ISA Implementations Partly in Software

Run programs for one ISA on hardware with different ISA Techniques:

- **Emulation**
  - **– OS software interprets instructions at run-time**
  - **– E.g., OS for PowerPC Macs had emulator for 68000 code**

- **Binary Translation**
  - **– convert at install and/or load time**
  - **– IBM AS/400 to modified PowerPC cores**
  - **– DEC tools for VAX->MIPS->Alpha**
- **Dynamic Translation (or Dynamic Compilation)**
  - **– compile non-native ISA to native ISA at run time**
  - **– Sun's HotSpot Java JIT (just-in-time) compiler**
  - **– Transmeta Crusoe, x86->VLIW code morphing**

- **Run-time Hardware Emulation**
  - **– Hardware supports two ISAs!**
  - **– IBM 360 had IBM 1401 emulator in microcode**
  - **– Intel Itanium converts x86 to native VLIW (two software-visible ISAs)**

Dynamic Translation

- Translate code sequences as needed at run-time, but cache results.
- Can optimize code sequences based on dynamic information (e.g., branch targets encountered).

- Tradeoff between optimizer run-time and time saved by optimizations in translated code
- Technique used in Java JIT compilers
- Also, Transmeta Crusoe for x86 emulation

Transmeta Crusoe

- Converts x86 ISA into internal native VLIW format using software at run-time ->Code Morphing"
- Translations cached to avoid translator overhead on repeated execution
- Completely invisible to operating system – looks like x86 hardware processor

Transmeta Translation

x86 code:

```
addl %eax, (%esp)  # load data from stack,
                   add to eax
addl %ebx, (%esp)  # load data from stack,
                   add to ebx
movl %esi, (%ebp)  # load esi from memory
subl %ecx, 5       # sub 5 from ecx
```

first step, translate into RISC ops:

```
ld %r30, [%esp]              # load from stack into temp
add.c %eax, %eax, %r30 # add to %eax, set
                                cond.codes
ld %r31, [%esp]
add.c %ebx, %ebx, %r31
ld %esi, [%ebp]
sub.c %ecx, %ecx, 5
```


Compiler Optimizations

Optimize:

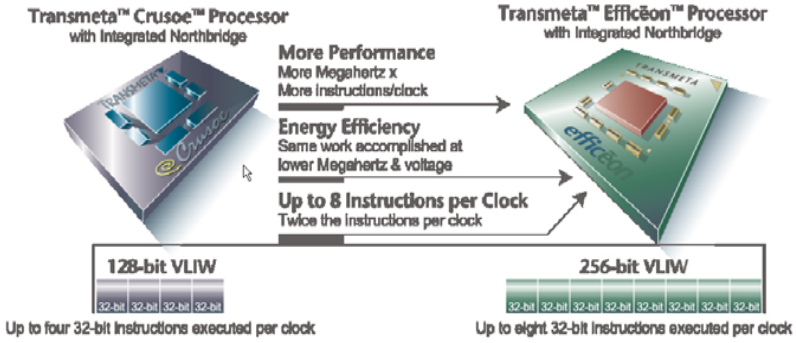
```
ld %r30, [%esp]           # load from stack only once
add %eax, %eax, %r30
add %ebx, %ebx, %r30  # reuse data loaded earlier
ld %esi, [%ebp]

sub.c %ecx, %ecx, 5      # only this cond. code needed
```


Scheduling

Schedule into VLIW code:

```
ld %r30, [%esp]; sub.c %ecx, %ecx, 5
ld %esi, [%ebp]; add %eax, %eax, %r30; add %ebx, %ebx,%r30
```


More Work Per GHz plus More GHz



A New Processor for Power Efficient Computing

Efficeon is the sum of

x86 Code Morphing Software +  =

Code Morphing Software

- Provides Compatibility
- Translates the 1's and 0's of x86 instructions to equivalent 1's and 0's for a simple VLIW processor
- Learns and improves with time

