01 Jan 1968

A Simple Design Algorithm For Synthesis Of Multilevel Combinational Networks

James H. Tracey
Missouri University of Science and Technology

Recommended Citation

J. H. Tracey, "A Simple Design Algorithm For Synthesis Of Multilevel Combinational Networks," *IEEE Transactions on Education*, vol. 11, no. 2, pp. 120 - 122, Institute of Electrical and Electronics Engineers, Jan 1968.
The definitive version is available at https://doi.org/10.1109/TE.1968.4320360

Short Papers and Classroom Notes

A Simple Design Algorithm for Synthesis of Multilevel Combinational Networks

JAMES H. TRACEY, MEMBER, IEEE

Abstract—**An algorithm is presented for the synthesis of combinational networks with a constrained arrangement of ANDs, ORs, NANDs, NORs, and inverters. The algorithm is easily presented to students in an introductory logic design course and has considerable application in modern practical logic design. Many texts available for a first course in logic design give very limited practical guidance for synthesis of networks under constraints outlined here. It is suggested that the algorithm presented here be furnished as supplementary material in the introductory course.**

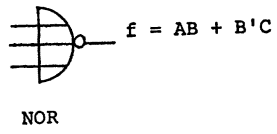
Introduction

Texts in introductory logic design and switching theory include a section on the synthesis of combinational networks with a set of gates such as ANDs, ORs, NOTs, NANDs, NORs, and perhaps some others. The synthesis technique usually begins with truth-table specifications which specify the output for combinations on the input. At this point one may write Boolean equation(s) in canonical form directly from the truth table and simplify these equations with simplification theorems. Alternately, the canonical equation(s), or coded representations, may be simplified with map techniques or tabular procedures. In all cases just mentioned, the result of simplification is a set of normal form equations (sum-of-products or product-of-sums). Such equations are readily implemented with AND–OR or OR–AND logic with unlimited fan-in. It is usually assumed that double-rail inputs are available or that inverters are available for complementing certain input variables. Texts frequently give very limited practical guidance for synthesis of networks composed of constrained arrangements of ANDs, ORs, NANDs, NORs, etc. For example, classroom experience has shown that, based on textbook knowledge, students have considerable difficulty synthesizing with AND–NOR modules (ANDs connected internally to a NOR) or synthesizing with only 3-input NOR gates. Difficult functional decomposition procedures are sometimes suggested as a means of solving the problem. In some instances it is suggested that the equation be manipulated algebraically until it is in implementation form. For example, to synthesize the function $f=AB+B'C$ with an AND–NOR module, one could algebraically manipulate the equation into the form $f=[A'B+B'C']'$ and synthesize directly. At least one text gives a set of design rules for eight different two-level modules [1]. An extremely simple procedure for synthesizing under the constraints given above is presented here. The procedure requires almost no memorization, is systematic, and is easy to explain to students.

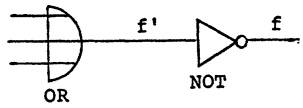
Design Procedure

The design technique for combinational circuits advocated here consists merely of designing from output back to input instead of the more conventional input-to-output design. Roth's decomposition algorithms take a similar approach and his work is presented in Miller [2]. However, the treatment is quite complicated because of its generality. Furthermore, such sophistication is frequently not warranted in practical design applications.

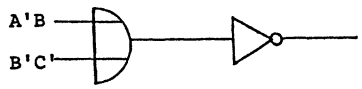
The procedure to be presented here can best be explained with an example. Suppose that the design problem is one of synthesizing the expression $f=AB+B'C$ with an AND–NOR module. It will be assumed that double-rail inputs are available and there is no fan-in limitation. The function to be synthesized must clearly be the output of a NOR gate as shown below.



The NOR gate can be thought of as an OR gate followed by a NOT.



It is obvious that the input to the inverter gate must be $f'=(A'+B')(B+C')$. Clearly, f' must be converted to simplified sum-of-products form in order to identify the OR gate inputs. The expression f' can be multiplied out and simplified to $f'=A'B+B'C'$. The inputs to the OR gate are as shown below.



Each of the OR inputs must be the output of an AND gate. Expressions representing OR gate inputs are already in the proper form for determining the AND gate inputs. The completed synthesis is as shown in Fig. 1.

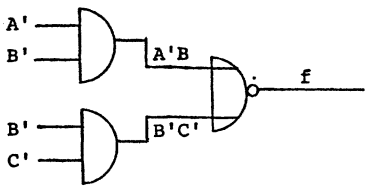
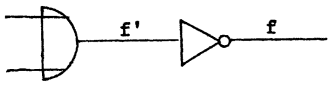


Fig. 1. Example 1.

