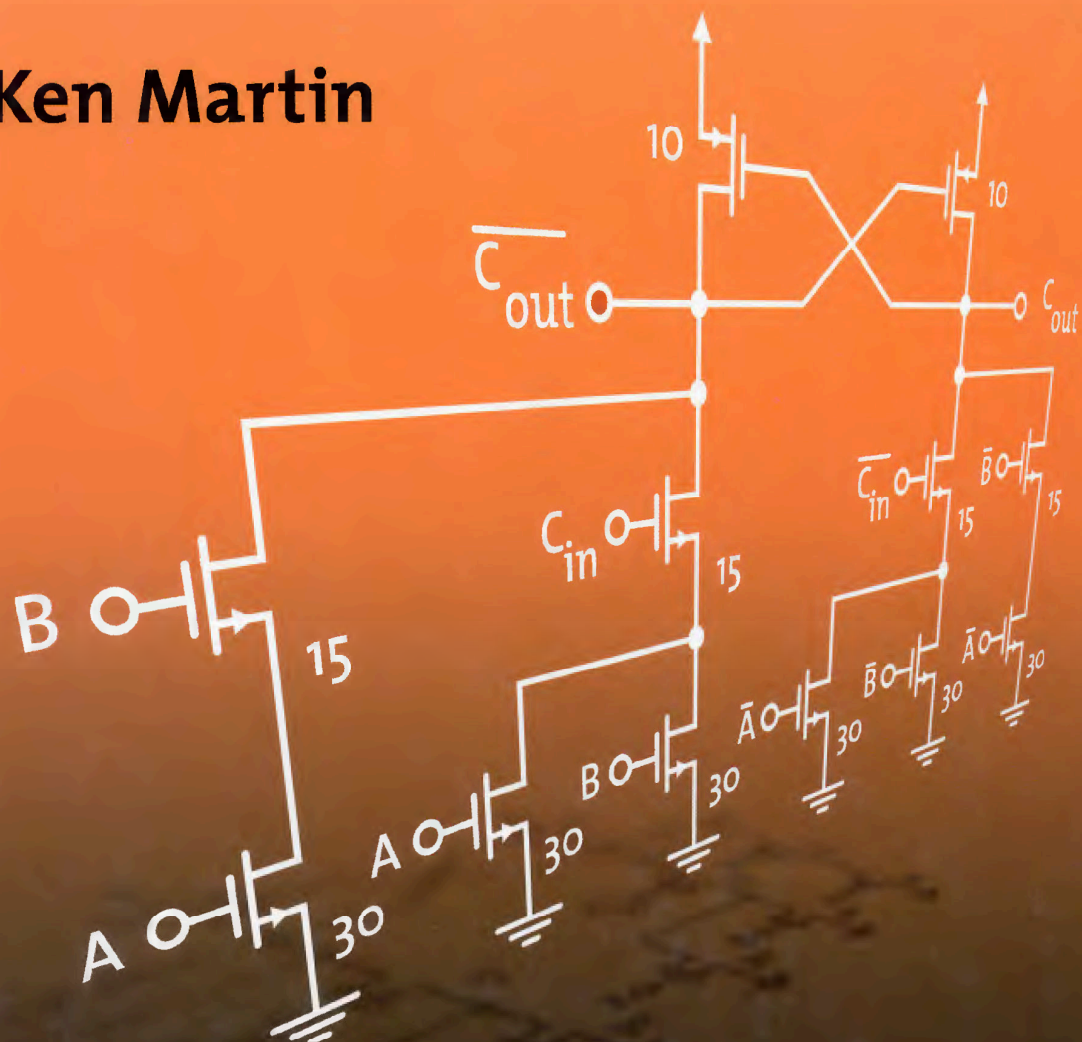


Digital Integrated Circuit Design

Ken Martin



Oxford University Press

Oxford New York
Athens Auckland Bangkok Bogotá Buenos Aires Calcutta
Cape Town Chennai Dar es Salaam Delhi Florence Hong Kong Istanbul
Karachi Kuala Lumpur Madrid Melbourne Mexico City Mumbai
Nairobi Paris São Paulo Singapore Taipei Tokyo Toronto Warsaw

and associated companies in
Berlin Ibadan

Copyright © 2000 by Oxford University Press, Inc.

