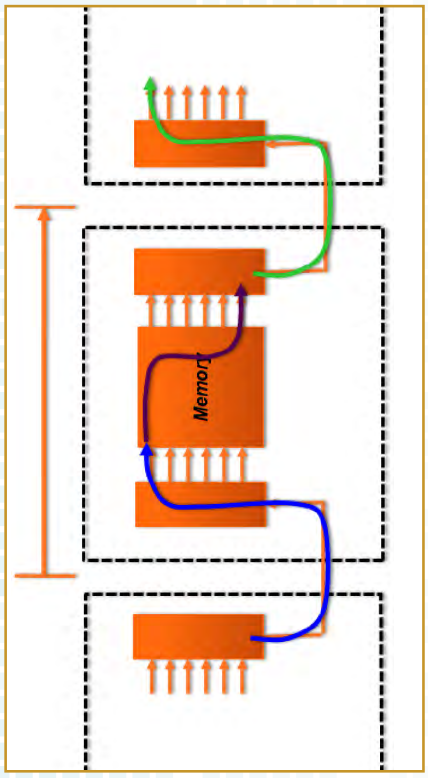

GOING SERIAL:

Breaking the Billion Access Per Second Bandwidth Barrier

by Michael Miller
VP, Technology Innovation & Systems Applications
MoSys, Inc.

Breaking the Billion Access Per Second Bandwidth Barrier

INTRODUCTION:

Since the earliest days of electronic communication, I/O has been and still is the bottleneck. Unlike chip speeds, the "Moore's Law" of packaging technology scales upward at a much slower pace, approximately once every 8 to 10 years, which limits the growth in physical connections. The next major gains in connection bandwidth must therefore come from advances in electrical and protocol layers. For today's designer, some of the key I/O questions are: What are the current trends in Serial vs. Parallel interconnect design strategy? How can you optimize bandwidth in each interconnection and transaction? How much bandwidth can you achieve by going serial? Can you achieve efficiency with a serial interface, particularly in small data units of 4 and 8 Bytes? Can a serial interface break the billion access per second bandwidth barrier?

SERIAL PROGRESSES FROM MACRO TO MICRO

Serial communication originally appeared in the earliest days of electronic communication on a macro level such as the telegraph (Morse Code), undersea cables, and microwave. As electronic designers became more sophisticated and demands for speed and ease of connectivity increased, serial interfaces emerged such as Ethernet connected boxes on a network. In the next step towards the micro world, new standards facilitated widespread adoption of serial communications on backplanes. These standards included PCI Express, XAUI, FDDI (for disk drives), and more. The subsequent implementations resulted in cost-effective, low pin count designs.

Designers today are increasingly selecting serial to communicate between chips on the board. With each new generation, the thought process and technical momentum has tended toward reusing the protocol concepts of the past because they were reliable and worked. Of course when applied to board level point-to-point applications, the standards' protocols for backplane communication resulted in unnecessary overhead for communicating small messages between devices.

THE MICRO WORLD HITS THE PARALLEL ROAD BLOCK

Speed in the world of computation and network processors has doubled approximately every two years. To double the bandwidth, system architects need to either double the pins or double the clock rate. However, when we view

pins per package from the perspective of Moore's Law, the pin counts have doubled at a much slower pace, approximately every eight to ten years.

Meanwhile, parallel clock rates have been slowly climbing from 250 MHz in 2001 to 500MHz in 2007 to over 1GHz in 2010. These higher clock rates require designers to maintain tight control and balance trace lengths, vias, and spacing. When operating at such high bandwidth, limitations due to signal integrity diminish the number of parallel lines when using a synchronous clock rate.

As a result, parallel interfaces progressed from asynchronous to synchronous and then progressed to Double Data Rate (DDR) and Quad Data Rate (QDR). For instance, a DDR interface operating at 500 MHz can transmit one Gigabit per second (Gbps) per pin on the rising and falling edge of the I/O clock. QDR not only uses DDR techniques for the separate I/O but also DDR for the address and requires an alternating read/write address. Each of these interfaces use 2 to 4 I/O clocks for each memory core clock, thus multiplexing the I/O bus up to 8 times for each transfer. At the extreme, GDDR5 promises a 5 Gbps single ended transmission rate with very tight routing constraints across the whole interface. The challenge: Meet the tight timing and layout constraints at this speed. To even achieve these rates, the GDDR5 is starting to specify eye diagrams like SerDes I/O. In contrast, current serial data rates transmit up to 10 Gbps per pin pair (5 Gbps per pin) with road maps to higher rates. Serial standards like XUAI have shown how to deskew and bound multiple lanes to eliminate the tight layout and timing constraints between channels across the chip interface. System architects and designers need an alternative to achieve doubling of the data rate.

