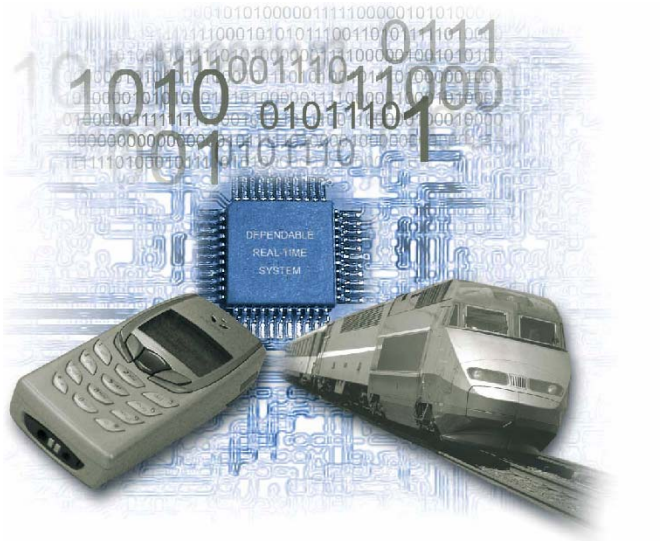
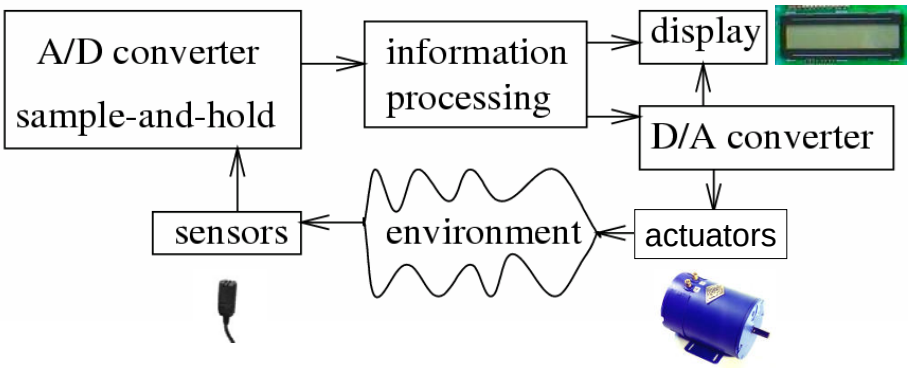
Embedded Systems 12



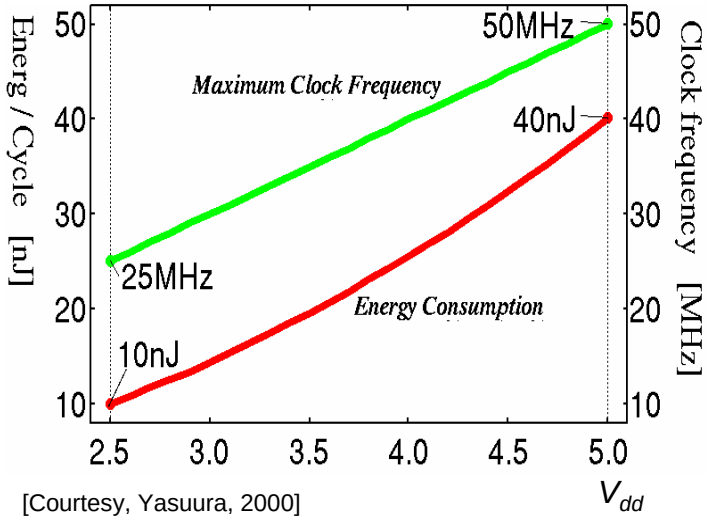
Embedded System Hardware REVIEW

- Embedded system hardware is frequently used in a loop (***„hardware in a loop“***):



Key requirement #1: Energy efficiency Voltage scaling

REVIEW

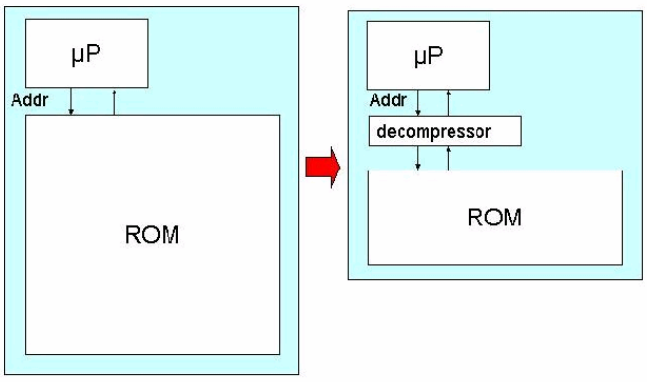


[Courtesy, Yasuura, 2000]

Key requirement #2: Code-size efficiency

REVIEW

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



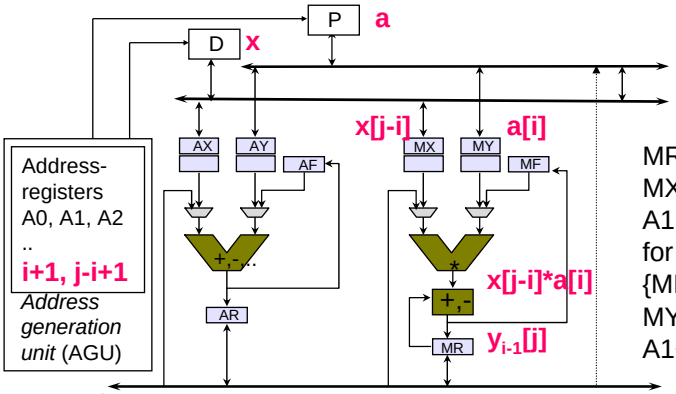
Key requirement #3: Run-time efficiency - Domain-oriented architectures -

REVIEW

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



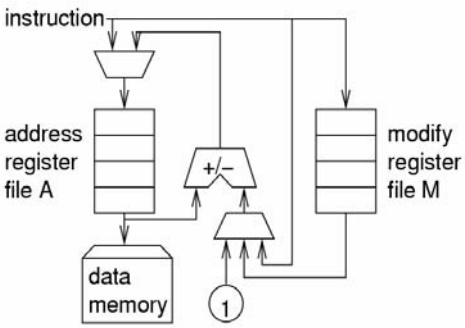
Application maps nicely onto architecture

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


Separate address generation units (AGUs)

REVIEW

Example (ADSP 210x):



- Data memory can only be fetched with address contained in A,
- but this can be done in parallel with operation in main data path (takes effectively 0 time).
- A := A ± 1 also takes 0 time,
- same for A := A ± M;
- A := <immediate in instruction> requires extra instruction

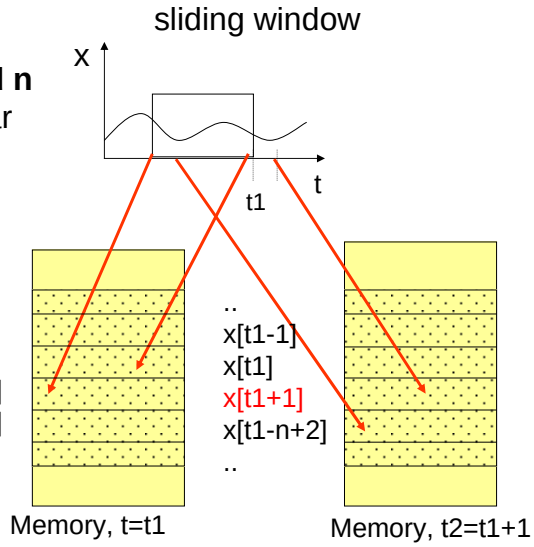
☞ Minimize load immediates

☞ Optimization in optimization chapter

Modulo addressing

REVIEW

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



Saturating arithmetic

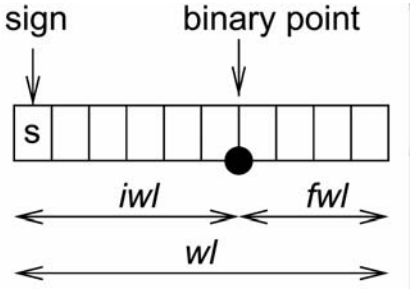
REVIEW

- 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“

- Appropriate for DSP/multimedia applications:
  - No timeliness of results if interrupts are generated for overflows
  - Precise values less important
  - Wrap around arithmetic would be worse.

