

HB0090

Corel2C v7.2 Handbook

05 2017





Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

©2017 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi Corporate Headquarters
 One Enterprise, Aliso Viejo,
 CA 92656 USA
 Within the USA: +1 (800) 713-4113
 Outside the USA: +1 (949) 380-6100
 Sales: +1 (949) 380-6136
 Fax: +1 (949) 215-4996
 E-mail: sales.support@microsemi.com
www.microsemi.com

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 9.0

Updated changes related to CoreI2C v7.2.

1.2 Revision 8.0

Updated changes related to CoreI2C v7.1.

1.3 Revision 7.0

Updated:

- Supported Families section.
- Description of CoreI2C's clock stretching capabilities added to Serial and APB Interfaces section.

1.4 Revision 6.0

Updated:

- The "Master Mode Example" section was updated.
- Table 12: Status Register – Master Transmitter Mode was updated.

1.5 Revision 5.0

Updated:

- The Core Version was updated to v6.0 and the Overview section was updated to include information about the multiple I²C channel configuration option.
- The utilization tables were updated.
- Signals and parameters in the "Design Description" section were updated in text and figures for multiple I2C channel functionality.
- Figure 7: Data Read Cycle was updated.
- The Optional SMBus/IPMI Logic section is new.
- Table 7 CoreI2C Internal Register Address Map was updated.
- Table 9: Control Register Bit Fields was updated for R/W properties.
- Table 19 Slave0 Address Register was updated for R/W properties.
- Table 23: Slave1 Address Register was updated for R/W properties.
- The Obfuscated section was updated for SmartDesign.
- Figure 8: CoreI2C Full I/O View is new and Figure 9 CoreI2C SmartDesign Configuration with Callouts to Associated Parameters was updated.

- The Simulation Flows section was updated for SmartDesign. The Testbench Operation and Modification section was updated for the multiple I2C channel configuration.
- Removed Ordering Information section

1.6 Revision 4.0

Updated:

- Figure 11: Example System Using Cortex-M1 with CoreI2C in a Two Channel Configuration was updated
- The Core Version was updated to v5.0. Text, figures, signal names, parameters/generics, and register maps and descriptions have been revised accordingly.
- CoreMP7 references were removed and replaced with Cortex-M1.
- "Ordering Codes" have been included.

1.7 Revision 3.0

Updated:

- The CoreI2C Handbook and CoreSMBus Handbook have been condensed and combined into the current document.

1.8 Revision 2.0

Updated:

- The Supported Families section was added.
- The APB Interface section was updated to include the Cortex-M1 processor.
- The "Use with Core8051s" section was updated to change Core8051 to Core8051s.

1.9 Revision 1.0

Revision 1.0 was the first publication of this document.

Contents

1	Revision History	3
1.1	Revision 9.0	3
1.2	Revision 8.0	3
1.3	Revision 7.0	3
1.4	Revision 6.0	3
1.5	Revision 5.0	3
1.6	Revision 4.0	4
1.7	Revision 3.0	4
1.8	Revision 2.0	4
1.9	Revision 1.0	4
2	Preface	9
2.1	About this Document	9
2.2	Intended Audience	9
3	Introduction	10
3.1	Overview	10
3.2	Features	10
3.3	Core Version	10
3.4	Supported Interfaces	10
3.5	Supported Families	11
3.6	Device Utilization and Performance	11
3.7	Configuration Example	15
4	Functional Description	16
4.1	APB Interface	16
4.2	Input Glitch/Spike Filters	16
4.3	Arbitration and Synchronization Logic	17
4.4	Serial Clock Generator	17
4.5	Address Comparator	17
4.6	Optional SMBus/IPMI Logic	17
5	Operation	18
5.1	I2C Operating Modes	18
5.2	Slave Mode Example	18
5.3	Master Mode Example	19
5.4	SMBus Clock Low Reset Example	19
5.5	Register Map and Descriptions	20
5.5.1	Control Register	21
5.5.2	Status Register	22
5.5.3	Data Register	29

- 5.5.4 SLAVE0 Address Register 30
- 5.5.5 Optional SMBus/IPMI Register 31
- 5.5.6 Optional SLAVE1 Address Register..... 32
- 6 Interface 33**
 - 6.1 I/O Signals 33
 - 6.2 Verilog/VHDL Parameters 35
 - 6.3 Serial and APB Interfaces 36
 - 6.3.1 Serial Interface..... 36
 - 6.3.2 APB Interface 38
- 7 Tool Flow 39**
 - 7.1 License 39
 - 7.1.1 RTL 39
 - 7.2 SmartDesign 39
 - 7.3 Configuring CoreI2C in SmartDesign..... 40
 - 7.4 Simulation Flows 41
 - 7.5 Synthesis in Libero 41
 - 7.6 Place-and-Route in Libero 41
- 8 Testbench 42**
- 9 System Integration 43**

List of Figures

Figure 1 CoreI2C SMBus Application Example.....	15
Figure 2 CoreI2C Block Diagram (Single Channel)	16
Figure 3 SMBus Bus Reset Sequence	19
Figure 4 CoreI2C I/O Signal Diagram	33
Figure 5 Serial Interface Byte Transfer	36
Figure 6 Data Write Cycle	38
Figure 7 Data Read Cycle	38
Figure 8 CoreI2C Full I/O View	39
Figure 9 CoreI2C SmartDesign Configuration with Callouts to Associated Parameters	40
Figure 10 CoreI2C User Testbench	42
Figure 11 CoreI2C System Integration	43

List of Tables

Table 1 CoreI2C Device Utilization and Performance (Slave-only I ² C configuration)	11
Table 2 CoreI2C Device Utilization and Performance (Master/Slave I ² C configuration)	12
Table 3 CoreI2C Device Utilization and Performance (IPMI Master-TX/Slave-RX I ² C configuration)	13
Table 4 CoreI2C Device Utilization and Performance (Master/Slave SMBus configuration).....	14
Table 5 CoreI2C Device Utilization and Performance (13 Channel IPMI configuration).....	15
Table 6 CoreI2C Per Channel Pointer Addressing.....	20
Table 7 CoreI2C Internal Register Address Map	20
Table 8 Control Register	21
Table 9 Control Register Bit Fields	21
Table 10 Status Register	22
Table 11 Status Register Bit Fields.....	22
Table 12 Status Register Master Transmitter Mode.....	23
Table 13 Status Register– Master Receiver Mode.....	24
Table 14 Status Register– Slave Receiver Mode.....	25
Table 15 Status Register– Slave Transmitter Mode.....	27
Table 16 Status Register– Miscellaneous States.....	29
Table 17 Data Register	30
Table 18 Data Register Bit Fields	30
Table 19 Slave0 Address Register	30
Table 20 Slave0 Address Register Bit Fields.....	30
Table 21 SMBus/ IPMI Register	31
Table 22 SMBus/ IPMI Register Bit Fields	31
Table 23 Slave1 Address Register	32
Table 24 Slave1 Address Register Bit Fields.....	32
Table 25 CoreI2C I/O Signal Descriptions	34
Table 26 CoreI2C Parameters/Generics Descriptions.....	35
Table 27 Clock Stretching Periods (Maximum).....	37

2 Preface

2.1 About this Document

This handbook provides details about the CoreI2C DirectCore IP, and how to use it.

2.2 Intended Audience

FPGA designers using Libero® System-on-Chip (SoC), Libero® System-on-Chip (SoC) PolarFire, or Libero Integrated Design Environment (IDE).

3 Introduction

3.1 Overview

CoreI2C provides an APB-driven serial interface, supporting Philips Inter-Integrated Circuit (I2C), SMBus, and PMBus data transfers. Several Verilog/VHDL parameters are available to minimize FPGA fabric area for a given application. CoreI2C also allows for multiple I2C channels, reusing logic across channels to reduce overall tile count.

3.2 Features

CoreI2C supports the following features:

- Conforms to the I2C v2.1 Specification (7-bit addressing format at 100 Kbps and 400 Kbps data rates)
- Supports SMBus v2.0 Specification
- Supports PMBus v1.1 Specification
- Data transfers up to at least 400 kbps nominally; faster rates can be achieved depending on external load and/or I/O pad circuitry
- Modes of operation configurable to minimize size
- Advanced Peripheral Bus (APB) register interface
- Multi-master collision detection and arbitration
- Own address and general call address detection
- Second Slave address decode capability
- Data transfer in multiples of bytes
- SMBus timeout and real-time idle condition counters
- IPMI 3 ms SCL low timeout
- Optional SMBus signals, SMBSUS_N and SMBALERT_N, controllable via APB interface
- Configurable spike suppression width
- Multiple channel configuration option

3.3 Core Version

This handbook is for CoreI2C version 7.2.

3.4 Supported Interfaces

CoreI2C is available with the following interfaces:

- Serial I2C/SMBus/PMBus Interface
- APB Interface for register access

These interfaces are further described in the **Serial and APB Interfaces** section.

3.5 Supported Families

CoreI2C v7.2 is a generic core and supports all the device families.

3.6 Device Utilization and Performance

CoreI2C has been implemented in several of Microsemi’s device families. A summary of various implementation data is listed in [Table 1](#) through [Table 5](#).