Going serial with a Serializer/Deserializer (SerDes) device provides an answer already present in today's FPGA, ASIC and NPU. The new wave of recently introduced devices sport 11+ Gbps interfaces. SerDes converts serial transmission through I/O into parallel for operations within the device. For each system clock, data transfer occurs eight or ten times faster than the system clock. Of course, the SerDes reconverts parallel output from the device back to serial for output. In effect, implementing DDR and QDR transmissions suggests that parallel is converging on serial as well. So let's call it what it is: We now have serial interconnects.

THE MICRO WORLD GOES SERIAL

Several major issues support the transition from parallel to serial in the micro world: differential design, electrical standards, clocking, serial protocols, scalability, and layout. Low voltage differential design uses two lines whose

swings are less than rail-to-rail, thus reducing power (CV^2f). Single-ended I/O requires more power to achieve noise immunity because it compares the incoming signal to the V_{ref}, which has noise in it. The designer must thus establish a guard band around the V_{ref}. In short, the transmitted voltage must be above the highest voltage of the guard band to register as a "1" and likewise, below the lowest voltage of the guard band to register as a "0". The input voltage thus swings between the guard bands defined by the V_{ref}. This process expends power to guarantee noise immunity.

In a low voltage differential design, the system transmits the complementary voltages, high or low. The receiver only senses the polarity of those two voltages and eliminates the need for V_{ref}. The two voltages have sufficient separation in value to determine which is a "1" and which is a "0". As a result, a differential design uses less power to guarantee reliable data transmission because the voltage swing is smaller.

So for the same amount of power, differential designs transmit much more data. Because the system uses paired traces, the design exhibits better transmission line characteristics as well. In fact, data transmits faster with a differential design using two pins than with two individual single-ended lines. Differential designs result in highly efficient transmission as exhibited by better bandwidth-to-power ratio per pin, over two times better noise immunity, and over twice the data transmission rate for the same power with a superior noise environment.

Below in Figure 1, the board is laid out using parallel interconnects. On the right, by comparison, the same design was executed using serial interconnects. One immediately observes the reduced real estate required by the serial layout on the right. The cause: Using serial communication increases bandwidth density, resulting in a reduced number of traces, since serial only requires a matched differential pair.

Serial protocols like XAUI, InterLaken and PCIe include deskewing mechanisms to handle any mismatch between data lines.

SerDes electrical standards groups support this development with the Optical Interface Forum (OIF), which provides for common electrical interface for 3, 6, 11 and 28 Gigabit onboard and backplane interconnects. Other relevant standards include IEEE 1, 10, 40, and 100 Gigabit Ethernet and FDDI (for disk drives), SATA, and USB.

Standards development requires 3-5 years to develop channel models, set clocking and jitter budgets, determine electrical signal coding, and encourage ecosystem existence. In addition, standards development is a costly

