Journal of Computing and Information Technology - CIT 15, 2007, 1, 85–92
doi:10.2498/cit.1000731

Optimum Gate Ordering of CMOS Logic Gates Using Euler Path Approach: Some Insights and Explanations

Kuntal Roy

Department of Electronics and Tele-Communication Engineering, Jadavpur University, India

The paper addresses some insights into the Euler path approach to find out the optimum gate ordering of CMOS logic gates. Minimization of circuit layout area is one of the fundamental considerations in circuit layout synthesis. Euler path approach suggests that finding a common Euler path in both the NMOS and PMOS minimizes the logic gate layout area. In this article, the minimization of layout area has been placed as equivalent to minimization of the total number of odd vertices in NMOS and PMOS networks. It has been logically proved that a MOS network will always have an even number of odd vertices. Moreover, it intuitively explains how to organize the sequence of a NMOS network when deriving the corresponding PMOS network from it, so that the number of odd vertices is minimized. The algorithm to determine the total number of Euler paths is also presented in this article.

Keywords: minimization of layout area, CMOS logic gates, Euler path, odd and even vertices, minimization of odd vertices, diffusion breaks.

1. Introduction

A CMOS circuit consists of two separate blocks – NMOS and PMOS, which is the dual of NMOS. The symbolic and stick diagram of a MOS are shown in Figure 1 and Figure 2 respectively.

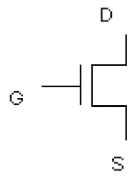

Fig. 1. Symbolic Diagram of a MOS.

In a NMOS or PMOS, diffusion intersects the polysilicon line perpendicularly. This is presented in the stick diagram of a MOS as in Figure 2. Also, the channel formed at the intersection of the two has been shown.

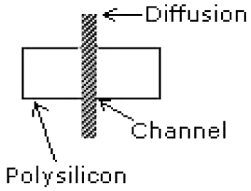

Fig. 2. Stick Diagram of a MOS.

As in [1], a MOS can be abstracted as a connected edge between two vertices as given in Figure 3.

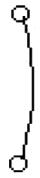

Fig. 3. Abstraction of a MOS.

A vertex may be odd or even. An odd vertex is one at which odd number of edges is connected and similarly for an even vertex. Thus, a single MOS has two odd vertices.

Now, a CMOS 2-input NOR gate will be demonstrated with its layout, stick diagram and the abstraction for the same to have ease in understanding of the readers. Accordingly, the logic function is **f** where,

$$\mathbf{f'} = \mathrm{A} + \mathrm{B}$$

f' is the complement function of **f**.

The corresponding layout and stick diagram have been shown in Figure 4 and Figure 5 respectively.

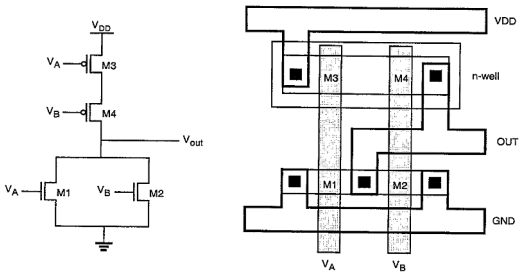

Fig. 4. CMOS 2-input NOR Gate with Layout.

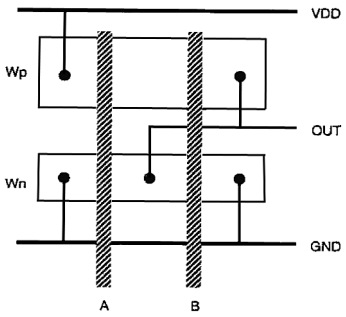

Fig. 5. Stick Diagram for CMOS 2-input NOR Gate.

The NMOS network corresponding to the logic function **f** is shown in Figure 6. The corresponding PMOS network derived from the NMOS network is given in Figure 7.

