I2C\_MAS

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date | Description of Change | Author |
| V0.0 | 9/20/2022 | Draft version | Su Aixue |

Table of Contents

[I2C\_MAS 2](#_Toc121236108)

[Introduction 2](#_Toc121236109)

[Register Definition 2](#_Toc121236110)

[Register Map 2](#_Toc121236111)

[I2C\_MAS\_CTRL 2](#_Toc121236112)

[I2C\_TR 3](#_Toc121236113)

[I2C\_RD 3](#_Toc121236114)

[Function Details 3](#_Toc121236115)

[Block Diagram 3](#_Toc121236116)

[I2C\_MAS IO Descriptions 4](#_Toc121236117)

[I2C Key Signal Descriptions 5](#_Toc121236118)

[I2C Communication Formats 7](#_Toc121236119)

[I2C\_MAS Function Descriptions 8](#_Toc121236120)

# I2C\_MAS

## Introduction

The I2C bus is a simple, bidirectional two-wire synchronous serial bus. It requires only two wires to transfer information between devices connected to the bus.

The I2C master device is used to start the bus to transmit data and generate a clock to open the device for transmission. At this time, any addressed device is regarded as a slave device. The relationship between master and slave, sending and receiving on the bus is not constant, but depends on the direction of data transfer at this time. If the master device wants to send data to the slave device, the master device first addresses the slave device, then actively sends the data to the slave device, and finally the master device terminates the data transfer; if the master device wants to receive data from the slave device, the master device first addresses the slave device first, then the master device receives the data sent from the device, and finally the master device terminates the receiving process. In this case, the master device is responsible for generating the timing clock and terminating the data transfer.

The I2C\_MAS module has the following features:

• Bidirectional two-wire synchronous serial bus;

• Support i2c master writing command;

• Output SCL and SDA\_OUT according to I2C\_MAS\_EN and I2C\_CTRL; (HWR001\_I2C\_MAS)

• Output RD\_DATA and ACK\_BIT to DS\_REG according to SDA\_IN. (HWR002\_I2C\_MAS)

## Register Definition

### Register Map

Table 1 I2C\_MAS Register Map

|  |  |  |  |
| --- | --- | --- | --- |
| **ADDRESS** | **NAME** | **DESCRIPTION** | **RESET VALUE** |
| **I2C\_MAS** | | | |
| 0x2200 | **I2C\_MAS\_CTRL** | I2C\_MAS control register | 0x00 |
| 0x2201 | **I2C\_TR** | I2C send register | 0x00 |
| 0x2202 | **I2C\_RD** | I2C receive register | 0x00 |

### I2C\_MAS\_CTRL

Register 1. I2C\_MAS\_CTRL (I2C\_MAS control register, offset 0x000)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BIT** | **NAME** | **ACCESS** | **RESET** | **DESCRIPTION** |
| 7 | **STOP** | R/W | 1’b0 | Stop Command  0: Ready  1: Execute |
| 6 | **RX** | R/W | 1’b0 | Receive Command  0: Ready  1: Execute |
| 5 | **SR** | R/W | 1’b0 | Restart Command  0: Ready  1: Execute |
| 4 | **ACK** | R | 1’b0 | Acknowledge from Slave device  0: NACK  1: ACK |
| 3:1 | **REV** | R | 3’b0 | Reserved |
| 0 | **TX** | R/W | 1’b0 | Send Command  0: Ready  1: Execute |

### I2C\_TR

Register 2. I2C TR (I2C send register, offset 0x001)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BIT** | **NAME** | **ACCESS** | **RESET** | **DESCRIPTION** |
| 7:0 | **DATA** | R/W | 8’h00 | Data Sent |

### I2C\_RD

Register 3. I2C\_RD (I2C receive register, offset 0x002)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BIT** | **NAME** | **ACCESS** | **RESET** | **DESCRIPTION** |
| 7:0 | **DATA** | R/W | 8’h00 | Data Received |

## Function Details

### Block Diagram

The main elements of I2C\_MAS and their interactions are shown in Fig 1.



Fig 1. I2C\_MAS Block Diagram

### I2C\_MAS IO Descriptions

This section provides the I2C\_MAS IO descriptions.

