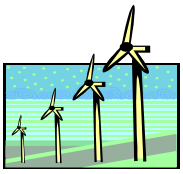
Processing units

Need for efficiency (power + energy):

Why worry about energy and power?



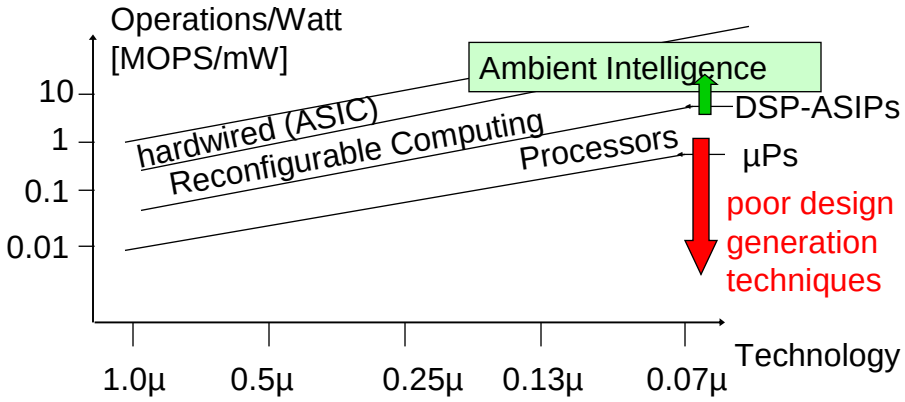
„Power is considered as the most important constraint in embedded systems“

[in: L. Eggermont (ed): Embedded Systems Roadmap 2002, STW]

Current UMTS phones can hardly be operated for more than an hour, if data is being transmitted.

[from a report of the Financial Times, Germany, on an analysis by Credit Suisse First Boston; http://www.ftd.de/tm/tk/9580232.html?nv=se]

The energy/flexibility conflict - Intrinsic Power Efficiency -



Necessary to optimize!

[H. de Man, Keynote, DATE‘02; T. Claasen, ISSCC99]

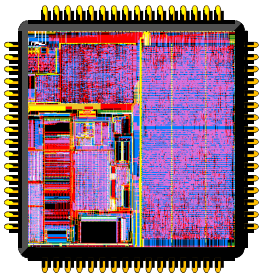
Target technologies

1. ASICs
2. Processors
   - Energy efficient
   - Code-size efficient
   - Run-time efficient
     - Special features of DSP processors
     - Multimedia instructions
     - Very long instruction word machines
3. Reconfigurable hardware

Application Specific Circuits (ASICS) or Full Custom Circuits

Custom-designed circuits necessary if ultimate speed or energy efficiency is the goal and large numbers can be sold.

Approach suffers from long design times and high costs (e.g. Mill. $ mask costs).



ASIC synthesis not covered in this course.

Processors

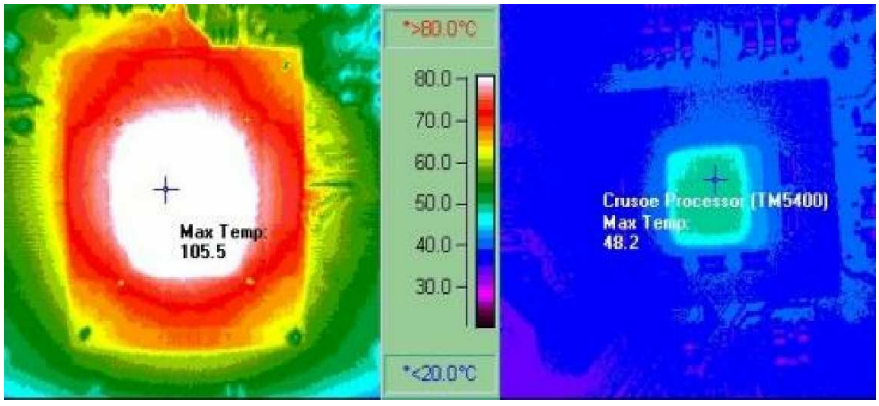
At the chip level, embedded chips include micro-controllers and microprocessors. Micro-controllers are the true workhorses of the embedded family. They are the original 'embedded chips' and include those first employed as controllers in elevators and thermostats [Ryan, 1995].

Key requirements:

1. **Energy-efficiency**
2. **Code-size efficiency:**
   Memory is a scarce resource in embedded systems, in particular for „systems-on-a-chip" (SoC).
3. **Run-time efficiency**

New ideas can actually reduce energy consumption

Pentium Crusoe



Running the same multimedia application.

As published by Transmeta [www.transmeta.com]

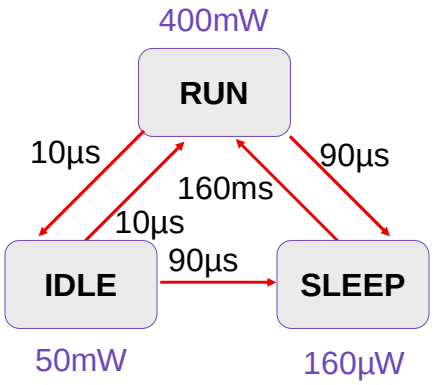
Dynamic power management (DPM)

Example: STRONGARM SA1100

RUN: operational

IDLE: a sw routine may stop the CPU when not in use, while monitoring interrupts

SLEEP: Shutdown of on-chip activity



Fundamentals of dynamic voltage scaling (DVS)

Power consumption of CMOS circuits (ignoring leakage):

$P = \alpha C_L V_{dd}^2 f$ with

α: switching activity

C_L: load capacitance

V_{dd}: supply voltage

f: clock frequency

Delay for CMOS circuits:

