Music Applications for the MSSP System

John W. Gordon
Schlumberger Palo Alto Research
3340 Hillview Avenue
Palo Alto, CA 94304
(415) 496-4608

Abstract

The Serial Signal Processor (SSP) chip, designed by Richard F. Lyon, is a VLSI realization of a multipurpose register mem-
er-y (64 × 32 bits) interconnected with two multiply-accumulate functional units and one arithmetico-logic unit. Each mi-
crocode instruction consists of 8 source and 8 destination register addresses, thereby controlling data flow in and out
of the chip as well as between register memory and the three functional units. All microcode and data are in bit-
serial format, with 32 bits per word; the word clock is external to the chip and can be as high as 100 KHz. The
MSSP system consists of a network of inter-communicating SSP chips, each chip executing the same program in par-
allel, but with different data and/or coefficients. MSSP is thus a single-instruction, multiple-data (SIMD) parallel processing system.

MSSP was not designed to be a music processor, and many conventional synthesis algorithms cannot be directly im-
plemented on the MSSP system. However, most synthe-
sis techniques can be accomplished with MSSP by using alternative algorithms — that is, by using combinations of
multiples, adds, simple nonlinear operations, and logic
functions, rather than table lookup. Moreover, MSSP has the processing power to synthesize or process many voices or
channels of sound in real time; at an audio sampling rate of 40 KHz, each SSP chip can perform 20 multiplications
and 30 adds per sample. This approach of using specially
designed VLSI chips for music synthesis is not unlike that
adopted by Wawrzynk and Mead, though the implementa-
tion is strikingly different; some comparison of the two
systems is given in more detail.

1. INTRODUCTION

The Multi-Serial Signal Processor (MSSP) is a system de-
signed to perform signal processing algorithms at high speed [Lyon (1984)]. Its basic unit is the Serial Signal
Processor (SSP), a custom NMOS VLSI chip, consisting of
2 multiply-accumulate functional units, an arithmetico-logic unit, and multi-port register memory. Microcode controls data
flow in and out of the chip as well as between register mem-
ory and the three functional units; all microcode and data

Figure 1. Example of a complete MSSP system. All unmarked pins are data streams. Some number of SSP
chips are configured into an array and perform identical
programs in parallel. The controller transmits micro-
instructions to all chips at once, and manages 5 data chan-
nels (3 input and 2 output). One data channel is broadcast
to all chips; the other 4 connect with the end chips of the
array.

are in bit-serial format, with 32 bits per word. The MSSP
system consists of a network of inter-communicating SSP
chips, each chip executing the same program in parallel,
but with different data and/or coefficients. MSSP is thus a
single-instruction, multiple-data (SIMD) parallel process-
ing system.

The MSSP was designed primarily to support research in
speech recognition based on signal processing techniques
operating in parallel over many channels [Lyon (1985)]. Al-
though the system was not designed to be used as a music
processor, its architecture is general enough that it can be
used to implement many sound synthesis and processing
techniques. Furthermore, exploring the use of MSSP for
musical applications is motivated by the following reasons:
(1) MSSP has the potential for being a cost-effective way
to obtain significant real-time processing power at high-
quality audio sampling rates; (2) MSSP offers an alterna-
tive or complementary approach to the work being done in

159

ICMC '85 Proceedings
VLSI music synthesis at the California Institute of Technology [Wawrzynek and Mead (1983)]; (3) by discovering what functional additions or extensions to MESP would make its use for music more practical, future-generation designs can be made more versatile for a number of signal processing applications.

2. MESP ARCHITECTURE

The MESP system consists of a controller communicating with an array of SSP chips. Programma run on the controller determine the scheduling of microcode and data streams sent to the array as well as on-chip and inter-chip data flow. An overview of this system is shown in Figure 1. This figure shows the SSP chips organized in a linear array; examples of other configurations are discussed below. Let us now examine the components of this system in more detail.

2.1. SSP Chip Architecture

The essence of the SSP architecture is a set of 8 data simultaneously being read from memory, processed by 3 functional units, and written back into memory. The sources and destinations for these data are determined by microcode: each microinstruction contains 8 read addresses and 8 write addresses. Since each functional unit has more than one input but only one output, only 3 of the write addresses are used to determine unit destination registers; the other 5 are used to store data entering off-chip. Also, any of the 8 unit sources can be simultaneously sent off chip to other SSPs or the controller. Figure 2 illustrates the functional operation of the SSP.

