

Lecture 19

The Future of High Performance Computing

Today's lecture

- Architectural directions
 - Blue Gene / L
 - STI CELL, including a brief introduction to instruction level parallelism

Blue Gene/L

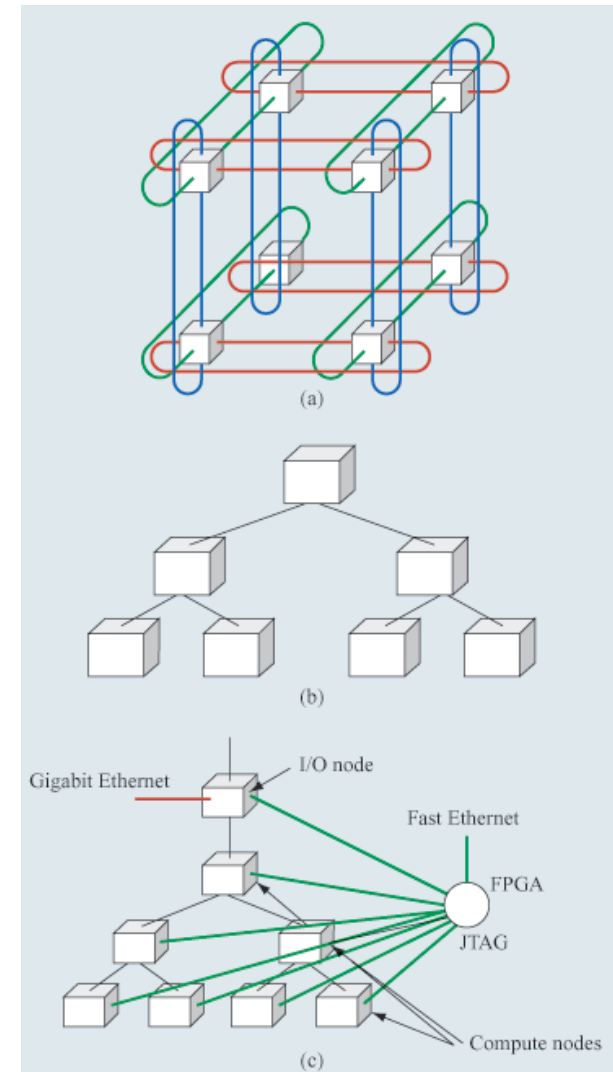
- IBM-US Dept of Energy collaboration
- 64K dual processor nodes: 180 (360) TF peak
 - One CPU dedicated to communication on each node (But may be used for computation)
- Low power
 - Relatively slow processors; power PC 440
 - Small memory (256 MB)
- High performance interconnect
- Lightweight kernel runs on each node
 - Single user, single 2-thread process
 - No context switching, no demand paging
- SDSC has 3072 nodes (6144 processors)

Blue Gene/L Interconnect

- 3D toroidal mesh (end around)
 - 175 MBsec Bandwidth peak
 - bidirectional
- Rapid combining-broadcast network
 - 350MB/sec
 - 1.5 μ s latency (1-way)
 - Fast barriers (1.5 μ s)

