BSC~Microsoft Research Centre

New Parallel Programming Models

Siena, 9-13 Feb. 2009

Osman S. Unsal
Adrian Cristal

BSC – Computer Architecture for Programming Paradigms Group

Tentative course schedule

- MON 9/2: 9:00-->12:00
  - Introduction to Computer Architecture
  - Why is parallel programming so important now?
  - Basic TM concepts
- TUE 10/2: 9:00-->12:00
  - STM and HTM
  - TM Issues: I/O, privatization, failure atomicity
- WED 11/2: 10:00-->12:00 + 14:00-->16:00
  - Lab exercise I (working with Intel STM Compiler)
- THU 12/2: 10:00-->12:00 + 14:00-->17:00
  - Lab exercise II (writing TM applications)
- FRI 13/2: 9:00-->12:00
  - Discussion on other emerging programming models
  - Wrap-up
  - Quiz?

Where do we stand?

- Disclaimer: We will make a best effort at being impartial and not favoring HW over SW. BUT our background is more on HW.

Computation Evolution

•1854: Boolean Algebra by G. Boole

•1904: Diode vacuum tube by J.A. Fleming

•1938: Boolean Algebra and Electronics Switches,by C. Shannon

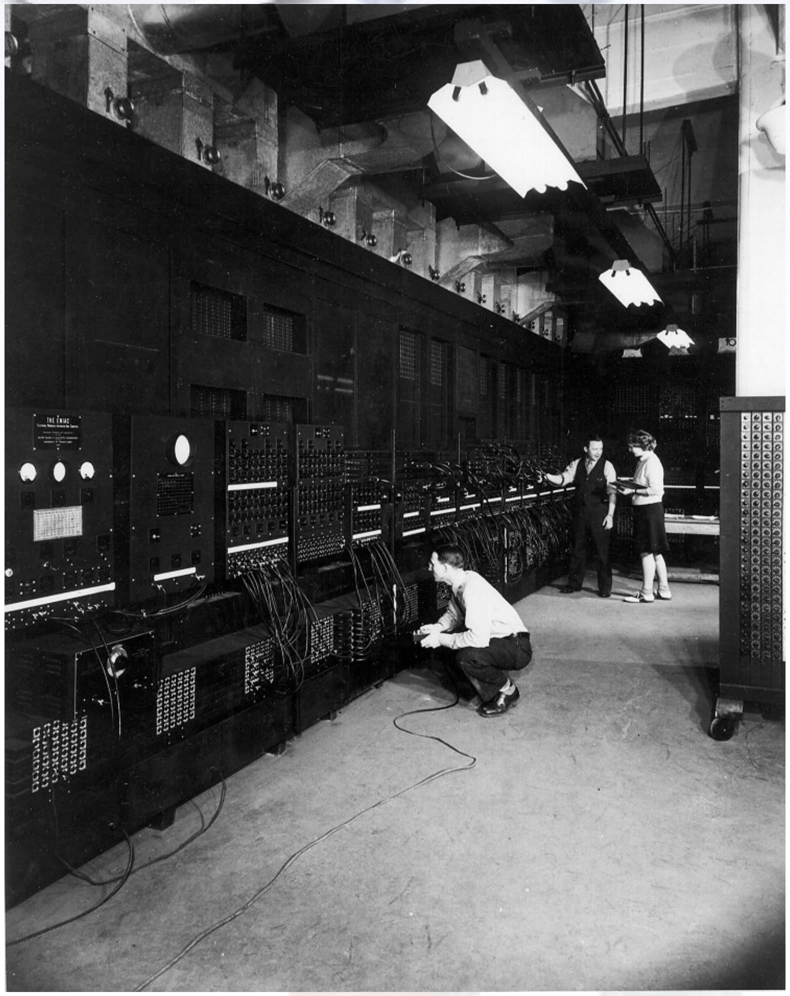
•1946: ENIAC by J.P. Eckert and J. Mauchly

•1945: Stored program by J.V. Neuman

•1949: EDSAC by M. Wilkes

•1952: UNIVAC I and IBM 701

Eniac

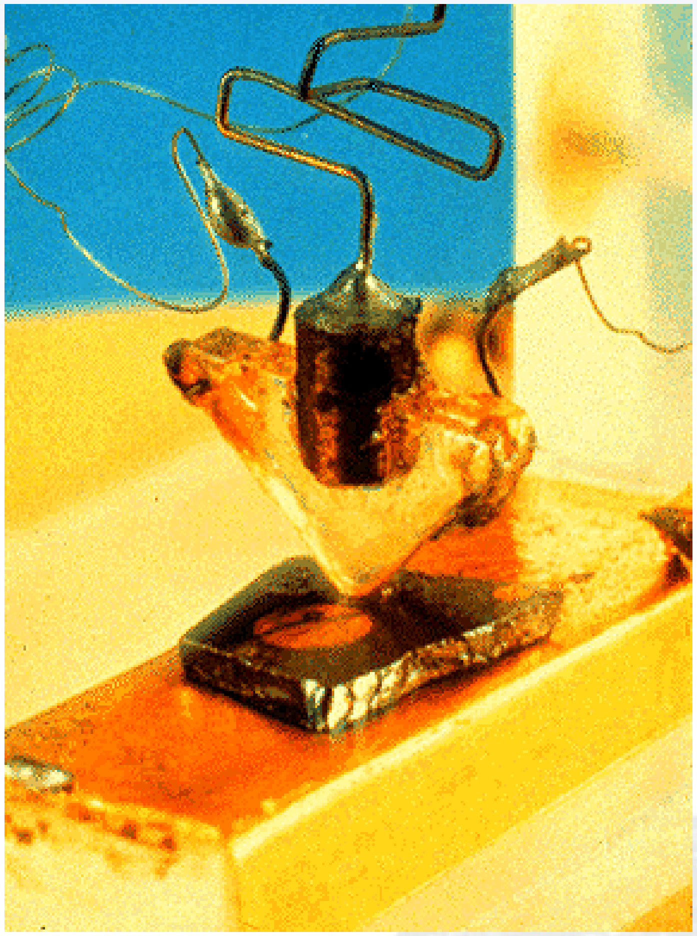


Eniac, 1946, Moore School
18000 vacuum tubes, 70000 resistors and 5 million soldered joints.
Consumed 140 Kilowatts.
It was 8 by 3 by 100 feet and weighted more than 30 tons.
It could do 5000 additions and 360 multiplications per second.

BSC - Microsoft Research Centre

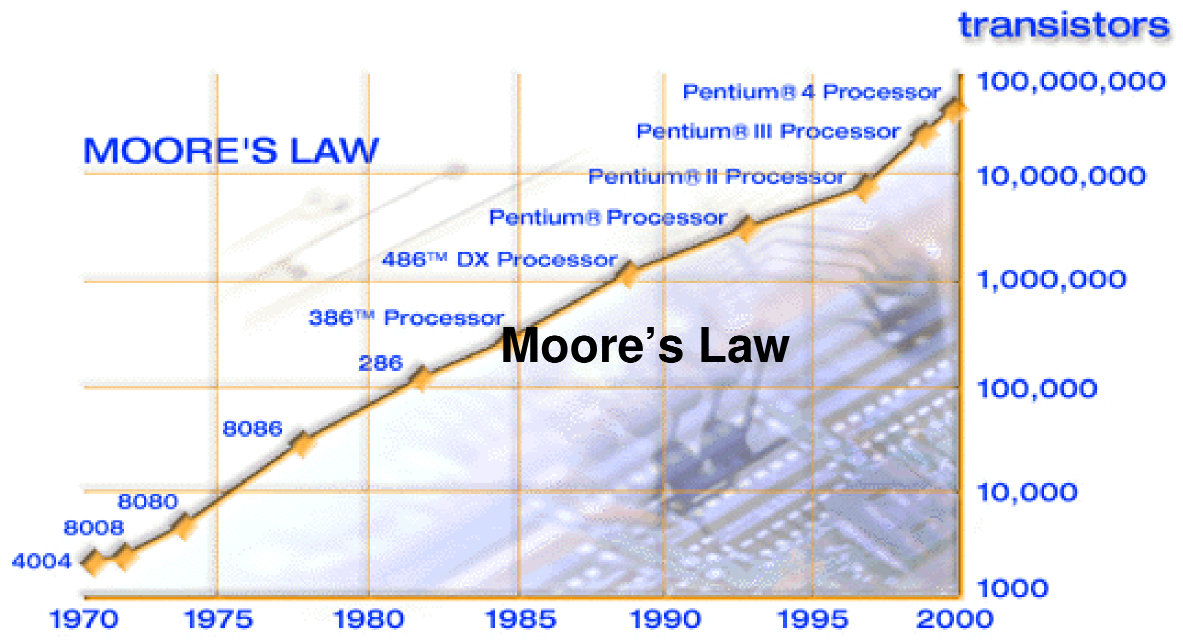
Technological Achievements

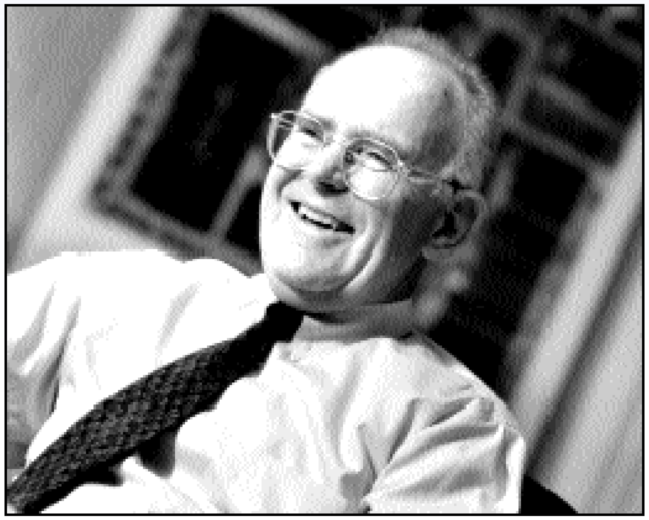
- **Transistor (Bell Labs, 1947)**
  - **DEC PDP-1 (1957)**
  - **IBM 7090 (1960)**
- **Integrated circuit (1958)**
  - **IBM System 360 (1965)**
  - **DEC PDP-8 (1965)**
- **Microprocessor (1971)**
  - **Intel 4004**



BSC Microsoft Research Centre

Technology Trends: Microprocessor Capacity





2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful. Not just processors, bandwidth, storage, etc