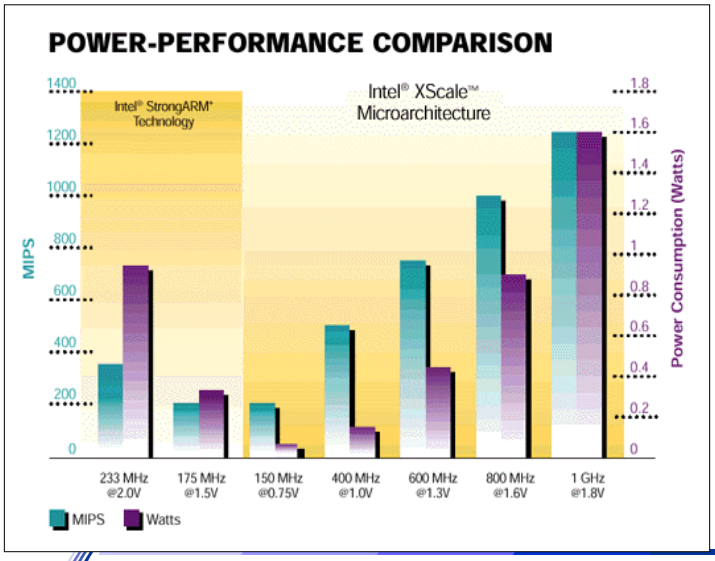
$$\tau = k C_L \frac{V_{dd}}{(V_{dd} - V_t)^2} \text{ with}$$

V_t: threshhold voltage

(V_t substancially < than V_{dd})

☞ Decreasing V_{dd} reduces P quadratically, while the run-time of algorithms is only linearly increased (ignoring the effects of the memory system).

Variable-voltage/frequency example: INTEL Xscale

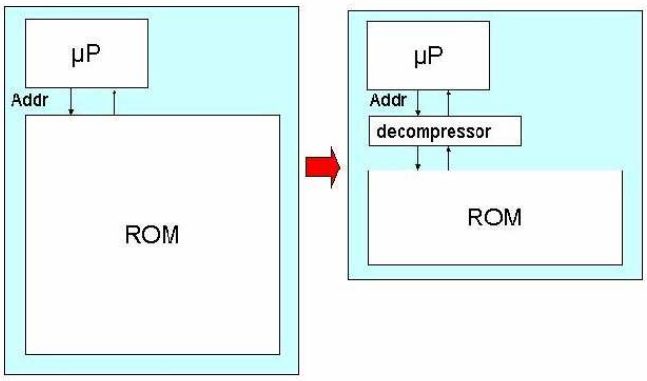


OS should schedule distribution of the energy budget.

From Intel's Web Site

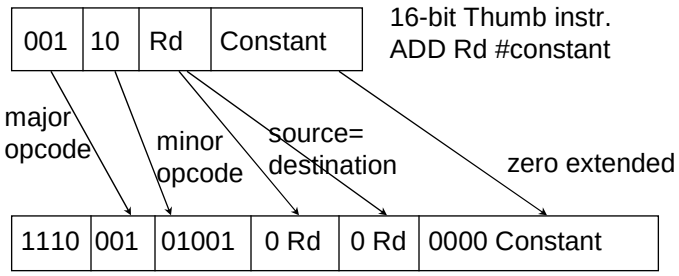
Code-size efficiency

- **CISC machines**: RISC machines designed for run-time-, not for code-size-efficiency
- **Compression techniques:** key idea



Code-size efficiency

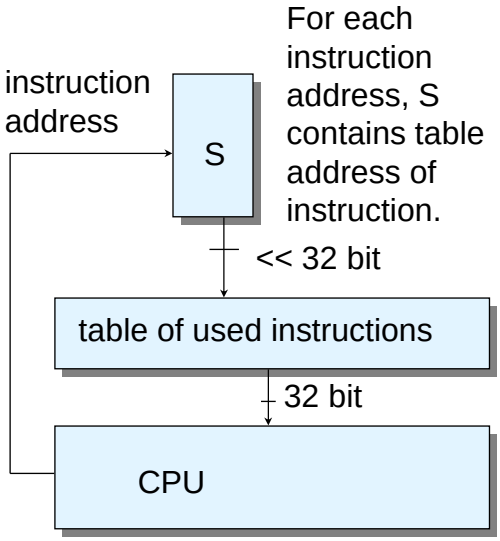
- **Compression techniques (continued):**
  - 2nd instruction set, z.B. ARM Thumb instruction set:



- Reduction to 65-70 % of original code size
- 130% of ARM performance with 8/16 bit memory
- 85% of ARM performance with 32-bit memory

[ARM, R. Gupta]

Two-level control store concept (indirect addressing of instructions)



Each instruction pattern is stored only once, and not repeatedly stored for each instruction address for which it is needed.

Similar to concept of colour lookup table.

Can be extended to include subroutines in lookup table.

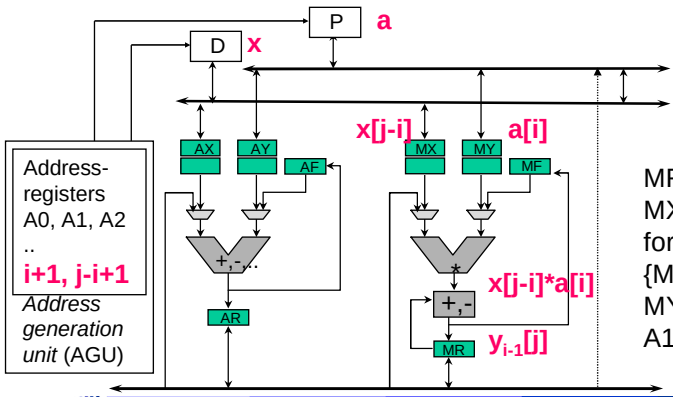
Called nanoprogramming in the Motorola 68000.

