UNIT-GENERATOR IMPLEMENTATION ON THE NeXT DSP CHIP

Julius O. Smith III
NeXT Inc. and CCRMA, Stanford University

This is the original submission for pp. 303-306 of the
Proceedings of the International Computer Music Conference (ICMC-89)
published by the Computer Music Association, 1989.

Introduction

Each NeXT computer includes a DSP56001 digital signal processing chip (the "DSP") and CD-quality sound output hardware. The DSP architecture is excellent for sound synthesis. Since Music V, instruments have been built using elementary building blocks called *unit generators*. This paper describes the unit generator implementation on the DSP in the NeXT Computer.™

The DSP

The Motorola DSP56001 is a state-of-the-art digital signal processing chip that can execute (with the clock at 25 MHz) 12.5 million instructions per second and, in a single instruction, can perform a 24 by 24-bit multiply, a 48 plus 56-bit addition, two parallel data moves, an instruction fetch and decode, and two general index updates. The 24-bit data paths and computational architecture make the DSP56001 exceptionally well suited for real-time digital audio synthesis.

DSP on-chip memory is divided into three banks of 512, 24-bit words. The *p* memory bank provides 512 words of program RAM on chip. The *x* and *y* memory banks each provide 256 words of on-chip data ROM, and 256 words of on-chip data RAM. The internal *x* ROM contains mu-law and A-law expansion tables, while the internal y ROM contains one complete cycle of a sine wave. The concatenation of *x* and *y* memories into a single 48-bit word ("long") memory is referred to as the *l* memory space.

Off-chip DSP memory on the NeXT computer exists in two address ranges, each of which spans all of the 8K words of external memory. In the first address range (8K to 16K), *x*, *y*, and *p* memories are *overlaid*; that is, an external memory reference points to the same off-chip location regardless of the memory space specified. Note that in this address range, there is no *l* memory space support. (The high and low words are mapped to the same word.) In the second address range (40K to 48K), *x* and *y* are split into separate 4K partitions, and *p* overlays the entire 8K. This address region allows external *l* memory use, and supports algorithms (such as the Motorola benchmarks involving complex data) which expect *x* and *y* memories to be physically separate.

The three on-chip DSP memory banks can be accessed in parallel, with as many as three moves occurring in one instruction cycle (80ns). Off-chip DSP memory can't be accessed in parallel, requiring three instruction cycles to move a word in all three memory spaces off-chip.

Software Architecture

Music synthesis on the DSP is accomplished in the *orchestra loop* which is a block of DSP code executing once for each *tick* of digital sound output. A *tick* is some number (currently 16) of digital sound samples. If not for efficiency considerations, the orchestra loop would compute a single sample of digital sound per iteration to allow parameter updates every sample rather than every tick. However, the architecture of the DSP56001 (being pipelined) is much more efficient when computing digital sound in small blocks at a time.

The orchestra loop is structured as follows:

- Perform system updates
- Execute unit generators

System updates include:

- Issue sound-out DMA request if needed
- Execute all timed messages whose time has come
- Add the tick size to the 48-bit current-sample count
- Reset the unit-generator memory-argument pointers

Unit Generators

Unit generators are fundamental building blocks of sound synthesis. They can be regarded as "black boxes" in a real-time signal-processing diagram. The unit-generator input and output signals are called *patch-points*. Each patch-point is a short contiguous vector of digital sound data, of length equal to the tick size. Almost every unit generator has an output patch-point, and most unit generators have one or more input patch-points. At the heart of every unit generator is a hardware **do** loop that executes once per sample to produce an output tick.

Unit generators are loaded *dynamically* using *timed messages* by the Music Kit Orchestra object. Each unit generator is a section of DSP code which has been assembled from a corresponding unit-generator *macro*. Unit generators are not subroutinesÐthey are expanded in line to avoid subroutine-call overhead. For real time synthesis, there is plenty of program memory to hold an orchestra of expanded unit generator macros. To make it possible to substantially rebuild the orchestra loop in real time between ticks, without sound-out or timed-message under-run, we use two 512-word stereo sound-out DMA buffers and a Timed Message Queue (TMQ) of approximately 1000 words in the DSP static RAM.

