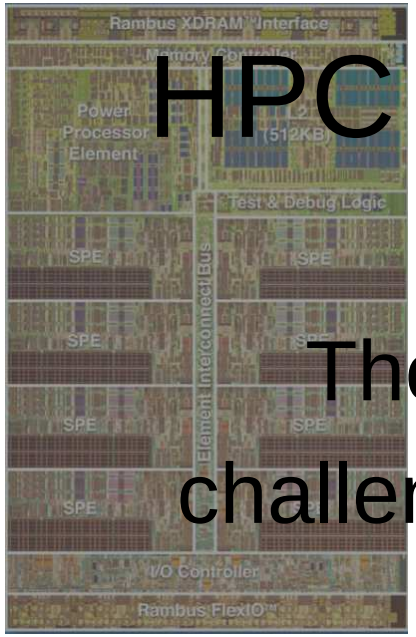


HPC for Fun and Profit

The CELL processor and the challenges it will present to us…

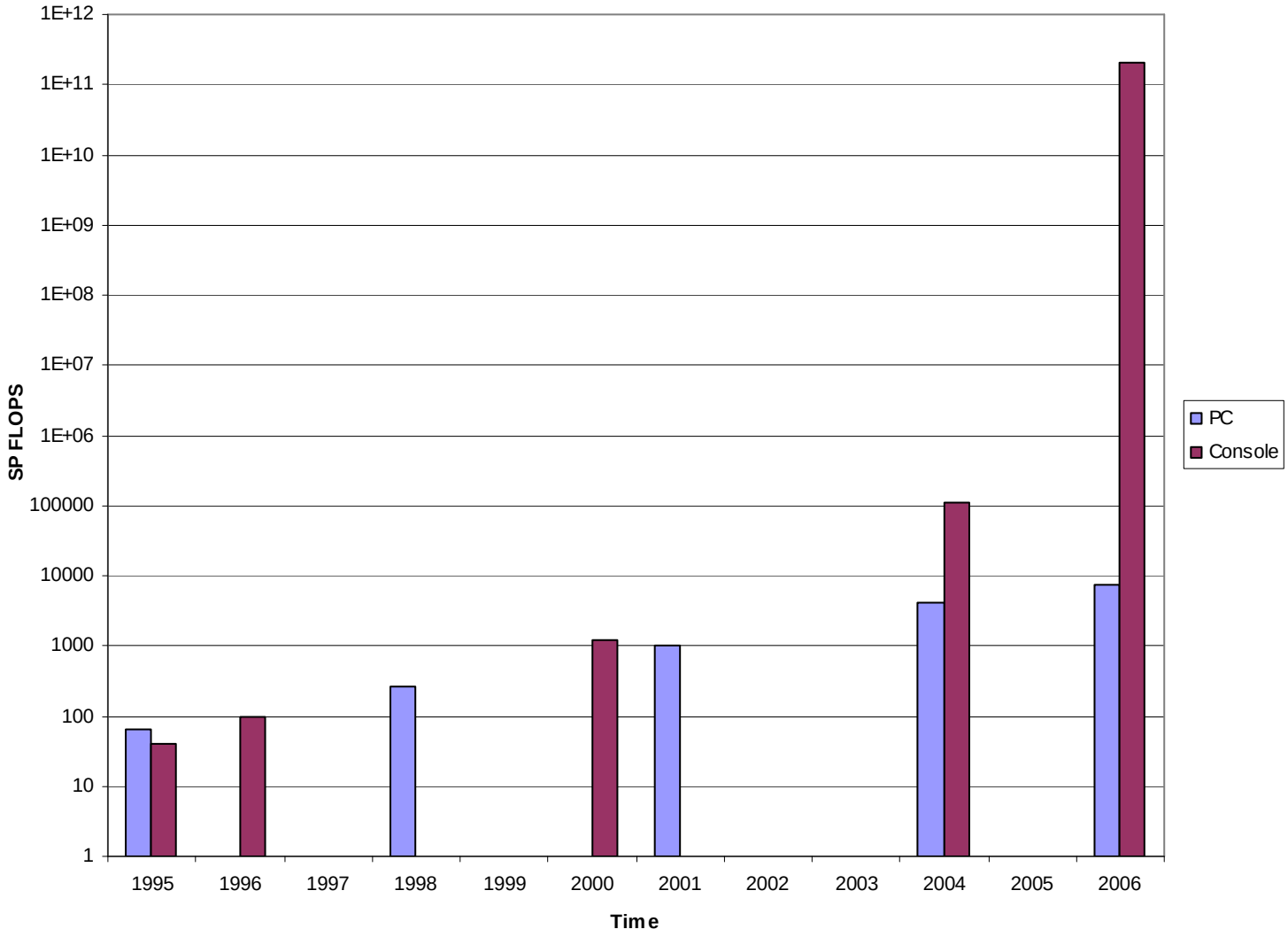
Supercomputing

Year	Rank	Manufacturer	Computer	Procs	Rpeak
1993	1	TMC	CM-5	1024	131
2002	500	IBM	Power3	132	198

Basics

- Multi-core microprocessor (9 cores)
- ~250M transistors
- ~235mm2
- Top frequency >3GHz
- 9 cores, 10 threads
- > 200+ GFlops (SP) @3.2 GHz
- > 20+ GFlops (DP) @3.2 GHz
- Up to 25.6GB/s memory B/W
- Up to 50+ GB/s I/O B/W

