



US 20170366186A1

(19) **United States**

(12) **Patent Application Publication**

**Reese**

(10) **Pub. No.: US 2017/0366186 A1**

(43) **Pub. Date: Dec. 21, 2017**

(54) **SELECTIVELY DISABLED OUTPUT**

is a division of application No. 11/435,416, filed on May 16, 2006, now Pat. No. 7,479,798.

(71) Applicant: **Altera Corporation**, San Jose, CA (US)

(72) Inventor: **Laura Reese**, Campbell, CA (US)

(21) Appl. No.: **15/639,869**

(22) Filed: **Jun. 30, 2017**

**Publication Classification**

(51) **Int. Cl.**  
**H03K 19/177** (2006.01)

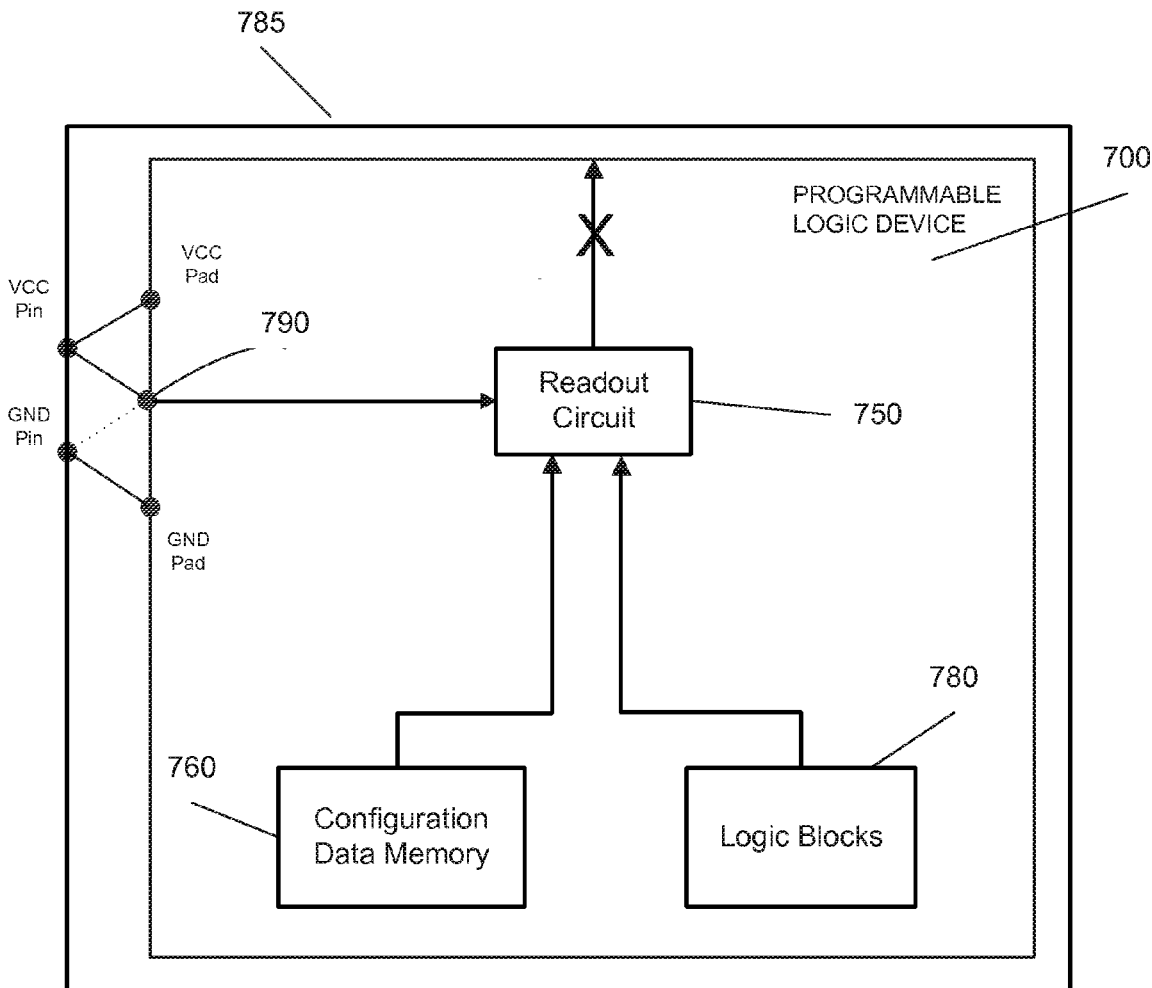
(52) **U.S. Cl.**  
CPC ... **H03K 19/17768** (2013.01); **H03K 19/1776** (2013.01)

**Related U.S. Application Data**

(60) Continuation of application No. 15/400,833, filed on Jan. 6, 2017, which is a continuation of application No. 15/184,921, filed on Jun. 16, 2016, now Pat. No. 9,755,650, which is a continuation of application No. 14/089,364, filed on Nov. 25, 2013, now abandoned, which is a division of application No. 12/332,789, filed on Dec. 11, 2008, now Pat. No. 8,604,823, which

(57) **ABSTRACT**

Circuits, methods, and apparatus are directed to an integrated circuit having a disabling element that can disable a reading of data from the circuit. Once the disabling element is set to not allow a reading of the data, the disabling element cannot be changed to allow a reading of the data. The data may be configuration data or internal data stored within the integrated circuit. Examples of the disabling element include a memory element, a break in a circuit line, and an input pad configuration.



