

Motion Control Computing Architectures for Ultra Precision Machines

Mile Erlic
Precision MicroDynamics, Inc.,
#3 - 512 Frances Avenue, Victoria, B.C., Canada, V8Z 1A1

INTRODUCTION

Several computing architectures are used in the industry and in academic and government research labs for control of ultra precision machines. The issues involved with such architectures will be discussed in this paper.

For maximum benefit, a motion control computing architecture should assist the ultra-precision machine tool design in achieving its primary goals, which are: increased performance in terms of form accuracy, surface finish and machining cycle times; improved reliability and security; faster development cycles; and unhindered scaling as the scope of the application grows.

Two hardware architectures and three software architectures are presented. The hardware architectures are shown in Figs. 1 and 2 and the software architectures are shown in Figs. 6, 7 and 8.

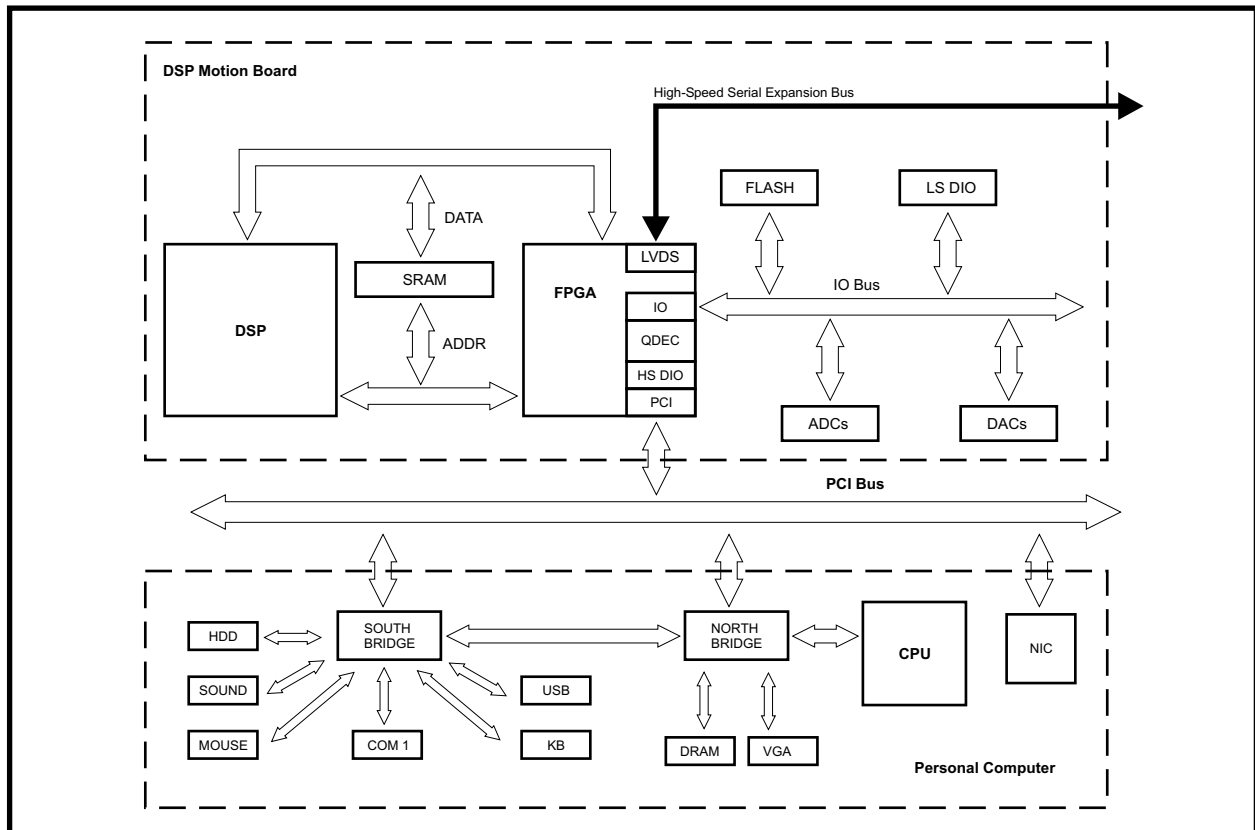
The hardware architecture in Fig. 1 represents an asymmetric dual processor motion control computer. The two processors used are a specialized digital signal processor (DSP) and a general purpose central processor unit (CPU).

