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

(11) Publication number: **0 165 665 A2**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 85302464.4

(22) Date of filing: 09.04.85

(51) Int. Cl.⁴: G 09 G 1/16
G 09 G 1/28

(30) Priority: 16.04.84 US 600688

(43) Date of publication of application: 27.12.85 Bulletin 85/52

(84) Designated Contracting States: DE FR GB

(71) Applicant: TEXAS INSTRUMENTS INCORPORATED
13500 North Central Expressway
Dallas Texas 75265(US)

(72) Inventor: Leach, Jerald G.
6710 Leandra
Houston Texas(US)

(74) Representative: Abbott, David John et al,
Abel & Imray Northumberland House 303-306 High Holborn
London, WC1V 7LH(GB)

(54) **Sprite collision detector.**

(57) A sprite collision detector includes the combination (1) of a video processor including a color palette for processing of data, a memory (31) for storing of data to be processed, a monitor (33) and (35) for displaying of the processed data, a processor (30) for controlling the transfer of data from the memory (30) to the monitor (33) and (35). The video processor (1) includes a control logic (65) for controlling the transfer of data and instructions between the processor (30) and the video processor (1); a memory control logic (67) for controlling, in response to the instructions from the processor (1), the transfer of data from the memory (31); sprite processor (10) for converting data to color data and to provide the color data for images displayed by the monitor (33) and (35); register (63) for arranging data from the memory (31) in a predetermined order and to apply the data according to the prearranged order to the sprite processor (10); and video output logic (57) for converting the color data to a video signal to which the monitor (33) and (35) will respond to provide a display of a predetermined pattern and of a predetermined color.

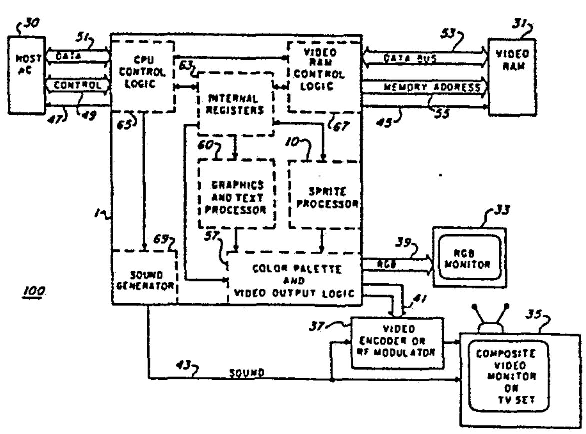



Fig.1

EP 0 165 665 A2

Croydon Printing Company Ltd.

# SPRITE COLLISION DETECTOR

## BACKGROUND OF THE INVENTION

The invention relates generally to video display devices and, more particularly, but not by way of limitation, to a video display processor which can superimpose one or more mobile patterns at selected locations on a larger, fixed pattern image and to indicate when two or more of the module multiple patterns trv to occupy the same location or intersect one another.

The basic principle for superimposing one or more mobile patterns at selected locations on a larger, fixed pattern image was described and claimed in U.S. Patent 4,243,984, assigned to the assignee of the present invention. Other systems which disclose moveable patterns are provided in the following U.S. Patent Numbers 4,112,422; 4,129,858; 4,034,990; 4,107,664; 4,016,362; 4,116,444; 3,771,155; 4,296,476; 4,232,374; 4,177,462; and 4,119,955.

## SUMMARY OF THE INVENTION

A moveable graphics pattern collision detector hereafter referred to as a sprite collision detector, detects a collision between two or more sprites during the display of video graphics. The graphics are generated and are displayed on a monitor and includes a plurality of moveable sprites divided into a plurality of sprite groups. The location addresses on the monitor of each group of sprites is stored in a register and a coincidence dector detects when sprites from two different groups occupy the same position that is stored in the video RAM.

The sprite collision detector includes a terminal for accepting data inputs and converting the data inputs to digital signals representative of the data inputs. A memory stores the sprite and background information and a processor processes the digital signals from the terminal to obtain program sequences for transferring sprite and background data from the memory means to be displayed on a video monitor.

A coincidence register indicates when two or more sprites try to occupy the same position on the monitor screen and will through the implementation of a single bit change indicate the occurance of a collision. The number of the two groups enables the collision detector to indicate the collision and identify the groups that were involved in the collision with a minimum amount of microprocessor processing time.

It is the object of the invention to provide an advance video processor that has included therein a collision detector circuit that will detect the collisions between two groups of moveable patterns on a graphics display system.

It is yet another object of the invention to provide an advance video display system in which a large group of moveable patterns, sprites, are broken into a smaller plurality of groups by a software routine. Each group is assigned a unique bit in a group register which is used to flag a coincidence of sprites. When a sprite from a first group collides with a sprite from a different group, two bits are set. The collision register is cleared by the microprocessor reading of the register and thus a minimal amount of software is required and computer processing time.

It is yet another object of the invention to provide an advance video processor in which the sprites are divided into a plurality of groups which may be represented by a minimal number of bits.

It is yet another object of the invention to provide the capabilities of using the remainder bits in a byte to indicate color and early sprite.

These and other objects and advantages of the present invention will be more evident from a reading of the specification in conjunction with the figures in which:

## BRIEF DESCRIPTION OF THE FIGURES

Figure 1 is a block diagram of a video display system according to the invention.

Figure 2 is a block diagram of the advance video processor of Figure 1;

Figure 3 is a diagram illustrating theapproaching coincidence of two sprites;

Figure 4 is a diagram indicating the use of sprites for a computer game;

Figure 5 is a diagram illustrating the use of sprites to create a graphics display;

Figure 6 is a diagram of a byte and bit assignments of the byte according to the invention;

Figure 7 is a block diagram of an alternate embodiment of the invention;

Figure 8 is a block diagram illustrating the use of a direct memory address capabilities of the advance video processor according to the invention;

Figure 9 is a bus assignment of the advance video processor's data bus;

Figure 10 through 14 are register assignment layouts;

Figure 15 is a color assignment design;

Figure 16 is a status bit assignment design;

Figure 17 is a schematic diagram of the advance video display processor

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In Figure 1, to which reference should now be made, there is shown a block diagram of a video display system 100 incorporating an advance video processor 1 according to the invention. A host microcomputer 30 (CPU) interfaces with an Advance Video Processor (AVDP) 1 via a bidirectional data bus 51, a control bus 49 and an interrrupt line 47. The AVDP 1 is used to interface microprocessor 30 to a color video monitor 33. The AVDP 1 uses a dynamic RAM 31 to store the information displayed on the video screen. The microprocessor 30 loads the AVDP's 1 configuration registers through the 8 bit CPU to AVDP data bus 51. The microprocessor 30 then loads the video RAM 31 with the information that is to be displayed on a video screen 32. The AVDP 1 refreshes the video screen 32 independently of CPU accesses. The video RAM 31 is accessed by the AVDP 1 through an 8 bit address bus and 8 bit data bus. The AVDP 1 also supplies the necessary RAS (Row Address Strobe) and CAS (Column Address Strobe) to interface the dynamic video RAM 31 to AVDP 1. Additionally, connected to the advance video processor 1 is a random access memory, video RAM 31, which is connected to the advance video processor 1 via bydirectional data bus 53, memory address bus 55 and a control lines 45. The graphics are displayed on either one or two possible systems, a Red, Green, and Blue (RGB) monitor 33 which is connected to the advance video processor 1 via an RGB bus 39 or a composite video monitor or TV set 35 which is connected to the advance video processor 1 via a color difference bus 41 and a video encoder or RF monitor 37. Additionally, sound is provided to the composite video monitor or TV set 35 via a sound bus 43. The advanced video processor 1 includes 7 basic function blocks. These include the CPU control logic 65 which handles the interface between the host microcomputer 30 and the advance video processor 1 and is the termination portion of the control lines 49, the input and output of data to data bus 51 and the providing of interrupts to the host microcomputer 30 via interrupt line 47. CPU control logic 65 enables the host microcomputer 30 to conduct five basic operations. These include the writing of data into the video RAM 31, the reading of data from the video RAM 31, the writing of data to the advance video display processor (AVDP) 1's internal registers 63, the reading of data from some of the advance video processor 1's internal registers 63 and the writing to an internal sound generator 69 that is contained within the advance video processor display logic.

The type and direction of data transfers are controlled by the control lines 49 and in particular CSW, CSR, and mode input lines. CSW is the CPU 30 to AVDP 1 write select line. When CSW is active low the eight bits on the CDO-CD7 of the data lines 51 are strobed into the video display processor. CSR is the CPU to AVDP read select line. When CSR is active low the AVDP outputs eight bits of data onto the CDO-CD7 lines for the CPU to read. When CSW and CSR are both active low the sound generator 69 is addressed.

Mode determines the source or destination of a read or write transfer. Mode is generally connected to a CPU low order address line.

Figure 9 provides an illustration of the data transfer between the host CPU 30 to the AVDP 1. A video RAM control logic 67 controls the interface between the advance video processor 1 and the video RAM 31 and handles the transfer of data from the data bus 53 that is provided to the video RAM 31 at the memory address location that is provided on the memory address bus 55 in response to the control signals that are provided on the control lines 4. In the embodiment shown, the data bus 53 is an 8 bit bidirectional bus and the memory address bus 55 is an 8 bit multiplex address bus. The advance video processor illustrated in Figure 1 can directly address 16K bytes, two (TMS4416s or equivalent, 32 bytes, 4 TMS4416s or equivalent, or 65K bytes 8 (TMS41664S) (all TMS parts are manufactured by Texas Instruments or equivalent) while currently providing dynamic refresh to the video RAM 31.

The internal registers 63 in the embodiment shown in Figures 1 and 2 contain two read only registers, a status register and a sprite collision register in Figure 16 and forty nine write only registers illustrated in Figures 11. The write only registers provide the following functions. Three of the write ony registers define the mode of operation of the advance video display processor 1 and specify options such as the mode of operation and type of video signal output necessary to drive the RGB monitor 33 or the composite video mointor or TV set 35. Six of the write only registers that are contained within the internal register block 63 are designated advance video display procesor 1 to display memory address mapping reigsters and specify locations in the video RAM 31. One write only register is a color code register and defines colors when the video display processor 10 is operating in the text mode. Two separate registers are scrolling registers; one is horizontal scrolling the other is for vertical scrolling. One programmable interrupt register enables the advance video processor 1 to be reconfigured during a horizontal retrace interval that occurs in all television monitor signals. Four block move address and decament counter registers allow a defined block of video memory to be moved to another video memory location. Thirty two palette pilot registers define up to 16 displayable colors (from a 52 color palette) per horizontal scan lines.

The read only registers provide the following functions. A status register contains flags for interrupts, coincidence and eleventh sprite occurance on any one horizontal line. The AVDP has a single 8-bit status register 28 which can be read by the CPU 1. The format of the status register 28 is shown in Figure 12. The status register contains the interrupt pending flag (F), the sprite coincidence flag (C), the eleventh sprite flag (11S), and the eleventh sprite number if one exists.

The status register 28 may be read at any time to test the F,C and 11S status bits. Reading the status will clear the interrupt flag F. However, asynchronous reads of the status will cause the frame flag (F) bit to be reset and therefore possibly missed. Therefore the status register should only be read when the AVDP 1 interrupt is pending. It requires only one data transfer to read the status register 28.

Interrupt Pending Flag (F)

The F status flag in the status register is set to 1 whenever there is an interrupt pending. This bit will be set one of three ways; when a block move has completed, when a programmable interrupt is selected, or when an end of frame has occurred (Vertical Retrace Period). The interrupt pending flag is reset to 0 when the status register is read or by the external reset.

When the appropriate interrupt enable bit (IE bit 2 of write only register 1 or PIE bit 2 of write only register 10) is set to 1 (INT) will be active low whenever the F status flag is a logic 1.

Note the status register needs to be read after each interrupt in order to clear the interrupt and receive the new interrupt on the next occurence.

Coincidence Flag (C)

The C status flag in the status register is set to a 1 if two or more sprites coincide. Coincidence occurs if any two sprites on the screen have one overlapping pixel. Transparent colored sprites, as well as those that are partially or completely off the screen, are also considered. The C flag is cleared to a 0 after the status register is read or the AVDP is externally reset. The status register 28 should be read immediately upon powerup to ensure that the coincidence flag is reset.

The AVDP 1 checks each pixel position for coincidence during the generation of the pixel regardless of where it is located on the screen. This occurs every 1/60th of a second. Therefore when moving more than one pixel position during these intervals it is possible for the sprites to have multiple pixels overlapping or even to have passed completely over one another when the AVDP 1 checks for coincidence.

Eleventh Sprite Flag (11S) and Number

The 11S status flag in the status register is set to a 1 whenever there are 11 or more sprites on a horizontal line (lines 0 to 209 depending on the mode chosen) and the frame flag (F) is equal to 0. The 11S status flag is cleared to a 0 after the status register is read or the AVDP is externally reset. The number of the 11th sprite is placed into the lower 5 bits of the status register when the 11S flag is set and is valid whenever the 11S flag is 1. The setting of the 11th sprite flag will not cause an interrupt.

A sprite collision detection register detection register defines which group or groups of sprites have collided.

A sprite collision register 83 is an 8 bit register that can be used to determine which groups of sprites collided. The sprite color byte is composed of 4 color bits, an early clock bit, and 3 remaining bits; these 3 reamining bits are used to divide the sprites into eight groups. Each bit in the sprite collision register 83 corresponds to one group. Therefore, whenever 2 sprites collide one or more of these bits are set. This register is cleared by a CPU read to this register. Figure 6 shows the layout of these groups in the sprite collision register 83. It requires 3 data transfers to read this register.

A sprite processor 10 incorporates full sprite control on the advance video display processor 1 which in the embodiment shown on a single chip. The sprite processor 10 includes the features which with as many as 10 sprites may occur (in the embodiment shown in Figure 1) on a single horizontal scan line. Previous video display processors were limited to only four sprites per line. The sprites may be multi-color or single color with each horizontal half scan line of the sprite having the option of being a different color from the sprite. Additionally, unique sprite coincident detection is provided in the embodiment of the disclosure. A coincidence occurs if any two sprites on the display have at least one overlapping pixel. Sprite mapping necessary to provide this feature is contained in the video RAM 31.

Graphics and text processing is provided by a graphics and text processor 60 in which the host microprocessor 30 configures the advance video display processor 1 to operate in one of the following display modes in the embodiment shown in Figure 1:

A first graphic display mode in which resolution with two colors for each of an 8x8 pixel block in a 256 x 192 pixels display;

Graphics 2 made which provides two colors for each 8 x 1 pixel block in a 256 x 192 pixel display;

Graphics 3 provides two colors for each 4 x 2 pixel blocks for a 256 x 192 pixel display;

Graphics 4 provides high resolution with two colors for each 8 x 1 pixel block in a 512 x 192 total pixel resolution;

A graphics 5 provides a full bit map of 256 x 210 pixel resolutions;

A first text mode provides 40 columns by 24 rows of text; and

A second text mode provides 80 columns x 24 rows of text. All text and graphics modes with the exception of the full bit map mode designated as graphics 5 are table driven.

A sound generator 1 provides in the embodiment shown in Figure 1 on chip sound generation that is compatible with the devices such as an SN764889 device manufactured by Texas Instruments Incorporated. The circuit provides 3 programmable tone generators; one programmable noise generator; a 120 to 100,000 HTZ frequency response and 15 programmable attenuation steps from 2dB to 28dB in steps of 2dB.

Figure 2, to which reference should now be made, is a block diagram of the advance video processor 1 of Figure 1. As was discussed earlier in conjunction with Figure 1, there are included in the internal registers 63 two read-only registers and forty nine write-only registers. Included in these are color palette registers 2 which are 16 registers of 9 bits each for 16 colors. The color palette registers 2 are addressed by a sprite control logic 59; a first color buffer 61; a second color buffer 62 and a third color buffer 64 which are a part of the grphics and text processor 60; a border color register 29; and a text color register 30 which provide program colors.

It should be noted that in the advance video display processor 1 the embodiments of Figures 1 and 2 does not fetch color for each character in the text mode as it does in the graphics mode. A color palette read logic 65 addresses the color palette registers 2 to place the contents contained within the color palette registers on a D-to-A logic 67 which as was discussed in conjunction with the color palette and video output logic 57 of Figure 1, provides the Red, Green and Blue colors to either the RGB monitor 33 or the different signal to the video encoded RF modulator 37. Depending on the configuration of the advance video processor 1, the output of the D-to-A logic 67 is placed on either the RGB bus 39 or the different color bus 41.

A color palette write logic 3 controls the loading of the color codes into the color palette register 2 which includes registers R32 through R63 of Figure 11. The format for the palette is shown in Figures 13 and 14. The palette consists of sixteen 9 bit registers which allows the user to display 16 of 512 colors on the screen at one time. On an external reset the color palette is initiallized with the default values shown in Figure 15 for the color difference outputs.

A horizontal counter, Programmable Logic Array.(PLA) 5, counts positions on the horizontal scan lines and decodes instructions based upon the beam position of the scan and provides timing to the D-to A control logic control logic 67 which is used to identify the sprite position and color. The vertical counter PLA 6 counts rows positions on the scan lines, decodes instructions and provides timing to the sprite register 11 as does horizontal counter PLA as to position color data. Not shown in Figure 2 is the fact that the horizontal counter PLA 5, and vertical counter PLA 6 are connected to the following logic functions.

A color priority logic 7 decides priority of color logics between border color logic 29 text color logic 30, color buffers logic 61, and 64 and sprite control logic 59. The priority is based first on border, then on sprite when in active area, or other sprites and there are three or more dependent colors and 7 modes of operations by which the color·priority logic provides the appropriate color for the advance video display processor 1.

A interrupt logic 8 provides interrupt to the hose CPU 30 that is based upon a timing signal interrupt to load one of the registers. Refer to Figure 16 wherein:

IE = INTERRUPT ENABLE BIT 2 OF REGISTER 28.
F = INTERRUPT FRAME FLAG BIT 0 OF STATUS REGISTER; and
PIE= PROGRAMMABLE INTERRUPT ENABLE BIT 2 OF REGISTER 10

A programmable interrupt logic 29 can provide an interrupt for any horizontal scan or line and in the embodiment shown in Figure 1 includes an eight bit register the contents of which is compared with the contents of the vertical counter PLA 6 and provides an interrupt request to the interrupt logic 8 when there is a comparison between the contents of the two registers indicating that that scan line requires an interrupt in the program sequences being executed by the host CPU 30.

The sprite control logic 59 controls the sprites fetch and checks vertical position from the vertical counter PLA 6 and causes the sprite horizontal position pattern and color data to be fetched.

A sprite control logic 59 processes and checks all of the sprites which in the embodiment of Figure 1 includes 32 sprites to see if their positions are valid. If a sprite is to be loaded on the next scan line, the sprite control logic 59 loads the sprite number or vertical position into a sprite stack 11. The sprite stack 11 places the address of the sprite on the RAM address bus 69 for retrieval from the video RAM 31.

A CPU register 12 interfaces the host microcomputer 30 with the video RAM 31 via the data bus 51 and 51A which is contained within the video processor 1. A name register 13 contains the name of the background pattern (an 8 bit number) which is used to fetch the pattern and color bytes for the next character to be displayed. An address register 14 addresses the video RAM 31 based upon the host microprocessor 30 instructions (whether the instruction is a read or a write instruction) and also addresses the video display processor 1, internal register 63 and color palette registers 57.

The scroll logic includes a vertical state register 22, vertical scroll register 23, character counter 24, horizontal scroll register 25, and horizontal state register 26.

For graphics modes 1,2,3,4 and text modes 1 and 2, the screen is broken up into characters. The character counter 24 counts the characters as the TV scan horizontally and vertically. The horizontal state register 26 determines which pixel of the character is being displayed. The vertical state counter 22 determines which row of the charater is being displayed.

Graphics mode 5 is bit mapped and is not broken up into characters. The horizontal state 26, vertical state 22, and character counter 24 will count pixel by pixel as the TV scans horizontally and vertically in this mode. These counters are usxed to address the video RAM 31. The horizontal scroll register 25 contains an 8 bit number which determines the horizontal scroll location of the screen. At the beginning of each horizontal line the contents of the horizontal scroll register 25 is loaded into the horizontal state register 26 and character counter 24. By changing the starting position of the counters the screen can be scrolled up to 256 different horizontal positions.

The vertical scroll register 23 contains an 8 bit number which determines the vertical scrolling of the screen. At the beginning of each screen scan, the vertical scroll register 23 is loaded into the vertical state register 22 and the character counter 24. By changing the starting position of the counters the screen can be scrolled up to 256 different vertical positions.

The base registers 15, 16, 17, 18 defines the locations in video memory 31 where the sections of video information will be stored. The name base register defines the location of the name table in memory. The color base register 16 defines the location of the video color information. The pattern base register 17·defines the location of the pattern bits used to map each character. The sprite location register 18 defines the location of the sprite patterns, sprite colors, sprite horizontal position, and sprite vertical position. The command registers 19, 20, 21 control the mode of operation of the advanced video processor 1. The operation of each bit is explained in the Table 1 sections 3.2.1, 3.2.2, and 3.2.11.

A status register 28 provides status via data bus 51A to the host microcomputer 30 that reflects the following interrupt information; a programmable interrupt via occured; more than 10 sprites are being used; two sprites collide; and five bits addition status bits for the 11th sprit· on a line. The CPU control logic 65 provides interrupts to the host microcomputer 30 and receives the write commands, the read commands, and mode commands indicating operation; if writing or reading to the video internal registers 63 or video RAMs 31.

The blank name registers 27 (2) 16 bit registers are used to move data from one section of memory to another section of memory. One register contains the number of bytes to be moved; the other register contains the read memory location. The write memory destination is located in the address register 14.

The color buffers 60 contain 3 bytes of pattern plane color information. Buffer 64 contains the colors which are ready to be loaded onto the color Buss 86. This buffer contains 1 byte of information or (2) 4 bit colors. For graphics modes 1,2,3,4 the LSB nibble of the color byte is loaded onto the color buss if the pattern bit=1 and the MS nibble of the color byte is loaded onto the color buss if the pattern bit=0. For graphics (5), (the bits mapped mode) the LSB nibble is the first color pixel to be displayed and the MSB nibble is the second color pixel to be displayed. Buffers 61 and 62 are temporary storage buffers which will be loaded into buffer 64.

The pattern buffer 84 contains the 1's and 0's which will determine which color in buffer 64 will be displayed. The pattern buffer 84 is loaded into the pattern shift register 86 and shifted out serially. The output of the shift register 86 loads the colors from buffer 64 onto the color buss 86 depending on the color priority logic.

The sprite registers 100 contain the sprite horizontal pointer 82, the sprite pattern register 81, the sprite color register 80, and the sprite coincidence selection logic 70. This is repeated 10 times for 10 sprites per horizontal line. The sprite horizontal counter 82 is loaded with the horizontal sprite position and decrements to the value of zero. Then the sprite pattern register 81 begins shifting bits out serially. 1's load this sprite color onto the color buss 86 and 0's are not used.

The sprite color register 80 contains 4 bits for the sprite color, 1 bit for early clock, and 3 bits to indicate the sprite group.

The sprite coincidence detection logic 70 determines if two or more sprites are shifting 1's out of the sprite pattern register 80 at the same time. If this happens 2 or more sprites have collided on the screen. The sprite groups are decoded from the three bits stored in the 10 sprite color registers 80, and the bits corresponding to the sprite groups are set in the sprite coincidence register 83. If the sprites are in the border area and will not be displayed, the bits will not be set. The three bits in the sprite color register 80 can be decoded into 8 groups, each group corresponds to a bit in the sprite coincidence register 83.

Figure 4, the coincidence detector of Figures 1 and 2 is useful in the application of the invention to video games; for example a space game in which a space ship 110 which is defined as sprite 1 belonging to group 1, and a plurality of rocket ships which are defined as sprites 2, 3 and 4, all assigned to group 2, a flying saucer 11 which is sprite 8 of group 4 and a plurality of meteors 115, 116 and 117 all are sprites belonging to group 3 are used to implement the game. If one of the rocket ships 112 a, b or c which are in group 2 collide with one another, a coincidence will be deteced and bit 2 of the sprite coincidence register 53 will be set . If the spaceship 110 collides with one of the missiles 112, a coincidence will be detected, and bits 1 and 2 of the sprite coincidence register 83 will be set. The host CPU 30 can check to see if the spaceship 10 has collided with another object by reading the sprite coincidence register 83 and checking bit 1.

Figure 5 demonstrates multicolor sprites. Sprites can have a different color on each horizontal line. Sprite (1) which contains the hat, eyes, nose, and mouth is only one sprite, even though there are four different colors. Sprite (2) is the face of the sprite and has to be drawn as a separate sprite since it is on the same horizontal lines as the eyes, nose, and mouth. When sprite 1 and sprite 2 are combined together the sprite 129 is created.

The sprite coincidence detector, 70 of Figure 2, will provide to the sprite register 83 when one of the groups

Figure 7 combines the necessary processing on a single chip that allows both grahics and alphanumeric data (video-text) to be generated. In Figure 7, two way communication is provided in a video text example over standard lines 237 using a modem 235, a data access arrangement 234, and a UART 233. The host CPU 30 has additional interface to a ROM memory 231 and a RAM memory 232, as well as operator interface by a keyboard 236. The Advance Video Data Processor is connected to four RAM's that represent the video RAM 31, and includes an A RAM, B RAM, C RAM, and D RAM as illustrated in Figure 7. The use of the four RAMs which in the preferred embodiment are TMS44116s manufactured by Texas Instruments, provides the memory necessary for the video data storage. The video data is sequenced out by the advance video display processor 1 and then encoded by the video encoder 37 to dot data for each horizontal scan line. The information can then be viewed on the TV set 35. The video display processor 1 provides all the video information and synchronization required to refresh and display the images on the TV set 35.

In Figure 8, to which reference should now be made there is shown the Direct Memory Access (DMA) via a DMA controller 103 and a DMA pin 101 which allows the host microcomputer 30 to directly access the video RAM 31. This pin goes to a logic '1' when there is no CPU access

Figure 17 is a schematic diagram of the implementation of the Advance Video Processor 1 with field effect transistors technology.

Thus, although the best modes contemplated for carrying out the present invention have been herein shown and described, it will be apparent that modification and variation may be made without departing from what is regarded as the subject matter of the invention.

CLAIMS:

1. A combination of a video processor means with color palette for processing of data, a memory means for storing of data to be processed, monitor means for displaying of the processed data, a processor means for controlling the transfer of data from the memory means to the monitor means via the video processor means and wherein the video processor comprising:

a processor control logic means for controlling the transfer of data and instructions between the processor means and the video processor means;

a memory control logic means for controlling, in response to the processor means, the transfer of data from the memory means;

sprite processor means for converting data to color data and to provide the color data for images displayed by the monitor means;

register means for arranging data from the memory means in a predetermined order and to apply the data according to the prearranged order [and] to the sprite processor means;

video output means for converting the color data to a video signal to which the monitor means will respond to provide a display of a predetermined pattern and of a predetermined color.

2. The combination according to Claim 1 further comprising:

sound generation means for generating of sound for applications to the monitor means.

3. The combination according to Claim 1 or 2 wherein the sprite processor means comprises:

color palette means for storing a plurality of color fields; and

fetch means, responsive to program instruction from the memory means, for fetch members of the color field.

4. A method of processing of video data with a video processor means, a memory means for storing of data to be processed, monitor means for displaying of the processed data, a processor means for controlling the transfer of data from the memory means to the monitor means via the video processor means and wherein the method comprises the step of:

controlling the transfer of data and instructions between the processor means and the video processor means;

controlling, in response to the processor means, the transfer of data from the memory means;

storing data from a color palette;

converting data to a color signal with a sprite processor means;

arranging data from the memory means in a predetermined order, and applying the data according to the prearranged order to the sprite processor means; and

converting the color data to or monitor signal to which the video means will respond and to provide a display of a predetermined pattern of a predetermined color.

5. The method according to Claim 4 further comprising the steps of:

generating of sound for applications to the monitor means..

6. The method according to Claim 4 or 5 wherein the steps of converting data to a color signal comprises the steps of:

storing a plurality of color fields; and

fetching members of the color field in response to a program instruction from the memory means.

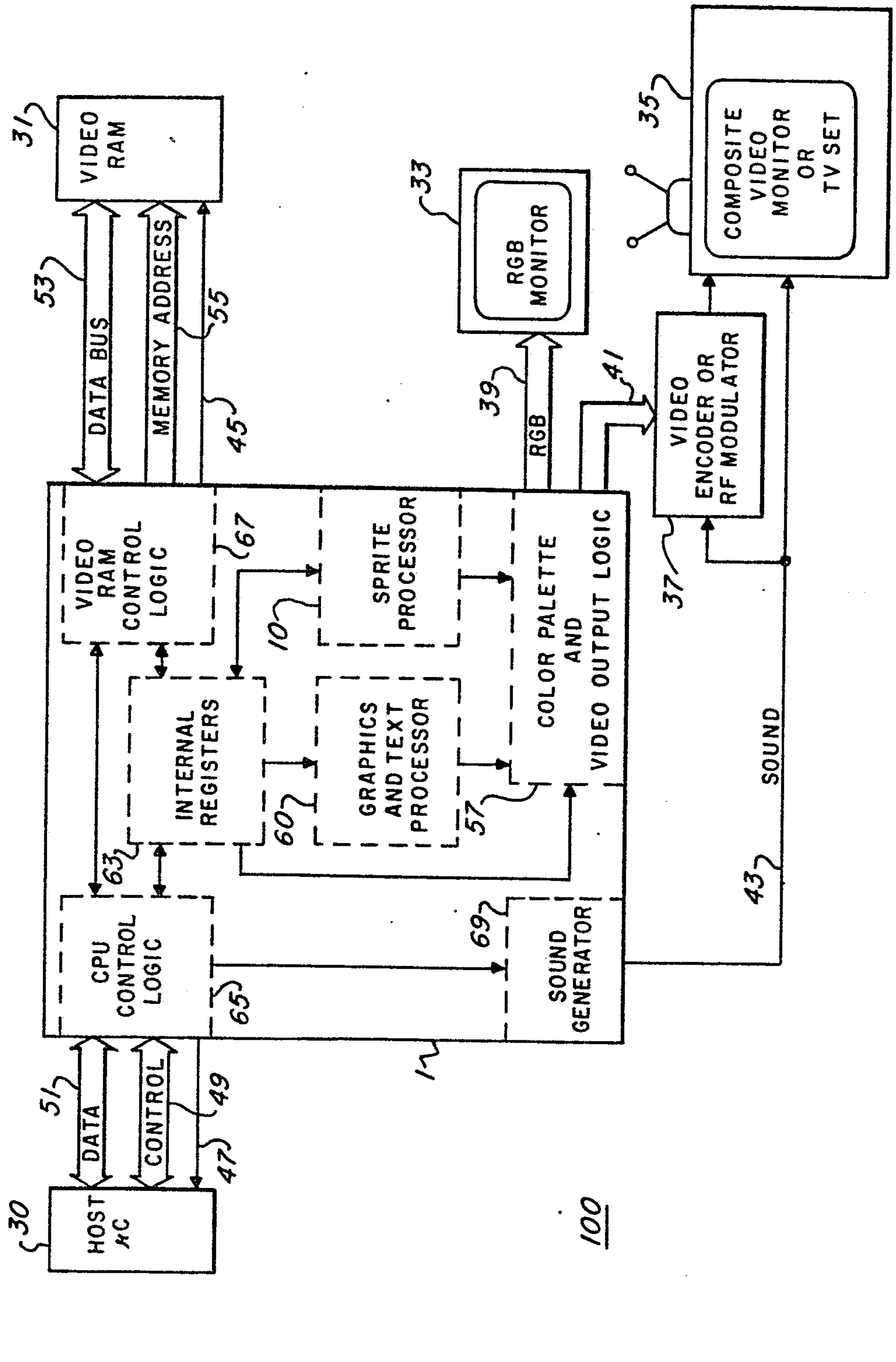

VIDEO RAM
31
DATA BUS
53
MEMORY ADDRESS
55
45
RGB MONITOR
33
COMPOSITE VIDEO MONITOR OR TV SET
35
RGB
39
41
VIDEO ENCODER OR RF MODULATOR
37
VIDEO RAM CONTROL LOGIC
67
SPRITE PROCESSOR
10
COLOR PALETTE AND VIDEO OUTPUT LOGIC
INTERNAL REGISTERS
GRAPHICS AND TEXT PROCESSOR
60
57
63
SOUND
43
CPU CONTROL LOGIC
65
69
SOUND GENERATOR
1
DATA
51
CONTROL
49
47
HOST μC
30
100

Fig. 1

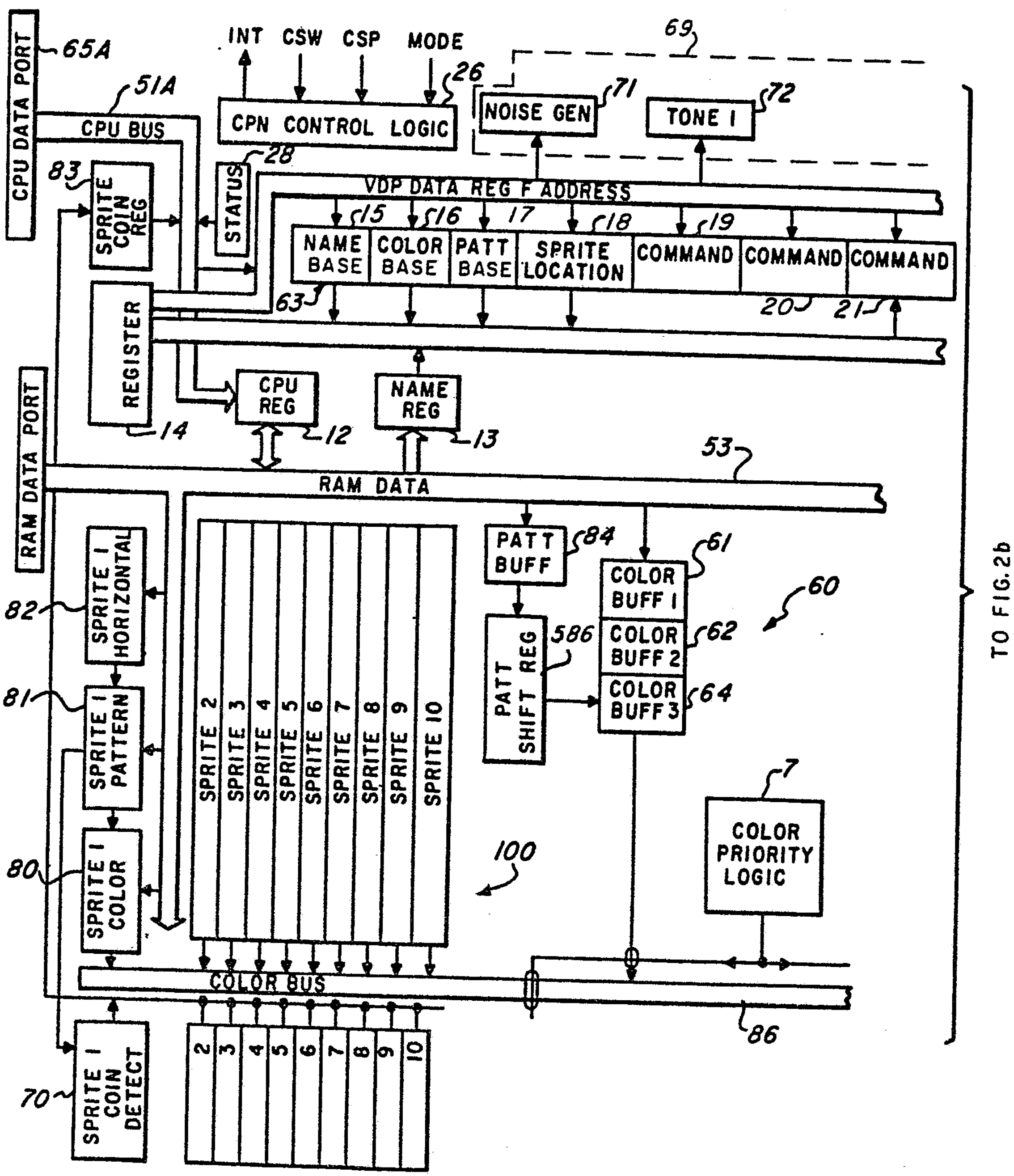

CPU DATA PORT
65A
51A
CPU BUS
INT CSW CSP MODE
CPN CONTROL LOGIC
26
69
NOISE GEN
71
TONE I
72
83
SPRITE COIN REG
STATUS
28
VDP DATA REG F ADDRESS
15 16 17 18 19
NAME BASE
COLOR BASE
PATT BASE
SPRITE LOCATION
COMMAND
COMMAND
COMMAND
63
20 21
REGISTER
14
CPU REG
12
NAME REG
13
RAM DATA PORT
53
RAM DATA
PATT BUFF
84
61
COLOR BUFF I
COLOR BUFF 2
62
COLOR BUFF 3
64
60
PATT SHIFT REG
586
82
SPRITE I HORIZONTAL
81
SPRITE I PATTERN
80
SPRITE I COLOR
SPRITE 2
SPRITE 3
SPRITE 4
SPRITE 5
SPRITE 6
SPRITE 7
SPRITE 8
SPRITE 9
SPRITE 10
100
7
COLOR PRIORITY LOGIC
COLOR BUS
86
70
SPRITE I COIN DETECT
2 3 4 5 6 7 8 9 10
TO FIG.2b

