ACCELERATOR, ACCELERATION METHOD, AND ELECTRONIC DEVICE

TECHNICAL FIELD

[0001] The present application relates to the computer technology and, in particular, to an accelerator, an acceleration method, and an electronic device.

BACKGROUND

[0002] ECC (Elliptic Curves Cryptography) is a main trend asymmetric cryptography algorithm that is widely apply to many security fields, such as a secure transmission, a key exchange, a digital signature, a privacy calculation, a secure multi-party calculation, etc. The ECC algorithm includes a dot multiplication, a modular addition, a modular multiplication, a modular subtraction and other operations. The dot multiplication operation can be implemented by several modular additions, modular multiplications, modular subtractions, etc. The ECC algorithm is currently implemented through an accelerator.

[0003] Related art 1: each calculation link of the accelerator is strictly controlled by a state machine to implement a specific calculation process to achieve a specific function. Since control and modification of the state machine is relatively cumbersome, these accelerators can only achieve the specific function and are not flexible enough to use.

[0004] Related art 2: different calculating units in the accelerator are called through different instructions to implement operations such as the dot multiplication, the modular multiply, and the modular addition respectively, which leads to a waste of a hardware resource.

SUMMARY

[0005] Based on this, the present application at least discloses an accelerator. The accelerator includes a calculating unit configured to implement an operation related to an elliptic curves algorithm; where the calculating unit includes a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs; the accelerator further includes: an acquiring unit, configured to acquire a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; where the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation; a parsing unit, configured to parse the target microcode instruction to obtain a parsing result corresponding to the target input data; and a controlling unit, configured to, in response to the parsing result, control the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

[0006] In some embodiments, the accelerator is communicatively connected to a central processing unit (CPU); the accelerator or the CPU is configured to: acquire an operation program that performs the target type operation on an operand; and generate the target microcode instruction corresponding to the target type operation according to the target type operation based on the operand included in the operation program.

[0007] In some embodiments, the accelerator further includes a data storage space; the data storage space stores a preset operand used to generate a microcode instruction; the accelerator or the CPU is further configured to: store the operand obtained by parsing in the data storage space; and generate the target microcode instruction according to the target type operation based on a storage address of the operand and/or the preset operand in the data storage space.

[0008] In some embodiments, the parsing result includes a storage address of the target input data; the controlling unit is configured to control, according to the storage address, the calculating unit to read the target input data from the data storage space

and perform the target type operation based on the target input data.

[0009] In some embodiments, the basic operation logic includes a modular addition operation and a modular multiplication operation, the general operation logic is to perform a modular multiplication on a calculation result of two modular addition operations.

[0010] In some embodiments, an input required by the general operation logic includes a first input, a second input, a third input and a fourth input; an operation process of the general operation logic includes: performing a modular addition operation on the first input and the second input to obtain a first modular addition result; performing a modular addition operation on the third input and the fourth input to obtain a second modular addition result; and performing a modular multiplication operation on the first modular addition result and the second modular addition result to obtain a final calculation result.

[0011] In some embodiments, the operand includes a first operand and a second operand; the accelerator or the CPU is further configured to: in response to the target type operation being a modular addition operation on the first operand and the second operand, set the first input to the first operand, set the second input to the second operand, set the third input to 1 and set the fourth input to 0; in response to the target type operation being a modular subtraction operation on the first operand and the second operand, set the first input to the first operand, set the second input to the second operand and adjust a symbol corresponding to the second input to a negative sign, set the third input to 1 and set the fourth input to 0; in response to the target type operation being a modular multiplication operation on the first operand and the second operand, set the first input to the first operand, and set the third input to the second operand, and set the second input and the fourth input to 0; in response to the target type operation being a modular addition squared operation on the first operand and the second operand, set the first input and the third input to the first operand, and set the second input and the fourth input to the second operand; and generate the target microcode instruction according to the first input, the second input, the third input and the fourth input.

[0012] In some embodiments, the operation program further includes a dot

multiplication scalar; the target type operation is a dot multiplication operation on the dot multiplication scalar and the operand; the accelerator or the CPU is further configured to: generate a plurality of target microcode instructions for implementing the dot multiplication operation according to the dot multiplication scalar and the operand included in the operation program; the accelerator is further configured to: control the calculating unit to execute each target microcode instruction in the plurality of target microcode instructions, so as to complete the dot multiplication operation.

[0013] In some embodiments, the modular multiplication operation includes a modular multiplication operation of a first type and a modular multiplication operation of a second type; the accelerator further includes a data storage space; the data storage space stores a preset constant; the preset constant is used to mutual convert between a first data field and a second data field; the first data field is used for the modular multiplication operation of the first type; the second data field is used for the modular multiplication operation of the second type; the calculating unit is configured to, in response to the modular multiplication operation converting from the first type to the second type, acquire the preset constant and convert the first data field into the second data field according to the preset constant to implement the modular multiplication operation of the second type; in response to the modular multiplication operation converting from the second type to the first type, acquire the preset constant and convert the second data field into the first data field according to the preset constant to implement the modular multiplication operation of the first type.

[0014] In some embodiments, the controlling unit includes a breakpoint subunit; the breakpoint subunit is configured to interrupt the calculating unit to execute subsequent microcode instruction in response to reception of a interrupt instruction, so as to enable a debugging unit corresponding to the accelerator to acquire a calculation result of the calculating unit, and output a comparison result between the calculation result and a preset calculation result.

[0015] In some embodiments, the accelerator is a coprocessor implemented based on FPGA or ASIC for accelerating the elliptic curves algorithm; or, an auxiliary accelerating module implemented based on CPU for performing an algorithm

acceleration for the elliptic curves algorithm.

[0016] The present application also proposes an accelerating method, applied to the accelerator. the accelerator includes a calculating unit configured to implement an operation related to an elliptic curves algorithm; where the calculating unit includes a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs; the methods includes: acquiring a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; where the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation; parsing the target microcode instruction to obtain a parsing result corresponding to the target input data; and in response to the parsing result, controlling the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

[0017] The present application also proposes an electronic device, the electronic device includes the accelerator as shown in any one of the above-mentioned embodiments.

