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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 January 2020 (02.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/002441 A1

(51) International Patent Classification:
G06F 21/44 (2013.01)

(21) International Application Number:
PCT/EP2019/067023

(22) International Filing Date:
26 June 2019 (26.06.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
1810544.5 27 June 2018 (27.06.2018) GB

(71) Applicant: **NORDIC SEMICONDUCTOR ASA** [NO/NO]; Otto Nielsens veg 12, 7052 Trondheim (NO).

(72) Inventors: **BRELOT, Jean-Baptiste**; C/O Nordic Semiconductor ASA, Otto Nielsens veg 12, 7052 Trondheim (NO). **AUNE, Frank**; C/O Nordic Semiconductor ASA, Otto Nielsens veg 12, 7052 Trondheim (NO).

(74) Agent: **DEHNS**; St Bride's House, 10 Salisbury Square, London Greater London EC4Y 8JD (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(54) Title: METHOD OF DEBUGGING A DEVICE

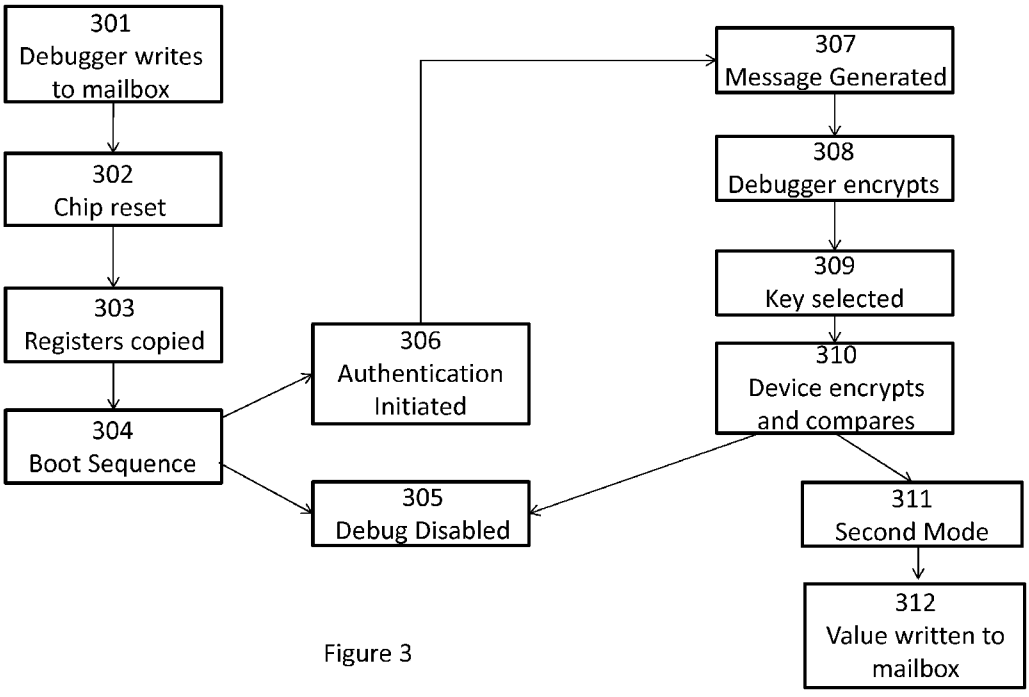

Figure 3

(57) Abstract: A method of debugging a device(1) is provided. The device (1) comprises a processor (6),and anon-volatile memory (8) connected to the processor(6) and an access port (12).The device (1) has a protected area in which protected software (24) is run. The device (1) has a first mode in which debug features are disabled and a second mode in which they are enabled. A debugger (2) is connected to the access port (12) of the device (1) in the first mode. The protected software (24) generates a message and sends it to the access port(12). The debugger (2) receives the message and encrypts it, generating a first encrypted message.The debugger (2) sends this encrypted message back to the protected software via the access port(12), which attempts to authenticate the debugger (2) by determining whether the message was encrypted using a predetermined encryption.If authentication is successful, the software (24) changes the device from the first mode to the second mode.

WO 2020/002441 A1

[Continued on next page]

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

Method of debugging a device

BACKGROUND OF THE INVENTION

This invention relates to interfaces to integrated circuit microprocessor devices, for example interfaces that might be used by a product designer incorporating the device into a larger product; more specifically, to a method of debugging a device using these physical interfaces.

Modern electronic devices, particular system-on-chip (SoC) devices, are often equipped with a number of ports, which may be connected to a physical pin on the device so that the device may interact with peripheral devices. When designing a system that utilises such a device, the designer will usually configure the numerous ports for various functions as desired depending on a particular intended use. For example, some of the ports may be used for data input, data output, connection to an antenna etc. The designer or subsequent developer, such as an OEM, will also usually need to carry out debugging (i.e. identifying and removing errors) at various stages during the design process.

In order to carry out debugging, the designer or developer might access the device using an access port. This access port allows the designer or developer to interface with the device following an error becoming apparent, analyse the situation to identify the cause of the error and then perform some corrective action (such as resetting the device, clearing registers etc.) in order to rectify the error and continue the design or configuration process. An issue can arise wherein the error may cause the whole device to be "locked up" or "bricked", preventing access in order to correct the error. WO2017/072500 discloses an invention which overcomes this problem by locating an access port in a separate power domain to the rest of the device so that the access port is always accessible.

