CoreSMBus

Handbook

Actel®

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 50200082-1

Release: July 2007

Trademarks

Table of Contents

Introduction

Core Overview

Key Features

- Supports SMBus v2.0
- Master/Slave transmit/receive modes
- Advanced Peripheral Bus (APB) register interface
- Data transfers up to at least 400 kbps nominally; faster rates can be achieved, depending on external load circuitry.
- Multi-master collision detection and arbitration
- Own address and general call address detection
- 7-bit addressing format
- Fixed data width of 8 bits
- Data transfer in multiples of bytes
- SMBus timeout and idle condition real-time counters
- Optional SMBus signals, SMBSUS_N and SMBALERT_N, controllable via APB IF
- Accompanying Master/Host SMBus microcontroller C source code included in the CoreSMBus release database

The system management bus uses two wires to transfer information among devices connected to the bus:

- SMBCLK (serial clock line)
- SMBDAT (serial data line)

Byte transfers across the serial line and APB interface are handled autonomously by CoreSMBus. The core also keeps track of serial transfers with a status register (serial_sta). The status register monitors both the serial bus and the state of the core. Figure 1 on page 6 illustrates the CoreSMBus functional block diagram.

Each device connected to the bus is software-addressable by a unique address. CoreSMBus supports multi-master collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer.

The input and output filtering rejects spikes on the bus data line to preserve data integrity. CoreSMBus performs 8-bit oriented, bidirectional data transfers, and may operate in the following four modes:

- Master transmitter mode (when SLAVE_ONLY_EN parameter = 0)
- Master receiver mode (when SLAVE_ONLY_EN parameter = 1)
- Slave receiver mode
- Slave transmitter mode

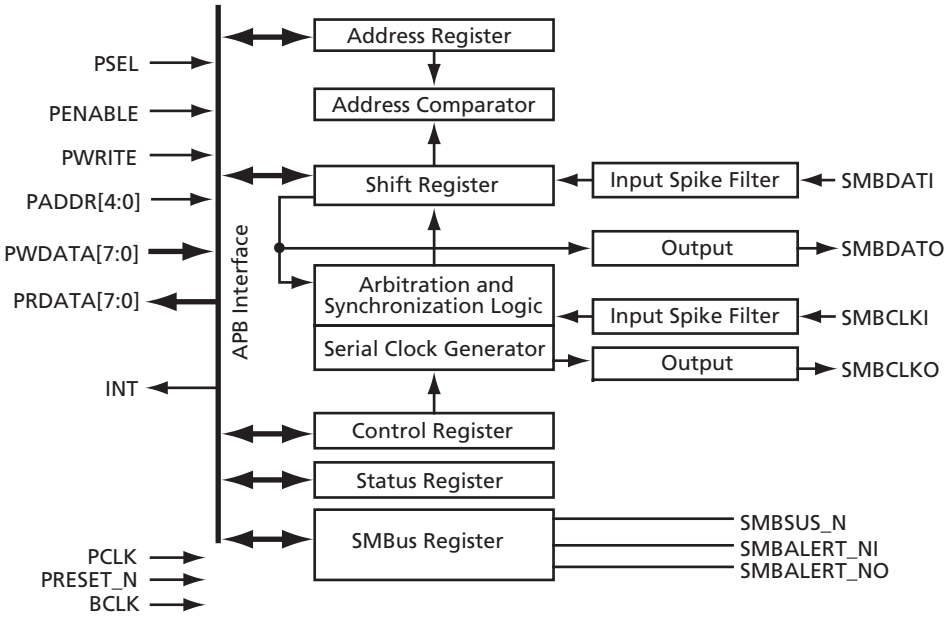

Figure 1 · CoreSMBus Block Diagram

Figure 2 illustrates a typical application of CoreSMBus. In this example, FPGA #1 is the SMBus host controller, with CoreMP7 controlling CoreSMBus. CoreMP7, coupled with CoreSMBus in FPGA #1, controls communication between FPGA #1 and an SMBus temperature sensor, as well as FPGA #2, where CoreSMBus is configured in Slave-only mode with CoreABC.

