INTEGRATED CIRCUITS

DATA SHEET

UART

PRODUCT SPECIFICATION

2001 Nov 23

UARTs

The XA-G3 includes 2 UART ports that are compatible with the enhanced UART used on the 8xC51FB.

Baud rate selection is somewhat different due to the clocking scheme used for the XA timers.

Some other enhancements have been made to UART operation.

The first is that there are separate interrupt vectors for each UARTs transmit and receive functions. The UART transmitter has been double buffered, allowing packed transmission of data with no gaps between bytes and less critical interrupt service routine timing.

A break detect function has been added to the UART. This operates independently of the UART itself and provides a start-of-break status bit that the program may test. Finally, an Overrun Error flag has been added to detect missed characters in the received data stream.

The double buffered UART transmitter may require some software changes in code written for the original XA-G3 single buffered UART.

Each UART baud rate is determined by either a fixed division of the oscillator (in UART modes 0 and 2) or by the timer 1 or timer 2 overflow rate (in UART modes 1 and 3).

Timer 1 defaults to clock both UART0 and UART1. Timer 2 can be programmed to clock either UART0 through T2CON (via bits R0CLK and T0CLK) or UART1 through T2MOD (via bits R1CLK and T1CLK).

In this case, the UART not clocked by T2 could use T1 as the clock source.

The serial port receive and transmit registers are both accessed at Special Function Register SnBUF. Writing to SnBUF loads the transmit register, and reading SnBUF accesses a physically separate receive register.

The serial port can operate in 4 modes:

Mode 0: Serial I/O expansion mode. Serial data enters and exits through RxDn. TxDn outputs the shift clock. 8 bits are transmitted/received (LSB first). (The baud rate is fixed at 1/16 the oscillator frequency.)

Mode 1: Standard 8-bit UART mode. 10 bits are transmitted (through TxDn) or received (through RxDn): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in Special Function Register SnCON. The baud rate is variable.

Mode 2: Fixed rate 9-bit UART mode. 11 bits are transmitted (through TxD) or received (through RxD): start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8_n in SnCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8_n.

On receive, the 9th data bit goes into RB8_n in Special Function Register SnCON, while the stop bit is ignored.
The baud rate is programmable to 1/32 of the oscillator frequency.

Mode 3: Standard 9-bit UART mode. 11 bits are transmitted (through TxDn) or received (through RxDn): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1).

In fact, Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SnBUF as a destination register. Reception is initiated in Mode 0 by the condition RI_n = 0 and REN_n = 1. Reception is initiated in the other modes by the incoming start bit if REN_n = 1.

Serial Port Control Register

The serial port control and status register is the Special Function Register SnCON, shown in Figure 12. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8_n and RB8_n), and the serial port interrupt bits (TI_n and RI_n).

TI Flag

In order to allow easy use of the double buffered UART transmitter feature, the TI_n flag is set by the UART hardware under two conditions. The first condition is the completion of any byte transmission.

This occurs at the end of the stop bit in modes 1, 2, or 3, or at the end of the eighth data bit in mode 0. The second condition is when SnBUF is written while the UART transmitter is idle.

In this case, the TI_n flag is set in order to indicate that the second UART transmitter buffer is still available.

Typically, UART transmitters generate one interrupt per byte transmitted. In the case of the XA UART, one additional interrupt is generated as defined by the stated conditions for setting the TI_n flag. This additional interrupt does not occur if double buffering is bypassed as explained below. Note that if a character oriented approach is used to transmit data through the UART, there could be a second interrupt for each character transmitted, depending on the timing of the writes to SBUF. For this reason, it is generally better to bypass double buffering when the UART transmitter is used in character oriented mode. This is also true if the UART is polled rather than interrupt driven, and when transmission is character oriented rather than message or string oriented.

The interrupt occurs at the end of the last byte transmitted when the UART becomes idle.

Among other things, this allows a program to determine when a message has been transmitted completely. The interrupt service routine should handle this additional interrupt.

