EMBEDDED PROCESSORS

DSP Processors

- Suppose that an FIR filter is provided with samples at a rate of 1 MHz (one million samples per second), and that N = 32
- Outputs must be computed at a rate of 1 MHz, and each output requires 32 multiplications and 31 additions
- A processor must be capable of sustaining a computation rate of 63 million arithmetic operations per second to implement this application
- To sustain the computation rate:
  - the arithmetic hardware must be **fast enough**;
  - the mechanisms for getting data in and out of memory and on and off chip must be **fast enough**

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) \qquad x : \mathbb{Z} \to D \qquad x(n) = 0 \text{ for all } n < 0.$$

EMBEDDED PROCESSORS

DSP Processors

- An image can be similarly modeled as a function $x : H \times V \to D$, where $H \subset \mathbb{N}$ represents the horizontal index, $V \subset \mathbb{N}$ represents the vertical index, and D is the set of all possible pixel values.
- A pixel (or picture element) is a sample representing the color and intensity of a point in an image.
- There are many ways to do this, but all use one or more numerical values for each pixel. The sets H and V depend on the resolution of the image.

$$\forall\, i \in H, j \in V, \quad y(i,j) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{n,m} x(i-n, j-m)$$

EMBEDDED PROCESSORS

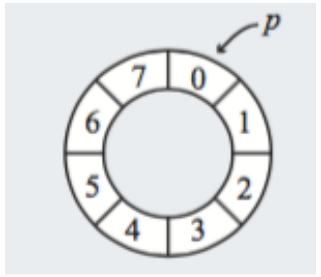
DSP Processors

- ATSC is a set of standards developed by the Advanced Television Systems Committee. DVB-T is the European standard.
- ATSC supports a number of frame rates ranging from just below 24 Hz to 60 Hz and a number of resolutions.
- High-definition video under the ATSC standard supports, for example, a resolution of 1080 by 1920 pixels at a frame rate of 30 Hz: H = {0, …,1919} and V = {0, …,1079}.
- This resolution is called 1080p in the industry.
- Professional video equipment today goes up to four times this resolution (4320 by 7680). Frame rates can also be much higher than 30 Hz. Very high frame rates are useful for capturing extremely fast phenomena in slow motion.

$$\forall\, i \in H, j \in V, \quad y(i,j) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{n,m} x(i-n, j-m)$$

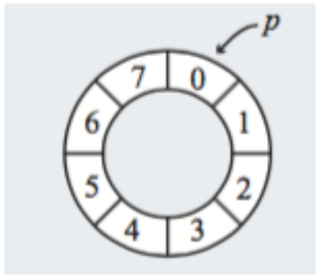
EMBEDDED PROCESSORS

- **DSP Processors**
  - An FIR filter requires a delay-line
  - A naive implementation would allocate an array in memory, and each time an input sample arrives, move each element in the array to the next higher location to make room for the new element in the first location
  - This would be enormously wasteful of memory bandwidth.
  - A better approach is to use a circular buffer.



