Design for a Fast DSP Machine

Robert McLaughlin
PRA, Inc.
8027 Leesburg Pike, Suite 700
Vienna VA 22182

Abstract

In the following we propose a design for a parallel machine for GHz rate general application Digital Signal Processing, DSP.

Introduction

A DSP is a computational device that has limited instructions that are optimized to do multiplications and additions, for use in digital signal processing. We give a design for parallel DSP that can be built in silicon.

The design we present is based on parallelism. Which is relatively easy to implement in DSP since processes are composed of many sums and multiplications that do not require knowledge of another's outcome to be correct. Avoiding many of the problems generally encountered in the real-world use of parallel machines.

The difficulty encountered in this design is generality versus specific use. In an application such as DSP it is easy to design a chip that will meet a specific set of needs, and will fulfill a particular application in a cost-effective manner. In the following we attempt to develop a general DSP ASIC chip set. Such an adventure has the advantage of letting us see to what extent a dynamicly configurable machine is possible.

Design Parameters

The design goal was to make a machine that can be made out of silicon that can do 1024-point Fast Fourier Transform (FFT) [FFTs need not be multiples of 2. The use of PFA allows us to have FFT's of any length.] FFT's in less than 2 nano-seconds[1], with sustained through-put. This allows the filtering of data in real-time at rates in access of 1-GHz. The design should also to allow multiple FFT's at the sametime; such as 4 256-point FFTs at the sametime.

---

[1] This can be done by the use of ASIC's made out of ECL (6). At lower speeds the design can be implemented in CMOS (7).

To be able to complete 1024-point FFT in 2 nano-seconds one needs 1 butterfly that operates in 400 femto-seconds, or 512 butterflies that operate in 200 pico-seconds or 10240 butterflies that operate in 2 nano-seconds. We assume the need for 10000 butterflies, with other units used for filtering. If an FFT, processing and an IFFT are all required in close to 2 nano-seconds then over 20000[2] butterflies may be needed. Such a machine is shown in figure 1.

The problem that presents itself is not just in designing an optimized butterfly block. But in designing a block that does butterflies in addition to other operations well. For one trades generality for performance. The goal is not just to do FFT's well, but to do the whole gambit of DSP operations well. Meaning we need blocks that can do butterflies, in addition to multiplications, additions and bit-operations. We require an array of multi-stepped ALU's with internal twiddle units.

The Design of the ALU's

In general ALU's are taken to be blocks that perform general arithmetical operations. The blocks we will discuss are similar to this concept. These blocks are designed to be more than just butterflies. One should keep in mind that is cost-effective to divide the ALU's into butterfly and non-butterfly blocks if the usage of the DSP Machine is fixed.

The command unit loads into the ALU's a short program. The ALU's operate on 2 complex numbers at a time, with two complex numbers as the output. The ALU's have the ability to do bit-operations and comparisons. The results of these operations are either outputs or the raising of flags. The twiddle units need to know which point of the FFT they are; this is provided as part of the configuration information loaded in by the command unit.

From the outside the ALU's consist of a command register, two complex number registers for input, two complex numbers for output, and one register of flags for status information and outputs. All of which is standard in the design of DSP chips. It is possible to replace the ALU's with standard DSP chips at a reduced through-put.

---

[2] It is important to reduce the number of units to close to number required seen this will reduce the switch network. Reduction in the size of the switch network will reduce the cost the most.

Switching Network

The purpose of the switching network is to allow dynamic reconfiguration of the system. The switching network is a 5-stage non-blocking network. However if the propose of the machine is fixed, or large sections of the network are fixed, the size of the network is reduced[3].

The switching network represents the greatest single cost of the machine. Infact over two-thirds of the ASIC space is devoted to it. The problem becomes how to the switch network without reducing generality. This is a long standing problem in any parallel machine. It is not separate from the general problem of ensuring that all parts of the machine have the most recent version of the data they need without waiting for some other part.

In DSP it is possible to minimize the amount of data that one ALU needs from another. This is because the structure of butterflies, convolutions and other numerical processes are such that parallelism can reduce the need for data to the data from the structure one back. This is much easier to deal with then, say, differential equations where a solution at one point is directly dependent on solutions from the last iteration at several neighboring points.

We have chosen to use a concept borrowed from telephony of the n-stage non-blocking switch. For the purpose of seeing what was possible we decided to assume that the connections where always in use. It is possible to use multiplexing and greatly reduce the size of the switch by the use of SDM/TDM combined.

It is possible to use many different connection networks for this machine. The one given by Hillis in (8) would be very effective. The connection network we have chosen was done so on simplicity considerations in the Proof-of-Principle stage of design.

Command

It is necessary to have complete control over the entire network. This requires the development of a block that has knowledge of the state of the system and is able to make changes to the network as required.

---

[3] If for example it is known that a 1024-point FFT and 1024-point IFFT are always going operating the switch network is reduced to 1500 points. Reducing the switch network size by roughly 10000.