[0018] In the above-mentioned method, first, the target microcode instruction can be generated according to the target type operation to be implemented, and the relevant configuration of the input required for the general operation logic to implement the target type operation in the calculating unit is completed according to the target microcode instruction, so as to enable the accelerator to complete the target type operation. Compared with related art 1, the accelerator can perform different type operations by changing the microcode instruction, which simplifies the configuration operation on the accelerator and reduces the difficulty of configuring the accelerator, thereby the accelerator can be used more flexible.

[0019] Second, the general operation logic composed of the basic operation logic is deployed in the calculating unit; the general operation logic perform the operation of different operation types in response to different inputs. Compared with related art 2, at least two types of operations can be implement by one calculating unit, thereby saving

the hardware resource.

[0020] It should be understood that the above-mentioned general description and the following detailed description are exemplary and explanatory only, and do not limit the present application.

BRIEF DESCRIPTION OF DRAWINGS

[0021] To describe the technical solutions in one or more embodiments of the present application or related arts more clearly, the following briefly introduces the accompanying drawings needed for describing the embodiments or the related arts. Apparently, the accompanying drawings in the following description illustrate merely some embodiments recorded in one or more embodiments of the present application, and persons of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative effort.

[0022] FIG. 1 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0023] FIG. 2 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0024] FIG. 3 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0025] FIG. 4 is a schematic flowchart of an accelerating method shown in an embodiment of the present application.

DESCRIPTION OF EMBODIMENTS

[0026] Exemplary embodiments will be described in detail below, examples of which are illustrated in the accompanying drawings. When the following description refers to the drawings, same number in different drawings refer to the same or similar element unless otherwise indicated. The implementations described in the following exemplary embodiments do not represent all implementations consistent with the

present application. On the contrary, they are merely examples of devices and methods consistent with some aspects of the present application as detailed in the appended claims.

[0027] The term used in the present application is merely for the purpose of describing particular embodiments and is not intended to limit the present application. As used in the present application and the appended claims, the singular forms "a", "the" and "this" are intended to include the plural forms as well, unless the context clearly dictates otherwise. It should also be understood that the term "and/or" as used herein refers to and includes any and all possible combinations of one or more of the associated listed items. It should also be understood that the word "if" as used herein, depending on the context, may be interpreted as "based on that……" or "when……" or "in response to determination of".

[0028] Based on this, the present application proposes an accelerator. The accelerator includes a calculating unit configured to implement an operation related to an elliptic curves algorithm; where the calculating unit includes a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs; the accelerator further includes: an acquiring unit, configured to acquire a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; where the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation; a parsing unit, configured to parse the target microcode instruction to obtain a parsing result corresponding to the target input data; and a controlling unit, configured to, in response to the parsing result, control the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

[0029] In the above-mentioned method, first, the target microcode instruction can be generated according to the target type operation to be implemented, and the relevant configuration of the input required for the general operation logic to implement the target type operation in the calculating unit is completed according to the target

microcode instruction, so as to enable the accelerator to complete the target type operation. Compared with related art 1, the accelerator can perform different type operations by changing the microcode instruction, which simplifies the configuration operation on the accelerator and reduces the difficulty of configuring the accelerator, thereby the accelerator can be used more flexible.

[0030] Second, the general operation logic composed of the basic operation logic is deployed in the calculating unit; the general operation logic perform the operation of different operation types in response to different inputs. Compared with related art 2, at least two types of operations can be implement by one calculating unit, thereby saving the hardware resource.

[0031] Reference please be made to FIG. 1, FIG. 1 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0032] As shown in FIG. 1, the accelerator 10 may include an acquiring unit 11, a parsing unit 12, a controlling unit 13, and a calculating unit 14 connected in sequence. Where the calculating unit 14 includes a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs.

[0033] The accelerator can implement ECC acceleration based on a combination of hardware and software. Various functional units included in the accelerator may be a hardware unit, a software apparatus unit, or a unit combined a software apparatus and a hardware.

[0034] In some embodiments, the accelerator is implemented based on a coprocessor or a CPU; where the accelerator is a coprocessor implemented based on FPGA or ASIC for accelerating an elliptic curves algorithm; or, an auxiliary accelerating module implemented based on CPU for performing an algorithm acceleration for the elliptic curves algorithm.

[0035] The acquiring unit 11 can acquire a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type

operation.

[0036] The target type operations may be operations related to the elliptic curves algorithm. In some embodiments, the target type operation may include at least one of the following:

a modular addition; a modular subtraction; a modular multiplication; a modular addition squared; a modular subtraction squared; a modular addition multiplying modular subtraction; a modular addition multiplying modular addition; and a modular subtraction multiplying modular subtraction.

[0037] The target microcode instruction is used to trigger the accelerator 10 to complete the target type operation. Since the simplicity of logic and ease of writing, the microcode instruction can simplify the development difficulty for developers.

[0038] The target microcode instructions may indicate the target input data in at least two manners as following. First, the target microcode instruction directly carries the target input data; second, the target microcode instruction carries a storage address of the target input data.

[0039] In some manners, the target microcode instruction may be a program instruction manually written according to a requirement. The target microcode instruction can be sent to the acquiring unit 11 via a configuration bus after being encoded by an encoder. The accelerator may acquire the target microcode instruction through the acquiring unit 11.

[0040] In some manners, the target microcode instruction can be an instruction obtained after performing a conversion operation on an operation program in a non-manual manner. The operation program is a code program written according to requirement. The conversion operation may be performed in the accelerator 10 or the CPU communicatively connected to the accelerator. After completing the conversion operation, the target microcode instruction may be sent to the acquiring unit 11.

[0041] The accelerator or the CPU may be configured to: acquire an operation program that performs the target type operation on an operand; and generate the target microcode instruction corresponding to the target type operation according to the target type operation based on the operand included in the operation program. Thereby, a

conversion operation is performed on the operation program in a non-manual manner to generate a target operation operation that can enable the accelerator to perform the target type operation, which eliminates the need for developers to understand the internal operation logic of the accelerator, simplifies the difficulty of developing code, and thus improves the practicality of the accelerator.

