



Macrocad Development Inc. VCM1016 Synthesizable UART Model

Description:

The VCM 1016 model from Macrocad is a synthesizable behavior HDL for creating a 16550 compatible UART function. Implementation is made easy for both FPGAs and ASICs. Synchronous design and small module size assures worry free synthesis. Well commented code provides insight into operations. Bi-directional signals are contained in the buffer (shell) level, the core contains unidirectional signals only. Several clocking method are supported, including 1, (synchronous), 2, or 3 clock schemes.

Features:

- ? Synthesizable RTL HDL code
- ? Modular design provides flexibility
- ? Synchronous design
- ? Well commented code for clarity
- ? Test Bench is included
- ? Receive and transmit buffers are expandable
- ? Compatible with 16550 UART
- ? Dual port SRAM, or edge triggered register array used for buffer functions
- ? Available in Verilog and VHDL versions
- ? Approximately 6k gates (asic gates)
- ? Designed by hardware engineers for hardware engineers

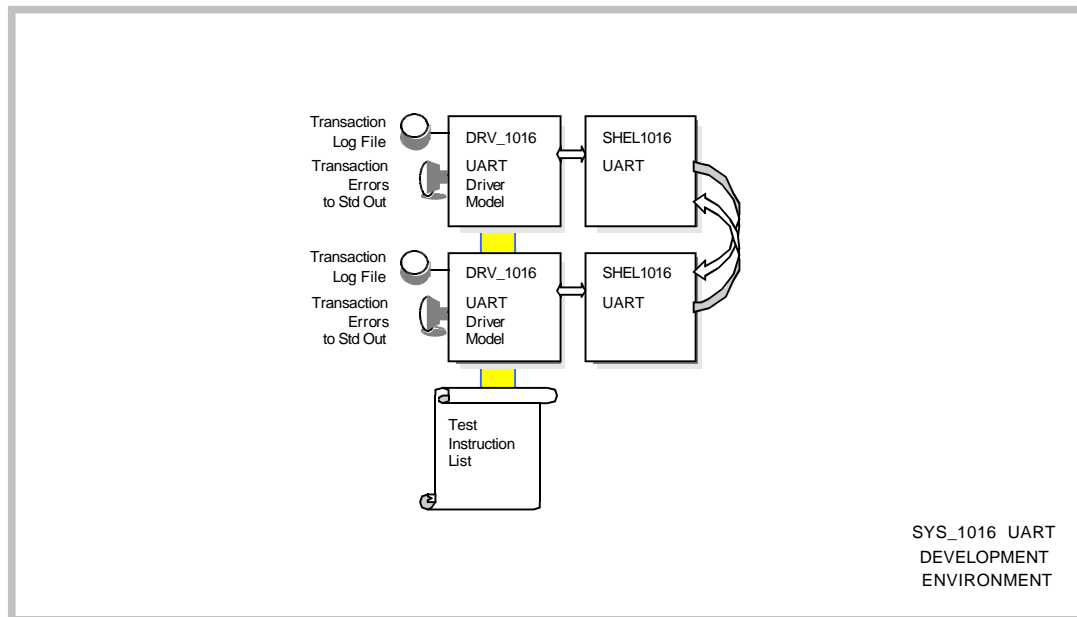


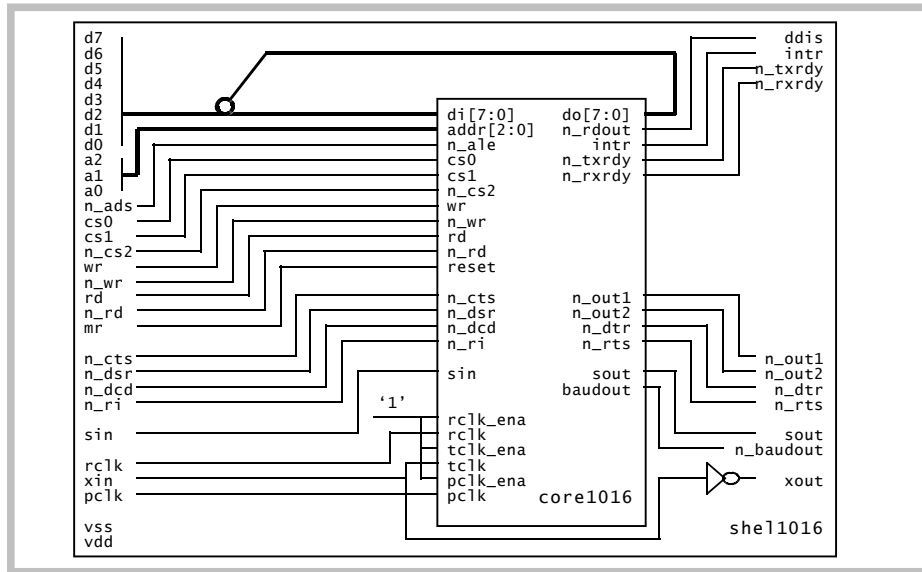
Figure 1

Figure 1 shows the development environment for the UART bus.

