Application Note
January 2002

ORCA® Series 3 Microprocessor Interface

ORCA Series 3 Microprocessor Interface

Introduction

With the increased demand of larger and faster FPGAs, one of the goals of FPGA designers is to utilize as much programmable logic as possible. To reach this goal, system-level features have been added to the Series 3 base architecture. These system-level features reduce glue-logic requirements and, hence, conserve programmable logic. Of the many system-level features, a multiuse microprocessor interface (MPI) can be used for configuration, readback, general-purpose interface to the FPGA, as well as other basic device control and status functions. The MPI is a synchronous, 8-bit interface, programmable to operate with both the *PowerPC* * MPC800 series microprocessor and *Intel i960*† J core processors (see Figure 1). As will be shown later, this interface is also very powerful when used to interface to many other processors as well, such as a DSP device, with minimal amounts of glue logic.

* *PowerPC* is a registered trademark of International Business Machines Corporation.

† *Intel* and *i960* are registered trademarks of Intel Corporation.

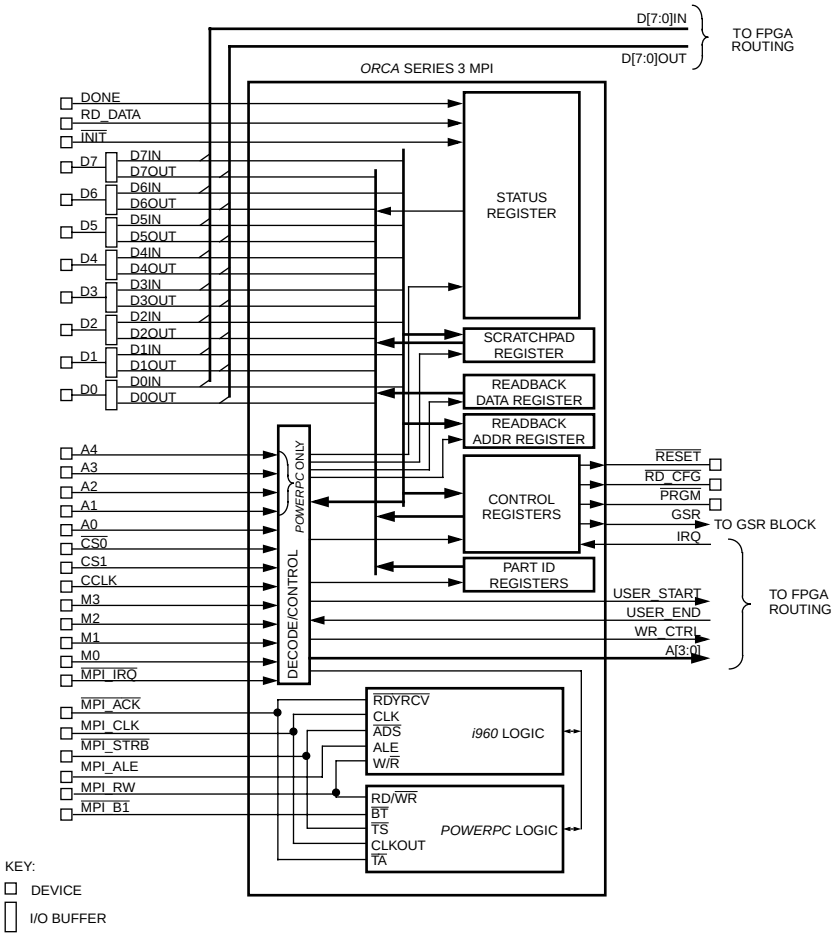


Figure 1. MPI Block Diagram

PowerPC System

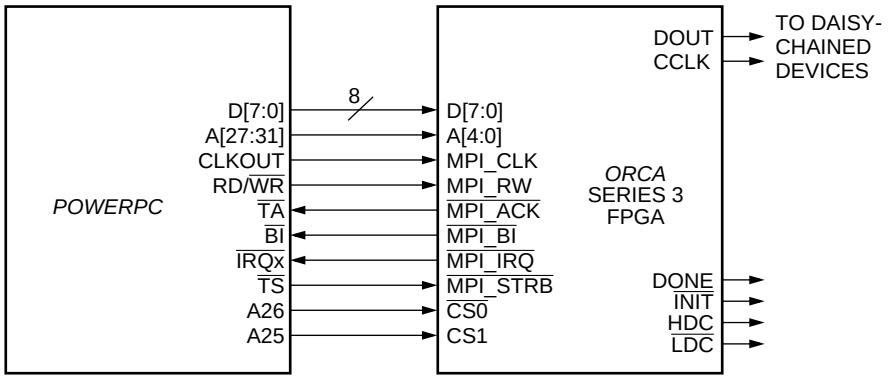
In Figure 2, the *ORCA* FPGA is a memory-mapped peripheral to the *PowerPC* processor. The A26 and A25 address lines are used to decode the memory map. Other forms of selection are possible by using the FPGA chip selects (CS0 and CS1) in a different way. If the MPI is not used to configure the FPGA, then decoding logic can be implemented internally to the FPGA. If the decode is implemented in the FPGA, then the decode logic is routed out on output pins and connected externally to the select lines. (Note this is not possible before configuration, so this is only available if the MPI is not used to configure the FPGA.)

Figure 2 shows the basic connection of the *PowerPC* to the FPGA. Pin descriptions are shown in Table 1. For both read and write transactions, the address, chip select, and read/write (read high, write low) signals are driven at the FPGA pins by the *PowerPC*. The *PowerPC* then asserts its transfer start signal (TS) low. Data is available to the MPI during a write at the rising clock edge after the clock cycle during which TS is low. The transfer is acknowledged to the *PowerPC* from the MPI by the low assertion of the TA signal. The MPI *PowerPC* interface does not support burst transfers, so the burst inhibit signal (BI) is also asserted low during the transfer acknowledge. The same process applies to a read from the MPI except that the read data is expected at the FPGA data pins by the *PowerPC* at the rising edge of the clock when TA is low. The MPI only drives TA low for one clock cycle.