Fixed-point arithmetic



- Less significant digits are chopped off.
- Example:
  x= 0.5 x 0.125 + 0.25 x 0.125 = 0.0625 + 0.03125 = 0.09375
  For *iwl*=1 and *fwl*=3 decimal digits, the less significant digits are automatically chopped off: x = 0.093
  Like a floating point system with numbers ∈ (-1..1), with no stored exponent.
- Appropriate for DSP/multimedia applications (well-known value ranges).

Real-time capability

- **Timing behavior has to be predictable**
  Features that cause problems:
  - Unpredictable access to shared resources
    - Caches with difficult to predict replacement strategies
    - Unified caches (conflicts betw. instructions and data)
    - Pipelines with difficult to predict stall cycles ("bubbles")
    - Unpredictable communication times for multiprocessors
  - Branch prediction, speculative execution
  - Interrupts that are possible any time
  - Memory refreshes that are possible any time
  - Instructions that have data-dependent execution times

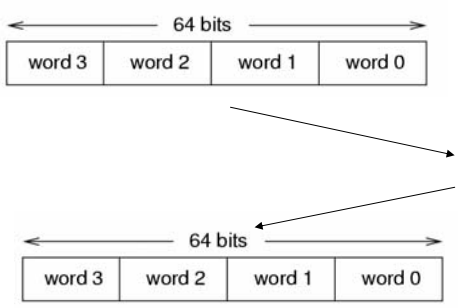
  ☞ Trying to avoid as many of these as possible.

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

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)

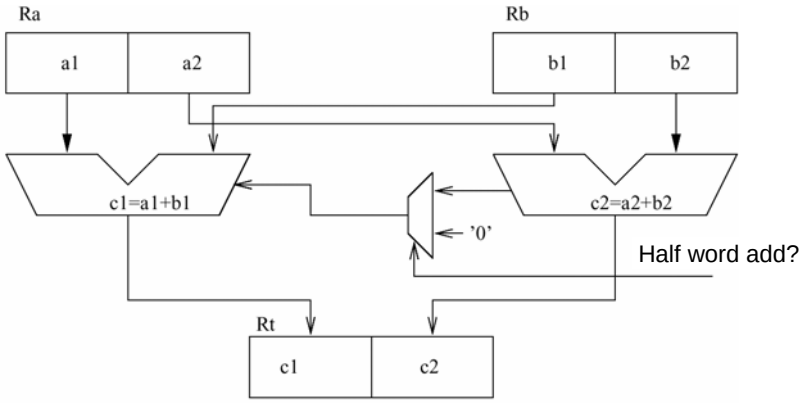
☞ 2-8 values can be stored per register and added. E.g.:



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)

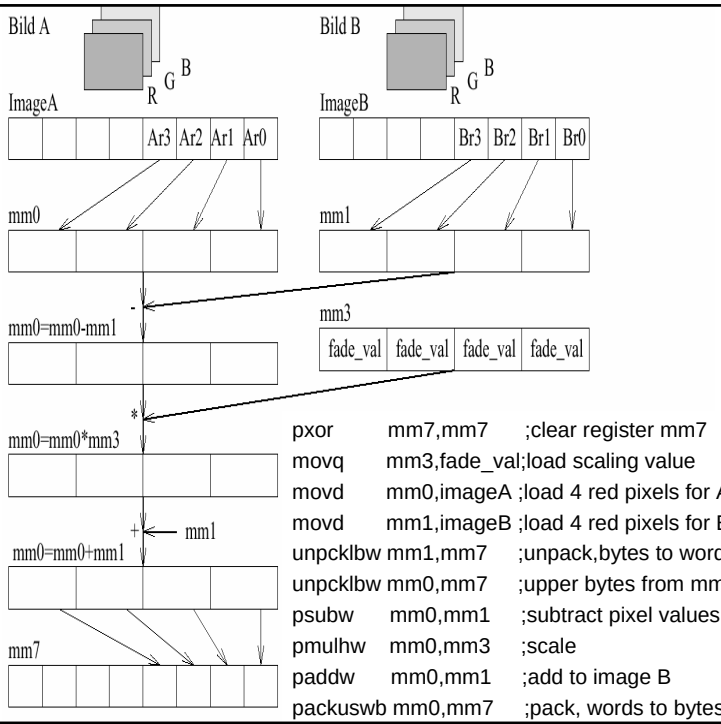
Psra[w/d] Psll[w/d/q] Psrl[w/d/q]	No. of positions in register or instruction	Parallel shift of words, double words or 64 bit quad words
Punpckl[bw/wd/dq] Punpckh[bw/wd/dq]		Parallel unpack Parallel unpack
Packss[wb/dw]	*saturating*	Parallel pack
Pand, Pandn Por, Pxor		Logical operations on 64 bit words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


Very long instruction word (VLIW) architectures

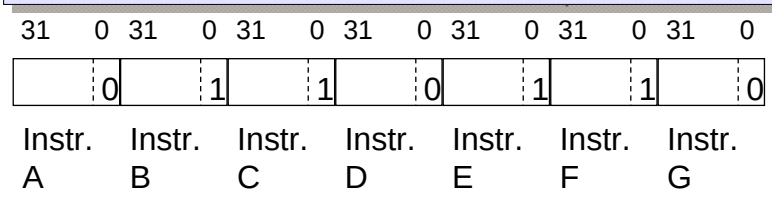
- Very long instruction word (“instruction packet”) contains several instructions, all of which are assumed to be executed in parallel.
- Compiler is assumed to general these “parallel” packets
- Complexity of finding parallelism is moved from the hardware (RISC/CISC processors) to the compiler;
  Ideally, this avoids the overhead (silicon, energy, ..) of identifying parallelism at run-time.

☞ A lot of expectations into VLIW machines

- Explicitly parallel instruction set computers (EPICs) are an extension of VLIW architectures: parallelism detected by compiler, but no need to encode parallelism in 1 word.

EPIC: TMS 320C6xx as an example

Bit in each instruction encodes end of parallel execution



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

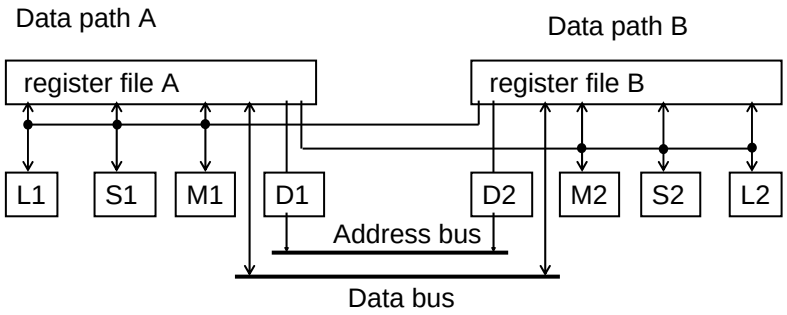
Instructions B, C and D use disjoint functional units, cross paths and other data path resources. The same is also true for E, F and G.

