Introduction to Programmable Systems on a Chip

Bob Zeidman
President
Zeidman Technologies
bob@zeidman.biz

Course #EPD-504, Easy Paths to Silicon Design Seminar, San Francisco 2005

TABLE OF CONTENTS

1 INTRODUCTION

This paper examines various programmable systems on a chip (SOCs) and their underlying architectures and technologies. Specifically oriented toward hardware engineers and system engineers, the paper discusses tradeoffs between the different programmable SOC devices based on their architectures and technologies. An overview is given of the different programmable SOCs on the market and the different vendors that offer them.

1.1 Intended Audience

This paper is valuable to hardware engineers and system engineers selecting a programmable SOC for use in a system to be designed soon.

1.2 Results

By the end of the paper, the reader will understand in general terms the technologies and architectures of various programmable SOCs, and the families of devices currently available from the various vendors.

1.3 Prerequisites

The reader should be knowledgeable about logic design and microprocessor architecture.

2 FIELD PROGRAMMABLE GATE ARRAYS (FPGAS)

We should begin with a short refresher on FPGAs. Each FPGA vendor has its own FPGA architecture, but in general terms they are all a variation of that shown in Figure 1. The architecture consists of configurable logic blocks, configurable I/O blocks, and programmable interconnect. Also there is clock circuitry for driving the clock signals to each logic block, and additional logic resources such as ALUs, memory, and decoders may be available. The two basic types of programmable elements for an FPGA are static RAM and anti-fuses.

2.1 Configurable Logic Blocks

Configurable Logic Blocks contain the logic for the FPGA. Typically these CLBs contain enough logic to create a small state machine. A CLB usually contains RAM for creating arbitrary combinatorial logic functions, also known as lookup tables (LUTs). It also contains flip-flops for clocked storage elements, and multiplexers in order to route the logic within the block and to and from external resources. The muxes also allow polarity selection for outputs and reset and clear input selection for flip-flops.

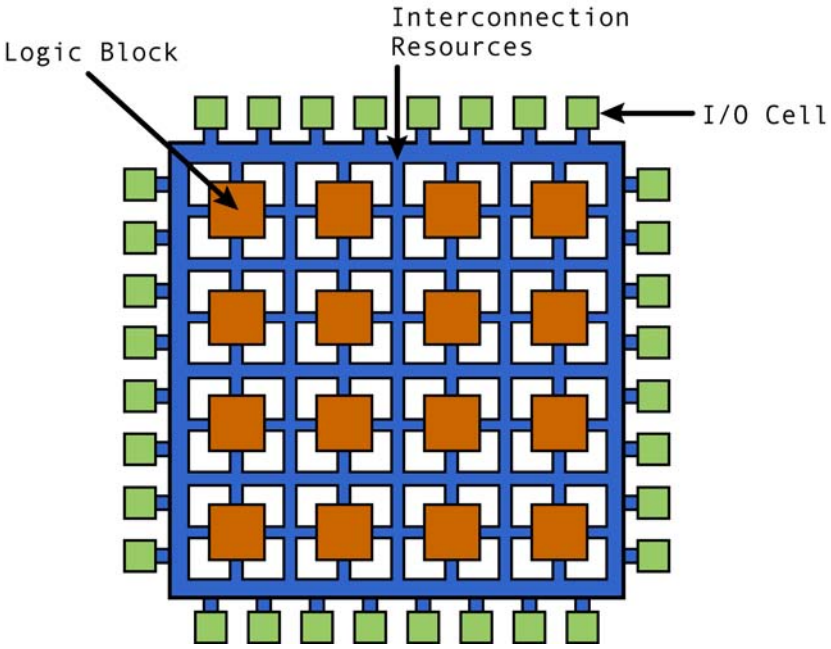

Figure 1. FPGA Architecture

2.2 Configurable I/O Blocks

A Configurable I/O Block is used to bring signals onto the chip and send them back off again. It typically consists of an input buffer, an output buffer, and two flip-flops. One flip-flop is used to clock the output signal to shorten the clock-to-output delay for signals going off chip. The other flip-flop is used to register the chip input, decreasing the device hold time requirement.

