Microelectronics Journal 31 (2000) 991–998

Microelectronics Journal

www.elsevier.com/locate/mejo

# A general method in synthesis of pass-transistor circuits

D. Marković[a,*], B. Nikolić[b], V.G. Oklobdžija[c]

[a]Department of Electrical Engineering and Computer Sciences, 211-149 Cory Hall #1772, University of California, Berkeley, CA 94720-1772, USA
[b]Department of Electrical Engineering and Computer Sciences, 570 Cory Hall, University of California, Berkeley, CA 94720-1770, USA
[c]Integration Corporation, 1285 Grizzly Peak Blvd, Berkeley, CA 94708, USA

**Abstract**

A general method in synthesis and signal arrangement in different pass-transistor network topologies is analyzed. Several pass-transistor logic families have been introduced recently, but no systematic synthesis method is available that takes into account the impact of signal arrangement on circuit performance. In this paper we develop a Karnaugh map based method that can be used to efficiently synthesize pass-transistor logic circuits, which have balanced loads on true and complementary input signals. The method is applied to the generation of basic two-input and three-input logic gates in CPL, DPL and DVL. The method is general and can be extended to synthesize any pass-transistor network. © 2000 Elsevier Science Ltd. All rights reserved.

*Keywords*: CMOS; Digital integrated circuits; Pass-transistor logic; Logic synthesis; Karnaugh maps

## 1. Introduction

During the last decade, extensive considerations have been given to the use of pass-transistor logic networks in the implementation of digital systems [1], however, to the best of our knowledge, there is still no systematic pass-transistor circuit synthesis approach exists, that takes into account the impact of circuit technology and signal arrangements on circuit performance.

A literature survey [1] shows that there are two main approaches to synthesize pass-transistor circuits: (1) transistor-level synthesis based on decision diagrams [2,3], and (2) library-based synthesis [4]. Both approaches are suitable for development of CAD algorithms. The first approach has more flexibility in circuit optimization because it is transistor based, but the optimization of large logic networks is difficult. The difficulties are problems related to termination of signals and insertion of repeaters into long transistor chains, and problems with verification of circuit functionality and correctness of output logic levels. The second approach is based on building a small set of optimized basic library components, and synthesizing large networks from them.

The focus of this work is to develop a unified method for mapping logic functions into circuit realizations using different pass-transistor logic styles. The method is based on the Karnaugh map representation of a logic function, with optimization of the corresponding circuit realization. This method is convenient for library-based synthesis since it can easily generate optimized basic logic gates, which are the main building blocks in library-based designs. It is simple and intuitive, and can be extended to generate larger logic functions, as will be discussed in more detail in Section 4.

Section 2 gives an overview of three conventional pass-transistor logic families (Complementary pass-transistor logic (CPL), double pass-transistor logic (DPL), and dual value logic (DVL)). Section 3 describes a general method to synthesize basic two-input and three-input logic gates (AND/NAND, OR/NOR, and XOR/XNOR) in each of these families, and discusses balancing input loads of the gates. Section 4 introduces an example how the same general framework can be used to synthesize complementary CMOS circuits. Section 5 concludes the paper.

## 2. Pass-transistor logic families

There are two main pass-transistor circuit styles: those that use NMOS only pass-transistor circuits, like CPL [7], and those that use both NMOS and PMOS pass-transistors, DPL [5] and DVL [6].

### 2.1. Complementary pass-transistor logic

Complementary pass-transistor logic [7] consists of complementary inputs/outputs, a NMOS pass-transistor network, and CMOS output inverters. The circuit function

* Corresponding author. Tel.: +1-510-666-3124; fax: +1-510-883-0270.
*E-mail address:* dejan@eecs.berkeley.edu (D. Marković).


PII: S0026-2692(00)00088-4

is implemented as a tree consisting of pull-down and pull-up branches. Since the threshold voltage drop of NMOS transistor degrades the "high" level of pass-transistor output nodes, the output signals are restored by CMOS inverters. CPL has traditionally been applied to the arithmetic building blocks [7–9] and has been shown to result in high-speed operation due to its low input capacitance and reduced transistor count.

### 2.2. Double pass-transistor logic

To avoid problems of reduced noise margins in CPL, twin PMOS transistor branches are added to N-tree in DPL [5]. This addition results in increased input capacitances. However its symmetrical arrangement and double-transmission characteristics compensate for the speed degradation arising from increased loading. The full swing operation improves circuit performance at reduced supply voltage with limited threshold voltage scaling.

### 2.3. Dual value logic

The main drawback of DPL is its redundancy, i.e. it requires more transistors than actually needed for the realization of a function. To overcome the problem of redundancy, a new logic family, DVL [6,10], is derived from DPL. It preserves the full swing operation of DPL with reduced transistor count. As introduced in [6], DVL circuit can be derived from DPL circuits in three steps, consisting of:

- Elimination of redundant branches
- Signal rearrangement (resize)
- Selection of the faster halves.