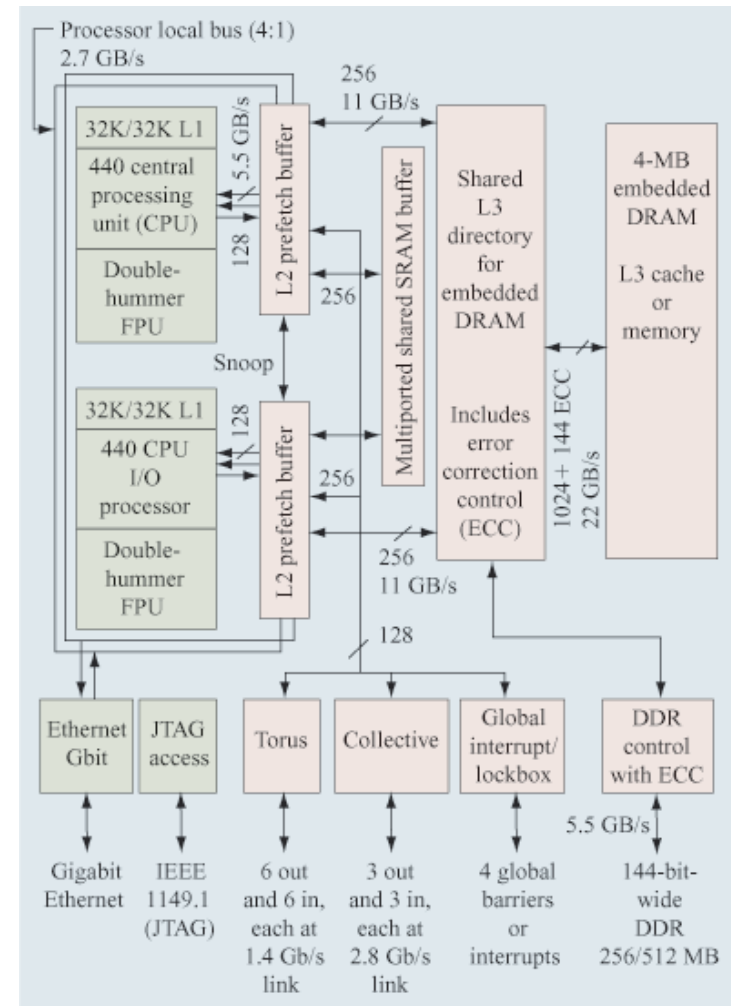
Image courtesy of IBM

<http://www.research.ibm.com/journal/rd/492/gara4.gif>



Blue Gene/L Compute chip

- Shared L3 Cache
- L1 caches not coherent



G. Almasi et. Al, © 2002 IEEE

STI CELL

A brief introduction to Instruction Level Parallelism

What is instruction-level parallelism?

- The potential for executing certain instructions in parallel, because they are independent
- Any technique for identifying and exploiting such opportunities
 - Static: can be implemented by a compiler
 - Dynamic: requires hardware support

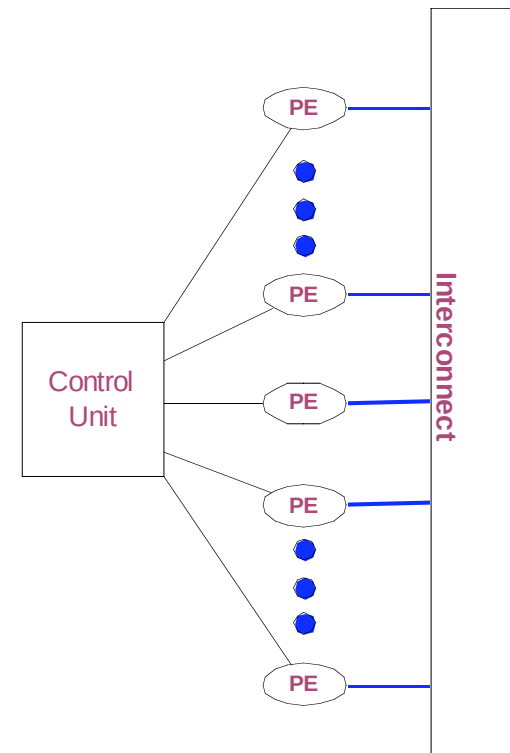
How does it manifest itself?

- Basic blocks
 - Sequences of instruction that appear between branches
 - Usually no more than 5 or 6 instructions!
- Loops

```
for ( i=0; i<N; i++)  
    x[i] = x[i] + s;
```
- We can only realize ILP by finding sequences of independent instructions
- Dependent instructions must be separated by a sufficient amount of time, that is functional unit latency

SIMD Instruction Set

- Operations defined on vectors
- Modern architectural extensions
 - Intel SSE = Streaming SIMD extensions
 - AltiVec: Apple Computer, IBM and Freescale Semiconductor
 - Both support 128-bit vector registers
 - Represent 16 x 8-bit chars, ... 4 x 32-bit integers, floating point ...



