

# Malleable Signal Processor: A General-purpose Module for Sensor Integration

P. McGuirk<sup>1</sup>, G.W. Donohoe<sup>2</sup>, and J.C. Lyke<sup>2</sup>

<sup>1</sup>Mission Research Corporation, Albuquerque, NM 87112

<sup>2</sup>Air Force Research Laboratory, Kirtland AFB, NM 87117-5776

## *Abstract*

The paper presents the Malleable Signal Processor (MSP), a reconfigurable computing module being developed to simplify integration of a wide variety of sensors and actuators into an on-board processing system. Sensors and actuators often require a host of control signals with complex timing relationships. In addition, sensor data usually requires some signal conditioning, such as calibration, format conversion and feature extraction, before it is useable by the host system. The concurrent processing required by interface signals and signal conditioning often exceed the computational capacity of a microcontroller, necessitating custom interface circuitry. The MSP employs programmable logic devices with on-board memory to generate control signals and to condition the data in a programmable module. The MSP's versatility will be demonstrated in a flight system, where it is used to interface to two very different kinds of sensors.

## I. INTRODUCTION

Integrating complex sensors into flight systems is expensive, due to the high engineering costs required to generate control signals for the sensor, to read and condition the sensor data, and to interface to other computational elements. The generation of these signals are a blend of rudimentary, "time-intensive" signals, along with digital waveforms of more sophisticated structure, but all real time. A focal plane imaging array is a typical complex sensor. Command and control signals include frame synchronization, a pixel clock, and region-of-interest extraction. Post-acquisition pixel processing include nonuniformity correction (gain and offset), bad pixel replacement, pixel reordering, and frame buffering. Control for an active sensor, such as a focal plane array with pointing mirrors, requires a feedback control loop that reads mirror position and generates mirror control signals. Higher-level, mission specific functions might include simple pattern recognition and target detection and tracking. These functions, individually relatively simple, constitute a complex system when combined. The set of concurrent processes, with hard-real-time deadlines and complex timing relationships, quickly exceeds the computational capacity of a sequential microcontroller. To bring the computational burden within the scope of a microcontroller, custom peripheral processors can be designed, but this is expensive, time-

consuming and inflexible. There is need for a programmable interface element that combines the real-time processing power and concurrency of hardware with the flexibility of software.

The Malleable Signal Processor (MSP) is being developed as a general-purpose, reconfigurable building block to facilitate the digital aspects of complex sensor integration. The principal design objective is to reduce the time and cost of integrating a sensor to a spacecraft system by providing a reconfigurable hardware module that incorporates programmable logic devices, data memory and a predetermined set of programmable I/O lines on a multichip module. With the MSP hardware available off the shelf, the bulk of the system engineer's design effort is reduced to programmable logic. Using commercial development tools, the designer can complete design-implement-test cycle in minutes or hours, as opposed to weeks or months for custom hardware.

## II. RECONFIGURABLE COMPUTING

The advent of high-performance, high-density field-programmable logic devices has opened up the possibility of computers that can be reconfigured after fabrication to optimize their architecture for the problem at hand [1]. Possibilities can now be discussed for configure these devices at a finely granular level during system integration and after field deployment. Reconfigurable computers combine the benefits of hardware with the flexibility of software. Conventional computers, including microprocessors and microcontrollers, are built on a Von Neumann model [2], shown in Figure 1. The fixed datapath consists of memory, communication busses, and a bit-transforming element such as an Arithmetic-Logic Unit (ALU). The hardware which defines the datapaths and instruction set architecture are fixed at design time. The machine is programmed by selecting source and destination of the data in memory, and by choosing operations for the ALU. Only one instruction can be performed at a time. The enormous success of the Von Neumann model is due to its versatility. However, the fixed-instruction-set architecture is primary oriented towards efficient utilization of the ALU and is not optimal for all algorithms, particularly those that can take advantage of higher concurrency in hardware. In particular, embedded

microcontrollers, which must handle many concurrent processes, quickly bog down in the “Von Neumann Bottleneck”. Modern high-performance processors, including Digital Signal Processors (DSP’s), achieve some instruction-level parallelism through pipelining, but they are still essentially sequential machines. The machine can only handle complex concurrent tasks by multitasking or “context switching” between them, which requires hardware and often software overhead, giving the illusion of concurrency. Increased performance requires increasing processor speed, which comes at a cost: faster devices require more power, and faster switching induces radio frequency and signal line noise.