2.3 Programmable Interconnect

The interconnect in an FPGA consists of a hierarchy of interconnect resources. There are long lines that can be used to connect critical CLBs or as buses within the chip. There are short lines that are used to connect local CLBs. There are often one or several switch matrices to connect these long and short lines together. Programmable switches inside the chip allow the connection of CLBs to interconnect lines and interconnect lines to each other and to the switch matrix, enabling routing of your design.

2.4 Clock Circuitry

Special I/O blocks with special high drive clock buffers, known as clock drivers, are distributed around the chip. These buffers connect to clock input pads and drive the clock signals from the outside world onto fast, low-skew global clock lines within the FPGA.

2.5 Programming Technology

There are basically two competing methods of programming FPGAs. The first, SRAM programming, involves small static RAM bits for each programming element. The other

method involves anti-fuses, which consist of microscopic structures that, unlike a regular fuse, normally make no connection. A certain amount of current during programming of the device causes the two sides of the anti-fuse to connect. Some FPGAs use a third technology, utilizing flash RAM bits as programming elements.

	SRAM	Antifuse	Flash
Theoretical Speed	Slow	Fast	Slow
Actual Speed	Fast	Fast	Slow
Density	Larger structures Less dense	Smaller structures More dense	In-between
Volatility	Volatile	Non-volatile	Non-volatile
Reprogrammable	Yes	No	Yes
Security	Low (bitstream accessible)	High (bitstream not accessible)	High (bitstream not accessible)
Power Consumption	High	Low	Low

Table 1. SRAM vs. Anti-fuse FPGA Architecture

2.5.1 SRAM Programming

An advantage of SRAM based FPGAs is that they use a standard fabrication process, meaning it is easier to improve the technology to make them faster and lower power. Since the SRAMs are reprogrammable, these FPGAs can be reprogrammed any number of times, even while they are in the system and operating. SRAM based devices can easily use the internal SRAMs as small memories in the design. The disadvantages are that they are volatile, which means they must be reprogrammed each time the system is powered up, and a power glitch could potentially change their state. Also, SRAM-based devices have large routing delays.

2.5.2 Anti-fuse Programming

An advantage of anti-fuse based FPGAs are that they are non-volatile and the delays due to routing are very small, so they tend to be faster in theory. In practice, because SRAM technology is a mature, well-understood process, the speed of SRAM-based FPGAs and anti-fuse FPGAs are about the same. Antifuse based FPGAs tend to require lower power and they are better for keeping your design information secure because they do not require an external device to program them upon power-up. The disadvantages are that they require a complex fabrication process, they require an external programmer to program them, and once they are programmed, they cannot be changed.

2.5.3 Flash Programming

Flash programming offers the potential for combining the advantages of both SRAM and anti-fuse technologies. Flash based devices have the advantage of SRAM based devices in that they use a standard semiconductor process. They are nonvolatile and thus use less power and the intellectual property of the design is secure. Flash based devices can still be reprogrammed multiple times, even while operating in the system. A drawback is that flash devices are slower than either SRAM or anti-fuse devices

3 PROGRAMMABLE SYSTEMS ON A CHIP

The definition of a programmable system on a chip (SOC) varies somewhat, depending on the use. One definition is an FPGA that is so large – contains so many logic gates – that it takes the place of an entire system that would have taken an entire PCB full of chips only a few years ago. For the purposes of this paper, we will use the definition that a programmable SOC must include a microprocessor. Thus any FPGA is a programmable SOC if it includes a microprocessor so that it is both hardware programmable and software programmable. Also, an FPGA is a programmable SOC if it includes enough gates to allow the inclusion of a microprocessor design and provides support for such a design. These two definitions encompass the two different types of processors in a programmable SOC, a hard processor or a soft processor, as described in the following sections. There is also a third type of programmable SOC, one in which a chip contains various blocks of a microprocessor and peripherals that can be programmatically connected or disconnected. This third kind of programmable SOC differs from an FPGA-based programmable SOC in that the programmability is at a very high level of functionality and does not allow as much flexibility for low-level, user-defined functions to be designed on chip.