Fig. 2a

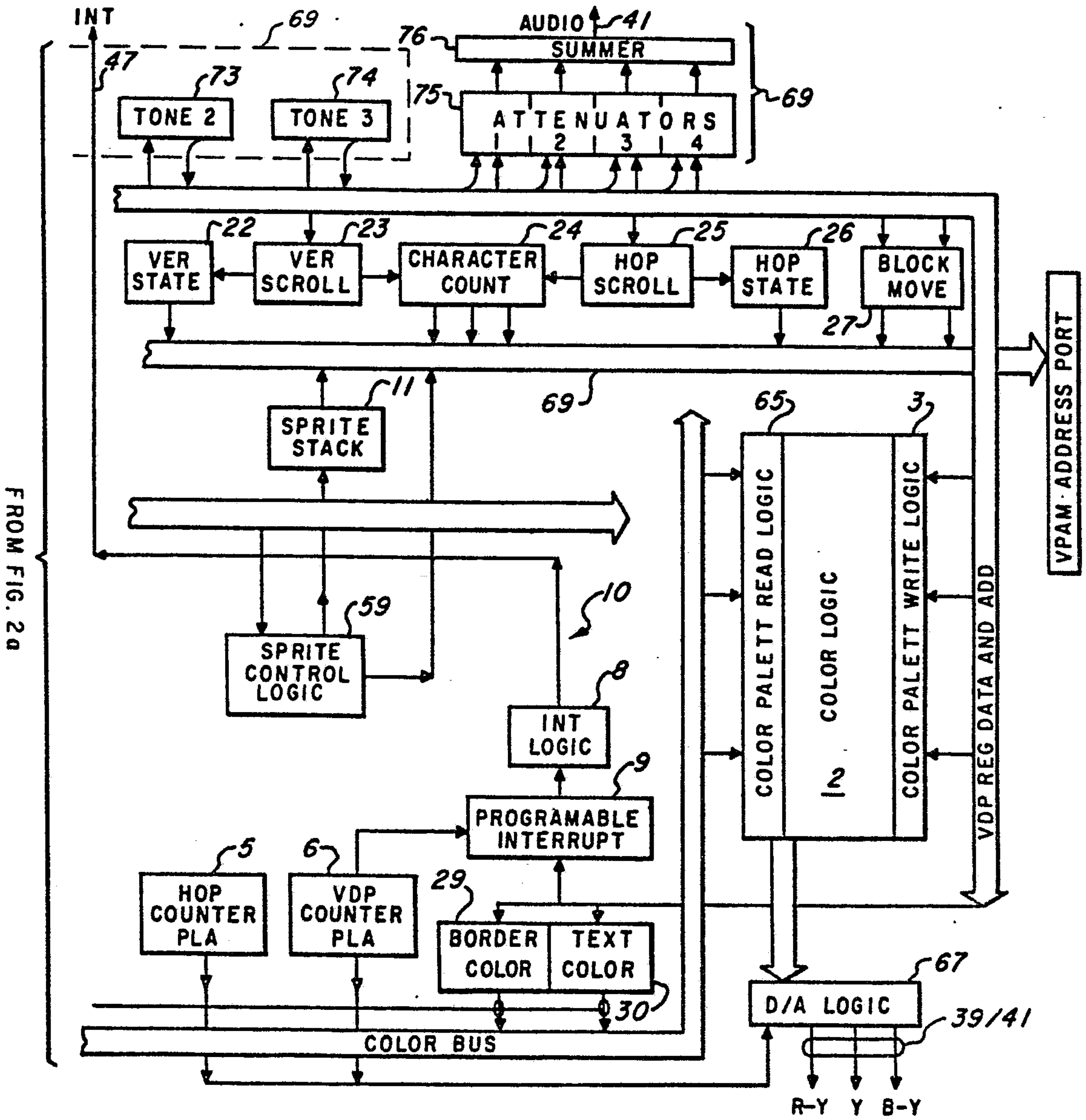

INT
69
AUDIO
41
76
SUMMER
47
73
74
TONE 2
TONE 3
75
ATTENUATORS
1 2 3 4
69
22
23
24
25
26
VER STATE
VER SCROLL
CHARACTER COUNT
HOP SCROLL
HOP STATE
BLOCK MOVE
27
VPAM-ADDRESS PORT
69
65
3
11
SPRITE STACK
COLOR PALETT READ LOGIC
COLOR LOGIC
COLOR PALETT WRITE LOGIC
2
FROM FIG. 2a
VDP REG DATA AND ADD
10
59
SPRITE CONTROL LOGIC
8
INT LOGIC
9
PROGRAMABLE INTERRUPT
5
6
29
HOP COUNTER PLA
VDP COUNTER PLA
BORDER COLOR
TEXT COLOR
67
D/A LOGIC
30
39/41
COLOR BUS
R-Y Y B-Y

Fig. 2b

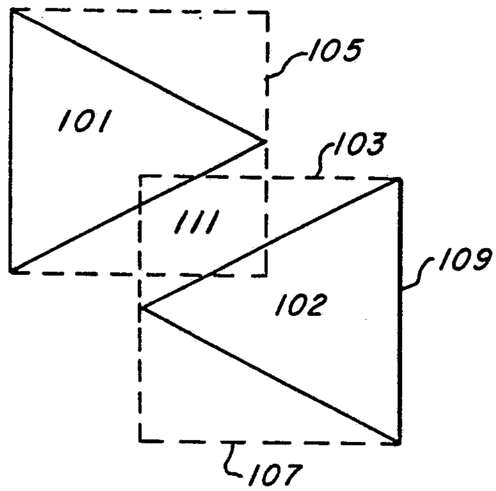



Fig.3

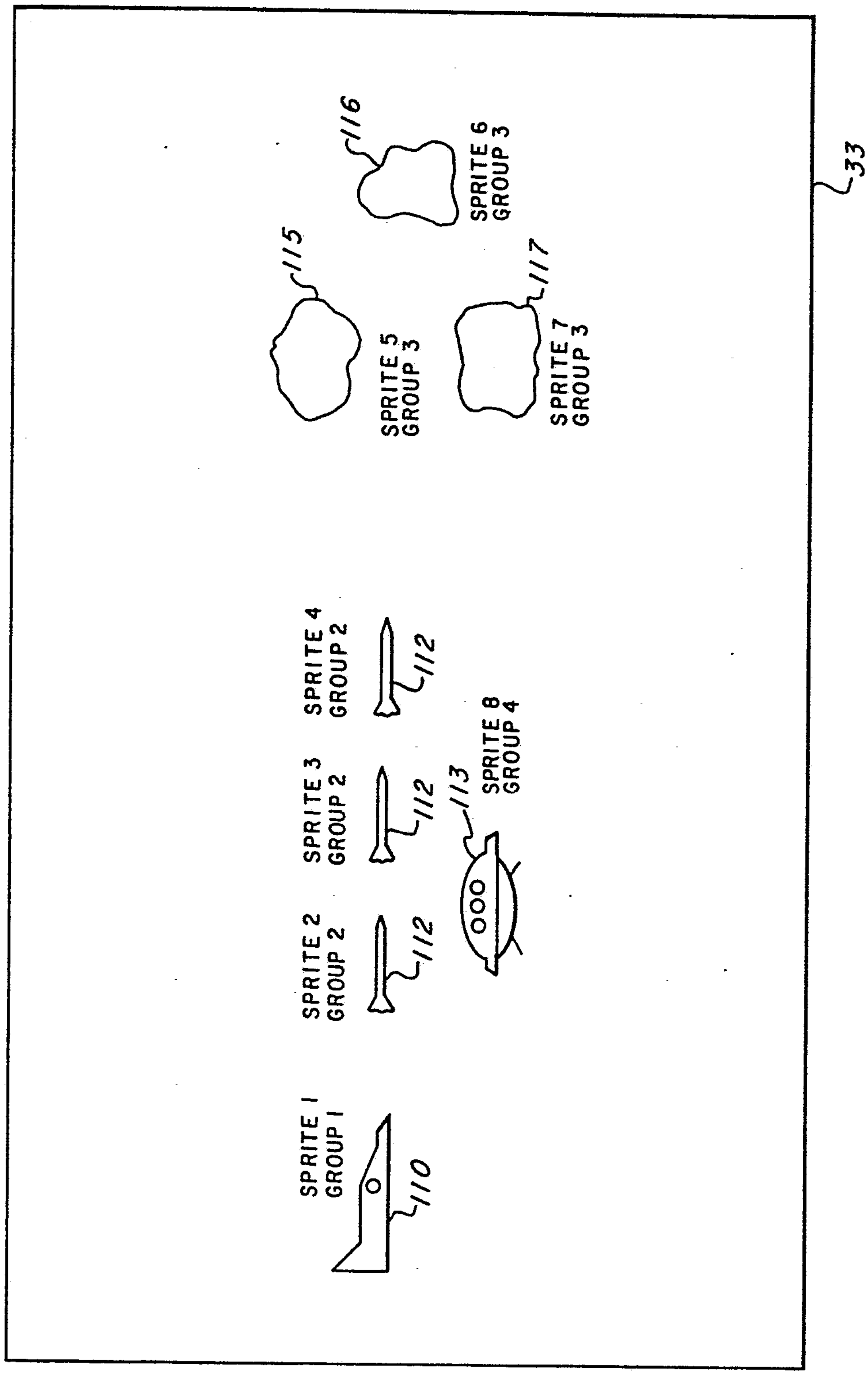

33
SPRITE 1
GROUP 1
110
SPRITE 2
GROUP 2
112
SPRITE 3
GROUP 2
112
SPRITE 4
GROUP 2
112
113
SPRITE 8
GROUP 4
115
SPRITE 5
GROUP 3
116
SPRITE 6
GROUP 3
117
SPRITE 7
GROUP 3

Fig. 4

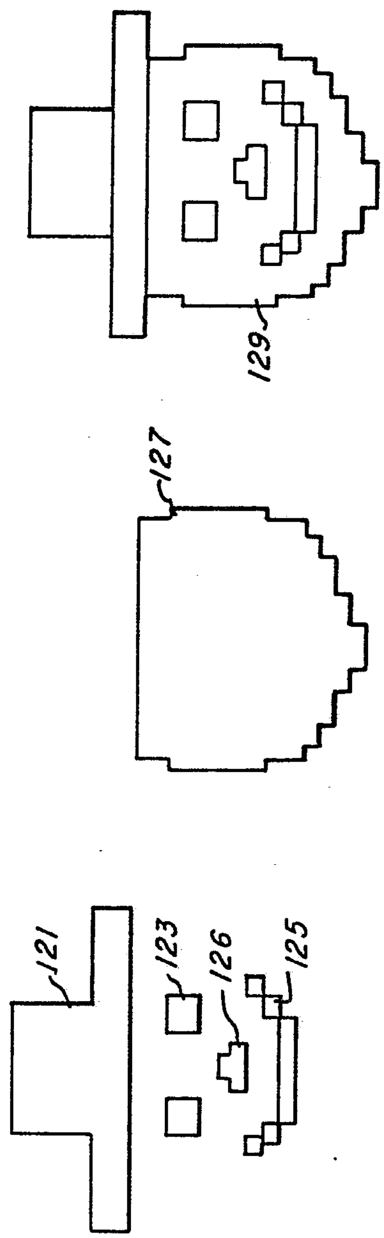



Fig.5

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| SPRITE | | GROUPS | | | COLLIDE | | |

Fig.6

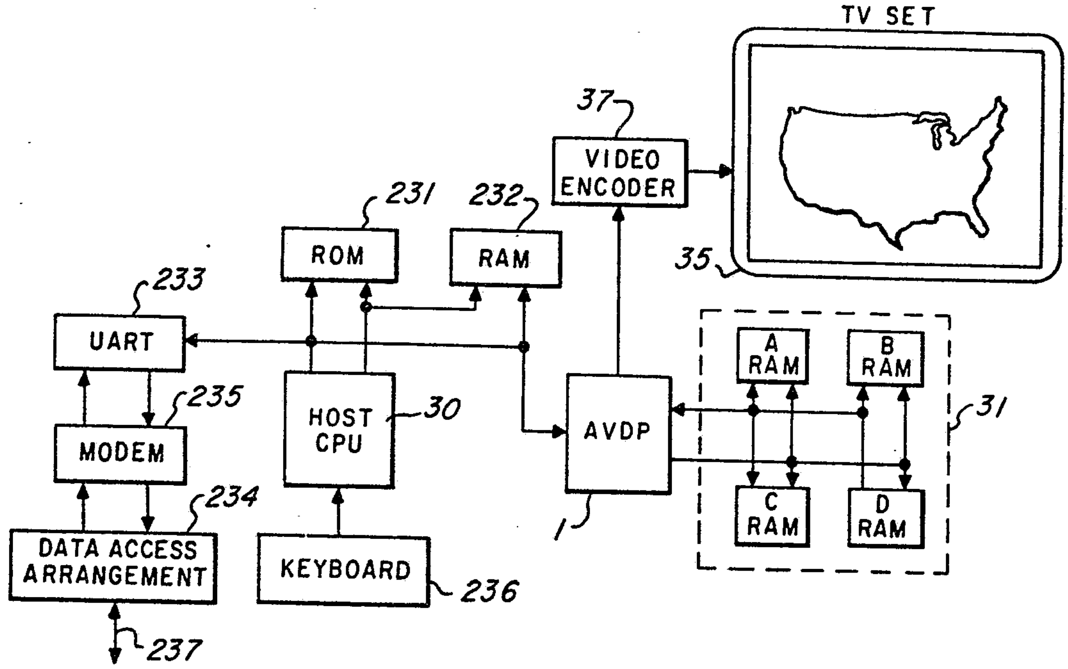

TV SET
37
VIDEO ENCODER
231
232
ROM
RAM
35
233
UART
A RAM
B RAM
235
HOST CPU
30
AVDP
31
MODEM
234
C RAM
D RAM
1
DATA ACCESS ARRANGEMENT
KEYBOARD
236
237

Fig. 7

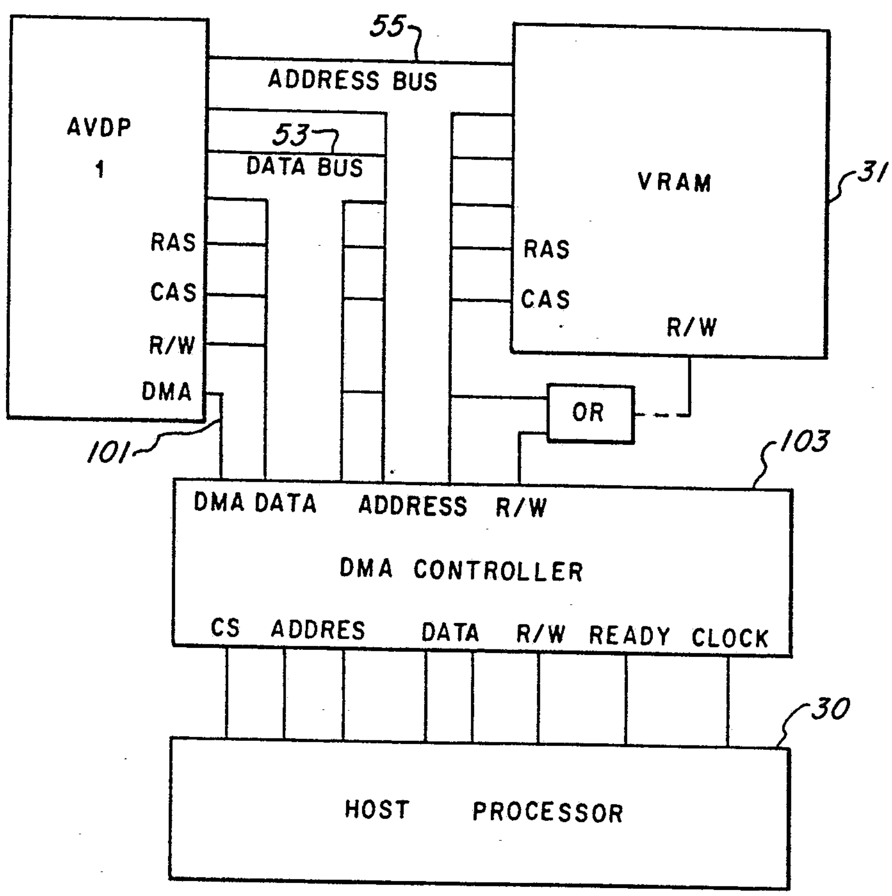

55
ADDRESS BUS
AVDP
1
53
DATA BUS
VRAM
31
RAS
CAS
R/W
DMA
RAS
CAS
R/W
OR
101
103
DMA DATA
ADDRESS
R/W
DMA CONTROLLER
CS
ADDRES
DATA
R/W
READY
CLOCK
30
HOST PROCESSOR

Fig. 8

| OPERATION | CPU/AVDP DATA BUS 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | $\overline{CSW}$ | $\overline{CSR}$ | MODE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| WRITE TO AVDP REGISTER | | | | | | | | | | | |
| BYTE 1 DATA TRANSFER | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 0 | 1 | 1 |
| BYTE 2 REGISTER SELECT | 1 | 0 | RS0 | RS1 | RS2 | RS3 | RS4 | RS5** | 0 | 1 | 1 |
| READ STATUS REGISTER | | | | | | | | | | | |
| BYTE 1 DATA READ | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 1 | 0 | 1 |
| READ SPRITE COLLISION REGISTER | | | | | | | | | | | |
| BYTE 1 NO OPERATION | × | × | × | × | × | × | × | × | 0 | 1 | 1 |
| BYTE 2 REGISTER SELECT | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| BYTE 3 DATA READ | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 1 | 0 | 0 |
| WRITE TO VIDEO RAM | | | | | | | | | | | |
| BYTE 1 LOW ADDRESS SET | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | 0 | 1 | 1 |
| BYTE 2 HIGH ADDRESS SET | 0 | 1 | A0 | A1 | A2 | A3 | A4 | A5 | 0 | 1 | 1 |
| BYTE 3 DATA WRITE | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 0 | 1 | 0 |
| READ FROM VIDEO RAM | | | | | | | | | | | |
| BYTE 1 LOW ADDRESS SET | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | 0 | 1 | 1 |
| BYTE 2 HIGH ADDRESS SET | 0 | 1 | A0 | A1 | A2 | A3 | A4 | A5 | 0 | 1 | 1 |
| BYTE 3 DATA WRITE | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 1 | 0 | 0 |
| WRITE TO SOUND GENERATOR | | | | | | | | | | | |
| BYTE 1 DATA WRITE | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | 0 | 0 | 1 |

*Fig. 9*

| REGISTER NUMBER | DESCRIPTION |
|---|---|
| 0 | STATUS REGISTER |
| 1 | SPRITE COLLISION REGISTER |

*Fig. 10*

| REGISTER NUMBER | DESCRIPTION |
|---|---|
| | 19 |
| 0 | CONFIGURATION REGISTER 20 |
| 1 | CONFIGURATION REGISTER 15 |
| 2 | BASE ADDRESS OF NAME TABLE SUB-BLOCK 16 |
| 3 | BASE ADDRESS OF COLOR TABLE SUB-BLOCK (8 LSBs) |
| 4 | BASE ADDRESS OF COLOR TABLE SUB-BLOCK (2 MSBs) & BASE ADDRESS OF PATTERN OR TEXT GENERATOR SUB-BLOCK |
| 5 | BASE ADDRESS OF SPRITE ATTRIBUTE TABLE SUB-BLOCK (8 LSBs) |

17

*Fig. 11a*

| | |
|---|---|
| 6 | BASE ADDRESS OF SPRITE ATTRIBUTE TABLE SUB-BLOCK (MSB)<br>&<br>BASE ADDRESS OF SPRITE PATTERN GENERATOR SUB-BLOCK |
| 7 | COLOR CODE IN TEXT MODE BACKDROP COLOR IN ALL MODES 30 |
| 8 | HORIZONTAL SCROLLING REGISTER 25 |
| 9 | VERTICAL SCROLLING REGISTER 23 |
| 10 | CONFIGURATION REGISTER 21 |
| 11 | LSB BLOCK MOVE REGISTER 27 |
| 12 | MSB BLOCK MOVE REGISTER 29 |
| 13 | PROGRAMMABLE INTERRUPT REGISTER 9 |
| 14 | LSB FOR READ ADDRESS ON BLOCK MOVE |
| 15 | MSB FOR READ ADDRESS ON BLOCK MOVE |
| 16 | CPU ADDRESS PAGE REGISTER |
| 32 | COLOR PALETTE COLOR 0 RRR0GGG0 |
| 33 | COLOR PALETTE COLOR 0 0000BBB0 |
| 34 | COLOR PALLETTE COLOR 1 RRR0GGG0 |
| 35 | COLOR PALLETTE COLOR 1 0000BBB0 |
| 36 | COLOR PALLETTE COLOR 2 RRR0GGG0 |
| 37 | COLOR PALLETTE COLOR 3 0000BBB0 |

13

2

Fig. 11b

| | |
|---|---|
| 38 | COLOR PALETTE COLOR 3 RRR0GGG0 |
| 39 | COLOR PALETTE COLOR 3 0000BBB0 |
| 40 | COLOR PALETTE COLOR 4 RRR0GGG0 |
| 41 | COLOR PALETTE COLOR 4 0000BBB0 |
| 42 | COLOR PALETTE COLOR 5 RRR0GGG0 |
| 43 | COLOR PALETTE COLOR 5 0000BBB0 |
| 44 | COLOR PALETTE COLOR 6 RRR0GGG0 |
| 45 | COLOR PALETTE COLOR 6 0000BBB0 |
| 46 | COLOR PALETTE COLOR 7 RRR0GGG0 |
| 47 | COLOR PALETTE COLOR 7 0000BBB0 |
| 48 | COLOR PALETTE COLOR 8 RRR0GGG0 |
| 49 | COLOR PALETTE COLOR 8 0000BBB0 |
| 50 | COLOR PALETTE COLOR 9 RRR0GGG0 |
| 51 | COLOR PALETTE COLOR 9 0000BBB0 |
| 52 | COLOR PALETTE COLOR 10 RRR0GGG0 |
| 53 | COLOR PALETTE COLOR 10 0000BBB0 |
| 54 | COLOR PALETTE COLOR 11 RRR0GGG0 |
| 55 | COLOR PALETTE COLOR 11 0000BBB0 |

2

Fig. 11c

| | |
|---|---|
| 56 | COLOR PALETTE COLOR 12 RRR0GGG0 |
| 57 | COLOR PALETTE COLOR 12 0000BBB0 |
| 58 | COLOR PALETTE COLOR 13 RRR0GGG0 |
| 59 | COLOR PALETTE COLOR 13 0000BBB0 |
| 60 | COLOR PALETTE COLOR 14 RRR0GGG0 |
| 61 | COLOR PALETTE COLOR 14 0000BBB0 |
| 62 | COLOR PALETTE COLOR 15 RRR0GGG0 |
| 63 | COLOR PALETTE COLOR 15 0000BBB0 |

Fig. 11d

MSB D0 … LSB D7

| | | | |
|---|---|---|---|
| F | C | 11S | 11 sprite number |

Fig. 12

COLOR DIFFERENCE

R32,34,36,38,40 42,44,46,48,50 52,54,56;58,60 62

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| R-Y | R-Y | R-Y | 0 | Y | Y | Y | 0 |

R33,35,37,39,41 43,45,47,49,51 53,55,57,59,61 63

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | B-Y | B-Y | B-Y | 0 |

R-Y Y B-Y

*Fig. 13*

RED-GREEN-BLUE

R32,34,36,38,40 42,44,46,48,50 52,54,56;58,60 62

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| R | R | R | 0 | G | G | G | 0 |

R33,35,37,39,41 43,45,47,49,51 53,55,57,59,61 63

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | B | B | B | 0 |

R = RED G = GREEN B = BLUE

*Fig. 14*

| | DEFAULT VALUE AT RESET | | | COLOR NAME |
|---|---|---|---|---|
| COLOR 0 | 0 0 X | 1 1 X | 0 X X | TRANSPARENT |
| COLOR 1 | 0 1 1 | 0 0 0 | 1 0 0 | BLACK |
| COLOR 2 | 0 0 0 | 1 0 1 | 0 1 0 | MEDIUM GREEN |
| COLOR 3 | 0 0 0 | 1 0 0 | 0 1 0 | LIGHT GREEN |
| COLOR 4 | 0 1 1 | 0 1 1 | 1 1 0 | DARK BLUE |
| COLOR 5 | 0 1 1 | 1 0 0 | 1 1 0 | LIGHT BLUE |
| COLOR 6 | 1 0 1 | 0 1 1 | 0 1 1 | DARK RED |
| COLOR 7 | 0 0 1 | 1 0 1 | 1 0 1 | CYAN |
| COLOR 8 | 1 1 1 | 1 0 0 | 0 1 0 | MEDIUM RED |
| COLOR 9 | 1 0 1 | 1 0 1 | 0 1 1 | LIGHT RED |
| COLOR A | 1 0 0 | 1 0 1 | 0 0 0 | DARK YELLOW |
| COLOR B | 1 0 0 | 1 1 0 | 0 0 0 | LIGHT YELLOW |
| COLOR C | 0 0 0 | 0 1 1 | 0 1 0 | DARK GREEN |
| COLOR D | 1 1 1 | 0 1 1 | 1 1 1 | MAGENTA |
| COLOR E | 0 1 1 | 1 0 1 | 1 0 0 | GRAY |
| COLOR F | 0 1 1 | 1 1 1 | 1 0 0 | WHITE |

Fig. 15

| INTERRUPT IE SELECTED | STATUS BITS AFFECTED: IE | F | PIE |
|---|---|---|---|
| BLOCK MOVE | ENABLES INTERRUPT | SET TO 1 ON BLOCK MOVE COMPLETE | DISABLED |
| VERTICAL RETRACE | ENABLES INTERRUPT | SET TO 1 ON VERTICAL RETRACE | NOT AFFECTED |
| PROGRAMMABLE INTERRUPT | ENABLES INTERRUPT | SET TO 1 ON ON SELECTED HORIZONTAL LINE | ENABLES INTERRUPT |
| BLOCK MOVE & VERTICAL RETRACE | ENABLES INTERRUPT | SET TO 1 ON VERTICAL RETRACE AND BLOCK MOVE COMPLETE | DISABLED DURING BLOCK MOVE |

*Fig. 16*

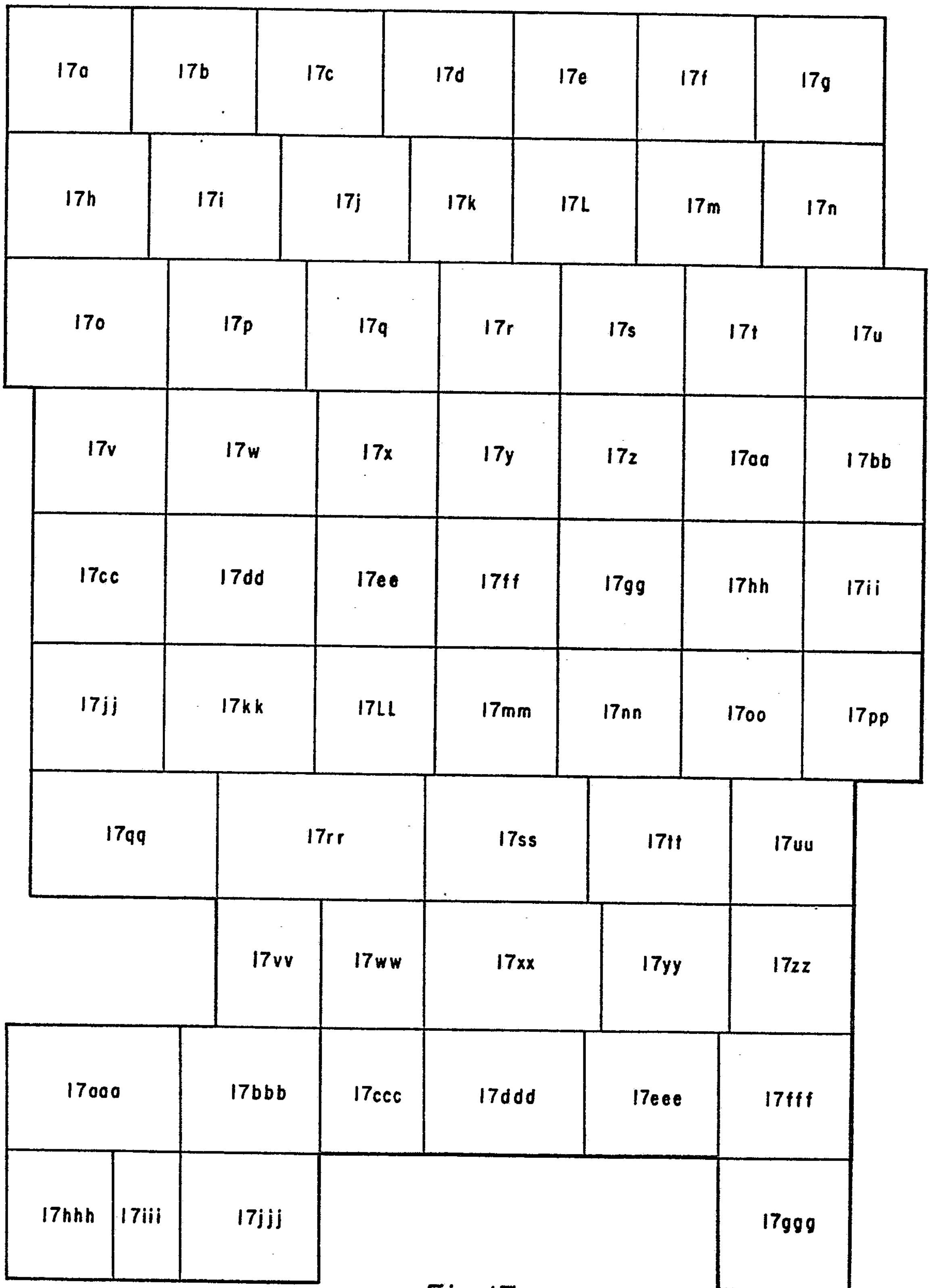

17a
17b
17c
17d
17e
17f
17g
17h
17i
17j
17k
17L
17m
17n
17o
17p
17q
17r
17s
17t
17u
17v
17w
17x
17y
17z
17aa
17bb
17cc
17dd
17ee
17ff
17gg
17hh
17ii
17jj
17kk
17LL
17mm
17nn
17oo
17pp
17qq
17rr
17ss
17tt
17uu
17vv
17ww
17xx
17yy
17zz
17aaa
17bbb
17ccc
17ddd
17eee
17fff
17hhh
17iii
17jjj
17ggg

