PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : H03K 19/173

A1

(11) International Publication Number: WO 94/06211

(43) International Publication Date: 17 March 1994 (17.03.94)

(21) International Application Number: PCT/US93/07212

(22) International Filing Date: 30 July 1993 (30.07.93)

(30) Priority data:
940,125 3 September 1992 (03.09.92) US

(71) Applicant: XILINX, INC. [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US).

(72) Inventors: PARLOUR, David, B. ; 4231 Parkman Avenue, Pittsburgh, PA 15213 (US). GOETTING, F., Erich ; 11659 Olive Spring Court, Cupertino, CA 95014 (US). TRIMBERGER, Stephen, M. ; 14575 Ramstad Drive, San Jose, CA 95127 (US).

(74) Agents: YOUNG, Edel, M. et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).

(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: ADAPTIVE PROGRAMMING METHOD FOR ANTIFUSE TECHNOLOGY

(57) Abstract

For antifuse programmable integrated circuit devices, in particular FPGA devices, the invention allows for alternative routing around antifuses (A5) which fail to program. The chip architecture includes wiring segments (Hn, Vn) and antifuses (An) which together allow for alternative routes around every antifuse (An) in the event of failure of that antifuse. The method includes programming the device under control of a computer which can recalculate routes in the event of an antifuse (i.e. A5) which fails to program. Preferably the initial routing distributes unused wiring segments (i.e. H4, V5) through the chip to be available for routing around a failed antifuse (A5). When a failure occurs, the method includes determining an alternative route around every failed antifuse. The alternative route may be established directly after the antifuse has failed or after all initially selected antifuses have been programmed. The method also includes swapping of logic cell inputs (signal 1, signal 2), logic cells (cell n), and/or logic blocks (Bn) from their original layout to adapt to a failed antifuse without changing the timing of signals which would have used the failed antifuse.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	MR	Mauritania
AU	Australia	GA	Gabon	MW	Malawi
BB	Barbados	GB	United Kingdom	NE	Niger
BE	Belgium	GN	Guinea	NL	Netherlands
BF	Burkina Faso	GR	Greece	NO	Norway
BG	Bulgaria	HU	Hungary	NZ	New Zealand
BJ	Benin	IE	Ireland	PL	Poland
BR	Brazil	IT	Italy	PT	Portugal
BY	Belarus	JP	Japan	RO	Romania
CA	Canada	KP	Democratic People's Republic of Korea	RU	Russian Federation
CF	Central African Republic			SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovak Republic
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	UA	Ukraine
DE	Germany	MG	Madagascar	US	United States of America
DK	Denmark	ML	Mali	UZ	Uzbekistan
ES	Spain	MN	Mongolia	VN	Viet Nam
FI	Finland				

ADAPTIVE PROGRAMMING METHOD FOR ANTIFUSE TECHNOLOGY

FIELD OF THE INVENTION

The present invention relates to programming logic devices, more particularly to programming field programmable devices having a programmable interconnect structure interconnectable by antifuses.

BACKGROUND OF THE INVENTION

FPGAs - SRAM and Antifuse

Field programmable gate arrays (FPGAs) comprise programmable logic blocks and a programmable interconnect structure for interconnecting the blocks. The logic blocks are programmed to perform a desired function and the interconnect structure comprises wiring segments which may be interconnected to connect the logic blocks together as desired. Some interconnect structures are interconnected by turning on transistors which interconnect the wiring segments. Others are interconnected by programming antifuses which interconnect the wiring segments. Antifuse programming is achieved by applying sufficiently different voltages to the wiring segments which contact the two terminals of the antifuse to cause current to pass through the antifuse and cause the antifuse to become permanently conductive.

Full Testability

When manufacturing FPGAs which are programmed by turning on transistors, it is possible to fully test the device before the device is sold to a customer who will program the device, to assure that all transistors operate properly. Thus the yield of devices which are successfully configured by the user is very high.

On the other hand, antifuse devices can only be tested for nonconductivity of the antifuses in their unprogrammed

state. Antifuse devices can not be tested for antifuse programmability before going to a customer, because antifuse devices are one-time programmable, and full testing (which includes determining whether each antifuse will program) would destroy the programmability the customer desires. Given the large number of antifuses that must be programmed in a typical FPGA, even relatively low antifuse failure rates can lead to unacceptably high FPGA device failure rates, since each connection in the device must be successfully completed for the device to work as designed.

Some manufacturers of antifuse devices include extra test antifuses, which are programmed at the factory to determine if the device antifuses will program properly. If any of these test antifuses fail to program, the entire part is rejected. However, these test antifuses do not guarantee that the device antifuses will all function properly. Sometimes all test antifuses may program properly and yet a device antifuse will fail, just due to statistical variation. Thus even with test antifuses, a manufacturer will sell some number of devices which fail to function properly, and the perceived quality of the manufacturer's product suffers accordingly. The effective cost to the user is also increased accordingly.

Typical Programming Method

A user typically enters a logic design into a computer using a schematic capture package or a hardware description language. The computer then proceeds through an elaborate set of steps to generate a list of transistors to turn on or antifuses to program (or both) in order to cause a particular field programmable logic device to implement the user's design. Typical steps for converting a user's logic design into a list of antifuses or transistors include

1) mapping the user's logic design into logic elements of a suitable FPGA device (called "technology mapping");

1 2) placing each portion of the user's logic into a
2 corresponding logic cell in the device; and
3 3) routing the signals on particular wiring segments to
4 interconnect the portions of the user's logic to
5 form the overall design.
6 Additional steps can also be performed, for example
7 optimizing the logic before and during the technology mapping
8 step, and iteratively improving the placement as well as the
9 routing. The computing power to perform the above steps is
10 considerable.
11 By contrast, the computing power needed to actually
12 program the logic device is relatively trivial. A personal
13 computer or work station (called the host) is typically used
14 for the mapping, placing and routing steps, and a simple
15 programming box attached to the computer, which includes a
16 simple microprocessor, typically receives data or
17 instructions from the computer or work station and applies
18 the voltages necessary to program the device.
19 A simple programming box can indicate to the host
20 computer if an antifuse did not program properly. In the
21 past, an indication that an antifuse did not program properly
22 has meant that the entire antifuse device failed and may not
23 be used.
24
25 Distribution of Programming Voltages and Acceptable Parts
26 Fig. 1 shows a curve of voltage distribution and ranges
27 over which devices are accepted and rejected. After the
28 devices are manufactured and before they leave the factory,
29 the devices are tested for a variety of purposes, including
30 whether any of the antifuses become programmed under the
31 highest operating voltage (for example 7 volts) for which the
32 device is rated. In a successful device, no antifuses will
33 program. If any antifuses do become programmed, the device
34 must be rejected. Those devices in which antifuses become
35 programmed below the maximum operating voltage are
36 illustrated in the shaded region labeled FACTORY REJECT.

1 These devices will not be sold to customers, and represent
2 lost profits but not lost reputation for reliability. When
3 devices are shipped to customers, they are programmed in the
4 field, at a programming voltage considerably higher than the
5 operating voltage. The programming voltage is limited by the
6 voltage at which programming transistors and other elements
7 will be destroyed, for example the transistor breakdown
8 voltage. Any device for which an antifuse was designated to
9 be programmed but failed to program will be rejected in the
10 field, and represents both lost reputation and lost profit,
11 either to the manufacturer or to the user. Devices in which
12 at least one antifuse failed to program are indicated by the
13 shaded region FIELD REJECT. The middle region, labeled GOOD,
14 shows a device in which all antifuses program at a voltage
15 higher than the operating voltage and lower than the
16 programming voltage which will be used.
17
18 Acceptable Yield
19 Traditional one-time programmable logic devices which
20 cost a few dollars have been considered acceptable if the
21 yield of successfully programmed devices is on the order of
22 95%. That is, if 95% of the devices shipped to a customer
23 fall into the category labeled GOOD. However, for an FPGA
24 device costing several hundred dollars, a customer is not
25 likely to be satisfied with a yield of successfully
26 programmed devices of only 95%. The customer is more likely
27 to require a yield of at least 99% successfully programmed
28 devices to consider the money well spent. In other words,
29 the customer is likely to require that no more than 1% of the
30 devices have antifuses which fail to program at the
31 programming voltage used. The manufacturer must cover the
32 cost of yield loss one way or another, usually by shipping
33 replacement devices or by reimbursing the user for failed
34 devices.
35
36

