Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 3003265 A1 2019/11/01

(21) 3 003 265

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2018/05/01

(41) Mise à la disp. pub./Open to Public Insp.: 2019/11/01

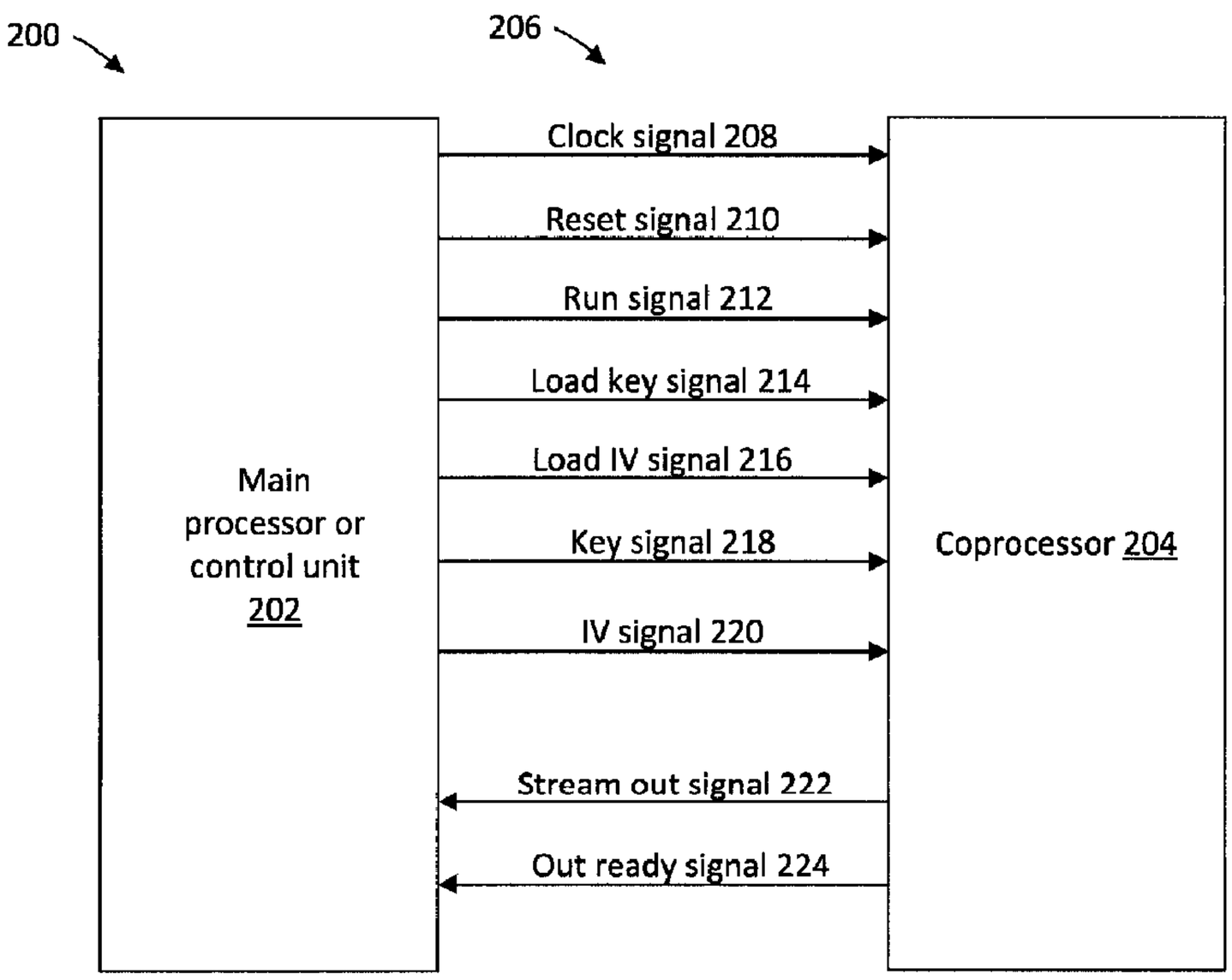
(51) Cl.Int./Int.Cl. *G06F 7/58* (2006.01), *G06F 21/72* (2013.01), *G06F 21/75* (2013.01)

(71) Demandeurs/Applicants:
THE UNIVERSITY OF WESTERN ONTARIO, CA;
VIRGINIA TECH INTELLECTUAL PROPERTIES, INC., US

(72) Inventeurs/Inventors:
TAHA, MOSTAFA, CA;
REYHANI-MASOLEH, ARASH, CA;
SCHAUMONT, PATRICK, US

(74) Agent: HINTON, JAMES W.

(54) Titre : SYSTEMES ET PROCEDES POUR GENERER UNE SEQUENCE DE NUMEROS ALEATOIRES
(54) Title: SYSTEMS AND METHODS FOR GENERATING A RANDOM NUMBER SEQUENCE



(57) Abrégé/Abstract:
A system and method for generating a random number sequence is provided. The system includes a main processor and a coprocessor connected to the main processor. The coprocessor includes a plurality of storage registers each storing a portion of a

CA 3003265 A1 2019/11/01

(21) 3 003 265

(13) A1

(57) Abrégé(suite)/Abstract(continued):
key, and wherein an initialization vector is loaded into the plurality of storage registers, a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register, a multiple input multiple output block to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly connects the result to an input of each of the plurality of shift registers. Outputs of each of the plurality of shift registers are added to generate a random output sequence.

Abstract

A system and method for generating a random number sequence is provided. The system includes a main processor and a coprocessor connected to the main processor. The coprocessor includes a plurality of storage registers each storing a portion of a key, and wherein an initialization vector is loaded into the plurality of storage registers, a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register, a multiple input multiple output block to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly connects the result to an input of each of the plurality of shift registers. Outputs of each of the plurality of shift registers are added to generate a random output sequence.

SYSTEMS AND METHODS FOR GENERATING A RANDOM NUMBER SEQUENCE

Technical Field

[0001] The embodiments disclosed herein relate to systems and methods for generating a random number sequence. In particular, the embodiments disclosed herein relate to systems and methods for generating random number sequences that are resistant against side-channel analysis attacks.

Introduction

[0002] Side-Channel Analysis (SCA) generally is an implementation attack that targets the underlying realization of a cryptographic protocol rather than its mathematical structure. The attack generally depends on harvesting information from side-channel outputs, which include power consumption, electromagnetic radiation, and execution time. Side channel attacks are both passive (i.e. normal operation of the cryptographic protocol is not compromised) and invasive (i.e. nothing is changed, modified, or altered after the attack).

[0003] SCA generally can be realized by either Simple Power Analysis (SPA), where only a single leakage trace is used, or Differential Power Analysis (DPA), where many leakage traces collected under different cipher inputs are used.

[0004] Regular countermeasures against SCA attacks generally depend on either minimizing the signal-to-noise ratio in the leakage trace (hiding), or injecting randomness in the execution of every internal operation (masking). Existing Solutions implementing these countermeasures suffer from a variety of implementation issues. Moreover, hiding and masking do not prevent an attack but rather make it more difficult. Hiding designs can be broken by increasing the number of analyzed traces while masking designs can be broken by higher order and multi-variate attacks.

[0005] Leakage resiliency aims at designing new cryptographic schemes that are inherently secure against SCA attacks. Leakage resiliency can be implemented through a variety of ways. For example, leakage resiliency can be achieved through updating the secret key faster than the expected lifetime in view of SCA. However, this method cannot

directly protect stateless functions. Currently, there are two methods to achieve stateless leakage resiliency: tree structures and key-dependent algorithmic noise.

[0006] Tree structures generally work by applying a non-linear updating function over the entire secret state after accepting every isolated bit of the nonce. However this method of achieving leakage resiliency is computationally intensive and falls behind the performance of regular countermeasures.

[0007] Key-dependent algorithmic noise generally works by using a small part of the public input to activate many hardware modules simultaneously, each with a different part of the secret key. Therefore, after a successful SCA attack, the adversary is left with many key hypotheses ranked first with no tool to put them in the correct order. Hence, a mandatory searching step is required at a high time-complexity.

[0008] The time-complexity after an SCA attack is defined as the number of key hypotheses that need to be tested in order to find the correct secret key. The time-complexity can be expressed directly in a number of trials, or equivalent bits of security (log2(#trials)). Unfortunately, the time-complexities of existing solutions of achieving leakage resiliency through the implementation of key-dependent algorithmic noise fall short of what would be required to achieve robust SCA protection.

[0009] Accordingly, there is a need for a system that is inherently resilient against SCA attacks.

Summary

[0010] According to some embodiments, there is a system for generating a random number sequence. The system includes a main processor and a coprocessor connected to the main processor. The coprocessor includes a plurality of storage registers each storing a portion of a key, and wherein an initialization vector is loaded into the plurality of storage registers, a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register, a multiple input multiple output block connected to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly

connects the result to an input of each of the plurality of shift registers; and wherein outputs of each of the plurality of shift registers are added to generate a random output sequence.

[0011] According to some embodiments, there is a processor for generating a random number sequence. The processor includes a plurality of storage registers each storing a portion of a key, and wherein an initialization vector is loaded into the plurality of storage registers, a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register, a multiple input multiple output block connected to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly connects the result to an input of each of the plurality of shift registers, and wherein outputs of each of the plurality of shift registers are added to generate a random output sequence.

[0012] The plurality of storage registers may be shift registers.

[0013] The plurality of storage registers include at least four shift registers.

[0014] The plurality of shift register lengths may be co-prime to each other.

[0015] The multiple input multiple output block may be a rotating cross-connect.

[0016] The initialization vector may be added modulo-2 with the each output of the feedback circuits.

[0017] The coprocessor may be resilient to side channel attacks.

[0018] According to some embodiments, there is a method for generating a random number sequence. The method includes storing a portion of a key on a plurality of storage registers, loading an initialization vector into the plurality of storage registers, generating, on a plurality of feedback circuits each connected to the plurality of storage registers, an output based on a non-linear function of a state of the corresponding storage register, receiving the initialization vector and pseudo-randomly connecting the result to an input of each of the plurality of storage registers, and generating a random number sequence from the outputs of each of the plurality of storage registers.

[0019] The method may further include initially resetting inputs and outputs of the storage registers.

[0020] The method may further include providing resiliency to side channel attacks.

[0021] The method may further include during each clock cycle computing an output based on a non-linear function on the state of the corresponding shift register.

[0022] Other aspects and features will become apparent, to those ordinarily skilled in the art, upon review of the following description of some exemplary embodiments.

