Optimizations for a Highly Cost-Efficient Programmable Logic Architecture

Kerry Veenstra, Bruce Pedersen, Jay Schleicher, Chiakang Sung
Altera Corporation, 101 Innovation Drive, San Jose, CA 95134
kerry@altera.com

1. ABSTRACT

Architects of programmable logic devices (PLDs) face several challenges when optimizing a new device family for low manufacturing cost. When given an aggressive die-size goal, functional blocks that seem otherwise insignificant become targets for area reduction. Once low die cost is achieved, it is seen that testing and packaging costs must be considered. Interactions among these three cost contributors pose trade-offs that prevent independent optimization. This paper discusses solutions discovered by the architects optimizing the Altera FLEX 6000 architecture.

2. OVERVIEW

The FLEX 6000 family of devices is optimized for low manufacturing cost while maintaining target levels of performance and usability. Reminiscent of the FLEX 8000 architecture [1], the FLEX 6000 architecture provides better cost efficiency in the interconnect, Logic-Array Blocks (LABs), and I/O Elements (IOEs). FLEX 6000 capacity ranges from 800 LEs to 1,960 LEs.

The configuration bit in FLEX 6000 is not a traditional RAM (random-access memory) cell but a smaller, *sequential*-access memory cell, providing further cost reduction. It was discovered that in addition to smaller die area, a sequential-access memory cell provides an opportunity for faster configuration times during testing, thereby lower testing cost.

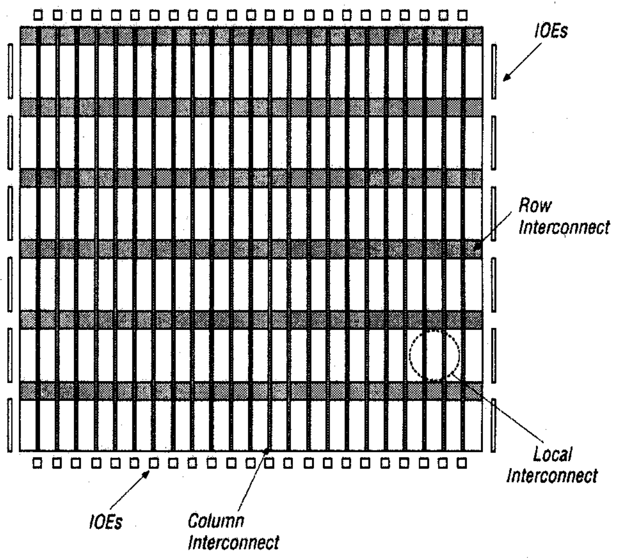

Figure 1: The FLEX 6000 Architecture Block Diagram

Finally, IC packaging, which normally does not affect PLD architectures, had an unexpectedly strong impact on die cost. We discovered that by focusing on the packaging requirements of our industrial partners' future products, rather than looking at PLD shipments supporting their current products, we could eliminate costly support for out-of-date package technologies.

3. INTERCONNECT

FLEX 6000 interconnect improves on the interconnect of the FLEX 8000 architecture. FLEX-style interconnect is organized into three hierarchies: *local*, *row*, and *column* (figure 1). Local interconnect provides communication among the 8 or 10 LEs (FLEX Logic Elements) that constitute a LAB. LABs communicate with other LABs as necessary using the row (GH) and column (GV) hierarchies of interconnect.

This LAB-based approach makes it possible to support fitting through well-known partitioning algorithms [3][4][5]. With appropriate extensions, these algorithms can assist in the fitting of commercial PLDs [6].

3.1 Local Interconnect

FLEX 6000 interconnect differs from that of FLEX 8000 devices primarily in the local interconnect. In FLEX 8000 devices, local interconnect provides communication for the LAB on *one side* of the local interconnect region. But in the FLEX 6000 family, each local interconnect region communicates with LABs on *both sides* (figure 2). Within a row, the LABs and local interconnect regions are interleaved. This change provides die-size savings through two different effects.

First, each LAB can access twice as many local signals. For a given placement, this change results in a greater number of local interconnections and reduces the architecture's need for GH lines, thereby saving die size.

Second, with interleaved LABs, it is not necessary for a partitioning algorithm to equalize the number of cuts leading into each partition from the row interconnect. During placement, partitions with above-average fan-in are paired with partitions that have below-average fan-in, thereby equalizing the fan-in requirements of the partition pairs. Where the FLEX 8000 architecture provides each LAB with sufficient row-to-LAB interconnect for the maximum partition fan-in, the FLEX 6000 architecture can provide a more efficient amount of row-to-LAB interconnect, closer to the average partition fan-in (figure 3.)