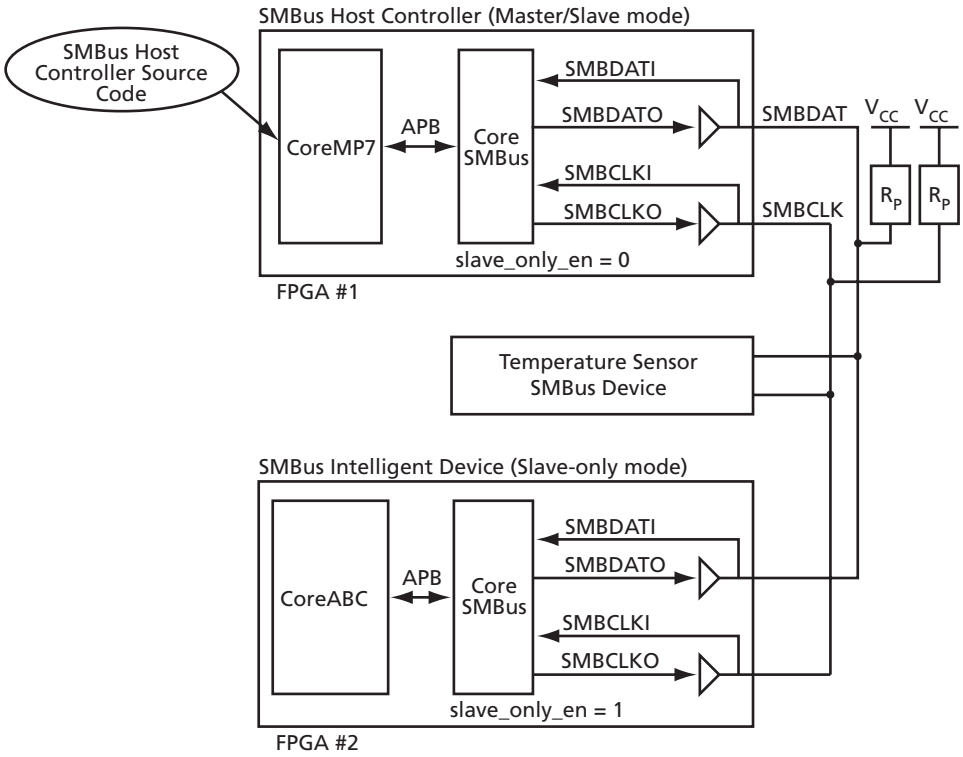

Figure 2 · CoreSMBus Application Example

Utilization and Performance

CoreSMBus can be implemented in any Actel device. A summary of CoreSMBus utilization and performance for various devices is listed in Table 1 through Table 4 on page 9.

Table 1 · CoreSMBus Device Utilization and Performance (Slave-only configuration)

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	134	587	721	AFS600	5%	83 MHz
ProASIC®3/E	134	599	733	M7A3P250	12%	95 MHz
ProASIC^{PLUS}®	128	727	855	APA075	28%	66 MHz
Axcelerator®	146	438	584	AX250	14%	121 MHz
RTAX-S	146	438	584	RTAX250S	14%	86 MHz

Note: Data in this table were achieved using typical synthesis and layout settings. Top-level parameters/generics were set as follows: SMB_EN = 1, SLAVE_ONLY_EN = 1, FREQUENCY = 50.

Table 2 · CoreSMBus Device Utilization and Performance (Master/Slave configuration)

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	182	891	1073	AFS600	8%	90 MHz
ProASIC3/E	182	896	1078	M7A3P250	18%	92 MHz
ProASIC^{PLUS}	180	1043	1223	APA075	40%	64 MHz
Axcelerator	189	571	760	AX250	18%	120 MHz
RTAX-S	189	571	760	RTAX250S	18%	89 MHz

Note: Data in this table were achieved using typical synthesis and layout settings. Top-level parameters/generics were set as follows: SMB_EN = 1, SLAVE_ONLY_EN = 0, FREQUENCY = 50.

Table 3 · CoreSMBus Device Utilization and Performance (I2C Slave-only configuration)

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	84	388	472	AFS600	3%	128 MHz
ProASIC3/E	91	430	521	M7A3P250	12%	127 MHz
ProASIC^{PLUS}	84	466	550	APA075	18%	72 MHz
Axcelerator	97	274	371	AX250	9%	141MHz
RTAX-S	97	274	371	RTAX250S	9%	98 MHz

Note: Data in this table were achieved using typical synthesis and layout settings. Top-level parameters/generics were set as follows: SMB_EN = 0, SLAVE_ONLY_EN = 1.

Table 4 · CoreSMBus Device Utilization and Performance (I2C Master/Slave configuration)

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	134	679	813	AFS600	6%	119 MHz
ProASIC3/E	126	628	754	M7A3P250	12%	126 MHz
ProASIC^{PLUS}	121	676	797	APA075	26%	72 MHz
Axcelerator	134	391	525	AX250	12%	133 MHz
RTAX-S	134	391	525	RTAX250S	12%	94 MHz

Note: Data in this table were achieved using typical synthesis and layout settings. Top-level parameters/generics were set as follows: SMB_EN = 0, SLAVE_ONLY_EN = 0.

Supported Interfaces

CoreSMBus is available with the following interfaces:

- APB interface for register access
- SMBus serial interface

These interfaces are further described in "Interface Definitions" on page 16.

1 Functional Block Descriptions

CoreSMBus, as shown in Figure 1 on page 6, consists of APB interface registers, serial input spike filters, arbitration and synchronization logic, and a serial clock generation block. The following sections describe each design block briefly.

APB Interface

CoreSMBus supports the APB interface, compatible with the Actel Core8051 and CoreMP7 processor cores, as well as the CoreABC generic state machine control core (for simple FSM applications).

"Register Map" on page 19 defines and details the use of the APB registers.

Input Spike Filters

Input signals are synchronized with the internal clock, PCLK. Spikes shorter than three clock periods 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. CoreSMBus 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 the timeout requirements set forth in the SMBus Specification version 2.0 (50 µs wait before attempting to gain control of an idle bus, 25 ms reset for Slave mode devices, and 35 ms bus reset control for Host/Master mode devices).

Serial Clock Generator

This programmable clock pulse generator provides the serial bus clock pulses when CoreSMBus is in Master mode. The clock generator is switched off when CoreSMBus 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 can be used to set the serial clock frequency when the cr[2:0] bits in the Control Register are set to '111'; otherwise, PCLK is used to determine the serial clock frequency. The actual non-stretched serial bus clock frequency can be calculated based on the setting in the cr[2:0] fields of the Control Register and the frequencies of PCLK and BCLK. Refer to Table 4-2 on page 20 for configuration.

Address Comparator

The comparator checks the received 7-bit slave address against its own slave address. It also compares the first received 8-bit byte with the general call address (00h). If a match is found, the Status Register is updated and an interrupt is requested.

2 Tool Flows

Available Versions

CoreSMBus is available in three versions. The Evaluation version does not require a license. Depending on your license tool flow, functionality may be limited.

Evaluation

Precompiled simulation libraries are provided, allowing the core to be instantiated in CoreConsole and simulated within the Actel Libero® Integrated Design Environment (IDE), as described in "Simulation Flows" on page 14. The design may not be synthesized, as source code is not provided.

