What is Real-Time DSP?

Digital Signal Processing: An operation or transformation of a signal on a computer or other special purpose digital hardware.

Real-Time Digital Signal Processing: An operation or transformation of a signal on a computer in synchronization with events occurring in the physical system and with time. One such event is the sampling of the signal.

How do we do DSP in real time?

Real-time Digital Signal Processing is done by writing software for a specialized microprocessor called (of course) a digital signal processor (DSP). We will be programming Motorola's (now Freescale's) 3rd generation signal processor, DSP56302 and Texas Instruments' latest processor, TMS320C6416. Software for the Freescale DSP will be written in assembly language while that for the TI DSP will be written in C.

The DSP communicates with the analog world through A/D and D/A converters (we'll call this a codec) and with a host PC through an RS232 or USB interface. The codec on the Motorola EVM is the Crystal Semiconductor's CS4215. The processor and codec reside on aboard called the Evaluation Module (EVM) or DSP Starter Kit (DSK). We will be using Motorola's DSP56302EVM and TI's TMS320C6416DSK.

Figure: Typical EVM/DSK

There are three tools used in developing the software (code): an editor, assembler (or compiler), and debugger. The editor is used to type in code and comments (you can use Emacs, NotePad, PFE, Microsoft Word, or vi). The ASM56300 assembler translates Motorola 56300 instructions to object code. The Domain Technologies Debug-EVM is used to load the object code onto the Motorola EVM and initiate execution, trace execution, and examine registers/memory among other things. For the TI DSK, the editor, compiler (translates C code into object code), and debugger are integrated into a single development environment (IDE) called Code Composer Studio.

Pseudo-code for a typical real-time DSP application is as follows:

```
initialize_DSP
initialize_codec
initialize_memory
main
      wait_for_input_samples
      get_input_samples
      process_samples
      put_output_samples
      goto main
```

Clearly, real-time DSP applications are limited to cases where the required sampling rate is sufficiently less than the DSP's instruction rate so a reasonable number of instructions can be performed between sample periods. Unlike most code you have probably written, real-time *must* be executed within a certain period of time (usually the sampling period, T) to maintain synchronization. If the code does not execute quickly enough, there is the risk of not properly processing samples. Sometimes output samples may not be passed to the codec and will be "dropped."

Since the sample period for audio-band applications is relatively large, DSPs are used extensively in audio

applications since many instructions can be performed between samples. These applications include telephony, personal audio, and home theater applications. The following two examples will determine bounds on the amount of processing per sample that can be performed real-time using common audio-band sampling rates.

Example: Assume that the Motorola DSP56302 is clocked at 66 MHz and can effectively perform 66 MIPs. Furthermore, assume the highest sampling rate on the CS4215 codec is 48 kHz (standard audio rate typical of high-performance digital audio applications). Simple division dictates that

Instructions per sample = 66,000,000 / 48,000 = 1,375

This will be the lower bound on the number of instructions we theoretically can execute in a sample period with the highest sample rate available on the Motorola DSP56302EVM.

Example: Now, assume the lowest sampling rate on the CS4215 codec is 8kHz (standard audio rate typical of voice-band and telephony applications). Again simple division dictates that

Instructions per sample = 66,000,000 / 8,000 = 8,250

This will be the upper bound on the number of instructions we can theoretically execute in a sample period with the lowest sample rate available on the Motorola DSP56302EVM. Thus depending on the sample rate, the number of instructions which can be executed in a sample period on the Motorola DSP56302EVM is bounded by

$$1375 < \#\text{instructions} < 8250.$$

The Texas Instruments C6416 process we will use is clocked at either 720 MHz or 1 GHz. With an advanced processor design, high clock rates, and low sample rates, practically speaking, we do not need to worry about code not executing fast enough!

Benefits of DSP

Popular DSPs were introduced in the early 1980's and have caused a revolution in product design. Current manufacturers and devices include:

Analog Devices – ADSP-21xx, SHARC, TigerSHARC, and Blackfin
Agere – DSP16000
Freescale (spunoff from Motorola) – DSP563xx, 56800, MSC711x (StarCore), MSC81xx (StarCore)
Texas Instruments – TMS320C2000, TMS320C5000, TMS320C6000

DSPs differ from ordinary processors in that they are specially *designed to rapidly compute FFTs and inner products*

$$y(n) = \sum_{k=0}^{N-1} h(k)\,x(n-k)$$
$$¿\, h^T x(n)$$

where the vectors are defined as

$$h = \begin{bmatrix} h(0) & h(1) & L & h(N-1) \end{bmatrix}^T$$
$$x(n) = \begin{bmatrix} x(n) & x(n-1) & L & x(n-N+1) \end{bmatrix}^T.$$

In addition, DSPs are often *designed to use low power* and are *designed with a wide variety of on-chip peripherals* such as interfaces to communications and audio/video systems.

What has brought about the rapid shift to DSP-based electronics? It can often be advantageous to implement many functions with digital signal processing techniques rather than by hard-wired analog circuits:

FLEXIBILITY: Complicated signal processing when performed on a DSP is done in software. Modifications to the signal processing is achieved by modifying *software*. These changes can be rapidly deployed in the manufacturing line by changing a few lines in a ROM or erasable programmable read-only memory (EPROM) or even delivered over the Internet. Changes in analog electronics can often be much more difficult.

RELIABILITY: Integrated digital circuits are very reliable and can be automatically inserted in boards. Once the code is perfected, the chip function does not change with age or temperature as is sometimes the case with analog electronics.

LOW POWER: DSPs are designed from the ground up to have low power consumption making them ideally suited for applications such as cellular telephones, MP3 players and iPods, and digital cameras.

ECONOMICS: Low-cost DSPs have made it more economical to digitally implement signal processing applications, particularly audio-band applications like modems. DSPs have made it possible to manufacture products with tremendous complexity and sophistication at prices ordinary consumers can afford. For fixed cost, DSP performance has doubled every two years for past 20 years.

The DSP approach is not always the best solution for the problem even if DSP can accomplish the task. For example, a commercial AM radio signal (carrier frequency on the order of 1MHz) can be trivially demodulated with a simple envelope detector consisting of a diode, resistor, and capacitor. As engineers you should always look for the most reasonable and economical method of solving a design problem.

DSP Chip History

The Intel 2920 (1979) was the first commercially-available, general-purpose single chip DSP. The device was not fully appreciated at the time, possibly because there were few engineers conversant with DSP theory and applications. We note that the *Digital Signal Processing* by A. Oppenheim and R. Schafer was first published in 1975. Although, the 2921 appeared a year or so later, Intel soon discontinued manufacturing DSP devices. In the 1990s, Intel released the i960 microprocessor which is used in DSP applications and added multimedia/DSP capabilities (MMX technology) to the Pentium family.

Several years later (1982) other DSP chips were designed including the NEC PD7720 and Texas Instruments TMS32010. The chips were capable of 16-bit integer arithmetic at the rate of 5 million instructions per second (MIPS) and had limited random access memory (RAM), read-only memory (ROM), and input/output (I/O) capabilities. These processors, however, were commercially successful and paved the way for future generations of DSPs.

Another milestone in DSP history occurred in 1986 when AT&T (now Lucent Technologies) introduced the first commercially available floating-point DSP, the DSP32. A floating point DSP frees the engineer from signal scaling issues such as under- or over-flowing, however, they are often more expensive and consume more power thereby limiting their application. In fact, most commercial products utilize fixed-point DSPs while floating point DSPs are mainly used in image-processing applications. All members of the Motorola DSP56300 family are fixed-point devices while those in the DSP96000 family are floating-point.

