Embedded System Hardware
- Processing -

Peter Marwedel
Informatik 12
TU Dortmund
Germany

2012年 11 月 20 日

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Efficiency:
slide from lecture 1 applied to processing

- CPS & ES must be **efficient**
  - Code-size efficient
    (especially for systems on a chip)
  - Run-time efficient
  - Weight efficient
  - Cost efficient
  - Energy efficient
Key idea of very long instruction word (VLIW) computers (1)

- Instructions included in long instruction packets.
- Instruction packets are assumed to be executed in parallel.
- Fixed association of packet bits with functional units.

- Compiler is assumed to generate these “parallel” packets
Key idea of very long instruction word (VLIW) computers (2)

- 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

• However, possibly low code efficiency, due to many NOPs

• 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 320C6xxx as an example

1 Bit per instruction encodes end of parallel exec.

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B, C, D</td>
</tr>
<tr>
<td>3</td>
<td>E, F, G</td>
</tr>
</tbody>
</table>

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.
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 C6xxx:

Parallel execution cannot span several packets IA64
More encoding flexibility with IA-64 Itanium

3 instructions per **bundle**: 

<table>
<thead>
<tr>
<th>instruc 1</th>
<th>instruc 2</th>
<th>instruc 3</th>
<th>template</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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

The following combinations can be encoded in templates:

- **MII, MMI, MFI, MIB, MMB, MFB, MMF, MBB, BBB, MLX**
  
  with **LX** = *move 64-bit immediate* encoded in 2 slots

*Instruction grouping information*
Templates and instruction types

End of parallel execution called **stops**. Stops are denoted by underscores. Example:

Very restricted placement of stops within bundle. Parallel execution within groups possible. Parallel execution can span several bundles.
Itanium® 9300 (Tukwila), 2010

- 2 G transistors
- 4 cores
- 8-fold hyper-threading/core
- 1.5 GHz at 1.3V

Large # of delay slots, a problem of VLIW processors

- add
- sub
- and
- or
- sub
- mul
- xor
- div
- ld
- st
- mv
- beq

instruction fetch
instruction decode
instruction execute
register writeback
Large # of delay slots, a problem of VLIW processors

- **Delay slots**
- **Pipeline stages**: fetch, decode, execute, writeback
- **Instructions**: add, sub, and, or, sub, mult, xor, div, ld, st, mv, beq
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 (some expression involving condition code regs)
- instruction I

\[ c = \text{true} \quad \Rightarrow \quad \text{I executed} \]
\[ c = \text{false} \quad \Rightarrow \quad \text{NOP} \]
Predicated execution: Implementing IF-statements “branch-free“: TI C6xxx

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
Energy efficiency reached with VLIW processors

"inherent power efficiency of silicon"

© Silicon Hive
41 Issue VLIW for SDR
130 nm, 1,2 V, 6,5mm², 16 bit
30 operations/cycle (OFDM)
150 MHz, 190 mW (incl. SRAMs)
24 GOPs/W, ~1/5 IPE

© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, Keynote Slides, ACACES, 2007
Microcontrollers
- MHS 80C51 as an example -

- 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
Trend: multiprocessor systems-on-a-chip (MPSoCs)

3G Multi-Media Cellular Phone System

Previous System

New System Using G1

HPA
High Power Amplifier

Baseband Processor

Application Processor

Multi-Media Accelerator

RFIC
Radio Frequency IC

HPA
High Power Amplifier

RFIC
Radio Frequency IC

One Chip
SH-MobileG1

Embedded System Hardware
- Reconfigurable Hardware -

Peter Marwedel
Informatik 12
TU Dortmund
Germany

2012年 11 月 20 日

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Energy Efficiency of FPGAs

© Hugo De Man, IMEC, Philips, 2007
Reconfigurable Logic

Custom HW may be too expensive, SW 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:
- algorithms like de/encryption,
- pattern matching in bioinformatics,
- high speed event filtering (high energy physics),
- high speed special purpose hardware.