Architecture

Macrocad's VCM 1016 is a synthesizable 16550 compatible behavior model. This model provides the system designer with a drop in UART function. This model is functionally partitioned to provide the most flexibility for various ASIC and FPGA implementations. It includes two 16 byte buffer memories for

improved serial communications. The IO buffer shell level contains the only bi-directional signals in the model. The functional units are all contained in the core level. The core function model also contains debug features which display internal states during serial data transactions.



Core and Shell Pins

Figure 2

Pin Description

Shell	Core	Type	Description
d7		Bi-direct	Programming Data bit 7
d6		Bi-direct	Programming Data bit 6
d5		Bi-direct	Programming Data bit 5
d4		Bi-direct	Programming Data bit 4
d3		Bi-direct	Programming Data bit 3
d2		Bi-direct	Programming Data bit 2
d1		Bi-direct	Programming Data bit 1
d0		Bi-direct	Programming Data bit 0
	do[7:0]	Out	Programming Data bus output
	di[7:0]	In	Programming Data bus input
ddis	n_r dout	Out	read enable for data bus <not>
	addr[2:0]	In	Programming register selects (address bus)
a2		In	Programming register selects (address bit 2)
a1		In	Programming register selects (address bit 1)
a0		In	Programming register selects (address bit 0)
n_ads	n_ale	In	Address latch enable <not>
cs0	cs0	In	Chip select 0
cs1	cs1	In	Chip select 1
n_cs2	n_cs2	In	Chip select 2 <not>
rd	rd	In	Programming read enable
n_rd	n_rd	In	Programming read enable
wr	wr	In	Programming write enable <not>
n_wr	n_wr	In	Programming write enable <not>
intr	intr	Out	Interrupt
n_txr dy	n_txr dy	Out	Transmit buffer available <not>
n_r xr dy	n_r xr dy	Out	Receive data available <not>
mr	reset	In	Reset
pclk	pclk	In	Programming clock <master clock>
	pclk_ena	In	Programming clock enable
xin	tclk	In	Transmit baud generator clock
	tclk_ena	In	Transmit baud generator clock enable
xout		Out	Reflects XIN input <not>
rclk	rclk	In	Receive clock
	rclk_ena	In	Receive clock enable
n_cts	n_cts	In	Clear to send - modem <not>
n_dcd	n_dcd	In	Data carrier detect - modem <not>
n_dsr	n_dsr	In	Data set ready - modem <not>
n_ri	n_ri	In	Ring indicator <not>
n_dtr	n_dtr	Out	Data terminal ready <not>
n_out1	n_out1	Out	Programmable output 1 <not>
n_out2	n_out2	Out	Programmable output 2 <not>
n_rtsb	n_rts	Out	Request to send <not>
sin	sin	In	Serial receive data in
sout	sout	Out	Serial data stream transmit output
	baud_eq1	Out	Baud rate equals 1 indicator
n_baudout	baudout	Out	Baud rate <x 16> output

table 1

Description

The VCM1016 is a UART model and test bench for implementing a UART function compatible with the 16550 UART from National Semiconductor. It is partitioned into

several functional sub-sections. These are the PRG_ITF, REG_SET, TXT_ENG, RCV_ENG, and BAD_GEN modules.

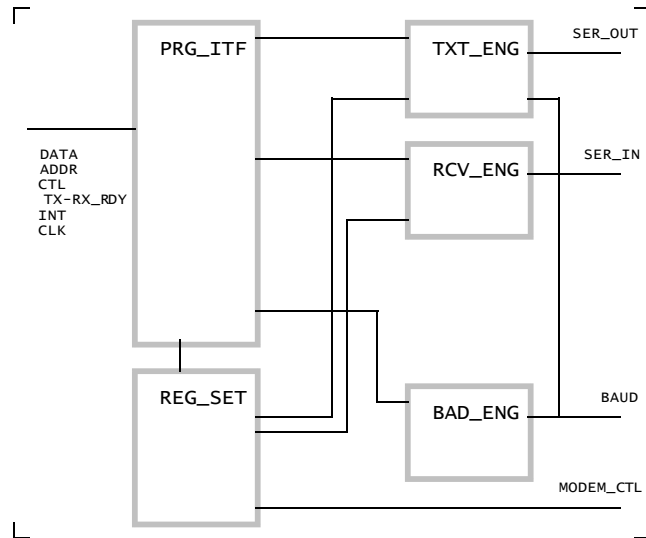


figure 3

Functional Blocks

The VCM1016 UART is partitioned into a core logic model and a shell model. The shell model contains bi-directional buffers and some clock and configuration logic only. The core model contains only unidirectional signals. All shell level output and bi-direct pins can be tri-stated with the TRIN signal. (This is a test feature, and is currently not pinned out.) The VCM 1016 synthesizable core is broken into several functional sub-sections. These are the PRG_ITF, REG_SET, TXT_ENG, RCV_ENG, and BAD_GEN modules.

