CoreI2C

Handbook

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 50200090-1

Release: July 2007

Trademarks

Table of Contents

Introduction

Core Overview

Key Features

- Supports Philips Inter-Integrated Circuit (I^2C) v2.1
- 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

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

- I2CCLK (serial clock line)
- I2CDAT (serial data line)

Byte transfers across the serial line and APB interface are handled autonomously by CoreI2C. 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 CoreI2C functional block diagram.

Each device connected to the bus is software-addressable by a unique address. CoreI2C 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. CoreI2C 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

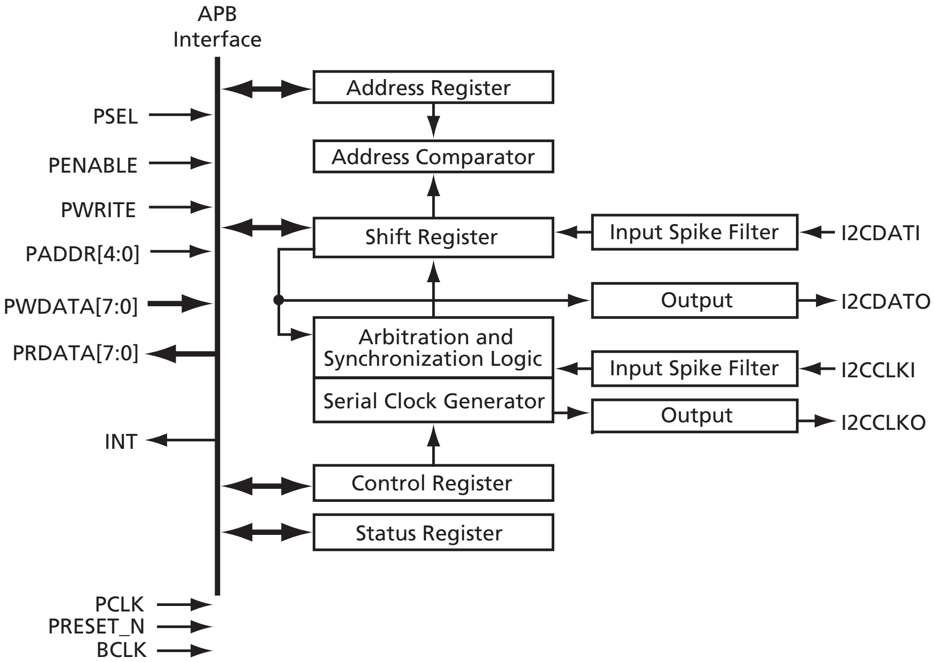


Figure 1 · CoreI2C Block Diagram

Figure 2 illustrates a typical application of CoreI2C. In this example, FPGA #1 is the I^2C master, with CoreMP7 controlling CoreI2C. CoreMP7, coupled with CoreI2C in FPGA #1, controls communication between FPGA #1 and an I^2C device, as well as FPGA #2, where CoreI2C is configured in Slave-only mode with CoreABC.

