EMBEDDED PROCESSORS

Instruction-Level Parallelism may be implemented as:

- **CISC instructions**
- **subword parallelism**
- **superscalar**
- **VLIW**

EMBEDDED PROCESSORS

- A processor with complex (and typically, rather specialized) instructions is called a **CISC machine** (complex instruction set computer).
- The philosophy behind such processors is **distinctly different from that of RISC machines** (reduced instruction set computers)
- **DSPs are typically CISC machines**, and include instructions specifically **supporting FIR filtering,** and often other algorithms such as FFTs (fast Fourier transforms) and Viterbi decoding
- In fact, **to qualify as a DSP, a processor must be able to perform FIR filtering in one instruction cycle per tap**

EMBEDDED PROCESSORS

The Texas Instruments TMS320c54x family of DSP processors is intended to be used in power-constrained embedded applications that demand high signal processing performance, such as wireless communication systems and personal digital assistants (PDAs). The inner loop of an FIR computation is :

1. **`RPT numberOfTaps - 1`** (zero-overhead loop)
2. **`MAC *AR2+, *AR3+, A`**

$$a := a + x * y$$

- Registers `AR2` and `AR3` may be set up to implement circular buffers
- The **c54x processor includes** a section of **on-chip memory** that **supports two accesses in a single cycle**, and as long as the addresses refer to this section of the memory, the MAC instruction will execute in a single cycle
- Each cycle, the processor performs **two memory fetches**, **one multiplication**, **one addition**, and **two** (possibly modulo) **address increments**

EMBEDDED PROCESSORS

If the coefficients of the FIR filter are symmetric:

- N is even and $a_i = a_{N-i-1}$
- The number of multiplications can be reduced by rewriting the formula as:

$$y(n) = \sum_{i=0}^{(N/2)-1} a_i(x(n-i) + x(n-N+i+1))$$

- The Texas Instruments TMS320c54x instruction set includes a FIRS instruction that functions similarly to the MAC but using this calculation
- This takes advantage of the fact that the c54x has two ALUs, and hence can do twice as many additions as multiplications
- The time to execute an FIR filter now reduces to 1/2 cycle per tap

EMBEDDED PROCESSORS

Subword parallelism

- Many embedded applications operate on data types that are considerably smaller than the word size of the processor (e.g. RBG data)
- A wide ALU is divided into narrower slices enabling simultaneous arithmetic or logical operations on smaller words
- **Intel introduced subword parallelism into the** widely used general purpose **Pentium processor and called the technology MMX**
- **Similar techniques were introduced by Sun Microsystems** for **Sparc** processors and **by Hewlett Packard for** the **PA RISC** processor
- **Many processor architectures** designed **for embedded applications**, including many DSP processors, **also support subword parallelism**
- A **vector processor** is one where the instruction set includes operations on multiple data elements simultaneously
- **Subword parallelism is a particular form of vector processing**

EMBEDDED PROCESSORS

Superscalar processors

- Use fairly conventional sequential instruction sets, but the hardware can simultaneously dispatch multiple instructions to distinct hardware units when it detects that such simultaneous dispatch will not change the behavior of the program.
- The execution of the program is identical to what it would have been if it had been executed in sequence.
- Such processors even support out-of-order execution, where instructions later in the stream are executed before earlier instructions.
- Superscalar processors **have a significant disadvantage for embedded systems**, which is that **execution times may be extremely difficult to predict**, and in the context of multitasking (interrupts and threads), **may not even be repeatable**
- The execution times may be very **sensitive to** the exact **timing of interrupts**, in that **small variations** in such timing **may have big effects on** the **execution times** of programs.

