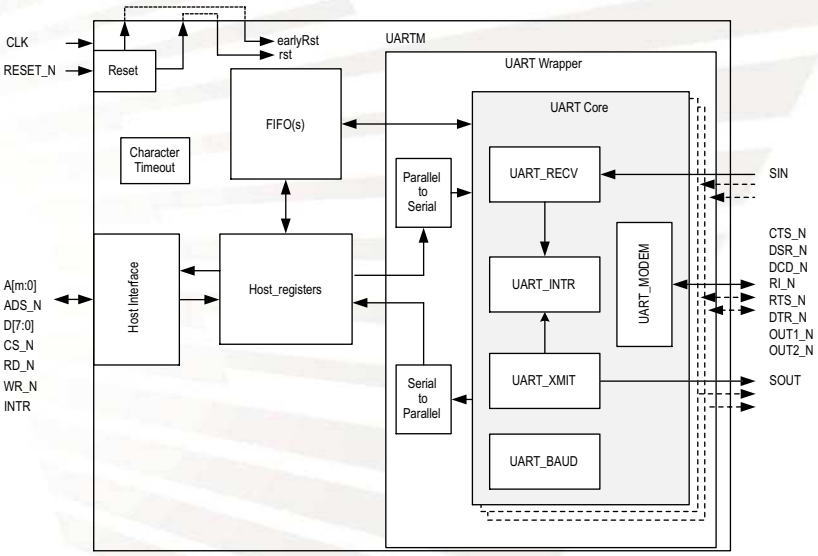
AvnetCore: Datasheet

Version 1.0, July 2006

Multi-Channel UART Controller



Block Diagram

TThe Multichannel UART (Universal Asynchronous Receiver/Transmitter) is a FPGA core that implements up to 16 UARTs in a single Actel device. These UARTs are completely independent in functionality, but share common logic to reduce its overall size as compared to individual instantiations. Each UART channel is similar to the industry standard 16550 device and is an upward solution to standard UARTs by providing FIFOs in both the transmit and the receive paths.

Intended Use:

—

Features:

- Function similar to industry standard 16550
- Configurable number of channels of 4, 8 or 16
- Configurable FIFO depths
- Channel baud rates to 115 K Baud
- Inserts or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data
- Line break and detection

Targeted Devices:

- ProASIC®3
- ProASIC PLUS® Family

Core Deliverables:

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

Synthesis and Simulation Support:

- Synthesis: Synplicity®
- Simulation: Model*Sim*®
- Other tools supported upon request

Verification:

- Test Bench
- Test Cases

Actel CompanionCore

AVNET Memec

General Description

Each channel performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to serial conversion on data characters received from the CPU. The CPU can read the complete status of each channel at any time. Reported status information includes the type and condition of the transfer operations being performed, as well as any error conditions (parity, overrun, framing, or break interrupt). Synchronization for the serial data stream is accomplished by adding start and stops bits to the transmit data to form a data character (character orientated protocol). An optional parity bit can be attached to the data character to enhance data integrity. The receiver checks the parity bit for any transmission bit errors. The channels of the Multichannel UART (UARTM) are designed to reduce CPU overhead when working with highspeed modems and other devices, because the core can buffer several bytes and burst them to the CPU instead of generating a costly interrupt cycle on every byte. Therefore, the interrupt overhead can be amortized over several bytes, thus increasing performance. The core was designed to be compact by time-slicing a single UART engine instead of instantiating multiple instances. The time-slicing technique allows for the core to be smaller and more space efficient. Each channel has its own baud rate generator, interrupt controller and prioritizer, receive and transmit FIFOs, and CPU registers. The user has control over the configuration of the core by modifying the parameters in the top-level source file. This allows the core to be modified and reused easily. These parameters include number of channels and FIFO depth. The core comes with a testbench to aid the customer with integration and verification.