Limitations of ILP

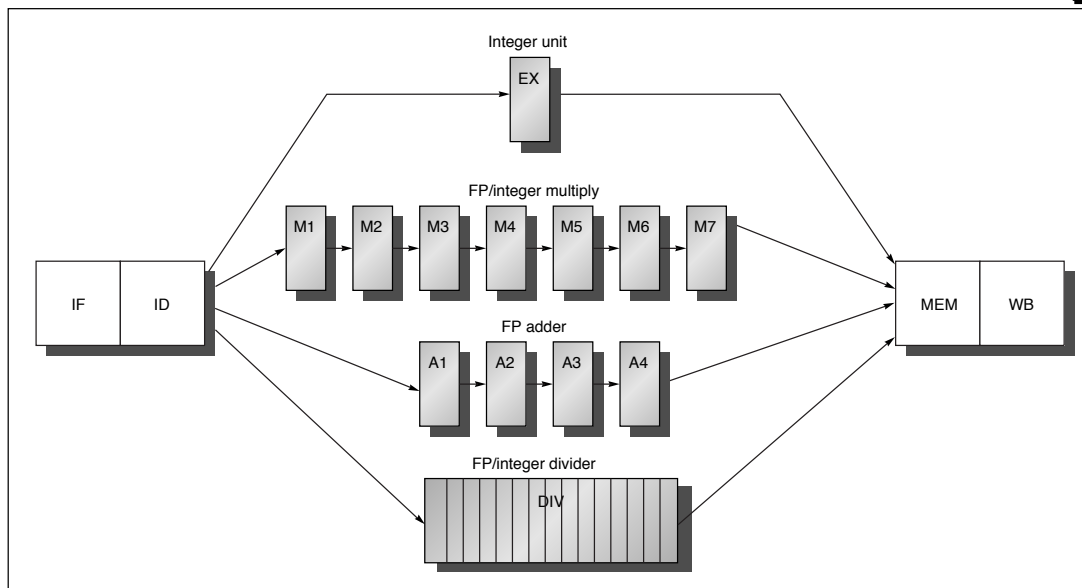
- Recall the loop:
for (i=0; i<N; i++)
 x[i] = x[i] + s;

L:LD	F0, 0(R1)	; F0 is the vector element
ADDD	F4, F0, F2	; add the scalar
SD	0(R1), F4	; store the result
SUBI	R1, R1, #8	; decrement by 8
		; to previous word
BNEZ	R1, L	; Branch if R1 $\neq$ 0
NOP		; Delayed branch slot

Delays

```
for ( i=0; i<N; i++)
    x[i] = x[i] + s;
```

Unit	Latency	Init Interval
Integer	0	1
Memory	1	1
FP Add	3	1
FP Multiply	6	1
FP Div	24	24