The recommended method of using the double buffering in the application program is to have the interrupt service routine handle a single byte for each interrupt occurrence. In this manner the program essentially does not require any special considerations for double buffering. Unless higher priority interrupts cause delays in the servicing of the UART transmitter interrupt, the double buffering will result in transmitted bytes being tightly packed with no intervening gaps.

9- bit Mode

Please note that the ninth data bit (TB8) is not double buffered.
Care must be taken to insure that the TB8 bit contains the intended data at the point where it is transmitted. Double buffering of the UART transmitter may be bypassed as a simple means of synchronizing TB8 to the rest of the data stream.

Bypassing Double Buffering

The UART transmitter may be used as if it is single buffered. The recommended UART transmitter interrupt service routine (ISR) technique to bypass double buffering first clears the TI_n flag upon entry into the ISR, as in standard practice. This clears the interrupt that activated the ISR. Secondly, the TI_n flag is cleared immediately following each write to SnBUF. This clears the interrupt flag that would otherwise direct the program to write to the second
transmitter buffer.

If there is any possibility that a higher priority interrupt might become active between the write to SnBUF and the clearing of the TI_n flag, the interrupt system may have to be temporarily disabled during that sequence by clearing, then setting the EA bit in the IEL register.

Baud Rate for UART Mode 0 :

Baud_Rate = Osc/16

Baud Rate calculation for UART Mode 1 and 3 :

Baud_Rate = Timer_Rate/16

Timer_Rate = Osc/(N*(Timer_Range– Timer_Reload_Value))

where N = the TCLK prescaler value: 4, 16, or 64. and Timer_Range = 256 for timer 1 in mode 2.
65536 for timer 1 in mode 0 and timer 2 in count up mode.

The timer reload value may be calculated as follows:

Timer_Reload_Value = Timer_Range–(Osc/(Baud_Rate*N*16))

NOTES :

1. The maximum baud rate for a UART in mode 1 or 3 is Osc/64.
2. The lowest possible baud rate (for a given oscillator frequency and N value) may be found by using a timer reload value of 0.
3. The timer reload value may never be larger than the timer range.
4. If a timer reload value calculation gives a negative or fractional result, the baud

rate requested is not possible at the given oscillator frequency and N value.

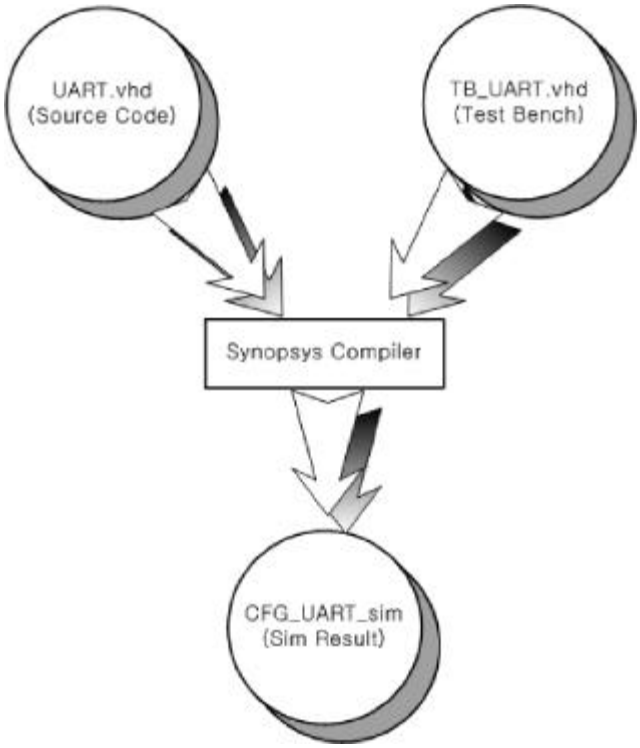
Baud Rate for UART Mode 2 :