The SSP register memory consists of 64 32-bit words. All data flow is accomplished serially on single wires. The bit clock is generated off chip and can be as high as 16 MHz, which means that word-cycle times are as low as 2 μs. Eight new words are read from register memory every word cycle and pipelined into the functional units; it takes 3 cycles for these values to be processed by the units and written back into memory. Due to bit-serial data flow, operators and ports are naturally pipelined at the bit-time level; therefore, no special pipeline registers at the word-time level are needed.

The adder input to each multiply-accumulate functional unit passes through a limiting function; this can be used for saturation arithmetic to prevent non-linear overflow oscillations. All numbers are represented as 32-bit integer fractions, with the binary point positioned such that the operating range is between 0.5 and 1. The limiting will force values to clip (saturate) at ±±. The arithmetic-logic unit (ALU) contains standard boolean operations such as AND, OR, and XOR; adds and subtracts with or without carry inputs; and an unusual switching function that can be used for full-wave or half-wave rectification.

2.2. MESP Array Structure

The linear array of SSP chips shown in Figure 1 is perhaps the most standard configuration. Cascaded or parallel filter banks and other common signal processing algorithms map easily onto this arrangement of SSP processors. The size of the array depends on the intended application; the intent behind the design of the MESP system was that 80-100 SSPs would be used in a single linear array. Because so many read and write memory ports are accessible off chip, more intricate networks, including two-dimensional arrays, are possible; indeed, other configurations may be more appropriate for certain applications. An example of a slight rearrangement of the linear array is given in the subsection on additive synthesis.

2.3. Controller Architecture

The controller serves as an interface between the MESP and the host computer. It must communicate bit-serially with the MESP, but in bit-parallel with the host. The controller also needs to generate the SSP timing and clock signals, and synchronize all microcode and data-flow operations. Microinstructions are 160 bits wide (16 8-bit memory addresses and functional unit control bits), transmitted to the MESP as 32-bit serial streams. A new 160-bit instruction is sent every word cycle. In addition, 5 data channels are managed by the controller, as indicated in Figure 1.

A more detailed diagram of the controller is shown in Figure 3. The program counter (PC) shown is simply a counter—no program branching is possible. This means that loops need to be unrolled, which in turn implies that many instructions are repeated within a program. The controller architecture takes advantage of this by using a two-level memory scheme; actual instructions are stored in microcode RAM, and the program RAM stores only addresses. Data streams between controller and MESP are buffered into 64-word FIFOs at the controller end, thereby obviating any need for real-time data management by the host.
3. MAPPING MUSIC ONTO MSSP

Describing programs for the synthesis and processing of music on the MSSP is significantly different from describing programs for conventional synthesizers or general-purpose computers. Memory is a premium, so algorithms that conventionally employ tables need to be restructured to avoid table lookup. Multipliers, often a scarce resource in other architectures, are plentiful on the MSSP; thus, algorithms that involve many multipliers are not considered undesirable, and can even be attractive alternatives.

On a different level, one must keep in mind that the MSSP is really a vector processor. The power and efficiency of the MSSP are not exploited if only one or two SSP chips are used per controller; this implies that algorithms need to be designed so that many SSP chips can execute the same program in parallel.

Examples of how certain signals can be generated on the MSSP are given in the next subsection, followed by a discussion of possible ways to program common synthesis and processing techniques. One assumption being made in these examples is that all processing is done in real-time, with SSP output fed directly into a DAC buffer. It is, of course, possible to relax this restriction to realize more sophisticated algorithms; however, this would entail a much larger data management task for the controller, as well as a secondary storage medium.

3.3 Signal Generation

Perhaps the most important signal for music synthesis is the sine wave. Since table-lookup is not a viable means for generating sines, these waveforms need to be computed either by series expansion or by some recursive technique.

Since multipliers are so plentiful on the MSSP, series expansion is worthy of consideration; however, for low distortion, the order of the polynomial needs to be quite high, and pipeline backtraces would introduce complications.

This leaves recursive techniques. To generate sine waves in their system, Wawrzynek and Moad (1988) use a second-order resonator, expressed as

$$y(n) = 2\cos(\omega)\cdot y(n-1) - y(n-2) + x(n),$$

where $x(n)$ is essentially impulsive in nature. This method also works well on the MSSP, since word lengths are 32 bits. Another possibility is the modified coupled first-order form, discussed by Gordon and Smith (1983):

$$y(n) = x(n) - r \cdot y(n-1),$$

where $r$ is not used directly for frequency modulation (FM), since phase is not a direct parameter; however, the coupled form can be extended to accomplish FM without much additional computation.

It is also important to have the capability to generate waveforms other than sines, such as exponentials, pulses, and ramps. These are in general straightforward on the MSSP. Ramps are easily generated by integrating values using the ALU. Exponentials can be synthesized with successive multipliers by a constant; there should be no danger from limit cycles using this method, since 32-bit word lengths are used throughout. It is possible to generate pulse trains by combining exponentials with the proper ALU directly function. Band-limited pulses are certainly very difficult to synthesize, but could possibly be obtained by doubling two sines, however, divide algorithms are probably too time- and power-consuming to make this practical.