The PRG_ITF module handles the data transactions between the internal registers, and the programming interface. The incoming control strobes, address selects, and data are all

synchronized in this module to PCLK, the programming clock.

The REG_SET is the register set model for the line and modem, control and status registers. The logic for the modem control and the scratch register is in this module also. The scratch register, SCR, is a register which has no function in the UART, other than a storage location. The REG_SET also generates the interrupt for the program interface.

The BAD_GEN is the baud rate generation engine. It provides the baud rate clock for transmit and receive operations. It is a programmable periodic counter.

The TXT_ENG and RCV_ENG are the transmit and receive engines for serial communications. These modules contain the memory buffers used for multiple data transfers. The memory is accessed just like most static memory. There is a pointer to the location, and controls for reading

and writing the data. Two signals are provided for interfacing with DMA operations which indicate the availability of buffer locations for transmitting data, and the amount of data which has been received, (N_TXRDY and N_RXRDY).

PRG_ITF : Program interface module

This module handles the interface to the parallel IO programming pins. It synchronizes data and provides synchronous register write enables. The UART registers are inputs to this module. They are multiplexed into the output data bus for reading.

The CS1, CS0, N_CS2 and A[2:0] inputs are stored when N_ALE is de-asserted, (high) and flow through when asserted, (low). The WR (asserted active high) and N_WR (asserted active low) inputs are or'ed together; when either is asserted a write operation is selected. The RD (asserted active high) and N_RD (asserted active low) inputs are or'ed together; when either is asserted a read operation is selected.

This module converts the CS1, CS0, N_CS2, N_ALE, WR, N_WR, RD, N_RD, and ADDR[2:0] inputs into the appropriate strobes and gate signals to write and read the CORE1016 registers.

NOTE: The logic does not prohibit the simultaneous selection of both a read and write cycle.

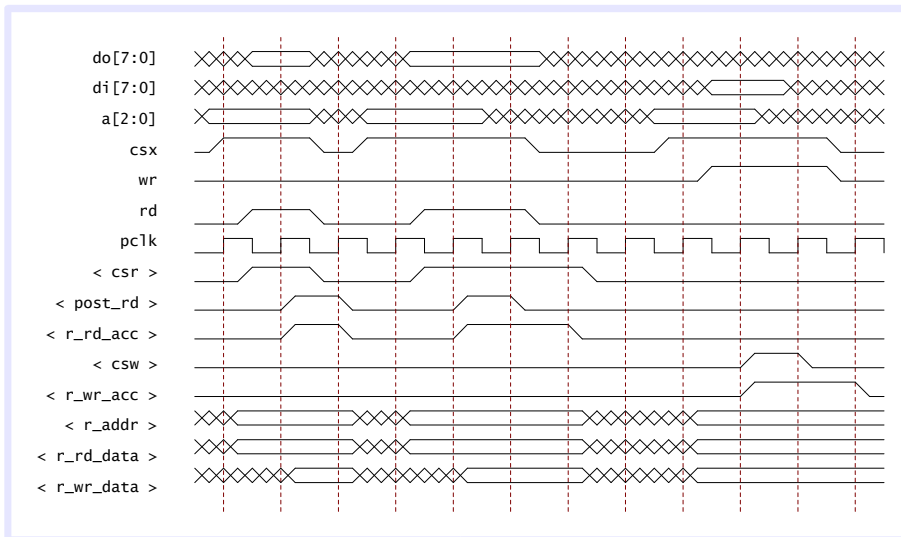


figure 4

Figure 4 shows register access timing. The signals in angle brackets, < >, are internal signals to the core. The signals CSW and CSR are qualified internal write and read signals. The UART will store the data output on a read. This stored read data is used to reset some status bits, and will be gated with the appropriate posted read signal. The posted read signals are for updating register contents after the read has

happened. The CSW signal is used to write the stored input data into the selected register.

The programming interface to the UART can be considered synchronous or asynchronous. If the read strobes are very short in duration, as short as 1 pclk, then the interface should be considered synchronous and set up an hold times should be observed as needed due to

synthesis and layout results. If the programming interface has strobes which are longer in duration, in excess of 2 pclk, then synchronization will occur in the UART for the interface signals. The minimum duration for a UART cycle will be 3 clocks. In this case, the address will be required to be stable before the first pclk of the cycle. Then on the next pclk, the chip select and read or write strobes will be required to be stable prior to the rising edge of pclk. The last pclk of the cycle is a recovery time between cycles.

If the programming interface has more than 1 pclk for the address phase, and more than 1 pclk for the strobe phase, then these signals can be considered asynchronous, since they are always clocked prior to use in the UART. In the case of longer strobes, the address phase can be used for the previous cycle's recovery phase.

In general, all the register access is done by signals which are synchronized to pclk by the UART.

REG_SET : Register set module

The REG_SET (register set) module is comprised of the Line Control, Line Status, Modem Control, Modem Status, Interrupt Enable, Interrupt ID, Scratch, Baud Rate, and Buffer Control Registers.

