A Synthesis Methodology for Ternary Logic Circuits in Emerging Device Technologies

B. Srinivasu, *Student Member, IEEE*, and K. Sridharan, *Senior Member, IEEE*

***Abstract*— Automatic synthesis of digital circuits has played a key role in obtaining high-performance designs. While considerable work has been done in the past, emerging device technologies call for a need to re-examine the synthesis approaches, so that better circuits that harness the true power of these technologies can be developed. This paper presents a methodology for synthesis applicable to devices that support ternary logic. We present an algorithm for synthesis that combines a *geometrical representation with unary operators of multivalued logic*. The geometric representation facilitates scanning appropriately to obtain simple sum-of-products expressions in terms of unary operators. An implementation based on *Python* is described. The power of the approach lies in its applicability to a wide variety of circuits. The proposed approach leads to the savings of 26% and 22% in transistor-count, respectively, for a ternary full-adder and a ternary content-addressable memory (TCAM) over the best existing designs. Furthermore, the proposed approach requires, on an average, less than 10% of the number of the transistors in comparison with a recent decoder-based design for various ternary benchmark circuits. Extensive HSPICE simulation results show roughly 92% reduction in power-delay product (PDP) for a 12 × 12 TCAM and 60% reduction in PDP for a 24-ternary digit barrel shifter over recent designs.**

***Index Terms*— Synthesis methodology, ternary logic, ternary digit (Trit), cube representation, unary operators, emerging devices.**

I. Introduction

THE automatic synthesis of digital circuits meeting various design objectives has been actively pursued during the last few decades. Several synthesis tools have been developed that can transform high level descriptions into logic designs while also incorporating ways to decrease circuit area and delay. These tools have helped accelerate the development of application-specific integrated circuits. They have also played an important role in general-purpose processor designs [1].

However, emerging device technologies present new opportunities for a relook at the synthesis approaches since these technologies often have special characteristics. Some are based on new computational paradigms [2] while others support new device models [3], [4]. Existing software tools, in general, either do not have features for handling these technologies or may not take advantage of the special characteristics to output efficient circuit designs.

Manuscript received July 5, 2016; revised February 4, 2017; accepted March 16, 2017. This paper was recommended by Associate Editor F. J. Kurdahi.

The authors are with the Department of Electrical Engineering, IIT Madras, Chennai 600036, India (e-mail: sridhara@iitm.ac.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2686446

Our interest in this paper is on a synthesis methodology for devices that support multivalued logic. Considerable research has been done on multivalued logic during the last few decades [5]–[8]. A recent tutorial [9] on multivalued logic captures the various aspects of current interest to the circuits community. Multivalued logic offers several advantages. In particular, as the radix for computation increases, the interconnection complexity is reduced since more information is provided per line. Also, circuit-level benefits exist. For instance, 14-bit binary addition can be done (roughly) by a 9-trit adder. However, the non-availability of appropriate devices to realize ternary and other multivalued logic systems has for long remained a concern [10] but the emergence of several new device technologies [3], [11], [12] has led to renewed interest in ternary and quaternary logic in particular. Synthesis techniques are still at an early stage and as observed in [13], novel logic synthesis methods for emerging devices would help generate better circuits and facilitate exploiting the true power of these technologies.

We develop a synthesis technique applicable to devices such as quantum dot gate field effect transistor (qFET) [3], carbon nanotube FET (CNTFET) [11] and finFET [12] that support ternary logic. Research on ternary function minimization has been pursued as early as 1968 [14]. A graphical approach for three variables based on a ternary cube is reported in [14] where the author also presents algebraic relationships to facilitate minimization of functions of four (or more) variables. However, the focus in [14] is only on function minimization and not on the circuit realization. There are also prior efforts on design of basic ternary arithmetic circuits. Halpern and Yoeli [15] present details of a ternary full-adder while ternary memory units have been studied in [16]. An approach for computer minimization of multi-valued switching functions is reported in [17]. A theorem-proving based synthesis procedure for combinational logic is described in [18]. An algorithm called *KUAI-EXACT* is reported for multi-valued logic minimization in [19]. Wang *et al.* [20] present the notion of algebraic division for multi-level synthesis of multi-valued logic circuits. Recently, a universal set of multivalued logic gates has been proposed [21]. Transistor-level designs for specific operations in the ternary setting have been briefly explored. Mateo and Rubio [22] present the design of a 5 × 5 trit Wallace-tree based multiplier in CMOS. Ternary single digit addition in the context of CNTFET-based realization has been studied in [23]–[26]. Dedicated synthesis methodologies for ternary logic directed towards transistor-count reduction for different types of circuits, however, appear to be scarce.

The proposed synthesis procedure works in two stages. In the first stage, we develop an algorithm to express a given ternary function in terms of (of a small number of) unary operators of multivalued logic [27]. The output in terms of unary operators is appropriate across technologies that support ternary logic. In the second stage, we take advantage of device-specific properties (and in particular, the relationship between unary operators and transistor count) and get a low-transistor count design for the technology.

While there are some prior efforts on utilizing unary operators for adder designs [25], [28], their potential is yet to be fully explored. The work described here is motivated by the cube representation in [14]. We combine unary operators with the cube representation to arrive at a synthesis procedure. In particular, we develop expressions for various ternary functions of two and three variables in terms of unary operators. One or more layers (some layers constitute faces) of the cube participate in the synthesis process and our procedure determines an appropriate direction of 'scan' of the layers for ternary functions of two variables. Ternary functions of three (and more) variables are handled by a decomposition procedure. The notion of decomposition itself is not new. Early work in the context of programmable logic array-based synthesis is reported in [29]. We present an algorithm for decomposition of an n-variable ternary function. We then present an algorithm for transistor-level synthesis in terms of unary operators and apply the algorithm to various types of circuits. We have developed a synthesis tool in *Python* that takes as input a truth table and outputs a sum-of-products expression in terms of unary operators. This can be further optimized based on the exact realization technique adopted (as shown in section V). The proposed approach is applicable to a wide variety of circuits and handles a large number of variables.

We present case studies for the proposed synthesis methodology in CNTFET technology. CNTFETs provide the possibility of realizing two distinct threshold voltages merely by use of different diameters of the carbon nanotube [23]. We present efficient CNTFET-based circuits for realization of select unary operators. The operators are then used to obtain ternary multiplexer-based designs of various circuits. Based on these designs, a ternary full-adder is developed. This is followed by design of a ternary barrel shifter to establish the usefulness of our approach for circuits *with a large number of inputs*. The performance of the proposed approach is also studied for various ternary benchmark circuits presented in [8]. We also investigate design of a CNTFET-based ternary content addressable memory (CAM). Ternary CAMs have been studied recently in the context of network search engines [30] and the complexity in terms of transistor-count has been of interest.

Comparative studies with state-of-the art ternary circuits indicate that the methodology proposed is very promising. The approach leads to a reduction of 26% in transistor-count for a ternary full-adder over the design in [26]. Further, the proposed ternary CAM requires 22% fewer transistors than the design in [31]. Also, a 24-trit barrel shifter here requires 12% fewer transistors than the shifter described in [3]. Further, the proposed approach requires, on an average, less than 10% of the number of the transistors in comparison to a prior decoder-based design [25] for various ternary benchmark circuits.

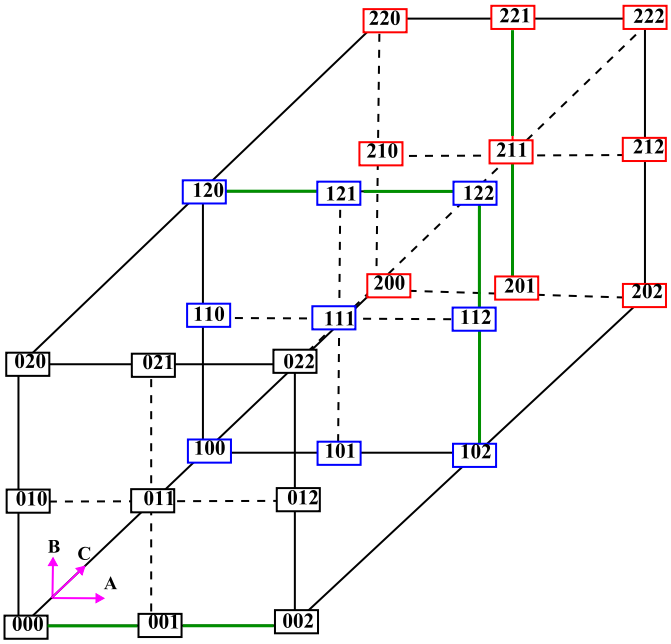


Fig. 1. Three-input function shown via a cube; here 000 corresponds to $c_0b_0a_0$; the black, blue and red colored squares correspond to $c = 0, 1$ and 2 respectively.

Comparisons of transistor counts with CMOS-based binary logic designs are also encouraging. In particular, a proposed (27:3) priority encoder takes about 35% transistors in comparison to an equivalent binary priority encoder [32] while the proposed ternary CAM takes about 86% of the transistors of an equivalent binary design [31]. In addition, detailed HSPICE simulations using the MOSFET-like CNTFET library described in [4] reveal that the proposed ternary full-adder and ternary CAM lead to savings in power-delay product (PDP) of approximately 64% and 94% over the designs in [26] and [31] respectively. In addition, the proposed ternary barrel shifter (24-trit) has a PDP that is approximately 40% of that of the shifter design in [3].

II. Preliminaries

A geometric interpretation of ternary functions can be obtained using the cube structure [14] shown in Fig. 1. Layers of the cube may correspond to faces or entities at *unit-distance* from the faces. There are 3^3 (or 27) distinct minterms for ternary functions of three ternary variables, namely a, b and c and these are depicted in Fig. 1. The cube representation is useful for reducing ternary functions of three variables. For example, consider the function given by $a_0\ b_0\ c_0 + a_1\ b_0\ c_0 + a_1\ b_0\ c_2 + a_2\ b_0\ c_0 + a_2\ b_1\ c_1 + a_1\ b_1\ c_2 + a_0\ b_2\ c_1 + a_1\ b_2\ c_1 + a_1\ b_2\ c_2 + a_2\ b_2\ c_1 + a_2\ b_0\ c_1$. By inspection of the cube, we note that the function reduces to $a_0\ c_0 + b_2\ c_1 + a_1\ c_2$. Fig. 1 shows in green color the "lines" that are relevant for the reduction.

Unary operators play a valuable role in digital design. There are several (27) unary operators in the ternary case [27].