## 3. General synthesis method

The logic gate design presents a systematic implementation of a logic function. When new pass-transistor families are introduced, [5,7], the emphasis is usually given on their suitability for block design, and less attention is paid to the tradeoffs in the design of basic logic gates. The method presented in this paper is based on Karnaugh map coverage and circuit transformations as an approach to logic gate design. The algorithm is used to synthesize basic logic gates (AND/NAND, OR/NOR, and XOR/XNOR) in three conventional pass-transistor techniques. The algorithm efficiency is enhanced by duality and complementarity principles. Using these principles, starting for example from

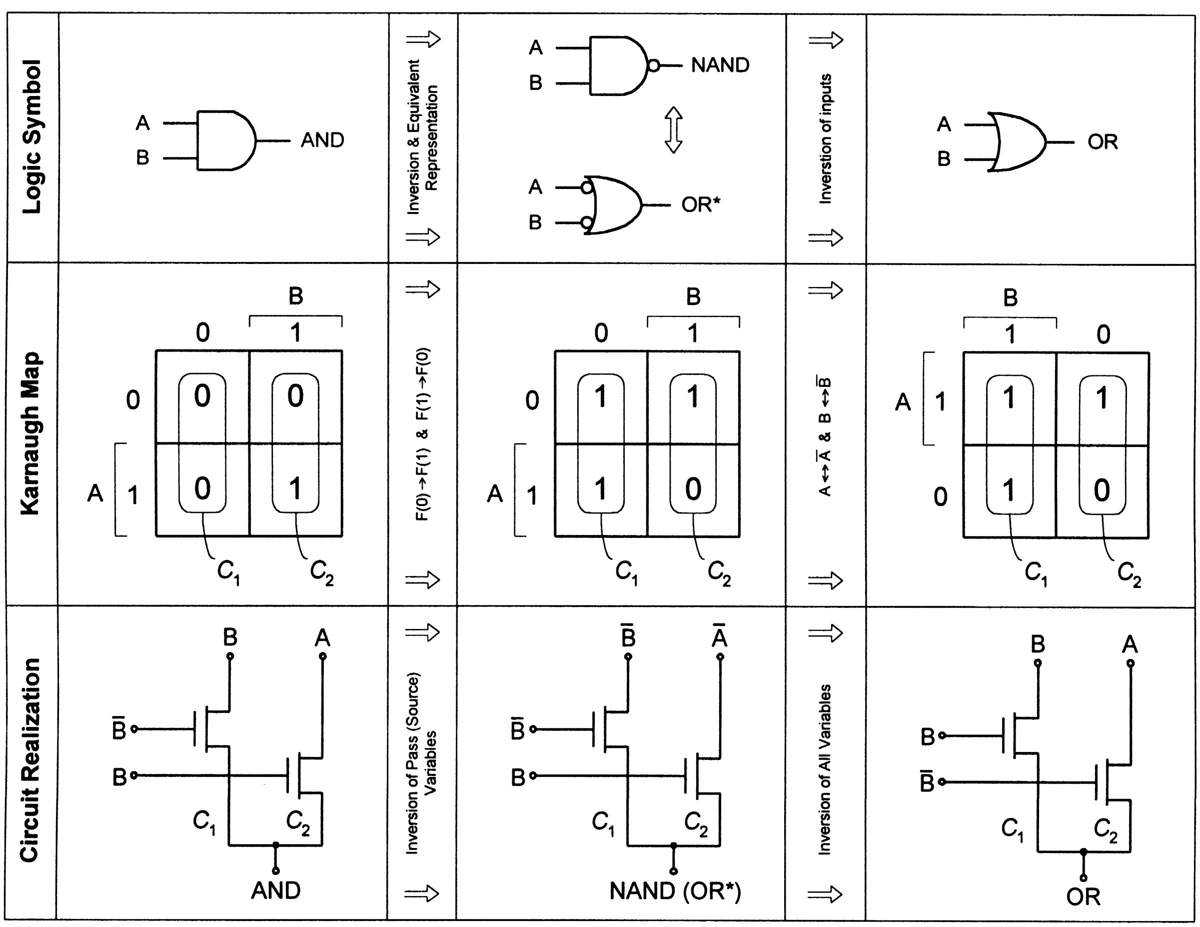


Fig. 1. Illustration of duality principle applied to synthesize CPL gates.