Table 2 I2C\_MAS IO descriptions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Signal** | **Width** | **Duration** | **I/O** | **Default Value** | **Register** | **Description** |
| CLK\_I2C\_SC | 1 | -- | I | -- | -- | 400kHz |
| resetb\_SR\_CLK\_I2C | 1 | -- | I | -- | -- | -- |
| SCL | 1 | -- | O | 1’b0 | -- | -- |
| SDA\_OUT | 1 | -- | O | 1’b0 | -- | -- |
| SDA\_IN | 1 | -- | I | -- | -- | -- |
| i2c\_tx | 1 | 1~2 CLK\_I2C\_SC\_DIV4 | I | -- | TX | 8us pulse |
| i2c\_rx | 1 | 1~2 CLK\_I2C\_SC\_DIV4 | I | -- | RX | 8us pulse |
| i2c\_sr | 1 | 1~2 CLK\_I2C\_SC\_DIV4 | I | -- | SR | 8us pulse |
| i2c\_stop | 1 | 1~2 CLK\_I2C\_SC\_DIV4 | I | -- | STOP | 8us pulse |
| i2c\_tx\_data | 8 | -- | I | -- | I2C TR | -- |
| i2c\_tr\_go\_status | 1 | -- | O | 1’b1 | -- | -- |
| i2c\_comm\_valid | 2 | 2036 CLK\_I2C\_SC | O | 2’b0 | -- | -- |
| i2c\_ack\_out | 1 | -- | O | 1’b0 | ACK | -- |
| i2c\_rd\_data | 8 | -- | O | 8’b0 | I2C\_RD | -- |

