New Non-Volatile Memory Structures for FPGA Architectures

David Choi, *Student Member, IEEE*, Kyu Choi, and John D. Villasenor, *Senior Member, IEEE*

***Abstract*—A new set of programmable elements (PEs) using a new non-volatile device for use with routing switches and logical elements within a field-programmable gate array (FPGA) is described. The PEs have small area, can be combined with components that use low operational voltage on the same CMOS logic process, are non-volatile, enable the use of fast thin-oxide pass transistors, and are reprogrammable. A novel non-volatile flip-flop for use within the logical elements is presented as well. In combination, these methods enable programmable logic devices with improved area efficiency, the speed advantages of SRAM-based FPGAs, and a wide range of opportunities for power down strategies.**

***Index Terms*—Field-programmable gate arrays (FPGAs), non-volatile memory.**

I. Introduction

FIELD-PROGRAMMABLE gate arrays (FPGAs) have experienced dramatic increases in speed, size, flexibility, performance per unit cost, and commercial impact since the early 1990's. While many aspects of FPGA architecture have changed throughout this period, the underlying nature of FPGAs as configurable devices and the attendant need to express circuits through on-chip storage of externally supplied configuration data have remained constant. Memory technology, therefore, is central to the concept of an FPGA, and the performance characteristics of FPGAs directly reflect the memory technology used to construct them. Thus, FPGAs can be generally classified according to the type of memory structure used to store configuration data.

SRAM-based FPGAs such as those manufactured by Xilinx and Altera comprise the largest fraction of the overall market. These FPGAs utilize SRAM for expressing routing and core programmable computational functions, typically through the use of lookup tables and multiplexers. In Flash-based FPGAs such as those manufactured by Actel, all configuration and routing data are stored in Flash-based switches that retain their states even when the power is off. Anti-fuse-based FPGAs made by Actel and Quicklogic utilize one-time programmable anti-fuses to store configuration data, with programming occurring by applying a programming voltage to create a permanent connection between two interconnect wires.

Each of the previous approaches has advantages and disadvantages. Anti-fuse devices have the lowest cell size and can generally deliver fast performance, but can be programmed only once, restricting them to applications that do not require in-system reconfiguration. Flash-based FPGAs are generally slower than anti-fuse and SRAM-based FPGAs, but have the advantage that configuration data can be retained even when power is off, as well as a smaller cell size than SRAM. SRAM-based FPGAs are generally quite fast, but the volatile nature of SRAM requires that the configuration data be supplied externally at power up, often from an electrically erasable programmable read-only memory (EEPROM) chip. The configuration process can require high current, and the SRAM cells are significantly larger than the corresponding cells in anti-fuse or Flash-based FPGAs.

In the present work we present configurable circuit design approaches that utilize both non-volatile and volatile memory in the core computational and routing logic of a configurable device, thereby enabling large area savings and while enabling comparable speeds to SRAM-based FPGAs. Traditionally, this would have been unfeasible due to manufacturing constraints and current flow incompatibilities between SRAM and traditional Flash. However, recent advances enabled by the use of polysilicon-oxide-nitride-oxide-silicon (SONOS) technology in combination with select gate memory cell structures for reducing programming currents [1]–[4] have made it possible to integrate both SRAM and Flash on the same chip using a conventional CMOS logic process [5]–[7]. These developments in memory technology create the opportunity to design FPGAs offering the benefit of high performance while still retaining the ability to store data in the absence of power. The inclusion of non-volatile memory can also enable significant area savings over SRAM-only solutions as noted previously, a cost savings stemming from the reduced area, reduced mask count during manufacturing, and lower power in-system reprogrammability. In the context of applications, the presence of on-chip non-volatile memory allows selective power-down of portions of the chip during computation as well as persistence of application-specific data across overall system power-down events. Thus, in combination, these methods offer the potential to significantly change the ways FPGAs are designed and used.

The remainder of this paper is organized as follows. Section II briefly reviews the use of programmable elements (PEs) in FPGAs and establishes the foundation for the subsequent discussion of non-volatile logic. Section III discusses the non-volatile device which serves as the basis for the proposed

Manuscript received May 3, 2006; revised April 4, 2007. This work was supported in part by the Office of Naval Research under Contract N00014-06-1-0253 and by the National Science Foundation under Grant CCR-0120778 and Grant CCF-0541453.

D. Choi and J. D. Villasenor are with the Electrical Engineering Department, University of California, Los Angeles, CA 90095-1594 USA (e-mail: dschoi@ee.ucla.edu; villa@icsl.ucla.edu).

K. Choi is with the O2IC Company Ltd, Santa Clara, CA 95051 USA (e-mail: kyu@o2ic.com).

Digital Object Identifier 10.1109/TVLSI.2008.2000461

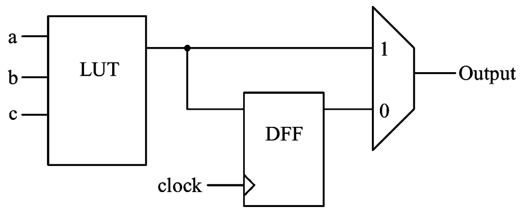

Fig. 1. FPGA Logical element with three-input LUT, flip-flop (DFF), and two-input multiplexer.

