Lab 7
UART Transmitter

Name__________________________ **Class** __________ **Date** ___________________

OBJECTIVES:

Upon completion of this laboratory exercise, you should be able to:

- Create and simulate a shift register that can be used as a transmitter in a universal asynchronous receiver/transmitter (UART)
- Develop a test circuit for the UART transmitter that will repeatedly load and transmit a serial character to be monitored on an oscilloscope. The character will be transmitted at a fixed rate of 9600 baud.
- Modify the test circuit so that its output (a repeating ASCII character) can be monitored on a PC running HyperTerminal or similar communication software.

EQUIPMENT REQUIRED:

DE1 or DE2 FPGA Circuit Board with USB Blaster Download Cable
Quartus II Software
PC Running HyperTerminal
AC Adapter, minimum output: 7 VDC, 250 mA DC
Anti-static wrist strap

EXPERIMENTAL NOTES:

A common use of serial shift registers is in a serial data transmission system, which is used to send data from one circuit to another via a single transmission path. A simplified block diagram of a serial transmission system is shown in Figure 1.

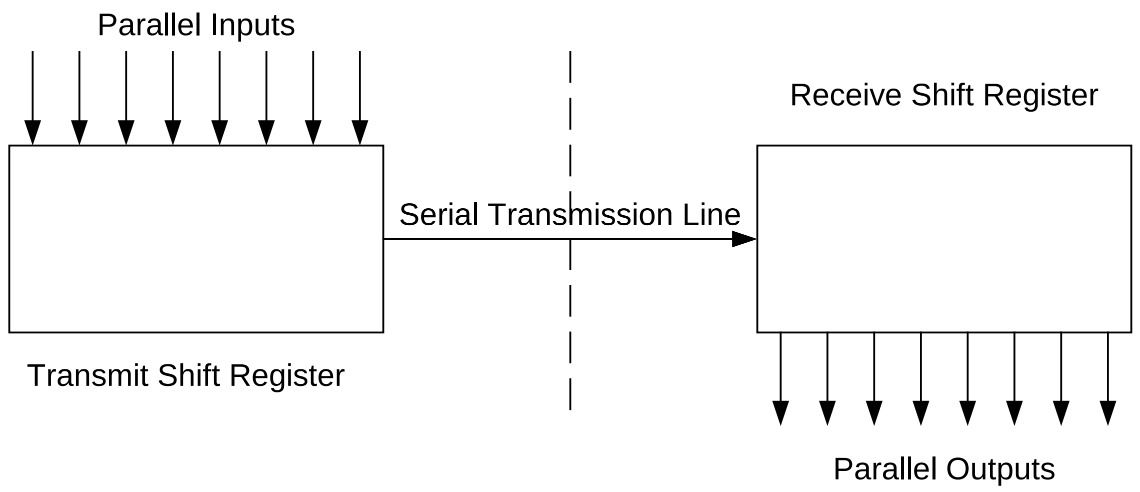


Figure 1 Simplified Serial Transmission System

In the system in Figure 1, parallel data are loaded into a shift register on the transmit side of the system, then shifted across a single transmission line to a serial shift register on the receive side of the system. The received data are read in parallel from the receive shift register and stored in some appropriate location, such as system memory, or else sent to an output device such as a display monitor or printer.

The receiver and transmitter in Figure 1 are assumed to be some distance apart, as indicated by the dotted line. This distance could be as close as a couple of inches away, or as far as many miles away, or even around the globe, depending on the particular system in use. The idea of serial transmission is that there can be great savings in the amount of connector hardware and cabling, if data can be sent one bit after the other over one line, as opposed to many bits in parallel over multiple lines. The character could be transmitted synchronously, where the transmitter and receiver are clocked with the same signal, or asynchronously, where the transmitter and receiver have their own clocks.

Figure 2 shows the format of an asynchronously-transmitted ASCII character at TTL levels. The transmission line remains HIGH when no characters are transmitted. To indicate the start of a character, the line goes LOW for one bit time (Start bit). This is followed by 8 data bits, LSB first. The character ends when the line goes HIGH for one bit time (Stop bit).

