# DATA PROCESSING APPARATUS AND DATA PROCESSING METHOD

## BACKGROUND

With the development of science and technology, human society is rapidly entering an era of intelligence. The important feature of the era of intelligence is that people may acquire more and more types of data, the amount of data that people may acquire is becoming larger and larger, and the demand for the data processing speed is becoming higher and higher.

Chips are the cornerstone of data processing, which essentially determine the ability to process data. From the perspective of application fields, the chips are mainly developed in two routes. One is general-purpose chips, for example, central processing units (CPU) and the like, which can provide great flexibility but have low effective computing power when processing algorithms in a specific field; the other is special-purpose chips, for example, tensor processing units (TPU) and the like, which have high effective computing power in certain specific fields but are poor in processing capacity or even have no processing capacity in flexible and universal fields.

Due to various data types and numerous data volumes in the era of intelligence, the chips are required to not only process changing algorithms in different fields with extremely high flexibility, but also rapidly process a great number of dramatically increasing data volumes with extremely strong processing capacities.

## SUMMARY

1. Objective

The present disclosure aims to provide a data processing apparatus and data processing method. The data processing apparatus is provided with a synchronization scheduler and a direct memory access controller, and the direct memory access controller is instructed by the synchronization scheduler to read a program corresponding to each of the processing cores from an external storage unit, and send the program to the corresponding processing core. On the one hand, in the data processing apparatus provided by the embodiment of the present disclosure, the processing cores are not required to fetch data from the external storage unit, which avoids the delay caused by multiple processing cores reading data, and improves the computing power of the chip. On the other hand, in the data processing apparatus provided by the embodiment of the present disclosure, the programs executed by each

of the processing cores can be the same or different, and the synchronization scheduler responds to the update of the programs of multiple processing cores, so that the tasks can be allocated flexibly, the computing power of each processing core can be fully utilized, and the computing power of the apparatus can be further improved.

2. Technical Solution

To solve the above problems, a first aspect of the present disclosure provides a data processing apparatus, including: at least two processing cores; a synchronization scheduler, configured to generate and send a configuration signal in response to a program update signal from each of the processing cores connected to the synchronization scheduler; and a direct memory access controller, configured to read a program corresponding to each of the processing cores from an external storage unit and send the program to the corresponding processing core, based on the configuration signal.

The data processing apparatus provided by the embodiment of the present disclosure is provided with a synchronization scheduler and a direct memory access controller, and the direct memory access controller is instructed by the synchronization scheduler to read a program from an external storage unit, and send the program to the corresponding processing core. On the one hand, in the data processing apparatus provided by the embodiment of the present disclosure, the processing cores are not required to fetch data from the external storage unit, which avoids the delay caused by multiple processing cores reading data, and improves the computing power of the chip. On the other hand, in the data processing apparatus provided by the embodiment of the present disclosure, the programs executed by each of the processing cores can be the same or different, and the synchronization scheduler responds to the update of the programs of multiple processing cores, so that the tasks can be allocated flexibly, the computing power of each processing core can be fully utilized, and the computing power of the chip can be further improved.

Further, the synchronization scheduler is further configured to, in response to the program update signal, send a synchronous operation signal to each of the processing cores connected to the synchronization scheduler, wherein the synchronous operation signal is sent to each of the processing cores after the synchronization scheduler receives a predetermined number of the program update signals, and the synchronous

operation signal is configured to instruct each of the processing cores to start executing its respective program simultaneously.

Further, the program includes a plurality of program segments.

Further, the processing cores are configured to send the program update signal to the synchronization scheduler after executing each of the program segments.

Further, the direct memory access controller is configured to read the program segment corresponding to each of the processing cores from the external storage unit, and send the program segments to the corresponding processing cores; and the direct memory access controller is further configured to send a configuration completion signal to the synchronization scheduler after sending the program segments to the corresponding processing cores.

Further, the synchronization scheduler is configured to, in response to the program update signal and the configuration completion signal, send the synchronous operation signal to each of the processing cores connected to the synchronization scheduler.

Further, the program includes a computation instruction and a program update instruction; the processing core is configured to execute the program update instruction based on the completion of the computation instruction, and to generate and send the program update signal based on the program update instruction.

Further, the synchronization scheduler includes a counter, and the counter is configured to record the number of program update signals received; and the synchronization scheduler is further configured to prepare to send the synchronous operation signal in response to the number of the program update signals being equal to the predetermined number.

Further, the synchronization scheduler is further configured to send the configuration signal in response to the number of program update signals being equal to the predetermined number, and then send the synchronous operation signal after receiving a configuration completion signal sent by the direct memory access controller.