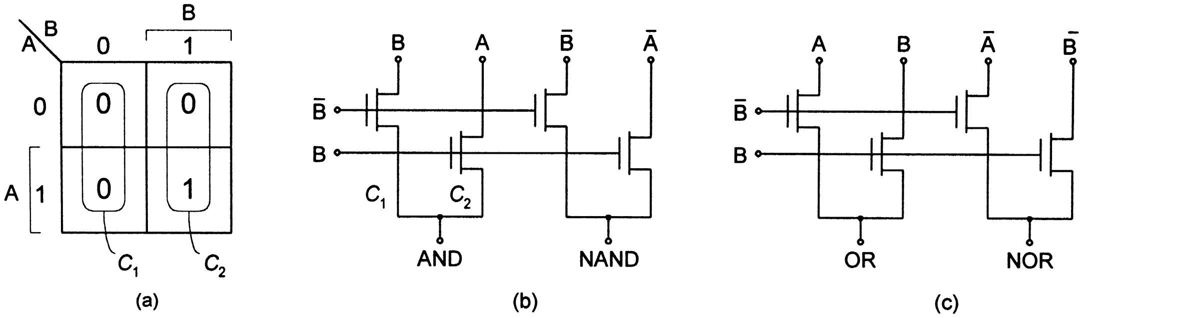


Fig. 2. Synthesis of two-input functions: (a) Karnaugh map of AND function; (b) circuit realization of AND/NAND function; and (c) circuit realization of OR/NOR function.

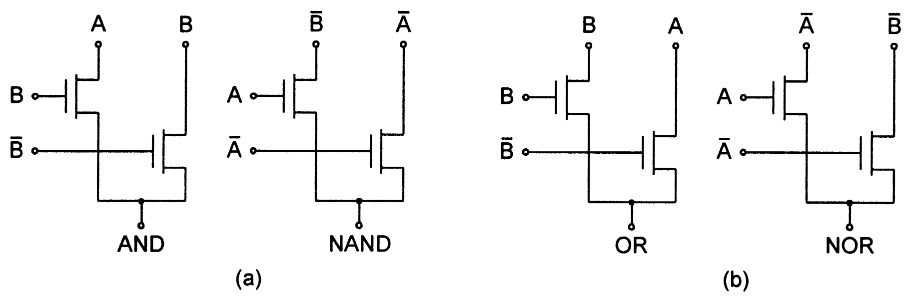


Fig. 3. Synthesis of two-input functions with balanced input load.

realization of AND circuit with simple transformations, one can obtain complementary and dual functions (NAND, OR/NOR) using the same circuit structure, improving the library versatility. Such framework applies to the design of basic logic gates in three conventional pass-transistor logic techniques, CPL, DPL, and DVL.

### *3.1. Complementary pass-transistor logic*

A general method of Karnaugh map coverage and mapping into circuit realizations is applied to design logic AND/NAND, OR/NOR, and XOR/XNOR gates in CPL. The method consists of the implementation of the gates from Karnaugh maps, and further simplifications: complementarity and duality principles that easily generate entire set of two-input and three-input logic gates.

The rules for Karnaugh map coverage and circuit realization in CPL are given below:

1. Cover Karnaugh-map with largest possible cubes (overlapping allowed).
2. Derive the value of a function in each cube in terms of input signals.
3. Assign one branch of transistor(s) to each of the cubes and connect all branches to one common node, which is the output of NMOS pass-transistor network.

The generation of complementary and dual functions is simple, by observing the basic properties of these gates. The

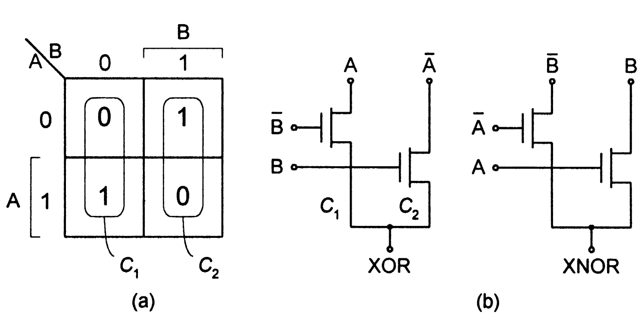


Fig. 4. Circuit realization of two-input XOR/XNOR function.

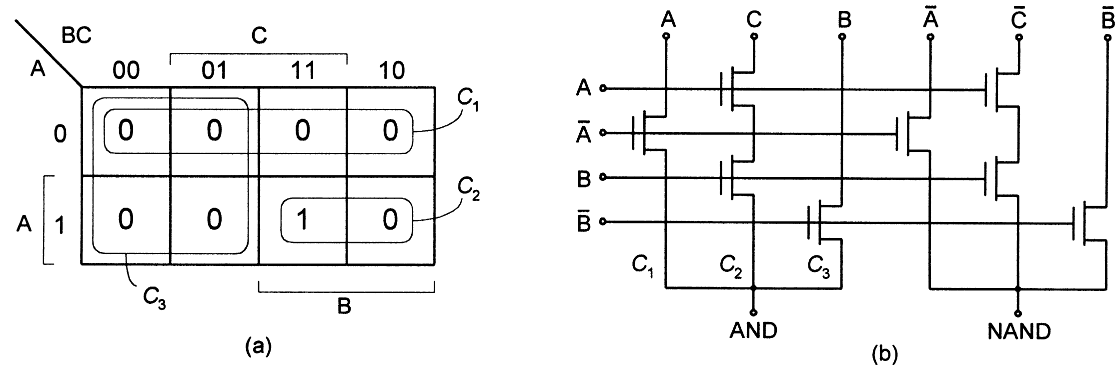


Fig. 5. Circuit realization of three-input AND/NAND function.

complementary logic function can be obtained from the same circuit structure by applying the complementarity principle.