### I2C Key Signal Descriptions

Table 3 I2C key signal descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal** | **Width** | **Duration** | **Default Value** | **Description** |
| div\_cnt | 4 | 1 CLK\_I2C\_SC | 4’d0 | It is used to count the posedge of CLK\_I2C\_SC from 0 to 3 circularly. The encoding format is gray code. |
| pulse\_i2c\_div4\_0 | 1 | 1 CLK\_I2C\_SC | 1’b1 | It is equal to “(div\_cnt == ‘h0)”. |
| pulse\_i2c\_div4\_1 | 1 | 1 CLK\_I2C\_SC | 1’b0 | It is equal to “(div\_cnt == ‘h1)”. |
| pulse\_i2c\_div4\_2 | 1 | 1 CLK\_I2C\_SC | 1’b0 | It is equal to “(div\_cnt == ‘h3)”. |
| pulse\_i2c\_div4\_3 | 1 | 1 CLK\_I2C\_SC | 1’b0 | It is equal to “(div\_cnt == ‘h2)”. |
| CLK\_I2C\_SC\_DIV4 | 1 | 2 CLK\_I2C\_SC | 1’b0 | It is high level when “(div\_cnt == ‘h1) || (div\_cnt == ‘h3)”; and it is low level when “(div\_cnt == ‘h0) || (div\_cnt == ‘h2)”. Thus, the period of this signal is two times the period of CLK\_I2C\_SC. |
| i2c\_stop\_r | 1 | 52 CLK\_I2C\_SC | 1’b0 | It is a level signal between i2c\_stop\_sync (i2c\_stop after synchronization) and “i2c\_stop\_rr\_ddd & i2c\_stop\_real\_dd”. In last words, i2c\_stop\_rr\_ddd is generated by delaying i2c\_stop\_rr three clock cycles, i2c\_stop\_real\_dd is generated by delaying i2c\_stop two clock cycles. |
| i2c\_stop\_rr | 1 | 12 CLK\_I2C\_SC | 1’b0 | It is a level signal between “i2c\_stop\_r & (dcnt == 9)” and “i2c\_stop\_rr\_ddd & i2c\_stop\_real\_dd”. In last words, i2c\_stop\_rr\_ddd is generated by delaying i2c\_stop\_rr three clock cycles, i2c\_stop\_real\_dd is generated by delaying i2c\_stop two clock cycles. |
| i2c\_stop\_real | 1 | 8 CLK\_I2C\_SC/  5 CLK\_I2C\_SC | 1’b0 | The value of this signal is equal to “(i2c\_stop\_rr\_d & (~i2c\_stop\_rr\_ddd) | (i2c\_stop\_sync & ~i2c\_sr\_r))”, in which, the i2c\_stop\_rr\_d is generated by delaying i2c\_stop\_rr one clock cycle, the i2c\_stop\_rr\_ddd is generated by delaying i2c\_stop\_rr three clock cycles, i2c\_stop\_sync is after synchronization of i2c\_stop. This signal is used to note the stop information after the last read data that behind the i2c\_stop pulse. |
| i2c\_stop\_real\_d | 1 | 8 CLK\_I2C\_SC/  4 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal i2c\_stop\_real through a trigger. Clock pins of the trigger is connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_stop\_real\_dd | 1 | 8 CLK\_I2C\_SC/  4 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal i2c\_stop\_real through two triggers. Clock pins of two triggers are connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_sr\_r | 1 | -- | 1’b0 | It is a level signal between i2c\_sr\_sync (i2c\_sr after synchronization) and “i2c\_stop\_rr\_ddd & i2c\_stop\_real\_dd”. In last words, i2c\_stop\_rr\_ddd is generated by delaying i2c\_stop\_rr three clock cycles, i2c\_stop\_real\_dd is generated by delaying i2c\_stop two clock cycles. |
| i2c\_sr\_sync | 1 | 5 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal i2c\_sr through a trigger. Clock pins of the trigger is connected to CLK\_I2C\_SC. |