3.1 FPGA with a Hard Processor

A programmable SOC with a hard processor (or hard processor core) is an FPGA that has circuitry for a microprocessor embedded in the chip, surrounded by programmable logic. As shown in Figure 2, the processor in the lower right corner is fixed circuitry that takes the place of some programmable logic in an ordinary FPGA.

An advantage of a hard processor is that the processor circuitry has been optimized for timing and power consumption and can be characterized fairly precisely by the vendor. A hard processor specified to run at 100 MHz will run at that speed, though there is no guarantee that the rest of your design can keep up. A hard processor specified to consume a specific amount of power, will meet that criteria. The circuitry is fixed and the vendor has characterized the circuitry independently of the rest of the design.

Another advantage of a hard processor is that the architecture is usually a standard one that has much support in terms of existing code, libraries, and tools such as compilers, debuggers, and operating systems available from multiple vendors. The processor may be a high end one such as an IBM PowerPC, MIPS, or ARM processor that is capable of high speed processing of very complex algorithms. The processor may be a low end processor such as an

Intel 8051 that is low-power, takes up a small area, and is good at simple hardware-related control functions. Often each programmable SOC vendor offers only one hard processor and your specific system needs may limit the vendors and SOC families that you can use in your system.

Yet another advantage of a hard processor is that the design is compact, rather than spread among CLBs, so it uses less space on the chip die. This translates to lower costs since as the customer you will be paying based on the size of your chip. But there is a tradeoff with regard to die area because you have all the processor functionality whether you need it or not. This means that if you use a hard processor with a memory manager, but your code does not need a memory manager, you are stuck with the memory manager anyway. The unused memory manager raises the cost of the FPGA and the unused logic increases the power consumption.

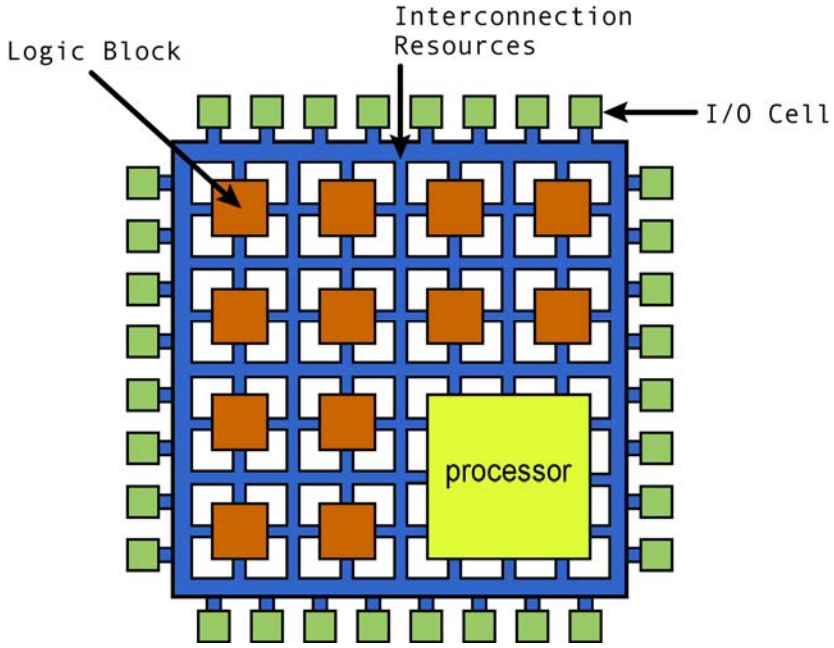

Figure 2. Programmable SOC with Hard Processor

3.2 FPGA with a Soft Processor

Any processor with enough logic resources can be an SOC by including a soft processor. A soft processor is a logic description of a processor that can be included with the rest of your design, compiled into a gate level description, and placed and routed onto the FPGA. Typically, a soft processor is described in Verilog or VHDL and then combined with the rest of your Verilog or VHDL design.

