CoreI2C v6.0

Handbook

Actel
POWER MATTERS

Printed in the United States of America

Part Number: 50200090-5

Release: November 2009

Trademarks

Table of Contents

Introduction

Core Overview

Intended Use

CoreI2C provides an APB-driven serial interface, supporting I^2C, 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 I^2C channels, reusing logic across channels to reduce overall tile count.

Key Features

- Conforms to the Philips Inter-Integrated Circuit (I^2C) 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 IF
- Configurable spike suppression width
- Multiple channel configuration option

Core Version

This handbook supports CoreI2C version 6.0.

Supported Interfaces

CoreI2C is available with the following interfaces:

- Serial I^2C/SMBus/PMBus Interface
- APB Interface for register access

These interfaces are further described in the "Serial and APB Interfaces" section on page 14.

Utilization and Performance

CoreI2C has been implemented in several of Actel's device families using standard speed grades. A summary of various implementation data is listed in Table 1 through Table 5 on page 8.

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

Family	Tiles			Utilization		Performance MHz
	Sequential	Combinatorial	Total	Device	Total %	
Fusion	51	310	361	AFS600	2.6	130
IGLOO®/e	51	310	361	AGLE600V2	2.6	54
ProASIC®3/E	51	310	361	M1A3P250	5.9	127
ProASIC PLUS®	58	355	413	APA075	13	68
Axcelerator®	58	199	257	AX250	6.1	135
RTAX-S	58	299	257	RTAX250S	6.1	101

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. 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 = 0, 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.

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

Family	Tiles			Utilization		Performance MHz
	Sequential	Combinatorial	Total	Device	Total %	
Fusion	73	451	524	AFS600	3.8	116
IGLOO/e	73	451	524	AGLE600V2	3.8	52
ProASIC3/E	73	451	524	M1A3P250	8.5	125
ProASIC PLUS	81	499	580	APA075	18.9	69
Axcelerator	82	303	385	AX250	9.1	135
RTAX-S	82	303	385	RTAX250S	9.1	100

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. 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 = 0, 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.

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

Family	Tiles			Utilization		Performance MHz
	Sequential	Combinatorial	Total	Device	Total %	
Fusion	92	492	584	AFS600	4.2	121
IGLOO/e	92	492	584	AGLE600V2	4.2	52
ProASIC3/E	92	492	584	M1A3P250	9.5	118
ProASIC PLUS	96	556	652	APA075	21	65
Axcelerator	101	325	426	AX250	10	111
RTAX-S	101	325	426	RTAX250S	10	86

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. 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
	Sequential	Combinatorial	Total	Device	Total %	
Fusion	117	587	704	AFS600	5.1	112
IGLOO/e	117	587	704	AGLE600V2	5.1	46
ProASIC3/E	117	587	704	M1A3P250	11.5	111
ProASIC PLUS	125	673	798	APA075	26	54
Axcelerator	127	400	527	AX250	12	109
RTAX-S	127	400	527	RTAX250S	12	80

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. 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.

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

Family	Tiles			Utilization		Performance MHz
	Sequential	Combinatorial	Total	Device	Total %	
Fusion	989	6,001	6,990	AFS600	51	97
IGLOO/e	989	6,001	6,990	AGLE600V2	51	44
ProASIC3/E	989	6,001	6,990	M1A3P600	51	105
ProASIC^{PLUS}	1,099	6,887	7,986	APA600	37	47
Axcelerator	1,166	4,082	5,248	AX1000	29	69
RTAX-S	1,166	4,082	5,248	RTAX1000	29	64

Note: Data in this table were achieved using the Verilog RTL with typical synthesis and layout settings. 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=0, FIXED_SLAVE0_ADDR_VALUE=32, ADD_SLAVE1_ADDRESS_EN=1, FIXED_SLAVE1_ADDR_EN=1, and FIXED_SLAVE1_ADDR_VALUE=20.

Configuration Example

Figure 1 illustrates a typical application. Cortex™-M1, coupled with CoreI2C, masters communication with a SMBus Temperature Sensor slave, and an I^2C slave in FPGA #2. In FPGA #2, CoreI2C is configured in Slave-only mode with CoreABC as its control.