TABLE I
TRUTH TABLE OF THE UNARY OPERATORS

A	A_P	A_N	$\overline{A}$	A^1	A^2	A_0	A_1	A_2
0	2	2	2	1	2	2	0	0
1	2	0	1	2	0	0	2	0
2	0	0	0	0	1	0	0	2

TABLE II
RELATIONS BETWEEN UNARY OPERATORS

$A_0 + A_1 = A_P$	$A_1 + A_2 = \overline{A_N}$
$A_N + A_P = A_P$	$A_0 + A_2 = \overline{A_1}$
$A_P + \overline{A_P} = 2$	$A_N + \overline{A_N} = 2$
$A_P + \overline{A_N} = 2$	$A_N + \overline{A_P} = \overline{A_1}$
$A_P + A^1 = A_P$	$A_N + A^1 = A_P$
$A_P + \overline{A} = A_P$	$A_N + \overline{A} = \overline{A}$
$A_P + A^2 = 1 + A_P$	$A_N + A^2 = A^2$
$\overline{A_N} + A^2 = 2$	$\overline{A_P} + A^2 = \overline{A_1}$
$\overline{A} + A^1 = A_P$	$\overline{A_N} + A^1 = 1 + \overline{A_N}$
$A_P \cdot A^1 = A^1$	$A_P \cdot A^2 = A_N$
$A_P \cdot \overline{A} = \overline{A}$	$A_N \cdot A^1 = 1 \cdot A_N$
$A_N \cdot A^2 = A_N$	$A_N \cdot \overline{A} = A_N$
$A \cdot 1 = 1 \cdot \overline{A_N}$	$A \cdot 2 = A$

The choice in Table I is motivated by the following. A_0, A_1 and A_2 are used in [14] while A_P and A_N are the positive and negative ternary inverters discussed in [16]. In addition, we have *cycle operators* A^1 and A^2 and the complement of A. Table II lists the various relationships among the operators. It is conventional to specify ternary functions in terms of the *min* operator [25]. For example, $F = 1 \cdot a_1 b_1 + 2 \cdot a_2 b_1$ denotes a ternary function with the prefixes 1 and 2 indicating that we are operating in ternary. This notation will be used throughout the paper.

III. KEY IDEAS IN THE PROPOSED SYNTHESIS METHODOLOGY

The proposed synthesis technique uses the graphical three-dimensional representation of ternary variables [14] shown in Fig. 1. We derive a set of results that will permit us to develop a procedure that can be automated for ternary logic synthesis.

We seek a sum-of-products expression with few terms (each of which contains a unary operator from the list in Table I). One approach is to introduce the unary operators one by one and check. However, to reduce complexity, it is advantageous to introduce the operators in a specific order. The order is identified by the representation in Fig. 1. We begin with 'scanning-related' results for the case of two-variable ternary functions.

A. The Case of Two Variables

Fig. 2 represents one layer of the cube (Fig. 1) for handling two-variable functions. Three results pertaining to the graphical representation are presented via Propositions 1, 2 and 3. The proofs of these are given in the Appendix.

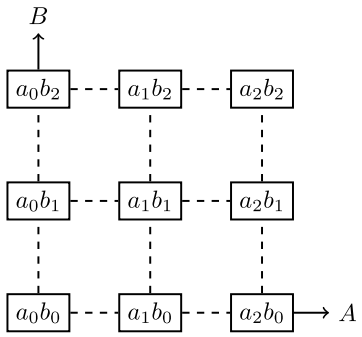


Fig. 2. One layer of the cube in Fig. 1 for handling two-variable functions.

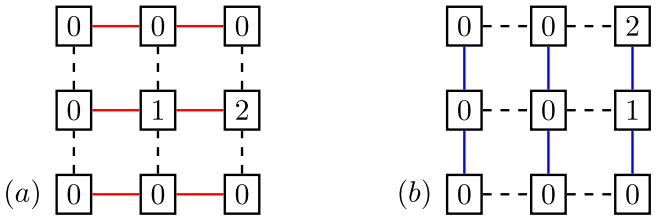


Fig. 3. (a) Scenario to select unary operators along horizontal direction (b) Scenario to select unary operators along vertical direction.

Proposition 1: Given inputs $A = a_2\ a_1\ a_0$ and $B = b_2\ b_1\ b_0$, the output in sum-of-products form is given by $F = A$ or $F = B$, depending on whether the corresponding terms in the rows or columns match.

Proposition 2: Consider a layer of the graphical representation in Fig. 2. Suppose the set of points corresponding to the ternary function evaluating to zero are along two horizontal lines. Then, a sum-of-products realization with just one minterm is obtained by choosing the unary operators along the horizontal direction.

Remark 1: Fig. 3(a) shows the case corresponding to Proposition 2. Instead of B, if we had chosen A as the 'common signal', it would lead to Equation (1) which involves extra circuit elements to derive unary operators $(1 \cdot B_1)$ and (B_1).

$$\begin{aligned} F &= 1 \cdot a_1 b_1 + 2 \cdot a_2 b_1 \\ &= a_0 \cdot 0 + a_1 \cdot [0 \cdot b_0 + 1 \cdot b_1 + 0 \cdot b_2] \\ &\quad + a_2 \cdot [0 \cdot b_0 + 2 \cdot b_1 + 0 \cdot b_2] \\ &= a_1 \cdot (1 \cdot B_1) + a_2 \cdot B_1 \end{aligned} \tag{1}$$

Corollary 1: Suppose the set of points corresponding to the ternary function evaluating to zero are along two vertical lines. Then, a sum-of-products realization with one minterm is obtained by choosing the unary operators along the vertical direction. This is illustrated in Fig. 3 (b).

Proposition 3: Suppose a ternary function evaluates to zero along the horizontal and vertical directions as shown in Fig. 4. Then, the reduced sum-of-products expression is obtained by scanning along both the directions and taking the one with least complexity.

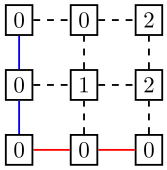


Fig. 4. Ternary function evaluating to zero along horizontal and vertical directions.

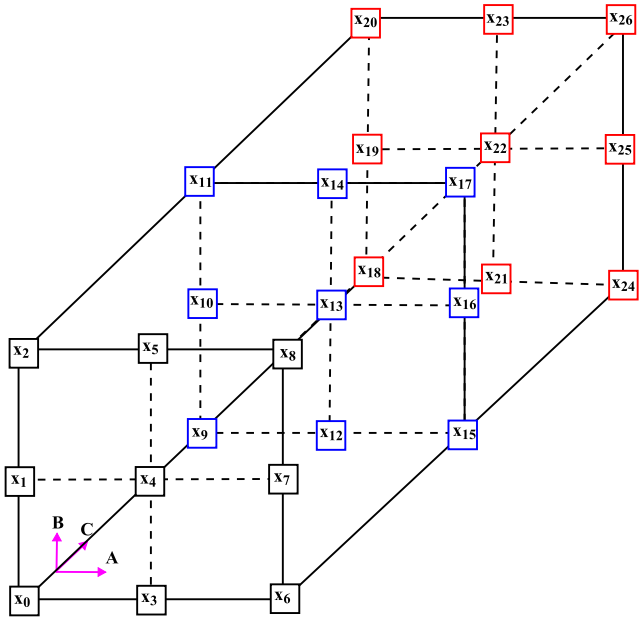


Fig. 5. Cube representation of a three variable function x_i, $i = 0, 1, 2, \cdots, 26$.

Proposition 4: A ternary logic function in sum-of-products form with unary operators can be minimized further by using the properties of the unary operators given in Table II.

Proof: We show this via an illustration. Several other possibilities also exist. Consider a ternary logic function Y given by Equation (2).

$$\begin{aligned} Y &= B_0 \cdot A^1 + B_1 \cdot A^1 + B_2 \cdot (1 + A^1) \\ &= (B_0 + B_1) \cdot A^1 + B_2 \cdot (1 + A^1) \end{aligned} \tag{2}$$

Using the property that $B_0 + B_1 = B_P$ and $B_2 = \overline{B_P}$, this equation can be rewritten as Equation (3).

$$\begin{aligned} Y &= B_P \cdot A^1 + \overline{B_P} \cdot (1 + A^1) \\ &= (B_P + \overline{B_P}) \cdot A^1 + B_2 \cdot 1 \\ &= 2 \cdot A^1 + B_2 \cdot 1 \end{aligned} \tag{3}$$

Q.E.D.

B. The Case of Three Variables

In this case, we have 27 minterms and any ternary function can be represented as a sum of these. Denoting the 27 'outputs' by x_i, $i = 0, 1, \cdots, 26$, we have the graphical representation shown in Fig. 5. A three-variable ternary function can be expressed as a sum-of-product expression by decomposition. The calculation of decomposition starts by considering one of the three variables as the 'common signal' to obtain three sets of two variable functions. In particular, we have three sets (one each for A, B and C) where a given set has three layers (for $A = 0, 1, 2$; similarly for B and C). Altogether, we have nine layers of two variable functions. The decomposed two variable functions are shown along direction B in Fig. 6 and along direction C in Fig. 7. Let P_i denote the sum of the x_i in the decomposed functions along direction C while Q_i and R_i have similar meaning with respect to directions B and A respectively. The complete list is given in Table III. We now present three results (as Propositions 5, 6 and 7) on choosing an appropriate direction for decomposition so that the number of minterms is small. The proofs of these appear in the Appendix.

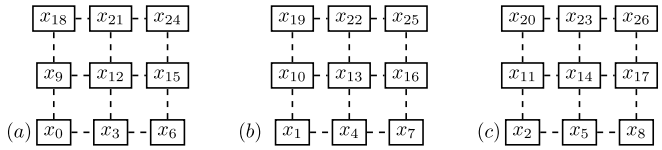


Fig. 6. Decomposition of a three-variable function into three layers of two-variable functions with B as common signal; (a) $B = 0$, (b) $B = 1$ and (c) $B = 2$.

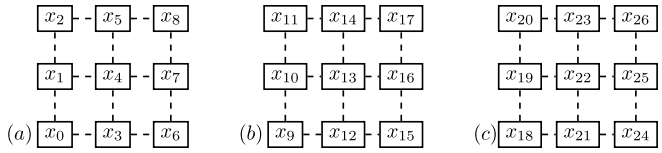


Fig. 7. Decomposition of a three-variable function into three layers of two-variable functions with C as common signal; (a) $C = 0$, (b) $C = 1$ and (c) $C = 2$.

