PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 12/08 | A1

(11) International Publication Number: **WO 97/48048**

(43) International Publication Date: 18 December 1997 (18.12.97)

(21) International Application Number: PCT/US97/10184

(22) International Filing Date: 13 June 1997 (13.06.97)

(30) Priority Data:
08/663,386 13 June 1996 (13.06.96) US

(71) Applicant: MICRON TECHNOLOGY, INC. [US/US]; 8000 So. Federal Way, Boise, ID 83707 (US).

(72) Inventor: PAWLOWSKI, Joseph, Thomas; 12171 West Musket Drive, Boise, ID 83704 (US).

(74) Agent: SLIFER, Russell, D.; Schwegman, Lundberg, Woessner & Kluth, P.O. Box 2938, Minneapolis, MN 55402 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: WORD WIDTH SELECTION FOR SRAM CACHE

(57) Abstract

A logic which enables implementation of an 80-bit wide or a 96-bit wide cache SRAM using the same memory array. The logic implementation is accomplished by merging tag, and data into an order block of information to maximize bus utilization. The logic reduces the bus cycles from four cycles for an 80-bit to three cycles for a 96-bit implementation.

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		

WORD WIDTH SELECTION FOR SRAM CACHE

Field of the Invention

The present invention relates to digital computers in general, and to computer memory in particular. More particular still, this invention relates to logic implementation of cache memory.

Background of the Invention

The performance of computer systems, especially personal computers, has improved dramatically due to the rapid growth in computer architecture design and in particular to the performance of computer memory.

Computer processors and memories however have not pursued the same pace of development through the years. Memories are not able to deliver enough response speed to processors. To reduce the gap in speed between the processors and memories, the concept of memory hierarchy was introduced. A memory hierarchy comprises a number of different memory levels, sizes and speeds. The memory located near or inside the processor is usually the smallest and fastest and is commonly referred to as cache memory. Cache memory needs to be fast to accommodate the demand of the processor therefore it is usually constructed from static-type memory or static random access memory (SRAM).

Cache memory plays an important role in the computer memory hierarchy. Computer instructions and data which are most likely to be reused are stored temporarily in the cache memory because the processor can access these instructions or data much faster than accessing them from the slower computer main memory.

Almost all of cache memories are managed by hardware meaning that the cache operation is physically controlled by logic circuits. Implementation of cache memory is not the same in different type of processors since the logic control circuits are different. In some implementations, a processor-cache interface uses a 64-bit bus data an additional bus for tag. The tag bus width varies, but has nominally been 16 bits for a total of 80 bits wide for tag plus data.

If the cache block (or cache line) size is four times the data bus width, then no useful information appears on the tag bus for three out of every four bus cycles therefore the bus is not utilized efficiently.

There is a need for a logic to implement a cache SRAM so that the utilization of data and tag bus can be more efficient. This logic could implement a 64-bit data bus plus a 16-bit or more tag bus but the same logic is also usable to implement a 96-bit bus.

Summary of the Invention

The present invention describes a selection logic which enables the implementation of an 80-bit wide or a 96-bit wide SRAM in a computer system comprising a microprocessor and a cache memory. In one embodiment, the logic enables the implementation of an 80-bit or a 96-bit wide cache SRAM. This logic could also be used to implement two SRAMs having half the width, that is two 40-bit or 48-bit cache SRAMs. The present invention permits higher useful data throughput on an 80 or a 96-bit bus than what has been previously achieved with an 80-bit bus. This logic implementation is accomplished by merging tag, error checking and correction (ECC), and data into an ordered block of information to maximize bus utilization.

The important advantages of this logic implementation is that it utilizes useful information on every bus cycle for an 80-bit bus case and it reduces the number of bus cycles from four to three in the case of a 96-bit bus.

Brief Description of the Drawings

Figure 1 is a block diagram of a simplified computer system comprising a microprocessor and an 80-bit or a 96-bit cache SRAM.

Figure 2 is a block diagram of the 80-/96-bit cache SRAM of Figure 1.

Figure 3 is a block diagram of available routes for data transfer from the memory array to the output blocks for the case of a 96-bit implementation.

Figures 4A-4D are possible output block selection combinations when the initial address is 00, 01, 10, and 11 respectively for the case of a 96-bit implementation.

Figures 5A-5E are the combinations of the logic for the case of a 96-bit implementation according to the present invention.

Figure 6 is a block diagram of available routes for data transfer from the memory array to the output blocks for the case of an 80-bit implementation.

Figures 7A-7D are possible output block selection combinations for the case of an 80-bit implementation.

Figures 8A-8E are the combinations of the logic for the case of an 80-bit implementation according to the present invention.

Figures 9A-9E are the combinations of the logic for both cases of an 80-bit and a 96-bit implementation according to the present invention.

Description of the Preferred Embodiment

In the following detailed description of the preferred embodiment, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that structural changes may be made without departing from the spirit and scope of the present inventions. The following detailed description is, therefore, not to be taken in limiting sense, and the scope of the invention is defined by the appended claims.

Figure 1 illustrates a simplified computer system comprising a microprocessor 150 connected to an 80-/96-bit cache SRAM 100 through a processor-cache interface 160. Processor-cache interface 160 comprises a system clock (CLK), an address data strobe (ADS#), a read or write request (RW#), an address bus, a tag bus and an data bus.

Figure 2 is block diagram of the 80-/96-bit cache SRAM 100 of Figure 1. Cache SRAM 100 can support an 80 to 96 bit data bus. The 80-bit or 96-bit operations are accomplished by a data ordering scheme and logic selections of input select logic 106 and output select logic 108. Input and output logic 106 and 108 are logically identical. Data are transferred to and from a data and tag memory array 110 in three bus cycles in the case of an 96-bit and in four bus

cycles in the case of a 80-bit system. The sequence of the bus cycles is monitored by a bus cycle counter 102. Cycle counter 102 begins when ADS# is low and resets to zero and holds after three counts (cycle1, cycle2 then cycle3 for a 96-bit system), or after four counts (cycle1, cycle2, cycle3 then cycle4 for an 80-bit system). Data is written to or is read from memory array 110 by a write operation or read operation respectively. In the diagram, RW# indicates whether a read or write operation is requested, the sign # indicating that it writes if the signal is low. Address represents the sough memory location in memory array 110. Data represents a composite collection of data and tag bits.

Figure 3 is a block diagram of available routes for data to be transferred from the memory array to the output blocks for the case of a 96-bit implementation. This embodiment shows a portion of memory array 210 which comprises four 64-bit longwords A, B, C and D and two tag words totaling up to 32 bits and are indicated as tag.1 and tag.2. Tag in this and in other embodiments represents additional information such as status, ECC, tag, etc. Each of the four 64-bit longwords is divided into four 16-bit words. Longword A has four 16-bit words indicated as 1.1, 1.2, 1.3 and 1.4. Longword B has four 16-bit words indicated as 2.1, 2.2, 2.3 and 2.4. Longword C has four 16-bit words indicated as 3.1, 3.2, 3.3 and 3.4 and longword D has four 16-bit words indicated as 4.1, 4.2, 4.3 and 4.4. In this embodiments, 1.1 represents datum 1; word 1, 1.2 represents datum 1 word 2 , 1.3 represents datum 1; word 3, and etc.

Words A, B, C and D, in that order, represent the order of data criticality to the processor. The actual physical address which is considered critically ordered differs from processor to processor in existing implementations and may entail a modula-4 linear burst, a modula-4 interleaved order, etc. For the typical line addressing microprocessor (e.g. PowerPC or Cyrix M1), the optimal order is modula-4 linear burst. This ordering is shown in Table A. Any other ordering for this type of processor will prevent maximization of performance for a processor designed to utilize the 96-bit operation. The reason for this is that, in the course of operating on a whole block of data, the highest probability of utilizing data in the block is 100% for the initial address, and less for each

subsequent address. The probability is much lower for the previous address. Therefore, if the initial address is 01, the address before it, namely 00, is probably the least necessary to have and should therefore have less priority. Hence, A, D, C, and D would show the following sequence represented in binary form in which x stands for "any":

Table A: Linear Burst Data Ordering in a Four Entry Cache Line

Initial address	A	B	C	D
x00	x00	x01	x10	x11
x01	x01	x10	x11	x00
x10	x10	x11	x00	x01
x11	x11	x00	x01	x10

For processors which required interleaved burst order (e.g. Intel Pentium), a modula-4 interleaved burst order could be used. This ordering is shown in Table B.