*Complementarity principle:*

The same circuit topology, with inverted pass signals, produces the complementary logic function in CPL.

The complementarity principle holds in CPL since the pass variables are directly passed from the inputs to the outputs, so an inversion of the pass variables gives complementary function.

The dual logic function can be obtained by applying the duality principle.

*Duality principle:*

The same circuit topology, with inverted gate signals, gives the dual logic function. The dual logic functions are:

- AND–OR;
- NAND–NOR; and
- XOR, XNOR are self-dual (dual to itself).

The duality principle follows from DeMorgan's rules, and it is illustrated by the example of AND to OR transformation, Fig. 1.

The procedure of a logic gate design is shown using an example of two-input AND function, Fig. 2. Cubes $C_1$ and $C_2$ cover all input vectors in Fig. 2a. The value of the function covered by cube $C_1$ is equal to $B$, which becomes pass signal terminating to the source of the transistor branch. Branch $C_1$ is driven with $\bar{B}$ on its gate, which passes $B$ signal when $\bar{B}$ is high. Similarly for cube $C_2$, pass variable is the input signal $A$, driven by the gate signal $B$. These two cubes correspond to two NMOS transistor branches, implementing the desired function. By applying complementarity principle to the AND circuit, the NAND circuit is obtained, Fig. 2b. Furthermore, by applying duality principle on AND, two-input OR function is synthesized. The NOR circuit is then generated from OR (complementarity) or from NAND (duality), Fig. 2c.

### *3.1.1. Complementary pass-transistor logic gates with balanced input loads*

In AND/NAND circuit, Fig. 2b, loads on input signals $A$, $\bar{A}$, $B$, and $\bar{B}$ are not equal. However, two-input gates shown in Fig. 2 are commutative with respect to input signals, i.e. signals $A$ and $B$ could be swapped without affecting the circuit operation. With swapped inputs in the NAND circuit in Fig. 2a, resulting AND/NAND circuit has the same input loading, Fig. 3.

$$C_{\text{in}} = C_{\text{drain}} + C_{\text{gate}} \qquad (1)$$

However, the layout of circuits shown in Fig. 3 would require more area due to increased wiring complexity. Balanced loads also do not always result in balanced delay of true and complementary signals, because signal transitions do not traverse the same paths in true and complementary circuits. For example, when $B$ changes from the logic "high" to logic "low", and $A$ is "high", the output of the AND gate changes from "low" to "high", because *gate* signals ($B$, $\bar{B}$) are switching. On the other hand, the output of the NAND gate changes from "high" to "low", because *source* signal $\bar{B}$ is switching. These two different paths: gate-to-output, and source-to-output, result in difference in rising and falling delays of true and complementary output signals.

Circuit realization of two-input XOR/XNOR function with balanced input loads is shown in Fig. 4.

The XNOR function is obtained from XOR by applying the complementarity principle, and swapping input variables for balanced input load. Since the number of transistors is the same, and input signals are passed to the output in

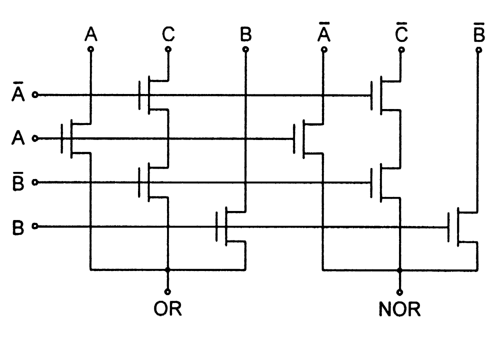


Fig. 6. Circuit realization of three-input OR/NOR function.

