Advanced Programmable System Configuration Management

This Technical Brief describes how the MLE “Soft” Hardware Platform implements advanced configuration management by utilizing CPLD devices that can be used for failsafe configuration of a Programmable System and for monitoring the system’s health status. After a short introduction to the startup methodology this paper describes how MLE uses a lean command interpreter implemented inside the CPLD and interoperating with the FPGA-based System-on-Chip to deliver a powerful configuration management. This communication between CPLD and FPGA allows failsafe reconfiguration of the FPGA as well as hardware watchdog functionality.
Programmable Systems as used in Next-Generation Embedded Systems and networked compute platforms for Industrial, Scientific and Medical (ISM) systems require high degrees of optimization of the hardware and the software components. For designing such systems Missing Link Electronics provides platforms where a configurable microcontroller can be optimized towards the Open Source Software that runs on it and the particular application that the system must serve. Most functionality is implemented as an FPGA-based System-on-Chip which runs a full software stack comprising a boot-loader, a Linux operating system kernel and userspace application software. Because of the tight integration of hardware and software the CPU instruction sets, the kernel device trees, the interrupt hierarchy, etc may be different for each implementation and as a result these components may not be interchangeable.

Therefore, MLE has dedicated special attention to ensure that components depending on each other can be associated and loaded/booted together. The concept is to use a so-called Unified Design Image (UDI) which combines an FPGA configuration bitfile, a boot-loader executable, a Linux kernel image, and possibly even userspace application software (see Figure 1). Now, during the system design phase it is important to try out many different system configurations - for functionality, performance and for robustness testing. To facilitate this, MLE has implemented a powerful and flexible multi-boot concept for Programmable System hardware and software which can hold multiple UDIs and offers failsafe system update functionality even for remote access [1].

Figure 1: Unified Design Images for MLE

The configuration management functionality is described in a triplet of MLE Technical Briefs: [2] (this paper) goes into details of the implementation using CPLD devices and how to operate this functionality in general. [3] describes how to use this functionality from a hardware designers point of view. [4] explains how to take advantage of this functionality for efficient firmware and software design.

Most Field Programmable Gate Arrays (FPGAs) incorporate volatile memories for the implementation of logic functions. This so-called configuration has to be loaded after every
There are various ways for configuring FPGAs. During development the most common is using the FPGA vendor’s tools and debug cables to download the configuration data via a Joint Test Action Group (JTAG) connection. After development the configuration data is placed in an on-board non-volatile memory, typically Flash memory. Most modern FPGAs include dedicated configuration logic with interfaces to parallel or serial Flash memory to configure without the help of external circuitry. Some FPGAs even include an automatic fallback to a “golden” configuration, if the first attempt fails for some reason. There are other additional interfaces available, where the FPGA does not control the configuration process itself. Instead this process is controlled by an external microcontroller or a logic circuit. For example, with their Platform Flash [5] Xilinx offers a non-volatile memory chip with special integrated logic capable of actively configuring Xilinx FPGAs.

For maximum flexibility of the configuration process, however, we recommend using a separate device for configuring the FPGA. In the following we will show how a Complex Programmable Logic Device (CPLD) can be used for configuring an FPGA and controlling and monitoring the whole system. As CPLDs incorporate non-volatile configuration memory technology, they are fully functional immediately after applying power to them and thus can operate without first being configured. The MLE 1000 Series Rapid Prototyping System as one of the many instantiations of the MLE “Soft” Hardware Platform uses such CPLD devices to implement a very powerful configuration management.

The boot process of the MLE 1000 Series Rapid Prototyping System as seen in Figure 2 consists of several stages. Initially, after the power comes up, control over the system is transferred to the CPLD. The CPLD selects which out of eight system configuration UDIs shall be used according to user configurable DIP-Switches. These system configuration UDIs are stored in eight partitions in the onboard Flash memory, one partition for each UDI. Each partition holds one UDI’s components: An FPGA design, a bootloader executable and an operating system kernel. The CPLD configures the FPGA with the configuration data found in the selected partition. Once the FPGA is configured the embedded processor within the FPGA design executes the associated initial bootloader executable, which in turn loads the corresponding operating system kernel. After initialization of the MLE Linux kernel the system boots, just like any other workstation, into a userspace environment.