*Fig. 17*

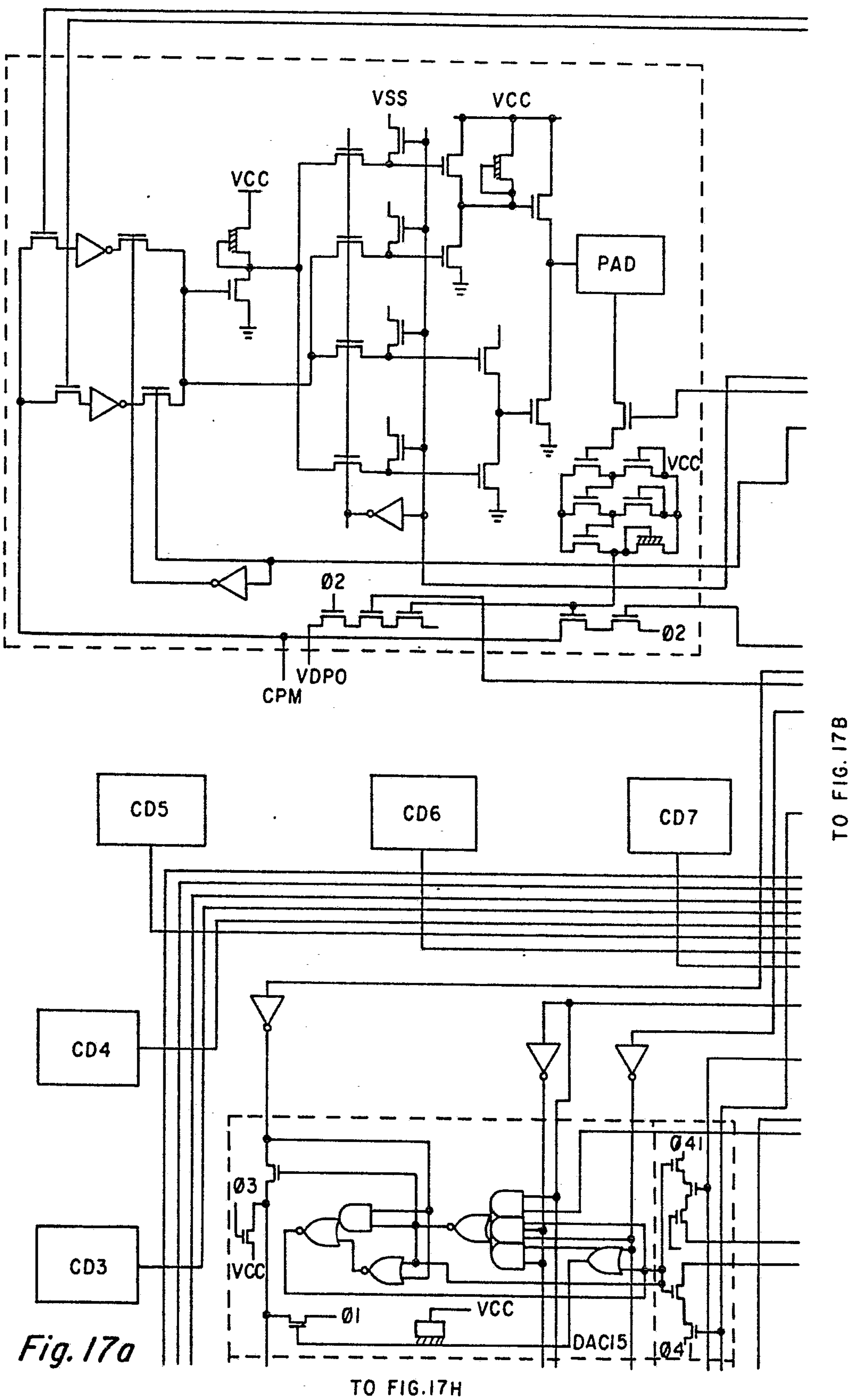

VSS
VCC
VCC
PAD
VCC
Ø2
Ø2
VDPO
CPM
TO FIG.17B
CD5
CD6
CD7
CD4
CD3
Ø41
Ø3
VCC
Ø1
VCC
DAC15
Ø4L
TO FIG.17H

Fig. 17a

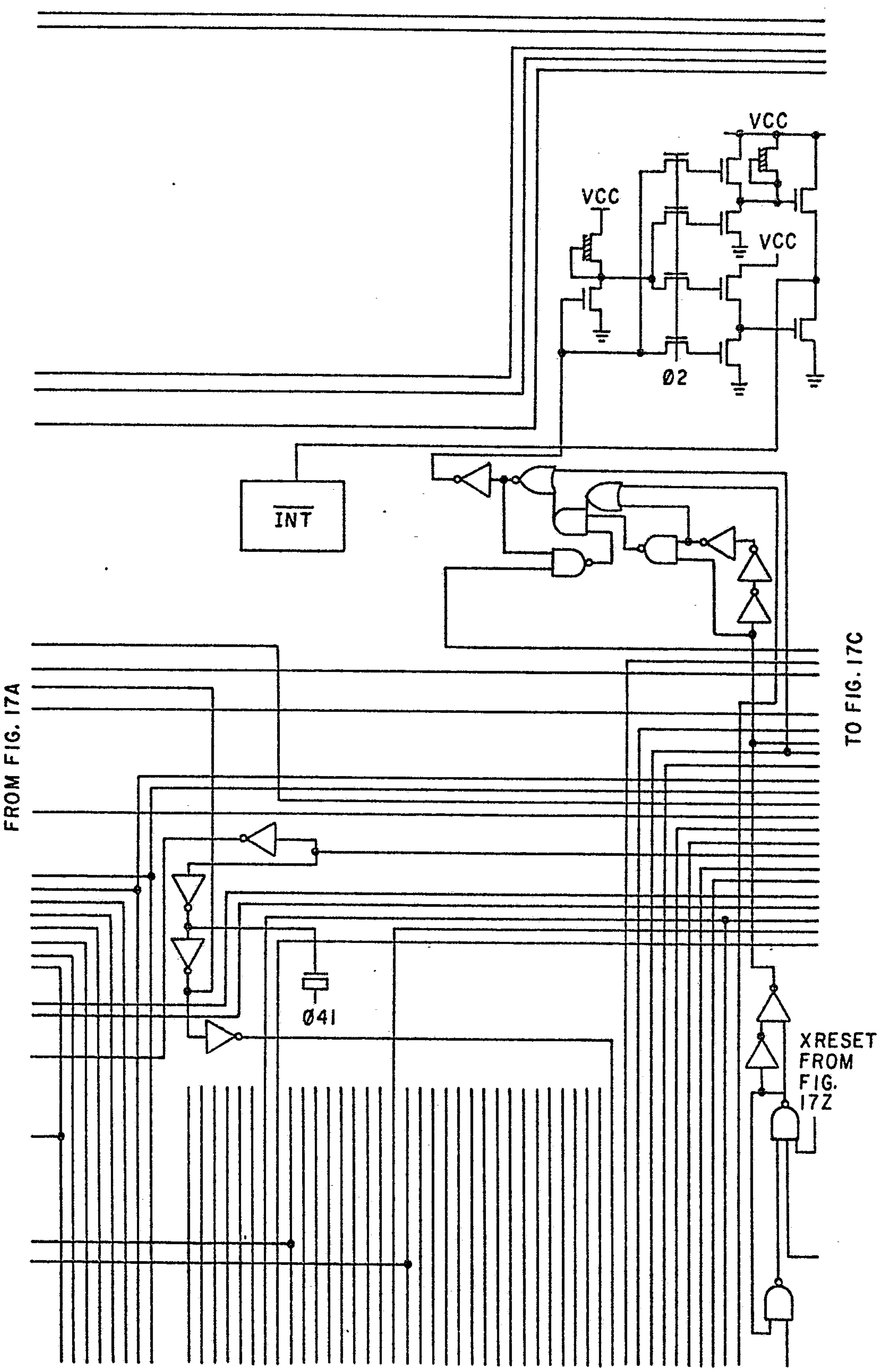



Fig. 17b

TO FIG. 17I

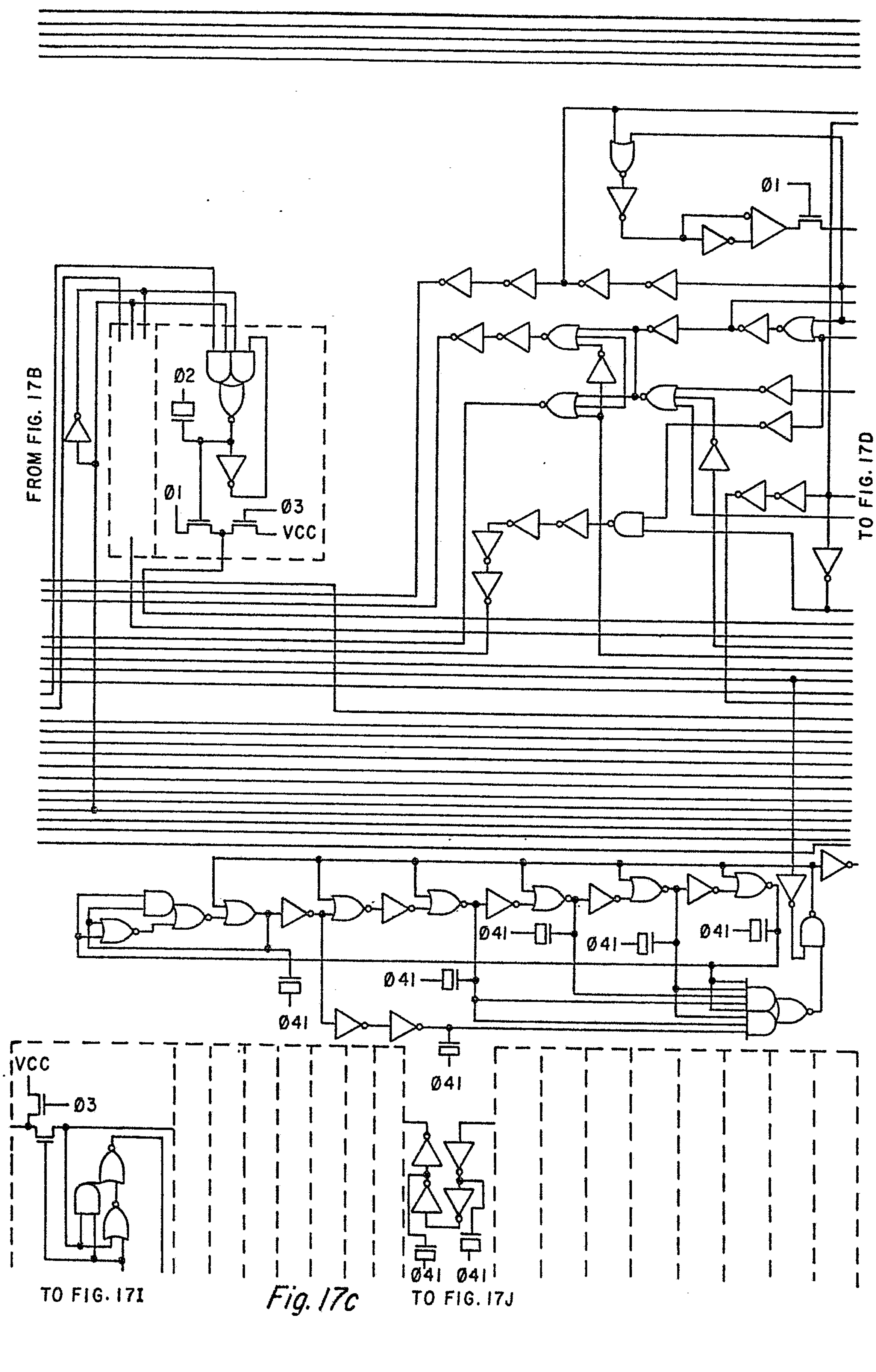



Fig. 17c

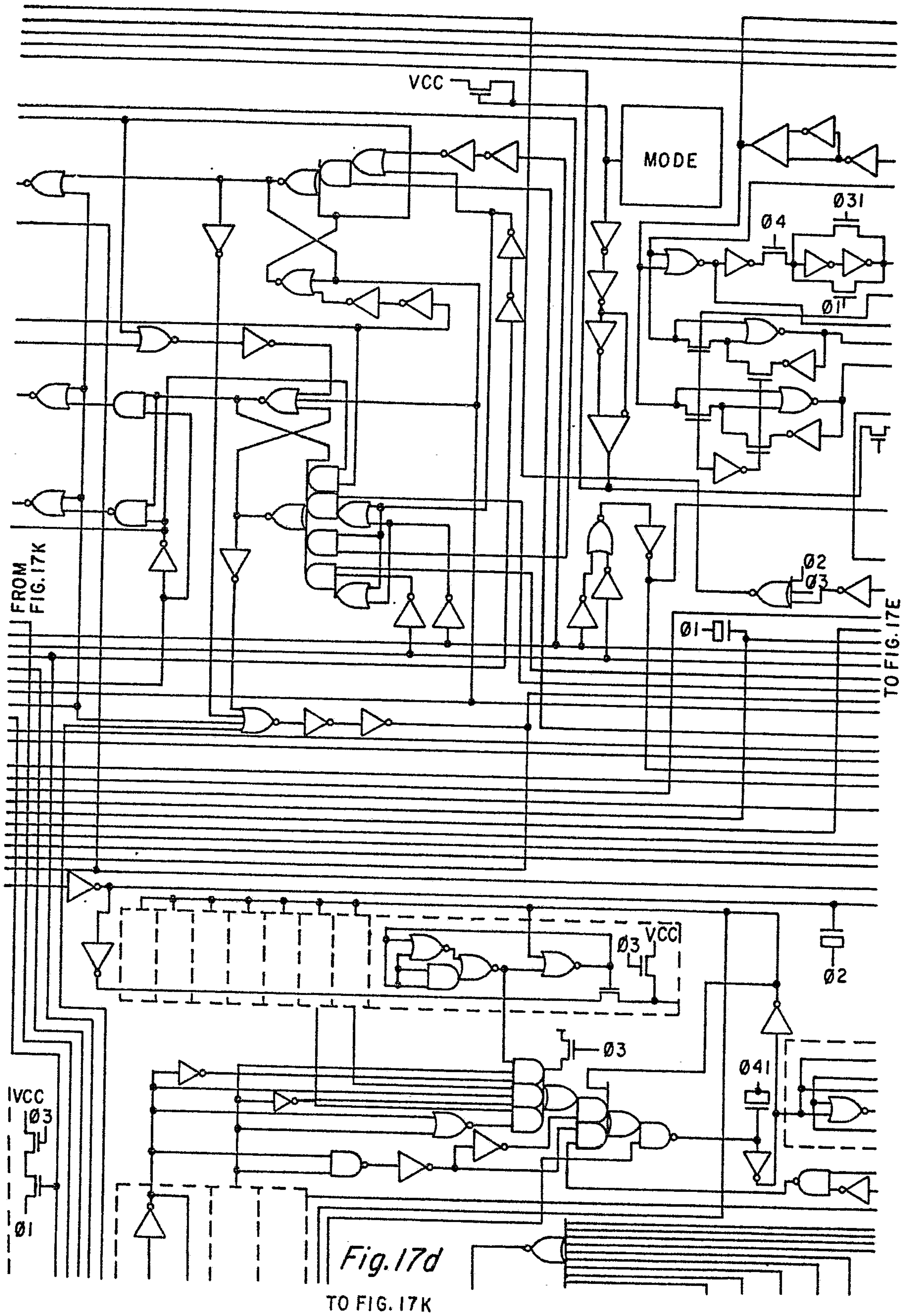



Fig.17d

TO FIG.17K

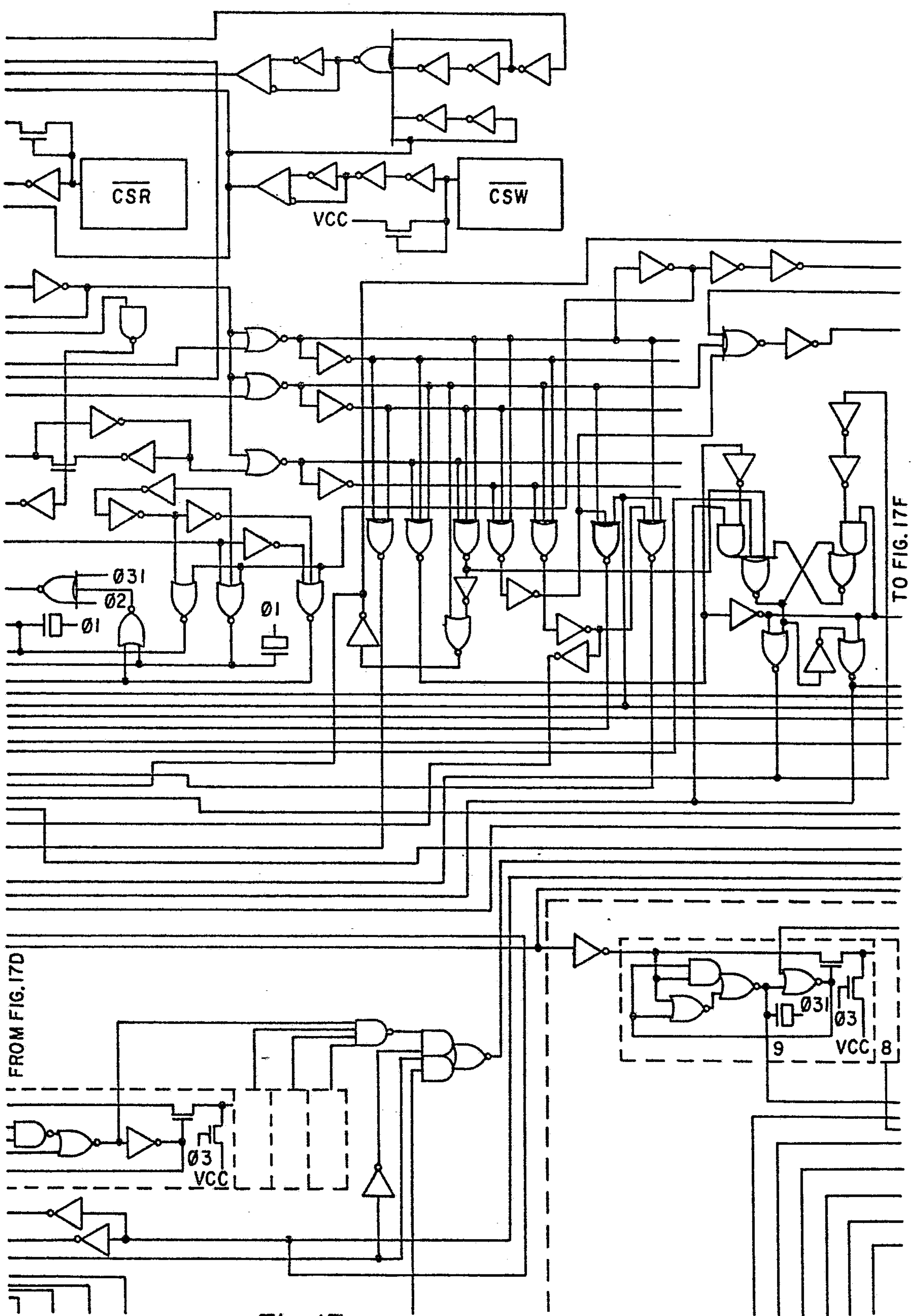
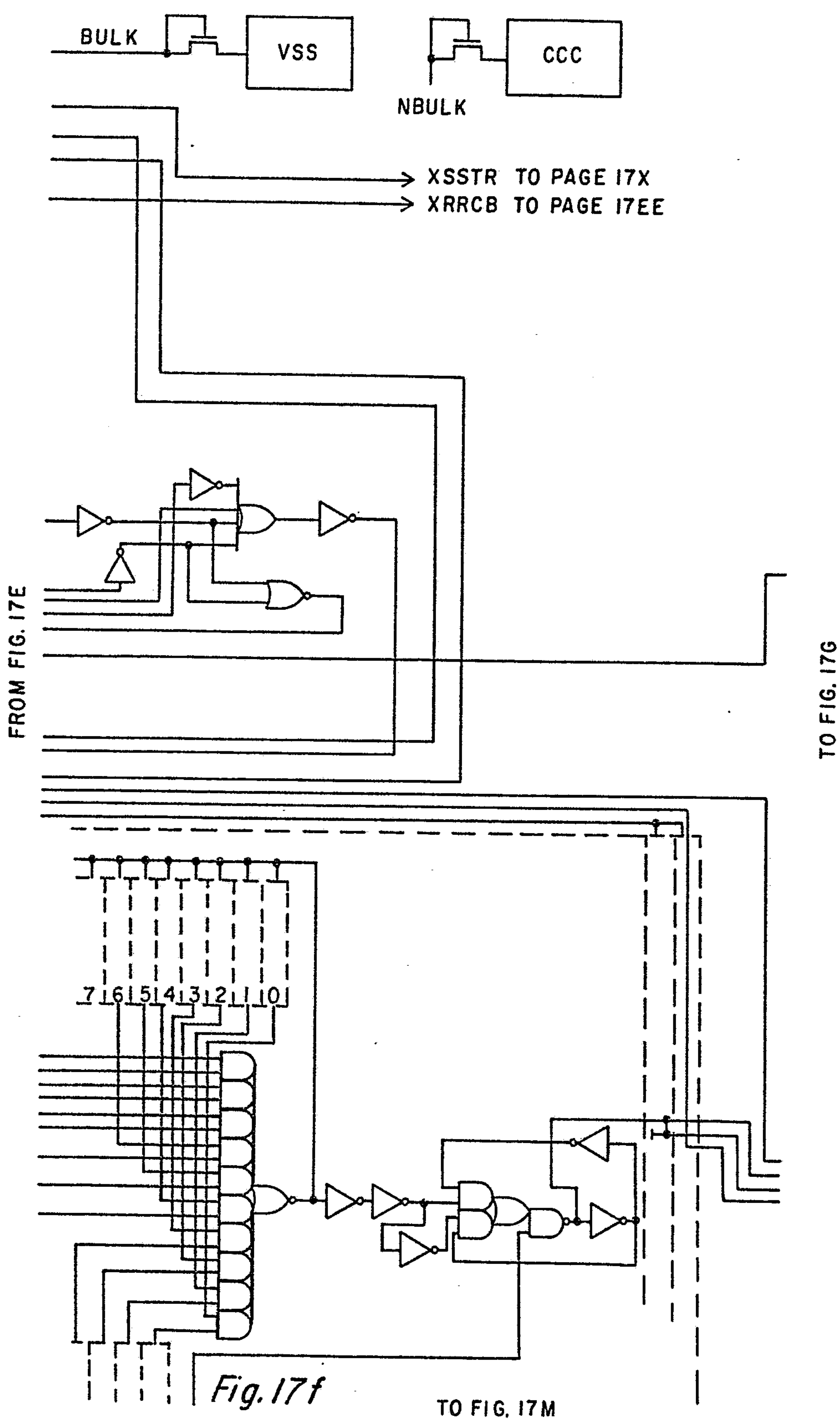


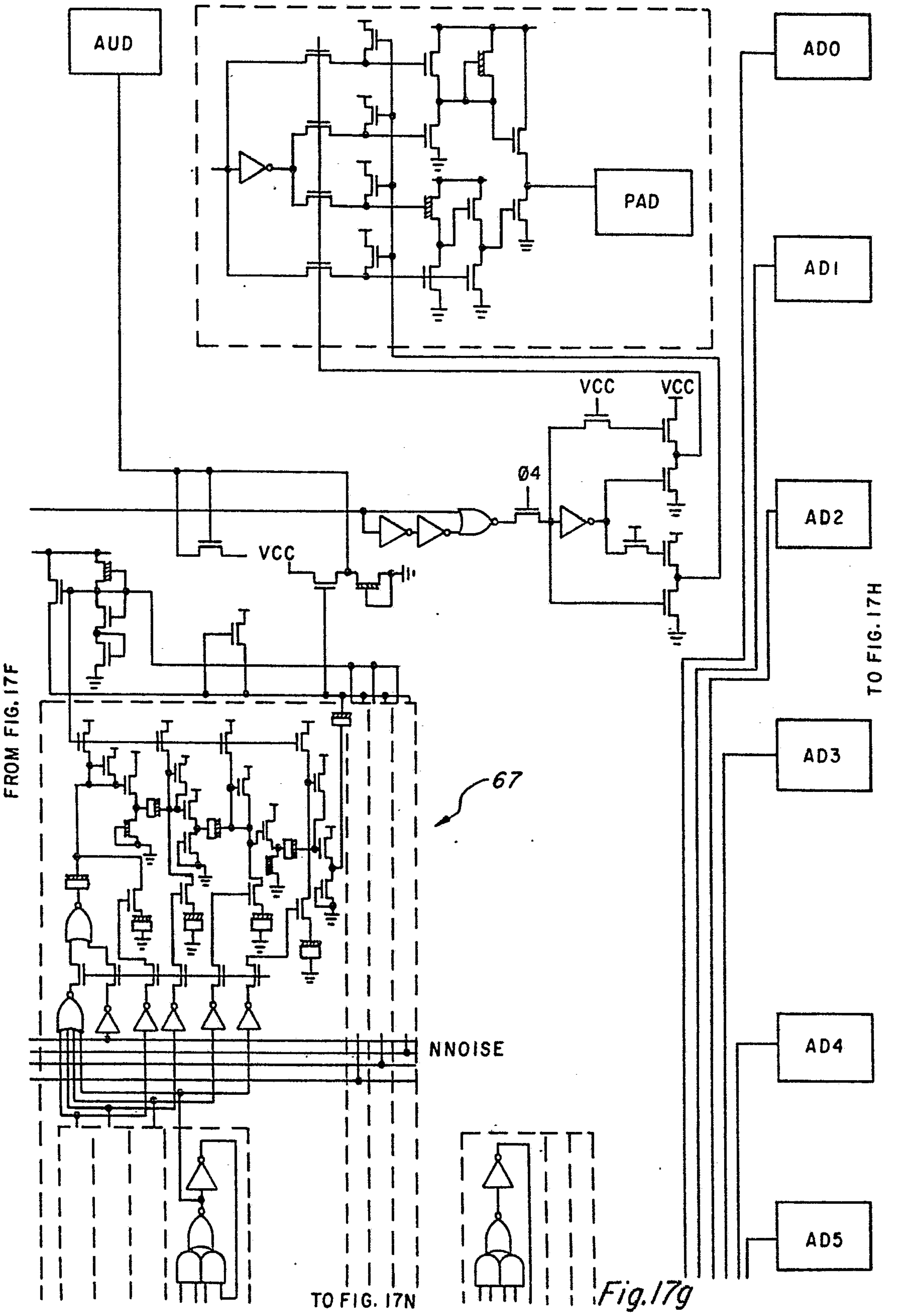

Fig. 17e

TO FIG. 17L



BULK
VSS
CCC
NBULK
XSSTR TO PAGE 17X
XRRCB TO PAGE 17EE
FROM FIG. 17E
TO FIG. 17G
7 6 5 4 3 2 1 0
TO FIG. 17M


Fig. 17f



AUD
PAD
AD0
AD1
AD2
AD3
AD4
AD5
VCC
VCC
VCC
Ø4
FROM FIG. 17F
TO FIG. 17H
TO FIG. 17N
NNOISE
67


Fig.17g

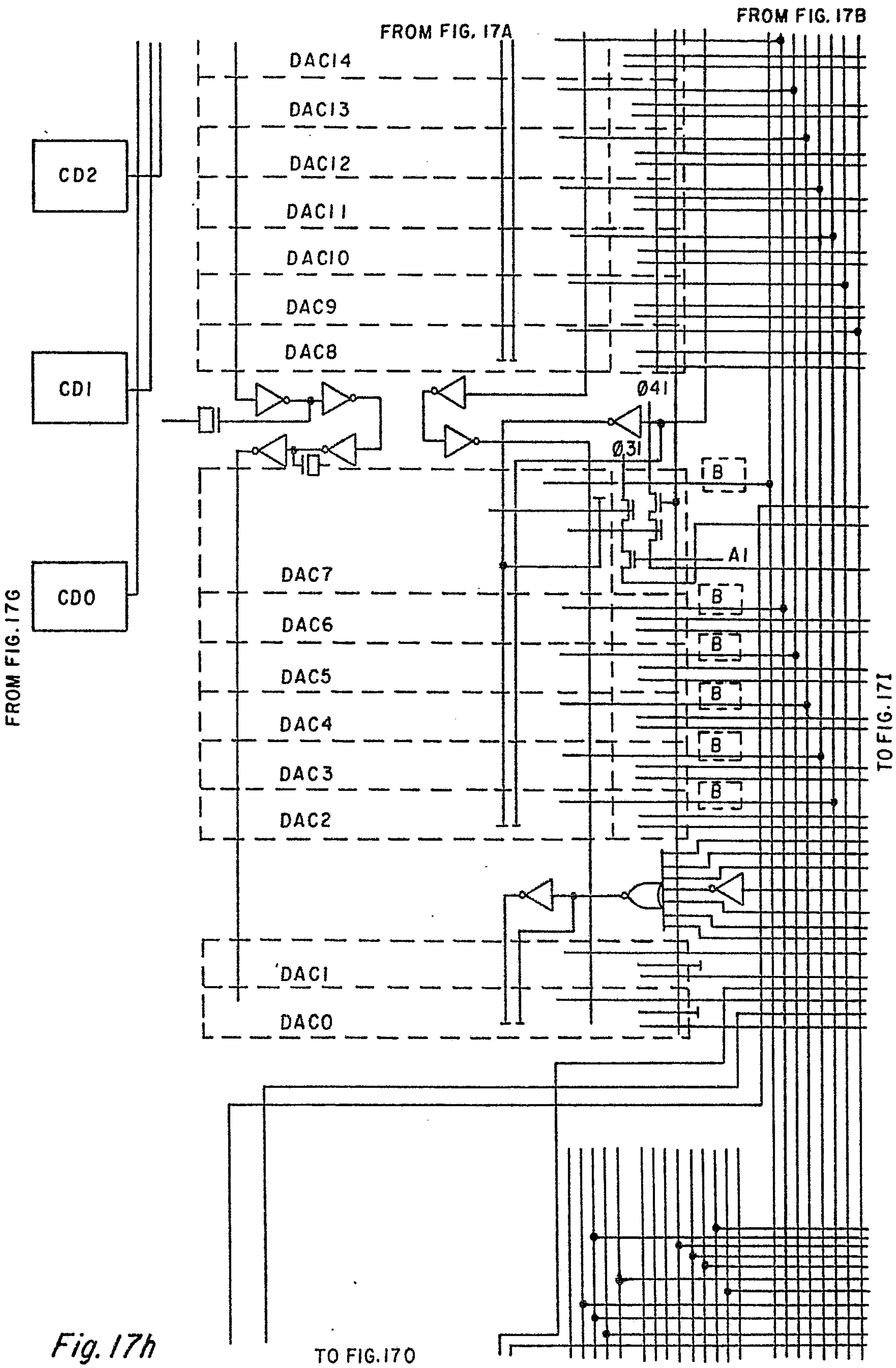

FROM FIG. 17A
FROM FIG. 17B
DAC14
DAC13
DAC12
DAC11
DAC10
DAC9
DAC8
CD2
CD1
CD0
Ø41
Ø31
B
A1
DAC7
DAC6
DAC5
DAC4
DAC3
DAC2
DAC1
DAC0
FROM FIG. 17G
TO FIG. 17I
TO FIG. 17O

Fig. 17h

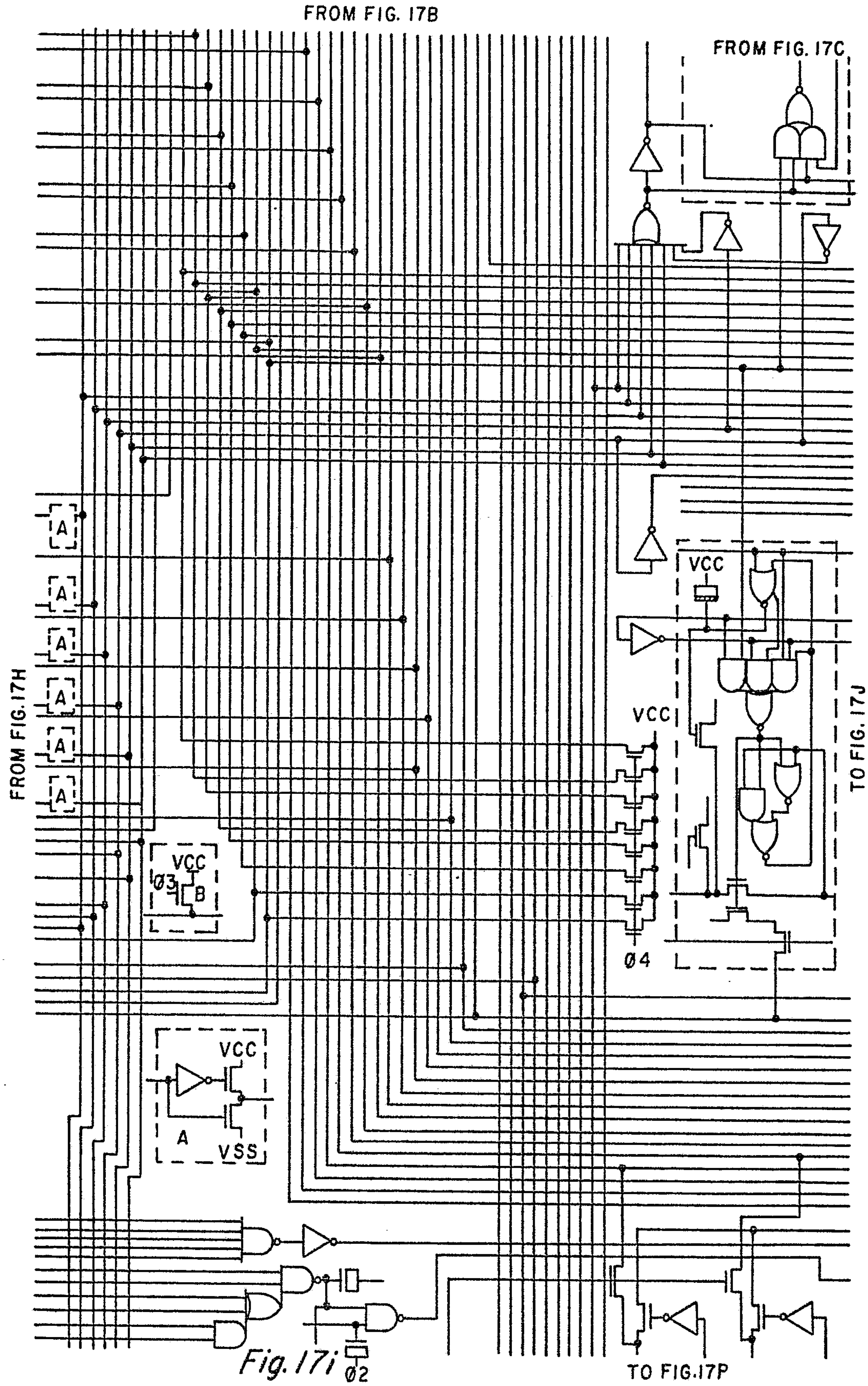

FROM FIG. 17B
FROM FIG. 17C
FROM FIG. 17H
TO FIG. 17J
VCC
A
B
Ø3
Ø4
VSS
Ø2
TO FIG.17P

Fig. 17i

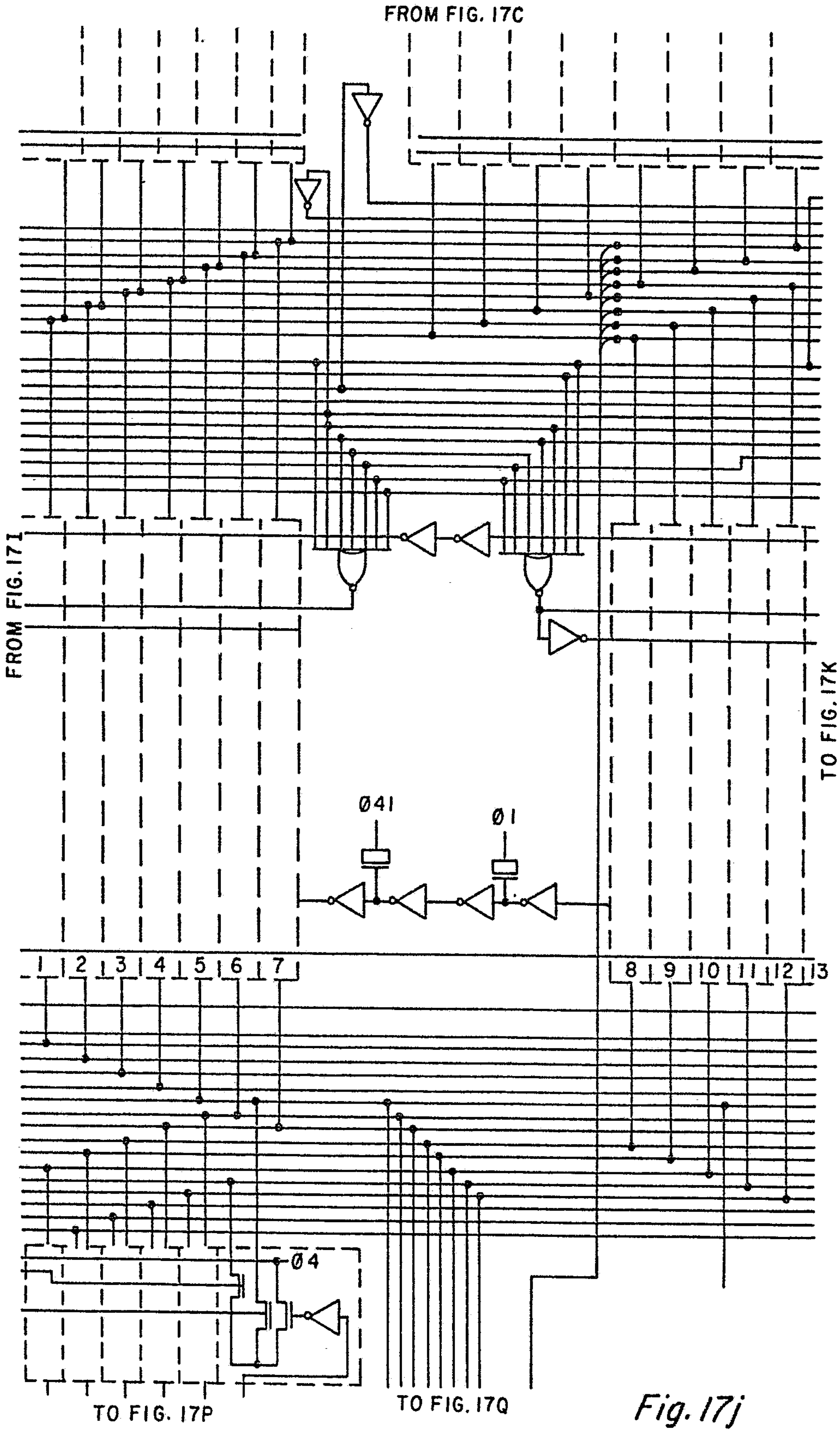
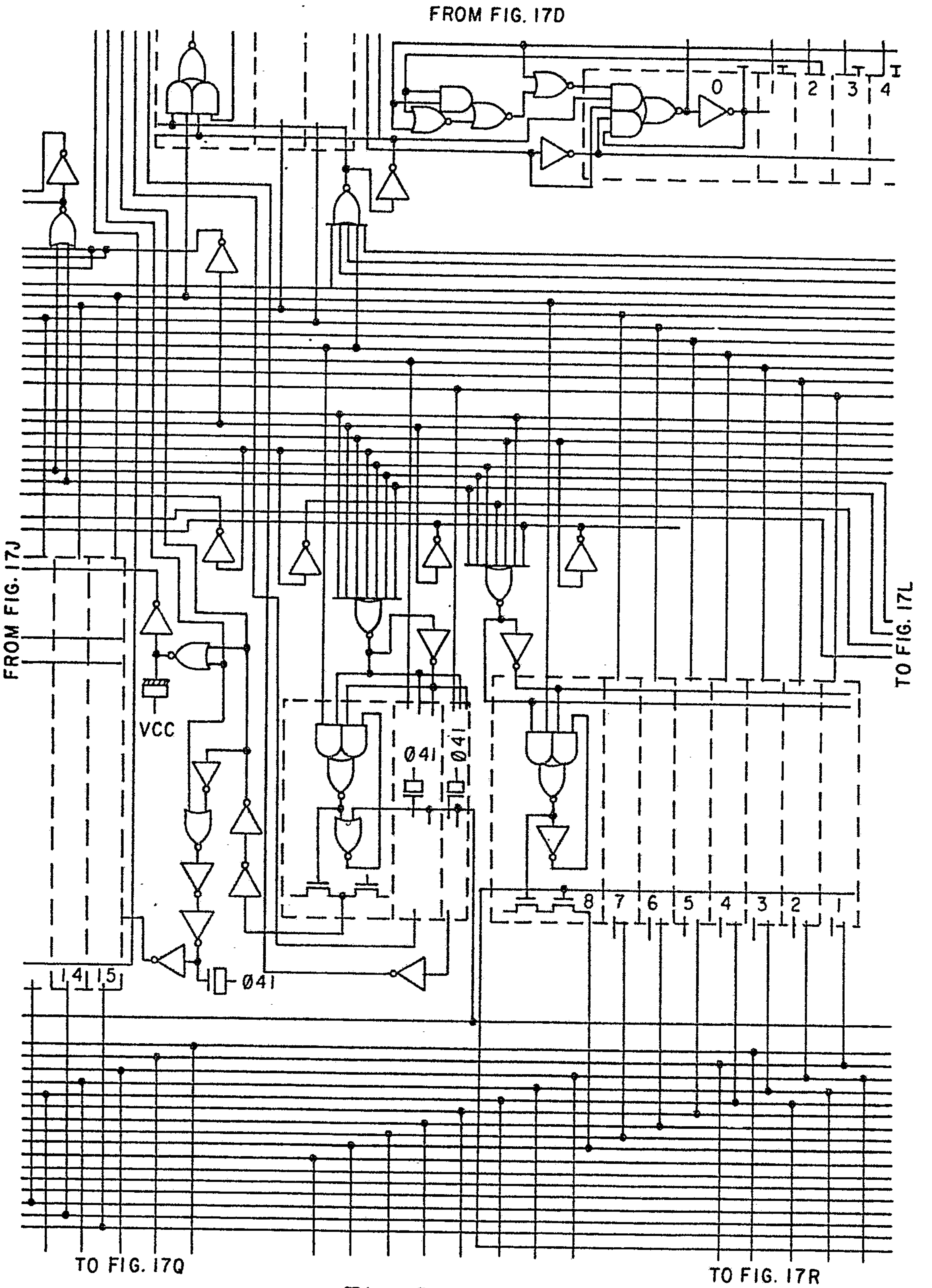
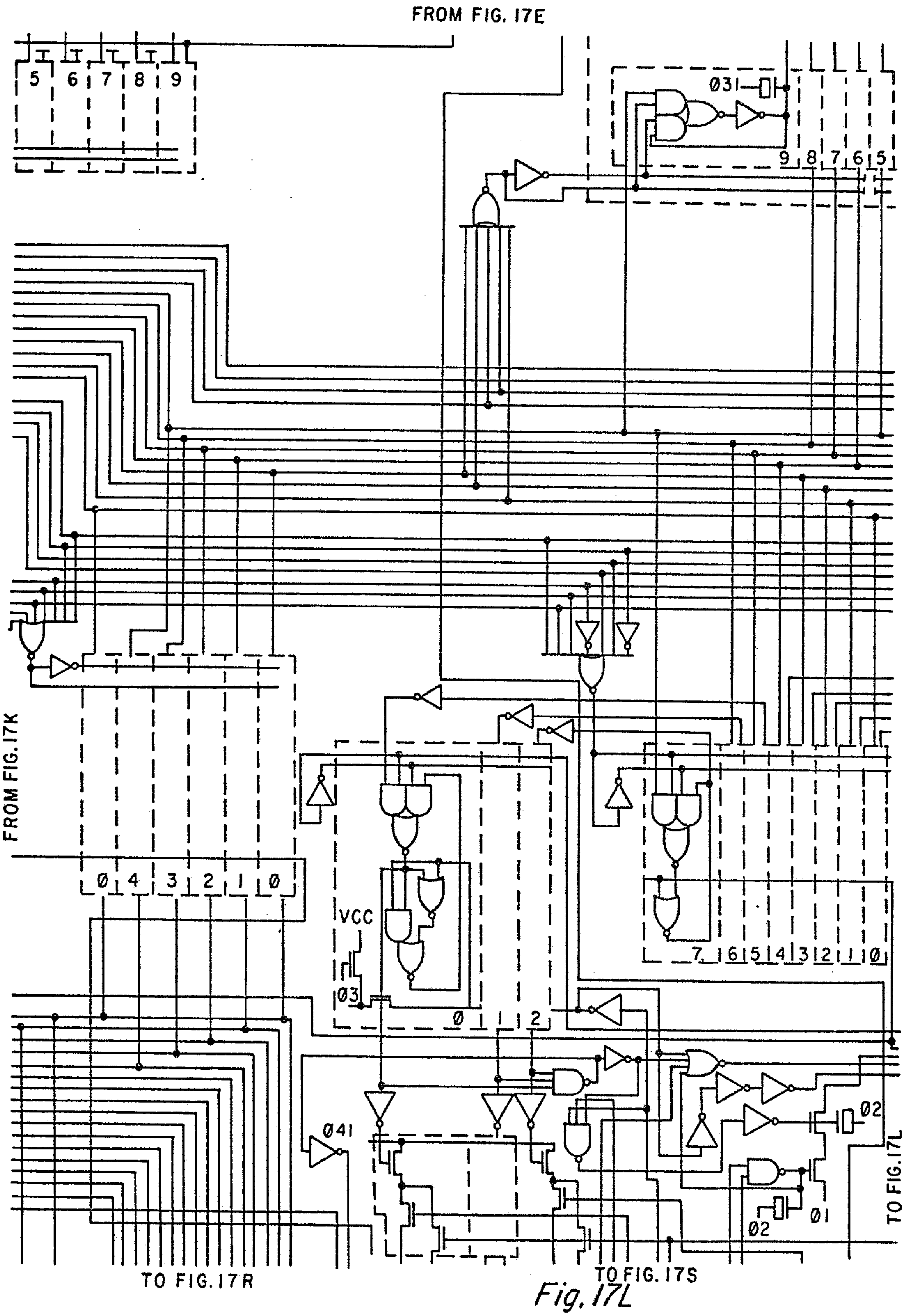

FROM FIG. 17C
FROM FIG. 17I
TO FIG. 17K
Ø41
Ø1
1
2
3
4
5
6
7
8
9
10
11
12
13
Ø4
TO FIG. 17P
TO FIG. 17Q

Fig. 17j



FROM FIG. 17D
0
1
2
3
4
FROM FIG. 17J
TO FIG. 17L
VCC
Ø41
Ø41
Ø41
8
7
6
5
4
3
2
1
14
15
TO FIG. 17Q
TO FIG. 17R


Fig. 17k



FROM FIG. 17E
5
6
7
8
9
Ø31
9
8
7
6
5
FROM FIG.17K
Ø
4
3
2
1
Ø
VCC
Ø3
Ø
1
2
7
6
5
4
3
2
1
Ø
Ø41
Ø2
Ø2
Ø1
TO FIG.17L
TO FIG.17R
TO FIG.17S