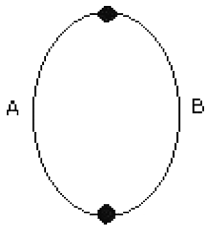

Fig. 6. NMOS Network corresponding to **f**.

Fig. 7. PMOS Network corresponding to NMOS in Figure 6, 2 Odd Vertices.

The NMOS network can be organized so that the derived PMOS network has minimum possible number of odd vertices. Minimization of the number of odd vertices ensures minimization of diffusion breaks [1] and hence corresponding to the optimal solution [2]. It is possible to have more than one optimum solution (i.e. of minimum diffusion break) for a CMOS network.

The rest of the article is organized as follows. In Section 2, it has been proved that a MOS network will always have an even number of odd vertices. Section 3 determines the number of diffusion breaks as a function of the number of odd vertices in a MOS network. Section 4 explains the need to organize the NMOS network, so that the derived PMOS network has minimum number of odd vertices. Section 5 determines the number of ways the odd vertices can be connected and accordingly develops the algorithm to calculate the number of total possible optimized solutions. In Section 6, experimental results are presented with some interesting points. Finally, Section 7 concludes the paper.

2. Number of Odd Vertices in a MOS Network

The proof that a MOS network will always have an even number of odd vertices has been performed in an inductive way as follows.

Suppose, we have an arbitrary MOS network. With this network, a single MOS can be merged into two ways – serially with one node (node_s) or parallely with two different nodes (node_p1, node_p2) in the network. A single MOS has two odd vertices. With this consideration, the Table 1 (for serial merging) and Table 2 (for parallel merging) can be formed.

	Before Merging	**After Merging**	**Increase in odd vertices**
Degree of node_s	odd	even	0
	even	odd	2

Table 1. Status of Vertices corresponding to Serial Merging.

	Before Merging	After Merging	Increase in odd vertices
Degree of node_p1, node_p2	odd, odd	even, even	-2
	odd, even	even, odd	0
	even, odd	odd, even	0
	even, even	odd, odd	2

Table 2. Status of Vertices corresponding to Parallel Merging.

So, increase or decrease in the number of odd vertices is always even after merging. It ensures that if the previous MOS network has even number of odd vertices, the new network after merging will also have an even number of odd vertices. Since the smallest possible network is a single MOS, which has 2 (that is even number) odd vertices, we can say, by induction, that any MOS network contains an even number of odd vertices.

3. Number of Diffusion Breaks in a MOS Network

To determine the number of diffusion breaks for a MOS network with 2k odd vertices, the theorem given in [3] is stated below.

Theorem. *In a connected graph G with exactly 2k odd vertices, there exist k edge-disjoint subgraphs such that they together contain all edges of G and that each is a unicursal graph.*

A unicursal graph contains a unicursal line or open Euler line. For an Euler line (closed Euler line), the start and end vertex are same. For a graph with 2 odd vertices, the open Euler path starts from an odd vertex and ends at another odd vertex.

With this theorem, it is obvious that a MOS network with O ($O \geq 2$) odd vertices, the number of diffusion breaks,

$$D = O/2 - 1 \qquad (1)$$

Layout area increases as the number of diffusion breaks increases. So, from equation (1), it is an obvious consequence that minimizing the number of odd vertices will eventually minimize the layout area of CMOS logic gates.

4. Minimization of Odd Vertices in PMOS Network

NMOS network controls the number of odd vertices in the corresponding PMOS network as it is derived from that. A NMOS network performing same logic function can be organized in different ways. We need to choose the configuration which minimizes the number of odd vertices in the corresponding PMOS network. One example to clarify the matter is as follows.

Let us consider the logic function **f**, where

$$\mathbf{f'} = ABC + DE + FG + HIJ$$

f' is the complement function of **f**.

This logic function can be implemented by any of the two NMOS configurations as in Figure 8 and Figure 9.

