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

(51) International Patent Classification:
G11C 29/52 (2006.01)   G11C 17/14 (2006.01)
G06F 11/00 (2006.01)   G11C 7/12 (2006.01)

(21) International Application Number:
PCT/CA2007/002284

(22) International Filing Date:
20 December 2007 (20.12.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/871,519   22 December 2006 (22.12.2006)   US
60/871,571   22 December 2006 (22.12.2006)   US

(71) Applicant (for all designated States except US): SIDENSE CORP. [CA/CA]; 84 Hines Road, Suite 260, Ottawa, Ontario K2K 3G3 (CA).

(72) Inventor; and
(75) Inventor/Applicant (for US only): KURJANOWICZ, Wlodek [CA/CA]; 56 Sherring Crescent, Ottawa, Ontario K2K 2T1 (CA).

(74) Agents: HUNG, Shin et al.; BORDEN LADNER GERVAIS LLP, World Exchange Plaza, 100 Queen Street, Suite 1100, Ottawa, Ontario K1P 1J9 (CA).

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

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL,

[Continued on next page]

(54) Title: A PROGRAM VERIFY METHOD FOR OTP MEMORIES
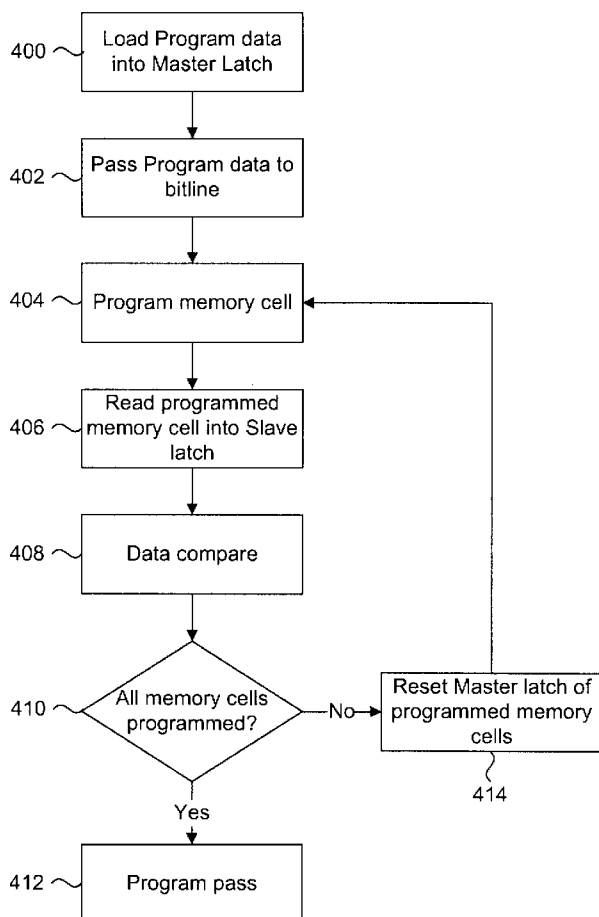
(57) Abstract: A method for executing a program verify operation in a non-volatile memory. A data register having master and slave latching circuits is used for concurrently storing two different words of data. In a program operation, the master latch stores program data which is used for programming selected memory cells. In a program verify operation, the data programmed to the memory cells are read out and stored in the slave latches. In each data register stage, the logic states of both latches are compared to each other, and a status signal corresponding to a program pass condition is generated if opposite logic states are stored in both latches. The master latch in each stage is inverted if programming was successful, in order to prevent re-programming of that bit of data.

PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report*

A PROGRAM VERIFY METHOD FOR OTP MEMORIES

CROSS REFERENCE TO RELATED APPLICATIONS

[001]   This application claims the benefit of priority of U.S. Provisional Patent Application No. 60/871,519 filed December 22, 2006, and U.S. Provisional Patent Application

5   No. 60/871,571 filed December 22, 2006, both of which are incorporated herein by reference in their entirety.

FIELD OF THE INVENTION

[002]   The invention is directed to non-volatile memories. More specifically, the invention is directed to data registers of non-volatile memories.

10   **BACKGROUND OF THE INVENTION**

[003]   Over the past 30 years, anti-fuse technology has attracted significant attention of many inventors, IC designers and manufacturers. An anti-fuse is a structure alterable to a conductive state, or in other words, an electronic device that changes state from not conducting to conducting. Equivalently, the binary states can be either one of high resistance

15   and low resistance in response to electric stress, such as a programming voltage or current. There have been many attempts to develop and apply anti-fuses in microelectronic industry, but the most successful anti-fuse applications to date can be seen in FGPA devices manufactured by Actel and Quicklogic, and redundancy or option programming used in DRAM devices by Micron. Anti-fuse technology is well known in the art, and example anti-

20   fuse transistors are shown in Figures 1 to 5b.

[004]   Anti-fuse memory is one type of one-time programmable (OTP) memory in which the device can be permanently programmed (electrically) with data once. This data is programmed by an end user for a particular application. There are several types of OTP memory cells which can be used. OTP memories provide users with a level flexibility since

25   any data can be programmed.

[005]   Anti-fuse memory can be utilized in all one time programmable applications, including RF-ID tags. RF-ID tagging applications are gaining more acceptance in the industry, particularly in sales, security, transport, logistics, and military applications for

example. The simplicity and full CMOS compatibility anti-fuse memory allows for application of the RF-ID tag concept to integrated circuit manufacturing and testing processes. Therefore, IC manufacturing productivity can be increased by utilizing anti-fuse memory in combination with an RF communication interface on every wafer and/or every die on the

5   wafer allowing for contact-less programming and reading chip specific or wafer specific information during IC manufacturing and packaging, as well as during printed circuit board assembly.

[006]   Figure 1 is a circuit diagram illustrating the basic concept of an anti-fuse memory cell, while Figures 2 and 3 show the planar and cross-sectional views respectively,

10   of the anti-fuse memory cell shown in Figure 1. The memory cell of Figure 1 includes a pass, or access transistor 10 for coupling a bitline BL to a bottom plate of anti-fuse device 12. A wordline WL is coupled to the gate of access transistor 10 to turn it on, and a cell plate voltage Vcp is coupled to the top plate of anti-fuse device 12 for programming anti-fuse device 12.

15   [007]   It can be seen from Figures 2 and 3 that the layout of access transistor 10 and anti-fuse device 12 is very straight-forward and simple. The gate 14 of access transistor 10 and the top plate 16 of anti-fuse device 12 are constructed with the same layer of polysilicon, which extend across active area 18. In the active area 18 underneath each polysilicon layer, is formed a thin gate oxide 20, also known as a gate dielectric, for electrically isolating the

20   polysilicon from the active area underneath. On either side of gate 14 are diffusion regions 22 and 24, where diffusion region 24 is coupled to a bitline. Although not shown, those of skill in the art will understand that standard CMOS processing, such as sidewall spacer formation, lightly doped diffusions (LDD) and diffusion and gate silicidation, can be applied. While the classical single transistor and capacitor cell configuration is widely used, a transistor-only

25   anti-fuse cell is further desirable due to the semiconductor array area savings that can be obtained for high-density applications. Such transistor-only anti-fuses must be reliable while simple to manufacture with a low cost CMOS process.

[008]   Figure 4a shows a cross-sectional view of an anti-fuse transistor that can be manufactured with any standard CMOS process. Variants of this anti-fuse transistor are

30   disclosed in commonly owned U.S. Patent Application No. 11/762,552, filed on June 13, 2007, the contents of which are incorporated by reference. In the presently shown example, the anti-fuse transistor is almost identical to a simple thick gate oxide, or input/output MOS

transistor with one floating diffusion terminal. The disclosed anti-fuse transistor, also termed a split-channel capacitor or a half-transistor, can be reliably programmed such that the fuse link between the polysilicon gate and the substrate can be predictably localized to a particular region of the device. The cross-section view of Figure 4a is taken along the

5   channel length of the device, which in the presently described example is a p-channel device.

[009]   Anti-fuse transistor **26** includes a variable thickness gate oxide **28** formed on the substrate channel region **30**, a polysilicon gate **32**, sidewall spacers **34**, a field oxide region **36**, a diffusion region **38** and an LDD region **40** in the diffusion region **38**. A bitline

10   contact **42** is shown to be in electrical contact with diffusion region **38**. The variable thickness gate oxide **28** consists of a thick oxide and a thin gate oxide such that a portion of the channel length is covered by the thick gate oxide and the remaining portion of the channel length is covered by the thin gate oxide. Generally, the thin gate oxide is a region where oxide breakdown can occur. The thick gate oxide edge meeting diffusion region **38** on the

15   other hand, defines an access edge where gate oxide breakdown is prevented and current between the gate **32** and diffusion region **38** is to flow for a programmed anti-fuse transistor. While the distance that the thick oxide portion extends into the channel region depends on the mask grade, the thick oxide portion is preferably formed to be at least as long as the minimum length of a high voltage transistor formed on the same chip.

20   [0010]   In this example, the diffusion region **38** is connected to a bitline through a bitline contact **42**, or other line for sensing a current from the polysilicon gate **32**, and can be doped to accommodate programming voltages or currents. This diffusion region **38** is formed proximate to the thick oxide portion of the variable thickness gate oxide **28**. To further protect the edge of anti-fuse transistor **26** from high voltage damage, or current leakage, a resistor

25   protection oxide (RPO), also known as a salicide protect oxide, can be introduced during the fabrication process to further space metal particles from the edge of sidewall spacer **34**. This RPO is preferably used during the salicidiation process for preventing only a portion of diffusion region **38** and a portion of polysilicon gate **32** from being salicided. It is well known that salicided transistors are known to have higher leakage and therefore lower breakdown

30   voltage. Thus having a non-salicided diffusion region **38** will reduce leakage. Diffusion region **38** can be doped for low voltage transistors or high voltage transistors or a combination of the two resulting in same or different diffusion profiles.

[0011]   A simplified plan view of the anti-fuse transistor 26 is shown in Figure 4b. Bitline contact 42 can be used as a visual reference point to orient the plan view with the corresponding cross-sectional view of Figure 4a. The active area 44 is the region of the device where the channel region 30 and diffusion region 38 are formed, which is defined by an OD mask during the fabrication process. The dashed outline 46 defines the areas in which the thick gate oxide is to be formed via an OD2 mask during the fabrication process. More specifically, the area enclosed by the dashed outline 46 designates the regions where thick oxide is to be formed. OD simply refers to an oxide definition mask that is used during the CMOS process for defining the regions on the substrate where the oxide is to be formed, and OD2 refers to a second oxide definition mask different than the first. Details of the CMOS process steps for fabricating anti-fuse transistor 26 will be discussed later. According to an embodiment of the present invention, the thin gate oxide area bounded by edges of the active area 44 and the rightmost edge of the OD2 mask, is minimized. In the presently shown embodiment, this area can be minimized by shifting the rightmost OD2 mask edge towards the parallel edge of active area 44. Previously mentioned U.S. Patent Application No. 11/762,552 describes alternate single transistor anti-fuse memory cells which can be used in a non-volatile memory array. Two transistor anti-fuse memory cells are known in the art, as shown in the example of Figures 5a and 5b.

[0012]   Figure 5b shows a planar view of a two-transistor anti-fuse memory cell 48 having a minimized thin gate oxide area that can be manufactured with any standard CMOS process, according to an embodiment of the present invention. Figure 5a shows a cross-sectional view of the memory cell 48 of Figure 5b, taken along line B-B. Two-transistor anti-fuse memory cell 48 consists of an access transistor in series with an anti-fuse transistor. The access transistor includes a polysilicon gate 50 overlying a thick gate oxide 52, which itself is formed over the channel 54. On the left side of the channel 54 is a diffusion region 56 electrically connected to a bitline contact 58. On the right side of the channel 54 is a common diffusion region 60 shared with the anti-fuse transistor. The anti-fuse transistor includes a polysilicon gate 62 overlying a thin gate oxide 64, which itself is formed over the channel 66. Dashed outline 68 represents the OD2 mask which defines the area where thick oxide is to be formed. The thick gate oxide 52 can correspond to that used for high voltage transistors while the thin gate oxide 64 can correspond to that used for low voltage transistors. It is well known that polysilicon gates 50 and 62 can be independently controlled, or alternatively can be connected to each other as shown in Figure 5b. In the example of Figure 5b, both

polysilicon gates **50** and **62** are part of the same polysilicon structure, and connected to a wordline through wordline contact **70**. Both diffusion regions **56** and **60** can have LDD regions, which can be identically doped or differently doped, depending on the desired operating voltages to be used. Commonly owned U.S. Patent Application No. 11/762,552 filed on June 13, 2007 describes alternate two-transistor anti-fuse memory cells which can be used in a non-volatile memory array.

[0013]   The programming speed of OTP memories is relatively slow, since each programming cycle will attempt to program a certain number of data words at the same time. Following each programming cycle is a program verify cycle to ensure that the data words were successfully programmed. Any bits that do not pass the program verify step are reprogrammed. This process continues until all the memory cell states have been successfully programmed. Figure 6a shows how an unprogrammed anti-fuse memory cell, such as anti-fuse transistor **26**, is programmed. Anti-fuse transistor **26** has its gate terminal connected to a wordline WL and its single diffusion region connected to a bitline BL. Programming is effected by biasing the bitline to VSS and driving the wordline to a high voltage level VPP. VPP is selected based on the process technology and thickness of the thin gate oxide that is sufficient for forming a conductive link between the polysilicon gate and the channel region.

[0014]   A successfully programmed anti-fuse transistor **26** is shown in Figure 6b, where a conductive link **72** is formed between the polysilicon gate and the channel region. Conductive link **72** is schematically represented as a resistive connection between the wordline and the channel region under the thin gate oxide region of anti-fuse transistor **26**. Therefore a programmed anti-fuse transistor having a conductive link stores one logic state of one bit of data. Accordingly, an unprogrammed anti-fuse transistor will by default store the other logic state of one bit of data. To prevent programming of the anti-fuse transistor **26**, the bitline is biased to VDD while the wordline is driven to VPP. This will be sufficient for inhibiting the conductive link from forming.

[0015]   Reading the anti-fuse transistor is achieved by driving the wordline to a read voltage VREAD, and by precharging the bitline to VSS, as shown in Figure 7. If the anti-fuse transistor **26** has a conductive link **72**, then the wordline will pull the bitline towards the VREAD voltage level via the conductive link **72** and the positive gate voltage of the anti-fuse transistor. This bitline voltage can be sensed and amplified by sense amplifier circuits. On

the other hand, if the anti-fuse transistor **26** is not programmed, ie. does not have a conductive link **72**, then the bitline will remain at approximately VSS.

[0016]   Most electrically programmable non-volatile memories, such as Flash, EEPROM or anti-fuse memories, require the use of high voltages to program the memory

5   cells. These high voltages are generated on chip and distributed to wordlines and/or bitlines during programming operations. Depending on the architecture of the memory, any number of words can be programmed simultaneously, where each word is made up of a predetermined number of bits. Assuming that the memory cells start in the unprogrammed state, logic 0 for example, only memory cells for storing a logic 1 state actually undergo

10   programming. The number of logic 1 states to be programmed at the same time depends on the data to be programmed. Because a large number of bits may be programmed simultaneously, the finite on-chip high voltage generators may not have sufficient current to program all the selected memory cells. Hence, after a first program cycle, the cells subjected to programming must be verified to ensure that they were successfully programmed.

15   [0017]   Verification is done by reading out the memory cells and comparing their stored logic states (logic 0 or 1) to the desired programmed state (logic 1 for example). If specific bits were not successfully programmed, as determined by the program verify cycle, then the programming cycle is repeated. However, there is no need to re-program cells which were successfully programmed. This is especially problematic for anti-fuse transistors.

20   A programmed anti-fuse transistor has a conduction path formed between its wordline and its bitline, thus repeating a program cycle on such a cell will draw more current away from the cells where programming is still required. Hence, those memory cells must be excluded from the next programming cycle. This iterative program-verify-program sequence continues until every memory cell to be programmed has been deemed to be successfully programmed.

25   [0018]   Figure 8 is a block diagram of a simplified anti-fuse memory device of the prior art, illustrating the logic circuits required for implementing program verify operations. The anti-fuse memory device of Figure 8 includes a memory array **80** consisting of anti-fuse memory cells, or other non-volatile memory cells, connected to wordlines and bitlines. Wordline drivers **82** apply the read and programming voltage levels to the wordlines, while

30   column decoders and sense amplifiers **84** are coupled to the bitlines for sensing and multiplexing the sensed data to a data register **86**. The data register **86** is primarily responsible for storing data to be programmed and for storing read data. The program-verify

circuits include additional latch circuits, which can be a second data register **88**, and comparison logic **90**. In use program data P_DATA is received by data register **86**, which is also stored in the second data register **88**. Those skilled in the art will understand that other circuits required for proper operation of the memory device are intentionally omitted, in order to simplify the schematic.

[0019]   Once a programming cycle has been completed, the data is read out from the memory cells being programmed, and stored in data register **86**. The comparison logic **90** then compares each bit position of data registers **86** and **88** to each other. If all the bit positions match, meaning that each bit position was successfully programmed, then the status signal STATUS will have a logic level indicating that all bits were programmed. Otherwise, if just one bit position did not successfully program, then the status signal STATUS will have another logic level, indicating that at least one bit was not properly programmed. Then further program cycles are executed, while the successfully programmed bits are masked, or inhibited from further programming.