[0042] In some manners, a configuration rule of the input data required by the calculating unit to implement each type of operation can be maintained in advance. When performing the conversion operation, a corresponding target configuration rule can be determined according to the target type operation, and then the input data is configured for the target microcode instruction according to the target configuration rule and the operand to generate the target microcode instruction.

[0043] The parsing unit 12 can parse the target microcode instruction and obtain a parsing result corresponding to the target input data.

[0044] In some embodiments, the acquiring unit 11 can store the acquired target microcode instructions in a preset instruction storage space. The parsing unit 12 can acquire an unexecuted target microcode instruction from the instruction storage space, and perform relevant parsing to obtain the parsing result.

[0045] If the target microcode instruction directly carries target input data, the parsing result includes the target input data.

[0046] If the target microcode instruction carries the storage address of the target input data, the parsing result includes the storage address.

[0047] In some embodiments, the parsing unit 12 can also parse an instruction type of the target microcode instruction to assist the controlling unit 13 to execute the microcode instruction. For example, some microcode instructions may be data movement (MOV) instructions. At this time, the controlling unit 13 only need to execute the data movement operation, and does not need to control the calculating unit to perform a calculation. For another example, some microcode instructions may involve a subtraction operation (such as the modular subtraction, the modular addition multiplying modular subtraction, the modular subtraction squared, etc.), and the controlling unit 13 can control the calculating unit to adjust the symbol of the acquired

input data to a negative sign.

[0048] The controlling unit 13 can, in response to the parsing result, control the calculating unit 14 to enable the calculating unit 14 to acquire the target input data and run the general operation logic based on the target input data to complete the target type operation.

[0049] If the parsing result includes the target input data, the calculating unit can acquire the target input data from the parsing result.

[0050] If the parsing result includes a storage address of the target input data, the calculating unit can read the target input data from a data storage space according to the storage address.

[0051] In some embodiments, the basic operation logic includes a modular addition operation and a modular multiplication operation, the general operation logic is to perform a modular multiplication on a calculation result of two modular addition operations. Thereby at least two operations can implement through one general operation logic.

[0052] For example, the general operation logic is expressed as ((A+B)mod(p)* (C+D)mod(p))mod(p). If C is set to 0 and D is set to 1, the general operation logic can become (A+B)mod(p), which is equivalent to perform the modular addition operation on A and B. If B and D are set to 0, the general operation logic can become (A*C)mod(p), which is equivalent to perform the modular multiplication operation on A and C.

[0053] In some embodiments, the basic operation logic includes the modular addition operation and the modular multiplication operation, an input required by the general operation logic includes a first input, a second input, a third input and a fourth input;

an operation process of the general operation logic includes:

performing a modular addition operation on the first input and the second input to obtain a first modular addition result;

performing a modular addition operation on the third input and the fourth input to obtain a second modular addition result; and

performing a modular multiplication operation on the first modular addition result and the second modular addition result to obtain a final calculation result.

[0054] Through the above-mentioned design of the calculating unit, the calculating unit can respond to the input data indicated by different microcode instructions and implement the following at least two types of operations by running the general operation logic: a modular addition; a modular subtraction; a modular multiplication; a modular addition squared; a modular subtraction squared; a modular addition multiplying modular subtraction; a modular addition multiplying modular addition; and a modular subtraction multiplying modular subtraction. This allows one calculating unit to perform multiple types of operations, thereby improving hardware utilization, and saving hardware resource.

[0055] It should be noted that solutions obtained by simply deforming the above-mentioned general operation logic are also within the protection scope of the present application.

[0056] The process of the calculating unit implementing different types of operations is further described below. It is assumed that the operand corresponding to each operation includes a first operand and a second operand, and the target microcode instruction are is based on the CPU or the accelerator.

[0057] In order to simplify the explanation, the formula ((A+B)mod(p)*(C+D)mod(p))mod(p) is used to represent the above-mentioned general operation logic of the calculating unit.

[0058] A, B, C, and D indicate the first input to the fourth input respectively. p is a data field, which means that ABCD are all in the range of 0 to p. mod(p) indicates modulo p. (A+B)mod(p) indicates to perform the modular addition operation on A and B, that is, the sum of A and B modulo p. ((A+B)mod(p)*(C+D)mod(p))mod(p) can indicate: to perform the modular addition on A and B, to perform the modular addition on C and D, then perform the modular multiply to both of these modular addition results.

[0059] In a first operation situation, the target type operation is a modular addition operation on the first operand and the second operand.

[0060] The accelerator or the CPU is further configured to:

in response to the target type operation being a modular addition operation on the first operand and the second operand, set the first input to the first operand, set the second input to the second operand, set the third input to 1 and set the fourth input to 0; and

generate the target microcode instruction according to the first input, the second input, the third input and the fourth input. It can be understood that the target microcode instruction can include the above four inputs (including the first input, the second input, the third input and the fourth input), or the storage addresses of the above four inputs.

[0061] The calculating unit 14 is further configured to:

acquire a first input, a second input, a third input and a fourth input;

perform a modular addition operation on the first input and the second input to obtain a first modular addition result;

perform a modular addition operation on the third input and the fourth input to obtain 1; and

perform a modular multiplication operation on the first modular addition result and 1 to obtain a first modular addition result, so as to complete the modular addition operation for the operand.

[0062] Taking the above formula as an example, where A is the first operand, B is the second operand, C=1, D=0, ((A+B)mod(p)*(C+D)mod(p))mod(p)=(A+B)mod(p) is equivalent to that the calculating unit realizes the modular addition operation on the first operand and the second operand.

[0063] In the second operation situation, the target type operation is a modular subtraction operation on the first operand and the second operand.