1 Achievable Yield
2 In a device having 700,000 antifuses, of which 2% or
3 14,000 are typically programmed for a design, a failure rate
4 of 1 antifuse per million produces a yield of about 98.6%. A
5 failure rate of 100 parts per million produces a yield of
6 only 24.7%. Some companies have been programming antifuse
7 chips at the factory according to customer specifications, in
8 order to not burden the user with the inconvenience of
9 handling failed devices and to maintain a reputation for
10 reliability. This procedure decreases the convenience of
11 field programmability, and increases the time required to
12 turn a design into a programmed device.
13 The statistics become quickly worse for larger devices.
14 In a larger device having 2.5 million antifuses of which 2%
15 or 50,000 will be programmed, a failure rate of 1 antifuse
16 per million produces a yield of about 95.1% and a failure
17 rate of 100 antifuses per million produces a yield of only
18 0.7%. These poor yields have prevented the manufacture of
19 large antifuse devices.
20
21 SUMMARY OF THE INVENTION
22 According to the present invention, the chip
23 architecture includes more wiring segments and antifuses than
24 are typically needed for routing of signals through the
25 interconnect structure. The software algorithms attempt to
26 preserve an even distribution of unused routing resources.
27 Antifuses and wiring segments not used in an initial routing
28 selection can be used for patches, that is, to route around
29 antifuses which fail to become conductive during programming.
30 Programming of antifuses is controlled by a computer which
31 can store the initially determined routing list and which can
32 run software which uses the initial routing list plus other
33 necessary information to revise the initial routing. Each
34 antifuse is tested after programming. If an antifuse fails
35 to become conductive upon programming, according to the
36 present invention, adaptive routing software selects an

1 alternate route to complete the connection which would have
2 used the failed antifuse. This alternative route may use the
3 initially unused wiring segments and antifuses, or may be
4 chosen by recalculating the routes for all paths not yet
5 routed.
6 As mentioned earlier, simple programming boxes used in
7 prior art programming methods are merely able to indicate
8 whether a device programmed properly. They do not have the
9 computing power to select alternative routes.
10 According to the invention, programming of the device is
11 controlled by a computer which can re-calculate routing in
12 response to information provided by the programming device.
13 This is preferably the same computer which formed the initial
14 list of antifuses. That computer also has access to data
15 files generated during the initial layout steps and can
16 revise the routing as necessary to generate an alternative
17 set of routes to bypass the failed antifuse. Need for a
18 separate programming box with a separate microprocessor is
19 eliminated. The computer which controls programming is also
20 preferably able to analyze timing, and perform other
21 computations performed when the initial routing was selected.
22 The ratio of reserved wiring segments to wiring segments
23 used during the routing calculation is preferably several
24 times as high as the ratio of expected antifuse failures to
25 total antifuses programmed. However, since the ratio of
26 failed antifuses is easily less than 1 to 1000, a ratio of
27 reserved segments on the order of 1 to 100 is sufficient to
28 accommodate almost all failures with only a 1% increase in
29 area.
30
31 BRIEF DESCRIPTION OF THE DRAWINGS
32 Fig. 1 shows a curve of voltage distribution and ranges
33 over which devices are accepted and rejected.
34 Fig. 2 shows a portion of an antifuse programmable logic
35 cell array which can be adaptively routed in response to an
36 antifuse which fails to program.

1 Fig. 3 shows signals to be supplied to an AND gate and
2 equivalent paths available upon failure of an antifuse.
3 Fig. 4 shows a structure with which the invention is
4 preferably used.
5 Fig. 4A shows the structure of Fig. 4 in which a planned
6 path for interconnecting two logic elements is shown.
7 Fig. 4B shows an alternative path to that of Fig. 4A
8 which resulted when antifuses on the planned path failed.
9 Fig. 5 shows an alternative adaptation which can correct
10 the routing failure of Fig. 2.
11
12 DETAILED DESCRIPTION OF THE INVENTION
13 Fig. 2 shows an overview of an antifuse programmable
14 logic cell array which can be adaptively programmed in
15 response to an antifuse which fails to program. Logic blocks
16 B1 through B8 represent programmable logic blocks which can
17 be programmed to perform multiple functions. The blocks must
18 be connected to each other through an interconnect structure
19 to cause the device to perform an overall function desired by
20 the user. The interconnect structure comprises horizontal
21 interconnect lines H1 through H7, vertical interconnect
22 lines, such as V1 through V5 and antifuses represented by
23 circles. For simplicity, some interconnect lines are not
24 labeled and most antifuses are not labeled. An antifuse is
25 positioned at each or at most intersections of the horizontal
26 and vertical interconnect lines. Unprogrammed antifuses are
27 represented by white circles. Programmed antifuses are
28 represented by black circles. Some lines are shown as
29 segmented, so that different segments of the same line can be
30 used for different signals. These segmented lines can be
31 interconnected by antifuses. In the example of Fig. 2, it is
32 desired to connect horizontal line H1 to vertical line V4
33 through antifuse A1. Thus sufficiently different programming
34 voltages are applied to horizontal line H1 and vertical line
35 V4 to cause antifuse A1 to become programmed. The black
36 circle shows that antifuse A1 has been programmed. Likewise,

it is desired to connect horizontal line H3 to vertical lines V6 and V7, which is accomplished by programming antifuses A2, A3, and A4. It is further desired to connect horizontal line H2 to vertical line V2, which in turn connects to a particular port in block B1. However, the **x** at the intersection of horizontal line H2 and vertical line V2 indicates that antifuse A5 at that intersection failed to become conductive when programming voltages were applied to lines H2 and V2. According to the invention, in order for the device to perform the desired function, another path is located and programmed. Segmented redundant horizontal line H4 and vertical line V5 and antifuses S1, S2, and S3 are used to form an alternative path from H2 to V2. Thus an alternative path to the desired destination is completed.

If the antifuse defect density is even moderately high, for example one hundred defects per million antifuses, a small number of redundant lines is sufficient to tremendously

TABLE I

DEVICE YIELD VS ANTIFUSE PROGRAMMING YIELD

N =	14,000 ANTIFUSES TO PROGRAM			50,000 ANTIFUSES TO PROGRAM		
CONDITION	F=1 ppm	10 ppm	100 ppm	1 ppm	10 ppm	100 ppm
1. No adaptive routing Yield = $(1 - F)^N$	98.6%	86.9%	24.7%	95.1%	60.7%	0.7%
2. 90% antifuses repairable Yield = $(1 - F \cdot U)^N$	99.9%	98.6%	86.9%	99.5%	95.1%	60.7%
3. 99% antifuses repairable Yield = $(1 - F \cdot U)^N$	99.99%	99.9%	98.6%	99.95%	99.5%	95.1%
4. 100% antifuses repairable Yield = $(1 - F^2)^N$	~100%	99.9999%	99.99%	~100%	99.9995%	99.95%

F = fraction of antifuses which fail to program
U = fraction of antifuses for which no repair is available
N = number of antifuses to be programmed
ppm = parts per million, here failed antifuses per million

1 increase the yield of successfully programmed devices. As
2 shown in TABLE I, without adaptive routing, yield falls
3 sharply with both increased defect density and with device
4 size. The first condition in TABLE I shows yield when no
5 adaptive routing is used. The second condition in TABLE I
6 shows yield for different defect densities and device sizes
7 when 90% of the antifuses have a repair option. Condition 3
8 shows yield when 99% of the antifuses have a repair option,
9 and condition 4 shows yield when 100% of the antifuses have a
10 repair option, that is, when another path can be found for
11 any failed antifuse. The values in TABLE I are based on the
12 following assumptions: (1) failures do not cluster so that
13 repair options fail together with initial failures, (2) only
14 one repair option will be tried (thus the numbers are lower
15 bounds; if a first attempted repair failed and a second
16 attempt were made, the second attempt may succeed, further
17 decreasing the failure rate), and (3) timing degradation can
18 be tolerated. The improvement offered by the present
19 invention is clearly several orders of magnitude.
20 In particular, if there are 50,000 antifuses to program,
21 the right column of TABLE I shows that providing alternative
22 paths around every antifuse in a device having an antifuse
23 failure rate of 100 antifuses per million brings yield from a
24 totally unacceptable 0.7% to an acceptably 99.95%. Referring
25 back to Fig. 1, the region labeled "ANTIFUSES WHICH CAUSE
26 REJECTION" is largely dealt with by the method of the
27 invention, to the extent shown in TABLE I.
28 The architecture is preferably arranged for 100%
29 repairability, so that each wiring segment may be connected
30 to each other wiring segment along more than one path. If an
31 antifuse on a first path fails to program, a second path not
32 including the antifuse which failed to program can connect
33 the same logic elements as would have been connected by a
34 path including the failed antifuse. There should be a way to
35 bypass any antifuse, so that failure of a single antifuse
36 never causes failure of the entire device. Short segments