3.4 Synthesis Techniques

3.4.1 Additive synthesis. The formula for additive synthesis in the digital domain can be expressed as

$$y(nT) = \sum_{k=0}^{N-1} A_k(nT) \cdot \sin(2\pi \cdot f_k(nT)T + \phi_k).$$

Conceptually, this is a sum of partials, each partial being a sinusoidal function with time-varying frequency multiplied by a time-varying amplitude envelope. Since frequency changes are assumed to be very small between adjacent samples, the modified coupled first-order filter is easily (and relatively cheaply) extended to generate the sinusoidal function. Amplitude envelopes can be generated either with line segments (ramps) or with exponentials. K line segments are used, two harmonics per SSP can be synthesized at a 50 KHz sampling rate.

On a global level, it is evident that a large linear array of SSP chips can generate many, many partials for additive synthesis, since each chip can have its own set of frequency and amplitude parameters, and all chips can perform the same program in parallel. It is less evident...
how to sum all the partials into one or more waveforms, g(t). This can be accomplished by taking advantage of the extra SSE write port; the outputs of 5 chips can be forwarded simultaneously to a 4th chip in the array, and this 4th chip can add these three values to its own output. The outputs of these chips in turn be grouped similarly. The only added complexity involved is in the software; pipeline latency need to be carefully controlled and non-accumulating chips need to have the constant 0 fed into their extra write ports so that their output values aren't changed.

3.2.2 Frequency modulation. As was mentioned above, frequency modulation can be realized by extending the modified coupled first-order filter method for generating sinusoids. Unlike additive synthesis, FM synthesis involves wide and rapid frequency deviations; hence, the simple extension mentioned for additive synthesis is inadequate for FM. However, a more sophisticated extension is possible that should prove viable under most circumstances (Gordon [1985]).

Using this more complicated extension, a single FM instrument (one carrier and one modulator) can be realized on an SSE chip at 50 KHz—with about 6 multiply-adds left over for other applications. If a mixing scheme similar to that discussed under additive synthesis is employed, a complete FM-synthesis system with one instrument per chip can be had with a single linear array of SSEs.

3.2.3 Other processing. A synthesis technique that might be desirable to have realized on the MSSP is subtractive synthesis. This can be described as a time-varying linear filter excited by either noise or a pulse train. Subtractive synthesis should not be difficult to map onto the MSSP; filters map directly into sequences of multiplexes and adds, noise can be generated with a multiply-add unit (with appropriate coefficients), and the pulse train can be generated with the ALU. Time-varying filter parameters can be achieved by means of ramps. If an all-pole filter is used (commonly found in linear prediction), a filter of order 16-18 can be realized on a single SSE chip, assuming a 50 KHz sampling rate. However, it might be more appropriate to use an entire MSSP array to compute a few voices, with each chip computing one or two second-order sections of the filter(s).

Time-domain formant synthesis (FOF) [Roden 1980] is another technique to consider mapping onto MSSP. This technique involves the addition of many, many damped sinusoids, each scaled by a simple envelope; these are referred to as "FOFs" by Roden. A FOF is mapped easily onto an SSE, and up to 4 FOFs can be placed on each chip, again assuming a 50 KHz sampling rate. The accumulation configuration described under additive synthesis and FM can be used to sum the FOFs; thus, MSSP appears to be a good architecture for realizing this technique.

Wavervynck and Mead [1985] have paved the way for music synthesis by means of solving linear difference equations. It should be pointed out that MSSP is particularly well suited for this approach as well.

3.2.4 Difficult or impossible techniques to realize. MSSP is not a panacea for all music synthesis problems; indeed, there are certain techniques not realizable with this system. One such technique is waveshaping; there is simply not enough memory for storing tables. One could consider computing the waveshaping function directly, but only if the order of the polynomial were small. Lack of memory also prevents the realization of reverberation and tech- niques involving delay lines, such as the plucked-string algo- rithm [Jaffe and Smith [1985]]. A good solution to these limitations is off-chip memory, preferably in bit-serial form. In general, this memory would not need to be accessed at rates faster than audio sample rates (e.g., 50 KHz).

3.3 Updating Parameters and Programs

One of the most difficult system problems to solve in regard to music processing is the updating of parameters and/or algorithms. MSSP does not make updating easy; in fact, updating is probably the major bottleneck of the system. The general paradigm followed is to shift values into the array from one end towards the other, with all SSEs updating the same parameter in synchrony. While values are being shifted from one chip to the next, dummy register locations must be used for temporary storage. If one chip updates, all update, implying that unchanging parameters must be updated with their same values; this of course involves wasted data flow.

