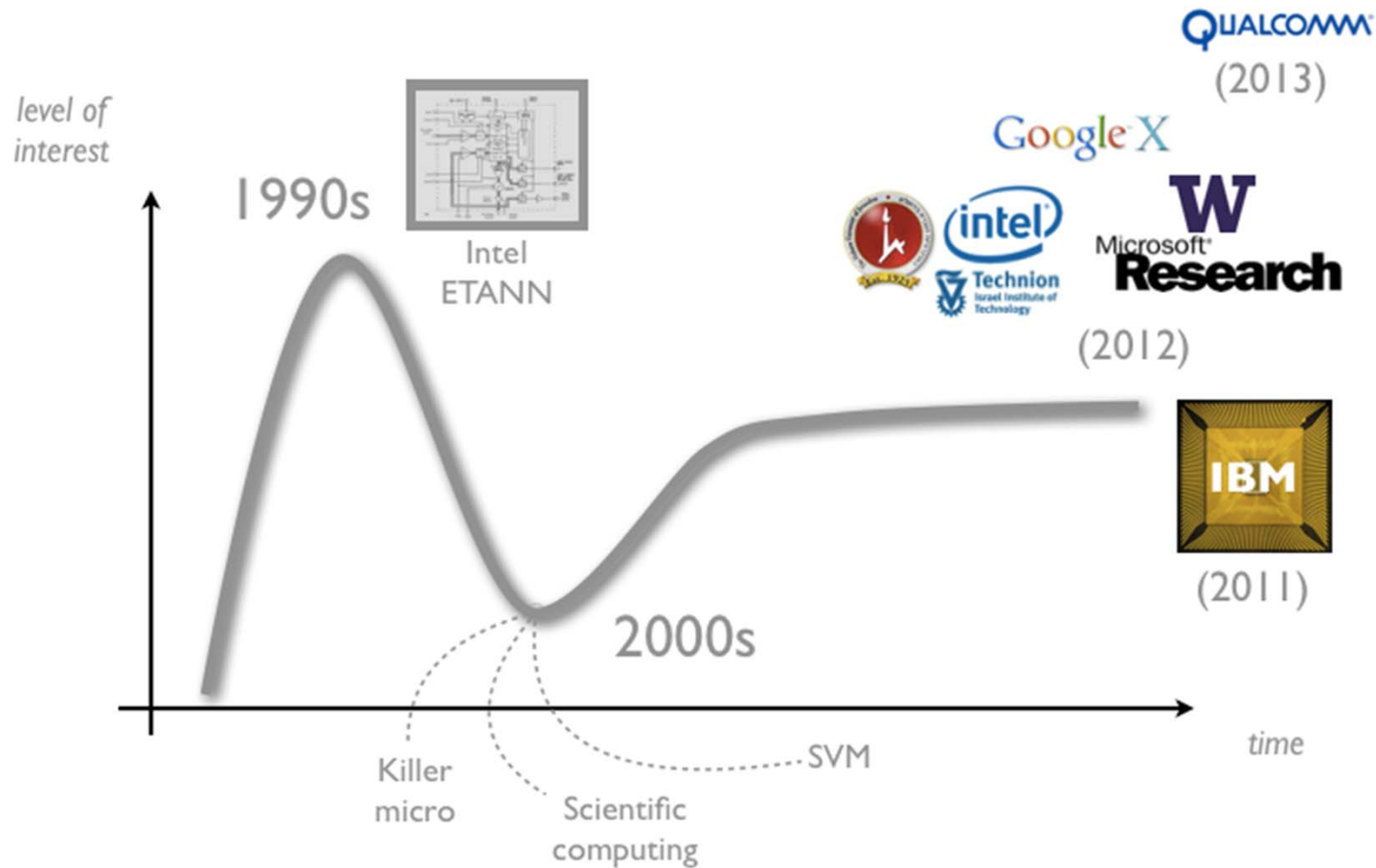
Another approach:
Simulation (i.e. software)

Large Scale Simulations

- ◆ Human Brain Project, EU 2013
- ◆ BRAIN Initiative, USA 2012

- ◆ Previously:
 - IBM:
 - ◆ Blue Brain Project: IBM & EPFL (Switzerland)
 - ◆ IBM Almaden Research Center
 - Los Alamos National Lab
 - Air Force Research Laboratory
 - Academia:
 - ◆ Portland State University
 - ◆ Royal Institute of Technology (KTH, Sweden)

Going anywhere?



Hype Curve of (Hardware) Neural Networks