Memory Arguments

The parameters and run-time state of the unit generators are held in so-called *memory arguments*. There are three memory-argument€blocks, corresponding to *x*, *y*, and *l* DSP memory. The *l* memory arguments must be on chip since we use the overlaid external memory partition. The *x* and *y* memory arguments, however, are off chip, because on-chip memory is more valuable for patch-points (accessed in the inner loop) than for memory arguments (which are accessed outside the inner loop). Long memory arguments are used for oscillator phase and table increment (which determines frequency of oscillation), exponential envelope state, and the slope and current value of the linear amplitude ramp.

In the execution of an orchestra loop, each unit generator loads its ALU and index registers from its memory arguments. Any state needed for the next tick's computation (such as delayed signals in a digital filter) is written by the unit generator into the memory arguments on exit. To minimize pointer initialization overhead, the memory arguments are accessed sequentially in internal *x* and *y* RAM in the order needed by the unit generator. This eliminates initialization of the three memory argument pointers, except once at the beginning of the orchestra loop. Loading of addresses from memory arguments typically looks as follows:

```
move y:(R4)+,R6          ; output address to output pointer R6
move x:(R0)+,R1          ; input 1 address to input pointer R1
move y:(R4)+,R5          ; input 2 address to input pointer R5
```

Note that addresses cannot be loaded in parallel. Loading of data memory arguments (such as oscillator amplitude or digital filter coefficients), are handled the same way, except that sometimes two data arguments can be load using parallel move syntax, e.g.,

```
move x:(R0)+,X0          y:(R4)+,Y0          ; load gain1 to X0 and gain2 to Y0
```

Each unit generator is responsible for leaving R0, R4, and R2 (*l* arguments) pointing one past its argument block on exit. If a unit generator needs to save run-time state (such as current envelope amplitude), it's best to place this state last in the memory arguments. Having all running state in the memory arguments facilitates memory compaction needed with dynamic loading.

All memory arguments are given symbolic names. These symbols are collected into the assembly output file and are needed to write (or read) the arguments from a host task. The Music Kit™ takes care of this linkage for most users, but these hooks can support more general real-time signal processing applications which need not involve the Music Kit.

The Importance of Vectorized Computations

In the current implementation, the tick size is 16 samples. Since the overhead of setting up DSP address and ALU registers is often comparable to the amount of actual work done in the inner loop of a tick computation, a tick size of 16 reduces the set-up overhead for each unit generator from around fifty percent to well under ten percent in most cases.

It's important not to make the tick size any larger than necessary to amortize register set-up overhead, however, because the tick size equals the size of a patch-point, memory that's used for communication between unit generators. It's highly desirable that all patch-points fit on chip in the DSP. This is because the three-way parallel data move capability of the DSP requires that at least two of the data moves be to or from on-chip memory.

The hardware do instruction available in the Motorola DSP56001 further enhances the benefits of vectorized computations by performing the loop test and branch in parallel with the block iteration; while there is a three instruction-cycle (six clock cycle) overhead incurred to set up the loop, the individual loop iterations suffer no test-and-branch overhead.

Finally, the dual parallel indexing ALUs provide zero-overhead memory address updates for two parallel data transfers, with skip factors, modulo (wrap-around) addressing, and even bit-reverse indexing for FFT data shuffling provided as indexing modes.

Thus, vectorized computations are far more efficient than computing a single audio sample per iteration of the orchestra loop. The price for this efficiency is a loss of control bandwidth since parameter updates (envelope break-points, filter coefficients, etc.) are only installed once at the beginning of each tick.

Constraints on input and output memory spaces

We have found it useful to constrain the input signal memory spaces (for optimum performance) before constraining its output memory space. For example, given a choice between reading two inputs in parallel (which requires them to be in opposite memory spaces) and writing the output signal as a single parallel move (in which case either x or y may be specified) versus reading one input in parallel with writing the output (which forces the input to be in an opposite space relative to the output), the former is preferred. This allows, when building a "synth-patch", choosing whichever output space optimizes the subsequent reading unit generator. This scheme provides optimal structures easily whenever only one unit generator reads each output signal.

Interconnection and Execution Order

Conceptually, there is a patch-point for each signal interconnection between unit generators. However, if the order of unit-generator execution is properly arranged, several patch-points can share physical memory. Also, the order in which unit generators are executed in the orchestra loop determines whether or not there is a 16-sample delay in the connection between them. For these reasons, the Music Kit provides a way to control the order of execution of unit generators.

Acknowledgements

The design of the computer music engine on the DSP56001 was helped by valuable suggestions from David Jaffe, Andy Moorer, and Roger Dannenberg.