The Applicant has now devised certain improvements to the device of WO2017/072500.

SUMMARY OF THE INVENTION

From a first aspect, the invention provides a method of debugging a device; wherein the device comprises:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger; and

a protected area in which protected software is run;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the method comprises:

connecting a debugger to the access port of the device when the device is in the first mode;

the protected software generating a message and sending said message to the access port;

the debugger receiving the message from the access port, encrypting the message to generate a first encrypted message, and sending the first encrypted message back to the protected software by means of the access port;

the protected software receiving the first encrypted message, attempting to authenticate the debugger using the first encrypted message by determining whether the message has been encrypted by a predetermined encryption; and

if authentication is successful, the software changing the device from being in the first mode to being in the second mode.

Thus it will be seen that, in accordance with the invention, a method is provided which allows device-external software with appropriate cryptographic credentials to authenticate itself towards secure software running inside the device, and in turn to request temporary re-enabling of previously disabled debug features of the device. This provides the benefit of restricting the use of debug features to certain external software having the correct credentials, thereby making the debugging process secure. This invention furthermore provides the benefit of allowing the use of debug pins already existing e.g in devices of the kind described in WO2017/072500 to carry out the authentication process, thereby avoiding a need to use any other input/output

(IO) pins, for example those related to a universal asynchronous receiver-transmitter (UART), to achieve this function.

The Communication with the device may be carried out in any convenient way. However, in a set of embodiments the access port comprises a mailbox comprising mailbox registers. Preferably in such embodiments the processor sends the generated message to a mailbox register, and the debugger sends the processed message back to the mailbox register.

In a set of embodiments the debugger can only read and write to the mailbox registers and cannot access the non-volatile memory or the processor. This further ensures the safety of the device since it prevents a possibly non-secure debugger from accessing the internal resources of the device.

The message generated by the protected software could take a number of different forms, but in a set of embodiments the message generated by the protected software comprises a random or pseudo-random value. The generated message can be a nonce. A nonce is an arbitrary number, often a random or pseudo-random number, which can be used just once. The advantage of the use of a random value is that it improves the security of the system as previously generated messages cannot be used in what is known as a “replay attack”.

The debugger could in theory be authenticated by decrypting the encrypted message from it and comparing this to the message generated internally. In a set of embodiments, attempting to authenticate the debugger comprises: the software using an encryption key to encrypt the message to generate a second encrypted message; and, the software comparing the second encrypted message to the first encrypted message; . If the first encrypted message and the second encrypted message match determining that authentication is successful.

In a set of embodiments the non-volatile memory comprises a User Information Configuration Register (UICR) which preferably comprises a secure region (SICR). In such embodiments the secure software comprises this SICR. In certain embodiments at least one decryption key is stored in the SICR.

In a set of embodiments, the device further comprises a non-volatile memory control unit (NVMC), configured to control the non-volatile memory, and the NVMC preferably comprises a Key Management Unit. In such embodiments the secure software further comprises this Key Management Unit. The Key Management Unit may select the decryption key from the keys stored in the SICR.

In a set of embodiments the secure software further comprises a Crypto Hardware Accelerator. The Crypto Hardware Accelerator may generate the message to be sent to the mailbox of the control access port. The key from the SICR may be sent to the Crypto Hardware Accelerator; and it may be the Crypto Hardware Accelerator which encrypts the message generated by the protected software, and compares the encrypted message to the message generated by the debugger.

In a set of embodiments the protected software comprises a secure bootloader; said secure bootloader being configured to read and write to the mailbox registers of the access port mailbox. In a set of embodiments the secure bootloader sends the generated message from the Crypto Hardware Accelerator to the access port mailbox. Preferably the secure bootloader instructs the Key Management Unit to select the key from the UICR and send the key to the Crypto Hardware Accelerator.

In a set of embodiments the protected software further comprises a System Protection Unit (SPU) which configures which parts of the device are considered secure or non-secure. If the authentication is successful the secure bootloader may write to a register of the SPU and thereby allow the debugger access to the non-volatile memory and the processor.

In a set of embodiments the method further comprises, after a successful authentication, the secure bootloader writing a value to the access port mailbox, the value being sent to the debugger, and the debugger writing the same value to the access port mailbox. This is a security measure to ensure that the secure bootloader is not able to perform an Erase All command on its own without a debugger actually being connected. Once the values of the two registers are equal, the debugger automatically issues an Erase All command, this further increases security and prevents non-secure software from being able to write to the NVMC register using secure code.

The invention furthermore extends to a debugging system. Thus from a further aspect the invention provides a debugging system comprising:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger;

a protected area in which protected software is run; and

a debugger;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the device is configured such that when the debugger is connected to the access port of the device, when the device is in the first mode, the protected software generates a message and sends said message to the access port;

the debugger is configured to receive the message from the access port, encrypt the message to generate a first encrypted message, and send the first encrypted message back to the protected software by means of the access port;

the protected software is configured to receive the first encrypted message, attempt to authenticate the debugger using the first encrypted message by determining whether the message has been encrypted by a predetermined encryption; and

if authentication is successful, the software is configured to change the device from being in the first mode to being in the second mode.