Table 1. *PowerPC* MPI Configuration

***PowerPC* Signal**	***ORCA* Pin Name**	**MPI I/O**	**Function**
D[0:7]	D[7:0]	I/O	8-bit data bus.
A[27:31]	A[4:0]	I	5-bit MPI address bus.
$\overline{TS}$	$\overline{RD/MPI\_STRB}$	I	Transfer start signal.
—	$\overline{CS0}$	I	Active-low MPI select.
—	CS1	I	Active-high MPI select.
CLKOUT	A7/MPI_CLK	I	*PowerPC* interface clock.
RD/$\overline{WR}$	A8/MPI_RW	I	Read (high)/Write (low) signal.
$\overline{TA}$	A9/$\overline{MPI\_ACK}$	O	Active-low transfer acknowledge signal.
$\overline{BI}$	A10/$\overline{MPI\_BI}$	O	Active-low burst transfer inhibit signal.
Any of $\overline{IRQ}$[7:0}	A11/$\overline{MPI\_IRQ}$	O	Active-low interrupt request signal.

Interrupt requests can be sent to the *PowerPC* asynchronously to the read/write process. Interrupt requests are generated by the FPGA user logic. When an interrupt is requested, the MPI will assert the request to the *PowerPC* as a direct interrupt signal and/or by a bit in the MPI status register. The interrupt will continue to be asserted until the user logic in the FPGA deasserts the signal.



5-8507(F)

Figure 2. *PowerPC*/MPI Configuration Schematic

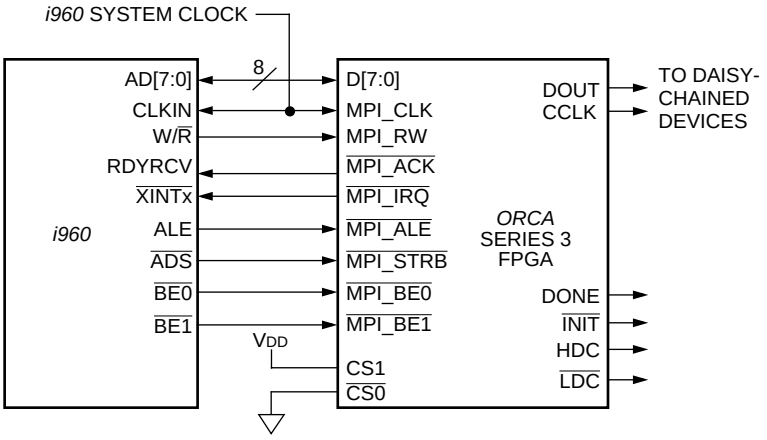
i960 System

Figure 3 shows the schematic for the connection of the *Intel i960* processor. Pin descriptions are shown in Table 2. The schematic shows the FPGA as only a system peripheral with fixed-chip selects. As mentioned in the *PowerPC* System section, the *i960* system can also memory map the FPGA.

The basic flow of a transaction with the *i960* is as follows. For both read and write transactions, the address latch enable (ALE) is driven by the *i960* at the FPGA to the falling edge of the clock. The address, byte enables, chip selects, and read/write (read low, write high) signals are normally driven at the FPGA pins by the *i960* at the next rising edge of the clock. At this same rising clock edge, the *i960* asserts its address/data strobe (ADS) low. Data is available to the MPI during a write at the rising clock edge of the following clock cycle. The transfer is acknowledged to the *i960* by the MPI driving a low assertion of the ready/recover (RDYRCV) signal. The same process applies to a read from the MPI except that the read data is expected at the FPGA data pins by the *i960* at the rising edge of the clock when RDYRCV is low. The MPI only drives RDYRCV low for one clock cycle. Interrupt requests are handled with the *i960* in the same way described for *PowerPC* System (see *PowerPC* System section).

Table 2. *i960*/MPI Configuration

***i960* Signal**	***ORCA* Pin Name**	**MPI I/O**	**Fuction**
AD[7:0]	D[7:0]	I/O	Multiplexed 5-bit address/8-bit data bus. The address appears in D[4:0].
ALE	RDY/RCLK/MPI_ALE	I	Address latch enable used to capture address from AD[4:0] on falling edge of clock.
$\overline{ADS}$	$\overline{RD}$/$\overline{MPI\_STRB}$	I	Address/data strobe to indicate start of transaction.
—	CS0	I	Active-low MPI select.
—	CS1	I	Active-high MPI select.
System Clock	A7/MPI_CLK	I	*i960* system clock. This clock is sourced by the system and the *i960*.
W/$\overline{R}$	A8/MPI_RW	I	Write (high)/read (low) signal.
RDYRCV	A9/$\overline{MPI\_ACK}$	O	Active-low ready/recover signal indicating acknowledgement of the transaction.
Any of $\overline{XINT}$[7:0]	A11/$\overline{MPI\_IRQ}$	O	Active-low interrupt request signal.
BE0	A0/$\overline{MPI\_BE0}$	I	Byte-enable 0 used as address bit 0 in *i960* 8-bit mode.
BE1	A1/$\overline{MPI\_BE1}$	1	Byte-enable 1 used as address bit 1 in *i960* 8-bit mode.



5-8508(F)

Figure 3. *i960*/MPI Configuration Schematic

Download via MPI

The M[2:0] pins determine the download method of the FPGA. In order to download through the MPI, the M[2:0] has to be set for the appropriate microprocessor. See Table 3 for the correct configuration mode.

Table 3. Configuration Modes

M2	M1	M0	CCLK	Configuration Mode	Data
0	1	0	Output	*Motorola** *PowerPC*	Parallel
0	1	1	Output	*Intel i960*	Parallel

The MPI handles all configuration control and handshaking with the host processor. For a single FPGA configuration, the host processor sets the configuration control register PRGM bit to zero then back to a one. Then, after reading that the INIT signal is high in the MPI status register, transfers data 8 bits at a time to the FPGA's D[7:0] input pins. If configuring multiple FPGAs through daisy-chain operation, the MP_DAISY bit must be set in the configuration control register of the MPI by the host processor. Because of the latency involved in a daisy-chain configuration, the MP_HOLD_BUS bit may be set to zero rather than one for daisy-chain operation. This allows the MPI to acknowledge the data transfer before the configuration information has been serialized and transferred on the FPGA daisy chain. The early acknowledgment frees the host processor to perform other system tasks. Configuring with the MP_HOLD_BUS bit at zero requires that the host microprocessor poll the RDY/BUSY bit of the MPI status register and/or use the MPI interrupt capability to confirm the readiness of the MPI for more configuration data. There are two options for using the host interrupt request in configuration mode. The configuration control register offers control bits to enable the interrupt on either a bit stream error or to notify the host processor when the FPGA is ready for more configuration data. The MPI status register may be used in conjunction with, or in place of, the interrupt request options. The status register contains a 2-bit field to indicate the bit stream error status. A flow chart of the MPI configuration process is shown in Figure 4. The status and configuration registers are described in more detail in the MPI Control and Status Registers section.