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

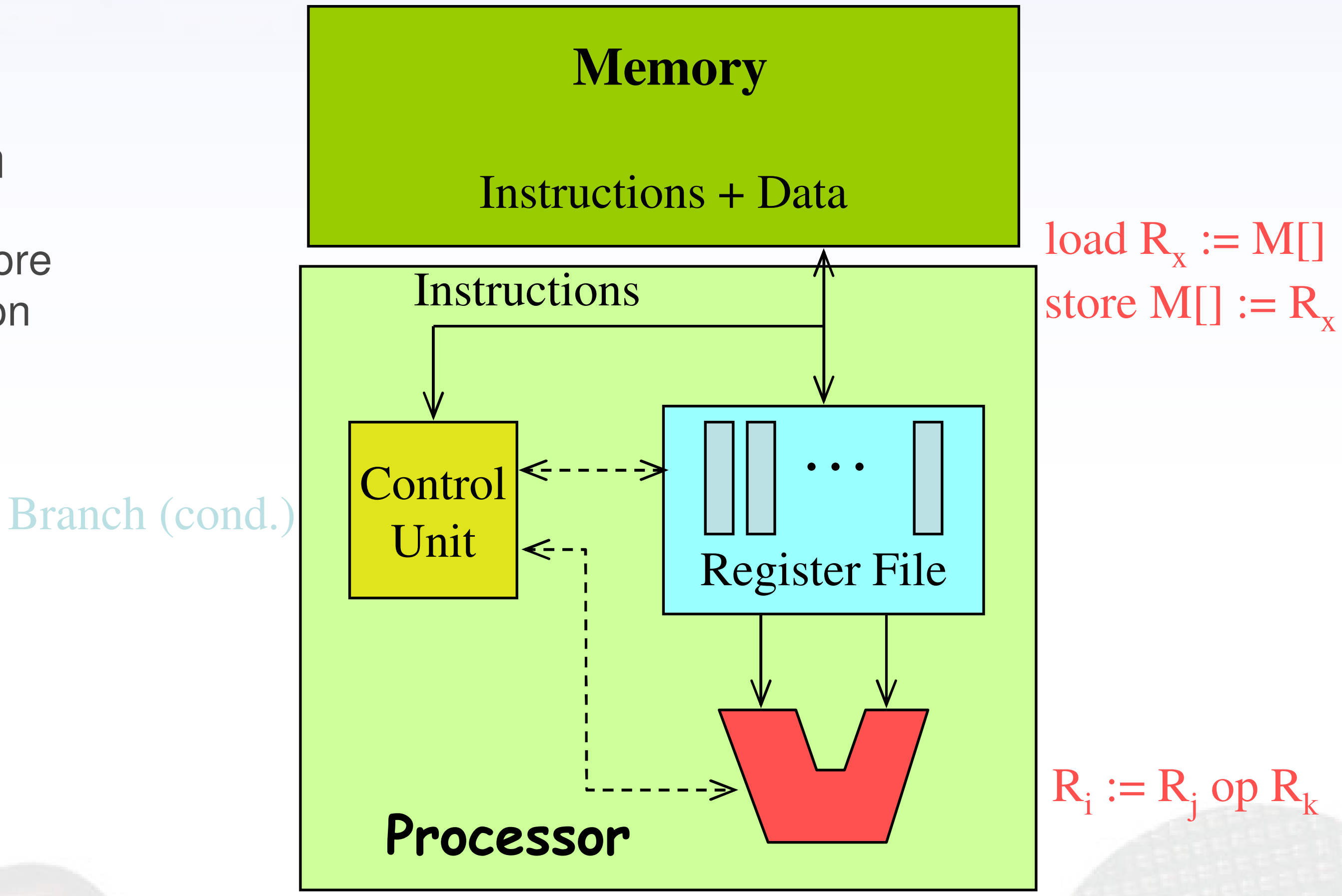
BSC Microsoft Research Centre

Computer Architecture is .....

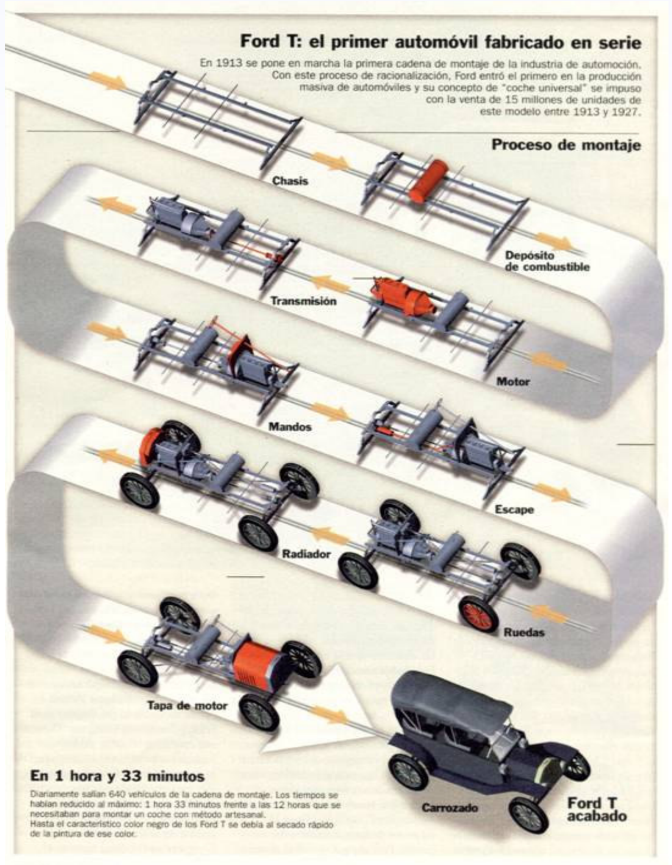
- About the interface between what technology can provide and what the people/market demand
- At that interface we have our design point:
  - Reliability
  - Availability
  - Cost
  - Power
  - Performance

Processor Organization: Basic Concepts

- Instruction types:
  - Load/Store
  - Operation
  - Control



Pipeline (H. Ford)



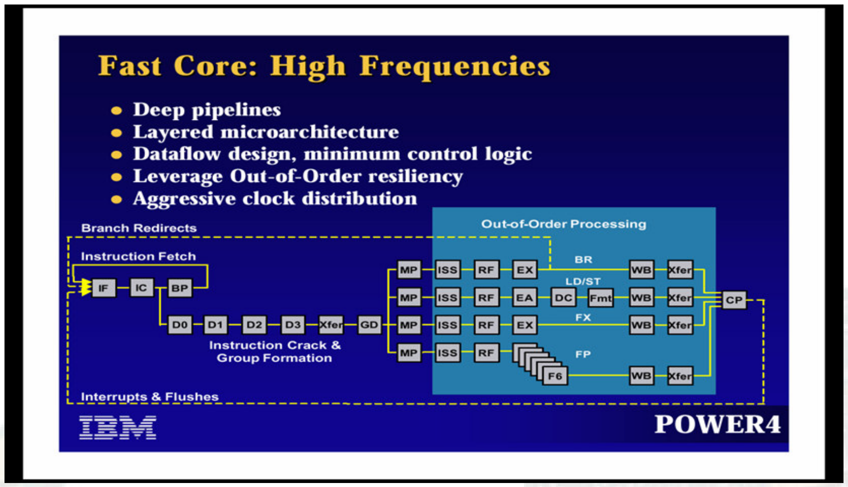
Ford T: el primer automóvil fabricado en serie

En 1913 se pone en marcha la primera cadena de montaje de la industria de automoción. Con este proceso de racionalización, Ford entró el primero en la producción masiva de automóviles y su concepto de "coche universal" se impuso con la venta de 15 millones de unidades de este modelo entre 1913 y 1927.

Proceso de montaje

En 1 hora y 33 minutos

Diariamente salían 640 vehículos de la cadena de montaje. Los tiempos se habían reducido al máximo: 1 hora 33 minutos frente a las 12 horas que se necesitaban para montar un coche con método artesanal.
Hasta el característico color negro de los Ford T se debía al secado rápido de la pintura de ese color.



Fast Core: High Frequencies

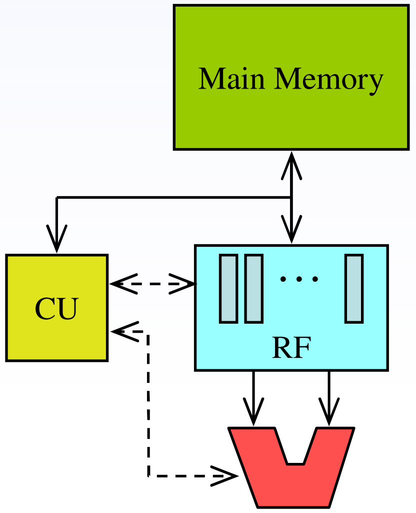
- Deep pipelines
- Layered microarchitecture
- Dataflow design, minimum control logic
- Leverage Out-of-Order resiliency
- Aggressive clock distribution

Program Dependences

- Data dependences

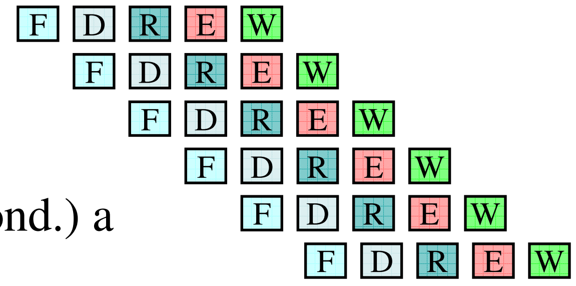
a R_i := …
b := R_i op R_j
c R_i := ...

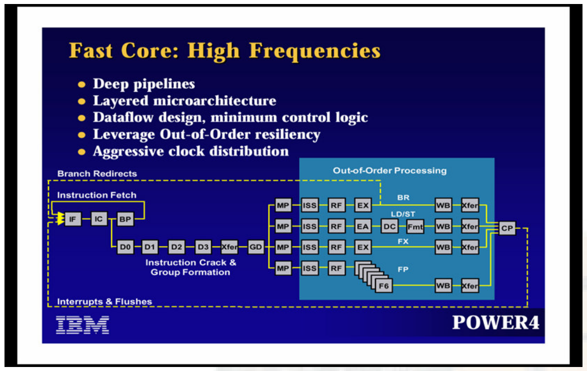
Data dependences
a and b (RAW)
Name dependences
b and c (WAR)
a and c (WAW)



- Control dependencies

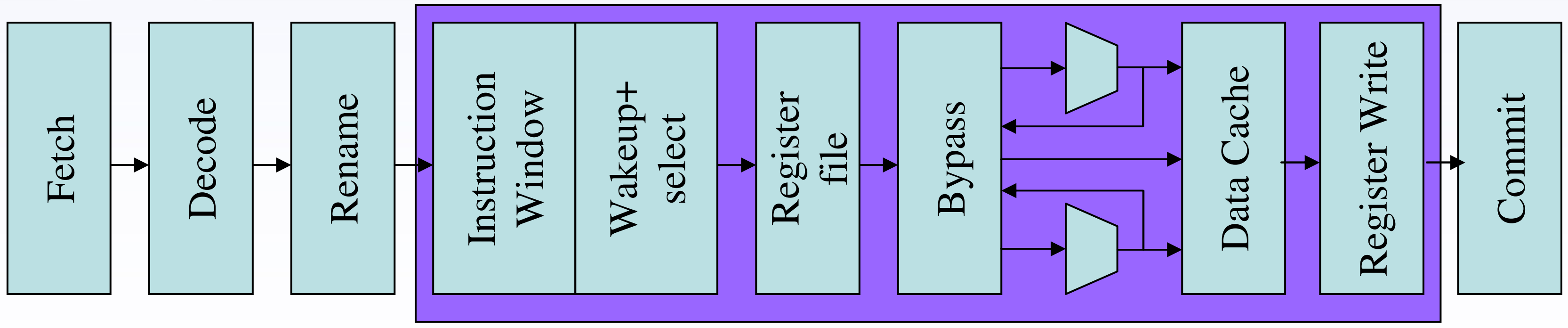
.
.
.
a
.
.
.
b branch (cond.) a
↓
b+1




Fast Core: High Frequencies
- Deep pipelines
- Layered microarchitecture
- Dataflow design, minimum control logic
- Leverage Out-of-Order resiliency
- Aggressive clock distribution

Branch Redirects
Instruction Fetch
Out-of-Order Processing
Instruction Crack & Group Formation
Interrupts & Flushes
IBM
POWER4

