Serial Memory Interfaces: Trends and Options

by Carlos Martinez, November 1999

Introduction

In the dynamic semiconductor industry, conventions taken for gospel today are heresy tomorrow. In one of these classic battlegrounds, parallel busses, once undisputed leader in system data transfer, face a serious challenge from new serial interfaces. Recent introductions and improved standards make the serial interface an attractive choice for both memory and peripheral control applications. In the peripheral arena, IEEE1394, gigabit ethernet and Universal Serial Bus (USB) are gaining favor in many systems over parallel busses such as PCI, PCCard, IDE (disk drive), and ISA/EISA.

In the semiconductor memory world, code storage devices almost exclusively use the parallel bus. However, serial access data memories now find increasing acceptance as faster serial busses allow the designer to achieve the benefits without the drawbacks. This application note examines serial memory interface trends and evaluates the three leading interface standards.

Memory Density

As Parallel Flash memories and E^2PROMs increase in density, manufacturing lower density devices becomes increasingly less cost effective, so these densities tend to “disappear”. This provides opportunities for serial memories to increase in density to “fill the gap” left by the discontinued parallel types. These serial memory devices prove ideal for many data storage applications.

The Emergence of Serial Memories

Serial memory devices have been around for more than 14 years. The early low density devices found their way into many applications and, because of their low cost, sold in high volumes. Typical early uses included system configuration or device identification. These application generally required very little storage and had few interface speed constraints. In recent years, the use of serial nonvolatile memories has increased at a phenomenal rate into applications ranging from cellular telephones to LAN systems to automobiles to industrial equipment. In many of these applications serial memories are beginning to displace parallel devices.

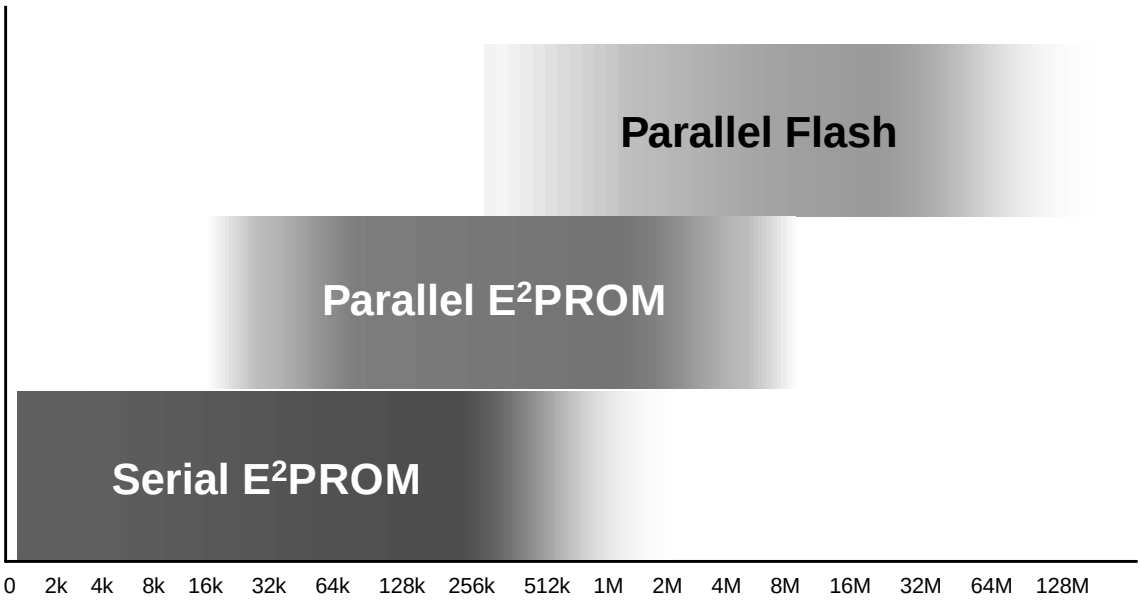

Figure 1. Trends in Memory Density

Serial memories began to make inroads against parallel devices in response to two related market requirements. First was a rapid growth of portable products, as demonstrated by the proliferation of laptop computers, palmtop computers, pocket cellular phones, pagers, etc. Portability needs drove component vendors to find ways to reduce the size and power of their product. Success in this area fueled more miniaturization and the trend points to continued miniaturization in the foreseeable future.

As devices become more portable, there is a growing desire by consumers to customize the product that they use. Products now are small enough that their owners carry them everywhere. They become part of the person's "life-style". When this happens, the product must adapt to the owners preferences and habits. This leads to an increasing need for user programmable customization.

As devices get smaller, there are different interface media, since a keyboard is not always practical. This leads to voice or pen inputs. To optimize performance the system must adapt to the user or be capable of adjustment for more desirable response.

Customization in a system also includes personal phone numbers, individual schedules, stylized sounds or pictures. To personalize a product the user might include records of credit card numbers, access code lists, medical history, insurance information, etc. As personalization and customization becomes more common, there is a desire for more. This results in an ever growing need for more nonvolatile memory.