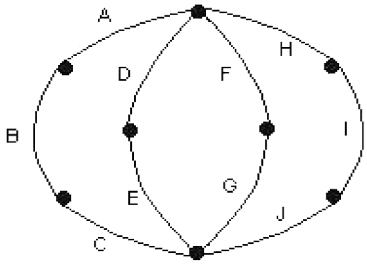

Fig. 8. NMOS Network corresponding to **f** (one way).

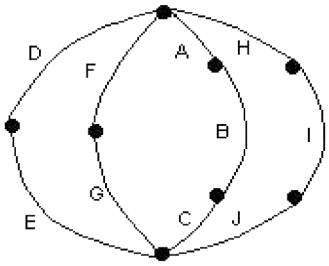

Fig. 9. NMOS Network corresponding to **f** (another way).

All the vertices in Figure 8 and Figure 9 are of even degree. The PMOS network configurations corresponding to the above NMOS networks are shown in Figure 10 and Figure 11 respectively.

So, the CMOS configuration corresponding to the NMOS network as in Figure 9 (odd vertices = 2) achieves less layout area than that as in Figure 8 (odd vertices = 4).

From Figure 8 and Figure 9, we can determine the underlying reason. If we count the number of NMOS sequentially from top node to bottom node for the two configurations, it is 3-2-2-3 for Figure 8 and 2-2-3-3 for Figure 9. So, Figure 8 has one more odd-to-even or even-to-odd transition than Figure 9. This is the reason that Figure 10 has 2 more odd vertices than Figure 11. So, the solution is to minimize this odd-to-even/even-to-odd transition in NMOS network. Then, the corresponding PMOS network will have minimum possible number of odd vertices.

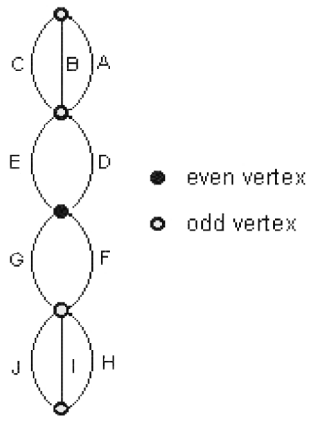

Fig. 10. PMOS Network corresponding to NMOS in Figure 8, 4 Odd Vertices.

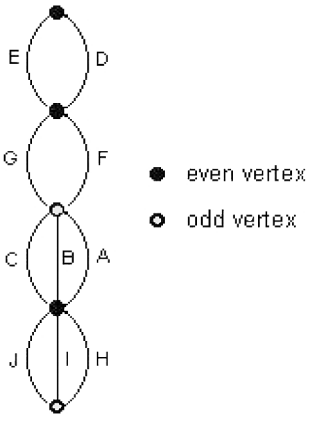

Fig. 11. PMOS Network corresponding to NMOS in Figure 9, 2 Odd Vertices.

5. Computation of Total Number of Optimized Solutions

One CMOS logic circuit may have more than one optimized solutions (i.e. gate ordering) on the basis of diffusion breaks. To get all the optimized solutions, it is necessary to calculate in how many ways the odd vertices can be connected to get an all-even-vertex network.

Let a MOS network have N odd vertices. From Section 2, it may be written

$$N = 2m, \text{ where } m = 0, 1, 2, \ldots$$

From the N number of odd vertices, a pair of odd vertices can be selected in ${}^{N}C_2$ ways. Another pair of odd vertices can be selected from the remaining $N-2$ odd vertices in ${}^{N-2}C_2$ ways. In this manner, we have $m(=N/2)$ pair of connections. The number of ways these m connections can be established (with duplicate connections allowed) is

$$\begin{aligned} f_{N,Dup} &= {}^{N}C_2 \times {}^{N-2}C_2 \times \ldots \times {}^{2}C_2 \\ &= \frac{N(N-1)}{2} \times \frac{(N-2)(N-3)}{2} \times \ldots \times \frac{2.1}{2} \\ &= \frac{N!}{2^{N/2}}. \end{aligned}$$