Superscalar Processor

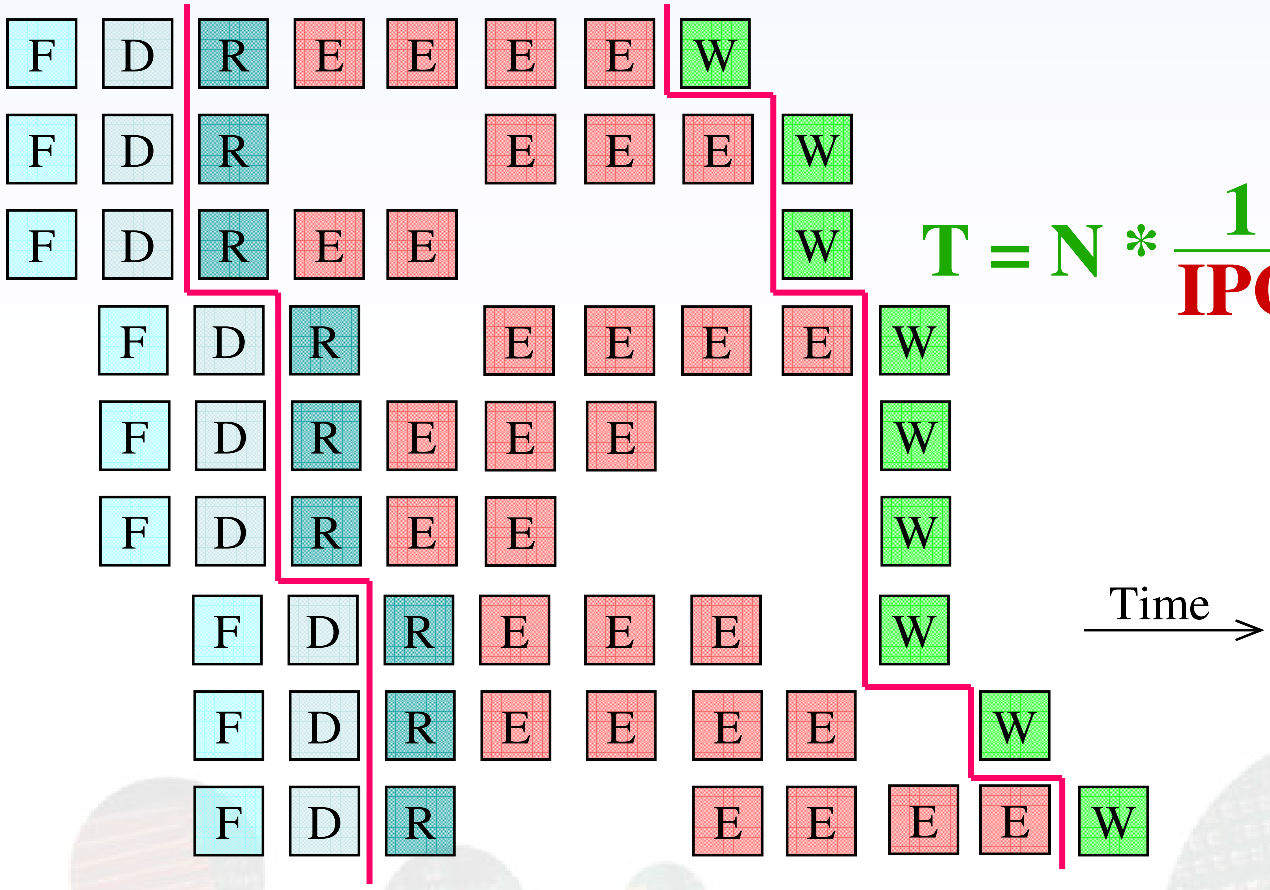


Fetch of multiple instructions every cycle.
Rename of registers to eliminate added dependencies.
Instructions wait for source operands and for functional units.
Out- of -order execution, but in order graduation.
Predict branches and speculative execution

J.E. Smith and S.Vajapeyam.¨Trace Processors…¨ IEEE Computer.Sept. 1997. pp68-74.

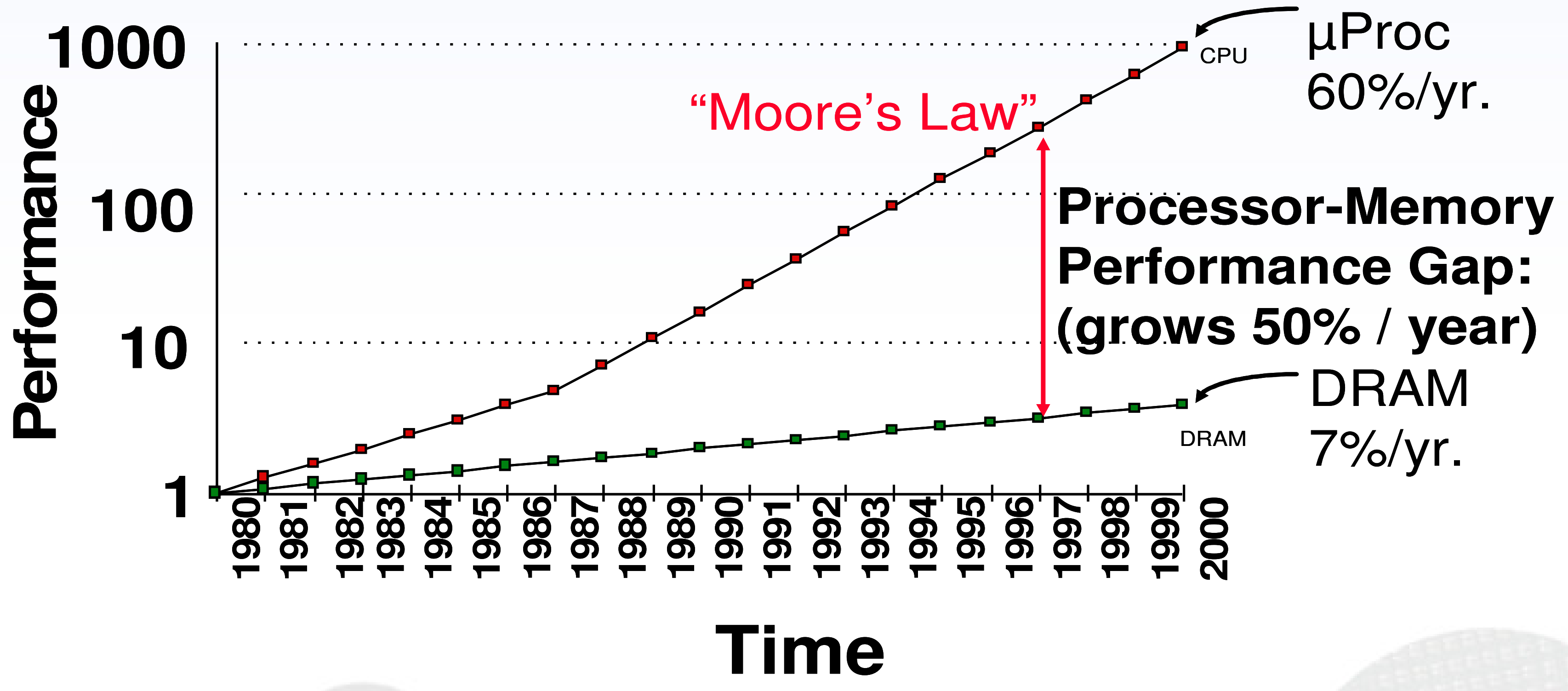
Superscalar Processors

- Out of order (IPC <= 3)



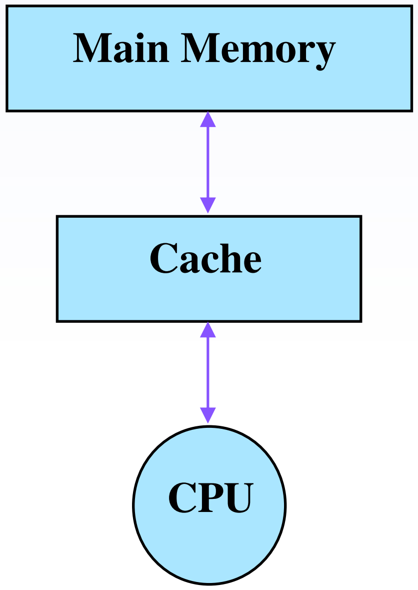
$$T = N * \frac{1}{IPC} * t_c$$

Processor-DRAM Gap (latency)



D.A. Patterson " New directions in Computer Architecture" Berkeley, June 1998

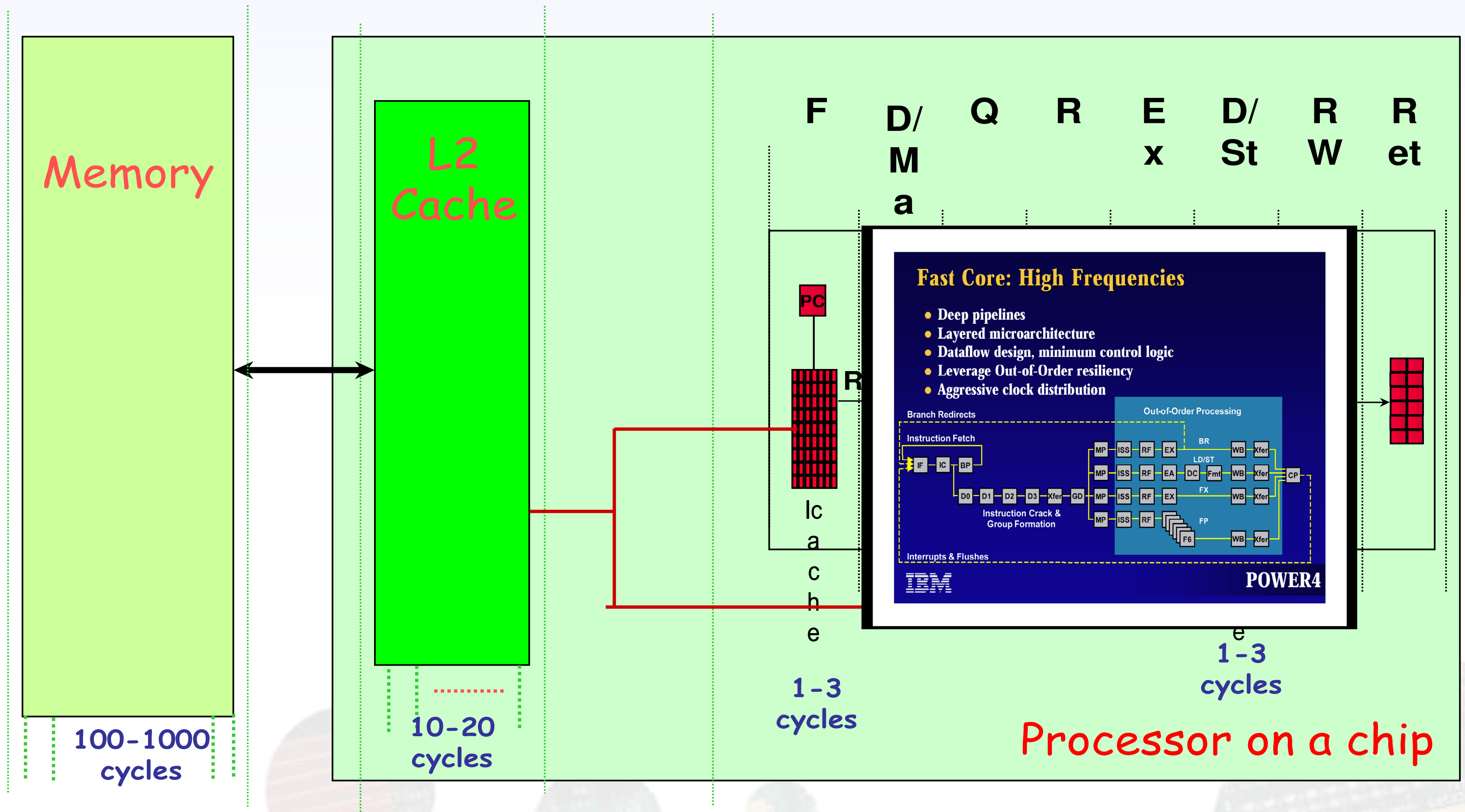
Cache Memories

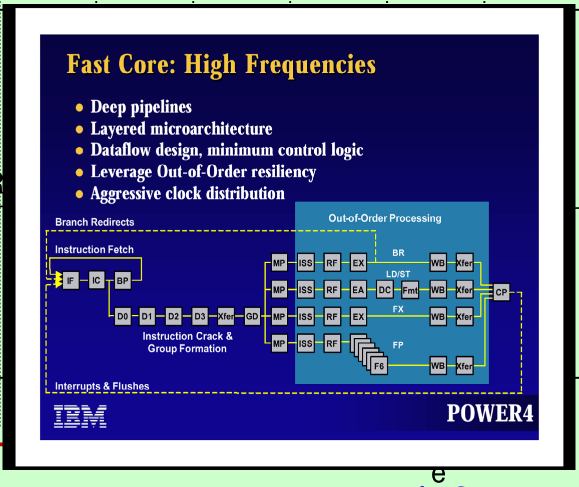


- Definition:
  - Small and fast memory between the CPU and main memory.
- Objective:
  - To reduce the access time for instructions and data.
- Feasibility:
  - Temporal and spatial locality of the programs.