Published by Oxford University Press, Inc.,
198 Madison Avenue, New York, New York 10016
<http://www.oup-usa.org>

Oxford is a registered trademark of Oxford University Press

All rights reserved. No part of this publication may be reproduced,
stored in a retrieval system, or transmitted, in any form or by any means,
electronic, mechanical, photocopying, recording, or otherwise,
without the prior permission of Oxford University Press.

Library of Congress Cataloging-in-Publication Data

Martin, Kenneth W. (Kenneth William), 1952-
Digital integrated circuit design / Ken Martin.
p. cm. — (Oxford series in electrical and computer
engineering)
Includes bibliographical references and index.
ISBN 0-19-512584-3 (cloth)
1. Digital integrated circuits—Design and construction.
I. Title. II. Series.
TK7874.65.M37 1999 98-53666
621.3815--dc21 CIP

Printing (last digit): 9 8 7 6 5 4 3 2 1

Printed in the United States of America
on acid-free paper

CHAPTER 1

The Basics

Having an understanding of digital integrated circuits (at least to a moderate degree) is becoming almost essential for an engineer doing system design. This is true because practically every modern system contains digital integrated circuits as critical subcomponents. As the capability to integrate a greater number of transistors in a single die or integrated circuit (IC) grows, it is becoming more common that an *application-specific IC (ASIC)* is required, at least for high-volume applications. The chances of being able to design competitive products using only *off-the-shelf* ICs is continually diminishing. Very often, the systems designer must design the ICs, as well as the systems. Thus, this text is intended not only for electrical engineering students who will be designing digital ICs, but also for other engineers who may eventually need to design digital ICs or work cooperatively with a digital IC designer.

In the early 1980s it was commonly thought that future VLSI chips would be designed using high-level computer tools or silicon compilers. This has seldom proven to be the case. Although, *computer-aided design (CAD)* tools are now indispensable when designing digital ICs, almost always, critical parts of an IC often must be designed at a transistor level to save on space and/or to improve speed. This is particularly true for commodity ICs that will be produced in large volume where an imaginative design, based on a thorough understanding of the IC technology and transistor models, can result in a 50–100% increase in speed over what would result if only standard cells based on traditional approaches and CAD-based logic synthesis were used.

Even for digital IC designers primarily involved in using standard libraries and automatic logic synthesis, a good understanding of logic design at the transistor level is very useful in designing new cells and understanding the capabilities of existing cells.

This text is primarily directed toward teaching the design of digital ICs at the transistor level. It covers the design of logic gates in depth for most of the modern logic families. Silicon metal-oxide semiconductor (MOS), silicon bipolar, and gallium arsenide (GaAs) technologies are covered, although MOS is emphasized. The text starts with the basics and slowly progresses to more complicated and advanced circuits such as buffers, flip-flops and registers, arithmetic circuits, programmable logic devices, semiconductor memories, and testable logic. Whenever possible, the most important underlying principles are emphasized, rather than giving an

exhaustive survey of current circuits and practices. For this reason, some topics such as *NMOS* logic gate design is covered because it is representative of a generic design style that has been and may become important in future technologies, even though this logic family is currently losing favor.

It is assumed the reader will have had prior experience with state-machine-based logic design and with transistors, although Chapter 3 does contain a review of relevant transistor modeling and theory. The bulk of the present chapter is used to introduce the reader to the design of simple MOS logic gates where the transistors are modeled simply as switches. In addition, some definitions are given for noise margins, and it is shown how the transient response of first-order RC networks can be used to approximate the delay of *MOS* inverters. Also, computer modeling based on SPICE is introduced.