Table B: Interleaved Burst Data Ordering in a Four Entry Cache Line

Initial address	A	B	C	D
x00	x00	x01	x10	x11
x01	x01	x00	x11	x10
x10	x10	x11	x00	x01
x11	x11	x10	x01	x00

In one embodiment, the order in which the cache line data words are transferred is programmable. Such a device would permit, for example, both interleaved and linear burst data ordering with the same cache device. In another embodiment, data ordering could be changed to reflect characteristics of the programs or programs being executed (e.g. a program operating at particular stride through memory).

Referring again to Figure 3, data are transferred from memory array 210 to output blocks 230 by a logic selection from a plurality of pathways 220. Pathways 220 comprises thirty-four routes in which six routes 221 through 226 are connected to the output blocks 230 at each 16-bit output blocks 231 through 236 indicated respectively as OB1 to OB6. An output block is comprised of output buffers and optionally data registers or latches. The logic which enables six of these thirty-four available routes will be described below.

Figures 4A-4D are possible output block selection combinations when the initial address is 00, 01, 10, and 11 respectively for the case of a 96-bit implementation. These Figures clearly shows that a 96-bit bus can be implemented using only three bus cycles. Tag only appears in the first bus cycle (cycle1), freeing the input/output lines for data transfer during cycle two (cycle2) and cycle three (cycle3). This ordering simplies the logic needed to transfer the cache line data words and reduces the number of paths that must be available. The logic which enables these possible output block selection combinations is described in Figures 5A-5E.

Figures 5A-5E are the combinations of the logic for the case of a 96-bit implementation. For this 96-bit case, only three bus cycles are needed and the order of data transaction is cycle1, then cycle2, and finally cycle3. In this embodiment, the logic comprises a combination of input 410, logic gates 420 and a plurality of outputs 430. Logic gates 420 comprises a plurality of logic AND and logic OR gates. Input 410 driving the logic comprises cycle1, cycle2, cycle3, A0 and A1. A0 and A1 represent the two least significant bits of the initial address. Cycle1, cycle2 or cycle3 is the current bus cycle determined by bus cycle counter 102. Outputs 430 from this logic enables the data to be transferred to the appropriate blocks OB1 through OB6 of output blocks 230. Detailed combinations of the logic enabling at output 430 are described in Table 1. In this table, OB represents output block, IA represents the least two significant bits of the initial address, tag.1 and tag.2 represent the miscellaneous additional information such as status, ECC, tag, etc. and 1.1 represents datum 1: word 1 in the current cache line , 1.2 represents datum 1; word 2, etc.

Table 1: Logic for a 96-bit Linear Burst Data Ordering Implementation

---

Enable tag.1 to OB1 = Cycle1

Enable tag.2 to OB2 = Cycle1

Enable 1.1 to OB3 = Cycle1 AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10)

Enable 1.2 to OB4 = Cycle1 AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10)

Enable 1.3 to OB5 = Cycle1 AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10)

Enable 1.4 to OB6 = Cycle1 AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10)

Enable 1.1 to OB1 = Cycle2 AND (IA=01)

Enable 1.2 to OB2 = Cycle2 AND (IA=01)

Enable 1.3 to OB1 = Cycle3 AND (IA=01)

Enable 1.4 to OB2 = Cycle3 AND (IA=01)

Enable 2.1 to OB3 = Cycle1 AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11)

Enable 2.2 to OB4 = Cycle1 AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11)

Enable 2.3 to OB5 = Cycle1 AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11)

Enable 2.4 to OB6 = Cycle1 AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11)

Enable 2.1 to OB1 = Cycle2 AND (IA=10)

Enable 2.2 to OB2 = Cycle2 AND (IA=10)

Enable 2.3 to OB1 = Cycle3 AND (IA=10)

Enable 2.4 to OB2 = Cycle3 AND (IA=10)

Enable 3.1 to OB3 = Cycle1 AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00)

Enable 3.2 to OB4 = Cyclel AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00)

Enable 3.3 to OB5 = Cyclel AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00)

Enable 3.4 to OB6 = Cyclel AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00)

Enable 3.1 to OB1 = Cycle2 AND (IA=11)

Enable 3.2 to OB2 = Cycle2 AND (IA=11)

Enable 3.3 to OB1 = Cycle3 AND (IA=11)

Enable 3.4 to OB2 = Cycle3 AND (IA=11)

Enable 4.1 to OB3 = Cyclel AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01)

Enable 4.2 to OB4 = Cyclel AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01)

Enable 4.3 to OB5 = Cyclel AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01)

Enable 4.4 to OB6 = Cyclel AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01)

Enable 4.1 to OB1 = Cycle2 AND (IA=00)

Enable 4.2 to OB2 = Cycle2 AND (IA=00)

Enable 4.3 to OB1 = Cycle3 AND (IA=00)

Enable 4.4 to OB2 = Cycle3 AND (IA=00)

Those skilled in the art will readily recognize that the above description for a 96-bit bus implementation could also be used to implement a 96-bit wide device using two 48-bit wide devices. The 96-bit implementation for the two 48-bit wide devices would be implemented as such: all the even words are in one device, and all the odd words are in the other device. For example, words 1.4, 2.4, 3,4 , 4.4, 1.2, 2.2, 3.2, 4.2 (x.4, x.2), OB6, OB4, OB2 are in one device, and x.3, x.1, OB5, OB3, OB1 are in the other device. The described logic operates

exactly as explained and the devices operate together seamlessly, and only one design is necessary; two identical devices are used in the implementation.

Figure 6 is a block diagram of available routes for data to be transferred from the memory array to the output blocks for the case of an 80-bit implementation. In this embodiment, longwords A, B, C and D are arranged in the same structure in the portion of memory array 510 as in Figure 3 for the case of 96-bit implementation however, up to four tag words indicated as tag.1, tag.2, tag.3 and tag.4 can be utilized. The output blocks 530 in this embodiment comprises five 16-bit output block 531, 533, 534, 535 and 536 which are indicated respectively as OB1, OB3, OB4, OB5 and OB6. Data are transferred from the memory array 510 to output blocks 530 by a logic selection from a plurality of pathways 520. Pathways 520 comprises up to twenty routes in which five routes 521, 523, 524, 525 and 526 are connected to output blocks 530.

Figures 7A-7D are possible output block selection combinations when the initial address is 00, 01, 10, and 11 respectively for the case of an 80-bit implementation. These Figures shows that four bus cycles are needed for data transfer. In this case, tag or useful information appears on every bus cycles (cycle1 through cycle4) therefore it is an efficient utilization of bus. In this 80-bit implementation, to maximize performance, the tag limit is 16 bits. If more tag bits are need, the 80-bits would be expanded within reason to accommodate the additional necessary bits. For example, if a 20-bit tag is essential, this would entail an 84-bit bus. 11 bits of ECC is sufficient regardless of tag size, within reason. The logic which enables these possible output block selection combinations is described in Figures 8A-8E.

Figures 8A-8E are the combinations of the logic for the case of an 80-bit implementation. For this 80-bit case, four bus cycles are needed and the order of data transaction is cycle1, then cycle2, then cycle3, and finally cycle4. In this embodiment, the logic comprises a combination of input 710, logic gates 720 and a plurality of outputs 730. Logic gates 720 comprises logic a plurality of logic AND and logic OR gates. Input 710 driving the logic comprises cycle1,

cycle2, cycle3, cycle4, A0 and A1. A0 and A1 represent the two least significant bits of the initial address. Cycle1, cycle2 or cycle3 is the current bus cycle determined by bus cycle counter 102. Outputs 730 from this logic enables the data to be transferred to the appropriate blocks of output blocks 530. Detailed combinations of the logic enabling at output 730 are described in Table 2. In this table, OB represents output block, IA represents the least two significant bits of the initial address, tag.1 and tag.2 represent the miscellaneous additional information such as status, ECC, tag, etc. and 1.1 represents datum 1: word 1 in the current cache line , 1.2 represents datum 1; word 2, etc.

