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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 April 2015 (16.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/054433 A1

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

(21) International Application Number:
PCT/US2014/059776

(22) International Filing Date:
8 October 2014 (08.10.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/888,475	8 October 2013 (08.10.2013)	US
61/927,102	14 January 2014 (14.01.2014)	US
61/974,910	3 April 2014 (03.04.2014)	US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; ATTN: International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventor: **SENGOKU, Shoichiro**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: **LOZA, Julio**; Loza & Loza LLP, 305 North Second Avenue #127, Upland, California 91786 (US).

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

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

Published:

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

(54) Title: COEXISTENCE OF I2C SLAVE DEVICES AND CAMERA CONTROL INTERFACE EXTENSION DEVICES ON A SHARED CONTROL DATA BUS

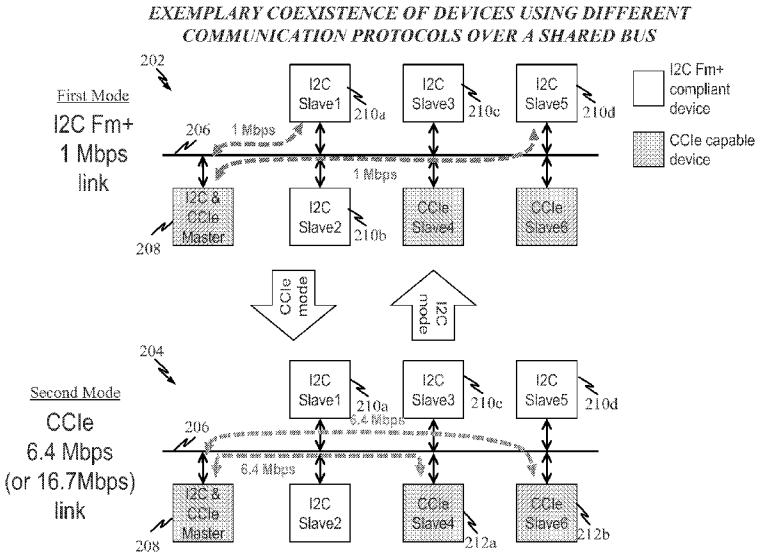


FIG. 2

(57) Abstract: A plurality of slave devices is coupled to a control data bus along with at least one master device that is managing access of slave devices to the control data bus. At least one slave device operates in a sI2C protocol mode of operation and at least one other slave device operates in a CCIe mode of operation. At least the slave devices using sI2C protocol mode use the control data bus for interrupt requests. In order to maintain the integrity of CCIe communications, the slave devices using the sI2C protocol mode disables issuing IRQs when the control data bus operates according to the CCIe mode.

WO 2015/054433 A1

COEXISTENCE OF I2C SLAVE DEVICES AND CAMERA CONTROL INTERFACE EXTENSION DEVICES ON A SHARED CONTROL DATA BUS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present Application for Patent claims priority to Provisional Application No. App. No.: 61/888,475, entitled "Coexistent of I2C Slave Devices and Camera Control Interface Extension Devices on a Shared Control Data Bus" filed October 8, 2013, and to Provisional Application No. App. No.: 61/974,910, entitled "Coexistent of I2C Slave Devices and Camera Control Interface Extension Devices on a Shared Control Data Bus" filed April 3, 2014, and to Provisional Application No. App. No.: 61/927,102, entitled "Camera Control Interface Extension With In-Band Interrupt" filed January 14, 2014, all of which are assigned to the assignee hereof and hereby expressly incorporated by reference herein.

Field

[0002] The present disclosure pertains to enabling multimode operations over a shared bus and, more particularly, enabling devices with different protocols to share a single bus.

Background

[0003] Inter Integrated Circuit (hereinafter "I2C" and also referred to as I^2C) is a multi-master serial single-ended bus used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic devices. The I2C bus includes a clock (SCL) and data (SDA) lines with 7-bit addressing. The bus has two roles for nodes or devices: master device and slave device. A master node/device is a node/device that generates the clock and initiates communication with a slave node/device. A slave node/device is a node that receives the clock and responds when addressed by the master. The I2C bus is a multi-master bus which means any number of master devices can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent). I2C defines basic types of messages, each of which begins with a START and ends with a STOP.

[0004] In this context of a camera implementation, unidirectional transmissions may be used to capture an image from a sensor and transmit such image data to memory in a baseband processor, while control data may be exchanged between the baseband processor and the sensor as well as other peripheral devices. In one example, a Camera Control Interface (CCI) protocol may be used for such control data between the baseband processor and the image sensor (and/or one or more slave devices). In one example, the CCI protocol may be implemented over an I2C serial bus between the image sensor and the baseband processor.

[0005] As technology evolves, there is a need for "hot plug" functionality on an I2C bus. By "hot plug", it is meant that devices such as a slave device may be plugged into an already active bus (i.e., without shutting down the bus). This hot plug functionality is achieved with what is referred to as subset I2C (i.e., sI2C). The sI2C utilizes the control data bus (SCL and SDA together) as the means for slave devices to perform interrupt requests (IRQ). Additionally, because of a desire to increase speed of the CCI protocol, a CCI extension (CCIe) protocol is described herein that increases speed over the CCI protocol. However, having slave devices performing IRQs using the SDA is incompatible with the CCIe protocol which provides a dedicated IRQ line.

[0006] Therefore, a way is needed to increase speed over the CCI protocol and at the same time enable sI2C protocol slave devices to coexist with CCIe devices and be hot pluggable onto the CCIe bus.

SUMMARY

[0007] A device is provided comprising a shared bus, a first slave device, a second slave device, and/or a master device. The first slave device may be coupled to the shared bus and is configured to: (a) operate according to a first protocol mode including issuing in-band interrupt requests over the shared bus; (b) monitor the shared bus for an entry call indicating the shared bus is switching to a second protocol mode; and/or (c) upon detection of the entry call, disable the first slave device from making in-band interrupt requests over the shared bus.

[0008] The second slave device may be coupled to the shared bus and is configured to: (a) operate according to the second protocol mode including issuing side-band interrupt requests over an interrupt bus or in-band interrupt request over the shared bus; and/or (b) detect the entry call over the shared bus, the entry call transmitted according to the second protocol mode.

[0009] The master device may be coupled to the shared bus and configured to: (a) operate according to both the first protocol mode and the second protocol mode; (b) manage communications over the shared bus; (c) send the entry call over the shared bus indicating the shared bus is switching to a second protocol mode, where the entry call is sent according to the first protocol mode; (d) detect interrupt requests from slave devices according to both the first protocol mode and the second protocol mode; and/or (e) respond to the interrupt requests by granting a requesting slave device access to the shared bus.

[0010] The master device may be further configured to send an exit call over the shared bus indicating the shared bus is switching to the first protocol mode, where the exit call is sent according to both the second protocol mode and the first protocol mode.

[0011] For in-band interrupts, the second protocol defines an interrupt period within symbols transmitted over the shared bus during which one or more slave devices coupled to the shared bus can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission. In one implementation, the master device and second slave device may be configured to internally mask the first line of the shared bus during the interrupt period.

[0012] The master device may be configured to send an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

[0013] The second slave device may be further configured to receive data or commands over the shared bus according to the second protocol mode when the shared bus operates according to the second protocol mode.

[0014] The first slave device may be further configured to send an interrupt request over a dedicated interrupt line or bus separate from the shared bus while the shared bus operates according to the second protocol mode.

[0015] The device may also include an interrupt router slave device configured to receive interrupt requests from the first slave device over a dedicated line and send the received interrupt request over the dedicated interrupt line or bus according to the second protocol.

[0016] In one example, the shared bus may include a first line and a second line. When the shared bus operates according to the first protocol mode, the first line is used for data transmissions and the second line is used for a first clock signal. When the

shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

[0017] A multi-mode master device may include a first bus interface, a second bus interface, and a processing circuit. The first bus interface may serve to couple to other devices coupled to a shared control data bus, the data bus dynamically configured to operate in either a first protocol mode or a second protocol mode. The second bus interface may server to couple to a dedicated interrupt line used by at least a subset of the other devices and which issue interrupt requests over the dedicated interrupt line in the second protocol mode. The processing circuit may be configured to: (a) manage data transfers over a shared bus to which a plurality of other devices are coupled, wherein at least a first subset of the other devices operate according to a first protocol mode and a second subset of the other devices operate according to a second protocol mode incompatible with the first protocol mode; (b) dynamically switch operation of the shared bus between the first protocol and second protocol mode by: (1) sending an entry call over the shared bus indicating that the shared bus is to operate according to the second protocol mode, and/or (2) sending an exit call over the shared bus indicating that the shared bus is to operate according to the first protocol mode

[0018] The master device may be further configured to: (a) receive interrupt requests from the first subset of devices over the shared bus when the shared bus operates according to the first protocol mode; (b) receive interrupt requests from the second subset of devices over a dedicated interrupt bus when the shared bus operates according to the second protocol mode, and/or (c) receive interrupt requests from the first subset of devices over the dedicated interrupt bus when the shared bus operates according to the second protocol mode. In one implementation, no interrupt requests are received from the first subset of devices when the shared bus operates according to the second protocol mode.

[0019] The shared bus includes a first line and a second line. When the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal. When the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

[0020] The second protocol may define an interrupt period within symbols transmitted over the shared bus during which the second subset of the other devices can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission.

[0021] The master device may also be configured to internally mask the first line of the shared bus during the interrupt period.

[0022] In one implementation, in response to an in-band interrupt request while operating according to the second protocol mode, the master device may send an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

[0023] A slave device may comprise a bus interface and a processing circuit. The bus interface may serve to couple to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate according to a first protocol mode and the slave device operates according to a second protocol mode. The processing circuit may be configured to:

[0024] (a) detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; (b) send an interrupt request (IRQ) either in-band over the shared bus or side-band over a separate path to the master device; (c) communicate over the shared bus according to the second protocol mode; (d) monitor the shared bus for an exit call from the master device; and/or (c) disable the slave device from communicating over the shared bus upon detection of an exit call from the master device. The exit call may serve to indicate to the slave device that the shared bus is to operate according to the first protocol mode.

[0025] The shared bus may include a first line and a second line. When the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal. When the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

[0026] A slave device may include a bus interface and a processing circuit. The bus interface may serve to couple to a shared bus shared with a plurality of other devices, wherein the slave device operates according to a first protocol mode and at least a first

subset of the other devices operate according to a second protocol mode. The processing circuit may serve to coupled to the bus interface and configured to: (a) detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; and/or (b) disable the slave device from making in-band interrupt requests over the shared bus upon detection of the entry call.

[0027] The processing circuit is further configured to monitor the shared bus for an exit call from the master device, the exit call indicating that the shared bus is to operate according to the first protocol mode. The shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal, and when the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

[0028] An interrupt request router slave device may include a first bus interface, a second bus interface, a third bus interface, and a processing circuit. The first bus interface may serve to couple to a shared control data bus, the data bus dynamically configured to operate in either a first protocol mode or a second protocol mode. The second bus interface may serve to couple to at least one subset of other slave device coupled to the shared bus, the subset of other slave devices operating according to the first protocol mode including in-band interrupt requests over the shared control data bus. The third bus interface may serve to couple to a dedicated interrupt line used in the second protocol mode to issue interrupt request to a master device that manages the shared bus. The processing circuit may be configured to: (a) receive an interrupt request over the second bus interface from a slave device within the first subset of slave devices; and/or (b) route the received interrupt request to the master device via the third bus interface.

DRAWINGS

[0029] Various features, nature, and advantages may become apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout.

[0030] FIG. 1 is a block diagram illustrating a device having a baseband processor and an image sensor and implementing an image data bus and a multi-mode control data bus.

[0031] FIG. 2 is a block diagram illustrating the exemplary coexistence of I2C mode (e.g., legacy CCI or first mode) and CCIe mode (e.g., second mode) over a shared I2C bus (e.g., control data bus).

[0032] FIG. 3 is a block diagram illustrating the coexistence of I2C mode (i.e., legacy CCI) and CCIe mode over a common I2C bus including one or more sI2C slave devices, a multi-mode I2C and CCIe master device, one or more I2C slave devices, and one or more CCIe slave devices.

[0033] FIG. 4 illustrates how a clock may be embedded within data symbols, thereby allowing the use of both I2C wires (i.e., SDA line and SCL line) for data transmissions.

[0034] FIG. 5 is a block diagram illustrating an exemplary method for transcoding of data bits into transcoded symbols at a transmitter to embed a clock signal within the transcoded symbols.

[0035] FIG. 6 illustrates an exemplary conversion between transition numbers and sequential symbols.

[0036] FIG. 7 further illustrates an exemplary conversion between transition numbers and sequential symbols.

[0037] FIG. 8 illustrates a timing diagram of an I2C one byte write data operation.

[0038] FIG. 9 illustrates a timing diagram of a sI2C one byte write data operation wherein the most significant bit (MSB) is one (1), in accordance with the sI2C protocol.

[0039] FIG. 10 illustrates one example of the CCIe protocol.

[0040] FIG. 11 illustrates a diagram of an exemplary in-band interrupt signal (IRQ) sent by a sI2C-compatible slave device over a shared bus.

[0041] FIG. 12 illustrates an additional way of a sI2C slave device to issue an interrupt signal (IRQ) in a mixed protocol mode environment (e.g., where a shared bus can operate according to or dynamically switch between distinct communication protocols).

[0042] FIG. 13 illustrates an exemplary slave device comprising a transmitter and receiver circuit and a processing/logic circuit.

[0043] FIG. 14 illustrates a method for permitting a single-mode slave device to coexist on a shared data bus that switches between multiple communications modes.

[0044] FIG. 15 is a block diagram illustrating a system having a master device and a plurality of slave devices coupled to a shared control data bus and a shared interrupt bus/request.

[0045] FIG. 16 is a block diagram illustrating how the plurality of slave devices in FIG. 15 may be logically grouped for purposes of issuing interrupts.

[0046] FIG. 17 illustrates a first example of how IRQ signals may be arbitrated over the IRQ line or bus.

[0047] FIG. 18 illustrates a second example of how IRQ signals may be arbitrated over the IRQ line or bus.

[0048] FIG. 19 illustrates a shortest IRQ signal length that may be feasible in one example.

[0049] FIG. 20 is a block diagram illustrating an exemplary system in which the interrupt bus has been eliminated in favor of in-band interrupts.

[0050] FIG. 21 illustrates an exemplary timing diagram of an I2C one byte write data operation.

[0051] FIG. 22 illustrates an exemplary CCIe transmission in which data bits have be transcoded into twelve symbols for transmission over the SDA line and the SCL line.

[0052] FIG. 23 illustrates that a slave device should not be allowed to drive the control data bus to assert an IRQ while the master device is driving the control data bus in order to avoid collisions.

[0053] FIG. 24 illustrates a solution to avoid the potential for collisions of FIG. 23.

[0054] FIG. 25 illustrates that when the slave device sends the in-band IRQ on the SDA line, it may cause an erroneous clock to be detected.

[0055] FIG. 26 illustrates a solution to avoid the extra receiver clock pulse and synchronization loss illustrated in FIG. 25.

[0056] FIG. 27 illustrates one approach to implementing an in-band IRQ period while supporting both I2C mode and CCIe mode.

[0057] FIG. 28 illustrates the bit 19 (i.e., the 20^{th} bit when the bit count starts at the first bit being bit 0).

[0058] FIG. 29 illustrates that bit 19 may span the numbers $2221\_2201\_2002_3$ to $2222\_2222\_2222_3$, and that range of numbers may be subdivided into six subdivisions on the left side of FIG. 29.

[0059] FIG. 30 illustrates a range within the bit 19 number space that may be used to define the heartbeat.

[0060] FIG. 31 illustrates one example of how the heartbeat clock may be transmitted over the SDA line and SCL line of a shared control data bus.

[0061] FIG. 32 illustrates another example of a heartbeat clock may be transmitted over the SDA line and SCL line.

[0062] FIG. 33 illustrates one example of transition number to symbol number conversion.

[0063] FIG. 34 further expands on the symbol number to transition number conversion of FIG. 33.

[0064] FIG. 35 illustrates the condition during the in-band IRQ period in which the SDA line is masked.

[0065] FIG. 36 illustrates a side-effect of SDA Mask of FIGS. 32 and 35.

[0066] FIG. 37 illustrates that the heartbeat used for in-band IRQs occupies the number space 0x81BD6~0x81BF0 (i.e., 27 addresses) within the ternary number space.

[0067] FIG. 38 illustrates an example of how bit 19 of the ternary number used in FIG. 37 for CCIe mode transmissions may be mapped.

[0068] FIG. 39 illustrates an alternative technique for implementing an in-band IRQ over the control data bus in CCIe mode.

[0069] FIG. 40 illustrates the condition during the in-band IRQ period of FIG. 39 in which the SDA line is masked.

[0070] FIG. 41 illustrates a side-effect of SDA Mask of FIGS. 39 and 40.

[0071] FIG. 42 illustrates that the heartbeat used for in-band IRQs occupies the number space 0x81BBB~0x81BD5 (i.e., 27 addresses) within the ternary number space.

[0072] FIG. 43 illustrates how heartbeats may be transmitted when the master device is active mode and in a power savings mode.

[0073] FIG. 44 illustrates the combination synchronization word and heartbeat.

[0074] FIG. 45 illustrates the synchronization and heartbeat mapping within bit 19 of the CCIe protocol.

[0075] FIG. 46 illustrates an interrupt group inquiry general call within an exemplary CCIe protocol.

[0076] FIG. 47 illustrates the response to a group inquiry call. In this example, one or more response periods (i.e., inquiry words) may be defined on the SDA line by transferring the heartbeat to the SCL line and using an SDA mask.

[0077] FIG. 48 illustrates an exemplary "terminator word" that may be used to indicate the end of an IRQ group inquiry general call to make the length (i.e., word count) of the IRQ group inquiry general call flexible.

[0078] FIG. 49 illustrates one example of how a DDR clock read general call may be implemented

[0079] FIG. 50 illustrates an exemplary timing diagram for a global clock read word.

[0080] FIG. 51 illustrates the coexistence of I2C-compatible and CCIe-compatible devices on a shared bus within a device, where all master/slave devices use in-band interrupts over the shared bus.

[0081] FIG. 52 illustrates the coexistence of the master/slave devices of FIG. 51 over a shared bus

[0082] FIG. 53 illustrates a method operational by an I2C-compatible slave device that facilitates coexistence with CCIe-compatible slave devices over a shared bus, where all devices can use in-band interrupts over the shared bus.

[0083] FIG. 54 illustrates a method operational by a CCIe-compatible slave device that facilitates coexistence with I2C-compatible slave devices over a shared bus, where all devices can use in-band interrupts over the shared bus.

[0084] FIG. 55 illustrates an exemplary multi-mode master device.

[0085] FIG. 56 illustrates an exemplary slave device.

[0086] FIG. 57 illustrates an exemplary interrupt request router slave device.

DETAILED DESCRIPTION

[0087] In the following description, specific details are given to provide a thorough understanding of the embodiments. However, it will be understood by one of ordinary skill in the art that the embodiments may be practiced without these specific details. For example, circuits may be shown in block diagrams in order not to obscure the embodiments in unnecessary detail. In other instances, well-known circuits, structures, and techniques may not be shown in detail in order not to obscure the embodiments.

Overview

[0088] A first feature provides a way to concurrently operate devices on a common/shared bus (e.g., a control data bus) according to multiple modes of operation. A plurality of slave devices may be coupled to a control data bus along with at least one

master device that controls access to the control data bus. At least a first slave device may operate in a subset I2C (sI2C) protocol mode of operation and at least a second slave device may operate in a camera control interface extension (CCIe) mode of operation. Note that the use of sI2C mode (e.g., first mode) and CCIe mode (e.g., second mode) are exemplary modes and other operating modes (e.g., for different communication standards). The first slave device (e.g., sI2C-compatible slave device) operating in the sI2C protocol mode (e.g., first mode) may use the control data bus for interrupt requests or IRQs (e.g., referred to as in-band IRQs). In one example, such IRQs may be issued by the first slave device to indicate to the master device that it wishes to use the shared bus. By contrast, the second slave device (CCIe slave device) operates in CCIe mode (e.g., second mode) and uses a dedicated IRQ line for IRQs. In order to maintain the integrity of CCIe communications over the shared bus, the first slave device operating in the sI2C protocol mode disables its IRQ ability (e.g., does not issue in-band IRQs) upon a CCIe communication starting over the shared control data bus. That is, while the shared control data bus is used for CCIe communications, the first slave device operating in the sI2C protocol mode is prevented from sending interrupts over the shared bus.

[0089] Although the first slave device operating in the sI2C protocol mode does not understand CCIe communications, entry into the CCIe mode is initiated by the master device operating in the CCIe mode by sending an I2C protocol message over the shared control data bus announcing entry into the CCIe mode. The first slave device, operating in the sI2C protocol mode, understands this CCIe entry announcement and may desist from using the shared control data bus. Additionally, after sending the CCIe entry announcement over the shared control data bus (i.e., a message in the I2C protocol, which the first slave device operating in the sI2C protocol mode understands), the master device operating in the CCIe mode sends an exit/stop message (e.g., in the CCIe protocol and the I2C protocol) announcing an exit or stop from using the shared bus for CCIe communications. The first slave device operating in the sI2C protocol mode disables its own ability to send IRQs over the shared bus while the shared bus is being used for CCIe mode communications.

[0090] From the perspective of the first slave device operating in the sI2C protocol mode (e.g., the "sI2C slave device"), the first slave device receives an understandable I2C protocol message announcing entry into the CCIe mode and immediately disables itself from sending in-band IRQs over the shared control data bus. The first slave

device may then receive one or more CCIe communications which the first slave device (which operates in in the sI2C protocol mode) does not understand and/or ignores. Subsequently, the first slave device receives an understandable I2C protocol message announcing an exit or end of the CCIe mode over the shared bus, and the first slave device enables itself to send in-band IRQs. This self-disabling and enabling to send in-band IRQs by sI2C slave devices prevents data collision over the shared control data bus from a sI2C slave device performing an in-band IRQ while a slave device in the CCIe mode is communicating in the CCIe protocol on the same shared control data bus.

[0091] A second feature provides for allowing hot plugging (also called hot swapping) of a sI2C slave device onto a CCIe-enabled bus (i.e., the shared control data bus). More generally, this second feature permits a slave device operating according to a first protocol mode (e.g., sI2C compatible communication protocol) to be dynamically plugged or added onto a bus operating according to a different second protocol mode (e.g., CCIe mode). By enabling sI2C slave devices to coexist on a CCIe enabled shared control data bus along with CCIe devices, the hot plug functionality is implemented for devices operating on a CCIe enabled shared bus.

[0092] A third feature provides adding an additional way for a sI2C slave device to issue an interrupt (IRQ). In addition to the sI2C slave device using the shared bus (e.g., control data bus) to issue in-band interrupts (IRQs), a separate line (e.g., dedicated IRQ bus or line for sI2C devices) may be utilized to couple the sI2C slave device to an IRQ router CCIe-compatible slave device. The IRQ router CCIe-compatible slave device receives the sI2C protocol interrupts (IRQs) and routes the sI2C protocol IRQ to a CCIe IRQ bus or line. In one example, the sI2C slave device may monitor the shared bus for an entry into the CCIe mode. However, instead of disabling the ability of the sI2C slave device to perform an IRQ, internal logic switches the output of the IRQ from the shared bus to the line coupling the sI2C slave device to the IRQ router CCIe-compatible slave. Subsequently, upon sensing an exit signal while the shared bus is operating according to CCIe mode, the sI2C slave device may again start using the shared bus for issuing in-band IRQs. Accordingly, both sI2C slave devices and slave devices in the CCIe mode can coexist on a CCIe enabled bus without data collisions. The IRQ router CCIe-compatible slave device routes all IRQs received from sI2C slave devices to a dedicated IRQ line or bus operating in the CCIe mode.

[0093] A fourth feature provides for sI2C-compatible devices and CCIe-devices to coexist over the same shared bus while both types of devices use in-band interrupt requests (IRQs).

Exemplary Coexistence of Devices Using Different Communication Protocols Over A Shared Bus

[0094] FIG. 1 is a block diagram illustrating a device 102 having a baseband processor 104 and an image sensor 106 and implementing an image data bus 116 and a multi-mode control data bus 108. While FIG. 1 illustrates the multi-mode control data bus 108 within a camera device, it should be clear that this control data bus 108 may be implemented in various different devices and/or systems. Image data may be sent from the image sensor 106 to the baseband processor 104 over an image data bus 116 (e.g., a high speed differential DPHY link). In one example, the control data bus 108 may be an I2C bus comprising two wires, a clock line (SCL) and a serial data line (SDA). The clock line SCL may be used to synchronize all data transfers over the I2C bus (control data bus 108). The data line SDA and clock line SCL are coupled to all devices 112, 114, 118, 122, and 124 on the I2C bus (control data bus 108). Some or all of the devices 112, 114, and 118 may also be coupled with a one line interrupt (IRQ) bus 120. The slave devices 114, 118, 122, and 124 may have different protocol modes (e.g., may operate using different protocols). For example, a first slave device 122 may be a sI2C-compatible slave device (e.g., communicates according to the sI2C protocol), a second slave device 124 may be a CCIe-compatible slave device (e.g., communicates according to the CCIe protocol), and a third slave device 118 may be capable of multi-mode operation (e.g., communicates according to the I2C/sI2C and CCIe protocols). In one example, control data may be exchanged between the baseband processor 104 and the image sensor 106 as well as the slave devices 118, 122, 124 via the shared control data bus 108. The standard clock (SCL) speed for I2C communications/signaling is up to 100KHz. The standard clock SCL speed in I2C fast mode is up to 400KHz, and in I2C fast mode plus (Fm+) it is up to 1 MHz. These operating modes over an I2C bus may be referred to as a camera control interface (CCI) mode when used for camera applications.

[0095] According to one aspect, an improved mode of operation (i.e., greater than 1 MHz) may be implemented over the multi-mode control data bus 108 to support camera operation. This improved mode of operation over an I2C bus may be referred to as a

camera control interface extension (CCIe) mode when used for camera applications. In this example, the baseband processor 104 includes a master device/node 112 and the image sensor 106 includes a slave device/node 114, both the master device 112 and slave device 114 may operate according to the camera control interface extension (CCIe) mode over the control data bus 108 without affecting the proper operation of other legacy I2C devices coupled to the control data bus 108. According to one aspect, this improved mode of operation over the control data bus 108 may be implemented without any bridge device between CCIe devices and any legacy I2C slave devices. According to one aspect, legacy I2C devices may operate in a first mode having a first clock, first bus speed, and/or first signal protocol, while CCIe-capable devices may operate in a second mode having a second clock, second bus speed, and/or second protocol. The first clock, first bus speed, and/or first signal protocol may be distinct from the second clock, second bus speed, and/or second protocol. For example, the second clock and/or second bus speed may be faster or have greater speed than the first clock and/or first bus speed, respectively.

[0096] According to one aspect, when all slave devices 118 and 124 are CCIe-capable devices there is no need to switch between the first mode and second mode of operation. That is, all signaling and/or communications over the control data bus 108 may be performed according to the second mode (e.g., at a second clock, second bus speed, and/or a second protocol). For example, because the second mode may provide a greater bit rate than the first rate of the first mode, there is no need to switch back and forth between the first mode and second mode. In fact, because legacy devices compatible with the first mode need not be accommodated, a third mode of operation may be implemented which provides a higher/greater bit rate than the second mode.

[0097] According to another aspect, at least the first slave device 122 is sI2C compatible to allow the first slave device 122 to be hot pluggable. The sI2C slave device 122 disables its own ability to send in-band IRQs over the control data bus 108 whenever any other device on the control data bus 108 sends a CCIe entry message (in I2C protocol format) over the control data bus 108. The IRQ disabled slave device 122 may be totally disabled or partially disabled. When totally disabled, the sI2C slave device 122 sends no in-band IRQs over the control data bus 108. In some implementations, when partially disabled, rather than sending an in-band IRQ over the control data bus 108, the sI2C slave device 122 may send side-band IRQs over a

connection to an IRQ router CCIe slave device that may serve to route side-band IRQs from the sI2C slave device 122 to a dedicated IRQ bus or line.

[0098] FIG. 2 is a block diagram illustrating the exemplary coexistence of I2C mode 202 (e.g., legacy CCI or first mode) and CCIe mode 204 (e.g., second mode) over a shared I2C bus 206 (e.g., control data bus). A master device 208 which manages access to the bus 206 may be capable of operating according to the I2C mode (e.g., first mode) and the CCIe mode (e.g., second mode).

[0099] In I2C mode 202, a CCIe-capable master device 208 may support full CCI or I2C Fm+ capability, while a CCIe-capable slave device 210 may not support full I2C capability. In I2C mode, the CCIe master 208 communicates with I2C slaves 210a, 210b, 210c, 210d on the bus 206 using CCI or I2C Fm+ protocol at a maximum of 1Mbps link rate.

[00100] Like the CCI standard operation, CCIe operation 204 only supports a single master device operation (e.g., multiple masters are not supported). In CCIe mode 204 (e.g., the bus 206 is being used for CCIe protocol communications), the CCIe master device 208 communicates with only CCIe-capable slave devices 212a and 212b on the control data bus 206, at either 6.4 Mbps or 16.7 Mbps for example.

[00101] At a start-up, by default, the bus 206 may operate in legacy I2C mode 202 (e.g., the bus 206 is being used for CCIe protocol communications). When CCIe master 208 wants to access to a CCIe slave 212a and/or 212b, it switches from I2C mode 202 to CCIe mode 204 by means of an I2C general call.

[00102] In CCIe mode 204, when CCIe master 208 wants to access the I2C slaves 210a, 210b, 210c, and/or 210d, it switches from CCIe mode 204 back to I2C mode 202 by means of a combination of CCIe "exit" protocol and an I2C general call.

[00103] FIG. 3 is a block diagram illustrating the coexistence of I2C mode 302 (i.e., legacy CCI) and CCIe mode 304 over a common I2C bus 306 including one or more sI2C slave devices 308a and 308b, a multi-mode I2C and CCIe master device 304, one or more I2C slave devices 312a and 312b, and one or more CCIe slave devices 310a and 310b. The various features and operations described with respect to FIG. 2 may be applicable with respect to FIG. 3 and is not repeated in the interest of brevity. In one aspect, the sI2C slave devices 308a and 308b are only capable of sI2C protocol mode operation (wherein most, if not all, I2C communications are understandable by the sI2C slave devices 308a and 308b, and few, if any, CCIe communications are understandable by the sI2C slave devices 308a and 308b). That is, the sI2C slave devices 308a and

308b operate in sI2C mode while CCIe slave devices 310a and 310b operate in CCIe mode. The sI2C slave devices 308a and 308b disable their own ability to send IRQs over the control data bus 306 whenever any device sends a CCIe entry message (in I2C protocol format) over the control data bus 306.

Exemplary Communication Protocol and Encoding Method for Increasing Bit Rates Over Shared Bus

[00104] FIG. 4 illustrates how a clock may be embedded within data symbols, thereby allowing the use of both I2C wires (i.e., SDA line and SCL line) for data transmissions. In one example, this embedding of the clock may be achieved by transition clock transcoding. For instance, the data 404 to be transmitted over the physical link (wires) is transcoded so that it changes state at every symbol cycle of the transmitted symbols 406. Consequently, the original clock 402 is embedded in the change of symbol states at every symbol cycle. A receiver recovers clock information 408 from the state transition at each symbol (in the transmitted symbols 406) and then reverses the transcoding of the transmitted symbols 406 to obtain the original data 410. This allows both wires of the I2C bus (control data bus 108 in FIG. 1, SDA line and SCL line) to be used to send data information. Additionally, the symbol rate can be doubled since it is no longer necessary to have a setup and hold time between clock and data signals.

[00105] FIG. 5 is a block diagram illustrating an exemplary method for transcoding of data bits into transcoded symbols at a transmitter to embed a clock signal within the transcoded symbols. At the transmitter 502, a sequence of data bits 504 are converted into a ternary (base 3) number (i.e., a "transition number"), and the ternary numbers are then converted into (sequential) symbols which are transmitted over the clock line SCL 512 and the data line SDA 514.

[00106] In one example, an original 20 bits of binary data is input into a bit-to-transition number converter block 508 to be converted to a 12-digit ternary number. Each digit of a 12-digit ternary number represents a "transition number". Two consecutive transition numbers may have be the same numbers (i.e., consecutive digits of the ternary number may be the same). Each transition number is converted into a sequential symbol at a transition-to-symbol block 510 such that no two consecutive sequential symbols have the same values. Because a transition is guaranteed at every sequential symbol, such sequential symbol transition may serve to embed a clock signal.

Each sequential symbol 516 is then sent over a two wire physical link (e.g., I2C bus comprising a SCL line 512 and a SDA line 514).

[00107] FIG. 6 illustrates an exemplary conversion between transition numbers 602 and sequential symbols 604. An individual digit of ternary number, base-3 number, also referred to as a transition number, can have one of the three (3) possible digits or states, 0, 1, or 2. While the same digit may appear in two consecutive digits of the ternary number, no two consecutive sequential symbols have the same value. The conversion between a transition number and a sequential symbol guarantees that the sequential symbol always changes (from sequential symbol to sequential symbol) even if consecutive transition numbers are the same.

[00108] FIG. 7 further illustrates an exemplary conversion between transition numbers and sequential symbols. On the transmitter side (TX: T to S) 702, a transition number (T) may be converted to a sequential symbol (S). For instance, a current sequential symbol (Cs) may be obtained based on a previous sequential symbol (Ps) and a temporary transition number (T_{tmp}) that is a function of a current transition number (T). The temporary transition number (T_{tmp}) may be obtained by comparing the current transition number T to zero and when T = zero, the temporary transition number (T_{tmp}) becomes equal to 3, else (when T not equal zero) T_{tmp} becomes equal to T (i.e., T_{tmp} = T = 0 ? 3 : T). The current sequential symbol may be obtained as a sum of the current sequential symbol (C_s) plus the previous sequential symbol (P_s) plus the temporary transition number (T_{tmp}) (i.e., $C_s = P_s + T_{tmp}$).

[00109] On the receiver side (RX: S to T) 704 the conversion operation is reversed to obtain a transition number from a current sequential symbol (Cs) and a previous sequential symbol (Ps). A temporary transition number (T_{tmp}) may be obtained as the sum of the current sequential symbol (Cs) plus 4 minus the previous symbol (Ps) (i.e., $T_{tmp} = C_s + 4 - P_s$). The current transition number (T) is equal to the temporary transition number (T_{tmp}), but the temporary transition number (T_{tmp}) is compared to three (3) and when T_{tmp} = 3, the temporary transition number (T_{tmp}) becomes equal to zero (0), else (when T_{tmp} not equal 3) T becomes equal to T_{tmp} (i.e., T = T_{tmp} = 3 ? 0 : T).

[00110] A table 706 illustrates the conversion between transition numbers and sequential symbols.

[00111] Referring again to FIG. 6, an example of the conversion between transition numbers and sequential symbols is illustrated therein. For example, in a first cycle 606,

the current transition number (Ta) is 2, so T_{tmp} is also 2, and with the previous sequential symbol P_s being 1, the new current sequential symbol C_s is now 3.

[00112] In a second cycle 608, the transition number (Tb) is 1. Since the transition number (Tb) is not equal to zero, the temporary transition number T_{tmp} is equal to the transition number (Tb) value of 1. The current sequential symbol (Cs) is obtained by adding the previous sequential symbol (Ps) value of 3 to the temporary transition number T_{tmp} of 1. Since the result of the addition operation equals 4, which is greater than 3, the rolled over number 0 becomes the current sequential symbol (Cs).

[00113] In a third cycle 610, the current transition number (T) is 1. Because the transition number T is 1, the temporary transition number T_{tmp} is also 1. The current sequential symbol (Cs) is obtained by adding the previous sequential symbol (Ps) value of 0 to the temporary transition number T_{tmp} of 1. Since the result of the addition operation equals 1, which is not greater than 3, the current symbol (Cs) is equal to 1.

[00114] In a fourth cycle 612, current transition number (T) is 0. Because the transition number T is 0, the temporary transition number T_{tmp} is 3.

[00115] The current sequential symbol (Cs) is obtained by adding the previous sequential symbol (Ps) value of 1 to the temporary transition number T_{tmp} of 3. Since the result of the addition operation is 4, which is greater than 3, the rolled over number 0 becomes the current sequential symbol (Cs).

[00116] Note that even if two consecutive ternary digits Tb and Tc have the same numbers, this conversion guarantees that two consecutive sequential symbols have different state values. Because of this, the guaranteed transition in the sequential symbols 604 may serve to embed a clock signal, thereby freeing the clock line SCL in an I2C bus for data transmissions.

[00117] Referring again to FIG. 5, at the receiver 520 the process is reversed to convert the transcoded symbols back to bits and, in the process, a clock signal is extracted from the symbol transition. The receiver 520 receives a sequence of sequential symbols 522 over the two wire physical link (e.g., I2C bus comprising a SCL line 524 and a SDA line 526). The received sequential symbols 522 are input into a clock-data recovery (CDR) block 528 to recover a clock timing and sample the transcoded symbols (S). A symbol-to-transition number converter block 530 then converts the transcoded (sequential) symbols to a transition number, i.e., one ternary digit number. Then, a transition number-to-bits converter 532 converts 12 transition numbers to restore 20 bits of original data from the 12 digit ternary number.

[00118] This technique illustrated herein may be used to increase the link rate of a control bus 108 (FIG. 1) beyond what the I2C standard bus provides and is referred hereto as CCIe mode. In one example, a master node/device and/or a slave node/device coupled to the control data bus 108 may implement transmitters and/or receivers that embed a clock signal within symbol transmissions (as illustrated in FIGS. 4 and 5) in order to achieve higher bit rates over the same control data bus than is possible using a standard I2C bus.

Exemplary Method for Supporting I2C, sI2C, and CCIe Devices On A Shared Bus

[00119] One feature provides for implementing a shared bus that supports both I2C devices and CCIe devices at the same time. As noted in FIGS. 1-7, it is possible to embed a clock signal (SCL line for I2C bus) within symbol transitions, thereby allowing the use of a clock line (SCL line) for data transmissions. This may permit, for example, a CCIe mode of operation with higher bit rates than typical I2C mode buses.

[00120] However, one challenge is to permit the operation of both legacy I2C-compatible devices and CCIe-compatible devices on the same shared bus at the same time. Some details about the requirements of the I2C protocol and CCIe protocol may be useful in formulating a way to support devices operating according to these modes.

[00121] The I2C standard requires that all I2C compatible slave devices must reset their bus logic on receipt of a START condition (e.g., indicated by a high-to-low transition on the SDA line while the SCL line is high on the shared bus).

[00122] FIG. 8 illustrates a timing diagram of an I2C one byte write data operation. The I2C master device sends a 7-bit slave ID in the SDA line 802 to indicate which slave device on the I2C bus the master device wishes to access, then one bit to indicate a write operation. Only the slave device whose ID matches with the 7-bit slave ID can cause intended actions. In order for an I2C slave device to detect its own ID, the master device has to send at least 8-bits on the SDA line (or 8 clock pulse on the SCL line 804). Therefore, this can be exploited when operating the shared bus in CCIe mode to prevent legacy I2C slave devices from reacting to any CCIe operations over the shared bus. In particular, if the CCIe protocol sends less than 7-bits between start (S) indicators, then the legacy I2C slave devices (which expect at least 7-bits) treat this as an incomplete slave identifier (SID) and resets its logic.

