first data corporation

LOGIC

12/72
Reprinted
July 1973

CONTENTS

		Page
1.	INTRODUCTION TO LOGIC SIMULATION	1
2.	THE LS747 LOGIC SIMULATOR	2
3.	HOW LOGIC SIMULATION WORKS	4
4.	GATES	11
5.	SETTING UP A SIMULATION	13
	5.1 The Network Data Block	13
	5.2 The Injected Signal Data Block	14
	5.3 The Output Report Data Block	15
6.	RUNNING THE LOGIC SIMULATOR	16
7.	ADVANCED EDITING COMMANDS	21
8.	OUTPUT TO DISK FILES	23
9.	MACROS	24
	9.1 Defining a Macro	25
	9.2 Macro Useage	27
	9.3 Printout with Macros	29
	9.4 Building Macro Libraries	29
10.	OPTIONS AND FEATURES	29
	10.1 Delay Codes	29
	10.2 Injected Signals	30
	10.3 Output Report	31

10.4 Remarks 32

10.5 Continuation Lines 32

1. SOME USEFUL MACROS 33

11.1 Inverter 33

11.2 Clock 34

11.3 Exclusive Or Gate 35

11.4 Set - Reset Flip Flop 36

11.5 Clocked D Type Flip Flop 36

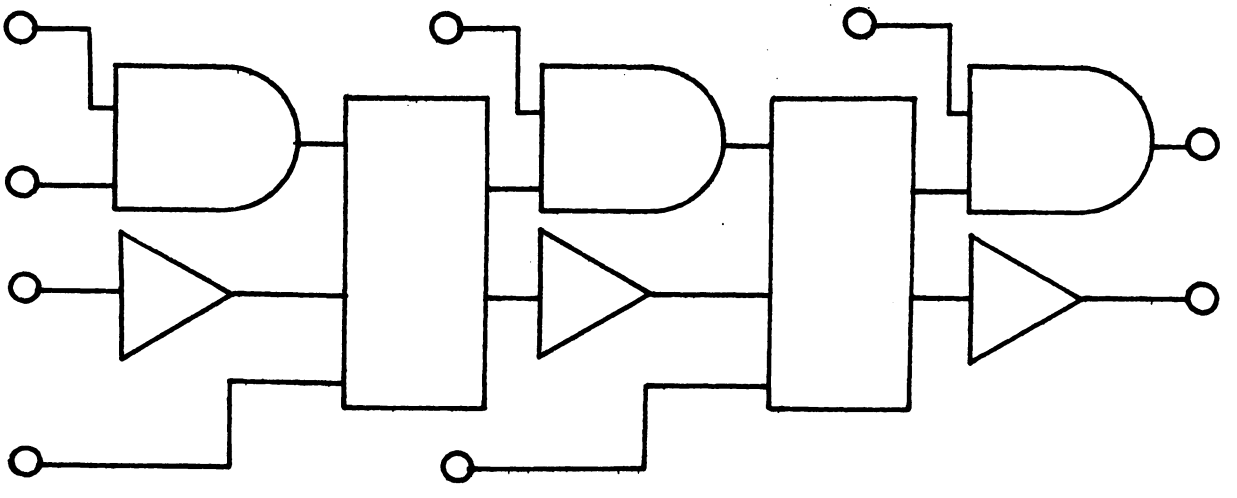
11.6 JK Flip-Flop 38

11.7 Master Slave Flip Flop 39

11.8 Four Bit Shift Register 41

1. INTRODUCTION TO LOGIC SIMULATION

Digital circuits consist of logical elements such as AND and OR gates. These gates are connected together to form modules such as shift registers which are usually made on a single semi conductor chip. These modules in their turn are connected together to form a system which is usually built on a printed circuit card.



Logic simulation is the process of analyzing the performance of either modules or systems to ensure that they will work correctly once fabricated. These analyses not only evaluate the logical correctness of the design but also the effect of delays in each of the gates or modules.

Logic simulation is primarily used to check designs before they are fabricated. In the case of designs for MSI or LSI modules, this is very important because the masks used to fabricate the modules cost many

thousands or dollars, and these masks cannot be reworked to correct a faulty logic design.

Simulation can also significantly reduce costs and time in the design of systems made up of modules. Using simulation, an engineer can check out his design in a few hours, bypass the breadboard stage, and proceed directly to the prototype printed circuit board.

The use of simulation also results in better designs with higher production yields and lower field maintenance costs, because engineers can rapidly evaluate different design approaches to determine the optimum one. Further, they can easily compare different manufacturers components in a circuit and test out the what-if questions that arise with any new design.

2. THE LS747 LOGIC SIMULATOR

LS747 is an advanced logic simulator written by logic design engineers. This program can simulate circuits with up to 2000 gates, including the effect of delays. LS747 has a powerful MACRO capability whereby users can define a complete MSI or LSI chip and then use this in their logic description.

LS747 can analyze both synchronous and asynchronous logic or a mixture of these with the individual logic elements having different rise and fall times. Further simultaneous level changes and parallel/serial operations are accurately simulated without user intervention.

LS747 uses look ahead simulation techniques, taking advantage of the fact that only a small percentage of the elements in a digital system change state at any time. It provides simulation of an element if and only if one of the inputs to the elements change state. By this means, extremely efficient simulations can be performed.

In addition to providing a wide range of simulation outputs, LS747 can provide such useful information as a cross reference listing for all the logic elements, giving the fan-outs for each of the elements and their connections as well as detecting undefined logic states.

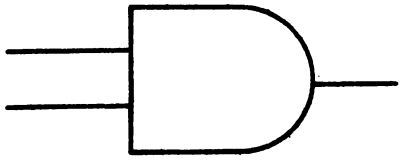
All this capability has been obtained without sacrificing ease of use. Data input and commands, for example, are in free format engineering language, making LS747 a truly useful tool for the logic design engineer.

3. HOW LOGIC SIMULATION WORKS

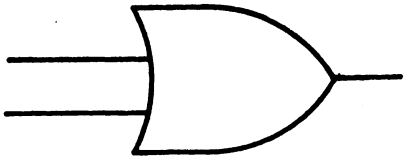
To perform a logic simulation the user enters a network description into the computer. This network description consists of logic gates such as AND and OR as well as collections of them, forming component descriptions, called MACROS.

The gates used in LS747 are shown in the diagram on the following page. These gates can be regarded as physical gates or as basic logical functions which can be used in forming complex logical functions.

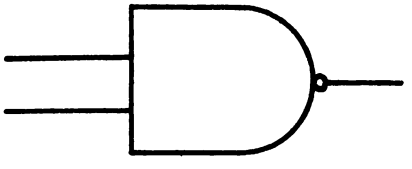
LS747 GATES



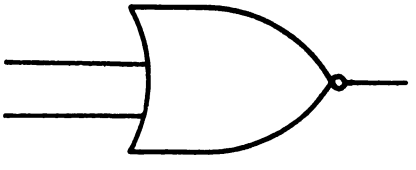
AND



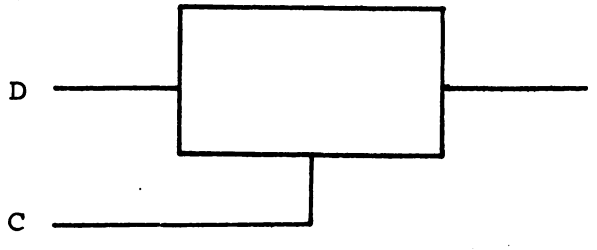
OR



NAND



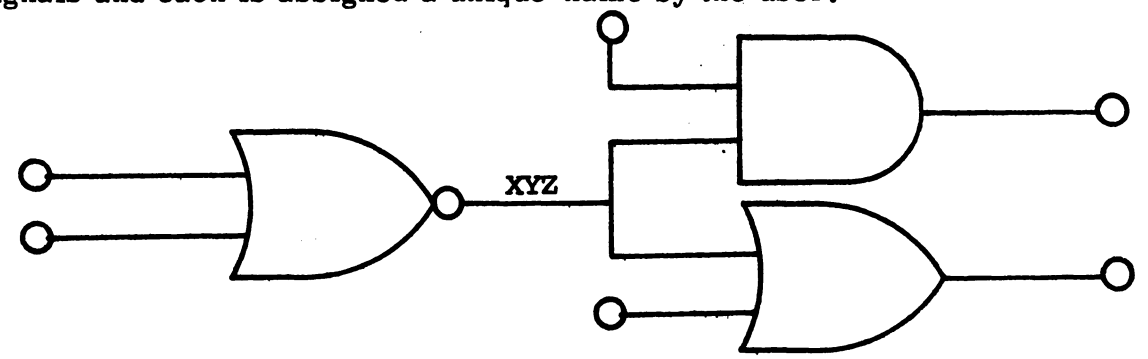
NOR



TRAM (MOS transmission gate)

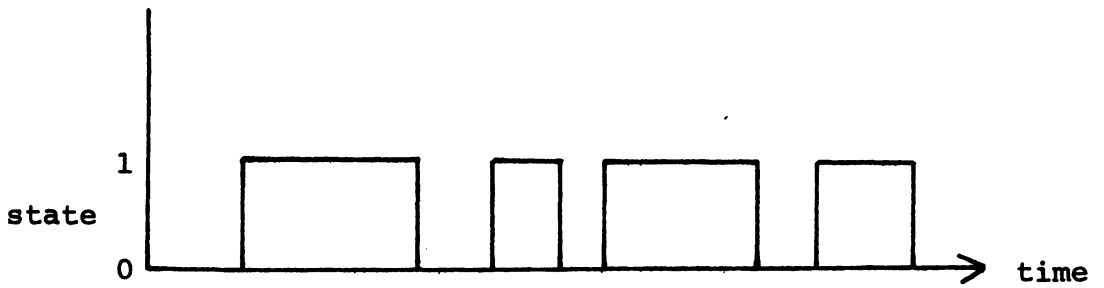
Each gate has one output signal, up to seven input signals and has associated with it a rise and fall delay.

