Grant Merat

See page 17

CHAPTER VI

SYNCHRONOUS SEQUENTIAL SYNTHESIS I - COUNTERS & REGISTERS

PERSPECTIVE

The preceding chapter has presented the concept of the sequential network, as opposed to a combinational network, and has developed rigorous logical descriptions of the basic building-blocks of sequential networks called flip-flops or memories. Each of these memory devices was shown to have one or more inputs (clock, J, K, T, D, R, S) and an output Q defining the state of the memory (0 or 1). For each memory we derived a describing equation known as a characteristic equation which defined the logical behavior of the memory, independent of its application. Our task now is to define the desired logical behavior of the network in the form of tables and logical equations, known as application equations, and in effect simultaneously solve these characteristic and application equations to determine the specific combinational interconnections between the memories' outputs and their inputs. Humphrey[1] classifies all sequential switching circuits into three types.

Type one	-	Where the input is a fixed number of pulse periods and the circuit always returns to the initial state in the last period.

Type two - Where the network has an initial state but does not have a fixed cycle.

Type three - When the network does not have an identifiable initial state.

In this chapter we will deal with the most frequent synthesis task of type one networks - the design of counters and registers. The writer hastens to point out that the procedures presented here with regard to counter and register synthesis do not embrace all of the difficulties and considerations involved in the formal synthesis of a generalized synchronous sequential circuit. This general approach will be covered in the chapter to follow titled "State Tables, Reduction, and State Assignment".

SYNCHRONOUS COUNTER SYNTHESIS

The formal procedures presented by Pfister[2] for the design of synchronous counters are both seminal and pedagogically satisfying. Therefore, this tabular/algebraic method will be first presented and will subsequently be followed by transition map method referred to here as the Case Method.[3]

A sequential circuit that follows a prescribed sequence of states when driven by successive input pulses is called a counter. Such networks are a subset of a type one network according to Humphrey's classification. Let us consider the elementary case of a synchronous sequential network whose operation is prescribed by the state diagram of Fig. 6-1.

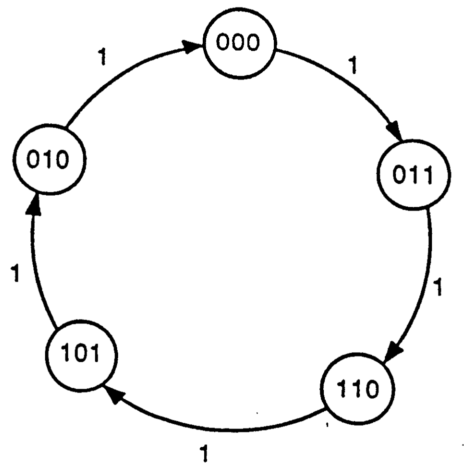


Figure 6-1 State diagram of a 3-bit sequential network

<u>The Application Equation</u> - From Fig. 6-1 we may construct an elementary state table, Table (6-1), which relates each state combination at t = n to the next-state combination at t = n + 1 where n and n + 1 are the synchronous clock times. Note that three bits of memory have the capacity of eight different state combinations. In this problem we define only five state combinations. Thus state combinations 001, 100, and 111 may be treated as logical redundancies.

(a	b	c)$_n$	(a	b	c)$_{n+1}$
0	0	0	0	1	1
0	1	1	1	1	0
1	1	0	1	0	1
1	0	1	0	1	0
0	1	0	0	0	0
- -	- -	- -			
0	0	0			

Table (6-1).

From Table (6-1) we may write three next-state equations defining when each memory will be in a logical one state at time n + 1 as a function of the states of the other memories at t = n.

$$a_{n+1} = (\bar{a}bc + ab\bar{c})_n \tag{6-1}$$

$$b_{n+1} = (\bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c)_n \tag{6-2}$$

$$c_{n+1} = (\bar{a}\bar{b}\bar{c} + ab\bar{c})_n \tag{6-3}$$

Factoring the above equations yields

$$a_{n+1} = a_n(b\bar{c})_n + \bar{a}_n(bc)_n \tag{6-4}$$

$$b_{n+1} = b_n(\bar{a}c)_n + \bar{b}_n(\bar{a}\bar{c} + ac)_n \tag{6-5}$$

$$c_{n+1} = c_n(0) + \bar{c}_n(\bar{a}\bar{b} + ab)_n \tag{6-6}$$

Thus we see that each next-state equation may be written in the factored form

$$Q_{n+1} = Q_n(g_1)_n + \bar{Q}_n(g_2)_n \tag{6-7}$$

where g_1 and g_2 are functions of the states of the other memories at t = n.

Pfister calls this equation (6-7) the <u>application equation</u>. Note that g_1 and g_2 are defined <u>only</u> by the problem definition.

<u>The Input Equation(s)</u> - We now wish to solve equation (6-7) simultaneously with the characteristic equation of the memory type we chose to implement the counting network. To illustrate this process let us choose to implement the counter using J-K memories. The characteristic equation for the J-K flip-flop is

$$Q_{n+1} = (J\bar{Q} + \bar{K}Q)_n \qquad (5\text{-}27)$$

To simultaneously solve equations (5-27) and (6-7) for J and K we will use truth table technique. Here, we will tabulate all combinations of g_1, g_2, and Q_n and solve for Q_{n+1} via equation (6-7). Then, having established the values for Q_{n+1} for each combination these values and their respective values of Q_n are used to solve for J_n and K_n. This process is shown in Table (6-2).

g_1	g_2	Q_n	$g_1Q_n + g_2\bar{Q}_n = Q_{n+1} =$	$(J\bar{Q} + \bar{K}Q)_n$	J_n	K_n
0	0	0	$0\cdot0 + 0\cdot1 = 0$	$J\cdot1 + \bar{K}\cdot0$	0	a_0
0	0	1	$0\cdot1 + 0\cdot0 = 0$	$J\cdot0 + \bar{K}\cdot1$	a_1	1
0	1	0	$0\cdot0 + 1\cdot1 = 1$	$J\cdot1 + \bar{K}\cdot0$	1	a_2
0	1	1	$0\cdot1 + 1\cdot0 = 0$	$J\cdot0 + \bar{K}\cdot1$	a_3	1
1	0	0	$1\cdot0 + 0\cdot1 = 0$	$J\cdot1 + \bar{K}\cdot0$	0	a_4
1	0	1	$1\cdot1 + 0\cdot0 = 1$	$J\cdot0 + \bar{K}\cdot1$	a_5	0
1	1	0	$1\cdot0 + 1\cdot1 = 1$	$J\cdot1 + \bar{K}\cdot0$	1	a_6
1	1	1	$1\cdot1 + 1\cdot0 = 1$	$J\cdot0 + \bar{K}\cdot1$	a_7	0

Table (6-2). Tabular Solution for J_n and K_n.

From Table (6-2) we may write equations for J_n and K_n as

$$J_n = a_1\bar{g}_1\bar{g}_2Q_n+\bar{g}_1g_2\bar{Q}_n+a_3\bar{g}_1g_2Q_n+a_5g_1\bar{g}_2Q_n+g_1g_2\bar{Q}_n+a_7g_1g_2Q_n \quad (6\text{-}8)$$

$$K_n = a_0\bar{g}_1\bar{g}_2\bar{Q}_n+\bar{g}_1\bar{g}_2Q_n+a_2\bar{g}_1g_2\bar{Q}_n+\bar{g}_1g_2Q_n+a_4g_1\bar{g}_2\bar{Q}_n+a_6g_1g_2\bar{Q}_n \quad (6\text{-}9)$$

To make the best assignment of the arbitrary constants we plot J_n and K_n on Karnaugh maps as shown in Fig. 6-2.

g_1	g_2	0	1	Q_n
0	0		a_1	
0	1	1	a_3	
1	1	1	a_7	
1	0		a_5	

J_n

g_1	g_2	0	1	Q_n
0	0	a_0	1	
0	1	a_2	1	
1	1	a_6		
1	0	a_4		

K_n

Figure 6-2 Karnaugh maps for equations (6-8) and (6-9)

Letting $a_3 = a_7 = a_0 = a_2 = 1$ and $a_1 = a_5 = a_4 = a_6 = 0$

$$J_n = g_2 \quad (6\text{-}10)$$

$$K_n = \bar{g}_1 \quad (6\text{-}11)$$

Equations (6-10) and (6-11) are the <u>input</u> <u>equations</u> of a J-K memory.