But, as stated, there will be duplicate connections with this formulation. To clarify this matter, let us assume that a MOS network has 6 odd vertices named a, b, c, d, e, and f. Now, the sequences – (a,b), (c,d), (e,f) and (a,b), (e,f), (c,d) are the same. There will be $m!$ duplicacy for a sequence. After removing this duplicacy, the number of ways the m connections can be established between the odd vertices is

$$f_N = \frac{N!}{(N/2)! \times 2^{N/2}} \qquad (2)$$

So,

$$\begin{aligned} f_{N+2} &= \frac{(N+2)!}{(N/2+1)! \times 2^{(N/2+1)}} \\ &= \frac{(N+2)(N+1)N!}{(N/2+1) \cdot (N/2)! \times 2.2^{(N/2)}} \\ &= (N+1) f_N. \end{aligned}$$

Hence,

$$(f_{N+2} - f_N)/f_N = N.$$

When $N = 2, f_N = 1$ i.e. there is only one way to connect the two odd vertices as expected. With this formulation, the algorithm to find out all the optimized solutions for a MOS network is given below.

Procedure allOptimizedSolutions (∧, totalNoOfSolutions)

// *Input*: ∧ is an $N \times N$ matrix where, N is the number of nodes in the corresponding MOS network. Each entry $\wedge_{i,j}$ denotes the number of connections from i^{th} node to j^{th} node. $\wedge_{i,j} = \wedge_{j,i}$ for all i, j where, $1 \leq i \leq N$ and $1 \leq j \leq N$. $\wedge_{i,i} = 0$ for all i.

Output: totalNoOfSolutions ← total number of solutions for the MOS network characterized by ∧. //

noOfOddVertices ← calculate the number of odd vertices.

If noOfOddVertices = 0 *then*

totalNoOfSolutions ← use exhaustive search to find out the number of solutions.
// If there are E edges, the number of search will be E! //

Else

noOfWaysConnectingOddVertices ← Calculate the number of ways the odd vertices can be connected using equation (2) [assume, N = noOfOddVertices].

totalNoOfSolutions = 0.

For no = 1 to noOfWaysConnectingOddVertices

newEdges ← the edges after connecting the corresponding odd vertices.

Merge the previous edges with the newEdges to get an all-even-vertex network.

noOfSolutions ← use exhaustive search to find out the number of solutions.

// In this case, we need to ensure that each solution contains any one edge from the set of newEdges as the last or first traversed edge. This consideration ensures that the Euler path starts from a previously odd vertex and ends at another previously odd vertex //

// If there are E edges, the number of search will be E! //

totalNoOfSolutions = totalNoOfSolutions + noOfSolutions.

End //End For

End //End If-Else

End // End Procedure

The algorithm to calculate the common Euler paths in NMOS and PMOS network is given below.

Procedure calculateCommonEulerPaths($\wedge_{NMOS}$, commonSolutions)

// *Input*: $\wedge_{NMOS}$ is an $N \times N$ matrix where, N is the number of nodes in the corresponding NMOS network. Entries are the same as previously stated for ∧ in procedure *allOptimizedSolutions*.

Output: commonSolutions ← common solutions for the CMOS network. //

totalNoOfSolutionsNMOS ← *allOptimizedSolutions* ($\wedge_{NMOS}$).

Determine $\wedge_{PMOS}$ from $\wedge_{NMOS}$ with the consideration as stated in Section 4, so that $\wedge_{PMOS}$ has minimum possible number of odd vertices.

totalNoOfSolutionsPMOS ← *allOptimizedSolutions*($\wedge_{PMOS}$).

commonSolutions ← *checkCommonSequences* (totalNoOfSolutionsNMOS, totalNoOfSolutionsPMOS).

// The function *checkCommonSequences* returns the common sequences of totalNoOfSolutionsNMOS and totalNoOfSolutionsPMOS //

End // End Procedure

6. Experimental Results

This section presents some interesting observations regarding common Euler paths in NMOS and PMOS networks for a CMOS circuit with the help of the algorithms presented in Section 5.