Fig. 17L

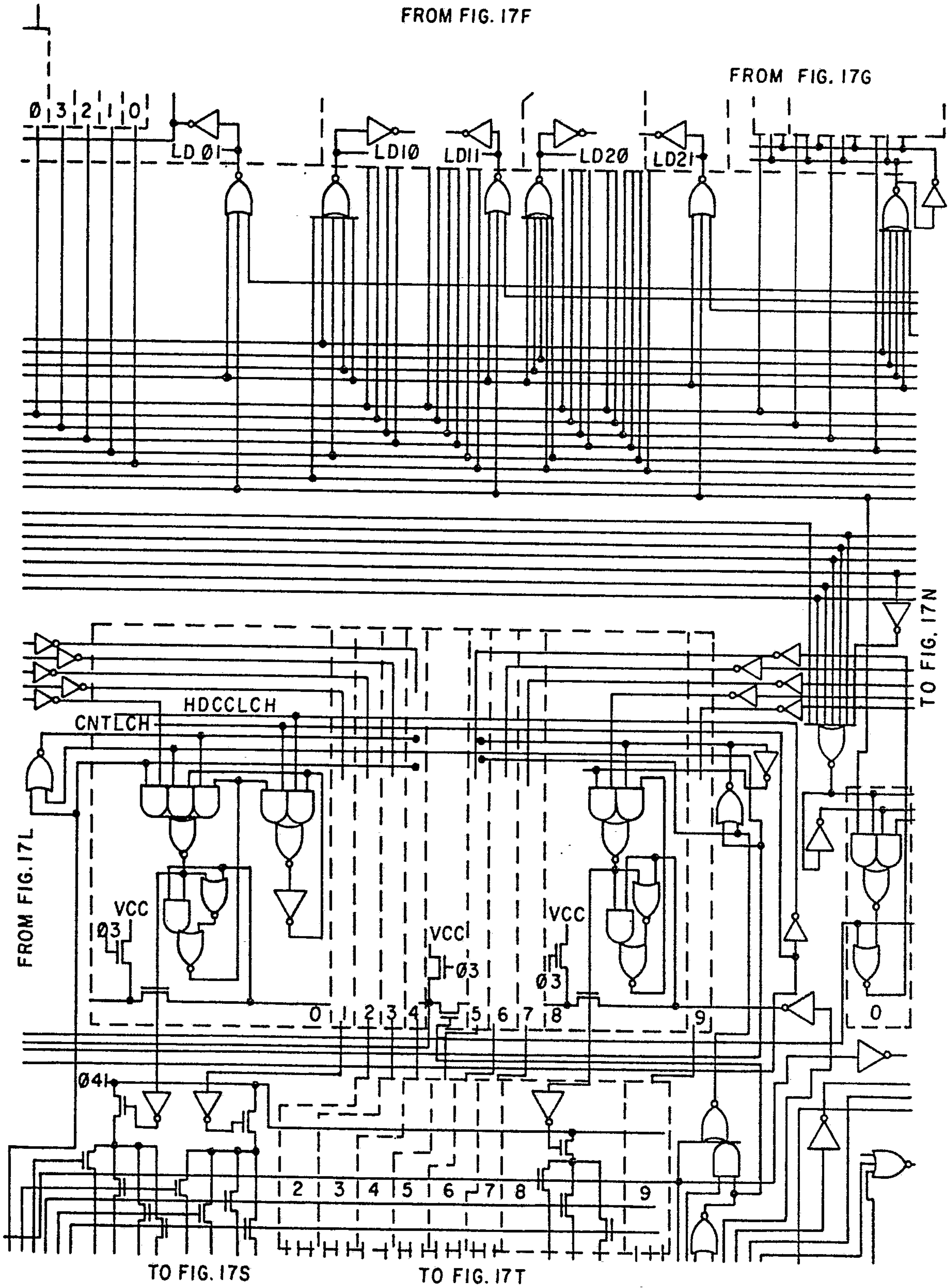

FROM FIG. 17F
FROM FIG. 17G
Ø 3 2 1 0
LD Ø1
LD1Ø
LD11
LD2Ø
LD21
HDCCLCH
CNTLCH
VCC
Ø3
0 1 2 3 4 5 6 7 8 9
0
Ø41
2 3 4 5 6 7 8 9
FROM FIG. 17L
TO FIG. 17N
TO FIG. 17S
TO FIG. 17T

Fig. 17m

0165665

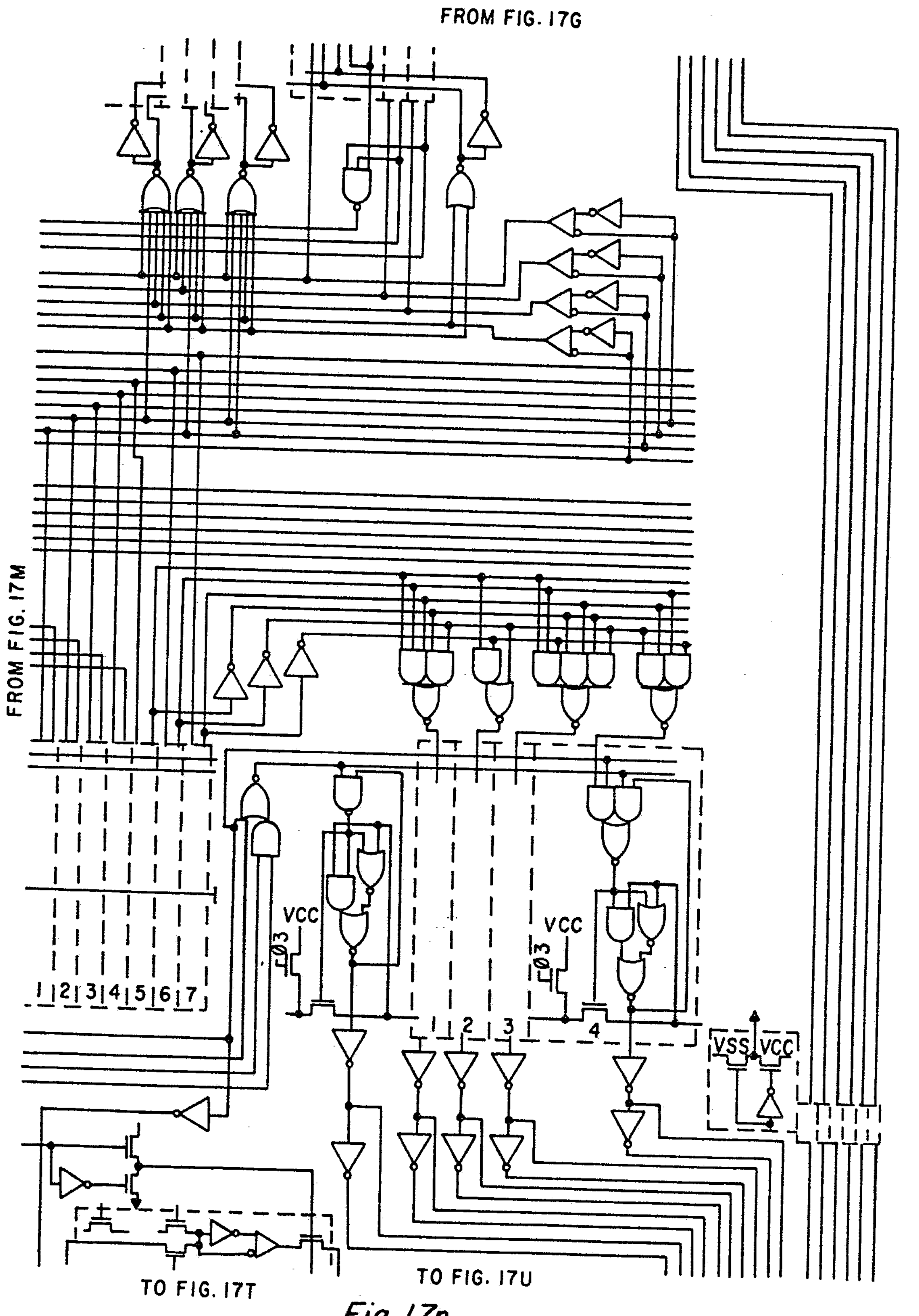



*Fig. 17n*

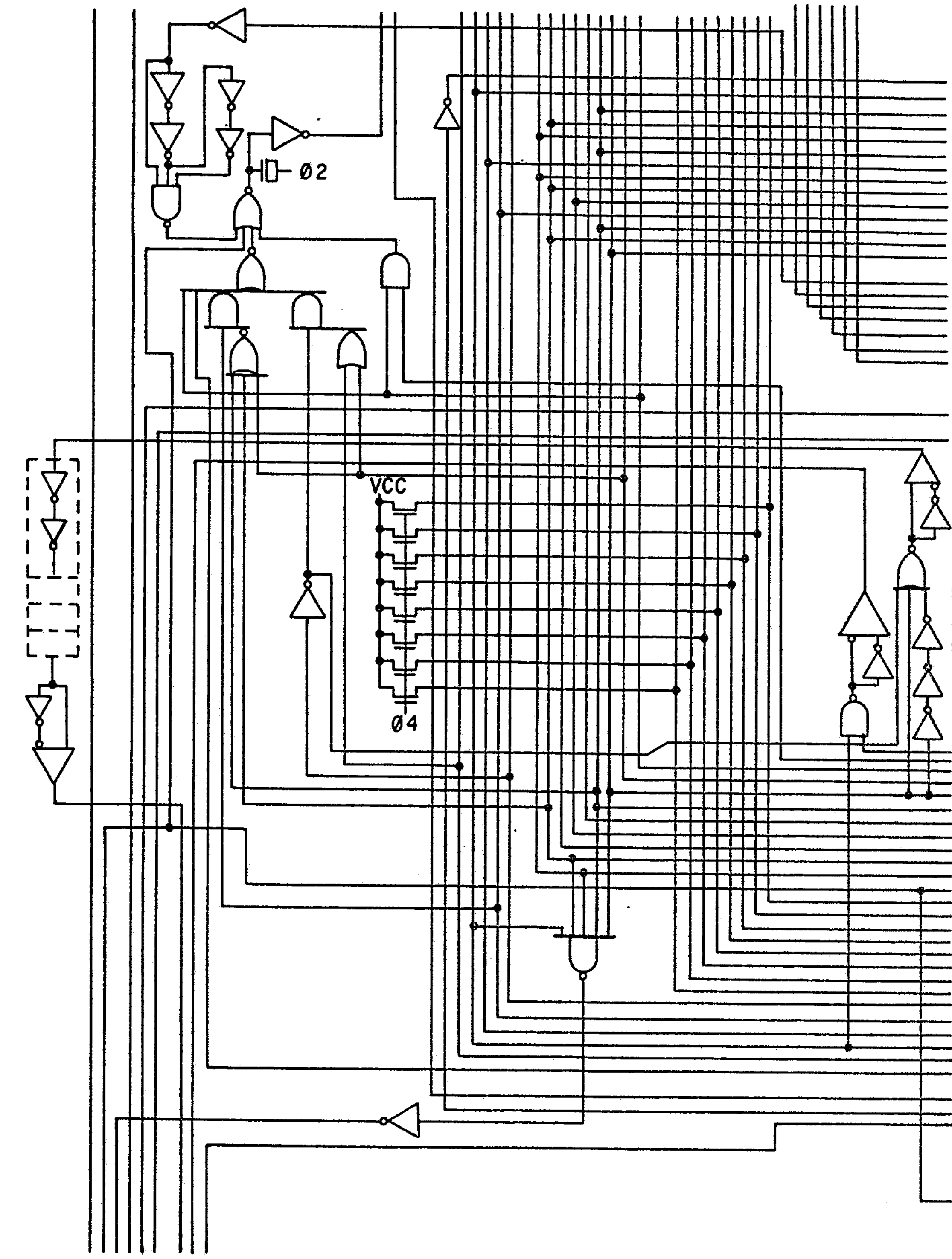



Fig. 17o

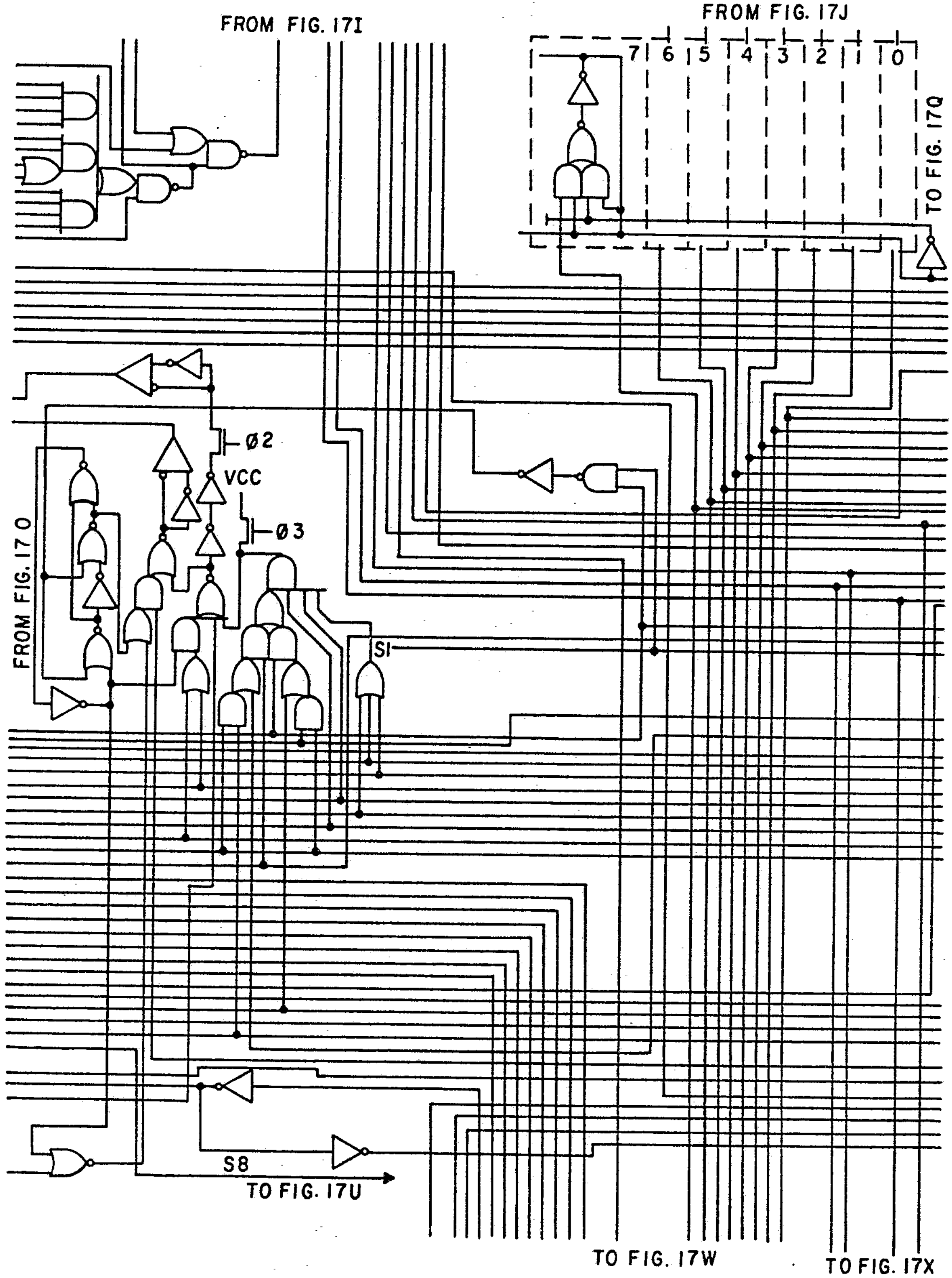

FROM FIG. 17I
FROM FIG. 17J
7
6
5
4
3
2
1
0
TO FIG. 17Q
FROM FIG. 17 O
Ø2
VCC
Ø3
SI
S8
TO FIG. 17U
TO FIG. 17W
TO FIG. 17X

Fig. 17p

0165665

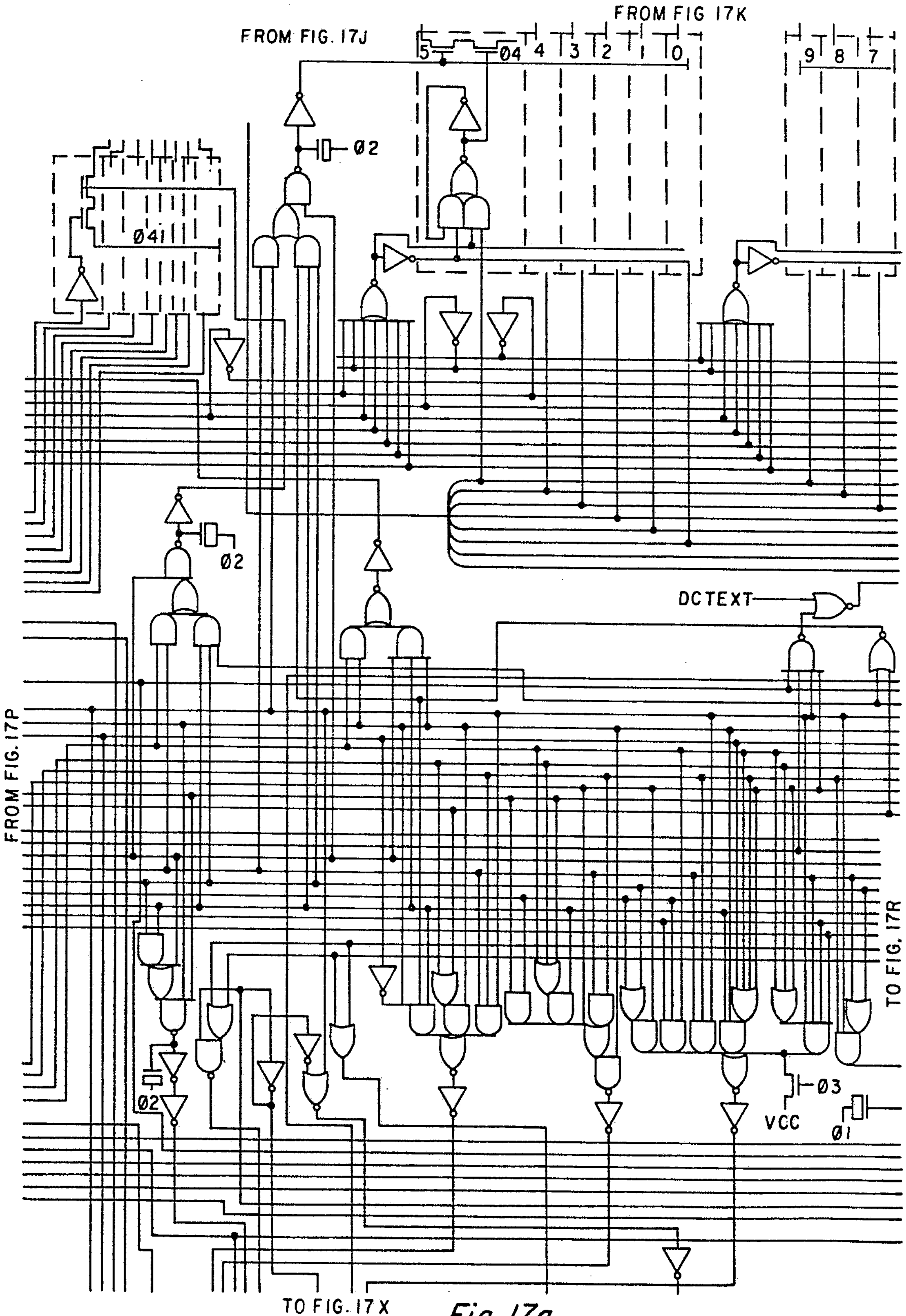



Fig. 17q

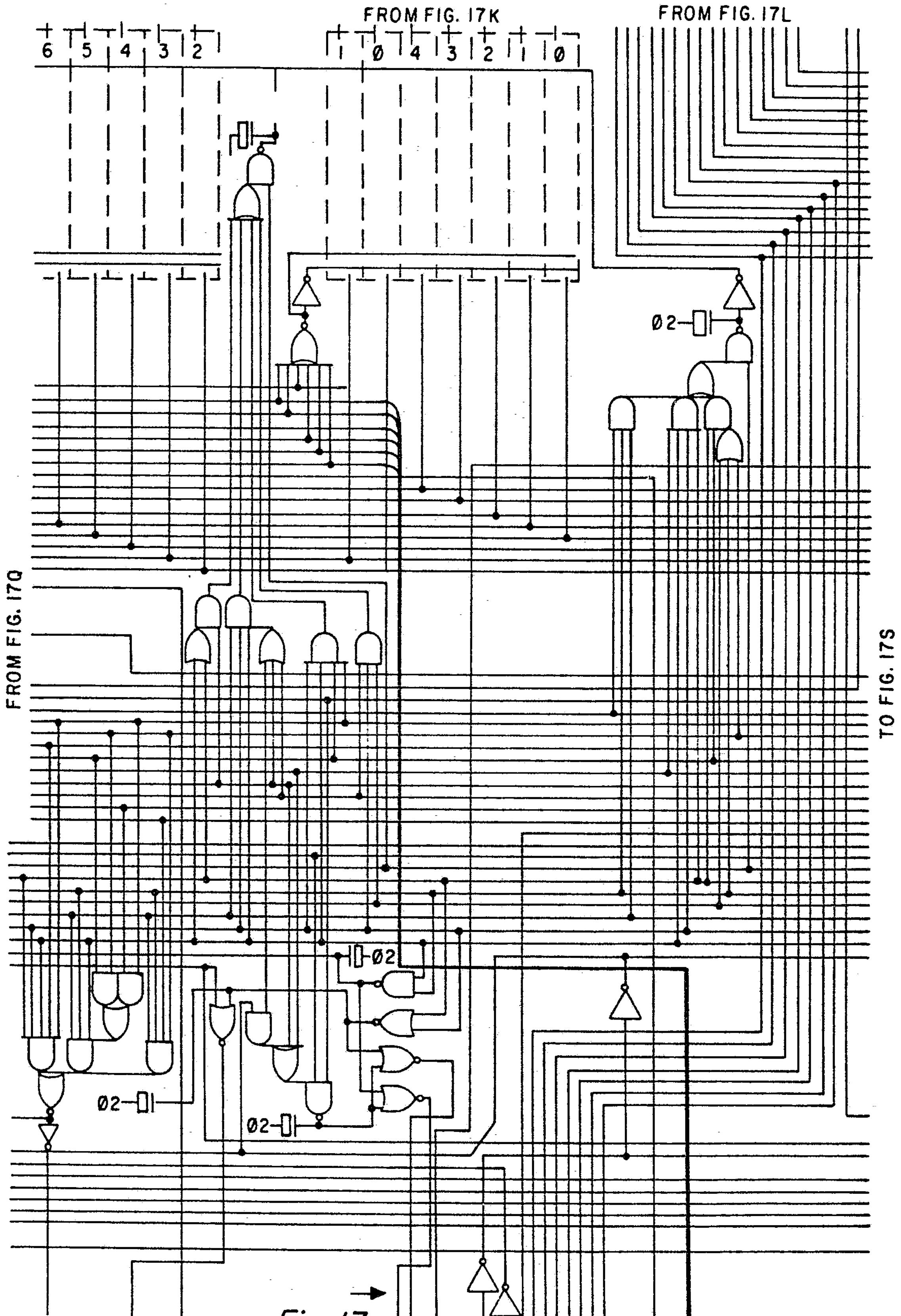

FROM FIG. 17K
FROM FIG. 17L
6
5
4
3
2
1
Ø
4
3
2
1
Ø
FROM FIG. 17Q
TO FIG. 17S
Ø2

Fig. 17r

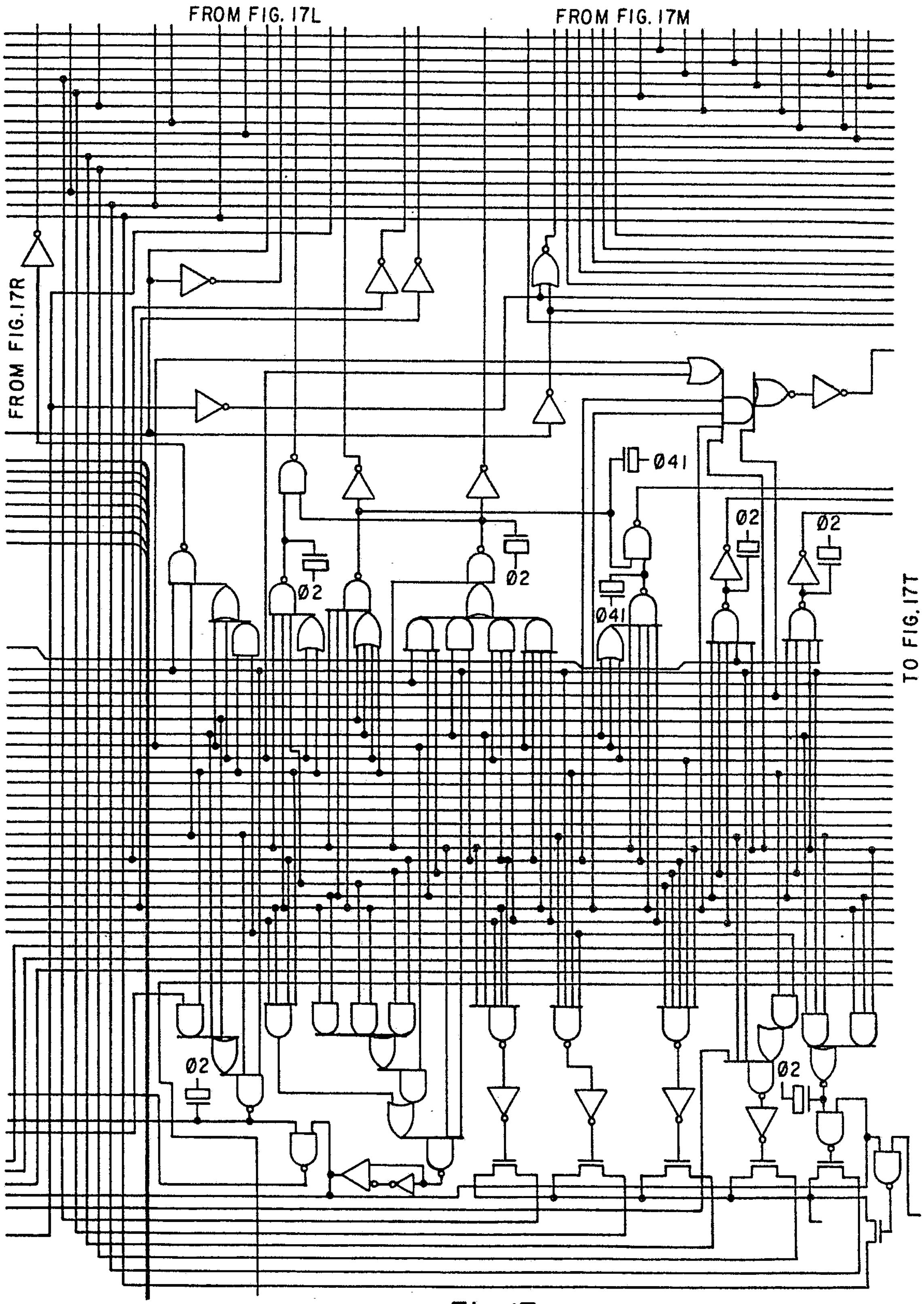
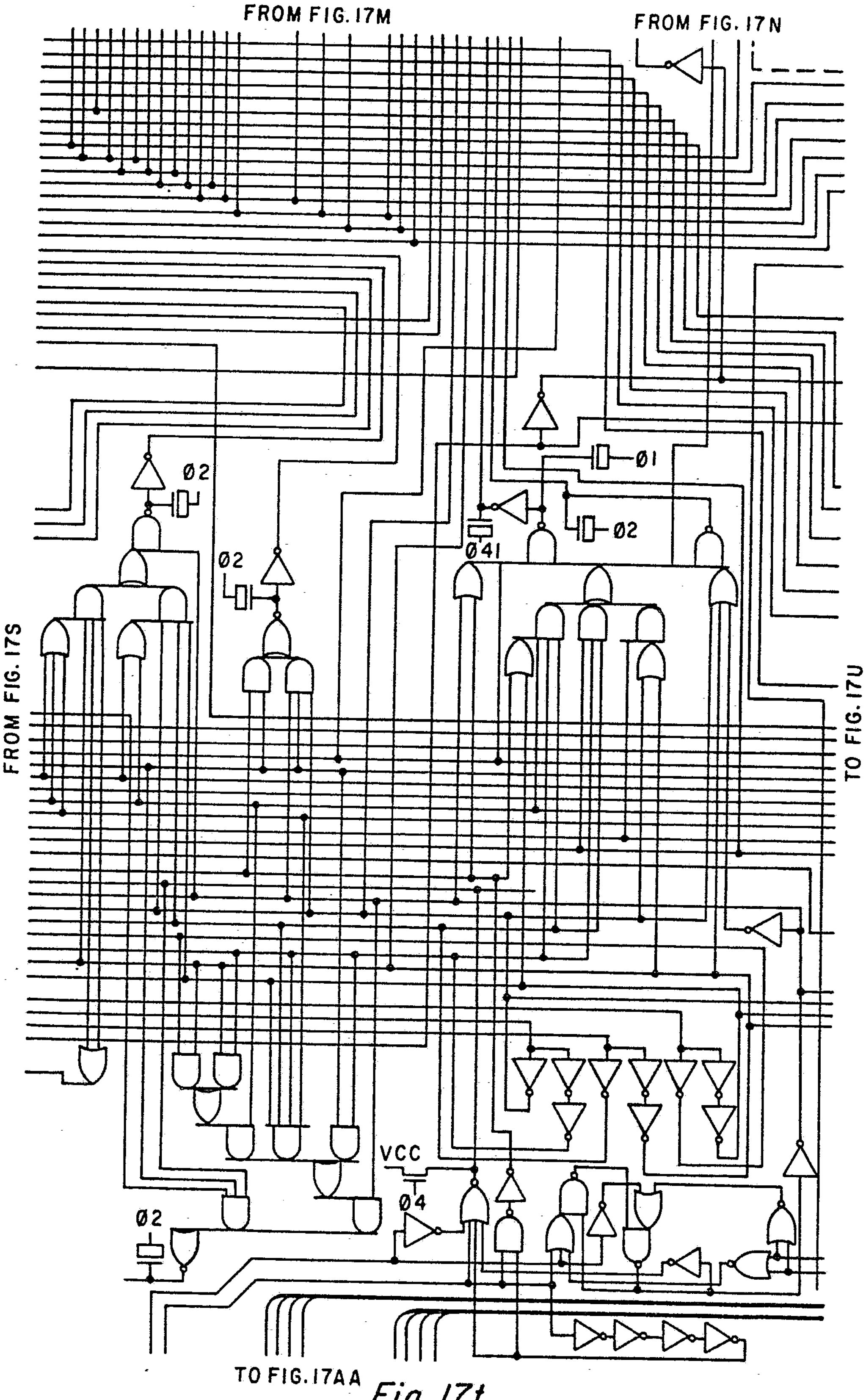