Parallel execution cannot span several packets.

Partitioned register files

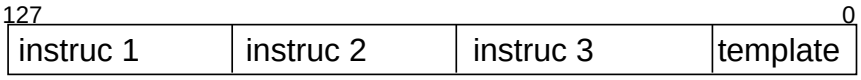
- Many memory ports are required to supply enough operands per cycle.
- Memories with many ports are expensive.

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

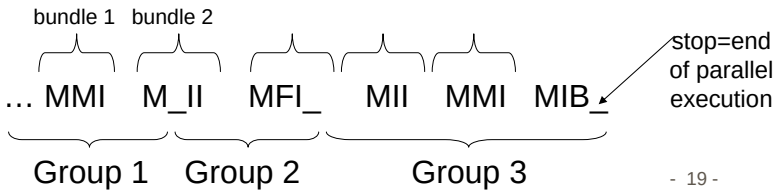


More encoding flexibility: IA-64 Itanium

3 instructions per **bundle:**

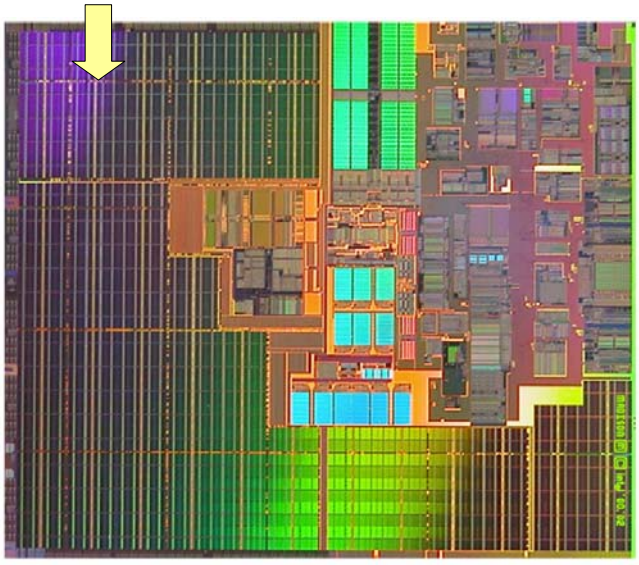


- There are 5 instruction types:
  - A: common ALU instructions
  - I: more special integer instructions (e.g. shifts)
  - M: Memory instructions
  - F: floating point instructions
  - B: branches



IA-64 Itanium

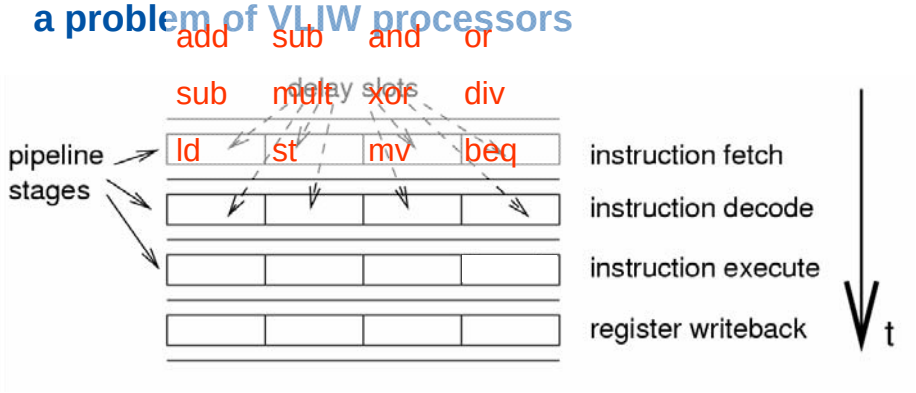
L3 cache



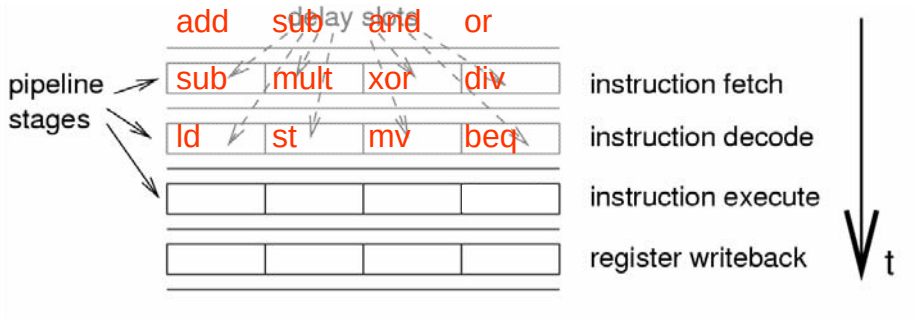
- 410M transistors
- 374 mm^2 die size
- 6MB on-die L3 cache
- 1.5 GHz at 1.3V

[ftp://download.intel.com/design/itanium2/download/madison_slides_r1.pdf]

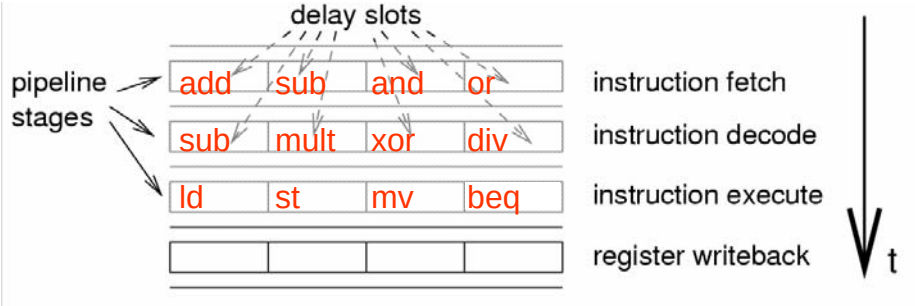
Large # of delay slots, a problem of VLIW processors



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

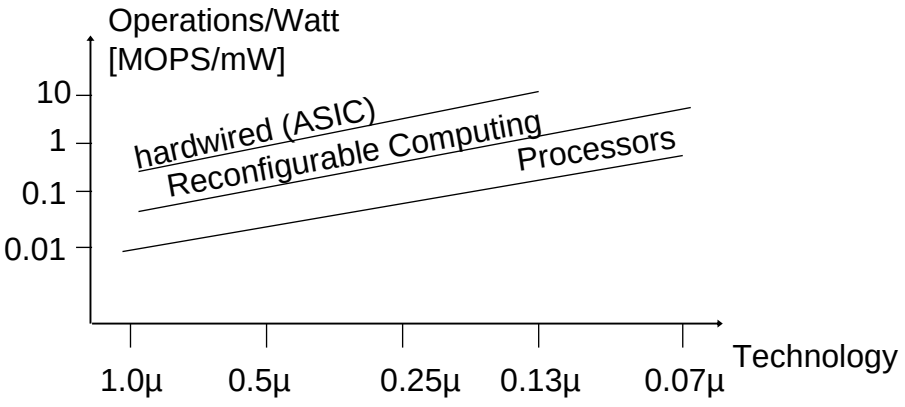
Predicated execution

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

1 cycle

Hardware Efficiency

REVIEW

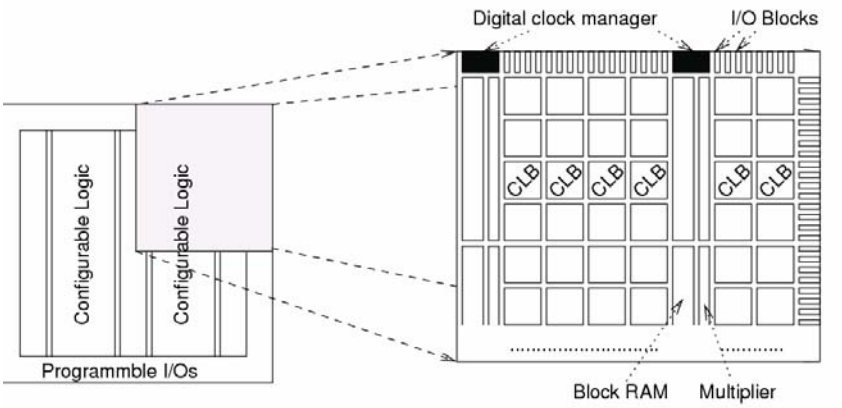


[H. de Man, Keynote, DATE‘02; T. Claasen, ISSCC99]

