(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 185 676 B1**

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: **18.09.91**

(51) Int. Cl.5: **G06F 13/40**, G06F 13/36, G06F 9/00

(21) Application number: **85901904.4**

(22) Date of filing: **12.04.85**

(86) International application number:
PCT/US85/00656

(87) International publication number:
WO 86/00436 (16.01.86 86/02)

(54) DATA PROCESSOR HAVING DYNAMIC BUS SIZING.

(30) Priority: **26.06.84 US 624660**

(43) Date of publication of application:
02.07.86 Bulletin 86/27

(45) Publication of the grant of the patent:
18.09.91 Bulletin 91/38

(84) Designated Contracting States:
DE FR GB IT NL

(56) References cited:
GB-A- 2 075 226
US-A- 4 181 974
US-A- 4 453 211
US-A- 4 503 495

(73) Proprietor: **MOTOROLA, INC.**
1303 East Algonquin Road
Schaumburg, IL 60196(US)

(72) Inventor: **MOTHERSOLE, David, S.**
903 Huntridge Drive
Austin, TX 78758(US)
Inventor: **CRUDELE, Lester, M.**
69 Hill Road
Groton, MA 01450(US)
Inventor: **TIETJEN, James, L.**
3423 Rosefinch Trail
Austin, TX 78746(US)
Inventor: **THOMPSON, Robert, R.**
11104 Township Cove
Austin, TX 78759(US)

(74) Representative: **Holt, Michael et al**
MOTOROLA European Intellectual Property Operations Jays Close Viables Industrial Estate
Basingstoke Hampshire RG22 4PD(GB)

EP 0 185 676 B1

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid (Art. 99(1) European patent convention).

Description

Cross Reference to Related Applications

Related subject matter is disclosed in U.S. Patent Application Serial Number SC-05421, entitled DATA PROCESSOR HAVING MULTIPLE BUS CYCLE OPERAND CYCLES, inventors David S. Mothersole, Jay Alen Hartvigsen and Robert R. Thompson, filed on xx June 1984, and assigned to the Assignee hereof

Field of the Invention

The present invention relates generally to data processors and, more particularly, to a data processor which is capable of communicating with system resources having different data port sizes.

Background of the Invention

In general, data processors communicate with all of the different types of system resources using the same communication bus. For example, the data processor would communicate with both the primary and the secondary memories using the same communication bus. Similarly, the same communication bus would be used to communicate with input/output controllers and the like. If communication is necessary in a particular system with a resource which is unable to utilize the existing common communication bus because of data port size incompatibility, an interface adapter must be employed to buffer data transfers between the processor's bus and that of the resource. In addition to adding additional circuitry to the system, the interface devices require that the data processor provide specific directions for each such transfer.

United Kingdom Patent No. GB-A-2 075 226 discloses an interface unit provided for interfacing between a CPU in a central subsystem and a plurality of memories via a system bus. Each memory may store operands having fixed widths of either: i) a double word ii) a single word, double pull, or iii) a single word, single pull. The CPU selectively issues double word memory calls. The interface unit transfers these calls to the memory and receives type information therefrom. If the memory call is to a memory having a double word stored, the double word is received in the same system clock cycle. Whenever one of the other types of memory is called, interface unit functions to store the first word and either awaits the second word in subsequent cycles or increments an address and sends the incremented address to memory to obtain the second word. At least three system clock cycles are required to retrieve a double wide word from a memory storing a single word, whether the memory is double pull or single pull. Although the system bus has a constant bit width, single or double width words may be formed by the interface unit.

Summary of the Invention

Accordingly, it is an object of the present invention to provide a data processor having a bus controller which is able to communicate with any of a plurality of system resources having different data port sizes, using a communication bus which is a common multiple of these port sizes.

Most generally, it is an object of the present invention to provide the capability in any bus master to communicate with any of a plurality of available bus slaves having different data port sizes, using a communication bus which is a common multiple of these port sizes.

This object is achieved as specified in claims 1 and 4.

Brief Description of the Drawings

Figure 1 is a block diagram of a data processor having a bus controller constructed in accordance with the present invention.

Figure 2 is a block diagram of the address bus interface of the data processor of Figure 1.

Figure 3 is a block diagram of the A0 and A1 interfaces of the address bus interface of Figure 2.

Figure 4 is a detailed schematic of the address restore portion of the A0/A1 interface of Figure 3.

Figure 5 is a detailed schematic of the A0 interface of Figure 3, the A1 interface being identical.

Figure 6 is a block diagram of the A2 through A16 interfaces of the address bus interface of Figure 2.

Figure 7 is a block diagram of the A17 through A32 interfaces of the address bus interface of Figure 2.

Figure 8 is a detailed schematic of the A2 interface of Figure 6, the A4, A6, A8, A10, A12, A14, A16, A18, A20, A22, A24, A26, A28, A30 and A32 interfaces being identical.

Figure 9 is a detailed schematic of the A3 interface of Figure 6, the A5, A7, A9, A11, A13, A15, A17, A19, A21, A23, A25, A27, A29 and A31 interfaces being identical.

Figure 10 is a block diagram of the data bus interface of the data processor of Figure 1.

Figure 11 is a detailed schematic diagram of the internal data bus precharge portion of the data bus interface of Figure 10.

Figure 12 is a detailed schematic diagram of the input enable portion of the data bus interface of Figure 10.

Figure 13 is a block diagram of the D0 through D7 interfaces of the data bus interface of Figure 10.

Figure 14 is a detailed schematic diagram of the control for the D0-D7 interfaces of Figure 13.

Figure 15 is a block diagram of the D8 through D15 interfaces of the data bus interface of Figure 10.

Figure 16 is a block diagram of the control for the D8-D23 interfaces of the data bus interface of Figure 15.

Figure 17 is a detailed schematic diagram of the control for the D8-D15 interfaces of the data bus interface of Figure 16.

Figure 18 is a detailed schematic diagram of the control for the D16-D23 interfaces of the data bus interface of Figure 16.

Figure 19 is block diagram of the D16 through D23 interfaces of the data bus interface of Figure 10.

Figure 20 is a block diagram of the D24 through D31 interfaces of the data bus interface of Figure 10.

Figure 21 is a detailed schematic diagram of the D31 interface of the data bus interface of Figure 20, all of the other interfaces D0 through D30 being identical.

Figure 22 is a detailed schematic diagram of the control of the D24-D31 interfaces of Figure 20.

Figure 23 is a block diagram of the bus controller of the data processor of Figure 1.

Figure 24 is a detailed schematic diagram of the size control portion of the bus controller of Figure 23.

Figure 25 is a detailed schematic diagram of the byte latch control of the bus controller of Figure 23.

Figure 26 is a detailed schematic diagram of the next address control of the bus controller of Figure 23.

Figure 27 is a detailed schematic diagram of the data address buffers of the bus controller of Figure 23.

Figure 28 is a block diagram of the microsequencer of the bus controller of Figure 23.

Figure 29 is a detailed schematic diagram of the data size input synchronizer of the microsequencer of Figure 28.

Figure 30 is a detailed schematic diagram of the termination control of the microsequencer of Figure 28.

Figure 31 is a detailed schematic diagram of the state control of the microsequencer of Figure 28.

Figure 32 is a detailed schematic diagram of the start bus cycle control of the microsequencer of Figure 28.

Description of the Invention

Shown in Figure 1 is a data processor 10 comprising a central processing unit (CPU) 12, a bus controller 14, an address bus interface 16, a data bus interface 18, and a storage device 20. In general, the CPU 12 executes a user specified sequence of instructions, each of which is comprised of one or more 16-bit words. Each of these instructions must be read from the storage device 20 in the appropriate sequence. In the course of executing each such instruction, the CPU 12 may be required to perform a specified operation upon an 8-bit byte, a 16-bit word or a 32-bit long word. Most of these data operands must be either read from or written to the storage device 20. In order to assure optimum performance on long word operations, the CPU 12 is provided with a 32-bit data port. On the other hand, it may be advantageous (or unavoidable) that the storage device 20 have a data port which is smaller than that of the CPU 12. Even when the port sizes are the same, the operand required by the CPU 12 may still reside at an address within the storage device 20 which does not align evenly with the data port of that particular storage device 20. It is the responsibility of the bus controller 14 to coordinate the activities of the address bus interface 16 and the data bus interface 18 in actually transfering the requested data or instruction operands between the CPU 12 and the storage device 20, regardless of operand misalignment or any mismatch between the port sizes of the CPU 12 and the storage device 20.

In general, the CPU 12 requests an operand transfer by asserting an OPeration-PENDing signal (OPPEND) to the bus controller 14. Simultaneously, the CPU 12 will provide a Read/Write-ReQuest signal (RQRW) indicating the direction of operand transfer and a Requested-Size signal (*RQS[0:1]) indicating the size of the operand to be transferred. The CPU 12 also provides a 32-bit Address (A[0:31]) to or from which the operand is to be transferred on a 32-bit Internal Address Bus (*IAB[0:31]).

Assuming for the moment that the CPU 12 has requested an operand write, the bus controller 14 will

briefly assert a Start-OPerand-CYcle signal (SOPCY) directing the address bus interface 16 to latch the operand address on the *IAB. Simultaneously, the bus controller 14 will negate a TRISTATE signal (*TRISTATE) to enable the address bus interface 16 to transfer the address to the storage device 20 on a 32-bit external ADDRESS BUS (ADDRESSBUS). A brief time later, the bus controller 14 will assert an Address-Strobe signal (*AS) to the storage device 20 indicating that a valid operand address is on the ADDRESSBUS.

The bus controller 14 will then assert a Data-Output-Buffer-to-Internal-Data-Bus signal (DOBIDB) directing the CPU 12 to provide the operand to the data bus interface 18 on a 32-bit Internal Data Bus (IDB-[0:31]). The bus controller 14 will also provide to the data bus interface 18: a CURrent-Size signal (*CURS-[0:1]) indicating the size of the operand to be placed on the DATABUS; a DATA-ADDress signal (DATAADD[0:1]) corresponding to the two low order address bits A0 and A1 of the address on the ADDRESSBUS; and a CURrent-Read/Write signal (*CURRW;CURRW) corresponding to the current state of the RW signal.

In the illustrated form, the IDB is partitioned into four bytes: I0 consisting of internal Data bits D31 through D24; I1 consisting of Data bits D23 through D16; I2 consisting of internal Data bits D15 through D8; and I3 consisting of internal Data bits D7 through D0. Depending upon the size of the operand being transferred, these internal bytes must be selectively coupled to the external DATABUS which is also partitioned into four bytes: D0 consisting of external Data bits D31 through D24; E1 consisting of external Data bits D23 through D16; E2 consisting of external Data bits D15 through D8; and E3 consisting of external Data bits D7 through D0.

Depending upon the current operand size (*CURS[0:1]) and the current operand address (DATAADD-[0:1]), the data bus interface 18 will provide the available bytes on the IAB to the appropriate bytes on the DATABUS as follows:

CURS		DATAADD		DATABUS
0	1	0	1	E0 E1 E2 E3
0	0	0	0	|I0|I1|I2|I3|
0	0	0	1	|I0|I0|I1|I2|
0	0	1	0	|I0|I1|I0|I1|
0	0	1	1	|I0|I0|I1|I0|
0	1	x	x	|I3|I3|I3|I3|
1	0	x	0	|I2|I3|I2|I3|
1	0	x	1	|I2|I2|I3|I2|
1	1	0	0	|I1|I2|I3|I0|
1	1	0	1	|I1|I1|I2|I3|
1	1	1	0	|I1|I2|I1|I2|
1	1	1	1	|I1|I1|I2|I1|

where small "|" indicates a connection for convenience rather than a required connection. After the data bus interface 18 has had sufficient time to establish the operand on the DATABUS, the bus controller 14 will assert a Data-Strobe signal (*DS) to advise the storage device 20 that the operand on the DATABUS is valid.

Upon receiving the Address-Strobe (*AS), the storage device 20 will decode the address on the ADDRESSBUS. If the address is determined to be within the address range for that particular storage device 20, the storage device 20 will prepare to latch the operand. To best facilitate this, the storage device 20 has its data port connected to the DATABUS so that the high order byte (00) of the data port of the storage device 20 will be aligned with the high order byte (E0) of the DATABUS as follows:

DATA PORT	E0 E1 E2 E3
32-bits	100101102103
16-bits	10010 11
8-bits	1001

Thus, upon receiving the Data-Strobe (*DS), the storage device 20 will always be able to latch at least the high order portion of the operand during the first bus cycle of every operand cycle. After successfully capturing the respective portion of the operand, the storage device 20 will provide a Data-transfer-and-Size-ACKnowledge signal (*DSACK[0:1]) acknowledging the operand transfer. In addition, however, the *DSACK signal also indicates the size of the data port of that particular storage device 20 as follows:

DSACK		WIDTH OF
0	1	DATA PORT
0	0	(bus cycle incomplete)
0	1	8-bits
1	0	16-bits
1	1	32-bits

Using the known operand Size (S[0:1]) and CURrent-ADdress (*CURAD[0:1]), and the size of the port (*DSACK[0:1]), the bus controller 14 can determine the size of residual portion of the operand, if any, which has not yet been received, as follows:

current				returned		:	next		cycle
S1	S0	A1	A0	DSACK1	DSACK0	:	S1	S0	done?
0	1	0	0	0	0	:	x	x	i
0	1	0	0	0	1	:	x	x	y
0	1	0	0	1	0	:	x	x	y
0	1	0	0	1	1	:	x	x	y
0	1	0	1	0	0	:	x	x	i
0	1	0	1	0	1	:	x	x	y
0	1	0	1	1	0	:	x	x	y
0	1	0	1	1	1	:	x	x	y
0	1	1	0	0	0	:	x	x	i
0	1	1	0	0	1	:	x	x	y
0	1	1	0	1	0	:	x	x	y
0	1	1	0	1	1	:	x	x	y
0	1	1	1	0	0	:	x	x	i
0	1	1	1	0	1	:	x	x	y
0	1	1	1	1	0	:	x	x	y
0	1	1	1	1	1	:	x	x	y
1	0	0	0	0	0	:	x	x	i
1	0	0	0	0	1	:	0	1	n
1	0	0	0	1	0	:	x	x	y
1	0	0	0	1	1	:	x	x	y
1	0	0	1	0	0	:	x	x	i
1	0	0	1	0	1	:	0	1	n
1	0	0	1	1	0	:	0	1	n
1	0	0	1	1	1	:	x	x	y
1	0	1	0	0	0	:	x	x	i
1	0	1	0	0	1	:	0	1	n
1	0	1	0	1	0	:	x	x	y
1	0	1	0	1	1	:	x	x	y
1	0	1	1	0	0	:	x	x	i
1	0	1	1	0	1	:	0	1	n
1	0	1	1	1	0	:	0	1	n
1	0	1	1	1	1	:	0	1	n
1	1	0	0	0	0	:	x	x	i

current				returned		:	next		cycle
S1	S0	A1	A0	DSACK1	DSACK0	:	S1	S0	done?
1	1	0	0	0	1	:	1	0	n
1	1	0	0	1	0	:	0	1	n
1	1	0	0	1	1	:	x	x	y
1	1	0	1	0	0	:	x	x	i
1	1	0	1	0	1	:	1	0	n
1	1	0	1	1	0	:	1	0	n
1	1	0	1	1	1	:	x	x	y
1	1	1	0	0	0	:	x	x	i
1	1	1	0	0	1	:	1	0	n
1	1	1	0	1	0	:	0	1	n
1	1	1	0	1	1	:	0	1	n
1	1	1	1	0	0	:	x	x	i
1	1	1	1	0	1	:	1	0	n
1	1	1	1	1	0	:	1	0	n
1	1	1	1	1	1	:	1	0	n
0	0	0	0	0	0	:	x	x	i
0	0	0	0	0	1	:	1	1	n
0	0	0	0	1	0	:	1	0	n
0	0	0	0	1	1	:	x	x	y
0	0	0	1	0	0	:	x	x	i
0	0	0	1	0	1	:	1	1	n
0	0	0	1	1	0	:	1	1	n
0	0	0	1	1	1	:	0	1	n
0	0	1	0	0	0	:	x	x	p
0	0	1	0	0	1	:	1	1	n
0	0	1	0	1	0	:	1	0	n
0	0	1	0	1	1	:	1	0	n
0	0	1	1	0	0	:	x	x	i
0	0	1	1	0	1	:	1	1	n
0	0	1	1	1	0	:	1	1	n
0	0	1	1	1	1	:	1	1	n

where:

x => don't care
i => bus cycle incomplete
y => operand cycle complete
n => operand cycle incomplete

Thus, for example, if the port size of the storage device 20 is the same as the size of the DATABUS or if the size of the operand is less than or equal to the port size of the storage device 20, the bus controller 14 will know that all of the operand has been received and that the operand cycle can be terminated. At this time, if another bus master (not shown) is awaiting use of the communication bus, the bus controller 14 will assert the *TRISTATE signal to force the address bus interface 16 to remove the address from the ADDRESSBUS. In any event, the bus controller 14 will then assert a Tristate-Data-Bus signal (*TSDB) to force the data bus interface 18 to remove the operand from the DATABUS. Simultaneously, the bus controller 14 will assert an OPerand-CYcle-COMplete signal (OPCYCOM) to advise the CPU 12 that the requested operand write has been completed. Finally, the bus controller 14 will terminate the bus cycle by negating the Address and Data Strobes (*AS and *DS). In response, the storage device 20 will withdraw the *DSACK signal. At this time, the communication bus again becomes available for use by the CPU 12 or any other bus master (not shown) which may be present in the system.

If additional bus cycles are required to complete the operand cycle, the bus controller 14 will recompute the two low order bits A0 and A1 of the address of the residual operand as follows:

CURAD		DSACK		: NXTA		address
1	0	1	0	: 1	0	rollover?
0	0	0	0	: x	x	p
0	0	0	1	: 0	1	n
0	0	1	0	: 1	0	n
0	0	1	1	: x	x	x

CURAD		DSACK		: NXTA		address
1	0	1	0	: 1	0	rollover?
0	1	0	0	: x	x	p
0	1	0	1	: 1	0	n
0	1	1	0	: 1	0	n
0	1	1	1	: 0	0	y
1	0	0	0	: x	x	p
1	0	0	1	: 1	1	n
1	0	1	0	: 0	0	y
1	0	1	1	: 0	0	y
1	1	0	0	: x	x	p
1	1	0	1	: 0	0	y
1	1	1	0	: 0	0	y
1	1	1	1	: 0	0	y

where:

- x =) don't care
- p =) bus cycle incomplete
- n =) no address rollover
- y =) address rollover.

The bus controller 14 will then provide a NeXT-Address signal (NXTA[0:1]) to the address bus interface 16 indicating the new low order address bits A0 and A1. If the communication bus has been used by a different bus master (not shown) since the previous bus cycle of the current operand cycle, the bus controller 14 will

assert an Address-Restore signal (ARESTORE) requesting the address bus interface 16 to restore the original higher order address bits (*IAD[2:31]), but use the two new low order address bits (NXTA[0:1]). On the other hand, if the new address bits have rolled over, the bus controller 14 will assert an INCrement- A2-through-A31 signal (INCA2A31) requesting the address bus interface 16 to increment the original higher order address bits (*IAD[2:31]), and use the incremented address together with the two new low order address bits (NXTA[0:1]). In anticipation of this request, the address bus interface 16 has already incremented the higher order address bits A2-A31. Thus, the bus controller 14 can immediately assert a Start-NeXT-Bus-Cycle signal (SNXTBC) requesting the address bus interface 16 to start the next bus cycle using the new address. From this point on, the bus controller 14 cooperates with the address bus interface 16 and the data bus interface 18 as described above. If necessary, this sequence is repeated until all of the requested operand has been received and latched into the storage device 20.

In general, the write operand cycle can be summarized with respect to any bus master writing an operand to a bus slave as follows:

BUS MASTER:

1) Set Read/Write (RW) to Write
2) Drive Address on ADDRESSBUS
3) Drive Size (S[0:1])
4) Assert Address-Strobe (*AS)
5) Drive operand byte(s) on DATABUS
6) Assert Data-Strobe (*DS)

BUS SLAVE:

1) Decode Address on ADDRESSBUS
2) Latch operand byte(s) on DATABUS
3) Assert Data-transfer-and-Size-ACKnowledge (*DSACK[0:1])

BUS MASTER:

7) Negate Data-Strobe (*DS)
8) Negate Address-Strobe (*AS)
9) Remove operand byte(s) from DATABUS

BUS SLAVE

4) Negate Data-transfer-and-Size-ACknowledge (*DSACK[0:1])

BUS MASTER:

10) If all operand byte(s) not received, recompute Address and Size and return to 1)
11) Otherwise, operand cycle complete

Assume now that the CPU 12 has requested an operand read. As in the write case, the bus controller 14 will again briefly assert the Start-OPerand-CYcle signal (SOPCY) directing the address bus interface 16 to latch the operand Address on the *IAB. Simultaneously, the bus controller 14 will negate *TRISTATE (if then asserted) to enable the address bus interface 16 to transfer the Address to the storage device 20 on the ADDRESSBUS. The bus controller 14 will also provide RW in the Read state.

A brief time later, the bus controller 14 will assert *AS to the storage device 20 indicating that a valid operand Address is on the ADDRESSBUS. Internally, the bus controller 14 will assert a Data-bus-Start-PreCHarGe signal (DSPCHG) directing the data bus interface 18 to start precharging the IDB. In addition, the bus controller 14 will pass the current operand size (*CURS[0:1]), the current low order address bits (DATAADD[0:1]), and the current direction of operand transfer (*CURRW;CURRW) to the data bus interface 18.

Upon receiving *AS, the storage device 20 will decode the address on the ADDRESSBUS. If the address is determined to be within the address range for that particular storage device 20, the storage device 20 will provide on the DATABUS as much of the requested operand as possible for the port size of that particular storage device 20. The storage device 20 will then provide *DSACK to indicate that the

requested operand (or at least a portion thereof) is available on the DATABUS. As explained above, the *DSACK signal also indicates the size of the data port of that particular storage device 20.