FROM FIG. 17L
FROM FIG. 17M
FROM FIG. 17R
Ø41
Ø2
Ø41
Ø2
Ø2
Ø2
Ø2
Ø2
Ø2
TO FIG. 17T
TO FIG. 17Z

Fig. 17s



FROM FIG. 17M
FROM FIG. 17N
FROM FIG. 17S
TO FIG. 17U
Ø2
Ø2
Ø1
Ø2
Ø41
VCC
Ø4
Ø2
TO FIG. 17AA


Fig. 17t

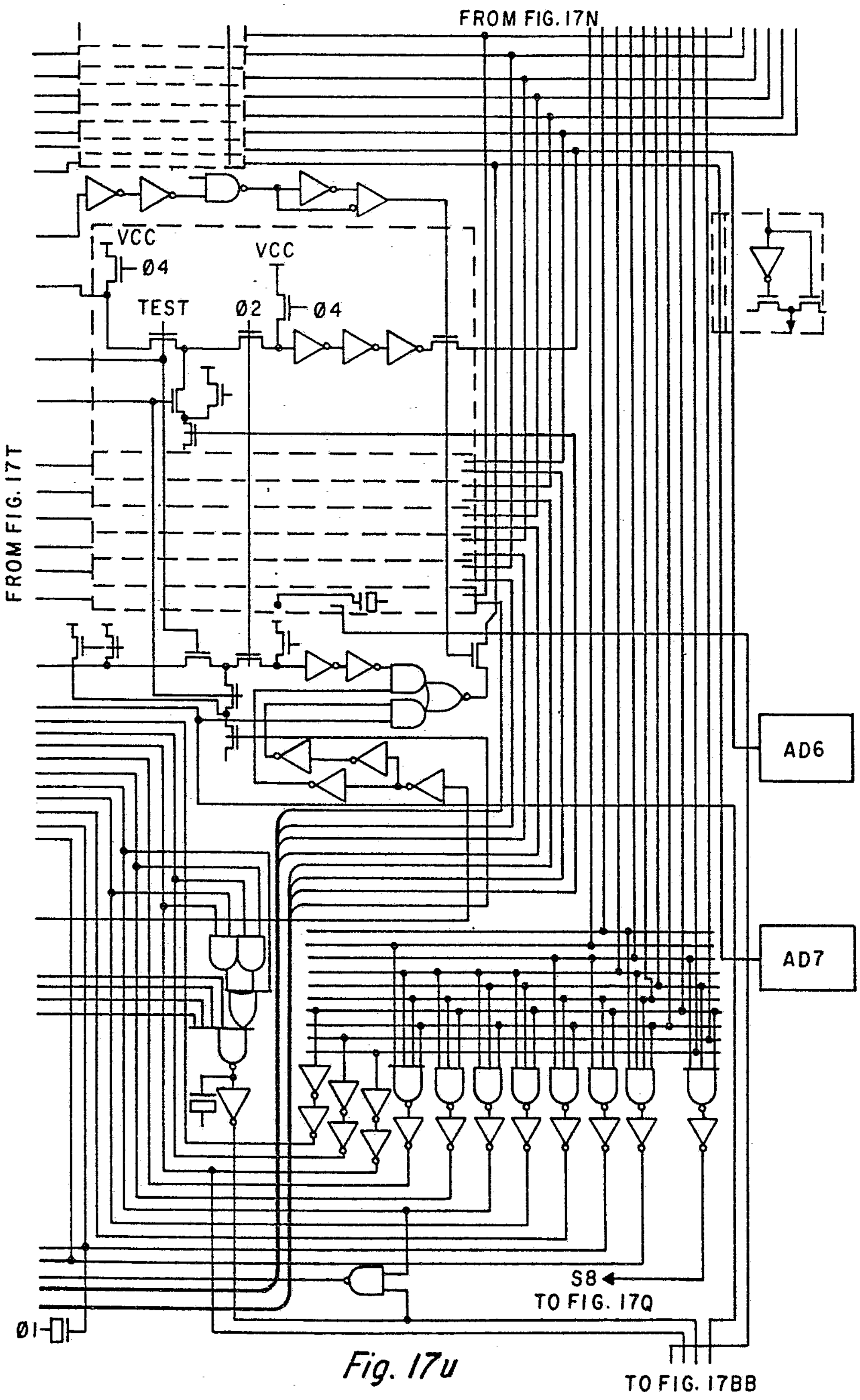

FROM FIG. 17N
FROM FIG. 17T
VCC
Ø4
VCC
TEST
Ø2
Ø4
AD6
AD7
S8
TO FIG. 17Q
Ø1
TO FIG. 17BB

Fig. 17u

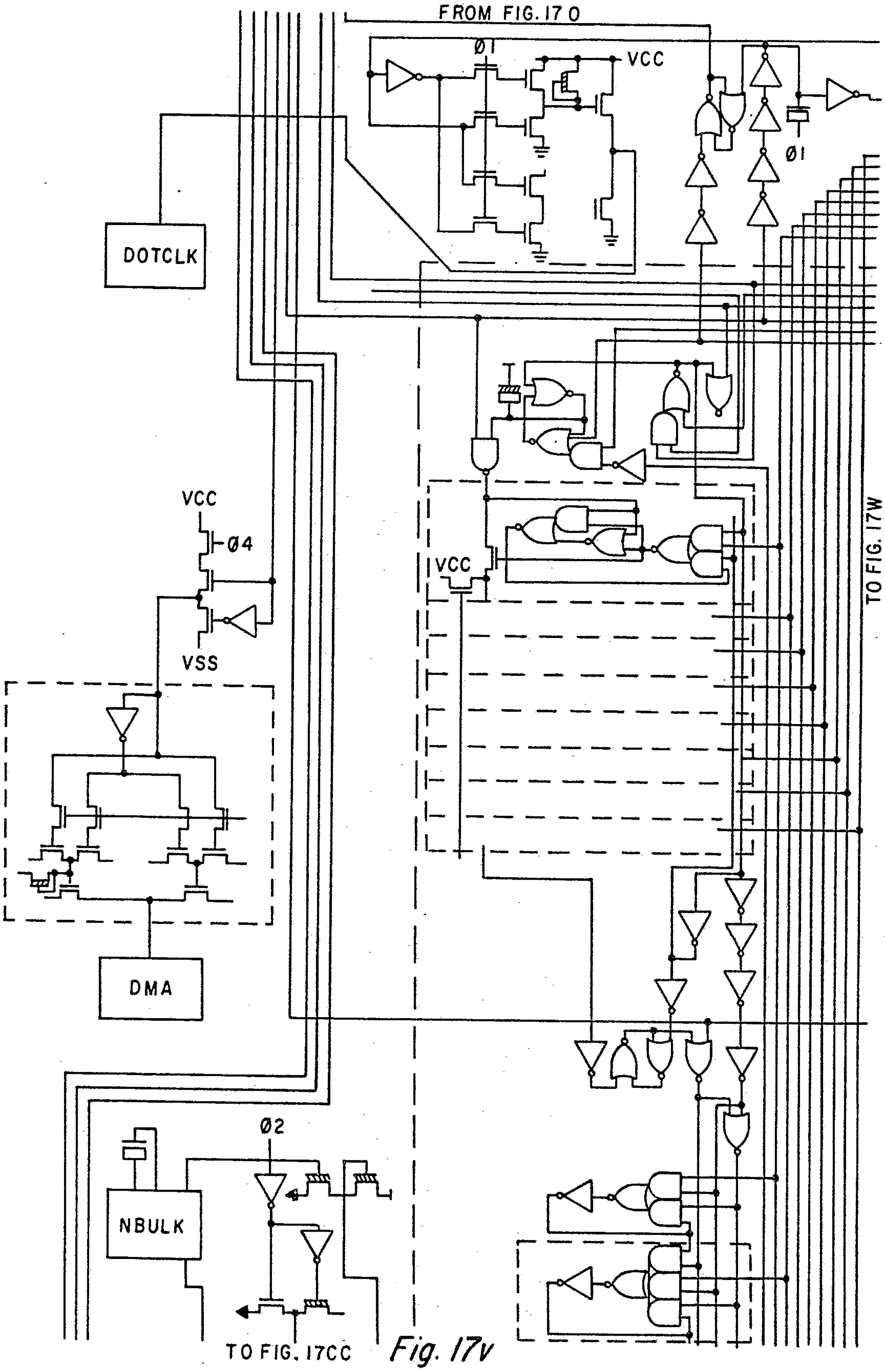

FROM FIG. 17 O
Ø1
VCC
Ø1
DOTCLK
TO FIG. 17W
VCC
Ø4
VSS
VCC
DMA
Ø2
NBULK
TO FIG. 17CC

Fig. 17v

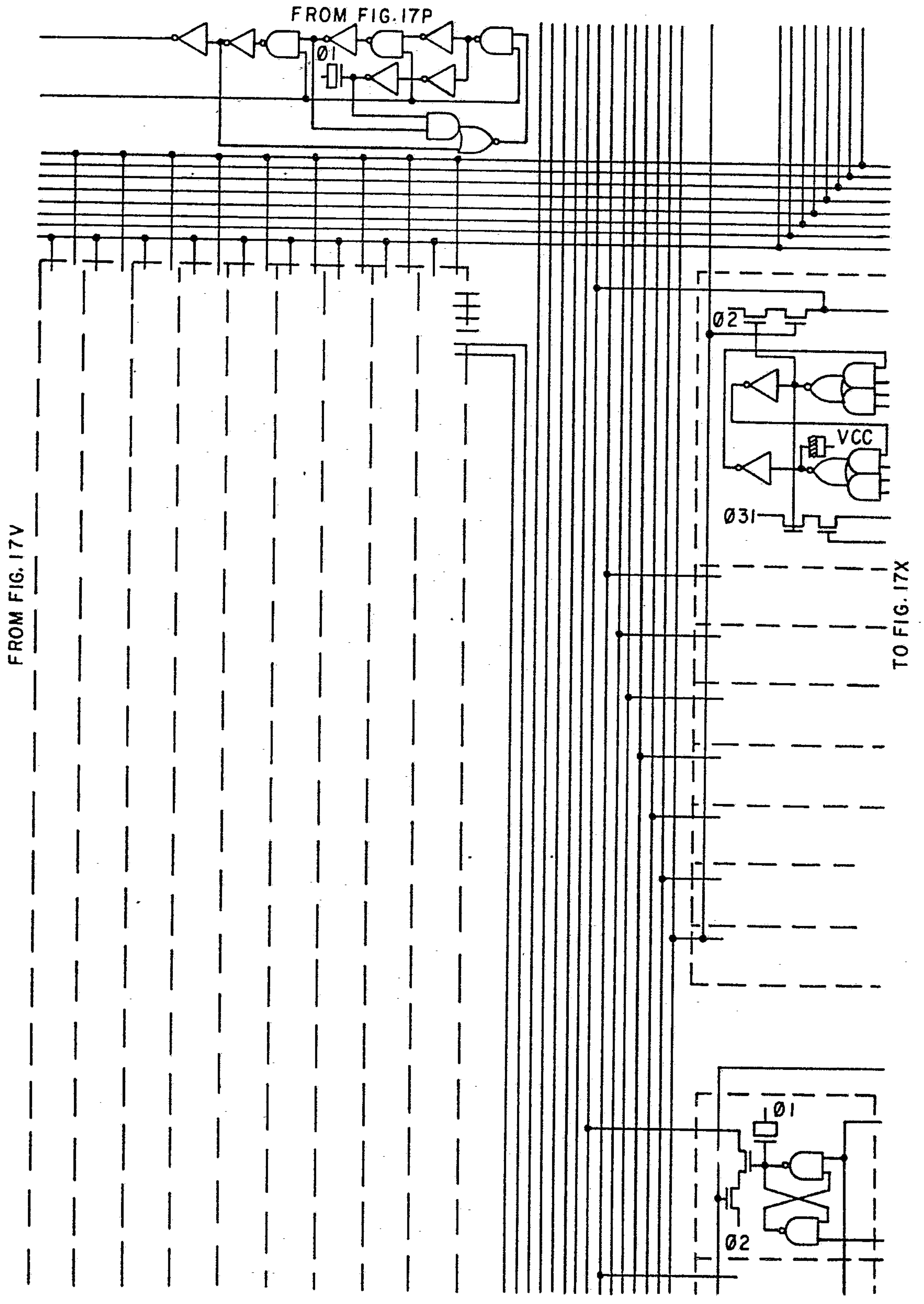



Fig. 17w

TO FIG. 17DD

0165665

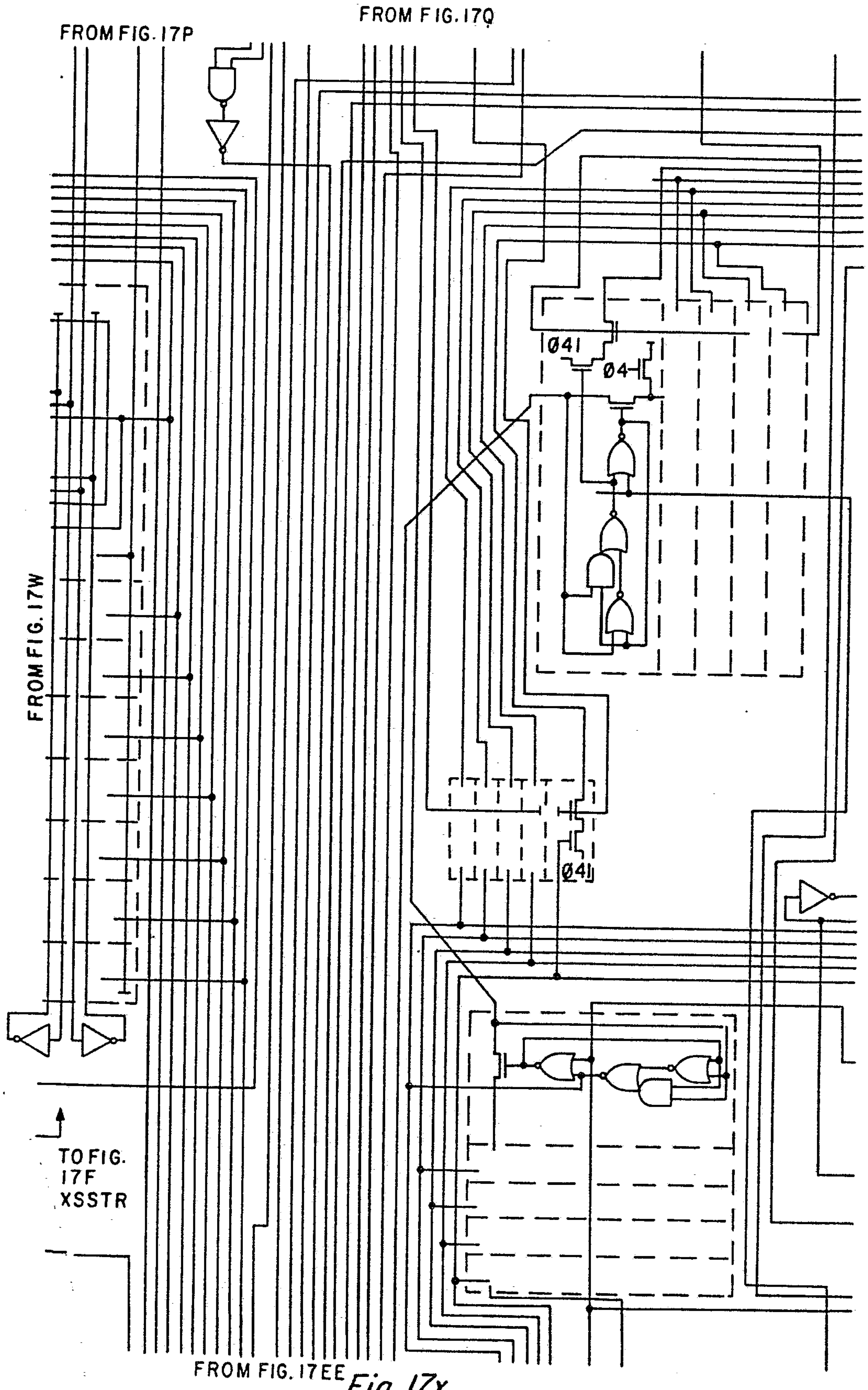



Fig. 17x

0165665

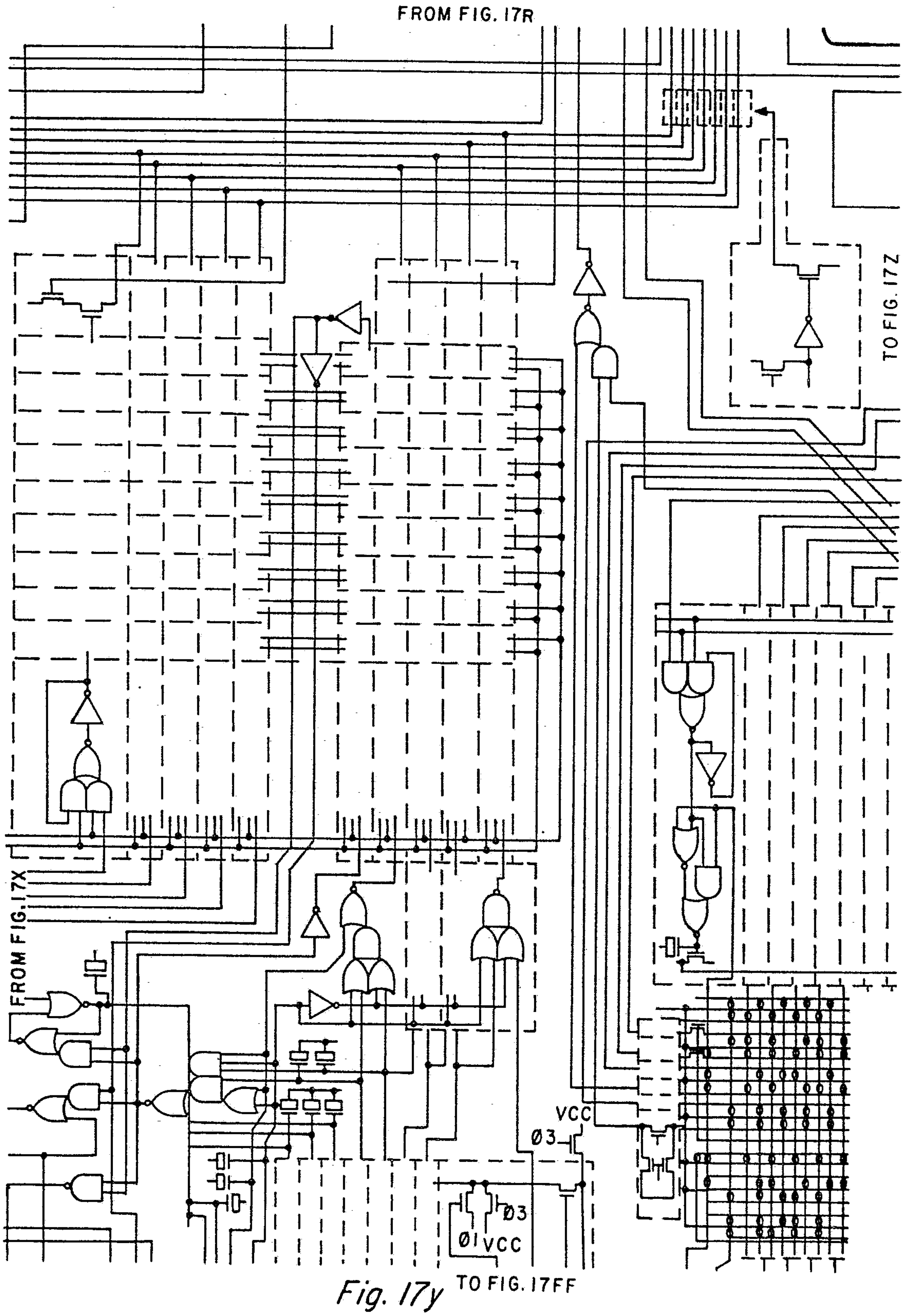



Fig. 17y

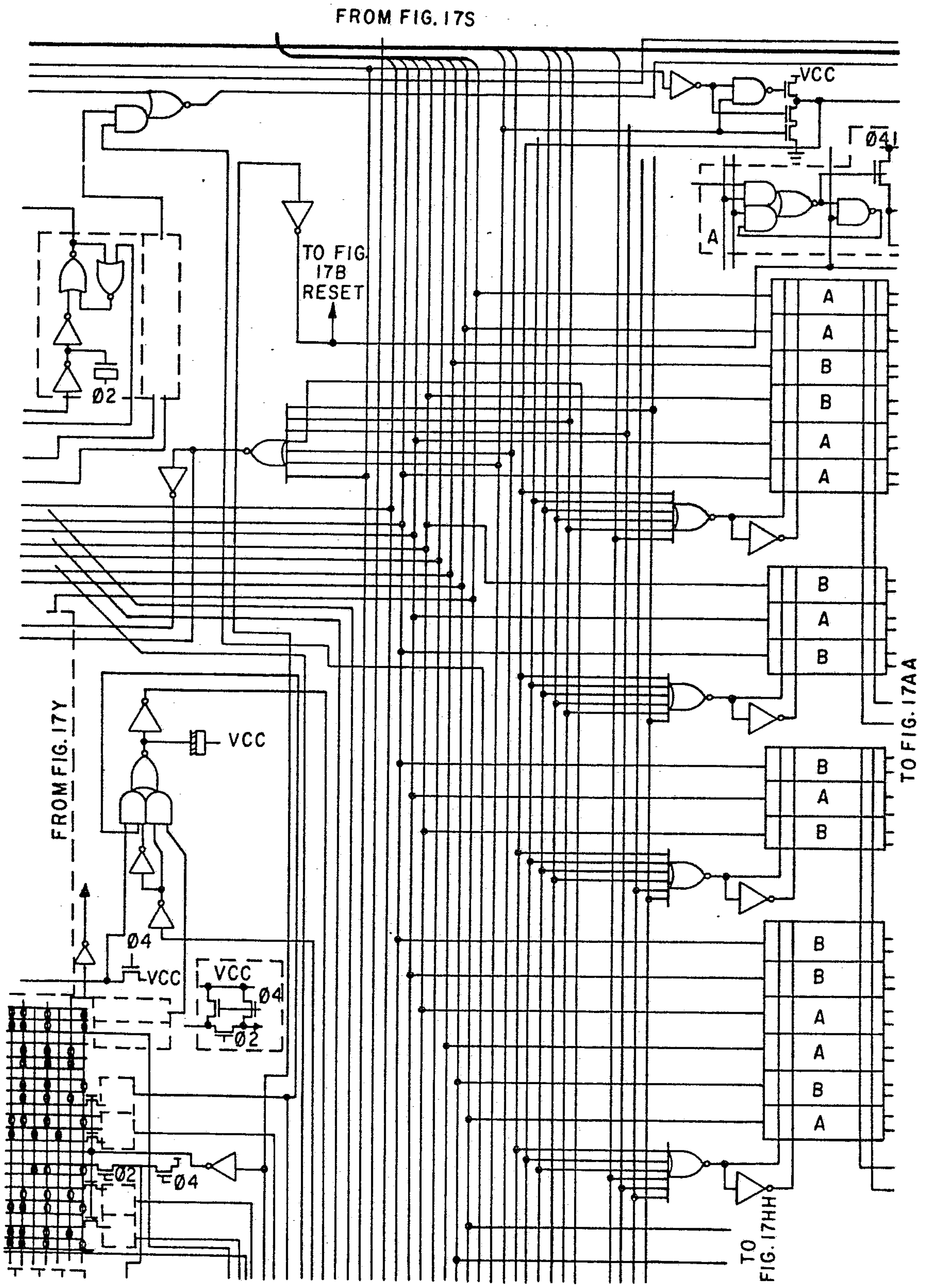

FROM FIG. 17S
VCC
Ø41
A
TO FIG. 17B RESET
Ø2
A
A
B
B
A
A
B
A
B
TO FIG. 17AA
FROM FIG. 17Y
VCC
B
A
B
Ø4
VCC
VCC
Ø4
Ø2
B
B
A
A
B
A
Ø2
Ø4
TO FIG. 17HH
TO FIG. 17GG

Fig. 17z

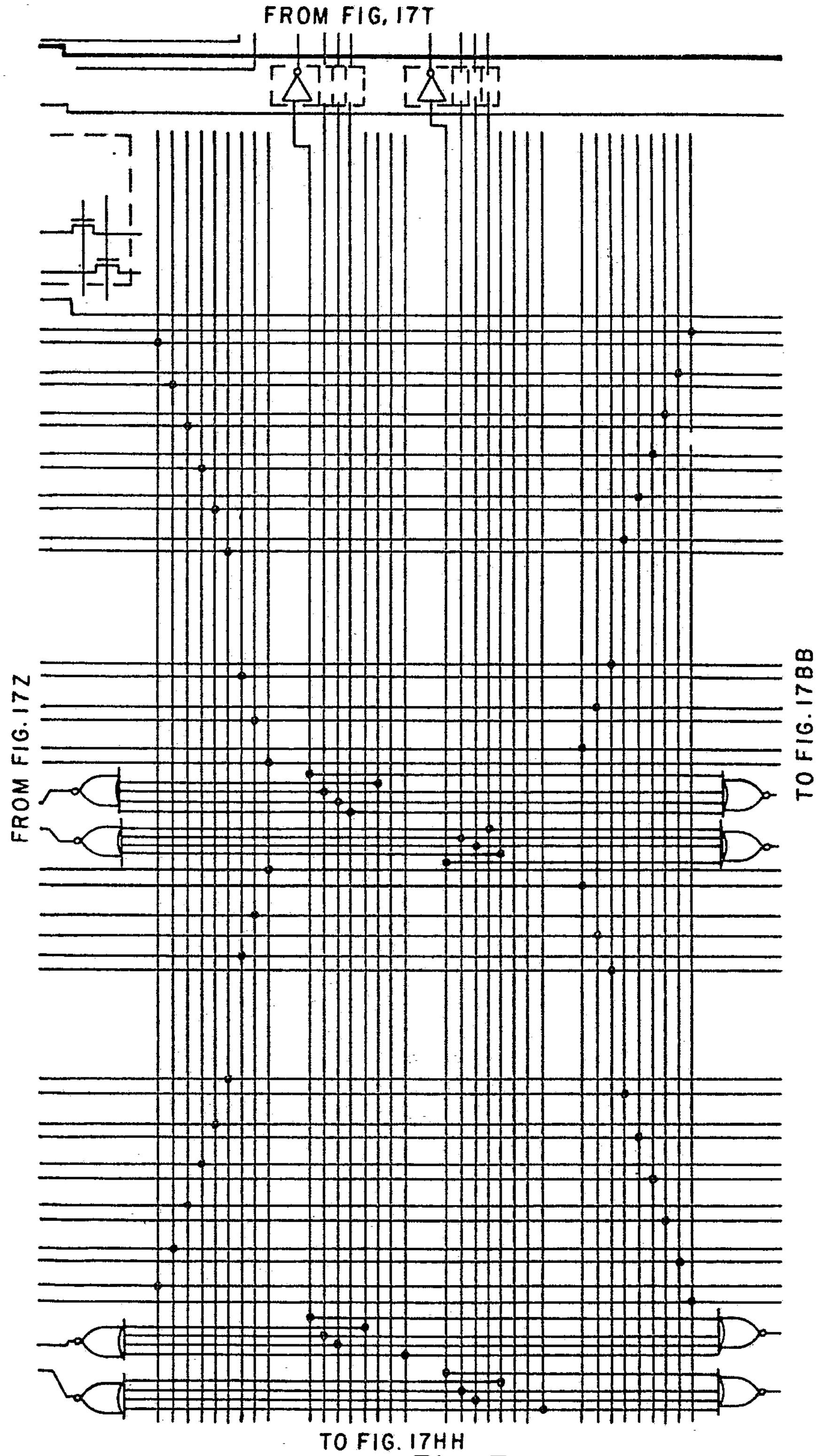



Fig. 17aa

0165665



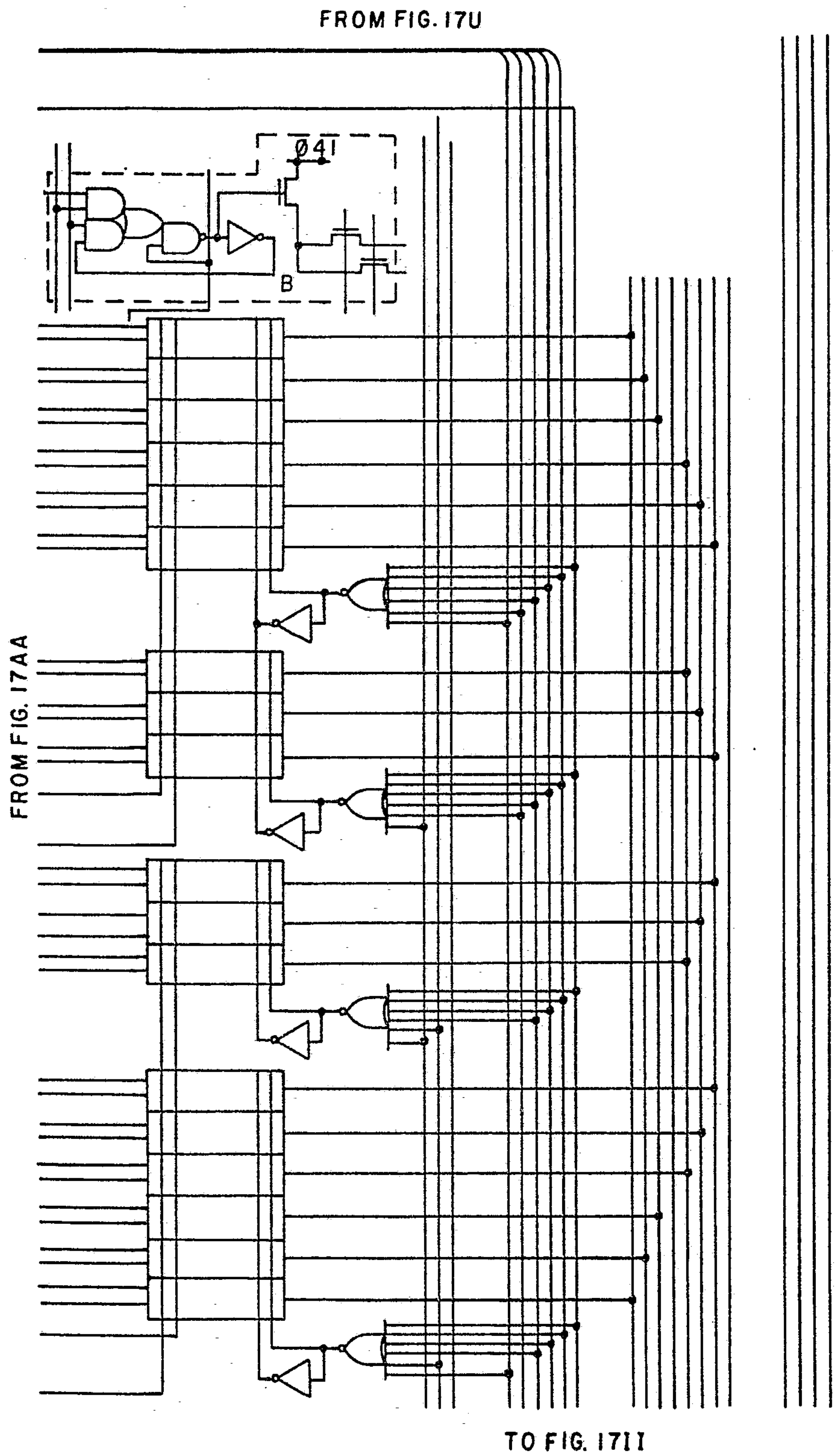



Fig. 17bb

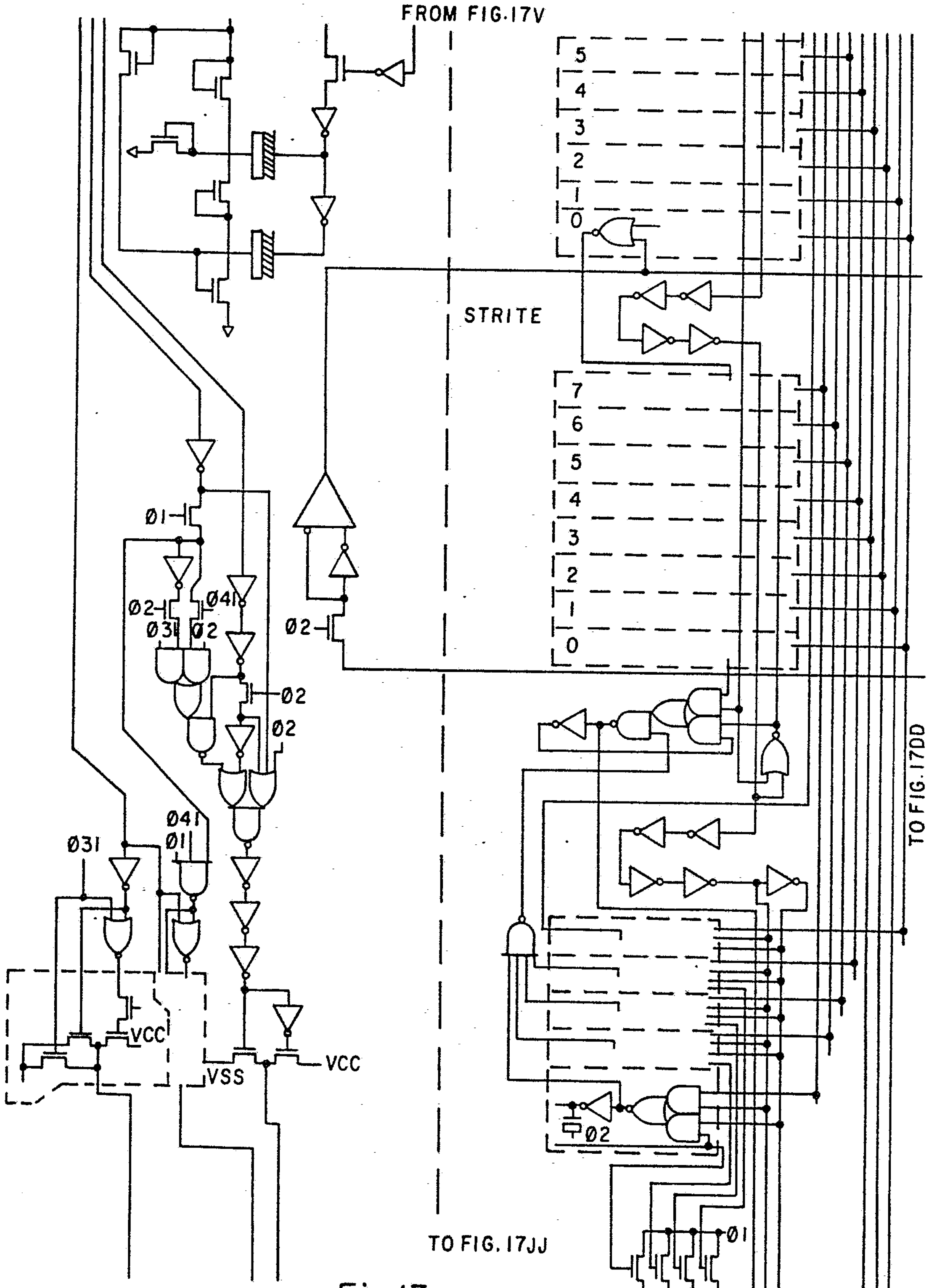

FROM FIG.17V
5
4
3
2
1
0
STRITE
7
6
5
4
3
2
1
0
Ø1
Ø2
Ø41
Ø31
Ø2
Ø2
Ø2
Ø2
Ø41
Ø1
Ø31
VCC
VSS
VCC
Ø2
Ø1
TO FIG.17DD
TO FIG. 17JJ

Fig. 17cc

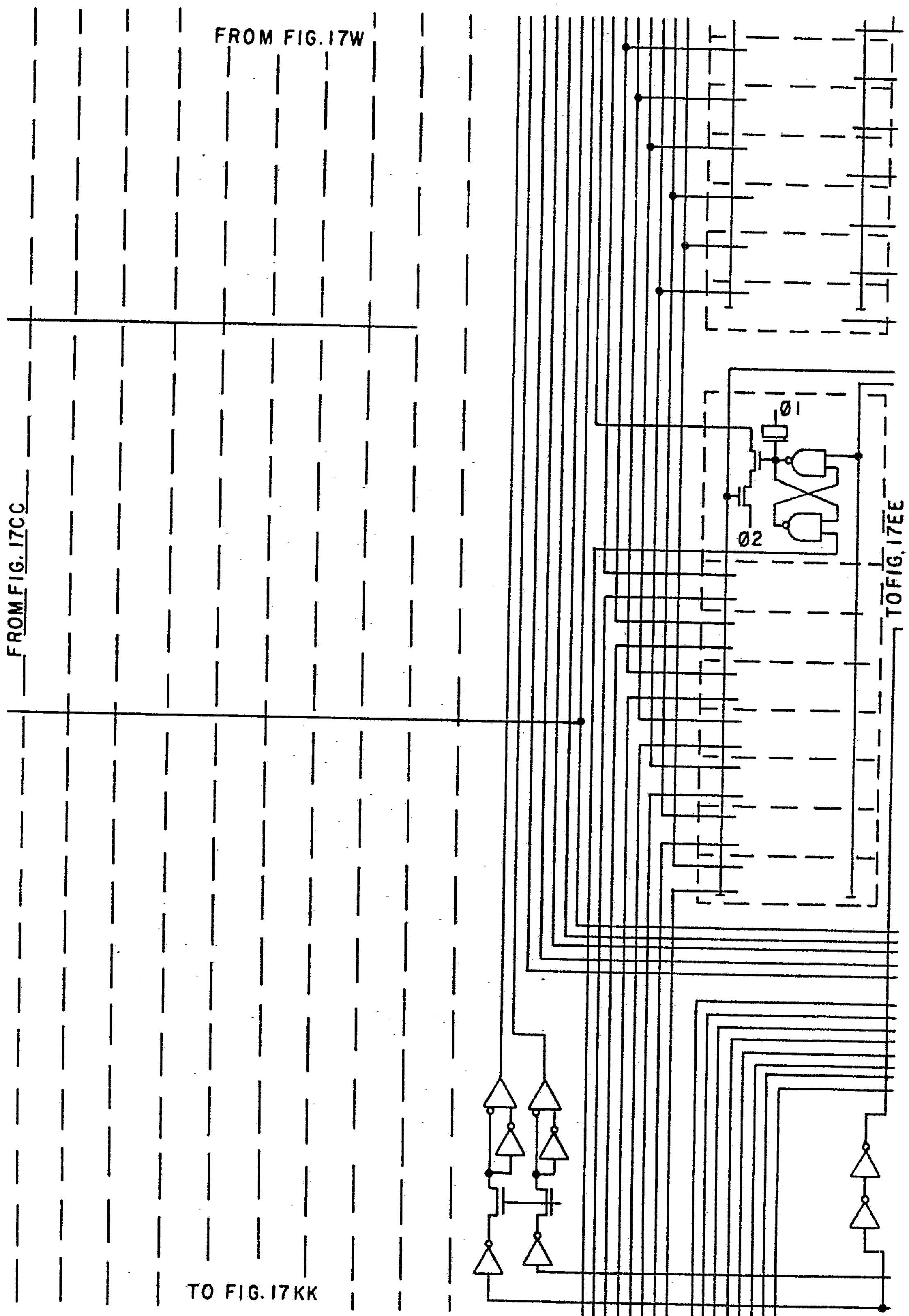
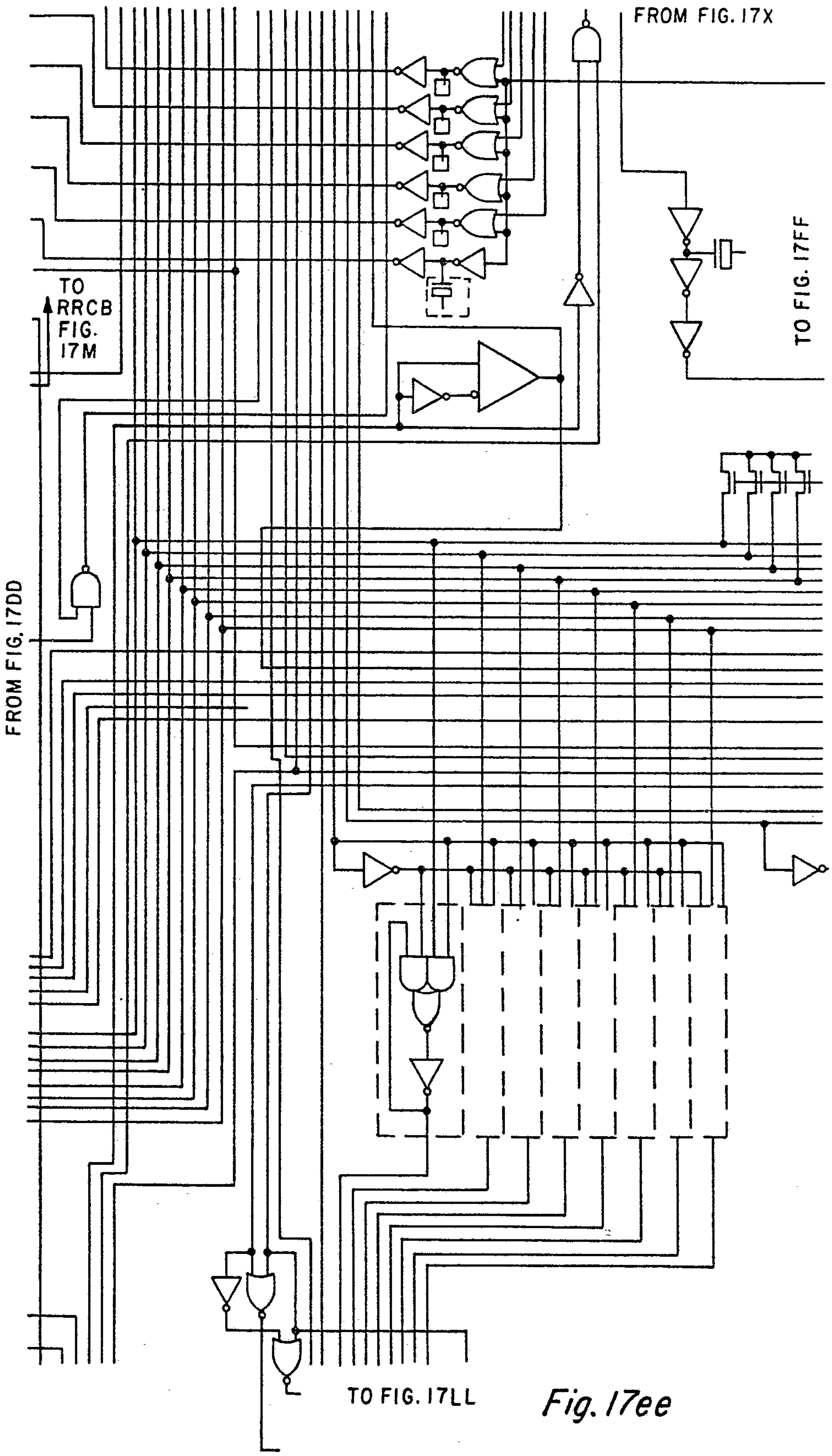



