



eSi-I2C

1 Contents

1	Contents	2
2	Overview	3
3	Hardware Interface	4
4	Software Interface	5
4.1	Register Map	5
4.2	Master Transactions	7
4.3	Slave Transactions	9
4.4	Interrupts	11
5	Revision History	12

2 Overview

The eSi-I2C core supports the following features:

- Multi-master / slave operation.
- Clock stretching.
- 7 and 10-bit addresses.
- Programmable bit rate.
- Configurable TX and RX FIFOs.
- AMBA 3 APB slave interface.
- DMA flow-control interface.

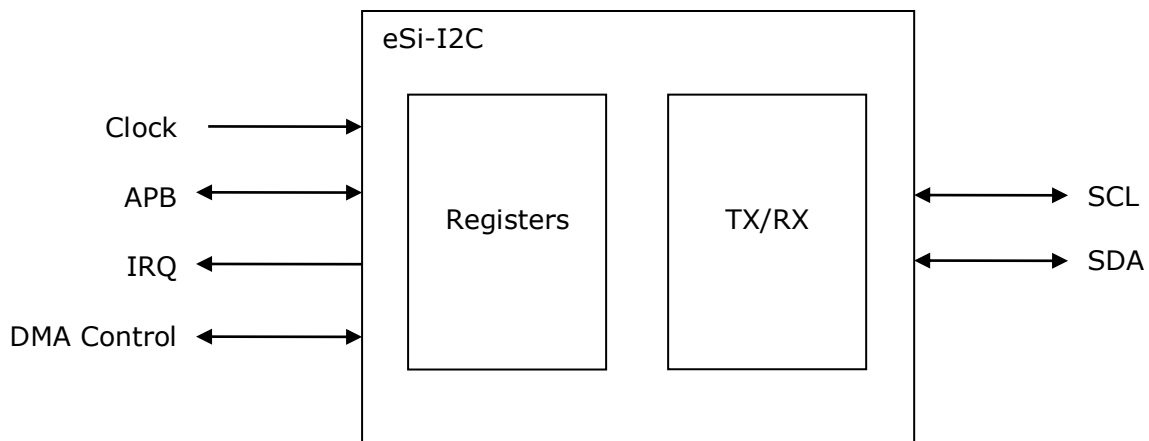


Figure 1: eSi-I2C

3 Hardware Interface

Module Name	cpu_apb_i2c
HDL	Verilog
Technology	Generic
Source Files	cpu_apb_i2c.v, cpu_fifo.v, cpu_peripheral_flow_control.v

Port	Type	Description
rx_fifo_depth	Integer	Specifies the depth of the RX FIFO
tx_fifo_depth	Integer	Specifies the depth of the TX FIFO

Table 1: Parameters

Port	Direction	Width	Description
clk	Input	1	Clock used for transmission and reception. This clock must be enabled when <code>cactive</code> is asserted. This clock must be synchronous to <code>pclk</code> .
pclk	Input	1	APB clock
presetn	Input	1	APB reset, active-low
paddr	Input	8	APB address
psel	Input	1	APB slave select
penable	Input	1	APB enable
pwrite	Input	1	APB write
pwdata	Input	16	APB write data
scl_in	Input	1	I2C clock in
sda_in	Input	1	I2C data in
tx_ack	Input	1	Acknowledges <code>tx_ready</code> after transfer complete
rx_ack	Input	1	Acknowledges <code>rx_ready</code> after transfer complete
cactive	Output	1	Clock active
pready	Output	1	APB ready
prdata	Output	16	APB read data
pslverr	Output	1	APB slave error
scl_out	Output	1	I2C clock out
scl_out_enable	Output	1	I2C clock output enable
sda_out	Output	1	I2C data out
sda_out_enable	Output	1	I2C output enable
interrupt_n	Output	1	Interrupt request, active-low
tx_ready	Output	1	Indicates device can accept new data
rx_ready	Output	1	Indicates device has data to be read

Table 2: I/O Ports

For complete details of the APB signals, please refer to the AMBA 3 APB Protocol v1.0 Specification available at <http://www.arm.com/products/solutions/AMBAHomePage.html>

For details of the I²C-bus specification, please refer to http://www.nxp.com/documents/user_manual/UM10204.pdf

The I2C does not include internal synchronizing flip-flops. These should be implemented externally for the `scl_in` and `sda_in` ports if the transmitting clock domain is asynchronous to `clk`.