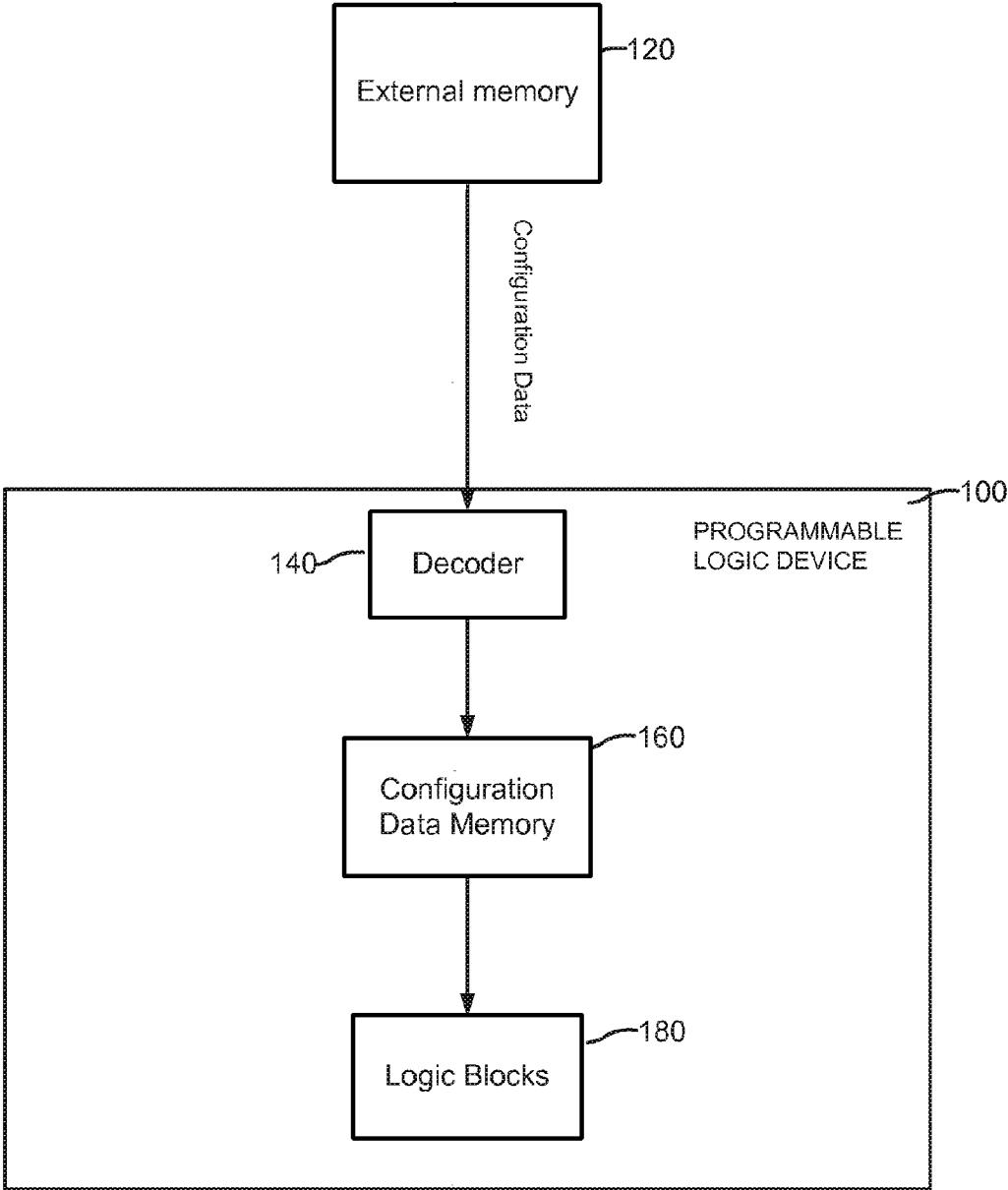


FIG. 1

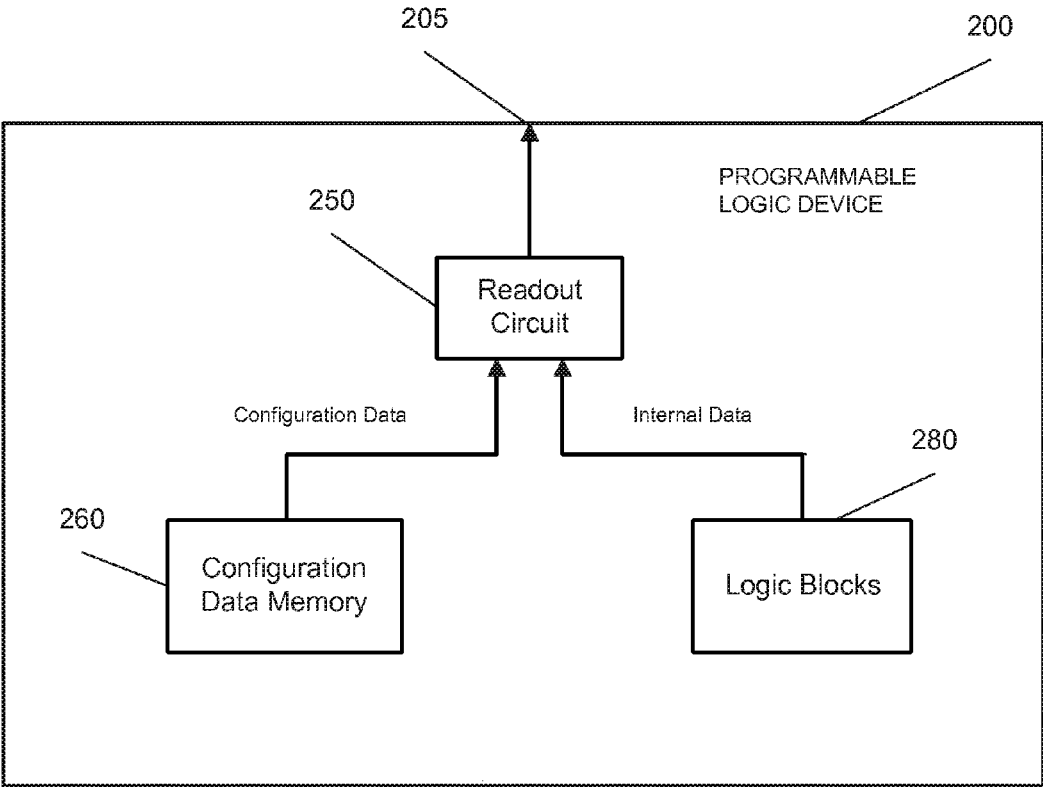


FIG. 2

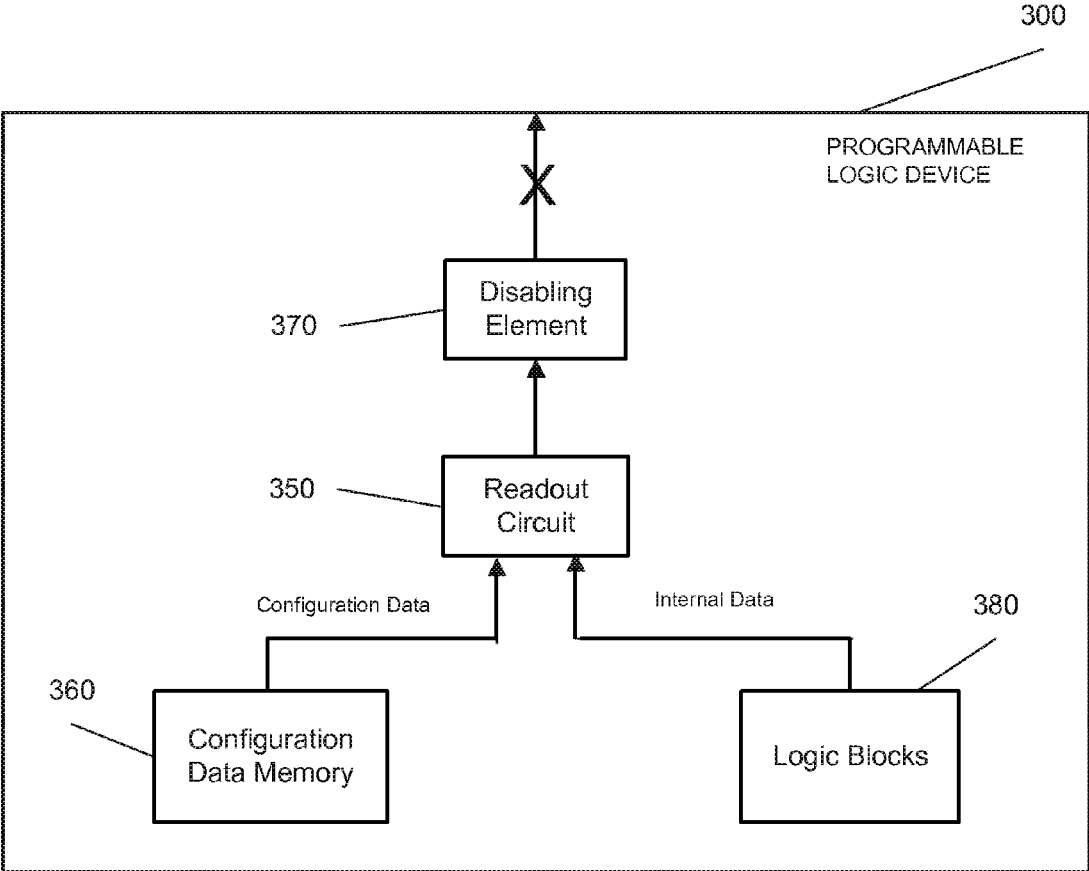


FIG. 3

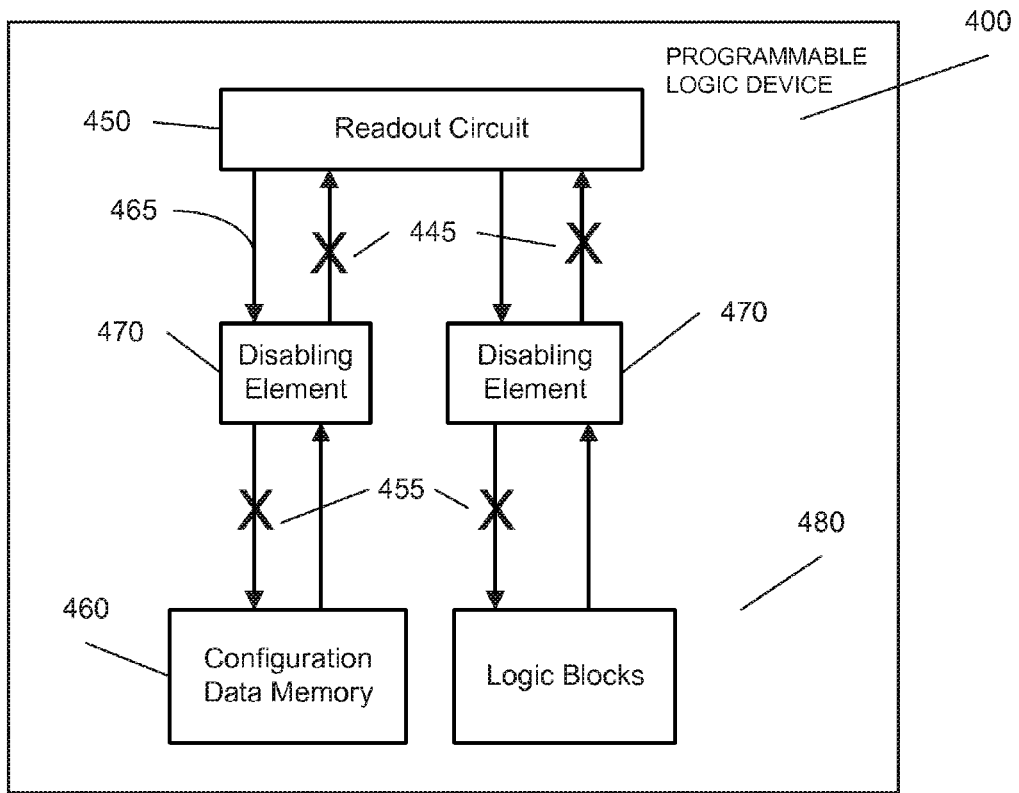


FIG. 4

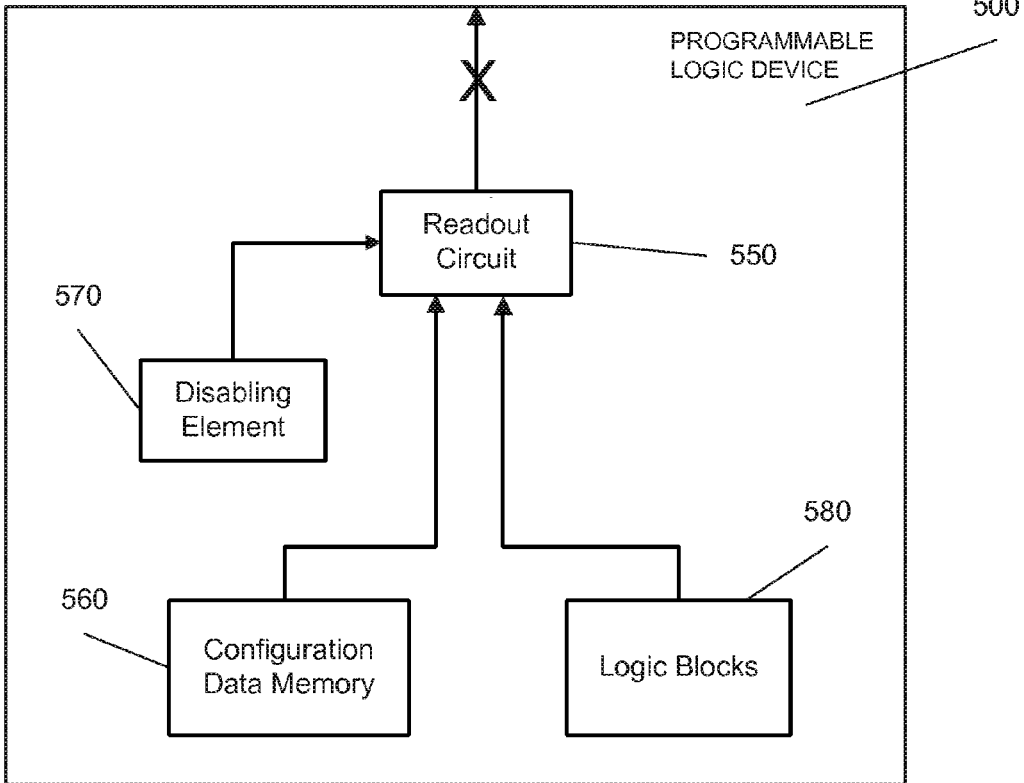


FIG. 5

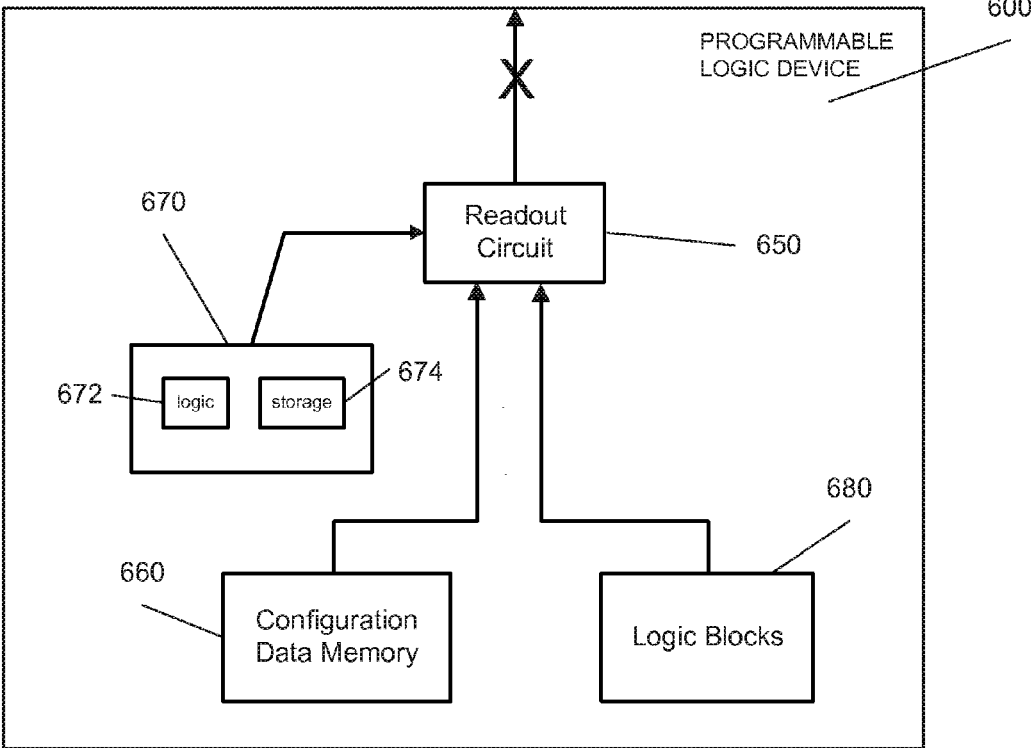


FIG. 6

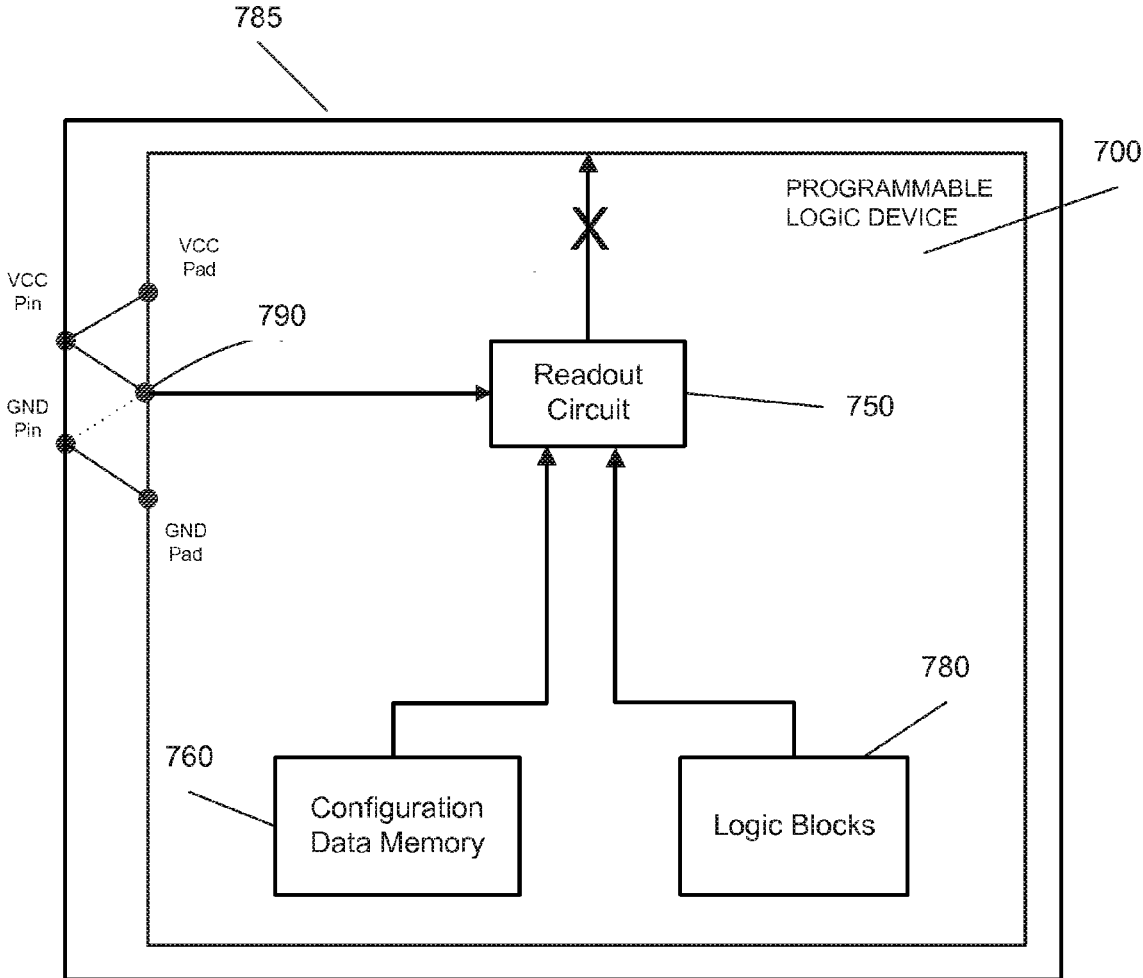


FIG. 7



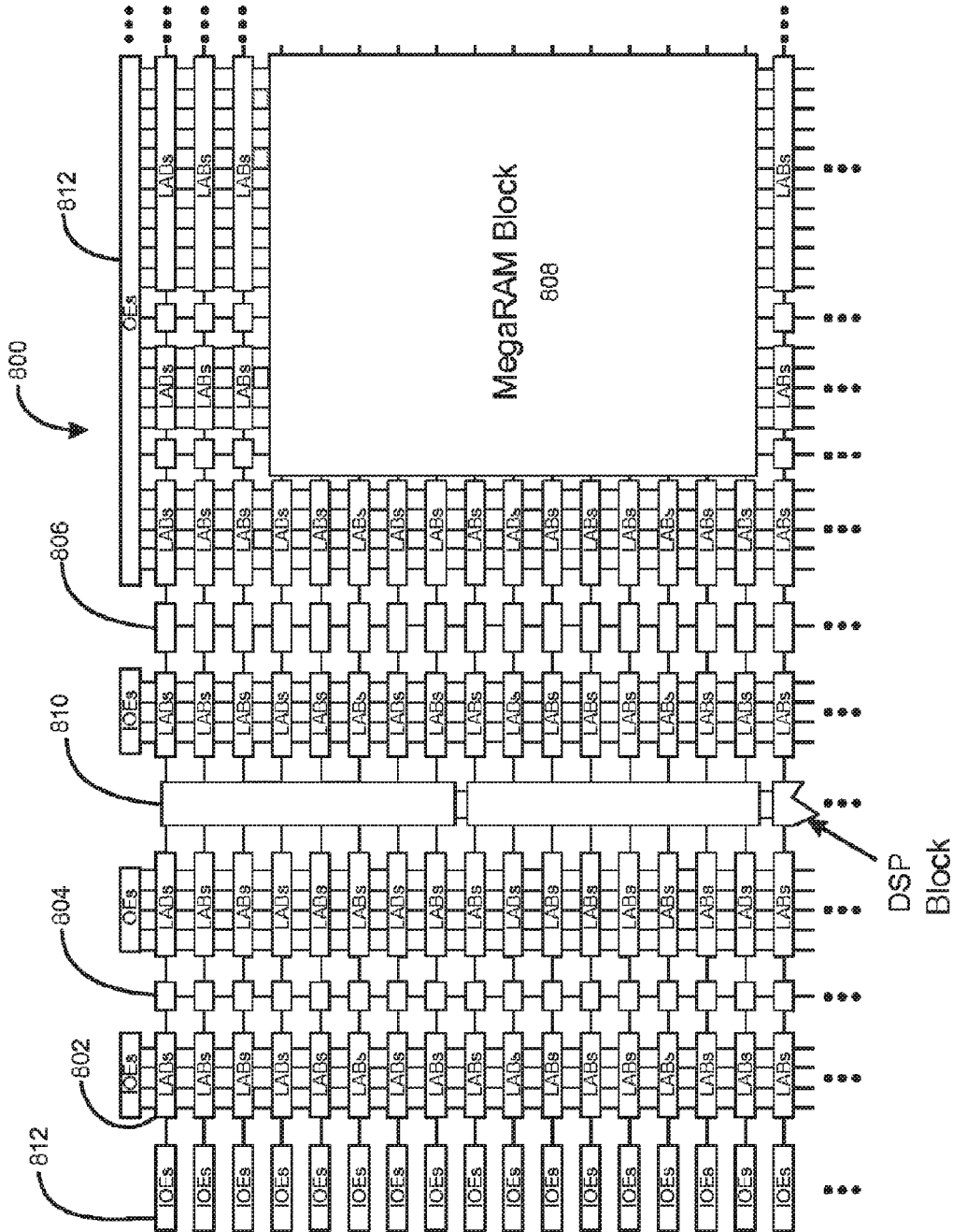


FIG. 8

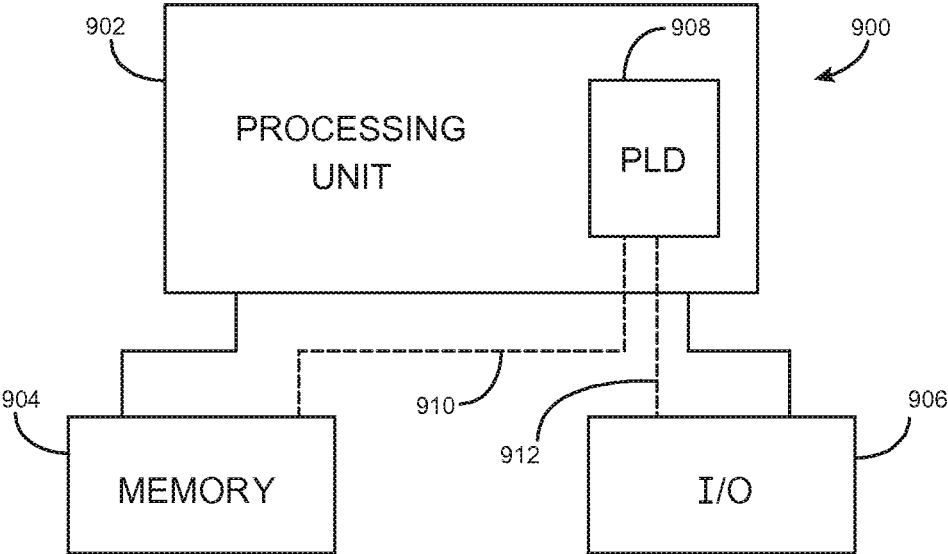


FIG. 9

## SELECTIVELY DISABLED OUTPUT

### CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] Under 35 U.S.C. §120, this application is a continuation of U.S. patent application Ser. No. 15/400,833 filed Jan. 7, 2017, which is a continuation of U.S. patent application Ser. No. 15/184,921 filed Jun. 16, 2016, which is a continuation of U.S. patent application Ser. No. 14/089,364 filed on Nov. 25, 2013, which is a divisional of U.S. patent application Ser. No. 12/332,789 filed on Dec. 11, 2008, now U.S. Pat. No. 8,604,823, which is a divisional of U.S. patent application