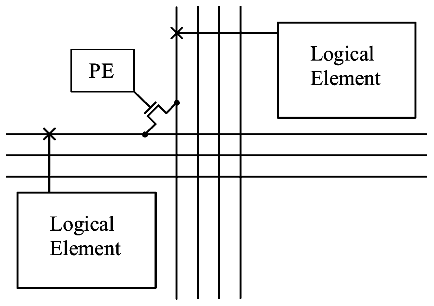

Fig. 2. Example of two logical elements connected together using a switch that is controlled by PE.

FPGA structures. Section IV describes new PEs for logical circuits and routing switches as well as a new type of nonvolatile flip-flop, and discusses several example applications. Section V addresses area reductions and the conclusion is presented in Section VI.

II. Programmable Elements in FPGAs

Programmable elements in an FPGA comprise a significant portion of the overall FPGA area and are used to configure its functionality. The two core components of an FPGA which depend on PEs are the logical elements and the routing switches that connect the logic elements together. The speed, power, area, and functionality of these components will depend on PE used. A typical logical element consists of a k-input lookup table (LUT), multiplexers, and flip-flops. Fig. 1 shows an example of generic logical element with a three-input LUT that can be used to implement any three-input function. Within the logical element, the LUT and the multiplexer both require configuration data that will determine the functionality of the logic block. For example, the three-input LUT requires eight lookup values, and the multiplexer requires knowledge of whether to select the “1” input or the “0” input. This configuration data must be supplied to the associated PEs before the logical element can be used.

Routing switches are used to connect different logic blocks together and are also controlled by PEs. Fig. 2 illustrates how two logical elements can be connected together using a switch that is controlled by a PE.

In SRAM-based FPGAs, the PE is SRAM and the switch is a thin-oxide pass transistor. An example of a conventional SRAM controlled switch is shown in Fig. 3. Depending on the value of the switch gate voltage, the switch can either allow data to pass from switch input to switch output, or break the connection between the switch input and the switch output. The switch gate voltage is controlled by the output of the SRAM, at node *mb*. The SRAM is configured with input data through the use of the word line *wl*.

The transfer speed in switching the pass transistor is relatively fast. However, there are two main drawbacks with using SRAM as the PE in an FPGA. First, when the power is cut off, the SRAM does not retain its data. Thus, the configuration of the switch must be reloaded to the SRAM through an external non-volatile device such as EEPROM when the power is turned on. Second, the large unit cell area of SRAM increases the overall chip size and the chip cost.

In Flash-based FPGAs, both the PE and the switch are combined into a single Flash-based switch, which maintains data after power-off and also has low cell area. An example of a widely used Flash-based switch (based on the Actel ProASIC design [8]) is shown in Fig. 4.

In Fig. 4, the Flash device on the left is programmed through the use of the word line, which connects to the control gate and the two sensing lines. The charge stored in floating gate is shared with the Flash transistor on the right, which acts as a programmable switch that controls the data from the switch in port and the switch out port. The Flash-based switch has a smaller unit cell size than the corresponding SRAM element, and offers data persistence as well. However, it is typically much slower than the switches used in SRAM-based FPGAs due to the higher resistance value of the Flash device. In addition, the drain voltage of a conventional Flash during the read operation is typically held to less than 1.5 V in order to avoid drain stress on the floating gate charge, which will reduce the performance of the FPGA. Yet another difference lies in the Flash process technology, which typically differs substantially from the conventional CMOS logic process and requires a higher mask count than conventional CMOS logic processes.

Thus, there exist advantages and disadvantages in speed, volatility, and area to using SRAM and Flash in the PEs associated with logical elements and routing switches. In what follows, we introduce a non-volatile device that enables new designs that simultaneously offer both the non-volatile storage and reduced unit cell area of Flash and the speed of SRAM.

III. Non-Volatile Device Structure and Performance

A. Non-Volatile Device Structure

A CMOS-based reprogrammable device such as Flash typically relies on the storage of extra charge between the gate of the device and the gate oxide as its data storage mechanism. The storage of extra charge, either in a floating gate or a nitride trap, results in a threshold voltage shift on the gate which would be higher than the threshold voltage if no charge was present. Having the threshold voltage above a certain voltage is defined as a programmed or off-state, and having the threshold voltage below a certain voltage by eliminating the charge is defined as an erased or on-state. For a conventional Flash device, a high voltage operation along with high programming currents is required to increase the threshold voltage. This high programming current is what forces Flash-based switches to have thicker gate oxides and to have a larger channel length, thereby decreasing switch performance.

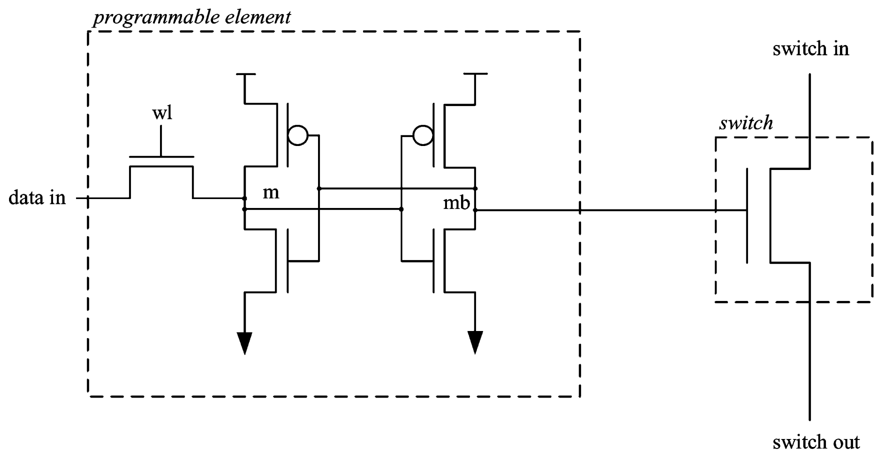

Fig. 3. Switch with SRAM as its PE.

