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

(11) **EP 3 844 620 B9**

(12) **CORRECTED EUROPEAN PATENT SPECIFICATION**

(15) Correction information:
Corrected version no 1 (W1 B1)
Corrections, see
Claims EN 7

(48) Corrigendum issued on:
19.03.2025 Bulletin 2025/12

(45) Date of publication and mention of the grant of the patent:
11.12.2024 Bulletin 2024/50

(21) Application number: **19768971.4**

(22) Date of filing: **29.08.2019**

(51) International Patent Classification (IPC):
G06F 13/16 *(2006.01)* ***H04L 12/18*** *(2006.01)*
G06F 9/52 *(2006.01)* ***G06F 9/50*** *(2006.01)*
G06N 3/063 *(2023.01)* ***G06N 3/10*** *(2006.01)*

(52) Cooperative Patent Classification (CPC):
G06F 9/52; G06F 13/1668; G06N 3/063; G06N 3/105; G06F 9/505; H04L 12/18; Y02D 10/00

(86) International application number:
PCT/US2019/048938

(87) International publication number:
WO 2020/047337 (05.03.2020 Gazette 2020/10)

(54) **METHOD, APPARATUS, AND SYSTEM FOR AN ARCHITECTURE FOR MACHINE LEARNING ACCELERATION**

VERFAHREN, VORRICHTUNG UND SYSTEM FÜR EINE ARCHITEKTUR ZUR BESCHLEUNIGUNG VON MASCHINENLERNEN

PROCÉDÉ, APPAREIL ET SYSTÈME POUR UNE ARCHITECTURE D'ACCÉLÉRATION D'APPRENTISSAGE AUTOMATIQUE

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **29.08.2018 US 201862724051 P**

(43) Date of publication of application:
07.07.2021 Bulletin 2021/27

(60) Divisional application:
24211735.6

(73) Proprietor: **QUALCOMM INCORPORATED**
San Diego, California 92121-1714 (US)

(72) Inventors:
- **VERRILLI, Colin Beaton**
  San Diego, California 92121-1714 (US)
- **VAIDHYANATHAN, Natarajan**
  San Diego, California 92121-1714 (US)
- **HILL, Rexford Alan**
  San Diego, California 92121-1714 (US)

(74) Representative: **Klang, Alexander H.**
Wagner & Geyer Partnerschaft mbB
Patent- und Rechtsanwälte
Gewürzmühlstrasse 5
80538 München (DE)

(56) References cited:
EP-A1- 3 343 460
WO-A1-2008/046716
US-A1- 2017 032 487
US-B1- 10 019 668

- **RUSSELL W DUREN ET AL: "Application of Data Compression to the MIL-STD-1553 Data Bus", 2008 IEEE AEROSPACE CONFERENCE; 1-8 MARCH 2008; BIG SKY; MT, USA, IEEE, PISCATAWAY, NJ, USA, 1 March 2008 (2008-03-01), pages 1 - 10, XP031256421, ISBN: 978-1-4244-1487-1**

EP 3 844 620 B9

Description

TECHNICAL FIELD

[0001] This disclosure relates to an inference accelerator computing architecture and corresponding method.

BACKGROUND

[0002] WO 2008/046716 A1 discloses a multi-processor computing system. The multi-processor computing system comprises a host processor, a global memory and at least one processing units, wherein the multi-processor computing system further comprises: a micro task sequencer, comprising a task acquisition device and a task scheduling device, wherein the task acquisition device is configured to fetch a command including micro task descriptions from the global memory if the micro task sequencer is capable to accommodate further commands, and the task scheduling device is configured to dispatch each micro task instruction defined by each of the micro task descriptions to one of the at least one processing units, so that the indicated processing unit can execute the micro task instruction, and the task scheduling device is further configured to detect the completion of all the micro tasks in the command and notify the host processor of the completion, wherein the processing unit can access the global memory by itself or by another of the at least one processing units.

[0003] Artificial Neural Networks (ANNs) are used to perform an increasing number and variety of tasks, such as, for example, object recognition, speech recognition, speech generation, providing recommendations, and predicting user behavior. Performing these tasks may be referred to as inferencing using an ANN model. To provide useful inferences, an ANN model needs to be designed and trained for the particular task. The ANN design establishes parameters such as the number of layers of the ANN model and the characteristics of each layer. The training of the ANN uses training data, inferencing using the ANN model, feedback based on evaluation of the inference, and backpropagation to adjust the weights of the ANN model in response to the feedback. After numerous training cycles of inferencing and backpropagation, the resultant model may provide satisfactory results in response to new input data. Note that many ANNs have multiple hidden layers between an input layer and an output layer and may consequently be referred to as Deep Neural Networks (DNNs).

[0004] To provide a satisfactory user experience, not only do the inference results need to be correct, but they also need to be provided fairly quickly - often within a fraction of a second (response latency within service level agreement). To do this, service providers use large arrays of inference accelerators located "in the cloud" - that is, communicatively coupled to, and located remotely from, a client device.

[0005] Client computer devices may include, for example, computers, automobiles, smartphones, smart wearable devices, and internet-of-things (IoT) devices. The so-called cloud may comprise a plurality of interconnected servers located at a data center and may be managed by a cloud provider entity such as, for example, Amazon.com, Inc. of Seattle, WA or Facebook, Inc., of Menlo Park, CA. Each host server comprises a plurality of interconnected inference accelerators, which may be provided by an inference-accelerator provider entity. Each accelerator comprises processor and memory components.

[0006] The cloud may support many millions of neural network applications. A neural network application running on a client computer device communicates with the cloud to receive inference acceleration and/or assistance. For example, a speech-translation neural-network application (NNA) may transmit a raw or encoded audio snippet to the cloud for rapid translation and provision of the translation in response to the NNA. A media-recommendation program that recommends, e.g., songs or videos - where the media may comprise many millions, or even billions, of options hosted by the cloud provider in the cloud - may communicate with the cloud to have the cloud perform an inference to generate a recommendation for provision to a user of the client computer device.

