Electronic Engineering
TIMES

Issue 432 The Industry Newspaper For Engineers And Technical Management Monday, May 4, 1987

The Acid Test

Using an EPLD vendor's application, the same circuit was built with a Logic Cell Array. The comparison provides real insight to the differences between these two technologies.

By Steven K. Knapp
Field Applications Engineer
Xilinx Inc.

As with a processor, a PLD's architecture determines its functional logic density and its performance.

Among processors, general-purpose microprocessors and application-oriented devices like digital signal processors share many similar internal logic structures and functions. DSPs have a more rigid architecture, designed for particular applications, while general-purpose microprocessors address a much wider set of applications. The same situation exists in the world of PLDs.

Until recently, the sum-of-products (SOP, also known as AND-OR) structure was the only one available for PLDs. Its simple, fixed architecture fits a variety of low-density, high-speed applications. These include fast, wide logic functions found in decoders, multiplexers and counters.

Some recently introduced high-density PLDs are based on extensions of the conventional AND-OR architecture. Examples include the MMI 64R32 MegaPAL and the Altera EP1800 Erasable Programmable Logic Device (EPLD).

Like a DSP processor, the sum-of-products architecture of these devices efficiently meets certain needs. However, this architecture is not suitable for higher-density, register-intensive random logic structures commonly found in logic designs.

Logic Cell Array

The Programmable Gate Array differs vastly from the conventional sum-of-products architecture. Its flexible, register- and I/O-rich, array-style architecture addresses a wider set of common logic design problems.

Its Logic Cell Array architecture consists of three basic programmable elements: input/output blocks, configurable logic blocks and programmable interconnects.

Each I/O block can be individually configured as a direct or registered input, a direct or three-state output or as a bidirectional I/O.

The configurable logic blocks contain combinatorial logic plus storage elements. The combinatorial logic within each block implements any possible single function of four variables, or any two functions of up to three variables. The storage element is configurable as an edge-triggered flip-flop, or as a level-sensitive latch, both with asynchronous SET and RESET inputs. The programmable interconnect allows each of the storage elements to be clocked either synchronously or asynchronously.

Three levels of programmable interconnect resources provide the interconnection between blocks:

- Direct interconnect allows fast connections between adjacent blocks.
- General-purpose signals travel through an array of switching matrices that yield an efficient means of connecting scattered random logic.
- Long metal lines that traverse the chip distribute clock or other signals with high fan-out or requirements for minimum skew.

The best way to illustrate the difference between the Logic Cell Array and the more conventional AND-OR architectures is through a design example.

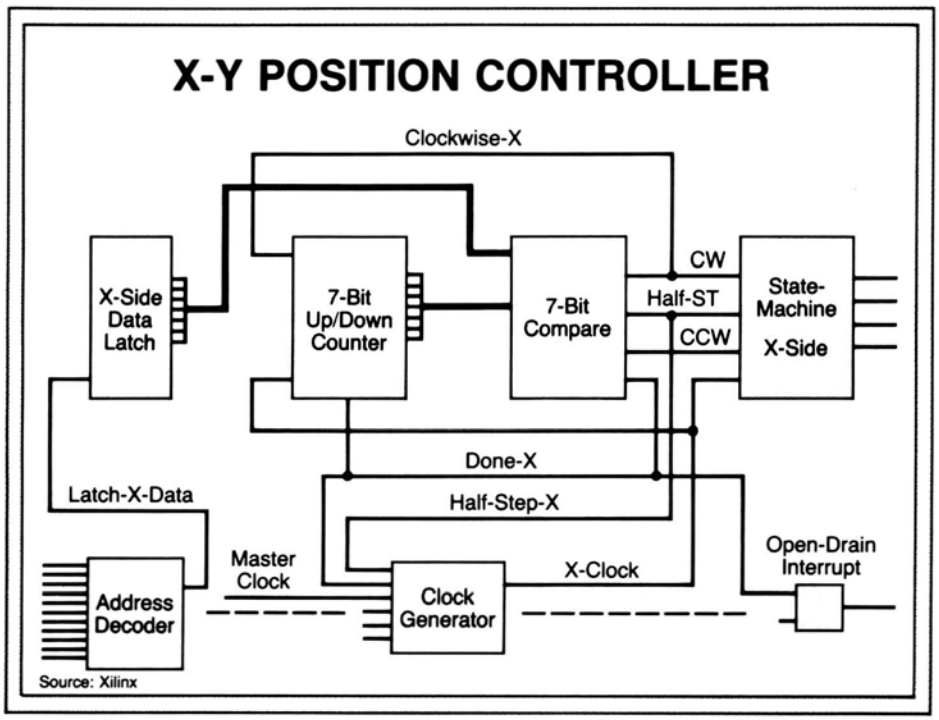
An X-Y position controller design was originally developed by Altera as an application example for its EP1800. The design is fairly simple, and illustrates the capability of programmable logic to address the problems faced by logic designers. The example can be used to demonstrate the capabilities of both the Altera EPLD, and the Xilinx Logic Cell Array.