Throughout this manual the interconnections between the gates are called signals and each is assigned a unique name by the user.

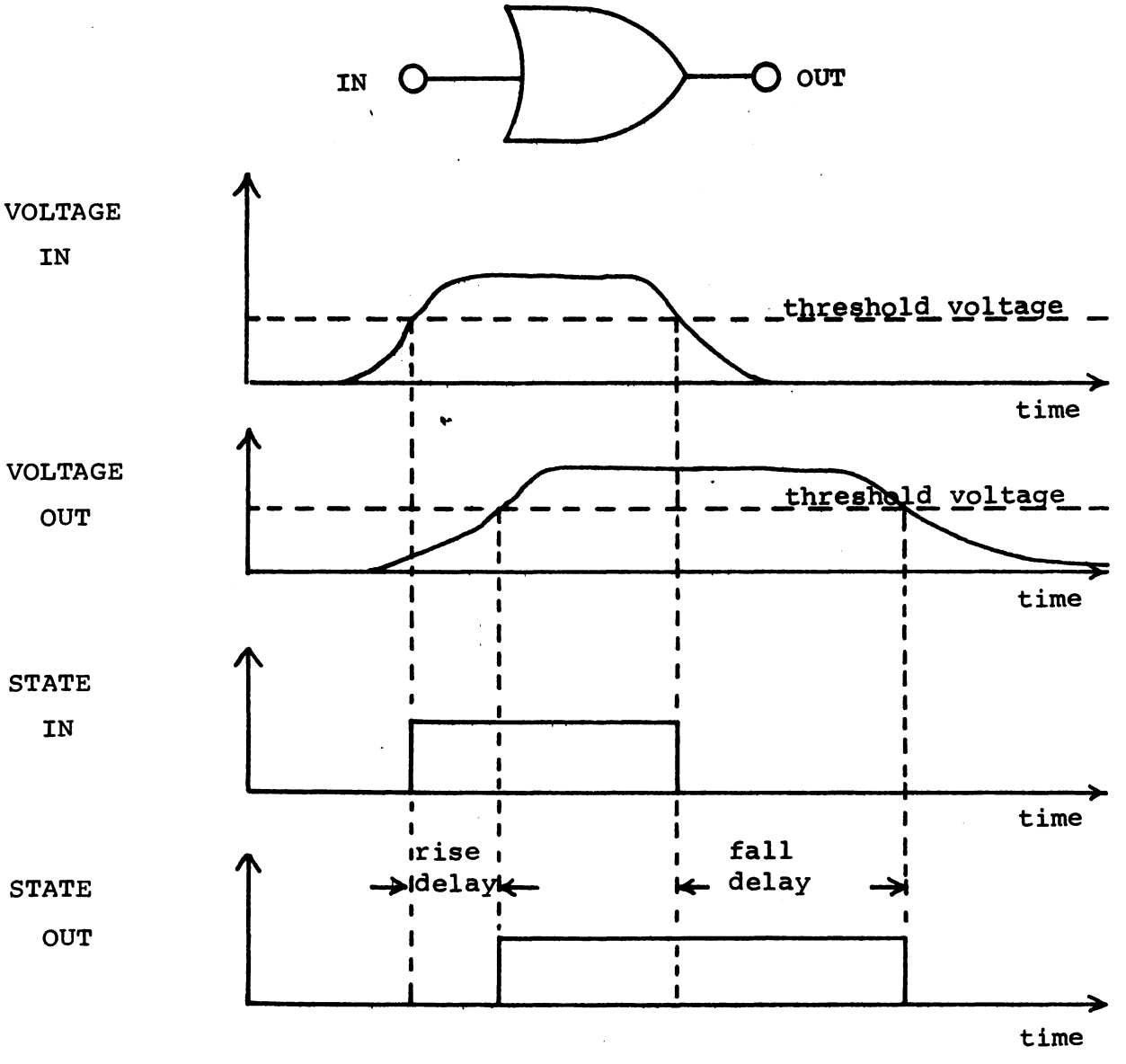


Here XYZ identifies the interconnection between the NOR gate and the AND and OR gates.

Signals are not characterized by their voltage levels, but by their states of either true (1) or false (0). Output from the simulation is in terms of the timing diagram showing the times of which the signal goes from one state to another.



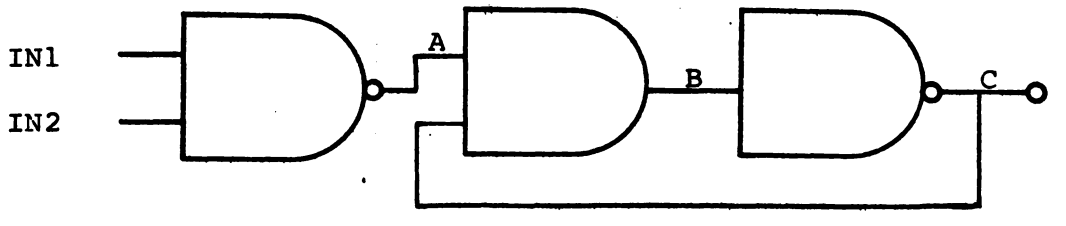
Gates and components physically delay both the rising and falling edge of an input pulse.

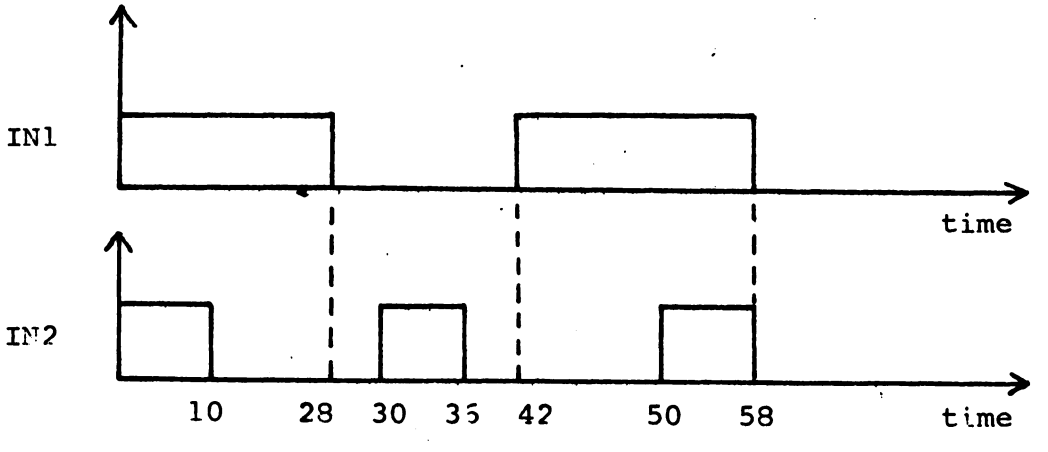


Because the simulator has only two states, the times are taken as those of which the circuit voltages cross the threshold from one state to the other. This threshold is the voltage at which the gates or modules switch states, and the rise and fall delays are specified as shown on the preceding diagram.

Times are specified in whole numbers such as 3, 97, 398 or 0. Fractional numbers such as 39.7 cannot be used. The user is free to use any time units he thinks are appropriate. However, it must be noted that one time unit is the smallest that can be differentiated by the system. Hence, if we have gates with rise/fall delays of tens of nanoseconds, a time unit of one nanosecond would be appropriate. Within LS747, the maximum delay that can be assigned to any gate is 511 time units, so that the user should not make the time unit too small either.

Injected signals are specified separately from the network by their initial states and the times at which they change states.





When analyzing a digital network, such as the above example, the first step taken by the program is to check the logical consistancy of the inputs. Here such errors as gates having input signals which are not connected anywhere are caught.

Then, prior to running the simulation, the network is initialized. In this the initial states of all signals are computed by the foraword propagating the signals through the network. Hence in the above example A is the NAND of IN1 and IN2, which are both initially 1, and hence is zero. B is the AND of A and C. Now at this point in the propagation process C is undefined, and the result of an AND of a zero (A) and an undefined (C) is zero. Hence B is zero and C is a 1. The process of propagation is extended throughout the network until all possible states have changed from undefined to either a zero or a one state.

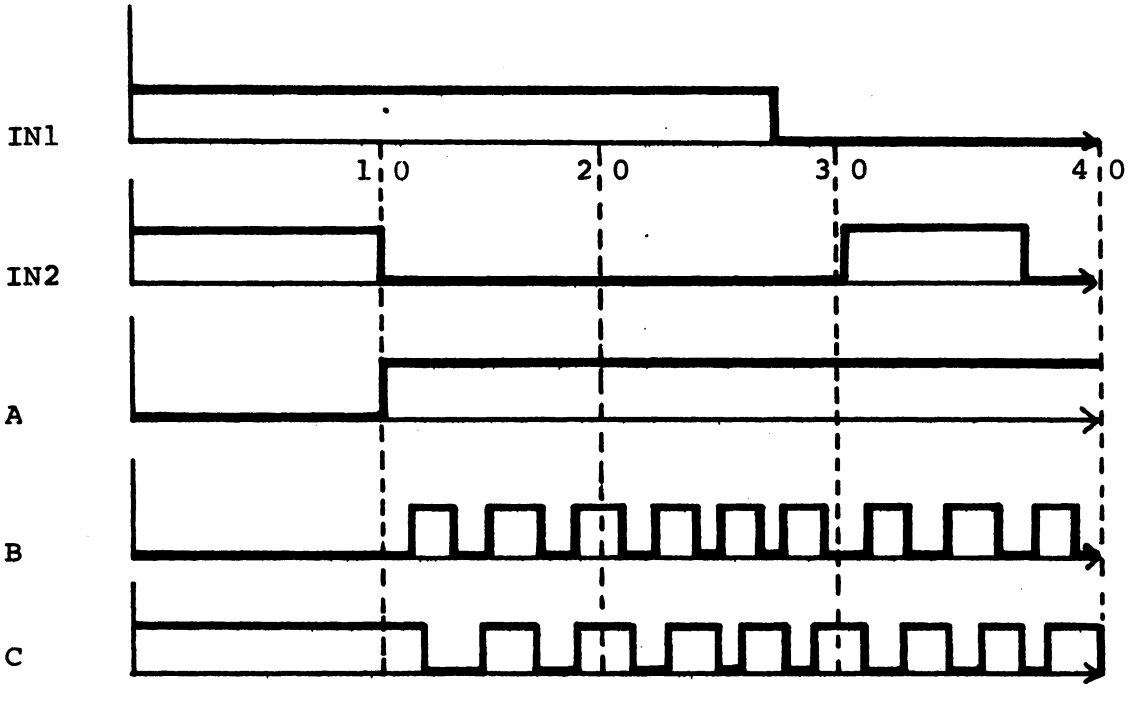
Each gate output is only allowed to change from an undefined output to a one or zero state and not from zero to one or vice versa. The initialization is complete when all gates that could change from undefined to defined states have done so. Those gates that have undefined outputs are then flagged as such in the listing of initial states.

