

Hardware/Software Optimization of Embedded Systems

Xilinx FPGAs provide a excellent opportunity to improve system performance during the embedded system design process.

by Peter Thorwartl
CEO/Sr. Systems Architect
So-Logic Electronic Consulting
thor@so-logic.co.at

Franz Summerauer
Hardware Engineer
AVL List GmbH
franz.summerauer@avl.com

Alfred Pöelzl
Software Engineer
AVL List GmbH
alfred.poelzl@avl.com

At the beginning of a new embedded system design, there are so many different solution options that it is hard to predict their impact on system cost, development time, and performance.

Some example of possible choices are:

- Memory devices (DDR, SRAM, flash, SPI-Flash)
- Bus architecture and topologies (PLB, OPB, OCM)
- Number of bus masters and arbitration schemes
- Processor architecture (PowerPC 405, PowerPC 440, Xilinx® MicroBlaze™ processor, ARM)
- Operating system (stand-alone, Xilinx microkernel, Embedded Linux, commercial RTOS)
- Single or multiple processors
- Interprocessor communication (shared memory, shared bus, interrupts)
- Interfaces to the outside world (Ethernet, PCI Express)

In this article, we'll show how you can evaluate these different choices during the whole design process to build an optimized system.



Figure 1 - Typical environment for the data acquisition system

System Architecture

AVL offers modular test bed components, highly reliable instrumentation, and advanced tools supporting the development and calibration process for all kinds and sizes of combustion engines (Figure 1). One part of this system is the embedded Gigabit Ethernet card discussed in this article, which collects data from as many as four data acquisition boards through parallel burst interfaces, performs basic data pre-processing, and streams the pre-processed data to a host system through a Gigabit Ethernet interface. Multiples of such acquisitions should be cascaded; therefore, we planned to use two Ethernet interfaces, as shown in Figure 2.

The main components of this card are a Virtex™-4 FX60 device, three DDR RAMs, two Gigabit Ethernet interfaces, and one XC9500XL family CPLD. Two PowerPC processors determine the functionality of the interface board.

Because of high speed requirements, the multi-port memory controller executes checksum generation and the data management of four burst data interfaces. The measurement data is placed into an external high-speed DDR RAM. Both PowerPC processors are responsible for the data flow management and Ethernet communication. Five high-speed interfaces control the external data acquisition boards.

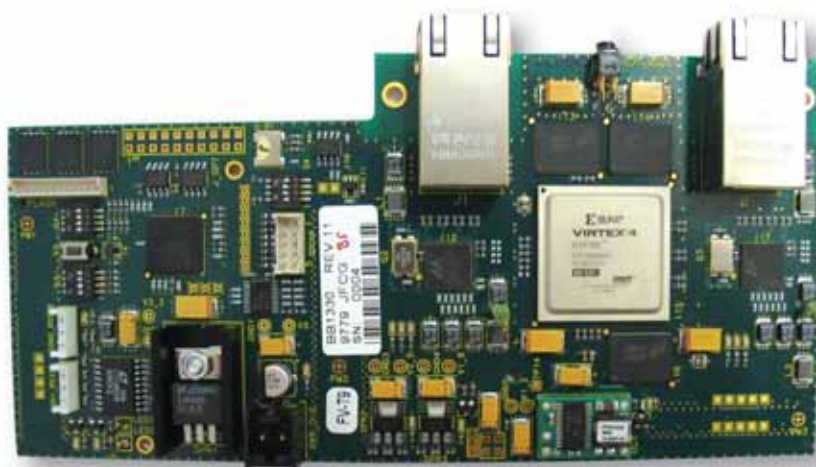


Figure 2 – Board of data acquisition system

Exploring with a Demo Board

A good way to explore the capabilities of a new FPGA family is to use a demo board like the ML403. First, we implemented the simplest possible embedded system running on one PowerPC (see Figure 3). The CPU executes the software from internal block RAMs. With this small system, we could evaluate the throughput of the Gigabit Ethernet interface. We could also determine the memory bandwidth and speed of the PowerPC.

We estimated the required logic resources like the number of flip-flops, block RAMs, PowerPCs, DCMs from the

data sheet to select the device, and the number of I/O blocks, I/O standards, and I/O banks to select the package.