Ser. No. 11/435,416 filed on May 16, 2006, now U.S. Pat. No. 7,479,798, which are all incorporated by reference herein in their entireties for all purposes.

### BACKGROUND

[0002] The present invention relates to field programmable gate arrays (FPGAs) generally, and more particularly to selectively disabling the reading of output of configuration and/or internal data from an FPGA.

[0003] Field programmable gate array devices are logic or mixed signal devices that may be configured to provide desirable user-defined functions. FPGAs are typically configured by receiving configuration data from a configuration device. This configuration data may be referred to as a configuration bitstream or program object file (POF). The configuration bitstream opens and closes switches formed on an FPGA such that a desired electrical configuration is achieved.

[0004] Modern FPGAs contain hundreds of thousands of logic gates, as well as processors, memories, dedicated analog function blocks, and other circuits. This extensive circuitry requires a correspondingly long configuration bitstream to configure it. For example, 55 Megabits of configuration data are now needed by some FPGAs.

[0005] This configuration data represents an FPGA user design that is the outcome of a huge investment in manpower and research and development costs, often in the million dollar range. To protect this investment, it would be beneficial to prevent reading out of the configuration data, as well as any internal data, particularly internal data that may give away the configuration data.

[0006] However, it can be beneficial in error correction during field use, as well as during design debugging, for an end user to have access to the data on an FPGA. In some applications where the error correction is critical, some FPGA designers and manufacturers are willing to forego security concerns, such as in the medical field, aerospace, and telecommunications.

[0007] Thus, what is needed are circuits, methods, and apparatus that selectively determine a setting for the allowance of access to certain data while preventing the end user from altering this setting. Additionally, it would be beneficial for the selection to be easily implemented, cost effective, and secure.

### SUMMARY

[0008] Accordingly, embodiments of the present invention provide circuits, methods, and apparatus directed to an integrated circuit having a disabling element that allows for

the protection of data on the circuit. Once the disabling element is set to disable access, it cannot be changed to allow access.

[0009] In one exemplary embodiment of the present invention, an integrated circuit has storage circuits capable of storing data. The data may be configuration data for configuring the functionality of the integrated circuit or internal data. The circuit also has a readout circuit for outputting the data. A first state of a disabling element allows a user to read the data, and a second state disables the reading of the data. Once the disabling element is in the second state, the disabling element cannot enter the first state.