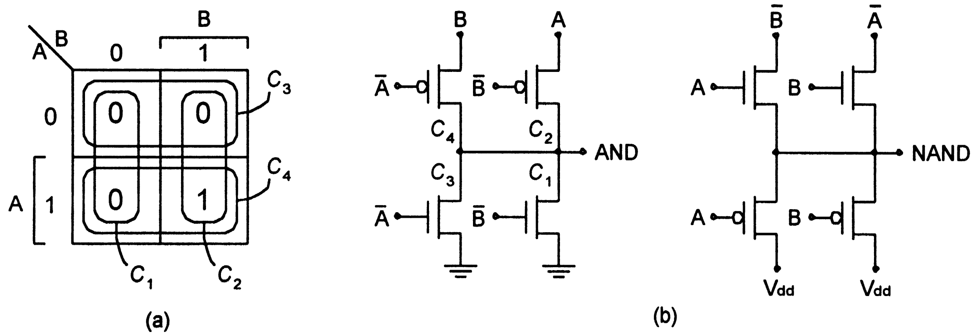


Fig. 7. Circuit realization of two-input AND/NAND function in DPL.

function implementation, the load presented to the input variables is the same as in circuits shown in Fig. 3, and is given by Eq. (1).

However, balancing loads is not always possible, because it depends on circuit realization and number of inputs. Fig. 5 shows a single-stage implementation of three-input AND/NAND function.

There are a total of 14 terminals, and 8 signals in Fig. 5, resulting in imbalanced inputs. If the functions were implemented as two-stage realizations composed of two-input modules with balanced loads, the loading would remain balanced.

Example in Fig. 5 also illustrates the reduction in transistor count by overlapping the cubes $C_1$ and $C_3$. It has a consequence that the corresponding branches are active for the input vectors where the cubes are overlapped, saving the circuit area.

Realization of three-input OR/NOR circuit, Fig. 6, is straightforward if complementarity and duality is applied to circuit shown in Fig. 5. A three-input XOR/XNOR circuit in CPL is typically composed of two-input XOR/XNOR modules [7].

### 3.2. Double pass-transistor logic

DPL has twice as many transistors as CPL for the same function. Design of DPL is based on covering every input vector in the Karnaugh map twice.

The rules to synthesize random logic function in DPL from the Karnaugh map are:

1. Two NMOS branches cannot be overlapped on logic "1"s. Similarly, two PMOS branches cannot be overlapped on logic "0"s.
2. Pass signals are expressed in terms of input signals or supply. Every input vector has to be covered with exactly two branches.

At any time, excluding transients, exactly two transistor branches are active, and any of the pairs NMOS–PMOS, NMOS–NMOS, and PMOS–PMOS are possible, depending on circuit implementation and input vectors.

*Complementarity principle:*

The complementary logic function in DPL is generated after the following modifications of the true function:

- Swap PMOS and NMOS transistors; and
- Invert all pass and gate signals.

To obtain a complementary function, it is necessary to invert both the pass signals and the gate signals since the PMOS and NMOS transistors are swapped.

*Duality principle:*

The dual logic function in DPL is generated when PMOS and NMOS transistors are swapped, and $V_{dd}$ and GND are swapped.

The design of a DPL two-input AND function is shown in Fig. 7. Cube $C_1$ of Fig. 7a is mapped to a NMOS transistor, with the source connected to ground and the gate connected to $\bar{B}$. Cube $C_2$ is mapped to a PMOS transistor, which passes $A$, when gate signal $\bar{B}$ is "low". The NMOS transistor of $C_3$ pulls down to ground, when $A$ is "low", and the PMOS transistor of $C_4$ passes $B$, when A is "high". Complementary circuit (NAND), Fig. 7b, is generated from AND, according to the complementarity principle.

Following the duality principle, OR circuit is formed from AND circuit, Fig. 8.

Different two-input XOR/XNOR circuit arrangements are possible. Fig. 9 shows a realization with balanced load on both true and complementary input signals.

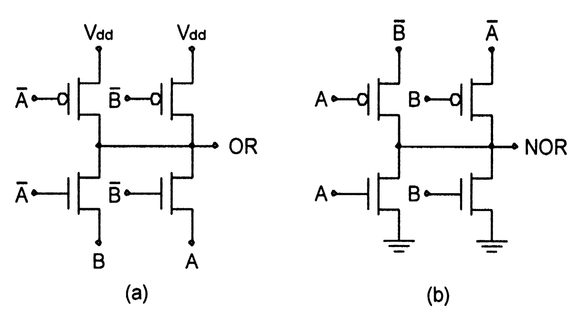


Fig. 8. Circuit realization of two-input OR/NOR function in DPL.