One way to provide customized memory is to use a small piece of a large parallel Flash device often seen in many products. The Flash contains the operating code and can be updated in the field. Why not reserve a little of this space for data storage? The emerging consensus is that loss of some personalization data is manageable, but loss of program is catastrophic. Allowing writes of personalization data to the same device that contains the program code increases the likelihood of these catastrophic program failures in the field.

With each new generation of product, engineers find new ways to reduce the physical size of the unit and to reduce it's power, while at the same time increasing the performance. To do this, the engineer uses device integration (through the use of ASICs and more highly integrated semiconductors), smaller packages, lower

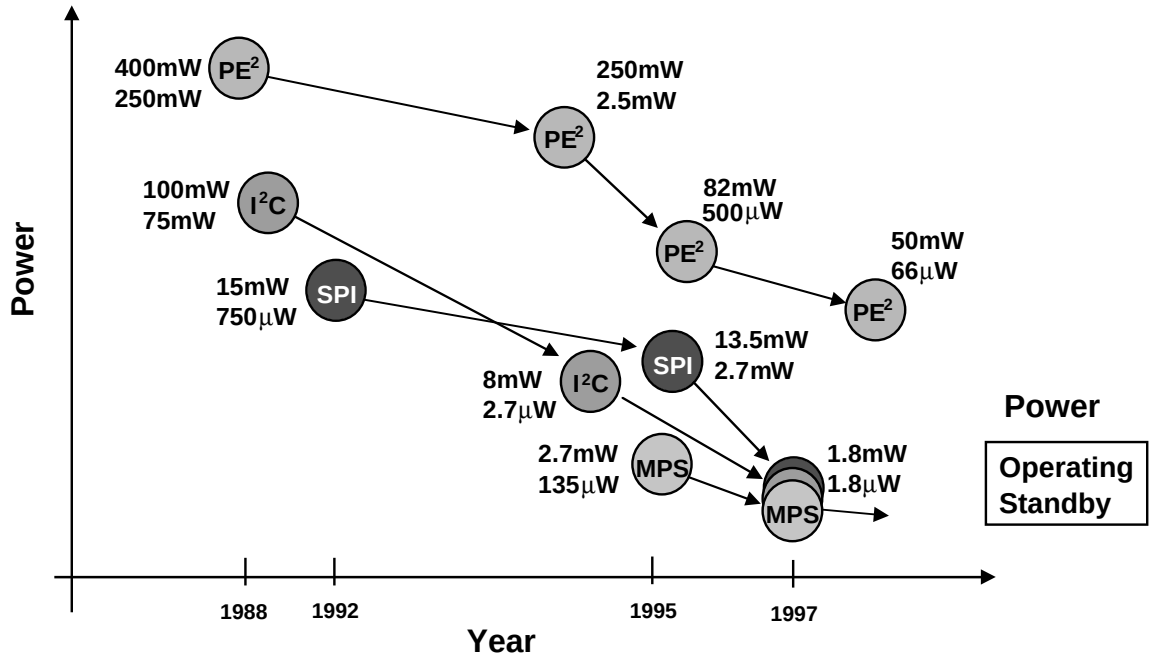

Figure 2. Memory Power Trends—With each new generation of devices power consumption is reduced. Future devices will have zero standby power