[0007] In the data center context, various heterogenous architectures have been employed to handle machine learning workloads. For example, cloud compute may use server-class central processing units (CPUs) or graphics processing units (GPUs) and may adapt their workloads to those architecture. However, these architectures may not be tailored to the specific characteristics of machine learning algorithms, with the effect that their performance is not as efficient as desired, and/or they consume more power to achieve a given level of performance than would be desirable. As there may be many millions of NNAs accessing the inference accelerators of the cloud at any one time, efficient inference accelerators would be beneficial for reducing power usage and/or reducing inference time.

[0008] Thus, it would be desirable to provide an inference-accelerator computing architecture that is scalable to cloud computing and data center application, while providing improved performance per watt when compared to existing server-class CPU and GPU-based solutions.

SUMMARY OF THE DISCLOSURE

[0009] The invention is set out in the appended set of claims.

[0010] Some advantages of the disclosed aspects may include providing a scalable architecture for cloud computing that provides improved interconnection between processing elements, and a compiler that produces a more efficient mapping of neural network operations onto available hardware.

BRIEF DESCRIPTION OF THE FIGURES

[0011]

FIG. 1 is block diagram of an exemplary inference accelerator in accordance with an embodiment of the disclosure.

FIG. 1A is a simplified schematic diagram of an exemplary implementation of the processing element of FIG. 1.

FIG. 2 is a flow diagram of an exemplary operation of a compiler for an inference accelerator in accordance with an embodiment of the disclosure.

DETAILED DESCRIPTION

[0012] With reference now to the drawing figures, several exemplary aspects of the present disclosure are described. The word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any aspect described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects.

[0013] FIG. 1 is a simplified schematic diagram of an exemplary inference accelerator 100 in accordance with an embodiment of the disclosure. The inference accelerator 100 comprises a system-on-chip (SoC) 190 coupled to a first double data rate (DDR) dynamic random-access memory (DRAM) 122 and a second DDR DRAM 126. The SoC 190 comprises a first processing element 102, a second processing element 104, a third processing element 106, and a fourth processing element 108. The processing elements 102, 104, 106, and 108 are coupled together via a compute network-on-chip (NoC) 142. Note that the terms "NoC" and "network" may be used interchangeably herein. Note that inference accelerators in accordance with this disclosure are not limited to any particular number of processing elements and alternative implementations may have more or fewer than four processing elements.

[0014] The SoC 190 further comprises a first memory interface 112, a second memory interface 114, a third memory interface 116, a fourth memory interface 118, and a PCI Express (PCIe) block 134, all coupled to each other and to the processing elements 102, 104, 106, and 108 via a system/memory (sys/mem) NoC 144. The PCIe block 134 is an interface used by the inference accelerator 100 to receive the inputs for inferences (e.g., images, videos, audio clips, or other data tensors) received by the host server and to provide results back to the host server. The system-on-chip 190 further comprises a management controller 132, which is coupled to the PCIe block 134, the memory controllers 112, 114, 116, and 118, and the processing elements 102, 104, 106, and 108 via the system/memory NoC 144. Note that, in some implementations, the compute network 142 may also connect to the PCIe block 134 and/or the memory controllers 112, 114, 116, and 118.

[0015] Further, a global synchronization manager (GSM) module 136 is coupled to the PCIe block 134 and a local sync manager (see FIG. 1A) in each processing element 102, 104, 106, and 108 via a private NoC 146. It should be noted that in alternative implementations, one or more of the compute NoC 142, sys/mem NoC , and private NoC 146 may be replaced by a corresponding simple bus, or other communication fabric, other than a NoC. It should be further noted that some alternative implementations of the inference accelerator 100 do not include a private NoC and, instead, the GSM 136 communicates with other elements (e.g., the processing elements 102, 104, 106, and 108) via other means (e.g., sys/mem NoC 144).

[0016] The processing elements 102, 104, 106, and 108 may be neural processing units (NPUs), neural signal processors (NSPs), digital signal processors (DSPs), or any other suitable type of processor (e.g., CPUs or GPUs). In some homogenous embodiments (where the processing elements 102, 104, 106, and 108 are substantially the same) each of the processing elements 102, 104, 106, and 108 may include scalar, vector, matrix processing capabilities (e.g., multiplication, convolution, point-wise addition, point-wise multiplication), and data-movement capabilities (e.g., load, store, and direct memory access (DMA)). In some alternative embodiments, the scalar, vector, matrix, and data-movement processing capabilities may be distributed across different processing elements (in other words, the processing elements 102, 104, 106, and 108 may be heterogeneous). Additionally, whichever of the processing elements 102, 104, 106, and 108 provide matrix processing capabilities may further include floating point capabilities as part of the matrix processing capabilities. Providing these capabilities in each of the processing elements 102, 104, 106, and 108 may enable a compiler for the inference accelerator 100 to more efficiently schedule code on the individual processing elements, as will be explained in greater detail with respect to FIG. 2.

[0017] FIG. 1A is a simplified schematic diagram of an exemplary implementation of the processing element 102 of FIG. 1. As noted above, in some embodiments, processing elements 104, 106, and 108 may be configured identically. The processing element 102 comprises tightly-coupled memory (TCM) 150, vector processing module 151, matrix processing module 152, scalar processing (e.g., DSP) module 153, memory processing module 154, and an optional local synchronization manager (LSM) 155. The TCM 150 is directly connected to at least the vector processing module 151, the matrix processing module 152, and the memory processing module 154. The LSM 155 is directly connected to at least the scalar processing module 153. The processing element 102 is connected to NoCs 142, 144, and 146.

[0018] In some implementations, each LSM 155 of a processing element is connected to the GSM 136 of FIG. 1, where the processing elements 102, 104, 106, and 108 implement hardware memory synchronization using

LSM 155 working with GSM 136 to coordinate and synchronize data transfers among the processing elements 102, 104, 106, and 108 and the DRAMs 122 and 126, by setting and resetting semaphores that allow or prohibit corresponding data operations. In this implementation, the LSM 155 may be referred to as a synchronization module. In some implementations, the GSM 136 works directly with the TCMs 150 of the processing elements 102, 104, 106, and 108 (forgoing LSMs 155) to set and reset values at known locations in the TCMs 150, where those values similarly allow or prohibit corresponding data operations. In this implementation, the TCM 150 may be referred to as a synchronization module.