Table 2: Logic for an 80-bit Linear Burst Data Ordering Implementation

---

Enable tag.1 to OB1 = Cyclel

Enable tag.2 to OB1 = Cycle2

Enable tag.3 to OB1 = Cycle3

Enable tag.4 to OB1 = Cycle4

Enable 1.1 to OB3 = Cyclel AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10) OR Cycle4 AND (IA=01)

Enable 1.2 to OB4 = Cyclel AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10) OR Cycle4 AND (IA=01)

Enable 1.3 to OB5 = Cyclel AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10) OR Cycle4 AND (IA=01)

Enable 1.4 to OB6 = Cyclel AND (IA=00) OR Cycle2 AND (IA=11) OR Cycle3 AND (IA=10) OR Cycle4 AND (IA=01)

Enable 2.1 to OB3 = Cyclel AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11) OR Cycle4 AND (IA=10)

Enable 2.2 to OB4 = Cyclel AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11) OR Cycle4 AND (IA=10)

Enable 2.3 to OB5 = Cyclel AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11) OR Cycle4. AND (IA=10)

Enable 2.4 to OB6 = Cycle1 AND (IA=01) OR Cycle2 AND (IA=00) OR Cycle3 AND (IA=11) OR Cycle4 AND (IA=10)

Enable 3.1 to OB3 = Cycle1 AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00) OR Cycle4 AND (IA=11)

Enable 3.2 to OB4 = Cycle1 AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00) OR Cycle4 AND (IA=11)

Enable 3.3 to OB5 = Cycle1 AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00) OR Cycle4 AND (IA=11)

Enable 3.4 to OB6 = Cycle1 AND (IA=10) OR Cycle2 AND (IA=01) OR Cycle3 AND (IA=00) OR Cycle4 AND (IA=11)

Enable 4.1 to OB3 = Cycle1 AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01) OR Cycle4 AND (IA=00)

Enable 4.2 to OB4 = Cycle1 AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01) OR Cycle4 AND (IA=00)

Enable 4.3 to OB5 = Cycle1 AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01) OR Cycle4 AND (IA=00)

Enable 4.4 to OB6 = Cycle1 AND (IA=11) OR Cycle2 AND (IA=10) OR Cycle3 AND (IA=01) OR Cycle4 AND (IA=00)

Those skilled in the art will readily recognize that the implementation for a 80-bit wide device described above could also be used to implement an 80-bit bus wide in a memory device using more than one device. For example, if the 80-bit bus is split across two devices, OB1 would have to be split into two 8-bit halves so that two such identical devices would comprise the 80-bit bus device. As such only one device type is needed, and that device is used twice. The same principle applied in a four device implementation. In this case, the tag and OB1 is split evenly among four devices.

From the illustrations and descriptions of Figure 3 through Figure 8E, it is apparent that there are commonality in the available pathways and in the logic selections between the 80-bit and 96 bit implementations. Further examining of Figure 3 (available routes for a 96-bit implementation) and Figure 6 (available

routes for an 80-bit implementation), a conclusion can be drawn that Figure 6 is a subset of Figure 3. Further examining of Figures 5A-5E (logic for a 96-bit implementation) and Figures 8A-8E (logic for an 80-bit implementation) and also Table 1 and Table 2, a modification could be made to the logic so that it can implement both 80-bit and 96-bit implementation from the same memory array. Thus the block diagram of the routes in Figure 3 can be used for both cases of an 80-bit and 96-bit implementation, and the modified logic to implement both cases is illustrated in Figures 9A-9E.

Figures 9A-9E are the combinations of the logic for both cases of an 80-bit and a 96-bit implementation according to the present invention. This embodiment shows the logical differences between the 80-bit and the 96-bit implementation and identifies the logic that is common and specific to each implementation. In Figures 9A-9E, the common logic in each Figure is the entire logic excluding optional logic indicated at 96 and optional logic indicated at 80. The common logic and optional logic 96 are active only in the case of a 96-bit implementation. The common logic and optional logic 80 are active only in the case of an 80-bit implementation.

From the detailed description of the invention, an 80-bit implementation is carried out by four bus cycles and useful information is on every cycle therefore the bus utilization is more efficient. A 96-bit implementation requires only three instead of four cycles thus speeding up the data transaction process. The block selections described in these embodiments are in terms of output but it is also understood that the input ordering is identical and follows the same logic. Furthermore it is clear that implementation of an 80-bit and a 96-bit device using the same memory array can be obtained by the logic described in the present invention.

What is claimed is:

1. A computer system comprising:

a processor;

a cache memory having a plurality of cache lines and tag lines; and

a reconfigurable processor-cache interface including an N-bit data bus wherein the range of N is 64 to 96.

2. The computer system according to claim 1 wherein the N-bit data bus is configurable as 96-bit wide data bus, the width of the cache lines is four times the width of the data bus and a data transaction between the processor and the cache memory is completed in three bus cycles.

3. The computer system according to claim 1 wherein the N-bit data bus is configurable as 80-bit wide data bus, the width of the cache lines is four times the width of the data bus and a data transaction between the processor and the cache memory is completed in four bus cycles.

4. The computer system according to claim 2 wherein the cache lines comprises a plurality of longwords, the tag lines comprises a plurality of tag words, wherein each of the longwords comprises a plurality of words which together with the tag words are merged into an ordered block of information by an ordering scheme to allow the optimal use of the N-bit bus.

5. The computer system according to claim 4 wherein the cache memory further comprises a plurality of data routing pathways which enables the words and tag words to be merged into the ordered block.

6. The computer system according to claim 5 wherein the cache memory further comprises an input selection logic and an output selection logic, which provide logical control to the routing of data on the data routing pathways such

that the words and tag words are properly transferred to the ordered block according to the data ordering scheme.

7. The computer system according to claim 6 wherein the input selection logic and the output selection logic are logically the same.

8. The computer system according to claim 3 wherein the cache lines comprises a plurality of longwords, the tag lines comprises a plurality of tag words, wherein each longword comprises a plurality of words which together with the tag words are merged into an ordered block of information by an ordering scheme to allow the optimal use of the N-bit bus.

9. The computer system according to claim 3 wherein the cache memory further comprises a plurality of data routing pathways which enables the words and tag words to be merged into the ordered block.

10. The computer system according to claim 3 wherein the cache memory further comprises an input selection logic and an output selection logic, which provide logical control to the routing of data on the data routing pathways such that the words and tag words are properly transferred to the ordered block according to the data ordering scheme.

11. The computer system according to claim 3 wherein the input selection logic and the output selection logic are logically the same.

12. A cache memory, comprising:

a memory array having a plurality tag words, and a plurality of longwords including first, second, third and fourth longwords, and wherein each longword comprises a plurality of words;

a plurality of output blocks connected to the memory array for holding data;

a plurality of data routing pathways comprising thirty-four pathways connected to the output blocks for data transfer between the memory array and the output blocks;

a cache interface having an N-bit wide data bus connected to the output blocks;

an input select logic having a plurality of common logics, and a plurality of optional logics including first and second optional logics, and a plurality of inputs and outputs;

an output select logic having a plurality of common logics, and a plurality of optional logics including first and second optional logics, and a plurality of inputs and outputs;

a bus cycles counter producing a plurality of sequential count cycles to keep track of the count cycles of the N-bit wide data bus.

13. The cache memory according to claim 12 wherein the input select logic and the output select logic are logically the same.

14. The cache memory according to claim 13 wherein the input and the output select output logic operate in a way such that the common logics are always operative, the first optional logics are operative only when N is 80, and the second optional logics are operative only when n is 96.

15. The cache memory according to claim 14 wherein the inputs and the outputs of the input and output select logic comprises sequential count cycles of the cycle counter and two least significant digits of the initial address of the longwords.

