

Independent-Gate FinFET Circuit Design Methodology

Michael C. Wang

Abstract—FinFET, which is a double-gate field effect transistor (DGFET), is more versatile than traditional single-gate field effect transistors because it has two gates that can be controlled independently. Usually, the second gate of FinFETs is used to dynamically control the threshold voltage of the first gate in order to improve circuit performance and reduce leakage power. However, we can also utilize the second gate to implement circuits with fewer transistors. This is important since area efficiency is one of the main concerns in modern circuit design. In this paper, a methodology for effectively synthesizing logic circuits using both gates of FinFETs as inputs is presented. Simulation results show that independent-gate FinFET circuit implementation has significant advantages over single-gate FinFET circuit implementation in terms of power consumption and cell area.

Index Terms—FinFET, CMOS, Karnaugh map, circuit synthesis, combinational logic.

I. INTRODUCTION

The two major members of the logic family are static CMOS logic and pass-transistor logic. In general, circuits implemented in static CMOS logic use more transistors, but consume less power and achieve full voltage swing, while circuits implemented in pass-transistor logic consume more power and have reduced voltage swing, but use fewer transistors.

As the size of transistors has scaled down, so have many digital applications. Cell phones, laptops, sensors, and many other applications all shrunk in size over the last few decades and they are more and more portable. For this to happen, chips in these digital applications have to be designed to optimize the number of transistors used, the fewer the better. For this reason, pass-transistor logic is an attractive option. However, pass-transistor logic creates possible situations in which NMOS transistors have to drive a high logic value or PMOS transistors have to drive a low logic value. Since NMOS transistor is not a good pull-up device (and PMOS transistor is not a good pull-down device), outputs of pass-transistor circuits will suffer from a voltage drop V_{th} and never achieve full voltage swing to V_{DD} . With the continuing scaling of supply voltage, the voltage swing reduction cannot be tolerated. Therefore, another approach to reduce the transistor count (and area) in digital circuits is needed.

The emergence of FinFET provides a promising solution. FinFET, a double-gate device in which a second gate is added

opposite to the traditional (first) gate, has long been recognized for its potential to better control short-channel effects [1] [2]. Fabrication of FinFET is very close to that of conventional CMOS process. As a result, planar product designs have been converted to FinFET without disruption to the physical area, thereby demonstrating its compatibility with today's planar CMOS design methodology and automation techniques [3].

The additional back gate of FinFETs gives circuit designers many options. It can serve as a secondary gate that enhances the performance of the front (first) gate. For example, if the front gate voltage is V_{DD} (transistor is ON) the back gate can be biased to V_{DD} to provide bigger current drive, which reduces transistor delay. If the front gate voltage is 0 (transistor is OFF), the back gate can be biased to 0, which raises the threshold voltage of the front gate and reduces the leakage current. This can be achieved by simply tying the front gate and the back gate together. FinFETs configured this way are called single-gate FinFET, or SG-FinFET. Most recent FinFET circuit researches, such as FinFET SRAM [4], focus on utilizing the back gate to improve circuit performance. On the other hand, the back gate can also be used independently as an input to reduce the number of transistors needed to implement many logic functions. For an N-FinFET, the transistor turns on if either the front gate or the back gate is V_{DD} – this is equivalent to two NMOS transistors in parallel. FinFETs configured this way are called independent-gate FinFET, or IG-FinFET. Recent researches, such as a 3-transistor FinFET NAND gate [5], utilize this property. However, we have not seen any research that utilizes this property beyond a simple logic gate such as a NAND gate. The main contribution we make in this paper is proposing a general methodology for effectively synthesizing any logic circuit by using both gates of FinFETs as inputs.

This paper is organized as followed: In Section II, we explain the IG-FinFET circuit design methodology in detail. In Section III, we present several circuits synthesized using the proposed methodology. In Section IV, we examine simulation results of a majority gate and a 2-1 MUX implemented using SG-FinFET static CMOS logic, SG-FinFET pass-transistor logic, and IG-FinFET static CMOS logic. We conclude the paper in Section V.

II. DESIGN METHODOLOGY

In this section, we propose an IG-FinFET circuit design methodology. In Section A, we give a brief overview of the logic family. In Section B, we describe the proposed

Manuscript received January 16, 2010.

Michael C. Wang received his Master of Engineering in Electrical Engineering from Princeton University, Princeton, NJ 08544 USA (phone: 949-433-7789; fax: 949-679-7359; e-mail: mcwang@alumni.princeton.edu).

methodology. In Section C, we describe another type of FinFET and discuss how it affects the proposed methodology.

A. Logic Family

In static CMOS logic circuits, outputs are always connected to either V_{DD} through a pull-up network, or 0 through a pull-down network, so that they are never floating. Since NMOS transistors are good at passing 0 and PMOS transistors are good at passing V_{DD} , NMOS transistors are used in the pull-down network and PMOS transistors are used in the pull-up network.

Pass-transistor logic circuits can be designed by first generating binary decision diagrams, and then mapping nodes to transistors and branches to wires [6]. In pass-transistor logic circuits, transistors can pass 0 and V_{DD} . In other words, they “do more work” than transistors in static CMOS logic circuits that only pass either 0 (NMOS transistors) or V_{DD} (PMOS transistors). Intuitively, the number of transistors needed to implement pass-transistor logic circuits is generally smaller than the number of transistors needed to implement static CMOS logic circuits. Unfortunately, for that reason, pass-transistor logic circuits do not achieve full voltage swing at output nodes because transistors have to pass 0 and V_{DD} , and as mentioned earlier, each type of transistor is only good at passing either 0 or V_{DD} . To restore output voltage, techniques such as appending a three-transistor level restorer at output nodes are required.

Both static CMOS logic and pass-transistor logic were developed for conventional MOSFETs. If FinFET technology is available, we can easily adapt both circuit design methodologies by replacing NMOS transistors with N-FinFETs and PMOS transistors with P-FinFETs, then, tie both gates of FinFETs together. In other words, we can design the FinFET version of static CMOS logic circuits or pass-transistor logic circuits that retains the same functionalities as the MOSFET version by using SG-FinFETs, which, in the mean time, provide better circuit performance and reduce leakage current through effective suppression of short-channel effect and near-ideal subthreshold swing.