[0019] The processing element 102 may forgo implementing a memory coherency protocol. Implementing a memory coherency protocol typically includes having a shared cache connected to a plurality of clients and an interconnecting bus with a coherency protocol to ensure that each client is referencing the latest version of corresponding data. Using caches and implementing a coherency protocol are useful when data movement and sharing is not sequential and not deterministic - in other words, what is conventionally referred to as "random." Caches and coherency are also useful where data movements and sharing are relatively fine-grained. Semaphores, on the other hand, use the setting and modifying of so-called semaphores to gate data movement among a plurality of clients and gate computations involving the data - without using a cache or a bus implementing a coherency protocol. Neural-network inferencing involves large movements of data, and calculations based on that data, whose pattern is known ahead of time. Consequently, the integrity of that data may be maintained using a relatively simple semaphore mechanism. Since implementing memory coherency protocols requires relatively significant power levels, substituting hardware synchronization for coherency allows the inference accelerator 100 to maintain the needed level of memory synchronization at a relatively reduced power level.

[0020] Returning to FIG. 1, the compute network 142 coupling the processing elements 102, 104, 106, and 108 may be a relatively higher-bandwidth network (as compared to the sys/mem network 144 and the private network 146), and supports multicast operations (i.e., sending data produced by a single processing element to multiple other processing elements of the inference accelerator 100). The processing elements 102, 104, 106, and 108 may each include tightly-coupled memory (e.g. TCM 150 of FIG. 1A), and interact with the first DRAM 122 and the second DRAM 126 via the sys/mem network 144 and the memory controllers 112, 114, 116, and 118. Both the compute network 142 and the sys/mem network 144 may support DMA operations from the TCMs (e.g. TCM 150) of each of the processing elements 102, 104, 106, and 108, including read operations, write operations, and, in the case of the compute network 142, multicast operations.

[0021] The private network 146 may be a relatively slower and lower-bandwidth network (as compared to the compute network 142 and the sys/mem network 144), as its use may be limited to a configuration time (as opposed to run time) and, thus, would not have a specific performance requirement (as opposed to the compute network 142 and the sys/mem network 144). Having separate networks for these specific purposes allows each of the networks to be designed to match its corresponding expected traffic type and allows each to be individually performance and power optimized to match.

[0022] For example, since the workloads handled by the inference accelerator 100 may often involve data words that are all zeros (but that must still be transmitted among the processing elements 102, 104, 106, and 108), the compute network 142 may implement a "zero" encoding protocol, where setting a single override bit on the network bus indicates that the value of the corresponding data word is zero, without having to actually set all the bits of the data bus for that data word to zero or read all of the corresponding bits of the data word. This may reduce power usage both directly and by allowing for the implementation of power-saving operations based on the override bit.

[0023] Further, as indicated above, the inference accelerator 100 does not implement a memory coherency protocol, instead managing dependencies that do occur using hardware semaphores and compiler design (as explained later with respect to Figure 2) in accordance with the global sync manager 136, which is configured to interact with the processing elements 102, 104, 106, and 108 to provide hardware semaphore support. Essentially, each of the processing elements 102, 104, 106, and 108 may set semaphores in the global sync manager 136, which may be cleared by the other processing elements 102, 104, 106, and 108 to allow for interdependencies in workloads being processing by the processing elements 102, 104, 106, and 108.

[0024] The latency involved in communications between the processing elements 102, 104, 106, and 108 and the global sync manager 136 may be important for the overall performance of the inference accelerator 100. Thus, the topology of the private network 146 providing connectivity between the global sync manager 136 and the processing elements 102, 104, 106, and 108 may depend on the relative number of processing elements that will be coupled to the global sync manager 136. In systems with relatively few processing elements, a ring topology may be used instead of the network 146 shown. In systems with larger numbers of processing elements, a star topology may be used. Those having skill in the art will recognize that the choice of topology may be informed by many factors involved in the overall system design, and the teachings of the present disclosure do not depend on the use of a particular topology.

[0025] FIG. 2 is a hybrid schematic and flow diagram 200 for exemplary operation of a compiler which may be configured to schedule operations on inference accelerator 100 of FIG. 1. A neural network description 210 is

provided to the compiler, which, in a first phase, transforms, in step 220, the neural network description 210 into a form that may be represented by directed acyclic graph 230. A directed acyclic graph is a graph that has forward progress, without loopbacks, among its nodes (e.g., a tree structure progressing from the trunk to the leaves). The graph 230 comprises a plurality of tasks represented by graph nodes 231, 232, 233, 234, 235, 236, and 237. Graph 230 shows that task 231 must be performed first, and then task 232, but then any of tasks 233, 234, and 235 may be performed. In addition, graph 230 shows that both tasks 234 and 235 have to be completed before task 236 can be executed (in other words, task 236 is dependent on tasks 234 and 235). Similarly, task 237 is dependent on tasks 233 and 236.

[0026] In a second phase, in step 240, the compiler converts the tasks 231-237, shown in graph 230, into command lists 252, 254, 256, and 258 and schedules them for processing on corresponding hardware processing elements such as scalar, vector, matrix, and data movement blocks of the processing elements 102, 104, 106, and 108 of FIG. 1. In other words, command lists 252, 254, 256, and 258 may correspond, respectively, to vector processing module 151, matrix processing module 152, scalar processing module 153, and memory processing module 154 of FIG. 1A. The scheduling may be optimized for factors such as, for example, time, power, or resource requirements.

