

## Fast, Flexible FPGA/FPGA Processor Memory Configuration

*If serial Flash is taking too long to configure your large FPGAs, here's a low-cost, PLD-based strategy for accelerating code loads in your next design.*

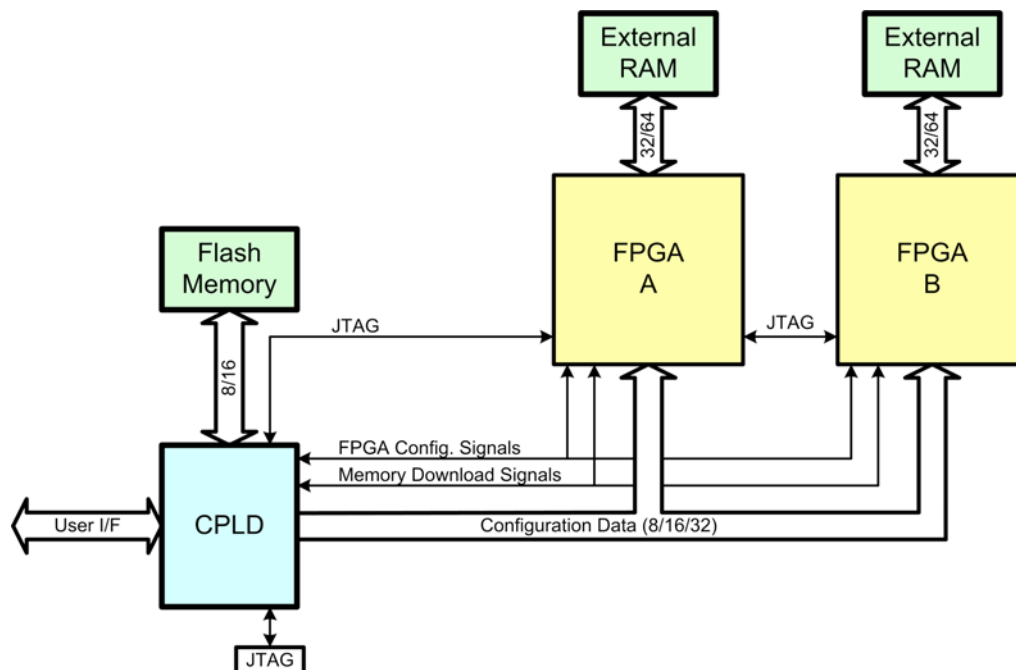
*by Jim Weyand, Principal Systems Engineer, Embedded Systems Design, Inc*

In many applications, FPGAs and the external volatile memory used for embedded FPGA processors are configured at power-up from serial EPROMs. Although easy to implement, this configuration scheme is extremely slow, provides very limited capacity for multiple FPGA images and processor software applications, and is difficult for a user to maintain. For true reconfigurable processing applications this simply is not an adequate solution.

What's needed is an architecture that addresses these issues. This new architecture should:

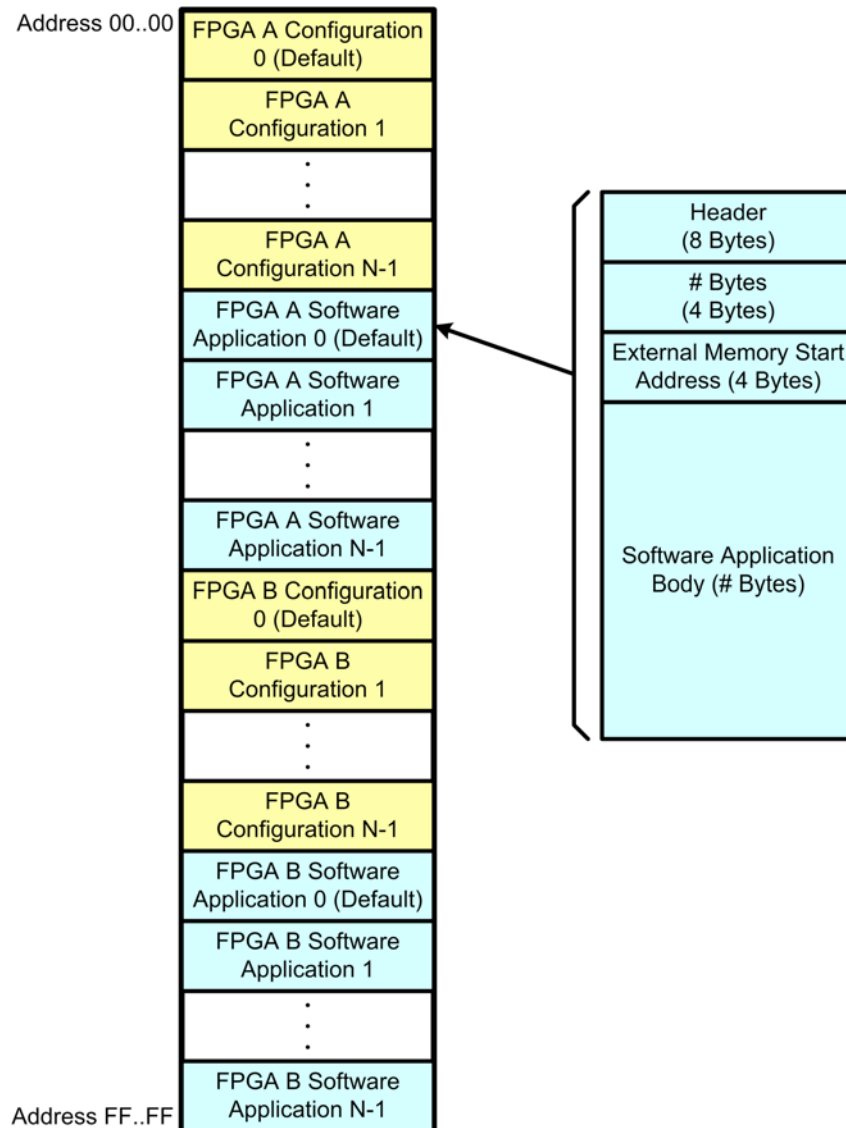
- Use the parallel configuration modes available in today's FPGAs to dramatically shorten download times
- Leverage commercially-available high-density Flash memories to provide storage for multiple FPGA configurations and processor software applications
- Allow users to modify FPGA configurations and processor software applications stored in Flash memory and activate downloading of these files from Flash