The DSP is a device optimized for vector processing of integer and floating point data types; whereas, the CPU is a device well suited for decisions, branching and for balanced processing of integer, floating point and character data types.

The hardware architecture in Fig. 2 represents a single processor motion control computer. All of the data processing is accomplished by the general purpose CPU.

The motion control interface is the same in both hardware architectures. The central component in the motion control

Fig. 1: Asymmetric Dual Processor Motion Control Computer



interface is the field programmable gate array (FPGA) which manages the motion control input and output (IO) resources. The keyboard, mouse, sound and video are managed by the CPU.

The "DSP Motion Control" (DSP-MC) software architecture shown in Fig. 6 and "DSP Servo Host Motion Control" (DSP-SERVO/HOST-MC) software architecture shown in Fig. 8 run in the asymmetric dual-processor hardware architecture shown in Fig. 1.

The Host Motion Control (HOST-MC) software architecture shown in Fig. 7 runs in the single processor hardware architecture shown in Fig. 2.

In DSP-MC control, the real-time motion control functions run in the DSP and the CPU behaves as a host that runs a non real-time motion event processor and a graphical user interface (GUI).

In HOST-MC control a real time operating system (RTOS) is used. The real-time motion control functions run in the CPU as do the motion event processor and the GUI.

In DSP-SERVO/HOST-MC control a RTOS is used. The motion control functions are split between by the CPU and the DSP, each doing what it does best. The DSP runs the

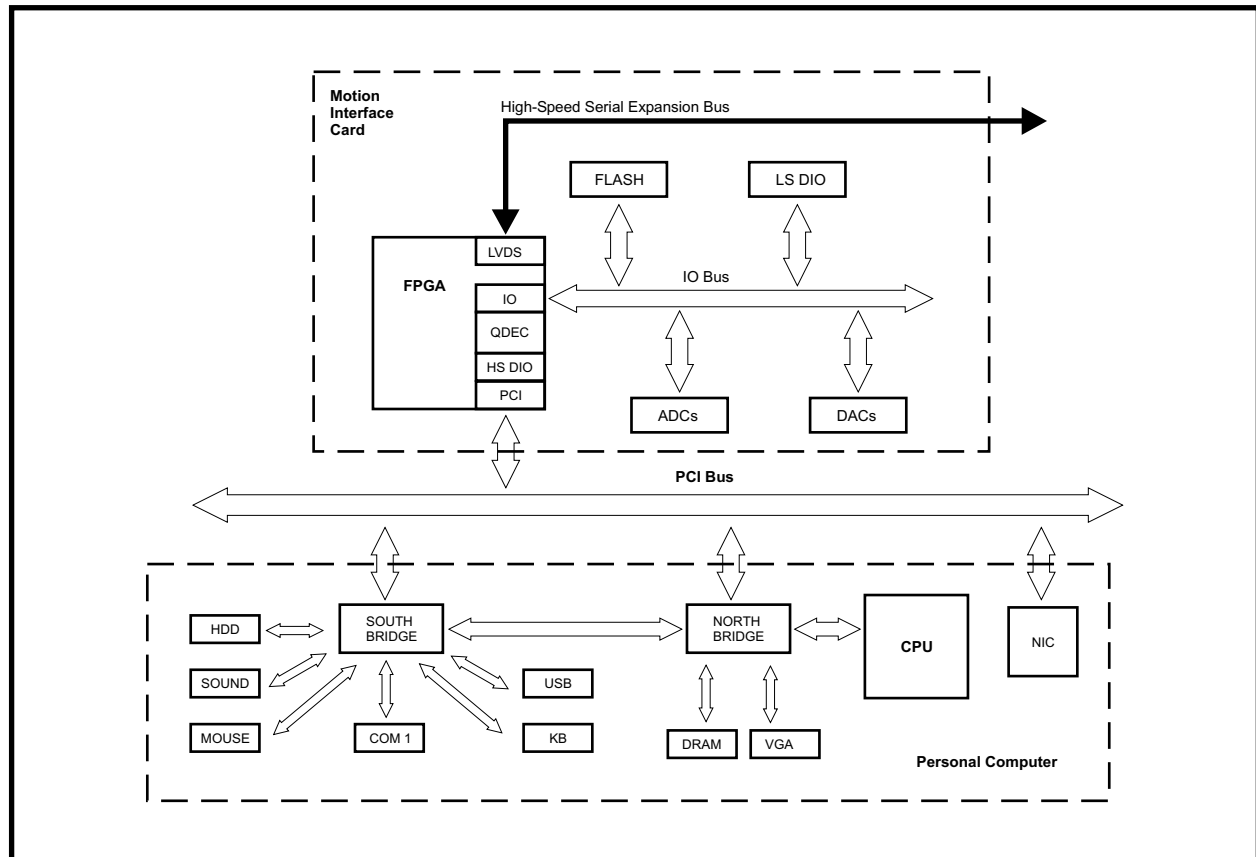
commutation, servo loop, and fast tool servo and the CPU runs the motion event processor and GUI.