Baud_Rate = Osc/32

Pin Description

Pin Name	Type	Description
clock	Input	main clock pin
reset	Input	main reset pin
ri_ff_out	Input	RI_x in SxCON register
sm2	Input	RI in SxCON register
smod	Input	SM0, SM1 in SxCON register
ren	Input	REN_x in SxCON register
timer1_ov	Input	timer1 overflow pin
timer2_ov	Input	timer2 overflow pin
rclk	Input	receive clock
tclk	Input	transmit clock
sbuf_wr_en	Input	serial buffer write enable pin
sbuf_r_en	Input	serial buffer read enable pin
tb8	Input	TB8_x in SxCON register
int_data_bus	Input	data bus pins
ProgtamedAddress	Input	address bus pins
rxd_pin_in	Input	from external (RxDx) pin
ri	Output	RI_x (input enable) in SxCON register
ri_load	Output	RI_x (write enable) in SxCON register
rb8_in	Output	RB8_x (input enable) in SxCON register
rb8_load	Output	RB8_x (write enable) in SxCON register
sbuf_out	Output	serial output buffer
ti	Output	TI_x (input enable) in SxCON register
rx_clock	Output	SxSTAT Bit write enable(FE,BR,OE)
ti_load	Output	TI_x (write enable) in Sx_CON register
txd	Output	External output pin (Sync. clock output)
FE	Output	FEx in SxSTAT register
BR	Output	BRx in SxSTAT register
OE	Output	OEx in SxSTAT register
smod0_rxd	Output	Data out on Mode 0 (rxd_pin_in's out)