In a board-level architecture that provides these capabilities (see Fig. 1) a CPLD with a user interface provides an 8-bit or 16-bit interface to Flash memory. Via the user interface, Flash maintenance functions such as chip/sector erasures, programming, and read accesses may be performed. A CPLD is ideal for this role since it is a non-volatile device with the structures necessary to implement the required control and dataflow functions efficiently. Likewise, commercially-available Flash memory with its large storage capacity (up to 32 Gbit) can easily accommodate numerous FPGA configuration files (a Xilinx Virtex-5 XC5VLX330 requires 79,704,832 configuration bits) and processor software applications.



**Fig. 1: Board-level Architecture Block Diagram**

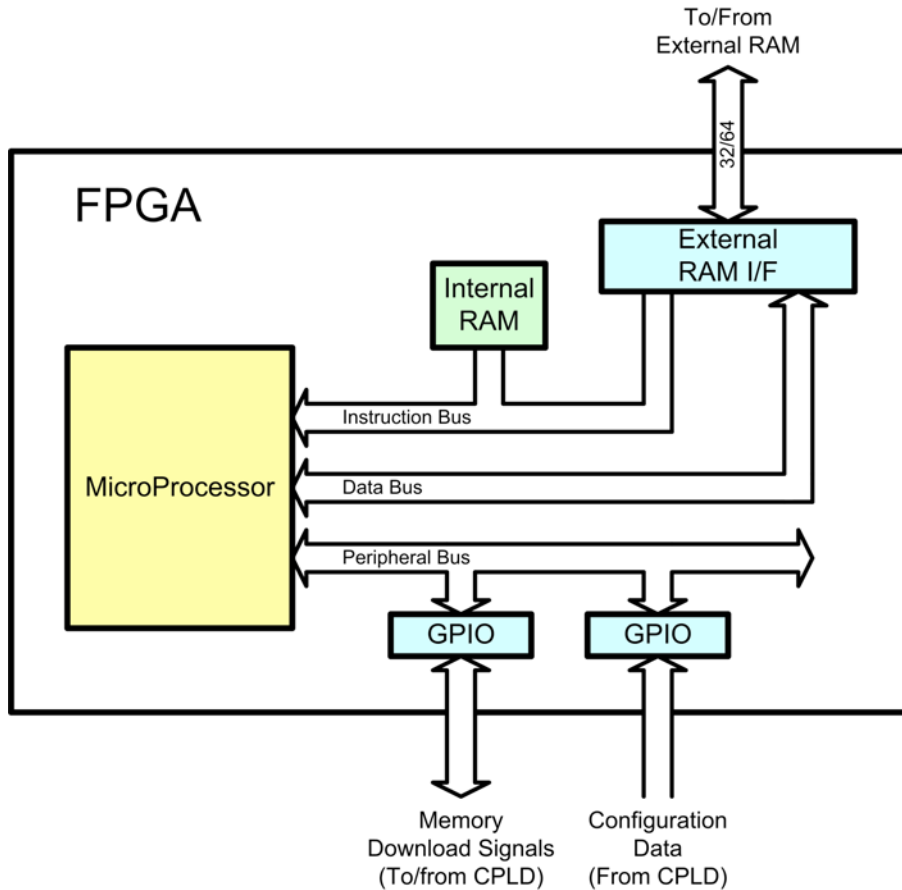
An example of a Flash memory map is shown in Fig. 2. There are up to N FPGA configuration files provided for FPGA A followed by up to N software applications for an embedded processor located in FPGA A. The same structure follows for FPGA B. Each software application file is typically structured to contain a header, byte count, external memory start address and the body of the software application itself. This Flash memory map can easily be scaled to accommodate any number of FPGAs, FPGA configuration files, and software applications.



