Chapter 7

UART Transmitter

The purpose of this laboratory is to design a Universal Asynchronous Transmitter (a UART without the receiver). This transmitter will be used to send ASCII characters from your FPGA board to your PC workstation. You will use this transmitter throughout the semester.

Exercises

Exercise 1 – Transmitter Overview

The lab is the first of a two part series of labs involved in the construction of a Universal Asynchronous Receiver/Transmitter (UART). In the first part, you will design, code, and test a serial transmitter. In the next lab, you will design, code, and test a serial receiver in VHDL. To begin this laboratory, read the first half of page 13 of the Nexys2 reference manual to learn how the serial port is hooked up on the Nexys2 board. After reading the Nexys2 manual, watch the following screencast to learn more about the UART transmitter and the lab:

Screencast: UART Overview – Part 1

After watching the screencast and before proceeding with your design, answer the following questions:

Question: What is the purpose of the ST-3232 chip used on the Nexys2 board?

Question: What is the purpose of the "start" bit in the serial protocol?

Question: In what order are the bits sent over the UART? Least significant bit (LSB) first or Most significant bit (MSB) first?

Question: For a 19,200 baud system serial communication link, how long does it take to transfer an 8-bit character (including the start and stop bits)?

Question: At 19200 baud, how many characters per second can be transmitted?

Question: What percent of transmission time is overhead (time of the start and stop bits divided by the total frame time)?

Question: Suppose a file of 10,000 bytes is to be sent over a line at 19200 bps. How much time will it take in seconds?

Exercise 2 – Transmitter Design

Watch the second part to the UART screencast to get an an overview of the UART transmitter design. In this lab we will be providing you with most of the design architecture. This screencast describes a block diagram, a state machine description, and sufficient information for you to create a transmitter in VHDL. You will need to understand this design before you proceed with your VHDL coding.

Screencast: Transmitter Design Overview – Part 2

The slides used in the screencast are available for your reference. Go back and review these slides if you have questions about the design.

Download: UART Transmitter Overview Slides

After watching the screencast and reading through the UART transmitter overview slides, answer the following questions:

Question: What is the purpose of the BIT_TIMER_COUNT constant used within the bit-timer circuit?

Question: Why should the "transmit out" signal be registered with a flip-flop?

Question: What is the purpose of the RETRN state in the state machine?

Question: What value should be asserted on the TX output during the IDLE and RETRN states?

Once you have watched the UART transmitter design screencast, begin coding and simulating your reusable UART transmitter. Begin by creating a VHDL source file called `tx.vhd` (ensure your entity is named this way so it will work with the testbench). Create an entity with the following input ports, output ports, and generics (the purpose of these ports and generics is described in the screencast):

Port Name	Direction	Width	Purpose
clk	Input	1	50 MHz clock
rst	Input	1	Asynchronous reset
data_in	Input	8	Byte to send over transmitter
send_character	Input	1	Control signal to initiate transmission of byte
tx_out	Output	1	Serial data to be transmitted
tx_busy	Output	1	Indicates that the transmitter is busy sending a character

Generic	Type	Purpose
CLK_RATE	NATURAL	Indicates the frequency of the input clock (Default=50_000_000)
BAUD_RATE	NATURAL	Baud rate of the transmitter (Default=19_200)

You will need to determine the width of your bit timer to measure the proper time for a bit period. You can determine this width based on the CLK_RATE and BAUD_RATE generics. To determine the width of this bit timer, you need a way to compute the base-2 logarithm. Add the function, log2c, listed on page 492 of your textbook to the declaration section of your architecture. Use the log2c function to create a constant within your architecture indicating the bit width of your bit timer. It is necessary to declare the log2c function before you use it. Here is some sample code that demonstrates how to use this function to determine the size of your counter:

```
constant BIT_COUNTER_MAX_VAL : Natural := CLK_RATE / BAUD_RATE - 1;
constant BIT_COUNTER_BITS : Natural := log2c(BIT_COUNTER_MAX_VAL);
```

Design each of the "datapath" components described in the design document individually (bit timer, shift register, and transmit out). In other words, create each of the components one at a time and make sure they work individually before you integrate them. Students frequently waste a lot of time integrating design components that do not work. Create the FSM as described in the screencast (again, do this after you have verified the correct function of the datapath components). You may want to review state machine design from Section 10.5 in the text. After creating your VHDL, remove all compilation errors and perform a few simple tests to see if the circuit is working as you expect it to. Write a simple .tcl script to simulate the behavior of your circuit.