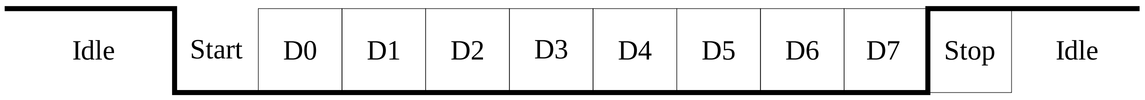


Figure 2 ASCII Composite Character

The Start and Stop bits are appended to the character when it is parallel-loaded into the transmit shift register, as shown in Figure 3. Notice that the character in Figure 2 seems "backwards" compared to the way it is loaded in Figure 3. This is because the Start bit is transmitted first, which would show up on the left of an oscilloscope trace, since this represents the earliest point in time.

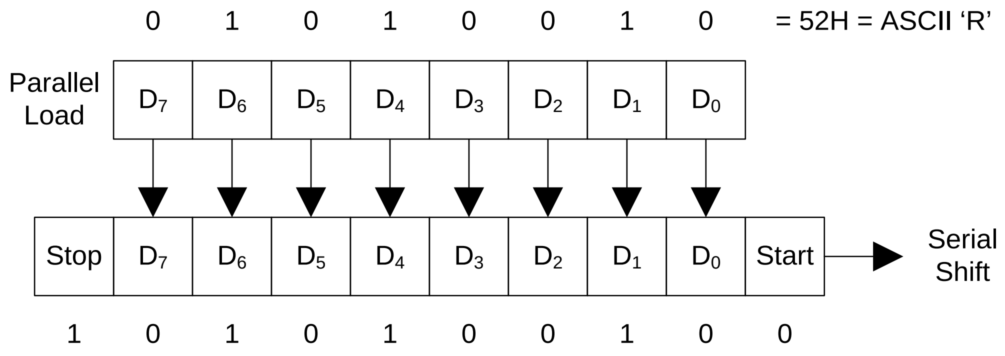

0 1 0 1 0 0 1 0 = 52H = ASCII 'R'
Parallel Load
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
Stop D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 Start
Serial Shift
1 0 1 0 1 0 0 1 0 0

Figure 3 Appending Start and Stop Bits to an ASCII Character

In this lab, we will monitor the bits of the character will be derived from 8 switches on the Altera DE1 or DE2 education board. A clock divider and a decoded counter will repeatedly load and transmit the character, first to an oscilloscope, then later to a PC serial port, where it will be observed using HyperTerminal, a communication program that is packaged with Windows.

Lab 7
UART Transmitter

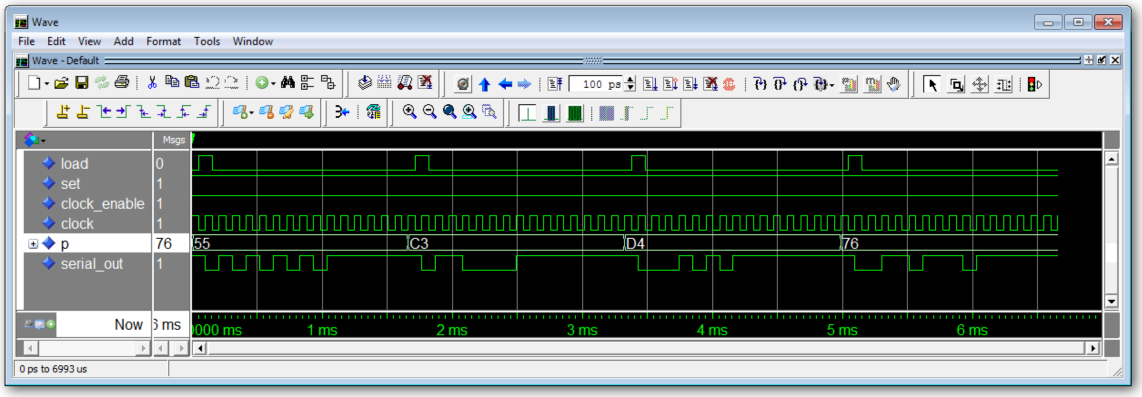
PROCEDURE:

1. Create a serial transmitter in VHDL that loads 8-bit data to a shift register, appends start and stop bits, and transmits the ASCII composite character in serial format. The device on the DE2 board is in the Cyclone II family, part number EP2C35F672C6.

