OPEN AND EASY MICROPROCESSOR DESIGNS USING THE LatticeMico32

A Lattice Semiconductor White Paper

February 2008

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Embedded Microprocessor Trends and Challenges

The last few years have witnessed a growing trend to use embedded microprocessors in FPGA designs. Figure 1 illustrates this trend.

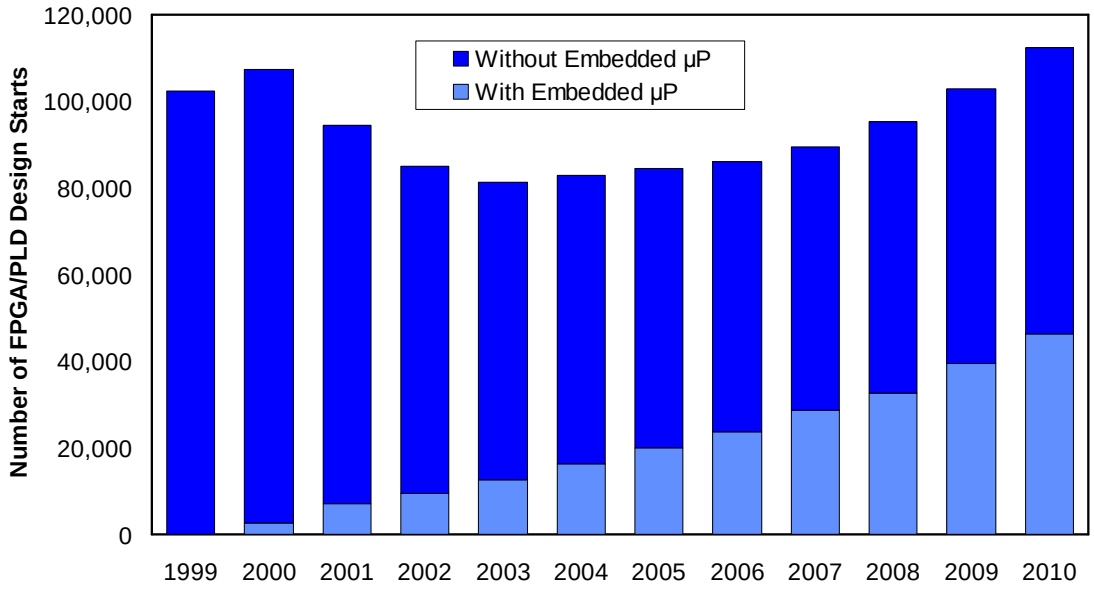

Figure 1 – FPGA Design Starts With Embedded µP
Source: Gartner, August 9, 2005

As illustrated in the graph, this trend is not showing any signs of slowing down. Three important benefits of the embedded microprocessor approach are driving this trend. The first is that a soft processor provides the preferred way to implement control-plane functionality in an FPGA while leaving the datapath functionality to the programmable hardware. Control plane functionality that can be implemented in software rather than hardware allows the designer much greater freedom to make changes. Also, many control plane functions are just too difficult to reasonably implement in hardware. The second benefit is that, with software-based processing, it is possible for the hardware logic to remain stable because functional upgrades can be made through software modification. The third benefit is that FPGA embedded soft processors will not become obsolete. One of the risks that any user faces when designing with an off-the-shelf microprocessor is obsolescence. Having to deal with supply issues or last time buy

orders can become a concern. Also, moving to a new microprocessor entails the task of implementing an incompatible instruction set. Considering the length of some product life cycles, redesigning to support a new microprocessor or simply re-writing and porting older code can be challenging, costly and time consuming.

Another reason for the increased use is that the challenges that embedded FPGA microprocessor designs have historically faced, including cost, speed of design and performance, are now being overcome. As these roadblocks are removed, more and more designers are willing to consider and use an embedded FPGA microprocessor.

The Challenge of Cost

Historically, off-the-shelf microprocessors have been significantly less expensive than their embedded counterparts. Today, with the latest low-cost Field Programmable Gate Arrays (FPGAs) at 90nm or smaller technology, the costs are equivalent. For cost sensitive applications, such as those found in equipment designed for use in consumer products, using an embedded microprocessor is now a viable approach. Low-cost FPGAs are proving to be a cost-effective solution because they consume minimal resources and, if a design already uses an FPGA, the processor can be integrated into the FPGA, saving the cost of a discrete part or a new FPGA.