In the following we will have a detailed look at the functionality of the CPLD. For more detailed information on kernel and software development we recommend reading the Technical Brief 20100817.
nical Brief [4]. For a closer look at how to use the CPLD from the MLE Linux userspace and how hardware development can benefit from those features we recommend reading the Technical Brief [3].

Figure 3 shows the connections of the main devices involved in the configuration and system monitoring tasks. The FPGA and CPLD are connected to a JTAG chain, which is controlled via an FTDI USB attachment chip allowing to configure both programmable devices using a standard USB connection. The CPLD furthermore has two communication interfaces labeled COMM, one connected to the FPGA and one to said FTDI USB chip. Additionally, a serial console to the FPGA is also available via USB. Using these interfaces, one can adjust various system parameters, as we will explain later. For the task of configuring the FPGA a configuration data storage such as a NOR-Flash memory plus multiple connections to it are needed: These include control lines (CFG) to supervise the configuration process, DATA lines transporting the actual configuration data from Flash memory to the FPGA as well as address lines (ADDR) from the CPLD to the Flash memory, selecting the data to be sent. The address lines are also available to the FPGA, for it to have full access to the Flash memory once the FPGA is configured. To manually select a new configuration to boot, a DIP switch is connected to the CPLD.

We will now have a look at the configuration process, as it is described in the Xilinx Configuration User Guide [6]. The MLE 1000 Series Rapid Prototyping System uses the so-called SelectMap8 mode, which basically means, that configuration data has to be fed to the FPGA in portions of 8 bits at a time. Therefore the CPLD generates ascending addresses and a clock signal for the Flash memory, whose one byte wide data output is directly attached to the FPGA. The CPLD also has to control the CFG lines. For further details please refer to the mentioned Xilinx user guide. After successful configuration the CPLD stops driving the address pins of the Flash memory and sets these outputs to high impedance, thus allowing the system in the FPGA full control over the Flash memory. Three DIP switches connected to the CPLD allow the selection of up to eight different configuration UDIs for the FPGA. The reference design from Missing Link Electronics utilizes
the configuration 7 with all DIP switches set to “ON” as a designated Linux based rescue system.

This setup is very common and widely utilized in a lot of FPGA based systems. However, with additional communication lines between FPGA and CPLD together with the programmability of the CPLD a much more powerful configuration management can be implemented. For example, the additional signals can be used as a Serial Peripheral Interface (SPI). The CPLD is then attached as a slave to the FPGA operating as a SPI master. Inside the CPLD a small MLE-designed command interpreter executes commands transmitted via SPI. The interpreter operates on 8 bits, where the first 4 bits are interpreted as a command and the other 4 bits are data. The command set is shown in Table 1.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 (0x0)</td>
<td>NOP</td>
<td>No operation / read.</td>
</tr>
<tr>
<td>0011 (0x3)</td>
<td>SDWN</td>
<td>Shutdown command for powering off.</td>
</tr>
<tr>
<td>0100 (0x4)</td>
<td>RIR</td>
<td>Reinitialization request.</td>
</tr>
<tr>
<td>0101 (0x5)</td>
<td>SNP</td>
<td>Set next partition: configures which partition to boot next.</td>
</tr>
<tr>
<td>1000 (0x8)</td>
<td>WDK</td>
<td>Watchdog kick. Sends alive signal to watchdog.</td>
</tr>
<tr>
<td>1001 (0x9)</td>
<td>WDEN</td>
<td>Watchdog enable.</td>
</tr>
<tr>
<td>1010 (0xA)</td>
<td>WDREN</td>
<td>Watchdog rescue enable.</td>
</tr>
</tbody>
</table>

Table 1: Command set of CPLD

Via a “Set next partition” command the FPGA – and therewith any program running under MLE Linux on the MLE 1000 Series Rapid Prototyping System – can select the next configuration partition to boot from. This is shown as (1) in Figure 4. Afterwards the FPGA can issue a “Reinitialization request” command (2), which will cause the CPLD to reset (3) and reconfigure the FPGA from Flash memory using the previously selected configuration. Instead of the “blue” partition selected via DIP switches, the “green” partition, which was previously selected in (1), is used for configuring the system. Additionally, all peripheral chips connected to the FPGA can be reset during (re-)configuration of the FPGA. This way the newly configured FPGA will see a completely “fresh” system afterwards.