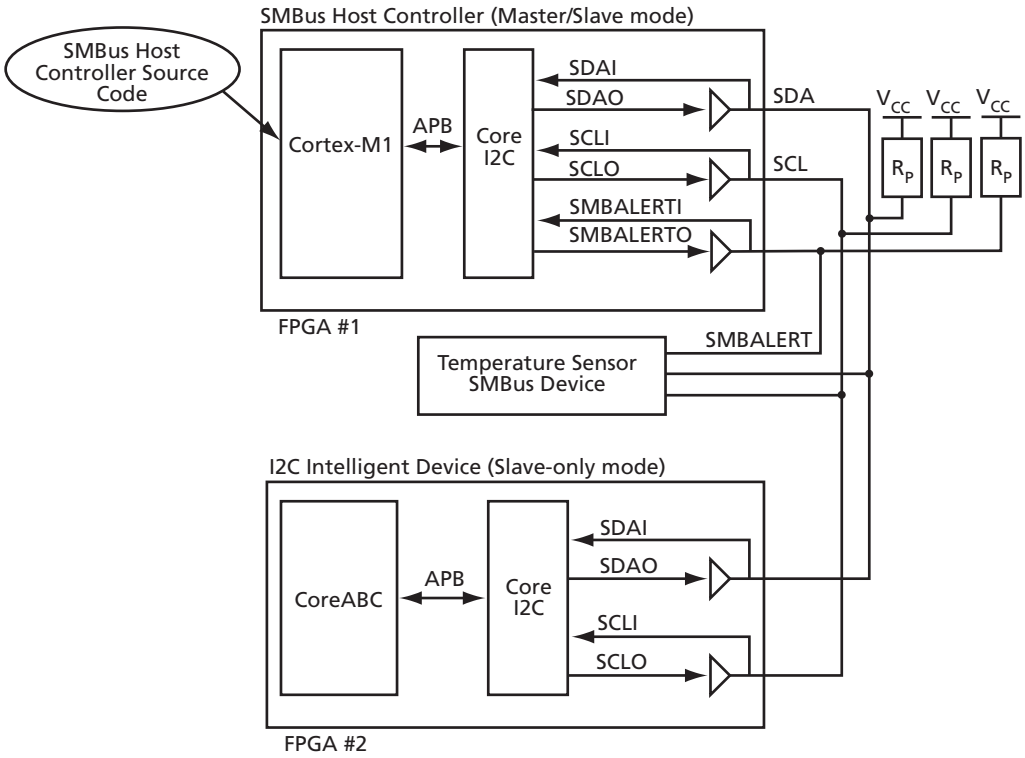

***Figure 1* • CoreI2C SMBus Application Example**

1 – Design Description

I/O Signals

The port signals for the CoreI2C macro are illustrated in Figure 1-1 and defined in Table 1-1.

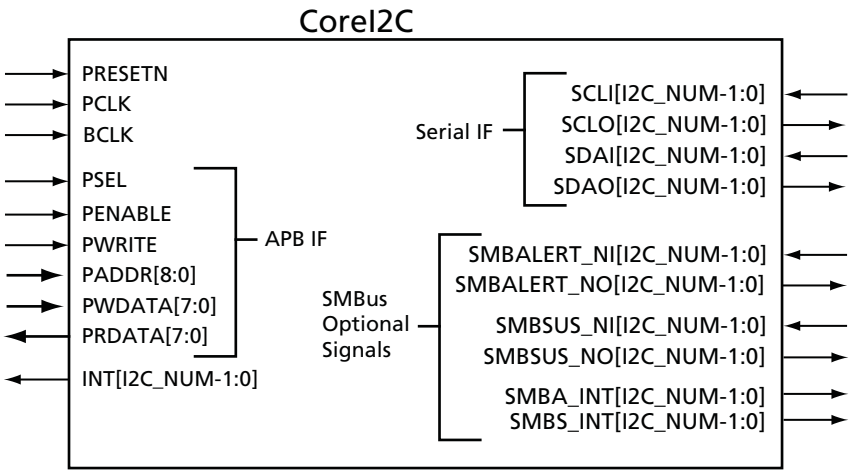

Figure 1-1 • **CoreI2C I/O Signal Diagram**

Table 1-1 • **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; address internal registers. Bits 8 to 5 function as address pointers to one of the 16 channels.
PSEL	Input	APB Slave Select; select signal for register for reads or writes
PENABLE	Input	APB Strobe. 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.

Note: All signals are active high (logic 1) unless otherwise noted.

Table 1-1 • **CoreI2C I/O Signal Descriptions (continued)**

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 Signals		
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 cr[2:0] = 111; otherwise, various divisions of PCLK are used.

Note: All signals are active high (logic 1) unless otherwise noted.

Verilog/VHDL Parameters

CoreI2C has parameters (Verilog) or generics (VHDL) for configuring the RTL code, described in Table 1-2. All parameters and generics are integer types.

Table 1-2 • **CoreI2C Parameters/Generics Descriptions**

Parameter Name	Valid Range	Description	Default
I2C_NUM	1 to 16	Number of I^2C channels	1
FREQUENCY	1 to 255	PCLK frequency value in MHz. FREQUENCY parameter is only necessary to configure optional SMBus or IPMI timeout counters.	30
OPERATING_MODE	0 to 3	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.	0
BCLK_ENABLED	0 or 1	0: BCLK input is disabled, reducing tile count. 1: BCLK input is enabled.	1
BAUD_RATE_FIXED	0 or 1	0: Baud rate value (bits cr[2:0] in the Control Register) modified by an APB-accessible register. 1: Baud rate value [bits cr[2:0] in the Control Register) is fixed to the parameter BAUD_RATE VALUE, reducing tile count.	0