Table 1 CoreI2C Device Utilization and Performance (Slave-only I²C configuration)

Family	Tiles			Utilization		Performance MHz
	DFF	4LUT	Total	Device	Total%	
Fusion	80	438	518	AFS600	3.75	66
IGLOO/e	76	449	525	AGLE600V2	3.80	25
ProASIC3/E	80	438	518	M1A3P250	8.43	66
ProASIC ^{PLUS}	87	493	580	APA075	19	46
Axcelerator	87	312	399	AX250	10	105
RTAX-S	87	317	404	RTAX250S	10	108
SmartFusion	80	438	518	A2F500M3G	4.50	66
SmartFusion2	55	258	313	M2S050	0.28	122
IGLOO2	55	258	313	M2GL050	0.28	122
RTG4	55	262	317	RT4G150	0.10	108
PolarFire	55	293	348	MPF300T_ES	0.06	183
<p>Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was standard. Top-level parameters/generics were set as follows: I2C_NUM=1, OPERATING_MODE = 1, BAUD_RATE_FIXED = 1, BAUD_RATE_VALUE = 6, BCLK_ENABLED = 0, GLITCHREG_NUM = 3, SMB_EN = 0, IPMI_EN = 0, FREQUENCY = 30, FIXED_SLAVE0_ADDR_EN = 1, FIXED_SLAVE0_ADDR_VALUE = 0x20, ADD_SLAVE1_ADDRESS_EN = 0, FIXED_SLAVE1_ADDR_EN = 0, FIXED_SLAVE1_ADDR_VALUE = 0.</p>						

Table 2 CoreI2C Device Utilization and Performance (Master/Slave I²C configuration)

Family	Tiles			Utilization		Performance MHz
	DFF	4LUT	Total	Device	Total%	
Fusion	91	582	673	AFS600	4.87	61
IGLOO/e	91	576	667	AGLE600V2	4.82	29
ProASIC3/E	92	566	658	M1A3P250	10.71	79
ProASIC ^{PLUS}	96	568	664	APA075	21	52
Axcelerator	104	410	514	AX250	15	92
RTAX-S	104	413	517	RTAX250S	13	110
SmartFusion	92	576	668	A2F500M3G	5.80	69
SmartFusion2	84	419	503	M2S050	0.44	129
IGLOO2	84	419	503	M2GL050	0.44	129
RTG4	84	418	502	RT4G150	0.17	112
PolarFire	84	363	447	MPF300T_ES	0.07	154
<p>Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was standard. Top-level parameters/generics were set as follows: I2C_NUM=1, OPERATING_MODE = 0, BAUD_RATE_FIXED = 1, BAUD_RATE_VALUE = 6, BCLK_ENABLED = 0, GLITCHREG_NUM = 3, SMB_EN = 0, IPMI_EN = 0, FREQUENCY = 30, FIXED_SLAVE0_ADDR_EN = 1, FIXED_SLAVE0_ADDR_VALUE = 0x20, ADD_SLAVE1_ADDRESS_EN = 0, FIXED_SLAVE1_ADDR_EN = 0, FIXED_SLAVE1_ADDR_VALUE = 0.</p>						

Table 3 CoreI2C Device Utilization and Performance (IPMI Master-TX/Slave-RX I²C configuration)

Family	Tiles			Utilization		Performance MHz
	DFF	4LUT	Total	Device	Total%	
Fusion	108	608	716	AFS600	5.18	65
IGLOO/e	108	615	723	AGLE600V2	5.23	23
ProASIC3/E	108	608	716	M1A3P250	11.65	65
ProASIC ^{PLUS}	116	605	721	APA075	23	54
Axcelerator	132	423	555	AX250	14	81
RTAX-S	128	417	545	RTAX250S	13	101
SmartFusion	108	608	716	A2F500M3G	6.22	65
SmartFusion2	103	369	472	M2S050	0.41	140
IGLOO2	103	366	469	M2GL050	0.41	139
RTG4	103	371	474	RT4G150	0.15	113
PolarFire	103	355	458	MPF300T_ES	0.07	196

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was standard. Top-level parameters/generics were set as follows: I2C_NUM=1, OPERATING_MODE = 2, BAUD_RATE_FIXED = 1, BAUD_RATE_VALUE = 6, BCLK_ENABLED = 0, GLITCHREG_NUM = 3, SMB_EN=0, IPMI_EN = 1, FREQUENCY = 30, FIXED_SLAVE0_ADDR_EN = 1, FIXED_SLAVE0_ADDR_VALUE = 0x20, ADD_SLAVE1_ADDRESS_EN = 1, FIXED_SLAVE1_ADDR_EN = 1, FIXED_SLAVE1_ADDR_VALUE = 0x33.

Table 4 CoreI2C Device Utilization and Performance (Master/Slave SMBus configuration)

Family	Tiles			Utilization		Performance MHz
	DFF	4LUT	Total	Device	Total%	
Fusion	135	705	840	AFS600	6.08	67
IGLOO/e	135	725	60	AGLE600V2	6.22	26
ProASIC3/E	135	705	840	M1A3P250	13.67	67
ProASIC ^{PLUS}	140	735	875	APA075	28	45
Axcelerator	136	456	592	AX250	15	100
RTAX-S	135	475	610	RTAX250S	15	115
SmartFusion	135	705	840	A2F500M3G	7.29	67
SmartFusion2	131	519	650	M2S050	0.57	113
IGLOO2	131	519	650	M2GL050	0.57	113
RTG4	129	533	662	RT4G150	0.21	91
PolarFire	127	492	619	MPF300T_ES	0.10	164
<p>Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was standard. Top-level parameters/generics were set as follows: I2C_NUM=1, OPERATING_MODE = 0, BAUD_RATE_FIXED = 1, BAUD_RATE_VALUE = 6, BCLK_ENABLED = 0, GLITCHREG_NUM = 3, SMB_EN = 1, IPMI_EN = 0, FREQUENCY = 30, FIXED_SLAVE0_ADDR_EN = 1, FIXED_SLAVE0_ADDR_VALUE = 0x20, ADD_SLAVE1_ADDRESS_EN = 0, FIXED_SLAVE1_ADDR_EN = 0, FIXED_SLAVE1_ADDR_VALUE = 0.</p>						

Table 5 CoreI2C Device Utilization and Performance (13 Channel IPMI configuration)

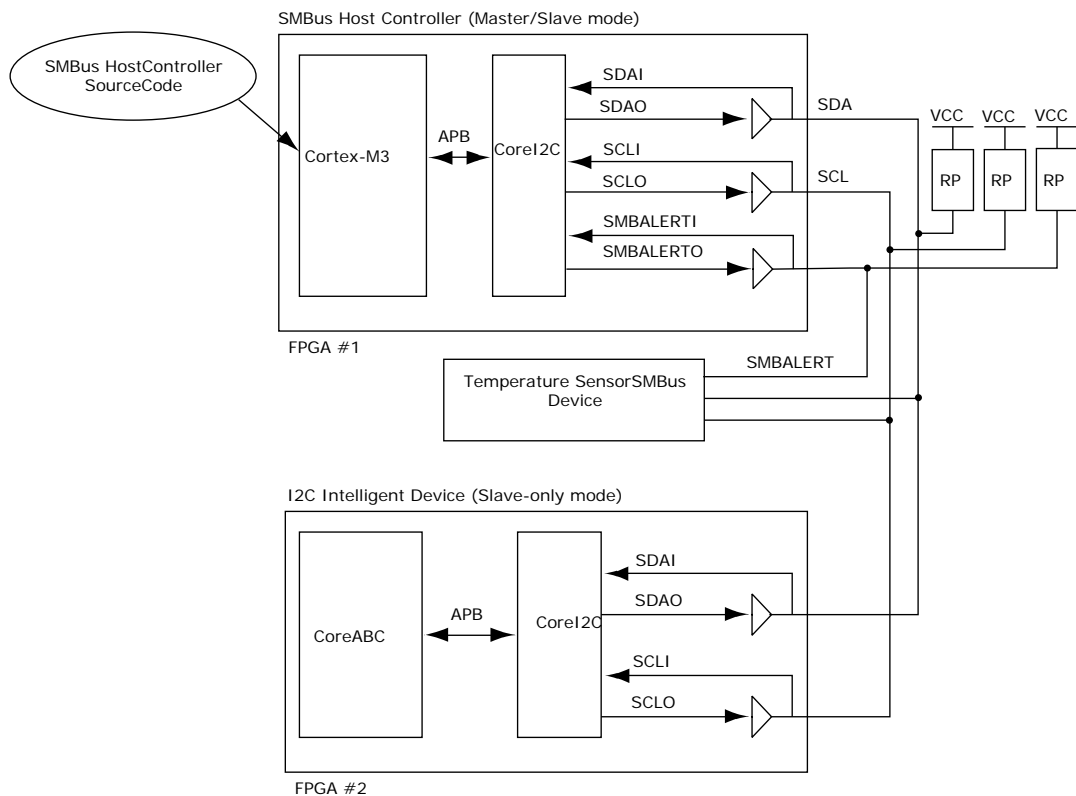
Family	Tiles			Utilization		Performance MHz
	DFF	4LUT	Total	Device	Total%	
SmartFusion2	1,134	4,437	5,571	M2S050	4.94	118
IGLOO2	1,134	4,437	5,571	M2GL050	4.94	118
RTG4	1,134	4,448	5,582	RT4G150	1.84	103
PolarFire	1,134	4,450	5,584	MPF300T_ES	0.93	148

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was standard. Top-level parameters/generics were set as follows: I2C_NUM=13, OPERATING_MODE=2, BAUD_RATE_FIXED=1, BAUD_RATE_VALUE=7, BCLK_ENABLED=1, GLITCHREG_NUM=3, SMB_EN=0, IPMI_EN=1, FREQUENCY=30, FIXED_SLAVE0_ADDR_EN=1, FIXED_SLAVE0_ADDR_VALUE=32, ADD_SLAVE1_ADDRESS_EN=1, FIXED_SLAVE1_ADDR_EN=1, and FIXED_SLAVE1_ADDR_VALUE=20.