HARDWARE ARCHITECTURES

In both of the two hardware architectures (Figs. 1 and 2), the FPGA is used to implement IO for motion control functions and it is used to implement the PCI bridge. The motion control IO consists of quadrature decoders (QDECs), high-speed digital inputs and outputs (HS DIO), low-speed digital inputs and outputs (LS DIO), digital to analog converters (DACs), and analog to digital converters (ADCs). National Semiconductor's Low Voltage Differential Signal (LVDS) hardware communication protocol is used to implement a high-speed serial expansion bus for communication to auxiliary IO modules, sin/cos interpolators or customer specific devices.

The quadrature decoder is a key component in motion control systems. A quadrature decoder counts digital pulses originating from a quadrature position encoding device. The decoder implements an up/down counter and creates a digital word that represents the angular or linear displacement of a motion axis. The decoder may include register preload, capture or compare functionality. For capture, the current state of the count register is recorded on a HS DIO input event. For compare, a HS DIO output event is triggered when a register count is identical to a

Fig. 2: Single Processor Motion Control Computer



stored value.

The DAC outputs are used to signal dc motor amplifiers; the LS DIO are used for home, left and right limits and emergency stop; and the ADCs are used for position and velocity transducer inputs.

Communication between the motion control IO system and the general purpose CPU occurs via PCI bus. The CPU provides access to keyboard, mouse, video, sound, mass storage and other peripherals.

In a single processor motion control architecture, the general purpose CPU performs all software operations. In an asymmetric dual processor architecture, software operations are divided by the CPU and DSP.

SOFTWARE ARCHITECTURES

A DSP-MC software architecture (Fig. 6) is necessary when a non real time operating system such as Microsoft Windows or Linux are used on an asymmetric dual processor motion control system (Fig. 1). The real-time functions must be computed by the DSP since the OS is not deterministic.

A HOST-MC software architecture (Fig. 7) using a real time operating system such as QNX Neutrino or VenturCom RTX is necessary for single processor motion control systems (Fig. 2). The real time functions can be computed by the CPU since the OS is deterministic.

A DSP-SERVO/HOST-MC software architecture (Fig. 8) is used on an asymmetric dual processor system (Fig. 1). This architecture delivers optimal performance. A real-time operating system such as QNX is used on the CPU and the DSP runs the faster real-time motion control processes such as the PID loop.

Motion Control Flow

Fig. 6 graphs the DSP-MC software architecture. The rates shown are typical and can vary from one design to another. The GUI and user application run at a relatively low rate, but sufficiently fast for interface to a human. The path planner, setpoint server, controller and data server processes run at the highest rate and the monitor process runs at an intermediate rate. The motion event processor receives messages from the monitor and is interrupt driven.

The user application sends path descriptions to the path buffer. The path planner pulls paths from the path buffer and generates set points that are consumed by the controller. The controller compares the setpoint positions and velocities to actual positions and velocities read from the QDEC or ADC registers. The controller then sends corrective signals to the DACs, completing the servo feedback loop.