X-Y position controllers are employed in a variety of design applications to control motors for printers, plotters, robotics, and numerical controllers. The controller compares the desired location loaded from an external processor with the present motor position stored within the PLD. Based on the results of the comparison, the controller drives two four-phase stepper motors (an X- and a Y-position motor) to its desired position.

In this design, X- and Y-position data is loaded into the device from an external microprocessor. The 1,800-gate EPLD has no input storage elements, so the data must be held valid by some external device until both motors reach their final position. Since the Logic Cell Array has input flip-flops which hold the X- and the Y-position data, it offers a simpler interface to the processor.

The desired position data is compared against the present position data. The result of the comparison drives the 7-bit up/down counters which, in turn, drive the state-machine motor control. The stepper motors used in the application have 7.5 degree full-step increments with optional 3.75 degree half-step increments available. A 7-bit binary up-down counter is required to keep track of the 96 motor steps required to rotate each motor a full 360 degrees.

In the EPLD designs, seven macrocells are used to implement the binary up/down counter for each half of the



The schematic above illustrates the X-axis half of the position controller used as the design example. The other half is identical to this circuit, and drives the Y axis instead of the X.

Four EPLD macros are needed for each of the state-machines in the circuit, while seven Logic Cell Array logic blocks are required for the same function.

position controller. Because of the low-frequency clock (500 kHz), this same counter requires only seven Logic Cell Array logic blocks. If this were a high-speed counter, a few additional blocks would perform some level of carry-look-ahead.

If the desired position is greater than the present position, the state-machine-based motor control drives the stepper motor clockwise. If the position is less, the controller drives the motor counter-clockwise.

Four EPLD macrocells are required to implement the state-machine for each half of the position controller, while seven Logic Cell Array logic blocks are required to implement the same function.

To signal the processor that the entire operation is complete, an open drain interrupt signal is generated when both the X- and the Y-position motor have reached their final location. This also signals the processor that further motor action may be taken. A master RESET signal resets the present position to zero.

Resource Requirements

Based on the requirements of the design, the X-Y position controller example requires 47 of the 48 macrocells in the 1,800-gate EPLD and all of its 64 I/O pins. A number of the I/O pins are used simply because a register is hard-connected to the pin. The two 7-bit counters, for example, use 14 I/O pins because they cannot be buried. The X-Y controller design uses 98 percent of the macrocells and 100 percent of its I/O pins.

The same design (plus the input data registers) fits in 49 configurable logic blocks and 27 input/output blocks in a Logic Cell Array (the MASTER RESET input uses the Logic Cell Array's dedicated master reset pin). Fewer I/O pins are required, since the 7-bit up/down counters and the state-machines were buried in the Logic Cell Array design.

The X-Y controller design would occupy 77 percent of the 1,200-gate XC2064's logic blocks and 47 percent of its I/O pins. Or if the 1,800-gate

Logic Cell Array were used, the X-Y controller would occupy just 49 percent of the logic blocks and 36 percent of its I/O. The remaining logic and I/O blocks can be used to build chip select logic to make a clean interface to the microprocessor.