Obfuscated

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

RTL

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

CoreConsole

CoreSMBus is preinstalled in the CoreConsole Intellectual Property Deployment Platform (IDP). To use the core,[2] drag it from the IP Core list into the main window. The CoreConsole project can be exported to Actel Libero IDE at this point, providing access to CoreSMBus. Other IP blocks can interconnect, allowing the complete system to be exported from CoreConsole to Actel Libero IDE.

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.

2. A CoreSMBus license is required to generate the design for export to Actel Libero IDE for simulation and synthesis.

The core can be configured using the configuration GUI within CoreConsole, as shown in Figure 2-1. The parameters are fully described in the "Top-Level Generics/Parameters" section on page 16.

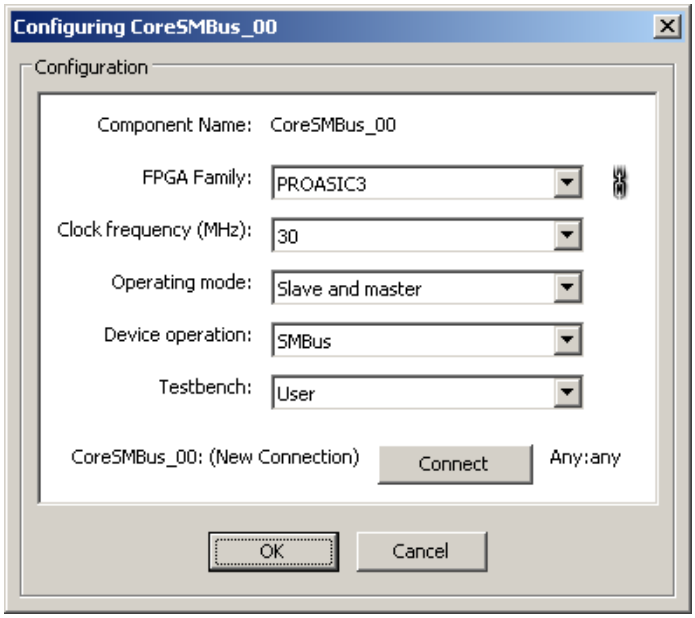

Figure 2-1. CoreSMBus Configuration within CoreConsole

Simulation Flows

To run simulations, select the required testbench flow within CoreConsole and run **Save & Generate** from the Generate pane. Select the required testbench through the Core Testbench Configuration GUI.

Simulating with the Simple User Application Testbench

To simulate, **simply select the instance of CoreSMBus as the top level in Libero IDE**, and select **Simulate** in the Design Flow window. Libero IDE will automatically generate a *run.do* file, invoke Model*Sim*,® and execute *run.do*.

Synthesis in Actel Libero IDE

Having set the design route appropriately, click the **Synthesis** icon in Actel 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, click the **Run** icon.

Place-and-Route in Actel Libero IDE

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

3

Top-Level I/O and Interface Definitions

Top-Level I/O Descriptions

I/O Port Descriptions

The port signals for the CoreSMBus macro are illustrated in Figure 3-1 and defined in Table 3-1. CoreSMBus has 34 I/O signals.

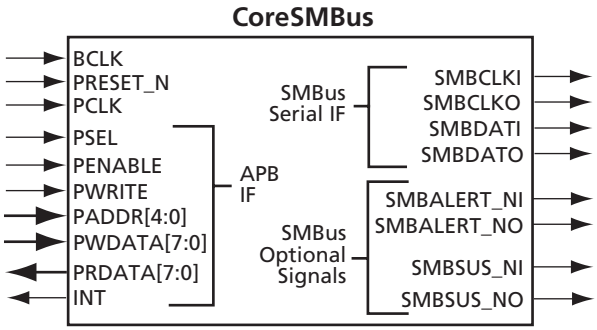

Figure 3-1 · CoreSMBus I/O Signal Diagram

Table 3-1 · CoreSMBus I/O Signal Descriptions

Name	Type	Description
APB Interface		
PCLK	Input	APB System Clock – Reference clock for all internal logic
PRESET_N	Input	APB active low asynchronous reset
PADDR[4:0]	Input	APB address bus – This port is used to address internal CoreSMBus registers.
PSEL	Input	APB Slave Select – This signal selects CoreSMBus 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 will occur when an APB transfer to CoreSMBus takes place; if LOW, a read from CoreSMBus will take place.
PWDATA[7:0]	Input	APB write data
PRDATA[7:0]	Output	APB read data
INTERRUPT	Output	Microprocessor interrupt output
SMBus Serial Interface		
SMBCLKI	Input	Wired-AND SMBus serial clock input
SMBCLKO	Output	Wired-AND SMBus serial clock output
SMBDATI	Input	Wired-AND SMBus serial data input
SMBDATO	Output	Wired-AND SMBus serial data output
SMBus Optional Signals		
SMBALERT_NI	Input	Input wired-AND interrupt signal; used in Master/Host mode to monitor whether slave/devices want to force communication with the Host.

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

Table 3-1 · CoreSMBus I/O Signal Descriptions (continued)

SMBALERT_NO	Output	Output wired-AND interrupt signal; used in Slave/Device mode if the core wants to force communication with a host.
SMBSUS_NI	Input	Input Suspend Mode signal; used if core is Slave/Device. NOTE: *Not a wired-AND signal.*
SMBSUS_NO	Output	Output Suspend Mode signal; used if core is Master/Host. *NOTE: Not a wired-AND signal.*
Other Signals		
BCLK	Input	Pulse for SMBCLK speed control. Used only if the configuration bits cr[2:0] = '111'; otherwise, various divisions of PCLK are used. If unused, connect to logic 0.

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

Top-Level Generics/Parameters

CoreSMBus has parameters (Verilog) or generics (VHDL), described in Table 3-2, for configuring the RTL code. All parameters and generics are integer types.