[0064] the accelerator or the CPU is further configured to:

in response to the target type operation being a modular subtraction operation on the first operand and the second operand, set the first input to the first operand, set the second input to the second operand and adjust a symbol corresponding to the second input to a negative sign, set the third input to 1 and set the fourth input to 0; and

generate the target microcode instruction according to the first input, the second input, the third input and the fourth input.

[0065] The calculating unit 14 is further configured to:

acquire a first input, a second input, a third input and a fourth input;

adjust the symbol corresponding to the second input to a negative sign in response to the target type operation being the modular subtraction operation.

perform a modular addition operation on the first input and the second input to obtain a first modular addition result;

perform a modular addition operation on the third input and the fourth input to obtain 1; and

perform a modular multiplication operation on the first modular addition result and 1 to obtain the first modular addition result, so as to complete the modular subtraction operation for the operand.

[0066] Taking the above formula as an example, where A is the first operand, B is the second operand, C=1, D=0, the symbol of B is "-", ((A+B)mod(p)*(C+D)mod(p))mod(p)=(A-B)mod(p) is equivalent to that the calculating unit realizes the modular subtraction operation on the first operand and the second operand.

[0067] In the third operation situation, the target type operation is a modular multiplication operation on the first operand and the second operand.

[0068] the accelerator or the CPU is further configured to:

in response to the target type operation being a modular multiplication operation on the first operand and the second operand, set the first input to the first operand, and set the third input to the second operand, and set the second input and the fourth input to 0;

generate the target microcode instruction according to the first input, the second input, the third input and the fourth input.

[0069] The calculating unit 14 is further configured to:

acquire a first input, a second input, a third input and a fourth input;

perform a modular addition operation on the first input and the second input

to obtain the first input;

perform a modular addition operation on the third input and the fourth input to obtain the third input; and

perform a modular multiplication operation on the first input and the third input to obtain a calculation result, so as to complete the modular multiplication operation for the operand.

[0070] Taking the above formula as an example, where A is the first operand, C is the second operand, B=D=0, ((A+B)mod(p)*(C+D)mod(p))mod(p)=(A* C)mod(p) is equivalent to that the calculating unit realizes the modular multiplication operation on the first operand and the second operand.

[0071] In the fourth operation situation, the target type operation is a modular addition squared operation on the first operand and the second operand.

[0072] The accelerator or the CPU is further configured to:

in response to the target type operation being a modular addition squared operation on the first operand and the second operand, set the first input and the third input to the first operand, and set the second input and the fourth input to the second operand; and

generate the target microcode instruction according to the first input, the second input, the third input and the fourth input.

[0073] The calculating unit 14 is further configured to:

acquire a first input, a second input, a third input and a fourth input;

perform a modular addition operation on the first input and the second input to obtain a first modular addition result;

perform a modular addition operation on the third input and the fourth input to obtain a second modular addition result; and

perform a modular multiplication operation on the first modular addition result and the second modular addition result to obtain a calculation result, so as to complete the modular addition squared operation for the operand.

[0074] Taking the above formula as an example, where A=C= the first operand, B=D= the second operand, ((A+B)mod(p)*(C+D)mod(p))mod(p)= ((A+B)mod(p))2

mod(p) is equivalent to that the calculating unit realizes the modular addition squared operation on the first operand and the second operand.

[0075] With reference to the description of the four operations implemented on the calculating unit in the above-mentioned first to fourth operation situations, the calculating unit can also implement a modular subtraction squared, a modular addition multiplying modular subtraction, a modular addition multiplying modular addition, a modular subtraction multiplying modular subtraction by adjusting the above four inputs required by the general operation logic. No detailed explanation is provided here.

[0076] In some embodiments, the modular multiplication operation included in the basic operation logic includes a modular multiplication operation of a first type and a modular multiplication operation of a second type; the accelerator further comprises a data storage space; the data storage space stores a preset constant; the preset constant is used to mutual convert between a first data field and a second data field; the first data field is used for the modular multiplication operation of the first type; the second data field is used for the modular multiplication operation of the second type.

[0077] The calculating unit 14 is configured to, in response to the modular multiplication operation converting from the first type to the second type, acquire the preset constant and convert the first data field into the second data field according to the preset constant to implement the modular multiplication operation of the second type; in response to the modular multiplication operation converting from the second type to the first type, acquire the preset constant and convert the second data field into the first data field according to the preset constant to implement the modular multiplication operation of the first type.

[0078] The present application does not limit the first type and the second type. For example, the first type may be a conventional type and the second type may be a Montgomery type.

[0079] One of the differences between the two different types of modular multiplication operation is the different data fields. Data field conversion can be performed through the preset constant. For example, the preset constant is a, the first data field is 0 to p, and p+a is to obtain the second data field 0 to (p+a). By storing the

preset constant in the data storage space, when the modular multiplication operation type changes, the corresponding data field can be adjusted through the preset constant, so that the calculating unit is compatible with multiple modular multiplication operation types.

[0080] Taking the first type is a conventional type and the second type is a Montgomery type as example. The preset constant used for a conversion between a Montgomery data filed and a conventional data filed can be stored in the data storage space in advance via the configuration bus. The calculating unit can implement conversion of the data filed by obtaining the preset constant to achieve compatibility of a conventional type modular multiplication operation and a Montgomery type modular multiplication operation.

[0081] In some embodiments, the accelerator can be debugged by a debugging unit corresponding to the accelerator.

[0082] The controlling unit in the accelerator includes a breakpoint subunit

[0083] The breakpoint subunit is configured to interrupt the calculating unit to execute subsequent microcode instruction in response to reception of a interrupt instruction, so as to enable a debugging unit corresponding to the accelerator to acquire a calculation result of the calculating unit, and output a comparison result between the calculation result and a preset calculation result. Thereby, the accelerator can be debugged by monitoring the calculating process of the calculating unit.

