
Microcontroller I/O Expander Design Example

Contents

General Description	1
Design Description	1
Interface Description	3
Utilization Details	3
Software Interface and Design Details	3
Testing Scheme	5
Timing Diagrams	6
Conclusion	7

General Description

The general purpose inputs and outputs (GPIO) ports available for a microcontroller are usually limited in number, but many applications require more ports than are available on the microcontroller. This design example illustrates how to add four 8-bit ports to any 8-bit microcontroller using Actel's low-power FPGAs. This example uses Actel Core8051s as the example 8-bit microcontroller. The direction of each port can be configured as input or output on the fly. The microcontroller bus is CoreAPB3 compatible, and is tested and verified with an Actel ProASIC[®]3 device.

Design Description

This design block can be used as a memory-mapped device of the 8051s embedded processor. The base address of this core is created during the processor configuration with Actel's SmartDesign tool. An active high chip select enables the block and several registers within the core are assigned offset addresses. The absolute address of each of these registers will be the base address plus the offset address.

Figure 1 on page 2 shows the top-level block diagram.

Design files can be downloaded from the Actel website:
www.actel.com/download/rsc/?f=MC_IO_Expander_DF.

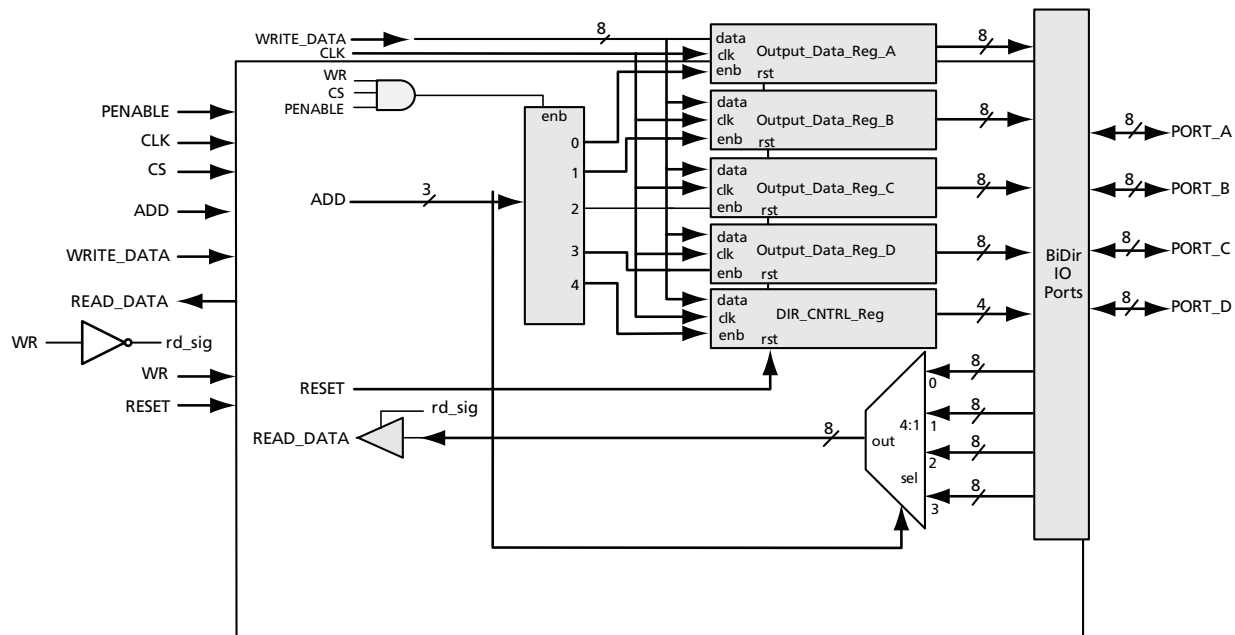


Figure 1 • Top-Level Block Diagram

This design has four ports: PORT_A, PORT_B, PORT_C, and PORT_D. Each of these ports can be configured as either an input or output 8-bit port. For writing or reading data from any port, the direction (input or output) bit of the direction control register (DIR_CNTRL_Reg) must be configured (Figure 2).

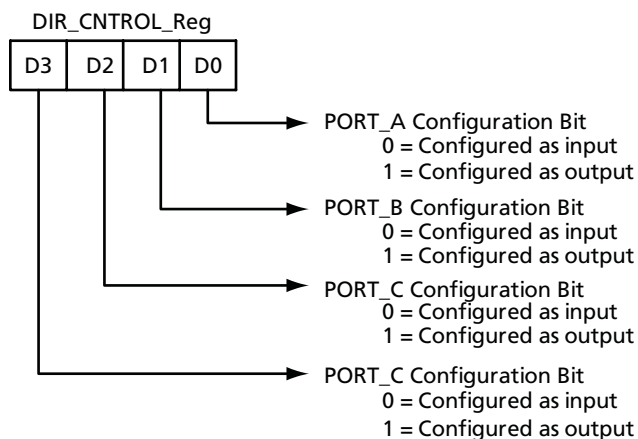


Figure 2 • Direction Control Register

To perform a write operation to any of these ports, first set the direction as output for the port, address the corresponding port, and then write the data (data are written to the corresponding port output register.) When the WR signal is asserted (with CS and PENABLE), the output registers of that particular port are enabled. The content of the WRITE_DATA bus will be written to the output data register of the selected port on the rising edge of clock. The content of WRITE_DATA will remain at the port until the port content is overwritten.

