PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/30	A1	(11) International Publication Number: WO 00/48070
		(43) International Publication Date: 17 August 2000 (17.08.00)

(21) International Application Number: PCT/US00/03566

(22) International Filing Date: 11 February 2000 (11.02.00)

(30) Priority Data:

60/119,841	12 February 1999 (12.02.99)	US
09/382,402	24 August 1999 (24.08.99)	US

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

Published
With international search report.

(71) Applicant *(for all designated States except US)*: MICROUNITY SYSTEMS ENGINEERING, INC. [US/US]; 475 Potrero Avenue, Sunnyvale, CA 94086 (US).

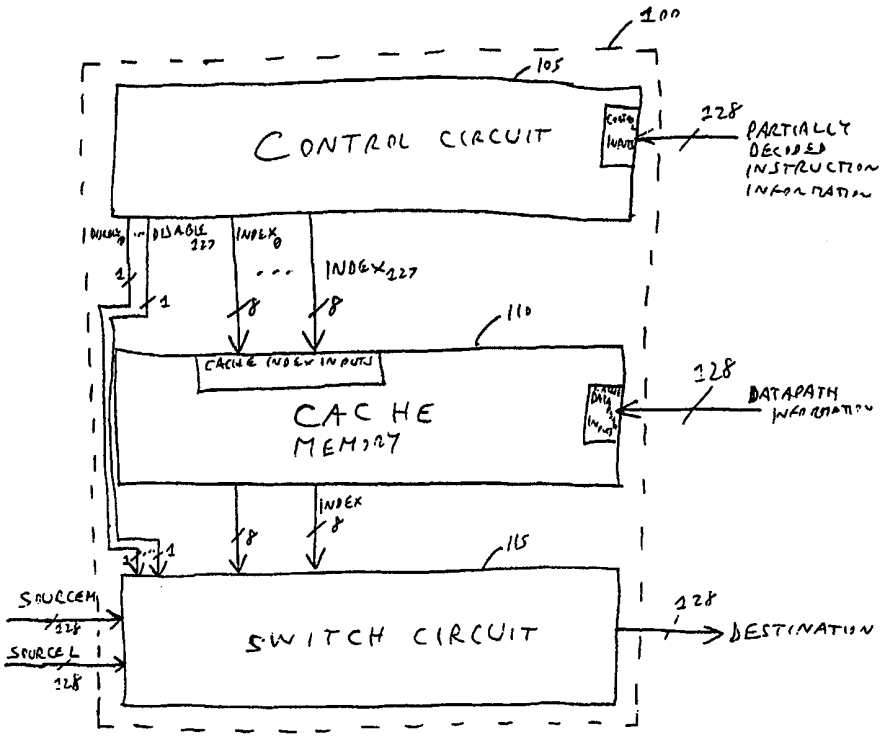
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HANSEN, Craig [US/US]; 350 Yerba Santa Avenue, Los Altos, CA 94022 (US). BATEMAN, Bruce [US/US]; 3808 Monte Sereno Terrace, Fremont, CA 94530 (US). MOUSSOURIS, John [US/US]; P.O. Box 719, Palo Alto, CA 94302 (US).

(74) Agent: EAKIN, James, E.; McDermott, Will & Emery, 2700 Sand Hill Road, Menlo Park, CA 94002 (US).

(54) Title: SYSTEM AND METHOD TO IMPLEMENT A CROSS-BAR SWITCH OF A BROADBAND PROCESSOR

(57) Abstract

The present invention provides a cross-bar circuit (100) that implements a switch (115) of a broadband processor. The cross-bar circuit (100) includes: a switch circuit (115) which includes $2^m.2^n$:1 multiplexor circuits (202–204) where each of the 2^n:1 multiplexor circuits (202–204) has a unique n-bit index input, one disable input, and a 2^n-bit wide source input receives an n-bit index at the n-bit index input, a disable bit at the disable input, and the 2^n-bit input source word at the 2^n-bit wide source input, and decodes the n-bit index either to select and output as an output destination bit one bit from the 2^n-bit input source word if the disable bit has a logic low value; a cache memory (110) that has 2^m cache datapath inputs; and 2^m cache index input; and a control circuit (105) that has a plurality of control inputs receives the partially decoded instruction information on the plurality of control inputs, provides a second set of the n-bit indexes for the switch circuit (115), and provides the disable bits for the switch circuit (115) where the control circuit (105) is logically coupled to the switch circuit (115) and to the cache memory (110).



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.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD TO IMPLEMENT A CROSS-BAR SWITCH OF A BROADBAND PROCESSOR

SPECIFICATION

Related Applications

This application is related to Provisional Application No. 60/119,841, filed February 12, 1999, and is a continuation-in-part of U.S. Patent Application No. 09/382,402, filed on August 24, 1999, which is in turn related to U.S. Patent Application No. 09/169,963, filed October 13, 1998, now U.S. Patent No. 6,006,318, which is in turn related to U.S. Patent Application No. 08/516,036, filed August 16, 1995, now U.S. Patent No. 5,742,840. The contents of Provisional Application No. 60/119,841, filed February 12, 1999, are hereby incorporated by reference.

Field of the Invention

The present invention relates to cross-bar circuits, and particularly relates to a cross-bar circuit implementing a switch of a broadband processor.

BACKGROUND AND SUMMARY OF THE INVENTION

In digital processing systems, one of the basic operations is rearranging or copying portions of an operand. Prior art systems, such as current general purpose microprocessors (Pentium, MIPS, ARM, SPARC, PowerPC, etc.), implement this basic operation in various special circuits, such as shifters, rotators, field extractors, or byte permuters.

Digital processing is now being extended to new applications such as broadband processing in which very general rearrangements of bits are required to accomplish various sophisticated mathematical algorithms needed for encryption and error correction. Such digital processing requires switching circuits, cross-bar circuits, to help accomplish these algorithms. General switching requires a cross-bar circuit, which conventionally contains more transistors than the specialized shifters, rotator, etc. used in the prior art.

Prior art cross-bar circuits require far more bits of control to specify the mapping between the location of output operand bits and the input operand bits from which they are generated.

The present invention provides a cross-bar circuit that implements a switch of a broadband processor. The present invention describes a system and method for implementing switching operations that perform completely general permutations and copies at the individual bit level within large operands.

The present invention describes a system and method for reducing the transistor count, wire count, and area of a general cross-bar network down to levels comparable to what is achieved in more specialized circuits. This invention involves a novel combination of circuit techniques, including precharged dynamic multiplexors, ground selection, and pseudo-differential sensing against dummy bit lines, which have individually been practiced in digital logic and memory devices.

The present invention implements the system and method for utilizing wide operands further described in commonly-assigned U.S. Patent Application No. 09/382,402 to provide these control bits from a small cache memory physically located close to the cross-bar circuit.

The present invention also describes a system and method for utilizing the cross-bar circuits to perform conventional specialized operations as well. These operations have a great deal of redundancy in their control state. The present invention describes a system and method for generating the large number of control bits of the cross-bar circuit (e.g. typically more than 1,000 bits) from a much smaller number of control bits for conventional operations (e.g. typically fewer than 100 bits). This invention conserves the resources and delays that

would otherwise be incurred in the use of the wide operand memory referred to in the previous paragraph.

The present invention describes a system and method to implement a cross-bar circuit, memory, and control of a switch of a broadband processor, that (1) requires (a) a small area of silicon, (b) very low power, (c) a small number of transistors, (d) a small number of wires, and (e) only two metal layers, (2) operates at a high speed, and (3) has high functionality.

In an exemplary embodiment, the present invention provides a cross-bar circuit that, in response to partially-decoded instruction information and in response to datapath information, (1) allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and (2) provides the ability to set-to-zero any bit in said 2^m-bit output destination word. The cross-bar circuit includes: (1) a **switch circuit** which includes 2^m 2^n:1 multiplexor circuits, where each of the 2^n:1 multiplexor circuits (a) has a unique n-bit (e.g. 8-bit) index input, one disable input, and a 2^n-bit wide source input, (b) receives (i) an n-bit index at the n-bit index input, (ii) a disable bit at the disable input, and (iii) the 2^n-bit input source word at the 2^n-bit wide source input, and (c) decodes the n-bit index either (i) to select and output as an output destination bit one bit from the 2^n-bit input source word if the disable bit has a logic low value or (ii) outputs a logic low as the output destination bit if the disable bit has a logic high value; (2) a **cache memory** that (a) has 2^m cache datapath inputs and 2^m cache index inputs, (b) receives (i) the datapath information on the 2^m cache datapath inputs and (ii) 2^m n-bit indexes on the 2^m cache index inputs, (c) provides a first set of the n-bit indexes for the switch circuit, and (d) includes a small tightly coupled memory array that stores p (e.g. eight) entries of 2^m n-bit indexes for the switch circuit, where the cache memory is logically coupled to the switch circuit; and (3) a **control circuit** that (a) has a plurality (e.g. 100) of control inputs, (b) receives the partially-decoded instruction information on the plurality of control inputs, (c) provides a second set of the n-bit indexes for the switch circuit, and (d) provides the disable bits for the switch circuit, where the control circuit is logically coupled to the switch circuit and to the cache memory.

In an exemplary embodiment, the present invention provides a **switch circuit** that allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and that provides the ability to set-to-zero any bit in the 2^m-bit output destination word. The switch circuit includes 2^m 2^n:1 multiplexor circuits, where each of the 2^n:1 multiplexor circuits (a) has a unique n-bit (e.g. 8-bit) index input and one disable input, (b) decodes an n-bit index received at the n-bit index

input to select one bit from the 2^n-bit input source word if a disable bit received at the disable input has a logic low value, and (c) outputs a logic low if the disable bit has a logic high value. Also, each of the 2^n:1 multiplexor circuits includes: (1) a 2^q:1 pass gate selector (e.g. 4:1 pass gate selector), where the 2^q:1 pass gate selector has 2^q precharge/discharge wire-OR bitline inputs; (2) a sense amplifier logically coupled to the 2^q:1 pass gate selector, where the sense amplifier (a) receives the output of the 2^q:1 pass gate selector and (b) receives a dummy bitline input to allow differential sensing of small swing signals on the wire-OR bitline inputs; and (3) 2^{n-q} unit switch cells (e.g. 64 unit switch cells) per precharge/discharge wire-OR bitline input, where each of the unit switch cells (a) is logically coupled to a wire-OR bitline input of the 2^q:1 pass gate selector, (b) is logically coupled to one of 2^r (e.g. one of 8) active-LOW "SELA" select wires, and (c) is logically coupled to one of 2^{n-q-r} (e.g. one of 8) active-HIGH "SELC" select wires.

In addition, an exemplary implementation of the present invention provides a **control circuit** that provides indexes and disable bits. The control circuit includes: (1) an arithmetic logic unit (ALU), where the ALU includes a plurality of ALU modules; (2) a plurality of multiplexors logically coupled to the ALU, where each of the multiplexors includes a plurality of multiplexor modules; and (3) a plurality of decoders logically coupled to the ALU and to the plurality of multiplexors, where each of the decoders includes a s-stage (e.g. 5 stage) chain of NAND/NOR gates.

Some of the advantage of the cross-bar circuit are as follows:

1. It requires a uniquely small area of silicon, as provided by the arrangement of the switch circuit, the cache memory, and the control circuit;

2. It requires uniquely low power, as provided by the use of a minimum number of logic gates to implement the switch circuit and the control circuit;

3. It requires a small number of transistors, as provided by the use of a minimum number of logic gates to implement the switch circuit and the control circuit;

4. It requires a small number of wires, as provided by the use of a minimum number of logic gates to implement the switch circuit and the control circuit and by the arrangement of the switch circuit, the cache memory, and the control circuit; and

5. It requires only two layers of metal to implement, as provided by the arrangement of the switch circuit, the cache memory, and the control circuit .

Since the switch circuit requires no more silicon area than a barrel shifter and since the switch circuit can perform the functions of a barrel shifter, the switch circuit can be used in place of a barrel shifter in a circuit design, thus saving space in the circuit design.

Also, the control circuit generates a relatively large number of bits (e.g. 1024 bits), which comprise a pattern of bits. This pattern of bits is often repetitive. Taking advantage of the repetitiveness of this pattern of bits, the control circuit can use a relatively small number of bits (e.g. 64 bits) to generate a relatively large number of bits (e.g. 1024 bits). In an exemplary embodiment, the control circuit generates $n^* 2^m$ bits (e.g. 1024 bits) as a pattern of bits by using only 2^{n-2} input bits (e.g. 64 input bits).

THE FIGURES

Figure 1A is a first system level diagram showing the functional blocks of a cross-bar circuit according to an exemplary embodiment of the present invention.

Figure 1B is a second system level diagram showing the functional blocks of a cross-bar circuit according to an exemplary embodiment of the present invention.

Figure 1C is a third system level diagram showing the functional blocks of a cross-bar circuit according to an exemplary embodiment of the present invention.

Figure 2A is a block diagram of the switch circuit according to an exemplary embodiment of the present invention.

Figure 2B is a block diagram of a typical 256:1 multiplexor circuit according to an exemplary embodiment of the present invention.

Figure 2C is a block diagram of a typical 1x256 switch module according to an exemplary embodiment of the present invention.

Figure 2D is a block diagram of a typical 1x128 switch module according to an exemplary embodiment of the present invention.

Figure 2E is a block diagram of a typical 1x32 switch module according to an exemplary embodiment of the present invention.

Figure 3A is a block diagram of a typical 1x8 switch module according to an exemplary embodiment of the present invention.

Figure 3B is a diagram of a typical unit switch cell according to an exemplary embodiment of the present invention.

Figure 4 is a diagram of the cache memory according to an exemplary embodiment of the present invention.

Figure 5A is a block diagram of the control circuit according to an exemplary embodiment of the present invention.

Figure 5B is a block diagram of a typical control slice according to an exemplary embodiment of the present invention.

Figure 5C depicts gate-level diagrams of a typical Arithmetic Logic Unit (ALU) bit when a physical address bit input to the ALU bit is "0" according to an exemplary embodiment of the present invention.

Figure 5D depicts gate-level diagrams of a typical Arithmetic Logic Unit (ALU) bit when a physical address bit input to the ALU bit is "1" according to an exemplary embodiment of the present invention.

Figure 5E depicts gate-level diagrams of a typical ALU bit when a physical address bit input to the ALU bit is "0" according to an exemplary embodiment of the present invention.

Figure 5F depicts gate-level diagrams of a typical ALU bit when a physical address bit input to the ALU bit is "1" according to an exemplary embodiment of the present invention.

Figure 5G is a circuit diagram of a prior art CMOS transfer-gate.

Figure 5H is a gate-level diagram of a MUX module according to an exemplary embodiment of the present invention.

Figure 5I depicts gate-level diagrams of two MUX modules according to an exemplary embodiment of the present invention.

Figure 6 is a gate-level diagram of an ALU, 8x 2:1 MASK multiplexor, and SCALE multiplexor arrangement according to an exemplary embodiment of the present invention.

Figure 7 is a block diagram of a MSB/LSB decoder according to an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Cross-bar Circuit

Referring first to Figure 1A, an exemplary embodiment of a cross-bar circuit 100 which achieves the objectives of the present invention is illustrated in block diagram form. Cross-bar circuit 100 is organized as a 128-bit wide 256:1 multiplexor. Cross-bar circuit 100, in response to partially-decoded instruction information and in response to datapath information, (1) allows any bit from either of two 128-bit input source words, SOURCE M and SOURCE L, to be switched into any bit position of a 128-bit output destination word,

DESTINATION and (2) provides the ability to set-to-zero any bit in the 128-bit output destination word, DESTINATION. Although this exemplary embodiment and others discussed thereafter refer to exemplary bit sizes for inputs, outputs, and circuit components, these bit sizes are merely examples and are not intended to limit the invention to the exemplary bit sizes.

In an exemplary embodiment, cross-bar circuit 100, in response to partially-decoded instruction information and in response to datapath information, (1) allows any bit from a 256-bit input source word, comprised of a 128-bit input source word SOURCE M and a 128-bit input source word SOURCE L, to be switched into any bit position of a 128-bit output destination word DESTINATION and (2) provides the ability to set-to-zero any bit in the 128-bit output destination word DESTINATION.

Referring next to Figure 1B, an exemplary embodiment of cross-bar circuit 100 is shown in block diagram form in greater detail than Figure 1A. In this exemplary embodiment, cross-bar circuit 100 includes a control circuit 105, a cache memory 110, and a switch circuit 115. In an exemplary embodiment, control circuit 105 (i) receives partially decoded instruction information on 128 control inputs and (ii) generates 128 8-bit indexes and 128 disable bits. In an exemplary embodiment, cache memory 110 (i) receives datapath information on 128 cache datapath inputs, (ii) receives 128 8-bit indexes from control circuit 105, (iii) can generate its own 128 8-bit indexes, and (iv) can store at least 128 8-bit indexes. In an exemplary embodiment, switch circuit 115 (i) receives 128 8-bit indexes, generated by either control circuit 105 or cache memory 110, (2) receives 128 disable bits generated by control circuit 105, (3) receives a 128-bit input source word SOURCE M and a 128-bit input source word SOURCE L, and (4) generates a 128-bit output destination word DESTINATION in response to the received 128 8-bit indexes and the received 128 disable bits.

Referring next to Figure 1C, the exemplary embodiment of cross-bar circuit 100 is illustrated in still greater detail in block diagram form. In this exemplary embodiment, cross-bar circuit 100 includes a control top 120, a cache top 130, a switch circuit 140, a cache bottom 150, and a control bottom 160. In this exemplary embodiment, cross-bar circuit 100 is comprised of three basic building blocks: a switch circuit 140, a cache memory, and a control circuit. In this embodiment, the control circuit is physically divided into the following two blocks: (i) control top 120 which is physically located above switch circuit 140; and (ii) control bottom 160 which is physically located below switch circuit 140. Also in this embodiment, the cache memory is physically divided into the following two blocks: (i)

cache top 130 which is physically located above switch circuit 140; and (ii) cache bottom 150 which is physically located below switch circuit 140. In addition, in this embodiment, switch circuit 140 includes decoders, drivers, sense amplifiers, and switch modules. In this embodiment, the decoders, drivers, and sense amplifiers are physically divided into the following two blocks: (i) a decoders, drivers, and sense amplifiers top 142 which is physically located above the switch modules; and (ii) a decoders and drivers bottom 148 which is physically located below the switch modules. In this embodiment, the switch modules are physically divided into the following two blocks: (1) a switch modules top 144; and (2) a switch modules bottom 146, which is physically located below switch modules top 144.