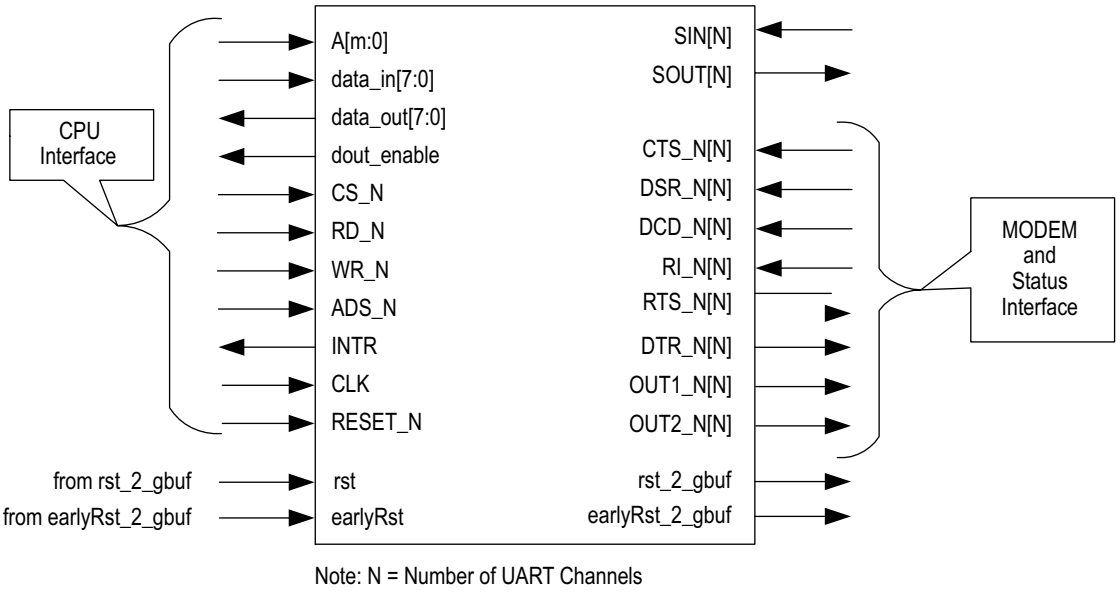


Figure 1: MC-ACT-UARTM Logic Symbol

Functional Description

The 16550 UARTM is includes a time-division multiplexed (time-sliced) version of the MC-ACT-UART core. The microprocessor interface is similar to the standard 16550 UART interface with the possible inclusion of up to 4 additional address bits to select up to 16 channels. Integration of the core into existing designs that are replacing an array of discrete chips is easy since each channel is mapped to the standard 16550 register set using the lower three address bits.

TIME-SLICED ARCHITECTURE

The core is time multiplexed across NUM_CHANNELS serial input lines, where NUM_CHANNELS is the number of channels and is either 4, 8 or 16. It is set by the user when instantiating the core. This means that instead of duplicating the core logic NUM_CHANNELS times, the logic is shared between all channels. This also means that the logic must retain the state of each channel after each time slice is finished. The state machine and sampling register values of a given channel are stored in memory or in registers so that they can be reloaded when the time multiplexing returns to the channel NUM_CHANNELS time slices later. This technique reduces the size of the core.

The time slice period is a function of three constants – the highest baud rate in the system, 16 samples per serial bit, and the number of channels in the system, NUM_CHANNELS. The time slice period determines what the system clock should be. For example, a system clock of 29.4912 MHz is needed for a 16-channel system with all channels running at the maximum baud rate of 115,200 bps. (115,200 bps x 16 samples per bit x 16 channels = 29.4912 MHz).