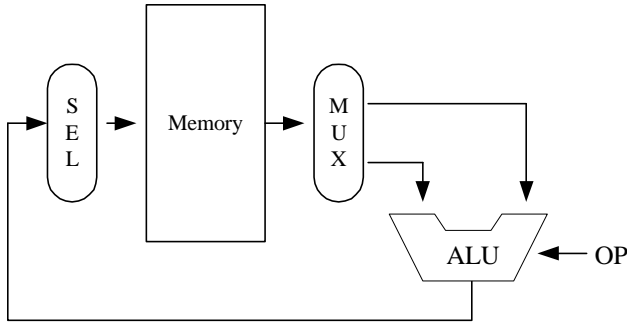


Figure 1. Von Neumann computer datapath.

An alternative to multitasking is to perform these concurrent tasks in a parallel architecture that can rewire its architecture, including datapaths and computational elements (like ALUs), under software control to permit better optimization for the problem at hand. Von Neumann architectures were originally intended to maximally exploit limited numbers of computational resources, which were relatively expensive in the early days of computer science. To think that an ALU could be idle, even for a single cycle, was considered wasteful. Hence, early computers were completely structured around the proposition that such resources would be heavily utilized. Operands could be shuttled to and results shuttled away from an ALU. Reconfigurable computers take advantage of more modern “silicon economics”, in which hardware is no more expensive (and sometimes far less expensive) than software. When possible, tens or hundreds of thousands of gates, can be thrown at specific computation tasks. Where a Von Neumann computer is programmed by changing the ALU operations at different times, reconfigurable computers are programmed by changing the spatial connection of the computation elements. The considerable amount of hardware needed to do this in general is no longer the most dominant consideration [3].

In reconfigurable computing, hardware can be “shaped” to more efficiently and directly to solve problems. Figure 2 shows an architecture for the direct computation of the equation:

$$y = Ax^2 + Bx + c$$

It is convenient to define three different levels of reconfigurability. (1) In *design time* reconfigurability, the connections are programmed once, after the system is manufactured, but before it is used. This model is the normal case for programmable logic devices, and can be an inexpensive alternative to custom integrated circuits. (2) In *static reconfigurable computing*, computational hardware can be built to reconfigure itself at run time in a matter of milliseconds. In order to gain efficiency, it is necessary to exploit natural idle moments, if they exist, or amortize the dead time (milliseconds) between the computational “epochs” that are delineate between different configurations [4]. Each different configuration can solve a different piece of a problem. (3) In *context switching* or *dynamically reconfigurable computing*, a system can be reconfigured in the middle of a computation; for example, during a Fast Fourier Transform (FFT) calculation, part of the circuitry could be used for multiplication, then re-used for addition. This requires very fast configuration, on the order of microseconds or nanoseconds.

### III. THE MALLEABLE SIGNAL PROCESSOR

Despite a significant base of research in configurable computers [4], little work has been done to realize a practical reconfigurable computer for embedded systems. The Malleable Signal Processor (MSP) is a hardware platform consisting of a configurable computing element, memory, a generous supply of input and output (I/O) lines, and a

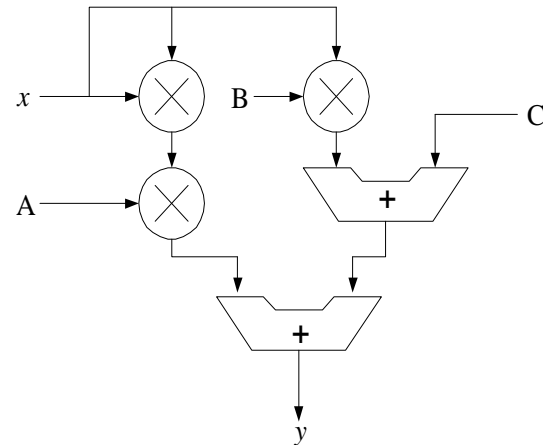


Figure 2. Direct computation model

sequential processor to manage configuration and reconfiguration. The MSP is designed to be reconfigured in-situ during the course of a mission. For example, one configuration can be used for in-system testing prior to flight; a second would be used for in-flight sensor calibration; and a third, for operation. The commercial field programmable gate array devices employed in the MSP design require several hundred milliseconds of total reconfiguration time. As such, MSP is necessarily classified as a static reconfigurable computer. The basic MSP architecture consists of two functional modules: the reconfigurable MSP Core, and the Configuration Management Processor. For high-performance interfaces, an optional embedded version of the Myrinet