Performance Development



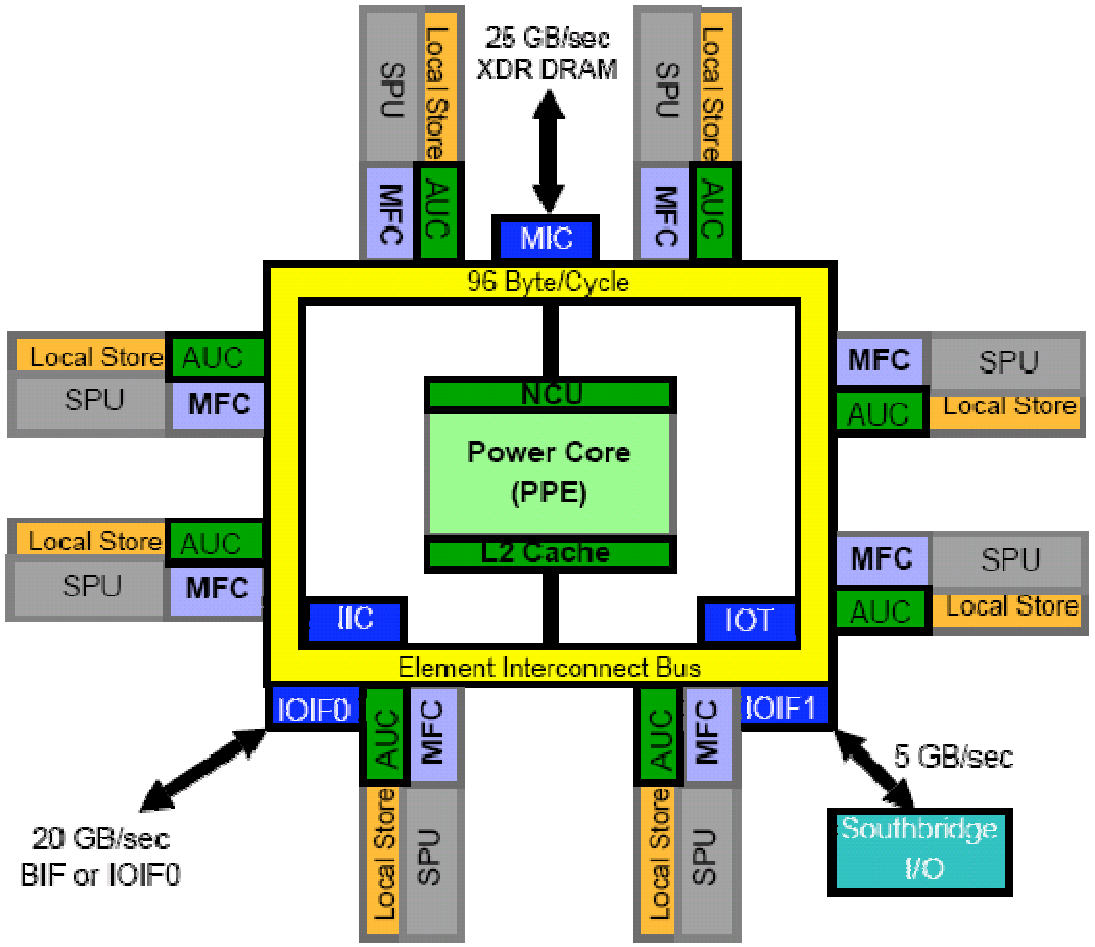
The challenges

- Memory Wall
  - Latency induced bandwidth limitations
- Power Wall
  - Must improve efficiency and performance equally
- Frequency Wall
  - Diminishing returns from deeper pipelines

Addressing the challenges with CELL

- Multi-Core Non-Homogeneous Architecture
  - Control Plane vs. Data Plane processors
  - Attacks Power Wall
- 3-level Model of Memory
  - Main Memory, Local Store, Registers
  - Attacks Memory Wall
- Large Shared Register File & SW Controlled Branching
  - Allows deeper pipelines
  - Attacks Frequency Wall

CELL Overview



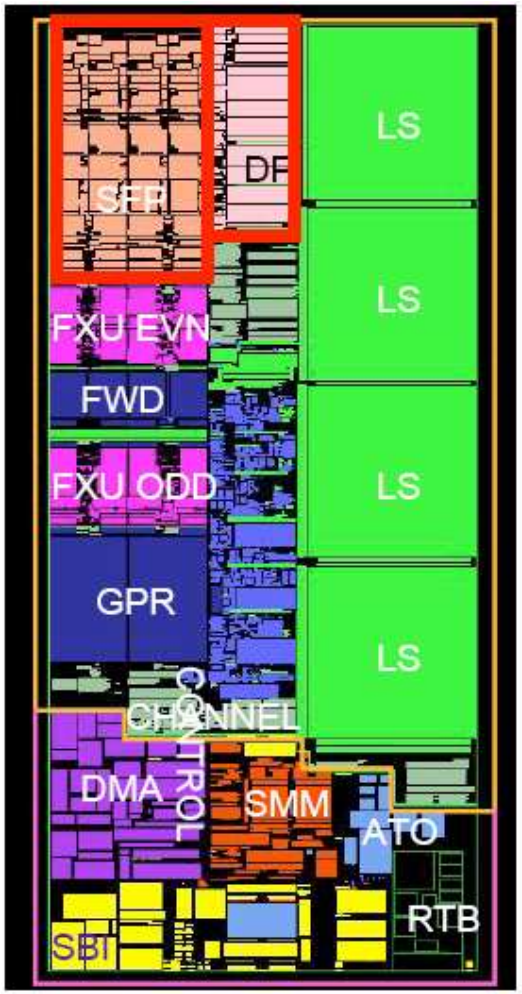
EIB

- 4 ring busses
  - 2 in each direction
  - 16 B per ring
  - 32 outstanding transactions per bus
- Peak at 96B per cycle

