Celoxica Implements "Soft Hardware" in Internet Reconfigurable MultiMedia Terminals

Xilinx FPGAs enable rapid deployment of reconfigurable "soft hardware" over the Internet.

by Ranjith Kumaran
Field Applications Engineer, Celoxica Inc.
ranjith.kumaran@celoxica.com
Stephen P. G. Chappell
Design Services Manager, Celoxica Ltd.
chappell@celoxica.com

Figure 1 - MultiMedia Terminal

So, you want your video game console to play MP3s, and you wish you could send your voice over the Internet using that digital audio appliance. Is it too much to ask one device to do it all? We don't think so. This article details the development of a flexible multimedia device with hardware that can be reconfigured over a network connection and that can run software applications built directly into silicon.

We call the platform we developed for this purpose an MMT (MultiMedia Terminal –Figure 1). The MMT features no dedicated stored programs and no CPU. It has no operating system, and the applications that it runs in hardware are kept on a server rather than on the board itself. Instead, programs are implemented in field programmable gate arrays. With FPGAs, you can control peripherals and process data to mimic CPU flexibility using only reconfigurable logic and a software design methodology.

FPGAs can be used to host "soft hardware" that runs applications without the overhead associated with microprocessors and operating systems. Such hardware can be totally reconfigured over a network connection to install program enhancements and fixes – or a completely new application.

Hardware platforms populated by FPGAs can stave off premature obsolescence because they are able to support evolving standards, as well as applications not imagined when the platform was designed. This soft hardware model also allows manufacturers to use Xilinx IRL™ (Internet Reconfigurable Logic) to remotely access and maintain their hardware designs at any time, regardless of where the MMT units reside.

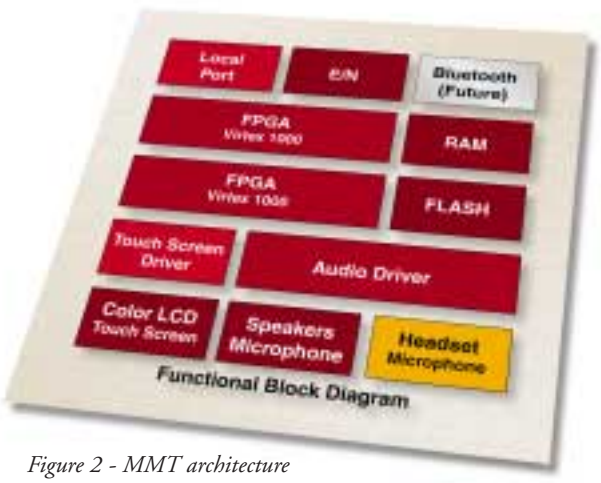

Figure 2 - MMT architecture

Master/Slave Architecture Enables Reconfigurability

The MMT achieves reconfigurability by using two independent, one million-gate Xilinx XCV1000 Virtex™ FPGAs (Figure 2). One of the FPGAs – the master – remains statically configured with networking functionality when the device is switched on. The other FPGA – the slave – is reconfigured with data provided by the master. The two FPGAs communicate directly via a 36-bit bus with 4 bits reserved for handshaking and two 16-bit unidirectional channels. This protocol ensures that reliable communication is available even when the two FPGAs are being clocked at different speeds. The other components of the MMT are an LCD touch screen, audio chip, 10-Mbps Ethernet interface, parallel and serial ports, three RAM banks, and a single nonvolatile flash memory chip.

FPGA reconfiguration can be performed by using one of two methods. The first method implements the Xilinx SelectMAP™ programming protocol on the master FPGA, which can then program the slave. The second method supplies reconfiguration data from the network interface or from the flash memory on the MMT.

Reconfiguration from flash memory is used only to load the GUI (Graphical User Interface) for a VoIP (Voice-over-Internet Protocol) telephone into the slave FPGA upon power-up, when an application has finished, or when configuration via the network fails.

Network-based reconfiguration uses HTTP (HyperText Transfer Protocol) over a TCP (Transmission Control Protocol) connection to a server. A text string containing a file request is sent by the MMT to the server, which then sends back the reconfiguration data (a bitfile).

It's all well and good to have a flexible architecture that can run your application-of-the-day in an FPGA, but how do you write all those applications and how do you do it in a reasonable amount of time?