**Fig. 2.: Flash Memory Map**

The CPLD controls configuration of the FPGAs at power-up or when commanded via the user interface. At power-up, the CPLD simply configures each FPGA with its default Flash load over an 8-, 16- or 32-bit parallel FPGA configuration bus. Almost all FPGAs support byte-wide programming (referred to as *Slave Parallel Mode* in Xilinx literature) with more recent FPGA families providing support for up to 32-bit programming. When FPGA configuration is commanded via the user interface, the user simply provides the CPLD with the FPGA to configure and FPGA configuration file number and the CPLD executes the FPGA configuration process.

Once an FPGA is configured (see Fig. 3), transfer of the default processor application from Flash to external memory begins automatically. The microprocessor within the FPGA starts running a small boot loader software application, resident in internal RAM, within the FPGA. The contents of this internal RAM are downloaded as part of the FPGA configuration file. During execution of the boot loader program, the microprocessor accesses a GPIO port over its peripheral bus to activate a data request line to the CPLD. The microprocessor then continuously polls a GPIO port waiting for an acknowledge signal from the CPLD.



**Fig. 3: FPGA Internal Architecture Block Diagram**

Unlike FPGA configuration, during this transfer the CPLD acts as a slave to the FPGA. When the CPLD receives a data request, it reads software application data from the default section of Flash memory, provides it to the FPGA over the configuration bus and activates the acknowledge signal. When the FPGA receives the acknowledge signal via its GPIO port, it captures and stores the received configuration data. This process is repeated until the entire software application has been read into the FPGA by the microprocessor. Initially, the boot loader program reads and stores the header, byte length, and external memory address words. The boot loader then forwards the remaining words to external memory using the proper offset from the initial external memory address.

If desired, this architecture may be enhanced to provide additional capabilities. The CPLD design may be augmented to support FPGA configuration directly from the user interface. For this function, the CPLD would translate configuration data received over the user interface to FPGA configuration accesses. Also, a user interface may be added to the FPGAs along with the

necessary application software to support download of external memory directly from the user interface or Flash. In the case of download from Flash, the CPLD would first be commanded via its user interface to select the desired software application.

Perhaps the most useful feature of this architecture is its scalability. It is easily scaled to support configuration of multiple FPGAs with multiple configuration files and multiple embedded processors with multiple applications. It is also extensible to various FPGA devices, Flash memory devices, and user interfaces.

## Useful Links

Xilinx Virtex-5 FPGA Documentation:

[http://www.xilinx.com/support/documentation/data\\_sheets/ds100.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf) (Virtex-5 Family Overview)

[http://www.xilinx.com/support/documentation/data\\_sheets/ds202.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf) (Virtex-5 FPGA Data Sheet: DC and Switching Characteristics)

[http://www.xilinx.com/support/documentation/user\\_guides/ug190.pdf](http://www.xilinx.com/support/documentation/user_guides/ug190.pdf) (Virtex-5 FPGA User Guide)

[http://www.xilinx.com/support/documentation/user\\_guides/ug191.pdf](http://www.xilinx.com/support/documentation/user_guides/ug191.pdf) (Virtex-5 FPGA Configuration User Guide)

Xilinx CoolRunner-II CPLD Data Sheets:

[http://www.xilinx.com/support/documentation/data\\_sheets/ds090.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds090.pdf) (CoolRunner-II CPLD Family Datasheet)

[http://www.xilinx.com/support/documentation/data\\_sheets/ds094.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds094.pdf) (XC2C256 CoolRunner-II CPLD Datasheet)

[http://www.xilinx.com/support/documentation/data\\_sheets/ds096.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds096.pdf) (XC2C512 CoolRunner-II CPLD Datasheet)

Xilinx Application Notes:

[http://www.xilinx.com/support/documentation/application\\_notes/xapp137.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp137.pdf) (Configuring Virtex FPGAs from Parallel EPROMs with a CPLD)

[http://www.xilinx.com/support/documentation/application\\_notes/xapp502.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp502.pdf) (Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode)

## About The Author

Jim Weyand has been a Principal Systems Engineer at Embedded Systems Design, Inc <http://www.embedded-sys.com> in Elkridge, Maryland since 2004. His previous employers include General Dynamics Advanced Information Systems and Northrop Grumman Electronic Sensor and Systems. He has over twenty years experience in the specification, design, simulation, integration and test of real-time embedded processing systems. During this time, his primary specialization has been in the areas of ASIC and FPGA design. He earned a BSEE at Lehigh University and an MSEE at Johns Hopkins University. [jim.weyand@embedded-sys.com](mailto:jim.weyand@embedded-sys.com)