TABLE III
DEFINITION OF P_i, Q_i AND R_i

$P_0 = \sum_{i=0}^{8} x_i$	$Q_0 = \sum_{i=0}^{8} x_{3i}$
$P_1 = \sum_{i=9}^{17} x_i$	$Q_1 = \sum_{i=0}^{8} x_{3i+1}$
$P_2 = \sum_{i=18}^{26} x_i$	$Q_2 = \sum_{i=0}^{8} x_{3i+2}$
$R_0 = \sum(x_0, x_1, x_2, x_9, x_{10}, x_{11}, x_{18}, x_{19}, x_{20})$	
$R_1 = \sum(x_3, x_4, x_5, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23})$	
$R_2 = \sum(x_6, x_7, x_8, x_{15}, x_{16}, x_{17}, x_{24}, x_{25}, x_{26})$	

Proposition 5: Consider a three-variable ternary logic function with output x_i, for $i = 0, 1, \cdots, 26$. where the condition $(Q_0 = 0 | Q_1 = 0 | Q_2 = 0)$ is satisfied. Then, decomposition of the three-variable function into two-variable functions along direction B leads to a sum-of-products expression with only six minterms.

Remark 2: Proposition 5 suggests that decomposition along other directions is not advantageous for the case specified $(Q_0 = 0 | Q_1 = 0 | Q_2 = 0)$. For instance, the decomposition along direction A leads to a function of the form $a_0 \cdot f_0(B, C) + a_1 \cdot f_1(B, C) + a_2 \cdot f_2(B, C)$, where f_0, f_1 and f_2 are sum-of-products of two-variable functions decomposed along $A = 0, 1, 2$ respectively. This results in a sum-of-products expression with nine minterms.

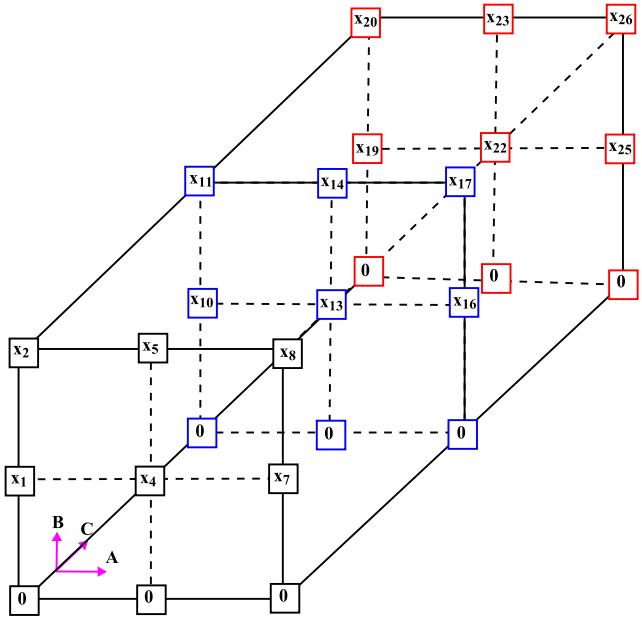


Fig. 8. Cube representation of a three variable function with some elements equal to zero.

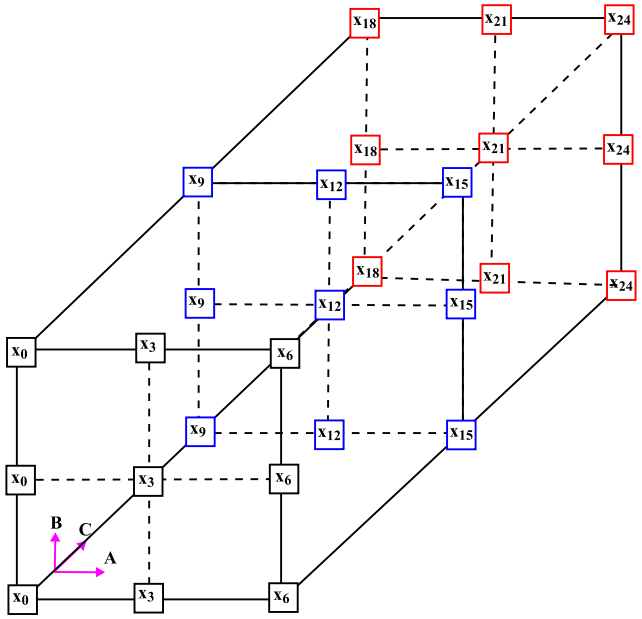


Fig. 9. Cube representation of a three-variable function with similar elements along the vertical direction.

Corollary 2: Proposition 5 can be extended to other cases and the direction X to be chosen (to get six minterms) is given by Equation (4).

$$\begin{aligned} &if\ (P_0 = 0 | P_1 = 0 | P_2 = 0), \quad then\ X = C \\ &if\ (R_0 = 0 | R_1 = 0 | R_2 = 0), \quad then\ X = A \end{aligned} \tag{4}$$

Proposition 6: Consider a three-variable ternary logic function with output x_i, for $i = 0, 1, \cdots, 26$, where the condition $(Q_0 + Q_1 = 0 | Q_1 + Q_2 = 0 | Q_2 + Q_0 = 0)$ is satisfied. Then, the decomposition of the three-variable function along direction B leads to a sum-of-products expression with just three minterms.

Corollary 3: Proposition 6 can be applied in a similar fashion to other cases. The direction X to be chosen to obtain three minterms is given by Equation (5)

$$\begin{aligned} &if\ (P_0 + P_1 = 0 | P_1 + P_2 = 0 | P_2 + P_0 = 0), \quad then\ X = C \\ &if\ (R_0 + R_1 = 0 | R_1 + R_2 = 0 | R_2 + R_0 = 0), \quad then\ X = A \end{aligned} \tag{5}$$

Proposition 7: Consider a three variable ternary logic function. Suppose the elements along a direction are the same (as shown in Fig. 10), then decomposition of the function along direction B (or C) leads to a sum-of-products expression with just three minterms.

Corollary 4: Analogous to Proposition 7, when the elements along the horizontal direction are the same as shown in Fig. 9, we can decompose along direction A (or C) as shown in Fig. 11. This leads to layers of two-variable functions with a final sum-of-product expression containing just three minterms.

Remark 3: Propositions 5 and 6 can be extended to check for a layer with complete 1's and 2's, which can be written directly as $F = 1$ or $F = 2$ respectively. Similarly, Proposition 7 and Corollary 4 can be applied to check similarity between two layers.

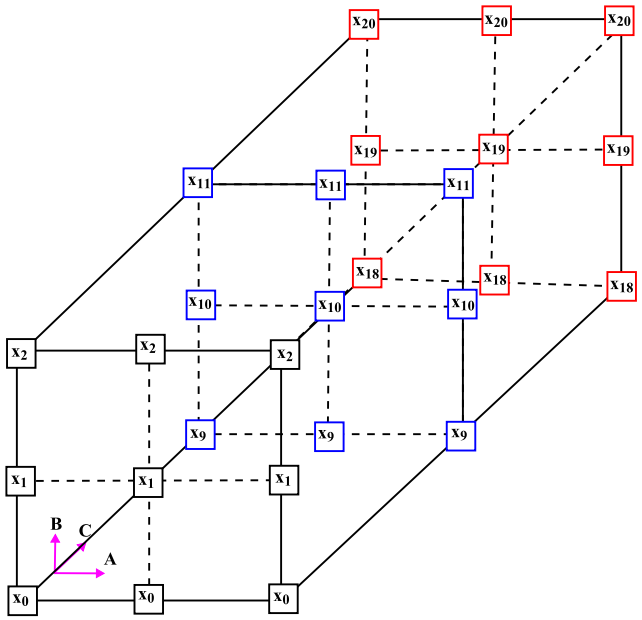


Fig. 10. Cube representation of a three variable function with similar elements along the horizontal direction.

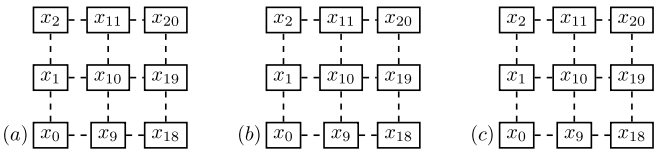


Fig. 11. Decomposition along direction A of a three-variable function (in Fig. 9) when the elements along horizontal axis are same; three layers of A: (a) $A = 0$, (b) $A = 1$ (c) $A = 2$.

Algorithm 1 Ternary Decomposition

Input: Truth table of the n-variable ternary logic function as $(X = x_0, x_1, \cdots, x_{m-1})$ where X is arranged as per Tables IV and V (and m $= 3^n$).
Output: $\frac{m}{9}$ two variable layers for an n-variable ternary function.

begin
  if $(\sum_{i=0}^{(m-1)} x_i = 0)$ **then**
    No decomposition required (since it is a zero function).
  else if *(*$A(X,1) = 1$ *or* $A(X,2) = 2$*)* **then**
    No decomposition required (since it is a constant function).
    `/* A(X,k) denotes AND of all elements of X with k */`
  else
    if *C(0,X) or C(1,X) or C(2,X)* $\geq \frac{m}{3}$ **then**
      `/* C(u,X) is count of u in X */`
      for $k \leftarrow$ *0 to (n-1)* **do**
        if $\sum J_k = 0$ *or* $\sum K_k = 0$ *or* $\sum L_k = 0$ *or* $A(J_k,1) = 1$ *or* $A(K_k,1) = 1$ *or* $A(L_k,1) = 1$ *or* $A(J_k,2) = 2$ *or* $A(K_k,2) = 2$ *or* $A(L_k,2) = 2$ **then**
          `/* J, K and L are defined in Tables IV and V */`
          Decompose along the direction of input I_k.
        else if k=n **then**
          Check for symmetry. (S=1)
      end
  end
  if *S=1* **then**
    for $k \leftarrow$ *0 to (n-1)* **do**
      if $C(i, J_k) = C(i, K_k) = C(i, L_k)$ **then**
        Decompose along I_k.
        Break
      else if k=n **then**
        Decompose along any direction.
    end
  end
  Proceed to step 2 setting n to $n-1$ (and repeat until $n = 2$).
end

Algorithm 2 SOP Unary Operator Synthesis

Input: Truth table of the n-variable ternary logic function $(x_0, x_1, \cdots, x_{m-1})$.
Output: Function (Y) expressed in terms of sum of products of input and unary operators of the inputs.