1.1 Simple NMOS Logic Gates

Logic gates are used to take one or more inputs, apply a boolean function to them, and then produce a single boolean output.¹ The majority of digital ICs are based on binary logic where the signals represent a “0” or a “1” (as opposed to a multiple-valued logic). In binary logic, an input is considered to be a “0” if its voltage is below the gate threshold voltage, which will be denoted V_{TH} , whereas an input is a “1” if its voltage is above the gate threshold voltage. The simplest logic gate is an *inverter*, which has an output that is the complement of its input. The symbol for an inverter, is shown in Fig. 1.1a. A very simplistic transistor-level realization of an inverter, using an n-channel MOS transistor and a resistor, is shown in Fig. 1.1b. This inverter is similar to a common-source amplifier. If it were to be used as an amplifier, the typical input signals would have very small changes only, and the transistor would typically be operated in its active or saturation region. When used as an inverter, the input signal has large voltage changes, often equal to the power-supply voltage, and the transistor is usually either cut off,

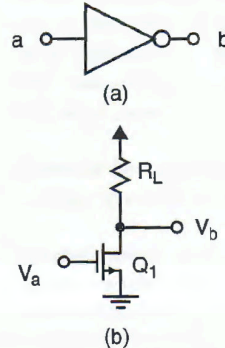


Figure 1.1 (a) The symbol for an inverter and (b) a simple realization using a resistor and an n-channel MOS transistor.

¹Some special logic gates, such as decoders, have multiple outputs.

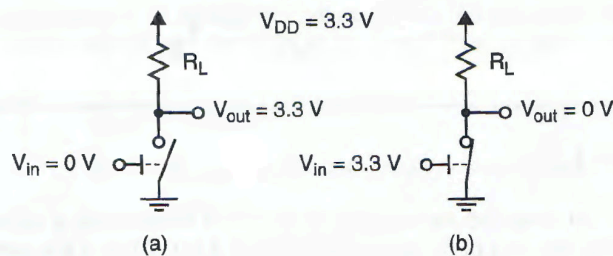
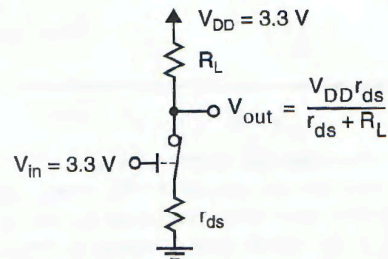


Figure 1.2 A switch-level interpretation of the inverter of Fig. 1.1 with (a) a “0” input and (b) a “1” input.

Figure 1.3 A more accurate modeling of the inverter of Fig. 1.1b.



and has a large voltage between its drain and source, or operates in the triode region, where its drain-source voltage is very small.

In a first-order analysis of the inverter of Fig. 1.1b, the *n*-channel transistor can be considered to operate as a voltage-controlled switch; if its gate voltage is large, the switch is “on” and connects the drain to the source; if its gate voltage is low, the transistor is equivalent to an open switch and can be ignored when analyzing the circuit. Thus, referring to Fig. 1.2a, we see that for a 0-V input signal to the drive transistor, the equivalent switch is open; therefore there is no current through the load resistor, and thus no voltage across it, and the output voltage will be equal to V_{DD} or 3.3 V.² Similarly, for a 3.3-V input signal, the equivalent switch is closed, which pulls the output voltage down to 0 V.

A more accurate model would be to replace the transistor by a small resistor when its gate voltage is high (or a “1”) as is shown in Fig. 1.3. The output voltage for a “1” input would then not be 0 V, but would be given by (using simple circuit analysis)

$$V_{out} = \frac{V_{DD} r_{ds}}{r_{ds} + R_L} \quad (1.1)$$

²In this text, V_{DD} , the power-supply voltage, is assumed to be 3.3 V unless specifically stated otherwise.

which is slightly larger than 0 V. As long as the load resistor, R_L , is much larger than r_{ds} , then the output voltage, although not 0 V, will still be much less than the gate threshold voltage.³

Example 1.1

Assume that when the transistor is on, it can be modeled by a 300- Ω resistor and that the size of the load resistor R_L is 4 k Ω . What is the output voltage when the input voltage is high.