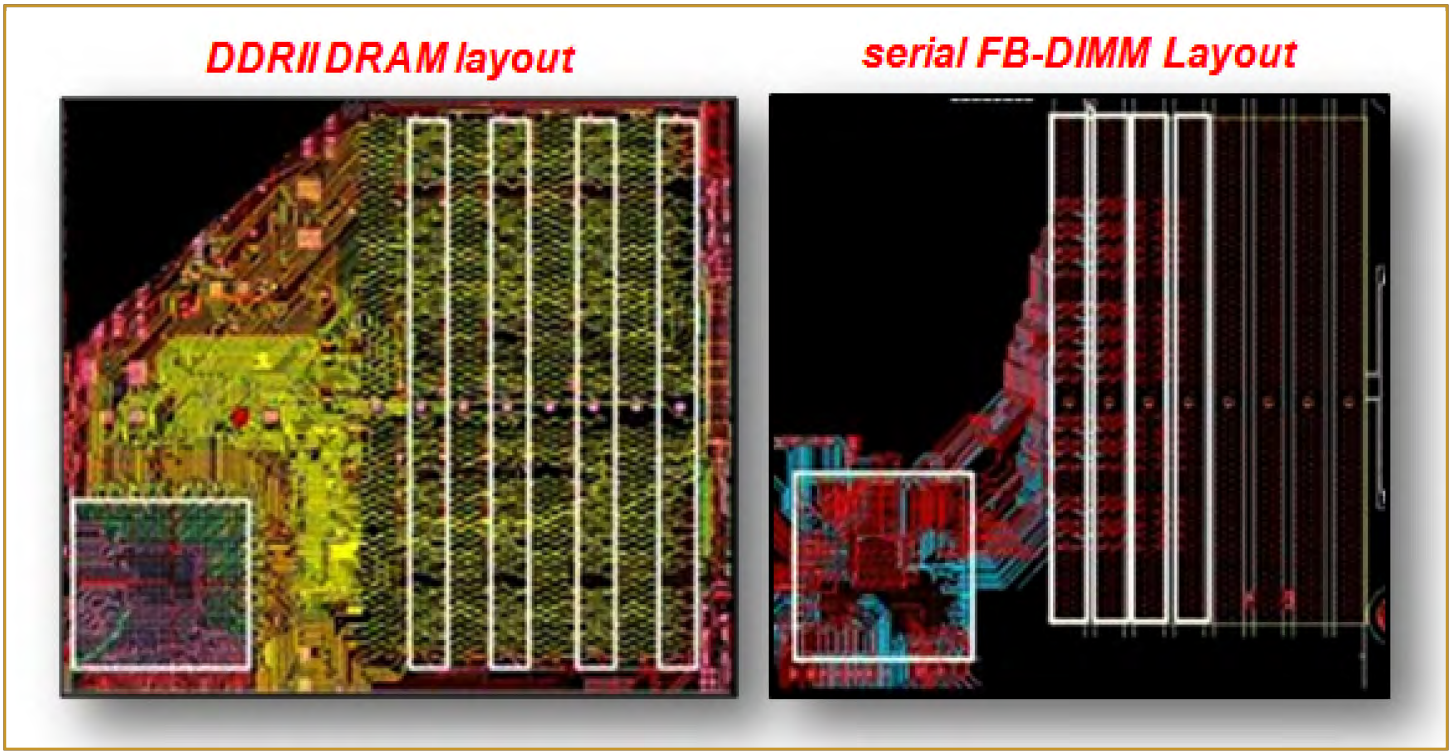

Figure 1: Same Design – Parallel on left , Serial on right

undertaking. The conclusion for designers: It's far more efficient to piggyback designs on existing serial standards because they provide a design roadmap for the future. Today that roadmap extends out to 28 Gbps for a serial pair or 14 Gbps/pin.

CLOCKING IN THE SERIAL WORLD

Historically, designers have used at least three methods of clocking: source synchronous, plesiosynchronous, and mesosynchronous. With source synchronous, the clock is transmitted by the source to the receiver.

Corresponding signals operate plesiosynchronously if they function at the same nominal frequency provided by separate sources like two crystals, but may have a slight frequency mismatch, which leads to a drifting phase. The source (transmitter) and destination (receiver) each have independent clocks, nominally the same frequency. A Phase Lock Loop (PLL) analyzes the transitions in the data. From that analysis, the receiver derives the clock that must have been used for transmitting the data. Sometimes called Clock Data Recovery (CDR), the comparator tunes the PLL to match the frequency from the local clock. The downside to plesiosynchronous clocking is that this method requires ongoing clock adjustment at the destination and the associated power.

The PCI Express standard utilizes a mesosynchronous clocking methodology, a return to the original idea of the source synchronous method – eliminating the requirement for constant frequency adjustment. Both source and destination have the same clock source and must have a connection wire between them. Of course, this approach is not very practical for macro applications like an undersea cable, but it is practical at the board and device level.

SERIAL LATENCY

Figure 2 demonstrates the phenomenon of serial latency. Serial latency is measured from the first bit in to the first bit out. System designers can approximate serial latency by applying the following formula: Serial latency = # of clocks to deserialize + deskew + memory latency + # of clocks on the first bit out.

Comparing a 1 GHz DDR parallel to 10 Gbps serial transmission rate moving 80 bits of information, the parallel bus uses 40 data lines, a 2:1 mux register, and clocks one half of the 80 bits on the rising and falling edges of the I/O clock in 1 ns. In contrast, serial transmission uses 8 pairs, serialization/deserialization registers operating at 10 GHz and completes the same task in 1ns as well. The data rate for serial transmission is thus approaching the vanishing point of break-even compared with parallel buses. The only real difference is in the deserialization time. In short, a serial bus running at 10 Gbps runs as fast as a parallel bus.