An advantage of a soft processor is that it can be configured in a way that is optimal for your system. Unnecessary functionality can be removed in many cases, though significant changes can make the software compiler and other software tools unusable, so you must be careful and consult with the FPGA vendor before making any changes. For example, if you do not need a memory manager in your system, it can simply be removed, as long as it is done

without disturbing the other functionality of the processor. However, many off-the-shelf real-time operating systems (RTOSes) require a memory manager.

A disadvantage of a soft processor is that it will be specific to the FPGA vendor. There are some third-party soft processors that run on FPGAs from multiple vendors and that are well supported with software tools. It is unlikely, though, that any major high performance processor vendor like Intel or IBM will offer a soft version of its processor because that would be giving away their most valued intellectual property. For this reason, there is generally less support and fewer software tools available for soft processors.

A comparison of advantages and disadvantages of soft and hard processors is shown in Table 2. Although it looks like hard processors win hands-down, it is important to make a fair comparison. Because a soft processor can be configured to remove unnecessary functionality, a soft processor can be reconfigured to beat out a hard processor in each category.

	Soft Processor	Hard Processor
Configurability	**High**	**Low**
Power Consumption	High	Low
Predictability	Low	High
Software Tools	Fewer	More
Speed	Low	High

Table 2. Soft Processors vs. Hard Processors

3.3 Vendors and Processors for FPGAs

A current list of FPGA vendors and the hard and soft processors they offer is shown in Table 3. A list of third party vendors that provide soft processors is shown in Table 4.

FPGA Vendor	Hard Processor	Soft Processor
Actel	none	third-party only
Altera	ARM	NIOS, NIOS II
Atmel	AVR	third-party only
Lattice	none	third-party only
Quicklogic	MIPS	third-party only
Xilinx	IBM PowerPC	MicroBlaze, PicoBlaze

Table 3. FPGA Vendors and Processors

Many programmable SOCs also include hard cores that provide additional functionality. The advantage is that this functionality is compact, optimized, and predictable.

The disadvantage is that the hard cores use die space and consume power, even if your system does not require the functionality provided by these hard cores. Table 5 contains examples of hard cores offered by FPGA vendors on their programmable SOCs.

Third Party Vendor	Soft Processor
ARC	ARC 600, 700, A4
ARM	ARM 7, 9E, 10E, 11
	SecurCore
	MPCore
Cast	AMD 2901, 2910A, 29116A
	Dallas 80530
	Infineon R80515
	Intel C8051, C8051, C80186tx
	Motorola C6805, C6811, C68000
	Ocean Logic AES Cryptoprocessor
	PIC C165X, C1655X
	Texas Instruments C32025 DSP
	X_DES Cryptoprocessor
	Zilog CZ80CPU, CZ80PSC
Digital Core Design	DR8051
	DP80390
	DFPIC1655X
	DF6805, DF6808, DF6811
MIPS	MIPS32, MIPS64
	Pro Series

Table 4. Third Party Vendors and Processors

Example Hard Cores
Bus Interfaces
Cache Memory and Controller
Communication Cores
Counters
Digital Signal Processing Logic
Ethernet MAC
Logic Analyzer
Math Cores
Memory Controller
Peripheral Cores
SRAM Blocks
Timers
UART

Table 5. Other Hard Cores for SOCs

3.4 Configurable Processor Chips

Configurable processor chips come in many different flavors. Unlike FPGA SOCs, there are no common architectures for these chips. Each vendor has a very different architecture and, in fact, a very different design philosophy for its SOCs, so we will examine several vendors individually. Because technology is continually changing, this is not meant to be an exhaustive list of vendors, but only a sampling of the major vendors at the time this article was written.