TRANSMIT SHIFT REGISTER:

- Clock drives synchronous operations.
- Active-LOW asynchronous set that sets all bits HIGH, tied to the system reset line. (Sets output to idle state when not transmitting a character.)
- Active-HIGH synchronous load.
- Internal signal acts as shift register with 10-bit width. (See Figures 2 and 3.)
  - o 8-bit parallel inputs, which are used to load transmission data into middle 8 bits of shift register.
  - o Internal signals to append start and stop bits to data when parallel-loaded.
  - o Serial output, to transmit data to receiver via RS232 driver when enabled.
  - o Serial output, to transmit shift register output to oscilloscope for waveform monitoring. (Use the same source , but different output pins, for the two serial outputs.)
  - o 8-bit parallel outputs, configured as mode BUFFER, which are:
    - ▪ asynchronously cleared when the set input goes LOW and
    - ▪ synchronously loaded with the parallel input data when the load input goes HIGH.
  - o Internal serial input, set HIGH, to fill shift register with ones while shifting (Required to maintain idle state between characters. You can use a SIGNAL and set it to a constant value. E.g., `serial_in <= '1';`)

2. Create a simulation that shows the operation of the serial transmitter created in part 1, above. Show the simulation to your instructor.



Instructor's Initials_________

3. Create a test circuit for the serial transmitter as shown in Figure 4 (next page). The tester consists of:

- the transmit shift register;

- two hexadecimal-to-seven-segment decoders (**hex7seg.vhd**) to display the parallel data that has been loaded into the shift register.
- a 4-bit counter (**uart_load_counter**) that will load parallel data into the shift register once every 16 clock pulses. It does so by decoding "1111" with an active-HIGH output called **cout** (carry output). The load pulse enables the shift register at a rate of 9600 baud. Use an LPM counter with a carry output to generate the load pulse for the shift register.
- a clock divider called **clock_divider.vhd** that divides a 50 MHz clock input and produces a 9600 Hz clock output for the transmit clock (**tx_clock**) and an output with frequency f_{rx} = 9600 baud × 16 for the receiver clock (**rx_clock**; needed in a later lab). Instructions for the clock divider structure are given on the next page.

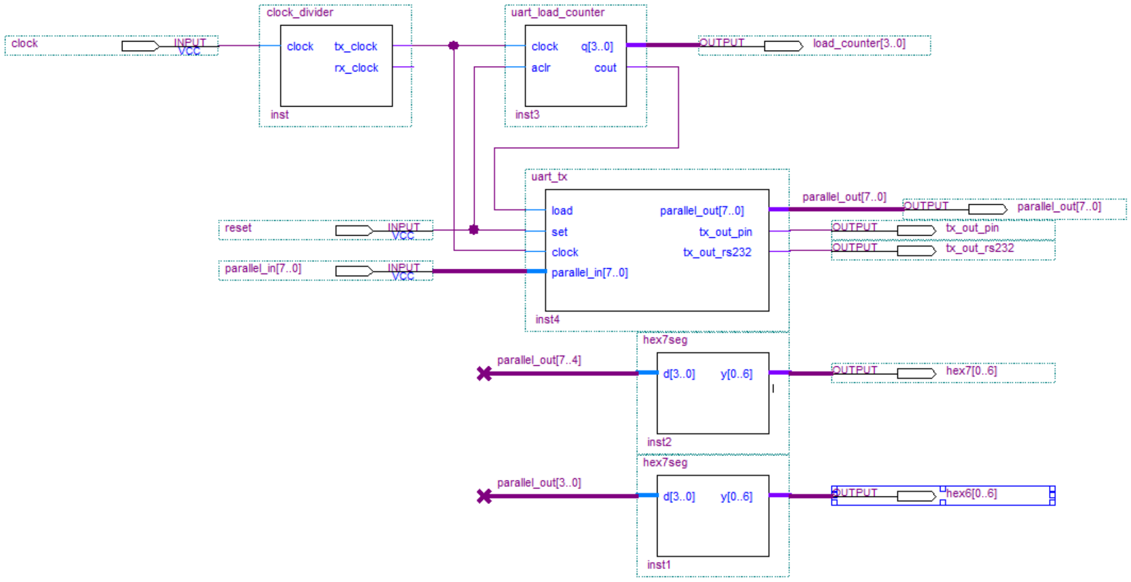


Figure 4 Serial Transmitter Test Circuit

Clock Divider

The clock divider is a top-level VHDL or Block Diagram File with two LPM counters clocked by the 50 MHz DE1/DE2 Board clock. A block diagram in shown in Figure 5.