In a preferred embodiment, the circuit design and layout of the cross-bar circuit 100 can be completely implemented in a first metal layer, M1, and a second metal layer, M2. This permits the cross-bar circuit to be placed under global bussing connecting other circuits in a datapath logically coupled to cross-bar circuit 100, where the global bussing for the datapath is implemented in metals layers located physically higher than M1 and M2, such as a third metal layer, M3, and a fourth metal layer, M4. In an exemplary embodiment, the global bussing is implemented vertically in M3 and M4. In sum, the entire circuit layout of cross-bar circuit 100 can be implemented in only M1 and M2 so that cross-bar circuit 100 can fit under the M3 and M4 global bussing.

Switch Circuit

Referring to Figure 2A, an exemplary embodiment of a switch circuit 200 which helps to achieve the objectives of the present invention is illustrated in block diagram form. In this exemplary embodiment, switch circuit 200 includes the following components, logically interconnected as shown:

(1) 128 256:1 multiplexor circuits 202-204, referred to as 256:1 multiplexor circuit 0 - 256:1 multiplexor circuit 127;
(2) 128 8-bit wide index input lines 206-208;
(3) 128 disable input lines 209-211;
(4) a 128-bit wide SOURCE M input line 213;
(5) a 128-bit wide SOURCE L input line 214; and
(6) 128 DESTINATION output lines 216-218.

Each of the 256:1 multiplexor circuits 202-204 (1) can also be referred to as a bit-slice of switch circuit 200 and (2) has (a) a unique 8-bit index input logically coupled to an 8-bit wide

index input line and (b) a unique disable input logically coupled to a disable input line. Each of the 256:1 multiplexor circuits 202-204 (1) receives a 256-bit input source word in the form of a 128-bit input source word SOURCE M on 128-bit wide SOURCE M input line 213 and a 128-bit input source word SOURCE L on 128-bit wide SOURCE L input line 214 and (2) receives an 8-bit index at its 8-bit index input and a disable bit at its disable input. Each of the 256:1 multiplexor circuits 202-204 decodes the received 8-bit index to do the following: (1) either select and output as an output destination bit, DESTINATION[x], one bit from the 256-bit input source word if the disable bit has a logic low value (2) or output a logic low as the output destination bit, DESTINATION[x], if the disable bit has a logic high value.

256:1 Multiplexor Circuit

Referring next to Figure 2B, an exemplary embodiment of a typical 256:1 multiplexor circuit 220, referred to as 256:1 multiplexor circuits 202-204 in Figure 2A, is illustrated in block diagram form. In this exemplary embodiment, 256:1 multiplexor circuit 220 includes the following components, logically interconnected as shown:

(1) 128-bit wide SOURCE M input line 213;
(2) 128-bit wide SOURCE L input line 214;
(3) eight index input lines 222-229, referred to as INDEX[0] - INDEX[7];
(4) a disable input line 230;
(5) a dummy-bitline 232;
(6) a 1-of-4 decoder 234;
(7) a SELC 1-of-8 decoder 235;
(8) a SELA 1-of-8 decoder 236;
(9) an 8-bit wide SELC select line 237;
(10) an 8-bit wide SELA select line 238;
(11) four wire-OR bitlines 240-243;
(12) a 1x256 switch module 245;
(13) a 2-bit wide bitline select line 246;
(14) a 4:1 pass gate selector 247;
(15) a sense amplifier (sense amp) 248; and
(16) a DESTINATION[x] output line 249.

In an exemplary embodiment, four index input lines enter at the top of 256:1 multiplexor circuit 220, while the remaining four index input lines enter at the bottom of 256:1 multiplexor circuit 220. In an embodiment, index input lines 222-225, INDEX[0] - INDEX[3], enter at the top of 256:1 multiplexor circuit 220, while index input lines 226-229,

INDEX[4] - INDEX[7], enter at the bottom of 256:1 multiplexor circuit 220. In an exemplary embodiment, SELC 1-of-8 decoder 235 is physically located at the top of 256:1 multiplexor circuit 220, and SELA 1-of-8 decoder 236 is physically located at the bottom of 256:1 multiplexor circuit 220.

In an exemplary embodiment, 1-of-4 decoder 234 is physically located at the top of 256:1 multiplexor circuit 220. 1-of-4 decoder 234 decodes one index bit, INDEX[3], bussed up from the bottom of multiplexor circuit 220 and another index bit, INDEX[4], from the top of multiplexor circuit 220 (a) to select one of the four wire-OR bitlines 240-243 and (b) to pass the value on the selected wire-OR bitline to 4:1 pass gate selector 247. In an exemplary embodiment, 4:1 pass gate selector 247 is physically located at the top of multiplexor circuit 220. In an exemplary embodiment, sense amplifier 248 is physically located at the top of multiplexor circuit 220,. A signal on dummy-bitline 232 is passed into sense amplifier 248. Sense amplifier 248 uses the signal on dummy-bitline 232 to perform differential sensing and to accurately sense the output of 4:1 pass gate selector 247, especially small signal swings.

1x256 Switch Module

Referring next to Figure 2C, an exemplary embodiment of a typical 1x256 switch module 250, referred to as 1x256 switch module 245 in Figure 2B, is illustrated in block diagram form. In this exemplary embodiment, 1x256 switch module 250 includes the following components, logically interconnected as shown:

(1) a 128-bit wide SOURCE M input line 213, referred to as SRCM_BM<127..0>;

(2) a 128-bit wide SOURCE L input line 214, referred to as SRCL_BM<127..0>;

(3) an 8-bit wide SELC select line comprised of (a) a first 4-bit SELC select line 252, referred to as SELC_BM<3..0>, and (b) a second 4-bit wide SELC select line 253, referred to as SELC_BM<7..4>;

(4) an 8-bit wide SELA select line comprised of (a) a first 4-bit SELA select line 255, referred to as SELA_BNM<3..0>, and (b) a second 4-bit wide SELA select line 256, referred to as SELA_BNM<7..4>;

(5) four wire-OR bitlines 240-243, referred to as BL_BNM<0> - BL_BNM<3>;

(6) a first 1x128 switch module 258; and

(7) a second 1x128 switch module 259.

The operation of the 1x256 switch module will be described later.

1x128 Switch Module

Referring next to Figure 2D, an exemplary embodiment of a typical 1x128 switch module 260, referred to as 1x128 switch modules 258 and 259 in Figure 2C, is illustrated in block diagram form. In this exemplary embodiment, 1x128 switch module 260 includes the following components, logically interconnected as shown:

(1) four wire-OR bitlines 240-243, referred to as BL_BNM<0> - BL_BNM<3>;

(2) four 16-bit wide right input source lines 261-264, referred to as SRCRT_BM<15..0>, SRCRT_BM<31..16>, SRCRT_BM<47..32>, and SRCRT_BM<63..48>;

(3) four 16-bit wide left input source lines 265-268, referred to as SRCLF_BM<15..0>, SRCLF_BM<31..16>, SRCLF_BM<47..32>, and SRCLF_BM<63..48>;

(4) a 3-bit wide right SELC select line 269, referred to as SELCRT_BM<3..0>;

(5) a 3-bit wide left SELC select line 270, referred to as SELCLF_BM<3..0>;

(6) first 4-bit SELA select line 255, SELA_BNM<3..0>;

(7) second 4-bit wide SELA select line 256, referred to as SELA_BNM<7..4>; and

(8) four 1x32 switch modules 271-274.

The operation of the 1x128 switch module will be described later.

1x32 Switch Module

Referring next to Figure 2E, an exemplary embodiment of a typical 1x32 switch module 275, referred to as 1x32 switch modules 271-274 in Figure 2D, is illustrated in block diagram form. In this exemplary embodiment, 1x32 switch module 275 includes the following components, logically interconnected as shown:

(1) four wire-OR bitlines 240-243, referred to as BL_BNM<0> - BL_BNM<3>;

(2) four 4-bit wide right input source lines 276-279, referred to as SRCRT_BM<3..0>, SRCRT_BM<7..4>, SRCRT_BM<11..8>, and SRCRT_BM<15..12>;

(3) four 4-bit wide left input source lines 280-283, referred to as SRCLF_BM<3..0>, SRCLF_BM<7..4>, SRCLF_BM<11..8>, and SRCLF_BM<15..12>;

(4) a right SELC select line 284, referred to as SELCRT_BM;

(5) a left SELC select line 285, referred to as SELCLF_BM;

(6) first 4-bit SELA select line 255, referred to as SELA_BNM<3..0>;

(7) second 4-bit wide SELA select line 256, referred to as SELA_BNM<7..4>; and

(8) four 1x8 switch modules 286-289.

The operation of the 1x32 switch module will be described later.

1x8 Switch Module

Referring next to Figure 3A an exemplary embodiment of a typical 1x8 switch module 300, referred to as 1x8 switch modules 286-289 in Figure 2E, is illustrated in block diagram form. In this exemplary embodiment, 1x8 switch module 300 includes the following components, logically interconnected as shown:

(1) a wire-OR bitline 302, BL_BNM, corresponding to one of the four wire-OR bitlines 240-243 in Figure 2E;

(2) four right input source lines 304-307, referred to as SRCRT_BM<0> - SRCRT_BM<3>, where the set SRCRT_BM<0>-SRCRT_BM<3> is exemplary of the four sets SRCRT_BM<3..0>, SRCRT_BM<7..4>, SRCRT_BM<11..8>, and SRCRT_BM<15..12> in Figure 2E;

(3) four left input source lines 308-311, referred to as SRCLF_BM<0> - SRCLF_BM<3>, where the set SRCLF_BM<0>-SRCLF_BM<3> is exemplary of the four sets SRCLF_BM<3..0>, SRCLF_BM<7..4>, SRCLF_BM<11..8>, and SRCLF_BM<15..12> in Figure 2E;

(4) right SELC select line 284, referred to as SELCRT_BM;

(5) left SELC select line 285, referred to as SELCLF_BM;

(6) four SELA select lines 313-316, SELA_BNM<0> - SELA_BNM<3>, which form a set of four SELA select lines corresponding to SELA_BNM<3..0> in Figure 2E;

(7) four SELA select lines 317-320, SELA_BNM<4> - SELA_BNM<7>, which form a set of four SELA select lines corresponding to SELA_BNM<7..4> in Figure 2E; and

(8) eight unit switch cells 322-329.

The operation of the 1x8 switch module will be described later.

Unit Switch Cell

Referring next to Figure 3B an exemplary embodiment of a typical unit switch cell 340, referred to as unit switch cells 322-329 in Figure 3A, is illustrated in block diagram form. In this exemplary embodiment, unit switch cell 340 includes the following components, logically interconnected as shown:

(1) a first NFET transistor 342;

(2) a second NFET transistor 344,

(3) a SELA select line 346, referred to as SELA_BNM, which is exemplary of one of the eight SELA select lines 313-320 in Figure 3A;

(4) a SELC select line 347, referred to as SELC_BM, which is exemplary of one of the two SELC select lines 284 or 285 in Figure 3A;

(5) a SRC source line 350, referred to as SRC_BM, which is exemplary of one of the eight input source lines 304-311, referred to as SRCRT_BM<0> - SRCRT_BM<3> and SRCLF_BM<0> - SRCLF_BM<3 in Figure 2E; and

(6) wire-OR bitline 302, referred to as BL_BNM, also in Figure 3A.

If SELA select line 346 has a logic low value and if SELC select line 347 has a logic high value, then the logic value on SRC source line 350 will be transferred to wire-OR bitline 302. In other words, if SELA select line 346 has a logic low value, if SELC select line 347 has a logic high value, and if SRC source line 350 has a logic high value, then wire-OR bitline 302 will have a logic high value. If SELA select line 346 has a logic low value, if SELC select line 347 has a logic high value, and if SRC source line 350 has a logic low value, then wire-OR bitline 302 will have a logic low value. When SELA select line 346 has a logic low value and if SELC select line 347 has a logic high value, then unit switch cell 340 is selected and the logic value on if SRC source line 350 will be transferred to then wire-OR bitline 302.

1x8 Switch Module

In an exemplary embodiment, in the layout of 1x8 switch module 300 eight unit switch cells 322-329 are grouped together in order to meet the following objectives:

1. Fit in the vertical pitch of eight M2 source SRC data input lines;
2. Fit in the practical minimum number of vertical M1 wire pitches; and
3. Within the constraints of #1 and #2, provide the maximum transistor width for the series NFETs, 342 and 344.

Objective #1 is driven by the desire to minimize the height of the XLU structure.

Objective #2 seeks to minimize the bit-pitch of the datapath. The driving limitation is the number of vertical wires needed to accomplish the 1-of-256 wire-OR select plus any

overhead needed for interconnection and data sensing. The minimum number of select lines that could be used would be 16: 3 groups of 1-of-4 hot busses and one group of 4 wire-OR bitlines. However, this would require the unit switch cell 340 to use three series connected NFETs which would require additional interconnection overhead and give lower read current for a given transistor width. The preferred embodiment uses two 1-of-8 hot busses for the SELA and SELC lines, and 4 separate wire-OR bitlines (plus one dummy bitline) which go through a 4:1 MUX for the final selection. The 4:1 MUX is placed outside the switch and thus theoretically does not require any additional wire tracks within the array of switch cells to bus the select wires for the MUX. This gives a total of 21 vertical wires for data selection and sensing. Another 4 M1 tracks plus 2 spare tracks were needed for connection of the horizontal M2 "SRC" wires to the gates of the data-in transistors. Finally, 2 M1 Vss wires were needed to shield the small swing bitlines from the full swing select lines and to allow for P-well taps in the array.

Objective #3 attempts to maximize the "read" current of unit switch cell 340 by obtaining the largest possible transistor width. The limiting factor here was the number of transistor that could be stacked in the height of the cell - 8 M2 pitches.

In an exemplary embodiment, by putting the wire-OR bitlines in the middle of the cell with four of the eight NFET transistor pairs on each side, a much superior layout of 1x8 switch module 300 is achieved, as shown in Figure 3A.

Now minimum M2 wires could be used and the transistor width was now limited by the stack of four parallel SELC NFETS, giving a maximum transistor width of:

W = {8*(M2 pitch) - 2*(diff sp) - (diff sp with poly end cap) - (diff sp with contacted poly)} / 4

W = {8*(0.60um) - 2*(0.30um) - (0.225um + 0.105um) - (2*(0.12um+0.105um)+0.225um)} / 4

W = 0.79875um ==> 0.795um.

One other consideration in the design and layout of unit switch cell 340 was which NFETs should be connected to which signal. Referring to Figure 3B, the choice of placing the SELC transistor 342next to wire-OR bitline 302 was made to both minimize the capacitance on the bitline 302 and to make the capacitance non-varying. The problem was that any data combination can be present on the data-in wires, ranging from all high to all low. If the data-in transistors were placed next to bitline 302, then bitline 302 would see the channel and source terminal capacitance of every data-in transistors whose data-in signal was high. This would make the bitline capacitance higher and also variable as a function of the state of the data-in signals. Since the read current of unit switch cell 340 is limited by the

transistor size, it was deemed better to put the variable capacitance on SELA line 346 where the signal drivers can be made large enough to handle the higher capacitance.

Each 1x8 switch module 300 has bits 0-3 of the 8 SELA wires connected on the left side of the cell to four SELA select lines 313-316, SELA_BNM<0> - SELA_BNM<3>, and bits 4-7 of the 8 SELA wires connected on the right side of the cell to four SELA select lines 317-320, SELA_BNM<4> - SELA_BNM<7>. This means that bits 0-3 of the SELA lines are gated by a different SELC signal than bits 4-7 of the SELA lines. Thus a 1x8 switch module can only select 1-of-4 data inputs depending on whether the left or right SELC signal is "High".

1x32 Switch Module

Referring to Figure 2E, 1x32 switch module 275 includes four 1x8 switch modules 286-289, with 1x8 switch module 286 connected to wire-OR bitline<0> 240, 1x8 switch module 287 connected to wire-OR bitline<1> 241, 1x8 switch module 288 connected to bitline<2> 242, and 1x8 switch module 289 connected to bitline<3> 243.

1x128 Switch Module

Referring to Figure 2D, 1x128 switch module 260 includes four 1x32 switch modules 271-274, with the 1x32 switch module 271 having the left SELC input connected to bit<0> of the left group of 4 SELC wires 270 and the right SELC input connected to bit<0> of the right group of 4 SELC wires 269. 1x32 switch module 272 has the SELC inputs connected to bit<1> of the left and right groups of 4 SELC wires, and so forth for the third and fourth cells. At this point, switch circuit 200 has decoded 128 inputs (32 inputs on each of 4 wire-OR bitlines) where the "left" 64 inputs (SRCLF) are selected by the intersection of the "left" 4 SELA bits and the "left" 4 SELC bits, and the "right" 64 inputs are selected by the intersections of the "right" 4 SELA bits and the "right" 4 SELC bits. However, since SELA and SELC are both 1-of-8 hot busses, this only covers half of the possible combinations because when the "hot" SELA wire is on the left side, the "hot" SELC wire could be on the right side and vice-versa.

1x256 Switch Module

Referring to Figure 2C, 1x256 switch module includes two 1x128 switch modules, 258 and 259, where the left and right SELC groups of 4 bits are reversed (or twisted) between the two 1x128 switch modules. In this manner, the 1x128 switch module 259 decodes the 128 combinations missed by 1x128 switch module 258. The hierarchy of this programming scheme is that the SELA bits have the least significance, then the 1-of-4 wire-OR bitline select, and finally the SELC bits have the highest significance. This hierarchy is driven by the

fact that the 8 index bits for each 256:1 multiplexor circuit are physically divided with 4 index bits coming from the bottom of cross-bar circuit 100 and the other 4 coming from the top of the cross-bar circuit 100. This physical division is in conflict with the decoding requirements that want the index bits divided into two groups of 3 bits for the two 1-of-8 decoders and one group of 2 bits for the 1:4 decoder 234. This dictated the decision to place the SELA decoder 236 at the bottom of the slice, and the 1:4 decoder 234 and the SELC decoder 235 at the top of the slice. Thus the first 3 of the 4 bottom index bits go to the SELA decoder 236. The 4th bit is bussed through the column (using one of two spare M1 tracks) and combined with the 1st bit of the 4 top index bits to drive the 1:4 decoder 234. Finally, the last 3 of the 4 top index bits go to the SELC decoder 235.

The result of the "left"/"right" programming of the SELA and SELC wires is that the physical mapping of the SRCL and SRCM data inputs is not a simple linear progression from the bottom to the top of the wire-OR bitlines. Instead, the mapping jumps back and forth between the top and bottom halves of the column in groups of 4 inputs bits.