Further, the synchronization scheduler receiving the predetermined number of the program update signals includes: the synchronization scheduler receives the predetermined number of the program update signals sent from all the processing cores connected to the synchronization scheduler; or the synchronization scheduler receiving the predetermined number of the program update signals includes: the synchronization

scheduler receives the predetermined number of the program update signals sent from each of the processing cores connected to the synchronization scheduler.

Further, the synchronization scheduler configured to generate and send the configuration signal in response to the program update signal from each of the processing cores includes: the synchronization scheduler is configured to send the configuration signal after receiving the program update signals sent from every processing core connected to the synchronization scheduler.

Further, the at least two processing cores include a first processing core and a second processing core; the first processing core and the second processing core execute different programs, and a computation result of the program executed by the first processing core is an input of the program executed by the second processing core.

According to a second aspect of the present disclosure, there is provided a chip, including one or more data processing apparatuses provided in the first aspect.

According to a third aspect of the present disclosure, there is provided a circuit board, including one or more chips provided in the second aspect.

According to a fourth aspect of the present disclosure, there is provided an electronic device, including one or more circuit boards provided in the third aspect.

According to a fifth aspect of the present disclosure, there is provided a data processing method, including: executing programs by processing cores; generating and sending a configuration signal by a synchronization scheduler in response to a program update signal from each of the processing cores connected to the synchronization scheduler; and reading a program corresponding to each of the processing cores from an external storage unit and sending the program to the corresponding processing core by a direct memory access controller, based on the configuration signal.

According to a sixth aspect of the present disclosure, there is provided a computer storage medium storing a computer program, which, upon execution by a processor, implements the data processing method of the fifth aspect.

According to a seventh aspect of the present disclosure, there is provided an electronic device, including a memory, a processor, and a computer program stored in the memory and executable by the processor, wherein the processor executing the program implements the data processing method of the fifth aspect.

According to an eighth aspect of the present disclosure, there is provided a computer program product, including computer instructions, wherein, when the

computer instructions are executed by a computing device, the computing device implements the data processing method of the fifth aspect.

3. Beneficial Effects

The technical solutions of the present disclosure have the following beneficial technical effects:

The data processing apparatus provided by the embodiment of the present disclosure is provided with a synchronization scheduler and a direct memory access controller, and the direct memory access controller is instructed by the synchronization scheduler to read a program from an external storage unit, and send the program to the corresponding processing core. On the one hand, in the data processing apparatus provided by the embodiment of the present disclosure, the processing cores are not required to fetch data from the external storage unit, which avoids the delay caused by multiple processing cores reading data, and improves the computing power of the chip. On the other hand, in the data processing apparatus provided by the embodiment of the present disclosure, the programs executed by each of the processing cores can be the same or different, and the synchronization scheduler responds to the update of the programs of multiple processing cores, so that the tasks can be allocated flexibly, the computing power of each processing core can be fully utilized, and the computing power of the chip can be further improved.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic structural diagram of a chip provided by the prior art.

FIG. 2 is a schematic structural diagram of a chip provided by another prior art.

FIG. 3 is a schematic structural diagram of a data processing apparatus according to the present disclosure.

FIG. 4 is a schematic structural diagram of another data processing apparatus according to the present disclosure.

FIG. 5 is a schematic structural diagram of a neural network according to the present disclosure.

FIG. 6 is a schematic diagram of computation in the data processing apparatus applied to the neural network according to the present disclosure.

FIG. 7 is a schematic diagram of scheduling of processing cores by a synchronization scheduler according to the present disclosure.

FIG. 8 is a schematic flowchart of a data processing method according to the present disclosure.

## DETAILED DESCRIPTION

In order to make the objective, technical solutions and advantages of the present disclosure clearer, the present disclosure is further described in detail below in conjunction with specific embodiments and with reference to the accompanying drawings. It should be understood that these descriptions are exemplary only and are not intended to limit the scope of the present disclosure. In addition, in the following description, descriptions of well-known structures and techniques are omitted to avoid unnecessary confusion about the concepts of the present disclosure.

In neural network computing, chips with multi-core or many-core architecture are often used. Generally, the processing cores in a chip with a multi-core or many-core architecture all have a certain ability to process data independently, and also have a relatively large internal storage space. The relatively large storage space is generally used to store its own programs, data and weights.

How to efficiently exploit the computing power of the many cores is the key to determining the performance of the entire chip. The utilization of computing power of each core depends on a variety of factors, such as task scheduling and allocation, chip architecture, core structure, and core circuitry. Among them, task scheduling and allocation is a very critical factor. If the scheduling and allocation of tasks are reasonable, the effective computing power of each core can be fully utilized; otherwise, the effective computing power of each core is low.