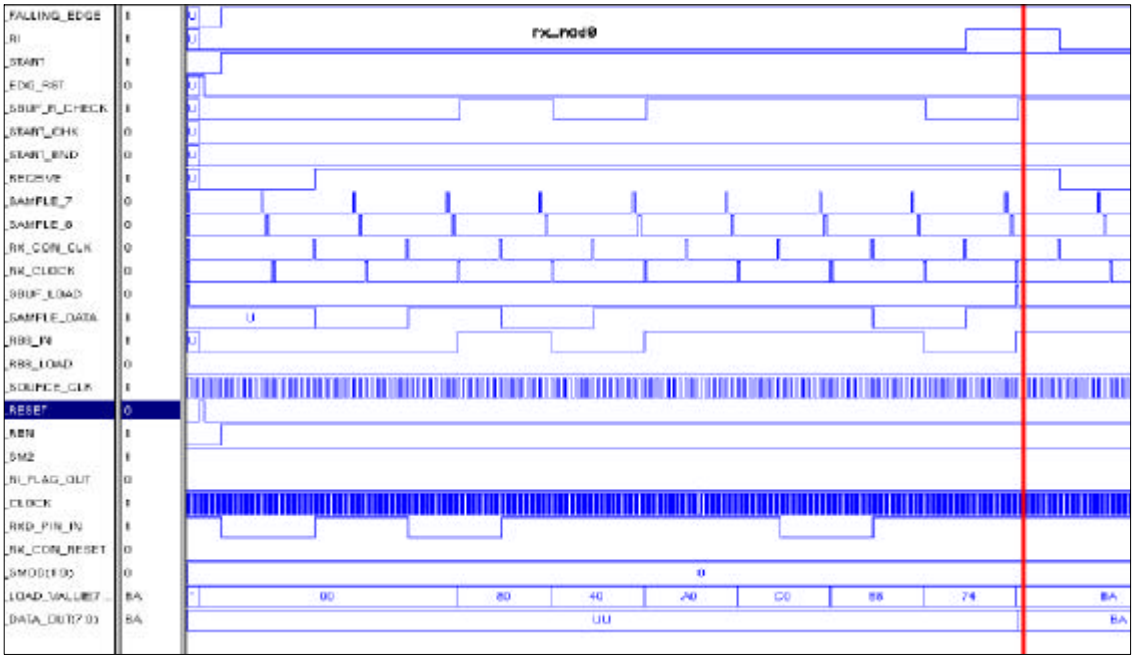
Prototype method



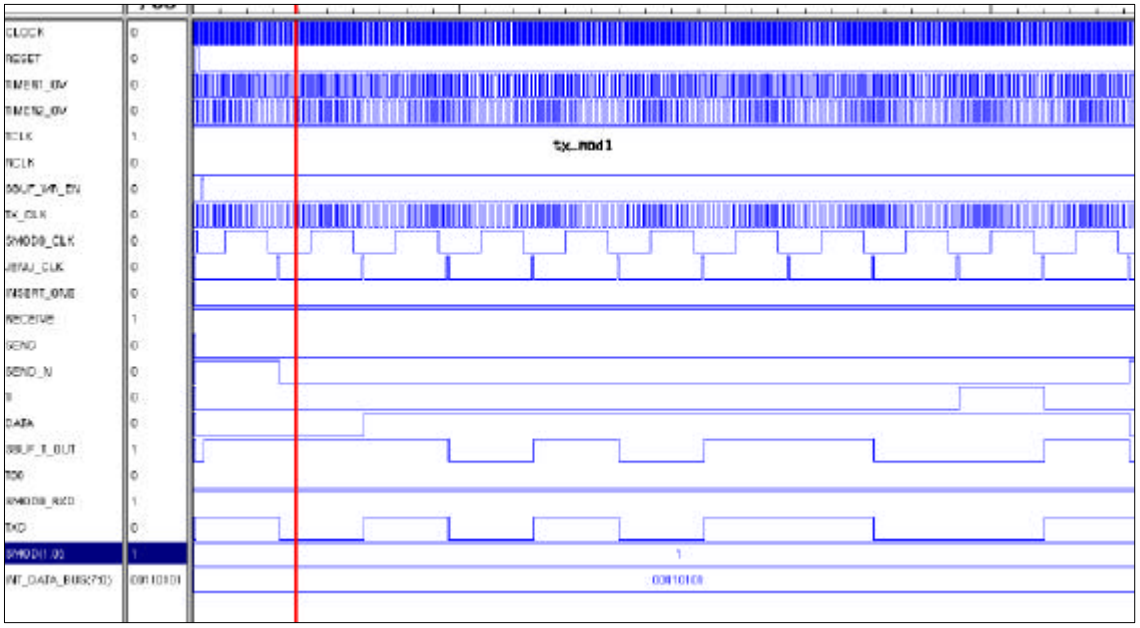
uart.vhd and tb_uart.vhd compile to verified simulation