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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 November 2006 (09.11.2006)

PCT

(10) International Publication Number
WO 2006/117751 A2

(51) International Patent Classification:
G06F 13/42 (2006.01)

(21) International Application Number:
PCT/IB2006/051364

(22) International Filing Date: 1 May 2006 (01.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/676,164 29 April 2005 (29.04.2005) US

(71) Applicant *(for all designated States except US)*: **KONINKLIJKE PHILIPS ELECTRONICS, N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **DESHPANDE, Amrita** [US/US]; 1109 Mckay Drive, MS41-SJ, San Jose, CA 95131 (US). **ANDERSON, Alma** [US/US]; 1109 Mckay Drive, MS41-SJ, San Jose, CA 95131 (US). **IRAZABAL, Jean-Marc** [US/US]; 1109 Mckay Drive, MS41-SJ, San Jose, CA 95131 (US). **BLOZIS, Stephen** [US/US]; 1109 Mckay Drive, MS41-SJ, San Jose, CA 95131 (US). **BOOGAARDS, Paul** [US/US]; 1109 Mckay Drive, MS41-SJ, San Jose, CA 95131 (US).

(74) Common Representative: **KONINKLIJKE PHILIPS ELECTRONICS, N.V.**; 1109 McKay Drive, MS41-SJ, c/o Daniel L. Michalek, San Jose, CA 95131 (US).

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

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: PROGRAMMING PARALLEL I2C SLAVE DEVICES FROM A SINGLE I2C DATA STREAM

(57) Abstract: Consistent with one example embodiment, communications systems (100,300), using a serial data transfer bus having a serial data line (110) and a clock line (120) used to implement a communications protocol, incorporate programming of parallel slave devices (320,330,340,350) concurrently using an I2C serial bus. At least two slave devices are coupled in parallel on the data transfer bus and configured to load serial data over the serial data line using the communications protocol. Each slave device includes a programmable configuration register configured to be programmed, using the communications protocol, to select one of a plurality of selectable slave device configurations. One of the selectable slave device configurations causes the at least two slave devices to load the serial data in parallel, and another of the selectable slave device configurations causes the at least two slave devices to be loaded one at a time.

WO 2006/117751 A2

PROGRAMMING PARALLEL I2C SLAVE DEVICES FROM A SINGLE I2C DATA STREAM

The present invention is directed generally to communication devices and methodologies, and more particularly, to methods and arrangements for programming parallel slave devices concurrently using an I2C serial bus.

The Inter-Integrated Circuit (I2C) bus developed by Philips Corporation allows integrated circuits to communicate directly with each other via a simple bi-directional 2-wire bus (plus power and ground). A device connects to each of the two wires on the bus, one serial data line (SDA) for the communication of data, and the other serial clock line (SCL) for the control and synchronization of the communication of data between the devices. Each device is connected in parallel to each of the other devices, and each of the bus lines, SDA and SCL, function as a wired-AND of all the lines on the bus. The output of each device is configured as an open-collector/open-drain device, and one or more pull-up resistors maintain a `soft` logic high value on the bus while the bus is in the quiescent state. When a device desires access to the bus, the device pulls the bus to a logic low value, via the open-collector/open-drain device that is placed in a conductive state to ground potential.

Each device that is connected to an I2C bus is identifiable by an address, and can operate as either a transmitter or a receiver, or both. Data transfers are effected using a master-slave communications protocol. A master is a device that initiates a data transfer and generates the clock signals to permit the transfer; any device that is addressed is considered a slave for this transfer. The data transfer can be initiated by a master to either transmit data to the slave (herein designated as write), or to request data from the slave (herein designated as read). For example, an output device, such as a display screen, is typically not able to initiate a data transfer, and therefore would be configured to only operate as a slave device. A microprocessor, on the other hand, will typically be configured to operate as either a master or a slave, as the situation demands.

