A Flash-based Digital Circuit Design Flow

Monther Abusultan Sunil P. Khatri

ECE Department, Texas A&M University, College Station, TX 77843.

ABSTRACT

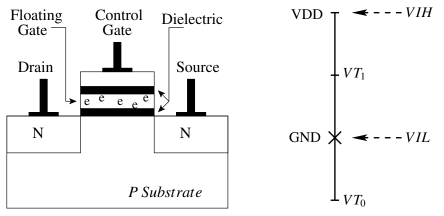
Traditionally, floating gate (flash) transistors have been used exclusively to implement non-volatile memory in its various forms. Recently, we showed that flash transistors can be used to implement digital circuits as well. In this paper, we present the details on the realization and characteristics of the block-level flash-based digital design. The current work describes the synthesis flow to decompose a circuit block into a network of interconnected FCs. The resulting network is characterized with respect to timing, power and energy, and the results are compared with a standard-cell based realization of the same block (obtained using commercial tools). We obtain significantly improved delay ($0.59\times$), power ($0.35\times$) and cell area ($0.60\times$) compared to a traditional CMOS standard-cell based approach, when averaged over 12 standard benchmarks. It is generally rare that a circuit methodology yields results that are better than existing commercial standard-cell based flows in terms of delay, area, power *and* energy, and in this sense, we submit that our results are significant. Additional benefits of a flash-based digital design is that it allows for precision speed binning in the factory, and also enables in-field re-programmability (we note that our flash-based design is *not* an FPGA, but rather an ASIC style design) to counteract the speed degradation of a design due to aging. These benefits arise from the fact that the threshold voltage of flash devices can be controlled with precision.

Keywords: Flash-based Circuits; Logic Synthesis.

1. INTRODUCTION

Flash transistors are effectively dual-gate field effect transistors (FET) devices. The two gates are referred to as a *control gate* and a *floating gate*. The control gate is similar to a regular transistor's gate both physically and functionally. The floating gate is buried within the device structure, between the substrate and the control gate. The floating gate is never contacted, and hence it is never driven directly. Figure 1(a) shows the cross section of a flash transistor. The phenomenon by which electrons tunnel through a barrier is referred to as Fowler-Nordheim (FN) tunneling [1]. In our approach, we only utilize two threshold voltages (the erase threshold voltage (VT_0 in Figure 1(b)) and a program threshold voltage (VT_1 in Figure 1(b)) to implement digital circuits. Note that VT_1 is approximately $VDD/2$.

Programming a flash transistor is performed by holding the bulk, source and drain terminals at ground and applying a high voltage (10-20 Volts) to the control gate terminal of the flash transistor. This creates an electric field that forces electrons to tunnel from the substrate into the floating gate, which causes an increase in the threshold voltage of the flash transistor. The value of the resulting threshold voltage is dependent on the number of electrons that tunnel into the floating gate, which is in turn dependent on the duration of the programming pulse. Once electrons are trapped in the floating gate, they remain trapped



(a) Cross Section (b) V_T levels

Figure 1: Floating Gate (Flash) Transistor Structure and V_T Levels

for several years [2, 3], or until removed by an erase operation. A flash transistor is erased by holding the control gate voltage at ground, floating the drain and source terminals, and applying a high voltage at the bulk of the transistor. This results in an electric field that forces the electrons to tunnel from the floating gate back into the substrate. Whereas programming can be done for a specific transistor by driving the programming voltage to its control gate, erasing is performed on many transistors at once (all the flash transistors that share the same bulk node are erased at once, provided their drain, source and gate are connected as mentioned above).

In this paper, we briefly describe the circuit structure of the FC, while omitting finer electrical details regarding programming and erasing of the transistors of the FCs. This is done so as to familiarize the reader with our flash-based design approach.

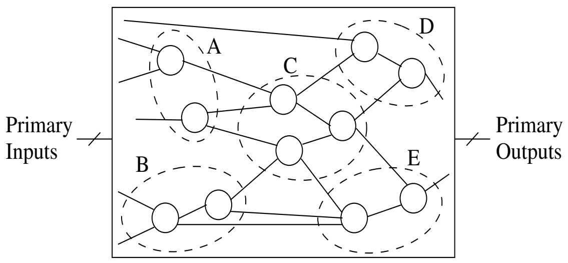
In this paper, we show through the circuit simulations, that our flash-based digital circuit design approach, yields significantly improved delay ($0.59\times$), power ($0.35\times$) and area ($0.60\times$) characteristics compared to a traditional CMOS standard cell-based approach, when averaged over 12 standard benchmark designs.

It is very important to note that the proposed flash-based design is *not* an FPGA-like reprogrammable structure. Rather, it targets an ASIC or custom/semi-custom design flow. The FC structure is not fully programmable because the metalization of the design is fixed (i.e. interconnects are hardwired and not adjustable after fabrication). Also, we note that the flash fabrication process is inherently compatible with the CMOS fabrication process. In fact, flash memories use both flash and CMOS transistors simultaneously on the same die [4]. In flash memories, the CMOS devices are used for the controller, addressing logic, driver logic, and analog functions such as program voltage pulse generation. Our work assumes that both flash and CMOS devices are present on the same die.