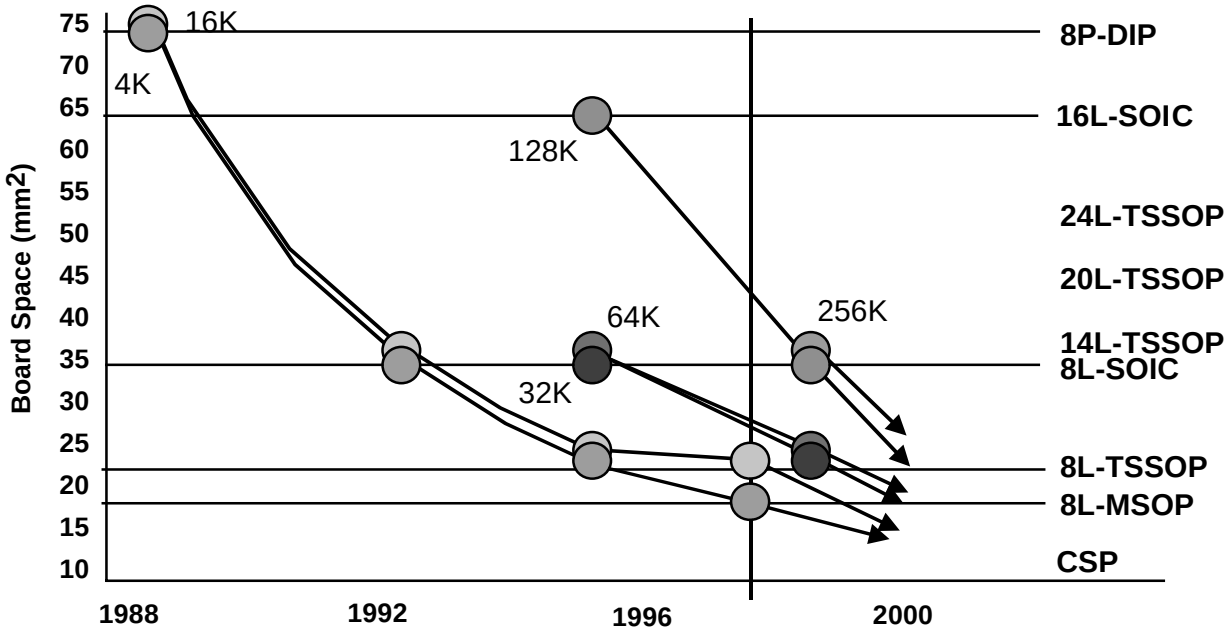

Figure 3. Memory Density/Package Trends—As process technologies shrink device geometries, more memory can fit into smaller packages. The ultimate package is the chip itself. Chip Scale Package technology, in development now, promises to radically reduce future package sizes.

operating voltage, lower current and innovative packaging. Serial memories are becoming increasingly important in managing programmability, board space and power. Historical trends show serial memory power requirements (Figure 2) and memory density per square millimeter (Figure 3) decreasing rapidly, with the trend continuing into the near future. Because of their small size, low power characteristics and isolation from program memory areas, serial memories are key in providing new portable solutions.

While serial memories seem to be the ideal solution, they have been troubled by two limitations, the limits of speed and the need for special serial ports.

The Importance of Speed

Two new serial peripheral interface standards, the IEEE1394 and USB, are examples of an increased push in the industry toward higher serial bus speeds. Promotions for the 1Gbit IEEE1394 standard indicate that it may be the preferred choice over the parallel PCI bus in the PC because of data throughput. At 1Gb/s, the IEEE1394 serial bus transfers 64-bits of data in less than 65nS. This is reportedly faster than current implementations of the parallel PCI bus, because of PCI protocol overhead. Designers predict the USB will replace the ISA bus and both the PC parallel and serial bus in the PC for low speed peripherals. The USB handles up to 12Mb/s. This is fast enough to handle audio I/O, telephone interfaces, keyboard and mouse control, printers and other standard PC peripherals. The USB is thought to have the inside track over other serial busses like ACCESS.bus (a derivative of I^2C), mainly due to it's much higher speed.

In serial memory applications speed becomes more important as the density increases. Reading 64kbits, at 100kHz takes over 600ms seconds, while a 10MHz interface takes only 6ms. This can make a big difference in system performance, power consumption and user interaction.

Slow interface speeds impact system performance by adding overhead to the CPU. The need to wait for the next bit strains the ability of the processor to deal with real time activities or the demands of ever more complex software. This overhead can also affect the user of a product. The average person perceives a delay with response times in excess 200 to 300mS. Faster response times do not interrupt thought processes and this gives the user a more solid feel about the product.

Interface speed, coupled with low standby current, can increase the battery life of a product. In many applications, a product operates at full power only a fraction of the time. Most often it is in some standby, low power state. Executing an operation faster reduces the time a product is in full power mode, allowing it to remain in a standby state longer (Figure 4).

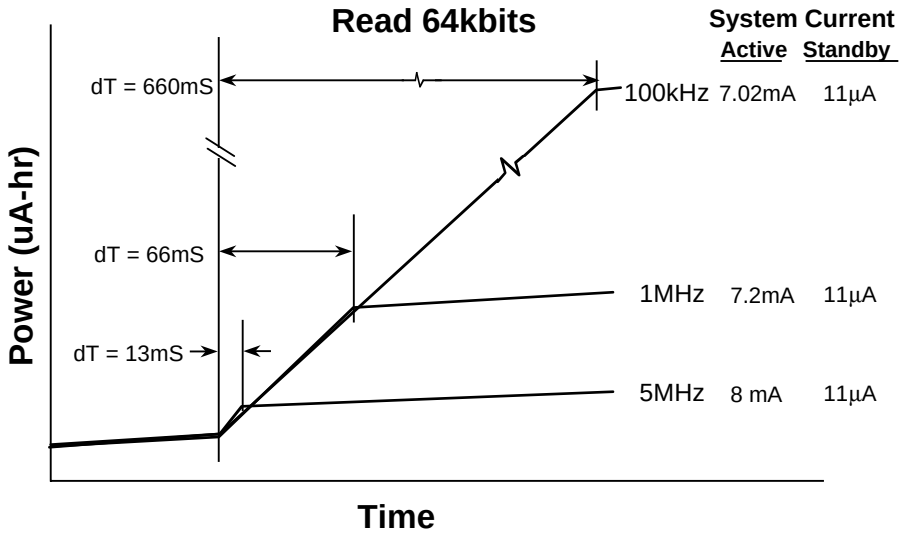

Figure 4. Power vs. Interface speed. This figure shows an example of a processor that draws 7mA active and 10µA in standby. The serial memory current increases linearly as the speed increases. Having the processor active for less time reduces system power, demonstrating one value of a high speed serial device