In some applications, latency is not as important. For example, networking equipment transmits packets through a line card and across the backplane. In this case, high throughput is more valuable than overall latency.

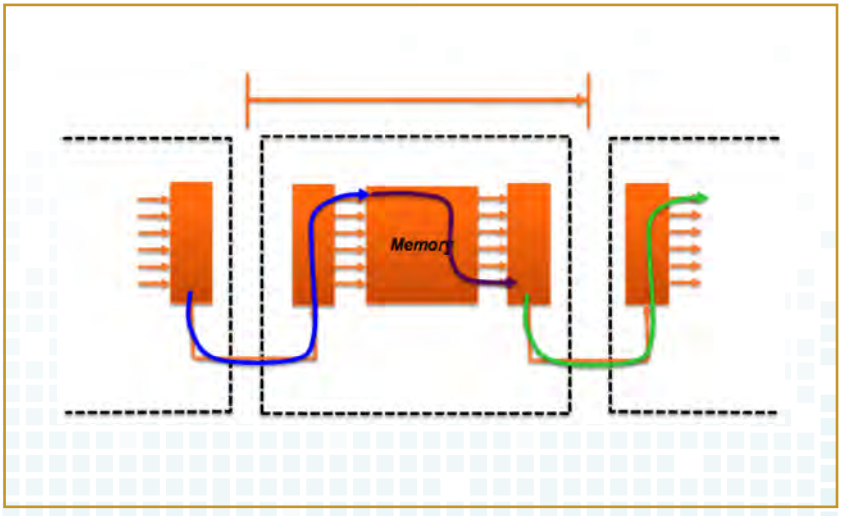

Figure 2: Serial Latency

On the other hand, when the system needs to repeatedly access a memory block to process a packet, the latency of the memory is multiplied several fold. This determines how many packets must be stored and processed in parallel thus increasing the complexity of the storage architecture. In other cases, some tasks MUST be completed before the next packet in the same flow arrives. The latency of the memory and the interface again come into play. So latency can in fact be important.

SERIAL PROTOCOL AND INTERFACE EFFICIENCY: EXAMPLE OF 72B TRANSMISSION

As data rates increase and electrical signals shrink, the probability of an incorrectly sampled data bit increases, regardless of whether the interface is serial or parallel. At one GHz and beyond for telecommunications interfaces, even parallel interfaces need to consider some sort of error checking and handling solution. Because of their higher data rates and farther reach, serial interfaces have always required a higher-level protocol. As such, serial designs have always included mechanisms to insure data integrity. For example, today's assumed error rate is 10^{-15}. In PC board applications, SerDes devices may achieve a much better rate (10^{-18}), but it's hard to measure.

Many serial protocols are available to designers today including PCIe, SRIO (Serial Rapid IO), Interlaken, and SFI-S. These protocols reliably transport variable length packets of information between multiple endpoints across congested switches with priority control. Point-to-point protocols like Interlaken handle multiple flows with variable length packets and per flow rate control, which requires multiple fields. Each of these fields adds to the minimum

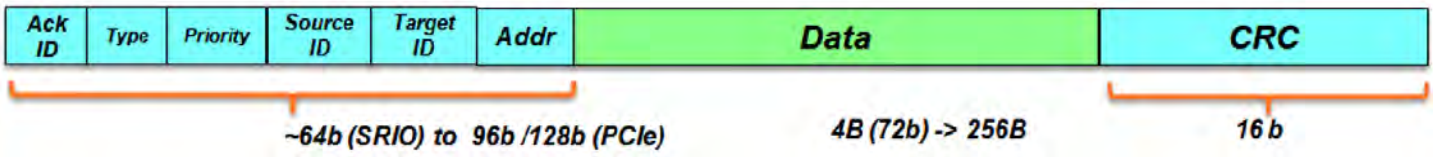

Figure 3: Serial Protocol including Overhead

packet overhead, which lowers efficiency when transmitting small packets. When transporting 100 byte messages, the overhead for the data protocol consists of only 10% of the entire message, meaning the transmission efficiency is 90%.