3.2 Row Interconnect

In the FLEX 8000 architecture, which was designed on a 0.8-μm process, partially populated crossbar switches route row signals onto local interconnect. SPICE simulations of crossbar switches on a 0.5-μm process showed that capacitance reductions of two process generations would allow the FLEX 6000 crossbar switches not only to route a row signal onto local interconnect but to allow some local signals to be routed onto row interconnect.

As a result, FLEX 6000 row interconnect is more flexible than FLEX 8000 row interconnect. In FLEX 8000 devices, each LE can drive a selected row line. But in the FLEX 6000 architecture, each LE can drive any of ten selected row lines. This added flexibility helps avoid routing blockages.

3.3 Column Interconnect

For performance and fitting reasons, FLEX architectures do not enforce strict boundaries between the interconnect hierarchies; e.g., an LE can drive directly up to the top of the interconnect hierarchy (GV) rather than being required to first route its signal through the local and GH interconnect layers. Connections between pairs of

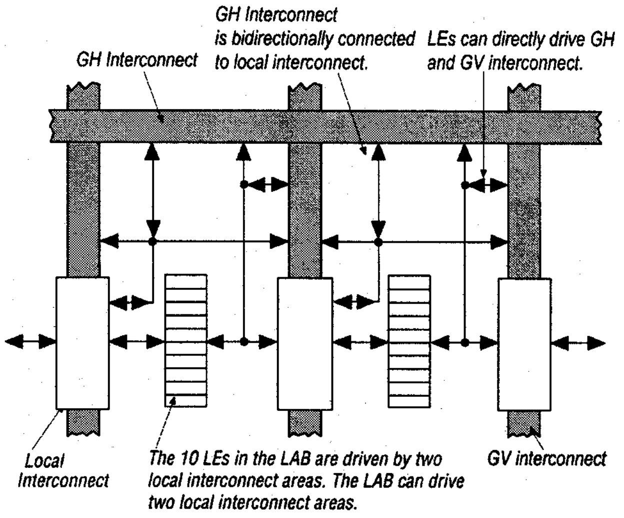
GH Interconnect
GH Interconnect is bidirectionally connected to local interconnect.
LEs can directly drive GH and GV interconnect.
Local Interconnect
The 10 LEs in the LAB are driven by two local interconnect areas. The LAB can drive two local interconnect areas.
GV interconnect

Figure 2: FLEX 6000 interleaves LABs and local interconnect regions

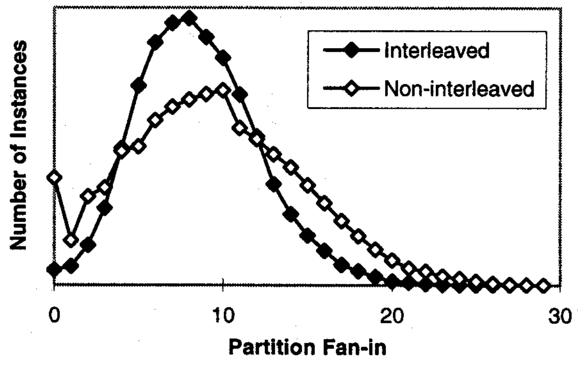

Figure 3: Improvement in partition fan-in resulting from interleaving LABs

interconnect hierarchies are provided in the architecture based on fitting experiments using designs from industrial partners.

Analysis of these designs' performance revealed a need for a direct route from GH to GV lines, complementing the existing connections from GV to GH lines found in FLEX 8000 devices. Such a GH→GV connection was added to the architecture of the FLEX 6000 family.

During fitting trials, a second, and perhaps more interesting, use for this new feature was seen occasionally by software engineers analyzing fitting results. It was discovered that when the fitter routed non-critical lines, it would sometimes specify multiplexer patterns that connected two GV lines together through a GV→GH connection and a GH→GV connection in series. Other times, the multiplexer patterns would connect two GH lines of the same row in a similar way through a series GH→GV connection and GV→GH connection. Analysis showed that in these cases the fitter was taking the "easy way out," using circuitous routes rather than adjusting the placement to allow a more direct approach.

4. LOGIC-ARRAY BLOCK

Based on the analysis of industrial-partner designs, the FLEX 6000 LAB improves on the efficiency of the FLEX 8000 LAB.