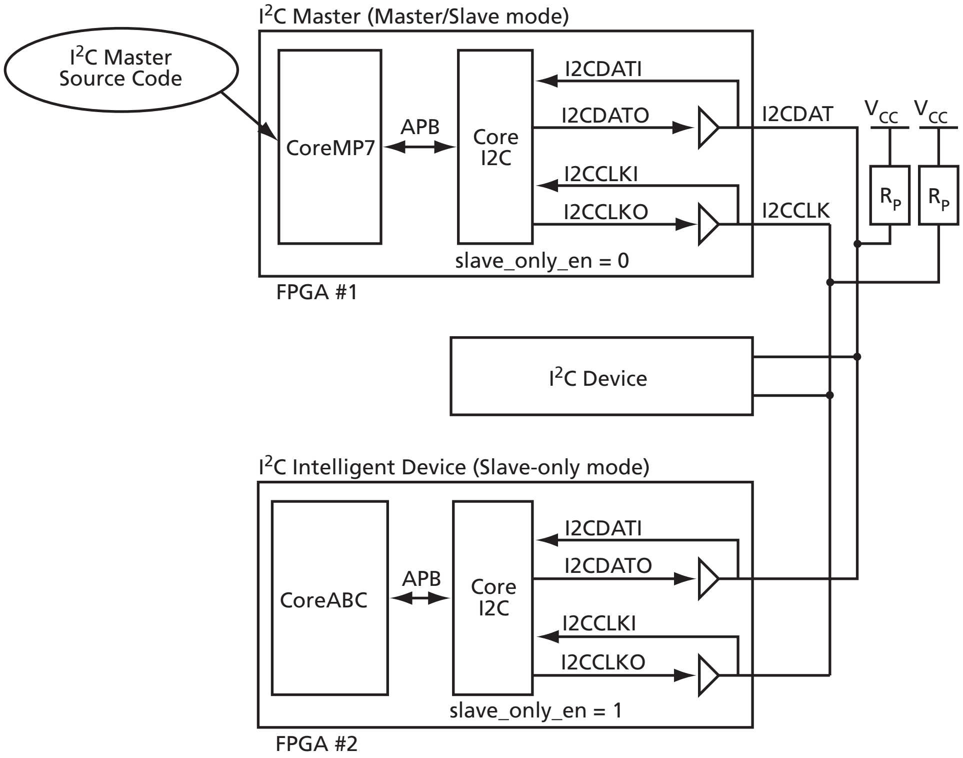


Figure 2 · CoreI2C Application Example

Utilization and Performance

CoreI2C can be implemented in any Actel device. A summary of CoreI2C utilization and performance for various devices is listed in Table 1 and Table 2.

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

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	80	372	452	AFS600	3%	113 MHz
IGLOO™	80	368	448	AGL600	3%	75 MHz
ProASIC®3/E	80	368	448	M7A3P250	7%	116 MHz
ProASIC<u>PLUS</u>®	75	419	494	APA075	16%	79 MHz
Axcelerator®	88	265	353	AX250	9%	142 MHz
RTAX-S	88	265	353	RTAX250S	8%	106 MHz

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

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

Family	Tiles			Utilization		Performance
	Sequential	Combinatorial	Total	Device	Total	
Fusion	117	582	699	AFS600	5%	112 MHz
IGLOO	119	581	700	AGL600	5%	71 MHz
ProASIC3/E	119	581	700	M7A3P250	11%	118 MHz
ProASIC<u>PLUS</u>	113	632	745	APA075	24%	69 MHz
Axcelerator	123	380	503	AX250	12%	151 MHz
RTAX-S	123	380	503	RTAX250S	12%	104 MHz

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

Supported Interfaces

CoreI2C is available with the following interfaces:

- APB interface for register access
- Serial interface

These interfaces are further described in "Interface Definitions" on page 14.

1

Functional Block Descriptions

CoreI2C, 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

CoreI2C 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 17 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. 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.

Serial Clock Generator

This programmable clock pulse generator provides the serial bus clock pulses when CoreI2C is in Master mode. The clock generator is switched off when CoreI2C is in Slave mode. The baud rate clock (BCLK) is a pulse-for-transmission speed control signal and is internally synchronized with the clock input. BCLK 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 18 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

CoreI2C 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 12. 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

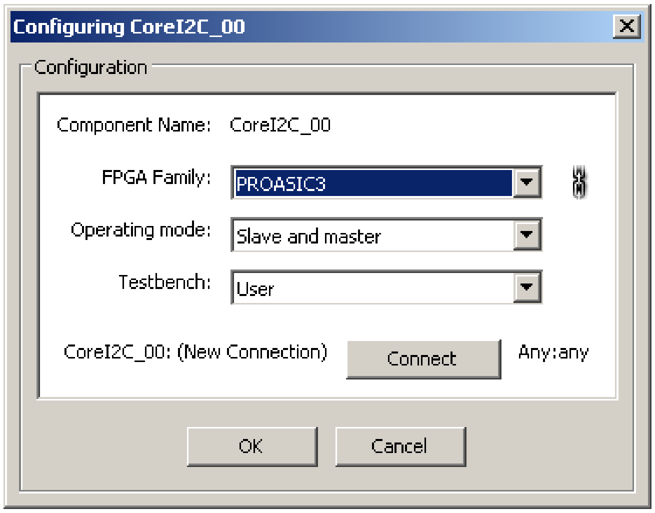
CoreI2C is preinstalled in the CoreConsole Intellectual Property Deployment Platform (IDP). To use the core,[2] drag it from the IP Core list into the main window. The CoreConsole project can be exported to Actel Libero IDE at this point, providing access to CoreI2C. Other IP blocks can interconnect, 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 CoreI2C 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 14.