This module contains the registers which control serial, interrupt, and modem operations, and provide status to the parallel programming interface. The Line Control and Status, Modem Control and Status, Scratch, Baud Rate Low,

Baud Rate High, Interrupt ID, Interrupt Enable, and Buffer Control registers are contained in this module. The interrupt control circuits contained in this module generate an interrupt (INTR signal pin). Interrupts can result from the modem, transmit, and receive sources. This module prioritizes interrupts and generates them depending on the interrupt enable register. It will also clear the interrupts if the appropriate clear procedure for the active interrupt is executed.

TXT_ENG : Transmit engine module

This module contains the serial transmit engine. It includes the data path, and transmit stack (buffer), serial conversion, parity generation and the state machine for serial transmission of data. This module generates the pointers to the transmit buffer memory, and generates the buffer full and empty signals. It serializes the data contained in the transmit holding register. A counter is used to divide the baud clock by 16 to control the rate of the serial bit stream. This is done by pacing the serial transmit state machine, (statem), which tracks the bit positions of the serial data stream which is being transmitted. A loop back feature is included for diagnostics.

The statem, (serial state machine tracking model), controls the transmission of serial data. This includes all of the appropriate low level protocol such as start bits, stop bits, parity bits, character length, etc.

The module has dual serial outputs to enable the loop-back mode of operation. When in loop-back mode the SOUT module output will be held at a logic 1 and the transmit serial output data stream gets diverted to the TXDO input replacing the SIN data stream in the receiver.

RCV_ENG : Receive engine module

The receive module receives data from the serial interface, and presents it as a byte for reading by the programming interface. It includes the state machine, (statem) and serial to parallel

conversion as well as parity checking. Odd and even parity checks are supported, and are selectable in the Line Control Register. This module also checks for framing errors, and

break indications. It samples and synchronizes the incoming serial data stream. It reports stack (buffer memory) overages. It generates the pointers for the stack. It also checks for timeout conditions on the serial bit stream, when in DMA mode. In this case, the data received may not have enough characters to reach the DMA trigger level, (see for[7:6]). The timeout will set

the trigger anyway to avoid having characters sitting in the receive buffer, but not being read. The statem module controls the serial to parallel conversion. It tracks the serial protocol, which is dependent on the line control register. Once a character has been received, the receive data holding register contents are written to the current buffer location.

BAD_GEN : Baud generation module

This module generates the baud rate clock for the serial interface. There is a 16 bit modulo binary counter contained in this module. This counter is loaded from a 16 bit baud rate generation register. These are accessed as two byte registers. Writing either baud rate register will reload the counter to the baud rate register values. The counter is clocked by the TCLK which is program interface clock. TCLK_ENA enables the counter to operate, and the counter runs at this rate.

The BAD_GEN (baud rate generation) module contains the divisor registers and clock divider network for the BAUDOUT output. The BAUDOUT frequency will be the TCLK clock

rate divide by the value in the divisor registers. The BAUDOUT signal determines the transmitted serial rate of data transmission. Data will be transmitted at a rate of one bit for every 16 BAUDOUT cycles. Frequently the BAUDOUT output gets tied to the RCLK input to be used in the receiver section also.

A divisor of one will generate a time delayed and buffered version of the TCLK input. A divisor of two generates a 50% duty cycle half rate version of the TCLK input. When the divisor(DIV) is 3 or greater the BAUDOUT will be low for two TCLK clock cycles and high for DIV-2 XIN clock cycles. A divisor(DIV) of 0 is modified to a divisor of 16 by the logic.

The register set

DLL - Baud rate generator register (least significant).

This register holds the 8 least significant bits of the baud rate divisor value. The DLAB bit must be set to one to enable write/read access to this register.

DLM - Baud rate generator register (most significant).

This register holds the 8 most significant bits of the baud rate divisor value. The DLAB bit must be set to one to enable write/read access to this register.

IER - Interrupt enable register

This register allows for individual enabling of interrupts.

7:4 RESERVED
3: Enable modem status change interrupt
2: Enable receive error detect interrupt

1: Enable transmitter empty interrupt
0: Enable data received interrupt

IIR - Interrupt ID register

When read, this register returns the encoded value of the highest priority interrupt pending. Bit 0 indicates whether an enabled interrupt is currently pending.

7: 550 mode enabled < buffer mode >
6: 550 mode enabled < buffer mode >
5:4 RESERVED
3:1 Interrupt ID
011 : receive error detected
 < highest priority >
110 : character timeout
010 : data received
001 : transmitter empty
000 : modem status change
 < lowest priority >
0: Interrupt inactive
 < interrupt pin inverted >

FCR - Buffer control register

This register enables and controls buffer operations.