The Challenge of Design Cycle Time

Designers of system-level chips that include embedded microprocessors have two key time-to-market concerns. First, how quickly can the hardware associated with the microprocessor sub-system be architected and implemented? Second, how long will it take to write, test and debug the code that runs on their microprocessor? Over the last few years, development software for embedded microprocessors has improved greatly in its overall functionality and ease of use. As a result, a design can be up and running in a matter of minutes. Design tools also allow for software and hardware debug, change and upgrade activity, which make an embedded microprocessor even more appealing. Time to market is reduced because it is much quicker and simpler to implement functionality in software than it is to design it in hardware.

The Challenge of Performance

Performance has been historically better with off-the-shelf microprocessors. With improved technology, however, FPGAs have advanced significantly in their feature sets and in overall system speeds. With FPGAs now able to handle greater bandwidth, embedded processors have become attractive choices for many designs.

The LatticeMico32 Microprocessor

The LatticeMico32 is a RISC architecture microprocessor based on Harvard style bus organization. The RISC architecture provides a simpler instruction set and faster performance. The Harvard style busing allows for single cycle instruction execution because separate 32-bit instruction and data buses allow for simultaneous access. The LatticeMico32 provides 32 general-purpose registers and can handle up to 32 external interrupts. Optional Instruction and Data caches are available. Figure 2 highlights these features, as well as other components.

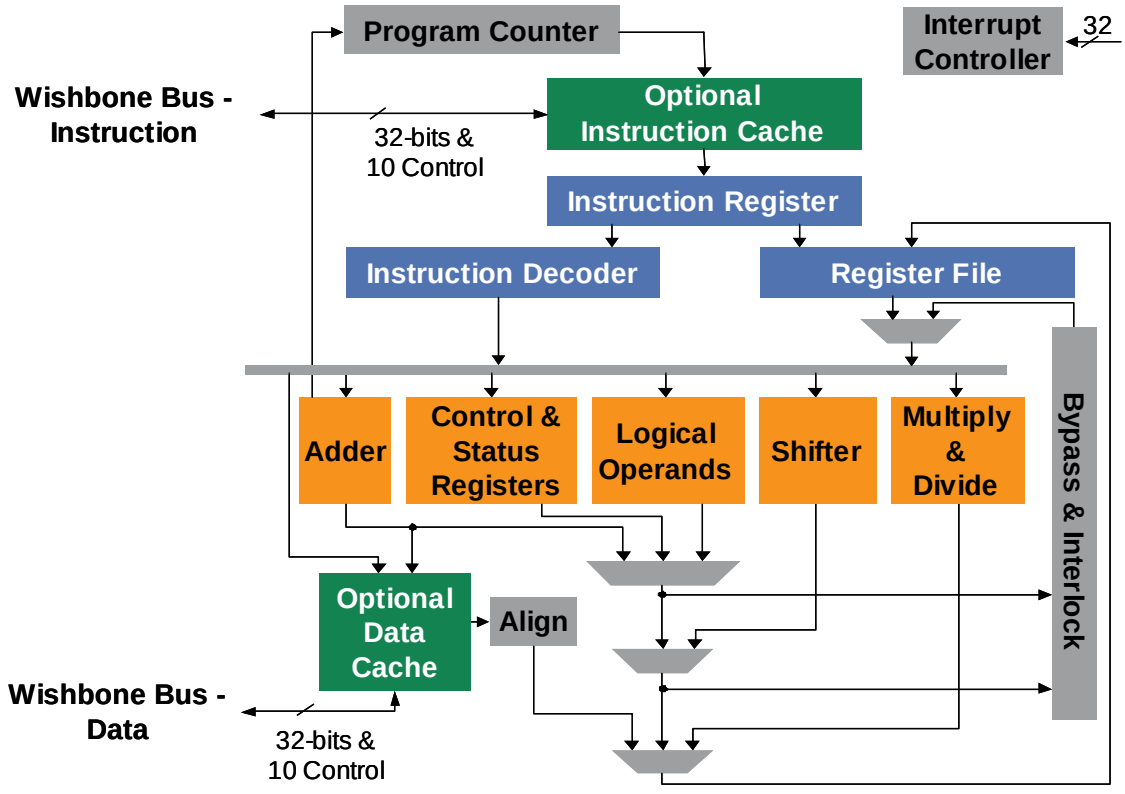

Figure 2 – LatticeMico32 Block Diagram

To accelerate the development of processor systems, several optional peripheral components are available with the LatticeMico32 microprocessor. These components are connected to the processor via a WISHBONE bus interface, which is a royalty-free, public domain specification maintained by OpenCores.org. By using this open source bus interface, users can incorporate their own WISHBONE components into their embedded designs. The peripheral components include:

- Memory controllers
  - Asynchronous SRAM
  - On-Chip Block Memory
  - SPI Flash ROM
  - SDRAM Controller
- Input/Output
  - Direct Memory Access (DMA) Controller
  - General Purpose IO (GPIO)
  - I^2C Master Controller
  - Serial Peripheral Interface (SPI)
  - Universal Asynchronous Receiver Transmitter (UART)
  - TriSpeed Ethernet MAC (10/100/1G)*
- Other
  - 32-bit Timer
  - DDR1/DDR2 Controllers*
  - PCI Target 33MHz*

* Requires free evaluation license or purchased license.

In addition to the peripherals available for the WISHBONE bus, the bus structure generator supports both master side and slave side bus arbitration. Master side bus arbitration provides a simple low cost solution if its restrictions meet system

performance requirements. Given multiple bus masters and multiple slave blocks in a design, it restricts communication to a single bus master at any single point in time. In many designs, slave side arbitration improves performance by allowing two or more bus masters to communicate with separate slave devices simultaneously.

Software Development

The LatticeMico32 System software development tools provide a fast and easy way to implement microprocessor designs, from platform definition to software development and debug. It is based on the Eclipse C/C++ Development Tools (CDT) environment, which is an industry standard open-source development and application framework for building software (Figure 3).

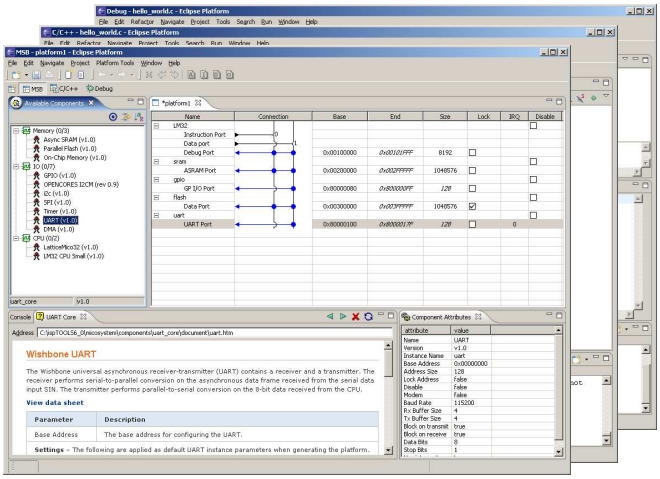

Figure 3 – Easy to Use GUI-based Development Tools

The LatticeMico32 System includes three integrated tools:

- MicoSystem Builder (MSB)

  The MSB generates platform descriptions and the associated Hardware Description Language (HDL) code for hardware implementation. Designers can choose which peripheral components to attach to the microprocessor, as well as specify the connectivity between them.

- C/C++ Software Project Environment (SPE)

  The C/C++ SPE calls a compiler, assembler and linker and enables the development of code targeted to run on platforms created with the MSB. It is through the C/C++ SPE that platforms created in MSB can be referenced.

- Debugger & Reveal Analyzer

  The C/C++ source code debugger provides the ability to debug in assembly and watch the processor registers and memory. Designers can also observe and

control the execution of the code in the physical hardware using the Reveal analyzer.

The tool chain, which is supported on both Windows and Linux, provides development support for the following:

- Small C Lib
- Standard Make
- Program Trace Counter
- SPI Flash Deployment

Operating Systems

The LatticeMico32 has support for the following Operating Systems:

- uClinux and U-Boot from Theobroma Systems in Austria
- µC/OS-II RTOS from Micriμm
- µITRON RTOS per TOPPERS/JSP

Resource Utilization and Performance

The LatticeMico32 provides high performance as well as minimal resource utilization. For designers who are concerned about resources, the Basic configuration uses no instruction or data cache, a single cycle shifter and no multiplier. For those concerned more with performance, the Full configuration uses 8k bytes of instruction cache, 8k bytes of data cache, a 3-cycle shifter and a multiplier. For users who need a compromise, the Standard configuration is similar to the Full configuration, but without the 8k bytes of data cache. Table 1 shows resource utilization and performance for the LatticeECP2M FPGAs.