From a further aspect the invention provides an integrated circuit device comprising:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger; and

a protected area in which protected software is run;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the device is configured such that when a debugger is connected to the access port of the device, when the device is in the first mode, the protected software generates a message and sends said message to the access port;

the protected software is configured to receive a first encrypted message from the access port, determine whether the first encrypted message has been encrypted by a predetermined encryption; and

if the first encrypted message has been encrypted by a predetermined encryption, the software is configured to change the device from being in the first mode to being in the second mode.

It will be understood that any of the optional features described above in relation to the method of debugging a device are also optional features of the debugging system and the integrated circuit device described herein.

Features of any embodiment described herein may, wherever appropriate, be applied to any other embodiment described herein. Where reference is made to different embodiments or sets of embodiments, it should be understood that these are not necessarily distinct but may overlap.

BRIEF DESCRIPTION OF THE DRAWINGS

Certain preferred embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 shows a system-on-chip (SoC) integrated circuit device in accordance with an embodiment of the present invention.

Figure 2 shows the protected software of the device of Figure 1 in more detail.

Figure 3 is a flow diagram, showing the authentication process by which disabled debug features of the device of Figure 1 can be re-enabled.

DETAILED DESCRIPTION

Figure 1 shows a system-on-chip (SoC) integrated circuit device 1 in accordance with an embodiment of the present invention connected to an external debugger 2. The device 1 includes a number of external pins 4 to which the external debugger 2 is connected.

In this particular embodiment, the debugger 2 utilises the Serial Wire Debug (SWD) interface, an ARM® standard protocol that utilises two bi-directional wires 42. The protocol itself is defined in the ARM® Debug Interface v5 and ARM® Debug Interface v5.1, both of which are incorporated herein by reference. However, this particular embodiment is not limiting, and the principles of this invention can be readily applied to other interfaces such as the Joint Action Test Group (JTAG) interface, as well as other standard and proprietary debugging interfaces. The set of external pins 4 in this particular embodiment are suitable for connection to either a Serial-Wire-Debug (SWD) debugger, or a Joint Action Test Group (JTAG) debugger in accordance with the IEEE-1149.1 standard, as the device 1 is provided with a hybrid Serial Wire and Joint Test Action Group Debug Port (SWJ-DP) 20.

The ARM® Debug Interface (ADI) includes: Debug Ports (DPs), which are used to access the Debug Access Port (DAP) from an external debugger such as the debugger 2; and Access Ports (APs), to access on-chip system resources within the integrated circuit device 1.

The device 1 includes a processor 6 e.g. an ARM® Cortex®-M4. The device 1 also includes flash memory (i.e. non-volatile memory) 8, which is used to store firmware uploaded to the device 1 by the designer, as well as for use by the firmware itself. The flash memory 8 is arranged to be accessed using a memory access port 16 within the processor 6.

Within the device 1 is a control access port 12 which is connected to the SWJ-DP 20 via a Debug Access Port (DAP) Bus Interface 14, as defined within the ADI. The DAP Bus Interconnect 14 acts as an intermediate layer between debug ports (i.e. the SWJ-DP 20) and the control access port 12 and allows the debugger 2 to access the processor 6 in real-time without interrupts. The DAP Bus Interconnect 14 is implemented as a multiplexer (mux) which allows the SWJ-DP 20 to access both the memory access port 16 within the processor 2 and the control access port 12.

The control access port 12 is then connected to a non-volatile memory control (NVMC) unit 18, which has direct control over the flash memory 8. The flash memory 8 contains a number of user information configuration registers (UICR) 10. These registers 10 can be used to store user specific settings, and in this case are used to

store a first protection flag and a second protection flag. The firmware uploaded to the flash memory 8 by the designer is usually sensitive. The setting of the first protection flag prevents data being read from the flash memory 8 via the control access port 12. In order to disable this protection, the end user would need to clear the first protection flag, which requires erasing all of the flash memory 8 including anything else that may be stored in it. The second protection flag is discussed in more detail below.

The device 1 is divided into two power domains 100, 200. In this embodiment the first power domain 100 includes the processor 6 and associated memory access port 16, while the second power domain 200 includes the external pins 4, SWJ-DP 20, DAP Bus Interconnect 14, control access port 12 NVMC 18, and flash memory 8. In an alternative embodiment, not shown, the NVMC 18 and the flash memory 8 may be included in the first power domain 100.

The device 1 further includes protected software 24. This protected software 24 comprises a secure bootloader 26, a Key Management Unit (KMU) 30 which is a component of the NVMC 18, a Secure User Information Configuration Register (SICR) 36 which is a secure part of the UICR 10, Crypto Hardware Accelerator 34 and System Protection Unit (SPU) 28.

If the device 1 is "hard reset", i.e. the device 1 is powered off and subsequently powered on again, both power domains 100, 200 will be reset. However, in the case of a "soft reset" wherein an external reset command is given to the device 1, this will only cause the reset of the first power domain 100, thus resetting the processor 6, leaving the second power domain 200 unaffected.

