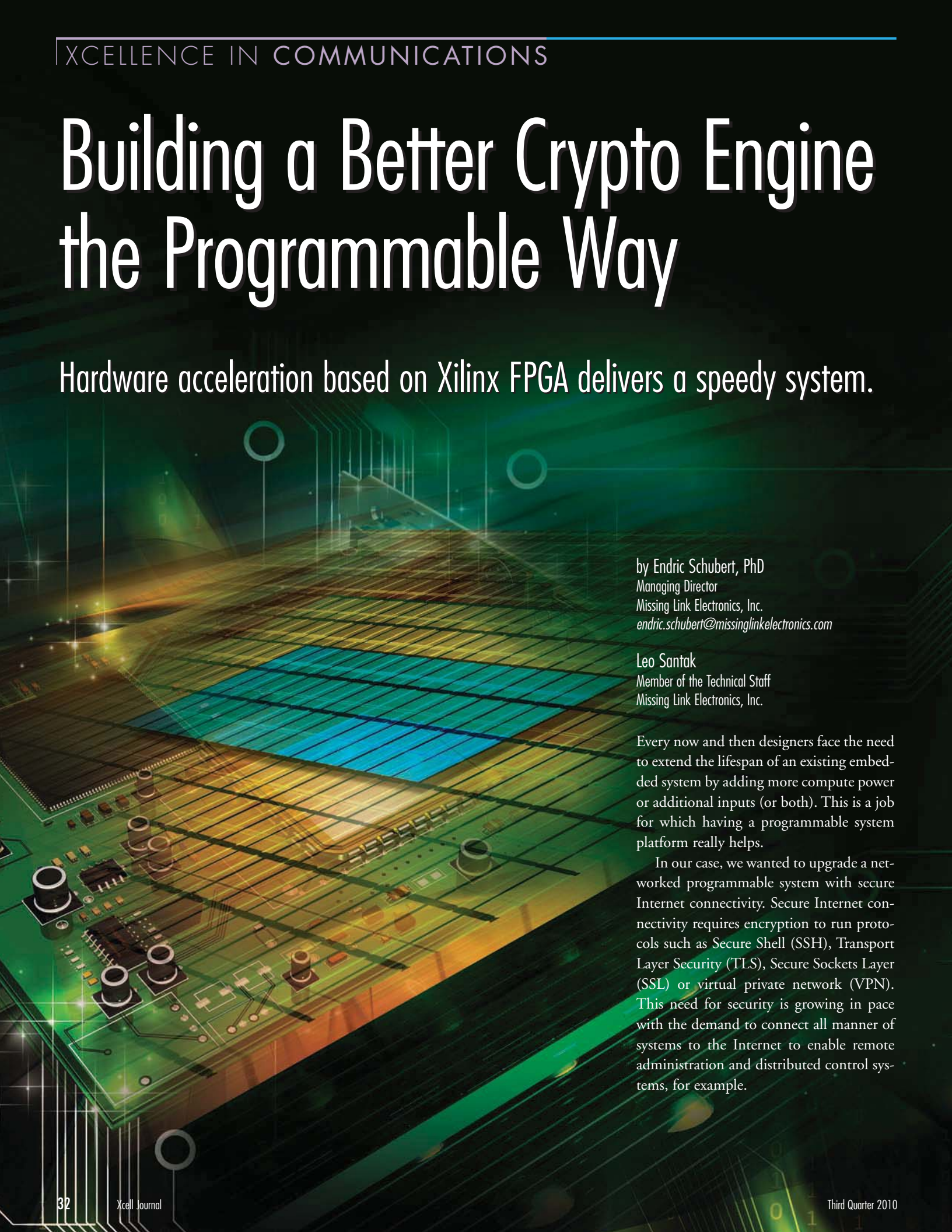
Building a Better Crypto Engine the Programmable Way

Hardware acceleration based on Xilinx FPGA delivers a speedy system.



by Endric Schubert, PhD
Managing Director
Missing Link Electronics, Inc.
endric.schubert@missinglinkelectronics.com

Leo Santak
Member of the Technical Staff
Missing Link Electronics, Inc.

Every now and then designers face the need to extend the lifespan of an existing embedded system by adding more compute power or additional inputs (or both). This is a job for which having a programmable system platform really helps.

In our case, we wanted to upgrade a networked programmable system with secure Internet connectivity. Secure Internet connectivity requires encryption to run protocols such as Secure Shell (SSH), Transport Layer Security (TLS), Secure Sockets Layer (SSL) or virtual private network (VPN). This need for security is growing in pace with the demand to connect all manner of systems to the Internet to enable remote administration and distributed control systems, for example.

Because this field is still evolving and standards are not yet set, costs are dominated by nonrecurring-engineering fees. Therefore, FPGA technology offers the best value for implementations.