Depending upon the size of the port (*IDSACK[0:1]), the current operand size (*CURS[0:1]) and address (DATADDD[0:1]), the data bus interface 18 can determine which bytes (E[0:3]) of the DATABUS are valid, as follows:

IDSACK		CURS		DATAADD			:	valid E bytes			
1	0	1	0	1	0	RW	:	0	1	2	3
x	x	x	x	x	x	0	:	0	0	0	0
x	x	0	1	0	0	1	:	1	0	0	0
0	x	0	1	0	1	1	:	1	0	0	0
1	x	0	1	0	1	1	:	0	1	0	0
0	x	0	1	1	0	1	:	1	0	0	0
1	0	0	1	1	0	1	:	1	0	0	0
1	1	0	1	1	0	1	:	0	0	1	0
0	x	0	1	1	1	1	:	1	0	0	0
1	0	0	1	1	1	1	:	0	1	0	0
1	1	0	1	1	1	1	:	0	0	0	1
x	x	1	0	0	0	1	:	1	1	0	0
0	x	1	0	1	0	1	:	1	x	0	0
1	0	1	0	1	0	1	:	1	1	0	0
1	1	1	0	1	0	1	:	0	0	1	1
0	x	1	0	0	1	1	:	1	0	x	0
1	x	1	0	0	1	1	:	0	1	1	0
0	x	1	0	1	1	1	:	1	0	x	0
1	0	1	0	1	1	1	:	0	1	x	0
1	1	1	0	1	1	1	:	0	0	x	1
x	x	0	0	0	0	1	:	1	1	1	1
0	x	0	0	0	1	1	:	1	0	x	x
1	x	0	0	0	1	1	:	0	1	1	1
0	x	0	0	1	0	1	:	1	x	0	0
1	0	0	0	1	0	1	:	1	1	0	0
1	1	0	0	1	0	1	:	0	0	1	1
0	x	0	0	1	1	1	:	1	0	x	0
1	0	0	0	1	1	1	:	0	1	x	0
1	1	0	0	1	1	1	:	0	0	x	1
0	x	1	1	0	1	1	:	1	0	x	x
1	x	1	1	0	1	1	:	0	1	1	1
0	x	1	1	1	0	1	:	1	x	0	0
1	0	1	1	1	0	1	:	1	1	0	0
1	1	1	1	1	0	1	:	0	0	1	1

where:

IDSACK		CURS		DATAADD			: valid E bytes			
1	0	1	0	1	0	RW	: 0	1	2	3
0	x	1	1	1	1	1	: 1	0	x	0
1	0	1	1	1	1	1	: 0	1	x	0
1	1	1	1	1	1	1	: 0	0	x	1
x	x	1	1	0	0	1	: 1	1	1	x

x => don't care.

Depending upon the current operand size (*CURS[0:1]) and the current operand address (DATAADD-(0:1]), the data bus interface 18 will couple the valid byte(s) on the DATABUS to the proper byte(s) of the IDB as described above. Using just the current operand size (S[0:1]), the bus controller 14 can then provide a Data-Bus-INput:Latch-Byte signal (DBINLB[0:3]) indicating which bytes (I[0:3]) of the IDB are valid, as follows:

S1	S0	:	I0	I1	I2	I3
0	0	:	1	1	1	1
0	1	:	0	0	0	1
1	0	:	0	0	1	1
1	1	:	0	1	1	1

In response to the DBINLB signal, the CPU 12 will latch the valid bytes provided by the data bus interface 18 on the IDB into the appropriate destination register (not shown).

Using the current operand size (S[0:1]) and address (*CURAD[0:1]) and the size of the port (*DSACK-[0:1]), the bus controller 14 can determine how much of the requested operand remains to be provided by the storage device 20, in a similar manner to that described above in the write case. Thus, for example, if the port size of the storage device 20 is the same as the size of the DATABUS or if the size of the operand is less than or equal to the port size of the storage device 20, the bus controller 14 will know that all of the operand has been received and that the operand cycle can be terminated. In this event, the bus controller 14 will terminate the bus cycle by negating *AS and *DS. Simultaneously, the bus controller 14 will assert *TSDB to force the data bus interface 18 to decouple from the DATABUS. The bus controller 14 will also remove DBINLB and then assert OPCYCOM to advise the CPU 12 that the requested operand read has been completed. A brief time later, if another bus master (not shown) has requested the use of the communication bus, the bus controller 14 will assert *TRISTATE to force the address bus interface 16 to remove the address from the ADDRESSBUS. In response to the negation of *AS and *DS, the storage device 20 will withdraw the operand byte(s) from the DATABUS, and then terminate *DSACK. At this time, the communication bus again becomes available for use by the CPU 12 or any other bus master (not shown) which may be present in the system.

If additional bus cycles are required to complete the operand cycle, the bus controller 14 will recompute the two low order bits A0 and A1 of the address of the residual operand as described above. The bus controller 14 will then provide the address bus interface 16 with the new low order address bits A0 and A1 (NXTA[0:1]). If the communication bus has been used by another bus master (not shown) since the previous bus cycle of the current operand cycle, the bus controller 14 will assert ARESTORE requesting the address bus interface to restore the original higher order address bits (*IAD[2:31]), but use the two new low order address bits (NXTA[0:1]). On the other hand, if the new address bits have rolled over, the bus controller 14 will assert INCA2A31 requesting the address bus interface 16 to increment the original higher order address bits (*IAD[2:31]), and use the resultant address together with the two new low order address bits (NXTA[0:1]-). As indicated before, the address bus interface 16 has already incremented the higher order address bits A2-A31 in anticipation of this request. Thus, the bus controller 14 can immediately assert (SNXTBC) requesting the address bus interface 16 to start the next bus cycle using the new address. From this point

on, the bus controller 14 cooperates with the address bus interface 16 and the data bus interface 18 as described above. If necessary, this sequence is repeated until all of the requested operand has been received and latched into the CPU 12.

In general, the read cycle can be summarized with respect to any bus master reading an operand from a bus slave as follows:

BUS MASTER:

1) Set Read/Write to Read
2) Drive address on ADDRESSBUS
3) Drive Size (S[0:1])
4) Assert Address-Strobe (*AS)
5) Assert Data-Strobe (*DS)

BUS SLAVE:

1) Decode address on ADDRESSBUS
2) Drive operand byte(s) on DATABUS
3) Assert Data-transfer-and-Size-ACKnowledge (*DSACK[0:1])

BUS MASTER:

6) Latch operand byte(s) into register
7) Negate Data-Strobe (*DS)
8) Negate Address-Strobe (*AS)

BUS SLAVE

4) Remove operand byte(s) from DATABUS
5) Negate Data-transfer-and-Size-ACKnowledge (*DSACK[0:1])

BUS MASTER:

9) If all operand byte(s) not received, recompute Address and Size and return to 1)
10) Otherwise, operand cycle complete

As shown in Figure 2, the preferred embodiment of the address bus interface 16 is comprised of an A0A1 interface 22, an A2A16 interface 24, and an A17A31 interface 26. As can be seen in Figure 3, the A0A1 interface 22 is comprised of an ADDress RESTore 28, an A0 interface 30 and an A1 interface 32 which is identical to the A0 interface 30. Detailed schematic diagrams of the ADDREST 28 and the A0 interface 30 are shonw in Figures 4 and 5, respectively. As shown in Figure 6, the A2A16 interface 24 is comprised of A2 through A16 interfaces 34 through 62, respectively. Similarly, the A17A31 interface 26 is comprised of A17 through A31 interfaces 64 through 92, respectively. A detailed schematic diagram is shown in Figure 8 of the A2 interface 34, the A4, A6, A8, A10, A12, A14, A16, A18, A20, A22, A24, A26, A28, and A30 interfaces 38, 42, 46, 50, 54, 58, 62, 66, 70, 74, 78, 82, 86 and 90, respectively, being identical. Similarly, a detailed schematic diagram is shown in Figure 9 of the A3 interface 36, the A5, A7, A9, A11, A13, A15, A17, A19, A21, A23, A25, A27, A29, and A31 interfaces 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88 and 92, respectively, being identical.

As shown in Figure 10, the preferred embodiment of the data bus interface 18 is comprised of an Internal Data Bus PreCHarGe (IDBPCHG) 94, and INPUT ENable (INPUTEN) 96, a D0-D7 interface 98, a D8-D15 interface 100, a D16-D23 interface 102 and a D24-D31 interface 104. A detailed schematic diagram of the IDBPCHG 94 is shown in Figure 11. A detailed schematic diagram of the INPUTEN 96 is shown in Figure 12. As can be seen in Figure 13, the D0-D7 interface 98 is comprised of a D0-D7 ConTroL (D07CTL) 106, and D0 through D7 interfaces 108 through 122, respectively. A detailed schematic diagram of the D07CTL 106 is shown in Figure 14. As can be seen in Figure 15, the D8-D15 interface 100 is comprised of a D8-D23 ConTroL (D823CTL) 124, and D8 through D15 interfaces 126 through 140, respectively. As shown in Figure 16, the D823CTL 124 is comprised of a D8-D15 ConTroL (D815CTL) 142 and a D16-D23 ConTroL (D1623CTL) 144. A detailed schematic diagram of the D815CTL 142 is shown in Figure 17. A detailed schematic diagram of the D1623CTL 144 is shown in Figure 18. As can be seen in Figure 19, the D16-D23

interface 102 is comprised of D16 through D23 interfaces 146 through 160, respectively. As can be seen in Figure 20, the D24-D31 interface 104 is comprised of a D24-D31 ConTroL (D2431CTL) 162, and D24 through D31 interfaces 164 through 178, respectively. A detailed schematic diagram is shown in Figure 21 of the D31 interface 178, the D0 through D30 interfaces 108-122, 126-140, 146-160, and 164-176, respectively, being identical. A detailed schematic diagram of the D2431CTL 162 is shown in Figure 22.

As shown in Figure 23, the bus controller 14 is comprised of a SIZE circuit (SIZE) 180, a Byte LATCH enable circuit (BLATCH) 182, a NeXT ADDress generator (NXT-ADD) 184, a DATA ADDress buffer (DATA ADD) 186, and a MICRO SEQUencer (MICRO SEQU) 188. A detailed schematic diagram of the SIZE circuit 180 is shown in Figure 24. A detailed schematic diagram of the BLATCH 182 is shown in Figure 25. A detailed schematic diagram of the NXT-ADD generator 184 is shown in Figure 26. A detailed schematic diagram of the DATA-ADD buffer 186 is shown in Figure 27. As can be seen in Figure 28, the MICRO-SEQUencer 188 is comprised of a Data Size Input SYNCHronizer (DSISYNCH) 190, a TERMination ConTrol 192, a STATe ConTroL 194, and a STroBe Bus Cycle control (STBBC) 196. A detailed schematic diagram of the DSISYNCH 190 is shown in Figure 29. A detailed schematic diagram of the TERMCTL 192 is shown in Figure 30. A detailed schematic diagram of the STATCTL 194 is shown in Figure 31. A detailed schematic diagram of the STBBC 196 is shown in Figure 32.

As will be clear to those skilled in the art, the CPU 12 may take any of a number of well known forms. For example, the CPU 12 may be constructed along the lines of that described in US Patent Number 4,325,121. On the other hand, the bus controller 14, address bus interface 16 and data bus interface 18 may be readily adapted to perform operand cycles for any of the other well known forms of bus master such as direct memory access controllers and the like. Similarly, although the storage device 20 has been described as being a memory device, the present invention is as readily adaptable to any of the other well known forms of bus slave such as peripheral controllers and the like. In addition, more than one different kind of bus slave may be used together to form a composite storage device 20. In such a system, it is quite possible that a particular operand transfer would span an address transition between two such different bus slaves. Depending upon the system configuration, the data port sizes of these bus slaves may be different. However, since the bus controller 14 recomputes the operand alignment, address and residue size on a bus cycle by bus cycle basis, the operand transfer will still be performed correctly even if the reported port size is different for each bus cycle. Thus, the bus controller 14 is fully capable of dynamically sizing the communication bus on a cycle by cycle basis.

Claims

1. Data processing system (10) which has a data processor (12) capable of communicating with a storage device (20) via a communication bus, the storage device (20) having any of a plurality of differing port sizes, wherein the data processor (12) comprises:

means for providing to the storage device (20) a first signal in a first system clock cycle indicating that an operand is to be communicated using the communication bus, and for communicating the operand between the data processor and the storage device upon receipt of a second signal from said storage device, said data processing system (10) being further characterized by:

communication bus controller means (14) coupled between the data processor (12) and the storage device (20) and to the communication bus for continuously asserting bus mastership during the first, and if necessary, an immediately following second system clock cycle;

said first signal (*RQS, S1, S0) indicating to the communication bus controller (14) a length size of the operand to be communicated;

first means (188, 196) in the communication bus controller means (14) for establishing and continuously maintaining bus mastership for the data processor (12) over the storage device (20) during operand transfer operation;

said storage device (20) providing said second signal (*DSACK0, *DSACK1) to the communication bus controller means (14) indicating the data port size of the storage device (20);

second means (188, 190) in the communication bus controller means (14) for receiving the second signal from the storage device (20) in the first system clock cycle;

third means (18; 94, 96, 98, 100, 102, 104) coupled to the data processor (12), to the communication bus controller means (14) and via the communication bus to the storage device (20) for communicating during the first system clock cycle all, or as much as possible, of the operand to completely communicate the operand in the first system clock cycle by sizing the communication bus to execute the communication according to the first and second signals; and

fourth means (180) in the communication bus controller means coupled to the third means for

providing a third signal (*CURS) to the third means (18) indicating the actual size of the operand placed on the communication bus,

said third means communicating any remainder of said operand exceeding the communication bus's capacity in the first system clock cycle in the immediately following second system clock cycle, by sizing the communication bus to accommodate the remainder, if any.

2. The data processing system (10) of claim 1 wherein said communication bus controller means (14) is arranged to allow said data processor (12) to be able to perform other data processing tasks when the operand is being communicated.

3. The data processing system (10) of claim 1 or 2 wherein the first means (188, 196) provides the first signal at the beginning of each of a plurality of continuous system clock cycles.

4. Method of communicating operands between a data processor and a storage device, in a data processing system (10) which has a data processor (12) capable of communicating with the storage device (20) via a communication bus, the storage device (20) having any of a plurality of differing port sizes, said method comprising the steps of:

providing to the storage device (20) a request signal from the data processor (12) in a first system clock cycle indicating that an operand is to be communicated using the communication bus;

receiving a response signal from the storage device (20) in response to the request signal; and

communicating the operand between the data processor (10) and the storage device (20); said method being further characterised by the steps of:

establishing and continuously maintaining bus mastership during the first and an immediately following second system clock cycle, if necessary, for the data processor (12) over the storage device (20) with a communication bus controller (14) connected between said data processor (12) and said storage device (20),

transferring said request signal (*RQS, S1, S0) communication bus controller (14), said request signal indicating the length size of the operand to be communicated;

receiving with the bus controller (14) said response signal (*DSACK0, *DSACK1) from the storage device (20) in the first system clock cycle, said response signal indicating the data port size of the storage device (20);

providing an operand size signal (*CURS) from the communication bus controller to an interface circuit (18) indicating the actual size of the operand placed on the communication bus, said interface circuit being coupled to the data processor (12), to the communication bus controller (14) and via the communication bus to the storage device (20);

communicating via the communication bus, controlled by the bus controller (14), during the first system clock cycle all, or as much as possible, of the operand by sizing the communication bus to execute the communication according to the request signal and the response signal; and

communicating a remainder, if any, of the operand which does not fit within the communication bus in the first system clock cycle in the immediately following second system clock cycle by sizing the communication bus to accommodate the remainder, if any.

5. The method of communicating operands in the data processing system (10) of claim 4 wherein the request signal is provided at the start of each of a plurality of consecutive system clock cycles.

Revendications

1. Système informatique (10) comportant un processeur de données (12) capable de communiquer avec un dispositif de mémoire (20) par l'intermédiaire d'un bus de communication, le dispositif de mémoire (20) ayant une taille d'accès quelconque parmi un ensemble de tailles d'accès différentes, dans lequel le processeur de données (12) comprend :

des moyens pour appliquer au dispositif de mémoire (20) un premier signal dans un premier cycle d'horloge de système, indiquant qu'un opérande doit être transmis en utilisant le bus de communication, et transmettre l'opérande entre le processeur de données et le dispositif de mémoire à la réception d'un second signal provenant du dispositif de mémoire, ce système informatique (10) étant en outre caractérisé par :

des moyens contrôleurs de bus de communication (14), connectés entre le processeur de données (12) et le dispositif de mémoire (20) et au bus de communication, pour assurer continuellement la

commande du bus pendant le premier cycle d'horloge de système et, si nécessaire, pendant un second cycle d'horloge de système immédiatement suivant;

le premier signal (*RQS, S1, S0) indiquant aux moyens contrôleurs de bus de communication (14) une taille ou longueur de l'opérande à transmettre;

des premiers moyens (188, 196) incorporés dans les moyens contrôleurs de bus de communication (14), pour établir et maintenir continuellement la commande du bus pour le processeur de données (12) plutôt que pour le dispositif de mémoire (20), pendant une opération de transfert d'opérande;

le dispositif de mémoire (20) appliquant le second signal (*DSACK0, *DSACK1) aux moyens contrôleurs de bus de communication (14) pour indiquer la taille de l'accès de données du dispositif de mémoire (20);

des seconds moyens (188, 190) dans les moyens contrôleurs de bus de communication (14), pour recevoir le second signal provenant du dispositif de mémoire (20) dans le premier cycle d'horloge de système;

des troisièmes moyens (15; 94, 96, 98, 100, 102, 104) connectés au processeur de données (12), aux moyens contrôleurs de bus de communication (14) et au dispositif de mémoire (20) par l'intermédiaire du bus de communication, pour transmettre pendant le premier cycle d'horloge de système la totalité de l'opérande, ou la plus grande partie possible de celui-ci, pour transmettre complètement l'opérande dans le premier cycle d'horloge de système, en dimensionnant le bus de communication de façon à accomplir la transmission conformément aux premier et second signaux; et

des quatrièmes moyens (180) incorporés dans les moyens contrôleurs de bus de communication et connectés aux troisièmes moyens, pour appliquer aux troisièmes moyens (18) un troisième signal (*CURS) indiquant la taille réelle de l'opérande qui est placé sur le bus de communication,

les troisièmes moyens transmettant dans le second cycle d'horloge de système immédiatement suivant, une partie restante éventuelle de l'opérande qui dépasse la capacité du bus de communication dans le premier cycle d'horloge de système, en dimensionnant le bus de communication de façon à prendre en charge la partie restante éventuelle.

2. Le système informatique (10) de la revendication 1, dans lequel les moyens contrôleurs de bus de communication (14) sont conçus pour permettre au processeur de données (12) d'accomplir d'autres tâches de traitement de données pendant la transmission de l'opérande.

3. Le système informatique (10) de la revendication 1 ou 2, dans lequel les premiers moyens (188, 196) fournissent le premier signal au début de chaque cycle parmi un ensemble de cycles d'horloge de système continus.

4. Procédé de transmission d'opérandes entre un processeur de données et un dispositif de mémoire, dans un système informatique (10) qui comporte un processeur de données (12) capable de communiquer avec le dispositif de mémoire (20) par l'intermédiaire d'un bus de communication, le dispositif de mémoire (20) ayant une taille d'accès quelconque parmi un ensemble de tailles d'accès différentes, ce procédé comprenant les étapes suivantes :

on applique au dispositif de mémoire (20) un signal de demande provenant du processeur de données (12), dans un premier cycle d'horloge de système, ce signal indiquant qu'un opérande doit être transmis en utilisant le bus de communication;

on reçoit un signal de réponse provenant du dispositif de mémoire (20), en réponse au signal de demande; et

on transmet l'opérande entre le processeur de données (10) et le dispositif de mémoire (20); ce procédé étant en outre caractérisé par les étapes suivantes :

on établit et on maintient continuellement la commande du bus pendant le premier cycle d'horloge de système et, si nécessaire, pendant un second cycle d'horloge de système immédiatement suivant, pour le processeur de données (12) plutôt que pour le dispositif de mémoire (20), au moyen d'un contrôleur de bus de communication (14) qui est connecté entre le processeur de données (12) et le dispositif de mémoire (20);

on transfère le signal de demande (*RQS, S1, S0) vers le contrôleur de bus de communication (14), ce signal de demande indiquant la taille ou la longueur de l'opérande à transmettre;

on reçoit avec le contrôleur de bus (14) le signal de réponse (*DSACK0, *DSACK1) provenant du dispositif de mémoire (20) dans le premier cycle d'horloge de système, ce signal de réponse indiquant la taille de l'accès de données du dispositif de mémoire (20);

on applique à un circuit d'interface (18) un signal de taille d'opérande (*CURS) provenant du

contrôleur de bus de communication et indiquant la taille réelle de l'opérande qui est placé sur le bus de communication, ce circuit d'interface étant connecté au processeur de données (12), au contrôleur de bus de communication (14), et au dispositif de mémoire (20) par l'intermédiaire du bus de communication;

on transmet par l'intermédiaire du bus de communication, sous la commande du contrôleur de bus (14), pendant le premier cycle d'horloge, la totalité de l'opérande, ou la plus grande partie possible de celui-ci, en dimensionnant le bus de communication de façon à exécuter la transmission conformément au signal de demande et au signal de réponse; et

on transmet dans le second cycle d'horloge de système immédiatement suivant une partie restante éventuelle de l'opérande, qui dépasse la capacité du bus de communication dans le premier cycle d'horloge de système, en dimensionnant le bus de communication de façon à prendre en charge la partie restante éventuelle.

5. Le procédé de transmission d'opérandes dans le système informatique (10) de la revendication 4, dans lequel le signal de demande est produit au début de chaque cycle parmi un ensemble de cycles d'horloge de système consécutifs.

Patentansprüche

1. Datenverarbeitungssystem (10) mit einem Datenprozessor (12), der über eine Nachrichten-Sammelleitung mit einem Speicher (20) Daten auszutauschen imstande ist, der über mehrere sich unterscheidende Ein-/Auslaßformate verfügt, wobei zu dem Datenprozessor (12) gehören:

Hilfsmittel zum Versorgen des Speichers (20) mit einem ersten Signal, das in einem ersten Taktzyklus des Systems anzeigt, daß unter Benutzung der Nachrichten-Sammelleitung ein Operand zu übertragen ist, und zur Übertragung des Operanden zwischen dem Datenprozessor und dem Speicher beim Empfang eines zweiten Signals aus dem Speicher, ferner dadurch gekennzeichnet,

daß eine Steuerung (14) der Nachrichten-Sammelleitung zwischen dem Datenprozessor (12) und dem Speicher (20) und an der Nachrichten-Sammelleitung angeschlossen ist, um ununterbrochen die Führung der Sammelleitung während des ersten und, wenn notwendig, eines zweiten unmittelbar folgenden Taktzyklus des Systems zu behaupten, wobei das erste Signal (*RQS, S1, S0) der Steuerung (14) der Nachrichten-Sammelleitung eine Länge des zu übertragenden Operanden anzeigt,

daß in der Steuerung (14) der Nachrichten-Sammelleitung eine erste Vorrichtung (188, 196) vorgesehen ist, um während des Übertragungsvorganges des Operanden die Führung der Sammelleitung für den Datenprozessor (12) über den Speicher (20) aufzubauen und ununterbrochen beizubehalten,

daß der Speicher (20) an die Steuerung (14) der Nachrichten-Sammelleitung das zweite Signal (*DSACK0, *DSACK1) liefert, das das Datenein-/Auslaßformat des Speichers (20) anzeigt,

daß in der Steuerung (14) der Nachrichten-Sammelleitung eine zweite Vorrichtung (188, 190) vorgesehen ist, um in dem ersten Taktzyklus des Systems aus dem Speicher (20) das zweite Signal zu empfangen,

daß eine dritte Vorrichtung (13; 94, 96, 98, 100, 102, 104) an dem Datenprozessor (12), an der Steuerung (14) der Nachrichten-Sammelleitung und über die Nachrichten-Sammelleitung an dem Speicher (20) angeschlossen ist, um während des ersten Taktzyklus des Systems den gesamten Operanden oder einen möglichst großen Abschnitt des Operanden zu übertragen, so daß zur Ausführung der Übertragung entsprechend den ersten und zweiten Signalen der Operand bei dem ersten Taktzyklus des Systems durch Ordnen der Nachrichten-Sammelleitung vollständig übertragen wird,

daß in der Steuerung der Nachrichten-Sammelleitung eine vierte Vorrichtung (180) an die dritte Vorrichtung zur Lieferung eines dritten Signals (*CURS) an die dritte Vorrichtung (18) angeschlossen ist, das die tatsächliche Größe des auf die Nachrichten-Sammelleitung gelegten Operanden anzeigt,

und daß die dritte Vorrichtung, wenn überhaupt, den Rest des Operanden, der die Kapazität der Nachrichten-Sammelleitung bei dem ersten Taktzyklus des Systems übersteigt, bei dem zweiten unmittelbar folgenden Taktzyklus des Systems in der Weise überträgt, daß zur Anpassung an den Rest die Nachrichten-Sammelleitung der Größe nach geordnet wird.

