# A 65nm Flash-Based FPGA Fabric Optimized for Low Cost and Power

Jonathan Greene, Sinan Kaptanoglu, Wenyi Feng, Volker Hecht,
Joel Landry, Fei Li, Anton Krouglyanskiy, Mihai Morosan, Val Pevzner

Microsemi Corporation SOC Products Group
2061 Stierlin Court, Mountain View, CA 94043 USA
jonathan.greene@microsemi.com

## ABSTRACT
This paper describes a non-volatile reprogrammable FPGA fabric, whose configuration data are provided directly by flash memory. The fabric is optimized for low-cost, low-power applications, leveraging the density of flash and the elimination of conventional configuration SRAM and its attendant static power. After surveying the necessary background on flash and its application to FPGAs, the 1T flash cell is described along with relevant novel aspects of the fabric architecture. The addition of a third level of switching between inter-cluster signals and logic inputs helps to reduce area and raise typical utilization above 95%. Despite the longer signal path, performance is maintained by synergism between the improved routing flexibility and extreme minimization of the fastest LUT input delay. Test cost is reduced by built-in circuits that can test all switches without reprogramming the flash memory. The fabric has been implemented in a 65nm CMOS embedded flash process.

## Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Style—*gate arrays*

## General Terms
Design, Experimentation, Theory.

## Keywords
FPGA architecture, flash memory, Clos network, built-in self-test, programmable routing, input interconnection block, IIB, packing

## 1. INTRODUCTION
The market for integrated circuits with embedded flash memory, principally microcontrollers, is expected to be about $8 billion in 2010 [1]. (This compares to about $4 billion for all types of FPGAs [2].) Embedded flash is clearly serving important market segments.

Flash can also bring significant advantages to FPGAs, especially for low-cost, low-power applications and when security or high reliability is important. Dense blocks of non-volatile memory can be integrated with the programmable logic, and the high-voltage transistors included in the flash process can be leveraged for analog circuitry, providing a multi-faceted system-on-a-chip.

There are two general ways embedded flash memory can be used to store configuration data in an FPGA. One is to integrate a conventional SRAM-based FPGA with a flash memory block from which the configuration can be loaded [3]. The other way is the focus of this paper: using flash memory to provide configuration data directly to the FPGA fabric's logic and interconnect. By replacing the configuration SRAM entirely, these truly flash-based FPGAs eliminate the attendant static power. Flash-based FPGAs are also immune to errors in configuration (firm errors) that may be caused when high-energy particles flip an SRAM bit (single-event upsets).

Flash-based FPGAs have been in the market for many years [4], but their special architectural considerations have not been previously described in the literature. This paper seeks to rectify that, and to describe a new flash FPGA fabric that is more efficient than its predecessors. It has been implemented in 65nm CMOS, and is targeted at low-cost and low-power applications.

Low cost is not simply a matter of raw logic elements per unit die area. By a few architecture changes, we demonstrate a significant increase in the utilization of the logic elements above the level typical of SRAM-based FPGAs. Test cost is another important issue, especially for smaller FPGAs.

SRAM-based FPGAs are generally tested by programming them with a sequence of hundreds of test designs [5][6][7]. Because flash-based FPGAs take longer to program, special care must be taken to limit the number of test designs required. This is not a problem for the simple, flat architectures of previous flash FPGAs [4], but would be for the more complex, clustered architecture described here. We therefore developed a novel built-in test method that eliminates the need for most test designs.

Section 2 covers essential background on flash technology, and its implications for flash FPGA architecture. Section 3 gives an overview of the architecture, focusing on its unique aspects. Section 4 reports some theoretical and experimental results concerning the architecture. Section 5 describes the new test scheme.


*FPGA'11*, February 27–March 1, 2011, Monterey, California, USA.

# 2. FLASH MEMORY ESSENTIALS

## 2.1 Technology

Figure 1 gives a diagram of an embedded flash process. The flash device has a second, floating gate on which charge may be stored. The amount of stored charge determines the threshold voltage Vt of the transistor. An additional deep n-well layer is used to isolate the flash devices, preventing high voltages present on them during programming from damaging the ordinary low-voltage logic. Special spacing rules apply between the floating gate and the ordinary low-voltage logic.

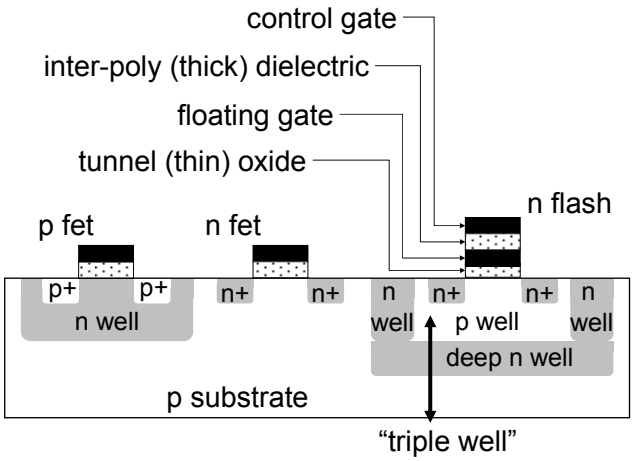


**Figure 1: Embedded flash process.**

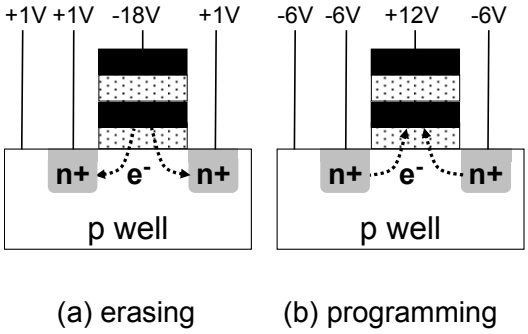


**Figure 2: Erasing and programming.**

Figure 2 shows how charge is stored or removed from the floating gate. The voltages shown are rough indications only.