| i2c\_sr\_d[0] | 1 | -- | 1’b0 | It is the signal obtained by signal i2c\_sr\_sync through a trigger. Clock pins of the trigger is connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_sr\_d[1] | 1 | -- | 1’b0 | It is the signal obtained by signal i2c\_sr\_sync through two triggers. Clock pins of two triggers are connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_sr\_d[2] | 1 | -- | 1’b0 | It is the signal obtained by signal i2c\_sr\_sync through three triggers. Clock pins of three triggers are connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_start\_r | 1 | 812 CLK\_I2C\_SC/  800 CLK\_I2C\_SC | 1’b0 | It is a level signal between “(~i2c\_busy) & i2c\_tx\_sync” and i2c\_stop\_real, in which, i2c\_tx\_sync is after synchronization of i2c\_tx. |
| i2c\_start | 1 | 8 CLK\_I2C\_SC | 1’b0 | It is a pulse signal generated by detecting the rising edge of i2c\_start\_r. It is used to start the communication. |
| i2c\_start\_d | 1 | 8 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal i2c\_start through a trigger. Clock pins of the trigger is connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_start\_dd | 1 | 8 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal i2c\_start through two triggers. Clock pins of two triggers are connected to CLK\_I2C\_SC\_DIV4. |
| i2c\_busy | 1 | 804 CLK\_I2C\_SC/  792 CLK\_I2C\_SC | 1’b0 | It is a level signal between i2c\_start\_d and i2c\_stop\_real. In last words, i2c\_start\_d is generated by delaying i2c\_start one clock cycle. |
| i2c\_rx\_ex | 1 | 5 CLK\_I2C\_SC | 1’b0 | The value of this signal is equal to “i2c\_rx\_sync | i2c\_stop\_sync”. In last words, the i2c\_rx\_sync is after synchronization of i2c\_rx, i2c\_stop\_sync is after synchronization of i2c\_stop. |
| dcnt\_go | 1 | 36 CLK\_I2C\_SC/  44 CLK\_I2C\_SC/  48 CLK\_I2C\_SC | 1’b0 | It is a level signal between “i2c\_tx\_sync || i2c\_sr\_sync || i2c\_rx\_ex” and “(dcnt == 8) & i2c\_busy”. In last words, the i2c\_tx\_sync is after synchronization of i2c\_tx, i2c\_sr\_sync is after synchronization of i2c\_sr. It is used to enable the internal counter. |
| dcnt\_go\_d | 1 | 36 CLK\_I2C\_SC/  44 CLK\_I2C\_SC/  48 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal dcnt\_go through a trigger. The clock pin of the trigger is connected to CLK\_I2C\_SC\_DIV4. |
| dcnt | 4 | 4 CLK\_I2C\_SC | 4’h0 | This signal is set to 0 when “i2c\_start || i2c\_stop\_real || i2c\_sr\_sync || i2c\_sr\_d[0] || i2c\_sr\_d[1] = 1” or “dcnt == 9”. In last words, the i2c\_sr\_sync is after synchronization of i2c\_sr, the i2c\_sr\_d[0] is generated by delaying i2c\_sr one clock cycle. I2c\_sr\_d[1] is generated by delaying i2c\_sr two clock cycles. If the above condition is not established and dcnt\_go = 1, this signal is increased by 1 every clock. After sampling the SDA\_IN (input signal), the value of this signal is used as the note of clock cycles for putting the sample values into corresponding bit of i2c\_rx\_data (output signal) and i2c\_ack\_out (output signal). |
| read\_en\_1 | 1 | -- | 1’b0 | It is a level signal between i2c\_sr\_d[0] and i2c\_stop\_real. In last words, i2c\_sr\_d[0] is generated by delaying i2c\_sr one clock cycle. |
| read\_en\_2 | 1 | 448 CLK\_I2C\_SC | 1’b0 | It is a level signal between “read\_en\_1 & (dcnt == 8)” and i2c\_stop\_real. It is used to enable the operation of reading data. |
| read\_en\_2\_dddd | 1 | 448 CLK\_I2C\_SC | 1’b0 | It is the signal obtained by signal read\_en\_2 through four triggers. The clock pins of four triggers are connected to CLK\_I2C\_SC\_DIV4. |