result file (cfg_uart_sim).

※ Compiler Tool : Synopsys Compiler

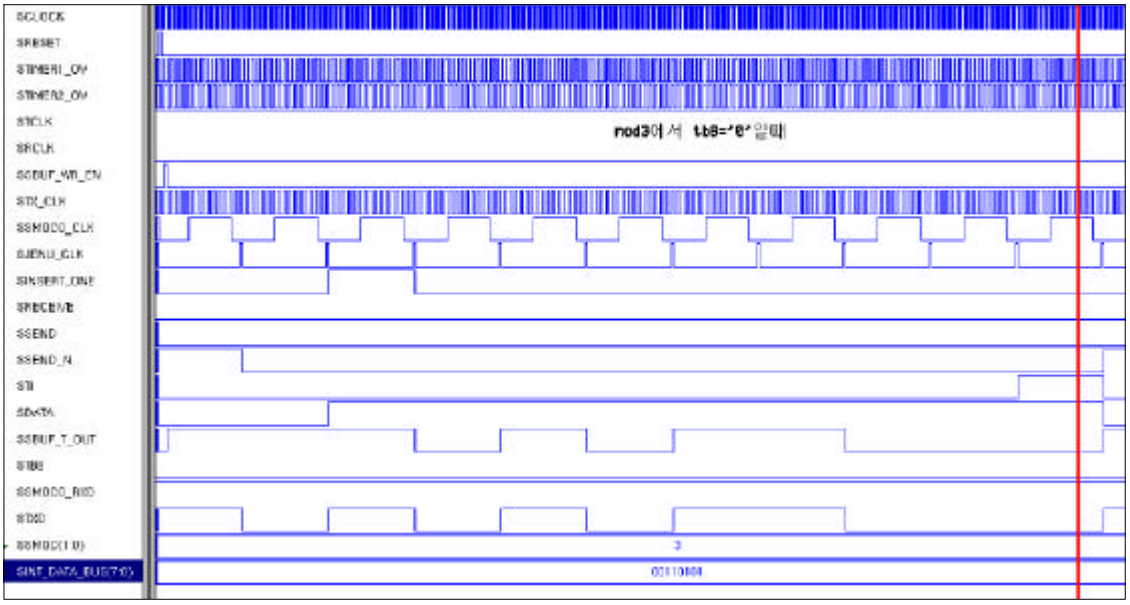
Result Simulation Graph (Transmitter part)