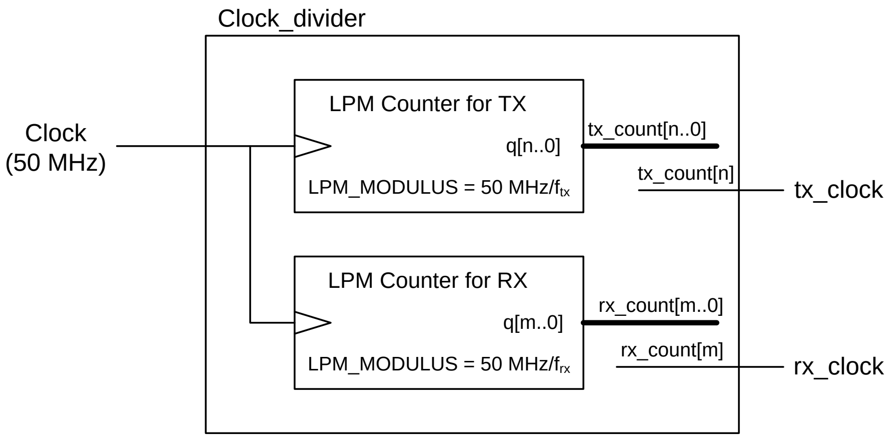


Figure 5 Clock Divider for Tx and Rx

- Create a top-level file with input and outputs as shown on the clock divider block in Figures 4 and 5.
- Instantiate two LPM counters: one for tx_clock and one for rx_clock.
  - Required parameters: LPM_WIDTH, LPM_MODULUS.
    - Calculate LPM_MODULUS by dividing 50 MHz/f_output and using the nearest integer value. (Tx: 9600 Hz; Rx: (16 × 9600) Hz)
    - LPM_WIDTH is greater than or equal to the number of bits required to hold the binary equivalent of LPM_MODULUS.
    - Note: $LPM\_WIDTH \geq \frac{\log(LPM\_MODULUS)}{\log(2)}$
  - Required ports for each LPM counter: **clock** and **q**.
  - MSB of **q** maps to **tx_clock** or **rx_clock**, as shown in Figure 5.

4. Verify the operation of the test circuit by creating a simulation like that in Figure 6. (Use ModelSim.) Make the end time 4 ms. Make the clock frequency 50 MHz, the same as the DE1/DE2 clock.

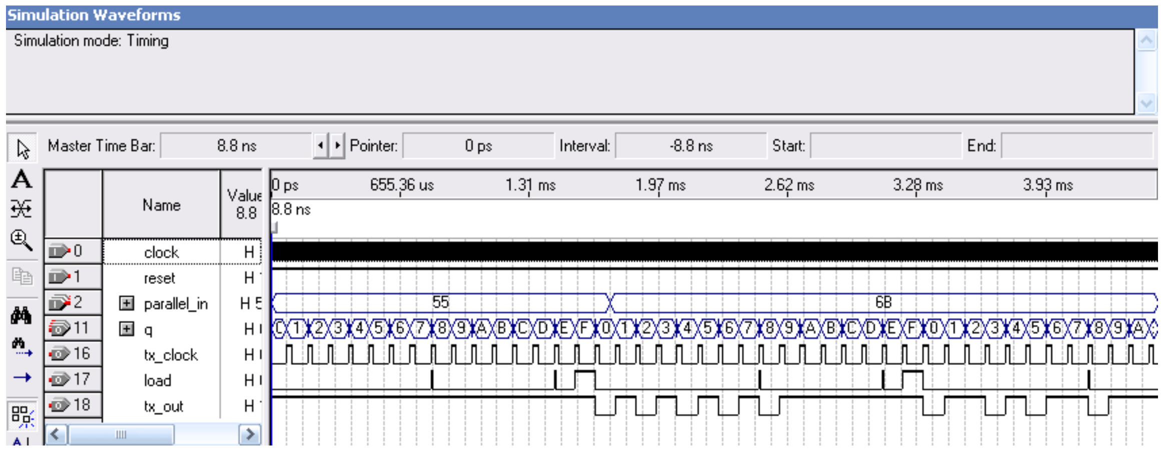


Figure 6 Simulation Waveforms for Transmitter Test Circuit

5. Assign pins to the circuit, as shown in the table below and recompile the project.

Signal	Pin	Device
clock		50 MHz clock (internal)
reset		KEY[0]
parallel_in[7]		SW[7]
parallel_in[6]		SW[6]
parallel_in[5]		SW[5]
parallel_in[4]		SW[4]
parallel_in[3]		SW[3]
parallel_in[2]		SW[2]
parallel_in[1]		SW[1]
parallel_in[0]		SW[0]
tx_out_rs232		On-board RS-232 Tx
tx_out_pin		JP2-40 (40-pin header)
load		JP2-39 (40-pin header)
ground		JP2-30 (40-pin header)