After this first experience, we decided to use the Virtex-4 FX family and FF676 package, which allowed us to use different devices (FX20 with one PowerPC, FX40 and FX60 with two PowerPCs). We had to use different TCP/IP stacks to use the full performance of the Gigabit Ethernet interface, so we selected the Treck stack.

A single PowerPC is too slow to move the data from the burst interfaces to the memory and then to the Ethernet interfaces, so we had to connect the burst data

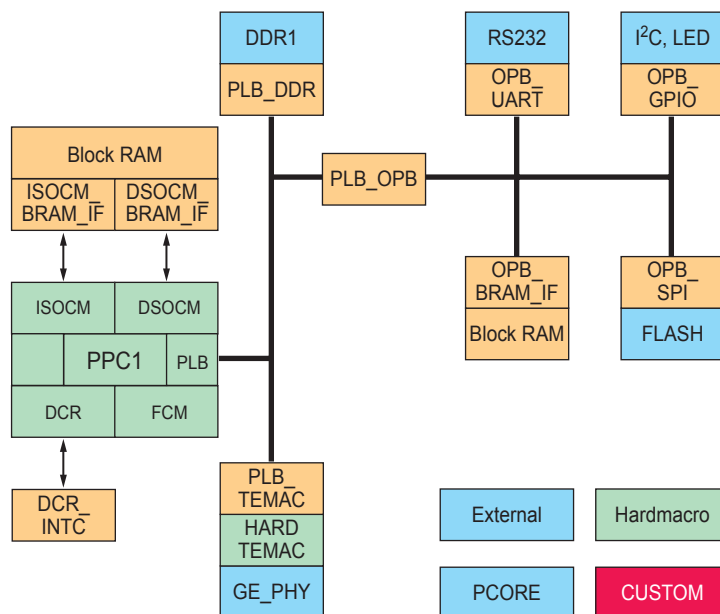


Figure 3 – Block diagram of simple system with ML403 demo board

and Ethernet interfaces directly to the memory. We decided to connect one network interface to each PowerPC.

We are sure that a platform without a real-time operating system will make the development of this application easier. This has to be decided on a case-by-case basis.

Hardware Design

The system uses three DDR RAMs: one for each PowerPC and one for the measurement data storage. We had two design possibilities: a special peripheral that is capable of bus mastering DMA transfer or a multi-port memory controller MPMC2 to enable simultaneous access to the data acquisition RAM. We selected the second option.

We were then ready to start developing our new platform. We mainly used Mentor Graphics tools: DxDesigner for schematic capture, Boardstation for PCB layout, Hyperlynx for signal simulation, and IODesigner for the interface between hardware and ISE™ software. The PCB design was awarded the first prize in the Transportation & Automotive category in the Mentor Graphics' 2007 PCB Technology Leadership Awards.

We also developed the peripheral cores for the custom peripherals. We needed two custom cores: one for transmitting data to the transmission link (SO_TLINK_FSL) and one for capturing the block data (SO_NPI_FSL_BURST).

SO_TLINK_FSL

We already had VHDL code for the transmission link with a register interface, so we decided to use a block RAM interface to connect the register memory mapped to the PowerPC. One advantage of this solution is that for every bus (OPB, PLB, OCM, and even LMB for the MicroBlaze processor), a block RAM interface controller exists, so we could easily try the performance on different bus configurations. If connected to the OPB bus, we could share this resource between the two PowerPCs. With this universal approach, we had the freedom to move the control software modules from one PowerPC to the other.