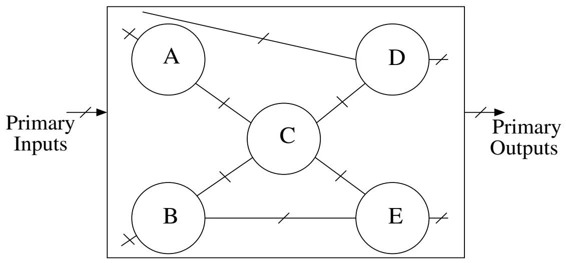
The market need for dense and compact flash memory has fueled the advancement in flash technology. The technology node of flash devices has recently been able to track the CMOS technology node. Currently memory devices are being fabricated at sub-20nm minimum feature dimensions similar to CMOS technology node [5, 6, 7, 8]. In this work, we only explore our design approach with a 45nm technology to ensure that our results are realistic. Electrical characteristics for flash transistors at lower technology nodes are not easily available.

Our flash-based digital design flow has some key advantages over standard cells. The ability to perform fine adjustments to the device threshold voltage will allow the manufacturer to perform fine grained speed binning at the factory. Also, as the IC ages, and the transistors slow down as a result, another round of fine grained threshold voltage adjustment can be performed, to bring the IC performance back within specifications. In addition, by leaving a portion of flash devices unprogrammed, our circuit has the ability to support post-manufacturing engineering change orders (ECOs). This is not possible in present day

ICCAD '16, November 07-10, 2016, Austin, TX, USA
Copyright 2016 ACM. ISBN 978-1-4503-4466-1/16/11 ...$15.00
DOI: http://dx.doi.org/10.1145/2966986.2966990



(a) Technology Independent Logic Circuit



(b) Flash-based Digital Circuit

Figure 2: Converting a Binary Netlist into a Flash-based Design

CMOS technology, since ECOs in CMOS technology require metal mask changes, which can be expensive. Also, in present day CMOS, speed binning is a function of the process variations. Finally, in-field performance adjustment (to counteract aging problems) is not possible in present-day CMOS designs.

Flash transistors have a finite number of write cycles (1K to 100K) [2, 3], which is an issue for flash memory devices. However, this will not be an issue when using flash transistors to implement digital circuits. This is because the number of write cycles needed to realize the desired digital circuit will be very limited (a handful at most). In our approach, the flash devices will be programmed after fabrication (in the factory), and then again to possibly adjust for aging effects (in the field).

The key contributions of this paper are:

- To the best of the authors' knowledge, this is the first paper to present a flash transistor-based synthesis flow to implement digital circuits.
- The electrical behavior of the flash-based circuit blocks realized by our flow are characterized using circuit level simulations, and we demonstrate significantly improved delay, area, power and energy characteristics when compared to a commercial CMOS standard cell based approach.
- The reasons for this improvement are an extremely dense layout and reduced gate capacitance.
- Our approach supports fine-grained in-factory programmability, allowing the manufacturer to perform precise speed binning.
- By re-programming the threshold voltages of the transistors in-field, our scheme provides the ability to mitigate effects like aging.
- With the flash-based digital design flow, engineering change orders (ECOs) can be achieved easily, because flash transistors are re-programmable post-manufacturing. In CMOS designs, ECOs invariably require metal mask changes.

The remainder of this paper is organized as follows. Section 2 discusses some previous work in the area of flash technology. Section 3 describes our flash-based circuit design flow, while Section 4 reports the results of experiments we performed to compare our design with a standard-cell based equivalent. We conclude our paper in Section 5.

2. PREVIOUS WORK

There have been several research efforts which study the flash devices and their use in memory. A short list includes [9, 10]. These papers report details of flash devices and their characterization. However, they do not describe the use of flash transistors for logic circuits.

A good deal of work in flash has been reported in the area of architectural techniques to increase flash memory endurance. Some representative works include those on wear leveling techniques, which are used in flash-based memory blocks [11], to compensate for the fact that flash transistors typically have a finite (10k - 100k) number of times they can be written [2, 3]. In traditional flash memory, *wear leveling* is performed at the architectural level to spread the wear of the cells.

The authors of [12] present a flash-based design approach to implement ternary-valued logic circuits. The use of ternary-valued logic in [12] is aimed towards reducing the number of transistors connected in series in each NAND stack. However, the use of multiple threshold voltages results in reduced V_{gs} values, which results in increased delays. Another factor that contributes to the increased delays in [12] is the large number of transistors in each NAND stack (8 flash transistors for a 4-input ternary-valued function). In contrast, our implementation uses 6 flash transistors in series for a 6 input binary-valued function, which is a reduction of $2\times$ for each input. We use single-level cells (SLC) cells instead of multi-level cells (MLC) cells (which were used in [12]) to implement our flash-based digital circuits. This makes our design more immune to read and write disturbs. The work in [12] reported a $3\times$ delay penalty compared to a CMOS standard cell-based implementation. In our work, we achieve improved delays at the cell-level (16% faster) as well as the block-level (41% faster) compared to a CMOS standard cell-based implementation.

In [13], the authors exploit the ability to control the threshold voltage of flash transistors to implement binary-valued digital circuits. The circuit topology utilized is a cluster of unprogrammed flash transistors arranged in a NAND configuration (*Flash Clusters (FCs)*), which are programmed in the factory to implement the desired logic function. The work in [13] overcomes the delay hit shown in [12] and achieves a delay improvement (16% faster) when compared to CMOS standard cell-based implementation. The improvement in [13] is mainly due to limiting the number of V_T levels to 2 instead of 3 V_T levels (in [12]). The work of [13] only describes the design, electrical details and circuit-level characterization results for an FC. An FC implements a logic function of a small number of inputs (up to 6 in [13]) and outputs (up to 3 in [13]). In contrast, the current paper focuses on the design of large circuit blocks which are comprised of 1000s of interconnected FCs.