It is noteworthy that in our example if the input NAND gate had been an AND gate the states of B and C would have been undefined as the AND of 1 (A) and undefined (C) is undefined. This implies that these states depended on the state changes of signals IN1 and IN2 prior to time zero and hence we have

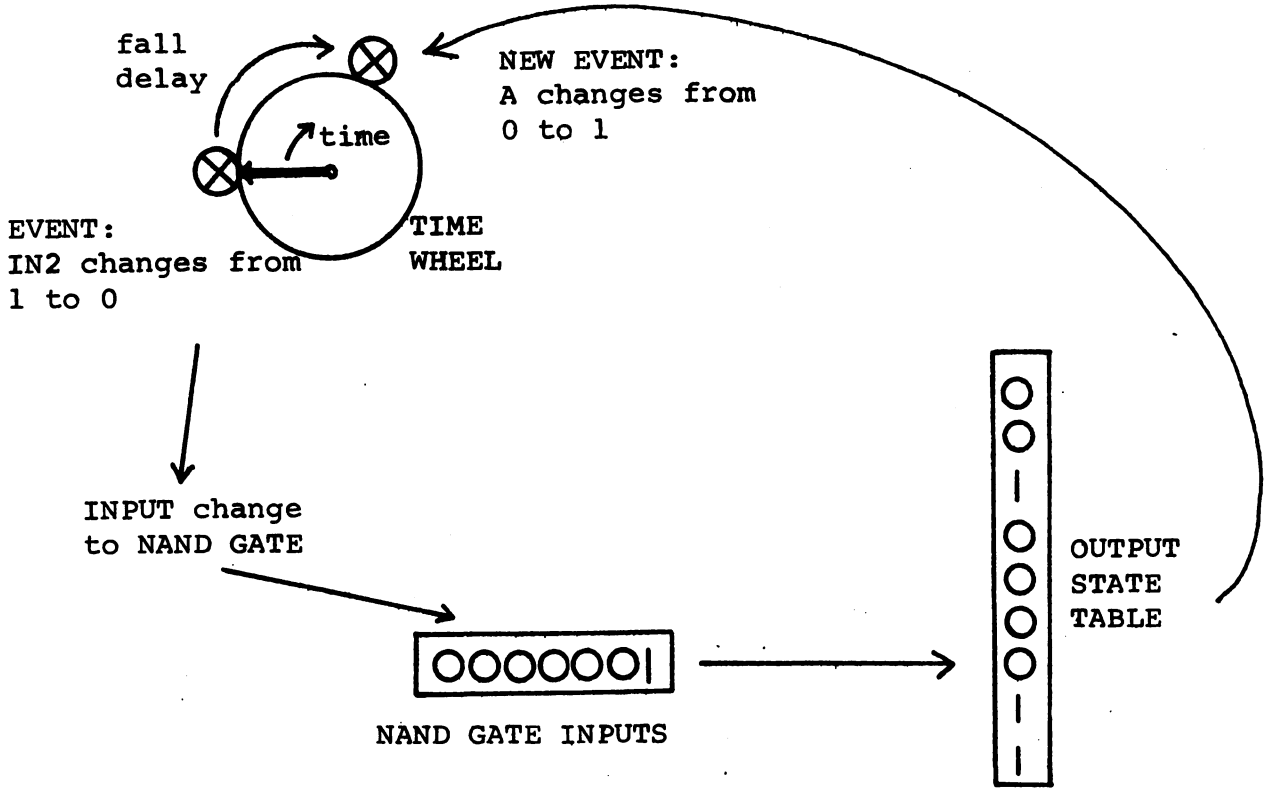
not got a valid test. These undefined signals are set to zero state prior to running the simulation.

Those signals that are not connected to the input of any gates, and are not requested for output are printed as undefined. LS747 does not save the states of the outputs the user is not interested in, to speed the running of the simulation. Hence these are printed as undefined after initialization.

The simulation is then performed to determine the times at which all the signals change state in the network resulting from changes in the inputs. The result is a timing diagram for the signals.

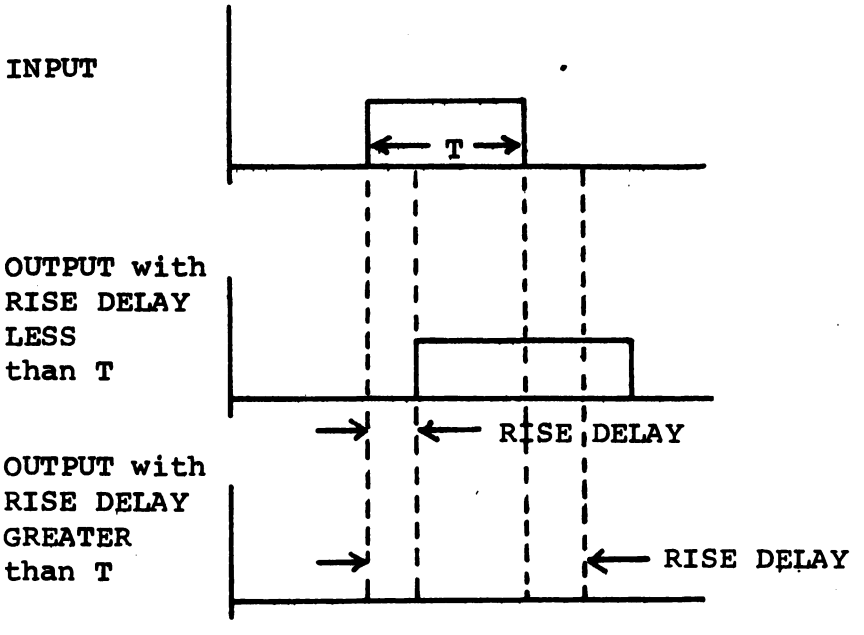


This is achieved by using a time wheel simulation technique. In this, events are placed on a time wheel every time a signal changes state.



This process is illustrated in the diagram shown above. The event, when signal IN2 changes from 1 to 0, is placed on the time wheel. When the time pointer reaches this event, the event triggers changes in all gates to which that signal is connected. The new output state is found for each gate using a truth-table look up, and if the output changes state, this new event is placed on the time wheel at a time delayed by the rise or fall delay specified for that gate. This event will in turn trigger other events. In this way the propagation of signal changes through the network are simulated for all the changes in the injected signals.

During the simulation a check is made for spike conditions. If an input pulse turns off again within the propagation time (rise or fall) of a gate then no output change is generated.



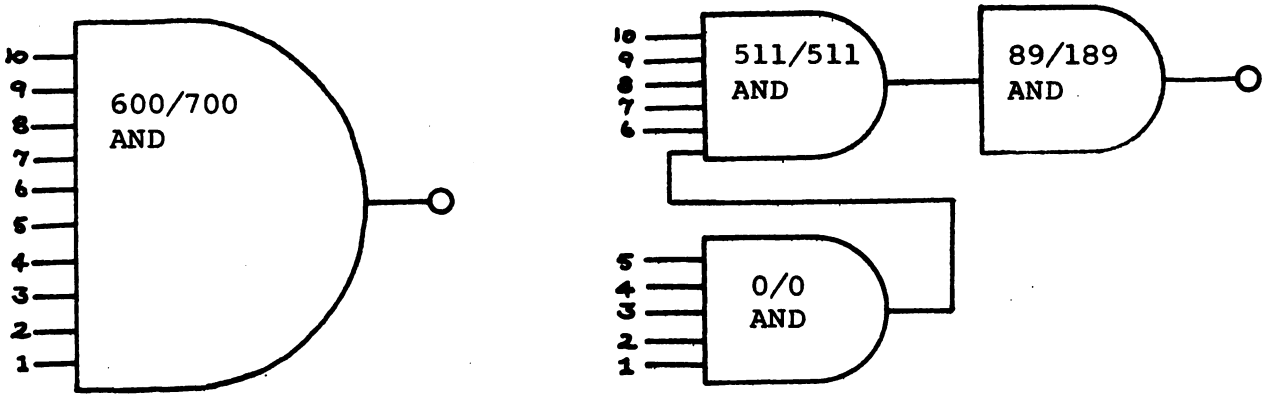
4. GATES

The LS747 gates are AND, OR, NAND, NOR and the MOS-transmission gate (TRAM). Negation in the case of NAND and NOR gates is on the output. These gates operate as conventionally defined but are subject to the following implementation restrictions:

1) Each gate is limited to a maximum of 7 inputs.

2) The maximum RISE or FALL delay is 511 time units.

These restrictions are, however, easily overcome by directly replacing a gate which has many inputs and large delays with an arrangement of substitute gates, or indirectly and more elegantly via the LS747 MACRO facility (see section on Macros).



The MOS-transmission gate (TRAM) is a two-input element which has a control input and a data input. When the control input is ONE, the element acts as a memory device which retains its output state regardless of any changes in the data input. When the control input is changed to ZERO, the element loses it's memory characteristics and transmits, i.e., the output assumes the same state, and goes through the same state changes as the data input.