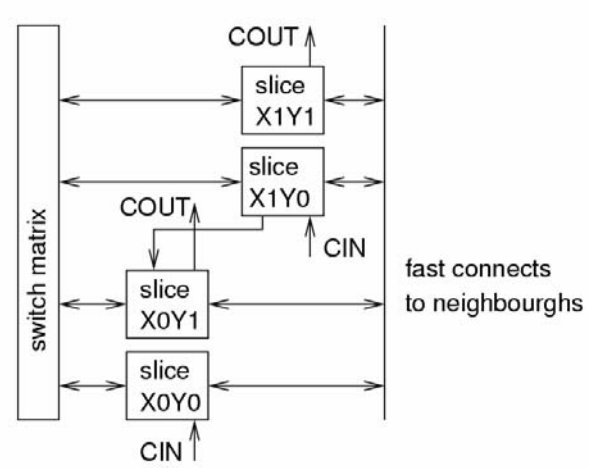
Reconfigurable Logic

- Full custom chips may be too expensive, software too slow.
- Combine the speed of HW with the flexibility of SW
  - ☞ HW with programmable functions and interconnect.
  - ☞ Use of configurable hardware;
    common form: field programmable gate arrays (FPGAs)
- ☞ Applications: bit-oriented algorithms like
  - encryption,
  - fast “object recognition“ (medical and military)
  - Adapting mobile phones to different standards.
- Very popular devices from
  - XILINX (XILINX Vertex II are recent devices)
  - Actel, Altera and others

Floor-plan of VIRTEX II FPGAs

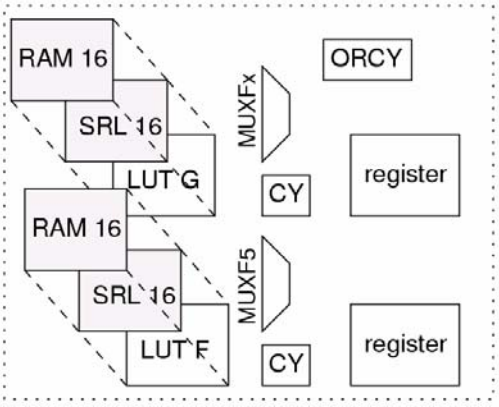


Virtex II Configurable Logic Block (CLB)



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**

Number of resources available in Virtex II Pro devices

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

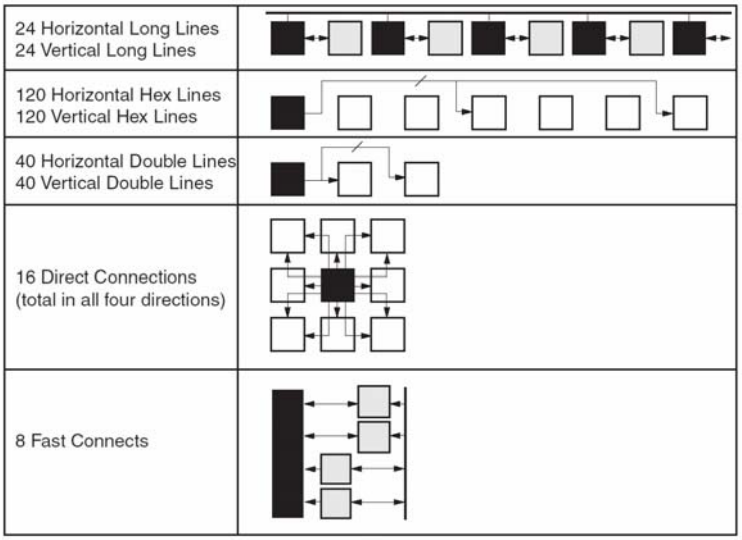
Device	CLB Array: Row x Column	Number of Slices	Number of LUTs	Max Distributed SelectRAM+ or Shift Register (bits)	Number of Flip-Flops	Number of Carry Chains(1)	Number of SOP Chains(1)
XC2VP2	16 x 22	1,408	2,816	45,056	2,816	44	32
XC2VP4	40 x 22	3,008	6,016	96,256	6,016	44	80
XC2VP7	40 x 34	4,928	9,856	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.

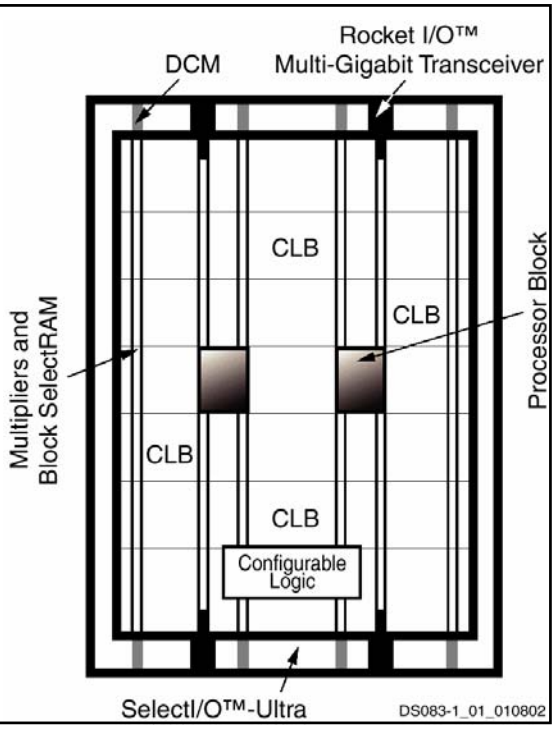
[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]

Interconnect

- Hierarchical Routing Resources

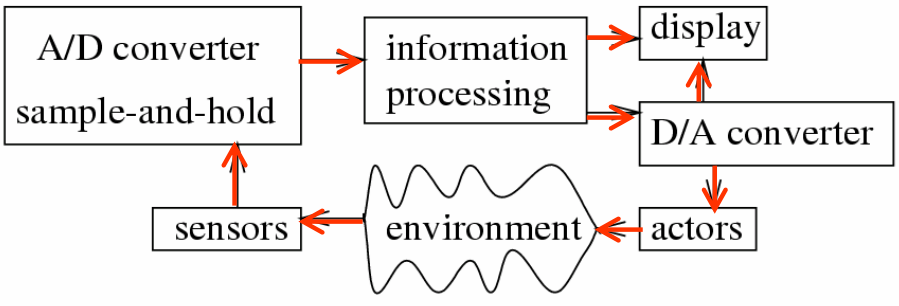


Virtex II Pro Devices include up to 4 PowerPC processor cores

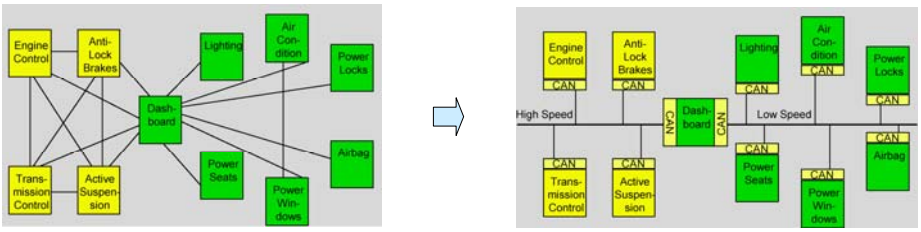


[© and source: Xilinx Inc.: Virtex-II Pro™ Platform FPGAs: Functional Description, Sept. 2002, //www.xilinx.com]