Brief Description of the Drawings

[0023] The drawings included herewith are for illustrating various examples of articles, methods, and apparatuses of the present specification. In the drawings:

[0024] Figure 1 is a block diagram of a system for generating a random number sequence, in accordance with an embodiment;

[0025] Figure 2 is a block diagram of a system for generating a random number sequence, in accordance with an embodiment;

[0026] Figure 3 is block diagram of an internal hardware circuit of a system for generating a random number sequence while processing an input key, in accordance with an embodiment;

[0027] Figure 4 is a block diagram of an internal hardware circuit of a system for generating a random number sequence while processing an initialization vector, in accordance with an embodiment;

[0028] Figure 5 is a block diagram of a rotating cross-connect, in accordance with an embodiment;

[0029] Figure 6 is flow charts of internal states of a system for generating a random number sequence;

[0030] Figure 7 is a diagram of a system for generating a random number sequence, in accordance with an embodiment;

[0031] Figure 8 is an graph of how the equivalent bits of security varies for a given number (N) of registers and a given key size according to one embodiment of the invention;

[0032] Figure 9 is a flow chart illustrating a method for generating a random number sequence, in accordance with an embodiment;

[0033] Figure 10 is a diagram of a one-register NLFSR at clock cycle I; and

[0034] Figure 11 is a diagram of an input initialization vector connected to two registers.

Detailed Description

[0035] Various apparatuses or processes will be described below to provide an example of each claimed embodiment. No embodiment described below limits any claimed embodiment and any claimed embodiment may cover processes or apparatuses that differ from those described below. The claimed embodiments are not limited to apparatuses or processes having all of the features of any one apparatus or process described below or to features common to multiple or all of the apparatuses described below.

[0036] The description sets forth various embodiments of the systems, devices and/or processes via the use of block diagrams, schematics, and examples. Insofar as such block diagrams, schematics, and examples contain one or more functions and/or operations, it will be understood by those skilled in the art that each function and/or operation within such block diagrams, flowcharts, or examples can be implemented, individually and/or collectively, by a wide range of hardware, software, firmware, or virtually any combination thereof. In one embodiment, the present subject matter may be implemented via Application Specific Integrated Circuits (ASICs). However, those skilled in the art will recognize that the embodiments disclosed herein, in whole or in part, can be equivalently implemented in standard integrated circuits, as one or more computer programs executed by one or more computers (e.g., as one or more programs running on one or more computer systems), as one or more programs executed by on one or more controllers (e.g., microcontrollers) as one or more programs executed by one or

more processors (e.g., microprocessors, central processing units, graphical processing units), as firmware, or as virtually any combination thereof, and that designing the circuitry and/or writing the code for the software and or firmware would be well within the skill of one of ordinary skill in the art in light of the teachings of this disclosure.

[0037] Referring to Figure 1, illustrated therein is a system 100 for generating random numbers, including random number sequences, in accordance with an embodiment. The system 100 includes a main processor or control unit 102 communicatively connected to a coprocessor 104 via an interface 106.

[0038] Referring to Figure 2, illustrated therein is a system 200 for generating random numbers, including random number sequences, in accordance with an embodiment. The system 200 includes a main processor or control unit 202 communicatively connected to a coprocessor 204 via an interface 206.

[0039] The interface 206 includes a clock signal 208, a reset signal 212, a run signal 212, a load key signal 214, a load IV signal 216, a key signal 218, and an IV signal 220, which are transmitted from the main processor 202 to the coprocessor 204. The interface 206 includes an out ready signal 222 and a stream out signal 224, which are transmitted from the coprocessor 204 to the main processor 202.

[0040] The clock signal 208 may be a 1 bit signal which represents the system synchronous clock. Furthermore, the clock signal 208 may be a periodic signal. Furthermore the clock signal 208 may be configured to synchronize the timing of all components of the coprocessor 204.

[0041] The reset signal 210 may be a one bit signal. The main processor or control unit 202 may use the reset signal 210 to reset the coprocessor 204 to the coprocessor's initial condition. The coprocessor 204 may start from an initial condition whenever the reset signal 210 is active.

[0042] The run signal 212 may be a one bit signal. The run signal 212 may be used to signal the coprocessor 204 to start processing the key signal 218 and/or the IV signal 220. The run signal 212 may also be used to halt the internal core of the coprocessor 204.

[0043] The key signal 218 may include a plurality of bits and may be loaded into internal registers of the coprocessor 204 over a key data bus wherein the key data bus may be any size equal to or smaller than the key signal 218. As an example, the key signal 218 is a 128 bit signal and the key data bus is an 8 bit bus. The key signal 218 may be loaded into the coprocessor 204 over 128/8 = 16 clock cycles. In another embodiment, the key signal 218 is hard coded into the coprocessor 204.

[0044] The load key signal 214 may be a one bit signal. The load key signal 214 may be used to instruct the coprocessor 204 to accept the key signal 218 as an input. Once raised, the load key signal 214 is kept active as long as the key signal 218 is being loaded, (for example at 128/8 = 16 clock cycles).

[0045] The IV signal 210 may include a plurality of bits and may be loaded into the internal registers of the coprocessor 204 over an IV data bus wherein the IV data bus may be any size equal to or smaller than the IV signal 220. The size of the IV signal 220 determines the maximum number of sessions that can be established using any secret key. The IV signal 220 may be selected to be the same size as the key signal 218. In the illustrated example, the IV signal 220 is a 128 bit signal and the IV data bus is an 8 bit bus. The IV signal 220 may be loaded into the coprocessor 204 over 128/8 = 16 clock cycles. The IV signal 220 may be changed each time the stream out signal 222 is generated.

[0046] The load IV signal 216 may be a one bit signal. The load IV signal 216 may be used to instruct the coprocessor 204 to accept the IV signal 220 as an input. Once raised, the load IV signal 216 is kept high as long as the IV signal 220 is being loaded (for example, 128/8 = 16 clock cycles).

[0047] The out ready signal 224 may be a one bit signal. The coprocessor 204 may use the out ready signal 224 to signal to the main processor 202 that the initialization and warm up phases are complete and the stream out signal 222 may thus be read.

[0048] The stream out signal 222 may include a plurality of bits and may include a random binary stream. The stream out signal 222 may be transmitted from the coprocessor 204 to the main processor 202 over a stream out data bus wherein the stream out data bus may be any size equal to or smaller than the stream out signal 222.

In an example, the stream out signal 222 may be a 128 bit signal and the stream out data bus may be an 8 bit bus. The full stream out signal 222 may be transmitted over 128/8 = 16 clock cycles.

[0049] Referring to Figure 3, illustrated therein is a hardware circuit 300 for a processor or coprocessor, in accordance with an embodiment. The hardware circuit 300 may be part of a coprocessor (e.g., coprocessor 204). The hardware circuit 300 is in a load key state (e.g. 604).

[0050] The hardware circuit 300 includes a plurality of storage registers 308, 310, 312, 314, which may be shift registers. In some cases, the hardware circuit 300 may include any number (N) of storage registers.

[0051] As an example, the hardware circuit 300 includes a first storage register 308 coupled to a first feedback circuit 316, a second storage register 310 coupled to a second feedback circuit 318, a third storage register 312 coupled to a third feedback circuit 320, and a fourth storage register 314 coupled to a fourth feedback circuit 322. As an example, the first storage register 308 may be 31 bits long, the second storage register 310 may be 32 bits long, the third storage register 312 may be 32 bits long, and the fourth storage register 314 may be 33 bits long.

[0052] While the hardware circuit is in a load key state 604, the output of the first storage register 308 is connected to the input of the second storage register 310, the output of the second storage register 310 is connected to the input of the third storage register 313, and the output of the third storage register 313 is connected to the input of the fourth storage register 314.

[0053] A load key signal 324 and a key signal 326 are input into the first storage register 308. A clock signal 328, a reset signal 330, and control logic 332 are input into the plurality of storage registers 308, 310, 312, 314.

[0054] The plurality of storage registers 308, 310, 312, 314 are initialized to a random value then filled with the key signal 326 when the load key signal 324 is raised. For example, during each clock cycle, the storage registers 316, 318, 320, 322 accept 8 bits of the key signal 324 and each register 308, 310, 312, 314 is shifted by 8 bits.

[0055] In other embodiments the number of bits from the key signal 326 and the number of bits that each storage register 308, 310, 312, 314 is shifted by in each clock cycle will differ.

[0056] System security may rise as the number of storage registers 308, 310, 312, 314 increases. The storage registers 308, 310, 312, 314 may include individual bits. The storage registers 308, 310, 312, 314 may also include symbols wherein each symbol represents several bits. The storage registers 308, 310, 312, 314 may be mapped to composite field, which may be particularly advantageous where a high throughput is desired.

[0057] The storage registers 308, 310, 312, 314 may be of any length. System security may rise as the register length increases. The lengths of the plurality of storage registers 308, 310, 312, 314 may be relatively close to each other. The lengths of the plurality of storage registers 308, 310, 312, 314 may be co-prime to each other. The number of storage registers 308, 310, 312, 314 and the length of each register may be selected based on the size of the key signal 324 so that the number of registers x length of each register ≥size of the key signal.

[0058] Referring to Figure 4, illustrated therein is an internal hardware circuit 400 of a processor or coprocessor, in accordance with an embodiment. The hardware circuit 400 may be in a load IV state (e.g., 606), a warm up state (e.g., 608), or a generate output state (e.g., 610).

[0059] The hardware circuit 400 includes a first storage register 402 coupled to a first feedback circuit 404, a second storage register 406 coupled to a second feedback circuit 408, a third storage register 410 coupled to a third feedback circuit 412, and a fourth storage register 414 coupled to the fourth feedback circuit 418.

[0060] In an embodiment, the hardware circuit 400 includes a six bit counter 420, an output multiplexer 422, a two bit counter 424, a multiple input multiple output block 426, a first input XOR gate 428, a second input XOR gate 430, a third input XOR gate 432, a fourth input XOR gate 434, an output XOR gate 436, and an input multiplexer 438.

[0061] The multiple input multiple output block 426 may be for example, a rotating cross-connect. The effect of the output XOR gate 436 may be achieved via a cascade of XOR gates (for example, as in Figure 7).

[0062] The input multiplexer 438 has first input multiplexer input, a second input multiplexer input and an input multiplexer output. The output of input multiplexer 438 is controlled by the load IV signal 440. The first input multiplexer input is connected to the IV signal 442 and the second input multiplexer input is connected to a “0” signal. The input multiplexer output is connected to the first input XOR gate 428, the second input XOR gate 430, the third input XOR gate 432, and the fourth input XOR gate 434. The input multiplexer 438 is configured to connect the first input multiplexer input (IV signal 442) to the input multiplexer output when the load IV signal 440 is “1”. The input multiplexer 438 is configured to connect the second input multiplexer input (“0” signal) to the input multiplexer output when the load IV signal 440 is “0”.

[0063] The first input XOR gate 428 performs the XOR operation on the output of the first feedback circuit 404 and the output of input multiplexer 438 and directs the output to a first input of the rotating cross-connect 426.

[0064] The second input XOR gate 430 performs the XOR operation on the output of the second feedback circuit 408 and the output of input multiplexer 438 and directs the output to a second input of the rotating cross-connect 426.

[0065] The third input XOR gate 432 performs the XOR operation on the output of the third feedback circuit 412 and the output of input multiplexer 438 and directs the output to a third input of the rotating cross-connect 426.

[0066] The fourth input XOR gate 434 performs the XOR operation on the output of the fourth feedback circuit 416 and the output of input multiplexer 438 and directs the output to a fourth input of the rotating cross-connect 426.