[0020]   One of the main problems with this program verify scheme is that the circuit implementation consumes too much valuable circuit area. Mainly, a second data register is required for storing the program data, which will be very large if the word to be programmed is very wide. The logic required to disable successfully programmed bit positions can be complex, as can be the comparison logic used to detect the failed programming of at least one bit position. Those skilled in the art will understand that increased circuit area will directly impact the manufacturing cost of the device as more chip area is required per device.

[0021]   It is, therefore, desirable to provide a program verify scheme that does not require a second data register, and can simplify program verification operations.

SUMMARY OF THE INVENTION

[0022]   It is an object of the present invention to obviate or mitigate at least one disadvantage of prior OTP memories. More specifically, it is an object of the present invention provide a data register that can integrate program verify operations with data storage functionality.

[0023]   In a first aspect, the present invention provides a method for executing a program verify operation. The method includes loading program data, programming a

memory cell, reading the programmed data, comparing logic states stored in the first latch and the second latch, and providing a local status signal. The program data is loaded into a first latch of a register stage coupled to at least one bitline of a memory array. The memory cell being programmed is coupled to the at least one bitline. The programmed data is read

5   from the memory cell coupled to the at least one bitline into a second latch of the register stage. The local status signal corresponds to a program pass condition if opposite data states are stored in the first latch and the second latch. According to one embodiment, the step of loading includes shifting the program data serially into the first latch, the first latch and the second latch being arranged in a master-slave flip-flop configuration, where the first latch

10   is coupled to one of the bitlines prior to the step of programming. According to another embodiment, the step of loading includes providing the program data in parallel into the first latch, the first latch and the second latch being arranged in a master-slave flip-flop configuration. The step of reading can include sensing the programmed data with a sense amplifier and providing sensed data to the second latch.

15   [0024]   In a further embodiment of the present aspect, the step of comparing includes providing a single bit output corresponding to the result of comparing, and the step of comparing further includes combining the single output with a previous local status signal to provide the local status signal. The previous local status signal corresponds to a comparison result from a previous register stage. Furthermore, the program data stored in the first latch

20   is inverted if the second latch stores a data state corresponding to a programmed memory cell. Then the step of comparing further includes comparing the data states of the first latch and another first latch in the previous register stage. This further comparison includes providing another local status signal if the first latch and the another first latch have the same predetermined data state, the predetermined data state corresponding to the inverted

25   program data.

[0025]   In a second aspect, the present invention provides a method for operating a register stage having a first latch and a second latch arranged in a master-slave flip-flop configuration for storing a first bit of data and a second bit of data respectively. The method includes decoupling the first latch from a serial input terminal and decoupling the second

30   latch from the first latch at substantially the same time while a source clock signal oscillates, and initiating a shifting operation on either a high logic level of the source clock or a low logic level of the source clock for shifting one of the first bit of data and the second bit of data. In

an embodiment of the present aspect, the step of decoupling includes turning off a first gating device between the first latch and a serial input terminal, and turning off a second gating device between the first latch and the second latch. The step of turning off can include driving a first clock signal to an inactive logic level for turning off the first gating device, and

5   driving a second clock signal to the inactive logic level for turning off the second gating device. The first clock signal and the second clock signal are driven to the inactive logic level by maintaining a shift enable signal at the inactive logic level while the source clock signal oscillates.

[0026]   In another embodiment of the present aspect, the step of initiating includes

10   turning on the first gating device while the second gating device is turned off for shifting the second bit of data. The step of turning on the first gating device includes driving a first clock signal to an active logic level, where the first clock signal is driven to the active logic level when a shift enable signal is enabled while the source clock signal is at a first logic level. The second clock signal is driven to an inactive logic level when the shift enable signal is enabled

15   while an inverted source clock signal is at a second logic level. The step of initiating can include turning on the second gating device while the first gating device is turned off for shifting the first bit of data, where turning on the second gating device includes driving the second clock signal to the active logic level. The second clock signal is driven to the active logic level when the shift enable signal is enabled while the source clock signal is at the

20   second logic level. The first clock signal is driven to the inactive logic level when the shift enable signal is enabled while an inverted source clock signal is at the first logic level.

[0027]   Other aspects and features of the present invention will become apparent to those ordinarily skilled in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

25   **BRIEF DESCRIPTION OF THE DRAWINGS**

[0028]   Embodiments of the present invention will now be described, by way of example only, with reference to the attached Figures, wherein:

Fig. 1 is a circuit diagram of an anti-fuse memory cell;

Fig. 2 is a planar layout of the anti-fuse memory cell of Figure 1;

30   Fig. 3 is a cross-sectional view of the anti-fuse memory cell of Figure 2 along line x-x;

Fig. 4a is a cross-sectional view of a variable thickness gate oxide anti-fuse transistor;

Fig. 4b is a planar layout of the variable thickness gate oxide anti-fuse transistor of Figure 4a;

5   Fig. 5a is a cross-sectional view of a two-transistor anti-fuse memory cell;

Fig. 5b is a planar layout of the two-transistor anti-fuse memory cell of Figure 5a;

Fig. 6a is a schematic of the anti-fuse transistor of Figure 4a under programming conditions;

10   Fig. 6b is a schematic of a programmed anti-fuse transistor of Figure 4a;

Fig. 7 is a schematic of a programmed anti-fuse transistor of Figure 4a under read conditions;

Fig. 8 is a block diagram of a prior art non-volatile memory device having program verify circuitry;

15   Fig. 9 is a block diagram of a non-volatile memory device having a dual function data register with integrated program verify circuitry, according to an embodiment of the present invention;

Fig. 10 is a block diagram showing details of the dual function data register of Figure 9;

20   Fig. 11 is a circuit schematic of a folded bitline sensing scheme for an anti-fuse memory array;

Fig. 12 is a circuit schematic of one register stage of the dual function data register shown in Figure 10, according to an embodiment of the present invention;

Fig. 13 is a circuit schematic of the clock controller shown in Figure 9,
25   according to an embodiment of the present invention;

Fig. 14 is a circuit schematic of the program verify logic shown in Figure 10, according to an embodiment of the present invention;

Fig. 15 is a flow chart of a method for executing a program verify operation using the dual function data register, according to an embodiment of the present
30   invention;

Fig. 16 is an illustration of an example serial shift operation of the dual function data register of Figure 12 with a corresponding sequence diagram of the clock signals;

Fig. 17 is an illustration of another example serial shift operation of the dual function data register of Figure 12 with a corresponding sequence diagram of the clock signals;

Fig. 18 is a block diagram of a power up detection system, according to an embodiment of the present invention;

Fig. 19 is an illustration of the data states stored in memory cells of the test memory array of Figure 18;

Fig. 20a is a plan view of mask programmed skewed ROM cells, according to an embodiment of the present invention;

Fig. 20b is a plan view of mask programmed two transistor skewed ROM cells, according to an embodiment of the present invention;

Fig. 21a is a plan view of the test memory array of Figure 18 using the mask programmed skewed ROM cells of Figure 20a;

Fig. 21b is a plan view of the test memory array of Figure 18 using the mask programmed two transistor skewed ROM cells of Figure 20b;

Fig. 21c is an illustration of the data states stored in the skewed ROM cells of Figure 21a;

Fig. 22 is a circuit schematic of a register stage of the dual function data register of Figure 18, according to an embodiment of the present invention;

Fig. 23 is a flow chart of a power up detection method using the power up detection system of Figure 18;

Fig. 24 is an illustration of a serial shift operation of the dual function data register of Figure 18 with a corresponding sequence diagram of the clock signals; and,

Fig. 25 is a circuit schematic of one register stage of the dual function data register, according to an alternate embodiment of the present invention.

DETAILED DESCRIPTION

[0029]   Generally, the present invention is directed to a dual function serial and parallel data register with integrated program verify functionality. The dual function serial and parallel operation gives the user the option for serial or parallel data interfacing in a single circuit, while integrating program verify functionality for reducing circuit logic overhead by

utilizing the master and slave latching circuits of the data register to concurrently store two different words of data. In a program verify operation, the master latch stores program data and the slave latch will receive and store read data. Comparison logic in each register stage will compare the data of both latches and integrate the comparison result to that of the

5   previous register stage. The final single bit result will indicate the presence of at least one bit that has not been programmed. If reprogramming is necessary, automatic program inhibit logic in each stage will prevent successfully programmed bits from being re-programmed in each subsequent reprogram cycle. Because two data words can be concurrently stored in the dual function data register embodiments of the present invention, either data word can be

10   serially clocked out by selectively starting the shift operations on either the low or high active logic level of a clock signal. Significant circuit area is conserved relative to the prior art program verify schemes because a second data register is not required.

[0030]   Figure 9 is a block diagram of a non-volatile memory device having a dual function serial and parallel data register according to an embodiment of the present

15   invention. The non-volatile memory device **100** includes a memory array **102** consisting of non-volatile memory cells connected to wordlines and bitlines, such as the anti-fuse memory cells shown in Figures 4a, 4b, 5a and 5b. It is noted that memory array **102** can be implemented with other non-volatile memory cells, such as EEPROM or Flash memory cells. Wordline drivers **104** drive the wordlines to read or programming voltages, while column

20   decoder and sense amplifiers **106** are coupled to the bitlines for sensing and multiplexing the sensed data to the dual function serial and parallel data register **108**. A clock controller **110** provides the clock signals to control the shifting function of the dual function serial and parallel data register **108**.

[0031]   Dual function serial and parallel data register **108**, referred simply as the dual

25   function data register **108** from this point forward, is configured to facilitate execution of many useful features. Dual function data register **108** can receive and provide data in both a parallel format and a serial format. This is to accommodate systems that memory device **100** is integrated in or used with, the flexibility for a parallel or serial interface with the same design. For a parallel data interface, parallel data P_DATA can be asynchronously coupled

30   to and from dual function data register **108** via a bidirectional bus. Alternately, separate input and output busses can be used for carrying the parallel input data and the parallel output data, respectively. For a serial data interface, serial input data is received at input port

S_DATA_IN and serial output data is provided at output port S_DATA_OUT. Shifting of serial data is controlled by phase adjustable clock signals CK1 and CK2, which are controlled by source clock SCK and a shift enable signal SEN. As will be discussed later, one of two data words stored in dual function data register 108 can be synchronously shifted out by controlling the time that SEN is driven to the active logic level relative to SCK. Logic for automatically inhibiting programmed cells from being further programmed, and cascaded logic for generating the program verify status signal STATUS, is integrated into dual function data register 108 for program verify operations.

[0032]   Figure 10 is a block diagram showing further details of dual function data register 108, according to an embodiment of the present invention. The dual function data register can be used for temporary data storage for two different data words, to provide data to be programmed in the anti-fuse memory cell array, and to store data retrieved from the memory cell array. It is also used for comparing the data to be programmed with contents of the programmed cells for program verify purposes. The dual function data register 108 is an n-bit serial/parallel shift register, where n is an integer value greater than 0, and consists of individual register stages. Figure 10 shows first register stage 200, second register stage 202 and the last (nth) register stage 204. Each register stage has a parallel input P_IN controlled by a WRITE signal, a parallel output P_OUT, and receives clock signals CK1 and CK2. The WRITE signal allows for asynchronous parallel write to the register, while CK1 and CK2 allow shifting of the register contents in and out synchronously with a source clock SCK (not shown). Each register stage has a serial input terminal for receiving data from a previous register stage, and a serial output terminal for providing data to a subsequent register stage. For example, register stage 202 has a serial input terminal for receiving serial output data S1_OUT from register stage 200, and has a serial output terminal for providing serial output data S2_OUT to the next register state. The first register stage 200 receives serial input data through input port S_IN, and provides serial output data through output port S_OUT.

[0033]   With specific reference to register stage 202, each register stage will provide program data to a memory cell through terminal SA1_in during a program operation, while receiving read data from a memory cell through terminal SA_out during a read operation or a program verify operation. Those skilled in the art will understand that the program data can be provided either directly to the selected bitline or to the selected bitline through a sense amplifier, and that the read data is provided from a sense amplifier. Each register stage can

provide two program verify signals, noted as C1_a and C1_b for register stage **200** for example. C1_a indicates if the initial programming operation was successful for all bits, while C1_b indicates if a re-programming operation is successful for all bits. The reason for having two different comparison results will be discussed in more detail later. The two comparison results from each register stage are fed to a subsequent register stage, which then combines the result with its own comparison results. The combined result is then fed to the next stage. A final stage of logic, the program verify logic **206**, receives the last two comparison results from the last register stage **204**, and generates a single STATUS signal. The STATUS signal will be used to indicate if an initial program operation was successful or if a subsequent re-program operation was successful. Program verify logic **206** can be implemented outside of dual function data register **108**.

[0034]   Figure 11 is a schematic of a portion of a folded bitline anti-fuse memory array **102** and its associated bitline sensing circuitry **106** of Figure 9. In order to simplify the schematic, only one folded bitline pair BL/BL* and two wordlines are shown. Column decoder circuitry is not shown for selectively coupling multiple folded bitline pairs to the bitline sense amplifier circuitry. Folded bitline anti-fuse memory array **210** includes wordlines WL0 and WL1 connected to the gate terminals of n-channel anti-fuse transistors **212** and **214**, n-channel isolation transistors **216** and **218** for coupling the upper portion of the bitlines to the lower portion of the bitlines in response to signal ISO, and bitline sensing circuitry. The bitline sensing circuitry includes a precharge circuit **220**, a reference charge circuit **222**, and a bitline sense amplifier **224**.

[0035]   The precharge circuit **220** includes two n-channel precharge transistors **226** and **228** connected in series between BL and BL* and having their gate terminals connected to precharge signal BLPCH. The shared source/drain terminal of precharge transistors **226** and **228** receives a precharge voltage VPCH. In operation, both precharge transistors **226** and **228** will turn on to precharge bitlines BL and BL* to VPCH in response to an active high logic level of BLPCH, in preparation for a read operation.

[0036]   The reference charge circuit **222** includes n-channel steering transistors **230** and **232** connected in series between BL and BL*, a capacitance circuit implemented as an n-channel transistor **234**, and a p-channel precharge transistor **236**. Steering transistor **230** has its gate terminal connected to even selection signal E_REF, while steering transistor **232** has its gate terminal connected to odd selection signal O_REF. Capacitance circuit **234**

has its gate terminal connected to voltage supply VCC, and is connected in series with

precharge transistor **236** between the shared source/drain terminal of steering transistors

230 and **232** and voltage supply VCC. Precharge transistor **236** has its gate terminal

connected to a precharge or enable signal PCH*. Generally, capacitance circuit **234** will be

5   precharged when a low logic level PCH* pulse is received. The duration of the PCH* pulse

can be predetermined based on the size of transistor **234** and the desired reference charge

to be provided. Once precharged, either steering transistor **230** or **232** is turned on to couple

the reference charge of capacitance circuit **234** to the corresponding bitline. By example, the

charge being added to a bitline can be approximately 50 millivolts. It is noted that signals

10   E_REF and O_REF can be controlled by the same even/odd addressing bit used for

selecting WL0 or WL1. In one embodiment, activation of WL0 will cause E_REF to be

activated, thereby coupling the reference charge to the complementary bitline.

[0037]   The bitline sense amplifier **224** consists of a standard cross-coupled inverter

circuit that is well known in the art. The circuit includes p-channel transistors both connected

15   in series to respective n-channel transistors. The common drain terminal of the p-channel

transistors receives a high logic level enable signal H_EN, while the common source terminal

of the n-channel transistors receives a low logic level enable signal L_EN. H_EN can be a

lowered internal VCC level, while L_EN can be a VSS level. The operation of bitline sense

amplifier **224** in the DRAM art is well known. When enable signals H_EN and L_EN are

20   activated, either at the same time or at different times, bitline sense amplifier **224** will sense a

small voltage differential between BL and BL*, and quickly drive both BL and BL* to the full

logic level states of H_EN and L_EN.

[0038]   Because bitline sense amplifier **224** is connected to both bitlines BL and BL*,

the logic state being programmed or read from the memory array will depend on the memory

25   cell that is accessed. For example, if both anti-fuse transistors **212** and **214** store a logic 1,

bitline sense amplifier **224** will latch two different logic states depending on which anti-fuse

transistor is accessed. Therefore, a data state corrector **238** is used for ensuring that the

voltage level corresponding to logic 1 and 0 states is read and programmed. In the present

example, if WL0 is activated to read anti-fuse transistor **214**, then signal EVEN will be at the

30   logic state for coupling BL* to gating transistor **242**. Alternately, of WL1 is activated to read

anti-fuse transistor **212**, then signal EVEN will be at the opposite logic state for coupling BL

to gating transistor **242**. The operation of data state corrector **238** is similar when program

data is to be coupled to either BL or BL* from gating transistor **240**. Data state corrector **238** can be implemented as a simple bi-directional multiplexor controlled by signal EVEN, which can be related to the address used to select wordline WL0 and WL1. Signal EVEN can be related to signals E_REF and O_REF as well. Data to be programmed to the bitlines is

5   provided through n-channel gating transistor **240** which is coupled to SAi_in and controlled by program signal PGM. Data to be read from the bitlines is provided through n-channel gating transistor **242** which is coupled to SAi_out and controlled by read signal READ. Accordingly, gating transistor **240** is turned on during a program operation while gating transistor **242** is turned on during a read operation. Variable "i" is an integer value between 1

10   and max number n, which represents the specific register stage of the dual function data register **108**. Terminals SAi_in and SAi_out are connected to one register stage of the dual function data register **108**.