Unfortunately, this solution was too slow for two reasons. First, we saw that bus

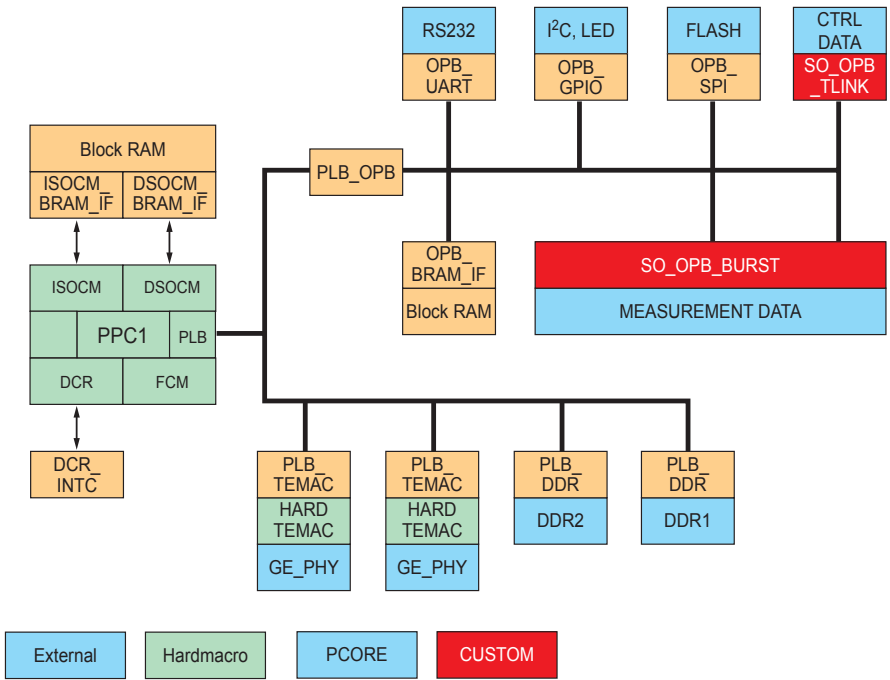


Figure 4 – Block diagram collecting performance data with PLB

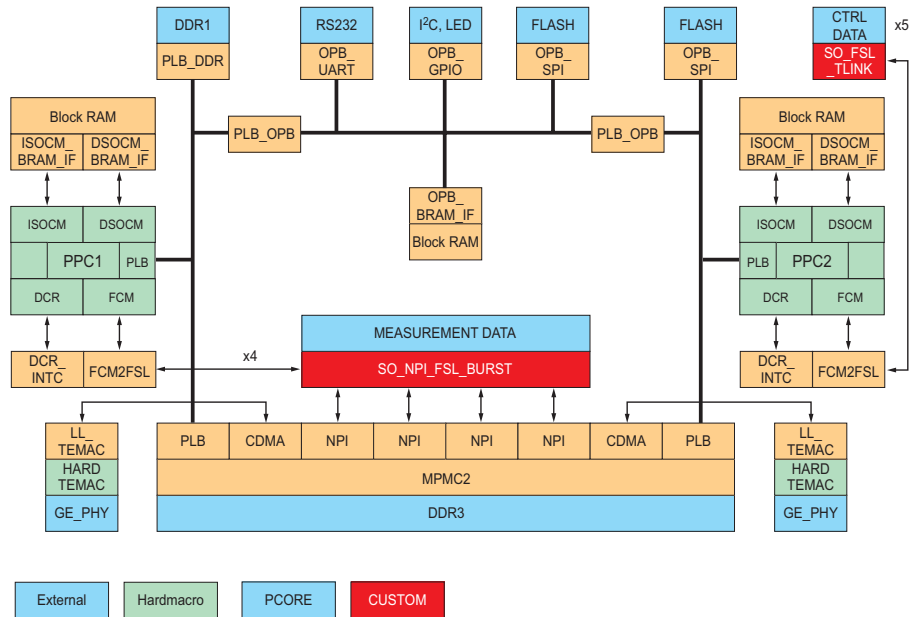


Figure 5 – Block diagram of optimized system with MPMC2

arbitration takes too many clock cycles; even worse, because we used the OPB bus, we needed an additional PLB2OPB bridge. For the OCM bus, time for bus arbitration would be lower, but we would need a second DSOCM_block RAM interface, which

slows down the whole bus frequency.

The other problem was that we could not send further data until we had acknowledgment that the packet was received. One solution would have been to integrate a FIFO in our core, but we could

also use the FSL links instead and eliminate both disadvantages. We estimated the best FIFO size, because we could optimize it later under different load conditions. The FSL interface automatically uses the SRL16 for small FIFOs and the FIFO16B hard block for larger depth.