[0067] The (two bit, where N=4) counter 424 receives a clock signal 444, a reset signal 446, and control logic 448 as inputs and supplies an output value to the rotating cross connect 410. The two bit counter 424 has a first state “00”, a second state “01”, a third state “10”, and a fourth state “11”. When the reset signal is active the two bit counter

424 enters the first state. The two bit counter 424 then cycles through the four states, entering new state on each clock cycle in the following order: the first state, the second state, the third state, the fourth state, the first state ... etc.

[0068] Referring to Figure 5, illustrated therein is internal architecture of a rotating cross-connect 500 (for example, 426), in accordance with an embodiment. The rotating cross connect 500 includes a first de-multiplexer 502, a second de-multiplexer 504, a third de-multiplexer 506, and a fourth de-multiplexer 508. The rotating cross-connect 500 may connect to four storage registers 512, 514, 516, 518.

[0069] The de-multiplexers 502, 504, 506, 508 receive, respectively, an output 520 of a first input XOR gate (e.g., 428), an output 522 of a second input XOR gate (e.g., 430), an output 544 of third input XOR gate (e.g., 432), and an output 546 of a fourth input XOR gate (e.g., 434). The de-multiplexers 502, 504, 506, 508 receive an output 528 of a two bit counter (e.g., 424). The de-multiplexers 502, 504, 506, 508 send output signals to the four storage registers 512, 514, 516, 518.

[0070] Figure 5 illustrates the rotating cross connect 500 for four registers. In cases where the number of storage registers used is N, the number of de-multiplexers required will also be N and the size of each de-multiplexer will be 1 to N. The number of bits for counter 528 is changed from 2 (N=4) to log N (log is in base of 2).

[0071] Where the level-of-parallelism is an integer multiple of the number of storage registers, the effect of the rotating cross-connect can be achieved by a direct connection between the output of the feedback circuits and input of the storage registers, without the use of any de-multiplexer.

[0072] Referring again to Figure 4, when the two bit counter 424 indicates "00" the output of the first input XOR gate 428 is connected to the input of the first storage register 308, the output of the second input XOR gate 430 is connected to the input of the second storage register 310, the output of the third input XOR gate 432 is connected to the input of the third storage register 312, and the output of the fourth input XOR gate 434 is connected to the input of the fourth storage register 314.

[0073] In the illustrated example, when the two bit counter 404 indicates "01" the output of the first input XOR gate 428 is connected to the input of the second storage register 406, the output of the second input XOR gate 430 is connected to the input of the third storage register 410, the output of the third input XOR gate 432 is connected to the input of the fourth storage register 414, and the output of the fourth input XOR gate 434 is connected to the input of the first storage register 402.

[0074] In the illustrated example, when the two bit counter 404 indicated "10" the output of the first input XOR gate 428 is connected to the input of the third storage register 410, the output of the second input XOR gate 430 is connected to the input of the fourth storage register 414, the output of the third input XOR gate 432 is connected to the input of the first storage register 402, and the output of the fourth input XOR gate 434 is connected to the input of the second storage register 406.

[0075] In the illustrated example, when the two bit counter 404 indicated "11" the output of the first input XOR gate 428 is connected to the input of the fourth storage register 414, the output of the second input XOR gate 430 is connected to the input of the first storage register 402, the output of the third input XOR gate 432 is connected to the input of the second storage register 406, and the output of the fourth input XOR gate 434 is connected to the input of the third storage register 410.

[0076] The plurality of feedback circuits are configured to compute non-linear feedback functions over the internal state of their respective registers. The algebraic normal forms of the feedback functions that may be used in conjunction with the illustrated example where the register lengths are 31, 32, and 33 bits may be those defined by equations 1 to 3 below:

$$\begin{aligned} A_{10}(x_0, x_1, \ldots, x_{30}) = {} & x_0 + x_2 + x_5 + x_6 + x_{15} + x_{17} + x_{18} + x_{20} + x_{25} + x_8x_{18} \\ & + x_8x_{20} + x_{12}x_{21} + x_{14}x_{19} + x_{17}x_{21} + x_{20}x_{22} + x_4x_{12}x_{22} + x_4x_{19}x_{22} \\ & + x_7x_{20}x_{21} + x_8x_{18}x_{22} + x_8x_{20}x_{22} + x_{12}x_{19}x_{22} + x_{20}x_{21}x_{22} \\ & + x_4x_7x_{12}x_{21} + x_4x_7x_{19}x_{21} + x_4x_{12}x_{21}x_{22} + x_4x_{19}x_{21}x_{22} \\ & + x_7x_8x_{18}x_{21} + x_7x_8x_{20}x_{21} + x_7x_{12}x_{19}x_{21} + x_8x_{18}x_{21}x_{22} \\ & + x_8x_{20}x_{21}x_{22} + x_{12}x_{19}x_{21}x_{22}; \end{aligned}$$

Equation 1

$$A_{11}(x_0, x_1, \ldots, x_{31}) = x_0 + x_3 + x_{17} + x_{22} + x_{28} + x_2x_{13} + x_5x_{19} + x_7x_{19} + x_8x_{12} + x_8x_{13} + x_{13}x_{15} + x_2x_{12}x_{13} + x_7x_8x_{12} + x_7x_8x_{14} + x_8x_{12}x_{13} + x_2x_7x_{12}x_{13} + x_2x_7x_{13}x_{14} + x_4x_{11}x_{12}x_{24} + x_7x_8x_{12}x_{13} + x_7x_8x_{13}x_{14} + x_4x_7x_{11}x_{12}x_{24} + x_4x_7x_{11}x_{14}x_{24};$$

Equation 2

$$A_{12}(x_0, x_1, \ldots, x_{32}) = x_0 + x_2 + x_7 + x_9 + x_{10} + x_{15} + x_{23} + x_{25} + x_{30} + x_8x_{15} + x_{12}x_{16} + x_{13}x_{15} + x_{13}x_{25} + x_1x_8x_{14} + x_1x_8x_{18} + x_8x_{12}x_{16} + x_8x_{14}x_{18} + x_8x_{15}x_{16} + x_8x_{15}x_{17} + x_{15}x_{17}x_{24} + x_1x_8x_{14}x_{17} + x_1x_8x_{17}x_{18} + x_1x_{14}x_{17}x_{24} + x_1x_{17}x_{18}x_{24} + x_8x_{12}x_{16}x_{17} + x_8x_{14}x_{17}x_{18} + x_8x_{15}x_{16}x_{17} + x_{12}x_{16}x_{17}x_{24} + x_{14}x_{17}x_{18}x_{24} + x_{15}x_{16}x_{17}x_{24}.$$

Equation 3

[0077] wherein A_{10} is the feedback function used in conjunction with the 31 bit register, A_{11} is the feedback function used in conjunction with the 32 bit registers, and A_{12} is the feedback function used in conjunction with the 33 bit register.

[0078] During each clock cycle, each feedback circuit may execute one of the aforementioned feedback functions over a plurality of bits of a register to compute an output and in the subsequent clock cycle the contents of the register is shifted by one and new bits are connected to the feedback function.

[0079] The plurality of feedback circuits may be configured to compute feedback functions over the internal state of their respective registers. The feedback functions may be non-linear functions. The feedback functions may be selected so that the plurality of registers generate the maximum period.

[0080] There may be two replicas of the feedback function, one connected to the bits in clock cycle N – 1, and one connected to the bits in clock cycle N can be used to compute two outputs in a single clock cycle. The level-of-parallelism may be two. There may be M replicas of the feedback function where m denotes the level-of-parallelism.

[0081] The level of parallelism may also be achieved if the computations (additions and multiplications) in equations 1 to 3 are performed over GF(2^m), where m>1 in which each bit of the registers is replaced with m bits.

[0082] An increase in the level-of-parallelism increases the number of bits that are processed during each clock cycle. The level-of-parallelism may not affect either mathematical security or SCA security and may not change the output of the binary stream.

[0083] The counter 420 (e.g., six bit, based on the number of registers n and number of bits in each register) receives the clock signal 444 and the reset signal 446 as inputs. The six bit counter 420 generates an out ready signal 450. The six bit counter acts as the controller for the output multiplexer 422. The six bit counter 420 is reset to "0" when the reset signal 446 is active. The six bit counter 420 increments by one on each clock cycle. The six bit counter 420 generates the out ready signal 450 when the six bit counter 420 has reached a predetermined value. The six bit counter 420 causes the output multiplexer 422 to relay the output of the output XOR gate 436 as the stream out signal once the six bit counter 420 has reached a predetermined value.

[0084] The output XOR gate 436 may perform modular addition on the outputs of the plurality of storage registers and send the result to the output multiplexer 422.

[0085] The output multiplexer 422 has first output multiplexer input, a second output multiplexer input and an output multiplexer output. The first output multiplexer input is the output of output XOR gate 436, the second output multiplexer input is a "0" signal, and the output multiplexer output is a stream out signal 452. The output of output multiplexer 422 is controlled by the output of the six bit counter 420. The first output multiplexer input (output of output XOR gate 436) is relayed to the output of the output multiplexer 422 when the output of the six bit counter 420 is one of a predetermined set of values. The second output multiplexer input ("0" signal) is relayed to the output of the output multiplexer when the output of the six bit counter 420 is a different one of a predetermined set of values.

[0086] The out ready signal 450 includes a "1" when the output of the six bit counter 420 is one of a predetermined set of values. The out ready signal 450 includes a "0" when the output of the six bit counter 420 is a different one of a predetermined set of values.

[0087] Referring to Figure 6, illustrated therein is a flow chart of internal states 600 of a coprocessor, in accordance with an embodiment. The internal states 600 may be for a coprocessor having, for example, internal circuitry 300, 400.

[0088] The internal states 600 include a reset state 602, a load key state 604, a load IV state 606, a generate output state 610, a halt state 612, and a warm up state 608.

[0089] The coprocessor will enter the reset state 602 when the reset signal is "1". In the reset state 602, a stream out signal, an out ready signal, a six bit counter, and a two bit counter are set to zero. The coprocessor may stay in the reset state 602 until the load key signal is high upon which the coprocessor may enter the load key state 604.

[0090] In the load key state 604, the coprocessor loads a plurality of bits of the key signal during each clock cycle. The coprocessor will stay in the load key state 604 as long as the load key signal is high. While still in the load key state 604, the coprocessor may enter the load IV state 606 if a load IV signal is high. While still in the load key state 604, the coprocessor may enter the halt state 612 when the load key signal is low and the load IV signal is low.

[0091] In the load IV state 606, the coprocessor may load a plurality of bits of the IV signal during each clock cycle. The coprocessor may stay in the load IV state 606 as long as the load IV signal is high. While still in the load IV state 606, the coprocessor may enter the warm up state 608 when the run signal is high. While still in the load IV state 606, the coprocessor may enter the halt state 612 when the load IV signal is low.