Very popular devices from
- XILINX, Actel, Altera and others
Floor-plan of VIRTEX II FPGAs
## Interconnect for Virtex II

<table>
<thead>
<tr>
<th>Hierarchical Routing Resources;</th>
<th>More recent: Virtex 5, 6, 7 no routing plan found for Virtex 7.</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Horizontal Long Lines 24 Vertical Long Lines</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>120 Horizontal Hex Lines 120 Vertical Hex Lines</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>40 Horizontal Double Lines 40 Vertical Double Lines</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>16 Direct Connections (total in all four directions)</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>8 Fast Connects</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Virtex 7 Configurable Logic Block (CLB)

http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf
Memories typically used as look-up tables to implement any Boolean function of $\leq 6$ variables.

Processors typically implemented as “soft cores” (microblaze)
Virtex 7 SliceM

SliceM supports using memories for storing data and as shift registers

http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

© Xilinx
## Resources available in Virtex 7 devices (max)

<table>
<thead>
<tr>
<th>Device</th>
<th>XC7V2000T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic cells</td>
<td>1,954,560</td>
</tr>
<tr>
<td>Slices</td>
<td>305,400</td>
</tr>
<tr>
<td>Max distributed RAM [Kb]</td>
<td>21,550</td>
</tr>
<tr>
<td>DSP slices</td>
<td>2,160</td>
</tr>
<tr>
<td>Block RAM blocks 18 Kb</td>
<td>2,584</td>
</tr>
<tr>
<td>Block RAM blocks 36 Kb</td>
<td>1,292</td>
</tr>
<tr>
<td>Block RAM blocks Max [Kb]</td>
<td>46,512</td>
</tr>
<tr>
<td>PCIe</td>
<td>4</td>
</tr>
<tr>
<td>GTX Transceivers</td>
<td>36</td>
</tr>
<tr>
<td>A/D converters</td>
<td>1</td>
</tr>
<tr>
<td>User I/O</td>
<td>1,200</td>
</tr>
</tbody>
</table>

## Virtex-7 FPGAs

<table>
<thead>
<tr>
<th>Maximum Capability</th>
<th>Virtex-7 T</th>
<th>Virtex-7 XT</th>
<th>Virtex-7 HT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic density</td>
<td>1,995 k</td>
<td>1,139 k</td>
<td>864 k</td>
</tr>
<tr>
<td>Peak transceiver speed</td>
<td>12.5 Gb/s (GTX)</td>
<td>13.1 Gb/s (GTH)</td>
<td>28.05 Gb/s (GTZ)</td>
</tr>
<tr>
<td>Peak bi-directional bandwidth</td>
<td>0.9 Tb/s</td>
<td>2.515 Tb/s</td>
<td>2.784 Tb/s</td>
</tr>
<tr>
<td>DSP throughput (symmetric filter)</td>
<td>2,756 G MACS</td>
<td>5,314 GMACS</td>
<td>5,053 GMACS</td>
</tr>
<tr>
<td>Block RAM</td>
<td>46.5 Mb</td>
<td>85 Mb</td>
<td>64.4 Mb</td>
</tr>
<tr>
<td>I/O pins</td>
<td>1,200</td>
<td>1,100</td>
<td>700</td>
</tr>
</tbody>
</table>

Memory

2012年 11 月 20 日

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Memory

For the memory, efficiency is again a concern:

- capacity
- energy efficiency
- speed (latency and throughput); predictable timing
- size
- cost
- other attributes (volatile vs. persistent, etc)
Memory capacities expected to keep increasing
Where is the power consumed?
- Stationary systems -


- Switching power, logic dominating
- Overall power consumption a nightmare for environmentalists
Where is the power consumed?
- Consumer portable systems -

- Based on current trends
  - Memory and logic, static and dynamic relevant
  - Following current trends will violate maximum power constraint (0.5-1 W).
Memory energy significant even if we take display and RF of mobile device into account

-[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.
Energy consumption and access times of memories

Example CACTI: Scratchpad (SRAM) vs. DRAM (DDR2):

16 bit read; size in bytes;
65 nm for SRAM, 80 nm for DRAM

Source: Olivera Jovanovic, TU Dortmund, 2011
Access times and energy consumption for multi-ported register files

- Cycle Time (ns)
- Area ($\lambda^2 \times 10^6$)
- Power (W)