[0027] The compiler may be optimized for use with neural networks, and thus it may generate "static" workloads. Specifically, since branching and iteration counts may be known ahead of time, they may be used to generate static workloads, as opposed to, for example, conventional CPU or GPU code, which may have unpredictable branching behavior and iteration counts and, consequently, would require generating dynamic workloads. Because these workloads are static, the command lists generated by the compiler may permit workload balancing in the computing device 100 by dispatching a portion of a total workload to be executed to the computing device 100 after which the computing device 100 may wait (and may possibly even enter a low-power state) for further instructions. This workload distribution and balancing is referred to herein as "dispatch scaling." Note that, in generating parallel workloads, the compiler may direct the replication of data sets between processing elements, where the replication may be performed using the multicast capabilities of the compute network 142.

[0028] The above is possible because, since the workload is static, dispatching one-fourth of a total workload (e.g., one fourth of the total operations), for example, will result in the one-fourth of the total workload being completed. This contrasts with a conventional CPU/GPU workload, in which it may be essentially impossible to predict ahead of time how much of a total workload may be completed by providing one-fourth of the workload to the computing device, and thus, in order to save power, conventional methods such a frequency and voltage scaling may be used. Further, instead of generating command lists, which would conventionally be interpreted by software running on the processing elements 102, 104, 106, and 108, the compiler may alternatively generate static code which is executed in sequence. Dispatch scaling may be used in either case (command lists or statically generated code).

[0029] Although the compiler attempts to generate command lists that are fully parallelizable and do not have interdependencies, sometimes this may not be feasible. In cases where interdependencies exist, since the computing device 100 does not implement coherency, the compiler will insert a synchronization indicator (e.g., a semaphore) that is mapped to a hardware semaphore resource. Different processing elements may interact, via, e.g., GSM 136, using the semaphore to guarantee that dependencies are satisfied. The compiler may schedule tasks to command lists based on optimistic estimated completion times and the semaphores may be relied on to guarantee that dependencies are satisfied where actual completion times exceed the estimated completion times.

[0030] Those of skill in the art will further appreciate that the various illustrative logical blocks, modules, circuits, and algorithms described in connection with the aspects disclosed herein may be implemented as electronic hardware, instructions stored in memory or in another computer readable medium and executed by a processor or other processing device, or combinations of both. The devices described herein may be employed in any circuit, hardware component, IC, or IC chip, as examples. Memory disclosed herein may be any type and size of memory and may be configured to store any type of information desired. To clearly illustrate this interchangeability, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. How such functionality is implemented depends upon the particular application, design choices, and/or design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present disclosure.

[0031] The various illustrative logical blocks, modules, and circuits described in connection with the aspects disclosed herein may be implemented or performed with a processor, a Digital Signal Processor (DSP), an Application Specific Integrated Circuit (ASIC), a Field Programmable Gate Array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein. A processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing

devices (e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration).

[0032] The aspects disclosed herein may be embodied in hardware and in instructions or design data that are stored in hardware, and may reside, for example, in Random Access Memory (RAM), flash memory, Read Only Memory (ROM), Electrically Programmable ROM (EPROM), Electrically Erasable Programmable ROM (EEPROM), registers, a hard disk, a removable disk, a CD-ROM, or any other form of computer readable medium known in the art. An exemplary storage medium is coupled to the processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. In the case of design data, the data may be an electronic representation of a physical design of a circuit, may be readable by integrated circuit fabrication equipment, and may be in a file format such as GDSII, GERBER, or the like. The processor and the storage medium may reside in an ASIC. The ASIC may reside in a remote station. In the alternative, the processor and the storage medium may reside as discrete components in a remote station, base station, or server.

[0033] It is also noted that the operational steps described in any of the exemplary aspects herein are described to provide examples and discussion. The operations described may be performed in numerous different sequences other than the illustrated sequences. Furthermore, operations described in a single operational step may actually be performed in a number of different steps. Additionally, one or more operational steps discussed in the exemplary aspects may be combined. It is to be understood that the operational steps illustrated in the flowchart diagrams may be subject to numerous different modifications as will be readily apparent to one of skill in the art. Those of skill in the art will also understand that information and signals may be represented using any of a variety of different technologies and techniques. For example, data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

[0034] The previous description of the disclosure is provided to enable any person skilled in the art to make or use the disclosure. Various modifications to the disclosure will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other variations without departing from the scope of the claims. Thus, the disclosure is not intended to be limited to the examples and designs described herein, but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

Claims

1. An inference accelerator (100) comprising:

a memory system (112, 114, 116, 118);
a plurality of processing elements (102, 104, 106, 108), each processing element:

having a tightly coupled memory (150), TCM;
coupled to the memory system; and
adapted to access the memory system;

a global synchronization manager (136), GSM, module coupled to the plurality of processing elements and to the memory system, the GSM adapted to synchronize operations of the plurality of processing elements and memory system using corresponding synchronization modules of each of the plurality of processing elements;
a second network (144) connecting each of the plurality of processing elements to the memory system,
characterised in further comprising:

a first network (142) interconnecting the plurality of processing elements and the first network configured to support multicast operations,
wherein the second network (144) is separate from the first network.

2. The inference accelerator of claim 1, wherein each processing element further comprises:

a vector processor (151) adapted to perform floating point operations;
a scalar processor (153); and

a matrix processor (152) adapted to perform floating point operations.

3. The inference accelerator of claim 1, wherein the inference accelerator further comprising a controller (132) connected to the second network.

4. The inference accelerator of claim 1, wherein:

the GSM is coupled to each of the processing elements via a third network (146) separate from the first network and the second network;
each of the processing elements comprises a local sync manager (155); and
the GSM is configured to provide configuration information to the local sync manager of each processing element of the plurality of processing elements via the third network.

5. The inference accelerator of claim 1, wherein the first network is configured to implement zero encoding.

6. The inference accelerator of claim 1, wherein:

the synchronization modules of the plurality of processing elements used by the GSM to synchronize operations of the plurality of processing elements are the corresponding TCMs;
each TCM is adapted to store a set of synchronization variables; and
the GSM is adapted to store and adjust the synchronization variables in the TCMs.