4.1 LE

In the FLEX 8000 architecture, all 8 LEs in a LAB are identical. To improve the area efficiency of the family, each FLEX 6000 LAB contains 8 identical general-purpose LEs and two enhanced LEs. The first of the enhanced LEs participates in the generation of LAB control signals. The second improves synchronous counter speed.

4.2 LAB Control Signals

Design analysis revealed that signals such as register clears and counter loads tend to act on LAB-size groups of LEs, questioning the need to provide all LEs of a LAB with independent control signals. The architectural change that we selected allowed us to add to the LAB the first enhanced LE mentioned above with little effect on die cost.

The FLEX 8000 architecture supports LAB-wide clocks and asynchronous clears, but in FLEX 6000 devices the LAB-wide signals also include synchronous clear, synchronous load, and clock enable. Removing the local generation of these three signals from the general-purpose LE reduced its die area and allowed 9 LEs to exist where previously there had been only 8.

Then, to generate the LAB-wide control signals, LE 1 was modified (figure 4). Although this change added complexity to LE 1, it saved area by allowing removal of local interconnect formerly dedicated to the LAB-wide signals.

When a LAB requires one or more non-global control signals, it uses inputs of LE 1. Then, if possible, the fitter places logic in LE 1 that uses the same control signals. In the case where no logic can be found for LE 1, the fitter chooses logic requiring fewer inputs.

A slight impact on fitting was expected for designs that used a large number of independent controls, since logic of these designs would be more limited in its placement. But fitting trials confirmed an overall improvement in device logic capacity relative to FLEX 8000.

Since LE 1 is treated specially, and can have its fan-in reduced, the carry and cascade chains of the LAB do not pass through this LE.

4.3 Counter Sneak Path

A second enhanced LE was added to the LAB to improve counter performance. All FLEX 8000 LEs have a fast register feedback. But only the LSB of the counter, the bit that starts the carry chain, is in the critical path. Therefore in FLEX 6000, only LE 2 has a fast register feedback. Counters requiring the highest performance are placed with their LSBs in this location. Other counters are allowed to shift within the LAB during fitting to provide flexibility.

5. I/O ELEMENT

The choice to interleave LABs and local interconnect regions within a FLEX 6000 row required us to choose a structure for the *end* of the row. In addition, an aggressive goal for support of the PCI 11-ns t_{CO} and 7-ns t_{SU} specifications [7] needed to be met.

The row architecture was balanced so that each interconnect region was exactly sufficient to support the input requirements of the two LABs on either side. Our options for the structure at the end of the row and the consequences of choosing each are listed on the next page.

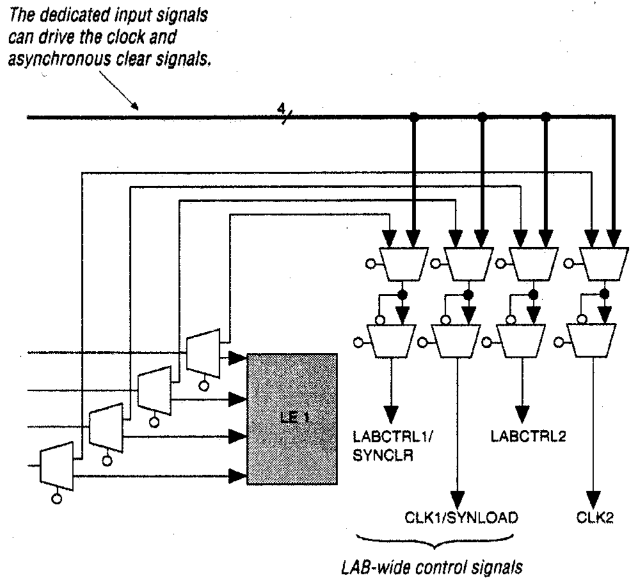
The dedicated input signals can drive the clock and asynchronous clear signals.
4
LE 1
LABCTRL1/ SYNCLR
LABCTRL2
CLK1/SYNLOAD
CLK2
LAB-wide control signals

Figure 4: Enhanced LE 1 generates LAB Control Signals

LAB. A full LAB on the end of a row would be starved for interconnect.

Half-LAB. Ending a row with a special half-size 5-LE "LAB-ette" would have complicated the fitter unacceptably.

Full interconnect region. Leaving a full interconnect region on the end of a row to support a single LAB would waste die area since it would provide an unnecessarily abundant amount of interconnect to the LAB.