Solution: Using (1.1), we have

$$V_{out} = \frac{3.3(300)}{300 + 4000} = 0.23 \text{ V} \quad (1.2)$$

Normally, the *inverter* of Fig. 1.1b would never be used in an actual integrated circuit, primarily because the resistor, R_L , would take up too much area. In an *NMOS* technology, a type of integrated circuit processing technology that was formerly popular, an n-channel depletion transistor can be used to replace the resistor R_L as is shown in Fig. 1.4.⁴ This type of transistor conducts when its gate is at the same voltage as its source, unlike an n-channel enhancement transistor.

This inverter operates very similarly to the inverter of Fig. 1.1b. Thus, the depletion transistor Q_2 operates somewhat like a resistor, but occupies much less chip area. This approximation will be made more exact in Chapter 4, but it is adequate for a first-order understanding. To obtain a low enough output voltage when the input is a "1", the width-to-length-ratio of the drive transistor Q_1 must be significantly greater than the W/L ratio of the load transistor. Normally, a factor of four is adequate and is typically used, as is the case for the inverter of Fig. 1.4.

It might also be emphasized at this time that for MOS transistors the ratio of the transistors width, W , to its length, L , is the most critical design parameter, as opposed to the absolute dimensions. Also, a transistor's W/L will sometimes be called its size and denoted as S .

A more complicated gate can be realized by adding more drive transistors (i.e., transistors that have their gates connected to an input signal). For example, the symbol for a *nor* gate and its realization using an NMOS technology are shown in Fig. 1.5a and b, respectively. This type of gate has a low output if any of the inputs are high. For the transistor realization of Fig. 1.5b, if any of the inputs is at a high voltage, the corresponding transistor can be approximated by a

³The gate threshold voltage V_{TH} is defined as the gate input voltage that will give an equal output voltage. This will be discussed in more detail in Chapter 4.

⁴The symbol of an n-channel *depletion* transistor is similar to that of an n-channel *enhancement* transistor, except there is an additional line between the drain and the source, which denotes that a conductive channel is present when the gate-source voltage is 0 V.

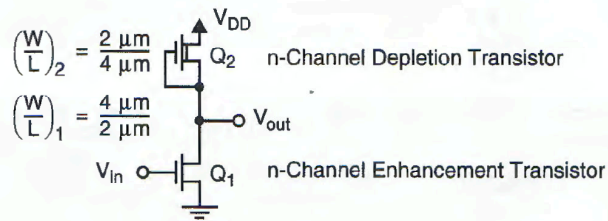


Figure 1.4 A typical inverter in an NMOS technology.

Figure 1.5 (a) The symbol for a *nor* gate and (b) its circuit-level realization using an NMOS technology.

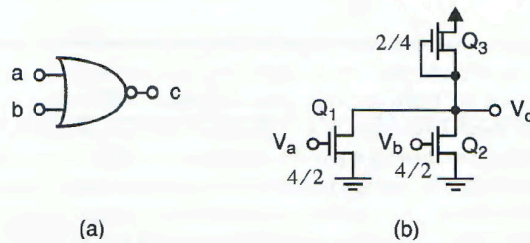
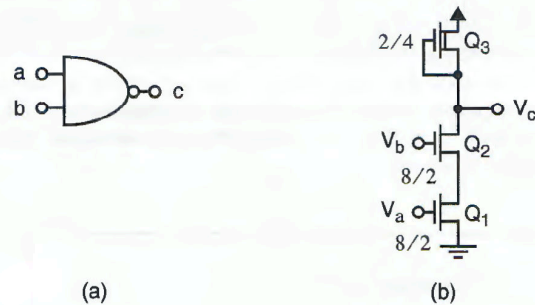


Figure 1.6 (a) The symbol for a *nand* gate and (b) its circuit-level realization using an NMOS technology.



closed switch, and the output voltage will be pulled low, assuming the W/L s (equivalently S s) of the drive transistors are large enough compared to the W/L s of the load transistor. A ratio of four is typically used, similar to what is used for the inverter. If both inputs are low, all the drive transistors will be off and can be ignored. In this case, the depletion load transistor Q_3 acts similar to a load resistor and pulls the output voltage high to V_{DD} .