7. A method for an inference accelerator (100) having a plurality of processing elements (112, 114, 116, 118), a memory system (102, 104, 106, 108) coupled to each of the processing elements, and a global synchronization manager (136), GSM, module coupled to the plurality of processing elements and to the memory system, wherein each processing element comprises a tightly coupled memory (150), TCM, the method comprising:

accessing, by each processing element, the memory system; and
synchronizing, by the GSM, operations of the plurality of processing elements and memory system using corresponding synchronization modules of each of the plurality of processing elements;

wherein:

each of the plurality of processing elements is connected to the memory system by a second network (144),
characterised in that:

the plurality of processing elements are interconnected by a first network (142);
the method further comprises performing multicast operations by the first network (142); and
the second network (144) is separate from the first network (142).

8. The method of claim 7, wherein:

each processing element further comprise a vector processor (151), a scalar processor (153), and a matrix processor (152); and
the method further comprises:
performing floating point operations by the vector processor; and performing floating point operations by the matrix processor.

9. The method of claim 7, wherein:

the GSM is coupled to each of the processing elements via a third network (146) separate from the first network and the second network;
each of the processing elements comprises a local sync manager (155);
the method further comprises providing, by the GSM, configuration information to the local sync manager of each processing element of the plurality of processing elements via the third network.

10. The method of claim 7, further comprising implementing zero encoding by the first network.

Patentansprüche

1. Ein Inferenzbeschleuniger (100), der Folgendes aufweist:

ein Speichersystem (112, 114, 116, 118);
eine Vielzahl von Verarbeitungselementen (102, 104, 106, 108), wobei jedes Verarbeitungselement:

einen eng gekoppelten Speicher bzw. TCM (TCM = tightly coupled memory) (150) hat; an das Speichersystem gekoppelt ist; und ausgelegt ist auf das Speichersystem zuzugreifen;
ein Globalsynchronisationsmanager- bzw. GSM-Modul (GSM = global synchronization manager) (136), das an die Vielzahl von Verarbeitungselementen und das Speichersystem gekoppelt ist, wobei der GSM ausgelegt ist zum Synchronisieren von Operationen der Vielzahl von Verarbeitungselementen und des Speichersystems unter Nutzung entsprechender Synchronisationsmodule von jedem der Vielzahl von Verarbeitungselementen; und
ein zweites Netzwerk (144), das jedes der Vielzahl von Verarbeitungselementen mit dem Speichersystem verbindet,
dadurch gekennzeichnet, dass er weiter Folgendes aufweist:

ein erstes Netzwerk (142), das die Vielzahl von Verarbeitungselementen und das erste Netzwerk miteinander verbindet, das konfiguriert ist zum Unterstützen von Multicast-Operationen,
wobei das zweite Netzwerk (144) von dem ersten Netzwerk getrennt ist.

2. Inferenzbeschleuniger nach Anspruch 1, wobei jedes Verarbeitungselement weiter Folgendes aufweist:

einen Vektorprozessor (151), der ausgelegt ist zum Durchführen von Gleitkomma-Operationen;
einen Skalarprozessor (153); und
einen Matrixprozessor (152), der ausgelegt ist zum Durchführen von Gleitkomma-Operationen.

3. Inferenzbeschleuniger nach Anspruch 1, wobei der Inferenzbeschleuniger einen Controller bzw. eine Steuervorrichtung (132) aufweist, der bzw. die mit dem zweiten Netzwerk verbunden ist

4. Inferenzbeschleuniger nach Anspruch 1, wobei:

der GSM an jedes der Verarbeitungselemente über ein drittes Netzwerk (146) gekoppelt ist, das separat von dem ersten Netzwerk und dem zweiten Netzwerk ist;
jedes der Verarbeitungselemente einen lokalen Sync-Manager (155) aufweist; und
der GSM konfiguriert ist zum Vorsehen von Konfigurationsinformation an den lokalen Sync-Manager jedes Verarbeitungselements der Vielzahl von Verarbeitungselementen über das dritte Netzwerk.

5. Inferenzbeschleuniger nach Anspruch 1, wobei das erste Netzwerk konfiguriert ist Null-Codierung zu implementieren.

6. Inferenzbeschleuniger nach Anspruch 1, wobei:

die Synchronisationsmodule der Vielzahl von Verarbeitungselementen, die durch den GSM zum Synchronisieren von Operationen der Vielzahl von Verarbeitungselementen genutzt werden, die entsprechenden TCMs sind;
jeder TCM ausgelegt ist zum Speichern eines Satzes von Synchronisationsvariablen; und
der GSM ausgelegt ist zum Speichern und Anpassen der Synchronisationsvariablen in den TCMs.

7. Ein Verfahren für einen Inferenzbeschleuniger (100), der eine Vielzahl von Verarbeitungselementen (112, 114, 116, 118), ein Speichersystem (102, 104, 106, 108), das an jedes der Verarbeitungselemente gekoppelt ist, und ein Globalsynchronisationsmanager- bzw. GSM-Modul (GSM = global synchronization manager) (136), das an die Vielzahl von Verarbeitungselementen und an das Speichersystem gekoppelt ist, hat, wobei jedes Verarbeitungselement einen eng gekoppelten Speicher bzw. TCM (TCM = tightly coupled memory) (150) aufweist, wobei das Verfahren Folgendes aufweist:

Zugreifen, durch jedes Verarbeitungselement, auf das Speichersystem; und
Synchronisieren, durch den GSM, von Operationen der Vielzahl von Verarbeitungselementen und des Speichersystems unter Nutzung von entsprechenden Synchronisationsmodulen von jedem der Vielzahl von Verarbeitungselementen;
wobei:

jedes der Vielzahl von Verarbeitungselementen mit dem Speichersystem durch ein zweites Netzwerk (144) verbunden ist, **dadurch gekennzeichnet, dass**:

die Vielzahl von Verarbeitungselementen durch ein erstes Netzwerk (142) miteinander verbunden sind;
wobei das Verfahren weiter Durchführen von Multicast-Operationen durch das erste Netzwerk (142) aufweist; und
das zweite Netzwerk (144) separat von dem ersten Netzwerk (142) ist.