Table 1-2 • **CoreI2C Parameters/Generics Descriptions (continued)**

BAUD_RATE_VALUE	0 to 7	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	0
SMB_EN	0 or 1	1: Generates the SMBus logic: SMBus register, real-time checks and timeout values. 0: SMBus logic not generated.	0
IPMI_EN	0 or 1	1: Generates 3 ms SCL Low IPMI Required Timeout Counter with error status and interrupt. 0: IPMI Timeout Counter not generated.	0
GLITCHREG_NUM	3 to 15	Number of registers in the Glitch Filter. Correct value to meet I^2C fast mode (400 kbps) and fast mode plus (1 Mbps). 50 ns spike suppression will depend on the PCLK frequency. Guideline: **PCLK Freq (MHz)** — **GlitchReg_Num for 50 ns or Less Spike Suppression** 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	3
FIXED_SLAVE0_ADDR_EN	0 or 1	0: SLAVE0 address has APB write access. 1: SLAVE0 address is hardcoded, reducing tile count.	0
FIXED_SLAVE0_ADDR_VALUE	0x00 to 0x7F	Hardcoded SLAVE0 address value.	0
ADD_SLAVE1_ADDRESS_EN	0 or 1	0: SLAVE1 address is not enabled. 1: SLAVE1 address is enabled.	0
FIXED_SLAVE1_ADDR_EN	0 or 1	0: SLAVE1 address has APB write access. 1: SLAVE1 address is hardcoded, reducing tile count.	0
FIXED_SLAVE1_ADDR_VALUE	0x00 to 0x7F	Hardcoded SLAVE0 address value	0

Serial and APB Interfaces

Serial Interface

A typical I²C/IPMI/SMBus/PMBus 8-bit data transfer cycle is shown in Figure 1-2. 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.

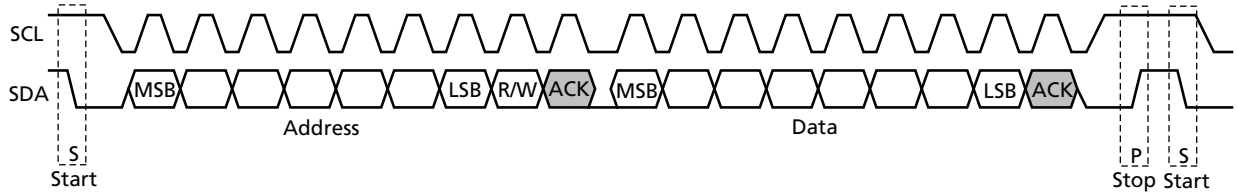

Figure 1-2 • **Serial Interface Byte Transfer**

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.

Note: 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).

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

APB Interface

Figure 1-3 and Figure 1-4 depict typical write cycle and read cycle timing relationships relative to the system clock.

Figure 1-3 • **Data Write Cycle**

Figure 1-4 • **Data Read Cycle**

Functional Block Descriptions

CoreI2C, as shown in Figure 1-5, 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.