16. The cache memory according to claim 15 wherein the bus cycle counter resets to zero after counting the first, the second and the third count cycle when the N-bit data bus is a 96-bit wide data bus, and the cycle counter resets to zero

after counting the first, the second, the third and the fourth count cycle when the N-bit data bus is a 80-bit wide data bus.

17. The cache memory according to claim 16 wherein all the tag words are transferred on the first bus cycle when the N-bit data bus is a 96-bit wide data bus, and the tag words are present in every bus cycle when the N-bit data bus is an 80-bit wide data bus.

18. The cache memory according to claim 17 wherein the input and output select logic enable up to six of the thirty-four data routing pathways to transfer data from the memory array to the associated output blocks.

19. The cache memory according to claim 18 wherein the input select logic and the output select logic is configured the same.

20. A method of operating a cache memory in a computer system, comprising the steps of:

configuring a processor-cache interface as an N-bit data bus where N is between 64 and 96;

configuring a cache memory to have a plurality of longwords;

ordering the data in the cache memory by priority according to an order critical to the processor to produce a critical order;

retrieving the data in the order according to the critical order;

logically selecting the data in the order according to the critical order; and

outputting the data in the order according the critical order.

21. The method of claim 20 wherein the priority depends on the initial address of the longwords such that the priority is a first priority for an initial address a second priority for each subsequent address and a third priority for the previous address.

22. The method of claim 20 wherein the retrieving of the data is achieved in three bus cycle when the N-data bus is configured as a 96-bit wide data bus.

23. The method of claim 20 wherein the retrieving of the data is achieved in four bus cycles when the N-data bus is configured as a 80-bit wide or more data bus.

24. The method of claim 20 wherein the logical selection of the data to and from the memory array are the same.

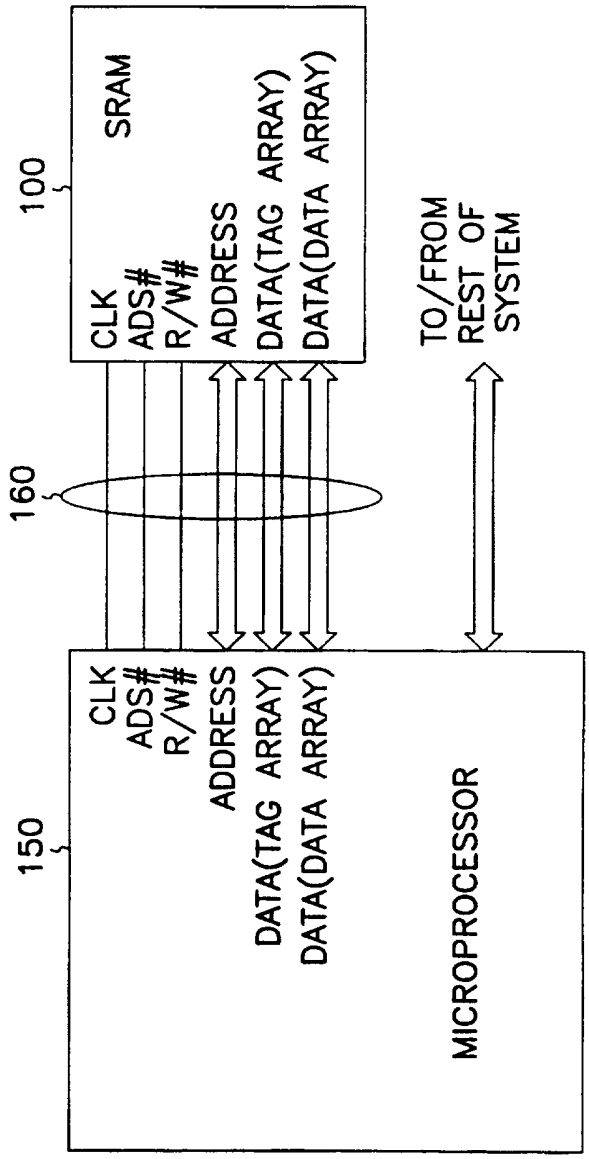
100
SRAM
CLK
ADS#
R/W#
ADDRESS
DATA(TAG ARRAY)
DATA(DATA ARRAY)
TO/FROM
REST OF
SYSTEM
160
150
CLK
ADS#
R/W#
ADDRESS
DATA(TAG ARRAY)
DATA(DATA ARRAY)
MICROPROCESSOR

FIG. 1

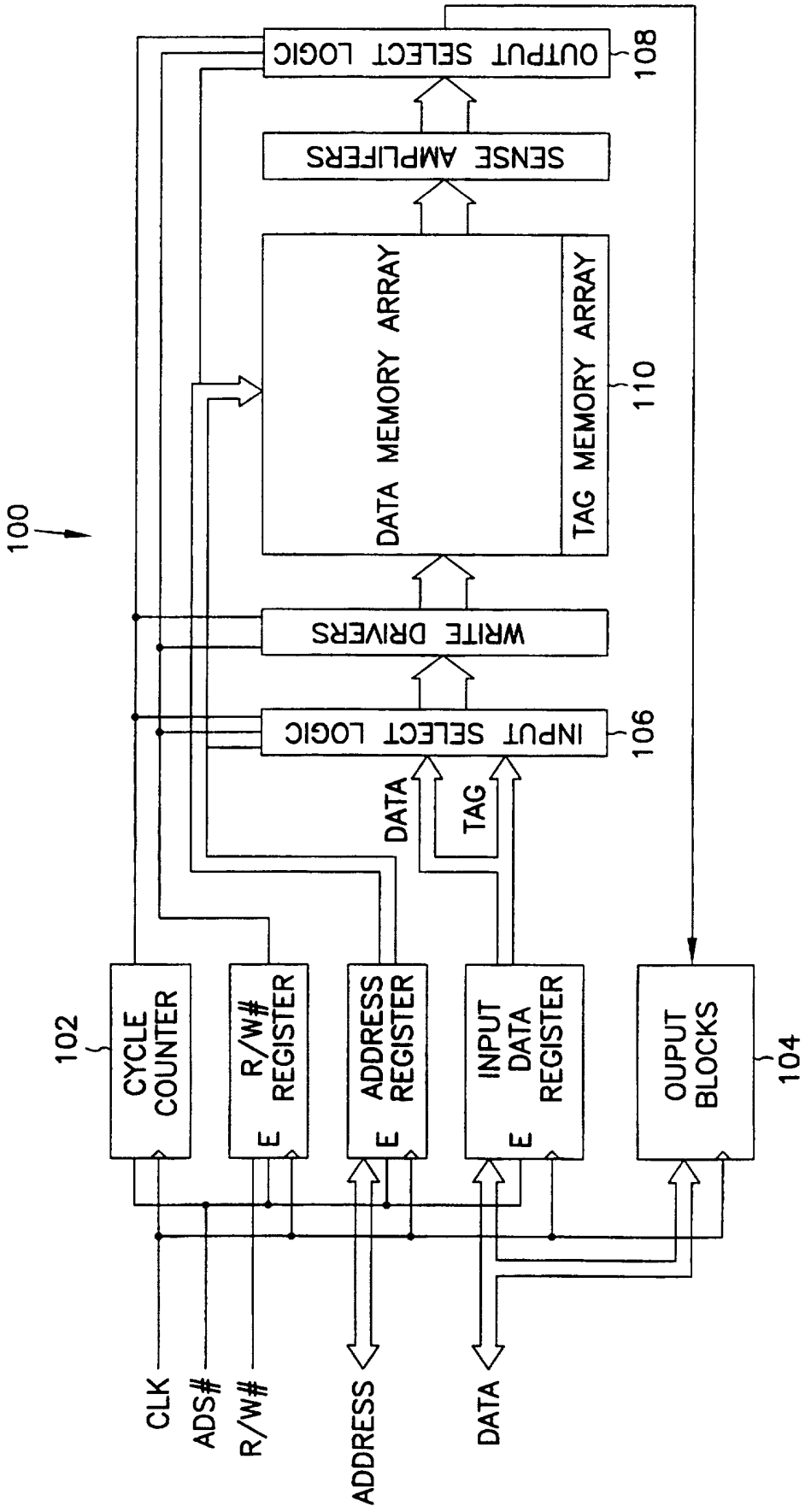
100
OUTPUT SELECT LOGIC
108
SENSE AMPLIFERS
DATA MEMORY ARRAY
TAG MEMORY ARRAY
110
WRITE DRIVERS
INPUT SELECT LOGIC
106
DATA
TAG
102
CYCLE COUNTER
R/W# REGISTER
ADDRESS REGISTER
INPUT DATA REGISTER
OUPUT BLOCKS
104
E
E
E
CLK
ADS#
R/W#
ADDRESS
DATA