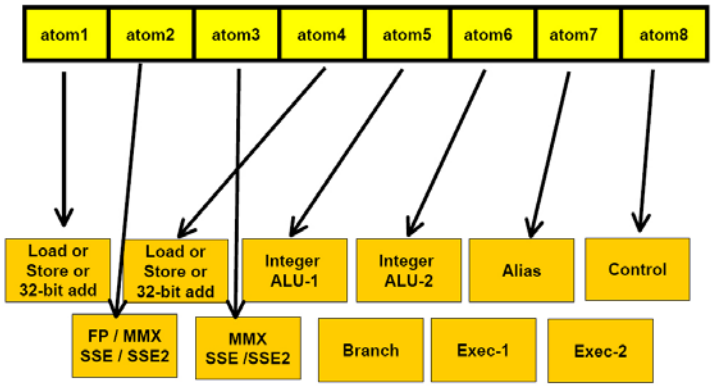
VLIW Hardware

- Very Long Instruction Word processor
- Simple and fast
- Fewer transistors

ICCA´04 5th Internal Conference Computer Architecture

A New Processor for Power Efficient Computing

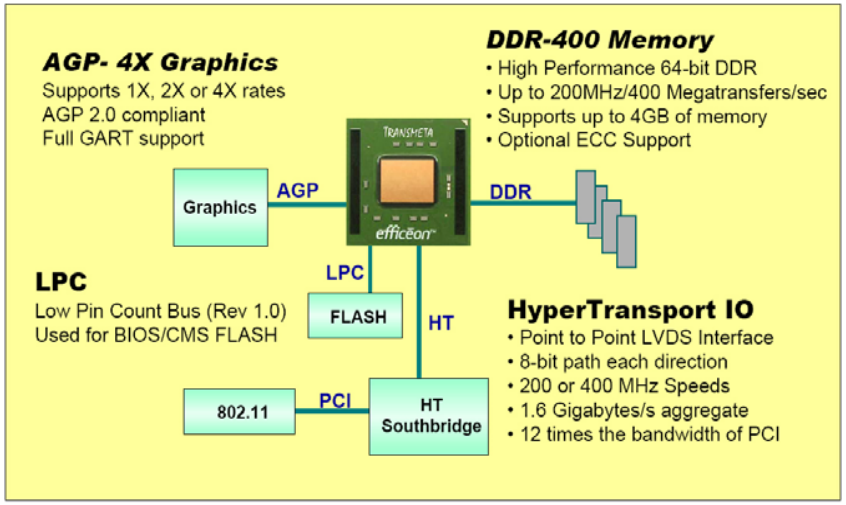
Each clock, Efficeon can issue from one to eight 32-bit instruction "atoms". . .



. . . to any of the above eleven logical execution units.

ICCA´04 5th Internal Conference Computer Architecture

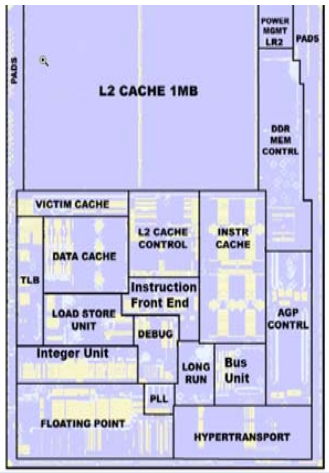
A New Processor for Power Efficient Computing



Support for High Performance Graphics, Memory, and Communications

ICCA´04 5th Internal Conference Computer Architecture

A New Processor for Power Efficient Computing



Efficeon die size includes Northbridge and AGP Port:

130 nm technology:
119 mm^2 die size

90 nm technology:
68 mm^2 die size

A New Processor for Power Efficient Computing

Trades Transistors for Software

ICCA´04 5th Internal Conference Computer Architecture

A New Processor for Power Efficient Computing

TDP = maximum Thermal Design Power

- For many form factors, TDP often limits the maximum MHz
- 7 Watts TDP tends to be about the upper limit for fanless notebook systems

What MHz can you achieve within a 7 Watt TDP?

- Different processors achieve different MHz for the same TDP limit
- To compare performance, you must compare at the maximum MHz for a given TDP limit

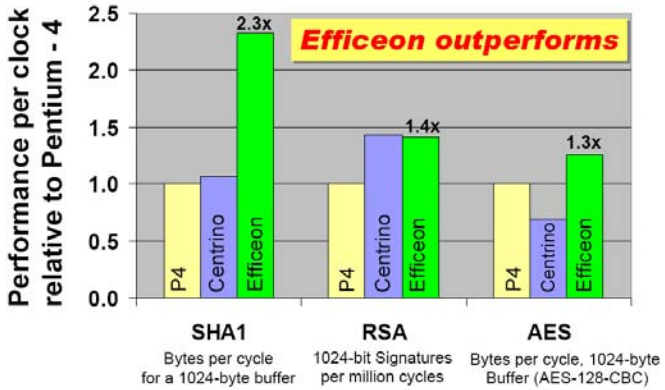
For example, maximum MHz for the common 7W TDP envelope:

Intel Centrino	Transmeta Efficeon
7 Watt TDP	7 Watt TDP
900 MHz	1100 MHz

ICCA´04 5th Internal Conference Computer Architecture

A New Processor for Power Efficient Computing

SHA1, RSA and AES are the basis for modern encryption suites, and are a good real-world benchmark of computationally intensive integer codes.



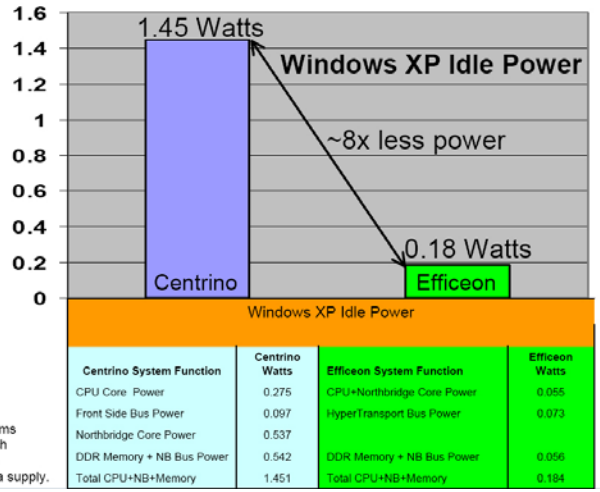
ICCA´04 5th Internal Conference Computer Architecture

A New Processor for Power Efficient Computing

Standby power is key for preserving battery life.

Closest *"apples-to-apples"* comparison is to compare key components of

- CPU Core
- CPU IO
- Northbridge Co
- Northbridge IO
- DRAM



Centrino System Function	Centrino Watts	Efficeon System Function	Efficeon Watts
CPU Core Power	0.275	CPU+Northbridge Core Power	0.055
Front Side Bus Power	0.097	HyperTransport Bus Power	0.073
Northbridge Core Power	0.537		
DDR Memory + NB Bus Power	0.542	DDR Memory + NB Bus Power	0.056
Total CPU+NB+Memory	1.451	Total CPU+NB+Memory	0.184

Measured in comparable systems using supply voltages from each rail to subsystem components. DRAM and Northbridge share a supply.