5. SETTING UP A SIMULATION

To set up a simulation the user has to provide three blocks of data:

NETWORK

INJECTED SIGNALS

OUTPUT REPORT

The blocks of data are specified in the order shown with each data block starting with one of the above, and ending with the word END as shown below:

NETWORK

network description — Network data block

END

INJECTED SIGNALS

injected signal description — Injected signal data block

END

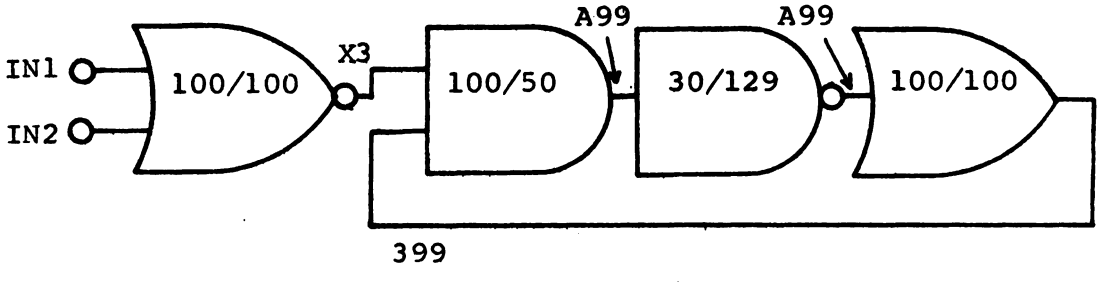
OUTPUT REPORT

output signal specification — Output report data block

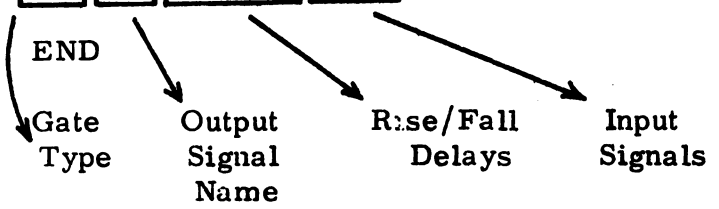
END

5.1 The Network Data Block

The format used for describing a network is best illustrated by example



```
NETWORK

NOR X3  100/100  IN1  IN2

AND A99 100/50  X3 399

NAND  A99/  30/129  A99

OR  399  100/100  A99/

END
```



One gate description is entered on each line. Each entry is separated by one or more blanks. There must be no blanks within the rise/fall entry.

Signal names can be up to 10 characters in length and can be either alphabetic or numeric or a mixture, and can include the special character / .

5.2 The Injected Signal Data Block

The example given below shows how initialization and simulation activity is induced into the small network previously defined.

```
INJECTED SIGNALS

IN1   0  1000  1200

IN2   1  100   650

END
```

This data block gives the following information to control an LS747 run:

. The network has two external inputs, i.e. IN1 and IN2.

. For the purpose of initialization, IN1 is forced to ZERO and IN2 is forced to ONE.

. During simulation, IN2 is forced to ZERO at time 100 and back to ONE at time 650. Also, IN1 is forced to ONE at time 1000 and ZERO at time 1200.

The injected signals are entered one to a line.

5.3 The Output Report Data Block

Returning to the small network previously defined, this example shows how a request for printing signals of interest is formulated.

```
OUTPUT REPORT
IN1, IN2, X3, 399, A99, A99/
END
```

The output report specifies those signals to be printed on the timing diagram output. (Initial states of all signals are printed in the fanout listing. The maximum number of output report signals that can be printed on the teletype is 50 or on the line printer (via the disk) is 100.

The output signals are printed every time one of the signals specified changes state, unless the user chooses to strobe the output using the MODE option as described in Section 9.3.

6. RUNNING THE LOGIC SIMULATOR

In the following, everything typed by the user is underlined, everything else by the computer. The symbol ↵ indicates pressing the carriage return key.

After logging into the computer the program can be run by typing

.R LS747 ↵

The following dialog is then typical of that used to enter the simulation data into the program.

```
LS747  REL  3.1  11/17/71
NEW or OLD ? NEW ↵                    to create a new description
NAME ? EXAM ↵                         with a file name EXAM.DAT
> 10  NETWORK ↵
> 20  NOR   X3   100/100  IN1  IN2 ↵
> 30  AND  A99   100/50  X3  399 ↵
>40  NAND  A99/  30/129  A99 ↵
>50  OR  399  100/100  A99/↵
>60  END↵                            enter simulation description
>70  INJECTED SIGNALS ↵
>80  IN1  0  1000 ↵
>90  IN2  1  100  650 ↵
95 END
>100  OUTPUT REPORT ↵
>110  IN1, IN2, X3, 399, A99, A99/ ↵
```

> 120 END ↵

> 130 XXX ↵

>

The name file can be up to five characters in length and is used to form the file name with a .DAT extension into which the circuit description is saved whenever a simulation is run or an exit is made from the program.

The circuit description is entered and edited in a manner similar to that used for the BASIC language. That is, the user starts each line with a number between 1 and 99999 and the program uses these line numbers to place the lines typed by the user in ascending numeric order, irrespective of the order in which the lines are typed. Hence to insert a line between 90 and 100, the user would simply type

> 92 END ↵

A line can be changed simply by retyping it. For example to alter line 80, the user would type

> 80 IN1 0 1000 12000 ↵

The program knows whether to alter or add lines by comparing the new line number with the old ones.

A line can be deleted by simply typing the statement number, for example:

> 130 ↵

would delete the line with statement number 130.

The current simulation description can be listed with the editing command

```
> LIST ↵
  10 NETWORK
  20 NOR  X3  100/100  IN1  IN2
  30 AND  A99  100/50  X3  399
  40 NAND  A99/  30/129  A99
etc.
```

Alternately, a single line can be listed, as for example, by

```
> LIST 80 ↵
  80  IN1  0  1000  1200
```

Or a range of lines can be listed

```
> LIST 10,20 ↵
  10 NETWORK
  20  NOR  X3  100/100  IN1  IN2
>
```

Other editing commands are described in the next section. All these and the following commands can be interspersed with changes to the simulation description specified by the line numbers.

To run a simulation the user simply gives the command

```
RUN ↵
```

and the simulation proceeds as shown below:

COMPILING — program is converting users description to tables of data

INITIALIZATION — program starts initialization process

GATE NO	GATE NAME	GATE TYPE	STATUS AND STATUS	FANOUT FANOUT
30	A99	AND	0	40 A99/
40	A99/	NAND	1	50 399
80	IN1	GEN	0	20 X3
90	IN2	GEN	1	20 X3
20	X3	NOR	0	30 A99
50	399	OR	1	30 A99

line number on which gate is described

Signal name of gate output

GEN is injected signal

State 0, 1, or U

line numbers and output signal names of gates to which gate is connected

SIMULATION

```
                          A
                       II 3A9
EVENT                  NNX999      output signal names printed vertically
COUNTER  TIME          12399/
                       ------
                       ABCDEF      letter identifying signal
                       ------
   0         0          B D F
   1       100            D F     letter printed when signal = 1
   1       200           CDEF     blank when signal = 0
   1       429           CDE
   1       529           C E      signals printed whenever
   1       579           C        any output signal changes state.
   1       609           C  F
   2       650          BC  F
   2       709          BCD F
   0       750          B D F
```

SYSTEM DIES NATURALLY BEFORE TIME = 1024

The event counter gives the number of gates changing state at the printout time.

The simulation will stop whenever there are no more events to take place, or when the stop time specified by

>HALT 700↵

for example, is reached. If no stop time is specified, then the simulation will run until no more events occur.

When the simulation is finished, a return will be made to the editing mode, and after the program has typed

>

the user can proceed to alter his circuit description before giving another RUN request.

After the user has finished his session he types

>EXIT↵

and the current simulation description will be saved on the disk, and then a return will be made to the monitor whereupon the user can log out.

The simulation description will be saved on the disk file unless the user deliberately chooses to delete it. This file can then be retrieved for use in subsequent sessions by proceeding as follows

.R LS747↵

NEW OR OLD ? OLD↵ To use old file with name EXAM.DAT

NAME ? EXAM↵

The description in the file will be retrieved and, as soon as the symbol > has been typed, the user can proceed to alter the description and run other simulations.

If the user wishes to run only the initialization but not the simulation, he should give the command

>INITIALIZE ↵

instead of RUN.

7. ADVANCED EDITING COMMANDS

A range of statements can be deleted by the command

>DELETE 40,256 ↵

For example, to delete lines 40 to 256 inclusive.

In all commands where a range of statement numbers is given, all lines falling within this range are affected. If there are no statements in the range, the message

SNE

for Statement Number Error is given. This message is also given if a single specified line does not exist. With the exception of the DELETE and

RESEQUENCE commands the absence of any line numbers is taken to imply the whole description. To DELETE the complete description type

> SCRATCH ↵

All the commands can be abreviated to their first three letters.

The RESEQUENCE command is used when the user has used up all the intermediate numbers and wishes to insert more lines between some existing ones. An example of useage is

> RESEQUENCE 100,10 ↵

to resequence the statement numbers so they start at 100 and are incremented by 10.

The current description can be saved under a different name by the command

> SAVE ↵

NAME? TEMP ↵

Alternately, only part of the description such as a MACRO description can be saved by giving the range of line numbers

> SAVE 106,128 ↵

NAME ? MAC1 ↵

Similarly data from another file can be added to the current description by the command

> GET ↵

NAME ? MAC1 ↵

Here it must be remembered that if some of the statement numbers in this file are the same as in those in the current description, then those in the incoming file will overwrite the lines in the current description.

When a range is specified for the GET request this range applies to the data in the incoming file, so that particular lines can be selected from this file.

If you mess up your current description you can leave the program without overwriting the last saved version (at INIT, RUN or EXIT) by giving the request QUIT.

Or alternately, you can SCRATCH your current description and GET your last saved version.