Component Name: CoreI2C_00

FPGA Family: PROASIC3

Operating mode: Slave and master

Testbench: User

CoreI2C_00: (New Connection) Any:any

Figure 2-1. CoreI2C 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 CoreI2C 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. CoreI2C 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 CoreI2C macro are illustrated in Figure 3-1 and defined in Table 3-1. CoreI2C has 32 I/O signals.

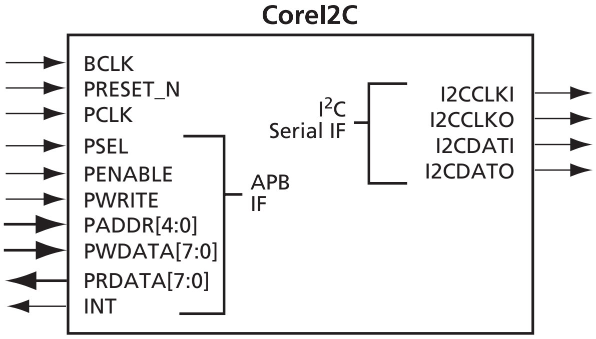


Figure 3-1 · CoreI2C I/O Signal Diagram

Table 3-1 · CoreI2C 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 CoreI2C registers.
PSEL	Input	APB Slave Select – This signal selects CoreI2C 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 CoreI2C takes place; if LOW, a read from CoreI2C will take place.
PWDATA[7:0]	Input	APB write data
PRDATA[7:0]	Output	APB read data
INTERRUPT	Output	Microprocessor interrupt output
I^2C Serial Interface		
I2CCLKI	Input	Wired-AND I^2C serial clock input
I2CCLKO	Output	Wired-AND I^2C serial clock output
I2CDATI	Input	Wired-AND I^2C serial data input
I2CDATO	Output	Wired-AND I^2C serial data output
Other Signals		
BCLK	Input	Pulse for I2CCLK 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

CoreI2C 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 · CoreI2C Parameter/Generic Descriptions

Name	Valid Range	Description
FAMILY	0 to 99	Must be set to match the supported FPGA family: 11 – Axcelerator 12 – RTAX-S 14 – ProASIC^{PLUS} 15 – ProASIC3 16 – ProASIC3E 17 – Fusion 20 – IGLOO 21 – IGLOOe
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.

Interface Definitions

CoreI2C includes the following interfaces:

- I^{2}C serial interface
- APB register interface

I^{2}C Serial Interface

The AC timing diagram is shown in Figure 3-2.

For detailed timing parameter information, refer to the Philips Inter-Integrated Circuit v2.1 Specification.