[00123] FIG. 9 illustrates a timing diagram of a sI2C one byte write data operation wherein the most significant bit (MSB) is one (1), in accordance with the sI2C protocol.

In this example, the I2C transmission over the SDA line 902 may include a one (1) in the most significant bit of the slave identifier (SID) 904. The MSB being one (1) allows for hot plugging or swapping of sI2C slave devices because a master device can easily know a newly added slave device is a slave device by merely examining the MSB.

[00124] FIG. 10 illustrates one example of the CCIe protocol. The blocks indicate CCIe specific timing and signaling as well as legacy I2C timing and signaling.As illustrated in 1002, there are protocols to enter into CCIe mode, and exit from CCIe mode. Once entry to CCIe mode is executed, the shared bus mode stays in CCIe mode until an exit from CCIe mode is executed. In this example, entry into CCIe mode is accomplished by a CCIe mode entry sequence 1004 as part of a general call during I2C mode. To announce entry into CCIe mode to I2C-compatible devices, the master device operating in CCIe mode may send an I2C general call 1030 over the shared bus announcing entry into the CCIe mode 1032. Similarly, to announce the exit 1006 from CCIe mode the master device may send a CCIe mode exit sequence 1034 as part of a general call. These entry/exit sequences 1030 and 1034 within a general call are used by CCIe-enabled devices to switch from I2C mode to CCIe mode and from CCIe mode to I2C mode, respectively. These general calls 1030 and 1034 also allow I2C-only slave devices to know when the shared bus is switching to CCIe mode and withhold issuing interrupts or otherwise transmitting over the shared bus.

[00125] A write data protocol 1008 can send an arbitrary number of address words 1014 and data words 1016 to a slave node/device identified by a Slave Identifier (SID) 1018. Similarly, a read data protocol 1010 can read a plurality of data words 1022 from a slave node/device identified by a Slave Identifier (SID), while the number of address words 1020 is still arbitrary.

[00126] One last special protocol is the clock data recovery (CDR) calibration 1012, that may be used by the CCIe master device to cause an indicated CCIe device (including the master device itself) to start a sequence to calibrate its clock-data recovery logic to maximize the link rate. For this purpose, the CCIe master device also has to have its own slave ID.

[00127] Any CCIe words may be sent in 12-symbols that carry 19-bits information. Except for the CDR calibration protocol 1012, 16 bits of the 19-bits may be data information while 3 bits of the 19-bits are used for other information such as control information.

[00128] FIG. 11 illustrates a diagram of an exemplary in-band interrupt signal (IRQ) 1102 sent by a sI2C-compatible slave device over a shared bus. The slave device pulls down or grounds 1106 the SDA line 1104 of a shared bus which a master device detects and then releases the SDA line 1104 to the requesting slave device. The slave device is then able to transmit its IRQ signal 1102. Upon termination of the IRQ signal, the master device may acknowledge receipt of the interrupt signal by holding the SDA line 1104 low for a period of time.

[00129] Referring again to FIG. 10, like a I2C-compatible slave device, a sI2C slave device may receive an understandable I2C protocol message 1030 announcing entry into the CCIe mode and immediately disables itself from sending in-band IRQs (i.e., interrupt signals over the shared bus). The sI2C slave device may then receive a CCIe communication 1032 which the sI2C slave device does not understand and ignores. Lastly, the sI2C slave device may receive an understandable I2C protocol message 1034 announcing an exit from the CCIe mode, and the sI2C slave device enables itself for sending in-band IRQs again

[00130] FIG. 12 illustrates an additional way of a sI2C slave device to issue an interrupt signal (IRQ) in a mixed protocol mode environment (e.g., where a shared bus can operate according to or dynamically switch between distinct communication protocols). As previously noted, while a shared bus 1204 is managed by a master device 1208 in CCIe mode, sI2C-compatible slave devices 1210a and 1210b may desist from sending or issuing in-band interrupt signals over the shared bus 1204. However, in this alternative approach, these sI2C-compatible slave devices 1210a and 1210b may be coupled to an IRQ router CCIe slave device 1216 via one or more separate lines 1212 and 1214. When the shared bus 1204 is operating in CCIe mode, the sI2C-compatible slave devices 1210 may be configured to send their interrupt signals to the IRQ router CCIe slave device 1216 instead. The IRQ router CCIe slave device 1216 receives the sI2C protocol interrupt signals and routes the interrupt to a CCIe IRQ bus or line 1206 so that they may be processed by the master device 1208. The master device 1208 may then decide whether to switch the shared bus 1204 to I2C mode and grant the requesting I2C-compatible slave device its interrupt request.

[00131] Consequently, while the shared bus 1204 may be operating in CCIe mode, instead of the sI2C-compatible slave devices 1210 disabling their ability to issue an interrupt signal (e.g., requesting a service or use of the shared bus 1204), internal logic within each sI2C-compatible slave device 1210 may switch the output of the interrupt

signal from the shared bus 1204 to the separate line 1212 or 1214 coupling the sI2C slave devices 1210 to the IRQ router CCIe slave device 1216. Also, upon detecting an exit code from the CCIe mode over the shared bus 1204, the sI2C-compatible slave devices 1210 switch back to using the shared bus 1204 for issuing interrupt signals. Accordingly, the sI2C-compatible slave devices 1210 and the CCIe-compatible slave devices 1218 can coexist while the shared bus 1204 operates in CCIe mode without data collisions.

Exemplary Single-Mode Slave Device Operating Over A Multi-Mode Shared Control Data Bus

[00132] FIG. 13 illustrates an exemplary slave device 1302 comprising a transmitter and receiver circuit 1306 and a processing/logic circuit 1304. The transmitter and receiver circuit 1306 may be coupled to the processing circuit 1304 to transmit data to and from the processing/logic circuit 1304 and one or more buses. The transmitter and receiver circuit 1306 may be coupled, for example, to a multi-mode shared bus 1320 that may include a first line and a second line. In a first mode of operation (e.g., I2C mode), the transmitter and receiver circuit 1306 may be configured to use the first line for data transmissions and the second line for a first clock signal. In a second mode of operation (e.g., CCIe mode), the transmitter and receiver circuit 1306 may be configured to use both the first line and the second line for data transmissions while embedding a second clock signal within symbol transitions of the data transmissions. The multi-mode shared bus may be an I2C-compatible bus and/or a CCIe-compatible bus. In a third mode of operation, the transmitter and receiver circuit 1306 operate similar to the second mode but without concurrent support for first mode-only legacy devices (e.g., I2C-compatible legacy devices).

[00133] The slave device 1302 may coexist with a set of other devices coupled to the multi-mode shared bus 1320 but operates only in the first mode while constantly monitoring at least the first line and/or second line during both the first mode and second mode of operations. In the first mode of operation, the slave device 1302 may transmits data to another device over the first line of the shared bus 1320. In the second mode of operation, the first line and second line of the shared bus 1320 may both transmit data according to the second mode for other devices that support the second mode.

[00134] In various examples, the devices coupled to the shared bus 1320 may support multiple different modes of operation (e.g., distinct communication protocols). For example, a first slave device may be a sI2C-compatible slave device, a second slave device may be a CCIe-compatible slave device, and a third slave device may be capable of multiple modes (e.g., I2C-compatible and CCIe-compatible modes).

[00135] The first mode of operation may implement a first protocol for data transmissions over the shared bus 1320 and the second mode implements a second protocol for data transmissions over the shared bus 1320.

[00136] In one example, the slave device 1302 may be a sI2C-compatible slave device. The processing circuit 1304 may include a shared bus monitoring circuit/module 1310 that, when the shared bus 1320 operates according to the first mode, serves to monitor for communications over the shared bus 1320. When the shared bus 1320 operates according to the second mode, the shared bus monitoring circuit/module 1310 serves to monitor for an exit call/command indicating that the bus is switching back to first mode. An interrupt request generator circuit/module 1312 may serve to generate an interrupt signal. An interrupt request inhibiting circuit/module 1314 may serve to ascertain when the slave device 1302 should withhold from generating and/or issuing an interrupt request. For instance, if the shared bus monitoring circuit/module 1310 detects that the shared bus 1320 is operating according to a second mode, the interrupt request inhibiting circuit/module 1314 may prevent interrupts from being issued over the shared bus 1322.

[00137] The transmitter and receiver circuit 1306 may include a clock recover circuit 1308 (to recover a clock signal from the transmitted data). In an optional/alternative approach, an IRQ switching circuit 1316 may allow the slave device 1302 to issue interrupt request to an interrupt router device 1322 over a separate line when the shared bus 1320 operates in the second mode.

[00138] FIG. 14 illustrates a method for permitting a single-mode slave device to coexist on a shared data bus that switches between multiple communications modes. The slave device may operate in a first protocol mode that includes in-band interrupt requests over the shared data bus 1402. The slave device may then monitor the shared data bus for an entry call indicating the shared data bus is switching to operate according to a second protocol mode that does not allow for in-band interrupt requests over the shared data bus 1404. Upon detecting the entry call, the slave device disables issuing any in-band interrupt requests over the shared data bus 1406. Optionally, the

slave device may switch to sending interrupt requests to an interrupt request router via a separate line 1408. The slave device may monitor the shared data bus for an exit call from the second protocol mode 1410.Upon receiving the exit call, the slave device enables making in-band interrupt requests over the shared data bus 1412.

Coexistent of I2C Slave Devices and Camera Control Interface Extension Devices on a Shared Control Data Bus with In-Band IRQ

[00139] The previous discussion provided for routing IRQs between I2C-compatible slave devices and CCIe-compatible devices using a dedicated interrupt bus. The following discussion provides for I2C-compatible devices and CCIe-compatible devices to both use in-band interrupts over the shared control data bus.

[00140] A first feature provides for eliminating dedicated interrupt lines and pins for all devices coupled a shared bus. Instead, all devices use the shared control data bus to issue interrupt requests, thereby allowing slave devices coupled to the shared control data bus to transmit data over the shared control data bus.

[00141] A second feature provides for defining an interrupt period within the symbols transmitted over the shared control data bus during which one or more slave devices coupled to the bus can assert an interrupt request on a first line of the bus while a second line of the bus is used by the master device for a heartbeat transmission, where such heartbeat transmission serves to synchronize the one or more slave devices.

[00142] A third feature provides for internally masking the first line of the shared bus, at a receiver device during the interrupt period, for purposes of decoding the transcoded data bits received over the shared bus. For instance, the master device and slave devices may mask the first line input to a clock data recovery circuit (CDR) by using a locally (internally) generated mask signal.

[00143] A fourth feature provides for a master device to monitor the first line of the shared control data bus during the interrupt period to ascertain whether a slave device has asserted an interrupt request.

[00144] A fifth feature provides for a master device, upon detecting an interrupt request over the first line of the shared control data bus, to scan the slave devices over the shared control data bus to identify the asserting/requesting slave device.

Interrupt Mechanism Using Dedicated IRQ Line

[00145] FIG. 15 is a block diagram illustrating a system 1502 having a master device 1508 and a plurality of slave devices 1510a-1510e coupled to a shared control data bus 1504 and a shared interrupt bus/request 1506. In one example, the control data bus 1504 may be an I2C bus comprising two wires, a clock line (SCL) and a serial data line (SDA). The clock line SCL may be used to synchronize all data transfers over the I2C bus (control data bus 1504). The data line SDA and clock line SCL are coupled to all devices 1508 and 1510a-1510e on the I2C bus (control data bus 1504). When used in CCIe mode, both the SDA line and SCL line of the control data bus 1504 may be used for data transmissions.

[00146] According to one aspect, the shared interrupt bus 1606 may be a single line coupled to the slave devices 1610a-1610e as well as to the master device 1608. This shared interrupt bus 1506 may be pulled up (e.g., pull high) when not in use and may be pulled low (e.g., grounded) when a slave device asserts an interrupt request (IRQ) signal. That is, each slave device 1510a-1510e may independently request access to transmit on the shared control data bus 1504 by sending an IRQ signal (e.g., request) to the master device 1508.

[00147] In some examples, the single line IRQ bus may be an asynchronous bus (e.g., unmanaged by a master device or any other device). This means that the slave devices can unilaterally assert an IRQ signal at any time.

[00148] In another example, the single line IRQ bus may be dedicated to unidirectional signal transmissions from slave devices to the master device. That is, the single line IRQ bus may be used for only IRQ signals and no other types of signals.

[00149] In one example, the control data bus 1504 may be a camera control interface (CCI) or CCI extension compatible bus.

[00150] In another example, the control data bus 1504 may be a bidirectional bus between the slave devices and the master device.

[00151] FIG. 16 is a block diagram illustrating how the plurality of slave devices 1510a-1510e in FIG. 15 may be logically grouped for purposes of issuing interrupts. In this example, a first plurality of slave devices 1510a and 1510b may be in a first group 1602 and a second plurality of slave devices 1510c and 1510d may be in a second group 1604. Such groupings may be, for example, pre-configured or dynamically defined (e.g., by enumeration) upon boot-up by the master device 1508. Such groupings allow the master device 1508 to more quickly identify which slave device triggered an IRQ signal on the IRQ bus 1506 without unacceptable delays.

[00152] Each group of slave devices 1602 and 1604 may have a distinct IRQ signal. For instance, the first group 1602 may use a first signal having a first period, and the second group 1604 may use a second signal having a second period, and so on. For example, the "period" may be a length of time for which the IRQ bus 1506 is pulled low by the asserting slave device. Note that other forms of signal differentiation may be used, e.g., different voltage levels for the IRQ signals used by different groups of slave devices, etc. In one implementation, each "group" may include a single slave device. In other implementations, each "group" may include 2, 3, and/or 4 slave devices or more. The number of slave devices per group may be a function of how long it would take to query and identify an asserting slave device. For instance, if a large number of slave devices coupled to the IRQ bus 1506 have to be queried by the master device 1508, this may cause an unacceptably long delay. Consequently, grouping slave devices and using distinct IRQ signals for each group allows the master device 1508 to identify an asserting slave device in a relatively short period or an acceptable period of time.

[00153] The master device 1508 detects the occurrence of an IRQ signal on the shared single line IRQ bus 1506 and queries each slave device in the group to identify which slave device triggered or asserted the IRQ signal. For example, if the IRQ signal identifies a Group-2 1604 slave device, then the master device 1508 may send a register status request (via the control data bus 1504) to a first slave device 1510c within Group-2 1604. If the first slave device 1510c status response indicates that it is not the asserting slave device, then the master device 108 may send another register status request (via the control data bus 1504) to a second slave device 1510d within Group-2 1604. This process is repeated for all slave devices in Group-2 1604 until the slave device that asserted the IRQ signal is identified.

[00154] In an alternative approach, the master device 1508 may scan through all slave devices in Group-2 1604 even if the first slave device 1510c is identified the issuer of the IRQ signal. For instance, it is possible that the more than one slave device in the same group issue an IRQ signal at the same time. Consequently, the master device may know all the IRQ requests from devices in group at once and handle them one-by-one. In one implementation, concurrent or overlapping IRQ requests from multiple devices in a single group may be handled by the master device 1608 in order of urgency, importance, and/or priority.

[00155] FIG. 17 illustrates a first example of how IRQ signals may be arbitrated over the IRQ line or bus. In this example, different slave device groups 1702 and 1704 may be defined, with each slave device group 1702 and 1704 having interrupts 1706 and 1708 of different widths. When a slave device detects that the IRQN line 1506 low, it waits for the IRQN line 1506 to become high plus minimum bus free time 1702 before asserting IRQN line low. In this example, a first interrupt 1706 is asserted by a slave device within a first group 1702, followed by a second interrupt request 1708 asserted by another slave device within a second group 1704.

[00156] FIG. 18 illustrates a second example of how IRQ signals may be arbitrated over the IRQ line or bus 1506. When two slave devices assert IRQN line low (to indicate an interrupt request) at the same time (or overlapping times), the slave group with longest IRQN low period wins. In this example, a first interrupt signal 1806 has been asserted/issued by a first slave device in a first group 1802 while, concurrently or contemporaneously (e.g., overlapping in time), a second slave device in a second group 1804 has asserted/issued a second interrupt signal 1808 over the shared interrupt bus 1506. Because the second interrupt signal 1808 is longer than the first interrupt signal 1806, the first slave device in the first group 1802 loses arbitration. That is, the second interrupt signal 1808 is recognized on the interrupt bus 1506 but not the first interrupt signal 1806. The requesting first slave device can detect a loss of arbitration when it releases the interrupt bus 1506 at the end of the first interrupt signal 1806 but the interrupt bus 1506 remains low (e.g., pull to ground) when it should have returned to high (e.g., pulled high). Consequently, the first slave device reissues its interrupt signal 1810 at a later time (e.g., after the interrupt bus 1506 has been pulled up again for an amount of time 1812 upon the second interrupt signal 1808 ending.

[00157] FIG. 19 illustrates a shortest IRQ signal length that may be feasible in one example. This example assumes that, for an interrupt line 1902, a “high” state can be detected when the IRQ signal 1904 is 70% of VDD level (e.g., high state) at latest and a “low” state can be detected when the IRQ signal 1904 is 30% of VDD level at latest. Note that a high or low state maybe detected when the IRQ signal 1902 is anywhere between 30% to 70% of VDD level depending on receiver input levels. Here, TRF max is the maximum fall-rise time and TLOW is the intended low period. The main purpose of the tLOW constraint is so that the master device is able to distinguish between IRQ signals from different groups of slave devices. Additionally, from the master device’s

point of view, the 2TRF max must be less than TLOW (i.e., 2TRF max < TLOW) so that the IRQ signal 1904 is guaranteed to be detected as low state.

[00158] A first IRQ signal length (period) tLOW is greater than TLOW-TFR and less than 2TLOW+TFR (i.e., TLOW-TFR < tLOW < 2TLOW+TFR). Similarly, a second IRQ signal length (period) tLOW' is greater than 2TLOW-TFR and less than 2TLOW+TFR (i.e., 2TLOW-TRF < tLOW < 2TLOW+TFR).

[00159] Note that after a first slave device asserts IRQ signal low, a second slave device may not detect IRQ signal low for a period of time extending TFR max to tLOW min which must at least TFR max long. Therefore, TLOW > 3TFR max, and tLOW min > 2TFR max.

[00160] Side-band IRQ (e.g., over a dedicated IRQ bus/line) has a clear advantage over in-band IRQ (e.g., over the shared control data bus) in terms of interrupt latency. Side-band IRQs may be preferable in some implementations that require very short latencies in the detection of interrupt signals.

In-Band Interrupt Mechanism Over Shared Bus

[00161] The side-band IRQ method illustrated in FIGS. 15 to 19 requires the use of an extra pin for each slave device and master device. Slave devices in particular are often limited in size/space available and it would be desirable to eliminate the use of side-band interrupts or the requirement of a dedicated interrupt line/bus. Thus, an alternative to side-band IRQ method is sending in-band IRQs over the shared control data bus.

[00162] FIG. 20 is a block diagram illustrating an exemplary system 2002 in which the interrupt bus has been eliminated in favor of in-band interrupts. The system 2002 may include a master device 2008 and a plurality of slave devices 2010a-2010e coupled to a shared control data bus 2004. In one example, the control data bus 2004 may be an I2C bus comprising two wires, a clock line (SCL) and a serial data line (SDA). The clock line SCL may be used to synchronize all data transfers over the I2C bus (control data bus 2004). The data line SDA and clock line SCL are coupled to all devices 2008 and 2010a-2010e on the I2C bus (control data bus 2004). Relative to FIGS. 15-16, this system 2002 does not have a separate interrupt line or bus. Instead, interrupts are sent in-band over the shared control data bus 2004.

[00163] One feature provides for implementing a shared control data bus that supports both I2C devices and Camera Control Interface extension (CCIe) devices at the same time (e.g., dynamically switch the shared bus between CCIe mode and I2C mode).

Exemplary CCIe and I2C Transmissions Over Shared Bus

[00164] FIG. 21 illustrates an exemplary timing diagram of an I2C one byte write data operation. In this example, the shared control data bus 2004 (FIG. 20) includes a serial data line SDA 2102 and a serial clock line SCL 2104. The transmission scheme illustrated in FIG. 21 may be referred to as "I2C mode". The SCL line 2104 is used to send a clock from the master device to all slave devices while the SDA line 2102 transmits data bits. An I2C master device sends a 7-bit slave ID 2208 in the SDA line 2102 to indicate which slave device on the I2C bus the master device wishes to access, then one bit to indicate a write operation. Only the slave device whose ID matches with the 7-bit slave ID 2108 can cause intended actions. In order for an I2C slave device to detect its own ID, the master device has to send at least 8-bits on the SDA line (or 8 clock pulses on the SCL line 2104).

[00165] The I2C standard requires that all I2C compatible slave devices reset their bus logic on receipt of a START condition 2106 (e.g., indicated by a high-to-low transition on the SDA line while the SCL line is high).

[00166] The CCIe protocol uses both the SDA line 2102 and the SCL line 2104 for data transmissions while embedding a clock signal within the data transmissions. For example, data bits may be transcoded into a plurality of symbols which are then transmitted over lines. By embedding the clock signal (SCL line for I2C bus in FIG. 21) within symbol transitions, both the SDA line 2102 and SCL line 2104 may be used for data transmission.

[00167] FIG. 22 illustrates an exemplary CCIe transmission in which data bits have be transcoded into twelve symbols for transmission over the SDA line 2102 and the SCL line 2104. The transmission scheme illustrated in FIG. 22 may be referred to as "CCIe mode".

[00168] CCIe mode is source synchronous, driven by push-pull drivers. Whoever sends out data over the shared control data bus also sends out clock information embedded in the data. Consequently, only one device on the control data bus is allowed to drive the bus at any one time.

[00169] In order to support both legacy I2C devices and CCIe devices over the same bus, CCIe mode operations use the same START condition 2200, 2202, 2204, which prevents legacy I2C slave devices from reacting to any CCIe operations (e.g., the Start condition during CCIe mode causes the legacy I2C slave devices to reset). In this example, the START condition 2200, 2202, and 2204 (i.e., indicated by a high to low transition on the SDA line 2102 while the SCL line 2104 is high) is detected before a full slave ID (i.e., a full 7 bits) is transmitted, therefore this is an incomplete slave ID (less than 7 bits). If a master device sends 6 SCL pulses then issues a START condition 2200, 2202, or 2204, then all legacy I2C slave devices reset their bus logic before they recognize the data as an I2C Slave ID. Since the 6-bit sequences (e.g., corresponding to every two symbols) are sent between two START conditions 2200, 2202, and 2204, these 6-bit sequences are not decoded as a valid slave ID by any slave devices. Consequently, legacy I2C slave devices will not act upon the incomplete Slave IDs.

[00170] In this system, the master device controls access to the shared bus. So, any device that wishes to transmit over the control data bus must request such access from the master device, for example, by issuing an interrupt request. Prior art mechanisms for issuing interrupts have relied on dedicated interrupts lines or a dedicated interrupt bus. However, such dedicated interrupt lines or bus means that the devices must include at least one additional pin to accommodate such interrupt line or bus. In order to eliminate the need for such dedicated interrupt pin and lines/bus, a mechanism for in-band interrupts within CCIe is needed.

[00171] The use of in-band interrupts should also avoid bus contention or collisions. For example, as illustrated in FIG. 23, to avoid collisions, a slave device should not be allowed to drive the control data bus (e.g., either SDA line 2102 or SCL line 2104) to assert an IRQ while the master device is driving the control data bus.

[00172] FIG. 24 illustrates a solution to avoid the potential for collisions of FIG. 23. In this approach, the CCIe protocol defines when an in-band IRQ may be issued. As illustrated here, the master device may drive a clock on the SCL 2102 line while a period of time is defined to allow the slave to drive the SDA line 2104 to issue an in-band interrupt. In particular, CCIe-compatible slave devices first check whether the SDA line 2104 is in use (e.g., pulled low) and, only if it is not, does it issue an interrupt request by pulling the SDA line 2104 low.

[00173] Another problem, illustrated in FIG. 25, is that when the slave device sends the in-band IRQ on the SDA line, it may cause an erroneous clock to be detected. That

is, in CCIe mode, symbol transitions are used to generate a receiver clock RXCLK. This means that all receiving devices recover clock timing from state transition of the shared bus. The state transition, the state change of the SDA and SCL lines, has to be timing aligned between SDA and SCL lines. Although the CCIe clock data recovery (CDR) circuit can tolerate some skews between the SDA and SCL lines, skew larger than CDR's tolerance will cause the CDR to generate an extra receiver clock pulse 2502, resulting a synchronization loss to the CCIe word boundary.

[00174] FIG. 26 illustrates a solution to avoid the extra receiver clock pulse and synchronization loss illustrated in FIG. 25. The signal that is used for in-band IRQ has to be masked at the clock data recovery circuit input by each device (e.g., master device and slave devices). For instance, each CDR circuit masks 2602 the SDA line 2104 or SCL line (whichever line is used for in-band IRQ) during in-band IRQ transmissions. If, for example, a master device lets a slave device drive the SDA line 2104 with a particular in-band IRQ protocol 2604, all devices on the control data bus then must mask their SDA line input during that period to prevent erroneous/extra RXCLK pulses from being detected. In one example, each device must gate the SDA line to hold its value into the CDR circuit as 1 (or high) during the in-band IRQ period.

[00175] FIG. 27 illustrates one approach to implementing an in-band IRQ period while supporting both I2C mode and CCIe mode. In this approach, an exit indicator 2702 and 2704 is send in CCIe mode as well as I2C mode to indicate that the shared bus is to switch from CCIe mode to I2C mode. Thereafter, an sI2C slave device may issue an in-band IRQ 2706 while in I2C mode. After the in-band IRQ is issued, the shared bus may revert back to CCIe mode when the master device issues an entry call 2708 while in I2C mode.

Exemplary Embedding/Encoding of Heartbeat Clock Within CCIe Mode

[00176] According to one example, the "heartbeat" may be encoded or embedded within a ternary number space used to encode data bits for transmission over the two-line control data bus.

[00177] Referring to FIG. 5, the transcoding method illustrated includes encoding 20 bits into a ternary number which is then converted into twelve symbols. The use of this encoding provides an extra or spare bit within the 20 bit that may be used to send commands within the CCIe protocol.

[00178] FIG. 28 illustrates the bit 19 (i.e., the 20^{th} bit when the bit count starts at the first bit being bit 0). In other words, as is typical in the computer sciences, counting bit wise begins at zero, and bit 19 is the 20^{th} bit. Here, the bits 0-18 are represented within the ternary number range of $0000\_0000\_0000_3$ to $2221\_2201\_2001_3$. The ternary numbers in the range of $2221\_2201\_2002_3$ to $2222\_2222\_2222_3$ are unused for data transmission. Consequently, the ternary number range $2221\_2201\_2002_3$ to $2222\_2222\_2222_3$ may be used to represent bit 19 (i.e., 20^{th} bit). In other words, $2221\_2201\_2002_3$ ternary is 1000_0000_0000_0000_0000 binary (0x80000 hexadecimal) and $2222\_2222\_2222_3$ ternary (0x81BF0) is the largest 12 digit ternary number possible. It is within the number space of this 20^{th} bit (bit 19) that a heartbeat may be transmitted.

[00179] FIG. 29 illustrates that bit 19 may span the numbers $2221\_2201\_2002_3$ to $2222\_2222\_2222_3$, and that range of numbers may be subdivided into six subdivisions on the left side of FIG. 29. CCIe is a multi-master control data bus architecture and management of the control data bus can be transferred from one master device to another master device. Consequently, a "master bus request" command is available (within subrange $2222\_1121\_0210_3$ to $2222\_2112\_1121_3$ as well as a "master handover" (within subrange $2222\_2220\_0002_3$ to $2222\_2221\_1210_3$).

[00180] FIG. 30 illustrates a range within the bit 19 number space that may be used to define the heartbeat.

FIG. 31 illustrates one example of how the heartbeat clock may be transmitted over the SDA line and SCL line of a shared control data bus. A receiver clock (RXCLK) 3108 may be extracted from state transitions of transmitted symbols over the control data bus (e.g., SDA line 3110 and SCL line 3112). This example may illustrate how the heartbeat clock may appear before transmission or encoding and/or after reception and decoding. As can be perceived, a first portion 3102 of the heartbeat clock is transmitted on the SDA line, while a second portion 3104 of the heartbeat clock may be transmitted on the SCL line. In this manner, a space 3106 is created on the SDA line by moving part of the heartbeat clock to the SCL line.

First Exemplary In-Band IRQ Technique for CCIe Devices

[00181] FIG. 32 illustrates another example of a heartbeat clock may be transmitted over the SDA line and SCL line. In this example, the heartbeat clock includes a first portion 3202 of the heartbeat clock is transmitted on the SDA line, while a second

portion 3204 of the heartbeat clock may be transmitted on the SCL line, thereby creating a larger space 3206 for the in-band IRQ on the SDA line.

[00182] According to the protocol, a receiving slave device may detect, for example, the n^{th} RXCLK 3214 after the start S indicator 3212. The n^{th} RXCLK 3214 may trigger an internal SDA mask 3224 to internally (e.g., within a receiving slave device) mask the SDA line 3226.

[00183] At the n+1 RXCLK 3216, the slave device may trigger an IRQ by pulling the SDA line low. The SDA line is weakly pulled high by the master device, so that when it is pulled low (by a slave device) this serves to indicate an in-band IRQ. That is, by weakly pulling the SDA line high, this allows a slave device to pull the SDA line low to assert an interrupt signal.

[00184] At the n+2 RXCLK 3218, the master device may sample the SDA line to ascertain whether an in-band IRQ has been asserted.

[00185] At the n+3 RXCLK 3220, the slave device may release the SDA line (e.g., de-assert the in-band IRQ).

[00186] Between n+3 and n+4 RXCLK, the master device re-enables the SDA driver and starts driving the SDA line high. That is why the receiver device (e.g., slave device) can safely release SDA mask at n+4 RXCLK.

[00187] At the n+4 RXCLK 3222, the slave device may release the SDA mask 3224.

[00188] In this manner, an IRQ may be transmitted by a slave device during the IRQ period 3506 defined on the SDA line.

[00189] FIG. 33 illustrates one example of transition number to symbol number conversion.

[00190] A symbol S is sent over the control data bus (e.g., SDA line and SCL line) in CCIe mode. In one example, each symbol may be made up of 2 bits, with the LSB assigned to the SCL line and the MSB assigned to the SDA line.

[00191] In one example, each ternary transition number T is defined such that:

T=1 when S transitions from previous state to current state clockwise by one state on the symbol ordering circle;

T=2 when S transitions from previous state to current state clockwise by two states on the symbol ordering circle; and

T=0 when S transitions from previous state to current state clockwise by three states on the symbol ordering circle.

Typical data transmissions over the data control bus in CCIe mode may employ any transition number.

[00192] FIG. 34 further expands on the symbol number to transition number conversion of FIG. 33. The conversion may be defined as:

T=1 when S transitions from previous state to current state clockwise by one state on the symbol ordering circle;

T=2 when S transitions from previous state to current state across the symbol ordering circle;

T=0 when S transitions from previous state to current state counter-clockwise by one states on the symbol ordering circle.

While the SCL line toggle always happens when T=0 or 1, the SCL line is not toggled when T=2.

[00193] FIG. 35 illustrates the condition during the in-band IRQ period in which the SDA line 3502 is masked. Since the SCL line 3504 does not toggle when transition number T = 2 is sent, and the SDA line 3502 is seen as always high regardless of its actual state when the SDA line 3502 is masked 3508, there is no symbol transition. Consequently, therefore is no clock RXCLK 3506 generated if T=2 is sent during the SDA Mask period 3510. For that reason, T=2 is prohibited while SDA Mask=1.

[00194] FIG. 36 illustrates a side-effect of SDA Mask of FIGS. 32 and 35. Even if T is not equal to 2, since the SDA line is always seen as a logic 1 state during the in-band IRQ period, any transition T values that would result in logic 0 for the SDA line is aliased to T[2:0]=010 that assume SDA bit is always 1.

[00195] FIG. 37 illustrates that the heartbeat used for in-band IRQs occupies the number space 0x81BD6~0x81BF0 (i.e., 27 addresses) within the ternary number space. The fact that T=2 is prohibited and any other T combinations are alias to T=010 while SDA Mask=1 means that the heartbeat word that supports in-band IRQ occupies not only one address, but it actually occupies 27 addresses of Bit19 region. The use of this particular heartbeat pattern prohibits use of the ternary number 2222_2222_2222, which is 81BF0 hex, and is very useful as the first word of the two word CCIe synchronization. Without the ternary number 2222_2222_2222, absolute synchronization may not be easy or possible.

[00196] FIG. 38 illustrates an example of how bit 19 of the ternary number used in FIG. 37 for CCIe mode transmissions may be mapped. In this example, the heartbeat may be assigned to the ternary number $2222\_2222\_2010_3$. Note that in this example, so

long as a ternary number in the range of $2222\_2222\_2xxx_3$ is detected, this may be interpreted as a heartbeat and/or in-band IRQ (e.g., 0x81BD9 hex).

Second Exemplary In-Band IRQ Technique for CCIe Devices

[00197] FIG. 39 illustrates an alternative technique for implementing an in-band IRQ over the control data bus in CCIe mode. This example reduces the number of receiver clock RXCLK cycles needed in FIG. 32 to perform in-band IRQ.

[00198] According to this protocol for providing an in-band IRQ period 3906, a receiving slave device may detect, for example, the n^{th} RXCLK 3914 after the start S indicator 3912. The n^{th} RXCLK 3914 may trigger an internal SDA mask 3924 to internally (e.g., within a receiving slave device) mask the SDA line.

[00199] At the n+1 RXCLK 3916, the slave device may trigger an IRQ by pulling the SDA line low. The SDA line is weakly pulled high by the master device, so that when it is pulled low (by a slave device) this serves to indicate an in-band IRQ.

[00200] Rather than waiting until the next clock cycle, between the n+1 RXCLK 3916 but before the n+2 RXCLK 3918, the master device may monitor the SDA line to ascertain if and/or when it goes low, meaning an in-band IRQ request has been asserted. Note that such monitoring of the SDA line by the master device is performed only during the IRQ period to asynchronously detect any IRQ requests from slave devices.

[00201] At the n+2 RXCLK 3918, the slave device may release the SDA line (e.g., de-assert the in-band IRQ).

[00202] Between n+2 and n+3 RXCLK, master device may re-enable SDA driver and starts driving the SDA line high. Consequently, the receiver (asserting slave device) can safely release SDA mask at n+3 RXCLK.

[00203] At the n+3 RXCLK 3920, the slave device may release the SDA mask 3924. In this manner, an in-band IRQ may be transmitted by a slave device during the IRQ period 3906 defined on the SDA line.

[00204] FIG. 40 illustrates the condition during the in-band IRQ period of FIG. 39 in which the SDA line is masked. Since the SCL line 4004 does not toggle when transition number T = 2 is sent, and the SDA line 4002 is seen as always high regardless of its actual state when the SDA line 4002 is masked 4008, there is no symbol transition. Consequently, there is no receiver RXCLK 4006 generated if T=2 is sent during SDA Mask period 4010. For that reason, T=2 is prohibited while SDA Mask=1.

[00205] FIG. 41 illustrates a side-effect of SDA Mask of FIGS. 39 and 40. Even if T is not equal to 2, since SDA line is always seen as logic 1 state during the in-band IRQ period, any transition T values that would result in logic 0 for the SDA line is aliased to T[2:0]=010 that assume SDA bit is always 1.

[00206] FIG. 42 illustrates that the heartbeat used for in-band IRQs occupies the number space 0x81BBB~0x81BD5 (i.e., 27 addresses) within the ternary number space. Similar to the heartbeat in FIGS. 32 and 39, the alternative heartbeat of FIGS. 39-41 also occupies 27 addresses of Bit19 region. However, this alternative heartbeat does not prohibit use of the 2222_2222_2xxx ternary number space, so the 2222_2222_2222 word is still available for synchronization. This heartbeat pattern also requires a master device to use an asynchronous in-band IRQ detection circuit in order to accommodate a shorter in-band IRQ period, which should not be significant sacrifice.

[00207] Ideally, the protocol to implement in-band interrupts should be as compact as one or two CCIe words so that in-band IRQs can be issued as often as possible with as minimum protocol overhead as possible. For example, a periodic in-band IRQ window maybe defined. Among other consideration, the in-band IRQ period should be available even when the bus system is in low-power mode to prevent "starvation" by the slave devices. One solution to this may be to define an in-band IRQ within a CCIe "heartbeat" word which is periodically transmitted by the master device over the shared control data bus to allow synchronization of the slave devices.

[00208] FIG. 43 illustrates how heartbeats may be transmitted when the master device is active mode and in a power savings mode. During normal operations when the master device is in active mode 4302, the master device periodically sends the heartbeat word 4608 in between CCIe frames 4306 in order to allow slaves devices to issue in-band IRQs. The heartbeat word interval may be such that it will not starve slave devices of opportunities to issue an interrupt request.

[00209] When the master device is in power saving mode, the heartbeat 4316 can also be transmitted, thereby allowing the slave devices on the shared bus (e.g., SDA line 4310 and SCL line 4312) an opportunity to issue an in-band IRQ during power saving mode.

[00210] The master device may send this "heartbeat" CCIe word 4316 at a rate that is slow enough for power savings by the master device but fast enough not to starve slave devices of opportunities to issue an IRQ signal. This "heartbeat" CCIe word may serve as an indicator to slave devices that they may issue IRQs. The term "heartbeat" is used

herein to indicate a slow running clock to slave devices during a power savings mode for their minimal functionality to slay active (alive).

[00211] FIG. 44 illustrates the combination synchronization word and heartbeat. The heartbeat word also has a role as the 2nd word of synchronization word sequence. The two word synchronization sequence start with the all-2 word 2222_2222_2222 as denoted as "SY-", and the heartbeat word 2222_2222_1101 as denoted as "-NC".

[00212] "SY-" word causes RX devices to generate 14 transition state 2s including one "2" by START condition and one "2" after the last symbol when the bus state (symbol) turns form 1 (SDA=0, SCL=1) to 3 (SDA=1, SCL=1) .

[00213] "-NC" word causes RX devices to generate 9 transition state 2s including one "2" by START condition.

[00214] The "SY-" word and the "-NC" word are put together or combined (e.g., hence "SYNC"), resulting a total of 23 transition state 2s followed by "1101" sequence, which is a unique sequence and can't occur in any other CCIe transactions. CCIe devices can use the sequence to synchronize to CCIe word boundary.

[00215] FIG. 45 illustrates the synchronization and heartbeat mapping within bit 19 of the CCIe protocol. Numerical space 0x81BC6~0x81BEF hex, spanning 42 numbers, are prohibited in order to make the two word pattern "SY-NC" 23 "2"s and "1101" absolutely unique only for synchronization. The master device can periodically send "SY-" word right before the heartbeat which is "-NC" word, in order to allow slave devices to resynchronize in case of synchronization loss, or to let hot-plugged slaves to synchronize to the bus.

Exemplary IRQ Group Inquiry

[00216] According to one feature, each slave device may be assigned or associated with a "group", each "group" including one or more slave devices. Upon detecting an IRQ signal associated with a particular "group", the master device may send an inquiry (e.g., over the control data bus) to each of the slave devices in the "group" in order to identify the slave device(s) that asserted the IRQ signal (e.g., where the IRQ signal is asserted either in-band IRQ over the control data bus or side-band IRQ over a dedicated IRQ bus). A slave device that asserted the IRQ may respond to the inquiry only within its assigned group, thereby identifying the asserting slave device to the master device. Note that each "group" of slave devices may have one or more slave devices. Note that, when power savings is being implemented by the master device, a slave device may

issue an in-band interrupt within a heartbeat word. Also, multiple slave devices may issue an interrupt within the same heartbeat word.