[0010] In another exemplary embodiment of the present invention, when the disabling element is in the second state, the data is encrypted before being output thus preventing the reading of the data. A plurality of bits, which are not the data, may also be output.

[0011] In one embodiment, when the disabling element is in the second state, the outputting of the data is prevented. The disabling element may be a break in a circuit line. The disabling element may inhibit the read enable signal to the storage circuits. The disabling element may include many different devices, such as a fuse, an input pad, and a memory element. The memory element may be a one-time programmable memory. The disabling element also may include logic as well as additional memory elements.

[0012] In another exemplary embodiment of the present invention, a method prevents a reading of data on an integrated circuit capable of outputting the data. The method determines whether to allow a reading of the data based on a setting of a disabling element. Once set to not allow a reading of the data, the disabling element cannot be changed to allow a reading of the data. To prevent reading the data, the data may be encrypted before the data is output. The method may also output a plurality of bits instead of the data when the disabling element is set to not allow reading of the data. Output of data may also be prevented. The preventing may be by not transmitting a signal on the integrated circuit. The not transmitting may be due to a blown fuse. The preventing may also include inhibiting a read enable signal.

[0013] In another exemplary embodiment of the present invention, a method is provided to manufacture an integrated circuit. The method provides storage circuits capable of storing data; couples the storage circuits with a readout circuit for outputting the data; couples the readout circuit with an output pad of the integrated circuit; and activates a disabling element such that the reading of the data is permanently disabled. The activating may include providing an encryption code; providing at least one bit for outputting, instead of outputting the data; blowing a fuse; coupling an input pad with a specified voltage; and loading a value into the disabling element. The disabling element may be a memory element. The method also may couple logic with the memory element, where the logic has the loaded value as an input, and couple the logic with the readout circuit.

### BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIG. 1 illustrates a programmable logic device which receives encrypted configuration data;

[0015] FIG. 2 illustrates a programmable logic device which is capable of reading out configuration data;

[0016] FIG. 3 illustrates a programmable logic device having a disabling element after the readout circuit and

capable of preventing access to configuration and/or internal data according to an embodiment of the present invention.

[0017] FIG. 4 illustrates a programmable logic device having a disabling element before the readout circuit and capable of preventing access to configuration and/or internal data according to an embodiment of the present invention.

[0018] FIG. 5 illustrates a programmable logic device having a disabling element coupled with the readout circuit and capable of preventing access to configuration and/or internal data according to an embodiment of the present invention.

[0019] FIG. 6 illustrates a programmable logic device having a disabling element with storage and logic and capable of preventing access to configuration and/or internal data according to an embodiment of the present invention.

[0020] FIG. 7 illustrates a programmable logic device having a disabling element including an input pad and capable of preventing access to configuration and/or internal data according to an embodiment of the present invention.

[0021] FIG. 8 is a simplified block diagram of a programmable logic device that does benefit by incorporating embodiments of the present invention.

[0022] FIG. 9 is a block diagram of an electronic system that does benefit by incorporating embodiments of the present invention.

#### DETAILED DESCRIPTION

[0023] Embodiments of the present invention are directed generally to integrated circuits, particularly programmable logic devices. These devices are typically configured with configuration data containing proprietary designs. This proprietary data may be encrypted before being input into the device, but the device may decrypt the data to implement the design. Embodiments of the present invention selectively disable access to the unencrypted data on the circuit.

[0024] FIG. 1 shows a programmable logic device 100. The external memory 120 contains configuration data, typically containing proprietary designs, that is used to configure the functionality of the logic device 100. The configuration of logic device 100 may occur upon powering up the device, rebooting, or at some other re-programming time. For example, upon powering up, the configuration data will be sent from the external memory 120 to the logic device 100. The configuration data may be encrypted in order to prevent copying when the data is in transit.

[0025] The encrypted data is sent to the logic device 100 where it is decrypted by a decoder 140. The decoder may use an array of one or more fuses, which are left intact or blown so as to create a sequence, such as a key, that decodes the data. The decrypted data is then stored in configuration data memory 160. The configuration data is used to configure the functionality of logic blocks 180. After configuration, the logic blocks 180 may start operating on input data. When in operation, the logic blocks 180 may store internal data, e.g. in data registers, RAM, or other suitable storage. This internal data may reflect specific aspects of the configuration data. Additionally, in non-programmable devices the internal data may reflect proprietary aspects of the circuit design which may be desired to be kept secret.

[0026] In some situations, it is beneficial to have the end user of the logic device 100 to have access to the decrypted configuration data and/or the internal data. For example, during operation of the circuit, an error can occur. Depending on the type of error, an end user might want to examine

the configuration and/or internal data of the logic device 100. This examination might be for initially detecting an error or for the debugging and analysis of the error after detection.

[0027] Such errors include a single event effect (SEE). An SEE is any measurable error which occurs in a circuit, such as device 100, when the circuit is struck by an energetic particle. One type of SEE is a single event upset (SEU) where the charged particle causes unwanted effects in a latch state or memory cell of the circuit. The effect is that a single bit changes state, thus SEUs are referred to as bitflips.

[0028] A bitflip in memory 160 can alter the configuration data, and thus possibly change the functionality of the device 100. SEUs are “soft” errors and can be fixed by reprogramming memory 160 with the configuration data in external memory 120. The external memory 120 is usually more stable, thus more reliable, than the configuration data memory 160.

[0029] As the error may be fixed, the out-of-service time of logic device 100 can be kept very short if the error is detected and debugged quickly. For example, the re-programming may be done automatically on redundant units once an error is detected. Accordingly, in instances where the out-of-service time needs to be very short or practically zero, the end user needs to have access to the configuration data. The internal data may also be affected by an SEU, and it may also be used to detect or debug the error.

[0030] FIG. 2 shows a logic device 200 where the configuration data or the internal data may be read out. The readout of the data is handled by a readout circuit 250. The readout circuit receives the configuration data from the configuration data memory 260 and the internal data from the logic blocks 280. The readout circuit 250 outputs the data through output port 205.

[0031] In some situations, the manufacturer or designer of a circuit does not want an end user to read out such data. This may be for the same reasons that the configuration data may be encrypted, which is to prevent copying or otherwise acquiring proprietary information concerning the circuit. Depending on which internal data is read out, this internal data may also give away functional information as to device 200, to which the manufacturer may not want the end user to have access. Examples of these situations may be in non-critical uses, such as in cell phones where the effects of a malfunction due to an error are minimal. Also, the end user can easily recognize a need to reboot or reprogram the circuit.

[0032] To prevent access to the data in these situations, a manufacturer could decide to not include a readout circuit in the circuit design. However, if the manufacturer would like to produce devices where the data can be read out as well as devices where that the data cannot be read out, then there must be two different designs and possibly two separate manufacturing processes. The additional design would increase the cost of each circuit. Also, circuits that could be read out would not be able to be converted to circuits that cannot be read out.

[0033] Alternatively, a manufacturer could include a readout circuit, but attempt to restrict access to the data using configuration data to configure the device to not allow reading out of the data. This would have some of the same drawbacks of not having a readout circuit. Additionally, an end user could change the encrypted configuration data as it is sent to the circuit to allow access to the unencrypted

configuration data. Thus, the purpose of preventing access to the proprietary designs would be defeated.

