A1.16

The architecture of DSP.*: A DSP multiprocessor

Mark Kahrst†

Digital Signal Processing Center
Dept. of Electrical and Computer Engineering
Rutgers University
P.O. Box 909
Piscataway, NJ 08854

Abstract:

Simple analog functions such as mixing and editing often have complex implementations in the digital domain. Previous machines designed expressly for such tasks, such as the Lucasfilm ASP [1] and Neve BBC [2] machine are heavily pipelined and complex machines with extremely fast cycle times. This paper describes the hardware architecture of a MIMD multiprocessor designed to handle the problems of high quality digital audio mixing, synthesis and editing.

1. Introduction

The computational demands of most DSP tasks are beyond any single available DSμP. For example, consider the problem of digital mixing. Moorer [1] estimates that at 44.1 kilosamples per second, a four band equalizer would need to execute at least one million multiply accumulates per second. Previous hardware solutions to this problem such as the Droidworks "SoundDroid" [3] (nee Lucasfilm ASP) were large and expensive machines. The Yamaha DMP-8 is a current example of how far the technology has come in just five years. But the Yamaha machine is limited to eight channels and is strictly a mixer with limited functions.

Synthesis is another task which can be broken down for multiple processors. Each processor can run a specific synthesis algorithm; the results of these computations can be combined by a mixing processor and output to external converters. A truly flexible research machine should be capable of performing most digital audio algorithms from mixing to editing and synthesis.

If the "one processor, one voice doctrine" is accepted, then there must be some way to combine the results of these independent processors. Now the interprocessor communication is the problem to solve (for large problems, this assumes that the problem can be decomposed for multiple processors).

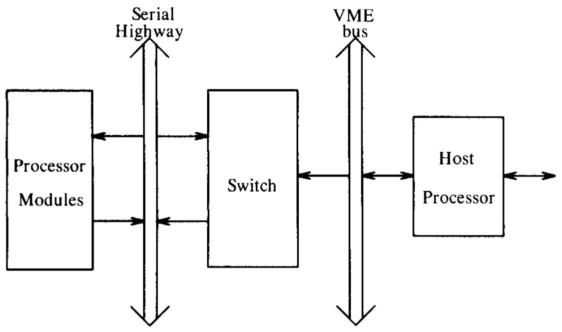
Fortunately, telephone switching provides a solution: The time slot interchange crossbar (for a history of telephone switching, see [4]). A time/space division switch allows multiple processors to feed results to a single processor through appropriate switch programming. A programmable switch also permits the use of "flexible topologies"; for different problems one need only change the connections of the switch. Of course, the switch has a delay associated with it. The prototype has a delay of one sample time between ingoing and outgoing samples.

Time division multiplexing on serial channels is an old telephone technology. It has also been used for high quality audio (i.e., the AES/EBU serial format [5]). For example, at 44.1 kilosamples per second and 16 bits per sample, the serial bandwidth is 705.6 kilobits/second. For a processor with an 8 megabit limitation, this means that the processor can input almost 12 samples in one output sample time. Since the data is serial, the data can be easily distributed to other processors (via the switch) on a backplane.

DSP.*[1] was designed around this philosophy. An architectural block diagram of the system is shown on the next page.

†work performed at AT&T Bell Laboratories, Murray Hill, NJ 07974

[1] The name is derived from Cm* [6], an early experimental multiprocessor designed at CMU



Above, processor modules are connected to the serial highway. Each processor module has a *unique* address on the highway (this address is wired on the card). There are two traces on the backplane per processor: one for the input to the processor (from the switch) and one from the processor (to the switch). All interprocessor communication is via the switch. The switch sits between the VME bus and the host processor and is programmed by the host processor (the switch is covered in more detail in section 2.3). The prototype uses a 68010 as a host processor connected via a serial line to either a VAX or an Alliant.