Our system was built on top of the Missing Link Electronics (MLE) "Soft" Hardware Platform, where the flexible I/Os of the FPGA enable connection to a wide range of sensors and actuators. This platform uses the programmable logic to implement a system-on-chip with either the MicroBlaze™ CPU or the PowerPC® CPU at its heart. The CPU runs the MLE Linux software stack for the operating system and the user-space application software. With the MicroBlaze or the PowerPC as the main CPU, the system was obviously not suitable for delivering the required compute performance when running embedded Linux plus strong encryption on top. And changing the physical hardware was not an option.

Instead, we utilized the power of programmable systems to migrate computations from the software domain to the hardware side for system acceleration.

Coprocessing Hardware

A programmable system is basically a combination of one or more CPUs—running an operating system and application software—plus an FPGA. The FPGA is there as a flexible interface "adapter" and as coprocessing hardware. You can make programmable systems from separate companion chips or integrate everything into one single device. Depending on how the FPGA device and the CPU are communicating with each other, you have different options in adjusting the system for performance and functionality.

One possibility is to add a peer processor, which synchronizes with the CPU via memory-mapped status and control registers. Because running all communication over the same system bus may quickly suffocate performance, you really want to separate the data stream of the CPU from the peer processor. This is easy to do by using system-on-chip components such as the Xilinx Central DMA or the Multiport Memory Controller (MPMC).

Alternatively, you can add a coprocessor, in which case you effectively extend the instruction set of the CPU by adding custom instructions (also called compiler-known functions). This is, for example, the case for floating-point units, and the Xilinx technology of Fabric Coprocessor Modules (FCM) readily supports that. The advantage here is to free up the memory-to-system bus by using a dedicated communication channel between the CPU and the coprocessor. For the PowerPC this is the Auxiliary Processing Unit (APU) and for MicroBlaze, the Fast Simplex Link (FSL).

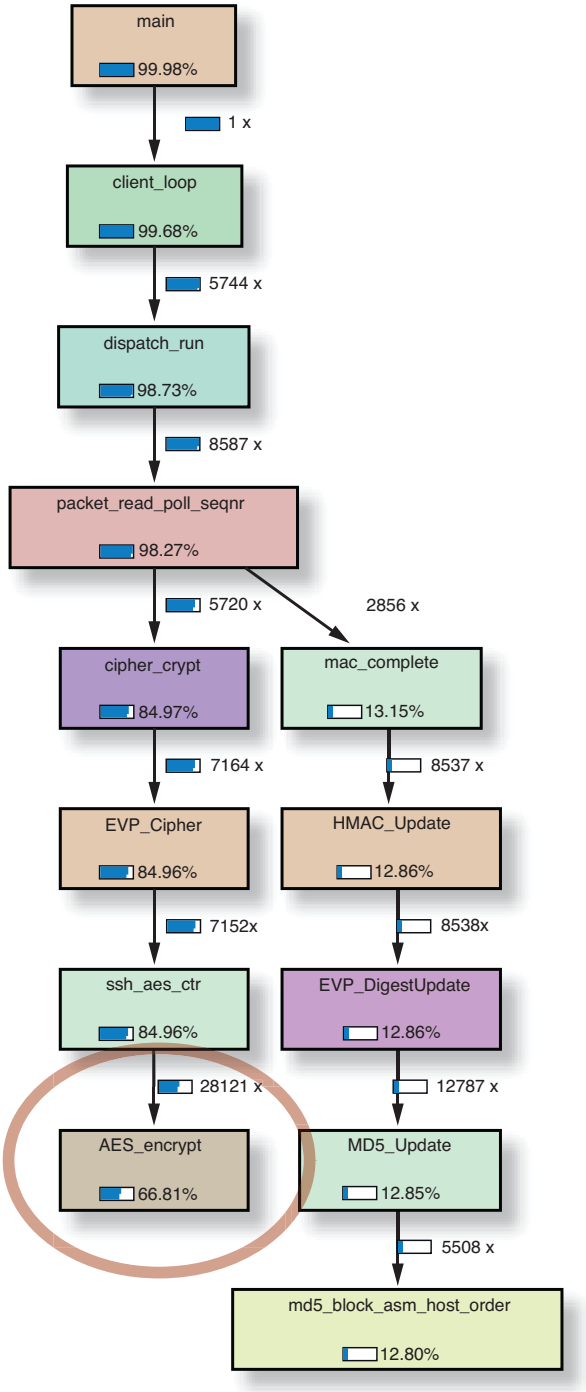


Figure 1 – In an SCP transfer using the Valgrind tool, the AES encryption occupies two-thirds of the computations.

In encryption and decryption, most of the operations are performed on either the rows or the columns, leaving four operations that can be calculated in parallel—a job well suited for hardware.