B. Algorithm

IG-FinFETs are FinFETs whose gates are tied to different inputs. Logically, an IG-FinFET is equivalent to two

Table 1. Algorithm for synthesizing logic functions.

1. From any logic function, construct a Karnaugh map. Let P be the set of all minterms.
2. Find the power set PS of P, let SPS be a subset of PS (See Tables 2 and 3 for how to determine SPS for a 2-input function).
3. Perform union and intersection operations on members of SPS.
4. Map members of SPS into transistors (NMOS, PMOS, N-FinFET, and P-FinFET).
5. Map the union and intersection operations in step 3 as parallel and serial connections. Let the resulting transistor network be TR.
6. Connect one end of TR to a driver and connect the other end to output.
7. Repeat steps 2-6 until the circuit is completed.

MOSFETs in parallel with different inputs. The remainder of this paper will focus on IG-FinFET circuit design.

We now develop a methodology for synthesizing logic functions using IG-FinFETs in addition to SG-FinFETs (which are logically equivalent to traditional MOSFETs). We use Karnaugh map as a tool to aid us in analyzing logic functions. Readers can find a brief introduction to Karnaugh map in [7]. Table 1 shows a general algorithm for synthesizing logic functions.

Circuits in either static CMOS logic or pass-transistor logic can be synthesized using the methodology shown in Table 1. For static CMOS logic, the driver in step 6 is either 0 (pull-down network) or V_{DD} (pull-up network). Once one of the networks is completed, the other network can then be constructed by transforming parallel (or serial) connections to serial (or parallel) connections. For pass-transistor logic, the driver in step 6 is an input, thus it is a variable value. Note that the SG-FinFET circuit design methodology is identical to the traditional MOSFET circuit design methodology, except NMOS transistors are replaced by N-FinFETs and PMOS transistors are replaced by P-FinFETs.

The remaining question is how to map members of SPS into transistors. This is where the IG-FinFET circuit design methodology excels. For circuits implemented using only SG-FinFETs, only sets whose elements are adjacent on a hypercube can be included in SPS, whereas for circuits implemented using both IG-FinFETs and SG-FinFETs, SPS can include sets not otherwise allowed.

Let us illustrate this point with an example. Assume we want to implement a function with only two inputs: A and B. The Karnaugh will contain only 4 minterms. Table 2 shows the comparison between implementations using only SG-FinFETs and implementations using only IG-FinFETs in terms of allowable sets for SPS.

It is easy to see that the sets allowed for an IG-FinFET “covers” more minterms. Since implementing any logic function is simply covering all minterms, circuits implemented using only IG-FinFETs usually requires fewer transistors.

We want to clarify an important point. IG-FinFET circuit design methodology does not exclude the use of SG-FinFETs. Therefore, in IG-FinFET circuit design

Table 2. Allowable sets for SPS comparison between an SG-FinFET and an IG-FinFET for a 2-input function.

P	PS	Sets allowed for SPS			
		SG-FinFET		IG-FinFET	
00	()	(00,01)	B'	(00,01,10)	A'+B'
01	(00)	(00,10)	A'	(00,01,11)	A+B'
10	(01)	(01,11)	A	(00,10,11)	A'+B
11	(10)	(10,11)	B	(01,10,11)	A+B
	(11)				
	(00,01)				
	(00,10)				
	(00,11)				
	(01,10)				
	(01,11)				
	(10,11)				
	(00,01,10)				
	(00,01,11)				
	(00,10,11)				
	(01,10,11)				
	(00,01,10,11)				

methodology, all 8 sets from Table 2 are allowed to be included in SPS. In other words, IG-FinFET circuit design methodology allows us to use IG-FinFETs in addition to SG-FinFETs, thus gives us more options to map minterms effectively to reduce the number of transistor required to implement a circuit.

C. "AND" FinFET

A FinFET transistor can be treated as two MOSFET transistors in parallel. In essence, FinFETs exhibit an "OR" property – an N-FinFET turns ON if either one or both of its gates are ON. The next logical question is whether or not there exists a transistor exhibiting an "AND" property – it turns ON if and only if both gates are ON.

Contrary to intuition, OR function and AND function are closely related from the perspective of a threshold function. A threshold function $f(x_1, x_2, \dots, x_n)$ is 1 if $x_1 w_1 + x_2 w_2 + \dots + x_n w_n \geq T$ and 0 otherwise, where T is the threshold and w_i is the weight. A weight vector $V = \langle w_1, w_2, \dots, w_n \rangle$ identifies a specific function [8]. For example, $V_1 = \langle 1, 1, 1 \rangle$ is an OR function because $f(x_1, x_2) = 1$ if and only if $x_1 + x_2 \geq 1$, and this inequality is satisfied by having inputs $(x_1, x_2) = (0, 1), (1, 0),$ or $(1, 1)$, which is exactly what an OR function does. Similarly, $V_2 = \langle 1, 1, 2 \rangle$ is an AND function. By comparing V_1 and V_2 , we notice that the only difference that distinguishes OR from AND is the threshold value.

Now let us try to apply this idea to FinFET. In Fig. 1 [9], we see that the front gate's threshold voltage reduces dramatically if the back gate voltage is high (and vice versa). We hypothesize that if we move the curve in Fig. 1 upward continuously, eventually the threshold voltage of both gates will rise to a point where both inputs have to be ON in order to cause conduction – we have just described the "AND" property that can theoretically be realized by raising the threshold voltage of both gates.

To confirm this behavior, we use MEDICI [10], which is a device level simulator, to simulate a prototype double gate device. Fig. 2 is a schematic of the double gate device that we simulated.