begin
  Decompose the n-variable function as per **Algorithm 1** and get $\frac{m}{9}$ two-variable layers.
  for $k \leftarrow$ *1 to* $\frac{m}{9}$ **do**
    Get the values of H_i and V_i as per Table VI
    if $(\sum_{i=0}^{8} x_i = 0)$ **then**
      $F = 0$
    else
      if *Proposition 1 is applicable* **then**
        Write output as identity function $(F = A)$ or as a constant function $(F = 1$ or $F = 2)$.
      end
      if $((H_0 = 0$ *or* $H_1 = 0$ *or* $H_2 = 0)$ *and* $(V_0 = 0$ *or* $V_1 = 0$ *or* $V_2 = 0))$ **then**
        if $(H_0{+}H_1 = 0$ *or* $H_1{+}H_2 = 0$ *or* $H_2{+}H_0 = 0)$ **then**
          choose the unary operators along the horizontal direction (Proposition 2).
        else if $(V_0 + V_1 = 0$ *or* $V_1 + V_2 = 0$ *or* $V_2 + V_0 = 0)$ **then**
          choose the unary operators along the vertical direction (Corollary 1).
        else
          Scan in both horizontal and vertical directions (Proposition 3).
        end
      else
        Scan in both horizontal and vertical directions (Proposition 3).
      end
    end
  end
  Compute the complexity of all layers and choose the best one.
  Check if further reduction is possible using Proposition 4.
end

C. More Than Three Variables

The approach can be extended to more than three variables. For example, a five-variable function will be decomposed into multiple four-variable functions. Each four-variable function will be decomposed into several three-variable functions and so on. The detailed algorithm for decomposition of an n-variable function is presented in the next section. It is based on Propositions 5, 6, 7 and related corollaries. For a given input I_i, the decomposition is into three layers each with I_{i-1} number of inputs and these are denoted as J_i, K_i and L_i as given in Tables IV and V.

IV. Proposed Algorithms

In this section, we present two algorithms (named **Algorithm 1** and **Algorithm 2**). Algorithm 1 calculates a ternary decomposition of an n-variable function while Algorithm 2 generates a sum-of-products expression in terms of unary operators.

A. Overview of the Algorithms

Algorithm 1 takes as input an n-variable function and ascertains first that a decomposition is required and then

TABLE IV
DEFINITION OF f_x

$f_x(i,j) = (x_i, x_{i+1}, \cdots, x_j)$
$f_x((i,j),(k,l)) = (x_i, x_{i+1}, \cdots, x_j, x_k, x_{k+1}, \cdots, x_l)$
$f_x((i,j),(k,l),(m,n)) = (x_i, x_{i+1}, \cdots, x_j, x_k, x_{k+1}, \cdots, x_l, , x_m, x_{m+1}, \cdots, x_n)$

TABLE V
I_i, J_i, K_i AND L_i DEFINITIONS FOR $i = n, n-1, \cdots, 2$

I_i	J_i	K_i	L_i
I_n	$f_x(0, \frac{m}{3} - 1)$	$f_x(\frac{m}{3}, \frac{2m}{3} - 1)$	$f_x(\frac{2m}{3}, (m-1))$
I_{n-1}	$f_x((0, \frac{m}{9} - 1), (\frac{m}{3}, \frac{4m}{9} - 1), (\frac{6m}{9}, \frac{7m}{9} - 1))$	$f_x((\frac{m}{9}, \frac{2m}{9} - 1), (\frac{4m}{9}, \frac{5m}{9} - 1), (\frac{7m}{9}, \frac{8m}{9} - 1))$	$f_x((\frac{2m}{9}, \frac{m}{3} - 1), (\frac{5m}{9}, \frac{6m}{9} - 1), (\frac{8m}{9}, (m-1)))$
I_{n-2}	$f_x((0, \frac{m}{27} - 1), (\frac{3m}{27}, \frac{4m}{27} - 1), \cdots, (\frac{24m}{27}, \frac{25m}{27} - 1))$	$f_x((\frac{m}{27}, \frac{2m}{27} - 1), (\frac{4m}{27}, \frac{5m}{27} - 1), \cdots, (\frac{25m}{27}, \frac{26m}{27} - 1))$	$f_x((\frac{2m}{27}, \frac{3m}{27} - 1), (\frac{5m}{27}, \frac{6m}{27} - 1), \cdots, (\frac{26m}{27}, (m-1)))$

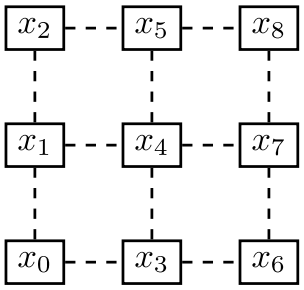


Fig. 12. One layer of the cube $(X_0, X_1, \cdots, X_8)$ for inputs in $00, 01, 02, \cdots, 21, 22$.

TABLE VI
DEFINITIONS OF H_i AND V_i

$V_0 = x_0 + x_1 + x_2$	$H_0 = x_0 + x_3 + x_6$
$V_1 = x_3 + x_4 + x_5$	$H_1 = x_1 + x_4 + x_7$
$V_2 = x_6 + x_7 + x_8$	$H_2 = x_2 + x_5 + x_8$

determines the direction of decomposition. The process is repeated until n becomes 2. Algorithm 2 considers one layer of a cube (as shown in Figures 6 and 7). It starts by assigning the truth table of one layer x_i (for $i = 0, 1, \cdots, 8$) as shown in Fig. 12 to the horizontal (H_0, H_1 and H_2) and vertical signals (V_0, V_1 and V_2) as defined in Table VI. Several cases are identified in the algorithm. These are (i) checking if the function is a constant (0, 1 or 2) or identity (ii) checking if two horizontal (and vertical) signals are zero and (iii) checking if either horizontal or vertical signal is zero.

B. Illustration for the Proposed Algorithms

A ternary 9-to-2 priority encoder is considered to illustrate the applicability of the proposed synthesis methodology to a large number of inputs. The truth table for the ternary 9-to-2 priority encoder is given in Table VII (Only a few cases are listed due to space constraints). The first step is decomposition of the n-variable function as per **Algorithm 1**. First, the algorithm leads to three 8-variable functions. Next, the three 8-variable functions are decomposed into nine 7-variable functions and so on. This decomposition continues till two-variable functions $\frac{m}{9}$ (or 3^7 two-variable layers) for each output are produced. Now we proceed with the application of **Algorithm 2** to obtain a sum-of-products expression in terms of unary operators. The resulting equations for the output of the 9 : 2 priority encoder are given in Table VIII. A ternary multiplexer based implementation is shown in Fig. 14 for Q_0. Results of implementation of a 27:3 priority encoder are presented (and compared) in Table XX.

TABLE VII
TRUTH TABLE OF A 9:2 PRIORITY ENCODER

$M8\ M7\ M6$	$M5\ M4\ M3$	$M2\ M1\ M0$	$Q_1\ Q_0$
0 0 0	0 0 0	0 0 2	0 0
0 0 0	0 0 0	0 2 X	0 1
0 0 0	0 0 0	2 X X	0 2
0 0 0	0 0 2	X X X	1 0
0 0 0	0 2 X	X X X	1 1
0 0 0	2 X X	X X X	1 2
0 0 2	X X X	X X X	2 0
0 2 X	X X X	X X X	2 1
2 X X	X X X	X X X	2 2

TABLE VIII
EXPRESSIONS FOR 9-TO-2 PRIORITY ENCODER OUTPUTS (Q_0 AND Q_1); Mi IS THE INPUT TO THE ENCODER FOR $i = 0, 1, \cdots, 8$; Mi_2 AND Mi_P ARE THE UNARY OPERATORS AS DEFINED IN TABLE I

$$
\begin{aligned}
Q_0 &= M8_2 \cdot 2 + M8_P \cdot (M7_2 \cdot 1 + M7_P \cdot (M6_P \cdot (M5_2 \cdot 2 \\
&\quad + M5_P \cdot (M4_2 \cdot 1 + M4_P \cdot (M3_P \cdot (M2_2 \cdot 2 \\
&\quad + M2_P \cdot (M1_2 \cdot 1)))))))\\
Q_1 &= M8_2 \cdot 2 + M8_P \cdot (M7_2 \cdot 2 + M7_P \cdot (M6_2 \cdot 2 \\
&\quad + M6_P \cdot (M5_2 \cdot 1 + M5_P(M4_2 \cdot 1 + M4_P(M3_2 \cdot 1)))))
\end{aligned}
$$

V. APPLICATIONS TO LOGIC SYNTHESIS IN AN EMERGING FET-BASED TECHNOLOGY

In this section, we present the applications of the proposed algorithm to synthesis of circuits made from carbon nanotube FET (CNTFET), an emerging device technology. It is worth noting that the algorithm as such is applicable to other technologies such as quantum dot gate FET [3] and finFET [12] that support ternary logic.

CNTFET exhibits ternary behaviour as indicated earlier. Due to space reasons, we omit the details and refer the reader to [23], [25], and [28]. In essence, *the ternary behaviour is*

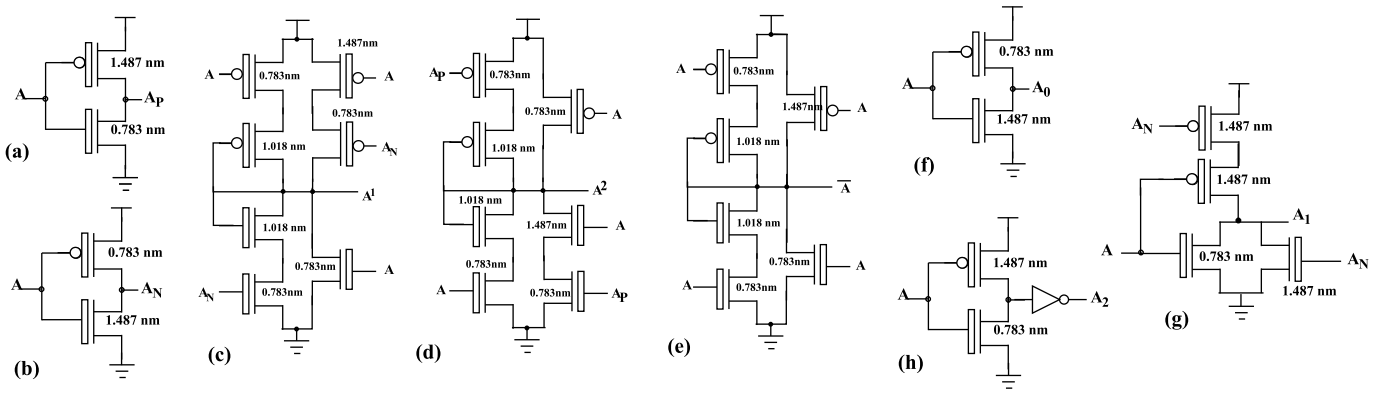


Fig. 13. CNTFET-based realization of unary operators: (a) A_P (b) A_N (c) A^1 (d) A^2 (e) $\overline{A}$ (f) A_0 (g) A_1 (h) A_2; The values 1.487, 1.018 and 0.783 correspond to diameters of the carbon nanotube and lead to ternary behaviour [23].