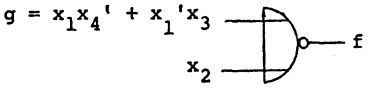
After performing the above mechanics, the student is in an excellent position to prove a short-cut algorithm for synthesis with AND–NOR modules. This algorithm states that to synthesize an arbitrary Boolean expression with AND–NOR modules, the complemented form of the expression is arranged in simplest sum-of-products form and the literals are inputs to the AND–NOR modules as though one were synthesizing this complemented expression with AND–OR modules. Any conventional simplification procedure may be used to obtain the simplest sum-of-products form for f. At this point, the student has the option of memorizing the design rules for AND–NOR, OR–NAND, NOR–NOR, and other two-level modules, or rederiving them on his own as the need presents itself.

The next example is an application of a similar technique to synthesis with gates having a limited number of inputs. Warfield [3] considers the synthesis of the function $f=x_1'x_2'x_3'+x_1x_2'x_4$ with 2-input NOR gates. The synthesis again proceeds from output to input as shown below.



Manuscript received January 9, 1968.
The author is with the Department of Electrical Engineering, University of Missouri at Rolla, Mo.

Note that f' is needed in simplest sum-of-products form in order to determine the input to the OR portion of the NOR gate. Through the use of map or algebraic techniques, one obtains $f'=x_2+x_1x_4'+x_1'x_3$. Since the OR gate will only accept two inputs, one may write the function as $f'=x_2+(x_1x_4'+x_1'x_3)$. In the general case, the writing of the sum function as the sum of two functions is accomplished by arbitrarily grouping the terms. The synthesis to this point is as shown below.



Next, one may attempt to synthesize the function g with two levels of NOR. Two levels of NOR are equivalent to OR–AND and therefore g should be converted to simplest product-of-sums form. The conversion gives $g=(x_1+x_3)(x_1'+x_4')$. It is recognized that g can be realized directly with only 2-input gates and the complete network is as shown in Fig. 2. Warfield's design shows 6 gates.

If one did not recall or did not wish to consider the direct realization of g with a two-level NOR network, he could continue to work backward, a gate at a time, until single-literal expressions appear on all gate inputs. This is illustrated in Fig. 3.

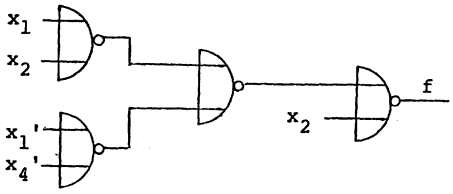


Fig. 2. Example 2.

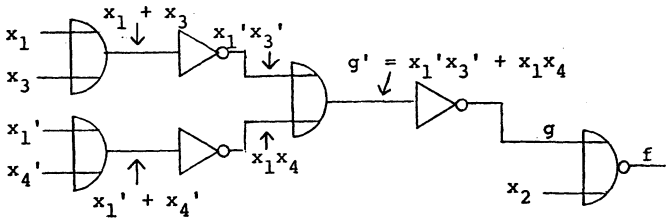
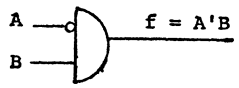


Fig. 3. Example 3 (rework of Example 2).

Another useful application of the technique presented here is the testing of a collection of gates for completeness. For example, suppose a student is to show that the 2-input gate



forms a complete set (if the gate is a complete set, any Boolean function may be realized through the interconnection of only the gates of this type). If the student has previously shown that a NAND gate forms a complete set, then all he needs to do is show that this gate is capable of performing the NAND operation. The procedure is illustrated in Fig. 4.

Limitations

Just as one might suspect, a procedure as informal and simple as the one described here is bound to have limitations. For example, no claim is made that the networks produced are minimal. However, absolute minimality is seldom a practical requirement as modern logic design. Some reasons for this limitation are as follows. In Example 2 (Fig. 2), there were a variety of ways in which the terms of f' could