While NVMC unit 18 may in general be able to write to memory, erase a page from memory, erase the entire memory etc., the control access port 12 is only able to issue Erase All commands to the NVMC 18. This further enhances the security of the device as it prevents an end-user being able to erase only the first protection flag in the UICR 10 without erasing the rest of the flash memory 8.

The registers 10 can further be used to store a second protection flag, which disables debug features of the device, for example, it prevents the control access port from issuing Erase All commands to the NVMC 18. The second protection flag can disable

the control access port from accessing all device memory map regions marked as non-secure, or marked as secure, or all regions of the control access port. Other debug features such as breakpoints, debug monitor exceptions and trace modes can be disabled by the second protection flag. This is particularly beneficial since there may be cases in which it is desirable to prevent an end-user from erasing the flash memory 8. The device therefore has a first mode, in which the second protection flag is set in the UICR 10 and therefore the debug features are disabled, and a second mode, in which debug features are enabled. In the second mode either the second protection flag has not been set in the registers 10, or the second protection flag has been set but debug features are separately enabled by means of an authentication process described below with reference to Figure 3.

The Applicant has furthermore appreciated that it is desirable that particular debuggers in possession of appropriate credentials are able to authenticate themselves to the device and re-enable debug features disabled by the second protection flag of the UICR 10. In the present embodiment the debug features which can be re-enabled include, but are not limited to, the feature of being able to issue Erase All commands to erase the flash memory 8.

For this purpose the device 1 is provided with a mailbox 22 comprising a number of mailbox registers, located in the control access port 12 and the protected software 24 running in a protected area of the device 1, described in more detail with reference to Figure 2. The debugger 2 cannot access the internal resources of the device, but can read and write to the registers of the control access port mailbox 22. The debugger 2 is therefore able to communicate with the protected software 24 over the control access port mailbox 22 and authenticate itself to the device 1 in order to enable the debug features previously disabled by the second protection flag of the UICR 10.

Figure 2 shows the protected software 24 in more detail. The protected software 24 comprises a secure bootloader 26, a Key Management Unit 30, which is component of the NVMC 18, an SICR 36 which is a secure part of the UICR 10, Crypto Hardware Accelerator 34 and System Protection Unit (SPU) 28.

The secure bootloader 26 is able to read and write to the registers of the mailbox 22 of the control access port 12.

The KMU 30 is a component of the NVMC 18 which uses parts of the flash memory 8, specifically a secure part of the UICR 10, referred to here as the SICR 36, for secure storage of keys 36. The KMU 30 is configured to revoke, delete, and transfer keys 36 directly to cryptographic peripherals such as the Crypto Hardware Accelerator 34, without the keys being accessible to any MCU. The KMU can hold multiple key types, where each key type can have different generations – but typically one key slot is reserved for the key for unlocking debug features as described herein.

The Crypto Hardware Accelerator 34 is configured to decrypt messages received from the secure bootloader 26 using the key received from the SICR 36. The SPU 28 configures which parts of the device are considered to be secure and non-secure, and makes it possible to assign security properties to peripherals and in input/output (IO) pins, such that they can be divided between secure and non-secure "worlds", so that non-secure software is blocked from accessing secure resources directly. In particular, it controls access from the debugger 2 to the internal bus system 38 of the device, and allows the debugger 2 access to this internal bus system 38 if the authentication procedure is successful. The authentication process, as well as the process for initiating authentication, is described in more detail below with reference to Figure 3.

Authentication of the debugger 2 is required only in the case where debug features have been disabled by the second protection flag in the UICR 10. In this case, the authentication initiation process begins with step 301. At this step 301 the debugger 2 writes some predefined data inside a particular one of the registers of the mailbox 22 of the control access port 12. The debugger then resets the device by using the control access port 12, at step 302. Following reset the registers of the control access port 12 are copied across to the CPU side at step 303.

At step 304, the secure bootloader 26 of the protected software 24 executes a boot sequence. At the end of this boot sequence the secure bootloader 26 reads the registers of the control access port mailbox 22. If it determines that the predefined data has not been written to the required register then the process proceeds to step 305, at which debug features remain disabled. If the required predefined data has been written to the required register, then at the end of the boot sequence the secure

bootloader 26 reads the value and proceeds to step 306. At this step the authentication procedure is initiated.

The authentication process begins at step 307 with the Crypto Hardware Accelerator 34 generating a message, which can for example be a random value to be used as a nonce, and sending this message to the debugger 2 by means of the control access port mailbox 22.

At step 308 the debugger encrypts this message. For example the debugger 2 can encrypt the random value using an encryption key which is stored in the debugger 2. The debugger 2 sends the encrypted message back to the secure bootloader 26 by means of the control access port mailbox 22. The term “encrypts” is considered here to cover any process which can be applied to a message to generate an “encrypted message” such that a further process applied to this “encrypted message” is able to determine whether the encryption applied to the initial message belonged to a certain group of “encryption” processes. The initial message itself need not be recoverable. For example, as an alternative, the debugger 2 could sign the nonce sent by the CPU and return this signature to the CPU. This would allow the CPU to verify this signature against an already trusted public key or certificate containing a public key. If the signature is valid, the CPU knows that the debugger possesses the private key belonging to the already trusted public key or certificate stored inside the device and can unlock debug features.