8. Verfahren nach Anspruch 7, wobei:

jedes Verarbeitungselement weiter einen Vektorprozessor (151), einen Skalarprozessor (153) und einen Matrixprozessor (152) aufweist; und
das Verfahren weiter Folgendes aufweist:

Durchführen von Gleitkomma-Operationen durch den Vektorprozessor; und
Durchführen von Gleitkomma-Operationen durch den Matrixprozessor.

9. Verfahren nach Anspruch 7, wobei:

der GSM an jedes der Verarbeitungselemente über ein drittes Netzwerk (146) gekoppelt ist, das separat von dem ersten Netzwerk und dem zweiten Netzwerk ist;
jedes der Verarbeitungselemente einen lokalen Sync-Manager (155) aufweist;
wobei das Verfahren weiter Vorsehen, durch den GSM, von Konfigurationsinformation an den lokalen Sync-Manager jedes Verarbeitungselementes der Vielzahl von Verarbeitungselementen über das dritte Netzwerk aufweist.

10. Verfahren nach Anspruch 7, das weiter Implementieren von Null-Codierung durch das erste Netzwerk aufweist.

Revendications

1. Accélérateur d'inférence (100) comprenant :

un système mémoire (112, 114, 116, 118) ;
une pluralité d'éléments de traitement (102, 104, 106, 108), chaque élément de traitement :

ayant une mémoire étroitement couplée (150), TCM ;
étant couplé au système mémoire ; et
étant adapté pour accéder au système mémoire ;

un module de gestion de synchronisation globale (136), GSM, couplé à la pluralité d'éléments de traitement et au système mémoire, le GSM étant adapté pour synchroniser des opérations de la pluralité d'éléments de traitement et du système mémoire à l'aide de modules de synchronisation correspondants de chacun de la pluralité d'éléments de traitement ;
un deuxième réseau (144) connectant chacun de la pluralité d'éléments de traitement au système mémoire,
caractérisé en ce qu'il comprend en outre :

un premier réseau (142) interconnectant la pluralité d'éléments de traitement et le premier réseau configuré pour prendre en charge des opérations de multidiffusion,
dans lequel le deuxième réseau (144) est distinct du premier réseau.

2. Accélérateur d'interférence selon la revendication 1, dans lequel chaque élément de traitement comprend en outre :

un processeur vectoriel (151) adapté pour effectuer des opérations en virgule flottante ;
un processeur scalaire (153) ; et

un processeur matriciel (152) adapté pour effectuer des opérations en virgule flottante.

3. Accélérateur d'interférence selon la revendication 1, dans lequel l'accélérateur d'inférence comprend en outre un contrôleur (132) connecté au deuxième réseau.

4. Accélérateur d'interférence selon la revendication 1, dans lequel :

le GSM est couplé à chacun des éléments de traitement par l'intermédiaire d'un troisième réseau (146) distinct du premier réseau et du deuxième réseau ;
chacun des éléments de traitement comprend un gestionnaire de synchronisation local (155) ; et
le GSM est configuré pour fournir des informations de configuration au gestionnaire de synchronisation local de chaque élément de traitement de la pluralité d'éléments de traitement par l'intermédiaire du troisième réseau.

5. Accélérateur d'interférence selon la revendication 1, dans lequel le premier réseau est configuré pour mettre en oeuvre un codage zéro.

6. Accélérateur d'interférence selon la revendication 1, dans lequel :

les modules de synchronisation de la pluralité d'éléments de traitement utilisés par le GSM pour synchroniser des opérations de la pluralité d'éléments de traitement sont les TCM correspondants ;
chaque TCM est adapté pour stocker un ensemble de variables de synchronisation ; et
le GSM est adapté pour stocker et régler les variables de synchronisation dans les TCM.

7. Procédé pour un accélérateur d'inférence (100) ayant une pluralité d'éléments de traitement (112, 114, 116, 118), un système mémoire (102, 104, 106, 108) couplé à chacun des éléments de traitement, et un module de gestion de synchronisation globale (136), GSM, couplé à la pluralité d'éléments de traitement et au système mémoire, dans lequel chaque élément de traitement comprend une mémoire étroitement couplée (150), TCM, le procédé comprenant :

l'accès, par chaque élément de traitement, au système mémoire ; et
la synchronisation, par le GSM, d'opérations de la pluralité d'éléments de traitement et du système mémoire à l'aide de modules de synchronisation correspondants de chacun de la pluralité d'éléments de traitement ;
dans lequel :

chacun de la pluralité d'éléments de traitement est connecté au système mémoire par un deuxième réseau (144),
caractérisé en ce que :

la pluralité d'éléments de traitement est interconnectée par un premier réseau (142) ;
le procédé comprend en outre la mise en oeuvre d'opérations de multidiffusion par le premier réseau (142) ; et
le deuxième réseau (144) est distinct du premier réseau (142) .

8. Procédé selon la revendication 7, dans lequel :

chaque élément de traitement comprend en outre un processeur vectoriel (151), un processeur scalaire (153) et un processeur matriciel (152) ; et
le procédé comprend en outre :

la mise en oeuvre d'opérations en virgule flottante par le processeur vectoriel ; et
la mise en oeuvre d'opérations en virgule flottante par le processeur matriciel.

9. Procédé selon la revendication 7, dans lequel :

le GSM est couplé à chacun des éléments de traitement par l'intermédiaire d'un troisième réseau (146) distinct du premier réseau et du deuxième réseau ;
chacun des éléments de traitement comprend un gestionnaire de synchronisation local (155) ;
le procédé comprend en outre la fourniture, par le GSM, d'informations de configuration au gestionnaire de synchronisation local de chaque élément de traitement de la pluralité d'éléments de traitement par l'intermédiaire du troisième réseau.

