Chapter 6
Sequential Circuit Aides

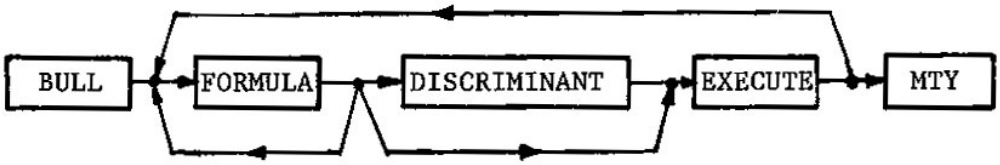
6.1 INTRODUCTION

This chapter describes via examples the use of the program BOOL in the study and solution of sequential circuits.

6.2 DESIGNATION OF FUNCTIONS

LIBRARY:)COPY library-number BOOL
BEGIN: BULL

Sequence of functions:



The printout defines the unknowns by their decimal equivalents. In the case where there is no solution, the program prints solutions belonging to restricted input variables (constants). The corresponding configurations of constants' validities become DON'T CAREs of the solutions. The constant validity configuration for which the unknowns can take on any value is also treated as a DON'T CARE.

For rules concerning the symbolism, see page 12 and Example 8.

The routine EXECUTE modifies the discriminant so that restriction of input variables and other DON'T CAREs

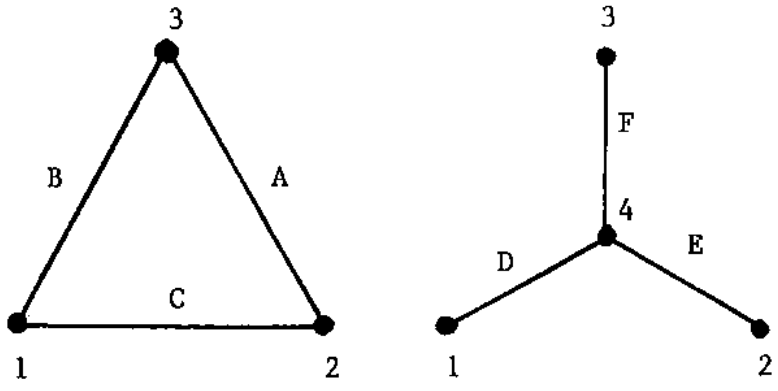
are accomodated. The symbol of the modified discriminant is MTY.

6.3 EXAMPLES

EXAMPLE 6.3.1. A switching network (relay type) contains a triangle of given switching functions: A, B, C. Problem: Find three switching functions D, E, F that transform the tringle into a star (Fig. 6.1) which is equivalent to the triangle within the switching network.

The system of equations is solved by calling:

```
      BULL
NUMBER OF CONSTANTS IS:
□:
      3
SYMBOLS FOR CONSTANTS: ABC
NUMBER OF UNKNOWNS IS:
□:
      3
SYMBOLS FOR UNKNOWNS: DEF
CALL FORMULA.
```



Terminals: 1, 2, 3
Triangle: 1-2; C
2-3; A
3-1; B
Star: 4-1; D
4-2; E
4-3; F

Equations: A + BC = EF
B + CA = FD
C + AB = DE

Fig. 6.1. Switching network of Example 6.3.1.


```
      FORMULA
WRITE DOWN THE FORMULA:
A+BC=EF
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

      FORMULA
WRITE DOWN THE FORMULA:
B+CA=FD
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

      FORMULA
WRITE DOWN THE FORMULA:
C+AB=DE
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

      DISCRIMINANT
THE DISCRIMINANT VALUE BEFORE CONSTRAINTS IS:
 1  0  0  0  0  0  0  0
 1  0  0  0  0  0  0  0
 1  0  0  0  0  0  0  0
 0  0  0  0  1  0  0  0          ← Number of solutions = 4.
 1  0  0  0  0  0  0  0
 0  0  1  0  0  0  0  0
 0  1  0  0  0  0  0  0
 0  0  0  1  0  1  1  1
AFTER ::EXECUTE:: HAS BEEN CALLED, CALL MTY TO GET
THE CONSTRAINT RECTIFIED DISCRIMINANT.
CALL EXECUTE AFTER ALL CONDITIONS HAVE BEEN PUT IN.
IF NOT, CALL FORMULA AGAIN.
```

The number of solutions of the form

$$D = f(A, B, C)$$
$$E = f(A, B, C)$$
$$F = f(A, B, C)$$

is found by finding the number of nonzeros in each column. The discriminant has four nonzeros in the column belonging to CBA = 1 (at the extreme left) and exactly one nonzero in each of the other columns. We expect four solutions (the product of nonzero counts taken for all columns).

After all of the equations have been written in, we call:

```
      EXECUTE
NUMBER OF SOLUTIONS UNDER ALL CONSTRAINTS:  SOL =  4
```

```
SOLUTION VECTOR:
□:
      1 2 3 4
SOLUTION NUMBER: 1
D = [2  3  4  5  6  7] ∪ ()
E = [1  3  4  5  6  7] ∪ ()
F = [1  2  3  5  6  7] ∪ ()
SOLUTION NUMBER: 2
D = [0  2  3  4  5  6  7] ∪ ()
E = [1  3  4  5  6  7] ∪ ()
F = [1  2  3  5  6  7] ∪ ()
SOLUTION NUMBER: 3
D = [2  3  4  5  6  7] ∪ ()
E = [0  1  3  4  5  6  7] ∪ ()
F = [1  2  3  5  6  7] ∪ ()
SOLUTION NUMBER: 4
D = [2  3  4  5  6  7] ∪ ()
E = [1  3  4  5  6  7] ∪ ()
F = [0  1  2  3  5  6  7] ∪ ()
```

The solutions are printed by listing the ONEs of the functions.

Solution No. 1, the simplest, is very well known. The Marquand charts of that solution are shown in Fig. 6.2.

The other solutions are not as well-known. They are less simple:

No. 2: $D = \underline{A} + B + C$ $E = C + A$ $F = A + B$

No. 3: $D = B + C$ $E = \underline{B} + C + A$ $F = A + B$

No. 4: $D = B + C$ $E = C + A$ $F = \underline{C} + A + B$

EXERCISE: Solve the last example by using the program SYSTEM.

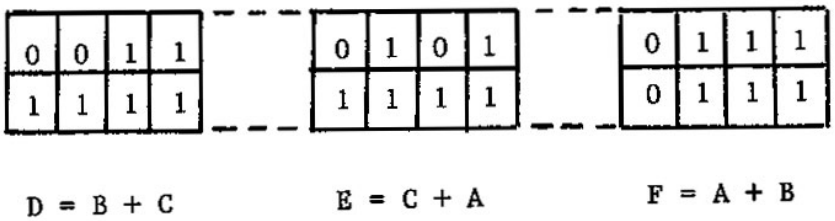


Fig. 6.2. Marquand map of Example 6.3.1, solution No. 1.

EXAMPLE 6.3.2. Find all of the steady states, as well as the non-steady-state input configurations, of the sequential circuit shown in Fig. 6.3.

The steady state must satisfy the system of Boolean equations:

$$D = \text{NOR}(A,E) = \underline{A}\ \underline{E}$$
$$E = \text{AND}(B,F) = B\ F$$
$$F = \text{OR}(C,D) = C + D$$

The computer-aided solution using BOOL is found by calling:

```
      BULL
NUMBER OF CONSTANTS IS:
□:
      3
SYMBOLS FOR CONSTANTS: ABC
NUMBER OF UNKNOWNS IS:
□:
      3
SYMBOLS FOR UNKNOWNS: DEF
CALL FORMULA.
```

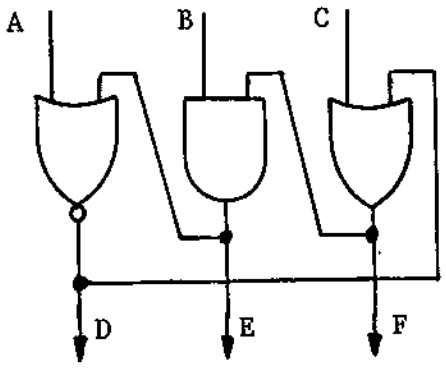


Fig. 6.3. Sequential circuit of Example 6.3.2.

```
       FORMULA
WRITE DOWN THE FORMULA:
D=AE
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

       FORMULA
WRITE DOWN THE FORMULA:
E=BF
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

       FORMULA
F=C+D
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
```

All equations are in. The existence function of the circuit is obtained by calling:

```
       DISCRIMINANT
THE DISCRIMINANT VALUE BEFORE CONSTRAINTS IS:
 0 1 0 1 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0
 1 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0
AFTER ::EXECUTE:: HAS BEEN CALLED, CALL MTY TO GET
THE CONSTRAINT RECTIFIED DISCRIMINANT.
CALL EXECUTE AFTER ALL CONDITIONS HAVE BEEN PUT IN.
IF NOT, CALL FORMULA AGAIN.
```

Again, the number of solutions of the form

$$D = f(C,B,A)$$
$$E = g(C,B,A)$$
$$F = h(C,B,A)$$

is equal to the product of the number of nonzeros taken for every column:

$$SOL = 1 \times 1 \times 0 \times 1 \times 1 \times 1 \times 1 \times 1$$
$$= 0$$

By constraining the input variables C, B, and A so that the configuration of their validities corresponding to a column filled only by zeros is forbidden, that is $(C\ B\ A)_2 = 2$ (or the $\underline{C}\ B\ \underline{A}$ minterm), exactly one solution

is possible. It is interesting that the circuit will behave as a combinational one, as long as the input configuration (C = A = 0, B = 1) is excluded. For this input signal configuration, the circuit will race (oscillate).