2. Datenverarbeitungssystem (10) gemäß Anspruch 1, bei dem die Steuerung (14) der Nachrichten-Sammelleitung derart angeordnet ist, daß der Datenprozessor (12) während der Übertragung des Operanden andere Aufgaben der Datenverarbeitung auszuführen imstande ist.

3. Datenverarbeitungssystem (10) gemäß Anspruch 1 oder 2, bei dem die erste Vorrichtung (188, 196) das erste Signal zu Beginn jedes der zusammenhängenden Taktzyklen des Systems vorsieht.

4. Verfahren zum Übertragen von Operanden zwischen einem Datenprozessor und einem Speicher in einem Datenverarbeitungssystem (10), das einen Datenprozessor (12) besitzt, der über eine Nachrichten-Sammelleitung mit dem Speicher (20) Nachrichten austauschen kann, der mehrere sich unterscheidende Ein-/Auslaßformate aufweist, mit den folgenden Schritten:

bei einem ersten Taktzyklus des Systems aus dem Datenprozessor (12) ein Anforderungssignal an den Speicher (20) heranzubringen, das anzeigt, daß ein Operand unter Benutzung der Nachrichten-Sammelleitung zu übertragen ist,

in Abhängigkeit von dem Anforderungssignal aus dem Speicher (20) ein Antwortsignal zu empfangen und den

Operanden zwischen dem Datenprozessor (10) und dem Speicher (20) zu übertragen, ferner gekennzeichnet durch die folgenden Schritte:

während des ersten Taktzyklus und, wenn notwendig, während eines zweiten unmittelbar folgenden Taktzyklus des Systems für den Datenprozessor die Führung der Sammelleitung über den Speicher (20) mit einer Steuerung (14) der Sammelleitung aufzubauen und ununterbrochen aufrechtzuerhalten, die zwischen dem Datenprozessor (12) und dem Speicher (20) angeschlossen ist,

zu der Steuerung (14) der Nachrichten-Sammelleitung das Anforderungssignal (*RQS, S1, S0) zu übertragen, das die Länge des zu übertragenden Operanden anzeigt,

bei dem ersten Taktzyklus des Systems aus dem Speicher (20) das Antwortsignal (*DSACK0, *DSACK1) mit der Steuerung (14) der Sammelleitung zu empfangen, das das Datenein-/Auslaßformat des Speichers (20) anzeigt,

aus der Steuerung der Nachrichten-Sammelleitung ein die Operandengröße angebendes Signal (*CURS) an eine Schnittstellenschaltung (18) zu liefern, das die tatsächliche Größe des auf die Nachrichten-Sammelleitung zu legenden Operanden anzeigt, wobei die Schnittstellenschaltung an den Datenprozessor (12), die Steuerung (14) der Nachrichten-Sammelleitung und über die Nachrichten-Sammelleitung an den Speicher (20) angeschlossen wird,

während des ersten Taktzyklus des Systems den gesamten Operanden oder einen möglichst großen Abschnitt des Operanden über die von der Steuerung (14) gesteuerte Nachrichten-Sammelleitung zu übertragen, um durch eine Ordnung der Nachrichten-Sammelleitung die Übertragung entsprechend dem Anforderungssignal und dem Antwortsignal auszuführen, und

einen möglichen Rest des Operanden,der beim ersten Taktzyklus des Systems nicht in die Nachrichten-Sammelleitung hineinpaßt, bei dem zweiten unmittelbar nachfolgenden Taktzyklus durch eine Ordnung der Nachrichten-Sammelleitung zur Anpassung an den möglichen Rest zu übertragen.

5. Verfahren zum Übertragen von Operanden in dem Datenverarbeitungssystem (10) des Anspruches 4, bei dem das Anforderungssignal zu Beginn jedes der aufeinanderfolgenden Taktzyklen des Systems vorgesehen wird.

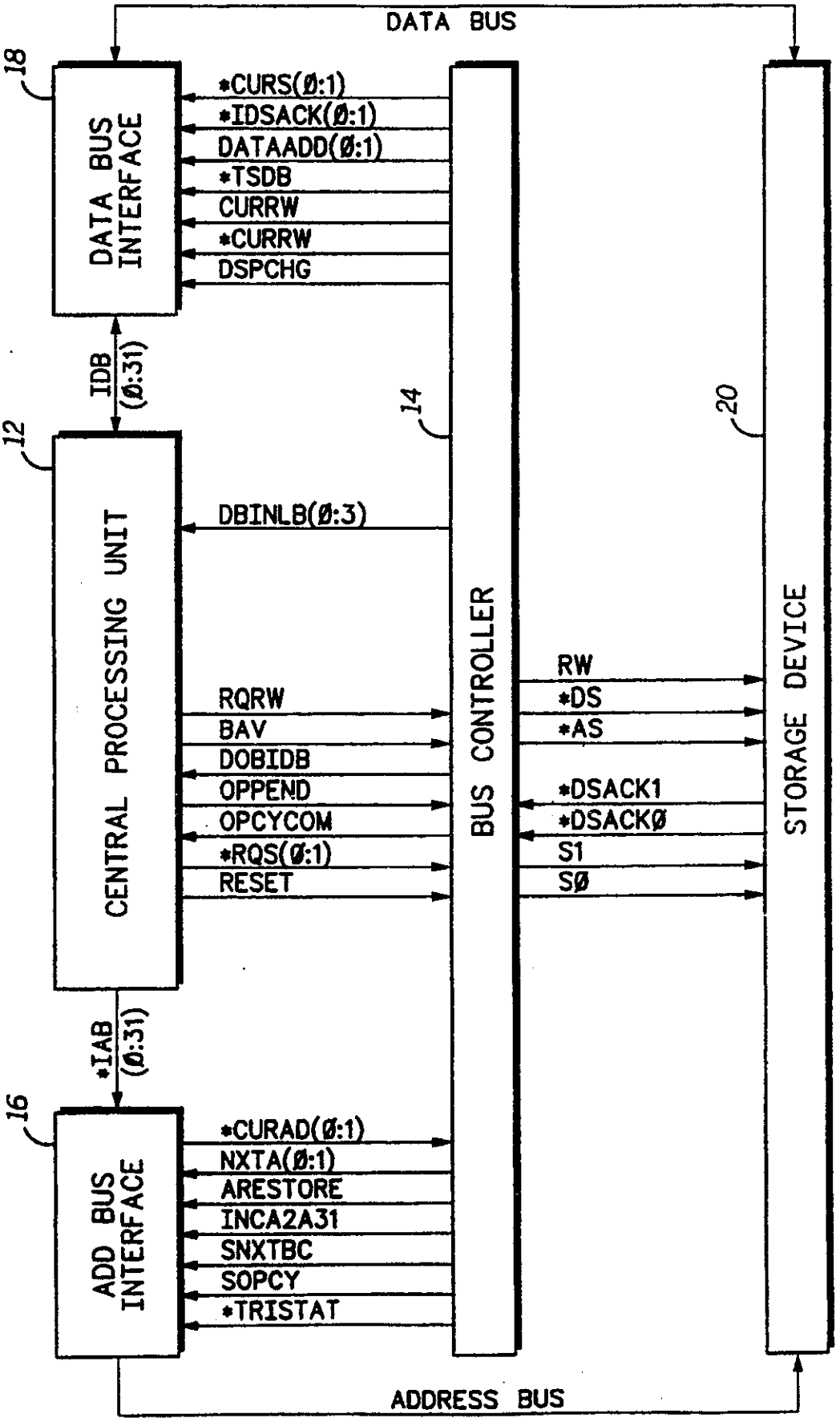
DATA BUS
18
DATA BUS INTERFACE
*CURS(Ø:1)
*IDSACK(Ø:1)
DATAADD(Ø:1)
*TSDB
CURRW
*CURRW
DSPCHG
IDB (Ø:31)
12
CENTRAL PROCESSING UNIT
14
BUS CONTROLLER
20
STORAGE DEVICE
DBINLB(Ø:3)
RQRW
BAV
DOBIDB
OPPEND
OPCYCOM
*RQS(Ø:1)
RESET
RW
*DS
*AS
*DSACK1
*DSACKØ
S1
SØ
10
*IAB (Ø:31)
16
ADD BUS INTERFACE
*CURAD(Ø:1)
NXTA(Ø:1)
ARESTORE
INCA2A31
SNXTBC
SOPCY
*TRISTAT
ADDRESS BUS

FIG.1

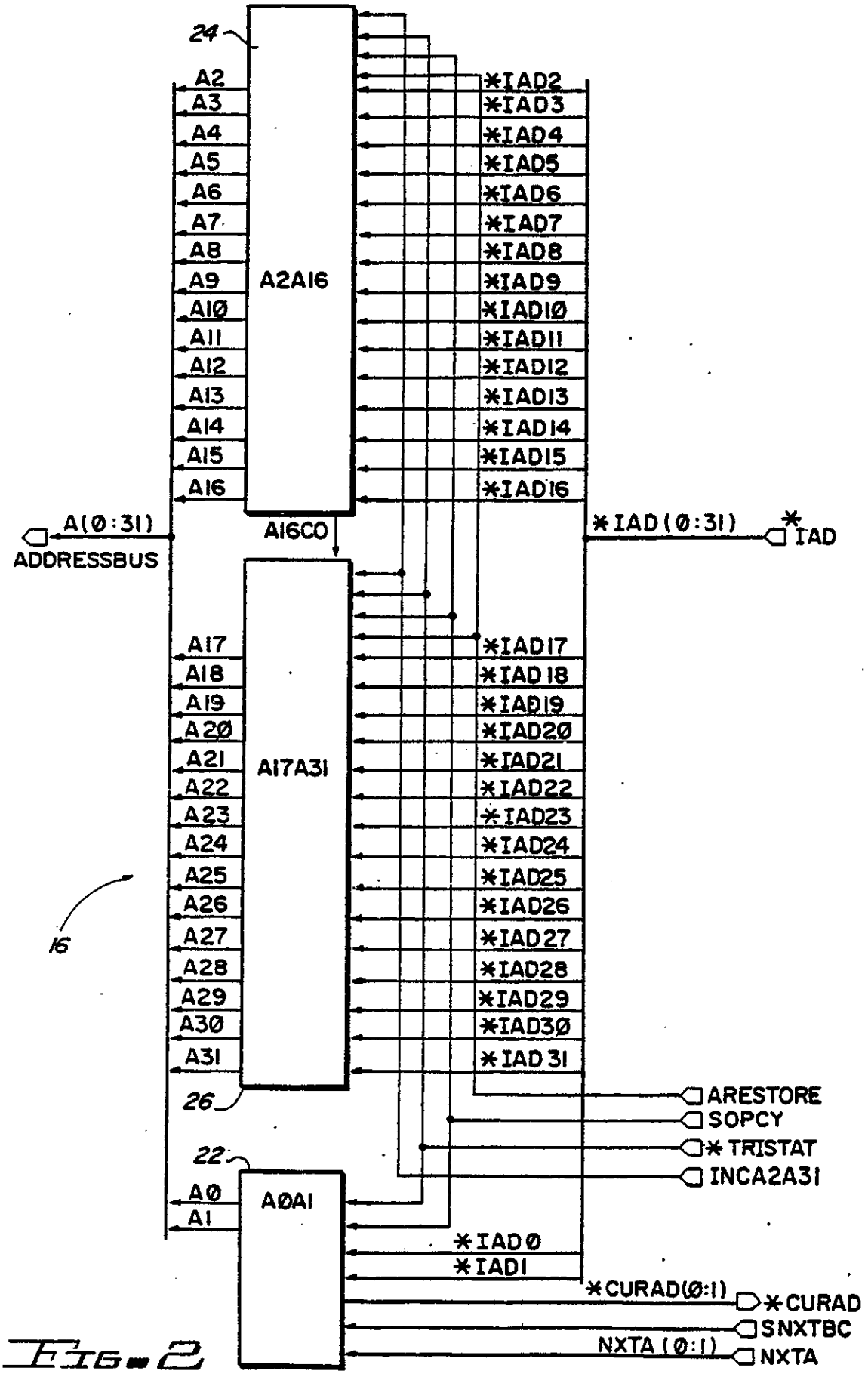
24
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A2A16
A16CO
*IAD2
*IAD3
*IAD4
*IAD5
*IAD6
*IAD7
*IAD8
*IAD9
*IADIØ
*IADII
*IADI2
*IADI3
*IADI4
*IADI5
*IADI6
A(0:31)
ADDRESSBUS
*IAD(0:31)
*IAD
A17
A18
A19
A20
A21
A22
A23
A24
A25
A26
A27
A28
A29
A30
A31
A17A31
*IAD17
*IAD18
*IAD19
*IAD20
*IAD21
*IAD22
*IAD23
*IAD24
*IAD25
*IAD26
*IAD27
*IAD28
*IAD29
*IAD30
*IAD31
16
26
22
A0
A1
A0A1
ARESTORE
SOPCY
*TRISTAT
INCA2A31
*IAD0
*IAD1
*CURAD(0:1)
*CURAD
SNXTBC
NXTA(0:1)
NXTA

Fig-2

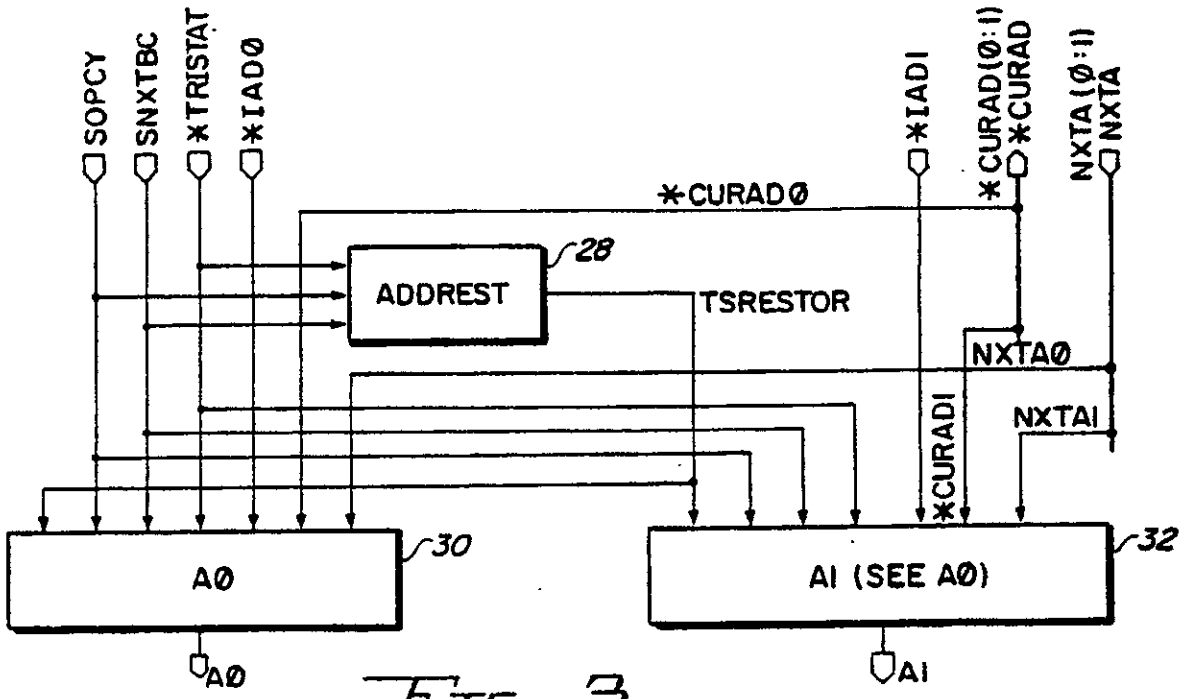

Fig-3

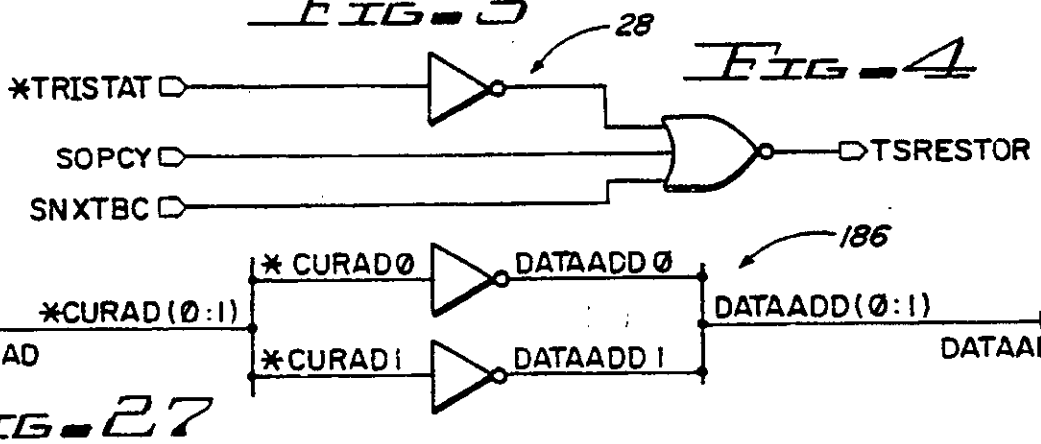

Fig-4

Fig-27

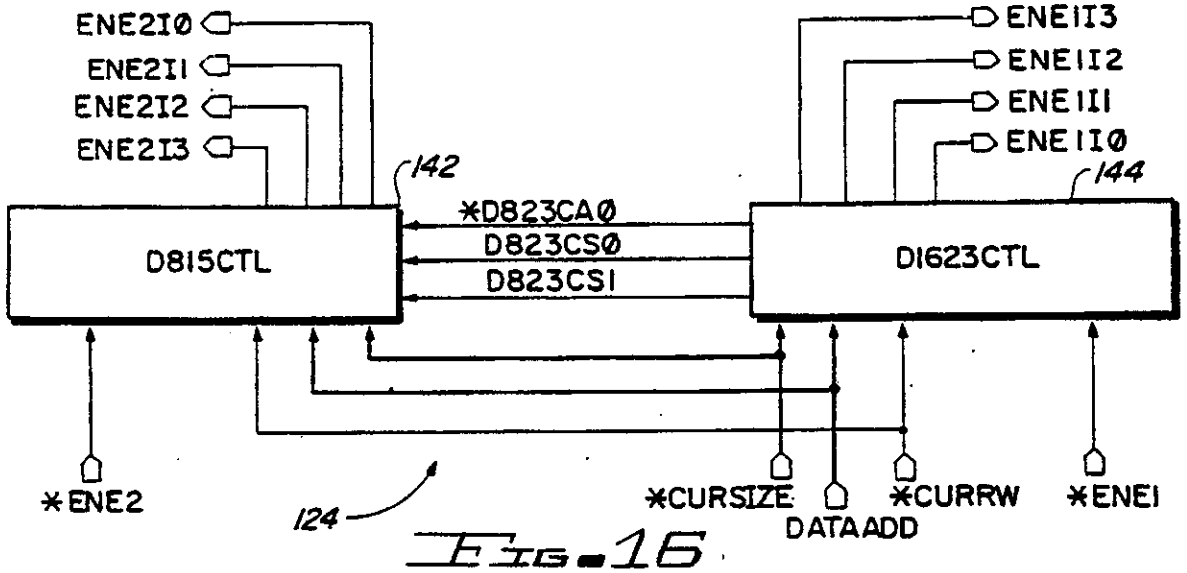

Fig-16

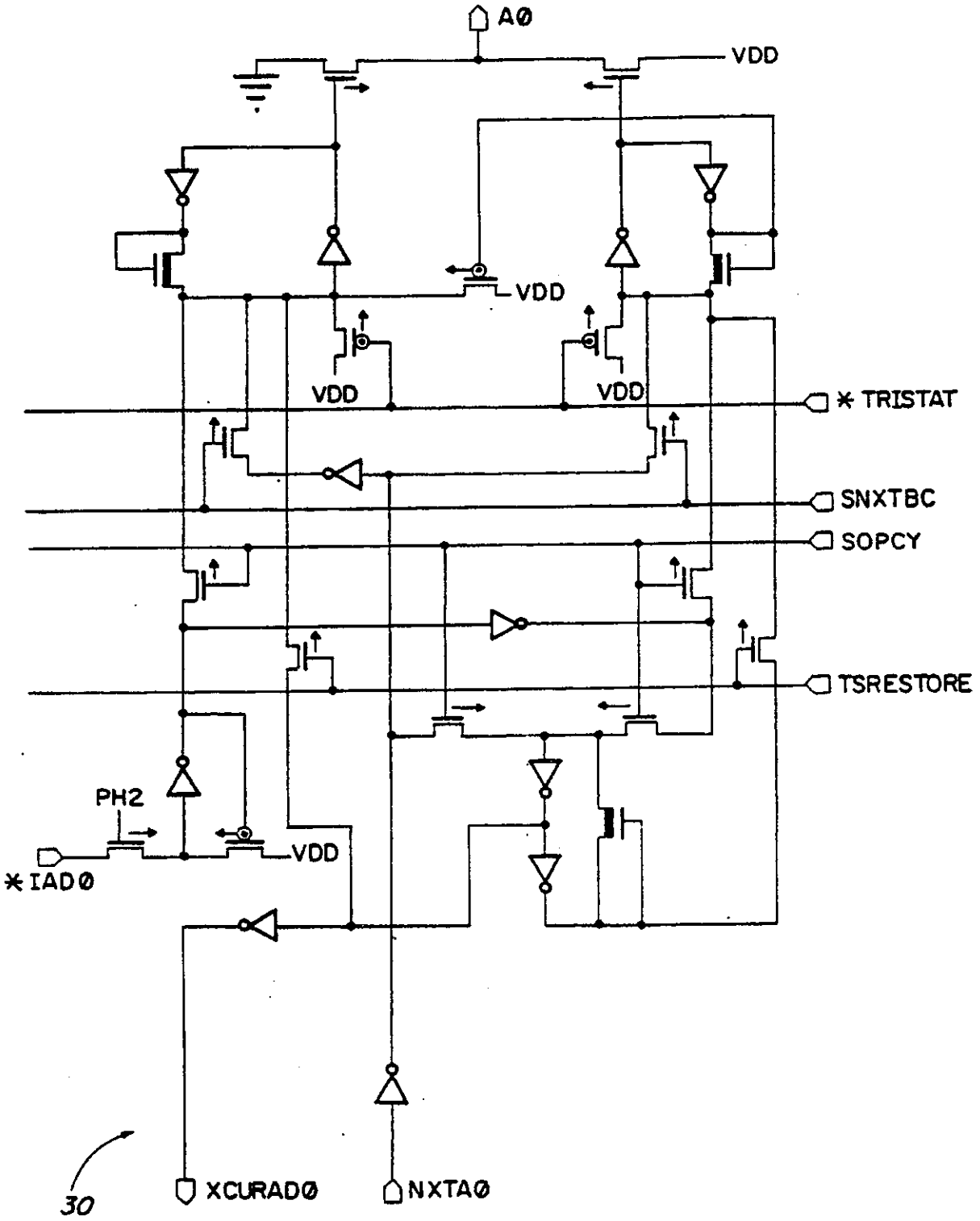
A0
VDD
VDD
VDD
VDD
* TRISTAT
SNXTBC
SOPCY
TSRESTORE
PH2
VDD
*IAD0
30
XCURAD0
NXTA0