10. Procédé selon la revendication 7, comprenant en outre la mise en oeuvre d'un codage zéro par le premier réseau.

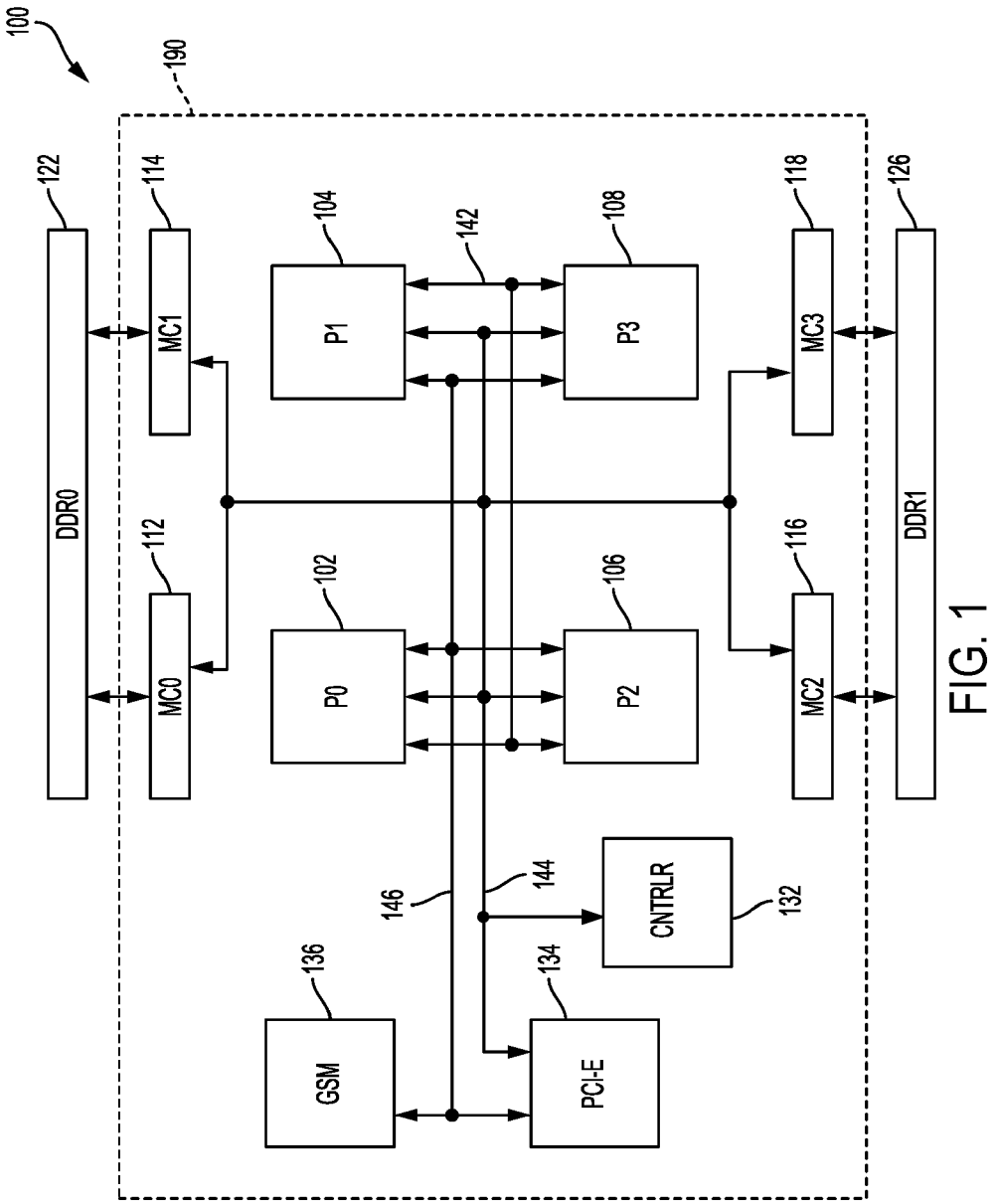
100
190
122
114
104
142
108
118
126
DDR0
MC1
P1
P3
MC3
DDR1
112
102
106
116
MC0
P0
P2
MC2
146
144
CNTRLR
132
136
134
GSM
PCI-E