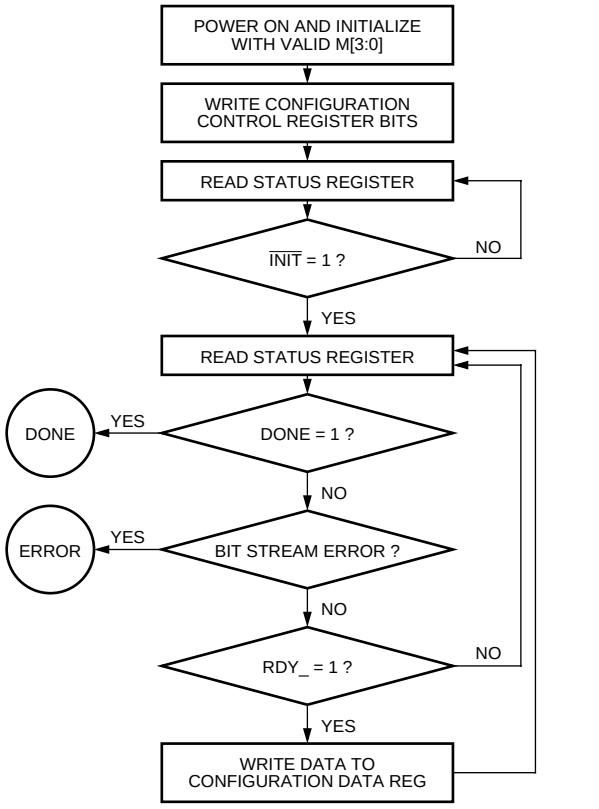


Figure 4. Configuration Through Microprocessor Interface

* *Motorola* is a registered trademark of Motorola, Inc.

Readback via MPI

Configuration readback can be performed via the MPI when it is in user mode. To put the MPI in user mode, either instantiate the MPI library macro in the HDL design code, schematic capture, or have the host processor set the MP_USER bit to 1 prior to the end of download process. The MP_USER bit, set by the host processor, ideally should be set while configuring the control registers for download. In addition to putting the MPI in user mode, the designer must instantiate the readback controller (READBK library element) in HDL design code or schematics capture. Lastly, the designer must enable readback in the download bit stream through the bit generation program of the *ORCA* Foundry control center. Under advanced bit options, the designer should activate Allow Readback and also set the type of readback under Capture Readback. Now the system is ready to perform readback. The host processor writes the 14-bit readback start address to the readback address registers and sets the RD_CFG bit to 0 in the control register. Readback data is returned 8 bits at a time to the readback data register and is valid when the DATA_RDY bit of the status register is 1. A flow chart of the MPI readback operation is shown in Figure 5. The RD_DATA pin used for dedicated FPGA readback is invalid during MPI readback.

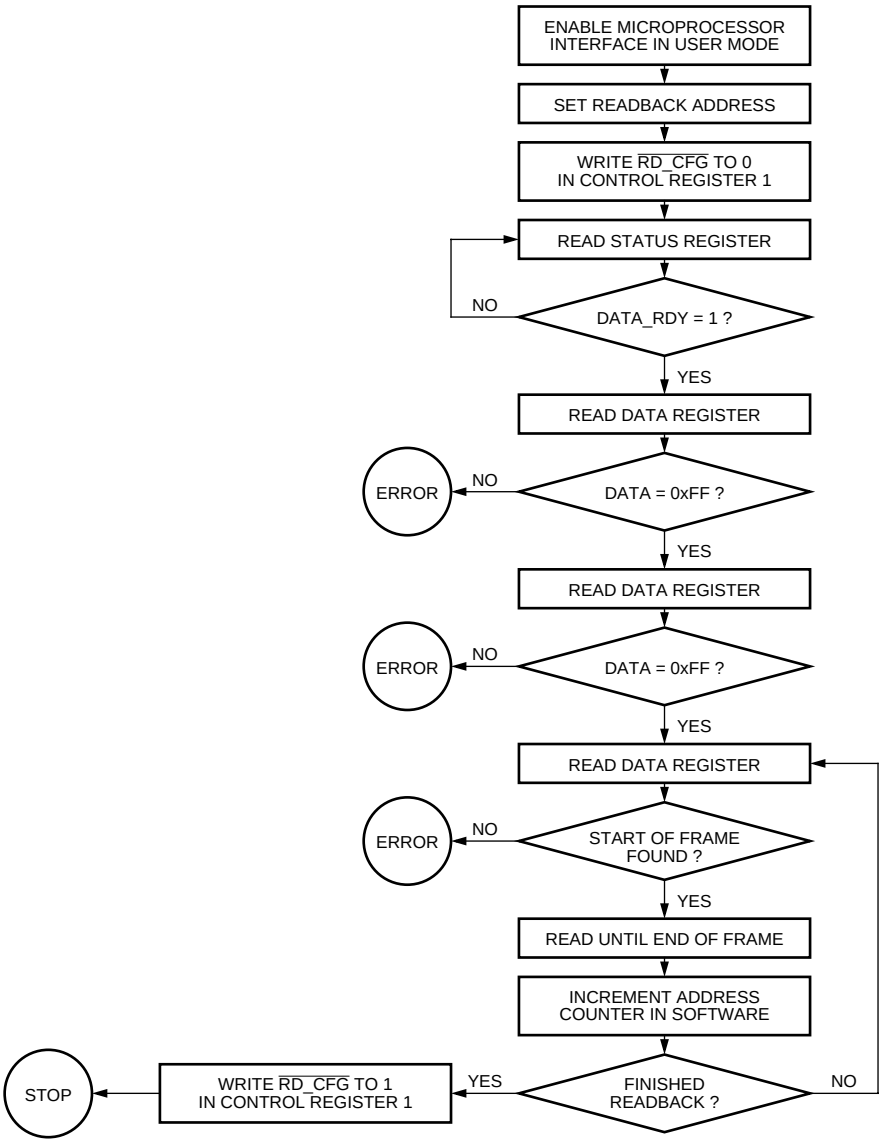


Figure 5. Readback Through Microprocessor Interface

MPI Control and Status Registers

The MPI has a series of addressable registers that provide MPI control and status, configuration and readback data transfer, FPGA device identification, and a dedicated user scratchpad register. All registers are 8 bits wide and bit 7 is the most significant bit, while bit 0 is the least significant bit. Table 4 illustrates the address map for these registers and its functionality.