In a quiescent state, both the SDA and SCL bus lines are in the logic-high state (herein designated as high, or logic state of 1). A master initiates a data transfer by asserting a transition to a logic-low state (herein designated as low, or logic state of 0) on the SDA line while the SCL line is high; this is termed a START condition. Thereafter, the master toggles the SCL line to control the synchronization of the data transfer; data value changes

occur on the SDA line when the SCL clock is low, and the state of the SDA line is considered valid only when the SCL clock is high.

Multiple STARTs can be asserted to effect a series of data transfers within the same transfer session. Generally, each data transfer requires an acknowledgement from the addressed recipient of the data transfer. To terminate the data transfer, the host asserts a low-to-high transition on the SDA line while the SCL clock is high; this is termed a STOP condition. Thereafter, any device may assume control of the bus as a master by asserting a high-to-low transition on the SDA line, as above. Note that, for ease of reference, the term assert is used herein for effecting, or attempting to effect, the specified logic state. In the example of a transition to a logic-high state, this is typically provided by a release of the bus from a forced pull-down state by the asserting device. This assertion of a logic-high state allows the aforementioned pull-up devices on the bus to bring the bus to a logic-high state, unless another device is also forcing the pull-down state.

The general format of an I2C data transfer involves signals on an SDA line and an SCL line forming the I2C bus. A START condition (S) corresponds to high-to-low transition of the signal on the SDA line while the SCL line is high. After the START, the host transmits an address, nominally seven bits, followed by a read/write-not indicator. After transmitting the address and the direction of data transfer (R/W-), the host releases the SDA line, allowing it to rise to a logic-high level. If a slave device recognizes its address, the slave device transmits an acknowledge signal (ACK) by pulling the bus low. The absence of a low signal when the host releases the SDA line, therefore, indicates a non-acknowledgement (NAK). If the address is acknowledged, via a low at SDA, the transmitting device transmits the data. If the direction of data transfer is a "read" relative to the host, then the slave device is the transmitting device; if the direction is a "write" relative to the host, then the master device is the transmitting device. The transmitting device releases control of the SDA line, and the receiving device acknowledges the receipt of the data by asserting a logic-low value on the SDA line. If the data is acknowledged, the transmitter sends additional data. This process continues until the entirety of the data is communicated, or until a transmitted data item is not-acknowledged. The master can subsequently reassert a START signal, and repeat the process above, or, can assert a STOP signal (P) to terminate this data-transfer session.

The above interface protocol can be implemented in a variety of ways. To minimize the development time for programming or designing an I2C interface, a variety of general-

purpose interface schemes have been published. "Design Of A Behavioral (Register Transfer Level, RTL) Model Of The Inter-Integrated Circuit Or I2C-Bus Master-Slave Interface", Master's Thesis of Amrita Deshpande, University of New Mexico, 1999, discloses an I2C master interface and slave interface that is intended to be embodied in an I2C device, and is incorporated by reference herein. By providing a verified I2C interface, system designers need not address the details of the I2C specification and protocol. Both the master and the slave interfaces of this thesis are state-machine based. State-machine based systems and methods are further described in US Patent Number 6,799,233, which is hereby incorporated herein by reference.

Various aspects of the present invention are directed to methods and arrangements for programming parallel slave devices concurrently using an I2C serial bus in a manner that addresses and overcomes the above-mentioned issues.

Consistent with one example embodiment, communications systems, using a serial data transfer bus having a serial data line and a clock line used to implement a communications protocol, incorporate programming of parallel slave devices concurrently using an I2C serial bus. At least two slave devices are coupled in parallel on the data transfer bus and configured to load serial data over the serial data line using the communications protocol. Each slave device includes a programmable configuration register configured to be programmed, using the communications protocol, to select one of a plurality of selectable slave device configurations. One of the selectable slave device configurations causes the at least two slave devices to load the serial data in parallel, and another of the selectable slave device configurations causes the at least two slave devices to be loaded one at a time.