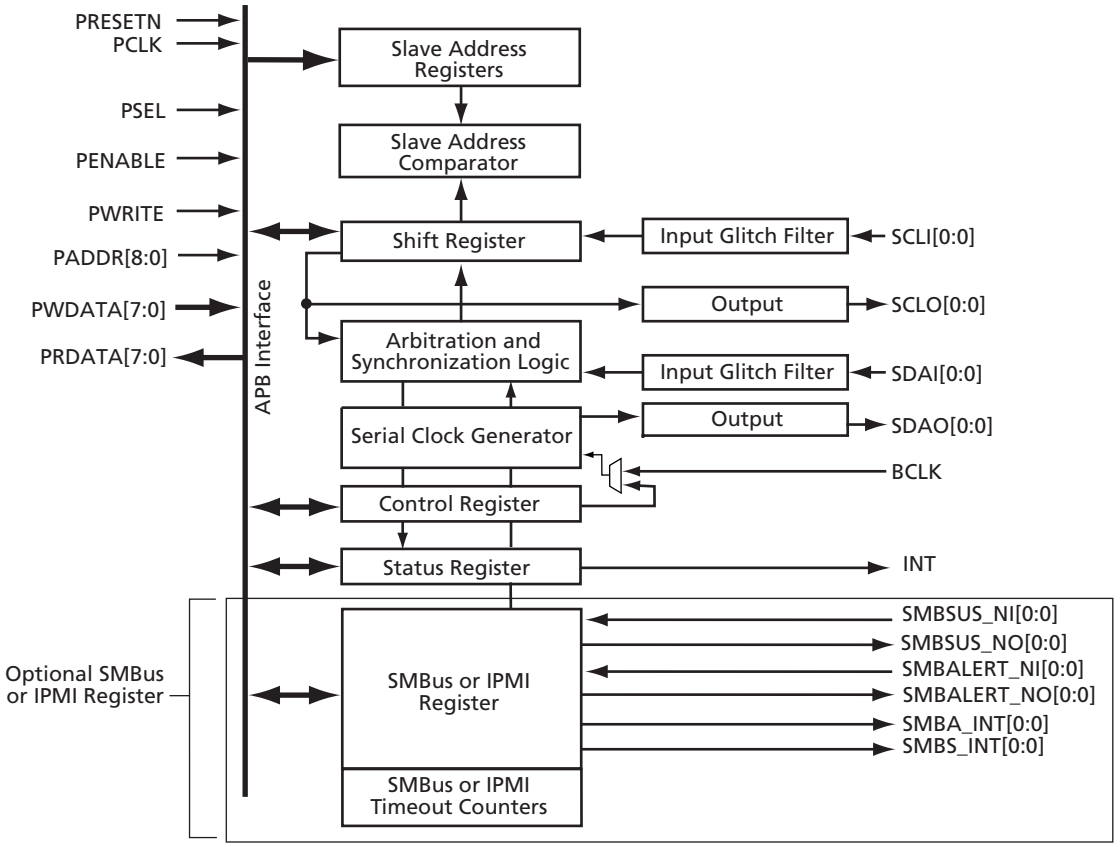

Figure 1-5 • **CoreI2C Block Diagram (single channel)**

APB Interface

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

The APB registers are defined and usage detailed in the "Register Map and Descriptions" section on page 1-19.

Input Glitch/Spike Filters

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

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.

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 Save 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 cr[2:0] 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 cr2-0 fields of the Control Register and the frequencies of PCLK and BCLK. Refer to Table 1-5 on page 1-20 for configuration.

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.

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 Details" section on page 1-17). 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.

Operation Details

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.

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 0.
- Core generates interrupt request; Status Register = 0x60 (Table 1-11 on page 1-26)
- 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 aa bit (Table 1-11 on page 1-26).
- Transfer is continued according to Table 1-11 on page 1-26.

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, sends the slave address and transfers the 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 1-9 on page 1-21).
- 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 (Table 1-9 on page 1-21).
- Transfer is continued according to Table 1-9 on page 1-21.

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 1-6.

- 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.

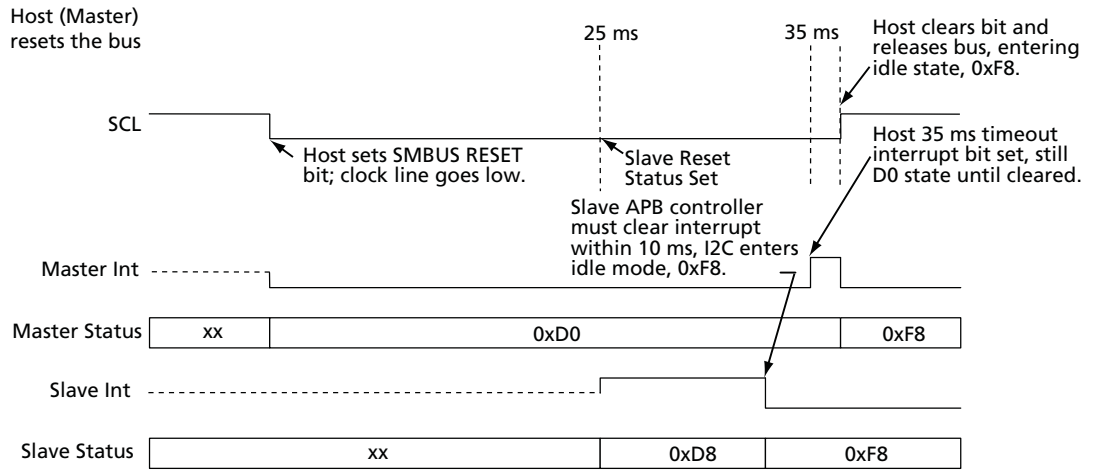

Figure 1-6 • **SMBus Bus Reset Sequence**

Register Map and Descriptions

PADDR[8:5] bits determine which I^2C channel is being addressed, as shown in Table 1-3. Table 1-4 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 1-3 • **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. Note that the Channel ID Value does not apply to the ADDR0 and ADDR1 registers shown in Table 1-4. The values in those registers are the same for all channels. **PADDR[8:5] Channel Number** 0000 0 0001 1 …… …. 1111 15

Table 1-4 • **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 I^2C channel.
0x04	STAT	R	8	0xF8	Status Register; read-only value yields the current state of the particular I^2C channel.
0x08	DATA	R/W	8	0x00	Data Register; I^2C channel read/write data to/from the serial IF.
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. 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	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. *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.*

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

Control Register

The Control Register is described in Table 1-5 and Table 1-6 on page 1-20. The CPU can read from and write to this 8-bit, directly addressable APB. 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 1-5 • **Control Register**

PADDR[4:0]	Register Name	Type	Width	Reset Value	Description
0x00	CTRL	R/W	8	0x00	Control Register; used to configure each I^2C channel.