Consider the case where bitline<0> is selected and SELC<0> is active (i.e. - high). The SELC<0> wire is connected to the **left** side of the bottom 4 "1x8" cells in the **bottom half** of the column and is connected to the **right** side of the bottom 4 "1x8" cells in the **top half** of the column. Therefore, as the SELA address is incremented, first the 4 **left** data-inputs in the bottom 4 "1x8" cells of the **bottom half** of the column are sequentially selected as SELA bits 0-3 are toggled. Then after we jump from SELA<3> to SELA<4>, the 4 **right** data-inputs in the bottom 4 "1x8" cells of the **top half** are sequentially selected as SELA bits 4-7 are toggled. Since bitline<0> is selected, this means that we sequentially select the first 4 **left** data-inputs from the **bottom half** of the column and then the first 4 **right** data-inputs from the **top half** of the column. If we then move from bitline<0> to bitline<1> and wrap SELA back to bit<0>, we next sequentially select the second 4 **left** data-inputs from the **bottom half** of the column and then the second 4 **right** data-inputs from the **top half** of the column.

This bottom-left-4/top-right-4 sequence continues as the "index" is incremented until we reach the end of the first set of 128 data-inputs and is reflected in the bit-ordering of the SRCL data-inputs in the schematic. At this point, when we jump from SELC<3> to SELC<4>, the pattern reverses and we next select the 4 **left** data-inputs from the bottom "1x8" cell of the **top half** of the column followed by the 4 **right** data-inputs from the bottom "1x8" cell of the **bottom half** of the column. This top-left-4/bottom-right-4 sequence

continues through the second set of 128 data-inputs and is reflected in the bit-ordering of the SRCM data-inputs in the schematic.

Dummy Bitline

Referring to Figure 2B, because the wire-OR bitlines are relatively high in capacitance and the unit switch cell has limited read current, it is desirable to use small signal swing on the wire-OR bitlines 240-243 to improve the speed performance. This also implies that it is desirable to use differential sensing of some form, but the bitlines are only single-end. Thus a dummy bitline 232 for use in sensing is needed. Only one dummy bitline 232 is needed for all 4 wire-OR bitlines since only one wire-OR bitline is selected for evaluation by the sense amp.

The critical design/layout objective for the dummy bitline 232 is to have it match the wire-OR bitlines 240-243 as closely as possible for both capacitance and read current. Thus we must first evaluate the electrical properties of the wire-OR bitlines.

Each wire-OR bitline is connected to 64 unit switch cells - 8 unit cells per "1x8" cell and 8 "1x8" cells per bitline. As was discussed above, the SELC transistor in the switch cell is connected to the bitline. Therefore, since the SELC bus is 1-of-8 hot, the bitline sees 8 unit cells where the SELC transistor is on, exposing the bitline to the capacitance of the intermediate node of the switch cell and to the gate edge capacitance of the data-in transistor.

In the other 56 unit cells the SELC transistor is off and the bitline only sees the gate edge capacitance of the SELC transistor.

For reading, the 8 unit switch cells that are selected have 7 cells where the SELA input is high. In these cells, the state of the data-input is essentially irrelevant since there is no discharge path from the bitline. In the 8th cell, the SELA input is low and thus if the data-input is high, the transistor turns on and provides a discharge path from the "high" bitline to the "low" SELA line.

In order to duplicate this configuration in the dummy bitline, a "dummy 1x8" cell is added at the bottom of the column. This dummy cell is similar to the standard "1x8" cell except for three details. First, the unit cell has a stack of four transistors rather than the two transistors in the standard switch cell. This is done to have the dummy cell provide half the read current of the standard cell. Second, the output of the cell is connected to the dummy bitline instead of to one of the 4 wire-OR bitlines. Third, all the gates in the dummy cell are tied to Vdd rather than to the SELC and data wires. In this manner the dummy "1x8" cell provides a discharge path from the dummy bitline to the "low" SELA line in the same manner

as the standard "1x8" cell, but at half the current. Furthermore, the dummy cell nearly duplicates the capacitance of the 8 unit switch cells on the normal wire-OR bitline that are selected by the "high" SELC line.

All that remains now is to duplicate the capacitance of the 56 unit cells on the wire-OR bitline that have the SELC transistor off. Note that in the length of an entire column, the dummy bitline crosses 32 "1x8" cells. Thus two "dummy" NFETs with their gates tied to Vss are included in the standard "1x8" cell that can have their drains connected to the dummy bitline. The layout of these transistors match the transistor width and drain diffusion area and perimeter of the SELC transistor in the unit switch cell. At the level (4 "1x8" cells), 7 of the 8 available dummy transistors is connected to the dummy bitline, giving a total of 56 dummy transistors on the dummy bitline.

At this point, with the dummy bitline placed in the middle of the wire-OR bitlines, the dummy bitline now matches the wire-OR bitlines in wire length and nearest neighbor coupling, discharge path, transistor overlap capacitance, and drain junction capacitance.

Cache Memory

Referring to Figure 1B, cache memory 110 is the first of two sources for the index inputs to the switch. In an exemplary embodiment, the cache memory is a small tightly coupled memory array that stores eight entries of 128 8-bit indexes for the switch (e.g. - 8 1024 bit words). The cache can be used to store random switching patterns provided by a datapath logically coupled to cross-bar circuit 100 or to capture index patterns generated by the instruction decode logic logically coupled to cross-bar circuit 100. These patterns can then be used repetitively without the datapath having to incur the overhead of repeatedly generating these patterns. The instruction decode logic is described **in Appendix 2 - Instruction Decode Logic**.

In an exemplary embodiment, as depicted in Figure 4, the "cache" includes 8 lines, 402-404, of 1024 bits - 8 columns per bit-slice of the MUX. As depicted in Figure 4, the each cache line 402-404 is divided into 128 cache slices 408-412, where each cache slice 408-412 has 8 bits - one column per index bit of the switch circuit 200. In other words, the 1024 bits of the cache line are bundled into 128 cache slices of eight bits each corresponding to the eight index bits for each 256:1 multiplexor circuit 220.

In an exemplary embodiment, cache memory 110 is physically divided into two SRAM arrays each having 8 rows of 512 columns - 128 slices" of 4 columns each. The arrays are pitch-matched to the switch circuit 115 such that one 4 column slice in the cache matches

up with one 256:1 multiplexor circuit 220 in the switch circuit 200. One array is physically located immediately below the switch circuit 140 and one is physically located immediately above the switch circuit 140, as shown in Figure 1C, thus providing a total of 8 index bits per 256:1 multiplexor circuit 220, bits 7-4 from cache top 130 and bits 3-0 from cache bottom 150.

Control Circuit

Referring to Figure 1B, the control circuit 105 is the second source for the index inputs to the switch circuit 115. It is also the source for the 128 disable inputs to the switch circuit 115. In an exemplary embodiment, the control circuit 105 takes partially decoded instruction information from datapath control logic logically coupled to cross-bar circuit 100 and completes the decoding into an appropriate 8-bit index and disable bit for each of the 128 256:1 multiplexor circuits 202-204 in the switch circuit 200. This decoding implements the cross-bar operations provided in the ISA such as rotate, shift, swap, shuffle, deal, and extract. The cross-bar operations provided in the ISA are included in **Appendix 1 - Cross-bar Operations Provided in the ISA**. The output of this control circuit 105 is passed through the cache memory 110 on its way to the switch circuit 115, thus allowing the indexes to be captured by the cache memory 110 for subsequent cross-bar operations. Note that the disable bits cached by cache memory 110.

The control circuit 105 generates 128 8-bit indexes for the switch circuit 115 and the cache memory 110 and generates 128 disable bits for the switch circuit 115. As shown in Figure 5A, in an exemplary embodiment, the control circuit 500 includes 128 nearly identical control slices 501-503. The control slices 501-503 (i) receive partially-decoded instruction information and (ii) in response to the received partially-decoded instruction information, generate 128 8-bit indexes, INDEX 0 - INDEX 127 and 128 disable bits, DISABLE 0 - DISABLE 127.

As shown in Figure 5B, in an exemplary embodiment, each control slice 510 includes an arithmetic logic unit (ALU) 512, an 8x 2:1 MASK multiplexor 514, an 8x 2:1 INDEX multiplexor 516, a Most Significant Bit (MSB) decoder 518, a Least Significant Bit (LSB) decoder 519, and a SCALE decoder 521. In an exemplary embodiment, each control slice 510 has unique programming of the decoders and the ALU/MUX inputs based on the bit position of the control slice 510.

ALU

In an exemplary embodiment, a main index for each control slice 510 is calculated by an ALU 512 which adds the 8-bit input IAdd0/IAdd1 amount to the physical location of the control slice to get the main index for the source bit. This function thus supports ISA commands such as shift and rotate.

8x 2:1 MASK Multiplexor

In an exemplary embodiment, the output of the ALU 512 then flows through 8x 2:1 MASK multiplexor 514 which selects either the ALU 8-bit output or a "mask" value provided by the IConst0/IConst1 8-bit inputs. The selection choice is set by the LSB and MSB decoders 518 and 519 and the two Select_IMask bits. Four cases are supported:

1. If neither of the two Select_IMask bits are set, then the main index from the ALU 512 is selected in all 128 control slices 501-503, causing 128 contiguous bits in the source field to be indexed;
2. If the Select_IMask "1" bit is set, then the "mask" input is selected for all control slices 501-503 where the output of the MSB decoder 518 is "high";
3. If the Select_IMask "0" bit is set, then the "mask" input is selected for all control slices 501-503 where the output of the LSB decoder 519 is "high"; and
4. If both the Select_IMask "0" **and** "1" bits are set, then the "mask" input is selected for all control slices 501-503 where the output of **either** the LSB decoder 519 or the MSB decoder 518 is "high".

MSB Decoders and LSB Decoders

The MSB and LSB decoders 518 and 519 are used to identify leading edge and/or trailing edge bits in bit-groups of the cross-bar circuit output. These bit groups may be nibbles, bytes, doublets, quadlets, octlets, or the full hexlet. The outputs of the MSB and LSB decoders 518 and 519 are used in each control slice 510 to do one or more of the following:

1. Select the "mask" index value provided on the IConst0/IConst1 8-bit inputs;
2. Force the index value of the control slice to address the upper input hexlet, SOURCE M, of the cross-bar circuit; and
3. Set the "Off" bit, or disable bit, to "0" which in turn forces the corresponding output bit of the cross-bar circuit to a "0".

IADDAM & ICONST Programming

In an exemplary embodiment, there are two copies of the IAdd and IConst buses. The "0" or "1" copy of each bit of these buses is connected to ALU 512 and the 8x 2:1 MASK multiplexor 514 depending on whether the corresponding bit of the physical address of the control slice is a "0" or a "1". Thus IAdd0 and IConst0 are connected to the ALU 512 and 8x 2:1 MASK multiplexor 514 inputs in the "even" control slices, while IAdd1 and IConst1 are connected to the ALU 512 and 8x 2:1 MASK multiplexor 514 inputs in the "odd" control slices. IAdd0 and IConst0 are connected to the ALU 512 and 8x 2:1 MASK multiplexor 514 inputs in the even pairs of control slices (0&1, 4&5, 8&9, etc) while IAdd1 and IConst1 are connected to the ALU 512 and 8x 2:1 MASK multiplexor 514 inputs in the odd pairs of control slices (2&3, 6&7, 10&11, etc).

In an exemplary embodiment, generally, IAdd1 and IConst1 are set to the same values as IAdd0 and IConst0, thus supporting identical shift calculations for all 128 indexes outputted from control circuit 100. However, some operations require the splitting of these values, usually with IAdd1 and IConst1 being the inverse of IAdd0 and IConst0.

8x 2:1 INDEX Multiplexor

In an exemplary embodiment, the output of the 8x 2:1 MASK multiplexor 514 then flows through the 8x 2:1 INDEX multiplexor 516 which selects either the calculated/masked index or the output of the SCALE decoder 521. This selection is done on a bit-by-bit basis within the control slice 510 as determined by the corresponding bits of the ISize 8-bit input bus. The value on the ISize 8-bit bus is a thermometer value which select the IMask output in the lower order bits and the IScale value in the higher order bits of the index.

In an exemplary embodiment, the SCALE decoder 521 takes a 7-bit field from the IMux_Select bus for each bit of the index and examines selected bits from that field. The bits that are examined are determined by the physical address of the control slice 510. If the physical address of the control slice 510 has a minority of "ones", then the bit positions of the 7-bit IMux_Select field that correspond to the "ones" in the 7-bit physical address are ANDed in the scale decoder. Conversely, if the physical address of the control slice 510 has a minority of "zeros", then the bit positions of the 7-bit IMux_Select field that correspond to the "zeros" in the 7-bit physical address are NANDed in the SCALE decoder 521.

Kill Bits

In an exemplary embodiment, bits 0-5 outputted by 8x 2:1 INDEX multiplexor 514 in each control slice 510 are output directly to the cache memory 110 but bits 6 and 7 go

through some additional processing. Both bits 6 and 7 can be overridden by the Kill_Bit6 and Kill_Bit7 inputs, which globally force their respective index bits to "zero".

Select Merge

In an exemplary embodiment, bit 7 sees one final level of processing. This level of processing forces bit 7 of the index to a "one" depending on the state of the two Select_Merge bits and the outputs of MSB decoder 518 and LSB decoder 519. Four cases are supported:

1. If neither of the Select_Merge bits are set, then bit 7 of the index is taken from the SCALE decoder 521 and Kill_Bit7 logic;

2. If Select_Merge "0" bit is set, all bits at or below the LSB Mask decoder 519 decode break point are selected from SOURCE M and all bits above the break point are selected from SOURCE L;

3. If Select_Merge "1" bit is set, all bits below the MSB Mask decoder 518 decode break point are selected from SOURCE L and all bits at or above the break point are selected from SOURCE M; and

4. If both the Select_Merge "0" **and** bits are set, all bits at or below the LSB Mask decoder 519 decode break point and at or above the MSB Mask decoder 518 decode break point are selected from SOURCE M. All bits between the LSB decoder 519 decode break points and the MSB decoder 518 decode break points are selected from SOURCE L.

Force Off

In an exemplary embodiment, the final function in the control slice 510 is the "Force Off" logic which determines the state of the "Off" output bit. This "Off" output bit passes over the cache memory 110 and sets the state of the disable bit in the 256:1 multiplexor circuit 220.

The state of the "Off" bit is determined by the state of the Select_Enable bits and the MSB and LSB decoders 518 and 519 in a manner identical to the Select_Merge override of bit 7, but with the additional provision that the upper 64 control slices of the control circuit 500 force the "Off" bit to a "one" if the Force_Off input is set.

Layout Information

In an exemplary embodiment, the control circuit 105 is divided into two blocks of 128 control slices 502-504, each in a manner similar to the cache memory 110. One block, control bottom 160 is placed immediately below the cache bottom 150 and generates the lower four index bit, INDEX[0] - INDEX[3]. The other block, control top 120 is placed immediately

above the cache top 130 and generates the upper four index bits, INDEX[4] - INDEX[7] plus the "off" bit.

In an embodiment, control bottom 160 contains the following logic for just the lower four index bit, INDEX[0] - INDEX[3]: the ALU 512; the 8x 2:1 MASK multiplexor 514; the 8x 2:1 INDEX multiplexor 516; and the SCALE decoder 521. The select input for the 8x 2:1 MASK multiplexor 514 is bussed down from the control top 120.

In an embodiment, control top 120 contains the same logic as control bottom except for the upper four index bits, INDEX[4] - INDEX[7] plus the "off" bit. Control top 120 also duplicates the carry chain for just the lower four index bit, INDEX[0] - INDEX[3]. Control top 120 also contains the LSB and MSB decoders 519 and 518 and the Kill_Bit, Select_Merge, and Force_Off logic.

The major task in the circuit implementation of the control circuit 500 was to reduce the logic to a form that would minimize global wire count and to minimize the transistor count that had to fit under the global wires in M3 and M4. Wire count reduction required that differential global signals be avoided (such as multiplexor select lines) and that global wires be shared if possible.

ALU Reduction and Programming

Conventionally, as shown in Figure 5C and Figure 5D, the traditional ALU "bit" 532 or 542 includes a 3-input XOR 533 or 543 to do the ADD and a 3-input MAJORITY gate 534 or 544 to generate the CARRY.

However, one input to the ALU 512 is hardwired to the physical address of the control slice 510. In an exemplary embodiment, the hardwired input to the ALU bit is eliminated through logic reduction, as shown in typical ALU bits 536 and 546 in Figure 5C and Figure 5D, respectively. The "tradeoff" here is that the programming of the physical address then has to be done by programming which reduced circuits are placed in the control slice 510 rather than by having a contact/metal wiring option to tie the physical address inputs to the appropriate rail. There are two cases to consider:

1. The physical address bit input to the ALU "bit" is "0", traditional ALU bit 532 and an exemplary embodiment of an ALU bit 536, as depicted in Figure 5C; and
2. The physical address bit input to the ALU "bit" is "1", traditional ALU bit 542 and an exemplary embodiment of an ALU bit 546, as depicted in Figure 5D.

There is also the situation where the sum and carry bits have opposite polarity. Since the polarity of the individual IAdd inputs can be selected, gate conversions are done that

provide consistency in polarity between the outputs of one ALU "bit" and the inputs of the following ALU "bit". The preferred embodiment has "true" inputs and "complement" outputs, or "complement" inputs and "true" outputs. In applying this to the two cases above in Figure 5C and Figure 5D, we get:

1. The physical address bit input to the ALU "bit" is "0", represented by ALU modules 552 and 556, as depicted in Figure 5E; and

2. The physical address bit input to the ALU "bit" is "1", represented by ALU modules 552 and 556, as depicted in Figure 5F.

As a result, two types of ALU modules, (i) ALU module 552 including an XOR 557 and a NOR 558 logically coupled as shown in Figure 5E and Figure 5F and (ii) ALU module 552 including an XNOR 553 and a NAND 554 logically coupled as shown in Figure 5E and Figure 5F, cover the four permutations of physical address bit value and input/output inversions required for a typical ALU 512.

Table 1 - ALU "Bit" Programming

ALU "Bit"	Physical Add Bit = "0"	Physical Add Bit = "1"
"Even"	XOR-NOR (ALU Module 556)	XNOR-NAND (ALU Module 552)
"Odd"	XNOR-NAND (ALU Module 552)	XOR-NOR (ALU Module 556)

Given that these two types of ALU modules have the same layout footprint, the ALU programming is done by simple substitution of the ALU modules.