(gigabit/sec) link technology has been developed for direct use with the MSP core.

### A. MSP Core

The MSP Core is equipped with two Altera FLEX 10K100 Complex Programmable Logic Devices (CPLD), each with nominally 100,000 equivalent gates. Each CPLD is equipped with 512 K by 24 bits of Static RAM. The native 10K100 devices have 406 I/O pins. In the MSP core

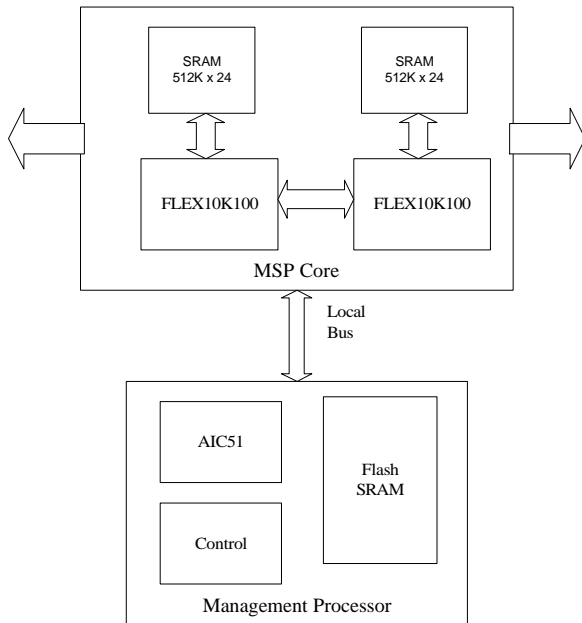


Figure 3. MSP Architecture.

application, some of these are dedicated to memory access and inter-CPLD communication, and others are used for input and output from the MSP core. The CPLDs can be programmed individually. However, each CPLD can not be partially programmed. Reconfiguration requires hundreds of milliseconds, using either a JTAG serial or byte-parallel programming mode.

### B. Management Processor

The MSP Core is not self-booting; this function is handled by the Configuration Management Processor (CMP). CMP provides overall control of the MSP, by handling not only the sequencing of one or more configuration sets for the MSP core, but also by generating the system clock. The CMP is also capable of preserving sensor calibration information, service history, and other state information in non-volatile memory data structures. The CMP is based on the Advanced Instrument Controller, a chip-scale multichip module processor design that employs a specially modified 8051 microcontroller core and on-board memory and I/O, developed for space applications by the Air Force Research Laboratory [6]. Low performance processors are used to emphasize the “background” nature of a CMP. In general, CMPs are used primarily to keep an MSP core in an appropriate operational mode. One megabyte of flash memory contains the configuration files for the CPLDs in the

MSP core, sufficient to hold seven 10K100 configuration files. A local bus serves to connect the MSP Core and the Management Processor. The local bus is designed to deal with the special issues of in situ reconfiguration, in which the logic and routing patterns of the 10K100 devices can be completely reconstituted without loss of memory contents. The provision of a “hold/freeze” signal on the local bus serves as a warning to applications coded into the MSP core fabric (as well as outside the core) that a reconfiguration is taking place.