[0039]   Figure 12 is a circuit schematic of one register stage of the dual function data register **108** shown in Figure 10, according to an embodiment of the present invention. Data

15   register stage **300** includes two main sections. The first section is the data storage circuit **302**, which is responsible for data input, output and latching operations. The second section includes status circuit **304** and auto-program inhibit circuit **306**. All the register stages of the dual function data register **108** shown in Figure 10 will have the same circuit configuration shown in Figure 12. The data storage circuit **302** will now be described. It is noted that

20   variable "i" in the signal names denotes the register stage it is associated with, where "i" is any integer value greater than 0 and less than n, where n is the maximum number of register stages in the dual function data register **108**.

[0040]   Data storage circuit **302** includes a master latch **310** and a slave latch **312** connected as a master-slave flip-flop, serial input gating device **314**, transfer gating device

25   **318**, and a parallel input gating device **320**. Latches **310** and **312** can be implemented as simple cross-coupled inverter circuits with a non-inverting output relative to its input, but slave latch **312** will be configured to be overwritten by master latch **310** in a program operation. Those skilled in the art will understand that transistor sizing can be configured to achieve this desired effect. Gating devices **314** and **318** are shown as n-channel transistors,

30   but can be transmission gates or p-channel devices. Gating device **314** has its gate terminal connected to clock signal CK1, while gating device **318** has its gate terminal connected to clock signal CK2. CK1 and CK2 are controlled clock signals. Gating devices **314** and **318** are

controlled by CK1 and CK2 to shift data serially from the serial input terminal Si_in to the master latch **310**, or from master latch **310** to slave latch **312**, or from slave latch **312** to serial output terminal Si_out. Parallel input data P_IN_i is provided to the input of master latch **310** via gating device **320** when signal WRITE is at the active logic level, which in the present example is the high logic level. Parallel output data P_OUT_i is provided from the output of slave latch **312**. The output of master latch **310**, typically being program data, is coupled to a sense amplifier via terminal SAi_in, while read data from the sense amplifier is provided from terminal SAi_out and stored by slave latch **312**.

[0041]   As previously mentioned, the program verify scheme utilizes two related circuits; the status circuit **304** and the auto-program inhibit circuit **306**. The status circuit **304** includes an XOR logic gate **330**, AND logic gates **332** and **334**, and an inverter **336**. Inverter **336**, XOR gate **330** and AND gate **332** form a data match circuit, while AND gate **334** forms a reprogram verify circuit. XOR gate **330** has a first input connected to the output of slave latch **312** and a second input coupled to an output of master latch **310** via inverter **336**. The output of XOR gate **330** is connected to a first inverted input of AND gate **332**, which has its second input connected to input Ci-1_a. The output of AND gate **332** is a local status signal Ci_a. Input Ci-1_a is the local status signal of the previous register stage, and more specifically, the output of an AND gate **332** of the previous register stage. Local status signal Ci_a is connected to the second input of an AND gate **332** of the next register stage. AND gate **334** has a first input connected to the output of master latch **310** and a second input connected to input Ci-1_b, which is another local status signal from the previous register stage. The output of AND gate **334** is second local status signal Ci_b. Input Ci-1_b is connected to the output of an AND gate **334** of the previous register stage, while Ci_b is connected to the second input of an AND gate **334** of the next register stage. AND gate **334** simply combines the logic state of master latch **310** with the logic state of the master latch **310** of the previous register stage.

[0042]   XOR gate **330** compares the logic state of slave latch **312** to the inverted state of master latch **310**. The comparison result of XOR gate **330** is then compared and combined with the comparison result from the previous register stage using AND gate **332**. This portion of the circuit is used to verify if the initial programming operation was successful or not speed up memory testing and verification process. It compares all the bits in a row simultaneously and the result is combined into a single output signal STATUS in Figure 14. In a compare

operation, STATUS goes high upon COMP activation and remains HIGH if all data bits match (logic 1 propagates through all data register stages) or goes LOW if a mismatch is detected (when any of the outputs of the XOR gates exit logic 1 instead of 0, indicating mismatch between the inverted data from master latch and data from slave latch). STATUS

5   remains active for as long as COMP is held HIGH and is reset to its default LOW state when COMP is LOW. Before the COMP operation, master latches of the data register should be reloaded with the original desired data, and the slave latches should contain the actual data read from the memory cells after programming.

[0043]   The auto-program inhibit circuit **306** is used to verify if a reprogramming

10   operation was successful or not, when the auto-program inhibit circuit **306** is engaged after an initial program operation failure. The auto-program inhibit circuit **306** includes a precharge device **340** and a coupling device **342** connected in series between a voltage supply such as VDD and the input of master latch **310**. Both devices **340** and **342** are shown as being n-channel transistors in the present embodiment. The gate of precharge device **340** is

15   connected to precharge signal PCH and the gate of coupling device **342** is connected to the output of slave latch **312**. The selection of the voltage supply depends on the logic state stored by master latch **310** for selecting a memory cell to be programmed. For example, if master latch **310** stores a logic 0 (VSS) to indicate programming of the memory cell connected to that bitline, then the voltage supply connected to precharge device **340** will be

20   VDD. Hence, VDD is the logic state stored in a master latch **310** for a memory cell that is not to be programmed. The auto-program inhibit circuit will therefore change the state of the master latch **310** if the memory cell was successfully programmed. In the present example, a successfully programmed memory cell will result in slave latch **312** storing a high a logic state in a program verify read operation following a program operation. Therefore, when PCH

25   is driven to the high logic level, VDD is coupled to the input of master latch **310** to flip its state.

[0044]   The data shifting operation and the logic comparison operation of the data register stage **300** will be discussed later in further detail. Prior to such a discussion, a description of the clock controller **110** of Figure 9 follows. Figure 13 is a circuit schematic of

30   the clock controller **110**, according to an embodiment of the present invention. Clock controller **110** includes a first AND logic gate **350** and a second AND logic gate **352**. Both AND gates **350** and **352** have a first input for receiving shift enable signal SEN, while AND

gate **352** has a second input for receiving source clock signal SCK and AND gate **350** has a second input for receiving the complement of SCK. The output of AND gate **350** is clock signal CK2 and the output of AND gate **352** is clock signal CK1. Therefore while SEN is at the inactive low logic state, the output of AND **350** and CK1 will be at the inactive low logic

5   state. A notable feature is the ability to control the starting phase of CK1 and CK2 by selecting when to drive SEN to the active high logic level when SCK is either at the high or low logic level. Hence CK1 and CK2 are phase adjustable clock signals.

[0045]   Following the discussion of the program verify logic **206** will be an overview of the operation of the entire dual function data register. Figure 14 is a circuit schematic of the

10   program verify logic **206**, according to an embodiment of the present invention. Program verify logic **206** includes AND logic gates **360**, **362** and OR logic gate **366**. AND gate **360** receives Cn_a, which is the first program verify signal from the last register stage, and a compare enable signal COMP. The output of AND gate **360** is connected to a first input of OR gate **366**. AND gate **362** receives Cn_b at its inverted terminal, which is the second

15   program verify signal from the last register stage, and the precharge signal PCH at its non-inverted terminal. The output of AND gate **362** is connected to a second input of OR gate **366**.

[0046]   Generally, the operation of the program verify logic **206** is described. In the high speed program verify operation for example, after program operation signal STATUS

20   goes high (logic 1 for example) upon COMP activation to the high logic state, and remains high if all data bits match (for example, the logic 1 propagates through all data register stages) or goes low (logic 0 for example) if a mismatch is detected. A mismatch is detected when any of the outputs of the XOR gates exit logic 1 instead of 0, indicating a mismatch between the inverted data from master latch **310** and data from slave latch **312**. The

25   STATUS signal remains active for as long as COMP is held high and is reset to its default low state when COMP is low. Before the COMP operation, master latches of the data register should be loaded with the original desired data, and the slave latches should contain the actual data read from the memory cells after programming. During this high speed program verify operation, PCH is held inactive at the low logic level.

30   [0047]   The program verify and data shifting operations of the dual function page register **108** will now be described, as reference will be made to the circuits shown in Figures

10, 12, 13 and 14. Following is a description of the operation of dual function page register 108 for program and program verify operations, with reference to the flow chart of Figure 15.

[0048]   Starting at step 400, the program operation begins with loading of the dual function page register 108 with data to be programmed. Program data to be stored in the

5   anti-fuse memory cells are either loaded in parallel through the P_IN_i ports or serially by shifting in the bits from the S_IN port of the first register stage 200 of Figure 10. In the parallel loading operation, signal WRITE is raised to the active high logic level to load the master latch 310 with the data bit. In the serial loading operation, the bits are to be shifted into the master latch 310 of each register stage. Referring to Figure 13, this is done by

10   driving SEN to the active high logic level while SCK is at the high logic level. Therefore, the shifting operation starts with CK1 at the high logic level and CK2 at the inactive low logic level. Therefore with SEN held at the high logic level, SCK continues to oscillate to alternately drive CK1 and CK2 to the active high logic level until all the bits of the program data are shifted into the master latches 310 of all the register stages. In the present example

15   where the memory cells are the anti-fuse transistors shown in Figures 4a, 4b, 5a and 5b, any master latch 310 storing a logic 0 (VSS) will cause an anti-fuse transistor connected to the corresponding bitline to be programmed. Any master latch 310 storing a logic 1 (VDD) will inhibit an anti-fuse transistor connected to the corresponding bitline from being programmed.

[0049]   Following at step 402, column decoders are activated based on a particular

20   column address, and the bitlines are driving to voltage levels corresponding to the logic states stored in the master latches 310. With the bitlines biased based on the program data stored in the dual function data register 108, a selected wordline is driven to the VPP programming voltage at step 404 to begin programming. Only the anti-fuse memory cells connected to the selected wordline and a bitline biased to VSS should be programmed,

25   meaning that a conductive link between its polysilicon gate and its channel region should be formed. Now the program verify operation commences at step 406 by reading out the states of the anti-fuse memory cells that were just subjected to the programming operation. This is done by executing a normal read operation by driving the same wordline with a read voltage. The data is sensed and loaded into the slave latches 312 of each register stage. It is noted

30   that SEN is set to the inactive low logic level to keep all the gating devices 314 and 318 turned off. Now the master latches 310 are storing the program data for the selected memory cells and the slave latches 312 store the read data from the same selected memory cells.

- 20 -

[0050]   Using XOR gate **330**, the stored data in master latch **310** and slave latch **312** of each register stage are compared to each other. If the master latch **310** stored a logic 0 for programming the memory cell, then a read operation on a successfully programmed memory cell will yield a logic 1 stored in the slave latch **312**. Therefore, because of inverter **336**, XOR gate **330** will provide a low logic level output in such a case. In the opposite case where the master latch **310** stored a logic 1 for inhibiting programming of the memory cell, a read operation on a non-programmed memory cell will yield a logic 0 stored in the slave latch **312**. Therefore, XOR gate **330** will again provide a low logic level output. Both these cases indicate that programming was successful. On the other hand, if the memory cell was not properly programmed, or insufficiently programmed, then the read operation will yield a logic 0 stored in the slave latch **312**. XOR gate **330** will then provide a high logic level output, thereby indicating unsuccessful programming of the memory cell. It should be noted that the first register stage will have its Ci-1_a and Ci-1_b terminals tied to logic 1 (ie. VDD). These program verify results from each stage are fed serially to the next register stage until the last Ci-1_a program verify output is provided to AND gate **360** of program verify logic **206**.

[0051]   In program verify logic **206**, PCH is held at the low logic level and COMP is driven to the high logic level for the initial program operation to assess the result of all the comparisons in each register stage. This corresponds to a data compare step **408** in Figure 15. If Cn_a is at the high logic level when COMP is at the high logic level, status signal STATUS will be at the high logic level. This means that all the register stages have reported successful programming of their respective memory cells. Alternately, if at least one register stage reports failed programming, then Cn_a will be at the low logic level, and STATUS will be driven to the low logic level to signal failed programming to the system. At step **410**, if STATUS is at the high logic level to indicate successful programming of all the bits, then the program verify operation ends at step **412**. If STATUS is at the low logic level to indicate failed programming, then the method proceeds to step **414** to commence reprogramming after the initial program operation. At step **414**, the master latches **310** corresponding to successfully programmed cells are reset to a logic level that inhibits further programming for that particular memory cell. In Figure 12 for example, this is done by driving PCH to the high logic level, which will couple VDD (logic 1) to master latch **310** if the slave latch **312** stores a logic 1. Hence each register stage automatically inhibits further programming of its corresponding memory cell if it was successfully programmed. Unsuccessfully programmed

memory cells will have their corresponding slave latch **312** store a logic 0, thereby allowing its master latch **310** to retain the logic 0 data.

[0052]   The method then loops back to step **404** to repeat programming of only the unsuccessfully programmed memory cells, as selected by the register stages. The data is once again read out from the memory cells at step **406**. Now at step **408**, a different data compare operation is executed by the register stages in order to confirm successful programming of all the bits.

[0053]   At step **410** the output of all master latches **310** are compared to each other via AND gates **334**, and the final output Cn_b is received by program verify circuit **206** of Figure 14. The precharge signal PCH enables the auto-program inhibit circuit **306** of all the register stages to remove all the bits that are already programmed from the master latch by flipping their states to an opposite logic level, ie. a logic 1, and activates the STATUS signal indicating if any of the bits require further programming. The STATUS signal remains active as long as PCH is kept at the high logic level. PCH at the high logic level enables AND gate **362** of the program verify circuit **206** to pass the Cn_b signal. Cn_b at the high logic level, when all the master latches **310** store a logic 1 for example, indicates successful programming of all the bits. A high logic level Cn_b is inverted at the input of AND gate **362** to drive STATUS to the low logic level, indicating successful programming of all the memory cells. On the other hand, if even one master latch still retains a logic 0 state, then Cn_b will be at the low logic level, and STATUS will be at the high logic level to indicate the presence of at least one unsuccessfully programmed bit of data. If STATUS is high, then the method loops back to step **404** for a further reprogram iteration. It is noted that step **414** has essentially been executed in step **408**, and is therefore redundant for reprogramming iterations. The number of reprogram iterations can be monitored, and the entire process can be halted if it appears that certain memory cells are not programmable.

[0054]   The dual function register **108** of the present embodiment can be used for applications other than for program verify operations, as previously shown. The dual function register **108** can be used to temporarily store two different words of data for other applications. For example, in the parallel mode of operation where program data has been loaded into the register stages in parallel in preparation for a programming operation, the programming operation can be suspended while a read operation is executed. Hence data read from the memory array can be stored in the slave latches and read out in parallel while

- 22 -

program data is stored in the master latches of the register stages. Once the read operation has been completed, the program operation can resume. Once again in the parallel mode of operation, a first data word can be loaded into the master latches 310 through parallel input ports P_IN_i, and then shifted to the slave latches 312 by pulsing SEN to the high logic level

5   while SCK is at the low logic level. Then a second data word can be loaded into the master latches 310. The first data word can be immediately read out through the P_OUT_i port, then the second data word is shifted to the slave latches 312 and read out through the P_OUT_i port. If only the second data word is desired, then the first data word is overwritten as the bits of the second data word are shifted into the slave latches 312. In the previously described

10   operations, there is only data shifting within each register stage.

[0055]   Serial output of one of the two data words stored in the dual function data register 108 is possible, according to an embodiment of the present invention. For example, one data word provided by the memory system can be serially shifted into the master latches 310 or provided in parallel into the master latches 310, while a second data word can be read

15   from the memory array and stored in the slave latches 312. Figure 16 illustrates an example serial shift operation of dual function data register 108, where the data stored in the slave latches 312 of each register stage are read out through the serial output port S_OUT. In Figure 16, the example dual function register includes three register stages, 500, 502 and 504, where each register stage includes a master latch represented by the box on the left,

20   and a slave latch represented by the box on the right. It is assumed that each register stage has the circuit configuration shown in Figure 12. A sequence diagram shows the signal traces for signals SCK, SEN, CK1 and CK2.

[0056]   In the initial step (1), there are two data words stored in dual function register. Data bits A1, A2 and A3 correspond to a first data word stored in the master latches 310

25   while data bits B1, B2 and B3 correspond to a second data word stored in the slave latches 312. In step (1) there is no data shifting, hence signal SEN remains at the inactive low logic level. Therefore CK1 and CK2 are at the inactive low logic level. Serial output of the first data word from the slave latches 312 is initiated by driving shift enable signal SEN to the high logic level while SCK is at the high logic level. As shown in the sequence diagram of Figure

30   16, CK1 will follow SCK while CK2 will be the complement of CK1 after SEN rises to the high logic level. Gating device 314 will open, thereby allowing the data from slave latches 312 to propagate to the next register stage. In Figure 16, this is shown in step (2) where A1 is

shifted to S_OUT, A2 is shifted to the master latch **310** of register stage **504** and A3 is shifted to the master latch **310** of register stage **502**. In step (3) CK1 and CK2 change logic states, thereby executing an internal shift of data from the master latch **310** to the slave latch **312**.

[0057]   As source clock SCK oscillates, the data bits of the first data word will be shifted from register stage to register stage, until all the data bits of are output through output port S_OUT. Accordingly, the data stored in the master latches **310** is overwritten due to the shifting operations. The "-" designates invalid data which propagates from the S_IN input port during the shifting operations.

[0058]   The above-described shifting sequence in Figure 17 illustrates an example of a serial shift operation of dual function data register **108**, where the data stored in the master latches **310** of each register stage are read out through the serial output port S_OUT. In Figure 17, the example dual function register includes the same three register stages shown in Figure 16. A sequence diagram shows the signal traces for signals SCK, SEN, CK1 and CK2.