Fig. 5

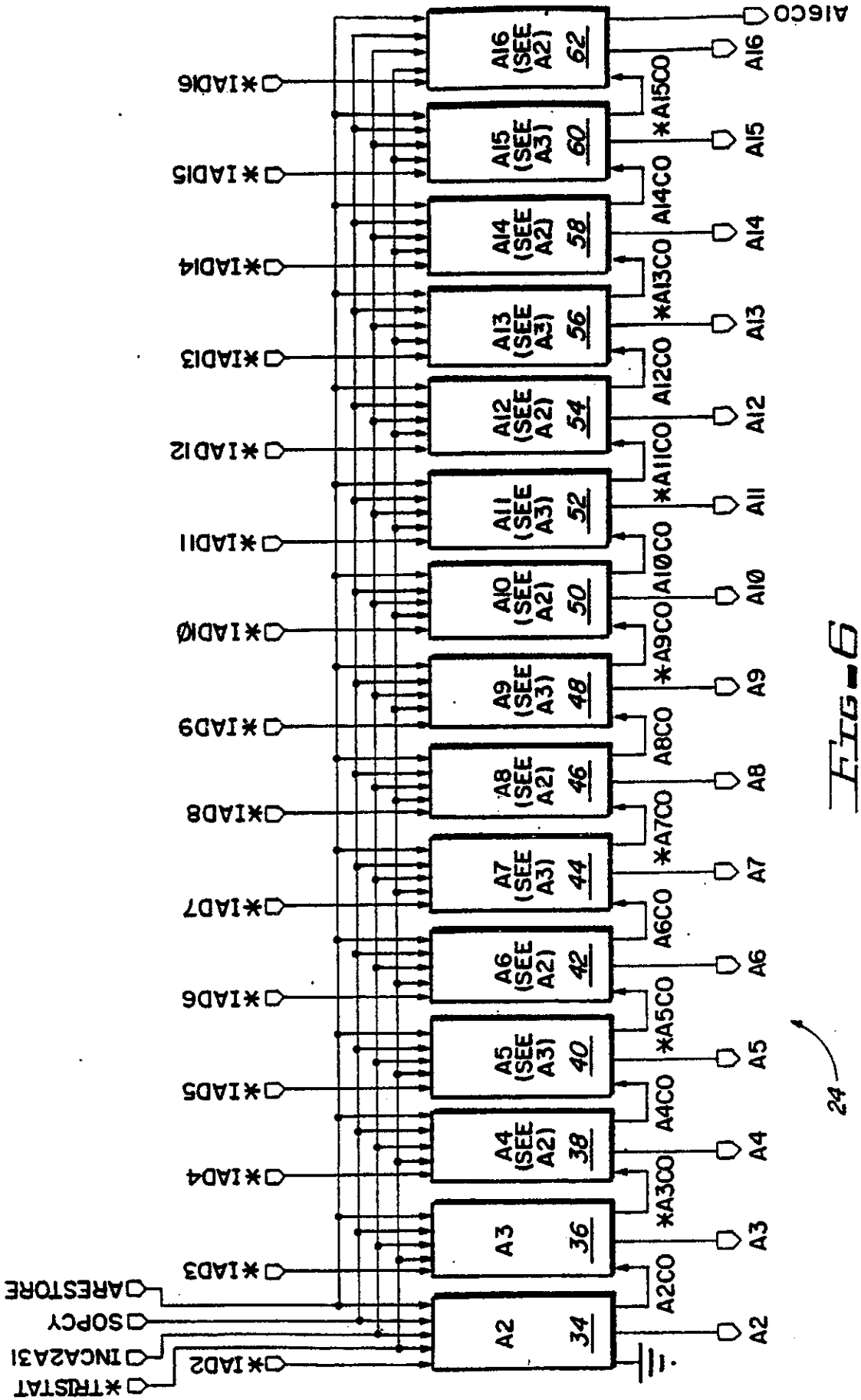
A16 (SEE A2) 62
A15 (SEE A3) 60
A14 (SEE A2) 58
A13 (SEE A3) 56
A12 (SEE A2) 54
A11 (SEE A3) 52
A10 (SEE A2) 50
A9 (SEE A3) 48
A8 (SEE A2) 46
A7 (SEE A3) 44
A6 (SEE A2) 42
A5 (SEE A3) 40
A4 (SEE A2) 38
A3 36
A2 34
A16CO
A16
*A15CO
A15
A14CO
A14
*A13CO
A13
A12CO
A12
*A11CO
A11
A10CO
A10
*A9CO
A9
A8CO
A8
*A7CO
A7
A6CO
A6
*A5CO
A5
A4CO
A4
*A3CO
A3
A2CO
A2
*IAD16
*IAD15
*IAD14
*IAD13
*IAD12
*IAD11
*IAD10
*IAD9
*IAD8
*IAD7
*IAD6
*IAD5
*IAD4
*IAD3
*IAD2
ARESTORE
SOPCY
INCA2A31
*TRISTAT
24

FIG-6

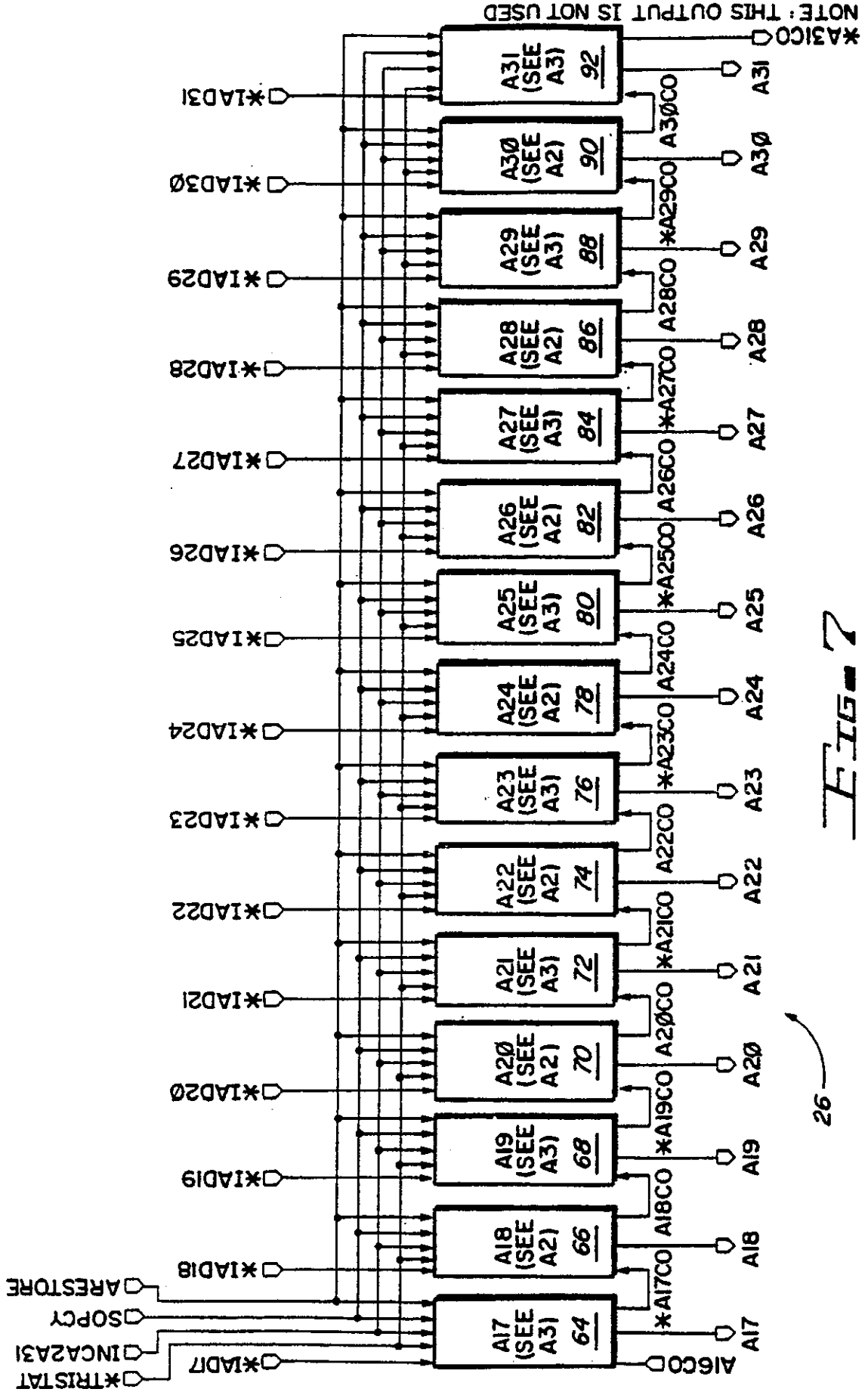
NOTE: THIS OUTPUT IS NOT USED
*A31CO
A31 (SEE A3) 92
A30 (SEE A2) 90
A29 (SEE A3) 88
A28 (SEE A2) 86
A27 (SEE A3) 84
A26 (SEE A2) 82
A25 (SEE A3) 80
A24 (SEE A2) 78
A23 (SEE A3) 76
A22 (SEE A2) 74
A21 (SEE A3) 72
A20 (SEE A2) 70
A19 (SEE A3) 68
A18 (SEE A2) 66
A17 (SEE A3) 64
A16CO
A31
A30
A29
A28
A27
A26
A25
A24
A23
A22
A21
A20
A19
A18
A17
A30CO
*A29CO
A28CO
*A27CO
A26CO
*A25CO
A24CO
*A23CO
A22CO
*A21CO
A20CO
*A19CO
A18CO
*A17CO
*IAD31
*IAD30
*IAD29
*IAD28
*IAD27
*IAD26
*IAD25
*IAD24
*IAD23
*IAD22
*IAD21
*IAD20
*IAD19
*IAD18
*IAD17
ARESTORE
SOPCY
INCA2A31
*TRISTAT
26

Fig. 7

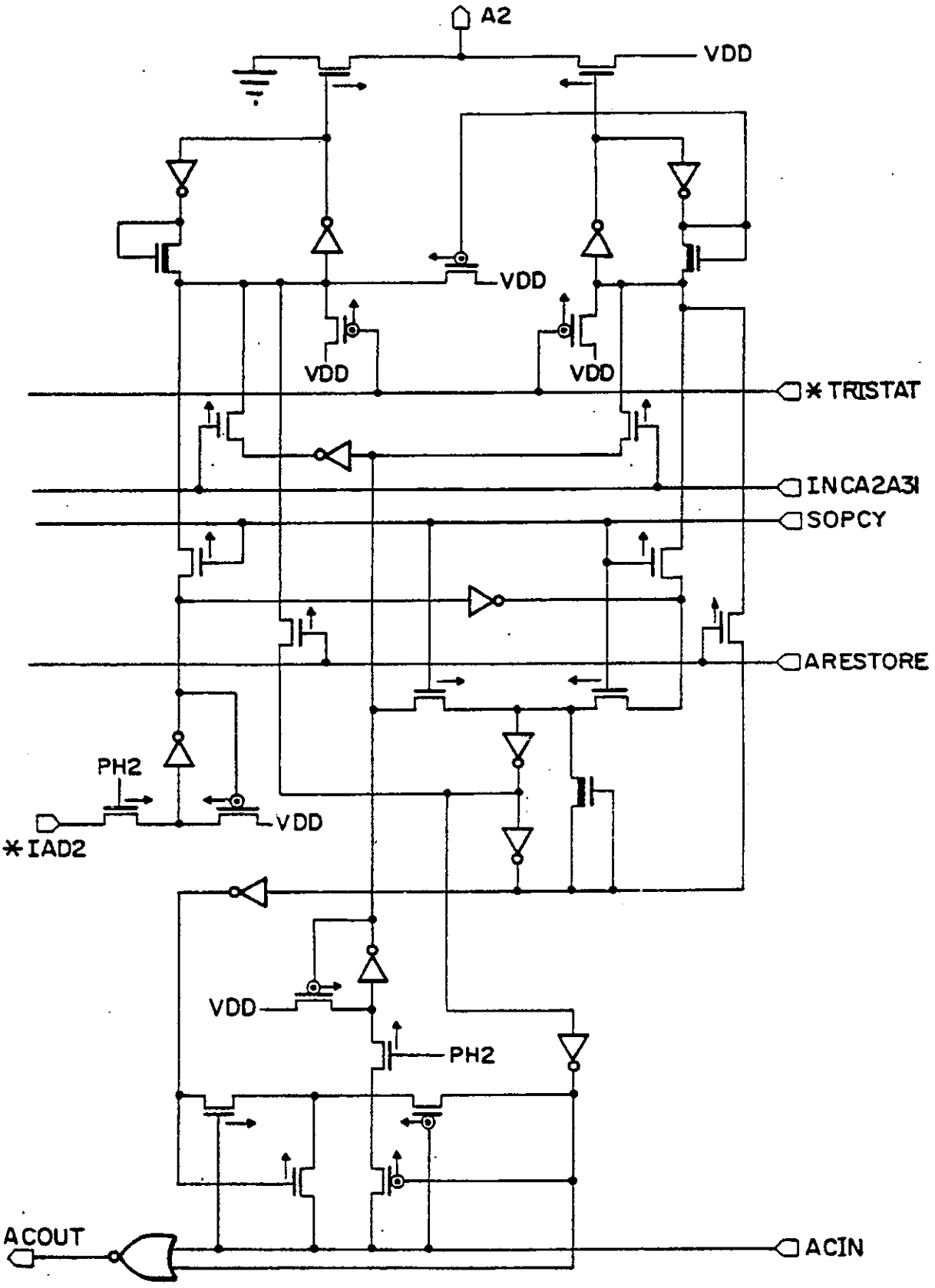
A2
VDD
VDD
VDD
VDD
*TRISTAT
INCA2A3I
SOPCY
ARESTORE
PH2
VDD
*IAD2
VDD
PH2
ACOUT
ACIN
34

Fig. 8

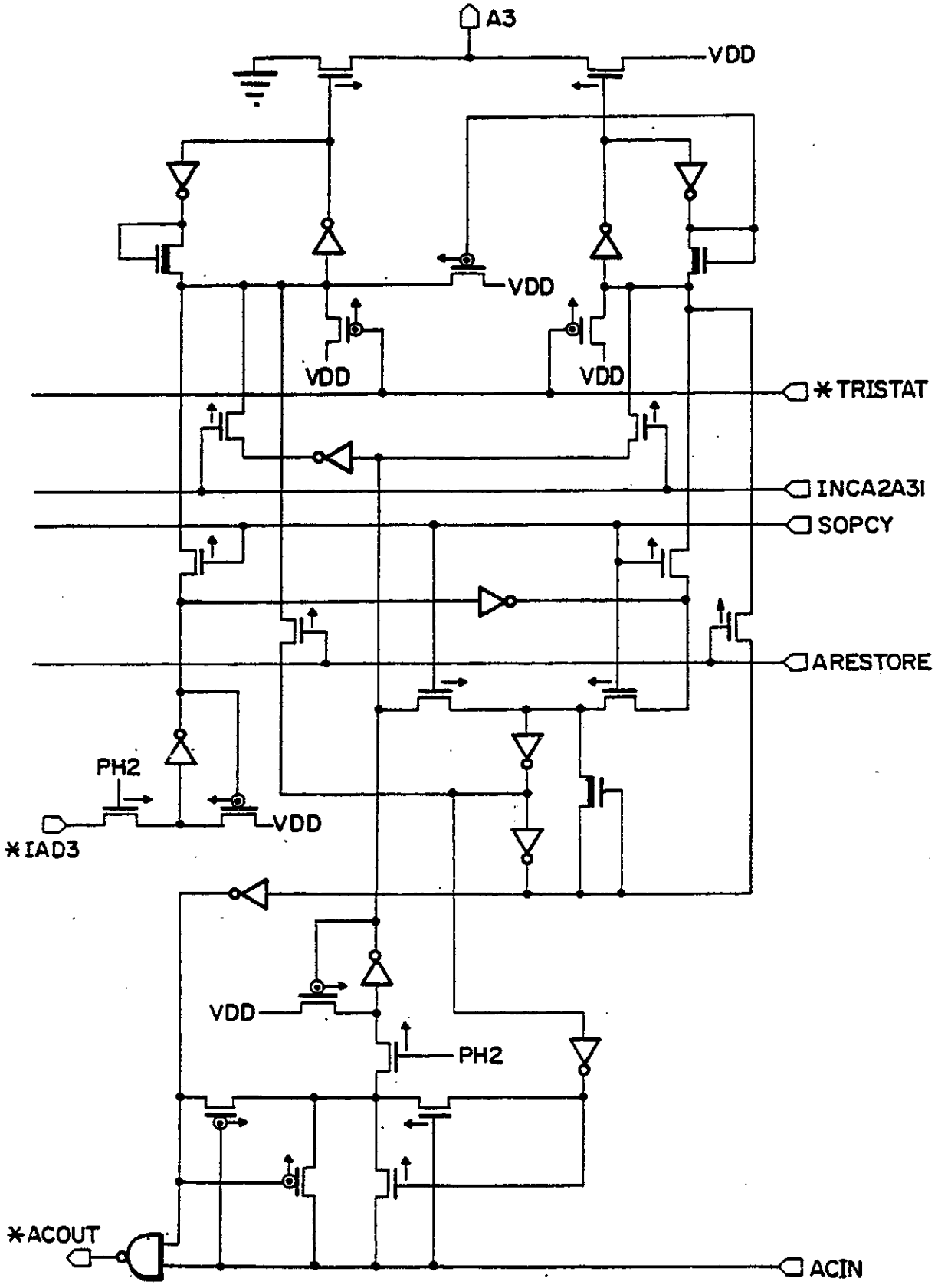
A3
VDD
VDD
VDD
VDD
*TRISTAT
INCA2A3I
SOPCY
ARESTORE
PH2
VDD
*IAD3
VDD
PH2
*ACOUT
ACIN
36

Fig. 9

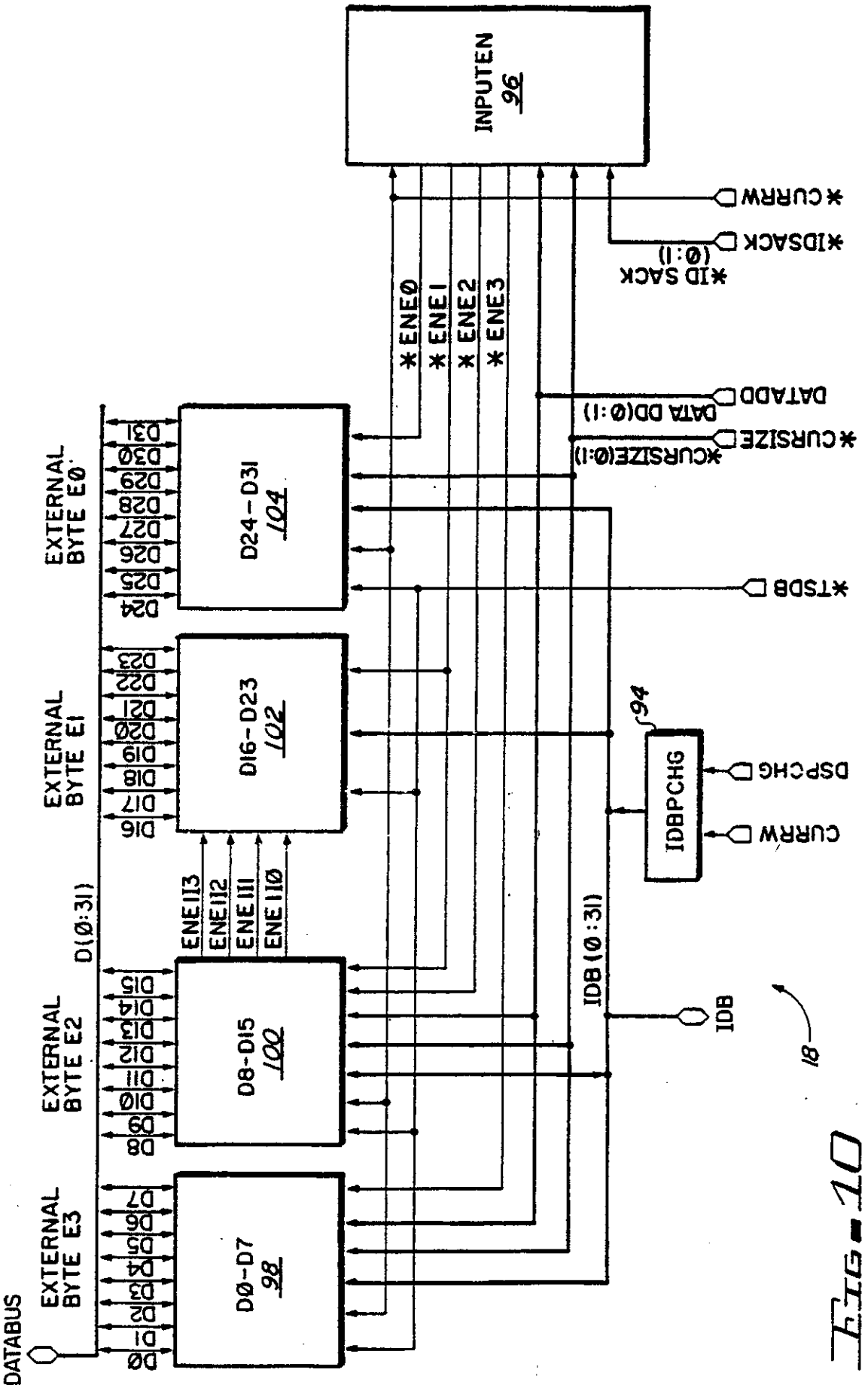
INPUTEN
96
*CURRW
*IDSACK
(0:1)
*ID SACK
*ENE0
*ENE1
*ENE2
*ENE3
DATADD
DATA DD(0:1)
*CURSIZE
*CURSIZE(0:1)
*TSDB
94
IDBPCHG
DSPCHG
CURRW
IDB(0:31)
IDB
18
D24-D31
104
D16-D23
102
D8-D15
100
D0-D7
98
ENE113
ENE112
ENE111
ENE110
D(0:31)
EXTERNAL BYTE E0
EXTERNAL BYTE E1
EXTERNAL BYTE E2
EXTERNAL BYTE E3
DATABUS