Consistent with another example embodiment, the present invention is directed to a method involving programming a configuration register in each slave device, using the communications protocol over the I2C serial data transfer bus, to select from two or more selectable slave device configurations. One of the two or more selectable slave device configurations causes the at least two slave devices to load data provided on the serial data transfer bus in parallel. Another of the two or more selectable slave device configurations causes the at least two slave devices to load data provided on the serial data transfer bus one at a time. Data is loaded into the at least two slave devices based on their selected configuration.

The above summary of the present invention is not intended to describe each embodiment or every implementation of the present invention. Advantages and attainments, together with a more complete understanding of the invention, will become apparent and appreciated by referring to the following detailed description and claims taken in conjunction with the accompanying drawings.

The invention may be more completely understood in consideration of the following detailed description of various embodiments of the invention in connection with the accompanying drawings, in which:

Figure 1 is a block diagram of a data communications system implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention;

Figure 2 is an illustration of a data stream for a data communications system implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention;

Figure 3 is a block diagram of a system implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention; and

Figure 4 is a flow chart of a method for implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention.

While the invention is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the intention is not to limit the invention to the particular embodiments described. On the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the scope of the invention as defined by the appended claims.

The present invention is generally applicable to methods and arrangements for programming parallel slave devices concurrently using an I2C serial bus. The invention has been found to be particularly advantageous for Inter Integrated Circuit (I2C) serial data communications busses, but is also advantageous for other busses and communications protocols, such as system management bus (SMBus) architectures and/or protocols or other serial data communications systems. For purposes of illustration, and not of limitation, the

invention will be described in the context of an I2C bus having a master device controlling communication to a slave device.

Masters control the communication with I2C slaves on the I2C bus architecture. I2C slaves find numerous applications in fields ranging from cell phones, PDAs and SmartPhones to LCD TVs, Medical Equipment, Gaming, and other applications. One particular application of an I2C slave is as a General Purpose Input/Output (GPIO) device. In this type of device, there are a number of multi-function pins that can be used as inputs or outputs. When used as inputs, these pins typically indicate the state of certain signals that are being monitored.

A particular application of an I2C slave is as a General Purpose Input/Output (GPIO). In this type of device, there are a number of multi-function pins that can be used as inputs or outputs. When configured as outputs, these devices can be used in a system for e.g. to drive many light emitting diodes (LEDs). These outputs are typically divided into banks and programmed individually via the I2C bus.

Consistent with one example embodiment, communications systems, using a serial data transfer bus having a serial data line and a clock line used to implement a communications protocol, incorporate programming of parallel slave devices concurrently using an I2C serial bus. At least two slave devices are coupled in parallel on the data transfer bus and configured to load serial data over the serial data line using the communications protocol. Each slave device includes a programmable configuration register configured to be programmed, using the communications protocol, to select one of a plurality of selectable slave device configurations. One of the selectable slave device configurations causes the at least two slave devices to load the serial data in parallel, and another of the selectable slave device configurations causes the at least two slave devices to be loaded one at a time.

Parallel slave devices that load data concurrently may be configured as general purpose Input/Output (GPIO) devices, or other slave devices. The communication system may conform to I2C, SMBus, and/or other serial communication specifications.

Figure 1 is a block diagram of a data communications system 100 implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention. An SDA line 110 and an SCL line 120 are arranged as an I2C data bus 125. A master device 130 and a slave device 140 are attached to the I2C data bus 125. The master device 130 is electrically connected to the I2C data bus

125 using a clock connection 134 and a data connection 132 electrically connected to the SCL line 120 and the SDA line 110 respectively.

Two or more of the slave device 140 are electrically connected to the I2C data bus 125 using a clock connection 144 and a data connection 142 electrically connected to the SCL line 120 and the SDA line 110 respectively. The master device 130 addresses 152 two or more of the slave device 140, and programs 154 the slave device 140 to operate in a particular configuration. Programming 154 of slave devices may be performed, for example, by designating an address within the slave device 140, when observed on the I2C data bus 125, configuring any slave device recognizing the designated address to receive the data following the address. All slave devices recognizing this address will subsequently receive the data placed on the bus in parallel. Due to the I2C communications protocol, the slowest slave device will control the transmission rate, such that the data will be received by all designated parallel slave devices.