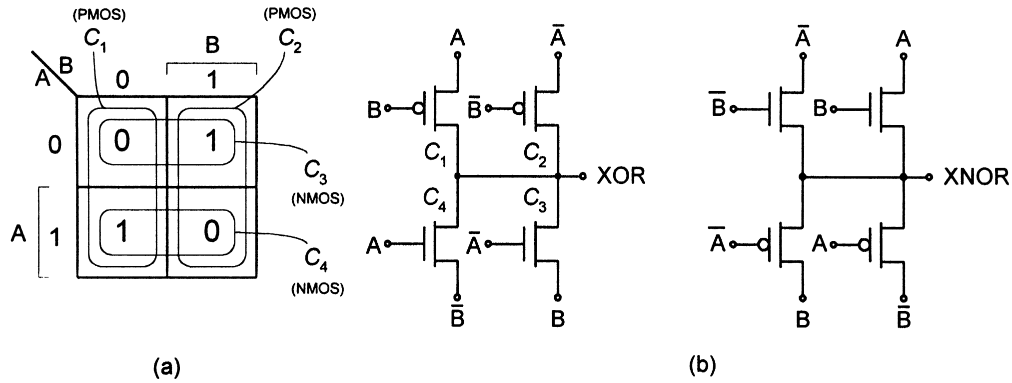


Fig. 9. Circuit realization of two-input XOR/XNOR function in DPL, with balanced input load.

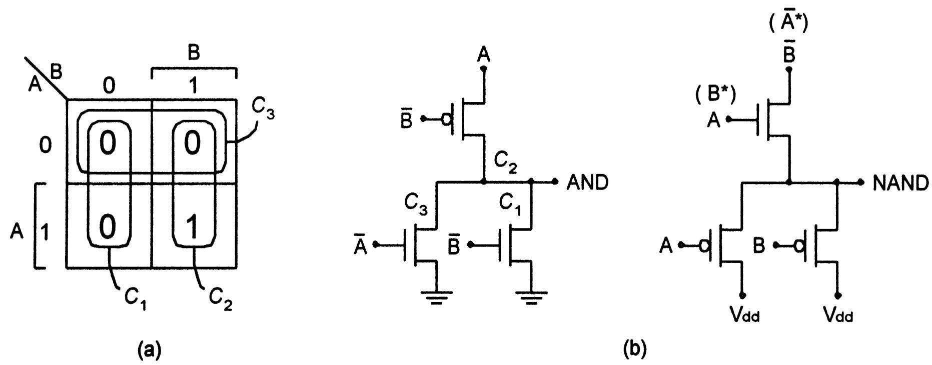


Fig. 10. Circuit realization of two-input AND/NAND function in DVL.

Three-input functions in DPL are implemented as cascaded combinations of two-input DPL modules.

### 3.3. Dual value logic

The rules to synthesize random logic function in DVL from the Karnaugh map are given below:

1. Cover with largest possible cubes all input vectors that produce "0" at the output, (overlapping allowed) and represent those cubes with NMOS devices, with sources connected to GND.
2. Repeat Step 1 for input vectors that produce "1" at the output and represent those cubes with PMOS devices, with sources connected to $V_{dd}$.
3. Finish with mapping input vectors, not mapped in Steps 1 and 2 (overlapping with cubes from Steps 1 and 2 allowed) that produce"0" or "1" at the output. Represent those cubes with parallel NMOS (good pull-down) and PMOS (good pull-up) branches, with sources connected to one of the input signals.

The complementarity and duality principles are identical as in DPL, since both DPL and DVL are CMOS structures.

Generation of two-input AND/NAND function is shown in Fig. 10.

Note that circuit realizations with balanced loads are not possible. Signals in brackets in Fig. 10b show another possible signal arrangement in the implementation of NAND function. Lower part of both circuits remains the same if input signals are swapped, only the upper branch changes. Optimal signal arrangement depends on circuit environment and switching probabilities of input signals.

Efficient realization of two-input OR/NOR circuits is shown in Fig. 11.

Realization of two-input XOR/XNOR circuit is identical to DPL, Fig. 9. Circuit implementation of three-input circuits in DVL is shown in Fig. 12.

Overlapping cubes $C_1$ and $C_2$, Fig. 12, saves area, which allows for wider transistors for cube $C_3$. The OR/NOR

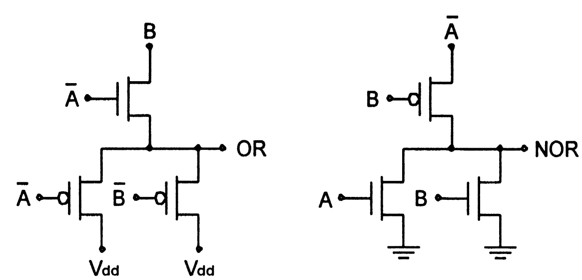


Fig. 11. Circuit realization of two-input OR/NOR circuit in DVL.