The MPI control registers 1 and 2 are read/write registers. The host processor writes a control byte to configure the MPI. It is readable by the host processor to verify the status of control bits previously written.

The MPI scratchpad register is a read/write register with no defined operation. It may be used for any user-defined function.

The MPI status register is a read-only register, providing various information to the host processor. Data ready, IRQ pending, error flags, INIT, and DONE are status bit information provided to the host processor.

The configuration data register is a writable register in configuration mode and readable in readback mode. During configuration, the data register is where the host processor writes the configuration data bytes sequentially. Conversely, for readback mode, the MPI provides the readback data in the configuration data register.

The MPI readback address registers 1 and 2 are writable registers used to accept the address bytes of the configuration data location to be read back. Readback address register 1 accepts the least significant address byte (bits [7:0]), while the readback address register 2 accepts the most significant address byte (bits [15:8]).

Finally, the device ID registers are four registers holding 1 byte each. These registers provide the device ID code. This is the same as the boundary-scan ID code.

Please refer to the MPI Setup and Control subsection of the MPI section in the FPGA data book for more details of all registers.

Table 4. MPI Setup and Control Registers

Address Hex	Register
00	Control register 1.
01	Control register 2.
02	Scratchpad register.
03	Status register.
04	Configuration/readback data register.
05	Readback address register 1 (bits [7:0]).
06	Readback address register 2 (bits [15:8]).
07	Device ID register 1 (bits [7:0]).
08	Device ID register 2 (bits [15:8]).
09	Device ID register 3 (bits [23:16]).
0A	Device ID register 4 (bits [31:24]).
0B—0F	Reserved.
10—1F	User-definable address space.

MPI Interface

The user-programmable FPGA logic interfaces use a 4-bit address, read/write control signal, interrupt request signal, and user handshaking signals. Figure 6 is the *ORCA* Foundry Library macro for both the *PowerPC* and *i960*. These library macros represent the MPI hardware in the FPGA. In order to use MPI in a conventional situation, 3-state and bidirectional buffers are needed to be included. The *ORCA* Foundry *SCUBA*® macrocell generation tool automatically inserts the 3-state and bidirectional buffers along with the MPI macro. This is the recommend flow. As shown in Figures 7 and 8, the MPI module created by *SCUBA* is attached to the host processor. The signals on the left side of the MPI module communicate with the off-chip host processor (*i960* or *PowerPC*) shown in Table 5 and 6, while the signals on the right side communicate with the user-logic portion of the FPGA shown in Table 7. Table 7 explains the user-logic signals for the FPGA side. These signals are common for both the *PowerPC* and *i960*.

Table 5. PowerPC Signal Definition—External MPI Side

Signal	Direction into MPI	Function
CLK	Input	Connected to the host processor CLK to synchronize communication.
CS0N	Input	Active-low chip enable.
CS1	Input	Active-high chip enable.
RDWRN	Input	Read/Write signal Low specifies a write, high specifies a read.
TSN	Input	Active-low transfer start signal.
A[31:27]	Input	5-bit address bus.
D[7:0]	Input/Output	8-bit bidirectional data bus coming from the host processor .
TAN	Output	MPI generated open-drain, active-low transfer acknowledge.
BIN	Output	MPI generated burst inhibits signal, which informs the host that the MPI is incapable of sending more data.
IRQN	Output	MPI generated open-drain, active-low interrupt request.

Table 6. *i960* Signal Definition—External MPI Side

Signal	Direction into MPI	Function
CLK	Input	Connected to the *i960* system CLK in order to synchronous communication between MPI and host processor.
CS0N	Input	Active-low chip enable.
CS1	Input	Active-high chip enable.
WRRDN	Input	Read/Write signal. High specifies a write, low specifies a read.
ALE	Input	Address latch enable used to capture address from AD[4:0] on falling edge of clock.
ADSn	Input	Address/data strobe used to indicate start of transaction.
BE0n	Input	Byte-enable 0 used as address bit 0 in *i960* 8-bit mode.
BE1n	Input	Byte-enable 1 used as address bit 1 in *i960* 8-bit mode.
ACKn	Input/Output	Active-low ready/recover signal indicating acknowledgment of the transaction. AKA buf_rdrcvn.
AD[7:0]	Input/Output	8-bit bidirectional address and data bus.
IRQN	Output	MPI generated open-drain, active-low interrupt request.

MPI Interface (continued)

Table 7. User-logic Signal Definition—Internal FPGA Side

Signal	Direction out of MPI	Function
UA[3:0]	Output	User logic address. Addresses up to 16 unique user registers for use as control signals.
URDWRN	Output	User logic read/write control signal. High indicates a read from user logic by the host processor, low indicates a write to user-logic by the host processor.
USTART	Output	Active-high user start signal. Indicates the start of an MPI transaction between the host processor and user logic.
UEND	Input	Active-high user END signal. Indicates that the user-logic is finished with the current MPI transaction.
UIRQn	Input	Active-low interrupt. Sends interrupt request from the user-logic to the host processor.
UDOUT[7:0]	Output	User data out. Eight data bits going to user-logic*.
UDIN[7:0]	Input	User data in. Eight data bits from user-logic to host processor*.
MPI_CLK	Input	Clock generated external to the FPGA and can be used to synchronize user-logic in the FPGA.
MPI_GSR	Input	GSR signal produced by the MPI to set/reset user-logic registers.

* The figures and *SCUBA* generated code indicate that the user data is sent through the MPI. This is for modeling and simulation purposes, but is not true of the hardware. In hardware, the user data is actually routed externally to the MPI block. The model emulates the hardware functionality correctly.