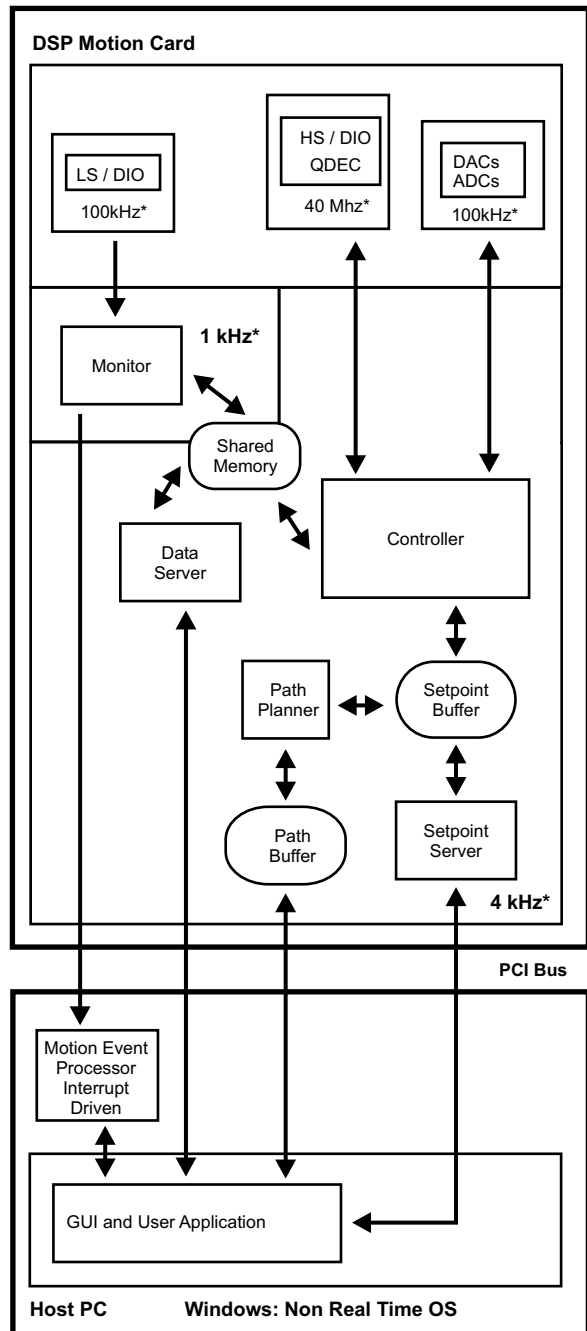
The data server is used to communicate kinematic and system variables in the shared memory to the user application. The monitor is responsible for communicating

error states such as hardware limits and other events such as motion done to the motion event processor. In certain applications, the path planner is written by the user and is executed in the CPU. In this case the path planner is bypassed and setpoints are communicated directly to the setpoint server.

DSP-MC Software Architecture

In DSP-MC control, motion control computations are performed in the DSP and the CPU is used for the GUI and

Fig. 6: DSP Motion Control (DSP-MC) Software Architecture



*Frequencies shown are typical and can vary across designs.

motion event processor.

The biggest advantage of this architecture is the familiar programming environment and wide selection of software development tools such as Visual C++ and Java. There is a large pool of programming labor from which to draw. The disadvantages are performance based. The DSP is a specialized vector math processor but it runs at slower clock frequencies than a general purpose CPU. The DSP is highly optimized for vector integer and floating point operations but performs less optimally on text processing, decision making and branching.

At this current time, DSPs have native support for single and extended single precision data types only. These have an accuracy of 1 part in 2^{24} for the former and 1 part in 2^{32} for the latter. See Figs. 3-4 for floating-point IEEE standards and Table 1 for range and accuracy data [Goldberg 1991].

If single precision is used, then 1mm accuracy is maintained over a 16mm range. Beyond this range, the accuracy reduces to 2nm at best. See Tables 1 and 2.

If extended single precision is used then the range for a 1mm accurate system is 4m.