For 72-bit data word transmissions, as shown in Figure 3 above, the protocol overhead dominates and results in transmission efficiency of approximately 37%. For a 72-bit transmission, the protocol overhead for SRIO totals 64 bits, for PCIe totals 96b/128b, and the CRC totals 16b. Total overhead for SRIO totals 80 bits and for PCIe, up to 144b.

The question for designers: How to reliably transport data without the overhead, since these protocols address issues not relevant in point-to-point control? When transmitting a large packet of information, such as 64 bytes or more, a few bytes of overhead don't matter. However, when transmitting a small packet of information using the same protocol, the number of bytes devoted to the protocol makes a significant difference.

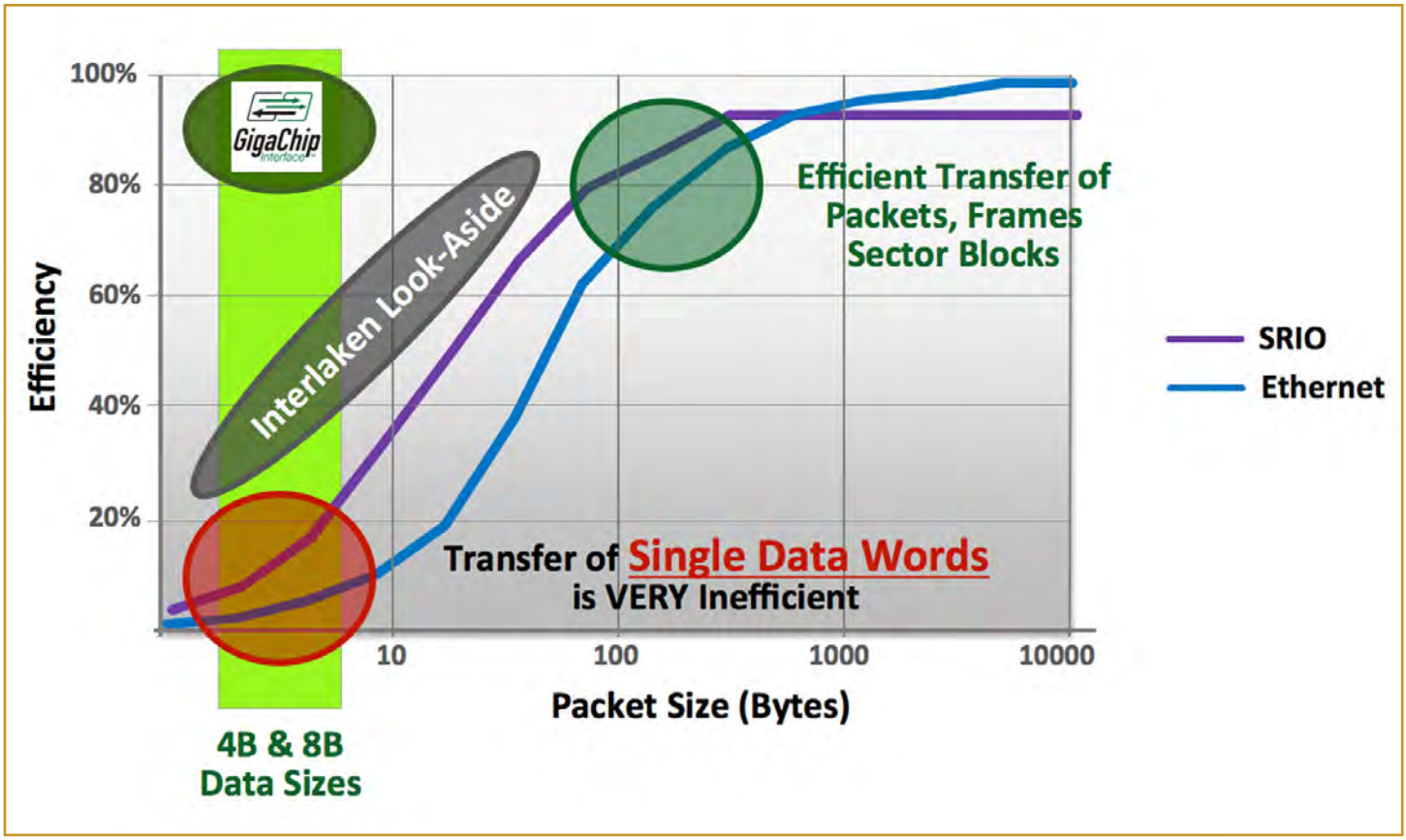

Figure 4

Figure 4 demonstrates that if the transmission can eliminate most of the protocol overhead, the result is high efficiency. For Packet Data Units (PDUs) between four bytes and eight bytes such as transmission of single data words between an SOC device and memory, the transmission is very inefficient.