The slave device 140 receives 162 the program 154, such as by placing a word in a register that designates the configuration of the slave device 140. The master device 130 transmits 156 serial data over the I2C data bus 125 to the slave device 140, which the slave device 140 receives 164. The slave device then loads 166 the serial data in paralle with any other slave devices recognizing the designated address.

Figure 2 is an illustration of a serial data stream 200 for a data communications system implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention. The serial data stream 200 is illustrated in Figure 2 consistent with an I2C communications protocol. An ALL CALL address 202 follows a START condition 222 transmitted by a master on an I2C bus. The ALL CALL address 202 is followed by a READ/WRITE bit 210, and a subsequent ACKNOWLEDGE signal 212 from all slave devices that recognize the ALL CALL address 202. The READ/WRITE signal 210 is illustrated in Figure 2 as a logic 0, indicating a write from the master to the slave. A first data byte 204, second data byte 206, and a third data byte 208 are transmitted by the master device, each data byte 204, 206, 208 followed by a respective ACKNOWLEDGE 214, 216, 218 from any slave device that recognizes the ALL CALL address 202. In this manner, all slave devices that recognize the ALL CALL address 202 are programmed in parallel with the data bytes 204, 206, 208.

Figure 3 is a block diagram of a system 300 implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the

present invention. Slave devices 320, 330, 340 and 350 are connected to an I2C bus 310. The slave devices 320, 330, 340 and 350 are illustrated in Figure 3 as GPIO devices for purposes of illustration, but not limitation. The slave devices 320, 330, 340 and 350 may include a configuration register 325, 335, 345, and 355 respectively to direct and/or control the assembly and flow of data to and/or from the I2C bus 310.

In one specific example, slave devices 320, 330, and 340 may be programmed to recognize a particular ALL CALL address, and slave device 350 may not be programmed to recognize the particular ALL CALL address. If the particular ALL CALL address is placed on the I2C bus 310, the slave devices 320, 330, and 340 will load the data following the particular ALL CALL address, whereas the slave device 350 will not load the data following the particular ALL CALL address.

Figure 4 is a flow chart of a method 400 for implementing programming of parallel slave devices concurrently using an I2C serial bus in accordance with embodiments of the present invention. The method 400 involves a group of slave devices, such as a GPIO devices, updating, either one at a time, or in parallel, based on the slave device configuration selected by a programmable configuration register.

The method involves programming 410 the slave device for a particular configuration, such as by using a configuration register in the slave device or such as by writing a particular address recognized by the slave device, for example. The programming may be performed using an I2C communications protocol, to designate whether the slave device updates sequentially or updates in parallel relative to other slave devices on the I2C bus. Serial data is received 420 using a serial data transfer bus, such as an I2C bus, by every slave device on the bus. The selected or configured slave devices load 430 the data, updating either one at a time or in parallel relative to other slave devices, based on the programming 410. The use of a GPIO device as the slave device implementing the method 400 is for purposes of illustration only, and not for limitation.

Hardware, firmware, software or a combination thereof may be used to perform the various embodiments of programming parallel slave devices concurrently using an I2C serial bus as described herein. The master device functionality used in connection with the invention may reside in an I2C master device as described, or may alternatively reside on a stand-alone or networked computer attached to the serial data communications system 100. The serial data communications system 100 illustrated in Figure 1 is an example structure

that can be used in connection with such communications systems, computers, or other computer-implemented devices to carry out operations of the present invention.