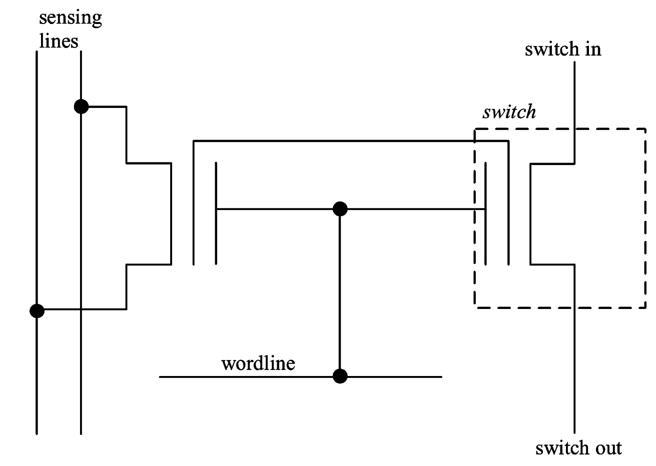

Fig. 4. Flash-based switch.

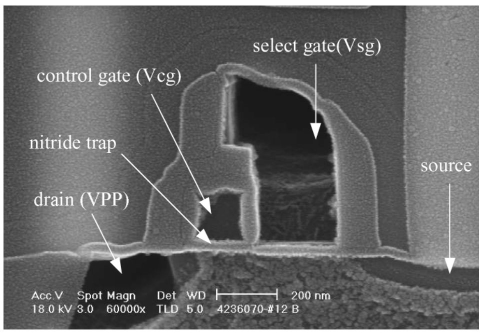
select gate(Vsg)
control gate (Vcg)
nitride trap
source
drain (VPP)
Acc.V Spot Magn Det WD 200 nm
18.0 kV 3.0 60000x TLD 5.0 4236070-#12 B

Fig. 5. Cross section of the select gate SONOS Flash device.

The importance of current and voltage compatibility can be realized by noting that conventional Flash devices utilize programming currents on the order 200–500 μA, and programming voltages 10 V and higher. By contrast, for enabling high speed at low power, low voltage devices typically require thin gate oxide thicknesses of less than 70 Å during 3.0 V operation or lower and a shorter channel length. Thus, unless a special isolation structure is built between the conventional Flash device and the thin gate oxide device, the thin gate oxide transistors will share the high voltages and high currents used in the programming and erase operations of the Flash devices. If integrated, these high programming currents and high programming voltages would cause damage to the low voltage devices directly connected to the Flash devices.

To address these shortcomings, we utilize the non-volatile Flash device presented in Fig. 5. The device is composed of a source, drain, control gate, select-gate, and body terminals. The storage element for the trapped charge is a nitride trap composed of oxide-nitride-oxide (ONO) materials underneath the control gate. When electrons are injected into the nitride trap, the charge is stored in the nitride and is isolated by the top and bottom oxides, and the threshold voltage of the control gate is shifted into a relatively higher value, thereby programming the device. When electrons stored in the nitride are injected into the substrate, the threshold voltage of the control gate is shifted toward a lower value. The difference in threshold voltage between programmed and erased devices determine the data value stored and read in the device.

The select gate acts both as a current limiter and as the mechanism for generating the high lateral electric field across the device channel. This enables the select gate to provide high efficiency source side injection of electrons into the nitride trap underneath the control gate. While select gate structures in SONOS have been studied before [1]–[4], the device used for the configurable hardware is based on an improved self-aligned structure enabling a more practical manufacturing process [5], [6]. The high efficiency source side injection of this structure provides a means of lowering the programming current down to 1.0 μA. It is also significant to note that in this structure, the source node is always fixed as a source during programming and read operations, in contrast with other select gate structures where the source node becomes switched with the drain during programming and read operations. In addition, the voltage of the source node during programming is fixed at either 0 V or supply voltage VCC, which ensures that the source node voltage will always operate with relatively low voltage. The combination of low programming current and the low voltage of the source node is one of the key factors that enables integration of the non-volatile device with thin oxide transistors. Safe integration between an array of select-gate non-volatile

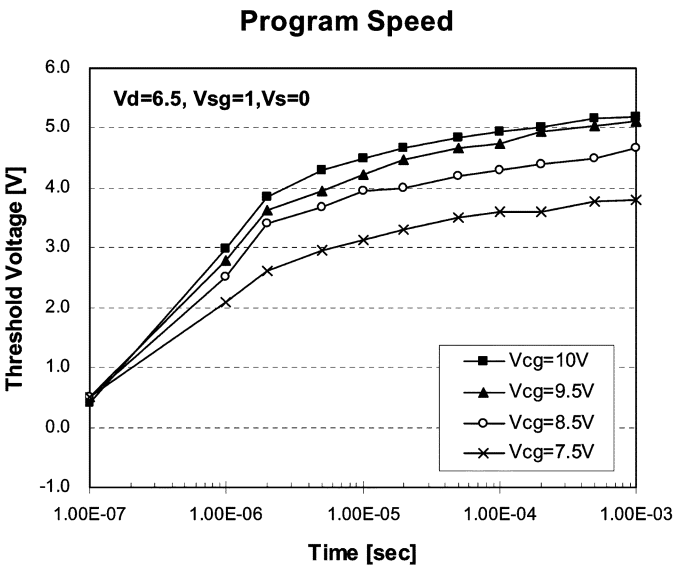

Fig. 6. Threshold voltage shift as a function of the programming time.