Fig-10

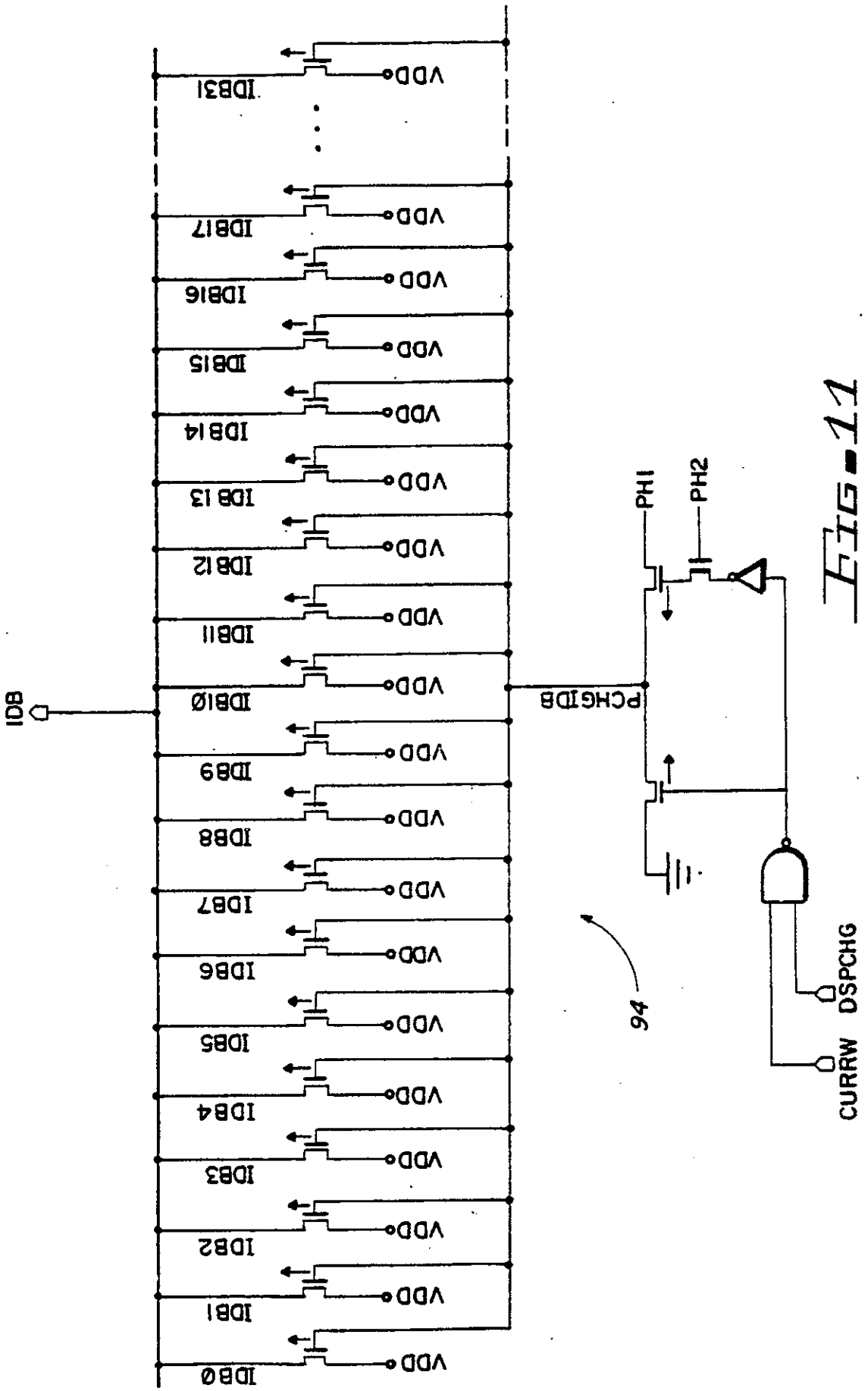
VDD
IDB31
IDB17
IDB16
IDB15
IDB14
IDB13
IDB12
IDB11
IDB10
IDB9
IDB8
IDB7
IDB6
IDB5
IDB4
IDB3
IDB2
IDB1
IDB0
IDB
PCHGIDB
PH1
PH2
94
DSPCHG
CURRW

Fig-11

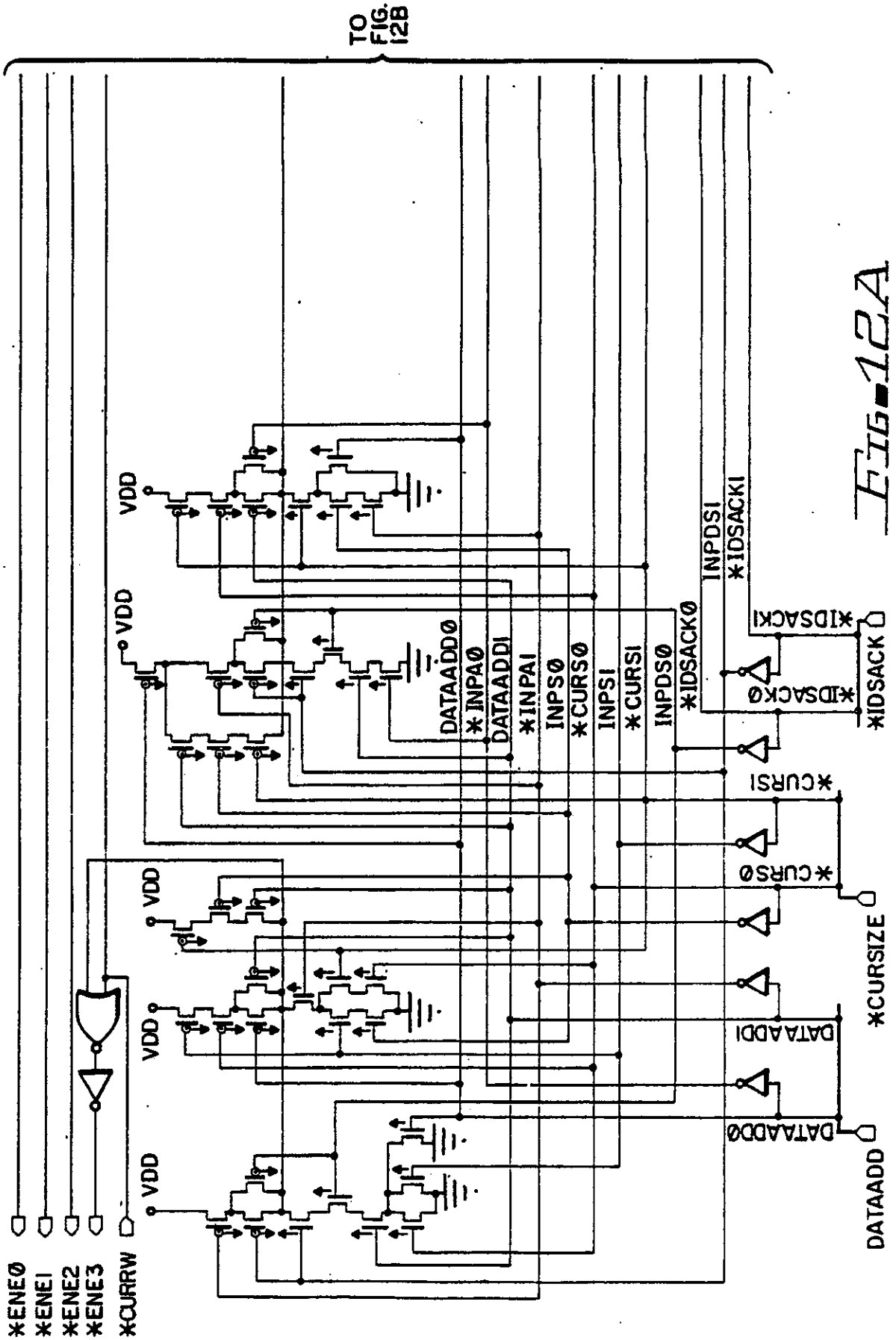
TO FIG. 12B
VDD
VDD
VDD
VDD
VDD
*ENE0
*ENE1
*ENE2
*ENE3
*CURRW
DATAADD0
*INPA0
DATAADDI
*INPAI
INPS0
*CURS0
INPSI
*CURSI
INPDS0
*IDSACK0
INPDSI
*IDSACKI
*IDSACKI
*IDSACK0
*IDSACK
*CURSI
*CURS0
DATAADDI
DATAADD0
*CURSIZE
DATAADD

Fig. 12A

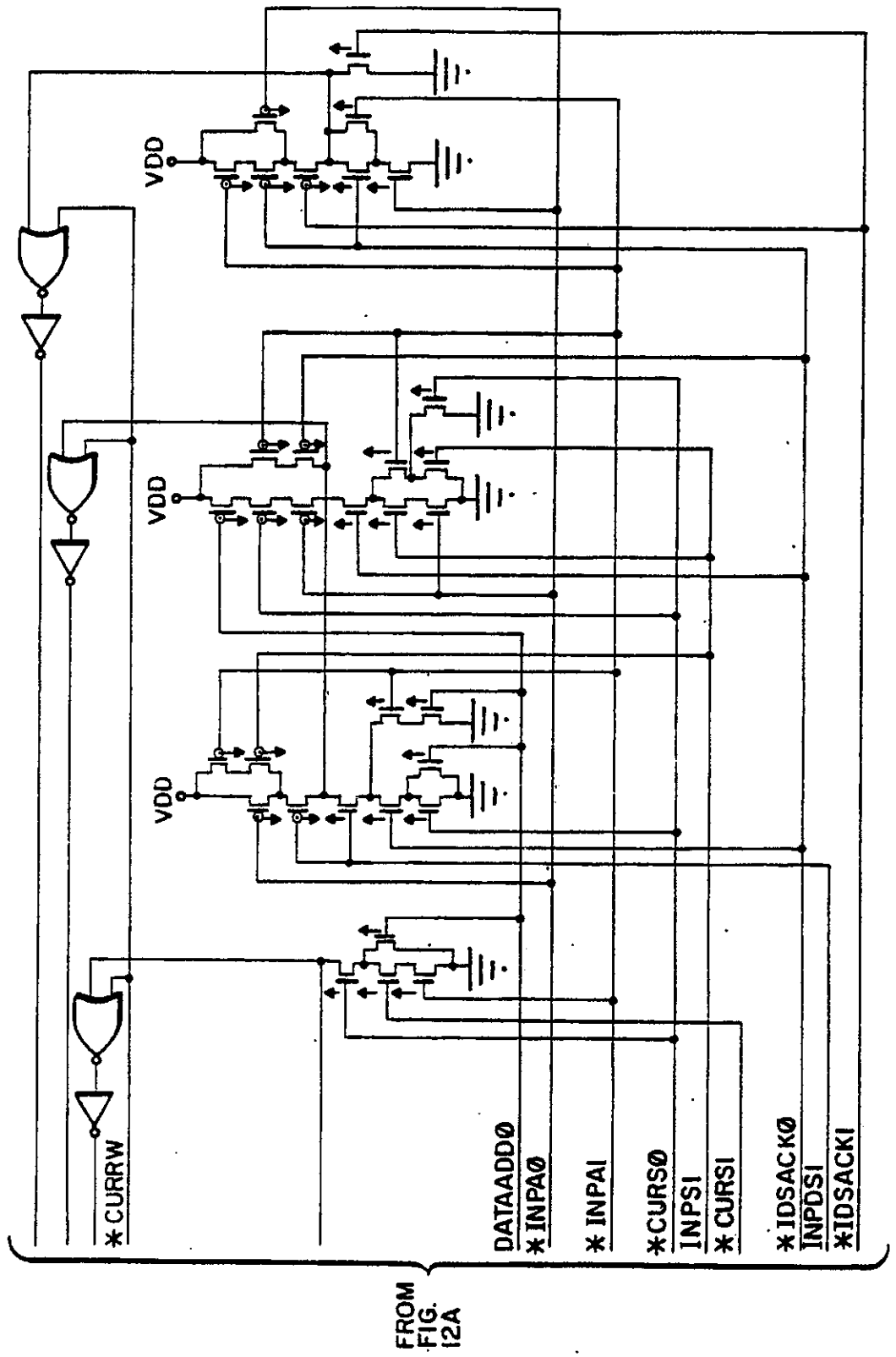
VDD
VDD
VDD
*CURRW
DATAADDØ
*INPAØ
*INPAI
*CURSØ
INPSI
*CURSI
*IDSACKØ
INPDSI
*IDSACKI
FROM FIG. 12A
96

Fig-12B

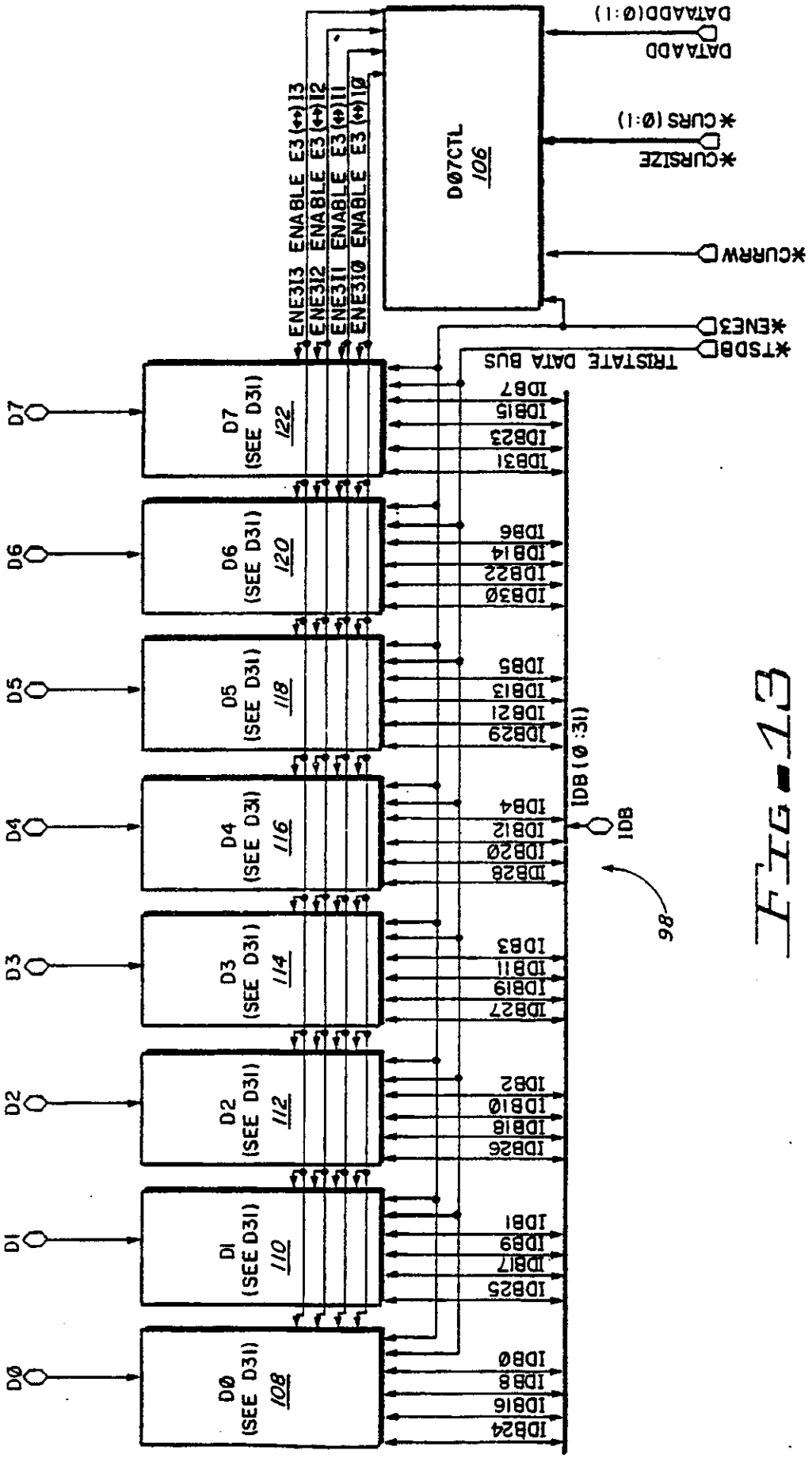
D0
D1
D2
D3
D4
D5
D6
D7
D0 (SEE D31) 108
D1 (SEE D31) 110
D2 (SEE D31) 112
D3 (SEE D31) 114
D4 (SEE D31) 116
D5 (SEE D31) 118
D6 (SEE D31) 120
D7 (SEE D31) 122
ENE313 ENABLE E3 (↔) 13
ENE312 ENABLE E3 (↔) 12
ENE311 ENABLE E3 (↔) 11
ENE310 ENABLE E3 (↔) 10
D07CTL 106
DATAADD(0:1)
DATAADD
*CURS(0:1)
*CURSIZE
*CURRW
*ENE3
*TSDB
TRISTATE DATA BUS
IDB(0:31)
IDB
98

Fig. 13

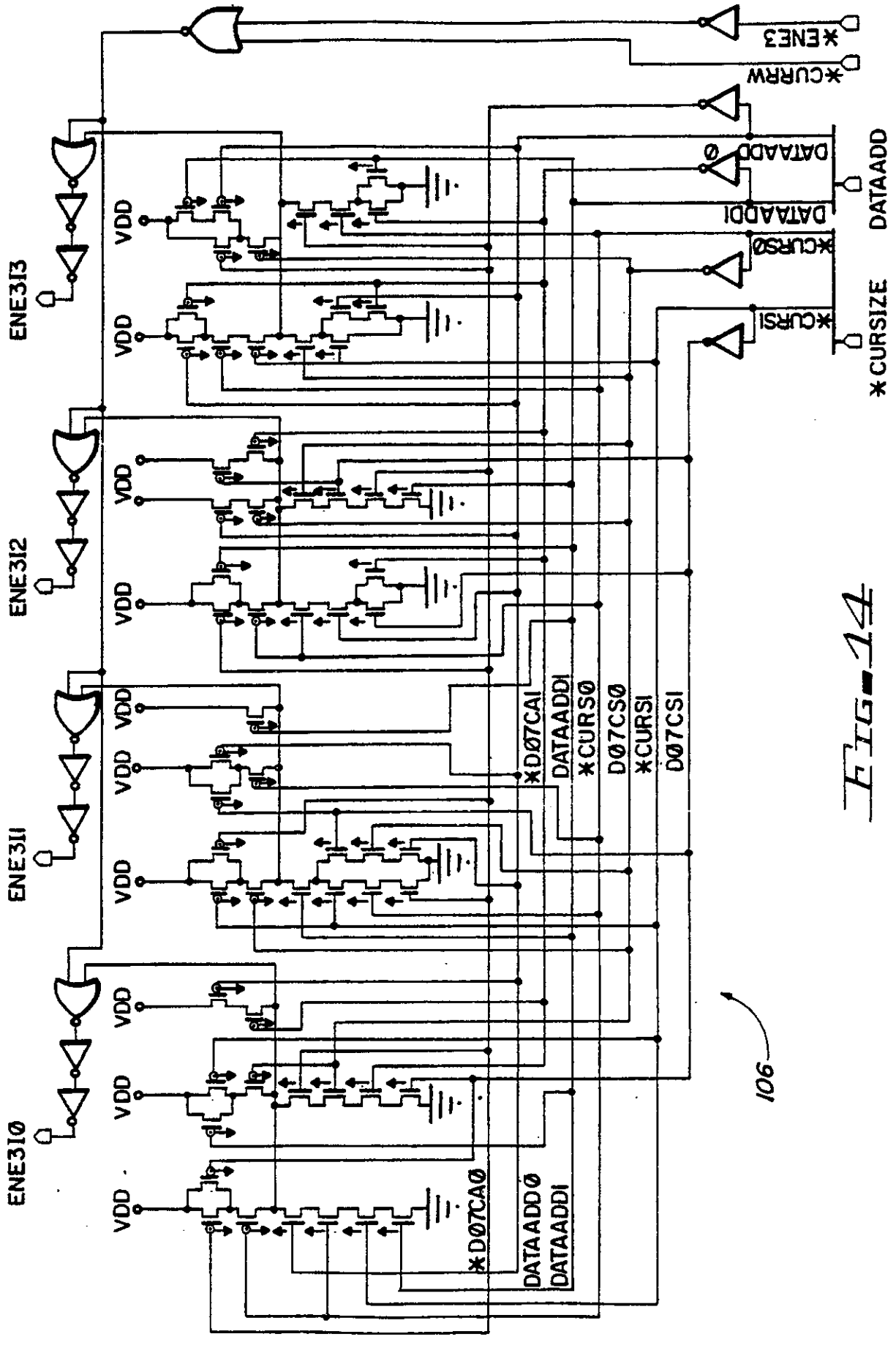
*ENE3
*CURRW
DATAADD
DATAADDI
*CURS0
*CURSI
DATAADD
*CURSIZE
ENE313
ENE312
ENE311
ENE310
VDD
*D07CAI
DATAADDI
*CURS0
D07CS0
*CURSI
D07CSI
*D07CA0
DATAADD0
DATAADDI
106

Fig-14

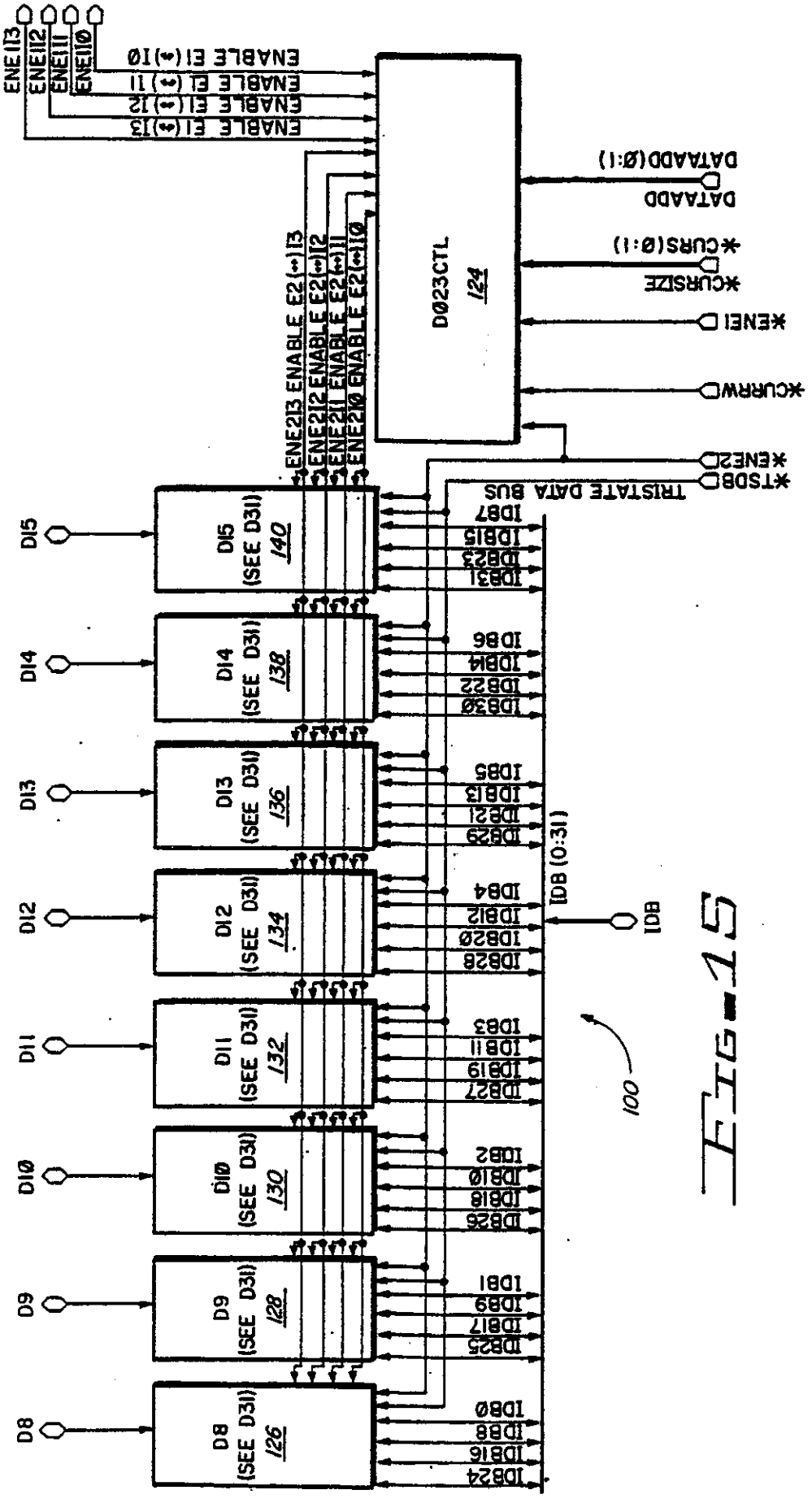
ENE I13
ENE I12
ENE I11
ENE I10
ENABLE E1(↔)I0
ENABLE E1(↔)I1
ENABLE E1(↔)I2
ENABLE E1(↔)I3
ENE2I3 ENABLE E2(↔)I3
ENE2I2 ENABLE E2(↔)I2
ENE2I1 ENABLE E2(↔)I1
ENE2I0 ENABLE E2(↔)I0
D023CTL
124
DATAADD(0:1)
DATAADD
*CURS(0:1)
*CURSIZE
*ENEI
*CURRW
*ENE2
*TSDB
TRISTATE DATA BUS
D15
(SEE D31)
140
D14
(SEE D31)
138
D13
(SEE D31)
136
D12
(SEE D31)
134
D11
(SEE D31)
132
D10
(SEE D31)
130
D9
(SEE D31)
128
D8
(SEE D31)
126
IDB (0:31)
IDB
100