Fig. 17dd



FROM FIG. 17X
TO FIG. 17FF
TO RRCB FIG. 17M
FROM FIG. 17DD
TO FIG. 17LL


Fig. 17ee

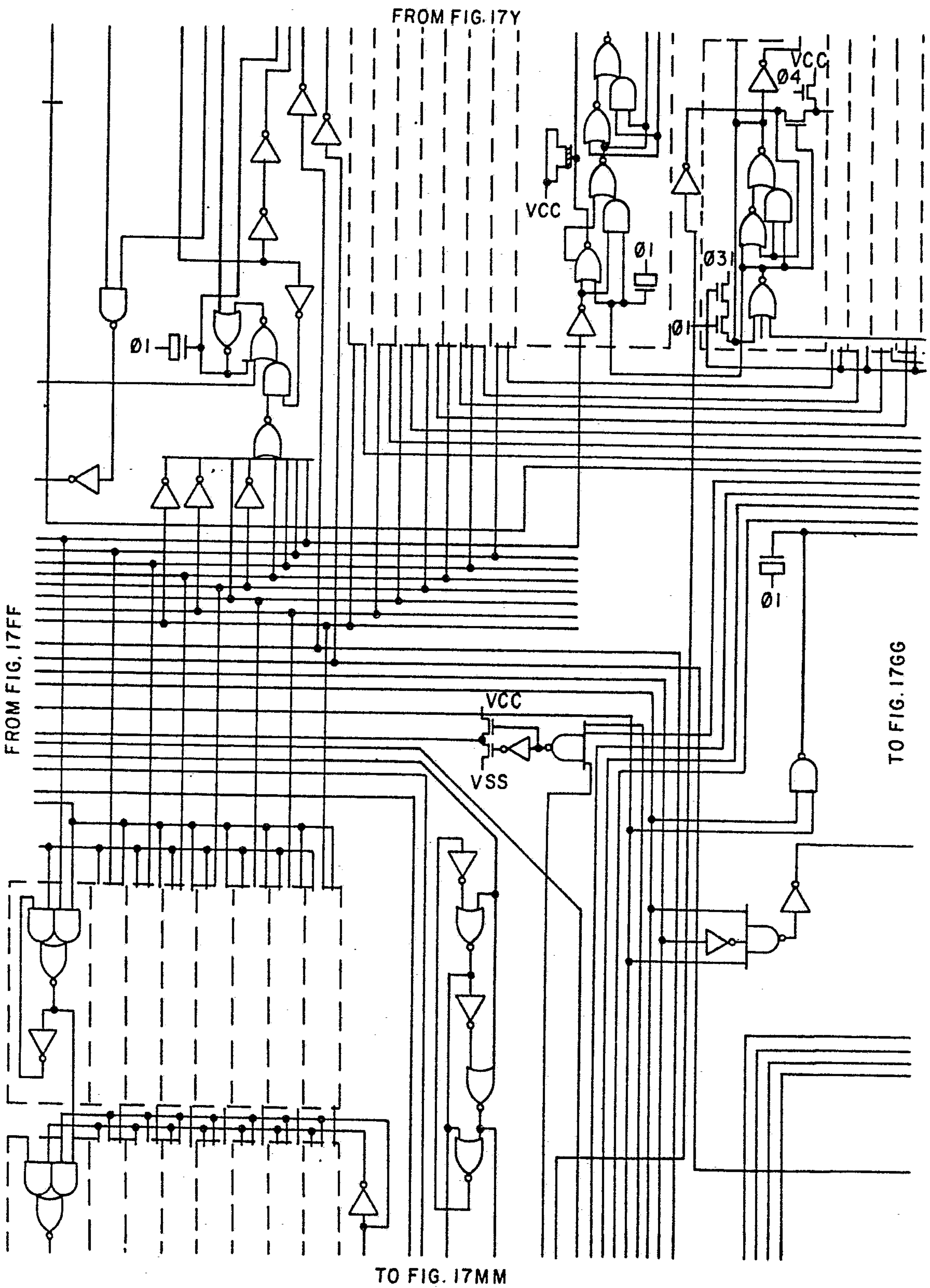

FROM FIG. 17Y
VCC
Ø4
VCC
Ø1
Ø3
Ø1
Ø1
FROM FIG. 17FF
Ø1
TO FIG. 17GG
VCC
VSS
TO FIG. 17MM

Fig. 17ff

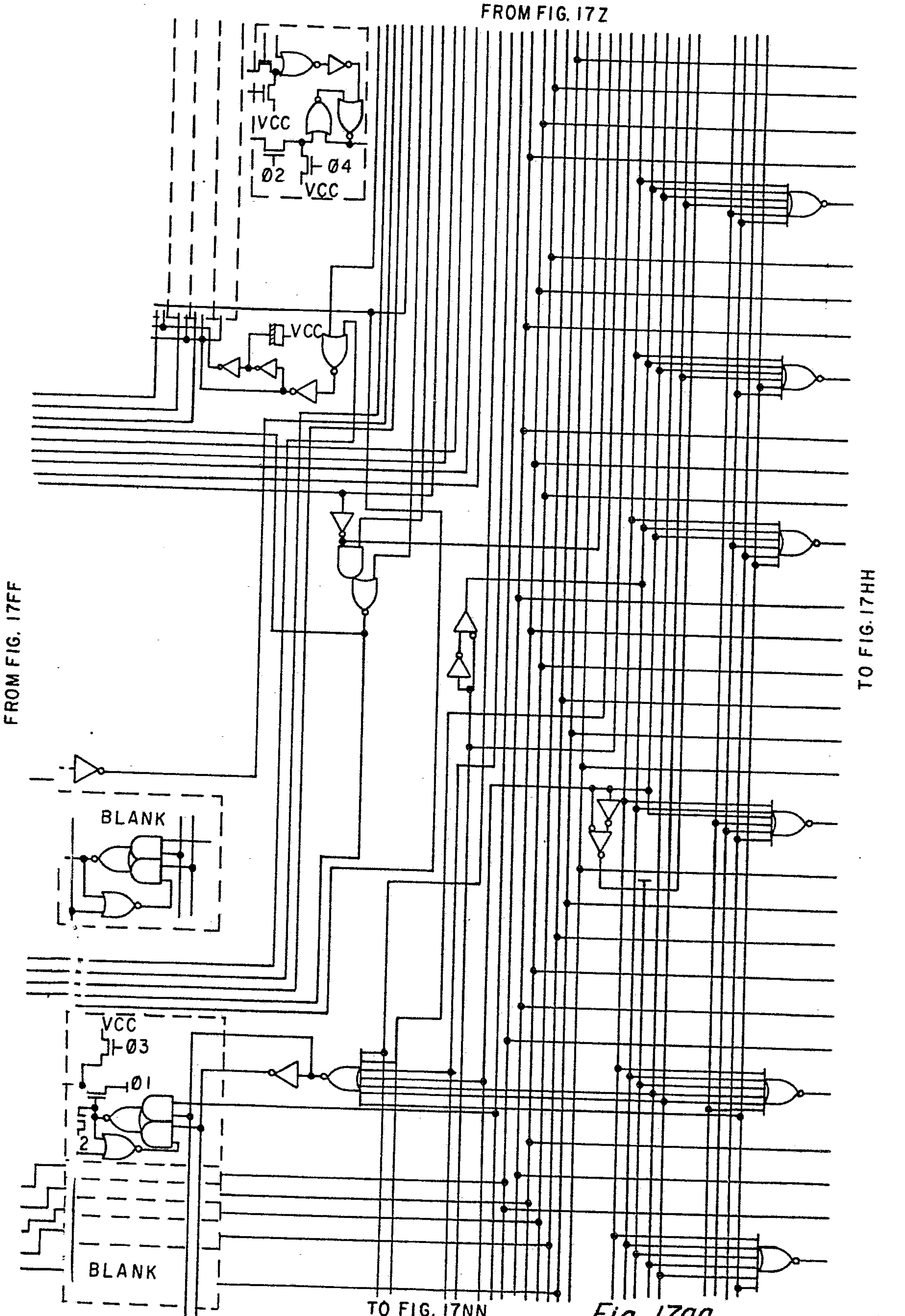

FROM FIG. 17Z
VCC
Ø2
Ø4
VCC
VCC
FROM FIG. 17FF
BLANK
TO FIG. 17HH
VCC
Ø3
Ø1
2
BLANK
TO FIG. 17NN

Fig. 17gg

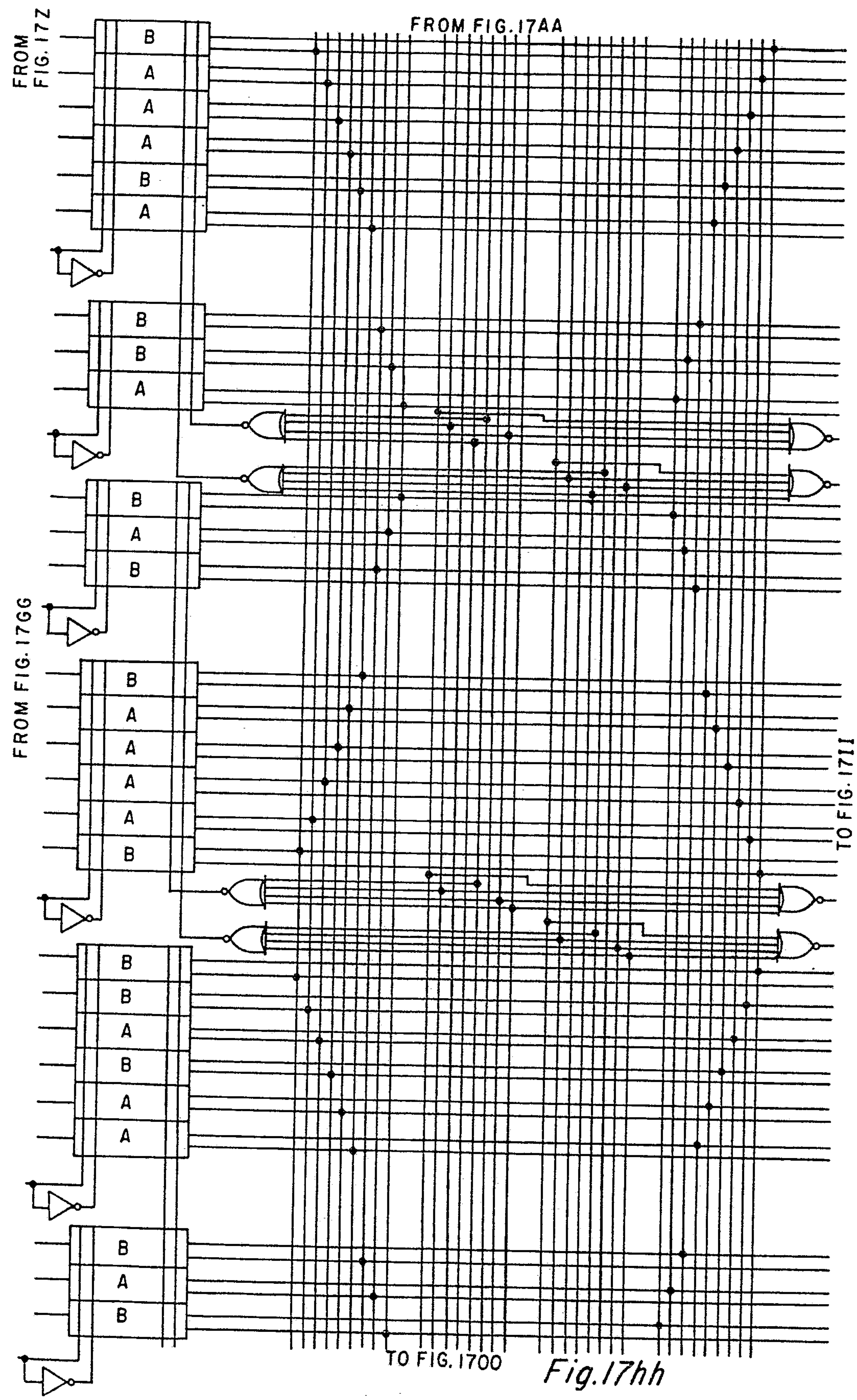

FROM FIG. 17Z
FROM FIG. 17AA
B
A
A
A
B
A
B
B
A
B
A
B
FROM FIG. 17GG
B
A
A
A
A
B
B
B
A
B
A
A
B
A
B
TO FIG. 17II
TO FIG. 17OO