[0084] It is assumed that the accelerator is executing multiple consecutive target microcode instructions, and these target microcode instructions include an interrupt instruction. After receiving the interrupt instruction, the breakpoint subunit can initiate an interrupt parse request to the parsing unit to interrupt the accelerator to continue executing the microcode instructions. The breakpoint subunit can further send a request to the debugging unit to enable the debugging unit to read the calculation result of the calculating unit from the data storage space via the configuration bus and compare it with the preset calculation result. If the two are consistent, a green light will be emitted through a connector; if the two are inconsistent, a red light will be emitted through the connector, which enables a developer to know an operation condition of the accelerator

and facilitates debugging.

[0085] Reference please be made to FIG. 2, FIG. 2 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0086] As shown in FIG. 2, the accelerator 20 may include an acquisition unit 21, a parsing unit 22, a controlling unit 23, a calculation unit 24 and a data storage space 25 connected in sequence.

[0087] The same functional units in the accelerator 20 and the above-mentioned accelerator 10 can achieve the same purpose, which will not be described in detail here.

[0088] The configuration bus illustrated in FIG. 2 can connect the accelerator 20 with other components. For example, the accelerator 20 can be connected to a CPU or a graphics processor (GPU) via the configuration bus. The transmission of data and instruction via the configuration bus can improve the stability of the transmission.

[0089] A target microcode instruction as well as a target input data that can enable the calculating unit to complete target type operation can be input to the accelerator 20 via the configuration bus. The microcode instruction may be stored in a preset instruction storage space, and the target input data may be stored in the data storage space 25.

[0090] In some manners, the target microcode instruction is obtained by converting an operation program by the accelerator or the CPU. The accelerator or the CPU is further configured to:

store the operand obtained by parsing in the data storage space; and

generate the target microcode instruction according to the target type operation based on a storage address of the operand and/or the preset operand in the data storage space.

[0091] The operand is an operand parsed from the operation program and used to perform the target type operation.

[0092] The operand can be sent to the data storage space for storage via the configuration bus, and the data storage space can return the storage address corresponding to the operand. The preset operand for generating the microcode instruction can also be stored in the data storage space in advance via the configuration

bus.

[0093] The preset operand can assist the calculating unit to implement the target type operation. The preset operand can be set according to a requirement, for instance, may include 0 and 1. For example, the calculating unit includes the general operation logic shown in the above-mentioned formula. If the calculating unit needs to implement a modular addition operation, then A and B can be set as operands parsed from the operation program, C can be set as a preset operand 0, and D can be set as a preset operand 1.

[0094] After the operand storage is completed, the target configuration rule for configuring each input included in the target microcode instruction can be determined according to the target type operation, then each input is configured according to the target configuration rule to generate the target microcode instruction. Specifically, each input can be configured according to storage addresses of the operand and/or the preset operand in the data storage space, and the target microcode instruction may be generated.

[0095] The target microcode instruction obtained from this may include the storage addresses of these target input data in the data storage space. In this case, the parsing result obtained by parsing the target microcode instruction may also include the storage address of the target input data. The controlling unit, according to the storage address, controls the calculating unit to read the target input data from the data storage space and perform the target type operation based on the target input data.

[0096] In some embodiments, the data storage space 25 may include a data buffer and a data register. The data buffer may include an input buffer and an output buffer. The input data transmitted by the configuration bus and the calculation result obtained by the calculating unit 24 can be stored in the data buffer and the data register as needed, and are not specifically limited in the present application. Of course, if the calculating unit 24 continuously performs multiple calculations, the intermediate calculation results can be stored in the data register, thereby improving calculation efficiency.

[0097] In some embodiments, the operation program further includes a dot multiplication scalar; the target type operation is a dot multiplication operation on the

dot multiplication scalar and the operand; the accelerator or the CPU is further configured to:

generate a plurality of target microcode instructions for implementing the dot multiplication operation according to the dot multiplication scalar and the operand included in the operation program;

the accelerator is further configured to:

control the calculating unit to execute each target microcode instruction in the plurality of target microcode instructions, so as to complete the dot multiplication operation.

[0098] Thus, by parsing the operation program, the multiple target microcode instructions are obtained, and the calculating unit is controlled to implement the dot multiplication operation based on the multiple target microcode instructions.

[0099] Reference please be made to FIG. 3, FIG. 3 is a schematic structural diagram of an accelerator shown in an embodiment of the present application.

[0100] As shown in FIG. 3, the accelerator 30 may include an acquiring unit 31, an parsing unit 32, a controlling unit 33, a calculating unit 34 and a data storage space 35 connected in sequence; and a cache unit 36 connected to the controlling unit 33 and the calculating unit 34. The relevant explanations of the units 31 to 35 can refer to the above-mentioned embodiments and will not be described in detail here.

[0101] The cache unit 36 stores a dot multiplication scalar used for the dot multiplication. The dot multiplication scalar may be a preset binary number with a preset number of bits, such as a 256-bit binary number. In some ways, the dot multiplied scalar may be stored in the cache unit 36 via the configuration bus.

[0102] It can be understood that the operation process of the dot multiplication is actually to perform several simple operations based on the number of bits in each bit of the dot multiplication scalar. The simple operations may include a modular addition, a modular subtraction, a modular multiplication, a modular addition squared, a modular subtraction squared, a modular addition multiplying modular subtraction, a modular addition multiplying modular addition, and a modular subtraction multiplying modular subtraction. Therefore, in order to implement the dot multiplication operation, the

multiple target microcode instructions corresponding to these simple operations respectively can be obtain by writing manually or converting the operation program by the accelerator or the CPU to control the calculating unit to complete the above-mentioned simple operations to complete the dot multiplication.

[0103] Taking the implementation of asymmetric encryption through ECC as an example to illustrate in following.

[0104] Given a base point G on an ellipse, two points d and Q on the ellipse can be selected, where Q = dG, as a private key and a public key respectively.

[0105] When the public key Q is used to encrypt a plaintext M, a random number r can be acquired, and a ciphertext C can be obtained through a formula M+rQ.

[0106] When the private key d is used to decrypt the ciphertext C, the plaintext M can be obtained by decrypting through a formula C-d(rG)=M+r(dG)-d(rG)=M.