FIG. 1 is a schematic structural diagram of a chip provided by the prior art.

As shown in FIG. 1, the chip includes a scheduler and a plurality of processing cores C1 to Cn. In the chip shown in FIG. 1, the scheduler receives an instruction sent from outside the chip, for example, the scheduler receives an instruction from an instruction source outside the chip, and then transmits the instruction to each processing core according to a preset strategy (e.g., in a preset order); each processing core executes the same instruction but processes different data. For example, the instruction is to process a+b, but a or b corresponding to two processing cores may be of different values; and then the data processed by the two processing cores are different data.

For the chip architecture shown in FIG. 1, each processing core may have a relatively simple structure, such as Single Instruction Multiple Data (SIMD) structure, or Single Instruction Multiple Threads (SIMT) structure.

Generally, this method has the following disadvantages:

The scheduler can only passively receive instructions from the outside, and then assign the instructions to each processing core; whether it is a SIMD structure or SIMT structure, each processing core can only execute the same instructions, resulting in a single functionality and a lack of flexibility of the chip.

FIG. 2 is a schematic structural diagram of a chip provided by another prior art.

As shown in FIG. 2, the chip includes a plurality of processing cores C1 to Cn and a storage unit Memory. In the chip shown in FIG. 2, each core can independently read instructions from Memory (such as DDR) and perform computations. Usually, each core has a complete control circuit, register group and other circuits. This structure is common in multi-core CPUs or ASICs.

Generally, this method has the following disadvantages:

(1) Each processing core is highly autonomous and is able to run instructions independently, but due to the high autonomy of the processing cores, it is difficult for multiple processing cores to cooperate with each other to accomplish a complete task efficiently.

(2) The circuit control in the chip is relatively complex, and each core is almost a complete CPU. In order to utilize the efficient cooperation among the processing cores to accomplish a complete task, the circuit design is difficult, and the power consumption and chip area are large.

(3) Multiple processing cores may frequently access the instruction storage area, resulting in a decrease in storage access efficiency, which in turn affects the utilization of the chip's computing power.

In order to solve the above problems, the technical solution of the present disclosure is provided.

The chip provided by an embodiment of the present disclosure will be described in detail below. In the description of the present disclosure, it should be noted that the terms "first", "second", "third" and "fourth" are only used for descriptive purposes and cannot be construed as indicating or implying relative importance. In addition, the technical features involved in the different embodiments of the present disclosure

described below can be combined with each other as long as they do not conflict with each other.

FIG. 3 is a schematic structural diagram of a data processing apparatus according to a first embodiment of the present disclosure.

As shown in FIG. 3, the data processing apparatus includes: at least two processing cores, a synchronization scheduler (which may also be referred to as Synchronizer and Scheduler, S_S) and a direct memory access controller (DMAC).

S_S is connected to the at least two processing cores, and the at least two processing cores may be all processing cores in the data processing apparatus, such as processing cores C1 to Cn; DMCA is connected to the at least two processing cores, the S_S, and an external storage unit Memory.

The S_S is for generating and sending a configuration signal in response to a program update signal from each of the processing cores connected to the S_S.

The DAMC is for reading a program corresponding to each of the processing cores from the external Memory and sending the program to the corresponding processing core, based on the configuration signal.

The data processing apparatus provided by the embodiment of the present disclosure is provided with a synchronization scheduler and a direct memory access controller, and the direct memory access controller is instructed by the synchronization scheduler to read a program from an external storage unit, and send the program to the corresponding processing core. On the one hand, in the data processing apparatus provided by the embodiment of the present disclosure, the processing cores are not required to fetch data from the external storage unit, which avoids the delay caused by multiple processing cores reading data, and improves the computing power of the chip. On the other hand, in the data processing apparatus provided by the embodiment of the present disclosure, the programs executed by each of the processing cores can be the same or different, and the synchronization scheduler responds to the update of the programs of multiple processing cores, so that the tasks can be allocated flexibly, the computing power of each processing core can be fully utilized, and the computing power of the chip can be further improved.

In an embodiment, the S_S is further configured to send a synchronous operation signal to the processing cores connected to the S_S in response to the program update signals. The synchronous operation signal is send to each of the processing cores after

the S_S receives a predetermined number of the program update signals, and the synchronous operation signal is used to instruct each of the processing cores to start executing its respective program simultaneously.

Further, the S_S is also configured to first send the configuration signal in response to the number of program update signals being equal to the predetermined number, and then send the synchronous operation signal after receiving a configuration completion signal sent by the direct memory access controller.