[00217] FIG. 46 illustrates an interrupt group inquiry general call within an exemplary CCIe protocol. The master device may broadcast a general call IRQ group inquiry 4600 to all slave devices on the shared bus. Following the IRQ group inquiry command 0x0007 hex 4602, a plurality of IRQ group inquiry words 4604 are sent. In one example, each inquiry word has 3 inquiry response slots, and total 33 slots, for group 0 to group 32. In one example, the inquiry words 4604 may include from one (1) to eleven (11) IRQ group inquiry words (IQ) and one terminator word (Term) 4606 at the end. For each inquiry word 4604, all slave devices mask the SDA line of the shared bus.

[00218] At each IRQ group inquiry (IQ) word 4608 of the payload 4604 of the general call, each slave receiver must start masking SDA at T_{11} RXCLK and release the mask at dummy (T_{-1}) RXCLK.

[00219] FIG. 47 illustrates the response to a group inquiry call. In this example, one or more response periods (i.e., inquiry words) may be defined on the SDA line 4708 by transferring the heartbeat to the SCL line 4710 and using an SDA mask 4712. In this example, three separate slots 4702, 4704, and 4706 have been defined for each inquiry word 4608 (FIG. 46). Each of the three time slots 4702, 4704, and 4706 in each IQ word 4608 may be assigned to three different IRQ groups. The slave devices assigned to each slot 4702, 4704, and 4706 can drive the SDA line 4708 during the assigned slot 4702, 4704, and 4706 as an inquiry response to indicate it has issued an IRQ or is has an IRQ that has not been serviced. Since each IRQ group inquiry (IQ) word 4608 has three inquiry response slots 4702, 4704, and 4706, and maximum 11 IRQ group inquiry (IQ) words can be in the general call payload 4604, there can be maximum 33 group slots in one call.

[00220] Up to thirty-two (32) devices may be assigned to groups such that only one device is in one group, thereby providing immediate identification of IRQ issuers. This approach identifies multiple IRQ groups at once, thereby reducing the number of IRQ scans necessary (e.g., fewer IRQ nesting). Alternatively, multiple devices may be assigned to each group, but an additional inquiry may be necessary by the master device to identify which of the plurality of devices in the group issued the IRQ.

[00221] The master device may chose the number of IRQ group inquiry (IQ) words to include in a general call based on number of IRQ groups on the bus system. In some

examples, the master device may send a lesser number of inquiry words 4608 (e.g., less than a maximum number of eleven (11)). This may allow shortening the time for the IRQ group inquiry general call.

[00222] The sequence of the IRQ group inquiry (IQ) words ends with a terminator word (Term) 4606. The symbol pattern of the terminator word 4606 may be chosen so that each receiver slave device can recognize the word is a terminator (Term), not an IRQ group inquiry (IQ) at T11 RXCLK to know when to stop masking the SDA line 4708 and end of the IRQ group inquiry general call.

[00223] The three slots 4702, 4704, and 4706 for the first IRQ group inquiry (IQ) may be assigned to Group 0, 1, and 2. A group with smaller number may be assigned to earlier response slots.

[00224] In one example, Group 0 may be reserved for hot-plugged devices or devices that the master device has not yet recognized on the shared bus. Since at least one IRQ group inquiry (IQ) word must be sent, any hot-plugged device that issued an IRQ is always recognized.

[00225] Thanks to the use of the terminator (Term) word 4606, the length of a payload 4604 can be flexibly set, and the length of the IRQ group inquiry (IQ) word sequence can exceed 11 words if necessary.

[00226] FIG. 48 illustrates an exemplary "terminator word" that may be used to indicate the end of an IRQ group inquiry general call to make the length (i.e., word count) of the IRQ group inquiry general call flexible.

[00227] The IRQ group inquiry method described herein may be applicable to both in-band IRQs and side-band IRQs. In one example, using the IRQ group inquiry method described avoids use of IRQ assertion periods to distinguish between slave device groups. That is, since all slave devices are being scanned, there is no need to distinguish them using interrupt requests having different widths. Consequently, side-band IRQs can now have any arbitrary period. A slave device no longer has to precisely time the IRQ period, so a free running clock may no longer be necessary.

[00228] Additionally, IRQ arbitration may also be eliminated using IRQ group inquiry. Since the master device scans all groups and slave devices therein, the master device can identify multiple IRQ issuers at once, which simplifies slave device logic since it no longer has to address arbitration loss.

[00229] One additional benefit to slave devices of using IRQ group inquiry may be power savings since a slave device no longer has to hold the IRQ line low for a long period of time which may cause DC current from the pull-up resistor to ground.

Exemplary Global Clock Read

[00230] CCIe is a source synchronous symbol transition clocking system. Whoever sends data over the control data bus also sends clock embedded within the data. Unlike I2C, all slaves devices must have their clock source to generate read data with clock information. However, the technique used for IRQ group inquiry, e.g., always toggling SCL line while having all slave devices mask their SDA input and allow slave devices to drive the SDA line, is actually achieving a global clock read.

[00231] As can be appreciated in FIGS. 46 and 47, an IRQ group inquiry word can carry only 3 slave device responses, mainly in order to allow multiple slaves with different RXCLK timings to drive the SDA line within the same time slot. However, allowing only a single slave device to drive the SDA line during an SDA mask permits implementing a double data rate (DDR) global clock read.

[00232] FIG. 49 illustrates one example of how a DDR global clock read general call may be implemented. Before starting a DDR global clock read sequence, the master device issues some CCIe protocol (such as general call) that indicates:

1. Following sequence is DDR global clock read.
2. Number of words of DDR global clock read.
3. SID of the device from which to read data.
4. Register address of the device from which to read data.

All the devices on the bus understand all the CCIe transactions after that call (until a specified number of words are sent) are DDR global clock read.

[00233] In the DDR global clock mode, all the devices on the bus must mask the SDA input to their clock data recovery (CDR) circuit during the symbol period including dummy symbol at the end of a word.

[00234] The addressed slave device drives the SDA line low at the 2nd RXCLK (not including RXCLK by the START condition), and that logic 0 is used by the master device to calibrate its clock to sample the SDA line (SDACLK).

[00235] From the 3rd RXCLK, the addressed slave device can drive out 9-bits of data serially. The 9-bits of data can be MSB first or LSB first or other format depending on

system requirement. During this period, the master device provides or drives a DDR global clock on the SCL line.

[00236] The slave device drives the SDA line high at the 12th RXCLK, and release the SDA line at the 13th RXCLK.

[00237] The master device releases the SDA line after the 1st symbol "3" is sent, and enables its SDA driver and drives the SDA line high after the last symbol "2".

[00238] The master samples in SDA line at SDACLK timing to shift-in the 9 read data bits.

[00239] FIG. 50 illustrates an exemplary timing diagram for a global clock read word. A first clock signal SDACLK 5002 and a second clock signal RXCLK 5004, indicate internal signals within the master device. The second clock signal RXCLK 5004 may be generated by a clock data recovery circuit (CDR) of the master device. The first clock signal SDACLK 5002 may be generated by a clock generation circuit 5006 which may be used by the master device to sample data values from a SDA line 5008 (which is part of the shared bus) when driven by a slave device signal. The first clock signal SDACLK 5002 may be generated only when an SDAMASK signal 5010 is 1 (SDA line 5008 input to CDR is masked).

[00240] Since it is a slave device that drives the SDA line 5008 during a global clock read period, all the devices on the bus (including the master device) must mask the SDA line 5008 input to their CDR during this period, which starts at the second clock signal RXCLK rising 5012 from the START condition and ends at the last RXCLK rising edge 5014 for the word by the dummy symbol.

[00241] In this example, the master device sends 0x5BE75 (2010_1010_10103) for a global clock read word. Since this is the payload portion of the global clock read general call, each device on the shared bus knows that global clock read words follows after the call message "6", each device also knows when to start and end the SDAMASK 5010.

[00242] Each device expects the global clock read word for next word unless next word is the "Terminator" word that has distinct signal pattern at the first symbol.

[00243] Since the data signal on the SDA line 5008 is driven by a slave device using a RXCLK 5016 from the slave device's CDR, the master device must delay "properly" its second clock signal RXCLK 5004 from the master device's CDR in order for the master device to sample the data with enough setup and hold time. The master device learns that "proper delay at the first falling edge of the SDA line 5008 that is driven by the slave device per the global clock read protocol after the master device sent out the

second symbol of the global clock read word (i.e., T10 cycle). The "calibration logic" 5018 measures delay of the SDA line 5008 falling from the beginning of the T10 cycle, and used the delay to configure "SDACLK delay", so that the master device reliable samples the SDA line 5008 transmissions from the slave device from next symbol.

Exemplary Coexistence of sI2C and CCIe Devices Using In-Band IRQs

[00244] FIG. 51 illustrates the coexistence of I2C-compatible and CCIe-compatible devices on a shared bus 5104 within a device 5102, where all master/slave devices use in-band interrupts over the shared bus 5104. In this example, an I2C-compatible and CCIe-compatible master device 5108 may manage communications over the shared bus 5104. The master device 5108 may be configured to operate in both I2C mode and CCIe mode (i.e., according to two or more distinct protocols). An I2C-compatible and CCIe-compatible slave device 5110 may also be coupled to the shared bus 5104 and is configured to operate in both I2C mode and CCIe mode. An I2C-compatible slave device 5112 may also be coupled to the shared bus 5104 and is configured to operate in I2C mode. A CCIe-compatible slave device 5114 may also be coupled to the shared bus 5104 and is configured to operate in CCIe mode.

[00245] FIG. 52 illustrates the coexistence of the master/slave devices of FIG. 51 over a shared bus. The I2C-compatible and CCIe-compatible master device 5108 may initially operate in a first protocol mode (e.g., I2C mode) 5202. The slave devices 5110, 5112, 5114 may simply monitor the shared bus 5204 for commands/transmissions from the master device 5108 or other slave devices.

[00246] The I2C-compatible and CCIe-compatible master device 5108 may initiate a switch to a second protocol mode (e.g., CCIe mode) by sending an Entry Call message over the shared bus 5208. All slave devices 5110, 5112, and 5114 monitor the shared bus and recognize the Entry Call message as a change in how the shared bus is used (e.g., change from first protocol mode to second protocol mode). The entry call may be such that devices operating in the first protocol mode (e.g., I2C mode) and the second protocol mode (e.g., CCIe mode) can detect and/or decode the entry call. The I2C & CCIe Slave 1 5110 may still use the bus and can issue interrupt requests 5210a since it is able to communicate in both the first and second protocol modes. The I2C-compatible Slave 2 5112 may not use the bus and cannot issue interrupt requests 5210b since it is not able to communicate in the second protocol mode (CCIe mode). The CCIe-

compatible Slave 3 5114 may use the bus and can issue interrupt requests 5210c since it is able to communicate in the second protocol mode (CCIe mode).

[00247] The I2C-compatible and CCIe-compatible master device 5108 may subsequently switch back to the first protocol mode (e.g., I2C mode) by sending an Exit Call message over the shared bus 5212. All slave devices 5110, 5112, and 5114 monitor the shared bus and recognize the Exit Call message as a change in how the shared bus is used (e.g., change from second protocol mode to first protocol mode). The Exit Call may be such that devices operating in the first protocol mode (e.g., I2C mode) and the second protocol mode (e.g., CCIe mode) can detect and/or decode the exit call. The I2C & CCIe Slave 1 5110 may still use the bus and can issue interrupt requests 5216a since it is able to communicate in both the first and second protocol modes. The I2C-compatible Slave 2 5112 may use the bus and can issue interrupt requests 5216b since it able to communicate in the first protocol mode (I2C mode). The CCIe-compatible Slave 3 5114 may not use the bus and cannot issue interrupt requests 5216c since it is not able to communicate in the first protocol mode (I2C mode).

[00248] FIG. 53 illustrates a method operational by an I2C-compatible slave device that facilitates coexistence with CCIe-compatible slave devices over a shared bus, where all devices can use in-band interrupts over the shared bus. A first slave device (e.g., I2C-compatible slave device) may be operated according to a first protocol mode including in-band interrupt requests over a shared bus, the shared bus shared with a plurality of other slave devices 5302. The shared bus may be monitored by the first slave device for an entry call from a master device capable of operating in accordance with the first protocol mode and a second protocol mode 5304. Upon detecting the entry call, the first slave device may disable the first slave device from making any in-band interrupt requests according to the first protocol mode over the shared bus 5306. The first slave device may monitor the shared bus for an exit call from the master device 5308. Upon detection of an exit call from the master device over the shared bus, the first slave device may enable the first slave device to make an in-band interrupt request 5310. The plurality of other slave devices include one or more slave devices that operate according to a second protocol mode. The entry call may serve as an indicator to the one or more slave devices operating in the second protocol mode that the shared bus can operate according to the second protocol mode

[00249] In one example, the first slave device may comprise a bus interface coupled to a processing circuit. The bus interface may serve to couple to a shared bus shared

with a plurality of other slave devices. The processing circuit may be configured to: (a) operate the first slave device according to a first protocol mode including in-band interrupt requests over a shared bus, the shared bus shared with a plurality of other slave devices; (b) monitor the shared bus for an entry call from a master device capable of operating in accordance with the first protocol mode and a second protocol mode; (c) disable the first slave device from making any in-band interrupt requests according to the first protocol mode over the shared bus upon detection of the entry call; (d) monitor the shared bus for an exit call from the master device; and/or (e) enable the first slave device to make an in-band interrupt request over the shared bus upon detection of an exit call from the master device. The plurality of other slave devices may include one or more slave devices that operate according to a second protocol mode. The entry call may serve as an indicator to the one or more slave devices operating in the second protocol mode that the shared bus can operate according to the second protocol mode.

[00250] FIG. 54 illustrates a method operational by a CCIe-compatible slave device that facilitates coexistence with I2C-compatible slave devices over a shared bus, where all devices can use in-band interrupts over the shared bus. The slave device is coupled to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate according to a first protocol mode and the slave device operates according to a second protocol mode 5402. An entry call may be detected by the slave device over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode 5404. An interrupt request (IRQ) may be sent by the slave device either in-band over the shared bus or side-band over a separate path according to the second protocol mode 5406. The entry call may serve as an indicator to the slave device that the shared bus can operate according to the second protocol mode. The slave device may send data or commands over the shared bus according to the second protocol mode 5408.

[00251] The slave device may also monitor the shared bus for an exit call from the master device 5410. Upon detection of an exit call from the master device, the slave device may disable or stop making an in-band interrupt request over the shared bus 5412. The exit call may serve to indicate to the slave device that the shared bus is to operate according to the first protocol mode.

[00252] In one example, a slave device may include a bus interface coupled to a processing circuit. The bus interface may serve to couple to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate

according to a first protocol mode and the slave device operates according to a second protocol mode. The processing circuit may be configured to: (a) detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode; (b) send an interrupt request (IRQ) either in-band over the shared bus or side-band over a separate path according to the second protocol mode; (c) send data or commands over the shared bus according to the second protocol mode; (d) monitor the shared bus for an exit call from the master device; and/or (e) disable the slave device from making an in-band interrupt request over the shared bus upon detection of an exit call from the master device.

The entry call may serve as an indicator to the slave device that the shared bus can operate according to the second protocol mode. The exit call serves to indicate to the slave device that the shared bus is to operate according to the first protocol mode.

[00253] Similarly, a system is provided in which I2C-compatible devices and CCIe-compatible devices may coexist while both types of devices may use interrupt requests over a shared bus. A first set of devices may be coupled to the shared bus. A second set of devices may also be coupled to the shared bus.

[00254] A first device within the first set of devices may be configured to: (a) operate according to a first protocol mode including send/receiving in-band interrupt requests over the shared bus; (b) monitor the shared bus for an entry call from a device in the second set of devices; and/or (c) upon detection of the entry call, disable the first slave device from making any in-band interrupt requests over the shared bus.

[00255] A second device within the second set of devices may be configured to: (a) detect an entry call over the shared bus according to the first protocol mode; (b) send an interrupt request (IRQ) over the shared bus according to the second protocol mode; (c) send data or commands over the shared bus according to the second protocol mode; and/or (d) disable sending any in-band interrupt over the shared bus according to the second protocol mode if an exit call is detected on the shared bus.

[00256] A master device may be coupled to the shared bus and configured to: (a) operate according to both a first protocol mode and a second protocol mode; to manage communications over the shared bus; (b) send an entry call over the shared bus according to the first protocol mode; (c) detect interrupt requests from slave devices according to both the first protocol mode and the second protocol mode; and/or (d) respond to the interrupt requests by granting a requesting slave device access to the shared bus.

Exemplary Multi-Mode Shared Bus Architecture

[00257] A device is provided comprising a shared bus, a first slave device, a second slave device, and/or a master device. The first slave device may be coupled to the shared bus and is configured to: (a) operate according to a first protocol mode including issuing in-band interrupt requests over the shared bus; (b) monitor the shared bus for an entry call indicating the shared bus is switching to a second protocol mode; and/or (c) upon detection of the entry call, disable the first slave device from making in-band interrupt requests over the shared bus.

[00258] The second slave device may be coupled to the shared bus and is configured to: (a) operate according to the second protocol mode including issuing side-band interrupt requests over an interrupt bus or in-band interrupt request over the shared bus; and/or (b) detect the entry call over the shared bus, the entry call transmitted according to the second protocol mode.

[00259] The master device may be coupled to the shared bus and configured to: (a) operate according to both the first protocol mode and the second protocol mode; (b) manage communications over the shared bus; (c) send the entry call over the shared bus indicating the shared bus is switching to a second protocol mode, where the entry call is sent according to the first protocol mode; (d) detect interrupt requests from slave devices according to both the first protocol mode and the second protocol mode; and/or (e) respond to the interrupt requests by granting a requesting slave device access to the shared bus.

[00260] The master device may be further configured to send an exit call over the shared bus indicating the shared bus is switching to the first protocol mode, where the exit call is sent according to both the second protocol mode and the first protocol mode.

[00261] For in-band interrupts, the second protocol defines an interrupt period within symbols transmitted over the shared bus during which one or more slave devices coupled to the shared bus can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission. In one implementation, the master device and second slave device may be configured to internally mask the first line of the shared bus during the interrupt period.

[00262] The master device may be configured to send an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared

bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

[00263] The second slave device may be further configured to receive data or commands over the shared bus according to the second protocol mode when the shared bus operates according to the second protocol mode.

[00264] The first slave device may be further configured to send an interrupt request over a dedicated interrupt line or bus separate from the shared bus while the shared bus operates according to the second protocol mode.

[00265] The device may also include an interrupt router slave device configured to receive interrupt requests from the first slave device over a dedicated line and send the received interrupt request over the dedicated interrupt line or bus according to the second protocol.

[00266] In one example, the shared bus may include a first line and a second line. When the shared bus operates according to the first protocol mode, the first line is used for data transmissions and the second line is used for a first clock signal. When the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

Exemplary Multi-Mode CCIe/I2C Master Device

[00267] FIG. 55 illustrates an exemplary multi-mode master device 5502 that may include a first bus interface 5516, a second bus interface 5518, and a processing circuit 5504. The first bus interface may serve to couple to other devices coupled to a shared control data bus, the data bus dynamically configured to operate in either a first protocol mode or a second protocol mode. The second bus interface may server to couple to a dedicated interrupt line used by at least a subset of the other devices and which issue interrupt requests over the dedicated interrupt line in the second protocol mode. The processing circuit may be configured to: (a) manage data transfers over a shared bus to which a plurality of other devices are coupled, wherein at least a first subset of the other devices operate according to a first protocol mode and a second subset of the other devices operate according to a second protocol mode incompatible with the first protocol mode; (b) dynamically switch operation of the shared bus between the first protocol and second protocol mode by: (1) sending an entry call over the shared bus indicating that the shared bus is to operate according to the second

protocol mode, and/or (2) sending an exit call over the shared bus indicating that the shared bus is to operate according to the first protocol mode

[00268] The master device may be further configured to: (a) receive interrupt requests from the first subset of devices over the shared bus when the shared bus operates according to the first protocol mode; (b) receive interrupt requests from the second subset of devices over a dedicated interrupt bus when the shared bus operates according to the second protocol mode, and/or (c) receive interrupt requests from the first subset of devices over the dedicated interrupt bus when the shared bus operates according to the second protocol mode. In one implementation, no interrupt requests are received from the first subset of devices when the shared bus operates according to the second protocol mode.

[00269] The shared bus includes a first line and a second line. When the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal. When the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

[00270] The second protocol may define an interrupt period within symbols transmitted over the shared bus during which the second subset of the other devices can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission.

[00271] The master device may also be configured to internally mask the first line of the shared bus during the interrupt period.

[00272] In one implementation, in response to an in-band interrupt request while operating according to the second protocol mode, the master device may send an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

Exemplary CCIe Slave Device

[00273] FIG. 56 illustrates an exemplary slave device 5602 that may comprise a bus interface 5616 and a processing circuit 5604. The bus interface may serve to couple to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate according to a first protocol mode and the slave device operates

according to a second protocol mode. The processing circuit may be configured to: (a) detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; (b) send an interrupt request (IRQ) either in-band over the shared bus or side-band over a separate path to the master device; (c) communicate over the shared bus according to the second protocol mode; (d) monitor the shared bus for an exit call from the master device; and/or (c) disable the slave device from communicating over the shared bus upon detection of an exit call from the master device. The exit call may serve to indicate to the slave device that the shared bus is to operate according to the first protocol mode.

[00274] The shared bus may include a first line and a second line. When the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal. When the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

Exemplary I2C Slave Device

[00275] An exemplary CCIe slave device 1302 (FIG. 13) may include a bus interface 1306 and a processing circuit 1304. The bus interface may serve to couple to a shared bus shared with a plurality of other devices, wherein the slave device operates according to a first protocol mode and at least a first subset of the other devices operate according to a second protocol mode. The processing circuit may serve to coupled to the bus interface and configured to: (a) detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; and/or (b) disable the slave device from making in-band interrupt requests over the shared bus upon detection of the entry call.

[00276] The processing circuit is further configured to monitor the shared bus for an exit call from the master device, the exit call indicating that the shared bus is to operate according to the first protocol mode. The shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal, and when the shared bus operates according to the second protocol mode, both the first line and

second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

Exemplary Interrupt Request Router Slave Device

[00277] FIG. 57 illustrates an exemplary interrupt request router slave device 5702 that may include a first bus interface 5716, a second bus interface 5718, a third bus interface 5720, and a processing circuit 5704. The first bus interface may serve to couple to a shared control data bus, the data bus dynamically configured to operate in either a first protocol mode or a second protocol mode. The second bus interface may serve to couple to at least one subset of other slave device coupled to the shared bus, the subset of other slave devices operating according to the first protocol mode including in-band interrupt requests over the shared control data bus. The third bus interface may serve to couple to a dedicated interrupt line used in the second protocol mode to issue interrupt request to a master device that manages the shared bus. The processing circuit may be configured to: (a) receive an interrupt request over the second bus interface from a slave device within the first subset of slave devices; and/or (b) route the received interrupt request to the master device via the third bus interface.

[00278] One or more of the components, steps, features, and/or functions illustrated in the Figures may be rearranged and/or combined into a single component, step, feature, or function or embodied in several components, steps, or functions. Additional elements, components, steps, and/or functions may also be added without departing from novel features disclosed herein. The apparatus, devices, and/or components illustrated in the Figures may be configured to perform one or more of the methods, features, or steps described in the Figures. The novel algorithms described herein may also be efficiently implemented in software and/or embedded in hardware.

[00279] In addition, it is noted that the embodiments may be described as a process that is depicted as a flowchart, a flow diagram, a structure diagram, or a block diagram. Although a flowchart may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be re-arranged. A process is terminated when its operations are completed. A process may correspond to a method, a function, a procedure, a subroutine, a subprogram, etc. When a process corresponds to a function, its termination corresponds to a return of the function to the calling function or the main function.

[00280] Moreover, a storage medium may represent one or more devices for storing data, including read-only memory (ROM), random access memory (RAM), magnetic disk storage mediums, optical storage mediums, flash memory devices, and/or other machine readable mediums for storing information. The term "machine readable medium" includes, but is not limited to portable or fixed storage devices, optical storage devices, wireless channels and various other mediums capable of storing, containing, or carrying instruction(s) and/or data.

[00281] Furthermore, embodiments may be implemented by hardware, software, firmware, middleware, microcode, or any combination thereof. When implemented in software, firmware, middleware, or microcode, the program code or code segments to perform the necessary tasks may be stored in a machine-readable medium such as a storage medium or other storage(s). A processor may perform the necessary tasks. A code segment may represent a procedure, a function, a subprogram, a program, a routine, a subroutine, a module, a software package, a class, or any combination of instructions, data structures, or program statements. A code segment may be coupled to another code segment or a hardware circuit by passing and/or receiving information, data, arguments, parameters, or memory contents. Information, arguments, parameters, data, etc. may be passed, forwarded, or transmitted via any suitable means including memory sharing, message passing, token passing, network transmission, etc.

[00282] The various illustrative logical blocks, modules, circuits, elements, and/or components described in connection with the examples disclosed herein may be implemented or performed with a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other programmable logic component, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein. A general purpose processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing components, e.g., a combination of a DSP and a microprocessor, a number of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[00283] The methods or algorithms described in connection with the examples disclosed herein may be embodied directly in hardware, in a software module executable by a processor, or in a combination of both, in the form of processing unit,

programming instructions, or other directions, and may be contained in a single device or distributed across multiple devices. A software module may reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. A storage medium may be coupled to the processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor.

[00284] Those of skill in the art would further appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system.

[00285] The various features of the invention described herein can be implemented in different systems without departing from the invention. It should be noted that the foregoing embodiments are merely examples and are not to be construed as limiting the invention. The description of the embodiments is intended to be illustrative, and not to limit the scope of the claims. As such, the present teachings can be readily applied to other types of apparatuses and many alternatives, modifications, and variations will be apparent to those skilled in the art.

CLAIMS

WHAT IS CLAIMED IS:

1. A device, comprising:

a shared bus;

a first slave device coupled to the shared bus and configured to

operate according to a first protocol mode including issuing in-band interrupt requests over the shared bus;

monitor the shared bus for an entry call indicating the shared bus is switching to a second protocol mode; and

upon detection of the entry call, disable the first slave device from making in-band interrupt requests over the shared bus; and

a second slave device coupled to the shared bus and configured to

operate according to the second protocol mode including issuing side-band interrupt requests over an interrupt bus or in-band interrupt request over the shared bus;

detect the entry call over the shared bus, the entry call transmitted according to the second protocol mode.

2. The device of claim 1, further comprising:

a master device coupled to the shared bus and configured to:

operate according to both the first protocol mode and the second protocol mode;

manage communications over the shared bus; and

send the entry call over the shared bus indicating the shared bus is switching to a second protocol mode, where the entry call is sent according to the first protocol mode.

3. The device of claim 2, wherein the master device is further configured to:

detect interrupt requests from slave devices according to both the first protocol mode and the second protocol mode; and

respond to the interrupt requests by granting a requesting slave device access to the shared bus.

4. The device of claim 2, wherein the master device is further configured to:

send an exit call over the shared bus indicating the shared bus is switching to the first protocol mode, where the exit call is sent according to both the second protocol mode and the first protocol mode.

5. The device of claim 2, wherein for in-band interrupts, the second protocol defines an interrupt period within symbols transmitted over the shared bus during which one or more slave devices coupled to the shared bus can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission.

6. The device of claim 5, wherein the master device and second slave device are configured to internally mask the first line of the shared bus during the interrupt period.

7. The device of claim 5, wherein the master device is configured to send an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

8. The device of claim 1, wherein the second slave device is further configured to:

receive data or commands over the shared bus according to the second protocol mode when the shared bus operates according to the second protocol mode.

9. The device of claim 1, wherein the first slave device is further configured to

send an interrupt request over a dedicated interrupt line or bus separate from the shared bus while the shared bus operates according to the second protocol mode.

10. The device of claim 1, further comprising:

an interrupt router slave device configured to receive interrupt requests from the first slave device over a dedicated line and send the received interrupt request over the dedicated interrupt line or bus according to the second protocol.

11. The device of claim 1, wherein the shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line is used for data transmissions and the second line is used for a first clock signal, and when

the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

12. A multi-mode master device, comprising:

a first bus interface to couple to other devices coupled to a shared control data bus, the data bus dynamically configured to operate in either a first protocol mode or a second protocol mode;

a second bus interface to couple to a dedicated interrupt line used by at least a subset of the other devices and which issue interrupt requests over the dedicated interrupt line in the second protocol mode; and

a processing circuit coupled to the first bus interface and the second bus interface and configured to:

manage data transfers over a shared bus to which a plurality of other devices are coupled, wherein at least a first subset of the other devices operate according to a first protocol mode and a second subset of the other devices operate according to a second protocol mode incompatible with the first protocol mode; and

dynamically switch operation of the shared bus between the first protocol and second protocol mode by:

sending an entry call over the shared bus indicating that the shared bus is to operate according to the second protocol mode, and/or (

sending an exit call over the shared bus indicating that the shared bus is to operate according to the first protocol mode

13. The master device of claim 12, wherein the processing circuit is further configured to:

receive interrupt requests from the first subset of devices over the shared bus when the shared bus operates according to the first protocol mode.

14. The master device of claim 12, wherein the processing circuit is further configured to:

receive interrupt requests from the second subset of devices over a dedicated interrupt bus when the shared bus operates according to the second protocol mode,

15. The master device of claim 12, wherein the processing circuit is further configured to:

receive interrupt requests from the first subset of devices over the dedicated interrupt bus when the shared bus operates according to the second protocol mode.

16. The master device of claim 12, wherein no interrupt requests are received from the first subset of devices when the shared bus operates according to the second protocol mode.

17. The master device of claim 12, wherein the shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal, and when the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

18. The master device of claim 12, wherein the second protocol defines an interrupt period within symbols transmitted over the shared bus during which the second subset of the other devices can assert an interrupt request on a first line of the shared bus while a second line of the shared bus is used by the master device for a heartbeat transmission.

19. The master device of claim 18, wherein the processing circuit is further configured to internally mask the first line of the shared bus during the interrupt period.

20. The master device of claim 12, wherein response to an in-band interrupt request while operating according to the second protocol mode, the master device sends an interrupt group inquiry call to all slave devices that operate according to the second protocol mode on the shared bus, where such interrupt group inquiry call provides slots in which any asserting slave devices can respond.

21. A slave device, comprising:

a bus interface to couple to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate according to a first protocol mode and the slave device operates according to a second protocol mode; and

a processing circuit coupled to the bus interface and configured to:

detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; and

send an interrupt request (IRQ) either in-band over the shared bus or side-band over a separate path to the master device.

22. The slave device of claim 21, wherein the processing circuit is further configured to:

communicate over the shared bus according to the second protocol mode.

23. The slave device of claim 21, wherein the processing circuit is further configured to:

monitor the shared bus for an exit call from the master device; and

disable the slave device from communicating over the shared bus upon detection of an exit call from the master device.

24. The slave device of claim 21, wherein the exit call serves to indicate to the slave device that the shared bus is to operate according to the first protocol mode.

25. The slave device of claim 21, wherein the shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal, and when the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

26. A slave device, comprising:

a bus interface to couple to a shared bus shared with a plurality of other devices, wherein the slave device operates according to a first protocol mode and at least a first subset of the other devices operate according to a second protocol mode; and

a processing circuit coupled to the bus interface and configured to:

detect an entry call over the shared bus from a master device capable of operating according to the first protocol mode and the second protocol mode, the entry call indicating that the shared bus is to operate according to the second protocol mode; and

disable the slave device from making in-band interrupt requests over the shared bus upon detection of the entry call.

27. The slave device of claim 26, wherein the processing circuit is further configured to:

monitor the shared bus for an exit call from the master device, the exit call indicating that the shared bus is to operate according to the first protocol mode.

28. The slave device of claim 26, wherein the shared bus includes a first line and a second line, when the shared bus operates according to the first protocol mode, the first line used for data transmissions and the second line used for a first clock signal, and when the shared bus operates according to the second protocol mode, both the first line and second line are used for data transmissions while a second clock signal is embedded in symbol-to-symbol transitions with the data transmissions.

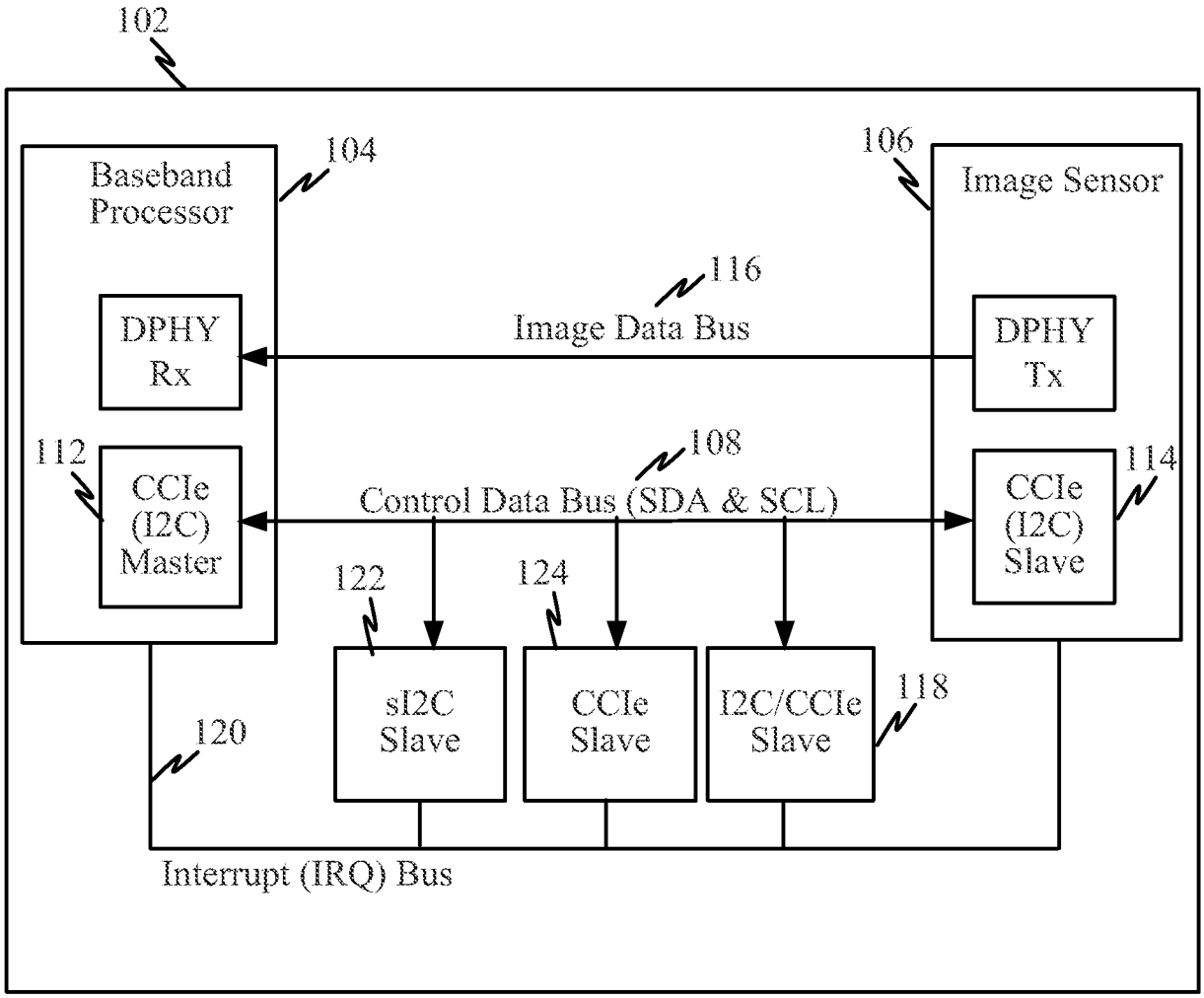

102
Baseband Processor
104
Image Sensor
106
116
Image Data Bus
DPHY Rx
DPHY Tx
112
CCIe (I2C) Master
108
Control Data Bus (SDA & SCL)
CCIe (I2C) Slave
114
122
124
sI2C Slave
CCIe Slave
I2C/CCIe Slave
118
120
Interrupt (IRQ) Bus

FIG. 1

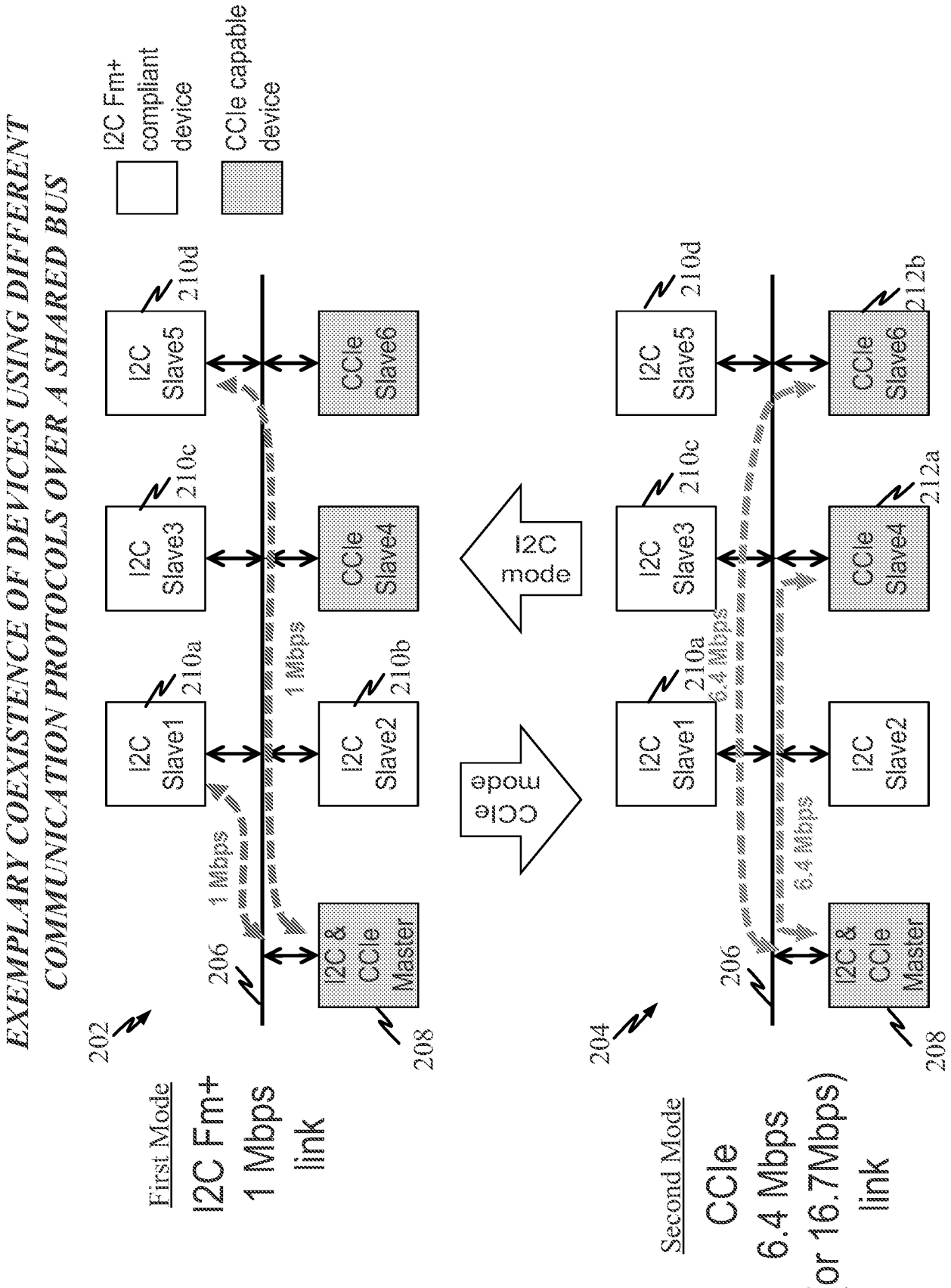

EXEMPLARY COEXISTENCE OF DEVICES USING DIFFERENT COMMUNICATION PROTOCOLS OVER A SHARED BUS
I2C Fm+ compliant device
CCIe capable device
First Mode
I2C Fm+
1 Mbps
link
202
206
208
I2C Slave1
210a
I2C Slave2
210b
I2C Slave3
210c
I2C Slave5
210d
CCIe Slave4
CCIe Slave6
I2C & CCIe Master
1 Mbps
1 Mbps
I2C mode
CCIe mode
Second Mode
CCIe
6.4 Mbps
(or 16.7Mbps)
link
204
6.4 Mbps
6.4 Mbps
212a
212b