[0092] In the warm up state 608, the coprocessor may run for a plurality of clock cycles without generating any output. The coprocessor may run for four clock cycles without generating any output. The number of clock cycles that the coprocessor will spend in the warm up state 608 is defined by equation 4 below:

$$Clock\ cycles\ in\ warmup\ phase = \frac{size\ of\ IV\ signal\ in\ bits\ + size\ of\ longest\ register}{(level - of - parallelism)}$$

Equation 4

[0093] During each clock cycle, the coprocessor may compute the output of a plurality of parallel feedback functions, and shifts each register by a plurality of bits. In the

illustrated example, during each clock cycle, the coprocessor computes the output of 8 parallel feedback functions and shifts each register by 8 bits. During the four clock cycles the system spends in the warm up state 608, 32 bits will be shifted in each register. The coprocessor will stay in the warm up state 608, until the plurality of clock cycles has finished upon which the coprocessor will enter the generate output state 610. While in the warm up state 608, during each clock cycle, the six bit counter will be incremented by one.

[0094] In the generate output state 610, the coprocessor generates a plurality of bits of the random output binary stream during each clock cycle. In the illustrated example, the coprocessor generates 8 bits of the stream out signal 216 during each clock cycle. The coprocessor will stay in the generate output state 610 as long as the run signal is high. The coprocessor will enter the halt state 612 when the run signal is low.

[0095] The coprocessor will enter the halt state 612 whenever the internal state is paused for any reason. While still in the halt state 612, the coprocessor will enter the load key state 604 if the load key signal is high. While still in the halt state 612, the coprocessor will enter the load IV state 606 when the load IV signal is high. While still in the halt state 612, the coprocessor will enter the warm up state 608 when the run signal is high and the six bit counter is less than a predetermined warmup value.

[0096] While still in the halt state 612, the coprocessor will enter the generate output state 610, when the run signal is high and the six bit counter is equal to the predetermined warmup value. In the illustrated example, the predetermined warmup value is four. The predetermined warmup value is defined by Equation 4.

[0097] Referring to Figure 7, illustrated therein is a processor or coprocessor 700 for generating a random number sequence, in accordance with an embodiment. The coprocessor 700 includes a first storage register 702 coupled to a first feedback circuit 704, a second storage register 706 coupled to a second feedback circuit 708, a third storage register 710 coupled to a third feedback circuit 712, and a fourth storage register 714 coupled to a fourth feedback circuit 716.

[0098] The coprocessor 700 includes a rotating cross-connect 718, an input IV signal 720, and a stream out signal 722. The IV signal 720 and the output(s) of the rotating

cross connect 718 are sent through the four XOR operators 724, 726, 728, 730 which feed their respective registers 702, 706, 710, 714.

[0099] The outputs of the first and second registers 702, 706 are sent through a first XOR gate 732. The output of the first XOR gate 732 and the output of third register 710 are sent through a second XOR gate 734. The output of the second XOR gate 734 and the output of fourth register 714 is sent through a third XOR gate 736. The output of the third XOR gate 736 is the stream out signal 722.

[0100] Figure 8 provides a graphical 800 of $E(T_{gen,b})$ for different values of N and L in the case where the key size = N x L.

[0101] Referring to Figure 9, illustrated therein is a method 900 of generating a random number sequence, in accordance with an embodiment. At 902, a coprocessor is initially reset. When the coprocessor is reset, a stream out signal, an out ready signal, a six bit counter, and a two bit counter may all be set to zero.

[0102] At 904, a key is loaded into a plurality of shift registers, wherein each of a plurality of shift registers is configured to hold a portion of the key.

[0103] At 906, an initialization vector is loaded into the plurality of shift registers. Each of the plurality of shift registers is coupled to a feedback circuit which generate an output based on a non-linear function on a state of the feedback circuit's coupled shift register during each clock cycle. The initialization vector is added modulo-2 with the each output of the feedback circuits. The initialization vector is sent through a rotating cross connect which pseudo-randomly connects the result of each modulo-2 addition to a distinct input of each of the plurality of shift registers.

[0104] At 908, there is a warm up. In the warm up, during each clock cycle each feedback circuit computes an output based on a non-linear function on the state of the feedback circuit's coupled shift register. The output is transmitted to a rotating cross connect which pseudo-randomly connects each of the outputs to the input of one of the plurality of shift registers.

[0105] At 910, a random number sequence is generated. During each clock cycle each feedback circuit includes an output based on a non-linear function on the state of

the feedback circuit's coupled shift register. The output is sent to the rotating cross connect which pseudo-randomly connects each of the outputs to the input of one of the plurality of shift registers. The outputs of each of the plurality of shift registers is added modulo-2 to generate a random output sequence.

[0106] Figure 10 is an illustration of a shift register 1006 coupled to a feedback function 1004. In the illustrated example, the input to the shift register 1006 is connected to the output of an XOR gate 1012 which accepts the output of the feedback function 1004 and an initialization vector 1002 as inputs. In the illustrated example, each clock cycle causes a binary shift-left by one bit while the internal state of the shift register 1006 is appended by the addition modulo-2 between the output of the feedback function 1004 and a bit from the initialization vector 1002. The objective of an SCA attack would be to retrieve the internal state of the shift register 1006.

[0107] The power consumption of the shift register 1006 of length L in clock *i*, represented by P_i depends on the change in the value of its internal register (R). Equation 5 shows the power consumption P_i as derived from the Hamming Distance (HD) power model.

$$P_i = HD\left(R^i,\ R^{i-1}\right) = HW\left(R^i \oplus\ R^{i-1}\right) = HW((R_1^{i-1}, \dots ,\quad R_{L-1}^{i-1}, (F\left(R^{i-1}\right) \oplus \mathrm{IV}_i) \oplus R^{i-1}$$

Equation 5

where HD is the Hamming Distance function, HW is the Hamming Weight function, and IV_i represents the bit number i of the initialization vector.

[0108] For linear feedback shift registers, the difference (PD) between the power at clock i and at clock $i + 1$ is shown by equation 6.

$$PD = P_i - P_{i+1} \in \{-1,0,1\}$$

Equation 6

Here, if the activity in R_0 of clock *i* equals the activity in R_{L-1} of clock *i* + 1, PD will equal 0. Otherwise PD $\in \{-1,1\}$. This observation can be used to recover the internal state.

[0109] This technique can be extended to break NLFSRs through using equation 6 to directly recover relations between neighbor bits in the internal state (e.g., if PD = -1, R_0 had not changed its value between clock *i* and clock *i* + 1, while R_{L-1} must have flipped.

[0110] The attack can be further improved by controlling (or monitoring) the IV. Here, instead of analyzing the difference in P between consecutive clock cycles, the difference in P is analyzed between different input values (IV_0 = 0) and (IV_0 = 1), at the same clock cycle. The difference in P between the two different input values can be used to recover the relation between every two neighbor bits in the structure, on relation as a time as the difference in P depends exclusively on the activity in R_{L-1}.

[0111] The resiliency of a system against the attacks described above can be improved through connecting the input initialization vector (IV) to many registers at the same time in order to raise the level of key-dependent algorithmic noise and render the aforementioned attacks much more computationally intensive to implement.

[0112] Referring to Figure 11, illustrated therein is a first shift register 1108, a first feedback function 1104, a second shift register 1110, a second feedback function 1106, a first XOR gate 1112, a second XOR gate 1114, and an input signal 1102.

[0113] Here N=2, the first feedback function 1104 is coupled to the first shift register 1108 and is configured to compute a first output based on the state of the shift register 1108. Here, the first XOR gate 1112 takes the first output and the input signal 1102 as inputs and outputs a first value to the input of the first shift register 1108. Similarly, the second feedback function 1106 is coupled to the second shift register 1110 and is configured to compute a second output based on the state of the second shift register 1110. Here, the second XOR gate 1112 takes the second output and the input signal 1102 as inputs and outputs a second value to the input of the second shift register 1108.

Here, the power consumption during the first clock cycle with input IV_0, following the HD power model, is shown by equation 7:

$$P(IV_0) = HW(R0^1 \oplus R0^0) + HW(R1^1 \oplus R1^0)$$

$$= HW\left((R0_1^0, \ldots, R0_{L-1}^0, F0(R0^0) \oplus IV_0) \oplus R0^0\right) + \text{HW}((R1_1^0, \ldots, R1_{L-1}^0, \text{F1}(R1^0) \oplus IV_0) \oplus R1^0)$$

Equation 7

At a fixed value of the initial state ($R0^0 \quad R1^0$) and changing the input binary stream IV, the IV-dependent power consumption is illustrated in equation 8.

$$P(IV_0) = F0(R0^0) \oplus IV_0 \oplus R0_{L-1}^0 + F1(R1^0) \oplus IV_0 \oplus R1_{L-1}^0$$

Equation 8

With two registers, the power consumption will have three different levels. There is thus a slight improvement with two registers in parallel over a single register. Analysis of the first clock cycle will result in three, instead of two, levels of power consumption which has two effects. Firstly, the allowable level of noise is reduced as the adversary will have to distinguish between three levels of power consumption. Secondly, one particular level when P(0) = P(1), will show up in two cases, raising the required searching time (time-complexity) after a successful attack.

Generally, the average time-complexity after a successful SCA attack for a system analogous to that in Figure 11 employing N registers each with L stages and where the input binary stream (IV) is connected to all the N registers is given by equation 9 below:

$$E(T_{gen,b}) = (2-L)N + (L-1)\sum_{i=2}^{2N} \alpha \; x \; log_2(i) bits, where \; \alpha = \begin{cases} -1 \; if \; i \leq N \\ +1 \; otherwise \end{cases}$$

Equation 9

[0114] While the above description provides examples of one or more apparatus, methods, or systems, it will be appreciated that other apparatus, methods, or systems may be within the scope of the claims as interpreted by one of skill in the art.

Claims:

1. A system for generating a random number sequence, the system comprising:

a main processor; and

a coprocessor connected to the main processor, the coprocessor comprising:

a plurality of storage registers each storing a portion of a key, and wherein an initialization vector is loaded into the plurality of storage registers;

a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register;

a multiple input multiple output block connected to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly connects the result to an input of each of the plurality of shift registers; and

wherein outputs of each of the plurality of shift registers are added to generate a random output sequence.

2. The system of claim 1, wherein the plurality of storage registers are shift registers.

3. The system of claim 1, wherein the plurality of storage registers include at least four shift registers.

4. The system of claim 2, wherein the plurality of shift register lengths are co-prime to each other.

5. The system of claim 1, wherein the multiple input multiple output block is a rotating cross-connect.

6. The system of claim 1, wherein the initialization vector is added modulo-2 with the each output of the feedback circuits.

7. A processor for generating a random number sequence, the processor comprising:

a plurality of storage registers each storing a portion of a key, and wherein an initialization vector is loaded into the plurality of storage registers;

a plurality of feedback circuits each connected to the plurality of storage registers, wherein the plurality of feedback circuits generate an output based on a non-linear function of a state of the corresponding storage register;

a multiple input multiple output block connected to the plurality of storage registers, wherein the multiple input multiple output block receives the initialization vector and pseudo-randomly connects the result to an input of each of the plurality of shift registers; and

wherein outputs of each of the plurality of shift registers are added to generate a random output sequence.