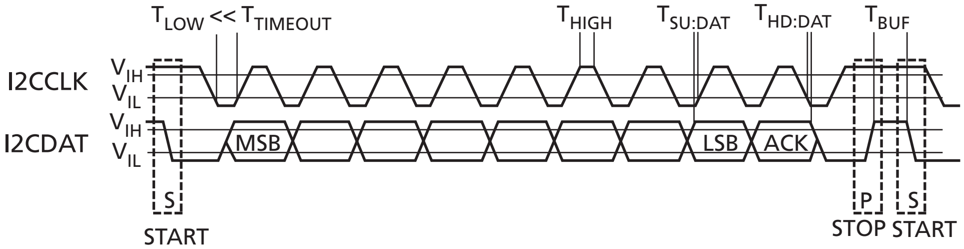


Figure 3-2 · I^{2}C Timing Diagram

APB Interface

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

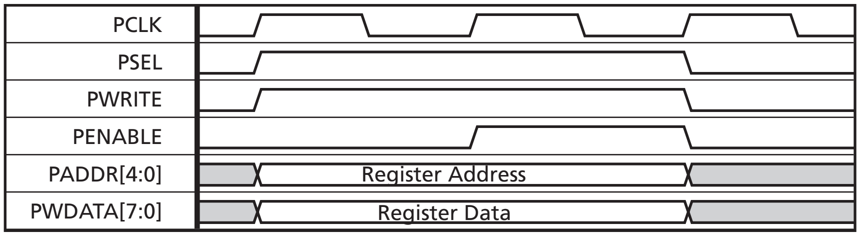


Figure 3-3 · Data Write Cycle

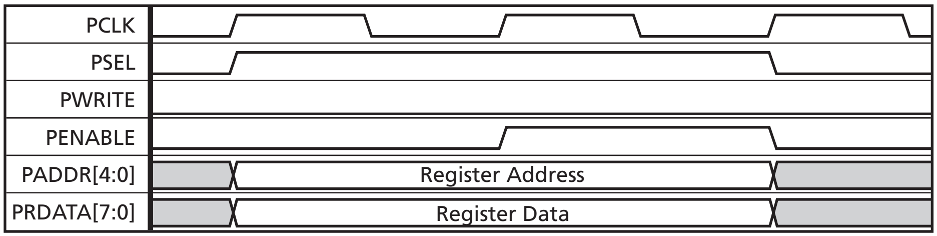


Figure 3-4 · Data Read Cycle

4

Register Map

APB Register Map

The internal register address map and reset values of each APB-accessible register for CoreI2C 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 · CoreI2C 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.

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

Register Descriptions

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

Control Register

The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits are affected by 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 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 CoreI2C 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	I2CCLK 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 22. 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

CoreI2C logic can operate in the following four modes:

1. Master Transmitter Mode
   Serial data output through I2CDAT while I2CCLK puts out the serial clock.
2. Master Receiver Mode
   Serial data is received via I2CDAT while I2CCLK puts out the serial clock.
3. Slave Receiver Mode
   Serial data and the serial clock are received through I2CDAT and I2CCLK.
4. Slave Transmitter Mode
   Serial data is transmitted via I2CDAT while the serial clock is received through I2CCLK.

Below are examples of using the Status Register to operate CoreI2C 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 22).
- 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 22).
- Transfer is continued according to Table 4-6 on page 22.

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 20).
- 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 20).
- Transfer is continued according to Table 4-4 on page 20.

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.
38h	Arbitration lost in SLA + R/W or data bytes.	No action	0	0	0	X	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.

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; CoreI2C will be switched to MST/TRX mode.
38h	Arbitration lost in not-ACK bit.	No action	0	0	0	X	Bus will be released; CoreI2C 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.

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.

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.

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.

5

Testbench Operation and Modification

User Testbench

An example user testbench is included with the Evaluation, Obfuscated, and RTL releases of CoreI2C. 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 CoreI2C macro into a design and verifying it. A block diagram of the example user design and testbench is shown in Figure 5-1.

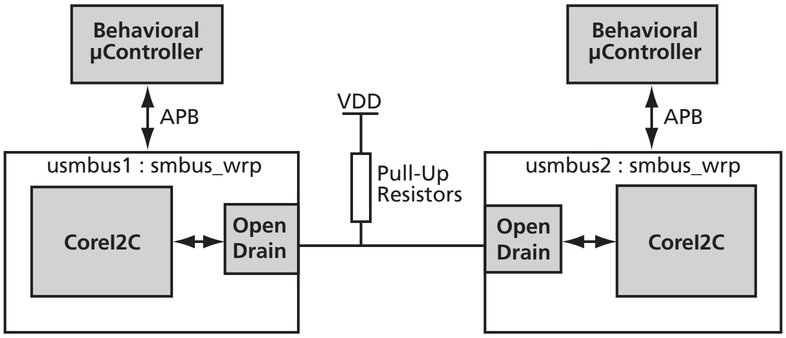


Figure 5-1 · CoreI2C User Testbench

The user testbench includes a simple example design that serves as a reference for users who want to integrate CoreI2C 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 · CoreI2C Sample User Testbench RTL Source Code