3.4.1 Cypress Semiconductor

The Cypress PSoC™ family of chips has an architecture that includes an 8-bit processor and various peripherals that can be connected, or left unconnected, using SRAM-based programming elements. One unique aspect of these chips is that in addition to digital peripherals, they include analog peripherals. A typical PSoC architecture is shown in Figure 3. The analog units of a PSoC that can be configured in the chip include comparators and analog-to-digital converters (ADCs). The digital units include timers, counters, pulse width modulators (PWMs), Cyclic Redundancy Check (CRC) modules, a full-duplex UART, and a Serial Peripheral Interface (SPI™) module. The PSoC also includes RAM and Flash memory.

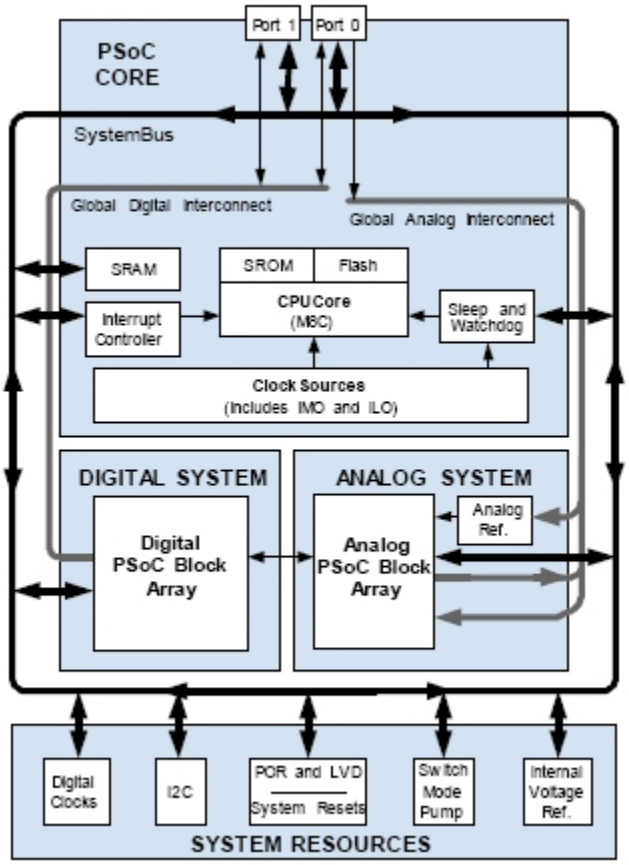

Figure 3. Cypress PSoC™ Block Diagram

3.4.2 Quicksilver Technology

Quicksilver Technology has a programmable SOC that consists of many tiny processors that can each be programmed to execute low-level logic functions. The architecture of one of these devices, called an Advanced Computing Machine (ACM) is shown in Figure 4. At the bottom of the figure are the elements that make up the most unique aspects of the SOC. These are the processing units, consisting of three different types – Adaptive Execution Nodes (AXNs), Domain Bit Manipulation Nodes (DBNs), and Programmable Scalar Nodes (PSNs). These nodes are connected via Matrix Interconnect Networks (MINs). The PSNs are 32-bit RISC processors. The AXNs can implement ALUs, MACs, multipliers, and address generators as well as other functions. The DBNs perform bit manipulation, useful for encryption, decryption, and error checking functions.

At the top of Figure 4 are the sections of the device that communicate with the outside world. These include network I/O devices, other system I/O devices, a JTAG interface for testing purposes, and a memory controller. The ACM architecture also includes a system controller that distributes tasks to each processor and allocates time for them. This system controller acts as a real-time operating system (RTOS) in hardware.