HDLs (Hardware Description Languages) are well suited for creating interface logic and for defining hardware designs with low-level timing issues. However, networking, VoIP, MP3s, and video games - that almost sounds like software. ...

Getting a Handel–C on "Soft Hardware"

To meet the challenges of the system described above, the MMT was designed with a language called Handel-C™, which is featured in the Celoxica™ DK1 software design suite for reconfigurable hardware. The Handel-C language extends ANSI-C for efficient hardware implementation. If you are familiar with C software development, you can learn Handel-C quickly.

The Handel-C extensions support parallelism, variables of arbitrary width, and other features familiar in hardware design, but Handel-C very much targets software design methodologies (Figure 3). Unlike some of the other C-based solutions out there that translate C into an HDL, the Handel-C compiler directly synthesizes an EDIF (Electronic Design Interchange Format) netlist in a format that can be immediately placed and routed, and put onto an FPGA as soft hardware.

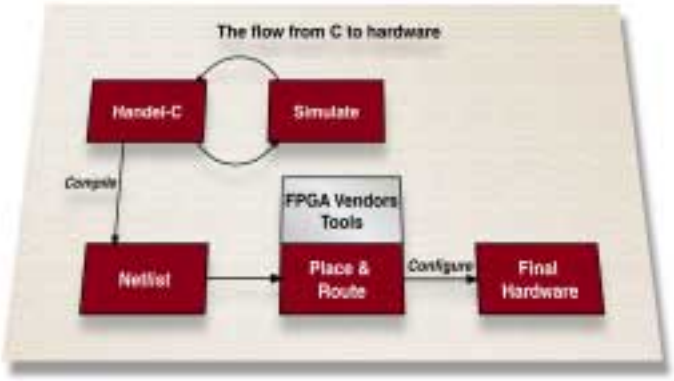

Figure 3 - Handel-C design flow

The default application that runs on the MMT upon power-up is a VoIP telephone complete with GUI. The VoIP consists of a call state machine, a mechanism to negotiate calls, and an RTP (Real Time Protocol) module for sound processing. A combination of messages from the GUI and the call negotiation unit are used to drive the call state machine. The protocol implemented by the call negotiation unit is a subset of H.323 Faststart (including H.225 and Q.931). This protocol uses TCP to establish a stream-based connection between the two IP (Internet Protocol) telephones. The RTP module is responsible for processing incoming sound packets and generating outgoing packets sent over UDP (User Datagram Protocol).

Pipeline to Success

The development team consisted of four Celoxica engineers. It took the team only three weeks to get the VoIP prototype to send and receive voice calls. Algorithms for protocols such as RTP, TCP, IP, and UDP were derived from existing public domain C sources. The team optimized the source code to use features available in Handel-C, such as parallelism. Parallelism is useful for network protocols that generally require fields in a packet header to be read in succession and that can usually be performed by a pipeline with stages running in parallel. Each stage was tested and simulated within a single Handel-C environment. Once each stage tested okay, it was put directly into hardware by generating an EDIF netlist. The development team was able to quickly perform further optimizations and tuning simply by downloading the latest version onto the MMT over the network.

In order to demonstrate the flexibility of the architecture and to take advantage of Internet reconfigurability, the team developed a mixed bag of applications. These applications all run successfully in hardware on the MMT. Among them are a fully functional MP3 player with GUI, several video games, and some impressive graphics demonstrations that were all developed using Handel-C. These applications are hosted as bitfiles on a server that supplies these files upon demand from the user of the MMT over a network connection.

Three software engineers using Handel-C and one board designer took the MMT design from specification to hardware in just three months – a near impossible schedule using alternative hardware development methods. The final design included a TCP/IP stack that used about 48% of an XCV1000 Virtex device (13% of which could be moved off the FPGA into external or block RAM). A simple video game used 2% of an XCV1000, and the MP3 decoder used about 80% of an XCV1000.

Because the MP3 design implemented many pipeline stages in parallel, it required only an 8 MHz clock to perform real-time decoding. The MMT developers believe they can reduce the size of the decoder dramatically given time for further optimizations.

To find out more about the rapid design capabilities of Handel-C, the DK1 software design suite for reconfigurable hardware, and the MMT architecture, visit Celoxica Inc. at www.celoxica.com.