PPE

- 64 bit Power CPU
- 128 bit VMX
  - If you want this thing to do graphics
- 32 KB + 32 KB Level 1 cache
  - Instruction and data
- 512 KB Level 2 cache
- 2 hardware threads

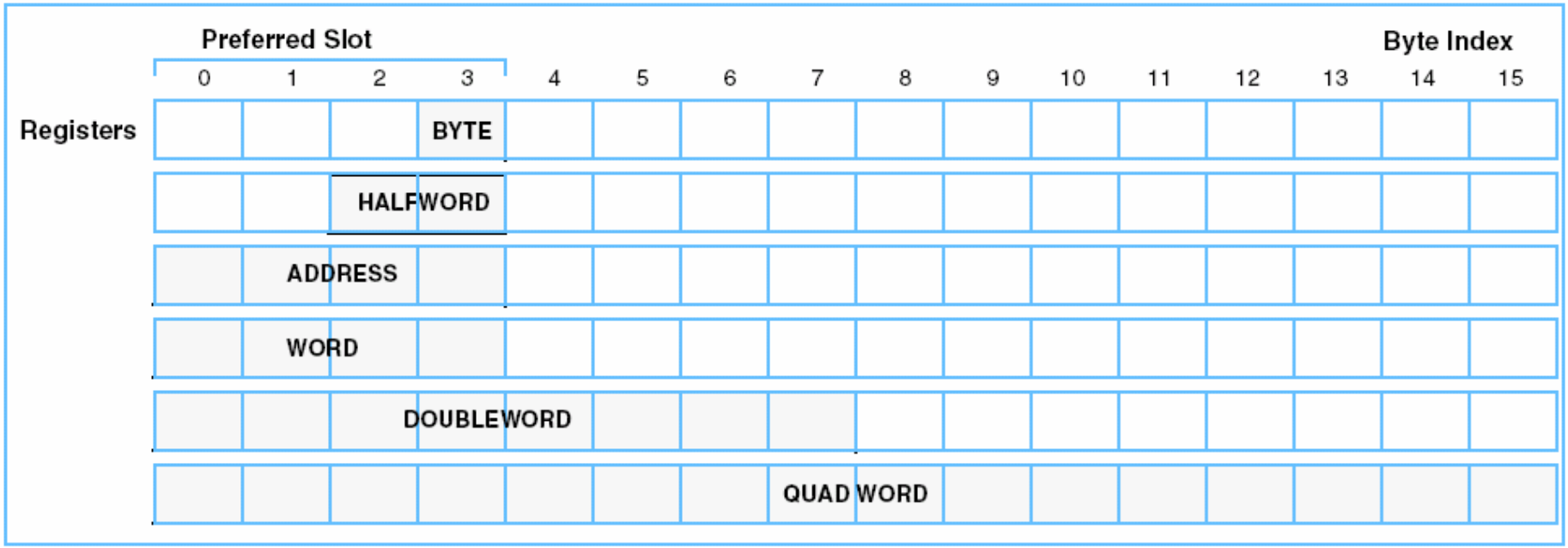
SPE



- RISC like organization
  - 32 bit fixed instructions
  - Clean design – unified Register file
- User-mode architecture
  - No translation/protection within SPU
  - DMA is full Power Arch protect/x-late
- VMX-like SIMD dataflow
  - Broad set of operations (8 / 16 / 32 Byte)
  - Graphics SP-Float
  - IEEE DP-Float
- Unified register file
  - 128 entry x 128 bit
- 256KB Local Store
  - Combined I & D
  - 16B/cycle L/S bandwidth
  - 128B/cycle DMA bandwidth

The register file

- 128 general purpose registers
  - 2KB!!!
- May be used in any size



SPE is a SCB

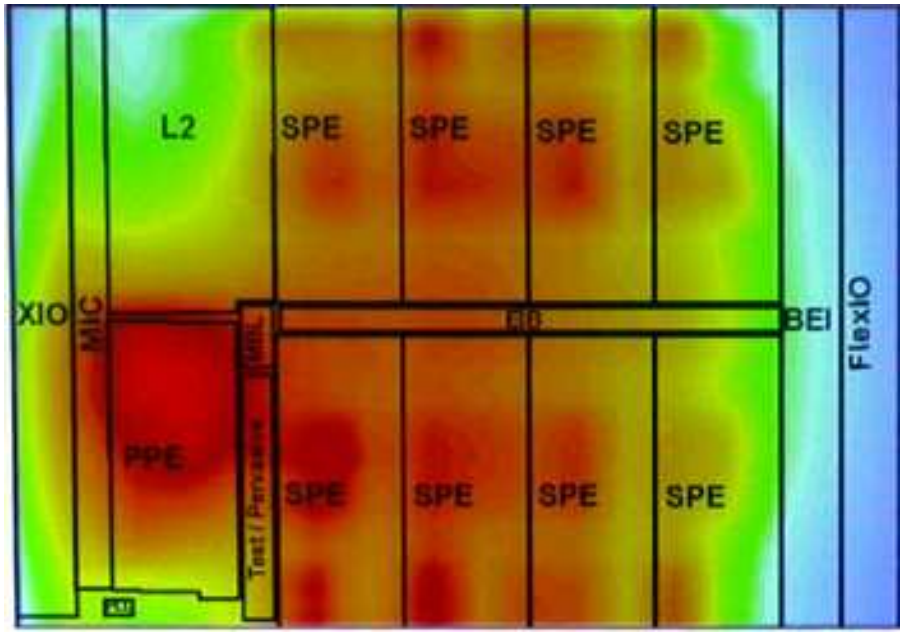
Unit	Instructions	Execution pipe	Unit Pipeline Depth	Instruction Latency
Simple Fixed	word arithmetic, logicals, counting leading zeros, selects and compares	Even	2	2
Simple Fixed	word shifts and rotates	Even	3	4
Single Precision	multiply-accumulate	Even	6	6
Single Precision	integer multiply-accumulate	Even	7	7
Byte	pop count, absolute sum of differences, byte average, byte sum	Even	3	4
Permute	Quadword shifts, rotates, gathers, shuffles as well as reciprocal estimates	Odd	3	4
Load Store	Load and Store	Odd	6	6
Channel	Channel Read/Write	Odd	5	6
Branch	Branches	Odd	3	4