Source and © H. Valero, 2001
Trends for the Speeds

Speed gap between processor and main DRAM increases

Similar problems also for embedded systems & MPSoCs

Memory access times >> processor cycle times

“Memory wall” problem

[Andrea Manzoni: Approaches to Addressing the Memory Wall, TR Nov. 2002, U. Brisbane]
However, clock speed increases have come to a halt

The speed gap does not disappear – may be, it is not getting larger any more.
Set-associative cache $n$-way cache

$|\text{Set}| = 2$

Address

Tag | Index
--- | ---
way 0 | $\$(€)$ | way 1

Tags | data block | Tags | data block

$|\text{Set}| = 2$

Data
Hierarchical memories using scratch pad memories (SPM)

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

Address space

0

scratch pad memory

FFF..

no tag memory

Examples:

- Most ARM cores allow tightly coupled memories
- IBM Cell
- Infineon TriCore
- Many multi-cores, due to high costs of coherent caches

Hierarchy

main

SPM

processor

Selection is by an appropriate address decoder (simple!)
Comparison of currents using measurements

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

![ATMEL board with ARM7TDMI and ext. SRAM](image)

**Current**
32 Bit-Load Instruction (Thumb)

<table>
<thead>
<tr>
<th></th>
<th>Main Memory Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog Main/ Data</td>
<td>116</td>
</tr>
<tr>
<td>Prog SPM/ Data</td>
<td>77,2</td>
</tr>
<tr>
<td>Prog Main/ Data</td>
<td>44,4</td>
</tr>
<tr>
<td>Prog SPM/ Data SPM</td>
<td>1,16</td>
</tr>
</tbody>
</table>

- Core+SPM (mA)
Why not just use a cache?

2. Energy for parallel access of sets, in comparators, muxes.

![Graph showing energy per access vs memory size]

[Scratch pad, Cache, 2way, 4GB space, Cache, 2way, 16 MB space, Cache, 2way, 1 MB space]

[R. Banakar, S. Steinke, B.-S. Lee, 2001]
Influence of the associativity

Parameters different from previous slides

[P. Marwedel et al., ASPDAC, 2004]
Summary

- Processing
  - VLIW/EPIC processors
  - MPSoCs
- FPGAs
- Memories
  - “Small is beautiful”
    (in terms of energy consumption, access times, size)
Instruction types are mapped to functional unit types

There are 4 functional unit (FU) types:
- M: Memory Unit
- I: Integer Unit
- F: Floating-Point Unit
- B: Branch Unit

Instruction types $\rightarrow$ corresponding FU type, except type A (mapping to either I or M-functional units).
Nevertheless, (parallel) performance keeps increasing.
Philips TriMedia-Processor

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

http://www.nxp.com/acrobat/datasheets/PNX15XX_SER_N_3.pdf (incompatible with firefox?) © NXP
Similar information according to other sources

Strong ARM

IEEE Journal of SSC
Nov. 96

ICache 26%

DCache 16%

DMMU 8%

IMMU 9%

Ibox 18%

EBox 8%

Clock 10%

Others 5%

[Based on slide by and ©: Osman S. Unsal, Israel Koren, C. Mani Krishna, Csaba Andras Moritz, U. of Massachusetts, Amherst, 2001]

SysCtl 3%

BIU 8%

PATag 2%

RAM 1%

arm9 25%

I MMU 4%

D MMU 5%

I Cache 25%

D Cache 19%

Clocks 4%

Other 4%
Multiprocessor systems-on-a-chip (MPSoCs) (2)

SH-MobileG1: Chip Overview

<table>
<thead>
<tr>
<th>Die size</th>
<th>11.15mm x 11.15mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>90nm LP 8M(7Cu+1Al) CMOS dual-Vth</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>1.2V(internal), 1.8/2.5/3.3V(I/O)</td>
</tr>
<tr>
<td># of TRs, gate, memory</td>
<td>181M TRs, 13.5M Gate 20.2 Mbit mem</td>
</tr>
</tbody>
</table>
TriMedia-Processor

Lastest version: starting 5-8 operations per cycle

http://www.tridentmicro.com/producttree/stb/media-processor/pnx100x/
© Trident, 2010
(since 2012/1 under chapter 11)