ELEC6234
Embedded Processor Synthesis

2. RISC architecture in embedded applications

Tom Kazmierski (tjk@ecs.soton.ac.uk)
The number of different embedded processors is growing.

There are many embedded processors on the market because there is a need for many different embedded processor architectures.

Intel dominates the desktop only because all computers are more or less the same. One processor can serve them all. That's not true of embedded systems at all.

Many processors, both CISC and RISC, which started out as high-end computer processors have been used in embedded applications: MIPS, 68K, SPARC, ARM, PowerPC.

- Interestingly, they are all failed desktop processors and none of them started out as an embedded processor.
No Instruction Set Computing – a variant of RISC

NISC is a derivative of RISC where the instruction decoder has been removed and instruction code replaced with a full-length control word.

A high-level synthesis compiler can perform the scheduling and binding to generate a NISC structure.

The main pro NISC argument is that giving low-level control to the compiler enables better utilization of datapath resources, which can result in better performance.

Main benefits of NISC technology are:
• Simpler controller: no hardware scheduler, no instruction decoder
• Better performance: more flexible architecture, better resource utilization
• Easier to design: no need for designing instruction-sets
Sample NISC architecture

NISC (No Instruction Set Computer) Architecture

- Control
- Address
- Data

PC control

PAUSE
ALU flags

PC

[+1]

Next CW (branch)

Control Word Memory

Reg selection

Reg File

Wdata

ALU

Result

[\text{n}]

MUL

Result

[\text{n}]

Bus A

Bus B

External Input 1

Buf

External Input 2

Buf

Out

Control signals

External Input 1

Buf

External Input 2

Buf

ALU flags

External Input 1

Buf

External Input 2

Buf

ALU flags
Sample RISC (picoMIPS) for comparison

pMIPS version 1 – no RAM, no branches
No Instruction Set Computing – further advantages

In processor design, the instruction set and controller are the most tedious and time-consuming parts. By eliminating these two, design of custom processing elements becomes significantly easier.

The datapath of NISC processors can even be generated automatically for a given application as high-level synthesis tools can be used for this purpose.

NISC technology is comparable to high level synthesis (HLS) or ”C to HDL” synthesis.

NISC bridges the two approaches; custom RISC design and HLS.
RISC

• RISC is a microprocessor that is designed to perform a smaller number computer instructions so that it can operate at a higher speed. It stands for - Reduced Instruction Set Computer

• John Cocke of IBM Research in Yorktown, New York, noticed that about 20% of the instructions in a computer did 80% of the work.

• The RISC architecture was pioneered by David Patterson from the University of California in Berkeley.
  – Patterson led the development of a new microprocessor which later became SPARC

• The SPARC architecture was adopted in Sun Microsystems' processors and led to the founding of what later became MIPS Technologies, part of Silicon Graphics.

• SPARC is still marketed by Fujitsu
• MIPS is now one of the main architectures used in embedded designs
Pipelining: a key RISC technique

• Pipelining is a design technique where the computer's hardware processes more than one instruction at a time, and doesn't wait for one instruction to complete before starting the next.

• RISC machine has the same four stages of instruction execution as in our typical CISC machine: fetch, decode, execute, and write. But these stages are executed in parallel. As soon as one stage completes, it passes on the result to the next stage and then begins working on another instruction.

• In a typical pipelined RISC design, each instruction takes 1 clock cycle for each stage, so the processor can accept 1 new instruction per clock.
RISC characteristics

- **Small instruction set.**
  
  In a RISC machine, the instruction set contains simple, basic instructions; more complex operations are performed using sequences of simple instructions.

- **Fixed length instructions.**
  
  Each instruction is the same length, so that it may be fetched in a single operation.

