Product Summary

Intended Use
- Interfacing to external serial RAM, EEPROM, FLASH
- Interchip communication
- Serial communication between processors and controllers

Key Features
- AMBA (APB) compliant interface
- 32 bytes fifo read and write data
- Interrupts and status register
- Transmission speed can be up to 1/8 of the system clock
- NOP character handling for half duplex traffic
- Needs 3 clocks (system clock, SPI clock and SPI chip select)
- Evaluation platform available

Targeted Devices
- Axcelerator Family
- ProASIC®PLUS Family

General Description
The Serial Peripheral Interface (MX-ACT-SPI_F) from Memec Design is SPI compatible with an AMBA (APB) compliant interface. It offers serial high-speed communication and access to SPI peripheral parts like EEPROMs. Master and slave functionality is integrated in the core. Interrupts and status registers are used for communication control. Data is buffered through a 32-byte FIFO, and data traffic can be half- or full-duplex. Automatic NOP character generation and checking is integrated, to ensure easy protocol handling. The VHDL implementation allows the SPI to be integrated into FPGA parts. For SPI communication, two additional clock domains guarantee SPI timing. The evaluation platform and the C-utility library allow users to immediately code and test software.
Core Deliverables

- Netlist Version
  - Netlist compatible with the Actel Designer place and route tool
- RTL Version
  - VHDL Source Code
  - Test Bench
- All
  - User Guide

Synthesis and Simulation Support

- Synthesis: Synplicity
- Simulation: ModelSim
- Other tools supported upon request

Verification

- Test Bench

Figure 1: Logic Symbol
Functional Description

Overview
The Serial Peripheral Interface MC-ACT-SPI_F is used as an interface between a processor system (e.g. ARM subsystem) and external device(s) which use SPI communication.

This core is composed with several registers allowing an easy to complex data transfers. They are configured through the APB bus.

On the left side, the MC-ACT-SPI_F is connected to the APB bus, or a similar bus interface. To take advantage of the interrupt signals, they have to be connected directly to the interrupt sources of an interrupt controller.

The right side is the SPI interface and goes out of the system.

The SPI module contains both master and slave unit which can be used as one or the other. Two 32 byte deep FIFOs are included for buffering, one for RX and one for TX.

FIFO
Both FIFOs have an interrupt line. The RX FIFO activates the ‘data available’ interrupt as long as at least 1 byte can be read from the RX FIFO. The TX FIFO issues an interrupt when it is half empty or when it is empty, depending on the configuration of the MC-ACT-SPI_F. When it is set to ‘empty’, this interrupt may serve as ‘message sent’ interrupt (if the message fits within the FIFO). This interrupt is active right after reset. Also a status bit is provided to mark the TX FIFO full situation. This is needed to avoid a TX FIFO overflow caused by too many writes from the processor.
Serial Peripheral Interface

Master Mode

Basically all transfers are initiated by an SPI master which also provides the clock for the transfer plus the CS signal to address an SPI slave. The clock polarity can be configured.

The master starts shifting out the data from the TX FIFO serially as soon as it will be filled with data. At the same time the master samples the RX data at the receive pin. Data transfer takes place as long as there is data in the TX FIFO, otherwise it is stopped.

The SPI master works with 8 times oversampling. The baudrate can be programmed by means of register SPI_CFG. The speed range is dependent on the system clock; it goes from 190bits/s up to 12.5Mbit/s at 100MHz clock.

The status of the SPI master can be read. It is either idle or busy, which means that the master is still transceiving data. This is needed to control the CS line of the slave.

Slave Mode

A slave is clocked directly from SCK. It starts receiving with CS_n = 0, which initializes the controller. The SCK may be at 0 or 1, the first active edge (depending on the clock polarity) is the one which samples the incoming data, the other edge is ignored. At the same time the slave is shifting out the TX data. When the CS_n is set to 1, the activity is terminated, regardless of the SCK and data pins. A partially transmitted character will be ignored. An empty TX FIFO will produce NOP characters (which can be configured), otherwise data from the TX FIFO is sent.

NOP Character

A NOP character checking is integrated to use the SPI for half duplex traffic. If the first received byte is a NOP character, all RX data is discarded for the whole transmission, if configured so. In slave mode a transmission starts at the falling edge of CS and ends at the rising edge of CS. In master mode a transmission starts with writing data into the TX buffer and ends with deactivate master CS output in the SPI_TX_CTRL. To guarantee a start of a transmission the master CS output has to be deactivated first.
Device Requirements

<table>
<thead>
<tr>
<th>Family</th>
<th>Device</th>
<th>Utilization</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProASIC&lt;sup&gt;PLUS&lt;/sup&gt;</td>
<td>APA075-STD</td>
<td>n/a</td>
<td>2</td>
</tr>
<tr>
<td>Axcelerator</td>
<td>AX500-3</td>
<td>291 (6%)</td>
<td>143 (6%)</td>
</tr>
</tbody>
</table>

Table 1: Device Utilization and Performance

Verification and Compliance

Functional and timing simulation has been performed on the SPI_F using VHDL and Verilog Test Benches. Simulation vectors used for verification are provided with the core. This core has also been used successfully in customer designs.