4 Software Interface

4.1 Register Map

Register	Address offset	Access	Description
tx_data	0x00	W	Transmit data register
rx_data	0x04	R	Receive data register
status	0x08	R/W	Status register
control	0x0c	R/W	Control register
cycles_per_bit	0x10	R/W	Cycles per bit register
address	0x14	R/W	Slave address

Table 3: Register Map

4.1.1 Transmit Data Register

Data to be transmitted over the I2C interface should be written to the transmit data register. The transmit data register should not be written to while the `TXF` bit in the `status` register is set, otherwise data loss may occur.

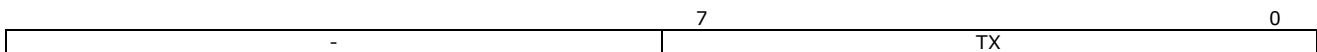


Figure 2: Format of the tx_data register

4.1.2 Receive Data Register

Data that is received over the I2C interface can be read in the receive data register.

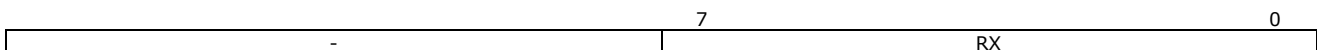


Figure 3: Format of the rx_data register

4.1.3 Status Register

The status register contains a selection of flags that indicate the current status of the I2C. To clear a bit in the status register, write a 1 to it. Writing 0 will leave it unchanged.

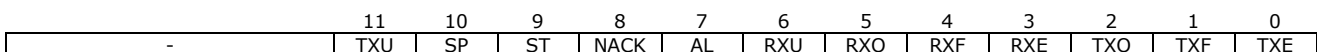


Figure 4: Format of the status register

Register	Values	Description
TXE	0 - Not empty 1 - Empty	Transmit FIFO empty
TXF	0 - Not full 1 - Full	Transmit FIFO full
TXO	0 - No overflow 1 - Overflow	Transmit FIFO overflow
RXE	0 - Not empty 1 - Empty	Receive FIFO empty
RXF	0 - Not full	Receive FIFO full

	1 - Full	
R XO	0 - No overflow 1 - Overflow	Receive FIFO overflow
R XU	0 - No underflow 1 - Underflow	Receive FIFO underflow
AL	0 - Arbitration not lost 1 - Arbitration lost	Arbitration lost (when master)
NACK	0 - ACK 1 - NACK	Transaction not acknowledged
ST	0 - No start 1 - Start	Repeated start condition observed (when slave)
SP	0 - No stop 1 - Stop	Stop condition observed (when slave)
T XU	0 - No underflow 1 - Underflow	Transmit FIFO underflow

Table 4: Fields of the `status` register

4.1.4 Control Register

The control register contains a selection of flags that control the operation of the I2C.

	10	9	8	7	6	5	4	3	2	1	0
-	CS	SPIE	STIE	NIE	ALIE	RXIE	TXIE	NACK	MS	RF	E

Figure 5: Format of the `control` register

Register	Values	Description
E	0 - Disabled 1 - Enabled	Enables the I2C. When disabled, data will not be received or transmitted
RF	0 - Disabled 1 - Enabled	Reset FIFOs. When written with a 1, transmit or receive FIFOs will be cleared
MS	0 - Master 1 - Slave	Controls whether the interface operates as a master or slave
NACK	0 - ACK 1 - NACK	When operating as a slave-receiver, this flag controls whether a received byte is ACKed or NACKed
TXIE	0 - Disabled 1 - Enabled	Transmit interrupt enable
RXIE	0 - Disabled 1 - Enabled	Receive interrupt enable
ALIE	0 - Disabled 1 - Enabled	Arbitration lost interrupt enable
NIE	0 - Disabled 1 - Enabled	NACK interrupt enable
STIE	0 - Disabled 1 - Enabled	Start interrupt enable
SPIE	0 - Disabled 1 - Enabled	Stop interrupt enable
CS	0 - Disabled 1 - Enabled	When operating as a slave, the clock will be stretched if the TX FIFO is empty during a read or the RX FIFO is full during a write