Run-time optimization: Domain-oriented architectures

Application: $y[j] = \sum_{i=0}^{n-1} x[j-i]*a[i]$

$\forall i: 0 \le i \le n-1: y_i[j] = y_{i-1}[j] + x[j-i]*a[i]$

Architecture: **Example: Data path ADSP210x**

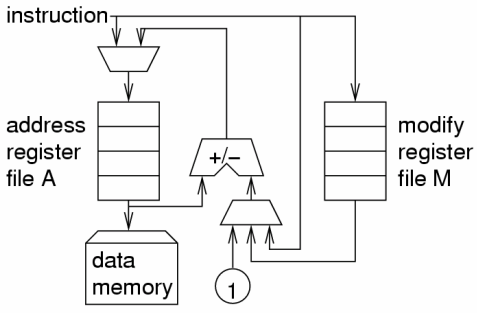


- Parallelism
- Dedicated registers

```
MR:=0; A1:=1; A2:=n-2;
MX:=x[n-1]; MY:=a[0];
for ( j:=1 to n)
{MR:=MR+MX*MY;
MY:=a[A1]; MX:=x[A2];
A1++; A2--}
```

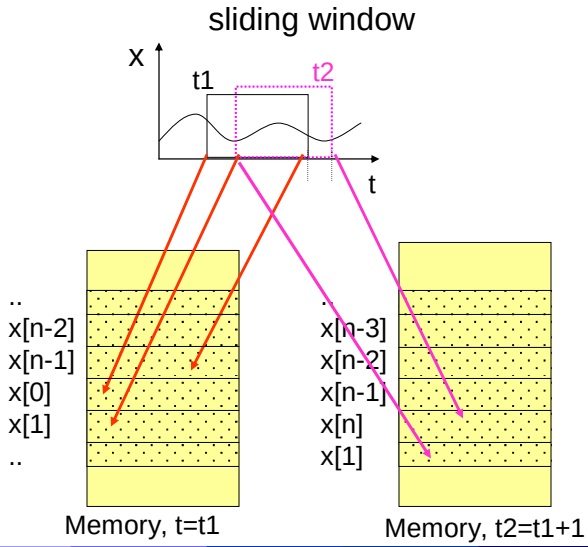

Digital Signal Processing (DSP) Processors - Features (1) -

- **Multiply/accumulate (MAC)** and **zero-overhead loop (ZOL) instructions** (as shown)
- **Heterogeneous registers** (as shown)
- **Separate address generation units** (AGUs) (as in ADSP 210x)



Digital Signal Processing (DSP) Processors - Features (2) -

- **Modulo addressing:**
  Am++ ≡
  Am:=(Am+1)
  mod n
  (implements ring or circular buffer in memory)

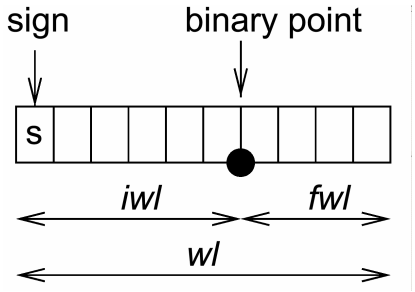


Saturating arithmetic

- **Saturating arithmetic**:
  returns largest/smallest number in case of over/underflows
  Example:

a		0111
b	+	1001
standard wrap around arithmetic		(1)0000
saturating arithmetic		1111
(a+b)/2: correct		1000
wrap around arithmetic		0000
saturating arithmetic + shifted		0111 „almost correct“

Fixed-point arithmetic



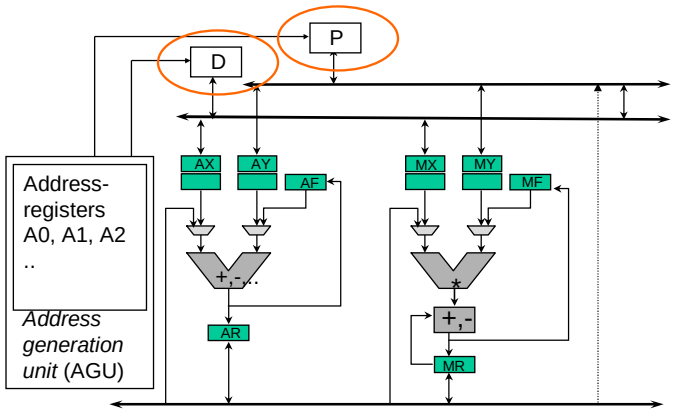
Shifting required after multiplications and divisions in order to maintain binary point.

Real-time capability

- **Timing behavior has to be predictable**
  Features that cause problems:
  - Caches, in particular with difficult to predict replacement strategies
  - Unified caches (conflicts between instructions and data)
  - Pipelines with difficult to predict stall cycles ("bubbles")
  - Interrupts that are possible any time
  - Memory refreshes that are possible any time
  - Instructions that have data-dependent execution times

[Dagstuhl workshop on predictability, Nov. 17-19, 2003]