EMBEDDED PROCESSORS

Very Long Instruction Word (VLIW) architectures

- **Embedded Processors** often use **VLIW architectures** instead of superscalar in order to get more repeatable and predictable timing
- **VLIW processors** include **multiple function units**, like superscalar processors but instead of dynamically determining which instructions can be executed simultaneously, **each instruction specifies what each function unit should do in a particular cycle**
- A **VLIW instruction set combines multiple independent operations** into a single instruction
- **Like superscalar** architectures, these **multiple operations are executed simultaneously on distinct hardware**
- **Unlike superscalar**, however, **the order and simultaneity** of the execution **is fixed in the program, not decided on-the-fly**. It is up to **the programmer** (working at assembly language level) **or the compiler** to **ensure that the simultaneous operations are indeed independent**. In exchange for this additional complexity in programming, **execution times become repeatable and (often) predictable**

EMBEDDED PROCESSORS

Very Long Instruction Word (VLIW) architectures

The Texas Instruments TMS320c55x, the next generation beyond the c54x, includes two multiply-accumulate units, and can support instructions that look like this:

```
1. MAC *AR2+, *CDP+, AC0

2. ::MAC *AR3+, *CDP+, AC1
```

`AC0` and `AC1` are two accumulator registers and `CDP` is a specialized register for pointing to filter coefficients. The notation `::` means that these two instructions should be issued and executed in the same cycle. It is up to the programmer or compiler to determine whether these instructions can in fact be executed simultaneously. Assuming the memory addresses are such that the fetches can occur simultaneously, these two MAC instructions execute in a single cycle, effectively dividing in half the time required to execute an FIR filter.

EMBEDDED PROCESSORS

Multicore Architectures

- A multicore machine is a combination of **several processors on a single chip**
- Although multicore machines have existed since the early 1990s, they have **only recently penetrated into general-purpose computing**
- This penetration accounts for much of the interest in them today
- **Heterogeneous multicore machines combine a variety of processor types on a single chip**, vs. multiple instances of the same processor type
- For embedded applications, **multicore architectures have a significant potential advantage over single-core architectures because real-time and safety-critical tasks can have a dedicated processor**
- This is the reason for the **heterogeneous architectures used for cell phones**, since the **radio and speech processing functions are hard real-time functions** with considerable computational load. **In such architectures, user applications cannot interfere with real-time functions**

EMBEDDED PROCESSORS

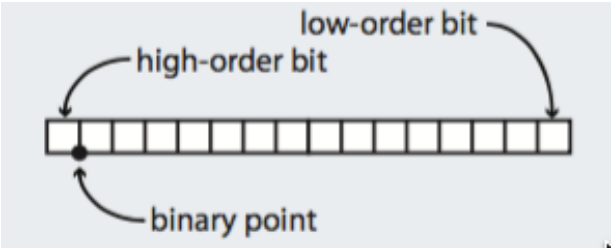
Multicore Architectures

- This lack of interference is **more problematic in general-purpose multicore architectures**
- It is common, for example, to use **multi-level caches**, where **the second or higher level cache is shared among the cores**
- Unfortunately, such sharing makes it very difficult to isolate the real-time behavior of the programs on separate cores, since **each program can trigger cache misses in another core**. Such multi-level caches are **not suitable for real-time applications**.
- A very **different type of multicore architecture** that is sometimes used in embedded applications **uses one or more soft cores** together with custom hardware on a field-programmable gate array (FPGA).
- **FPGAs are chips whose hardware function is programmable using hardware design tools**. **Soft cores are processors implemented on FPGAs**. The advantage of soft cores is that **they can be tightly coupled to custom hardware** more easily than off-the-shelf processors.

EMBEDDED PROCESSORS

Fixed-Point Numbers

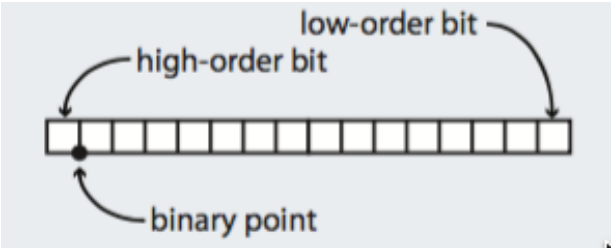
- Many embedded processors provide hardware for integer arithmetic only.
- Integer arithmetic can be used for non-whole numbers, with some care.
- Given, say, a 16-bit integer, a programmer can imagine a binary point, which is like a decimal point, except that it separates bits rather than digits of the number.
- For example, a 16-bit integer can be used to represent numbers in the range −1.0 to 1.0 (roughly) by placing a (conceptual) binary point just below the high-order bit of the number.



