Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2434031 C 2007/06/19

(11)(21) 2 434 031

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2001/12/10
(87) Date publication PCT/PCT Publication Date: 2002/07/18
(45) Date de délivrance/Issue Date: 2007/06/19
(85) Entrée phase nationale/National Entry: 2003/07/08
(86) N° demande PCT/PCT Application No.: US 2001/047743
(87) N° publication PCT/PCT Publication No.: 2002/056180
(30) Priorités/Priorities: 2001/01/09 (US09/757,760); 2001/07/27 (US09/917,304)

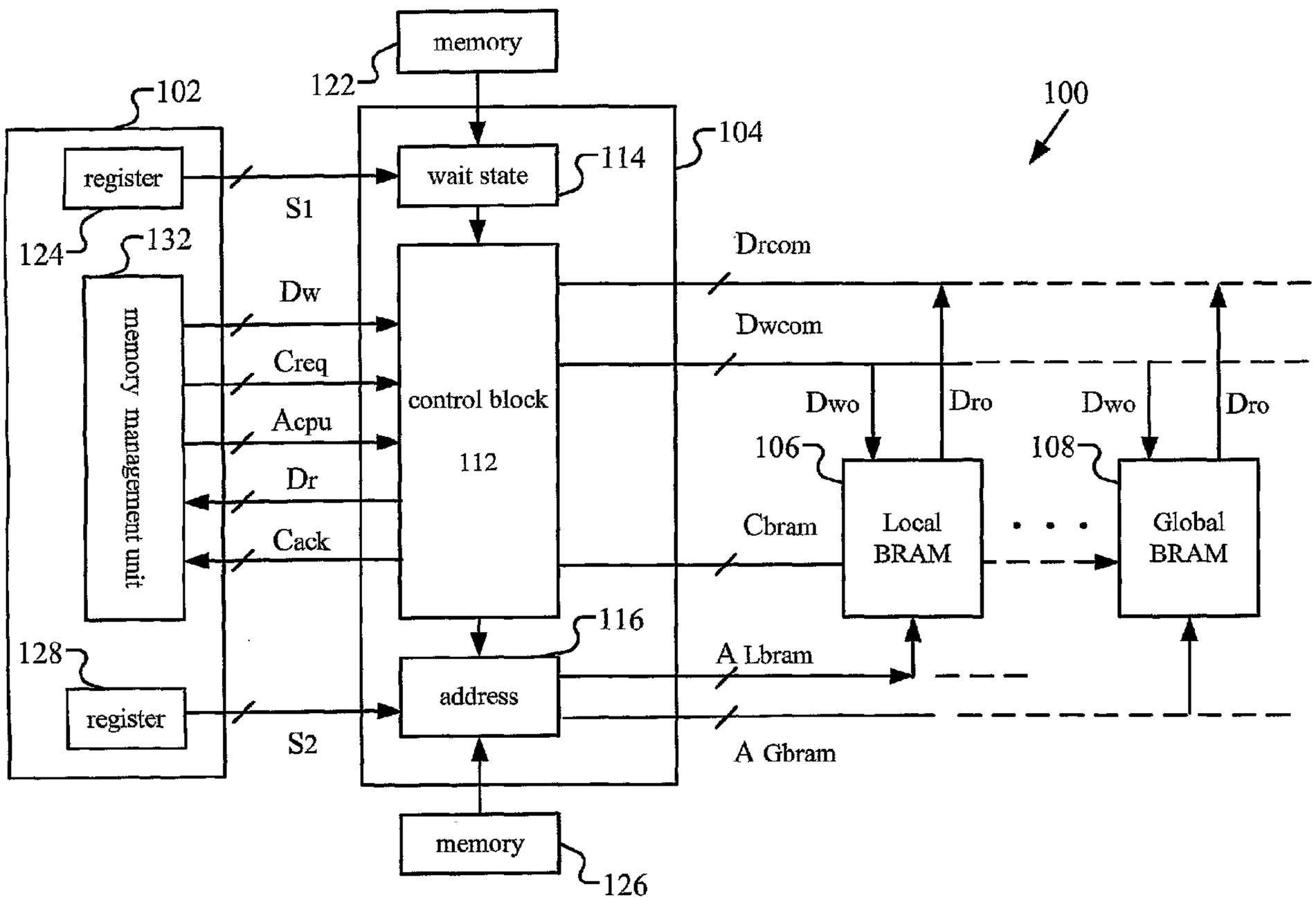
(51) Cl.Int./Int.Cl. *G06F 12/02* (2006.01), *G06F 13/16* (2006.01), *G06F 15/78* (2006.01)

(72) Inventeurs/Inventors:
ANSARI, AHMAD R., US;
DOUGLASS, STEPHEN M., US;
VASHI, MEHUL R., US;
YOUNG, STEVEN P., US;
SASTRY, PRASAD L., US;
YIN, ROBERT, US

(73) Propriétaire/Owner:
XILINX, INC., US

(74) Agent: SMART & BIGGAR

(54) Titre : SYSTEME MEMOIRE SUR PUCE CONFIGURABLE PAR L'UTILISATEUR
(54) Title: USER CONFIGURABLE ON-CHIP MEMORY SYSTEM



(57) Abrégé/Abstract:
A data processing system having a user configurable memory controller, local block RAMs, global block RAMs and a processor core can be configured in a single field programmable gate array (FPGA). The address depth of the global block RAMs and the number of wait states can be selected by a user, and they can be set either prior to configuration of the FPGA or programmed using instructions of the processor core. The number of wait states of the local block RAM is also user selectable. An algorithm that can optimize the address depth and the number of wait states to achieve a performance level is also disclosed. The present invention can be applied to designs having separate instruction and data sides.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 July 2002 (18.07.2002)

PCT

(10) International Publication Number
WO 02/056180 A2

(51) International Patent Classification[7]: G06F 12/00

(21) International Application Number: PCT/US01/47743

(22) International Filing Date:
10 December 2001 (10.12.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/757,760	9 January 2001 (09.01.2001)	US
09/917,304	27 July 2001 (27.07.2001)	US

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

(72) Inventors: **ANSARI, Ahmad, R.**; 4322 Montmorency Court, San Jose, CA 95118 (US). **DOUGLASS, Stephen, M.**; 13754 Orleans Court, Saratoga, CA 95070 (US). **VASHI, Mehul, R.**; 6588 Rolling Oaks Court, San Jose, CA 95120 (US). **YOUNG, Steven, P.**; 1180 Pintail Circle, Boulder, CO 80303 (US). **SASTRY, Prasad, L.**; 1064 Ridgemont Drive, Milpitas, CA 95035 (US). **YIN, Robert**; 16768 Columbia Drive, Castro Valley, CA 94552 (US).

(74) Agents: **CHANROO, Keith, A.** et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).

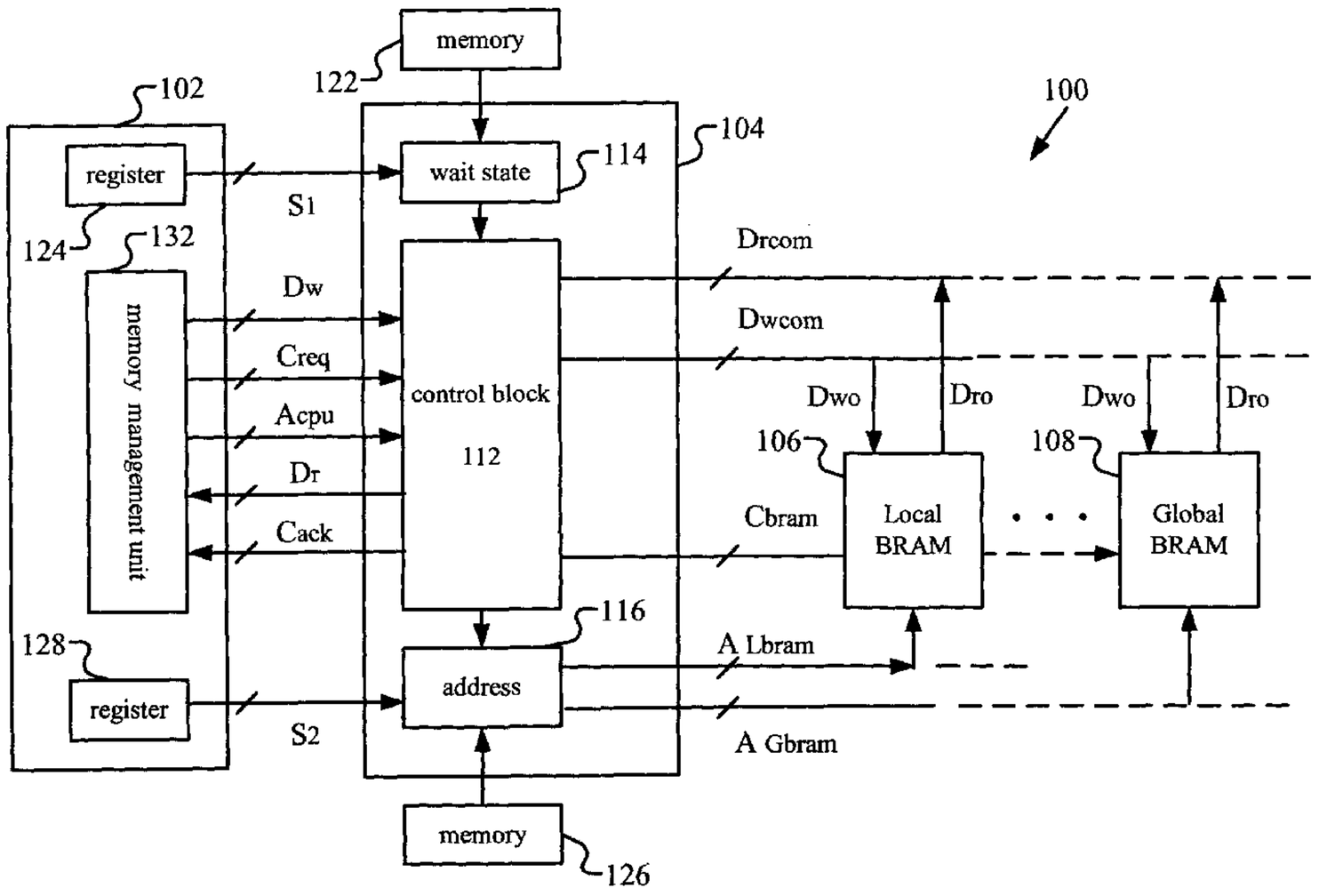
(81) Designated States *(national)*: CA, JP.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: USER CONFIGURABLE ON-CHIP MEMORY SYSTEM



(57) Abstract: A data processing system having a user configurable memory controller, local block RAMs, global block RAMs and a processor core can be configured in a single field programmable gate array (FPGA). The address depth of the global block RAMs and the number of wait states can be selected by a user, and they can be set either prior to configuration of the FPGA or programmed using instructions of the processor core. The number of wait states of the local block RAM is also user selectable. An algorithm that can optimize the address depth and the number of wait states to achieve a performance level is also disclosed. The present invention can be applied to designs having separate instruction and data sides.