8. The processor of claim 7, wherein the plurality of storage registers are shift registers.

9. The processor of claim 7, wherein the plurality of storage registers includes at least four shift registers.

10. The processor of claim 8, wherein the plurality of shift register lengths are co-prime to each other.

11. The processor of claim 7, wherein the initialization vector is added modulo-2 with the each output of the feedback circuits.

12. The processor of claim 1, wherein the coprocessor is resilient to side channel attacks.

13. A method for generating a random number sequence, the method comprising:

storing a portion of a key on a plurality of storage registers;

loading an initialization vector into the plurality of storage registers;

generating, on a plurality of feedback circuits each connected to the plurality of storage registers, an output based on a non-linear function of a state of the corresponding storage register;

receiving the initialization vector and pseudo-randomly connecting the result to an input of each of the plurality of storage registers; and

generating a random number sequence from the outputs of each of the plurality of storage registers.

14. The method of claim 13, wherein the plurality of storage registers are shift registers.

15. The method of claim 13, wherein there the plurality of storage registers include at least four shift registers.

16. The method of claim 14, wherein the plurality of shift register lengths are co-prime to each other.

17. The method of claim 13, further comprising initially resetting inputs and outputs of the storage registers.

18. The method of claim 13, wherein the initialization vector is added modulo-2 with the each output of the feedback circuits.

19. The method of claim 13, further comprising providing resiliency to side channel attacks.

20. The method of claim 13 further comprising during each clock cycle computing an output based on a non-linear function on the state of the corresponding shift register.

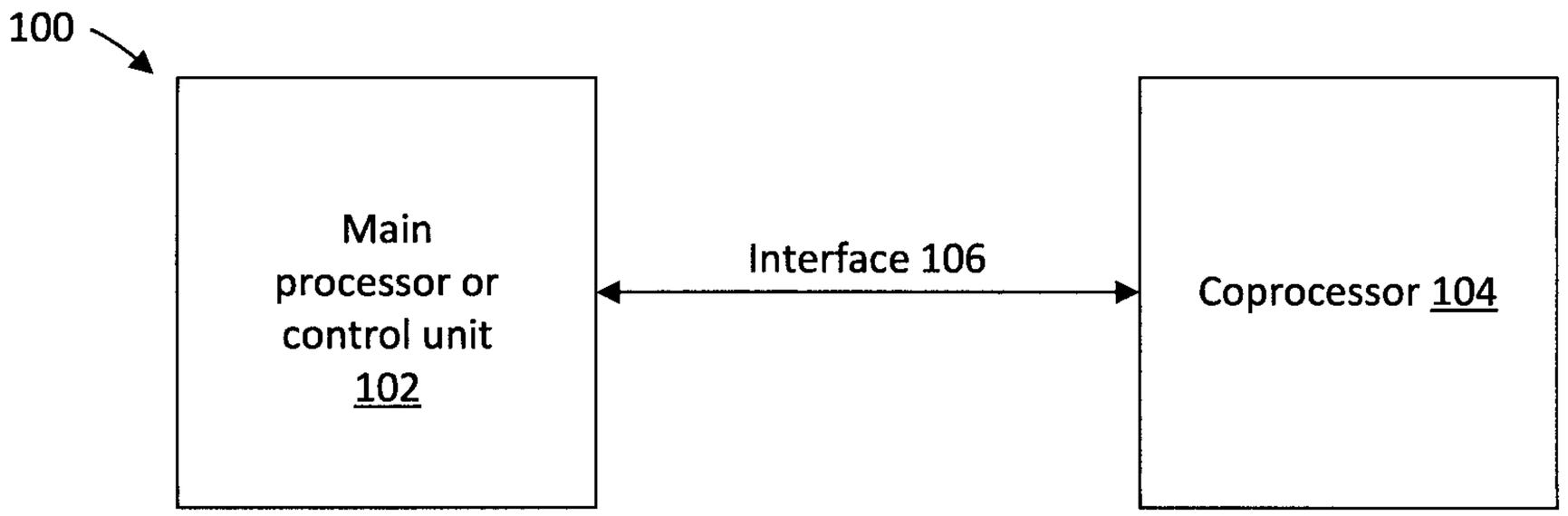


Figure 1

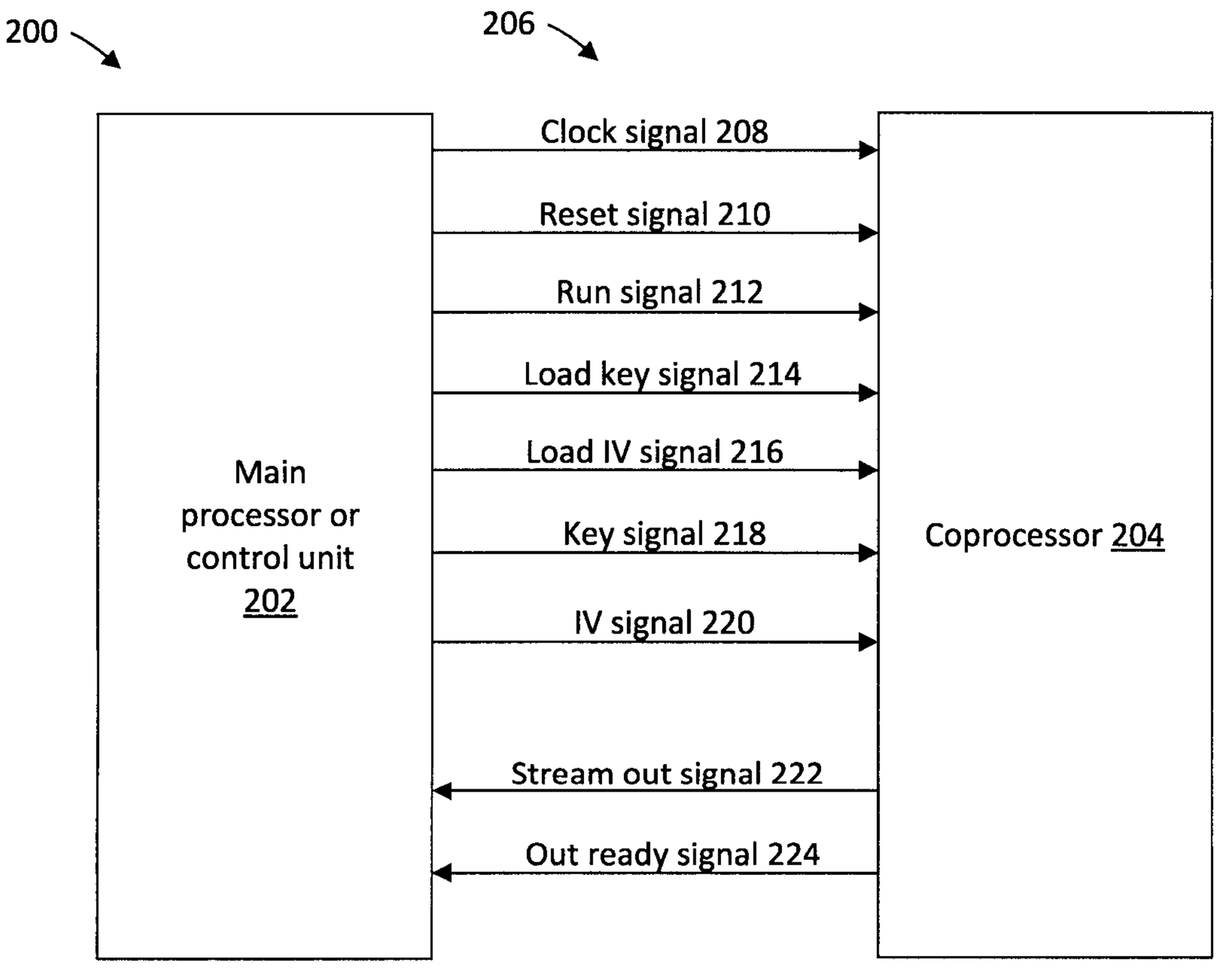


Figure 2

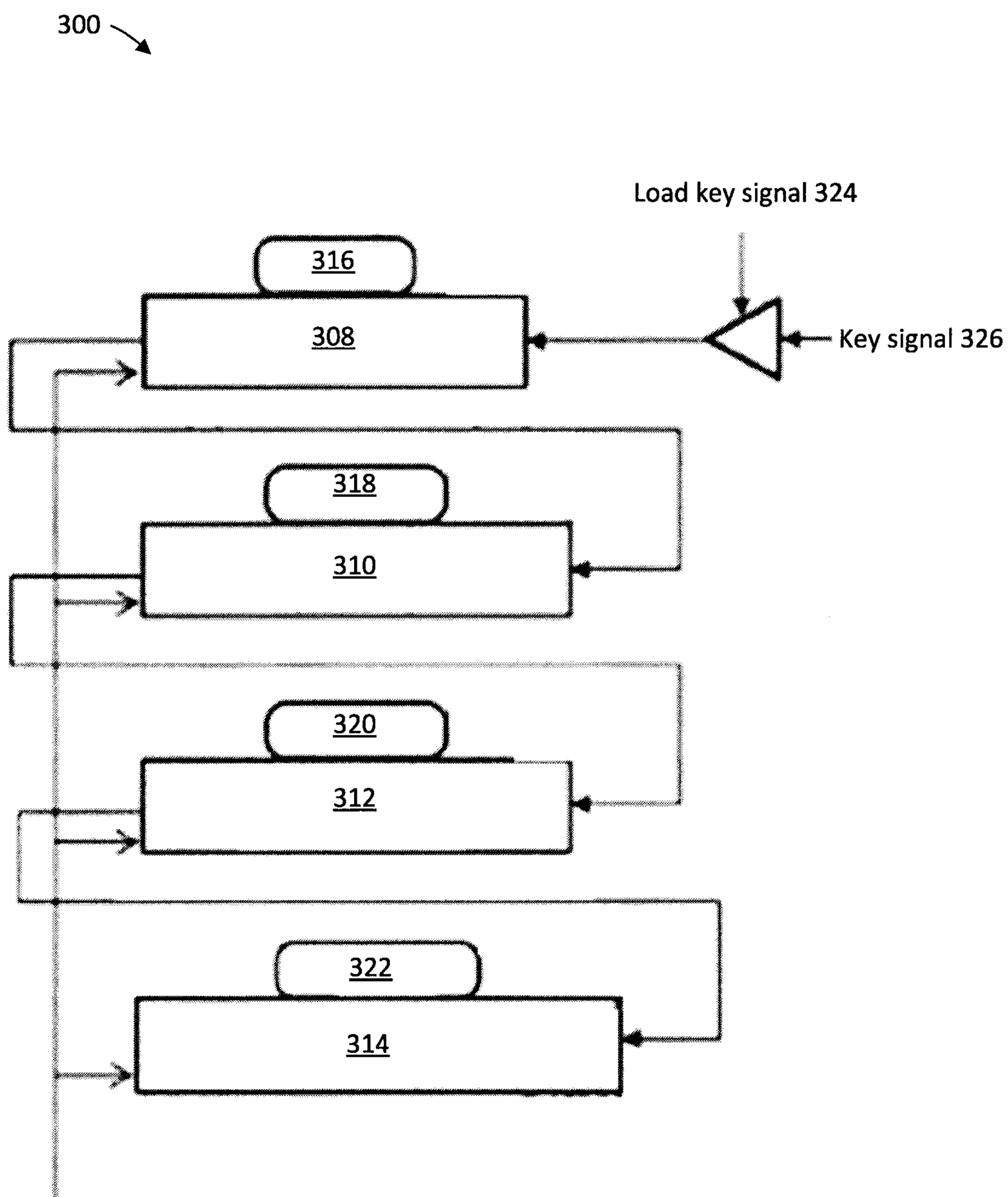

300
Load key signal 324
316
308
Key signal 326
318
310
320
312
322
314
Clock signal 328,
Reset signal 330,
Control logic 332