FIG. 2

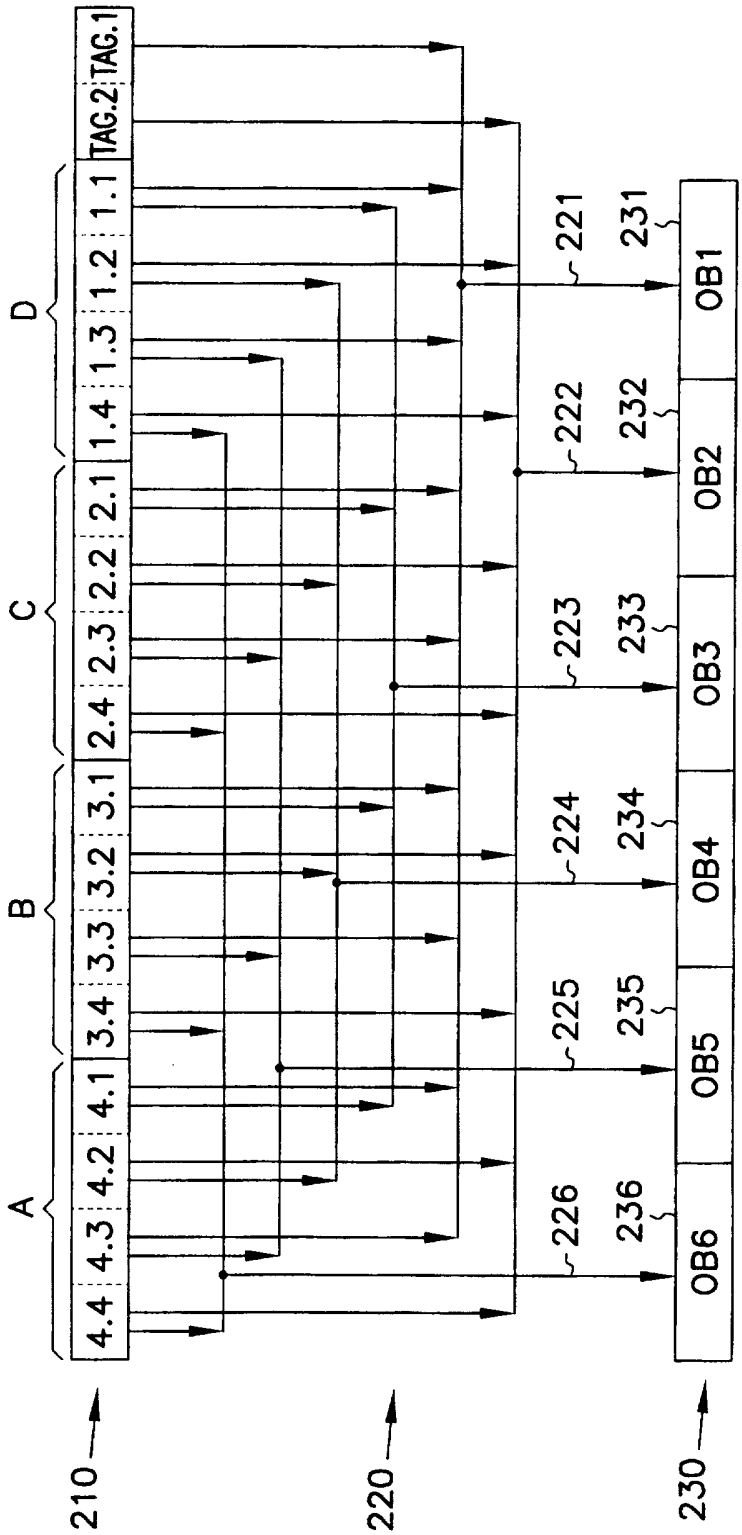
A
B
C
D
TAG.1
TAG.2
1.1
1.2
1.3
1.4
2.1
2.2
2.3
2.4
3.1
3.2
3.3
3.4
4.1
4.2
4.3
4.4
221
222
223
224
225
226
231
232
233
234
235
236
OB1
OB2
OB3
OB4
OB5
OB6
210
220
230

FIG. 3

INITIAL ADDRESS 00	OB6	OB5	OB4	OB3	OB2	OB1
CYCLE1	1.4	1.3	1.2	1.1	TAG.2	TAG.1
CYCLE2	2.4	2.3	2.2	2.1	4.2	4.1
CYCLE3	3.4	3.3	3.2	3.1	4.4	4.3

FIG. 4A

INITIAL ADDRESS 01	OB6	OB5	OB4	OB3	OB2	OB1
CYCLE1	2.4	2.3	2.2	2.1	TAG.2	TAG.1
CYCLE2	3.4	3.3	3.2	3.1	1.2	1.1
CYCLE3	4.4	4.3	4.2	4.1	1.4	1.3

FIG. 4B

INITIAL ADDRESS 10	OB6	OB5	OB4	OB3	OB2	OB1
CYCLE1	3.4	3.3	3.2	3.1	TAG.2	TAG.1
CYCLE2	4.4	4.3	4.2	4.1	2.2	2.1
CYCLE3	1.4	1.3	1.2	1.1	2.4	2.3

FIG. 4C

INITIAL ADDRESS 11	OB6	OB5	OB4	OB3	OB2	OB1
CYCLE1	4.4	4.3	4.2	4.1	TAG.2	TAG.1
CYCLE2	1.4	1.3	1.2	1.1	3.2	3.1
CYCLE3	2.4	2.3	2.2	2.1	3.4	3.3