Returning to equations (6-4), (6-5) and (6-6) we may extract the values of g_1 and g_2 for this example. Here

for memory a, $g_1 = b\bar{c}$ and $g_2 = bc$

for memory b, $g_1 = \bar{a}c$ and $g_2 = \bar{a}\bar{c} + ac$

and for memory c, $g_1 = 0$ and $g_2 = \bar{a}\bar{b} + ab$

Substituting these values in equations (6-10) and (6-11) yields

$$J_{a_n} = (bc)_n \qquad K_{a_n} = (\bar{b} + c)_n \tag{6-12}$$

$$J_{b_n} = (\bar{a}\bar{c} + ac)_n \qquad K_{b_n} = (a + \bar{c})_n \tag{6-13}$$

$$J_{c_n} = (\bar{a}\bar{b} + ab)_n \qquad K_{c_n} = 1 \tag{6-14}$$

These are design equations giving the specific inputs to each J-K memory. The logic diagram is shown in Fig. 6-4.

Note that we made no use of the redundant function

$$X(a,b,c) = \sum 1,\ 4,\ 7 \tag{6-15}$$

We will now draw Karnaugh maps for equations (6-1, 2 and 3) and include the redundant terms of equation (6-15) in Fig. 6-3.

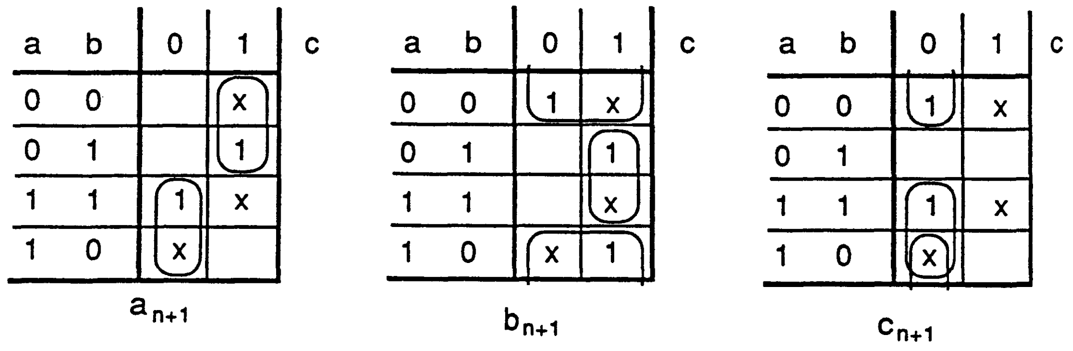


Figure 6-3 Karnaugh maps for equations (6-1,2,3,15)

These maps reduce to

$$a_{n+1} = a_n(\bar{c}_n) + \bar{a}_n(c_n) \qquad (6\text{-}16)$$

$$b_{n+1} = b_n(c_n) + \bar{b}_n(1) \qquad (6\text{-}17)$$

$$c_{n+1} = c_n(0) + \bar{c}_n(a + \bar{b})_n \qquad (6\text{-}18)$$

The parenthetical coefficients of Q_n are therefore

for memory a, $g_1 = \bar{c}$ and $g_2 = c$

for memory b, $g_1 = c$ and $g_2 = 1$

and for memory c, $g_1 = 0$ and $g_2 = a + \bar{b}$

Substituting these g values in equations (6-10, 11) yields

$$J_{a_n} = c \qquad K_{a_n} = c \tag{6-19}$$

$$J_{b_n} = 1 \qquad K_{b_n} = \bar{c} \tag{6-20}$$

$$J_{c_n} = a + \bar{b} \qquad K_{c_n} = 1 \tag{6-21}$$

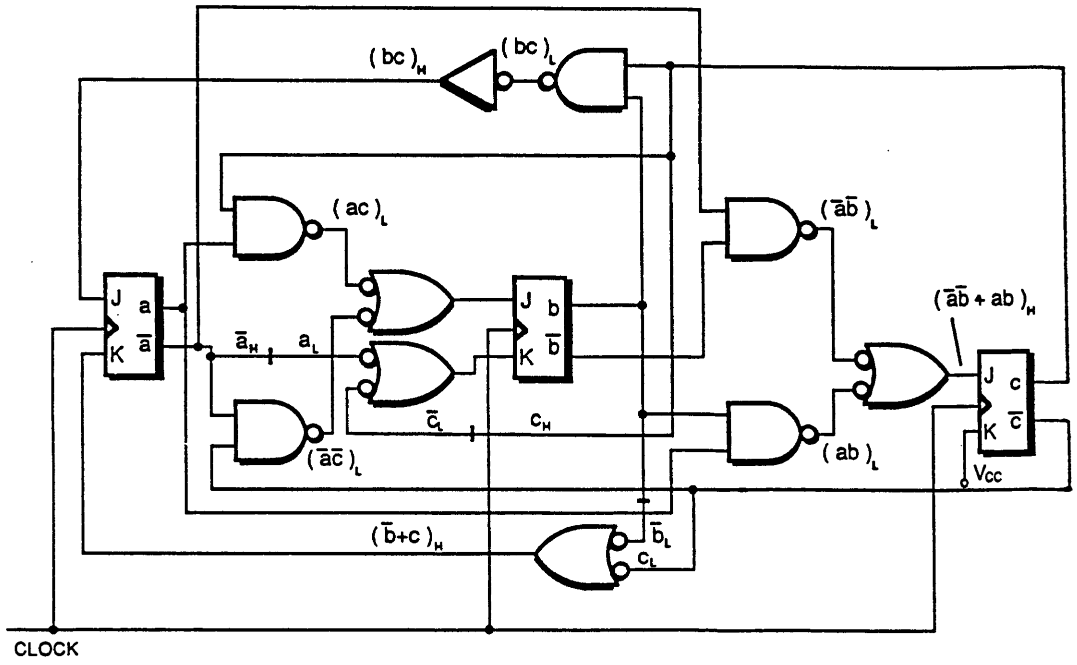


Figure 6-4 NAND (J-K) implementation of equations (6-12,13 and 14)

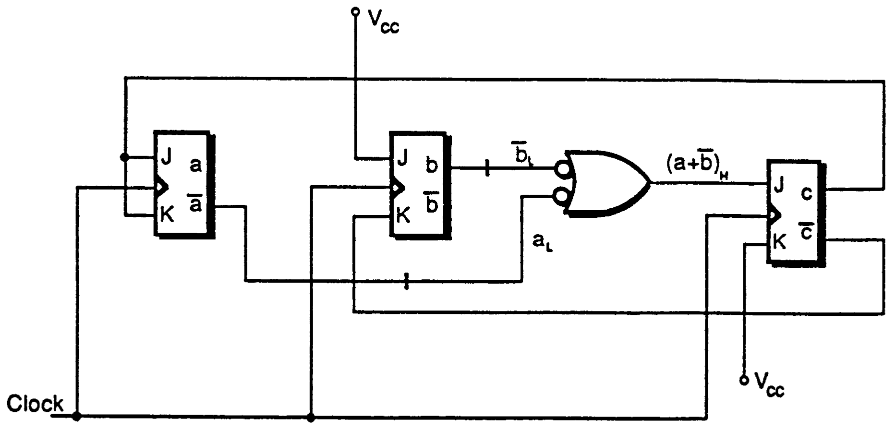


Figure 6-5 NAND (J-K) Implementation of equations (6-19,20 and 21)

Equation pairs (6-19, 20 and 21) are significantly simpler than equation pairs (6-12, 13 and 14) and result in the much-reduced logic diagram of Fig. 6-5.

<u>Input Equations for Other Memories</u> - Following the method used for the J-K memory we will now derive, with limited prose, the input equations for the D, T, R-S and J-K-T memories.

