5-1974

Asynchronous Logic Design with Flip-Flop Constraints

David Franklin Cox
Utah State University

ASYNCHRONOUS LOGIC DESIGN WITH FLIP-FLOP CONSTRAINTS

by

David Franklin Cox

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

1974

ABSTRACT

Asynchronous Logic Design with Flip-Flop Constraints

by

David Franklin Cox, Doctor of Philosophy

Utah State University, 1974

Major Professor: Ronald L. Thurgood, Ph.D.

Department: Electrical Engineering

Some techniques are presented to permit the implementation of asynchronous sequential circuits using standard flip-flops. An algorithm is presented for the RS flip-flop, and it is shown that any flow table may be realized using the algorithm (the flow table is assumed to be realizable using standard logic gates). The approach is shown to be directly applicable to synchronous circuits, and transition flip-flops (JK, D, and T) are analyzed using the ideas developed. Constraints are derived for the flow tables to meet to be realizable using transition flip-flops in asynchronous situations, and upper and lower bounds on the number of transition flip-flops required to implement a given flow table are stated.

(121 pages)

ASYNCHRONOUS LOGIC DESIGN WITH FLIP-FLOP CONSTRAINTS

by

David Franklin Cox

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

1974

ACKNOWLEDGMENTS

With sincere appreciation, I would like to thank Dr. Ronald Thurgood for his encouragement and help, and for being both advisor and friend during the preparation of this thesis.

I would also like to thank Dr. Alvin Despain for his inspiring discussions of pertinent points and critical review of the final paper. A special note of thanks go to Dr. Glen Smerage, Dr. William Jones, and Professor William Fletcher for their support. I am very grateful to Dr. Michael Windham of the Math department for taking the time to serve on my committee, and for providing a sympathetic ear in times of need.

Finally, to my wife, Chris, for her patience in typing the rough and final drafts, and to my children, Shaun and Kelli, I extend a husband's and father's gratitude for the sacrifices they have made.

David Franklin Cox

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

ABSTRACT

Asynchronous Logic Design with Flip-Flop Constraints

by

David Franklin Cox, Doctor of Philosophy

Utah State University, 1974

Major Professor: Ronald L. Thurgood, Ph.D.

Department: Electrical Engineering

Some techniques are presented to permit the implementation of asynchronous sequential circuits using standard flip-flops. An algorithm is presented for the RS flip-flop, and it is shown that any flow table may be realized using the algorithm (the flow table is assumed to be realizable using standard logic gates). The approach is shown to be directly applicable to synchronous circuits, and transition flip-flops (JK, D, and T) are analyzed using the ideas developed. Constraints are derived for the flow tables to meet to be realizable using transition flip-flops in asynchronous situations, and upper and lower bounds on the number of transition flip-flops required to implement a given flow table are stated.

(121 pages)

CHAPTER I

INTRODUCTION

Combinational and sequential systems

The realm of basic logic design is usually broken into two major divisions. The first is normally called "combinational" logic design and consists of a set of input variables $I_1, I_2, \ldots I_n$, a set of output variables $O_1, O_2, \ldots O_m$, and a mapping f from the input variables to the output variables wherein the combinational design objective is to physically implement a logic device that is isomorphic to the mapping function f relating two sets of physically realizable logic variables. i.e.

(a) $I = \{I_i | i = 1, 2, \ldots n\}$,

(b) $O = \{O_j | j = 1, 2, \ldots m\}$,

(c) $f : I \rightarrow O$.

At any instant of time for a given input I_k, a unique output O_q can be accurately predicted.

The second major division of logic design is sequential logic design. Sequential circuits have inputs and outputs, but also require positing internal "states" to adequately describe them. Briefly, a sequential circuit (or machine) can be described as having the following properties (Hartmanis and Stearns, 1966):

1. A finite set of inputs that may be applied to the circuit in a sequential order.

2. A finite set of internal states in which the circuit may be in.

3. The next state of the circuit is determined by the present circuit state and the present input.

4. A finite set of outputs that are determined by the combination of circuit state and input.

The above properties can be concisely stated as follows:

(a) S is a finite nonempty set of states;

(b) I is a finite nonempty set of inputs;

(c) O is a finite nonempty set of outputs;

(d) $N : S \times I \rightarrow S$ is called the next state function;

(e) $Z : S \times I \rightarrow O$ is called the output function.

It should be noted that the output function involves both the input and the state of the circuit. Sequential circuits that have this property are called "Mealy" circuits. If the output is a function of the circuit state only, the circuit is called a "Moore" circuit.

Synchronous and asynchronous sequential systems

Sequential circuits can be subdivided into what are called "synchronous" and "asynchronous" sequential circuits. Synchronous sequential circuits are sequential circuits whose events are constrained to occur only at specified instances of time, and are usually controlled by a device that provides "clock" pulses. The clock pulses need not be evenly spaced in time but must be spaced far enough apart so that all transient circuit action has vanished at the time the next clock pulse occurs.

Asynchronous sequential circuits do not have a "clocking" constraint on them and their circuit action commenses from the time an input changes (assuming the circuit has started in a stable state) and stops only when it can achieve a state coexistent with the new input (known as a "stable" state). In this report some current methods used in asynchronous logic design will be outlined, and an algorithm will be presented wherein the asynchronous design can be implemented using standard set-reset flip-flops. A less formal method will be given for the transition-type flip-flops, and an application to synchronous sequential circuits will be outlined.

CHAPTER II

CLASSICAL ASYNCHRONOUS LOGIC DESIGN

Asynchronous problem formulation

D. A. Huffman presented a method for designing an asynchronous circuit from problem specifications in 1954. His method basically consists of reducing the problem statement to a flow table from which the state function, N, can be derived after some flow table manipulation (Huffman, 1954).

It is assumed that the inputs are known, and a certain sequence of inputs is to give a specified output sequence. The flow table is then constructed from the problem statement in the following form:

1. The columns of the flow table are unique to a given input

2. The rows of the flow table are unique to a given state.

3. If the intersection of the input column and state row is to indicate a stable condition in the circuit, then the entry in that position is given the label of the row that it is in, and circled to indicate that it is stable.

4. If a column-row intersection is unstable, the entry in that position is labeled according to the row it is to move to (the new state).

5. If a "don't care" condition arises, or an impossible

input condition is evident, then a dash is entered into the column-row intersection.

It should be clear that the entries in the flow table merely indicate the next state the circuit is to move into from a given input/state condition; i.e. the flow table is a tabular representation of the "next state" function N. See Figure 1 for the flow table form.

states \ inputs	I_1	I_2	...	I_n	/ O
S_1	$\textcircled{S_1}$	S_3	...	S_2	Z_1
S_2				$\textcircled{S_2}$	Z_2
S_3		$\textcircled{S_3}$	...	S_m	Z_3
.		.		.	.
.		.		.	.
.		.		.	.
S_m		.		$\textcircled{S_m}$	Z_m

Figure 1. Flow table form.

If the flow table is constructed such that there is only one stable state per row, the table is called a primitive flow table. The output can be listed alongside each state-entry in the table, or it can be listed along the right side of the table as shown in Figure 1 if the table is primitive. If the outputs for transient states are known, they can be put in with the states shown. Usually these transient state outputs are left until later, however.

An example (from Unger, 1969) will now be given to illustrate the procedure. A sequential circuit is to have two inputs, x_1 and x_2, and one output, Z. Z is to go on when x_1 goes on only if x_2 was on during the previous "off" time of x_1. x_2 is allowed to come on and go off during the "off" time of x_1 as well as come on and stay on until after x_1 goes on. Z is to go off when x_1 goes off and wait for the sequence of events to reoccur.

The primitive flow table is constructed from the statement of the problem by assuming an initial input and output condition. It will be convenient here to start with inputs and output set to zero. Figure 2 shows the completed primitive flow table. We will assume that simultaneous input changes do not occur (which simplifies Unger's problem). When only single input changes occur the operation is known as "normal mode", and when the inputs are allowed to change only when the system is in a stable state, the operation is "fundamental mode". Hence, Figure 2 is a normal fundamental mode primitive flow table. Dashes are placed in forbidden input transition spaces to indicate the "don't care" condition. A state diagram is shown in Figure 3, and is equivalent to the flow table but may provide a more intuitive interface between the word statement of the problem and the flow table.

state \ x_1x_2	00	01	11	10	Z
1	①	3	-	2	0
2	1	-	7	②	0
3	4	③	5	-	0
4	④	3	-	6	0
5	-	3	⑤	6	1
6	1	-	5	⑥	1
7	-	3	⑦	8	0
8	4	-	7	⑧	0

Figure 2. Normal fundamental mode primitive flow table.

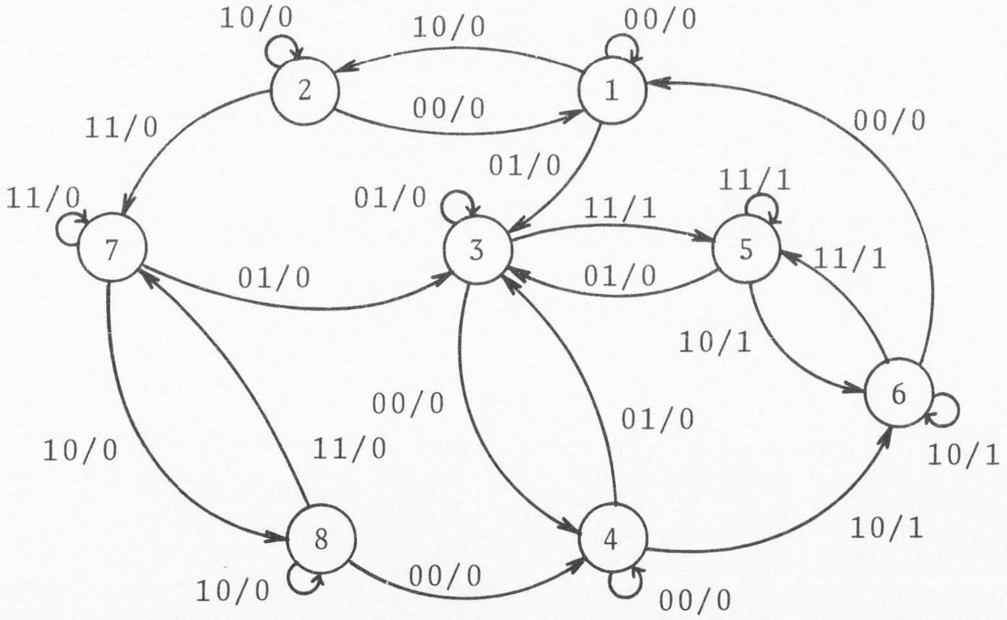


Figure 3. State diagram.

State reduction

The next step in the synthesis procedure is to try to reduce the number of states. This will generally (but not always) reduce the gate-count of the final circuit. The method, bacically, is to compare pairs of rows and see if any combination of input changes will lead to the same state. If such is the case, then the two compared states are called "equivalent" and further comparisons can be made to see if all input changes take the rows to the same or equivalent states. The process is continued until all the states (rows) have been placed into equivalence classes. This effectively partitions the set of states into non-intersecting sets, the union of which consists of the initial primitive set of states. Each equivalence class will then represent one state in the final circuit.

The logical behavior of the sequential circuit will not be modified by eliminating redundant states. The method of Paull and Unger (1959) will be used (an alternate method is given in Appendix A). The state reduction consists of eliminating all incompatible pairs of states. To define "compatible" we have to first define "cover".

Definition: A state S_a of a flow table A is said to cover state S_b of flow table B if, for any finite input sequence, the output from A when started in S_a is identical to the output from B when started in S_b whenever the B output is specified.

Definition: Two states of a flow table are compatible if they are both covered by some row of a flow table A. A set of states of a flow table that are covered by a single state of some flow table is called a compatible set, or a compatible.

A convenient way to check for incompatibles is by constructing a pair chart as shown in Figure 4. If two states that are assumed to be equivalent have different next state entries in one (or more) of their columns then the compatibility of the first two states implies the compatibility of the two different state entries in that column.

The states are now compared pair-wise and if an output incompatibility is noted, an "x" is placed in the pair intersection square. If no output incompatibility exists, then implied pairs are written in the square. If neither of the above occur, then the square is left blank.

	1	2	3	4	5	6	7
2							
3	14 x x	14 x 57 x					
4	26 x x	14 x 26 x					
5	x x	x x	x x	x x			
6	x x	x x	x x	x x			
7	28 x x	28 x x	57 x x	68 x x	68 x x	57 x 68 x	
8	14 x 28 x	14 x x	57 x x	68 x x	57 x 68 x	14 x 57 x	

Figure 4. Pair chart for flow table of Figure 2.

The chart is then filled in as follows: (Note that states 5 and 6 have outputs of "1" and the other states all have outputs of "0"). All combinations that have incompatible outputs have an "x" placed in them. Pairs that imply other pairs have the other pairs written in the box. After all pairs are entered, a check for incompatibles is made. A start is made from one of the output incompatible squares. Assume we start from square 15. Since it is output incompatible, we place an "x" in the square. We check all other squares for entries of "15" and if we find one, an "x" is placed within the square in which it is found and the square is considered to be an incompatible. This process is continued until no more x's can be entered. When the search for the entry of incompatible pairs in other squares is completed, a second "x" is placed in the original incompatible pair's square (15) to indicate that the square need not be referenced again. When the table is complete, the compatibles can be grouped so as to minimize the total number of states. The rules for checking the pair chart are as follows:

1. Start the compatible list (c-list) with the compatible pairs in the first column from the right having at least one entry without an x in it.

2. Move to the left, column by column. Let S_i be the set consisting of all non-x entries in column i. Take the intersection of S_i with current c-list. If the

intersection has more than one member, add the set consisting of i appended to the intersection. Delete duplicate entries and entries that are contained in other entries. Add pairs consisting of i and any members of S_i that did not appear in any of the intersections.

3. The final c-list plus those states not yet included in the c-list comprise the final groupings of states, and are called maximum compatibles.

The pair chart of Figure 4 yields the following:

Start:	$c = \{78\}$
$S_6 = \phi$	$c = \{78\}$
$S_5 = 6$	$c = \{78, 56\}$
$S_4 = \phi$	$c = \{78, 56\}$
$S_3 = 4$	$c = \{78, 56, 34\}$
$S_2 = \phi$	$c = \{78, 56, 34\}$
$S_1 = 2$	$c = \{78, 56, 34, 12\}$

All states are included in the final c-list, hence the final c-list is

$$c = \{78, 56, 34, 12\}.$$

Now we choose compatibles from the c-list such that every row of the original table is covered by at least one of the compatibles, and the set is "closed" in the sense that any set of rows implied by any compatible in the set is a subset of at least one compatible of the set. It is obvious that each element of the c-list is required to completely cover the original table so we have to use each maximum compatible.

The final flow table then looks like Figure 5.

state \ x_1x_2	00	01	11	10	
a	a,0	b	d	a,0	12
b	b,0	b,0	c	c	34
c	a	b	c,1	c,1	56
d	b	b	d,0	d,0	78

Figure 5. Minimized flow table.

The states are listed as letters so as not to be confused with the original state numbers (which are given on the right). The output is listed for each stable state.

State code assignment

The next step in the synthesis procedure is the state assignment. Each state is assigned a binary valued code such that the internal state transitions are not dependent on the binary values of the state variables. If more than one state variable is required to change for a given transition, then inherently unequal delays in the circuit make it virtually impossible for the two states to change value simultaneously. Such a condition is called a race and will be discussed later in this report. In most cases, then, what is needed is essentially a gray-code assignment for the states. Adjacent states are coded so that only one state variable changes per transition (adjacent states are states connected with a transition arrow on the state diagram, if one were drawn).

Starting with the flow table of Figure 5, we can draw a state diagram for the four states and assign binary values to the states so that each transition involves only a change in one state variable. Since there are only four states, the minimum number of state variables required for a unique code is two. Figure 6 shows the state diagram for the flow table of Figure 5.

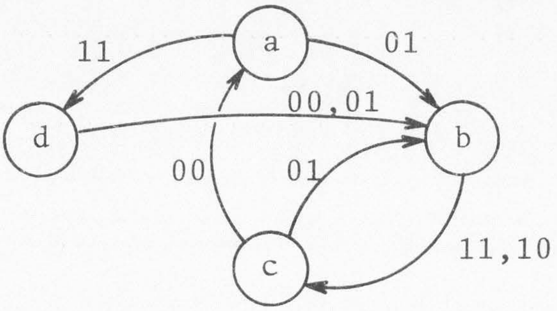


Figure 6. Minimized state diagram.

The state diagram of Figure 6 cannot be gray-coded with only two state variables as it's drawn. An extra state variable could be added to bring the total possible number of states to eight. This would give four extra states to use in which transitions could pass through one or more states before settling in the final state, and would give some flexibility to the state assignment. A Liu (1963) or Tracey (1966) assignment could also be made, but will not be commented on here.

An alternate method would be to note the availability of extra "b's" in column "01" of Figure 5. For an input of "01" every state has state "b" as its transition objective. It is possible in this case to have the transition pass through one or more of the other states before settling at state "b". Figure 7 shows the necessary adjacencies (those determined from columns other than "01"). The transition from "a" to "b" is seen to have been removed, and can be routed through state "d". The modified flow table is shown in Figure 8. Note that total state a-01 has been changed from "b" to "d". Since the output, Z, is constant for a given state, it is noted on the right of the flow table (it should be remembered that transition states do not, as yet, have specified outputs). Binary values for the states can now be assigned as follows:

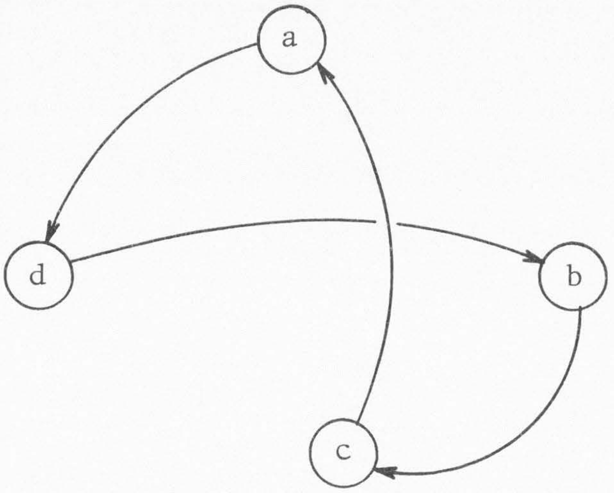


Figure 7. Necessary adjacencies.

a ←00; d ←01; b ←11; c ←10.

	x_1x_2				
state	00	01	11	10	Z
a	a	d	d	a	0
b	b	b	c	c	0
c	a	b	c	c	1
d	b	b	d	d	0

Figure 8. Flow table with "transition" state.

The state assignment can be modified (permuted)--the important point is that each entry must be adjacent to the one above and below it ("a" and "c" are adjacent).

The flow matrix (binary values assigned to the states) can now be drawn as shown in Figure 9. The state values are placed in "Karnaugh" order to facilitate the construction of the state maps.

	x_1x_2				
y_1y_2	00	01	11	10	Z
00	00	01	01	00	0
01	11	11	01	01	0
11	11	11	10	10	0
10	00	11	10	10	1

Figure 9. Flow matrix.

State and output maps

The state variable maps are simply maps taken from the flow matrix with one entry-variable shown. The maps are shown in Figures 10 and 11.