The example master device 130 illustrated in Figure 1, suitable for performing the programming in accordance with the present invention, typically includes a central processor (CPU) coupled to random access memory (RAM) and/or some variation of read-only memory (ROM). The ROM may also be other types of storage media to store programs, such as programmable ROM (PROM), erasable PROM (EPROM), etc. The processor may communicate with other internal and external components through input/output (I/O) circuitry and/or other bussing, to provide control signals, communication signals, and the like.

The master device 130 may also include one or more data storage devices, including hard and floppy disk drives, CD-ROM drives, and other hardware capable of reading and/or storing information, such as DVD, etc. In one embodiment, software for carrying out embodiments of programming parallel slave devices concurrently using an I2C serial bus in accordance with the present invention may be stored and distributed on a CD-ROM, diskette, or other form of media capable of portably storing information. These storage media may be inserted into, and read by, devices such as a CD-ROM drive, the disk drive, etc. The software may also be transmitted to the computing arrangement via data signals, such as being downloaded electronically via a network, such as the Internet. Further, as previously described, the software for carrying out the functions associated with the present invention may alternatively be stored in internal memory/storage of the computing device, such as in the ROM.

Any resulting program(s), having computer-readable program code, may be embodied within one or more computer-usable media such as memory devices or transmitting devices, thereby making a computer program product or article of manufacture according to the invention. As such, the terms "computer readable medium," "article of manufacture," "computer program product" or other similar language as used herein are intended to encompass a computer program which exists permanently, temporarily, or transitorily on any computer-usable medium such as on any memory device or in any transmitting device.

Each feature disclosed in this specification (including any accompanying claims, abstract, and drawings), is replacable by alternative features having the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise,

each feature disclosed is one example only of a generic series of equivalent or similar features.

The present invention should not be considered limited to the particular examples described above. Various modifications, equivalent processes, as well as numerous structures to which the present invention may be applicable fall within the scope of the present invention. For example, embodiments in accordance with the present invention can be implemented using a similarly constructed one-way or two-way interface for communication between devices on a common bus, such as an SMBus or other bus arrangement. Such variations may be considered as part of the claimed invention, as fairly set forth in the appended claims.

CLAIMS

What is claimed is:

1. A communications system (100), comprising: a serial data transfer bus (125) comprising: a serial data line; and a clock line; wherein the serial data line and clock line are used to implement a communications protocol; and at least two slave devices (320,330) coupled in parallel on the data transfer bus (310) and configured to load serial data over the serial data line using the communications protocol, each slave device comprising a programmable configuration register (325,335) configured to be programmed, using the communications protocol, to select one of a plurality of selectable slave device configurations; wherein one of the plurality of selectable slave device configurations causes the at least two slave devices to load the serial data in parallel, and another of the plurality of selectable slave device configurations causes the at least two slave devices to be loaded one at a time.

2. The device of claim 1, comprising three or more slave devices coupled in parallel and configured to load the serial data, wherein the programmable configuration register in each slave device is configured to be programmed to individually configure each of the three or more slave devices to load the serial data individually in one configuration, and in parallel relative to other slave devices in another configuration, based on the programmed configuration register.