Fig.17hh

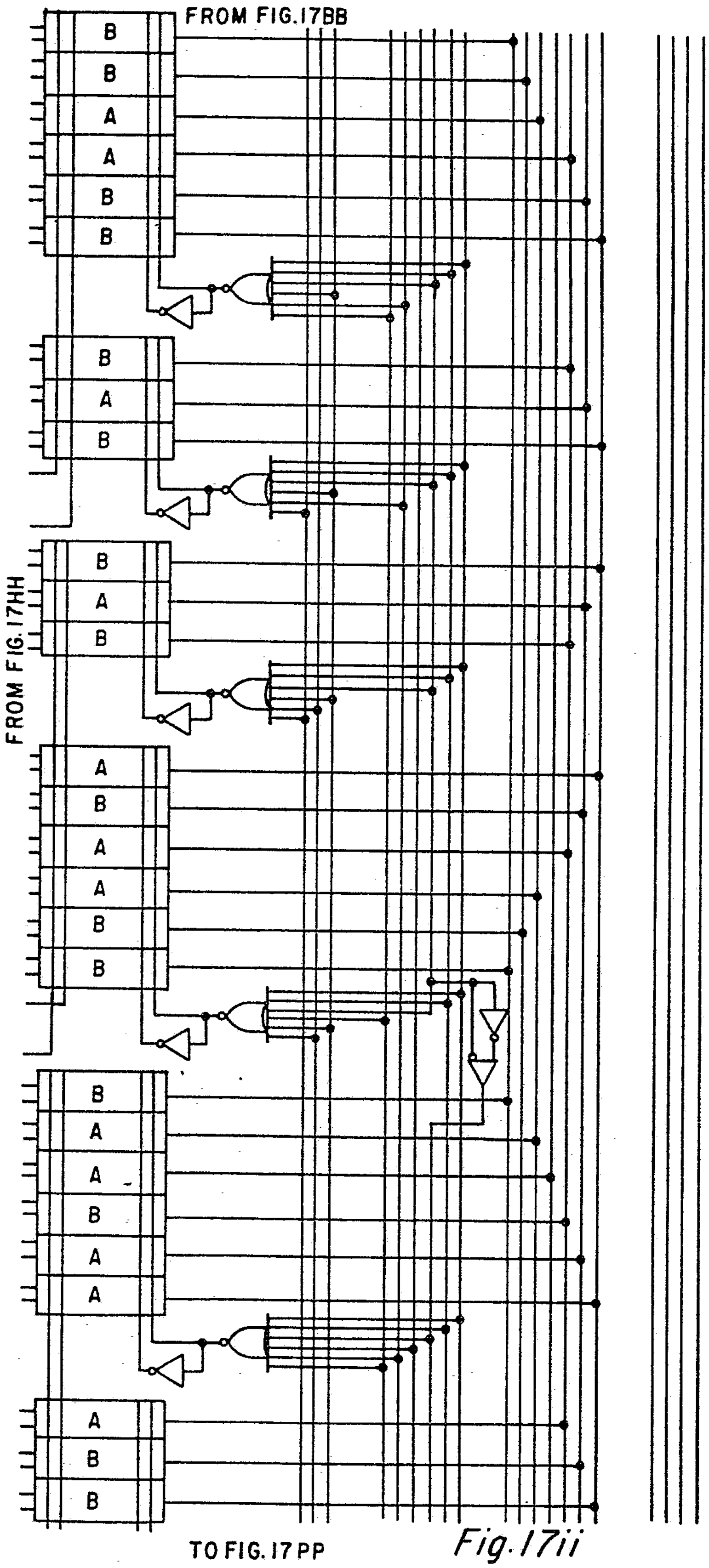

FROM FIG.17BB
B
B
A
A
B
B
B
A
B
FROM FIG.17HH
B
A
B
A
B
A
A
B
B
B
A
A
B
A
A
A
B
B
TO FIG.17PP

Fig.17ii

FROM FIG. 17CC

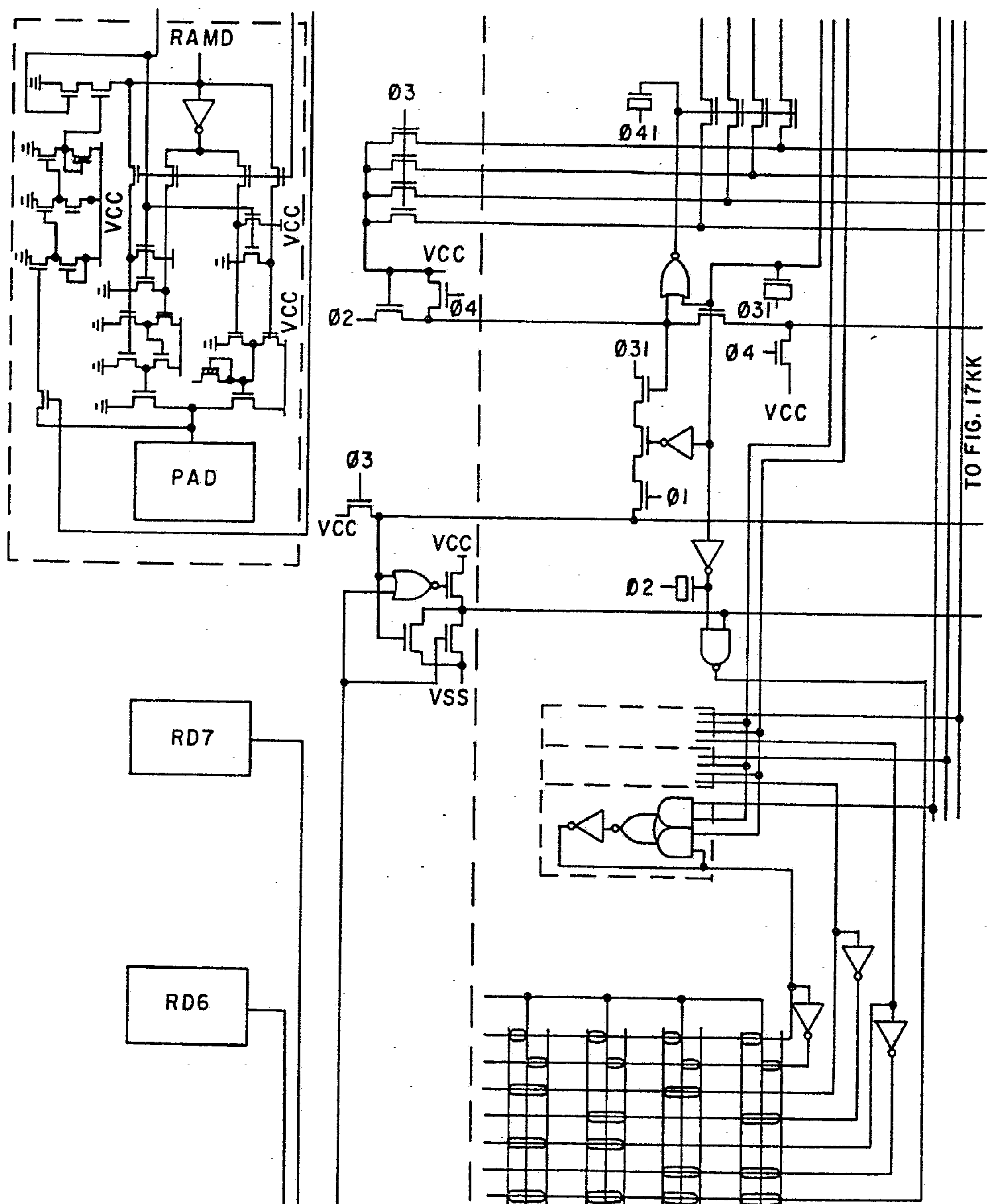



Fig. 17jj

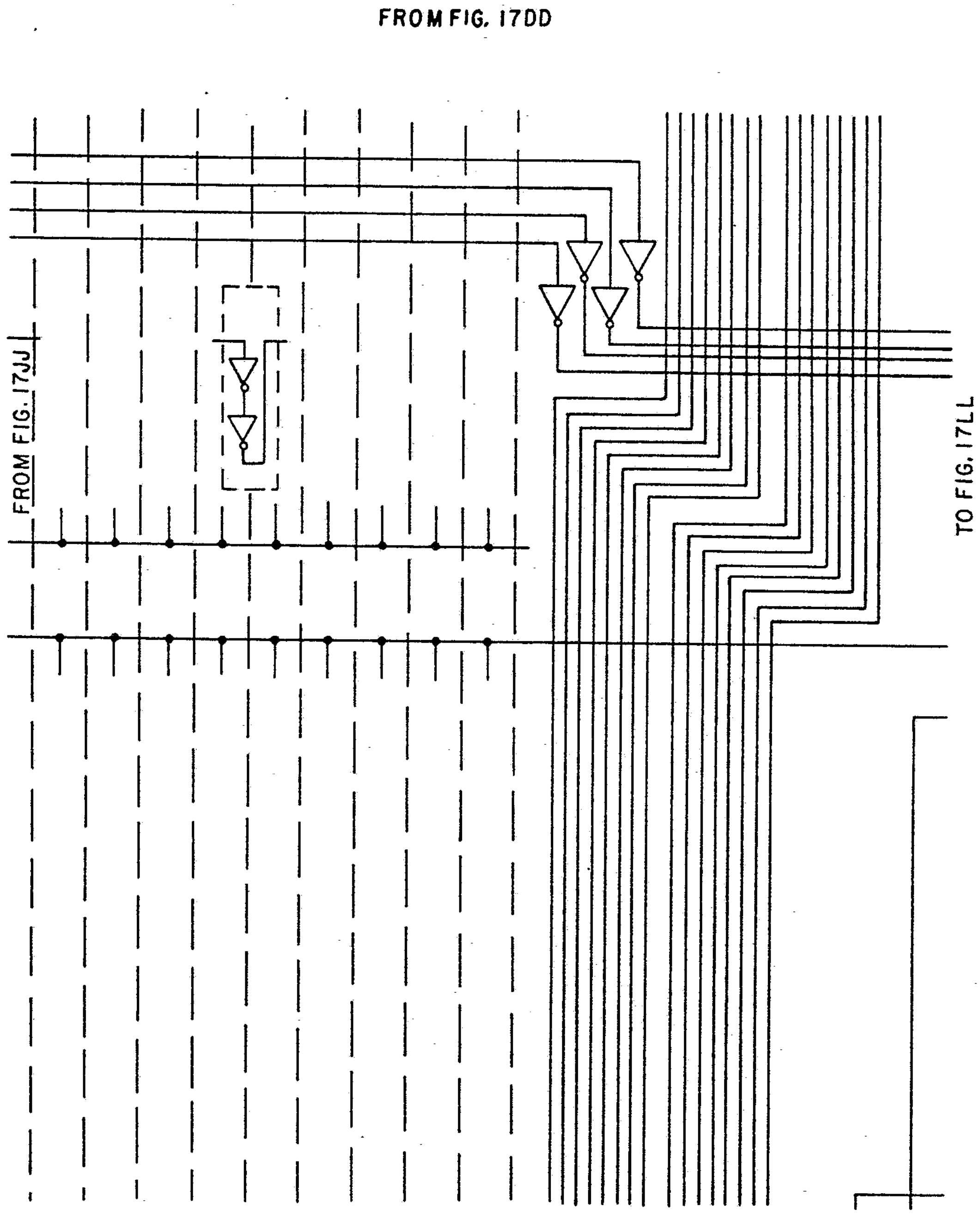



Fig. 17kk

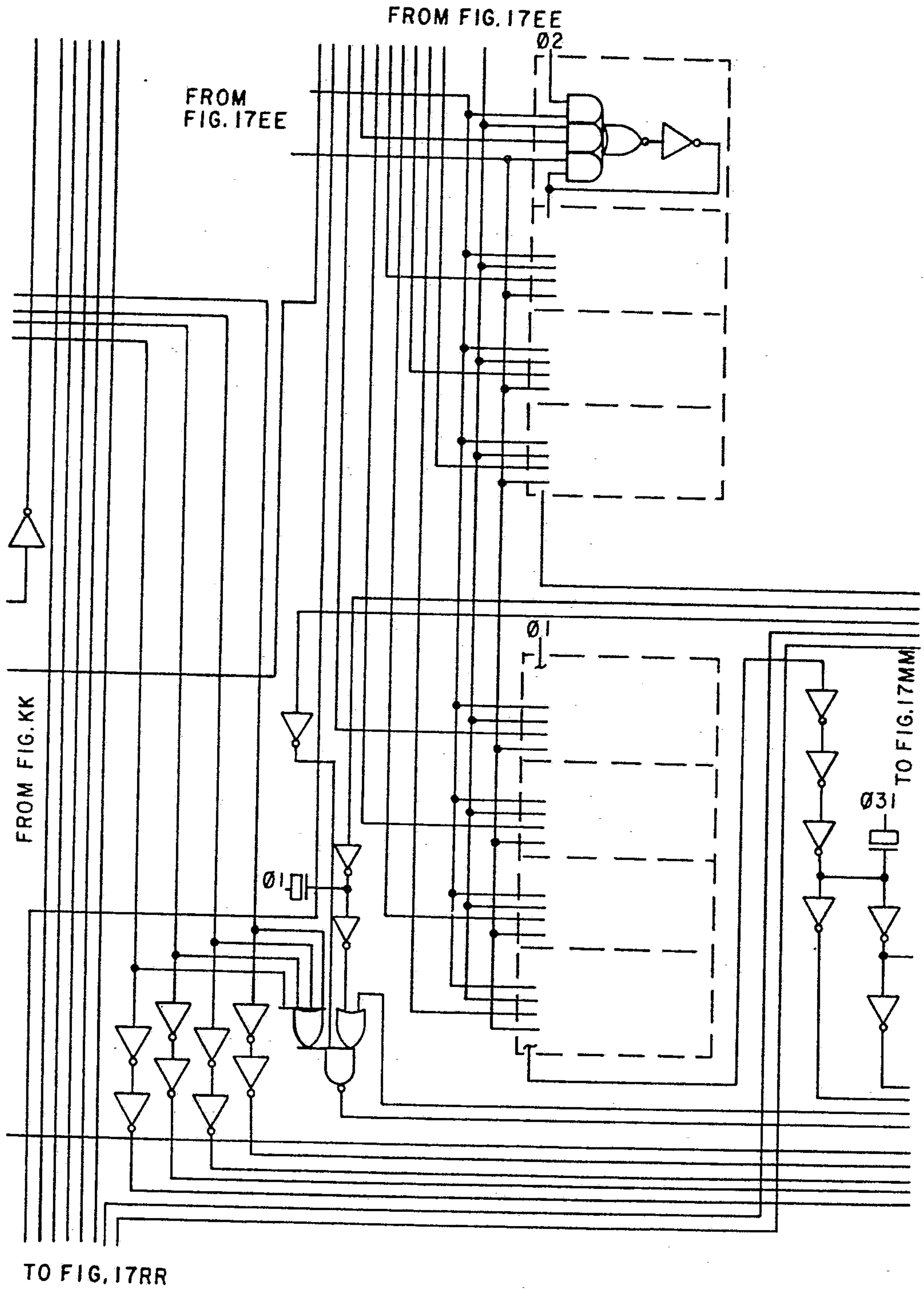

FROM FIG. 17EE
Ø2
FROM
FIG. 17EE
FROM FIG. KK
Ø1
Ø1
TO FIG. 17MM
Ø31
TO FIG. 17RR

Fig. 17LL

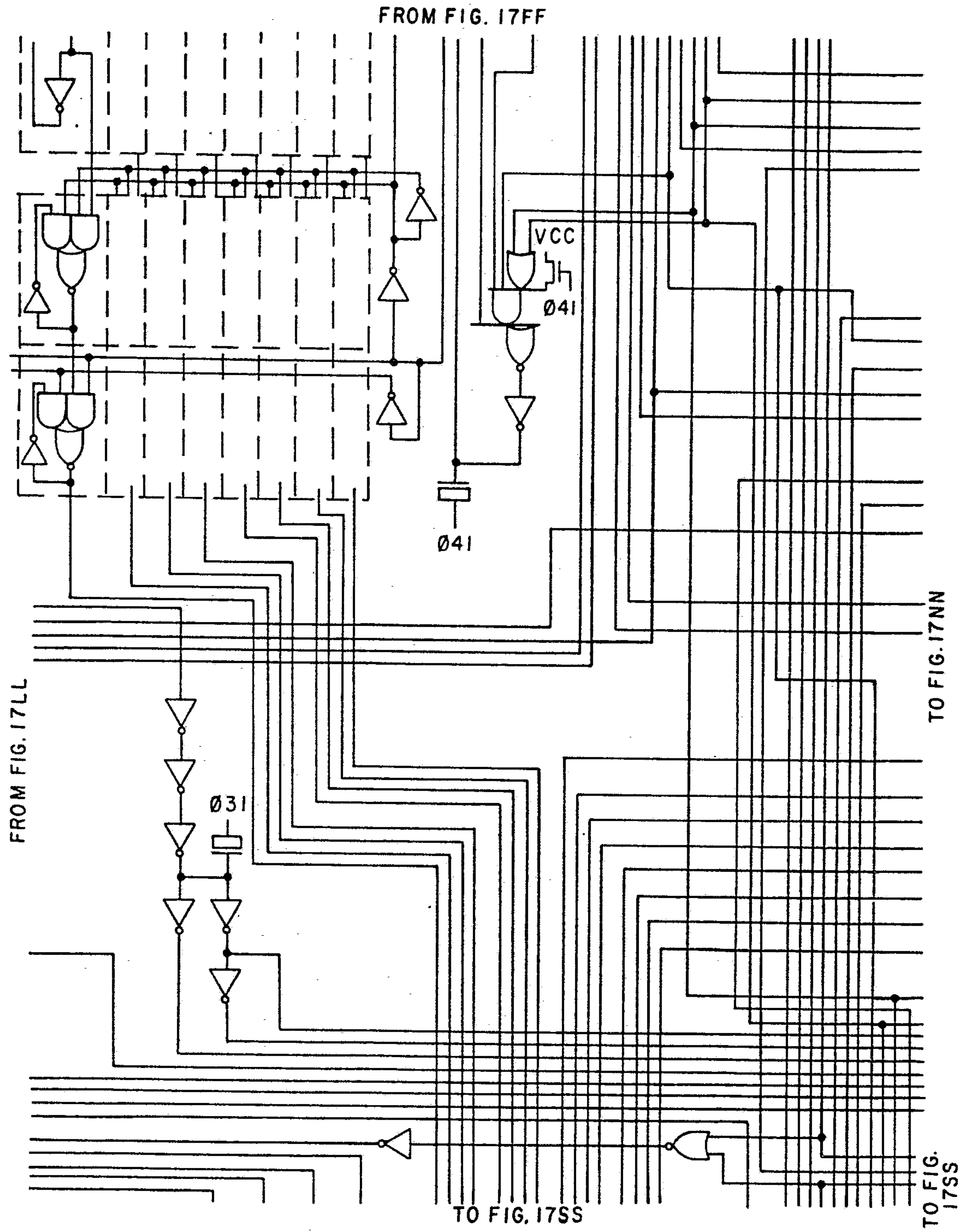

FROM FIG. 17FF
VCC
Ø41
Ø41
Ø31
FROM FIG. 17LL
TO FIG. 17NN
TO FIG. 17SS
TO FIG. 17SS

Fig. 17mm

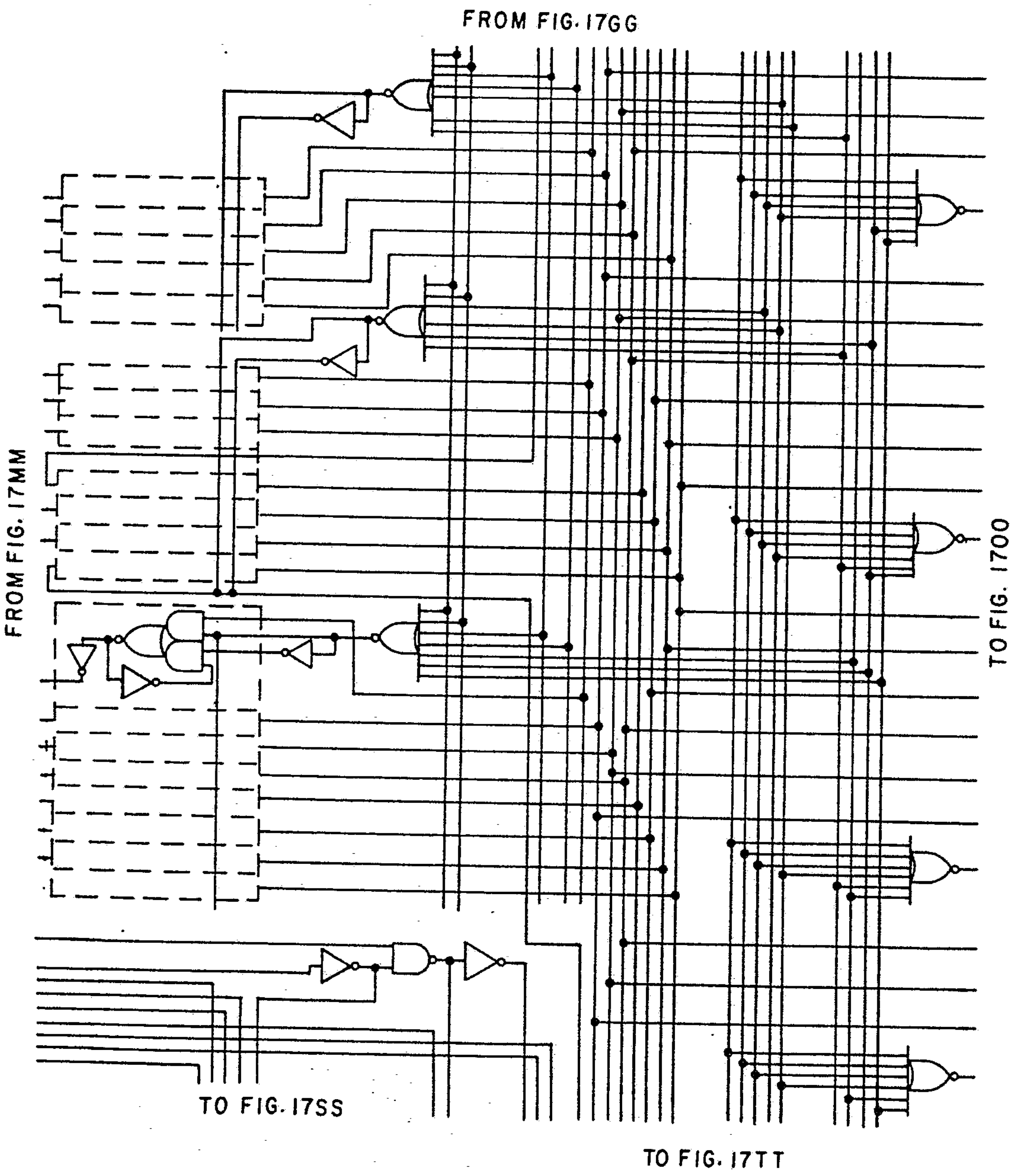



Fig.17nn

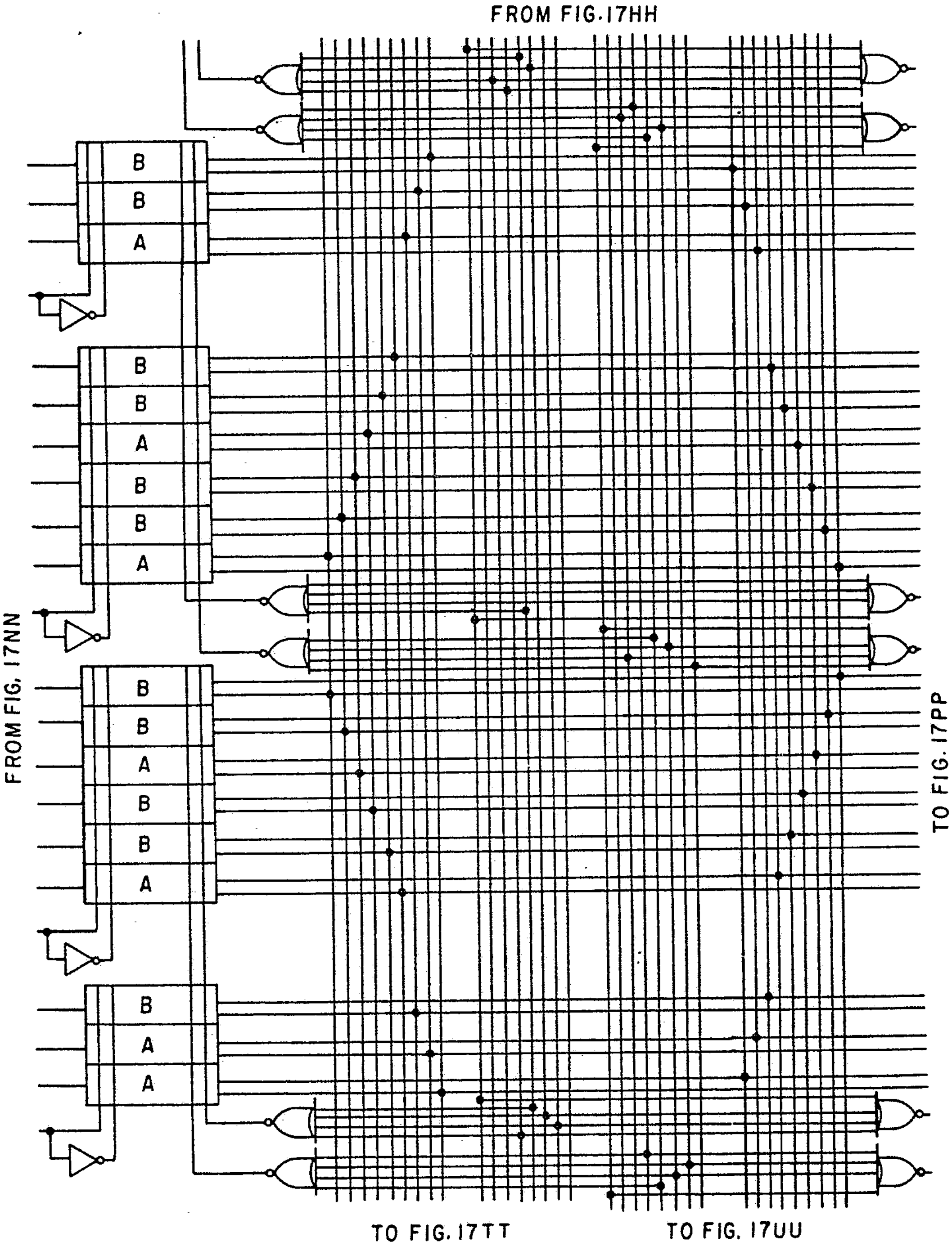

FROM FIG. 17HH
B
B
A
B
B
A
B
B
A
FROM FIG. 17NN
B
B
A
B
B
A
TO FIG. 17PP
B
A
A
TO FIG. 17TT
TO FIG. 17UU

Fig. 17oo

0165665

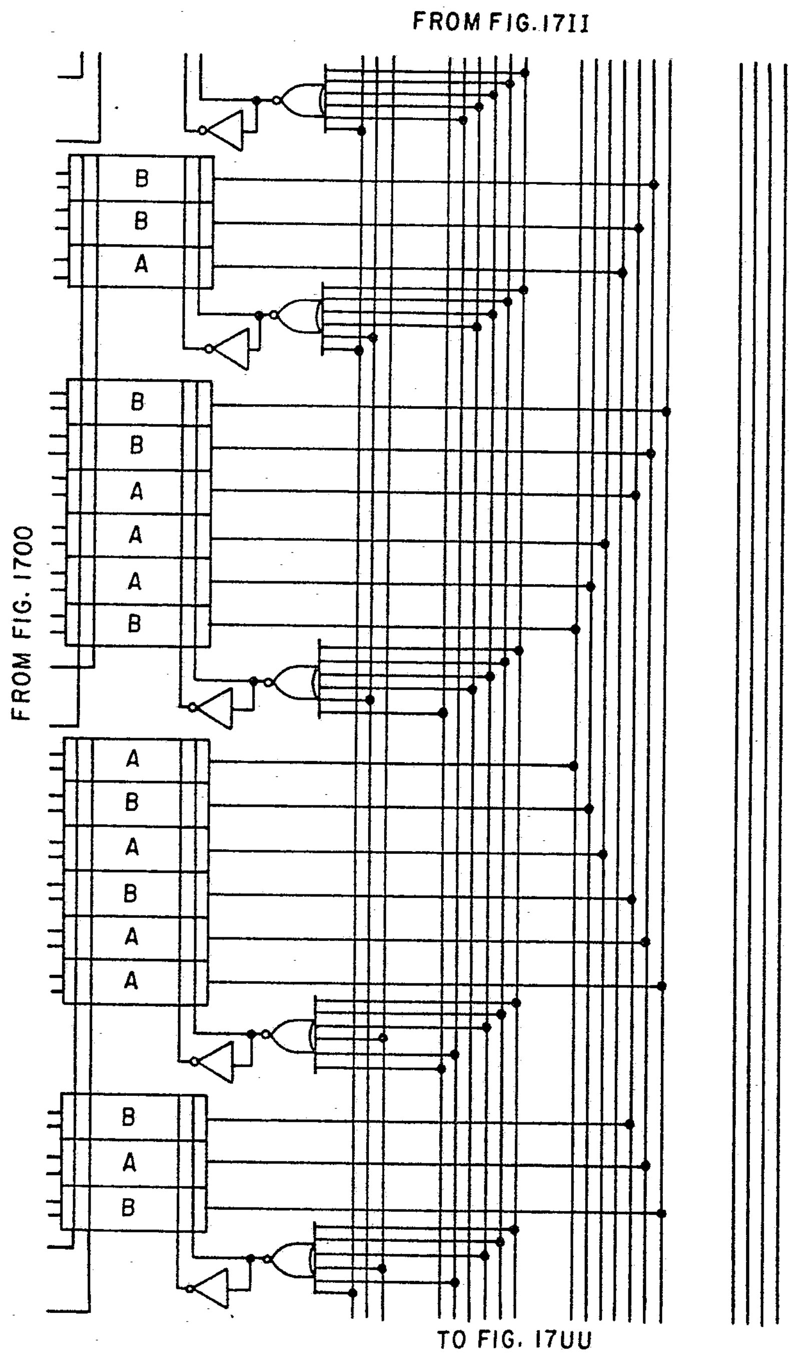



Fig.17pp

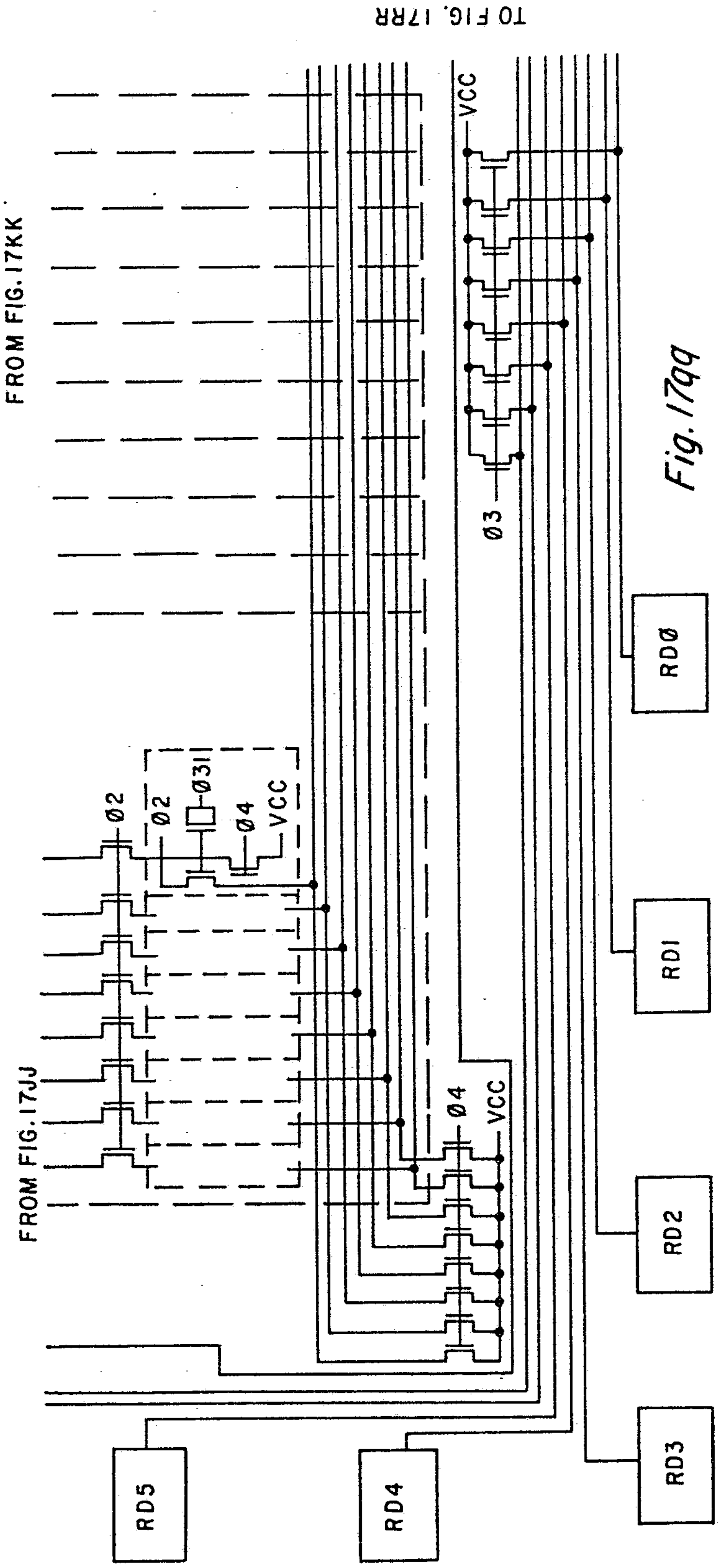

TO FIG. 17RR
FROM FIG. 17KK
VCC
Ø3
RDØ
RD1
RD2
RD3
RD4
RD5
Ø2
Ø2
Ø31
Ø4
VCC
Ø4
VCC
FROM FIG. 17JJ

Fig. 17qq

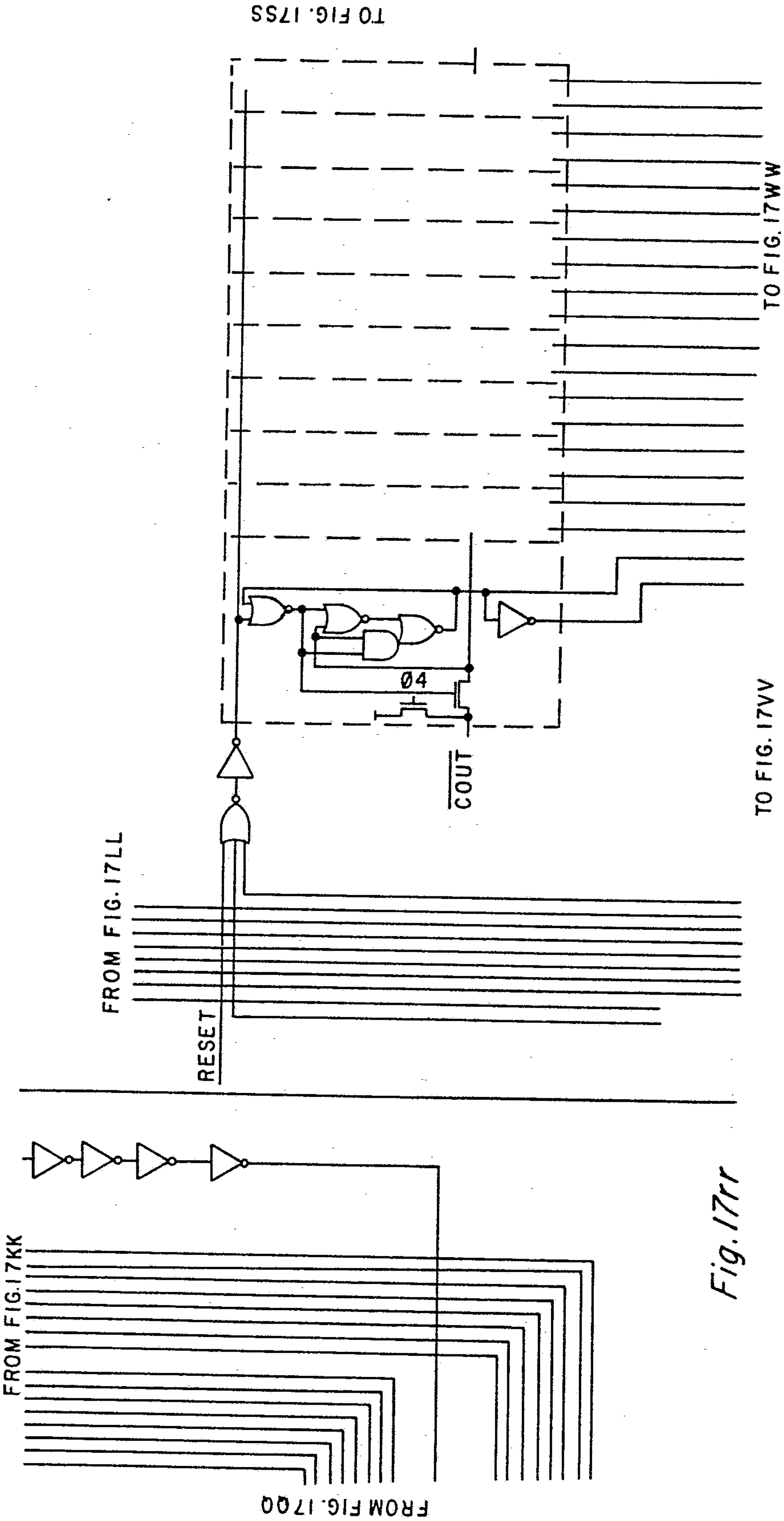

TO FIG. 17SS
TO FIG. 17WW
Ø4
COUT
TO FIG. 17VV
FROM FIG. 17LL
RESET
FROM FIG. 17KK
FROM FIG. 17OO

Fig. 17rr

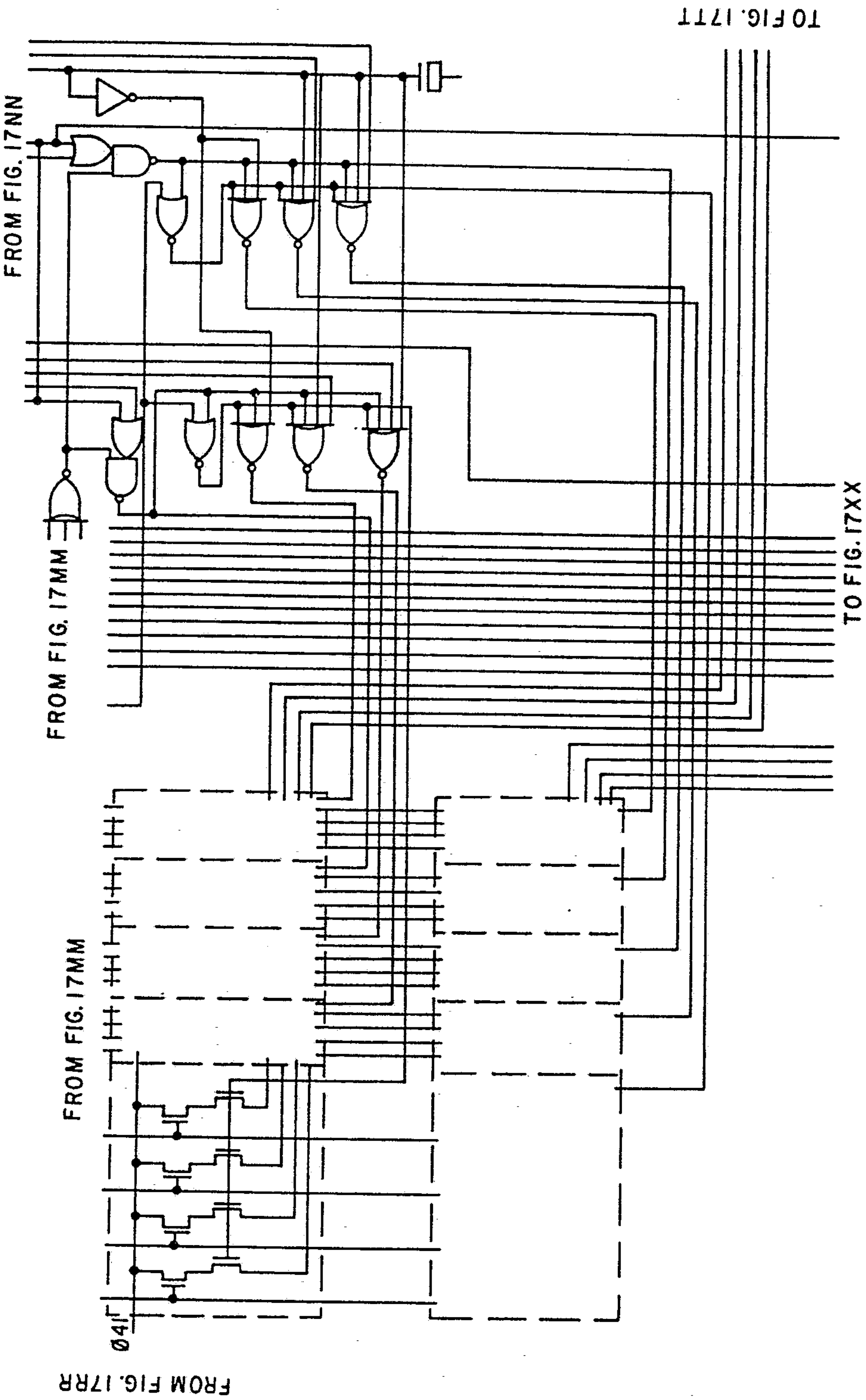

TO FIG. 17TT
FROM FIG. 17NN
FROM FIG. 17MM
TO FIG. 17XX
FROM FIG. 17MM
Ø4
FROM FIG. 17RR

Fig. 17ss

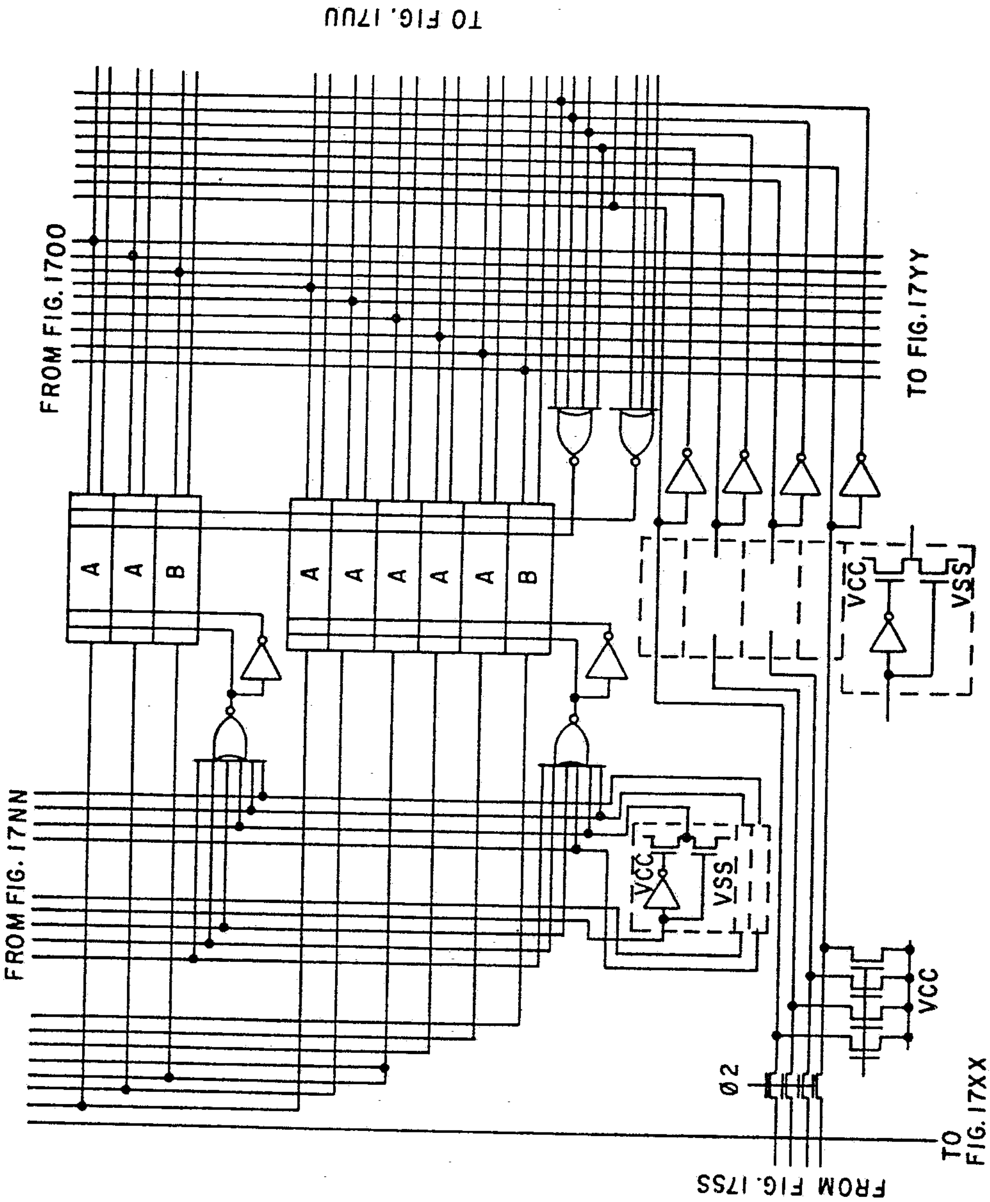

TO FIG. 17UU
FROM FIG. 17OO
TO FIG. 17YY
A
A
B
A
A
A
A
A
B
VCC
VSS
FROM FIG. 17NN
VCC
VSS
VCC
Ø2
TO FIG. 17XX
FROM FIG. 17SS

Fig. 17tt

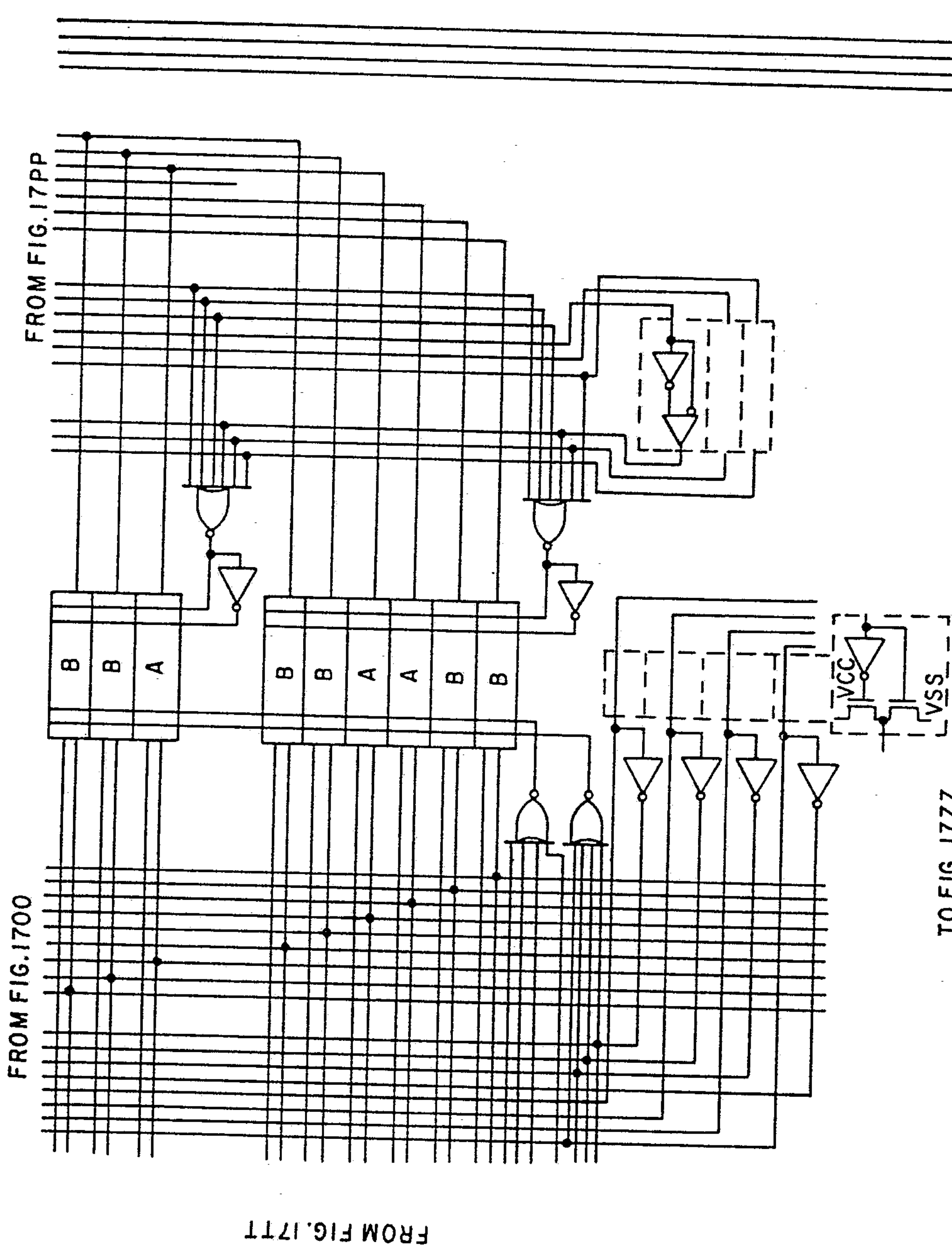

FROM FIG.17PP
B
B
A
B
B
A
A
B
B
VCC
VSS
FROM FIG.17OO
TO FIG.17ZZ
FROM FIG.17TT

Fig. 17uu

0165665

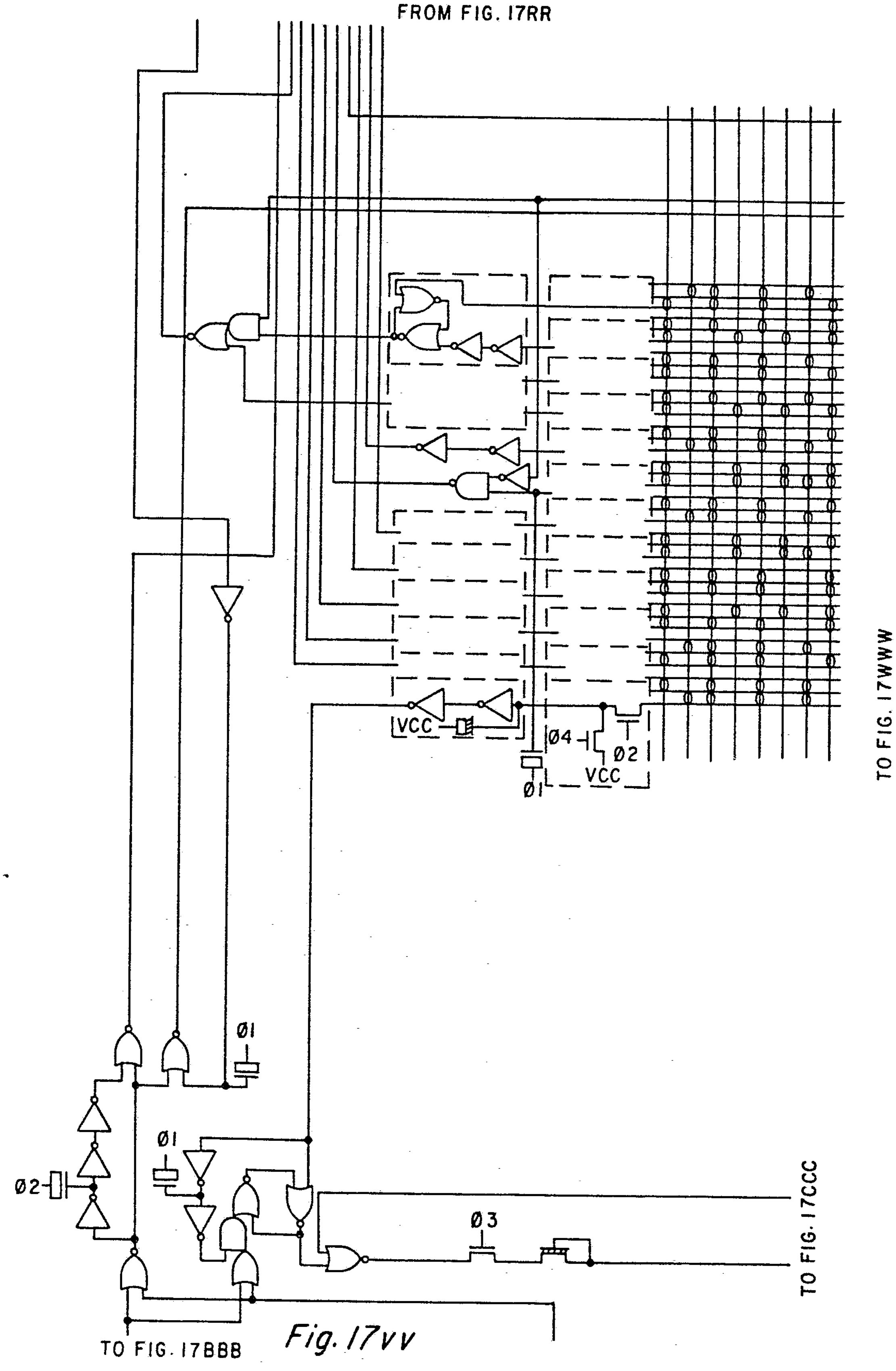



Fig. 17vv

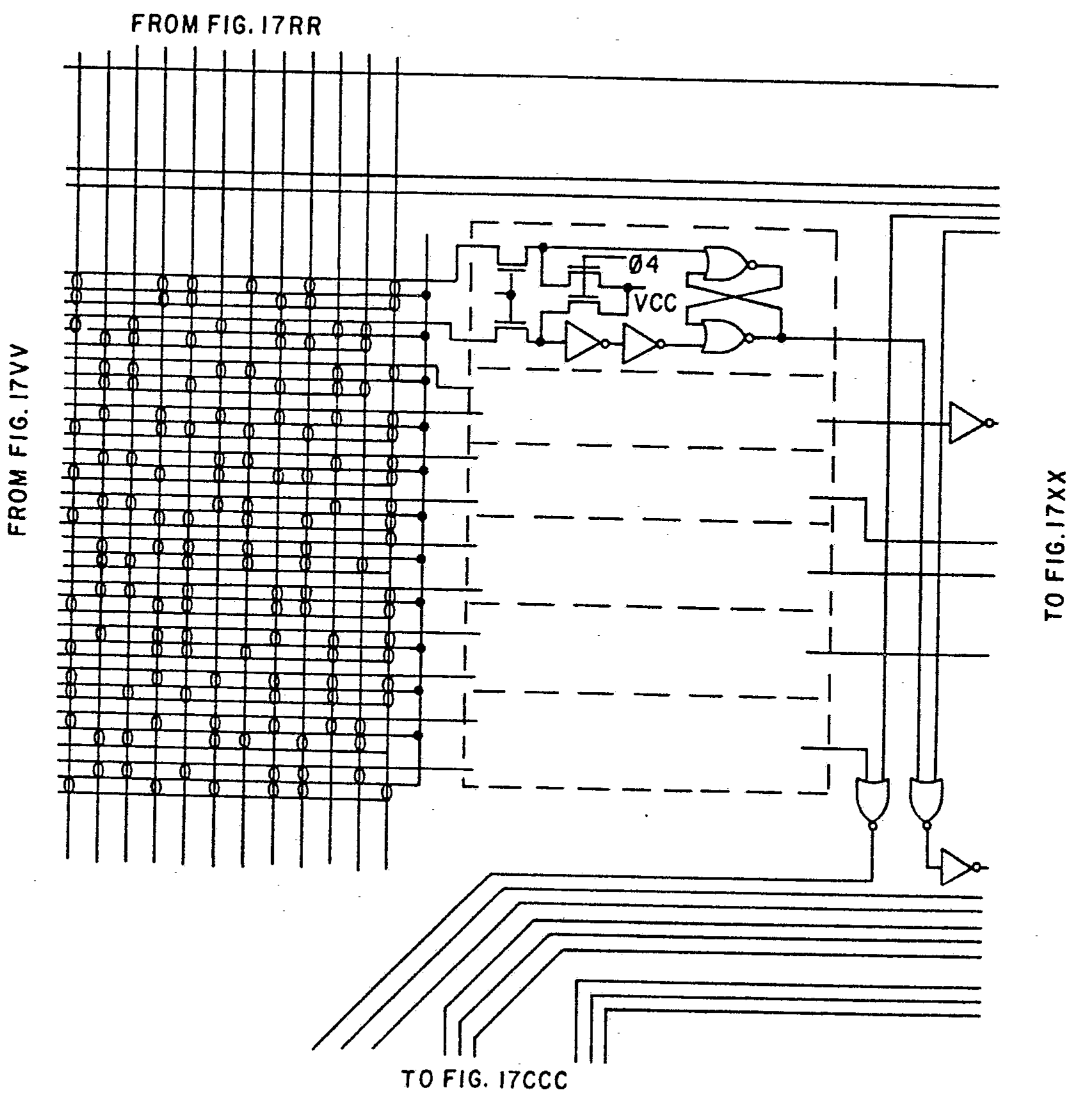



Fig. 17ww

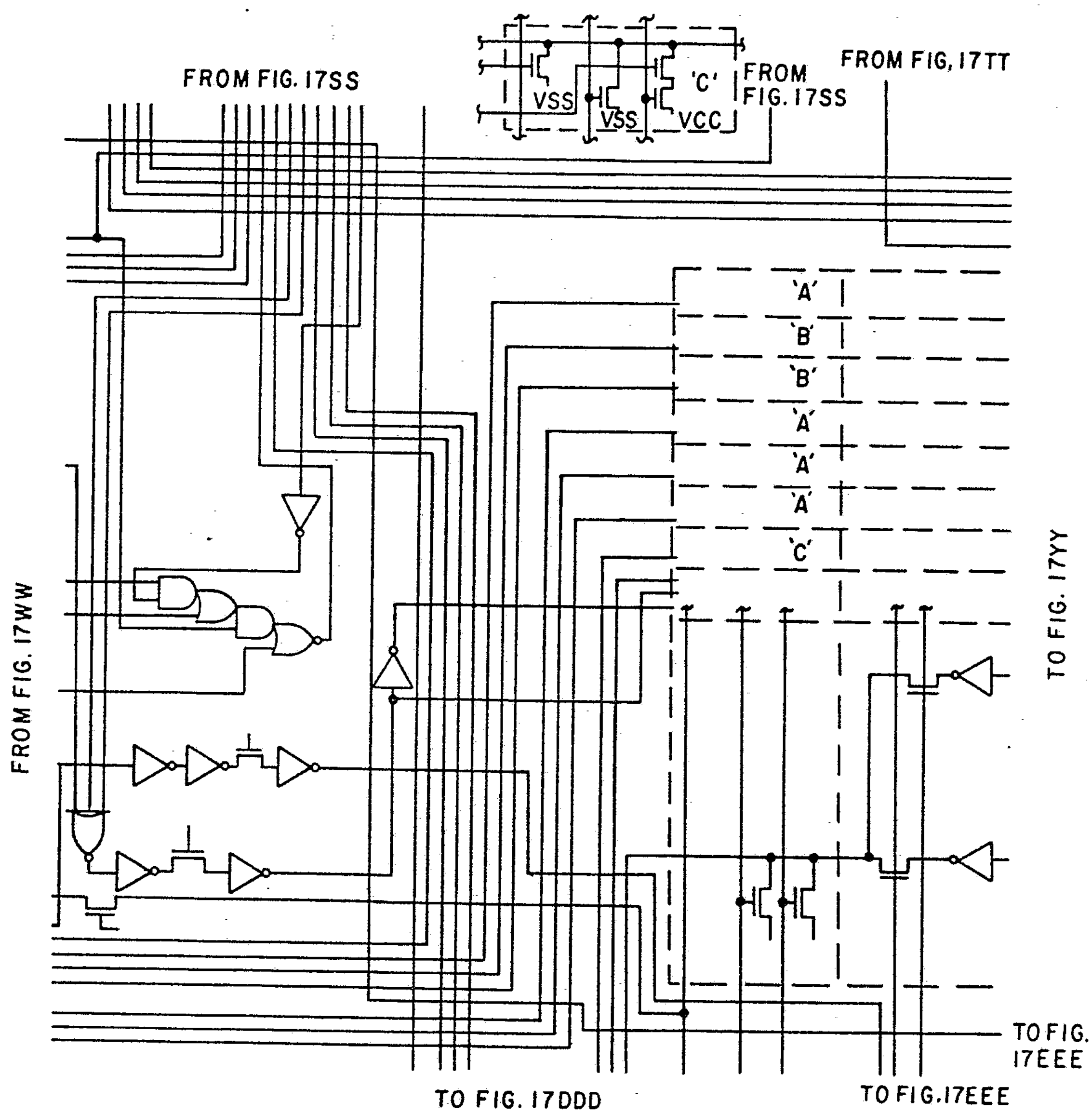

FROM FIG. 17SS
VSS
VSS
'C'
VCC
FROM FIG. 17SS
FROM FIG, 17TT
'A'
'B'
'B'
'A'
'A'
'A'
'C'
FROM FIG. 17WW
TO FIG. 17YY
TO FIG. 17EEE
TO FIG. 17DDD
TO FIG. 17EEE