Signal	Width	Direction	Description
a[m:0]	m+1	Input	Address Bus. The Address input width is 8 for a 16-channel, 7 for an 8-channel, and 6 for a 4-channel implementation. The value of "m" is 7, 6, and 5 respectively. Address bits [m-1:3] are used to select the port. Bits [2:0] are used to select the registers for a given channel. Address bit m is reserved for future use and must be tied to logic low.
data_in[7:0]	8	Input	Data Bus input.
data_out[7:0]	8	Output	Data Bus output.
dout_enable	1	Output	Data Bus direction control signal. This signal is high, active, during the time data_out[7:0] should be driven to the external data bus.
ads_n	1	Input	Address Strobe When low, allows the value of a[7:0] and cs_n to propagate into the design. A low to high transistion latches the contents of a[7:0 and cs_n.
cs_n	1	Input	Chip Select. When low, this signal indicates that the microprocessor has selected this device for an access operation.
rd_n	1	Input	Read Strobe. When rd transitions low, the contents of the register selected by A[2:0] on the channel selected by A[6:3] will be driven on the data bus D[7:0].
wr_n	1	Input	Write Strobe. When wr transitions from high to low, the levels of the data bus D[7:0] will be latched to the register selected by A[2:0] on the channel selected by A[6:3].
intrq	1	Output	Interrupt Request. This signal is high (active) when any channel has an interrupt available for the microprocessor. An assertion on intrq indicates that the microprocessor should check the status registers on the sourcing channel.
clk	1	Input	Synchronous clocking source. This clock should be driven at a rate to insure reliable bit timing on the serial interfaces and at a rate high enough to insure timely host interface A/C timing. A clock rate of of 29.4768 Mhz is the recommend frequency.
reset_n	1	Input	Master Reset (async). This low-active reset signal must conform to the setup time requirements specified in the data sheet for the device used.
sin	num_channels	Input	Receive Data. Receive Serial Data from the UART channel.
sout	num_channels	Output	Transmit Data. Transmit Serial Data to the UART channel.
cts_n	num_channels	Input	Clear to Send. When low, this signal indicates that the data set on the UART channel is ready to receive data. Bit 4 of the MSR register reflects the condition of this line, while bit 0, when high, indicates a change in the condition of this line.
dsr_n	num_channels	Input	Data Set Ready. When low, this signal indicates that the data set is powered on. Bit 5 of the MSR register reflects the condition of this line. Bit 1 of the MSR indicates a change in the condition of this signal when high.
dcd_n	num_channels	Input	Data Carrier Detect. When low, this signal indicates that the data set has detected a sinusoidal carrier on the channel it is monitoring. Bit 7 of the MSR register reflects the condition of this line, while bit 3, when high, indicates a change in the condition of this signal.
ri_n	num_channels	Input	Ring Indicator. When low, this signal indicates that the Data Set (a modem) has detected ringing voltage from the telephone line. Bit 6 of the MSR register reflects the condition of this line, while bit 2, when high, indicates a change in the condition of this signal.
rts_n	num_channels	Output	Request to Send. This signal, when driven low, indicates to the data set that the data terminal has data in the transmit buffer that is ready for transmission. This signal can be asserted by setting bit 1 of the MCR register.

Signal	Width	Direction	Description
dtr_n	num_channels	Output	Data Terminal Ready. This signal, when driven low, indicates to the data set that the data terminal is powered up. This signal can be asserted by setting bit 0 of the MCR register.
out1_n	num_channels	Output	Each channel has 2 general-purpose output pins.
data_out[7:0]	num_channels	Output	Each channel has 2 general-purpose output pins.
rst	1	Input	Buffered processed active high reset signal (should come from global buffer output)
earlyRst	1	Input	Buffered processed active high early reset signal (should come from global buffer output)
rst_2_gbuf	1	Output	Processed active high reset signal (should feed input to global buffer)
earlyRst_2_gbuf	1	Output	Processed active high early reset signal (should feed input to global buffer)

Table 1: Top Level Module Signal Descriptions

UARTM
This is the top level of the core. Its main purpose is to serve as a container to instantiate the modules shown in the block diagram.