In an embodiment, the program includes a plurality of program segments.

The processing cores are configured to send a program update signal to the S_S after executing each of the program segments.

Preferably, the number of program segments executed by each processing core is the same.

Optionally, the S_S includes a first counter, and the first counter is used to record the number of program update signals received.

The S_S is further configured to prepare to send the synchronous operation signal in response to the number of program update signals recorded by the first counter being equal to the predetermined number.

In the embodiment of the present disclosure, the number of the first counters may be, for example, one or more than one.

Optionally, when the number of the first counters is one, the predetermined number may be, for example, the sum of the program update signals from every core received by the S_S, that is, the predetermined number is the product of the number of processing cores connected to the S_S and the number of program segments.

Optionally, the predetermined number recorded by the first counter may be set to one value or multiple values.

For example, the data processing apparatus includes two processing cores, and each processing core executes four program segments.

When the predetermined number recorded by the first counter is set to one value, the predetermined number recorded by the first counter is set to 8. That is, when the first counter records that the number of received program update signals is 8, the S_S sends the configuration signal to the DMAC; and when the S_S receives the configuration completion signal returned by the DMAC, the S_S sends the synchronous

operation signal to each core. At this time, the first counter is cleared and the counting is restarted.

When the predetermined number recorded by the first counter is set to multiple values, the predetermined numbers recorded by the first counter are set to 8, 16, 24...These predetermined numbers are the accumulated numbers of the program update signals received by the synchronization scheduler. When the first counter records that the accumulated number of received program update signals is 8, the S_S sends the configuration signal to the DMAC, and when the DMAC returns the configuration completion signal, the S_S sends the synchronous operation signal to each core. At this time, the counter is not cleared and continues to count. When the counter records that the accumulated number of received program update signals reaches 16, the S_S sends the configuration signal to the DMAC; and when receiving the configuration completion signal returned by the DMAC, the S_S sends the synchronous operation signal to each core. The first counter continues counting, and when the next predetermined number is reached, the S_S repeats the above steps.

Optionally, when the number of the first counters is multiple, the predetermined number may be the number of program segments.

For example, each processing core connected to the S_S corresponds to one first counter, and the first counter is used to record the number of received program update signals sent by the corresponding processing core. After the S_S receives the predetermined number of program update signals sent from each processing core, that is, when each first counter receives the predetermined number of program update signals, the S_S sends the configuration signal to the DMAC; and after receiving the configuration completion signal sent by the DMAC, the S_S sends the synchronous operation signal to each core.

The above method of setting the counters is only an exemplary illustration. Any software and/or hardware structures, which can realize the configuration of DMAC and the synchronous operation of each processing core by the S_S under certain conditions, may be used in the embodiments of the present disclosure, and thus will not be further described here.

In a preferred embodiment, the S_S configured to generate and send the configuration signal in response to the program update signal from each of the processing cores, includes: the S_S is configured to send the configuration signal after

receiving the program update signals sent from every processing core connected to the S_S. For example, if there are five processing cores connected to the S_S, then the S_S sends the configuration signal after receiving the program update signals from all five processing cores.

In a preferred embodiment, the direct memory access controller is used to read a program segment from the external Memory, and send the program segment to the processing core corresponding to the program segment. The external Memory refers to, for example, the storage unit on the Host.

The direct memory access controller is further configured to send the configuration completion signal to the S_S after sending the program segments to the processing cores corresponding to the program segments.

In an embodiment, the S_S further configured to send the synchronous operation signal to each of the processing cores connected to the S_S in response to the program update signals, includes:

the S_S is further configured to, in response to the program update signals and the configuration completion signal, send the synchronous operation signal to each of the processing cores connected to the S_S.

Specifically, the S_S is configured to first send the configuration signal to the DMAC in response to the number of program update signals recorded by the counter being equal to the predetermined number, and the DMAC sends the configuration completion signal to the S_S after sending the programs or the program segments indicated by the configuration signal to the corresponding processing cores. The S_S sends the synchronous operation signal to each of the processing cores connected to the S_S after receiving the configuration completion signal.

Here, the "program indicated by the configuration signal" may be the same program as or a different program from the program just executed by the processing core.

Further, the program includes a computation instruction and a program update instruction; the processing core is configured to execute the program update instruction after completing the computation instruction, and to generate and send the program update signal based on the program update instruction.

When the program executed by the processing core includes a plurality of program segments, each program segment includes a computation instruction and a program update instruction.