In the solution, the forbidden input state is in round parentheses:

```
      EXECUTE
THE NUMBER OF SOLUTIONS IS ZERO, UNLESS
THE FOLLOWING INPUT IDENTIFIERS ARE FORBIDDEN:  2
NUMBER OF SOLUTIONS UNDER ALL CONSTRAINTS:  SOL =  1
SOLUTION VECTOR:
□:
      1
SOLUTION NUMBER: 1
D = [0 4] ∪ (2)
E = [6 7] ∪ (2)
F × [0  4  5  6  7] ∪ (2)
```

The value in parentheses can be proclaimed to be a DON'T CARE because the circuit is not permitted to use it. Then the solution is:

$$D = \underline{A}\ \underline{B}, \qquad E = B\ C, \qquad F = C + \underline{A}$$

The program block BOOL can be used for computer-aided design of sequential circuits. The design procedure mentioned here can be applied to any structural type (model). That means the design of circuits with or without memory elements in the feedback loop, clocked as well as non-clocked circuits, can be assisted by these programs. Roughly speaking, the procedure formulates engineering conditions (constraints) of the circuit in the form of Boolean equations and designs the combinational network by solving that system of equations.

Two examples are offered here. Both illustrate the design of a J-K flip-flop with the trailing edge control. The first example has no memory elements within the feedback loop; the second has memory elements.

EXAMPLE 6.3.3. Design a non-clocked J-K flip-flop with trailing edge control and without memory elements (latches) in the feedback loop.

The type of the circuit is shown in Fig. 6.4. The control (input) variables are A and $\bar{B}$. The feedback loop is shown interrupted to permit formulations of constraints in the form of cause-effect relation (implication or equivalence). The split in the feedback permits the use of different variables on each side of the split: C, D; E, F. The variables C, D belong to the inputs of the combinational circuit; E, F belong to its outputs. When the feedbacks are not interrupted, then C must take on the signal level of E (E represents the future signal level of C) and D must take on the signal level of F.

Fig. 6.5 will help us formulate the constraints of the system. The diagram resembles the conventional state diagram and is not a required step in finding a solution.

There are fundamental principles ruling the cause-effect relationship in physics which must be respected to get satisfactory results:

EACH CONSTRAINT MUST BE FORMULATED IN SUCH A WAY THAT THE CAUSE-EFFECT LOGICAL RELATION DEFINES THE TRANSITION OF THE SYSTEM UNIQUELY.

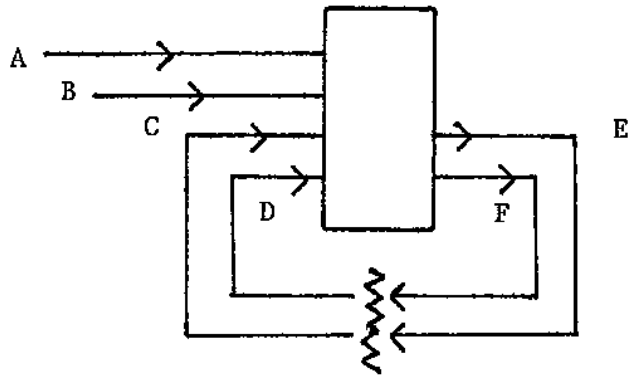


Fig. 6.4. Cause-effect chain.

Explanation: Let us suppose that the system in Fig. 6.4 is in the state: $S_1 \equiv \{A_1, B_1, C_1, D_1, E_1, F_1\}$ (the feedback is interrupted!). Assuming that the control A, B is constant and that the structure of the combinational circuit is known, let us inquire about the transition of the circuit if the feedback is not interrupted. The answer can be formulated by a timeless (purely logical) proposition only if $(E_1 = C_1) \vee (F_1 = D_1)$. For instance, when $E_1 = C_1$ but $F_1 \neq D_1$, we say simply $D_2 \neq D_1$ (in words: Only D changes the signal level).

Similarly, "only C changes the signal level" is a proposition that describes completely the transition of the system when $F_1 = D_1$.

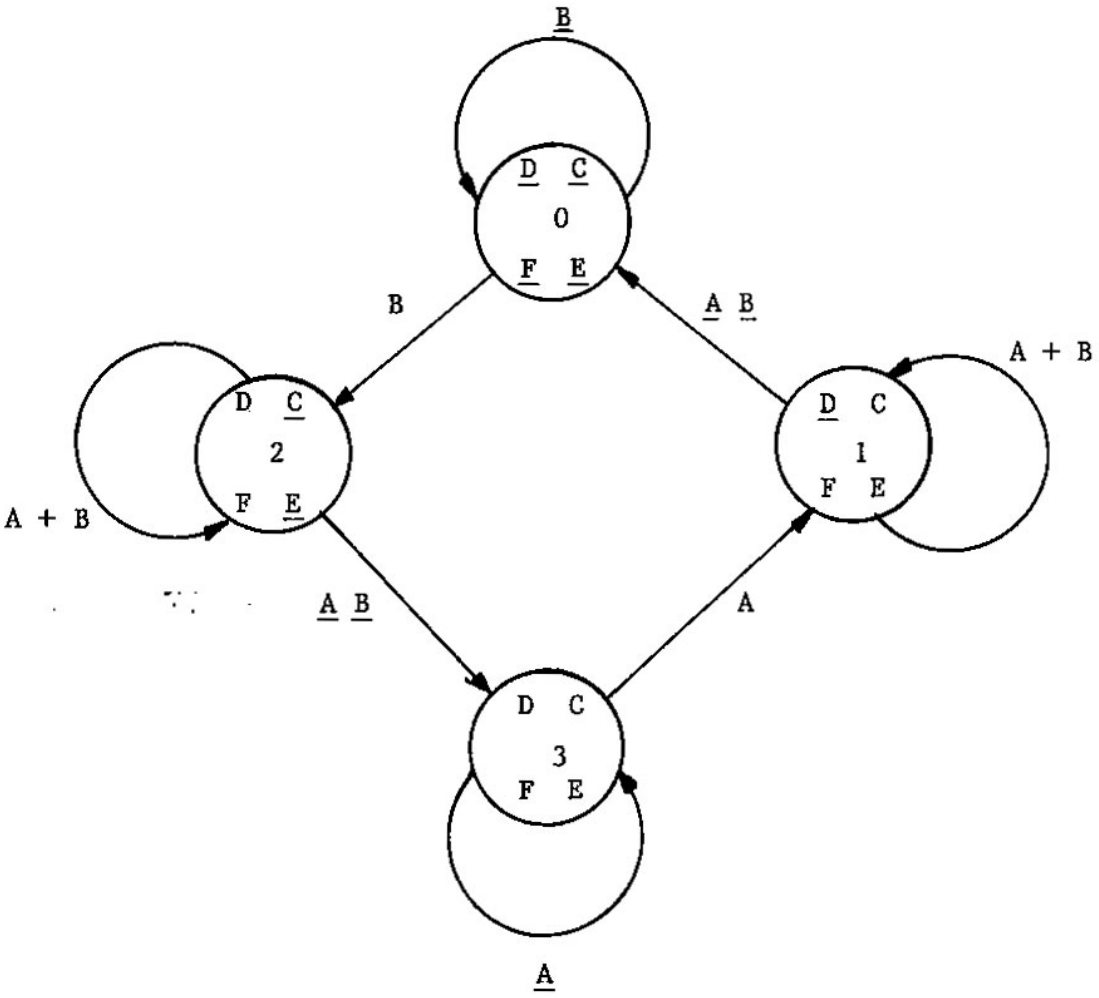


Fig. 6.5. Transition diagram.

If we suppose, however, that $(E_1 \neq C_1)$ AND $(F_1 \neq D_1)$, we are unable to describe the transition by a timeless proposition. Due to physical time delays, the variation in two signal levels will not occur at the same (physical) time.

Definition: The transition of a system is called CONTINUOUS IN LOGICAL TIME if and only if not more than one variable changes during the transition.

By using the continuity notion, the rules for constraint formulation are:

1. The transition induced by a constraint must be continuous in logical time.
2. The control variables may change only when the system is in a steady state.
3. Every transition belonging to a race must be continuous in logical time.