EMBEDDED PROCESSORS

Fixed-Point Numbers

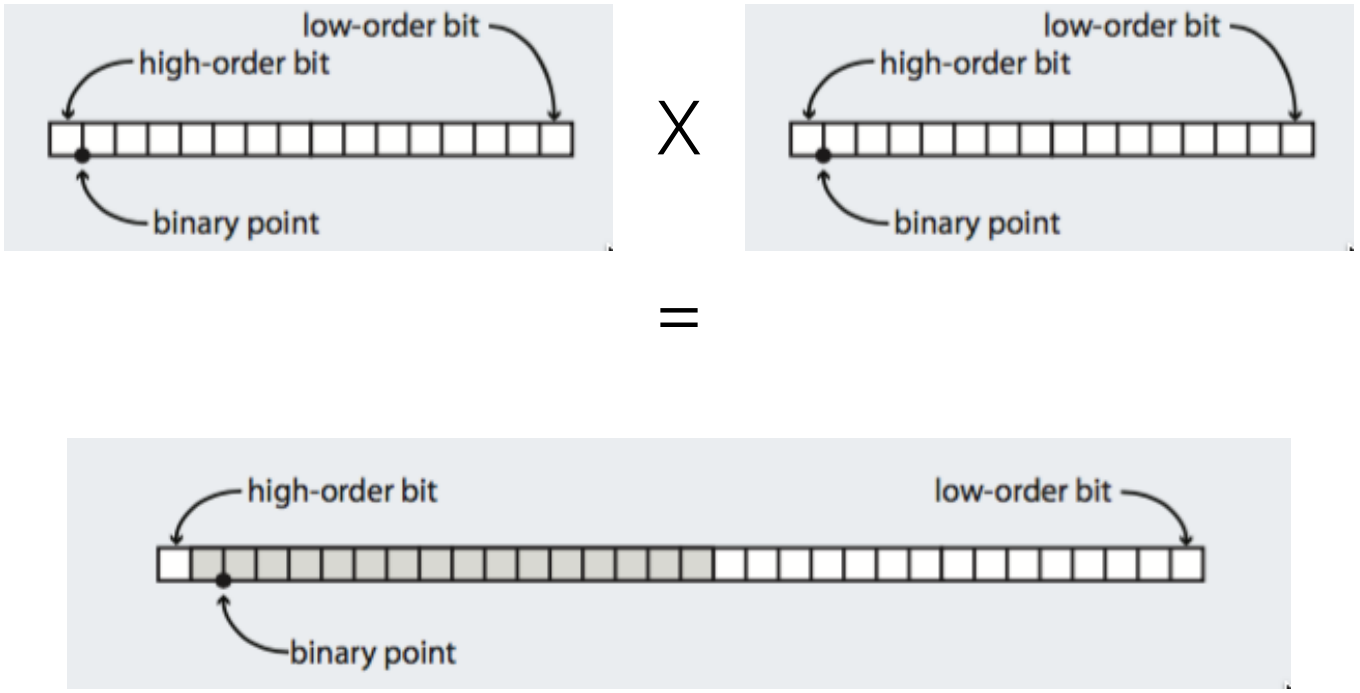
- Without the binary point, a number represented by the 16 bits is a whole number $x \in \{-2^{15}, \ldots, 2^{15} - 1\}$ (two's-complement).
- With the binary point, we interpret the 16 bits to represent a number $y = x/2^{15}$. Hence, y ranges from -1 to $1 - 2^{-15}$. This is known as a fixed-point number.
- The format of this fixed-point number can be written 1.15, indicating that there is one bit to the left of the binary point and 15 to the right. When two such numbers are multiplied at full precision, the result is a 32-bit number.