### C. Embedded Implementation and Malleability

The MSP system is unique among reconfigurable processor designs in that it was intended for real-time embedded applications. Furthermore, the MSP was designed to be miniaturized. In the ensuing “design for package-ability”, it was deemed necessary to minimize component types and quantities, as well as eliminating components that could not be obtained in die form. These constraints, along with the desire to apply the MSP across a range of applications led to a design that seems less aggressive in some respects than today’s technology limits might suggest (it is possible, for example, to obtain programmable devices with 1,000,000 equivalent gates). The principles of maximizing the raw numbers of gates in a processor are not central to the tenets of Malleable Signal Processing. Rather, the MSP is intended to establish a framework for embedded reconfigurable processing. Such a framework, it was felt, needed to provide a maximum level of flexibility for sensor interfacing. As such, it is the ability to rapidly alter the interface to a complex sensor, and not necessarily the aggregation of massive amounts of circuitry, that is important to the principle of a MSP. In principle, if the MSP were used in the front end of a complex system, it would be possible to rapidly interchange one complex sensor for another type without ripping apart electronics boxes and spending months redesigning interface circuitry. The MSP, since it can carry a “palette” of configurations, permits the assessment of several alternatives rapidly. A sequence of configurations can be queued from the palette, and obsolete configurations can be deleted or rewritten.

## IV. TECHNOLOGY DEMONSTRATION

The Malleable Signal Processor will be demonstrated in the Discriminating Interceptor Technology Program (DITP), an autonomous and experimental missile system designed to track bright targets. To demonstrate the versatility of the architecture, the MSP will provide sensing and control functions for two very different complex sensors: a LADAR and a two-color infrared focal plane array (FPA). Since the MSP is limited to digital signals, an intermediate module called the Sensor Adaption Segment (SAS) will be designed to handle the analog signal conversions. The sensor subsystems will communicate with the host through a System Area Network (SAN) adapted from the commercially available Myrinet, a 1.28 gigabit/second network developed to connect processing elements in high-performance

multiprocessing computers. The overhead for a full-blown Myrinet system would render it too bulky for flight, so a scaled-down interface node was designed. Called the Myri10K, it contains an Altera FLEX 10K100 CPLD and a custom Myrinet interface chip, the FI32. A client-server protocol was developed to enable message-based communication between the MSP/sensor modules and the host multiprocessor. Significantly, the MSP and Myri10K units are physically identical for the two sensors; only CPLD configuration software differs. The system will be packed into 2-1/4 inch square multichip modules and stacked, using the High-Density Interconnect Packaging Program (HIPP) technology [5].

### V. DISCUSSION

The fact that the sensor interface and System Area Network without a microprocessor (except for the configuration processor) is a testament to the capacity of dense complex programmable logic devices. Off-the-shelf libraries are available for most signal processing functions, including digital filters, Fast Fourier Transforms, multipliers, adders, coders, etc. The algorithm of Figure 1 can be programmed in a few minutes, and consumes about 7% of the logic resources, and none of the memory, of the Flex 10K100. Without optimization, it executes in about 120 ns. Nevertheless, in order to bring configurable computing into common use, the programming model must be improved. Currently, CPLDs are designed with hardware design tools, requiring intimate knowledge of the device architecture. A design may require several hours to compile. The designer must pay attention to low-level concerns like place-and-route and use of I/O pins. A higher-level design environment will be required to bring this technology into common use.

It is also clear that reconfiguration, while an advantage, requires that deliberate attention be made to the configuration process itself, particularly for static reconfigurable systems. Typical problem cases include the disruption of data memory contents and the introduction of faults in peripherals. The reconfiguration process cannot detract from the real-time nature of a system if in situ changes are required. The introduction of a “hold/freeze” signal serves as one mechanism, which warns other parts of the system that a reconfiguration is about to occur. Under those conditions, data transmissions into the MSP core are suspended, and MSP core outputs are ignored (assumed invalid). In systems with

multiple reconfigurable devices, it may be necessary to establish a protocol of “hold/freeze” signals, particularly if reconfiguration of only particular devices and not the entire network is desired.

It is important to consider how such technologies and architectures would benefit aerospace systems. In the first place, it is evident that reconfigurable computing allows one to evolve and refine concepts of on-board processing in systems even *after* they have been fielded. In space systems, the reconfiguration can be done while the platform is in orbit. This is important in ideas involving a distribution of several satellites of a homogeneous constellation, but perhaps even more so in a heterogeneous collection of satellites, which allows us to continue to extract new mission roles and capabilities from space assets. In some cases, these new roles and capabilities might not have been foreseen at the time of initial deployment. This allows users to continue to extract through innovation much more utility from every space asset for as long as possible.