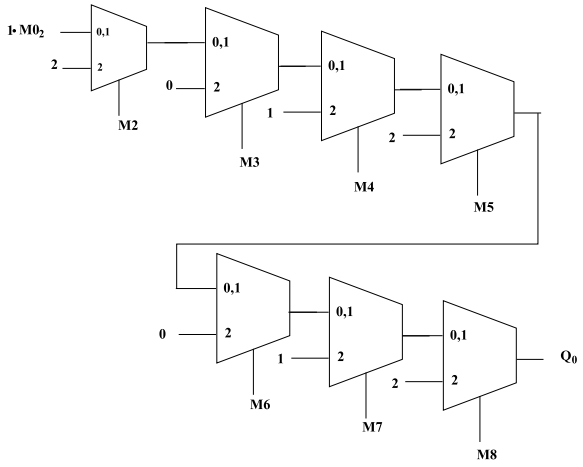


Fig. 14. 9:2 Ternary Priority Encoder Output Q_0.

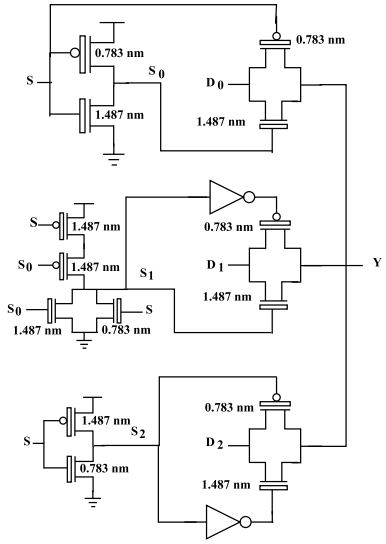


Fig. 15. CNTFET-based 3×1 Ternary Multiplexer.

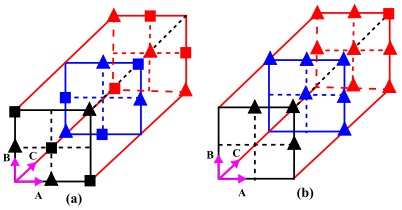


Fig. 16. Ternary Full-Adder cube representation (a) sum and (b) carry; ▲ and ■ represent 1 and 2 respectively while a blank represents a zero.

related to the chirality setting, denoted by a vector (m, n). Commonly-used [25], [28] chirality vectors are (19,0), (13,0) and (10,0) and they correspond to diameters (of the carbon nanotube) of 1.487 nm, 1.018 nm and 0.783 nm respectively. The CNTFET-based circuits for the unary operators (given in Table I) are shown in Fig. 13. A CNTFET-based realization of the ternary 3×1 multiplexer is shown in Fig. 15.

A. Ternary Full Adder

Consider a ternary full adder circuit which has three inputs A, B and C and two outputs namely, sum S and carry C_{out}. Fig. 16 depicts the sum and carry using the cube representation. Applying the proposed algorithms, we have Equations (6) and (7). The multiplexer-based realization of the ternary full adder is shown in Fig. 17.

$$\begin{aligned} sum &= c_0[b_0 \cdot A + b_1 \cdot A^1 + b_2 \cdot A^2] \\ &\quad + c_1[b_0 \cdot A^1 + b_1 \cdot A^2 + b_2 \cdot A] \\ &\quad + c_2[b_0 \cdot A^2 + b_1 \cdot A + b_2 \cdot A^1] \qquad (6) \\ carry &= c_0[b_0 \cdot 0 + b_1 \cdot (1 \cdot \overline{A_P}) + b_2 \cdot (1 \cdot \overline{A_N})] \\ &\quad + c_1[b_0 \cdot (1 \cdot \overline{A_P}) + b_1 \cdot (1 \cdot \overline{A_N}) + b_2 \cdot 1] \\ &\quad + c_2[b_0 \cdot (1 \cdot \overline{A_N}) + b_1 \cdot 1 + b_2(1 + \overline{A_P})] \qquad (7) \end{aligned}$$

Remark 4: The proposed unary operator-based ternary full-adder takes 105 CNTFETs. An alternative is using decoders and encoders as described in [25] and requires 318 CNTFETs.

B. Ternary Barrel Shifter

A barrel shifter completes multiple shifts with the same delay. The proposed ternary barrel shifter performs logical left shift depending on the signal $S1S0$. The truth table of the shifter is given in Table IX and the equations for the output of a 6-trit are given in Table X. The multiplexer-based barrel

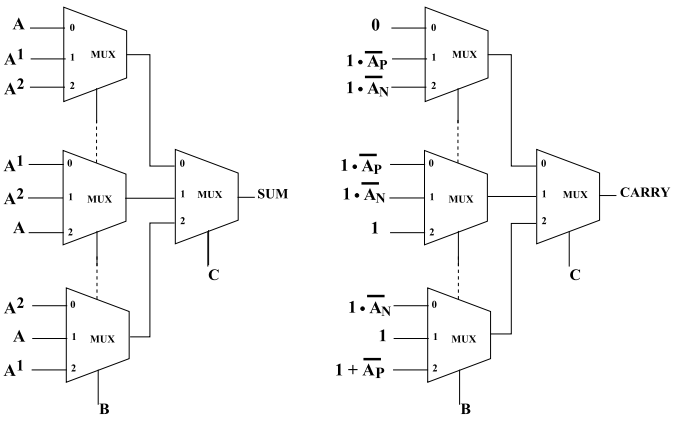


Fig. 17. Ternary MUX-based Full-Adder Using Unary Operators.

TABLE IX
TRUTH TABLE OF A 6-TRIT BARREL SHIFTER

S1	S0	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	D_5	D_4	D_3	D_2	D_1	D_0
0	1	D_4	D_3	D_2	D_1	D_0	0
0	2	D_3	D_2	D_1	D_0	0	0
1	0	D_2	D_1	D_0	0	0	0
1	1	D_1	D_0	0	0	0	0
1	2	D_0	0	0	0	0	0

TABLE X
EQUATIONS FOR 6-TRIT SHIFTER WHERE $S1$ AND $S0$ ARE SELECT SIGNALS, $D_0, D_1, \cdots, D_5$ ARE DATA AND $Y_0, Y_1, \cdots, Y_5$ ARE THE OUTPUTS

$$
\begin{array}{lcl}
Y_0 & = & S1_0 \cdot S0_0 \cdot D_0 \\
Y_1 & = & S1_0 \cdot (S0_0 \cdot D_1 + S0_1 \cdot D_0) \\
Y_2 & = & S1_0 \cdot (S0_0 \cdot D_2 + S0_1 \cdot D_1 + S0_2 \cdot D_0) \\
Y_3 & = & S1_0 \cdot (S0_0 \cdot D_3 + S0_1 \cdot D_2 + S0_2 \cdot D_1) \\
 & & + S1_1 \cdot (S1_0 \cdot D_0) \\
Y_4 & = & S1_0 \cdot (S0_0 \cdot D_4 + S0_1 \cdot D_3 + S0_2 \cdot D_2) \\
 & & + S1_1 \cdot (S1_0 \cdot D_1 + S1_1 \cdot D_0) \\
Y_5 & = & S1_0 \cdot (S0_0 \cdot D_5 + S0_1 \cdot D_4 + S0_2 \cdot D_3) \\
 & & + S1_1 \cdot (S1_0 \cdot D_2 + S1_1 \cdot D_1 + S1_2 \cdot D_0)
\end{array}
$$

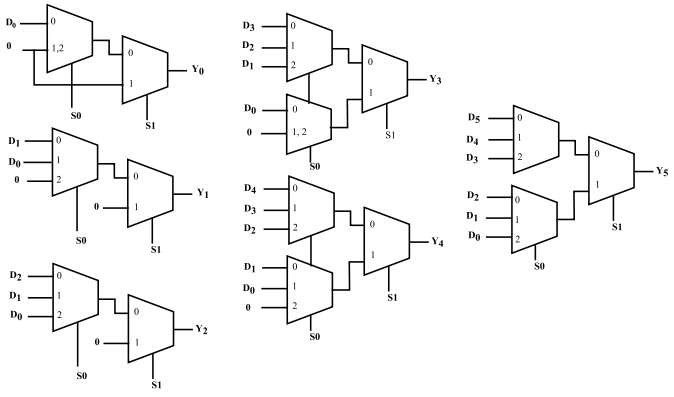


Fig. 18. 6-trit Barrel Shifter with $S1$ and $S0$ are select signals, $D_0, D_1, \cdots, D_5$ are data and $Y_0, Y_1, \cdots, Y_5$ are the outputs.

shifter is shown in Fig. 18. A feature of the circuit is the use of a ternary 2×1 multiplexer for some outputs of the shifter. The shifter can be extended to operate as a universal shifter (to perform left or right shift depending on a shift signal).

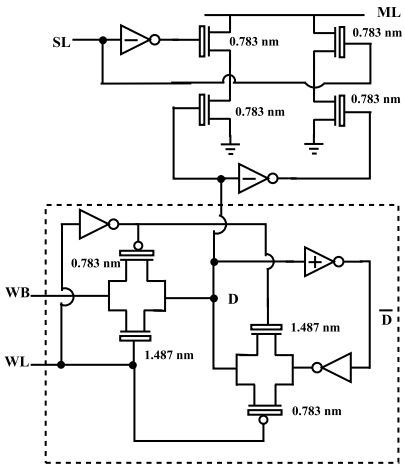


Fig. 19. CNTFET-based Ternary Content Addressable Memory (TCAM) cell; SL and ML stand for search line and match line respectively; WL and ML are two-valued while WB and SL are three-valued; Memory cell is indicated via a dashed box.