[0059]   In the initial step (1), there are two data words stored in dual function register. Data bits A1, A2 and A3 correspond to a first data word stored in the master latches **310** while data bits B1, B2 and B3 correspond to a second data word stored in the slave latches **312**. In step (1) there is no data shifting, because signal SEN remains at the inactive low logic level. Therefore, CK1 and CK2 are at the inactive low logic level. Serial output of the second data word from the master latches **310** is initiated by driving shift enable signal SEN to the high logic level while SCK is at the low logic level. As shown in the sequence diagram of Figure 17, CK2 will follow SCK while CK1 will be the complement of CK2 after SEN rises to the high logic level. Gating device **318** will open, thereby allowing the data from master latches **310** to propagate to the slave latch **312** of the register stage. In Figure 17, this is shown in step (2) where B1, B2 and B3 are all shifted to their respective slave latches **312**. In other words, the first operation is an internal shift of data from the master latch **310** to the slave latch **312**. In step (3) CK1 and CK2 change logic states, and B1 is output to output port S_OUT. At the same time, B2 is shifted to the next register stage, and B3 is shifted to the next register stage. In step (4), and internal data shift is executed, which is the same as in step (2).

[0060]   Therefore, by selecting when SEN is to be active relative to the source clock SCK for the first shift operation, the initial starting phase of CK1 and CK2 can be adjusted by

180 degrees. Therefore, either data word can be serially output by the dual function data register **108**.

[0061]   The capability of the dual function data register **108** to store two data words can be advantageously used in other applications, such as a power up detection application. The purpose of a power up detection operation is to ensure that the power supplies have stabilized sufficiently after the device is powered up, so that read and program operations can be reliably executed. Figure 18 is a block diagram illustrating a power up detection circuit according to an embodiment of the present invention. The power up test system **600** includes a test memory array **602**, wordline drivers **604**, bitline sense amplifiers **606**, and a dual function data register **608**. The test memory array **602** includes two rows of skewed read only memory (ROM) cells, where each memory cell has a threshold voltage that is higher than the programmed anti-fuse memory cells. There can be any number of memory cells per row. Further details of these skewed ROM cells will be shown later. The wordline drivers **604** apply a read voltage for reading the memory cells of a row, where the read voltage can be VREAD used for reading normal memory cells in the memory array **102** of Figure 9 for example. The bitline sense amplifiers **606** are directly connected to the bitlines of the memory array, as no column decoding is required, for sensing voltage levels of the bitlines during the read operations. The same bitline sensing circuitry shown in Figure 11 can be used in Figure 18. The dual function data register **608** includes register stages coupled to each sense amplifier, and each register stage will have the same data storage circuit shown in Figure 12. In the present embodiment, the output of the last register stage is connected to the input of the first register stage.

[0062]   The general principle of the power up detection application is as follows. The test memory array will have two rows of ROM memory cells, where the first row and the second row are hard-wired to store different predetermined data patterns. More specifically, the second row will store the same predetermined data pattern as the first row, but shifted by one bit to the right. Figure 19 illustrates by example a first row Row 1 having a five bit data pattern of "0,1,1,0,1" stored in five different ROM cells, and a second row Row 2 having the shifted data pattern of "1,0,1,1,0". It is noted that the last bit "1" of Row 1 now appears in the first bit position of Row 2. The first row is read out into the dual function data register **608** and then shifted by one bit position to the right. Because of the loop connection between the last register stage and the first register stage, the right-most bit is shifted to the left-most bit in the

first stage of the second row. Then the second row is read out, thereby resulting in dual function data register **608** storing both data words concurrently. The shifted version of the data in Row 1 is compared to the data of Row 2. If they are the same, then the power supplies are deemed to be ready for normal use.

5   [0063]   Therefore, if the power supply is not stable, then reading of the high threshold skewed ROM memory cells will likely yield improper data being read out from the first row or both the first and the second rows. Because the second row of data is shifted by one bit relative to the first row, the shifting functionality of the dual function data register will shift the stored first row of data to realign its bit positions with the second row of data. Once realigned, 10   data comparisons of each bit position can be executed to determine if the data words were properly read out.

[0064]   The skewed ROM memory cells are designed to be properly read out when the power supply voltage is stable and at the proper voltage level. Figures 20a and 20b show examples of mask programmed and unprogrammed skewed ROM cells, respectively. Figure 15   20a is a plan view of a pair of mask programmed single transistor skewed ROM cells. The skewed ROM cell on the left has a first diffusion region **620**, an additional diffusion region **622**, and a polysilicon line **624** overlying a channel region. The additional diffusion region **620** is connected to a diffusion line **626**, resulting in a memory cell that is mask programmed to store a first logic state. The skewed ROM cell on the right consists only of a diffusion region 20   **627**, but more specifically, is a memory cell that does not have a channel region formed under the polysilicon line **624** during fabrication. Accordingly, this memory cell is mask programmed to store a second logic state being opposite to the first logic state. In the present embodiment, an OD2 mask **628** defines areas in which thick gate oxide is to be formed. Since OD2 mask **628** extends over the entire channel region underneath polysilicon 25   line **624**, the gate oxide will have a uniform thickness. The polysilicon line **624** will include a conductive connection **629** to the additional diffusion region **622** via diffusion line **626**, , thereby configuring the skewed memory cell to be similar to the programmed anti-fuse transistor having a conductive link shown in Figure 6b. The first diffusion region **620** is coupled to a bitline, and is raised to VREAD when the polysilicon line is driven with VREAD.

30   [0065]   This skewed ROM cell is designed in such a way that it requires a more stable power supply voltage to turn on than the most weakly programmed anti-fuse memory cell. More specifically, when WL is driven to VREAD, its bitline BL will be pulled up towards VDD

if the power supply driving WL is sufficiently high. Accordingly, the voltage on BL will be at least the elevated threshold voltage. To achieve this desired electrical characteristic, the skewed ROM cell can be a high voltage, long channel transistor with an elevated threshold voltage via channel implantation. Those skilled in the art will understand that the skewed

5   ROM cell can be configured differently in order to achieve the desired electrical characteristics.

[0066]   Because the unprogrammed skewed ROM cell in the right of Figure 20a does not have a channel region under polysilicon line **624**, the bitline will remain at a VSS precharged voltage level when WL is driven to the read voltage level.

10

[0067]   Figure 20b is a plan view of a pair of mask programmed two transistor skewed ROM cells. The two transistor skewed ROM cell on the left side of Figure 20b includes the same first diffusion region **620**, additional diffusion region **622**, and polysilicon line **624**, now referred to as an OTP_WL, overlying a channel region. These features are present in the

15   single transistor skewed ROM cell of Figure 20a. Because memory cell includes two transistor devices, there is a second polysilicon line **621** referred to as the wordline, and a shared diffusion region **623** between the polysilicon line **621** and the polysilicon line **624**. In the present example, the left side two transistor skewed ROM cell is mask programmed to store a first logic level as it has the additional diffusion region **622** coupled to the diffusion

20   line **626**. The diffusion line **626** is coupled via conductive connection **629** to polysilicon line **624**. Therefore, when OTP_WL is biased to a positive voltage level and the WL is selected, diffusion region **620** will be coupled to OTP_WL, thereby raising its corresponding bitline to the positive voltage level from low precharge level. On the other hand, because the unprogrammed skewed ROM cell in the right of Figure 20b omits a channel region under

25   polysilicon line **624**, the bitline it is connected to will remain at a VSS precharged voltage level when WL is driven to the read voltage level. It is noted that OD2 mask **628** extends over the entire channel region underneath polysilicon line **624**, hence the gate oxide underneath it will have the same thickness the gate oxide underneath polysilicon line **621**.

[0068]   Figure 21a is a schematic showing an array of skewed ROM cells which can

30   be used in test memory array **602** of Figure 18, according to a present embodiment. This array includes two rows, and each row includes three single transistor skewed ROM cells having the configuration shown in Figure 20a. Therefore, the same reference numbers are

used to identify the features first shown in Figure 20a. The first row includes skewed ROM cells **630**, **632** and **634** having their gates connected to wordline WL1, where cells **630** and **634** are mask programmed to store the logic 1 state. The second row includes skewed ROM cells **636**, **638** and **640** having their gates connected to wordline WL2, where cells **636** and

5   **638** are mask programmed to store the logic 1 state. Therefore, the data pattern of the row corresponding to WL1 is 1,0,1 while the data pattern of the row corresponding to WL2 is 1,1,0. The data pattern of the row corresponding to WL2 is shifted by one bit to the right relative to the row corresponding to WL1, where the rightmost bit has been shifted to the leftmost bit position. Three test bitlines, T_BL1, T_BL2 and T_BL3 are connected to the

10   common bitline contacts shared by memory cell pairs **630/636**, **632/638**, and **634/640**.

[0069]   Figure 21b is a schematic showing an array of skewed ROM cells which can be used in test memory array **602** of Figure 18, according to another embodiment. This array includes two rows, and each row includes three two transistor skewed ROM cells having the configuration shown in Figure 20b. Therefore, the same reference numbers are used to

15   identify the features shown in Figure 20b. The first row includes skewed ROM cells **631**, **633** and **635** having their gates connected to wordline WL1, where cells **631** and **635** are mask programmed to store the logic 1 state. The second row includes skewed ROM cells **637**, **639** and **641** having their gates connected to wordline WL2, where cells **637** and **639** are mask programmed to store the logic 1 state. Therefore, the data pattern of the row corresponding

20   to WL1 is 1,0,1 while the data pattern of the row corresponding to WL2 is 1,1,0. The data pattern of the row corresponding to WL2 is shifted by one bit to the right relative to the row corresponding to WL1, where the rightmost bit has been shifted to the leftmost bit position. Three test bitlines, T_BL1, T_BL2 and T_BL3 are connected to the common bitline contacts shared by memory cell pairs **631/637**, **633/639**, and **635/641**.

25   [0070]   Figure 21c is an illustration showing the logic states mask programmed into the skewed ROM cells corresponding to the memory cell positions shown in Figures 21a and 21b.

[0071]   Figure 22 is a circuit schematic of one register stage of the dual function data register, according to the present embodiment. The register stage **650** includes a data

30   storage circuit **652** and a data comparison circuit consisting of XOR gate **654** and AND gate **656**. The data storage circuit **652** includes substantially the same components as data storage circuit **302** of Figure 12, but does not include parallel input or output terminals for

receiving or providing data in parallel. Data storage circuit **652** includes serial input gating device **658**, transfer gating device **660**, a master latch **662** and a slave latch **664**. The gate terminal of gating device **658** receives clock signal CK1 and the gate terminal of gating device **660** receives clock signal CK2. Clock signals CK1 and CK2 can be generated by the same clock controller shown in Figure 13. Accordingly, the data shifting operation of data storage circuit **652** will be the same as the data shifting operation previously described for data storage circuit **302** of Figure 12. The input of slave latch **664** receives read data from the sense amplifier provided from terminal SAi_out.

[0072]   The data comparison circuit performs two functions. First, it compares the data stored in the master latch **662** with the data stored in the slave latch **664**, and provides an output corresponding to the comparison. This first function is executed by XOR gate **654**, hence a low logic level output is provided when both the master latch **662** and the slave latch **664** store the same logic states. The second function is to combine the XOR gate output with a similar output from a previous register stage. This second function is executed by AND gate **656** which has a first input connected to the output or XOR gate **654** and a second input connected to POWERi-1, where i is any non-zero integer value between 1 and n, and n corresponds to the last register stage of the dual function data register **608**. In particular, POWERi-1 is provided from an AND gate **656** of the previous register stage. It is noted that in the first register stage, POWERi-1 is replaced with a connection to VDD. The output of AND gate **656** is signal POWERi, which is provided to the next register stage, and in particular to a second input of an AND gate **656** of the next register stage. Therefore, if every register stage has the same logic level stored in their master and slave latches, then the last POWERi signal will be at the low logic level, indicating that every data bit of the first row of data and of the second row of data match. This means that the power supplies are stable enough for normal operation of the memory device.

[0073]   Figure 23 is a flow chart of a power up detection method, according to an embodiment of the present invention, using the power up test system **600** of Figure 18. Figure 24 illustrates the propagation of the data bits during the power up detection method, and a sequence diagram shows the signal traces for signals SCK, SEN, CK1 and CK2. The method will be described using the example test memory array shown in Figure 18. In Figure 24, the dual function data register includes three register stages **670**, **672** and **674**, where each register stage has the circuit configuration shown in Figure 22. The method starts at

step **700** by reading out the first test word by activating wordline WL1. The data is sensed and latched in the slave latches **664** of each register stage **650**. It is noted that both CK1 and CK2 are maintained at the low logic level while data is latched by the slave latches **664**, by maintaining SEN at the low logic level. As shown for step (**700**) of Figure 24, the slave

5   latches **664** (right boxes) of register stages **670**, **672** and **674** will store logic 1,0,1 if the power supply is at the proper levels. If the power supply is not stable, then the data word stored in register stages **670**, **672** and **674** will be different. It is noted that the master latches **662** of register stages **670**, **672** and **674** stores invalid or don't care data.

[0074]   At step **702**, the first test word stored in register stages **670**, **672** and **674** is

10   shifted one bit position to the right. This is done by pulsing SEN to the active high logic level while SCK is at the high logic level. However, since only a single bit shift operation is required, SEN will be at the high logic level for a period of time no longer than the high logic level of SCK. Therefore, CK1 will be at the high logic level for the same time period that SEN is at the high logic level, to turn on the serial input gating devices **658** of each register stage.

15   As shown for step (**702**) of Figure 24, all the bits stored in the slave latches **664** are shifted to the master latches **662** of the next register stage, with the exception of the last bit of register stage **674** which is shifted, or fed back, to the master latch **662** of register stage **670**. Following at step **704** the second test word is read out, sensed, and then latched in the slave latches **664** of register stages **670**, **672** and **674**. As shown for step (**704**) of Figure 24, the

20   data pattern 1,1,0 is now latched into the slave latches **664** of the register stages if the power supply is stable.

[0075]   Now that both the first and second test words are stored in the dual function data register **608**, the shifted version of the first test word is compared to the second test word determine if they are identical in step **706**. In Figure 22, the XOR gates **654** of each

25   register stage will compare the logic states of the master latch **662** to the slave latch **664**. The output of each XOR gate **654** is combined with a comparison result POWERi-1of the previous stage, and the combined result POWERi is fed to the AND gate **656** of the next register stage. It is noted that the first register stage **670** will either not have an AND gate **656**, or will have an input of its AND gate **656** connected to VDD. At step **708**, if the POWERi

30   result from the AND gate **656** of the last stage is at the high logic level, this indicates that all bits of the shifted first test word match those of the second test word and the power up test

has passed, as indicated at step **710**. If the data mismatches, then the system will wait for a predetermined period of time at step **712** before repeating the method at step **700**.

[0076]   The previously described power up detection method of Figure 23 uses the power up test system **600** of Figure 18. According to a further embodiment of the present

5   invention, the test memory array **602** is integrated with a normal memory array, such as memory array **102** of Figure 9. For example, two rows of skewed ROM cells having the same number of cells in a row of memory array **102** can be added anywhere in the memory array **102** and connected to the same bitlines. The dual function data register **108** can be slightly modified to further provide the data match operation described in the method of Figure 23.

10   This embodiment will minimize circuit overhead since many of the circuit elements in use for normal operations can be used for power up detection.

[0077]   Figure 25 is a circuit schematic of an alternate register stage of the dual function data register **108** shown in Figure 10, according to an embodiment of the present invention. Register stage **800** is identically configured to register stage **300** shown in Figure

15   12, except that status circuit **304** of Figure 12 is replaced with modified status circuit **802** in the embodiment of Figure 25. Hence the same circuit elements will have the same reference numerals. Modified status circuit **802** includes the same XOR logic gate **330**, AND logic gate **332**, AND gate **334** and inverter **336** of status circuit **304**, but now includes a selector switch **804** positioned at an input of XOR logic gate **330**. More specifically, selector switch **804** is

20   implemented as a 2-to-1 multiplexor (MUX) having a first input terminal connected to an output of inverter **336**, a second input terminal connected to the input of inverter **336**, and an output connected to the input of XOR logic gate **330**. Therefore, MUX **804** is controlled to pass either an inverted or non-inverted output of master latch **310** to XOR gate **330**, in response to selection signal P_TEST.

25   [0078]   In a normal mode of operation, ie. a program verify mode of operation, P_TEST will be at a first logic state to couple the output of inverter **336** to XOR gate **330**. Accordingly, the functionality and operation of register stage **800** will be identical to that of register stage **300** of Figure 12. In the power up mode of operation, P_TEST will be at a second logic state to couple the input of inverter **336** to XOR gate **330**. Then the functionality

30   and operation of register stage **800** will be identical to that of register stage **650** of Figure 22. The last signal Cn_a can be monitored in order to determine if the data stored in the master latch **310** and the slave latch **312** of each register stage are identical.

[0079]   In the preceding description, for purposes of explanation, numerous details
are set forth in order to provide a thorough understanding of the embodiments of the
invention. However, it will be apparent to one skilled in the art that these specific details are
not required in order to practice the invention. In other instances, well-known electrical

5   structures and circuits are shown in block diagram form in order not to obscure the invention.
For example, specific details are not provided as to whether the embodiments of the
invention described herein are implemented as a software routine, hardware circuit,
firmware, or a combination thereof.

[0080]   The above-described embodiments of the invention are intended to be

10   examples only. Alterations, modifications and variations can be effected to the particular
embodiments by those of skill in the art without departing from the scope of the invention,
which is defined solely by the claims appended hereto.

What is claimed is:

1.   A method for executing a program verify operation, comprising:

a) loading program data into a first latch of a register stage coupled to at least one
     bitline of a memory array;

b) programming a memory cell coupled to the at least one bitline;

c) reading the programmed data of the memory cell coupled to the at least one bitline
     into a second latch of the register stage;

d) comparing logic states stored in the first latch and the second latch; and

e) providing a local status signal corresponding to a program pass condition if opposite
     data states are stored in the first latch and the second latch.