[0034] To this end, FIG. 3 shows a logic device 300 according to an embodiment of the invention. The disabling element 370 may be used to prevent an output of the data from the readout circuit 350 by an end user. If the disabling element 370 is activated, an end user does not have access to the configuration data from the memory 360 and the internal data from logic blocks 380. If the disabling unit is not activated, the end user does have access to the configuration data from the memory 360 and the internal data from logic blocks 380. Additionally, if the disabling element 370 is activated, the end user cannot change the disabling element 370 so as to allow access. In some embodiments, the disabling unit may be activated in a post-manufacturing step.

[0035] In one embodiment, the disabling element 370 is a break in a circuit line. The break may include a programmable interconnect that is permanently turned off. A programmable interconnect includes traces or wires that can be programmably connected or disconnected using pass transistors, pass gates, tristate gates, multiplexers, or other appropriate circuit. An interconnect may be turned off by affecting an output or an input, including control inputs such as a gate or enable signal. For example, an additional break in the circuit line from the gate to a voltage line may be made by using or blowing a fuse, polyfuse, a fuse-metal link, a polysilicon link, a zener diode, or an anti-fuse. These elements could also be used to select which input goes to the gate or enable signal.

[0036] Such a break in the electrical connection is not accessible to the end user because the end user does not know where the break occurs, and the break is not easily repaired to restore the electrical connection. If the break is designed to be permanent, the break may be described as a one-time programmable memory. In one embodiment, a polyfuse of the same type that is used for encrypting/decrypting the configuration data is used in the disabling element.

[0037] These fuses are typically polysilicon fuses that are blown or left as a connection. Alternately, the fuses in this figure and the other included examples may be replaced by other nonvolatile memory structures, for example where the means for writing or erasing data stored in the memory structures is permanently disabled once the key is written. For example, a flash memory may be used, where the write enable line is disabled by blowing a fuse once the key is written. In one embodiment, the disabling element in the circuit may be hidden from detection by the end user by obfuscating the disabling element.

[0038] The disabling element may also utilize logic, such as an AND gate, to prevent transmission of a signal. In this example, the break in a circuit line may be due to one input to an AND gate always being low. The value may be low as that is the value stored in a memory element or due to a break in the input circuit line to the AND gate. Other logic circuits may be used along with other suitable constant input values.

[0039] In another embodiment, the disabling element 370 may disable access to the data by encrypting the data before it is output. As encrypted data is output, the end user does not have access to the data. If desired, the encryption key may be given to an end user.

[0040] FIG. 4 shows a logic device 400 according to another embodiment of the invention. The disabling element 470 is positioned before the readout circuit 450. In one embodiment, the configuration data from the memory 460 and/or the internal data from logic blocks 480 are prevented from reaching the readout circuit 450, thus preventing reading out the data from the device 400. The blocking of the data signals is shown with an "X" 445 over the lines from the disabling elements 470 to the readout circuit 450.

[0041] In another embodiment, the disabling element 470 prevents the read enable signal of the readout circuit 450 from reaching the configuration data memory 460 or the logic blocks 480. The blocking of the data signals is shown with an "X" 455 over the lines from configuration data memory 460 or the logic blocks 480 to the disabling elements 470. In another embodiment, the disabling element 470 may disable access to the data by encrypting the data.

[0042] FIG. 5 shows a logic device 500 according to another embodiment of the invention. In this embodiment, the disabling element 570 is not directly in the path of the output of the data. Instead the disabling element 570 acts as an input to the readout circuit 550. Depending on the state of the disabling element 570, the readout of the data may be allowed or not allowed. The disabling element is not accessible by the end user.

[0043] For example, the disabling element 570 could be one or more registers, memory, or other suitable storage device whose contents determine whether reading out of the data will be allowed. The activation of the disabling element 570 may be done at a post-manufacturing step that loads a value into the disabling element that will disable reading out the data. Also, the logic device 500 may be made with the readout circuit 550 disabled and a value may be loaded into the disabling element 570 at a later point in time to enable the readout circuit 550. End user access may be prevented by not providing any mechanism for loading a value into the disabling element 570 via an input pin after activation.

[0044] In one embodiment, the storage is with a fuse or similar device as explained with respect to FIG. 3. The disabling element 570 could be effectively hidden from an end user as the disabling element 570 would not appear in a schematic or connected to an input. In another embodiment, the value that defines the state of the disabling element may be stored in the one or more logic blocks 580. The end user does not have access to the disabling element 570 as the user does not know which logic block or where in the logic block is the one or more bits that control the disabling element. In one embodiment, there is no mechanism for the end user to directly load a value into the specific memory of logic block 580, thus making it even less accessible.

[0045] In another embodiment, the state of the disabling element 570 may determine whether to encrypt the data. The disabling element 570 may also use a value stored within it to encrypt the data, or it may use data stored elsewhere.

[0046] FIG. 6 shows an integrated circuit 600 according to an embodiment of the invention. In one embodiment, the disabling element 670 contains logic elements 672 for manipulating stored values in order to determine whether to allow access to the data. Disabling element 670 may also contain the storage 674 of the values that are used in determining whether to allow access to the data.

[0047] FIG. 7 shows an integrated circuit 700 according to an embodiment of the invention. In this embodiment, the readout circuit 750 is connected to an input pad 790. The

value on an input pad 790 can be used by the readout circuit to determine whether to allow reading of the data. In another embodiment, input pad 790 is connected to additional logic or memory, which is then connected to the readout circuit 750. The value on the input pad may also be used to in other embodiments described herein, such as to cause a break in a circuit line before or after the readout circuit 750.

[0048] In one embodiment, when input pad 790 is connected to a lead VCC pin of the logic device package 785, the reading of the data is not allowed. Connecting input pad 790 to a lead GND pin may allow reading of the data. In another embodiment, when input pad 790 is connected to the GND pin, the reading of the data is not allowed. Connecting input pad 790 to the VCC pin may allow reading of the data. One skilled in the art will recognize the many different combinations available including lead pins with varying voltages.

[0049] In one embodiment, input pad 790 is connected to a lead VCC pin of the logic device package 785 such that the end user cannot access the input pad 790. The VCC pin of the logic device may also be connected to VCC pad, which requires the same voltage input as the one that disables the readout circuit 750. The VCC pad could be the input that powers the device, or other input that is required for the logic device 700 to function properly. Thus, the end user cannot change the disabling element by putting a new voltage on the VCC pin or else the logic device 700 will not function properly. One skilled in the art will recognize the many different combinations available that would prevent changing the disabling element.

[0050] FIG. 8 is a simplified partial block diagram of an exemplary high-density programmable logic device 800 wherein techniques according to the present invention can be utilized. PLD 800 includes a two-dimensional array of programmable logic array blocks (or LABs) 802 that are interconnected by a network of column and row interconnections of varying length and speed. LABs 802 include multiple (e.g., 10) logic elements (or LEs), an LE being a small unit of logic that provides for efficient implementation of user defined logic functions.