For reading data from any port, set the direction of that port as input and load the address bus with the address of that port. Data will be transferred when RD is asserted for the READ_DATA bus and the controller will read the data.

Interface Description

Table 1 describes the interconnections between the ports and the I/Os.

Table 1 • Block Interface Description

Port	Direction	Description
CLK	Input	System clock
PENABLE	Input	Enable signal for latching the write data
WRITE_DATA	Input	8-bit wide write data to the block
READ_DATA	Output	8-bit wide read data from the block
ADD	Input	3-bit address input to the block
WR	Input	Active high write signal
CS	Input	Chip select for the block
RESET	Input	Active high reset to the block <i>Note: The top level of the design utilizes the active low NRST_IN port, which is inverted and passed to RESET.</i>
PORT_A	Bidirectional	8-bit data port A
PORT_B	Bidirectional	8-bit data port B
PORT_C	Bidirectional	8-bit data port C
PORT_D	Bidirectional	8-bit data port D

Utilization Details

For testing purposes, this design was created for the M1A3P1000-FG484 device. Table 2 describes the utilization details for this design.

Table 2 • Utilization Details

Resource	Used/Available	Percentage
Core tiles	130/24,576	0.53%
GLOBAL (chip + quadrant)	0	0.0%
PLL	0	0.0%
RAM/FIFO	0	0.0%

Software Interface and Design Details

The application software is written in C language. The program must be initially downloaded to the program memory of the 8051s. The offset address corresponding to the various registers is hardcoded. When a write cycle has to be performed, the direction control bit is initially set, followed by a write to that port. For a read cycle, the corresponding direction control bit is set and a read is performed from that port.

Software Files

The *Actel_IO_Expander_regs.h* file defines the core register map, providing symbolic constants to access the low-level hardware.

Macros

IO_EXP_WR_CONFIG_REG (char Direction, char PortNumber)

This API is used to configure the direction control of the ports.

Parameters

Direction: Sets the direction control for the ports.

1: Sets the direction of the port to OUTPUT.

0: Sets the direction of the port to INPUT.

PortNumber: Sets the port numbers to be configured by the block for reading or writing a value.

00: PORT_A

01: PORT_B

02: PORT_C

03: PORT_D

Return Value:

None

IO_EXP_WR_TO_PORTA (char Data)

This API is used to write data to the output register of PORT_A.

Parameters

Data: Data to be written to PORT_A.

Return Value:

None

IO_EXP_WR_TO_PORTB (char Data)

This API is used to write data to the output register of PORT_B.

Parameters

Data: Data to be written to PORT_B.

Return Value:

None

IO_EXP_WR_TO_PORTC (char Data)

This API is used to write data to the output register of PORT_C.

Parameters

Data: Data to be written to PORT_C.

Return Value:

None

IO_EXP_WR_TO_PORTD (char Data)

This API is used to write data to the output register of PORT_D.

Parameters

Data: Data to be written to PORT_D.

Return Value:

None

IO_EXP_RD_FROM_PORTA (char Data)

This API is used to read data from the input register of PORT_A.

Parameters

Datum: Contains the value read from PORT_A.

Return Value:

None

IO_EXP_RD_FROM_PORTB (char Data)

This API is used to read data from the input register of PORT_B.

Parameters

Datum: Contains the value read from PORT_B.

Return Value:

None

IO_EXP_RD_FROM_PORTC (char Data)

This API is used to read data from the input register of PORT_C.

Parameters

Datum: Contains the value read from PORT_C.

Return Value:

None

IO_EXP_RD_FROM_PORTD (char Data)

This API is used to read data from the input register of PORT_D.

Parameters

Datum: Contains the value read from PORT_D.

Return Value:

None

Table 3 describes the register mapping for all ports.

Table 3 • Register Mapping

Address	Register Name	R/W	7	...	2	1	0
0xF101	PORT_A	R/W	Data to read/write on PORT_A				
0xF102	PORT_B	R/W	Data to read/write on PORT_B				
0xF103	PORT_C	R/W	Data to read/write on PORT_C				
0xF104	PORT_D	R/W	Data to read/write on PORT_D				
0xF105	DIR_CTRL	W	Bits [0:3] are used to set the direction for PORT_A, PORT_B, PORT_C, PORT_D				

Testing Scheme

The design was tested using the Actel ProASIC3 development board (M1A3P-DEV-KIT-SCS). Core8051s and the microcontroller I/O expander design are integrated and programmed to the M1A3P1000-FG484 device, although virtually any ProASIC3 or IGLOO® device may be used.