[0107] In the above-mentioned encryption and decryption process, the dot multiplication operation (such as Rq, Dg, etc.) need to be performed frequently. ECC calculation can be performed through the accelerator shown in the above-mentioned embodiments. On the one hand, the accelerator can implement different types operations by changing the microcode instruction, which simplifies the configuration operation of the accelerator and reduces the difficulty of configuring the accelerator, thereby use of the accelerator can be more flexible; on the other hand, at least two types of ECC operations can be implemented through one calculating unit, thereby saving hardware resource.

[0108] The present application also proposes an acceleration method, the acceleration method can be applied to the accelerator shown in any one of the above-mentioned embodiments. The accelerator includes a calculating unit configured to implement an operation related to an elliptic curves algorithm; where the calculating unit includes a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs.

[0109] Reference please be made to FIG. 4, FIG. 4 is a schematic flowchart of an accelerating method shown in an embodiment of the present application. As shown in

FIG. 4, the method may include S402-S406. Unless otherwise specified, the present application does not limit the execution order of these steps.

[0110] S402: acquiring a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; where the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation.

[0111] S404: parsing the target microcode instruction to obtain a parsing result corresponding to the target input data.

[0112] S406: in response to the parsing result, controlling the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

[0113] In the above-mentioned solution, first, the target microcode instruction can be generated according to the target type operation to be implemented, and the relevant configuration of the input required for the general operation logic to implement the target type operation in the calculating unit is completed according to the target microcode instruction, so as to enable the accelerator to complete the target type operation. Compared with related art 1, the accelerator can perform different type operations by changing the microcode instruction, which simplifies the configuration operation on the accelerator and reduces the difficulty of configuring the accelerator, thereby the accelerator can be used more flexible.

[0114] Second, the general operation logic composed of the basic operation logic is deployed in the calculating unit; the general operation logic perform the operation of different operation types in response to different inputs. Compared with related art 2, at least two types of operations can be implement by one calculating unit, thereby saving the hardware resource.

[0115] Some embodiments related to the acceleration methods are introduced below. The implementation process of these embodiments may refer to the previous embodiments regarding the accelerator, and will not be elaborated here.

[0116] In some embodiments, the accelerator is communicatively connected to a CPU; the accelerating method further includes:

acquiring an operation program that performs the target type operation on an operand; and

generating the target microcode instruction corresponding to the target type operation, according to the target type operation, based on the operand included in the operation program.

[0117] In some embodiments, the accelerator further includes a data storage space; the data storage space stores a preset operand used to generate a microcode instruction;

the generating the target microcode instruction corresponding to the target type operation according to the target type operation based on the operand included in the operation program includes:

storing the operand obtained by parsing in the data storage space; and

generating the target microcode instruction according to the target type operation, based on a storage address of the operand and/or the preset operand in the data storage space.

[0118] In some embodiments, the parsing result includes a storage address of the target input data;

the in response to the parsing result, controlling the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation includes:

controlling, according to the storage address, the calculating unit to read the target input data from the data storage space and perform the target type operation based on the target input data.

[0119] In some embodiments, the basic operation logic includes a modular addition operation and a modular multiplication operation, the general operation logic is to perform a modular multiplication on a calculation result of two modular addition operations.

[0120] In some embodiments, an input required by the general operation logic includes a first input, a second input, a third input and a fourth input;

an operation process of the general operation logic includes:

performing a modular addition operation on the first input and the second

input to obtain a first modular addition result;

performing a modular addition operation on the third input and the fourth input to obtain a second modular addition result; and

performing a modular multiplication operation on the first modular addition result and the second modular addition result to obtain a final calculation result.

[0121] In some embodiments, the operand includes a first operand and a second operand;

the accelerating method further includes:

in response to the target type operation being a modular addition operation on the first operand and the second operand, setting the first input to the first operand, setting the second input to the second operand, setting the third input to 1 and setting the fourth input to 0;

in response to the target type operation being a modular subtraction operation on the first operand and the second operand, setting the first input to the first operand, setting the second input to the second operand and adjusting a symbol corresponding to the second input to a negative sign, setting the third input to 1 and setting the fourth input to 0;

in response to the target type operation being a modular multiplication operation on the first operand and the second operand, setting the first input to the first operand, and setting the third input to the second operand, and setting the second input and the fourth input to 0;

in response to the target type operation being a modular addition squared operation on the first operand and the second operand, setting the first input and the third input to the first operand, and setting the second input and the fourth input to the second operand; and

generating the target microcode instruction according to the first input, the second input, the third input and the fourth input.

[0122] In some embodiments, the operation program further includes a dot multiplication scalar; the target type operation is a dot multiplication operation on the dot multiplication scalar and the operand;

the generating the target microcode instruction corresponding to the target type operation according to the target type operation based on the operand included in the operation program includes:

generating a plurality of target microcode instructions for implementing the dot multiplication operation according to the dot multiplication scalar and the operand included in the operation program;

the accelerating method further includes:

controlling the calculating unit to execute each target microcode instruction in the plurality of target microcode instructions, so as to complete the dot multiplication operation.

[0123] In some embodiments, the modular multiplication operation includes a modular multiplication operation of a first type and a modular multiplication operation of a second type; the accelerator further includes a data storage space; the data storage space stores a preset constant; the preset constant is used to mutual convert between a first data field and a second data field; the first data field is used for the modular multiplication operation of the first type; the second data field is used for the modular multiplication operation of the second type;

the accelerating method further includes:

in response to the modular multiplication operation converting from the first type to the second type, acquiring the preset constant and converting the first data field into the second data field according to the preset constant to implement the modular multiplication operation of the second type;

in response to the modular multiplication operation converting from the second type to the first type, acquiring the preset constant and converting the second data field into the first data field according to the preset constant to implement the modular multiplication operation of the first type.

[0124] In some embodiments, the accelerating method further includes:

interrupting the calculating unit to execute subsequent microcode instruction in response to reception of a interrupt instruction, so as to enable a debugging unit corresponding to the accelerator to acquire a calculation result of the

calculating unit, and output a comparison result between the calculation result and a preset calculation result.