1 are particularly useful for forming alternative routes, since
2 they add little capacitance (therefore delay) to the total
3 path. Some devices typically use long segments for carrying
4 a clock signal or other high fanout signal which needs low
5 skew. When the wiring segments include a mix of long and
6 short segments, one preferred architecture allows one long
7 segment to be connected to another long segment both directly
8 through a single antifuse and through a short segment plus
9 two antifuses.
10 It is possible to designate lines and antifuses not to
11 be used during the initial routing calculation, then program
12 all routes in the calculated design before making any
13 repairs, and finally make the repairs using the unused lines
14 and antifuses. The recalculation when unused wiring segments
15 and antifuses have been reserved is on the order of a few
16 milliseconds, short enough that a user will not be
17 inconvenienced by the delay, or even aware of the delay.
18 Alternatively, for any or all parts of paths in which
19 antifuses are not yet programmed, it is possible to recompute
20 routing directly after a failure is noted. Any paths, parts
21 of paths, and placements not yet committed by programmed
22 antifuses may be changed.
23 Placement and routing are typically selected in order to
24 meet certain timing requirements of the user. Some paths
25 will meet their timing requirements with more time to spare
26 than others. In order for any necessary repairs to also meet
27 timing requirements, the order of antifuse programming may be
28 selected so that paths having the least time to spare are
29 programmed first. In an embodiment in which recalculation is
30 performed directly after an antifuse failure, programming the
31 tightest paths first means that failure of an antifuse can
32 more likely be successfully repaired (within a tight timing
33 requirement) when more unprcgrammed resources remain.
34 Regarding timing degradation due to repair, the initial
35 placement and routing may be selected such that all timing
36 requirements are met with time to spare. This time to spare

1 can be used in the adapted routing paths and a user's timing
2 requirement will still be met.
3
4 Pin, Interconnect, Cell, and Block Swapping to Maintain
5 Timing
6 It is preferable to preserve the timing when adapting to
7 a failed antifuse. An alternative route as selected above is
8 almost always slower than the original route (which could not
9 be completed because of a failed antifuse). There are
10 situations in which placement and routing cannot be found in
11 which there is sufficient time to spare for a repair which
12 slows the timing. There are also situations in which the
13 user has performed timing simulations and expects a
14 particular timing for every path in a design. It may be
15 unacceptable to have a change in timing for some paths in one
16 device compared to the identical paths in other devices.
17 Thus a change in timing because of a failed antifuse in that
18 device may cause the device to be rejected. It is
19 preferable, in response to antifuse failure, to find an
20 alternative layout which does not change signal timing at
21 all. Instead of adapting the routing, it is sometimes
22 possible to adapt placement of the logic elements, or to
23 alter which signals will feed which inputs to a logic
24 function.
25
26 Interconnect Swapping
27 In one preferred embodiment, the method is used with a
28 structure having cells grouped as shown in Fig. 4. The cells
29 are preferably grouped into blocks of eight cells CELL1
30 through CELL8 with a ninth cell CELL9 used for programming
31 the antifuses but not performing logic. Logic performed in
32 cells CELL1 through CELL8 receives input from several of the
33 vertical segments shown in Fig. 4. The nine cells of a block
34 combine with the antifuse interconnect structure (antifuses
35 are represented by black dots) which are programmed to
36 interconnect the cells to each other to implement a circuit

1 design desired by a user. Four cell blocks are shown in Fig.
2 4. A typical integrated circuit array will comprise 100 to
3 1000 of these cell blocks such as shown in Fig. 4 plus
4 peripheral I/O circuitry, clock oscillators, and other
5 overhead circuitry usually positioned along the perimeter of
6 the cell. Cells CELL1 through CELL4 are interconnectable
7 through a special direct connection (called a cascade
8 connection) which does not use the antifuse interconnect
9 structure and is especially fast. CELL5 through CELL8 are
10 likewise interconnectable. Some horizontal line segments
11 positioned between the upper group of cells CELL1 through
12 CELL4 and the lower group of cells CELL5 through CELL8 are
13 minimum length interconnect line segments extending the
14 length of one cell block and not continuous across cell
15 CELL9. Longer horizontal segments extend more than one
16 block. Vertical segments extending between CELL1 and CELL4
17 are about one block high. The same is true for vertical
18 segments extending between cells CELL2, CELL3, CELL4 and
19 CELL6, CELL7, CELL8. Vertical segments to the left of CELL1
20 and CELL8 are about two blocks high, and allow cells in the
21 upper blocks to be connected to cells in the lower blocks.
22 The example shown is only two blocks wide and two blocks
23 high, thus no segments are more than two blocks in length.
24 However, in a typical integrated circuit array, some segments
25 will extend a longer length, especially those intended for
26 carrying global signals such as clock signals.
27 As can be seen in Fig. 4, many interconnect segments are
28 interchangeable. In the event that a failed antifuse is
29 adding a vertical segment which extends between cell blocks,
30 another vertical segment in the same cell block can be used
31 with no change in timing. This same result may be achieved
32 with many of the horizontal segments.
33 In general, if one cell output is being connected to
34 another cell input through a sequence of wiring segments,
35 there are multiple paths available which will have equal
36 timing. For example, Fig. 4A shows the circuit of Fig. 4 in

1 which a first path has been selected for connecting the
2 output of cell CELL1 to the input of another cell equivalent
3 to CELL1 in another block. Wiring segment 1 carries the
4 output of CELL1. In Fig. 4A, horizontal wiring segment 2 has
5 been selected to carry the signal out of the upper left block
6 in which CELL1 is located. Vertical wiring segment 3 has
7 been selected to propagate the signal to the lower left.
8 Horizontal wiring segment 4 was selected to propagate the
9 signal to the lower left block, and vertical wiring segment 5
10 leads to a logic input of the cell in the lower left block
11 which is to receive the signal. Assuming antifuses will be
12 programmed to connect the wiring segments in numerical order,
13 the antifuse at the intersection of segments 1 and 2 is
14 programmed first, and others later. Fig. 4B shows an
15 alternative routing which is found if every antifuse fails
16 except that connecting segments 4 and 5 (an unrealistically
17 bad result, which nevertheless illustrates the adaptability).
18 When failure of the antifuse connecting segments 1 and 2 is
19 detected, a new horizontal segment 2' is selected (either a
20 segment initially planned to be unused, or a used segment
21 which can be switched with segment 2 or another segment) and
22 the antifuse at the intersection of segments 1 and 2' is
23 programmed. Then the antifuse at the intersection of
24 segments 2' and 3 is attempted and fails, so the antifuse at
25 the intersection of segments 2' and 3' is programmed instead,
26 which is assumed to be successful. The antifuse at the
27 intersection of 3' and 4 fails and is replaced by the
28 antifuse at the intersection of 3' and 4'. Finally the
29 antifuse at the intersection of 4' and 5 programs
30 successfully. The successfully programmed path has the same
31 number of wiring segments, each of the same length as the
32 originally planned path, and the same number of antifuses.
33 Thus the path delay is the same within statistical variation,
34 and no delay has been added by the adapted routing.
35
36

Pin Swapping

The above example assumed that the last step in the path programmed successfully. It is of course not possible to predict which antifuses will fail, and some of the time it will be the last antifuse on a path which will fail. Such a failure may sometimes be accommodated by a method called pin swapping. Some inputs to a logic cell are interchangeable (for example inputs to an AND gate) such that signals coming to these inputs can be swapped with no change in the function performed by the cell. In the event that a failed antifuse leads to a logic cell input which can be swapped with another input to the same cell with no change in functionality, swapping the routes which carry two signals to the cell will repair the antifuse failure with no change in the timing. Also, such a swap does not consume additional wiring segments.