In an exemplary embodiment, for each 256:1 multiplexor circuit 220, there is a corresponding control slice 502-506. Each 256:1 multiplexor circuit 220 has a physical address bit. Each control slice 510 includes an ALU 512. In an embodiment, the ALU 512 is an 8-bit ALU. As an example of the ALU programming, if the physical address of 256:1 multiplexor circuit is 00000000_2, then for bit 0 of ALU 512, the following is true:

(1) ALU bit 0 is an Even bit (i.e. ALU "Bit" is "Even".);

(2) the value of the bit 0 of the physical address is 0 (i.e. Physical Address (Add) Bit = "0".), and, therefore can be considered to be tied to V_{ss} or ground;

(3) thus, the XOR-NOR module, ALU module 556, for the case when the ALU bit is EVEN and the Physical Address (Add) Bit is 0 is used.

As another example of the ALU programming, if the physical address of 256:1 multiplexor circuit is 00000000_2, then for bit 1 of ALU 512, the following is true:

(1) ALU bit 1 is an Odd bit (i.e. ALU "Bit" is "Odd".);

(2) the value of the bit 1 of the physical address is 0 (i.e. Physical Address (Add) Bit = "0".), and, therefore can be considered to be tied to V_{ss} or ground;

(3) thus, the XNOR-NAND module, ALU module 552, for the case when the ALU bit is Odd and the Physical Address (Add) Bit = "0" is used.

MUX Reduction and Programming

In an exemplary embodiment, since each control slice 510 contains 16 multiplexors (8 in control top 120 and 8 in the control bottom 160, one of the primary objectives in the cross-bar circuit design is to minimize the layout area of cross-bar circuit 100 by using the minimum number of transistors and wires per multiplexor. The number of wires is of particular concern since the height of the control slice 510 is primarily determined by the number of horizontal input wires that feed the control slice 510.

In control slice 510, a CMOS transfer-gate multiplexor 560, as shown in Figure 5G, was not used for the multiplexors in control slice 510 for following reason: since the multiplexors have a mixture of local and global inputs, the use of a transfer-gate multiplexor is problematic unless both the data inputs and the select inputs are buffered locally to prevent unwanted pass gate activation due to either mismatches between local and distance supply levels or to noise on the input lines. In a worst case scenario, a CMOS transfer-gate multiplexor 560 would add 4 inverters, 562-565, to each multiplexor with an additional 4 local wires. This impact could theoretically be reduced by buffering the global signals periodically, such as every eight control slices, but this would then require the addition of more horizontal tracks for the buffered signals. The impact could also be reduced by only buffering the global signals, but then unique designs for the 8x 2:1 MASK multiplexor 514 and SCALE multiplexor 521 would be required. For this reason, a preferred embodiment uses a "logic" implementation for the multiplexors in control slice 510.

22-AOI

In an exemplary embodiment, a 22-AOI 580 is used, as shown in Figure 5H. This approach solves the problem of needing to buffer global signals and has a minimal number of internal wires. It does have the drawback of requiring an additional inverter 582 to eliminate the need for a differential select line.

NOR-21-AOI

In an exemplary embodiment, given the necessity for this additional gate, some logic transformations are made to improve the utility of the multiplexor design. If the data input associated with the inverted select signal is inverted, then the select inverter 582 and the

AND gate 584 can be combined into a NOR gate and the 22-AOI reduced to a 21-AOI, as depicted in Figure 5I.

As shown in Figure 5I, the resulting circuit, 591, is attractive because it has the same number of transistors and internal wires as the 22-AOI implementation 580 and it offers one "true" data input and one "inverted" data input. The is particularly useful given that the ALU design discussed above outputs alternating "true" and "complement" index bits which then are fed to the 8x 2:1 MASK multiplexor 514. This multiplexor design allows the use of either the "true" or "complement" multiplexor input to convert all the index bits to the same polarity. Furthermore, if the two data inputs to this multiplexor circuit are tied together and the NOR output is pulled out, this circuit can now also be used to implement the XOR-NOR module, ALU module 556, required for the ALU implementation as discussed above. Thus the same primitive can be used for both the ALU and the multiplexors, which greatly simplifies the design.

NAND-21-OAI

In an exemplary embodiment, one additional transformation is still required for this circuit approach. The NOR-21-AOI module, MUX module 591, implementation of the multiplexor selects the non-inverting path when the select input is low and the inverting path when the select input is high. It is also desirable to have a multiplexor implementation that reverses this select sequence. This allows two multiplexors sharing the same select input to simultaneously invert the input data in one multiplexor and to retain the data polarity in the other multiplexor. This may be accomplished by transforming the design 591 to a NAND-21-OAI module, MUX module 596, as shown in Figure 5I.

The resulting circuit 596 reverses the selection of the inverting and non-inverting inputs. Furthermore, if the two data inputs to this multiplexor circuit are tied together and the NAND output is pulled out, this circuit can now also be used to implement the XNOR-NAND module, ALU module 552, required for the ALU implementation as discussed above. Now two primitives can satisfy the requirements for both the ALU and the IMask/IScale MUXes.

Using these two MUX modules, the design of the ALU/MUX portion of the control slice is complete as shown in Figure 6. The ALU 602 and 604 for even and odd index bits of the control slice is chosen per the rules given above with respect to the ALU Programming in the **Table - ALU "Bit" Programming.** The "even" ALU bit outputs are "true" while the "odd" ALU bit outputs are "inverted". This prompts the use of the NOR-21-AOI module, MUX module 591, for the 8x 2:1 MASK multiplexor 514 in the even index bits and the

NAND-21-OAI module, MUX module 596, for the 8x 2:1 MASK multiplexor 514 in the odd index bits. This restores uniform polarity of the index bits at the output of the 8x 2:1 MASK multiplexor 514. Finally, the NAND-21-OAI module, MUX module 596, is used for the SCALE decoder 521 in all the index bits.

SCALE Decoder Wire Sharing

The SCALE decoder 521 takes a 7-bit input for each index bit position, decodes it to determine a scale value, and if selected, passes it to the index bit. This selection is determined by the ISize bus which always uses a thermometer value, selecting 8x 2:1 MASK multiplexor 514 output in the lower bit positions and the SCALE decoder 521 output in the upper bit positions. A convenient consequence of this is that in these lower order bit positions, the value of the SCALE decoder 521 outputs and thus their inputs are irrelevant. Similarly, in these upper order bit positions, the outputs of the ALU 512 and the 8x 2:1 MASK multiplexor 514 and thus their inputs - IAdd0, IAdd1, IConst0, and IConst1 - are irrelevant. This means that these 4 wires can be shared with the 4 of the 7 inputs to SCALE decoder 521. This results in a height savings of 4 metal pitches per "bit" of the ALU 512/ the 8x 2:1 MASK multiplexor 514 /SCALE decoder 521 for a total saving of 32 metal pitches.

MSB/LSB Decoders

As depicted in Figure 7, each of the MSB and LSB decoders 518 and 519, a MSB/LSB decoder include a 5-stage chain of NAND/NOR gates where the connections to the MaskSel bus and the gate types are determined by the physical address of the control slice. The programming for the MSB decoder 518 is determined by the following rules:

Table 2A - MSB Mask Decode Programming

		Add Bit Value=0		Add Bit Value=1		Add Bit Value=2	Add Bit Value=3
Stage	Physical Add Bit	Mask Bit Tap	Gate Type	Mask Bit Tap	Gate Type	Mask Bit Tap	Mask Bit Tap
1a	1:0	0	na	1	na	2	3
1b	2	4	NOR	5	NAND		
2	3	6	NAND	7	NOR		
3	4	8	NOR	9	NAND		
4	5	10	NAND	11	NOR		

5	6	12	NOR	13	NAND		

The rules for the LSB decoder 519 are identical except that the NAND/NOR selections are reversed.

The nibble is the smallest unit that can be selected for masking. Individual bits in the nibble are selected for masking by bits 0, 1, 2, and 3 of the MaskSel bus. These bits are "inverted", so setting any combination of these bits "low" activates masking in the corresponding bits of the nibble of "slices".

The remaining bits of the MaskSelect bus are used to determine the "grouping" in which this nibble is repeated/expanded (byte, doublet, etc) and to enforce masking on one side of this "group" and to prevent masking on the other side of this "group". Masking is forced on the MSB (Most Significant Bit) side of the "group" in the MSB decoder 518 and on the LSB (Least Significant Bit) side of the "group" in the LSB decoder 519. The rules for programming this grouping are as follows:

Table 2B - Mask Decode Input Rules

Controlling Bits	Effected "Group"	MSB Decoder	LSB Decoder
/MaskSel[3:0]	Corresponding Bits in Nibble	"Hi" = Don't Mask "Lo" = Mask	"Hi" = Don't Mask "Lo" = Mask
/MaskSel[4]	Even Nibbles	"Hi" = Prevent Masking "Lo" = Allow Masking	"Hi" = Allow Masking "Lo" = Force Masking
/MaskSel[5]	Odd Nibbles	"Hi" = Allow Masking "Lo" = Force Masking	"Hi" = Prevent Masking "Lo" = Allow Masking
MaskSel[6]	Even Bytes	"Lo" = Prevent Masking "Hi" = Allow Masking	"Lo" = Allow Masking "Hi" = Force Masking
MaskSel[7]	Odd Bytes	"Lo" = Allow Masking "Hi" = Force Masking	"Lo" = Prevent Masking "Hi" = Allow Masking
/MaskSel[8]	Even Doublets	"Hi" = Prevent Masking "Lo" = Allow Masking	"Hi" = Allow Masking "Lo" = Force Masking

/MaskSel[9]	Odd Doublets	"Hi" = Allow Masking "Lo" = Force Masking	"Hi" = Prevent Masking "Lo" = Allow Masking
MaskSel[10]	Even Quadlets	"Lo" = Prevent Masking "Hi" = Allow Masking	"Lo" = Allow Masking "Hi" = Force Masking
MaskSel[11]	Odd Quadlets	"Lo" = Allow Masking "Hi" = Force Masking	"Lo" = Prevent Masking "Hi" = Allow Masking
/MaskSel[12]	Even Octlet	"Hi" = Prevent Masking "Lo" = Allow Masking	"Hi" = Allow Masking "Lo" = Force Masking
/MaskSel[13]	Odd Octlet	"Hi" = Allow Masking "Lo" = Force Masking	"Hi" = Prevent Masking "Lo" = Allow Masking

These selected "groups" overlay each other to determine the total masking pattern. The "groups" selected by the higher order MaskSel bits take priority over lower "groups".

As an example, suppose that we wished to have the mask pattern "5" (i.e. - 0101) occur in the two nibbles of byte number 5 of the cross-bar DESTINATION output word, with bytes 15-6 masked and bytes 4-0 not masked. This would be done as follows:

```
Byte #                  15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

/MMaskSel[3:0]=5        55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
/MMaskSel[5:4]=10       ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??
 MMaskSel[7:6]=00       ?? -- ?? -- ?? -- ?? -- ?? -- ?? -- ?? -- ?? --
/MMaskSel[9:8]=00       XX XX ?? ?? XX XX ?? ?? XX XX ?? ?? XX XX ?? ??
 MMaskSel[11:10]=00     ?? ?? ?? ?? -- -- -- -- ?? ?? ?? ?? -- -- -- --
/MMaskSel[13:12]=00     XX XX XX XX XX XX XX XX ?? ?? ?? ?? ?? ?? ?? ??

Resulting Mask Pattern  XX XX XX XX XX XX XX XX XX XX 55 -- -- -- -- --

                              "X" = masked    "-" = not masked
```

Note that bits 6,7,10, and 11 of the MaskSel bus are "true" while the remaining bits are "inverted". The schematics use two separate busses, one "true" and one "inverted", to carry these bits. This is done only as a convenience in the schematics since the schematic editor

does not allow mixing of signal names in a bus. The layout combines the used signals in one single bus.

The outputs of the MSB/LSB decoders 518 and 519 go to three sets of multiplexors IMsk 522, IOff 523, & ISet 524 which in turn feed the 8x 2:1 MASK multiplexor 514, the Force_Off logic 525, and the Kill_Bit7 logic 526. These multiplexors 522-524 are simple AND-OR multiplexors built with AOIs and inverters and/or NAND gates.

Kill_Bit Logic

The Kill_Bit logic 526 combines the Kill_Bit signals and the output of the ISet multiplexor 524 to override the state of index bits 6 and 7. The logic 526is implemented with AOI, inverter, and NAND gates.

Having fully described a preferred embodiment of the invention and various alternatives, those skilled in the art will recognize, given the teachings herein, that numerous alternatives and equivalents exist which do not depart from the invention. It is therefore intended that the invention not be limited by the foregoing description, but only by the appended claims.

CLAIMS

We claim:

1. A cross-bar circuit that, in response to partially-decoded instruction information and in response to datapath information, (1) allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and (2) provides the ability to set-to-zero any bit in said 2^m-bit output destination word, said cross-bar circuit comprising:

a **switch circuit** comprising 2^m 2^n:1 multiplexor circuits, wherein each of said 2^n:1 multiplexor circuits (a) has a unique n-bit (e.g. 8-bit) index input, one disable input, and a 2^n-bit wide source input, (b) receives (i) an n-bit index at said n-bit index input, (ii) a disable bit at said disable input, and (iii) said 2^n-bit input source word at said 2^n-bit wide source input, and (c) decodes said n-bit index either (i) to select and output as an output destination bit one bit from said 2^n-bit input source word if said disable bit has a logic low value or (ii) outputs a logic low as said output destination bit if said disable bit has a logic high value;

a **cache memory** that (a) has 2^m cache datapath inputs and 2^m cache index inputs, (b) receives (i) said datapath information on said 2^m cache datapath inputs and (ii) 2^m n-bit indexes on said 2^m cache index inputs, (c) provides a first set of said n-bit indexes for said switch circuit, and (d) comprises a small tightly coupled memory array that stores p (e.g. eight) entries of 2^m n-bit indexes for said switch circuit, said cache memory being logically coupled to said switch circuit; **and**

a **control circuit** that (a) has a plurality (e.g. 100) of control inputs, (b) receives said partially-decoded instruction information on said plurality of control inputs, (c) provides a second set of said n-bit indexes for said switch circuit, and (d) provides said disable bits for said switch circuit, said control circuit being logically coupled to said switch circuit and to said cache memory.

2. The cross-bar circuit of claim 1 wherein said cache memory stores switching patterns provided by a datapath logically coupled to said cross-bar circuit.

3. The cross-bar circuit of claim 1 wherein said cache memory stores index patterns generated by instruction decode logic logically coupled to said cross-bar circuit.

4. The cross-bar circuit of claim 1 wherein said **control circuit**, in response to said partially-decoded instruction information, completes the decoding on said partially-decoded instruction information and, thereby, provides indexes and disable bits, wherein said control circuit further comprises:

an arithmetic logic unit (ALU) for each of the 2^m 2^n:1 multiplexor circuits in said switch circuit, wherein said ALU adds an n-bit index addend to the m-bit physical address of each said 2^n:1 multiplexor circuits;

a plurality of multiplexors logically coupled to said ALU and controlled by masks; and

a plurality of decoders logically coupled to said plurality of multiplexors that generate said masks that control said multiplexors.

5. The cross-bar circuit of claim 4 wherein said decoding implements the "standard" cross-bar operations provided in an ISA.

6. A **switch circuit** that (1) allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and (2) provides the ability to set-to-zero any bit in said 2^m-bit output destination word, said switch circuit comprising:

2^m 2^n:1 multiplexor circuits, wherein each of said 2^n:1 multiplexor circuits (a) has a unique n-bit (e.g. 8-bit) index input and one disable input, (b) decodes an n-bit index received at said n-bit index input to select one bit from said 2^n-bit input source word if a disable bit received at said disable input has a logic low value, and (c) outputs a logic low if said disable bit has a logic high value.

7. A **switch circuit** that (1) allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and (2) provides the ability to set-to-zero any bit in said 2^m-bit output destination word, said switch circuit comprising:

2^m 2^n:1 multiplexor circuits, wherein each of said 2^n:1 multiplexor circuits (a) has a unique n-bit (e.g. 8-bit) index input and one disable input, (b) decodes an n-bit index received at said n-bit index input to select one bit from said 2^n-bit input source word if a disable bit received at said disable input has a logic low value, and (c) outputs a logic low if said disable bit has a logic high value;

wherein each of said 2^n:1 multiplexor circuits comprises:

a 2^q:1 pass gate selector (e.g. 4:1 pass gate selector), wherein said 2^q:1 pass gate selector has 2^q precharge/discharge wire-OR bitline inputs;

a sense amplifier logically coupled to said 2^q:1 pass gate selector, wherein said sense amplifier (a) receives the output of said 2^q:1 pass gate selector and (b) receives a dummy bitline input to allow differential sensing of small swing signals on said wire-OR bitline inputs; and

2^{n-q} unit switch cells (e.g. 64 unit switch cells) per precharge/discharge wire-OR bitline input, each of said unit switch cells (a) being logically coupled to a wire-OR bitline input of said 2^q:1 pass gate selector, (b) being logically coupled to one of 2^r (e.g. one of 8) active-LOW "SELA" select wires, and (c) being logically coupled to one of 2^{n-q-r} (e.g. one of 8) active-HIGH "SELC" select wires.

8. The switch circuit of claim 7 wherein each of said unit switch cells comprises:

two series connected transistors of the same polarity, wherein the transistor closest to said wire-OR bitline inputs is logically coupled to said one of 2^r active-LOW "SELA" select wires and wherein the other transistor is logically coupled to a "SRC" data input wire;

wherein in each of said unit switch cells, if said one of 2^r active-LOW "SELA" select wires has a logic LOW signal and said one of 2^{n-q-r} active-HIGH "SELC" select wires has a logic HIGH signal, then a wire-OR bitline logically coupled to said wire-OR bitline input discharges into said one of 2^r active-LOW "SELA" select wires if said "SRC" data input wire has a logic HIGH signal.

9. The cross-bar circuit of claim 1 wherein said **control circuit**, in response to said partially-decoded instruction information, completes the decoding on said partially-decoded instruction information and, thereby, provides indexes and disable bits, wherein said control circuit further comprises:

an arithmetic logic unit (ALU) for each of the 2^m 2^n:1 multiplexor circuits in said switch circuit, wherein said ALU adds an n-bit index addend to the m-bit physical address of each said 2^n:1 multiplexor circuits, wherein said ALU comprises a plurality of ALU modules;

a plurality of multiplexors logically coupled to said ALU and controlled by masks, wherein each of said multiplexors comprises a plurality of multiplexor (MUX) modules; and

a plurality of decoders logically coupled to said plurality of multiplexors that generate said masks that control said multiplexors, wherein each of said decoders comprises a s-stage (e.g. 5 stage) chain of NAND/NOR gates.

10. The control circuit of claim 9

wherein one type of said ALU modules comprises:

a two input XOR gate; and

a two input NOR gate, wherein the two inputs of said NOR gate are logically coupled to the two inputs of said XOR gate; and

wherein another type of said ALU modules comprises:

a two input XNOR gate; and

a two input NAND gate, wherein the two inputs of said XNOR gate are logically coupled to the two inputs of said NAND gate.