UART_WRAPPER
The DDR Interface (ddr_interface) module is responsible for maintaining the bi-directional ddr_data bus, and for asserting all address and command signals to the SDRAM. For a write operation, this module reads from the larger sys_data bus and, using the 2x clock and muxes, constructs the DDR data bus, writing out a new value on every rising edge of the 2x clock which is strobed into the DDR SDRAM with ddr_dqs. For read operations, the opposite must occur. The Data Path reads in the DDR data using sys_clk_fb rising edge as the time reference, and de-muxes the data into two separate 1x clock pipelines. These two 1x clock pipelines are then concatenated to form the larger sys_data bus, which is provided to the user.

Characteristics of this time-sliced UART core with portions of the design which are not time-sliced. To accomplish this interfacing, two modules are provided. A Parallel to Serial module, named muart_p2s, which receives parallel input (all channels simultaneously) and serializes these terms at the appropriate channel time slots to the UART core. A Serial to Parallel module, named muart_s2p, which receives the time-sliced output of the UART core, and updates the appropriate channel state in due time.

UART core
The UART core is a net-list processed standard 8250 UART. This UART consists of 5 blocks, Receive, Transmit, Interrupt, Baud Rate, and Modem which are briefly described below.

UART_RECV
Receiver. This block filters the serial input data (SIN), detects the start bit, controls the sampling of SIN, determines when a complete character is shifted into the receive shift register, and stores the received character into the receive FIFO. When the correct number of bytes have been stored in the FIFO as set by bits 6 and 7 of the FIFO control register (FCR), an interrupt is sent to the microprocessor which fetches the byte from the receive holding register (RHR) for the channel that caused the interrupt. Parity, framing, and overrun errors are detected and their corresponding bits are set in the line status register (LSR). All operations in this block are synchronous to the system clock CLK and enabled with the receiver clock enable, RX_CE. RX_CE must occur at 16x the expected serial bit rate.

UART_XMIT
Transmitter. This block accepts 8-bit parallel data, stores it onto the FIFO, retrieves it from the FIFO, serializes it, appends start, stop, and parity bits as needed, and shifts this data out on SOUT. This block generates an interrupt when the external FIFO is empty. All operations in this block are synchronous to CLK and enabled with the transmitter clock enable, TX_CE, except for writing data to the THR which is synchronous to CLK and enabled with the THR_CE. TX_CE must occur at 16x the desired serial bit rate.

UART_MODEM
Modem Control and Status Logic. This block provides status of the modem input lines for each channel, both current status and change of state. The modem control lines are also generated here. An interrupt is generated on the low to high transition of RI_L or any change of state of CTS_L, DCD_L, and DSR_L. All operations in this block are synchronous to CLK.

UART_INTR
Interrupt Logic. For each channel, this block prioritizes the four interrupt sources and encodes them into a 3-bit Internal version of the Interrupt Status Register value, ISR[2:0], and sends an internal interrupt request output to the HOST_REGISTERS module which adds FIFO interrupts to the internal version of ISr[2:0]. A 4-bit Interrupt Enable Register input, IER[3:0], allows the user to individually mask each interrupt input. Any active interrupt source that has its corresponding IER bit set, will cause IRQ to be asserted. In order to clear the interrupt, the action listed in the following table must be performed. In order to determine the exact cause of an interrupt, the microprocessor should read the ISR. If IIR[0] is low, there is a pending interrupt and the source is determined by decoding the Interrupt Status Bits, ISR [2:1]. For a Receiver Line Status Interrupt, the LSR must be read to further determine what type of error or errors have caused the assertion of INTRPT. If LSR[7] is set, there is either a parity error, framing error, or break indication associated with one of the characters in the Receive FIFO. LSR[4:2] will always reflect the error status for the character at the top of the FIFO. If the Receive FIFO is empty, LSR[4:2] will be all zeroes.