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	1

Y_1

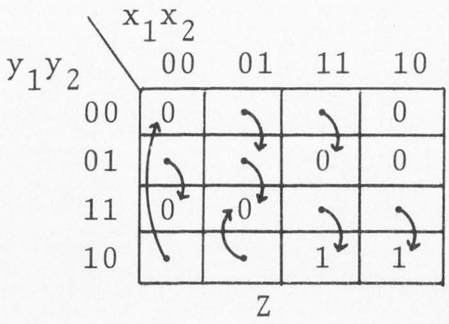
Figure 10. Karnaugh map for Y_1.

y_1y_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	0	0
10	0	1	0	0

Y_2

Figure 11. Karnaugh map for Y_2.

The output map, or Z-map, is taken from the flow matrix with entries placed only in the stable state positions. It is shown in Figure 12. Transition arrows are indicated to show beginning and end points for the transitions. Since



y_1y_2 \ x_1x_2	00	01	11	10
00	0			0
01			0	0
11	0	0		
10			1	1

Z

Figure 12. Partial Z-map.

the output (Z) is the same for indicated entries on a given row, only verticle transitions need to be indicated.

The transition entries can be derived as follows:

1. A "0" initial to "0" final entry should have a "0" in the transition position(s).

2. A "0" initial to "1" final entry (or vice-versa) may have a "0", "1", or "-" ("don't care") entry in the transition position(s). Usually a "-" is preferred for later flexibility in deriving the boolean expressions.

3. A "1" initial to "1" final entry should have a "1" in the transition position(s).

The reason for the above constraints is to prevent unnecessary "glitches" on the output. The complete Z-map is shown in Figure 13.

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	-	-
10	-	-	1	1

Z

Figure 13. Complete Z-map.

From the above maps the expressions for the states and output can be written. The next chapter describes subtleties to be aware of when going from the map to the boolean expressions (and hence to the circuit realization).

CHAPTER III

HAZARDS AND RACES

Logic delays

Inherent in any physical implementation of a digital circuit are logic delays. In combinational circuits these delays may not bother the circuit function but in sequential circuits random delays play funny tricks.

It was mentioned earlier that races should be avoided in asynchronous sequential design because critical races may lead to improper circuit action. An example of this is shown in Figure 14. If the input changes from $x_1x_2y_1y_2$ = 0111 to $x_1x_2y_1y_2$ = 0011 the state is unstable and tries to go to $x_1x_2y_1y_2$ = 0000. Since both state variables are unstable (y_1 and y_2) they will both try to change to "0" simultaneously. The probability that y_1 and y_2 will both switch simultaneously is virtually zero, and hence the circuit may end up in state 01 or 10, depending on which state variable switched first. The proper state assignment is one way of alleviating the critical race problem, and cycling through other states going to the same final state is another way.

A more subtle problem in sequential circuits due to random delays is the existence of combinational hazards. Each state variable is essentially generated through

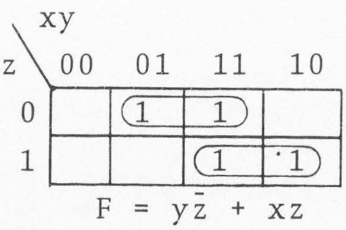
y_1y_2 \ x_1x_2	00	01	...
00	00	-	
01	01	-	
11	00	11	
10	10	-	

Figure 14. Critical race condition.

combinational logic, and any stray hazard pulses may cause the circuit to jump into the wrong state.

A "0" static hazard occurs when a single input change takes the output function from a "0" to a "0" and a momentary "1" may occur on the output at the time of the input change. A "1" static hazard is similar except for output polarity.

The reason for a hazard existing is illustrated in Figure 15. If adjacent implicants are not covered by a common cover, then the possibility of a hazard pulse occurring is present.



z \ xy	00	01	11	10
0		1	1	
1			1	1

$F = y\bar{z} + xz$

Figure 15. Hazard situation.

If the input goes from xyz = 110 to xyz = 111, the map function F can be expressed as $xy\bar{z} + xyz$, x = y = 1.

Note that only one variable is changing (due to the fact that only adjacent changes are assumed). With the variables assuming their respective values we have

$$F = xy(z + \bar{z})$$
$$= z + \bar{z}.$$

The common boolean reduction for the above situation is "1", but if z is generated from $\bar{z}$ through an inverter, there will be some delay after $\bar{z}$ switches from "1" to "0" before z switches from "0" to "1". This will give rise to a static "1" hazard as shown in Figure 16.

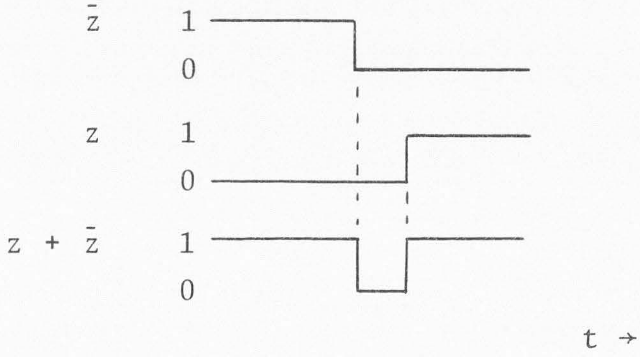


Figure 16. Static "1" hazard for change in z.

A dynamic hazard occurs when a single variable input change causes the output to change three times when normally only a single output change would be expected. The cause of the dynamic hazard is due to different signal paths reaching the output with different delays. A thorough treatment of hazards can be found in Unger (1969).

A fix for the static hazards is to include a common cover for all adjacent implicants. This would add the term "xy" for the example of Figure 15. The function would then be

$$F = y\bar{z} + xz + xy.$$

For the stated transition it would reduce to

$$\begin{aligned} F &= \bar{z} + z + 1 \\ &= 1. \end{aligned}$$

Hence no "0" pulse would occur. If all static hazards are fixed in this manner then the single-input-change dynamic hazards will have been corrected also (Unger 1969).

Multiple input change problems

For multiple input changes higher order hazards may occur with the output oscillating on and off two or more times before settling down. A hazard of this type, called a functional hazard, may occur due to the function implemented and not from the method of forming the combinational logic. An example of this is shown in Figure 17. An input change from xy = 00 to xy = 11 should cause the

y \ x	0	1
0	1	0
1	0	1

$$F = \bar{x}\bar{y} + xy$$

Figure 17. Functional hazard situation.

output, F, to remain at "1". However, due to unequal internal delays in the logic (the "AND" gate realizing

$\bar{x}\bar{y}$ may have a smaller delay time than the "AND" gate realizing xy) F would in reality put out a small "0" pulse. There is no known way to inhibit multiple-input-change hazards.

Essential hazards

Up to now, only combinational hazards due to inputs have been mentioned. Asynchronous sequential circuits have special problems due to arbitrary delays inherent in the feedback, or state, variables in conjunction with the delays in the input variables. Figure 18 shows a possible hazard situation with a single-input-change

	a	b	
1	①	2	
2	3	②	n>2
3	③	n	

Figure 18. Essential hazard condition.

constraint (only the two columns with the changing input are shown). The flow table has two input columns and three rows shown. If the initial state is assumed to be 1-a, the position is stable. If the input is then changed to "b", the circuit that realizes the flow table may have some of its state variables see changes in other state variables before the first state variables see the input change. In this case the initial movement on the table will be vertical instead of horizontal. For the input change in Figure 18 the initial transition may be from 1-a to 2-a instead of 1-a to 1-b. From 2-a the circuit is taken to 3-a and thence to 3-b when the input change is finally sensed. From there it goes to n-b. This is not the 2-b destination originally designed for. The above aberration is called an "essential hazard" and can be checked for in a flow table in the following manner: For some initial total state and an input variable x, if three changes in x take the system to a state different from the state that only one change in x takes it to (assuming both sequences start in the same total state), then an essential hazard exists. The only solution for an essential hazard is to insert delay elements in the state variable feedback paths to insure that input changes are seen by the system before the state variable changes are sensed.

A related hazard condition called a "nonessential hazard" (or d-trio) is shown in Figure 19. Starting in total state 1-a, three consecutive input changes of the same variable (only one shown in Figure 19) take the system to the same total state that only one input change would,

	a	b
1	①	2
2	3	②
3	③	2

Figure 19. Nonessential hazard condition.

but it is possible to cycle through state 3 if the state variable changes were sensed first. The static behavior of the system would be correct, but a possible output transient may occur if the output for state 3-a is different from the outputs for 1-a and 2-b. Some of the transient problems for the output were mentioned in conjunction with the Z-map of Chapter II.

Synthesis of example problem

With the above race and hazard conditions in mind we can review the example problem of Chapter II and specify the circuit realization. The flow table is redrawn in Figure 20. (The flow table is redrawn from Figure 9 with the states numbered for easy reference).

state \ x_1x_2	00	01	11	10	Z
1	①	2	2	①	0
2	3	3	②	②	0
3	③	③	4	4	0
4	1	3	④	④	1

Figure 20. Flow table derived from Figure 9.

A check for essential hazards shows that the condition exists for the 1-00 to 1-11 transition, the 2-11 to 2-00 transition, the 2-10 to 2-00 transition, the 3-00 to 3-11 transition, the 3-00 to 3-10 transition, and the 4-11 to 4-00 transition. There are two single-input-change essential hazards (2-10 to 2-00 and 3-00 to 3-10) so care must be taken to insure that the state variable changes are sensed after the input change. This may necessitate the addition of delays in the feedback paths.

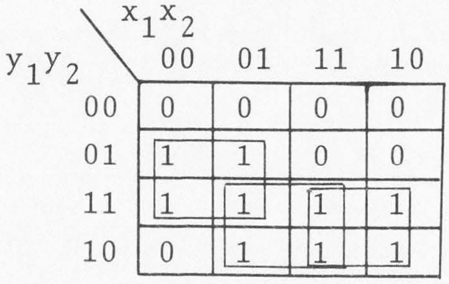
Boolean expressions can be derived from the Y-maps and Z-map as indicated in Figure 21. Note that the Y-map adjacencies are covered. If hazard pulses occur on the state variable outputs, the system would think it was in an entirely different state and improper circuit action could ensue. The Z-map adjacencies (none in this case) may be covered if transient output pulses are not desired.

It might be noted that a simpler expression could be written for Y_2 by covering the "0's",

i.e. $Y_2 = (x_2 + y_2)(\bar{x}_1 + \bar{y}_1)$.

The above expression could also be derived by factoring the "1's" cover expression. The circuit realization is shown in Figure 22.

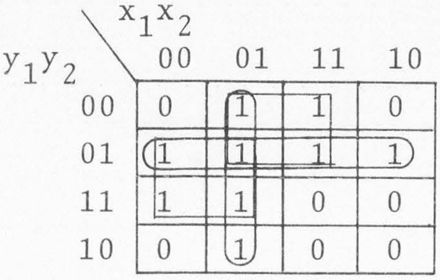
This completes the classical synthesis procedure. Enough of the procedure has been presented to give an idea of the methods and problems of asynchronous logic design, and the next four chapters will be concerned with flip-flops and their application to synchronous and asynchronous logic design.



y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	1

Y_1

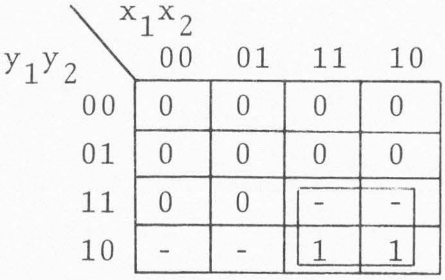
$$Y_1 = \bar{x}_1y_2 + x_2y_1 + x_1y_1$$



y_1y_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	0	0
10	0	1	0	0

Y_2

$$y_2 = x_2\bar{y}_1 + \bar{y}_1y_2 + \bar{x}_1y_2 + \bar{x}_1x_2$$



y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	-	-
10	-	-	1	1

Z

$$Z = x_1y_1$$

Figure 21. State and output maps for example of Chapter II.

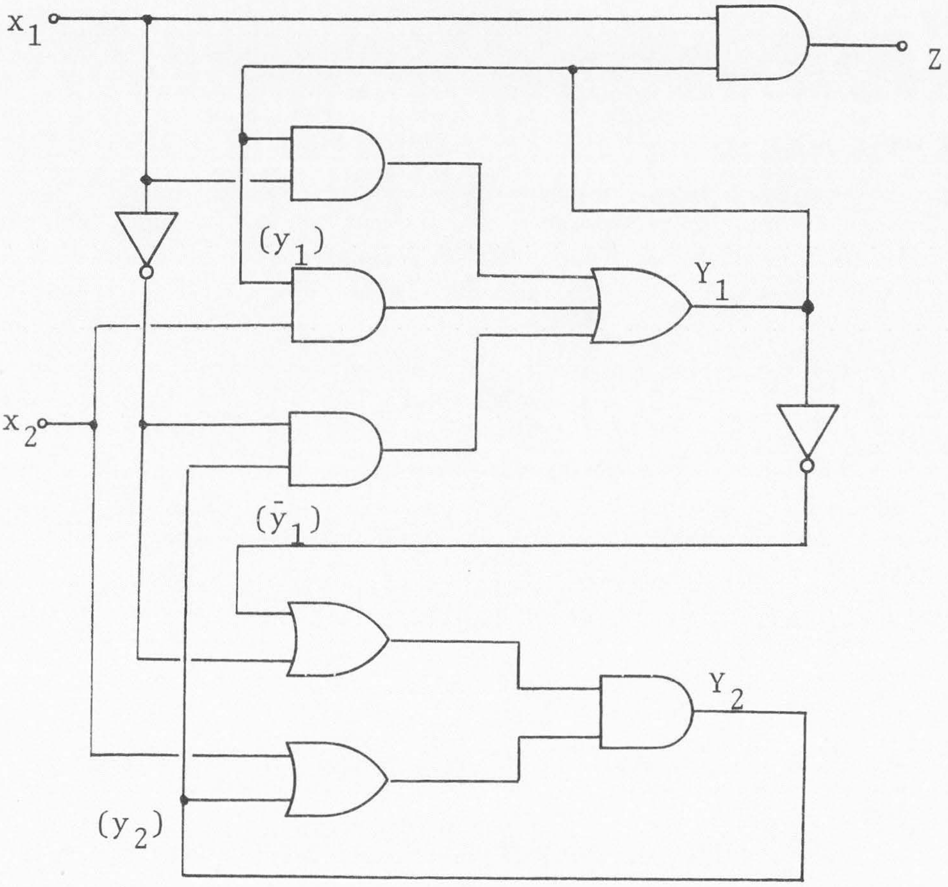


Figure 22. Circuit diagram for example problem.

CHAPTER IV

BISTABLE ASYNCHRONOUS SUBSYSTEMS

Flip-flop types

In synchronous sequential design, by far the majority of synthesis techniques involve clocked flip-flops. The four major types of flip-flops are the RS (Reset-Set), D (Data, or Delay), JK (it is unknown what J and K stand for), and T (Toggle). Sometimes the RS and T flip-flops are combined into one device with the R, S, and T inputs and called an RST flip-flop. The purpose of a flip-flop is to store a bit of information when a specified event occurs on one or more of the inputs (the bit could be a "0" or a "1"). The different types of flip-flops are categorized by the input sequencing required to store the required bit of information. The output of all flip-flops is labeled Q, and if its inverse is also available it is labeled $\bar{Q}$.

The RS flip-flop can be completely characterized through level logic inputs without positing internal states other than the output state. The D, JK, and T flip-flops are essentially transition flip-flops and require additional internal state variables to be accurately characterized using the Huffman method of asynchronous analysis. For this reason the present chapter will be concerned with the RS flip-flop only, and its application in the synthesis of asynchronous sequential logic circuits. Transition flip-flops will be treated in Chapters VI and VII.

RS flip-flop

The basic RS flip-flop has two inputs and one output. The inputs are labeled "Set" and "Reset", and the output is labeled "Q". Assume an initial state of "0" for Q, S, and R. If S and R remain at "0", then Q will also remain at "0". If R goes to "1" (with S remaining at "0"), Q will remain at "0". If S goes to "1" (with R remaining at "0") or oscillates between "0" and "1", Q will go to and remain at "1". With S set at "0", if R now goes to "1", then Q will go to "0" and remain there until R = 0 and S = 1 again (which repeats the process). If both S and R go to "1" at the same time, then the circuit action is undefined.

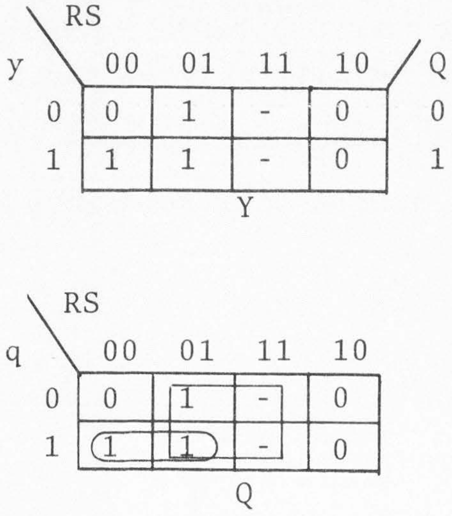
The RS flip-flop can now be described by a flow table as shown in Figure 23. Since the circuit operation was so

	RS				
state	00	01	11	10	Q
1	①	2	-	①	0
2	②	②	-	1	1

Figure 23. RS flip-flop flow table.

simple, the primitive flow table was bypassed and all entries were entered by inspection. Due to the fact that the output (Q) is different for each row, a merge (or simplification) of the rows is not possible. The flow table has a minimum number of rows as shown.

The flow matrix and state (Y) map are shown in Figure 24. Note that Q = y, so that the state could be labeled Q if desired. Since there is only one state, the flow matrix and Y-map are the same.



y \ RS	00	01	11	10	Q
0	0	1	-	0	0
1	1	1	-	0	1

q \ RS	00	01	11	10
0	0	1	-	0
1	1	1	-	0

Figure 24. RS flip-flop flow matrix and output map.

Characteristic equation

The "characteristic equation" for the RS flip-flop can be taken from the output map of Figure 24. If the constraint that R and S cannot be "1" simultaneously is added, we have

$$Q = S + \bar{R}q$$

$$SR = 0.$$

"Q" is the next state function defined by R, S, and q. "q" is the present state (or output) of the flip-flop and equals Q for static conditions. If an input change causes an output transition then q will eventually assume the value Q defined by the characteristic equation. "q" may be thought of as being the system state variable. The second equation (SR = 0) is a constraint to be met by the designer and not a functional relationship of the flip-flop.

Logic realization of RS flip-flop

A possible circuit diagram for the RS flip-flop is shown in Figure 25. A more common configuration is shown in Figure 26. The evolution from Figure 25 should be evident.

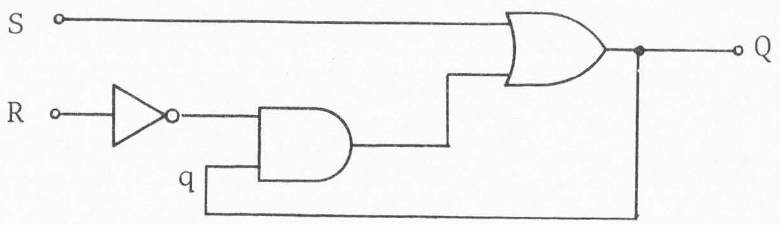


Figure 25. RS flip-flop circuit diagram.

An inverter may be placed on the output if Q is desired, or it may be noted that the point labeled "Q" in Figure 26 is actually the inverse of $\bar{Q}$ for all acceptable inputs.

However the RS flip-flop is realized physically, the diagrammatical symbol used for it in this report will be as shown in Figure 26.