To the best of our knowledge, there has been no work prior to the current paper which describes the synthesis, mapping and electrical characterization of digital circuits implemented as a network of flash-based circuit elements (FCs).

3. APPROACH

3.1 Overview

In this section we first discuss the top level flow that we use in this paper to convert a technology-independent digital circuit design into an equivalent (dynamic) flash-based digital circuit design (Section 3.2). The resulting flash-based design consists of several flash clusters (FCs). Each FC implements an (up to) n-output function with up to m inputs. The circuit structure of an FC is briefly described in Section 3.3. In our implementation, we chose the number of outputs ($n \leq 3$), and the number of inputs ($m \leq 6$). Each FC consists of several Flash Logic Arrays (FLAs), which in turn are made up of several Flash Logic Bundles (FLBs). A working overview of these components will be discussed in Section 3.3.

3.2 Flash-based Design Conversion

Figure 2 illustrates the conversion of a technology-independent digital circuit into a flash-based digital circuit. Starting with a technology-independent logic circuit (Figure 2(a)), we cluster the circuit nodes (in the dotted circles of Figure 2(a)). The input to this conversion step can alternately be technology dependent as well. The resulting clusters are multi-input, multi-output structures (with up to m inputs and n outputs), and are shown as solid circles in Figure 2(b). The solid circles in Figure 2(b) are referred to as Flash Clusters (FCs). The flash-based digital circuit implements the logic function of each cluster (dotted circles in Figure 2(a)) as an FC (solid circle in Figure 2(b)). In other words, each FC implements an up to n-output logic function of up to m inputs. We refer to this function as $F_{m,n}$. The solid circles labeled A, B, C, D and E in the flash-based digital circuit of Figure 2(b) have the same functionality and connectivity as the dashed ovals A, B, C, D and E in the technology independent digital circuit of Figure 2(a). The design of the FC is briefly discussed in the next section.

Note that this paper focuses on flash-based implementation of a digital design using an interconnected network of FCs. We describe the synthesis, mapping and electrical characterization of the resulting design, and compare the delay, area, power and energy with a CMOS standard-cell based realization of the same design (obtained using commercial tools). We touch upon the design of the FC only to the extent that it makes the rest of the paper comprehensible.

3.3 Flash Cluster Circuit Design

In our flash-based digital design flow, we construct the flash-based digital netlist by first identifying clusters of nodes in the technology independent design. Each cluster implements a logic function $F_{m,n}$ with n-outputs and m inputs. In particular, $n \leq 3$ and $m \leq 6$, for electrical reasons. Each such cluster is then mapped and implemented as an FC. An FC is a generic circuit structure that is capable of implementing any logic function with m inputs and n outputs ($F_{m,n}$). FCs are also equipped with the required logic for programming the threshold voltages of their floating gate devices. In this paper, we omit details about programming, on account of brevity.

Figure 3 shows the block diagram of any FC. The FC is a dynamic circuit, and is precharged during the low phase of the *clk* signal.

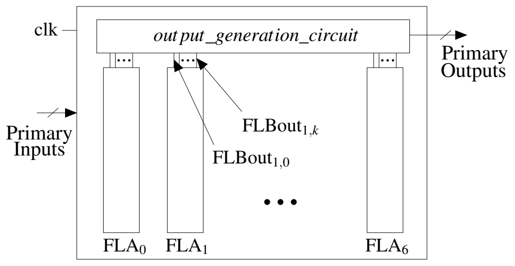


Figure 3: Flash Cluster (FC)

Internally, the FC consists of multiple *flash logic arrays* (FLAs) and an *output generation circuit* as shown in Figure 3. An FLA is a group of NAND flash-like pulldown stack structures. Each pulldown structure implements a logic cube of $F_{m,n}$. Each FLA (there are $2^n - 1$ FLAs in all in every FC) implements a group of input cubes that correspond to an output minterm of $F_{m,n}$. For example, if the FC output $< 010 >$ has 2 input cubes $< 011011 >$ and $< 110-11 >$, then FLA_2 implements these two input cubes. In other words, only one FLA output pulls down when any input is applied to the FC. For the i^{th} FLA, we refer to the number of cubes that this FLA implements as its *cubes per array* or CPA_i. In the example of this paragraph, $CPA_2 = 2$. The outputs of the FLAs are connected to the output generation circuit in the FC such that when the output of FLA_i pulls down, the output generation circuit produces the n-bit output vector i. For example, if $n = 3$ and if FLA_5 pulls down during evaluation, then the output generation circuit produces the 3-bit output $< 101 >$. Note that the FC is a dynamic circuit, so its default (precharged) output state is $2^n - 1$ (or $< 111 >$ for $n = 3$). Hence we do not need to implement $FLA_{(2^n-1)}$, and only need to implement $2^n - 1$ FLAs (shown as $FLA_0, FLA_1, \cdots, FLA_6$ in Figure 3, for $n = 3$).

3.3.1 Flash Logic Array

Each FLA consists of multiple flash logic bundles (FLBs). The number of FLBs that exist in FLA_i is $\lceil \frac{CPA_i}{CPB} \rceil$, where CPA_i is the number of cubes in FLA_i, and CPB is the maximum number of cubes that can be implemented in any single FLB. Notice that while CPA_i is determined by the logic function $F_{m,n}$, CPB is an electrical parameter that can be optimized to improve circuit delay, power, energy and physical area. A large CPB means increased capacitance and delay, so the choice of CPB is an important variable. The optimal value of CPB was found to be 3. The number of outputs of FLA_i is equal to the number of FLBs in FLA_i, namely ($\lceil \frac{CPA_i}{CPB} \rceil$). Exactly one of the FLB outputs ($FLBout_{i,k}$) is pulled down when an input combination is applied to the FLA inputs.