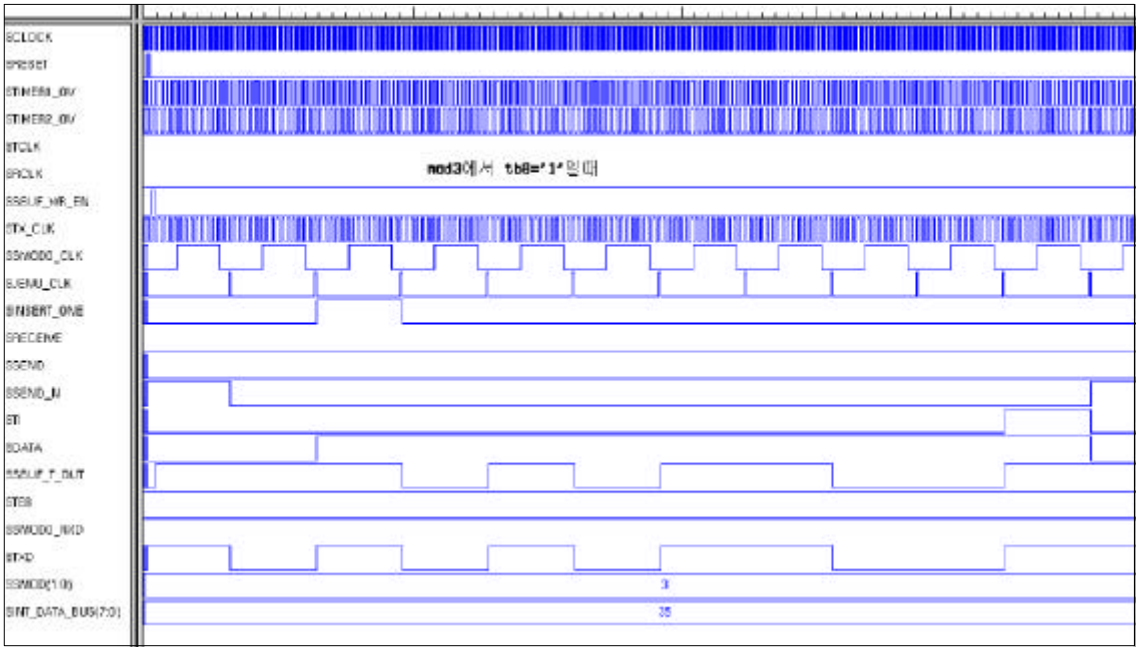
Transmitter part mode 0



Transmitter part mode 1

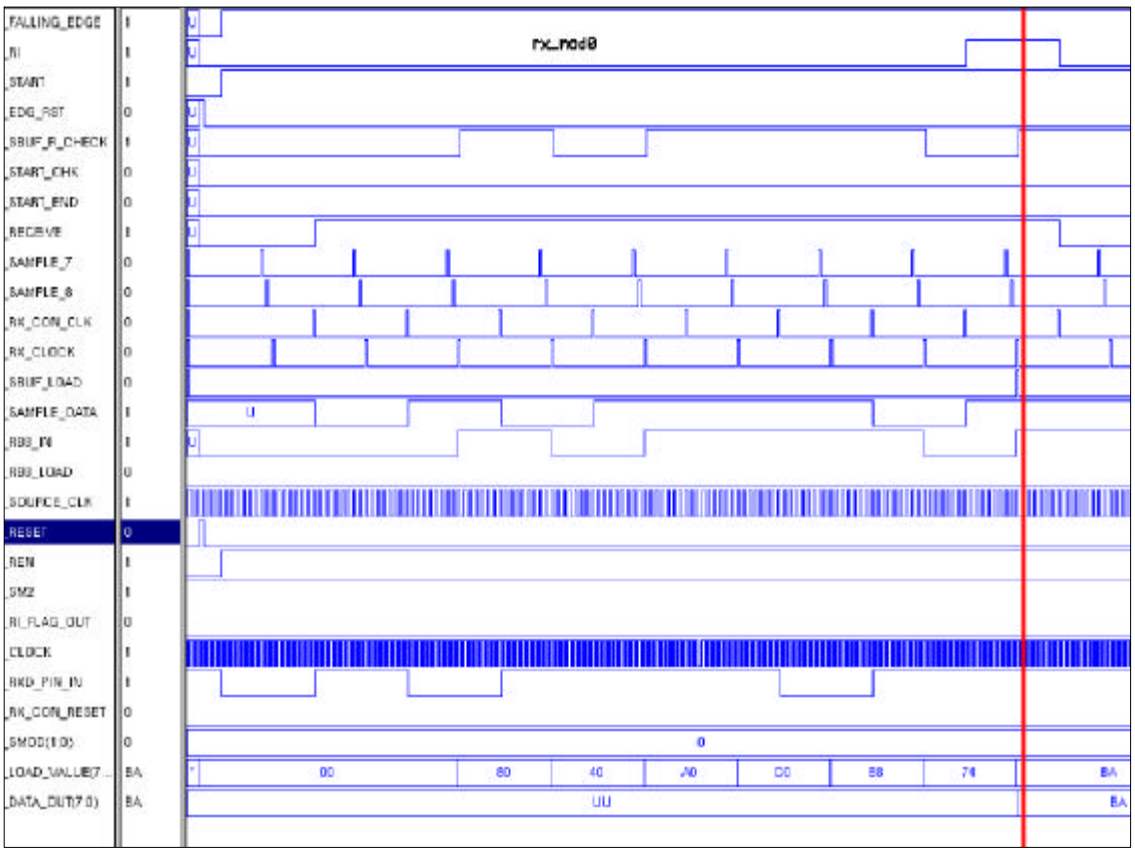


Transmitter part mode 3 (tb=0)