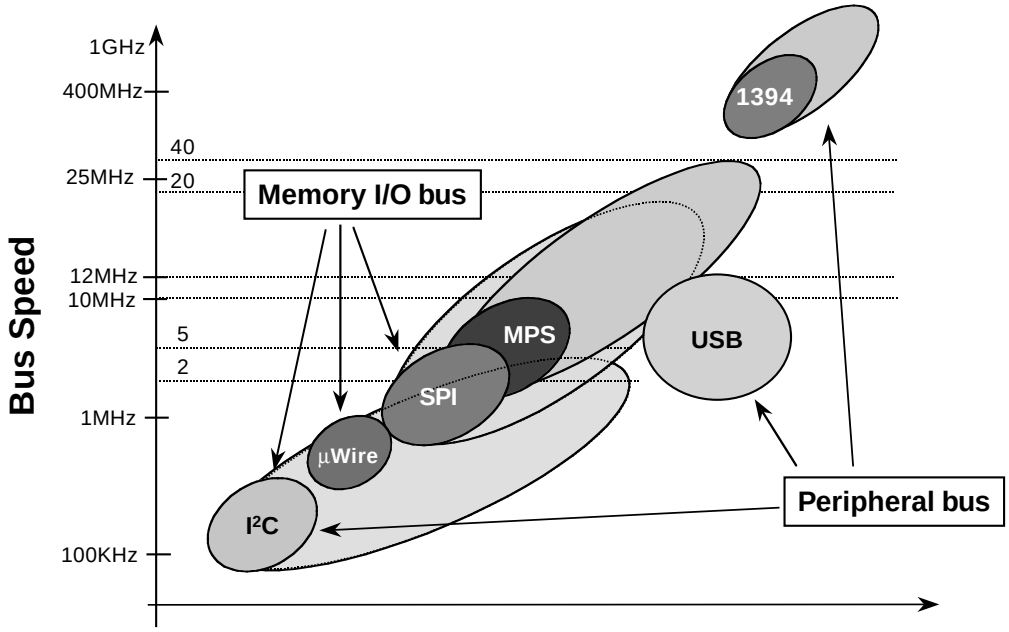

Figure 5. Serial Bus Speed Trends

Recent high speed serial memory product introductions from Xicor have generated tremendous interest. This is an early indication that the simple serial memory interface speeds will track the serial peripheral bus trends. The recent product introductions from Xicor include 400kHz 2-wire, 5MHz SPI and 10MHz MPS interfaces. Figure 5 shows a graphical representation of current and projected serial bus speeds. Trends for the SPI and MPS busses are based on projections for expected host processor hardware capability and memory I/O design in the near future.

The Significance of I/O Port Limitations

The second limitation of serial memories is in the need for dedicated serial ports or the use of general purpose serial ports to communicate with the serial memory. In

many systems there is a need for all the ports possible for system management or user interface functions. Giving up even two general purpose port pins can cause serious design restrictions.

With the need for ever smaller products, there is a growing use of ASICs and more highly integrated controllers in new designs. These new ASICs and controllers often have built in serial busses. The most common of these are I^2C and SPI. These memory interfaces off-load the general purpose ports for other functions and provide higher speed and more code efficient interfaces. However, it does this at a price. To put a serial port in an ASIC requires extra silicon, additional pins, more design and design verification resources and more testing. The addition of a serial port on a microcontroller is often associated with other "bells and whistles", driving up the cost of the controller. There are systems where the benefits of serial hardware justify the extra expense, but there is another option. As will be discussed later, a new interface from Xicor, called MPS™, allows connection of serial memories without the need for dedicated serial ports.

Engineers designing with microprocessors have a slightly different problem. Serial memories do not connect to microprocessors, since microprocessors have no serial ports. In order to connect a serial memory, the engineer must add a serial bus master to an ASIC or add a parallel to serial converter and memory map it into the address space. This added complexity, cost and board space precludes the use of standard serial memories. To give engineers a simple "no overhead" serial memory for microprocessors, Xicor introduced the MPS™ interface. This interface works by using only 4 pins of the already present parallel memory bus.

Interface Options

The earliest memory busses were the microwire bus and the I^2C bus. The microwire bus began as an interprocessor communication port for the COPs microcontroller. The I^2C (Inter-Integrated Controller) bus began as a peripheral bus, meant to interface to many devices with a single set of two wires.

SPI is a Synchronous Peripheral Interface that was created by Motorola. This interface is a more recent introduction that is proliferating as high speed becomes more important and as more products exploit the interface.