At any time you can get a different file as your current description by the command OLD or start creating a new description with a new name by using the command NEW.

8. OUTPUT TO DISK FILES

The simulation outputs are all directed to the teletype as standard.

However, this can be changed by the command

> SETUP ↵

WHICH OUTPUT ? FANOUT ↵

DSK OR TTY ? DSK ↵

NAME ? FAN1 ↵

The output which can be set up are FANOUT and SIMULATION. The first character is sufficient to specify the output to be directed. The devices to which output can be directed are disk (DSK) or the users teletype (TTY).

To use the line printer the output is directed to disk (DSK) and then spooled using the system PRINT command.

The name is the file name (up to 5 characters), with an implied .DAT extension to which the output is to be directed. This is only requested for DSK output.

Once set up, the output will remain directed as specified until a change is made. In the case of disk files, the old files will be overwritten with each successive simulation run unless the name is changed.

9. MACROS

A MACRO is a collection of gates which is described once and then used many places in the network description, in the same manner as an individual gate. These MACROS are used to describe such items as FLIP-FLOPS, SHIFT REGISTERS and complete MSI or even LSI chips.

They can also be used for logical functions and to give meaningful names to single gates, such as INV for a single input NAND gate.

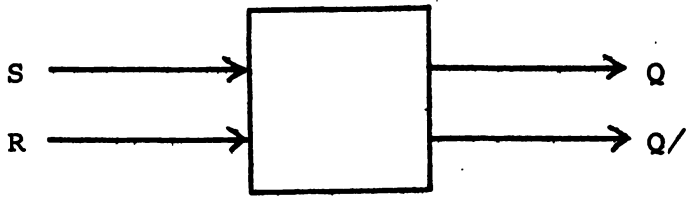
When a MACRO description is changed, such as when another manufacturer's part is being tried, all associated MACRO-usages are automatically changed. Hence, this significantly reduces the time to change a network description repeatedly using the same component.

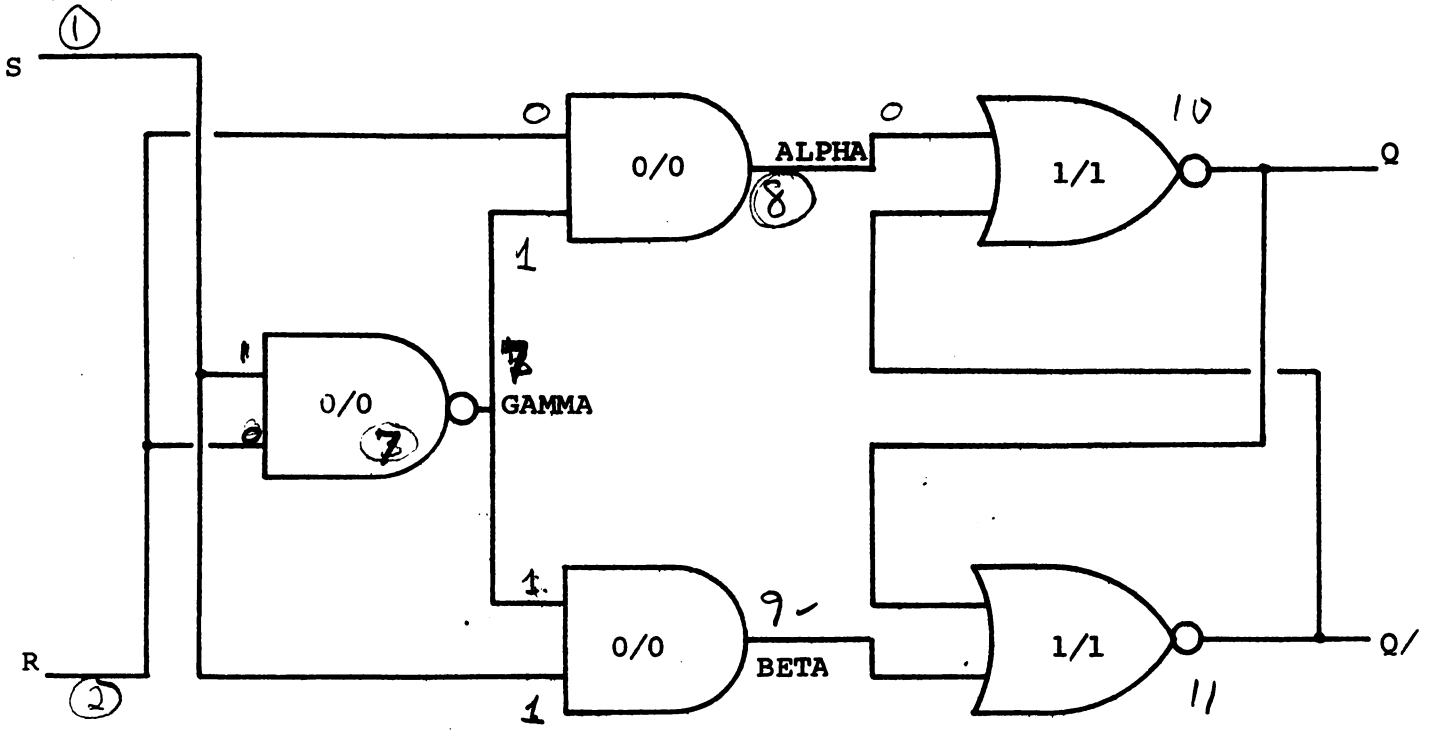
9.1 Defining a Macro

Each macro definition begins with a line having the word MACRO followed by at least one space and a macro name of 1 to 5 characters, and terminates with an END card.

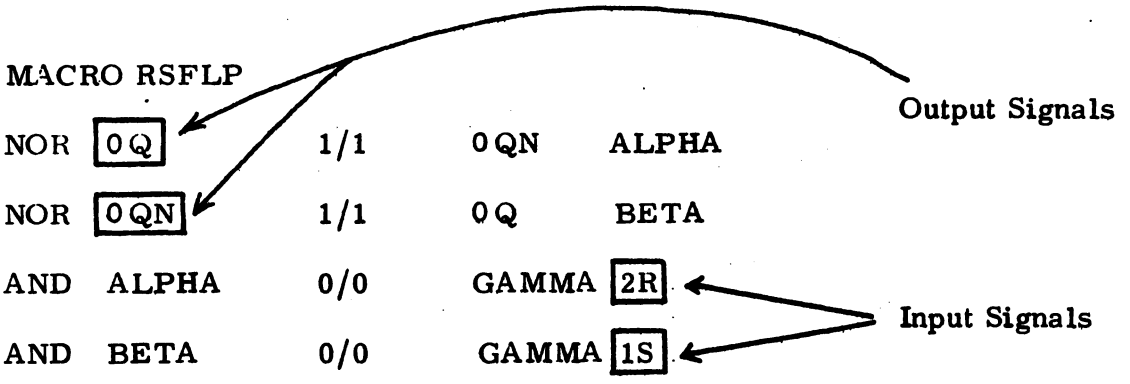
The gates that make up the macro are described in the same form as gates in the network description. Signal names must begin with an alphabetic character, except for input and output signals. The names of the macro output signals are preceded by the numeral 0, and the input signals by the numbers 1 through 99, indicating the order of input signal specification in the use of the macro.

The example shown below is an RS flip-flop:





Coding:

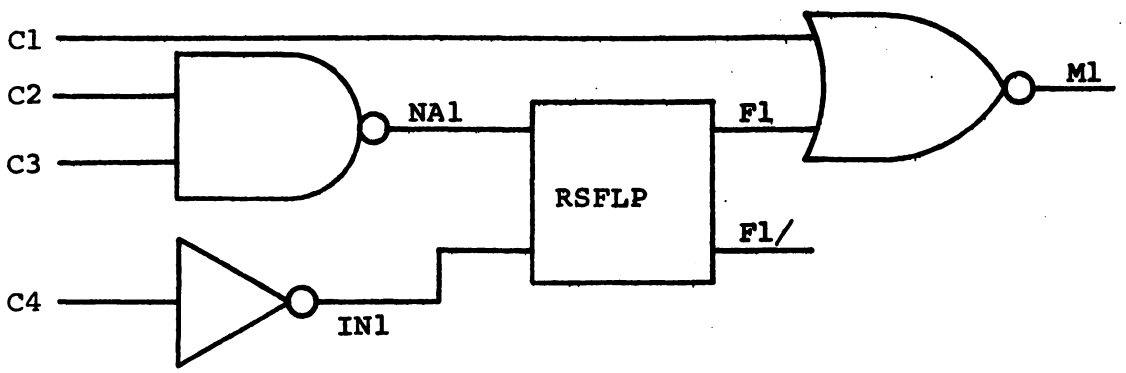


9.2 Macro Usage

When a macro element is used in a network, the output names must be coded in the same order as they appear in the macro definition. Input names must appear in the order specified by the integer prefix used for the macro definition input names.

Using a macro is the same as using a basic element with two exceptions. First, the macro name is used in place of the element type; second, a separate line is used for each output name. The macro name and input signals are dropped on the second and succeeding output lines but a delay code must accompany each output name. This delay code will override the delay code appearing on the corresponding output in the macro definition.

The example below shows an RSFLP used in a circuit.



The corresponding NETWORK data block is shown below

```
10  NETWORK
20  MACRO RSFLP
30  NOR 0Q 1/1 0QN ALPHA
40  NOR 0QN 1/1 0Q BETA
50  AND ALPHA 0/0 GAMMA 2R
60  AND BETA 0/0 GAMMA  1R
70  NAND GAMMA 0/0 1S 2R
80  END
90  MACRO INV
100 NAND 0OUT 0/0 1IN
110 END
120 NAND NA1 5/15 C2 C3
130 INV IN1 8/16 C4
140 RSFLP F1 30/47 NA1 NA2
150 F1/ 30/47
160 NOR NO1 7/18 F1 C1
170 END
```

RSFLP MACRO

INV (inverter) MACRO

NETWORK DESCRIPTION USING INV AND RSFLP MACROS

Use of RSFLP MACRO, one line for each output, all inputs on first line.