Table 3-2 · CoreSMBus Parameter/Generic Descriptions

Name	Valid Range	Description
SMB_EN	0 or 1	When set to '1', CoreSMBus utilizes the real-time checks and timeout values specified in the SMBus Specification version 2.0. When set to'0', real-time checks and timeouts are disabled. This could be useful for debug purposes or for special case design topologies. Essentially the core operates in I2C mode.
SLAVE_ONLY_EN	0 or 1	Set to '1' if the core is to be used in Slave-only mode. Set to '0' if the core is to be used in Master-only mode or dual Master/Slave mode.
FREQUENCY	Up to 100	PCLK frequency value in MHz; necessary to configure SMBus timeout counters.

Interface Definitions

CoreSMBus includes the following interfaces:

- SMBus serial interface
- APB register interface

SMBus Serial Interface

The SMBus interface is a derivative of the I^2C serial bus. A typical timing cycle is shown in Figure 3-2 on page 17, with some of the important timing requirements from the SMBus 2.0 Specification included. AC timing specifications are shown in Figure 3-2 on page 17 and listed for reference in Table 3-3 on page 17, with notes that explain how CoreSMBus interprets some of the requirements. Note that the SMBCLK frequency must be between 10 and 100 kHz; it is up to the designer to choose a suitable PCLK frequency. The designer should ensure that the software and CoreSMBus adhere to the SMBus timing specifications. For example, it is advised to run SMBCLK at or near the maximum frequency of 100 kHz ($F_{SMB-max}$) to ensure that other potential clock-stretching devices on the bus will not slow the clock frequency to below 10 kHz ($F_{SMB-min}$).

For more detailed timing requirement information, refer to the SMBus 2.0 Specification directly.

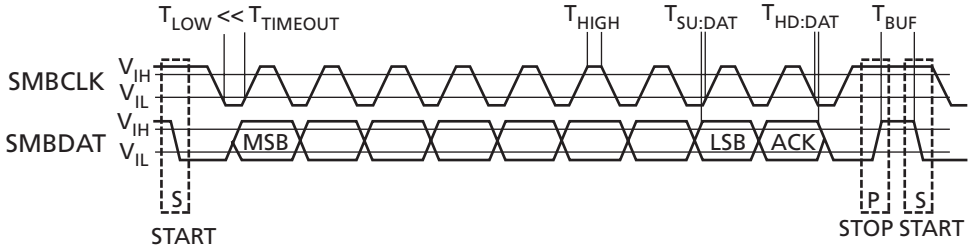

Figure 3-2 · SMBus Timing Diagram

Table 3-3 · SMBus Specification 2.0 AC Specifications

Symbol	Parameter	Limits		Units	Comments
		Min.	Max.		
F_{SMB}	SMBus operating frequency	10	100	kHz	
T_{BUF}	Bus free time between STOP and START conditions	4.7	–	µs	This limit must be ensured by software waits between STOP and START instructions.
$T_{HD:STA}$	Hold time after repeated START condition. After this period, the first clock is generated.	4.0	–	µs	
$T_{SU:STA}$	Repeated START condition setup time	4.7	–	µs	
$T_{SU:STO}$	STOP condition setup time	4.0	–	µs	
$T_{HD:DAT}$	Data hold time	300	–	ns	
$T_{SU:DAT}$	Data setup time	250	–	ns	
$T_{TIMEOUT}$	Detect clock LOW timeout	25	35	ms	Slave must start a reset sequence after a clock low time of 25 ms, and finish the reset sequence before 35 ms. If a Master wants to reset the bus, it must hold the bus low for at least 35 ms.
T_{LOW}	Clock LOW period	4.7	–	µs	
T_{HIGH}	Clock HIGH period	4.0	50	µs	50 µs idle time checked by a new Master to make sure bus is not being used
$T_{LOW:SEXT}$	Cumulative clock LOW extend period (Slave device)	–	25	ms	This limit is accounted for with the $T_{TIMEOUT}$ parameter as specified above.
$T_{HIGH:MEXT}$	Cumulative clock LOW extend time (Master device)	–	10	ms	
T_F	Clock/data fall time	–	300	ns	
T_R	Clock/data rise time	–	1,000	ns	
T_{POR}	Time in which a device must be operational after power-on reset	–	500	ms	

APB Interface

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

Figure 3-3 · Data Write Cycle

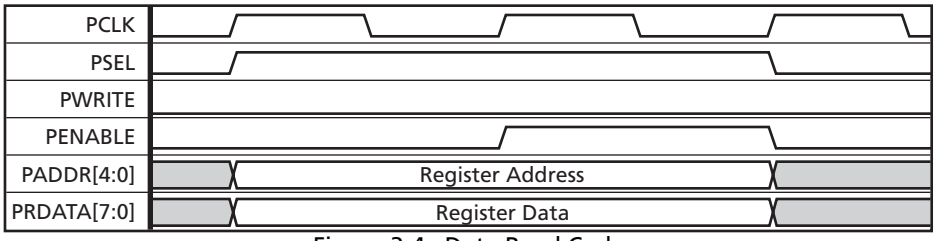

Figure 3-4 · Data Read Cycle

4

Register Map

APB Register Map

The internal register address map and reset values of each APB-accessible register for CoreSMBus are shown in Table 4-1. Values shown are in hexadecimal format. Type designations: "R" for read-only and "R/W" for read/write.

Table 4-1 · CoreSMBus Internal Register Address Map

PADDR[6:0]	Type	Reset Value	Brief Description
0x00	R/W	0x00	**Control Register**; used to configure the core.
0x04	R	0xF8	**Status Register**; read-only value yields the current state of the core.
0x08	R/W	0x00	**Data Register**; read/write data to/from the serial interface.
0x0C	R/W	0x00	**Address Register**; contains the programmable address of the core.
0x10	R/W	0x40	**SMBus Register**; configuration register for SMBus timeout reset condition and for the optional SMBus signals SMBALERT_N and SMBSUS_N.