Since the FPGA has full access to the Flash memory, any data can be written to this memory. Therefore the MLE “Soft” Hardware Platform connects to the Flash memory as
a Memory Technology Device (MTD) from Linux. By first writing a new configuration into Flash memory and then rebooting, a Programmable System can basically reconfigure itself. Obviously, this can be quite dangerous as it is possible to corrupt configuration data. Therefore, the configuration management comes with various safeguards: The configuration number, when set via `set` command, is automatically disabled after a reboot and the configuration number set by the DIP-switches has a higher priority than any configuration number set via software. This allows the user to trigger a “normal” boot, manually via a push button, using the configuration that was selected via DIP-switches. Sometimes manual user interaction by pressing buttons is not an option. For those cases there is another safeguard:

The CPLD, as shown in Table 1, also offers a programmable hardware/software watchdog. Typically this watchdog is used for issuing a reboot in case of a critical system failure, which may have left the system in a non-operating state. In this case the watchdog detects the lockup and tries to reintialize the system. In case even this configuration is corrupt and may also lead to a non-functioning system a secondary watchdog supersedes the first watchdog. After a configurable number of reboots issued by the first watchdog, the second watchdog forces the configuration logic to use the “golden” configuration in partition 7. This will bring up the known-good MLE rescue system. Once the second watchdog has been triggered, there are only two ways to boot any other configuration than the rescue system: either the system is power-cycled or the user has to deliberately reset the secondary watchdog. A reset of the rescue watchdog can be issued via the “Watchdog rescue enable” command.

All those provisions ensure that the MLE 1000 Series Rapid Prototyping System can be securely deployed in field and upgraded without any risk of having a non-operating, “frozen” system that cannot be accessed any more.

All CPLD functionality available to the FPGA can also be accessed via the FTDI USB chip, which is connected to the CPLD using a second SPI interface. In order to make the CPLD listen for SPI transactions coming from the FTDI, the DIP switch labeled “USB” has to be activated. If USB communication is enabled, the FTDI SPI has a higher priority than the SPI coming from the FPGA. Nevertheless the communication from the FPGA side still works. The main intention behind access to the CPLD via USB is simplifying maintenance. Although for maintenance a working UDI can be downloaded via JTAG, a much faster way for putting the MLE 1000 Series Rapid Prototyping System into maintenance mode is by selecting and booting the rescue system via USB and CPLD. Using the DIP switches to manually select the rescue configuration may not be an option, because in some situations the switches are not easily accessible with the MLE 1000 Series Rapid Prototyping System enclosed.

Another interesting aspect of the USB-SPI communication is the ability to kick the watchdog during system development via USB. So there is no need to have a full implementation of the watchdog chain from the FPGA up to the userspace.

The board layout of the MLE 1000 Series Rapid Prototyping System allows even more tasks to be delegated to the CPLD, especially tasks where response time is critical and tasks that should be independent from the operational status of the FPGA. For example,
the MLE 1000 Series Rapid Prototyping System has four power controllers for its USB ports which can be enabled and disabled via CPLD and can report overcurrent conditions via lines to the CPLD. A similar setup protects the camera ports. As a result the CPLD can be used as a “smart fuse” and immediately power down certain ports whenever an overcurrent condition occurs. With an additional lane between the CPLD and the FPGA (apart from the SPI communication) the CPLD can even report such a condition back to the FPGA, for example as an interrupt.

We have shown that deploying a CPLD in a Programmable System can provide a very powerful configuration management. Additional logic devices such as a CPLD are especially handy when it comes to boot from multiple different configurations during system architecture exploration, for example. With additional communication lines between the FPGA and the CPLD the resulting system can even allow the FPGA to reconfigure itself. We have also shown that the same logic inside the CPLD can also be used via USB, when the SPI interface of the logic is additionally connected to a USB-controller. Furthermore, when it comes to security concerns, the non-volatile logic a CPLD provides can be used both for time critical tasks, like implementing smart fuses, and for tasks that should be completely independent of the FPGA, like watchdog mechanisms.
References