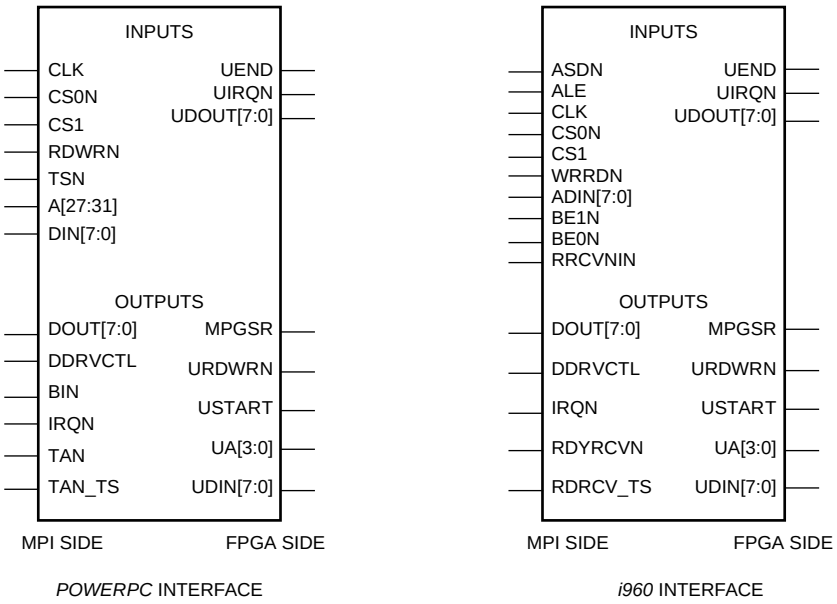


Figure 6. MPI Interface

MPI Interface (continued)

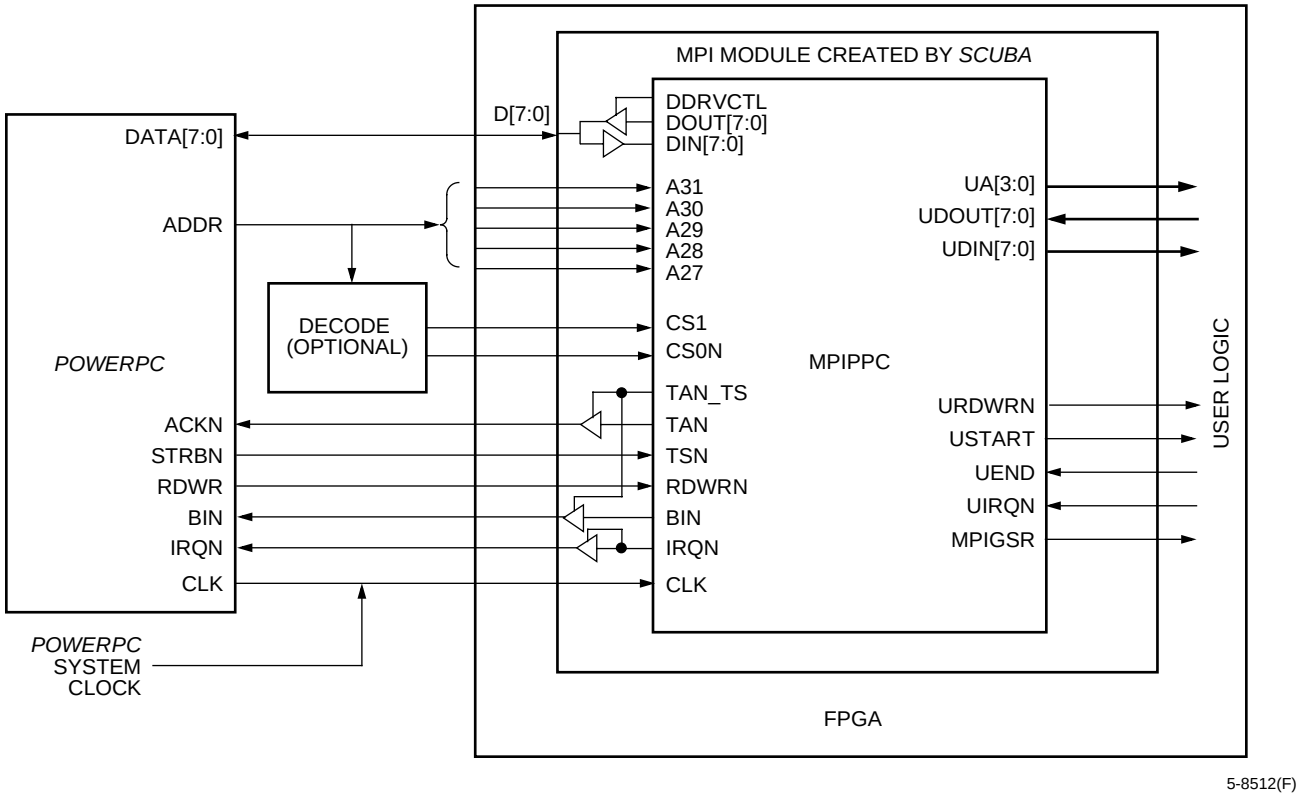


Figure 7. *PowerPC* MPI Interface

MPI Interface (continued)