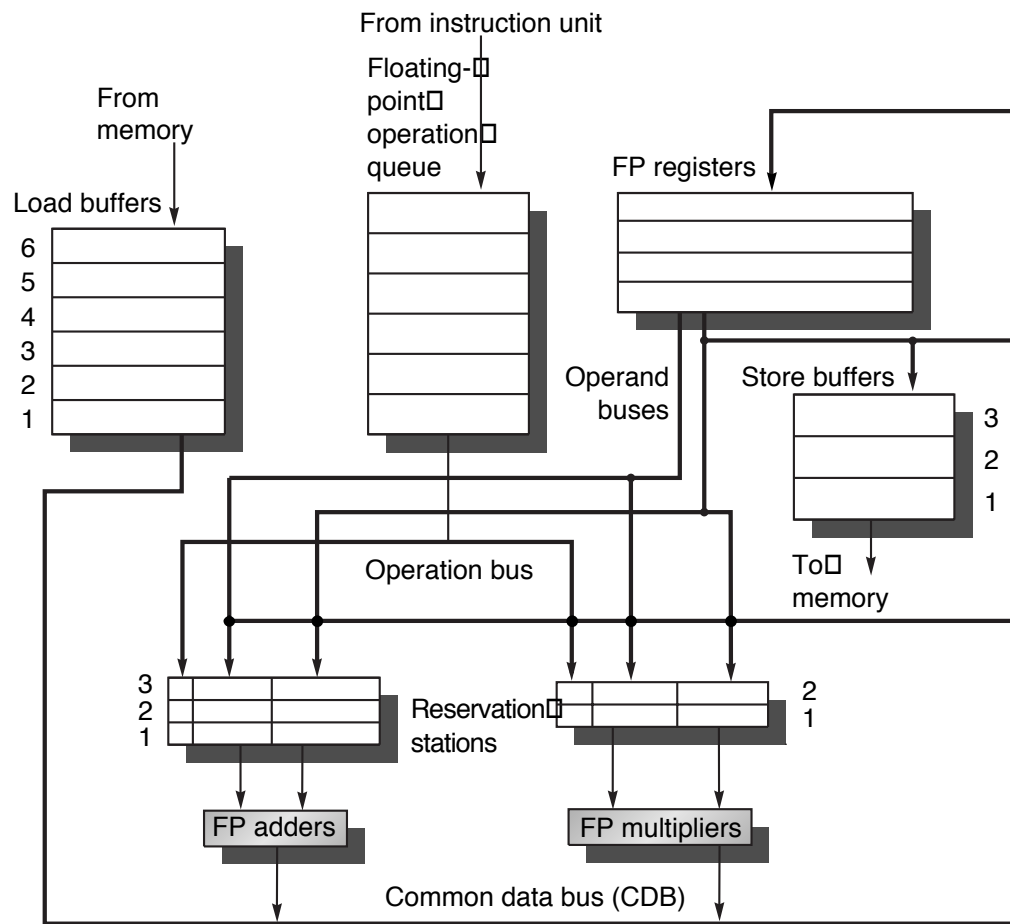


L: LD **F0**, 0(R1)
stall
 ADDD **F4**, **F0**, F2
stall
stall
 SD 0(R1), **F4**
 SUBI **R1**, R1, #8
 BNEZ **R1**, L
stall

Dynamic scheduling

- Idea: modify the processor to permit instructions to execute as soon as their operands become available
- This is known as *out-of-order execution*
- Complications: dynamically scheduled instructions also complete out of order
- Increased hardware complexity
 - Stations and execution units
 - Bookkeeping
 - Buffering
 - Slows down the processor

Dynamic scheduling (Tomasulo's Algorithm)



Unit	Latency
Integer	0
Memory	1
FP Add	2
FP Multiply	10
FP Div	40

The three steps of instruction execution

- Issue
 - Get instruction from fetch queue
 - If there are available resources buffer the register operands at station
 - If there is no such station, stall the instruction
- Execute
 - Monitor internal interconnect for any need operands
 - Intercept those operands and write to the station
 - When both operands are present, execute the operation
- Write result
 - Send to the result registers and any waiting units

Automatic, dynamic loop unrolling

```
L:  LD      F0, 0(R1)
    MULTD   F4, F0, F2
    SD      0(R1), F4
    SUBI    R1, R1, #8
    BNEZ    R1, L
```

Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	

Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	X
MULTD	F4, F0, F2	1	X	

Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	X
MULTD	F4, F0, F2	1	X	
SD	0(R1), F4	1	X	

Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	X
MULTD	F4, F0, F2	1	X	
SD	0(R1), F4	1	X	
LD	F0, 0(R1)	2	X	

Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	X
MULTD	F4, F0, F2	1	X	
SD	0(R1), F4	1	X	
LD	F0, 0(R1)	2	X	X