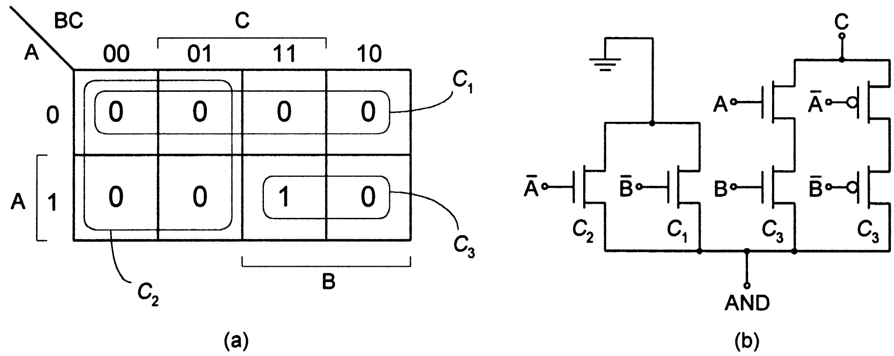


Fig. 12. Circuit realization of three-input AND function in DVL.

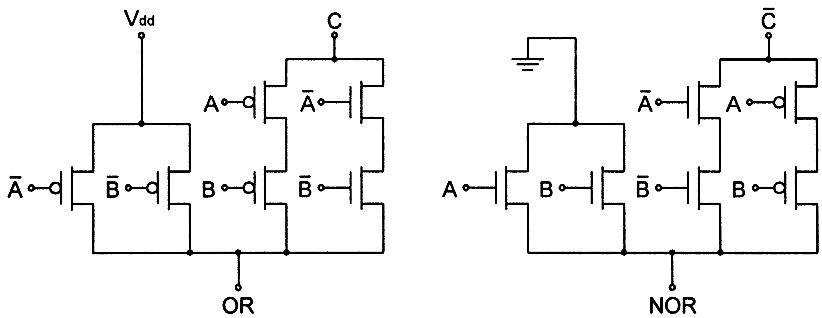


Fig. 13. Circuit realization of three-input OR/NOR functions in DVL.

circuit, directly generated from the AND circuit, is shown in Fig. 13.

## 4. Discussion

Synthesis of large pass-transistor networks is a challenging problem. The method presented in Section 3 can be extended to synthesize larger functions, as well as to synthesize complementary CMOS circuits. Complementary CMOS logic is a special case of pass-transistor logic, constrained that all source signals must terminate to $V_{dd}$ or *GND*. Complementary CMOS circuit synthesized from Karnaugh map can be compared to two different realizations in DVL technique that correspond to different mappings.

Consider the function

$$F = \bar{B}C + AB\bar{C} \quad (2)$$

and its three different realizations shown in Fig. 14. All three circuits have the same number of input signals, so the total input and output load determine the total switching capacitance and energy consumption. The complementary CMOS, Fig. 14d has small load on input signals and internal output load, compared to DVL realization in Fig. 14e. Two realizations of DVL show different mapping strategies: first strategy consists of covering the map with largest possible cubes, Fig. 14b, while the second strategy, Fig. 14c, is based on map decomposition and reduction to implementation of basic two-input functions. The DVL realization in Fig. 14f has smallest total load on input signals, and similar internal output load as complementary CMOS realization, as shown in Table 1.

This example illustrates the importance of efficient coverage of Karnaugh map, and circuit optimization. Straightforward coverage of Karnaugh map with largest cubes, as shown in Fig. 14b results in a circuit with lower performance, Fig. 14e, while more careful coverage with splitting of variables, Fig. 14c, results in a circuit with both smaller transistor stack and smaller transistor count, Fig. 14f.

Table 1
Comparison of different realizations of three-input function $F = B'C + ABC'$

| Realization | Number of input signals | Signal termination | Transistor Count | Output load |
|---|---|---|---|---|
| CMOS | 9 | 10G | 10 | 4S |
| DVL (e) | 9 | 8G + 6S | 8 | 6S |
| DVL (f) | 9 | 7G + 3S | 7 | 4S |