During erase, electrons are removed from the floating gate by a mechanism known as Fowler-Nordheim tunneling [8], which lowers Vt and turns the device on. Erasing is typically done in bulk for all devices simultaneously because it takes on the order of seconds.

Once all devices are erased, selected devices are programmed. A device is selected by raising its control gate and lowering its source and/or drain. This adds electrons to the floating gate, again by tunneling. Programming is significantly faster than erase, on the order of milliseconds. Multiple devices may be programmed in parallel as long as they can be properly addressed; however, it is important not to activate any one control gate too many times to avoid disturbing devices that should remain erased. Programming may also be done by hot carrier injection instead of tunneling. Injection takes even less time than tunneling, but requires more current. Hence fewer devices can be programmed simultaneously with a fixed current budget. For details see [8].

Because of the voltages required for programming and erasing, flash processes include special high-voltage transistors with thicker oxides. (As mentioned above, these may also be useful for implementing analog circuitry.)

## 2.2 Flash Configuration Memory Cells

Figure 3 shows a "1T" flash configuration cell [9]. It replaces the conventional 6T SRAM and NMOS pass device with a single minimum-sized flash transistor (sense device) and a larger flash switch device that share a common floating gate. Programming and erasing is done using the sense device and the word and bit lines to which it is connected. The cell's state can also be read back to verify correct programming in a similar way.

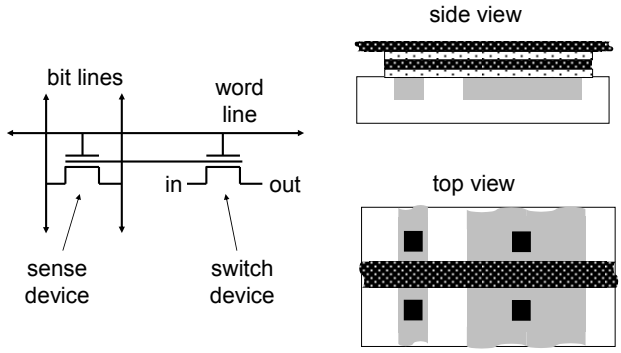


**Figure 3: 1T cell schematic and conceptual layout.**

During operation, the word line is set to an appropriate bias voltage. In the erased (on) state, the switch's Vt is low enough that the switch will pass full Vdd, eliminating the need for the level-restoring devices sometimes required with NMOS pass devices ([10], p. 107). The Vt in the off state is high enough that typical leakage is on the order of only 10pA.

The 1T cell is able to retain its configuration with zero static power. In addition, it is immune to firm errors caused by high-energy particles [11].

Non-volatile FPGAs have a unique niche in high-radiation environments, notably aerospace applications, so it is important to consider the cell's suitability for such uses. Radiation can cause some loss of charge from the floating gate, raising the on-resistance or lowering the off-resistance [9]. Studies on prior 130nm FPGAs using this cell indicate a performance loss of 10% occurs only after a total ionizing dose (TID) of 22Krad for gamma rays. However it is possible to extend the dose by periodically reprogramming the configuration [12], in a manner conceptually similar to refresh in DRAMs. Configuration errors would not be encountered between properly scheduled refreshes, as would be the case between periodic scrubbings of firm errors in SRAM FPGAs.

If superior radiation tolerance is required, the alternative "2T" cell of Figure 4 may be considered [13]. It includes a pair of n- and p-channel flash devices controlling an NMOS switch device. During normal operation, VSP is at a positive voltage, VSN is at ground, and appropriate biases are applied to VCGP and VCGN. Either one or the other flash device is programmed to conduct, thus turning the switch on or off. Provided that VSP is sufficiently high, the switch can still pass a full Vdd. As the name would imply, the 2T cell is less dense than the 1T, but is highly radiation tolerant.

Because our goal is to support low-cost commercial applications, we employed the 1T cell for this work. However many of our findings would also apply to architectures using the 2T cell.

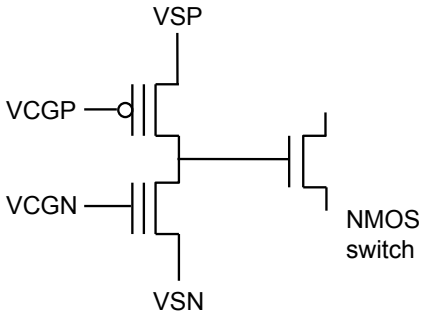


**Figure 4: 2T flash configuration cell [13].**

## 2.3 Implications for Architecture

Due to the required well spacings and separation between floating gates and logic, it is most efficient to segregate flash and low-voltage logic into separate stripes in the layout. An appropriate tradeoff must be made between using a few wide stripes to minimize the total separation area versus using many narrow stripes to minimize the length of wires connecting the switches and logic.

Minimizing the capacitance downstream of the switches is important for good performance, especially if the on-resistance of flash switches exceeds that of nmos pass gates. Also, the integration of the sense and switch devices in one cell means that it is very cheap to independently control each switch. These factors tend to favor use of routing muxes having a single level of switches and lower fan-in than is customary in SRAM FPGAs.

This reasoning is exemplified in Figure 5. Part (a) shows a typical 16-input routing mux for an SRAM FPGA, realized as a two-level structure comprised of 20 switches sharing 8 configuration bits. For comparison, part (b) shows two independent flash muxes with a reduced fan-in of 8. These use a total of sixteen 1T flash cells. (The sense devices, not shown, consume negligible area.) With 20% fewer switches and no configuration SRAM, the two flash muxes consume significantly less area. Yet they offer 8x8=64 routing alternatives, instead of only 16 for the SRAM alternative.