Fig. 3: IEEE Single Precision, 32-Bit

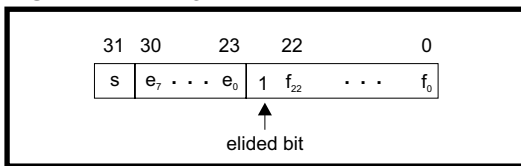


Fig. 4: IEEE Extended Single Precision, 40-Bit

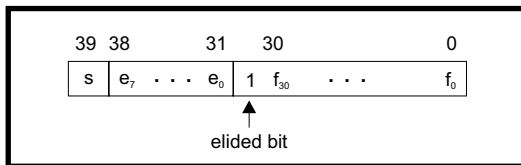
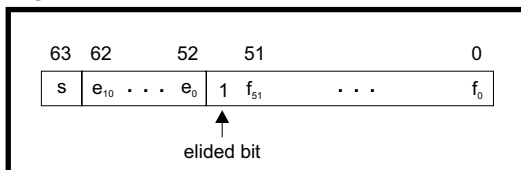


Fig. 5: IEEE Double Precision, 64-Bit



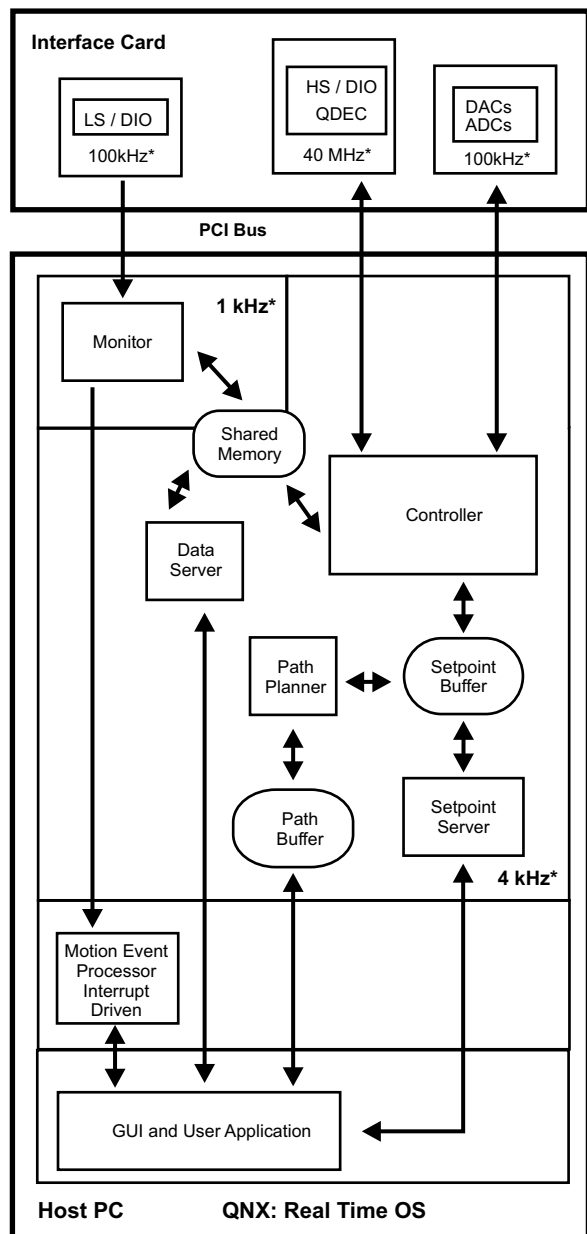
If a non real time OS is used, then the communication between the user program residing on the CPU and the real time motion control processes residing on the DSP is not real-time. Motion commands sent by the CPU are not guaranteed to transmit to the DSP within a specified time constraint.

A test was conducted in the Window XP environment. In

7.3 million cycles it was found that the average latency to a motion event was 4.8ms. Latencies were 11ms or less 99.96% of the time. Worst case latencies as high as 29ms were recorded. The test was conducted on a Celeron 400MHz. The only running application was the test code.

In some motion control systems, 5 ms latency for response to an event can be tolerated. In certain applications, a latency even as high as 29ms can be tolerated if it occurs rarely. However, in ultra high performance systems these latencies cannot be tolerated and a RTOS must be used.

Fig. 7: HOST Motion Control (HOST-MC) Software Architecture



*Frequencies shown are typical and can vary across designs.

CPU-BASED Software Architecture

In HOST-MC control, the motion control computations, GUI and motion event processor are all performed in the CPU. A real-time operating system such as QNX is employed. Tasks are guaranteed to complete within specified intervals of time. QNX context switch times are typically less than 10 microseconds.