Observation 1: There may or may not exist some common solutions (Euler paths) without a diffusion break for a CMOS network, even if the individual NMOS and PMOS network has Euler paths.

Example 1.1: This example shows that there exist common solutions (Euler paths) for a CMOS network when corresponding NMOS and PMOS network both have no odd vertices.

The NMOS network in Figure 12 has 72 solutions and the PMOS network in Figure 13 has 24 solutions. There exist 8 common solutions for them. The common solution sequences

are A-B-C-F-E-D, A-D-E-F-C-B, B-A-D-E-F-C, D-A-B-C-F-E, E-F-C-B-A-D, F-E-D-A-B-C, F-C-B-A-D-E and C-F-E-D-A-B.

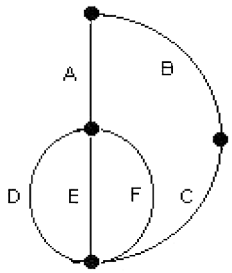

Fig. 12. NMOS Network corresponding to Observation 1, no Odd Vertices.

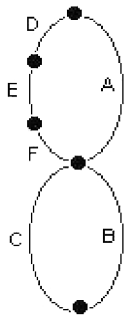

Fig. 13. PMOS Network corresponding to Observation 1, no Odd Vertices.

Example 1.2: This example shows that there is no common solution (Euler path) for a CMOS network when corresponding NMOS and PMOS network both have no odd vertices.

The NMOS network in Figure 14 has 256 solutions and the PMOS network in Figure 15 has 64 solutions. But, there is no common solution for them.

Observation 2: There may exist common solutions (Euler paths) without a diffusion break for a CMOS network, even if the individual NMOS and PMOS networks have 2 odd vertices each.

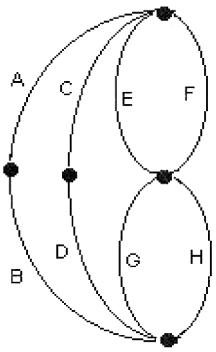

Fig. 14. NMOS Network corresponding to Observation 1, no Odd Vertices.

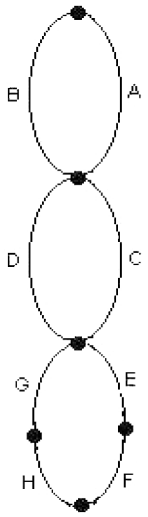

Fig. 15. PMOS Network corresponding to Observation 1, no Odd Vertices.

Example 2: This example shows that there are common solutions (Euler paths) for a CMOS network when corresponding NMOS and PMOS network both have 2 odd vertices.

The NMOS network in Figure 16 has 672 solutions and the PMOS network in Figure 17 has 8 solutions. There are 2 common solutions for them. The common solution sequences are A-B-E-F-H-G-D-C and E-F-H-G-D-C-B-A.

Observation 3: There may or may not exist common solutions (Euler paths) without a diffusion break for a CMOS network, even if individual NMOS and PMOS networks have in total 2 odd vertices.

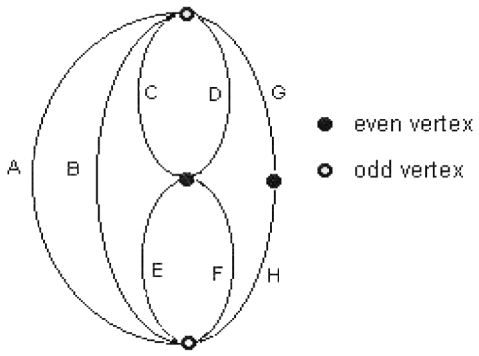

Fig. 16. NMOS Network corresponding to Observation 2, 2 Odd Vertices.