Note that the MACRO definitions must be the first entries in the NETWORK data block, followed by the network description itself.

One MACRO can call another MACRO, provided that the called MACRO has been defined before the call.

9.3 Printout With Macros

All signals that are internal to macros are not printed out either in the fanout listing or output report. Only those names that are given in the actual network can be accessed. On the fanout listing the gate type for a macro is its name.

9.4 Building Macro Libraries

Users can save macro descriptions using the SAVE command and retrieve them for use by using the GET command. In this way users can rapidly build up a library of commonly used MACROS for use in their networks.

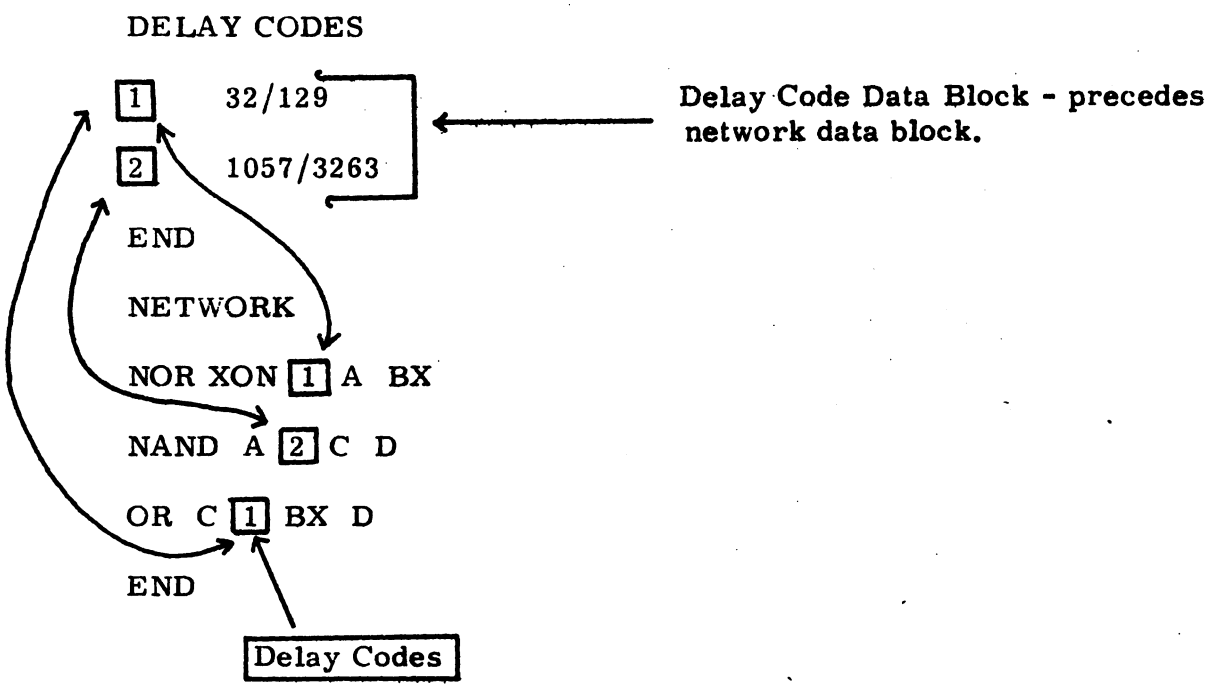
10. OPTIONS AND FEATURES

10.1 Delay Codes

Delays are usually entered in in-line format as shown below

NOR XON 32/129 A BX

However, they can also be entered in delay code format



This can save typing if the same rise/fall delay is used many places in the network. Also when the user wishes to change that delay, only the delay code line need be changed.

10.1 Injected Signals

All the injected signal time specifications can be multiplied by a constant by giving a RATE parameter.

```
INJECTED SIGNALS RATE = 500
CLK1  0   1,2,3,4,5,6,7,8,9
CLK2  1   3,9,11,15
END
```

In this case all the signal times would be multiplied by the specified rate.

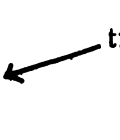
Multiple rate entries can be used.

INJECTED SIGNALS

RATE = 500

CLK1 0 1, 2, 3, 4, 5, 6, 7, 8, 9

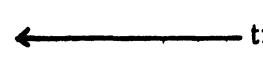
CLK2 1 3, 9, 11, 15



times multiplied by 500

RATE = 1000

CLK3 0 1, 2, 3, 4



times multiplied by 1000

END

10.3 Output Report

Instead of printing a letter code when the output signal is in the one state, the user can have the states printed in terms of 1's and 0's by specifying

OUTPUT REPORT PROPT = 1

The PROPT = 1 entry telling the program to switch to 1's and 0's.

Spaces can be left between vertical lines of printout by inserting additional commas in the output report data block:

OUTPUT REPORT

XA, XB, XC, , , , XE, XF

will cause one space between signals XB and XC and three spaces between XC and XE

The option

OUTPUT REPORT MODE = 1

will cause the output signals to be printed only when the first one in the list changes state. In this case the time is printed in terms of the pulse count of this signal.

The entries PROPT = 1 and MODE = 1 can both be entered on the OUTPUT REPORT line if so desired.

10.4 Remarks

Remarks can be inserted into the input data by following the line number by the symbol* and then a space

```
100*  TEST OF SHIFT REGISTER MEMORY
```

These lines will be ignored by the simulation itself.

10.5 Continuation Lines

Lines can be continued by starting continuation lines by the line number followed by a + then a space:

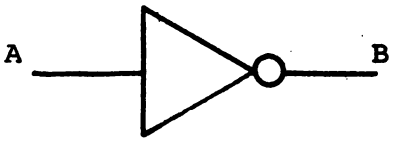
```
10  INJECTED SIGNALS
20  CLK  0  100,200,300,800,900,
30 +1000,1010
40  END
```