FIG. 2

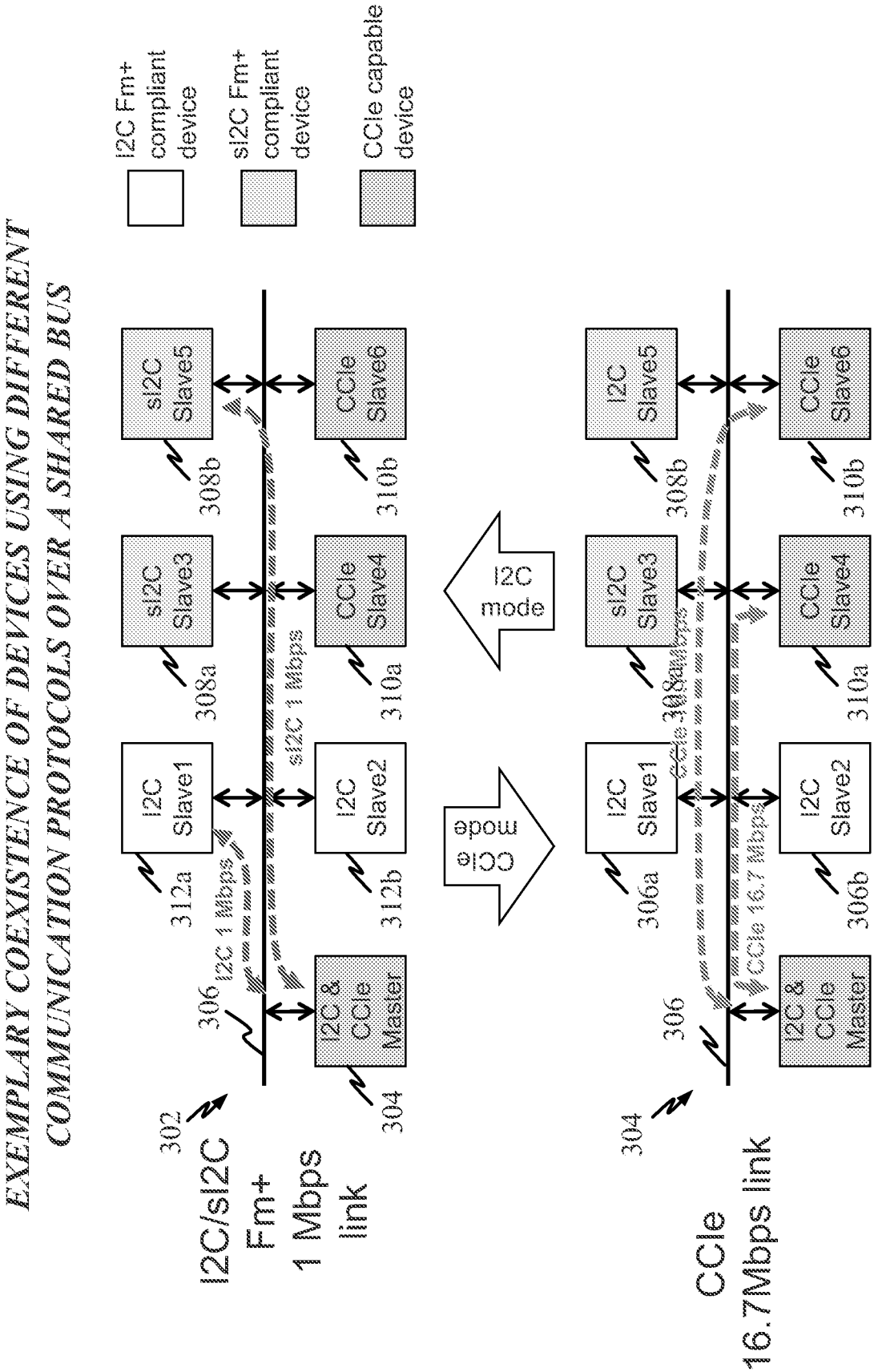

EXEMPLARY COEXISTENCE OF DEVICES USING DIFFERENT COMMUNICATION PROTOCOLS OVER A SHARED BUS
I2C Fm+ compliant device
sI2C Fm+ compliant device
CCIe capable device
I2C/sI2C Fm+ 1 Mbps link
302
306
304
I2C & CCIe Master
I2C Slave1
I2C Slave2
sI2C Slave3
sI2C Slave5
CCIe Slave4
CCIe Slave6
312a
312b
308a
308b
310a
310b
I2C 1 Mbps
sI2C 1 Mbps
CCIe mode
I2C mode
CCIe 16.7Mbps link
306a
306b
CCIe 16.7 Mbps

FIG. 3

EXEMPLARY TRANSMISSION AND RECEPTION OF CCIe MODE COMMUNICATIONS OVER TWO-LINE SHARED BUS

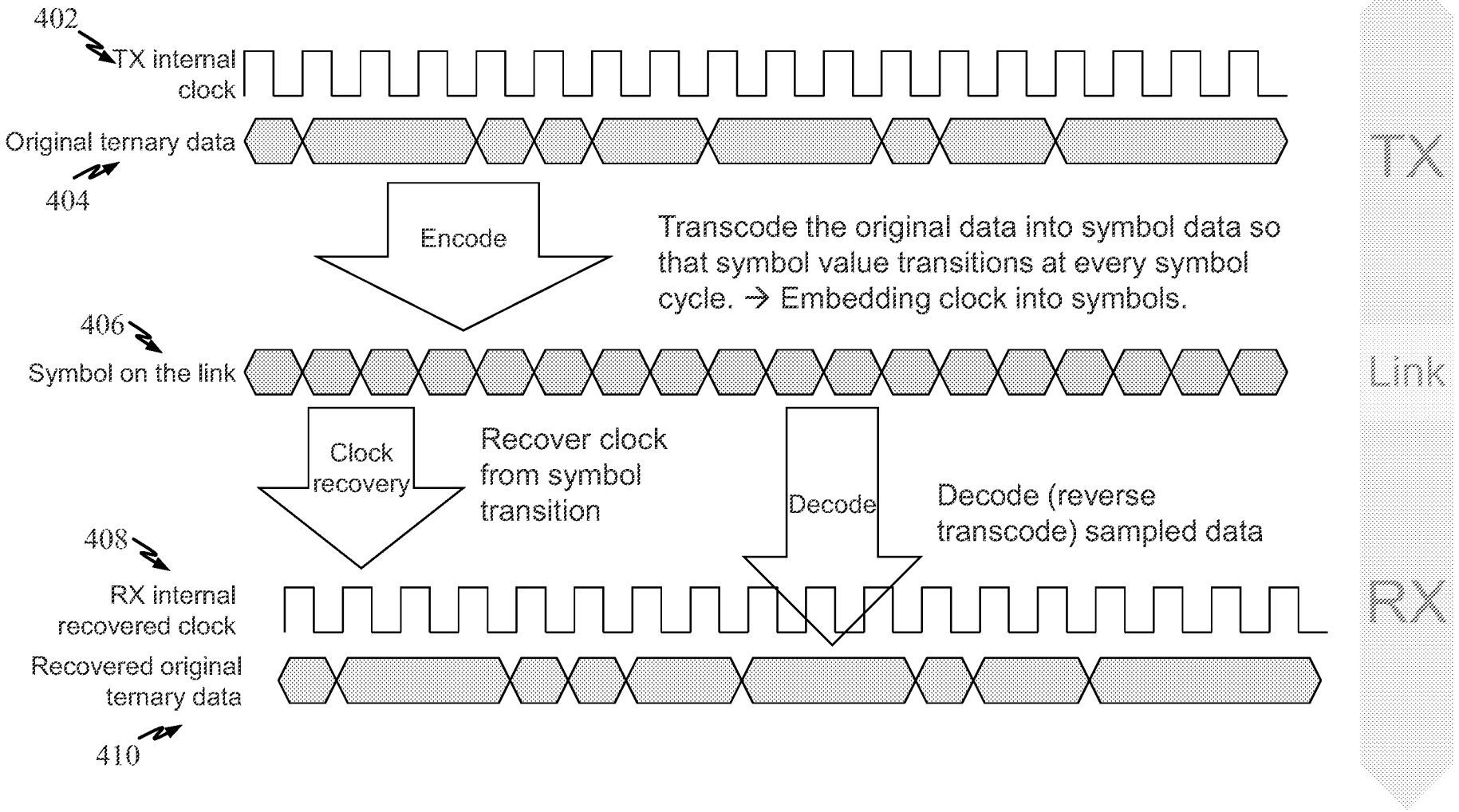


FIG. 4

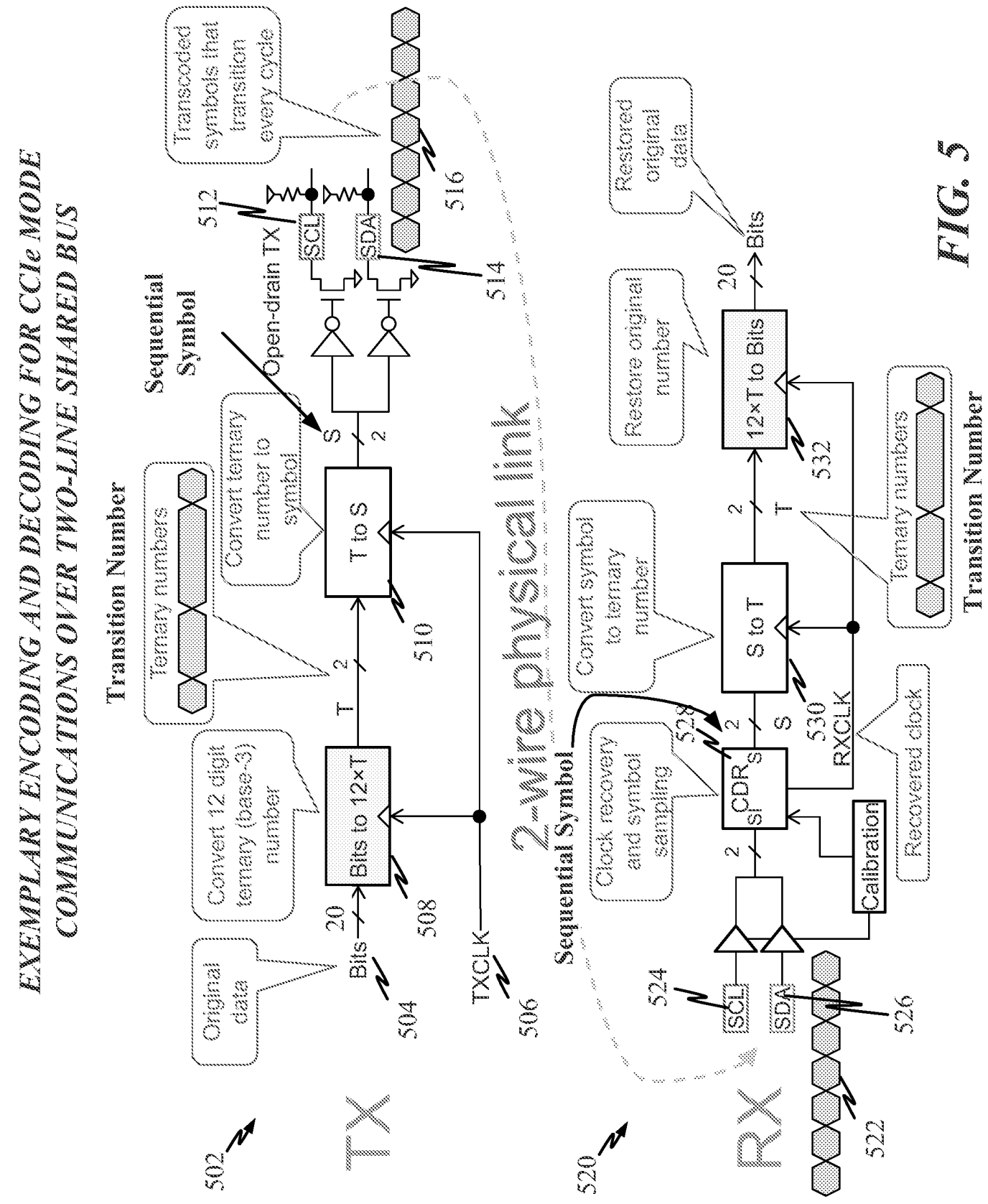

EXEMPLARY ENCODING AND DECODING FOR CCIe MODE COMMUNICATIONS OVER TWO-LINE SHARED BUS
Transition Number
Ternary numbers
Convert ternary number to symbol
Sequential Symbol
Open-drain TX
Transcoded symbols that transition every cycle
512
SCL
SDA
516
514
T to S
S
2
T
2
510
Convert 12 digit ternary (base-3) number
Bits to 12×T
20
Bits
508
Original data
504
TXCLK
506
TX
502
2-wire physical link
Sequential Symbol
Convert symbol to ternary number
Clock recovery and symbol sampling
528
CDR
S
2
S
530
RXCLK
S to T
2
T
532
12×T to Bits
20
Bits
Restore original number
Restored original data
Ternary numbers
Calibration
Recovered clock
524
SCL
SDA
526
RX
520
522
Transition Number

FIG. 5

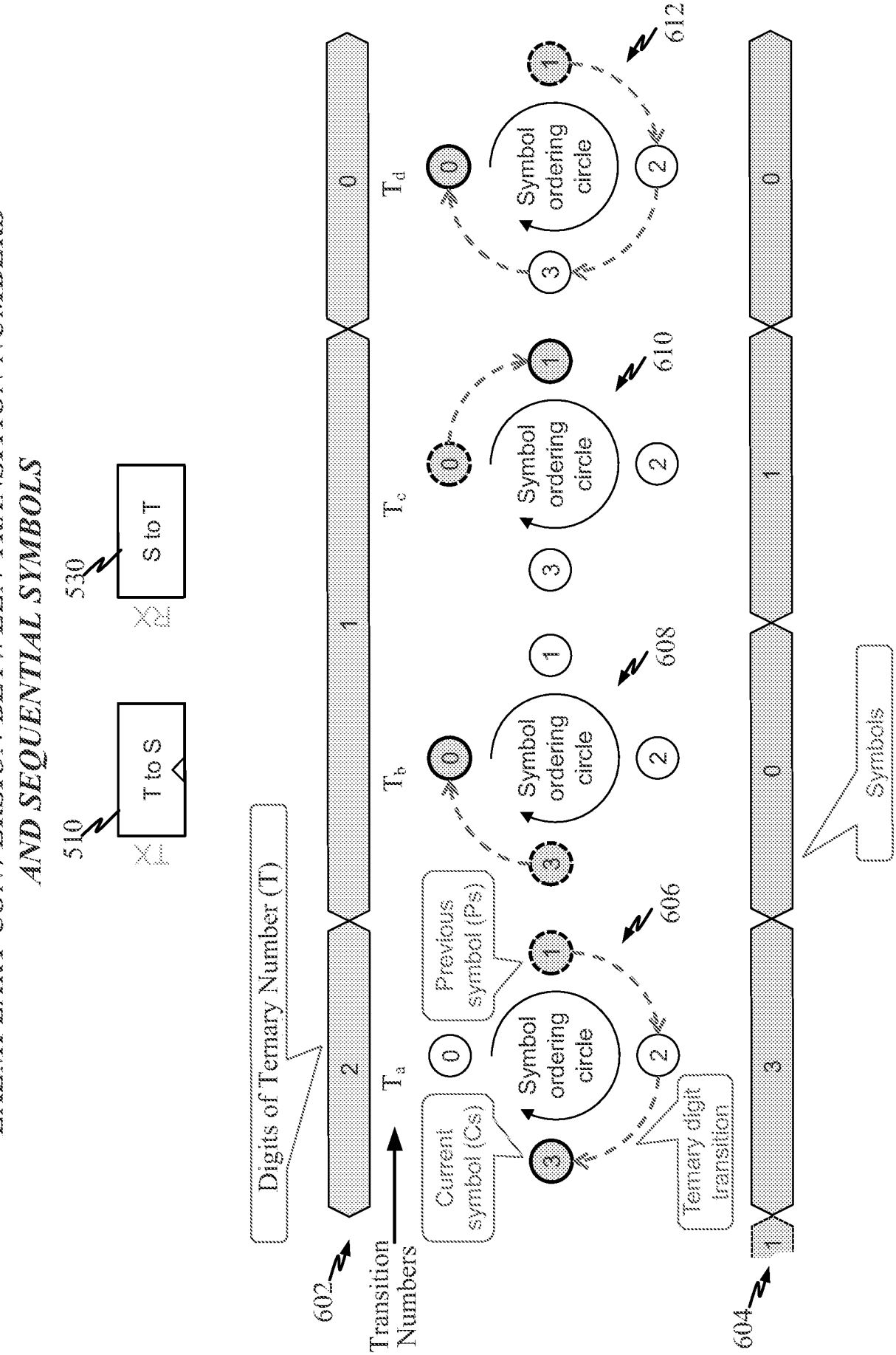

EXEMPLARY CONVERSION BETWEEN TRANSITION NUMBERS AND SEQUENTIAL SYMBOLS
510
T to S
530
S to T
Digits of Ternary Number (T)
Transition Numbers
602
604
606
608
610
612
Current symbol (Cs)
Previous symbol (Ps)
Ternary digit transition
Symbol ordering circle
Symbols
T_a
T_b
T_c
T_d

FIG. 6

Assign each symbol transition a number: T

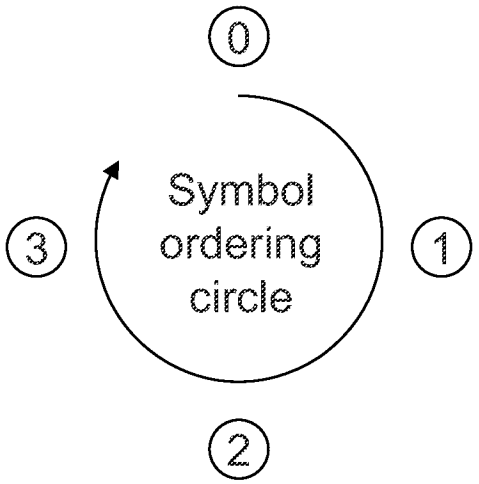
706

In CCIe system, there are 4 raw possible symbols; S=0, 1, 2, and 3

In any sequence of two raw symbols, same symbols can't appear
→ 3 possible cases per each transition

Assign sequential index to each transition

Previous symbol Ps	Current symbol Cs	Transition number T
0	1	1
	2	2
	3	0
1	2	1
	3	2
	0	0
2	3	1
	0	2
	1	0
3	0	1
	1	2
	2	0



702 TX: T to S

$T_{tmp} = T==0 \ ? \ 3 : T$

$Cs = Ps + T_{tmp}$

704 RX: S to T

$T_{tmp} = 4 + Cs - Ps$

$T = T_{tmp}==3 \ ? \ 0 : T_{tmp}$

FIG. 7

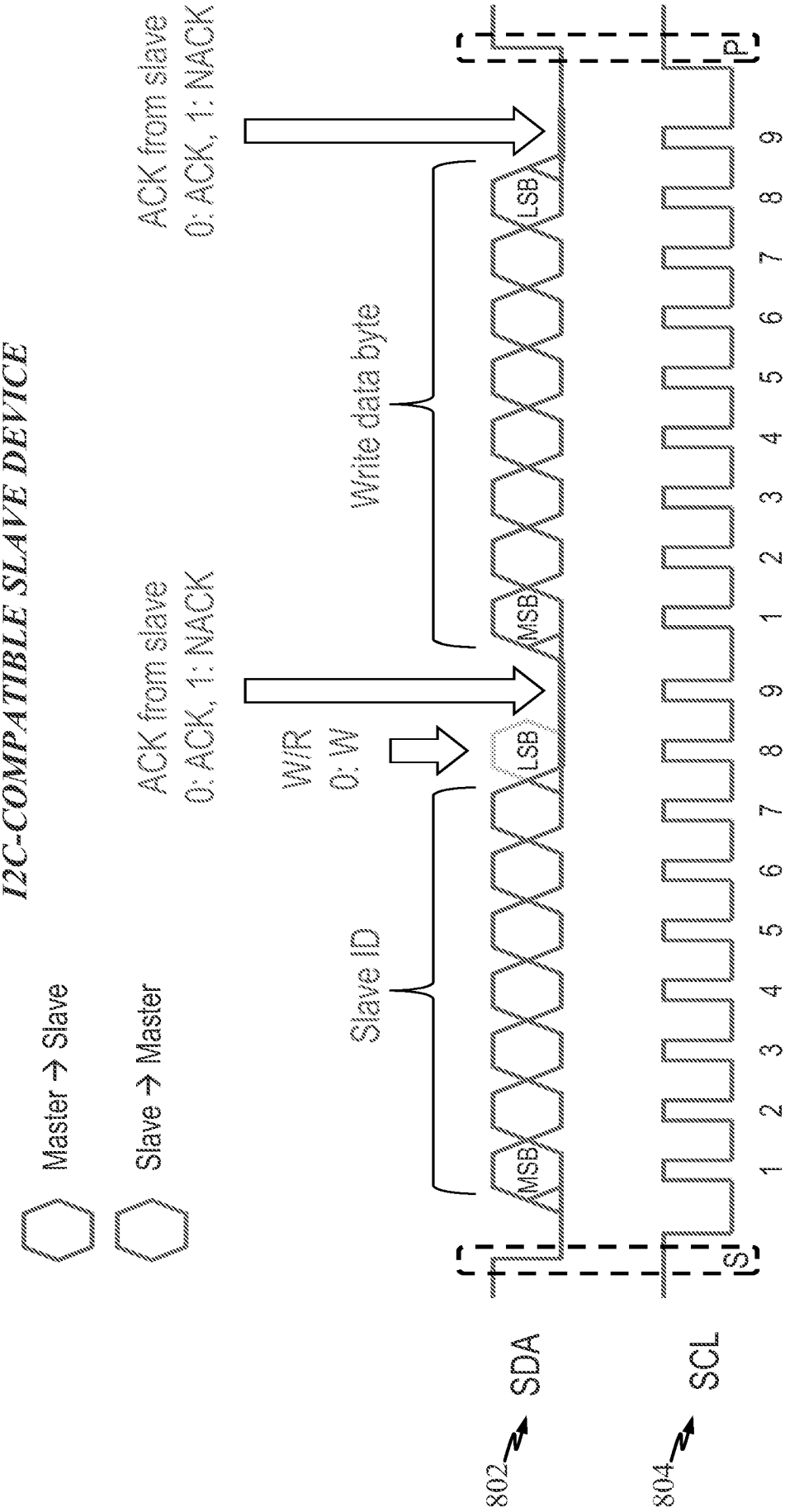

EXEMPLARY IN-BAND INTERRUPT SIGNAL FROM I2C-COMPATIBLE SLAVE DEVICE
Master → Slave
Slave → Master
ACK from slave
0: ACK, 1: NACK
Write data byte
ACK from slave
0: ACK, 1: NACK
WR
0: W
Slave ID
MSB
LSB
MSB
LSB
S
P
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
SDA
SCL
802
804

FIG. 8

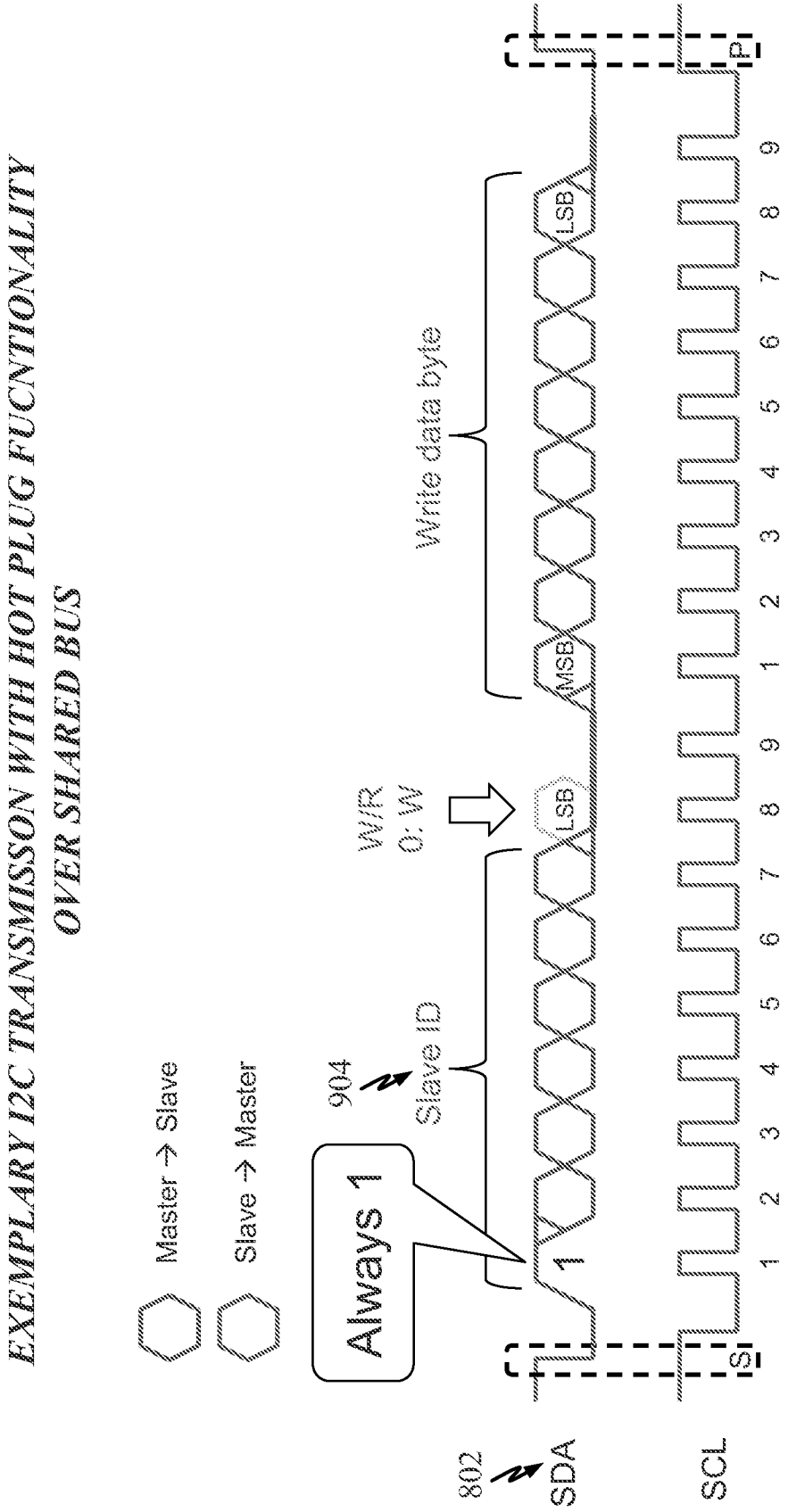

EXEMPLARY I2C TRANSMISSON WITH HOT PLUG FUCNTIONALITY OVER SHARED BUS
Master → Slave
Slave → Master
Always 1
904
Slave ID
W/R
0: W
Write data byte
MSB
LSB
LSB
S
P
1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9
802
SDA
SCL

FIG. 9

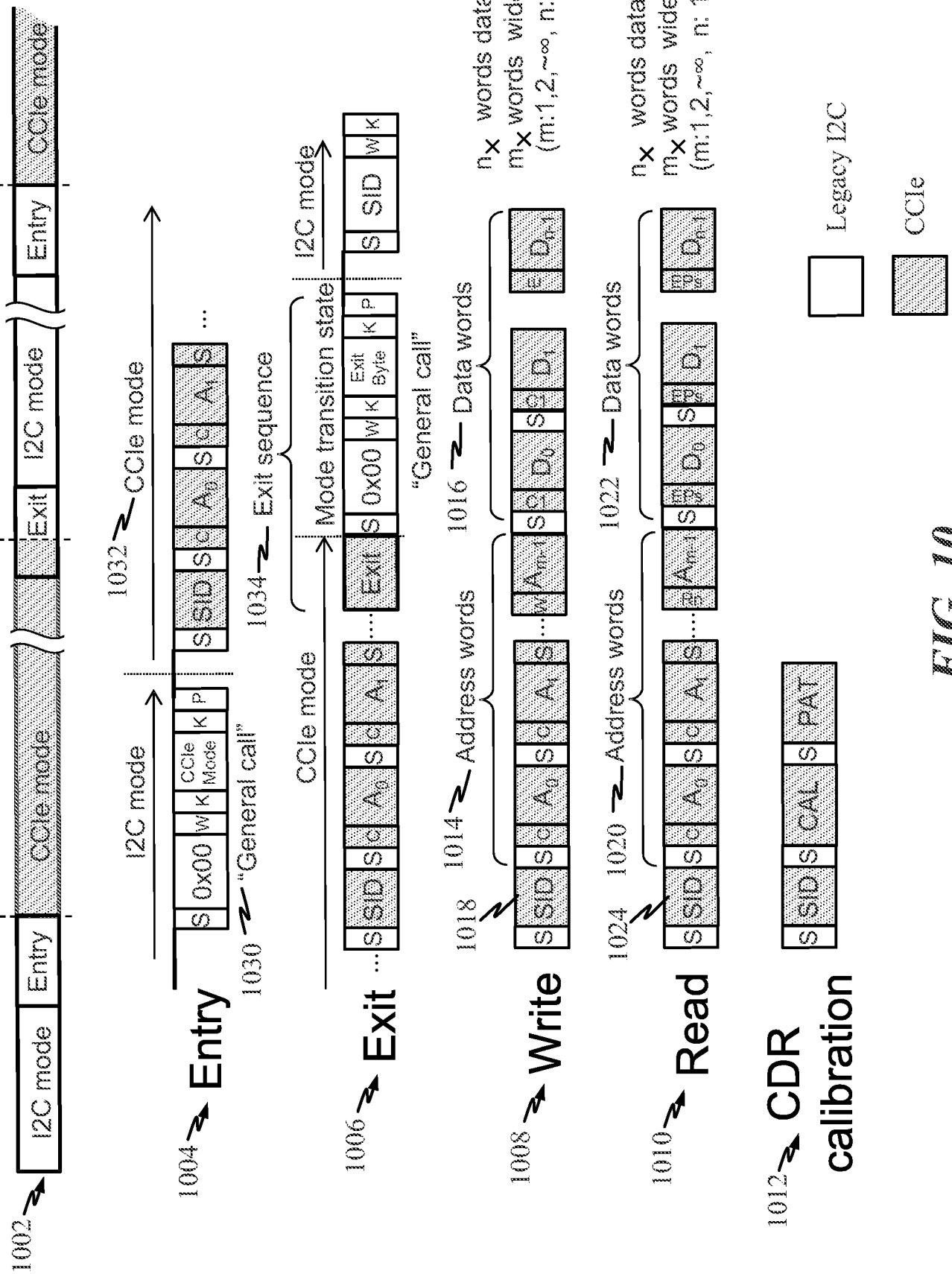

1002
I2C mode
Entry
6.4Mbps / 16.7Mbps
CCIe mode
Exit
1Mbps
I2C mode
Entry
CCIe mode
1004 Entry
I2C mode
CCIe mode
1030 "General call"
1032
1034 Exit sequence
1006 Exit
CCIe mode
Mode transition state
I2C mode
"General call"
1008 Write
1014 Address words
1018
1016 Data words
n× words data write to m× words wide address (m:1,2,~∞, n: 1,2,~∞)
1010 Read
1020 Address words
1024
1022 Data words
n× words data read from m× words wide address (m:1,2,~∞, n: 1,2,3,4,5)
1012 CDR calibration
Legacy I2C
CCIe

FIG. 10

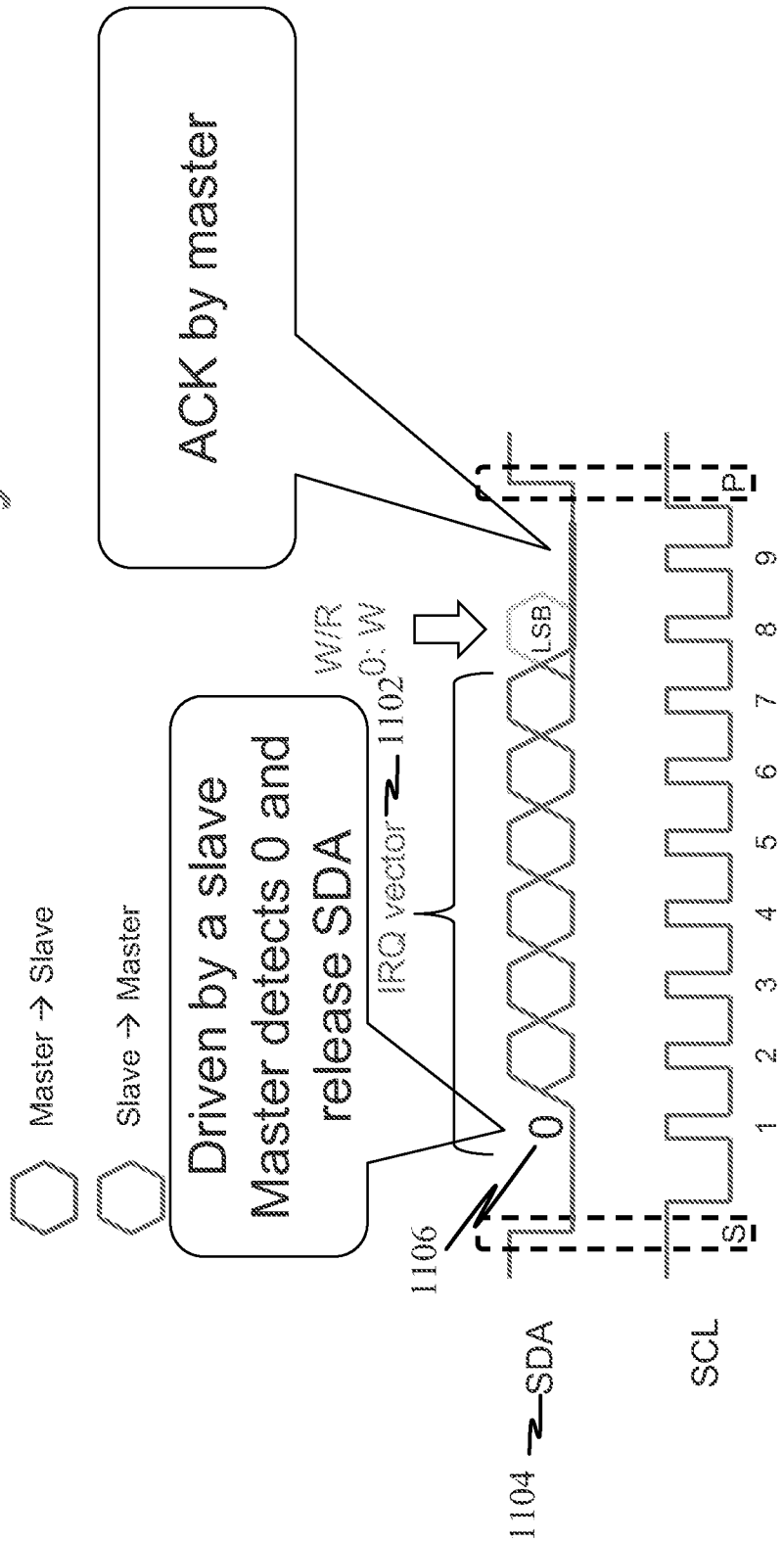

sI2C: IRQ by a slave
Master → Slave
Slave → Master
Driven by a slave
Master detects 0 and
release SDA
ACK by master
W/R
IRQ vector 1102 0: W
1106
1104 SDA
SCL
S
0
LSB
1 2 3 4 5 6 7 8 9
P

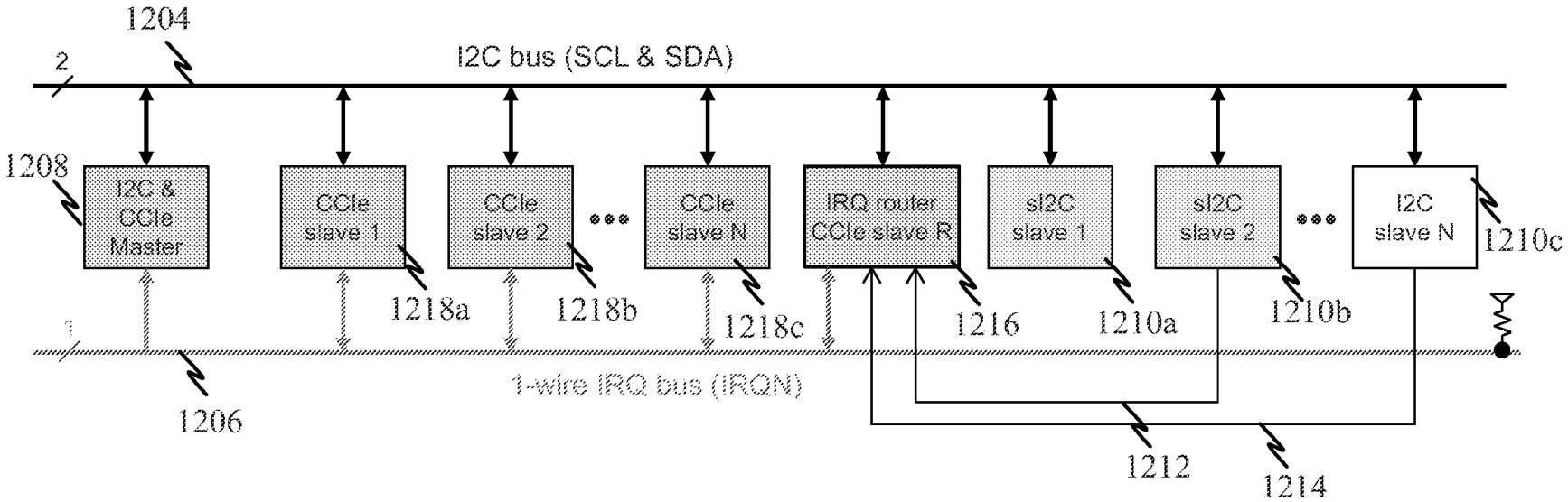
FIG. 11

IRQ router CCIe slave for mixed bus

1202

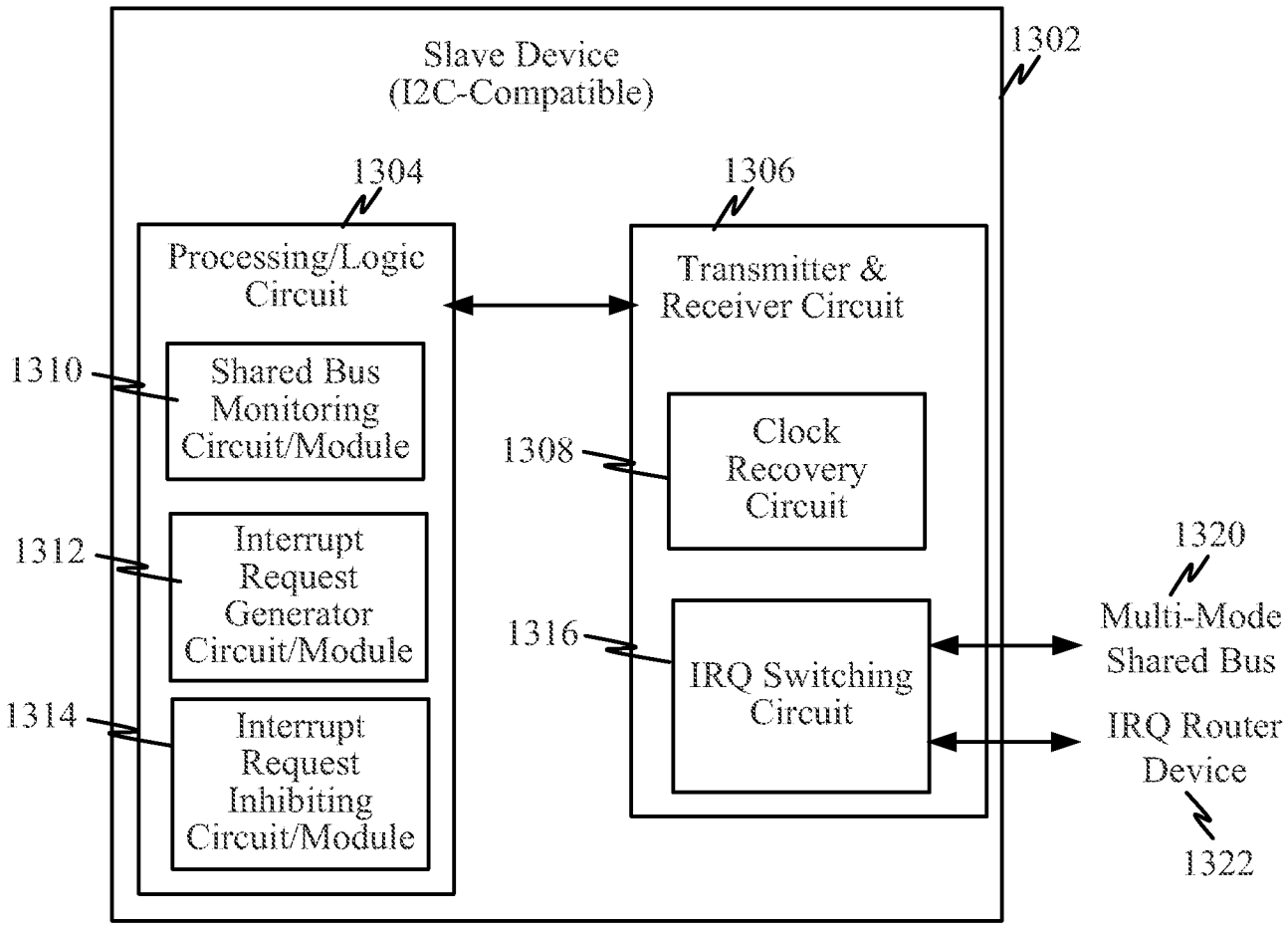
- If an sI2C device wishes to issue IRQ during CCIe mode ...