### I2C Communication Formats

The I2C communication timing diagram is shown in Fig 2.



Fig 2. I2C Communication Timing Diagram

The I2C data format is shown in Fig 3.

**Data writed from master to slave**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | SA | W | A | D | A | D | A | … | D | A/ | P |
| 1 bit | 9 bits | | | 9 bits | | 9 bits | | … | 9 bits | | 1 bit |

**Data read from salve to master**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | SA |  | A | D | A | Sr | SA | R | A | D | A | D | A | … | D |  | P |
| 1 bit | 9 bits | | | 9 bits | | 1 bit | 9 bits | | | 9 bits | | 9 bits | | … | 9 bits | | 1 bit |

Fig 3. I2C Data Format

**NOTE**:

S – Start bit. (1 bit)

SA – Address of slave device. (7 bits)

– The flag bit of writing, 1’b0. (1 bit)

R – The flag bit of reading, 1’b1. (1 bit)

A – Response bit, 1’b0. (1 bit)

– Non-response bit, 1’b1. (1 bit)

D – Data bits. (8 bits)

P – The flag bit of stopping. (1 bit)

|  |
| --- |
|  |

– The data direction is from master device to slave device.

|  |
| --- |
|  |

– The data direction is from slave device to master device.

### I2C\_MAS Function Descriptions

The I2C\_MAS module has two functions:

• Output SCL and SDA\_OUT according to I2C\_MAS\_EN and I2C\_CTRL; (Func 1 & Func 2) (HWR001\_I2C\_MAS)

• Output RD\_DATA and ACK\_BIT to DS\_REG according to SDA\_IN. (Func 3 & Func4) (HWR002\_I2C\_MAS)

Above functions can be found in the following timing diagrams.

Func 1: Signal SCL is the signal obtained by signal SCL\_R through a trigger. The clock pin of the trigger is connected to CLK\_I2C\_SC. Signal SCL\_R is equal to “(((scl\_and\_o | scl\_and\_o\_d) & dcnt\_go\_d & CLK\_I2C\_SC\_DIV4) | scl\_hold\_high\_r) & (~scl\_hold\_low\_r)”. The detailed description is as follows:

1. Signal dcnt\_go\_d, CLK\_I2C\_SC\_DIV4 can refer to Table 3.
2. Signal scl\_and\_o will become to high level when the high level of signal i2c\_start\_dd is detected using the posedge of CLK\_I2C\_SC\_DIV4. If the above condition is not satisfied, signal scl\_and\_o will become to low level when the high level of signal i2c\_stop\_real is sampled using the posedge of CLK\_I2C\_SC\_DIV4. If the above two conditions are not satisfied, the scl\_and\_o remains unchanged.

Note: Signal i2c\_start\_dd and i2c\_stop\_real can refer to Table 3.

1. Signal scl\_and\_o\_d is the signal obtained by signal scl\_and\_o through a trigger. The clock pin of the trigger is connected to CLK\_I2C\_SC\_DIV4.
2. Signal scl\_hold\_high\_r is the signal obtained by signal scl\_hold\_high through a trigger. The clock pin of the trigger is connected to CLK\_I2C\_SC. Signal scl\_hold\_high will become to high level when the high level of signal “i2c\_stop\_real\_d || i2c\_sr\_d[0]” is detected using CLK\_I2C\_SC. If the above condition is not satisfied, signal scl\_hold\_high will become to low level when the high level of signal “pulse\_i2c\_div4\_2 && (i2c\_start\_d || i2c\_sr\_d[1])”. If the above two conditions are not satisfied, the scl\_hold\_high remains unchanged.