3.7 Configuration Example

Figure 1 illustrates a typical application. Cortex-M3, coupled with CoreI2C, masters communication with a SMBus Temperature Sensor slave, and an I²C slave in FPGA #1. In FPGA #2, CoreI2C is configured in Slave-only mode with CoreABC as its control.

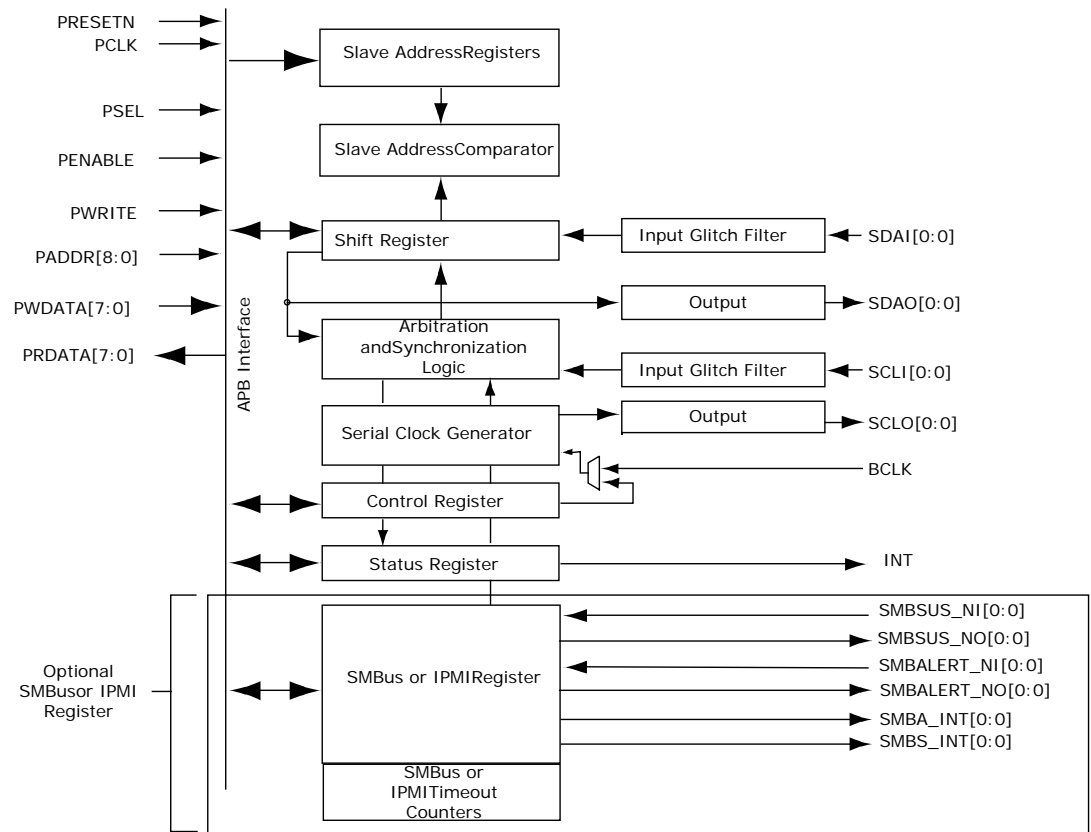
Figure 1 CoreI2C SMBus Application Example



4 Functional Description

CoreI2C, as shown in [Figure 2](#), consists of APB interface registers, serial input spike filters, arbitration and synchronization logic, and a serial clock generation block. The following sections briefly describe each design block.

Figure 2 CoreI2C Block Diagram (Single Channel)



4.1 APB Interface

CoreI2C supports the AMBA Advanced Peripheral Bus (APB) interface, compatible with Microsemi's Cortex-M1, Cortex-M3, and Core8051s processor cores as well as with the CoreABC generic APB-based state machine controller.

The APB registers are defined and usage detailed in the Register Map and Descriptions section.

4.2 Input Glitch/Spike Filters

Input signals are synchronized with the internal clock, PCLK. Spikes shorter than the parameterized glitch register length are filtered out.

4.3 Arbitration and Synchronization Logic

In Master mode, the arbitration logic checks that every transmitted logic '1' actually appears as a logic '1' on the bus. If another device on the bus overrules a logic '1' and pulls the data line low, arbitration is lost and CoreI2C immediately changes from Master transmitter to Slave receiver. The synchronization logic synchronizes the serial clock generator block with the transmitted clock pulses coming from another master device.

The arbitration and synchronization logic also utilizes timeout requirements set forth in the SMBus Specification Version 2.0, or creates a 3 ms IPMI SCL Low Timeout.

4.4 Serial Clock Generator

This programmable clock pulse generator provides the serial bus clock pulses when CoreI2C is in Master mode. The clock generator is switched off when CoreI2C is in Slave mode. The baud rate clock (BCLK) is a pulse-for-transmission speed control signal and is internally synchronized with the clock input. BCLK may be used to set the serial clock frequency when the cr2, cr1, and cr0 bits in the Control Register are set to 111; otherwise, PCLK divisions are used to determine the serial clock frequency. The actual non-stretched serial bus clock frequency can be calculated based on the setting in the cr2, cr1, and cr0 fields of the Control Register and the frequencies of PCLK and BCLK. Refer to Table 8 for configuration.

Note: The SCLO output of a CoreI2C slave must be connected to the SCL line of the I2C bus in order for the slave to implement clock stretching.

4.5 Address Comparator

The comparator checks the received seven-bit slave address with its own slave address, and optionally its own second address, slave1 (for dual-address applications). The comparator also compares the first received eight-bit byte with the general call address (00H). If a match is found, the Status Register is updated and an interrupt is requested.

4.6 Optional SMBus/IPMI Logic

The optional SMBus / IPMI logic includes the SMBus signals, clock-low timeout counters, and reset logic; or when in IPMI mode, the optional 3 ms clock-low timeout counters (an SMBus clock low master reset example is demonstrated in the Operation section). SMBus/IPMI logic includes a top-level prescale counter, which counts in increments of 215 microseconds. A second smaller counter in each channel increments based on the prescale count of 215 microseconds. This design was chosen to reduce overall area at the expense of timeout precision (when the clock-low condition occurs in IPMI mode, the free running 215 microsecond counter may be anywhere in its count). As such, the 3 ms timeout flag will occur between 3.010 and 3.225 ms. The 35 ms SMBus master-holding-clock-low flag will occur between 35.045 and 35.260 ms, and the 25 ms SMBus timeout flag will occur between 25.155 and 25.370 ms.

5 Operation

5.1 I2C Operating Modes

CoreI2C logic can operate in the following four modes:

1. Master Transmitter Mode:
Serial data output through SDA while SCL outputs the serial clock.
2. Master Receiver Mode:
Serial data is received via SDA while SCL outputs the serial clock.
3. Slave Receiver Mode:
Serial data and the serial clock are received through SDA and SCL.
4. Slave Transmitter Mode:
Serial data is transmitted via SDA while the serial clock is input through SCL.

5.2 Slave Mode Example

After setting the `ens1` bit in the Control Register, the core is in the not addressed Slave mode. In Slave mode, the core looks for its own slave address and the general call address. If one of these addresses is detected, the core switches to addressed Slave mode and generates an interrupt request. Then the core can operate as a Slave transmitter or a Slave receiver.

Transfer example:

- Microcontroller sets `ens1` and `aa` bits
- Core receives own address and transfer direction bit set to zero.
- Core generates interrupt request; Status Register = 0x60 ([Table 14](#))
- Microcontroller prepares for receiving data and then clears `si` bit.
- Core receives next data byte and then generates interrupt request. The Status Register contains 0x80 or 0x88 value depending on ACK bit ([Table 14](#)).
- Transfer is continued according to [Table 14](#).

5.3 Master Mode Example

When the microcontroller wishes to become the bus master, the core waits until the serial bus is free. When the serial bus is free, the core generates a start condition and transmits the slave address (Slave that it wishes to control) and transfer direction bit. The core can operate as a Master transmitter or as a Master receiver, depending on the transfer direction bit.

Transfer example:

- Microcontroller sets ens1 and sta bits.
- Core sends START condition and then generates interrupt request; Status Register = 0x08 (Table 12). If Status Register = 0x08, continue with the transmission; else clear the STA bit and continue with the reception.
- Microcontroller writes the Data Register (7-bit slave address and 0) and then clears si bit.
- Core sends Data Register contents and then generates interrupt request. The Status Register contains 0x18 or 0x20 value, depending on received ACK bit..
- Transfer is continued according to Table 11.