Remark 5: The shifter presented in [3] employs one decoder and three 3×1 multiplexers. Each decoder takes 16 CNTFETs while each multiplexer takes 6 CNTFETs. Altogether, the 3-trit shifter requires 34 CNTFETs while the proposed shifter takes 26 CNTFETs. The benefits for larger size barrel shifters can be seen from Table XIV.

C. Ternary CAM

Content accessible memories (CAMs) are expected to become a key part of the overall monolithic implementation [33] of various circuits. A multi-valued memory cell can reduce the number of transistors as well as the interconnections while performing more complex operations than traditional binary memory. The proposed CNTFET-based ternary content addressable memory cell (TCAM) is presented in Fig. 19. The synthesis methodology facilitates extraction of output by declaring a match as per Equation (8).

$$ML = D_0 \cdot SL_P + D_1 \cdot 2 + D_2 \cdot \overline{SL_N} \tag{8}$$

The manner in which the proposed TCAM cell works is expressed by Table XI where SL is the search line and ML is the match line. The data to be searched is placed on the search line (SL) and when the data matches, the output is obtained through the match line (ML). The data in the search line is compared with the data stored (D) in the memory cell.

Remark 6: It is assumed that the data is already prestored in the TCAM cell. The proposed synthesis methodology identifies the appropriate unary operators to facilitate search and match of data. The TCAM circuit presented in [31] uses standard ternary inverters (STIs) while the proposed circuit depicted in Fig. 19 involves a combination of binary and ternary inverters requiring fewer transistors. The use of two inverters in the feedback circuit can be justified as follows. The output of the TCAM (ML) as shown in Table XI is (logic) 2 when either the search line (SL) or data (D) is 1. When the data fed to the memory cell is 1, ($D = 1$), the outputs at

TABLE XI

TRUTH TABLE OF A TERNARY CAM; D, *SL* AND *ML* ARE DATA STORED, SEARCH LINE AND MATCHED LINE RESPECTIVELY

D	SL	ML
0	0	2
0	1	2
0	2	0
1	0	2
1	1	2
1	2	2
2	0	0
2	1	2
2	2	2

TABLE XII

TERNARY BENCHMARK CIRCUITS BASED ON [8]; VALUES OF $i \geq 8$ (AND ≤ 14) HAVE BEEN CONSIDERED

Function	Description
sum_i	sum of i ternary variables
$prod_i$	product of i ternary variables
avg_i	average of i ternary variables
${}_8cy_5$	abcde+bcdef+cdefg+defgh +efgha + fghab +ghabc + habcd
${}_{10}cy_6$	abcdef+bcdefg+cdefgh+defghi+efghij +fghija + ghijab + hijabc + ijabcd + jabcde
${}_{12}cy_7$	abcdefg+bcdefgh+cdefghi+defghij +efghijk+fghijkl + ghijklab + hijklabc + ijklabcd + klabcde + labcdef
${}_{14}cy_8$	abcdefgh+bcdefghi+cdefghij+defghijk +efghijkl +fghijklm + ghijklma + hijklmab + ijklmabc + jklmabcd + klmabcde + lmabcdef + mabcdefg
$counter_i$	Counter for number of 1's and 2's in an i-trit number

the positive ternary inverter ($\overline{D}$) and the inverter that follows are 2 and 0 respectively (in the feedback loop). The second transmission gate in the feedback loop passes the logic low (0) to the node *D*, which results in reduction in the voltage at D from logic 1 to a smaller value (that lies between 0 and 1). Hence, the voltage at D is less than the threshold voltage of the n-type CNTFETs (of the ML circuit), which causes the match line output to be high.

As observed from Fig. 19, the proposed TCAM cell takes 18 CNTFETs. The TCAM presented in [31] takes 23 CNTFETs hence, we obtain 22% savings over the existing design.

D. Multi-Input Ternary Benchmark Functions

We have also considered various benchmark functions [8] for testing the efficiency of our synthesis methodology. The functions considered are presented in Table XII. These include (i) sum_i to sum i ternary variables (ii) $prod_i$ to find the product of i ternary variables (iii) avg_i to calculate the average of i ternary variables (iv) ${}_ncy_r$ which corresponds to a ternary sum-of-product expression of n variables taken r at a time and (v) $counter_i$ which counts number of 1's and 2's in a given i trit number. For implementation purposes, we have chosen $i \geq 8$ to gather information on the performance of the proposed methodology for large number of variables. Further n and r in ${}_ncy_r$ have been chosen such that $n \geq 8$ and $r \geq 5$.

TABLE XIII

COMPARISON OF TRANSISTOR-COUNT OF TERNARY FULL ADDER AND TERNARY CONTENT ADDRESSABLE MEMORY (TCAM) CELL

Design	Transistor count
Ternary Full adder	
[25]	318
[26]	144
Proposed	105
TCAM cell	
[31]	23
Proposed	18

TABLE XIV

TRANSISTOR-COUNT COMPARISON OF BARREL SHIFTER FOR VARIOUS SIZES

Shifter Size (trit)	Design [3]	Proposed
3	34	26
6	122	94
12	372	312
24	1114	984

TABLE XV

TRANSISTOR-COUNT COMPARISON OF TERNARY CAM (SIZE IS IN $rows \times columns$)

CAM Size	Design [31]	Proposed
3×3	153	126
6×6	558	486
9×9	1215	1026
12×12	2124	1800

VI. IMPLEMENTATION AND COMPARISONS

A. Python Implementation and Transistor Count

The proposed algorithms have been implemented in Scientific PYthon Development EnviRonment (Spyder) using Python 2.7. Python has been chosen since it has powerful but simple syntax and provides a rich set of features for rapid application development. The tool developed takes a truth table as input and outputs the sum-of-products expressions along with the transistor count. We give below the details of transistor-count for various circuits. The complexity details of the proposed ternary full adder and the TCAM cell are presented in Table XIII. The proposed full adder design leads to CNTFET savings of 66% and 26% respectively over the designs in [25] and [26]. Further, the proposed TCAM cell leads to 22% savings over the design in [31] with respect to transistor count. For the complete TCAM-based search operation, the benefits of the proposed design are seen from Table XV. The proposed design leads to approximately 15% savings in the CNTFET count. Table XIV compares the proposed shifter with the design in [3]. The savings using the proposed methodology are higher for small size shifters. Python code has also been developed to study the performance of the proposed methodology on ternary benchmark circuits [8]. The results of transistor count are presented in Table XVI along with comparisons with an approach based on decoders in [25]. (the decoder-based designs in [25] are for single trit adder and multiplier and we have developed Python code for getting transistor counts for larger-size circuits corresponding to the ternary benchmarks.)

TABLE XVI

PERFORMANCE OF PROPOSED APPROACH ON TERNARY BENCHMARK CIRCUITS AND TRANSISTOR-COUNT COMPARISONS (DATA FOR THE APPROACH IN [25] WERE OBTAINED BASED ON OUR PYTHON IMPLEMENTATION)

Function	I/O	No. of MUXes	No. of CNTFETs	
			Proposed Approach	Decoder-Based [25]
sum_8	8/3	139	960	203744
sum_9	9/3	383	2346	$716*10^3$
sum_{10}	10/3	1112	6822	$2.5*10^6$
sum_{11}	11/3	3301	19968	$8.8*10^6$
sum_{12}	12/3	9862	59346	$30.5*10^6$
sum_{13}	13/3	29545	$177*10^3$	$10*10^7$
$prod_8$	8/6	137	934	12290
$prod_9$	9/6	380	2404	28990
$prod_{10}$	10/7	1110	6796	67556
$prod_{11}$	11/7	3298	19936	$155*10^3$
$prod_{12}$	12/8	9860	59320	$356*10^3$
$prod_{13}$	13/9	29544	$177*10^3$	$813*10^3$
$prod_{14}$	14/9	59422	$48*10^4$	$1.8*10^6$
avg_8	8/1	128	871	69102
avg_9	9/1	371	2341	$228*10^3$
avg_{10}	10/1	1100	6727	$748*10^3$
avg_{11}	11/1	3287	19861	$2.43*10^6$
avg_{12}	12/1	9848	59239	$7.8*10^6$
avg_{13}	13/1	29531	$177*10^3$	$25*10^6$
$_8cy_5$	8/1	140	908	46170
$_9cy_5$	9/1	378	2378	79460
$_{10}cy_6$	10/1	1107	6764	$191*10^3$
$_{11}cy_9$	11/1	3294	19898	$191*10^3$
$_{12}cy_8$	12/1	9861	61258	$2.1*10^6$
$_{13}cy_7$	13/1	29538	$177*10^3$	$8.6X10^6$
$_{14}cy_8$	14/1	88587	$531*10^3$	$12.5*10^6$
$counter_8$	8/4	141	958	$335*10^3$
$counter_9$	9/6	395	2508	$1.15*10^6$
$counter_{10}$	10/6	1129	6924	$3.81*10^6$
$counter_{11}$	11/6	3320	20098	$0.3*10^6$
$counter_{12}$	12/6	9886	59521	$38*10^6$
$counter_{13}$	13/6	29577	$177*10^3$	$38*10^6$

B. HSPICE Implementation for Power-Delay Product

Performance with respect to other metrics has been determined via SPICE simulations using the MOSFET-like CNTFET library in [4]. Tables XVII and XVIII report the comparisons for the ternary adder and TCAM with respect to worst-case delay, average power and power delay product.

Fig. 20 reports the comparison of the power-delay product of the proposed barrel shifter for various sizes. For a 24-trit barrel shifter, the PDP of the proposed approach is only 40 % of that of the existing design [3].

Figures 21 (a) and (b) present the comparison of the PDP of a TCAM (search operation) for load capacitances of $2fF$ and $3fF$ respectively. The TCAM cell in [31] is used to derive an $m \times n$ memory architecture. The plot reveals that the proposed TCAM architecture takes 7% of the PDP of the design derived from [31]. The power-delay product values for various benchmark circuits are given in Table XIX.