There are several things we can do to raise the threshold voltage of a gate. Threshold voltage of Gate 1 of a FinFET is as follow: $V_{thG1} = \phi_{msG1} + V_{th0} - \gamma [V_{G2} - (\phi_{msG2} + V_{th0})]$, where ϕ_{ms} is the work function, V_{th0} is the threshold voltage found

without any secondary effect, V_{G2} is the Gate 2 voltage, V_{thG1} is the Gate 1 threshold voltage, and γ is a fixed parameter that depends on the geometry of the device [11].

From the equation, it is clear that raising the work function of the gate is the easiest way to increase the threshold voltage. Therefore, in our simulation, we raise the work function from the original 4.5eV to 5.2eV. This choice of work function is practical since it can be achieved by using p+ polysilicon, which are widely used as transistor gate material today (p+ polysilicon has work function of 5.17eV ~ 5.2eV [12]). The simulation results are displayed in Fig. 3.

The solid horizontal line ($10^{-4}A$) in Fig. 3 represents the cutoff current between ON and OFF operation. We also limit the input value to discrete value 0V and 1V. First, let us look at a conventional FinFET operation, which is represented by diamond dot curve (WF=4.5eV, $V_{G2}=1V$) and square dots curve (WF=4.5eV, $V_{G2}=0V$). When the back gate (V_{G2}) is at 1V, the transistor current, which is represented by diamond dot curve, is always above the solid horizontal line. This means the transistor is turned ON if V_{G2} is high. If $V_{G2} = 0$ (represented by square dot curve), we see that the curve rises above the solid horizontal cutoff line only if the front gate (V_{G1}) is 1. If Both V_{G1} and V_{G2} are 0, then the transistor is OFF. This behavior is exactly what traditional FinFETs exhibit. Now let us look at the double gate device with work function of 5.2eV. We notice that the current is ON if and only if both V_{G1} and V_{G2} are 1 (represented by triangle dot curve at 1V, which crosses over the cutoff line). For any other combination of V_{G1} and V_{G2} , the current are far below the solid horizontal line and the transistors are OFF. Clearly, this is a FinFET with the "AND" behavior – it is equivalent to two MOS transistors in series.

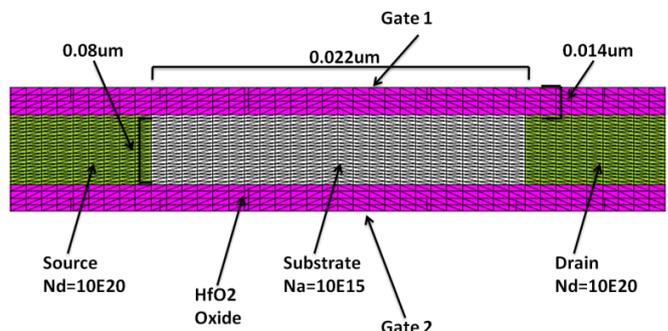


Fig. 2. Schematic of a double gate device.

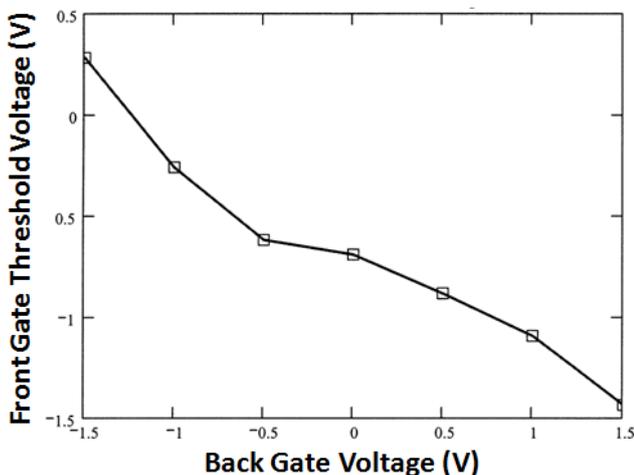


Fig 1. Front gate threshold voltage as a function of back gate voltage for N-FinFET.

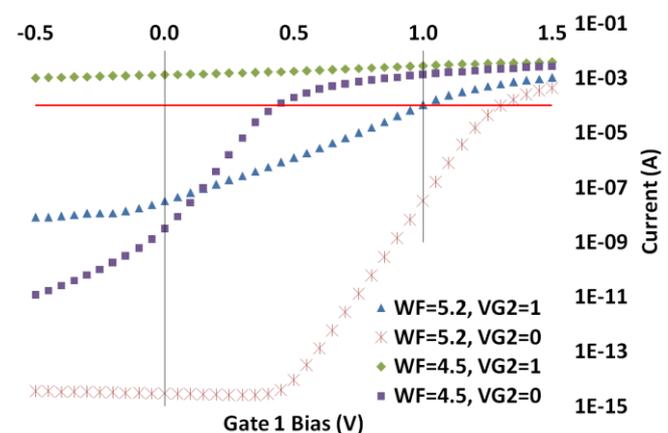


Fig. 3. Drain current vs. Front gate voltage of a double gate device under different work function 4.5eV and 5.2eV with different back gate (V_{G2}) bias 0V and 1V.

Table 3. Allowable sets for SPS comparison between an SG-FinFET and an IG-FinFET with the addition of “AND” FinFET for a 2-input function.

P	PS	Sets allowed for SPS			
		SG-FinFET		IG-FinFET	
00	()	(00,01)	B'	(00)	A'*B'
01	(00)	(00,10)	A'	(01)	A*B'
10	(01)	(01,11)	A	(10)	A'*B
11	(10)	(10,11)	B	(11)	A*B
	(11)			(00,01,10)	A'+B'
	(00,01)			(00,01,11)	A+B'
	(00,10)			(00,10,11)	A'+B
	(00,11)			(01,10,11)	A+B
	(01,10)				
	(01,11)				
	(10,11)				
	(00,01,10)				
	(00,01,11)				
	(00,10,11)				
	(01,10,11)				
	(00,01,10,11)				

Though the method of creating “AND” FinFET by raising the work function of the gate looks promising, there are some problems with this approach. The biggest challenge is the subthreshold swing associated with high back gate bias ($V_{G2}=1$). As we can see from the triangle dot curve, it takes entire 1V voltage swing to raise the current from OFF state to ON state, which means we cannot reduce V_{DD} any further. This does not bode well when many applications today are operating with supply voltage lower than 1V. The subthreshold swing problem also results in small ON current, which substantially increases the delay of this device. We currently do not have a solution to this subthreshold swing problem. However, we believe this problem can be solved, and we will leave that to device physicists. We will instead focus at the additional design advantages that “AND” FinFET presents. Table 3 shows the comparison between circuits implemented using only SG-FinFETs and circuits implemented using only IG-FinFETs in terms of allowable sets for SPS with the addition of “AND” FinFET.