- 7:6 Receive interrupt trigger level
 - 11: 14 bytes in the receive buffer
 - 10: 8 bytes in the receive buffer
 - 01: 4 bytes in the receive buffer
 - 00: 1 byte in the receive buffer
- 5:4 RESERVED
- 3: DMA mode
 - 1: TXRDY is activated when there are no bytes in the buffer and is deactivated only when the buffer is full, (16 bytes have been written to the buffer). RXRDY is activated when the trigger level in the buffer is reached and is deactivated when the buffer is below the trigger level, (fcr[7:6]).
 - 0: TXRDY is activated when there are no bytes in the buffer and is deactivated when there is at least one byte in the buffer. RXRDY is activated when there is at least one byte in the buffer and is deactivated only when the buffer is empty.
- 2: reset the transmit buffer pointer
- 1: reset the receive buffer pointer
- 0: enable buffer operational mode

THR - Transmit holding register

This register holds the data for the transmitter to send. The value from this register get serially transferred to the serial interface pin starting with the lowest order bit of the selected location. Bit 5 of the LSR register indicates whether the buffer currently has a value loaded waiting to transfer to the serial output pin.

RHR - Receive holding register

This register is the latched data from the receive buffer. It reflects the contents of the data buffer (stack) location which is currently pointed to. LSR bit 0 indicates if a character has been received.

LCR - Line control register

The LCR register provides the control bits necessary to control both the transmitter and receiver sections. Bit 7 of this register is the divisor latch access bit, which must be set to one to enable access to the Divisor registers.

LSR - Line status register

This register reflects the state of serial transfers. It indicates receive errors, and the status of the transmit buffer.

- 7: Any receive error
- 6: Serial transfer NOT active or pending
- 5: Transmit buffer empty
- 4: Break interrupt detected
- 3: Framing error detected
- 2: Parity error detected
- 1: Overrun in receive buffer detected
- 0: Data received <at least 1 character>

MCR - Modem control register

This register controls the interface with the MODEM or data set.

- 0: DTR This bit controls the Data Terminal Ready output. The register bit gets inverted and drives the N_DTR output.
- 1: RTS This bit controls the Request To Send output. The register bit gets inverted and drives the N_RTS output.
- 2: OUT1 This bit controls the Output 1 output. The register bit gets inverted and drives the N_OUT1 output.
- 3: OUT2 This bit controls the Output 2 output. The register bit gets inverted and drives the N_OUT2 output.
- 4: LOOP When this bit is set to one the macro goes into loop-back mode. In loop-back mode the SOUT bit is forced to one and the transmit output register is internally directed to the receiver logic replacing the SIN input. The modem outputs are forced into their inactive state (1) and the outputs from the Modem control register are internally directed to the modem inputs as follows: DTR -> CTS, RTS -> DSR, OUT1 -> RI, and OUT2 -> DSR.

MSR - Modem status register

This register indicates the state of the modem control and status bits.

- 7: DCD data carrier detect input signal
- 6: RI ring indicator input signal
- 5: DSR data set ready input signal
- 4: CTS clear to send input signal
- 3: DDCD delta - change in DCD detected since last read of MSR
- 2: TERI trail - trailing edge of RI detected since last read of MSR
- 1: DDSR delta - change in DSR detected since last read of MSR
- 0: DCTS delta - change in CTS detected since last read of MSR

SCR - Scratch register

This write/read register does not control any of the UART functions.

Register Address Map

ADDR:	DLAB:	OP:	pnu	BIT:	DESCRIPTION:	MODULE:
0	0	W	thr	7:0	Transmit data register	txt_eng
0	0	R	rhr	7:0	Receive data register	rcv_eng
0	1	R/W	dll	7:0	Baud rate register - low byte	reg_set
1	0	R/W	ier	3:0	Interrupt enable register	reg_set
1	1	R/W	d1m	7:0	Baud rate register - hi byte	reg_set
2	-	R	iir	7:0	Interrupt ID register	reg_set
2	-	W	fcr	7:0	Buffer control register	reg_set
3	-	R/W	lcr	7:0	Line control register	reg_set
4	-	R/W	mcr	4:0	Modem control register	reg_set
5	-	R/W	lsr	7:0	Line status register	reg_set
6	-	R/W	msr	7:0	Modem status register	reg_set
7	-	R/W	scr	7:0	Scratch register - user register	reg_set

table 2

Access to the registers is accomplished by 3 address pins and 3 chip select lines. Two strobes are used to read and write the internal registers. Bit 7 in the Line Control Register (DLAB = 1) is used to select the two baud rate registers when the part is being set up for operation. When in operation (DLAB = 0) the transmit, receive, and interrupt enable registers are enabled instead of the baud rate registers. The registers are written to on the rising edge of the PCLK (program clock) if the proper address has selected it. The two status registers (Line

and Modem) can be written in this same way. The bits in these registers are also written when the internal transmit and receive state machines reach certain conditions. The operational access to internal registers is synchronous, although asynchronous strobes can be used if there is at least two rising clock edges within the strobe active time. Read access is required to follow these same requirements not due to data bus output delays, but due to status register updates due to reading of the status registers.