Perhaps the ultimate interface is one created by Xicor. This interface, called MPS, provides a high speed serial solution for products that do not have ports available for conventional serial memories. Because MPS makes use of the standard system memory bus, it interfaces to most microcontrollers, microprocessors, DSPs and ASICs, with no additional hardware.

The Microwire Bus

As one of the oldest serial busses, the microwire bus has been a popular and high volume solution for a long time. It supports some of the lowest cost serial memories and is fast enough for many applications. However, there are a few drawbacks.

1. The microwire interface does not support as many density options as the other interfaces, with densities topping out at 16kbits. There also have been few new products introduced with this bus. This limits the designer's choices in new designs.

2. The microwire requires more port pins than the I^2C bus, so it's use can be more costly to the system designer.

3. The microwire bus is limited in speed and architecture. Microwire devices limit the interface speed to 1MHz and clock data out and in on the same edge. This imposes some limitations on the design. A similar bus (SPI) clocks data in and out on opposite edges of the clock and has higher speed specifications.

So while microwire has some advantages, it will likely be one of the first of today's serial busses to disappear.

The I^2C Bus

The I^2C bus, developed at about the same time as the microwire bus, will likely be around for some time. Since it was developed as a peripheral bus, it works very well in systems that have few ports available on the host controller and must connect to a number of peripheral devices. Some of the more common I^2C devices are A/D converters, LCD displays, Real Time Clocks and memories.

in 1985 Xicor introduced a 2-wire serial memory device that could operate on the I^2C bus. Xicor is still one of the industry leaders in 2-wire density, interface speed and features and has one of the widest selections of 2-wire memories with densities ranging from 128 bits to 128kbits.

The original I^2C bus specified 100kHz maximum speed. Based on industry feedback, this increased to 400kHz, and was recently increased by Philips to 3.4MHz. The I^2C bus speed was initially limited by it's peripheral roots. Potentially long I^2C bus lengths and the activity of a number of devices on the bus (including

multiple masters) increase the possibility of noise induced errors as the speed increases. This, coupled with the indeterminate loading and collision detection protocols, makes I^2C a noise sensitive interface.

The I^2C interface consists of two lines clock (SCL) and data (SDA). The protocol specifies that communication begins with a start bit and ends with a stop bit. Data going HIGH to LOW while the clock is HIGH defines a start bit (see Figure 6.) Data going LOW to HIGH while the clock is HIGH defines a stop bit. During transmission of data the SDA line cannot change while the clock is high. This protocol makes it difficult to discriminate between a start/stop bit and data on a higher speed, noisy bus.

Noise on the I^2C bus can sometimes result in corrupted data. In the I^2C protocol, a random read of the array (see Figure 7) consists of two parts. First, the host writes the address where the desired read will start. Then the host sends a repeated start bit followed by a current address read instruction. The host then clocks data in from the SDA line. Since the current address read instruction differs from a write instruction by a single bit, noise at a critical time turns a read operation into a write and clocks intended for a read operation become clocks writing data into the device. The result can be uncontrollable data corruption. This doesn't happen often, but when it does it can be serious. As I^2C speeds increase, there is much less margin for error, which leads to higher probability of failure.

To limit the probability of this type of error, Xicor introduced 2-wire devices that provided input noise filtering, schmidt triggers and input latching. To give engineers additional control over memory write operations, some devices include Block Lock features. This gives the system designer the ability to lock critical parts of the array, so unexpected and uncontrolled writes due to noise cannot damage critical system data.

In order to allow a number of peripherals to connect to the same set of two wires, the I^2C bus requires an open collector with pull-up on the SDA output of any device on the bus. For multi-master systems, the SCL line also needs an open collector and a pull-up. This configuration allows one device to "win" over another when two devices try to send data at the same time (collision). However, this configuration increases system power, since there is 270µA[1] current flowing through the pull-up resistor for the duration of a "0" bit .

In summary, the I^2C bus has the advantages of needing few port pins and supporting a number of peripheral devices. However, the I^2C bus has the disadvantages of lower speed, noise susceptibility and higher system current.

The SPI Bus

Motorola created the SPI port in the mid 1980's for use with their 68HC11 and 68HC05 product families. The SPI port shares similarities with the microwire port, using similar signal names and command protocols. SPI clocks data in and out differently from the microwire and can be clocked at a much higher rate.

Xicor was the first company to introduce an SPI serial memory (the X25C02 in 1991) and led the industry in developing higher density devices. Xicor also pioneered the use of Block Lock™ mechanisms on SPI

[1.] Assumes 100pF bus capacitance and 2.7V operation.