<u>D Memory</u> - The application and D characteristic equations are

$$Q_{n+1} = g_1 Q_n + g_2 \bar{Q}_n \qquad (6\text{-}7)$$

and $$Q_{n+1} = D_n \qquad (5\text{-}24)$$

Therefore $$D_n = g_1 Q_n + g_2 \bar{Q}_n \qquad (6\text{-}22)$$

<u>T Memory</u> - The application and T characteristic equations are

$$Q_{n+1} = g_1 Q_n + g_2 \bar{Q}_n \qquad (6\text{-}7)$$

and $$Q_{n+1} = T_n \bar{Q}_n + \bar{T}_n Q_n \qquad (5\text{-}32)$$

Using Table (6-3) we solve equations (6-7) and (5-32) simultaneously for $T = f(g_1, g_2, Q_n)$.

g_1	g_2	Q_n	$g_1Q_n + g_2\bar{Q}_n =$	Q_{n+1}	$= (\bar{T}Q + T\bar{Q})_n$	T_n
0	0	0	=	0	$= \bar{T}0 + T1$	0
0	0	1	=	0	$= \bar{T}1 + T0$	1
0	1	0	=	1	$= \bar{T}0 + T1$	1
0	1	1	=	0	$= \bar{T}1 + T0$	1
1	0	0	=	0	$= \bar{T}0 + T1$	0
1	0	1	=	1	$= \bar{T}1 + T0$	0
1	1	0	=	1	$= \bar{T}0 + T1$	1
1	1	1	=	1	$= \bar{T}1 + T0$	0

Table (6-3). Tabular Solution for T_n.

From Table (6-3) we may write the specific equation for $T = f(g_1, g_2, Q_n)$ as

$$T_n = \bar{g}_1\bar{g}_2Q_n + \bar{g}_1g_2\bar{Q}_n + \bar{g}_1g_2Q_n + g_1g_2\bar{Q}_n$$

which reduces to

$$T_n = \bar{g}_1Q_n + g_2\bar{Q}_n \tag{6-23}$$

<u>R-S Memory</u> - The application and R-S characteristic equations are

$$Q_{n+1} = g_1Q_n + g_2\bar{Q}_n \tag{6-7}$$

$$Q_{n+1} = (S + \bar{R}Q)_n \qquad (RS)_n \equiv 0 \tag{5-16}$$

Using Table (6-4) we solve equations (6-7) and (5-16) simultaneously for $R_n = f_1(g_1, g_2, Q_n)$ and $S_n = f_2(g_1, g_2, Q_n)$.

g_1	g_2	Q_n	$g_1Q_n + g_1\bar{Q}_n =$	$Q_{n+1} = S + \bar{R}Q_n$	S_n	R_n
0	0	0	$0 \cdot 0 + 0 \cdot 1$	$0 = S + \bar{R}\ 0$	0	a_0
0	0	1	$0 \cdot 1 + 0 \cdot 0$	$0 = S + \bar{R}\ 1$	0	1
0	1	0	$0 \cdot 0 + 1 \cdot 1$	$1 = S + \bar{R}\ 0$	1	0
0	1	1	$0 \cdot 1 + 1 \cdot 0$	$0 = S + \bar{R}\ 1$	0	1
1	0	0	$1 \cdot 0 + 0 \cdot 1$	$0 = S + \bar{R}\ 0$	0	a_4
1	0	1	$1 \cdot 1 + 0 \cdot 0$	$1 = S + \bar{R}\ 1$	a_5	0
1	1	0	$1 \cdot 0 + 1 \cdot 1$	$1 = S + \bar{R}\ 0$	1	0
1	1	1	$1 \cdot 1 + 1 \cdot 0$	$1 = S + \bar{R}\ 1$	a_7	0

Table (6-4). Tabular Solution for R_n and S_n.

From Table (6-4) we may write equations for R_n and S_n as functions of a, g_1, g_2 and Q_n.

$$S_n = \bar{g}_1 g_2 \bar{Q}_n + a_5 g_1 \bar{g}_2 Q_n + g_1 g_2 \bar{Q}_n + a_7 g_1 g_2 Q_n$$

$$R_n = a_0 \bar{g}_1 \bar{g}_2 \bar{Q}_n + \bar{g}_1 \bar{g}_2 Q_n + \bar{g}_1 g_2 Q_n + a_4 g_1 \bar{g}_2 \bar{Q}_n$$

Letting $a_0 = a_4 = a_5 = a_7 = 0$ the equations reduce to

$$S_n = g_2 \bar{Q}_n \qquad (6-24)$$

and $R_n = \bar{g}_1 Q_n$ (6-25)

<u>J-K-T Memory</u> - The application and J-K-T characteristic equations are

$$Q_{n+1} = g_1 Q_n + g_2 \bar{Q}_n \qquad (6-7)$$

and $Q_{n+1} = (T\bar{Q} + \bar{K}\bar{T}Q + J\bar{Q})_n \quad (JKT = JT = KT)_n \equiv 0$ (5-50)

Using Table (6-5) we solve equations (6-7) and (5-50) simultaneously for $J_n = f_3(g_1, g_2, Q_n)$, $K_n = f_4(g_1, g_2, Q_n)$ and $T_n = f_5(g_1, g_2, Q_n)$.

g_1	g_2	Q_n	$Q_{n+1} = (T\bar{Q} + \bar{K}\bar{T}Q + J\bar{Q})_n$	J	K	T
0	0	0	0 = 1 + 0 + 1	0	a_0	0
0	0	1	0 = 0 + 1 + 0	a_1	a_1	$\bar{a}_1$
0	1	0	1 = 1 + 0 + 1	a_2	a_2	$\bar{a}_2$
0	1	1	0 = 0 + 1 + 0	a_3	a_3	$\bar{a}_3$
1	0	0	0 = 1 + 0 + 1	0	a_4	0
1	0	1	1 = 0 + 1 + 0	a_5	0	0
1	1	0	1 = 1 + 0 + 1	a_6	a_6	$\bar{a}_6$
1	1	1	1 = 0 + 1 + 0	a_7	0	0

Table (6-5). Tabular Solution for J_n, K_n and T_n.

From Table (6-5) we may write equations for J_n, K_n and T_n as functions of a, g_1, g_2 and Q_n.

$$J_n = a_1\bar{g}_1\bar{g}_2Q_n + a_2\bar{g}_1g_2\bar{Q}_n + a_3\bar{g}_1g_2Q_n + a_5g_1\bar{g}_2Q_n + a_6g_1g_2\bar{Q}_n + a_7g_1g_2Q_n \quad (6\text{-}26)$$

$$K_n = a_0\bar{g}_1\bar{g}_2\bar{Q}_n + a_7\bar{g}_1\bar{g}_2Q_n + a_2\bar{g}_1g_2\bar{Q}_n + a_3\bar{g}_1g_2Q_n + a_4g_1\bar{g}_2\bar{Q}_n + a_6g_1g_2\bar{Q}_n \quad (6\text{-}27)$$

$$T_n = \bar{a}_1\bar{g}_1\bar{g}_2Q_n + \bar{a}_2\bar{g}_1g_2\bar{Q}_n + \bar{a}_3\bar{g}_1g_2Q_n + \bar{a}_6g_1g_2\bar{Q}_n \quad (6\text{-}28)$$

Figure 6-6 shows the Karnaugh maps of these equations.

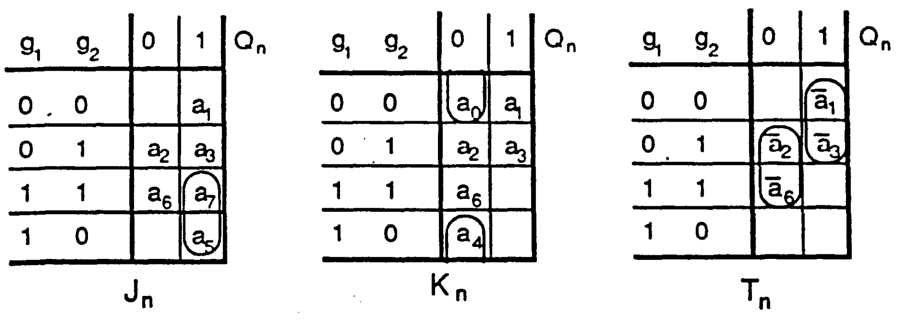


Figure 6-6 Karnaugh maps for equations (6-26,27 and 28)

A particular solution set is obtained by letting $a_0 = a_4 = a_5 = a_7 = 1$ and $a_1 = a_2 = a_3 = a_6 = 0$, yielding

$$J_n = g_1 Q_n \tag{6-29}$$

$$K_n = \bar{g}_2 \bar{Q}_n \tag{6-30}$$

$$T_n = \bar{g}_1 Q_n + g_2 \bar{Q}_n \tag{6-31}$$

<u>D-T Memory</u> - The application and D-T characteristic equations are

$$Q_{n+1} = g_1 Q_n + g_2 \bar{Q}_n \tag{6-7}$$

and $$Q_{n+1} = (D + \bar{T}Q + T\bar{Q})_n \quad \text{where } (DT)_n \equiv 0 \tag{5-52}$$