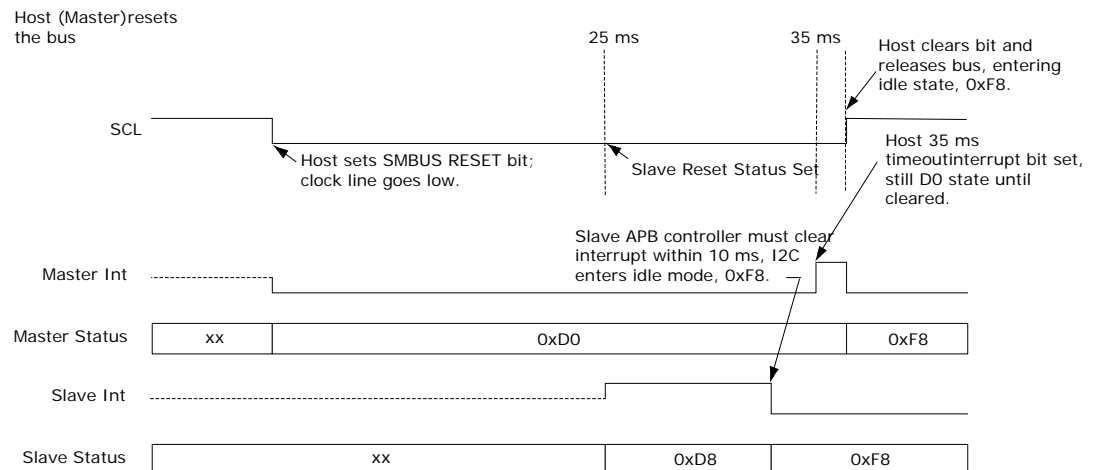
5.4 SMBus Clock Low Reset Example

If the clock line is held low by a Master who has initiated a bus reset with the SMBus register, the following sequence should occur. Refer to Figure 3.

Transfer example:

- The Master device sets SMBUS RESET bit, forcing the clock line low; the master device enters the resetting state, 0xD0, and an interrupt is generated after 35 ms.
- A Slave device will enter the reset state, 0xD8, after 25 ms and an interrupt will be generated. Once the interrupt is asserted, the APB controller of the slave device will need to clear the interrupt within 10 ms per the SMBus Specification v.2.0, and the Slave device will enter the idle state, 0xF8.
- After 35 ms, the Master device’s interrupt will be asserted, and the APB controller of the master device will eventually clear the interrupt, forcing the Master device into the idle state, 0xF8.

Figure 3 SMBus Bus Reset Sequence



5.5 Register Map and Descriptions

PADDR[8:5] bits determine which I2C channel is being addressed, as shown in Table 6. Table 7 defines the register map and reset values of each channel's APB-accessible registers. 0x denotes hexadecimal, 0b denotes binary, and 0d denotes decimal format. "X" implies an unknown condition. "-" implies don't care condition. Type designations: R is read-only, R/W is read/write.

Table 6 CoreI2C Per Channel Pointer Addressing

PADDR[8:5]	Type	Reset Value	Brief Description
Channel ID Value	N/A	N/A	Bits 8 to 5 of PADDR function as address pointers to one of the 16 channels. PADDR[8:5] Channel Number 0000 0 0001 1 1111 15

Note: The channel ID value does not apply to the ADDR0 and ADDR1 registers shown in Table 7. The values in these registers are the same for all channels.

Table 7 CoreI2C Internal Register Address Map

PADDR[4:0]	Register Name	Type	Width	Reset Value	Brief Description
0x00	CTRL	R/W	8	0x00	Control Register; used to configure each I2C channel.
0x04	STAT	R	8	0xF8	Status Register; read-only value yields the current state of the particular I ² C channel.
0x08	DATA	R/W	8	0x00	Data Register; I ² C channel read/write data to/from the serial interface.
0x0C	ADDR0	R/W	8	0x00	Slave0 Address Register; contains the programmable Slave0 address for all channels. Note: The Slave0 Address Register is a single register that is used in all channels. Only PADDR[4:0] are required to write ADDR0; PADDR[8:5] are "don't care" bits.
0x10	SMB	R/W	8	0b01X1X000	SMBus or IPMI Register SMBus Context: Configuration register for SMBus timeouts and reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N. IPMI Context: Enable/Disable IPMI SCL low timeout

0x1C	ADDR1	R/W	8	0x00	<p>Slave1 Address Register; contains the programmable Slave1 address of all channels. When this Slave1 address is enabled yet fixed, the register will have a R/W bit to enable/disable Slave1 comparisons. Only the enable/disable bit will be R/W. The address is write only.</p> <p>Note: The Slave1 Address Register is a single register that is used in all channels. Only the enable/disable bit is R/W. Only PADDR[4:0] are required to write ADDR0; PADDR[8:5] are "don't care" bits.</p>
------	-------	-----	---	------	---

The following sections and tables detail the APB-accessible registers within each CoreI2C channel.

5.5.1 Control Register

The Control Register is described in [Table 8](#) and [Table 9](#). The CPU can read from and write to this 8-bit, directly addressable APB register. Two bits are affected by the CoreI2C: the si bit is set when a serial interrupt is requested and the sto bit is cleared when a STOP condition is present on the bus.

Table 8 Control Register

PADDR[8:5]	Register Name	Type	Width	Reset Value	Description
0x00	CTRL	R/W	8	0x00	Control Register; used to configure each I ² C channel.

Table 9 Control Register Bit Fields

Bits	Name	Type	Description
7	cr2	R/W	Clock rate bit 2; refer to bit 0.
6	ens1	R/W	Enable bit. When ens1 = 0, the sda and scl outputs are in a high impedance state and sda and scl input signals are ignored. When ens1 = 1, the channel is enabled.
5	sta	R/W	The START flag. When sta = 1, the channel checks the status of the serial bus and generates a START condition if the bus is free.
4	sto	R/W	The STOP flag. When sto = 1 and the channel is in a Master mode, a STOP condition is transmitted to the serial bus. This bit is automatically cleared when a stop condition is present on the bus.
3	si	R/W	The Serial Interrupt flag. The si flag is set by the channel whenever there is a serviceable change in the Status Register. After the register has been updated, the si bit must be cleared by APB master. Delaying the clearing of this bit implements clock stretching when operating as a slave transmitter/receiver (Provided that the SCLO output of the slave is connected to the SCL line of the I2C bus). The si bit is directly readable via the APB INTERRUPT signal.

2	aa	R/W	<p>The Assert Acknowledge flag.</p> <p>When aa= 1, an acknowledge (ACK) will be returned when: The "own slave address" has been received. The general call address has been received while the gc bit in the Address register is set. A data byte has been received whilst CoreI2C operates in Master receiver mode. A data byte has been received whilst CoreI2C operates in Slave receiver mode.</p> <p>When aa = 0, a not acknowledge (NACK) will be returned when: A data byte has been received whilst CoreI2C operates in Master receiver mode. A data byte has been received whilst CoreI2C operates in Slave receiver mode</p>																																				
1	cr1	R/W	Serial clock rate bit 1; refer to bit 0.																																				
0	cr0	R/W	<p>Serial clock rate bit 0; Clock Rate is defined as follows:</p> <table border="1"> <thead> <tr> <th>cr2</th> <th>cr1</th> <th>cr0</th> <th>SCL Frequency</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>PCLK frequency/256</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>PCLK frequency/224</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>PCLK frequency/192</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>PCLK frequency/160</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>PCLK frequency/960</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>PCLK frequency/120</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>PCLK frequency/60</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>BCLK frequency/8</td> </tr> </tbody> </table>	cr2	cr1	cr0	SCL Frequency	0	0	0	PCLK frequency/256	0	0	1	PCLK frequency/224	0	1	0	PCLK frequency/192	0	1	1	PCLK frequency/160	1	0	0	PCLK frequency/960	1	0	1	PCLK frequency/120	1	1	0	PCLK frequency/60	1	1	1	BCLK frequency/8
cr2	cr1	cr0	SCL Frequency																																				
0	0	0	PCLK frequency/256																																				
0	0	1	PCLK frequency/224																																				
0	1	0	PCLK frequency/192																																				
0	1	1	PCLK frequency/160																																				
1	0	0	PCLK frequency/960																																				
1	0	1	PCLK frequency/120																																				
1	1	0	PCLK frequency/60																																				
1	1	1	BCLK frequency/8																																				

5.5.2 Status Register

The Status Register is read-only. The status values are listed, depending on mode of operation, in Table 12 through Table 16. Whenever there is a change of state, an INTERRUPT (INT) is asserted. After updating any registers, the APB interface control must clear the INTERRUPT (INT) by clearing the si bit of the Control Register.

Table 10 Status Register

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x04	STAT	R	8	0xF8	Status Register; read-only value yields the current state of each I2C channel.

Table 11 Status Register Bit Fields

Bits	Name	Type	Field Description
7:0	Status	R	Read-Only Status Code. Refer to the tables below for code descriptions based on operating mode.

Table 12 through Table 16 define Status register code descriptions and subsequent action based on the four possible operating modes.