Note: Only the lower seven bits of PADDR are used.

Register Descriptions

The following sections and tables detail the APB-accessible registers within CoreSMBus.

Control Register

The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits are affected by CoreSMBus: 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 SMBUS bus.

Table 4-2 describes the function of each control register.

Table 4-2 · Control Register

Bits	Name	Function
7	cr2	Clock rate bit 2; refer to bit 0.
6	ens1	Enable bit. When ens1 = 0, the sda and scl outputs are in a high-impedance state, and the sda and scl input signals are ignored. When ens1 = 1, the core is enabled.
5	sta	The START Flag. When sta = 1, the core checks the status of the serial bus and generates a START condition if the bus is free.
4	sto	The STOP Flag. When sto = 1 and the core is in Master mode, a STOP condition is transmitted to the serial bus.
3	si	The Serial Interrupt Flag. The si flag is set by the core whenever there is a serviceable change in the Status Register. After the register has been updated, the si bit must be cleared by software.*
2	aa	The Assert Acknowledge Flag. When aa = 1, an acknowledge will be returned in these cases: • 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 core is in Master receiver mode. • A data byte has been received while the core is in Slave receiver mode. When aa = 0, a not-acknowledge will be returned in these cases: • A data byte has been received while the core is in Master receiver mode. • A data byte has been received while CoreSMBus is in Slave receiver mode.
1	cr1	Serial clock rate bit 1; refer to bit 0.
0	cr0	Serial clock rate bit 0; clock rate is defined in Table 4-3.

*Note: *The si bit is directly readable via the APB INTERRUPT signal.*

Table 4-3 · Clock Rate Defined

cr2	cr1	c0	SMBCLK 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 4-2 through Table 4-6 on page 25. Whenever there is a change of state, INTERRUPT is requested. After updating any registers, the APB interface control must clear INTERRUPT by clearing the si bit of the Control Register.

Operation Details

CoreSMBus logic can operate in the following four modes:

1. Master Transmitter Mode
   Serial data output through SMBDAT while SMBCLK puts out the serial clock.
2. Master Receiver Mode
   Serial data is received via SMBDAT while SMBCLK puts out the serial clock.
3. Slave Receiver Mode
   Serial data and the serial clock are received through SMBDAT and SMBCLK.
4. Slave Transmitter Mode
   Serial data is transmitted via SMBDAT while the serial clock is received through SMBCLK.

Below are examples of using the Status Register to operate CoreSMBus in Slave and Master mode.

Slave Mode Example

After setting the ens1 bit in the Control Register, the core is in non-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 an interrupt is requested. Then the core may 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 = 60h (refer to Table 4-6 on page 25).
- Microcontroller prepares to receive data and then clears si bit.
- Core receives next data byte and then generates interrupt request. The Status Register contains 80h or 88h, depending on aa bit (Refer to Table 4-6 on page 25).
- Transfer is continued according to Table 4-6 on page 25.

Master Mode Example

When the microcontroller wishes to become the bus Master, the core waits until the serial bus is free. Then the core generates a START condition, sends the Slave address, and transfers the direction bit. The core may 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 = 08h (refer to Table 4-4 on page 22).
- Microcontroller writes the Data Register (7-bit Slave address and '0') and then clears the si bit.
- Core sends Data Register contents and then generates interrupt request. The Status Register contains 18h or 20h, depending on received ACK bit (refer to Table 4-4 on page 22).
- Transfer is continued according to Table 4-4 on page 22.

Status Register Codes per Mode of Operation

Table 4-4 · Status Register—Master Transmitter Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
08h	A START condition has been transmitted.	Load SLA + W	X	0	0	X	SLA + W will be transmitted; ACK will be received.
10h	A repeated START condition has been transmitted.	Load SLA + W	X	0	0	X	SLA + W will be transmitted; ACK will be received.
		or load SLA + R	X	0	0	X	SLA + W will be transmitted; core will be switched to MST/REC mode.
18h	SLA + W has been transmitted; ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
20h	SLA + W has been transmitted; not-ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
28h	Data byte in Data Register has been transmitted; ACK has been received.	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
30h	Data byte in Data Register has been transmitted; not-ACK has been received.	Data byte	0	0	0	X	Data byte will be transmitted; ACK will be received.
		or no action	1	0	0	X	Repeated START will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	X	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 4-4 · Status Register—Master Transmitter Mode (continued)

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
38h	Arbitration lost in SLA + R/W or data bytes.	No action	0	0	0	X	SMBUS bus will be released; non-addressed Slave mode will be entered.
		or no action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
D0h	smbus_mst_reset has been activated.	No action	X	X	X	X	Wait 35 ms for interrupt to be set. Clear interrupt and proceed to F8h state.

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 4-5 · Status Register—Master Receiver Mode

Status Code	Status	APB Config Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
08h	A START condition has been transmitted.	Load SLA + R	X	0	0	X	SLA + R will be transmitted; ACK will be received.
10h	A repeated START condition has been transmitted.	Load SLA + R	X	0	0	X	SLA + R will be transmitted; ACK will be received.
		or load SLA + W	X	0	0	X	SLA + W will be transmitted; CoreSMBus will be switched to MST/TRX mode.
38h	Arbitration lost in not-ACK bit.	No action	0	0	0	X	SMBUS bus will be released; CoreSMBus will enter Slave mode.
		or no action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
40h	SLA + R has been transmitted; ACK has been received.	No action	0	0	0	0	Data byte will be received; not-ACK will be returned.
		or no action	0	0	0	1	Data byte will be received; ACK will be returned.
48h	SLA + R has been transmitted; not-ACK has been received.	No action	1	0	0	X	Repeated START condition will be transmitted.
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
50h	Data byte has been received; ACK has been returned.	Read data byte	0	0	0	0	Data byte will be received; not-ACK will be returned.
		or read data byte	0	0	0	1	Data byte will be received; ACK will be returned.
58h	Data byte has been received; not-ACK has been returned.	Read data byte	1	0	0	X	Repeated START condition will be transmitted.
		or read data byte	0	1	0	X	STOP condition will be transmitted; sto flag will be reset.
		or read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset.
D0h	smbus_mst_reset has been activated.	No Action	X	X	0	X	Wait 35 ms for interrupt to be set. Clear interrupt and proceed to F8h state.