Table 5: Fields of the `control` register

4.1.5 Cycles per Bit Register

The cycles per bit register is a 16-bit integer that specifies how many cycles of the clock, `clk`, the I2C clock, SCL, is held high and low for, when operating as a master. Use of a 16-bit register provides support for a wide range of clock frequencies and bit rates. When operating as a slave, the bit rate is determined by the master. This must not be set to 0.

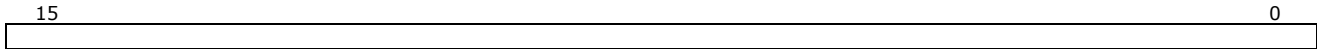


Figure 6: Format of the `cycles_per_bit` register

4.1.6 Slave Address Register

The slave address register contains a 10-bit address that determines the address the I2C interface will respond to (in addition to the general call address 0) when operating as a slave. If the address register is set to 0, then it will respond to any address.

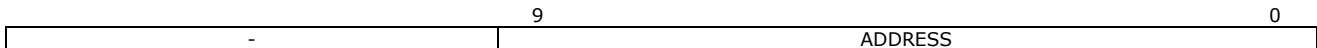


Figure 7: Format of the `address` register

4.2 Master Transactions

4.2.1 Write Transactions

To initiate a write transaction on the I2C bus when operating as a master, two bytes of control data need to be written to the transmit data register before the data to be transmitted, as illustrated in Figure 8: Transmit Data for Write Transactions with 7-bit Addresses and Figure 9: Transmit Data for Write Transactions with 10-bit Addresses.

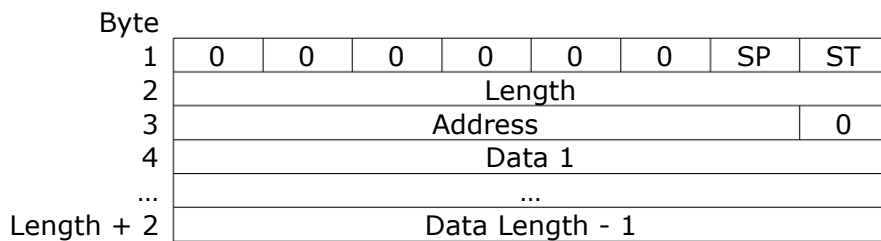


Figure 8: Transmit Data for Write Transactions with 7-bit Addresses

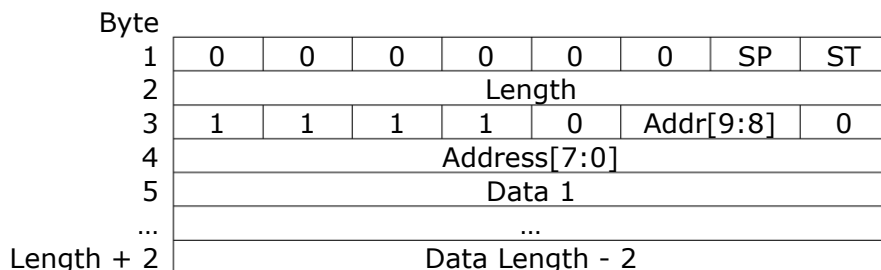


Figure 9: Transmit Data for Write Transactions with 10-bit Addresses

The first byte is a control byte and indicates whether or not start and stop conditions are transmitted before or after the data. If the ST bit is set, a start condition is transmitted before the data. If the SP bit is set, a stop condition is transmitted after the data. Other bits in this byte must be zero.