11. The control circuit of claim 9

wherein one type of said MUX modules comprises:

a first two input NOR gate;

a two input AND gate having its output logically coupled to the first input of said first NOR gate; and

a second two input NOR gate (a) having its output logically coupled to the second input of said first NOR gate and (b) having an input logically coupled to an input of said AND gate; and

wherein another type of said MUX modules comprises:

a first two input NAND gate;

a two input OR gate having its output logically coupled to the first input of said first NAND gate; and

a second two input NAND gate (a) having its output logically coupled to the second input of said first NAND gate and (b) having an input logically coupled to an input of said OR gate.

12. A **cross-bar circuit** that, in response to partially-decoded instruction information and in response to datapath information, (1) allows any bit from a 2^n-bit (e.g. 256-bit) input source word to be switched into any bit position of a 2^m-bit (e.g. 128-bit) output destination word and (2) provides the ability to set-to-zero any bit in said 2^m-bit output destination word, said cross-bar circuit comprising:

a switch circuit;

a cache, wherein said cache (a) is logically coupled to said switch circuit and (b) is physically split into two halves, one half being above said switch circuit and the other half being below said switch circuit; and

a control circuit, wherein said control circuit (a) is logically coupled to said switch circuit and to said cache and (b) is physically split into two halves, one half being above said switch circuit and the other half being below said switch circuit.

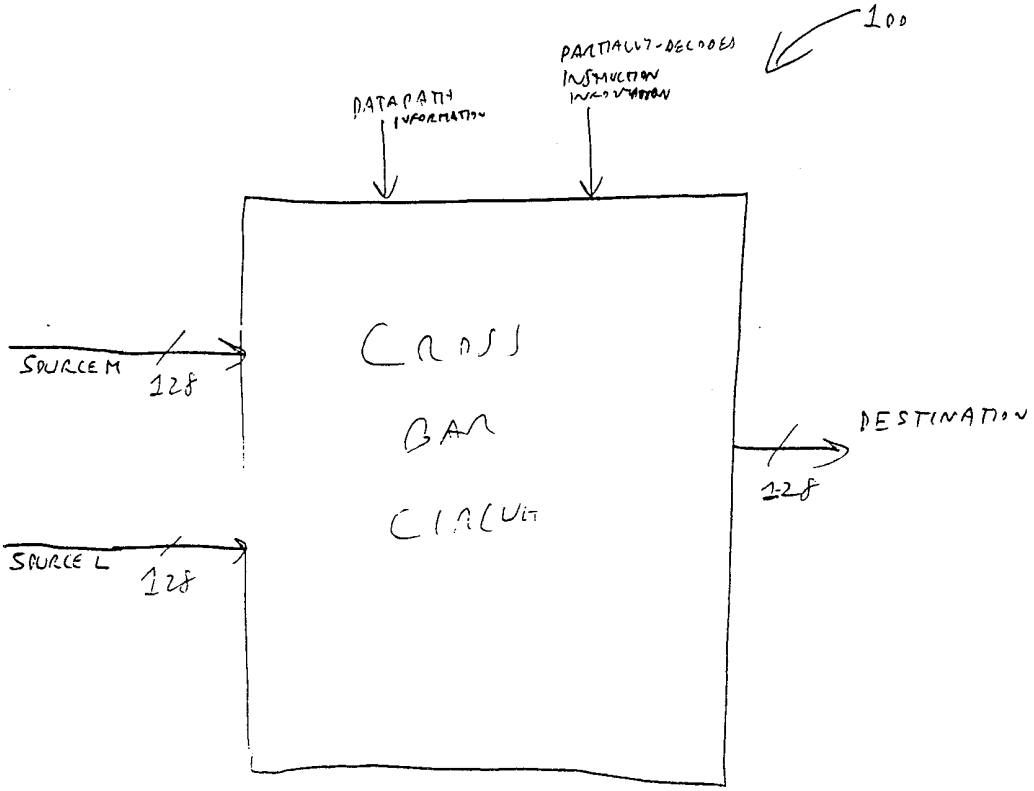

100
PARTIALLY-DECODED INSTRUCTION INFORMATION
DATAPATH INFORMATION
SOURCE M
128
SOURCE L
128
CROSS BAR CIRCUIT
DESTINATION
128

FIG 1A

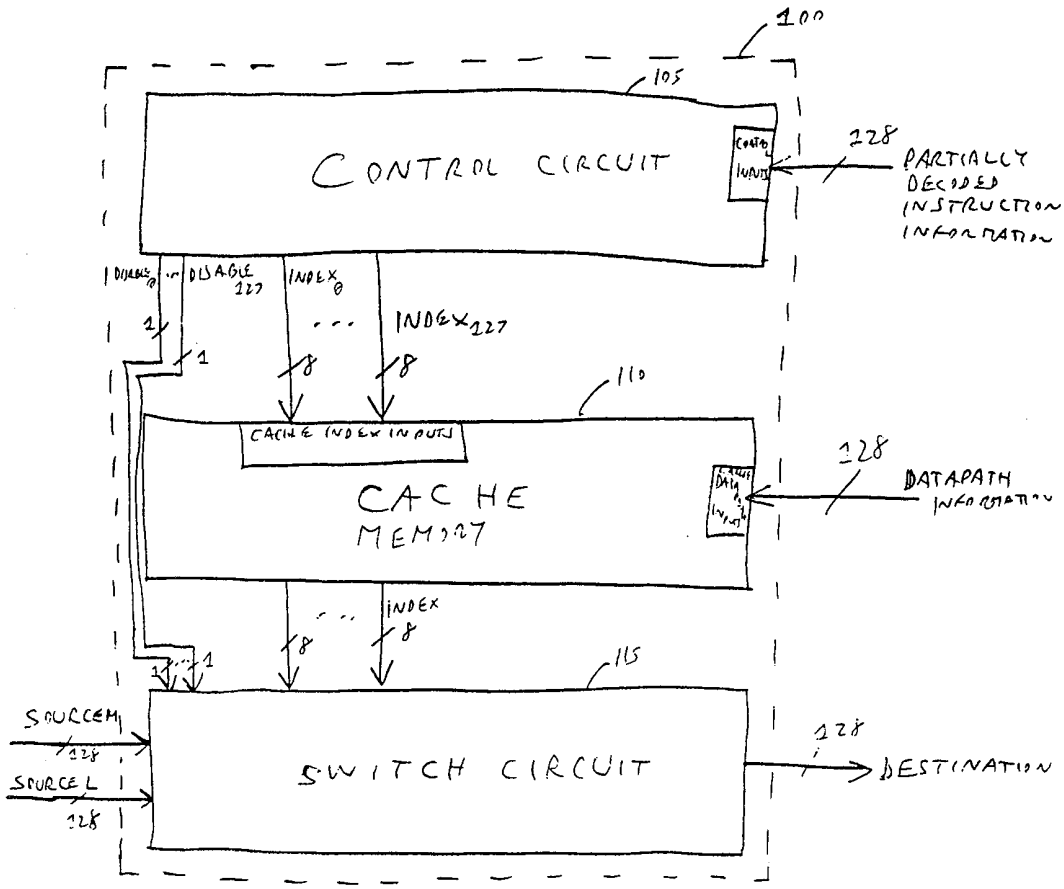
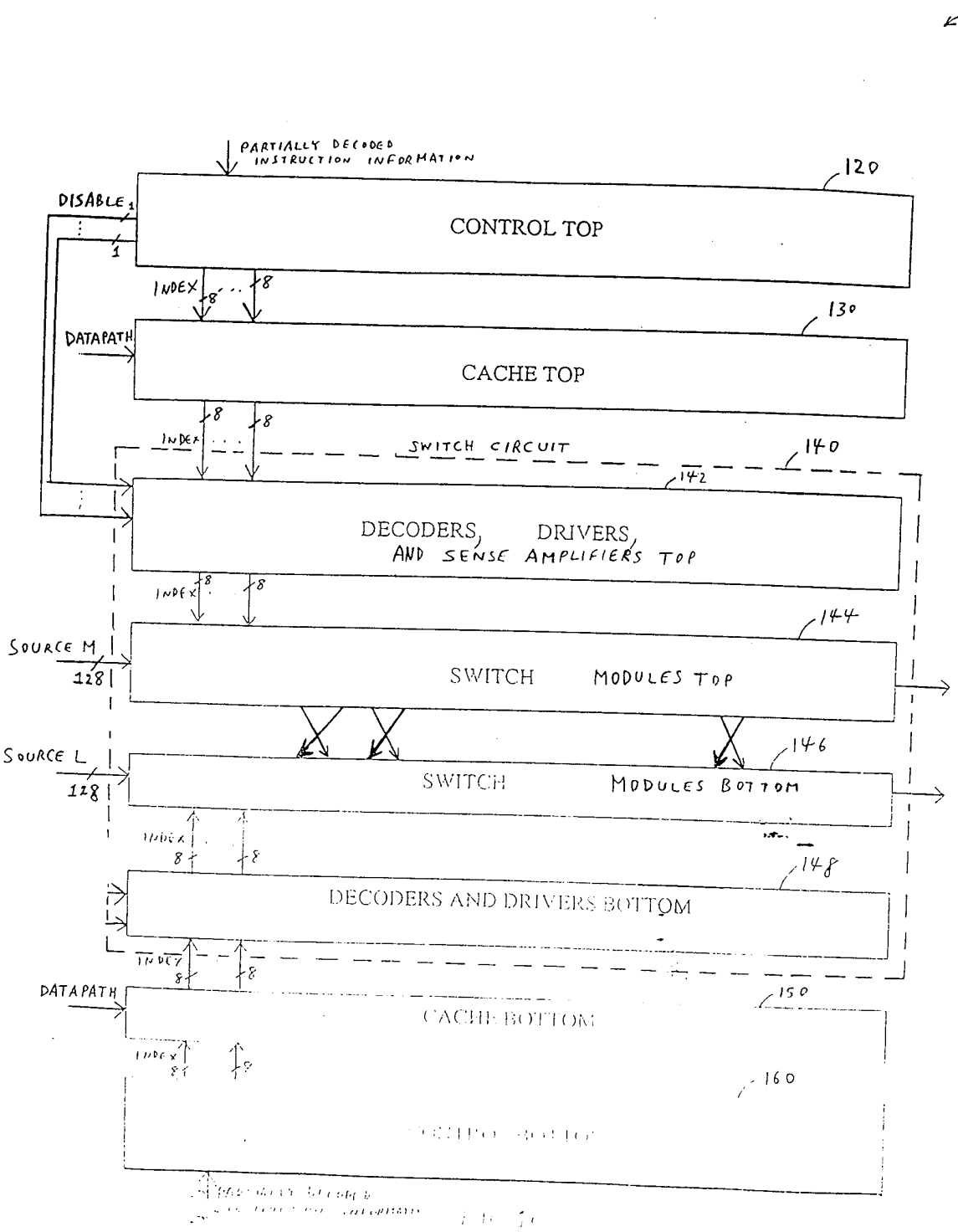

100
105
CONTROL CIRCUIT
128
PARTIALLY DECODED INSTRUCTION INFORMATION
USAGE 127
INDEX 0
INDEX 127
110
CACHE INDEX INPUTS
CACHE MEMORY
128
DATAPATH INFORMATION
INDEX
115
SOURCE M
128
SOURCE L
128
SWITCH CIRCUIT
128
DESTINATION

FIG. 1B


101
PARTIALLY DECODED INSTRUCTION INFORMATION
120
CONTROL TOP
DISABLE 1
INDEX 8 8
130
CACHE TOP
DATAPATH
INDEX 8 8
SWITCH CIRCUIT
140
142
DECODERS, DRIVERS, AND SENSE AMPLIFIERS TOP
INDEX 8 8
144
SOURCE M 128
SWITCH MODULES TOP
146
SOURCE L 128
SWITCH MODULES BOTTOM
INDEX 8 8
148
DECODERS AND DRIVERS BOTTOM
INDEX 8 8
DATAPATH
150
CACHE BOTTOM
INDEX 8 8
160

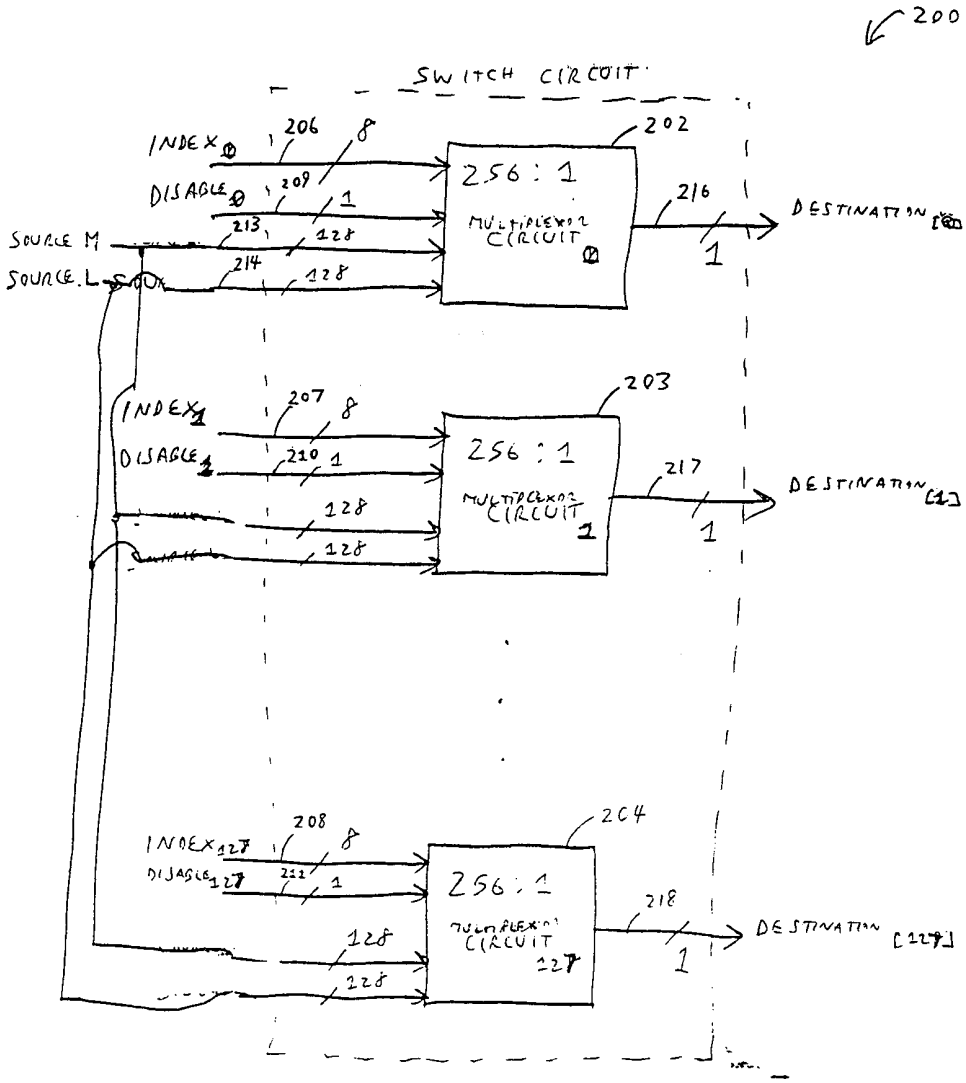

200
SWITCH CIRCUIT
INDEX 0
206
8
DISABLE 0
209
1
SOURCE M
213
128
SOURCE L
214
128
256 : 1
MULTIPLEXOR CIRCUIT 0
202
216
1
DESTINATION [0]
INDEX 1
207
8
DISABLE 1
210
1
128
128
256 : 1
MULTIPLEXOR CIRCUIT 1
203
217
1
DESTINATION [1]
INDEX 127
208
8
DISABLE 127
211
1
128
128
256 : 1
MULTIPLEXOR CIRCUIT 127
204
218
1
DESTINATION [127]

FIG. 2

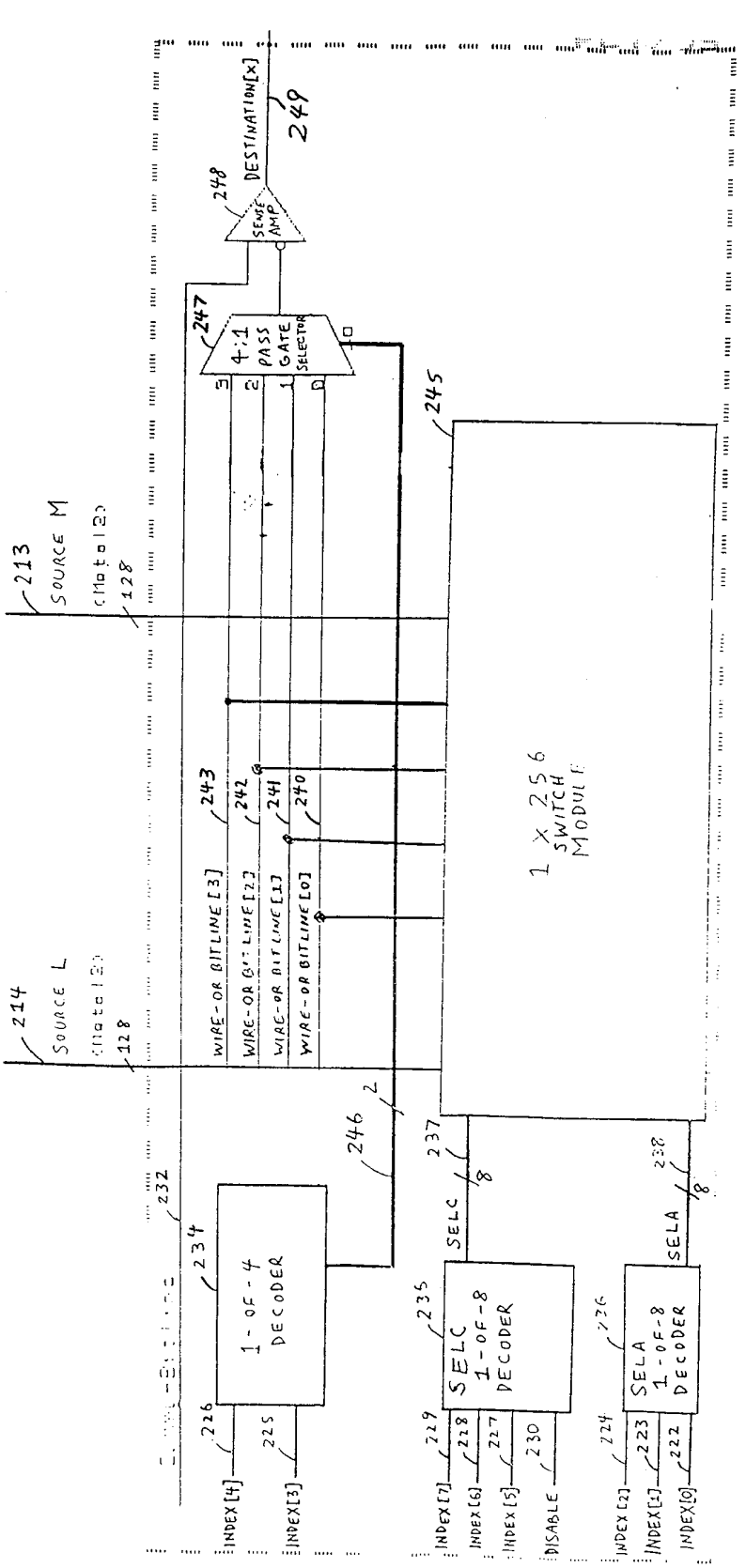

220
213
SOURCE M
128
214
SOURCE L
128
WIRE-OR BITLINE [3]
WIRE-OR BITLINE [2]
WIRE-OR BITLINE [1]
WIRE-OR BITLINE [0]
243
242
241
240
4:1 PASS GATE SELECTOR
247
SENSE AMP
248
DESTINATION[x]
249
245
1 X 256 SWITCH MODULE
246
2
232
234
1 - OF - 4 DECODER
226
225
INDEX[4]
INDEX[3]
237
8
SELC
235
SELC 1 - OF - 8 DECODER
229
228
227
230
INDEX[7]
INDEX[6]
INDEX[5]
DISABLE
238
8
SELA
236
SELA 1 - OF - 8 DECODER
224
223
222
INDEX[2]
INDEX[1]
INDEX[0]