For example, slave 1 is OK to disable IRQ during CCIe mode, but slave 2 still need to issue IRQ during CCIe mode.



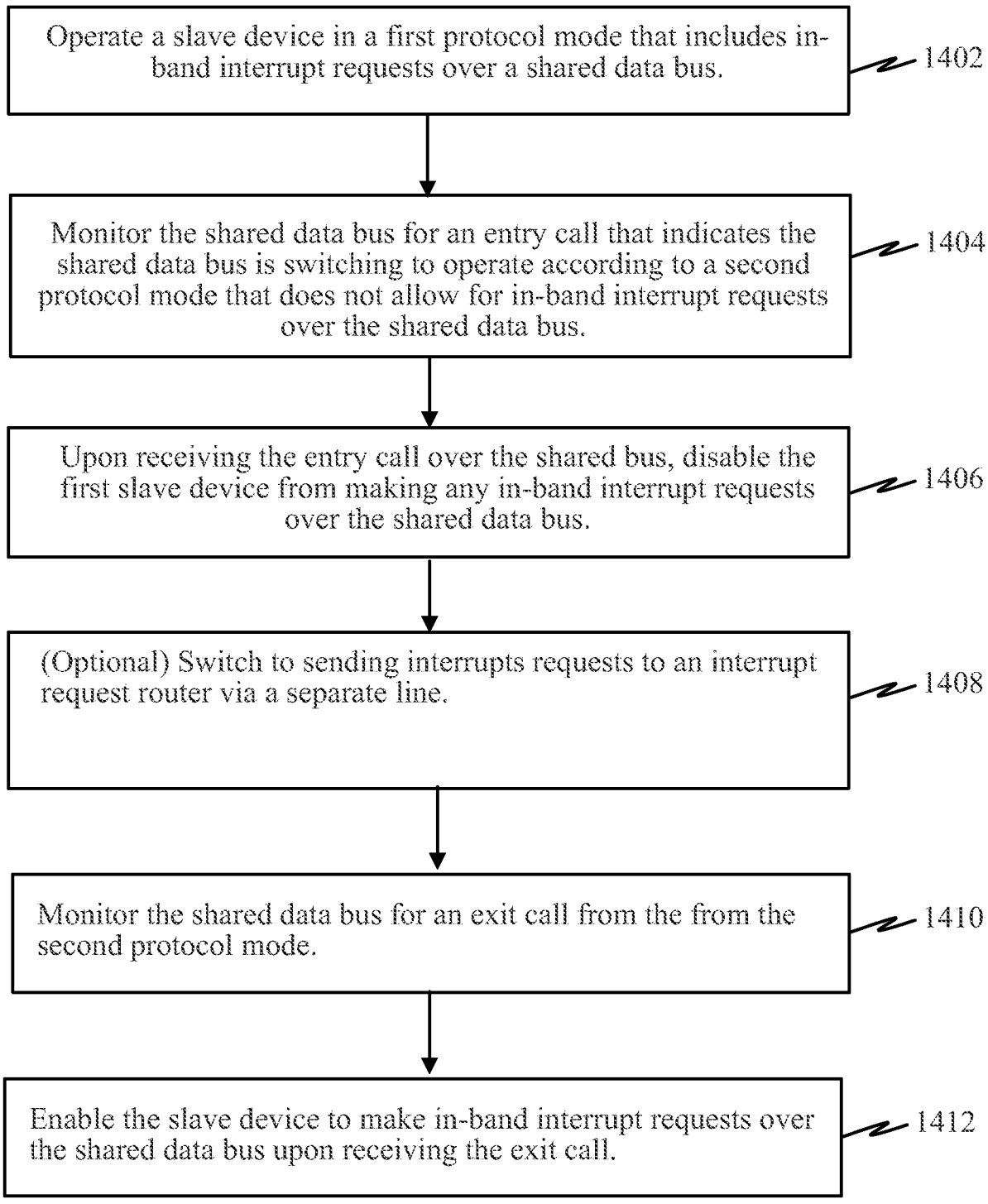
- Workaround: have slave 2 have an IRQ output via an CCIe IRQ router.

FIG. 12



EXEMPLARY LEGACY I2C SLAVE DEVICE

FIG. 13



EXEMPLARY METHOD OPERATIONAL ON LEGACY I2C SLAVE DEVICE OVER A SHARED BUS

FIG. 14

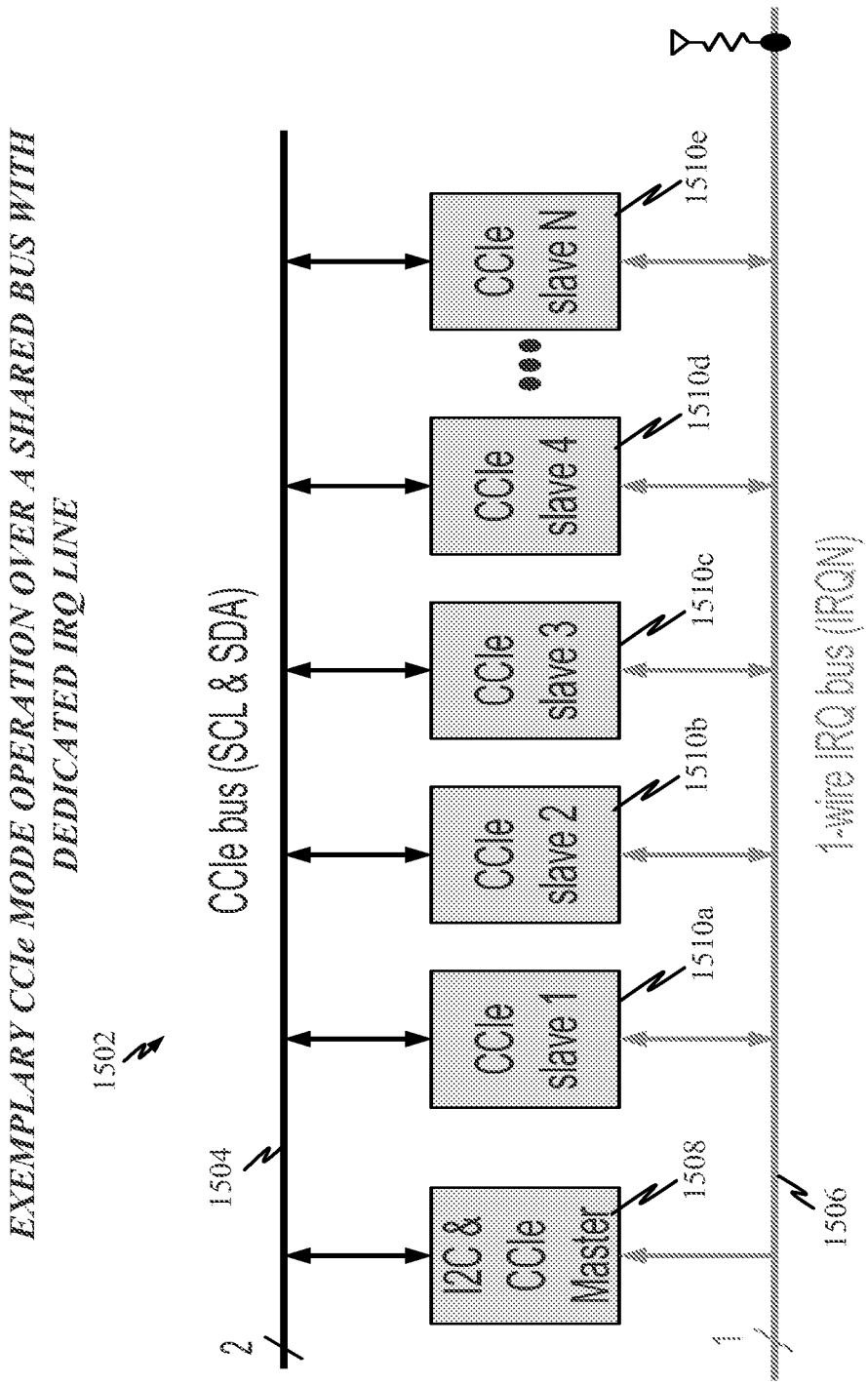

EXEMPLARY CCIe MODE OPERATION OVER A SHARED BUS WITH DEDICATED IRQ LINE
1502
1504
CCIe bus (SCL & SDA)
2
1
1506
1-wire IRQ bus (IRQN)
I2C & CCIe Master
1508
CCIe slave 1
1510a
CCIe slave 2
1510b
CCIe slave 3
1510c
CCIe slave 4
1510d
CCIe slave N
1510e

FIG. 15

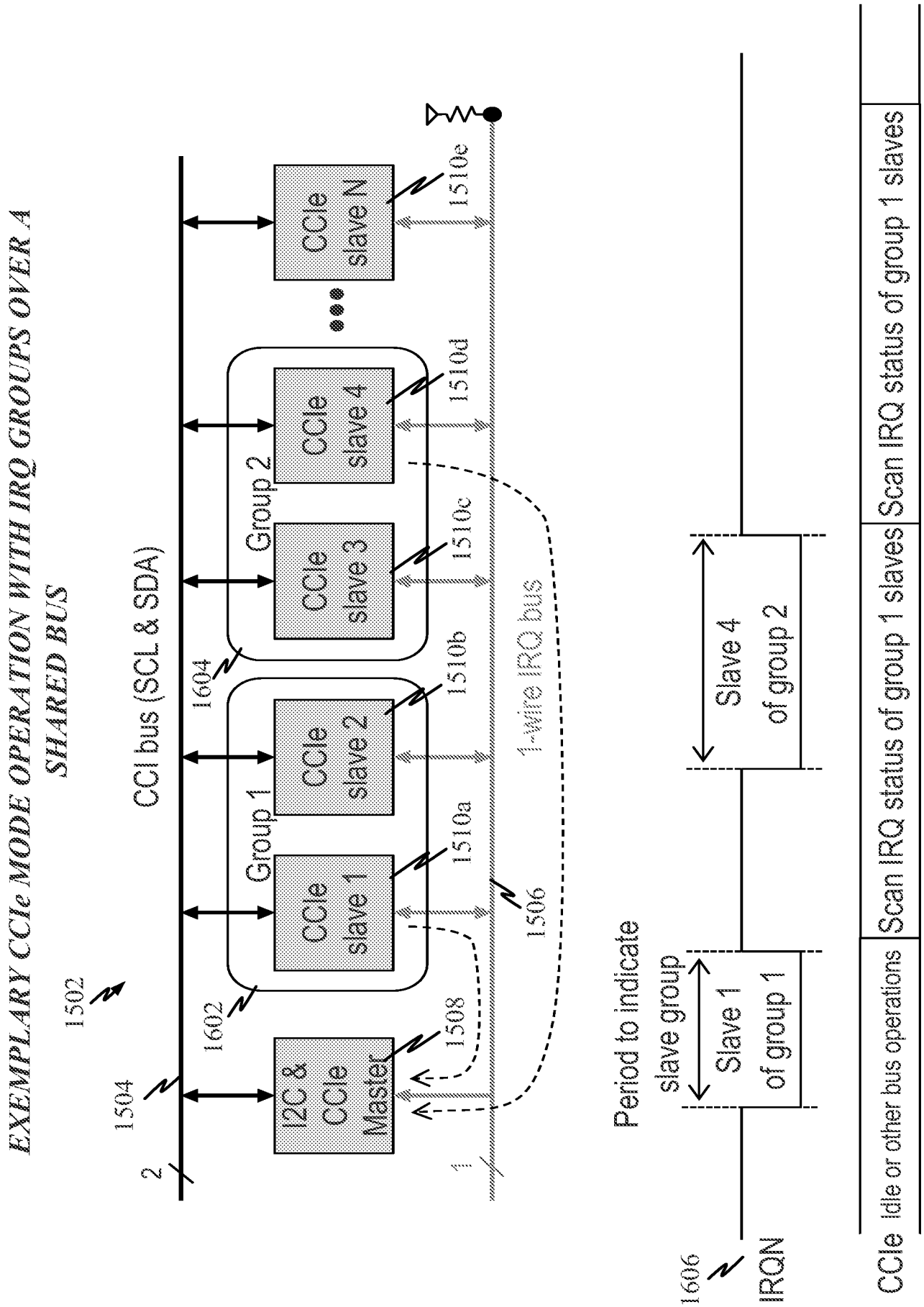

EXEMPLARY CCIe MODE OPERATION WITH IRQ GROUPS OVER A SHARED BUS
1502
CCI bus (SCL & SDA)
1504
2
I2C & CCIe Master
1508
1
1602
Group 1
CCIe slave 1
CCIe slave 2
1510a
1510b
1604
Group 2
CCIe slave 3
CCIe slave 4
1510c
1510d
CCIe slave N
1510e
1-wire IRQ bus
1506
Period to indicate slave group
Slave 1 of group 1
Slave 4 of group 2
1606
IRQN
CCIe
Idle or other bus operations
Scan IRQ status of group 1 slaves
Scan IRQ status of group 1 slaves

FIG. 16

EXEMPLARY CCIe MODE IRQ GROUPS OVER A SHARED BUS

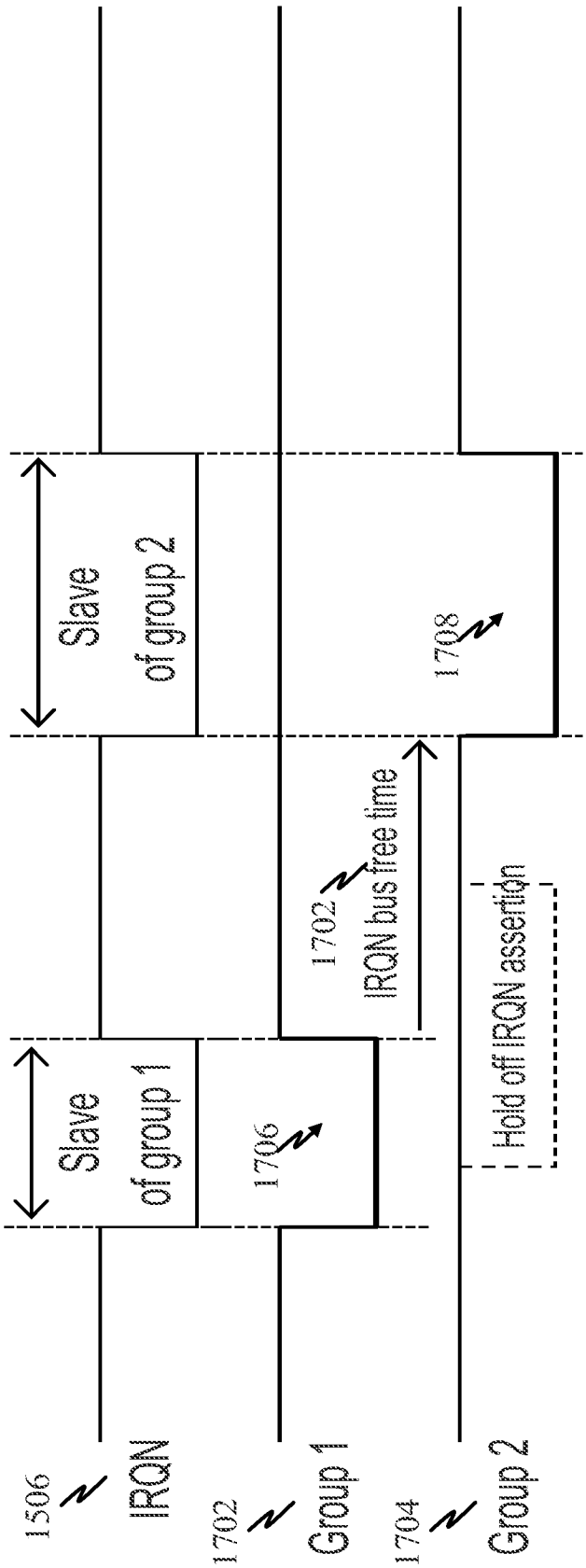


FIG 17

EXEMPLARY CCIe MODE IRQ ARBITRATION OVER A SHARED BUS

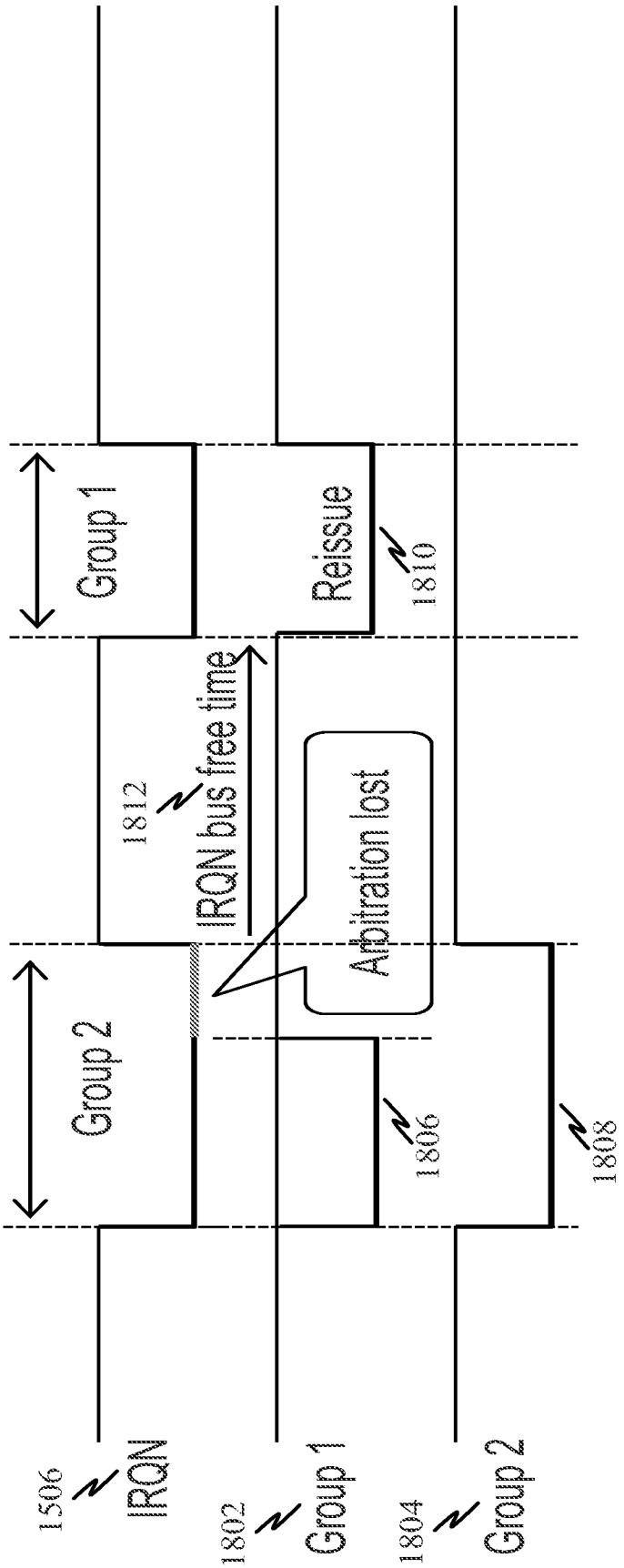
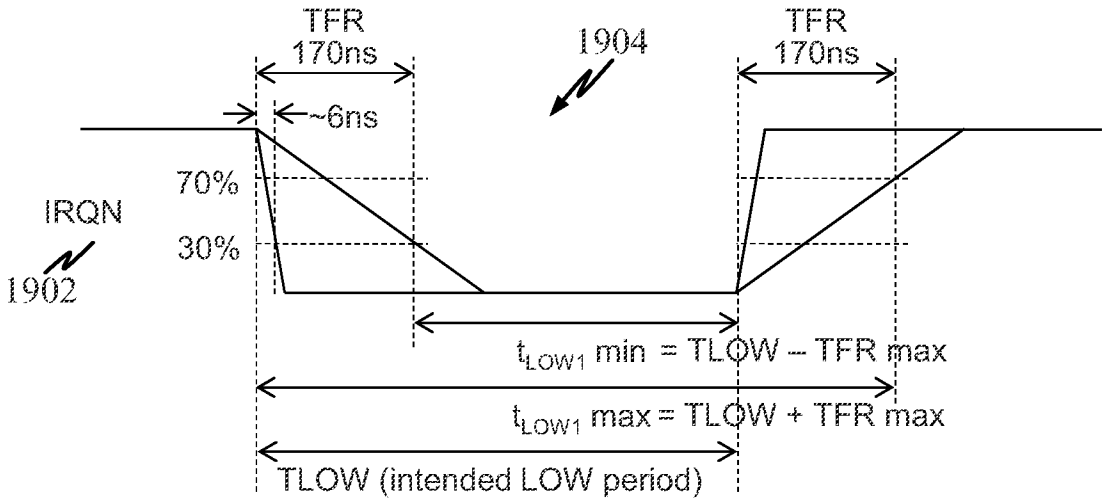


FIG. 18

EXEMPLARY INTERRUPT (IRQ) PERIOD



Term		Description
TLOW	:	Unit LOW driver period
TFR	:	Maximum signal fall/rise time from 100/0% to 30/70% of VDD
t_{LOW1}	:	1 unit receiver LOW period
t_{LOW2}	:	2 unit receiver LOW period
t_{LOWn}	:	n unit receiver LOW period

1. t_{LOW} min must be in a resolution enough for master to identify IRQ groups.

TLOW unit	min				max
1	TLOW - TFR	<	t_{LOW1}	<	TLOW + TFR
2	2TLOW - TFR	<	t_{LOW2}	<	2TLOW + TFR

t_{LOW1} max < t_{LOW2} min → 2TLOW - TFR > TLOW+TFR **→ 2TFR < TLOW**

2. After first slave asserts IRQN low, second slave may not detect IRQN low for TFR → t_{LOW} min has to include one TFR

→ 3TFR < TLOW

→ 2TFR < t_{LOW1}

Minimum TLOW & t_{LOW1} requirement

FIG. 19

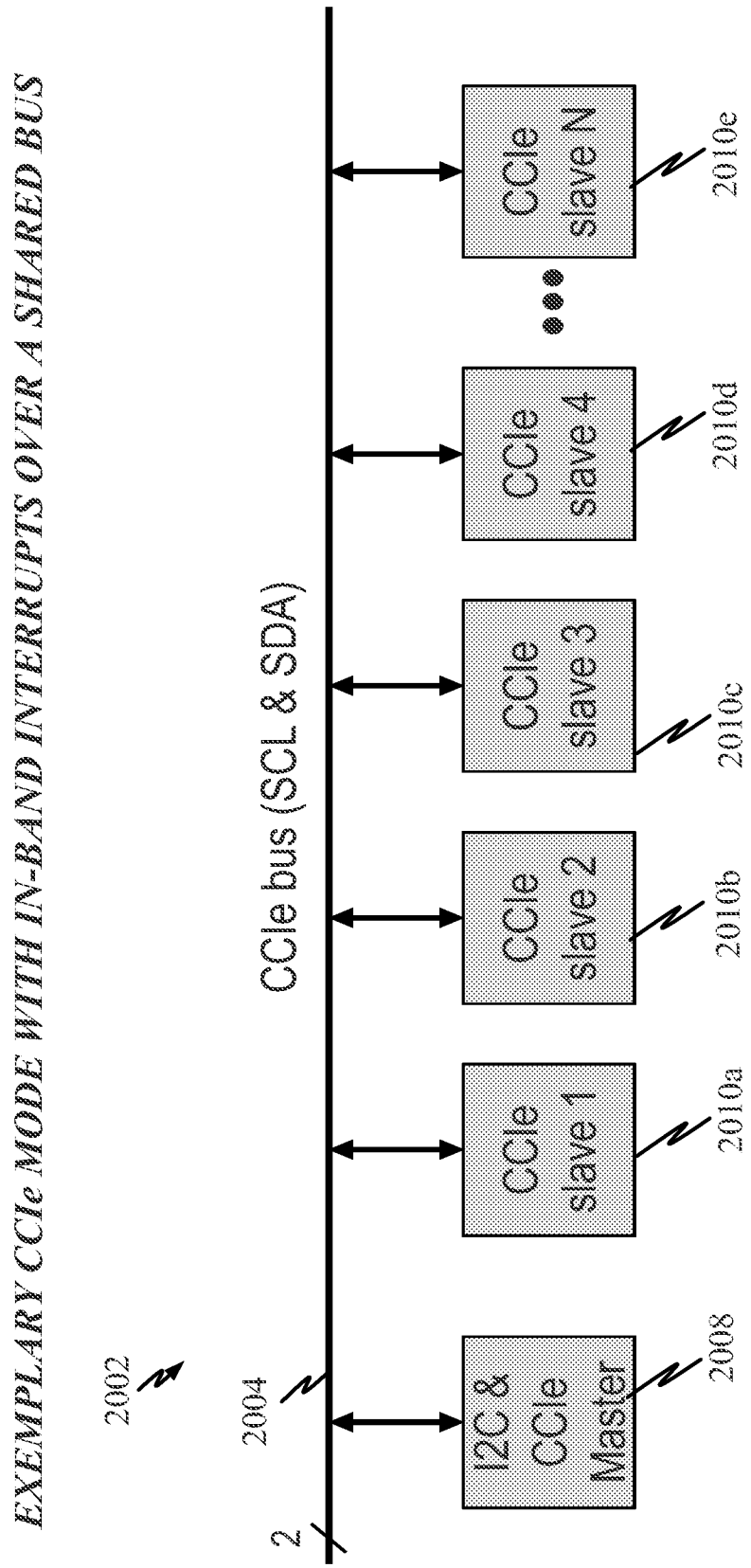

EXEMPLARY CCIe MODE WITH IN-BAND INTERRUPTS OVER A SHARED BUS
2002
2004
2
CCIe bus (SCL & SDA)
I2C & CCIe Master
2008
CCIe slave 1
2010a
CCIe slave 2
2010b
CCIe slave 3
2010c
CCIe slave 4
2010d
CCIe slave N
2010e

FIG. 20

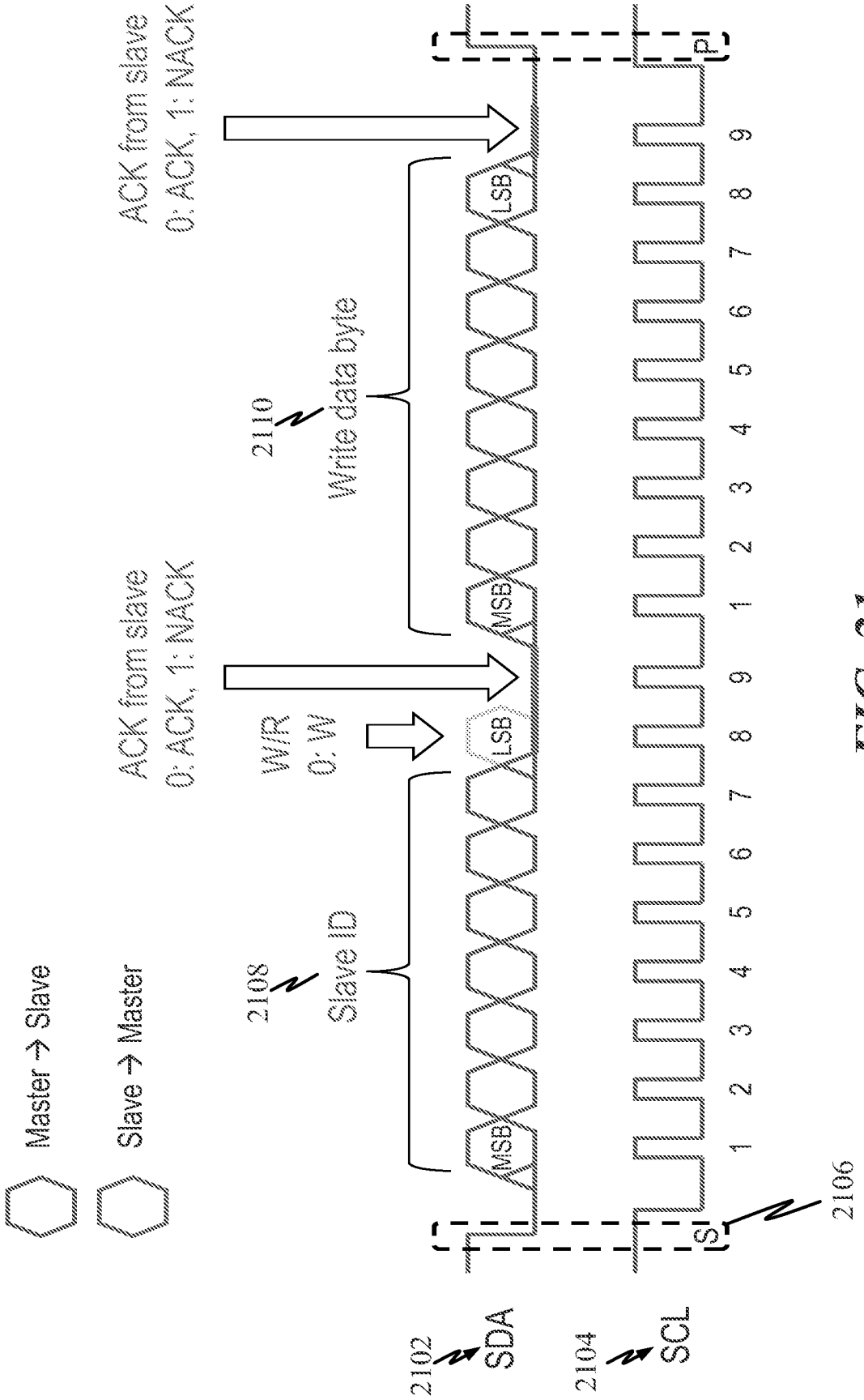

EXEMPLARY I2C-COMPATIBLE PROTOCOL TRANSMISSION
Master → Slave
Slave → Master
2102
SDA
2104
SCL
2106
S
P
2108
Slave ID
W/R
0: W
1: R
ACK from slave
0: ACK, 1: NACK
2110
Write data byte
ACK from slave
0: ACK, 1: NACK
MSB
LSB
1 2 3 4 5 6 7 8 9

FIG. 21

EXEMPLARY CCIe-COMPATIBLE PROTOCOL TRANSMISSION

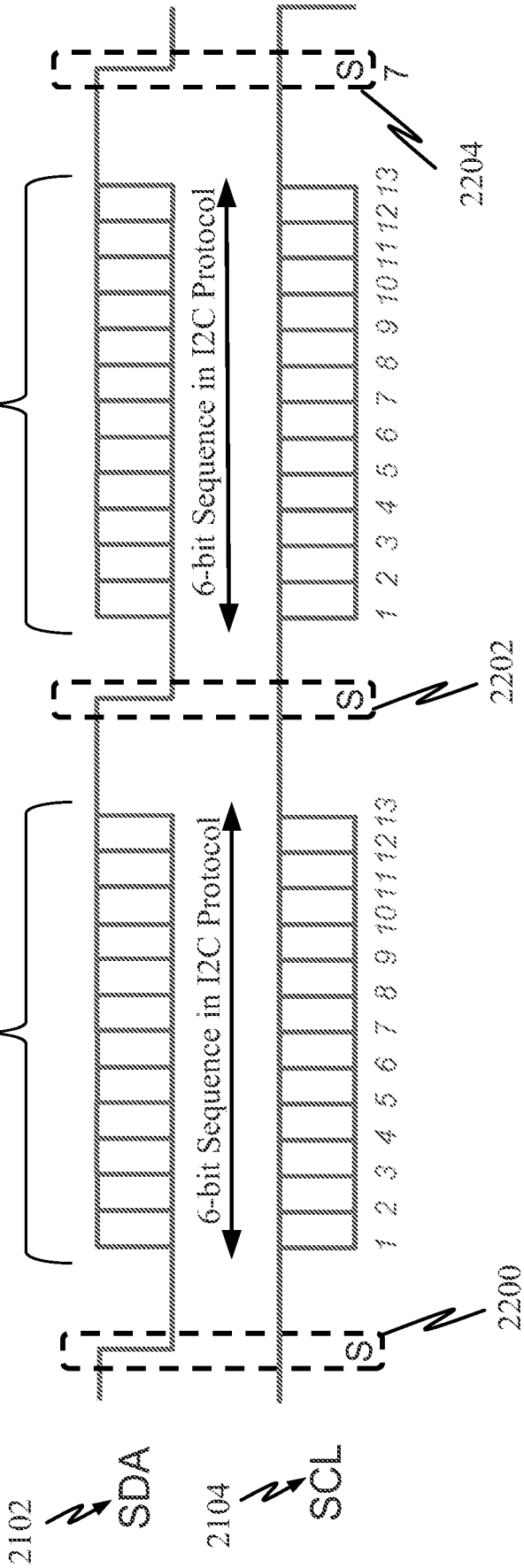


FIG. 22

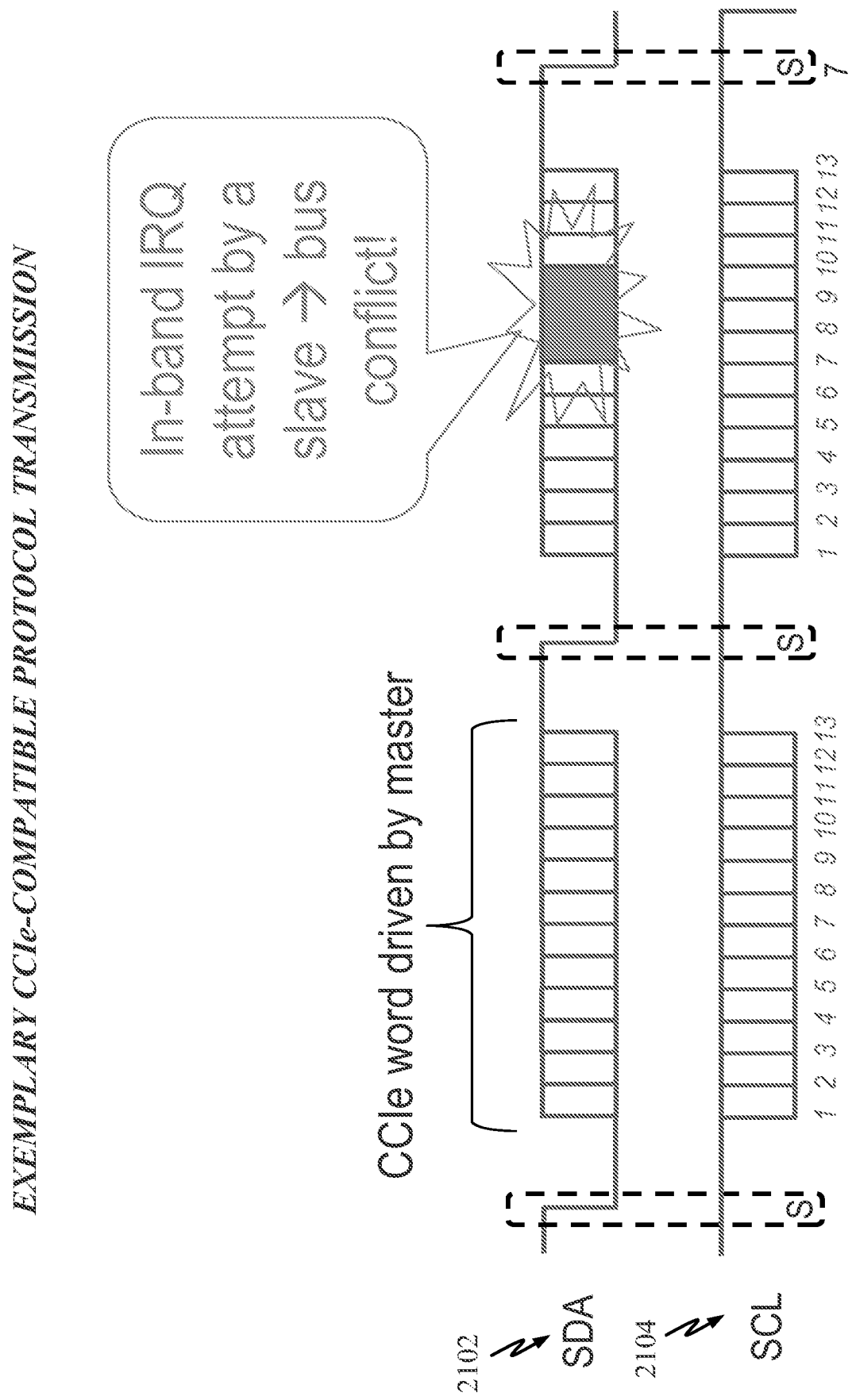

EXEMPLARY CCIe-COMPATIBLE PROTOCOL TRANSMISSION
In-band IRQ attempt by a slave -7 bus conflict!
CCIe word driven by master
S
7
1 2 3 4 5 6 7 8 9 10 11 12 13
2102
SDA
2104
SCL