Table 1-6 • **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 table 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.
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 software. The si bit is directly readable via the APB INTERRUPT signal.
2	aa	R/W	The Assert Acknowledge flag. 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 while the channel is in the Master receiver mode. A data byte has been received while the channel is in the Slave receiver mode. When aa = 0, a not acknowledge (NACK) will be returned when: A data byte has been received while the channel is in the Master receiver mode. A data byte has been received while the channel is in the Slave receiver mode.
1	cr1	R/W	Serial clock rate bit 1; refer to bit 0.
0	cr0	R/W	Serial clock rate bit 0; Clock Rate is defined as follows: **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

Status Register

The Status Register is read-only. The status values are listed, depending on mode of operation, in Table 1-9 through Table 1-13 on page 1-32. 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 1-7 • **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 I^2C channel.

Table 1-8 • **Status Register Bit Fields**

Bits	Name	Type	Field Description
7:0	Status	R	Read-Only Status Code. Refer to Following Tables for Code Descriptions based on Operating Mode.

Table 1-9 through Table 1-13 on page 1-32 define Status Register Code Descriptions and subsequent Action based on the four possible operating modes.

Table 1-9 • **Status Register – Master Transmitter Mode**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I^2C 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.
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-9 • **Status Register – Master Transmitter Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
0x20	SLA + W has been transmitted; NACK 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.
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 bit 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.
0x38	Arbitration lost in SLA + R/W or data bytes.	No action	0	0	0	–	The bus will be released; not-addressed slave mode will be entered.
		or no action	1	0	0	–	A START condition will be transmitted when the bus becomes free.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-9 • **Status Register – Master Transmitter Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-10 • **Status Register – Master Receiver Mode**

Status Code	Status	APB Config. Register Action	Control Register Bits				Next Action Taken by I²C 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 transmitted; NACK has been received.	No action	1	0	0	–	Repeated START condition 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.
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-10 • **Status Register – Master Receiver Mode (continued)**

Status Code	Status	APB Config. Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-11 • **Status Register – Slave Receiver Mode**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C 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.
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-11 • **Status Register – Slave Receiver Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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	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 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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-11 • **Status Register – Slave Receiver Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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 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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-11 • **Status Register – Slave Receiver Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-12 • **Status Register – Slave Transmitter Mode**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C 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 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.
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.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-12 • **Status Register – Slave Transmitter Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.

Notes:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-12 • **Status Register – Slave Transmitter Mode (continued)**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C Channel
			sta	sto	si	aa	
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:

1. *SLA = slave address*
2. *SLV = slave*
3. *REC = receiver*
4. *TRX = transmitter*
5. *SLA + W = Master sends slave address, then writes data to slave.*
6. *SLA + R = Master sends slave address, then reads data from slave.*

Table 1-13 • **Status Register – Miscellaneous States**

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by I²C 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.

Data Register

The Data Register (Table 1-14) 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 1-15 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 (i.e., Master transmit data or receive data from a Slave).

Table 1-14 • **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 1-15 • **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

SLAVE0 Address Register

The SLAVE0 Address Register (ADDR0, Table 1-16 and Table 1-17) is a read/write directly accessible register.

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

Table 1-16 • **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 1-17 • **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.

Optional SMBus/IPMI Register

The SMBus Register contains specific SMBus related functionality and is Read- or Write-able as defined in Table 1-19 on page 1-35. 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 enables/disables 3 ms IPMI SCL Low timeout.

Table 1-18 • **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 1-19 • **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.

Table 1-19 • **SMBus/IPMI Register Bit Fields**

Bits	Name	Type	SMBus Context (SMB_EN = 1)	IPMI Context (IPMI_EN = 1)
2	SMB_IPMI_EN	R/W	0: SMBus timeouts and status logic disabled, i.e., standard I^2C bus operation; 1: SMBus timeouts and status logic enabled.	0: IPMI timeout and status logic disabled, i.e., standard I^2C bus operation; 1: IPMI timeout and 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.

Optional SLAVE1 Address Register

The SLAVE1 Address Register (ADDR1, Table 1-20 and Table 1-21) is an 8-bit read/write directly accessible register with two separate contexts depending on parameter configuration.

Note: If the parameter FIXED_SLAVE1_ADDR_EN is enabled, the register is read-only.

Table 1-20 • **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 1-21 • **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.

2 – Tool Flows

CoreI2C is licensed in two ways. Depending on your license tool flow, functionality may be limited.

Obfuscated

Complete RTL code is provided for the core, allowing the core to be instantiated with SmartDesign. Simulation, Synthesis, and Layout can be performed within Libero® Integrated Design Environment (IDE). The RTL code for the core is obfuscated[1] and some of the testbench source files are not provided; they are precompiled into the compiled simulation library instead.

RTL

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

SmartDesign

CoreI2C (Figure 2-1) is preinstalled in the SmartDesign IP Deployment design environment.