UART_BAUD

The UART_BAUD block provides each channel with a divide-by-N clock enable based on the CLK input, where N is the 16-bit value of that channel's Divisor Latch Register (DLR). This sampling rate represents a frequency that is 16x the desired baud rate for the channel. This oversampling allows the Transmitter and Receiver Control blocks to properly detect start and stop bits, and to utilize samples during the middle of the bit transmission period to guarantee proper framing of data. Like the Transmitter and Receiver Control blocks, the UART_BAUD is time multiplexed across the NUM_CHANNELS channels. The logic is shared between all channels in the same manner as with the other blocks. The state of the UART_BAUD block is stored in memory for each time slice so that it can be restored when the time multiplexing returns to the channel NUM_CHANNELS time slices later.

The UART_BAUD block uses a counter to delay the toggling of the clock enable by the appropriate number of CLK cycles, as determined by the Divisor Latch Register for that channel. If the user modifies the contents of the Divisor Latch Register, the new value will be loaded into the counter on the next successive time slice and the clock enable signal will reflect the new sampling frequency.

With an input clock to the Core of 1.8423 MHz times the number of Channels, the recommended Divisor Latch settings are provided below for common baud rates.

Divisor Latch	BAUD RATE
2304	50
1536	75
1047	110
857	134.5
768	150
384	300
192	600
96	1200
48	2400
32	3600
24	4800
16	7200
12	9600
6	19.2K
3	38.4K
2	57.6K
1	115.2K

Table 2: Divisor Latch values for a Core clock of NUM_CHANNELS * 1.8423 MHz

FIFO

The FIFO module is comprised of the RX_FIFO, the TX_FIFO, and the FIFO_CTRL modules.

RX_FIFO

The Receiver FIFO module consists of NUM_CHANNELS independent configurable depth FIFOs. Each FIFO is 11-bits wide to support 8-bit characters plus LSR[4:2] for that symbol.

When the correct number of bytes have been stored in the FIFO as set by bits 6 and 7 of the FIFO control register (FCR), the Data Ready bit is set for that channel in the Line Status Register, LSR[0] and an interrupt is sent to the microprocessor which fetches the byte from the receive holding register (RHR) for the channel that caused the interrupt. Parity, framing, and overrun errors are detected and their corresponding bits are set in the line status register (LSR).

TX_FIFO

The Transmitter FIFO module consists of NUM_CHANNELS independent configurable depth FIFOs. Each FIFO is 8-bits wide to support 8-bit characters.

HOST_REGISTERS

This block implements the bulk of the functionality of the 16550 host interface for all channels. It also contains all the host accessible registers for the design. This block manages the Interrupt priority and subsequent Interrupt generation for all channels.

HOST_INTF

This block provides support for an asynchronous microprocessor interface, which is used to read and write the register set, and load or unload the FIFOs. The registers form the bridge between the asynchronous interface, and the all logic that is synchronous to the system clock. They also allow the microprocessor to use the upper bits of the address lines to select a particular time-slice in register space. The entire address bus, then, selects a particular register in a particular time-slice or channel.

RESET

The reset module is responsible for processing the reset signal into two global resets for the remaining logic. These two reset signals activate asynchronously from the RST_N input, but will deactivate synchronous to the clock. One of these reset outputs "rst_2_gbuf" will extend the duration of the reset by a multiple of the number of channels clock periods, to allow internal sequencing thru memory structures to initialize state to the required reset values. These two outputs, "rst_2_gbuf" and "earlyRst_2_gbuf" are driven out of the core. These two outputs should be connected two global buffers and fed back to the core into inputs "rst" and "earlyRst" respectively. If multiple identical Cores are instantiated into a single device, only one core's RESET module should be used.

MEMORY MAP & REGISTERS

Each of the channels has 8 registers. The NUM_CHANNELS parameter controls the number of channels actually present. Accesses to unused channels may alias.