Local store

- Default Size will be 256KB / SPE
- Using the Local Store is going to be all important for performance
  - Load from LS in 6 cycles
  - Enqueue DMA transfer in 20 cycles
  - 16B/cycle L/S bandwidth
  - 128B/cycle DMA bandwidth

Power

- Power consumption is expected in the 50-80W range
  - But it looks to be used correctly



Programming the beast

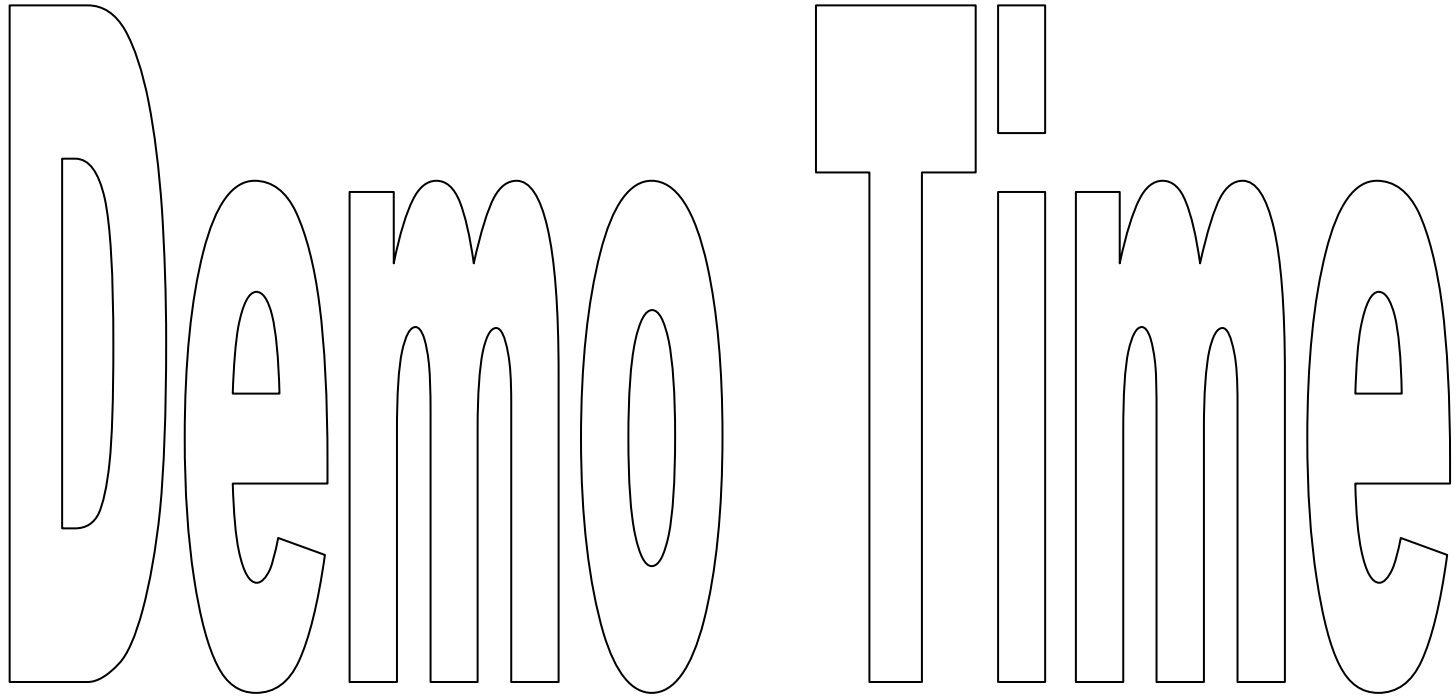
- For optimal performance we need to keep 10 threads running
  - 2 PPC threads
  - 8 SPE threads
- For these threads to run wee need to keep filling the Local Stores
  - Thus another 8 DMA engines
- This means we need to keep 18 functional units active at any time!!!

The responsibility returns to the programmer

- Now we have to fetch the memory in advance
  - No more relying on caches for performance
  - No more having to write your code to match the cache behavior
- Now we have to do branch prediction
  - You have to hint the CPU if a branch is take or not
- The SPE has two special pipelines
  - One for load, store and branch
  - One for arithmetic operations

The responsibility returns to the programmer

- The SPE cache is software managed
  - spe_load
  - spe_store
  - spe_lock
  - spe_flush
  - spe_prefetch