Note: Signal i2c\_stop\_real\_d, i2c\_sr\_d, pulse\_i2c\_div4\_2 and i2c\_start\_d can refer to Table 3.

1. Signal scl\_hold\_low\_r is the signal obtained by signal scl\_hold\_low through a trigger. The clock pin of the trigger is connected to CLK\_I2C\_SC. Signal scl\_hold\_low is equal to “(~scl\_hold\_high) & scl\_hold\_high\_d”. Signal scl\_hold\_high\_d is the signal obtained by signal “scl\_hold\_high & read\_en\_1” through a trigger. The clock pin of the trigger is connected to “~CLK\_I2C\_SC\_DIV4”.

Note: Signal read\_en\_1 can refer to Table 3.

Func 2: Signal SDA\_OUT is the signal obtained by signal sda\_out\_reg through two triggers. The clock pins of two triggers are connected to CLK\_I2C\_SC. The generation of signal sda\_out\_reg can refer to Fig 4. And the signal in Fig 4 can refer to Table 3.

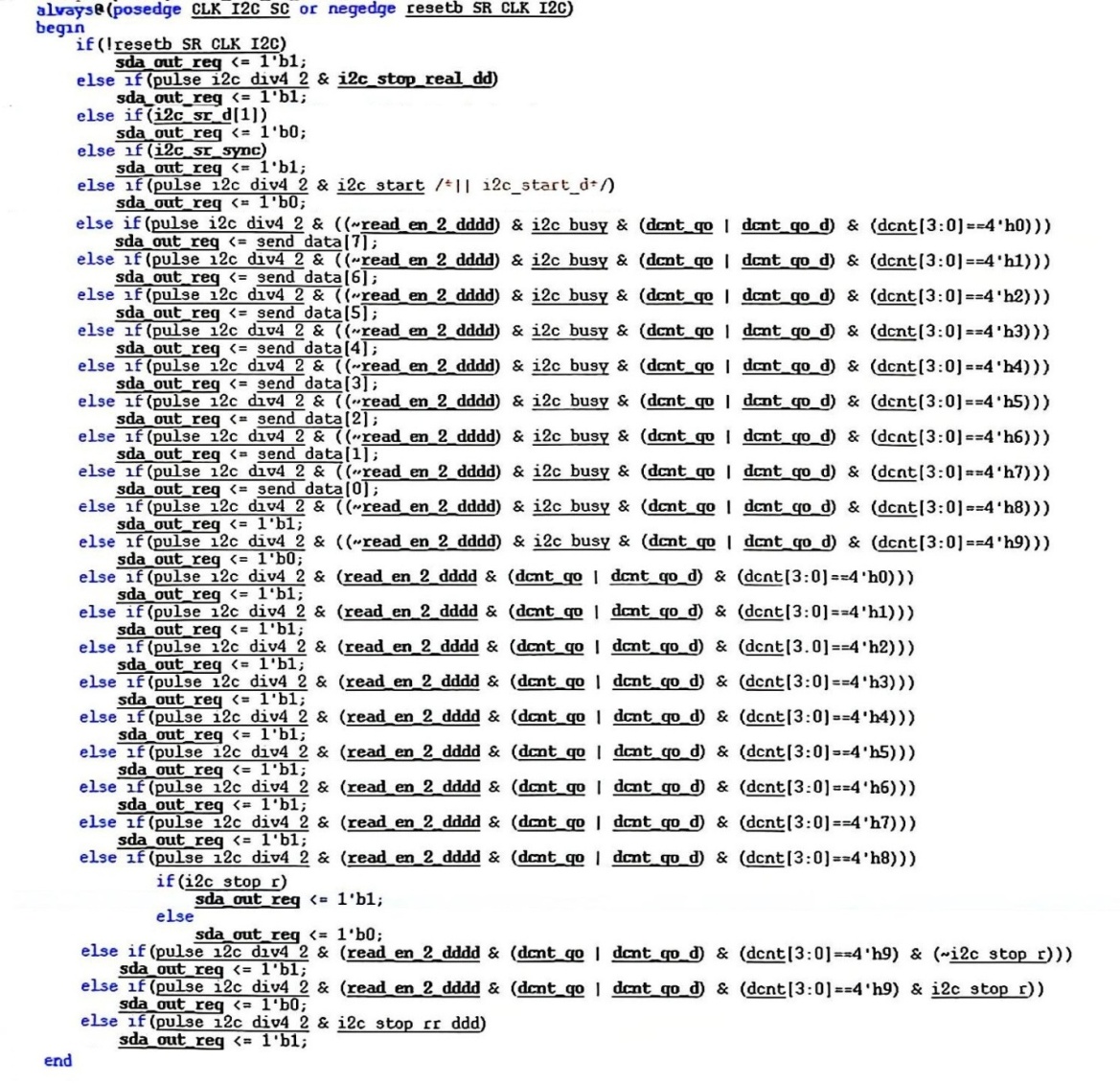


Fig 4. Generation of signal sda\_out\_reg

Func 3: i2c\_rx\_data[i] (RD\_DATA[i], i=0, 1, 2, …, 7) is the signal obtained by signal i2c\_rx\_data\_r[i] through a trigger. The clock pins of the triggers are connected to CLK\_I2C\_SC. While, the generation of i2c\_rx\_data\_r can refer to Fig 5. And the signal in Fig 5 can refer to Table 3.

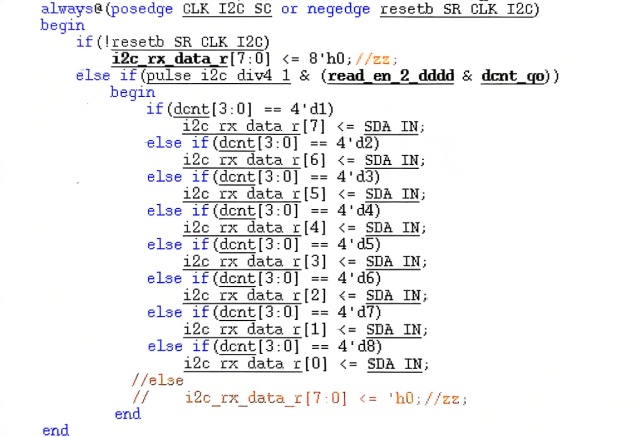


Fig 5. Generation of signal i2c\_rx\_data\_r

Func 4: i2c\_ack\_out (ACK\_BIT) is the signal obtained by signal i2c\_ack\_out\_r through a trigger. The clock pins of the trigger is connected to CLK\_I2C\_SC. While, i2c\_ack\_out\_r will become to “~SDA\_IN” when the high level of “(~read\_en\_2\_dddd) && (dcnt[3:0] == 4’h9)” is detected using the posedge of CLK\_I2C\_SC\_DIV4. And SDA\_IN is an input signal, signal read\_en\_2\_dddd and dcnt can refer to Table 3.

The result of implement is as following figures.