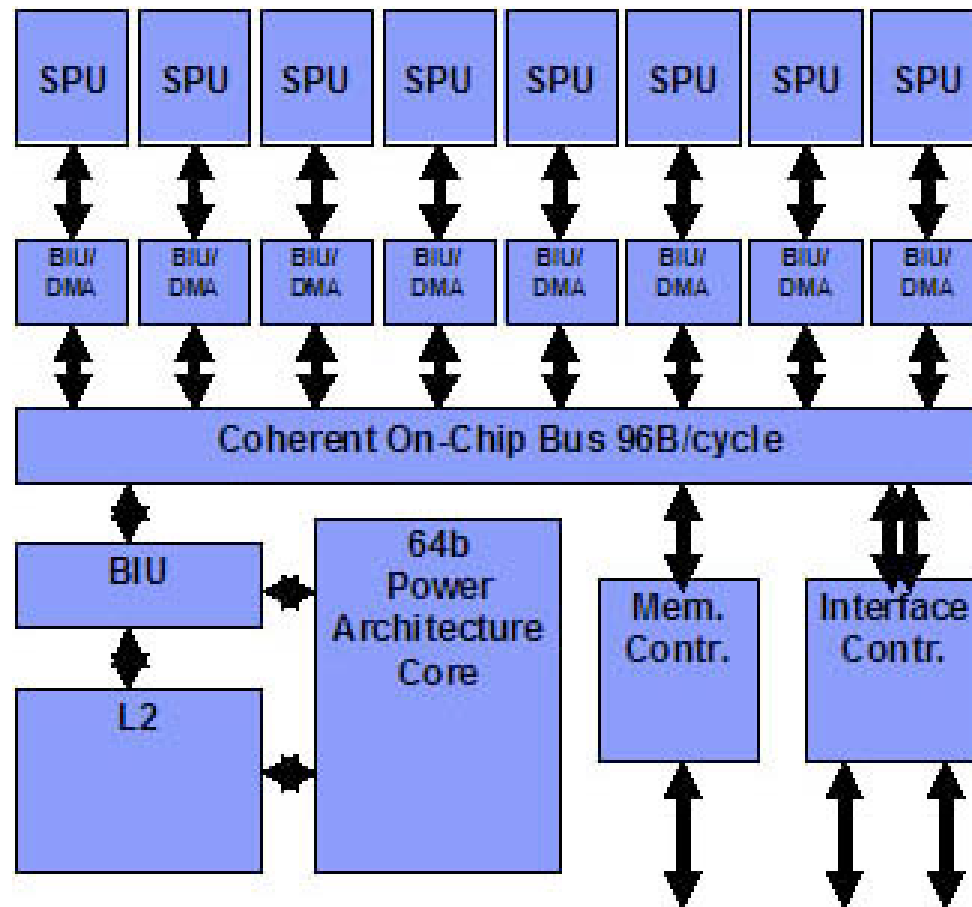
Automatic, dynamic loop unrolling

Instruction		Iteration	Issue	Exec
LD	F0, 0(R1)	1	X	X
MULTD	F4, F0, F2	1	X	
SD	0(R1), F4	1	X	
LD	F0, 0(R1)	2	X	X
MULTD	F4, F0, F2	2	X	
SD	0(R1), F4	2	X	