III. EXAMPLES

In this section, we will go over 5 logic gates: a 3-input majority gate, a 2-1 MUX, a 3-input XOR function, a 2-bit comparator, and a 2-bit equality checker.

All schematics and simulations presented in this paper are based on FinFET. We want to limit extra variables, such as

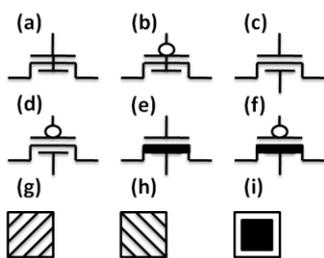


Fig. 4. List of legends. (a) SG N-FinFET (b) SG P-FinFET (c) IG N-FinFET (d) IG P-FinFET (e) IG “AND” N-FinFET (f) IG “AND” P-FinFET (g) Minterms allowed by an IG-FinFET (h) Minterms allowed by an IG “AND” FinFET (i) Minterms allowed by a pull-up, a pull-down network, or a pass-transistor network.

type of transistors, in order to highlight differences between SG-FinFET circuit design implementation and IG-FinFET circuit design implementation. Fig. 4 shows a list of legends that will be used in this section.

A. Majority Gate

A majority gate is commonly used in a full adder. A typical majority gate has three inputs and one output. If more than half of the inputs are 1, it returns 1 on the output, otherwise it returns 0. Schematics of SG-FinFET static CMOS logic implementation, SG-FinFET pass-transistor logic implementation, and IG-FinFET static CMOS implementation of a majority gate are shown in Fig. 5.

The IG-FinFET static CMOS majority gate, as shown in Fig. 5(c), contains just 6 transistors. By examining the circuit, we can obtain the output expression, $Out = (A'+B')*(A'+C')*(B'+C')$, which is actually in the OR-AND logic form (in contrast with the more intuitive AND-OR logic form). To construct the pull-down network, we want $(A'+B')*(A'+C')*(B'+C')=1$ (so that $Out=0$, which means pull-down network is triggered to pass 0). Implementation each of $A'+B'$, $A'+C'$, and $B'+C'$ requires two NMOS transistors, but since both transistors share the same source and drain, they can be merged into one single N-FinFET. Next, we place these N-FinFETs in series as implied by the product sign of the OR-AND logic expression. Similarly, we construct the pull-up network to complete the circuit shown in Fig. 5(c).

Karnaugh maps in Fig. 6 and Fig. 7 illustrate the same point graphically. For example, Fig. 6(a) shows minterms that satisfy $A+B=1$, Fig. 6(b) shows minterms that satisfy $A+C=1$, Fig. 6(c) shows minterms that satisfy $B+C=1$, and Fig. 6(d) shows minterms that satisfy all three conditions, and these minterms are exactly same as the 1-minterms for a majority gate. Since each condition can be realized by one FinFET, total of three FinFETs are needed to realize the pull-up network. The pull-down network is similarly constructed and illustrated in Fig. 7. As expected, the resulting circuit has six transistors as shown in Fig. 5(c).

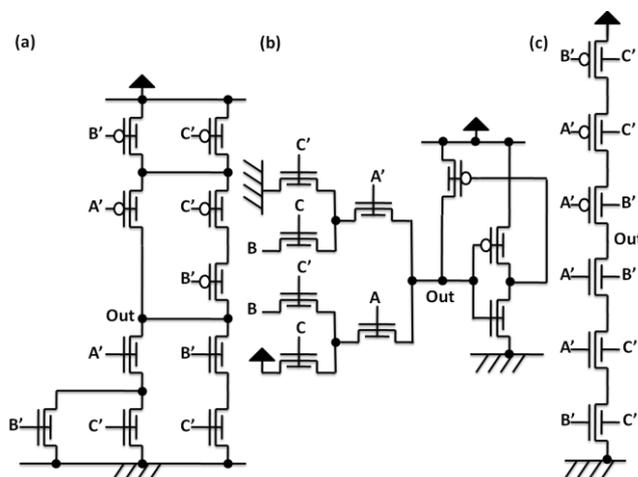


Fig. 5. Schematics of different implementations of a majority gate. (a) SG-FinFET static CMOS logic implementation (b) SG-FinFET pass-transistor logic implementation, and (c) IG-FinFET static CMOS logic implementation.

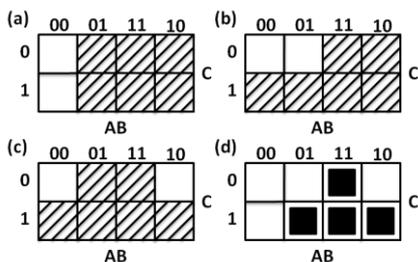


Fig. 6. Karnaugh maps for synthesizing the pull-up network of the circuit in Fig. 5(c). (a) Allow A+B (b) Allow A+C (c) Allow B+C (d) Allow $(A+B)*(A+C)*(B+C)$.

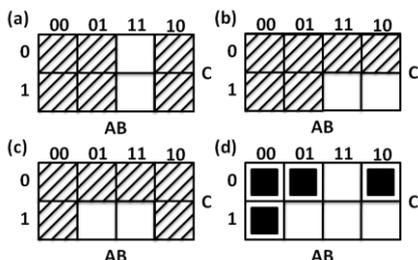


Fig. 7. Karnaugh maps for synthesizing the pull-down network of the circuit in Fig. 5(c). (a) Allow $A'+B'$ (b) Allow $A'+C'$ (c) Allow $B'+C'$ (d) Allow $(A'+B')*(A'+C')*(B'+C')$.