During program or erase operations, it is necessary for at least one of the switch device's source/drain terminals be held at Vdd. This is accomplished by replacing the last stage of all routing buffers with a circuit such as that shown in Figure 6. Here NGND is a global switched supply that is at ground during normal operation but raised to Vdd during programming. The pullup ensures that the output is high no matter what the state of the input. The pullup may be minimum-sized and has no impact on performance (unlike the similarly-situated level-restoring pullups found in some FPGAs).

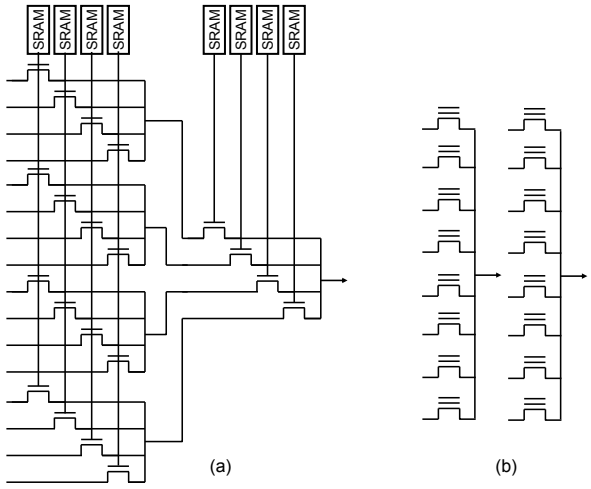


**Figure 5: Muxes configured by SRAM vs. flash.**

**(a) A conventional 16-input mux with 8 SRAM configuration bits. (b) Two 8-input muxes using the 1T flash cell. (The minimum-sized sense device associated with each switch is omitted for clarity.)**

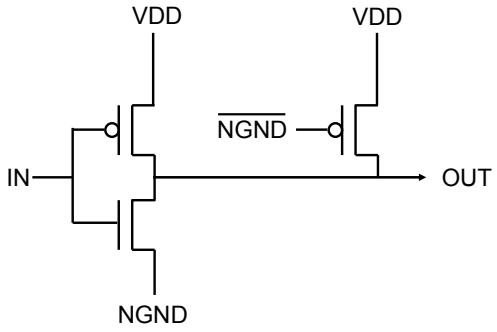


**Figure 6: Routing buffer power gating.**

# 3. ARCHITECTURE OVERVIEW

Our goal for the architecture was to make it efficient for flash technology while achieving a balance of area and delay suitable for low-cost applications.

## 3.1 Logic Cells

The basic logic element (BLE) comprises a 4-input look-up table (LUT) and a dedicated flip-flop, as shown in Figure 7. The LUT size is appropriate for a low-cost, low-power architecture [14]. The flip-flop has control inputs for enable, and synchronous and asynchronous set/reset. Dedicated carry-chain support is also provided.

Each BLE produces a LUT output Y and a flip-flop output Q. (The carry output is connected directly, and only, to the next BLE's carry input.) For efficient packing, the LUT and flip-flop in a BLE can both be used independently, without either one's output having to drive an input of the other. Unlike the standard VPR architecture [15] we do not reduce the Y and Q outputs via an "OUT" mux to a

single output. Instead, both Y and Q are made directly accessible to the routing.

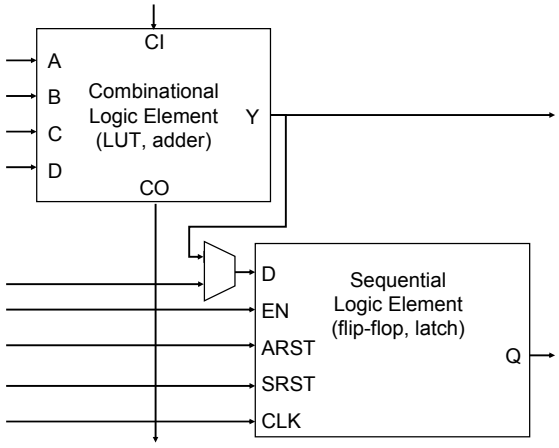


**Figure 7: Basic logic element.**

The LUT inputs are denoted A, B, C and D, in order of increasing delay. The LUT circuit was optimized to minimize the delay of the A input as much as possible without unduly impacting the average input delay, obtaining a ratio of 4.8 between the delay of the D and A inputs. The speed of the D input is relatively slow in part because a flash configuration mux selecting from among 0, 1, D and $\overline{D}$ is used instead of the traditional two-input mux with D as the select input; this saves area. Figure 8 illustrates this for the first two levels of a LUT.

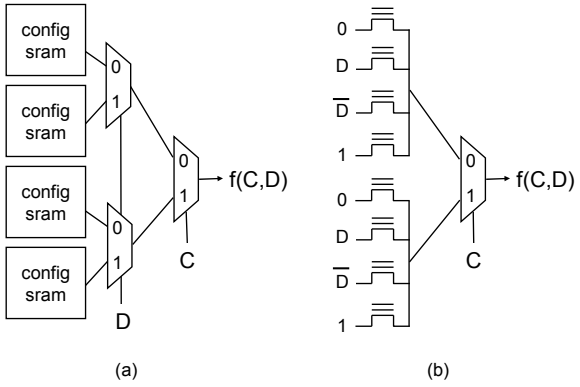


**Figure 8: LUT implementations: (a) SRAM; (b) flash.**

We found that a cluster size of 11 was optimum for area-delay product, but the curve was fairly flat over a range of 10 to 14. We chose a cluster size of 12 because it is divisible by a variety of factors, which helps maximize layout symmetry when providing routing resources in various appropriate quantities. For example, a resource of which three are required per cluster could be laid out in units of four BLEs, while a resource of which four are required per cluster could be laid out in units of three BLEs.