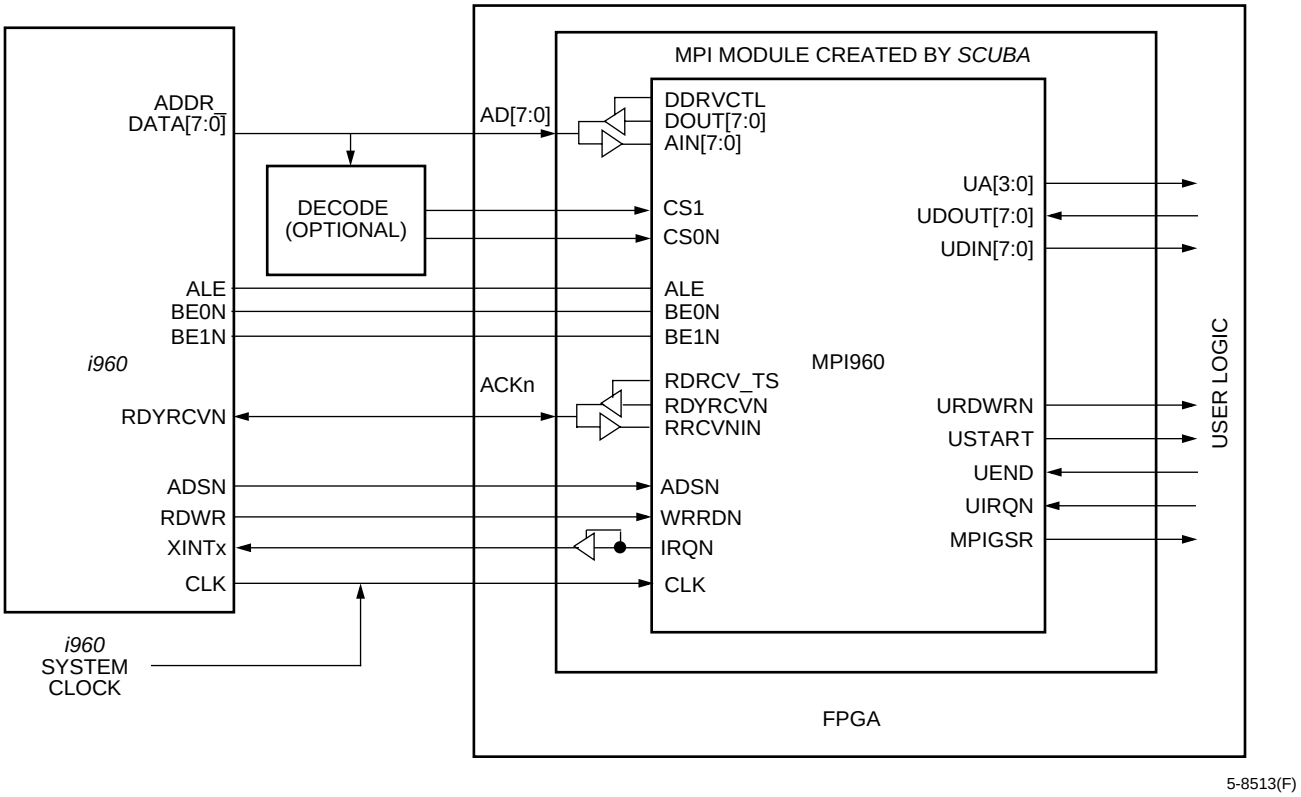


Figure 8. *i960* MPI Interface

Basic User-Logic Communication and Transactions

After the host processor initiates a transaction as discussed in the *PowerPC* System and *i960* System sections, the MPI outputs the 4-bit user address (UA[3:0]) and the read/write control signal (URDWR, which is read-high, write-low regardless of host processor). The MPI then asserts the user start signal, USTART. During a write from the host processor, the user logic can accept data written by the host processor from the D[7:0] pins once the USTART signal is asserted. The user logic ends a transaction by asserting an active-high user end (UEND) signal to the MPI. If the UEND signal is never returned after a USTART, the microprocessor bus will typically hang. Thus the guaranteed return of UEND is a potential issue. To avoid this issue, a watchdog timer circuit may have to be added to the FPGA logic

The MPI will insert wait-states in the host processor bus cycles, holding the host processor until the user-logic completes its task and returns a UEND signal, upon which the MPI generates an acknowledge signal. If the host processor is reading from the FPGA, the user logic must have the read data available on the D[7:0] pins of the FPGA when the UEND signal is asserted. If the user logic is fast or if the MPI user address is being decoded for use as a control signal, the MPI transaction time can be minimized by routing the USTART signal directly to the UEND input of the MPI.

The user-logic may also assert an active-low interrupt request (UIRQ) to the MPI, which, in turn, asserts an interrupt to the host processor. Assertion of an interrupt request is asynchronous to the host processor clock and any read or write transaction occurring in the MPI. The user-logic is responsible for providing any required interrupt vectors for the host processor, and the user-logic must deassert the interrupt request once serviced. If the interrupt request is not deasserted in the user logic, it will continue to be asserted to the host processor. See the *PowerPC* System section of this application note for more information.

Extending Address and Data

Although the MPI includes 16 internal address register locations and 16 programmable user-logic locations (external to the MPI, in the FPGA), a design may need more non-MPI specific locations. At the same time, the host processor may be larger than 8 data bits, therefore needing a larger data bus. With a few modifications, a designer will be able to create larger address and data buses.

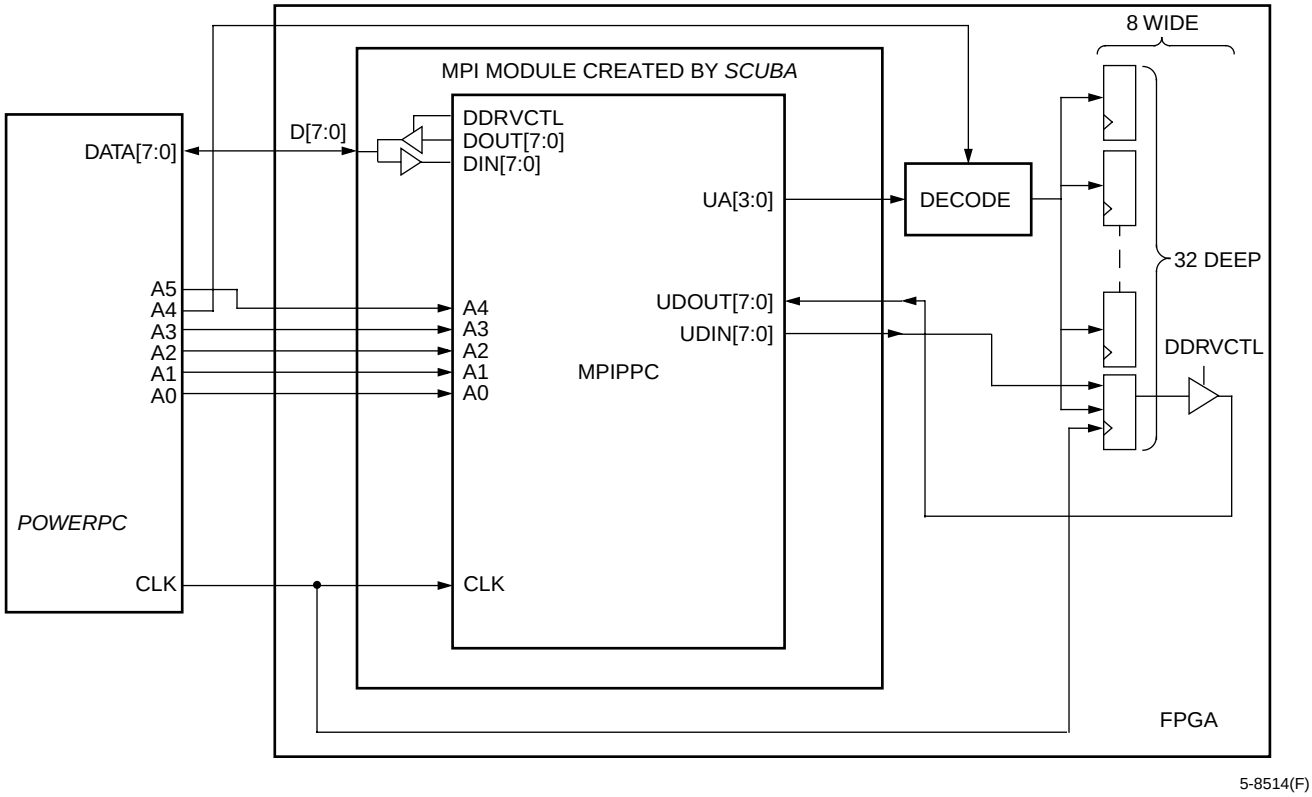
By default, the MPI uses address lines A[4:0] to decode the address locations. Signal A[4] determines which set of FPGA address locations will be accessed, internal or external of the MPI block. If A[4] is a 0, the internal 16 registers are used, if A[4] is a 1, the external 16 locations are used. The remaining address signals A[3:0], determine which of the 16 locations are to be used, internal or external. The MPI passes A[3:0] through to the FPGA as signals UA[3:0]. The designer should then take UA[3:0] and decode them to resolve the appropriate user location. The USTART signal determines when the data is ready, as described previously.