AES: the Gold Standard

But how do you really accelerate encryption without a major system redesign?

For encryption, the Advanced Encryption Standard (AES) is really the de facto standard. With AES encryption, the computations are irreducible by definition, bringing an embedded system quickly to its performance limits. This is clearly illustrated in Figure 1, which shows the profiling results of a file transfer with SCP (SSH session) using the Valgrind analysis tool. In this case, the AES encryption takes up two-thirds of the computations.

AES-128, with a key and block length of 128 bits, utilizes many concurrent 8-byte operations. AES is a block cipher and operates on fixed block sizes organized as a 4 x 4 array of bytes. We used a 128-bit block size, which withstands all known attacks and is even supposed to be more secure than the 192-bit and 256-bit versions.

With 128-bit AES, it takes 12 rounds, each with several steps, to perform the encryption and decryption. The first task is to compute the round keys from the secret key by means of the so-called key expansion process. In every round, the plain text is bit-wise XOR-ed with its own round key. Then sub-byte, row-shifting and column-mixing operations follow, and the round key gets XOR-ed once again.

The final round slightly differs, omitting some steps. The encryption process performs substitution using a so-called S-box, which provides nonlinearity. We can arrange it in a 16 x 16 x 8-bit matrix so that it generously fits into the common Xilinx BRAM primitives. Several S-box instances speed up the IP core and supply the core in place with the data needed, without waiting on long-lasting bus accesses to main memory. The decryption occurs in a similar fashion, using the same secret key, but in the opposite direction and with a different S-box.

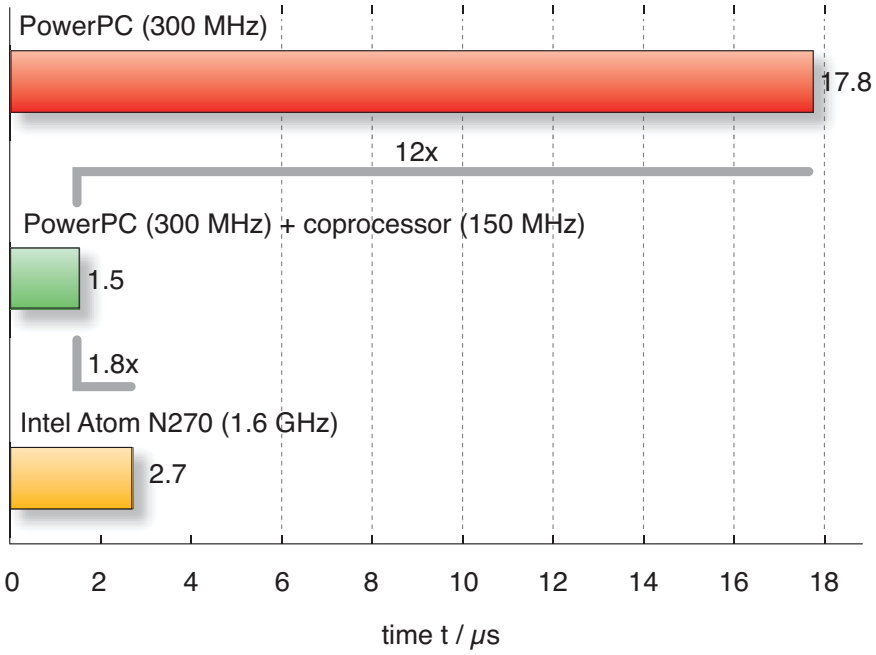


Figure 2 – The hardware-accelerated system (green bar, center) ran faster than a standalone PowerPC or an Atom processor.

12 Times Faster

In encryption and decryption, most of the operations are performed on either the rows or the columns, leaving four operations that can be calculated in parallel—a job well suited for hardware. Thus, various hardware implementations of AES are available from different sources. To accelerate our system, we took an AES core from the great and fast-growing OpenCores.org repository (*http://opencores.org/project,avs_aes*). We removed the original bus interface, which was targeted for another FPGA architecture, and added an interface for the APU to connect the AES core as an FCM coprocessor to a PowerPC. We used a total of eight so-called UDI commands to transfer data between the PowerPC and the AES FCM.

The result of that work was very satisfying (see Figure 2). The hardware-accelerated system ran 12 times faster than the original implementation. It took 17.8 microseconds to encrypt one single block using a standalone PowerPC running at 300 MHz, but only 1.5 µs to do this with an AES FCM running at 150 MHz. For those who are tempted to just switch to a faster CPU for a speedup, our hardware-accelerated speed of 1.5 µs outperformed a pure-software implementation on an Intel Atom 1.6-GHz CPU, which took 2.7 µs.

These results demonstrate the outstanding potential of hardware acceleration using FPGA technology. For details of the analysis and exemplary code, contact our applications team at *http://www.missinglinkelectronics.com.*