FIG-15

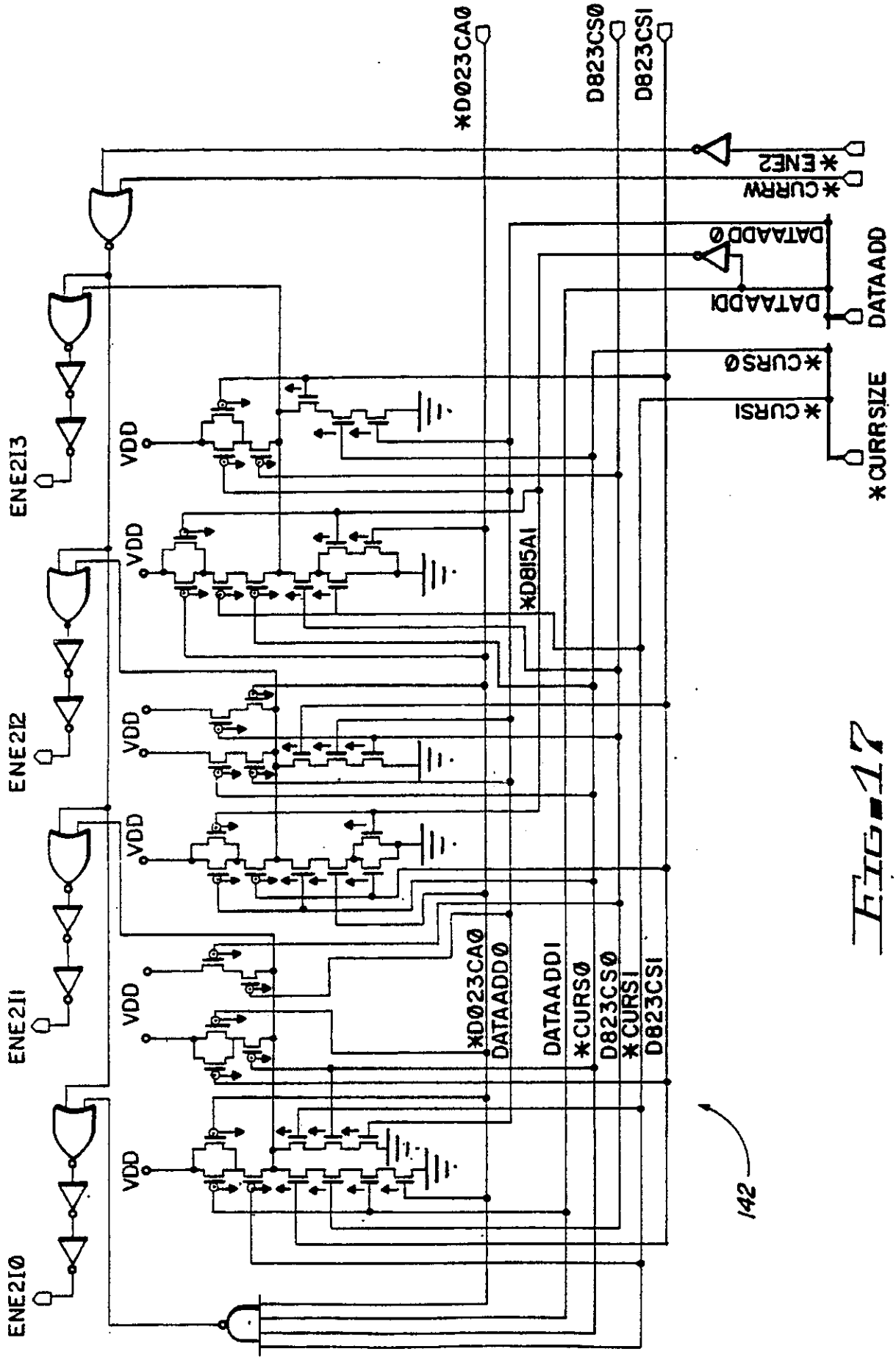
*D023CA0
D823CS0
D823CSI
*ENE2
*CURRW
DATAADD0
DATAADDI
DATAADD
*CURS0
*CURSI
*CURRSIZE
ENE213
ENE212
ENE211
ENE210
VDD
*D815AI
*D023CA0
DATAADD0
DATAADDI
*CURS0
D823CS0
*CURSI
D823CSI
142

FIG-17

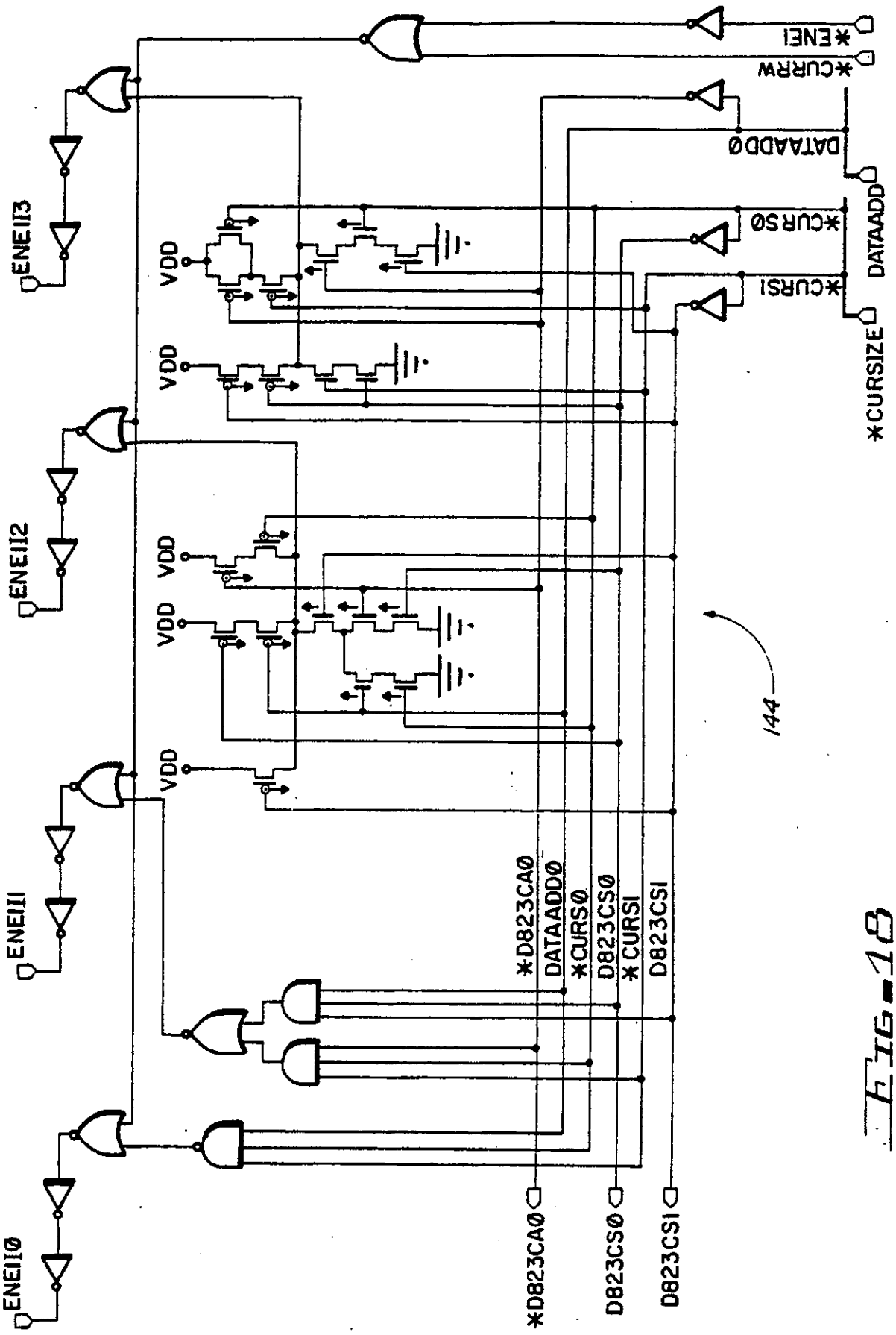
*ENEI
*CURRW
DATAADD0
DATAADD
*CURS0
*CURSI
*CURSIZE
ENEII3
ENEII2
ENEII1
ENEII0
VDD
144
*D823CA0
DATAADD0
*CURS0
D823CS0
*CURSI
D823CSI

FIG-18

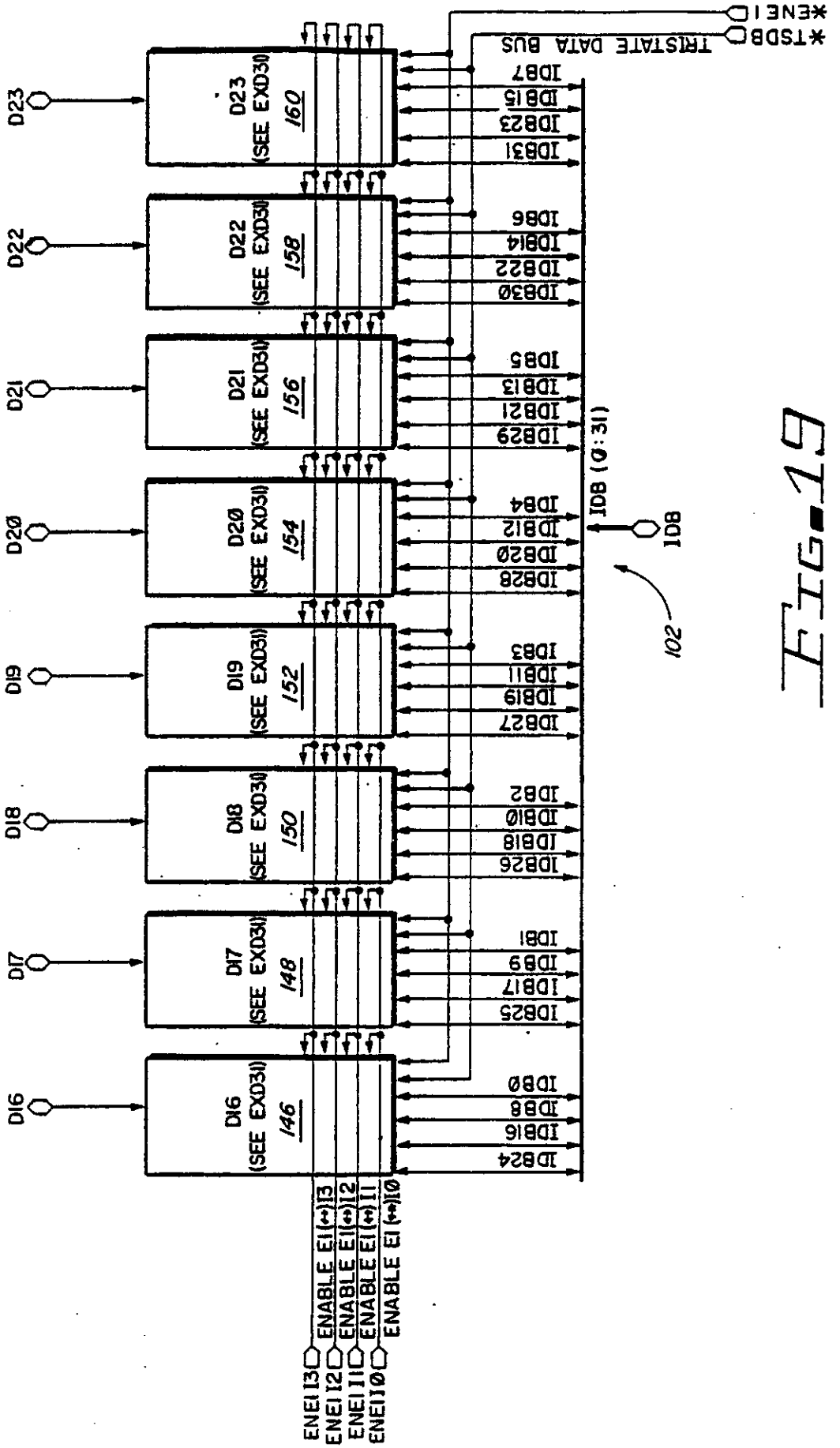
*ENE I
*TSDB
TRISTATE DATA BUS
D23
D22
D21
D20
D19
D18
D17
D16
D23 (SEE EXD31) 160
D22 (SEE EXD31) 158
D21 (SEE EXD31) 156
D20 (SEE EXD31) 154
D19 (SEE EXD31) 152
D18 (SEE EXD31) 150
D17 (SEE EXD31) 148
D16 (SEE EXD31) 146
IDB31
IDB23
IDB15
IDB7
IDB30
IDB22
IDB14
IDB6
IDB29
IDB21
IDB13
IDB5
IDB28
IDB20
IDB12
IDB4
IDB27
IDB19
IDB11
IDB3
IDB26
IDB18
IDB10
IDB2
IDB25
IDB17
IDB9
IDB1
IDB24
IDB16
IDB8
IDB0
IDB (0:31)
IDB
102
ENABLE EI (↔)13
ENABLE EI (↔)12
ENABLE EI (↔)11
ENABLE EI (↔)10
ENEI I3
ENEI I2
ENEI I1
ENEI I0

Fig. 19

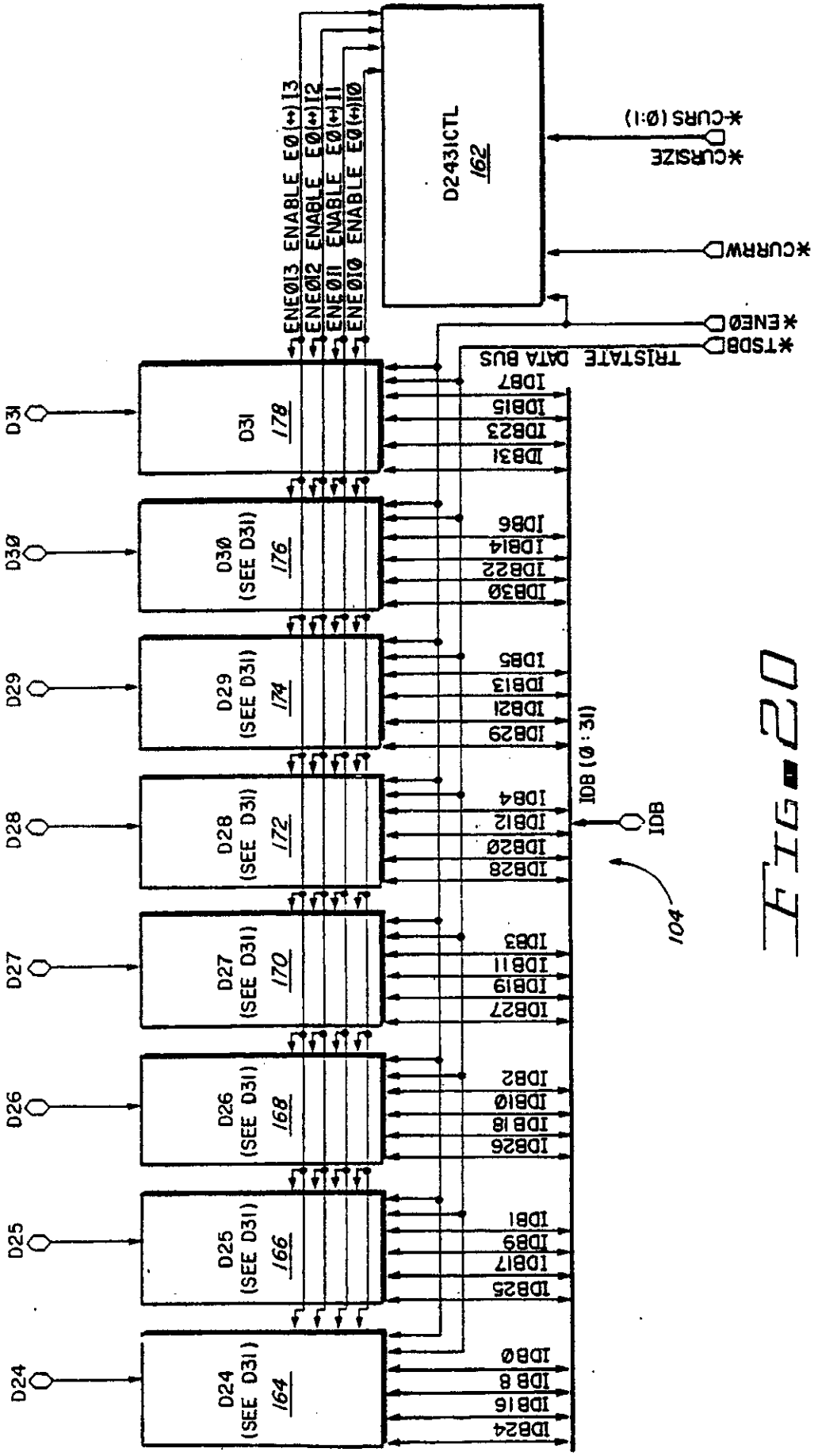
D2431CTL
162
*CURS(0:1)
*CURSIZE
*CURRW
*ENE0
*TSDB
TRISTATE DATA BUS
ENE013 ENABLE E0(+)13
ENE012 ENABLE E0(+)12
ENE011 ENABLE E0(+)11
ENE010 ENABLE E0(+)10
D31
178
D30
(SEE D31)
176
D29
(SEE D31)
174
D28
(SEE D31)
172
D27
(SEE D31)
170
D26
(SEE D31)
168
D25
(SEE D31)
166
D24
(SEE D31)
164
IDB7
IDB15
IDB23
IDB31
IDB6
IDB14
IDB22
IDB30
IDB5
IDB13
IDB21
IDB29
IDB4
IDB12
IDB20
IDB28
IDB3
IDB11
IDB19
IDB27
IDB2
IDB10
IDB18
IDB26
IDB1
IDB9
IDB17
IDB25
IDB0
IDB8
IDB16
IDB24
IDB(0:31)
IDB
104

FIG. 20

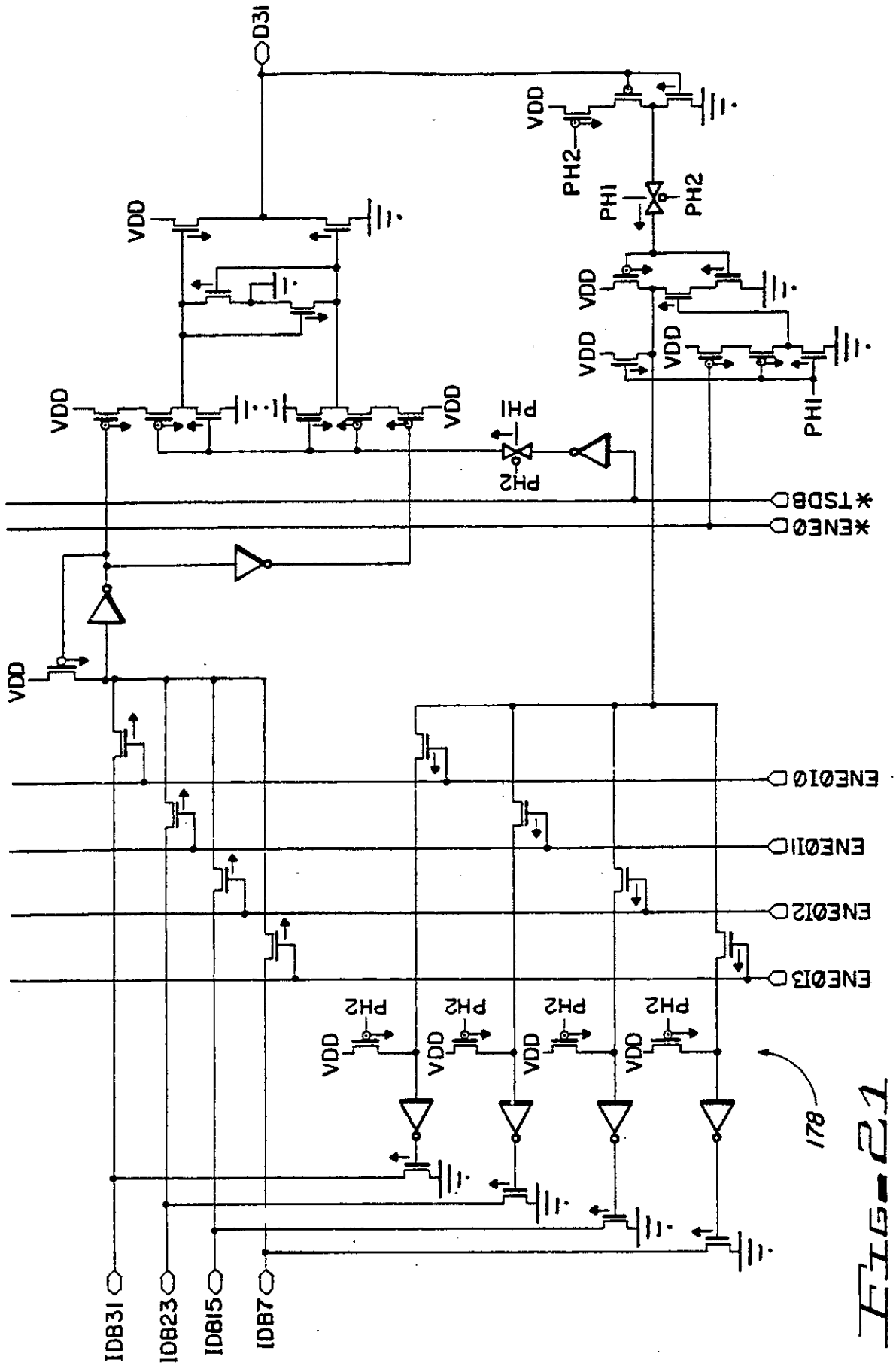
D31
VDD
PH2
PH1
PH2
VDD
VDD
VDD
VDD
VDD
VDD
PH1
PH1
PH2
*TSDB
*ENE0
VDD
ENE010
ENE011
ENE012
ENE013
PH2
PH2
PH2
PH2
VDD
VDD
VDD
VDD
178
IDB31
IDB23
IDB15
IDB7