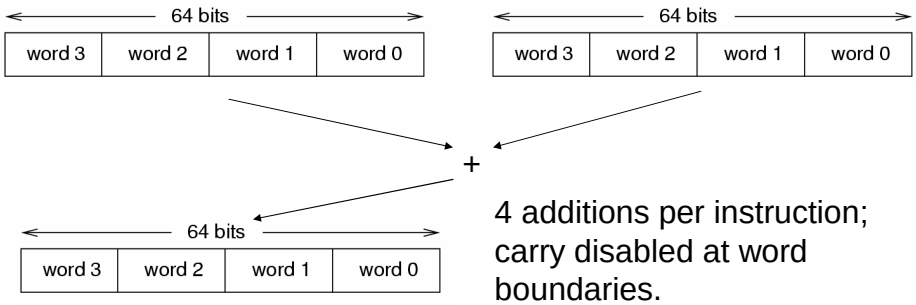
Multiple memory banks or memories



Simplifies parallel fetches

Multimedia-Instructions/Processors

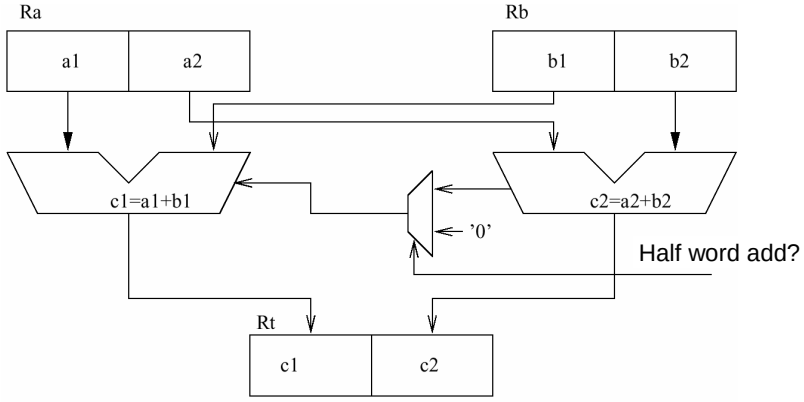
Multimedia instructions exploit that many registers, adders etc are quite wide (32/64 bit),
whereas most multimedia data types are narrow
(e.g. 8 bit per color, 16 bit per audio sample per channel)
☞ 2-8 values can be stored per register and added. E.g.:



4 additions per instruction; carry disabled at word boundaries.

Early example: HP *precision architecture* (hp PA)

Half word add instruction **HADD**:



Optional saturating arithmetic.
Up to 10 instructions can be replaced by **HADD**.

Pentium MMX-architecture (1)

64-bit vectors representing 8 byte encoded, 4 word encoded or 2 double word encoded numbers.
wrap around/*saturating* options.
Multimedia registers mm0 - mm7,
consistent with floating-point registers (OS unchanged).

Instruction	Options	Comments
Padd[b/w/d] PSub[b/w/d]	*wrap around,* *saturating*	addition/subtraction of bytes, words, double words
Pcmpeq[b/w/d] Pcmpgt[b/w/d]		Result= "11..11" if true, "00..00" otherwise Result= "11..11" if true, "00..00" otherwise
Pmullw Pmulhw		multiplication, 4*16 bits, least significant word multiplication, 4*16 bits, most significant word

Pentium MMX-architecture (2)

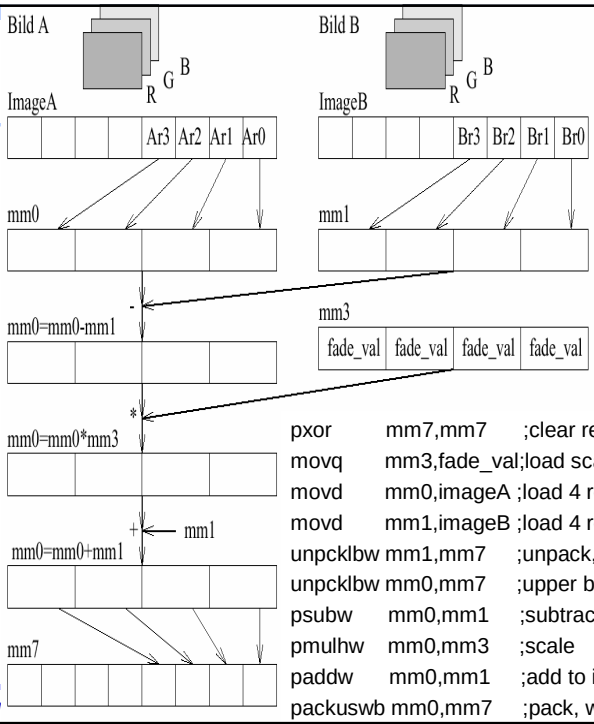
Psra[w/d] Psll[w/d/q] Psrl[w/d/q]	No. of positions in register or instruction	Parallel shift of words, double words or 64 bit quad words
Punpckl[bw/wd/dq] Punpckh[bw/wd/dq]		Parallel unpack Parallel unpack
Packss[wb/dw]	*saturating*	Parallel pack
Pand, Pandn Por, Pxor		Logical operations on 64 bit words
Mov[d/q]		Move instruction

Appli-cation

Scaled interpolation between two images

Next word = next pixel, same color.

4 pixels processed at a time.




```
pxor      mm7,mm7     ;clear register mm7
movq      mm3,fade_val;load scaling value
movd      mm0,imageA  ;load 4 red pixels for A
movd      mm1,imageB  ;load 4 red pixels for B
unpcklbw  mm1,mm7     ;unpack,bytes to words
unpcklbw  mm0,mm7     ;upper bytes from mm7
psubw     mm0,mm1     ;subtract pixel values
pmulhw    mm0,mm3     ;scale
paddw     mm0,mm1     ;add to image B
packuswb  mm0,mm7     ;pack, words to bytes
```

Ultra-SPARC Processor

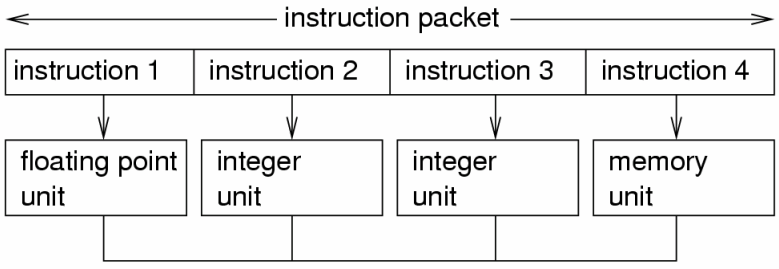
visual instruction set (VIS)

- Instruction for MPEG motion estimation,
  includes 8 subtractions, 8 additions and 8 absolute value computations on 8 bit data in a single cycle.
  Replaces up to 1500 instructions by 32 of such instructions.
- ..