3.3.2 Flash Logic Bundles

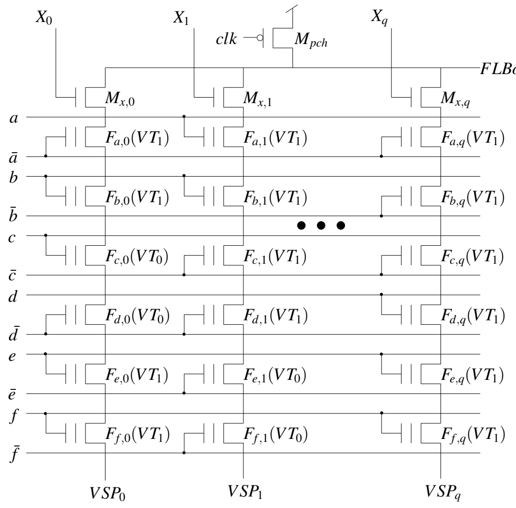


Figure 4: Flash Logic Bundle i, k (FLB$_{i,k}$)

Figure 4 shows the circuit structure of a flash logic bundle (FLB). An FLB consists of a number of NAND flash-like pulldown structures (stacks) that share the same output. Each pulldown stack implements one cube of $F_{m,n}$. The maximum number of NAND flash like pulldown stacks in each FLB is CPB, as described earlier.

Each pulldown stack has m flash transistors and 1 regular NMOS transistor (as shown in Figure 4), where m is the number of inputs of the function $F_{m,n}$. The flash transistors are programmed to implement cubes of $F_{m,n}$. The shared regular PMOS transistor shown at the top of Figure 4 (M_{pch}), serves as the precharge transistor for all pulldown structures of the FLB. It pulls up $FLBout_{i,k}$ (the k^{th} FLB of the ith FLA) during the precharge (low) phase of the clock signal (clk). The lines VSP_0 to VSP_q are connected to ground during operation, to allow the NAND stacks to pull down when they evaluate. They are also utilized during the programming operation of the flash NAND stack. The regular NMOS transistors (M_x^i) shown in Figure 4 serve dual purposes. During regular operation they are used as the evaluate transistors which are off during the precharge (low) phase of clk and only turn on during the evaluate (high) phase of clk, to allow the pulldown stack to evaluate the output $FLBout_{i,k}$. The NMOS transistors (M_x^i) are also utilized during the programming operation of the NAND flash stack.

The left-most stack of Figure 4 implements the cube $\overline{a}bef$. Note that transistors $F_{c,0}$ and $F_{d,0}$ are programmed to a threshold voltage VT_0 (erase threshold) which is below GND (see Figure 1(b)). Therefore, these two transistors are on irrespective of the values of the signals c and d. Now, transistors $F_{a,0}, F_{b,0}, F_{e,0}$ and $F_{f,0}$ are programmed to a threshold voltage VT_1, which is between VDD and GND (see Figure 1(b)). This means that these devices turn on only when their gate signals (respectively $\overline{a}, b, e$ and f) are greater than VT_1. As a consequence, the left-most stack of Figure 4 implements the cube $\overline{a}bef$.

3.3.3 Output Logic

Each FLA in the FC drives an output generation circuit that generates the final outputs of the FC, as shown in Figure 3. Figure 5 shows the circuit of the output generation circuit. Each output of the function $F_{m,n}$ is driven by an output buffer (sized to drive a fan-out of 3 FC input loads). The unbuffered outputs of the output logic are represented using a horizontal line which is precharged by a PMOS transistor (shown at the left side of the output line). The precharge PMOS transistor is driven by the clock signal (*clk*). Each of the unbuffered output lines is pulled down based on which FLA_i output ($FLBout_{i,k}$) is pulled down. Note that exactly one $FLBout_{i,k}$ pulls down for any applied input to the FC. Since the outputs of the FLAs are active low, we insert an inverter for each $FLBout_{i,k}$ before driving the gate of pulldown NMOS devices in the output logic.

For example, if any of $FLBout_{0,i}$ pull down, then all three f,g and h will pull down. If any of $FLBout_{3,i}$ pull down, on the other hand, then the output needs to be $< f,g,h > = < 011 >$, assuming that f is the most significant output bit. In this case, there will be NMOS devices pulling down the output of f in the output logic, and no NMOS devices connected to g and h, which will stay precharged, resulting in the output minterm $< f,g,h > = < 011 >$ being produced. Finally, the output minterm $< 111 >$ does not need to be produced, and therefore all input cubes mapping to the output minterm $< 111 >$ are never implemented.

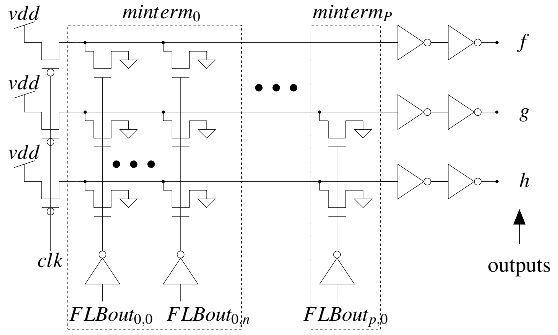


Figure 5: Flash Output Generation Circuit

3.4 FC-based CAD Flow

The CAD flow to convert an input logic netlist is described next. The input logic netlist is technology independent in our experiments, but it could be technology dependent as well. There are several steps in the flow, which are briefly described next, and then explained in detail.