Using Table (6-6) we solve equation (6-7) and (5-52) simultaneously for $D = f_4(g_1, g_2, Q_n)$.

g_1	g_2	Q_n	Q_{n+1}	=	(D	+	$\bar{T}$ Q	+	T $\bar{Q})_n$	D	T
0	0	0	0	=	D	+	$\bar{T}\cdot 0$	+	$T\cdot 1$	0	0
0	0	1	0	=	D	+	$\bar{T}\cdot 1$	+	$T\cdot 0$	0	1
0	1	0	1	=	D	+	$\bar{T}\cdot 0$	+	$T\cdot 1$	a_2	$\bar{a}_2$
0	1	1	0	=	D	+	$\bar{T}\cdot 1$	+	$T\cdot 0$	0	1
1	0	0	0	=	D	+	$\bar{T}\cdot 0$	+	$T\cdot 1$	0	0
1	0	1	1	=	D	+	$\bar{T}\cdot 1$	+	$T\cdot 0$	a_5	0
1	1	0	1	=	D	+	$\bar{T}\cdot 0$	+	$T\cdot 1$	a_6	$\bar{a}_6$
1	1	1	1	=	D	+	$\bar{T}\cdot 1$	+	$T\cdot 0$	a_7	0

Table (6-6). Tabular Solution for D_n and T_n.

From Table (6-6) we may write equations for D_n and T_n as functions of a, g_1, g_2, and Q_n.

$$D_n = a_2\bar{g}_1 g_2 \bar{Q}_n + a_5 g_1 \bar{g}_2 Q + a_6 g_1 g_2 \bar{Q}_n + a_7 g_1 g_2 Q_n \qquad (6-32)$$

$$T_n = \bar{g}_1 \bar{g}_2 Q_n + \bar{a}_2 \bar{g}_1 g_2 \bar{Q} + \bar{g}_1 g_2 Q_n + \bar{a}_6 g_1 g_2 \bar{Q}_n \qquad (6-33)$$

A particular solution set is obtained by letting $a_2 = a_6 = 1$ and $a_5 = a_7 = 0$.

$$D_n = g_2 \bar{Q}_n \qquad (6-34)$$

$$T_n = \bar{g}_1 Q_n \qquad (6-35)$$

A summary of the characteristic and input equations for various memory configurations is shown in Table (6-7).

MEMORY TYPE	CHARACTERISTIC EQUATION	RESTRICTIONS	INPUT EQUATIONS
T	$Q_{n+1}=(\bar{T}Q+T\bar{Q})_n$	-	$T_n=\bar{g}_1Q_n+g_2\bar{Q}_n$
R-S	$Q_{n+1}=(S+\bar{R}Q)_n$	$(RS)_n\equiv0$	$R_n=\bar{g}_1Q_n$ $S=g_2\bar{Q}_n$
J-K	$Q_{n+1}=(J\bar{Q}+\bar{K}Q)_n$	-	$J_n=g_2$ $K_n=\bar{g}_1$
D	$Q_{n+1}=D_n$	-	$D_n=g_1Q_n+g_2\bar{Q}_n$
R-S-T*	$Q_{n+1}=(S+T\bar{Q}+\bar{R}\bar{T}Q)_n$	$(RS=ST=RT=RST)_n\equiv0$	$R_n=\bar{g}_1\bar{g}_2$ $S_n=g_1g_2$ $T_n=\bar{g}_1g_2$
J-K-T	$Q_{n+1}=[(J+T)\bar{Q}+\bar{K}\bar{T}Q]_n$	$(JK=KT=JKT)_n\equiv0$	$J_n=g_1Q_n$ $K_n=\bar{g}_2\bar{Q}_n$ $T_n=\bar{g}_1Q_n+g_2\bar{Q}_n$
D-T	$Q_{n+1}=(D+\bar{T}Q+T\bar{Q})_n$	$(DT)_n\equiv0$	$D_n=g_2\bar{Q}_n$ $T_n=\bar{g}_1Q_n$

Table (6-7). Characteristic and Input Equations for Selected Memory Configurations Based on a General Application Equation of the Form $Q_{n+1} = g_1Q_n + g_2\bar{Q}_n$. *The Equations for this Memory are Not Derived in the Text.

Effective use of Pfister's algebraic method for determining the memory input equations requires that one make use of the redundancies when reducing the input equations and that one guard against reducing the input equations to a form where the terms g_1 and g_2 lose their identities as coefficients of Q_n and $\bar{Q}_n$ respectively.

The Case Method - An alternate procedure to Pfister's algebraic determination of the memory input equations in a synchronous sequential circuit is an algorithmic technique based on the interpretation of transition Karnaugh maps.[4] This method was first suggested by Professor Roger Brockett, while an undergraduate, in a class paper at Case Institute of Technology in 1958. The author subsequently expanded Brockett's work into what is referred to here as the Case Method.[3]

This procedure requires only a sequential tabulation of state assignments whose values and whose transitions are then plotted on Karnaugh maps for each memory. This process will be illustrated using the example given in Fig. 6-1. The sequence and next-state are listed in Table (6-8).

	PRESENT STATE			NEXT STATE		
	(a	b	$c)_n$	(a	b	$c)_{n+1}$
	0	0	0	0	1	1
↓	0	1	1	1	1	0
	1	1	0	1	0	1
	1	0	1	0	1	0
	0	1	0	0	0	0
	- -	- -	- -			
	0	0	0			

Table (6-8).

<u>The Transition Map</u> - Let us focus our attention on memory $\underline{a}$. When it is (0) in state (000) it stays (0) in the next state combination (011). We will call this a <u>static</u> <u>zero</u> and designate it as (0). When it is (0) in state combination (011) it goes to (1) in the next state combination (110). We will call this an (α) transition. Whenever $\bar{Q}_n Q_{n+1} = 1$ an (α) transition is indicated. When $\underline{a}$ is a (1) in the (110) state it stays a (1) in the next state combination. We will call this a <u>static</u> <u>one</u> and designate it as (1). Finally when $\underline{a}$ is a (1) in the (101) state combination it goes to (0) in the next state combination. We will call this a (β) transition. Whenever $Q_n \bar{Q}_{n+1} = 1$ a (β) transition is indicated. These symbols (0, 1, α, and β) are then plotted on Karnaugh maps for each memory. The formal definition of the transitions is shown in Table (6-9).

TRANSITION	LOGIC	TRANSITION SYMBOL
$0_n \rightarrow 1_{n+1}$	$\bar{Q}_n Q_{n+1} = 1$	α
$0_n \rightarrow 0_{n+1}$	$\bar{Q}_n \bar{Q}_{n+1} = 1$	0
$1_n \rightarrow 0_{n+1}$	$Q_n \bar{Q}_{n+1} = 1$	β
$1_n \rightarrow 1_{n+1}$	$Q_n Q_{n+1} = 1$	1

Table (6-9). Transition Symbols.

For our example, the transition maps are shown in Fig. 6-7. The Case Method algorithm involves the interpretation of these transition maps to permit the determination of the simplest memory input equations in a single, simple, visual step.

a	b	0	1	c
0	0	0	x	
0	1	0	α	
1	1	1	x	
1	0	x	β	

a_{n+1}

a	b	0	1	c
0	0	α	x	
0	1	β	1	
1	1	β	x	
1	0	x	α	

b_{n+1}

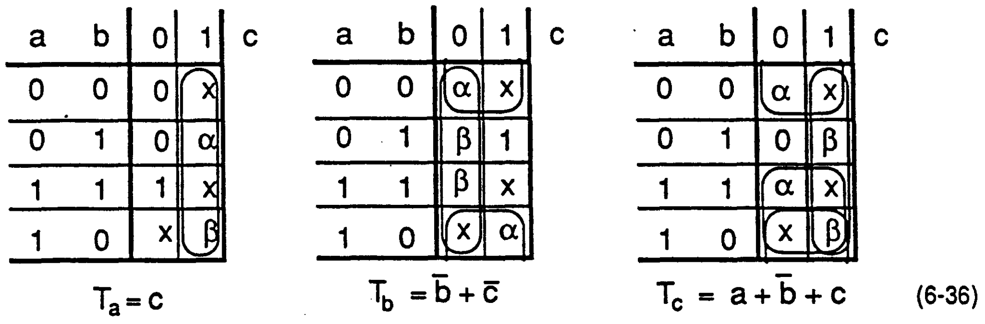
a	b	0	1	c
0	0	α	x	
0	1	0	β	
1	1	α	x	
1	0	x	β	

c_{n+1}

Figure 6-7 Transition maps for example given in figure 6-1

<u>Interpretive Rules for a T Memory</u> - The T memory has a single input which can produce <u>only</u> α or β transitions. Therefore, the <u>T</u> input must embrace all α and β map transitions and may embrace any redundant terms (X) which assist the reduction process. For our example problem the map reductions for the <u>T</u> memory are as shown in Fig. 6-8.