Most often, a LUT output in the fabric can be inverted and the downstream logic adjusted to compensate. Design software can thus automatically choose LUT output polarities to minimize static power, or to help balance rise and fall delays along a critical path [16]. Compensating at a downstream LUT input is simply a matter of adjusting the LUT function. An inversion at the D input of the flip-flop is handled by propagating the inversion forward to the flip-flop's output and interchanging set and reset options. Finally, all clock and control inputs to the flip-flops may be optionally inverted at the cluster level. This last feature also facilitates support for rising and falling edge flip-flops, and active-high or active-low control signals.

To aid debugging, the state of any two flip-flops may be read continuously at two IOs (analogous to a "peek" in memory terms). The flip-flop addresses are provided via a JTAG interface. In addition, the state of any flip-flop can also be set ("poked"), again via JTAG.

## 3.2 Routing

Inter-cluster routing tracks of various lengths are provided. Linear tracks of length 1, 2 and 5 clusters are provided in all four directions. A few longer tracks are also provided; their exact quantity and length depend on the size of the FPGA, which may range from less than 1K to over 200K BLEs.

Intra-cluster routing to the LUT inputs was a special challenge, primarily because such routing typically has muxes with high fan-ins that would be suboptimal for a flash implementation. Our solution is described in the next section.

A smaller but analogous intra-cluster routing network drives the flip-flop inputs. These include clock and control signals (enable, set/reset), as well as a separate data input so the flip-flop can be used independently of the LUT. This routing network primarily receives input from a dedicated global signal (e.g., clock) distribution network, but also receives input from a few of the ordinary external signals and muxes used to drive the LUT inputs.

Rows of RAM or math blocks can be embedded in the fabric. Because of the changes to our intra-cluster routing, the larger bandwidth requirements of these embedded blocks are accommodated without any alterations to the pattern of inter-cluster routing (such as providing additional tracks). The external connections of a RAM or math block interface tile exactly match those of an ordinary logic cluster.

With one exception described in the next section, the output of each routing mux is immediately buffered to minimize capacitive load.

## 3.3 Highlights of Intra-Cluster Routing

This section describes how external (inter-cluster) signals and feedback from BLE outputs reach the LUT inputs.

Initially, we considered a traditional two-level network (like figure 3.1 of [15]), as exemplified in Figure 9. Even with 50% depopulation of the cross bar, fan-in is 20 on the muxes comprising the crossbar, and 24 on the preceding muxes. Such high fan-ins are suboptimal for implementation in flash switches. In addition, the fact that each external signal drives more than one mux in the cluster adds capacitive loading due to the spur wires needed to tie each fan-out to the main inter-cluster trunk carrying the signal. Finally, this scheme permits only 28 external signals to be routed to the LUT inputs. This narrow bandwidth can be limiting, especially for logic using the carry chain or interfaces to RAM or math blocks embedded in the fabric. (Of course more first-level muxes can be

added, but this would require more flexibility in the crossbar and additional area).

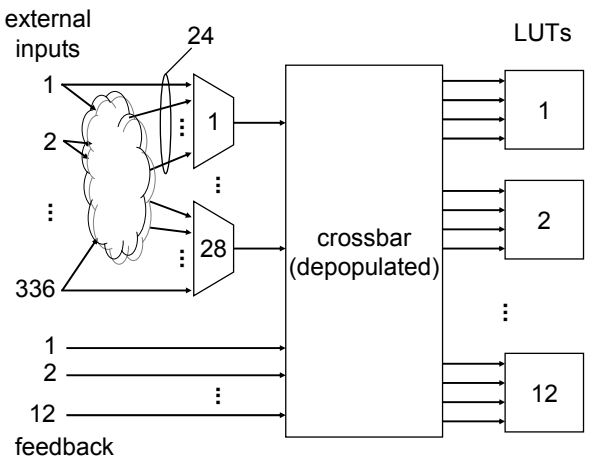


**Figure 9: Two-level cluster input routing scheme.**

**The cloud represents an arbitrary permutation.**

A recent paper [17] proposed an alternate architecture, reducing the crossbar population to 25% and the fan-out on external signals to one. It addresses the fan-in, spur, and bandwidth problems. However it allows only 25% of the external and feedback inputs to reach the fast A inputs of the LUTs, which can impact performance.

Instead, we replace the crossbar with a Clos network [10] [18], as shown in Figure 10. Normally a Clos network is built of three levels of (smaller) crossbars, but because the LUT inputs are interchangeable (except for speed differences), the third level is absorbed by the LUTs.[1] The total number of levels of switches between external signals and LUT inputs is thus increased only from two to three.

The efficiency of the Clos network allows a larger number of inputs to the second level than was the case with the single large crossbar. The added inputs were as follows. For improved performance, about 10% of the external signals are allowed to also bypass the first level muxes and drive the second level directly. The number of first level muxes driving the second level is significantly increased, which allows us to reduce their fan-in.

For improved intra-cluster routability, feedbacks from both LUTs and flip-flops are sent to the first level muxes. For improved performance, feedback from each LUT is also sent to the third-level crossbar driving the fastest LUT inputs. We did not provide direct feedback from flip-flops to levels 2 or 3 because flip-flops are less likely to drive only intra-cluster signals.

[1] Similar networks have been proposed for multi-FPGA emulation systems connected by multiple crossbar ICs where each FPGA, like one of our LUTs, has interchangeable inputs. See, e.g., [19].