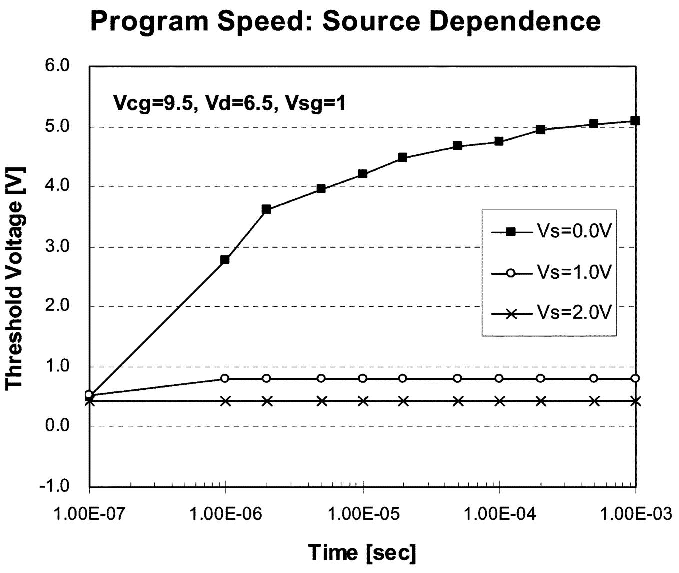

Fig. 7. Threshold voltage shift versus programming time at different source voltages.

device structures and thin-oxide SRAM along bit-lines has been demonstrated in a test chip [7].

B. Experimental Device Characteristics

To program the device, a set of programming bias conditions is required for the control gate, the select gate, the drain, and the source. The speed of the programming operation is determined by the desired level of threshold shift as illustrated in Fig. 6, which shows the experimentally measured relative threshold shift as a function of the programming time when the drain voltage (V_d) is 6.5 V, the select-gate voltage ($V_{\rm sg}$) is 1 V, and the source voltage ($V_{\rm sb}$) is 0 V.

For example, for a control gate voltage of 9.5 V and a threshold shift of 3.0 V, the programming speed of the device is 10 μs. Also, shown in Fig. 6, are different curves at various different levels of control gate voltages. This shows the relationship between higher programming voltages and faster programming speeds.

Programming of the device is strongly dependent on the source voltage, as shown in Fig. 7. If all the programming voltages are the same but the source voltage is raised to above 2 V, then the select gate (biased at 1.0 V) will not be turned on due to the fact that the select-gate to source voltage is -1.0 V which is lower than the threshold voltage of the select gate, and the device will not be programmed and no threshold shift on the control gate will occur. Thus, when the non-volatile device is integrated with other CMOS logic, the device programming can be controlled by applying a logical 0 or 3 V value to the source node.

The low programming current of the device allows many devices to be programmed at the same time and in parallel (up to 1024 devices with a 1.5 mA charge pumping capability was demonstrated in [7]), thus facilitating fast FPGA reconfiguration with relatively modest current. By contrast, for conventional Flash programmed using the hot-electron injection technique, the large currents limit the total number of simultaneous devices programmed and therefore the total number of bits programmed to 8–16 bits at the same time with a 3.0 mA charge pumping capability.

TABLE I
PROGRAM, READ, AND ERASE VOLTAGES

	Vcg	*Vsg*	*Vpp*	*source*
Program	9.5V	1V	6.5V	0V
Read	2.0V	3.0V	3.0 V	0V
Erase	-9.5V	-1.5V	floating	floating

Reading of the non-volatile device is performed by detecting the level of the threshold shift. 2.0 V is applied to the control gate, 3.0 V is applied to the select gate, and 3.0 V is applied to the drain of the device. If the device has not been programmed, then current of approximately 50 μA will flow. If the device has been programmed, then the current flow will be less than 1 μA. Erasing is accomplished using the F-N tunneling mechanism by applying -9.5 V to the control gate or 9.5 V to the bulk, and -1.5 V to the select gate. The negative voltage on the select gate is to prevent oxide breakdown between the select gate and the control gate. Over-erase problems are circumvented through the use of the select gate. Program, read, and erase voltage conditions are summarized in Table I.

IV. NEW NON-VOLATILE STRUCTURES FOR FPGAS

A. New Programmable Element for Routing Switches

We first consider a new PE for the design of high speed routing switches. The non-volatile device coupled with a word line transistor and a small pull down transistor acts as the PE for a thin gate oxide pass transistor that in turn acts as the switch, as shown in Fig. 8.