Half interconnect region. Designing and laying out a special half-size "end-cap" region that provided signals to only one LAB would save die area, but the added design complexity would push out the schedule and complicate the fitter.

IOEs. Replacing the outermost LABs with blocks that require local interconnect signals from only one side would allow us to use the same interconnect region throughout the device. Since each side of a local interconnect region generates 20 outputs, it was strongly suggested that the outermost interconnect regions be connected to 10 tri-state I/O buffers, each of which normally has 2 inputs: data and enable (figure 5). This choice would not affect the design schedule because the IOE needed to be designed anyway.

Our decision to use this last alternative—and simply attach the IOEs to the existing outermost local interconnect regions—provided three benefits. First, the IOEs' position allowed FLEX 6000 devices to comply with the PCI 11-ns t_{CO} and 7-ns t_{SU} specifications. Since the IOEs communicate using the same local interconnect region as the LEs of the adjacent LAB, delays were sufficient to achieve the goal of a 33 MHz PCI target.

Second, by providing independent output enables for every IOE, it is possible to emulate open-drain outputs by holding an IOE's data input low and toggling the output enable.

Third, fitting with pin assignments is improved. Rather than providing flexibility between LEs and IOEs through a ring of dedicated multiplexers, as is done in the FLEX 8000 architecture, the FLEX 6000 family uses existing local interconnect. In this way, the fitter can trade-off interconnect used for routing and interconnect used for pin assignments.

A final interesting consequence of the FLEX 6000 IOE interconnect architecture is that in some future device, IOEs could be placed anywhere *in the middle of the die* by replacing any LAB with a set of 20 IOEs. This alternative could be used if area-array bonding becomes more prominent in low-cost devices.

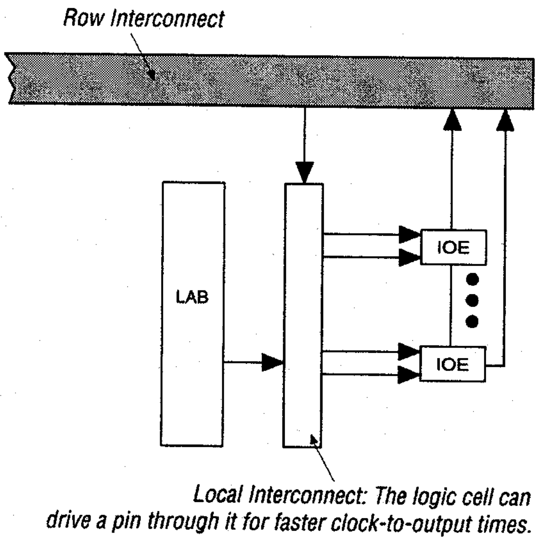
Row Interconnect
LAB
IOE
IOE
Local Interconnect: The logic cell can drive a pin through it for faster clock-to-output times.

Figure 5: FLEX 6000 IOEs and outermost LAB connect to local interconnect

6. CONFIGURATION BIT

FLEX 6000 devices are manufactured on a basic CMOS logic process. A traditional means for creating memory with such a process is to construct static RAM cells from two cross-coupled inverters, an access transistor, a word line, and a bit line (figure 6a).

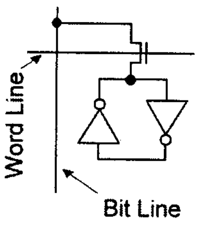

Figure 6a: RAM

FLEX 6000 configuration memory is not RAM. Instead it uses a smaller, *sequential*-access memory (SAM) cell (figure 6b). A SAM array has no bit lines, but instead uses a word line and a single access transistor to gate the data of each cell onto its neighbor. This organization saves one metal line per column of memory cells.

One way to program a SAM array is to turn on all address lines and then feed the data for the bottommost word into the top of the array. This data propagates to the bottom row of cells, which is then write-protected by turning off its address line. Then the *inverse* of the next word's data is fed into the top of the array. This inverse data propagates down to the second row from the bottom. At this point, the next address line up is turned off, write protecting the second word.

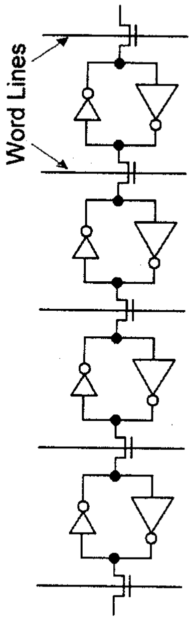

Figure 6b: SAM

This process continues, driving the top of the array with successive words (inverting alternate words) and turning off the address lines one at a time from the bottom of the array to the top, until the array is fully configured. Reading the array for testing purposes is accomplished in a similar way, turning on successive address lines starting at the bottom, and reading successive words as they emerge from the bottom of the array.