<u>The T input rule - Cover all α and β terms and optimally use X terms where they facilitate reduction.</u>



a	b	0	1
0	0	0	x
0	1	0	α
1	1	1	x
1	0	x	β

$T_a = c$

a	b	0	1
0	0	α	x
0	1	β	1
1	1	β	x
1	0	x	α

$T_b = \bar{b} + \bar{c}$

a	b	0	1
0	0	α	x
0	1	0	β
1	1	α	x
1	0	x	β

$T_c = a + \bar{b} + c$ (6-36)

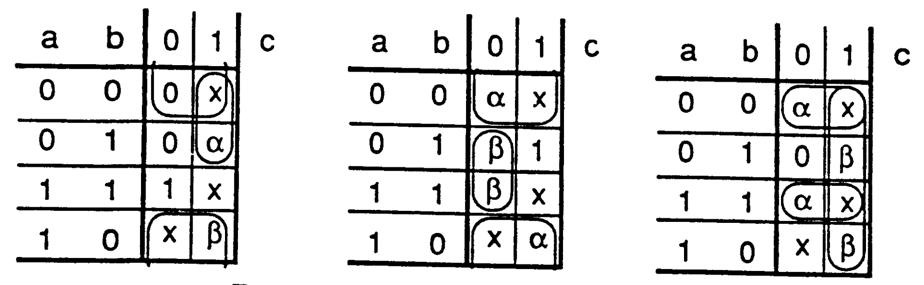
Figure 6-8 Transition map interpretations for T memories

Note the ease at which we determined the input equations compared to the algebraic tasks using Pfister's algebraic method. Note too that we have made full utilization of the algebraic redundancies.

<u>Interpretive Rules for a R-S Memory</u> - The R-S memory has two inputs which can either <u>reset</u> the memory to (0) or <u>set</u> it to a (1). Further, if the memory is in the (0) state, a <u>reset</u> input continues this (0) state (static 0). If the memory is in the (1) state, a <u>set</u> input continues this (1) state (static 1).

<u>The R-S input rules - The S input must cover all α terms; R must cover all β terms. Optional coverings with S are (1) and X terms; optional coverings with R are (0) and X terms.</u>

For our example problem the map reductions for a R-S memory are shown in Fig. 6-9.



$S_a = \bar{a}c \quad R_a = \bar{b} \qquad S_b = \bar{b} \quad R_b = b\bar{c} \qquad S_c = \bar{a}\bar{b} + ab \quad R_c = c \qquad (6\text{-}37)$

Figure 6-9 Transition map interpretation for R-S memories

<u>Interpretive Rules for a J-K memory</u> - The rules for interpreting the transition maps for a J-K memory are slightly less obvious than those for a T or a R-S memory. Here α transitions can be produced when J = 1 and K = 0 or when J = 1 and K = 1 (since a J-K memory complements on a simultaneous J-K input). β transitions can be produced when J = 0 and K = 1 or when J = 1 and K = 1. Static (1) transitions can be produced when J = 0 and K = 0 or when J = 1 and K = 0. Lastly, static (0) transitions can be produced when J = 0 and K = 0 or when J = 0 and K = 1. This prose is tabulated in Table (6-10).

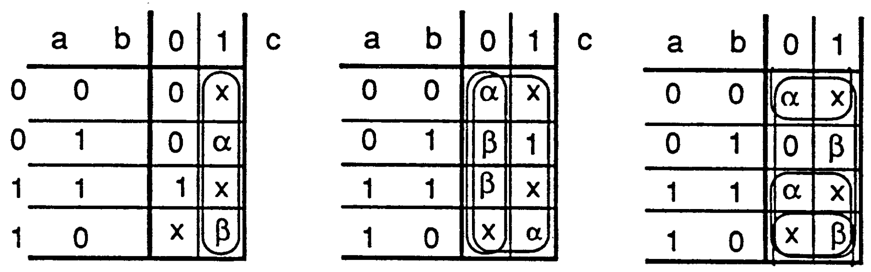
Q_n	J_n	K_n	TRANSITION
0	1	0	α
0	1	1	α
1	0	1	β
1	1	1	β
1	0	0	1
1	1	0	1
0	0	0	0
0	0	1	0

Table (6-10). Tabulation of Transitions as a Function of Q_n, J_n, and K_n.

It may be observed that J_n does not influence β or (1) transitions and that K_n does not influence α or (0) transitions.

<u>The J-K input rules - The J input must cover all α terms; the K input must cover all β terms. Optional coverings with J are the (1), β, and X terms; optional coverings with K are the (0), α, and X terms.</u>

For our example problem the map reductions for a J-K memory are shown in Fig. 6-10.



$J_a = c$ $K_a = c$ $J_b = 1$ $K_b = \bar{c}$ $J_c = a + \bar{b}$ $K_c = 1$ (6-38)

Figure 6-10 Transition map interpretations for J-K memories

<u>Interpretive Rules for a D Memory</u> - The single D input of a D memory must produce all α transitions and static (1) transitions. Static (0) and β transitions occur only when D = 0. Table (6-11) tabulates the memories' transitions as a function of Q_n and D.

Q_n	D_n	TRANSITION
0	0	0
0	1	α
1	0	β
1	1	1

Table (6-11). Tabulation of Transitions as a Function of Q_n and D.

<u>The D input rules - The D input must cover all α terms and static (1) terms; optional coverings with D are the X terms.</u>

For our example problem, the map reductions for a D memory are shown in Fig. 6-11.

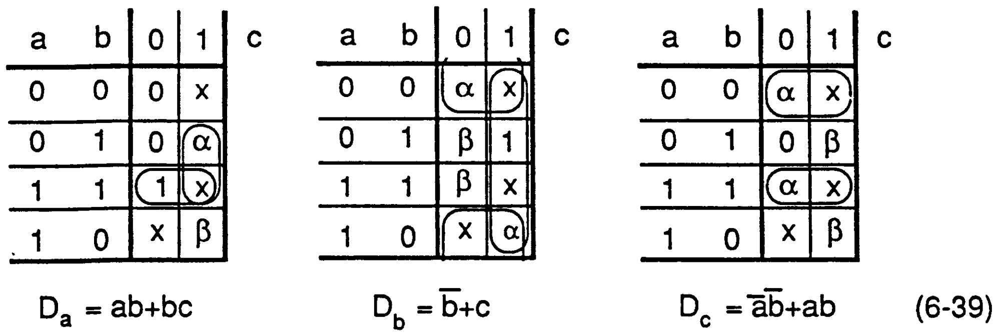


Figure 6-11 Transition map interpretations for D memories

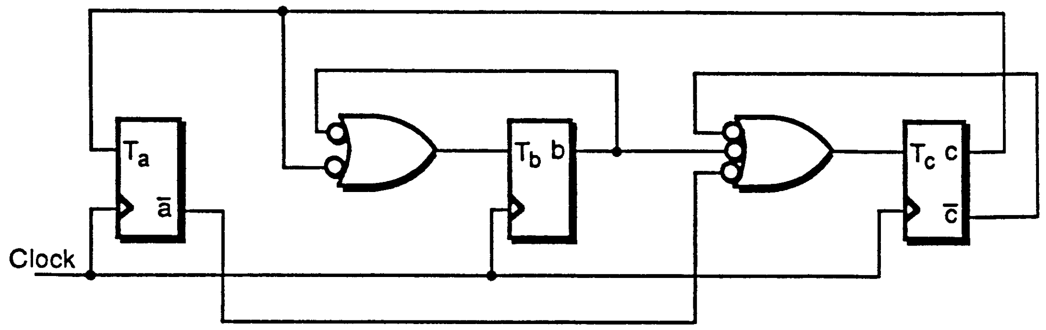


Figure 6-12 (a) -NAND-(T) implementation of equation set (6-36)