FIG. 2B

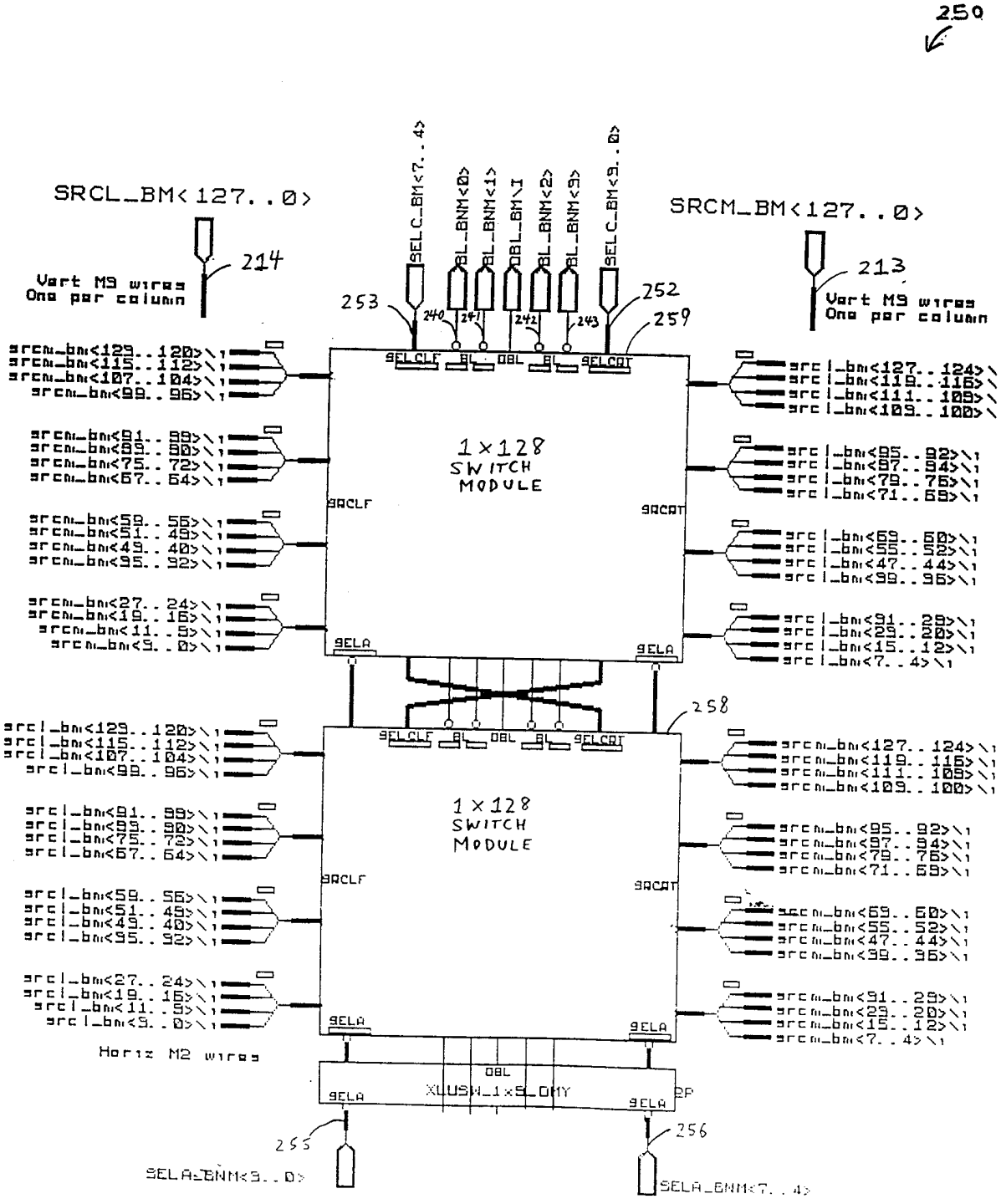
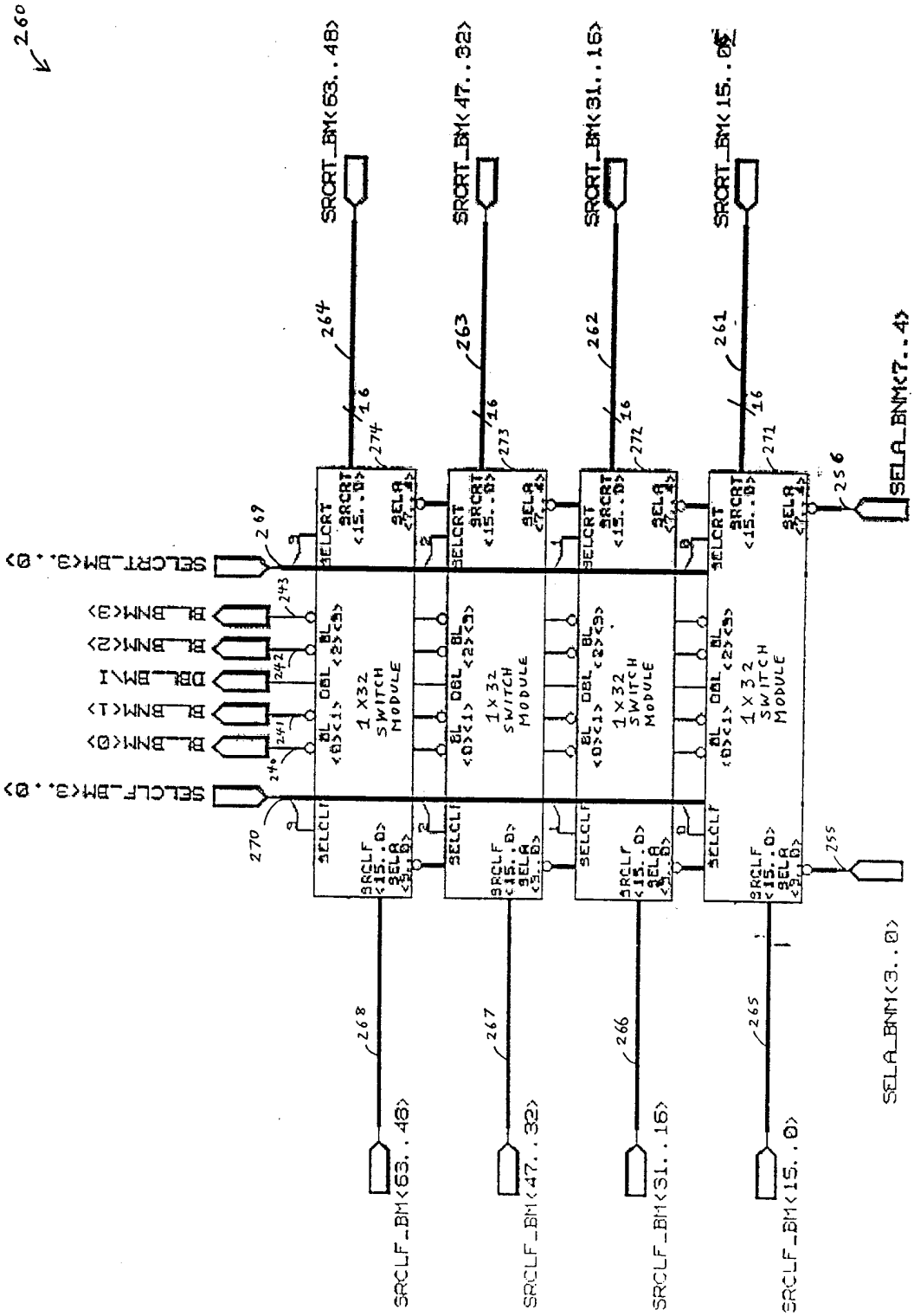

250
SRCL_BM<127..0>
214
Vert M3 wires
One per column
SRCM_BM<127..0>
213
Vert M3 wires
One per column
SELC_BM<7..4>
BL_BNM<0>
BL_BNM<1>
OBL_BM\1
BL_BNM<2>
BL_BNM<3>
SELC_BM<3..0>
253
252
259
240
241
242
243
1x128
SWITCH
MODULE
SRCLF
SRCRT
SELA
SELCLF
SELCRT
BL
OBL
258
Horiz M2 wires
OBL
XLUSW_1x8_DMY
255
256
SELA_BNM<3..0>
SELA_BNM<7..4>

FIG. 2C


260
SRCRT_BM<63..48>
SRCRT_BM<47..32>
SRCRT_BM<31..16>
SRCRT_BM<15..0>
264
263
262
261
SELA_BNM<7..4>
SELCRT_BM<3..0>
BL_BNM<3>
BL_BNM<2>
DBL_BM\1
BL_BNM<1>
BL_BNM<0>
SELCLF_BM<3..0>
1 X 32 SWITCH MODULE
SELCRT
SRCRT <15..0>
SELA <7..4>
SELCLF
SRCLF <15..0>
SELA <3..0>
SRCLF_BM<63..48>
SRCLF_BM<47..32>
SRCLF_BM<31..16>
SRCLF_BM<15..0>
SELA_BNM<3..0>
FIG. 2D

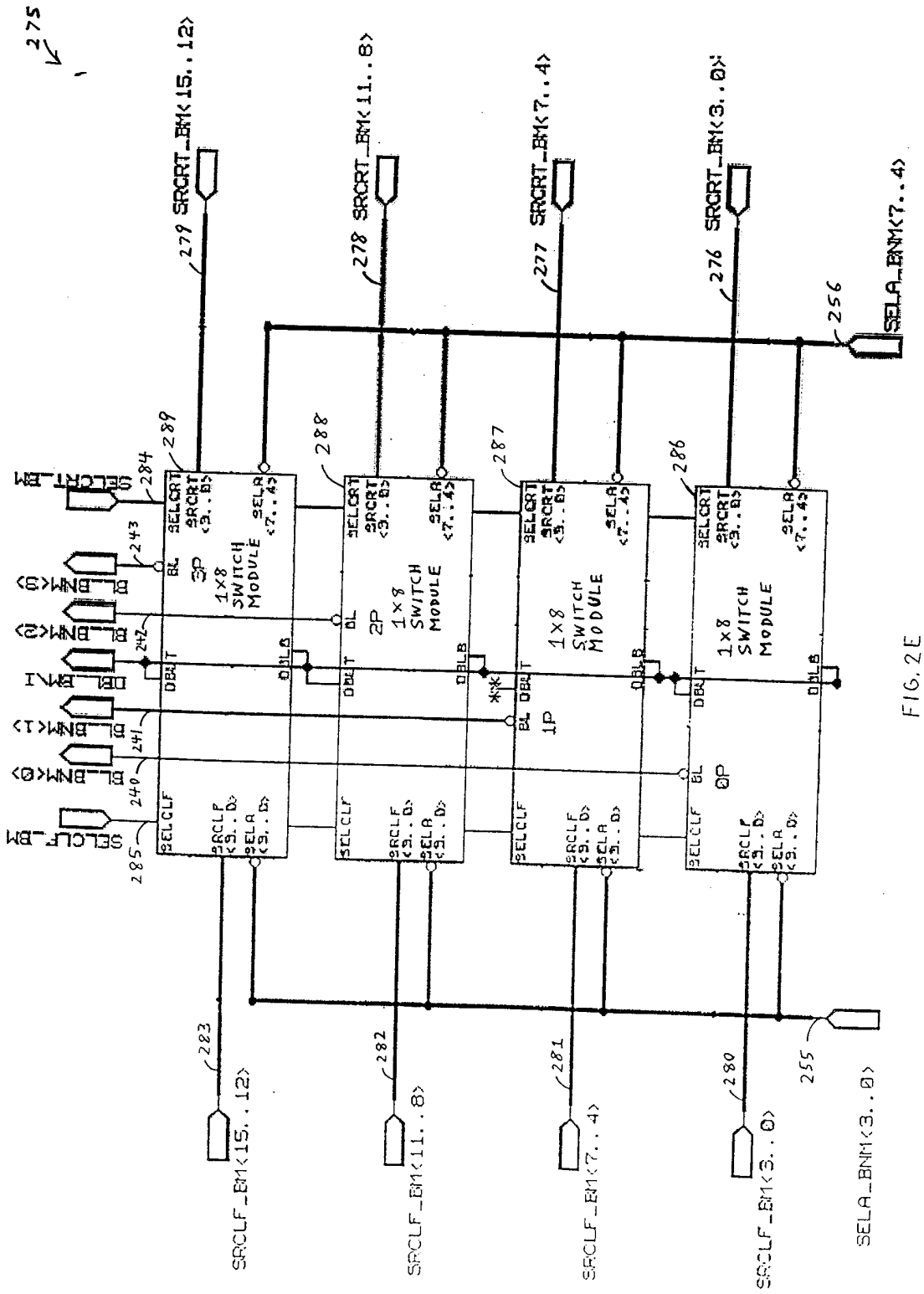

275
279 SRCRT_BM<15..12>
278 SRCRT_BM<11..8>
277 SRCRT_BM<7..4>
276 SRCRT_BM<3..0>
256 SELA_BM<7..4>
SELCRT_BM
284
289
288
287
286
243
BM<3>
242
BM<2>
BM<1>
241
BM<0>
240
SELCLF_BM
285
3P 1x8 SWITCH MODULE
2P 1x8 SWITCH MODULE
1P 1x8 SWITCH MODULE
0P 1x8 SWITCH MODULE
SELCRT SRCRT<3..0> SELA<7..4>
SELCLF SRCLF<3..0> SELA<3..0>
DBLT DBLB BL
283 SRCLF_BM<15..12>
282 SRCLF_BM<11..8>
281 SRCLF_BM<7..4>
280 SRCLF_BM<3..0>
255 SELA_BM<3..0>

FIG. 2E

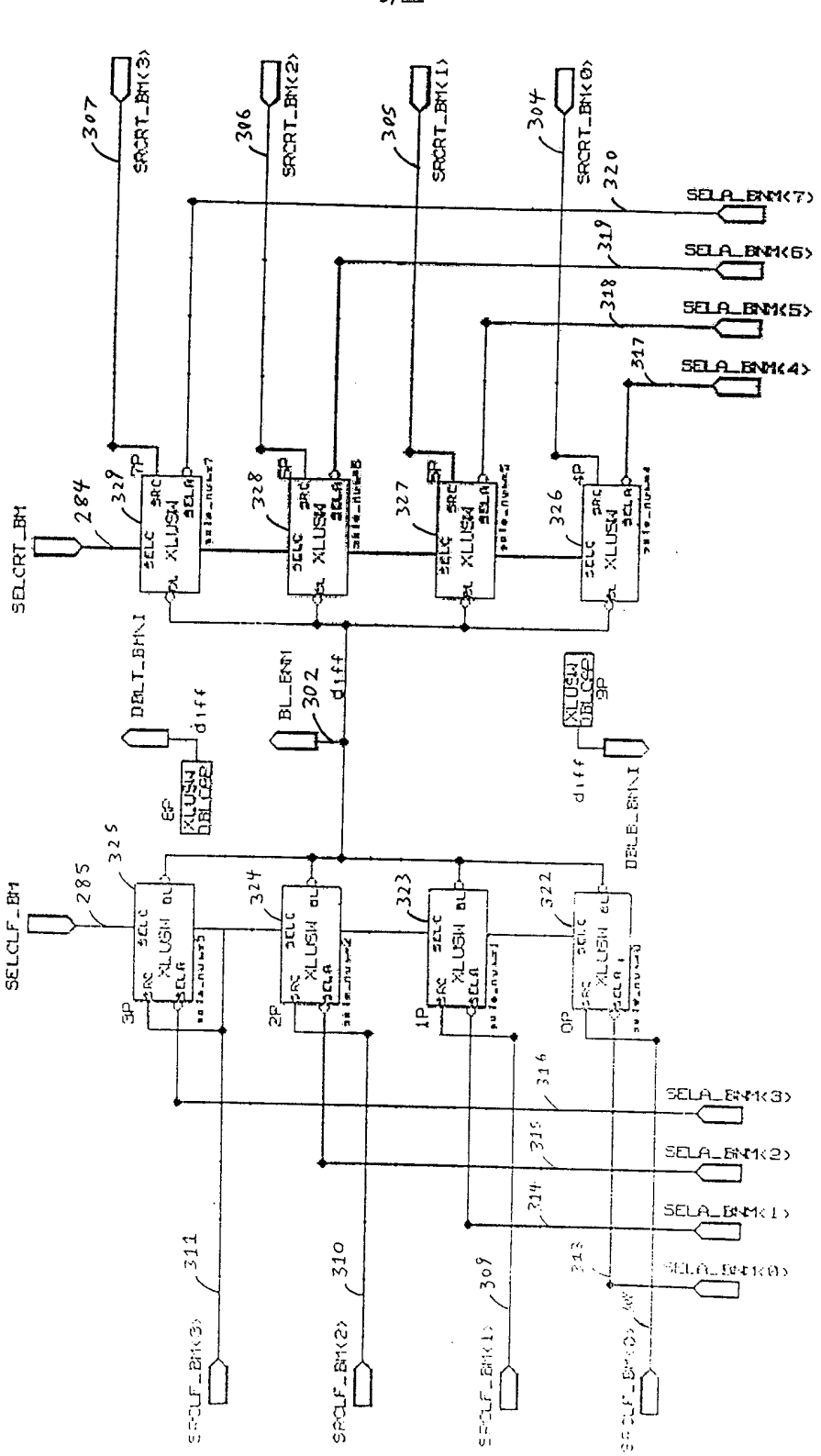

300
SRCRT_BM<3>
SRCRT_BM<2>
SRCRT_BM<1>
SRCRT_BM<0>
307
306
305
304
320
319
318
317
SELA_BNM<7>
SELA_BNM<6>
SELA_BNM<5>
SELA_BNM<4>
SELCRT_BM
284
329
328
327
326
DBLT_BMNI
BL_BMNI
302
diff
DBLB_BMNI
SELCLF_BM
285
325
324
323
322
316
315
314
313
SELA_BNM<3>
SELA_BNM<2>
SELA_BNM<1>
SELA_BNM<0>
311
310
309
SRCLF_BM<3>
SRCLF_BM<2>
SRCLF_BM<1>
SRCLF_BM<0>
XLUSW
SELC
SELA
SRC
DBL_CAP