Two feedback loops are suggested in Fig. 6.4. For that reason, four (2^2) possible states of the feedback are at the engineer's disposal. They are represented by circles (Fig. 6.5); identified by state numbers 0, 1, 2, 3; or by state binary identifiers: 00, 01, 10, 11. Binary identifiers indicate the values of the variables of the feedback by the rule

$$i = (D\ C)_2 = (F\ E)_2 \text{ for } i = 0,1,2,3.$$

The principle of continuity in logical time excludes the transitions: $0 \rightarrow 3$, $3 \rightarrow 0$, $1 \rightarrow 2$, and $2 \rightarrow 1$, in which two variables change at the same time. For that reason, the circles of the transition diagram are usually sketched in a circular way and numbered by the identifier i in a sequence where only one bit of the binary identifier changes with each step around the circular way. In Fig. 6.5, the sequence in i is 0, 1, 3, 2 (Gray code!).

The transitions are represented in Fig. 6.5 by directional lines (transition arrows) connecting the

circles representing the states of the transition. It has been said that the binary identifiers of these two states may disagree in a single bit (or not at all when the control has no effect on the feedback). The transition arrow can go back to the circle from which it originated (steady state).

The design procedure is started by choosing a number n of links in the feedback. That makes 2^n distinct states of the feedback loop available to the designer. The designer then tries to establish a transition pattern of the circuit so that all constraint formulation rules are respected. Success occurs when there is enough "elbow room" within the state diagram. If not, a link is added to the feedback loop of the circuit, and the constraint formulation is tried again.

In our example, the designer is faced with the trailing edge control condition. This means that the output signals, whether true or false, must occur only after both control signal variables (A, B) have returned to zero; i.e., when $\underline{A} \cdot \underline{B} = 1$.

The design will be simpler if one of the feedback links is used to produce the output signal variable. By choosing the link C-E for that purpose, we observe that the control signal $\underline{A} \cdot \underline{B}$ must operate with one of the transitions:

1. $(D\ C)_2 = 0\ 1 \leftrightarrow 0\ 0 \equiv 1 \leftrightarrow 0$
2. $(D\ C)_2 = 1\ 1 \leftrightarrow 1\ 0 \equiv 3 \leftrightarrow 2$

The sense of the transition is now chosen. By choosing $1 \rightarrow 0$ or $2 \rightarrow 3$, it is clear that the final state is either 0 or 3 (see Fig. 6.5).

The roles of the control signal variables are now chosen. If we decide that A = RESET, the output signal is toggled FALSE in state 0. B ≡ SET results in the output signal being toggled TRUE in state 3.

Table 6.1. Constraint formulations for Example 6.1.3.

	Formulation	Algebraic notation
1.	For idle control ($\underline{AB} = 1$), remain steady at $i = 1$ ($\underline{DC} = 1$). Constraint: IF ($\underline{AB} = 1$ and $\underline{DC} = 1$) THEN ($\underline{EF} = 1$).	$\underline{A}\ \underline{B}\ \underline{C}\ \underline{D} \rightarrow \underline{E}\ \underline{F}$
2.	For RESET: $\underline{AB} = 1$, remain steady at $i = 0$ ($\underline{DC} = 1$). IF ($\underline{AB} = 1$ and $\underline{DC} = 1$) THEN ($\underline{EF} = 1$).	$A\ \underline{B}\ \underline{C}\ \underline{D} \rightarrow \underline{E}\ \underline{F}$
3.	For control ($\underline{AB} = 1$), e.g., input B high alone at 0, initiate transition from 0 to 2 by causing the output variables to change so that ($\underline{FE} = 1$). IF ($\underline{AB} = 1$ and $\underline{DC} = 1$) THEN ($\underline{FE} = 1$).	$\underline{A}\ B\ \underline{C}\ \underline{D} \rightarrow \underline{E}\ F$
4.	For a J-K flip-flop, the same transition must be initiated for the control (AB = 1) (both inputs high). IF (AB = 1 and $\underline{D}\ \underline{C} = 1$) THEN ($\underline{FE} = 1$).	$A\ B\ \underline{C}\ \underline{D} \rightarrow \underline{E}\ F$
5.	After the transition caused by either of the previous constraints, remain steady at $i = 2$ ($\underline{DC} = 1$). (Note that $\underline{A}B + A\underline{B} = B$.) IF ($\underline{DC} = 1$ and $B = 1$) THEN ($\underline{FE} = 1$).	$B\ \underline{C}\ D \rightarrow \underline{E}\ F$
6.	To satisfy the trailing-edge condition, the circuit must wait at $i = 2$ until both variables A and B return to zero. When B returns to zero sooner than A, the state of the control will be $\underline{AB} = 1$, and the circuit must remain at $i = 2$. IF ($\underline{AB} = 1$ and $\underline{CD} = 1$) THEN ($\underline{EF} = 1$).	$A\ \underline{B}\ \underline{C}\ D \rightarrow \underline{E}\ F$

Table 6.1 (Continued).

	Formulation	Algebraic notation
7.	Trailing edge $\underline{AB}$ = 1 when the system is at i = 2 causes transition 2 → 3 by making EF = 1. IF ($\underline{AB}$ = 1 and $\underline{CD}$ = 1) THEN (EF = 1).	$\underline{A}\ \underline{B}\ \underline{C}\ D \rightarrow E\ F$
8.	For idle control ($\underline{AB}$ = 1), remain steady at i = 3. IF ($\underline{AB}$ = 1 and CD = 1) THEN (EF = 1).	$\underline{A}\ \underline{B}\ C\ D \rightarrow E\ F$
9.	For SET ($\underline{AB}$ = 1), remain steady at i = 3 (CD = 1).	$\underline{A}\ B\ C\ D \rightarrow E\ F$
10.	For control $\underline{AB}$ = 1 (input A high alone) at i = 3, initiate transition 3 → 1 by causing $\underline{EF}$ = 1.	$A\ \underline{B}\ C\ D \rightarrow E\ \underline{F}$
11.	Initiate the same transition at i = 3 for the control AB = 1 (both inputs high).	$A\ B\ C\ D \rightarrow E\ \underline{F}$
12.	After a transition caused by either of the two previous constraints, remain steady at i = 1 ($\underline{CD}$ = 1).	$A\ C\ D \rightarrow E\ \underline{F}$
13.	To satisfy the trailing-edge condition, wait at i = 1 until $\underline{AB}$ = 1. When A returns to zero sooner than B, the state of the control will be $\underline{AB}$ = 1 but the circuit must stay at i = 1.	$\underline{A}\ B\ C\ \underline{D} \rightarrow E\ \underline{F}$
14.	Trailing edge $\underline{AB}$ = 1 when the system is at i = 1 causes transition 1 → 0 by making $\underline{EF}$ = 1.	$\underline{A}\ \underline{B}\ C\ \underline{D} \rightarrow \underline{E}\ \underline{F}$

Keeping in mind that, as a J-K flip-flop, the circuit must always go from TRUE to FALSE (or vice-versa) when both inputs are stimulated (for AB = 1), we are ready to formulate the constraints. Refer to Table 6.1.

Implications that have identical expressions on their right sides can be merged into one implication. For instance, the constraints

1, 2, 14 merge into:

$$\underline{A}\ \underline{B}\ \underline{C}\ \underline{D} + A\ \underline{B}\ \underline{C}\ \underline{D} + \underline{A}\ \underline{B}\ C\ \underline{D} \rightarrow \underline{E}\ \underline{F}$$

$$\equiv \underline{B}\ \underline{C}\ \underline{D} + \underline{A}\ \underline{B}\ \underline{D} \rightarrow \underline{E}\ \underline{F}$$

3, 4, 5, 6 merge into:

$$\underline{A}\ B\ \underline{C}\ \underline{D} + A\ B\ \underline{C}\ \underline{D} + A\ \underline{B}\ \underline{C}\ D + B\ \underline{C}\ D \rightarrow \underline{E}\ F$$

$$\equiv A\ \underline{C}\ D + B\ \underline{C} \rightarrow \underline{E}\ F$$

7, 8, 9 merge similarly into:

$$\underline{A}\ \underline{B}\ D + \underline{A}\ C\ D \rightarrow E\ F$$

10, 11, 12, 13 merge finally into:

$$A\ C + B\ C\ \underline{D} \rightarrow E\ \underline{F}$$

To design the combinational network, call BULL. Note that the outputs E, F are sought as Boolean functions of the inputs A, B, C, D.

```
      BULL
NUMBER OF CONSTANTS IS:
□:
      4
SYMBOLS FOR CONSTANTS: ABCD
NUMBER OF UNKNOWNS IS:
□:
      2
SYMBOLS FOR UNKNOWNS: EF
CALL FORMULA.
```

The FORMULA call is repeated once for each of the four implications:

```
      FORMULA
WRITE DOWN THE FORMULA:
BCD+ABD→EF
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
```



```
      FORMULA
WRITE DOWN THE FORMULA:
AD_C_+B_C_→F_E_
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

      FORMULA
WRITE DOWN THE FORMULA:
D_AB_+DC_A_→EF
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.

      FORMULA
WRITE DOWN THE FORMULA.
AC+BC_D_→E_F_
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
```