Table 12 Status Register Master Transmitter Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x08	A START condition has been transmitted.	Load SLA + W	–	0	0	–	SLA + W will be transmitted; ACK will be received.
0x10	A repeated START condition has been transmitted.	Load SLA + W	–	0	0	–	SLA + W will be transmitted; ACK will be received.
		or load SLA + R	–	0	0	–	SLA + R will be transmitted; channel will be switched to MST/REC mode.
0xE0	A STOP Condition has been transmitted	No action	–	–	–	–	No action
0x18	SLA + W has been transmitted; ACK has been received.	Load data byte	0	0	0	–	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	–	Repeated START will be transmitted.
		or no action	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
0x20	SLA + W has been transmitted; NACK has been received.						
		or no action	1	0	0	–	Repeated START will be transmitted.
		or no action	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
0x28	Data byte in Data Register has been transmitted; ACK has been received.	Load data byte	0	0	0	–	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	–	Repeated START will be transmitted.
		or no action	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
0x30	Data byte in Data Register has been transmitted; NACK has been received.	No action	1	0	0	–	Repeated START will be transmitted.
		or no action	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x38	Arbitration lost in SLA + R/W or data bytes.	No action	0	0	0	–	The bus will be released; CoreI2C will enter slave mode.
		or no action	1	0	0	–	A start condition will be transmitted when the bus becomes free.
0xD0	SMB_EN = 1: SMBus Master Reset has been activated.	No action	–	–	–	–	Wait 35 ms for interrupt to be set, clear interrupt and proceed to 0xF8 state. Only valid when SMB_EN = 1.
0xD8	IPMI_EN = 1: 3 ms SCL low time has been reached.	No action	–	–	0	–	3 ms SCL low time has been reached. Only valid when IPMI_EN = 1.
<p>Notes:</p> <p>SLA = slave address</p> <p>SLV = slave</p> <p>REC = receiver</p> <p>TRX = transmitter</p> <p>SLA + W = Master sends slave address, then writes data to slave</p> <p>SLA + R = Master sends slave address, then reads data from slave.</p>							

Table 13 Status Register– Master Receiver Mode

Status Code	Status	APB Config Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x08	A START condition has been transmitted.	Load SLA + R	–	0	0	–	SLA + R will be transmitted; ACK will be received.
0x10	A repeated START condition has been transmitted.	Load SLA + R	–	0	0	–	SLA + R will be transmitted; ACK will be received.
		or load SLA + W	–	0	0	–	SLA + W will be transmitted; CoreI2C will be switched to MST/TRX mode.
0x38	Arbitration lost.	No action	0	0	0	–	The bus will be released; CoreI2C will enter slave mode.
		or no action	1	0	0	–	A start condition will be transmitted when the bus becomes free.
0x40	SLA + R has been transmitted; ACK has been received.	No action	0	0	0	0	Data byte will be received; NACK will be returned.
		or no action	0	0	0	1	Data byte will be received; ACK will be returned.
0x48	SLA + R has been	No action	1	0	0	–	Repeated START condition will be transmitted.

Status Code	Status	APB Config Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
	transmitted; NACK has been received.	or no action	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte	0	0	0	0	Data byte will be received; NACK will be returned.
		or read data byte	0	0	0	1	Data byte will be received; ACK will be returned.
0x58	Data byte has been received; NACK has been returned.	Read data byte	1	0	0	–	Repeated START condition will be transmitted.
		or read data byte	0	1	0	–	STOP condition will be transmitted; sto flag will be reset.
		or read data byte	1	1	0	–	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
0xD0	SMB_EN = 1: SMBus Master Reset has been activated.	No action	–	–	0	–	Wait 35 ms for interrupt to be set; clear interrupt and proceed to 0xF8 state. Only valid when SMB_EN = 1.
0xD8	IPMI_EN = 1: 3 ms SCL low time has been reached.	No action	–	–	0	–	3 ms SCL low time has been reached. Only valid when IPMI_EN = 1.
<p>Notes:</p> <p>SLA = slave address</p> <p>SLV = slave</p> <p>REC = receiver</p> <p>TRX = transmitter</p> <p>SLA + W = Master sends slave address, then writes data to slave.</p> <p>SLA + R = Master sends slave address, then reads data from slave.</p>							

Table 14 Status Register– Slave Receiver Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x60	Own SLA + W has been received; ACK has been returned.	No action	–	0	0	0	Data byte will be received and NACK will be returned.
		or no action	–	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA + R/W as master; own SLA + W has been received, ACK returned.	No action	–	0	0	0	Data byte will be received and NACK will be returned.
		or no action	–	0	0	1	Data byte will be received and ACK will be returned.

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x70	General call address (00H) has been received; ACK has been returned.	No action	–	0	0	0	Data byte will be received and NACK will be returned.
		or no action	–	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA + R/W as master; general call address has been received, ACK returned.	No action	–	0	0	0	Data byte will be received and NACK will be returned.
		or no action	–	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLV address; DATA has been received; ACK returned.	Read data byte	–	0	0	0	Data byte will be received and NACK will be returned.
		or read data byte	–	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NACK returned	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with general call address; DATA has been received; ACK returned.	Read data byte	–	0	0	0	Data byte will be received and NACK will be returned.
		or read data byte	–	0	0	1	Data byte will be received and ACK will be returned.
0x98	Previously addressed with general call address; DATA has been received; NACK returned.	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
							recognized; START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xD8	SMB_EN = 1: 25 ms SCL low time has been reached; device must be reset.	no action	–	–	0	–	Slave must proceed to reset state by clearing the interrupt within 10 ms, according to SMBus Specification 2.0. Only valid when SMB_EN = 1.
0xD8	IPMI_EN = 1: 3 ms SCL low time has been reached.	no action	–	–	0	–	3 ms SCL low time has been reached. Only valid when IPMI_EN = 1.
<p>Notes:</p> <p>SLA = slave address</p> <p>SLV = slave</p> <p>REC = receiver</p> <p>TRX = transmitter</p> <p>SLA + W = Master sends slave address, then writes data to slave.</p> <p>SLA + R = Master sends slave address, then reads data from slave.</p>							

Table 15 Status Register– Slave Transmitter Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0xA8	Own SLA + R has been received; ACK has been returned	Load data byte	–	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	–	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA + R/W as master; own SLA + R has been received; ACK has	Load data byte	–	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	–	0	0	1	Data byte will be transmitted; ACK will be received.

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
	been returned.						
0xB8	Data byte has been transmitted; ACK has been received.	Load data byte	–	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	–	0	0	1	Data byte will be transmitted; ACK will be received.
0xC0	Data byte has been transmitted; NACK has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xC8	Last data byte has transmitted; ACK has received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition will be transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode;

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
							own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
0xD8	SMB_EN = 1: 25 ms SCL low time has been reached; device must be reset.	no action	–	–	0	–	Slave must proceed to reset state by clearing the interrupt within 10 ms, according to SMBus Specification 2.0. Only valid when SMB_EN = 1.
0xD8	IPMI_EN = 1: 3 ms SCL low time has been reached.	no action	–	–	0	–	3 ms SCL low time has been reached. Only valid when IPMI_EN = 1.
Notes: SLA = slave address SLV = slave REC = receive TRX = transmitter SLA + W = Master sends slave address, then writes data to slave. SLA + R = Master sends slave address, then reads data from slave.							

Table 16 Status Register– Miscellaneous States

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I ² C Channel
			sta	sto	si	aa	
0x38	Arbitration lost	No action	0	0	0	–	Bus will be released.
		or no action	1	0	0	–	A start condition will be transmitted when the bus becomes free.
0xF8	No relevant state information available; si = 0	No action	No action				Idle
0x00	Bus error during MST or selected slave modes.	No action	0	1	0	–	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the state switched in non-addressed slave mode. Stop Flag is reset.

5.5.3 Data Register

The Data Register (Table 17) contains a byte of serial data to be transmitted or a byte that has just been received. The APB controller can read from and write to this 8-bit, directly addressable register while it is not in the process of shifting a byte (i.e., after an interrupt has been generated).

The bit description in Table 18 is listed in both data context and addressing context. Data context is the 8-bit data format from MSB to LSB. Addressing context is based on a Master sending an address call to a Slave on the bus, along with a direction bit (that is, Master transmit data or receive data from a Slave).

Table 17 Data Register

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x08	DATA	R/W	8	0x00	Data Register; read/write data to/from the serial IF.

Table 18 Data Register Bit Fields

Bits	Name	Type	Data Context Description	Addressing Context Description
7	sd7	R/W	Serial data bit 7 (MSB)	Serial address bit 6 (MSB)
6	sd6	R/W	Serial data bit 6	Serial address bit 5
5	sd5	R/W	Serial data bit 5	Serial address bit 4
4	sd4	R/W	Serial data bit 4	Serial address bit 3
3	sd3	R/W	Serial data bit 3	Serial address bit 2
2	sd2	R/W	Serial data bit 2	Serial address bit 1
1	sd1	R/W	Serial data bit 1	Serial address bit 0 (LSB)
0	sd0	R/W	Serial data bit 0 (LSB)	Direction bit: 0 = write; 1 = read

5.5.4 SLAVE0 Address Register

The SLAVE0 Address Register (ADDR0, [Table 19](#) and [Table 20](#)) is a read/write directly accessible register.

If the parameter FIXED_SLAVE0_ADDR_EN is enabled, the register is read-only.

Table 19 Slave0 Address Register

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x0C	ADDR0	R/W	8	0x00	Slave0 Address Register; contains the programmable Slave0 address of all channels. Note: The Slave0 Address Register is a single register that is used in all channels.

Table 20 Slave0 Address Register Bit Fields

Bits	Name	Type	Description
7	adr6	R/W	Own SLAVE0 address bit 6
6	adr5	R/W	Own SLAVE0 address bit 5
5	adr4	R/W	Own SLAVE0 address bit 4
4	adr3	R/W	Own SLAVE0 address bit 3
3	adr2	R/W	Own SLAVE0 address bit 2
2	adr1	R/W	Own SLAVE0 address bit 1
1	adr0	R/W	Own SLAVE0 address bit 0
0	gc	R/W	General Call Address Acknowledge. If the gc bit is set, the general call address is recognized; otherwise it is ignored.

