Look Ma, No Motherboard!

How one design team put a full single-board computer with SATA into a Xilinx FPGA.

by Lorenz Kolb
Member of the Technical Staff
Missing Link Electronics, Inc.
lorenz@missinglinkelectronics.com

Endric Schubert
Co-founder
Missing Link Electronics, Inc.
endric@missinglinkelectronics.com

Rudolf Usselmann
Founder
ASICS World Service Ltd.
rudi@asics.ws

Embedded systems for industrial, scientific and medical (ISM) applications must support a plethora of interfaces. That's why many design teams choose FPGA-based daughtercards that plug right into a PC's motherboard to add those special I/Os. Given the huge capacity of modern FPGAs, entire systems-on-chips can be implemented inside a Xilinx® device. These systems include hardware, operating system and software, and can provide almost the complete functionality of a PC, diminishing the need for a PC motherboard. The result is a more compact, less power-hungry, configurable single-board computer system.

Many of those ISM applications rely on fast, dependable mass storage to hold and store the results of data acquisition, for example. Solid-state drives have become the de facto standard in this application because of their high reliability and fast performance. These SSDs almost always connect via a Serial ATA (SATA) interface.

Let's examine the steps that we took to extend a single-board computer system, built around a Xilinx chip, with high-speed SATA connectivity to add SSD RAID functionality. For this task, the Xilinx Alliance Program ecosystem brought together ASICS World Service's (ASICS ws) expertise in high-quality IP cores and Missing Link Electronics' (MLE) expertise in programmable-systems design.

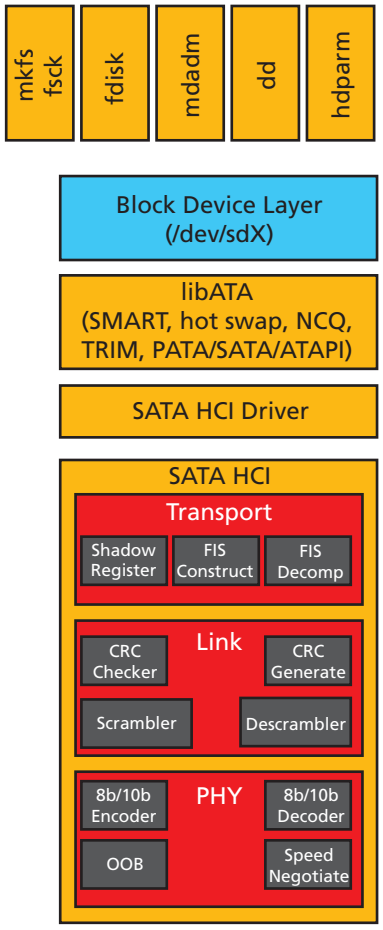

Figure 1 – Serial ATA function layers

But before delving into the details of the project, it's useful to take a deeper look at SATA itself. As shown in Figure 1, multiple layers are involved for full SATA host controller functionality. Therefore, when it comes to implementing a complete SATA solution for an FPGA-based programmable system, designers need much more than just a high-quality intellectual-property (IP) core. Some aspects of the design often get overlooked.

First, it makes sense to implement only the Physical (PHY), Link and some portions of the Transport Layer in FPGA hardware; that's why IP vendors provide these layers in the IP they sell. The SATA Host IP core from ASICS World Service utilizes the so-called MultiGigabit Transceivers, or MGT, [1] to implement the PHY layer—which comprises an out-of-band signaling block similar to the one described in Xilinx application note 870 [2]—completely within the FPGA. The higher levels of the Transport Layer, along with the Applications, Device and User Program layers, are better implemented in software and, thus, typically IP vendors do not provide these layers to customers. This, however, places the burden of creating the layers on the system design team and can add unanticipated cost to the design project.

The reason vendors do not include these layers in their IP is because each architecture is different and each will be used in a different manner. Therefore, to deliver a complete solution that ties together the IP core with the user programs, you must implement, test and integrate components such as scatter-gather DMA (SGDMA) engines, which consist of hardware and software.