Very long instruction word (VLIW) processors

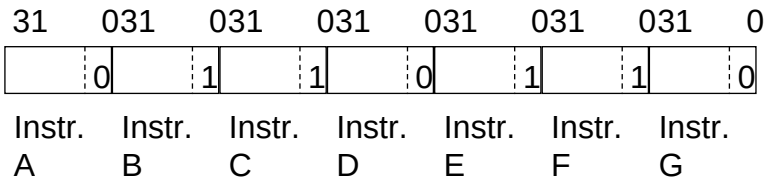
Key idea: detection of possible parallelism to be done by compiler, not by hardware at run-time (inefficient).

VLIW: parallel operations (instructions) encoded in one long word (instruction packet), each instruction controlling one functional unit. E.g.:



Avoiding unused instructions with explicit parallelism instruction computers (EPIC)

The TMS320C62xx VLIW Processor as an example:



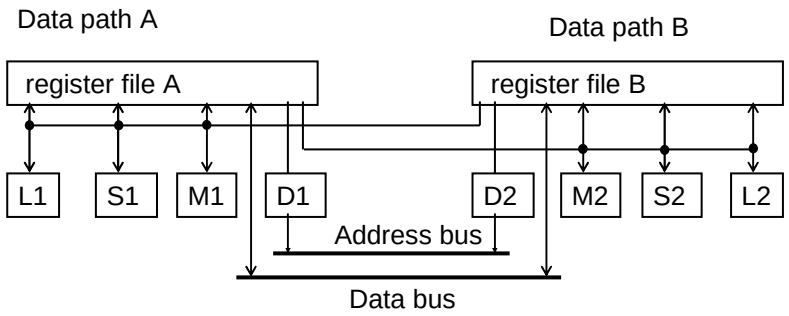
Cycle	Instruction		
1	A		
2	B	C	D
3	E	F	G

Instructions B, C and D cannot use any of the same functional units, cross paths or other data path resources. The same is also true for E, F and G.

Partitioned register files

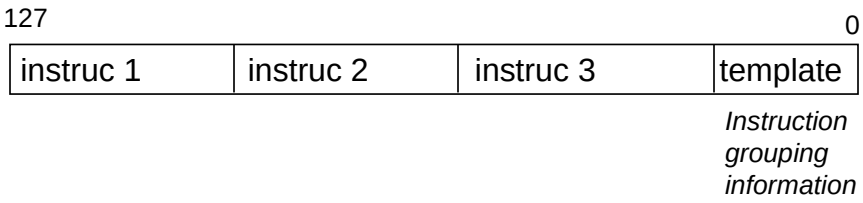
- Many memory ports are required to supply enough operands per cycle.
- Memories with many ports are expensive.

☞ Registers are partitioned into (typically 2) sets, e.g. for TI C60x:

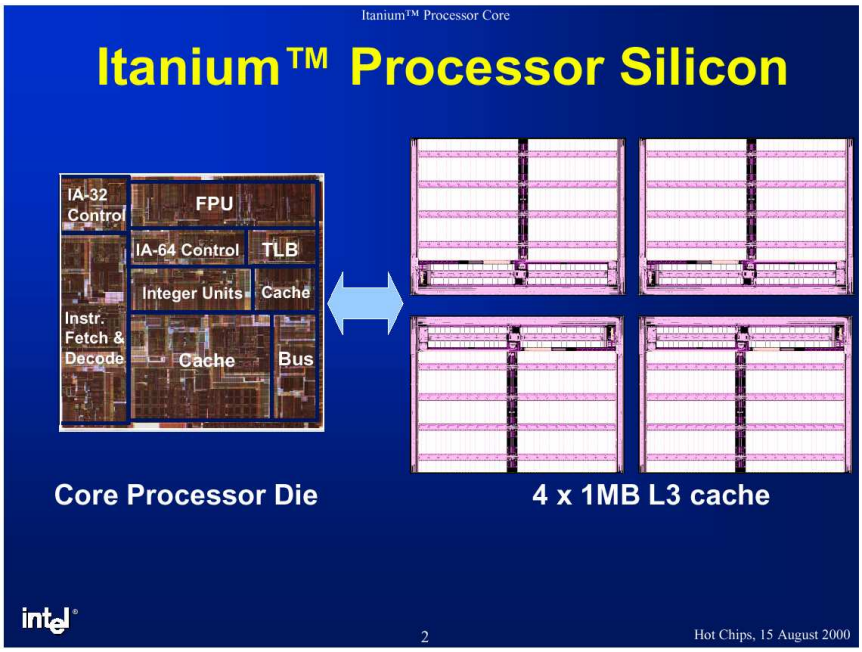


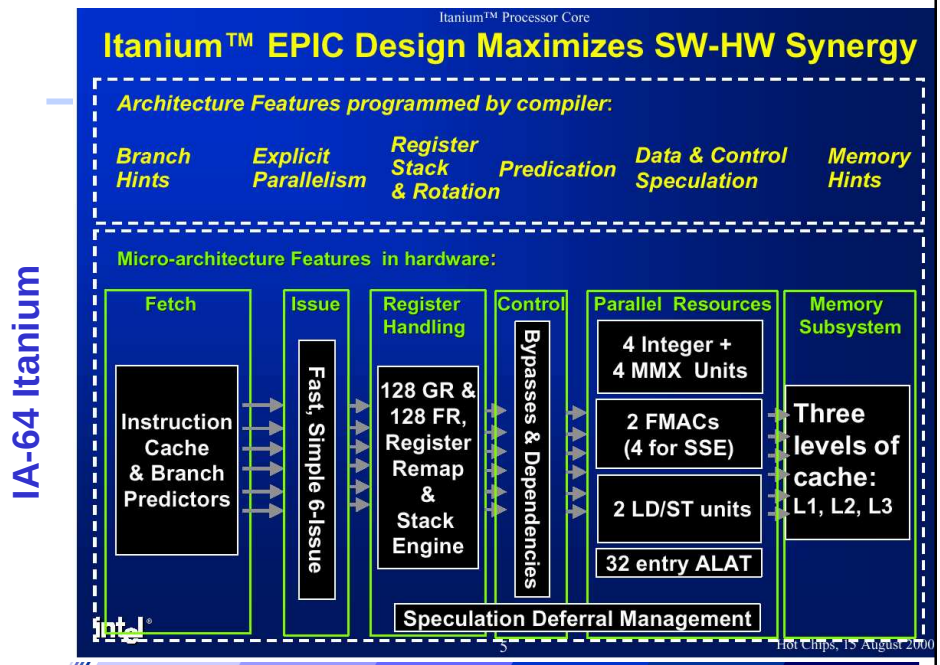
IA-64 Itanium

Succesor to Pentium-Architecture, EPIC



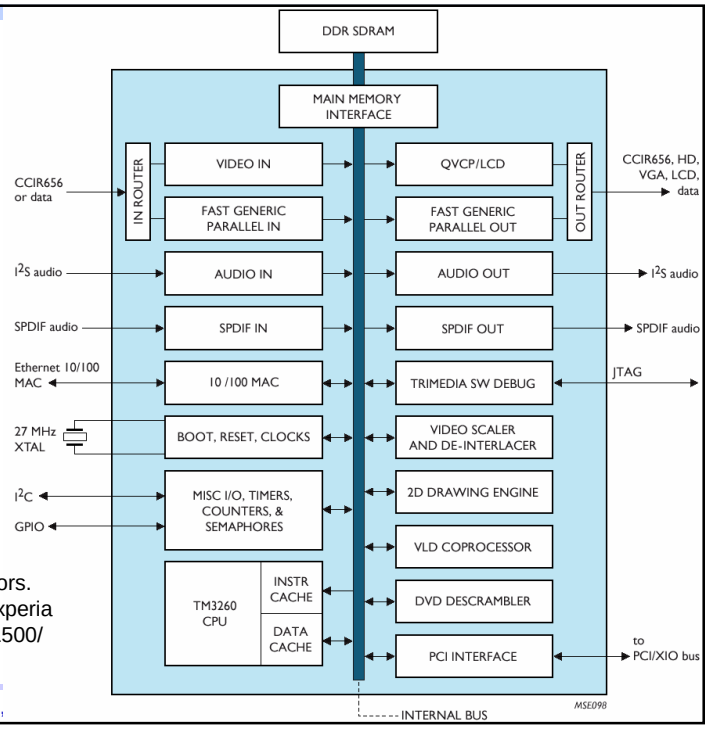
IA-64 Itanium





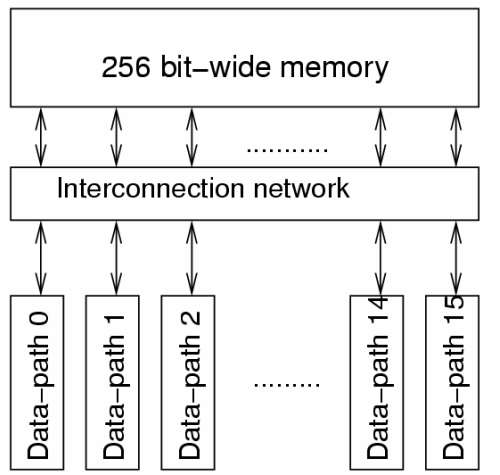
Philips TriMedia-Processor

For multimedia-applications, up to 5 instructions/ cycle.



http://www.semiconductors.philips.com/products/nexperia/media_processing/pnx1500/
© Philips

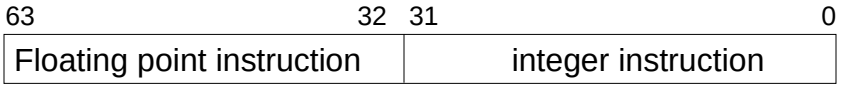
The M3 VLIW DSP Processor



Designed at TU Dresden
(G. Fettweis et al.)

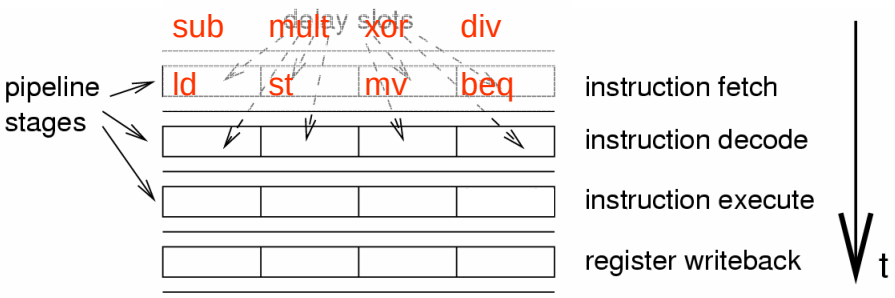
Other VLIW-/EPIC-Architectures

- **Intel i860**: 64-Bit instructions.
  32 bits for floating point operations
  32 bits for integer units.
  Not enough flexibility.