FIG. 3A

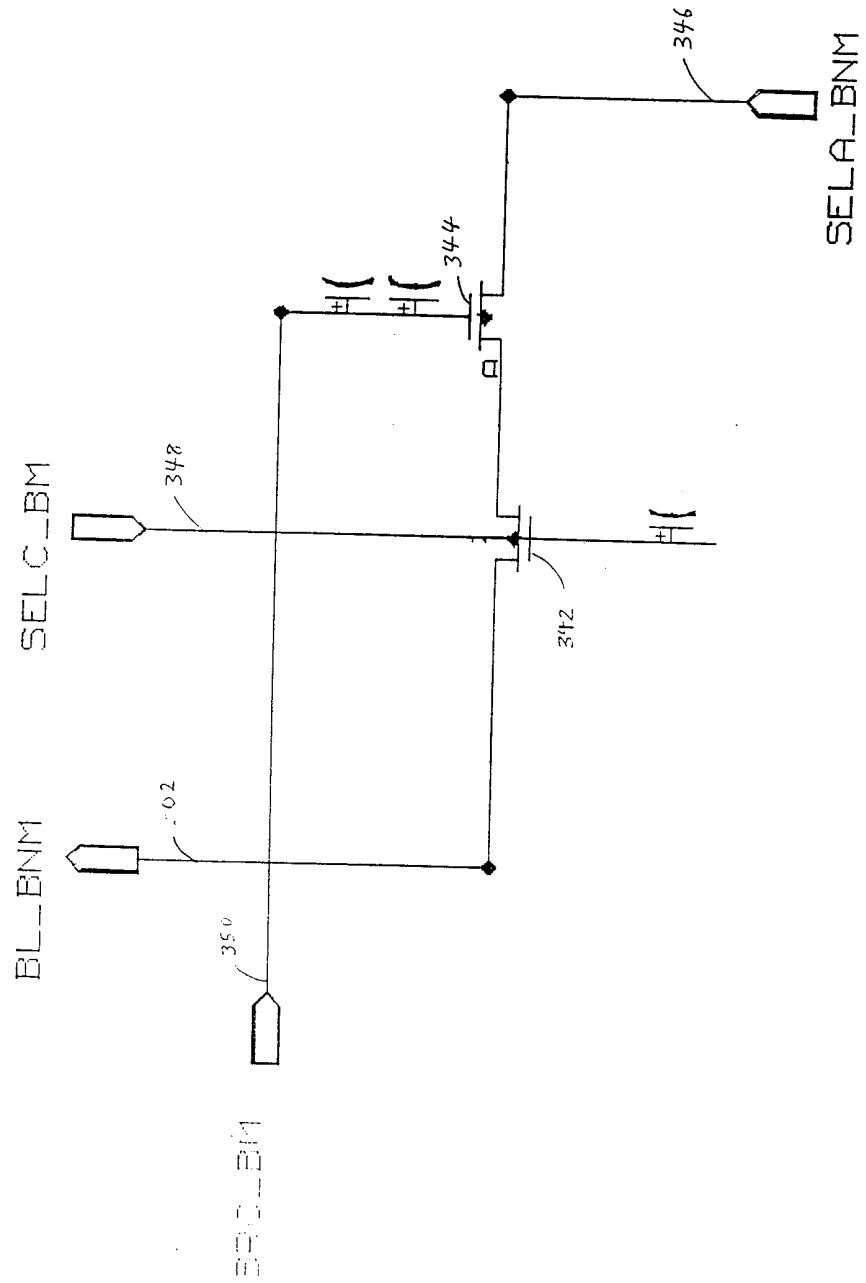


FIG. 3B

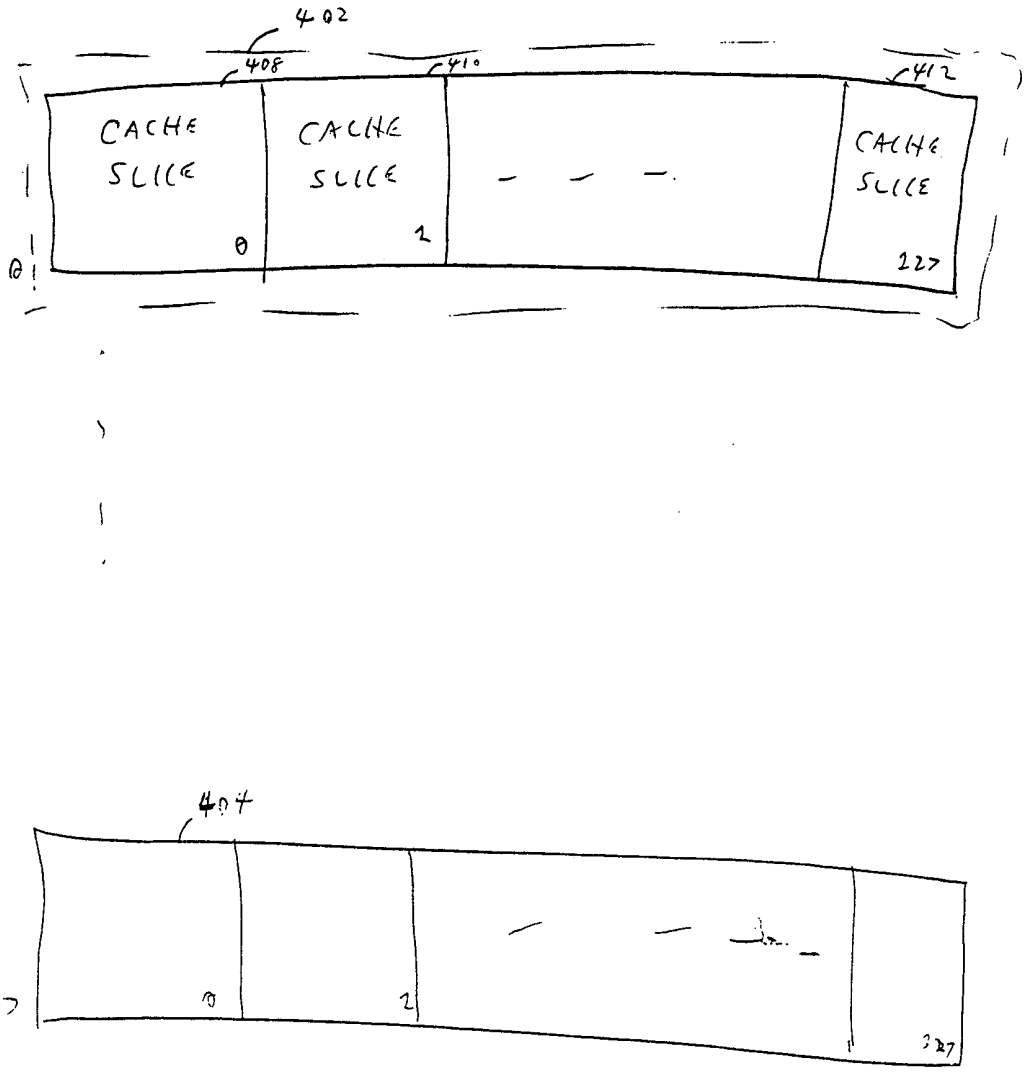

402
408
410
412
CACHE SLICE
0
CACHE SLICE
1
CACHE SLICE
127
404
0
1
127

FIG. 4

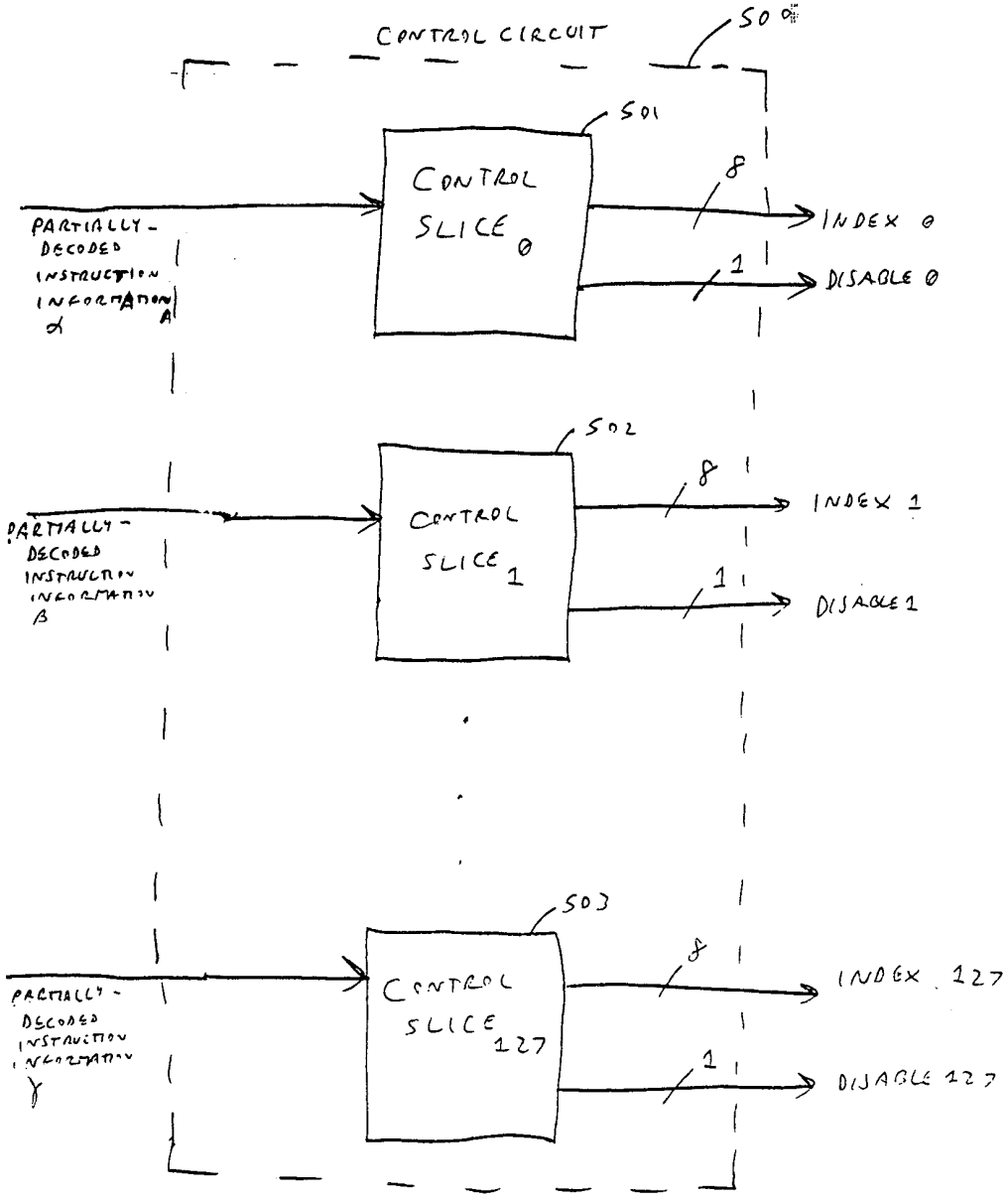

CONTROL CIRCUIT
500
501
CONTROL SLICE 0
PARTIALLY-DECODED INSTRUCTION INFORMATION A
8
INDEX 0
1
DISABLE 0
502
CONTROL SLICE 1
PARTIALLY-DECODED INSTRUCTION INFORMATION B
8
INDEX 1
1
DISABLE 1
503
CONTROL SLICE 127
PARTIALLY-DECODED INSTRUCTION INFORMATION Y
8
INDEX 127
1
DISABLE 127

FIG.5A

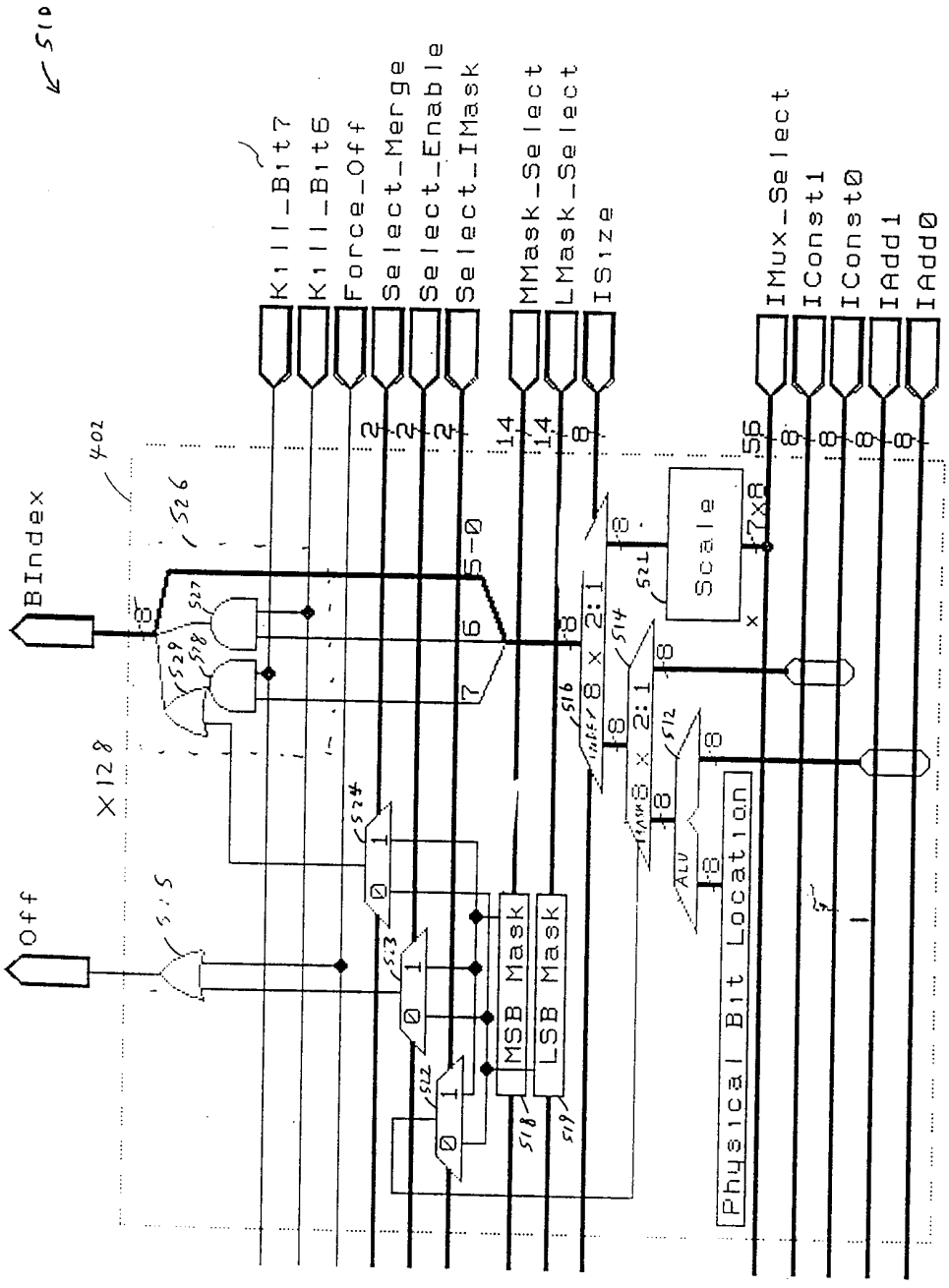

510
Kill_Bit7
Kill_Bit6
Force_Off
Select_Merge
Select_Enable
Select_IMask
MMask_Select
LMask_Select
ISize
IMux_Select
IConst1
IConst0
IAdd1
IAdd0
BIndex
Off
402
526
527
528
529
515
524
513
512
MSB Mask
LSB Mask
518
519
516
8 x 2:1
514
521
Scale
7x8
522
8 x 2:1
512
ALU
Physical Bit Location
X128

FIG. 5B

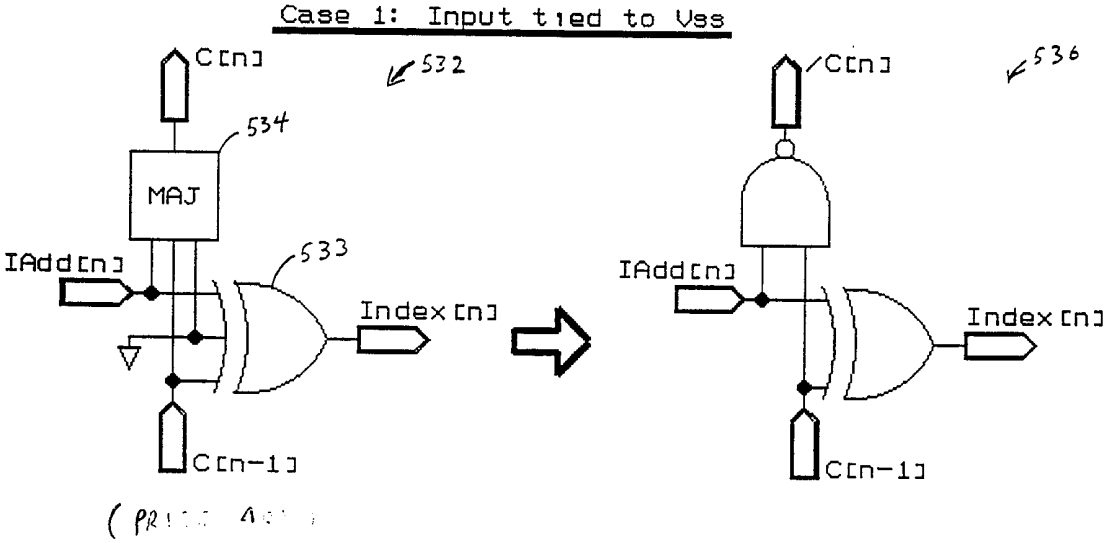

Case 1: Input tied to Vss
C[n]
532
534
MAJ
IAdd[n]
533
Index[n]
C[n-1]
C[n]
536
IAdd[n]
Index[n]
C[n-1]