The memory hierarchy has an important role towards efficiency

Latencies and Pipelines




Fast Core: High Frequencies

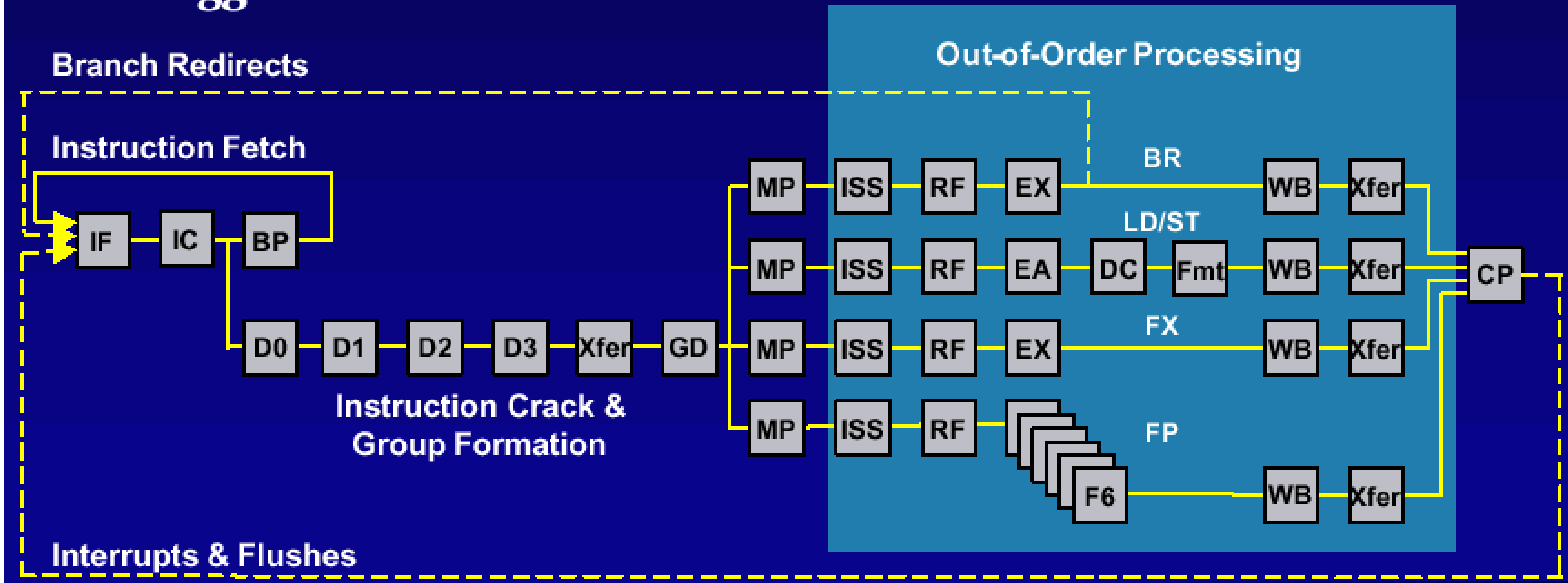
- Deep pipelines
- Layered microarchitecture
- Dataflow design, minimum control logic
- Leverage Out-of-Order resiliency
- Aggressive clock distribution

Branch Redirects
Out-of-Order Processing
Instruction Fetch
IF
IC
BP
D0
D1
D2
D3
Xfer
GD
Instruction Crack & Group Formation
MP
ISS
RF
EX
EA
DC
Fmt
F6
WB
Xfer
CP
BR
LD/ST
FX
FP
Interrupts & Flushes
IBM
POWER4

BSC Microsoft Research Centre

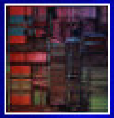
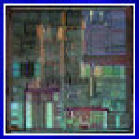
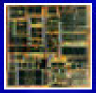
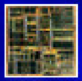
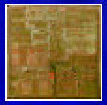
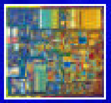
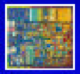
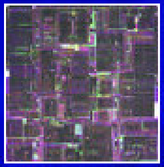
Fast Core: High Frequencies

- Deep pipelines
- Layered microarchitecture
- Dataflow design, minimum control logic
- Leverage Out-of-Order resiliency
- Aggressive clock distribution



POWER4

Processor Evolution

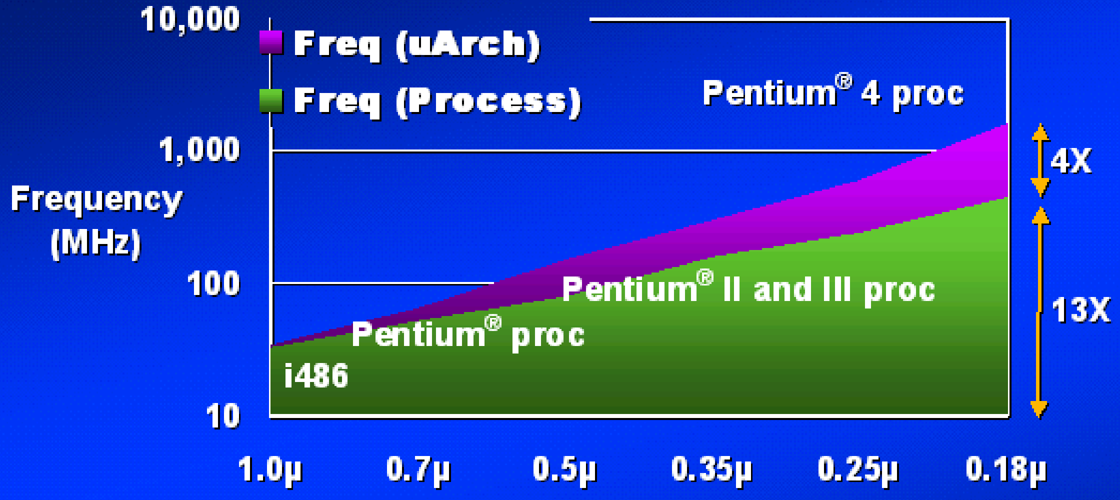
Silicon Process Technology	0.8µ	0.6µ	0.35µ	0.25µ	0.18µ	0.13µ*	Transistors (Millions)
Pentium® Processor							3.3
Pentium® Pro Processor							5.5
Pentium® II Processor							7.5
Pentium® III Processor							9.5 - 25
Pentium® 4 Processor							42^+
Itanium® Processor							40^+

* Next generation process technology, Q3 '01

Intel: Microprocessor Evolution

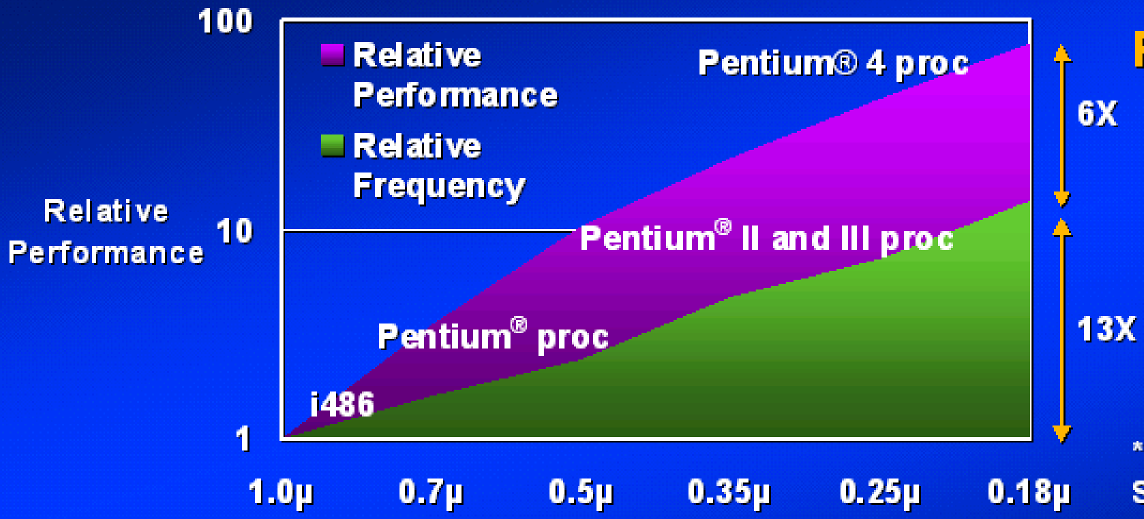
	Year/Month	Clock =1/tc.	Transistors.	Micras
I4004	1971/11	108 KHz.	2300	10
I8080	1974/04	2 MHz.	6000	6
I8086	1978/06	10 MHz.	29000	3
I80286	1982/02	12 MHz.	0.13 m.	1.50
I486DX	1989/04	25 MHz.	1.2 m.	1
Intel DX2	1992/03	100 MHz.	1.6 m	0.8
Pentium	1993/03	60 MHz.	3.1 m	0.8
Pentium Pro	1995/11	200 MHz.	5.5 m	0.35
Pentium II	1998/	450 MHz	7.5 m.	0.25
Pentium III	2000/01	1000 MHz.	28 m.	0.18
P4	2000/09	1400 MHz.	42 m.	0.18

Frequency and Performance Advances



Frequency Increased 50X

- 13X due to process technology
- Additional 4X due to microarchitecture

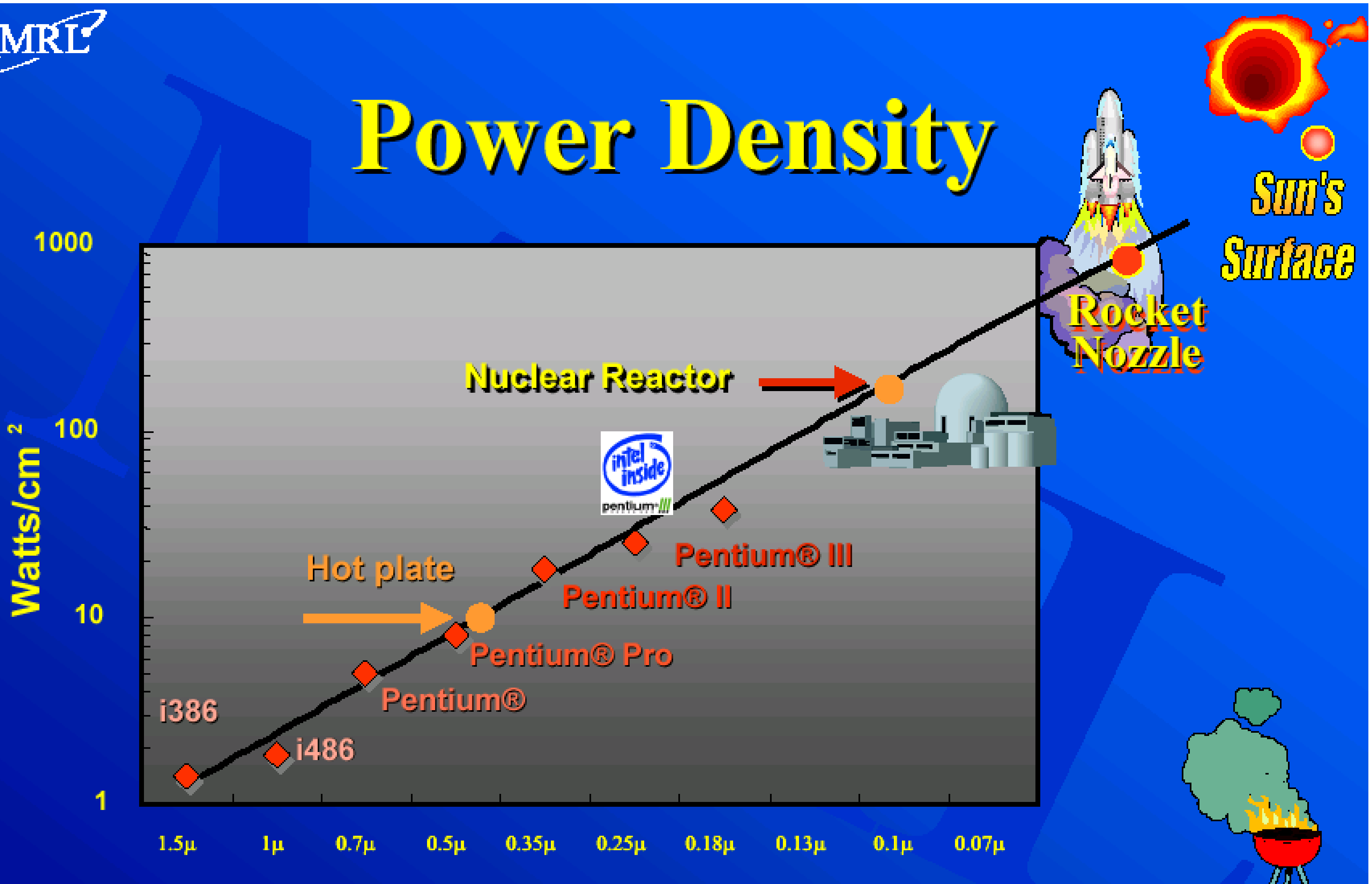


Performance Increased >75X

- 13X due to frequency
- Additional >6X due to microarchitecture and design

*Note: Performance measured using SpecINT and SpecFP

Power Density



* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

Process Scaling

- Power = $^1/_2\ C\ V^2\ F$
- On each silicon process step (every 2 yrs)
  - Capacitance decreases: 0.7x
  - Supply voltage decreases: 0.9x
  - Frequency increases: 1.4x
  - Area improves: 0.5x
  - Power: $0.7 * 0.9^2 * 1.4 = 0.8$x
    - *for the same number of transistors*
  - 2x transistors => 0.8 * 2 = 1.6x power