[0125] In some embodiments, the accelerator is a coprocessor implemented based on FPGA or ASIC for accelerating the elliptic curves algorithm; or, an auxiliary accelerating module implemented based on CPU for performing an algorithm acceleration for the elliptic curves algorithm.

[0126] The present application also proposes an electronic device, the electronic device includes the accelerator as shown in any one of the above-mentioned embodiments. ECC acceleration is performed through the electronic device. On the one hand, the accelerator can implement different types operations by changing the microcode instruction, which simplifies the configuration operation of the accelerator and reduces the difficulty of configuring the accelerator, thereby the accelerator can be more flexible; on the other hand, at least two types of ECC operations can be implemented through one calculating unit in the accelerator, thereby saving hardware resource.

[0127] The present application also proposes a computer-readable storage medium, where the storage medium stores a computer program, the computer program is used to enable an accelerator in a computer to execute the acceleration method shown in any of the above-mentioned embodiments.

[0128] It should be understood by those skilled in the art that one or more embodiments of the present application may be provided as a method, a system or a computer program product. Accordingly, one or more embodiments of the present application may take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment that combines software and hardware aspects. Furthermore, one or more embodiments of the present application may employ the form of the computer program product implemented on one or more computer-usable storage mediums (which may include, but are not limited to a disk storage, a compact disc read only memory (CD-ROM), an optical storage, etc.) including computer-usable program codes therein.

[0129] "And/or" in the present application means at least one of the two. For

example, "A and/or B" can include three solutions: A, B, and "A and B".

[0130] Each embodiment in the present application is described in a progressive manner. The same and similar parts between the various embodiments can be referred to each other. Each embodiment focuses on its differences from other embodiments. In particular, for the data processing device embodiment, since it is basically similar to the method embodiment, the description is relatively simple. For relevant details, please refer to the partial description of the method embodiment.

[0131] Specific embodiments of the present application have been described above. Other embodiments are within the scope of the appended claims. In some cases, the actions or steps recited in the claims can be performed in a different order than in the embodiments and still achieve desirable results. Additionally, the processes depicted in the accompanying drawings do not necessarily require the specific order as shown, or sequential order, to achieve the desirable results. In some implementations, multitasking and parallel processing are also possible or may be advantageous.

[0132] Embodiments of subject matter and functional operation described in the present application may be implemented in below: digital electronic circuits, tangibly embodied computer software or firmware, computer hardware that may include the structures disclosed in the present application and their structural equivalents, or a combination of one or more of them. Embodiments of the subject matter described in the present application may be implemented as one or more computer programs, i.e., one or more modules in a computer program instruction, encoded on a tangible non-temporary program carrier, is used to execute by a data processing device or control the operation of the data processing device. Alternatively or additionally, the program instruction can be encoded onto artificially generated propagation signals, such as machine generated electrical, optical, or electromagnetic signals, which are generated to encode information and transmit it to an appropriate receiver device for execution by the data processing device. The computer storage medium can be a machine readable storage device, a machine readable storage substrate, a random or serial access memory device, or a combination of one or more of them.

[0133] Processes and logic flows described in the present application can be

performed by one or more programmable computers executing one or more computer programs to generate output to perform corresponding functions by performing operation according to input data. The processes and logic flows can also be performed by a dedicated logic circuit, such as a Field Programmable Gate Array (FPGA) or an Application Specific Integrated Circuit (ASIC), and the apparatus may also be implemented as the dedicated logic circuit.

[0134] A computer suitable for executing a computer program may include, for example, a general and/or special purpose microprocessor, or any other types of processing units. Usually, the processing units will receive instructions and data from a read-only memory and/or random access memory. The basic components of the computer may include a processing unit for implementing or executing instructions and one or more memory devices for storing instructions and data. Usually, the computer also include one or more mass storage devices for storing data, such as magnetic, magneto-optical or optical disks, etc., or the computer be operably coupled to such mass storage devices to receive data therefrom or send data to it, or both. However, the computer does not necessarily have such devices. Additionally, the computer may be embedded in another device, such as a mobile phone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a global positioning system (GPS) receiver, or a portable storage device such as a universal serial bus (USB) flash drive, to name a few.

[0135] The Computer-readable medium suitable for storage of computer program instructions and data may include all forms of non-volatile memory, medium, and memory device, and may include, for example, a semiconductor memory device (such as an erasable programmable read only memory (EPROM), an electrically erasable programmable read only memory (EEPROM), and a flash memory device), a magnetic disk (such as an internal hard disk or a removable disk), a magneto-optical disk, and a CD-ROM and a digital video disk read only memory (DVD-ROM) disk. The processor and memory may be supplemented by the dedicated logic circuit or incorporated into the dedicated logic circuit.

[0136] Although the present application contains many specific implementation

details, these should not be construed to limit the scope of any disclosure or claim, but rather serve primarily to describe features of particular disclosed embodiments. Certain features described in multiple embodiments herein can also be combined in a single embodiment. On the other hand, various features that are described in the single embodiment can also be implemented in multiple embodiments separately or in any suitable sub-combination. Furthermore, while features may function in certain combinations as described and are even initially claimed as such, one or more features from a claimed combination may be removed from that combination in some cases, and the claimed protected combination may point to a sub-combination or a variant of the sub-combination.

[0137] Similarly, although the operations are depicted in a specific order in the accompanying drawings, this should not be understood as requiring these operations to be executed in the specific order or sequence as shown, or requiring all illustrated operations to be executed to achieve a desired result. In some cases, multitasking and parallel processing may be advantageous. Furthermore, the separation of various system modules and components in the described embodiments should not be understood as necessary in all embodiments, and it should be understood that the described program components and systems may generally be integrated together in a single software product, or packaged into multiple software products.

[0138] Thus, a specific embodiment of the subject matter have been described. Other embodiments are within the scope of the appended claims. In some cases, the actions recited in the claims can be performed in a different order and still achieve the desirable result. Furthermore, the processes depicted in the accompanying drawings are not necessarily in the specific or sequential order shown to achieve the desired result. In some implementations, multitasking and parallel processing may be advantageous.