Demo Time

Programming the CELL

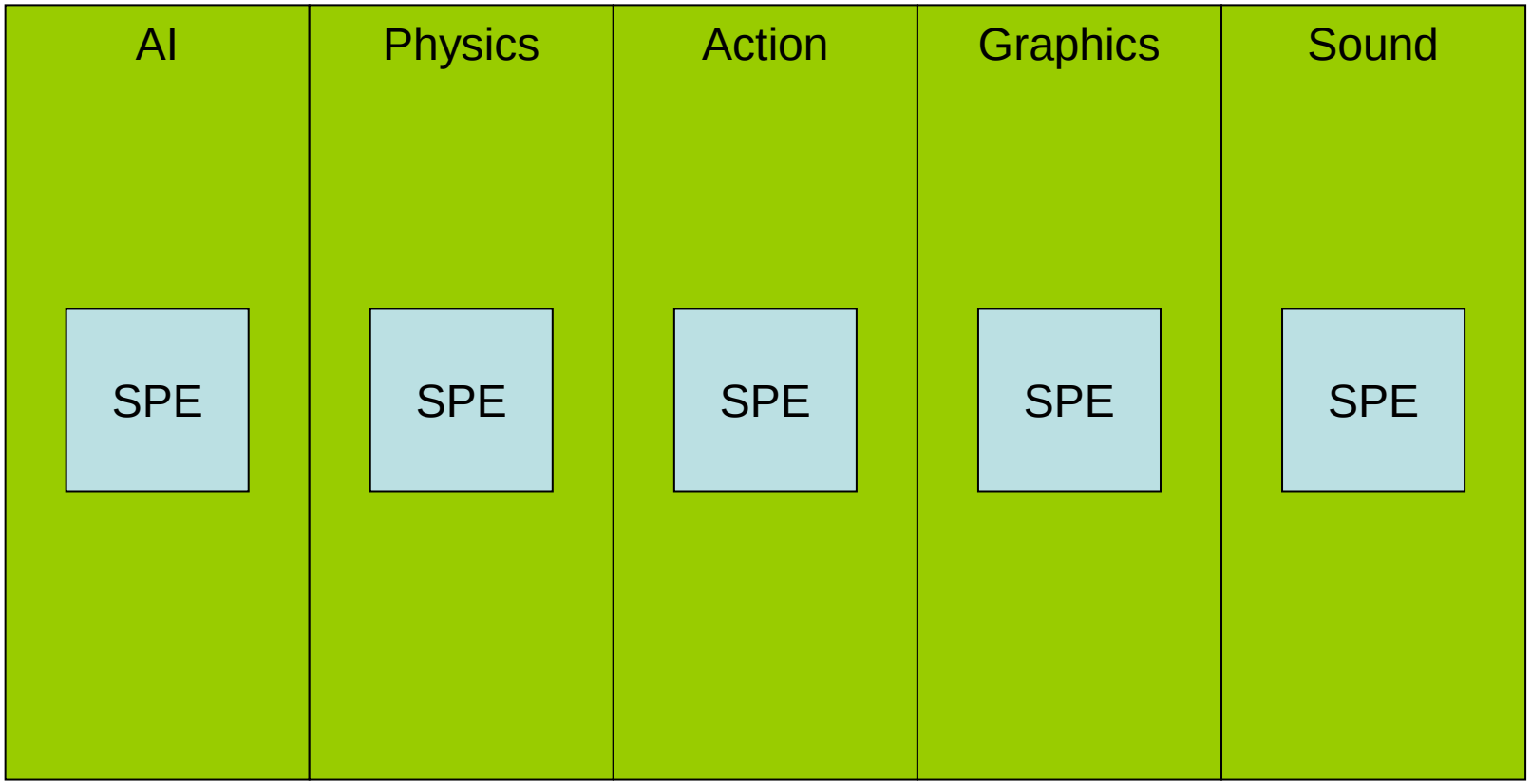
- Slice the processor
  - Use each SPE for something special
- Pipeline the processor
  - Combine the SPEs for execution
- Bag of tasks
- CSP

Sliced CELL

- Take the SPE processors and use them as special purpose processors
- Each processor is dedicated to each processor