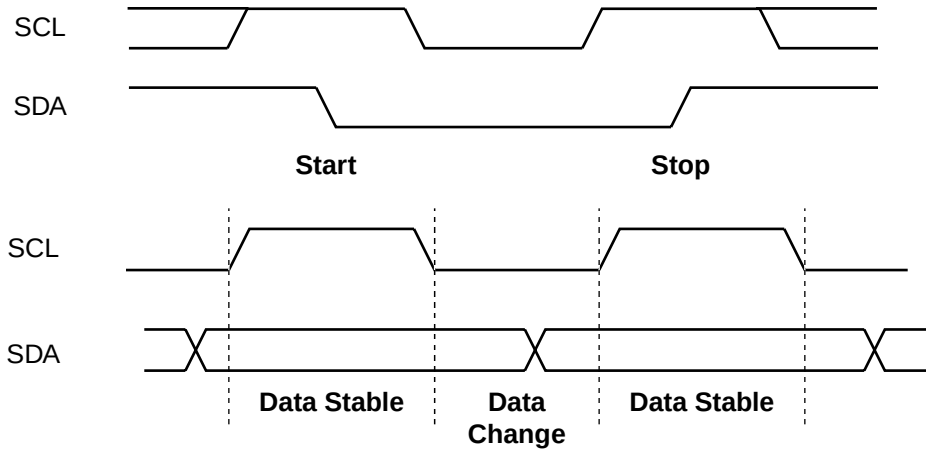

Figure 6. I^2C Start, Stop and Data bits

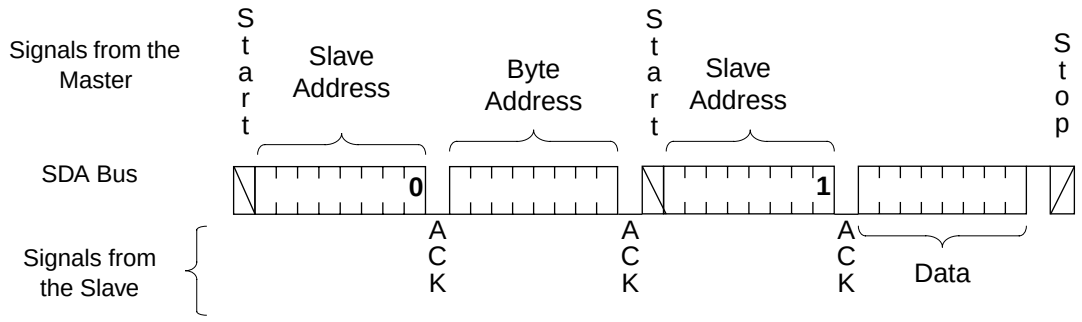

Figure 7. I^2C Random Read—This figure shows a random read of one byte.

devices to improve data integrity. In the last few years microcontrollers, DSPs, RISC processors and ASICs all feature built-in SPI ports. These developments prompted Dataquest in 1996 to report SPI as the fastest growing serial interface. Today, as processors increase in performance, the capabilities of the SPI port increase. To keep up with this trend, Xicor introduced SPI memories with a 5MHz clock rate and plans future speed increases.

The SPI interface consists of four control lines: $\overline{CS}$, SCK, SO, and SI. The host controls the $\overline{CS}$ line to select the serial memory, then uses the clock and data lines to transfer data back an forth. Current implementations of the SPI interface for memory devices are “half duplex”, meaning that data does not go out on the SO line while writing data on the SI line. So it is possible to connect SI and SO to get a three wire interface. Most microcontroller SPI implementations provide a “full duplex” mode. That means that data clocks into the device at the same time data clocks out. This can help increase the data throughput. Future SPI memories will likely take advantage of this full duplex capability.

The SPI interface is command driven (see Figure 8.) To read data from the memory, the host selects the device, sends a READ command, sends the address of the desired data, then clocks the data out of the device through the SO pin. At the end of the transaction, the host deselects the device. There are commands to READ and WRITE the array, read (RDSR) and write (WRSR) to a status register, and set (WREN) and clear (WRDI) a write enable latch.

Before any successful nonvolatile write to an SPI device, the host must select the device, send a WREN command, deselect the device, select the device again, send a WRITE command, send the 16 bits of address (older low density devices had 8 address bits), send a multiple of 8 bits and deselect the device. Any violation of this sequence terminates a write operation. Also, the completion of a nonvolatile write resets the write enable latch, forcing the entire sequence to be repeated for the next write operation. This makes SPI memories insensitive to noise. Block Locking further decreases the probability of inadvertent writes due both to noise and programming glitches.

SPI is a high speed memory port supported by a variety of microcontrollers, DSPs and ASICs. It is growing in popularity as host performance increases and as built in SPI ports become more common. New features as well as higher speeds indicate that the SPI interface will be around for quite a while.

The MPS bus

As fast and common as the SPI interface becomes and as versatile the I^2C has been, some applications will never be able use either type. As discussed, there are systems that simply have no ports to spare and there are other systems where the host has no ports that can handle serial interfaces. It is for these applications that Xicor developed the MPS interface. The MPS interface is not entirely new. In fact, most processors, microcontrollers, ASICs, DSPs and RISC chips already have the MPS interface built in. It is commonly called the memory bus! MPS simply takes one of the standard memory bus I/Os, two memory bus control lines and a chip select and uses a special protocol to create a high speed serial bus. This bus needs no special or general purpose port pins and gives the designer all the benefits of serial memories with none of the drawbacks.