2.   The method of claim 1, wherein the step of loading includes shifting the program data
serially into the first latch, the first latch and the second latch being arranged in a master-
slave flip-flop configuration.

3.   The method of claim 1, wherein the step of loading includes providing the program
data in parallel into the first latch, the first latch and the second latch being arranged in a
master-slave flip-flop configuration.

4.   The method of claim 2, wherein the first latch is coupled to one of the bitlines prior to
the step of programming.

5.   The method of claim 3, wherein the step of reading includes sensing the programmed
data with a sense amplifier and providing sensed data to the second latch.

6.   The method of claim 1, wherein the step of comparing includes providing a single bit
output corresponding to the result of comparing.

7.   The method of claim 6, wherein the step of comparing further includes combining the
single output with a previous local status signal to provide the local status signal, the
previous local status signal corresponding to a comparison result from a previous register

- 33 -

stage.

8. The method of claim 7, further including inverting the program data stored in the first latch if the second latch stores a data state corresponding to a programmed memory cell.

9. The method of claim 8, wherein the step of comparing further includes comparing the data states of the first latch and another first latch in the previous register stage.

10. The method of claim 9, wherein the step of comparing the data states of the first latch and another first latch in the previous register stage includes providing another local status signal if the first latch and the another first latch have the same predetermined data state, the predetermined data state corresponding to the inverted program data.

11. A method for operating a register stage having a first latch and a second latch arranged in a master-slave flip-flop configuration for storing a first bit of data and a second bit of data respectively, comprising:

   a. decoupling the first latch from a serial input terminal and decoupling the second latch from the first latch at substantially the same time while a source clock signal oscillates; and,

   b. initiating a shifting operation on either a high logic level of the source clock or a low logic level of the source clock for shifting one of the first bit of data and the second bit of data.

12. The method of claim 11, wherein the step of decoupling includes turning off a first gating device between the first latch and a serial input terminal and turning off a second gating device between the first latch and the second latch.

13. The method of claim 12, wherein turning off includes driving a first clock signal to an inactive logic level for turning off the first gating device, and driving a second clock signal to the inactive logic level for turning off the second gating device.

14. The method of claim 13, wherein the first clock signal and the second clock signal are driven to the inactive logic level by maintaining a shift enable signal at the inactive logic level while the source clock signal oscillates.

- 34 -

15.   The method of claim 12, wherein the step of initiating includes turning on the first gating device while the second gating device is turned off for shifting the second bit of data.

16.   The method of claim 15, wherein turning on the first gating device includes driving a first clock signal to an active logic level.

5   17.   The method of claim 16, wherein the first clock signal is driven to the active logic level when a shift enable signal is enabled while the source clock signal is at a first logic level, the second clock signal being driven to an inactive logic level when the shift enable signal is enabled while an inverted source clock signal is at a second logic level.

18.   The method of claim 17, wherein the step of initiating includes turning on the second gating device while the first gating device is turned off for shifting the first bit of data.

19.   The method of claim 18, wherein turning on the second gating device includes driving the second clock signal to the active logic level.

20.   The method of claim 19, wherein the second clock signal is driven to the active logic level when the shift enable signal is enabled while the source clock signal is at the second logic level, the first clock signal being driven to the inactive logic level when the shift enable signal is enabled while an inverted source clock signal is at the first logic level.

1/20
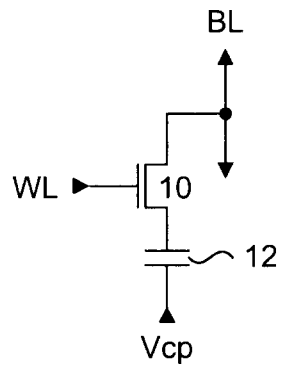
Figure 1 (Prior Art)
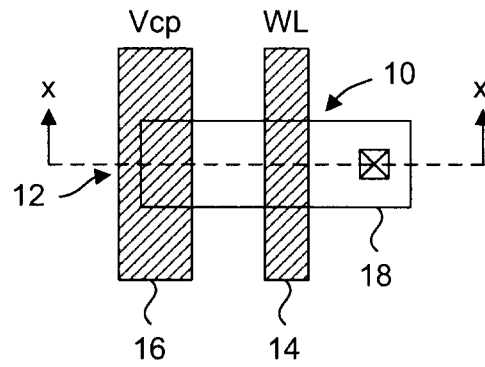
Figure 2 (Prior Art)
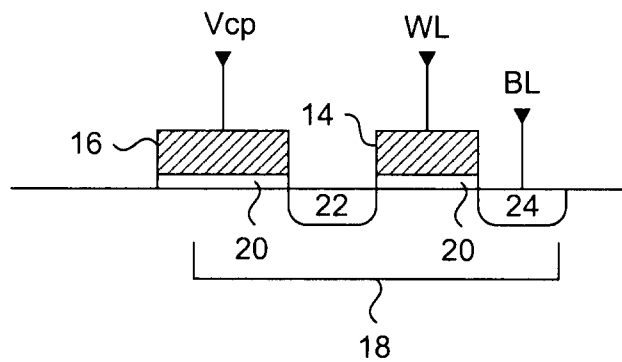
Figure 3 (Prior Art)

2/20
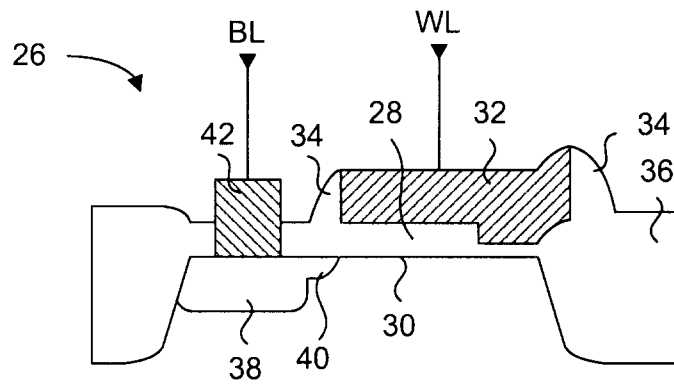
Figure 4a

Figure 4b

48

58   50   62

54   52   64   66

56   60

Figure 5a

48

70

58

B   B

50   62

68

Figure 5b

4/20

BL=VSS=0

———— WL=VPP

26

Figure 6a

BL

72 ⌇

———— WL

26

Figure 6b

BL=VSS=0

———— WL=VREAD

72 ⌇

26
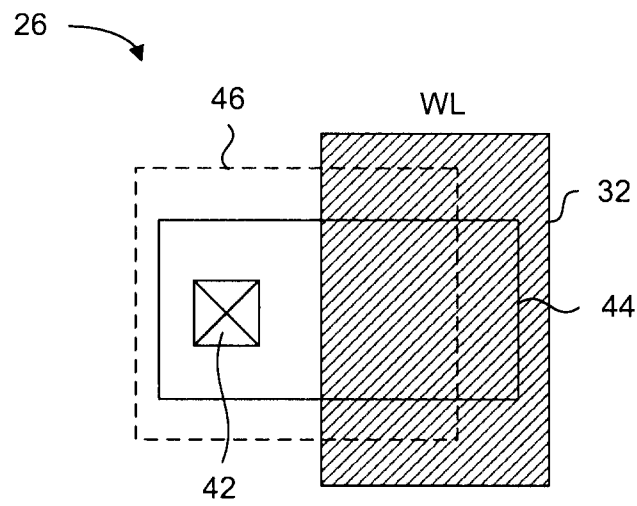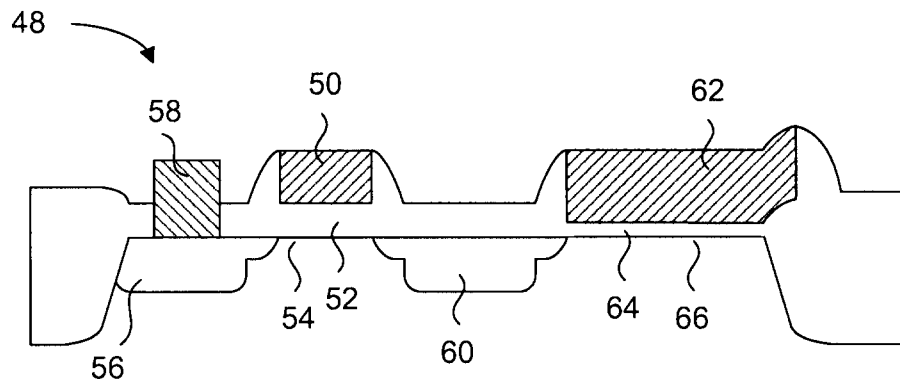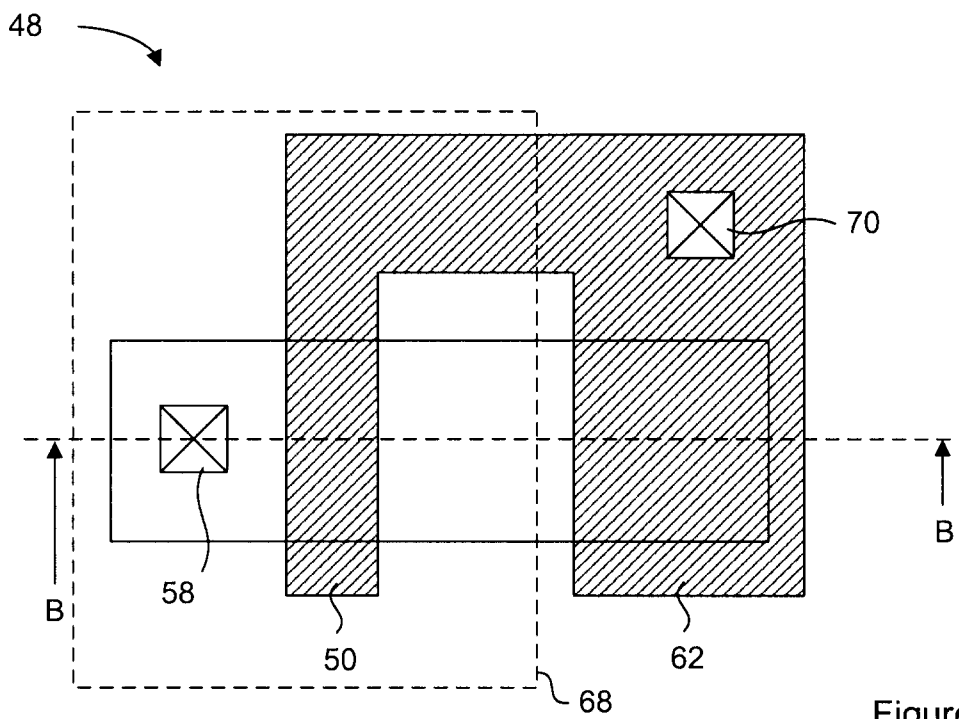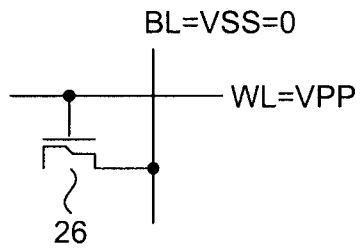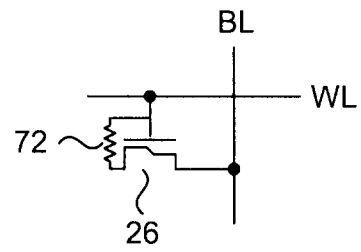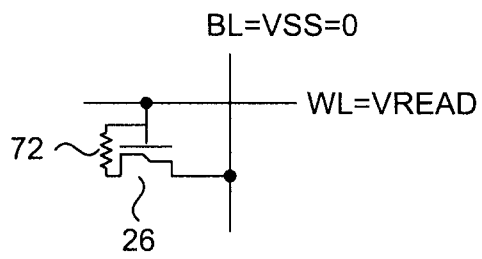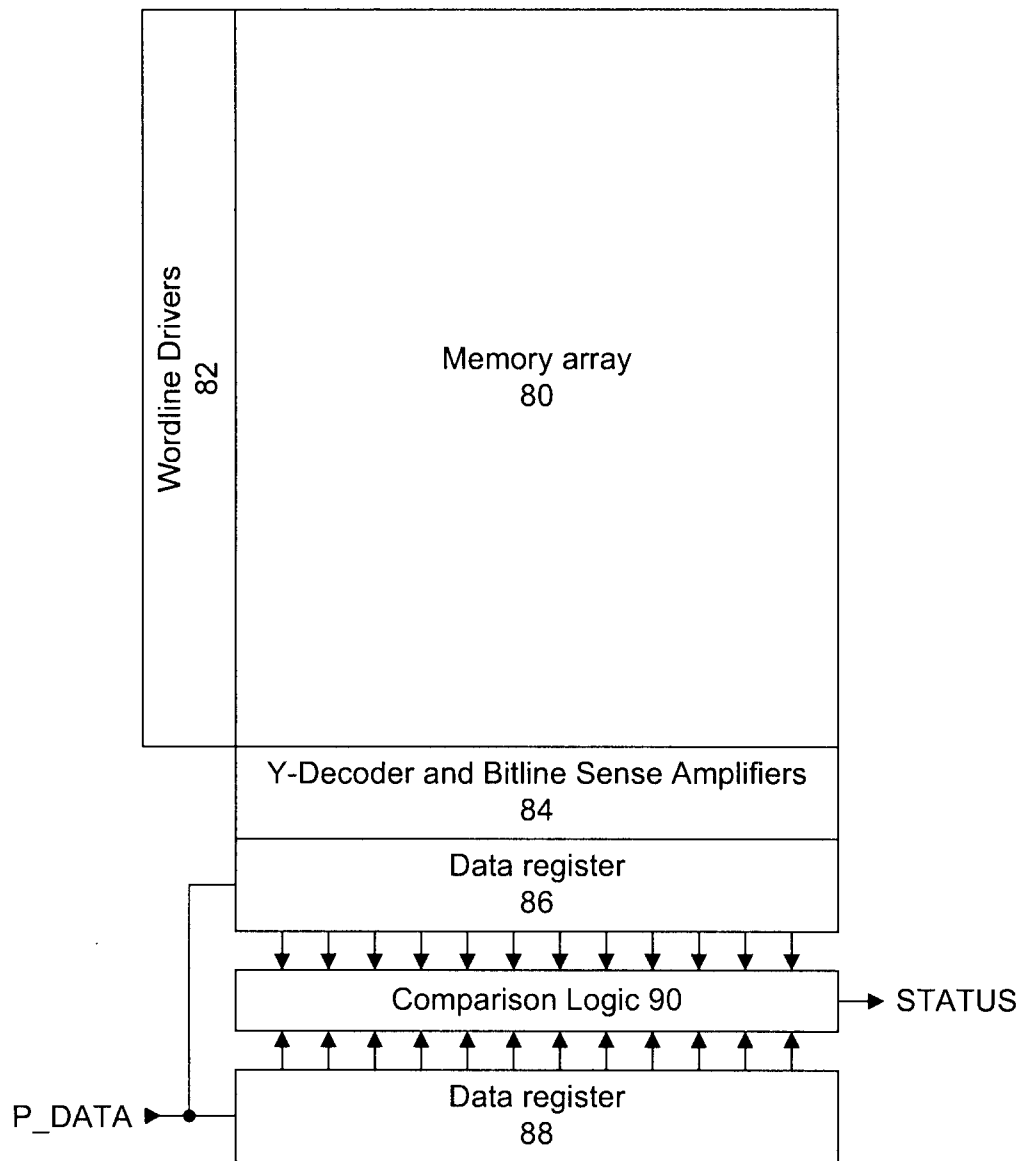
Figure 7

Figure 8 (Prior Art)