Signal Descriptions

The following signal descriptions define the IO signals.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset_n</td>
<td>in</td>
<td>asynchronous system reset, active low</td>
</tr>
<tr>
<td>pclk</td>
<td>in</td>
<td>APB clock and clock for the SPI</td>
</tr>
<tr>
<td>paddr[31:0]</td>
<td>in</td>
<td>APB Address input, bits [3:0] used only</td>
</tr>
<tr>
<td>pwdata[31:0]</td>
<td>in</td>
<td>APB Write data</td>
</tr>
<tr>
<td>psel</td>
<td>in</td>
<td>APB Select signal</td>
</tr>
<tr>
<td>penable</td>
<td>in</td>
<td>APB Enable signal</td>
</tr>
<tr>
<td>pwrite</td>
<td>in</td>
<td>APB Write signal</td>
</tr>
<tr>
<td>pdata[31:0]</td>
<td>out</td>
<td>APB Read data</td>
</tr>
<tr>
<td>int_spi_tx</td>
<td>out</td>
<td>The SPI TX interrupt line</td>
</tr>
<tr>
<td>int_spi_rx</td>
<td>out</td>
<td>The SPI RX interrupt line</td>
</tr>
<tr>
<td>spi_rxd</td>
<td>in</td>
<td>Pin for the incoming bit stream</td>
</tr>
<tr>
<td>spi_txd</td>
<td>out</td>
<td>Pin for the outgoing bit stream</td>
</tr>
<tr>
<td>spi_txd_drv_n</td>
<td>out</td>
<td>Enable signal to drive the tristate-buffer to the spi_txd</td>
</tr>
<tr>
<td>spi_sck_in</td>
<td>in</td>
<td>SPI clock input</td>
</tr>
<tr>
<td>spi_sck_out</td>
<td>out</td>
<td>SPI clock output</td>
</tr>
<tr>
<td>spi_sck_drv_n</td>
<td>out</td>
<td>Enable signal to drive the bidir-buffer for the spi_sck_out</td>
</tr>
<tr>
<td>spi_cs_in_n</td>
<td>in</td>
<td>SPI chip select input</td>
</tr>
<tr>
<td>spi_cs_out_n</td>
<td>out</td>
<td>SPI chip select output</td>
</tr>
<tr>
<td>spi_cs_drv_n</td>
<td>out</td>
<td>Enable signal to drive the bidir-buffer for the spi_cs_out_n</td>
</tr>
</tbody>
</table>

Table 2: Core I/O Signals

Timing

Figure 3: Transmit Timing
Recommended Design Experience

For the source version, users should be familiar with HDL entry and Actel design flows. Users should be familiar with Actel Libero v2.2 Integrated Design Environment (IDE) and preferably with Synplify and ModelSim.

Ordering Information

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC-ACT-SPI_F-NET</td>
<td>Core Netlist</td>
</tr>
<tr>
<td>MC-ACT-SPI_F-VHD</td>
<td>Core VHDL</td>
</tr>
</tbody>
</table>

Table 3: Core Part Numbers

The CORE is provided under license from Memec Design for use in Actel programmable logic devices. Please contact Memec Design for pricing and more information.

Information furnished by Memec Design is believed to be accurate and reliable. Memec Design reserves the right to change specifications detailed in this data sheet at any time without notice, in order to improve reliability, function or design, and assumes no responsibility for any errors within this document. Memec Design does not make any commitment to update this information.

Memec Design assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction, if such be made, nor does the Company assume responsibility for the functioning of undescribed features or parameters. Memec Design will not assume any liability for the accuracy or correctness of any support or assistance provided to a user.

Memec Design does not represent that products described herein are free from patent infringement or from any other third-party right. No license is granted by implication or otherwise under any patent or patent rights of Memec Design.

Memec Core products are not intended for use in life support appliances, devices, or systems. Use of a Memec Core product in such application without the written consent of the appropriate Memec Design officer is prohibited.

All trademarks, registered trademarks, or service marks are property of their respective owners.

Datasheet Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datasheet 1.0</td>
<td>December 20, 2002</td>
<td>Initial Release</td>
</tr>
<tr>
<td>Datasheet 1.1</td>
<td>January 23, 2003</td>
<td>Modification done in section core deliverables and signal descriptions; Added logo to footer</td>
</tr>
<tr>
<td>Datasheet 1.2</td>
<td>February 25, 2003</td>
<td>Modification done in section device requirements, new URL and address inserted</td>
</tr>
</tbody>
</table>