Power is increasing at the rate of 1.6x every 2 years

BSC - Microsoft Research Centre

Technology Outlook

High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Technology Node (nm)	90	65	45	32	22	16	11	8
Integration Capacity (BT)	2	4	8	16	32	64	128	256
Delay = CV/I scaling	0.7	~0.7	>0.7	Delay scaling will slow down				
Energy/Logic Op scaling	>0.35	>0.5	>0.5	Energy scaling will slow down				
Bulk Planar CMOS	High Probability					Low Probability		
Alternate, 3G etc	Low Probability					High Probability		
Variability	Medium			High		Very High		
ILD (K)	~3	<3	Reduce slowly towards 2-2.5					
RC Delay	1	1	1	1	1	1	1	1
Metal Layers	6-7	7-8	8-9	0.5 to 1 layer per generation				

Shekhar Borkar, Micro37, P

BSC Microsoft Research Centre

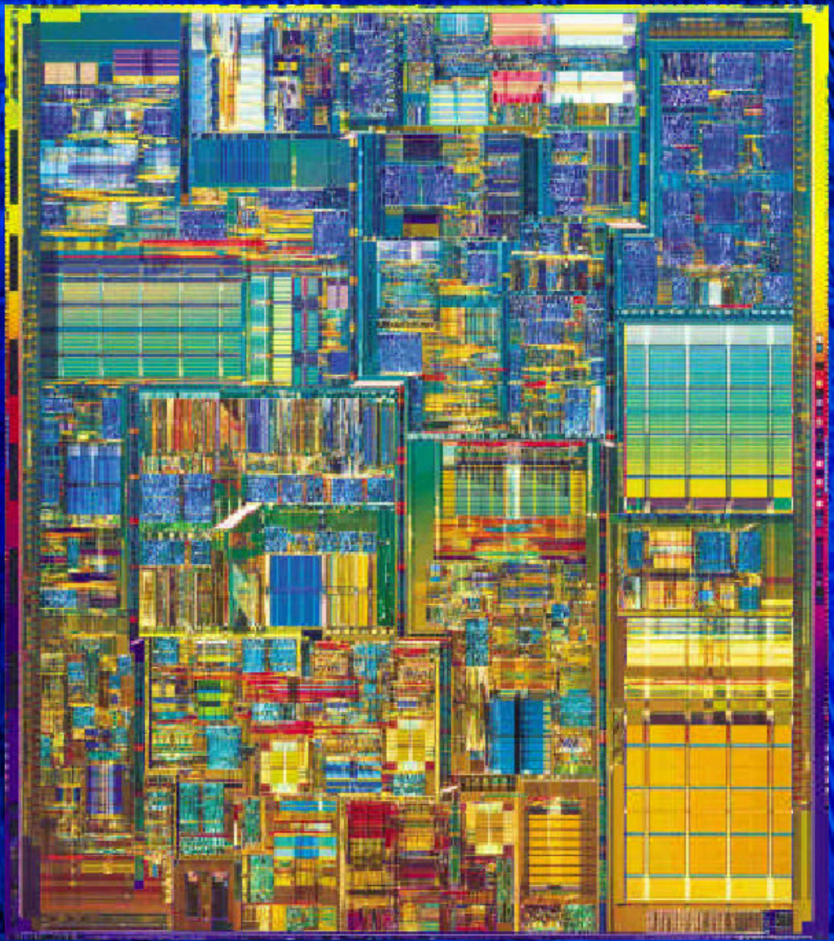
2001 – Pentium® 4 Processor

Introduced November 20, 2000

@1.5 GHz core, 400 MT/s bus
42 Million 0.18µ transistors

August 27, 2001

@2 GHz, 400 MT/s bus
640 SPECint_base2000*
704 SPECfp_base2000*



Source: http://www.specbench.org/cpu2000/results/

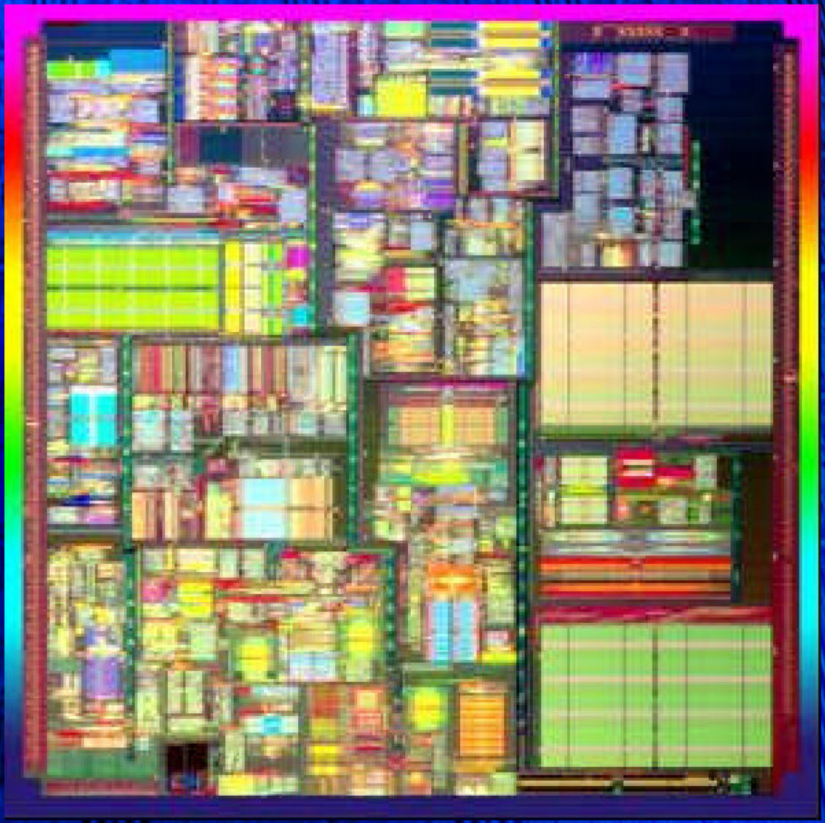
2002 – Pentium® 4 Processor

November 14, 2002

@3.06 GHz, 533 MT/s bus

1099 SPECint_base2000*
1077 SPECfp_base2000*

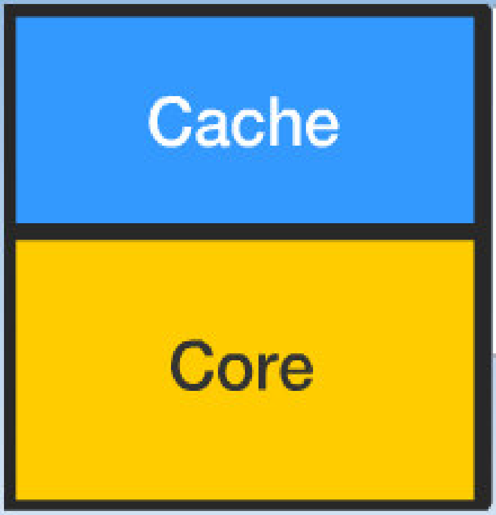
55 Million 130 nm process



Source: http://www.specbench.org/cpu2000/results/

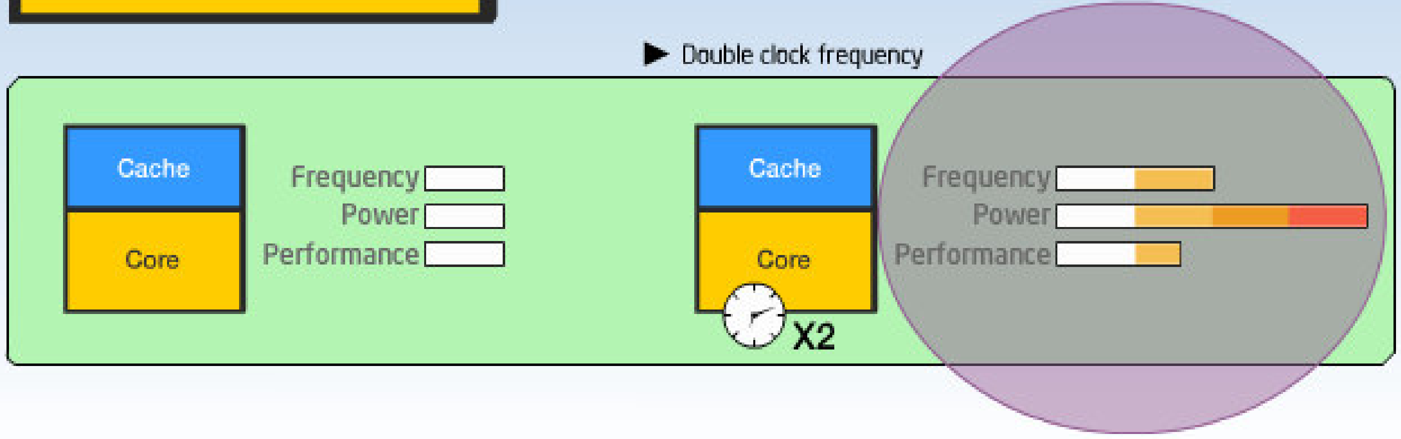
What processors looked like in the past

Past



In the past, chip designers improved performance by increasing clock frequencies, using techniques such as out-of-order execution and pipelining. This method now requires too much power.

Optimized for speed



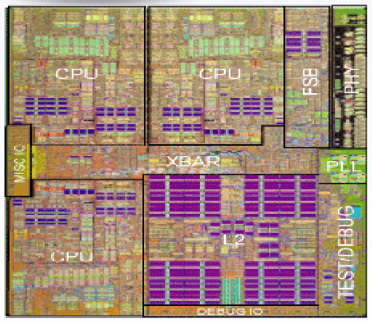
Play Back Forward
Restart Auto Play Loop Demo

Right-hand Turn

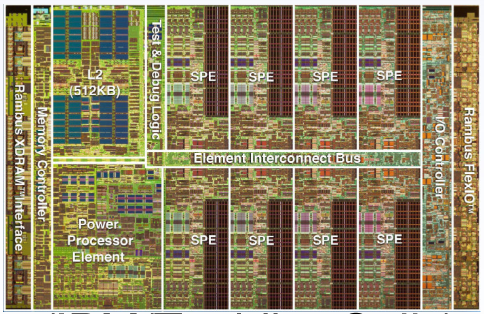
- Moore´s law enables doubling of transistors on chip every 18 months, increasing clock speed as well. However
  - Increase of clock speed is slowing down
  - Diminishing performance gain per unit area for single core design
  - Increase performance by replicating cores
  - Doubling the number of cores on chip ever 18 months, maybe a new law?
- Why should we care?
  - Power density
  - Additional transistors just waste Watts
- Enter chip multiprocessors
  - No more increase in single-core performance!