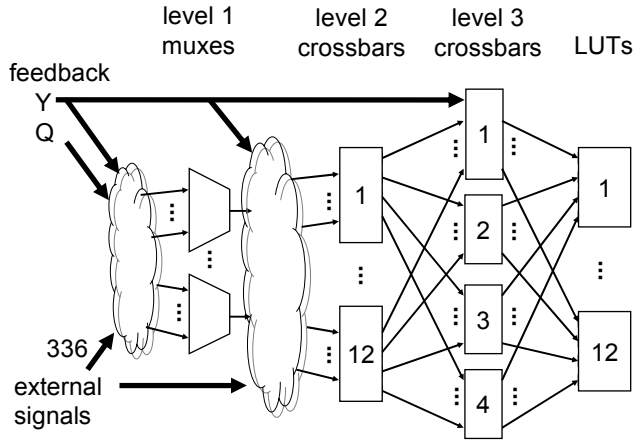


**Figure 10: Three-level cluster input routing.**

**The clouds represent permutations, and the thick lines represent bundles of wires.**

The obvious disadvantage of the added level of switching is extra delay. This is mitigated in three ways. First, the spur line capacitance is reduced because most external signals now drive only one load (per cluster). Also, notice that each crossbar in level 2 and the muxes in level 1 driving it can be laid out in a compact unit. This allows us to keep wires short and avoid intermediate buffers. As a result, the additional delay due to the added level of switching is under 100ps. Finally, as we show in Section 4.3 below, the increased flexibility of the three-level scheme helps route the most critical nets to the fastest LUT inputs.

To save area, we limit the number of middle-level crossbars in the Clos network to four, even though this is insufficient to make it rearrangeably non-blocking. Further savings are achieved by depopulating portions of the crossbars. The second-level crossbars are only 78% populated for inputs from the first-level muxes, and 25% for external inputs. The feedback inputs to the third-level crossbar are only 25% populated.

Despite the added routing paths, total switch and buffer area is reduced by about 20% compared with the two-level scheme. The maximum fan-in is reduced to 12 for most muxes in the third level and 15 for those receiving feedback. The fan-in for the first and second-level muxes is reduced to less than 10. Every external and feedback signal can (in the absence of conflicts) be routed to the fastest input of any LUT if it is necessary for speed. The bandwidth limit on incoming external signals is eliminated.

## 3.4 Low-power Features

A low-leakage process is used to minimize static power while still achieving the desired performance.

As described in Section 0, flash technology requires power gating the routing buffers. It is therefore a simple step to extend the power gating to the logic as well and support a power-down mode to reduce static power. During this mode, the state of the flip-flops is retained in low-power latches. Likewise the block RAMs may be put in a low-power mode that retains their state but reduces static power.

It is also possible to power down all but a portion of the fabric. This remaining live portion may be used to implement wakeup logic in soft gates, for example.

Dedicated clock gating circuitry is provided on the global signal distribution networks at the root and at an intermediate point along the H-tree.

The fabric circuitry and layout is designed to be manufactured in either of two versions: a performance version that uses a mixture of standard and high threshold voltage transistors, and a low-power version that uses only high threshold voltage devices.

# 4. ARCHITECTURE EVALUATION

This section presents a few theoretical and experimental results supporting the noted features of the architecture. (It is not intended as a complete validation of the architecture.)

The benchmark designs used included a variety of industrial circuits, including microprocessors, encoding circuits, etc. In addition, we employed synthetic designs constructed by stitching together smaller designs in a way that satisfied Rent's rule and maintained the statistical properties (e.g. fan-out distribution) of the industrial circuits. Timing-driven placement and routing was rerun for each experimental situation with multiple random seeds and the results averaged. A modified version of our commercial design software was used.

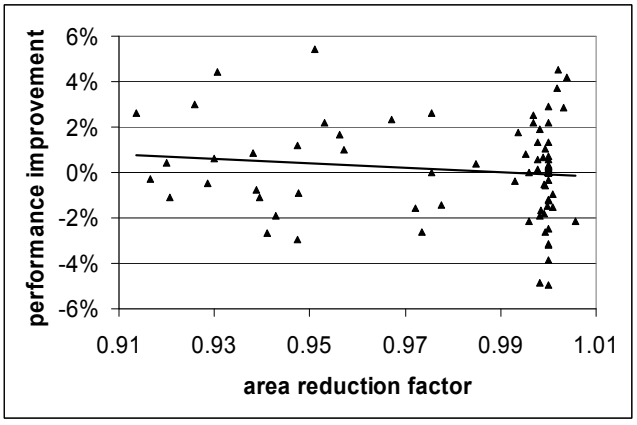


**Figure 11: Change in performance and area from allowing up to two I5-packed BLEs per cluster vs. none.**

## 4.1 Impact of Packing on Utilization

As mentioned above, the architecture supports packing an independent LUT and flip-flop in same BLE even when the flip-flop data input and four LUT inputs all come from outside the BLE. We refer to this as I5 packing. This support costs only a small amount of area, about 0.5%, but it is nevertheless interesting to see if it proves worthwhile. A limit on the number of I5-packed BLEs in a cluster is imposed by the clustering software to avoid undue congestion. We evaluated a set of test designs under various values of this limit. Allowing up to two I5 BLEs per cluster reduced the total number of clusters by about 2% on average, with diminishing returns for higher limits.

We compared the area (number of clusters used) and performance of our benchmark designs with an I5 limit of 2 versus a baseline of 0 (no I5 packing allowed). The results are plotted in Figure 11. While most designs were not greatly improved, some saw area reductions of up to 9%. As shown by the least-squares fit line, average performance was maintained.

## 4.2 Theoretical Comparison of Cluster Input Routability

For purposes of this section, we assume that the four signals driving each LUT may be routed to the LUT inputs in any permutation.

A useful way to quantify routing flexibility is the logarithm of the number of distinct sets of routable connections, or "routing entropy" [17]. For example, the entropy of a cluster's internal interconnect has been shown to correlate with the demands it imposes on usage of inter-cluster resources.