Data in this PE is non-volatile. As shown in Fig. 8, the source node of the non-volatile device is tied to the gate input of the pass transistor, and supplies the non-volatile data. If the non-volatile device is in the erased state, then applying read voltages to the non-volatile device causes a current to flow through

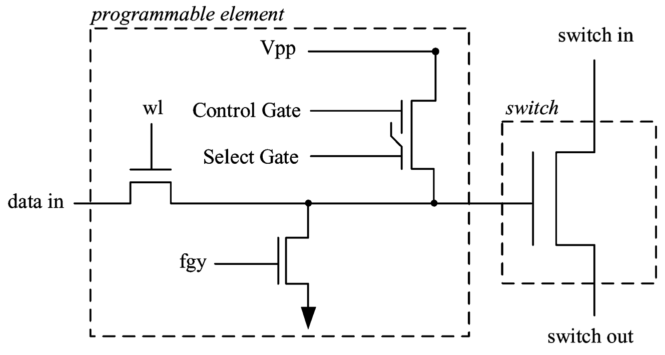

Fig. 8. Proposed non-volatile PE for routing switches.

the device. This will raise the gate voltage of the pass transistor to high, turning the switch on. If the non-volatile device is in the programmed state, no current will flow through the device. Any charge build up on the gate of the pass transistor due to leakage current from the device is pulled down through the bottom transistor (or alternatively, a diode) which is used to pull down the voltage.

Programming the non-volatile device is performed by supplying 0 or 3 V to the input, raising the word-line voltage (*WL*), applying 0 V to *fgy* (if the bottom transistor is used instead of a diode), and properly biasing the control gate (V_{cg}), select gate(V_{sg}), and drain lines (V_{pp}) according to the programming voltages. The low programming current of the device ensures that word-line transistor, the pull-down transistor, and the pass transistor will not be damaged during the programming cycle.

The total cell area of the non-volatile PE is 19.5 F^2, compared to the SRAM cell area of 130 F^2, i.e., more than a factor of 6 in area reduction. This area savings is particularly significant because silicon for routing and for programming routing configuration comprise a significant fraction of the total chip area in a conventional SRAM-based FPGA [9], for example, 70% in a Xilinx Virtex-II FPGA [10]. Furthermore, as Fig. 8 shows, the device still utilizes the same switch (right side of the figure) as SRAM-based approaches, so it achieves the same switching speed.

B. New Programmable Element for the Configuration of Logical Elements

An analogous approach can be applied to the non-volatile storage of configuration data for logical elements. Two of the new non-volatile memory devices are integrated with an nMOS latch as shown in Fig. 9. The inputs to the non-volatile memory devices consist of the control gate signal, the select gate signal, and the V_{pp} signal. The integrated non-volatile memory PE ("iNVM"), is used for LUT entries and configuration of multiplexers within a logical element.

An application of the PEs to a three-input LUT is shown in Fig. 10. The values of the LUT are supplied by eight iNVMs, while the inputs of the LUT are connected to pass transistors which connect the output of one of the iNVMs to the output.

Another application of the PE for storing configuration data is in the two-input multiplexer. The configuration of the multiplexer is controlled by the output of one iNVM. Depending on

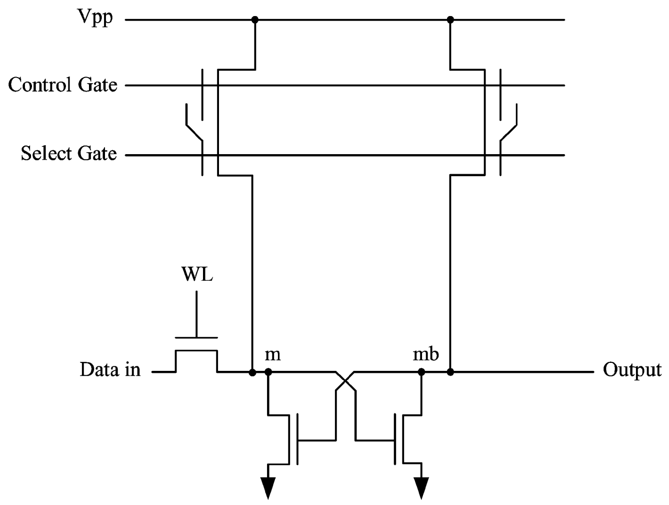

Fig. 9. PE for configuration of logical elements.

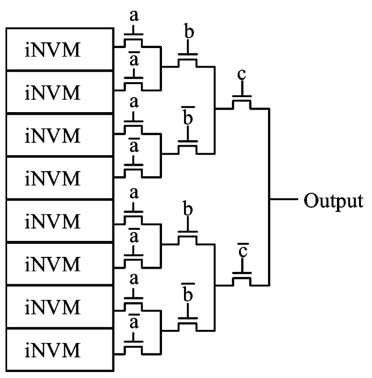

Fig. 10. Three-input LUT with integrated non-volatile memory PEs.

this value, the output of the multiplexer will reflect the values of one of the two inputs.

In order to store data onto the integrated non-volatile memory, data is placed onto the input port and the word-line is turned high. The nMOS latch is allowed to change its state to low-high or high-low. Then, programming voltages are applied to the control gate, the select gate, and the drain of the two non-volatile devices. The source nodes of the non-volatile devices are tied to nodes *m* and *mb*, which will be complementary, so the non-volatile devices will also be differentially programmed. The voltage on node *mb* also acts as the output of the PE. Note that the erase operation is done before any programming operation.

When the power is turned off, the data on the nMOS latch loses relevance. However, when the power is turned back on, the non-volatile device that is programmed will not conduct current while the non-volatile device that is erased will conduct current. This differential current is used to restore the state of the nMOS latch. This current is also used to supply and replace any charge lost through subthreshold leakage of the nMOS latch. Therefore, the PE will retain its configuration even when the power is turned off.