The architectural differences between the two technologies allow a design which requires an entire 1,800-gate EPLD to fit in just a portion of a 1,200-gate Logic Cell Array. The EPLD can perform any one of the required tasks quite well. The sum-of-products architecture allows EPLDs to implement fast counters, decoders, and multiplexers in a single pass. Combining these elements into a larger, more complex system with additional passes through the array however, hurts both performance and density.

The flexible Logic Cell Array architecture allows entire complex systems to be implemented efficiently and quickly. Given the same 1,800 gates of logic, an 1,800-gate Logic Cell Array could implement an X-Y-Z position controller in the same density used to implement an X-Y controller in an EPLD.

Conclusion

The benefits of programmable logic devices within any design are generally undeniable. Both devices shown in the design example (the EP1800 EPLD, and the XC2064 Logic Cell Array) replace many SSI/MSI components.

However, not all PLDs are created equal. A designer should contemplate his system needs and goals when deciding which PLD to use. Like making the correct choice of a system processor, the correct PLD choice will have an impact on the final system performance.

Some guidelines and cautionary notes are necessary for designers new to the PLD approach. These are:

- Be wary of PLD gates counts. As mentioned before, not all PLDs are created equal. Equivalent gate counting is used by many manufacturers to compare their devices against a competitor's. Two devices with approximately equal gate counts will differ on functional density within a system because of differences in architecture. A better way to determine density is to count the types and amounts of various resources on the chip, and compare those with the needs of your application.

- Understand which PLD architecture best suits your application. The sum-of-products (AND-OR) architecture is well suited to applications that require ANDing of a large number of inputs, like those found in address decoders and some counters. The interconnect structure of an AND-OR PLD makes single-pass logic functions quick and efficient. However, the complex random logic requirements of higher-density logic designs stretches the limits of conventional AND-OR devices because of the feedback requirements.

- Determine your I/O and register requirements. In most AND-OR PLDs, outputs are hard-connectors to the registers within the device. Therefore, each register either uses or wastes an I/O pin and vice versa. In the Logic Cell Array, I/O and registers are separated by programmable interconnects. Therefore, entire register resources like counters and shift registers can be buried without using or wasting I/O pins.

Major enhancements in high-density PLDs will stem from architectural innovations rather than process-related developments. Regardless of which production process is used, all PLD manufacturers are searching for the optimal mix of high performance, high density, and production cost to best meet designer's logic needs. ■

COMPARISON OF HIGH-DENSITY PLDS

	MegaPAL	MegaPAL	EPLD	EPLD	LCA	LCA
Claimed Gate Density	1,500	5,000	1,200	1,800	1,200	1,800
Architecture	AND-OR	AND-OR	AND-OR	AND-OR	Array	Array
Developed By	MMI	MMI	Altera	Altera	Xilinx	Xilinx
Inputs (max.)	32	64	36	64	58	74
Outputs (max.)	16	32	24	48	58	74
Storage Elements	16	32	28	48	64	100
Elements	0	0	12	0	58	74
Speed For 16-Input AND (ns, On-through-Off, fastest)	40	50	50	50	45	45
Register-To-Register Clock Frequency (Max. MHz)	16	16	20	23.2	58.8	58.8
Quiescent Power (W)	1.4	3.25	0.015	0.001	0.025	0.025
Packages	40 DIP 44 PLCC	84 PLCC 88 PGA	40 DIP 44 CJCC 44 PLCC	68 CJCC 68 PLCC 68 PGA	48 DIP 68 PLCC 68 PGA	68 PLCC 84 PLCC
Process	Bipolar	Bipolar	CMOS	CMOS	CMOS	CMOS
Technology	Fuse	Fuse	EPROM	EPROM	SRAM	SRAM

Shown above is data on several types of high-density EPLDs. Though the MegaPAL devices offer the most gates, they are of bipolar, fuse-based design, and so are not reprogrammable.