First, the input netlist is clustered into FCs (where FC_i implements $F^i_{m,n}$), with a goal of minimizing the wiring between FC's. In our experiments, $m \leq 6$ and $n \leq 3$. After this, we obtain a multi-level netlist of interconnected FCs.

Next, the layout of each FC is generated. The FCs, FLAs and FLBs are extremely regular in their physical characteristics, making them amenable to the on-the-fly physical synthesis flow that we use. Based on the fanout load of the i^{th} output of FC_j, additional buffers are added for that output.

To quantify the utility of our flash-based circuit design flow, the same input netlist is synthesized and mapped using commercial standard-cell based CAD tools. The resulting designs (flash-based and standard-cell based) are compared in terms of their delay, area, power and energy, over a number of designs.

3.4.1 FC-based Clustering

Problem Definition: Given an arbitrary logic netlist η, cluster η into a multi-level network η* of FCs, subject to the following constraints:
• the network η* is acyclic.
• each $FC_i \in \eta^*$ has a logic function $F^i_{s,t}$ where $s \leq m$ and $t \leq n$.

Algorithm 1 outlines our clustering strategy. We first decompose η into a network of nodes with at most p inputs. If this was not done, we could encounter a situation where the number of inputs to some node in η is greater than m, making it impossible to create the multi-level FC-based netlist. We choose $p < m$, and in particular we found that $p = 3$ yielded good results. Now η is sorted in a depth-first manner. The resulting array of nodes is sorted in *topological*[1] order, and placed into an array L.

[1]Primary inputs are assigned a level 0, and other nodes are assigned a level which is one larger than the maximum level of all their fanins

Algorithm 1 Clustering a logic netlist into a multi-level network of FCs

```
η = decompose_network(η, p)
L = dfs_and_levelize_nodes(η)
FC* = 0
η* = 0
while get_next_element(L) != NIL do
  FC* = FC* ∪ get_next_element(L)
  if (num_input(FC*) ≤ m) && (num_output(FC*) ≤ n) then
    continue
  else
    Q = remove_last_element(FC*)
    η* = η* ∪ FC*
    FC* = Q
  end if
end while
η* = wiring_recovery(η*)
```

Now we greedily construct the logic in each FC, by successively grouping nodes from L such that the resulting implementation of the grouped nodes FC^* does not violate the input or output cardinality constraints for the FCs. If so, we attempt to include another node into FC^*, otherwise we append the last FC satisfying the constraints to the result η^*.

In order to reduce the wiring between FCs, the *get_next_element* routine preferentially returns nodes in the fanout of the nodes of FC^*, provided that the inclusion of such a node into FC^* would not result in a cyclic dependency between the FCs of η^*. If such nodes are not available, the first un-mapped node from L is returned. At every step of the construction of η^*, we verify that the graph induced by the multi-level network of FCs is acyclic.

After the clustering step is completed, we invoke a procedure called *wiring_recovery*. This is a final effort in reducing the wiring between FCs. This procedure attempts to move individual nodes in L to a different FC than their currently assigned FC. If a wiring gain is realized by such a move, the move is made. If no more nodes can be gainfully moved, or if a specified number of iterations have been made through L, the procedure returns. On average, the *wiring_recovery* procedure is able to reduce wiring by about 9.6%. We note the following about this procedure:

- It is possible that a node n in L is the only node in some FC X, and if n can be moved to another FC, then FC X can be eliminated from η^*. We came across a few instances where an FC was removed in this manner.
- *wiring_recovery* returns when no node can be moved without increasing the wiring cost of the multi-level network of FCs. At this point, it is still possible that more than one node can simultaneously be moved to realize a gain in wiring. However, this condition is not checked.

The functional correctness of the resulting multi-level network of FCs was verified at the end of the clustering step.

3.4.2 On-the-fly Layout Synthesis

Once the multi-level netlist of FCs is generated in the previous step, we next generate the layout for each $FC^i \in \eta^*$. First, we construct a table of all the 2^n output minterms o_p and their corresponding input cubes $C_p = \Sigma c_{p,q}$. The set of cubes $\{C_p\}$ form a partition of the points in B^m, where $B = \{0,1\}$. This is inexpensive in practice, since m and n are small (6 and 3 respectively in our experiments).

This table is constructed from the truth table of $F^i_{m,n}$, simply by grouping all the input minterms for each output minterm. Now the input minterms for each output minterm are minimized using Espresso [14]. The output minterm which has the largest number of cubes is not implemented, and is mapped to the default output of the FC when it is precharged.

Table 1, shows the number of input minterms (and cubes) that correspond to each output minterm for a representative function $F_{m,n}$ with m = 6 and n = 3. The cubes corresponding to the '7' output are not implemented, since the number of cubes for this output is the largest, and can be mapped to the default output of the FC, since it is a precharged circuit.

Output minterm	0	1	2	3	4	5	6	7	Total
# Input minterms	8	5	8	11	6	7	7	12	64
# Input cubes	8	3	5	4	6	6	3	9	44

Table 1: Example of Minterm Distribution for $F_{m,n}$

For each $FC^i \in \eta^*$, our layout synthesis algorithm adds larger output buffers for output x whenever the fanout load (measured in terms of the total number of pulldown stack devices that x drives) exceeds a particular value.