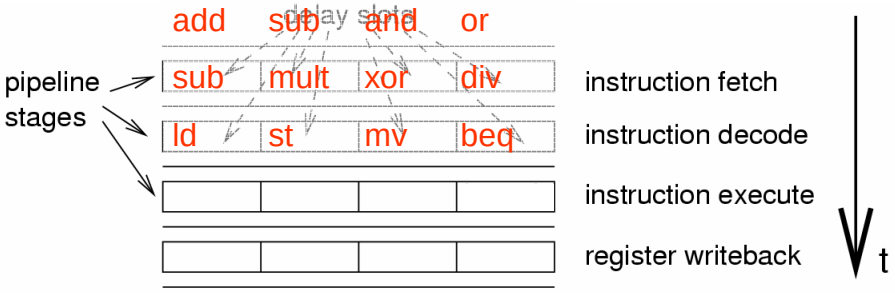


- **PROLOG-machine** designed by project group at Dortmund

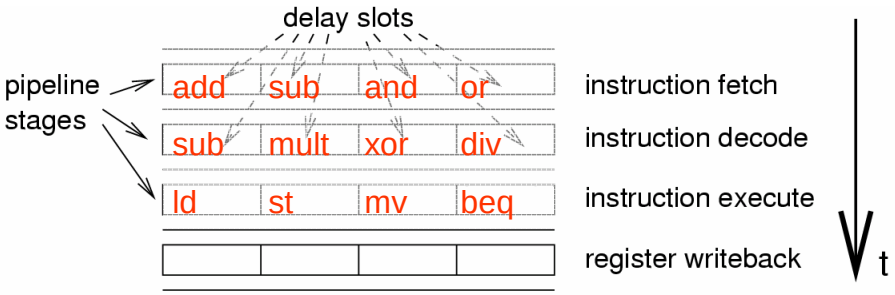
Large # of delay slots, a problem of VLIW processors



Large # of delay slots, a problem of VLIW processors



Large # of delay slots, a problem of VLIW processors



The execution of many instructions has been started before it is realized that a branch was required.
Nullifying those instructions would waste compute power

- ☞ Executing those instructions is declared a feature, not a bug.
- ☞ How to fill all „delay slots“ with useful instructions?
- ☞ Avoid branches wherever possible.

Predicated execution: Implementing IF-statements „branch-free“

Conditional Instruction „[c] I“ consists of:

- **condition c**
- **instruction I**

c = true => I executed
c = false => NOP

Predicated execution: Implementing IF-statements „branch-free“: TI C6x

```
if (c)
{ a = x + y;
  b = x + z;
}
else
{ a = x - y;
  b = x - z;
}
```

Conditional branch

```
      [c] B L1
          NOP 5
          B L2
          NOP 4
          SUB x,y,a
      ||  SUB x,z,b
L1:       ADD x,y,a
      ||  ADD x,z,b
L2:
```

max. 12 cycles

Predicated execution

```
      [c]  ADD x,y,a
|| [c]  ADD x,z,b
|| [!c] SUB  x,y,a
|| [!c] SUB  x,z,b
```

1 cycle

Microcontrollers - MHS 80C51 as an example -

Features for Embedded Systems

- 8-bit CPU optimised for control applications
- Extensive Boolean processing capabilities
- 64 k Program Memory address space
- 64 k Data Memory address space
- 4 k bytes of on chip Program Memory
- 128 bytes of on chip data RAM
- 32 bi-directional and individually addressable I/O lines
- Two 16-bit timers/counters
- Full duplex UART
- 6 sources/5-vector interrupt structure with 2 priority levels
- On chip clock oscillators
- Very popular CPU with many different variations

Reconfigurable Logic

Full custom chips may be too expensive,

software may be too slow.

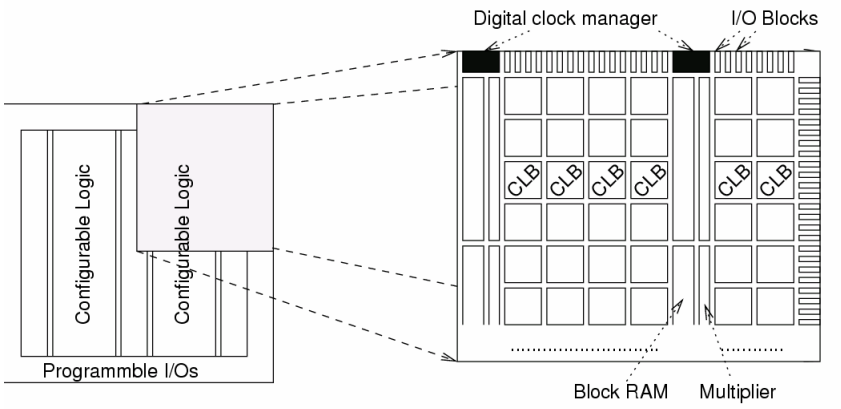
☞ Use of configurable hardware;
most common form: field programmable gate arrays (FPGAs)

Considered e.g. for configuring mobile phone according to local standards;

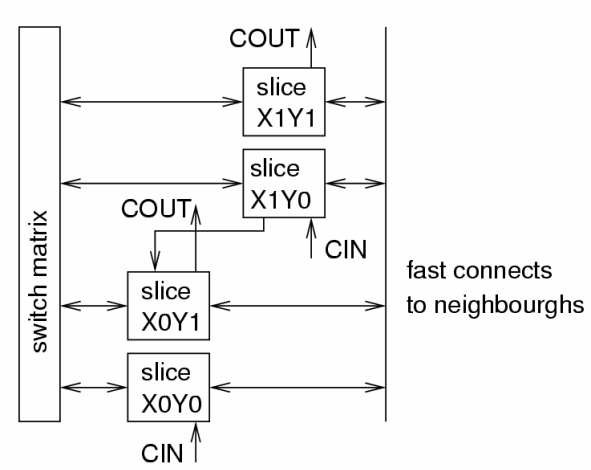
Fast „object recognition“.