Cell Overview

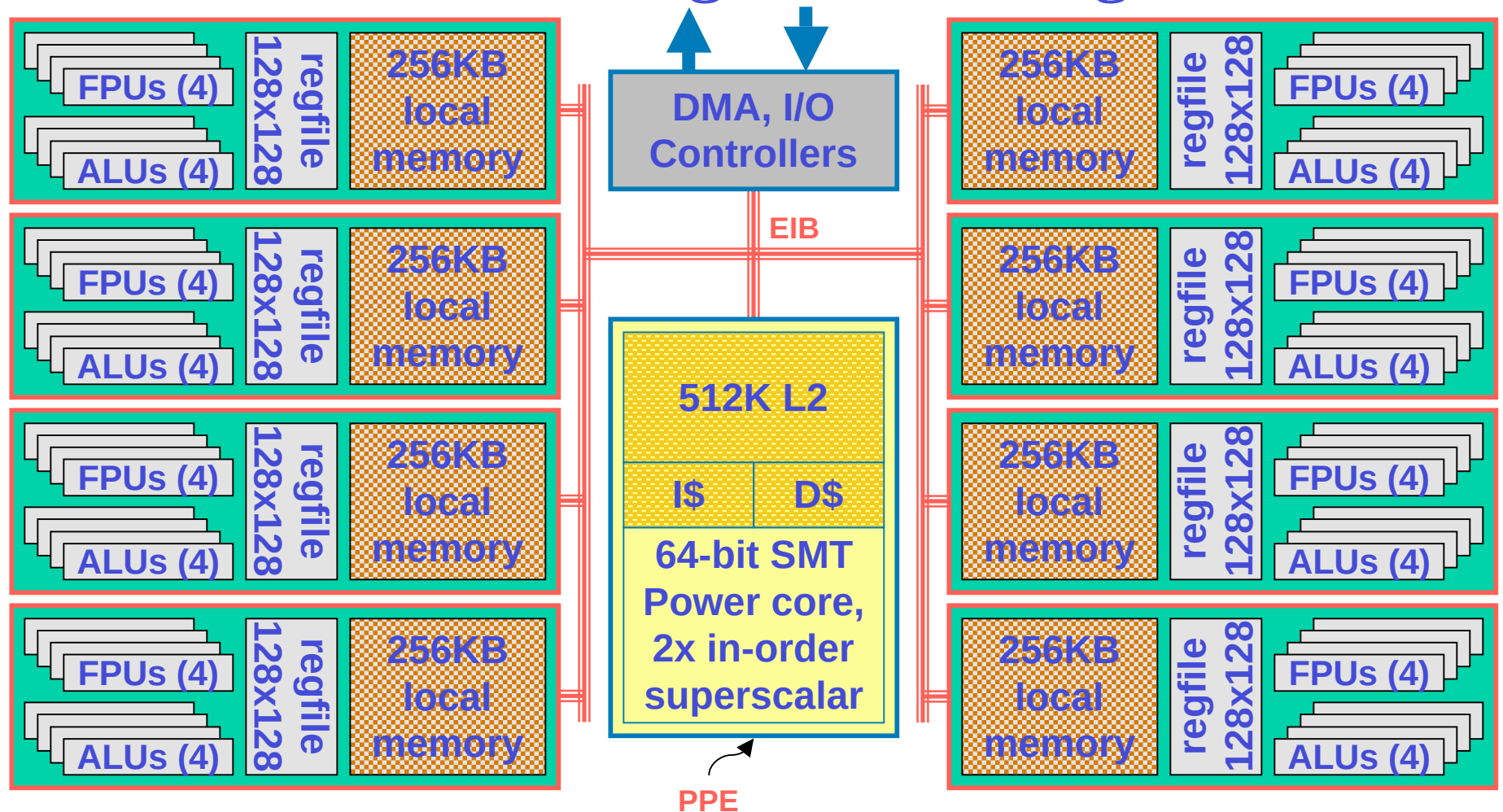
- Chip multiprocessor for multithreaded applications
- Adapted power PC: 8 SPE + 1 PPE
- Shared address space
- SIMD (VMX)
- Software managed resources on the SPE
- Optimize the common case

Block diagram



<http://domino.watson.ibm.com/comm/research.nsf/pages/r.arch.innovation.html>

CELL block diagram showing detail



Courtesy Sam Sandbote

SPE - Synergistic processing element

- 256 KB local store
- 128-bit x 128 entry register file
- DMAs $\Leftrightarrow$ Local Store via EIB
 - 4 x 128 bit wide
 - 200 GB/sec
- Rambus XDR DRAM interface
 - 3.2 Gbit/sec/SPE
 - 25.6 Gb/s aggregate