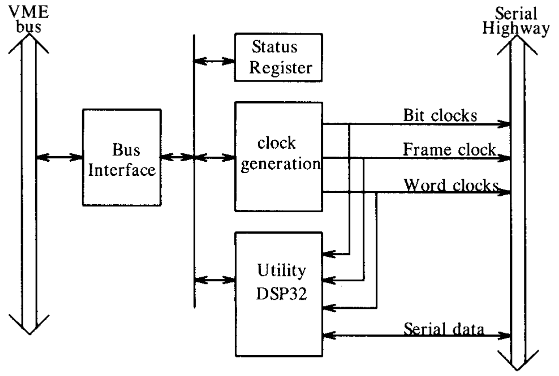
In the remainder of the paper, the details of the hardware architecture are described. Although the prototype uses WE® DSP-32s [7], any processor with a high speed serial interface could be used. This is an additional benefit of using parts designed for telephone switches.

2. Hardware architecture

This section describes the architectural and hardware details of the prototype machine. There are three sections, one for each card.

2.1. The host interface

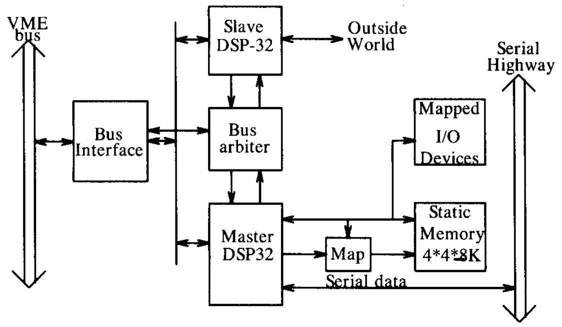
All processors must use the same serial clocks to stay in synchronization with the sample frame clocks. These clocks are buffered by the host interface card and put on the backplane. The card also generates the processor clocks, so the processors can also run in total synchronization (if required by the application). The host interface also provides a mechanism for spying on the output of individual processors without the intervention of the switch. Lastly, it provides a DSP-32 for putting data out on the bus. The DSP can serialize data as well as provide small filtering functions and generation of sync clocks (if desired). The datapath of the card is found on top of the next column:



On this card, the host can write registers that set the serial receive and transmit rates for all the processor modules. It also generates the clock for the processors. As mentioned above, a DSP-32 serializes data from the host and put it on reserved address 0. It can also read from any address.

2.2. The DSP-32 processor module

The DSP-32 is a powerful DSμP. If the floating point pipeline is full, it can run at 4 millions multiply accumulates (MACs) per second. The integer section can execute 4 MIPs at peak rate. The device comes in two packages, a 100 pin PGA (pin grid array) with address and data pins for an external RAM and a 40 pin DIP with strictly internal RAM and ROM. Both devices are otherwise identical. The processor has double buffered serial DMA so the processor need not be directly involved with stuffing data into I/O registers. The architecture of the card is shown below:

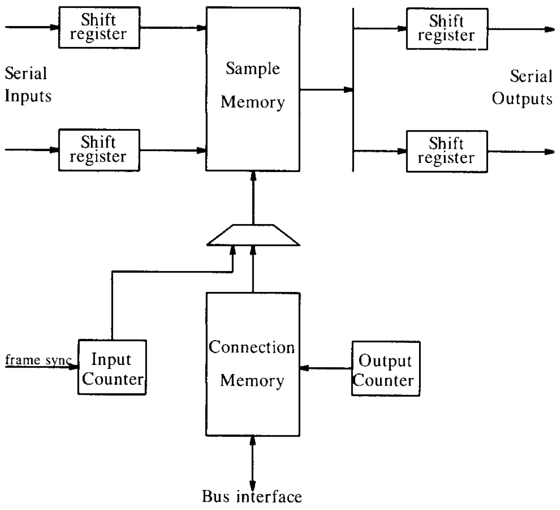


There are two DSP-32s per "processor module". The "master" (with external memory) is connected to the serial lines going to and from the crossbar switch. The "slave" is connected to external I/O devices such as A/D and D/A converters. It is responsible for various I/O operations such as pre and post filtering. The 8 bit parallel data ports of both DSP-32s are indirectly connected to the VME bus. Since the 100 pin DSP-32 is limited to 56 Kbytes of external memory (16 bits of address - 8 Kbytes of internal memory), a memory map expands the available memory to 224 Kbytes. The master DSP-32 can also select which word (load) clocks to use on the transmit side to the switch; this helps limit the transmission bandwidth[2]. The highest 512 bytes of the master DSP-32 memory address are stolen for a memory mapped I/O space; the memory map resides here as do other local I/O devices (such as the transmit word clock select logic).