In order to extend the number of external address locations, the designer must add additional address lines. Figure 9, demonstrates how this may be done. In order to increase the number of MPI-external address locations to 32, you must take an additional address bit from the host processor and bring it into the FPGA through a normal I/O buffer. The additional address bit should then be used to decode 32 address locations.

As seen in Figure 9, the host processor's sixth address bit (A5) is used to determine if the address is internal to the MPI or external. This sixth bit should be connected to A4 of the MPI address bits. The A4 address bit of the host processor is the bit that should be routed externally to the MPI block created by *SCUBA*. Setting up the address bits in this fashion organizes the location map such that the lower locations are internal to the MPI and the upper locations are external. Figure 10 shows the logic mapping. When A5 is 0, the address is located internally; likewise, when A5 is 1, the address is located externally to the MPI. Furthermore, when A4 is 0, it is the bottom half of the locations, and when A4 is 1, it is the top half of the address locations. Notice that locations 15 to 31 are simply mapped to the same registers as address 0 to 15. This is due to the fact that the address extension is only external and thus creates a 16-word repeated internal address mapping. If this address space is needed, further decodes for the chip select pins are required to disable this device for all but one of the repeated address blocks.

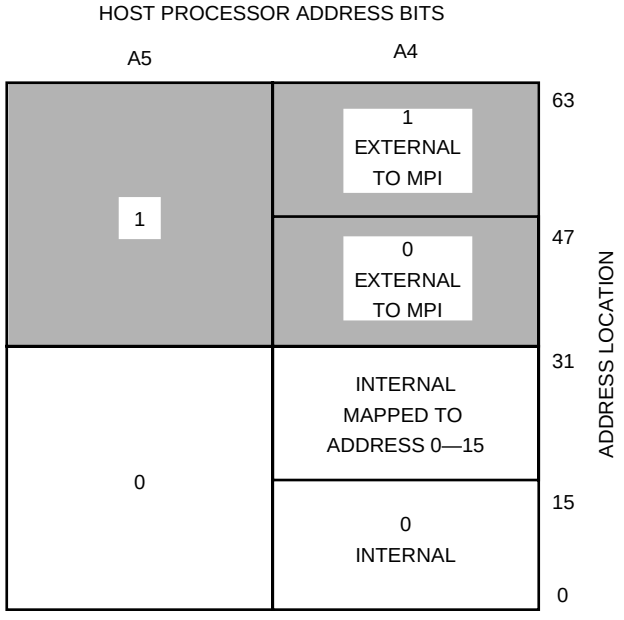
Extending the number of data bits is fairly easy. In Figure 11, the data bits D[15:8] from the host processor are brought into the FPGA through bidirectional I/O buffers and set up exactly the same as the other UDOUT and UDIN buses. The DDRVCTL signal is used to control the bidirectional I/O buffers. In order to use this signal you will need to slightly modify the *SCUBA* module by adding the DDRVCTL to the port list and connecting the DDRVCTL signal to the port in the port map.

Extending Address and Data (continued)



5-8514(F)

Figure 9. Extending Number of External Address Locations to 32



5-8517(F)

Figure 10. Address Location Mapping

Extending Address and Data (continued)

Note that when increasing the number of data bits, the internal data bits registers still remain 8 bits, only the external data width can be increased.