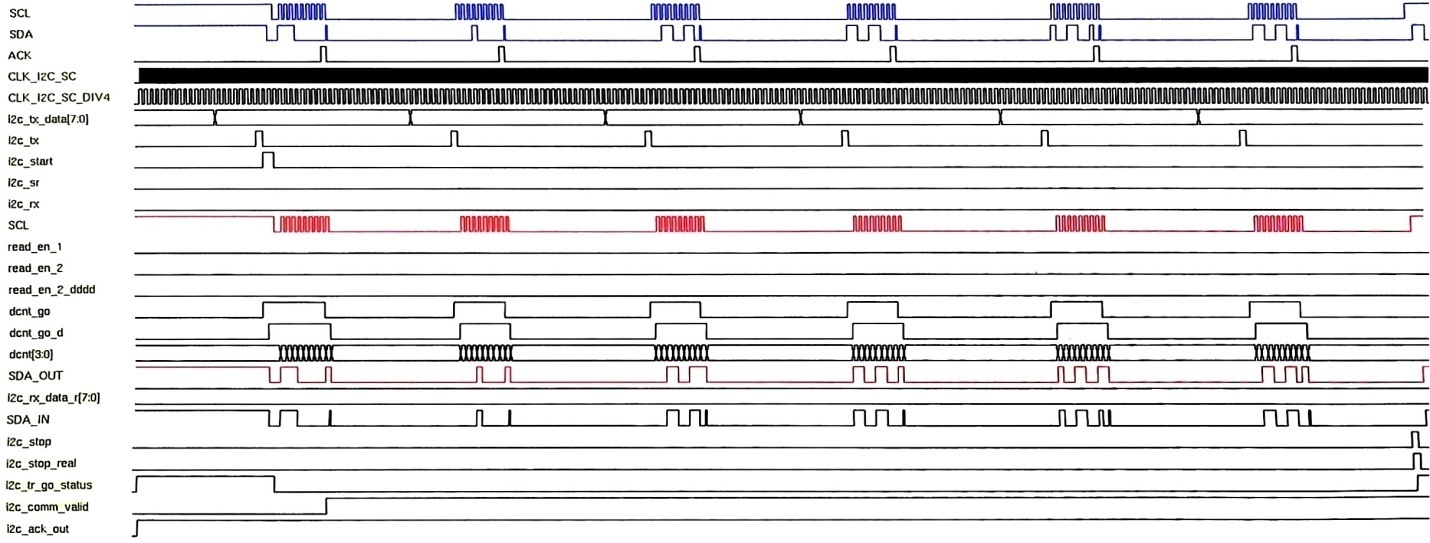


Fig 6. OVUV\_OTUT\_CMP Timing Diagram 1

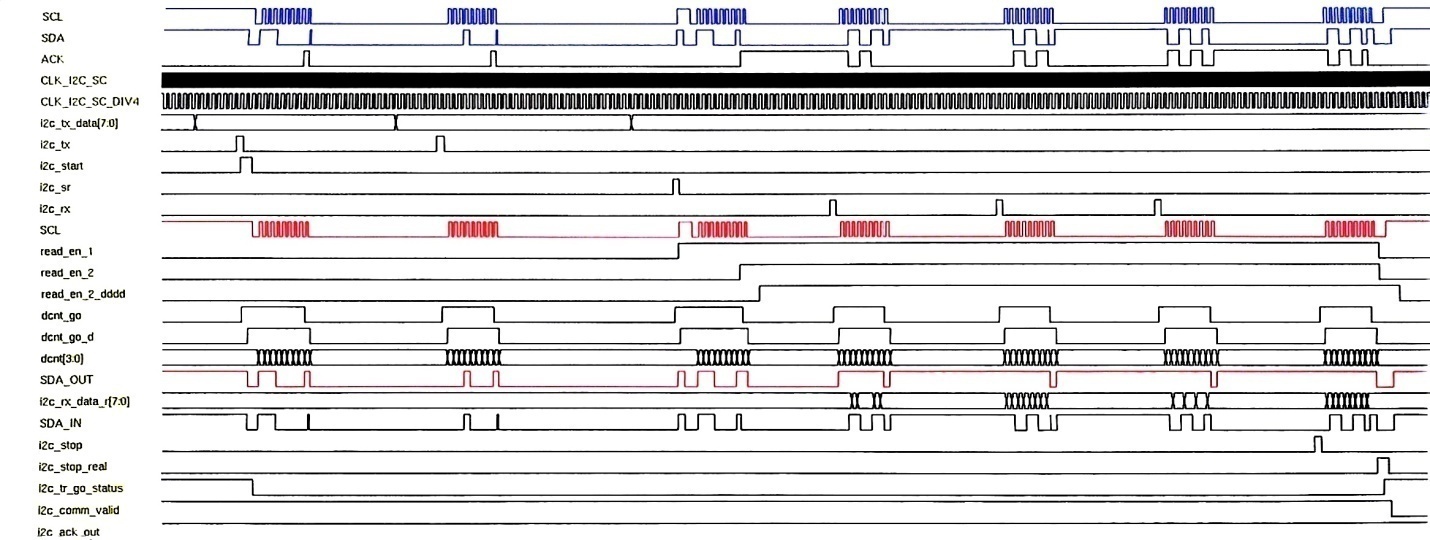


Fig 7. OVUV\_OTUT\_CMP Timing Diagram 2