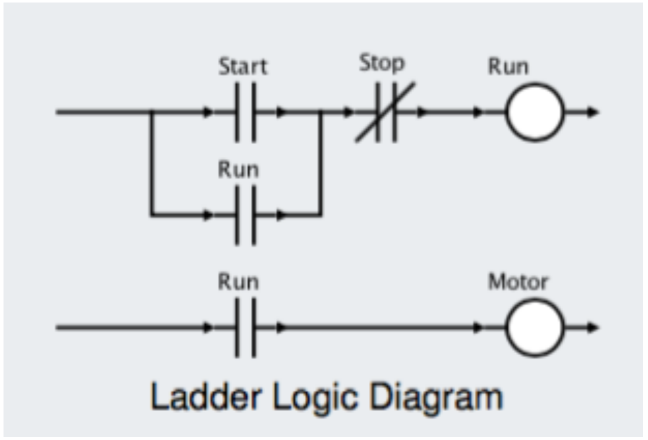
EMBEDDED PROCESSORS

- The implementation first accepts a new input value $x(n)$, and then calculates the summation backwards, beginning with the $i = N - 1$ term, where in the example, $N = 8$
- When the n-th input arrives, the value of p is $p_i \in \{0,...,7\}$ (for the first input $x(0)$, $p_i = 0$).
- The program writes the new input $x(n)$ into the location given by p and then increments p, setting $p = p_i + 1$
- All arithmetic on p is done modulo 8, so for example, if $p_i = 7$, then $p_i + 1 = 0$
- The FIR filter calculation then reads $x(n-7)$ from location $p = p_i + 1$ and multiplies it by a_7
- The result is stored in an accumulator register (previously been reset). It again increments p by one, setting it to $p = p_i + 2$
- It next reads $x(n-6)$ from location $p = p_i + 2$, multiplies it by a_6, and adds the result to the accumulator
- It continues until it reads $x(n)$ from location $p = p_i + 8$, the same location that the latest input $x(n)$ was written to, and multiplies that value by a_0
- Increments p, getting $p = p_i + 9 = p_i + 1$
- At the conclusion of this operation, the value of p is $p_i + 1$, which gives the location into which the next input $x(n+1)$ should be written



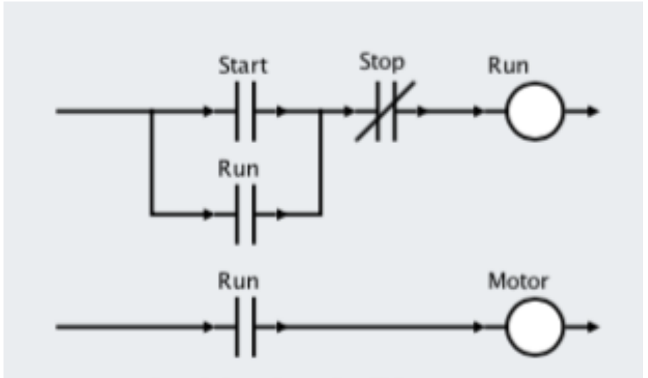
EMBEDDED PROCESSORS

- **Programmable Logic Controllers (PLC)**
  - A programmable logic controller (PLC) is a specialized form of a **microcontroller** in **rugged** package with **I/O interfaces suitable for industrial control**
  - PLCs are often programmed using **ladder logic**, a **graphical notation** originally used to specify logic constructed with **relays** and **switches**
  - A **relay** is a **switch** where the contact is **controlled by a coil**. When a **voltage** is **applied** to the **coil**, the **contact closes**, enabling **current** to **flow** through the relay
  - By interconnecting contacts and coils, **relays** can be used to **build digital controllers** that follow specified patterns
  - In common notation, a **contact** is represented by **two vertical bars**, and a **coil** by a **circle**



EMBEDDED PROCESSORS

- **Programmable Logic Controllers (PLC)**
  - The diagram has two **rungs**
  - The Motor coil on the lower rung turns a motor on or off.
  - The Start and Stop contacts represent pushbutton switches
  - Start is a **normally open** contact
  - The Stop contact is **normally closed**, indicated by the slash, meaning that it becomes open when the operator pushes the switch
  - When the operator pushes Start, current flows to the Run coil, causing both Run contacts to close
  - The motor will run, even after the Start button is released
  - When the operator pushes Stop, current is interrupted, and both Run contacts become open, causing the motor to stop
  - Contacts wired in parallel perform a logical **OR** function, and contacts wired in series perform a logical **AND**
  - The upper rung has **feedback**; the meaning of the rung is a fixed point solution to the logic equation implied by the diagram.



Ladder Logic Diagram

EMBEDDED PROCESSORS

- **Graphics Processors**
  - A graphics processing unit (GPU) is a specialized processor designed especially to perform the calculations required in graphics rendering.
  - Such processors date back to the 1970s, when they were used to render text and graphics, to combine multiple graphic patterns, and to draw rectangles, triangles, circles, and arcs. Modern GPUs support 3D graphics, shading, and digital video. Dominant providers of GPUs today are Intel, NVIDIA and AMD.
  - Some embedded applications, particularly games, are a good match for GPUs. Moreover, GPUs have evolved towards more general programming models, and hence have started to appear in other compute-intensive applications, such as instrumentation. GPUs are typically quite power hungry, and therefore today **are not a good match for energy constrained embedded applications**.