Address A[m:0]	Contents	R/W
	Channel 0 registers	
0x00 LCR[7] = 0 LCR[7] = 0 LCR[7] = 1	RHR: Receive Holding Register THR: Transmit Holding Register BDL: Baud Divisor ILOW	R W R/W
01 LCR[7] = 0 LCR[7] = 1	IER: Interrupt Enable Register BDH: Baud Divisor High	R/W R/W
02	IIR: Interrupt Identificiation Register FCR: FIFO Control Register	R W
03	LCR: Line Control Register	R/W
04	MCR: Modem Control Register	R/W
05	LSR: Line Status Register	R
06	MSR: Modem Status Register	R
07	SPR: Scratchpad Register	R/W
08 - 0F	Channel 1 registers	
10 - 17	Channel 2 registers	
18 - 1F	Channel 3 registers	
20 - 27	Channel 4 registers	
28 - 2F	Channel 5 registers	
30 - 37	Channel 6 registers	
38 - 3F	Channel 7 registers	
40 - 47	Channel 8 registers	
48 - 4F	Channel 9 registers	
50 - 57	Channel 10 registers	
58 - 5F	Channel 11 registers	
60 - 67	Channel 12 registers	
68 - 6F	Channel 13 registers	
70 - 77	Channel 14 registers	
78 - 7F	Channel 15 registers	

Device Requirements

Family	Channels	Device	Utilization		Performance
			Tiles	RAMS	
ProASIC^{PLUS}	16	APA300	95%	16	34 MHz STD
ProASIC^{PLUS}	8	APA150	83%	12	36 MHz STD
ProASIC^{PLUS}	4	APA075	97%	12	39 MHz STD
ProASIC3	16	A3P400	64%	10	54/41 MHz -2/STD
ProASIC3	8	A3P400	43%	7	61/45 MHz -2/STD
ProASIC3	4	A3P400	26%	7	60/45 MHz -2/STD
ProASIC3	8	A3P250	65%	7	61/45 MHz -2/STD
ProASIC3	4	A3P125	77%	7	60/44 MHz -2/STD

Table 3: Device Utilization and Performance

Verification and Compliance

Functional and timing simulation has been performed on the MultiChannel UART using a self-checking Verilog Test Bench with Verilog test cases.

Timing

Read and Write cycles have a recovery time due to time slicing latency. The recovery time is (3 * NUM_CHANNELS+3)*t_{CY} , where t_{CY} is the cycle time of the input clock. However, if a FIFO Read immediately follows a FIFO Read, the recovery time is 4*t_{CY}. If a FIFO Write immediately follows a FIFO Write, the recovery time is 4*t_{CY}.

The timing diagrams below reflect ads_n held active, or low.

Since this IP is a core, the effects of part speed grade, family, routing variations, and effects of user logic in placement and routing will have varying degrees of impact on the timing. The user must take these things in consideration to insure a successful implementation. The Timing information provided in this document, will address timing requirements based upon architectural aspects of this IP only.

READ TIMING

The figure below shows the setup times required for a successful read. It also shows the expected delay between the assertion of the rd_n signal, and valid data on the bus. The data remains valid as long as A, cs_n, and rd_n are held in their active states.

