

Quickly Build Distributed Systems

Mobius generates both hardware and software for XPS-based projects.

by Per Ljung
 President
 Codetronix LLC
ljung@codetronix.com

Many applications use a distributed architecture, including multi-FPGA systems and FPGA-accelerators for microprocessors. Creating master-slave or peer-to-peer distributed systems is complex and requires a design methodology that supports efficient partitioning, as well as an efficient implementation for communication and synchronization.

In this article, I'll present the Mobius tools and work flow for distributed systems. Mobius is a high-level language that can generate both efficient hardware (Verilog, VHDL) and software (C with scheduler) from a single source. Mobius uses handshaking for multi-threaded synchronization and communication, resulting in a target-independent, latency-insensitive handshaking processes. Mobius is suitable for both control- and data-dominated applications. Figure 1 shows how Mobius is a front-end to the Xilinx® Platform Studio tool.

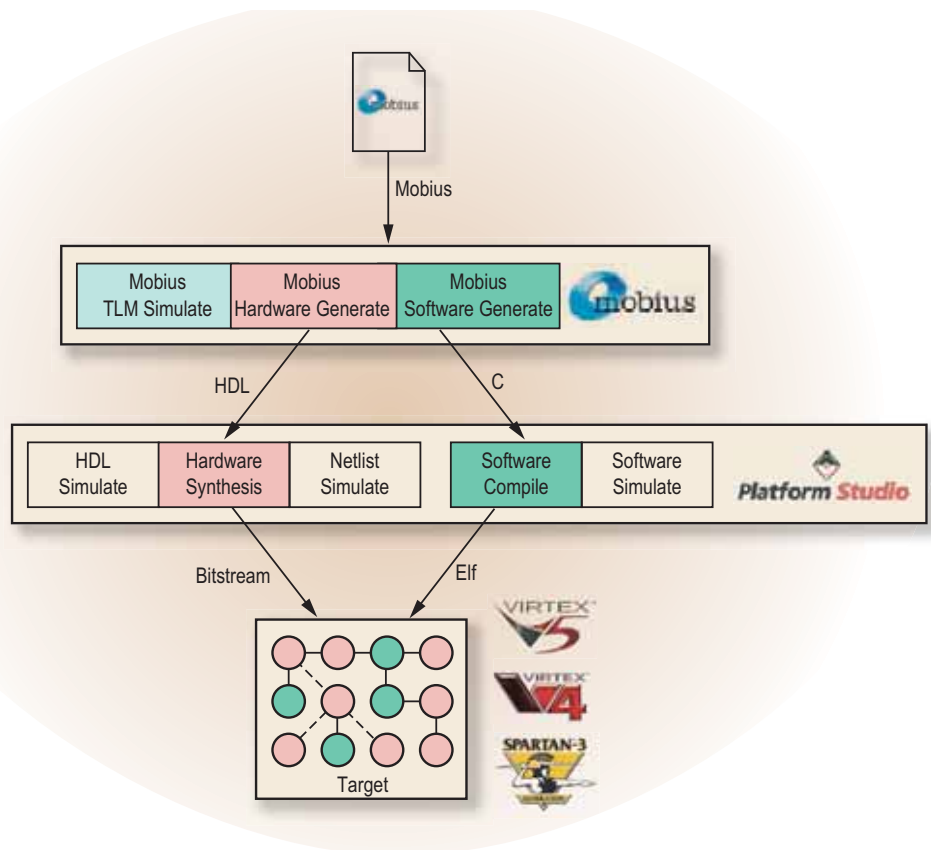


Figure 1 – Mobius is a front-end to Xilinx Platform Studio tools.

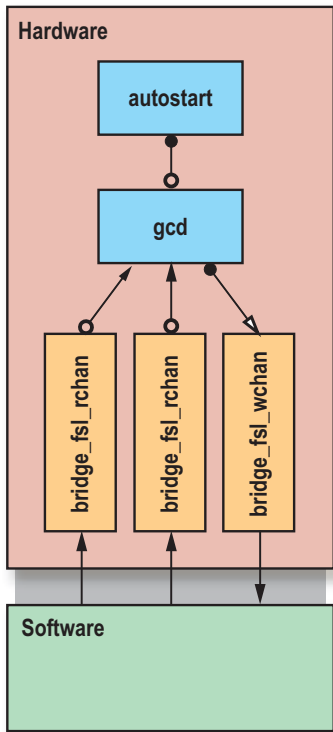


Figure 4 – Mobius-generated hardware and software connected by bridges

dot and a hollow dot. The solid dot denotes an active port that initiates the handshake, and the hollow dot denotes a passive port that waits for the handshake. A push or pull channel has arrow heads signifying the direction of data flow, where the arrow head may be hollow or solid, signifying an active or passive protocol.