Programs can be changed relatively quickly, since programs reside on the controller. One thing that can't be changed quickly, however, is the configuration of the MSSP array. Thus, if different programs require different array configurations, one cannot switch between them without making some hardware changes.

4. COMPARISON OF MSSP WITH UPE NETWORKS

The music synthesis system developed by Wavervynck and Mead at Caltech involves connecting "Universal Processing Elements" (UPEs) into a processing network. Each UPE performs a multiply and add, uses bit-serial format, and performs 13-bit fixed-point arithmetic (some values can be 64 bits). There is no on-chip memory (except for a register to store the multiplier coefficient), and algorithms are realized by directly connecting chips together as a UPE network. Since it is desirable to be able to change al- gorithms quickly, an interconnect matrix is employed to fa- cilitate the altering of network configurations.

The SSE and UPE chips have much in common; both are custom VLSI parts, both use bit-serial, and both are focused around multiply-add functional units. However, their respective implementations of signal processing algo- rithms are strikingly different. Data flow for the MSSP is determined by wide microcode instructions (16 addresses), whereas UPE networks achieve data flow by means of direct UPE connections and an interconnect matrix. The register memory on the SSE allows for more flexible states and task
switching, and also permits processing to be done at faster-than-audio rates. Processing on the UPE is done at audio rates, and chip delay times are used to obtain whatever state information is desired for particular algorithms. This latency dependency renders certain algorithms, such as the modified correlated first-order sine generator, unrealizable by a UPE network, unless processing is done at a multiple of the audio rate and intermediate values are discarded.

The MSSP is a SIMD parallel processing system. This means it is intended to increase its processing power by adding more SSP chips to the array. However, it is not clear that this kind of parallel processing is the best way to synthesize music. On the other hand, an entire UPE network has to be replicated for each voice that uses that network as an instrument. This allows for direct software control over each voice, but the amount of software can become unwieldy.

Finally, there are cost considerations. There are about 4 or 5 UPEs per chip, occupying approximately the same total silicon as an SSP chip; however, each UPE has half as many multipliers/adders as an SSP. This implies that the UPE is roughly twice as efficient in doing multiply-adds as an SSP, but the different processing speeds of the two systems have to be considered in the cost equation. It is also difficult to assess the relative costs of controller and interconnect matrix mechanisms.

11. NEXT-GENERATION CHANGES

Our discussion of synthesis techniques has hinted at possible extensions or additions to the SSP architecture that would make certain tasks easier or even possible. One such extension is a more general data select mechanism to replace or augment the special ALU rectify function. This would allow for some local (on-chip) data-driven decision making, and could also serve as a way to efficiently update parameters in a single chip. An addition that would make reverberation and delay-line techniques possible is some off-chip delay memory (at least several thousand words). One would want to be able to treat this memory as either a random-access table or as a delay line of variable length.

Two extensions that are worth considering are more register memory and faster bit clocks. It is not clear that more than 64 registers on an SSP are needed for music processing, but if memory is made enough larger, one could begin to consider storing tables on chip. (A memory size of 256 is being considered for the CMOS version, now in the design stage.) A faster bit clock (12 MHz is possible in 2p CMOS) would not make some algorithms realizable that aren't already realizable at 16 MHz, but it would obviously increase the system's processing power.

A mechanism that would be particularly useful for music is a chip-connection facility configurable by software. This would be prohibitively expensive and complicated if such a mechanism were made completely general, but even a highly restricted mechanism could prove valuable. For instance, being able to switch between a straight-linear array and a configuration appropriate for additive synthesis would allow one to realize most if not all of the techniques discussed above without having to make hard-wired connection changes. Another possible solution to this problem is to provide more than one controller, or a controller that can manage more than one SSP array. This may not add too much complexity to the controller, unless one desires interaction of data between arrays.

6. CONCLUSIONS

The MSSP system is a viable architecture for music synthesis and processing. Most known synthesis techniques can be mapped onto this architecture, though there are a few notable exceptions. There are problems involved (such as parameter updates) in using MSSP for music applications, but they do not significantly detract from such an implementation. In fact, the disadvantages appear to be outweighed by the potential for a high-power-to-cost ratio.

MSSP also appears to be an attractive alternative to the UPE network system proposed by Weaveray and Mead. However, there are advantages and disadvantages to both approaches, and it is not clear that one is better than the other in an overall sense. Some of the disadvantages of MSSP might be addressed with future-generation designs.

7. REFERENCES