3. The device of claim 1, wherein slave devices that are loaded in parallel are loaded simultaneously.

4. The device of claim 1, wherein slave devices that are loaded in parallel are loaded concurrently.

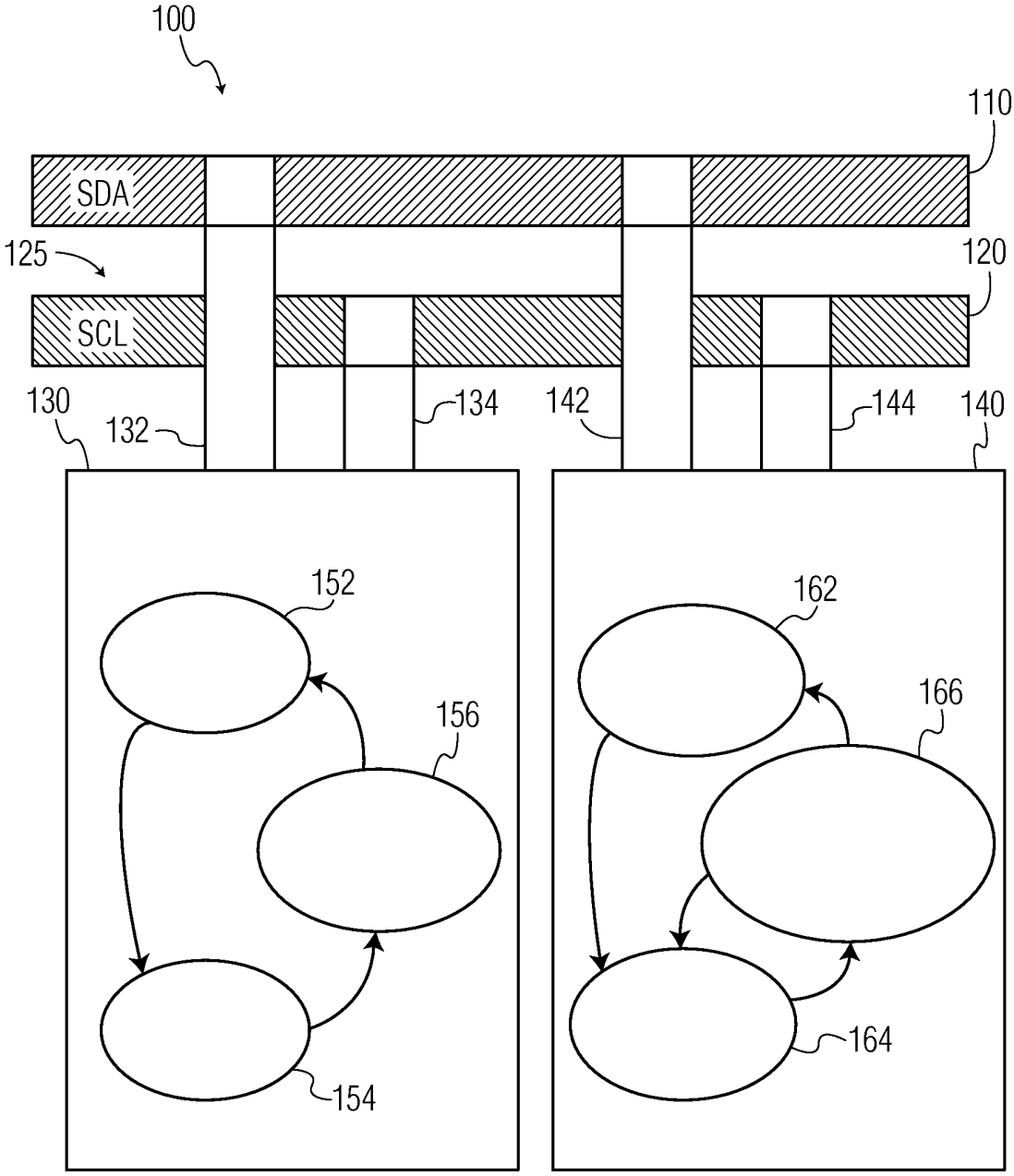
100
110
SDA
125
120
SCL
130
132
134
142
144
140
152
156
154
162
166
164