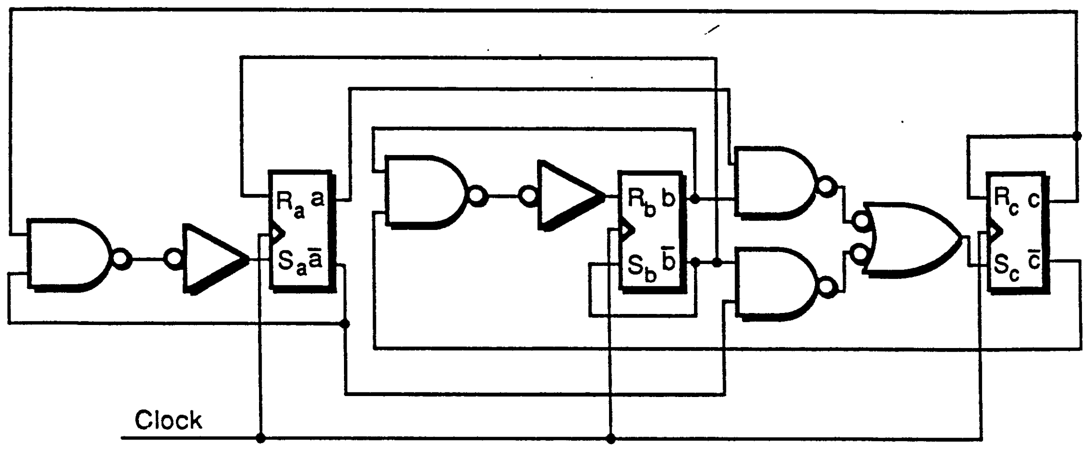


Figure 6-12 (b) -NAND-(R-S) implementation of equation set (6-37)

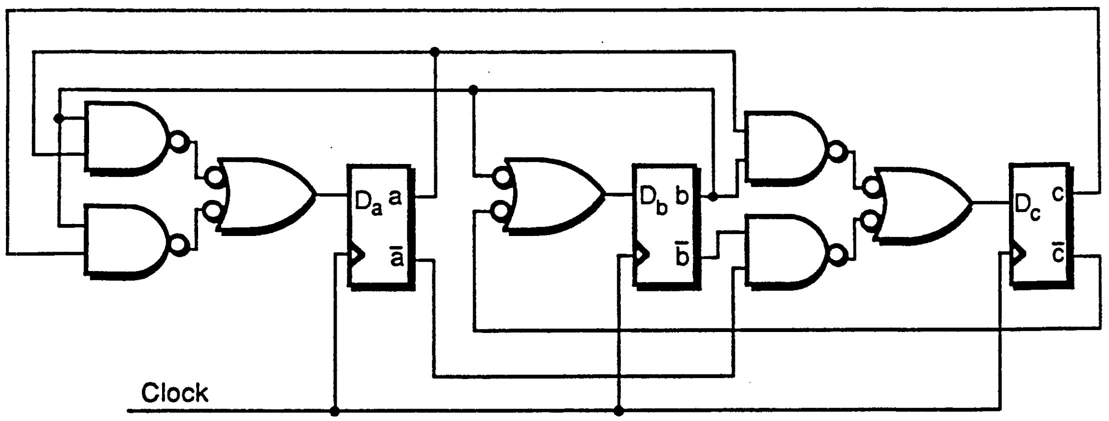


Figure 6-12 (c) -NAND-(D) implementation of equation set (6-39)

Note: Also see Fig. 6-5.

Table (6-12) summarizes the interpretive rules for transition maps for selected memories and Fig. 6-12(a),(b), and (c) shows the actual logic diagrams of the T, R-S, and D implementations of the example problem. Note that the J-K implementation was previously shown in Fig. 6-5.

The practical advantage of the transition map method over the algebraic method lies in speed of execution. Here, the designer may move directly from a state table to the transition maps to the reduced input equations of any memory configuration while avoiding <u>any</u> intermediate algebraic processes. Because of this ease of execution, the designer can rapidly explore alternate memory configurations and/or mixes to achieve the simplest possible logical solution for the original state table.

INPUT	ESSENTIAL COVERINGS			OPTIONAL COVERINGS				
T	α	β						X
R		β		0				X
S	α				1			X
J	α				1		β	X
K		β		0		α		X
D	α		1					X

Table (6-12). Transition Map Reduction Rules.

SYNCHRONOUS COUNTER DESIGNS

With the benefit of the preceding discussions on Pfister's algebraic method and the Case method for synchronous sequential synthesis we will now proceed to synthesize some common counter designs.

A Four-Bit Binary Counter - A four-bit (a,b,c,d) binary counter has sixteen state combinations of abcd - 0000, 0001, 0010, ----, 1111. These states are tabulated in Table (6-13). From this state table we may construct the transition maps for each bit as shown in Fig. 6-13. From these transition maps we may then write the appropriate reduced input equations for implementing the logic with the memory type of our selection. Here we will solve for J-K, T, and D inputs. The generic logic diagram for the J-K solution is shown in Fig. 6-14.

(a	b	c	$d)_n$	(a	b	c	$d)_{n+1}$
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Table (6-13). State Sequence for a Four-Bit Binary Counter.

a	b	0	1	1	0	d
		0	0	1	1	c
0	0	0	0	0	0	
0	1	0	0	α	0	
1	1	1	1	β	1	
1	0	1	1	1	1	

$(a)_{n+1}$

$J_a = bcd$

$K_a = bcd$

$D_a = \bar{a}bcd + a\bar{c} + a\bar{d} + a\bar{b}$
$= \bar{a}bcd + a(\bar{c}+\bar{d}+\bar{b})$

$T_a = bcd$

a	b	0	1	1	0	d
		0	0	1	1	c
0	0	0	0	α	0	
0	1	1	1	β	1	
1	1	1	1	β	1	
1	0	0	0	α	0	

$(b)_{n+1}$

$J_b = cd$
$k_b = cd$ } (6-40)

$D_b = \bar{b}cd + b\bar{c} + b\bar{d}$
$= \bar{b}cd + b(\bar{c}+\bar{d})$ (6-41)

$T_b = cd$ (6-42)

a	b	0	1	1	0	d
		0	0	1	1	c
0	0	0	α	β	1	
0	1	0	α	β	1	
1	1	0	α	β	1	
1	0	0	α	β	1	

$(c)_{n+1}$

$J_c = d$

$K_c = d$

$D_c = \bar{c}d + c\bar{d}$

$T_c = d$

a	b	0	1	1	0	d
		0	0	1	1	c
0	0	α	β	β	α	
0	1	α	β	β	α	
1	1	α	β	β	α	
1	0	α	β	β	α	

$(d)_{n+1}$

$J_d = 1$
$K_d = 1$ } (6-40)

$D_d = \bar{d}$ (6-41)

$T_d = 1$ (6-42)

Figure 6-13 Transition maps for a four bit binary counter

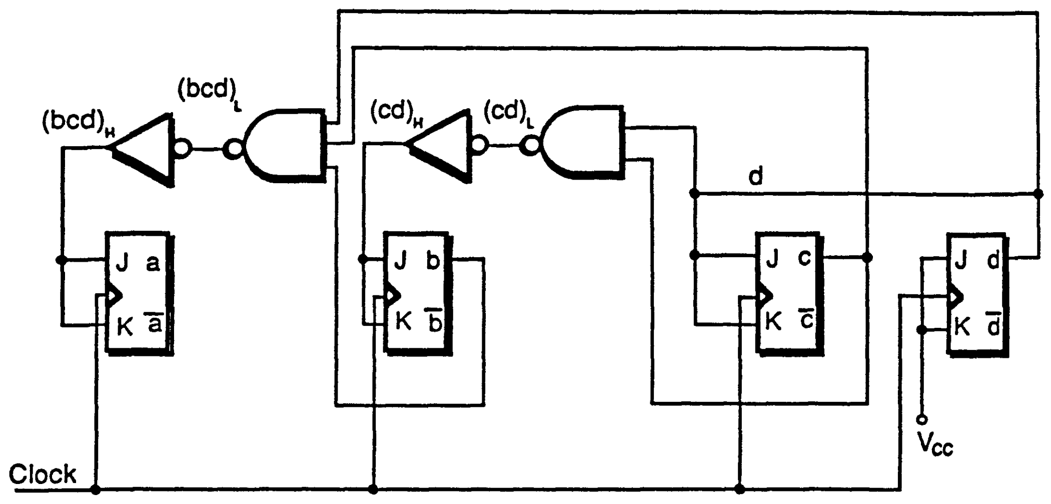


Figure 6-14 Four-bit binary counter implementing equation sets (6-40) or (6-42)

<u>An 8-4-2-1 Binary Coded Decimal Decade</u> - There are seventeen different weight assignments among four bits to represent the decimal digits 0 through 9. For example, 7-4-2-1, 3-3-2-1, etc. In the absence of further definition, a binary coded decimal (BCD) decade is assumed to have the weights 8-4-2-1. The forward-counting sequence for this decade is shown in Table (6-14).