BSC~Microsoft Research Centre

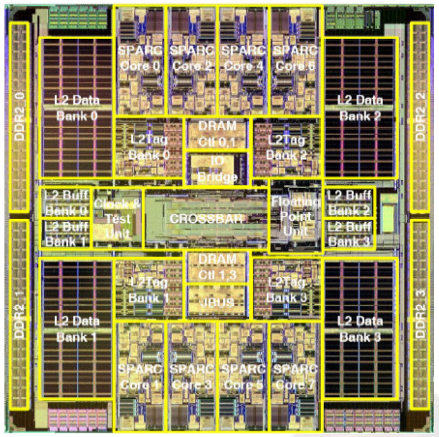
Some Examples of CMP



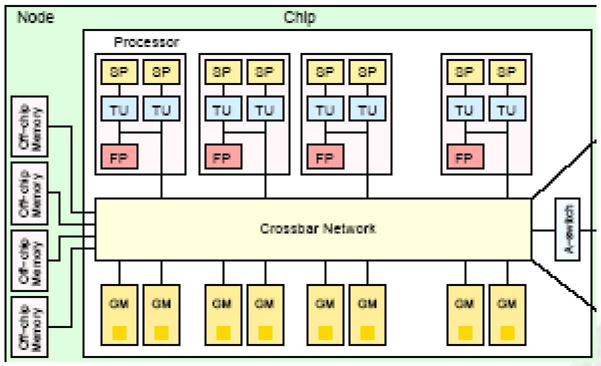
Microsoft/IBM Xbox360 (3 cores)



Sony/IBM/Toshiba Cell (9 cores)



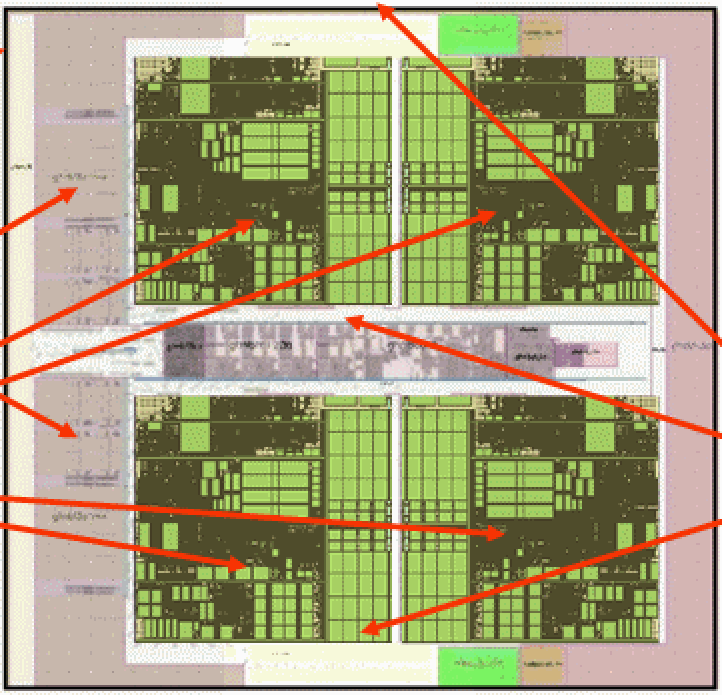
Sun Niagara (8 cores)



IBM Cyclops64 (80 cores, in development)

BSC Microsoft Research Centre

AMD's Next Generation Processor Technology



Native quad core die

Expandable shared L3 cache

IPC enhanced CPU cores

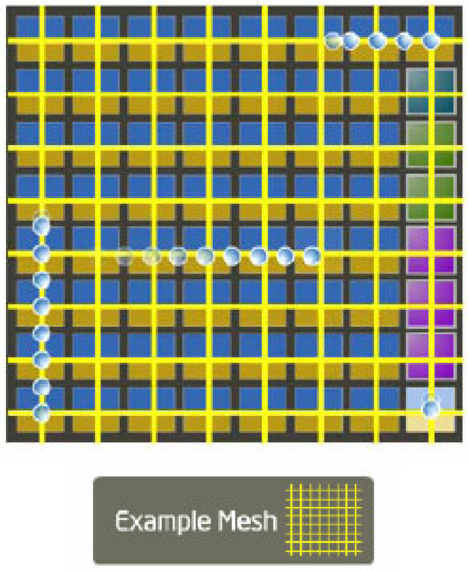
- 32B instruction fetch
- Improved branch prediction
- Out-of-order load execution
- Up to 4 DP FLOPS/cycle
- Dual 128-bit SSE dataflow
- Dual 128-bit loads per cycle
- Bit Manipulation extensions (LZCNT/POPCNT)
- SSE extensions (EXTRQ/INSERTQ, MOVNTSD/MOVNTSS)

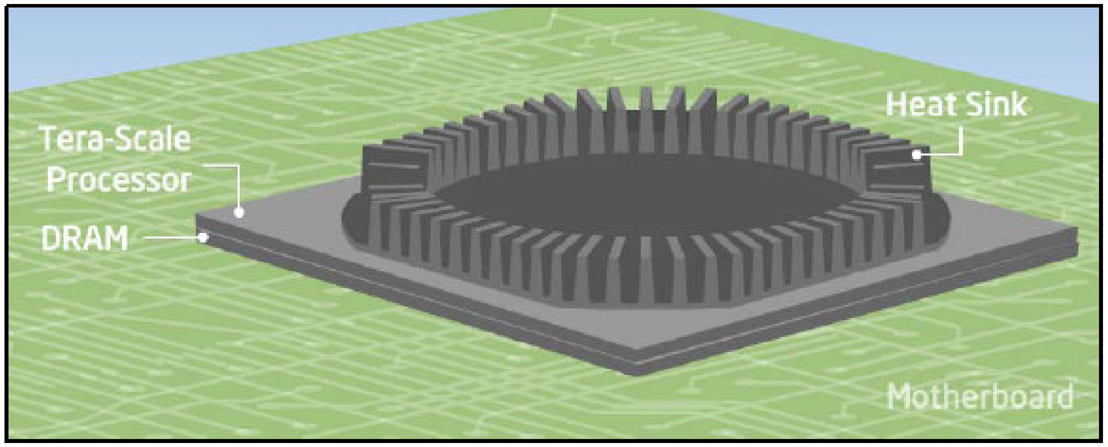
Ideal for 65nm SOI and beyond

Enhanced Direct Connect Architecture and Northbridge

- Four ungangable x16 HyperTransport^{TM} links (up to 5.2GT/sec)
- Enhanced crossbar
- Next-generation memory support
- FBDIMM *when appropriate*
- Enhanced power management and RAS

Intel´s Petaflop chip





The key technologies of this first Tera-scale Research Prototype are a mesh interconnect (left) and support for 3D stacked memory (above).

- 80 processors in a die of 300 square mm.
- Terabytes per second of memory bandwidth
- Note: The barrier of the Teraflops was obtained by Intel in 1991 using 10.000 Pentium Pro processors contained in more than 85 cabinets occupying 200 square meters ☺
- This will be possible soon

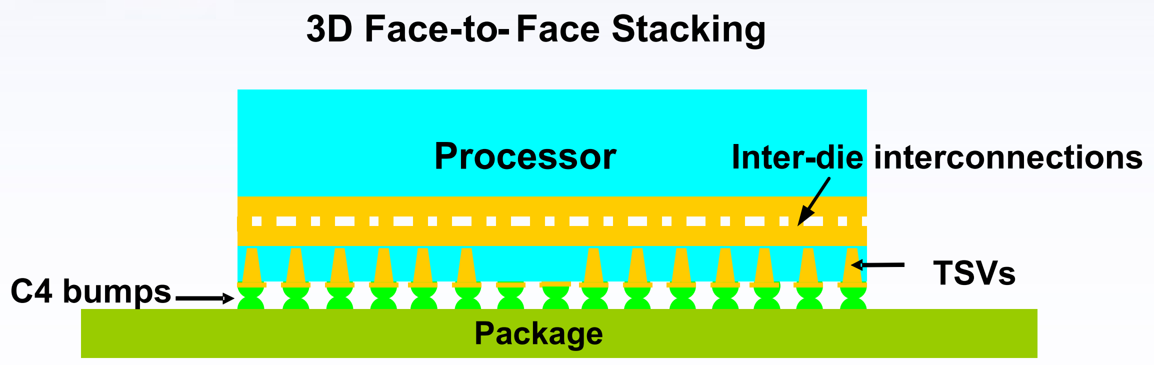
Intel 80-core chip

- First many-core silicon prototype
  - 80 special purpose processor cores configured as a 2D mesh (8 x 10)
- Tapeout Q2'06
  - Tiled processor architecture
  - Scalable on-die interconnect fabric
  - Memory bandwidth – 3D stacking
  - Dynamic power management

Intel 80-core chip

- Array of 80 tiles
  - Each tile has a compute element and router - reused from earlier projects
  - Tiling simplifies the implementation
  - Total die size: 300mm2
  - Transistor count: **100M**
  - Frequency: **5 GHz**
- Power efficiency
  - Achieves **8 GFLOPS/W** @ Teraflop performance
- Mapped applications
  - LINPACK routines using dedicated ISA