The diagram of Figure shows the connections of the MPS interface. In operation, the host loads the accumulator with the desired data address and sends the address to the MPS device through a series of write

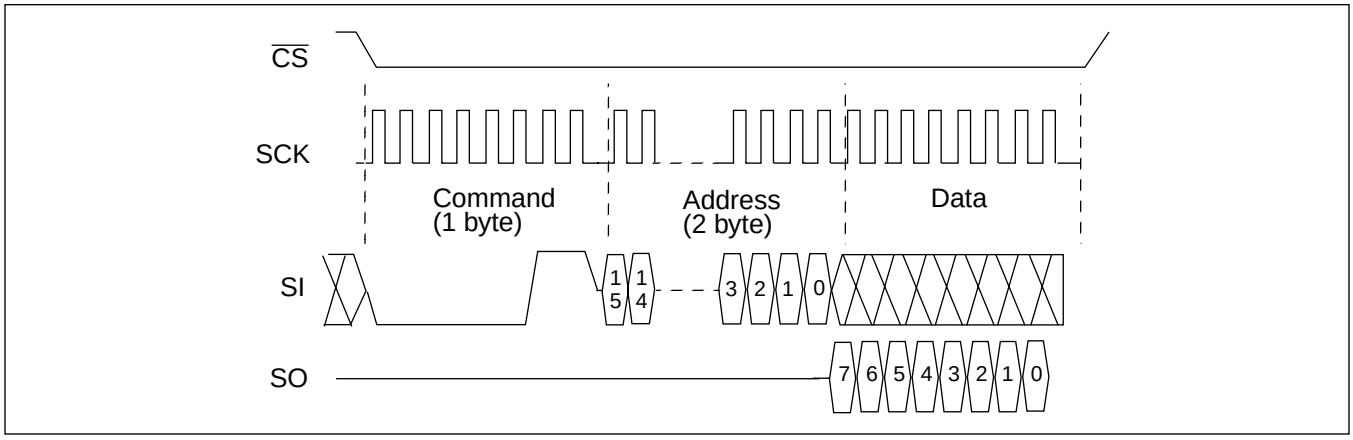

Figure 8. SPI Read Command

and rotate accumulator operations. The host then reads the data with a series of read and rotate accumulator operations. Since all CPUs contain read, write and rotate instructions, the MPS protocol is simply implemented in software. Also, since no host address lines connect to the MPS, it works equally well with 8, 16, 32 or 64 bit systems and with both multiplexed non-multiplexed busses.

Like all serial devices, the MPS has an interface protocol to manage the transfer of data. The read protocol consists of a “software reset” sequence, followed by 16 bits of address, followed by the reading of an unlimited number of data bits. The write protocol consists of the “software reset” sequence, followed by 16 bits of address, followed by a multiple write of 8 data bits, followed by a “start nonvolatile write” sequence. In a write operation, the host must meet all conditions exactly or the nonvolatile write will not happen. This protocol makes it very unlikely that noise will result in an inadvertent write. Future MPS devices will also provide Block Lock features to give the designer the ability to differentiate between nonvolatile system and user memory.

The clock frequency is the normal measure of speed for I^2C and SPI busses. The MPS uses a similar terminology with reference to the “10MHz” speed, however the terminology is not exactly correct. Since the MPS devices connect to the parallel memory bus, different timing parameters become important.

The most important factor in looking at the MPS interface is read access time. The host expects valid data on the bus within a certain fixed time after the read signal goes active. This assures a proper data setup time before the host latches in the data. The new MPS devices have a 25ns read access speed. This is fast enough for most processors to read data at full speed without adding bus wait states.

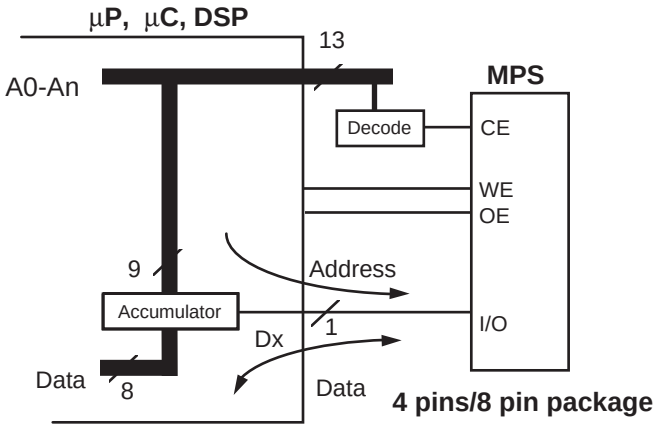

Figure 9. The MPS Interface makes use of one data line and two control lines on a standard parallel memory bus.