Register Bit Descriptions

ADDR:	REGISTER:	OP:	DESCRIPTION:	@RESET
000 0	RHR	R	Receive Holding register	
000 0	THR	W	Transmit Holding register	
001 0	IER	R/W	Interrupt enable register 7:4 always 0000 3: Modem Status 2: Receiver line status 1: Transmitter Holding register empty 0: Receive Data Available	00
000 1	DLL	R/W	Baud divisor register (LEAST)	
001 1	DLM	R/W	Baud divisor register (MOST)	
010 -	IIR	R	Interrupt ID register 7:6 550 mode enabled 5:4 RESERVED 3:1 Interrupt ID[2:0] 0: 0 = Interrupt Pending	01
010 -	FCR	W	FIFO Memory Control register 7:6 Receive FIFO trigger level 5:4 RESERVED 3: DMA mode select 2: Transmit FIFO reset 1: Receive FIFO reset 0: FIFO memory enable	00
011 -	LCR	R/W	Line control register 7: Divisor latch access enable (DLAB) 6: Set break 5: Stick parity 4: Even parity select 3: Parity enable 2: Stop bit 1:0 word length select	00
100 -	MCR	R/W	Modem control register 7:5 always 000 4: Loopback 3: OUT2 2: OUT1 1: Request to send 0: Data terminal ready	00
101 -	LSR	R/W	Line status register 7: Any receive error 6: Transmitter Empty 5: Transmitter Holding Register Empty 4: Break interrupt 3: Framing error 2: Parity error 1: Overrun 0: Data ready	60
110 -	MSR	R/W	Modem status register 7: Data carrier detect 6: Ring indicator 5: Data set ready 4: Clear to send 3: Delta data carrier detect 2: Trailing edge ring indicator 1: Delta data set ready 0: Delta clear to send	x0
111 -	SCR	R/W	Scratch register	00

table 3

Two strobes are used to read and write the internal registers. Bit 7 in the Line Control Register (DLAB = 1) is used to select the two baud rate registers when the part is being set up for operation. When in operation (DLAB = 0) the transmit, receive, and interrupt enable registers are enabled instead of the baud rate registers. The registers are written to on the rising edge of

the PCLK (program clock) if the proper address has selected it. The two status registers (Line and Modem) can be written in this same way. The bits in these registers are also written when the internal transmit and receive state machines reach certain conditions. The operational access to internal registers is synchronous, although asynchronous strobes can be used if

there is at least two rising clock edges within the strobe active time, and two rising edges between active strobes. Read access is required to

follow these same requirements not due to data bus output delays, but due to status register updates due to reading of the status registers.

Clock Considerations

Shell level clock descriptions

The PCLK is the main clock for the VCM1016 design. It is assumed that this clock is at least 4X the XIN frequency due to sampling requirements for the one clock option, (synchronous, all flip flops are driven by the PCLK clock signal).

PCLK - Programming clock (or master clock) used for the program access to registers.

XIN - This clock is used to determine the baud rate so that the programming clock (PCLK) can run at a faster rate than what might be

needed to generate the baud rates, which are typically much slower. Many systems use a 1.8432 MHz signal for this pin. The N_BAUDOUT signal is the result of TCLK driving the baud counter (divider) circuit. This is considered to be 16X the serial bit stream rate.

RCLK - Receive clock used for clocking the receive state machines. This clock should be 16x the serial bit stream rate. It is usually connected to the BAUDOUT signal.

Core level clock descriptions

There are several options for clock configuration. There are three clock signals which actually clock registers. These are PCLK, TCLK and RCLK. A rising edge on these clocks will activate the register functions.

The PCLK, TCLK, and RCLK can be connected in a variety of ways. Each clock has an enable signal, PCLK_ENA, TCLK_ENA, and RCLK_ENA. This provides the maximum flexibility for configuring the UART. The baud_eq1 signal at the core level is useful in gating the rclk_ena signal. This is necessary when the fastest baud rate is to be used, and the RCLK and TCLK clocks are the same.

Synchronous System

For a completely synchronous UART, connect all three clocks to the main programming clock, PCLK. Connect the PCLK_ENA signal to the desired programming clock qualifier. Connect the TCLK_ENA signal to the desired transmit frequency. This enable signal should be generated from a pulse generation circuit. This circuit should generate a pulse which should be active for only one PCLK period for each rising edge of the transmit frequency. Connect the RCLK_ENA signal to a similar pulse generation circuit. This enable should be active for only

one PCLK period for each rising edge of the receive frequency. The enable signals for RCLK and TCLK should also be qualified with the PCLK_ENA signal.

An example of this type of system would be to connect the PCLK_ENA to a logic 1, and the PCLK to the processor or bus clock which is connected to the UART. The TCLK_ENA could be connected to a circuit which will sense the rising edge of the XIN signal. The RCLK_ENA will be connected to a circuit which will sense the rising edge of the receive clock (usually the BAUDOUT signal from the same UART).