Intel 80-core chip



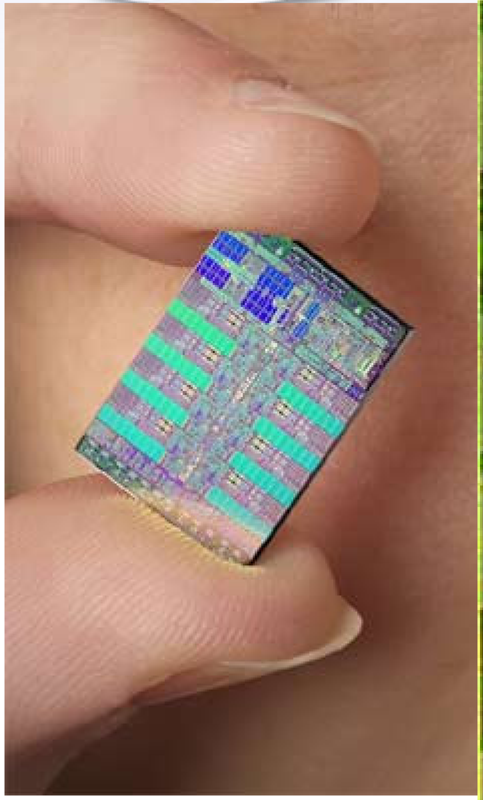
Technology

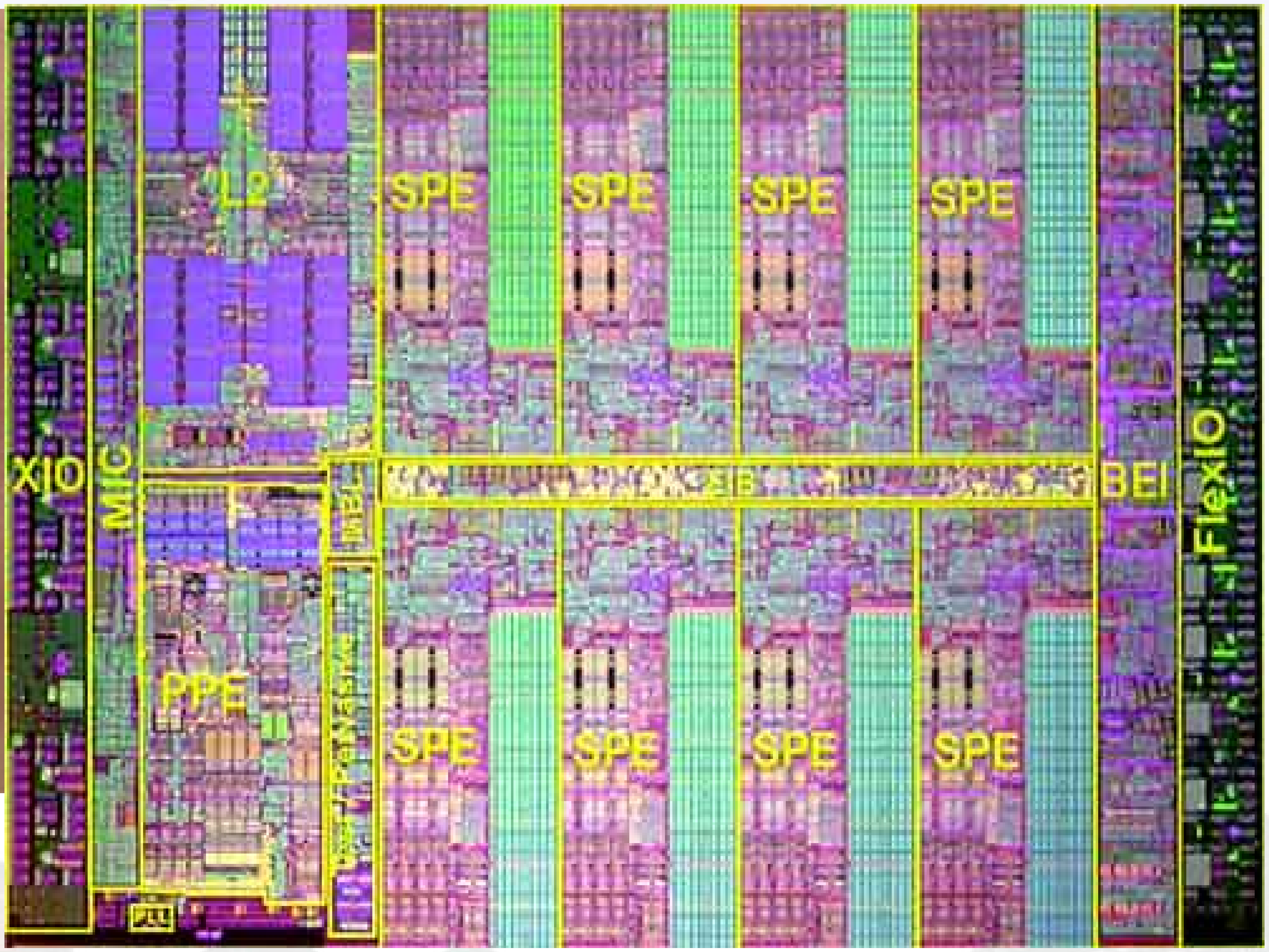
- Prototype SRAM with **face-to-face 3D die** stacking
- Memory bit density: 210 KBytes/tile
- 80 tiles in 13.75x22mm for 16 MB total

Bandwidth

- 40 GB/s/tile at 5 GHz, full duplex
- Aggregate **3.2 TB/s**

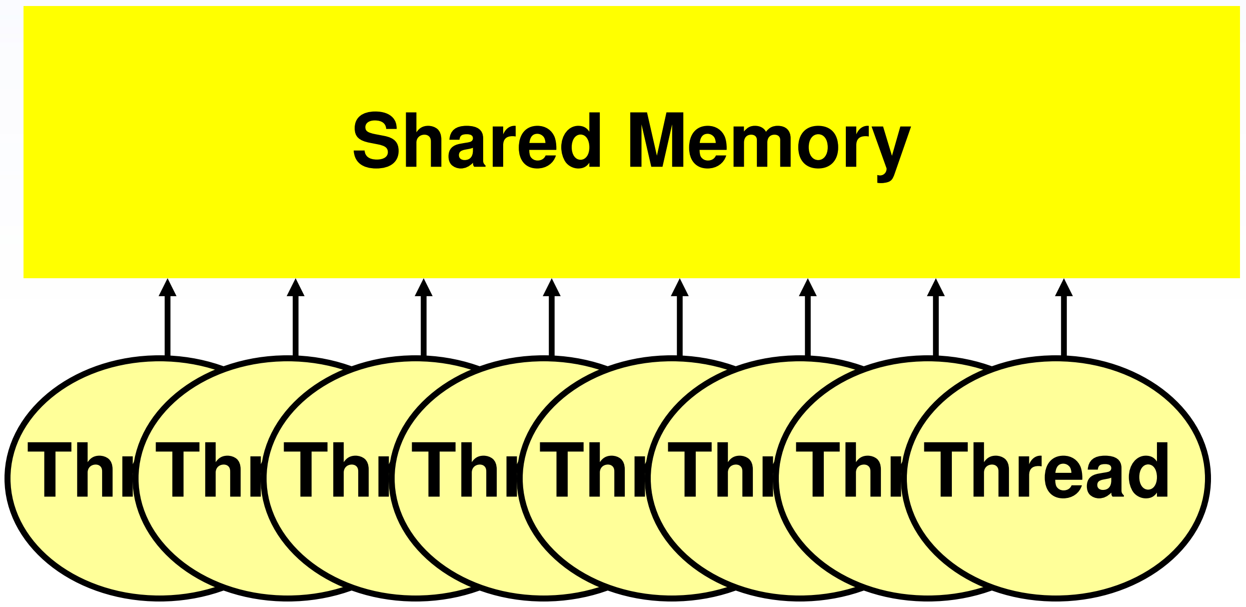
Cell processor architecture



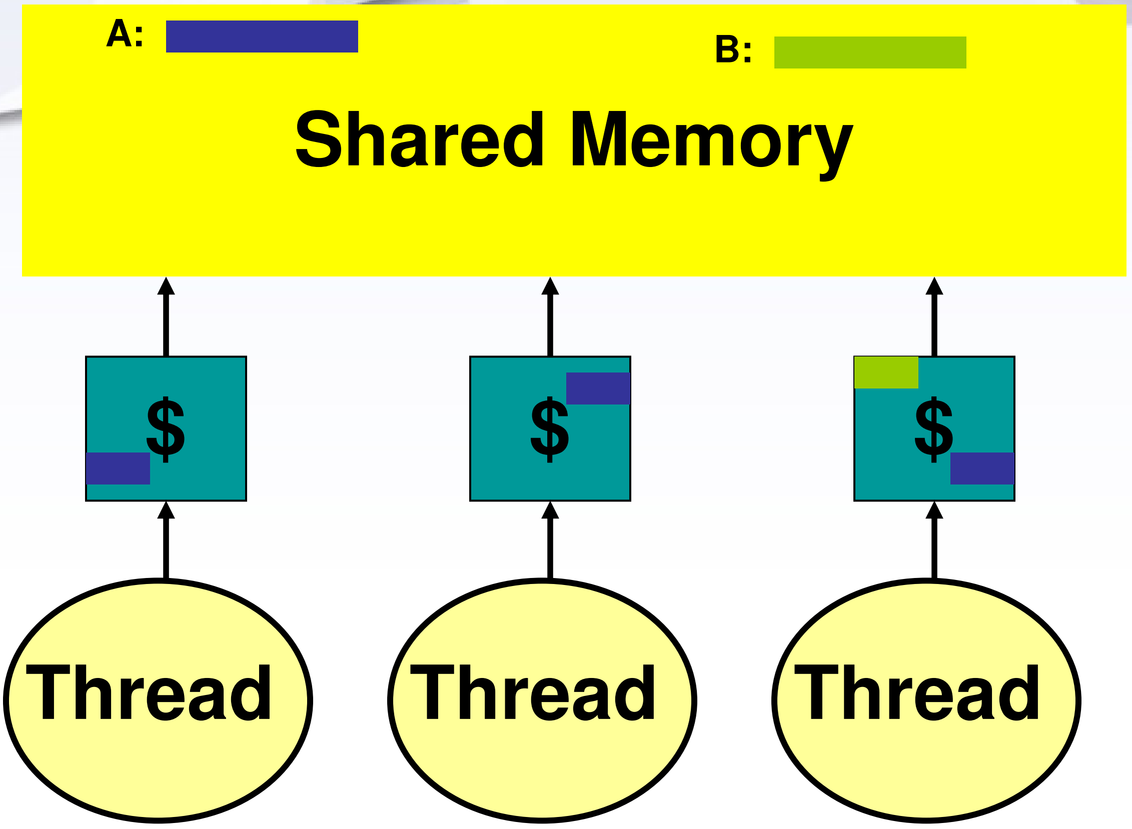


235 Mtransistors
235 mm^2

Programming Model:



Caches:
Automatic Migration and Replication of Data

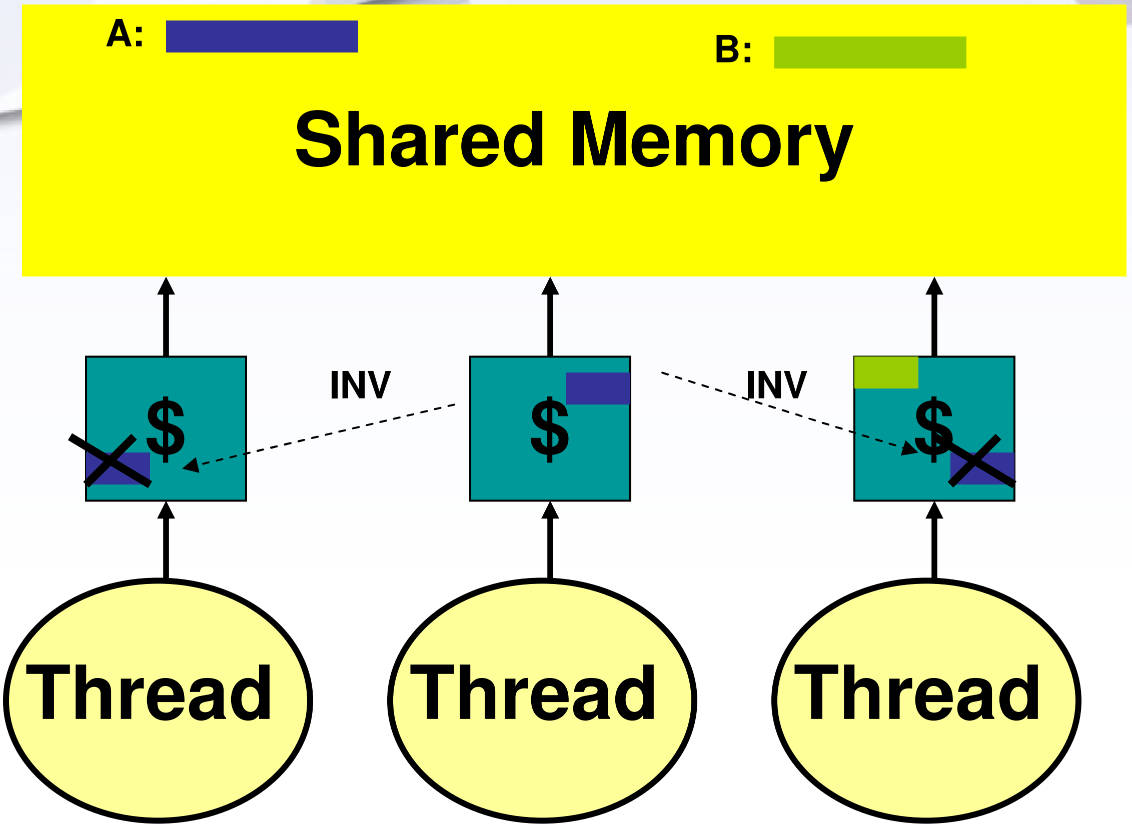


Read A	...	Read B
Read A	Read A	...
...	...	Read A
...		
Read A		

Erik Hagersten, ACACES-2006

BSC Microsoft Research Centre

The Cache Coherent Memory System

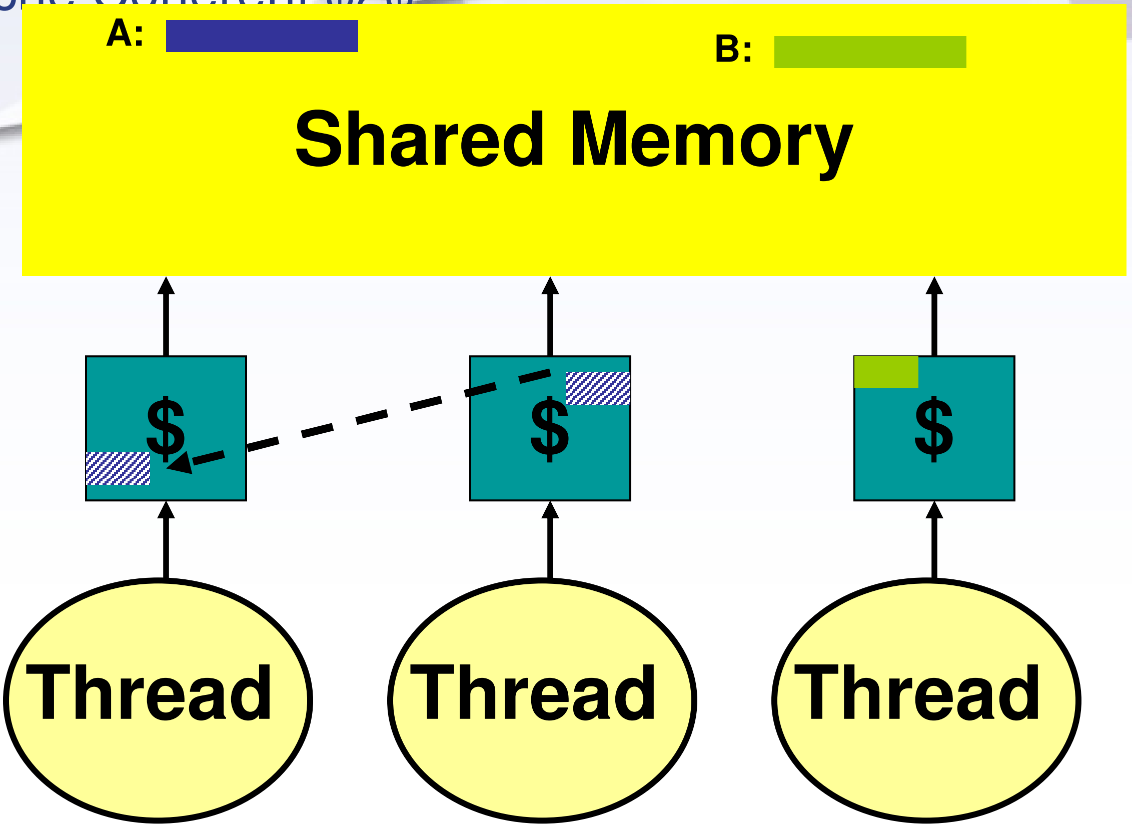


Read A	...	Read B
Read A	Read A	...
...	...	Read A
...	Write A	

Erik Hagersten, ACACES-2006

BSC Microsoft Research Centre

The Cache Coherent 2



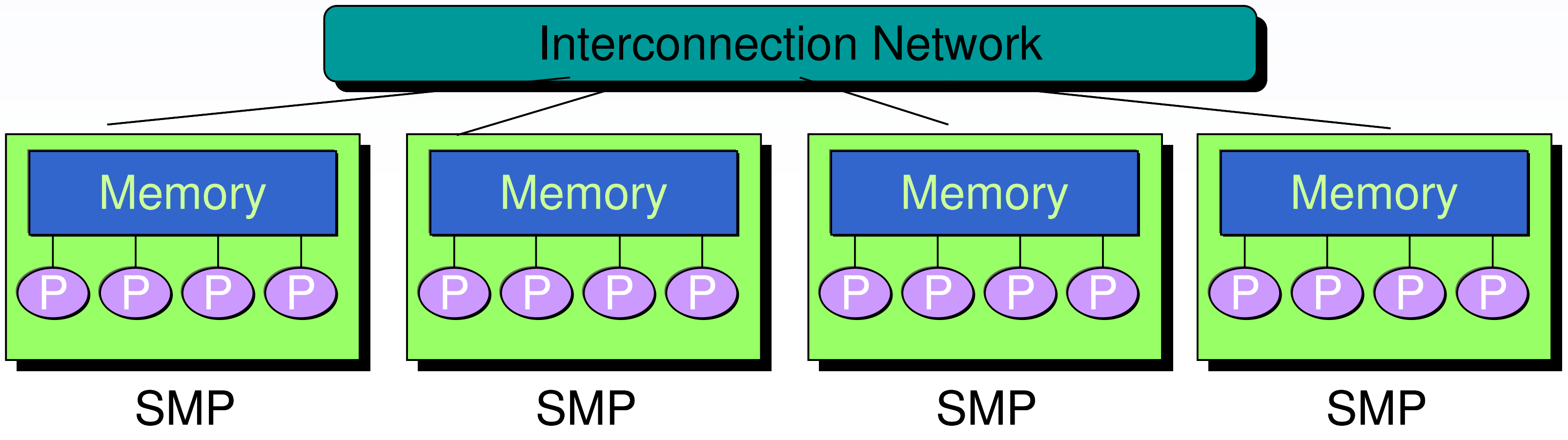
Read A	...	Read B
Read A	Read A	...
...	...	Read A
...	Write A	
Read A		

Erik Hagersten, ACACES-2006

BSC Microsoft Research Centre

Hybrid SMP-cluster parallel systems

- Most modern high-performance computing systems are clusters of SMP nodes (performance/cost trade-off)



- Programming models allow to specify:
  - How computation is distributed?
  - How data is distributed and how is it accessed?
  - How to avoid data races?

Increasing Complexity of Programming

Serial Code

```
   real*4 X(400)
   do 10 i=1,400
     X(i)=i
10  continue
   S=0
   do 30 i=1,400
   S=S+X(i)
30  continue
   write(6,*) '1+...+400=',S
   stop
   end
```

MPI

```
parameter(n=400,np=4)
   parameter(masterpid=0)
    real*4 X(400)
    integer to_p,from_p,tag,mypid,pnum
    call MPI_init(4)
    call MPI_comm_rank(MPI_COMM_WORLD,mypid)
    call MPI_comm_size(MPI_COMM_WORLD,pnum)
    if(mypid.eq.masterpid) then
      do 10 i=1,400
        X(i)=i
10      continue
      do 20 to_p=1,3
        tag=0
        call MPI_send(X(100*to_p+1),100,MPI_REAL,to_p,tag,MPI_COMM_WORLD)
20      continue
    else
      from_p=0
      tag=0
      call MPI_recv(X(1),100,MPI_real,from_p,tag,MPI_COMM_WORLD,idummy)
    endif
    S=0
    do 30 i=1,100
      S=S+X(i)
30    continue
    if(mypid.ne.masterpid) then
      to_p=0
      tag=1
      call MPI_send(S,1,MPI_REAL,to_p,tag,MPI_COMM_WORLD)
    else
      do 40 from_p=1,3
        tag=1
        call MPI_recv(SS,1,MPI_REAL,from_p,tag,MPI_COMM_WORLD,idummy)
        S=S+SS
40      continue
      write(6,*)'1+..+400=',S
    endif
    call MPI_barrier(MPI_COMM_WORLD)
    call MPI_finalize
    stop
    end
```

BSC - Microsoft Research Centre

A new wall is on the horizon

- Programmer productivity problem
- How to program 100s of cores on chip efficiently?
- If not prepared today, we will hit a productivity wall (we were unprepared and hit another wall, *power density,* in single cores)
- All those cores will only make sense if they can be used efficiently
  - Intel, AMD, Microsoft, … are more concerned than you think
  - This is a big gamble!!!

- Lock-based programming is highly problematic
- Transactional Memory is a promising solution
  - Context: Shared memory CMPs

BSC Microsoft Research Centre

Top-down Architecture

- Top-down architecture, include:
  - Application
  - Debugging
  - Programming models
  - Programming languages
  - Compilers
  - Operating Systems
  - Runtime environment

As design drivers

What is Transactional Memory (TM) ?

- TM *optimistically* runs transactions in parallel, in the hope that they do not perform conflicting memory accesses.
- If the transactions do not conflict then the optimism has paid off.
- If transactions do attempt conflicting accesses, then the TM must delay or abort the work of one or the other.
- The partial effects of aborted transactions are "rolled back" before they are re-executed .

Why is TM attractive?

- Consider a concurrent FIFO implementation.
  - One thread can enqueue items at the tail of the queue while at the same time
  - Another thread can dequeue items from the head of the queue
- Simple problem, right?
- Solving this problem with locks efficiently is quite difficult (Michael and Scott 1996)
- Solutions to such simple problems with fine-grained locking are considered *difficult enough to be publishable results!*

Why is TM attractive? (cont.)

```
class Queue {
  QNode head;
  QNode tail;
  public enq(Object x) {
    atomic {
      QNode q = new QNode(x);
      q.next = head;
      head = q;
    }
  }
  ...
}
```

- **Implementing a concurrent FIFO using TM is trivial**

BSC Microsoft Research Centre

Transaction Execution

- The TM system lets different threads to execute the atomic regions speculatively.
- The TM system guarantees:
  - *Atomicity* – all tentative memory changes become visible to the other threads simultaneously at the time when a transaction commits.
  - *Isolation* – while transaction executes the tentative memory updates are not visible by the other threads.

Transactions vs. Locks

Transactions

- Non-blocking synchronization
- Deadlock free
- Composable
- Easy programmable
- Efficiency of fine grain locks

Locks

- Blocking synchronization
- Deadlock risk
- Non-composable
- Coarse grain locks limit TLP
- Fine grain locks are difficult to program

B1

Slide 47

B1 Osman added
BSC-CNS, 9/2/2007

Hardware Transactional Memory

- Implementation on top of *caches* and *coherency protocol*.
- Examples: TCC [ISCA2004], LogTM [HPCA2006]
- Advantages:
  - very fast
  - strong isolation
- Disadvantages:
  - limited in time (context switch, page fault)
  - limited in space (overflows)
  - Inflexible (static management)

Software Transactional Memory

- Implemented as a library
- Examples: TL2, Nebelung, RTM, DSTM
- Advantages:
  - flexible (conflict management)
  - unlimited in time and space
- Disadvantages:
  - very slow
  - difficult to program without compiler support
  - strong isolation is very expensive

B2

B2 If you have time, have souces for all of those and the HyTM ones (e.g. Nebelung (Iteract 2007)

BSC-CNS, 9/2/2007

Hybrid Transactional Memory

- Attempts to compensate the disadvantages of both HTM and STM
- HTM
  - virtualizes HTM in time and space
  - examples: HyTM, VTM, PTM
- STM
  - accelerates the slow and frequent STM operations in hardware
  - examples: HASTM, RHTM, SigTM

Versioning and Conflict resolution

- Conflicts happen if
  - One transaction (attemps to) reads a data item while another one tries to write to the item
  - At least two transactions (attempt to) write a data item

- If conflict is detected one of the transactions could be aborted

- Basic implementation requirements
  - Data versioning
  - Conflict detection & resolution

Versioning

- Manage uncommited(new) and commited(old) versions of data for concurrent transactions

1. Eager (undo-log based)
    - •Update memory location directly; maintain undo info in a log
    - +Faster commit, direct reads (SW)
    - –Slower aborts, no fault tolerance, weak atomicity (SW)

2. Lazy (write-buffer based)
    - •Buffer writes until commit; update memory location on commit
    - +Faster abort, fault tolerance, strong atomicity (SW)
    - –Slower commits, indirect reads (SW)

BSC Microsoft Research Centre

Conflict Detection and Resolution

- Detect and handle conflicts between transaction
  - Read-Write and (often) Write-Write conflicts
  - For detection, a transactions tracks its read-set and write-set

1. Eager detection
   - Check for conflicts during loads or stores
     - HW: check through coherence lookups
     - SW: checks through locks and/or version numbers
   - Use contention manager to decide to stall or abort
     - Various priority policies to handle common case fast

2. Lazy detection
   - Detect conflicts when a transaction attempts to commit
     - HW: write-set of committing transaction compared to read-set of others
       - Committing transaction succeeds; others may abort
     - SW: validate write-set and read-set using locks and version numbers

Readset / Writeset

- Readset: The set of all the distict memory locations read by a transaction
- Writeset: The set of all the distinct memory locations written by a transaction