Table 1: Representable Ranges

	Representable Range (1nm accuracy)	Typical Representable Number
Single	16mm	15.000001 mm
Extended Single	4.0m	3.000000001 m
Double	9x10 ⁶ m	8,000.0000000000001 km

Table 2: Decimal to Binary Conversion

Decimal Number	Single-Precision Binary Floating Point Less Elided Bit
16,383.998	46 7FFFFE
16,383.999	46 7FFFFF
16,384.000	46 800000
16,384.001	46 800001
16,384.002	← 46 800001

One disadvantage of the HOST-MC architecture compared to the DSP-MC architecture is the smaller number of real-time programmers that are available in the labor market.

A second disadvantage is the fact that the CPU is a shared resource for all processes running. As such if vision or other compute or communication intensive systems are added, performance degradation may occur.

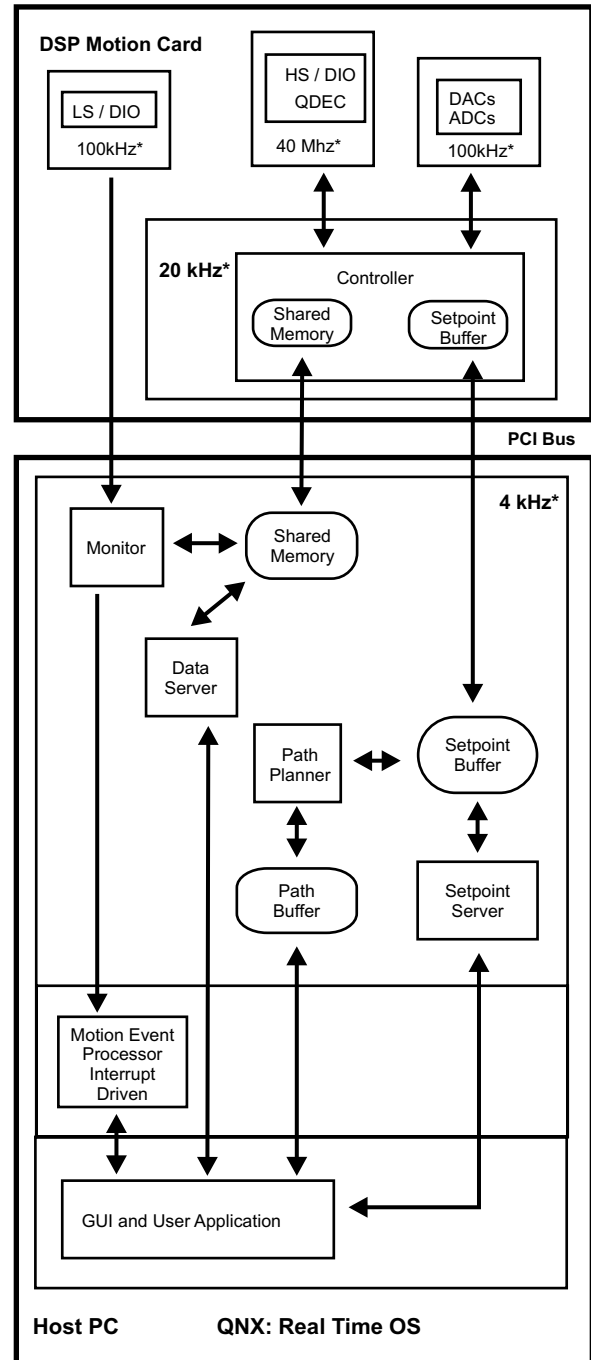
The HOST-MC architecture provides balanced performance and offers a single unified software environment in which to work. QNX Software Systems provides an advanced programming environment similar to that found in UNIX systems. Other real-time OS environments available on the market include Wind River VxWorks and VenturCom RTX.

In a QNX environment, a programmer can develop the application while running the machine and collecting data all at the same time, on a single computer. This would not be possible in a non real-time operating system environment. For example, in Windows, a CNC machine

should not be operated during software development. A compiler operation could introduce non-deterministic computer timing that could block user access to the machine and result in machine damage.