Notice that pairs of cells in a column form the master and slave latches of a shift register's flip-flops. A high-speed SAM-array test mode creates virtual shift registers through which data can propagate downwards rapidly by alternately turning on all even address lines, followed by turning on all odd address lines. The massive configuration-stream bandwidth provided by this technique precludes using a serial data input pin with a CRC check, and so in this mode, configuration data is provided in parallel from the topmost pins of the device. Although the mode lacks the safety of a CRC check, thereby rendering it unsuitable for customer use, it nonetheless provides faster configuration during testing and therefore lowers test costs.

Note that a SAM array lacks true random access since each cell must program its lower neighbor before it can be programmed itself. But since true random-access configuration complicates or makes impossible the CRC checks found in PLD configuration streams, random-access configuration was not made a requirement of the FLEX 6000 architecture. The alternative to a CRC, adding circuitry in the device to allow it to tolerate configuration data errors, was deemed inappropriate for a low-cost architecture.

7. PACKAGING TRADE-OFFS

PLCCs (plastic leaded chip carriers) are very popular semiconductor packages. However, if the FLEX 6000 architecture had attempted to support PLCCs, the die would have needed to use the 4-to-6 mil bond-pad pitch that is required for PLCC wire bonders [2].

Instead, the architecture was based on an internal analysis of new, high-volume design wins which determined that PQFP (plastic quad flat pack), TQFP (thin quad flat pack), and BGA (ball grid array) packages are the most popular packages for new, high-volume designs. New QFP wire bonders allow a die bond-pad pitch of 3.2 mils, which is 20% to 47% denser than the 4-to-6 mil pitch required by PLCC wire bonders.

As a result, we gave the FLEX 6000 family a quantity of IOEs sufficient to support pads placed on a 3.2-mil pitch around the die. In this way, the *cost per I/O* is reduced by between 20% and 47% compared to dice with 4-to-6 mil bond-pad pitches. We found that the small die-size increase necessary to provide drivers for the additional I/O pads was more than justified by the value of the additional I/Os. Specifically, the EPF6016 with a 3.2-mil bond-pad pitch has sufficient I/O to support 208-pin PQFPs, 240-pin PQFPs, and 256-ball BGAs. But if the die had used the 4-mil bond-pad pitch required by PLCC packaging vendors, it would have had only enough I/Os to support the 208-pin PQFP.

8. CONCLUSION

During the development of the FLEX 6000 device family, it was necessary to question all previous assumptions. For instance, the size of the control block became a cost factor in the low-cost FLEX 6000 devices while it had been given relatively less attention in the higher-density FLEX 10K devices.

It was learned that designers will not find single, simple solutions when optimizing an architecture for high cost efficiency. Traditional approaches, which expect to identify and reduce just the top cost contributors, fail when silicon, packaging, and testing all contribute nontrivial portions of total product cost.

9. ACKNOWLEDGEMENTS

The authors wish to thank those who contributed to the success of the *Botticelli* architecture development: Andrew, Bahram, Bonnie, Bruce, Cameron, Chris, Clive, Craig, David, Dustin, Frank, Fung Fung, Jay, John, Joseph, Peter, Richard, Robert, Srinivas, Susan, Tim, and Xiaobao.

10. REFERENCES

[1] Altera, *Data Book*, 1996, pp. 91-153.

[2] Amkor Electronics, Inc, *Quad Product Guide*, December 1996, p.16.

[3] Fiduccia, C. M. and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," *19th Design Automation Conference*, 1982, pp. 241-247.

[4] Kernighan, B. W. and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Technical Journal*, Feb. 1970, pp. 291-307.

[5] Krishnamurthy, B., "An Improved Min-Cut Algorithm for Partitioning VLSI Networks," *IEEE Transactions on Computers*, vol. C-33, No. 5, May 1984, pp. 438-446.

[6] Mendel, D. W., "Methods for allocating circuit elements between circuit groups," United States Patent no. 5,341,308, Issued Aug. 23, 1994.

[7] PCI Special Interest Group, *PCI Local Bus Specification, Revision 2.1*, June 1995.