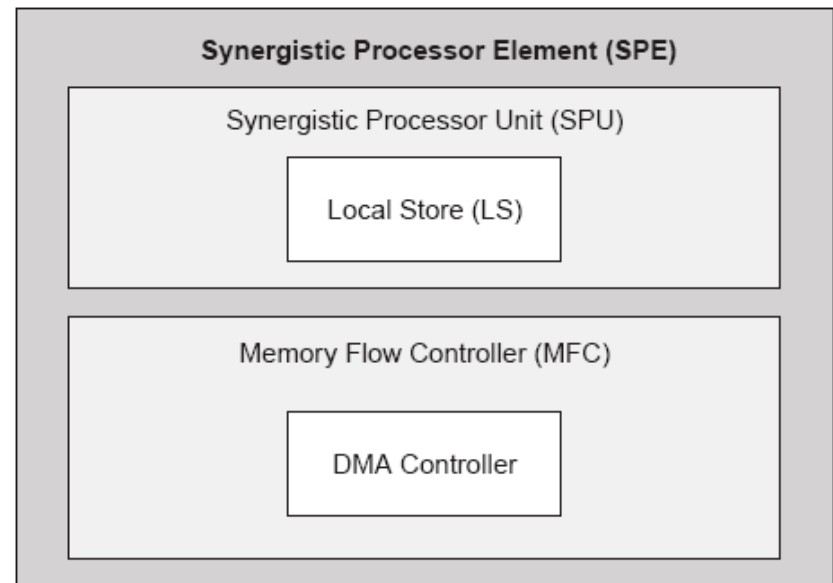
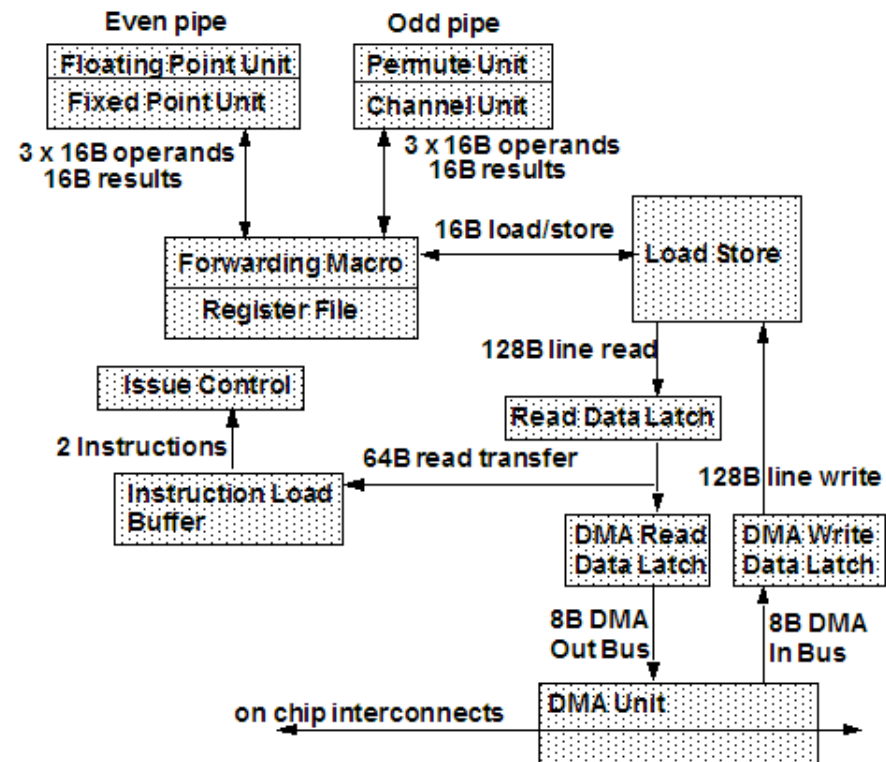


Image courtesy Keith O'Connor,
<http://isg.cs.tcd.ie/oconork/presentations/CellBroadbandEngine.ppt>

Inside the SPE

- SIMD (VMX)
 - Stacked operations
 - Unified vector and scalar operations
- Software managed resources
 - Optimize the common case
 - In order issue
 - Parallelism in VMX ops and across SPEs rather than ILP
 - Aligned accesses
 - Branch hints
 - No cache on SPWs: local memory
 - Large register files
 - Scheduling of SPEs



SPE Local Store

- Scheduling is an issue
 - Mostly in software
 - Want high utilization
- Multiple simultaneous requests
 - DMA
 - 16 x 16kb
 - Priority: DMA, L/S, Fetch