4. EXPERIMENTS

4.1 Simulation Environment

The designs presented in this paper are implemented in a 45nm process technology. The CMOS standard cell based digital circuits are synthesized and mapped using a 45nm Nangate FreePDK45 Open Cell Library [15, 16] using Synopsys Design Compiler [17]. The delay, power and area of the CMOS standard cell based digital circuits are extracted using Design Compiler. We used custom scripts to generate the flash-based digital circuit. For CMOS devices, we used a 45nm PTM process [18]. For the flash devices, we derived the 45nm flash device models from the measurements results presented in [9] and validated our models using [19]. We also calculated the gate capacitance of the flash transistor and found it 20× smaller than the gate capacitance of the corresponding regular NMOS transistor, and validated this reduction with existing literature which reported reductions of ∼25-30× for a 45nm technology node [20].

The target programmed threshold voltages used in our designs are (VT_0 = -0.5 V) and (VT_1 = 0.5 V). We simulated the flash-based FCs in HSPICE and also verified the correct logical operation of the flash-based digital circuit, realized as a network of interconnected FCs. Custom layouts for the FCs were generated using Cadence Virtuoso [21] to compare the physical area of the flash-based digital circuits to their standard-cell based counterparts. The layout of our FCs used design rules for flash devices that were obtained from the ITRS [22].

4.2 Flash-based Analysis Details

Section 3.4 described the conversion of an input logic netlist η into a multi-level netlist of FCs η^*, and also discussed how the layout of each FC was generated. In this section, we describe the methodology we used to extract the delay, power and area of our multi-level flash-based design.

We first characterized a generalized FC, and generated delay, power and area models for the FC in terms of m, n, and several other parameters of the FC. We use these models to estimate the delay, power and area of the mapped multi-level network of FCs.

In our flash-based methodology, any FC^i can realize a logic function of up to n outputs and m inputs. The main factors that determine the delay, power and area of FC^i are:

- The total number of cubes C^i_{tot} implemented in the FC (i.e. total number of pulldown stacks over all the FLAs in FC^i).
- The maximum number of cubes C^i_{FLB} in any FLB of FC^i (this number is bounded by CPB)
- The number of outputs N^i of the FC^i.

The delay of FC^i is proportional to both the total number of cubes in the FC (C^i_{tot}) and the maximum number of cubes over the FLBs of FC^i (C^i_{FLB}). The power of FC^i is proportional to the total number of cubes (C^i_{tot}) and the number of output of FC^i (N^i). The area of the layout of FC^i depends on all the three factors. We fix the number of inputs of the FC (m) to 6 in order to preserve the regularity of the FC and make it easier to place and route.

To characterize the delay, area and power of an FC, we constructed a library of FCs with number of outputs (N^i) varying from 1 to 3, and number of cubes per FC (C^i_{tot}) ranging from 1 up to 56 (this is the maximum number of cubes for a 6 input function, assuming that we perform the on-the-fly layout synthesis which maps the output minterm with the largest number of cubes to the default output minterm $< 111 >$), and the C^i_{FLB} varying from 1 to CPB (which is 3 in our work).

4.2.1 Delay Characterization and Estimation

For delay characterization, we measure the delays of FCs with C^i_{tot} ranging from 1 up to 56, and for C^i_{FLB} varying between 1 and CPB (which is 3). The latter condition corresponds to the case when the output of the FC^i is discharged through an FLB with 1, 2 or 3 cubes respectively. We also measure the maximum precharge delay D_{Pch} across all the FC^i configurations. The following equation explain how the delay of the flash-based design is computed.

$$Delay = D^{max}_{Pch} + \sum_{FC^i \in \Pi} \left[D_{FC}\left(C^i_{tot}, C^i_{FLB}\right) + (D_{OB}) \times \alpha_i \right] \quad (1)$$

Equation 1 summarizes the critical path delay calculation methodology. D^{max}_{Pch} is the pre-characterized maximum precharge delay for all the FCs. For all the FC^i on the critical path Π, we look up the delay D_{FC} of the FC itself, and the delay D_{OB} of the output driver (if the FC drives a load greater than a threshold). C^i_{tot} is the total number of cubes (pulldown stacks) in FC^i and C^i_{FLB} is the maximum number of cubes among all the FLBs in FC^i. D_{OB} is the delay of the output buffer, which is only added if the fanout of FC^i exceeds a certain threshold. The binary variable α_i is set to 1 when the fanout of FC^i exceeds the threshold, and α_i = 0 otherwise. In any FC, the output generation circuit is capable of driving an equivalent load of a 4×-5× buffer (which is equivalent to driving up to 100 gates of flash transistors in pulldown stacks). Recall that the flash transistor has 20× smaller input capacitance than a regular MOSFET, as discussed earlier. During layout synthesis, our CAD tool inserts an output buffer for each FC that has one or more outputs with a load higher than 96 flash transistor gates. This output buffer is a 4× buffer, which is capable of driving a load equivalent to the input load of a CMOS buffer of size 16×-20×, effectively guaranteeing that our output buffer is strong enough to drive about 400 flash transistor gates, which is larger than any load encountered in our experiments. The total delay is equal to the summation of the delays of the FCs in the critical path Π, the maximum precharge delay, and the sum of the delays of the inserted output buffers. The critical path delay is found by running a static timing analysis tool which we implemented, using a dynamic programming model.