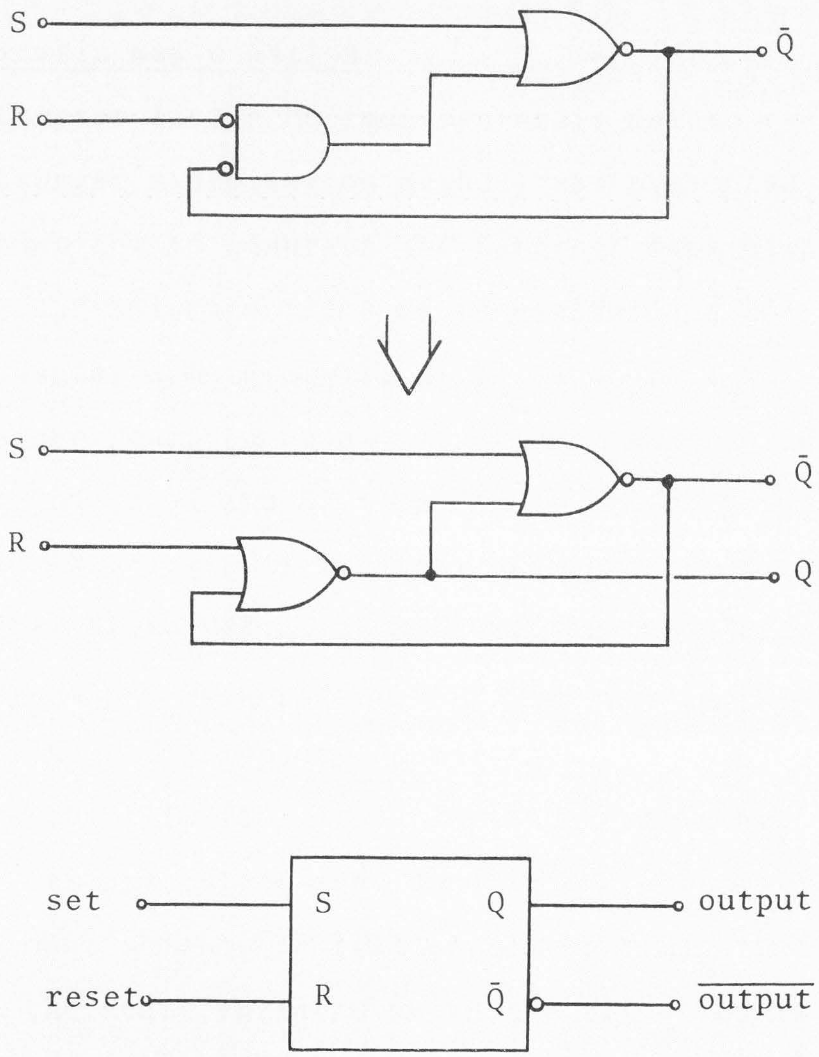


Figure 26. RS NOR flip-flop and symbol.

CHAPTER V

AN ALGORITHM FOR DESIGNING ASYNCHRONOUS SEQUENTIAL CIRCUITS USING RS FLIP-FLOPS

Characteristic map equation

In Chapter II the Huffman synthesis method (with the Paull-Unger minimization method) was presented and it was shown how to generate the Karnaugh maps defining the state variables in terms of themselves and the inputs. In this chapter a method will be given wherein the state variables are realized by RS flip-flops, and the Set and Reset inputs of the RS flip-flops will be functions of the flip-flop outputs (states) and the system inputs. The original state variable maps will be the starting point from which the maps defining the Set and Reset inputs to the flip-flops are generated.

Since the flip-flop is to represent the state variable, its next state function (or characteristic equation) must somehow be related to the Y-map that generates the state variable which the flip-flop is to replace. If we think of the variables of the characteristic equation as functions of other variables, then it is an easy step to go to maps to represent those functions. The characteristic equation is

$$Q_i = S_i + \bar{R}_i q_i,$$

where i represents the $i^{\underline{th}}$ state variable. Q_i, S_i and $\bar{R}_i$ will be functions of the system inputs, I, and the flip-flop outputs (or state variables) Q_i. i.e.

$$Q_i : Q \times I \rightarrow Q$$

$$S_i : Q \times I \rightarrow S$$

$$R_i : Q \times I \rightarrow R.$$

q_i is one of the independent variables and hence is not a function of the other independent variables. Note that

$$I = \{x_1 x_2 \ldots x_n | x_i = 0,1; i = 1,2, \ldots n\}$$

$$Q = \{q_1 q_2 \ldots q_m | q_j = 0,1; j = 1,2, \ldots m\}.$$

Since the Karnaugh map is essentially a graphical representation of the functional relationship, we can replace the symbols in the characteristic equation with the appropriate Karnaugh map. The Boolean operators are valid if we assume that the operations take place between corresponding minterms of the maps, and the map variables are identical and in the same position on each map (Caldwell, 1958).

The example of Chapters II and III will be used to illustrate the method. The state and output maps of Figure 21 are reproduced in Figure 27 for reference. The state variables (y_1 and y_2) will be relabeled q_1 and q_2 with the next state variables (Y_1 and Y_2) changed to Q_1 and Q_2.

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	1

Y_1

y_1y_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	0	0
10	0	1	0	0

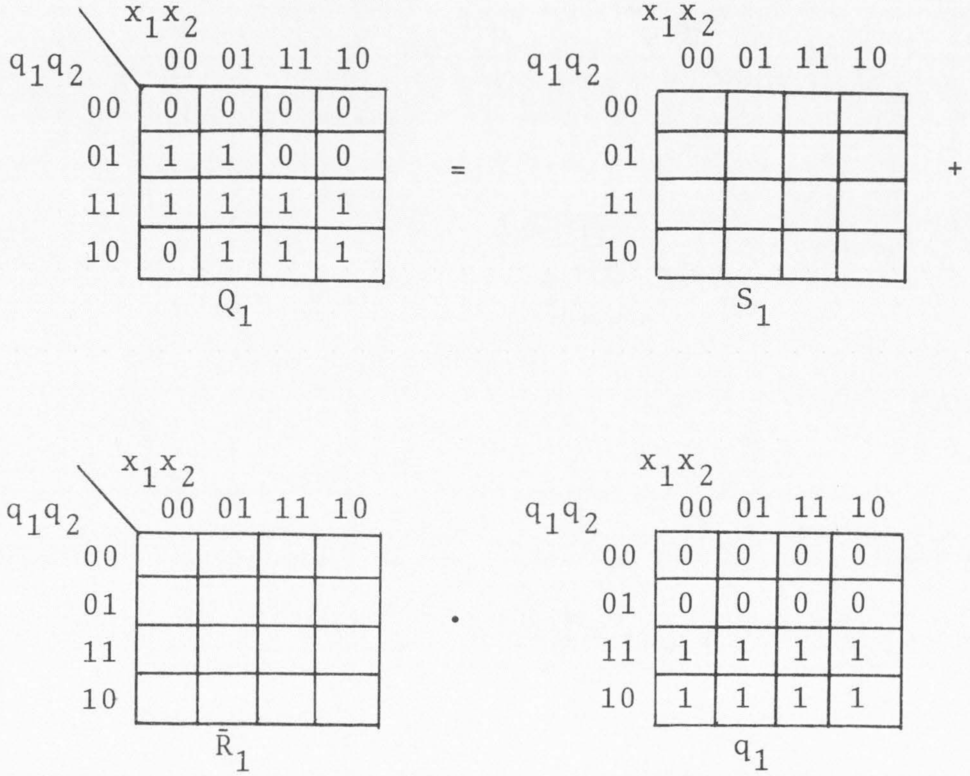
Y_2

y_1y_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	-	-
10	-	-	1	1

Z

Figure 27. State and output maps of Figure 21.

The characteristic equation is then written with maps replacing the functional symbology as shown in Figure 28. Each map must be a function of the same variables (x_1, x_2, q_1, q_2) as stated previously, and the variables must be in the same position on each map for the boolean operators between the maps to be valid.



q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	1

Q_1

=

q_1q_2 \ x_1x_2	00	01	11	10
00				
01				
11				
10				

S_1

+

q_1q_2 \ x_1x_2	00	01	11	10
00				
01				
11				
10				

$\bar{R}_1$

$\cdot$

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

q_1

Figure 28. Characteristic map equation for Q_1.

Selecting the minterm entries

The Q_1 map is identical to the Y_1 map derived from the classical synthesis, and the q_1 map simply shows that q_1 is identical to itself. This provides the starting point for filling in the S_1 and $\bar{R}_1$ maps. The procedure is outlined as follows:

1. Place "0's" in each minterm position of S_1 where a "0" occurs in the equivalent minterm of Q_1.

2. Place "0's" in each minterm position of $\bar{R}_1$ where $q_1 = 1$ occurs and where a "0" occurs in the equivalent minterm of Q_1.

3. Place a "-" ("don't care") in each minterm position of $\bar{R}_1$ where $q_1 = 0$ occurs and where a "0" occurs in the equivalent minterm of Q_1.

4. Place a "1" in each minterm position of S_1 where $q_1 = 0$ occurs and where a "1" occurs in the equivalent minterm position of Q_1.

5. Place a "1" in each minterm position of $\bar{R}_1$ where $q_1 = 0$ occurs and where a "1" occurs in the equivalent minterm position of Q_1.

6. Place a "-" in each minterm position of S_1 where $q_1 = 1$ occurs and where a "1" occurs in the equivalent minterm position of Q_1.

7. Place a "1" in each minterm position of $\bar{R}_1$ where $q_1 = 1$ occurs and where a "1" occurs in the equivalent minterm position of Q_1.

8. If a "-" occurs in the Q_1 map, then a "+" (a "don't care" with caution) is placed in the corresponding minterms of S_1 and $\bar{R}_1$. When the final coverings are placed to derive the expressions from the maps it should be noted that the choice of $S_1 = 1$ and $\bar{R}_1 = 0$ is not allowed. All other combinations are permitted (i.e. $S_1 = 1$, $\bar{R}_1 = 1$; $S_1 = 0$, $R_1 = 1$, etc.). Note that a "-" does not occur in Q_1 for this example.

The above procedure has accounted for the SR = 0 constraint, and the algorithm uniquely specifies the S and $\bar{R}$ map entries for a fully specified Q map. Figure 29 shows the completed S_1 and $\bar{R}_1$ maps (using the algorithm) of Figure 28.

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	1

Q_1

=

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	-	-	-	-
10	0	-	-	-

S_1

+

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	-	-	-	-
01	1	1	-	-
11	1	1	1	1
10	0	1	1	1

$\bar{R}_1$

·

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

q_1

Figure 29. Complete specification for Q_1 excitation.

A quick check could be made of the S and $\bar{R}$ maps to insure that the minterm positions of S that have a "1" or "-" entry correspond only to "1" entries in the respective minterm positions of $\bar{R}$.

The Set and Reset maps of Q_2 are similarly derived, and are shown in Figure 30. Expressions for S_1, R_1, S_2, and R_2 can now be written as shown in Figure 31. The R expressions are found from the "0" terms of the $\bar{R}$ maps.

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	0	0
10	0	1	0	0

Q_2

=

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	-	-	-	-
11	-	-	0	0
10	0	1	0	0

S_2

+

q_1q_2 \ x_1x_2	00	01	11	10
00	-	1	1	-
01	1	1	1	1
11	1	1	0	0
10	-	1	-	-

$\bar{R}_2$

·

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

q_2

Figure 30. Complete specification for Q_2 excitation.

The circuit diagram can be drawn once the excitation functions are derived, and is shown in Figure 32. Inputs are indicated on the left to simplify the circuit schematic. The output function is the same as derived in Chapter II (Figure 21) with q_1 substituted for y_1:

$$Z = x_1q_1.$$

The map procedure can be extended to cover any number of state variables, but the maps become unwieldy to work with for seven or more variables. The entries are specified for any number of variables, however, since the procedure given only deals with minterms and is not changed by the map size.

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	-	-	-	-
10	0	-	-	-

S_1

$S_1 = \bar{x}_1 q_2$

q_1q_2 \ x_1x_2	00	01	11	10
00	-	-	-	-
01	1	1	-	-
11	1	1	1	1
10	0	1	1	1

$\bar{R}_1$

$R_1 = \bar{x}_1 \bar{x}_2 \bar{q}_2$

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	1	0
01	-	-	-	-
11	-	-	0	0
10	0	1	0	0

S_2

$S_2 = \bar{x}_1 x_2 + x_2 \bar{q}_1$

q_1q_2 \ x_1x_2	00	01	11	10
00	-	1	1	-
01	1	1	1	1
11	1	1	0	0
10	-	1	-	-

$\bar{R}_2$

$R_2 = x_1 q_1$

Figure 31. Flip-flop excitation expressions.

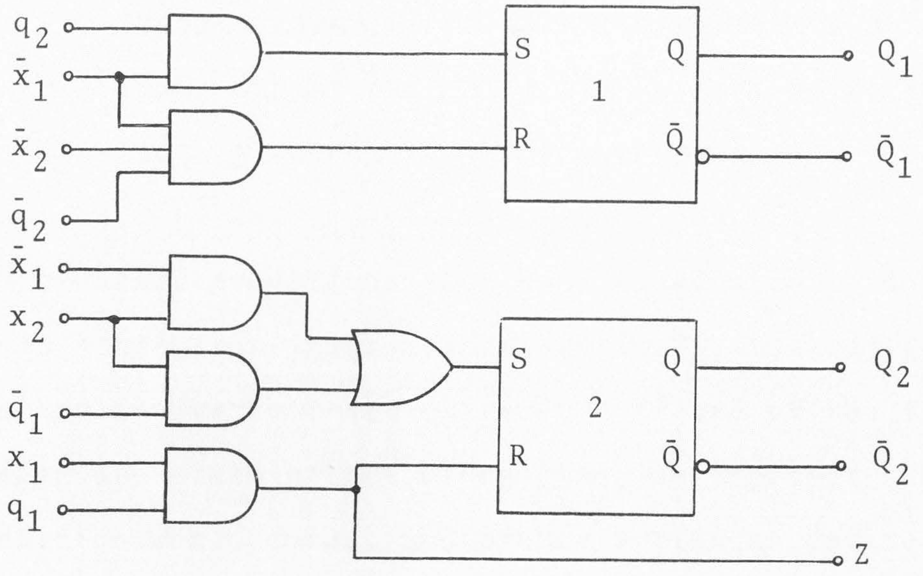


Figure 32. Sequential circuit with RS flip-flops.

Algorithm statement

Using the example given, an extension to the general case should be obvious. The algorithm is given for the set and reset maps of a given state variable. Reference need only be made to the original excitation maps, but the entire map equality can be drawn (as shown in Figures 28, 29, or 30) as a memory aid.

Given the excitation map for state variable Q_i, the S_i and R_i maps are derived as follows:

$Q_i = 0$, $q_i = 0$: $S_i = 0$; $R_i = -$.
$q_i = 1$: $S_i = 0$; $R_i = 1$.
$Q_i = 1$, $q_i = 0$: $S_i = 1$; $R_i = 1$.
$q_i = 1$: $S_i = -$; $R_i = 0$.
$Q_i = -$, all q_i: $S_i = +$; $R_i = +$.

The S_i and R_i maps are covered in the standard fashion with a caution on the "+". A "+" is considered equivalent to a "-" ("don't care"), but with the restriction that the S_i and R_i maps cannot have a "+" in corresponding minterm positions both grouped in a "1's" grouping (i.e., they both cannot be used as "1's" for a given S_i and R_i realization). Static hazard conditions should be eliminated to insure smooth state transitions since static "0" hazards could cause an inadvertent set or reset. If one of the flip-flops changed its state at the wrong time, an improper state transition would occur, and proper action as defined by the flow table could not be achieved in most cases.

CHAPTER VI

AN APPLICATION OF THE MAP-METHOD TO SYNCHRONOUS CIRCUITS

Problems in synchronous design

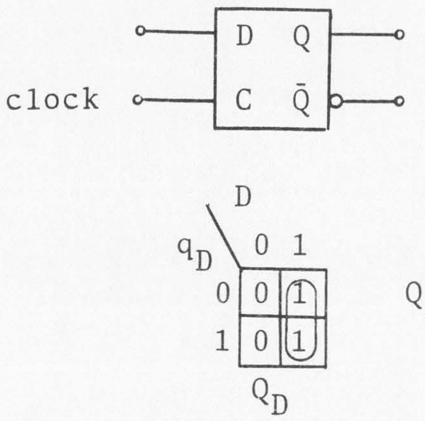
If the technique presented in Chapter V is tried for asynchronous design using T or D flip-flops, a problem arises (the unclocked JK flip-flop is excluded due to the improbability of simultaneous J and K transitions occuring, hence it reduces to an RS flip-flop). In describing the T and D flip-flops more than two states are required to adequately describe their total operation using logic levels as excitations. This means that a set of equations is required to describe the circuit action--hence more than one state variable is needed.

For the RS flip-flop only two states were required to describe the circuit operation, and hence only one state variable was needed. As it turned out the RS flip-flop output represented the state variable, and so was accessible. Where more than one state variable is required the relationship between them has to be maintained in the state assignment for the flow table to be achieved. Such is the case for only a small fraction of possible flow tables. This problem will be discussed in Chapter VII.

Synchronous analysis

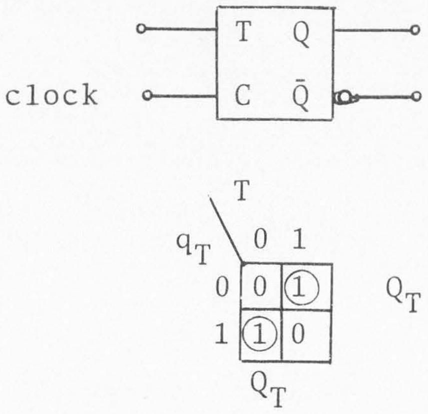
In synchronous sequential design the presence of a "clock" signal supresses the requirement for positing extra states for a complete description of the D, T, or JK flip-flops, and the method of Chapter V can be used effectively to produce maps for the inputs of the state flip-flops. Note that it is possible for the JK flip-flop to be used in synchronous design problems since the initiating signal transition occurs only on the "clock" input. The other flip-flop inputs are assumed to be at a definite logic level when the clock transition occurs (flip-flop action may occur on positive-going or negative-going clock transitions, depending on how the flip-flop is constructed. In this report excitation is assumed to occur on the positive-going edge of the clock).

The characteristic map, characteristic equation, and symbol are shown in Figures 33-36 for D, T, JK, and RST flip-flops. The RST flip-flop is included for completeness. Note that the clocked RS flip-flop would have the same characteristic equation and map-generation algorithm as described in Chapter V. In generating the RST characteristic map the technique of Marcus (1969) was used. Each minterm entry of the respective map indicates what the flip-flop output will be upon the occurrence of a positive-going transition on the clock input lead.



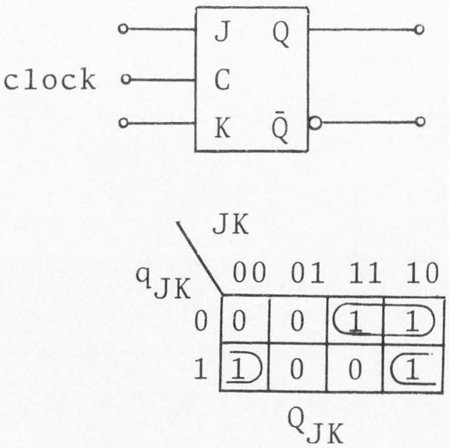
$$Q_D = D$$

Figure 33. D flip-flop symbol, characteristic map, and characteristic equation.