WO 02/056180 A2

USER CONFIGURABLE ON-CHIP MEMORY SYSTEM

FIELD OF THE INVENTION

The present invention generally relates to memory system design, and more particularly to a memory system that can be configured by users to optimize the size and performance of the memory system.

BACKGROUND OF THE INVENTION

Programmable integrated circuits (ICs) are a well-known type of integrated circuit that may be programmed by a user to perform specified logic functions. One type of programmable IC, the field programmable gate array (FPGA), is very popular because of a superior combination of capacity, flexibility and cost. A FPGA typically includes an array of configurable logic blocks (CLBs) surrounded by a ring of programmable input/output blocks (IOBs). The CLBs and IOBs are interconnected by a programmable interconnect structure. The CLBs, IOBs, and interconnect structure are typically programmed by loading a stream of configuration data (bitstream) into internal configuration memory cells that define how the CLBs, IOBs, and interconnect structure are configured. The configuration bitstream may be read from an external memory (e.g., an external PROM). The collective states of the individual memory cells then determine the function of the FPGA.

As processing technology improves, more and more CLBs, IOBs and interconnect structures can be fabricated inside a FPGA. Recently, it is possible to build an entire data processing system (containing a central processor unit, memory, and various controllers) inside a FPGA. In some cases, not all the CLBs, IOBs and interconnect structures in the FPGA are used for building the data processing system, and some of them can be used for other applications.

One of the most important resources in a data processing system is memory. Many FPGAs provide blocks of

random access memories (RAMs) each has thousands of memory cells (called "block RAMs"). These blocks can be organized into different configurations. As example, a block RAM may have a capacity of 16 Kilobits. This block RAM may be arranged by a user to have an address depth of either 16K, 8K, 4K, 2K, 1K and 0.5K, with the corresponding number of bits per address as 1, 2, 4, 8, 16 or 32, respectively. A user can also combine a number of blocks to increase the total size of a memory system. More information about block RAMs can be found in U.S. Pat. No. 5,933,023 entitled "FPGA Architecture Having RAM Blocks with Programmable Word Length and Width and Dedicated Address and Data Lines," assigned to Xilinx, Inc.

In general, it is desirable to allow a data processing system to have access to as much memory as possible. One of the reasons is that some software modules require a minimum amount of memory to run. Another reason is that it is sometimes possible to speed up computation by allocating more memory to a task. On the other hand, a large amount of memory requires a large number of block RAMs. With the addition of each block RAM, the memory data access time of the memory is lengthened. One way to solve this problem is to introduce delays between a request for memory access and the granting of the access. In other words, "wait states" need to be inserted. As a result, the performance of the data processing system at the memory interface is reduced.

Another problem reserving a large amount of memory for the data processing system is that the total amount of block RAMs in a FPGA is limited. In addition to the data processing system, other logic modules in the FPGA may need to have more memory. If all or most of the block RAMs are allocated to the data processing system, it may compromise the design of other logic modules.

The optimal amount of memory and number of wait states vary with different designs. For example, real-time applications tend to require fast execution because the data processing system has to complete computations within

a short period of time. Thus, it is desirable to eliminate wait states. On the other hand, it may be advantageous to enable a general purpose design to run many software applications. Thus, it would be advantageous to include more memory in the data processing system. In order to give users the most design flexibility, it is desirable to allow the users to configure the memory system to achieve an optimal performance.

SUMMARY OF THE INVENTION

In one aspect of the present invention, there is provided a field programmable gate array (FPGA) comprising: a configuration memory for storing configuration bits for controlling at least a portion of the FPGA; one or more block RAMs; a processor, said processor generating a first set of address lines; and a user programmable memory controller interfacing with said processor and controlling said one or more block RAMs, said controller comprising: a user programmable address manager generating a second set of address lines from said first set of address lines, said second set of address lines having a programmable depth and being coupled to said one or more blocks of RAM; and a user-programmable wait state manager that generates wait states as a function of said depth; such that the processor allocates the necessary amount of block RAM to data processing to optimize the use of the block RAMs.

In another aspect of the present invention, there is provided an on-chip data processing system comprising a user configurable on-chip memory system and an on-chip processor core. The memory system comprises at least a memory controller, block RAMs, and storage of design values related to the memory system. The number of block RAMs and

the number of address lines (i.e., address depth) associated with the block RAMs can be selected and configured by users. One advantage of this invention is that only the necessary amount of block RAMs used by the processor core is allocated to the data processing system. All the block RAMs that are not allocated can be used by other on-chip applications. As a result, it optimizes the use of a valuable resource: block RAMs.

One embodiment of the memory controller contains an address manager that can deactivate some of the address lines originated from the processor core. The number of deactivated address lines is user configurable. The deactivation may be accomplished by a combination of demultiplexers, multiplexers and memory cells that store user supplied information.

Users can apply the memory controller of the present invention to set up the number of wait states of the memory system. In order to make sure that the memory system functions properly, the number of wait states needs to be chosen so that block RAMs have time to respond to a request. The present invention also involves an algorithm

that allows users to select the optimal combination of wait states and associated address depth.

The number of wait states and/or the number of address lines may be set prior to configuration of a FPGA. In another embodiment, one or both of these two parameters may be set by programming instructions of the processor core. The memory system of the present invention may also be applied to a data processing system having separate instruction and data sides. In this system, an instruction memory controller is associated with block RAMs used for storing instructions and a data memory controller is associated with block RAMs used for storing data. In one embodiment, the instruction and data block RAMs can be physically the same. In this case, it may be desirable to use memory management unit (MMU) schemes in general, for memory protection.

The data processing system may have two types of block RAMs, local and global. Local block RAMs have direct connection to the processor core while the global block RAMs are connected to the processor core through the interconnect structure of the programmable logic device. As a result, the delays in accessing the local block RAMs is much less than that of the global block RAMs. Thus, the number of wait states of the local block RAMs are smaller than that of the global block RAMs.

The above summary of the present invention is not intended to describe each disclosed embodiment of the present invention. The figures and detailed description that follow provide additional example embodiments and aspects of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the detailed description and the following figures, in which like reference numerals refer to similar elements.

FIG. 1A is a block diagram showing a FPGA system of the present invention.

FIG. 1B is a block diagram of a portion of the FPGA system of FIG. 1A.

FIG. 2 is a schematic diagram showing a data processing system of the present invention that is implemented on a FPGA.

FIGS. 3A and 3B show a block diagram of a user configurable on-chip memory controller of the present invention.

FIG. 4 is a flow chart of an algorithm of the present invention to select sizes of block RAMs and associated wait states of the on-chip memory controller of the present invention.

FIG. 5 is a flow chart showing the use of the result of flow chart in FIG. 4 to construct a data processing system of the present invention.

FIG. 6 is a flow chart showing the use of a processor core to configure the memory controller of the present invention.

FIG. 7 is a schematic diagram of a data processing system of the present invention having separate instruction and data sides.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to constructing a data processing system using a programmable IC. In the following description, numerous specific details are set forth in order to provide a more thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail in order to avoid obscuring the present invention.

FIG. 1A is a block diagram showing a system containing a FPGA 20 and an associated external memory 12. FPGA 20 comprises a configuration memory 24 consisting of a plurality of bits. Each configuration bit defines the state of a static memory cell that controls a portion of the FPGA, such as a function lookup table bit, a

multiplexer input, or an interconnect pass transistor. Configuration is the process of loading design-specific data to define the functional operation of the internal blocks and their interconnection, by storing these values into configuration memory. FPGA 20 further comprises a first section 26 that can be configured, using bits in configuration memory 24, as a data processing system of the present invention. The data processing system contains a processor core, a configurable memory controller, block RAMs, and other auxiliary components. FPGA 20 optionally contains a second section 28 that can be configured, using bits in configuration memory 24, to perform other logic or memory functions. An example of a FPGA is the Virtex™ FPGA from Xilinx, Inc., assignee of the present invention. The Virtex™ FPGA is described in the Xilinx data books ("The Programmable Logic Data Book 1999" and "The Programmable Logic Data Book 2000"), Chapter 3. Portions of the Virtex architecture are described by Young et. al in U.S. Patent 5,914,616 issued June 22, 1999. Note that more advanced and higher capacity FPGAs may be used in the present invention.

When FPGA 20 is powered up, it receives configuration bits from external memory 12. The bits in configuration memory 24 are set accordingly. A user may change the configuration of FPGA 20 by changing the data in external memory 12. As explained below, the present invention can be used to optimize the design of the data processing system in FPGA 20.

One example of first portion 26 is shown in FIG. 1B. It contains a processor core and memory controller 42 connected to a plurality of local block RAMs (such as block RAMs 44-47) and a plurality of global block RAMs (such as block RAMs 52-57). Global block RAMs 52-57 are typically positioned far away from core and controller 42, and may be connected thereto through long and slow FPGA connection wires. As a result, the delay is relatively long. On the other hand, local block RAMs 44-47 are typically positioned

close to core and controller 42. These block RAMs may be connected directly (i.e., without the use of conventional FPGA fabric) to core and controller 42 through connections 62a-62d. As a result, the delay is relatively short. Alternatively, local block RAMs 44-47 may also be connected to the regular FPGA connection wires (such as wires 64a and 64b). One aspect of the present invention takes into account of this new architecture. Note that the geographic orientation of the various block RAMs relative to core and controller 42 in FIG. 1B is shown for illustrative purpose only. Other orientation may be used.

The data bus width and address depth of the global block RAMs can be selected by users. In one embodiment, the data bus width and address depth of the local block RAMs are predetermined. However, it is possible that these parameters for local block RAMs are also user selectable.