FIG. 1

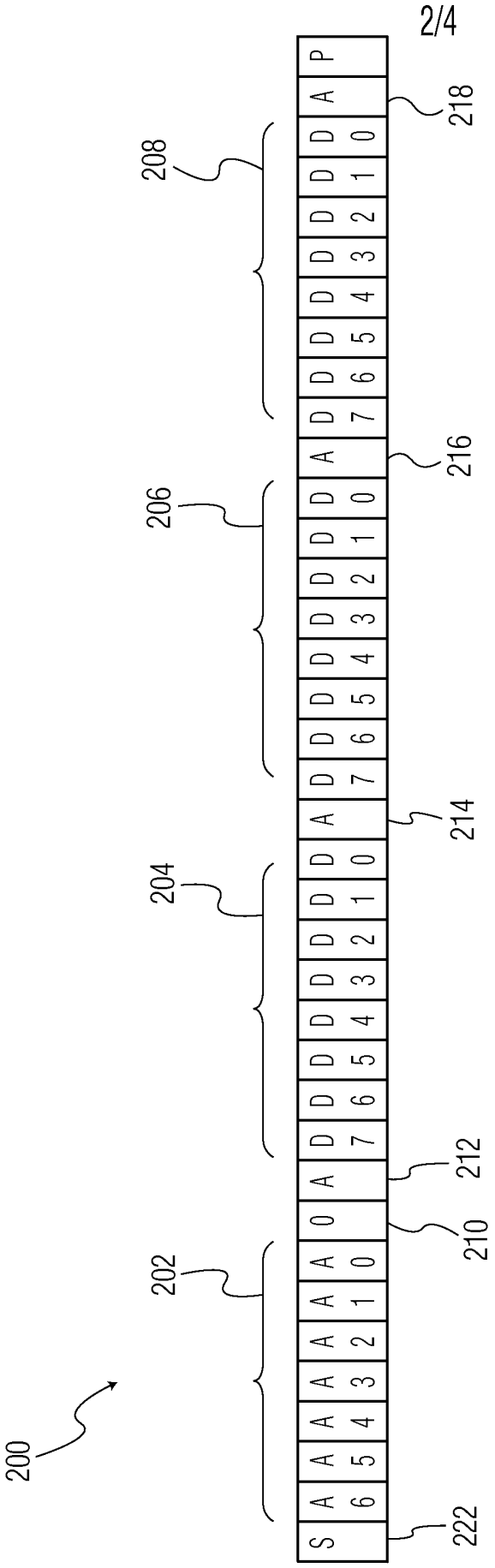
200
S A A A A A A 0 D D D D D D D D A D D D D D D D A P
6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
202
204
206
208
210
212
214
216
218
222