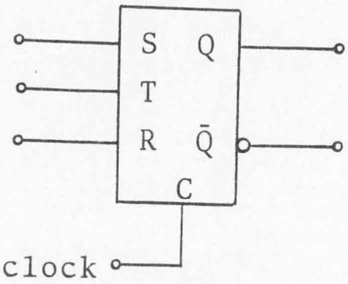
$$Q_T = T\bar{q}_T + \bar{T}q_T$$

Figure 34. T flip-flop symbol, characteristic map, and characteristic equation.



$$Q_{JK} = J\bar{q}_{JK} + \bar{K}q_{JK}$$

Figure 35. JK flip-flop symbol, characteristic map, and characteristic equation.

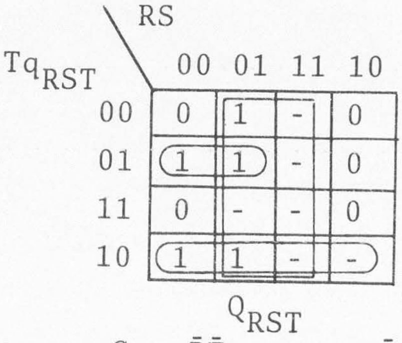


constraints:

$RS = 0$

$STq_{RST} = 0$

$RT\bar{q}_{RST} = 0$



Tq_{RST} \ RS	00	01	11	10
00	0	1	-	0
01	1	1	-	0
11	0	-	-	0
10	1	1	-	-

Q_{RST}

$$Q_{RST} = S + \bar{R}\bar{T}q_{RST} + T\bar{q}_{RST}$$

Figure 36. RST flip-flop symbol, characteristic map, characteristic equation, and constraint equations.

Only the RST flip-flop is seen to have constraints on its inputs. When realizing the R, S, and T input maps these constraints have to be accounted for.

The problem statement of a synchronous sequential circuit is similar to the asynchronous case, but the race and hazard problems need not be accounted for in the synchronous problem. A state diagram or flow table is constructed and a state reduction is attempted as in the asynchronous case. The state assignment can be taken as the minimum number of state variables to uniquely code each state. All destination state transitions are taken in one step since the clock essentially constrains the circuit activity to occur at discrete points in time, and all feedback paths (from flip-flop output to flip-flop input) are nonfunctional at points in time other than when the positive-going clock transition occurs. From the transition matrix (flow table with states coded) the state maps are drawn, and these are then used to generate the flip-flop input maps as determined from the characteristic equation.

Example

An example will be given to illustrate the procedure. The minimized flow table of Figure 5 is redrawn (with the states numbered) as shown in Figure 37. Since races are not a problem in synchronous design, cycles need not be introduced for proper circuit operation. All transitions are direct, as previously mentioned.

	x_1x_2				
state	00	01	11	10	Z
1	①	2	4	①	0
2	②	②	3	3	0
3	1	2	③	③	1
4	2	2	④	④	0

Figure 37. Flow table for synchronous design.

The states are assigned a binary code in a fashion (generally) to realize the output function (Z) in the simplest way. In this case (Figure 37) the output is coincident with state ③ only, so access will be required to both state variables (only two state variables are required to uniquely realize four states). The flow matrix is shown in Figure 38.

q_1q_2 \ x_1x_2	00	01	11	10
00	00	01	10	00
01	01	01	11	11
11	00	01	11	11
10	01	01	10	10

Figure 38. Flow matrix for synchronous problem.

The state maps can be drawn as shown in Figure 39. The state map is then used in the characteristic equation of whichever flip-flop (or combination thereof) is desired to implement the total system. If all flip-flops are available, each state map can be tried with all of the characteristic equations to see which one would give the cheapest realization in combinational logic.

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Q_1

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	0	0
01	1	1	1	1
11	0	1	1	1
10	1	1	0	0

Q_2

Figure 39. State maps for synchronous problem.

D realization

To realize the states with D flip-flops it is necessary to refer to the characteristic equation of the D flip-flop. Figure 40 shows the generation of the inputs for D_1 and D_2. It is seen that the input maps are simply the state maps. This may mean a lack of ability to generate "don't care" conditions from specified conditions of the state maps, which could provide a means of further simplification. However, the single input may make up for the lack of flexibility.

$$Q_D = D$$

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Q_1

=

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

D_1

$$D_1 = x_1(q_1 + q_2 + x_2)$$

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	0	0
01	1	1	1	1
11	0	1	1	1
10	1	1	0	0

Q_2

=

q_1q_2 \ x_1x_2	00	01	11	10
00	0	1	0	0
01	1	1	1	1
11	0	1	1	1
10	1	1	0	0

D_2

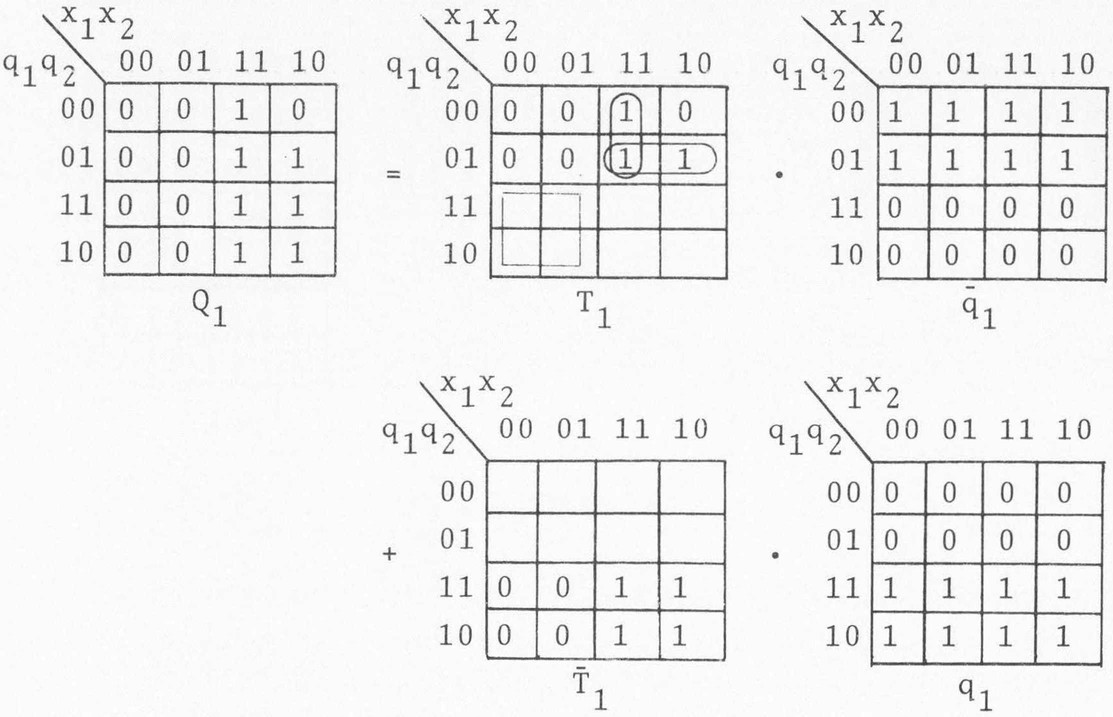
$$D_2 = \bar{x}_1x_2 + \bar{q}_1q_2 + q_1\bar{q}_2\bar{x}_1 + x_1q_2$$

Figure 40. Input excitation maps for realization with D flip-flops.

T realization

The realization using T flip-flops is shown in Figure 41. A fully specified state map generates a fully specified T-map as in the D flip-flop case. Only the generation of Q_1 is shown for brevity. It should be evident that the T-map is identical to the Q-map where q = 0, and is the inverse of Q where q = 1. T_1 and $\bar{T}_1$ are filled in only where they contribute to Q_1 to clarify the process.

$$Q_T = T\bar{q}_T + \bar{T}q_T$$



$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Q_1

=

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11				
10				

T_1

•

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

$\bar{q}_1$

+

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00				
01				
11	0	0	1	1
10	0	0	1	1

$\bar{T}_1$

•

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

q_1

$$T_1 = \bar{x}_1q_1 + x_1x_2\bar{q}_1 + x_1\bar{q}_1q_2$$

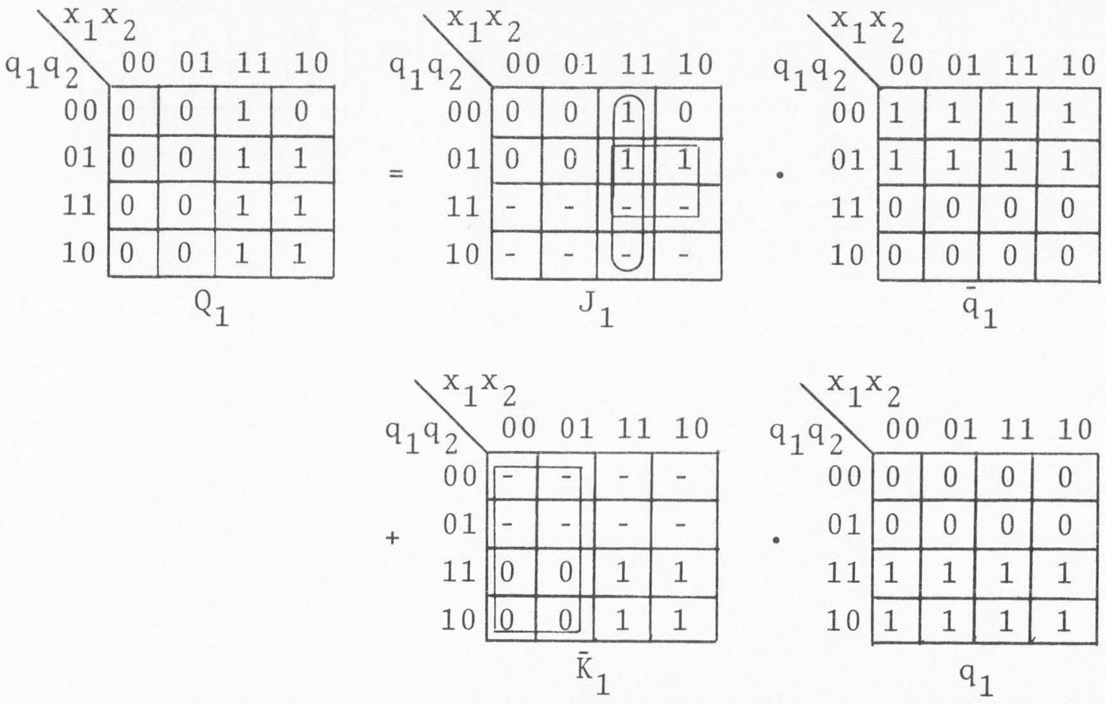
$$T_2 = \bar{x}_1x_2\bar{q}_2 + \bar{x}_1\bar{x}_2q_1$$

Figure 41. Realization expressions for T flip-flops.

JK realization

The JK realization expressions are given in Figure 42, along with the map equality for Q_1. The JK flip-flop is seen to be similar to the T flip-flop with the exception that half of the J and K maps consist of don't cares, which may eliminate some of the combinational logic altogether (as in the generation of K_1).

$$Q_{JK} = J\bar{q}_{JK} + \bar{K}q_{JK}$$



$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Q_1

=

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	-	-	-	-
10	-	-	-	-

J_1

•

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

$\bar{q}_1$

+

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	-	-	-	-
01	-	-	-	-
11	0	0	1	1
10	0	0	1	1

$\bar{K}_1$

•

$q_1q_2 \backslash x_1x_2$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

q_1

$J_1 = x_1x_2 + x_1q_2$

$K_1 = \bar{x}_1$

$J_2 = q_1x_1 + \bar{x}_1x_2$

$K_2 = \bar{x}_1\bar{x}_2q_1q_2$

Figure 42. Realization expressions for JK flip-flops.

RST realization

The RST flip-flop is the last flip-flop to be looked at in the synchronous sequential example. The realization expressions are given in Figure 43. Filling out the S, R, and T-maps is complicated by the fact that the three

$$Q_{RST} = S + \bar{R}\bar{T}q_{RST} + T\bar{q}_{RST}$$

$$RS = 0$$

$$STq_{RST} = 0$$

$$RT\bar{q}_{RST} = 0$$

$$S_1 = x_1x_2 + x_1q_2$$
$$= x_1(x_2 + q_2)$$
$$R_1 = 0$$
$$T_1 = \bar{x}_1q_1$$
$$S_2 = \bar{x}_1x_2$$
$$R_2 = 0$$
$$T_2 = \bar{x}_1\bar{x}_2q_1$$

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Q_1

=

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	-	-
10	0	0	-	-

S_1

+

q_1q_2 \ x_1x_2	00	01	11	10
00	-	-	1	-
01	-	-	1	1
11	-	-	1	1
10	-	-	1	1

$\bar{R}_1$

•

q_1q_2 \ x_1x_2	00	01	11	10
00	1	1	-	1
01	1	1	-	-
11	0	0	1	1
10	0	0	1	1

$\bar{T}_1$

•

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

q_1

+

q_1q_2 \ x_1x_2	00	01	11	10
00	0	0	-	0
01	0	0	-	-
11	1	1	0	0
10	1	1	0	0

T_1

•

q_1q_2 \ x_1x_2	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

$\bar{q}_1$

Figure 43. Realization expressions for RST flip-flop.

constraints must also be met. The best way to fill out the maps is to fill in the required minterm entries that give the required equality, and then check T and $\bar{T}$ for proper matching before further filling in to meet the constraints. Initially "0's" can be filled in the S_1-map where $Q_1 = 0$, and in the T_1-map where $Q_1 = 0$ and $q_1 = 0$. The "0's" in T_1 where $q_1 = 0$ go into "1's" in $\bar{T}_1$ where $q_1 = 0$. The corresponding minterms of $\bar{R}_1$ can be filled in with "-'s". For minterms where $Q_1 = 0$ and $q_1 = 1$, place "1's" in T_1 and "-'s" in $\bar{R}_1$ (or "-'s" in T_1 and "0's" in $\bar{R}_1$). Note that there is a choice that may be made, thereby making the R, S, and T-maps non-unique for fully specified Q-maps.

Where Q_1 had "1's", and $q_1 = 0$, set $S_1 = 1$, $\bar{R}_1 = 1$, and $\bar{T}_1 = -$ (or $S_1 = -$, $\bar{R}_1 = 1$, $\bar{T}_1 = 0$). For $Q_1 = 1$ and $q_1 = 1$, set $S_1 = -$, $\bar{R}_1 = 1$, and $\bar{T}_1 = 1$. Another choice may be made here. The entries could be placed side by side in the maps so that when the final covers are placed, the optimum choice can be made.

The availability of multiple minterm entry selections is due to the fact that there are two complete flip-flops combined into one. Each separate flip-flop is capable of setting or resetting the combined flip-flop, hence more than one choice is available in certain situations.

Summary for sequential flip-flops

The previous algorithms for determining the input equations for specified types of flip-flops is summarized as follows. The RS flip-flop is equivalent to the asynchronous case and the RS algorithm is taken from Chapter V.

RS flip-flop

The characteristic and constraint equations are: $Q = S + \bar{R}q$, $RS = 0$; where Q represents entries in the "next state" map, and q represents the "present state" variable.

$Q = 0$: (a) $S = 0$ in all corresponding minterms.
(b) $q = 0$: $R = -$.
(c) $q = 1$: $R = 1$.
$Q = 1$: (a) $q = 0$: $S = 1$, $R = 0$.
(b) $q = 1$: $S = -$, $R = 0$.

D flip-flop

The characteristic equation is: $Q = D$. The input maps are identical to the state variable maps.

T flip-flop

The characteristic equation is: $Q = T\bar{q} + \bar{T}q$.

$Q = 0$: (a) $q = 0$: $T = 0$.
(b) $q = 1$: $T = 1$.
$Q = 1$: (a) $q = 0$: $T = 1$.
(b) $q = 1$: $T = 0$.

JK flip-flop

The characteristic equation is: $Q = J\bar{q} + \bar{K}q$. It should be obvious that this is similar to the T flip-flop with the "T" function being shared by "J" and "K".

$Q = 0$: (a) $q = 0$: $J = 0$, $K = -$.

(b) $q = 1$: $J = -$, $K = 1$.

$Q = 1$: (a) $q = 0$: $J = 1$, $K = -$.

(b) $q = 1$: $J = -$, $K = 0$.

RST flip-flop

The characteristic and constraint equations are:

$Q = S + \bar{R}\bar{T}q + T\bar{q}$, $RS = 0$, $STq = 0$, and $RT\bar{q} = 0$.

$Q = 0$: (a) $S = 0$ in all corresponding minterms.

(b) $q = 0$: $T = 0$, $R = -$.

(c) $q = 1$: $T = -$, $R = 1$;

or $T = 1$, $R = -$.

$Q = 1$: (a) $q = 0$: $S = -$, $R = 0$, $T = 1$;

or $S = 1$, $R = 0$, $T = -$.

(b) $q = 1$: $S = -$, $R = 0$, $T = 0$.

"Don't care" conditions

If "don't cares" occur in the state maps, they are inserted in the corresponding minterm positions of the T, D, and JK input maps. The RS and RST input maps should have a "+" inserted in the corresponding minterm positions, and final covers placed so as not to contradict the constraint equations as was done for the RS flip-flop in Chapter V.

Each minterm position can be considered separately for a choice of which entry algorithm is desired (if a choice is available). In this way a more optimum map may be achieved in terms of the minimum number of logic gates required in the final implementation. The best way to do this is to draw out the maps in the characteristic equation and observe how each entry is being placed. Previous entries can be altered as the form of the map takes shape.

CHAPTER VII

AN APPLICATION OF THE MAP METHOD TO TRANSITION FLIP-FLOPS

Types of transition flip-flops

It was mentioned in Chapter IV that transition flip-flops required more than one state variable to adequately describe them using the Huffman approach. In this chapter the clocked JK, "one's-catching" JK, D, T, and unclocked T flip-flops will be analyzed and their use in realizing asynchronous flow matrices will be described. The method will consist of modifying a given flow table into a form that is realizable using the selected transition flip-flop.

Clocked JK flip-flop

The clocked JK flip-flop to be analyzed is the same one used in Chapter VI. Its flow table and reduction are shown in Figure 44. The table is set up with the enabling clock transition ("0" to "1") dividing the flow table into two vertical halves. Since circuit action occurs only during the clock transition, this facilitates filling out the flow table. Note that the table is normal only with respect to the clock transitions. The state reduction was chosen with "don't cares" to give simpler characteristic expressions.

CJK state	000	001	011	010	100	101	111	110	Q
1	①	2	3	4	5	-	-	-	0
2	1	②	3	4	-	6	-	-	0
3	1	2	③	4	-	-	15	-	0
4	1	2	3	④	-	-	-	16	0
5	1	-	-	-	⑤	6	7	8	0
6	-	2	-	-	5	⑥	7	8	0
7	-	-	3	-	5	6	⑦	8	0
8	-	-	-	4	5	6	7	⑧	0
9	⑨	10	11	12	13	-	-	-	1
10	9	⑩	11	12	-	6	-	-	1
11	9	10	⑪	12	-	-	7	-	1
12	9	10	11	⑫	-	-	-	16	1
13	9	-	-	-	⑬	14	15	16	1
14	-	10	-	-	13	⑭	15	16	1
15	-	-	11	-	13	14	⑮	16	1
16	-	-	-	12	13	14	15	⑯	1

CJK state	000	001	011	010	100	101	111	110	Q
a	①	②	3	4	⑤	⑥	⑦	⑧	0
b	1	2	③	④	-	-	15	16	0
c	⑨	10	11	⑫	⑬	⑭	⑮	⑯	1
d	9	⑩	⑪	12	-	6	7	-	1

Figure 44. Clocked JK flip-flop primitive flow table and merged flow table.