A bus controller is responsible for access to the internal 8 bit bus. Each processor (as well as the VME bus) must arbitrate for control of the bus. The bus controller also controls the exchange of data between the master and slave parallel ports of the DSPs.

2.3. The switch

The time slot interchange switch uses a commercially available telephone switch, the Siemens PEB2040. Besides being able to operate up to 8 megabits per second, it can be programmed to be either a time division, space division or "mixed" time/space division switch. Internally, the switch looks like:



[2] Although serial DMA doesn't directly involve the arithmetic sections of the DSP-32, it does cause wait states to be inserted.

Data from the serial lines is deserialized by shift registers. The data is then stored in the sample memory according to the address in the write counter. The input counter is synchronized with the frame sync generated by the host interface card. The output counter addresses the sample memory via the connection memory (e.g., the addresses are mapped). The connection memory is, in turn, loaded by the host. Unfortunately, the output connections must be fixed depending on the interconnection scheme chosen (time, space or mixed). A "mixed" space/time switch for 16 lines (processors) at 8 Mbits/second requires 32 switches.

In the worst case, changing a single connection in the switch can take a full frame time. This means that changing the entire topology of the switch is not an action to be taken lightly. However, simple changes can be done relatively rapidly.

3. Programming

The host machine (connected to the VME bus) is responsible for (a) loading each processor with its program, (b) setting up the interconnections in the switch, (c) selecting clock rates on the host interface card and finally (d) returning useful debugging data to the user.

The DSP-32s on the DSP-32 processor modules can be programmed either in C [8] or assembler. However, this forces a user to confront the details of frame sync and other ugly details.

Consider the following brief programming example: To mix the outputs of processors 3,5 and 7 into processor 8, the host must be program the switch to select the data of processors 3,5 and 7 on a given sample and deliver them sequentially to processor 8. The programmer must know what output "time slots" are used by processors 3,5 and 7. The program in processor 8 must know that data from processors 3,5 and 7 contain the data to be mixed.

Future plans call for a task level module interconnection language to hide the details of time slot assignment and use as well as a block level signal processing language.

4. Acknowledgements

Herbert Alrutz and Jim Snyder have given good advice and witty reparteé at times when both were appreciated. Johnsie Chinnery did a fair share of the wiring. Thanks to Max Mathews for encouragement.

5. References

1. J. A. Moorer, The Audio Signal Processor: The next step in Digital Audio, in *Digital Audio*, B. Blesser, B. Locanthi and T. G. Stockham (ed.), AES, 1983, 205-215.
2. M. H. Jones, Processing system for the Digital Audio Studio, in *Digital Audio*, B. Blesser, B. Locanthi and T. G. Stockham (ed.), AES, 1983, 221-225.
3. J. Borish, J. A. Moorer and P. Nye, SoundDroid: A New System for Electronic Post-Production of Sound, *SMPTE Journal*, May 1986, 567-571.
4. A. E. Joel, Digital switch - How it has developed, *IEEE Transactions on Communications COM-27*, 7 (July 1979), 948-959.
5. Serial Transmission Format for Linearly Represented Digital Audio Data, *J. of the AES*, Dec. 1985, 976-984.
6. E. F. Gehringer, D. P. Siewiorek and Z. Segall, *Parallel Processing: The Cm* experience*, Digital Press, 1987.
7. R. Kershaw and al., A Programmable DSP with 32 bit floating point arithmetic, *IEEE Journal of Solid State Circuits*, 1985.
8. B. Kernighan and D. M. Ritchie, *The C programming language*, Prentice-Hall, 1978.