FIG. 2 is a schematic diagram showing a data processing system 100 of the present invention that corresponds to the structure in FIG. 1B. System 100 comprises a processor core 102, a configurable memory controller 104, and block RAMs (such as a local BRAMs 106 and a global BRAM 108) controlled by memory controller 104. Controller 104 comprises (a) a control block 112 that interfaces with processor core 102, BRAM 106 and BRAM 108, (b) a wait state manager 114 that directs control block 112 to generate wait states in accordance with user requirements, and (c) an address manager 116 that sets up the address depth of the block RAMs in accordance with user requirements.

Wait state manager 114 accepts inputs from a register set 124 inside processor core 102 (via a bus labeled S_1) and a memory 122 in the FPGA. Data in memory 122 and register set 124 is determined by user requirements, as explained in more detail below. It should be noted that memory 122 may comprise one or more memory cells, depending on the size and number of data that need to be stored in memory 122. Similarly, register set 124 may store one or more pieces of data. This is because the local and global block RAMs may

need different wait states. Wait state manager 114 is connected to control block 112, and directs block 112 to generate wait states corresponding to the data in register set 124 or memory 122. Detailed structures of wait state manager 114 and control block 112 will be described below. Because the number of wait states affects the performance of data processing system 100, this is a factor that needs to be considered in the design of memory controller 104.

Address manager 116 accepts inputs from a memory 126 of the FPGA and a register 128 inside processor core 102 (via a bus labeled S_2). Data in memory 126 and register 128 is determined by user requirements, as explained in more detail below. It should be noted that memory 122 may comprise one or more memory cells, depending on the maximum number of address lines in the memory system. Address manager 116 accepts address lines from control block 112 and selects all or some of the address lines for the purpose of addressing the global block RAMs (shown as A_{Gbram}). This feature allows a user to select the optimal size and address depth of the global block RAMs for use by processor core 102. The remaining global block RAMs can be used for other purposes in the FPGA. Address manager 116 also generates a different address bus (shown as A_{Lbram}) for the local block RAMs. This address bus allows the local block RAMs to be accessed with minimal delays.

Control block 112 preferably interfaces with a memory management unit 132 of processor core 102. Memory management unit 132 sends out a read address for data load and instruction fetch operations. It also sends out a write address for data to be written into memory. Memory management unit 132 also generates and receives appropriate control signals from control block 112. For example, block 112 accepts a write data bus D_w, a control signal bus C_{req} and an address bus A_{cpu} from memory management unit 132. It drives a read data bus D_r and a control signal bus C_{ack} to memory management unit 132 of processor core 102. Control block 112 also delivers data to a write data bus D_{wocm} and controls signals to a control signal bus C_{bram} of block RAMs

106 and 108. Control block 112 receives a read data bus D_{rocm} from block RAMs 106 and 108.

It should be noted that even though Fig. 2 shows that a single write data bus is connected to both local and global block RAMs, it is possible to design different write data buses for local and though Fig. 2 shows that a single read data bus is connected to both local and global block RAMs, it is possible to design different read data buses for local and global block RAMs.

Additional details of user configurable memory controller 104 are shown in FIGS. 3A and 3B. FIG. 3A is logically divided into wait state manager 114 and control block 112 while FIG. 3B shows address manager 116. These divisions are similar to the corresponding divisions in FIG. 2. Wait state manager 114 comprises a multiplexer 152 and a wait state register set 154. In the present invention, local block RAMs and global block RAMs may have different wait states. Thus, wait state register set 154 may contain more than one register. Wait state register set 154 is used to store one or more values representing the number of wait states in memory access. This information is fed to a state machine 160 inside control block 112, which generates the wait states accordingly. Multiplexer 152 accepts a select signal 156 that selects one of the two inputs to couple to wait state register set 154 (i.e., either from register set 124 or memory 122). Because data in register set 124 and memory 122 is determined by a user, the number of wait states is user configurable. Signal 156 is controlled by FPGA configuration logic. In one embodiment, the data from memory 122 is loaded into wait state register set 154 during FPGA power up. After the FPGA is configured, signal 156 may relinquish control of wait state register set 154 to register set 124. In this way, users can overwrite previously loaded information in wait state register set 154 if there is a need to do so.

One aspect of the present invention is that the number of wait states can be set either via hardware or software. In one embodiment, memory 122 is designed to be loaded prior to FPGA configuration and the data therein used to configure memory controller 104. This is a hardware method of setting the wait states. After configuration, the number of wait states cannot be changed by hardware. On the other hand, register set 124 can be programmed via instructions of processor core 102. Thus, this is a software method of setting the wait states. The values of register set 124, and thus the number of wait states, may be changed repeatedly anytime after configuration.

Control block 112 comprises an address selector 162. It accepts the address bus A_{cpu} (from memory management unit 132) and selects some of the address lines to form a new bus A_{cpu}* for coupling to address manager 116. The number of lines in bus A_{cpu}* can be any number between 1 and the number of address lines in bus A_{cpu}. In one embodiment of the present invention, A_{cpu} has 30 lines and A_{cpu}* has 16 lines. It should be easy for persons skilled in the art to design address selector 162. Note that the number of lines in bus A_{cpu}* is not user configurable because the design of address selector 162 is predetermined. Further note that if the number of lines in buses A_{cpu} and A_{cpu}* is the same, address selector 162 may be omitted in control block 112. As explained in more detail below, the number of address lines may be further reduced by address manager 116. This reduction is user configurable.

Control block 112 may comprise a temporary memory 164 for temporarily storing data received from bus D_w prior to sending the same to bus D_{wocm}. If there is no need to service read operations during write operations, temporary memory 164 may not be needed. Control block 112 may also comprise a temporary memory 166 for temporarily storing data received from a block RAM through bus D_{rocm} prior to sending the same to bus D_r. A block RAM control block 168 is used to generate signals in control signal bus C_{bram}. As an example, it may comprise logic to enable reading/writing

of the block RAMs. The design of block RAM control block 168 depends on the specification of the block RAMs used, and should be easy for persons skilled in the art to do so.

State machine 160 is used to synchronize various signals and activities of address selector 162, temporary memory 164, temporary memory 166 and block RAM control block 168. As an example, it accepts control signal bus C_{req} (requesting read, write or abort operations) and generates control signal bus C_{ack} regarding the status of data transfer (read, write acknowledge or abort). The design of state machine 160 depends on the specification of processor core 102 and the block RAMs, together with the value in wait state register set 154. It should be easy for persons skilled in the art to design state machine 160 using these information.

Address manager 116 accepts bus A_{cpu}* and generates a local block RAM address bus A_{Lbram} and a global block RAM address bus A_{Gbram}. In one embodiment, the number of address lines in local block RAM address bus A_{Lbram} is predetermined, and is preferably less than the number of address lines in bus A_{cpu}*. On the other hand, the number of address lines in global block RAM bus A_{Gbram} is determined by the data in either memory 126 or register 128, and is equal to or less than the number of address lines in bus A_{cpu}*.

A portion 170 of address manager 116 that handles global address line generation is now described. It comprises a plurality of address line deactivation units, one of which is shown as unit 171 in Fig. 3. Each unit may deactivate an address line of bus A_{cpu}*. Unit 171 comprise a demultiplexer 172 that accepts as input an address line of bus A_{cpu}*. One of the two outputs of demultiplexer 172 (e.g., output 180) is a single address line of the global block RAM address bus A_{Gbram}. The other output of demultiplexer 172 (e.g., output 178) is not used. If the input of demultiplexer 172 is connected to output 180, the address line is activated. On the other hand, if the input of demultiplexer 172 is connected to output 178, the address line is deactivated. Whether output 180 is deactivated is controlled by another

multiplexer 174. One input of multiplexer 174 is a bit of memory 126 and another input is a bit of register 128. A select signal 176 is used to select whether the bit of memory 126 or register 128 is coupled to demultiplexer 174. As a result, whether an address line of bus A_{cpu}* is deactivated is controlled by either a bit of memory 126 or register 128. By using a plurality of address line deactivation units, it is possible to deactivate some of the address lines of bus A_{cpu}*, thereby reducing the number of address lines of the global block RAM address bus A_{Gbram}. Signal 176 is controlled by FPGA configuration logic. In one embodiment, the data from memory 126 is loaded into demultiplexer 172 select line during FPGA power up. After the FPGA is configured, signal 176 will relinquish control so that register 128 can affect the deactivation. The demultiplexer block is used as a generic logic representation only. It should be noted that, any combination of logic gates can be used to achieve the same result.

It can be seen from the above that the bits of either memory 126 or register 128 can be used to determine the number of address lines in bus A_{Gbram}. One aspect of the present invention is that the number of address lines of the block RAM address bus can be set either via hardware or software. In one embodiment, memory 126 is designed to be load prior to FPGA configuration and the data therein used to configure address manager 116. This is a hardware method of setting the number of address lines. On the other hand, register 128 can be programmed via instructions of processor core 102. Thus, this is a software method of setting the number of address lines.

A portion 190 of address manager 116 that handles the local address lines is now described. It comprises a plurality of multiplexers (such as multiplexer 186), one for each address line. One input of multiplexer 186 accepts one of the predetermined address lines of bus A_{cpu}* and the other input accepts one of the address lines of the global block RAM address bus A_{Gbram}. A control signal is applied to a

line 188 to select a desired address line. The output of multiplexer 186 is one of the address lines of bus A_{Lbram}. This arrangement allows the local block RAMs to be accessible by direct connection (e.g., through connections 62a-62d) or regular FPGA connection wires (e.g., through wires 64a-64d).

Control block 112 accepts a clock signal (Clock). This signal is used to synchronize the timing of wait state register set 154, state machine 160 and address selector 162.

In order to allow a user of a FPGA to more efficiently design a data processing system of the present invention, an algorithm that can automate some of the design considerations relating to the global block RAMs is disclosed. Fig. 4 shows a flow chart 200 of such an algorithm. In step 202, a user determines the amount of global block RAMs used by the data processing system based on his/her design criteria (designated in the formulas below by the symbol "S"). The user also enters the processor core data bus width ("D_u") and the maximum size of a global block RAM in the FPGA ("K"). The parameter D_u depends on the design of the processor core and the parameter K depends on the architecture of the FPGA. These serve as the inputs to flow chart 200. In step 204, the minimum number ("N") of global block RAMs needed to meet the requirement is determined using the following formula:

$$N = S/K.$$

In step 206, the address depth ("AD_o") and data bus width ("D_o") of the global block RAM are calculated. This provides the optimal aspect ratio of the global block RAM. The address depth is given by the following formula:

$$AD_o = S/N;$$

and the data bus width for the global block RAMs is determined by the following formula:

$$D_o = D_u/N.$$

Methods for configuration block RAMs to achieve a predetermined aspect ratio have been disclosed in the above mentioned U.S. Pat. No. 5,933,023.