Figure 3

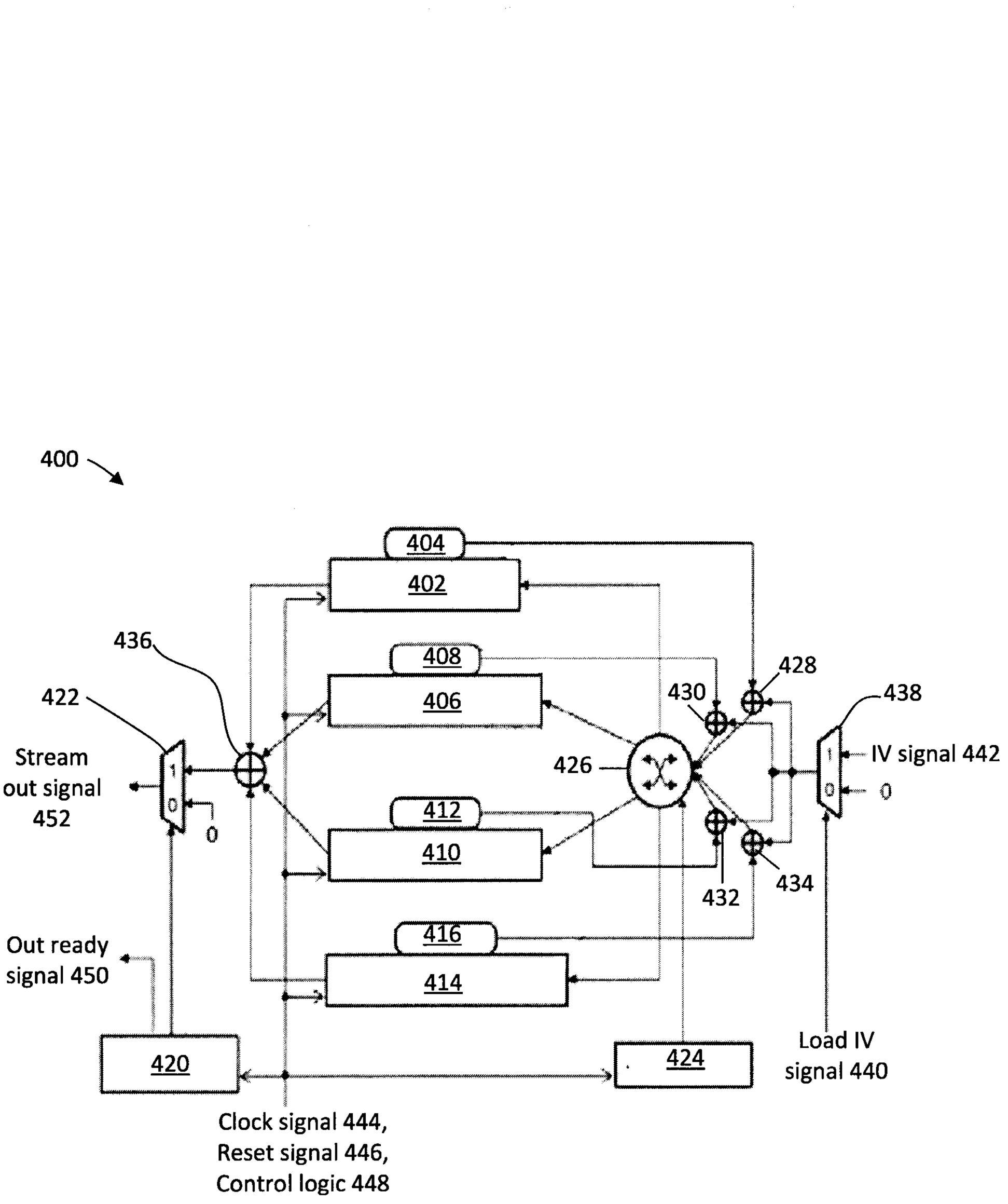

400
404
402
436
408
406
422
428
430
438
Stream
out signal
452
426
IV signal 442
0
412
410
432
434
416
414
Out ready
signal 450
420
424
Load IV
signal 440
Clock signal 444,
Reset signal 446,
Control logic 448

Figure 4

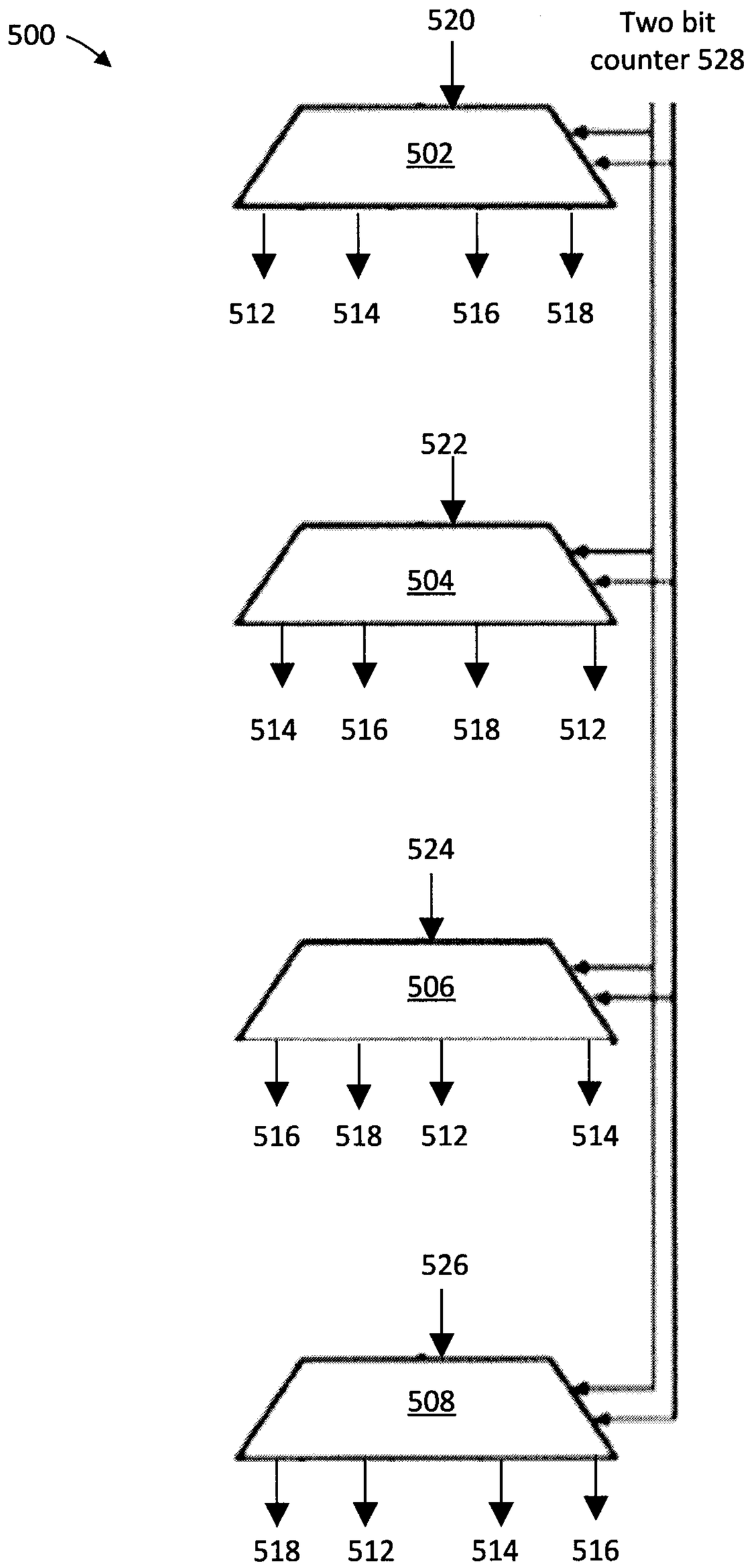

500
520
Two bit
counter 528
502
512
514
516
518
522
504
514
516
518
512
524
506
516
518
512
514
526
508
518
512
514
516