The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 2-2 on page 2-40. Callouts to associated parameters are shown in red.

For information on using SmartDesign to instantiate and generate cores, refer to the *Using DirectCore in Libero® IDE User's Guide*.

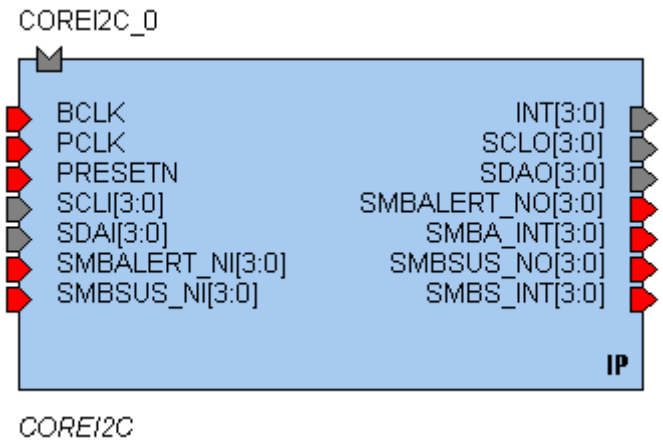

Figure 2-1 • **CoreI2C Full I/O View**

1. *Obfuscated means the RTL source files have had formatting and comments removed, and all instance and net names have been replaced with random character sequences.*

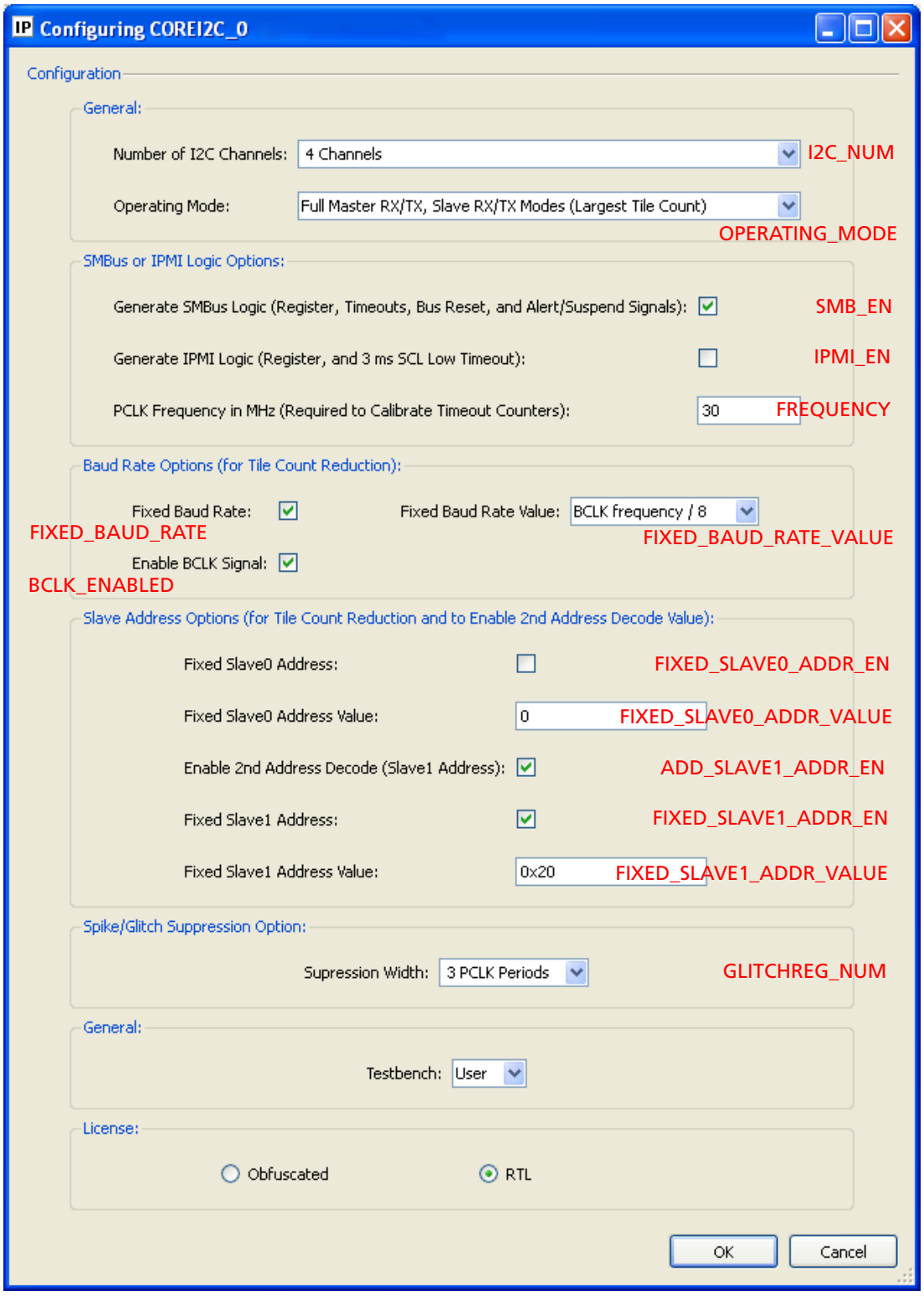

Figure 2-2 • **CoreI2C SmartDesign Configuration with Callouts to Associated Parameters**

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 IDE project, it will install the user testbench files.