The state assignment and state maps are shown in Figure 45. The characteristic equation consists of two equations. This means that in certain cases a pair of states in a flow matrix may be realized with one flip-flop. It will be seen that such is the case, and the conditions under which this is possible will be discussed.

q_1q_2 \ CJK	000	001	011	010	100	101	111	110	Q
00	00	00	01	01	00	00	00	00	0
01	00	00	01	01	-	-	11	11	0
11	11	10	10	11	11	11	11	11	1
10	11	10	10	11	-	00	00	-	1

	C=0				C=1			
q_1q_2 \ JK	00	01	11	10	00	01	11	10
00	0	0	0	0	0	0	0	0
01	0	0	0	0	-	-	1	1
11	1	1	1	1	1	1	1	1
10	1	1	1	1	-	0	0	-

Q_1

$$Q_1 = \bar{C}q_1 + Cq_2 = Q$$

	C=0				C=1			
q_1q_2 \ JK	00	01	11	10	00	01	11	10
00	0	0	1	1	0	0	0	0
01	0	0	1	1	-	-	1	1
11	1	0	0	1	1	1	1	1
10	1	0	0	1	-	0	0	-

Q_2

$$Q_2 = \bar{K}q_1 + \bar{C}J\bar{q}_1 + Cq_2$$

Figure 45. Clocked JK characteristic equations and maps.

The characteristic equations are required to derive the flip-flop input maps. As an example, a single input, two-state map will be used to illustrate the conditions to be met by the two states.

JK state constraints

The map relationships are shown in Figure 46 for a single input, two-state flow matrix without regard to

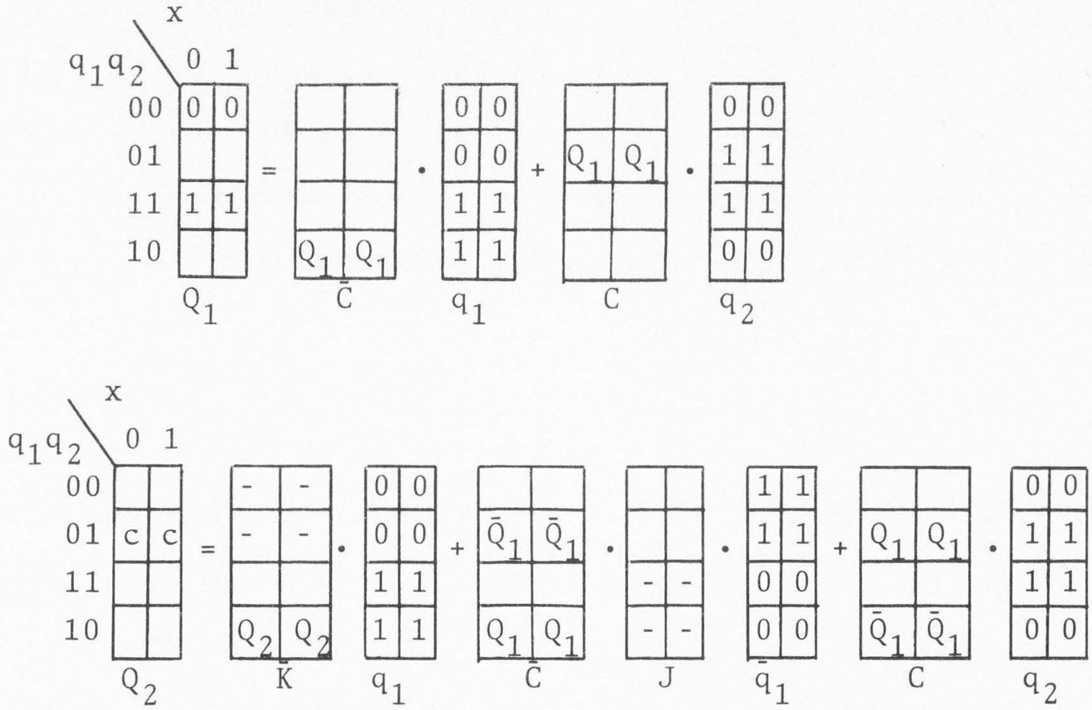


Figure 46. Map equations with static constraints for JK flip-flop.

the output function. It will be shown how the output function can be included in any acceptable (in terms of flip-flop constraints) flow table later in this chapter. A single input is assumed to keep the analysis simple, and is not necessarily a functional constraint on a realization using clocked JK flip-flops. Conditions on Q_1 are derived from observation of the required map relationships, and the minterm entries of Q_1 are placed in the pertinent positions of the "C" and "$\bar{C}$" maps in the Q_1 equation. Since it is implicit in the map equations that all of the maps have the same input, only the left-most map in any one equation has its inputs shown. This practice will be carried throughout the chapter. The clock entries from the Q_1 equation are copied into the Q_2 equation where the "C" and "$\bar{C}$" maps occur. The Q_1 map must have "0's" in its first row ($q_1q_2 = 00$), and "1's" in its third row ($q_1q_2 = 11$) to meet the static requirements of the Q_1 equation. The Q_2 map must have "1's" in its second row minterm positions ($q_1q_2 = 01$) if the Q_1 map has "1's" in the corresponding positions. Note that both minterms need not be "1" if only one position of the Q_1 map has a "1". These positions are marked with a "c" (for check) as a reminder. The input maps can be filled in while watching for transitions occurring in the "C" map and the "J" or "K" maps. One of the original assumptions was that when a transition on the clock line occurs (in the positive, or "0" to "1", direction), the "J" and "K" lines should be stable.

Input constraints

Transitions in the flow matrix should be compared with the corresponding minterm positions in the "C" map. If the "C" map has a "0" to "1" change in the direction of the transition, then the "J" and "K" maps should be checked to see that their corresponding minterm positions contain no logic changes for the same transition. This is to insure that simultaneous changes do not occur on the transition input (clock) and level inputs (J or K).

Flow table constraints for JK flip-flop

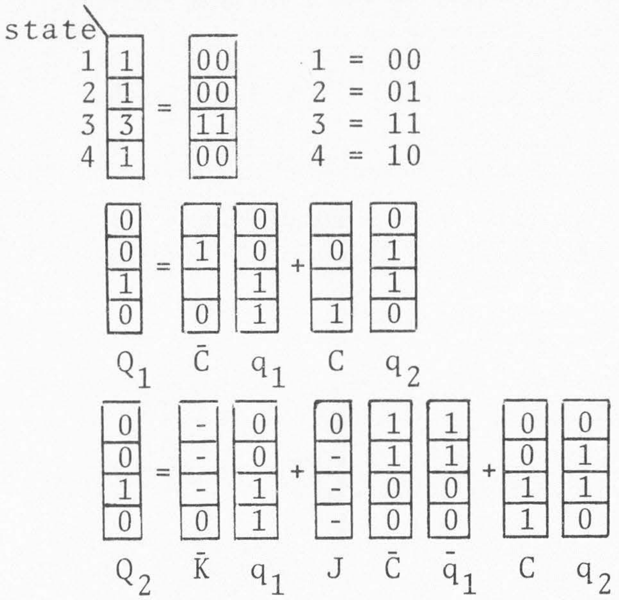
The initial flow table has to be set up to represent a realizable circuit using JK flip-flops. Using the characteristic equations for the JK flip-flop, it is easy to see that only the column transitions in the flow diagram are inherently constrained. The transition constraints, along with the static constraints, make up the characteristic equation-induced constraints.

Using the previously derived map relations for Q_1 and Q_2, it is evident that only certain states are allowed in each of the four possible vertical minterm positions of any given "next state" column. The allowed states are shown in Figure 47. For the states shown, there are 2x3x2x4 = 48 possible different choices for one column in a flow table with four states. Of these forty eight possible choices, only twelve meet the "transition" constraints as specified by the characteristic equations.

state \ input column	
1	1,2
2	1,2,3
3	3,4
4	1,2,3,4

Figure 47. Possible "next state" entries in JK realizable flow table.

The twelve allowed columns are shown in Figure 48, along



1 J=0	1 J=0	1 J=0	1 J=0	1 J=-	1 J=-
1 K=1	1 K=0	1 K=0	1 K=1	3 K=1	3 K=0
3 $C=q_1$	3 $C=q_1$	3 C=0	4 C=0	3 C=1	3 C=1
1	2	3	4	1	2

1 J=-	1 J=-	2 J=1	2 J=1	2 J=1	2 J=1
3 K=0	3 K=1	2 K=1	2 K=0	2 K=0	2 K=1
3 $C=\bar{q}_1$	4 $C=\bar{q}_1$	3 $C=q_1$	3 $C=q_1$	3 C=0	4 C=0
3	4	1	2	3	4

Figure 48. Acceptable vertical transitions for JK flip-flop.

with the derivation of the first one. The selection criteria was based on the fact that none of the derived flip-flop inputs could be functions of q_2 (the internal flip-flop variable). The flip-flop output is identical to q_1. A perusal of the possible transitions shows that different states with the same output (1 and 2, or 3 and 4) cannot coexist in the same column in their stable positions. It is for this reason that the Z-map can always be constructed such that the system output (Z) is not a function of q_2.

Three-variable flow table

A flow table with three state variables could have two of the three variables realized by a JK flip-flop if the flow table meets the constraints set by the characteristic equations. A single input-column of some arbitrary flow matrix is shown in Figure 49. It should be obvious that the transitions for the two state variables used to represent the JK flip-flop must meet the same vertical transition constraints in each of the four-state halves of the column (a simple application of the characteristic equations for the column will show this). The "next state" entries of the flow table column are a_1b_1 through a_8b_8, where "a" refers to a binary "1" or "0" and is the q_3 portion of the next state code, and "b" refers to a two-bit binary code that refers to the q_1q_2 portion of the next state code. It is assumed that the state variables q_1 and q_2 are to be realized with the JK flip-flop. That means columns b_1, b_2, b_3, b_4, and b_5, b_6, b_7, b_8 must be one of the twelve acceptable transitions.

$q_3q_1q_2$ \ I		a	b
000	a_1b_1	0	1
001	a_2b_2	0	2
011	a_3b_3	0	3
010	a_4b_4	0	4
100	a_5b_5	1	1
101	a_6b_6	1	2
111	a_7b_7	1	3
110	a_8b_8	1	4

q_1q_2 \ q_3	0	1
00	a_1	a_5
01	a_2	a_6
11	a_3	a_7
10	a_4	a_8

Q_3

$a_1 = a_2$

$a_3 = a_4$

$a_5 = a_6$

$a_7 = a_8$

Figure 49. Three-variable flow table column and Q_3 map.

If it is assumed that q_1 is the output of the JK realization, then Q_3 can be a function only of q_1, itself, and the system input variables because q_2 is an inaccessible variable. This puts the constraint on Q_3 such that $a_1 = a_2$, $a_3 = a_4$, $a_5 = a_6$, and $a_7 = a_8$. If the flow table for the desired function can be manipulated to meet the above constraints, then the function's state behavior can be realized with a JK flip-flop in combination with an RS flip-flop and logic gates. The RS flip-flop could also be realized with logic gates, if desired.

In manipulating the flow table, the output should not be a function of the JK flip-flop internal variable (q_2). If it is, then a transition position in the Z-map could be modified to eliminate this dependency.

Four-variable flow table

A four-variable flow table can be implemented using two JK flip-flops if the flow table conforms to the constraints of the characteristic equations. Figure 50 shows one column of a four-variable map with the columns

$q_1q_2q_3q_4$	I	a	b
0000	$a_1\ b_1$	1	1
0001	$a_2\ b_2$	1	2
0011	$a_3\ b_3$	1	3
0010	$a_4\ b_4$	1	4
0100	$a_5\ b_5$	2	1
0101	$a_6\ b_6$	2	2
0111	$a_7\ b_7$	2	3
0110	$a_8\ b_8$	2	4
1000	$a_9\ b_9$	3	1
1001	$a_{10}b_{10}$	3	2
1011	$a_{11}b_{11}$	3	3
1010	$a_{12}b_{12}$	3	4
1100	$a_{13}b_{13}$	4	1
1101	$a_{14}b_{14}$	4	2
1111	$a_{15}b_{15}$	4	3
1110	$a_{16}b_{16}$	4	4

Figure 50. Four-variable map column.

labeled with the binary code, and a corresponding two-tuple. The "next state" entries are labeled "ab". "a" refers to the destination major group (defined by state variables q_1q_2), and "b" refers to the destination subgroup (defined by state variables q_3q_4). Substituting the column into the characteristic equations shows that each of the subgroups (groups of four minterm positions where state label "a" = constant) must consist of one of the twelve vertical transition assignments and the inputs to the q_3q_4 flip-flop should not include q_2. This assures that the JK flip-flop representing the q_3q_4 state variable pair can be realized.

A rearrangement of the variables shows what constraints must be met by the "a's", "b's", q_1q_2 and q_3q_4. Figure 51 gives some insight to these constraints. Since only q_3 is available from the q_3q_4 flip-flop, any input to the q_1q_2 or q_3q_4 flip-flops cannot include q_4. Hence we have the equalities shown at the right of Figure 51. It is evident that each column of the Q_1Q_2 and Q_3Q_4 maps must meet the constraints imposed by the characteristic equations. That means each vertical column ($a_1a_5a_9a_{13}$, $a_2a_6a_{10}a_{14}$, a_3a_7-$a_{11}a_{15}$, $a_4a_8a_{12}a_{16}$, $b_1b_2b_3b_4$, $b_5b_6b_7b_8$, $b_9b_{10}b_{11}b_{12}$, and $b_{13}b_{14}b_{15}b_{16}$) must be one of the twelve acceptable vertical transition columns. The equalities in Figure 51 imply that the first two columns are equal and the last two columns

q_1q_2 \ q_3q_4	00	01	11	10
00	a_1	a_2	a_3	a_4
01	a_5	a_6	a_7	a_8
11	a_9	a_{10}	a_{11}	a_{12}
10	a_{13}	a_{14}	a_{15}	a_{16}

Q_1Q_2

$a_1 = a_2$; $a_3 = a_4$
$a_5 = a_6$; $a_7 = a_8$
$a_9 = a_{10}$; $a_{11} = a_{12}$
$a_{13} = a_{14}$; $a_{15} = a_{16}$

q_3q_4 \ q_1q_2	00	01	11	10
00	b_1	b_5	b_9	b_{13}
01	b_2	b_6	b_{10}	b_{14}
11	b_3	b_7	b_{11}	b_{15}
10	b_4	b_8	b_{12}	b_{16}

Q_3Q_4

$b_1 = b_5$; $b_9 = b_{13}$
$b_2 = b_6$; $b_{10} = b_{14}$
$b_3 = b_7$; $b_{11} = b_{15}$
$b_4 = b_8$; $b_{12} = b_{16}$

$q_1q_2q_3q_4$		
0000	a	k
0001	a	l
0010	b	m
0011	b	n
0100	c	k
0101	c	l
0110	d	m
0111	d	n
1000	e	o
1001	e	p
1010	f	q
1011	f	r
1100	g	o
1101	g	p
1110	h	q
1111	h	r

first column = q_1q_2

second column = q_3q_4

Necessary vertical transitions:

a b k o

c d l p

e f m q

g,h,n,r.

Figure 51. Q_1Q_2 and Q_3Q_4 maps with entry constraints and required column equalities.

are equal in the Q_1Q_2 map and Q_3Q_4 map. If each column of a four variable flow table meets the above criteria, then it can be realized with two JK flip-flops and some interconnection logic.

The above manipulations show the form that each of the two, three, and four variable flow tables must be in to achieve their realizations using JK flip-flops. Larger flow tables can be examined column by column using similar techniques. Some synthesis examples will be given at the end of this chapter.

D flip-flop

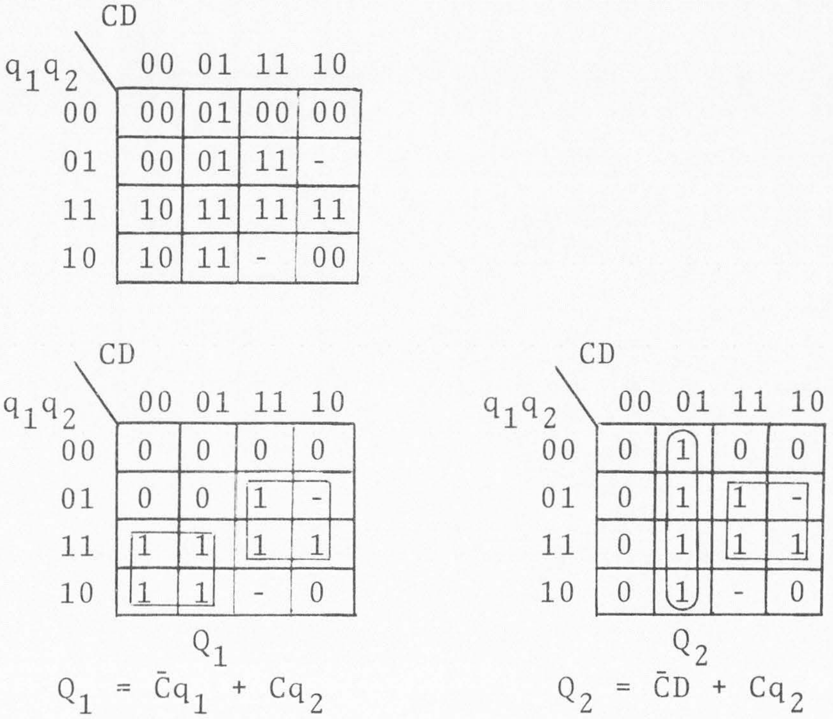
The primitive flow table and reduction for the D flip-flop are shown in Figure 52. The state assignment

state \ CD	00	01	11	10	Q
1	①	2	-	4	0
2	1	②	7	-	0
3	-	2	③	4	0
4	1	-	3	④	0
5	⑤	6	7	-	1
6	5	⑥	7	-	1
7	-	6	⑦	8	1
8	5	-	7	⑧	1

state \ CD	00	01	11	10	Q
a	①	2	③	④	0
b	1	②	7	-	0
c	5	⑥	⑦	⑧	1
d	⑤	6	-	4	1

Figure 52. Primitive and merged flow tables for D flip-flop.

and characteristic equations are given in Figure 53. Using the characteristic equations the characteristic



q_1q_2 \ CD	00	01	11	10
00	00	01	00	00
01	00	01	11	-
11	10	11	11	11
10	10	11	-	00

q_1q_2 \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	-
11	1	1	1	1
10	1	1	-	0

Q_1

$Q_1 = \bar{C}q_1 + Cq_2$

q_1q_2 \ CD	00	01	11	10
00	0	1	0	0
01	0	1	1	-
11	0	1	1	1
10	0	1	-	0

Q_2

$Q_2 = \bar{C}D + Cq_2$

Figure 53. Flow matrix and characteristic equations for D flip-flop.

maps can be checked as shown in Figure 54 to determine static constraints on the two states that represent the D flip-flop. It is seen from an inspection of the maps that the same type of static constraints that held for the JK flip-flop also hold for the D flip-flop. The "c" in the Q_2 map means that if the Q_1 map has a "1" in the same minterm position, then the "c" in Q_2 must be a "1". This means the first state row of a flow table column can have only one of two entries (1 or 2); the second state

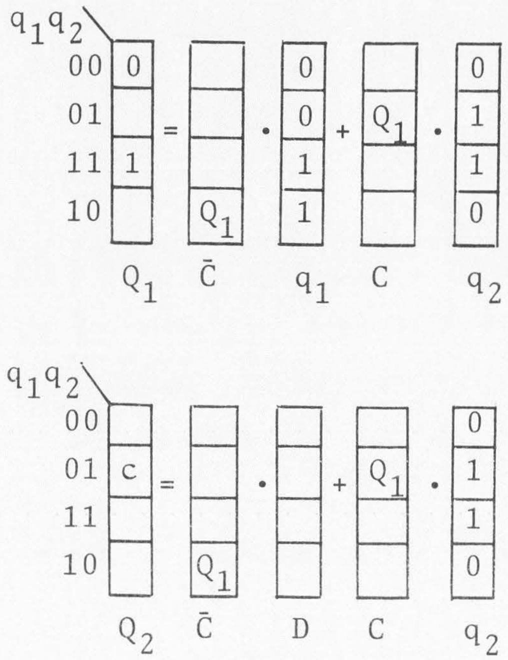


Figure 54. Static state constraints on D flip-flop.

row can have one of three entries (1, 2, or 3); the third state row can have one of two entries (3 or 4); and the fourth state row can have one of four entries (1, 2, 3, or 4). This is the same as the JK flip-flop, but out of the forty eight possible columns, only nine are acceptable to be used as entries in the D flip-flop flow table. The acceptable transitions are listed in Figure 55, along with their resultant input expressions. The acceptance criteria is such that each input variable of the flip-flop (C or D) cannot be a function of q_2 since it is inaccessible (as in the JK case).