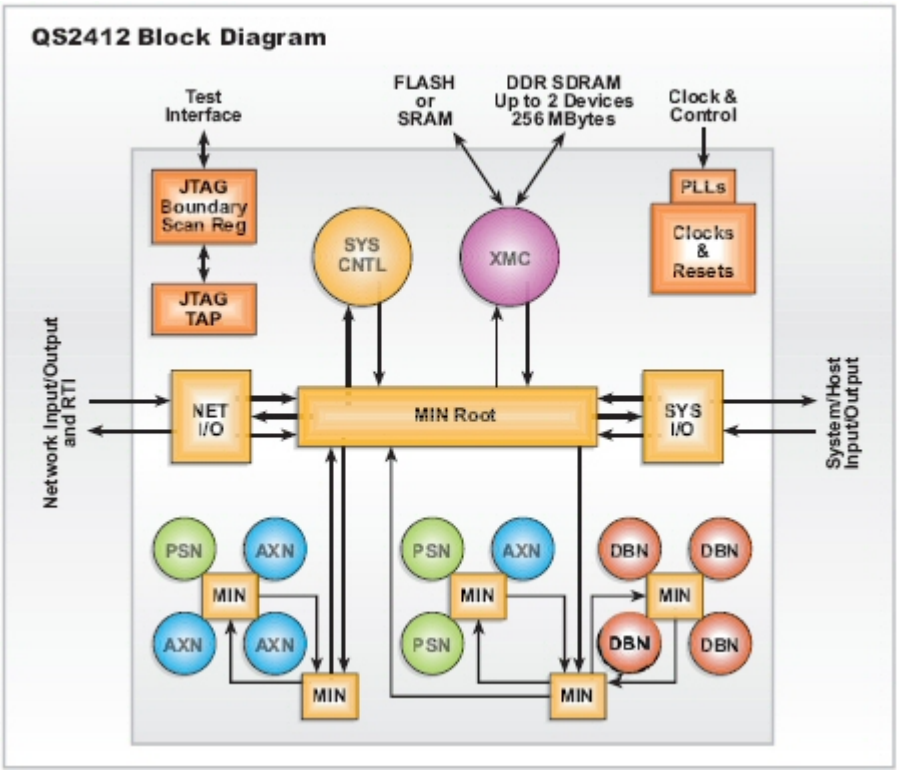

Figure 4. Quicksilver Adaptive Computing Machine

The ACM architecture is reminiscent of the Multiple Instruction Multiple Data (MIMD) parallel processors of the 1980s. Those machines did not succeed because of the difficulty compiling standard programming languages in a way that would make use of the processors efficiently. New programming languages were devised for these architectures, but

they were too difficult to use to write programs for all but very specialized applications. The MIMD approach is becoming successful in grid computing where large chunks of programs can be allocated to entire processors on a network. Quicksilver believes that the approach will also be successful for their very fine-grained devices that are controlling hardware directly rather than running high level applications.

3.4.3 Tensilica

The Tensilica Xtensa processor architecture really falls somewhere between a microprocessor and a programmable SOC. It is not truly an SOC by itself because it is only processor with some limited peripherals. Yet it is not just a microprocessor because it is programmable so that the processor functionality can be optimized for only that functionality needed for your system. The concept is that a user defines the execution datapaths, I/O ports, and registers that are needed for the particular application, using a language very much like Verilog. The user also defines extensions to the instruction set for the processor, which has a Very Long Instruction Word (VLIW) architecture. Alternately, the user can write C code and use Tensilica's XPRES Compiler tool to analyze the C code and suggest a processor implementation and instruction set extensions.