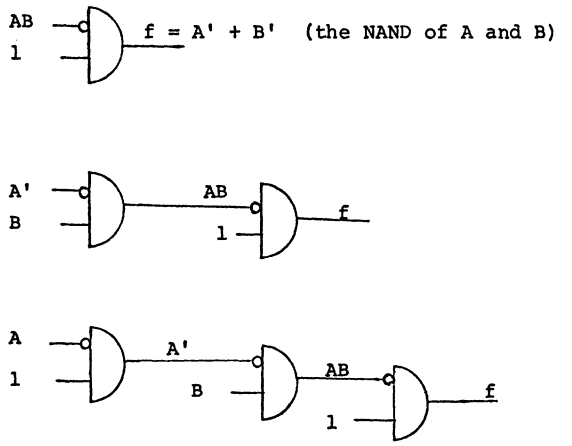
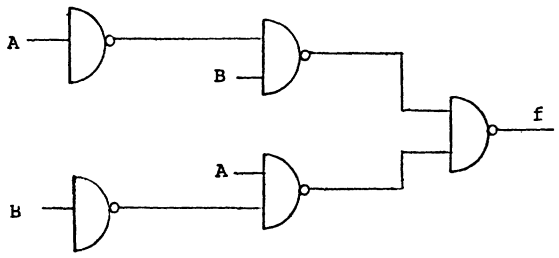
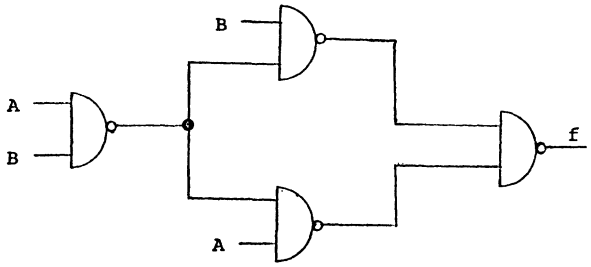


Fig. 4. Example 4.

have been grouped in determining the inputs to the output gate. Each grouping would produce a different network and it is to be expected that some might be simple than others. A second factor is that no serious attention is given to the possibility that one may simplify some networks by introducing appropriate redundancy in the equation to be implemented in order that branching and gate sharing may be exploited. For example, consider the function $f=AB'+A'B$. Implementation of this function with NAND gates by means presented here produces the following network (single-rail



inputs assumed). However, more sophisticated synthesis algorithms [4] would produce the following design.



One could have arrived at this last design by means of our procedure if he were clever enough to write the original equation in the form $f=A(A'+B')+B(A'+B')$.

Summary

A simple technique has been presented for realizing Boolean functions with a constrained collection of AND, OR, NOT, NAND, and NOR gates. The procedure can easily be applied to the design of single-rail or double-rail input networks, single- or multiple-output networks, and networks with or without gate fan-in constraints. The primary advantage of the technique is its simplicity; minimal networks are not guaranteed. Since many introductory texts in logic design give little practical guidance for synthesis of other than AND–OR and OR–AND networks, it is felt that the algorithm presented here ought to be furnished as supplementary material. No more than one class period is required for its presentation. More rigorous treatments of

synthesis techniques under the constraints considered here may be found in the literature [2], [5], [6].

References

[1] T. C. Bartee, I. L. Leblow, and I. S. Reed, *Theory and Design of Digital Machines*. New York: McGraw-Hill, 1962.
[2] R. E. Miller, *Switching Theory*, vol. 1. New York: Wiley, 1965.
[3] J. N. Warfield, *Principles of Logic Design*. Boston, Mass.: Ginn, 1963, p. 136.
[4] G. A. Maley and J. Earle, *The Logic Design of Transistor Digital Computers*. Englewood Cliffs, N.J.: Prentice-Hall, 1963.
[5] R. L. Ashenhurst, "The decomposition of switching functions," *Proc. Internat'l Symp. on the Theory of Switching*, 1959.
[6] H. A. Curtis, *A New Approach to the Design of Switching Circuits*. Princeton, N.J.: Van Nostrand, 1962.

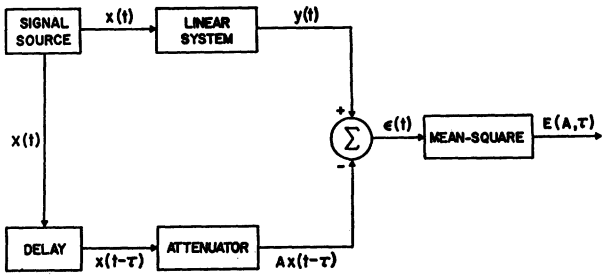


Fig. 1. Technique for evaluating waveform distortion introduced by a linear system.