Flow table constraints for D flip-flop

The flow table that is to be realized using D flip-flops must meet the requirements as set forth in the JK

1 $C=q_1$	1 $C=0$	1 $C=0$	1 $C=1$	1 $C=\bar{q}_1$
1 $D=0$	1 $D=q_1$	1 $D=0$	3 $D=-$	3 $D=1$
3	3	4	3	3
1	3	4	1	3

1 $C=\bar{q}_1$	2 $C=q_1$	2 $C=0$	2 $C=0$
3 $D=0$	2 $D=1$	2 $D=1$	2 $D=\bar{q}_1$
4	3	3	4
4	1	3	4

Figure 55. Acceptable vertical transitions for D flip-flop.

flip-flop section, since the derivation is identical. The only change is in the characteristic equations, and these give rise to the nine acceptable vertical transitions in the D flip-flop case as opposed to the twelve in the JK flip-flop case.

Clocked T flip-flop

The clocked T flip-flop primitive flow table and merged flow table are shown in Figure 56. The flow matrix

state \ CT	00	01	11	10	Q
1	①	2	-	4	0
2	1	②	7	-	0
3	-	2	③	4	0
4	1	-	3	④	0
5	⑤	6	-	8	1
6	5	⑥	3	-	1
7	-	6	⑦	8	1
8	5	-	7	⑧	1

state \ CT	00	01	11	10	Q
a	①	2	③	④	0
b	1	②	7	-	0
c	⑤	6	⑦	⑧	1
d	5	⑥	3	-	1

Figure 56. Clocked T flip-flop primitive flow table and merged flow table.

and characteristic equations are shown in Figure 57, and

q_1q_2 \ CT	00	01	11	10
00	00	01	00	00
01	00	01	11	-
11	11	10	11	11
10	11	10	00	-

q_1q_2 \ CT	00	01	11	10
00	0	0	0	0
01	0	0	1	-
11	1	1	1	1
10	1	1	0	-

Q_1

q_1q_2 \ CT	00	01	11	10
00	0	1	0	0
01	0	1	1	-
11	1	0	1	1
10	1	0	0	-

Q_2

$Q_1 = \bar{C}q_1 + Cq_2$ $\qquad$ $Q_2 = \bar{T}q_1 + T\bar{C}\bar{q}_1 + Cq_2$

Figure 57. Clocked T flip-flop flow matrix and characteristic equations.

the static state constraints are given in Figure 58. Note

$$\begin{array}{c|c|c|c|c|c|c|c|c|c}
q_1q_2 & & & & & & & & & \\
00 & 0 & & & & 0 & & & & 0 \\
01 & & = & & \cdot & 0 & + & & \cdot & 1 \\
11 & 1 & & & & 1 & & & & 1 \\
10 & & & & & 1 & & & & 0 \\
 & Q_1 & & \bar{C} & & q_1 & & C & & q_2
\end{array}$$

$$\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c}
q_1q_2 & & & & & & & & & & & & & & & \\
00 & & & & & 0 & & & & & & 1 & & & & 0 \\
01 & c & = & & \cdot & 0 & + & & \cdot & & \cdot & 1 & + & & & 1 \\
11 & & & & & 1 & & & & & & 0 & & & & 1 \\
10 & & & & & 1 & & & & & & 0 & & & & 0 \\
 & Q_2 & & \bar{T} & & q_1 & & T & & \bar{C} & & \bar{q}_1 & & C & & q_2
\end{array}$$

Figure 58. Clocked T flip-flop state constraints.

that the flip-flop output (Q) is defined to be identical to the state variable q_1 for all of the flip-flops. The static state constraints are seen to be the same as the JK and D flip-flops. The transition constraints as derived from the application of the characteristic maps to the forty eight possible transition cases are shown in Figure 59.

1 $C=q_1$	1 $C=q_1$	1 $C=0$	1 $C=0$	1 $C=1$	1 $C=1$
1 $T=q_1$	1 $T=0$	1 $T=0$	1 $T=q_1$	3 $T=1$	3 $T=0$
3	3	3	4	3	3
1	2	3	4	1	2
1 $C=\bar{q}_1$	1 $C=\bar{q}_1$	2 $C=q_1$	2 $C=q_1$	2 $C=0$	2 $C=0$
3 $T=0$	3 $T=1$	2 $T=1$	2 $T=\bar{q}_1$	2 $T=\bar{q}_1$	2 $T=1$
3	4	3	3	3	4
3	4	1	2	3	4

Figure 59. Clocked T flip-flop acceptable vertical transitions.

The derivation is identical to the JK case. The vertical transition set is identical to the JK flip-flop vertical transition set, so a flow table that can be realized with JK flip-flops can also be realized with clocked T flip-flops without having to change its form. Note that there is no vertical transition where "C" has a positive transition and T also changes.

Flow table constraints for clocked T flip-flop

If a given flow table is to be realized with clocked T flip-flops, it must conform to the specifications set forth in the JK flip-flop flow table constraint section.

Since the clocked T flip-flop acceptable vertical transition set is identical to the JK flip-flop set, all flow tables that can be realized with JK flip-flops can also be realized with T flip-flops. Simultaneous adverse clock and T transitions are more likely to occur using T flip-flops, however, and should be watched for. An adverse transition is one where the clock input has a positive transition coincident with a "T" transition (either positive or negative). If this occurs, an alternate column should be tried to see if the problem can be alleviated.

Unclocked T flip-flop

The unclocked T flip-flop primitive flow table and merged flow table are identical. The flow table, flow matrix, and characteristic equations are shown in Figure 60.

state \ T	0	1	Q
1	2	①	0
2	②	3	0
3	4	③	1
4	④	1	1

q_1q_2 \ T	0	1
00	01	00
01	01	11
11	10	11
10	10	00

q_1q_2 \ T	0	1
00	0	0
01	0	1
11	1	1
10	1	0

Q_1

q_1q_2 \ T	0	1
00	1	0
01	1	1
11	0	1
10	0	0

Q_2

$Q_1 = \bar{T}q_1 + Tq_2$

$Q_2 = \bar{T}\bar{q}_1 + Tq_2$

Figure 60. Unclocked T flip-flop flow table, flow matrix, and characteristic equations.

The output (Q) is again chosen to be identical to q_1 for convenience, and to conform with previous analysis. The unclocked T flip-flop is controlled by only one input (T) and changes its output state only upon the occurrence of a positive-going transition on it. The static state constraints can be found by plotting the characteristic maps, as shown in Figure 61. Only one column is shown with the state assignment on the left. Since all columns (external inputs) are derived in the same way, only one column is needed to show the relationship.

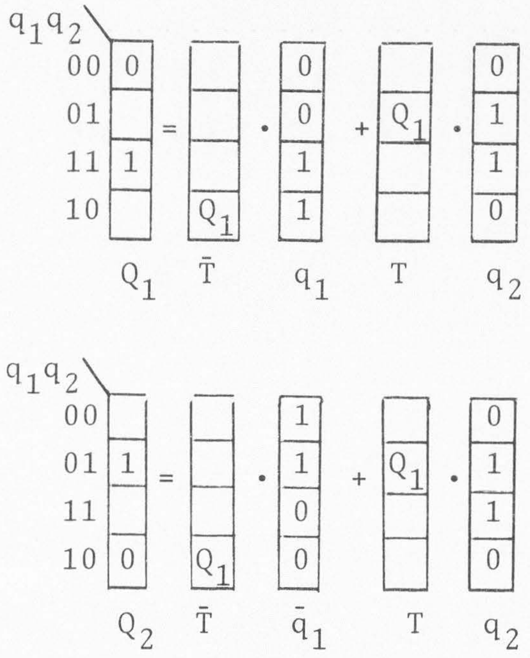


Figure 61. Unclocked T static state constraints.

The unclocked T flip-flop has more constraints on its vertical state transitions, as is evident from Figure 61. The state assignment possibilities and acceptable transitions are shown in Figure 62. There are only $2^4 = 16$ possible static assignments, and out of these sixteen only four are acceptable. These four are a subset of the D flip-flop set, and hence also a subset of the JK flip-flop and clocked T flip-flop set. It should be evident that the JK and clocked T flip-flops are the most versatile of the existing transition flip-flops available.

q_1q_2 \ I	
00	0-
01	-1
11	1-
10	-0

$\equiv$

state \ I		Q
1	1,2	0
2	2,3	0
3	3,4	1
4	4,1	1

1 T=1	1 $T=\bar{q}_1$	2 $T=q_1$	2 T=0
3	3	2	2
3	4	3	4
1	4	1	4

Figure 62. Unclocked T flip-flop acceptable states and transitions.

Flow table constraints for unclocked T flip-flop

The JK specifications on allowable flow tables also apply to the unclocked T flip-flop (due to the similarity of its derivation), except for the fact that only four vertical transition sets are allowed in the groupings. This makes the unclocked T flip-flop the least versatile of the group. However, due to the single input, if a flow table can be realized using the unclocked T flip-flop, then it may be a cheaper realization due to the decrease in required connections.

"One's-catching" negative-edge triggered JK flip-flop

A commercially available JK flip-flop will be analyzed to show how the flow table may be derived from the circuit configuration. Figure 63 shows an equivalent circuit representation of a "one's-catching" negative-edge triggered JK flip-flop, and the classical flip-flop as derived

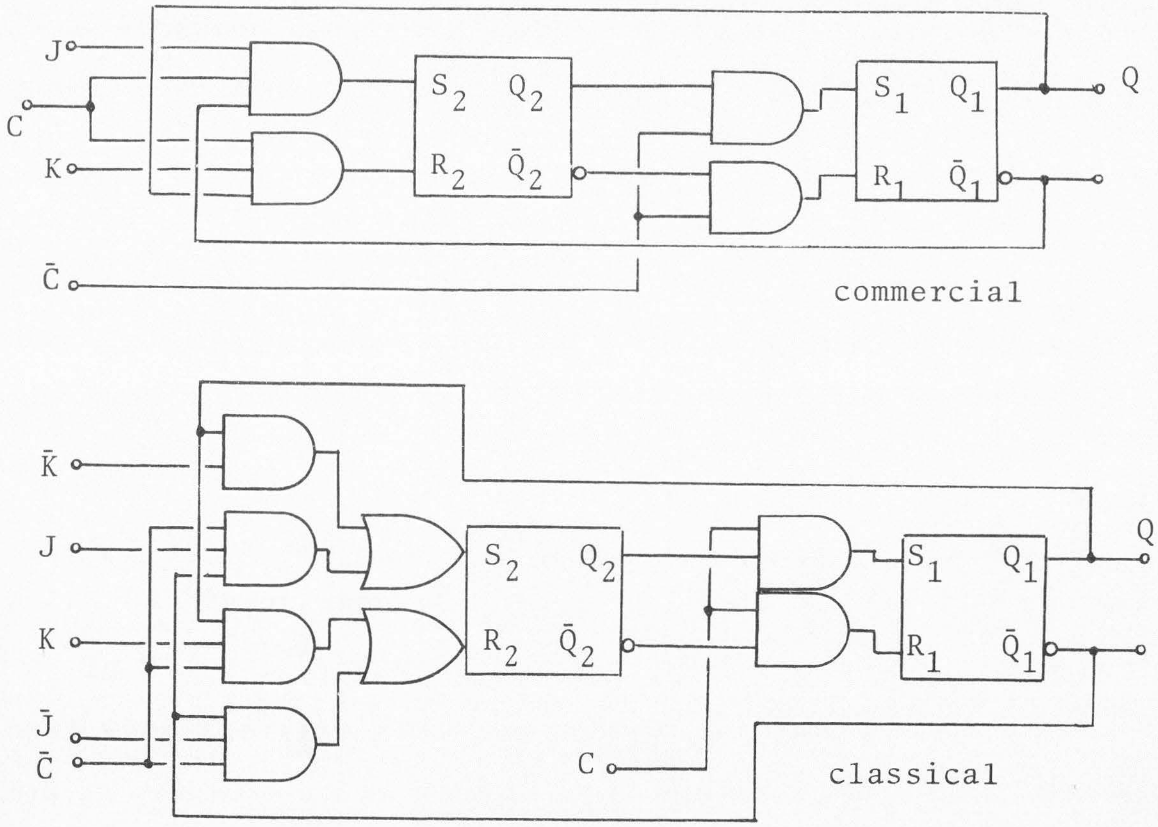


Figure 63. Commercial "one's-catching" and classical JK realizations.

from the flow table of Figure 44. The output is affected only on the negative-going transition of the clock line in the "one's-catching" JK, and was chosen to be coincident with q_1 for consistency. The classical JK realization is given for comparison. It's schematic was derived using state maps of Figure 45 along with the RS flip-flop algorithm of Chapter V.

Since the realization uses RS flip-flops, the RS flip-flop characteristic maps can be used in "reverse" order to derive the state maps (Q_1 and Q_2) for the commercial JK. The inputs from the flip-flops can be taken from the diagram (Figure 63) and plugged into the input expressions of the characteristic maps. The resulting flow matrix, flow table, characteristic equations, and acceptable vertical transition set are given in Figure 64. The operation of the flip-flop can be easily predicted from the flow table. This saves having to trace through the circuit schematic as all the functional specifications are contained in the table. It should be evident from the flow table that the flip-flop is susceptible to spikes on the "J" input when the flip-flop's output is "0", and the clock is high; and is susceptible to spikes on the "K" input when the flip-flop's output is "1", and the clock is high.

The acceptable vertical transition set is unique due to the fact that some of the columns have stable states flow matrix:

q_1q_2 \ CJK	000	001	011	010	100	101	111	110
00	00	00	00	00	00	00	01	01
01	11	11	11	11	01	01	01	01
11	11	11	11	11	11	10	10	11
10	00	00	00	00	10	10	10	10

flow table:

state \ CJK	000	001	011	010	100	101	111	110	Q
1	①	①	①	①	①	①	2	2	0
2	3	3	3	3	②	②	②	②	0
3	③	③	③	③	③	4	4	③	1
4	1	1	1	1	④	④	④	3	1

characteristic equations:

$$Q_1 = Cq_1 + \bar{C}q_2 = Q$$

$$Q_2 = \bar{K}q_2 + \bar{C}q_2 + CJ\bar{q}_1 + \bar{q}_1q_2$$

acceptable vertical transitions:

1 J=0	1 J=0	1 J=0	1 J=-	1 J=-
2 K=-	2 K=0	2 K=1	3 K=-	3 K=0
3 $C=\bar{q}_1$	3 C=1	4 C=1	3 C=0	3 $C=q_1$
1	4	4	1	4

1 J=-	2 J=1	2 J=1	2 $J=\bar{q}_1$
3 K=1	2 K=-	2 K=0	2 K=1
4 $C=q_1$	3 $C=\bar{q}_1$	3 C=1	4 C=1
4	1	4	4

Figure 64. "One's-catching" JK flip-flop functional descriptives.

in each row--thereby creating a problem in realizing the output function (Z) if the output differs in the common rows where q_1 is constant. If such is the case, the flow table can always be modified in such a way so as to be realizable. This will be discussed in the next section.

Bounds on transition flip-flops to realize a given flow table

It has been shown that two state variables are required to describe a transition flip-flop (the ones that are looked at in this chapter). The minimum bound on the number of transition flip-flops required to realize a minimized flow table with n state variables is therefore n/2 if n is even, or n/2 + 1 if n is odd. n is assumed to be the least number of state variables required to uniquely code all of the states (which number is denoted s_o) in the flow table, and is not less that $\log_2 s_o$. Note that it is assumed that none of the states of the flow table are equivalent, i.e. the table is in minimal form.

The maximum number of transition flip-flops required to realize a given flow table is simply equal to the number of state variables required to uniquely code each state. If N is the required number of flip-flops, then it is evident that

$$n/2 \leq N \leq n.$$

To show that n is an upper bound on N requires a demonstration of the equivalence of the RS flip-flop to the transition flip-flops. In Chapter V it was noted that an RS flip-flop can be used to represent each of the state variables, and a synthesis procedure was given where any flow table could then be realized using this flip-flop. The required number of flip-flops was n.

The flow table shown in Figure 65 represents an equivalent flow table for the RS flip-flop if the inputs

(S and R) are "zeroed" when the flip-flop is turned on. Since each of the transition flip-flops covered contains

state \ RS	00	01	11	10
1	2	①	①	2
2	②	3	3	②
3	4	4	③	③
4	④	④	1	1

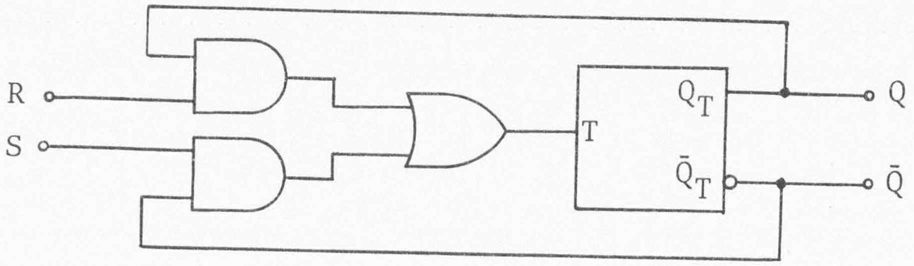


Figure 65. Generation of an RS flip-flop from a transition flip-flop.

all of the columns of the flow table in their respective vertical transition sets, it is possible to realize the flow table with all of them. The "T" realization is shown below the flow table. Note that the RS = 11 column is arbitrary since this input is not allowed.

The synthesis procedure is to use the RS algorithm for the generation of the input expressions, and then combine the expressions with the input buffer to the transition flip-flops. This realizes any n-state variable flow table with n transition flip-flops, and hence places an upper bound on the minimum number of flip-flops required.

After power is applied, a sanity cycle may be required to place all of the flip-flops in their usable states (2, 3, or 4).

Once the realization has been achieved, a backwards application of the characteristic maps will give the state maps, and from there the equivalent realizable flow table can be generated. This flow table will meet the acceptability criteria for column entries mentioned earlier in this chapter.

The bounds on the required number of flip-flops to realize a flow table only state that there exists some equivalent flow table whose flow matrix can be used to give realizable input equations for the type of flip-flop desired. The generation of a minimum realizable flow table seems to be an ad hoc procedure, and nothing can be said about it at this time.