Sliced CELL

Sliced CELL



Advantages of slicing

- Really really easy
- Easy to develop and test in components

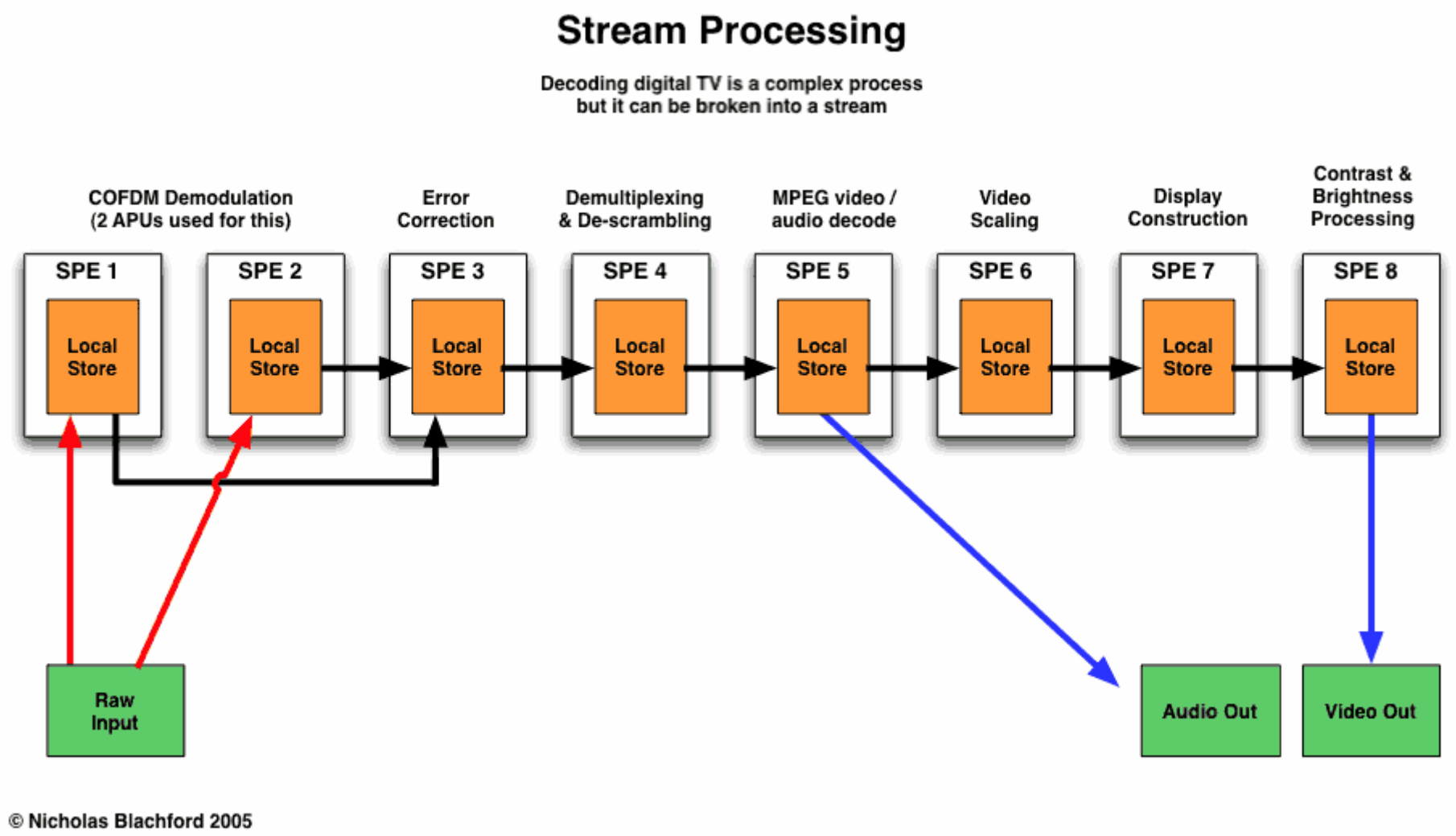
Problems with slicing

- Will you have exactly 7 dedicated tasks?
  - More than 7?
  - Less than 7?
- Will each task require equal amount of processing power?
- Will all tasks be able to run at the same time?
- What is the communication between tasks?

Pipelined CELL

- Divide your application into a software pipeline
- Complex operations are split over multible SPEs

Pipelined CELL



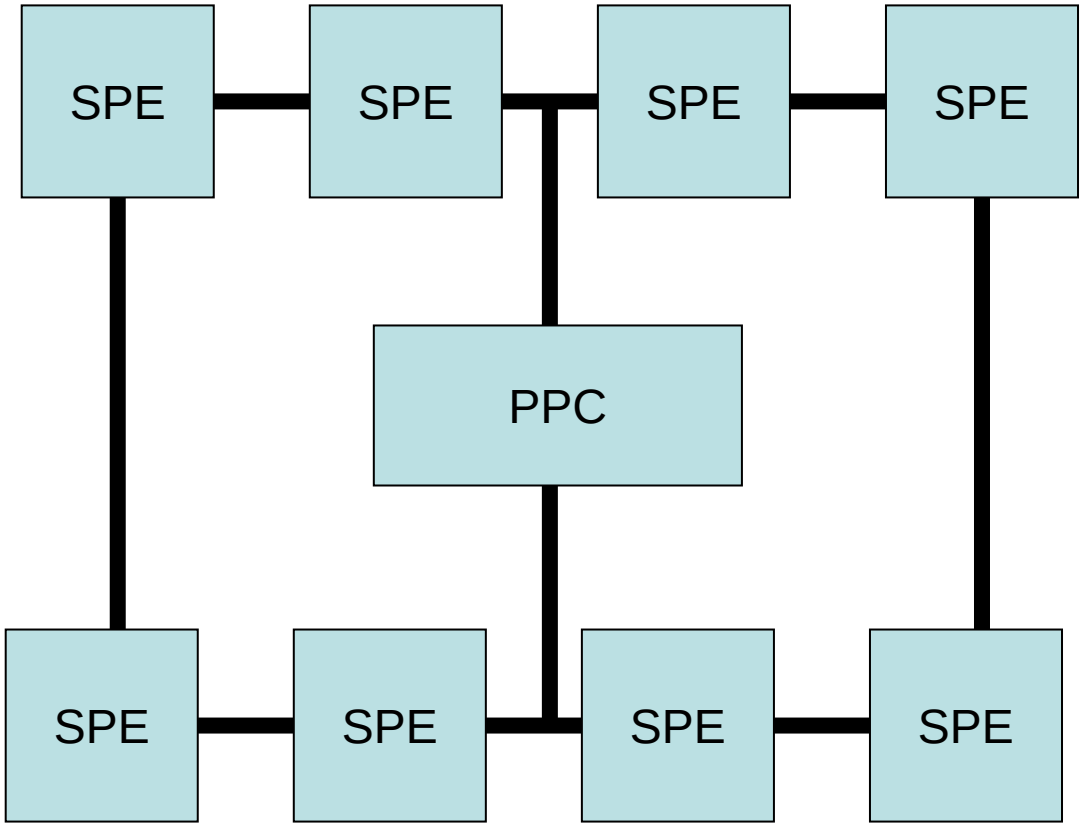
Advantages of pipelining

- Allow high ’operations per second’ for inherently sequential code
- Only nearest nighbor communication