6/20

100 —⟍

Figure 9

7/20
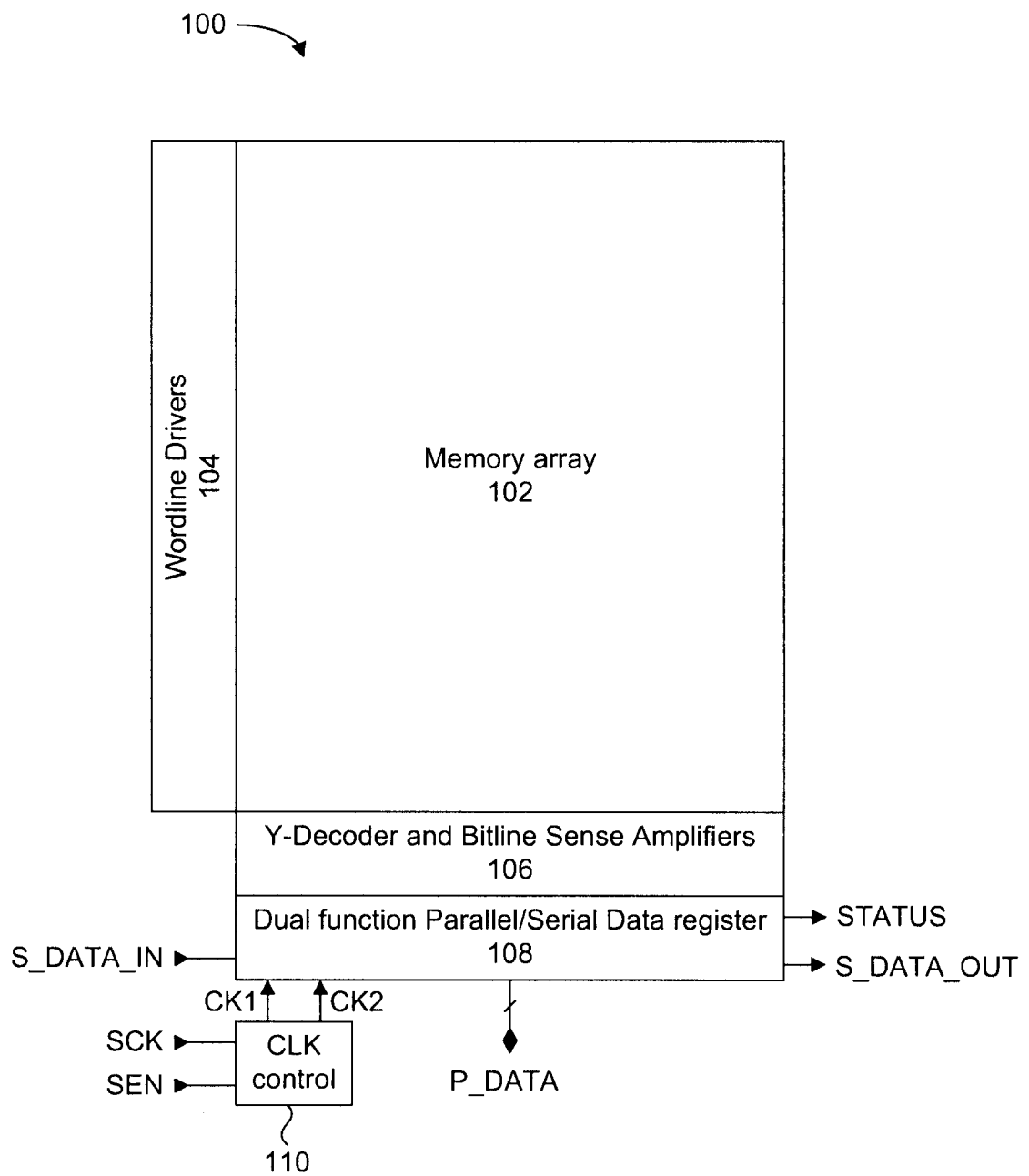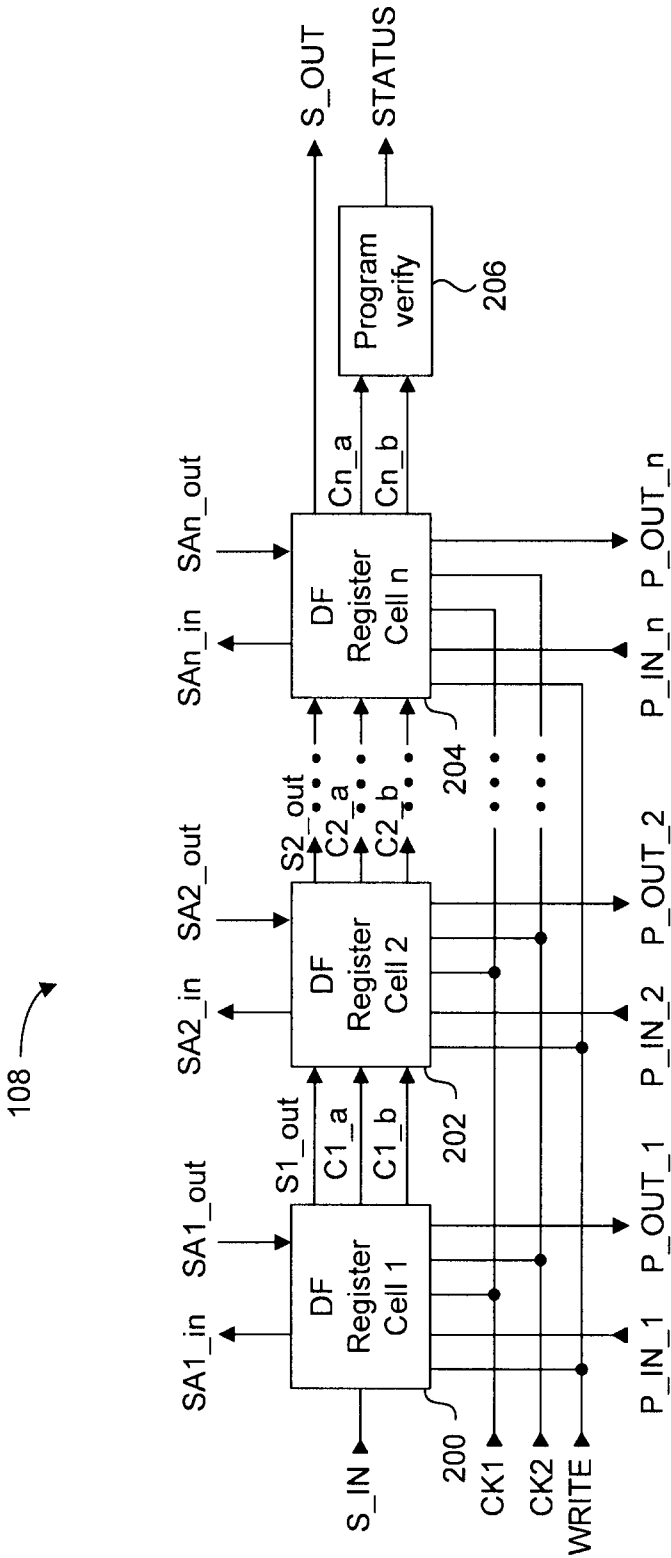
Figure 10

8/20
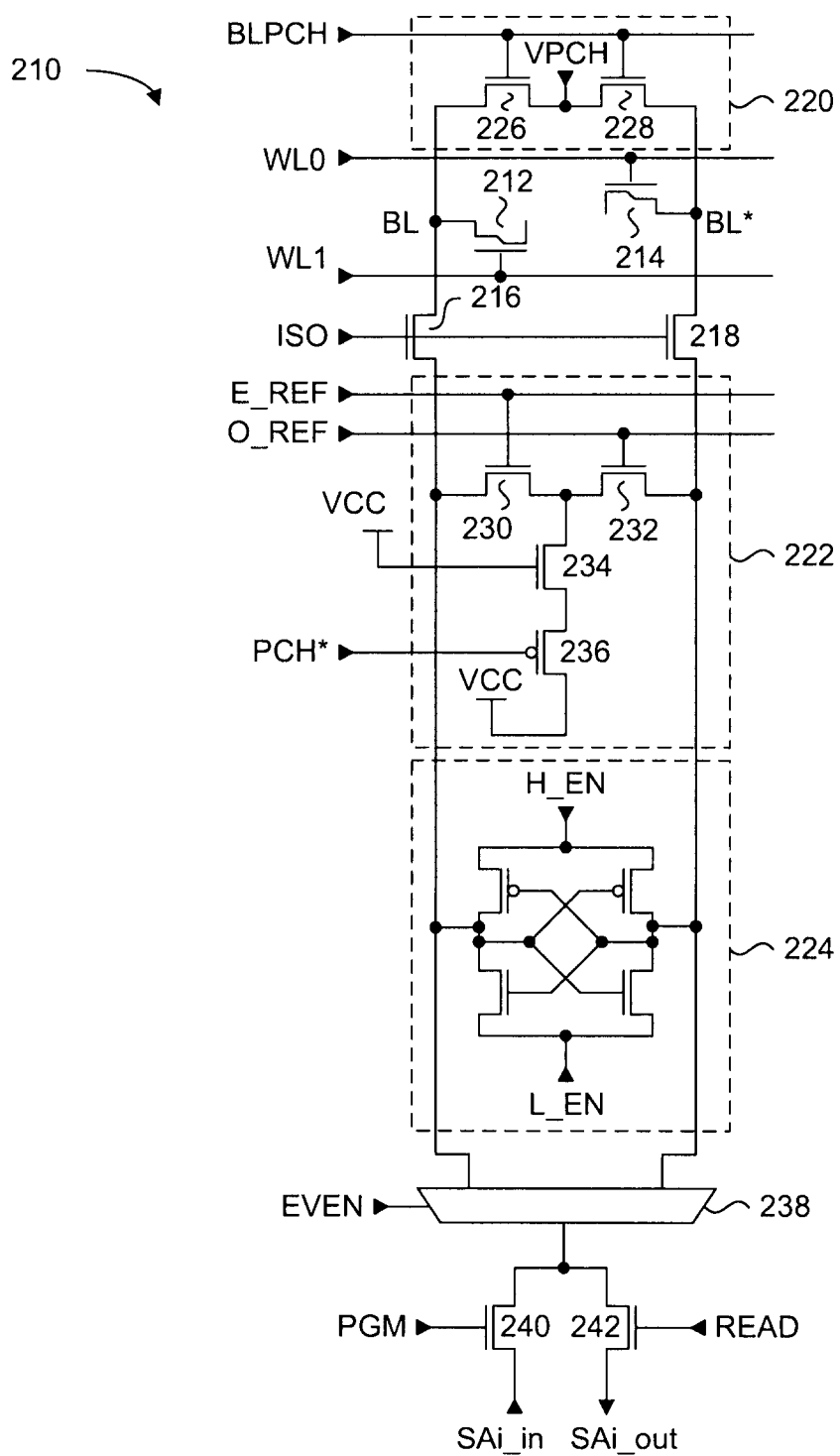
Figure 11

9/20
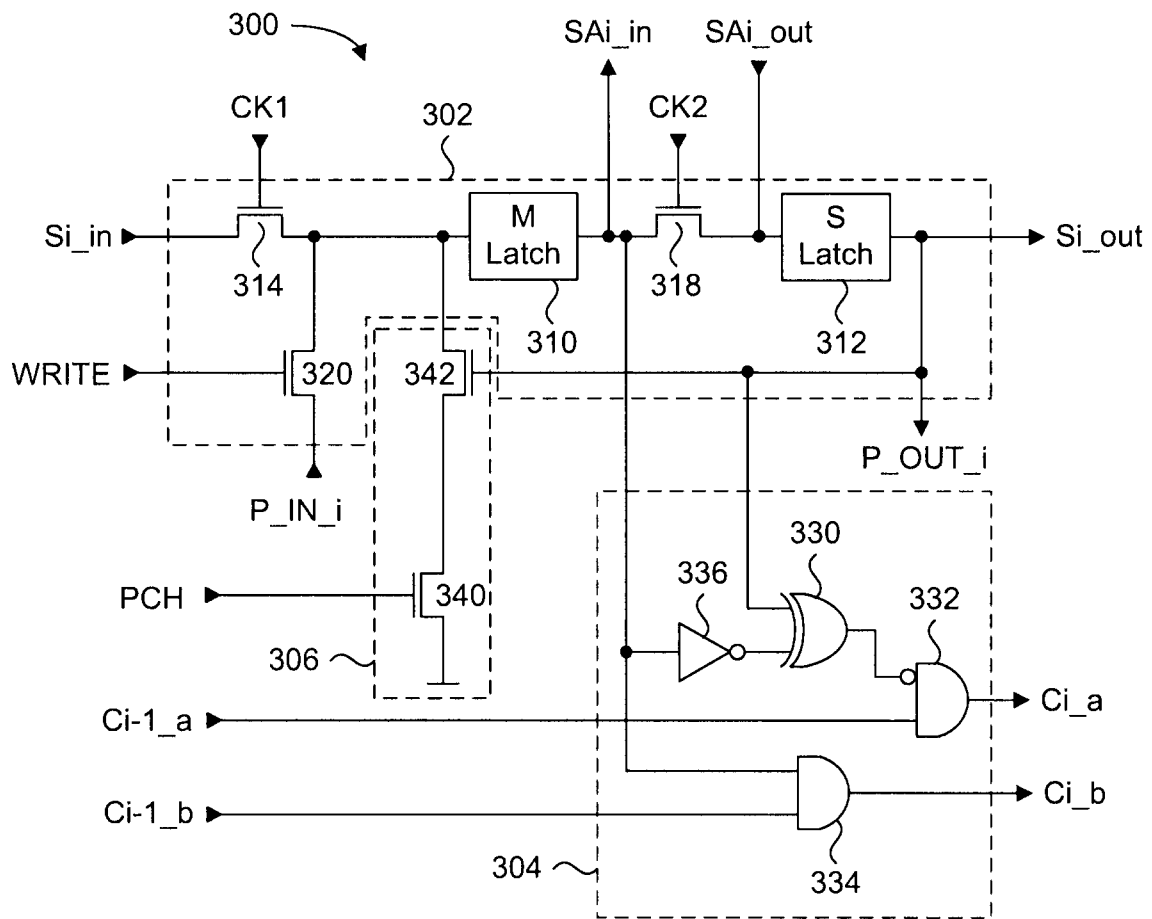
Figure 12

Figure 13

Figure 14

```
         ┌──────────────────────┐
400 ∿────│  Load Program data   │
         │   into Master Latch  │
         └──────────┬───────────┘
                    │
                    ▼
         ┌──────────────────────┐
402 ∿────│  Pass Program data to│
         │       bitline        │
         └──────────┬───────────┘
                    │
                    ▼
         ┌──────────────────────┐
404 ∿────│  Program memory cell │◄───────────────┐
         └──────────┬───────────┘                │
                    │                             │
                    ▼                             │
         ┌──────────────────────┐                │
         │   Read programmed    │                │
406 ∿────│ memory cell into Slave│               │
         │        latch         │                │
         └──────────┬───────────┘                │
                    │                             │
                    ▼                             │
         ┌──────────────────────┐                │
408 ∿────│    Data compare      │                │
         └──────────┬───────────┘                │
                    │                  ┌──────────────────────┐
                    ▼                  │  Reset Master latch of│
                 ╱──────╲              │  programmed memory   │
410 ∿──────── ╱ All memory ╲──No─────►│        cells         │
              ╲   cells    ╱          └──────────┬───────────┘
               ╲programmed?╱                     │
                ╲────┬───╱                      414
                   Yes
                    │
                    ▼
         ┌──────────────────────┐
412 ∿────│    Program pass      │
         └──────────────────────┘
```
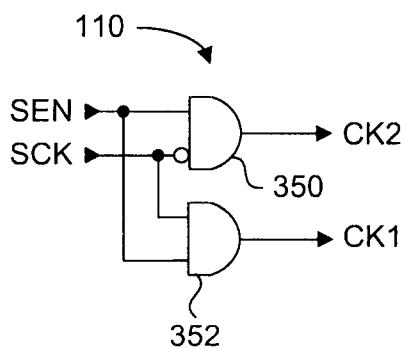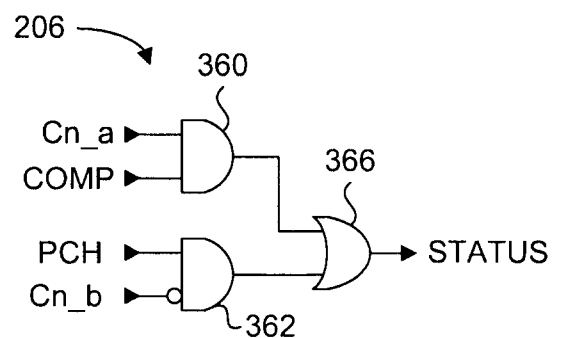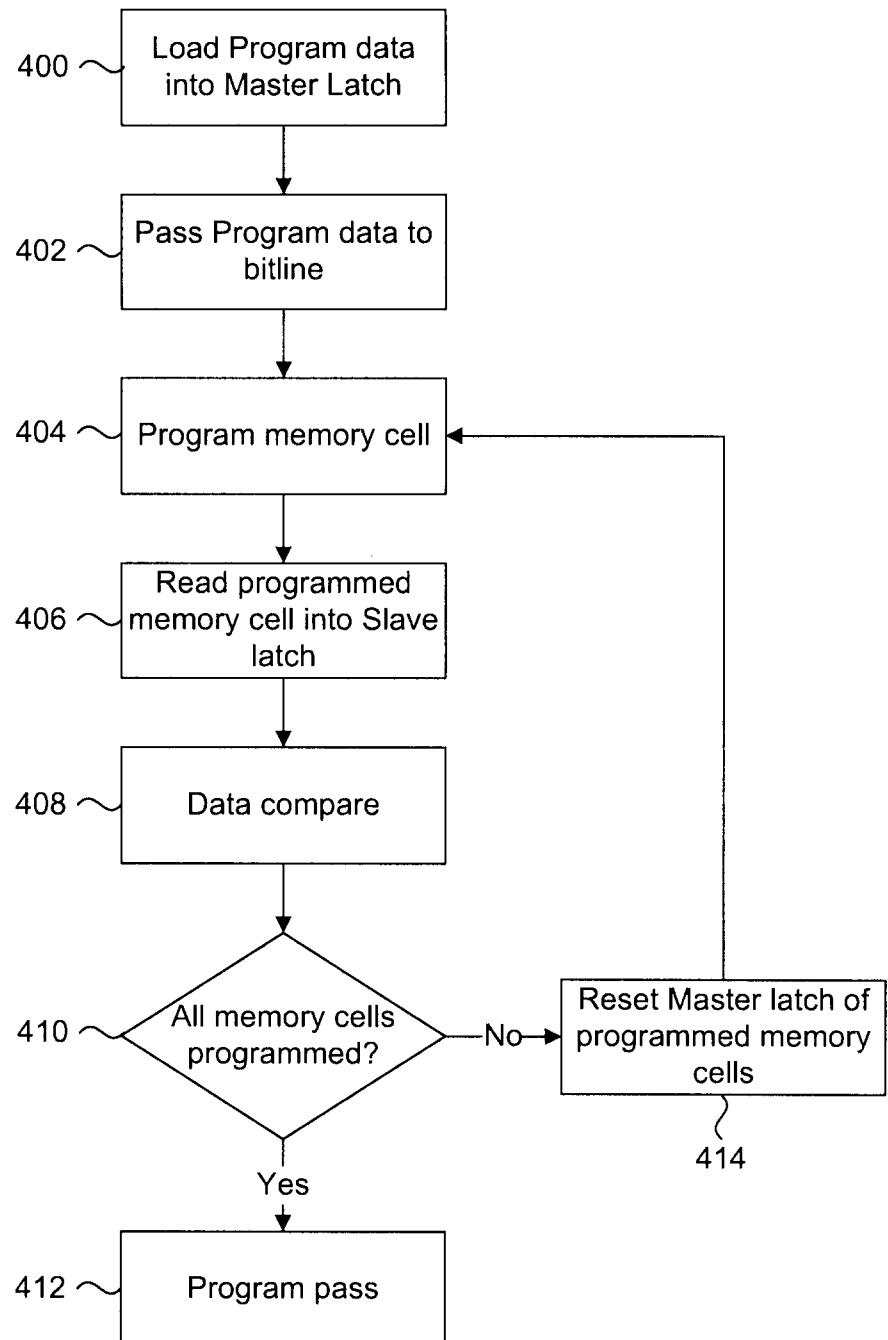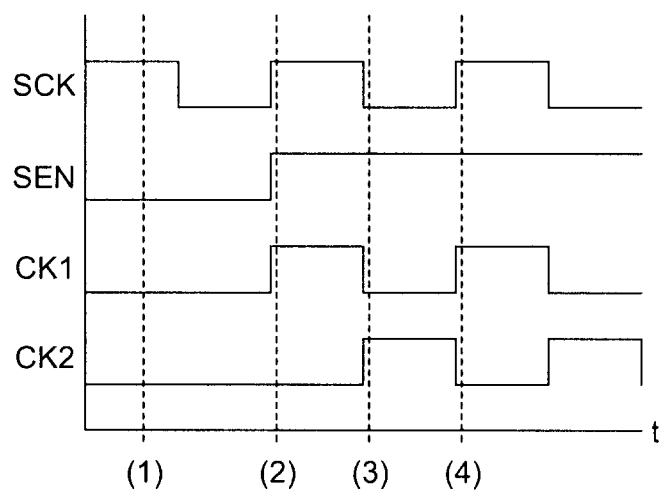
Figure 15