FIG. 4D

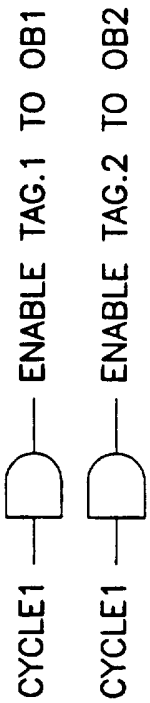

FIG. 5A

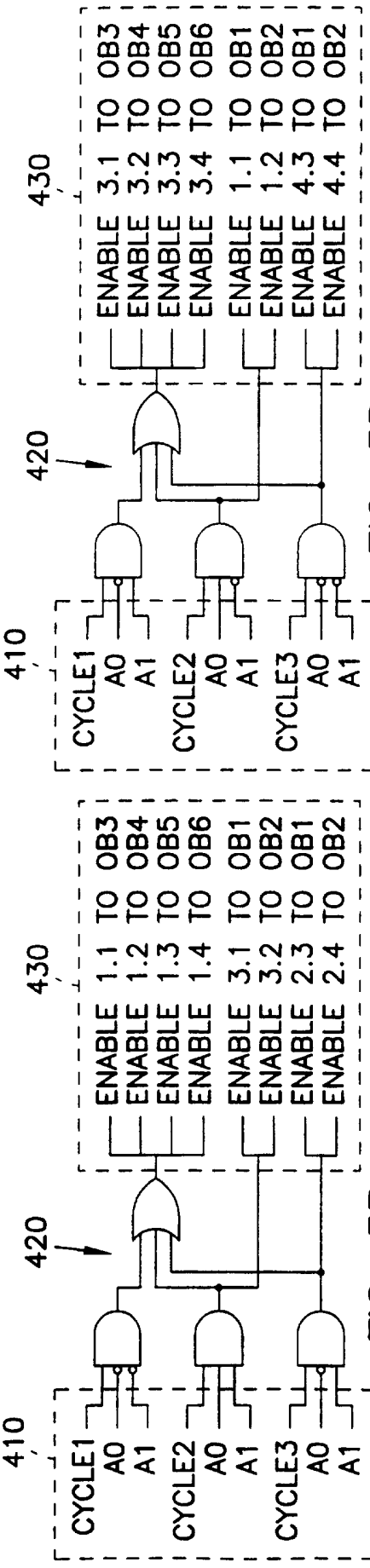

FIG. 5B

FIG. 5D

FIG. 5C

FIG. 5E

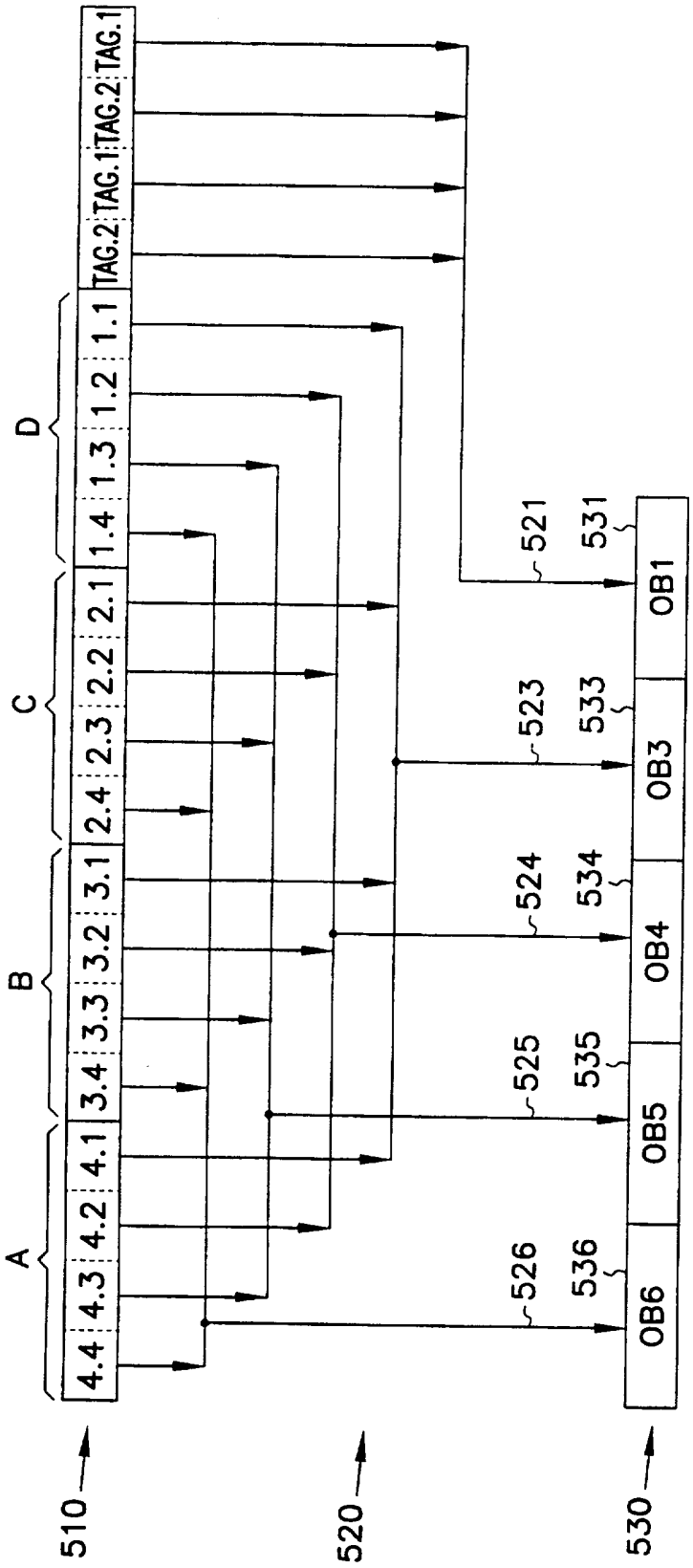
A
B
C
D
4.4
4.3
4.2
4.1
3.4
3.3
3.2
3.1
2.4
2.3
2.2
2.1
1.4
1.3
1.2
1.1
TAG.2
TAG.1
TAG.2
TAG.1
510
520
530
521
523
524
525
526
531
533
534
535
536
OB1
OB3
OB4
OB5
OB6

FIG. 6

INITIAL ADDRESS 00	OB6	OB5	OB4	OB3	OB1
CYCLE1	1.4	1.3	1.2	1.1	TAG.1
CYCLE2	2.4	2.3	2.2	2.1	TAG.2
CYCLE3	3.4	3.3	3.2	3.1	TAG.3
CYCLE4	4.4	4.3	4.2	4.1	TAG.4

FIG. 7A

INITIAL ADDRESS 01	OB6	OB5	OB4	OB3	OB1
CYCLE1	2.4	2.3	2.2	2.1	TAG.1
CYCLE2	3.4	3.3	3.2	3.1	TAG.2
CYCLE3	4.4	4.3	4.2	4.1	TAG.3
CYCLE4	1.4	1.3	1.2	1.1	TAG.4

FIG. 7B

INITIAL ADDRESS 10	OB6	OB5	OB4	OB3	OB1
CYCLE1	3.4	3.3	3.2	3.1	TAG.1
CYCLE2	4.4	4.3	4.2	4.1	TAG.2
CYCLE3	1.4	1.3	1.2	1.1	TAG.3
CYCLE4	2.4	2.3	2.2	2.1	TAG.4

FIG. 7C

INITIAL ADDRESS 11	OB6	OB5	OB4	OB3	OB1
CYCLE1	4.4	4.3	4.2	4.1	TAG.1
CYCLE2	1.4	1.3	1.2	1.1	TAG.2
CYCLE3	2.4	2.3	2.2	2.1	TAG.3
CYCLE4	3.4	3.3	3.2	3.1	TAG.4

FIG. 7D

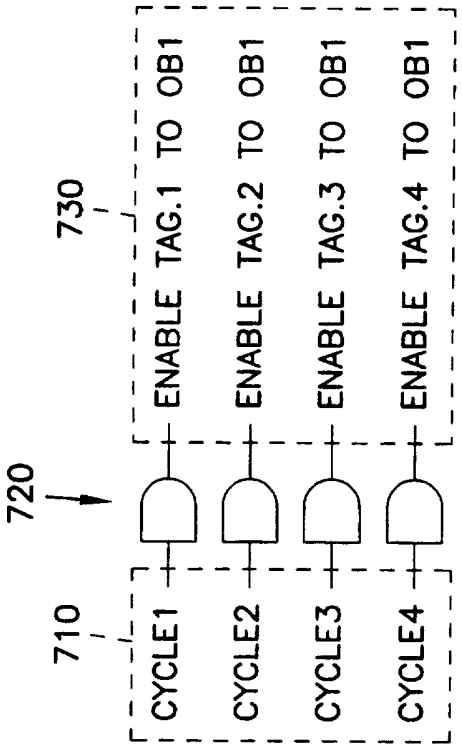
730
720
710
CYCLE1
CYCLE2
CYCLE3
CYCLE4
ENABLE TAG.1 TO OB1
ENABLE TAG.2 TO OB1
ENABLE TAG.3 TO OB1
ENABLE TAG.4 TO OB1