continuation line

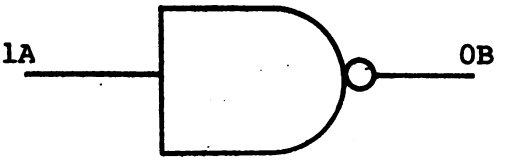
11. SOME USEFUL MACROS

11.1 Inverter

Symbol



Gating Structure



$B = \overline{A}$

Description

```
MACRO INV
NAND 0B 1/1 1A
END
```

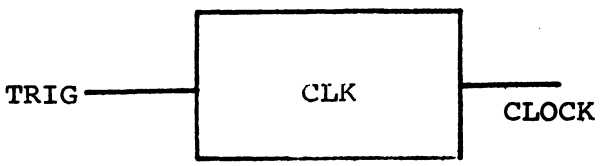
Use

```
INV B 5/19 A
```

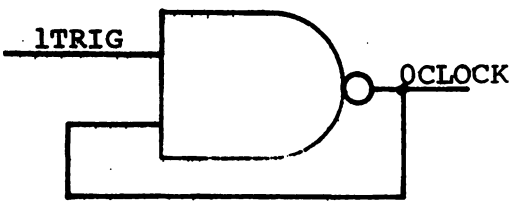
Actual Delay

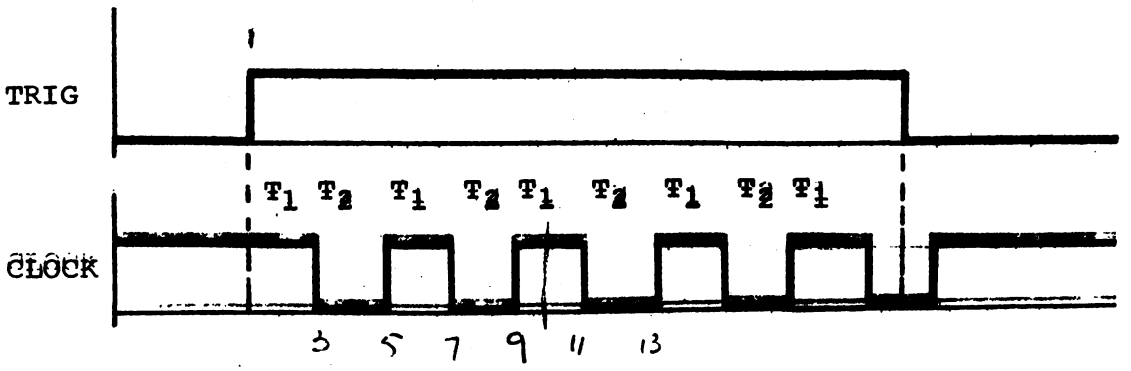
11.2 Clock

Symbol



Gating Structure





Description

```
MACRO  CLK
NAND   0CLOCK 1/1 0CLOCK 1TRIG
END
```

Use

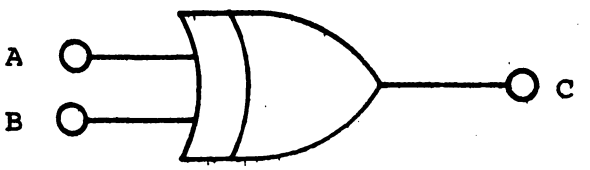
```
CLK   CLOCK   15/37   TRIG
```

Space = T_2 = 15

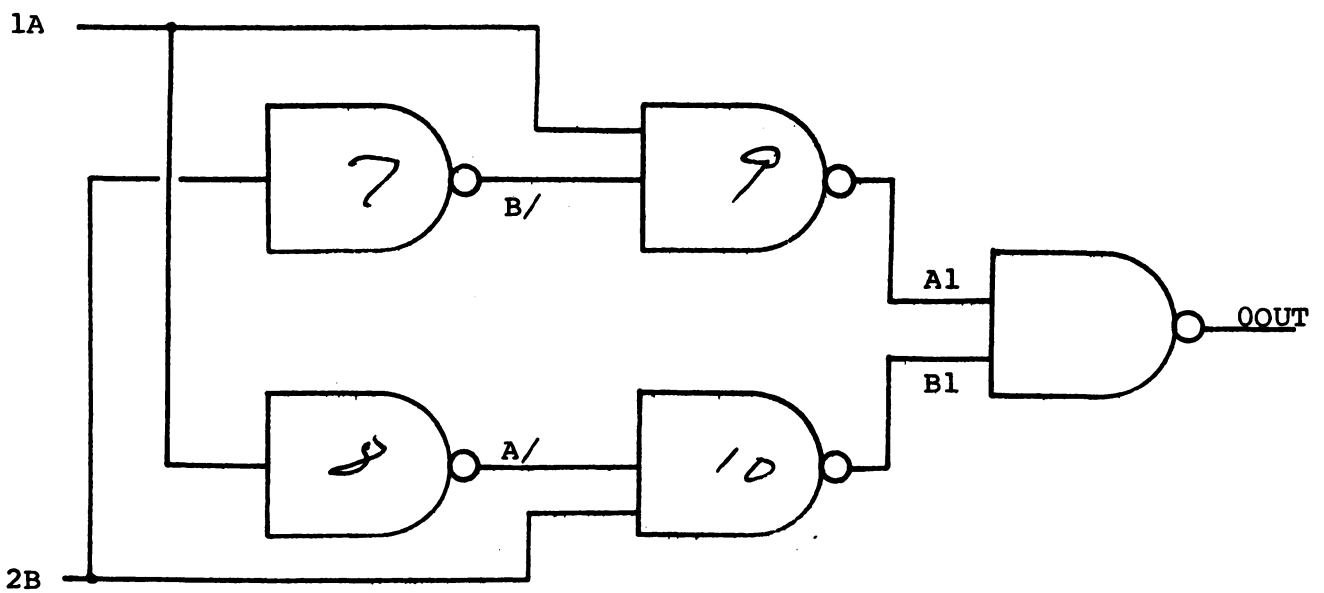
Mark = T_1 = 37

11.3 Exclusive Or Gate

Symbol



Gating Structure



$\overline{A * B}$

$\overline{O} = \overline{[A_1 * B_1]}$

$\overline{A * B}$

Description

```
MACRO XOR
NAND  0OUT 1/1 A1 B1
NAND  A1  0/0  1A  B/
NAND  B1  0/0  2B  A/
NAND  A/  0/0  1A
NAND  B/  0/0  2B
END
```

Use

```
XOR  C  32/17  A  B
```

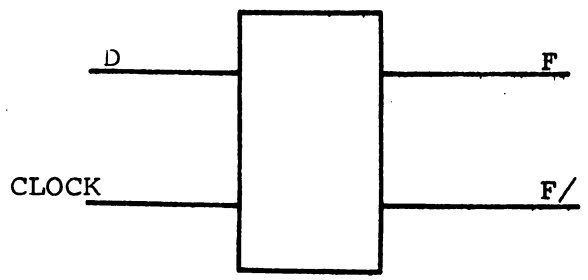
Actual Delays

11.4 Set-Reset Type Flip Flop

See pages 25 and 27 of this manual.

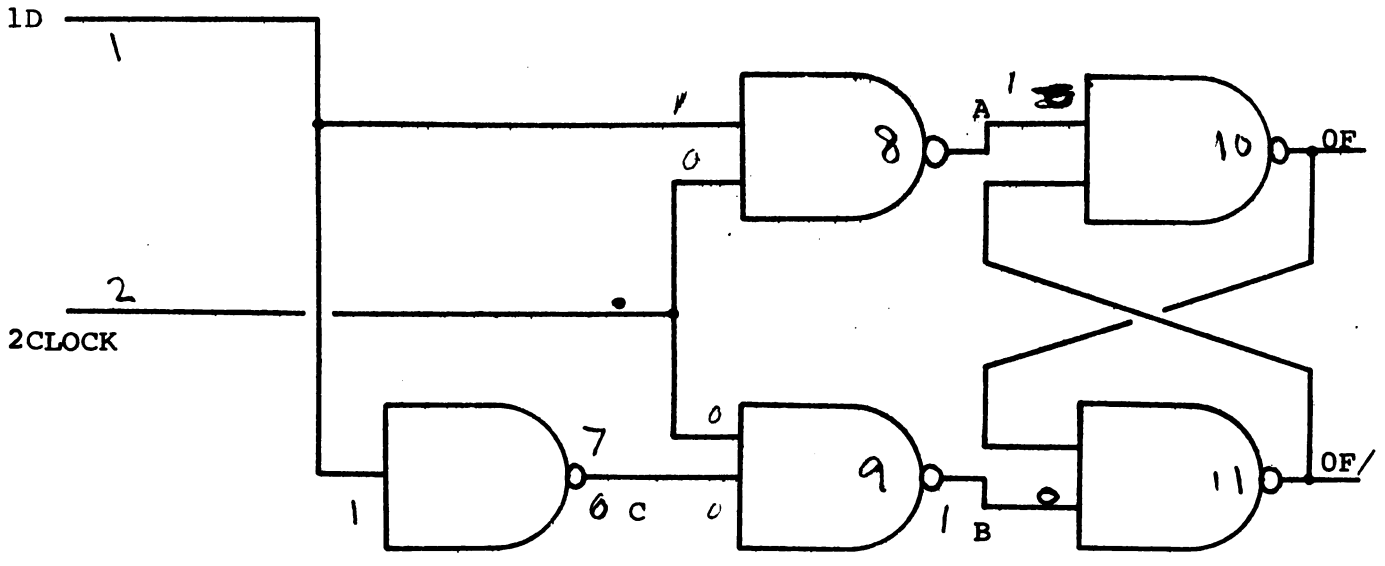
11.5 Clocked D Type Flip Flop

Symbol



Used in buffer registers, shift registers and binary counter applications. Has no ambiguous states.

Gating Structure



Description

```
MACRO CDFLP
NAND  OF 1/1 A OF/
NAND  OF/ 1/1 B OF
NAND  A 0/0 1D 2CLOCK
NAND  B 0/0 C 2CLOCK
NAND  C 0/0 1D
END
```

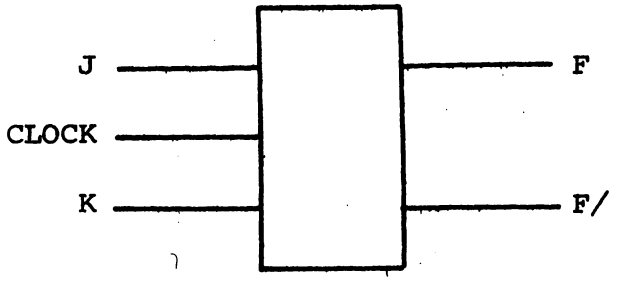
Use

CDFLP F 15/18 D CLOCK

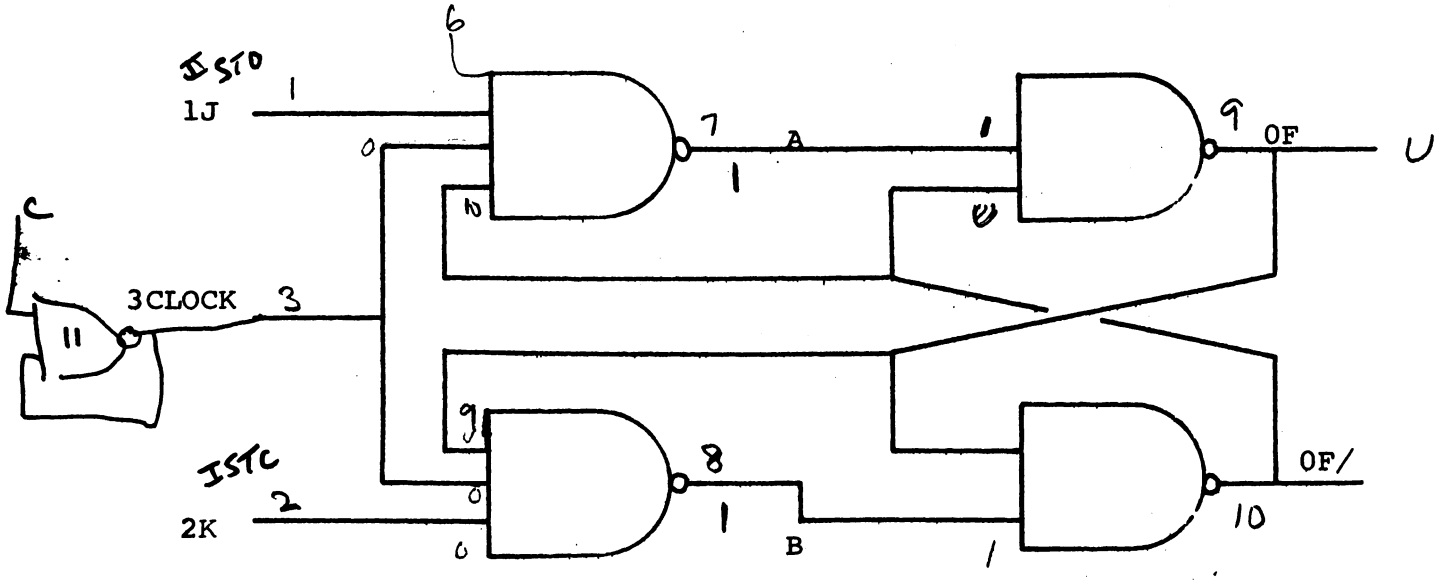
F/ 15/18 ← Actual Delay Times for Flip Flop

11.6 JK Flip Flop (JKFLP)

Symbol



Gating Structure



Description

```
MACRO JKFLP

NAND OF 1/1 A OF/
NAND OF/ 1/1 B OF
NAND A 1/1 1J 3CLOCK OF/
NAND B 1/1 2K 3CLOCK OF
END
```