(a	b	c	$d)_n$	(a	b	c	$d)_{n+1}$
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
- -	- -	- -	- - -				
0	0	0	0				

Table (6-14). Counting Sequence for an 8-4-2-1 Forward Counting Decade.

The transition maps for each position are shown in Fig. 6-15 together with their interpretation for a J-K implementation. The resulting logic diagram using generic J-K memories is shown in Fig. 6-16. It should be noted that there are six state combinations (1010, 1011, 1100, 1101, 1110 and 1111) that are redundant to this 8-4-2-1 sequence and are thus noted by (X) on the transition maps.

		0	1	1	0	d
a	b	0	0	1	1	c
0	0	0	0	0	0	
0	1	0	0	α	0	
1	1	x	x	x	x	
1	0	1	β	x	x	

$(a)_{n+1}$

$J_a = bcd$
$K_a = d$

		0	1	1	0	d
a	b	0	0	1	1	c
0	0	0	0	α	0	
0	1	1	1	β	1	
1	1	x	x	x	x	
1	0	0	0	x	x	

$(b)_{n+1}$

$$\left.\begin{aligned} J_b &= cd \\ K_b &= cd \end{aligned}\right\} \text{(6-43 a,b)}$$

		0	1	1	0	d
a	b	0	0	1	1	c
0	0	0	α	β	1	
0	1	0	α	β	1	
1	1	x	x	x	x	
1	0	0	0	x	x	

$(c)_{n+1}$

$J_c = \bar{a}d$
$K_c = \bar{a}d$

		0	1	1	0	d
a	b	0	0	1	1	c
0	0	α	β	β	α	
0	1	α	β	β	α	
1	1	x	x	x	x	
1	0	α	β	x	x	

$(d)_{n+1}$

$$\left.\begin{aligned} J_d &= 1 \\ K_d &= 1 \end{aligned}\right\} \text{(6-43 c,d)}$$

Figure 6-15 Transition maps for an 8-4-2-1 coded decimal counting decade

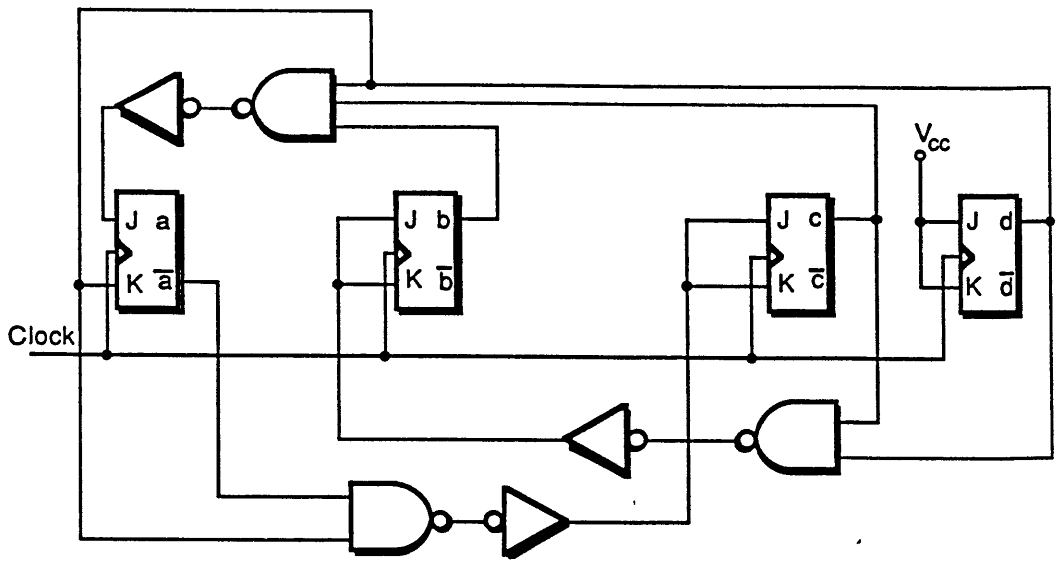


Figure 6-16 The logical J-K implementation of an 8-4-2-1 BCD counting decade

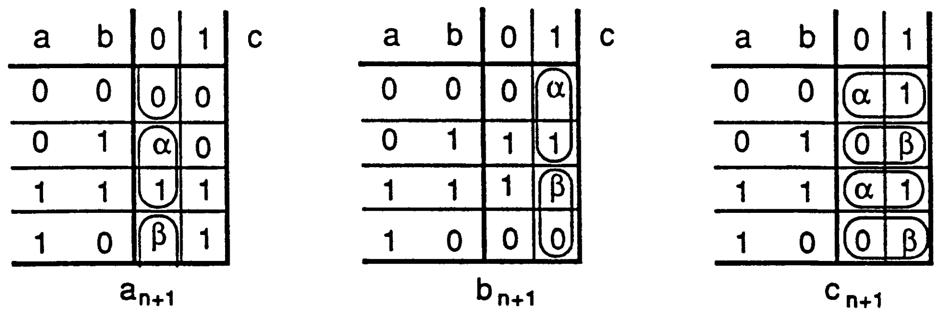
A Gray Code Counter - A Gray Code[5] is one of a class of codes called unit-distance codes. Its singular property is that only one bit of the code group changes when the code representation changes by a single unit. For example, in a three-bit Gray code, decimal 5 is represented by 111 and decimal 6 is represented by 101. Here only the middle digit changes. The bits on such a unit-distance code have no weights, i.e. the bit positions do not represent coefficients, a weighted number defined by a radix and a position number.

The advantages of this unit distance property will be discussed under the chapter on Codes. For our purposes here we will use it only as a vehicle to demonstrate the synchronous sequential design procedures. Table (6-15) shows the Gray sequence for the decimal digits 0 through 7.

DECIMAL DIGIT	GRAY CODE REPRESENTATION		
	a	b	c
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Table (6-15). Three-Bit Gray Code Sequence.

For the forward sequence given in Table (6-15) we may construct the transition maps and derive the input equations as shown in Fig. 6-17. The logic diagram implementing the input equations is shown in Fig. 6-18.



$$J_a = b\bar{c} \qquad J_b = \bar{a}c \qquad J_c = ab + \bar{a}\bar{b}$$
$$K_a = \bar{b}\bar{c} \qquad K_b = ac \qquad K_c = \bar{a}b + a\bar{b}$$
(6-44)

Figure 6-17 Transition maps and input equations for a J-K implemented three-bit Gray code forward counter