Actel's SoftConsole v2.1 is used for the software development and the FS2 ISA-Actel51 console is used for executing the program. The FPGA is initially programmed with the *MC_IO_Expander.stp* file using Actel's FlashPro programmer. Using ISA-Actel51 console, the *IO_Expander.hex* file is downloaded to the program memory. The "go" command is given from the console for executing the program.

Microcontroller I/O Expander Design Example

In this design example, the on-board switches are connected to one of the input ports (PORT_D) and the LEDs on the board are connected to one of the output ports (PORT_C). The software program continuously reads from the input port and sends data to the output port. Whenever the switch positions are changed, the LED indications also change correspondingly.

Figure 3 displays the test setup block diagram.

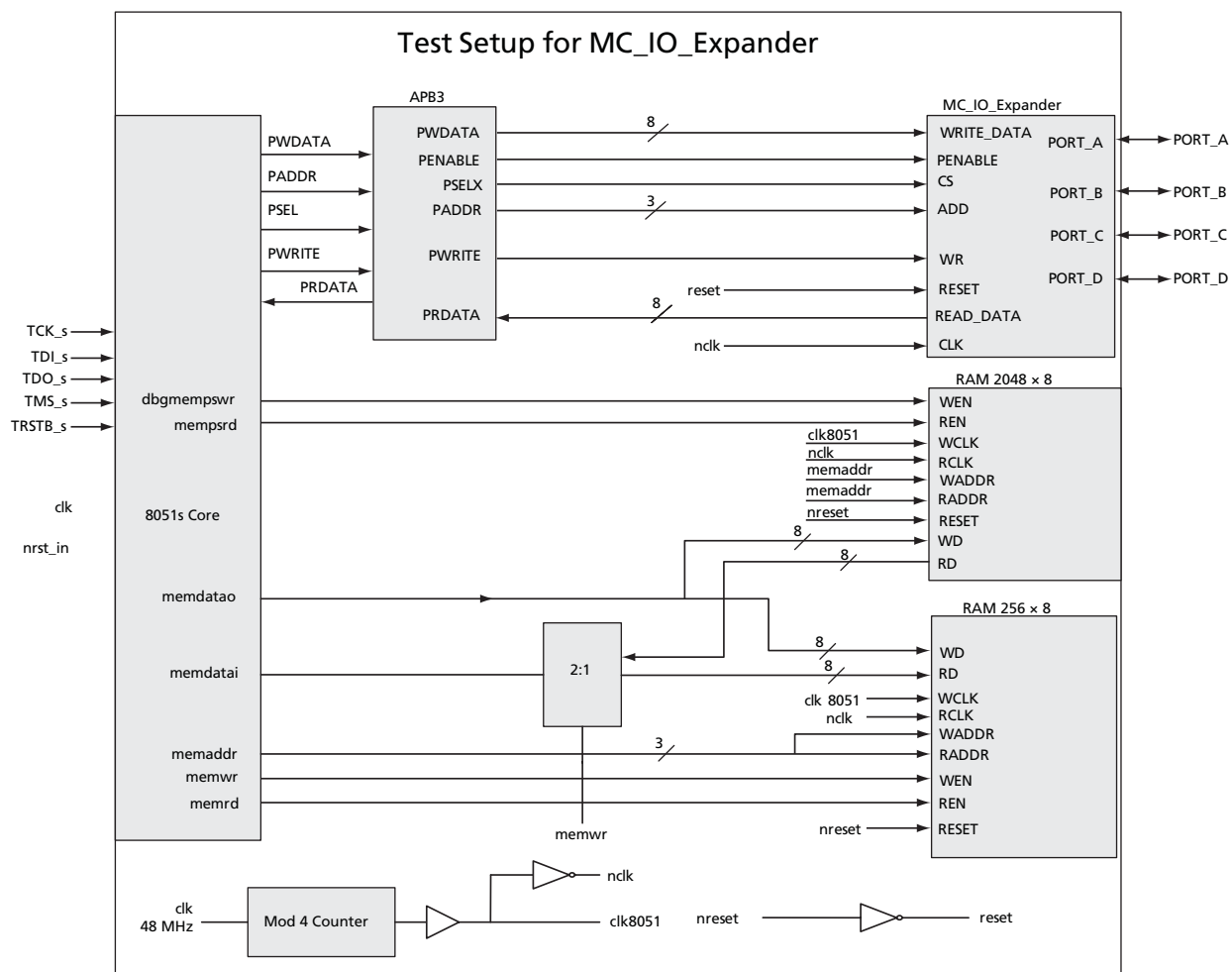


Figure 3 • Test Setup for the Microcontroller I/O Expander Design

Timing Diagrams

Figure 4 shows the sequence of a write cycle to PORT_C.