Exercise 3 – Testbench Simulation of Transmitter

Like the previous labs, a testbench has been provided for you to test your transmitter under a number of conditions. This testbench will simulate your transmitter sending a several bytes and perform a number of checks to make sure you send the data correctly (i.e., the correct amount of time and the correct order). Make sure you simulate until the "DONE" message is printed. Simulate your design until it passes all of the tests.

Question: How long does the simulation take to complete? (i.e., when the "Test Done" message is printed)

When your design has passed all of its tests, upload your design.

Upload: Submit your tx.vhd transmitter

Exercise 4 – Top-Level Transmitter Design

The second design required for this laboratory is a top-level VHDL entity. This top-level file will contain your transmitter, the seven segment display controller, debounce circuitry, and connections between the I/O and the logic. This top-level design will allow you to send ASCII characters from your FPGA board to a terminal on your workstation. You will select the ASCII character by setting the switches to the binary value of the ASCII character you want to send. When you have selected your ASCII character, you will press a button to initiate the transfer.

Follow these guidelines to create this top-level design:

- Instance your UART transmitter that you created earlier in the lab.
- Instance the seven segment display controller that you created in a previous lab.
- Wire the board switch inputs to the data_in of your transmitter. Also wire the switches to the lowest 8 bits of the data_in input of your seven segment display. This will allow you to see the current HEX value of the switches and make sure you are sending the appropriate character.
- Use one of the buttons for the "send_character" signal of your transmitter. You will need to add debouncing logic to this button (see notes in the screencast on debouncing).
- Use one of the buttons for the asynchronously resets in your design.

A debouncer circuit is needed to remove the "Bouncing" of the input button. Watch *Part 3: Debouncer* of the screencast and create a debouncer for your design.

Screencast: Debouncer – Part 3

Carefully simulate your entire design to make sure the reset logic, debouncer, transmitter and display controller work correctly. You may need to change the 'debounce time' to make it easier to simulate the full circuit.

Create a .ucf file for your design that includes all of the top-level ports used in the design. Make sure you have an entry for the `tx` signal. Also, make sure you have a timing constraint for the clock. Synthesize your design and carefully review the warnings generated during synthesis. Make sure you read and understand each one.

Question: Summarize and justify all of the synthesis warnings you received when synthesizing this circuit.

Question: Review the synthesis log to determine the state encoding of your transmitter state machine. Cut and paste the encoding for this question.

Upload: Submit your top-level VHDL file.

Exercise 5 – Testing the Transmitter on the Board

Before testing your transmitter, you need to setup your workstation to accept data from the FPGA board. Follow the instructions in Part 4 of the screen cast to setup your workstation terminal.

Screencast: Putty – Part 4

The last step of this lab is to synthesize your design and test it on the board. Review these guidelines and instructions when preparing to test your design.

Verify that a serial cable is connected between the lab board and the computer. Run Putty on the computer. Whatever ASCII code you put on the switches should be printed to the screen when you press button 0 (the right most button). If it works, then get passed off by a TA.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Figure 7.1: ASCII Table.

Personal Exploration

For your personal exploration, modify the top-level design to add some additional feature. Ideas include:

- Modify the top-level design to print a predetermined message by sequentially sending a set of characters (i.e., send ‘B’, ‘Y’, ‘U’, ‘ ’, ‘C’, ‘o’, ‘u’, ‘g’, ‘a’, ‘r’, ‘s’)
- Try a different baud rate and see if you can get the transmitter to work at a different speed

Pass Off

Demonstrate the following to a TA to passoff your lab:

- Show your simulation passing the testbench in Exercise #2
- Demonstrate to the TA a working circuit on your board. The TA will experiment with your circuit to see if it operates correctly under a number of circumstances (i.e. the TA will test the transmission of several different characters)