If the DISCRIMINANT has a column full of zeros, no solution exists unless the set of input signal configurations for that column is forbidden. A column full of ones means "don't care" for the corresponding input signal configuration. In both cases, the discriminant is modified to produce solutions with don't cares for input configurations for columns with either all ones or all zeros (these are listed within the round parentheses of the printout).

```
      DISCRIMINANT
THE DISCRIMINANT VALUE BEFORE CONSTRAINTS IS:
 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1
 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0
AFTER ::EXECUTE:: HAS BEEN CALLED, CALL MTY TO GET
THE CONSTRAINT RECTIFIED DISCRIMINANT.
CALL EXECUTE AFTER ALL CONDITIONS HAVE BEEN PUT IN.
IF NOT, CALL FORMULA AGAIN.
```

In the present case, the discriminant has none of the singularities previously mentioned. Each column contains exactly one non-zero. Exactly one solution exists for our problem:

```
      EXECUTE
NUMBER OF SOLUTIONS UNDER ALL CONSTRAINTS:  SOL =  1
SOLUTION VECTOR:
□:
      1
SOLUTION NUMBER: 1
E = [5  6  7  8  12  13  14  15] ∪ ()
F = [2  3  8  9  10  11  12  14] ∪ ()
```

The following steps are recommended from this point on:

1. Find N-minimal ΣΠ-forms of both (all) output functions (here, E and F).
2. Find all prime implicants of those functions.
3. Design the circuit.
4. Draw DYNAMIC SCHEMATIC to look for hazards.
5. Go through exhaustive hazard analysis.
6. Eliminate hazards.

The program block SYSTEM is used to complete steps 1 and 2.

```
      LOGIC
NUMBER OF X-VARIABLES:
□:
      4
NX= 4
SYMBOLS FOR X-VARIABLES: (X J), (X̲ J); J=1  2  3  4
CALL:  TABLE, SPACE
      TABLE
 NUMBER OF FUNCTIONS:
□:
      2
TABLE IS READY FOR FUNCTIONS  (Z K)  WITH K= 1  2
CALL OFFERINGS: FTRUE, FFALSE, FLIST
      FLIST
 DECIMAL EQUIVALENTS OF ONES (AT LEAST ONE ITEM):
□:
      5 6 7 8 12 13 14 15
 DECIMAL EQUIVALENTS OF DONT CARES:
□:
      ι0
 CALL:  FSTOR  K (WHERE K IS WELL SPECIFIED)
      FSTOR 1
 CALL:  FTRUE,  FFALSE,  FLIST  TO DEFINE THE NEXT
      FUNCTION  (F K)  WITH  K=2
      FLIST
 DECIMAL EQUIVALENTS OF ONES (AT LEAST ONE ITEM):
□:
      2 3 8 9 10 11 12 14
 DECIMAL EQUIVALENTS OF DONT CARES:
□:
      ι0
 CALL:  FSTOR  K (WHERE K IS WELL SPECIFIED)
      FSTOR 2
 TRUTH TABLE IS READY:
MAY CALL:  FTRUE, FFALSE,  FLIST FOR  K= 3 ≤2
MAY EXECUTE  FX  TO PRINT THE TABLE
```


The first row of the truth table contains (Z 1), designated by E in Fig. 6.4. The second row contains (Z 2), designated by F.

```
      FX
 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1
 0 0 1 1 0 0 0 0 1 1 1 1 1 0 1 0
```

To minimize, we call:

```
      MINIMA (Z 1)
AB̲D + BC + AC
CRITICAL SET: 8  6  5

      MINIMA (Z 2)
AD̲ + CD̲ + BC̲
CRITICAL SET: 14  9  3

      PRIMIMPLICANT (Z 1)
ALGEBRAIC FORM OF THE COMPLEMENTARY FUNCTION:
AB̲D̲ + BC̲ + AC̲
CRITICAL SET: 4  10  9
SUM OF ALL PRIME IMPLICANTS:
AC + BC + AB̲D + CD

      PRIMIMPLICANT (Z 2)
ALGEBRAIC FORM OF THE COMPLEMENTARY FUNCTION:
AC + CD̲ + B̲D̲
CRITICAL SET: 15  6  1
SUM OF ALL PRIME IMPLICANTS:
BC̲ + AD̲ + CD̲

      MINIMA (Z̲ 1)
AB̲D̲ + BC̲ + AC̲
CRITICAL SET: 4  10  9

      MINIMA (Z̲ 2)
AC + CD̲ + B̲D̲
CRITICAL SET: 15  6  1

      PRIMIMPLICANT (Z̲ 1)
ALGEBRAIC FORM OF THE COMPLEMENTARY FUNCTION:
AB̲D + BC + AC
CRITICAL SET: 8  6  5
SUM OF ALL PRIME IMPLICANTS:
AC̲ + BC̲ + AB̲D̲ + CD̲

      PRIMIMPLICANT (Z̲ 2)
ALGEBRAIC FORM OF THE COMPLEMENTARY FUNCTION:
```

$\underline{A}D + \underline{C}D + B\underline{C}$
CRITICAL SET: 14 9 3
SUM OF ALL PRIME IMPLICANTS:
$AC + B\underline{D} + C\underline{D}$

The shematic of the sequential circuit (step 4) is now drawn, preferably in the form called DYNAMIC SCHEMATIC (Fig. 6.6). Let us start with the variant implementing MINIMA (Z 1) and MINIMA (Z 2). There is one vertical line

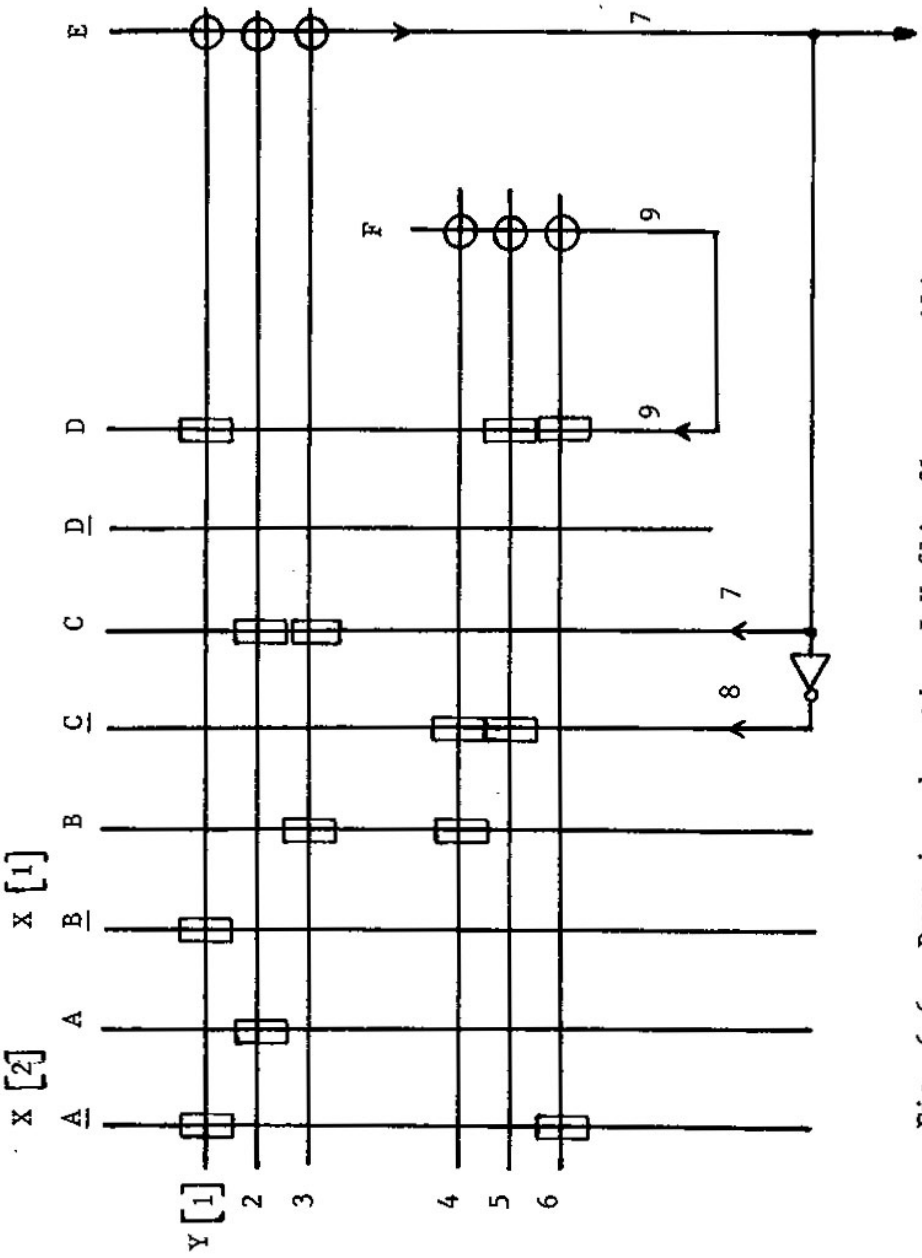


Fig. 6.6. Dynamic schematic: J-K flip-flop - trailing edge controlled, input inverters not considered.