Bridges between Hardware and Software

Channels are transport layer-independent and can use wires, SERDES, or packets as appropriate. You can insert bridges on any channel and allow full-featured support of legacy interconnects, including OCP-IP, Wishbone, PLB, FSL, PCI, Ethernet, or RocketIO™ transceivers. Similarly, channels allow domain crossing – between software and hardware, for instance – between different clock domains or between different voltage domains. As a result, implementing distributed and hardware/software systems using handshaking is a straightforward process.

Consider a simple example where the GCD is implemented as a hardware cir-

cuit, while the test bench runs in C on a MicroBlaze processor. The Mobius compiler identifies that the reader and writer of each channel are in different domains and emits C for the software domain, HDL for the hardware domain, and FSL bridges to connect the hardware and software.

The Xilinx fast simplex link (FSL) is a peer-to-peer, unidirectional, point-to-point synchronization and communication protocol using queues, which maps easily to Mobius channels. Figure 4 shows how the software writes two values to two FSL bridges and then reads a value from a third FSL bridge that is blocking until the output is available. Simultaneously, the hardware reads two values from its two input channels and computes a result that is written to its single output channel. An autostart() module generates a req pulse to start the Mobius-generated hardware on a system reset.

Automatic XPS Projects

Assembling a complex hardware and software design in Xilinx Platform Studio can be challenging. To enable users to rapidly generate correct XPS projects, Codetronix provides a utility called xpscustum that adds Mobius-generated

software applications and hardware IP to an existing Xilinx Platform Studio project.

As shown in Figure 5, xpscustum adds Mobius-generated hardware IP, bridges, and wire connections. On the software side, xpscustum automatically adds simple IP drivers, a software-only application, and a co-design application. This results in a push-button flow where you only have to compile the software and generate the bit-stream for a distributed implementation.

Conclusion

Mobius allows you to rapidly develop high-quality hardware and software solutions. You can map Mobius-generated HDL and C to arbitrary voltage, clock, physical, or co-design domains and then connect these domains using synchronous channels, asynchronous channels, or bridges to build distributed embedded systems on any Xilinx FPGA. Mobius also compiles your test bench to hardware and software, allowing functional verification in both software and hardware implementations.

Users benefit from significantly higher design productivity, letting them generate and explore alternative microarchitectures to optimize their system. For more information about using Mobius in your next design, visit www.codetronix.com.

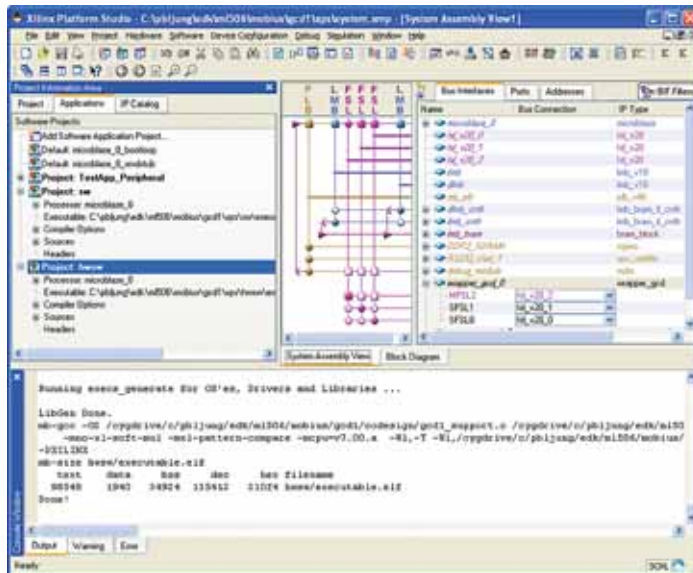


Figure 5 – Hardware and software integrated into a Xilinx Platform Studio project by xpscustum