FIG. 23

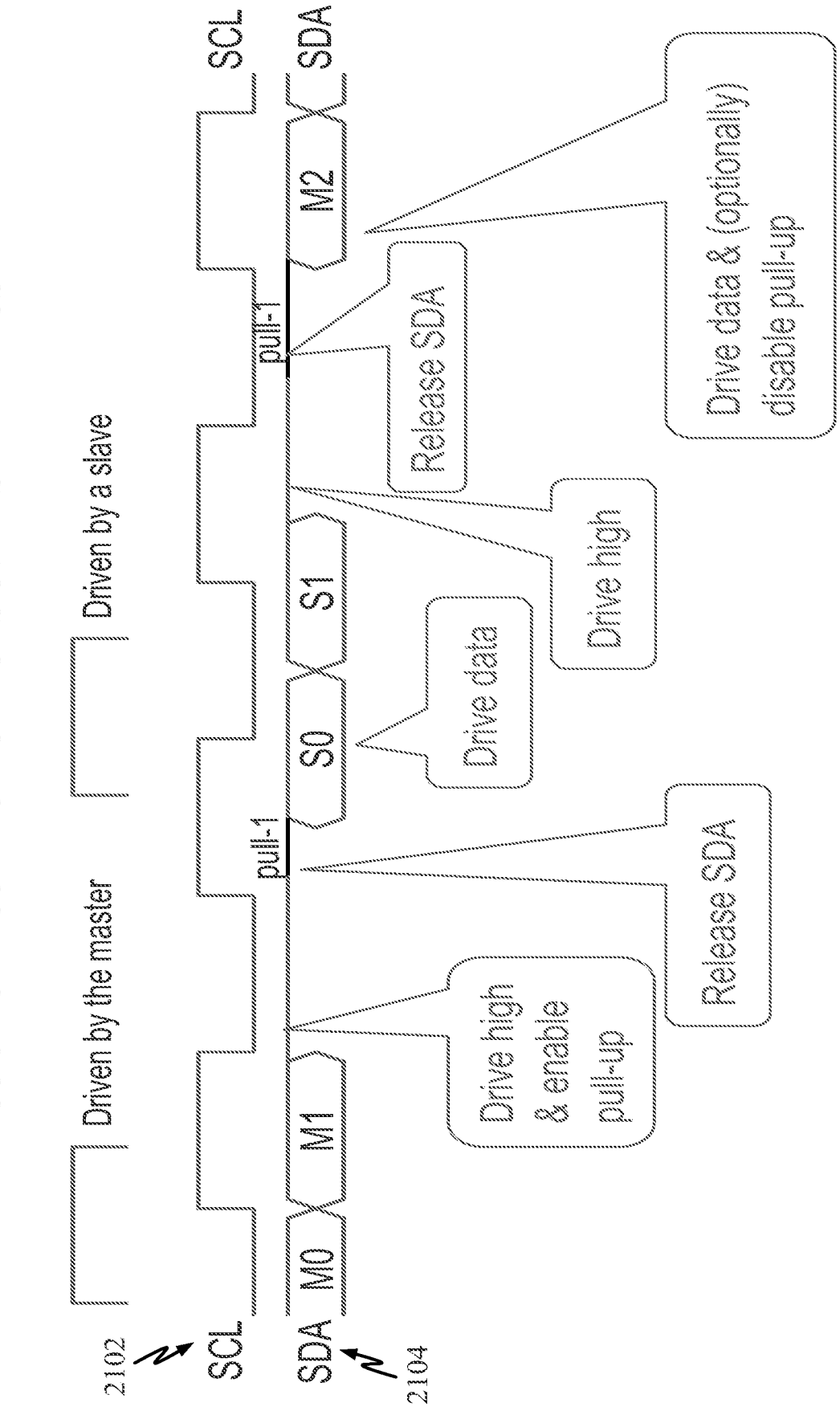

EXEMPLARY CCIe MODE COMMUNICATIONS OVER SHARED BUS
Driven by the master
Driven by a slave
2102
2104
SCL
SDA
M0
M1
S0
S1
M2
pull-1
pull-1
Drive high & enable pull-up
Release SDA
Drive data
Drive high
Release SDA
Drive data & (optionally) disable pull-up

FIG. 24

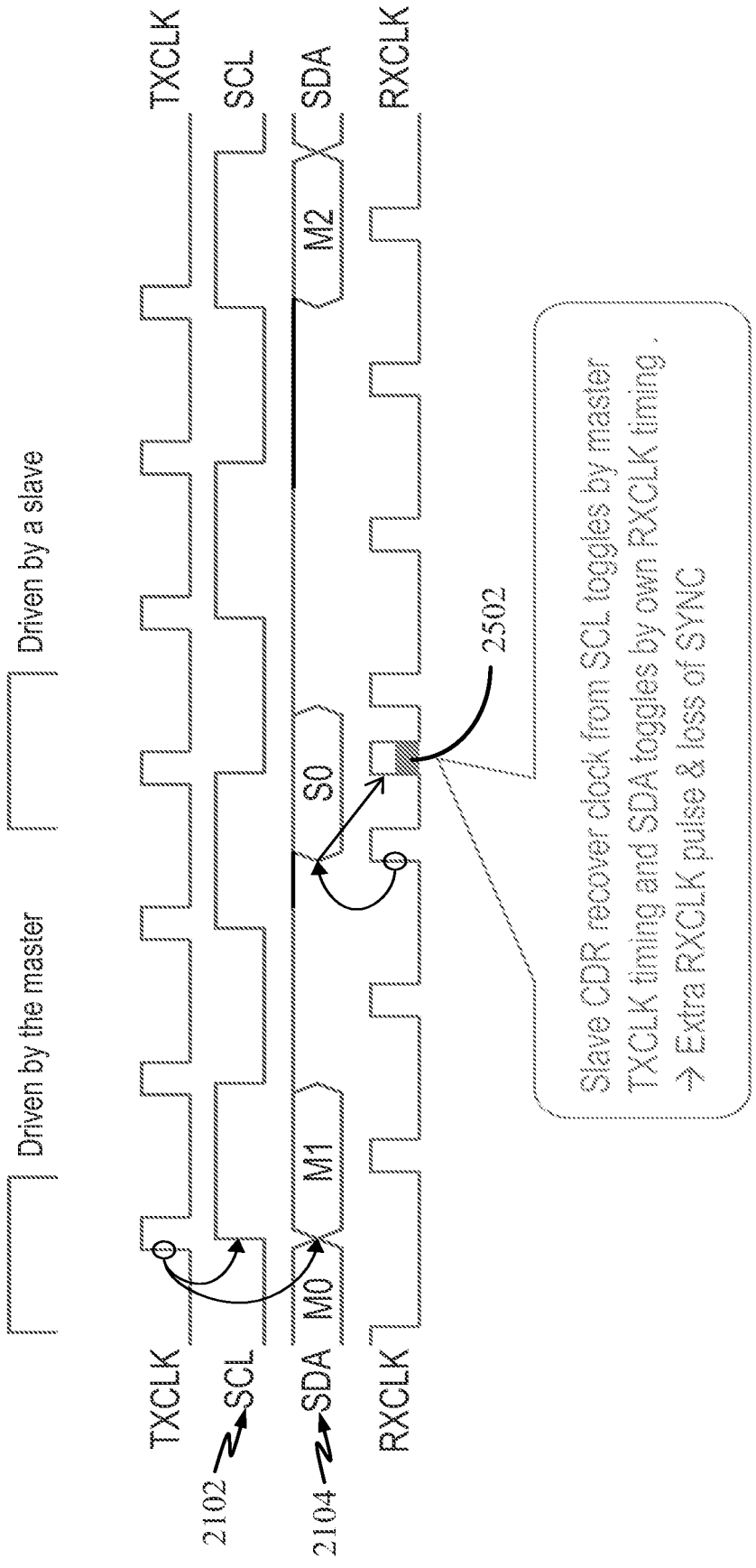

Driven by the master
Driven by a slave
TXCLK
SCL
SDA
RXCLK
M0
M1
M2
S0
2502
2102
2104
Slave CDR recover clock from SCL toggles by master
TXCLK timing and SDA toggles by own RXCLK timing.
→ Extra RXCLK pulse & loss of SYNC

FIG. 25

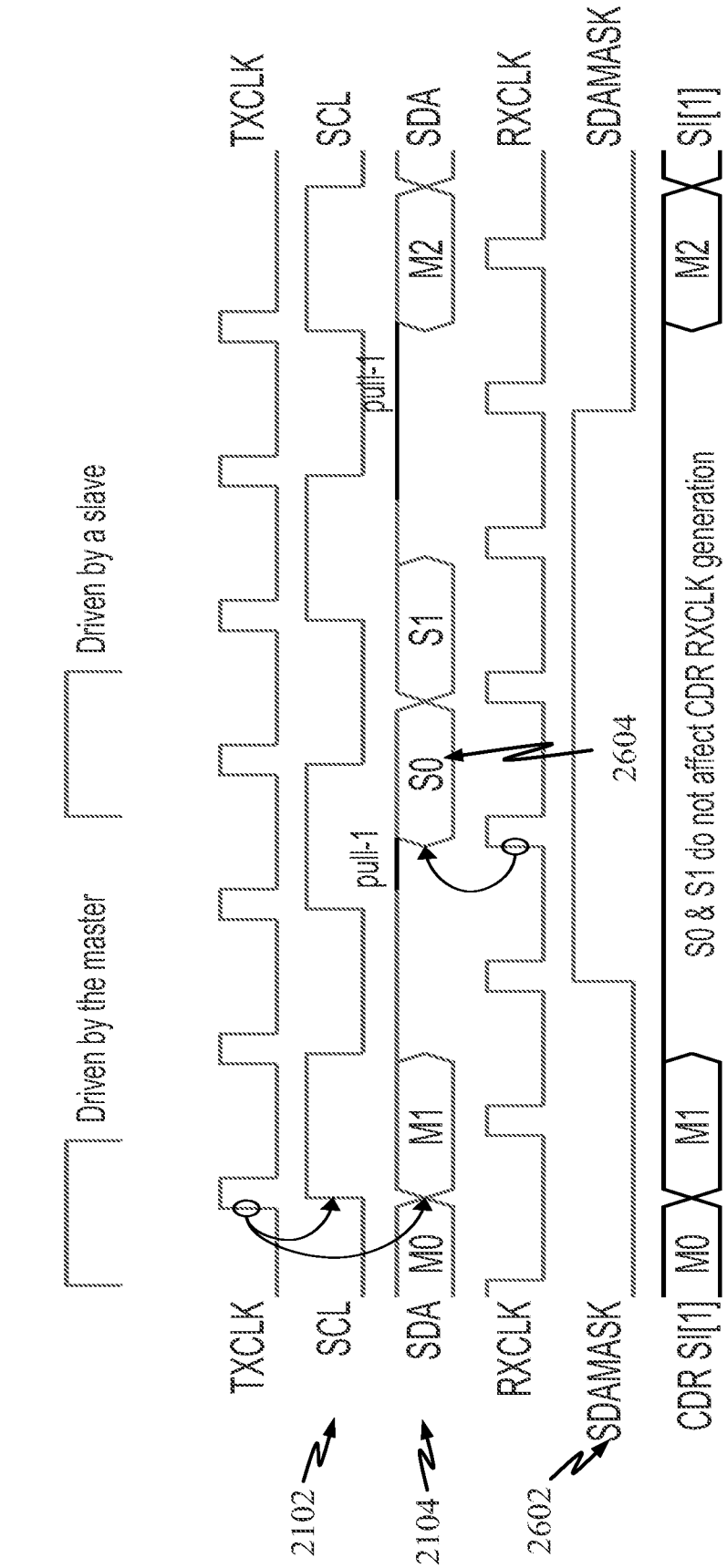

Driven by the master
Driven by a slave
TXCLK
SCL
SDA
RXCLK
SDAMASK
SI[1]
M0
M1
M2
S0
S1
pull-1
2604
CDR SI[1]
S0 & S1 do not affect CDR RXCLK generation
2102
2104
2602

FIG. 26

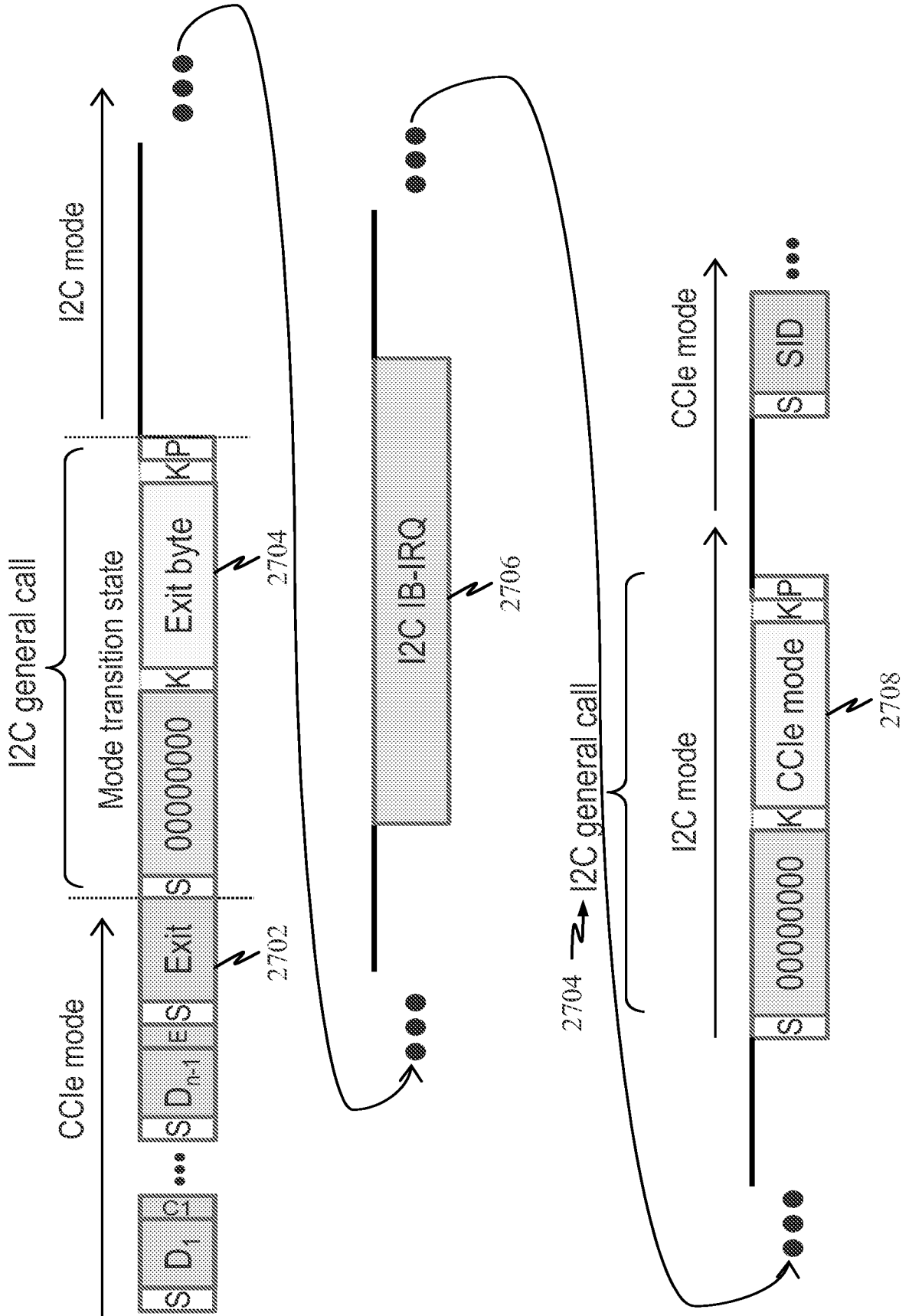

CCIe mode
I2C mode
I2C general call
Mode transition state
S
0000000
K
Exit byte
KP
2704
S
Exit
ES
D_{n-1}
S
D_1
C
2702
I2C IB-IRQ
2706
2704
I2C general call
I2C mode
CCIe mode
S
0000000
K
CCIe mode
KP
2708
S
SID

FIG. 27

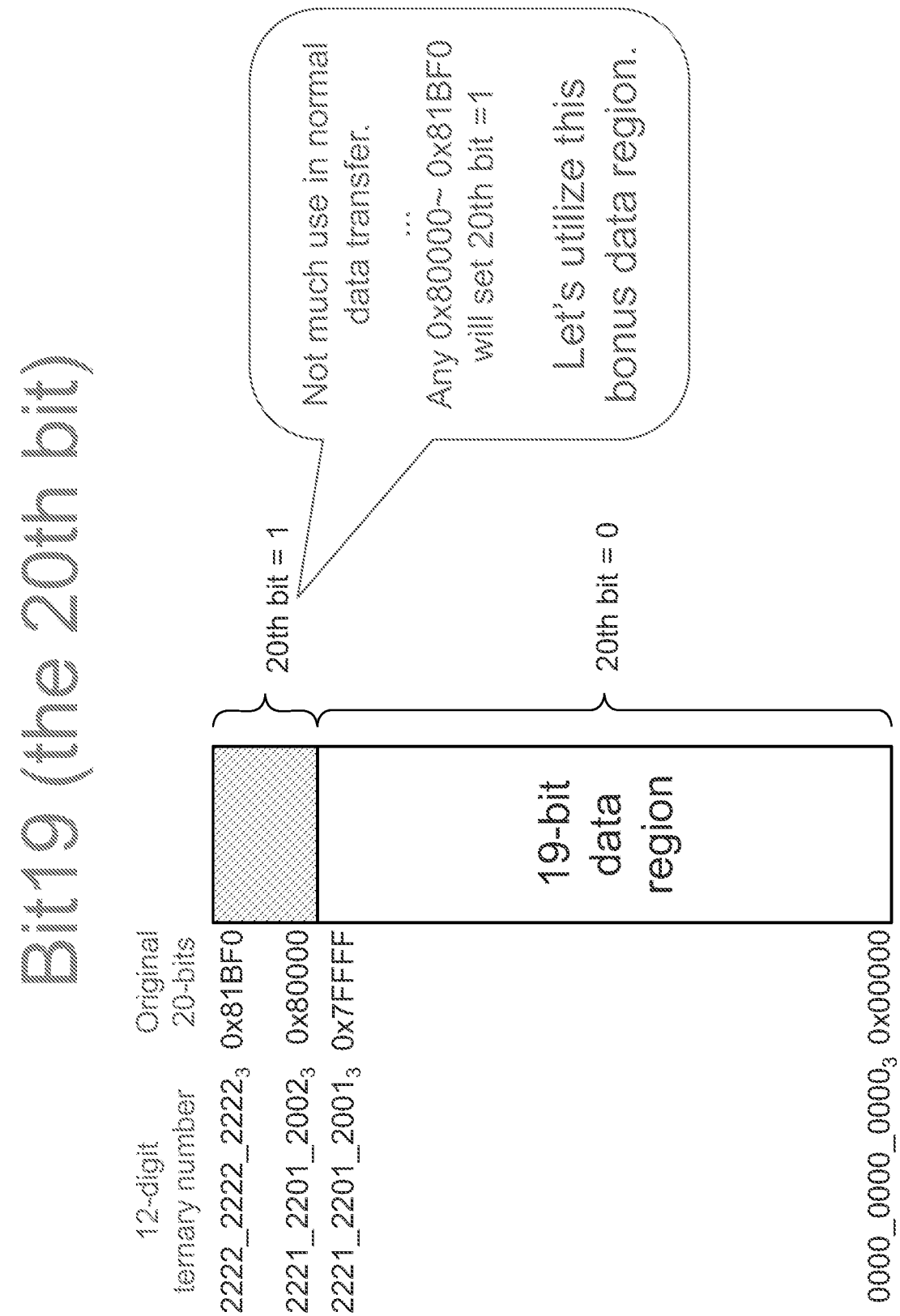

Bit19 (the 20th bit)
12-digit ternary number
Original 20-bits
2222_2222_2222$_3$ 0x81BF0
2221_2201_2002$_3$ 0x80000
2221_2201_2001$_3$ 0x7FFFF
0000_0000_0000$_3$ 0x00000
20th bit = 1
20th bit = 0
19-bit data region
Not much use in normal data transfer.
...
Any 0x80000~ 0x81BF0 will set 20th bit =1
Let's utilize this bonus data region.

FIG. 28

CCIe Bit19 mapping

Bit[19] = 1

Ternary	Bits[19:0]	Address	Write	Read	Bits[i] 19	12	11	9	8	7	6	5	4	3
$2222\_2222\_2222_3$	0x81BF0	*See FIG. 30*			1	1	1	1	1	1	1	?	?	?
0x31 (49) ↕														
$2222\_2222\_1012_3$	0x81BC0													
$2222\_2222\_1011_3$	0x81BBF	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	x	x	x
[6-bits] 0x40 (64) ↕														
$2222\_2221\_1211_3$	0x81B80													
$2222\_2221\_1210_3$	0x81B7F	Master handover	Reserved	Reserved	1	1	1	1	1	0	x	x	x	x
[7-bit] 0x80 (128) ↕														
$2222\_2220\_0002_3$	0x81B00													
$2222\_2220\_0001_3$	0x81AFF	Reserved	Reserved	Reserved	1	1	1	1	0	x	x	x	x	x
[8-bits] 0x100 (256) ↕														
$2222\_2202\_2121_3$	0x81A00													
$2222\_2202\_2120_3$	0x819FF	Reserved	Slave to slave grant	Slave to slave request	1	1	1	0	x	x	x	x	x	x
[9-bits] 0x200 (512) ↕														
$2222\_2112\_1122_3$	0x81800													
$2222\_2112\_1121_3$	0x817FF	CCIe register address	Reserved	Master bus request	1	1	0	x	x	x	x	x	x	x
[11-bits] 0x800 (2048) ↕														
$2222\_1121\_0210_3$	0x81000													
$2222\_1121\_0202_3$	0x80FFF	8-bit CHK	8-bit CHK	8-bit CHK	1	0	x	x	x	x	x	x	x	x
[12-bits] 0x1000 (4096) ↕														
$2221\_2201\_2002_3$	0x80000													
$2221\_2201\_2001_3$	0x7FFFF	19-bit data region			0	x	x	x	x	x	x	x	x	x
0x80000 (524288) ↕														
$0000\_0000\_0000_3$	0x00000													

FIG. 29

Ternary	Bits[19:0]	Address	Write	Read	Bits[j]												
					19	12	11	9	8	7	6	5	4	3	2	1	0
2222_2222_2222$_3$	0x81BF0	♫SY-		Prohibited	1	1	1	1	1	1	1	1	1	0	0	0	0
2222_2222_2221$_3$	0x81BEF 0x2A (42) ↕ 2222_2222_1102$_3$ 0x81BC6	Prohibited															
2222_2222_1101$_3$	0x81BC5	Heartbeat ♥ / -NC♫		Prohibited	1	1	1	1	1	1	1	0	0	0	1	0	1
2222_2222_1100$_3$	0x81BC4 0x2A (42) ↕ 2222_2222_1000$_3$ 0x81BBB	Prohibited															
2222_2222_0222$_3$	0x81BBA	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	1	0
2222_2222_0221$_3$	0x81BB9	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	0	1
2222_2222_0220$_3$	0x81BB8	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	0	0
2222_2222_0212$_3$	0x81BB7 [3-bits] 0x8 (8) ↕ 2222_2222_0121$_3$ 0x81BB0	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	0	x	x	x
2222_2222_0120$_3$	0x81BAF [4-bits] 0x10 (16) ↕ 2222_2222_0000$_3$ 0x81BA0	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	0	x	x	x	x
2222_2221_2222$_3$	0x81B9F 0x10 (16) ↕ 2222_2221_2102$_3$ 0x81B90	Prohibited															
2222_2221_2101$_3$	0x81B8F	Prohibited	SID scan resp	Prohibited	1	1	1	1	1	1	0	0	0	1	0	0	0
2222_2222_2100$_3$	0x81B8E 0xA (10) ↕ 2222_2221_2000$_3$ 0x81B85	Prohibited															
2222_2221_1222$_3$	0x81B84	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	0	0	0	1	0	0
2222_2221_1221$_3$	0x81B83 [2-bits] 0x4 (4) ↕ 2222_2221_1211$_3$ 0x81B80	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	0	0	0	0	x	x

FIG. 30

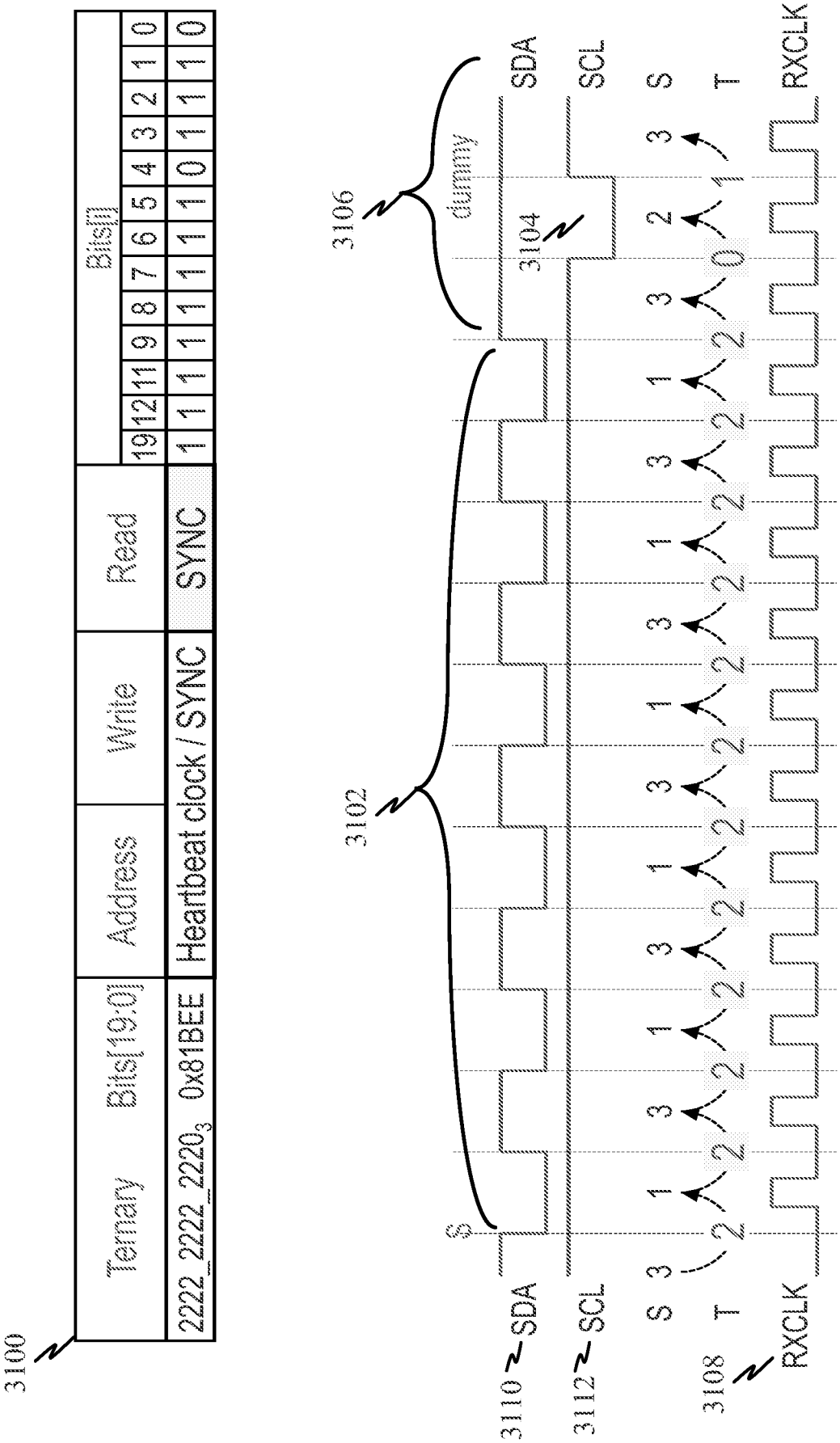

3100
Ternary
Bits[19:0]
Address
Write
Read
Bits[]
19 12 11 9 8 7 6 5 4 3 2 1 0
2222_2222_2220$_3$ 0x81BEE
Heartbeat clock / SYNC
SYNC
1 1 1 1 1 1 1 1 1 1 1 1 0
3102
3106
dummy
3104
SDA
SCL
S
3110 SDA
3112 SCL
3108 RXCLK
RXCLK

FIG. 31

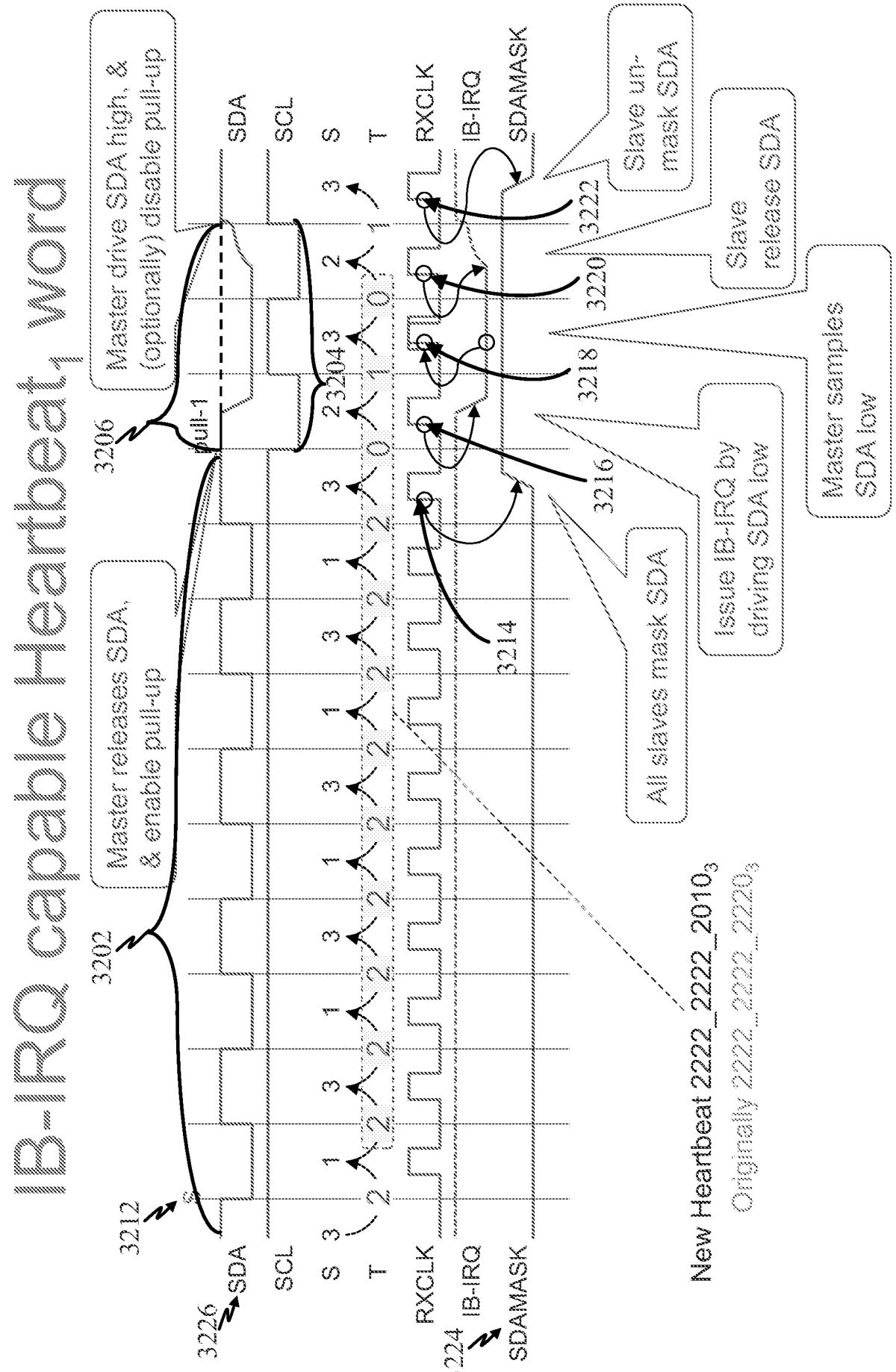

IB-IRQ capable Heartbeat1 word
Master drive SDA high, & (optionally) disable pull-up
Master releases SDA, & enable pull-up
3202
3206
3212
3226
3224
SDA
SCL
S
T
RXCLK
IB-IRQ
SDAMASK
pull-1
3216
3214
3218
3220
3222
All slaves mask SDA
Issue IB-IRQ by driving SDA low
Slave release SDA
Slave un-mask SDA
Master samples SDA low
New Heartbeat 2222_2222_2010$_3$
Originally 2222_2222_2220$_3$

FIG. 32

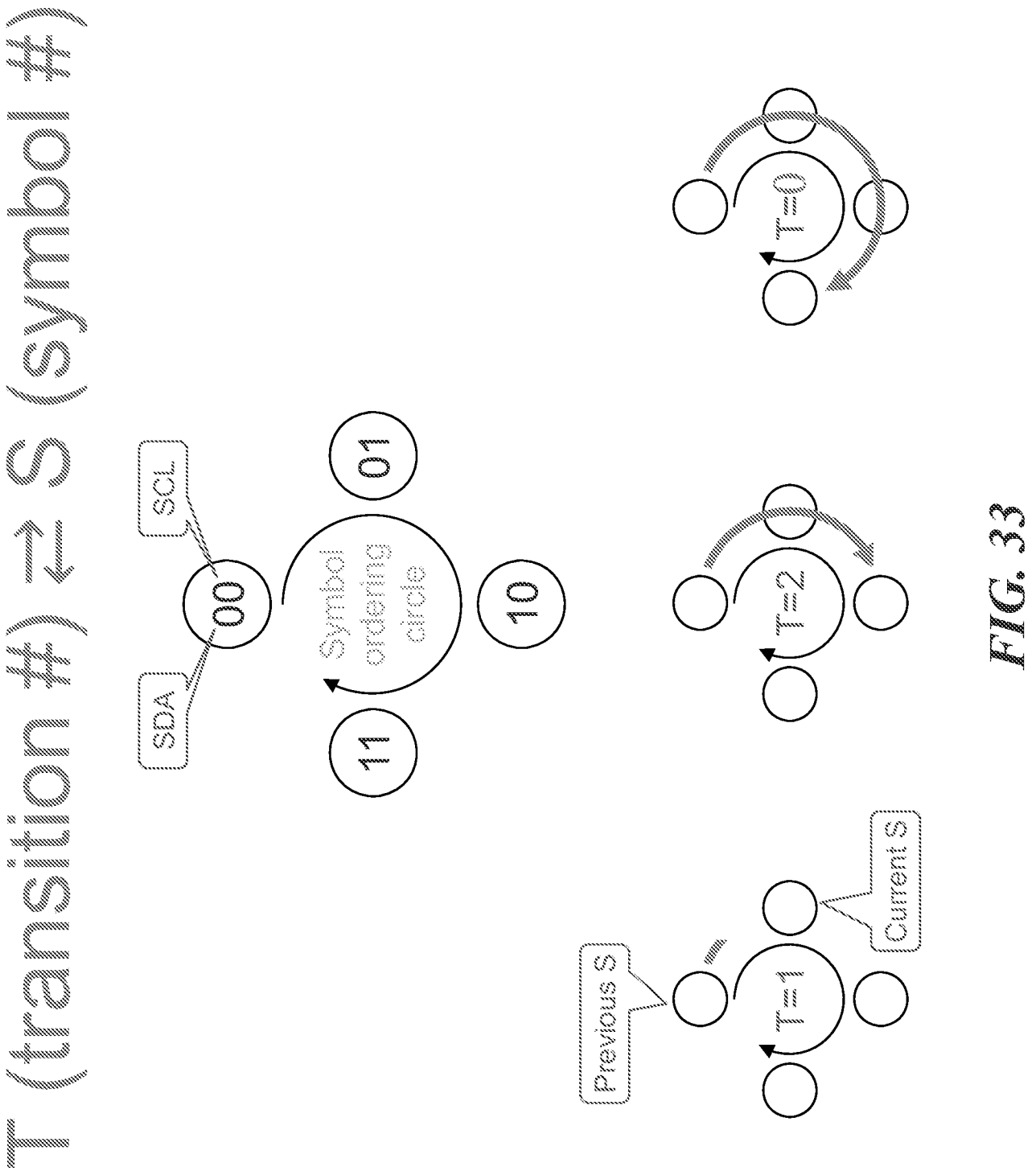

T (transition #) → S (symbol #)
SDA
SCL
00
01
10
11
Symbol ordering circle
T=0
T=1
T=2
Previous S
Current S

FIG. 33

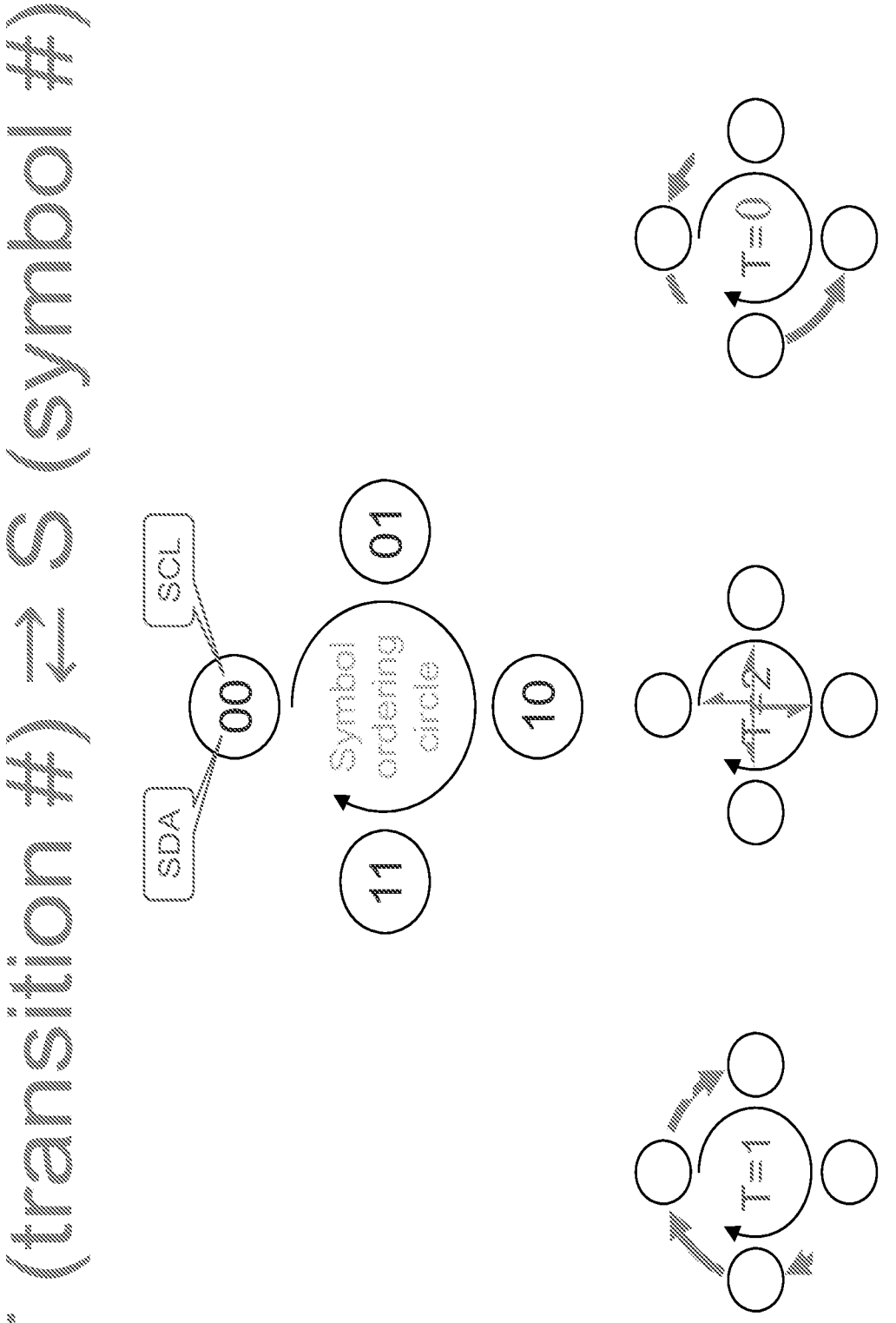

T (transition #) ⇄ S (symbol #)
SDA
SCL
00
01
10
11
Symbol ordering circle
T=1
T=2
T=0
00 01 10 11
⇩ ⇩ ⇩ ⇩
01 10 11 00
00 10 01 11
⇩ ⇩ ⇩ ⇩
10 00 11 01
00 11 10 01
⇩ ⇩ ⇩ ⇩
11 10 01 00
No toggle on SCL

FIG. 34

T=2 while SDA is masked is prohibited.