In an embodiment, the processing core is provided with a storage module PRAM, which is used to store and receive the program sent by the DMAC, and the program executed by the processing core is read from its own PRAM. In the embodiment, each processing core reads the instructions contained in the program from its own PRAM without the need to read from the external Memory, and thus complex cache circuit designs can be avoided. Compared with the prior art, it is not necessary to read data from the Memory, which reduces latency and greatly improves the execution efficiency of instructions.

Preferably, the storage space of the PRAM is greater than or equal to 16KB.

Optionally, the programs stored by any two processing cores are the same or different.

In a preferred embodiment, the at least two processing cores in the data processing apparatus include a first processing core and a second processing core; the first processing core and the second processing core execute different programs, and a computation result of the program executed by the first processing core may be an input of the program executed by the second processing core. Using the computation result of the program executed by the first processing core as the input to the second processing core enables the chip provided by the embodiment to be used for the computation of a neural network. And through the S_S, each processing core can execute its own stored program simultaneously, so that each processing core can exchange data in an orderly manner, and cooperate with each other to accomplish a complete task efficiently.

In the data processing apparatus provided by the embodiment of the present disclosure, by using the S_S and DMA to allocate programs to each of the processing cores, the program executed by each processing core and the exchange of data have been set before the programs run. The micro control unit (MCU) on the top layer of the chip or the Host of the system is only required to configure the counter of S_S to achieve the established strategy; and the MCU or the Host of the system may change the program executed by each core as well as the allocation and scheduling of the programs by changing the configuration of the counter of S_S and the stored program in the

Memory, so that it is convenient to modify the allocation and scheduling of tasks for each processing core in the chip, and is able to efficiently utilize the computing power of the processing cores.

According to another aspect of the present disclosure, there is provided a chip, including one or more data processing apparatuses provided in the above aspects.

For example, when the chip includes multiple data processing apparatuses, the chip may include multiple S_S, and each S_S is connected to a plurality of processing cores and a direct memory access controller.

According to yet another embodiment of the present disclosure, there is provided a circuit board, including one or more chips provided by the above embodiments.

According to yet another embodiment of the present disclosure, there is provided an electronic device, including one or more circuit boards provided by the above embodiments.

FIG. 4 is a schematic structural diagram of a data processing apparatus according to the present disclosure.

As shown in FIG. 4, the apparatus includes a first processing core C1, a second processing core C2, S_S and DMAC. Each processing core is provided with PRAM.

FIG. 5 is a schematic structural diagram of a neural network according to the present disclosure.

The embodiment takes a 2-layer neural network as an example. As shown in FIG. 5, the neural network has a 2-layer structure. The control program of each layer of the neural network is 128KB, and the computational load of each layer is the same; the network may be allocated to two processing cores for pipelined computation according to an equally divided task allocation strategy, that is, C1 and C2 are each responsible for the computation of a layer of the network, and each runs the program of the corresponding layer. For example, C1 computes the first layer of the network Layer1, and C2 computes the second layer of the network Layer2. The input data is sent to C1. C1 performs processing of the first layer on the input data, and sends the processing result of the first layer to C2; C2 takes the processing result of the first layer as input, performs processing of the second layer, and obtains the final result and then outputs. That is, the data flows through Layer1 and Layer2 in turn to achieve the computation of the entire neural network, and finally get the output.

FIG. 6 is a schematic diagram of computation in the chip applied to the neural network according to the present disclosure.

As shown in FIG. 6, Input1-1 represents the input of the entire neural network, and also represents the input of the first layer of neural network layer1 at the starting point of the time period t1, and Input2-1 represents the computation result of layer1 in the time period t1, and also represents the input of the second layer of neural network layer2 at the starting point of the time period t2; output1 represents the computation result of layer2 after the time period t2, and the computation result is also the output result of the neural network. When C1 processes the pipeline of layer1, its input is the data input of the neural network, and its output is used as the input of C2. The output of C2 is the final output result.

The control program of each layer is set to 128 KB. Since the PRAM of each processing core is only 32 KB, the control program of each layer needs to be scheduled according to a certain strategy to update the program of each core. For example, the program of each core can be equally divided into four program segments to be transferred to the corresponding core, with each transfer of 32KB.

The computation process of the processing cores in the chip when running the neural network is as follows.

At the beginning, C1 receives the first program segment of the first program and executes the first program segment at time t1. After executing the first program segment, C1 sends the program update instruction PU_S1 to the S_S. Since the output of C1 is the input of C2, it can be set that when C1 runs the program segments of the first program, the S_S sends a configuration signal to the DMAC each time it receives the program update signal sent by C1, until C1 runs the last program segment of the first program and sends the configuration signal to the DMAC. The DMAC then reads the first segment of the second program run by C1 and the first segment of the first program run by C2 from the external memory, and sends them to C1 and C2. When the sending is completed, the DMAC sends a configuration completion signal to the S_S, and the S_S sends a synchronous operation signal Sync to both C1 and C2, instructing C1 and C2 to start running the received program segments at the same time.