The total cell area is 115 F^2, which is approximately 12% less than the 130 F^2 required by a conventional SRAM latch. This difference arises because the non-volatile devices take up less area next to the nMOS latch than that of two pMOS gates; the

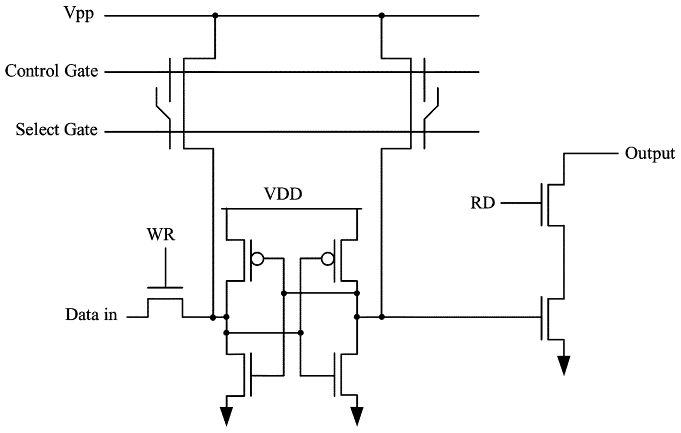

Fig. 11. Non-volatile flip-flop.

larger cell size when using two pMOS transistors is due to the spacing of $n+$ to $n-$ wells used to prevent latch up.

C. New Non-Volatile Flip-Flop

In addition to multiplexers and LUTs, a typical logical element also contains flip-flops. These flip-flops temporarily store clock-driven data and are an important part of the logic block functionality. Like SRAM, flip-flops lose their data when power is turned off. This data loss occurs not only in SRAM-based FPGAs, but also in Flash-based FPGAs, which while using Flash for some routing and computation storage, still utilize SRAM for flip-flops. The non-volatile flip-flop is shown in Fig. 11. It contains the traditional structure for a flip-flop latch in the form of an SRAM as well as two additional non-volatile devices attached to the *m* and *mb* nodes of the SRAM cell. The *WR* signal is the write control signal for the flip-flop and is typically controlled by the clock.

Operation of the flip-flop during normal operation is the same as that of a conventional flip-flop, utilizing the data-in and *WR* signals. However, in the case of an impending power failure, the latch data is used to program the two non-volatile devices. Since the latch data is stored using complementary voltage levels of 0 and 3 V at nodes *m* and *mb*, the source nodes of the non-volatile devices will also be biased with 0 or 3 V during programming. Thus, one non-volatile device is to be programmed while the other device is to remain erased, reflecting the data stored in the latch. This completes a direct transfer of data from the flip-flop to the non-volatile parts. When the power is restored, only the erased device conducts current, raising the corresponding node voltage on only one side of the SRAM latch high and thereby restoring the contents of the flip-flop to its previous state.

The possibility of storing the flip-flop states into non-volatile memory creates a number of new algorithm and application opportunities. For example, flip-flops are frequently configured to store state information in a state machine. By "remembering" the state of the state machine of the FPGA, a system can be quickly restored to its last state without having to reset the state machine. In another example, the non-volatile flip-flops can also be used to store pipelined data, which is valuable in applications with long pipelines containing critical data. Thus the non-volatile devices can be viewed as a high speed, low current "backup" memory for the flip-flop. The design of the non-volatile flip-flop cells can be applied to embedded block RAMs as well, which are frequently employed in modern FPGA designs. By combining SRAM with the non-volatile devices, these block RAMs will operate with the speed of conventional SRAM while maintaining the option to store the data into non-volatile memory as needed.

TABLE II
CELL SIZES OF THE PROPOSED PEs

	SRAM	PE for switches	PE for logical elements	Non-volatile flip-flop
Area	$135F^2$	$19.5F^2$	$115F^2$	$155F^2$

The presence of the non-volatile flip-flop and the non-volatile embedded block RAM create opportunities for power-down strategies, as some or all of the FPGA can be put into sleep mode or be powered down completely to reduce or eliminate leakage power. With the low programming current of 1 μA per device, it is possible to program up to 1024 devices with a 1.5-mA charge pump at the same time, and even more devices with a greater charge pumping capability. When power is restored, data from the non-volatile memory is used to restore the appropriate configuration information. This is performed for each cell in parallel, eliminating the need for serialized Flash to SRAM transfers.

Powering down the FPGA to reduce power consumption has previously been considered [11]; however, with the devices described here this concept can be taken one step further by storing the states of the flip-flops and the contents of the embedded block RAMs in non-volatile memory as well. Thus, the FPGA can be either locally or globally powered down when it will be known that the FPGA, or some portion of it, will not be used for a given period of time. The specific strategies with respect to which portions of an FPGA to power down, and when, will be highly application dependent, and offer a rich range of design choices and tradeoffs.