Real-Time Communication



Advantages of standardized communication media



1. Modular system development, support and evolution
2. Single network vs. wiring harness
   - Difficult to connect the wrong cable if there is only one
3. Low per-node cost permits repair by replacing entire node
   - Diagnosis by bus snooping
4. More flexible physical allocation

Requirements: Protocol Latency

- Low and almost constant (i.e., reliable) communication
  - Small latency = rapid communication, low distribution overhead
  - Low jitter = reliable correspondence between actual causal and observed temporal dependency
- Simultaneous or phase-constant delivery upon multicast
  - strict correlation of various distributed real-time images of dynamically evolving entities

Requirements: Support for Composability

- Temporal encapsulation of nodes
  - WCET analysis, scheduling, etc. are simplified if pace of program execution on nodes is independent of communication events
  - Analysis of protocol latency, jitter, etc. is simplified if communication is under autonomous control of the communication network
  - Established solution: computer/network interface (CNI) or I/O coprocessor autonomously performing message transfers
- Assume/guarantee between servers and clients
  - Server must be protected from burst loads; otherwise, it will be unable to fulfill its commitment
  - Established solution: design by contract — server delivers reliable service whenever clients stick to their contract concerning activation patterns
  - Communication layer enforcing the contract simplifies design through separation of concerns

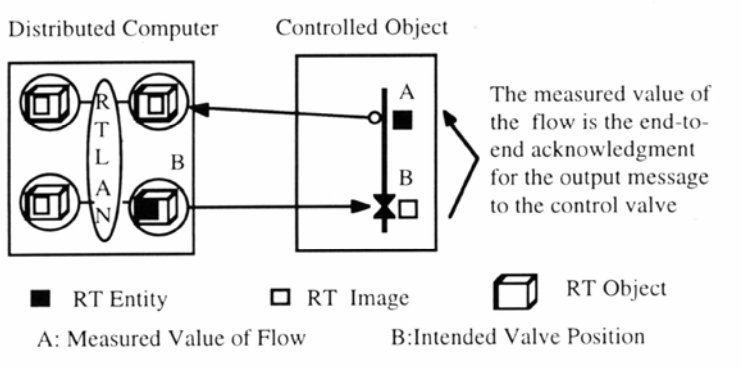
Requirements: Flexibility

- Multiple configurations
  - Customizable products, e.g. different equipment levels and optional equipment in cars, require spare and/or reassignable channel capacity.
  - Operational modes of the overall system: startup, normal operation, emergency, ...
  - Interleaving communications with extraordinary resource constraints: e.g., airbag deployment

Requirements: Error Detection

- Dependable service of communication system
  - Errors should be detected and corrected without seriously increasing communication jitter.
  - Unconcealable errors have to be reported asap to the affected communication partners to allow them to implement error masking.
- Detection of node errors
  - Node errors should be detected by communication system.
  - Erratic nodes should be isolated from the network in order to implement a fail-silent policy.
  - Communication partners relying on an erratic node's service have to be informed about lack of service (with low latency).
  - Recovered nodes should be reintegrated into network.
- Support of end-to-end acknowledgements
  - End-to-end acknowledgement permits fault detection over a whole causal chain.
  - Correlating requests and acknowledgements is difficult / impossible if end-to-end latency / jitter is larger than minimum separation between requests.

End-to-end Acknowledgement



- RT entity: an "object" whose state is dynamically evolving over real time
- RT image: an image of the state of such an "object", as obtained through measurements, communicated values/events, ...

Flow Control: Rationale

In any communication system, there is an inherent asymmetry between sender and receiver upon overload:

1. A sender can continue producing messages at its own pace if the receiver is faster than its own pace.
2. A receiver may become overloaded and fail to provide service if sender is faster than its own pace.

The receiver needs an extra instrument: flow control

Explicit Flow Control

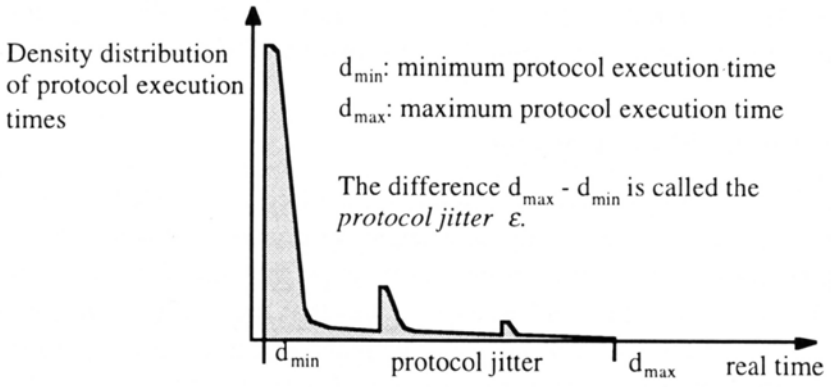
...puts the senders rate of transmission into the control of the receiver, e.g. through

1. explicit control messages being sent back:
   - acknowledgements,
   - start/stop signals inhibiting communication whenever undesired,
   - maskable interrupts
2. third-party-controlled or hidden channels:
   - sender observes RT entities that are in the SoC of the receiver to determine completion of receiver's response to its message.

Explicit Flow Ctrl.: Fault Tolerance

- Error detection and masking is typically implemented by the sender:
  1. Senders waits for acknowledgement (or other causal consequence of message).
  2. When absence of that causal consequence becomes permanent (error detection), it retries communication (an attempt on error concealment). ("becomes permanent" means can be definitely asserted due to maximum latency having expired)
  3. In hard real-time systems, number of retries has to be bounded: Receiver (and further affected parties) are informed.

Retries: Resulting Latency Distribution



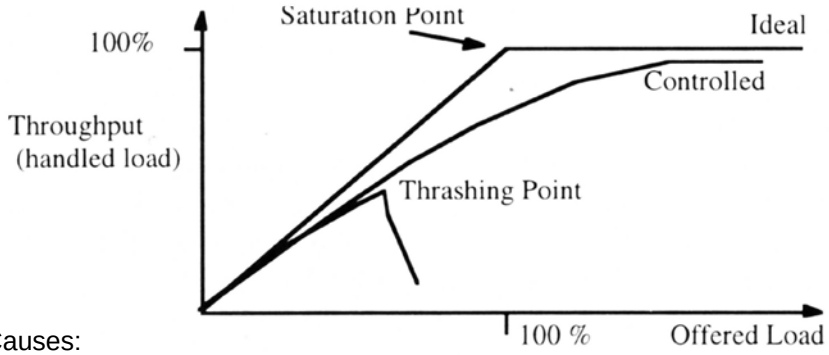
Implicit Flow Control

...subjects message flow to a contract, where obedience to the contract can be locally controlled by sender:

1. A (possibly situation-dependent) a priori rate/temporal pattern is agreed upon
   - at design time, or
   - at startup (example: modems exchanging test-"pings".)
2. Senders commits itself to send only at agreed time instants.
3. No acknowledgements are sent (normally).
4. Error detection is implemented by receiver, which detects non-contractual message streams.