In addition, communication at the Transport Layer is done via so-called

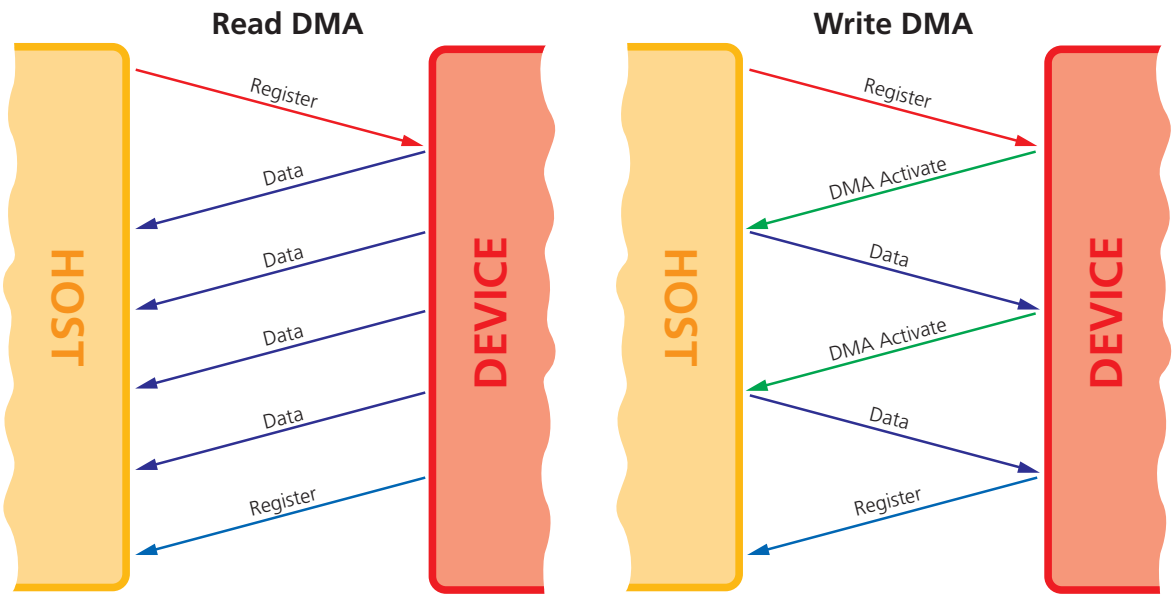

Figure 2 – FIS flow between host and device during a DMA operation

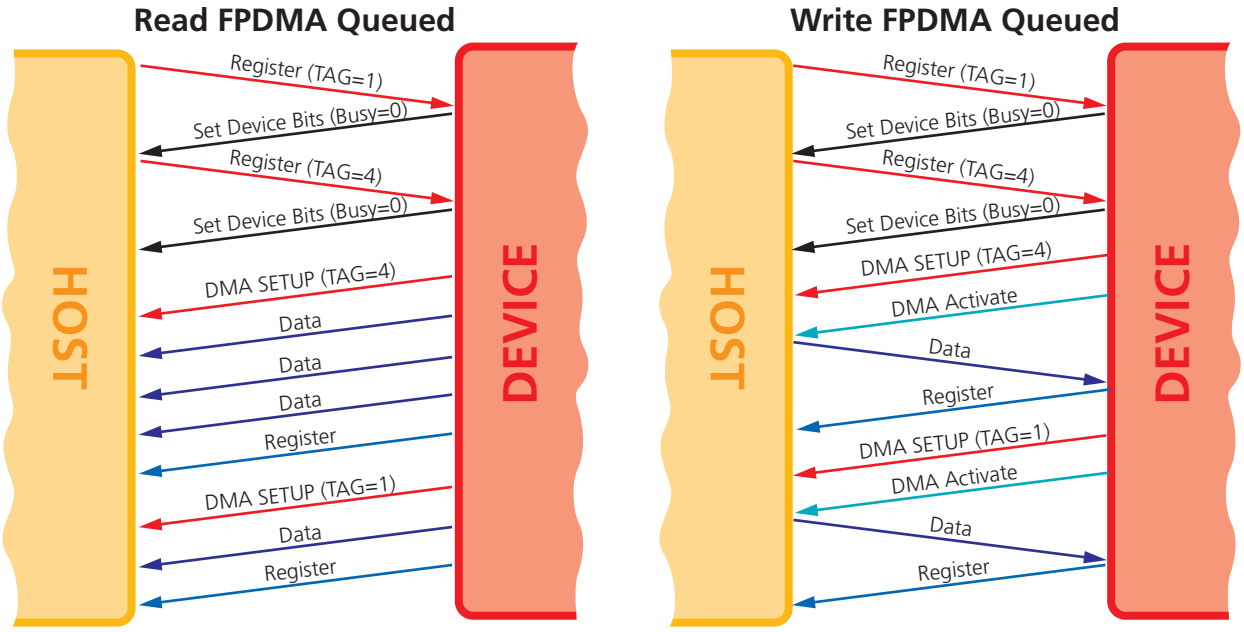

Figure 3 – FIS flow between host and device during first-party DMA queued operation

frame information structures (FIS). The SATA standard [3] defines the set of FIS types and it is instructive to look at the detailed FIS flow between host and device for read and write operations.

As illustrated in Figure 2, a host informs the device about a new operation via a Register FIS, which holds a standard ATA command. In case of a read DMA operation, the device sends one (or more) Data FIS as soon as it is ready. The device completes the transaction via a Register FIS, from device to host. This FIS can inform of either a successful or a failed operation.

Figure 2 also shows the FIS flow between host and device for a write DMA operation. Again, the host informs the device of the operation via a Register FIS. When the device is ready to receive data, it sends a DMA Activate FIS and the host will start transmitting a single Data FIS. When the device has processed this FIS and