Fig-21

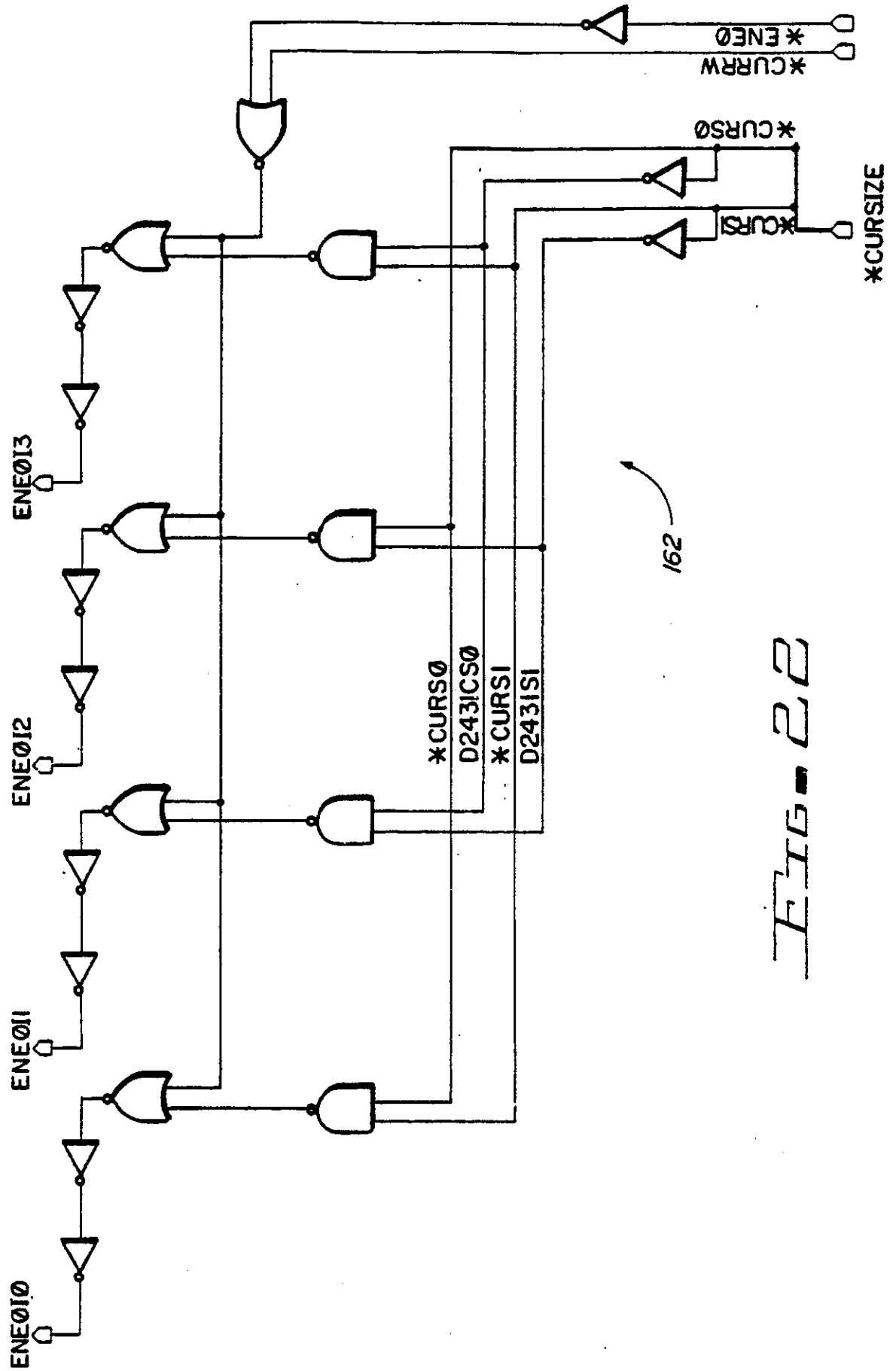
*ENE0
*CURRW
*CURS0
*CURSI
*CURSIZE
ENE013
ENE012
ENE011
ENE010
*CURS0
D243ICS0
*CURSI
D243ISI
162

Fig. 22

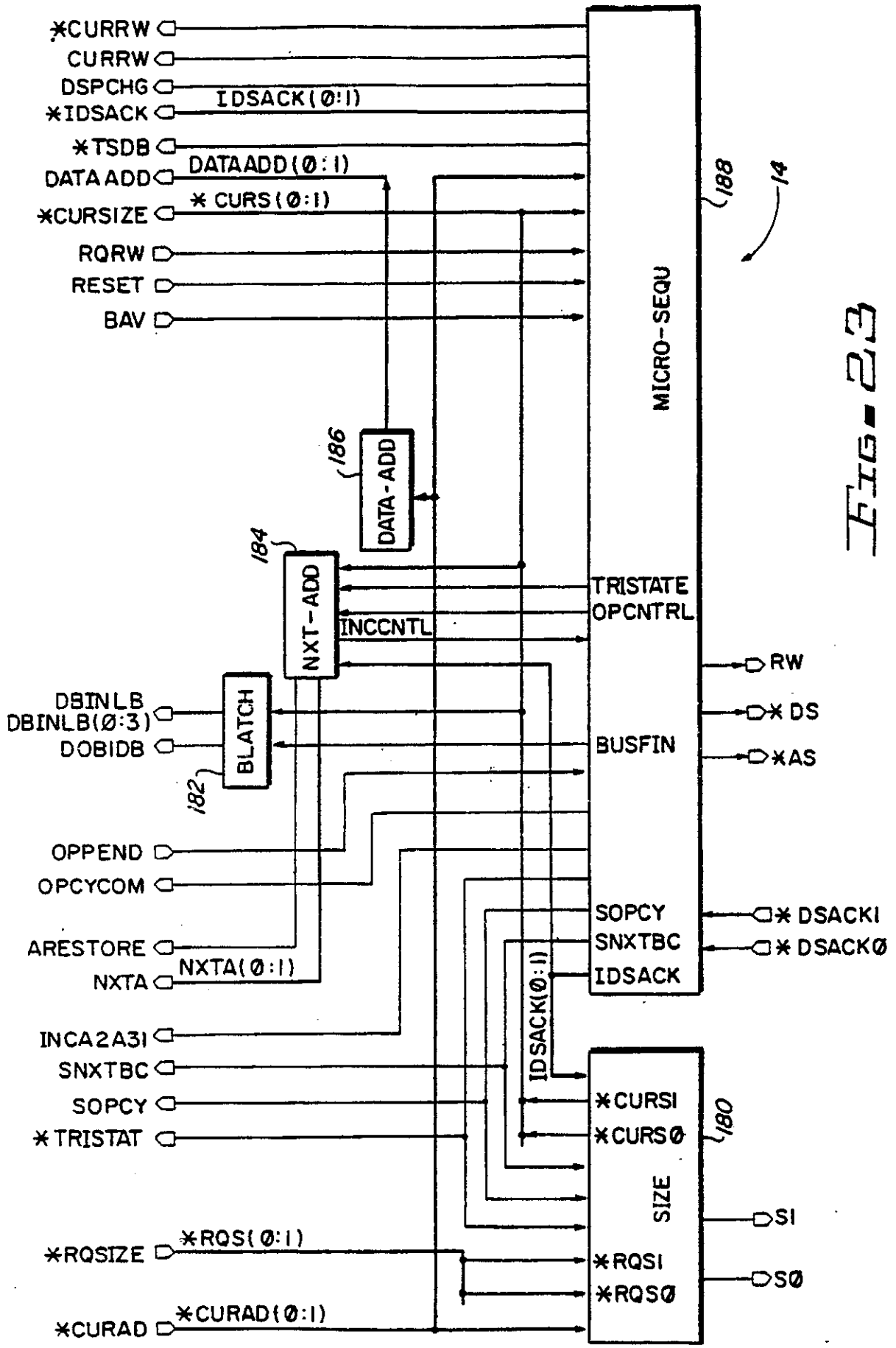
*CURRW
CURRW
DSPCHG
*IDSACK
IDSACK(0:1)
*TSDB
DATAADD
DATAADD(0:1)
*CURSIZE
*CURS(0:1)
RQRW
RESET
BAV
MICRO-SEQU
188
14
186
DATA-ADD
184
NXT-ADD
INCCNTL
TRISTATE
OPCNTRL
RW
*DS
*AS
DBINLB
DBINLB(0:3)
DOBIDB
BLATCH
182
BUSFIN
OPPEND
OPCYCOM
ARESTORE
NXTA
NXTA(0:1)
SOPCY
SNXTBC
IDSACK
*DSACKI
*DSACK0
IDSACK(0:1)
INCA2A3I
SNXTBC
SOPCY
*TRISTAT
*CURSI
*CURS0
180
SIZE
SI
*RQSIZE
*RQS(0:1)
*RQSI
*RQS0
S0
*CURAD
*CURAD(0:1)

Fig-2A

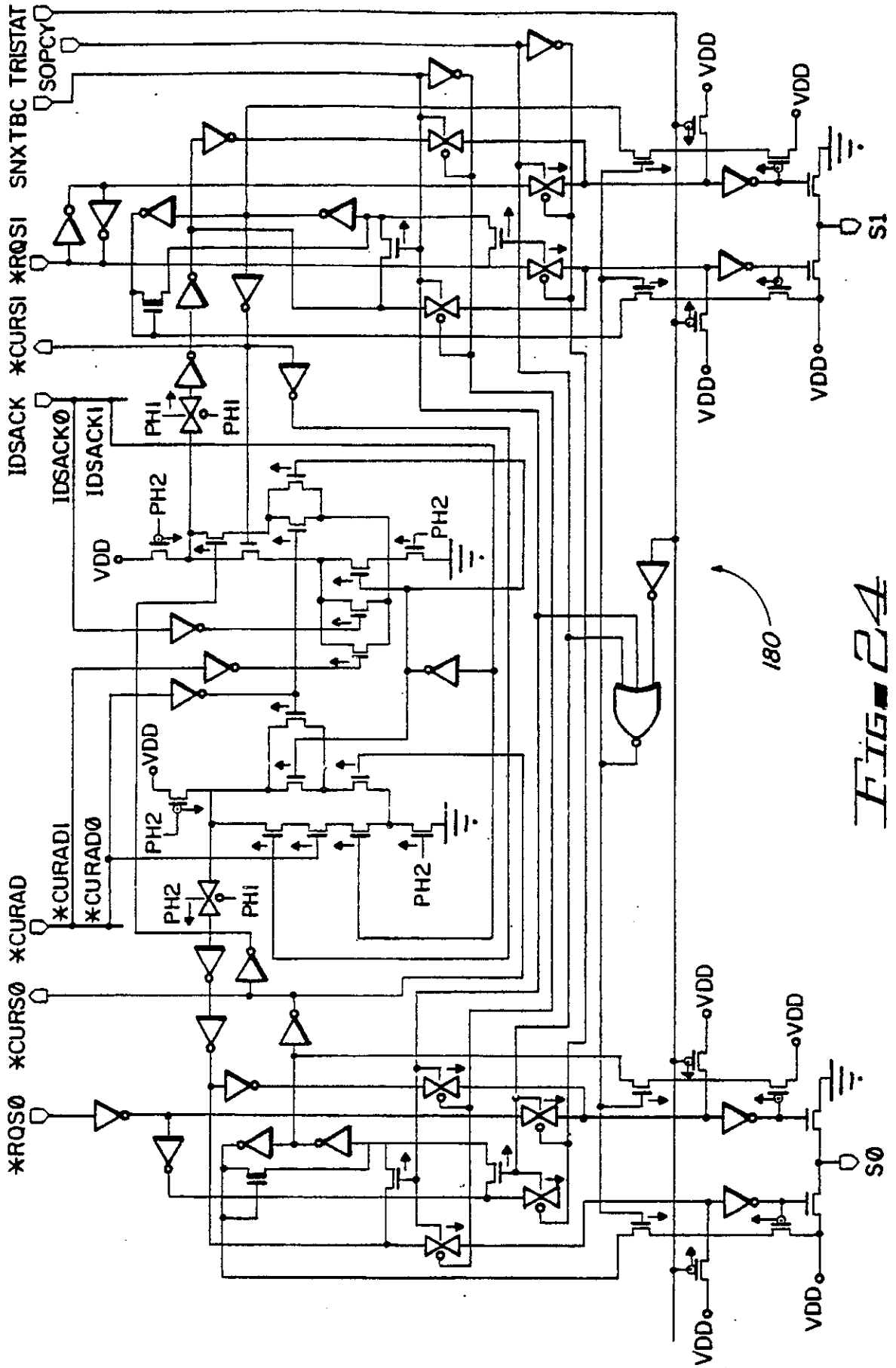
TRISTAT
SOPCY
SNXTBC
*RQSI
*CURSI
IDSACK
IDSACK0
IDSACK1
PHI
PH2
VDD
S1
*RQS0
*CURS0
*CURAD
*CURAD1
*CURAD0
S0
180

Fig. 24

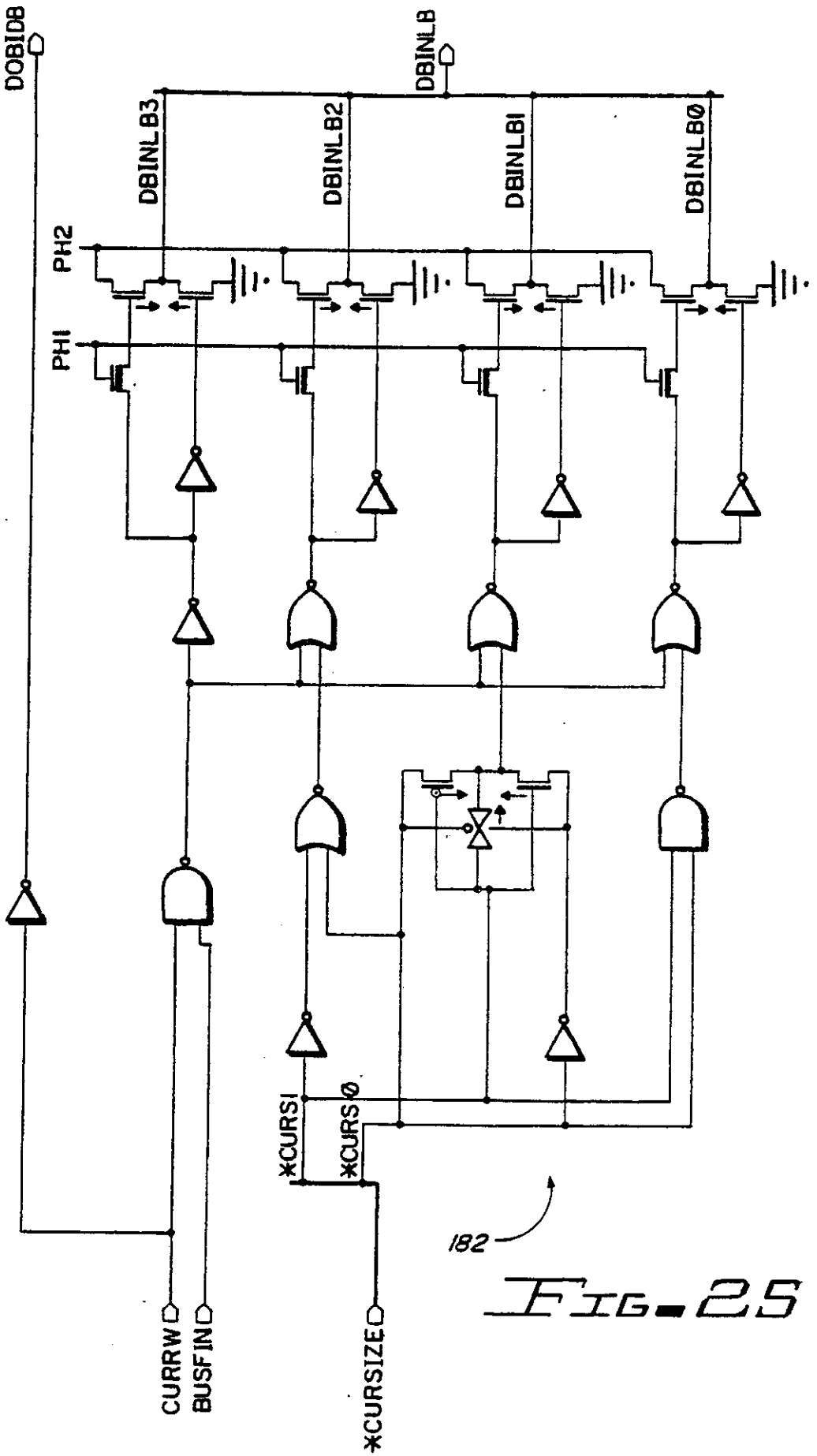
DOBIDB
DBINLB
DBINLB3
DBINLB2
DBINLBI
DBINLB0
PH2
PHI
*CURSI
*CURS0
182
CURRW
BUSFIN
*CURSIZE

FIG-25

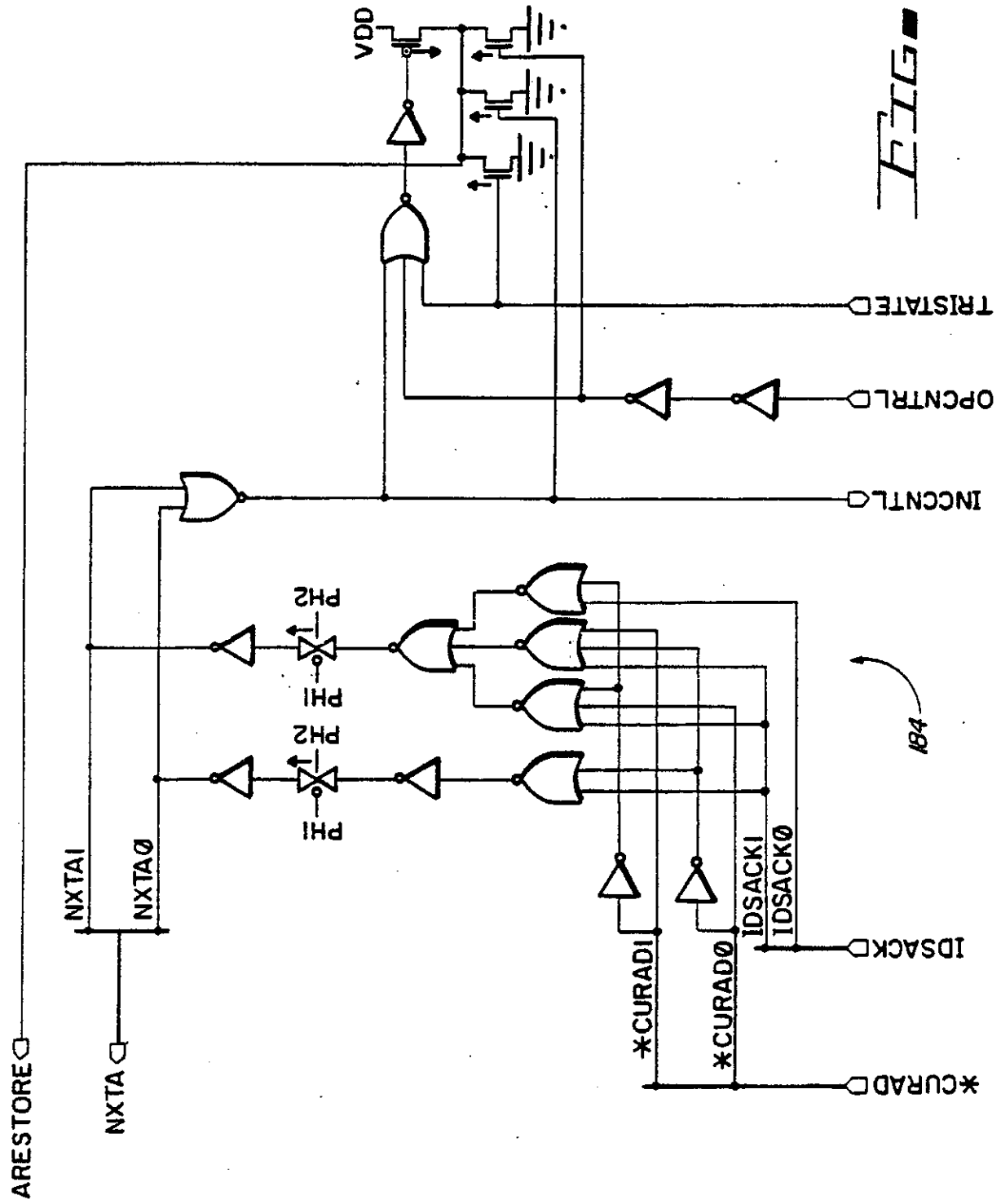
VDD
TRISTATE
OPCNTRL
INCCNTL
PH2
PH1
PH2
PH1
184
NXTAI
NXTAØ
IDSACKI
IDSACKØ
IDSACK
*CURADI
*CURADØ
*CURAD
ARESTORE
NXTA