5.5.5 Optional SMBus/IPMI Register

The SMBus Register contains specific SMBus related functionality and is Read-able or Write-able as defined in Table 22. Configuration register for SMBus timeout reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N. If IPMI mode is selected, then this register reduces to one enable/disable 3 ms IPMI SCL Low timeout.

Table 21 SMBus/ IPMI Register

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x10	SMB	R/W	8	0b01X1X000	SMBus or IPMI Register SMBus Context: Configuration register for SMBus timeouts and reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N. IPMI Context: Enable/Disable IPMI SCL low timeout

Table 22 SMBus/ IPMI Register Bit Fields

Bits	Name	Type	SMBus Context (SMB_EN =1)	IPMI Context (IPMI_EN =1)
7	SMBus_Reset	W	Writing a one to this bit will force the clock line low until 35 ms has been exceeded, thus resetting the entire bus as per the SMBus Specification Version 2.0. Usage: When the channel is used as a host controller (master), the user can decide to reset the bus by holding the clock line low 35ms. Slaves must react to this event and reset themselves.	Not used.
6	SMBSUS_NO	R/W	R/W SMBSUS_NO control bit; used in master/host mode to force other devices into power down / suspend mode. Active low. SMBSUS_NO and SMBSUS_NI are separate signals (not Wired-AND). If the CoreI2C is part of a host-controller, SMBSUS_NO could be used as an output; if CoreI2C is a slave to a host-controller that has implemented SMBSUS_N, then only SMBSUS_NI's status would be relevant.	Not used.
5	SMBSUS_NI	R	Read-only status of SMBSUS_NI signal. SMBSUS_NO and SMBSUS_NI are separate signals (not Wired-AND). If the CoreI2C is part of a host-controller, SMBSUS_NO could be used as an output; if CoreI2C is a slave to a host-controller that has implemented SMBSUS_N, then only SMBSUS_NI's Status would be relevant.	Not used.
4	SMBALERT_NO	R/W	Read/Write SMBALERT_NO control bit; used in slave/device mode to force communication with the master/host. Wired-AND.	Not used.
3	SMBALERT_NI	R	Read-only Status of SMBALERT_NI signal. Wired-AND.	Not used.
2	SMB_IPMI_EN	R/W	0: SMBus timeouts and status logic disabled, i.e., standard I2C bus operation; 1: SMBus timeouts and status logic enabled.	0: IPMI timeout and status logic disabled, i.e., standard I2C bus operation; 1: IPMI timeout and

Bits	Name	Type	SMBus Context (SMB_EN =1)	IPMI Context (IPMI_EN =1)
				status logic enabled.
1	SMBSUS_IE	R/W	0: SMBSUS Interrupt signal (SMBS) disabled. 1: SMBSUS Interrupt signal (SMBS) enabled.	Not Used.
0	SMBALERT_IE	R/W	0: SMBSUS Interrupt signal (SMBA) disabled. 1: SMBSUS Interrupt signal (SMBA) enabled.	Not Used.

5.5.6 Optional SLAVE1 Address Register

The SLAVE1 Address Register (ADDR1, Table 23 and Table 24) is an 8-bit read/write directly accessible register with two separate contexts depending on parameter configuration. If the parameter FIXED_SLAVE1_ADDR_EN is enabled, the register is read-only.

Table 23 Slave1 Address Register

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x1C	ADDR1	R/W	8	0x00	Slave1 Address Register; contains the programmable Slave1 address of all channels. When this Slave1 address is enabled yet fixed, the register will have a R/W bit to enable/disable Slave1 comparisons. Note: The Slave1 Address Register is a single register that is used in all channels.

Table 24 Slave1 Address Register Bit Fields

Bits	Name	Type	Enabled, APB accessible SLAVE1 Context (ADD_SLAVE1_ADDRESS_EN = 1 AND FIXED_SLAVE1_ADDR_EN = 0)	Enabled, Fixed SLAVE1 Context (ADD_SLAVE1_ADDRESS_EN = 1 AND FIXED_SLAVE1_ADDR_EN = 1)
7	adr6	R/W	Own SLAVE1 address bit 6	Not Used.
6	adr5	R/W	Own SLAVE1 address bit 5	Not Used.
5	adr4	R/W	Own SLAVE1 address bit 4	Not Used.
4	adr3	R/W	Own SLAVE1 address bit 3	Not Used.
3	adr2	R/W	Own SLAVE1 address bit 2	Not Used.
2	adr1	R/W	Own SLAVE1 address bit 1	Not Used.
1	adr0	R/W	Own SLAVE1 address bit 0	Not Used.
0	GC_or_EnAdr	R/W	General Call Address Acknowledge. If the gc bit is set, the general call address is recognized; otherwise it is ignored.	1: Enable the Fixed SLAVE1 Address comparisons. 0: Disable SLAVE1 Address comparisons.

6 Interface

6.1 I/O Signals

The port signals for the CoreI2C macro are illustrated in Figure 4 and defined in Table 25.

Figure 4 CoreI2C I/O Signal Diagram

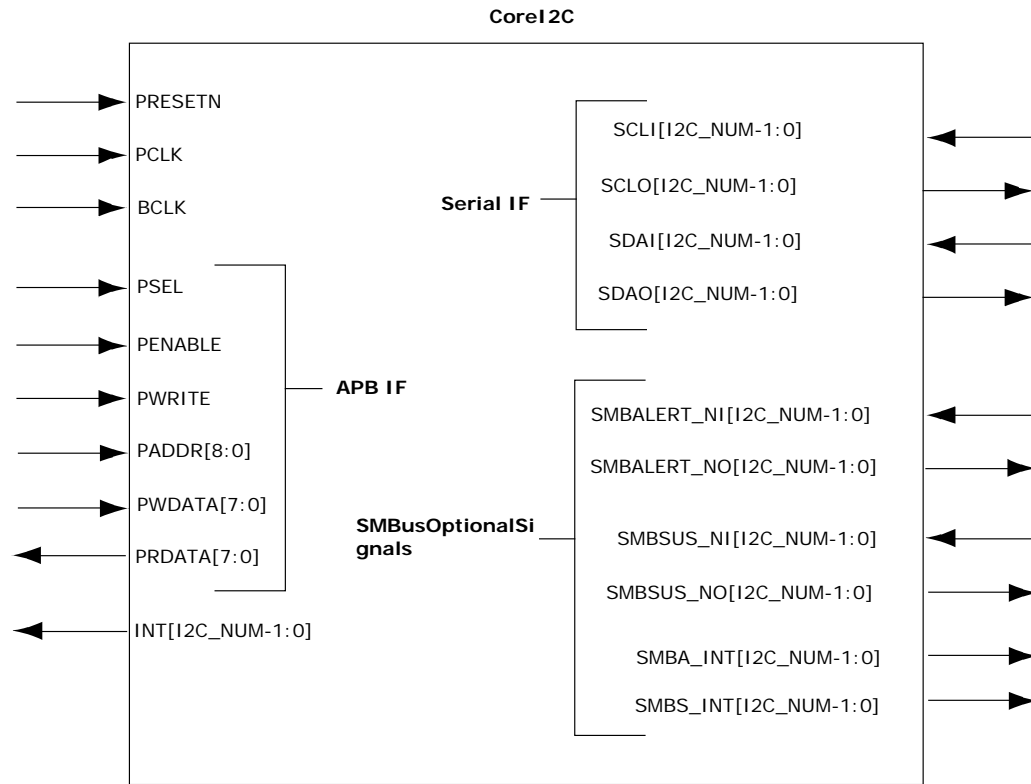


Table 25 CoreI2C I/O Signal Descriptions

Name	Type	Description
APB Interface		
PCLK	Input	APB System Clock; Reference clock for all internal logic
PRESETN	Input	APB active low asynchronous reset.
PADDR[8:0]	Input	APB address bus bits 4 to 0; address internal registers. Bits 8 to 5 function as address pointers to one of the 16 channels.
PSEL	Input	APB Select; select signal to registers for APB reads and writes.
PENABLE	Input	APB Enable. This signal indicates the second cycle of an APB transfer.
PWRITE	Input	APB Write/Read. If high, a write occurs when an APB transfer takes place. If low, a read takes place.
PWDATA[7:0]	Input	APB write data
PRDATA[7:0]	Output	APB read data
INT[I2C_NUM-1:0]	Output	Interrupt output; monitors status register.
SMBA_INT[I2C_NUM-1:0]	Output	Optional (if SMBus Enabled) interrupt output; monitors assertion of SMBALERT_NI. Level sensitive; hence only the deassertion of SMBALERT_NI will clear the interrupt.
SMBS_INT[I2C_NUM-1:0]	Output	Optional (if SMBus Enabled) interrupt output; monitors assertion of SMBSUS_NI. Level sensitive; hence only the deassertion of SMBALERT_NI will clear the interrupt.
Serial Interface		
SCLI[I2C_NUM-1:0]	Input	Wired-AND serial clock input
SCLO[I2C_NUM-1:0]	Output	Wired-AND serial clock output
SDAI[I2C_NUM-1:0]	Input	Wired-AND serial data input
SDAO[I2C_NUM-1:0]	Output	Wired-AND serial data output
SMBus Optional Signal		
SMBALERT_NI[I2C_NUM-1:0]	Input	Wired-AND interrupt signal input; used in Master/Host mode to monitor if slave/devices want to force communication with the host.
SMBALERT_NO[I2C_NUM-1:0]	Output	Wired-AND interrupt signal input; used in Slave/device mode if the core wants to force communication with a host.
SMBSUS_NI[I2C_NUM-1:0]	Input	Suspend Mode signal input; used if core is Slave/device. Not a Wired-AND signal.
SMBSUS_NO[I2C_NUM-1:0]	Output	Suspend Mode signal output; used if core is the Master/host. Not a Wired-AND signal.
Other Signals		
BCLK	Input	Pulse for SCL speed control. Used only if the configuration bits cr2, cr1, and cr0 are set to 111 in the Control Register. Otherwise, SCL is derived from PCLK.
Note: All signals are active high (logic 1) unless otherwise noted		

6.2 Verilog/VHDL Parameters

CoreI2C has parameters (Verilog) or generics (VHDL) for configuring the RTL code, described in [Table 26](#). All parameters and generics are integer types.