Two-state synthesis example

A simple two-state example will be given to show how a given flow table may be modified into a flow table that may be realized using JK, D, clocked T, and unclocked T flip-flops. Assume the flow table shown in Figure 66. A single positive transition on the "x" input will turn on the output and it will stay on indefinitely after that. Since all of the transition flip-flops use a minimum of four states (two state variables) for realization, the given flow table will have to be expanded. Keeping the transitions of the original flow table in mind, and

state \ x	0	1	Z
1	①	2	0
2	②	②	1

Figure 66. Simple two-state flow table.

checking the acceptable vertical transition lists for the flip-flops, equivalent flow tables can be constructed that may be realized using the appropriate flip-flop. Figure 67 gives an equivalent flow table for each flip-flop. It should be noted that the chosen flow tables

state \ x	0	1	Z
1	2	①	0
2	②	3	0
3	③	③	1
4	3	3	1

JK

state \ x	0	1	Z
1	2	①	0
2	②	3	0
3	③	③	1
4	3	3	1

clocked T

state \ x	0	1	Z
1	2	①	0
2	②	3	0
3	③	③	1
4	3	3	1

D

state \ x	0	1	Z
1	2	①	0
2	②	3	0
3	4	4	1
4	④	④	1

unclocked T

Figure 67. Acceptable flow tables.

are not unique, since different column choices could have been made for all but the unclocked T flip-flop. It should be emphasized that the output (Z) is inherently specified to be a function of q_1 in the state coding. Figures 68 and 69 show the flow matrices and final input

q_1q_2 \ x	0	1
00	01	00
01	01	11
11	11	11
10	11	11

JK

q_1 \ x	0	1
0	1	-
1	1	-

J

$J = 1$

q_1 \ x	0	1
0	0	0
1	0	0

K

$K = 0$

$Z = q_1$

q_1 \ x	0	1
0	0	1
1	0	0

C

$C = x\bar{q}_1$

q_1q_2 \ x	0	1
00	01	00
01	01	11
11	11	11
10	11	11

clocked T

q_1 \ x	0	1
0	1	1
1	0	0

T

$T = \bar{x}\bar{q}_1$

$Z = q_1$

q_1 \ x	0	1
0	0	1
1	0	0

C

$C = x\bar{q}_1$

Figure 68. Flow matrices and input maps for JK and clocked T flip-flops.

$$\begin{array}{c|cc} q_1 \backslash x & 0 & 1 \\ \hline 00 & 01 & 00 \\ 01 & 01 & 11 \\ 11 & 11 & 11 \\ 10 & 11 & 11 \end{array}$$

D

$Z = q_1$

$$\begin{array}{c|cc} q_1 \backslash x & 0 & 1 \\ \hline 0 & 1 & 1 \\ 1 & 1 & 1 \end{array} \quad D = 1$$

D

$$\begin{array}{c|cc} q_1 \backslash x & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 0 & 0 \end{array} \quad C = \bar{q}_1 x$$

C

$$\begin{array}{c|cc} q_1 q_2 \backslash x & 0 & 1 \\ \hline 00 & 01 & 00 \\ 01 & 01 & 11 \\ 11 & 10 & 10 \\ 10 & 10 & 10 \end{array}$$

unclocked T

$Z = q_1$

$$\begin{array}{c|cc} q_1 \backslash x & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 0 & 0 \end{array} \quad T = x\bar{q}_1$$

T

Figure 69. Flow matrices and input maps for D and unclocked T flip-flops.

equations for each flip-flop. The input maps are taken from the acceptable vertical transition specifications instead of using the characteristic maps to save space.

It should be noted that the $\bar{q}_1$-row of the clock input map for the clocked T flip-flop has a zero-to-one transition where the T input map has a one-to-zero transition. It was mentioned earlier in this chapter that in implementing the correct flow table, input transitions of this sort must be avoided. In this case a fix would be to substitute the 1, 3, 4, 4 vertical transition set into the second column of the flow table. This would give T = $\bar{q}_1$ + x, which would alleviate the problem.

Four-state synthesis example

For certain flow tables a definite reduction in hardware can be achieved by using transition flip-flops. Figure 70 shows one example of a four-state system being

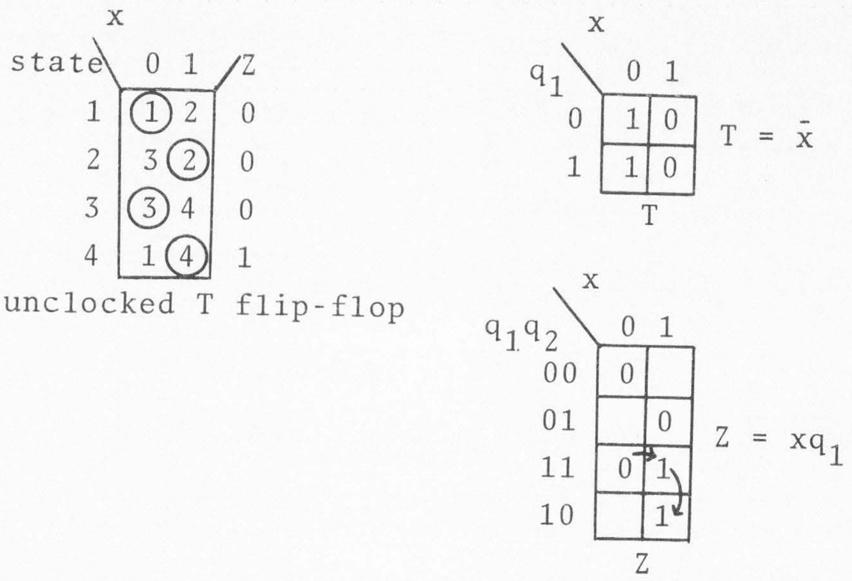


Figure 70. Four-state system implemented with one unclocked T flip-flop.

realized by a single flip-flop. The key to the implementation is in realizing the output (Z) without requiring access to the flip-flop internal variable (q_2). The output map shows the transitions that go to the "on" state, and the intermediate position (minterm q_1q_2x) has a choice of possible entries available to it. Placing a "1" in it assures that the output can be realized from accessible variables. If the output were a function of q_2 only, the flow table could have its state coding shifted by one row (every row shifted down or up). This would make the output a function of q_1 only.

A comparison of the classical realization and flip-flop realization is shown in Figure 71. It should be added that the essential hazard problem has been eliminated in the flip-flop realization. This is not an accident, but can also be achieved in other essential hazard flow tables by using transition flip-flops.

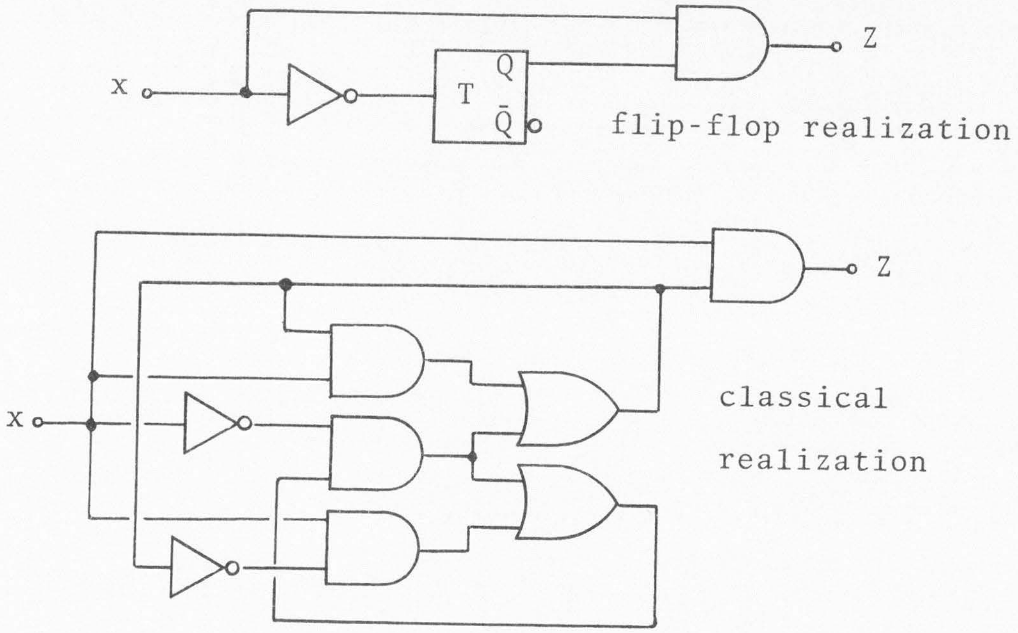


Figure 71. Schematic comparison of flip-flop and classical realizations.

Four-state synthesis using RS algorithm

If the example problem of Chapter II is to be realized using transition flip-flops, a problem is encountered--how to modify the flow table of Figure 20 into a flow table that meets the column transition constraints. The flow table to be realized is reproduced in Figure 72. Columns one and three are members of all the transition flip-flop transition sets, but column two is not a member of any while column four is a member of only the "one's-catching" flip-flop.

state \ x_1x_2	00	01	11	10	Z
1	①	2	2	①	0
2	3	3	②	②	0
3	③	③	4	4	0
4	1	3	④	④	1

Figure 72. Reproduction of Figure 20.

If the T flip-flop is to be used, then the RS synthesis algorithm can be used in conjunction with the T flip-flop in RS form. The RS flip-flop equations are given in Figure 31 and are as follows:

$$S_1 = \bar{x}_1 q_2$$

$$R_1 = \bar{x}_1 \bar{x}_2 \bar{q}_2$$

$$S_2 = \bar{x}_1 x_2 + x_2 \bar{q}_1$$

$$R_2 = x_1 q_1 .$$

The equivalent input of the T flip-flop can be taken from Figure 65:

$$T_i = R_i q_i + s_i \bar{q}_i,$$

where i indicates the i$\underline{\text{th}}$ flip-flop.

Combining the above two sets of equations (while checking for static hazard conditions) gives the following input equations for the T flip-flop:

$$T_1 = \bar{x}_1\bar{x}_2q_1\bar{q}_2 + \bar{x}_1\bar{q}_1q_2$$

$$T_2 = x_1q_1q_2 + \bar{x}_1x_2\bar{q}_2 + x_2\bar{q}_1\bar{q}_2$$

The circuit diagram is shown in Figure 73. If desired, the

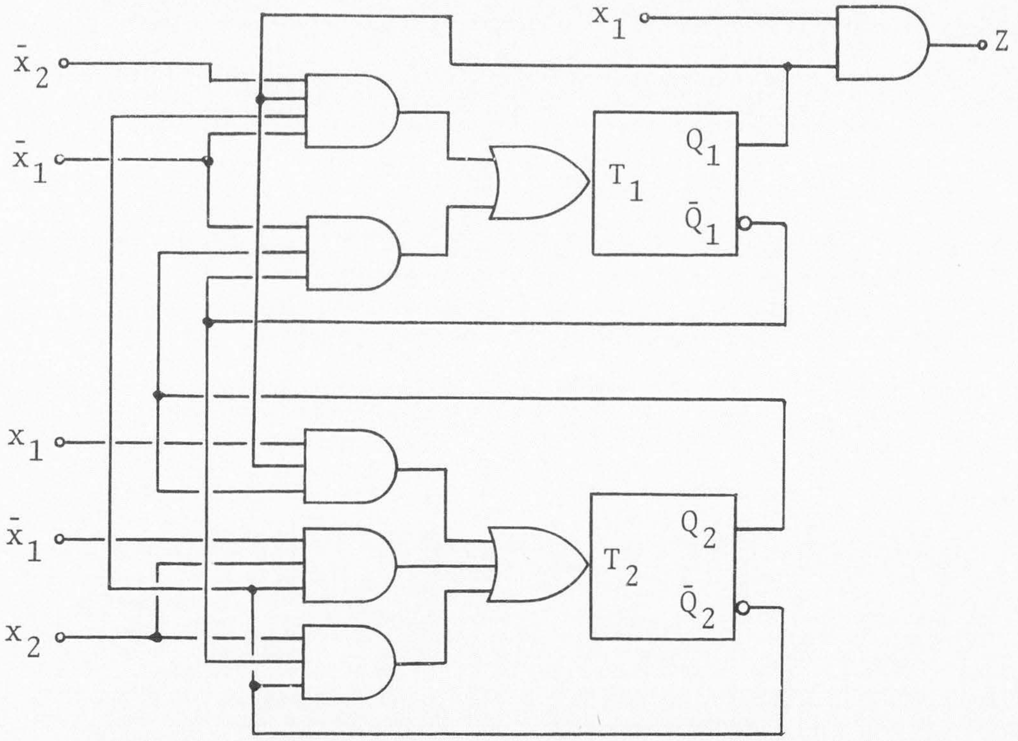


Figure 73. Example problem of Chapter II realized with T flip-flops.

sixteen-state flow table can be derived from an application of the characteristic maps, and is shown in Figure 74.

Q_1Q_2 \ x_1x_2	00	01	11	10
11	22	21	21	22
12	22	23	23	22
13	14	14	24	24
14	14	14	24	24
21	22	21	21	22
22	22	23	23	22
23	34	34	24	24
24	34	34	24	24
31	32	41	42	42
32	32	43	42	42
33	44	44	43	43
34	44	44	41	41
41	12	41	42	42
42	12	43	42	42
43	44	44	43	43
44	44	44	41	41

Figure 74. Transition-realizable flow table expansion for flow table of Figure 72.

The overall flow table can give an insight as to what type of sanity cycle would be required on power up. It is seen from an inspection of the flow table that the cycle consisting of x_1x_2 = 10, 00 would put the circuit into an acceptable state for the desired operation.

CHAPTER VIII

SUMMARY AND CONCLUSIONS

Summary

A basic outline of the procedures used in analyzing and synthesizing asynchronous sequential logic circuits has been presented. The generation of the state variable maps was seen to be immediately usable in the generation of asynchronous sequential circuits using standard flip-flops.

A method was presented in Chapter V that allowed the generation of input expressions for RS flip-flops by substituting functional expressions (in the form of Karnaugh maps) into the characteristic equation of the flip-flop. By using the state variable maps as "next state" descriptions of the flip-flops, the input expressions for the RS flip-flops were obtained. An algorithm was presented that specified what each minterm position of the R and S maps should be, once the state variable map was specified. The method was applied to synchronous sequential circuits in Chapter VI, and seemed to work quite well. The clock was assumed implicitly, and was not considered to be an input in the respective maps. This is standard in current synchronous sequential design.

Transition flip-flops were analyzed in Chapter VII using the same technique. The results were not as comprehensive as in the RS flip-flop case, but nevertheless some very interesting analysis tools were developed. All of the transition flip-flops described required a minimum of two state variables for complete characterization. It was noted that one of the state variables would necessarily be inaccessible, and this placed certain constraints on allowable state transitions in any given column of a flow table that was to be realized using transition flip-flops. The flip-flop output was chosen to be identical to one of the state variables.

The minterm-entry constraints of flow tables with up to four state variables (and any number of inputs) were specified. It was seen that each column of a flow table was independent of the other columns with respect to allowable minterm-entries.

Some examples were given to show possible approaches in synthesizing asynchronous sequential problems. A usable synthesis procedure was not specified, but bounds on the number of transition flip-flops required to realize any given flow table were specified, thereby assuring the designer of the existence of a workable solution.

Conclusions

The analysis techniques developed in this paper give a more complete picture of the action of flip-flops in both asynchronous and synchronous logic systems. From a given circuit configuration that contains flip-flops it is possible to "work backwards" and get a complete flow table description of the circuit. This lets the designer see at a glance exactly what his circuit action will be without having to trace through a circuit diagram. Problems, such as essential hazards, can be easily spotted and corrective techniques applied.

An existence proof was given for the generation of a transition flip-flop realizable flow table from any given flow table. This seems to imply that a flow table expansion algorithm may exist. Further work should be done in this area to see if one can be generated.

The structure of the analysis techniques lend themselves to computer programming. The RS flip-flop algorithm is especially suited for programming, and could be included in current design automation systems.

BIBLIOGRAPHY

Bartee, T. C., I. L. Lebow, and I. S. Reed. 1962. Theory and design of digital machines. McGraw-Hill, New York. 324 p.

Batra, Vinod. 1970. Design of asynchronous unit delays. IEEE Transactions on Computers, pp. 896-902.

Beizer, Boris. 1970. Towards a new theory of sequential switching networks. IEEE Transactions on Computers, pp. 939-956.

Beuscher, H. J., A. H. Budlong, M. B. Haverty, and G. Waldbaum. 1971. Electronic switching theory and circuits. Van Nostrand Reinhold, New York. 369 p.

Biswas, Nripendra N. 1974. State minimization of incompletely specified sequential machines. IEEE Transactions on Computers, pp. 80-84.

Bredeson, Jon G., and Paul T. Hulina. 1971. Generation of a clock pulse for asynchronous sequential machines to eliminate critical races. IEEE Transactions on Computers, pp. 225-226.

Bredeson, Jon G., and Paul T. Hulina. 1973. Synthesis of multiple-input change asynchronous circuits using transition-sensitive flip-flops. IEEE Transactions on Computers, pp. 524-531.

Brzozowski, J. A., and E. J. McCluskey. 1963. Signal flow graph techniques for sequential circuit state diagrams. IEEE Transactions on Electronic Computers, pp. 67-76.

Caldwell, Samuel H. 1958. Switching circuits and logical design. John Wiley & Sons, New York. 686 p.

Chuang, Y. H. 1969. Transition logic circuits and a synthesis method. IEEE Transactions on Computers, pp. 154-168.

Chuang, Henry Y. H., and Santanu Das. 1973. Synthesis of multiple-input change asynchronous machines using controlled excitation and flip-flops. IEEE Transactions on Computers, pp. 1103-1109.

Curtis, H. Allen. 1973. A new type double-rank sequential machine. IEEE Transactions on Computers, pp. 796-803.

Even, Shimon, and Albert R. Meyer. 1969. Sequential boolean equations. IEEE Transactions on Computers, pp. 230-240.

Friedman, A. D., and P. R. Menon. 1968. Synthesis of asynchronous sequential circuits with multiple-input changes. IEEE Transactions on Computers, pp. 559-566.

Gill, A. 1962. Introduction to the theory of finite state machines. McGraw-Hill, New york. 207 p.

Ginsburg, S. 1962. An introduction to mathematic machine theory. Addison-Wesley, Reading, Massachussetts. 148 p.

Harrison, Michael A. 1965. Introduction to switching and automata theory. McGraw-Hill, New York. 499 p.

Hartmanis, J., and R. E. Stearns. 1966. Algebraic structure theory of sequential machines. Prentice-Hall, New Jersey. 211 p.

Hennie, Frederic C. 1968. Finite state models for logical machines. John Wiley & Sons, New York. 466 p.

Hill, Frederick T., and Gerald R. Peterson. 1968. Introduction to switching theory and logical design. John Wiley & Sons, New York. 449 p.

Huffman, David A. 1954. The synthesis of sequential switching circuits. Journal of the Franklin Institute 257(3):161-190, 257(4):275-303.

Hurley, Richard B. 1961. Transistor logic circuits. John Wiley & Sons, New York. 363 p.

Langdon, Glen G. Jr. 1968. Analysis of asynchronous circuits under different delay assumptions. IEEE Transactions on Computers, pp. 1131-1143.

Liu, C. N. 1963. A state variable assignment method for asynchronous sequential switching circuits. Journal of the Association for Computing Machinery 10:209-216.

Maki, Gary K., and James H. Tracey. 1970. State assignment selection in asynchronous sequential circuits. IEEE Transactions on Computers, pp. 641-644.

Maki, Gary K., James H. Tracey, and Robert J. Smith, II. 1969. Generation of design equations in asynchronous sequential circuits. IEEE Transactions on Computers, pp. 467-472.