An example is now provided to illustrate the above equations. It is assumed that the processor core data bus width (D_u) is 32 bits, the size of on-chip memory required (S) is 8 Kbytes, and the maximum size of a block RAM (K) is 16 Kilobits. Applying these numbers to the above formulas, one gets the following results:

$N = S/K = 4;$

$D_o = D_u/N = 8;$ and

$AD_o = S/N = 2$ Kbytes.

The number of address lines in the global block RAM address bus A_{Gbram} is given by $\log_2(AD_o) = 11$. The aspect ratio is 2K x 8. The total number of on-chip memory can be verified to be 4 x (2K x 8) = 8 Kbytes, which is the desired value.

Once the above calculation is completed, the values of the parameters that may affect the performance of the memory system of the present invention can be determined either by measuring the appropriate timings or calculating from specifications listed in appropriate data books (step 208). In one embodiment of the present invention, the performance is affected by:

the clock period of the processor core ("CPU_{clk}");

the address routing delay from memory controller 104 to the farthest block RAM in the FPGA ("T_{ra}");

the address setup time required for a block RAM (T_{sa})

the block RAM access time ("B_{acc}");

the data routing delay from the farthest block RAM in FPGA back to the processor core ("T_{rd}");

the data setup time required for the processor core ("T_{sd}"); and

the address delay through memory controller 104 in order to latch the address from the processor core ("T_d").

Using these parameters, it is possible to determine the number of wait states required for this choice of global block RAM and memory controller configuration (step 212). The number of wait states (W) is given by:

$$W = R\,[(T_{ra} + T_{sa} + B_{acc} + T_{rd} + T_{sd} + T_d)/\ CPU_{clk}]$$

where R stands for a rounding operation. The operation R takes the decimal result to the next higher integer so that W is an integer value and meets the performance requirement.

In step 214, the performance is evaluated by the user to see if it meets his/her requirements. If the answer is negative, the number of global block RAMs used to serve the processor core needs to be reduced (step 216). This is because the delays between the processor core and the farthest global block RAM increases with the number of block RAMs. Flow chart 200 then branches back to step 202 to re-evaluate the performance of a new set of design values. If the answer of step 214 is positive, the result is accepted and the algorithm terminates.

An example is provided to illustrate the above calculation. It is assumed that

CPU_{clk} = 5 ns (i.e., 200 MHz);

T_{ra} = 1 ns;

T_{sa} = 0.5 ns;

B_{acc} = 2.5 ns;

T_{rd} = 3 ns;

T_{sd} = 2.5 ns; and

T_d = 2.5 ns.

Using these numbers,

W = R[12 ns/5 ns] = R[2.4]

= 3 wait states.

As pointed out about, wait states may also need to be inserted to access the local block RAMs. The number of wait states can be calculated using the above-described formula for W. Using typical values for the local block RAMs, the number of wait states is typically much smaller than that of the global block RAMs.

After the design is completed, the result is used to configure a FPGA. FIG. 5 shows a flow chart 230 of the steps used in setting up the data processing system of the present invention. In step 232, the result of the algorithm 200 is integrated with other information (e.g.,

the structure of the processor core and section 28 of FIG. 1A) to generate configuration bits for a FPGA. The configuration bits are saved in an external memory. When the FPGA is powered up, the configuration bits in the external memory is loaded into the FPGA (step 234). In step 236, the FPGA bitstream is used to configure memory 126 (affecting the number of address lines of the block RAMs). In step 238, the FPGA bitstream is used to configure memory 122 (affecting the number of wait states). In step 240, the bitstream is used to configure the other parts of the FPGA. Flow chart 230 then terminates. Note that the orders of steps 236, 238 and 240 may be changed without affecting the present invention.

A method to use the processor core to configure memory controller 104 in a FPGA is now described using a flow chart 250 of FIG. 6. In step 252, the user decides whether it is desirable to use the processor core to set the number of wait states. If the answer is negative, a step 258 (described below) is performed. If the answer is positive, multiplexer 152 in wait state manager 114 is set to couple register set 124 of processor core 102 to wait state register set 154 (step 254). In step 256, appropriate value(s) is written into register set 124 using programming instructions of processor core 102. Flow chart 250 then branches to step 258.

In step 258, the user decides whether it is desirable to use the processor core to set the status of address manager 116. If the answer is negative, flow chart 250 terminates. If the answer is positive, multiplexer 174 in address manager 116 is set to couple register 128 of processor core 102 to multiplexer 172 (step 260). In step 262, an appropriate value is written into register 128 using the programming instructions of processor core 102. Flow chart 250 then terminates.

In another embodiment of the present invention, the block RAMs can be configured as a dual port memory. In this case, the contents of the block RAMs can be accessible

to processor core 102 and other parts of the FPGA (such as second section 28 in FIG. 1A).

The memory system of the present invention can be extended to a data processing system having separate instruction and data sides. In the present invention, a separate user configurable memory controller is used for each side, as shown in the data processing system 300 of FIG. 7. System 300 comprises a processor core 302 that has a separate data side 304 and instruction side 306. Data side 304 comprises a memory management unit 344, a register set 322 and a register 324. These registers serve similar functions as registers set 124 and 128, respectively, of FIG. 2. Memory management unit 344 further contains a translation look-aside buffer ("TLB") 348. Instruction side 306 comprises a memory management unit 346, a register set 326 and a register 328. These registers serve similar functions as register set 124 and register 128, respectively, of FIG. 2. Memory management unit 346 further contains a TLB 350.

Data processing system 300 comprises a data side memory controller 308 and an instruction side memory controller 310. The structures of these two controllers are substantially the same as that of memory controller 104, except that data side memory controller 308 contains a base address register and comparator 340 and instruction side memory controller 310 contains a base address register and comparator 342. These two base address registers and comparators are used for aliasing. Data side controller 308 is associated with two memories 332 and 334. These two memories serve similar functions as memory 122 and 126 of FIG. 2. Instruction side controller 310 is associated with two memories 336 and 338. These two memories serve similar functions as memory 122 and 126 of FIG. 2.

Data processing system 300 comprises a local block RAM 356 connected to data side controller 308 and a local block RAM 358 connected to instruction side controller 310. The wait states associated with block RAMs 356 and 358 aren substantially the same. Data processing system 300 also

contains a bank of global block RAMs 360. A portion 362 of it is used for data, another portion 364 is used for instruction, and the rest 366 may be used for other purposes. Portions 362, 364, and 366 may have different address depths and memory sizes.

Flow chart 200 can be used to calculate the performance of data side memory controller 308 and instruction side memory controller 310. Note that wait states of the data and instruction sides may be different because their corresponding memory controllers each contains its own wait state register set.

In one embodiment of the present invention, block RAM portions 362 and 364 can be physically the same (i.e., their base addresses are the same). This would not cause confusion if processor core 302 segregates instruction and data references to the instruction memory controller 310 and data memory controller 308, respectively. This design has the downside of not being able to provide separate protection for the data and instruction memory blocks. For example, because data memory 362 must be readable and writable, the instruction memory 364 address range is mapped as being writable. This is not generally desirable because writing to instruction address space cannot be detected.

One method to solve this problem is to treat the overlapping address as if they are at different addresses using aliasing. For example, with base addresses both set to 8'h00, the instruction memory could be treated as being at address 32'h00800000 and the data memory could be treated as being at address 32h'00000000. Base address registers and comparators 352 and 354 are used to perform such aliasing, The mechanism for loading these two components is identical to that for loading the wait state register, as described above. While the instruction and data memory controllers are still in the same address region, in contrast with the previous method, this allows the instruction and data memory controllers to use TLB mappings which give them separate protection. TLB

protection can ensure that any aliased references are detected. For example, mapping the instruction memory controller as execute-only and data memory controller as non-execute can prevent inadvertent instruction reference. It is important to note that in a scenario where separate TLB exist for each instruction and data, address separation is not needed.

It can be seen from the above description that a novel on-chip memory system and methods for implementing the same have been disclosed. Those having skill in the relevant arts of the invention will now perceive various modifications and additions which may be made as a result of the disclosure herein. Accordingly, all such modifications and additions are deemed to be within the scope of the invention, which is to be limited only by the appended claims and their equivalents.

74842-31

CLAIMS:

1. A field programmable gate array (FPGA) comprising:

a configuration memory for storing configuration bits for controlling at least a portion of the FPGA;

one or more block RAMs;

a processor, said processor generating a first set of address lines; and

a user programmable memory controller interfacing with said processor and controlling said one or more block RAMs, said controller comprising:

a user programmable address manager generating a second set of address lines from said first set of address lines, said second set of address lines having a programmable depth and being coupled to said one or more blocks of RAM; and

a user-programmable wait state manager that generates wait states as a function of said depth;

such that the processor allocates the necessary amount of block RAM to data processing to optimize the use of the block RAMs.

2. The FPGA of claim 1 wherein at least one of said one or more blocks of RAM has a configurable aspect ratio, and wherein said depth is a function of said aspect ratio.

3. The FPGA of claim 1 wherein said address manager comprises:

a set of control signals having user-programmable states for activating or deactivating said plurality of logic components.

4. The FPGA of claim 3 wherein at least one of said one or more blocks of RAM has a configurable aspect ratio, and wherein said states of said set of control signals is a function of said aspect ratio.

5. The FPGA of claim 3 wherein said plurality of logic components comprise demultiplexers.

6. The FPGA of claim 3 wherein said address manager further comprises a plurality of multiplexers that generate said set of control signals.

7. The FPGA of claim 6 wherein one input to each of said plurality of multiplexers is coupled to said configuration memory.

8. The FPGA of claim 6 wherein said processor is associated with a register, and wherein one input to each of said plurality of multiplexers is coupled to said register.

9. The FPGA of claim 8 wherein said register is programmable by software instructions of said processor.

10. The FPGA of claim 5 wherein said address manager further comprises a plurality of multiplexers each having an output terminal that delivers a respectively one of said set of control signals to a respective one of said demultiplexers.