C. Area Comparison of Binary and Ternary Circuits

The ternary circuits synthesized using the proposed methodology have also been compared in terms of transistor count with the corresponding binary equivalent. The results are presented in Table XX. We observe that the proposed ternary barrel shifter requires only about 46% transistors in comparison to the corresponding binary design in [35]. The design

TABLE XVII

COMPARISON OF TERNARY FULL-ADDER DESIGNS WITH DIFFERENT DRIVE STRENGTHS

Design	Load (fF)	Delay (ps)	Power (μW)	PDP (fJ)
[34]	2	283.8	6.361	1.806
	3	327.1	6.719	2.197
[34]	2	261.4	19.71	5.152
	3	278.1	19.99	5.559
[26]	2	386.1	1.462	0.564
	3	569.6	1.511	0.860
[26]	2	166.1	2.209	0.367
	3	165.5	2.422	0.400
Proposed	2	127.4	1.034	0.131
	3	183.9	1.064	0.195

TABLE XVIII

COMPARISON OF TCAM DESIGNS WITH DIFFERENT DRIVE STRENGTHS

Design TCAM cell	Load (fF)	Delay (ns)	Power (μW)	PDP (fJ)
[31]	2	0.195	7.571	1.476
	3	0.254	8.025	2.038
Proposed	2	0.112	0.7671	0.085
	3	0.152	0.7915	0.121

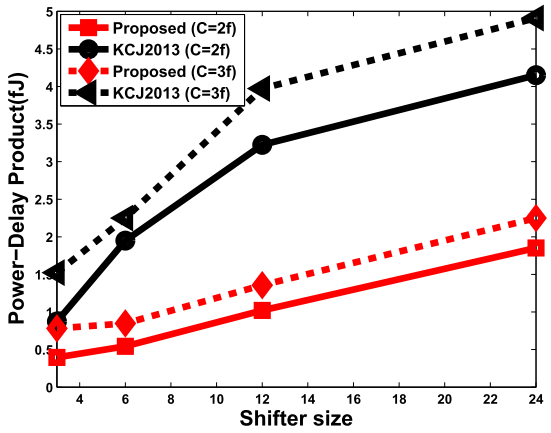


Fig. 20. Comparison of power-delay product of ternary barrel shifter for several capacitances; $KCJ2013$ corresponds to [3].

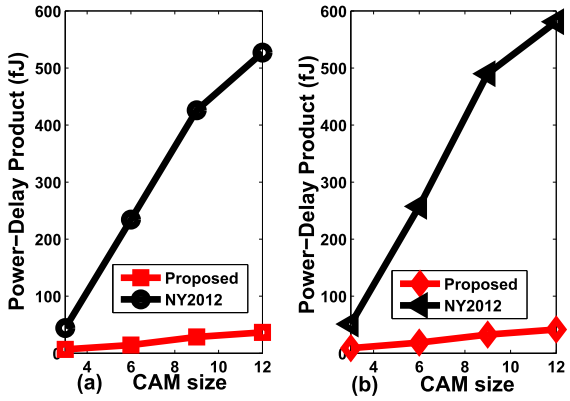


Fig. 21. Comparison of power-delay product of ternary content addressable memory for load capacitance (a) $2fF$ (b) $3fF$; $NY2012$ corresponds to [31]; CAM sizes are 3×3, 6×6, 9×9 and 12×12.

TABLE XIX
POWER-DELAY PRODUCT FOR BENCHMARK CIRCUITS [8] USING THE PROPOSED APPROACH

Function	Power-Delay Product (pJ)
sum_8	5.205
sum_{10}	7.537
sum_{12}	9.25
$prod_8$	3.781
$prod_{10}$	4.65
$prod_{12}$	6.651
$prod_{14}$	7.152
avg_8	3.17
avg_{10}	4.75
avg_{12}	5.97
${}_8cy_5$	1.377
${}_{10}cy_6$	2.251
${}_{12}cy_8$	3.15
${}_{14}cy_8$	4.902
$counter_8$	1.968
$counter_{10}$	2.815
$counter_{12}$	4.322

in [35] is based on an array of transmission gates, each of which requires two transistors. Consequently, a 32-bit shifter takes 2112 transistors (including a few transistors for the control signals). On the other hand, the proposed equivalent ternary design (24-trit) uses a combination of 2-to-1 and 3-to-1 multiplexers. Each of these multiplexers needs a small number of transistors by careful selection of unary operators. Further, the control circuitry is also simplified leading to considerable savings. A similar benefit is observed with the proposed ternary priority encoder (discussed in section IV-B). The 32-bit design presented in [32] uses several 4-input, 3-input and 2-input AND and OR gates and has a total requirement of 1106 transistors while, our equivalent 27:3 ternary priority encoder design is based on Propositions 4 and 7, and uses primarily 2-to-1 ternary multiplexers (each of which requires only two transmission gates). Careful selection of the unary operators also leads to fewer multiplexers leading to a total of 375 transistors (approximately 34% of the requirement in [32]). In addition, the proposed ternary CAM requires about 86 % (of transistors) of the binary CAM reported in [31] and this can be attributed to the use of the positive and negative ternary inverters (with lower transistor requirement than the STI). The binary CAM of size 32×32 requires 1024 CAM cells (where each cell uses 10 transistors) and 81 inverters (each requiring two transistors) leading to a total requirement of 10368 transistors. The equivalent TCAM requires only 729 cells (with each cell requiring 12 transistors) and 64 inverters. Consequently, we have an overall transistor requirement of 8910.

The proposed ternary full-adder requires considerably more transistors than a binary full-adder since the latter is optimized with respect to transmission gates. However, the difference in transistor counts becomes smaller as we consider large circuits. For example, sum_4 (sum of 4 ternary variables) takes roughly 1.9 times more transistors for ternary (than for binary) and further, there are advantages with respect to the interconnect requirement for the proposed ternary design (Table XX).

TABLE XX
COMPARISON OF TERNARY AND BINARY CIRCUITS; sum_4 CORRESPONDS TO THE SUM OF FOUR TERNARY VARIABLES

Content Addressable Memory (CAM)		
	Proposed Ternary	Binary [31]
Size of the CAM	27 X 27	32 X 32
Max. value searched	$7.6X10^{12}$	$4.29X10^9$
Number of cells	729	1024
CAM cell	12	10
Inverters required	81	64
Total No. of transistors	8910	10368
Priority Encoder		
	Proposed Ternary	Binary [32]
Size of Priority encoder	27 : 3	32 : 4
Number of transistors	375	1106
Barrel Shifter		
	Ternary	Binary [35]
Size of the shifter	24 trit	32 bit
Max. value	$282.4X10^9$	$4.29X10^9$
No. of transistors	984	2112
sum_4 (benchmark) circuit		
	Ternary	Binary [36]
No. of digits	4 trit	8 bits
Operation	2+2+... (4 times)	1+1+... (8 times)
Total No. of transistors	134	70
Total No. of interconnects	0	23
Total No. of I/O	4/2	8/4

VII. CONCLUSIONS

We have presented an approach for ternary logic synthesis in this paper with a view to application to emerging device technologies. The proposed algorithms are simple to implement. Python code has been developed and samples are provided at the URL http://www.ee.iitm.ac.in/∼sridhara/synthesis. Applications of the proposed approach for synthesis of CNTFET-based circuits are presented. The proposed approach lends itself readily to adaptation to other device technologies.

APPENDIX
PROOFS OF PROPOSITIONS

Proof of Proposition 1: This result can be established with reference to Fig. 1. Suppose the entries in the columns match. Then, we have the arrangement in Fig. 22 (a). The ternary equations can then be written as (9). Simplification and rewriting leads to Equation (10).

$$\begin{aligned} F &= 1 \cdot a_1 b_0 + 2 \cdot a_2 b_0 + 1 \cdot a_1 b_1 + 2 \cdot a_2 b_1 \\ &\quad +1 \cdot a_1 b_2 + 2 \cdot a_2 b_2 \qquad (9) \\ &= A \qquad (10) \end{aligned}$$

Along similar lines, we can show that the output is B when the entries in the rows match (as in Fig. 22 (b)). Q.E.D.

Proof of Proposition 2: We consider, without loss of generality, the ternary function $F = 1 \cdot a_1 b_1 + 2 \cdot a_2 b_1$. F can be expressed (by choosing B as the 'common signal') as given by Equation (11).

$$\begin{aligned} F &= 1 \cdot a_1 b_1 + 2 \cdot a_2 b_1 \\ &= b_0 \cdot 0 + b_1 \cdot [0 \cdot a_0 + 1 \cdot a_1 + 2 \cdot a_2] + b_2 \cdot 0 \\ &= b_0 \cdot 0 + b_1 \cdot A + b_2 \cdot 0 \\ &= b_1 \cdot A \qquad (11) \end{aligned}$$

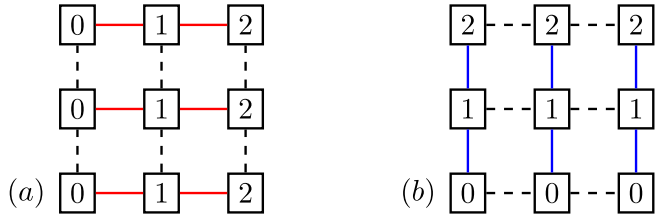


Fig. 22. (a) $F = A$ (b) $F = B$.

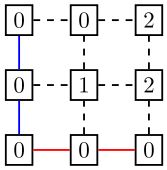


Fig. 23. Ternary function evaluating to zero along horizontal and vertical directions.

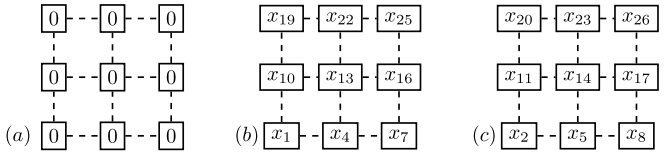


Fig. 24. Decomposition of a three-variable function into three layers of two-variable functions with B as common signal; (a) $B = 0$, (b) $B = 1$ and (c) $B = 2$.

The proof for obtaining a unary-operator based sum-of-products expression for functions $F = 1 \cdot a_1\ b_0 + 2 \cdot a_2 b_0$ and $F = 1 \cdot a_1 b_2 + 2 \cdot a_2 b_2$ is similar and hence omitted. Q.E.D.

Proof of Proposition 3: When a ternary function evaluates to zero along the horizontal and vertical directions, scanning along either of the two directions may lead to expressions with identical number of terms. Hence, the one with lower complexity (lower transistor-count) is determined based on device-specific information. For instance, in Fig. 23, scanning along the vertical direction (B) yields $a_1 \cdot\ (1 \cdot B_1) + a_2 \cdot\ (\overline{B_N})$. while scanning along the horizontal direction (A) yields $b_1 \cdot A + b_2 \cdot \overline{A_P}$ with the same number of terms. Q.E.D

Proof of Proposition 5: We consider the case of $Q_0 = 0$ (the proof for $Q_1 = 0$ and $Q_2 = 0$ is similar). $Q_0 = 0$ corresponds to the cube representation shown in Fig. 8. The decomposition of the ternary function along direction B, as shown in Fig. 24, will have only two layers of two-variable functions (in terms of A and C) corresponding to $B = 1$ and $B = 2$. We can therefore express as "$b_1 \cdot f_1(A, C) + b_2 \cdot f_2(A, C)$" where f_1 and f_2 are two-variable functions (each with three minterms) corresponding to $B = 1$ and $B = 2$ respectively. Hence, we have a sum-of-products expression with just six minterms. Q.E.D.