Figure 5

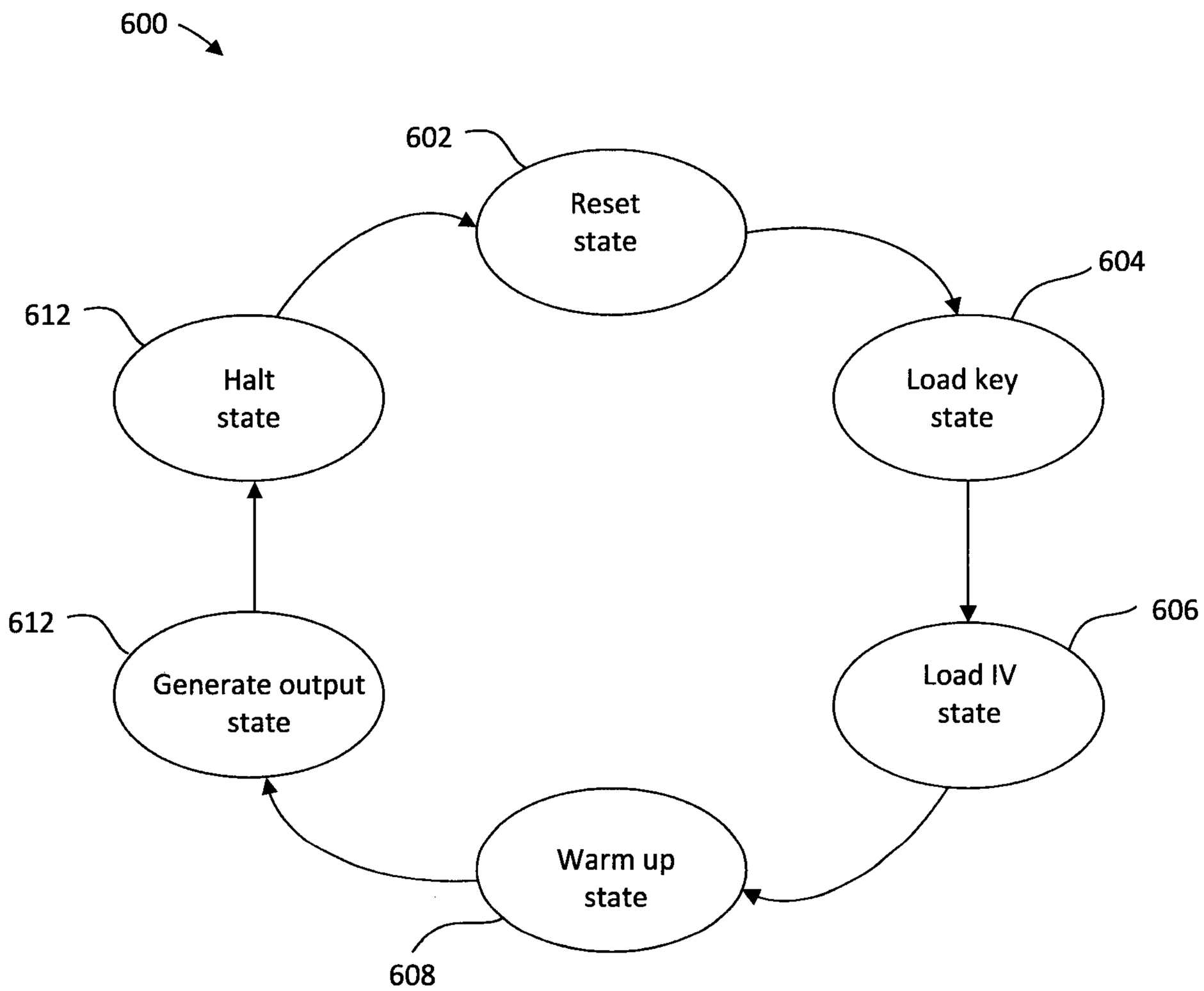

600
602
Reset state
604
612
Halt state
Load key state
612
Generate output state
606
Load IV state
Warm up state
608

Figure 6

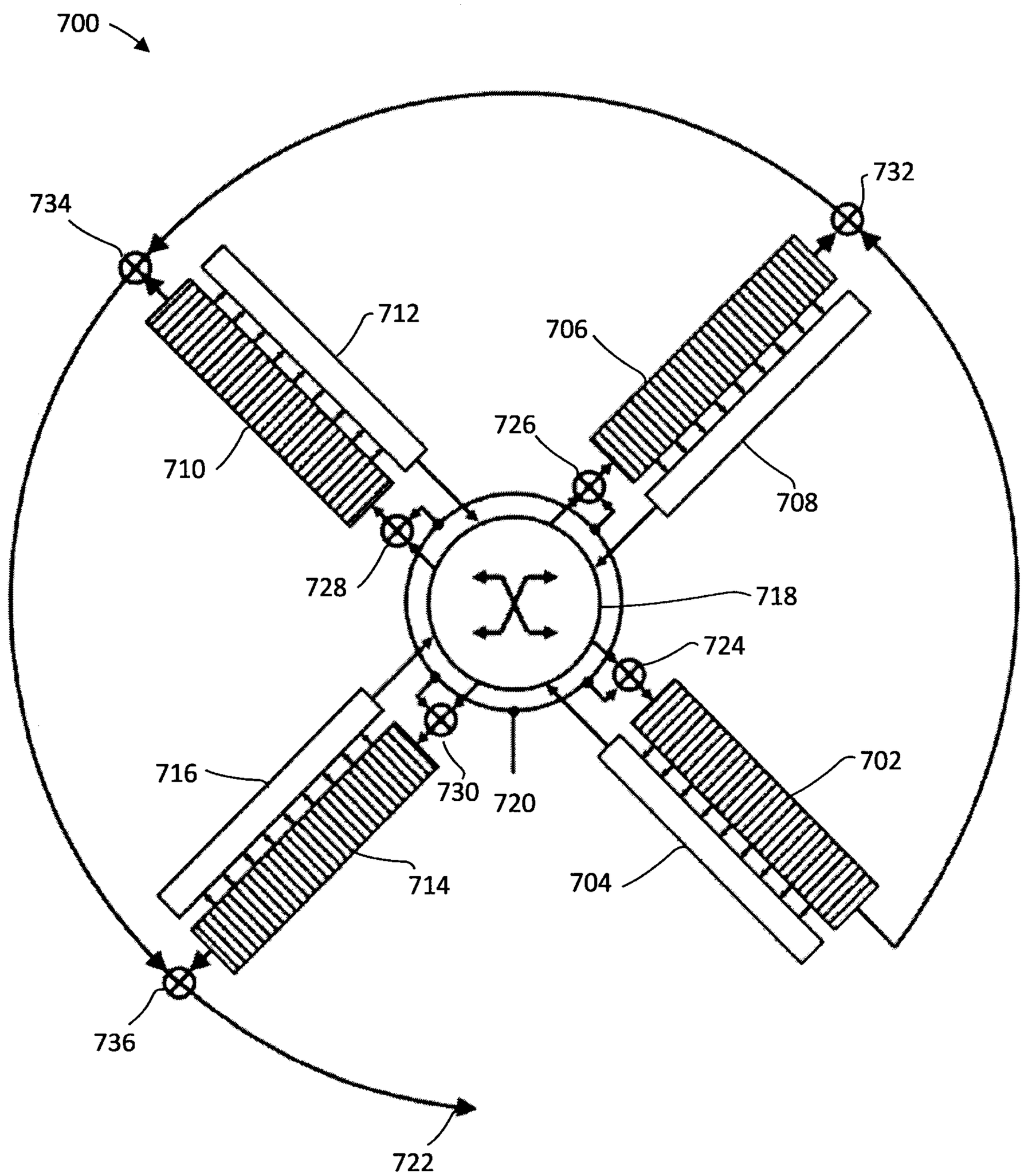

700
732
734
712
706
710
726
708
728
718
724
716
730
720
702
714
704
736
722

Figure 7

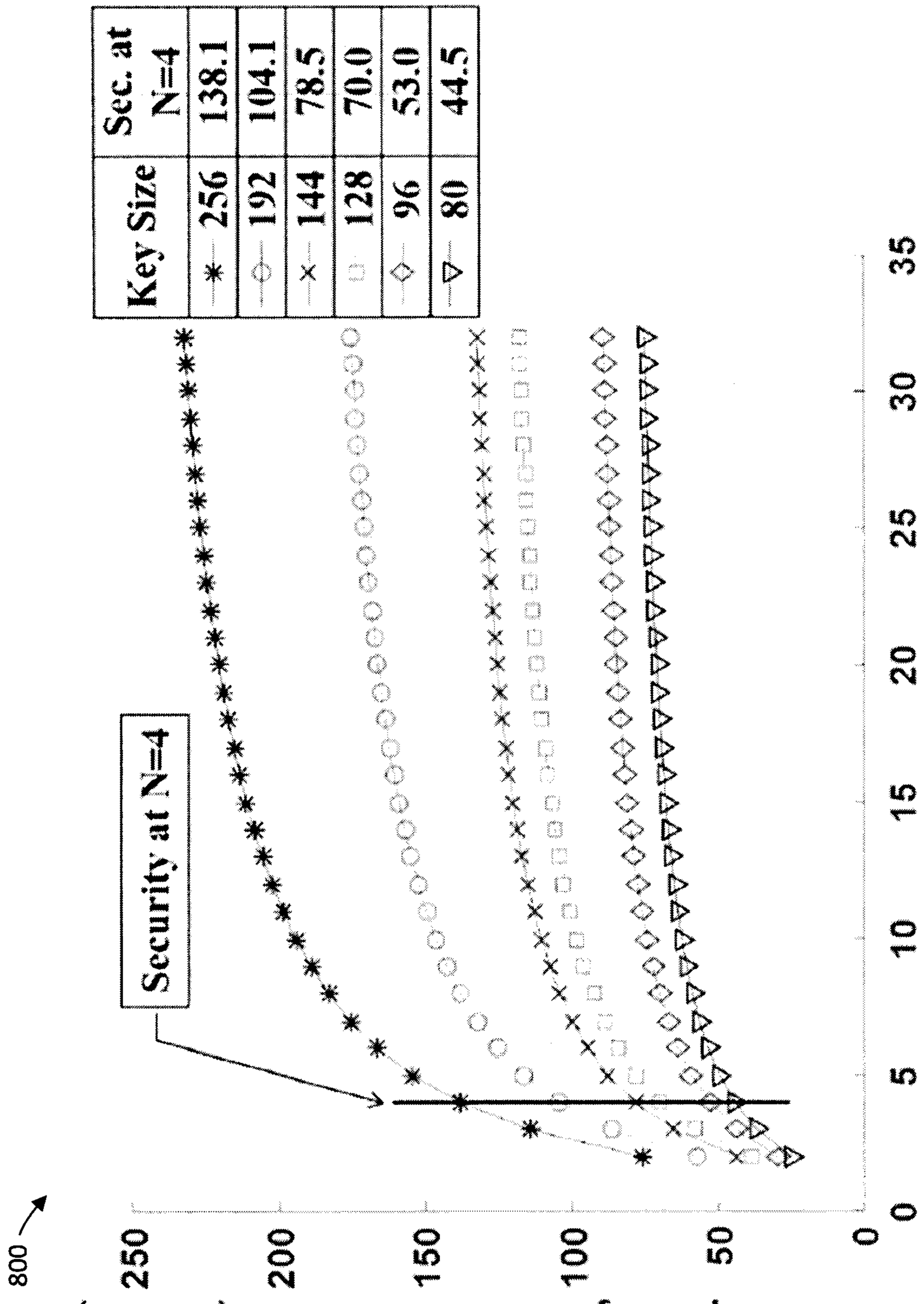

Key Size
256
192
144
128
96
80
Sec. at N=4
138.1
104.1
78.5
70.0
53.0
44.5
Security at N=4
Number of Registers (N)
Time-Complexity After SCA Attack (in Bits)
0
50
100
150
200
250
800
5
10
15
20
25
30
35