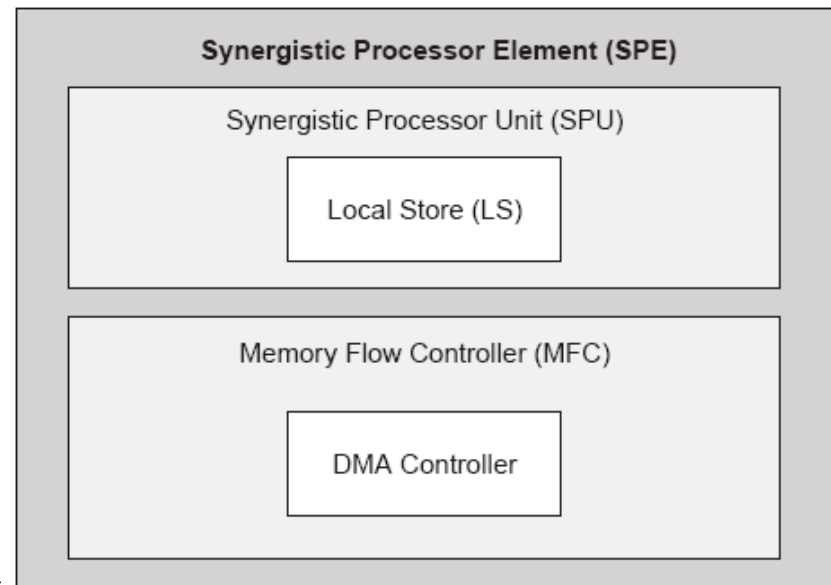


Image courtesy Keith O'Connor,
<http://isg.cs.tcd.ie/oconork/presentations/CellBroadbandEngine.ppt>

Usage scenarios

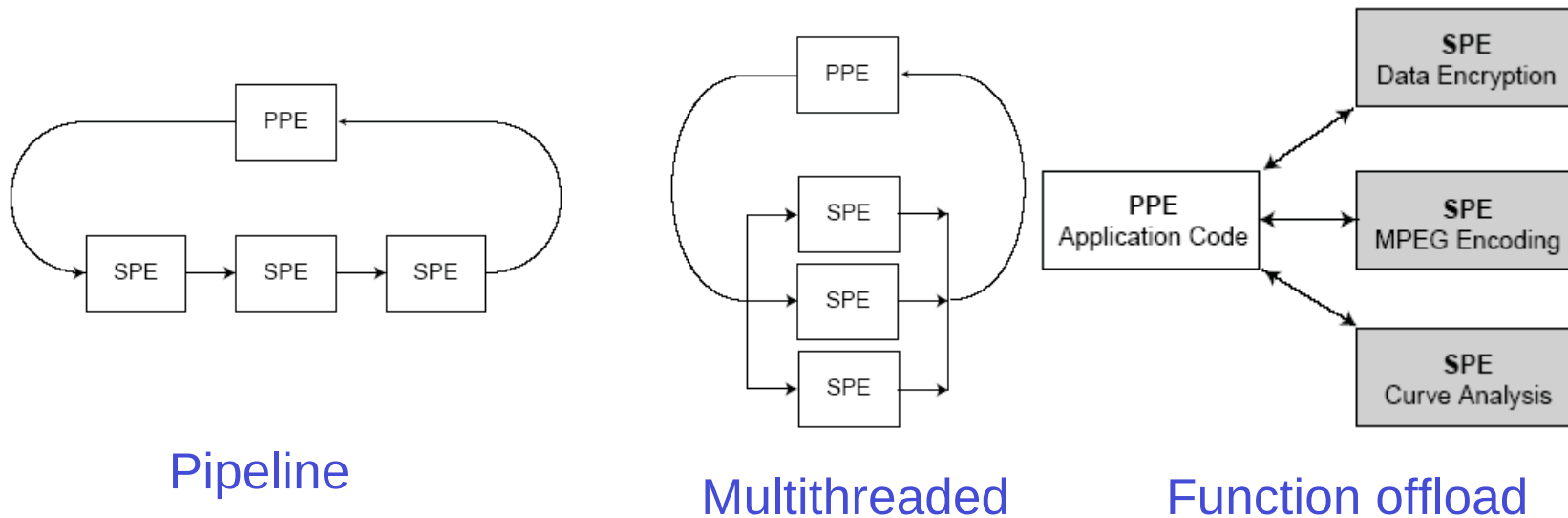


Image courtesy Keith O'Connor,
<http://isg.cs.tcd.ie/oconork/presentations/CellBroadbandEngine.ppt>