At step 309 the device selects a process to be applied to the initial message generated by the Crypto Hardware Accelerator 34. In this embodiment the Key Management unit (KMU) 30 selects the debug-enabling key from the keys stored in the SICR 36, and sends this key to the Crypto Hardware Accelerator 34. This key is invisible to the processor 6.

The initial message is then processed at step 310, and comparison of the message returned from the debugger with the processed message generated by the Crypto Hardware Accelerator 34 can be used to verify the data sent by the debugger, in a challenge-response protocol. In this particular embodiment the Crypto Hardware Accelerator 34 uses the selected key to encrypt the random value generated by the Crypto Hardware Accelerator 34. The CPU does not have direct access to the key but

can instruct the KMU to push the key to the cryptographic accelerator and use that to perform the cryptographic operation.

The Crypto Hardware Accelerator 34 then sends the processed random value to the CPU, and the CPU compares the processed random value generated by the Crypto Hardware Accelerator 34 to the encrypted value received from the debugger 2. If the two values do not match then the authentication process continues to step 305, in which the device remains in its first state and debugger access and the ability for the debugger 2 to erase the flash memory are both disabled.

If the two values match then it is verified that the debugger 2 is in possession of the appropriate cryptographic credentials, namely a particular encryption key which is one of the encryption keys stored in the SICR, and authentication is successful. The process then proceeds to step 311, in which the system is changed from being in a first mode to being in a second mode, by the secure bootloader 26 writing to a register of the SPU 28. In this second mode the debugger 2 is able to access the internal resources of the device 1 by means of the Advanced High Performance Bus Access Port (AHB-AP) 38 shown in Figure 2, and in which the debugger 2 is able to erase the contents of the flash memory 8 by issuing a command from the control access port mailbox 22 despite the second protection flag being set in the UICR 10.

Finally, at step 312, after authentication, the secure bootloader 26 writes a non-zero random value to a mailbox register 22 and sends this value (possibly encrypted) to the debugger 2, to write the same value to this control access port register 22. This is a security measure to ensure that the secure bootloader 26 is not able to perform an Erase All command on its own without a debugger actually being connected. Once the values of the two registers are equal, the debugger 2 automatically issues an Erase All command, this further increases security and prevents non-secure software from being able to write to the NVMC register using secure code.

It will be appreciated by those skilled in the art that the invention has been illustrated by describing one or more specific embodiments thereof, but is not limited to these embodiments; many variations and modifications are possible, within the scope of the accompanying claims.

Claims

1. A method of debugging a device; wherein the device comprises:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger; and

a protected area in which protected software is run;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the method comprises:

connecting a debugger to the access port of the device when the device is in the first mode;

the protected software generating a message and sending said message to the access port;

the debugger receiving the message from the access port, encrypting the message to generate a first encrypted message, and sending the first encrypted message back to the protected software by means of the access port;

the protected software receiving the first encrypted message, attempting to authenticate the debugger using the first encrypted message by determining whether the message has been encrypted by a predetermined encryption; and

if authentication is successful, the software changing the device from being in the first mode to being in the second mode.

2. The method of claim 1, wherein the access port comprises a mailbox comprising mailbox registers, wherein the processor sends the message to a mailbox register, and wherein the debugger sends the processed message back to the mailbox register.

3. The method of claim 2, wherein the debugger can only read and write to the mailbox registers and cannot access the non-volatile memory or the processor.

4. The method of any preceding claim, wherein the message generated by the protected software comprises a random or pseudo-random value.

5. The method of any preceding claim, wherein the protected software attempting to authenticate the debugger comprises:

the software using an encryption key to encrypt the message to generate a second encrypted message; and,

the software comparing the second encrypted message to the first encrypted message;

if the first encrypted message and the second encrypted message match, determining that authentication is successful.

6. The method of claim 5, wherein the non-volatile memory comprises a user information configuration register which comprises a secure region; and,

wherein the secure software comprises the secure region, in which at least one decryption key is stored.

7. The method of claim 5 or 6, wherein the device further comprises a non-volatile memory control unit, configured to control the non-volatile memory, and;

wherein the non-volatile memory control unit comprises a key management unit; and,

wherein the secure software further comprises the key management unit, which selects the decryption key from the keys stored in the secure region of the user information configuration register.

8. The method of any of claims 5-7, wherein the secure software further comprises a crypto hardware accelerator; and,

wherein the crypto hardware accelerator generates the message to be sent to the mailbox of the control access port;

wherein the key from the secure region of the user information configuration register is sent to the crypto hardware accelerator; and

wherein it is the crypto hardware accelerator which encrypts the message generated by the protected software, and compares the encrypted message to the message generated by the debugger.

9. The method of claim 8, wherein the protected software comprises a secure bootloader; and

wherein the secure bootloader is configured to read and write to the mailbox registers of the access port mailbox.

10. The method of claim 9, wherein it is the secure bootloader which sends the generated message from the crypto hardware accelerator to the access port mailbox.