When C1 or C2 finishes running the computation instruction of the program segment, it executes the program update instruction PUpdate, which sends the generated program update signal PU_s to the S_S, where PU_s indicates that the

program in the PRAM needs to be updated; after the S_S receives the signal, it determines whether it has received all PU_s sent by all processing cores connected to the S_S, and if the S_S does not receive the PU_s from all the processing cores connected to the S_S, it is in a waiting state until it receives the PU_s from all the processing cores connected to the S_S. If the S_S receives the PU_s sent by all the processing cores connected to the S_S, it sends the configuration signal to the DMAC, and the DMAC fetches the program segments indicated by the configuration signal from the external Memory and sends the program segments to the processing cores corresponding to the program segments, so as to update the PRAM of each core.

After the S_S receives the program update signals sent by all the processing cores connected to the S_S for the fourth time, indicating that all program segments of each layer of each core have been executed, the S_S first sends the configuration signal to the DMAC. According to the configuration signal, the DMCA reads the first program segments of the programs indicated by the configuration signal from the external Memory and sends them to the corresponding processing cores. After the sending is completed, the DMAC sends a configuration completion signal.

When the S_S receives the configuration completion signal from the DMAC, it generates and sends the synchronous operation signal Sync, which instructs that the program of each core needs to start execution at the same time. After receiving the synchronous operation signal, the cores can start to exchange data, that is, C1 sends the computation result to C2.

It should be noted that the program indicated by the configuration signal may be the same as or different from the program just completed execution.

It should also be noted that at the beginning, C1 may also be set to receive the first program segment of the first program at time t1, and C2 also receives the first program segment of the first program at time t1; both C1 and C2 execute the first program segment. The input of the first program segment of the first program run by C2 may be set to a preset value, so that both C1 and C2 execute the first program at the same time.

FIG. 7 is a schematic diagram of scheduling of processing cores by a synchronization scheduler according to the present disclosure.

As shown in FIG. 7, C1 and C2 receive the signal Sync sent by the S_S, and simultaneously starts running the program stored in the PRAM from the beginning. After C1 completes execution of the computation instruction in the first program

segment, it executes the last instruction in the first program segment, that is, the update instruction PUpdate, indicating that the program segment has been executed. PUpdate generates the program update signal PU_s1 and sends the update signal to the S_S, and then C1 starts to wait.

The S_S receives PU_s1 and finds that it has not received the update signals sent by every processing core connected to the S_S, that is, PU_s2 has not received for this time, and the S_S continues to wait.

After C2 completes execution of the computation instruction in the first program segment, it executes the last instruction in the first program segment, that is, the update instruction PUpdate, indicating that the program segment has been executed. PUpdate generates the program update signal PU_s2 and sends it to the S_S, and then C2 starts to wait.

The S_S receives PU_s2, and the first counter finds that it receives the program update signals sent by every processing core connected to the S_S; and the S_S configures the DMAC to start the program update for C1 and C2.

The DMAC reads out the new program segments of C1 and C2 from the Memory, and then sends them to the PRAMs of C1 and C2 respectively, until the new program segments have been updated.

When the first counter of the S_S records that the number of program updates received is 8, which indicates that the two processing cores have completed their respective four program segments, that is, the entire programs of the two processing cores have been executed, and the S_S resets the first counter to record the number of updates again from the beginning. The S_S first configures the DMAC, to start the program updates for C1 and C2, that is, to reload the first program segments of the next programs. When the DMAC finishes sending the program segments, it sends the configuration completion signal to the S_S; and the S_S generates the synchronous operation signal Sync and sends it to each core, instructing each core to start working at the same time, and the cores can transmit data to each other at this time.

It should be noted that the reset of the first counter, the configuration of DMAC and the transmission of the synchronous operation signal may be performed in any order.

All of the above are described in terms of a two-layer neural network with the computation result of C1 taken as the input data of C2. Of course, the data processing

apparatus provided in the embodiment of the present disclosure may be used in any neural network, and there may be no data association between C1 and C2.

FIG. 8 is a schematic flowchart of a data processing method according to the present disclosure.