SO_NPI_FSL_BURST

We wanted the SO_NPI_FSL_BURST core to move the data directly to the MPMC2, so we needed to develop an interface between the test card interface and NPI (native port interface similar to the local link interface) of the MPMC2. The PowerPC must control this interface to specify the address location and burst length. We decided to use the FSL interface instead of the GPIO because the FSL is really simple to handle and saves a few clock cycles.

For the simulation of the peripheral cores, we used ISIM to verify the fundamental functionality of our cores. For a system simulation with the PowerPCs, we used ModelSim PE with the SWIFT interface for the processor models.

In the implementation process, we generally used the generated make file from the Xilinx Platform Studio software and added a lot of features like checkout and tagging for source code repository, tar ball, and .zip file generation, different programming file format generation, debugging with the ChipScope™ analyzer core inserter, and compilation of different application software for both PowerPCs.

Our scripts ran the development tools on Windows XP and Linux (CentOS 5.1) and supported different boards and devices.

Real Life

We first migrated our design from ML403 to our new target board to check all external peripherals (DDR, Gigabit Ethernet, SPI, I²C) (see Figure 4). We can reuse this design later for hardware test for manufacturing.

We then used small internal block RAMs on the data and instruction side OCM buses, with boot loaders for each PowerPC and fast interrupt routines. The boot loader moves the data from the SPI flash to the dedicated DDR RAM. No parallel flash is needed. We use the same SPI

flash for booting the FPGA. The CPLD converts the signals from SPI to the usual serial interface for booting.

We implemented the I²C with the OPB_GPIO core and a software driver without the special core OPB_IIC core, because it was free of charge and we needed only to read an ID tag once after boot.

For the interprocessor communication between the two PowerPCs, we used a shared memory implemented in a single ported block RAM connected to the OPB bus with the OPB_BRAM controller.

We implemented two separated interrupt controllers, one for each PowerPC connected to the DCR bus.

The central core in our design is the multiport memory controller with eight ports: four for data acquisition (SO_NPI_FSL_BURST), two PLB ports, one for each PowerPC, and two CDMA ports to connect the Gigabit Ethernet interfaces (Figure 5).

An interesting design phase was finding performance bottlenecks and fine-tuning the system performance. We connected our board to real test equipment to get real-life data.


One very useful tool for qualifying the results is the combined XMD and ChipScope analyzer for hardware/software co-debugging. You can trigger with the ChipScope analyzer and stop the software debugger, and vice versa.

We also experimented with interrupt priorities, FIFO sizes on the MPMC2, and arbitration schemes inside the MPMC2. We also moved some software modules from PowerPC1 to PowerPC2 and vice versa to optimize system performance.

Conclusion

You can start with a very simple design and upgrade step by step to a higher performance design by boosting Ethernet throughput, boosting DDR throughput with a multiport memory controller, adding data filtering, FFTs, and so on.

In the future, it will be necessary more often than not to postpone important system architecture decisions to later stages during the design process. With an FPGA embedded system, you have many opportunities to remove bottlenecks from your system.

For more information, visit www.so-logic.net or www.avl.com. 

Further Improvements

- Upgrading the actual design environment to the newest versions of ISE software and EDK 9.2.
- Moving all OPB cores to the PLB bus, because all OPB cores are now also available for PLB. In this case, we had to decide which shared peripheral is connected to which PowerPC. As a positive consequence, we could remove the whole OPB bus and two relative large PLB2OPB bridges, each saving 500 slices.
- Running embedded Linux on one or both PowerPCs if you needed to run one of the many applications that are available for Linux.
- Adding cores for double-precision floating-point to offload the host PC application or integrate more DSP functionality into the FPGA.
- Downgrading to a lower performance system with only one PowerPC or, as an even more low-cost solution, replacing the PowerPC with the MicroBlaze processor v7.0, which also has a PLB bus interface.
- Upgrading to a higher performance Virtex-5 FXT FPGA with PowerPC 440 and removing the CPLD, because of the more user I/O pins available on the FPGA in the same package and native boot support from SPI flash.
- Upgrading the FIFO links to serial point-to-point, high-speed links with RocketIO™ transceivers to improve data transfer speed.