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 4-6 · Status Register—Slave Receiver Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
60h	Own SLA + W has been received; ACK has been returned.	No action	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
68h	Arbitration lost in SLA + R/W as Master; own SLA + W has been received, ACK returned.	No action	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
70h	General call address (00h) has been received; ACK has been returned.	No action	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
78h	Arbitration lost in SLA + R/W as Master; general call address has been received, ACK returned.	No action	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or no action	X	0	0	1	Data byte will be received and ACK will be returned.
80h	Previously addressed with own SLV address; DATA has been received, ACK returned.	Read data byte	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
88h	Previously addressed with own SLA; DATA byte has been received, not-ACK returned.	Read data byte	0	0	0	0	Switched to non-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	0	0	0	1	Switched to non-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	0	Switched to non-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 non-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
90h	Previously addressed with general call address; DATA has been received, ACK returned.	Read data byte	X	0	0	0	Data byte will be received and not-ACK will be returned.
		or read data byte	X	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 4-6 · Status Register—Slave Receiver Mode (continued)

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
98h	Previously addressed with general call address; DATA has been received, not-ACK returned.	Read data byte	0	0	0	0	Switched to non-addressed SLV mode; no recognition of own SLA or general call address.
		or read data byte	0	0	0	1	Switched to non-addressed SLV mode; own SLA or general call address will be recognized.
		or read data byte	1	0	0	0	Switched to non-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 non-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
A0h	A STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX.	No action	0	0	0	0	Switched to non-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to non-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to non-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 non-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
D8h	25 ms SMBCLK low time has been reached; device must be reset.	No action	X	X	0	X	Slave must proceed to reset state by clearing the interrupt within 10 ms, according to SMBus Specification 2.0.

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 4-7 · Status Register—Slave Transmitter Mode

Status Code	Status	Data Register Action	Control Register Bits				Next Action Taken by Core
			sta	sto	si	aa	
A8h	Own SLA + R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
B0h	Arbitration lost in SLA + R/W as Master; own SLA + R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
B8h	Data byte has been transmitted; ACK has been received.	Load data byte	X	0	0	0	Last data byte will be transmitted; ACK will be received.
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
C0h	Data byte has been transmitted; not-ACK has been received.	No action	0	0	0	0	Switched to non-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to non-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to non-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 non-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
C8h	Last data byte has been transmitted; ACK has been received.	No action	0	0	0	0	Switched to non-addressed SLV mode; no recognition of own SLA or general call address.
		or no action	0	0	0	1	Switched to non-addressed SLV mode; own SLA or general call address will be recognized.
		or no action	1	0	0	0	Switched to non-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 non-addressed SLV mode; own SLA or general call address will be recognized; START condition will be transmitted when the bus becomes free.
D8h	25 ms SMBCLK low time has been reached; device must be reset.	No action	X	X	0	X	Slave must proceed to reset state by clearing the interrupt within 10 ms, according to SMBus Specification 2.0.

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.*

Data Register

The Data Register 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 4-8 is listed in both data 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 4-8 · Data Register

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

Address Register

The Address Register, described in Table 4-9, is a read/write, directly accessible register.

Table 4-9 · Address Register

Bit	Name	Function
7	adr6	Own Slave address bit 6
6	adr5	Own Slave address bit 5
5	adr4	Own Slave address bit 4
4	adr3	Own Slave address bit 3
3	adr2	Own Slave address bit 2
2	adr1	Own Slave address bit 1
1	adr0	Own Slave address bit 0
0	gc	General Call Address Acknowledge. If the gc bit is set, the general call address is recognized; otherwise it is ignored.

SMBus Register

The SMBus Register contains specific SMBus-related functionality and is readable or writable as defined in Table 4-10.

Table 4-10 · SMBus Register

Bit	Type	Name	Function
7	R/W	smbus_mst_reset	Writing a '1' 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 core is used as a Host controller (Master), the user can decide to reset the bus by holding the clock line low for 35 ms. Slaves must react to this event and reset themselves.
6	R/W	SMBSUS_NO Control	SMBSUS_NO control; used in Master/Host mode to force other devices into power-down/suspend mode. Active low. Note: SMBSUS_NO and SMBSUS_NI are separate signals (not wired-AND). If CoreSMBus is part of a Host controller, SMBSUS_NO can be used as an output; if CoreSMBus is a Slave to a Host controller that has implemented SMBSUS_N, only SMBSUS_NI's status is relevant.
5	R	SMBSUS_NI Status	Status of SMBSUS_NI signal. Note: SMBSUS_NO and SMBSUS_NI are separate signals (not wired-AND). If CoreSMBus is part of a Host controller, SMBSUS_NO can be used as an output; if CoreSMBus is a Slave to a Host controller that has implemented SMBSUS_N, only SMBSUS_NI's Status is relevant.
4	R/W	SMBALERT_NO Control	SMBALERT_NO control; used in Slave/Device mode to force communication with the Master/Host. Wired-AND.
3	R	SMBALERT_NI Status	Status of SMBALERT_NI signal. Wired-AND.