B. 2-1 MUX

A 2-1 MUX is another widely used 3-input function. It has applications in both combination logic and sequential logic. Schematics of SG-FinFET CMOS logic implementation, SG-FinFET pass-transistor logic implementation, and IG-FinFET static CMOS logic implementation of a 2-1 MUX are shown in Fig. 8. A, B are the primary input bits, S is the select bit. The output bit is equal to A if S=0, and B if S=1. In other word, the output function is $Out=S'*A+S*B$, where S' is the complement of S.

The IG-FinFET static CMOS logic implementation of a 2-1 MUX can be constructed similar to the IG-FinFET static CMOS logic majority gate described in Section III.A. The OR-AND logic expression is $Out'=(S+A')*(S'+B')$, which can be mapped as two N-FinFETs in series in the pull-down network and two P-FinFETs in series in the pull-up network. The resulting circuit, which is shown in Fig. 8(c), contains only 4 transistors.

Karnaugh maps in Fig. 9 and Fig. 10 illustrate the same

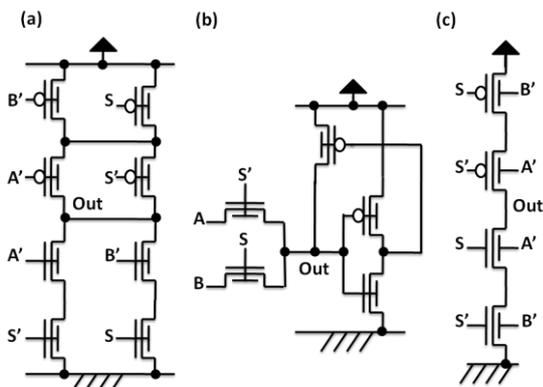


Fig. 8. Schematics of different implementations of a 2-1 MUX. (a) SG-FinFET static CMOS logic implementation (b) SG-FinFET pass-transistor logic implementation, and (c) IG-FinFET static CMOS logic implementation.

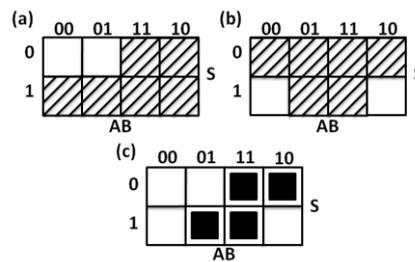


Fig. 9. Karnaugh maps for synthesizing the pull-up network of the circuit in Fig. 8(c). (a) Allow S+A (b) Allow $S'+B$ (c) Allow $(S+A)*(S'+B)$.

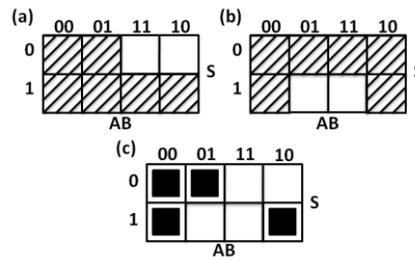


Fig. 10. Karnaugh maps for synthesizing the pull-down network of the circuit in Fig. 8(c). (a) Allow $S+A'$ (b) Allow $S'+B'$ (c) Allow $(S+A')*(S'+B')$.

point graphically. For example, Fig. 9(a) shows minterms that satisfy $S+A=1$, Fig. 9(b) shows minterms that satisfy $S'+B=1$, and Fig. 9(c) shows minterms that satisfy both conditions, and these minterms are exactly same as the 1-minterms for a 2-1 MUX. Since each condition can be realized by one FinFET, total of two FinFETs are needed to realize the pull-up network. The pull-down network is similarly constructed and illustrated in Fig. 10. As expected, the resulting circuit has four transistors as shown in Fig. 8(c). Compare to the SG-FinFET static CMOS logic implementation of a 2-1 MUX, the IG-FinFET static CMOS logic implementation of a 2-1 MUX effectively cuts the number of transistors needed in half.

Note that SG-FinFET pass-transistor logic implementation, which is shown in Fig. 8(b), actually achieves the smallest area among all three implementations, even though it uses one more transistor (include level-restorer circuit) than the IG-FinFET static CMOS logic implementation. However, its high power consumption (almost twice as much as the IG-FinFET static CMOS logic implementation) makes it less attractive for low-power applications.

C. 3-input XOR Gate

A 3-input XOR gate is another useful 3-input function. A full adder is essentially a majority gate and a 3-input XOR gate. Schematic of IG-FinFET pass-transistor logic implementation of a 3-input XOR gate is shown in Fig. 11. If an odd number s of inputs is 1, the output is 1, otherwise it is 0. Put it in another way, a 3-input XOR gate returns 1 if one or three inputs are 1, while a majority gate returns 1 if two or three inputs are 1. Like all pass-transistor logic circuits, the circuit in Fig. 11 suffers from reduced voltage swing. Therefore, level restorer circuits must be appended to the output node.

Karnaugh maps in Fig. 12 and Fig. 13 illustrate how the circuit is synthesized. For example, Fig. 12(a) shows minterms that satisfy $A+B'=1$, Fig. 12(b) shows minterms

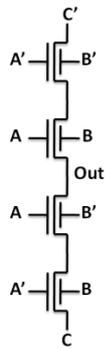


Fig. 11. Schematic of IG-FinFET pass-transistor logic implementation of a 3-input XOR gate.

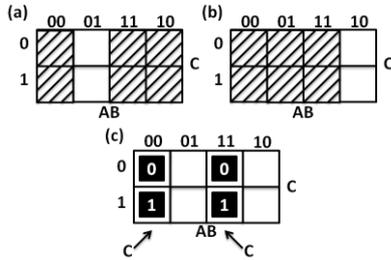


Fig. 12. Karnaugh maps for synthesizing the C network of the circuit in Fig. 11. (a) Allow $A+B'$ (b) Allow $A'+B$ (c) Allow $(A+B')*(A'+B)$.