4.2.2 Power Characterization and Estimation

We first characterize the power of any FC, computing the power (precharge as well as evaluate) for general FC configurations. The configurations varied C^i_{tot} from 1 to 56 and N^i from 1 to 3. The results of these characterization runs are stored in a lookup table.

Equation 2 shows the details of how power estimation is performed. Here, AF is the logic activity factor, which is taken to be 15%, a representative number for logic designs. N^i is the number of outputs in FC^i. The total power is computed as the sum of the power of all the FCs in the design.

$$Power = AF * \left[\sum_{\forall FC_i} P(C^i_{tot}, N^i) + (P_{OB}) \times \alpha_i \right] \quad (2)$$

4.3 Results and Analysis

We evaluated our flash-based digital circuit design approach by implementing a set of 12 of the largest benchmarks from ISCAS89 [23], ITC99 [24] and EFPL [25] benchmark suites. We compare the delay, power and area results of the flash-based implementation to a CMOS standard-cell based implementation of the benchmarks. Table 2 shows the benchmark name (Column 1), the number of cells used to implement the design using a traditional CMOS standard-cell based approach (Column 2), the number of FCs used to implement the design using the flash-based approach (Column 3), the average number of inputs over all the FCs (Column 4), the average number of outputs of the FCs (Column 5), the average number of cubes in the FCs (Column 6), the CMOS

Benchmark	CMOS	Flash				CMOS	Flash	CMOS	Flash	CMOS	FLASH
	No. Stdcells	No. FCs	m_{Avg}	n_{Avg}	Avg No. Cubes	Delay (ns)	Delay Ratio	Power (mW)	Power Ratio	Cell Area $(\mu m)^2$	Cell Area Ratio
b17	47500	6658	5.61	2.32	5.49	7.67	0.90×	25.97	0.49×	56964.97	0.67×
b20	22983	2901	5.60	2.32	6.02	6.11	0.50×	30.80	0.18×	27114.98	0.60×
b21	23324	2963	5.62	2.32	6.03	6.11	0.49×	30.93	0.18×	27521.69	0.61×
b22	34693	4362	5.61	2.33	5.97	6.16	0.47×	43.67	0.19×	40926.49	0.60×
s13207	2828	460	5.60	2.38	4.83	1.88	0.43×	0.98	0.91×	3365.70	0.67×
s15850	3735	594	5.50	2.30	4.90	2.63	0.67×	1.82	0.62×	4357.35	0.65×
s35932	10661	1290	5.24	2.60	6.62	0.74	0.29×	12.06	0.22×	12964.31	0.51×
s38417	10771	1593	5.68	2.22	5.04	1.48	0.89×	6.82	0.43×	12256.22	0.60×
s38584	13895	2077	5.59	2.16	4.93	2.03	0.90×	6.19	0.61×	16048.05	0.60×
multiplier	46363	5821	5.67	2.48	6.56	18.80	0.47×	105.41	0.11×	56460.10	0.57×
voter	22453	2708	5.38	2.53	6.10	5.91	0.58×	37.93	0.14×	26738.59	0.56×
square	38009	5109	5.63	2.49	5.61	18.26	0.49×	63.05	0.16×	46558.78	0.58×
Average	23101	3045	5.56	2.37	5.68	6.48	0.59×	30.47	0.35×	27606.43	0.60×

Table 2: Delay, Power, Energy and Cell Area Ratios of Flash-based Digital Circuits Relative to their CMOS Standard Cell-based Counterparts

delay and the flash-based design delay ratio (Columns 7 and 8), the CMOS power and the flash-based design power ratio (Columns 9 and 10), the CMOS area and the flash-based design area ratio (Columns 11 and 12). The delay, power and area ratios of the flash-based designs are relative to their CMOS standard-cell based counterparts.

Comparing the average number of standard cells to the average number of FCs across all the benchmarks we observe that on average, each FC is equivalent to $\sim 7.6\times$ standard cells. These FCs have an average of 5.56 inputs and 2.37 outputs, which are close to the maximum number of inputs (6) and outputs (3) in our design. The average number of cubes implemented in each FC, however, is low (5.68) compared to the maximum possible number of cubes for a 6-input logic function.

Table 2 shows that the flash-based design approach is $\sim$41% faster operation and consumes $\sim$65% lower power on average, compared to a traditional CMOS standard-cell based design approach. This is a significant improvement, and results in an energy improvement of $\sim 5\times$ over the standard-cell based approach.

The key reasons for the reduced delay are:

- Lowered gate capacitance of the flash FET (20× lower than a regular MOSFET).
- The increased parasitics of the standard cells (due to the use of NMOS as well as PMOS devices).
- The use of shared (un-contacted) diffusions in the NAND stack reduces parasitics significantly, thus reducing delays.
- The FCs used to implement the benchmarks have relatively small sizes (the average number of cubes implemented in each FC is 5.68, which reduces the input capacitive loads).

It is well known that dynamic designs consume greater power than static CMOS designs. Our FC based design consumes less power for several reasons. Despite being dynamic, the number of nodes being precharged is smaller than a CMOS (domino or other dynamic) approach. Further, the long transistor stacks (since we choose $m = 6$) result in smaller evaluation currents, reducing power further. Also, in our design, exactly one FLB pulls down during every evaluation, reducing switching activity and hence power consumption. Finally, the I_{ds} of a 45nm flash FET is lower than that of our 45nm MOSFET, which results in a lower power consumption.