drawn to represent each of the variables A, $\underline{A}$, B, $\underline{B}$, C, $\underline{C}$, D, $\underline{D}$, E, and F. Lines for $\underline{E}$ and $\underline{F}$ are not drawn since the expressions do not include them:

$$E = \underline{AB}D + BC + AC, \quad F = \underline{A}D + \underline{C}D + B\underline{C}.$$

A horizontal line is drawn for each term appearing on the right side of the expressions. In the two-level AND-OR implementation, each line will belong to an AND gate. The input signals entering such an AND gate are represented by markings in the form of small rectangles (for instance, the horizontal line at the top stands for an AND gate with inputs $\underline{A}$, $\underline{B}$, D). Vertical lines E and F, representing the output signals of the combinational network, also represent the corresponding output OR gate. Small circular markings designate which AND gate outputs are used as the OR gate inputs. In Fig. 6.6, the OR gate producing E is entered by the outputs of three AND gates (lines 1, 2, and 3).

In combination with a set of small rods*, a graphic-mechanical working model of the sequential circuit can be created that is well suited for use in analyzing the activity of the circuit. When a rod is placed to cover a vertical line representing a variable (for instance, $\underline{A}$), it means that that particular variable is ON (high). In Fig. 6.7, assume that four rods are placed to cover vertical lines $\underline{A}$, $\underline{B}$, C, and D. Some of the square markers become masked, some remain unmasked. All markings of the horizontal line No. 1 are masked to indicate that all inputs of the AND gate represented by the horizontal line are ON so that the output of the AND gate must be ON. Similarly, the output of the AND gate represented by line

---

* In the classroom, coffee stirrers have been used for this purpose.

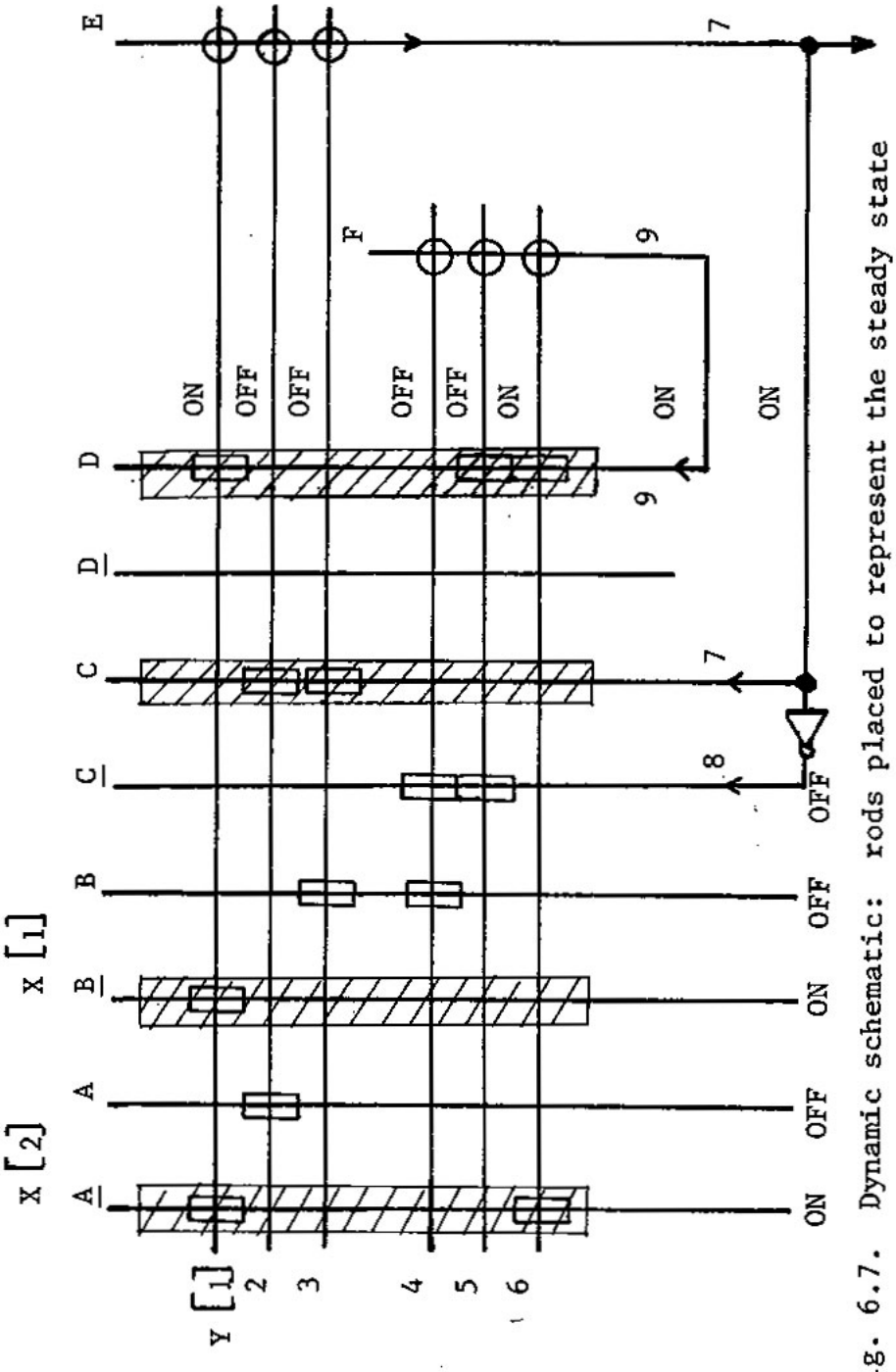


Fig. 6.7. Dynamic schematic: rods placed to represent the steady state A B C D E F = 1. Note the the variables A, B, C are represented by two wires: A, A; B, B; C, C. The transition from A to A in the case shown will cause a hazard if the wire A becomes low sooner than A becomes high. In that case E could become illegally low.

No. 6 is ON because all markers of that line are masked. Lines No. 2, 3, 4, and 5 are OFF because at least one square marker is visible in incidence with each of these lines. One of the inputs of the OR gate belonging to the output E being ON (vertical line No. 7), therefore E is ON. If E is ON then C is also ON but line No. 8 is OFF due to the inverter. Similarly, the OR gate belonging to the output F has one input ON so that line No. 9 must be ON. The model of the circuit represents the state of the circuit designated by i = 3 in Fig. 6.5 with idle control ($\underline{A}\underline{B}$ = 1).

The dynamic schematic can be used in this way to analyze the performance of the circuit perfunctorily or with gradually increasing care about detailed impulse-timing information.

The simplest use of the dynamic schematic is the checking of equations. Transitions between states can be observed without trying to discover hazards. For instance, starting with the state shown in Fig. 6.6, we want to observe the transition due to the change of the control from $\underline{A}\ \underline{B}$ to $A\ \underline{B}$. To do it, the rod that masks the vertical line $\underline{A}$ is moved to mask the line A. Now the schematic is inspected: Horizontal line No. 1 now shows a rectangular marker so that the output of the corresponding AND gate must be OFF (it was ON). All markers of horizontal line No. 2 are masked so that the output of the corresponding AND gate must be ON (it was OFF). Horizontal line No. 6 shows a marker, so its AND gate's output is OFF (it was ON). The OR gate whose output is line No. 9 has all inputs OFF. For that reason, line No. 9 is OFF (it was ON). The rod masking line No. 9 must be moved to mask the verical line $\underline{D}$. The last motion causes no change in the ON-OFF signal distribution. The circuit is stable with $C\underline{D}$ = $E\underline{F}$ = 1 after correct transition with i = 3 → 1.

The animation of the dynamic schematic can be used to detect hazards. Let us go through the preceding experiment more carefully. The motion of the rod from A to A caused a change in two lines at the same time: Line No. 1 went OFF, and line No. 2 went ON. Those two lines represent two AND gate outputs; the reaction times (time delay) of those gates are always different; thus, we must consider the case where AND gate No. 2 is very much slower than AND gate No. 1. In that case, line No. 1 will be OFF long before line No. 2 goes ON. The output of the OR gate controlling line No. 7 will go OFF. To represent that fact by animation, the rod masking line C (≡ No. 7) must be removed. Another marker of line No. 2 becomes unmasked so that the line will be prevented from going ON. The result represents a hazard: The circuit went through i = 3 → 2 → 0 or through i = 3 → 1 → 0, depending on the reaction time of the AND gate represented by line No. 6. The final state of both transitions is spurious. The designed version of the circuit has a hazard for the transition considered.