SMART & BIGGAR

OTTAWA, CANADA

PATENT AGENTS

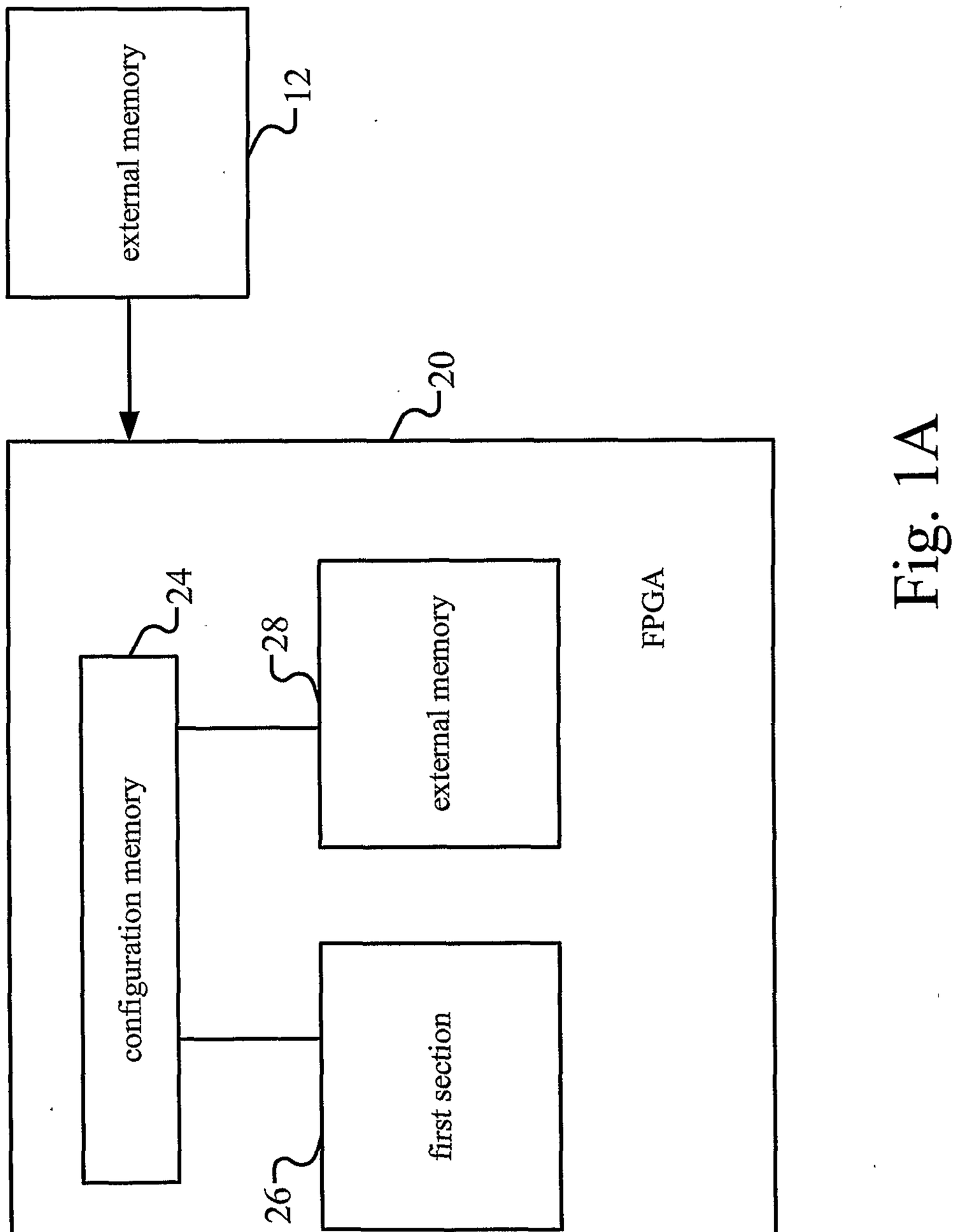

external memory
12
20
FPGA
configuration memory
24
28
external memory
first section
26

Fig. 1A

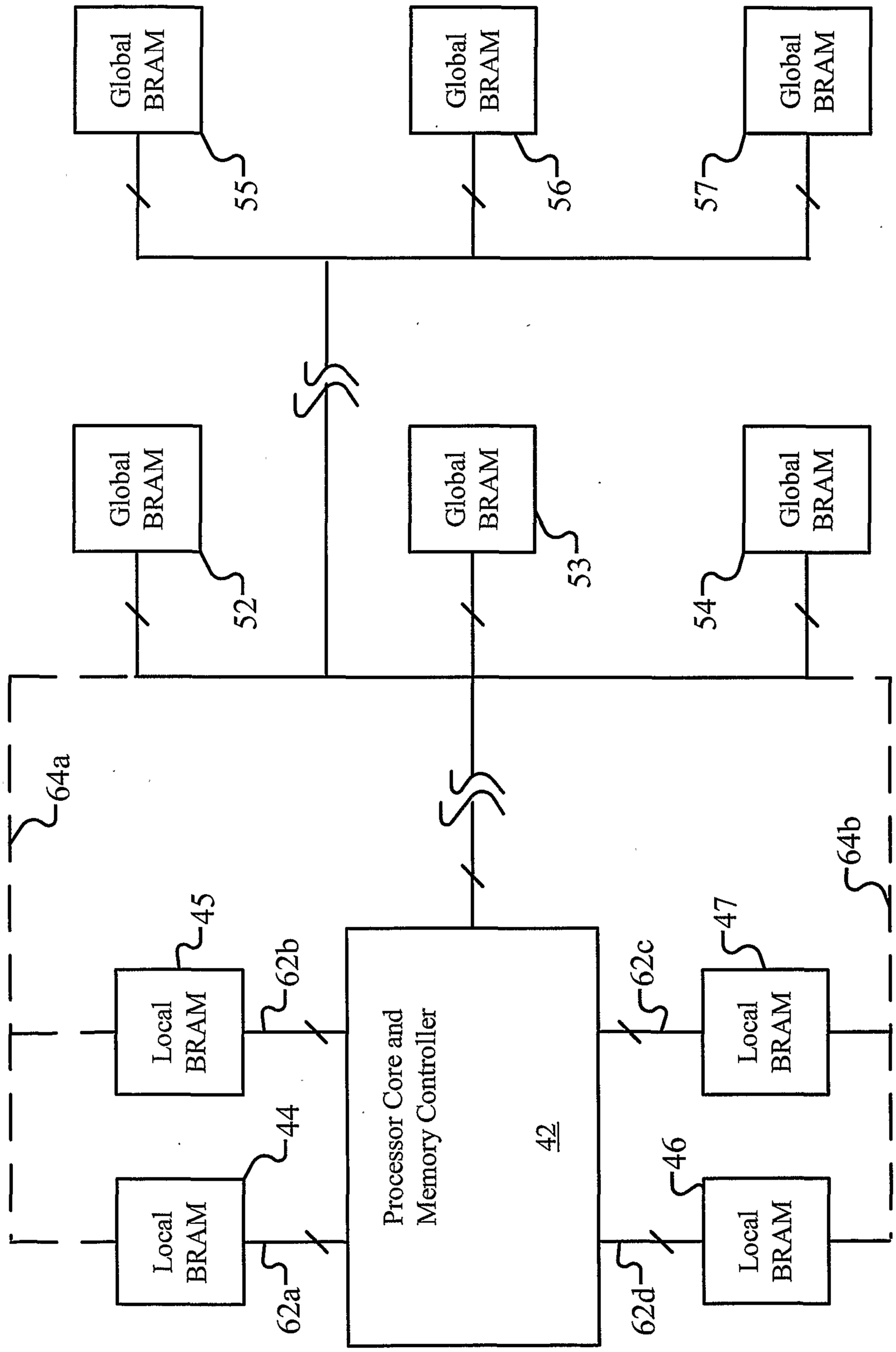

Global BRAM
55
Global BRAM
56
Global BRAM
57
Global BRAM
52
Global BRAM
53
Global BRAM
54
64a
Local BRAM
45
62b
Local BRAM
44
62a
Processor Core and Memory Controller
42
62c
Local BRAM
47
64b
Local BRAM
46
62d