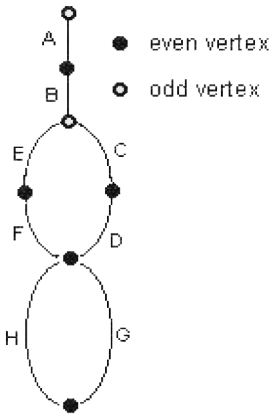

Fig. 17. PMOS Network corresponding to Observation 2, 2 Odd Vertices.

Example 3.1: This example shows that there are no common solutions (Euler paths) for a CMOS network when corresponding NMOS and PMOS networks have 0 and 2 odd vertices respectively.

The NMOS network in Figure 18 has 192 solutions and the PMOS network in Figure 19 has 4 solutions. But, there is no common solution for them.

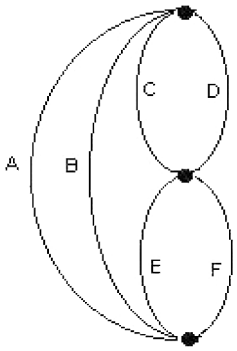

Fig. 18. NMOS Network corresponding to Observation 3, no Odd Vertices.

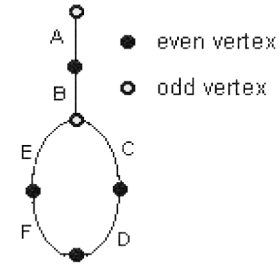

Fig. 19. PMOS Network corresponding to Observation 3, 2 Odd Vertices.

Example 3.2: This example shows that there is a common solution (Euler path) for a CMOS network when corresponding NMOS and PMOS networks have 2 and 0 odd vertices respectively.

The NMOS network in Figure 20 has 28 solutions and the PMOS network in Figure 21 has 24 solutions. There is only one common solution for them. The common solution sequence is F-E-B-A-C-D.

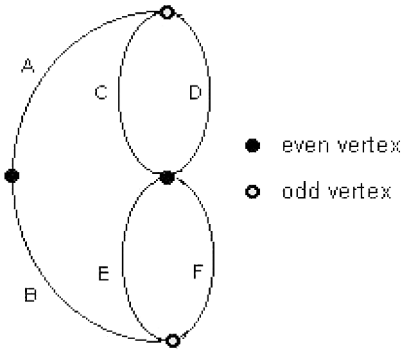

Fig. 20. NMOS Network corresponding to Observation 3, 2 Odd Vertices.

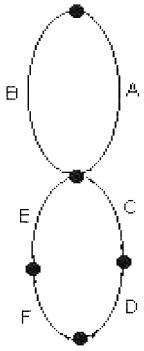

Fig. 21. PMOS Network corresponding to Observation 3, no Odd Vertices.

7. Conclusion

The paper presents some insights into the Euler path approach to find out the optimum gate order for CMOS logic gates. It explains the insights proposed and develops an algorithm to determine all the possible optimized solutions for a CMOS logic circuit. Also, some interesting observations have been presented exploiting the proposed algorithm.

References

[1] S. KANG AND Y. LEBLEBICI, CMOS Digital Integrated Circuits – Analysis and Design, *McGrawHill e2*, 1999, Chapter 7, pp. 281–287.

[2] T. UEHARA AND W. M. VANCLEEMPUT, Optimal layout of CMOS functional arrays, *IEEE Trans. Computer*. 1981 (May), pp. 305–312.

[3] NARSINGH DEO, Graph Theory with Applications to Engineering and Computer Science, *Prentice Hall India*, 2001, Chapter 1, pp. 23–26.

Received: August, 2005
Revised: May, 2006
Accepted: May, 2006

Contact address:
Kuntal Roy
Department of Electronics
and Tele-Communication Engineering
Jadavpur University
Kolkata – 700 032
India
e-mail: kuntal_ju@yahoo.com

KUNTAL ROY is a research assistant at the Electronics and Tele-Communication Engineering Department of Jadavpur University. His research interests are in VLSI, WDM optical networks, robotics, vision and image processing.