FIG. 35

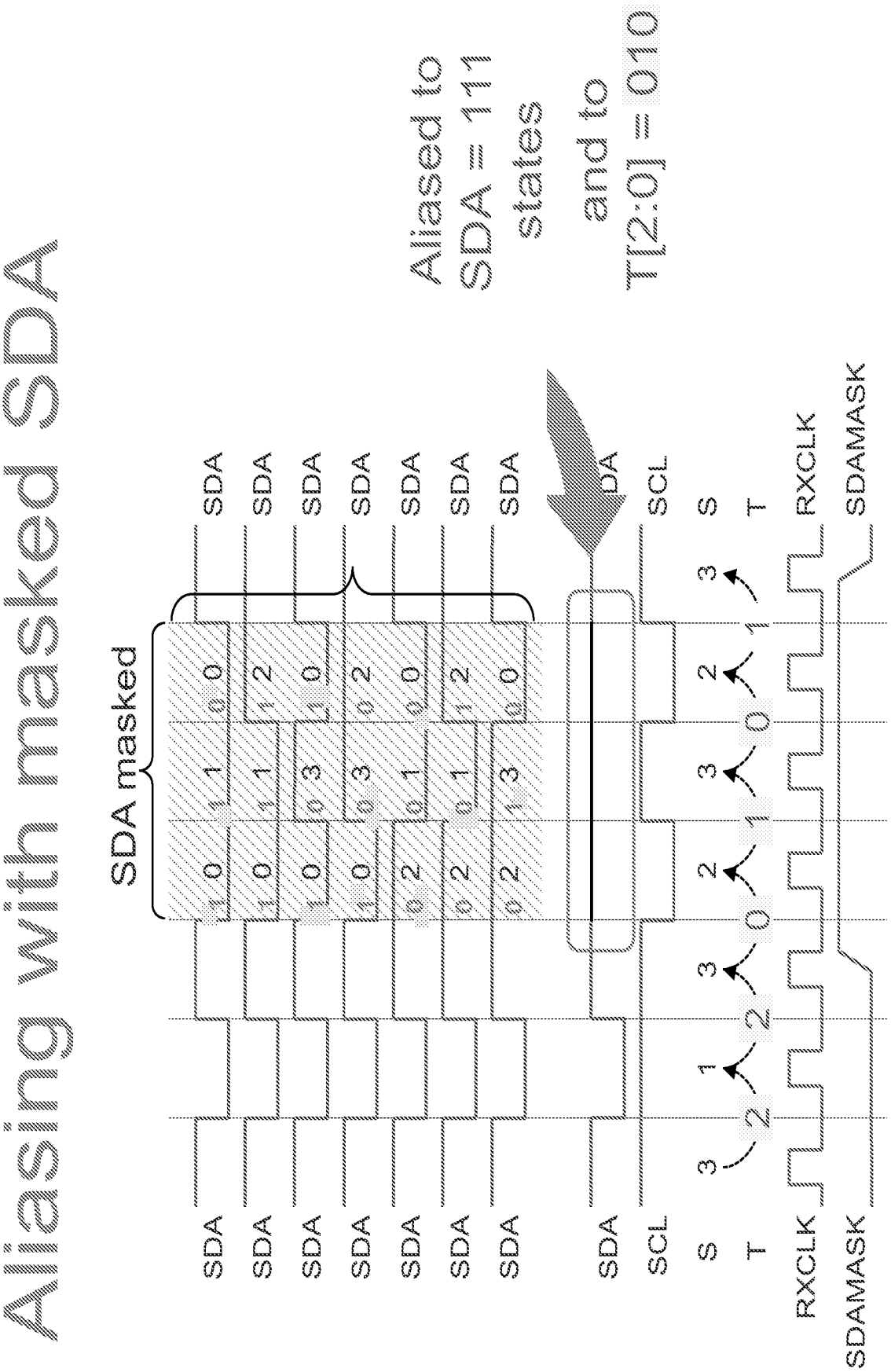
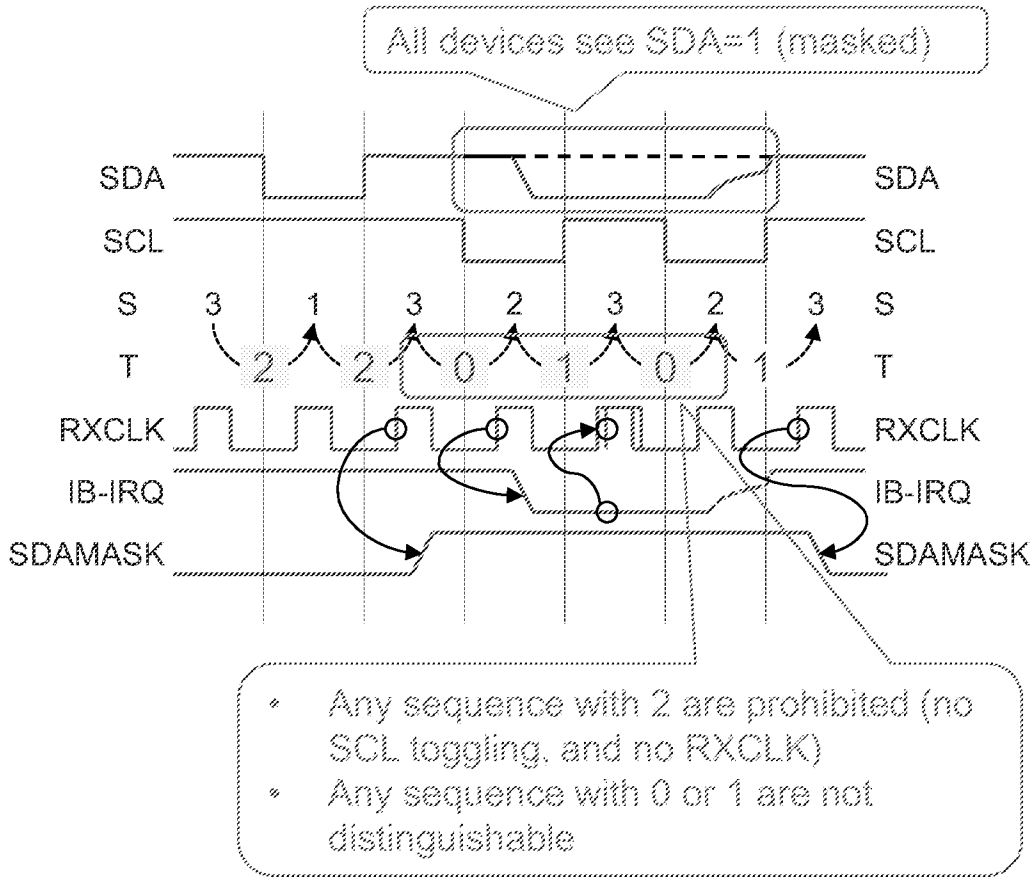

Aliasing with masked SDA
SDA masked
Aliased to
SDA = 111
states
and to
T[2:0] = 010
SDA
SCL
S
T
RXCLK
SDAMASK

FIG. 36

$Heartbeat_1$: sacrifice

Ternary	Bits[19:0]	
$2222\_2222\_2222_3$	0x81BF0	Prohibited
(9) ↕		
$2222\_2222\_2200_3$	0x81BE8	
$2222\_2222\_2122_3$	0x81BE7	Prohibited
(4) ↕		
$2222\_2222\_2112_3$	0x81BE4	
$2222\_2222\_2111_3$	0x81BE3	0x81BD9 alias
$2222\_2222\_2110_3$	0x81BE2	
$2222\_2222\_2102_3$	0x81BE1	Prohibited
$2222\_2222\_2101_3$	0x81BE0	0x81BD9 alias
$2222\_2222\_2100_3$	0x81BDF	
$2222\_2222\_2022_3$	0x81BDE	Prohibited
(4) ↕		
$2222\_2222\_2012_3$	0x81BDB	
$2222\_2222\_2011_3$	0x81BDA	0x81BD9 alias
$2222\_2222\_2010_3$	0x81BD9	
$2222\_2222\_2002_3$	0x81BD8	Prohibited
$2222\_2222\_2001_3$	0x81BD7	0x81BD9 alias
$2222\_2222\_2000_3$	0x81BD6	



- $Heartbeat_1$ occupies 0x81BD6~0x81BF0, 27 address
- Trashing $2222\_2222\_2222_3$ which is best (only?) word for word synchronization makes word synchronization mechanism hard (or impossible?) to design.

FIG. 37

CCIe Bit19 mapping cont.

Ternary	Bits[19:0]	Address	Write	Read	Bits[i] 19	12	11	9	8	7	6	5	4	3	2	1	0
2222_2222_2222$_3$	0x81BF0	2 in T[2:0] → Prohibited Other T[2:0] → Aliased to 0x81BD9			1	1	1	1	1	1	1	0	1	1	0	0	1
0x11 (17) ↕																	
2222_2222_2101$_3$	0x81BE0																
2222_2222_2222$_3$	0x81BDF																
0x6 (6) ↕																	
2222_2222_2011$_3$	0x81BDA																
2222_2222_2010$_3$	0x81BD9	Heartbeat/IBIRQ		Prohibited													
2222_2222_2002$_3$	0x81BD8	2 in T[2:0] → Prohibited Other T[2:0] → Aliased to 0x81BD9															
2222_2222_2001$_3$	0x81BD7																
2222_2222_2000$_3$	0x81BD6																
2222_2222_1222$_3$	0x81BD5	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	1	0	1
2222_2222_1221$_3$	0x81BD4	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	1	0	0
2222_2222_1220$_3$	0x81BD3	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	0	1	1
2222_2222_1212$_3$	0x81BD2	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	0	1	0
2222_2222_1211$_3$	0x81BD1	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	0	0	1
2222_2222_1210$_3$	0x81BD0	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	1	0	0	0	0
2222_2222_1202$_3$	0x81BCF	Reserved	Reserved	Reserved	1	1	1	1	1	1	1	0	0	x	x	x	x
0x16 (22) ↕																	
2222_2222_1012$_3$	0x81BC0																

FIG. 38

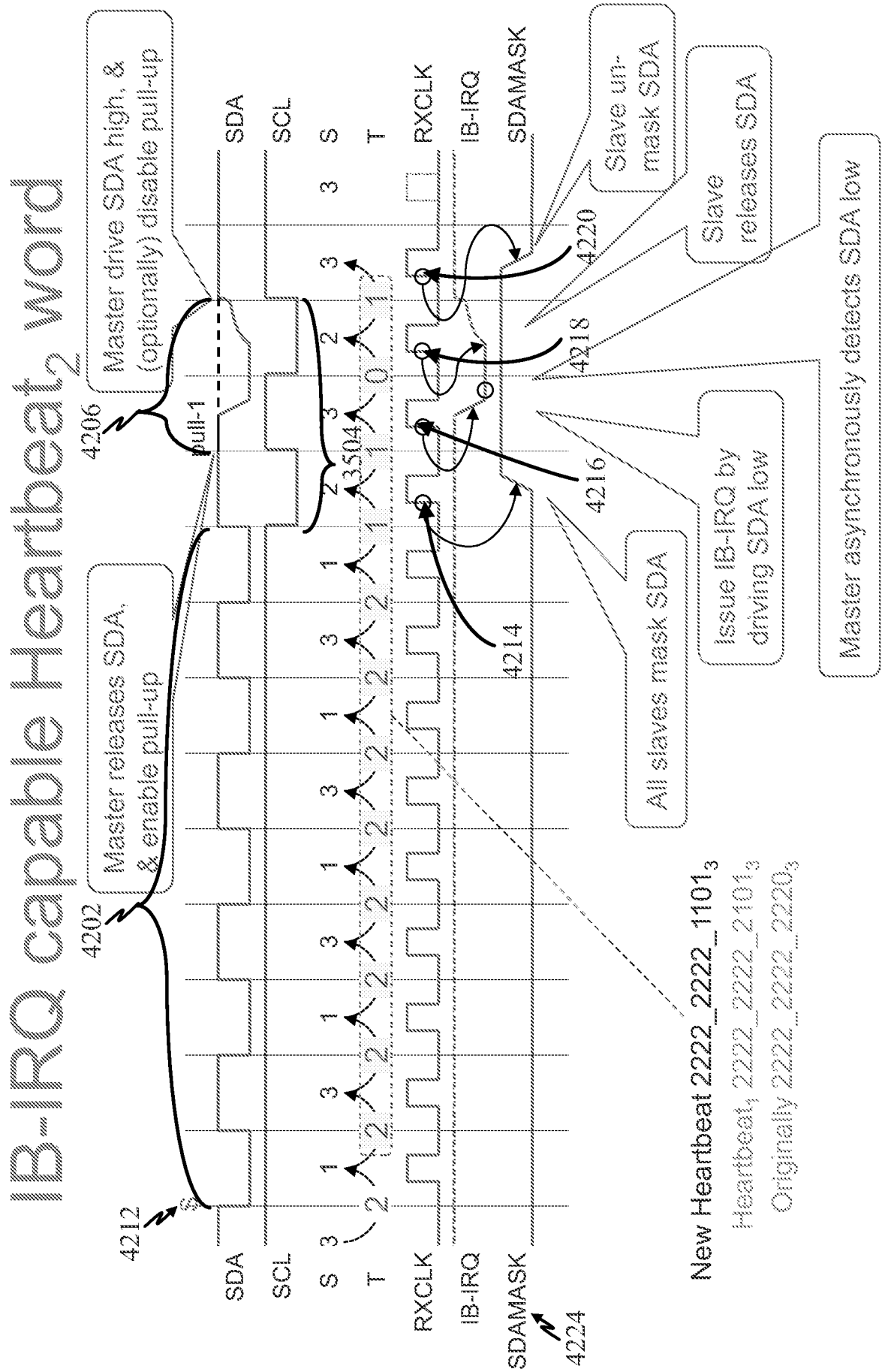

IB-IRQ capable Heartbeat2 word
Master drive SDA high, & (optionally) disable pull-up
Master releases SDA, & enable pull-up
Slave un-mask SDA
Slave releases SDA
Issue IB-IRQ by driving SDA low
All slaves mask SDA
Master asynchronously detects SDA low
SDA
SCL
S
T
RXCLK
IB-IRQ
SDAMASK
4202
4206
4212
4214
4216
4218
4220
4224
pull-1
New Heartbeat 2222_2222_1101₃
Heartbeat₁ 2222_2222_2101₃
Originally 2222_2222_2220₃

FIG. 39

T=2 while SDA is masked is prohibited.

FIG. 40

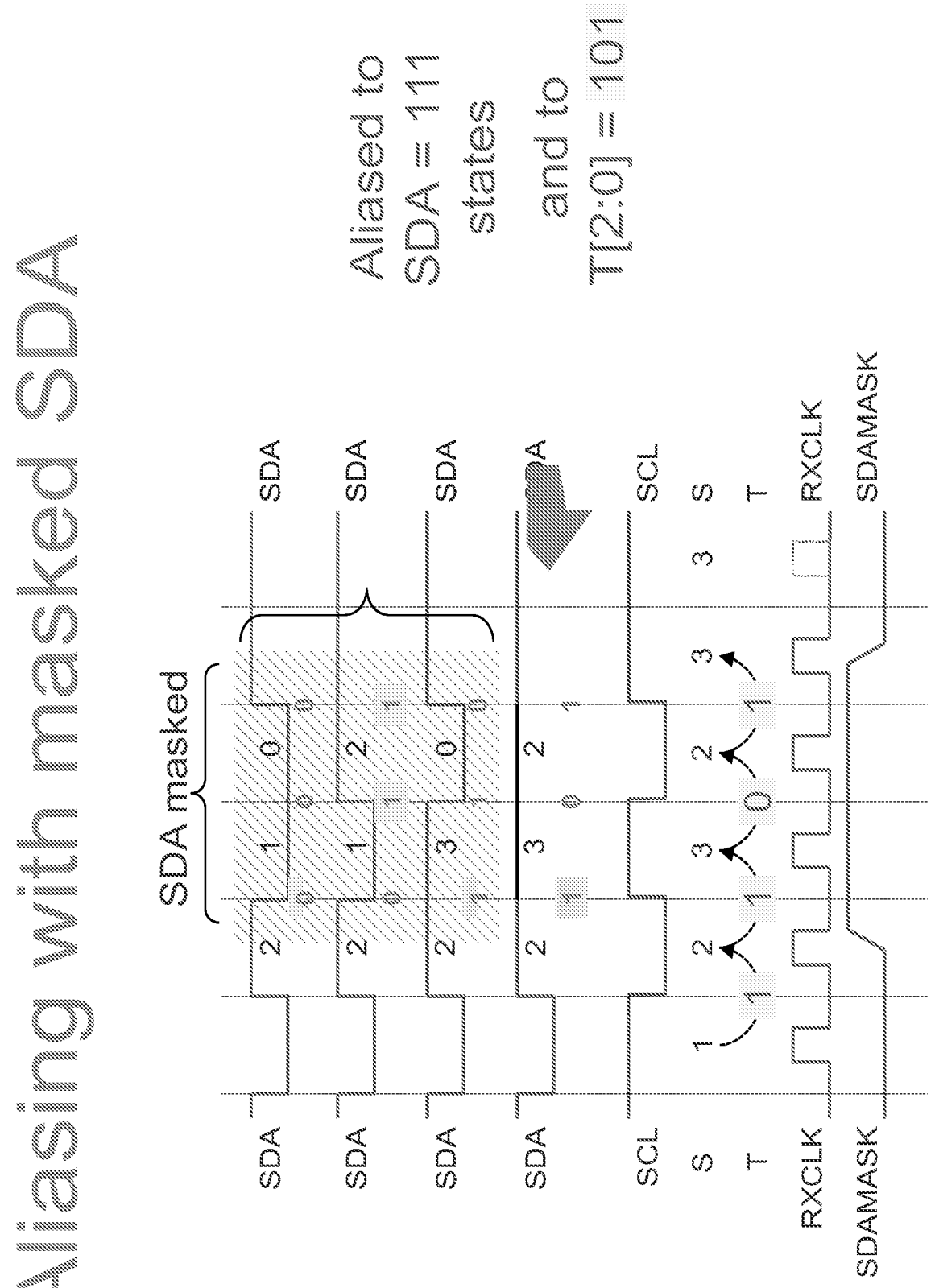
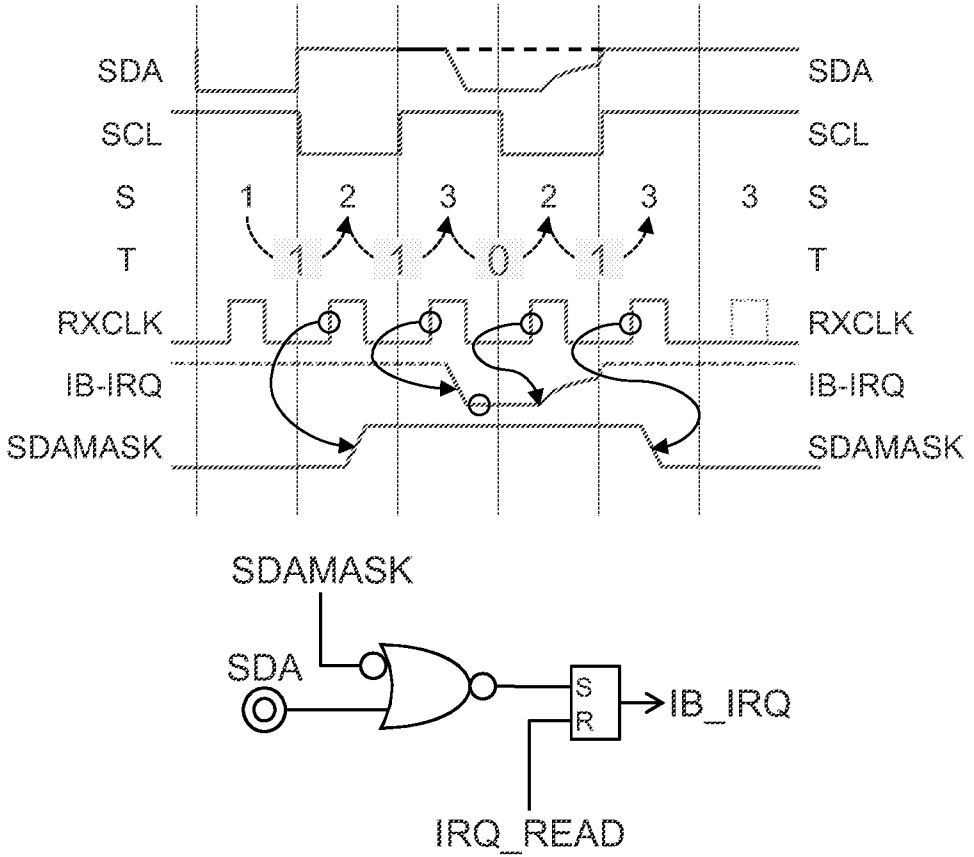

Aliasing with masked SDA
SDA masked
Aliased to
SDA = 111
states
and to
T[2:0] = 101
SDA
SCL
S
T
RXCLK
SDAMASK

FIG. 41

Heartbeat$_2$: sacrifice

Ternary	Bits[19:0]	
$2222\_2222\_1222_3$	0x81BD5	Prohibited
	(9) ↕	
$2222\_2222\_1200_3$	0x81BCD	
$2222\_2222\_1122_3$	0x81BCC	Prohibited
	(4) ↕	
$2222\_2222\_1112_3$	0x81BC9	
$2222\_2222\_1111_3$	0x81BC8	0x81BD9 alias
$2222\_2222\_1110_3$	0x81BC7	
$2222\_2222\_1102_3$	0x81BC6	Prohibited
$2222\_2222\_1101_3$	0x81BC5	0x81BD9 alias
$2222\_2222\_1100_3$	0x81BC4	
$2222\_2222\_1022_3$	0x81BC3	Prohibited
	(4) ↕	
$2222\_2222\_1012_3$	0x81BC0	
$2222\_2222\_1011_3$	0x81BBF	0x81BD9 alias
$2222\_2222\_1010_3$	0x81BBE	
$2222\_2222\_1002_3$	0x81BBD	Prohibited
$2222\_2222\_1001_3$	0x81BBC	0x81BD9 alias
$2222\_2222\_1000_3$	0x81BBB	



- Heartbeat occupies 0x81BBB~0x81BD5, 27 address
- Needs special asynchronous logic to sample IB-IRQ

- $2222\_2222\_2222_3$ is still available for SYNC → maybe better solution than Heartbeat1

FIG. 42

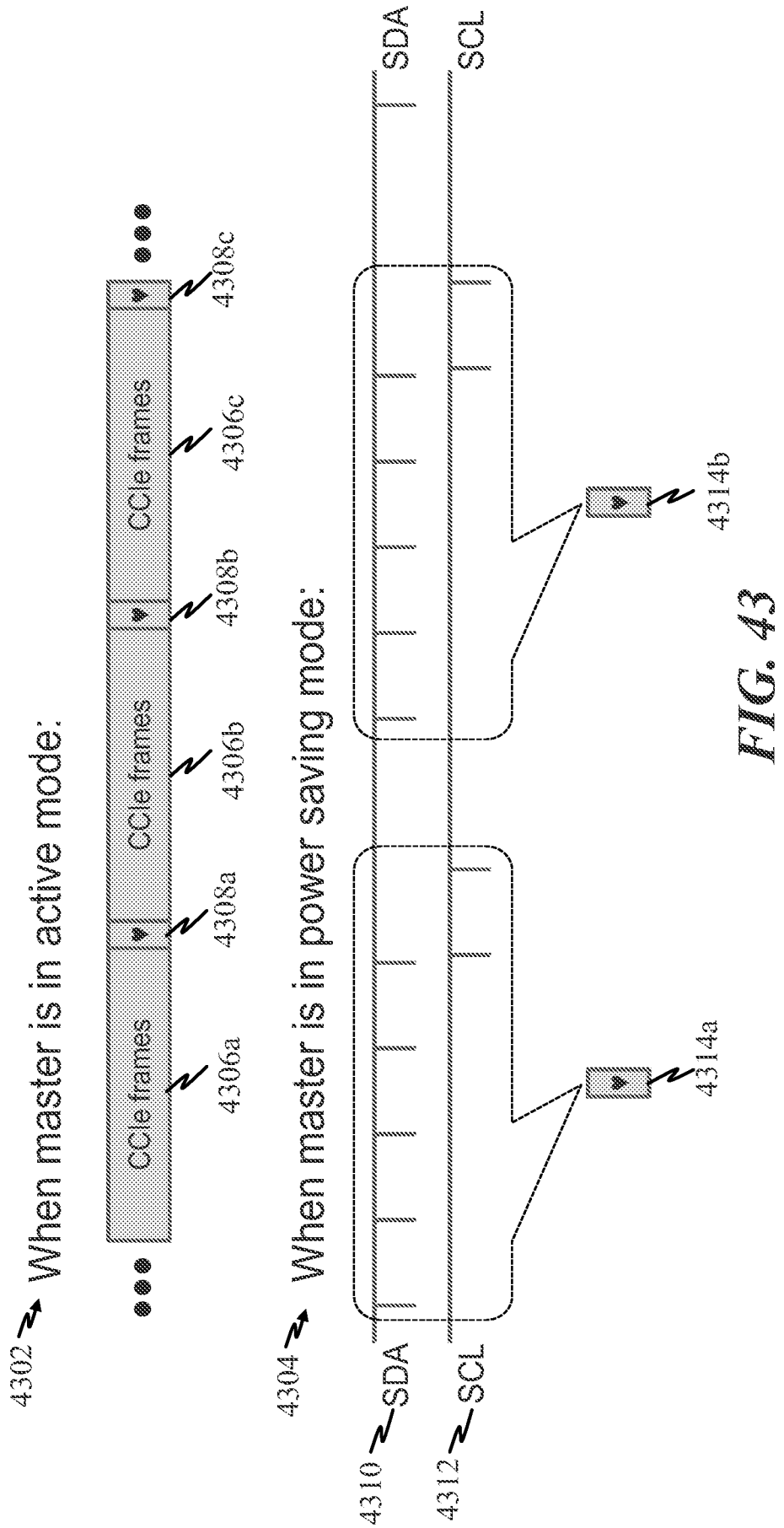

IB-IRQ Heartbeat usage
Minimal protocol overhead
4302
When master is in active mode:
CCIe frames
CCIe frames
CCIe frames
4306a
4308a
4306b
4308b
4306c
4308c
4304
When master is in power saving mode:
SDA
SCL
4310
SDA
4312
SCL
4314a
4314b

FIG. 43

SYNC: "SY-"+Heartbeat2("-NC")
All devices on the bus can synchronize to CCle word boundary by SYNC word followed by Heartbeat2
When master is in active mode:
CCle frames
CCle frames
CCle frames
S
SY-: 2_2222_2222_2222_2
-NC: 2_2222_2222_1101
S
Dummy
SYNC: 2_2222_2222_2222_2_2_2222_2222_1101
23 "2"s followed by "1101"

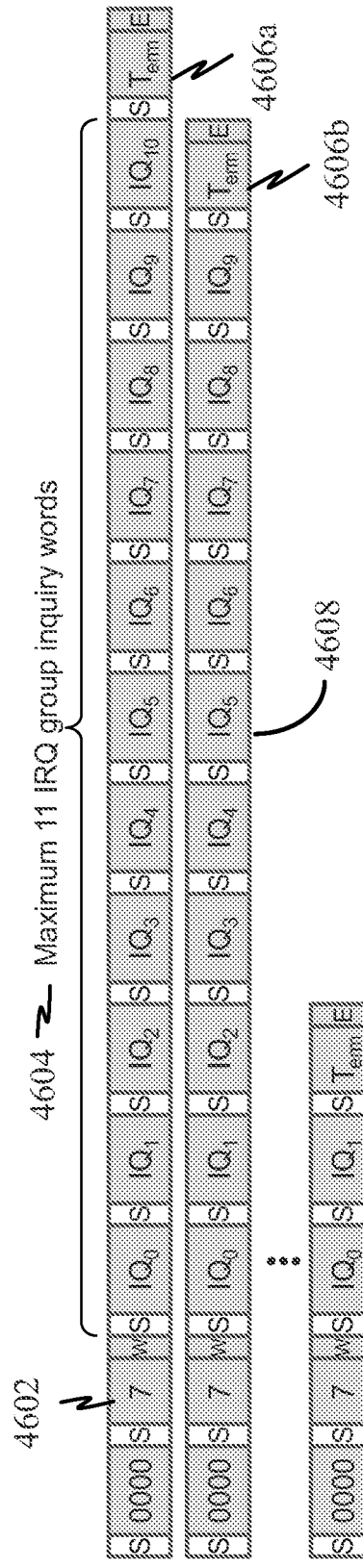
FIG. 44

CCIe Bit19 mapping cont.

Ternary	Bits[19:0]	Address	Write	Read	Bits[i]												
					19	12	11	9	8	7	6	5	4	3	2	1	0
$2222\_2222\_2222_3$	0x81BF0	♫SY-		Prohibited*	1	1	1	1	1	1	1	1	1	0	0	0	0
$2222\_2222\_2221_3$	0x81BEF 0x2A (42) ↕ $2222\_2222\_1102_3$ 0x81BC6	Prohibited															
$2222\_2222\_1101_3$	0x81BC5	Heartbeat ♥ / -NC♫		Prohibited*	1	1	1	1	1	1	1	0	0	0	1	0	1
$2222\_2222\_1100_3$	0x81BC4 0xA (10) ↕ $2222\_2222\_1000_3$ 0x81BBB	Prohibited															
$2222\_2222\_0222_3$	0x81BBA	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	1	0
$2222\_2222\_0221_3$	0x81BB9	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	0	1
$2222\_2222\_0220_3$	0x81BB8	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	1	0	0	0
$2222\_2222\_0212_3$	0x81BB7 [3-bits] 0x8 (8) ↕ $2222\_2222\_0121_3$ 0x81BB0	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	1	0	x	x	x
$2222\_2222\_0120_3$	0x81BAF [4-bits] 0x10 (16) ↕ $2222\_2222\_0000_3$ 0x81BA0	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	1	0	x	x	x	x
$2222\_2221\_2222_3$	0x81B9F [5-bits] 0x20 (32) ↕ $2222\_2221\_1211_3$ 0x81B80	Reserved	Reserved	Reserved	1	1	1	1	1	1	0	0	x	x	x	x	x

*) ♫SY--NC♫ is prohibited for read since there shall not be IB-IRQ during read

FIG. 45



4600

Message type	Name	Payload	Description
0x0007	IRQ group inquiry	1 up to11 IRQ group inquiry words + 1 terminator	Each inquiry word has 3 inquiry response slots, and total 33 slots, for group 0 to group 32. • Master must send at least one inquiry word, D_0. • Master may end inquiry word sequence at any word before D_{10}.

For each inquiry word;

- All devices must start masking SDA at T_{11} RXCLK and release the mask at dummy (T_{-1}) RXCLK
- See next slide for inquiry response slot of each inquiry word.

FIG. 46

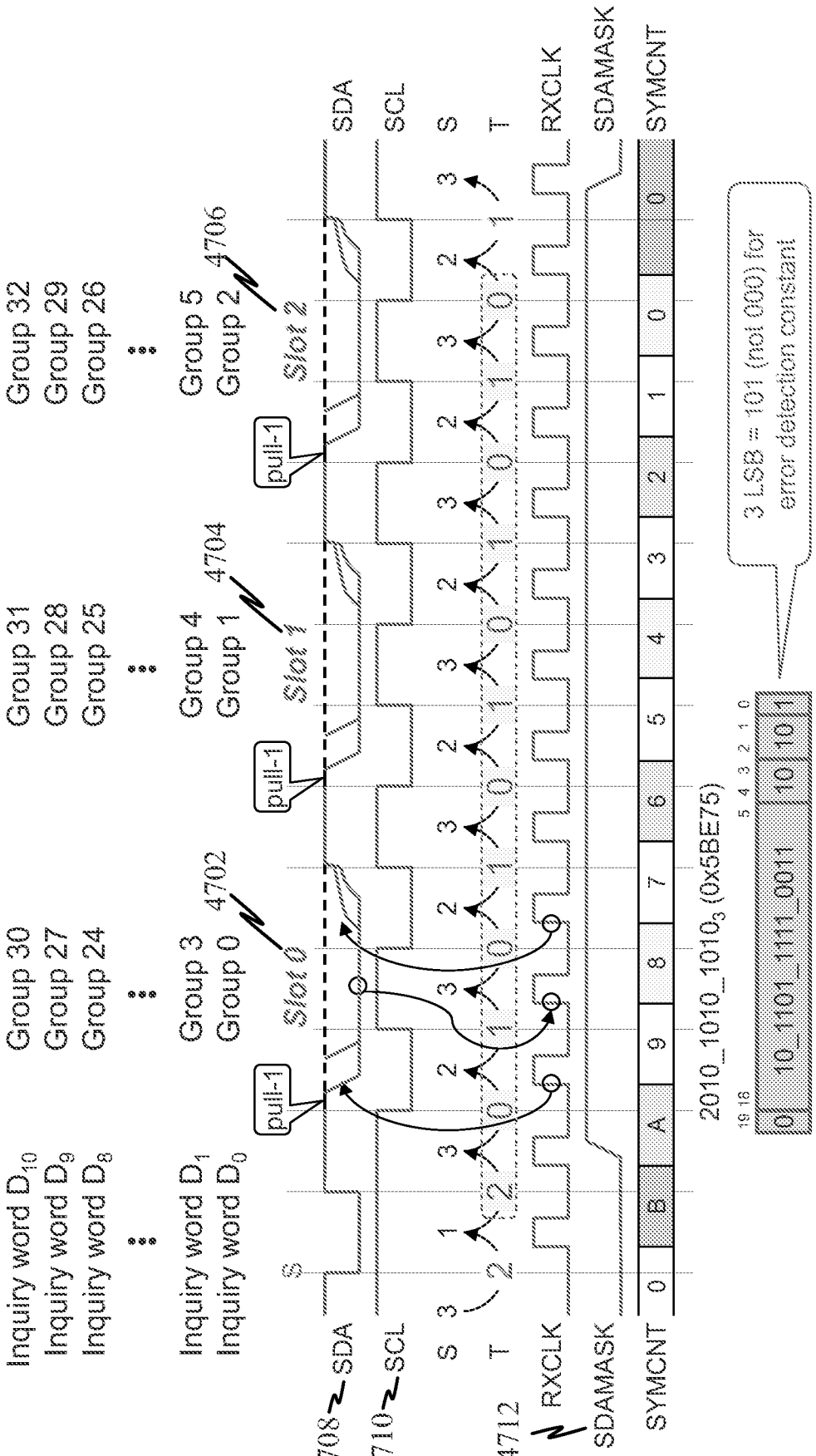

EXEMPLARY IRQ GROUP INQUIRY
Interrupting slave drives SDA low at corresponding slot of corresponding inquiry word.
Inquiry word D10 Group 30 Group 31 Group 32
Inquiry word D9 Group 27 Group 28 Group 29
Inquiry word D8 Group 24 Group 25 Group 26
Inquiry word D1 Group 3 Group 4 Group 5
Inquiry word D0 Group 0 Group 1 Group 2
Slot 0
Slot 1
Slot 2
4702
4704
4706
pull-1
4708 SDA
4710 SCL
4712 RXCLK
SDAMASK
SYMCNT
SDA
SCL
RXCLK
SDAMASK
SYMCNT
2010_1010_1010₃ (0x5BE75)
19 18
5 4 3 2 1 0
0 10_1101_1111_0011 10 10 1
3 LSB = 101 (not 000) for error detection constant

FIG. 47

EXEMPLARY TERMINATOR WORD FOR IRQ GROUP INQUIRY GENERAL CALL

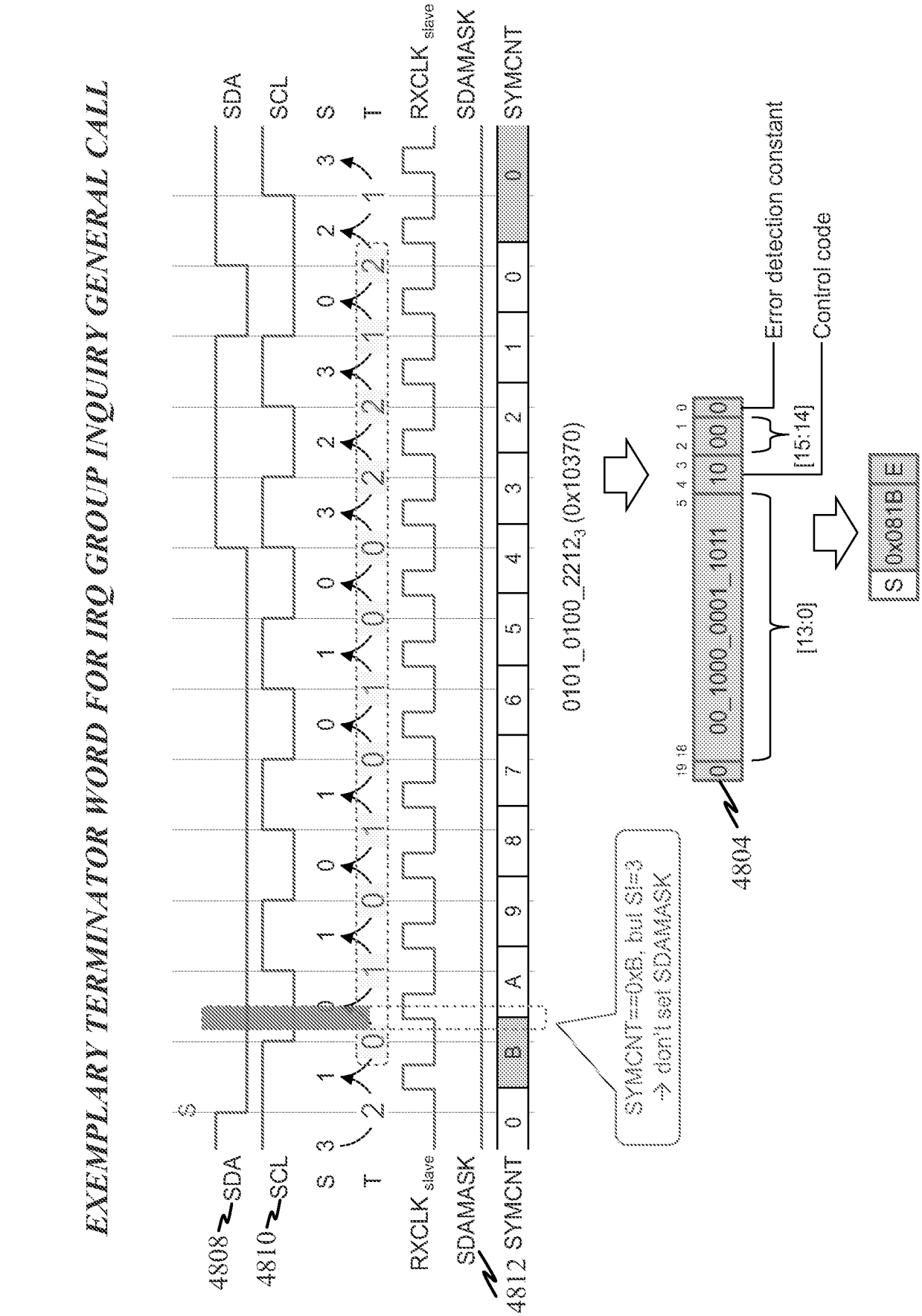
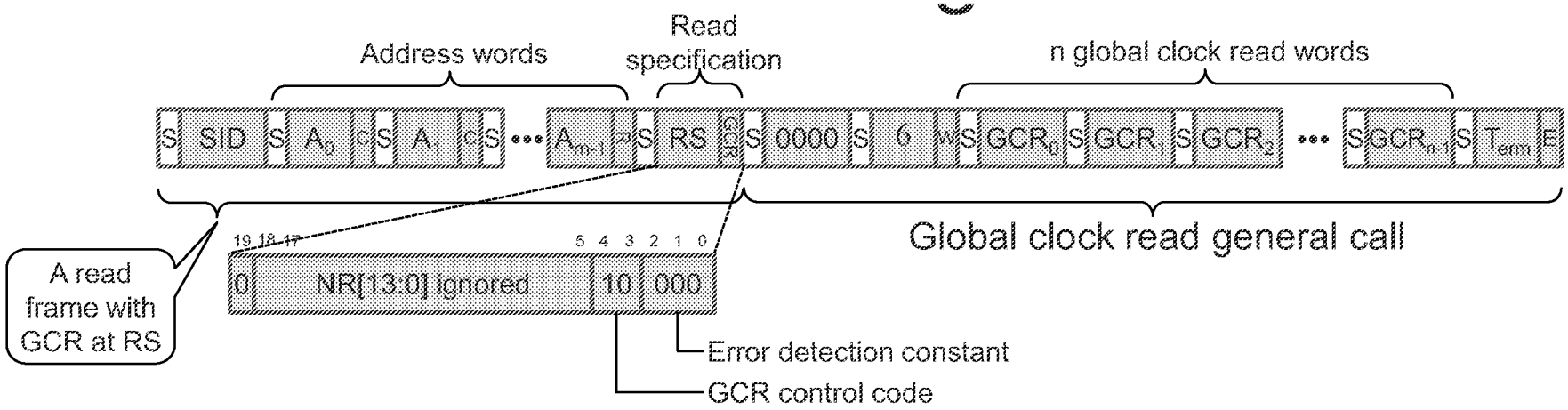


FIG. 48

EXEMPLARY GLOBAL CLOCK READ GENERAL CALL



Message type	Name	Payload	Description
0x0006	Global clock read	SID + N global clock read words + terminator	Master reads 9-bits data per each global clock read (GCR) word from selected device. All device on the bus must gate SDA during GCR period. Selected device can recover RXCLK from SCL.