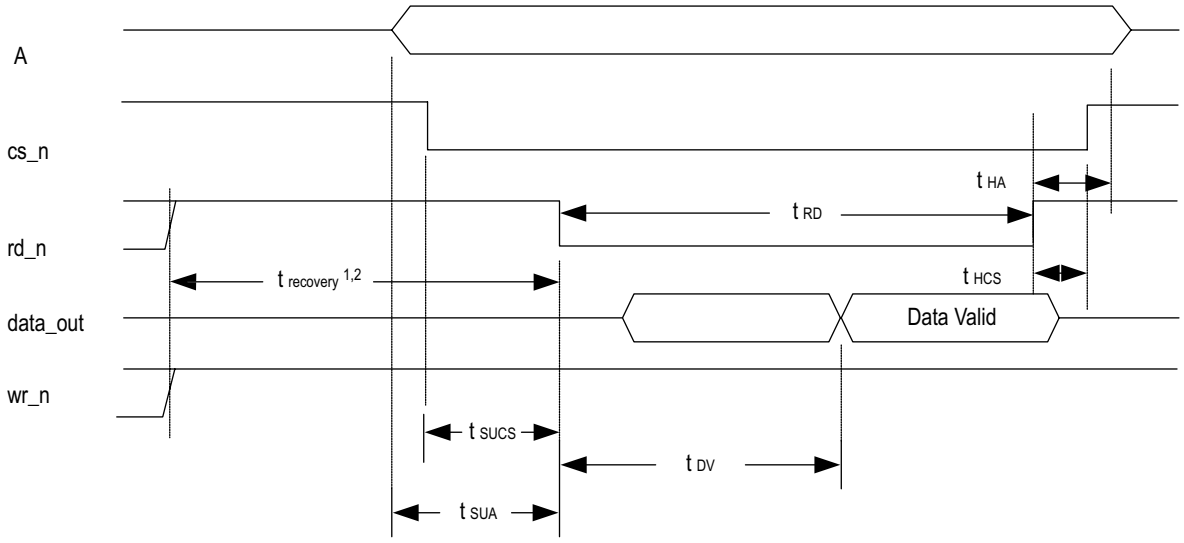


Figure 3: Asynchronous Timing For Reads

	Description	Min	Max
t_{SUCS}	Setup time for cs_n with respect to rd_n	t_{CY}	
t_{SUA}	Setup time for A with repect to rd_n	t_{CY}	
t_{HCS}	Hold time for CS with respect to rd_n	t_{CY}	
t_{HA}	Hold time for A with respect to rd_n	NUM_CHANNELS*t_{CY}	
t_{DV}[3]	Delay between rd_n assertion (going low) and valid data on the bus		2*t_{CY}
t_{RD}[3]	rd_n strobe width	2*t_{CY}	
$t_{recovery}$[1,2]	Delay from previous rd_n or wr_n rising edge to the falling edge of rd_n	(3* NUM_CHANNELS+3)*t_{CY}	

Table 4: Read Timing Descriptions

Where:

- NUM_CHANNELS is the number of ports selected of the implementation
- t_{CY} is the period of the system clock
- [1] A Read FIFO followed by a Read FIFO of the same channel requires a recovery time (trecovery) of a minimum of 4*t_{CY} .
- [2] A Write MCR followed by a Read MSR with MCR[4] (Loop) equal 1, requires a recovery time of (8 * NUM_CHANNELS+3)* t_{CY}.
- [3] The rd_n input signal is sampled a minimum of 2 times by the clock. If the host interface is not synchronous to the rising edge of the clock, increase the width of the rd_n signal accordingly.

WRITE TIMING

The figure below shows the setup and hold times required for a successful write. Write data is captured with the rising edge of wr_n and is written to the appropriate location within the UARTM within the recovery time. Read or write accesses during the recovery time is not permitted. The recovery time is relaxed during back to back write cycles to the TX FIFO.