Table 26 CoreI2C Parameters/Generics Descriptions

Parameter Name	Valid Range	Default	Description
I2C_NUM	1 to 16	1	Number of I2C channels
FREQUENCY	1 to 255	30	PCLK frequency value in MHz. This parameter is only necessary to configure optional SMBus or IPMI timeout counters.
OPERATING_MODE	0 to 3	0	0: Full Master/Slave Tx/Rx modes. 1: Slave Tx/RX modes only. 2: Master Tx and Slave Rx modes only. 3: Slave Rx mode only.
BCLK_ENABLED	0 or 1	1	0: BCLK input is disabled, reducing tile count. 1: BCLK input is enabled.
BAUD_RATE_FIXED	0 or 1	0	0: Baud rate value (bits cr2, cr1, and cr0 in the Control Register) modified by an APB-accessible register. 1: Baud rate value [bits cr2, cr1, and cr0 in the Control Register) is fixed to the parameter BAUD_RATE VALUE, reducing tile count.
BAUD_RATE_VALUE	0 to 7	0	Fixed Baud Rate Values Bit Value: SCL Frequency: 000 PCLK frequency/256 001 PCLK frequency/224 010 PCLK frequency/192 011 PCLK frequency/160 100 PCLK frequency/960 101 PCLK frequency/120 110 PCLK frequency/60 111 BCLK frequency/8
SMB_EN	0 or 1	0	1: Generates the SMBus logic: SMBus register, real-time checks and timeout values. 0: SMBus logic not generated.
IPMI_EN	0 or 1	0	1: Generates 3 ms SCL Low IPMI Required Timeout Counter with error status and interrupt. 0: IPMI Timeout Counter not generated.

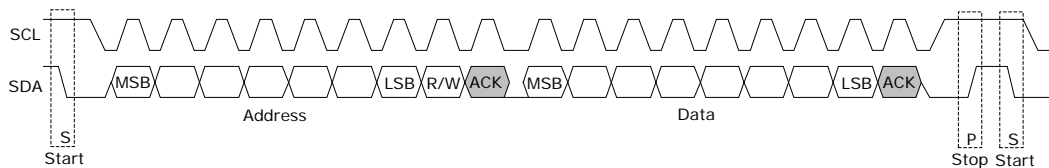
GLITCHREG_NUM	3 to 15	3	<p>Number of registers in the Glitch Filter. Correct value to meet I2C fast mode (400 kbps) and fast mode plus (1 Mbps). 50 ns spike suppression will depend on the PCLK frequency.</p> <p>Guideline:</p> <table border="1"> <thead> <tr> <th>PCLK Freq (MHz) Suppression</th> <th>GlitchReg_Num for 50 ns or Less Spike</th> </tr> </thead> <tbody> <tr> <td>Freq <= 60</td> <td>3</td> </tr> <tr> <td>60 < Freq <= 80</td> <td>4</td> </tr> <tr> <td>80 < Freq <= 100</td> <td>5</td> </tr> <tr> <td>100 < Freq <= 120</td> <td>6</td> </tr> <tr> <td>120 < Freq <= 140</td> <td>7</td> </tr> <tr> <td>140 < Freq <= 160</td> <td>8</td> </tr> <tr> <td>160 < Freq <=180</td> <td>9</td> </tr> <tr> <td>180 < Freq <= 200</td> <td>10</td> </tr> </tbody> </table>	PCLK Freq (MHz) Suppression	GlitchReg_Num for 50 ns or Less Spike	Freq <= 60	3	60 < Freq <= 80	4	80 < Freq <= 100	5	100 < Freq <= 120	6	120 < Freq <= 140	7	140 < Freq <= 160	8	160 < Freq <=180	9	180 < Freq <= 200	10
PCLK Freq (MHz) Suppression	GlitchReg_Num for 50 ns or Less Spike																				
Freq <= 60	3																				
60 < Freq <= 80	4																				
80 < Freq <= 100	5																				
100 < Freq <= 120	6																				
120 < Freq <= 140	7																				
140 < Freq <= 160	8																				
160 < Freq <=180	9																				
180 < Freq <= 200	10																				
FIXED_SLAVE0_ADDR_EN	0 or 1	0	<p>0: SLAVE0 address set via APB SLAVE0 Address register. 1: SLAVE0 address is hardcoded, reducing tile count.</p>																		
FIXED_SLAVE0_ADDR_VALUE	0x00 to 0x7F	0	Hardcoded SLAVE0 address value.																		
ADD_SLAVE1_ADDRESS_EN	0 or 1	0	<p>0: SLAVE1 address is not enabled. 1: SLAVE1 address is enabled.</p>																		
FIXED_SLAVE1_ADDR_EN	0 or 1	0	<p>0: SLAVE1 address set via optional APB SLAVE1 Address register. 1: SLAVE1 address is hardcoded, reducing tile count.</p>																		
FIXED_SLAVE1_ADDR_VALUE	0x00 to 0x7F	0	Hardcoded SLAVE1 address value																		

6.3 Serial and APB Interfaces

6.3.1 Serial Interface

A typical I²C/IPMI/SMBus/PMBus 8-bit data transfer cycle is shown in Figure 5. A Master start condition is signalled by the SDA line going low while the SCL line is high. After a start condition, the master sends a slave address along with a read or write bit. The addressed slave acknowledges its address with an ACK, and then multiple bytes can be transferred with an ACK/NACK for each byte. Eventually the Master asserts a stop condition, which occurs when the SDA line goes high while the SCL line is high.

Figure 5 Serial Interface Byte Transfer



Note: A user of CoreI2C must configure the system (logic, I/O pads, external circuitry and pull-up resistors) to ensure that the serial interface timings adhere to a given I²C/SMBus/PMBus specification.

To adhere to additional SMBus/PMBus Hold times and Minimum Clock High Times, configure PCLK to be within the 5 MHz to 20 MHz range. Additionally, choose a Baud Rate Value so that the serial SCL clock will transfer data at or near the maximum frequency of 100 KHz (FSMB-max) to ensure that other potential clock stretching devices on the bus will not slow the clock frequency to below the minimum allowed SMBus clock of 10 kHz (FSMB-min).

If a significant difference exists between the SCL period configured and the SCL period measured, the drive current of the I2C I/O's must be increased, to minimize the fall times for the I2C bus. The required drive strength is application specific and varies with bus capacitances.

CoreI2C supports clock-stretching when operating as a slave transmitter/receiver, allowing slaves to force communicating masters into wait states if extra processing time is required. Slaves perform clock stretching by holding the SCL line low after a master drives the SCL line low, exploiting the I2C clock synchronization feature to provide flow control. The period, which a CoreI2C slave can hold the SCL line low is limited when using IPMI, SMBus, or PMBus as follows:

Table 27 Clock Stretching Periods (Maximum)

Mode	Max Low Period (Stretch Period)
IPMI	3 ms
SMBUS	100 μ s
PMBUS	25 ms

A CoreI2C instance configured as either a slave transmitter or receiver can be forced to implement clock stretching, by delaying the clearing of the si bit in the Control Register. The CoreI2C slave instance drives the SCLO output low, whilst the si bit remains set in the Control Register.

Note: The SCLO output of a CoreI2C slave must be connected to the SCL line of the I2C bus in order for a slave to implement clock stretching.

For detailed timing information, refer to the I²C/IPMI/SMBus/PMBus specifications directly.

6.3.2 APB Interface

Figure 6 and Figure 7 depict typical write cycle and read cycle timing relationships relative to the system clock.

Figure 6 Data Write Cycle

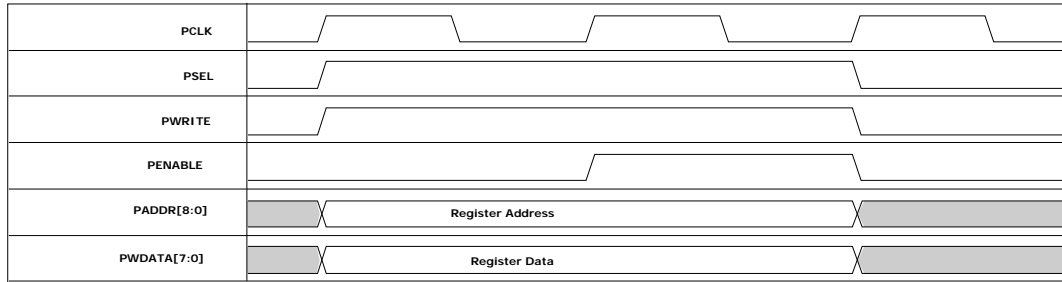


Figure 7 Data Read Cycle



7 Tool Flow

7.1 License

No license is required to use this core.