[0051] PLD 800 also includes a distributed memory structure including RAM blocks of varying sizes provided throughout the array. The RAM blocks include, for example, 512 bit blocks 804, 4K blocks 806 and an M-Block 808 providing 512K bits of RAM. These memory blocks may also include shift registers and FIFO buffers. PLD 800 further includes digital signal processing (DSP) blocks 810 that can implement, for example, multipliers with add or subtract features.

[0052] It is to be understood that PLD 800 is described herein for illustrative purposes only and that the present invention can be implemented in many different types of PLDs, FPGAs, and the other types of digital integrated circuits.

[0053] While PLDs of the type shown in FIG. 8 provide many of the resources required to implement system level solutions, the present invention can also benefit systems wherein a PLD is one of several components. FIG. 9 shows a block diagram of an exemplary digital system 900, within which the present invention may be embodied. System 900 can be a programmed digital computer system, digital signal processing system, specialized digital switching network, or other processing system. Moreover, such systems may be designed for a wide variety of applications such as telecom-

munications systems, automotive systems, control systems, consumer electronics, personal computers, Internet communications and networking, and others. Further, system 900 may be provided on a single board, on multiple boards, or within multiple enclosures.

[0054] System 900 includes a processing unit 902, a memory unit 904 and an I/O unit 906 interconnected together by one or more buses. According to this exemplary embodiment, a programmable logic device (PLD) 908 is embedded in processing unit 902. PLD 908 may serve many different purposes within the system in FIG. 9. PLD 908 can, for example, be a logical building block of processing unit 902, supporting its internal and external operations. PLD 908 is programmed to implement the logical functions necessary to carry on its particular role in system operation. PLD 908 may be specially coupled to memory 904 through connection 910 and to I/O unit 906 through connection 912.

[0055] Processing unit 902 may direct data to an appropriate system component for processing or storage, execute a program stored in memory 904 or receive and transmit data via I/O unit 906, or other similar function. Processing unit 902 can be a central processing unit (CPU), microprocessor, floating point coprocessor, graphics coprocessor, hardware controller, microcontroller, programmable logic device programmed for use as a controller, network controller, and the like. Furthermore, in many embodiments, there is often no need for a CPU.

[0056] For example, instead of a CPU, one or more PLD 908 can control the logical operations of the system. In an embodiment, PLD 908 acts as a reconfigurable processor, which can be reprogrammed as needed to handle a particular computing task. Alternately, programmable logic device 908 may itself include an embedded microprocessor. Memory unit 904 may be a random access memory (RAM), read only memory (ROM), fixed or flexible disk media, PC Card flash disk memory, tape, or any other storage means, or any combination of these storage means.

[0057] The above description of exemplary embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form described, and many modifications and variations are possible in light of the teaching above. The embodiments were chosen and described in order to best explain the principles of the invention and its practical applications to thereby enable others skilled in the art to best utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. An integrated circuit device comprising:
  - a field programmable gate array (FPGA) fabric that is programmable based at least in part on configuration data stored in configuration memory;
  - a controller configured to program the FPGA fabric by writing the configuration data to the configuration memory, wherein the controller is configured to:
    - receive encrypted data; and
    - determine the configuration data by decrypting the encrypted data based on an encryption key; and
  - a nonvolatile memory communicatively coupled to the controller, wherein a portion of the nonvolatile memory contains the encryption key; and
  - one-time programmable memory configured to selectably permanently write-protect the encryption key.

2. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to selectably permanently disable external access to the encryption key.

3. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to selectably permanently disable erasure of the encryption key.

4. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to selectably permanently disable debugging of the programmable logic device.

5. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to selectably permanently disable access to internal data of the programmable logic device other than configuration data or the encryption key that otherwise could be accessed.

6. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to be activated to permanently disable access post-manufacturing.

7. The integrated circuit device of claim 1, wherein the one-time programmable memory is configured to selectably permanently disable reading of internal data.

8. A method for operating a programmable logic device that has a field programmable gate array (FPGA) fabric, comprising:

storing an encryption key into the programmable logic device, thereby enabling a controller in the programmable logic device to decrypt an encrypted bitstream of configuration data and program the configuration data into configuration memory that controls functionality of the field programmable gate array (FPGA) fabric; and

programming a one-time programmable memory to permanently disable external access to the encryption key.

9. The method of claim 8, wherein the encryption key is stored into nonvolatile memory in the programmable logic device.

10. The method of claim 8, wherein programming the one-time programmable memory to permanently disable external access to the encryption key comprises permanently disabling write access to the encryption key.

11. The method of claim 8, wherein programming the one-time programmable memory to permanently disable external access to the encryption key comprises permanently disabling erasure of the encryption key.

12. The method of claim 8, wherein programming the one-time programmable memory takes place post-manufacturing.

13. The method of claim 8, wherein programming the one-time programmable memory comprises permanently disabling debugging of the programmable logic device.

14. The method of claim 8, wherein programming the one-time programmable memory permanently disables external access to an area of internal data of the programmable logic device.

15. An electronic device comprising memory that stores instructions executable by one or more processors to control

a programmable logic device post-manufacturing, wherein the instructions include instructions to:

inject a user encryption key into a programmable logic device communicatively coupled to the electronic device;

instruct the programmable logic device communicatively coupled to the electronic device to permanently disable erasure of the user encryption key; and

communicate encrypted data from the electronic device to the programmable logic device to enable a controller in the programmable logic device to decrypt the encrypted data to obtain configuration data for a field programmable gate array (FPGA) fabric of the programmable logic device.

16. The electronic device of claim 15, wherein the instructions to inject the user encryption key comprise instructions to inject the user encryption key into a nonvolatile memory of the programmable logic device communicatively coupled to the electronic device.

17. The electronic device of claim 15, wherein the instructions to permanently disable erasure of the user encryption key comprise instructions to permanently disable write access to the user encryption key.

18. The electronic device of claim 15, wherein the instructions comprise instructions to permanently disable external access to the user encryption key.

19. The electronic device of claim 15, wherein the instructions comprise instructions to permanently disable debugging of the programmable logic device.

20. The electronic device of claim 15, wherein the instructions to permanently disable erasure of the user encryption key comprise instructions to program a one-time programmable memory of the programmable logic device that permanently disables write access to the user encryption key.

21. An integrated circuit device comprising:

a field programmable gate array (FPGA) fabric that is programmable based at least in part on configuration data stored in configuration memory;

a controller configured to program the FPGA fabric by writing the configuration data to the configuration memory, wherein the controller is configured to: receive encrypted data; and

determine the configuration data by decrypting the encrypted data based on an encryption key; and

a nonvolatile memory communicatively coupled to the controller, wherein a portion of the nonvolatile memory contains the encryption key; and

one-time programmable memory configured to selectably permanently disable debugging of the programmable logic device.

22. The integrated circuit device of claim 21, wherein the one-time programmable memory is configured to selectably permanently write-protect the encryption key.

\* \* \* \* \*