Input signal inverters are added in Fig. 6.8. This schematic should be animated in a way that shows the role of the delay in the inverters. EXERCISE: Show that the signal B inverter induces hazards.

Fig. 6.9 illustrates ways to eliminate hazards. The hazard due to the difference in reaction times between the AND gates implementing the terms A B D and A C of the expression E = ABD + BC + AC is eliminated by adding the term CD (a prime implicant of E) to that expression so that E = CD + ABD + BC + AC (see the result of PRIMIMPLICANT (Z 1) !), and by adding a corresponding AND gate to the circuit (line No. 1 in Fig. 6.9). The output of this gate remains ON during the transition i = 3 → 1 so that the signal changes at the outputs of the gates corresponding to the terms ABD and AC do not influence the output

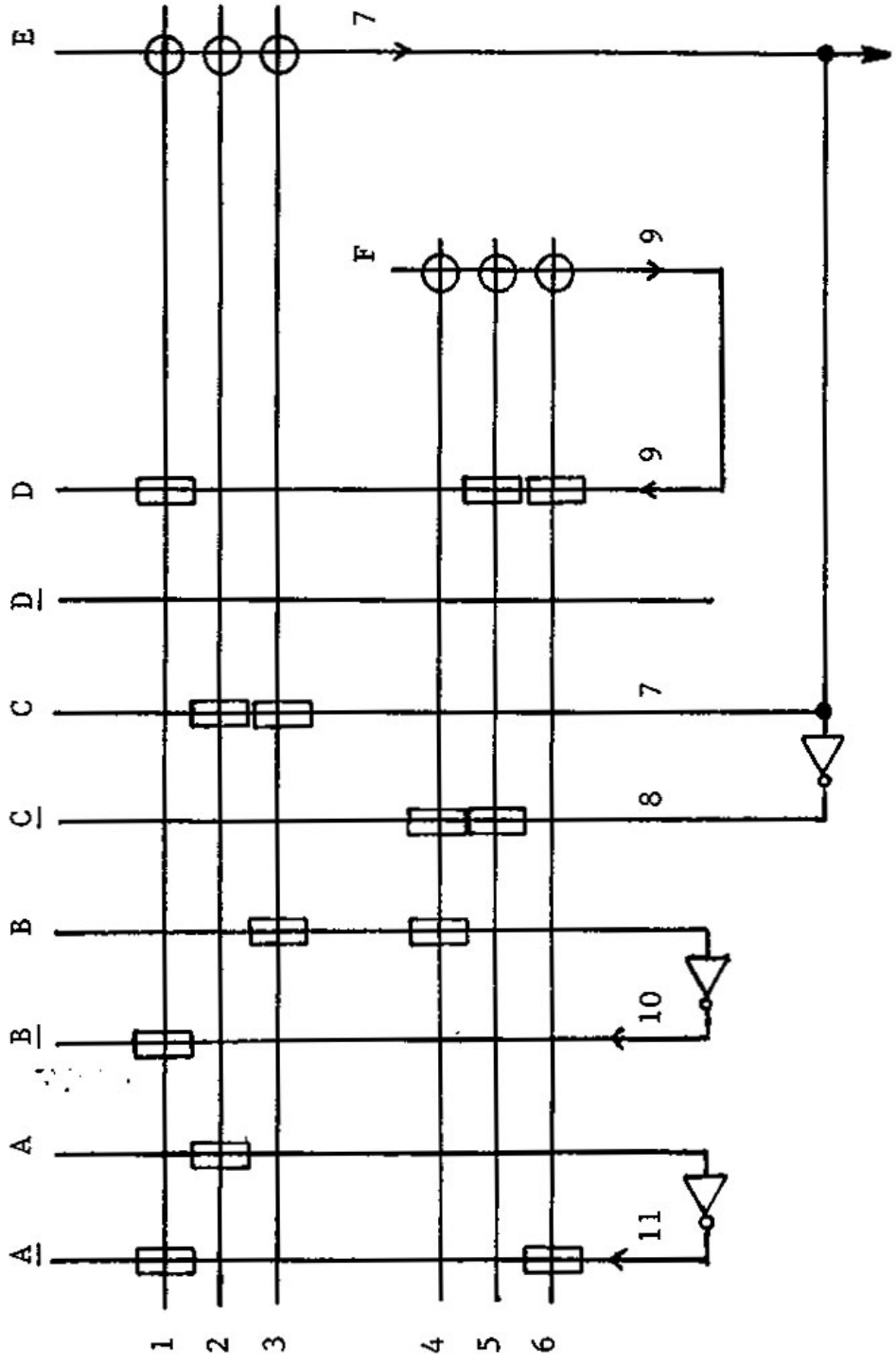


Fig. 6.8. Dynamic schematic: J-K flip-flop - trailing edge controlled, input inverters induce hazards.

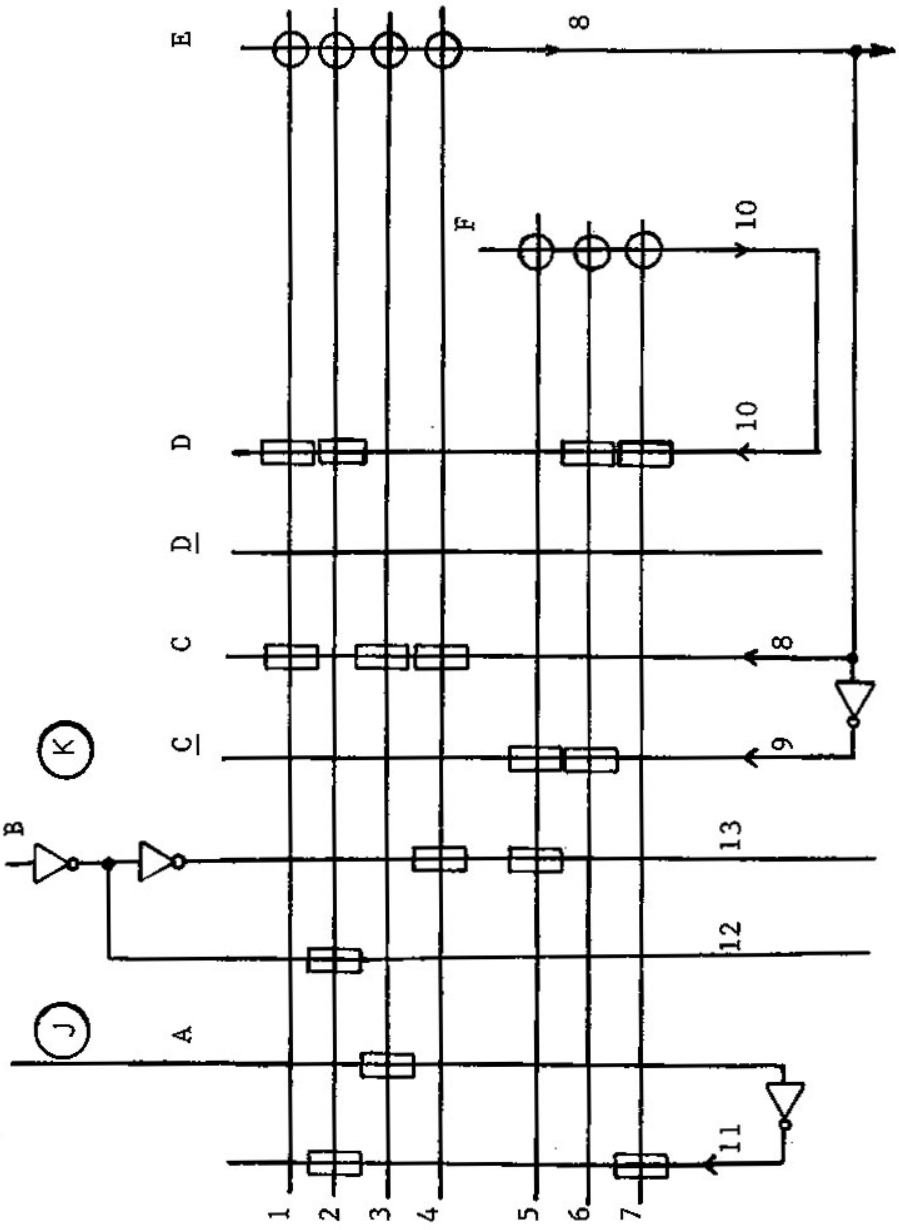


Fig. 6.9. Dynamic schematic: J-K flip-flop - trailing edge controlled, hazards eliminated. Speed independant circuit.

E. The hazard due to the delay of the B inverter shown in Fig. 6.8 is eliminated by the cascading of two inverters, as shown in Fig. 6.9, which causes line No. 12 to always change its signal level before line No. 13.

EXAMPLE 6.3.4. Design a nonclocked J-K flip-flop with trailing-edge control but with memory elements (S-R NOR latches) inserted in the feedback loop. A model with memory elements in the feedback loop is very attractive because it can be proven that there will be no hazards in the designed network if the logical time continuity design principles were respected during its design.