Example: Xilinx Virtex II FPGAs

Floor-plan of VIRTEX II FPGAs



Virtex II Configurable Logic Block (CLB)



2 carry paths per CLB (Vertex II Pro)

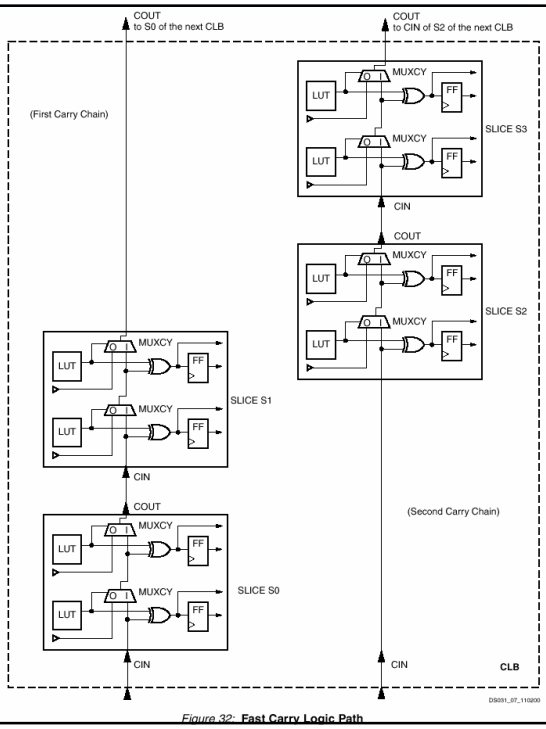
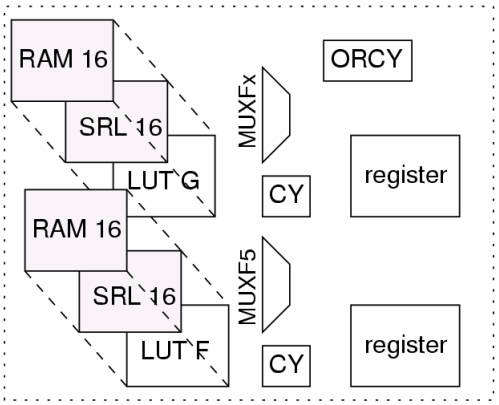


Figure 32: Fast Carry Logic Path

[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]

Virtex II Slice (simplified)

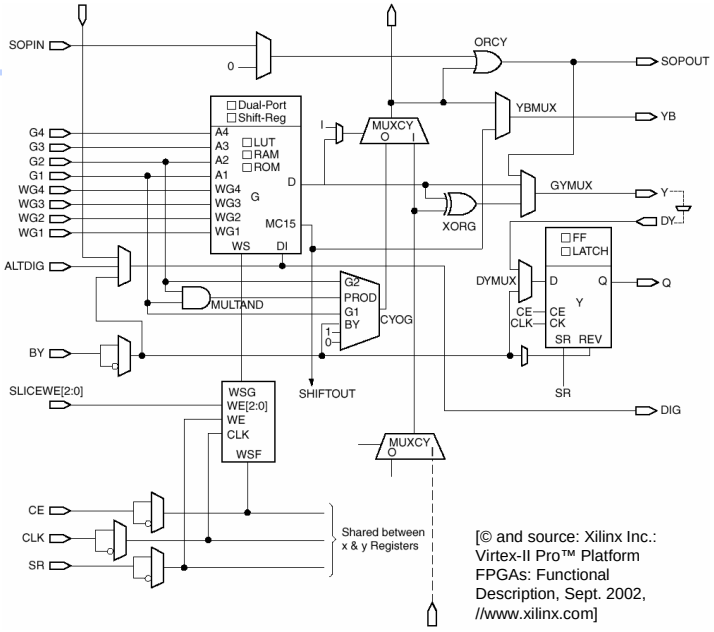
Look-up tables LUT F and G can be used to compute any Boolean function of ≤ 4 variables.



Example:

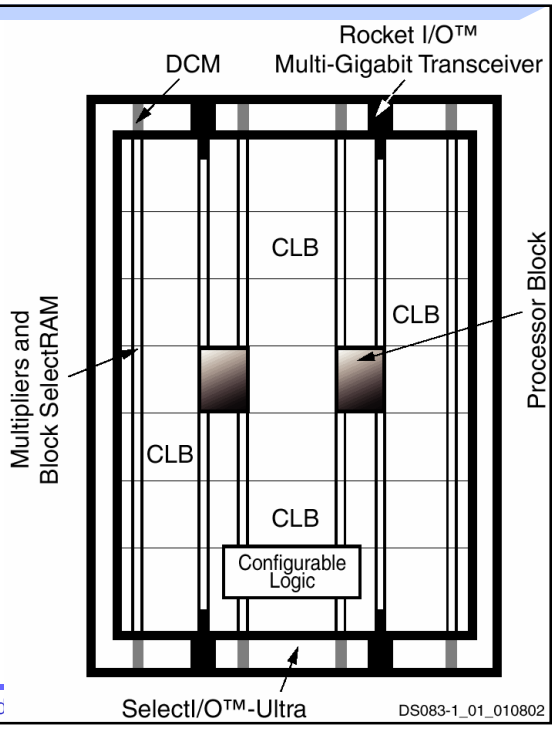
a	b	c	d	**G**
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**0**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**0**

Virtex II (Pro) Slice



[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]

Virtex II Pro Devices include up to 4 PowerPC processor cores



[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]

Number of resources available in Virtex II Pro devices

Table 16: **Virtex-II Pro Logic Resources Available in All CLBs**

Device	CLB Array: Row x Column	Number of Slices	Number of LUTs	Max Distributed SelectRAM+ or Shift Register (bits)	Number of Flip-Flops	Number of Carry Chains(1)	Number of SOP Chains(1)
XC2VP2	16 x 22	1,408	2,816	45,056	2,816	44	32
XC2VP4	40 x 22	3,008	6,016	96,256	6,016	44	80
XC2VP7	40 x 34	4,928	9,856	157,696	9,856	68	80
XC2VP20	56 x 46	9,280	18,560	296,960	18,560	92	112
XC2VP30	80 x 46	13,696	27,392	438,272	27,392	92	160
XC2VP40	88 x 58	19,392	38,784	620,544	38,784	116	176
XC2VP50	88 x 70	23,616	47,232	755,712	47,232	140	176
XC2VP70	104 x 82	33,088	66,176	1,058,816	66,176	164	208
XC2VP100	120 x 94	44,096	88,192	1,411,072	88,192	188	240
XC2VP125	136 x 106	55,616	111,232	1,779,712	111,232	212	272

Notes:
1. The carry-chains and SOP chains can be split or cascaded.

[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]