With implicit flow control, communication is typically unidirectional.

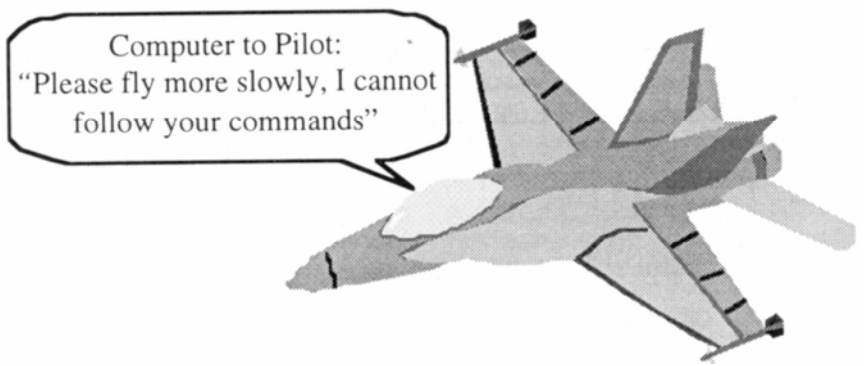
Thrashing ...occurs when throughput is non-monotic in load:



Causes:

- Increasing error rates (e.g. due to higher collision probability on communication network) reduce operational / error recovery ratio.
- Overhead for operating system services (e.g. scheduling) may increase.

Feasibility of Explicit Flow Control



Flow Control Mechanisms vs. Hard Real-Time

Characteristic	Explicit flow ctrl.	Implicit flow ctrl.	Requirements of hard RT
Control signal	Receiver's SoC has to include sender's transmissions	Transmission ctrl. is effected by progression of time	Receiver's ctrl. of sender may be very limited
Error detection	At sender	At receiver	?
Prone to thrashing	Yes	No	Predictability dictates avoidance of thrashing
Multicasting	Intricate and error-prone	Straightforward	Often required

[After H. Kopetz]

- Implicit flow control is often more appropriate.

No free lunch:

- Fundamental tension:
  - Small latency for important nodes/ messages vs.
  - Sufficient and consistent service for less important nodes/messages.
- Measures:
  - Local priority:
    - Each node (or its communication sub-system) can implement a policy of which message(s) shall go to the network when the node gets access
    - Can be static priority/dynamic priority/round robin/...
  - Global priority:
    - The network (and/or the node set) needs to implement policy for which node gets access when
    - Can be static node priority/dynamic node priority/round robin/...

Bus Master Approach to Coordination

Bus master polls for messages:

- Master sends polling messages and waits for response
- Error detection:
  1. unresponsive or slow slave
     - Master uses timeout based on worst-case response time when waiting for response
     - Protocol fails/is retried when timeout expires
  2. faulty master
     - As the master produces some heartbeat on the network, this is detectable
     - Use stand-by techniques (redundant masters)

Bus Master Approach to Coordination

Slave-to-slave communication

- Master polls a willing-to-send slave
- Slave transmits message only when polled
- Potential recipients listen to bus traffic

Examples

MIL-STD-1553B, 1773

Profibus (and many other fieldbusses)

Bus Master Approach to Coordination

Advantages

- Simple to implement, therefore historically very popular
- Bounded latency (if individual message length is constrained)
- Easy to make adaptive versions that adapt to different operational modes of the system

Disadvantages

- Centralized master detrimental to fault tolerance
- Polling consumes bandwidth
- Number of nodes fixed during installation (or additional bandwith required for dynamic detection)

TDMA - Time Division Multiplexed Access

Principle of operation:

1. All (operational) nodes agree on a global time frame via synchronisation
   - Either some master node sends out a frame sync to synchronize clocks (requires FT mechanisms), or
   - nodes use distributed clock synchronization
2. Progress of time is divided into TDMA rounds, within which the individual nodes have private time slots with different phase delay to star t of the round; the slots are non-overlapping
3. Each node transmits only during its private time slot

Examples:

TTP

FlexRay

TDMA - Time Division Multipl. Access

Advantages:

- Deterministic response time
- No polling overhead
- Easy to implement and to analyze

Disadvantages:

- Private slots waste bandwith
- Need for global clock synchronisation
- Number of nodes and their worst-case message lengths need to be fixed a priori
- This leads to either designs using huge safety margins or to complex interference between node performance and TDMA setup (lacking separation of concerns between computation and communication)

Carrier Sense Multiple Access (CSMA)

Operational principle:

- If communication medium is idle then send a message (node decides on its own — no global authority)

Problems:

- Multiple nodes may start almost synchronously, leading to collision on the medium
- Message may be crippled
- Message may be overwritten and thus not delivered
- If message delivery is vital (std. in ES) then collision has to be resolved
  - Collision detection or collision avoidance, arbiting the bus such that at least one of thecolliding messages is delivered uncrippled

CSMA with Detection (CSMA/CD)

Operational principle

- Node waits for an idle channel before transmitting
- Collisions occur if two or more nodes start (almost) simultaneously
- If a collision is detected, the nodes stop transmitting
- After collision, the nodes back off a certain time, which varies from node to node
  - Random assignment of backoff times (worst-case number of retries unbounded), or
  - Systematic assignment, e.g. static assignment a priori (requires known number of nodes at design time)

Examples

Ethernet/IEEE 802.3

LON, CEBus (home networks using power-line modulation)

CSMA with Detection

Advantages:

- Small latency upon low traffic load
- Disruption-free network extension and reduction (with randomized backoff)
- Prioritization is possible by adequate assignment of backoff times (weak variant: probabilistic prioritization via different distributions of random backoff times)
- Extensive installed base and support

Disadvantages:

- Inefficient under heavy loads — prone to thrashing
- Designed for aperiodic traffic — for periodic traffic, the necessity to explicitly tag messages with an identity (redundant in TDMA as the slot position conveys an identity) may cost bandwith
- Collision detection requires hardware

CSMA with Avoidance (CSMA/CA)

Operational principle:

1. Bus arbitration resolves conflicts such that at least one messageis delivered uncrippled, or
2. bus arbitration avoids collisions altogether

Examples:

CAN

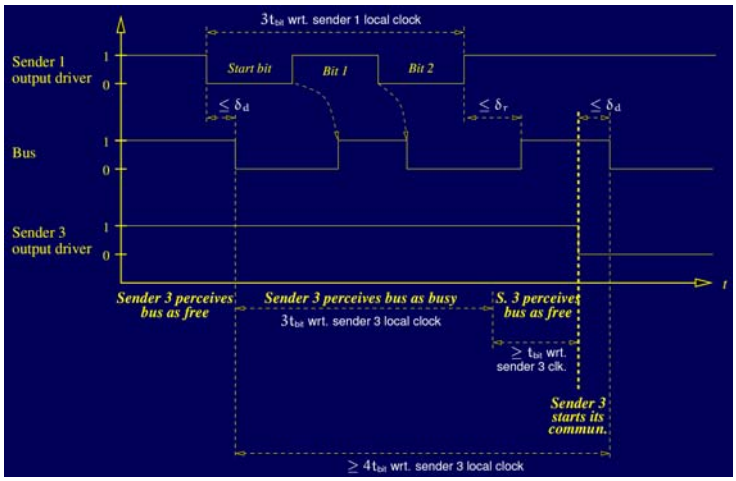
ARINC 629

The CAN bus

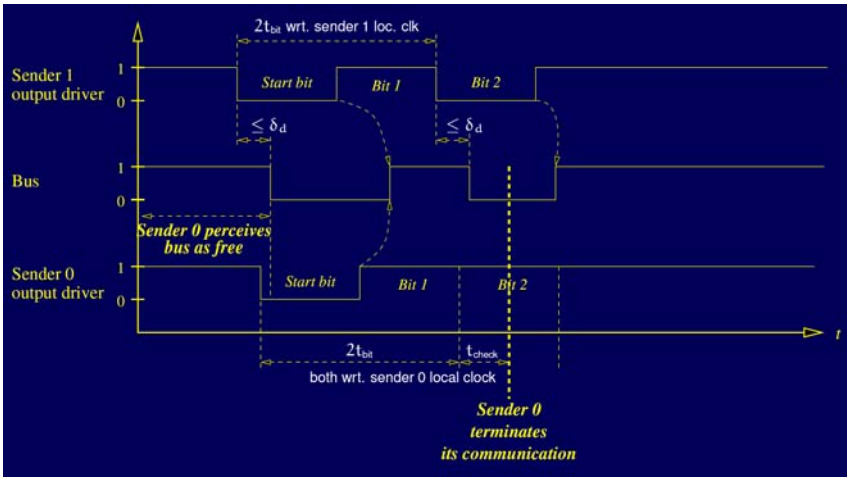
- two-valued bus:
  - “dominant” (denoted 0 in the sequel) and
  - “recessive” (denoted 1 in the sequel)
- if at least one sender sends a dominant bit (= 0) to the bus then the bus will go to “dominant” (= 0) within δ_d time units
- the bus will go to value “recessive” (= 1) if all senders have sent “recessive” for at least δ_d time units.

Length t_{bit} of a bit slot has to be significantly larger than propagation delay along the bus and rise/fall times!

Normal bustake-over (MiniCAN, 2-Bitident.)



Collision handling (MiniCAN)



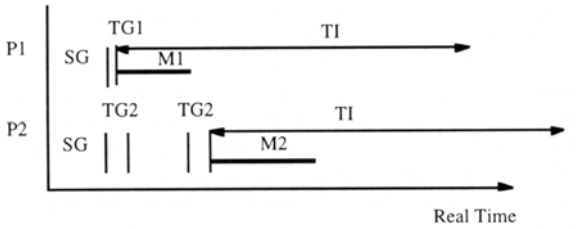
Message priority

- The rule governing early termination of message transfer introduces a priority order between messages: upon a collision, the sender with the lowest identity number wins control over the bus.

ARINC 629

- Based on "waiting room" protocol:
  - Each node is assigned a unique number of minislots that must elapse,with silence on the channel, before transmission
  - Three (groups of) time-out parameters:
    1. SG — synchronization gap controlling access to the waiting room
    2. TGi — terminal gap, the personal time-out of node i
    3. TI — transmit interval preventing monopolization of channel

    TI > SG > max{TGi }



Token passing

Operational principle:

- Token value says which node is transmitting/should transmit next
- Only token holder is allowed to transmit
- Master/slave polling is a special form where token is passed by master and returned to master by slave

Problems:

- Lost token
- duplicated/multiplicated token

Token passing

Advantages:

- Bounded latency
- More adaptive, i.e. higher bandwith under uneven load, than TDMA — node can pass token immediately
- Token passing yields heartbeat supporting fault detection

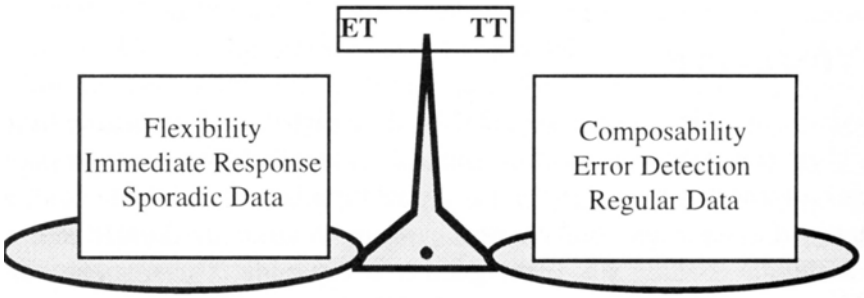
Disadvantages:

- Token passing induces unnecessary latencies under light traffic
- Fault tolerance intricate

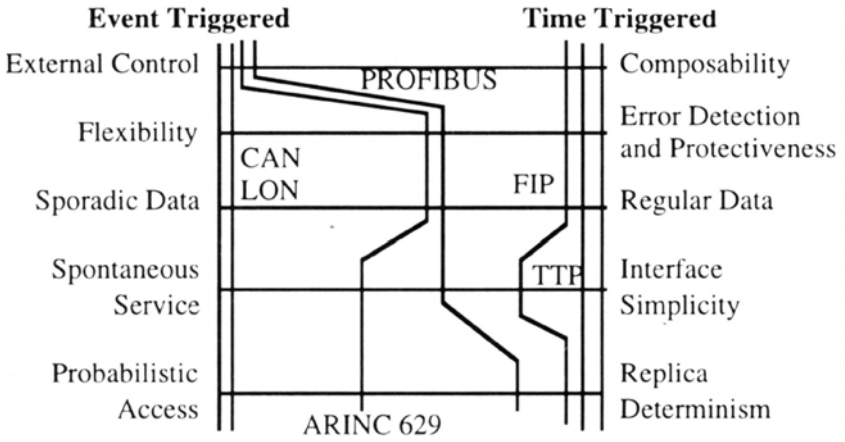
Event-Triggered vs. Time-Triggered

- Event Triggered (ET):
  - Computation/communication triggered by an external event
  - Events are generated by (primarily) state changes in the environment Efficient — only do things when they need to be done; rest and save energy/cpu time/bandwidth/. . . otherwise
  - High peak-load if multiple events happen at once
  - Hard to analyze due to asynchronous nature of events
- Time Triggered (TT):
  - Computation/communication triggered by progress of a system clock
  - Events happen according to a fixed schedule Inefficient — do things periodically, whether needed or not
  - Enhanced analizability due to easily characterizable load, predictable interaction sequences, bus use, etc.

A crude sorting



Evaluation

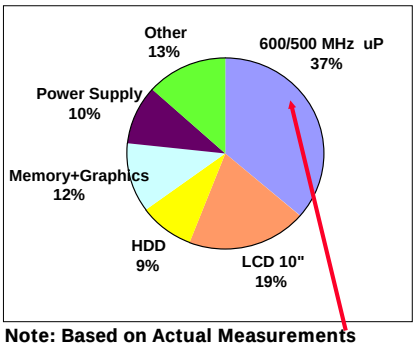


Memory

- For the memory, efficiency is again a concern:
  - speed (latency and throughput); predictable timing
  - energy efficiency
  - size
  - cost
  - other attributes (volatile vs. persistent, etc)

How much of the energy consumption of a system is memory-related?