We also report the area ratio of both implementations. The area reported for the CMOS standard cell based implementation is the sum of cell layout areas, while the area of our flash-based approach is the sum of the layout areas of all the FCs in the design. Design rules for flash were obtained from the ITRS 45nm flash technology node [22]. Digital circuits implemented in an FC use 0.6× the physical area of a CMOS-based design, on average.

5. CONCLUSION

Flash transistors are the workhorse technology for non-volatile data storage applications today. It was recently shown how flash transistors can be used to implement digital circuits in an ASIC-like manner. However, the focus was on a single flash cluster (FC) which implements a function with a small number of inputs and outputs.

This paper, in contrast, is the first, to the best of the authors' knowledge, to use flash transistors to implement complete digital circuit blocks, in the form of an interconnected network of FCs. The focus of this paper is on logic clustering, on-the-fly physical synthesis of all the FCs of a design, and the automatic characterization of the delay, power and area of the resulting circuit.

Our characterization results show that, averaged over 12 large designs, our approach yields 0.59× the delay, 0.35× the power, and 0.60× the area of the equivalent circuit implemented using CMOS standard cell-based design. It is generally rare that a circuit methodology yields results that are better than existing commercial standard-cell based flows in terms of delay, area, power *and* energy, and in this sense, we submit that our results are significant.

The threshold voltage of flash devices can be modified at a fine granularity during programming, which results in several advantages. First, speed binning at the factory can be done with precision. Secondly, an IC can be re-programmed in the field (we note that our flash-based design is *not* an FPGA, but rather an ASIC style design), to diminish or eradicate effects such as aging.

6. REFERENCES

[1] R. Fowler and L. Nordheim, "Electron emission in intense electric fields," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 119, pp. 173–181, May 1928.

[2] D. Jung, Y.-H. Chae, H. Jo, J.-S. Kim, and J. Lee, "A group-based wear-leveling algorithm for large-capacity flash memory storage systems," in *Proceedings of the 2007 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '07, pp. 160–164, ACM, 2007.

[3] S. Boboila and P. Desnoyers, "Write endurance in flash drives: Measurements and analysis," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, (Berkeley, CA, USA), pp. 9–9, 2010.

[4] K. Takeuchi, "Novel co-design of NAND flash memory and NAND flash controller circuits for sub-30 nm low-power high-speed solid-state drives (SSD)," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 1227–1234, April 2009.

[5] S. L. et al., "A 128Gb 2b/cell NAND flash memory in 14nm technology with tPROG=640us and 800MB/s I/O rate," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 138–139, Jan 2016.

[6] D. L. et al., "A 64Gb 533Mb/s DDR interface MLC NAND flash in sub-20nm technology," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 430–432, Feb 2012.

[7] K.-T. P. et al., "A 7MB/s 64Gb 3-bit/cell DDR NAND flash memory in 20nm-node technology," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pp. 212–213, Feb 2011.

[8] J.-R. H. et al., "20nm gate bulk-FinFET SONOS flash," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, pp. 154–157, Dec 2005.

[9] H. An, K. Kim, S. Jung, H. Yang, K. Kim, and Y. Song, "The threshold voltage fluctuation of one memory cell for the scaling-down NOR flash," in *2nd IEEE Int'l Conf. on Network Infrastructure and Digital Content*, Sept 2010.

[10] E. Choi and S. Park, "Device considerations for high density and highly reliable 3D NAND flash cell in near future," in *IEEE International Electron Devices Meeting (IEDM)*, (San Francisco, CA), pp. 9.4.1 – 9.4.4, Dec 2012.

[11] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, (New York, NY, USA), pp. 14–23, ACM, 2009.

[12] M. Abusultan and S. Khatri, "A ternary-valued, floating gate transistor-based circuit design approach," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016.

[13] M. Abusultan and S. Khatri, "Implementing low power digital circuits using flash devices," in *Computer Design (ICCD), 2016 34nd IEEE International Conference on*, October 2016.

[14] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA, USA: Kluwer Academic Publishers, 1984.

[15] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An open-source variation-aware design kit," in *IEEE International Conference on Microelectronic Systems Education (MSE)*, pp. 173–174, June 2007.

[16] "NanGate Library Optimization website." `http://www.nangate.com/`.

[17] "Synopsys website." `http://www.synopsys.com/`.

[18] "PTM website." `http://www.ptm.asu.edu/`.

[19] K. Zaitsu, K. Tatsumura, M. Matsumoto, M. Oda, S. Fujita, and S. Yasuda, "Flash-based nonvolatile programmable switch for low-power and high-speed FPGA by adjacent integration of MONOS/logic and novel programming scheme," in *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*, pp. 1–2, June 2014.

[20] S. G. Jung, K. W. Lee, K. S. Kim, S. W. Shin, S. S. Lee, J. C. Om, G. H. Bae, and J. H. Lee, "Modeling of Vth shift in NAND flash-memory cell device considering crosstalk and short-channel effects," *IEEE Transactions on Electron Devices*, vol. 55, pp. 1020–1026, April 2008.

[21] "Cadence Design Systems website." `http://www.cadence.com/`.

[22] "ITRS website." `http://www.itrs.net/`.

[23] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Circuits and Systems, 1989., IEEE International Symposium on*, pp. 1929–1934 vol.3, May 1989.

[24] F. Corno, M. S. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design Test of Computers*, vol. 17, pp. 44–53, Jul 2000.

[25] "The EPFL Combinational Benchmark Suite Webpage." `http://lsi.epfl.ch/benchmarks`.