In HOST-MC control, the general purpose CPU is able to process character data types as easily as it is able to perform floating point operations. Parsing character data types as found in G-code files are optimally handled and

Fig. 8: DSP Servo Host Motion Control (DSP-SERVO/HOST-MC) Software Architecture



*Frequencies shown are typical and can vary across designs.

the double floating point type is natively supported (see Fig. 5 and Table 1). In double float a range as high as 9×10^6 m may be represented with 1nm accuracy.

The reliability of a real-time OS such as QNX is another benefit. In a 911 emergency software system that uses QNX, a 99.9999% availability has been reported.

With a mission critical software system such as QNX, there are no critical updates. There are several upgrades per year but these are optional and are only needed if the new OS features are desired.

QNX and the other real-time OSs are much less susceptible to malicious virus attacks. These are not the favorite targets for hackers. The OS kernel is de-coupled from the device drivers and applications, making it practically impossible for hackers to access critical software areas. There are no known viruses or worms that have attacked a QNX system.

The HOST-MC software architecture is more scalable than the DSP-MC architecture. A DSP has relatively small volatile and non volatile memory access and storage. In HOST-MC control, volatile and nonvolatile memory storage is not practically limiting. Furthermore, symmetric multi-processing is naturally supported by QNX with its message passing protocol. Additional CPUs may be added for the application seamlessly using this protocol.

DSP-SERVO/HOST-MC Software Architecture

The software architecture that provides the possibility for highest performance is the DSP-SERVO/HOST-MC software architecture (see Fig. 8).

One disadvantage of this architecture is the increased complexity of environment. The developer will need to use separate environments for DSP and CPU programming.

A second disadvantage of the DSP-SERVO/HOST-MC architecture is the higher development expensive since a development system is needed for both DSP and CPU environments.

From a performance point of view there are only benefits. The communication between CPU and DSP is deterministic since a real-time OS is being used. Data is shared reliably and seamlessly.

The DSP runs the high-speed controller processes. Some of these processes may include PID feedback regulation, lowpass and notch filtering, velocity estimation, brushless motor commutation and fast tool servo.

The CPU executes the path planner, data server, monitor, motion event processor, GUI, and user application. The user application may execute other systems such as vision.

This division of labor between the DSP and CPU provides for optimal processing. High-speed functions such as fast tool servo and PID run very quickly on the DSP and G-

code files can be parsed efficiently by the CPU.

Excellent reliability, scalability, and security are features of the DSP-SERVO/HOST-MC architecture..

CONCLUSION

In the introduction of this paper it was presumed that the function of the motion control computer architecture was to facilitate increased performance in terms of form accuracy, surface finish and machining cycle times; improved reliability and stability; easier serviceability; faster development cycles; enhanced connectivity; and unhindered scaling as the scope of the application grows.

Subsequent facts and observations have demonstrated that the asymmetric dual processor computer hardware architecture running a combined DSP-SERVO/HOST-MC architecture satisfies these objectives.

This architecture delivers the high speed, range and accuracy necessary to provide the best form accuracy, surface finish and high cycle times.

The use of a real-time OS such as QNX delivers the other advantages. Compared to a non real time OS such as Windows, a real-time OS provides greater reliability and availability; it is much more secure against computer viruses and worms; it provides a unified development environment ideally suited for "develop and test" in the same computer; and it provides scalability to symmetric multi-processing computer systems.

ACKNOWLEDGMENTS

I have spent hundreds of hour discussing real time motion control with my colleagues at Precision MicroDynamics. Without them, this article would not be possible.

REFERENCES

- Analog Devices, 2000, *ADSP-2106X SHARC DSP Microcomputer Family*, http://www.analog.com/UploadedFiles/Data_Sheets/311997530ADSP-21061_L_b.pdf
- Furr, S., 2002, What is Real Time and Why Do I Need It?, <http://www.qnx.com/download/feature.html?programid=8090>
- Leroux, P.N., 2002, Real Time or Real Linux? A Realistic Alternative, <http://www.qnx.com/download/feature.html?programid=8085>
- Goldberg, D., 1991. What Every Computer Scientist Should Know About Floating-Point Arithmetic, In *ACM Computing Surveys*, Vol 23, No 1, pp 5-48.