As shown in FIG. 8, the data processing method includes:

Step S101, executing programs by processing cores;

Step S102, generating and sending a configuration signal by a synchronization scheduler in response to a program update signal from each of the processing cores connected to the synchronization scheduler; and

Step S103, reading a program corresponding to each of the processing cores from an external storage unit and sending the program to the corresponding processing core by a direct memory access controller, based on the configuration signal.

In an embodiment, the synchronization scheduler further sends the synchronous operation signal to each of the processing cores connected to the S_S in response to the program update signals, and the synchronous operation signal is used to instruct each of the processing cores connected to the synchronization scheduler to start executing its respective program simultaneously.

Specifically, the S_S further configured to send the synchronous operation signal to each of the processing cores connected to the S_S in response to the program update signals, includes: the S_S sends the synchronous operation signal to each of the processing cores connected to the S_S when the S_S receives the predetermined number of the program update signals.

In an embodiment, the program executed by the processing core includes a plurality of program segments.

The processing cores send the program update signal to the synchronization scheduler after executing each of the program segments.

Preferably, the number of program segments executed by each processing core is the same.

Preferably, the first counter of the S_S records the number of the program update signals received.

The S_S prepares to send the synchronous operation signal in response to the number of the program update signals recorded by the first counter being equal to the predetermined number.

In the embodiment of the present disclosure, the number of the first counters may be, for example, one or more than one.

Optionally, when the number of the first counters is one, the predetermined number may be, for example, the sum of the program update signals from every core received by the S_S, that is, the predetermined number is the product of the number of processing cores connected to the S_S and the number of program segments.

Optionally, the predetermined number recorded by the first counter may be set to one value or multiple values. For example, the data processing apparatus includes two processing cores, and each processing core executes four program segments.

When the predetermined number recorded by the first counter is set to one value, the predetermined number recorded by the first counter is set to 8. That is, when the first counter records that the number of received program update signals is 8, the S_S sends the configuration signal to the DMAC; and when the S_S receives the configuration completion signal returned by the DMAC, the S_S sends the synchronous operation signal to each core. At this time, the first counter is cleared and the counting is restarted.

When the predetermined number recorded by the first counter is set to multiple values, the predetermined numbers recorded by the first counter are set to 8, 16, 24...These predetermined numbers are the accumulated numbers of the program update signals received by the synchronization scheduler. When the first counter records that the accumulated number of received program update signals is 8, the S_S sends the configuration signal to the DMAC, and when the DMAC returns the configuration completion signal, the S_S sends the synchronous operation signal to each core; at this time, the counter is not cleared and continues to count. When the counter records that the accumulated number of received program update signals reaches 16, the S_S sends the configuration signal to the DMAC; and when receiving the configuration completion signal returned by the DMAC, the S_S sends the synchronous operation signal to each core. The first counter continues counting, and when the next predetermined number is reached, the S_S repeats the above steps.

Optionally, when the number of the first counters is multiple, the predetermined number may be the number of program segments.

For example, each processing core connected to the S_S corresponds to one first counter, and the first counter is used to record the number of received program update

signals sent by the corresponding processing core. After the S_S receives the predetermined number of program update signals sent from each processing core, that is, when each first counter receives the predetermined number of program update signals, the S_S sends the configuration signal to the DMAC; and after receiving the configuration completion signal sent by the DMAC, the S_S sends the synchronous operation signal to each core.

Optionally, the S_S generating and sending the configuration signal in response to the program update signal of each of the processing cores includes: the S_S sends the configuration signal after receiving the program update signals sent from all the processing cores connected to the S_S. For example, if there are five processing cores connected to the synchronization scheduler, then the synchronization scheduler sends the configuration signal after receiving the program update signals from all five processing cores.

Optionally, the DMAC reads the program segment corresponding to each of the processing cores from the external Memory, and sends the program segments to the corresponding processing cores. The DMAC sends a configuration completion signal to the S_S after sending the program segments to the corresponding processing cores.

In an embodiment, the S_S sending the synchronous operation signal to each of the processing cores connected to the S_S in response to the program update signals, includes:

the S_S is further configured to, in response to the program update signals and the configuration completion signal, send the synchronous operation signal to each of the processing cores connected to the S_S.

Specifically, the S_S first sends the configuration signal to the DMAC in response to the number of the program update signals recorded by the counter being equal to the predetermined number, and the DMAC sends the configuration completion signal to the S_S after sending the programs or the program segments indicated by the configuration signal to the corresponding processing cores. The S_S sends the synchronous operation signal to each of the processing cores connected to the S_S after receiving the configuration completion signal.

In an embodiment, the program includes a computation instruction and a program update instruction; the processing core executes the program update instruction after completing the computation instruction, and generates and sends the program update

signal based on the program update instruction. Completing the computation instruction refers to when the computation instruction is completed or after the computation instruction is completed.