Using the GigaChip Interface (developed by MoSys) to transmit data in the range of four to eight bytes addresses only one thing: since bit errors occur infrequently (10^{-15}), the transmission packets can eliminate most of the protocol overhead. As shown in Figure 5, this results in approximately a 90% efficiency to transmit 72 bits of data plus 8 bits for CRC and Ack.

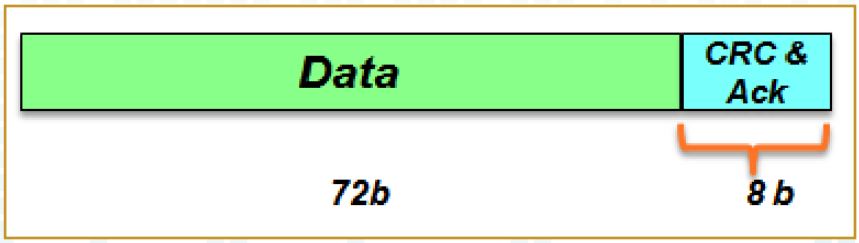

Figure 5

In Figure 6, note the effect of the protocol becoming independent of the number of lanes or connections — serial designs become scalable. If designers select protocols carefully, and make them independent of the physical number of lanes, the industry can implement solutions in 1, 2, 4, 8, 16 lanes or more. The designer can either scale the number of lanes and/or scale the data rate on each one of those serial lanes. The reason scalability matters is that bandwidth has its own cost in terms of board materials, packaging materials, board real estate, etc.

		3G		6G		10G		16G	
		Gops	Gbps	Gops	Gbps	Gops	Gbps	Gops	Gbps
Channels	2	0.075	5.4	0.15	10.8	0.25	18	0.4	28.8
	4	0.15	10.8	0.3	21.6	0.5	36	0.8	57.6
	8	0.3	21.6	0.6	43.2	1	72	1.6	115.2
	16	0.6	43.2	1.2	86.4	2	144	3.2	230.4

Figure 6: Scalability Through Lane Count & Data Rate

Using high overhead protocols for 4-8 Byte PDUs results in inefficiency, which means that the design sacrifices bandwidth. Eliminating the protocol overhead of each transmission solves the inefficiency problem and results in very high-speed transmission of raw data. For example, at a data rate per channel of 16 Gbps using 16 channels, a serial connection will transmit 3.2 Giga operations per second (Gops) and 230.4 Gbps.

SIGNAL INTEGRITY

Nothing comes for free. As signals enter the GHz range, designers must be aware of the signal integrity aspects of high-speed channels. This will be true whether the bus is parallel or serial. The trade-off being, of course, that parallel buses run slower, but have many more channels to model and control. Serial runs at a much higher rate proportionally, but the number of channels is proportionally less. Because of the push on serial channels for moving packets across backplanes, tools have been developed to solve this problem. Keep in mind that on the board, the channels will be much shorter and less complicated, thus making the task to design a high speed chip-to-chip link much more manageable.

CONCLUSION

In its simplest statement, the design choice between serial and parallel used to boil down to number of interconnects vs. speed. Because parallel requires a high number of interconnects, it increases device real estate, always a downside. Given the continued progression in transmission solutions, parallel buses are reaching the

end of their natural life. Based on Moore's Law as applied to packaging, to achieve double the bandwidth using parallel, the device must double the pins, which is no longer very likely. And for the same reason, parallel buses are not likely to double.

In response to the demise of parallel interconnects, MoSys will shortly be introducing a new generation of solutions that use serial interfaces running at 10 Gbps and beyond. They transmit at 90% efficiency. The introduction includes a new serial protocol and new memory products that will break the 1 billion access per second bandwidth barrier.

MoSys, Inc.
755 N. Mathilda Ave.
Sunnyvale, CA 94085
(408) 731-1800

www.mosys.com Picture Sources: Figure 1 –
http://www.theinquirer.net/inquirer/news/1037859/there-magic-intel-fb-dimm-buffer;
http://www.simmtester.com/page/news/showpubnews.asp?num=113;
http://www.icc-usa.com/Fully-Buffered-DIMM-0305.pdf

Written with the assistance of Lee Stein, Stein & Associates, Inc.