Another example of an NMOS logic gate is shown in Fig. 1.6. The logic symbol of a *nand* gate is shown in Fig. 1.6a, and a transistor-level realization is shown in Fig. 1.6b. In this case, any zero input results in the output being a "1". In the transistor realization of Fig. 1.6, if either Q_1 or Q_2 has a low voltage on its gate terminal, the corresponding transistor would be off. This

CHAPTER 7

Latches, Flip-flops, and Synchronous System Design

We have now seen a number of different ways to realize logic gates using *CMOS* technology. All of the logic gates described thus far have been *acyclic* logic gates. That is, they have no memory or clock signals; output changes in response to input changes occur almost immediately. In this chapter, certain types of digital memory circuits will be covered. These circuits are often called *cyclic* logic circuits. These include latches, flip-flops, registers, and memory storage cells. A latch consists of a single memory location with some access circuitry. A flip-flop in an IC typically consists of two cascaded latches and perhaps some additional input logic. A register usually consists of a larger number of special-purpose latches. An IC memory consists of one or more arrays of specialized latches, called memory storage cells, with additional circuitry to access individual cells. System-level issues when designing synchronous¹ digital circuits will be discussed. In addition, two examples of synchronous logic-circuit design will be presented.

¹A synchronous logic circuit is one that has all flip-flops clocked by a single global clock. This is a recommended design style as it is less prone to errors compared to alternative design styles.

7.1 CMOS Clocked Latches

STATIC CMOS DIGITAL LATCHES

A digital latch is the simplest digital storage element. The most common way to store a digital signal is to use two cross-coupled inverters, having positive feedback, as is shown in Fig. 7.1a and b. Two cross-coupled inverters have two stable states: inverter *a* has a "1" output and inverter *b* has a "0" output, or, alternatively, inverter *a* has a "0" output and inverter *b* has a "1" output. One of these states will be somewhat arbitrarily defined to represent storing a "1", whereas the other state is defined to represent storing a "0".

In addition to having two cross-coupled inverters for digital storage, a latch requires some means of accessing it in order to set its value. One method, based on transmission gates, which have already been described in Chapter 5, is shown in Fig. 7.2.² This latch is accessed when *Clk* is high. At this time, the positive feedback of the cross-coupled inverters is *broken* since transmission-gate *X*₂ is *off*. Also, *X*₁ is connected to the input of inverter *a*. The output of inverter *a* will be equal to $\bar{D}$ and the output of inverter *b* will be equal to *D*. After *Clk* goes low, the contents of the latch will be dependent on the value of *D* just before *Clk* went low.

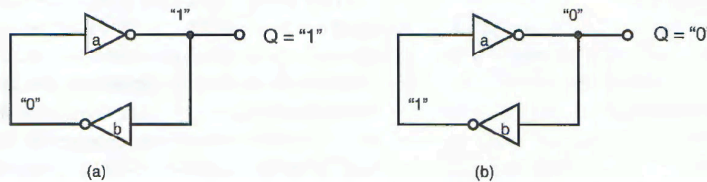


Figure 7.1 (a) Using two cross-coupled inverters to store a "1" and (b) to store a "0".

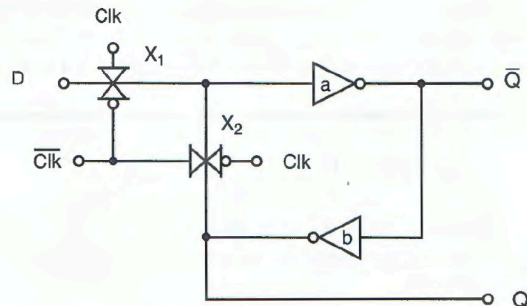


Figure 7.2 A transmission-gate-based latch.

²The transmission gates are now being represented by symbols with the circle representing the connection to the gate of the p-channel transistor.