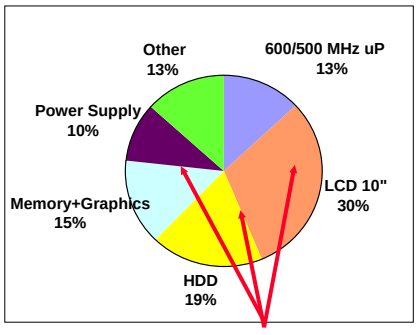
Mobile PC
Thermal Design (TDP) System Power



Note: Based on Actual Measurements

CPU Dominates Thermal Design Power

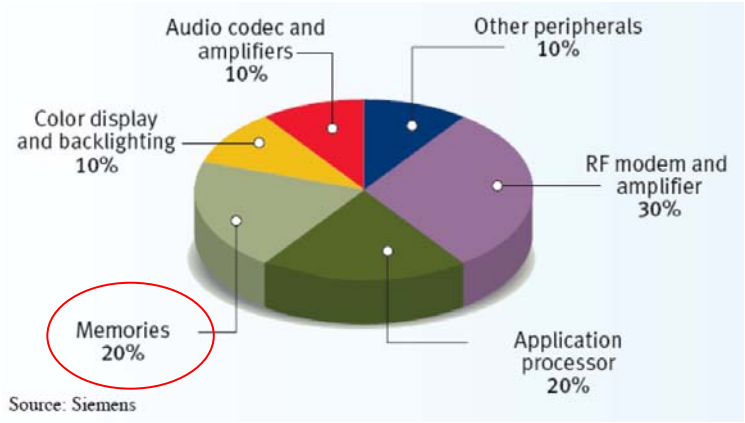
Mobile PC
Average System Power



Multiple Platform Components Comprise Average Power

[Courtesy: N. Dutt; Source: V. Tiwari]

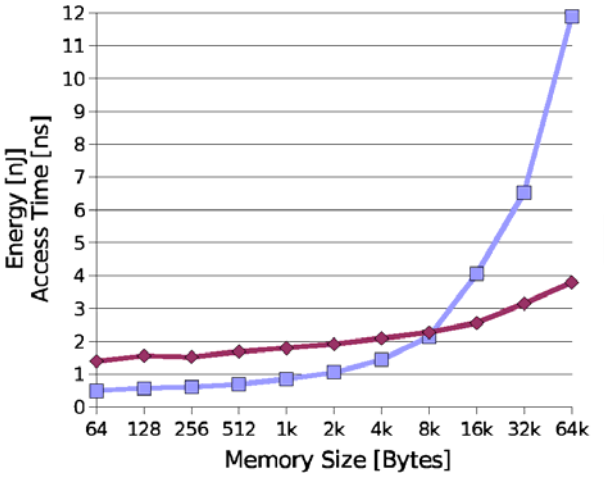
Energy consumption in mobile devices



▪[O. Vargas (Infineon Technologies): Minimum power consumption in mobile-phone memory subsystems; Pennwell Portable Design - September 2005;] Thanks to Thorsten Koch (Nokia/ Univ. Dortmund) for providing this source.

Access times and energy consumption increases with the size of the memory

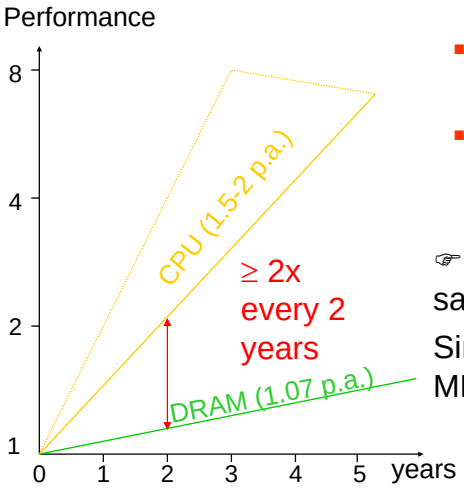
Example (CACTI Model):



"Currently, the size of some applications is doubling every 10 months"
[STMicroelectronics, Medea+ Workshop, Stuttgart, Nov. 2003]

Access-times will be a problem

- Speed gap between processing and main DRAM increases



- early 60ties (Atlas): page fault ~ 2500 instructions
- 2002 (2 GHz µP): access to DRAM ~ 500 instructions

☞ penalty for cache miss about same as for page fault in Atlas

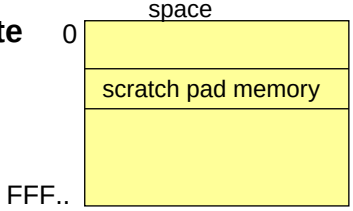
Similar problems for PCs and MPSoCs

[P. Machanik: Approaches to Addressing the Memory Wall, TR Nov. 2002, U. Brisbane]

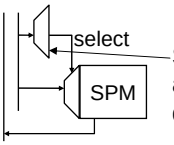
Hierarchical memories using scratch pad memories (SPM)

SPM is a small, physically separate memory mapped into the address space

- Address space

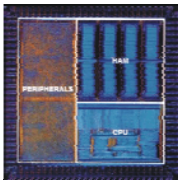


no tag memory



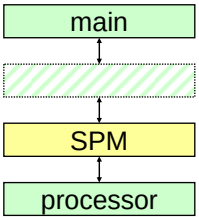
Selection is by an appropriate address decoder (simple!)

Example



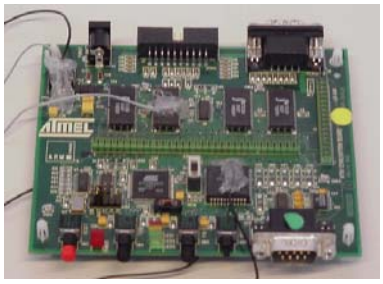
ARM7TDMI cores, well-known for low power consumption

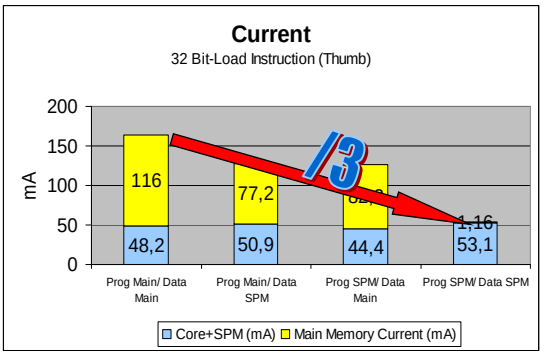
Hierarchy



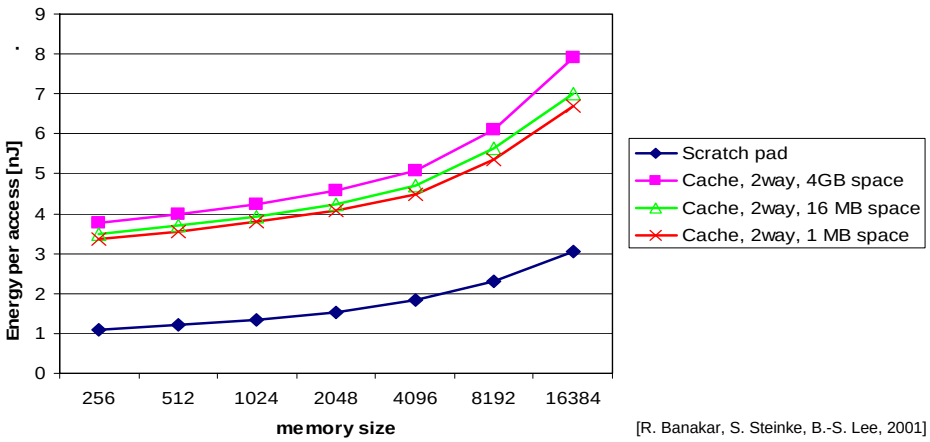
Comparison of currents using measurements

E.g.: ATMEL board with ARM7TDMI and ext. SRAM





Why not just use a cache ?



[R. Banakar, S. Steinke, B.-S. Lee, 2001]