11/20

Figure 16

12/20

Figure 17

600

604

WL Drivers	Test Memory array 602
	Bitline Sense Amplifiers 606
S_IN →	Dual function Parallel/Serial Data register 608	→ POWER
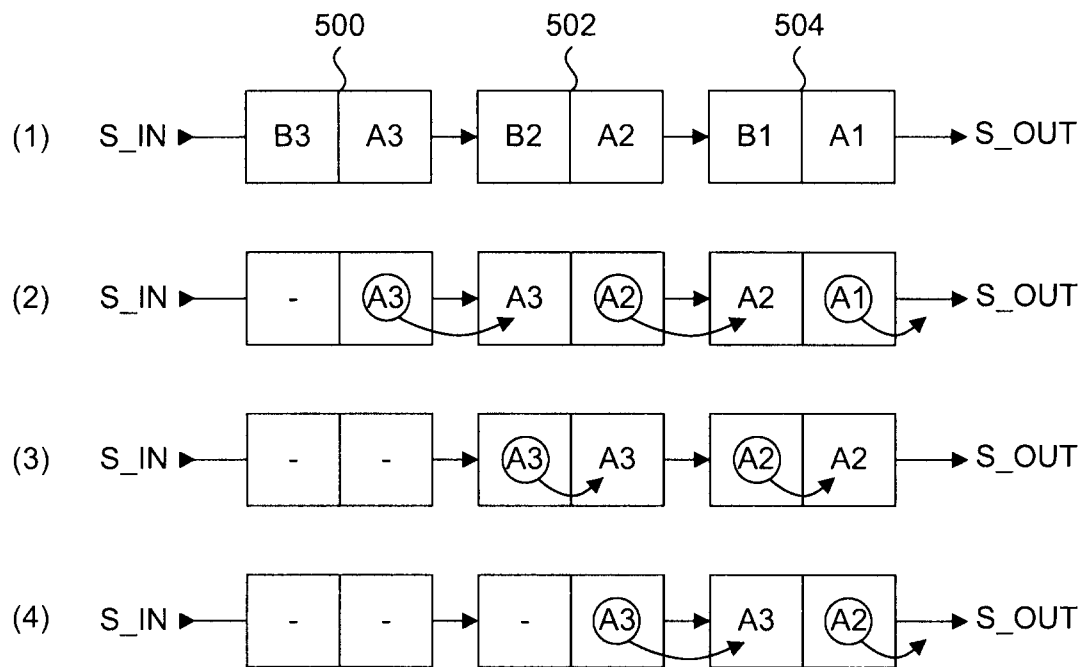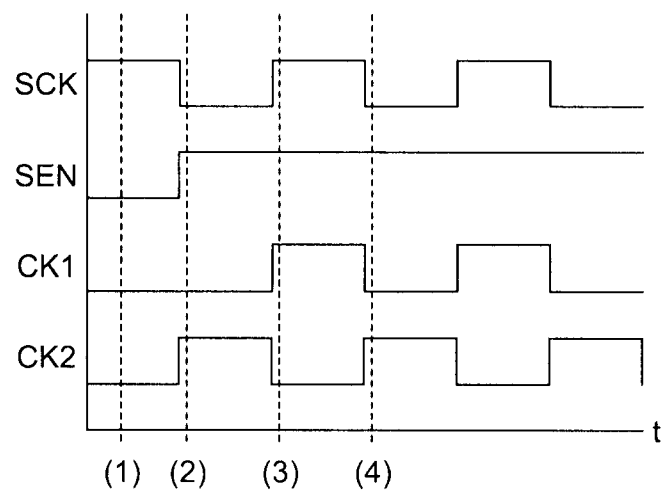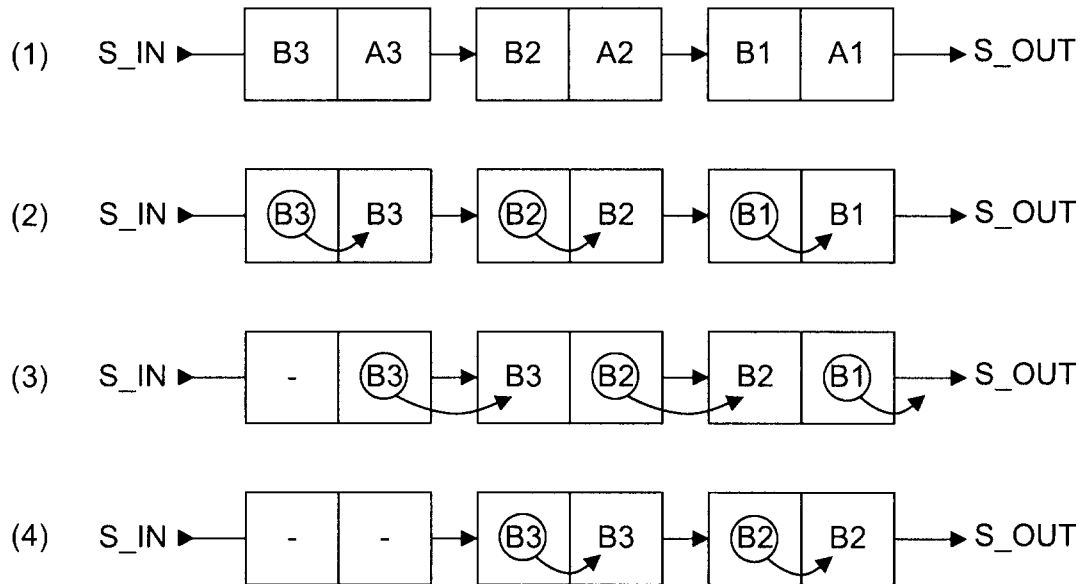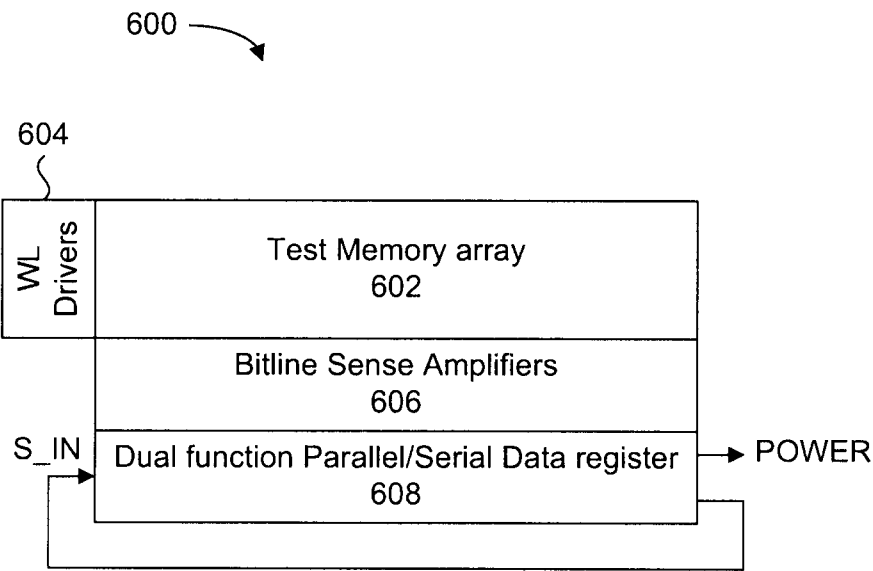
Figure 18

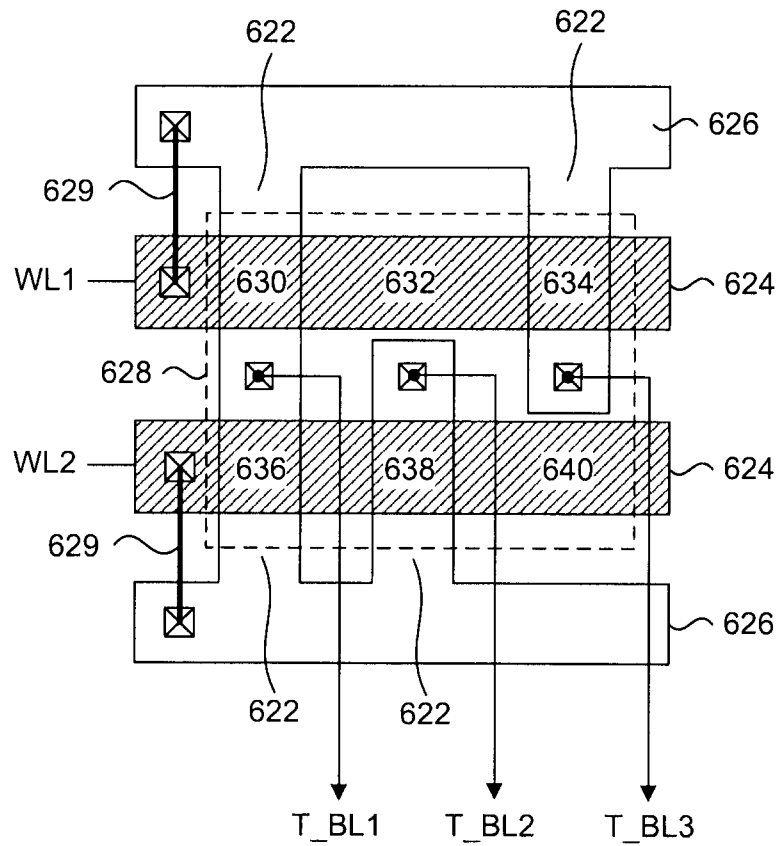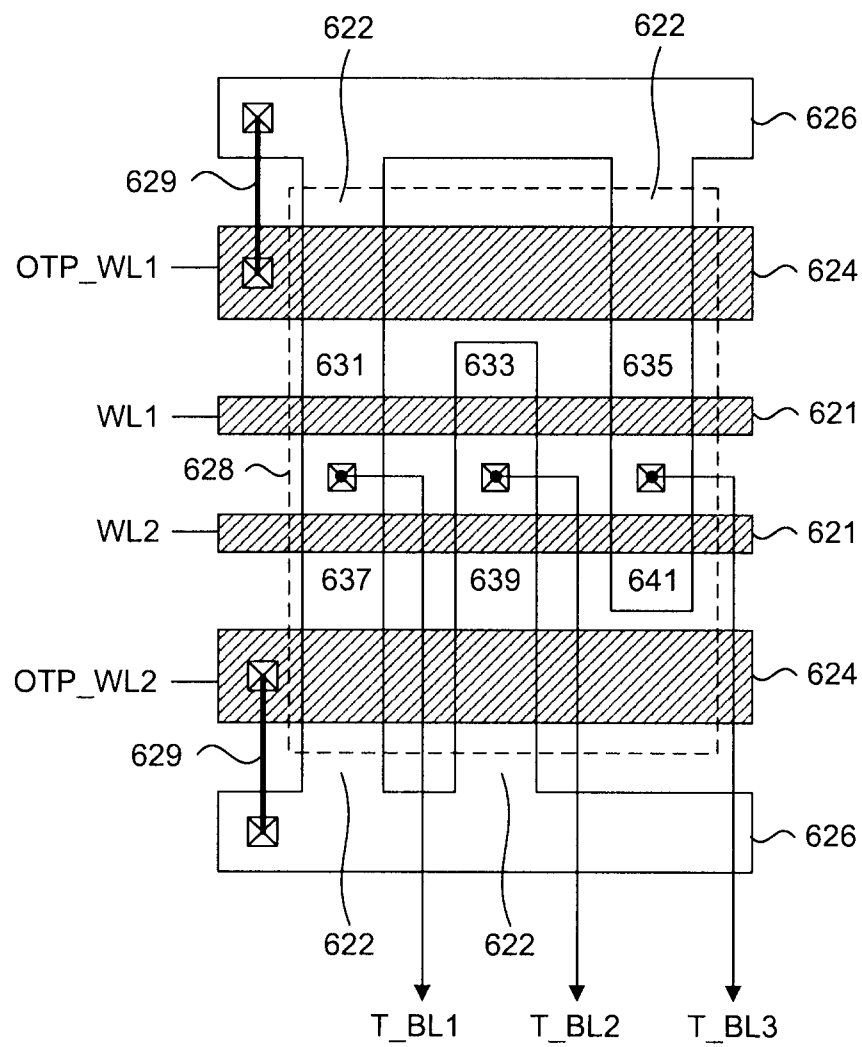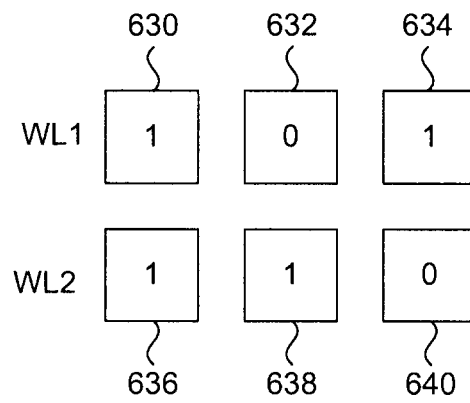| Row 1	0	1	1	0	1
Row 2	1	0	1	1	0

Figure 19

14/20
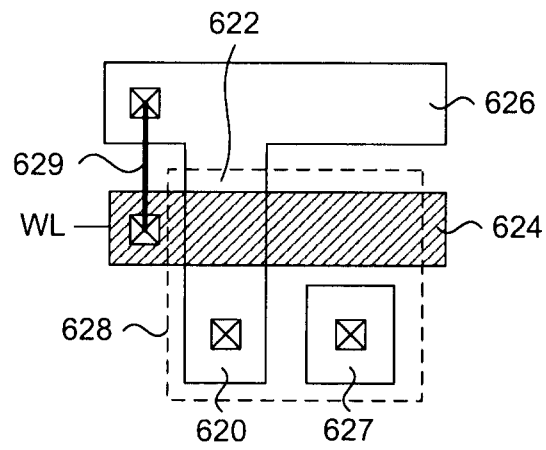
Figure 20a
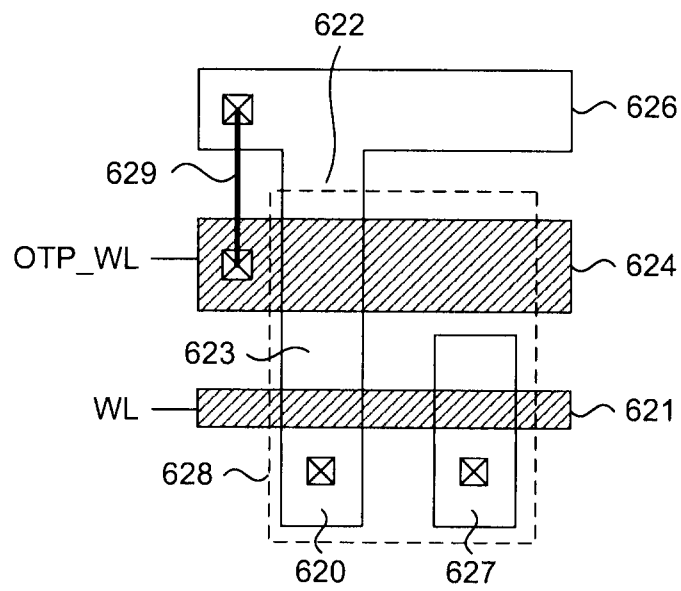
Figure 20b