The user communicates with the command section to set up the DSP machine to solve a particular kind of problem. In this situations the rate at which commands change the state of the machine compared to the rate at which date is expected to pass through is small. Or one can say the commands are static. In a general process machine the commands can be called dynamic. We can use the static nature of the commands to our advantage.

Commands are in one of two groups. The first group consists of commands that configure the switch network the second consists of commands that program the ALU's. We can use this to divide the command block into two parts. One part handles the switch network. The second part programs the ALU's. The command block knows how to combine butterflies so that they are multiplies of 2. It does not know, for instance, how to construct a non-linear filter. This removes some of the intelligence and hence difficulty off the command block.

Summary

In the above I have attempted to describe a DSP machine that is able to handle DSP tasks with high speed and adaptability. It is clear that by limiting the machine to the a configuration ability to handle a certain class of DSP problems the costs of construction go done. However our interest is a most general solution without limits on cost. From this standpoint our device is a useful benchmark.

Because of space the internals of the ALU's, and the Command block has not been dealt with in the depth required. The ALU's however are variants of devices that well documented. While the command block has a structure that is a result of the need to configure the resulting array of ALU's. The actual structure of the command block depends on the implementation of the device.

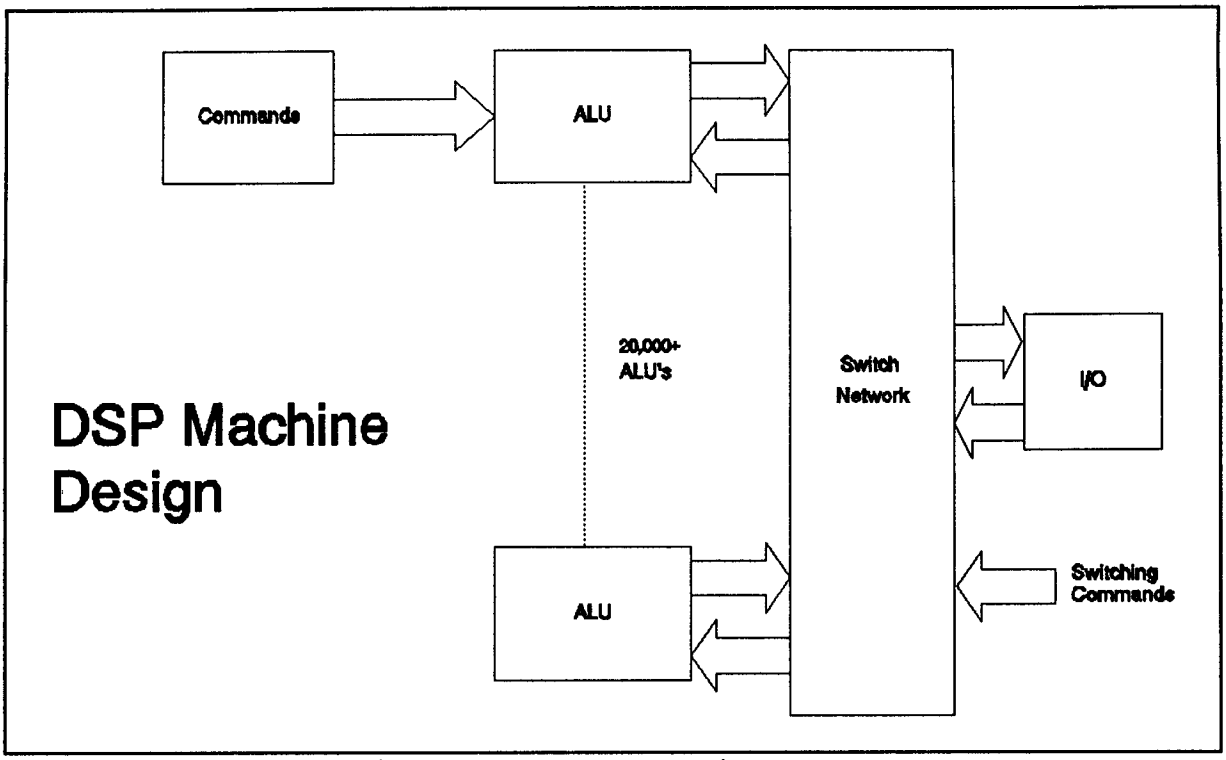

Figure 1 -- Block Diagram of DSP Machine

References

1. C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms*, Wiley-Interscience, NY, NY, 1985.

2. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

3. P. Papamichalis, and J. So, **Implementations of Fast Fourier Transform Algorithms with the TMS32020**, *DSP Applications with the TMS320 Family: Theory Algorithms and Implementations*, Texas Instruments, 1985.

4. J. Bellamy, *Digital Telephony*, Wiley-Interscience, NY. NY., 1982.

5. A. Kamas, and E. Lee, *Digital Processing Experiments*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

6. *Advanced TTL/ECL Gate-Arrays*, ATT Microelectronics, 1989.

7. *CS500 and TGC100 Series Gate-Arrays*, Texas Instruments, 1989.

8. W. Hillis, *The Connection Machine*, MIT Press, Cambridge MA, 1985.