Configuration	Family	LUTs	EBRs	MHz	$ of Logic
Basic	LatticeECP2M	1689	0	114	~$0.80
Standard	LatticeECP2M	2037	5	138	~$1.00
Full	LatticeECP2M	2268	10	133	~$1.00

Table 1 – LatticeMico32 Resource Utilization and Performance

The Basic configuration uses 1689 Look Up Tables (LUTs), while the Full configuration operates at a maximum clock frequency of 133 MHz. In terms of the implementation cost, with the LatticeECP2M in high volumes the price of the silicon is as low as 50 cents for a thousand LUTs. With the LatticecMico32 consuming under 1700 LUTs in the basic configuration, that translates to an implementation cost of approximately $0.80. With these configurations, Lattice is able to address the challenges of both cost-sensitive and higher performance designs.

Open Source Approach

Open source is gaining popularity in a variety of software areas and is already well established for desktop/server software. The benefits of using open source IP include more visibility, greater flexibility and improved portability.

Open source provides visibility into the details of the microprocessor. By having access to the source code, a designer has a complete understanding of the details of the core. Additionally, open source provides greater flexibility in that the IP is available and open for everyone, so designers can review it and make improvements to the IP. The entire user community helps to identify problem areas and to develop solutions. This means that not only are modifications permitted, they are encouraged. This community interaction results in an open source IP core that tends to be more robust and reliable than traditional, proprietary IP. Finally, open source provides improved portability. A user enjoys architecture independence because an open source IP core can be used in any FPGA, or even migrated to an ASIC for higher volume, mature designs. Architecture independence is valuable because it provides insurance in case last minute

changes need to be made in the silicon. Finally, the most commonly associated benefit of open source IP is that it is free of charge.

Standard Open Source Licensing Restrictions

Today, many open source licenses exist. Three of the most common ones are:

- GNU – GPL (GNU Is Not Unix – General Public License)
- GNU – LGPL (Lesser General Public License)
- BSD (Berkeley Source Development)

Although open source licensing is available, the hardware design community has not rushed to use open source IP. As compelling as the benefits are, the reality is that standard open source IP licenses, such as the ones mentioned above, have some restrictions that are major stumbling blocks for hardware.

For example, some of these licenses impose certain responsibilities for the distribution of derived works. In this case, FPGAs that utilize the source code are considered derived works. The entire derived work must therefore be made available in source form. As a result, any proprietary logic becomes public, which is clearly not desirable. Another issue is that open source licenses typically require the user to provide a copy of the license with the derived work. In many cases, having to send out a license with each end product can be at best inconvenient and at worst impractical. In effect, standard open source licensing has restrictions that may limit, or forfeit, the user's ability to maintain designs as proprietary.

Lattice Eliminates Open Source Licensing Restrictions

For those users who are apprehensive about standard open source cores, Lattice Semiconductor provides a unique open IP core licensing agreement. The agreement provides all the benefits of standard open source, while also addressing the drawbacks of standard open source licensing as it applies to hardware design. The first step is to break down the core into modules. Each module must be available free of charge, but the rest of a user's design can be licensed under any user chosen license agreement. Furthermore, the agreement allows programmable logic devices to be distributed

containing the design without the need to distribute the license agreement or source. In short, the Lattice open IP core licensing agreement allows users to mix proprietary designs with the open source core and distribute the designs in bitstream or FPGA format without an accompanying copy of the license. In addition, the Lattice license specifically grants the right to port the design to an ASIC.

With the LatticeMico32, the licensing structure is really twofold. The Lattice open IP core license agreement will be used with the HDL code that is generated by the MSB tool. Most of the graphical user interfaces will be licensed under an Eclipse license, while, for the internal workings of the software, such as the compiler, assembler, linker and debugger, the licensing scheme will follow the GNU-GPL.

Summary

The LatticeMico32 is a complete embedded microprocessor design solution. When used with Lattice's FPGAs, designers have a cost effective design alternative that takes less than $1 of logic to implement. The LatticeMico32 development tools make it easy to implement a microprocessor and the attached peripheral components in an FPGA. The ease of use ensures that design times are minimal, which results in an even quicker time to market. Through various flexible configurations, the LatticeMico32 can be tailored for customers with performance requirements. Lattice is a pioneer in the open source microprocessor arena. By providing the generated HDL under an open source license agreement and the software development tools under their respective open source licenses, such as Eclipse and GNU-GPL, Lattice allows users complete control of their designs. The open source nature provides designers with the visibility, flexibility and portability that are required.