Figure 8

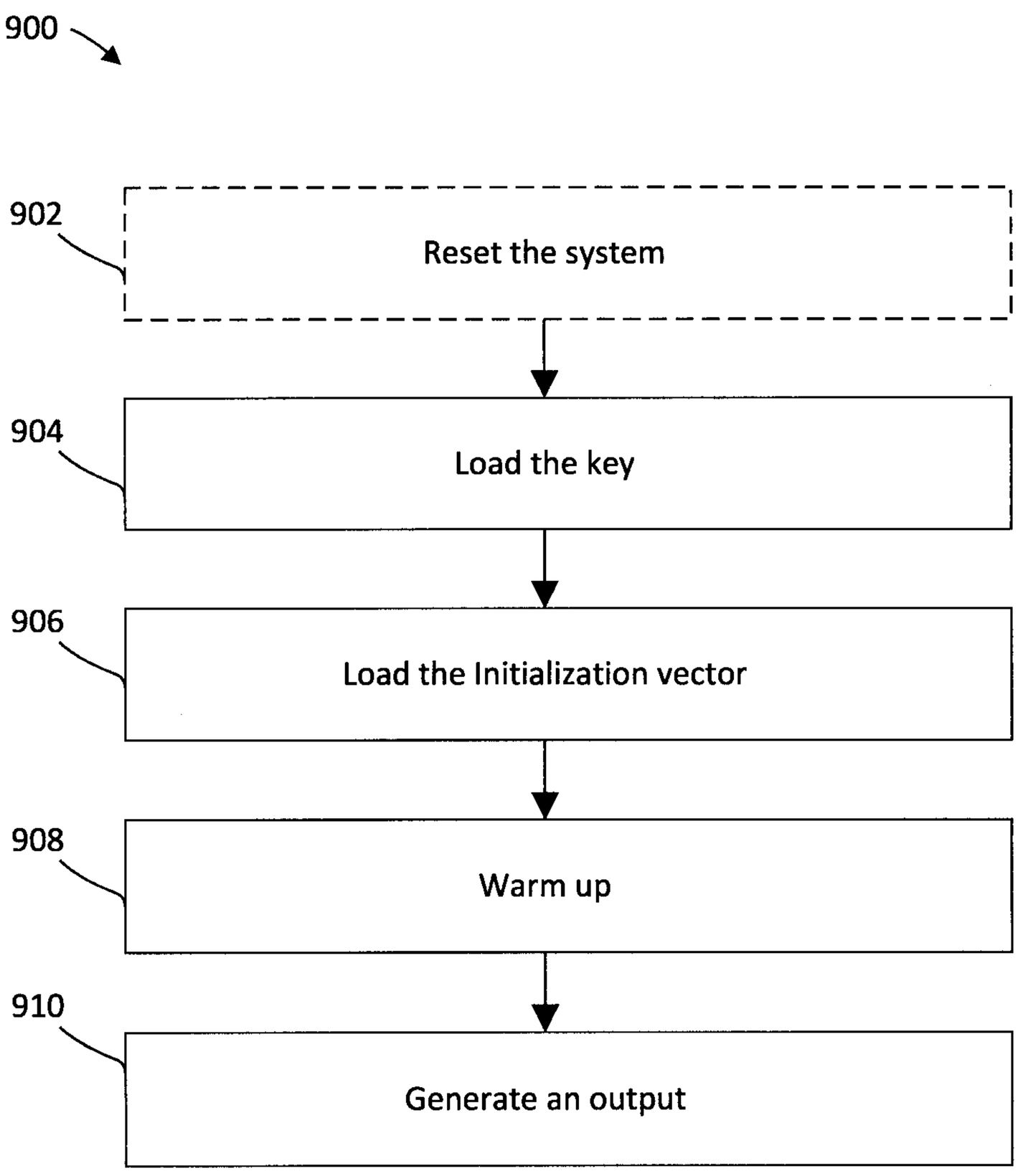

900
902
Reset the system
904
Load the key
906
Load the Initialization vector
908
Warm up
910
Generate an output

Figure 9

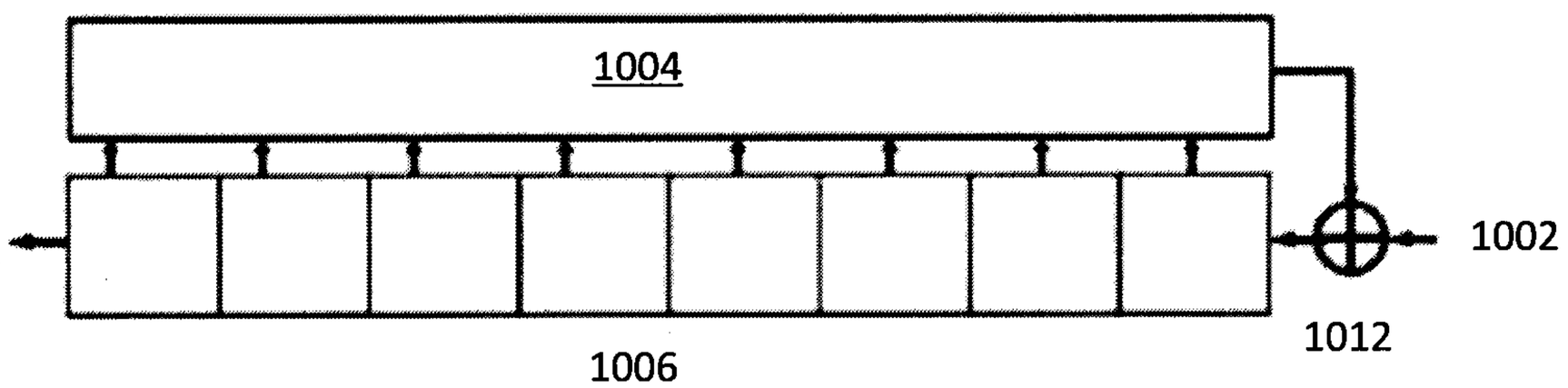


Figure 10

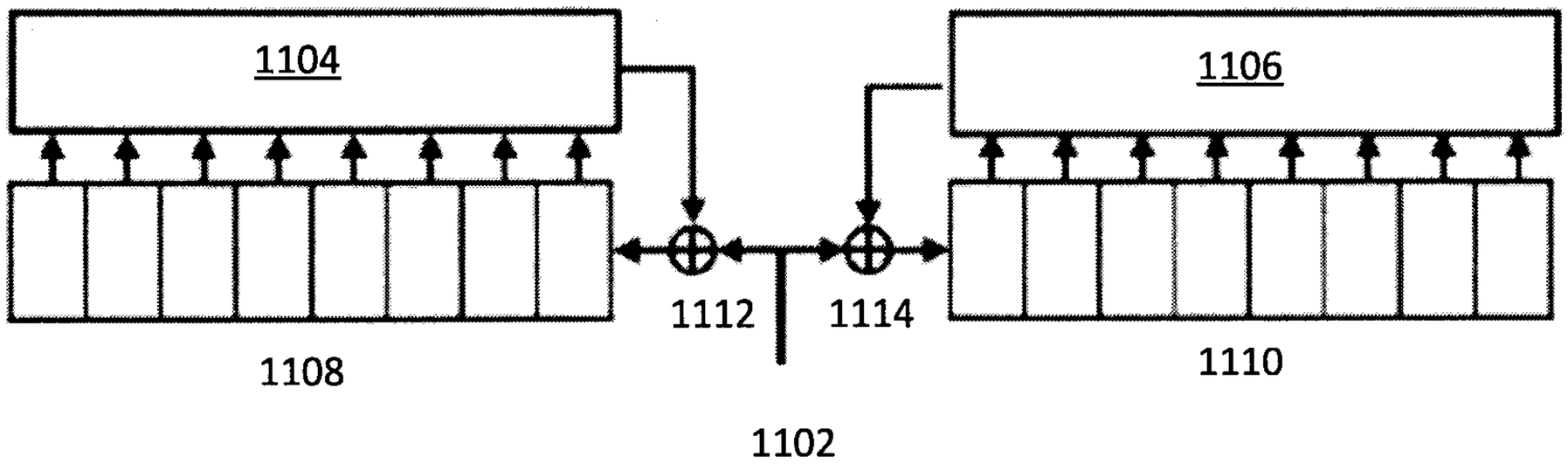
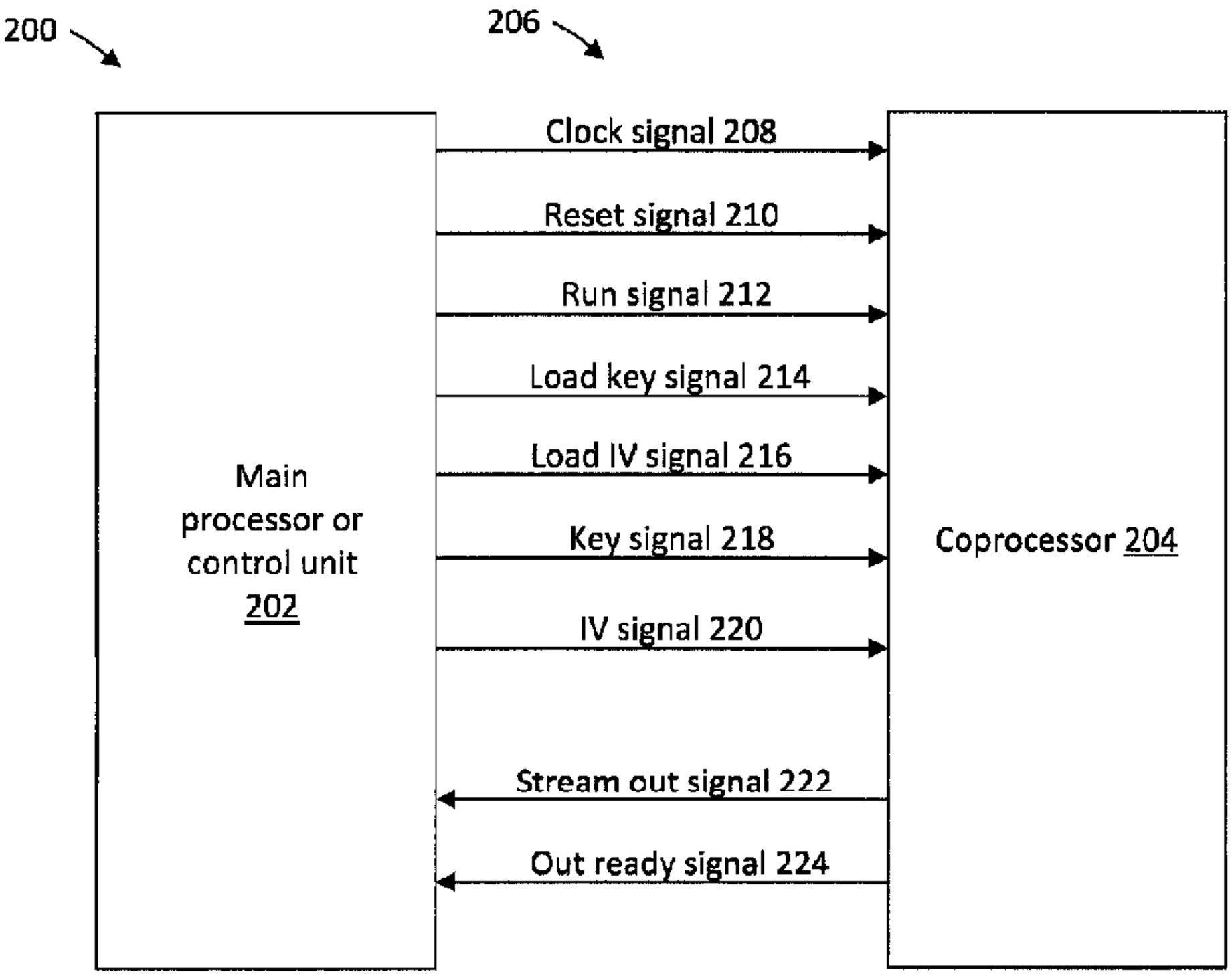


Figure 11


200
206
Clock signal 208
Reset signal 210
Run signal 212
Load key signal 214
Load IV signal 216
Key signal 218
IV signal 220
Stream out signal 222
Out ready signal 224
Main processor or control unit 202
Coprocessor 204