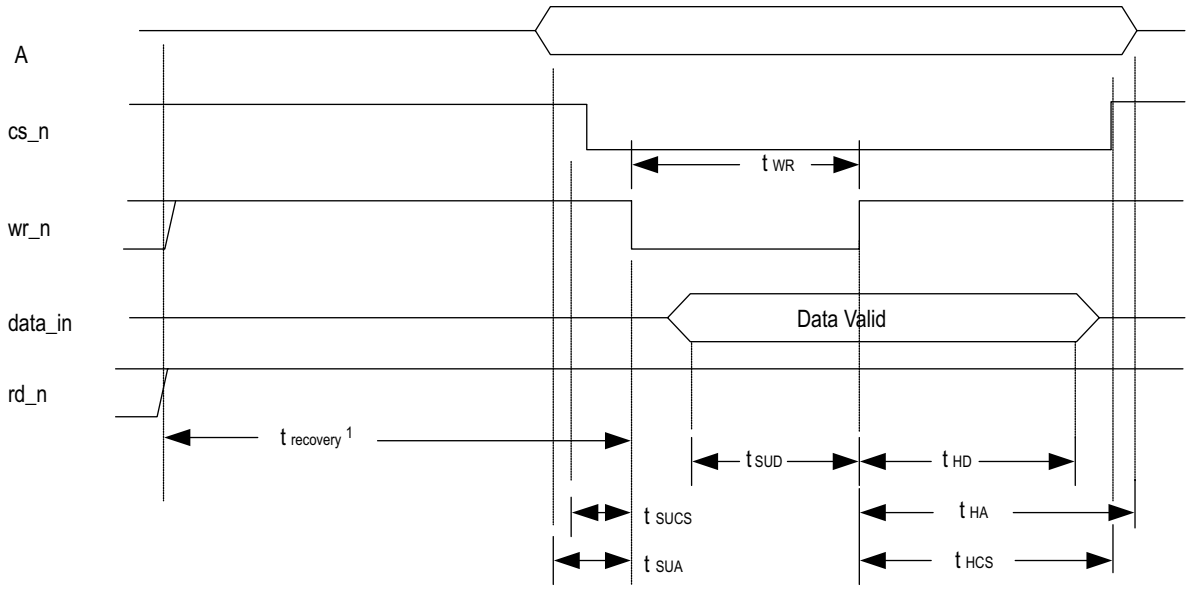


Figure 4: Asynchronous Timing For Writes

	Description	Min	Max
t_{WR}	wr_n strobe width	$2*t_{CY}$	
t_{SUD}	Setup time for D with respect to rising edge of wr_n	t_{CY}	
t_{SUCS}	Setup time for cs_n with respect to falling edge of wr_n	t_{CY}	
t_{SUA}	Setup time for A with respect to falling edge of wr_n	t_{CY}	
t_{HD}	Hold time for data_in with respect to rising edge of wr_n	t_{CY}	
t_{HA}	Hold time for A with respect to rising edge of wr_n	NUM_CHANNELS*t_{CY}	
t_{HCS}	Hold time for cs_n with respect to rising edge of wr_n	t_{CY}	
$t_{recovery}$[1]	Delay from previous rd_n or wr_n rising edge to the falling edge of wr_n	(3*NUM_CHANNELS+3)*t_{CY}	

Table 4: Write Timing Descriptions

Where:

- t_{CY} is the period of the system clock
- [1] A Write FIFO followed by a Write FIFO of the same channel requires a recovery time (trecovery) of a minimum of 4*t_{CY}.

RESET TIMING

After initiating a reset to the core, allow a minimum of (8 * NUM_CHANNELS+3)*t_{CY} after the trailing edge of reset prior to using the device. This delay is needed for the synchronous clearing of state in each of the channels. The reset pulse with should be greater than t_{CY} wide.

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 Integrated Design Environment (IDE) and preferably with Synplify and ModelSim. Users should also have experience with microprocessor systems and asynchronous communication controllers.

Ordering Information

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

Contact Information:

North America
10805 Rancho Bernardo Road
Suite 100
San Diego, California 92127
United States of America
TEL: +1 858 385 7500
FAX: +1 858 385 7770

Europe, Middle East & Africa
Mattenstrasse 6a
CH-2555 Brügg BE
Switzerland
TEL: +41 0 32 374 32 00
FAX: +41 0 32 374 32 01

Ordering Information:

Part Number	Hardware	Resale
MC-ACT-UARTM-NET	Actel MultiChannel UART Netlist	Contact for pricing
MC-ACT-UARTM-VHDL	Actel MultiChannel UART VHDL	Contact for pricing

Actel CompanionCore

AVNET Memec

www.em.avnet.com/actel