CHAPTER 4: DSP SOFTWARE

Number Precision

Number representation errors are very similar to quantization errors during ADC. To keep the desired values in a continuous array; however, it can only represent a limited number of quantized levels. Some programming languages allow for a variable called a long integer that is stored as 32 bits, fixed point, and the sum of two. In fixed point variables, the spaces between adjacent numbers are always exactly one. In floating-point notation, the spaces between adjacent numbers vary within the range of numbers represented. If we choose a random floating-point number, the space next to that number is about ten million times smaller than the number itself, and the rounding error in each of the arithmetic operations causes the value X to gradually move away from its initial value. This deviation can take two forms depending on how the errors come together. If the rounding errors are randomly positive and negative, the value of the variable will randomly increase and decrease. If the errors predominantly have the same sign, the value of the variable will move away much faster and more uniformly. The additional error is roughly equal to the rounding error of a single transaction and multiplied by the total number of transactions. In comparison, random error only increases in proportion to the square root of the number of transactions. It is almost impossible to control or predict which of these two behaviors a given algorithm will experience.

Execution Speed: Programming Language

DSP programming loosely consists of three complexities: Assembly, Compiled, and Application Specific. Since it is difficult to work in a binary system, these ones and zeros are assigned names according to the function they perform. This level of programming is called assembly. Executable code can be run directly on the microprocessor. The next level of complexity can manipulate abstract variables without any reference to specific hardware. These are called compiled or high level languages. C, BASIC, FORTRAN, PASCAL, APL, COBOL, LISP. Like a dozen or so in common use. The compiler is used to convert high-level source code directly to machine code. This requires the compiler to assign hardware memory locations to each of the referenced abstract variables. The compiler breaks down complex mathematical expressions into more basic arithmetic. Microprocessors know only addition, subtraction, multiplication and division. Anything more complex should be done as a sequence of these basic operations. High-level languages separate the programmer from the hardware. This makes programming much easier and allows the source code to be moved between different microprocessors. Most importantly, the programmer using a compiled language does not need to know anything about the inner workings of the computer. Let's say you buy a newly developed DSP microprocessor to add to your existing project. These devices usually have many built-in features for DSP: analog inputs, analog outputs, digital I / O, anti-aliasing and reconstruction filters. Writing a routine in assembly is equivalent to a nearly two-year leap in hardware technology, as the speed of personal computers increases by about 40% each year. There are 3 reasons to program the DSP application in a language other than C: First, DSP has grown so fast that some organizations and individuals are stuck in other languages such as FORTRAN and PASCAL. Second, some applications require the highest efficiency achievable only with assembly programming. Third, C is not an easy language to learn, especially for part-time programmers.

Execution Speed: Hardware

The original IBM PC was introduced in 1981, based on an 8088 microprocessor with a 4.77 MHz clock speed and 8 bit bus. This was followed by a new generation of personal computers introduced every 3-4 years: 8088 → 80286 → 80386 → 80486 → 80586 (Pentium). Each of these new systems increased the speed of computing about five times compared to the previous technology. In 1996, the clock speed increased to 200 MHz and the bus to 32 bits. With other improvements, this has resulted in an increase in computing power of about a thousand in just 15 years. Since computers are made up of many subsystems, the time required to execute a given task will depend on two main factors: 1) the speed of the individual subsystems and 2) the time it takes to transfer data between these blocks. The Central Processing Unit (CPU) is the heart of the system and consists of up to a dozen registers, each capable of holding 32 bits. The CPU also includes digital electronics necessary for moving bits around and for basic operations such as fixed point arithmetic.

Complex math is handled by transferring data into a special hardware circuitry, also called the arithmetic logic unit or ALU, which is the math processor. The math coprocessor can be on the same chip as the CPU or it can be a separate electronic device. Most personal computer software can be used with or without a math coprocessor. This is accomplished by having the compiler generate machine code to handle both states, and all of this is stored in the last executable program. The speed with which data can be transferred between subsystems depends on the number of parallel data lines provided and the maximum speed of digital signals that can be transmitted along each line. Compared to data transfer between chips, digital data can often be transferred within a single chip at a much higher speed. A pipeline architecture divides the hardware required for a given task into several successive stages. The internal architecture used to achieve the increased speed includes: 1) a large number of very fast caches located on the chip, 2) separate buses for programs and data, allowing these two to be accessed simultaneously (called the Harvard Architecture), 3) located directly on the microprocessor quick hardware for math calculations and 4) a pipeline design.

Execution Speed: Programming Tips

While computer hardware and programming languages are important for maximizing execution speed, they are not something that changes daily. In contrast, how it is programmed can be changed at any time, and it greatly affects how long the program will run. 3 important suggestions can be given. First, integers are used instead of floating point variables whenever possible. Traditional microprocessors such as those used in personal computers process integers 10 to 20 times faster than floating point numbers. In systems without a math processor, the difference can be 200 to 1. An exception to this is integer division, which is usually performed by converting values to floating point. This makes the process very slow compared to other integer calculations. Second, avoid using complex operations. Third, find out what is fast and what is slow in the computer system. It comes with experience and tests, and there will always be surprises. Pay special attention to graphic commands and I / O. There are usually several ways to meet these requirements and speeds can vary widely. Functions are computed as a series of addition, subtraction, and multiplication.