The second important timing parameter is the data transfer rate. The memory device must provide data fast enough to support successive read instructions without adding software wait states. In this case the host instruction cycle time becomes important. The new MPS parts support processors with 100nS instruction cycle times, providing a 10Mbps maximum data rate. This rate meets or exceeds that of most host processors.

Since the MPS works with different bus widths and with either multiplexed or non-multiplexed memory busses, a large number of host processors and application specific controllers can use the MPS memory. A summary of these processors appears in Table 1.

In summary, the MPS is a serial memory device with a fast interface that makes use of the existing parallel memory bus. MPS requires no extra hardware, no ASIC silicon design or design verification, no special serial ports, and no general purpose ports. MPS devices provide all the advantages of serial memories, (like low power, small size and low cost) without the need for special hardware.

Conclusion

Industry-wide, there is a trend to serial interfaces and away from parallel, as demonstrated by several new industry standards. The need for low power, small size and lower cost fuels this conversion and there is no sign that the trend will change in the near future.

In order to provide a communication medium to the serial memory, there are several types of interfaces. The discussion here focused on 2-wire, SPI and MPS solutions. The value of each lies in the design trade-off between how they connect to the host and how fast each transfers data.

In recent years, consumers have demanded more portability, performance and customization from the products they buy. These needs fueled decreases in package size and power and increases in memory density and speed. With state of the art speed and packaging, Xicor 2-wire, SPI and MPS serial memories provide designers and consumers with the right solutions for today and for the future.

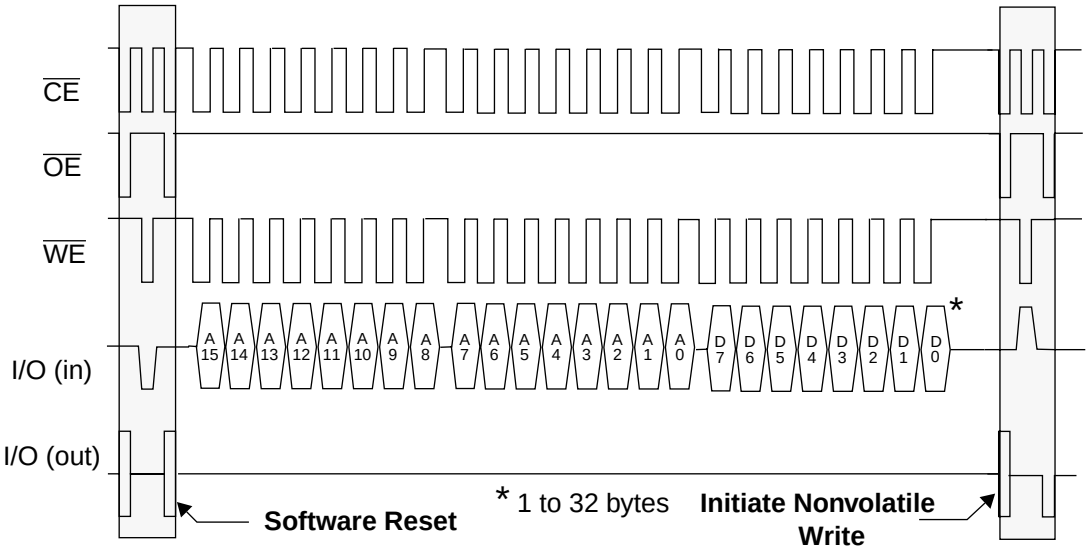

Figure 10. MPS Interface Protocol—Write to memory

Table 1: Controllers with an MPS Compatible Interface

Microcontrollers		Microprocessors	DSP	RISC	Application Specific Controllers
i8031/51	M68HC05	M68000	TMS320C2xx	SH70xx	MSM Cellular Chipset (Qualcomm)
i80151/251	M68HC08	M68300	TMS320C5x	SH77xx	
P51XA	M68HC11	i8086/186	TMS320C3x	µPD30101 (Vr4101)	GEM 300 GSM Chipset (GEC Plessey)
DS89Cxxx	M68HC12	i80188	TMS320C4x		
AT89Sxxx	M68HC16	i8096/196	M56000	µPD70xxxx (V8x)	DCAM-101 Digital Camera Chip (LSI/Minolta)
H8-300	M68330	µPD70xxx (V20/V25/V30/35)	M96000		
H8-300H	MPC82x		ADSP21xx	PR31500 (Philips)	PC Chip Sets (ISA Bus)
H8S2000	MPC860	Z80	ADSP21msp58/ 59		
µPD780xx	PIC16C662	Z180	D950-Core (SGS-Thomp- son)	IDT79R3500 (IDT-MIPS)	ARM7100 Single Chip Organizer
µPD782xx	PIC17C4x	Z380	DSP16xx (Lucent)	ARM Core	ARM7500 Single Chip Netsurfer
µPD783xx	Z8	MCF5202 (Motorola)			NAV-2100 GPS Chipset
µPD784xx					