- **1 clock-cycle instructions.**
  
  Most instructions complete in one clock cycle. This is achieved through pipelining which allows the processor to handle several instructions at the same time. Pipelining is a key technique used in RISC machines to improve performance.
Pipelining issues

- Pipelining enhances performance
- Scenario 1, a two-stage pipeline:
  - Fetch: retrieve instruction from memory
  - Execute: perform an ALU operation with register data
Pipelining issues

- Scenario 2, a three-stage pipeline for load/store
  - Fetch: retrieve instruction from memory
  - Execute: calculate memory address
  - Read/Write: transfer data from/to memory to/from CPU

\[
\begin{array}{ccc}
\text{n: load A} & \text{fetch n} & \text{exec n} & \text{r/w n} \\
\text{n+1: load B} & \text{fetch n+1} & \text{exec n+1} & \text{r/w n+1} \\
\text{n+2: add A+B} & \text{fetch n+2} & \text{STALL !} & \text{exec n+2} \\
\text{n+3: store C} & \text{fetch n+3} & \text{STALL !} & \text{exec n+3} & \text{r/w n+3}
\end{array}
\]
Pipelining issues

• Branching and pipelines
  – what to do with pre-fetched instructions if there is a branch?

```
<table>
<thead>
<tr>
<th>Time</th>
<th>n:</th>
<th>n+1:</th>
<th>n+2:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>add</td>
<td>jump</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fetch n</td>
<td>exec n</td>
<td>fetch n+1</td>
</tr>
<tr>
<td></td>
<td>fetch n+2</td>
<td>???</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fetch m</td>
<td>exec m</td>
<td>fetch m+1</td>
</tr>
</tbody>
</table>
```
Pipelining issues: delayed branch

• Branching and pipelines
  – Pipeline optimisation: execute the instruction after the branch to increase pipeline efficiency
  – In the example below ‘sub’ executes before ‘branch’!

n: add
n+1: branch to m
n+2: sub

m: …

branch: PC := m
instruction n+1 executes after n+2!

No stall cycle
Main RISC advantage for embedded applications

• Simpler hardware.

Because the instruction set of a RISC processor is so simple, it uses up much less chip space; extra functions, such as memory management units or floating point arithmetic units, can also be placed on the same chip.

Smaller chips allow a semiconductor manufacturer to place more parts on a single silicon wafer, which can lower the per-chip cost dramatically.
Other RISC advantages

• **Shorter design cycle.**

Since RISC processors are simpler than equivalent CISC processors, they can be designed more quickly, and can take advantage of other technological developments sooner than corresponding CISC designs, leading to greater leaps in performance between generations.

• **Low energy consumption**

Energy has become the primary performance characteristic in embedded designs, especially those aimed at mobile consumer markets. Here RISC processors have a clear advantage over CISC designs: simpler hardware, smaller control logic are features that naturally lead to low energy consumption.
RISC disadvantages

• Code Quality

The performance of a RISC processor depends greatly on the quality of the code that it is executing.

RISC processors are harder to program efficiently than their CISC equivalents. If the instruction scheduling in a program is poor, the processor can spend quite a bit of time stalling: waiting for the result of one instruction before it can proceed with a subsequent instruction.

Instruction scheduling rules can be complicated in RISC processors, hence the performance of a RISC application depends critically on the quality of the code generated by the compiler.

• System Design

They require more instructions, and hence memory, than CISCs to implement applications.

RISC processors require very fast memory systems to feed them instructions. RISC-based systems typically contain large memory caches, usually on the chip itself. This is known as a first-level cache.
Why is CISC still around?

• Why are there still CISC CPUs being developed?
• Why is Intel spending time and money to manufacture new versions of the architecture which started in 1970s?

• **Answer**

The short answer is: **backward compatibility**. The IBM compatible PC is the most common computer in the world. Intel wanted a CPU that would run all the applications that are in the hands of more than 100 million users.
CISC-RISC convergence

Much of the RISC philosophy has been adopted in recent CISC architectures: e.g. pipelines, branch prediction, hardware stacks.

Recent RISC developments, such as high degree of pipelining and parallelism, do not necessarily lead to simple control structures.

Modern RISCs and CISCs do not tend to have vastly different clock speeds.

The real performance issue seems to lie not in architecture but in optimization of instruction sets and details of machine organization.
Today, most CISC processors are based on hybrid CISC-RISC architecture.

These designs use a decoder to convert CISC instructions into RISC instructions before execution. They are then processed by a RISC core, which performs a few basic instructions very quickly.

Having a RISC core is advantageous because it allows performance enhancing features, such as pipelining and branch prediction.