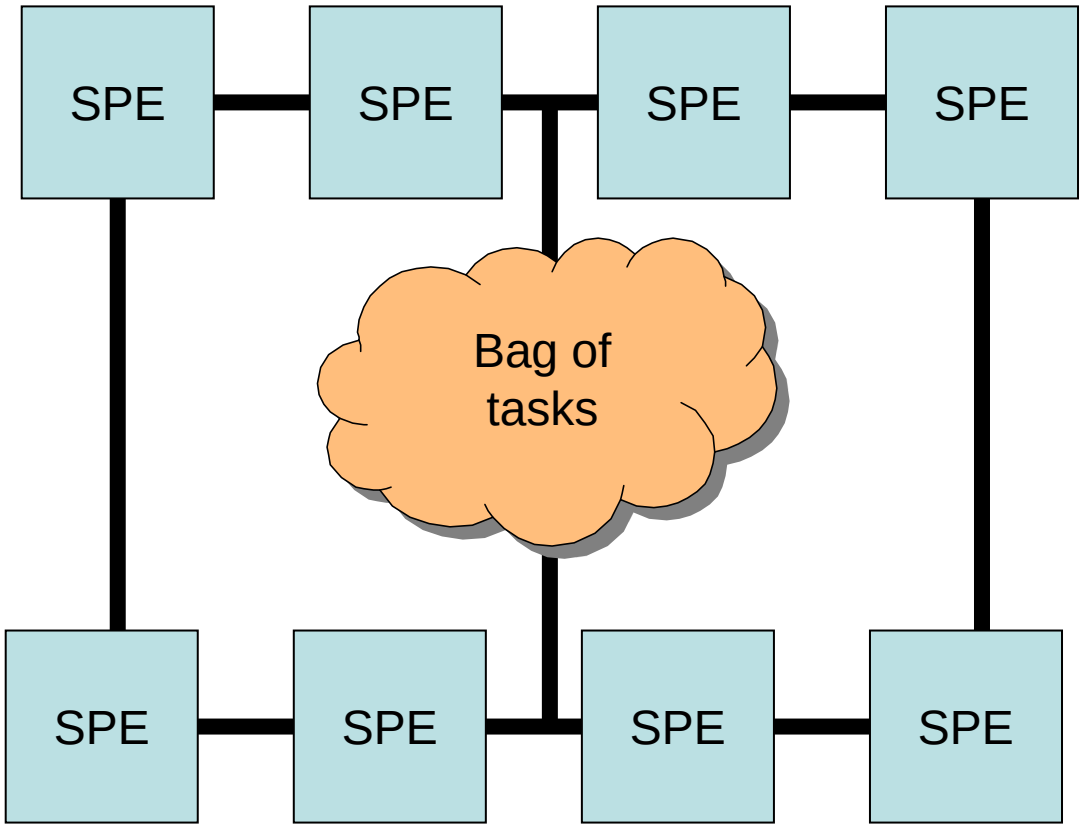
Problems with pipelining

- Splitting operations in a pipeline is often hard and sometimes impossible
- Making sure that all steps are equally hard is impossible
- Communication can be very high
  - Because we are exchanging intermediate data

Bag of Tasks



Bag of Tasks



Advantages of Bag-of-tasks

- Extremely easy
- Fits most gaming models
  - Event handling is simply added to the bag og tasks
- Utilizes all SPEs evenly
  - If there are enough tasks