The entropy of the three-level scheme is at least 25.9 bits per LUT. This lower bound is obtained by: giving credit to the second and third level switches only for unicast routes; ignoring the added feedback connections to levels 1 and 3; and ignoring external connections to level 2.

This compares to bounds of 24.6 to 26.2 bits per LUT for the two-level scheme even with a fully populated crossbar (see [17] for details). So the three-level scheme is likely to have routability as good or better despite its lower area.

The next section provides some experimental support for this conclusion.

## 4.3 Impact of Utilization on Performance

A prior study [20] of a particular two-level architecture found that as utilization is increased, performance declines. In particular, an 18% increase in utilization results in a 5% decrease in performance. We tested whether our architecture escaped this tradeoff by comparing the performance at low and high utilizations. This was accomplished by generating two appropriate-sized virtual FPGAs to test each design, a larger array for lower utilization and a smaller array for higher utilization. Here, utilization is defined as the fraction of BLEs in the chip that are used. The geometric mean utilization of the designs in their respective small arrays is 80%, and the geometric mean utilization of the designs in their respective large arrays is 97%. For each design, the utilization in the smaller array is always at least 1.12 times that in the larger array.

Results are shown in Figure 12. The vertical axis is the ratio of frequency at high utilization to frequency at low utilization. The geometric mean of this ratio across designs is 1.00, showing that performance is not degraded by higher utilization.

## 4.4 Impact of LUT Input Delay Distribution on Performance

The previous section shows that performance is maintained even for utilizations exceeding 95%. We hypothesize that this is so at least in part because the routing network (and the three-level cluster input network in particular) is able to preferentially route the most critical nets to the fastest LUT inputs. To quantify the magnitude of this effect, we compared the maximum achievable clock frequency using our skewed LUT input delays to that obtained when the LUT input delays are equalized to their average value. The results are shown in Figure 13. The vertical axis is the ratio of frequency with our stated LUT delays to frequency with equalized delays. The geometric mean of this ratio across all test designs is 1.13.

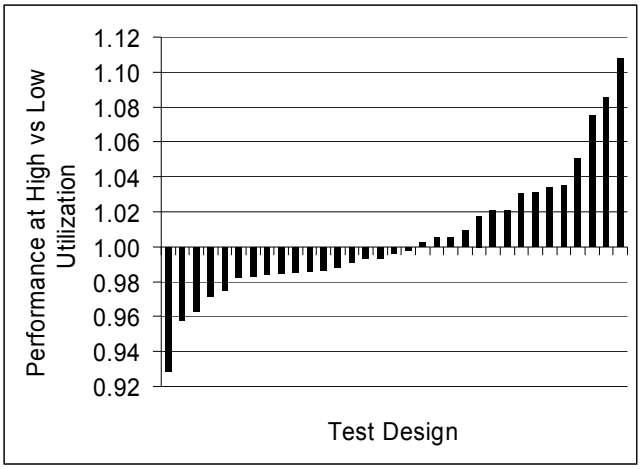


**Figure 12: Impact of utilization on performance.**

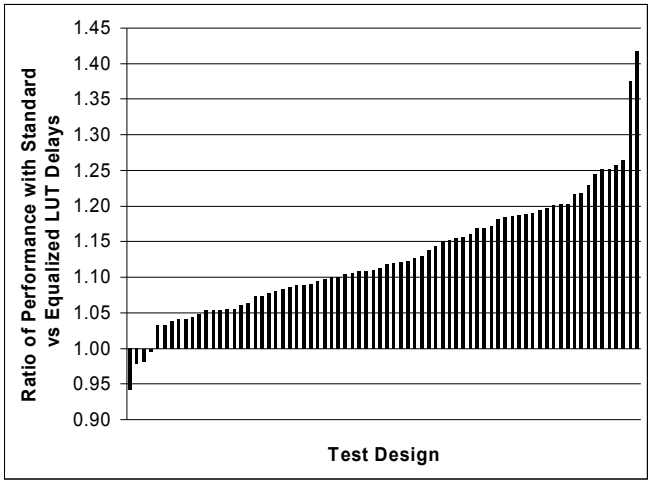


**Figure 13: Impact of LUT input delays on performance.**

## 5. TEST COST REDUCTION

Various categories of faults must be covered to ensure correct functionality of a reprogrammable FPGA, including:

- faults in logic cell and routing buffers
- bridging (shorts) between routing tracks
- routing switches that are stuck closed or open, or discontinuities in routing tracks

The third category accounts for the vast majority of the necessary tests in our fabric. (This has also been found to be the case for SRAM FPGAs [7]). In the remainder of this section we describe test circuitry we added to our fabric to cover such faults without needing to program any test designs.

The first requirement is some means to exert control of the switches in the 1T flash configuration cells. Recall that to support programming, the switches are arranged in an array, with a word line connected to the control gates of all switches in a row, as shown in Figure 14. (The sense devices have been omitted for clarity). If all switches are set to the same threshold voltage (e.g. all erased), selected rows of switches can all be turned on or off by raising or lowering the word lines to appropriate voltages. This addressing capability, though limited, suffices for our purpose.

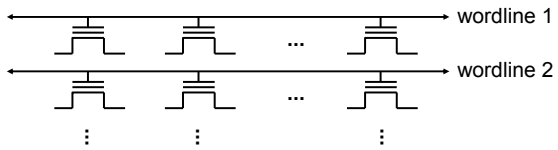


**Figure 14: Addressing switches by row.**

The next requirement is a means of driving all the routing tracks to a known logic state. Here we take advantage of the pullups and switched ground supply associated with the routing buffers (Figure 6). Finally, we must be able to observe at least one input to each mux. This is done by reusing the bit lines and sense amps already provided for programming. Figure 15 shows a typical test path for two switches in a mux. The word lines for these switches are activated, and TSTEN is raised to reconnect input B to the bit line. If switches A or B are stuck open, or if there is a break in the relevant portion of track 1, the sense amp input will not be pulled high. Most muxes have at least one input that is a constant, and thus is never in the speed path; these are the best choices for the input to be observed via the TSTEN devices. The only additional circuitry required are the devices controlled by TSTEN.