[0139] The above are only preferred embodiments of one or more embodiments of the present application, and are not intended to limit one or more embodiments of the present application. Any modifications, equivalent substitutions, improvements, etc. have been done within the spirit and principles of one or more embodiments of the present application should be included in the scope of protection of one or more

embodiments of the present application.

CLAIMS

1. An accelerator, comprising: a calculating unit configured to implement an operation related to an elliptic curves algorithm; wherein the calculating unit comprises a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs; the accelerator further comprises:

an acquiring unit, configured to acquire a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; wherein the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation;

a parsing unit, configured to parse the target microcode instruction to obtain a parsing result corresponding to the target input data; and

a controlling unit, configured to, in response to the parsing result, control the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

2. The accelerator according to claim 1, wherein the accelerator is communicatively connected to a central processing unit, CPU; the accelerator or the CPU is configured to:

acquire an operation program that performs the target type operation on an operand; and

generate the target microcode instruction corresponding to the target type operation according to the target type operation based on the operand comprised in the operation program.

3. The accelerator according to claim 2, wherein the accelerator further comprises a data storage space; the data storage space stores a preset operand used to generate a microcode instruction;

the accelerator or the CPU is further configured to:

store the operand obtained by parsing in the data storage space; and

generate the target microcode instruction according to the target type operation based on a storage address of the operand and/or the preset operand in the data storage space.

4. The accelerator according to claim 3, wherein the parsing result comprises a storage address of the target input data;

the controlling unit is configured to control, according to the storage address, the calculating unit to read the target input data from the data storage space and perform the target type operation based on the target input data.

5. The accelerator according to any one of claims 2 to 4, wherein the basic operation logic comprises a modular addition operation and a modular multiplication operation, the general operation logic is to perform a modular multiplication on a calculation result of two modular addition operations.

6. The accelerator according to claim 5, wherein an input required by the general operation logic comprises a first input, a second input, a third input and a fourth input;

an operation process of the general operation logic comprises:

performing a modular addition operation on the first input and the second input to obtain a first modular addition result;

performing a modular addition operation on the third input and the fourth input to obtain a second modular addition result; and

performing a modular multiplication operation on the first modular addition result and the second modular addition result to obtain a final calculation result.

7. The accelerator according to claim 6, wherein the operand comprises a first operand and a second operand;

the accelerator or the CPU is further configured to:

in response to the target type operation being a modular addition operation on the first operand and the second operand, set the first input to the first operand, set the second input to the second operand, set the third input to 1 and set the fourth input to 0;

in response to the target type operation being a modular subtraction operation on the first operand and the second operand, set the first input to the first operand, set the

second input to the second operand and adjust a symbol corresponding to the second input to a negative sign, set the third input to 1 and set the fourth input to 0;

in response to the target type operation being a modular multiplication operation on the first operand and the second operand, set the first input to the first operand, and set the third input to the second operand, and set the second input and the fourth input to 0;

in response to the target type operation being a modular addition squared operation on the first operand and the second operand, set the first input and the third input to the first operand, and set the second input and the fourth input to the second operand; and

generate the target microcode instruction according to the first input, the second input, the third input and the fourth input.

8. The accelerator according to claim 6, wherein the operation program further comprises a dot multiplication scalar; the target type operation is a dot multiplication operation on the dot multiplication scalar and the operand;

the accelerator or the CPU is further configured to:

generate a plurality of target microcode instructions for implementing the dot multiplication operation according to the dot multiplication scalar and the operand comprised in the operation program;

the accelerator is further configured to:

control the calculating unit to execute each target microcode instruction in the plurality of target microcode instructions, so as to complete the dot multiplication operation.

9. The accelerator according to claim 6, wherein the modular multiplication operation comprises a modular multiplication operation of a first type and a modular multiplication operation of a second type; the accelerator further comprises a data storage space; the data storage space stores a preset constant; the preset constant is used to mutual convert between a first data field and a second data field; the first data field is used for the modular multiplication operation of the first type; the second data field is used for the modular multiplication operation of the second type;

the calculating unit is configured to, in response to the modular multiplication

operation converting from the first type to the second type, acquire the preset constant and convert the first data field into the second data field according to the preset constant to implement the modular multiplication operation of the second type;

in response to the modular multiplication operation converting from the second type to the first type, acquire the preset constant and convert the second data field into the first data field according to the preset constant to implement the modular multiplication operation of the first type.

10. The accelerator according to claim 1, wherein the controlling unit comprises a breakpoint subunit;

the breakpoint subunit is configured to interrupt the calculating unit to execute subsequent microcode instruction in response to reception of a interrupt instruction, so as to enable a debugging unit corresponding to the accelerator to acquire a calculation result of the calculating unit, and output a comparison result between the calculation result and a preset calculation result.

11. The accelerator according to claim 1, wherein the accelerator is a coprocessor implemented based on field programmable gate array, FPGA, or application specific integrated circuit, ASIC, for accelerating the elliptic curves algorithm; or, an auxiliary accelerating module implemented based on a CPU for performing an algorithm acceleration for the elliptic curves algorithm.

12. An accelerating method, applied to an accelerator, wherein the accelerator comprises a calculating unit configured to implement an operation related to an elliptic curves algorithm; wherein the calculating unit comprises a general operation logic composed of a basic operation logic; the general operation logic perform the operation of different operation types in response to different inputs;

the methods comprises:

acquiring a target microcode instruction corresponding to a target type operation to be implemented generated according to the target type operation; wherein the target microcode instruction is used to indicate target input data required to enable the general operation logic to implement the target type operation;

parsing the target microcode instruction to obtain a parsing result corresponding

to the target input data; and

in response to the parsing result, controlling the calculating unit to enable the calculating unit to acquire the target input data and run the general operation logic based on the target input data so as to complete the target type operation.

13. An electronic device, comprising the accelerator according to any one of claims 1 to 11.