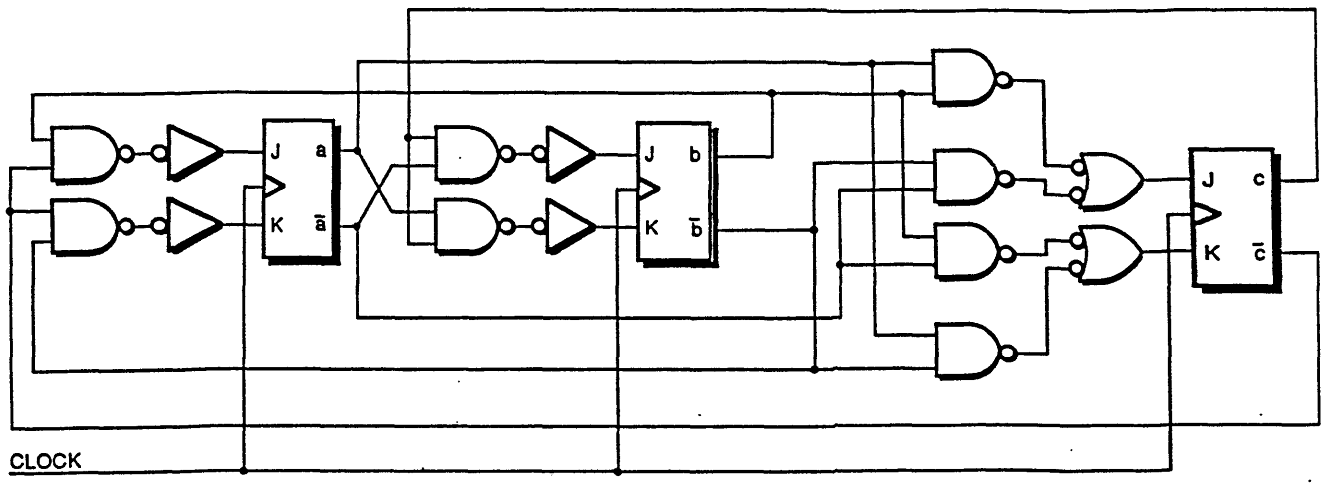


Figure 6-18 J-K implementation of a three-bit forward Gray code counter

<u>A Forward/Backward Three-Bit Binary Counter</u> - For this design we will assume two input data lines F (forward) and B (backward). Receipt of a F pulse increments the count; receipt of a B pulse decrements the count. We will not concern ourselves with the problem of coincidence. The state sequence is shown in Table (6-16). Here, D is the count direction control flip-flop. When D is set, the counter is to count in the forward direction. When D is reset, the counter is to count in the backward direction. Figure 6-19 shows the transition maps derived from Table (6-16).

D	(a	b	c)$_n$	D	(a	b	c)$_{n+1}$
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0
0	1	1	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	0	0	0	0	1	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0
0	0	0	0	0	1	1	1

Table (6-16). Forward-Backward Counting Sequence Under Control of D.

		0	1	1	0	c
D	a	0	0	1	1	b
0	0	α	0	0	0	
0	1	β	1	1	1	
1	1	1	1	β	1	
1	0	0	0	α	0	

$(a)_{n+1}$

$T_a = Dbc + \bar{D}\bar{b}\bar{c}$

		0	1	1	0	c
D	a	0	0	1	1	b
0	0	α	0	1	β	
0	1	α	0	1	β	
1	1	0	α	β	1	
1	0	0	α	β	1	

$(b)_{n+1}$

$T_b = \bar{D}\bar{c} + Dc$

		0	1	1	0	c
D	a	0	0	1	1	b
0	0	α	β	β	α	
0	1	α	β	β	α	
1	1	α	β	β	α	
1	0	α	β	β	α	

$(c)_{n+1}$

$T_c = 1$ (6-45)

Figure 6-19 Transition maps and input equations for table 6-16

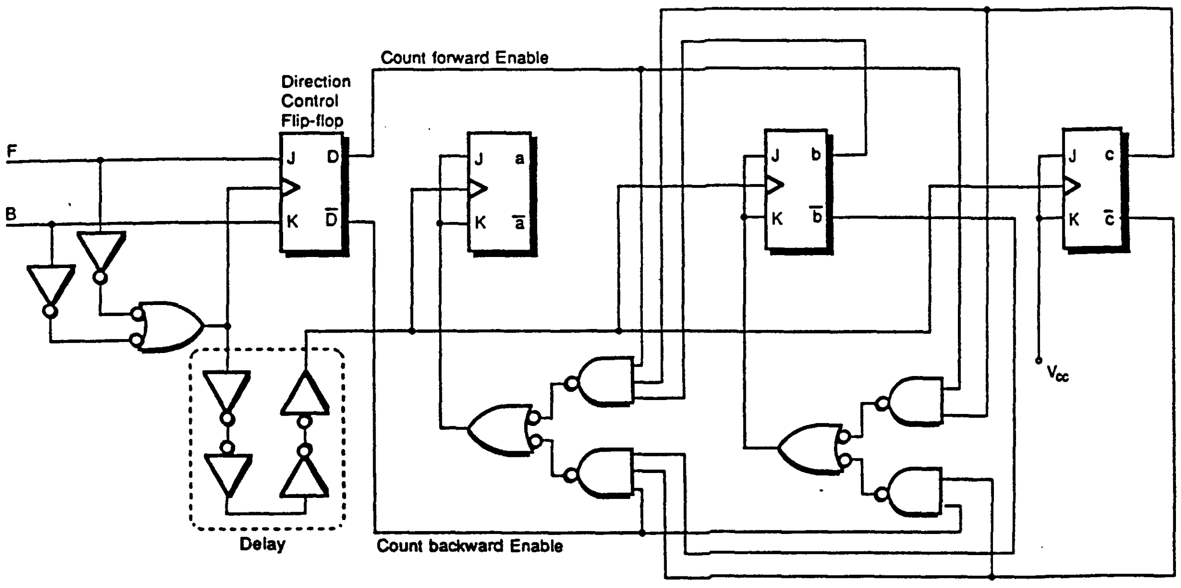


Figure 6-20 Forward-Backward three-bit synchronous binary counter

Although this exercise is primarily for demonstrating the derivation of the memory input equations for a forward/backward binary counter, there are some practical considerations not evident in the logic. The forward pulse (F) or the backward pulse (B) must set the direction control flip-flop D and then be counted (either up or down) by the memories a, b, and c. There are two gate delays between an input pulse (F or B) and the time the D memory is clocked. The D memory then has a transfer time to assume its proper state to enable either the forward or backward count. There are two additional gate delay times before the J-K inputs of memories $\underline{a}$ and $\underline{b}$ are set-up. The arrival of the clock input to memories $\underline{a}$, $\underline{b}$, and $\underline{c}$ must therefore be delayed until the J-K inputs are set-up. This delay must be greater than the sum of the memory transfer time plus two gate delays. The minimum period of the input pulses (F or B) must be greater than four gate delays plus the memory (D) transfer time. The serial inverters (labeled "delay") must provide this delay of the clock input to memories a, b, and c.

<u>A Four-Bit SYNCHRONOUS Binary Counter with Ripple Carries</u> - The input equations for this counter are identical to those given in equation set (6-40). The difference in the logic diagram of Fig. 6-14 and the logic diagram of Fig. 6-21 rests in the manner in which the carry is transferred between stages. Rewriting the equation set (6-40)

$$J_a=K_a=bcd \qquad J_b=K_b=cd \qquad J_c=K_c=d \qquad J_d=K_d=1 \qquad (6-40)$$

we may rewrite the equations for J_a and K_a as

$$J_a = K_a = J_b \cdot b \tag{6-46}$$

Equation (6-46) implies that we first generate the J_b input and then "ripple" it to the J_a input by "anding" it (J_b) with b. On a long binary string, rippling the carry between memories results in a considerable savings in gated inputs. The disadvantage of rippling is the limitation placed on the frequency of the input signal. Figure 6-21 illustrates the logical structure of a four-bit synchronous counter with ripple carry. This figure should be compared with Fig. 6-14.

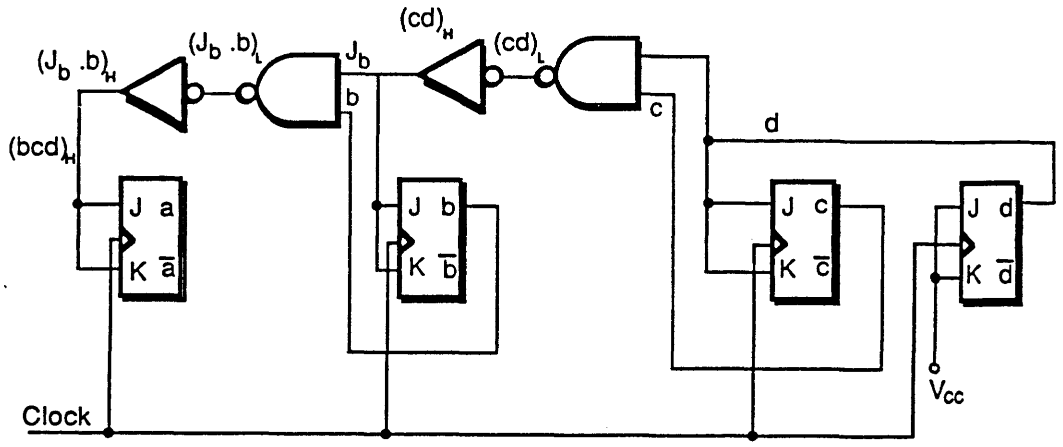


Figure 6-21 Four-bit asynchronous binary counter with ripple carry