7.1.1 RTL

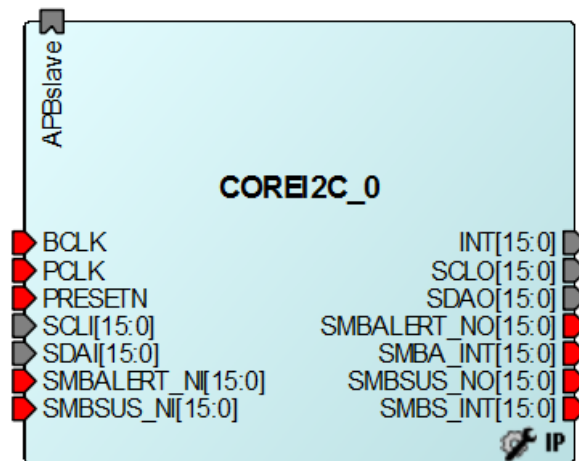
Complete RTL source code is provided for the core and testbench.

7.2 SmartDesign

CoreI2C is pre-installed in the SmartDesign IP deployment design environment or downloaded from the online repository. [Figure 8](#) shows an example instantiated. The core can be configured using the configuration GUI within SmartDesign, as shown in [Figure 9](#).

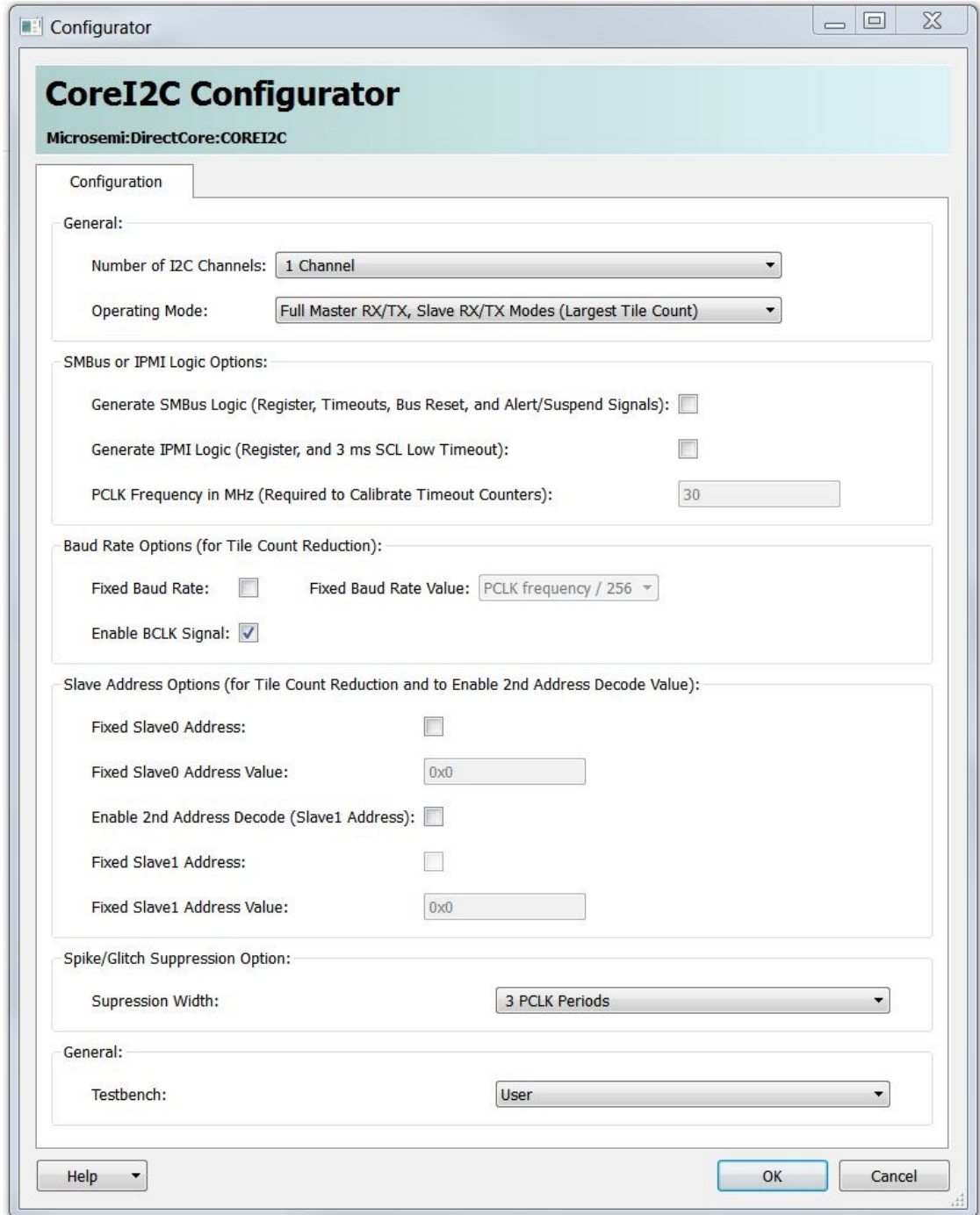
For information on using SmartDesign to instantiate and generate cores, refer to [Libero SoC User Guide](#), [Libero IDE User Guide](#), or [Libero SoC PolarFire User Guide](#).

Figure 8 CoreI2C Full I/O View



7.3 Configuring CoreI2C in SmartDesign

Figure 9 CoreI2C SmartDesign Configuration with Callouts to Associated Parameters



7.4 Simulation Flows

The User Testbench for CoreI2C is included in all releases.

To run simulations, select the User Testbench flow within SmartDesign and click **Save & Generate** on the Generate pane. The User Testbench is selected through the Core Testbench Configuration GUI.

When SmartDesign generates the Libero project, it will install the user testbench files.

To run the user testbench, set the design root to the **CoreI2C instantiation** in the Libero design hierarchy pane and click the **Simulation** icon in the Libero Design Flow window. This will invoke ModelSim® and automatically run the simulation.

7.5 Synthesis in Libero

To run synthesis on the CoreI2C, set the design root to the IP component instance and click on Synthesize in Libero Design Flow pane. This will invoke Synplify Pro and automatically runs the synthesis.

7.6 Place-and-Route in Libero

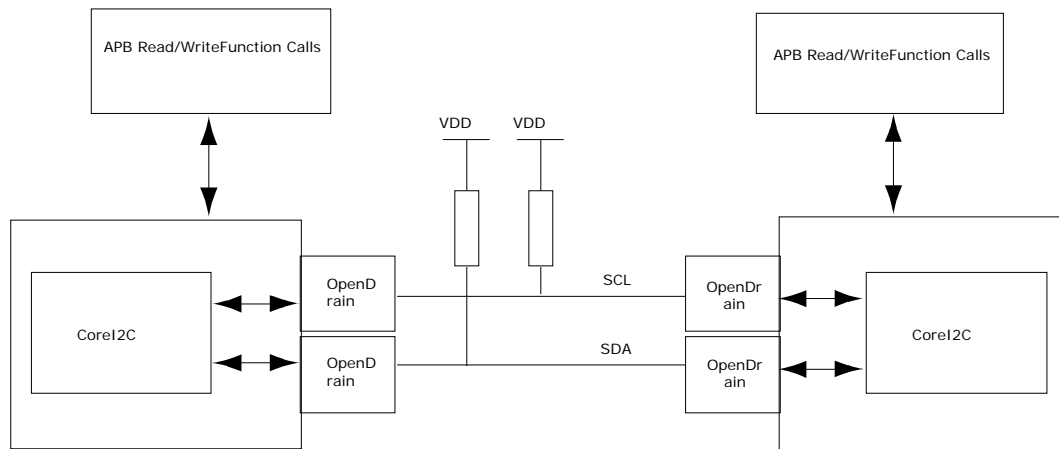
After design is synthesized, click Place and Route in the Libero Design Flow pane to run place and route on the CoreI2C. No special place and route settings are required.

8 Testbench

As shown in [Figure 10](#), two instantiations of the CoreI2C macro are connected to an I²C bus. The second CoreI2C instance is configured in multi-channel mode and uses the 13th channel. The top-level testbench (*tb_user_corei2c*) includes the open drain (WIRED-AND) connections. The testbench utilizes simple APB read/write function calls to initialize each module and send example transmit bytes from instance0 to instance1 across the I²C serial bus. After each transmission, APB read checks are performed to verify valid byte transfers.

Note: The user testbench does not import the user’s own configuration parameters; only a single suite of predefined parameters are tested, some of which may be altered directly in the *tb_user_corei2c.v* or *tb_user_corei2c.vhd* file.

Figure 10 CoreI2C User Testbench



9 System Integration

This section provides hints to ease the integration of CoreI2C.

- The design created using the CoreRISCV_AXI4 softcore processor and MSS subsystem.
- INIT_DONE signal of MSS subsystem is used for all resets in the design.
- Two CoreI2C instances are used in the design and is configured in different modes. CoreI2C_0 is configured in master mode and CoreI2C_1 is configured in slave mode.
- 83 MHz PCLK required for the CoreI2C_0 and CoreI2C_1 is driven FIC_0_CLK of MSS subsystem.
- CoreRISCV based application is used to transmit data from CoreI2C_0 and receive data which is looped back at board level from CoreI2C_1 (Slave).

Figure 11 CoreI2C System Integration