The second control byte specifies the length of the data to be transmitted, in bytes. This includes any address bytes, but does not include the two control bytes.

Following these control bytes are the address and data that will actually be transmitted on the I2C bus.

4.2.2 Read Transactions

To initiate a read transaction addressing a slave with a 7-bit address, two bytes of control data followed by the address need to be written to the transmit data register, as illustrated in Figure 10: Transmit Data for Read Transactions with 7-bit Addresses.

Byte									
1	0	0	0	0	0	AL	SP	ST	
2	Length								
3	Address								1

Figure 10: Transmit Data for Read Transactions with 7-bit Addresses

The first byte is a control byte and indicates whether or not start and stop conditions are transmitted before or after the data. If the ST bit is set, a start condition is transmitted before the data. If the SP bit is set, a stop condition is transmitted after the data has been received. The AL bit controls whether an ACK is generated for the last byte (as specified by the value in the Length byte). Typically this bit should be set to zero, to indicate that the last byte received should be NACKed, indicating the end of the transaction to the slave. Unused bits in this byte should be zero.

The second control byte specifies the total length of the data in the transaction, including the address byte to be transmitted and the data bytes to be received.

After the control bytes, the address byte must be written to the transmit data register. This will then be transmitted on the I2C bus. Data received from the slave can then be read from the receive data register.

To initiate a read transaction addressing a slave with a 10-bit address, two transactions must effectively take place. First, a write transaction transmits the full 10-bit address. This write transaction is not terminated by a stop condition. Instead, a repeated start condition is generated for the second transaction, which is the actual read transaction. Only the two most-significant bits of the 10-bit slave address are transmitted. The format of the data that must be written to the transmit data register is illustrated in Figure 11: Transmit Data for Read Transactions with 10-bit Addresses.

Byte								
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	1	1	1	1	0	Addr[9:8]		0
4	Address[7:0]							
5	0	0	0	0	0	AL	SP	1
6	Length							
7	1	1	1	1	0	Addr[9:8]		1

Figure 11: Transmit Data for Read Transactions with 10-bit Addresses

4.2.3 NACKs and Lost Arbitration

If a NACK is received from a slave, the current transaction will be terminated and the NACK flag in the status register will be set to 1.

If arbitration is lost during a transaction, the transaction will be terminated and the AL flag in the status register will be set to 1.

In both cases, before starting a new transaction, the transmit and receive FIFOs should be cleared by setting the RF flag in the control register. Following this, the corresponding flags in the status register should be cleared by writing the value 1 to them.

4.3 Slave Transactions

4.3.1 Write Transactions

When operating as a slave device, if a write transaction on the I2C bus is addressed to this device, as determined by the address register, the read/write bit and address bytes are first written to the receive FIFO, followed by the transmitted data as illustrated in Figure 12: Received Data for Write Transactions with 7-bit Addresses and Figure 13: Received Data for Write Transactions with 10-bit Addresses.

Byte	
1	Address 0
2	Data 1
...	...
N + 1	Data N

Figure 12: Received Data for Write Transactions with 7-bit Addresses

Byte							
1	1	1	1	1	0	Addr[9:8]	0
2	Address[7:0]						
3	Data 1						
...	...						
N + 2	Data N						

Figure 13: Received Data for Write Transactions with 10-bit Addresses

4.3.2 Read Transactions

When operating as a slave device, if a read transaction on the I2C bus is addressed to this device, as determined by the `address` register, the read/write bit and address bytes are first written to the receive FIFO, as illustrated in Figure 14: Received Data for Read Transactions with 7-bit Addresses and Figure 15: Received Data for Read Transactions with 10-bit Addresses.