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

User Testbench

As shown in Figure 2-3, two instantiations of the CoreI2C macro are connected to an I^2C bus. The second coreI2C instance is configured in multi-channel mode and uses the 13th channel. The top-level test bench (*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 I^2C 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 my be altered directly in the tb_user_corei2c.v or tb_user_corei2c.vhd file..

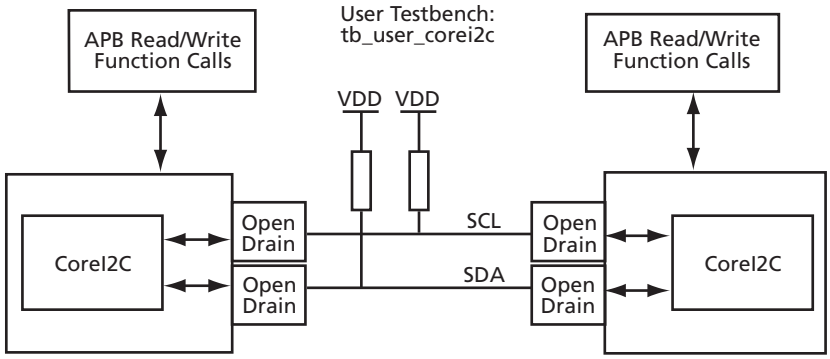

Figure 2-3 • **CoreI2C User Testbench**

Synthesis in Libero IDE:

Having set the design route appropriately, click the **Synthesis** icon in Libero IDE. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the **Run** icon.

Place-and-Route in Libero IDE

Having set the design route appropriately and run Synthesis, click the **Layout** icon in the Libero IDE to invoke Designer. CoreI2C requires no special place-and-route settings.

3 – Example Application and Hints

This chapter provides various hints to ease the process of implementation and integration of CoreI2C into your own design.

Software Driver

Drivers for CoreI2C are available via the Firmware Catalog tool provided with Libero IDE. For more information about the Firmware Catalog, refer to the Actel web site: www.actel.com/products/software/firmwarecat/default.aspx.

Usage with Cortex-M1

CoreI2C may be used with Cortex-M1, Actel's soft IP version of the popular ARM7TDMI-S™ microprocessor that has been optimized for the M1 Fusion flash-based FPGA devices. To create a design using Cortex-M1, internal flash memory, and CoreI2C, you should use the SmartDesign Intellectual Property Deployment Platform (IDP) software. Refer to the SmartDesign User's Guide on how to create your Cortex-M1–based design.

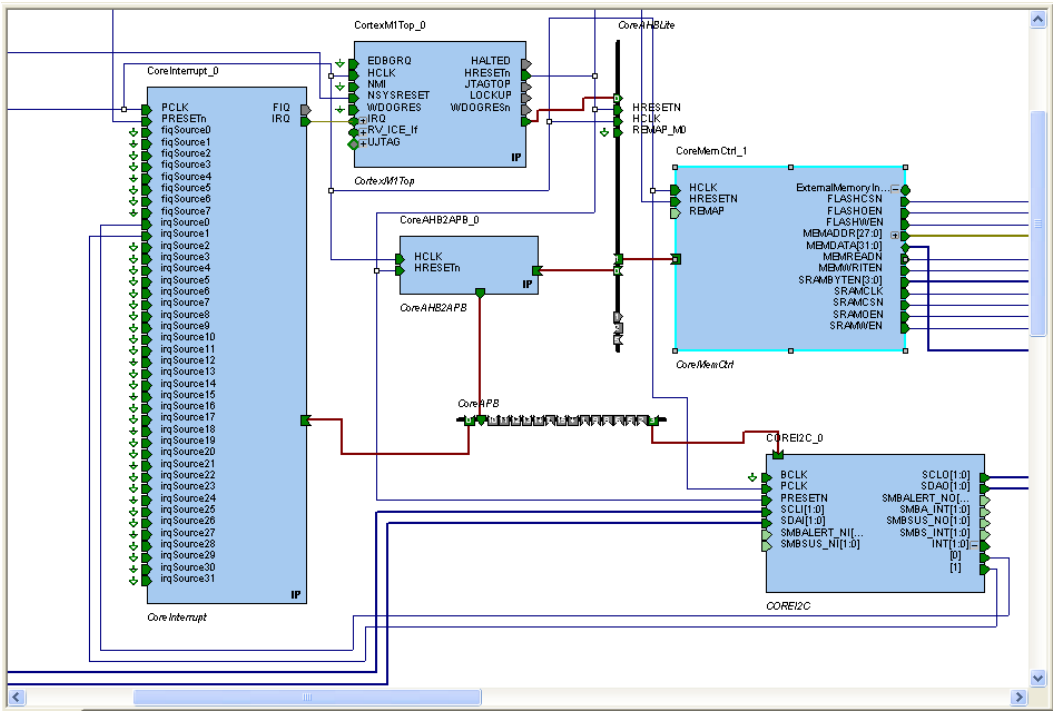

Figure 3-1 • **Example System Using Cortex-M1 with CoreI2C in a Two Channel Configuration**