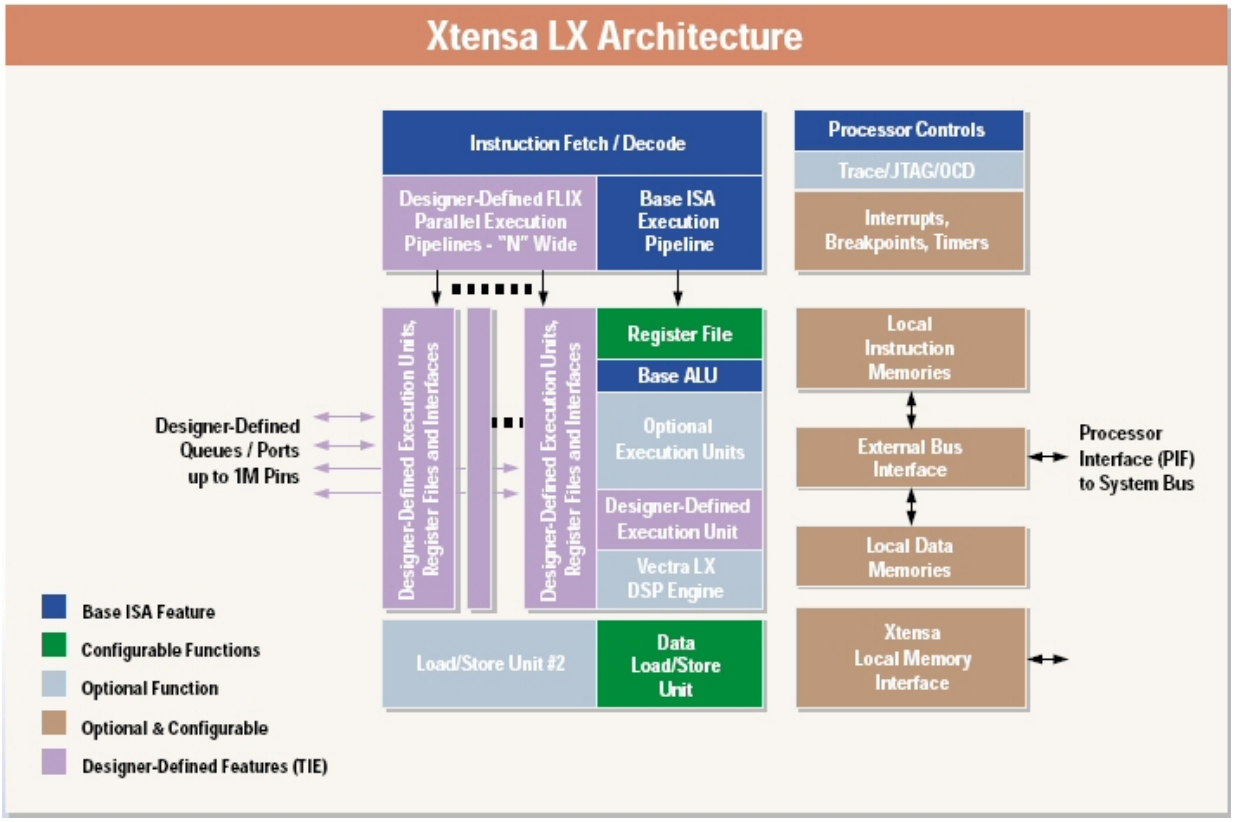

Figure 5. Tensilica XTensa LX Architecture

Figure 5 shows the Tensilica Xtensa LX architecture. Note that the diagram is color coded to illustrate which parts of the chip are fixed and which are configurable.

4 CONCLUSION

There are a variety of options when considering programmable systems on a chip. The traditional FPGA vendors offer two kinds of programmable SOCs, those incorporating hard processor cores and those supplying soft processor cores. Also, third party vendors offer soft processor cores that can be used with one or more FPGAs. There are certain tradeoffs between the hard and soft processors that must be considered, including speed, power consumption, configurability, cost, predictability, and software support.

There are also programmable SOCs based on configurable processors, available from several vendors. Each of these devices has a unique architecture and philosophy as well as different sets of tools to create and optimize the devices.

This paper has presented information about all of the types of programmable SOCs currently available and has given the reader enough initial knowledge to ask the right questions and make a more informed decision about which device should be used in a system being designed.

5 ABOUT THE AUTHOR

Bob Zeidman is the president of Zeidman Technologies (www.zeidman.biz), a vendor of tools for embedded systems hardware and software development. Bob has designed ASICs, FPGAs, and PC boards for RISC-based parallel processor systems, laser printers, network switches and routers, and other systems for clients including Apple Computer, Cisco Systems, Intel, and Texas Instruments. He has written papers on hardware and software design methods, and has taught courses at conferences throughout the world. He is the author of the textbooks *Verilog Designer's Library*, *Introduction to Verilog*, and *Designing with FPGAs and CPLDs*. Bob was the recipient of the 1994 Wyle/EE Times American By Design Award and other engineering, writing, and scholastic awards. Bob holds a patent in software synthesis and he has an MSEE from Stanford University, and a BSEE and a BA in physics from Cornell University.