Fig. 6.10 shows the model of the circuit used in this example. Two S-R flip-flops make the labeling of the variables quite easy: The input signals C and D of the combinational circuit are derived from the outputs of the flip-flops together with their complements, $\underline{C}$ and $\underline{D}$ (output signal inverters are eliminated). Four control signals, E, F, G, and H, must be provided to control the flip-flops.

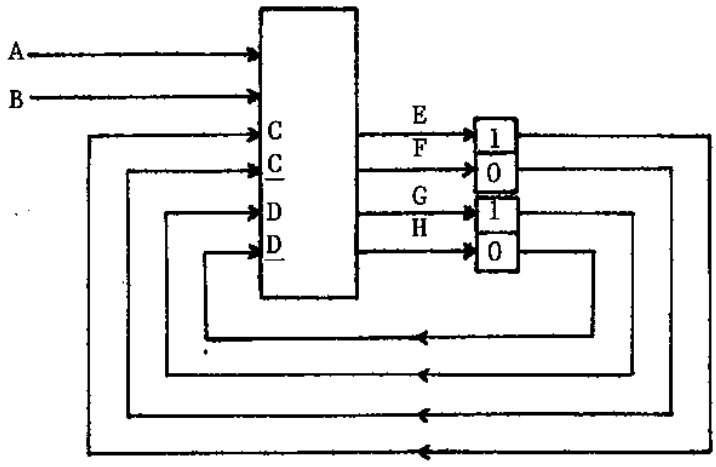


Fig. 6.10. Sequential circuit with memory elements (S-R flip-flops) in the feedback loop.

Fig. 6.11 shows the state diagram of the sequential circuit. The diagram is almost identical to that of Fig. 6.5; the trigger variables E, F, G, and H are not marked because they are not rigidly related to C and D.

The constraints formulated for the circuit are shown in Table 6.2.

To solve the set of simultaneous implications, we call:

```
     BULL
NUMBER OF CONSTANTS IS:
□:
     4
SYMBOLS FOR CONSTANTS: ABCD
NUMBER OF UNKNOWNS IS:
□:
     4
SYMBOLS FOR UNKNOWNS: EFGH
CALL FORMULA.
     FORMULA
WRITE DOWN THE FORMULA:
BCD→EG
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
BCD→GHE
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
ADC+BDC→EH
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
DABC→EFH
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
CDA→FH
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
ACD→HGF
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
     FORMULA
WRITE DOWN THE FORMULA:
ACD+BCD→FG
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
```


Table 6.2. Constraint formulations for Example 6.1.2.

	Formulation	Algebraic notation
1.	The circuit remains steady at i = 0 ($\underline{CD}$ = 1) for idle control ($\underline{AB}$ = 1) and for RESET ($\underline{AB}$ = 1). [Note that both flip-flops are reset (output at 0) to produce $\underline{CD}$. Signal E or G would set a flip-flop and change $\underline{CD}$ = 1.]	$\underline{B}\ \underline{C}\ \underline{D} \rightarrow \underline{E}\ \underline{G}$
2.	The control ($\underline{AB}$ = 1) or (AB = 1) at i = 0 ($\underline{CD}$ = 1) must cause the transition i = 0 → 2. [Note: To make D signal ON, the controlling flip-flop must be switched ON (output at 1): Trigger signal G, no trigger signal H. The other flip-slop must remain reset: No trigger signal E (anything at F).]	$B\ \underline{C}\ \underline{D} \rightarrow G\ \underline{H}\ \underline{E}$
3.	The transition i = 2 → 3 may proceed only when ($\underline{AB}$ = 1) = trailing-edge condition. Thus, the system must remain steady at i = 2 ($\underline{CD}$ = 1) as long as (A+B = 1).	$(A + B)\ \underline{C}\ D \rightarrow \underline{E}\ \underline{H}$
4.	The control ($\underline{AB}$ = 1) = trailing edge at i = 2 ($\underline{CD}$ = 1) causes transition i = 2 → 3. (Note: To set the C-controlling flip-flop ON, trigger E, but not F. Do not reset the other flip-flop; i.e., do not trigger H.)	$\underline{A}\ \underline{B}\ \underline{C}\ D \rightarrow E\ \underline{F}\ \underline{H}$
5.	The circuit remains steady at i = 3 (CD = 1) for idle control ($\underline{AB}$ = 1) and for SET ($\underline{AB}$ = 1). (No flip-flops to be reset.)	$\underline{A}\ C\ D \rightarrow \underline{F}\ \underline{H}$

Table 6.2 (Continued).

	Formulation	Algebraic notation
6.	The control ($\underline{AB}$ = 1) or (AB = 1) at i = 3 (CD = 1) causes the transition i = 3 → 1.	A C D → H $\underline{G}$ $\underline{F}$
7.	The transition i = 1 → 0 may proceed only when ($\underline{AB}$ = 1). For that reason, the circuit must remain steady at i = 1 ($\underline{CD}$ = 1) as long as (A+B = 1).	(A + B) C $\underline{D}$ → $\underline{F}$ $\underline{G}$
8.	The control ($\underline{AB}$ = 1) at i = 1 ($\underline{CD}$ = 1) causes the transition i = 1 → 0.	$\underline{A}$ $\underline{B}$ C $\underline{D}$ → F $\underline{E}$ $\underline{G}$

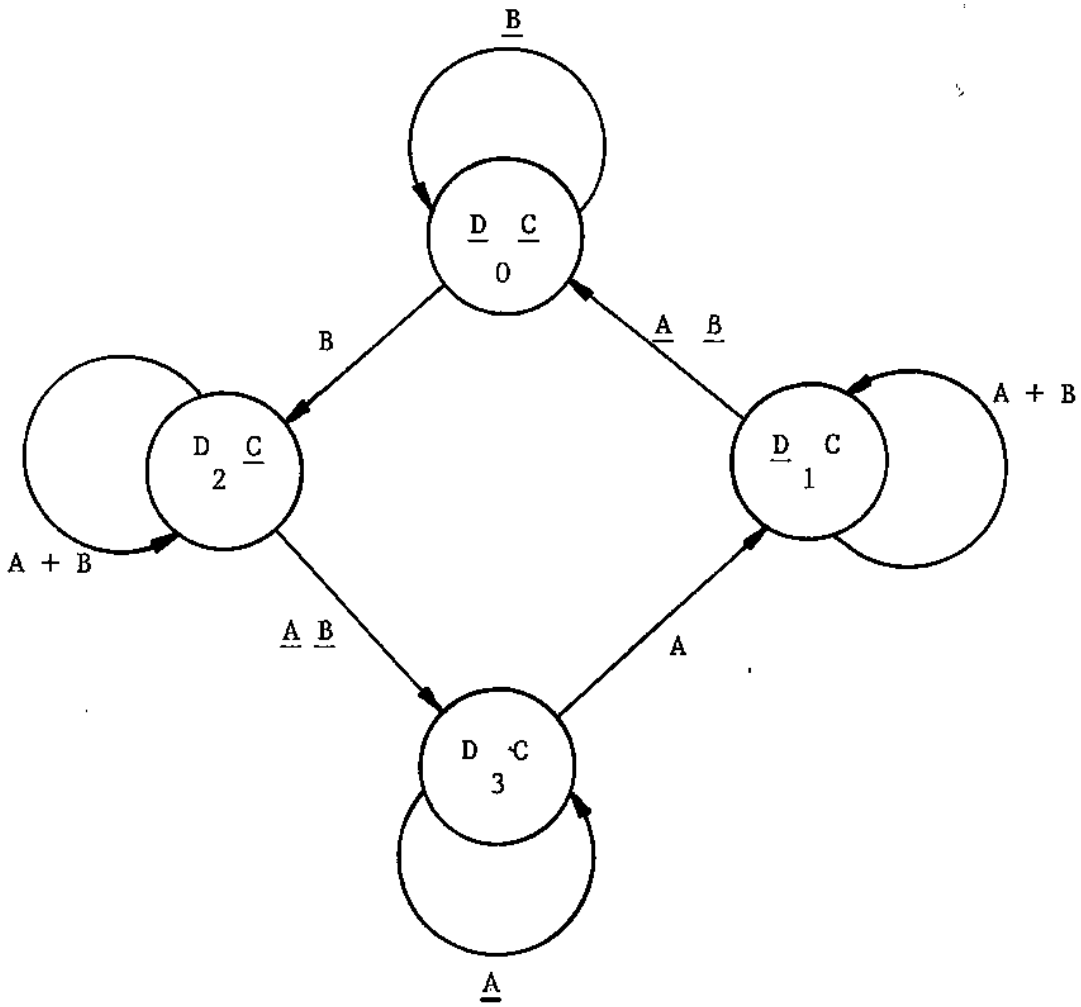


Fig. 6.11. State diagram for constraints generation.


```
      FORMULA
WRITE DOWN THE FORMULA:
CABD→FEG
MAY CALL EITHER (NEXT) FORMULA OR DISCRIMINANT OR EXECUTE.
```