Fig. 3 shows a portion of a logic array in which two signals, signal 1 and signal 2 are to be applied to inputs of an AND gate. As originally decided, signal 1 will be applied to input I1 and signal 2 will be applied to input I2. Applying these signals requires the programming of antifuses A1,1 and A2,2. But when the programming of the first antifuse A1,1 is attempted, the antifuse fails to program. The identical function can be achieved with the identical timing by programming antifuses A1,2 and A2,1. Thus when antifuse A1,1 fails to program, the mapping is recalculated. Programming of antifuse A2,2 is not attempted. Instead, antifuses A1,2 and A2,1 are programmed and the repair is made with no degradation in timing. Alternatively, if antifuse A2,2 were already programmed before the failure of antifuse A1,1 was discovered, since the logic function illustrated is a four-input AND gate, signal 1 may be brought in on input I3 or input I4. Such alternatives enhance the repair methods discussed in connection with Fig. 2 and Figs 4, 4A and 4B.

1 Cell Swapping
2 Since the capacitance of connections leading to any cell
3 in the same cell block of Fig. 4 will not be changed by
4 swapping cells in the cell block, timing of a signal path
5 will be the same for any cell in the cell block. Thus a
6 defective antifuse can be avoided by exchanging the logic in
7 one cell for the logic in another cell in the same block and
8 adjusting the antifuses to be programmed accordingly. Since
9 all cells in a cell block have equivalent connections to the
10 horizontal segments extending through the block, the
11 connections to the exchanged cells can be substituted with no
12 change in timing and with no additional use of resources.
13 However, if the logic cells to be connected by a failed
14 antifuse make use of the cascade connection, their
15 relationship to each other must be maintained in order for
16 the expected timing to be maintained. Thus any cell swapping
17 must take this cascading into account. Distributing unused
18 cells to blocks in an array such than many blocks include
19 unused cells increases the likelihood that one cell can be
20 swapped with another (an unused one in the block) when
21 antifuses leading to the first cell fail to program.
22 Fig. 5 shows an alternative repair to that of Fig. 2,
23 where cells B1 and B2 have been re-designated to swap
24 functions with cells B5 and B6. Such a swap may be performed
25 either if no antifuses have yet been programmed to commit
26 these two sets of cells to their original functions at the
27 time of the antifuse failure, if any antifuses which have
28 been programmed connect identically to the two cells, or if
29 an unused cell is available in the block. The example of
30 Fig. 5, in which cells B1, B2, B5 and B6 have identical
31 timing is assumed to have been planned such that the original
32 use of cells B1 and B2 required the cascade feature between
33 them. In the example of Fig. 2, the attempt to connect line
34 H2 to line V2 resulted in a failed antifuse A5. Here, in
35 order to maintain timing, which requires maintaining the
36 cascade connection and also maintaining the delay of each

1 path in the interconnect structure, cells B1 and B2 are
2 swapped for cells B5 and B6, and the attempt to program the
3 antifuses is repeated. The result is shown in Fig. 5.
4 Antifuse S4 is programmed to connect line H2 to line V1,
5 which connects to cell B5. Antifuse S5 is programmed to
6 connect line H6 to line V4, which connects to cell B1.
7 Antifuse S6 is programmed to connect line H1 to a vertical
8 line leading into cell B6. Thus antifuses A1 and A7 are not
9 used as originally planned. Of course, to make this
10 adjustment in the planned routing, antifuses A1 and A7 must
11 not yet have been programmed at the time the failure of
12 antifuse A5 occurs.
13 In order to allow further flexibility in correcting for
14 defects when it is preferred to maintain timing, some inputs
15 to a logic cell may be left initially unused and some logic
16 blocks may be reserved for use during the repair phase, in
17 addition to reserving some cells within a block for adaptive
18 swapping of cells.
19 In order to allow the user a choice between maintaining
20 timing control and achieving maximizing yield of programmed
21 parts, a provision can be made in the programming software to
22 let the user specify that the part must be rejected if a
23 repair can not be made without changing the timing.
24 Other embodiments of the invention will become obvious
25 to those skilled in the art in light of the above
26 description. For example, even though the embodiments above
27 are described in conjunction with a logic device, that is a
28 device having both interconnections and logic elements, the
29 invention works with a device which includes interconnect
30 only and does not include logic elements. Also, even though
31 the invention has been described in connection with
32 antifuses, it will work with other one-time programmable
33 technologies as well. Such other embodiments are intended to
34 fall within the scope of the present invention.

1 CLAIMS
2
3 We claim:
4 1. A method of programming an antifuse programmable
5 logic device having interconnect wiring segments, and
6 antifuses which programmably connect said wiring segments to
7 each other, said method comprising the steps of:
8 providing a design in machine readable form, said design
9 comprising connections between selected nodes in
10 said device;
11 for each of said connections, selecting active wiring
12 segments and active antifuses in said antifuse
13 programmable logic device to implement said
14 connection, thereby to form a route for said
15 connection;
16 programming at least some of said active antifuses;
17 if during programming any of said active antifuses fails
18 to become conductive, selecting from those
19 antifuses not yet programmed an alternative set of
20 antifuses to complete said connections;
21 programming said alternative set of antifuses, thereby
22 to form an implementation of said design.
23
24 2. A method as in Claim 1 comprising the further step
25 of leaving spare wiring segments and spare antifuses not used
26 initially for any of said connections.
27
28 3. A method as in Claim 1 in which
29 said design comprises a logic design comprising both
30 design elements and said connections, and
31 said device further comprises logic elements, said
32 antifuses programmably connecting said logic
33 elements to said wiring segments,
34 said method comprising the further steps of:
35 selecting one of said logic elements to implement each
36 of said design elements;

1 leaving spare logic elements not used initially for any
2 of said design elements.
3
4 4. A method as in Claim 1 in which
5 said design comprises a logic design comprising both
6 design elements and said connections, and
7 said device further comprises logic elements grouped
8 into blocks, said antifuses programmably connecting
9 said logic elements to said wiring segments,
10 said method comprising the further step of:
11 leaving spare blocks not used initially for any of said
12 design elements.
13
14 5. A method as in Claim 1 in which
15 said design comprises a logic design comprising both
16 design elements and said connections, and
17 said device further comprises logic elements, said
18 antifuses programmably connecting said logic
19 elements to said wiring segments,
20 said method comprising the further steps of:
21 selecting one of said logic elements to implement each
22 of said design elements;
23 leaving spare logic element inputs not used initially in
24 at least some of said logic elements.
25
26 6. A method as in Claim 1 in which said active
27 antifuses are programmed in the order of time to spare for a
28 path such that paths having the least time to spare in
29 meeting a timing requirement are programmed first.
30
31 7. A method as in Claim 1 in which said alternative set
32 of antifuses is selected such that paths selected after an
33 antifuse failure have the same timing as corresponding paths
34 selected before said antifuse failure.
35
36 8. A method as in Claim 1 in which said alternative set

1 of antifuses is selected so that said implementation of said
2 logic design meets the same timing requirements as would have
3 been met if no antifuses failed.
4
5 9. A method as in Claim 1 in which said step of
6 selecting an alternative set of antifuses is performed after
7 a first failed antifuse is found.
8
9 10. A method as in Claim 9 in which said alternative
10 set of antifuses is selected to swap equivalent input ports
11 of a logic element which will receive plural input signals,
12 thereby avoiding said failed antifuse.
13
14 11. A method as in Claim 1 in which all of said active
15 antifuses are programmed as part of a first programming step,
16 and said alternative set of antifuses are selected and
17 programmed after said first programming step.
18
19 12. A method as in Claim 1 in which said wiring
20 segments comprise both horizontal and vertical wiring
21 segments, and said antifuses are positioned at intersections
22 of said horizontal and vertical wiring segments, spare wiring
23 segments being selected from both horizontal and vertical
24 wiring segments.
25
26 13. A method of programming an antifuse programmable
27 logic device having logic elements, interconnect wiring
28 segments, and antifuses which programmably connect wiring
29 segments to each other and to said logic elements, said
30 method comprising the steps of:
31 providing a logic design in machine readable form, said
32 logic design comprising design elements and
33 connections between selected ones of said design
34 elements;
35 selecting one of said logic elements to implement each
36 of said design elements;