FIG. 2

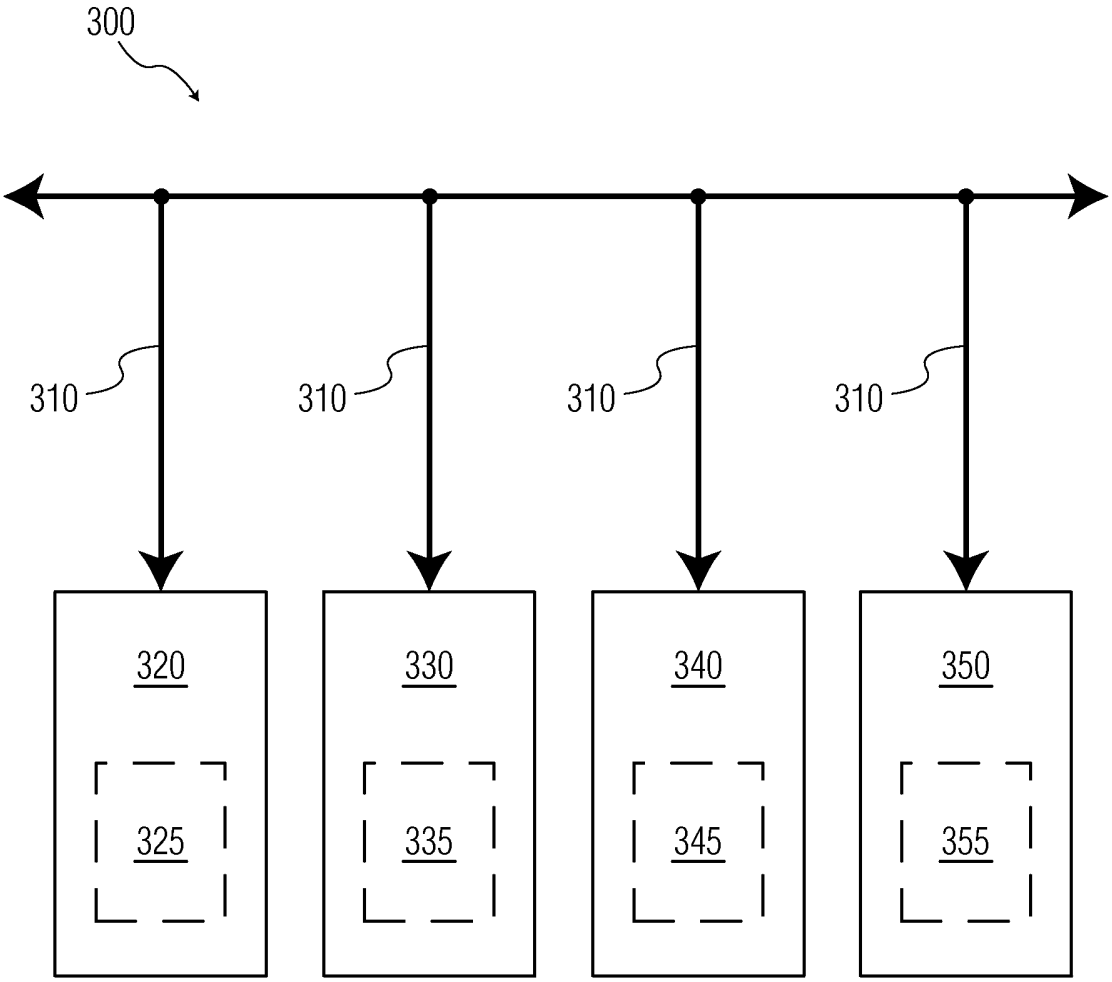

FIG. 3

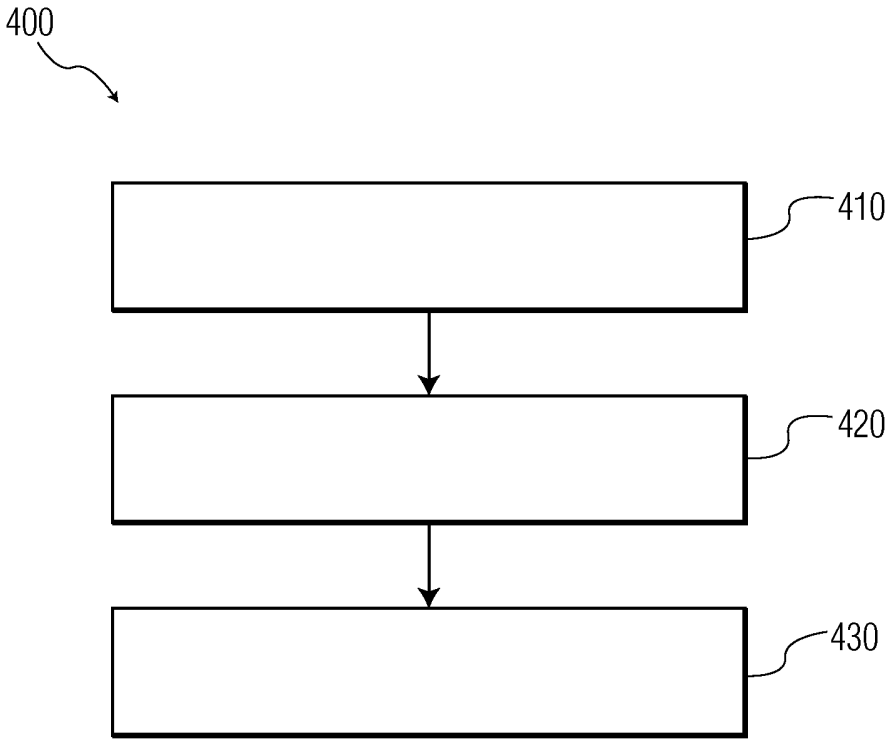

FIG. 4