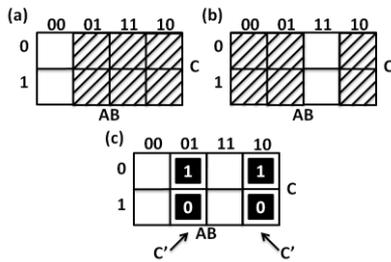


Fig. 13. Karnaugh maps for synthesizing the C' network of the circuit in Fig. 11. (a) Allow $A+B$ (b) Allow $A'+B'$ (c) Allow $(A+B)*(A'+B')$.

that satisfy $A'+B=1$, and Fig. 12(c) shows minterms that satisfy both conditions. To realize the function, all we need to do it is driving the network shown in Fig. 12(c) with the input C. Since each condition can be realized by one FinFET, total of two FinFETs are needed to realize this network. The other part of the circuit is similarly constructed and illustrated in Fig. 13. As expected, the resulting circuit has four transistors, which is a significant improvement over the conventional 3-input XOR gate implementation that requires eight transistors.

D. 2-Bit Comparator

Comparator is a widely used circuit that has many applications in computer architecture. A classic comparator is usually implemented by using many single-bit comparators, each one can be implemented with a single AND gate. In contrast of a single-bit comparator, a 2-bit comparator compares two 2-bit numbers, $X=[AB]$ and $Y=[CD]$. If $X > Y$, the output is 1, otherwise the output is 0. The circuit schematic of a 2-bit comparator in IG-FinFET static CMOS logic implementation is shown in Fig. 14.

Karnaugh maps in Fig. 15 and Fig. 16 illustrate how the circuit is synthesized. Note that this is an example of using both conventional FinFETs and "AND" FinFETs to implement a function. For example, Fig. 15(a) shows

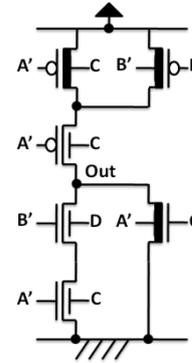


Fig. 14. Schematic of IG-FinFET static CMOS logic implementation of a 2-bit comparator.

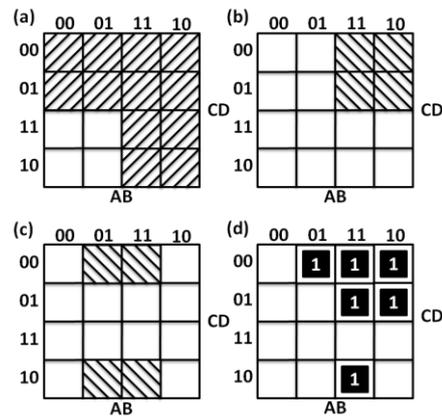


Fig. 15. Karnaugh maps for synthesizing the pull-up network of the circuit in Fig. 24. (a) Allow $A+C'$ (b) Allow $A*C'$ (c) Allow $B*D'$ (d) Allow $(A+C')*(A*C'+B*D')$.

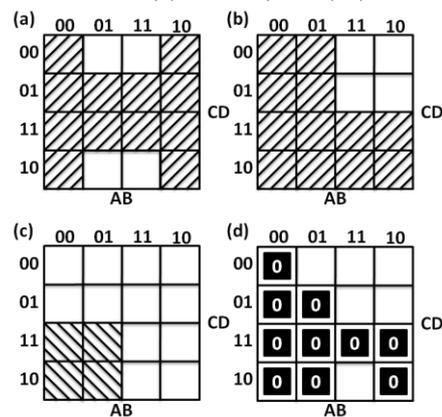


Fig. 16. Karnaugh maps for synthesizing the pull-down network of the circuit in Fig. 24. (a) Allow $B'+D$ (b) Allow $A'+C$ (c) Allow $A'*C$ (d) Allow $(B'+D)*(A'+C)+(A'*C)$.

minterms that satisfy $A+C'=1$, Fig. 15(b) shows minterms that satisfy $A*C'=1$, Fig. 15(c) shows minterms that satisfy $B*D'=1$, and Fig. 15(d) shows minterms that satisfy $(A+C')*(A*C'+B*D')=1$, and these minterms are exactly same as the 1-minterms for a 2-bit comparator. Since each condition can be realized by one FinFET, total of three FinFETs are needed to realize the pull-up network. The pull-down network is similarly constructed and illustrated in Fig. 16. As expected, the resulting circuit has six transistors as shown in Fig. 14. In term of the number of transistors per bit, the 2-bit comparator implementation is superior because it requires on average three transistors per bit (six transistors for two bits), while a classical single-bit comparator implementation requires at least four transistors.

E. 2-Bit Equality Checker

Like a comparator, an equality checker is also a widely used circuit that has many applications in computer architecture. A classic equality checker is usually implemented by using many single-bit equality checkers, each one can be implemented with a single AND gate. In contrast of a single-bit equality checker, a 2-bit equality checker checks to see if two 2-bit numbers are the same. For two 2-bit number $X=[AB]$ and $Y=[CD]$, if $X=Y$, the output is 1, otherwise the output is 0. The circuit schematic in IG-FinFET static CMOS logic implementation is shown in Fig. 17.

Karnaugh maps in Fig. 18 and Fig. 19 show how the circuit is synthesized. For example, Fig. 18(a) shows

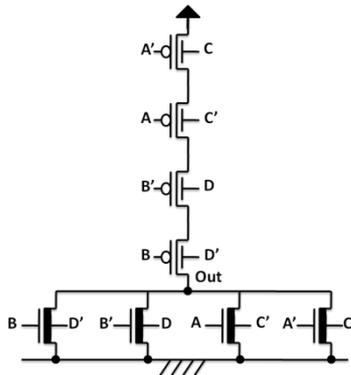


Fig. 17. Schematic of IG-FinFET static CMOS logic implementation of a 2-bit equality checker.