Fig. 1B

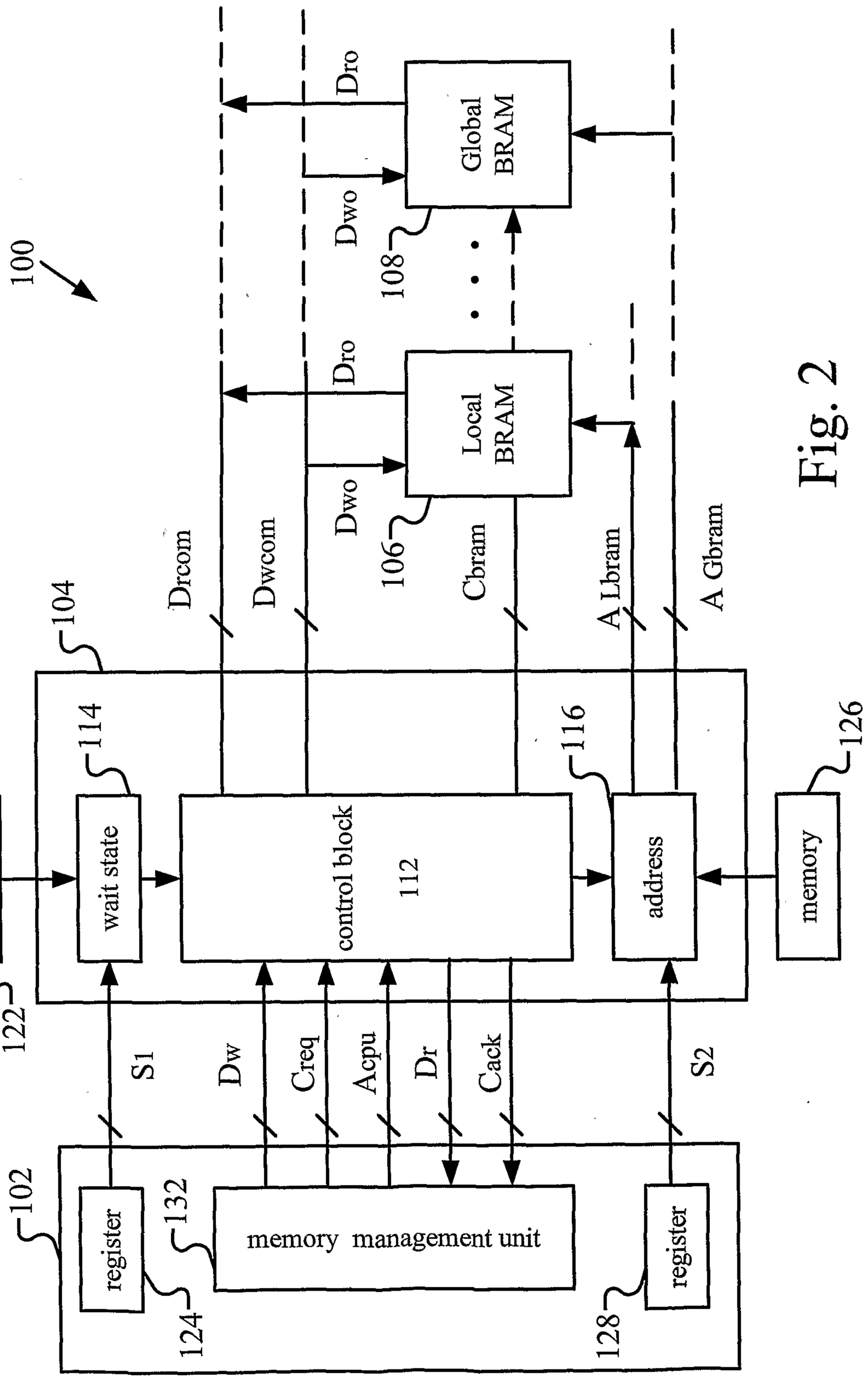

100
102
104
106
108
112
114
116
122
124
126
128
132
memory
wait state
control block
address
memory
register
memory management unit
register
Drcom
Dwcom
Dwo
Dro
Local BRAM
Global BRAM
Cbram
A Lbram
A Gbram
S1
Dw
Creq
Acpu
Dr
Cack
S2

Fig. 2

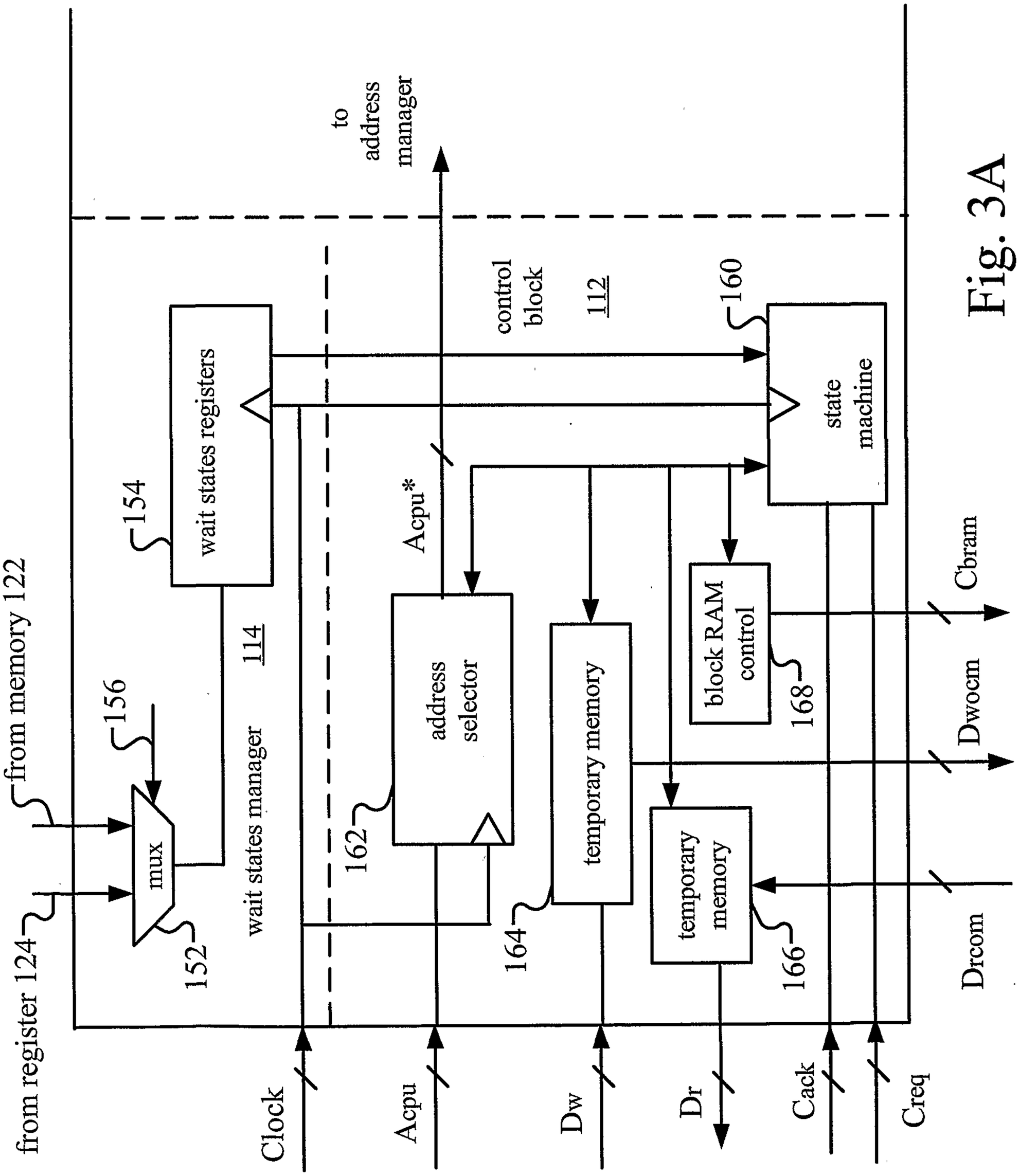

to address manager
control block
112
160
state machine
wait states registers
154
from memory 122
156
mux
152
from register 124
wait states manager
114
Acpu*
address selector
162
temporary memory
164
block RAM control
168
temporary memory
166
Cbram
Dwocm
Drcom
Clock
Acpu
Dw
Dr
Cack
Creq

Fig. 3A

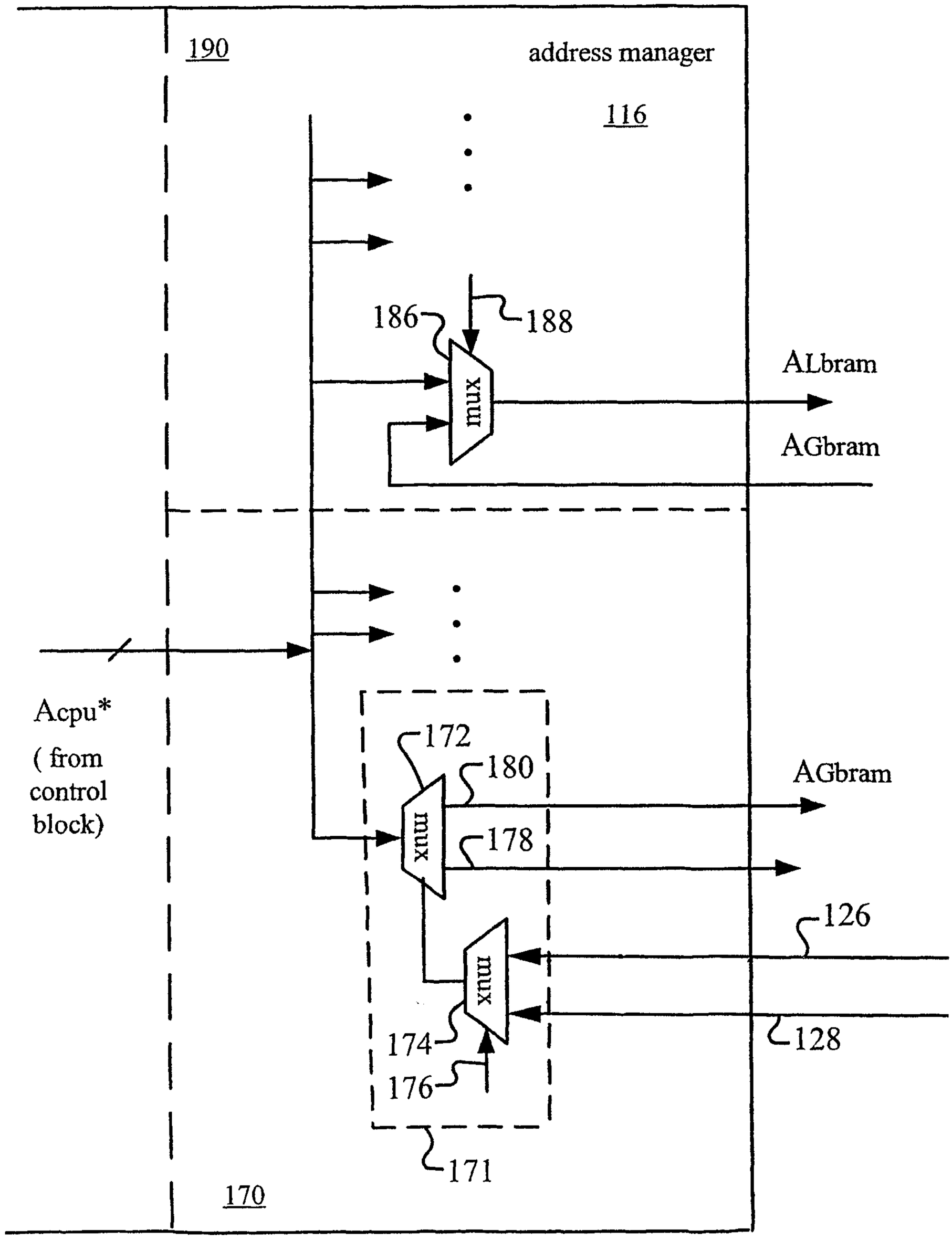
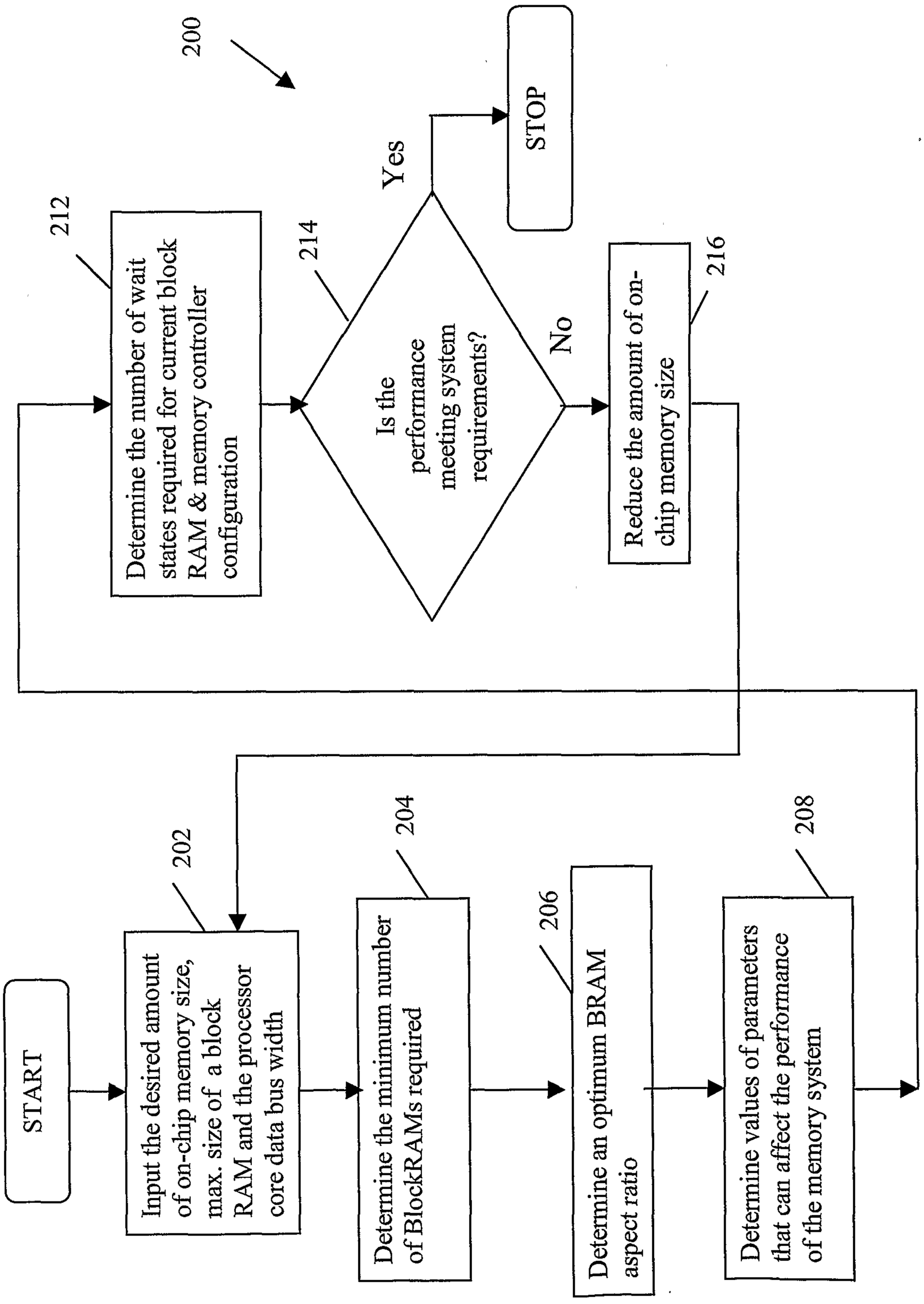