The existence function of the system A, B, C, D, E, F, G, H is obtained by calling

```
      DISCRIMINANT
THE DISCRIMINANT VALUE BEFORE CONSTRAINTS IS:
 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 0
 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0
 1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0
 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1
 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1
 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
AFTER ::EXECUTE:: HAS BEEN CALLED, CALL MTY TO GET
THE CONSTRAINT RECTIFIED DISCRIMINANT.
CALL EXECUTE AFTER ALL CONDITIONS HAVE BEEN PUT IN.
IF NOT, CALL FORMULA AGAIN.
```

The procedure EXECUTE produces the outputs E, F, G, and H as functions of A, B, C, and D. The number of distinct solutions is extremely large (over 67 million). By typing 0 (zero) instead of a solution vector, the user can cause the terminal to print the solutions in such an order that the first solution contains the smallest number of minterm implicants and the last solution (here, solution number 67,108,864) contains that largest number of minterm implicants. This is important, as we shall demonstrate.

```
      EXECUTE
NUMBER OF SOLUTIONS UNDER ALL CONSTRAINTS:  SOL =  67108864
SOLUTION VECTOR OR TYPE  0  FOR CIRCUIT DESIGN.
□:
      0
```

```
SOLUTION NUMBER: 1
E = [8] ∪ ()
F = [4] ∪ ()
G = [2  3] ∪ ()
H = [13  15] ∪ ()
SOLUTION NUMBER: 2
E = [8  15] ∪ ()
F = [4] ∪ ()
G = [2  3] ∪ ()
H = [13  15] ∪ ()
SOLUTION NUMBER: 3
E = [8  14] ∪ ()
F = [4] ∪ ()
G = [2  3] ∪ ()
H = [13  15] ∪ ()
SOLUTION NUMBER: 4
E = [8  14  15] ∪ ()
F = [4] ∪ ()
G = [2  3] ∪ ()
H = [13  15] ∪ ()
SOLUTION NUMBER: 5
E = [8] ∪ ()
F = [4] ∪ ()
G = [2  3  14] ∪ ()
H = [13  15] ∪ ()
   •
   •  (printout is interruptable)
   •
SOLUTION NUMBER: 67108864
E = [5  6  7  8  12  13  14  15] ∪ ()
F = [0  1  2  3  4  9  10  11] ∪ ()
G = [2  3  8  9  10  11  12  14] ∪ ()
H = [0  1  4  5  6  7  13  15] ∪ ()
SOLUTION NUMBER: 67108863
E = [5  6  7  8  12  13  14] ∪ ()
F = [0  1  2  3  4  9  10  11] ∪ ()
G = [2  3  8  9  10  11  12  14] ∪ ()
H = [0  1  4  5  6  7  13  15] ∪ ()
SOLUTION NUMBER: 67108862
E = [5  6  7  8  12  13  15] ∪ ()
F = [0  1  2  3  4  9  10  11] ∪ ()
G = [2  3  8  9  10  11  12  14] ∪ ()
H = [0  1  4  5  6  7  13  15] ∪ ()
SOLUTION NUMBER: 67108861
E = [5  6  7  8  12  13] ∪ ()
F = [0  1  2  3  4  9  10  11] ∪ ()
G = [2  3  8  9  10  11  12  14] ∪ ()
H = [0  1  4  5  6  7  13  15] ∪ ()
SOLUTION NUMBER: 67108860
E = [5  6  7  8  12  13  14  15] ∪ ()
F = [0  1  2  3  4  9  10  11] ∪ ()
G = [2  3  8  9  10  11  12] ∪ ()
H = [0  1  4  5  6  7  13  15] ∪ ()
```


IMPLEMENTATION OF THE RESULTS:

Version 1 -- without DON'T CAREs: The functions E, F, G, and H are expressed algebraically. They are too simple to require simplification by computer:

E = $\underline{A}$ $\underline{B}$ $\underline{C}$ D; F = $\underline{A}$ $\underline{B}$ C $\underline{D}$; G = B $\underline{C}$ $\underline{D}$; H = A C D.

The dynamic schematic for this implementation is presented in Fig. 6.12.

Version 2 -- with DON'T CAREs: By comparing solution No. 1 with solution No. 67108864, we observe that each minterm implicant of No. 1 is present in No. 67108864. We conclude that minterms in No. 1 must be accepted as minterm implicants; the additional minterms in No. 67108864 may by accepted (playing the role of DON'T CAREs).

To do the minimization by computer, call (in SYSTEM):

To get E: MINIMA 0 0 0 0 0 2 2 2 1 0 0 0 2 2 2 2
(=> E = $\underline{A}$ $\underline{B}$ D)

To get F: MINIMA 2 2 2 2 1 0 0 0 0 2 2 2 0 0 0 0
(=> F = $\underline{A}$ $\underline{B}$ $\underline{D}$)

To get G: MINIMA 0 0 1 1 0 0 0 0 2 2 2 2 2 0 2 0
(=> G = B $\underline{C}$)

To get H: MINIMA 2 2 0 0 2 2 2 2 0 0 0 0 0 1 0 1
(=> H = A C)

The dynamic schematic implementing these equations is shown in Fig. 6.13. Note that the use of DON'T CAREs in the latter solution simplifies the design.

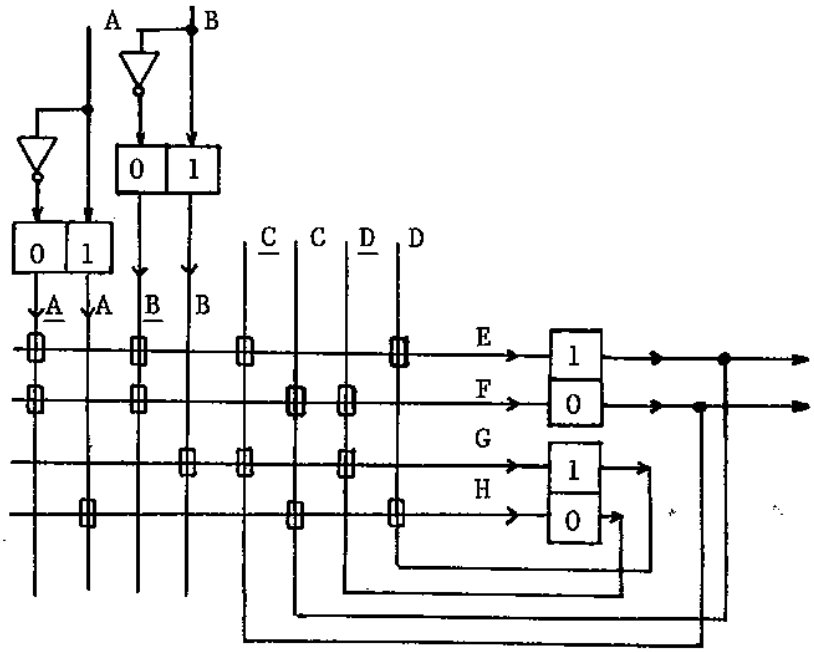


Fig. 6.12. J-K flip-flop with trailing edge control.

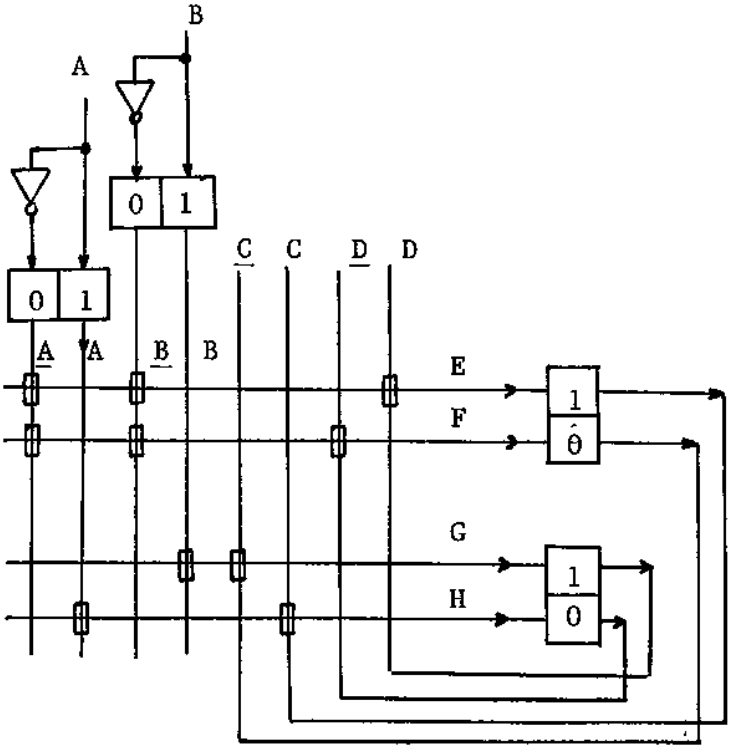


Fig. 6.13. Final design version of J-K flip-flop.