Popular examples of hybrid designs include the Intel Core and AMD family of processors. These processors are compatible with software written for their CISC predecessors yet perform competitively against processors based on RISC designs.
RISC has become the primary choice of architecture for embedded processor designs
SPARC

- SPARC is best known as the processor used in Sun workstations
- SPARC was one of the first RISC designs
- In the early 1990s, embedded SPARC designs were common. Now this pioneering architecture is almost nonexistent in the embedded market.
- 20 years ago there about ten companies making SPARC processors, all different. Sun Microsystems was the biggest customer for them, so following the demise of Sun Microsystems in 2010, almost all of the SPARC makers went out of business.
- Texas Instruments and Fujitsu are the only major SPARC chip developers left.
- OpenSPARC soft core versions exist
Intel i960

• The i960 was once the best-selling RISC architecture in the world.
• In the early '90s there was an i960 processor in almost every laser printer or network router made. The i960 was particularly popular in HP's LaserJet series of printers, just as LaserJet sales took off.
• The i960 it was originally designed to power workstations. It came out of a joint venture between Intel and Siemens called BiiN. BiiN was supposed to develop fault-tolerant Unix workstations
• Intel gained control of the processor it developed with Siemens.
• The i960 was expensive, slow, and very power-hungry. The processor also had complex fault-tolerant features that made it difficult to manufacture and debug and had no (apparent) use outside of the workstation market.
• Now the i960 processors faded away/
MIPS

• MIPS is a prime example of a high-end computer architecture that is more successful in embedded application than it ever was in engineering workstations.
• The acronym stands for “Microprocessor without Interlocked Pipeline Stages”
• MIPS, the company, originally acquired by Silicon Graphics (SGI) in the 1990s started using MIPS processors in all its workstations.
• MIPS/SGI made chips for Nintendo, the Japanese game maker who wanted to use a MIPS processor in its upcoming N64 video game. This turned out to be MIPS' biggest deal ever. The company got two-thirds of its money from Nintendo throughout the late 1990s.
• Now various flavours of the MIPS architecture are very popular in embedded applications.
MIPS...

• Although MIPS doesn't dominate the home video-game market like it once did, the architecture has comfortably settled into the number two RISC position.

• MIPS has extended its family of processors both at the high end, with its monstrous 64-bit 20Kc family, and at the low end, with SmartMIPS, a minimal 32-bit design for smart cards and other ultra-low-power systems.

• There's probably no other CPU family that reaches so high and so low while remaining software compatible throughout the line.
PowerPC

• PowerPC was created in 1996 and existed in both 32-bit and 64-bit implementations.

• It was used by Apple in Macintosh desktops.

• Within two years of its inception, there were more PowerPC chips being sold in embedded applications than in computers (such as Macintosh), making PowerPC "officially" an embedded processor.

• Now PowerPC is a marginal player, selling more than SPARC but less than most other 32-bit RISC designs.

• Numerically, the PowerPC is most found in controllers in cars.

• A version of PowerPC was used in many Cisco network routers in the late 1990s.
**ARM**

• ARM (Advanced RISC Machines) was also originally a computer processor. Now ARM is the **most popular 32-bit embedded design** in the world.

• ARM is a UK company which was originally called Acorn. Its BBC Micro computer of the 1980s was the UK equivalent to first personal computers such as Apple or Commodore 64 in the US. The BBC Micro was the first commercial deployment of RISC technology.

• The BBC Micro became obsolete by mid 1980s and the personal PC market was overshadowed by the IBM PC, but the ARM processor design lived on.

• In the last two decades ARM's biggest sales have been in a number of **mobile phones**

• ARM's simple design gives it small silicon footprint, which, in turn, gives it modest power consumption. Its low power combined with its ability to be embedded into high-volume ASICs was key in the ARM’s success in mobile phones.
Summary

- Embedded processor technology finds use in the Application-Specific Product (ASP) applications, as well as in customer-specific product designs.

- The most common embedded processor core today is the ARM architecture.

- Other major embedded processor architectures include MIPS, SPARC, and PowerPC,

- Dedicated, new application specific processor designs become popular
  - Mainly due to low energy requirements