FIG. 1

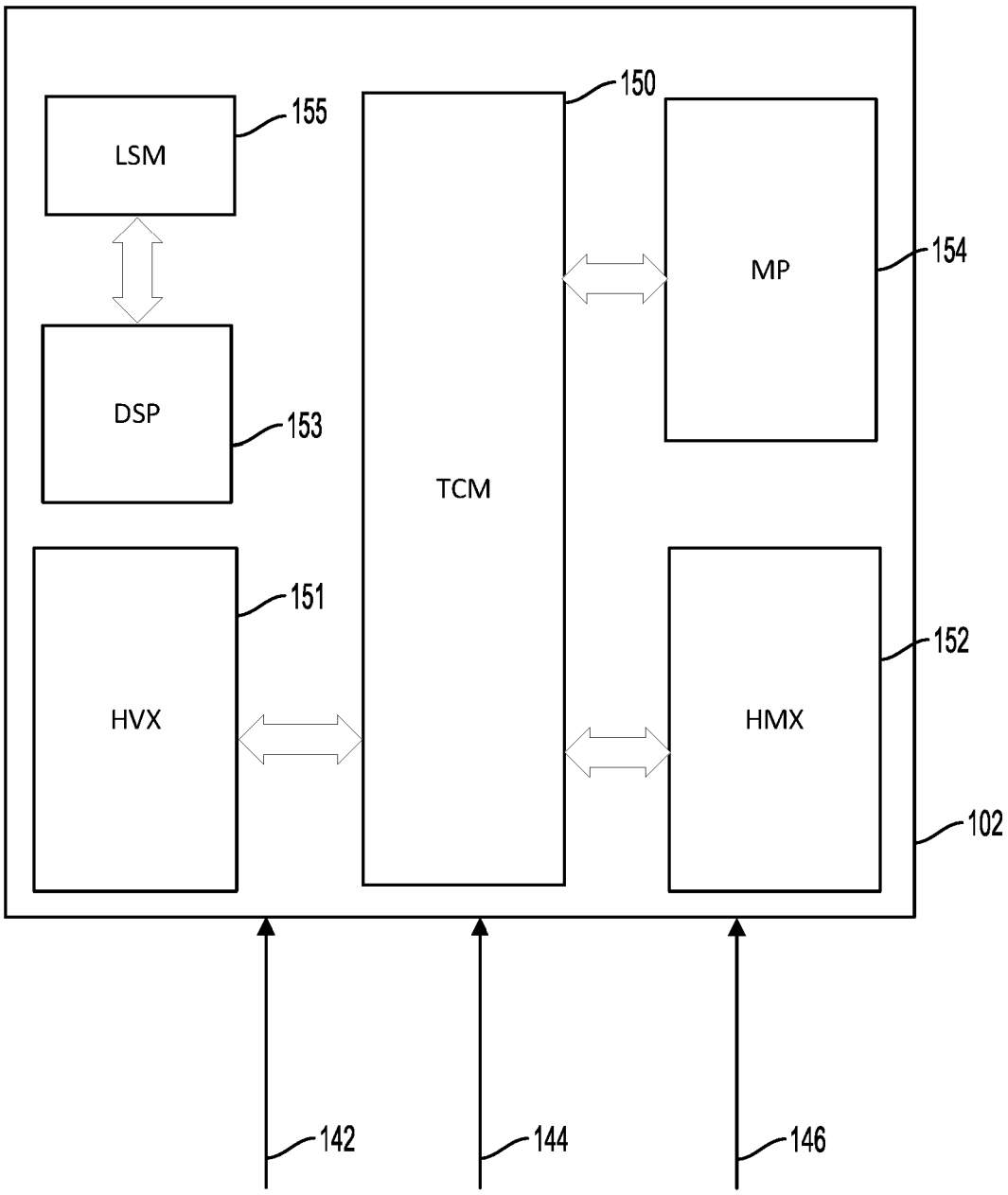

FIG. 1A

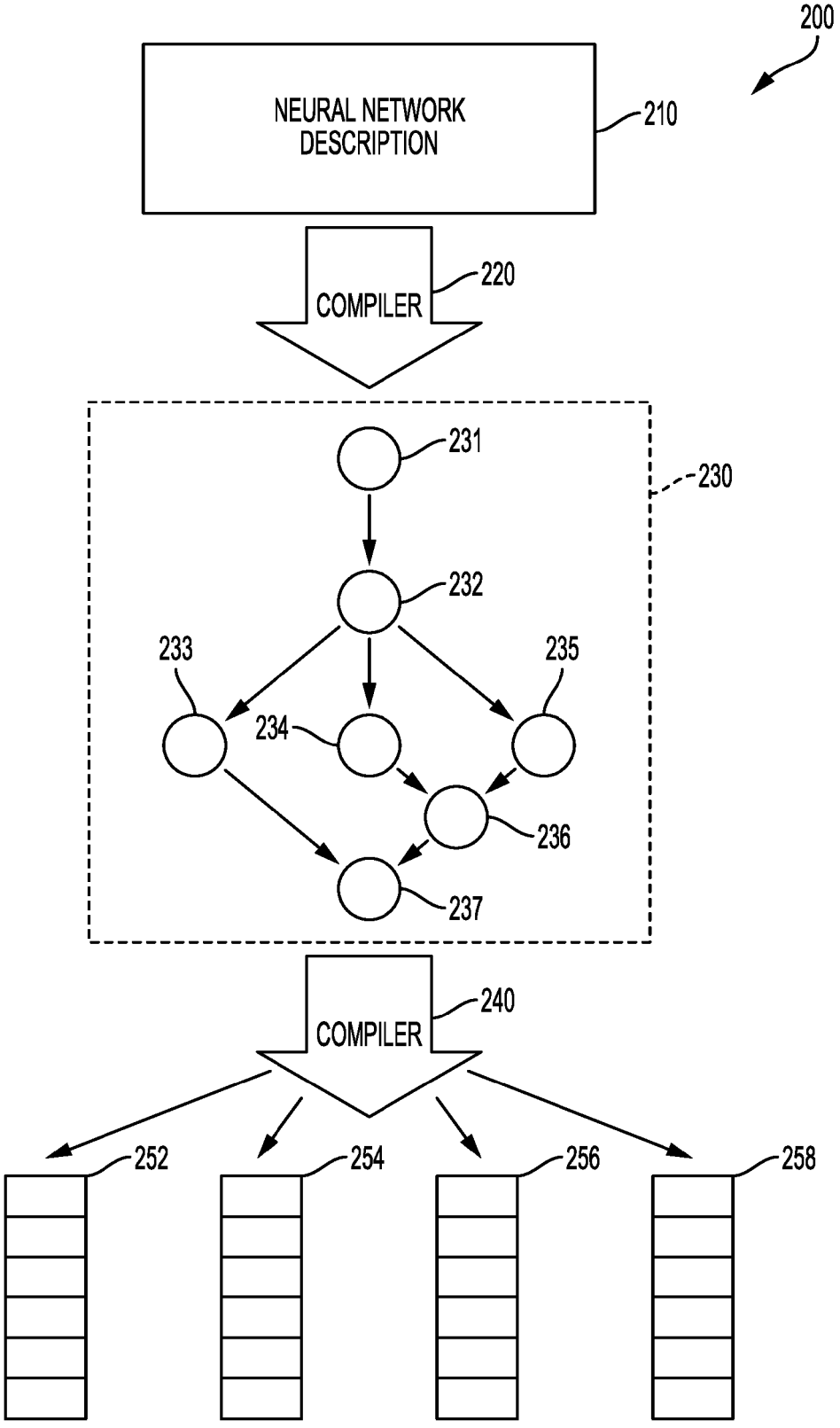
200
NEURAL NETWORK DESCRIPTION
210
COMPILER
220
230
231
232
233
234
235
236
237
COMPILER
240
252
254
256
258

FIG. 2

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 2008046716 A1 **[0002]**