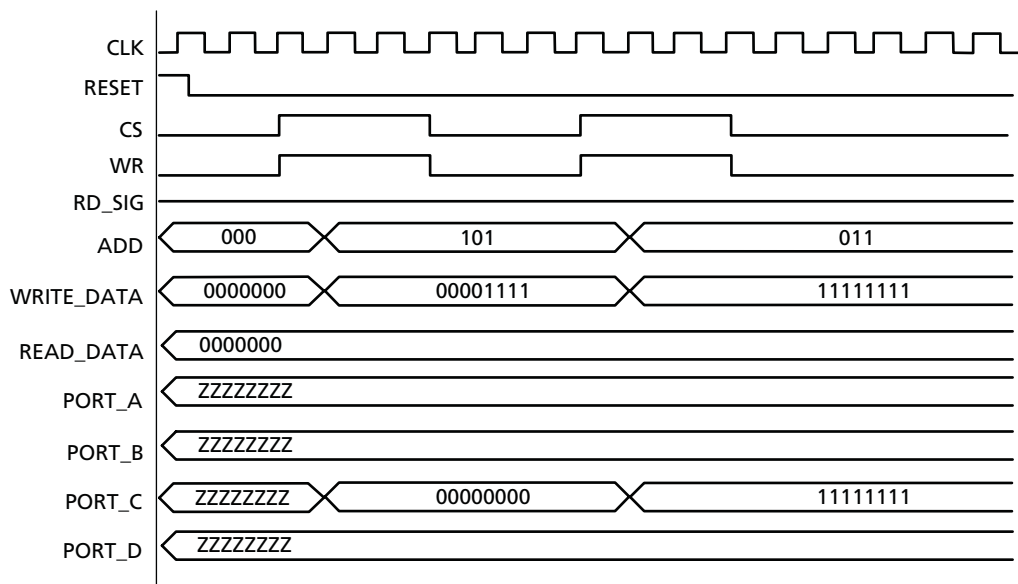


Figure 4 • Write Cycle to PORT_C

Figure 5 shows the sequence to be followed for a read from PORT_A.

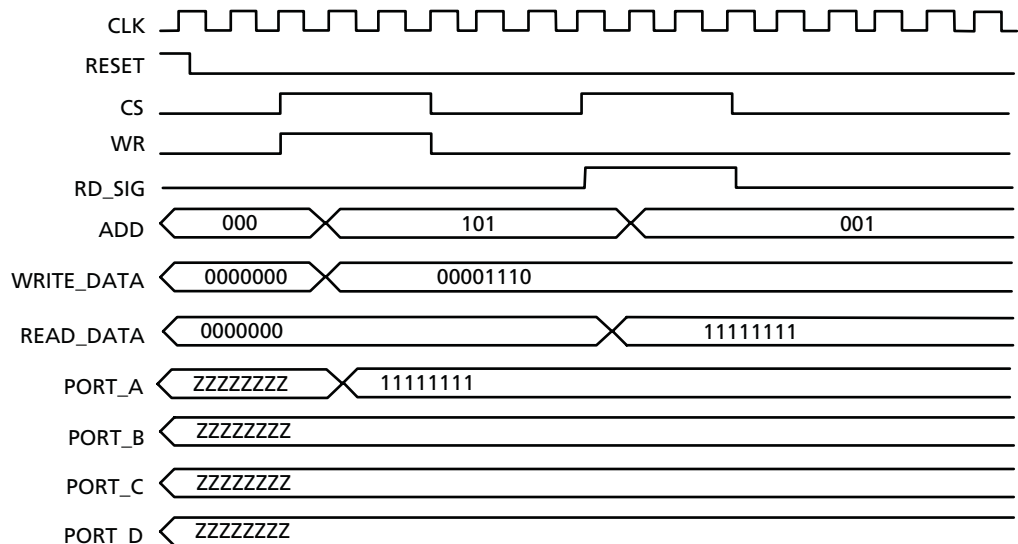


Figure 5 • Read Cycle from PORT_A

Conclusion

This design can be used in microcontroller-based setups where the number of ports needed exceeds the number of ports available on a microcontroller integrated circuit. This design requires minimal FPGA resources and can be accommodated by most of Actel's IGLOO and ProASIC3 FPGAs. Even though the design mentioned in this example is for four 8-bit ports, the code can be easily modified to add additional ports or change the width of the ports.

Actel and the Actel logo are registered trademarks of Actel Corporation.
All other trademarks are the property of their owners.



Actel is the leader in low-power 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 Building 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