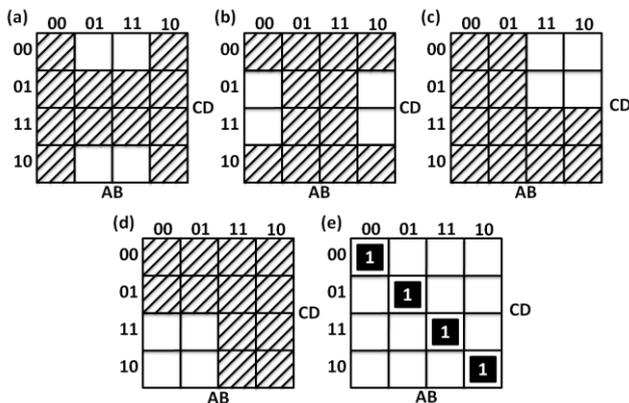


Fig. 18. Karnaugh maps for synthesizing the pull-up network of the circuit in Fig. 17. (a) Allow B^+D (b) Allow $B+D^+$ (c) Allow A^+C (d) Allow $A+C^+$ (e) Allow $(B^+D)*(B+D^+)*(A^+C)*(A+C^+)$.

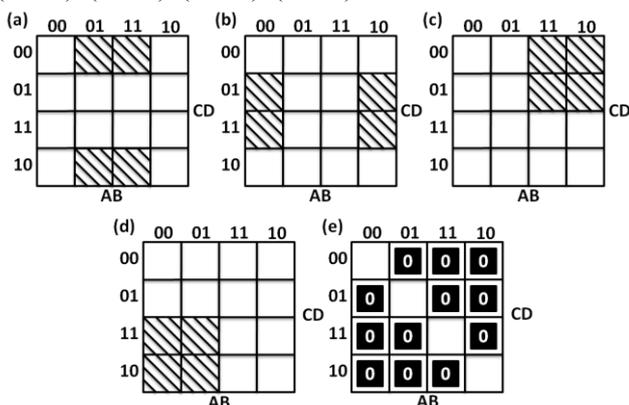


Fig. 19. Karnaugh maps for synthesizing the pull-down network of the circuit in Fig. 17. (a) Allow B^*D^+ (b) Allow B^*D (c) Allow A^*C^+ (d) Allow A^*C (e) Allow $(B^*D^+)+(B^*D)+(A^*C^+)+(A^*C)$.

minterms that satisfy $B^+D=1$, Fig. 18(b) shows minterms that satisfy $B+D^+=1$, Fig. 18(c) shows minterms that satisfy $A^+C=1$, Fig. 18(d) shows minterms that satisfy $A+C^+=1$, and Fig. 18(e) shows minterms that satisfy all conditions, and these minterms are exactly same as the 1-minterms for a 2-bit comparator. Since each condition can be realized by one FinFET, total of four FinFETs are needed to realize the pull-up network. The pull-down network is similarly constructed and illustrated in Fig. 19. As expected, the resulting circuit has eight transistors as shown in Fig. 17. This implementation is very efficient since it only requires eight FinFETs. The conventional implementation requires approximately 20 transistors (2 XNOR gates for each bit and 1 AND gate to combine the result of each bit).

One crucial point that we want to make is the duality relationship between conventional “OR” FinFETs and “AND” FinFETs. Parallel “OR” N-FinFETs in the pull-down network transform into serial “AND” P-FinFETs in the pull-up network, and serial “AND” N-FinFETs in the pull-down network transform into parallel “OR” P-FinFETs in the pull-up network. This can be easily proven by considering the NMOS-PMOS duality relationship and the fact that logically, a conventional “OR” FinFET is two MOSFETs in parallel, while an “AND” FinFET is two MOSFETs in series.

IV. SIMULATION RESULTS

In this section, we present simulation results in 30nm technology generated from Synopsys Sentaurus, which is a device level simulator [13], for all six circuits mentioned in Sections III.A and III.B. Schematics of these circuits can be found in Fig. 5 and Fig. 8. Before we present the results, we will explain how we perform simulations and extract data listed in Tables 4 and 5.

A. Layout Consideration

In FinFET technology, device widths are dispensed in units of whole fins only. [14] This is known as device width quantization, which limits our ability to size transistors effectively in FinFET circuit. On top of that, there is also problem with IG-FinFETs with even number of fins because of the difficulty in routing inputs. However, it is shown in [15] that both inputs can be easily routed in IG-FinFETs with three fins (or more generally, any odd number of fins). For this reason, we chose three as the number of fins of P-FinFET to the number of fins of N-FinFET (three fins P-FinFET to one fin N-FinFET). As shown in Tables 4 and 5, IG-FinFET circuit implementation achieves on average 25% reduction in cell area.

B. Delay Extraction

For a simple logic gate, delay can be estimated as the time difference between 10% and 90% of the voltage swing. For example, if we are trying to bring node A from 0V to 1V, then the delay is the time it takes for node A to go from 0.1V to 0.9V. In our simulation, we assumed that all the intermediate nodes have 1fF capacitance while the output nodes and input buses have 5fF capacitance.

Simulation results show that SG-FinFET static CMOS logic circuits are approximately 2.2 times faster than

Table 4. Summary of majority gate implementations.

FinFET	SG-FinFET	SG-FinFET	IG-FinFET
Logic Style	CMOS	Pass-Trans.	CMOS
# Transistor	10	9	6
Area (nm ²)	475200	529200	348300
F. Delay (ps)	10.913	31.227	24.270
R. Delay (ps)	14.204	31.883	70.314
Power (uW)	44.7	57.5	28.0

Table 5. Summary of 2-1 MUX implementations.

FinFET	SG-FinFET	SG-FinFET	IG-FinFET
Logic Style	CMOS	Pass-Trans.	CMOS
# Transistor	8	5	4
Area (nm ²)	415800	243000	273600
F. Delay (ps)	15.249	20.148	28.132
R. Delay (ps)	9.079	28.604	49.769
Power (uW)	31.5	41.4	22.0

SG-FinFET pass-transistor logic circuit and 3.5 times faster than IG-FinFET static CMOS logic circuit.