11. The method of claim 10; wherein it is the secure bootloader which instructs the key management unit to select the key from the user information configuration register and send the key to the crypto hardware accelerator.

12. The method of any of claims 9-11, wherein the method further comprises, after a successful authentication, the secure bootloader writing a value to the access port mailbox, the value being sent to the debugger, and the debugger writing the same value to the access port mailbox.

13. The method of any preceding claim, wherein the protected software further comprises a system protection unit which configures which parts of the device are considered secure or non-secure; and,

wherein if the authentication is successful the secure bootloader writes to a register of the system protection unit and thereby allows the debugger access to the non-volatile memory and the processor.

14. A debugging system comprising:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger;

a protected area in which protected software is run; and

a debugger;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the device is configured such that when the debugger is connected to the access port of the device, when the device is in the first mode, the protected software generates a message and sends said message to the access port;

the debugger is configured to receive the message from the access port, encrypt the message to generate a first encrypted message, and send the first encrypted message back to the protected software by means of the access port;

the protected software is configured to receive the first encrypted message, attempt to authenticate the debugger using the first encrypted message by determining whether the message has been encrypted by a predetermined encryption; and

if authentication is successful, the software is configured to change the device from being in the first mode to being in the second mode.

15. An integrated circuit device comprising:

a first power domain including a processor;

a non-volatile memory connected to the processor;

a second power domain including an access port connected to the non-volatile memory, the access port being further connected to an electrical interface suitable for connection to a debugger; and

a protected area in which protected software is run;

wherein the device has a first mode in which debug features are disabled and a second mode in which debug features are enabled;

wherein the device is configured such that when a debugger is connected to the access port of the device, when the device is in the first mode, the protected software generates a message and sends said message to the access port;

the protected software is configured to receive a first encrypted message from the access port, determine whether the first encrypted message has been encrypted by a predetermined encryption; and

if the first encrypted message has been encrypted by a predetermined encryption, the software is configured to change the device from being in the first mode to being in the second mode.