In this configuration, there is no need to sync some internal signals between clock domains, since one clock drives all the flip flops. The constant (parameter) "p_sync_sig_ena" at the shell level should be set to '0'. This compile time constant sets constants in the reg_set and rcv_eng models to sync between clock domains, (if '1'), or pass signals through, (if '0'). If set to '1', the UART will still operate as before, but it will synthesize into more gates, and there will be clock delays on some register bits and signals.

Three Clock System

In this case, three clocks are used independently. The programming interface clock is used for PCLK, (PCLK_ENA = 1), the baud rate X 16 or transmit frequency clock is used for TCLK, (TCLK_ENA = 1), and the receive clock baud rate X 16 is used for RCLK, (RCLK_ENA = 1). The constant (parameter) "p_sync_sig_ena" at the shell level must be set to '1'. Signals will "re-sync" when crossing between clock domains.

Two Clock System

In a two clock system, the PCLK and PCLK_ENA could be connected as in the example above. The TCLK could be connected to the XIN signal, (baud clock frequency X 16 signal). The TCLK_ENA would be connected to logic 1 in this case. The RCLK could be connected to the TCLK signal also. The RCLK_ENA could be connected to a circuit which provides a one TCLK period enable signal on the rising edge of the BAUDOUT signal.

One possibility for this configuration is to be able to stop the high speed clock, PCLK. In order to be able to stop the clock yet have the UART still function, certain functions run off the TCLK rather than the PCLK. Functions like the interrupt circuits, and the modem status circuits relate to this TCLK signal, not the PCLK signal.

Clock Domain Crossings

In order to make sure that no signals are missed in crossing between any two clock domains, special circuitry is included to accommodate these events. The general assumption is that the PCLK runs faster than the TCLK, and the TCLK runs faster than the RCLK.

Operations for programming registers are considered immediate with relationship to the PCLK. Signals which relate to the transmit serial data stream relate to the TCLK. The receive serial data stream relates to the RCLK. The modem status and interrupt relate to the TCLK, due to the fact that it is a faster clock than the RCLK, and it is always running. The interaction between the PCLK and the other clock domains means that bits in the Line Status Register (LSR) Modem Status Register, (MSR) and the Interrupt ID Register, (IIR) to relate to both clock domains. This causes some bits to react immediately to programming reads, (PCLK domain) in order to clear bits at the end of a status read. The update of these status bits due to modem changes, or serial bit stream operations will be the result of a slower clock domain, (TCLK or RCLK). If the PCLK is shut down, this has the effect of actually being a slower clock than the TCLK. This results in more special circuits to catch events on status signals, and keep these posted until the PCLK resumes operation.