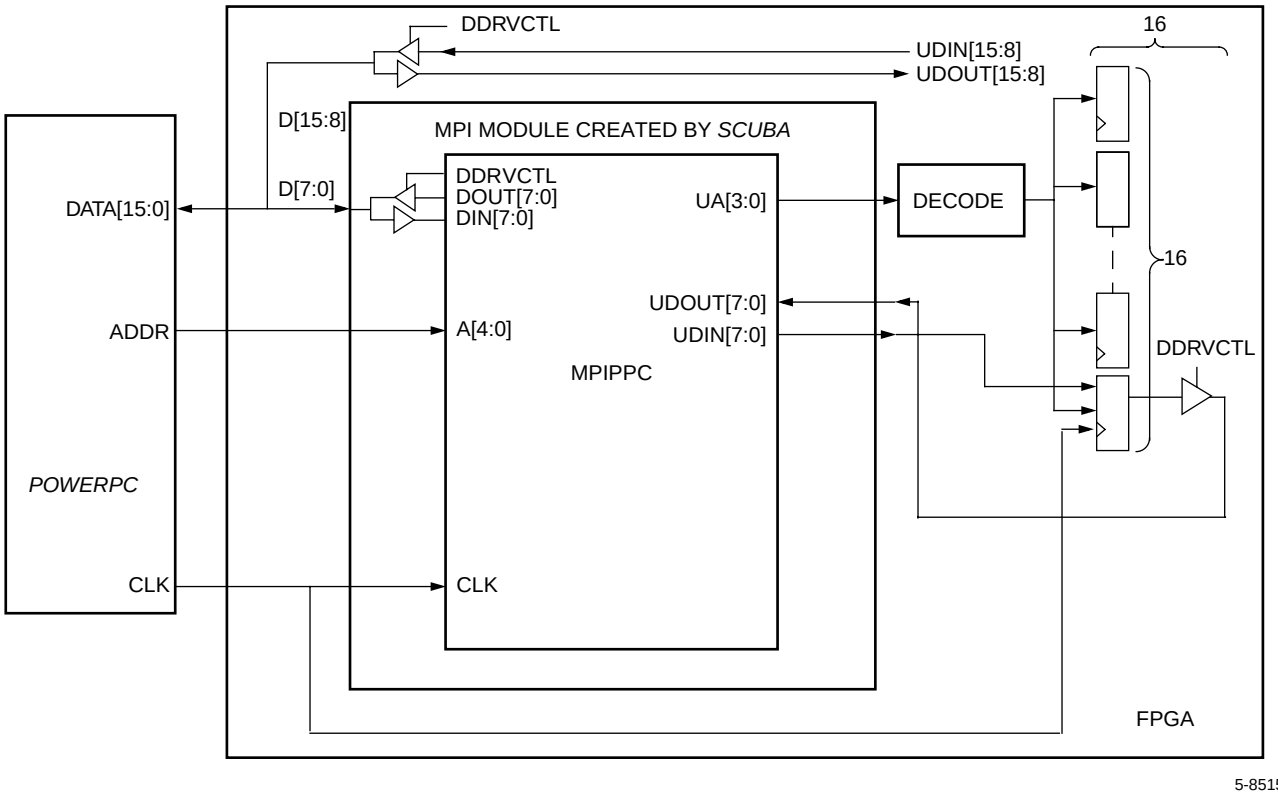


Figure 11. Extending Number of External Data Bits to 16

Use of Other Microprocessors

The *PowerPC* and *i960* are not the only host processors that may be hooked up to the MPI. The example shown in Figure 12 shows the FPGA hooked up to a digital signal processor. Although this setup appears seamless, many design considerations need to be taken.

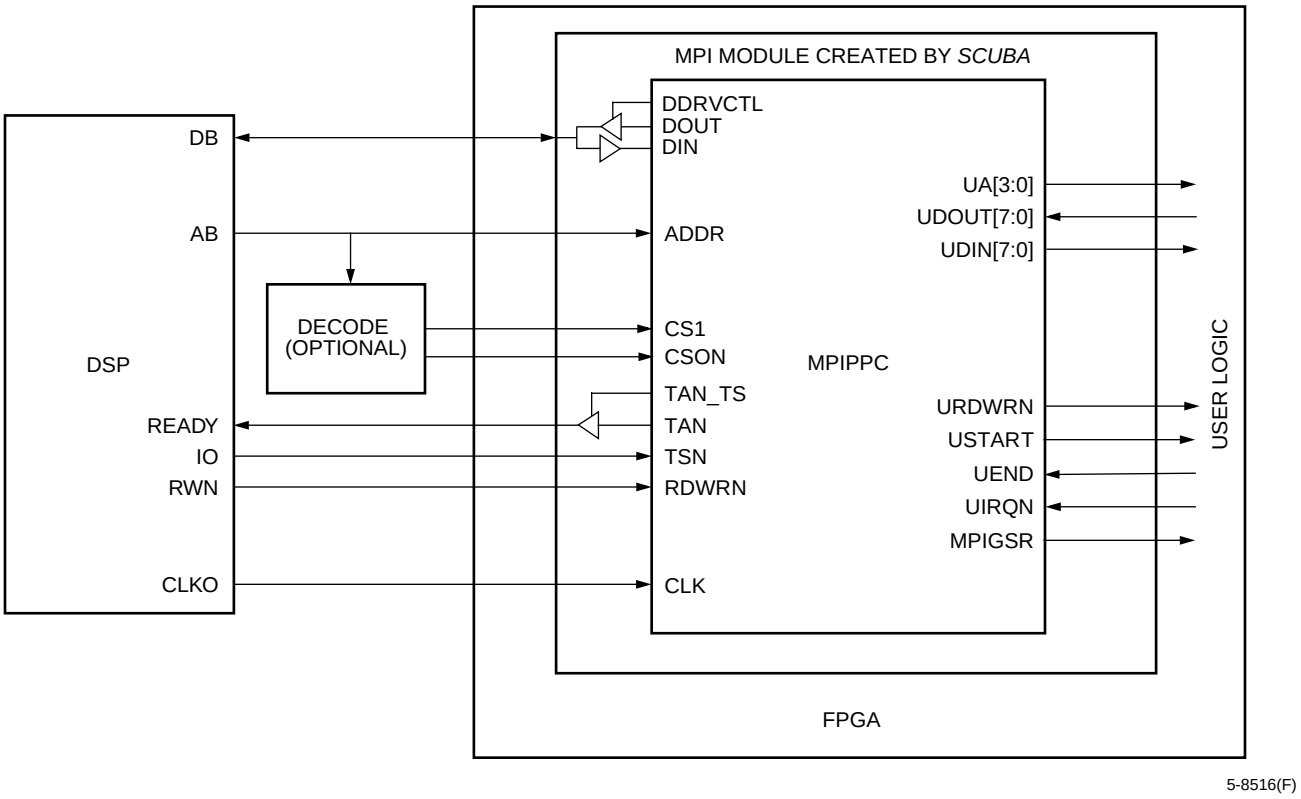


Figure 12. DSP MPI Interface

Conclusion

The microprocessor Interface feature reduces the glue-logic requirement and conserves programmable logic. As prevously discussed , the microprocessor interface (MPI) can be used for configuration, readback, general-purpose interface to the FPGA, as well as other basic device control and status functions. With a few small modifications, the 8-bit, 32 address locations (internal and external), can be easily increased to accommodate any designer's needs. For more detailed information, visit our website: **http://www.latticesemi.com**

www.latticesemi.com

January 2002
AP99-050FPGA

Lattice® Semiconductor Corporation