FIG 5C

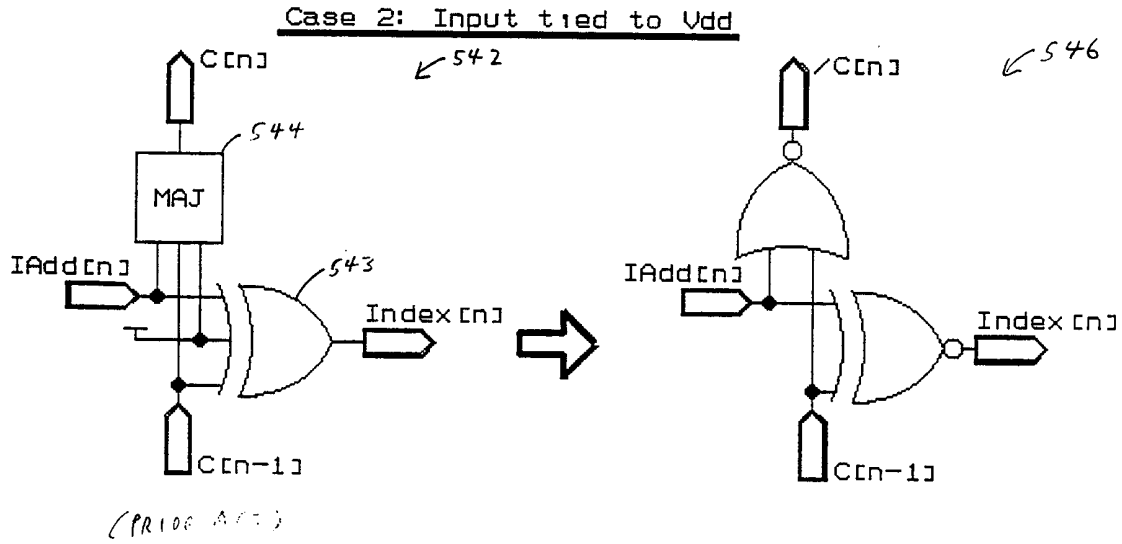

Case 2: Input tied to Vdd
542
C[n]
544
MAJ
IAdd[n]
543
Index[n]
C[n-1]
(PRIOR ART)
546
C[n]
IAdd[n]
Index[n]
C[n-1]

FIG. 5D

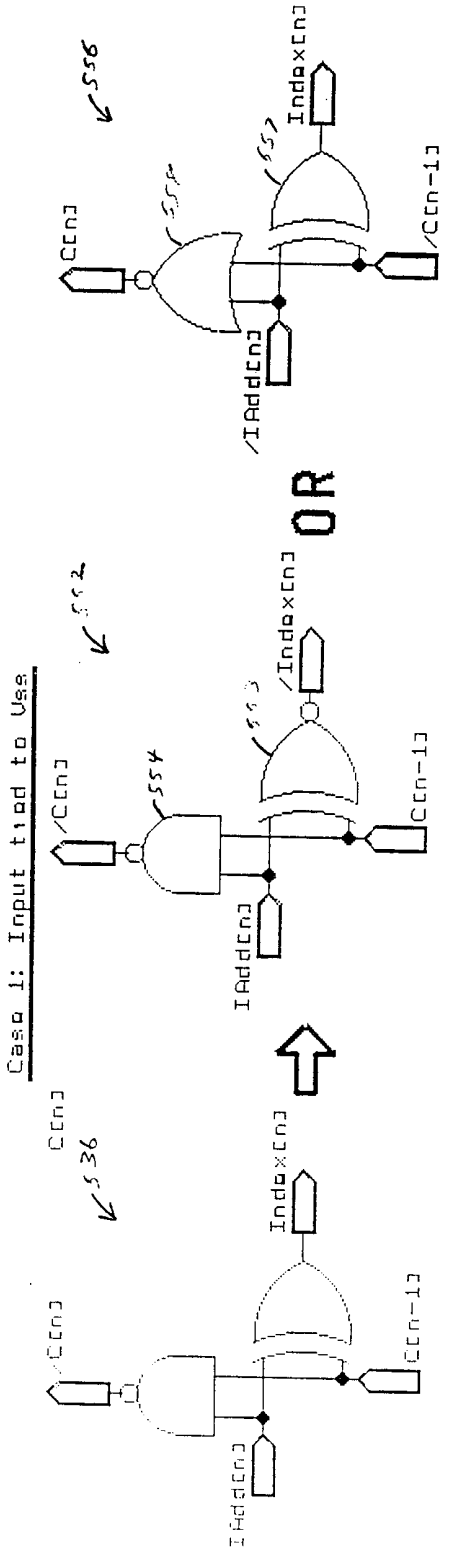

Case 1: Input tied to Vss
Cin
536
/Cin
IAddCin
IndexCin
Cin-1
554
556
OR

FIG. 5F

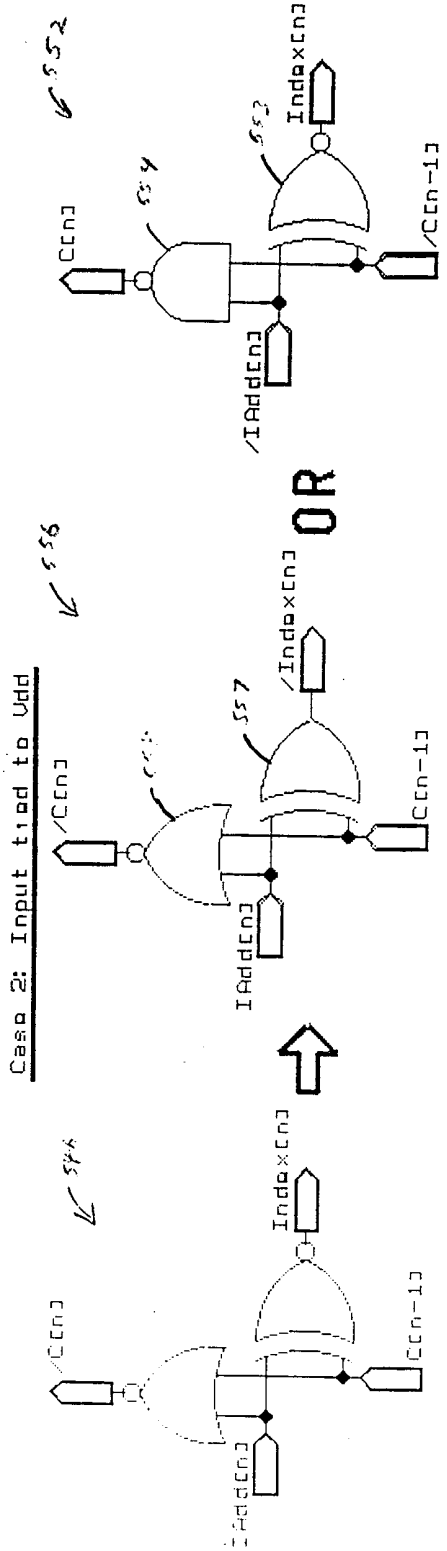
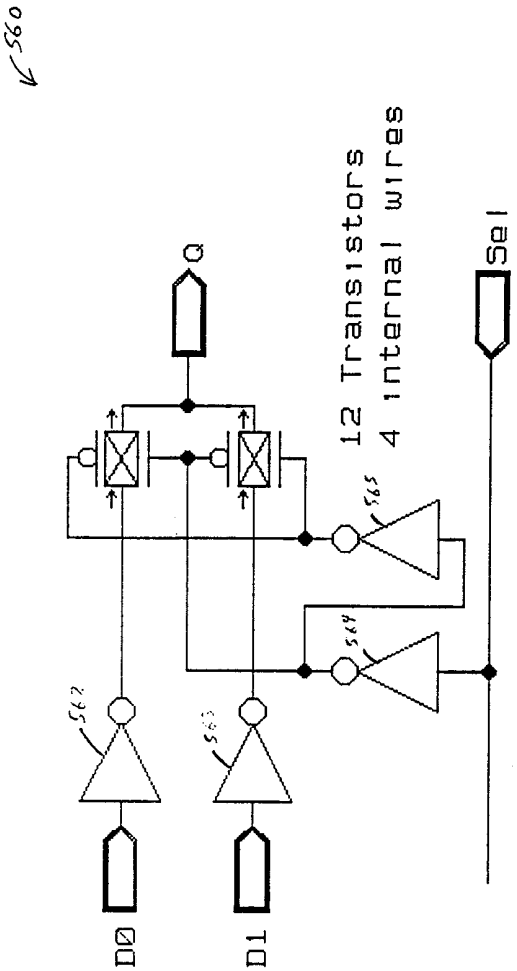

Case 2: Input tied to Vdd
OR
/Cn
/Cn
Cn
/IAddCn
IAddCn
/IndexCn
IndexCn
Cn-1
/Cn-1

FIG. 5F



(Prior Art)

FIG. 5G

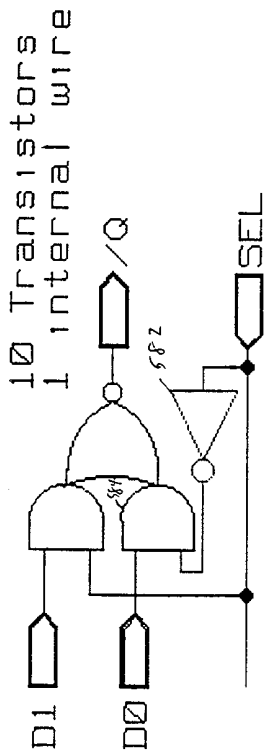


FIG. 5H

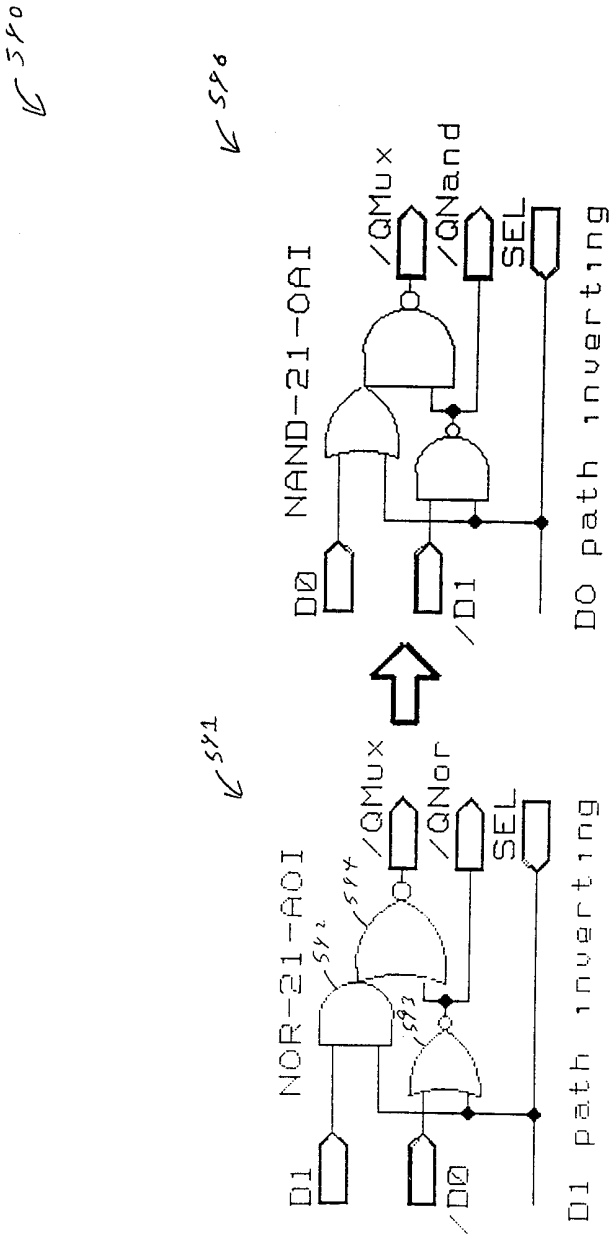

560
590
NAND-21-OAI
/QMux
/QNand
SEL
D0
/D1
D0 path inverting
591
NOR-21-AOI
592
594
/QMux
/QNor
SEL
D1
/D0
D1 path inverting

Fig. 51

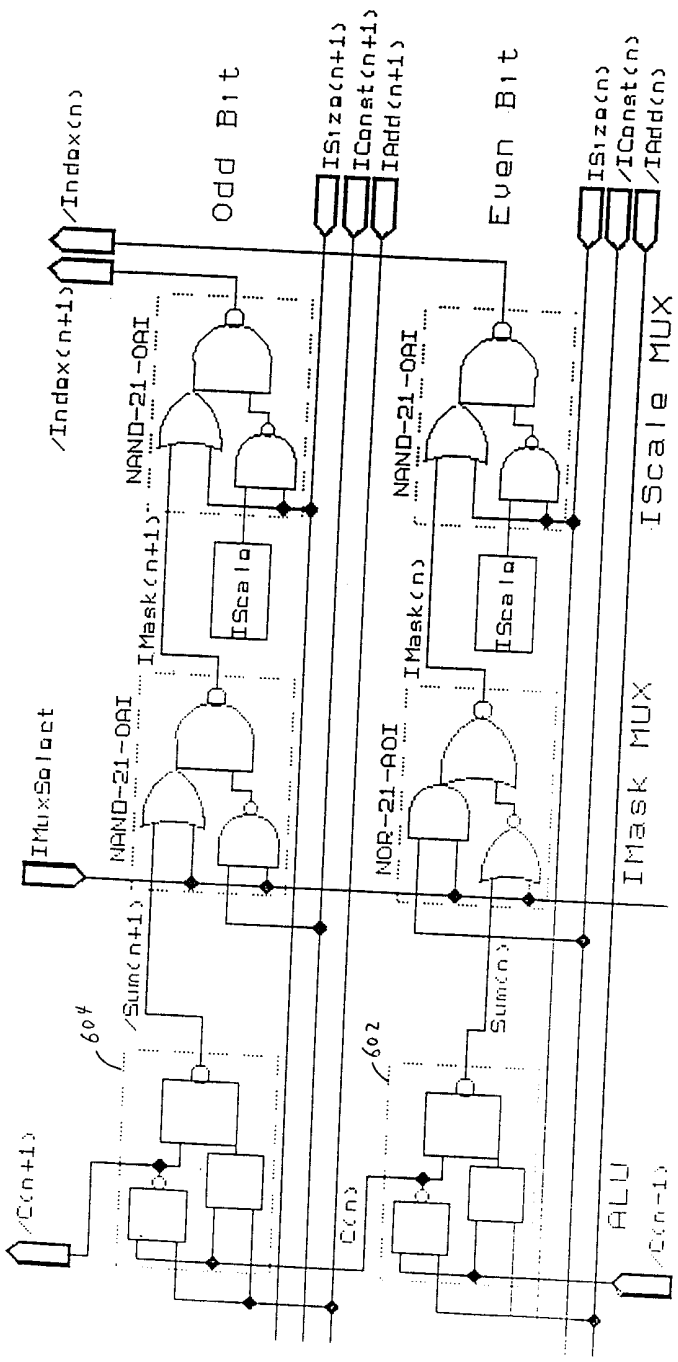

/Indax(n)
/Indax(n+1)
Odd Bit
ISiza(n+1)
ICanst(n+1)
IAdd(n+1)
Even Bit
ISiza(n)
/ICanst(n)
/IAdd(n)
IScale MUX
NAND-21-OAI
IMask(n+1)
IScale
NAND-21-OAI
IMask(n)
IScale
IMuxSelect
NAND-21-OAI
/Sum(n+1)
NOR-21-AOI
Sum(n)
IMask MUX
604
602
/C(n+1)
C(n)
ALU
/C(n-1)

FIG. 6

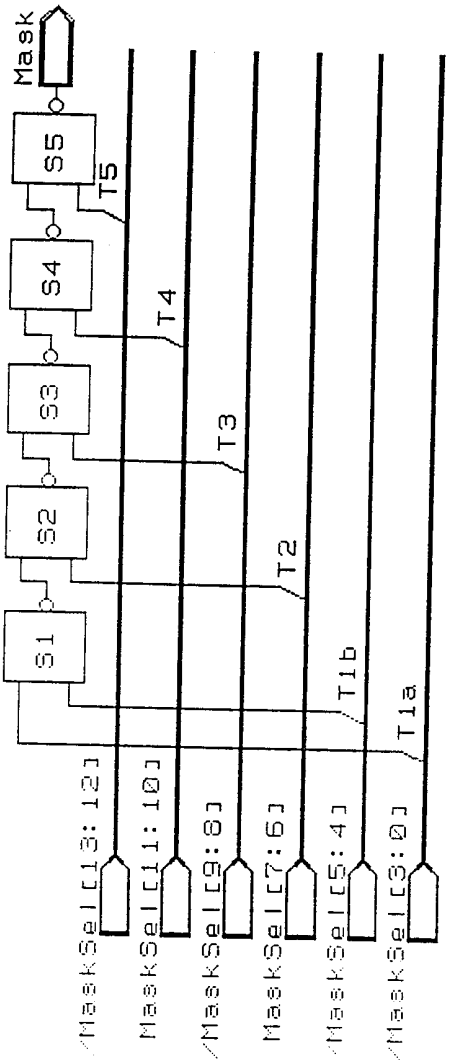

Mask
S5
S4
S3
S2
S1
T5
T4
T3
T2
T1b
T1a
/MaskSelC13:12]
MaskSelC11:10]
/MaskSelC9:8]
MaskSelC7:6]
/MaskSelC5:4]
/MaskSelC3:0]

Fig. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/03566

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :GO6F 9/30

US CL :712/215; 712/217; 712/219; 712 206; 712/210

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. : 712/215; 712/217; 712/219; 712 206; 712/210

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

West Search

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,794,003 A (SACHS) 11 august 1998, col. 2, lines 11-21; col. 3, lines 5-65)	1-12
A	US 5,931,944 A (GINOSAR et al.) 03 August 1999, col. 4, lines 43-64	1-12

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 30 MARCH 2000	Date of mailing of the international search report 25 APR 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer MENG AI AN James R. Matthews Telephone No. (703) 3053855

Form PCT/ISA/210 (second sheet) (July 1998)*