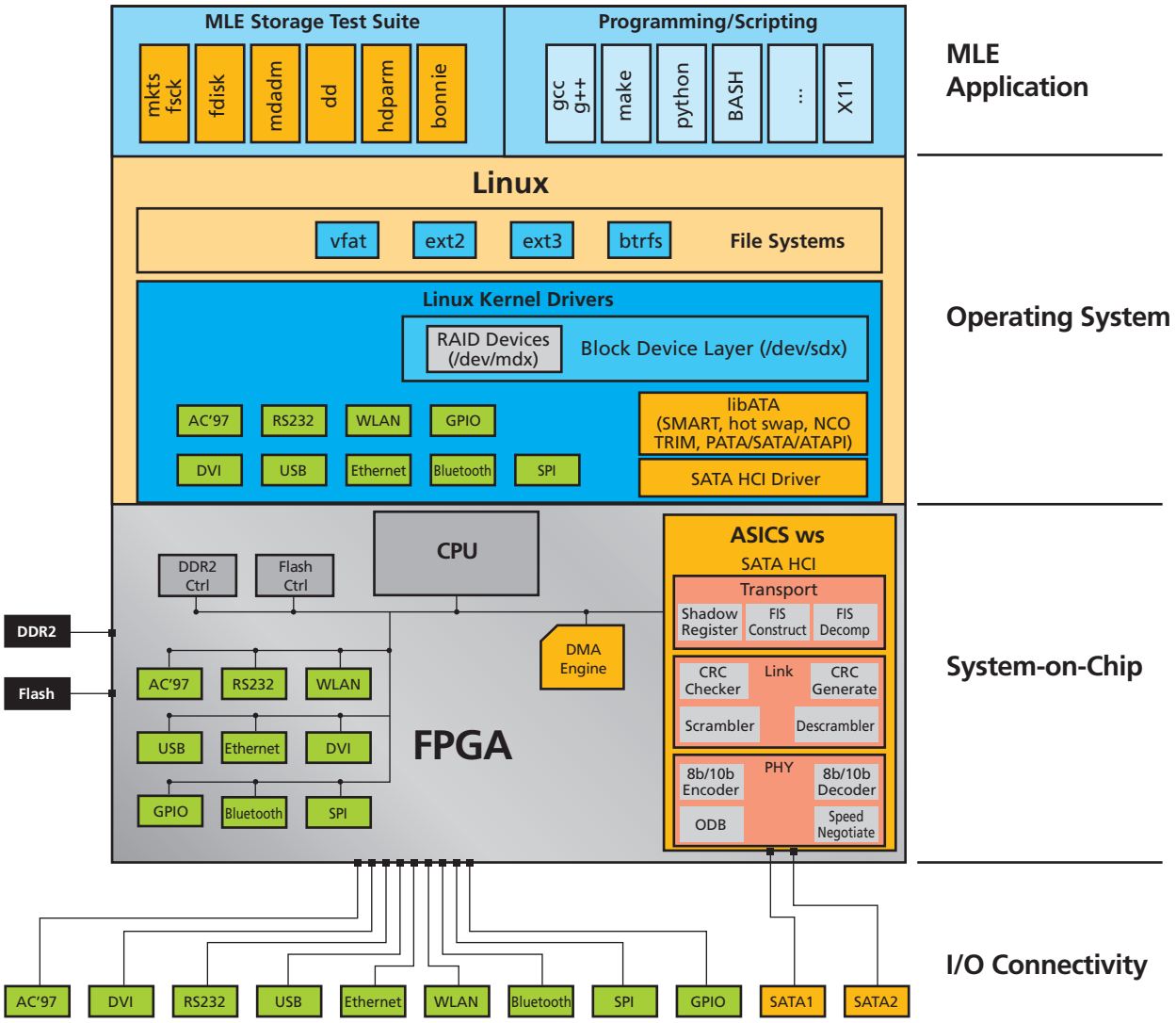

Figure 4 – Complete SATA solution

it still expects data, it again sends a DMA Activate FIS. The process is completed in the same way as the read DMA operation.

A new feature introduced with SATA and not found in parallel ATA is the so-called first-party DMA. This feature transfers some control over the DMA engine to the device. In this way the device can cache a list of commands and reorder them for optimized performance, a technique called native command queuing. New ATA commands are used for first-party DMA transfers. Because the device does not necessarily complete these commands instantaneously, but rather queues them, the FIS flow is a bit different for this mode of operation. The flow for a read first-party DMA queued command is shown on the left side of Figure 3.

Communication on the Application Layer, meanwhile, uses ATA commands. [4] While you can certainly implement a limited number of these commands as a finite state machine in FPGA hardware, a software implementation is much more efficient and flexible. Here, the open-source Linux kernel provides a known-good implementation that almost exactly follows the ATA standard and is proven in more than a billion devices shipped.

The Linux ATA library, libATA, copes with more than 100 different ATA commands to communicate with ATA devices. These commands include data transfers but also provide functionality for SMART (Self-Monitoring Analysis and Reporting Technology) and for security features such as secure erase and device locking.

The ability to utilize this code base, however, requires the extra work of implementing hardware-dependent software in the form of Linux device drivers as so-called Linux Kernel Modules. As Figure 4 shows, the Missing Link Electronics "Soft" Hardware Platform comes with a full

When integrating a SATA IP core into an FPGA-based system, there are many degrees of freedom. So, pushing the limits of the whole system requires knowledge of not just software or hardware, but both. Integration must proceed in tandem for software and hardware.

GNU/Linux software stack preinstalled, along with a tested and optimized device driver for the SATA host IP core from ASICS World Service.

When integrating a SATA IP core into an FPGA-based system, there are many degrees of freedom. So, pushing the limits of the whole system requires knowledge of not just software or hardware, but both. Integration must proceed in tandem for software and hardware.