One example scenario of powering down the FPGA to reduce power consumption is template matching. Consider a pattern matching application in which a set of 50 template images each of size 128 by 128 with 8 bits/pixel are captured, stored in block RAM memory, and processed using FPGA circuitry in response to events that occur only sporadically, but that are important when they do occur. Such scenarios are common in security and other environmental monitoring applications. Since standby power of FPGAs can be hundreds of milliwatts even in low power designs [12], it is advantageous to consider transfer of template data to non-volatile memory to allow local or global power down during the time between events of interest. In conventional FPGAs, the time and energy costs of restoring the template data are significant. For example, transferring 50 template images from the block RAMs of a traditional FPGA to external Flash memory would take over 3.7 s given typical Flash programming times of 9 μs per 16-bit word. The associated energy cost of programming and reading the Flash chip is 170 mJ total (assuming active current of 15 mA), and there are additional energy costs associated with powering down and powering up the FPGA. Assuming an FPGA standby power of 0.228 W (this is

TABLE III
COMPARISON OF TOTAL FPGA DEVICE AREA WHEN SRAM VERSUS NON-VOLATILE PES ARE USED. N AND k GIVE THE NUMBER OF LOGICAL ELEMENTS PER LOGIC BLOCK AND LOOKUP PER LOGICAL ELEMENT, RESPECTIVELY. AREA FIGURES UTILIZE THE MINIMUM WIDTH TRANSISTOR AREA METRIC AS DESCRIBED IN [14]. LOGIC AREA COMPRISES ALL NON-PROGRAMMABLE SILICON ON THE CHIP, INCLUDING THE NON-PROGRAMMABLE SILICON DEVOTED TO ROUTING. TOTAL DEVICE AREA IS THE SUM OF THE LOGIC AREA AND THE AREA CONSUMED BY THE PES. PE AREA INCLUDES AREA FROM PES USED IN BOTH ROUTING AND LOGICAL ELEMENTS

Architecture (N,k)	Logic Area (10^3)	Total No. of PEs (10^3)	SRAM PE		Non-volatile PE		Area reduction
			SRAM Area (10^3)	Total Area (10^3)	PE Area (10^3)	Total Area (10^3)	
(8,7)	6,809	649	4,733	11,541	684	7,492	35%
(6,7)	6,156	622	4,534	10,690	655	6,811	36%
(6,6)	6,691	470	3,423	10,114	494	7,186	29%
(10,5)	7,066	374	2,728	9,794	394	7,460	24%
(12,4)	6,860	317	2,314	9,174	334	7,194	22%

the cost for a Virtex-4), the minimum time the FPGA must be powered off before the energy costs due to the transfer are recovered is 0.74 s. In other words, the FPGA must be powered off for at least 0.74 s to see a savings in energy.

Performing the same data transfer using the proposed non-volatile flip-flop structures, however, results in a total programming time of 64 ms, which is faster due to the greater number of devices that can be programmed directly and in parallel (1024 at a time). The associated energy cost of loading the non-volatile devices with the flip-flop data is 0.3 mJ, which is due to the 1-μA programming current per device, and gives a savings of approximately three orders of magnitude over the traditional approach. Flip-flop data is loaded from the non-volatile cells as the flip-flop supply voltage is restored during power-up, which removes the necessity to perform a read operation as an extra step. Again using the same FPGA standby power of 0.228 W, the minimum time the FPGA must be powered off before the energy costs are recovered is less than 1 ms, which is smaller by approximately two orders of magnitude than the 0.74-s threshold in the traditional approach. In short, utilizing non-volatile flip-flops to store template information significantly decreases the time and energy costs of data transfers, which decreases the minimum power-down required time to recover the cost and provides for potentially shorter power-down cycles. This in turn leads directly to greater energy savings because, among other things, the use of integrated non-volatile memory enables the FPGA to be powered down when the traditional FPGA would need to be in standby mode. The ability to choose how and when to power down the FPGA in such a fine-grained manner opens up new opportunities for power savings within the context of how and when data is sampled and how template matching is performed.

Another example scenario in which the FPGA can leverage non-volatile storage during power-down is a system using one or more adaptive filters for acoustic echo cancellation [13] in which the coefficients of each filter are stored in flip-flops of the FPGA. The filter coefficients are updated using the least-mean-squares (LMS) algorithm, which converge after a training period. For conventional FPGA structures, it is often not practical to extract the coefficients of the filter from the flip-flops, store them in non-volatile memory, and restore the contents of the flip-flops. In contrast with the template matching scenario in which data is stored in easily accessible block RAMs, powering down the FPGA in this scenario may not be viable without losing the coefficient values. However, since the proposed non-volatile flip-flop structure allows for direct transfer of data from the flip-flop state to non-volatile memory, the filter coefficients can be readily preserved. This enables further energy savings: while the FPGA is powered off, energy is conserved when otherwise a standby power would have been required to maintain the filter tap values. The adaptive filter, using the appropriate filter coefficients, resumes operation when the power is restored without the need to retrain the filter coefficients.

V. AREA ESTIMATION

Table II lists the cell sizes for the PEs described before. As noted earlier, a significant portion of the FPGA chip area is consumed by the programming elements used for storing routing information and configuration data. Thus, a critical question concerns the impact of the area results in Table II on overall FPGA size.

In a conventional SRAM-based FPGA, the total number of SRAM cells used is directly related to the number of logical elements N in a logic block and the size k of the LUT in each logic element [14]. This is in large part because the functionality of the logic block impacts the routability and thus the area efficiency as well [15]. Using the estimated number of SRAM cells for each of the "dominant" (N,k) architectures for baseline FPGAs in [14], the area savings enabled by the non-volatile PEs can be computed for a wide cross section of conventional FPGA designs. These results are given in Table III, and are computed based on the assumption that the area of the non-volatile PE used for routing serves as the lower bound on the PE cell size. Table III shows that significant area savings can be realized.