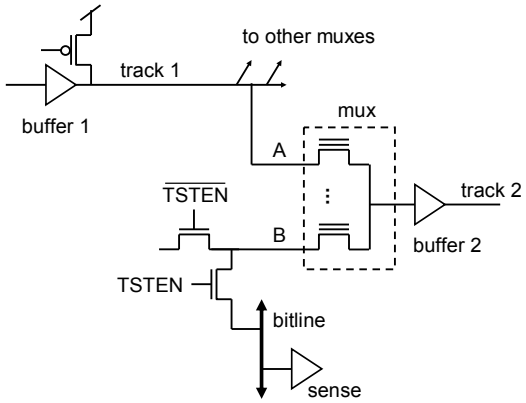


**Figure 15: A typical test path.**

Some care is required to ensure that all switches can be properly addressed for testing. For instance, no mux can have two of its switches in the same row. Similar techniques can also cover stuck-closed switches. A full explanation is beyond the scope of this paper, but we illustrate below two of the more interesting complications that must be dealt with.

Figure 16 illustrates how parallel paths may arise. When testing switches A and B in mux 1, switches C and D in mux 2 must also be turned on since they share the same word lines. This forms a parallel path from Vdd to A via track 2, C, Y, and D. As a result we may miss a break in track 1 at the indicated location. However this test in

combination with a separate test of switches D and E in mux 2 will assure continuity of all parts of track 1.

Another complication arises if there are unbuffered muxes (such as the level 1 muxes in Figure 10). Figure 17(a) represents a wiring tree of unknown topology with input pins A and B, and output pins C and D. Figure 17(b) shows how such a tree may arise when the tree is driven by an unbuffered mux. Suppose we have verified continuity from A to C with one test (c) and from B to D with another test (d). Have we completely verified continuity of the wiring tree? It depends on the layout of the tree. For the layout in Figure 17(e), the answer is yes. For the layout in (f), the answer is no; we might have missed a break at the indicated position in the center of the wiring tree.

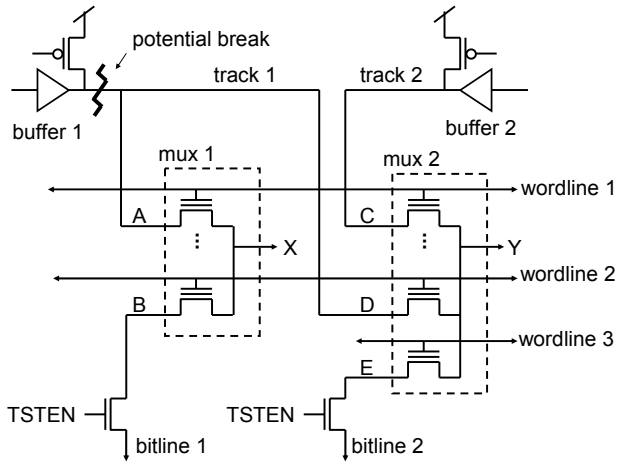


**Figure 16: A test path with a loop.**

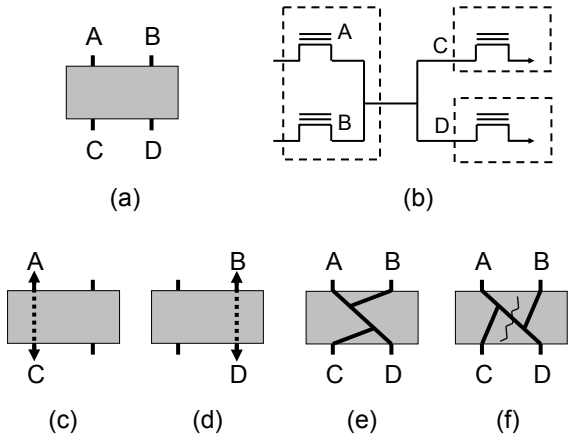


**Figure 17: Wiring tree connectivity.**

With careful handling of these and other issues, our new test method is able to cover all track discontinuity and stuck-open and stuck-closed switch faults, greatly reducing the number of test designs and reducing test cost.

A common observation is that another type of non-volatile FPGA, using one-time-programmable antifuses, can be tested without programming even once [21]. Why then do we still need any test designs at all for flash FPGAs? The answer is that mux-based routing lacks the regularity of antifuse routing. This makes our mixed approach of mostly built-in testing with a very limited number of conventional test designs less costly than supporting complete built-in testing as in antifuse FPGAs.

# 6. DISCUSSION

An integrated circuit embodying the fabric has been fabricated in a 65nm CMOS embedded flash process at UMC.

Typical static power per BLE for the high threshold-voltage implementation at 25C is 6 nW in operation, and 0.1 nW in power-down mode.

The design tools (synthesis, placement, and routing) are similar to what would be used for an SRAM-based FPGA. No flash-specific algorithms are required.

Our novel cluster input routing network reduces area while providing improved routability and good performance, at least in the context of our fabric. It may be interesting to reexamine other portions of FPGA routing, such as inter-cluster routing, to see if similar ideas can be applied.

Our testing method substantially reduced the number of test designs required to cover switch and routing track faults. This is vital for flash FPGAs. Similar techniques might be useful for other types of FPGAs, or even in other contexts such as programmable interconnect or switch network ICs.