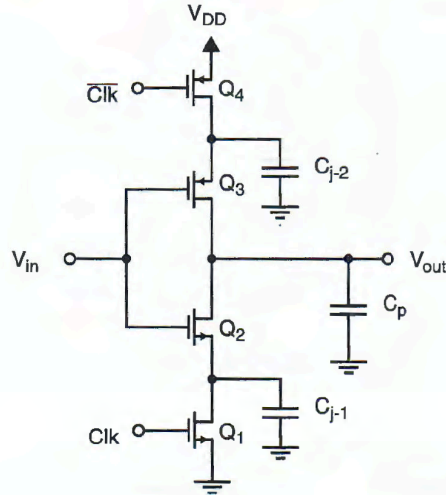


Figure 7.15 A poor realization of an inverter-based latch.

(and therefore $\overline{\text{Clk}}$ is high), the output is high impedance; in this case, the output value will be the inverse of the input value just before Clk went low. This output value will be stored on C_p , a parasitic capacitance that is due primarily to the input capacitance of the next stage and partially to the junction capacitance at the drains of Q_2 and Q_3 .

One of the reasons for the popularity of the inverter-based latch of Fig. 7.14 is that it requires less layout area than the transmission-gate-based latch of Fig. 7.13. This is true because Q_1 and Q_2 can share a common junction area that does not contain a contact. The same is true for Q_3 and Q_4 .

An inexperienced designer is warned against realizing the dynamic clocked latch as is shown in Fig. 7.15, as has often been done in the past. This realization has a charge-sharing problem after Clk has gone low, if V_{in} then changes. To see this problem, consider the case in which V_{in} is low just before Clk goes low. This will cause C_p to be charged to V_{DD} , a “1”. Also, consider the parasitic junction capacitances, C_{j-1} and C_{j-2} , shown in Fig. 7.15. Before Clk goes low, C_{j-1} will be discharged to 0 V, whereas C_{j-2} will be charged to 3.3 V. Next, consider what happens if V_{in} goes high after Clk has gone low. This will turn Q_3 off, and turn Q_2 on, which connects C_{j-1} to C_p . The charge on C_{j-1} and C_p will be shared; since C_{j-1} was originally discharged, the charge sharing will decrease the voltage on C_p . This type of detrimental charge sharing does not occur in the preferential latch of Fig. 7.14.

The detrimental charge sharing just seen is a simple example of a problem that a designer must always be wary of when designing dynamic circuits. There will be additional examples of this in Chapter 9 on advanced CMOS design techniques.

7.2 Flip-flops

Flip-flops are the traditional storage elements used to realize synchronous logic circuits. They save the state of the machine from one clock cycle to the next. They are also used to break up

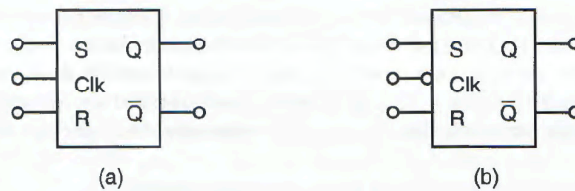


Figure 7.16 Commonly used symbols for an SR flip-flop that (a) changes at a positive-going clock edge and (b) changes at a negative-going clock edge.

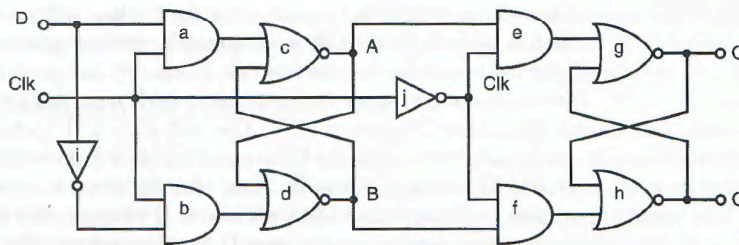


Figure 7.17 The logic diagram of a master-slave D flip-flop.

any feedback loops around a cyclic logic circuit to prevent the logic from having *race conditions* or oscillations. Flip-flops normally have signal inputs and a clock signal. Most flip-flops also have a differential output that is almost always denoted Q and $\bar{Q}$. The output values of a flip-flop change only just after a clock edge. The change is dependent on the input signals just before the clock changes. Some flip-flops also have asynchronous set and reset inputs. If either of these inputs becomes high, it immediately controls the internal storage and the output of the flip-flop, independent of the synchronous inputs and the clock input.