Second, it is important to consider the ability to rapid reconfigure assets to achieve a sort of tailor-able “processing-on-demand”. If information is power, then it must certainly be the case that the dynamic re-constitution and delivery of that information is even more power. Roles and missions of various classes of embedded systems can be re-focused rapidly at will through reconfigurable technologies.

Finally, reconfigurable capabilities can be channeled in ways that improve the robustness of systems. New concepts in fault-tolerance would be possible.

### VI. CONCLUSION

The Malleable Signal Processor is perhaps the first instance of reconfigurable computing to be demonstrated in space. The fact that the hardware for the MSP Core, the Configuration Processor, and the Myri10K is identical for both the focal plane array and LADAR subsystems is a demonstration of cost savings through hardware reuse, enable by reconfigurability. The MSP is capable of in-system reconfiguration for testing on the ground, and for in-flight context switches, from calibration to operational modes, for example.

With current technology, CPLD-based configurable computing is limited to design-time and run-time reconfiguration. Tens or hundreds of milliseconds are required

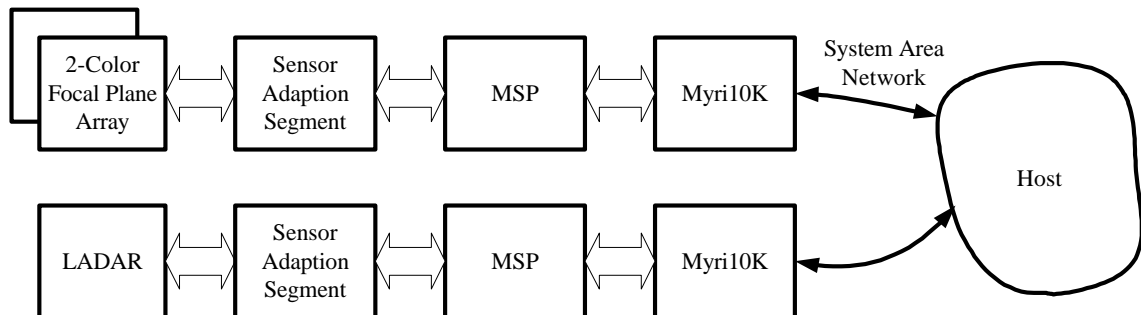


Figure 4. Sensor subsystems of the technology demonstration.

to perform a context switch. This is too long for many applications: the vehicle is essentially blind during the reconfiguration. The ability to perform partial reconfiguration would allow the vehicle to function while changing its program, and high-speed reconfiguration would enable run-time adaptation.

Currently, the benefits of reconfigurability are limited to the digital portion of the system. The analog portions of the sensor interface must be custom designed. The ability to extend reconfigurability to analog would reap great rewards in reduced design and integration time, and in in-flight versatility.

The Malleable Signal Processor is a first step toward reconfigurable systems for space applications. Experience with it will point the way for future research.

## REFERENCES

- [1] Villasenor, J., and W.H. Mangione-Smith, "Configurable Computing", *Scientific American*, June, 1997.
- [2] Patterson, D., and J.L. Hennessey, *Computer Organization & Design*, 2<sup>nd</sup> Ed., Morgan Kaufman, 1997.
- [3] Wilson, A., "Gate-Array-Based Adaptive Computing Speeds Image Processing", *Vision Systems Design*, September 1998, pp. 66-76.
- [4] DeHon, A., "Reconfigurable Architectures for General-Purpose Computing", AI Technical Report 1586, MIT Artificial Intelligence Laboratory, October 1996.
- [5] Lyke, J. "Highly Integrated Packaging and Processing", Government Microcircuits Application Conference (GOMAC) Digest of Papers, March 1999, Monterey, CA.
- [6] Lyke, J. "Design Of A Power-Optimized Micro Miniature Advanced Instrument Controller For Sensor Craft Applications", *Proc. of the AIAA/IEEE 15th Digital Avionics Systems Conference (DASC)*, Atlanta, GA, October 27-31, 1996, (IEEE Press Inc., Piscataway, NJ), vol. A96-45660, 1996.