5 Testbench Operation and Modification

User Testbench

An example user testbench is included with the Evaluation, Obfuscated, and RTL releases of CoreSMBus. The user testbench is provided in precompiled Model*Sim* format for the Evaluation release. The Obfuscated and RTL releases provide the precompiled Model*Sim* format, as well as the source code for the user testbench, to ease the process of integrating the CoreSMBus macro into a design and verifying it. A block diagram of the example user design and testbench is shown in Figure 5-1.

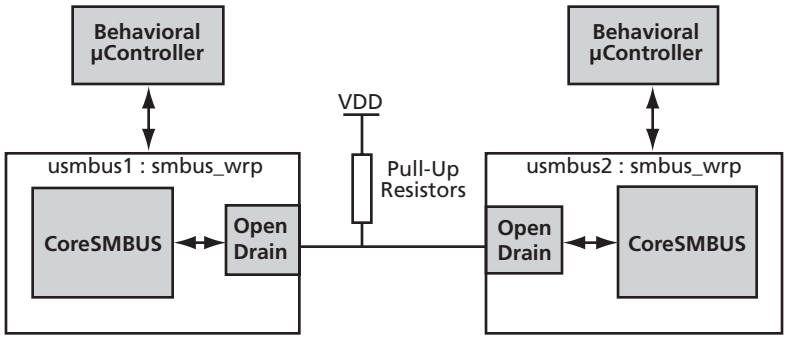

Figure 5-1 · CoreSMBus User Testbench

The user testbench includes a simple example design that serves as a reference for users who want to integrate CoreSMBus into their own designs. RTL source code for the example design and user testbench shown in Figure 5-1 is included in the *user* directory for all releases of the core. The example design source files and user testbench are listed in Table 5-1.

Table 5-1 · CoreSMBus Sample User Testbench RTL Source Code

CoreSMBus User RTL	
Verilog	**VHDL**
source/smbus_wrp.v	*coreconsole/ccproject/CORESMBUS/rtl/test/user/smbus_wrp.vhd*
source/user_smbustb.v	*coreconsole/ccproject/CORESMBUS/rtl/test/user/testbench.vhd*

Conceptually, as shown in Figure 5-1, two instantiations of the CoreSMBus macro are connected to an SMBUS bus emulated in the user testbench. Example transmit and receive between the two CoreSMBus units is demonstrated by the user testbench so you can gain a basic understanding of how to use this core.

The source code for the user testbench contains an example wrapper to aid in FPGA implementation of the open-drain connectors for SMBCLK, SMBDAT, and SMBALERT_N. Refer to the comments in the user testbench source code for details on the support routines (tasks for Verilog testbenches; functions and procedures for VHDL testbenches).

To run the user testbench, refer to "Simulating with the Simple User Application Testbench" on page 14.

6 System Operation

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

Use with CoreMP7

CoreSMBus can also be used with CoreMP7, the Actel soft IP version of the popular ARM7TDMI-S™ microprocessor that has been optimized for the M7 Fusion flash-based FPGA devices. To create a design using CoreMP7, internal flash memory, and CoreSMBus, you should use the CoreConsole IDP software. Refer to the CoreConsole documentation for information on creating your CoreMP7-based design. Figure 6-1 gives an example design.

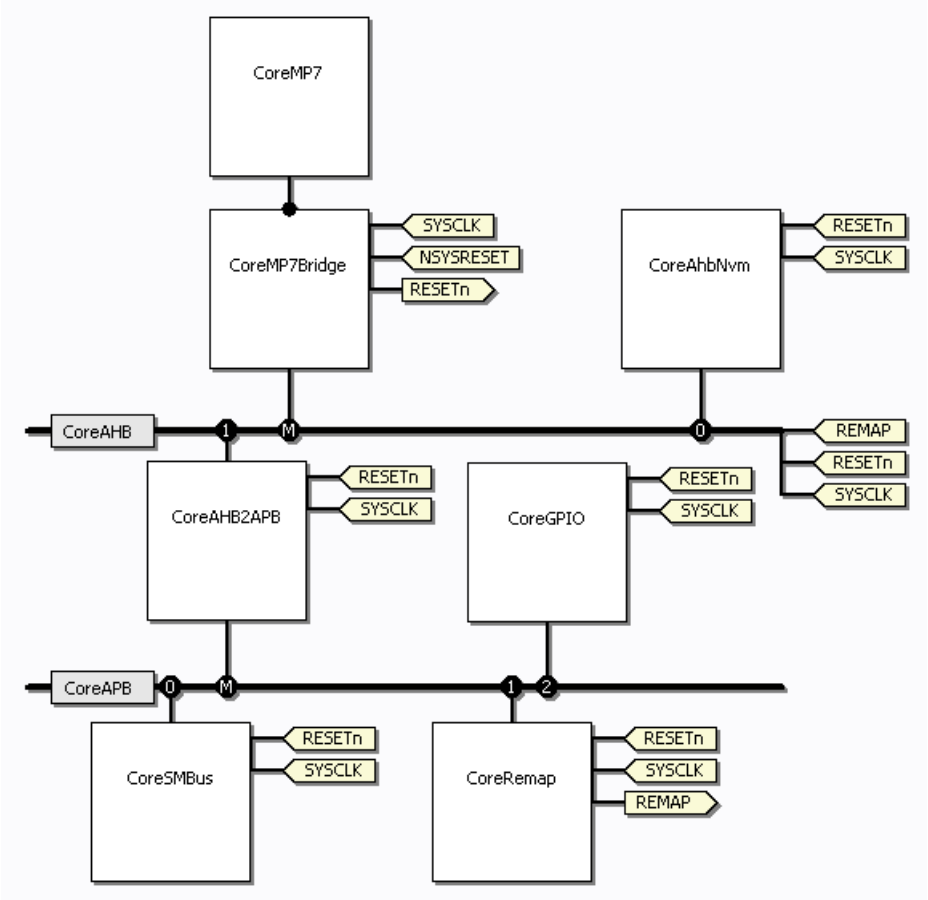

Figure 6-1 · Example System Using CoreMP7 and CoreSMBus

Use with Core8051

CoreSMBus can also be used with Core8051. An example FPGA design using Core8051 and CoreSMBus is shown in Figure 6-2. For this example, the internal flash memory is used for Core8051 program storage and can be programmed independently of the FPGA fabric by use of the FlashPro software and hardware (refer to the *FlashPro v5.0 User's Guide* for details on how to program the flash memory within the Fusion device).