Figure 5 shows examples of how to accomplish system integration of a SATA IP core. The most obvious way is to add the IP core as a slave to the bus (A) and let the CPU do the transfers between memory and the IP. To be sure, data will pass twice over the system bus, but if high data rates are not required, this easy-to-implement approach may be sufficient. In this case, however, you can use the CPU only for a small application layer, since most of the time it will be busy copying data.

The moment the CPU has to run a full operating system, the impact on performance will be really dramatic. In this case, you will have to consider reducing the CPU load by adding a dedicated copy engine, the Xilinx Central DMA (option B in the figure). This way, you are still transferring data twice over the bus, but the CPU does not spend all of its time copying data.

Still, the performance of a system with a full operating system is far away from a standalone application, and both are far from the theoretical performance limits. The third architecture option (C in the figure) changes this picture by reducing the load of the system bus and using simple dedicated copy engines via Xilinx's streaming NPI port and Multiport Memory Controller (MPMC). This boosts the performance of the standalone application up to the theoretical limit. However, the Linux performance of such a system is still limited.

From the standalone application, we know that the bottleneck is not within the interconnection. This time the bottleneck is the memory management in Linux. Linux handles memory in blocks of a page size. This page size is 4,096 bytes for typical systems. With a simple DMA engine and free memory scattered all over the RAM in 4,096-byte blocks, you may move only 4,096 bytes with each transfer. The final architectural option (D in the figure) tackles this problem.

For example, the PowerPC® PPC440 core included in the Virtex®-5 FXT FPGA has dedicated engines that are capable of SGDMA. This way, the DMA engine gets passed a pointer to a list of memory entries and scatters/gathers data to and from this list. This results in