Proof of Proposition 6: We prove this for $Q_0 + Q_1 = 0$ (the arguments for $Q_1 + Q_2 = 0$ as well as $Q_2 + Q_0 = 0$ are similar). The decomposition along direction B results in a function $f_2(A, C)$ (with three minterms) for $B = 2$ (and 0 for $B = 0, 1$). Hence, the final expression can be written as $b_2 \cdot f_2(A, C)$. Q.E.D.

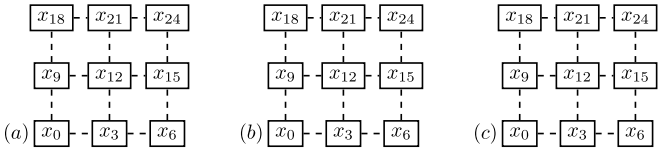


Fig. 25. Decomposition (along direction B) of three-variable function (in Fig. 10) when the elements along vertical axis are the same; three layers of B: (a) $B = 0$, (b) $B = 1$ (c) $B = 2$.

Proof of Proposition 7: Decomposition along the direction B gives the three layers shown in Fig. 25. The final expression can be written as Equation (12).

$$F = b_0 \cdot f_0(A, C) + b_1 \cdot f_1(A, C) + b_2 \cdot f_2(A, C) \quad (12)$$

These two layers in Figures 10 (a) and (b) can be represented with the same function. Equation (12) can then be simplified to (13)

$$\begin{aligned} F &= f_0(A, C) \cdot (b_0 + b_1 + b_2) \\ F &= f_0(A, C) \end{aligned} \quad (13)$$

Further, this process of decomposition leads to an expression which is independent of the third variable (B) and the sum-of-products expression has three minterms. Hence, we have the result. Q.E.D.

REFERENCES

[1] B. Stolt *et al.*, "Design and implementation of the POWER6 microprocessor," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 21–28, Jan. 2008.

[2] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, no. 4, pp. 541–557, Apr. 1997.

[3] S. Karmakar, J. A. Chandy, and F. C. Jain, "Design of ternary logic combinational circuits based on quantum dot gate FETs," *IEEE Trans. VLSI Syst.*, vol. 21, no. 5, pp. 793–806, May 2013.

[4] J. Deng and H. S. P. Wong, "A compact SPICE model for carbon-nanotube field-effect transistors including nonidealities and its application—Part I: Model of the intrinsic channel region," *IEEE Trans. Electron Devices*, vol. 54, no. 12, pp. 3186–3194, Dec. 2007.

[5] S. Kawahito, M. Kameyama, T. Higuchi, and H. Yamada, "A 32*32-bit multiplier using multiple-valued MOS current-mode circuits," *IEEE J. Solid-State Circuits*, vol. 23, no. 1, pp. 124–132, Feb. 1988.

[6] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," *Multiple-Valued Logic*, vol. 4, nos. 1–2, pp. 9–62, 1998.

[7] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 11, pp. 1645–1659, Nov. 2005.

[8] M. Hawash, M. Lukac, M. Kameyama, and M. Perkowski, "Multiple-valued reversible benchmarks and extensible quantum specification (XQS) format," in *Proc. IEEE 43rd Int. Symp. Multiple-Valued Logic*, May 2013, pp. 41–46.

[9] V. Gaudet, "A survey and tutorial on contemporary aspects of multiple-valued logic and its application to microelectronic circuits," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 1, pp. 5–12, Mar. 2016.

[10] S. L. Hurst, "Multiple valued logic: Its status and its future," *IEEE Trans. Comput.*, vol. C-33, no. 12, pp. 1160–1179, Dec. 1984.

[11] S. J. Tans, A. R. M. Verschueren, and C. Dekker, "Room-temperature transistor based on a single carbon nanotube," *Nature*, vol. 393, pp. 49–52, May 1998.

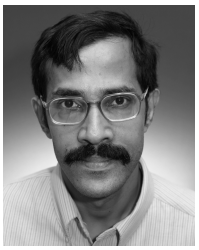
[12] M. Klein *et al.*, "Ternary logic implemented on a single dopant atom field effect silicon transistor," *Appl. Phys. Lett.*, vol. 96, no. 4, p. 043107, 2010.

[13] L. Amaru, P. E. Gaillardon, S. Mitra, and G. De Micheli, "New logic synthesis as nanotechnology enabler," *Proc. IEEE*, vol. 103, no. 11, pp. 2168–2195, Nov. 2015.

[14] S. L. Hurst, "An extension of binary minimization techniques to ternary equations," *Comput. J.*, vol. 11, no. 3, pp. 277–286, 1968.

[15] I. Halpern and M. Yoeli, "Ternary arithmetic unit," *Proc. IEE*, vol. 15, no. 10, pp. 1385–1388, Oct. 1968.
[16] H. T. Mouftah and I. B. Jordan, "Design of ternary COS/MOS memory and sequential circuits," *IEEE Trans. Comput.*, vols. C–26, no. 3, pp. 281–288, Mar. 1977.
[17] S. Y. H. Su and P. T. Cheung, "Computer minimization of multivalued switching functions," *IEEE Trans. Comput.*, vol. 21, no. 9, pp. 995–1003, Sep. 1972.
[18] W. C. Kabat and A. C. Wojcik, "Automated synthesis of combinational logic using theorem-proving techniques," *IEEE Trans. Comput.*, vol. C-34, no. 7, pp. 610–632, Jul. 1985.
[19] M. A. Perkowski, P. Wu, and K. A. Pirkl, "KUAI-EXACT: A new approach for multi-valued logic minimization in VLSI synthesis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 1989, p. 404.
[20] H. M. Wang, C. L. Lee, and J. E. Chen, "Algebraic division for multilevel logic synthesis of multi-valued logic circuits," in *Proc. 24th Int. Symp. Multiple-Valued Logic*, 1994, pp. 44–51.
[21] M. E. Romero, E. M. Martins, R. R. Santos, and M. E. D. Gonzalez, "Universal set of CMOS gates for the synthesis of multiple valued logic digital circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 3, pp. 736–749, Mar. 2014.
[22] D. Mateo and A. Rubio, "Design and implementation of a 5×5 trits multiplier in a quasi-adiabatic ternary CMOS logic," *IEEE J. Solid-State Circuits*, vol. 33, no. 7, pp. 1111–1116, Jul. 1998.
[23] A. Raychowdhury and K. Roy, "Carbon-nanotube-based voltage-mode multiple-valued logic design," *IEEE Trans. Nanotechnol.*, vol. 4, no. 2, pp. 168–179, Mar. 2005.
[24] K. Navi, M. Rashtian, A. Khatir, and P. Keshavarzian, "High speed capacitor-inverter based carbon nanotube full adder," *Nanoscale Res. Lett.*, vol. 5, no. 5, pp. 859–862, 2010.
[25] S. Lin, Y. B. Kim, and F. Lombardi, "CNTFET-based design of ternary logic gates and arithmetic circuits," *IEEE Trans. Nanotechnol.*, vol. 10, no. 2, pp. 217–225, Mar. 2011.
[26] P. Keshavarzian and R. Sarikhani, "A novel CNTFET-based ternary full adder," *Circuits Syst. Signal Process.*, vol. 33, no. 3, pp. 665–679, 2014.
[27] D. M. Miller and M. A. Thornton, *Multiple Valued Logic: Concepts and Representations*. San Rafael, CA, USA: Morgan & Claypool, 2008.
[28] M. H. Moaiyeri, A. Doostaregan, and K. Navi, "Design of energy-efficient and robust ternary circuits for nanotechnology," *IET Circuits, Devices, Syst.*, vol. 5, no. 4, pp. 285–296, Jul. 2011.
[29] T. Sasao, "Multiple-valued decomposition of generalized Boolean functions and the complexity of programmable logic arrays," *IEEE Trans. Comput.*, vol. C-30, no. 9, pp. 635–643, Sep. 1981.
[30] M. J. Akhbarizadeh, M. Nourani, D. S. Vijayasarathi, and P. T. Balsara, "A nonredundant ternary CAM circuit for network search engines," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 3, pp. 268–278, Mar. 2006.
[31] K. Nepal and K. You, "Carbon nanotube field effect transistor-based content addressable memory architectures," *IET Micro Nano Lett.*, vol. 7, no. 1, pp. 20–23, Jan. 2012.
[32] S. Abdel-Hafeez and S. Harb, "A VLSI high-performance priority encoder using standard CMOS library," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 53, no. 8, pp. 597–601, Aug. 2006.
[33] S. Mahapatra and A. M. Ionescu, "Realization of multiple valued logic and memory by hybrid SETMOS architecture," *IEEE Trans. Nanotechnol.*, vol. 4, no. 6, pp. 705–714, Nov. 2005.
[34] M. H. Moaiyeri, R. F. Mirzaee, K. Navi, and O. Hashemipour, "Efficient CNTFET-based ternary full adder cells for nanoelectronics," *Nano-Micro Lett.*, vol. 3, no. 1, pp. 43–50, 2011.
[35] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2003.
[36] J. F. Lin, Y. T. Hwang, M. H. Sheu, and C. C. Ho, "A novel high-speed and energy efficient 10-transistor full adder design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1050–1059, May 2007.

B. Srinivasu received the master's degree in micro electronics and VLSI design from the National Institute of Technology, Calicut, India, in 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, IIT Madras. His research interests include digital design in emerging device technologies and FPGA-based designs.



K. Sridharan (S'84–M'96–SM'01) received the Ph.D. degree from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1995. He was an Assistant Professor with IIT Guwahati, from 1996 to 2001. He was a Visiting Staff Member with Nanyang Technological University, Singapore, in 2000–2001 and 2006–2008. Since 2001, he has been with IIT Madras, where he is currently a Professor. He holds two patents and has published two books and approximately 85 papers in journals and conferences. His research interests include FPGA-based design, robotics, and nano architectures. He was a recipient of the 2011 Tan Chin Tuan Fellowship. He is an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.