Figure 21a

Figure 21b
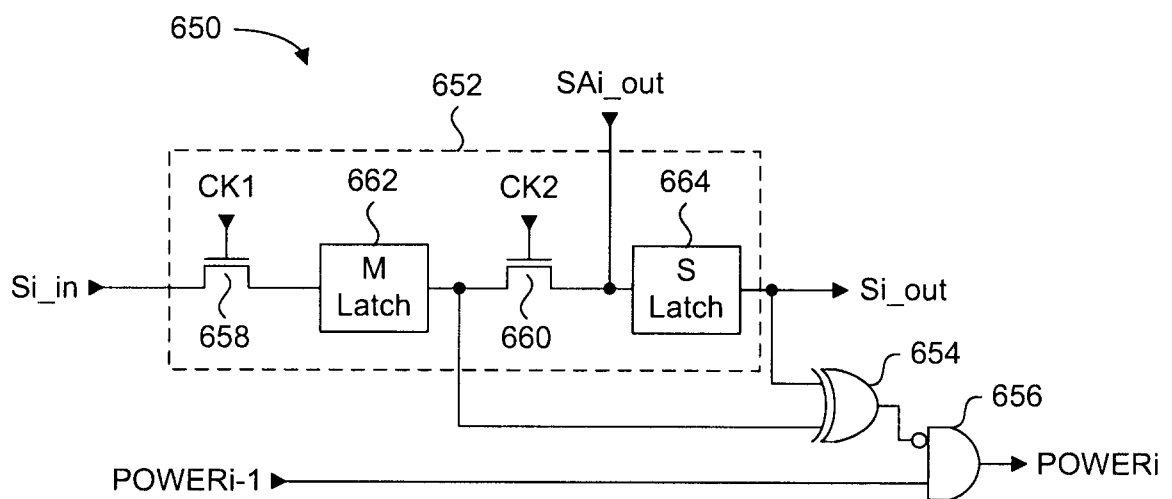
Figure 21c

Figure 22

18/20
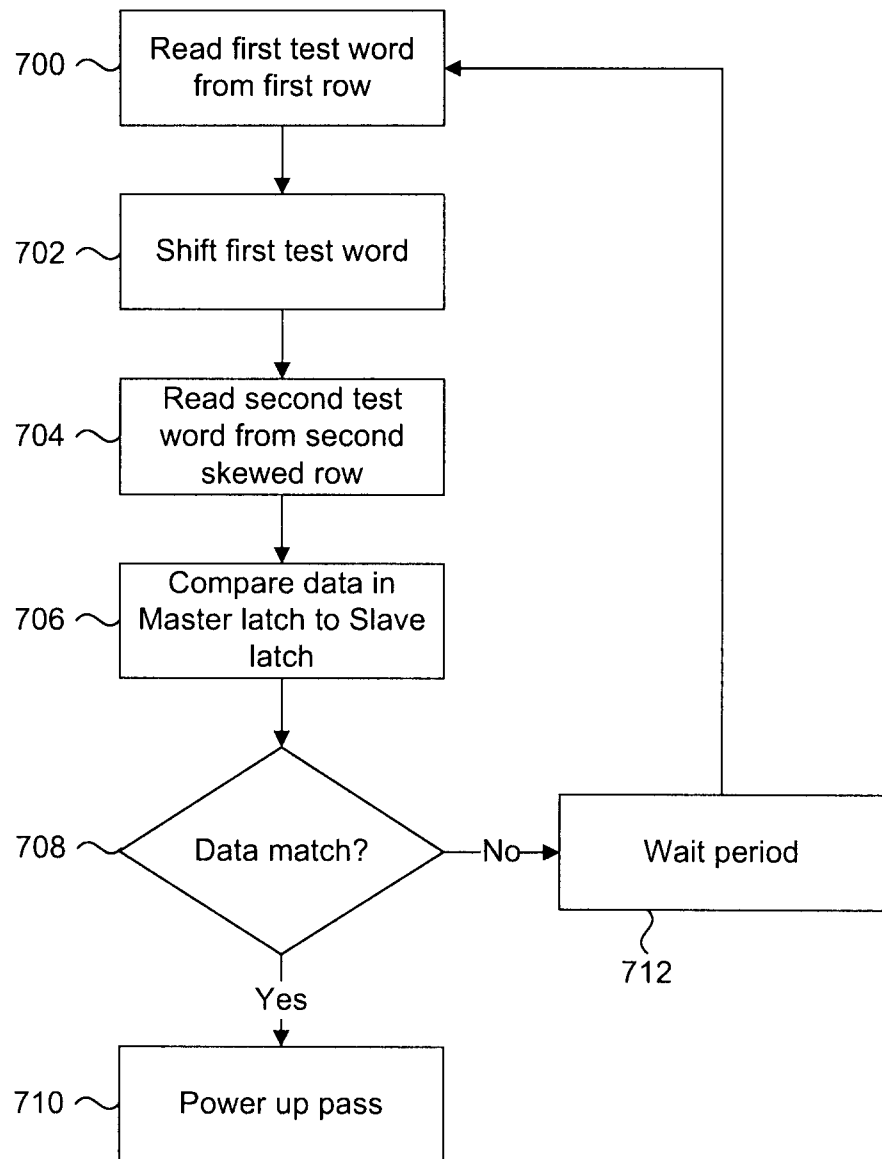
Figure 23

19/20
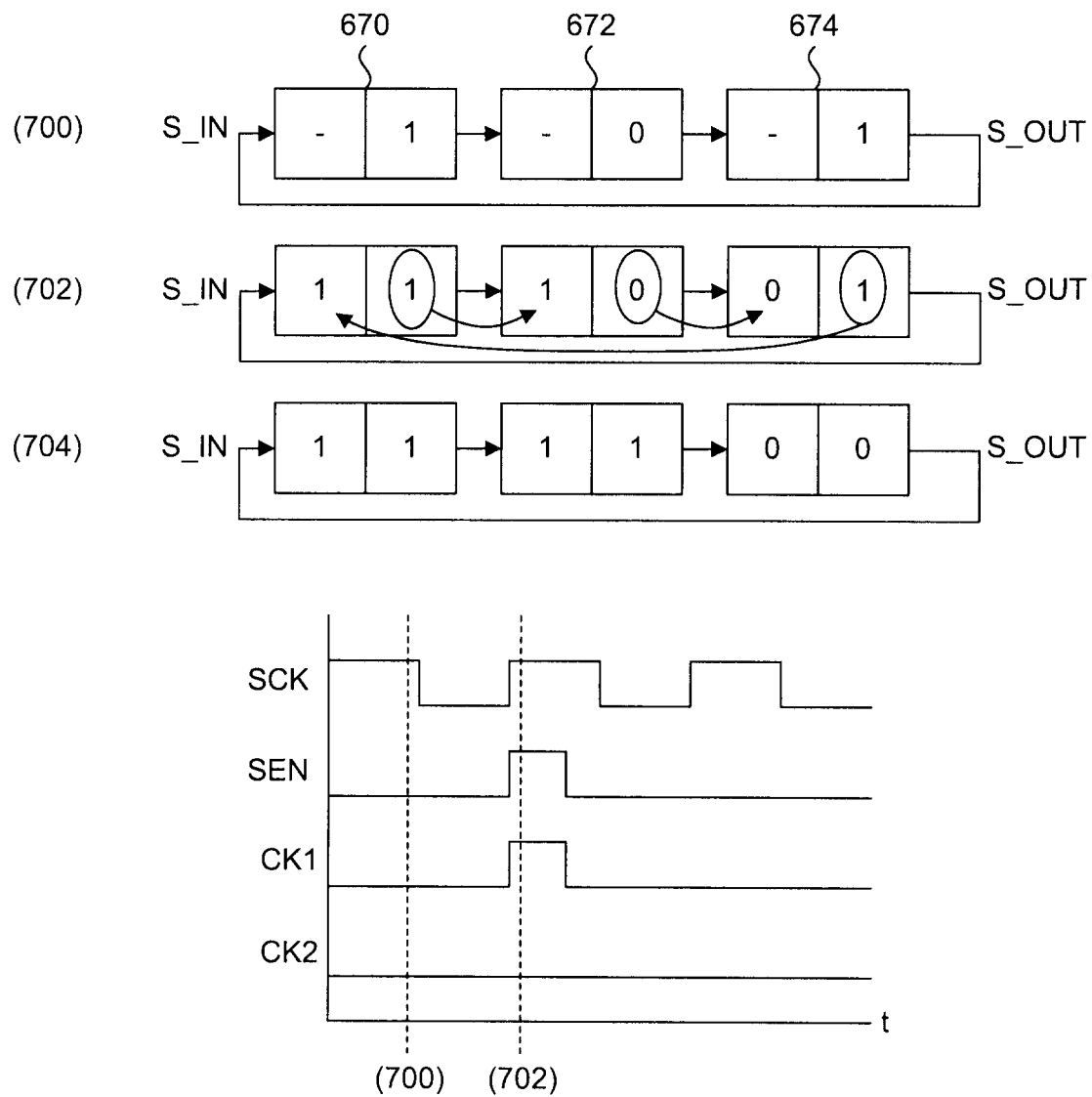
Figure 24
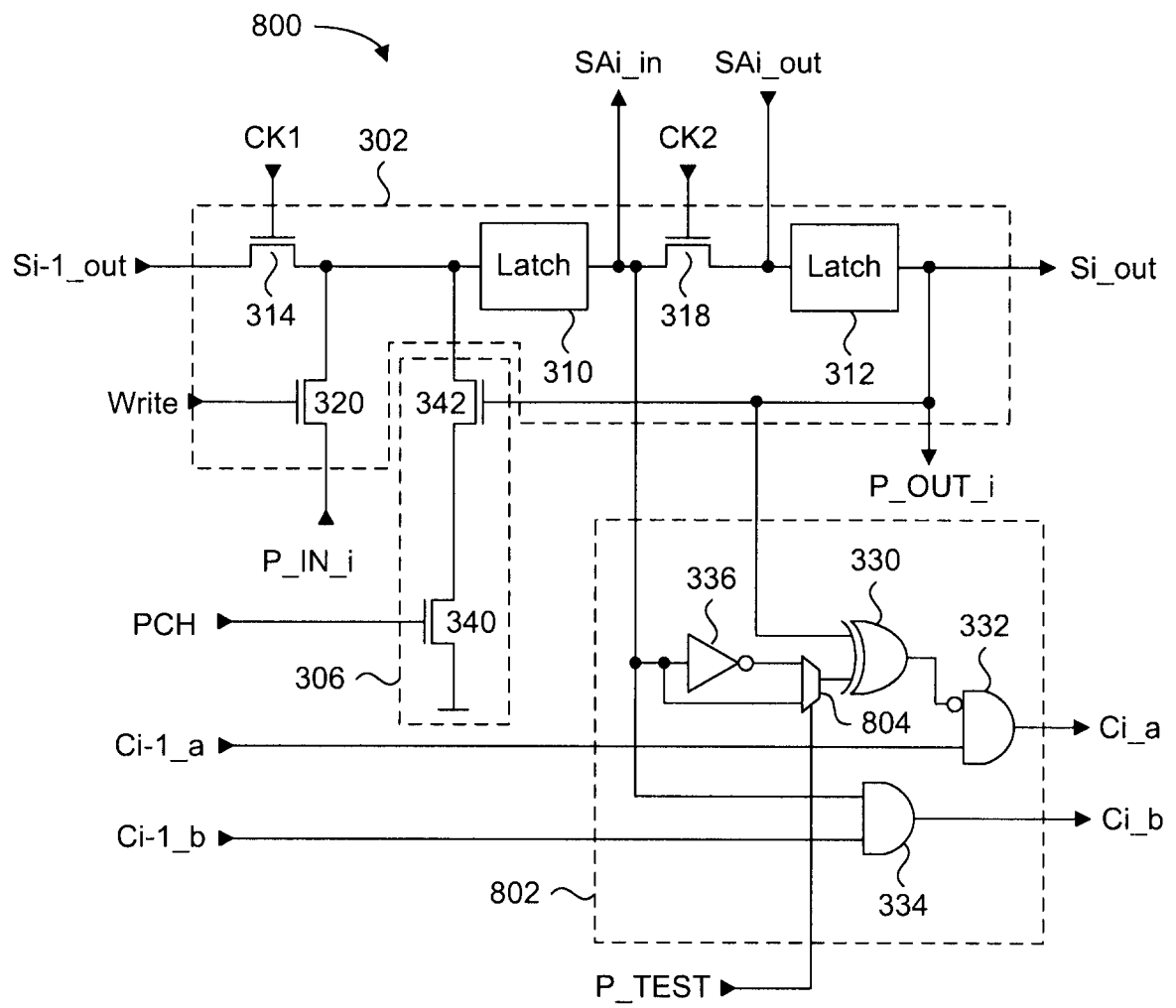
Figure 25

A.   CLASSIFICATION OF SUBJECT MATTER
IPC: *G11C 29/52* (2006.01) , *G06F 11/00* (2006.01) , *G11C 17/14* (2006.01) , *G11C 7/12* (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
        IPC: G11C

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

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Databases: WEST, Delphion, and Espacenet
Search terms used: OTP, verify, latch, slave, master, flash, memory

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/0195653 A1, 8 September 2005, Conley et al. *see abstract; paragraphs 0006, 0009, 0010, 0051-0056; and claim 1	1-10
X	US 5,784,317, 21 July 1998, Ha. *see column 4, lines 39-column 5, line 8; and claim 1	1
A	US 6,266,273 B1, 24 July 2001, Conley et al. *see entire document	1-10
A	US 2002/0006676 A1, 17 January 2002, Debenham et al. *see entire document	1-10
X	US 5,257,223, 26 October 1993, Dervisoglu *see column 2, lines 14-24; claim 1	11-20

[ ]   Further documents are listed in the continuation of Box C.   [X]   See patent family annex.

*	Special categories of cited documents :	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier application or patent but published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means	"&"	document member of the same patent family
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
15 April 2008 (15-04-2008)	17 April 2008 (17-04-2008)

Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001-819-953-2476	Authorized officer Kazem Ziaie 819- 934-2667

| **Box No. II** | **Observations where certain claims were found unsearchable (Continuation of item 2 of the first sheet)** |

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons :

1.   [   ] Claim Nos. :

     because they relate to subject matter not required to be searched by this Authority, namely :

2.   [   ] Claim Nos. :

     because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically :

3.   [   ] Claim Nos. :

     because they are dependant claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

| **Box No. III** | **Observations where unity of invention is lacking (Continuation of item 3 of first sheet)** |

This International Searching Authority found multiple inventions in this international application, as follows :

Group A-Claims 1-10 are directed to a method for executing a program verify operation, comprising: a) loading program data into a first latch of a register stage coupled to at least one bitline of a memory array; b) programming a memory cell coupled to the at least one bitline;c) reading the programmed data of the memory cell coupled to the at least one bitline into a second latch of the register stage;d) comparing logic states stored in the first latch and the second latch; and e) providing a local status signal corresponding to a program pass condition if opposite data states are stored in the first latch and the second latch.

Group B- Claims 11-20 are directed to a method for operating a register stage having a first latch and a second latch arranged in a master-slave flip-flop configuration for storing a first bit of data and a second bit of data respectively, comprising:a. decoupling the first latch from a serial input terminal and decoupling the second latch from the first latch at substantially the same time while a source clock signal oscillates; and b. initiating a shifting operation on either a high logic level of the source clock or a low logic level of the source clock for shifting one of the first bit of data and the second bit of data.

1.   [X] As all required additional search fees were timely paid by the applicant, this international search report covers all

     searchable claims.

2.   [   ] As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite

     payment of additional fees.

3.   [   ] As only some of the required additional search fees were timely paid by the applicant, this international search report

     covers only those claims for which fees were paid, specifically claim Nos. :

4.   [   ] No required additional search fees were timely paid by the applicant. Consequently, this international search report is

     restricted to the invention first mentioned in the claims; it is covered by claim Nos. :

     Remark on Protest   [   ] The additional search fees were accompanied by the applicant's protest and, where applicable,

          the payment of a protest fee.

          [   ] The additional search fees were accompanied by the applicant's protest but the applicable protest

          fee was not paid within the time limit specified in the invitation.

          [   ] No protest accompanied the payment of additional search fees.

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2005195653	08-09-2005	CN1620703 A	25-05-2005
		EP1488429 A2	22-12-2004
		JP2005507129T T	10-03-2005
		JP2007257831 A	04-10-2007
		TW241590B B	11-10-2005
		US6349056 B1	19-02-2002
		US6560143 B2	06-05-2003
		US6972993 B2	06-12-2005
		US7170782 B2	30-01-2007
		US2007109858 A1	17-05-2007
		WO02099806 A2	12-12-2002
US5784317	21-07-1998	GB2308701 A	02-07-1997
		KR100208433B B1	15-07-1999
US6266273	24-07-2001	AT368926T T	15-08-2007
		AU8340901 A	04-03-2002
		CN1329924C C	01-08-2007
		CN101067969 A	07-11-2007
		DE60129710D D1	13-09-2007
		EP1312095 A2	21-05-2003
		JP2004507007T T	04-03-2004
		TW511087B B	21-11-2002
		WO0217330 A2	28-02-2002
US2002006676	17-01-2002	US5895962 A	20-04-1999
		US6190972 B1	20-02-2001
		US6194738 B1	27-02-2001
		US6365421 B2	02-04-2002
US5257223A	26-10-1993	GB 2261562A	19-05-1993
		GB 2261562B	21-06-1995
		GB 9222585D0	09-12-1992
		JP 3359668B2	24-12-2002
		JP 5217393A	27-08-1993
		KR 100257415B1	15-05-2000