VI. CONCLUSION

We have presented a set of new designs for core FPGA logic, routing, and flip-flop circuits that offer the speed advantages of SRAM and the area savings and non-volatility advantages of Flash. These designs are enabled by a non-volatile device that has the advantage of a low programming current that will not damage low voltage switching devices. FPGA architectures using the core FGPA elements described here offer the potential to dramatically improve power efficiency, area, and cost, and enable a whole new class of adaptive, power-aware application mapping methods. Examples have been presented showing area savings ranging from 22% to 36% by replacing SRAM based PEs with the proposed non-volatile structures. In addition, we have described several application examples in which this approach enables new power-down strategies.

REFERENCES

[1] K.-T. Chang, W.-M. Chen, C. Swift, J. M. Higman, W. M. Paulson, and K.-M. Chang, "A new SONOS memory using source-side injection for programming," *IEEE Electron Devices Lett.*, vol. 19, no. 7, pp. 253–255, Jul. 1998.

[2] Y. K. Lee, B. Y. Choi, J. S. Sim, K. W. Song, J. D. Lee, B.-G. Park, D. Park, and C. Chung, "A highly scalable split-gate SONOS flash memory with programmable-pass and pure-select transistors for sub-90-nm technology," in *Proc. Int. Conf. Solid-State Devices Mater.*, Sep. 2004, pp. 252–253.

[3] M. Rosmeulen, J. Van Houdt, L. Haspeslagh, and K. De Meyer, "Scanrom, a novel non-volatile memory cell storing 9 bits," in *IEEE Symp. VLSI Tech. Dig.*, Jun. 2004, pp. 74–75.

[4] T. Ogura, N. Ogura, M. Kirihara, K. Park, Y. Baba, M. Sekine, and K. Shimeno, "Embedded twin MONOS flash memories with 4 ns and 15 ns fast access time," in *IEEE Symp. VLSI Circuits Dig.*, Jun. 2003, pp. 207–210.

[5] K. H. Choi, "Non-volatile memory device," U.S. 6 965 145, Nov. 15, 2005.

[6] K. H. Choi, "Method of manufacturing self-aligned non-volatile memory device," U.S. 6 972 229, Dec. 6, 2005.

[7] D. Choi, E.-P. Kwon, H. Lee, J. Chang, K. Choi, and J. Villasenor, "Single-chip integration of SRAM and non-volatile memory using bit-line sharing," in *Proc. Eur. Solid-State Circuits Conf.*, Sep. 2006, pp. 295–298.

[8] Actel, "ProASIC3 Flash family FPGAs datasheet, Ver. 0.6," 2006, White Paper.

[9] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Boston, MA: Kluwer, 1999.

[10] A. Gayasen, N. Vijaykrishnan, and M. Irwin, "Exploring technology alternatives for nano-scale FPGA interconnects," in *Proc. Des. Autom. Conf.*, Jun. 2005, pp. 921–926.

[11] S. Mohanty and V. K. Prasanna, "Duty cycle aware application design using FPGAs," in *Proc. IEEE Symp. Field Program. Custom Comput. Mach.*, Apr. 2004, pp. 338–339.

[12] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90 nm low-power FPGA for battery-powered applications," in *Proc. Int. Symp. Field Program. Gate Arrays*, Feb. 2006, pp. 3–11.

[13] D. J. Allred, W. Huang, V. Krishnan, H. Yoo, and D. V. Anderson, "An FPGA implementation for a high throughput adaptive filter using distributed arithmetic," in *Proc. IEEE Symp. Field Program. Custom Comput. Mach.*, Feb. 2004, pp. 324–325.

[14] Y. Lin, F. Li, and L. He, "Circuits and architectures for field programmable gate array with configurable supply voltage," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 9, pp. 1035–1047, Sep. 2005.

[15] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate array: The effect of logic block functionality on area efficiency," *IEEE J. Solid-State Circuits*, vol. 25, pp. 1217–1225, Oct. 1990.

David Choi (S'05) received the B.S. and M.S. degrees in electrical engineering from the University of California, Los Angeles, in 2002 and 2003, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include non-volatile memory, reconfigurable computing, communications, and video image processing.

Kyu Choi received the B.S. and M.S. degrees in physics from Seoul National University, Seoul, Korea, in 1969 and 1974, respectively, and the Ph.D. degree in solid-state physics from the University of Oregon, Eugene, in 1978.

He is currently at O2IC, Santa Clara, CA, where he is focusing on the integration of volatile and non-volatile memory in the same memory cell. He has held technical and management positions at Intel, Signetics, Synertek-Honeywell, Samsung Electronics, and Soft Device in various areas of device and process technology. He holds numerous patents in semiconductor memories.

John D. Villasenor (SM'97) received the B.S. degree from the University of Virginia, Charlottesville, in 1985, the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, in 1986 and 1989, respectively, all in electrical engineering.

From 1990 to 1992, he was with the Radar Science and Engineering Section, Jet Propulsion Laboratory, Pasadena, CA, where he developed methods for imaging the earth from space. In 1992, he joined the Electrical Engineering Department, University of California, Los Angeles (UCLA), where he is currently a Professor. He served as Vice Chair of the Department from 1996 to 2002. At UCLA, his research efforts lie in communications, computing, imaging and video compression, and networking.