CoreI2C User RTL	
Verilog	**VHDL**
coreconsole/ccproject/COREI2C/rtl/test/user/smbus_wrp.v	*coreconsole/ccproject/COREI2C/rtl/test/vhdl/user/smbus_wrp.vhd*
coreconsole/ccproject/COREI2C/rtl/test/user/testbench.v	*coreconsole/ccproject/COREI2C/rtl/test/vhdl/user/testbench.vhd*

Conceptually, as shown in Figure 5-1, two instantiations of the CoreI2C macro are connected to an I^2C bus emulated in the user testbench. Example transmit and receive between the two CoreI2C 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 I2CCLK and I2CDAT. 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 12.

System Operation

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

Use with CoreMP7

CoreI2C can also be used with CoreMP7, the Actel soft IP version of the popular ARM7TDMI-S^{TM} microprocessor that has been optimized for the M7 Fusion flash-based FPGA devices. To create a design using CoreMP7, internal flash memory, and CoreI2C, 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.

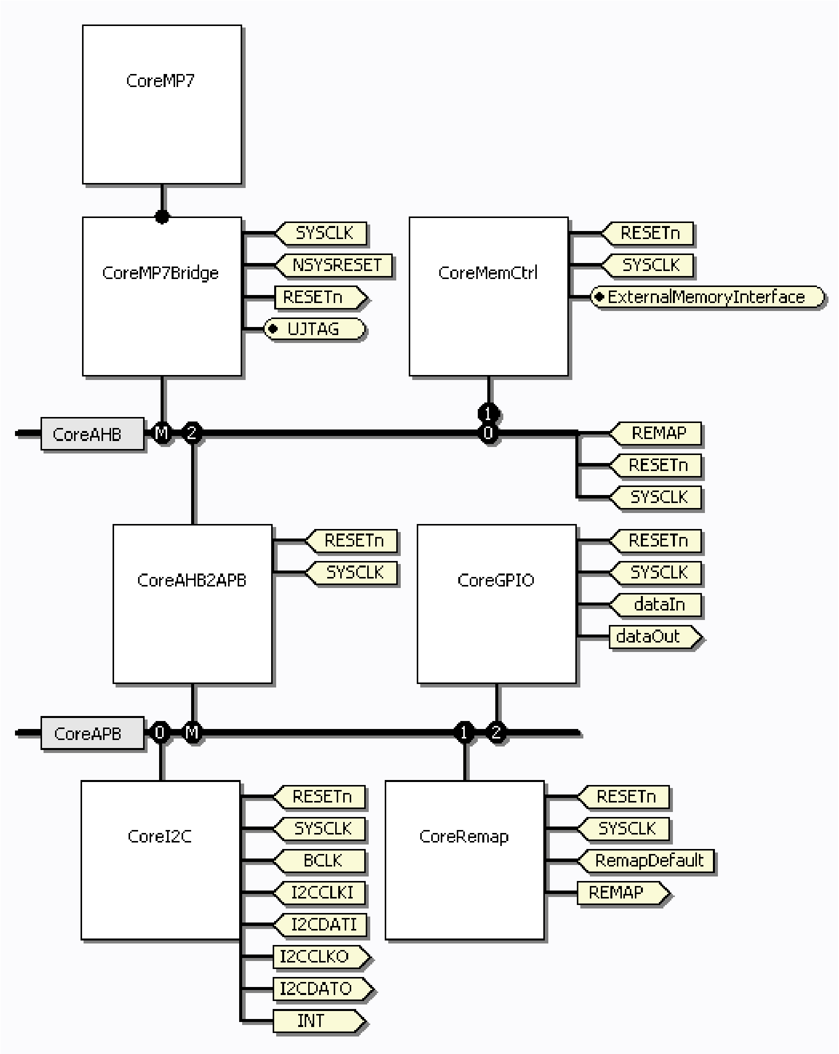


Figure 6-1 · Example System Using CoreMP7 and CoreI2C

Use with Core8051

CoreI2C can also be used with Core8051. An example FPGA design using Core8051 and CoreI2C 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).