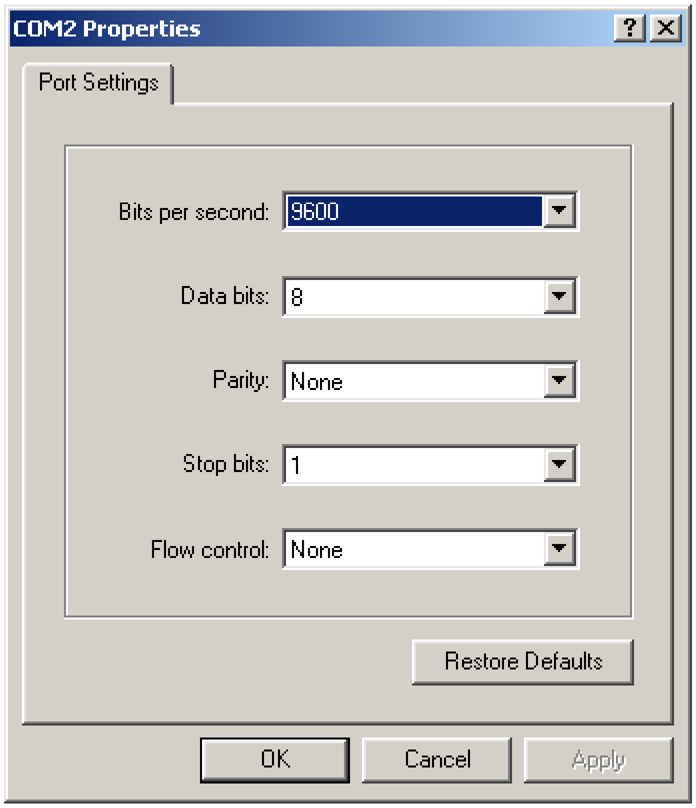
6. Program your DE1/DE2 lab board with the serial transmitter. Monitor the **load** and **tx_out** pins with an oscilloscope. (Plug a 40-pin ribbon cable into JP2 to get access to these pins.) Refer to the ASCII table at the back of this lab handout and transmit the code for 'u'. Draw the resultant waveform at TTL logic levels. Change the inputs to show transmission of several other ASCII characters. Show the operation of the transmitter to your instructor.

Instructor's Initials _________

TESTING THE TRANSMITTER USING HYPERTERMINAL

1. Connect one of the serial ports of your PC to the 9-pin D connector on the Altera DE1/DE2 board.

3. Open HyperTerminal in Windows and create a session with the settings shown in Figure 7 (next page). Make sure that the COM port selected in HyperTerminal (probably COM1 or COM2) is the one you connected to the DE2.



Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Figure 7 HyperTerminal Settings

5. Set the parallel-input data switches to send the letter 'u'. Change the switches to demonstrate several other letters. Show the operation to your instructor.

Instructor’s Initials _________

ASCII Table

	MSBs							
	000	001	010	011	100	101	110	111
LSBs	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
0000 (0)	NUL	DLE	SP	0	@	P	’	p
0001 (1)	SOH	DC1	!	1	A	Q	a	q
0010 (2)	STX	DC2	″	2	B	R	b	r
0011 (3)	ETX	DC3	#	3	C	S	c	s
0100 (4)	EOT	DC4	$	4	D	T	d	t
0101 (5)	ENQ	NAK	%	5	E	U	e	u
0110 (6)	ACK	SYN	&	6	F	V	f	v
0111 (7)	BEL	ETB	′	7	G	W	g	w
1000 (8)	BS	CAN	(	8	H	X	h	x
1001 (9)	HT	EM	)	9	I	Y	i	y
1010 (A)	LF	SUB	*	:	J	Z	j	z
1011 (B)	VT	ESC	+	;	K	[	k	{
1100 (C)	FF	FS	,	<	L	\	l	|
1101 (D)	CR	GS	-	=	M	]	m	}
1110 (E)	SO	RS	.	>	N	^	n	~
1111 (F)	SI	US	/	?	O	_	o	DEL