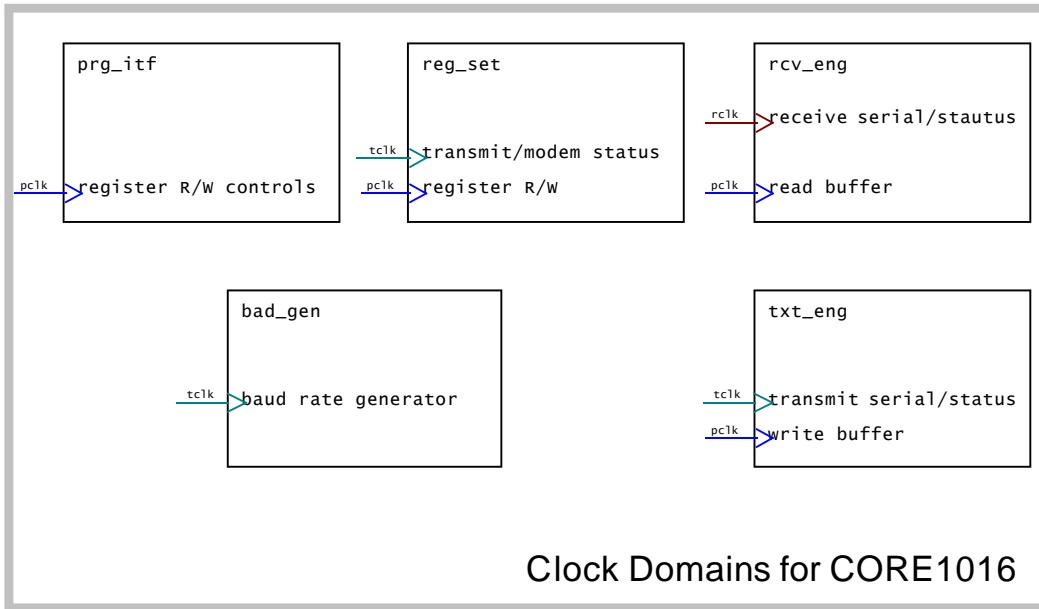


figure 5

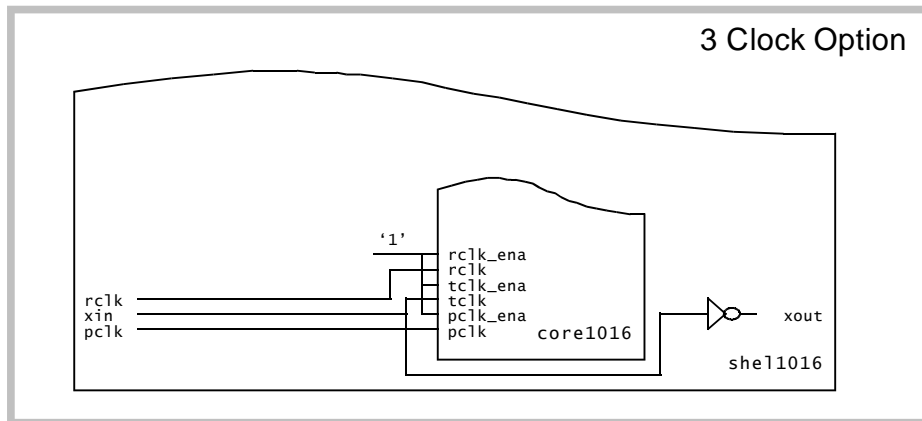


figure 6

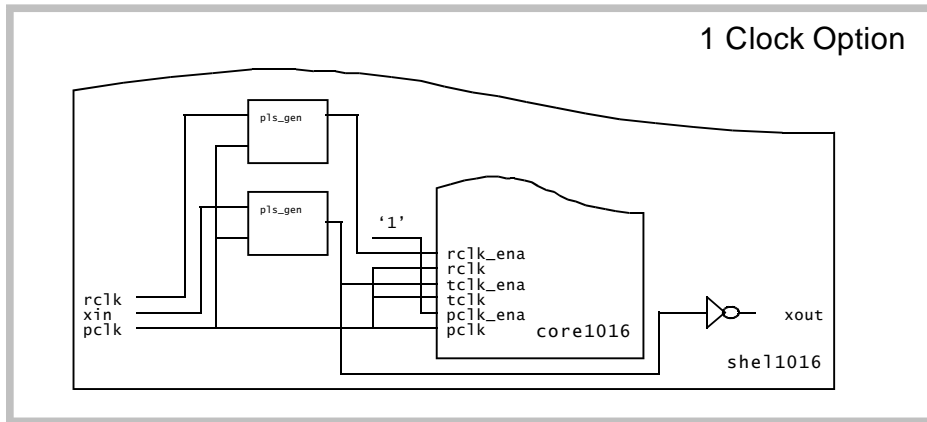


figure 7

Signal Clock Domains

Each core level signal relates to one clock domain. The signal may be sampled, or stored for another clock domain, but at the core level

there is only one clock domain used for each signal. Below is a chart showing the clock domains for each signal.

Core	Type	Core	Description
do[7:0]	OUT	PCLK	Programming Data bus output
di[7:0]	IN	PCLK	Programming Data bus input
n_r dout	OUT	PCLK	read enable for data bus <not>
addr[2:0]	IN	PCLK	Programming register selects (address bus)
n_ale	IN	PCLK	Address latch enable <not>
cs0	IN	PCLK	Chip select 0
cs1	IN	PCLK	Chip select 1
n_cs2	IN	PCLK	Chip select 2 <not>
rd	IN	PCLK	Programming read enable
n_rd	IN	PCLK	Programming read enable
wr	IN	PCLK	Programming write enable <not>
n_wr	IN	PCLK	Programming write enable <not>
intr	OUT	TCLK	Interrupt
n_txr dy	OUT	PCLK	Transmit buffer available <not>
n_r xrdy	OUT	PCLK	Receive data available <not>
reset	IN	- async -	Reset
pclk	IN	- na -	Programming clock <master clock>
pclk_ena	IN	PCLK	Programming clock enable
tclk	IN	- na -	Transmit baud generator clock
tclk_ena	IN	TCLK	Transmit baud generator clock enable
xout	OUT	TCLK	Reflects XIN input <not>
rclk	IN	- na -	Receive clock
rclk_ena	IN	RCLK	Receive clock enable
n_cts	IN	TCLK	Clear to send - modem <not>
n_dcd	IN	TCLK	Data carrier detect - modem <not>
n_dsr	IN	TCLK	Data set ready - modem <not>
n_ri	IN	TCLK	Ring indicator <not>
n_dtr	OUT	PCLK	Data terminal ready <not>
n_out1	OUT	PCLK	Programmable output 1 <not>
n_out2	OUT	PCLK	Programmable output 2 <not>
n_rts	OUT	PCLK	Request to send <not>
sin	IN	RCLK	Serial receive data in
sout	OUT	TCLK	Serial data stream transmit output
baudout	OUT	TCLK	Baud rate <x 16> output

table 4

Test Configuration

The VCM1016 UART is tested by a test behavior module which exercise the model and its various functions. Two UARTs are instantiated in the test. One is configured for receive, and the other is configured for transmit. There are also

two test drivers instantiated. There are two handshaking signals used as a way to communicate between the two instantiated test drivers. In this way, the two tests can keep in sync