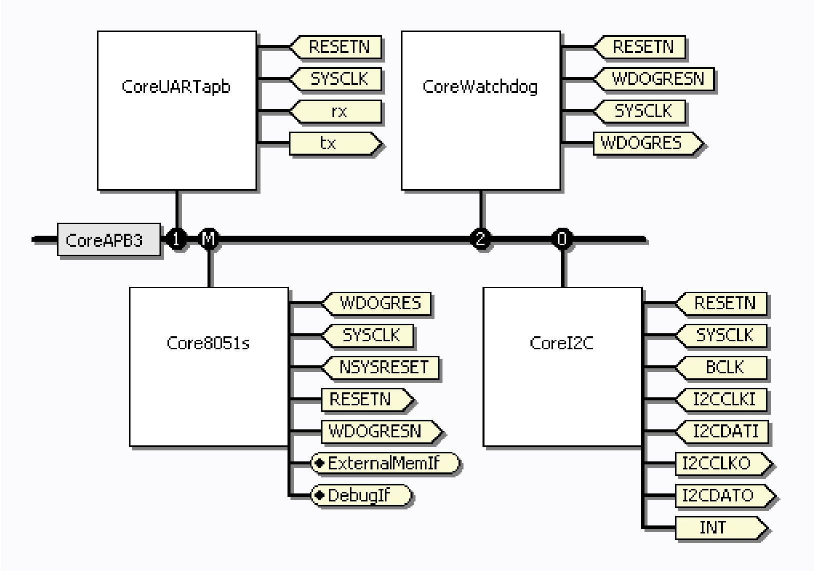


Figure 6-2 · Example System Using Core8051 and CoreI2C

Use with CoreABC

CoreI2C can also be used with CoreABC. An example FPGA design using CoreABC and CoreI2C is shown in Figure 6-3. CoreABC allows a simple set of APB read and write cycles that can be used to configure CoreI2C and then to read and compare the analog values to turn the digital outputs on and off.

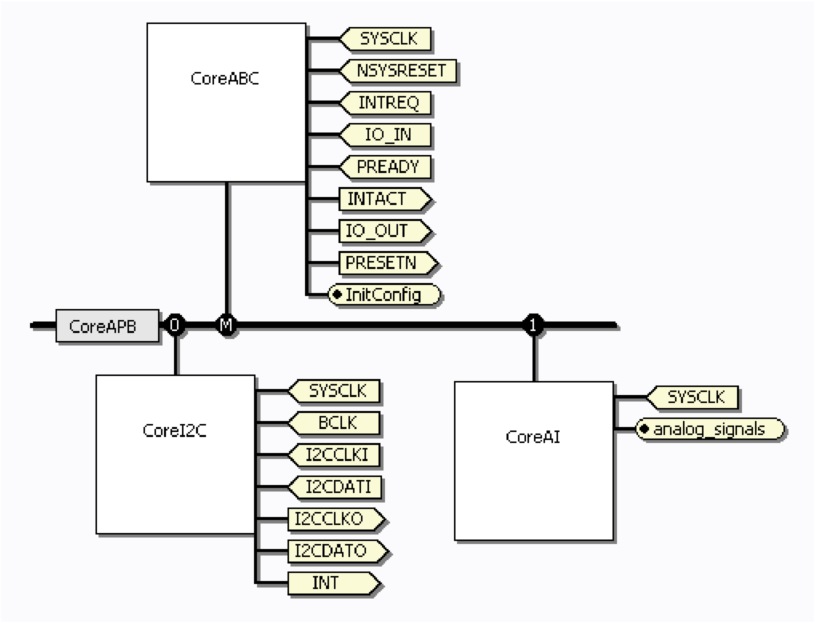


Figure 6-3 · Example System Using CoreABC and CoreI2C

7

Software Drivers

Example software drivers are available from Actel for CoreI2C. 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 · CoreI2C Block Diagram was updated.	6
	The first two paragraphs of the "I2C Serial Interface" section were updated.	14
	Figure 3-1 · CoreI2C I/O Signal Diagram was updated to remove the BCLK signal.	13
	Figure 3-3 · Data Write Cycle was updated to change the signal PRDATA to PWDATA.	15
	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.	18

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