190
address manager
116
186
188
mux
ALbram
AGbram
Acpu*
(from
control
block)
172
180
178
mux
AGbram
126
mux
174
176
128
171
170

Fig. 3B

FIG. 4


START
Input the desired amount of on-chip memory size, max. size of a block RAM and the processor core data bus width
202
Determine the minimum number of BlockRAMs required
204
Determine an optimum BRAM aspect ratio
206
Determine values of parameters that can affect the performance of the memory system
208
Determine the number of wait states required for current block RAM & memory controller configuration
212
Is the performance meeting system requirements?
214
Yes
STOP
No
Reduce the amount of on-chip memory size
216

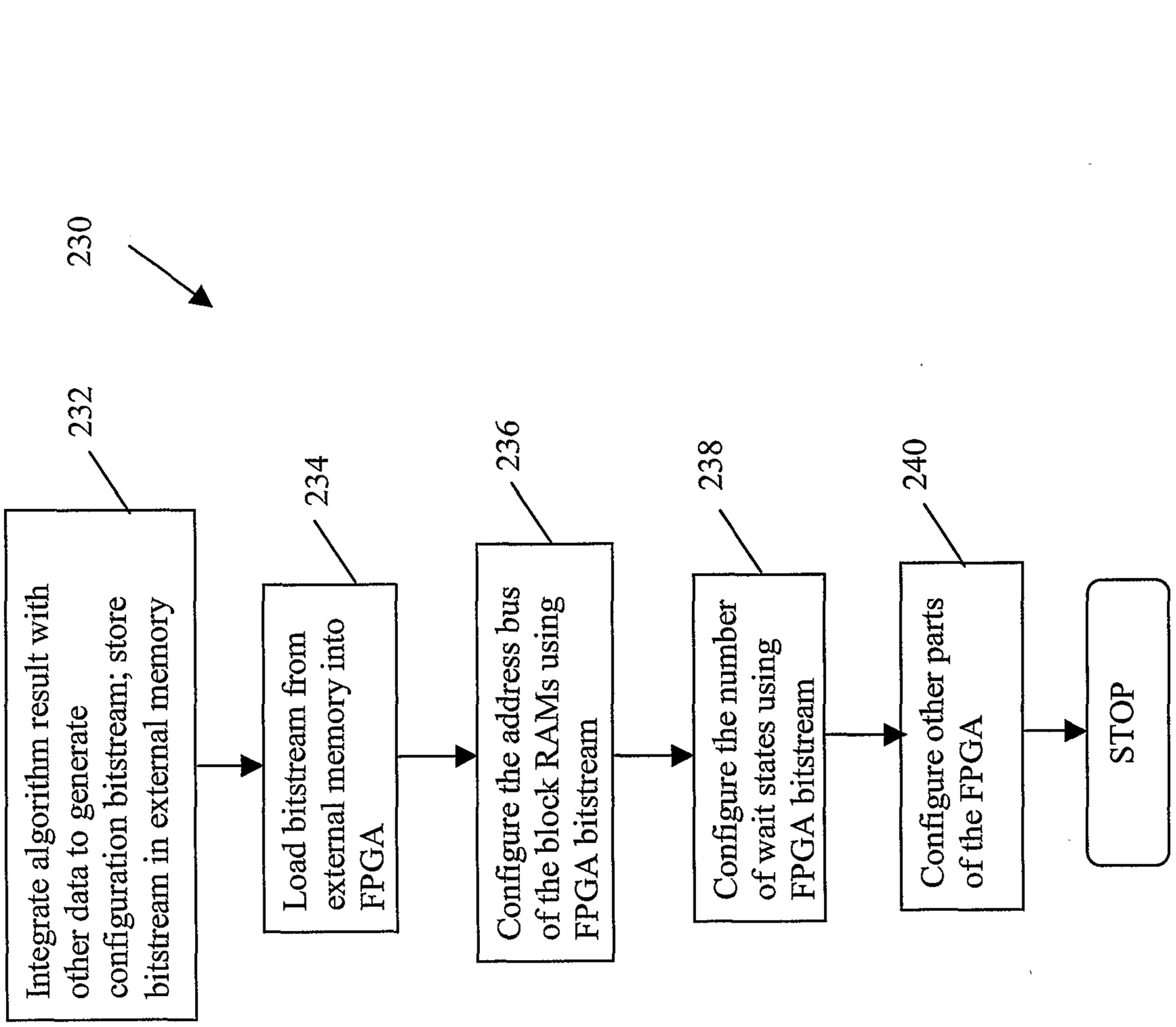

230
232
Integrate algorithm result with other data to generate configuration bitstream; store bitstream in external memory
234
Load bitstream from external memory into FPGA
236
Configure the address bus of the block RAMs using FPGA bitstream
238
Configure the number of wait states using FPGA bitstream
240
Configure other parts of the FPGA
STOP