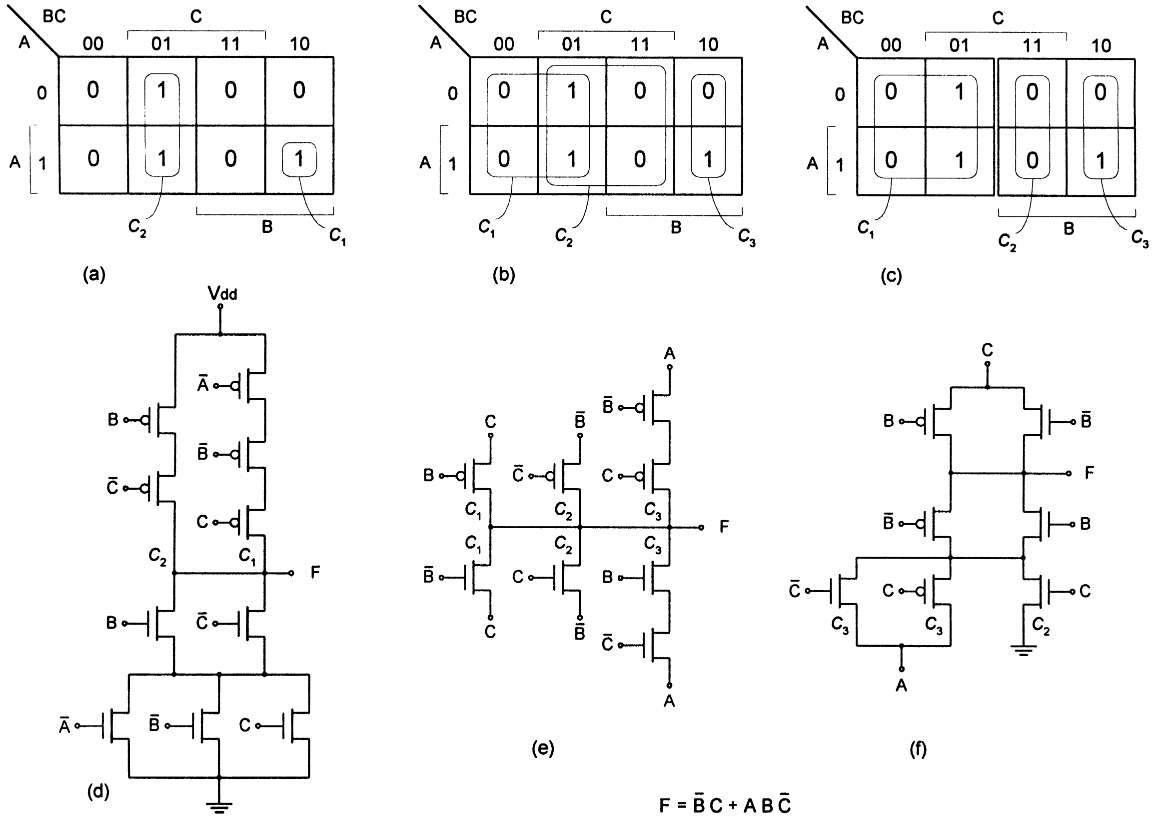


Fig. 14. Karnaugh map coverage of three-input function in: (a) complementary CMOS; (b) DVL; (c) DVL and corresponding circuit realizations in (d) complementary CMOS; (e) DVL; and (f) DVL.

## 5. Conclusions

General rules for synthesizing logic gates in three representative pass-transistor techniques were developed in this paper. An algorithmic way for generation of various circuit topologies (complementary and dual circuits) is presented. Generation of circuits with balanced input loads is suitable for library-based designs. The versatility of these circuits is increased by application of complementarity and commutative principles. This lays the foundation for development of computer aided design (CAD) tools capable of generating fast and power-efficient pass-transistor logic.

## References

[1] K. Taki, A survey for pass-transistor logic technologies — Recent researches and developments and future prospects, Proceedings of the ASP-DAC'98 Asian and South Pacific Design Automation Conference, Feb. 1998, pp. 223–226
[2] P. Buch, A. Narayan, A.R. Newton, A. Sangiovanni-Vincentelli, Logic synthesis for large pass transistor circuits, Proceeding of the IEEE International Conference on Computer-Aided Design (ICCAD), November 1997, pp. 633–670.
[3] A. Jaekel, S. Bandyopadhyay, G.A. Jullien, Design of dynamic pass-transistor logic using 123 decision diagrams, IEEE Trans. on CAS-I: Fundamental Theory and Applications 45 (11) (1998) 1172–1181.
[4] K. Yano, Y. Sasaki, K. Rikino, K. Seki, Top–down pass-transistor logic design, IEEE Journal of Solid-State Circuits 31 (6) (1996) 792–803.
[5] M. Suzuki, et al., 1.5 ns CMOS 16 × 16 multiplier using complementary pass-transistor logic, IEEE Journal of Solid-State Circuits 28 (11) (1993) 599–602.
[6] V.G. Oklobdžija, B. Duchene, Pass-Transistor Dual Value Logic for Low-Power CMOS, Proceedings of the 1995 International Symposium on VLSI Technology, Systems, and Applications, May–June, 1995, pp. 341–344.
[7] K. Yano, et al., 3.8 ns CMOS 16 × 16-b multiplier using complementary pass-transistor logic, IEEE Journal of Solid-State Circuits 25 (2) (1990) 388–395.
[8] I.S. Abu-Khater, A. Bellaouar, M.I. Elmasry, Circuit techniques for CMOS low-power high-performance multipliers, IEEE Journal of Solid-State Circuits 31 (10) (1996) 1535–1546.
[9] P.Y.K. Cheung et al., High speed arithmetic design using CPL and DPL logic, Proceedings of the 23rd European Solid-State Circuits Conference, September 1997, pp. 360–363.
[10] V.G. Oklobdžija, B. Duchene, Synthesis of high-speed pass-transistor logic, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 44 (11) (1997) 974–976.