In an embodiment, the at least two processing cores include a first processing core and a second processing core; the first processing core and the second processing core execute different programs, and a computation result of the program executed by the first processing core is an input of the program executed by the second processing core.

According to yet another embodiment of the present disclosure, there is provided a computer storage medium storing a computer program, which, upon execution by a processor, implements the data processing method provided by the above embodiment.

According to yet another embodiment of the present disclosure, there is provided an electronic device, including a memory, a processor, and a computer program stored in the memory and executable by the processor, wherein the processor executing the program implements the data processing method provided by the above embodiment.

According to a further aspect of the present disclosure, there is provided a computer program product, including computer instructions, wherein when the computer instructions are executed by a computing device, the computing device implements the data processing method provided by the above embodiment.

In the data processing method provided by the embodiment of the present disclosure, the direct memory access controller is instructed by the synchronization scheduler to read the program from the external storage unit, and send the program to the corresponding processing core. On the one hand, the processing cores are not required to fetch data from the external storage unit, which avoids the delay caused by multiple processing cores reading data, and improves the computing power of the chip. On the other hand, the programs executed by each of the processing cores can be the same or different, and the synchronization scheduler responds to the update of the programs of multiple processing cores, so that the tasks can be allocated flexibly, the computing power of each processing core can be fully utilized, and the computing power of the chip can be further improved.

It should be understood that the above specific embodiments of the present disclosure are used only for exemplary illustration or explanation of the principle of the present disclosure, but not to limit the present disclosure. Therefore, any modifications, equivalent replacements, improvements, etc. made without departing from the spirit

and scope of the present disclosure should be included within the protection scope of the present disclosure. Furthermore, the appended claims of this disclosure are intended to cover all variations and modifications that fall within the scope and boundaries of the appended claims, or the equivalents of such scope and boundaries.

## CLAIMS

1. A data processing apparatus, comprising:

at least two processing cores;

a synchronization scheduler, configured to generate and send a configuration signal in response to a program update signal from each of the processing cores connected to the synchronization scheduler; and

a direct memory access controller, configured to read a program corresponding to each of the processing cores from an external storage unit and send the program to the corresponding processing core, based on the configuration signal.

2. The data processing apparatus according to claim 1, wherein

the synchronization scheduler is further configured to, in response to the program update signal, send a synchronous operation signal to each of the processing cores connected to the synchronization scheduler, wherein the synchronous operation signal is sent to each of the processing cores after the synchronization scheduler receives a predetermined number of the program update signals, and the synchronous operation signal is configured to instruct each of the processing cores to start executing its respective program simultaneously.

3. The data processing apparatus according to claim 1 or 2, wherein

the program comprises a plurality of program segments.

4. The data processing apparatus according to claim 3, wherein

the processing cores are configured to send the program update signal to the synchronization scheduler after executing each of the program segments.

5. The data processing apparatus according to claim 3 or 4, wherein

the direct memory access controller is configured to read the program segment corresponding to each of the processing cores from the external storage unit, and send the program segments to the corresponding processing cores; and

the direct memory access controller is further configured to send a configuration completion signal to the synchronization scheduler after sending the program segments to the corresponding processing cores.

6. The data processing apparatus according to claim 5, wherein

the synchronization scheduler is further configured to, in response to the program update signal and the configuration completion signal, send the synchronous operation signal to each of the processing cores connected to the synchronization scheduler.

7. The data processing apparatus according to any of claims 2-6, wherein the synchronization scheduler comprises a counter configured to record the number of the program update signals received; and

the synchronization scheduler is further configured to prepare to send the synchronous operation signal in response to the number of the program update signals being equal to the predetermined number.

8. The data processing apparatus according to any of claims 2-7, wherein

the number of program segments executed by each of the processing cores is the same; and

the predetermined number is the number of the program segments, or the predetermined number is the product of the number of the processing cores connected to the synchronization scheduler and the number of the program segments.

9. The data processing apparatus according to any of claim 1-8, wherein the at least two processing cores comprise a first processing core and a second processing core; and

the first processing core and the second processing core execute different programs, and a computation result of the program executed by the first processing core is an input of the program executed by the second processing core.

10. A data processing method, comprising:

executing programs by processing cores;

generating and sending a configuration signal by a synchronization scheduler in response to a program update signal from each of the processing cores connected to the synchronization scheduler; and

reading a program corresponding to each of the processing cores from an external storage unit and sending the program to the corresponding processing core by a direct memory access controller, based on the configuration signal.