1 for each of said connections, selecting active wiring
2 segments and active antifuses in said antifuse
3 programmable logic device to implement said
4 connection and thereby to form a conductive route
5 for said connection;
6 leaving spare wiring segments and spare antifuses not to
7 be used initially for any of said connections;
8 programming each of said active antifuses;
9 for each of said active antifuses which failed to become
10 conductive upon said programming, selecting an
11 alternative route to complete that connection of
12 which said failed antifuse is a part, using
13 selected ones of said spare wiring segments and
14 said spare antifuses,
15 programming said selected spare antifuses, thereby to
16 form said alternative route for said connection.
17
18 14. A method of programming an antifuse programmable
19 logic device having logic elements, interconnect wiring
20 segments, and antifuses which programmably connect wiring
21 segments to each other and to said logic elements, said
22 method comprising the steps of:
23 providing a logic design in machine readable form, said
24 logic design comprising design elements and
25 connections between selected ones of said design
26 elements;
27 selecting one of said logic elements to implement each
28 of said design elements;
29 for each of said connections, selecting active wiring
30 segments and active antifuses in said antifuse
31 programmable logic device to implement said
32 connection and thereby to form a conductive route
33 for said connection;
34 programming said active antifuses in turn;
35 if an antifuse fails to become conductive upon
36 programming, for at least that connection of which

1 said failed antifuse is a part, calculating an
2 alternative route to complete that connection.
3
4 15. A method as in Claim 14 in which said step of
5 calculating an alternative route to complete that connection
6 further comprises calculating alternative routes for
7 additional unprogrammed connections.
8
9 16. A method as in Claim 14 in which said steps of
10 providing, selecting, programming, and calculating are
11 performed under control of a single computer.
12
13 17. An antifuse based field programmable logic device
14 comprising:
15 logic elements;
16 wiring segments;
17 antifuses positioned to be programmed to connect said
18 wiring segments to each other and to said logic
19 elements, each wiring segment being connectable to
20 another wiring segment or to a logic element along
21 more than one path, whereby if an antifuse on a
22 first path fails to program, a second path not
23 including the failed antifuse can connect the same
24 logic elements as would have been connected by a
25 path including the failed antifuse.
26
27 18. An antifuse based field programmable logic device
28 comprising:
29 logic elements;
30 wiring segments;
31 antifuses which can be programmed to connect said wiring
32 segments to each other and to said logic elements, each
33 antifuse being capable of being bypassed if programming
34 of said antifuse fails.
35
36 19. A device as in Claim 18 in which said wiring

1 segments comprise long segments and short segments, a short
2 segment spanning a dimension of one of said logic elements
3 and a long segment spanning a greater length, each long
4 segment being connectable to another wiring segment both
5 directly and through one of said short segments.
6
7 20. A device as in Claim 18 in which said wiring
8 segments comprise long segments and short segments and said
9 logic elements are grouped into blocks, a short segment
10 spanning a dimension of one of said blocks and a long segment
11 spanning a greater length, each long segment being
12 connectable to another wiring segment both directly and
13 through one of said short segments.
14
15 21. A device as in Claim 20 in which some of said short
16 segments are parallel to at least one of said long segments
17 and provide alternative routes for connecting to said long
18 segment from wiring segments perpendicular to said at least
19 one of said long segments.
20
21 22. A device as in Claim 21 in which said at least one
22 of said long segments carries a clock buffer signal.