Figure 14: Received Data for Read Transactions with 7-bit Addresses

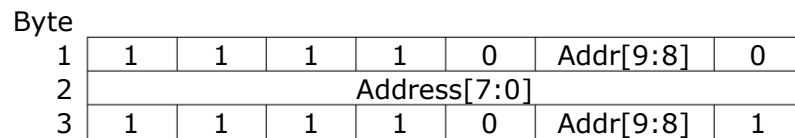


Figure 15: Received Data for Read Transactions with 10-bit Addresses

Data to be returned to the master should simply be written to the transmit FIFO as illustrated in Figure 16: Transmit Data for Read Transactions.

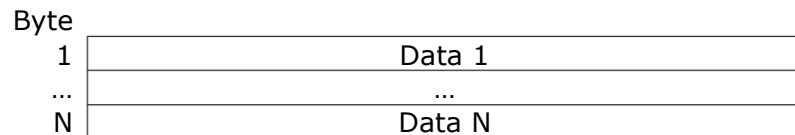


Figure 16: Transmit Data for Read Transactions

4.3.3 ACKs and NACKs

If when a data byte is received the receive FIFO is full and clock stretching is disabled, a NACK will be generated and the receive overflow flag will be set. For address bytes, if there is space in the receive FIFO or clock stretching is enabled, an ACK will be generated. For received data bytes from write transactions, an ACK will only be generated if the `control.NACK` flag is clear and there is space in the receive FIFO or clock stretching is enabled. For read transactions, the ACK is generated by the master, rather than the slave. If the master signals a NACK, the `status.NACK` flag will be set.

4.3.4 Clock Stretching

Clock stretching is enabled when the `control.CS` flag is set. When operating as a slave-receiver, SCL will be held low after an ACK until the received data has been written in to the receive FIFO. When operating as a slave-transmitter, SCL will be held low until data is available in the transmit FIFO, ready for transmission.

Clock stretching is not used for read transactions once `status.NACK` flag is set, as the master will signal a NACK to indicate the end of the read transaction and the slave must not stretch the clock after this, as this could prevent the master from generating the stop condition.

4.4 Interrupts

The I2C supports the following interrupts:

- Transmit interrupt
- Receive interrupt
- Arbitration lost interrupt
- NACK interrupt
- Start interrupt
- Stop interrupt

The transmit interrupt will be raised when the transmit FIFO is empty and the `TXIE` flag in the `control` register is set to 1. This indicates that the transmitter has no data to transmit. The transmit interrupt will only be raised after the ACK/NACK has been received for the byte that was transmitted.

The receive interrupt will be raised when the receiver FIFO is not empty and the `RXIE` flag in the `control` register is set to 1. This indicates that the receiver has received some data.

The arbitration lost interrupt will be raised when the `AL` flag in the `status` register and the `ALIE` flag in the `control` register are both set to 1. This indicates that arbitration was lost to another master and the transaction aborted.

The NACK interrupt will be raised when the `NACK` flag in the `status` register and the `NIE` flag in the `control` register are both set to 1. This indicates that a NACK was received from an I2C slave.

The start interrupt will be raised when the `ST` flag in the `status` register and the `STIE` flag in the `control` register are both set to 1. This indicates that a repeated start condition was observed on the I2C bus while operating as a slave.

The stop interrupt will be raised when the `SP` flag in the `status` register and the `SPIE` flag in the `control` register are both set to 1. This indicates that a stop condition was observed on the bus I2C while operating as a slave.

5 Revision History

Hardware Revision	Software Release	Description
1	1.0.0	Initial release
2	2.4.0	Added I2C slave support. Added <code>status.ST</code> and <code>status.SP</code> register fields. Added <code>control.MS</code> , <code>control.NACK</code> and <code>control.CS</code> register fields. Added <code>control.STIE</code> and <code>control.SPIE</code> register files. Added <code>address</code> register. Added <code>tx_fifo_depth</code> and <code>rx_fifo_depth</code> parameters. Added start and stop interrupts. Added arbitration lost and NACK interrupts. Added <code>tx_ack</code> and <code>rx_ack</code> ports. Added support for 10-bit addressing. Added AL flag. Increased length from 7-bits to 8-bits.

Table 6: Revision History