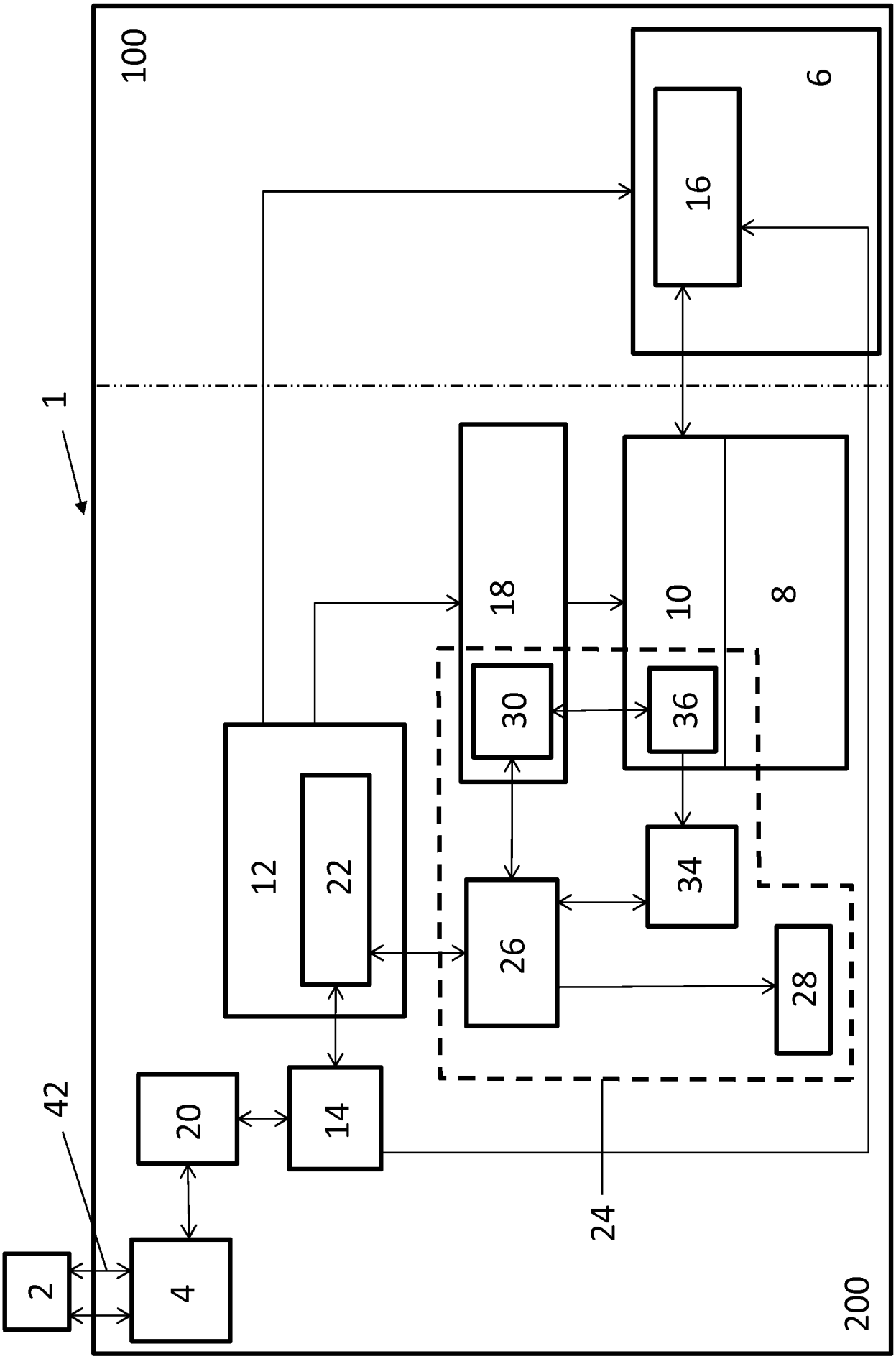

Figure 1

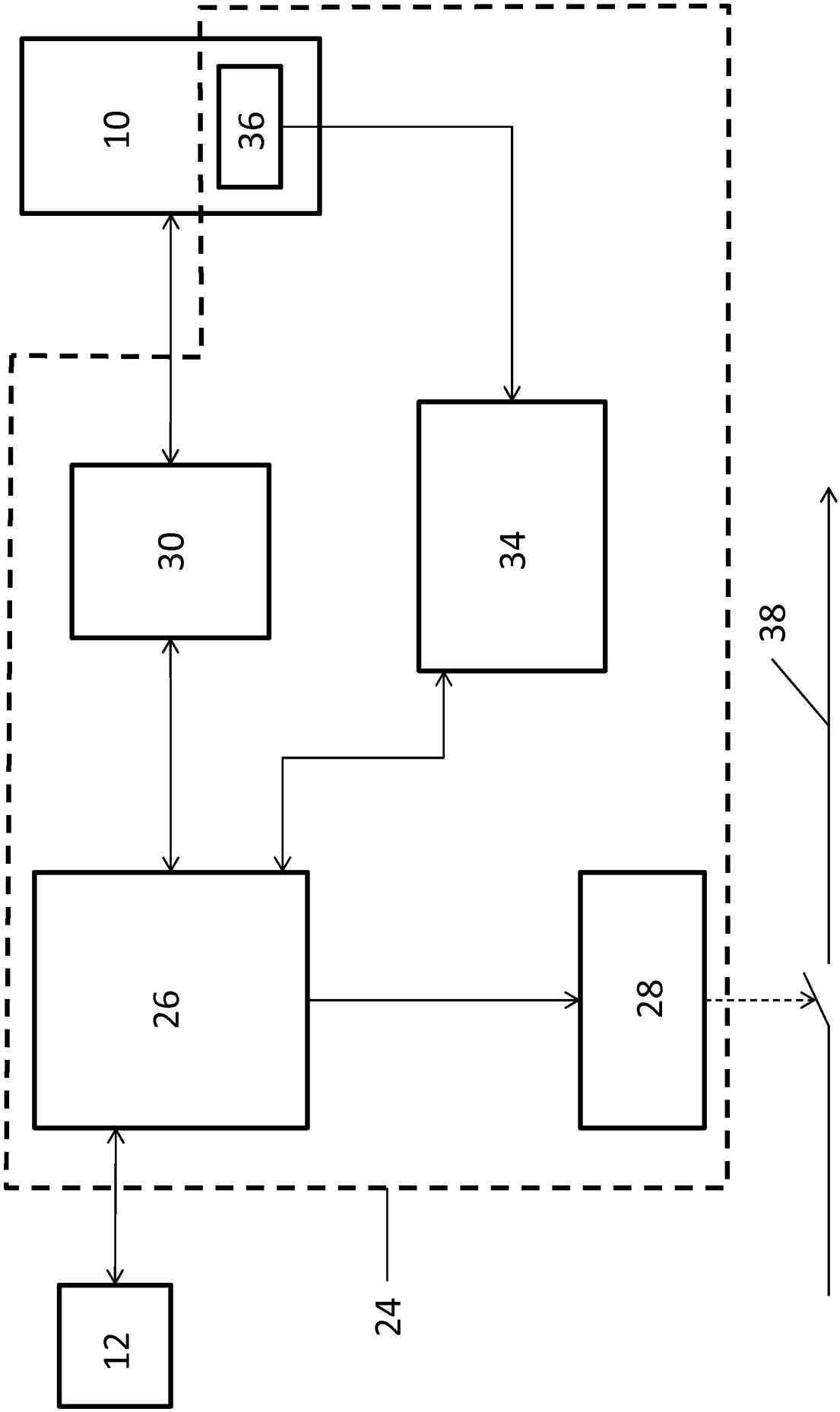

Figure 2

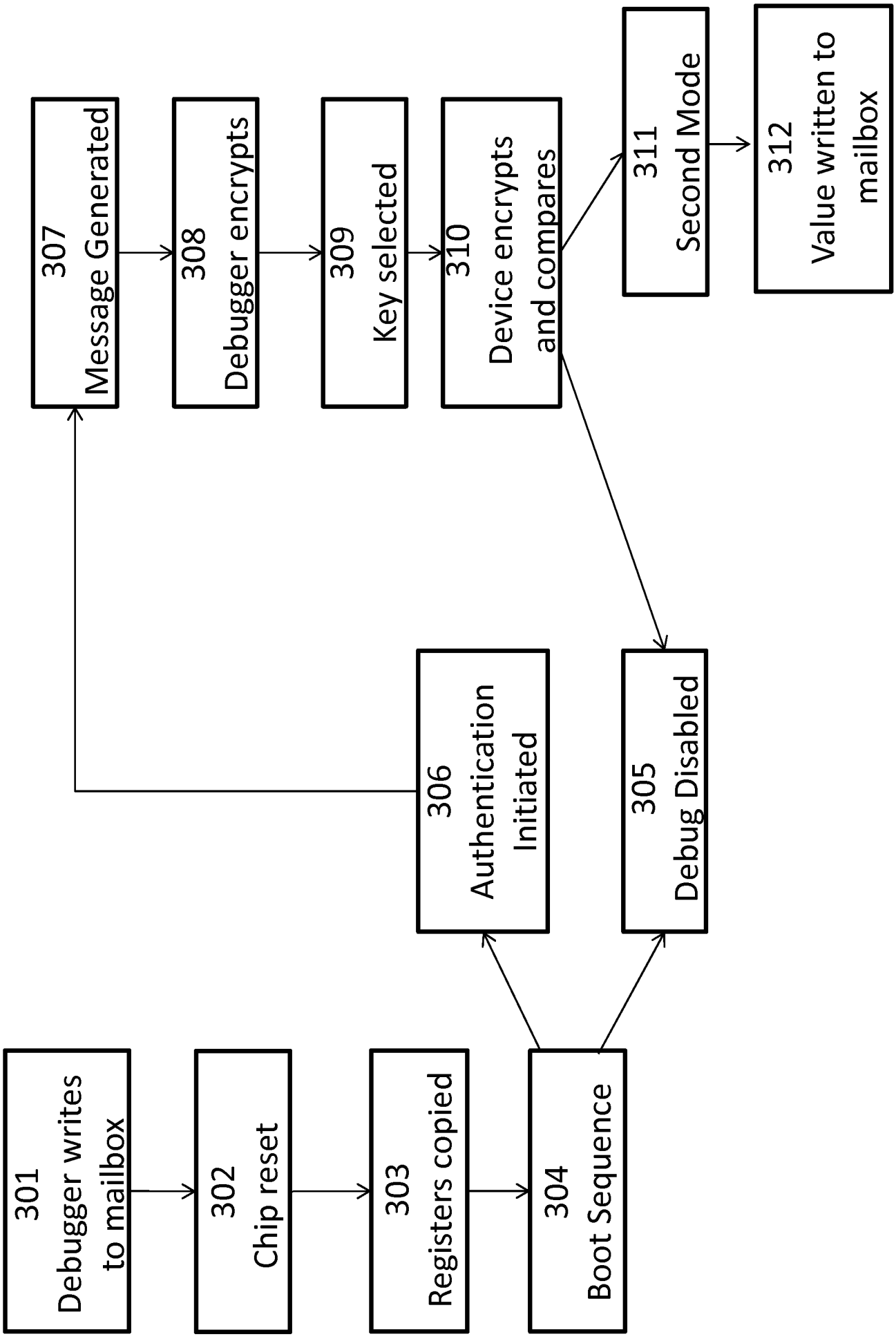
301
Debugger writes to mailbox
302
Chip reset
303
Registers copied
304
Boot Sequence
305
Debug Disabled
306
Authentication Initiated
307
Message Generated
308
Debugger encrypts
309
Key selected
310
Device encrypts and compares
311
Second Mode
312
Value written to mailbox

Figure 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2019/067023

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/44
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JERRY BACKER ET AL: "Secure and Flexible Trace-Based Debugging of Systems-on-Chip", ACM TRANSACTIONS ON DESIGN AUTOMATION OF ELECTRONIC SYSTEMS, ACM, NEW YORK, NY, US, vol. 22, no. 2, 28 December 2016 (2016-12-28), pages 1-25, XP058307069, ISSN: 1084-4309, DOI: 10.1145/2994601 the whole document abstract; figure 1(b) page 8, line 31 - line 32 page 8, line 41 - line 43 page 9, line 39 - line 41 page 9, line 32 - line 33 page 9, line 45 - line 48 page 9, line 48 - page 10, line 1 page 10, line 1 - line 7 page 15, line 1 - line 2 page 15, line 10 - line 14 -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 September 2019	24/10/2019

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Iannuzzi, Silvia

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	page 20, line 24 - line 25 -----	
Y	WO 2017/072500 A1 (NORDIC SEMICONDUCTOR ASA [NO]; SAMUELS ADRIAN JAMES [GB]) 4 May 2017 (2017-05-04) cited in the application the whole document abstract claim 1 page 5, line 29 - line 31 -----	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2019/067023

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2017072500 A1	04-05-2017	CN 108351380 A	31-07-2018
		EP 3368911 A1	05-09-2018
		GB 2543804 A	03-05-2017
		TW 201729094 A	16-08-2017
		US 2018306861 A1	25-10-2018
		WO 2017072500 A1	04-05-2017