C. Power Extraction Consideration

We will not consider dynamic power consumption on output nodes and input buses because in our simulations, outputs switch same number of times on these nodes. The sole exception is the SG-FinFET pass-transistor logic implementation for majority gate, which has 5 inputs (A, A', B, C, C') compare to 3 inputs (A', B', C') of the other two implementations. The extra inputs lead to more dynamic power consumption for SG-FinFET pass-transistor logic implementation, but since it already has the highest active and leakage power consumption, omitting dynamic power consumption calculation will not change the fact that SG-FinFET pass-transistor logic circuits are usually inferior in term of power consumption.

Active power is the power consumed when both pull-up and pull-down network are active, creating a direct current path from V_{DD} to ground, while leakage power is the power consumed when charges "leak" through a transistor that is off. Calculating the active and leakage components of power consumption separately is very difficult. Therefore, we will calculate the aggregated power consumption by first calculating instantaneous power $P(t) = V(t) \cdot I(t)$, then sum up $P(t)$ for all times (in our simulation, $t: 0 \sim 1000ps$), which will give us the total energy consumed for this operation. This is easy to do since Sentaurus can provide information about voltage and current across each transistor at any time. Finally, sum up the energy consumption for all transistors and divide by clock period to obtain the active and leakage power consumption, which is listed in Table 4 for majority gate and Table 5 for 2-1 MUX.

Note that the input sequence in our simulation implies that input C (for majority gate) or input S (for 2-1 MUX) switches most frequently, while input A switches least frequently. The placement of inputs in a circuit has some impacts on power consumption [16]. For consistency, we place the least frequent switching input closest to the output and the most frequent switching input farthest from the output.

Simulation results show that IG-FinFET static CMOS logic circuits consume least amount of power. In comparison, SG-FinFET static CMOS logic circuits consume about 52% more power and SG-FinFET pass-transistor logic circuits consume about 98% more power.

V. CONCLUSION

FinFET not only has superior performance over bulk silicon MOSFET, but is primed to take over bulk silicon MOSFET as the dominant transistor choice for sub-45nm technology. In this paper, we present a methodology for synthesizing logic circuits using independent-gate FinFETs, which leads to reduction in number of transistors (and chip area) needed to implement circuits. The methodology is based on the existing MOSFET circuit synthesis methodology and can be readily adapted for independent-gate FinFETs by making minor modifications. Simulation results show that compared to other logic implementations, FinFET logic circuits achieve significant area and power reduction without voltage or transistor scaling, even though they suffer greatly in circuit speed.

REFERENCES

- [1] H-S.P. Wong, D. Frank and P. Solomon, "Device design considerations for double-gate, ground-plane, single-gated ultra-thin SOI MOSFETs at the 25 nm channel length generation," in Tech. Digest IEDM 1998, San Francisco, CA, pp. 407-410.
- [2] S. Christoloveanu, T. Ernst, D. Munteanu and T. Ouisse, "Ultimate MOSFETs on SOI: Ultra thin, single gate, double gate, or ground plane," in Int. J. High Speed Electron. Syst., vol. 10, no. 1, pp. 217-230.
- [3] E. J. Nowak, I. Aller, T. Ludwig, K. Kim, R. V. Joshi, C.-T. Chuang, K. Bernstein and R. Puri, "Turning silicon on its edge," in IEEE Circuits Devices Mag., vol. 20, Jan.-Feb. 2004, pp. 20-31.
- [4] A. Carlson, Z. Guo, S. Balasubramanian, L.T. Pang, T.J. King Liu and B. Nikolic, "FinFET SRAM with Enhanced Read / Write Margins," in SOI Conference, 2006.
- [5] A. Muttreja, N. Agarwal and N.K. Jha, "CMOS logic design with independent-gate FinFETs," in ICCD 25th International Conference, 2007, pp. 560-567.
- [6] V. Bertacco, "Decision Diagrams and Pass Transistor Logic Synthesis," in Proc. of the ACM/IEEE International Workshop on Logic Synthesis, May 1997, pp. 1-5.
- [7] M. M. Vai, VLSI Design, Boca Raton, Florida, U.S.A: CRC Press, 2000, pp. 46-48.
- [8] Z. Kohavi, Switching & Finite Automata Theory, 2nd Ed., U.S.A.: McGraw-Hill Inc., 1979, pp. 7-2.
- [9] D. Fried, J.S. Duster and K.T. Korngay, "Improved Independent Gate N-Type FinFET Fabrication and Characterization," in IEEE Electron Device Letter, Vol. 24, No. 9, September 2003, pp. 593.
- [10] "Taurus Medici," 2009. [Online]. Available: <http://www.synopsys.com/Tools/TCAD/DeviceSimulation/Pages/TaurusMedici.aspx>. [Accessed: Dec. 5, 2009].
- [11] S. Ouchi, K. Sakamoto, K. Endo, M. Masahara, T. Matsukawa, Y.X. Liu, M. Hioki, T. Nakagawa, T. Sekigawa, H. Koike and E. Suzuki, "Variable-Threshold-Voltage FinFETs with a Control-Voltage Range within the Logic-Level Swing Using Asymmetric Work-Function Double Gates," in VLSI Technology, Systems and Applications, 2008.
- [12] R.S. Muller and T.I. Kamins, Device Electronics for Integrated Circuits, 3rd Ed, U.S.A: John Wiley & Sons, 2003, pp. 386.
- [13] "Sentaurus Device," 2009. [Online]. Available: <http://www.synopsys.com/Tools/TCAD/DeviceSimulation/Pages/SentaurusDevice.aspx>. [Accessed: Jun. 1, 2009].
- [14] K. Bernstein, C. T. Chuang, R. Joshi and R. Puri, "Design and CAD challenges in sub-90 nm CMOS technologies," in ICCAD, Nov.2003, pp. 129-136.
- [15] S. A. Tawfik and V. Kursun, "FinFET domino logic with independent gate keepers", Microelectronics Journal, March 2009.
- [16] L. Benini and G.D., Micheli, "Dynamic Power Management", 1st ed. Springer, Nov 1997, pp. 24-25.