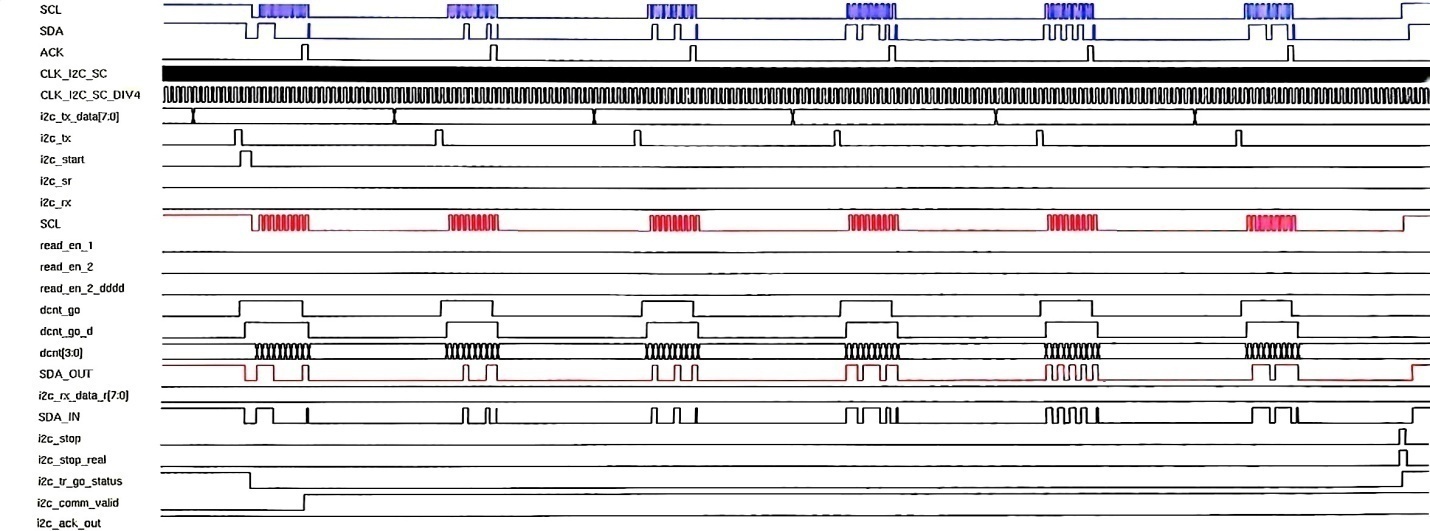


Fig 8. OVUV\_OTUT\_CMP Timing Diagram 3

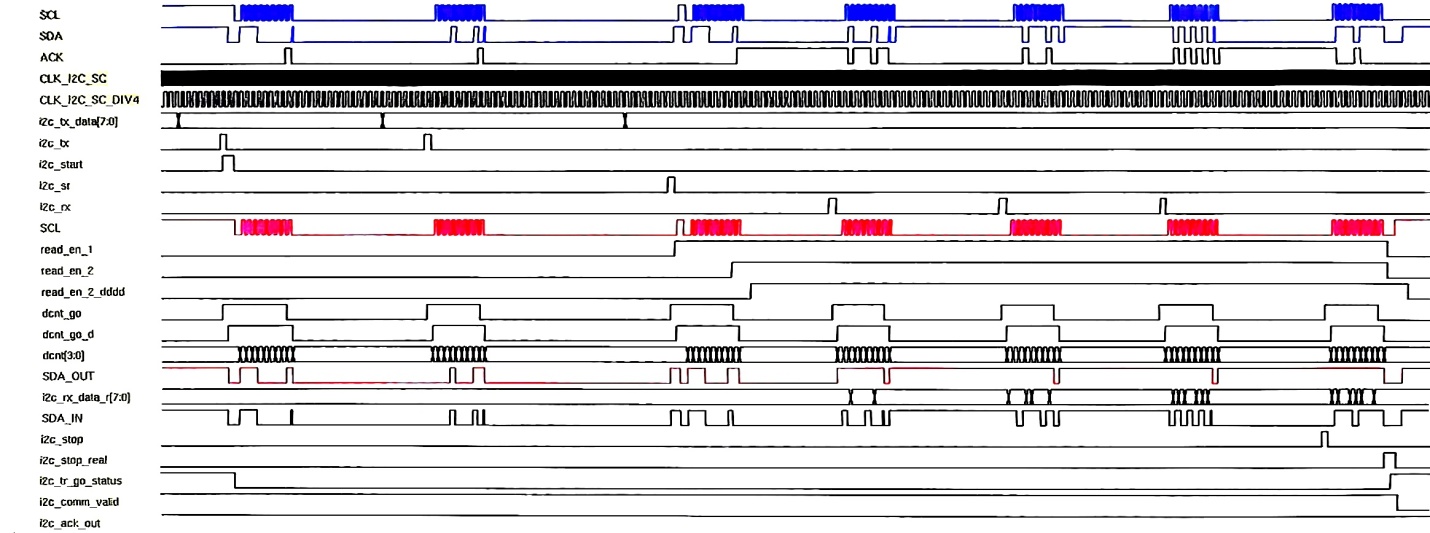


Fig 9. OVUV\_OTUT\_CMP Timing Diagram 4