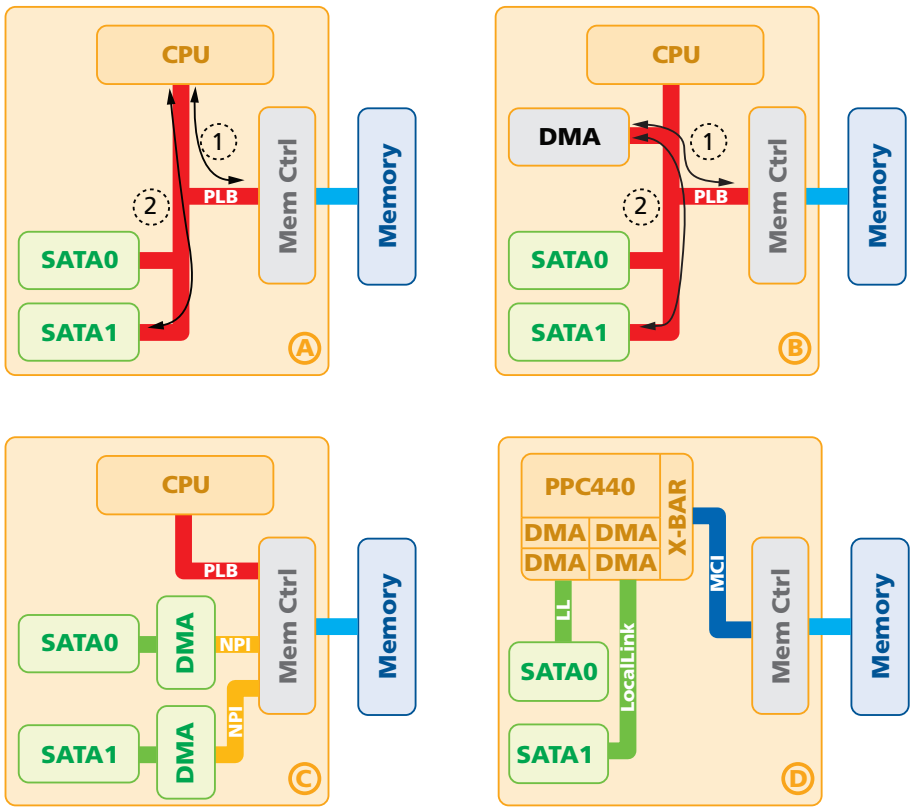

Figure 5 – Four architectural choices for integrating a SATA IP core

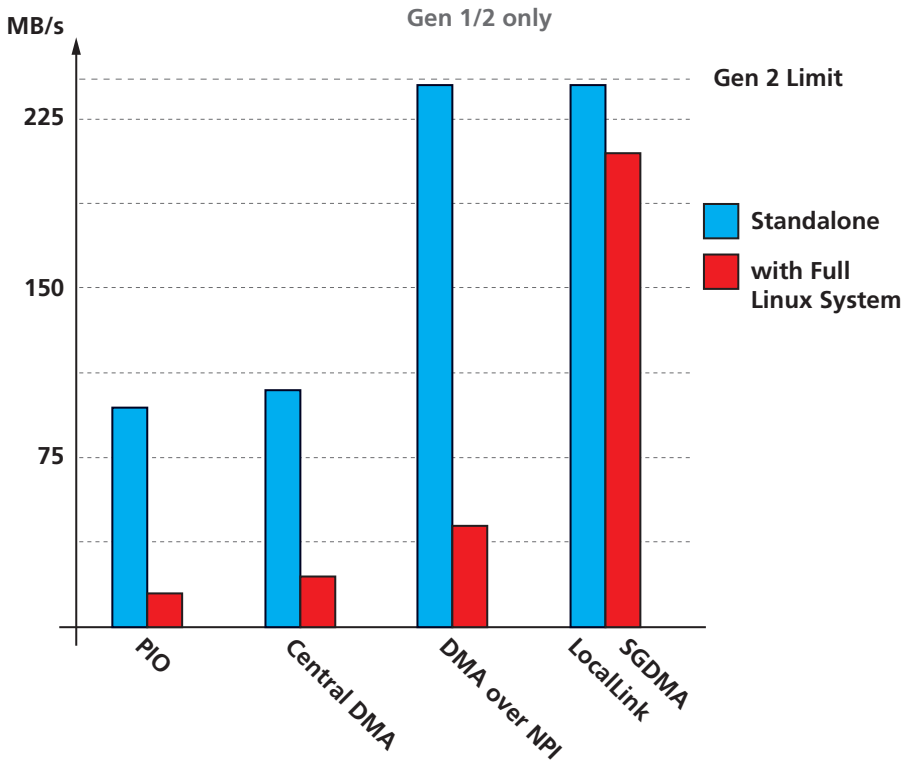

Figure 6 – Performance of complete SATA solution

larger transfer sizes and brings the system very close to the standalone performance. Figure 6 summarizes the performance results of these different architectural choices.

Today, the decision whether to make or buy a SATA host controller core is obvious: Very few design teams are capable of implementing a functioning SATA host controller for the cost of licensing one. At the same time, it is common for design teams to spend significant time and money in-house to integrate this core into a programmable system-on-chip, develop device drivers for this core and implement application software for operating (and testing) the IP.

The joint solution our team crafted would not have been possible without short turnaround times between two Xilinx Alliance Program Partners: ASICS World Service Ltd. and Missing Link Electronics, Inc. To learn more about our complete SATA solution, please visit the MLE Live Online Evaluation site at *http://www.missinglinkelectronics.com/LOE*. There, you will get more technical insight along with an invitation to test-drive a complete SATA system via the Internet.

References:

1. Xilinx, "Virtex-5 FPGA RocketIO™ GTX Transceiver User Guide," October 2009. http://www.xilinx.com/bvdocs/userguides/ug198.pdf.

2. Xilinx, Inc., "Serial ATA Physical Link Initialization with the GTP Transceiver of Virtex-5 LXT FPGAs," 2008 application note. http://www.xilinx.com/support/documentation/application_notes/xapp870.pdf.

3. Serial ATA International Organization, Serial ATA Revision 2.6, February 2007. http://www.sata-io.org/

4. International Committee for Information Technology Standards, AT Attachment 8 - ATA/ATAPI Command Set, September 2008. http://www.t13.org/