- Global clock read general call must be preceded by a read frame with "GCR" control code at the read spec (RS) word.
- For each global clock read (GCR) word;
  - All devices must start masking SDA at T_{11} RXCLK and release the mask at dummy (T_{-1}) RXCLK
  - See next slide for inquiry response slot of each inquiry word.
- The slave addressed by SID in the preceding read frame returns bytes data during the payload of the general call.

FIG. 49

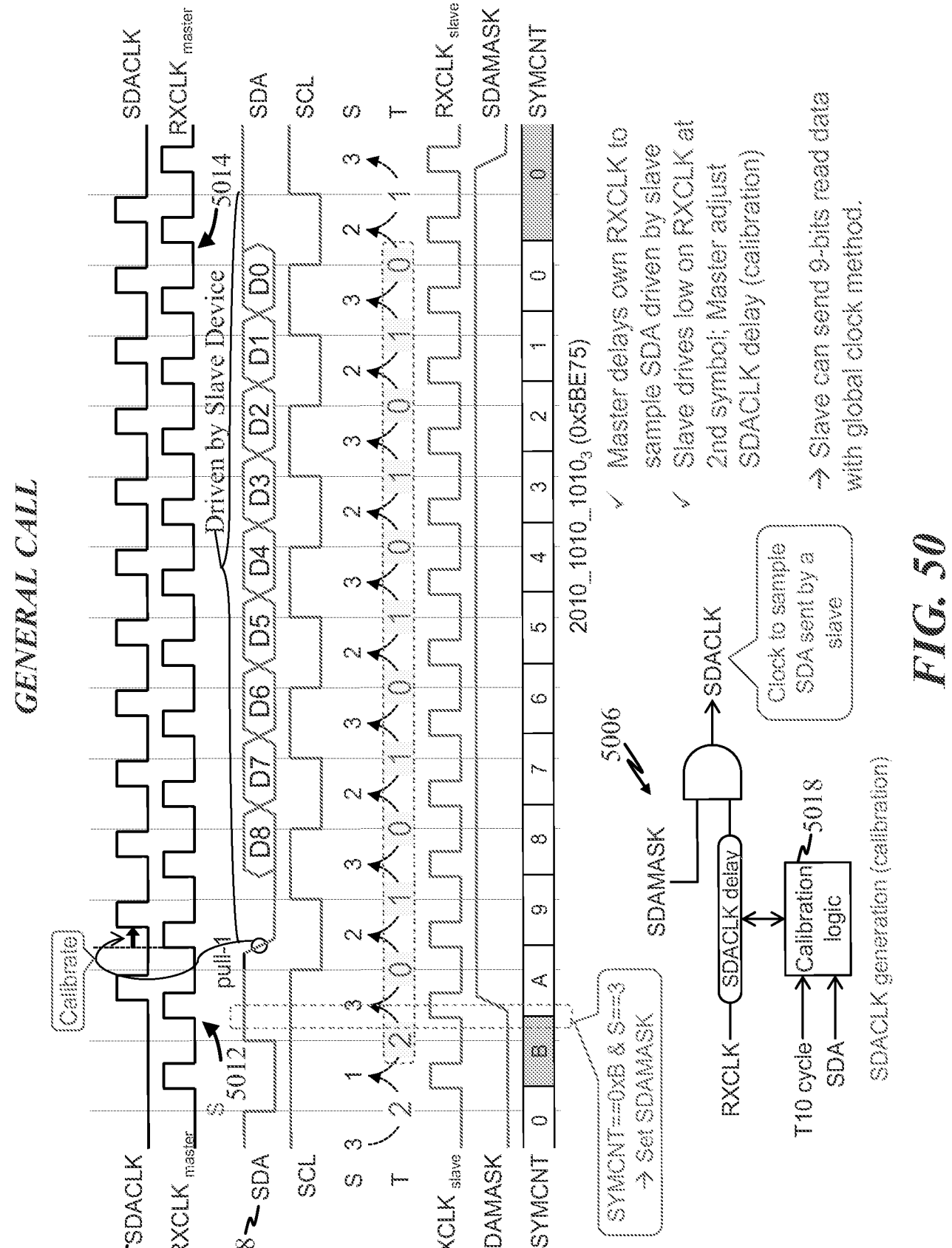

EXEMPLARY TIMING DIAGRAM FOR GLOBAL CLOCK READ GENERAL CALL
SDACLK
RXCLK master
SDA
SCL
S
T
RXCLK slave
SDAMASK
SYMCNT
Calibrate
pull-1
Driven by Slave Device
5002
5004
5008
5016
5010
5012
5014
5006
5018
2010_1010_1010$_3$ (0x5BE75)
SYMCNT==0xB & S==3
→ Set SDAMASK
Master delays own RXCLK to sample SDA driven by slave
Slave drives low on RXCLK at 2nd symbol; Master adjust SDACLK delay (calibration)
→ Slave can send 9-bits read data with global clock method.
Clock to sample SDA sent by a slave
SDACLK delay
Calibration logic
T10 cycle
SDACLK generation (calibration)

FIG. 50

EXEMPLARY DEVICE WITH COEXISTENCE OF sI2C and CCIe DEVICES OVER A SHARED BUS

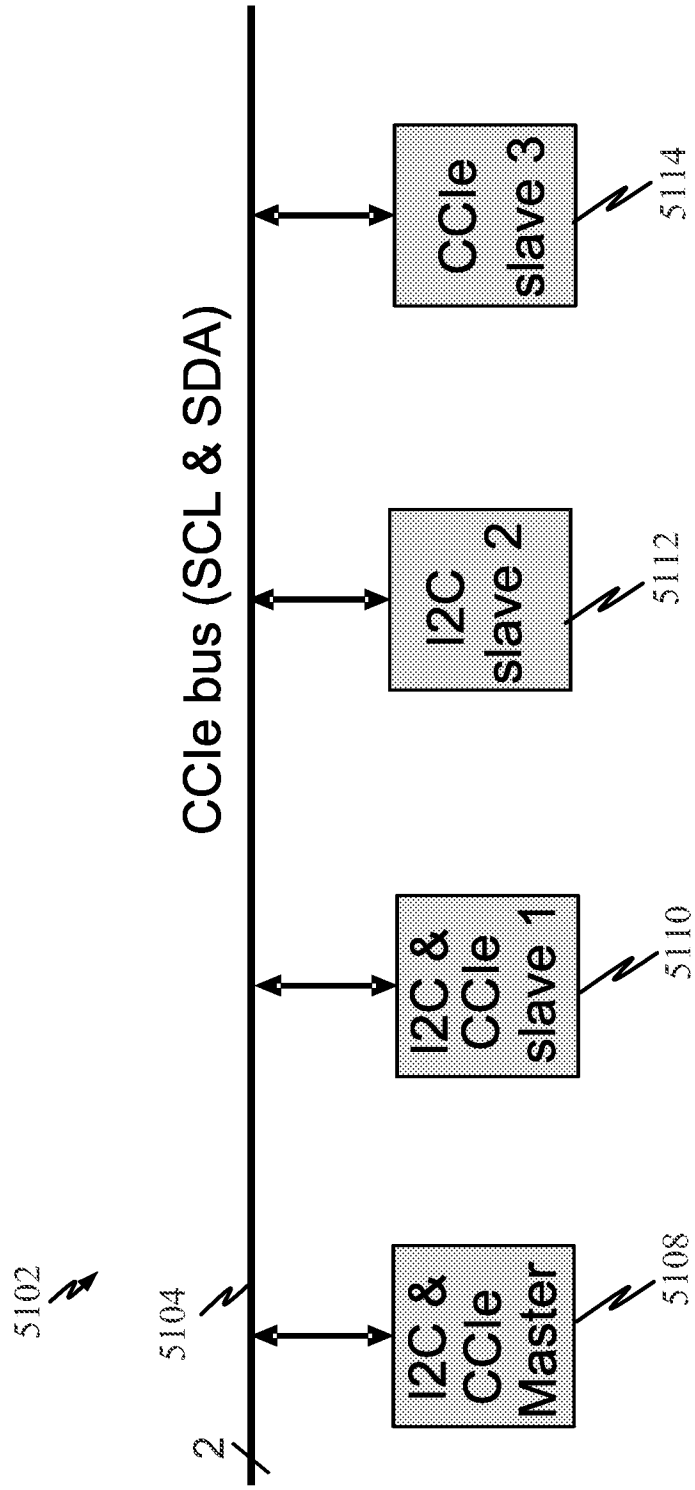


FIG. 51

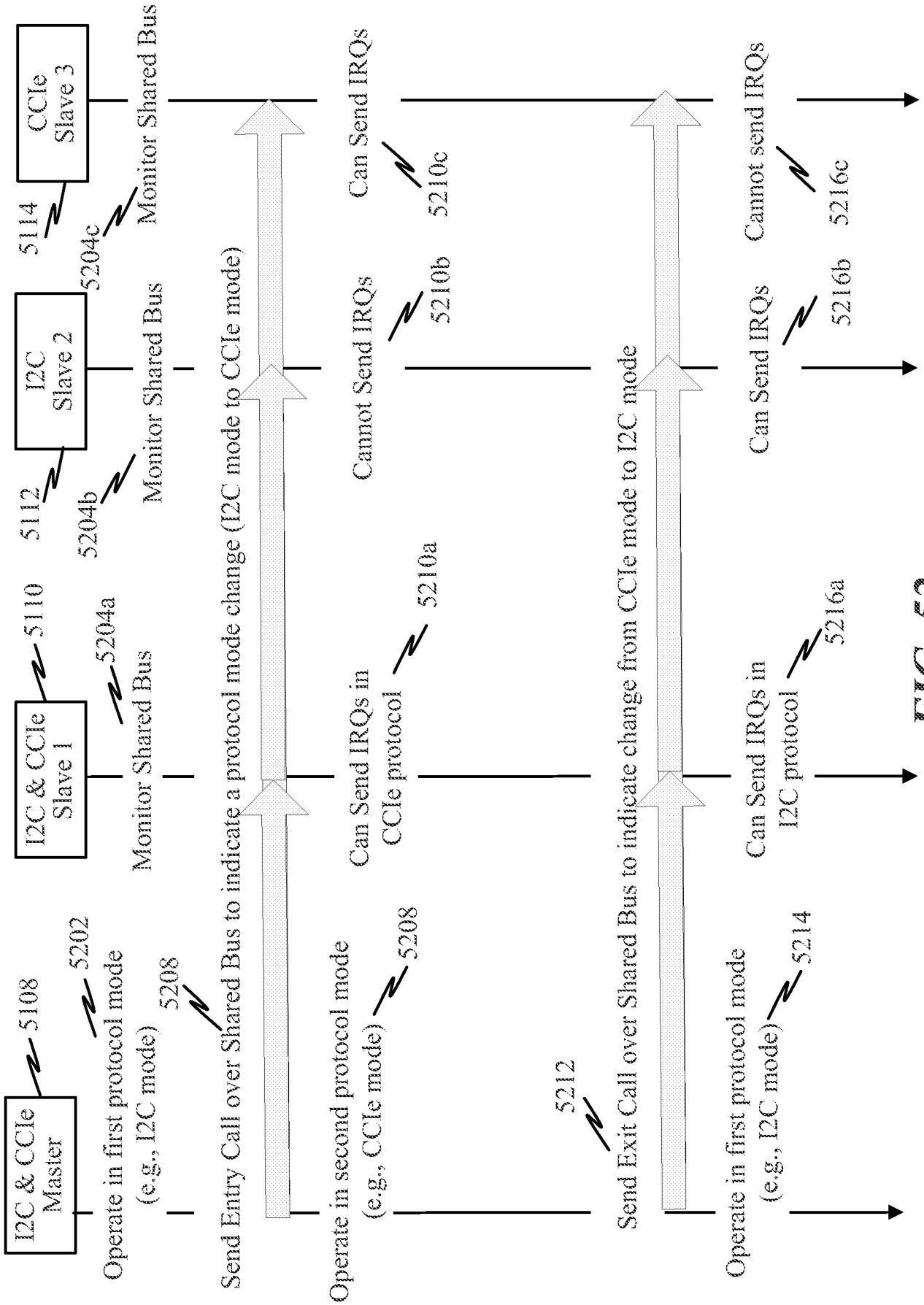

I2C & CCIe Master
5108
I2C & CCIe Slave 1
5110
I2C Slave 2
5112
CCIe Slave 3
5114
5202
Operate in first protocol mode (e.g., I2C mode)
5204a
Monitor Shared Bus
5204b
Monitor Shared Bus
5204c
Monitor Shared Bus
5208
Send Entry Call over Shared Bus to indicate a protocol mode change (I2C mode to CCIe mode)
Operate in second protocol mode (e.g., CCIe mode)
5208
Can Send IRQs in CCIe protocol
5210a
Cannot Send IRQs
5210b
Can Send IRQs
5210c
5212
Send Exit Call over Shared Bus to indicate change from CCIe mode to I2C mode
Operate in first protocol mode (e.g., I2C mode)
5214
Can Send IRQs in I2C protocol
5216a
Can Send IRQs
5216b
Cannot send IRQs
5216c

FIG. 52

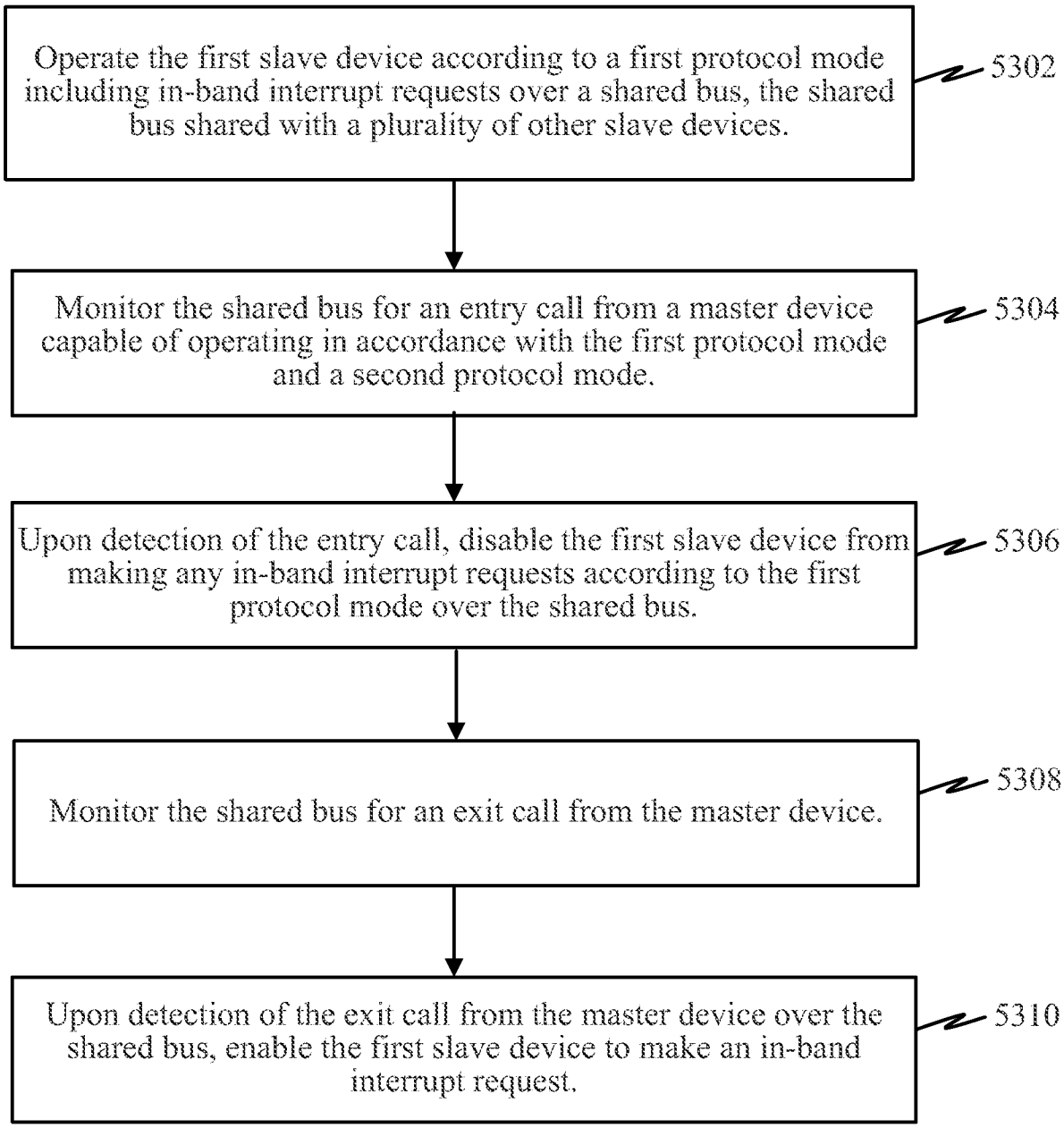

Operate the first slave device according to a first protocol mode including in-band interrupt requests over a shared bus, the shared bus shared with a plurality of other slave devices.
5302
Monitor the shared bus for an entry call from a master device capable of operating in accordance with the first protocol mode and a second protocol mode.
5304
Upon detection of the entry call, disable the first slave device from making any in-band interrupt requests according to the first protocol mode over the shared bus.
5306
Monitor the shared bus for an exit call from the master device.
5308
Upon detection of the exit call from the master device over the shared bus, enable the first slave device to make an in-band interrupt request.
5310

FIG. 53

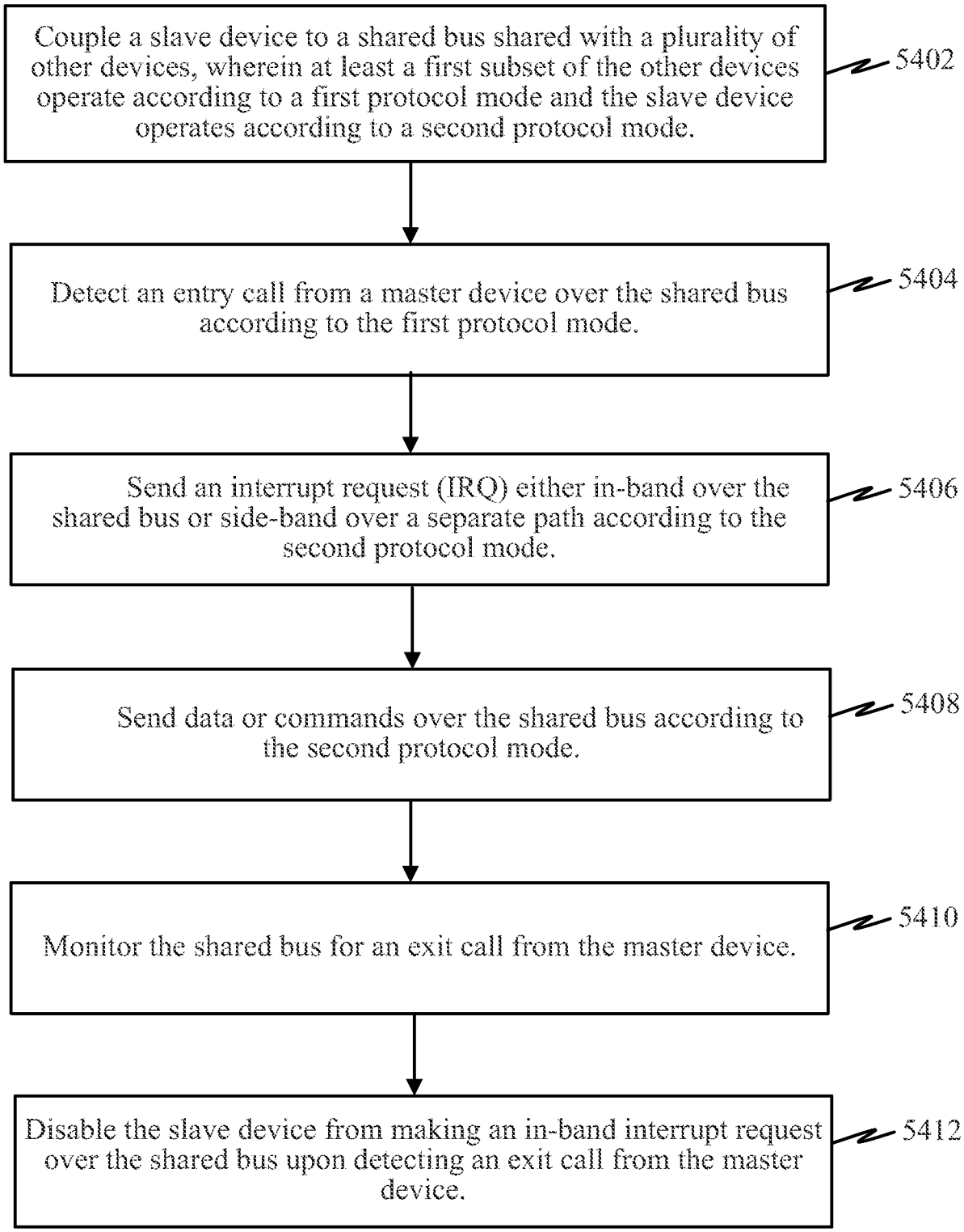
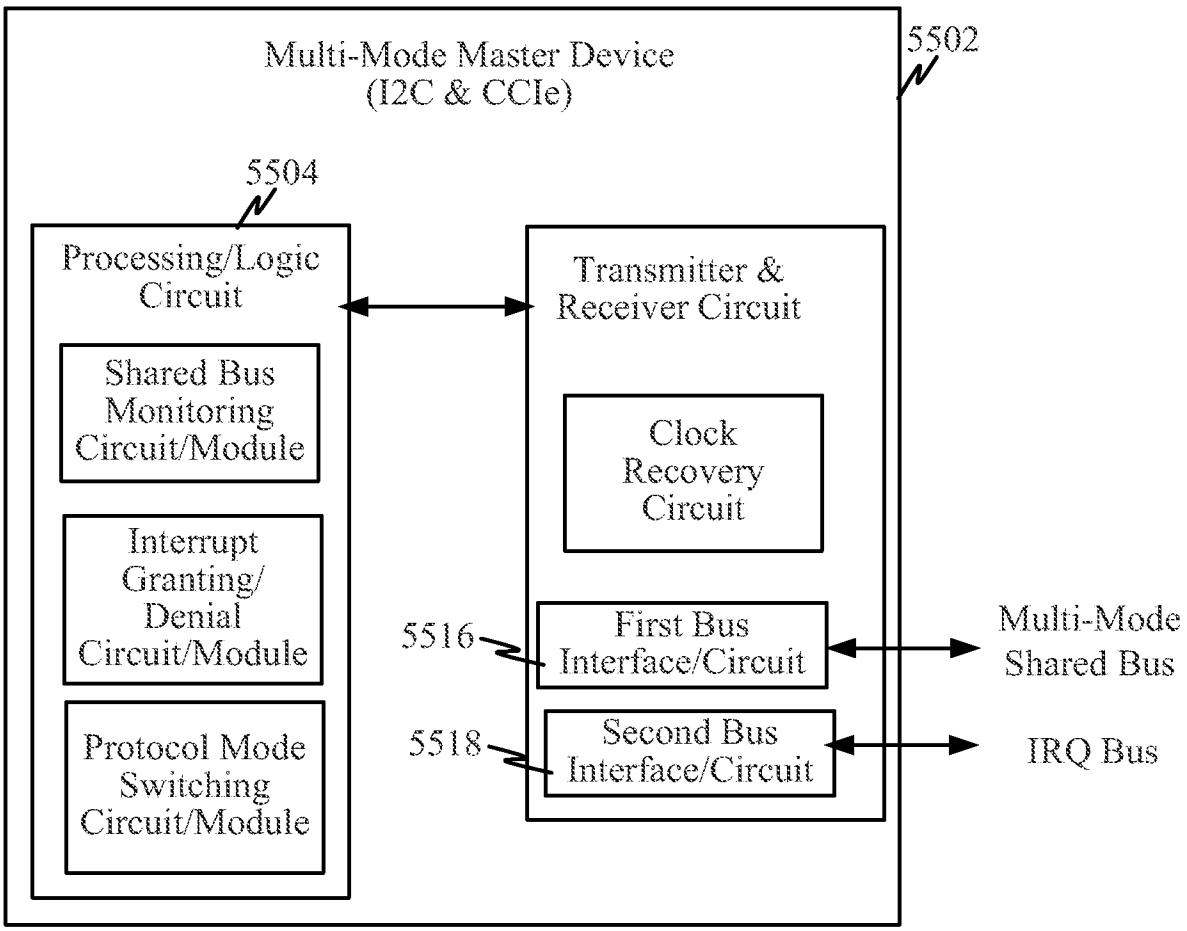
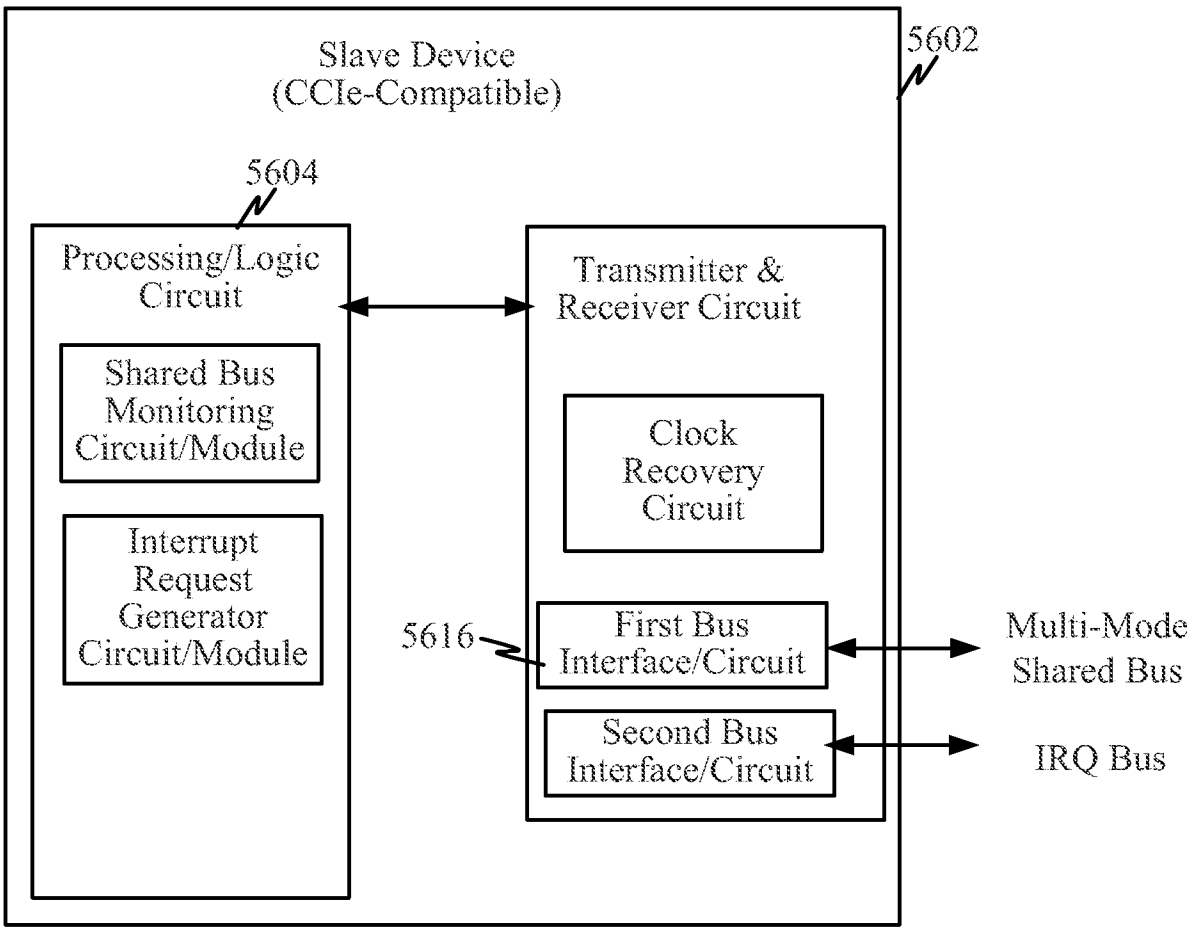

Couple a slave device to a shared bus shared with a plurality of other devices, wherein at least a first subset of the other devices operate according to a first protocol mode and the slave device operates according to a second protocol mode.
5402
Detect an entry call from a master device over the shared bus according to the first protocol mode.
5404
Send an interrupt request (IRQ) either in-band over the shared bus or side-band over a separate path according to the second protocol mode.
5406
Send data or commands over the shared bus according to the second protocol mode.
5408
Monitor the shared bus for an exit call from the master device.
5410
Disable the slave device from making an in-band interrupt request over the shared bus upon detecting an exit call from the master device.
5412

FIG. 54



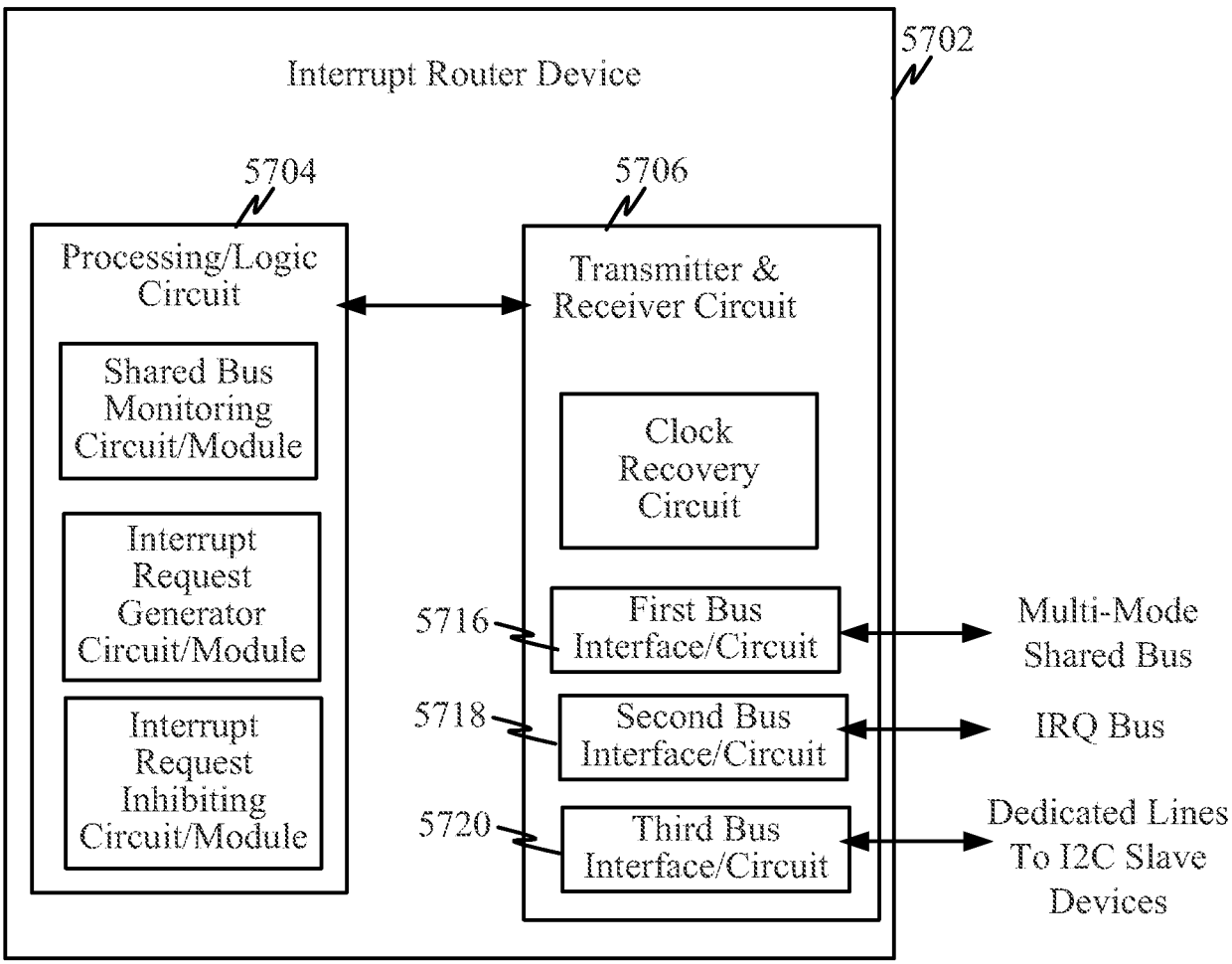
EXEMPLARY MULTI-MODE CCIe/I2C MASTER DEVICE

FIG. 55



EXEMPLARY CCIe SLAVE DEVICE

FIG. 56



EXEMPLARY INTERRUPT ROUTER SLAVE DEVICE

FIG. 57

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/059776

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F13/42
ADD.

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

B. FIELDS SEARCHED

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

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 00/42740 A1 (ERICSSON TELEFON AB L M [SE]) 20 July 2000 (2000-07-20) figures 1,3 page 1 - page 3	1-28
Y	US 7 089 338 B1 (WOOTEN DAVID [US] ET AL) 8 August 2006 (2006-08-08) figures 1,1A,3-5 column 9, line 25 - line 64	1-28
A	EP 0 588 191 A1 (THOMSON CONSUMER ELECTRONICS [US]) 23 March 1994 (1994-03-23) figure 1 abstract	1-28
A	US 2007/088874 A1 (BRABANT RICHARD [US]) 19 April 2007 (2007-04-19) the whole document	1-28

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 December 2014	07/01/2015

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

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/059776

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0042740 A1	20-07-2000	AR 022134 A1	04-09-2002
		AT 254369 T	15-11-2003
		AU 3089700 A	01-08-2000
		CN 1333964 A	30-01-2002
		DE 69912845 D1	18-12-2003
		EP 1142210 A1	10-10-2001
		HK 1043678 A1	21-10-2004
		JP 4480897 B2	16-06-2010
		JP 2002535882 A	22-10-2002
		MY 125638 A	30-08-2006
		RU 2231230 C2	20-06-2004
		TR 200102017 T2	21-12-2001
		US 6253268 B1	26-06-2001
		WO 0042740 A1	20-07-2000
US 7089338 B1	08-08-2006	NONE	
EP 0588191 A1	23-03-1994	CN 1084986 A	06-04-1994
		EP 0588191 A1	23-03-1994
		ES 2113458 T3	01-05-1998
		JP 2805579 B2	30-09-1998
		JP H06205072 A	22-07-1994
		SG 75764 A1	24-10-2000
		US 5376928 A	27-12-1994
US 2007088874 A1	19-04-2007	NONE	