We also hope that after becoming familiar with the benefits and requirements of non-volatile configuration memory, FPGA researchers may be motivated to invent new architectures that further exploit its advantages.

# 7. ACKNOWLEDGMENTS

Fethi Dhaoui and John McCollum were instrumental in helping us understand and apply flash technology. Kris Vorwerk and Kai Zhu provided software infrastructure and much useful expertise for architecture experiments. Vidya Bellippady, Marcel Derevlean, Dirk Kannemacher, Victor Nguyen, Bill Plants, and Nicola Telecco made key contributions to the silicon design and implementation. JJ Wang and Sana Rezgui provided references on radiation tolerance. We also thank the reviewers for their helpful suggestions.

# 8. REFERENCES

[1] Semico Research Corp. 2009. www.semico.com.

[2] Gartner Semiconductor Forecast, Nov. 25, 2009.

[3] Lattice Semiconductor Corp. 2008. *LatticeXP2 Family Data Sheet* (Feb. 2008).

[4] ProASIC Plus and ProASIC3 FPGA families, Actel Corp. www.actel.com.

[5] Huang, W., Meyer, F., Chen, X., Lombardi, F. 1998. Testing configurable LUT-based FPGAs. *IEEE Trans. VLSI Systems*, 6, 2 (June 1998), 276-283. DOI=http://dx.doi.org/10.1109/92.678888.

[6] Tahoori, M., Mitra, S. 2005. Application-independent testing of FPGA interconnects. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 24, 11 (Nov. 2005), 1774-1783. DOI=http://dx.doi.org/10.1109/TCAD.2005.852452.

[7] Dhingra, S., Garimella, S., Newalkar, A., and Stroud, C. 2005. Built-in self-test for Virtex and Spartan II FPGAs using partial reconfiguration. *Proc. IEEE North Atlantic Test Workshop*, 7-14. http://www.eng.auburn.edu/~strouce/class/bist/NATW05fpga.pdf

[8] Brown, W. and Brewer, J. 1998. *Nonvolatile Semiconductor Memory Technology*. IEEE Press.

[9] Wang, J.J., Samiee, S., Chen, H.-S., Huang, C.-K., Cheung, M., Borillo, J., Sun, S.-N., Cronquist, B., and McCollum, J. 2004. Total ionizing dose effects on flash-based field programmable gate array. *IEEE Trans. Nuc. Sci.*, 51, 6 (Dec. 2004), 3759-3766. DOI=http://dx.doi.org/10.1109/TNS.2004.839255.

[10] Lemieux, G. and Lewis, D. 2004. *Design of Interconnection Networks for Programmable Logic.* Kluwer Academic, Boston, MA.

[11] Rezgui, S., Wang, J., Sun, Y., Cronquist, B., and McCollum, J. 2008. New Reprogrammable and Non-Volatile Radiation Tolerant FPGA: RTA3P. *2008 IEEE Aerospace Conference* (Big Sky, MT, March 1-8, 2008), 1-11. DOI=http://dx.doi.org/10.1109/AERO.2008.4526472.

[12] Actel Corp. 2010. *Radiation-Tolerant ProASIC3 FPGAs Radiation Effects*. April, 2010. www.actel.com/documents/RT3P_Rad_Rpt.pdf

[13] Wang, J., Rezgui, S., Sun, Y., Issaq, F., Cronquist, B., McCollum, J., Chan, R., Pan, H., and Kabir, S. 2008. A novel radiation-tolerant floating-gate configuration cell for flash-based FPGA. *2008 IEEE Nuclear and Space Radiation Effects Conf.* (Tucson, AZ, July 14-18, 2008).

[14] Li, F., Chen, D., He, L., Cong, J. 2003. Architecture evaluation of power-efficient FPGAs. In *Proc. 2003 ACM Int'l Symp. on FPGAs* (Monterey, CA, Feb., 2003), 175-184. DOI=http://doi.acm.org/10.1145/611817.611844.

[15] Betz, V. Rose, J., Marquardt, A. 1999. *Architecture and CAD for Deep-submicron FPGAs.* Kluwer Academic Publishers, Boston.

[16] Zhu, K. 2007. Post-route LUT output polarity selection for timing optimization. In *Proc. 2007ACM Int'l Symp. FPGAs* (Monterey, CA, Feb. 18-20, 2007). 89-96. DOI=http://doi.acm.org/10.1145/1216919.1216932.

[17] Feng, W., Kaptanoglu, S., 2008. Designing efficient input interconnect blocks for LUT clusters using counting and entropy, *ACM Transactions on Reconfigurable Technology and Systems*, 1, 1 (March 2008). DOI=http://doi.acm.org/10.1145/1331897.1331902.

[18] Clos, C. 1953. A study of nonblocking switching networks. *Bell System Tech. J. 32*, 406-424.

[19] Lewis, D., Galloway, D., van Ierssel, M., Rose, J., Chow, P. 1998. The transmogrifier-2: a 1 million gate rapid-prototyping system. *IEEE Trans. VLSI Systems,* 6, 2 (June 1998), 188-198. DOI=http://dx.doi.org/10.1109/92.678867.

[20] Leventis, P., Chan, M., Lewis, D., Nouban, B., Powell, G., Vest, B., Wong, M., Xia, R., and Costello, J. 2003. Cyclone™: a low-cost, high-performance FPGA. In *Proc. IEEE 2003 Custom Integrated Circuits Conf.* (Sept. 21-24 2003). 49-52. DOI=http://dx.doi.org/10.1109/CICC.2003.1249357.

[21] Greene, J., Hamdy, E., and Beal, S. 1993. Antifuse field programmable gate arrays. *Proc. IEEE*, 81, 7 (July 1993), 1042-1056. DOI=http://dx.doi.org/10.1109/5.231343.