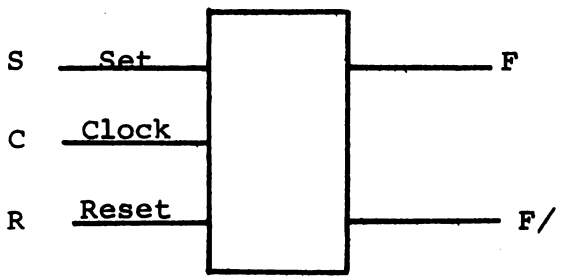
Use

JKFLP F 15/18 J K CLOCK

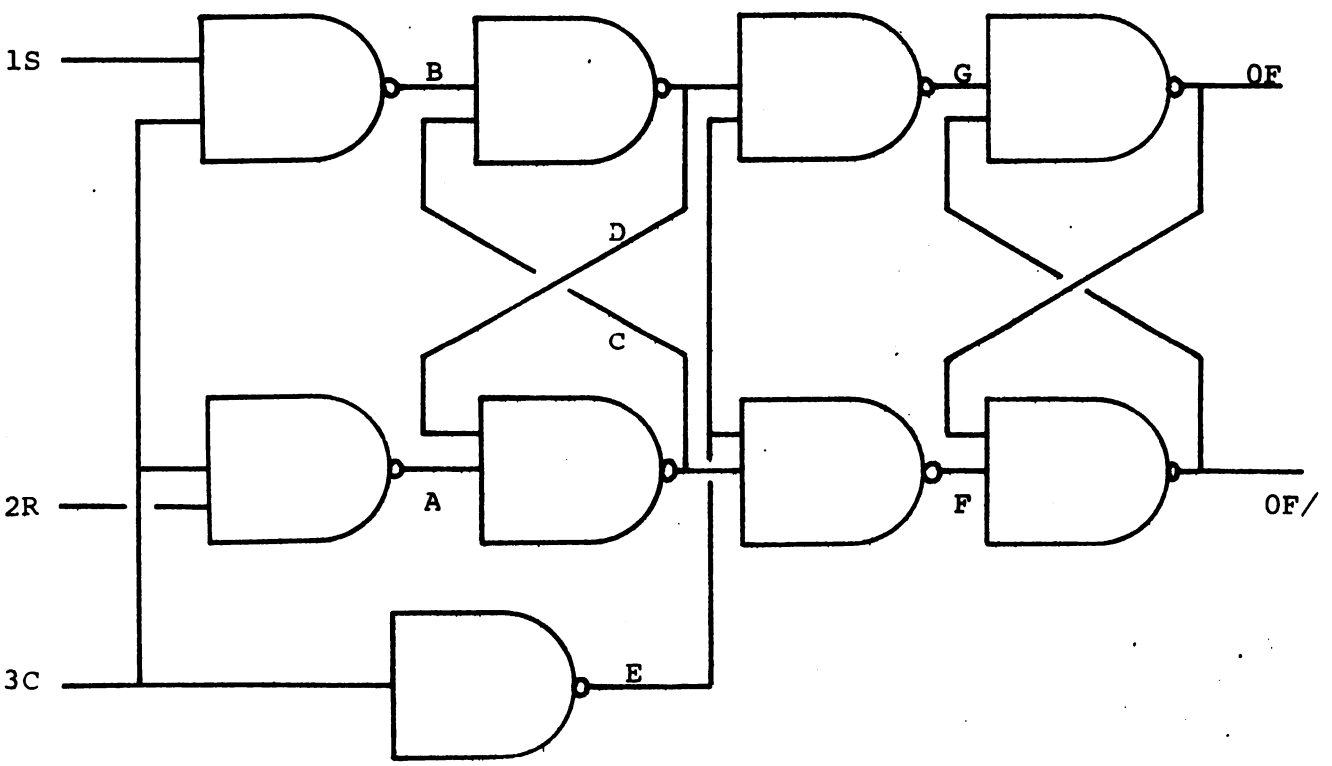
F/ 15/18

11.7 Master Slave Flip Flop

Symbol



Gating Structure



Description

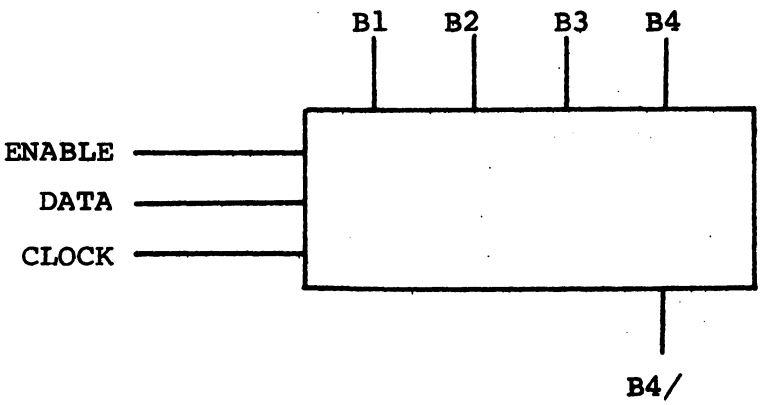
```
MACRO MSFLP
NAND OF 1/1 OF/ G
NAND OF/ 1/1 OF F
NAND G 0/0 E D
NAND F 0/0 E C
NAND D 1/1 C B
NAND C 1/1 D A
NAND B 0/0 1S 3C
NAND A 0/0 2R 3C
NAND E 0/0 3C
END
```

Use

```
MSFLP F 15/37 S R C
       F/ 15/37
```

11.8 Four Bit Shift Register

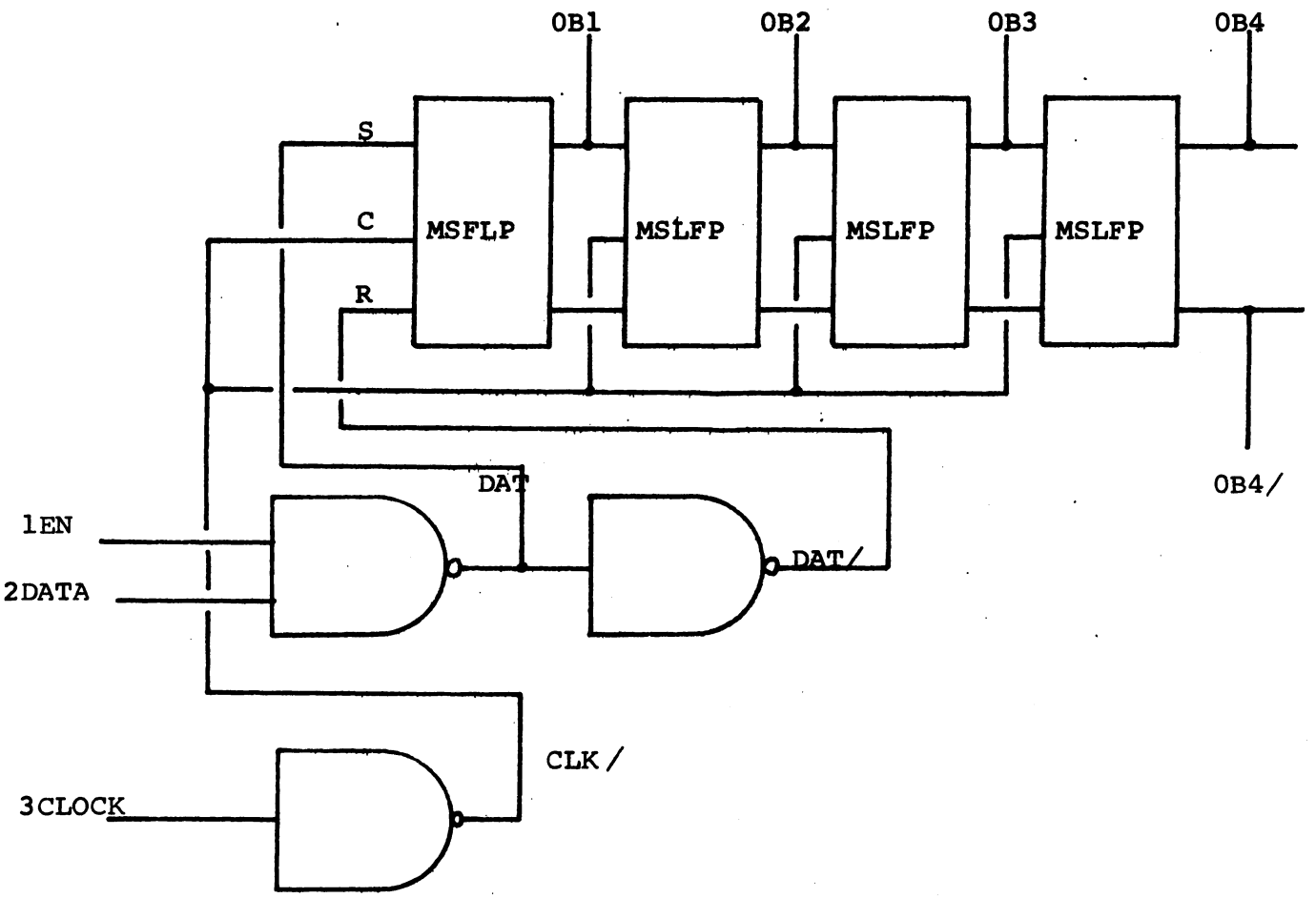
Symbol



$B4/ = \overline{B4}$

Serial load, parallel or serial out.

Gating Structure



Description

```
MACRO FBSR
NAND DAT 0/0 1EN 2DATA
NAND DAT/ 0/0 DAT
NAND CLK/ 0/0 3CLOCK
MSFLP 0B1 1/1 DAT DAT/ CLK/
        B1/ 1/1
MSFLP 0B2 1/1 0B1 B1/ CLK/
        B2/ 1/1
MSFLP 0B3 1/1 0B2 B2/ CLK/
        B3/ 1/1
MSFLP 0B4 1/1 0B3 B3/ CLK/
       0B4/ 1/1
END
```

Use

```
FBSR B1 5/8 ENABLE DATA CLOCK
     B2 5/8
     B3 5/8
     B4 5/8
     B4/ 5/8
```

↖Actual delay times.