Hints on I/O Pad Requirements

The I^2C, SMBus, and PMBus Specifications set minimum and maximum I/O buffer and pad requirements based on type of implementation.

CoreI2C can be used for all these potential applications, as long as the I/O buffer/pads are configured to comply with the specific I^2C or SMBus requirements. Typically, for 100 Kbps operation, standard default I/O buffer values are okay. For 400 kbps operation however, tighter buffer constraints may be necessary to fully conform to a given I^2C/SMBus/PMBus requirement. Refer to electrical characteristics for each specification type to correctly program the I/O pads:

I^2C Specifications at http://www.i2c-bus.org/

SMBus Specifications at http://smbus.org/specs/

PMBus Specifications at http://pmbus.org/specs.html

Hints on Configuring WIRED-AND Bidirectional Buffers in RTL

For an example on how to connect the WIRED-AND bidirectional SCL and SDA outputs in a design, refer to the Verilog tb_user_corei2c.v and/or the VHDL tb_user_corei2c.vhd RTL user testbench.

Hints on Meeting SMBus/PMBus Timing Requirements

Refer to the "Serial Interface" section on page 1-14 for specific PCLK requirements necessary to adhere to SMBus and PMBus specifications.

4 – List of Document Changes

The following table lists critical changes that were made in the current version of the document.

Previous Version	Changes in Current Document Version (50200090-5)	Page
50200090-4 (November 2009)	The "Core Version" was updated to v6.0 and the "Core Overview" section was updated to include information about the multiple I^2C channel configuration option.	5
	The utilization tables were updated.	6
	Signals and parameters in the "Design Description" section were updated in text and figures for multiple I^2C channel functionality.	11
	Figure 1-4 • Data Read Cycle was updated.	14
	The "Optional SMBus/IPMI Logic" section is new.	16
	Table 1-4 • CoreI2C Internal Register Address Map was updated.	19
	Table 1-6 • Control Register Bit Fields was updated for R/W properties.	20
	Table 1-16 • Slave0 Address Register was updated for R/W properties.	34
	Table 1-20 • Slave1 Address Register was updated for R/W properties.	37
	The "Obfuscated" section was updated for SmartDesign.	39
	Figure 2-1 • CoreI2C Full I/O View is new and Figure 2-2 • CoreI2C SmartDesign Configuration with Callouts to Associated Parameters was updated.	39, 40
	The "Simulation Flows" section was updated for SmartDesign.	41
	The "User Testbench" section was updated for the multiple I^2C channel configuration.	41
	Removed Ordering Information Section	N/A
	Figure 3-1 • Example System Using Cortex-M1 with CoreI2C in a Two Channel Configuration was updated	43
50200090-3	The "Core Version" was updated to v5.0. Text, figures, signal names, parameters/generics, and register maps and descriptions have been revised accordingly.	N/A
	CoreMP7 references were removed and replaced with Cortex-M1.	N/A
	"Ordering Codes" have been included.	43
50200090-2	The CoreI2C Handbook and CoreSMBus Handbook have been condensed and combined into the current document.	N/A
50200090-1	The "Supported Device Families" section was added.	5
	The "APB Interface" section was updated to include the Cortex-M1 processor.	14
	The "Use with Core8051s" section was updated to change Core8051 to Core8051s.	34
50200090-0	The data transfer rate and the SLAVE_EN_ONLY parameter for Master receiver mode were updated in the "Key Features" section section.	5
	Figure 1-5 • CoreI2C Block Diagram (single channel) was updated.	15
	The first two paragraphs of the "I2C Serial Interface" section were updated.	16

Previous Version	Changes in Current Document Version (50200090-5)	Page
	Figure 1-1 • CoreI2C I/O Signal Diagram was updated to remove the BCLK signal.	11
	Figure 1-3 • Data Write Cycle was updated to change the signal PRDATA to PWDATA.	14
	The second table note, which stated the clock rate frequency of 100 Kbps should not be exceeded, was removed from Table 1-5 • Control Register.	20

A – Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.

Index

Actel is the leader in low-power FPGAs and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at www.actel.com.

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court,Meadows Business Park
Station Approach, Blackwater
Camberley Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Buillding 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
http://jp.actel.com

Actel Hong Kong
Room 2107, China Resources Building
26 Harbour Road
Wanchai, Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn

50200090-5/11.09