Fig. 26

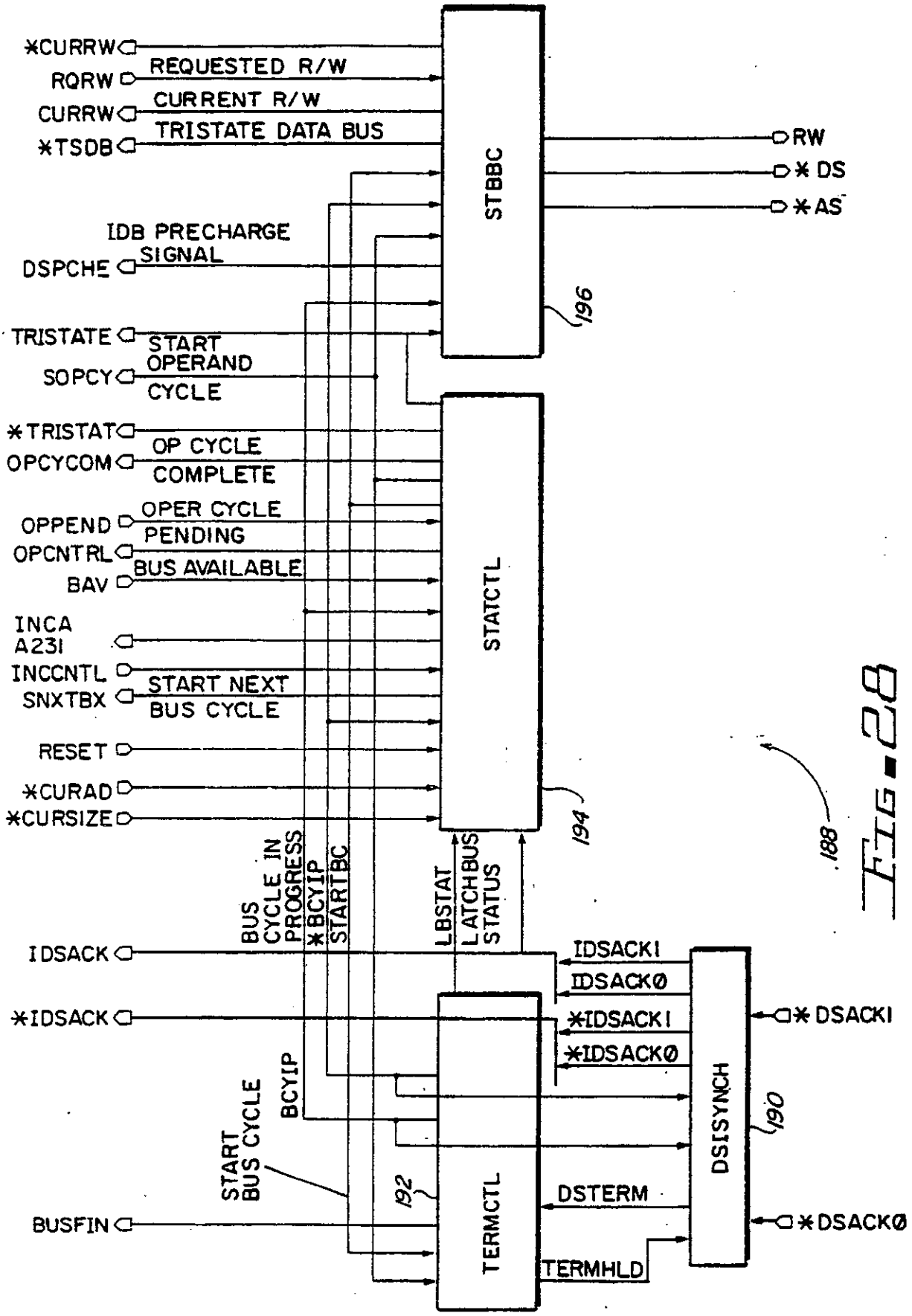
*CURRW
RQRW
REQUESTED R/W
CURRW
CURRENT R/W
*TSDB
TRISTATE DATA BUS
STBBC
RW
* DS
*AS
IDB PRECHARGE SIGNAL
DSPCHE
196
TRISTATE
START OPERAND CYCLE
SOPCY
*TRISTAT
OP CYCLE COMPLETE
OPCYCOM
OPER CYCLE PENDING
OPPEND
OPCNTRL
BUS AVAILABLE
BAV
INCA A23I
INCCNTL
START NEXT BUS CYCLE
SNXTBX
RESET
*CURAD
*CURSIZE
STATCTL
194
188
BUS CYCLE IN PROGRESS
*BCYIP
STARTBC
LBSTAT
LATCH BUS STATUS
IDSACK
*IDSACK
IDSACKI
IDSACKØ
*IDSACKI
*IDSACKØ
* DSACKI
DSISYNCH
190
BCYIP
START BUS CYCLE
192
TERMCTL
DSTERM
BUSFIN
TERMHLD
*DSACKØ

Fig-28

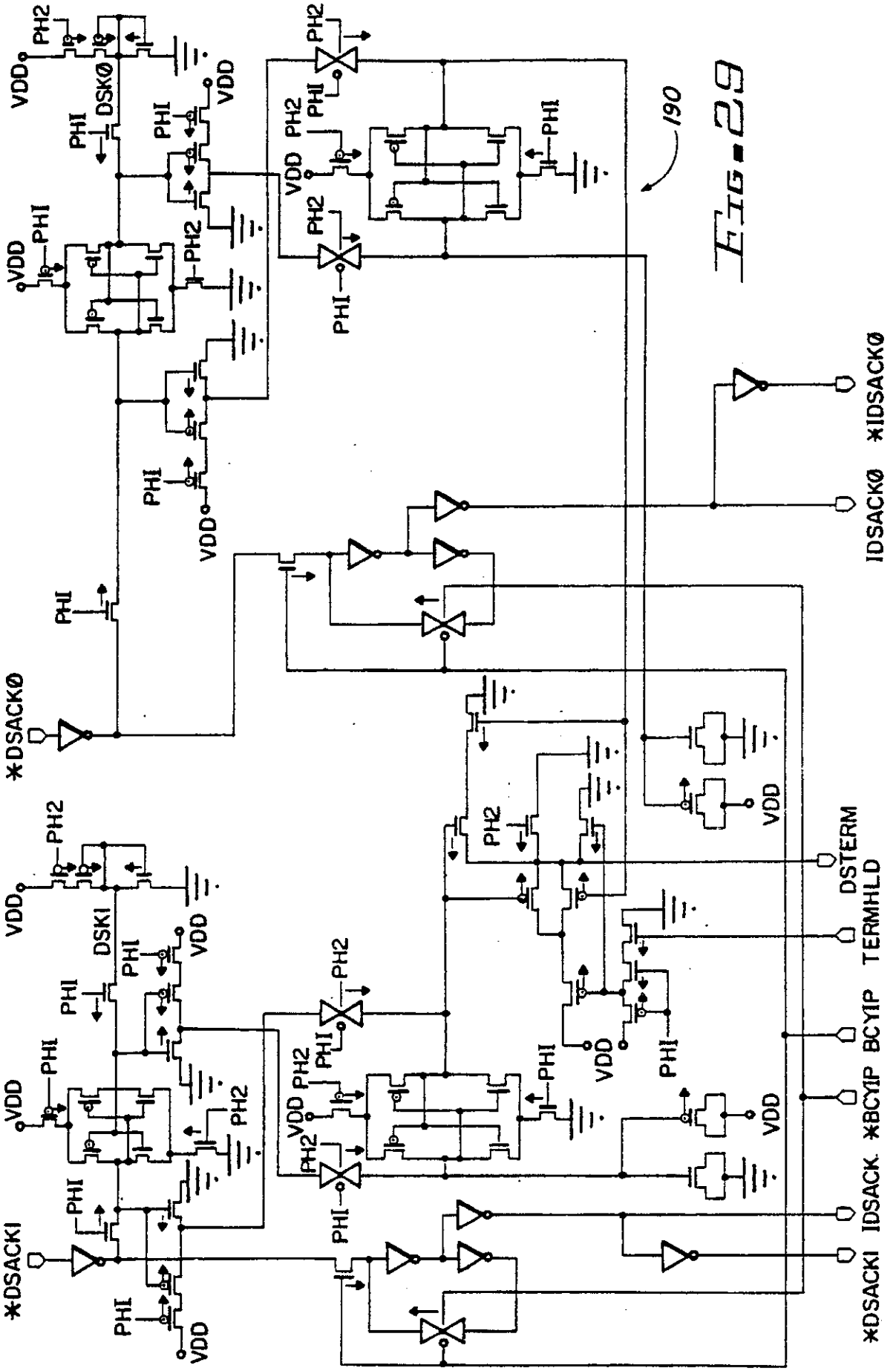
VDD
PH2
PHI
DSKØ
*DSACKØ
IDSACKØ
*IDSACKØ
DSKI
*DSACKI
IDSACK
*BCYIP
BCYIP
TERMHLD
DSTERM
190

FIG. 29

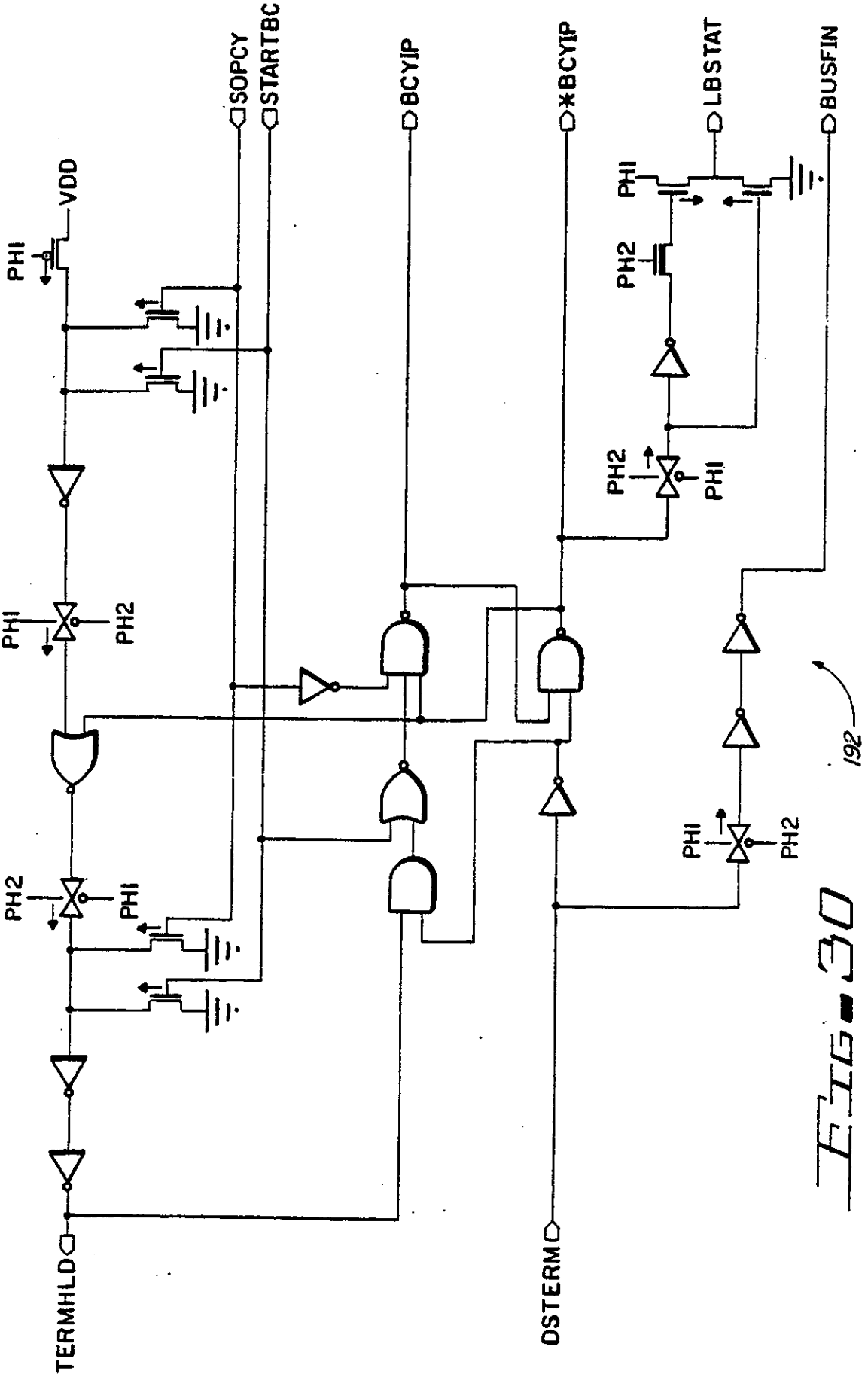
VDD
PHI
SOPCY
STARTBC
BCYIP
*BCYIP
LBSTAT
BUSFIN
PH2
PHI
TERMHLD
DSTERM
192

FIG. 30

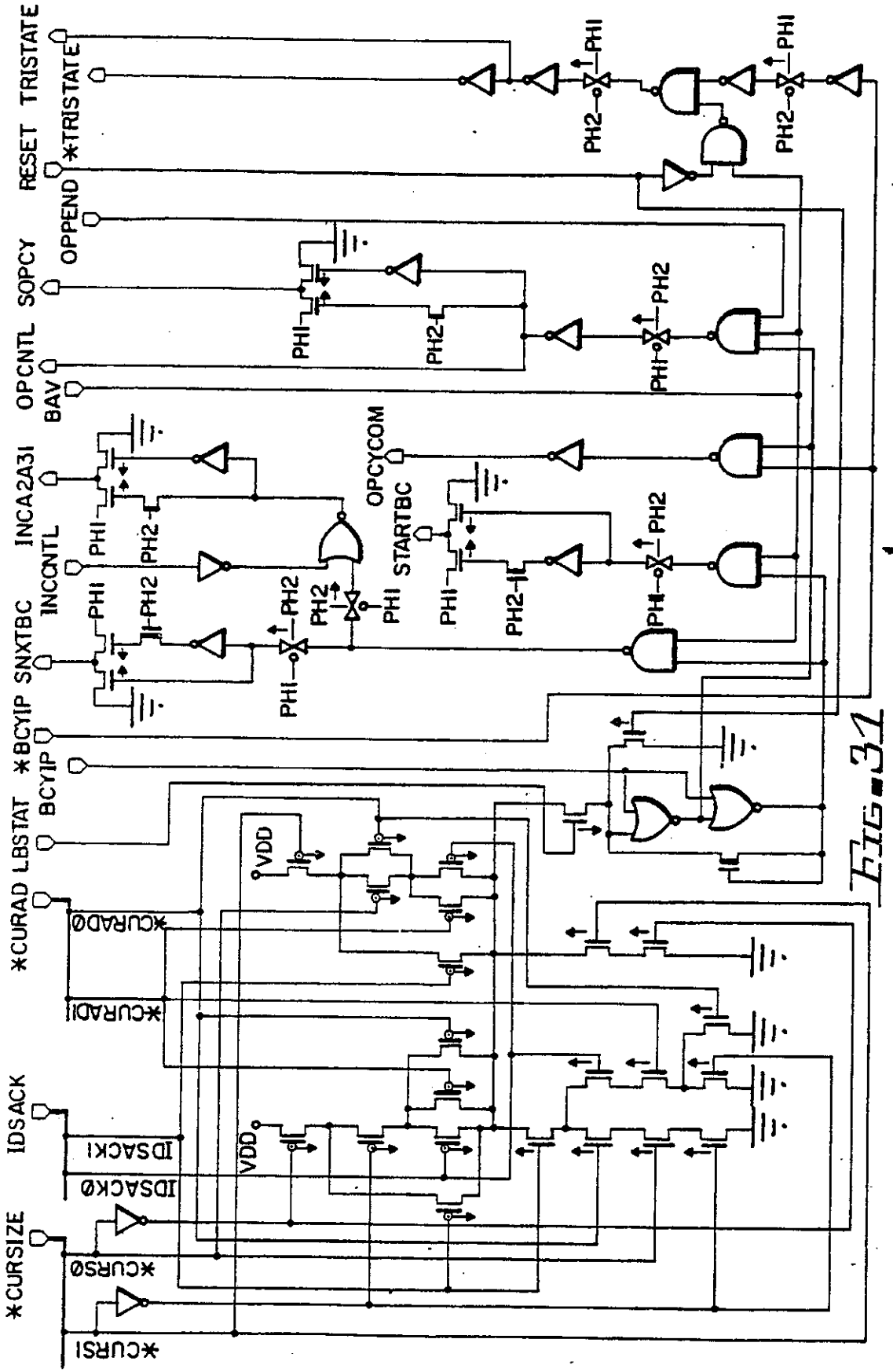
RESET TRISTATE
*TRISTATE
OPPEND
SOPCY
OPCNTL
BAV
INCA2A3I
INCCNTL
SNXTBC
*BCYIP
BCYIP
LBSTAT
*CURAD
IDSACK
*CURSIZE
OPCYCOM
STARTBC
PHI
PH2
VDD
*CURAD0
*CURAD1
IDSACK1
IDSACK0
*CURS0
*CURS1
194

Fig. 31

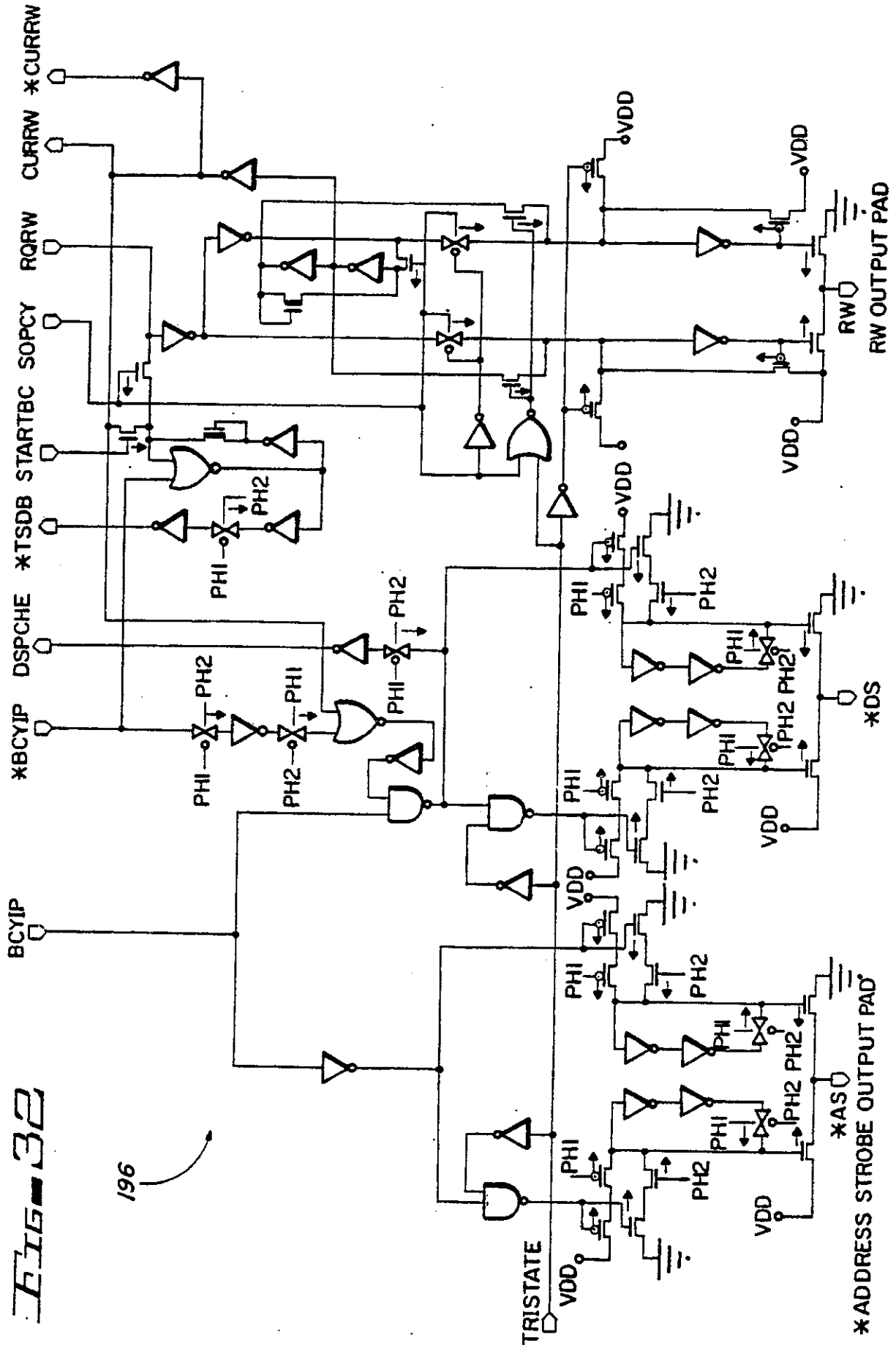
*CURRW
CURRW
RQRW
SOPCY
STARTBC
*TSDB
DSPCHE
*BCYIP
BCYIP
TRISTATE
RW
RW OUTPUT PAD
*DS
*AS
*ADDRESS STROBE OUTPUT PAD
VDD
PHI
PH2
196

Fig-32

REGISTER ENTRY FOR EP0185676

European Application No EP85901904.4 filing date 12.04.1985

Priority claimed:
26.06.1984 in United States of America - doc: 624660

PCT EUROPEAN PHASE
PCT Application PCT/US85/00656 Publication No WO86/00436 on 16.01.1986

Designated States DE FR GB IT NL

Title DATA PROCESSOR HAVING DYNAMIC BUS SIZING

Applicant/Proprietor
MOTOROLA, INC., 1303 East Algonquin Road, Schaumburg, Illinois 60196, United States of America [ADP No. 50019769003]

Inventors
S. DAVID MOTHERSOLE, 903 Huntridge Drive, Austin, TX 78758, United States of America [ADP No. 54133178001]

M. LESTER CRUDELE, 69 Hill Road, Groton, MA 01450, United States of America [ADP No. 54133186001]

L. JAMES TIETJEN, 3423 Rosefinch Trail, Austin, TX 78746, United States of America [ADP No. 54133194001]

R. ROBERT THOMPSON, 11104 Township Cove, Austin, TX 78759, United States of America [ADP No. 54133202001]

Classified to
G06F

Address for Service
MOTOROLA, European Intellectual Property Operation, Jays Close, Viables Industrial Estate, BASINGSTOKE, Hampshire, RG22 4PD, United Kingdom [ADP No. 00001180001]

EPO Representative
HAROLD IBBOTSON, Motorola Ltd Patent and Licensing Operations - Europe Jays Close Viables Industrial Estate, Basingstoke Hampshire RG22 4PD, United Kingdom [ADP No. 50515998001]

Publication No EP0185676 dated 02.07.1986
Publication in English

Examination requested 12.02.1986

Patent Granted with effect from 18.09.1991 (Section 25(1)) with title DATA PROCESSOR HAVING DYNAMIC BUS SIZING.

18.12.1989 Notification from EPO of change of EPO Representative details from
HAROLD IBBOTSON, Motorola Ltd Patent and Licensing Operations - Europe Jays Close Viables Industrial Estate, Basingstoke Hampshire RG22 4PD, United Kingdom [ADP No. 50515998001]
to
MICHAEL HOLT, MOTOROLA European Intellectual Property Operations Jays Close, Viables Ind. Est., Basingstoke Hants, RG22 4PD, United Kingdom [ADP No. 56887243001]
Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

08.02.1991 MOTOROLA, European Intellectual Property Operation, Jays Close, Viables Industrial Estate, BASINGSTOKE, Hampshire, RG22 4PD, United Kingdom [ADP No. 00001180001]
registered as address for service
Entry Type 8.11 Staff ID. CA1 Auth ID. AA

19.08.1991 Notification from EPO of change of Applicant/Proprietor details from
MOTOROLA, INC., 1303 East Algonquin Road, Schaumburg, Illinois 60196, United States of America [ADP No. 50019769003]
to
MOTOROLA, INC., 1303 East Algonquin Road, Schaumburg, IL 60196, United States of America [ADP No. 50019769003]
Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

19.08.1991 Notification from EPO of change of EPO Representative details from
MICHAEL HOLT, MOTOROLA European Intellectual Property Operations Jays Close, Viables Ind. Est., Basingstoke Hants, RG22 4PD, United Kingdom [ADP No. 56887243001]
to
MICHAEL HOLT, MOTOROLA European Intellectual Property Operations Jays Close Viables Industrial Estate, Basingstoke Hampshire RG22 4PD, United Kingdom [ADP No. 56887243001]
Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

**** END OF REGISTER ENTRY ****

RENEWAL DETAILS

PUBLICATION NUMBER	EP0185676

PROPRIETOR(S)

MOTOROLA, INC., 1303 East Algonquin Road, Schaumburg, IL 60196, United States of America

DATE FILED	12.04.1985
DATE GRANTED	18.09.1991
DATE NEXT RENEWAL DUE	12.04.1994
DATE NOT IN FORCE	
DATE OF LAST RENEWAL	29.04.1993
YEAR OF LAST RENEWAL	09
STATUS	PATENT IN FORCE