Mano, Morris M. 1971. Computer logic design. Prentice-Hall, New Jersey. 450 p.

Marcus, Mitchell P. 1962. Switching circuits for engineers. Prentice-Hall International, London. 296 p.

Marcus, Mitchell P. 1969. S-R-T flip-flop. IEEE Transactions on Computers, pp. 568-569.

McCluskey, E. J. 1965. Introduction to the theory of switching circuits. McGraw-Hill, New York. 318 p.

McIntosh, M. D., and B. L. Weinberg. 1969. On asynchronous machines with flip-flops. IEEE Transactions on Computers, p. 473.

Mealy, George H. 1955. A method for synthesizing sequential circuits. The Bell System Technical Journal 34:1045-1079.

Miller, Raymond. 1965. Switching theory, vol. II: sequential circuits and machines. John Wiley & Sons, New York. 250 p.

Millman, J., and H. Taub. 1965. Pulse, digital, and switching waveforms. McGraw-Hill, New York. 958 p.

Moore, Edward F. 1956. Gedanken-experiments on sequential machines. Found in Automata Studies, by C. E. Shannon and J. McCarthy, editors. Princeton University Press, pp. 129-153.

Muller, David E. 1959. Treatment of transition signals in electronic switching circuits by algebraic methods. IRE Transactions on Computers, p. 401.

Muller, David E. 1967. The general synthesis problem for asynchronous digital networks. IEEE Conference Record of Eighth Annual Symposium on Switching, Automata Theory. (October, 1967), pp. 71-82.

Muller, D. E., and W. Scott Bartky. 1956. A theory of asynchronous circuits I. Report no. 75, University of Illinois, Digital Computer Laboratory. 16 p.

Muller, D. E., and W. Scott Bartkey. 1957. A theory of asynchronous circuits II. Report no. 78, University of Illinois, Digital Computer Laboratory. 41 p.

Paull, M. C., and S. H. Unger. 1959. Minimizing the number of states in incompletely specified sequential switching functions. IRE Transactions on Electronic Computers, pp. 356-367.

Peatman, John B. 1972. The design of digital systems. McGraw-Hill, New York. 457 p.

Phister, Montgomery. 1958. Logical design of digital computers. John Wiley & Sons, New York. 408 p.

Saucier, Gabriéle. 1972. State assignment of asynchronous sequential machines using graph techniques. IEEE Transactions on Computers, pp. 282-288.

Sawin, Dwight H. 1974. Optimization of asynchronous sequential circuit realizations. IEEE Transactions on Computers, pp. 186-188.

Servit, Michal. 1973. Hazard correction in asynchronous sequential circuits using inertial delay elements. IEEE Transactions on Computers, pp. 1041-1042.

Singh, Shanker. 1969. Asynchronous sequential circuits with feedback. IEEE Transactions on Computers, pp. 440-450.

Singh, Shanker. 1971. On delayed-input asynchronous sequential circuits. IEEE Transactions on Computers, pp. 500-503.

Smith, John R., Jr., and Charles H. Roth, Jr. 1971. Analysis and synthesis of asynchronous sequential networks using edge-sensitive flip-flops. IEEE Transactions on Computers, pp. 847-855.

Tan, Chung-Jen. 1971. State assignments for asynchronous sequential machines. IEEE Transactions on Computers, pp. 382-391.

Texas Instruments Incorporated. 1973. The TTL data book for design engineers. First edition. Texas Instruments Incorporated, Dallas. 640 p.

Thayse, André, and Marc Davio. 1973. Boolean differential calculus and its application to switching theory. IEEE Transactions on Computers, pp. 409-420.

Torng, H. C. 1964. Introduction to the logical design of switching systems. Addison-Wesley, Reading, Massachussetts. 286 p.

Tracey, James H. 1966. Internal state assignments for asynchronous sequential machines. IEEE Transactions on Computers, pp. 551-560.

Unger, S. H. 1959. Hazards and delays in asynchronous sequential switching circuits. IRE Transactions on Circuit Theory, pp. 17-25.

Unger, Stephen H. 1969. Asynchronous sequential switching circuits. Wiley-Interscience, New York. 290 p.

Unger, Stephen H. 1971. Asynchronous sequential switching circuits with unrestricted input changes. IEEE Transactions on Computers, pp. 1437-1444.

APPENDIXES

Appendix A

Flow Table Reduction

The flow table reduction algorithm presented in Chapter II basically checked the states for incompatibility. If two states were not incompatible, they were considered to be compatible.

A more direct method that gives the same results as the above for fully specified flow tables is to check each pair of states for compatibility, and then draw a merger diagram to choose an optimum reduction. The procedure starts from the primitive flow table. The flow table is checked for redundant states (i.e., states that have the same output, and whose entries are at or go to the same, or equivalent, states). Two stable states are said to be equivalent (and hence redundant) if they are in the same column of the flow table, have the same output, and identical input changes give rise to transitions to the same or equivalent states. "Don't care" entries can be labeled as desired. One row of the primitive flow table can be eliminated for each pair of redundant states found.

After redundant states are eliminated, rows are checked pairwise to see if they can be merged. In the primitive flow table all state changes necessitated a change in the row location. After a flow table is merged it is generally possible to change stable states by merely changing the input without requiring a row change (which means changing one or more of the state variables). The merged flow table will give the desired flow table reduction.

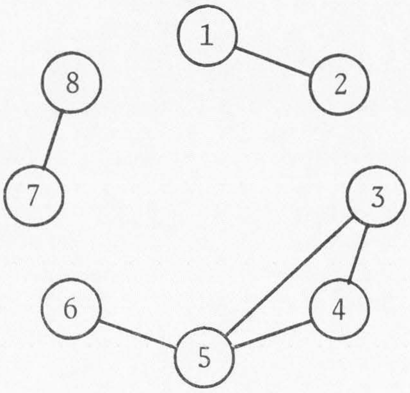
Two or more rows may be merged if there are no conflicting state numbers in any given column of the flow table for the rows in question. A "don't care" condition is considered "wild" and may assume any number to insure compatibility. All of the state numbers in the merging rows are written in the proper column of the merged row, and any entry is circled (to indicate a stable state) if it is circled in any of the merging rows. The outputs (Z) do not affect merging, and may be written alongside the stable states of the merged row, or left off entirely. The primitive flow table can always be referred to when generating the output map.

An optimum merger is generally effected through the use of a merger diagram. The diagram consists of a circular array of the stable states (rows) with lines drawn between those states that may be merged. An inspection of the merger diagram then gives the desired merged, or reduced, flow table. The merging is usually chosen to give the minimum number of rows in the merged flow table.

The flow table of the example problem of Chapter II is merged according the above criteria, and the process steps are shown in Figure 75. The primitive flow table is taken from Figure 2. An inspection of the primitive flow table shows there are no redundant states.

state \ x_1x_2	00	01	11	10	Z
1	①	3	-	2	0
2	1	-	7	②	0
3	4	③	5	-	0
4	④	3	-	6	0
5	-	3	⑤	6	1
6	1	-	5	⑥	1
7	-	3	⑦	8	0
8	4	-	7	⑧	0

Primitive flow table



Merger diagram

merged rows \ x_1x_2	00	01	11	10	Z
1,2	①	3	7	②	0
3,4	④	③	5	6	0
5,6	1	3	⑤	⑥	1
7,8	4	3	⑦	⑧	0

Merged flow table

Figure 75. Example of merging process.

The merger diagram gives two choices for a minimum-row reduction. They are:

$$\pi_1 = \{\overline{1,2};\ \overline{3,4,5};\ \overline{6};\ \overline{7,8}\},$$

$$\text{and } \pi_2 = \{\overline{1,2};\ \overline{3,4};\ \overline{5,6};\ \overline{7,8}\}.$$

A look at the outputs (Z) shows that ⑤ and ⑥ have outputs of "1", and ③ and ④ have outputs of "0". Although both final mergings have the same number of rows (states), it is generally better to group equal-output states together. This generally gives a realization using fewer gates in the final expression for the output (Z). The merged flow table is then seen to be the same as the one generated in Chapter II.

Appendix B

Column Transition Constraints

The requirements for acceptable state transitions within any given column of a flow table for implementation using transition flip-flops were given in Chapter VII. The comment was made that input changes (column-to-column transitions) may cause problems if level-input changes occurred at the time of the enabling transition-input change. This means that at the time the clock input changes in a positive direction (for positive edge-triggered flip-flops), the level inputs should not change (J, K, D, or T).

The acceptable columns of each flip-flop are listed along with an arrow indicating the columns to which a transition may cause problems. If such column-to-column transitions should occur, a check on the input excitation should be made to insure that the transition constraints are met. It should be noted that for flow tables with more than two state variables, transitions within a column may entail transitions between acceptable vertical transition sets. Such transitions should also be checked.

The JK inter-set transitions that may cause problems are as follows:

$$\begin{array}{cccccc} 1 & & 1 & 1 & 1 & 1 \\ 1 & \rightarrow & 3 & 3 & 3 & 3 \\ 3 & & 3 & 3 & 4 & 3 \\ 1 & & 1, & 3, & 4, & 2. \end{array}$$

$$\begin{matrix}1\\1\\3\\2\end{matrix} \rightarrow \begin{matrix}1&1&1&1\\3&3&3&3\\3&3&3&4\\1,&2,&3,&4.\end{matrix}$$

$$\begin{matrix}1\\1\\3\\3\end{matrix} \rightarrow \begin{matrix}1&1&1&1&1&2&2\\1&3&3&3&3&2&2\\3&3&3&3&4&3&3\\1,&1,&2,&3,&4,&1,&2.\end{matrix}$$

$$\begin{matrix}1\\1\\4\\4\end{matrix} \rightarrow \begin{matrix}1&1&1&1&1&2\\1&3&3&3&3&2\\3&3&3&3&4&3\\2,&1,&2,&3,&4,&1.\end{matrix}$$

$$\begin{matrix}1\\3\\3\\1\end{matrix} \rightarrow \text{All transitions allowed.}$$

$$\begin{matrix}1\\3\\3\\2\end{matrix} \rightarrow \text{All transitions allowed.}$$

$$\begin{matrix}1\\3\\3\\3\end{matrix} \rightarrow \begin{matrix}1&1&1&2&2\\1&1&3&2&2\\3&3&3&3&3\\1,&2,&1,&1,&2.\end{matrix}$$

$$\begin{matrix}1\\3\\4\\4\end{matrix} \rightarrow \begin{matrix}1&1&1&1&2&2\\1&1&3&3&2&2\\3&3&3&3&3&3\\1,&2,&1,&2,&1,&2.\end{matrix}$$

$$\begin{matrix}2\\2\\3\\1\end{matrix} \rightarrow \begin{matrix}1&1&1&1\\3&3&3&3\\3&3&3&4\\1,&2,&3,&4.\end{matrix}$$

$$\begin{array}{c} 2\\ 2\\ 3\\ 2 \end{array} \rightarrow \begin{array}{cccc} 1 & 1 & 1 & 1\\ 3 & 3 & 3 & 3\\ 3 & 3 & 3 & 4\\ 1, & 2, & 3, & 4. \end{array}$$

$$\begin{array}{c} 2\\ 2\\ 3\\ 3 \end{array} \rightarrow \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 2\\ 1 & 1 & 3 & 3 & 3 & 3 & 2\\ 3 & 3 & 3 & 3 & 3 & 4 & 3\\ 1, & 2, & 1, & 2, & 3, & 4, & 1. \end{array}$$

$$\begin{array}{c} 2\\ 2\\ 4\\ 4 \end{array} \rightarrow \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 2\\ 1 & 1 & 3 & 3 & 3 & 3 & 2\\ 3 & 3 & 3 & 3 & 3 & 4 & 3\\ 1, & 2, & 1, & 2, & 3, & 4, & 2. \end{array}$$

The D inter-set transitions that may cause problems are as follows:

$$\begin{array}{c} 1\\ 1\\ 3\\ 1 \end{array} \rightarrow \begin{array}{cc} 1 & 1\\ 3 & 3\\ 3 & 3\\ 1, & 3. \end{array}$$

$$\begin{array}{c} 1\\ 1\\ 3\\ 3 \end{array} \rightarrow \begin{array}{ccc} 1 & 1 & 2\\ 1 & 3 & 2\\ 3 & 3 & 3\\ 1, & 1, & 1. \end{array}$$

$$\begin{array}{c} 1\\ 1\\ 4\\ 4 \end{array} \rightarrow \begin{array}{ccc} 1 & 1 & 2\\ 3 & 3 & 2\\ 3 & 3 & 3\\ 1, & 3, & 1. \end{array}$$

$$\begin{array}{c} 1\\ 3\\ 3\\ 1 \end{array} \rightarrow \text{All transitions allowed.}$$

$$\begin{matrix}1 & & 1 & 1\\ 3 & \rightarrow & 1 & 3\\ 3 & & 3 & 3\\ 3 & & 1, & 1.\end{matrix}$$

$$\begin{matrix}1 & & 1 & 2\\ 3 & \rightarrow & 3 & 2\\ 4 & & 3 & 3\\ 4 & & 1, & 1.\end{matrix}$$

$$\begin{matrix}2 & & 1 & 1\\ 2 & \rightarrow & 3 & 3\\ 3 & & 3 & 4\\ 1 & & 1, & 4.\end{matrix}$$

$$\begin{matrix}2 & & 1 & 1 & 1\\ 2 & \rightarrow & 1 & 3 & 3\\ 3 & & 3 & 3 & 4\\ 3 & & 1, & 1, & 4.\end{matrix}$$

$$\begin{matrix}2 & & 1 & 1 & 2\\ 2 & \rightarrow & 3 & 3 & 2\\ 4 & & 3 & 4 & 3\\ 4 & & 1, & 4, & 1.\end{matrix}$$

The clocked T inter-set transitions that may cause problems are as follows:

$$\begin{matrix}1 & & 1 & 1\\ 1 & \rightarrow & 3 & 3\\ 3 & & 3 & 4\\ 1 & & 1, & 4.\end{matrix}$$

$$\begin{matrix}1 & & 1 & 1\\ 1 & \rightarrow & 3 & 3\\ 3 & & 3 & 4\\ 2 & & 1, & 4.\end{matrix}$$

$$\begin{matrix}1\\1\\3\\3\end{matrix} \rightarrow \begin{matrix}1&1&1&2\\1&3&3&2\\3&3&4&3\\1,&1,&4,&1.\end{matrix}$$

$$\begin{matrix}1\\1\\4\\4\end{matrix} \rightarrow \begin{matrix}1&1&1&1&2\\1&3&3&3&2\\3&3&3&4&3\\2,&1,&2,&4,&2.\end{matrix}$$

$$\begin{matrix}1\\3\\3\\1\end{matrix} \rightarrow \text{All transitions allowed.}$$

$$\begin{matrix}1\\3\\3\\2\end{matrix} \rightarrow \text{All transitions allowed.}$$

$$\begin{matrix}1\\3\\3\\3\end{matrix} \rightarrow \begin{matrix}1&1&2\\1&3&2\\3&3&3\\1,&1,&1.\end{matrix}$$

$$\begin{matrix}1\\3\\4\\4\end{matrix} \rightarrow \begin{matrix}1&1&2\\1&3&2\\3&3&3\\2,&2,&2.\end{matrix}$$

$$\begin{matrix}2\\2\\3\\1\end{matrix} \rightarrow \begin{matrix}1&1\\3&3\\3&3\\2,&3.\end{matrix}$$

$$\begin{matrix}2\\2\\3\\2\end{matrix} \rightarrow \begin{matrix}1&1\\3&3\\3&3\\2,&3.\end{matrix}$$

$$\begin{array}{c} 2\\ 2\\ 3\\ 3 \end{array} \rightarrow \begin{array}{ccccc} 1 & 1 & 1 & 1 & 2\\ 1 & 3 & 3 & 3 & 2\\ 3 & 3 & 3 & 3 & 3\\ 1, & 1, & 2, & 3, & 1. \end{array}$$

$$\begin{array}{c} 2\\ 2\\ 4\\ 4 \end{array} \rightarrow \begin{array}{cccc} 1 & 1 & 1 & 2\\ 1 & 3 & 3 & 2\\ 3 & 3 & 3 & 3\\ 2, & 2, & 3, & 2. \end{array}$$

The unclocked T flip-flop has no restrictions placed on its inter-set transitions because there is only one input.

The "one's-catching" JK inter-set transitions that may cause problems occur on the negative-going transition of the clock input, and are as follows:

$$\begin{array}{c} 1\\ 2\\ 3\\ 1 \end{array} \rightarrow \begin{array}{ccc} 1 & 1 & 1\\ 3 & 3 & 3\\ 3 & 3 & 4\\ 1, & 4, & 4. \end{array}$$

$$\begin{array}{c} 1\\ 2\\ 3\\ 4 \end{array} \rightarrow \begin{array}{ccccc} 1 & 1 & 1 & 1 & 2\\ 2 & 3 & 3 & 3 & 2\\ 3 & 3 & 3 & 4 & 3\\ 1, & 1, & 4, & 4, & 1. \end{array}$$

$$\begin{array}{c} 1\\ 2\\ 4\\ 4 \end{array} \rightarrow \begin{array}{ccccc} 1 & 1 & 1 & 1 & 2\\ 2 & 3 & 3 & 3 & 2\\ 3 & 3 & 3 & 4 & 3\\ 1, & 1, & 4, & 4, & 1. \end{array}$$

$$\begin{array}{c} 1\\ 3\\ 3\\ 1 \end{array} \rightarrow \text{All transitions allowed.}$$

$$\begin{array}{cccccc}
1 & & 1 & 1 & 2 \\
3 & \to & 2 & 3 & 2 \\
3 & & 3 & 3 & 3 \\
4 & & 1, & 1, & 1.
\end{array}$$

$$\begin{array}{cccccc}
1 & & 1 & 1 & 2 \\
3 & \to & 2 & 3 & 2 \\
4 & & 3 & 3 & 3 \\
4 & & 1, & 1, & 1.
\end{array}$$

$$\begin{array}{cccccc}
2 & & 1 & 1 & 1 \\
2 & \to & 3 & 3 & 3 \\
3 & & 3 & 3 & 4 \\
1 & & 1, & 4, & 4.
\end{array}$$

$$\begin{array}{ccccccc}
2 & & 1 & 1 & 1 & 1 & 2 \\
2 & \to & 2 & 3 & 3 & 3 & 2 \\
3 & & 3 & 3 & 3 & 4 & 3 \\
4 & & 1, & 1, & 4, & 4, & 1.
\end{array}$$

The above listings provide a caution to the designer and point out only possible problem spots. Many of the inter-set transitions have "don't cares" for some or all of their inputs, and a proper covering will alleviate any improper level-input transition at the time of the enabling clock transition.

VITA

David Franklin Cox

Candidate for the Degree of

Doctor of Philosophy

Dissertation: Asynchronous Logic Design with Flip-Flop Constraints

Major Field: Electrical Engineering

Biographical Information:

Personal Data: Born at Alhambra, California, May 14, 1940, son of Loren Elton and Alpha Mabel Hannah Pearl Cole Cox; married Christine Marie Edes October 26, 1968; father of Shaun David and Kelli Christine Cox.

Education: Graduated from Riverside Polytechnic High School in 1958; received Bachelor of Science degree in Electrical Engineering from the University of California at Berkeley in 1966; received Master of Science degree in Electrical Engineering from Stanford University in 1970; candidate for Doctor of Philosophy degree from Utah State University in 1974.

Professional Experience: 1966 to 1971, Electrical Engineer in disc file development at IBM Corp., San Jose, California; 1972 to 1973, teaching assistant at Utah State University.