FIG. 5

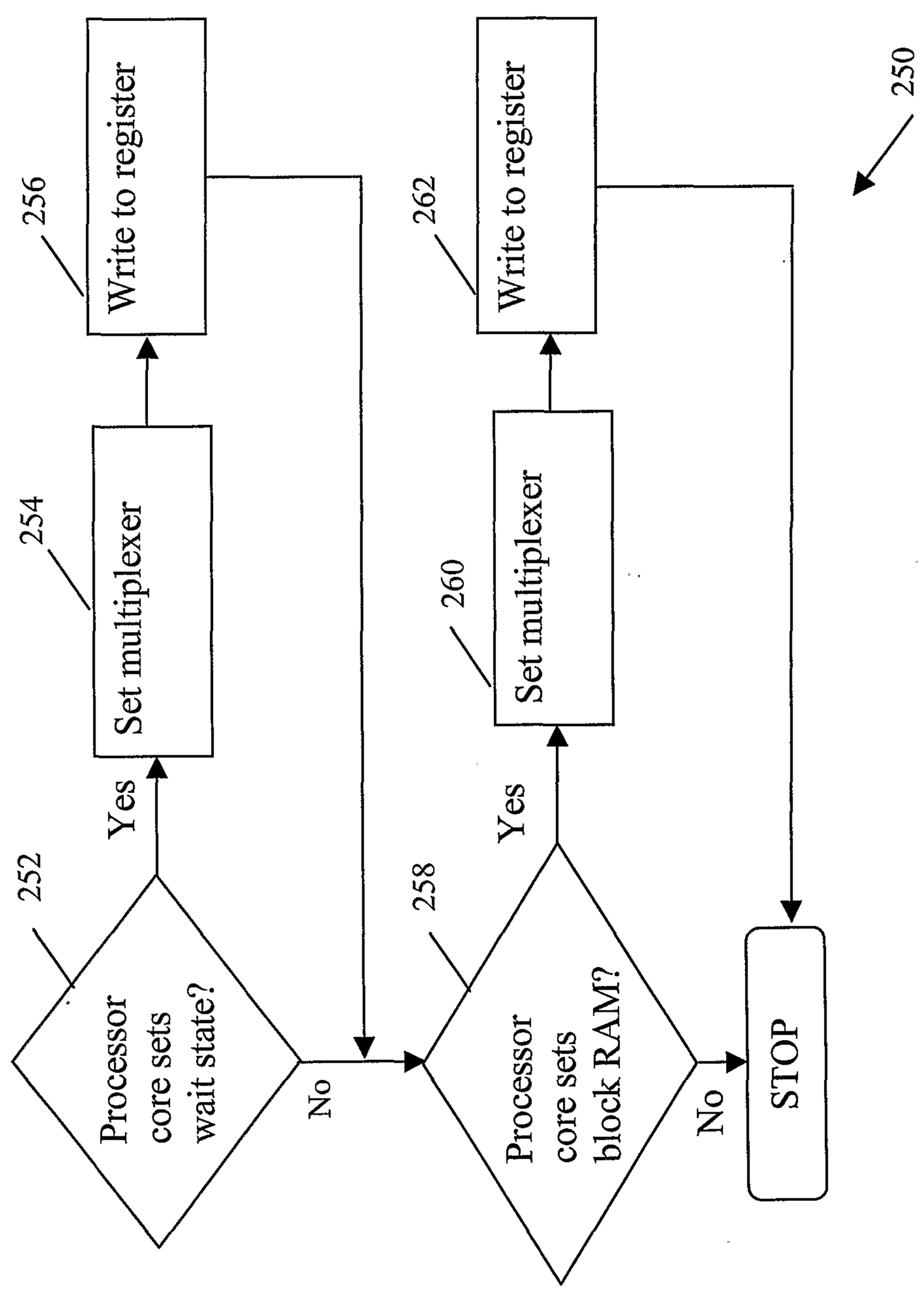

250
Processor core sets wait state?
252
Yes
Set multiplexer
254
Write to register
256
No
Processor core sets block RAM?
258
Yes
Set multiplexer
260
Write to register
262
No
STOP

FIG. 6

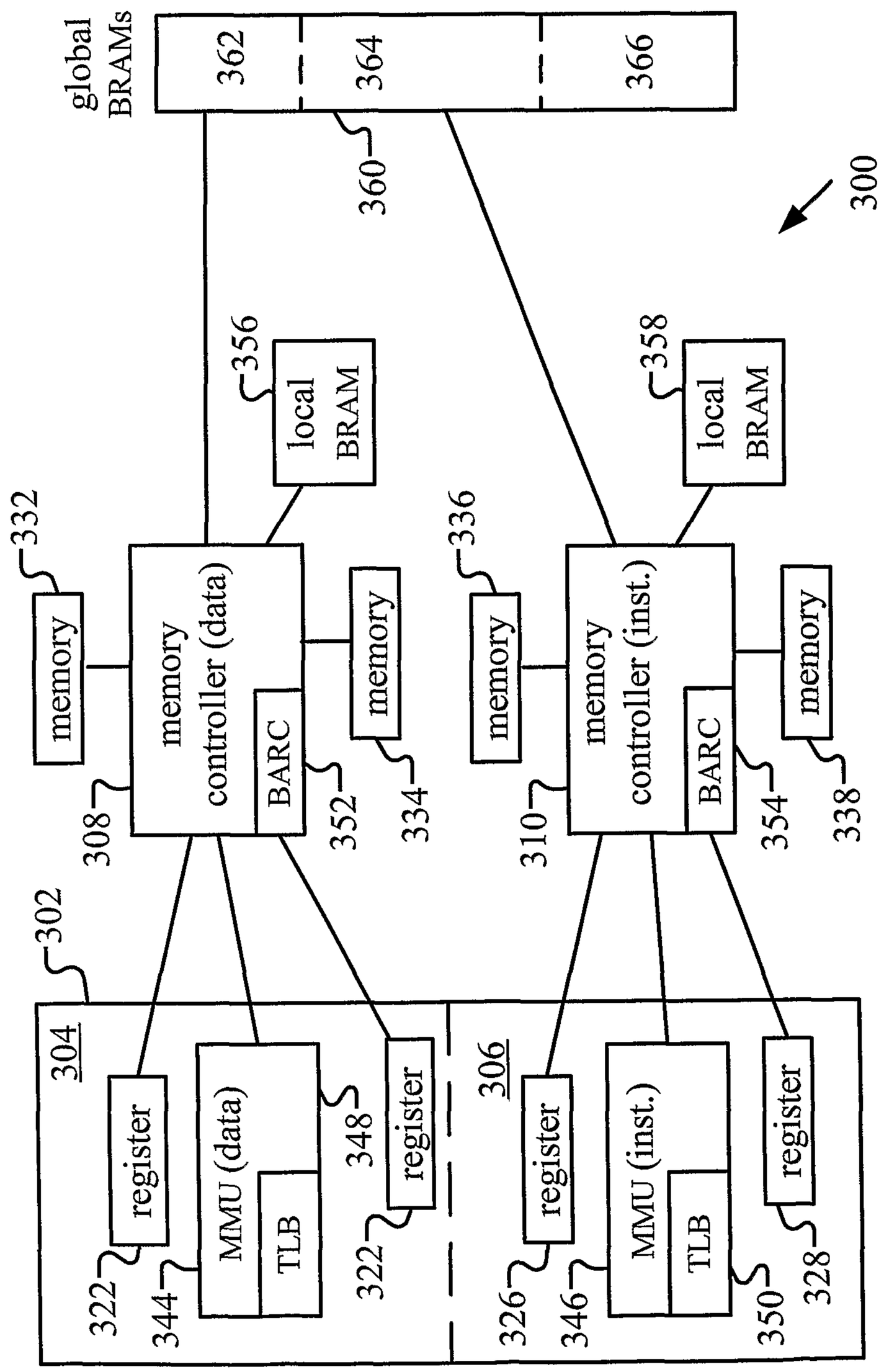
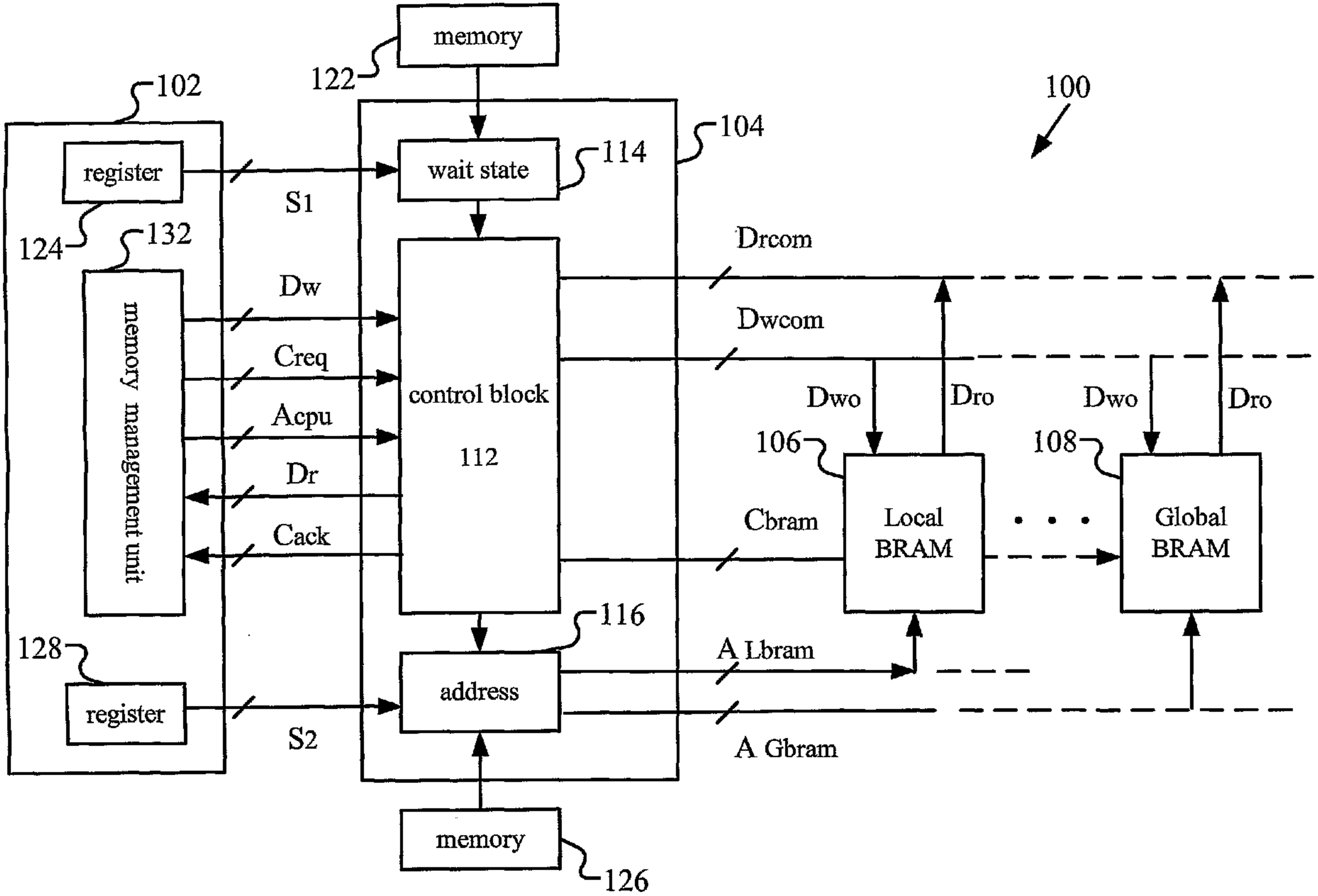

global BRAMs
362
364
366
360
300
356
local BRAM
358
local BRAM
332
336
memory
memory controller (data)
BARC
memory
memory
memory controller (inst.)
BARC
memory
308
352
334
310
354
338
302
304
register
MMU (data)
TLB
register
register
MMU (inst.)
TLB
register
306
322
344
348
322
326
346
350
328

Fig. 7


memory
122
102
100
104
register
114
wait state
S1
124
132
Drcom
Dw
Dwcom
Creq
control block
Dwo
Dro
Dwo
Dro
Acpu
112
106
108
Dr
Cbram
Local
BRAM
Global
BRAM
Cack
116
128
A Lbram
address
register
S2
A Gbram
memory
126