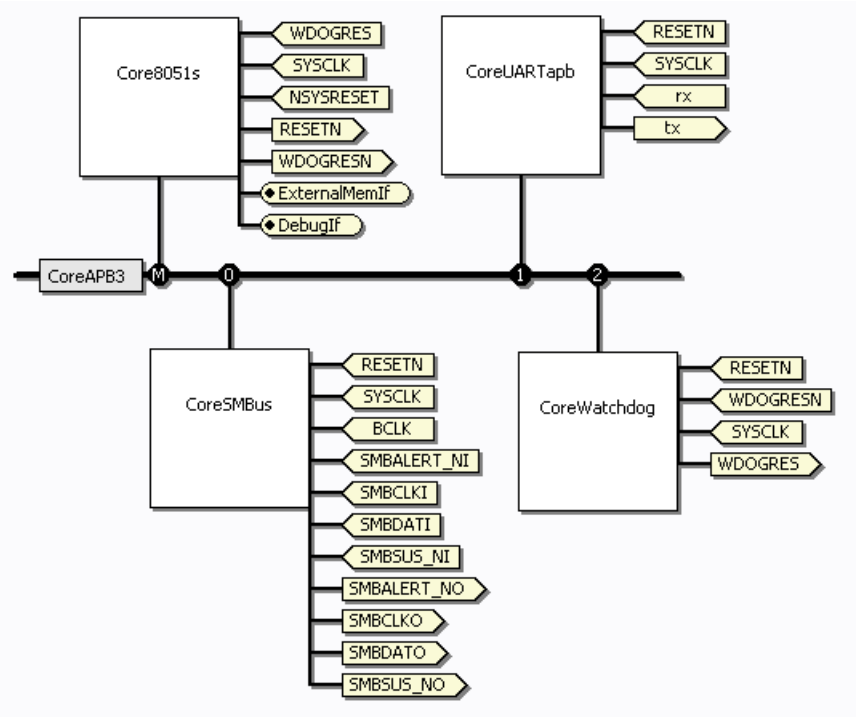

Figure 6-2 · Example System Using Core8051 and CoreSMBus

Use with CoreABC

CoreSMBus can also be used with CoreABC. An example FPGA design using CoreABC and CoreSMBus is shown in Figure 6-3. CoreABC allows a simple set of APB read and write cycles that can be used to configure CoreSMBus and then to read and compare the analog values to turn the digital outputs on and off.

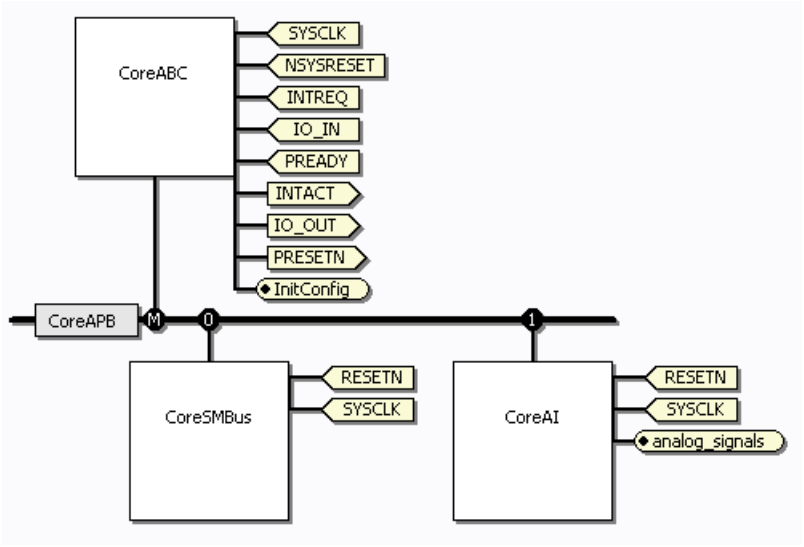

Figure 6-3 · Example System Using CoreABC and CoreSMBus

7 Software Drivers

Example software drivers are available from Actel for CoreSMBus. Contact Actel Technical Support for information (tech@actel.com).

A

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 Version (v3.0)	Page
v2.0	The data transfer rate and the SLAVE_EN_ONLY parameter for Master receiver mode were updated in the "Key Features" section.	5
	Figure 1 · CoreSMBus Block Diagram was updated.	6
	The first two paragraphs of the "SMBus Serial Interface" section were updated.	16
	Figure 3-3 · Data Write Cycle was updated to change the signal PRDATA to PWDATA.	18
	The second table note, which stated the clock rate frequency of 100 kbps should not be exceeded, was removed from Table 4-2 · Control Register.	20

B

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/custsup/search.html) 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/contact/offices/index.html.

Index

For more information about Actel's products, visit our website at http://www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 USA*
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

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 Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • www.jp.actel.com

Actel Hong Kong *• Suite 2114, Two Pacific Place • 88 Queensway, Admiralty Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

50200082-1 /7.07