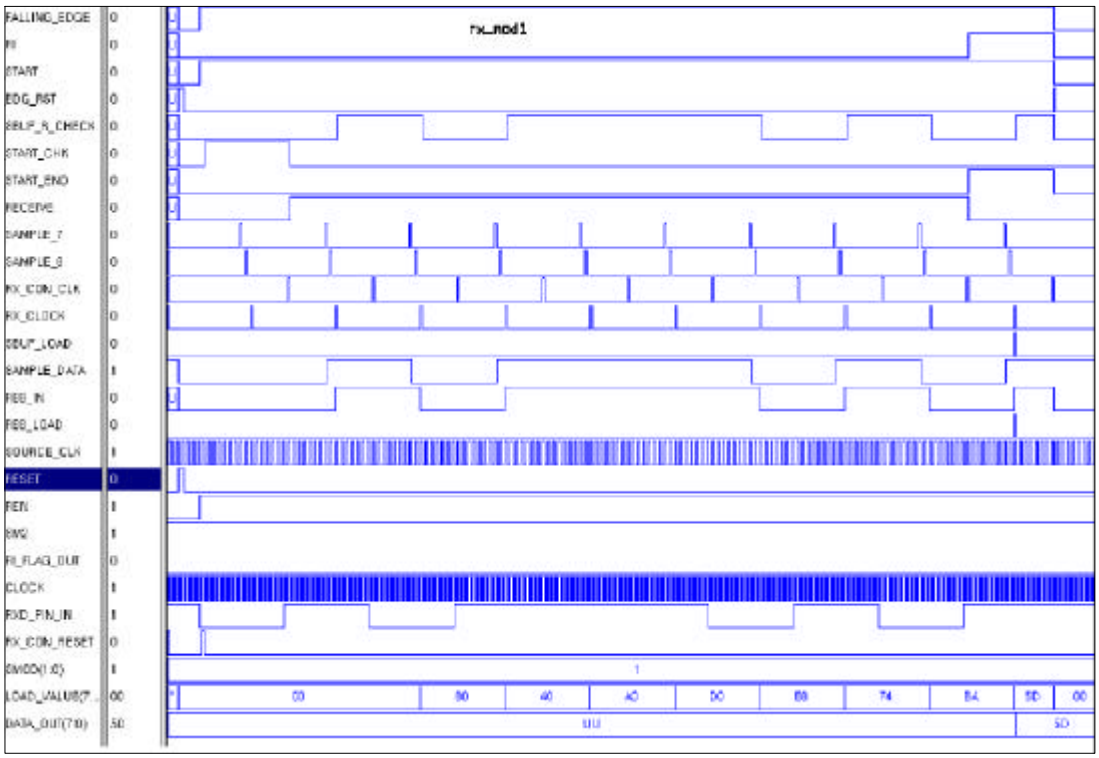


Transmitter part mode 3 (tb=1)

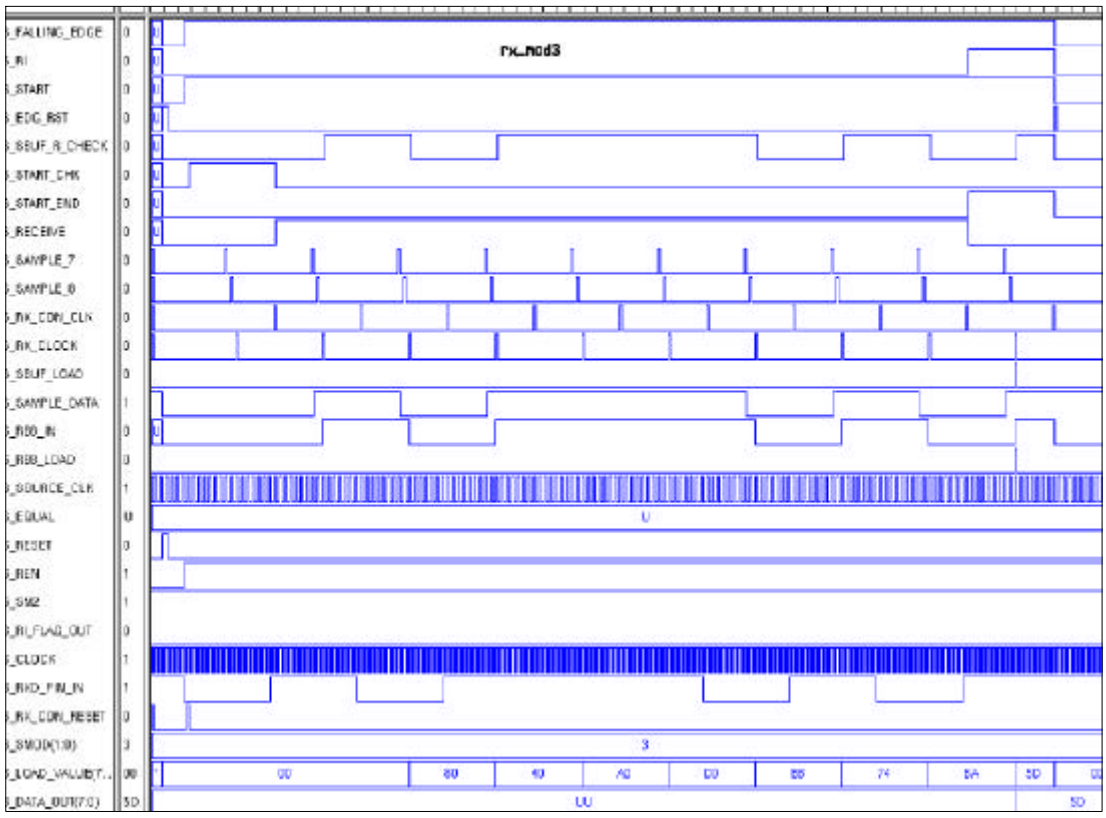
Result Simulation Graph (Receive part)



Receive part mode 0



Receive part mode 1



Receive part mode 3