Fig. 17xx

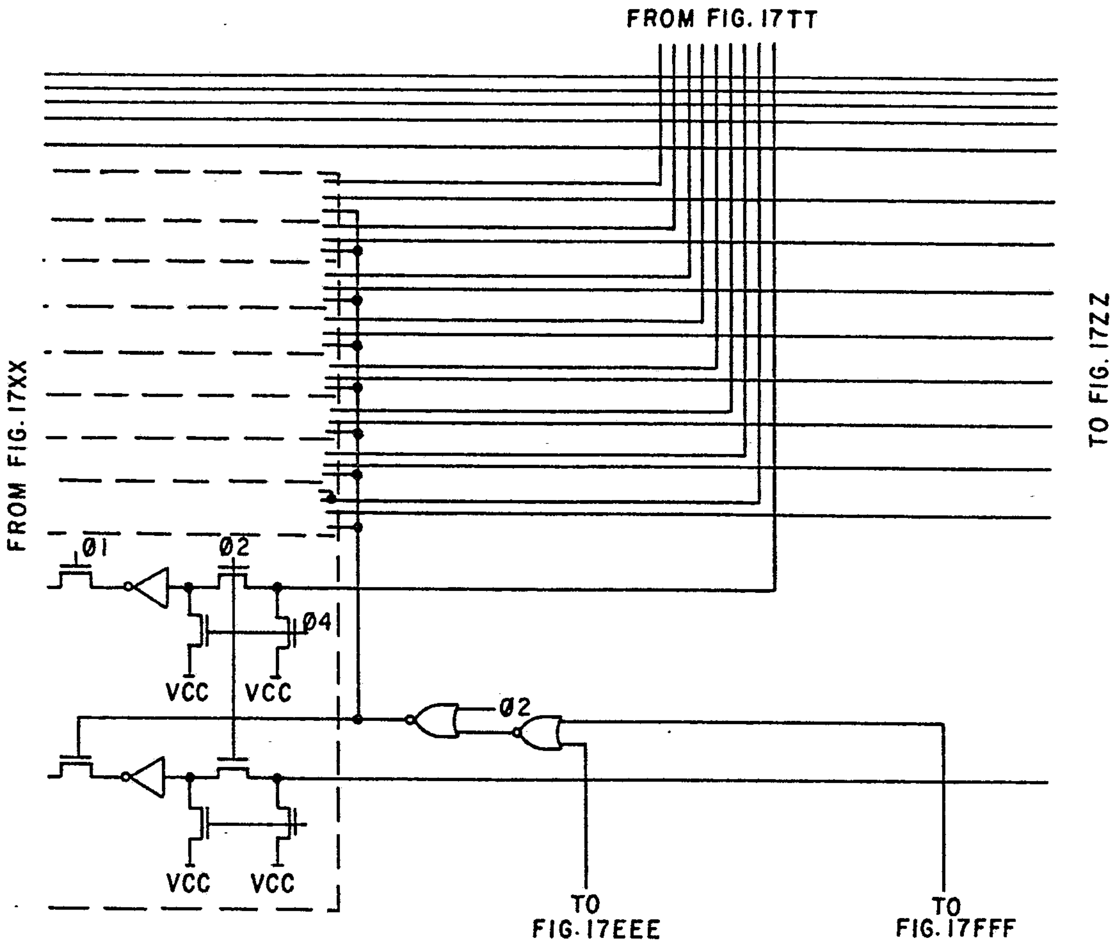

FROM FIG. 17TT
FROM FIG. 17XX
TO FIG. 17ZZ
Ø1
Ø2
Ø4
VCC
VCC
Ø2
VCC
VCC
TO
FIG. 17EEE
TO
FIG. 17FFF

Fig. 17yy

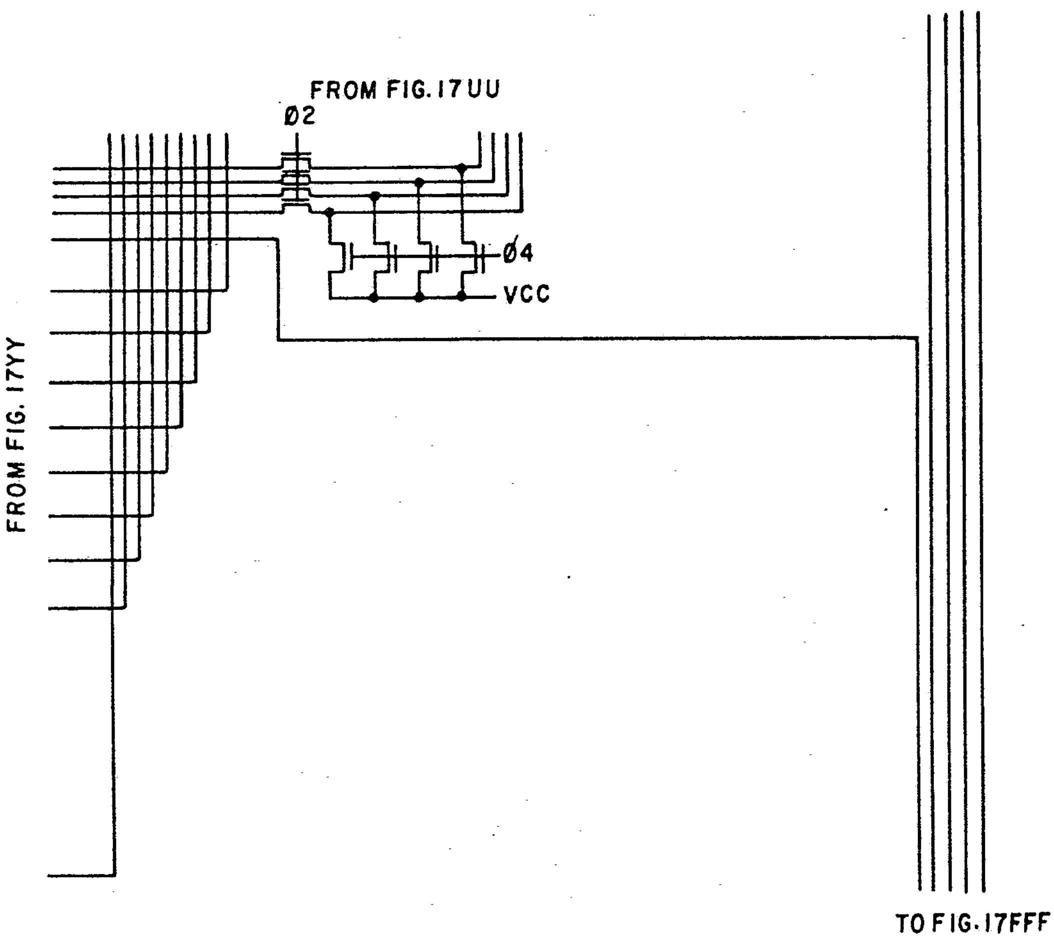



Fig. 17zz

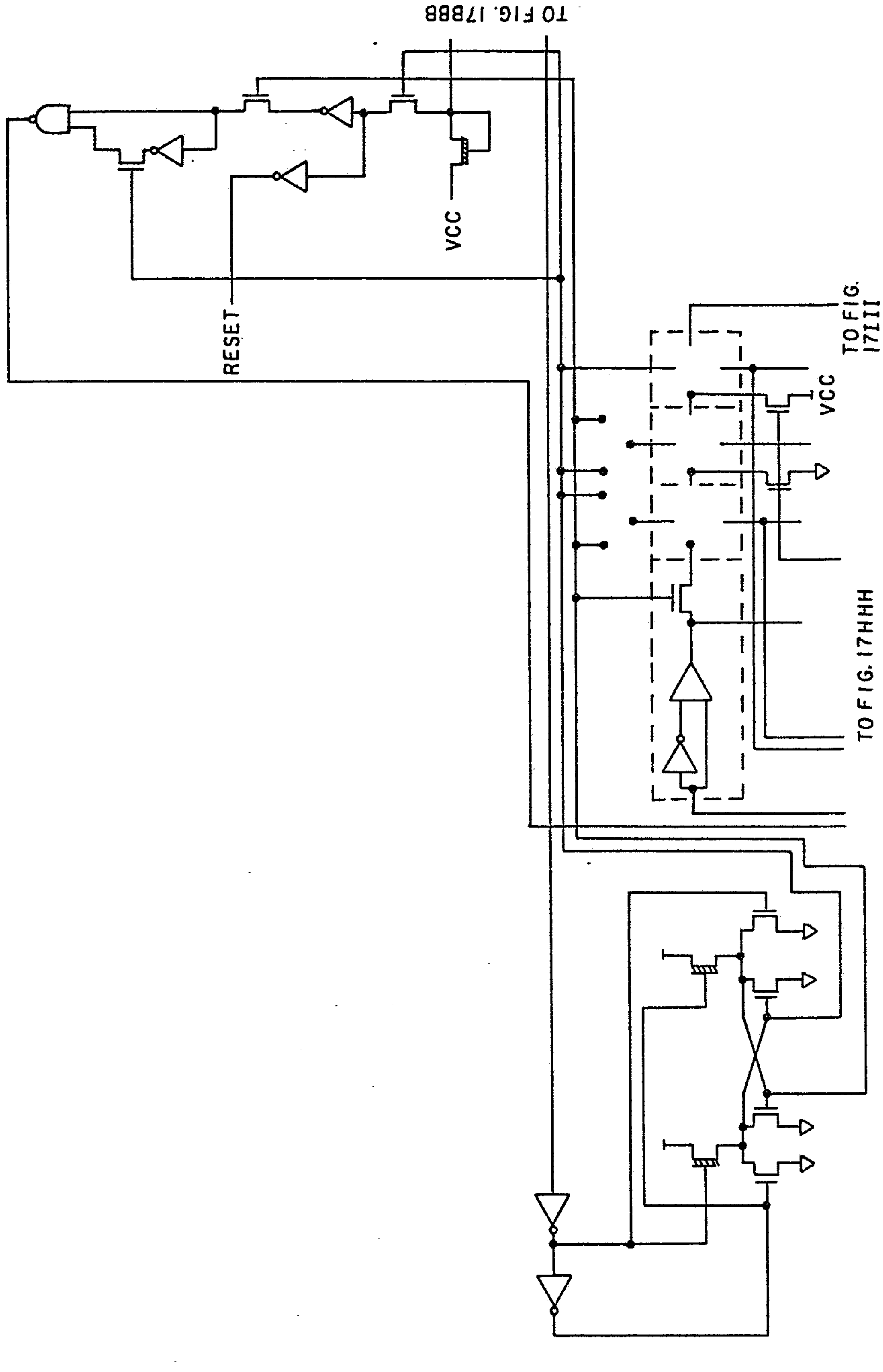



Fig. 17aaa

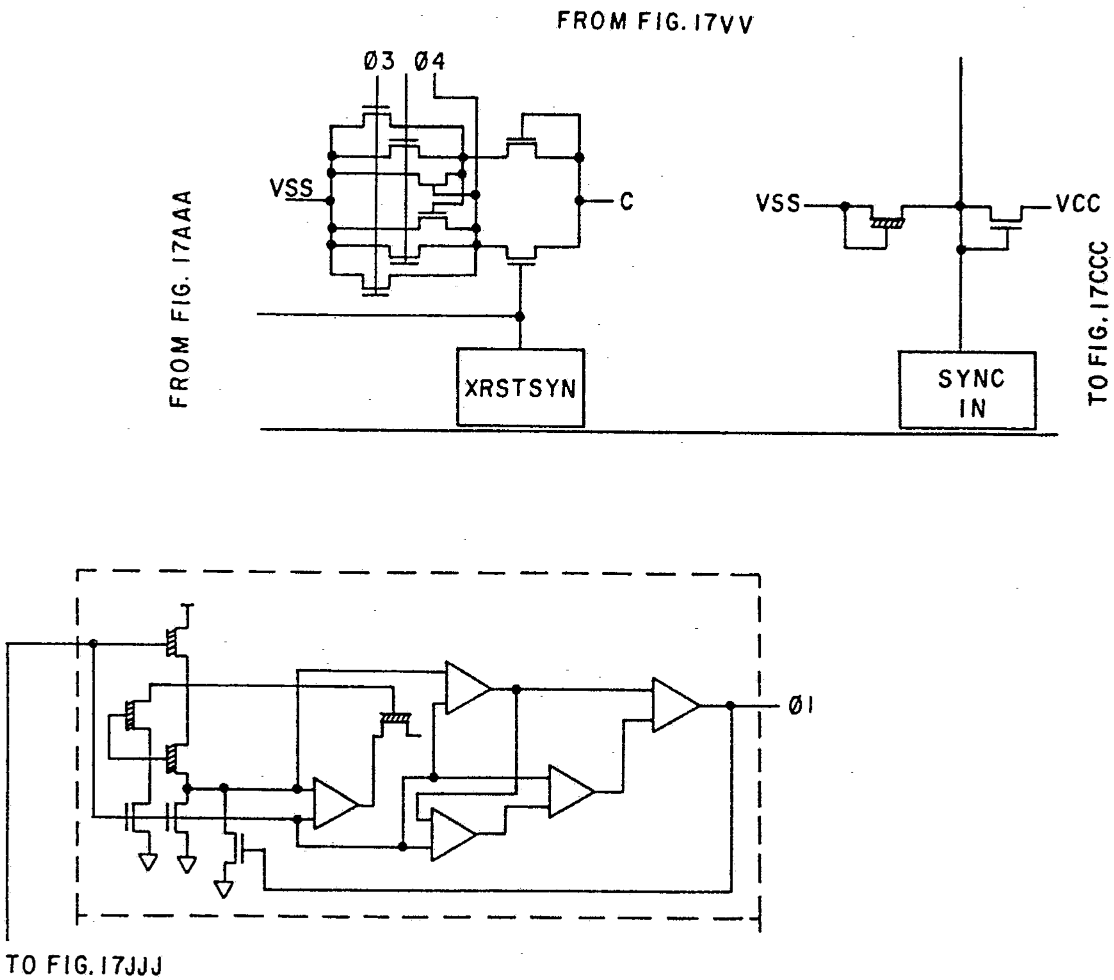

FROM FIG. 17VV
Ø3 Ø4
VSS
C
VSS
VCC
FROM FIG. 17AAA
TO FIG. 17CCC
XRSTSYN
SYNC
IN
Ø1
TO FIG. 17JJJ

Fig. 17bbb

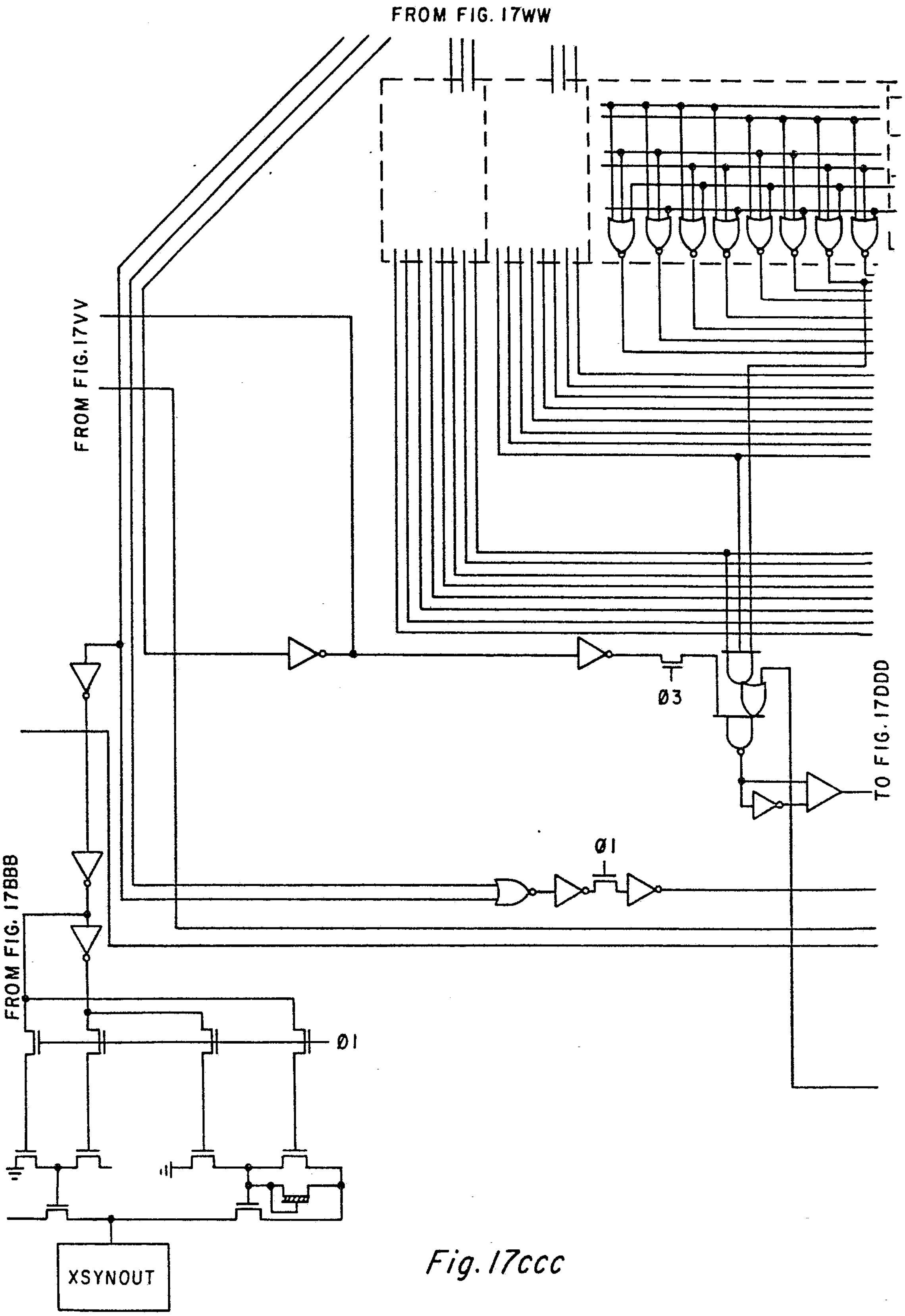

FROM FIG. 17WW
FROM FIG. 17VV
Ø3
TO FIG. 17DDD
Ø1
FROM FIG. 17BBB
Ø1
XSYNOUT

Fig. 17ccc

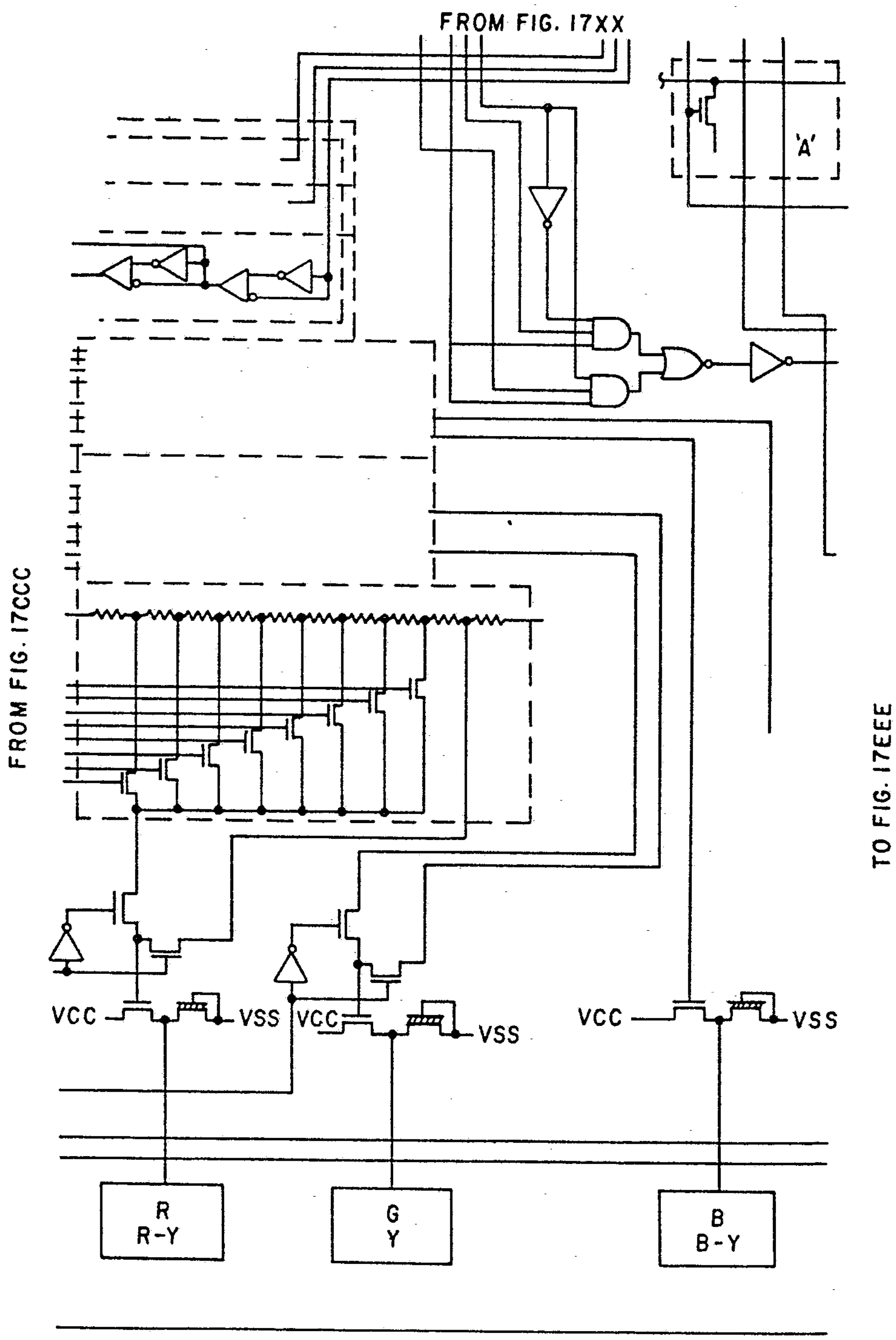

FROM FIG. 17XX
'A'
FROM FIG. 17CCC
TO FIG. 17EEE
VCC
VSS
VCC
VSS
VCC
VSS
R
R-Y
G
Y
B
B-Y

Fig. 17ddd

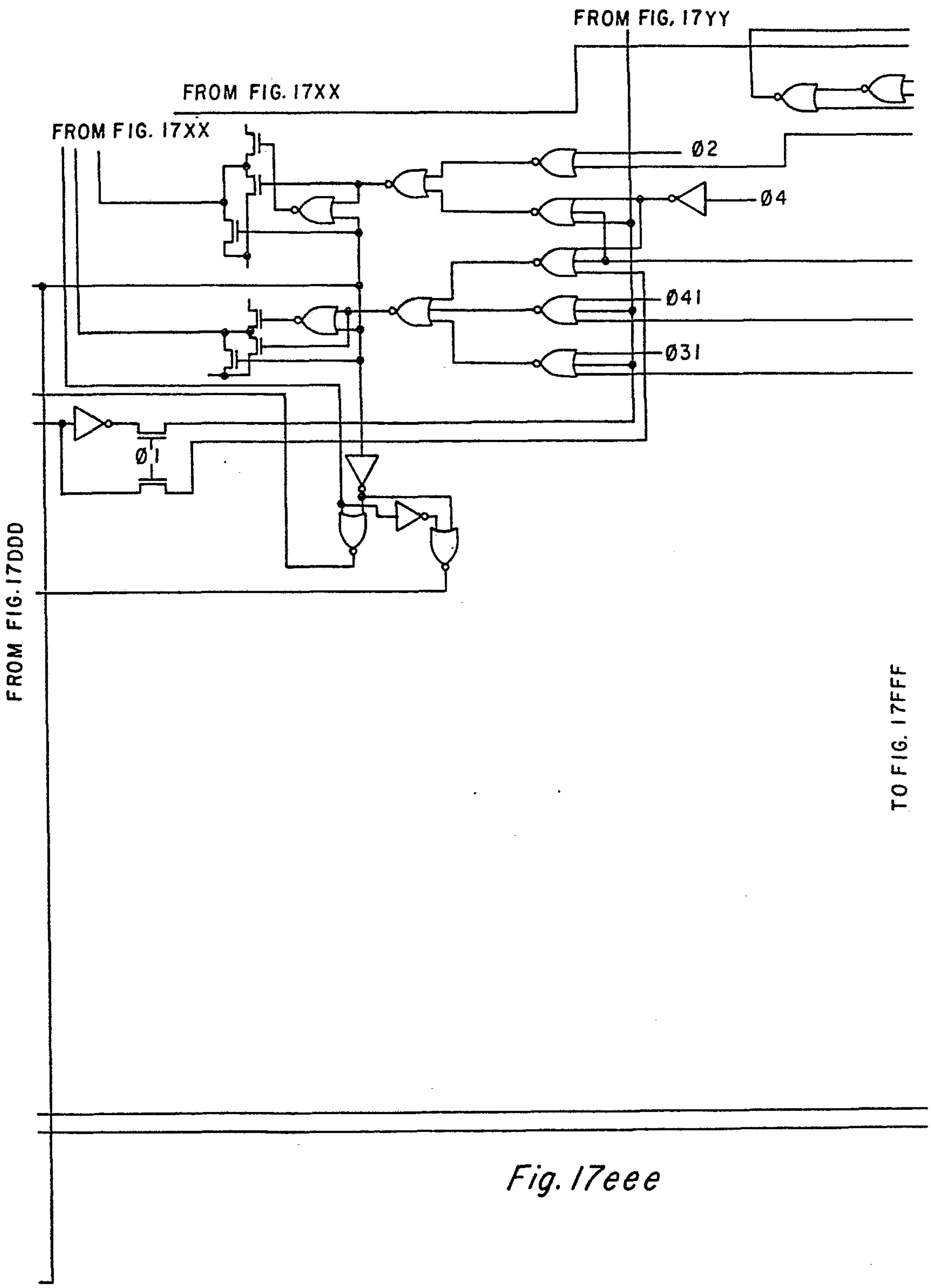

FROM FIG. 17YY
FROM FIG. 17XX
FROM FIG. 17XX
Ø2
Ø4
Ø41
Ø31
Ø1
FROM FIG. 17DDD
TO FIG. 17FFF

Fig. 17eee

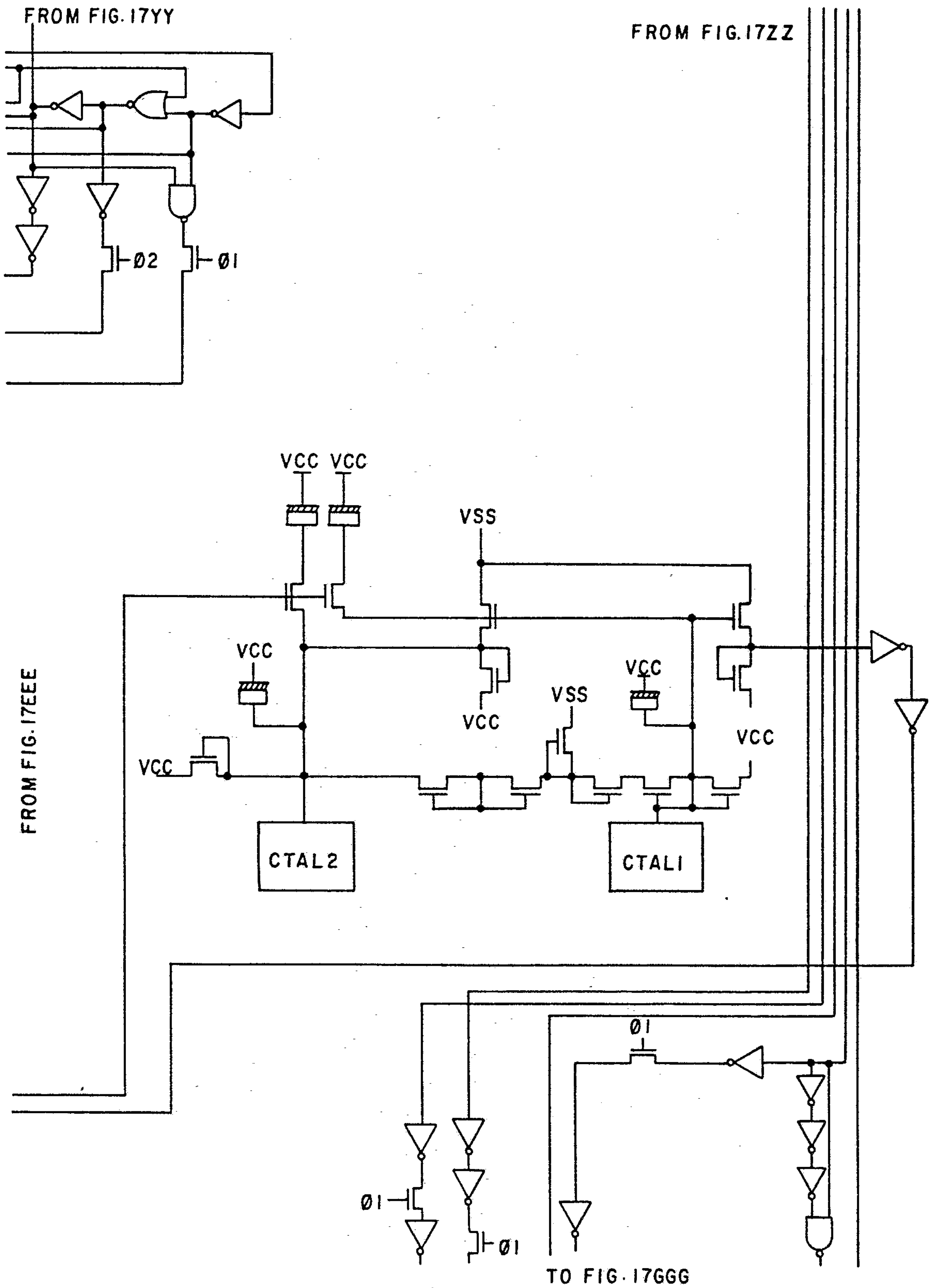

FROM FIG. 17YY
FROM FIG. 17ZZ
Ø2
Ø1
VCC
VCC
VSS
VCC
VCC
VSS
VCC
VCC
VCC
CTAL2
CTAL1
FROM FIG. 17EEE
Ø1
Ø1
Ø1
TO FIG. 17GGG

Fig. 17fff

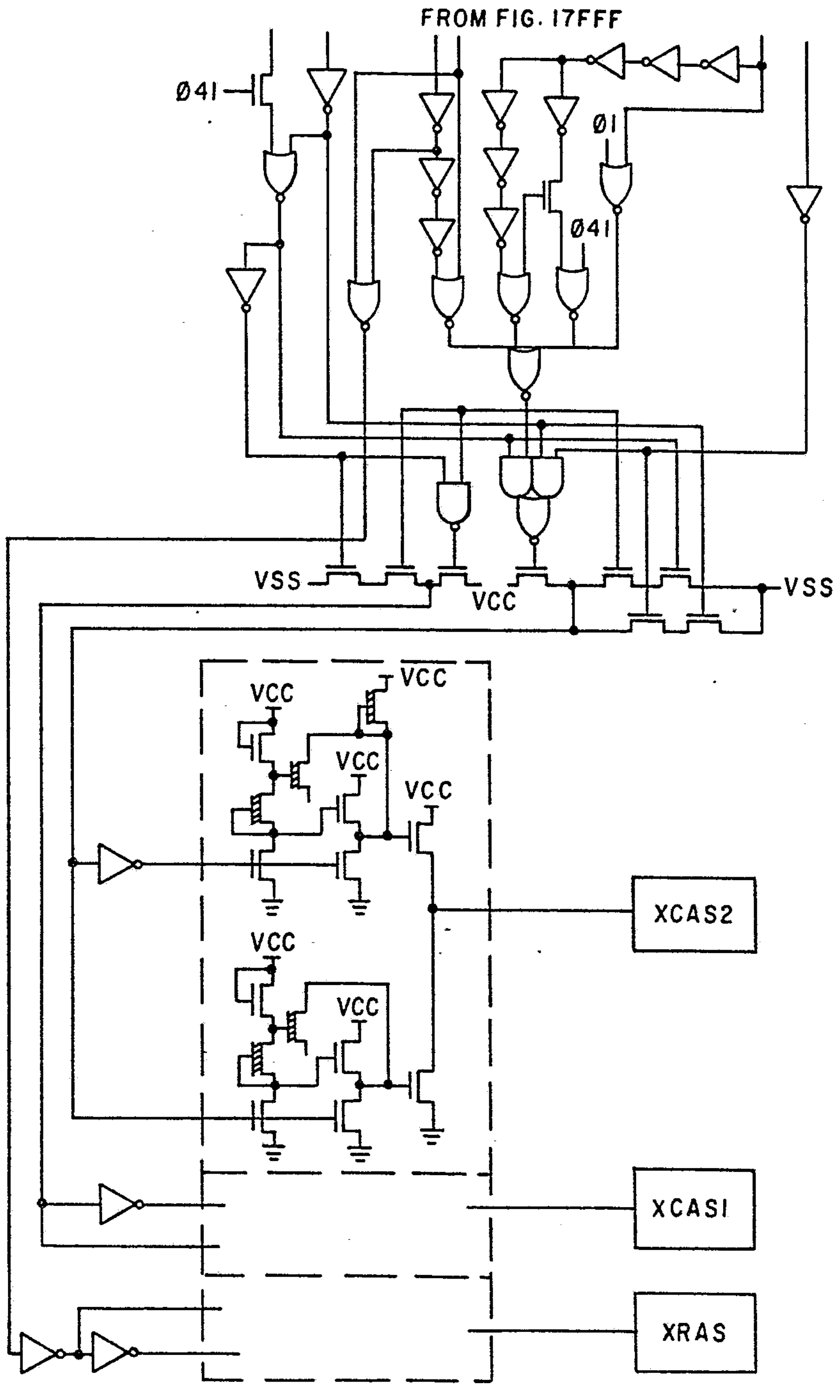

FROM FIG. 17FFF
Ø41
Ø1
Ø41
VSS
VCC
VSS
VCC
VCC
VCC
VCC
VCC
VCC
XCAS2
XCAS1
XRAS

Fig. 17ggg

0165665

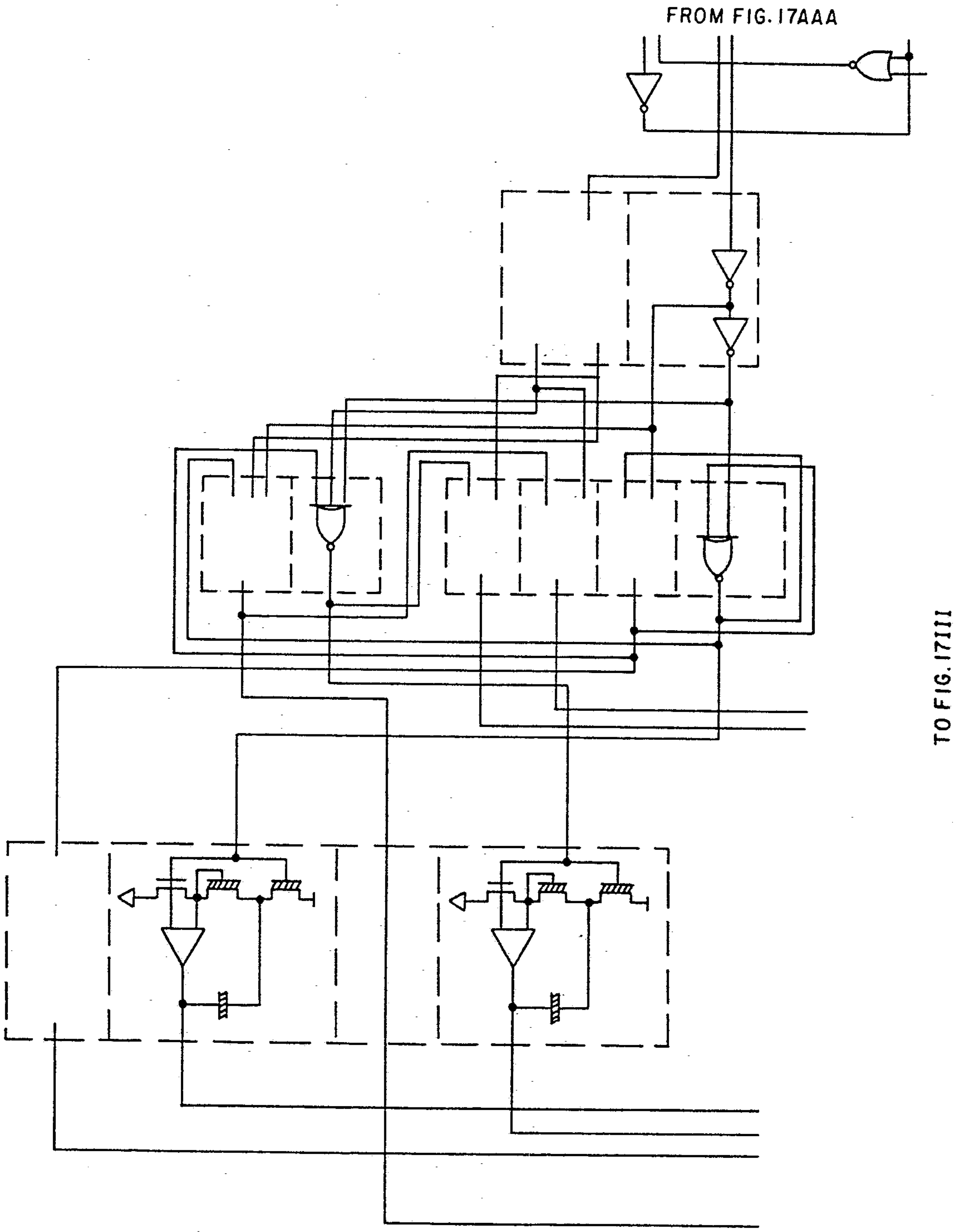



Fig. 17hhh

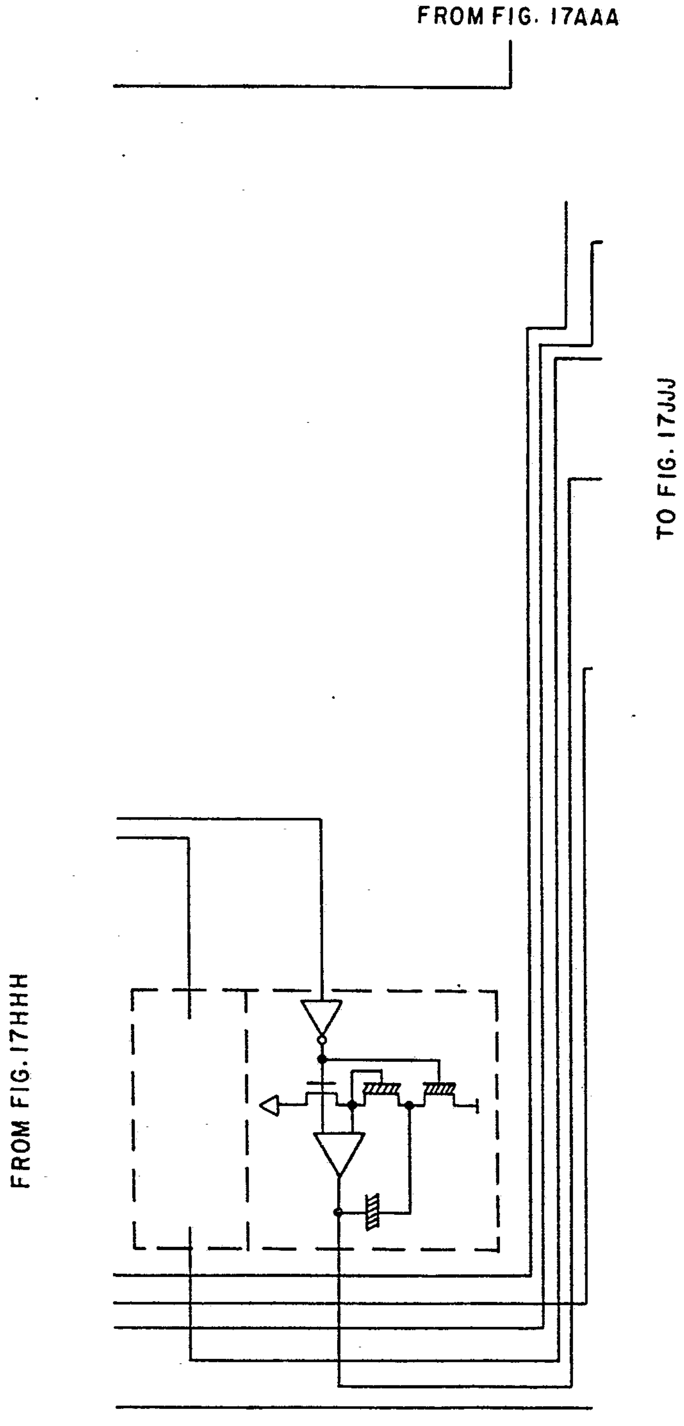



Fig. 17iii

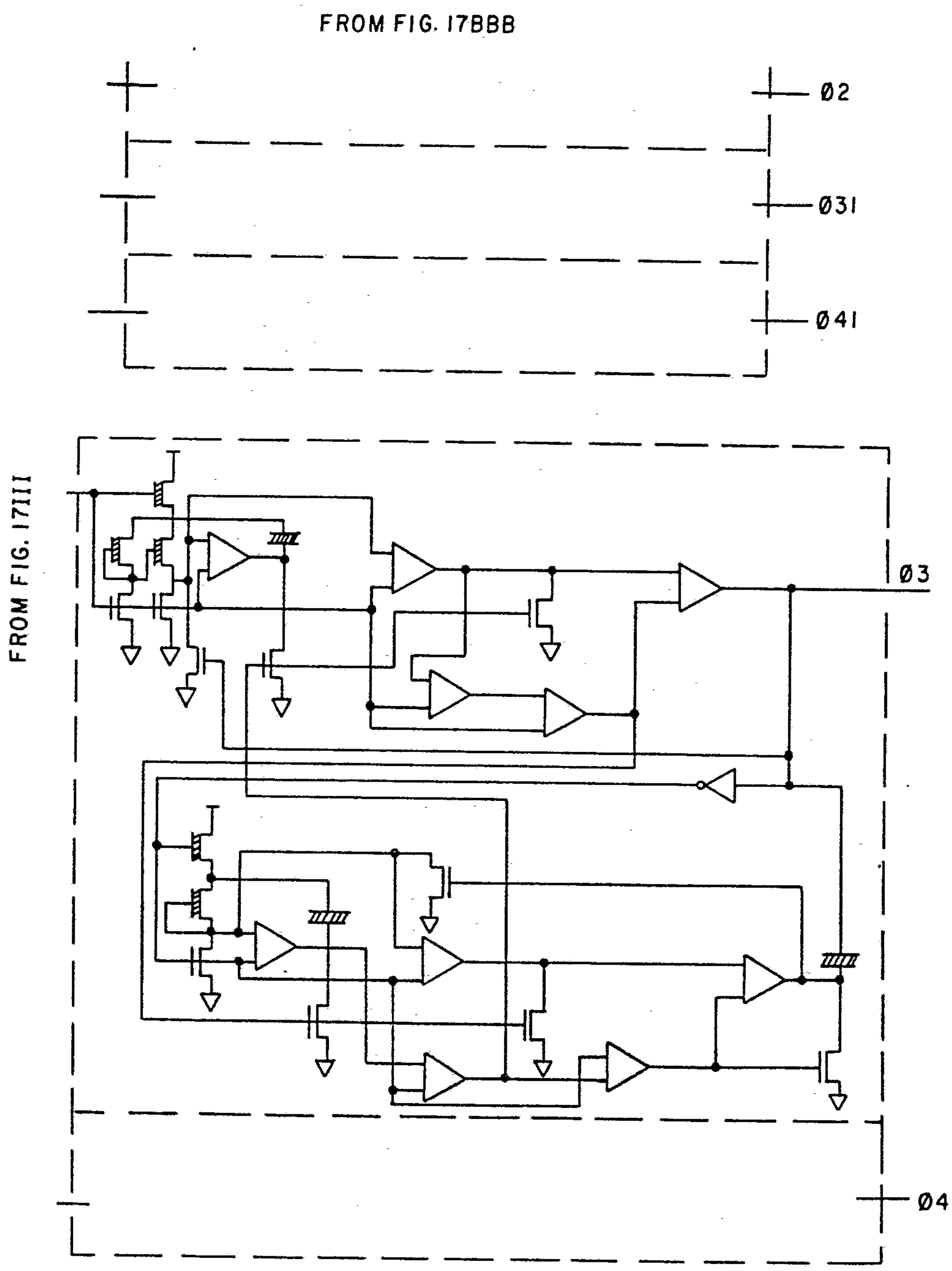



Fig. 17jjj