EMBEDDED PROCESSORS

Fixed-Point Numbers

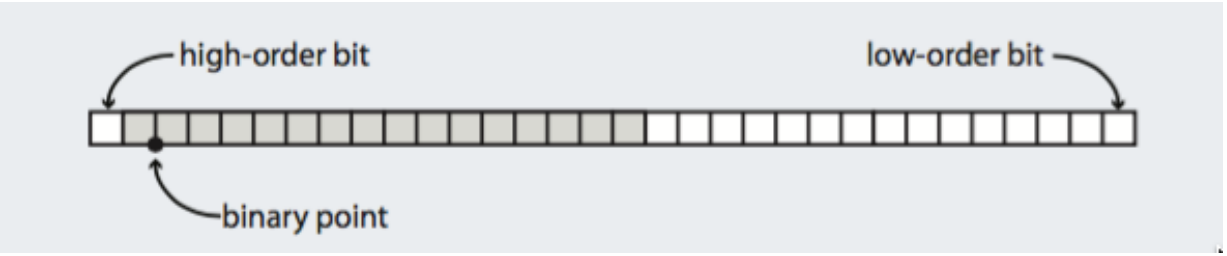
- The format of this fixed-point number can be written 1.15, indicating that there is one bit to the left of the binary point and 15 to the right. When two such numbers are multiplied at full precision, the result is a 32-bit number.



EMBEDDED PROCESSORS

Fixed-Point Numbers

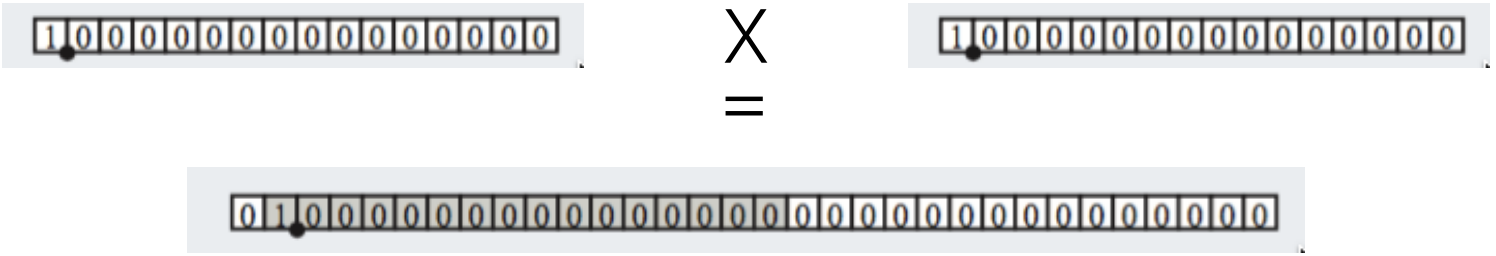
- The location of the binary point follows from the law of conservation of bits
- When multiplying two numbers with formats n.m and p.q, the result has format (n+p).(m+q)
- Processors often support such full-precision multiplications, where the result goes into an accumulator register that has at least twice as many bits as the ordinary data registers.
- To write the result back to a data register, however, we have to extract 16 bits from the 32 bit result. If we extract the shaded, then we preserve the position of the binary point, and the result still represents a number roughly in the range -1 to 1



EMBEDDED PROCESSORS

Fixed-Point Numbers

- There is a loss of information, however, when we extract 16 bits from a 32-bit result
- First, there is a possibility of **overflow**, because we are discarding the high-order bit
- Suppose the two numbers being multiplied are both -1



- This configuration in twos complement, represents 1, the correct result
- However, when we extract the shaded 16 bits, the result is now -1!
- Indeed, 1 is not representable in the fixed-point format 1.15, so overflow has occurred
- Programmers must guard against this, for example by ensuring that all numbers are strictly less than 1 in magnitude, prohibiting -1

EMBEDDED PROCESSORS

Fixed-Point Numbers

- A second problem is that when we extract the shaded 16 bits from a 32-bit result, we discard 15 low-order bits.
- There is a loss of information here.
- If we simply discard the low-order 15 bits, the strategy is known as **truncation**.
- If instead we first add the **bit pattern in the picture** to the 32-bit result, then the result is known as **rounding.**
- Rounding chooses the result that is closest to the full-precision result, while truncation chooses the closest result that is smaller in magnitude.
- DSP processors typically perform the above extraction with either rounding or truncation in hardware when data is moved from an accumulator to a general-purpose register or to memory.