Analysis of Waveform Distortion in Linear Systems

RONALD C. HOUTS, MEMBER, IEEE, AND
RICHARD S. SIMPSON, MEMBER, IEEE

***Abstract*—Analysis of waveform distortion in linear systems has been taught in several ways. The approach introduced here is based upon the mean-square difference between the input and output signals. Both deterministic and random signals can be accommodated, the only requirements for a mathematical analysis being knowledge of the input-signal autocorrelation function and the unit-impulse response of the linear system. The corresponding experimental procedure is based on comparison of the output signal with a replica of the input signal which has been delayed and attenuated in order to minimize the mean-square difference. Examples are included which analyze the effect of transmitting either a sinusoidal signal or a band-limited random signal through an RC low-pass filter. Agreement between theory and experiment is good.**

Causes of Waveform Distortion

A linear system may be described either in terms of its unit-impulse response $h(t)$ or its system function $H(\omega)$. The system function is defined as

$$H(\omega) = A(\omega)e^{+j\theta(\omega)} \tag{1}$$

where $A(\omega)$ is the amplitude response and $\theta(\omega)$ is the phase response. The principal sources of waveform distortion are the nonlinear phase and variable amplitude responses of the system function with respect to frequency. As a result of this distortion, each component in the spectrum is attenuated in amplitude and shifted in phase, or equivalently delayed in time, by different amounts. Consequently, the superposition of the individual frequency components at the output yields a waveform which differs from the input.

A linear system with a system function of the form

$$H(\omega) = Ke^{-j\omega t_0} \tag{2}$$

introduces into the output signal an attenuation K and a time delay t_0 which are independent of frequency. The mean-square difference between the input and output signals would be zero, provided the input signal was attenuated and delayed by the same amount. Therefore, the linear system represented by (2) does not introduce waveform distortion since its effect, except for time delay, can be eliminated simply by employing an amplifier with a frequency-independent gain of $1/K$.

Another source of waveform distortion is internal system noise. It will be shown in the following discussion that if the noise is independent of the input signal, an additive term will be contributed to the mean-square difference between the input and output signals.

Schemes for Evaluating Waveform Distortion

Several methods have been employed in textbooks for judging the performance of a linear system. One of the most common techniques uses a square-wave input signal [1]. The high-frequency response characteristic of the system is determined by observing the rise-time of the system response and the low-frequency characteristic is determined by observing the peak portions of the response. Although such an analysis of the time-domain response is useful in designing an amplifier with broad bandwidth, it does not indicate to what degree a given system distorts a typical information signal.

A second, more qualitative approach is to examine the amplitude and phase response of the system function [2]. Although a great deal of insight can be gained by such a frequency-domain analysis, a determination of the degree of waveform distortion introduced into a typical information signal is quite tedious. The effort involved in representing a signal by a Fourier series, delaying and attenuating each component according to the system function, and forming a new series which can be expressed in closed form is prohibitive.

An approach which overcomes the difficulties associated with the previous methods is a direct determination of the difference between the input and output signals [3]. This is accomplished by delaying and attenuating the input signal to obtain the minimum value of the mean-square difference. The concept for measuring the distortion is shown in Fig. 1. The mathematical analysis can be performed using either the input signal or its autocorrelation function combined with the unit-impulse response of the system. The autocorrelation function of the input signal is always used if the signal is random. Thus, the distortion introduced by the linear system can be analyzed for a wide variety of signals.

Theoretical Determination of Waveform Distortion

The criterion for measuring the waveform distortion D is the mean-square error $E(A, \tau)$ between input $x(t)$ and output $y(t)$, which can be expressed as

$$E(A,\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T}[y(t) - Ax(t-\tau)]^2 dt, \tag{3}$$

where A is the attenuation and τ the time delay as shown in Fig. 1. The waveform distortion D is defined as the minimum mean-square error which is obtained by proper selection of attenuation A_0 and time delay τ_0, i.e.,

$$D = E(A_0, \tau_0). \tag{4}$$

Since the input and output are not always expressed in a deterministic form, it is necessary to redefine (3) in terms of correlation functions. Squaring the integrand in (3) yields

$$E(A,\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T}[y^2(t) + A^2x^2(t-\tau) - 2Ax(t-\tau)y(t)]dt. \tag{5}$$

The first two terms of the integrand can be identified as the mean-square value of the output and the attenuated input, respectively. Alternatively, they can be represented by the input autocorrelation function $\phi_{xx}(\tau)$ and the output autocorrelation function $\phi_{yy}(\tau)$, both evaluated at $\tau = 0$, where the autocorrelation function $\phi_{ff}(\tau)$ of a random signal is defined as

$$\phi_{ff}(\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} f(t)f(t+\tau)dt. \tag{6}$$

The last term in the integrand of (5) can be identified as an input–

Manuscript received November 22, 1967.
The authors are with the Department of Electrical Engineering, University of Alabama, University, Ala. 35486