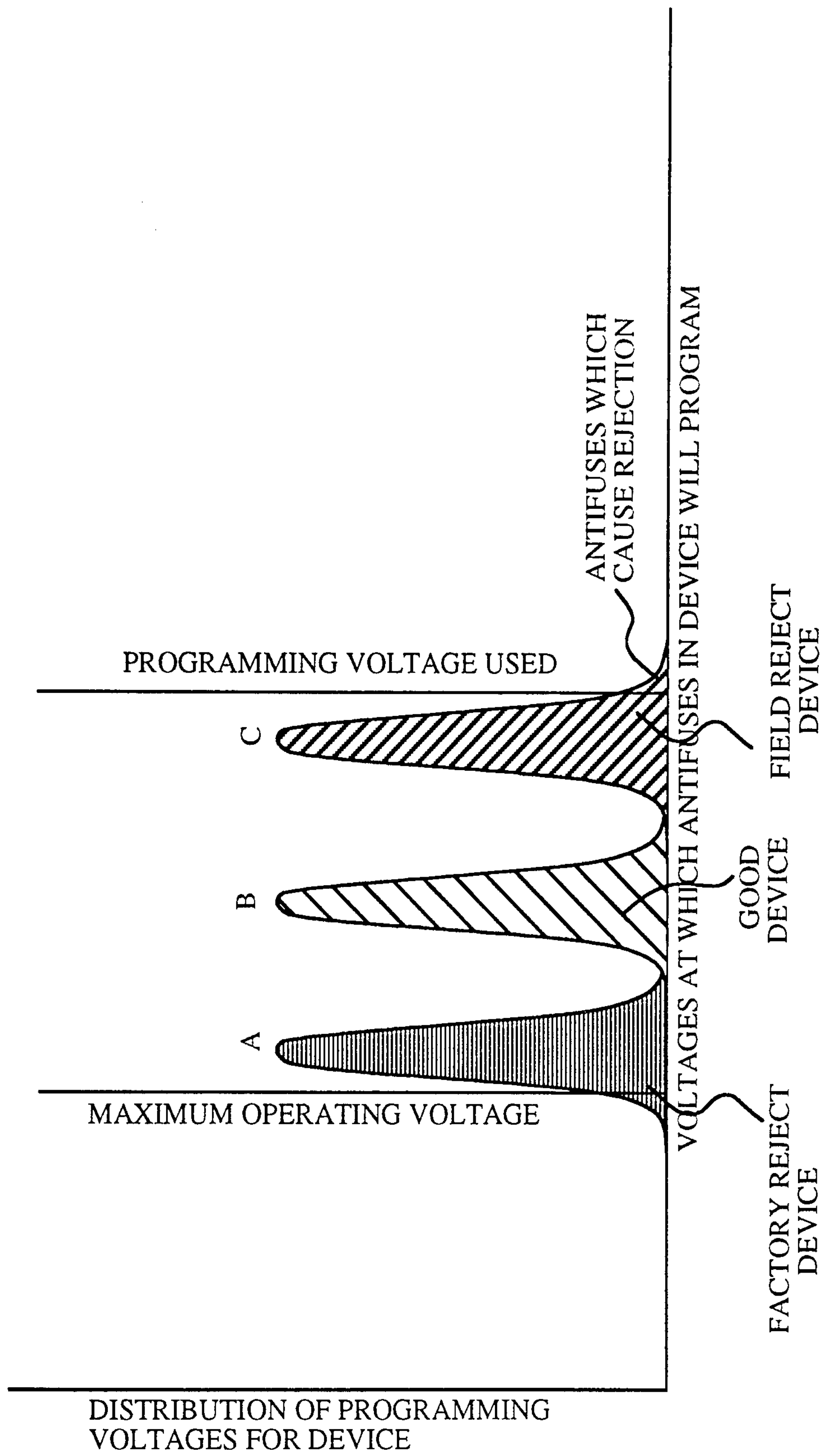


FIG. 1

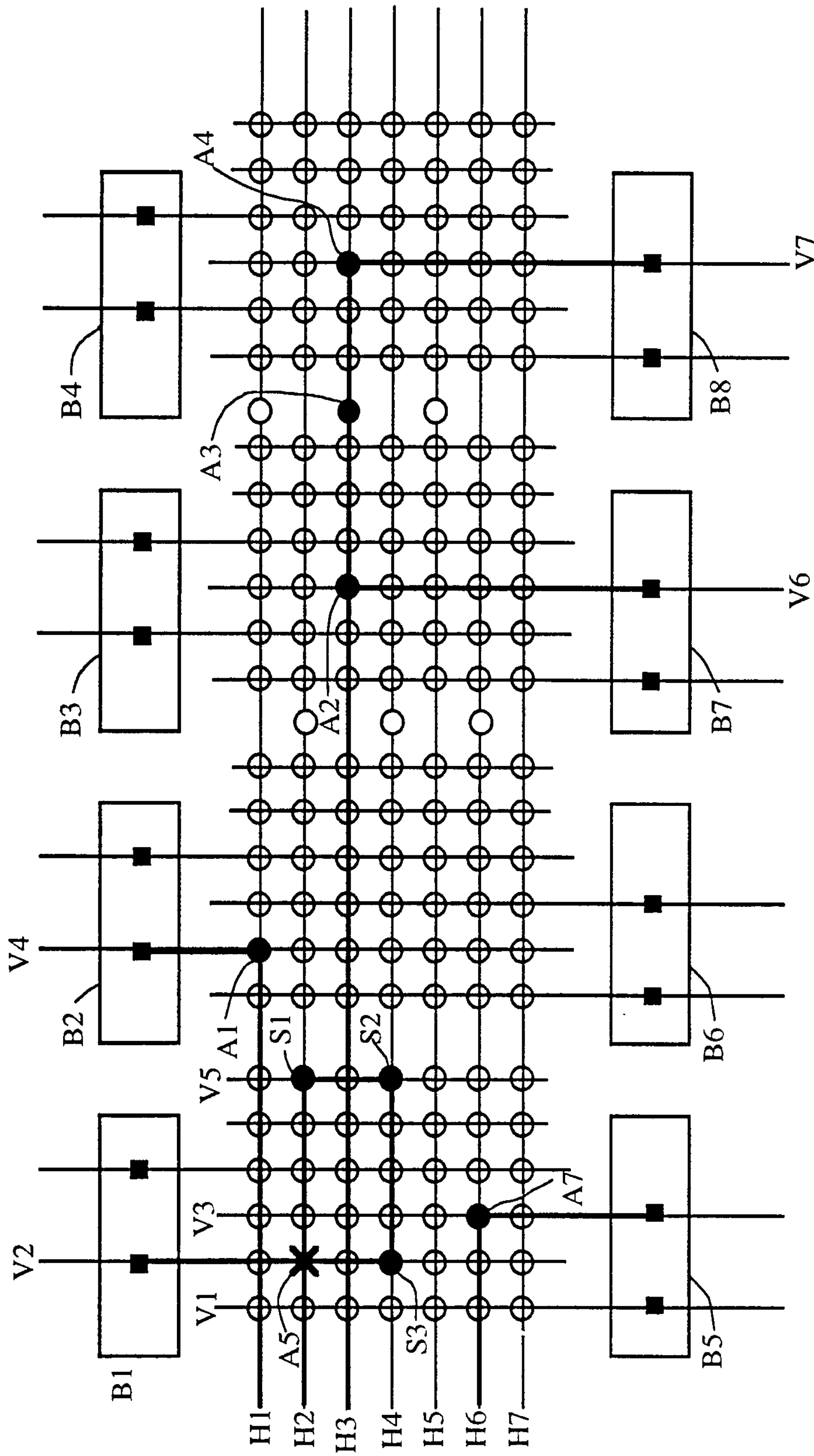


Fig. 2

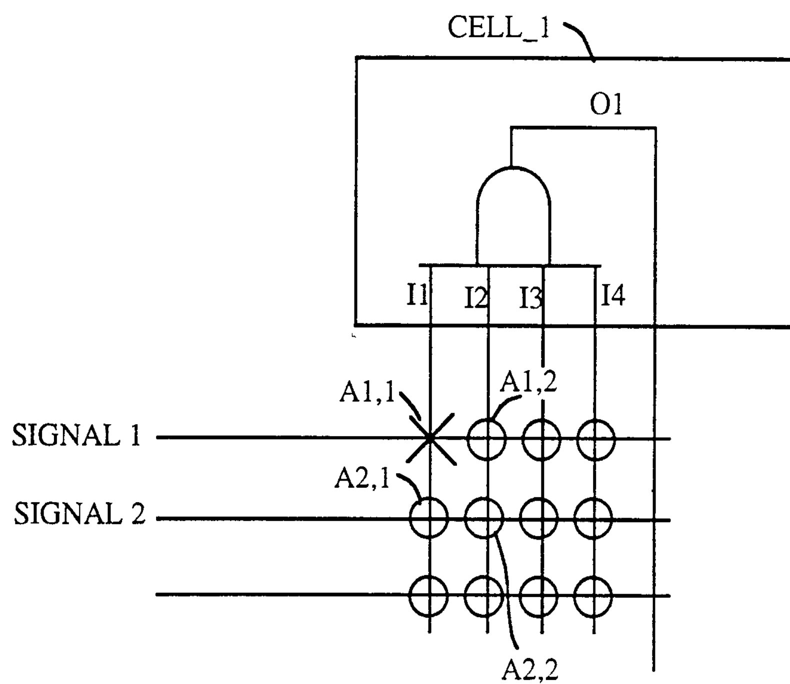


Fig. 3

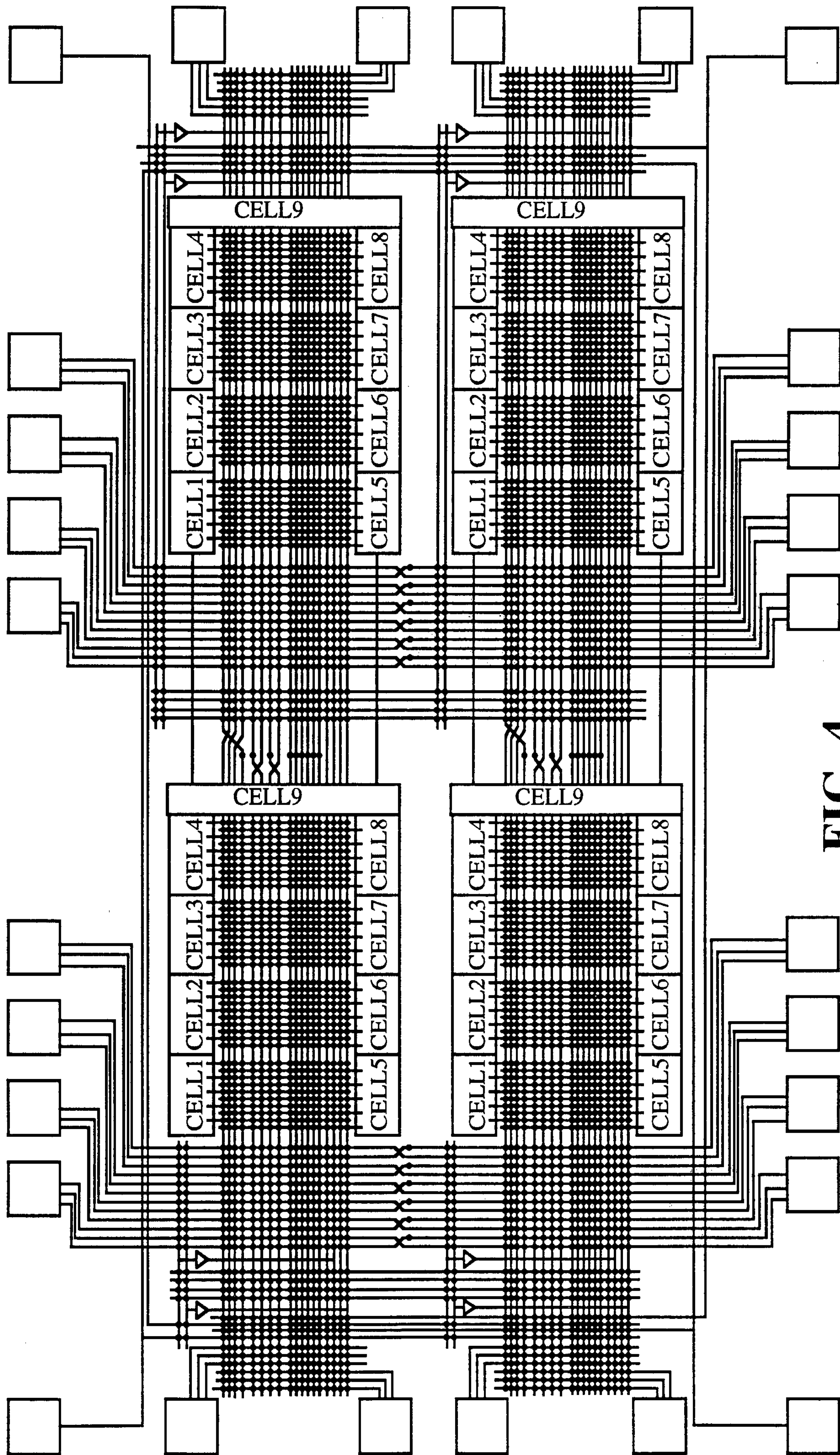


FIG. 4

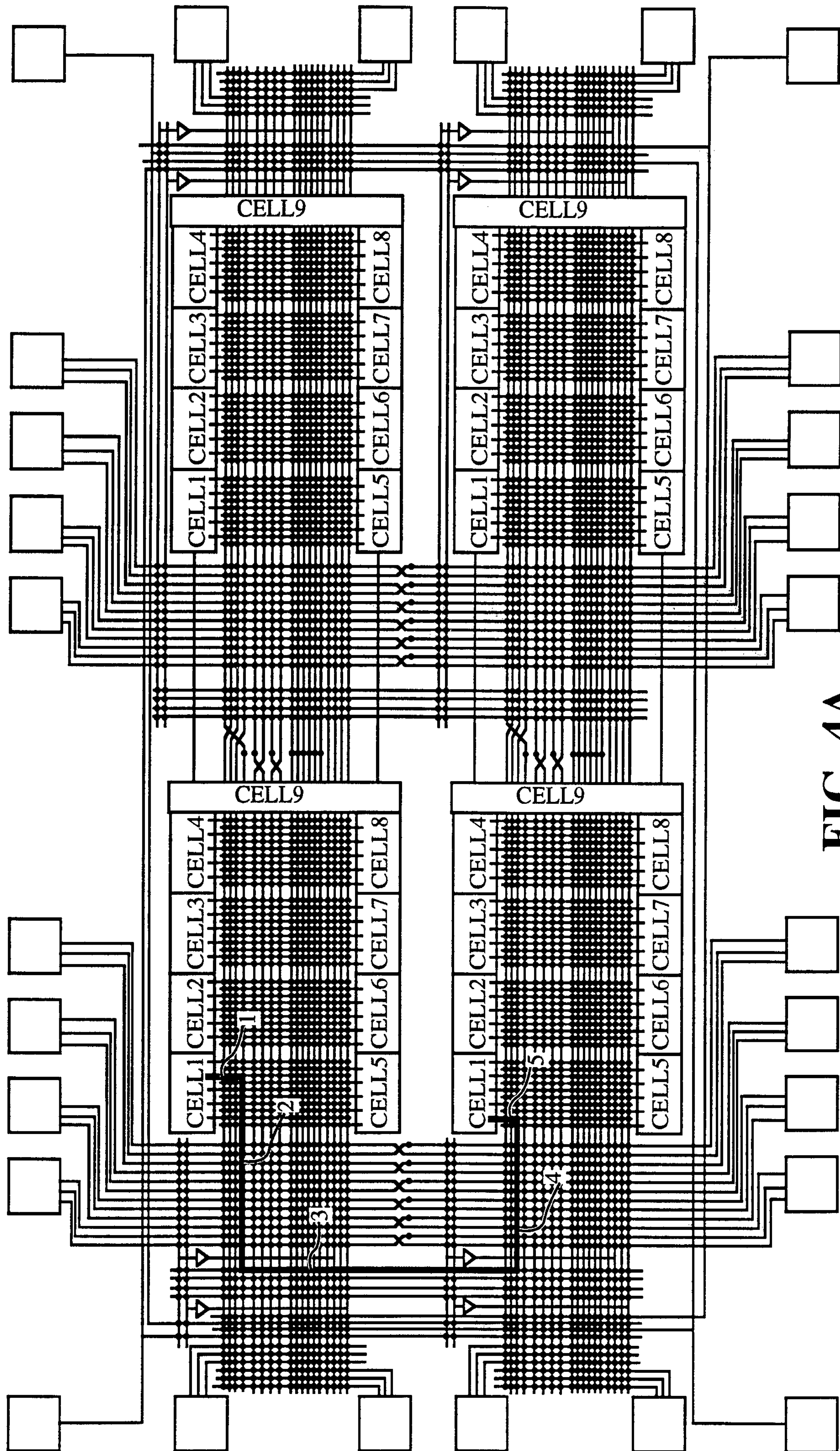


FIG. 4A

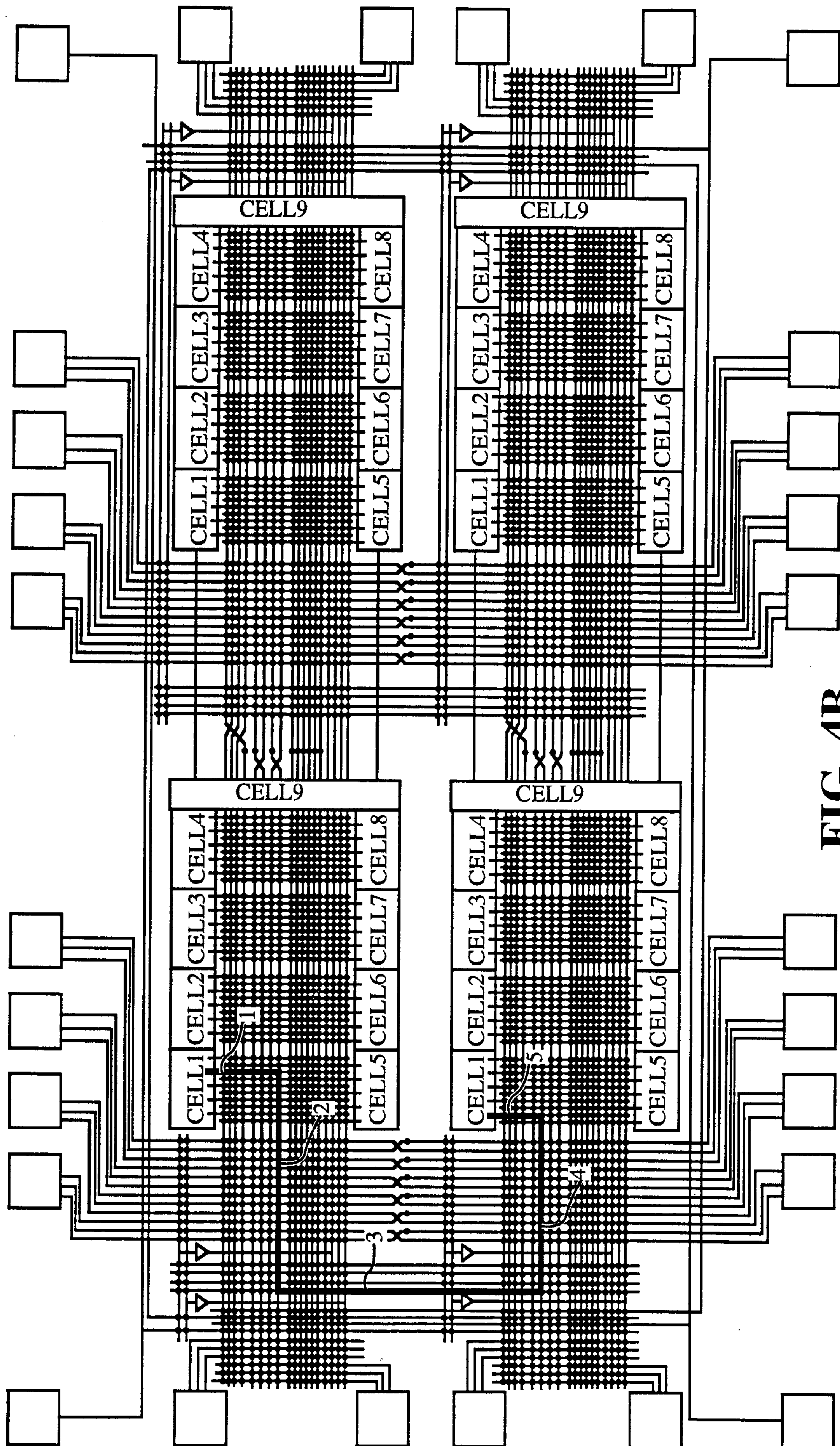


FIG. 4B

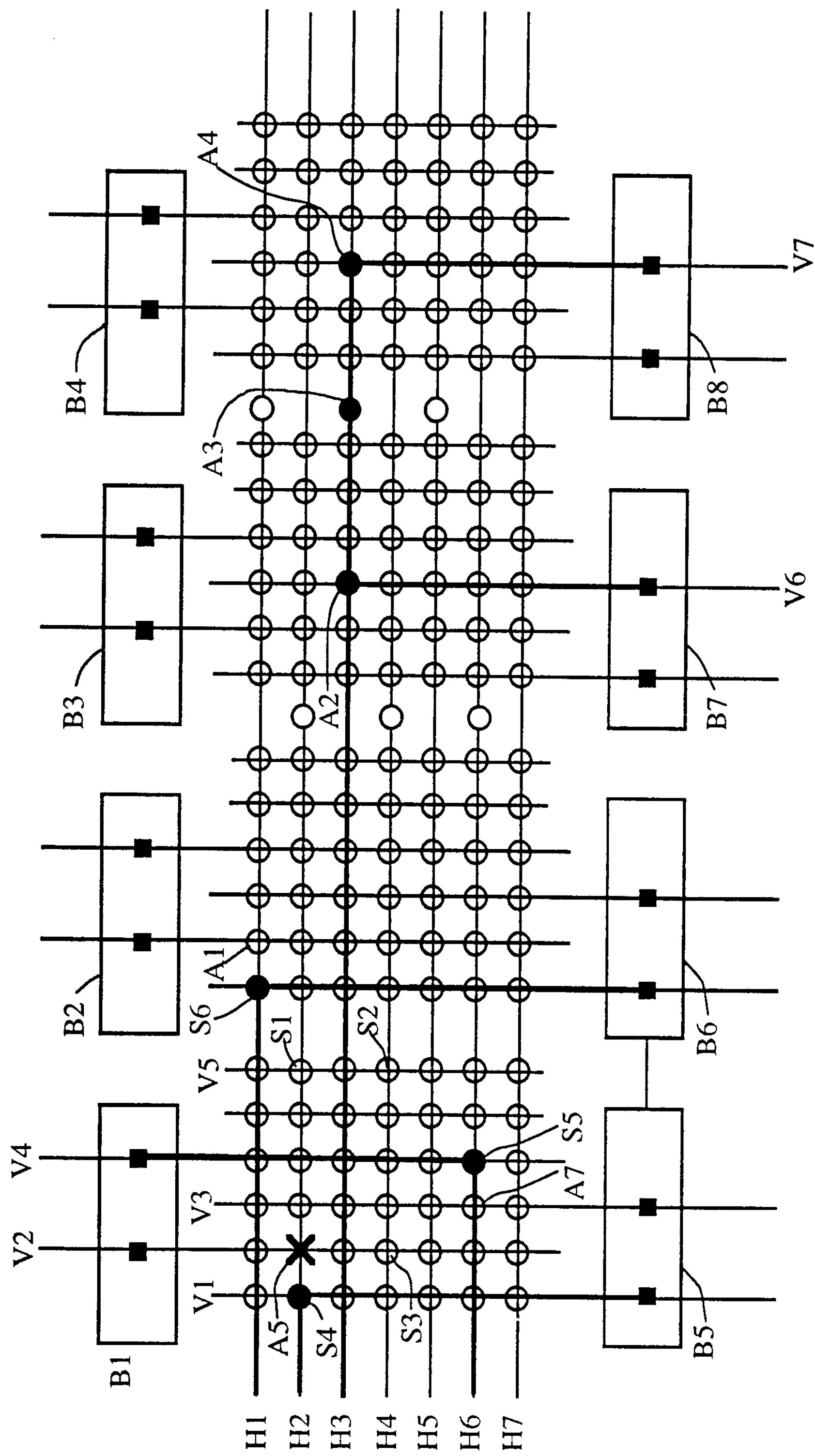
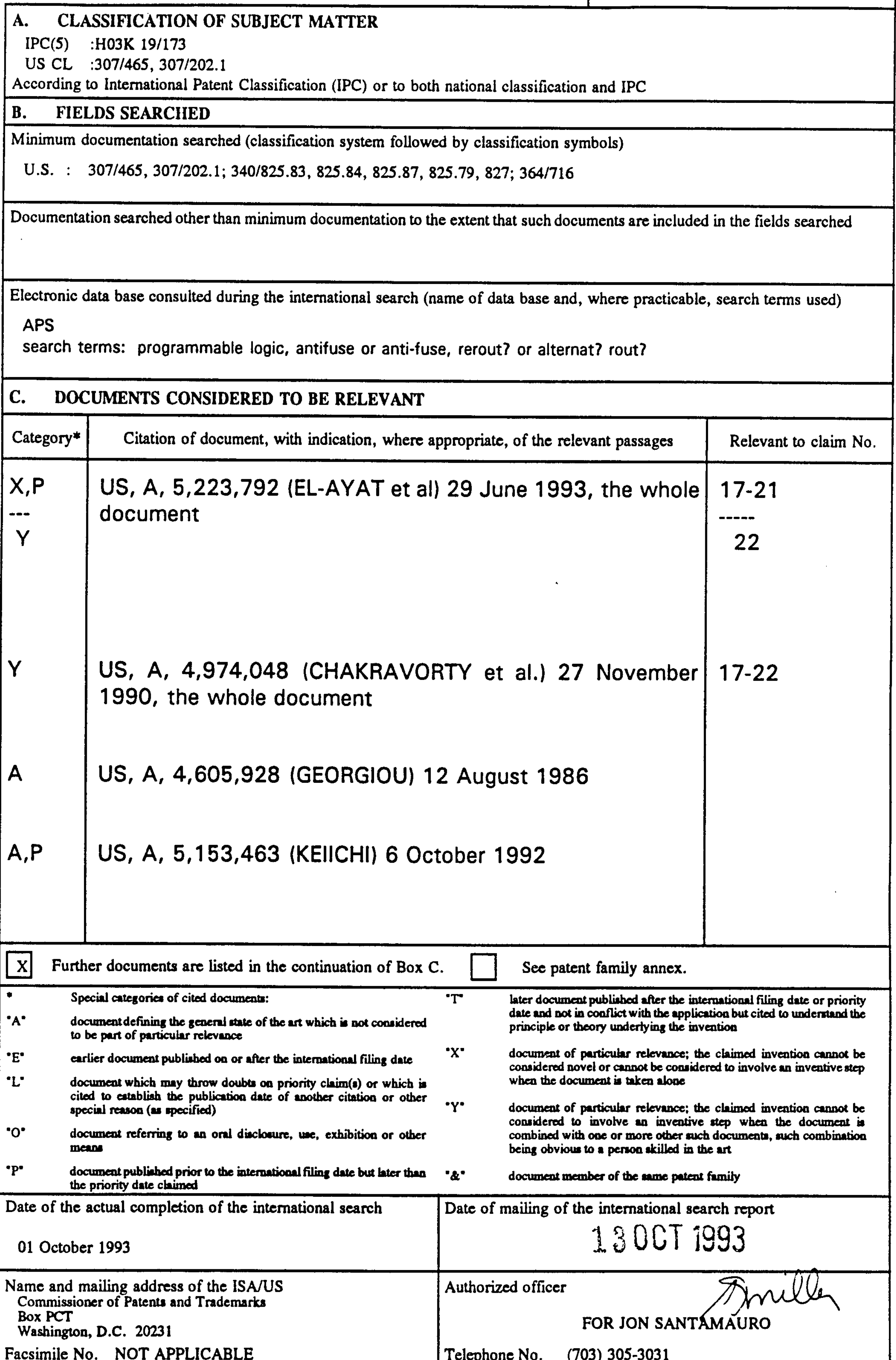


Fig. 5



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/07212

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H03K 19/173
US CL :307/465, 307/202.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 307/465, 307/202.1; 340/825.83, 825.84, 825.87, 825.79, 827; 364/716

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS
search terms: programmable logic, antifuse or anti-fuse, rerout? or alternat? rout?

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P --- Y	US, A, 5,223,792 (EL-AYAT et al) 29 June 1993, the whole document	17-21 ----- 22
Y	US, A, 4,974,048 (CHAKRAVORTY et al.) 27 November 1990, the whole document	17-22
A	US, A, 4,605,928 (GEORGIOU) 12 August 1986	
A,P	US, A, 5,153,463 (KEIICHI) 6 October 1992	

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be part of particular relevance
"E" earlier document published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
01 October 1993

Date of mailing of the international search report
13 OCT 1993

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. NOT APPLICABLE

Authorized officer
FOR JON SANTAMAURO
Telephone No. (703) 305-3031

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/07212

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US, A, 4,758,745 (ELGAMAL et al) 19 July 1988	