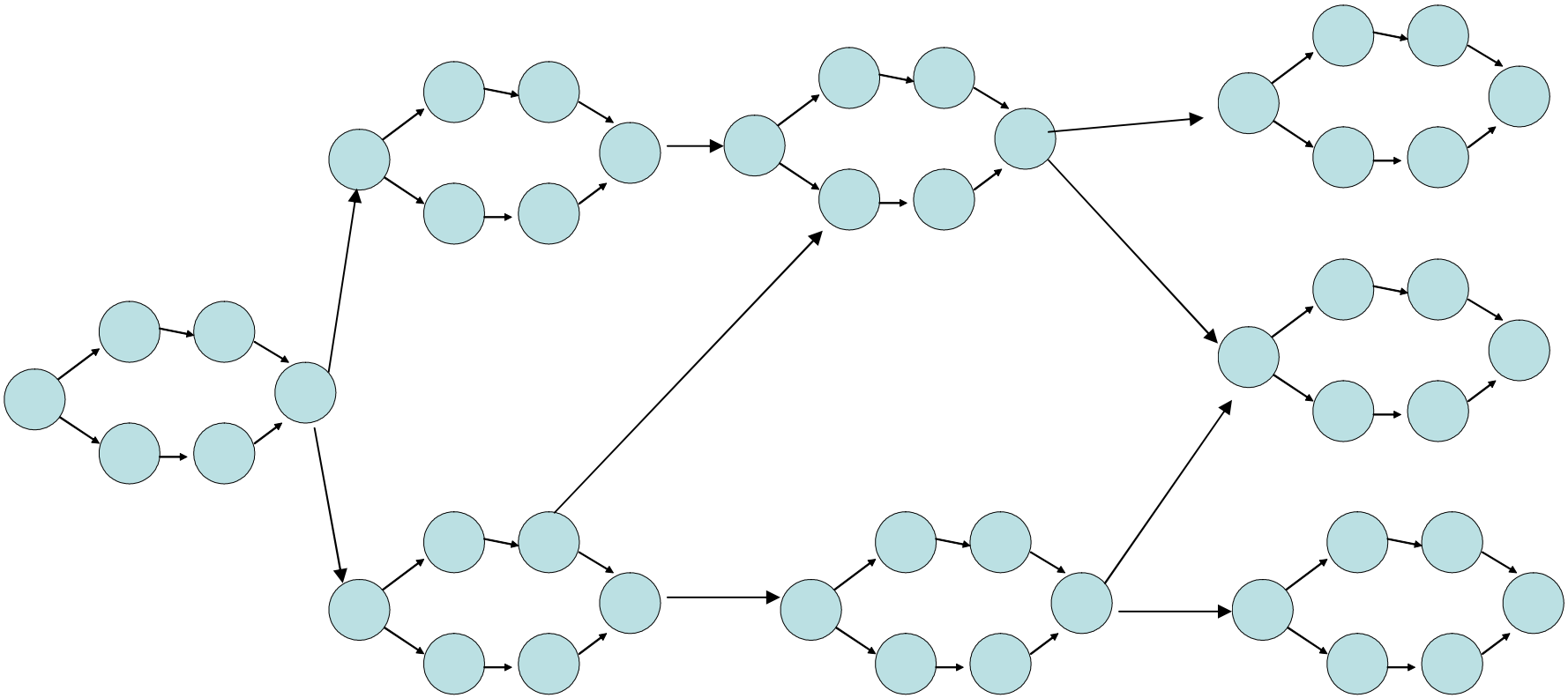
Problems with Bag of tasks

- PPC can become bottleneck
- PPC cannot be used for computations
- Communication can be very high
  - Local store use becomes very hard
- Latency of action can become very high

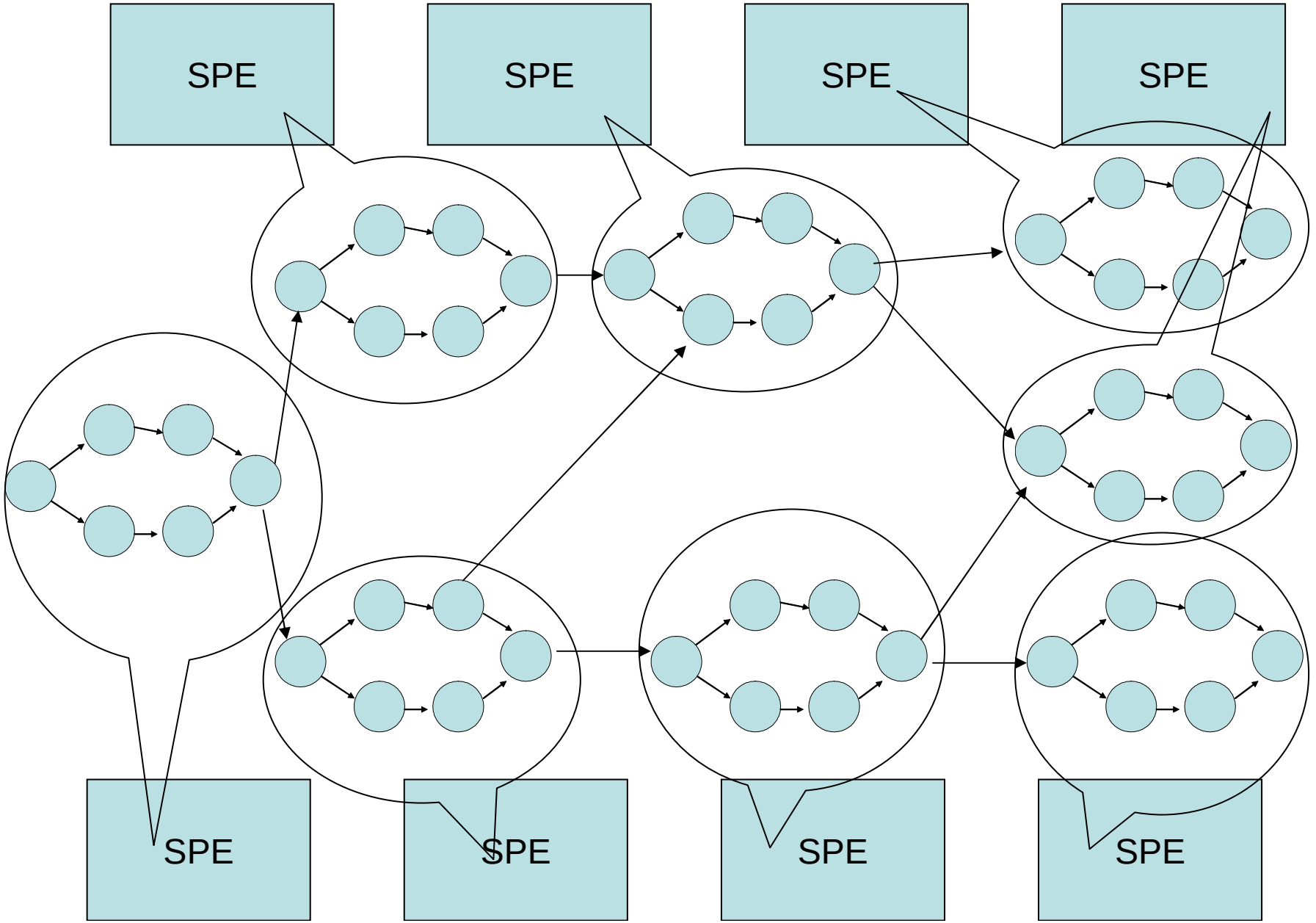
CSP

- Program your application as a CSP network
  - Make sure you have enough processes
    - Enough >> 7
- Let the scheduler place processes on SPEs
  - Each SPE schedules locally

CSP



CSP



Advantages of CSP

- No consideration of the underlying architecture when determining parallelism
  - Porting to other architectures is easy
- Dynamic load balancing
- Local control with global coordination

Problems with CSP

- No tools exists
- CPS kernel must be implemented with knowledge of the architecture
  - Special consideration to the heterogeneous aspects