FIG. 8A

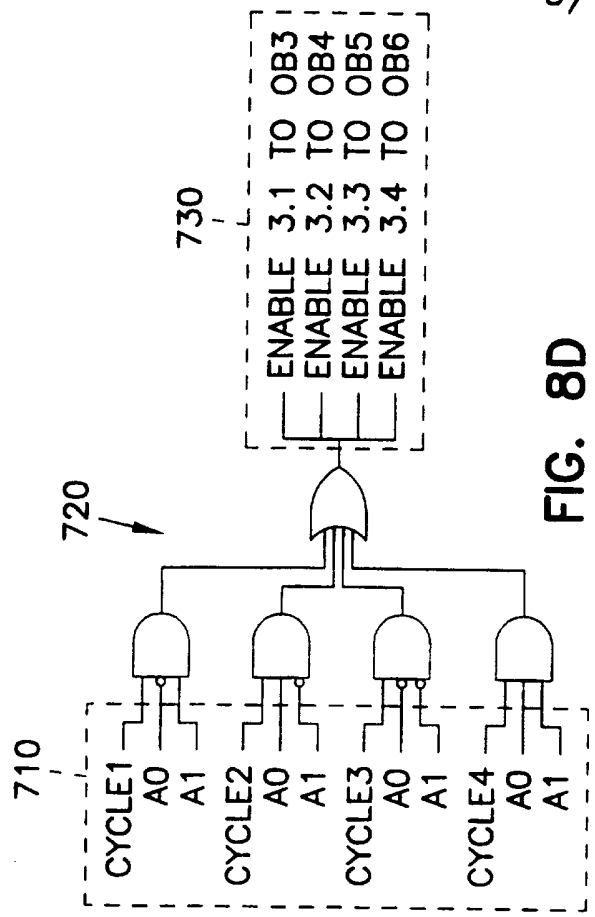

FIG. 8D

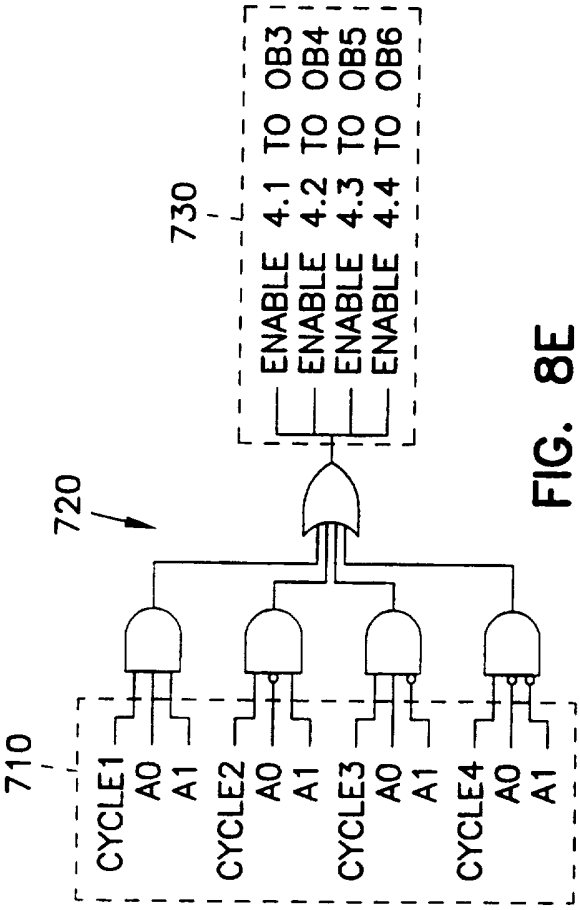

FIG. 8E

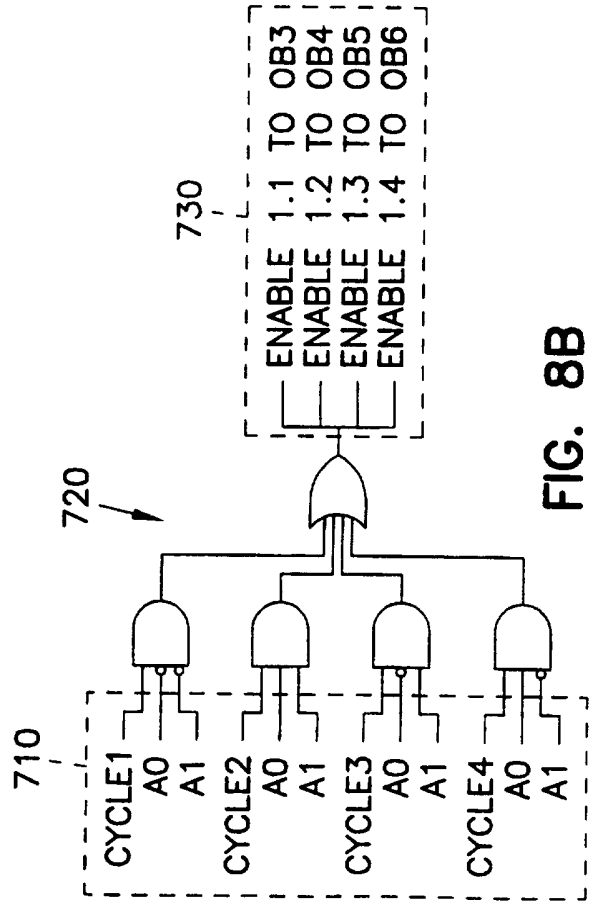

FIG. 8B

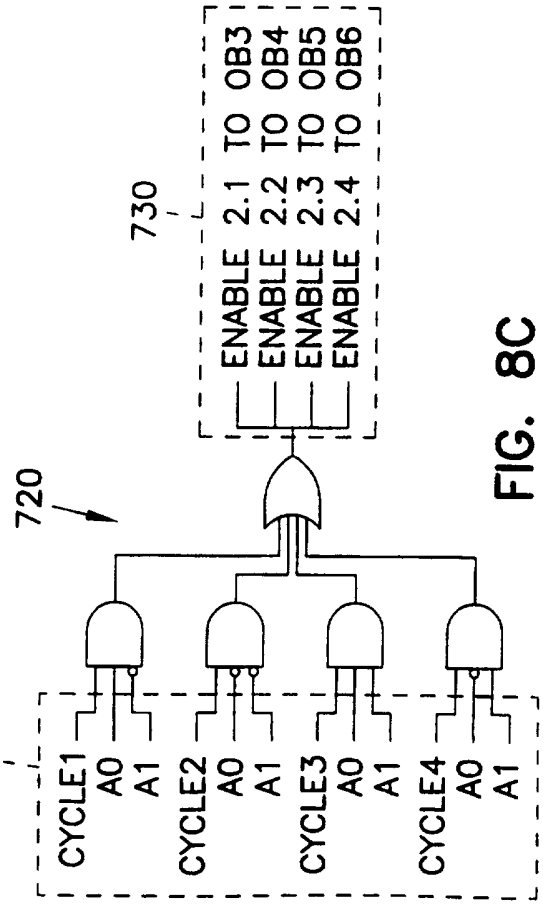

FIG. 8C

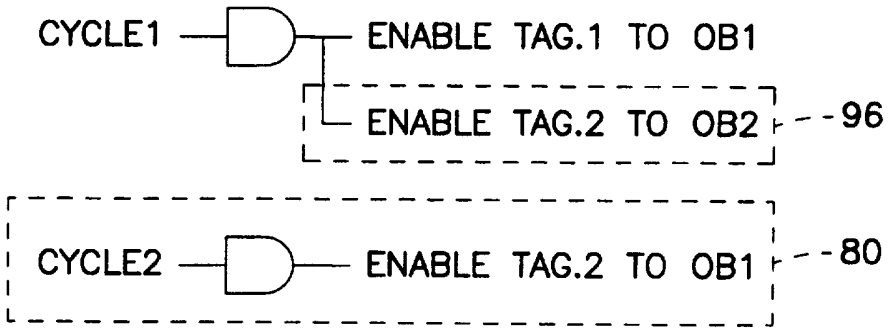

FIG. 9A

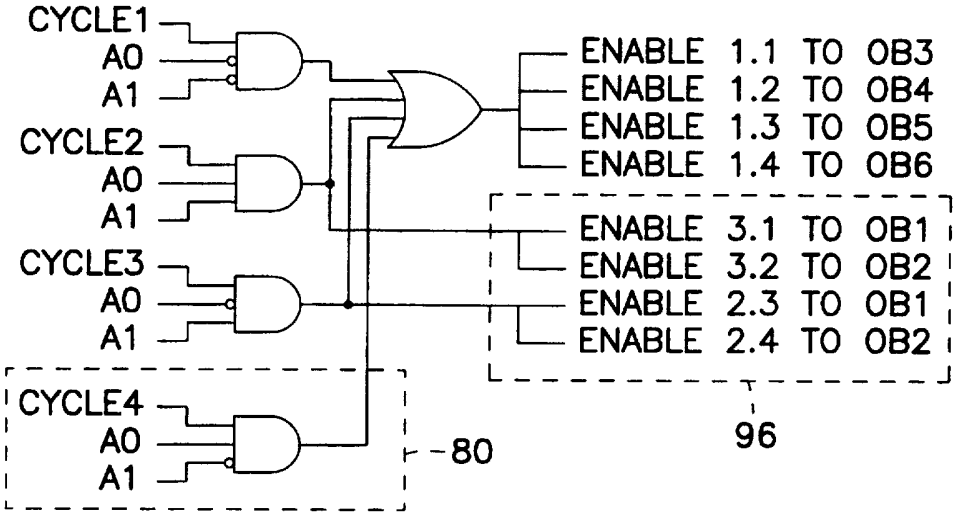

FIG. 9B

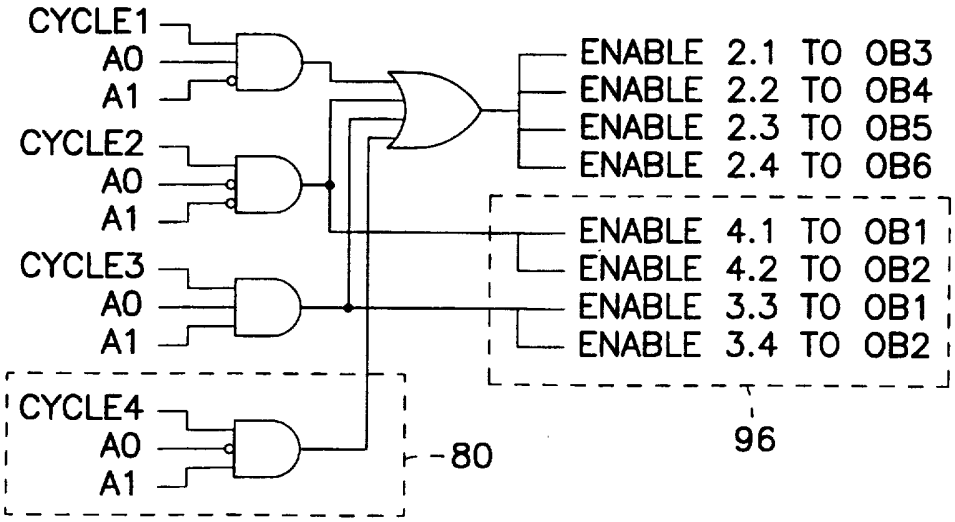

FIG. 9C

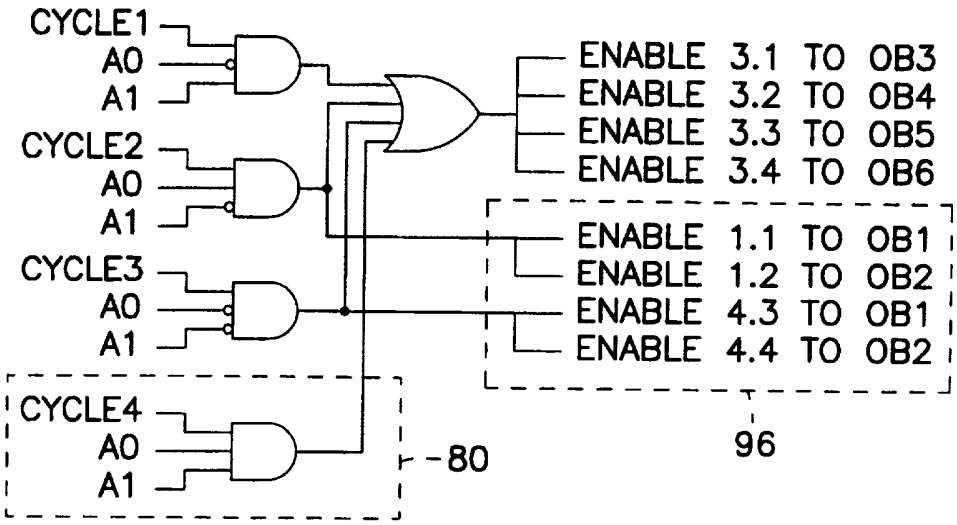
CYCLE1
A0
A1
CYCLE2
A0
A1
CYCLE3
A0
A1
CYCLE4
A0
A1
80
ENABLE 3.1 TO OB3
ENABLE 3.2 TO OB4
ENABLE 3.3 TO OB5
ENABLE 3.4 TO OB6
ENABLE 1.1 TO OB1
ENABLE 1.2 TO OB2
ENABLE 4.3 TO OB1
ENABLE 4.4 TO OB2
96

FIG. 9D

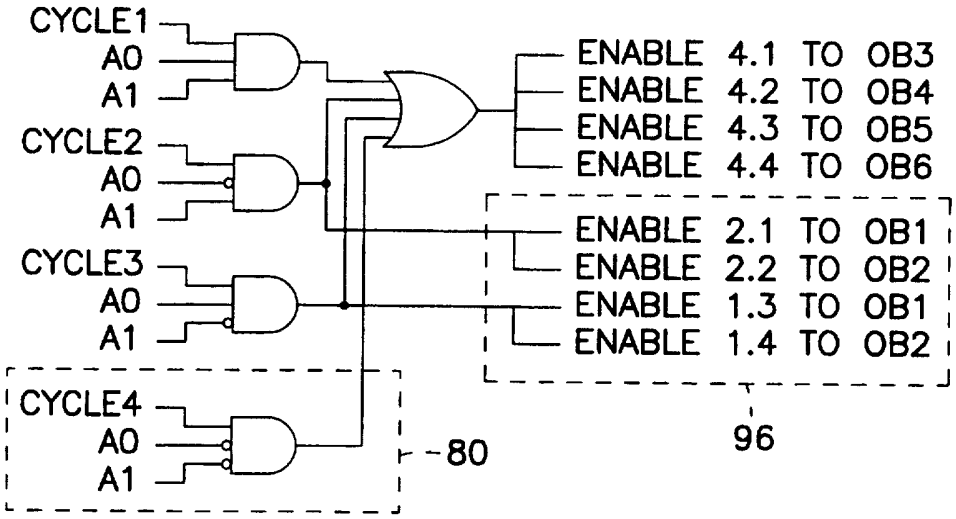
CYCLE1
A0
A1
CYCLE2
A0
A1
CYCLE3
A0
A1
CYCLE4
A0
A1
80
ENABLE 4.1 TO OB3
ENABLE 4.2 TO OB4
ENABLE 4.3 TO OB5
ENABLE 4.4 TO OB6
ENABLE 2.1 TO OB1
ENABLE 2.2 TO OB2
ENABLE 1.3 TO OB1
ENABLE 1.4 TO OB2
96

FIG. 9E

INTERNATIONAL SEARCH REPORT

Inter...onal Application No
PCT/US 97/10184

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F12/08

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)
IPC 6 G06F G11C

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

Electronic data base consulted during the international search (name of data base and, where practical, 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.
X	US 5 406 525 A (NICHOLES JAMES W) 11 April 1995 see abstract	1
A	see column 3, line 58 - column 4, line 49; figures 2,11	2-7, 12-14

X	US 5 280 598 A (OSAKI AKITOSHI ET AL) 18 January 1994	1-3,12
Y	see column 3, line 40 - column 4, line 20; figures 7-9	20,21

Y	"FAST TTL BURST CONTROLLER FOR MICROPROCESSOR" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 33, no. 8, 1 January 1991, pages 118-120, XP000107015 see the whole document	20,21

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
23 September 1997	08. 10. 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Nielsen, O

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter. ..onal Application No
PCT/US 97/10184

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5406525 A	11-04-95	NONE	
US 5280598 A	18-01-94	JP 4084253 A	17-03-92