Common techniques for realizing flip-flops are a *master-slave* approach whereby two clocked latches are cascaded, or an *edge-sensitive* approach; the latter is not popular for integrated-circuit design because it requires clock waveforms having fast rise and fall times.

Common types of flip-flops are SR flip-flops, D flip-flops, JK flip-flops, and toggle or T flip-flops. *D flip-flops are the preferred type for integrated circuit applications.* Figure 7.16a shows a commonly used symbol for an SR flip-flop that has output changes just after a positive-going clock edge, whereas the small circle on the clock input of the SR flip-flop shown in Fig. 7.16b indicates that its outputs change just after a negative-going clock edge. SR flip-flops are not very popular in ICs because they have an indeterminate state when both inputs are high.

D FLIP-FLOP

Figure 7.17 shows a logic diagram of a possible realization of a D flip-flop, which is the flip-flop perhaps most often used in integrated circuits. The implementation shown is based on cross-coupled *nor* latches similar to those shown previously in Fig. 7.6. The first latch is

Flip
Flops

clocked by Clk , whereas the second latch is accessed on the opposite phase by $\overline{\text{Clk}}$. Thus, it is not possible for a signal to propagate from the input through both latches at one time; either the first (i.e., the *master latch*) or the second (i.e., the *slave latch*) will be deactivated and in *hold mode*. This technique of using a cascade of two latches, clocked on the opposite phases, is called a master-slave approach. This is the most commonly used approach in ICs to realize flip-flops.

The D flip-flop shown is described by the characteristic Table 7.1.

Table 7.1 implies that after the flip-flop is clocked, its output will be equal to whatever value D had just before the clock changed. To see that the flip-flop of Fig. 7.17 realizes this function, consider initially that D and Clk are both “1”s. At this time, the master latch is being accessed, but the slave is isolated. The value of A will be $\overline{D}$ and B will be equal to D . If the input D changes, both A and B will reflect that change, but the output of the flip-flop will not change since the second or slave latch is isolated by the $\overline{\text{Clk}} = “0”$ inputs to the *and* gates e and f . When Clk goes low, the master latch becomes isolated from the D input by *and* gates a and b both having “0” outputs. The contents of the master latch will reflect the D input just before the clock went low. One inverter delay after Clk goes to a “0”, $\overline{\text{Clk}}$ will go to a “1”, which now connects the slave latch to the master. At this time, the Q output of the slave latch will change, if need be, to be equal to $\overline{A}$, and $\overline{Q}$ will be equal to $\overline{B}$. Thus, after the clock has gone low, Q_{n+1} will be equal to the value of D just before Clk went low. If D changes after Clk has gone low, its change is ignored as the master is now isolated. D flip-flops are one of the most commonly used flip-flops inside ICs.

SR FLIP-FLOP

Figure 7.18 shows a master-slave realization of a very similar flip-flop, the SR flip-flop. This flip-flop is described by the characteristic Table 7.2. This type of flip-flop can either be set, reset, or left in its previous state. If S is a “1” while Clk is high, then Q_{n+1} will be a “1” after Clk goes low. If R is a “1” while Clk is high, then Q_{n+1} will be a “0”. If neither S nor R go

Table 7.1 The Characteristic Table for a D Flip-flop

D	Q_{n+1}
0	0
1	1

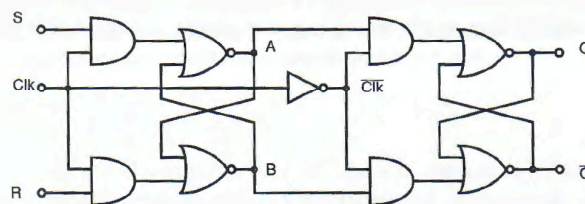


Figure 7.18 The logic diagram of a master-slave SR flip-flop.