EMBEDDED PROCESSORS

- **The x86 Architecture**
  - Dominant ISA for desktop and portable computers.
  - Originates with the Intel 8086, a 16-bit microprocessor chip designed by Intel in 1978.
  - A variant of the 8086, designated the 8088, was used in the original IBM PC, and the processor family has dominated the PC market ever since.
  - Subsequent processors in this family were given names ending in “86” and generally maintained backward compatibility.
  - The Intel 80386 was the first 32-bit version of this instruction set, introduced in 1985.
  - Today, the term “x86” usually refers to the 32-bit version, with 64-bit versions designated “x86-64.”
  - The Intel Atom, introduced in 2008, is an x86 processor with significantly reduced energy consumption. Although it is aimed primarily at netbooks and other small mobile computers, it is also an attractive option for some embedded applications. The x86 architecture has also been implemented in processors from AMD, Cyrix, and several other manufacturers.

EMBEDDED PROCESSORS

- **Parallelism**
  - Most processors today provide **various forms** of parallelism
  - These mechanisms strongly affect the **timing** of the **execution** of a program, so embedded system designers have to **understand** them

EMBEDDED PROCESSORS

- **Parallelism** vs. **Concurrency**
  - **Concurrency** is **central** to **embedded systems**
  - A program is said to be **concurrent** if different **parts** of the program **conceptually execute simultaneously**
  - A **program** is said to be **parallel** if different parts of the program **physically execute simultaneously** on **distinct hardware** (such as on multicore machines, servers in a server farm, or distinct microprocessors)

EMBEDDED PROCESSORS

- **Parallelism** vs. **Concurrency**
  - **Non-concurrent** programs specify a **sequence** of **instructions** to **execute**. A programming language that expresses a **computation** as a **sequence** of **operations** is called an **imperative** language. C is an imperative language.
  - When using C to write concurrent programs, we must step outside the language itself, typically using a **thread library**.
  - A **thread library** uses facilities provided not by C, but rather provided by the operating system and/or the hardware. Java is a mostly imperative language extended with constructs that directly support threads. Thus, one can write concurrent programs in Java without stepping outside the language.
  - Every (correct) **execution** of a program in an **imperative** language must behave as if the **instructions** were **executed exactly** in the **specified sequence**. It is often **possible**, however, to **execute** instructions in **parallel** or in an **order different** from that specified by the program and still **get behavior matching sequential execution**

EMBEDDED PROCESSORS

- **Parallelism** vs. **Concurrency**

```
double pi, piSquared, piCubed;
pi = 3.14159;
piSquared = pi * pi ;
piCubed = pi * pi * pi;
```

The last two assignment statements are independent, and hence can be executed in parallel or in reverse order without changing the behavior of the program.

```
double pi, piSquared, piCubed;
pi = 3.14159;
piSquared = pi * pi ;
piCubed = piSquared * pi;
```

The last statement depends on the third statement so the third statement must complete execution before the last statement starts

EMBEDDED PROCESSORS

Dataflow analysis

- A compiler may analyze the dependencies between operations in a program and produce parallel code, if the target machine supports it
- Many microprocessors today support parallel execution, using multi-issue instruction streams or VLIW (very large instruction word) architectures
- Processors with multi-issue instruction streams can execute independent instructions simultaneously
- The hardware analyzes instructions on-the-fly for dependencies, and when there is no dependency, executes more than one instruction at a time

EMBEDDED PROCESSORS

Dataflow analysis

- VLIW machines have **assembly-level instructions** that specify **multiple operations to be performed together**. In this case, the **compiler is** usually **required to produce** the appropriate **parallel instructions**
- The **dependency analysis** is done **at the level of assembly language or at the level of individual operations**, not at the level of lines of C
- **A line of C may specify multiple operations, or even complex operations like procedure calls**. In both cases (multi-issue and VLIW), an imperative program is analyzed for concurrency in order to enable parallel execution
- The overall **objective** is to **speed up execution of the program**. The **goal** is **improved performance**, where the **presumption** is that **finishing** a task **earlier is** always **better than finishing it later**