CDA 4203L

Computer System Design Lab

Lab 6 Report

Loopback System Using Picoblaze

|  |  |
| --- | --- |
| Today’s Date: | 03/20/2022 |
| Team Members: | Ahmed El Maliki – U18019330 |
| Stella Kariuki – U 19511653 |
| Ivan Gonzalez – U15904968 |
| Work Distribution: | Ahmed El Maliki– Worksheet 2 & edited UCF file  Ivan Gonzalez – Worksheet 1 & edited rs232 uart.v  Stella Kariuki – Assembly codes(cold\_start, led\_echo, rs232\_echo)  DEMO LINK: <https://youtu.be/rHbD2-wnN2E> |
| No. of Hours Spent: | 28 hrs |
| Exercise Difficulty:  (Easy, Average, Hard) | Hard |
| Any Other Feedback: | It would have been helpful if we had a review before the lab was assigned. Picoblaze was introduced almost at the beginning of the semester, and it was kinda difficult to re-call the information |

0.1. Answers for Worksheet 1.

1. What is the instruction bus width for the ROM?

**18 bits**

1. For the UART 16x bit rate counter, the maximum baud count using the following formula:

*max\_baud\_count = fclk/ (16\*req\_baud\_rate)*

If fclk = 100MHz and the required baud rate is 9600, what is the maximum baud count?

**Max baud count = 100MHz/(9600\*16)**

**= 100\*10E6 / 153600**

**= 651 approx**

1. Write the boolean expression for the write to uart signal.

**write\_to\_uart = pb\_write\_strobe & (pb\_port\_id == 8’h03)**

1. Write the boolean expression for the write to leds signal.

**write\_to\_leds = pb\_write\_strobe & (pb\_port\_id == 8’h01)**

1. What are the port\_id values for the following input ports?
   1. switches: **8’h00**
   2. data\_rx: **8’h02**
   3. data\_present: **8’h04**
   4. buffer\_full: **8’h05**
2. Write the boolean expression for the read\_from\_uart signal.

**Read\_from\_uart = pb\_read\_strobe & (pb\_port\_id ==8’h04)**

1. Continue with step 3 (b) on Page 2 above.

0.2. Answers for Worksheet 2.

1. Clock (100MHz): LOC = D11
2. Push Button BTN0 (for reset signal): LOC = E6
3. LED 0: LOC = W3
4. LED 1: LOC = Y4
5. LED 2: LOC = Y1
6. LED 3: LOC = Y3
7. LED 4: LOC = AB4
8. LED 5: LOC = W1
9. LED 6: LOC = AB3
10. LED 7: LOC = AA4
11. SW 0: LOC = V5
12. SW 1: LOC = U4
13. SW 2: LOC = V3
14. SW 3: LOC = P4
15. SW 4: LOC = R4
16. SW 5: LOC = P6
17. SW 6: LOC = P5
18. SW 7: LOC = P8
19. RS232 TX (FPGA\_SERIAL1): LOC = T19
20. RS232 RX (FPGA\_SERIAL1): LOC = T20

1. Briefly explain how the design works. Include assembly code of **cold\_start (Code 1).** *Use as many pages as needed.*

The overall idea of this code was to output a message to the serial ports. The message was “Welcome to loop-back program!” and we needed to encode the message in ASCII format.

We loaded s0 and then we would load the ASCII into S0 and call wait which would send the character and check if the buffer is full. We sent every single character of our file to the serial port which would display the message on the PuTTy terminal.

JUMP cold\_start

wait:

INPUT s4, buffer\_full

COMPARE s4, 01

JUMP z, wait

RETURN

cold\_start: ; Code 1

; Write code to output a message to the

; serial port. The message must be longer

; than 25 characters.

LOAD s0, s0 ; (nop)

CALL wait

LOAD s0, ascii\_W;ASCII "W".

OUTPUT s0, uart\_data\_tx

STORE s0, 00

CALL wait

LOAD s0, ascii\_e ;ASCII "e".

OUTPUT s0, uart\_data\_tx

store s0, 01

CALL wait

LOAD s0, ascii\_l ;ASCII "l".

OUTPUT s0, uart\_data\_tx

store s0, 02

CALL wait

LOAD s0, ascii\_c ;ASCII "c".

OUTPUT s0, uart\_data\_tx

store s0, 03

CALL wait

LOAD s0, ascii\_o ;ASCII "o".

OUTPUT s0, uart\_data\_tx

store s0, 04

CALL wait

LOAD s0, ascii\_m ;ASCII "m".

OUTPUT s0, uart\_data\_tx

store s0, 05

CALL wait

LOAD s0, ascii\_e ;ASCII "e".

OUTPUT s0, uart\_data\_tx

store s0, 06

CALL wait

LOAD s0, ascii\_SPACE ;ASCII " ".

OUTPUT s0, uart\_data\_tx

store s0, 07

CALL wait

LOAD s0, ascii\_t ;ASCII "t".

OUTPUT s0, uart\_data\_tx

store s0, 08

CALL wait

LOAD s0, ascii\_o ;ASCII "o".

OUTPUT s0, uart\_data\_tx

store s0, 09

CALL wait

LOAD s0, ascii\_SPACE ;ASCII " ".

OUTPUT s0, uart\_data\_tx

store s0, 0A

CALL wait

LOAD s0, ascii\_L ;ASCII "L".

OUTPUT s0, uart\_data\_tx

store s0, 0B

CALL wait

LOAD s0, ascii\_o ;ASCII "o".

OUTPUT s0, uart\_data\_tx

store s0, 0C

CALL wait

LOAD s0, ascii\_o ;ASCII "o".

OUTPUT s0, uart\_data\_tx

store s0, 0D

CALL wait

LOAD s0, ascii\_p ;ASCII "p".

OUTPUT s0, uart\_data\_tx

store s0, 0E

CALL wait

LOAD s0, ascii\_MINUS;ASCII "-".

OUTPUT s0, uart\_data\_tx

store s0, 0F

CALL wait

LOAD s0, ascii\_b ;ASCII "b".

OUTPUT s0, uart\_data\_tx

store s0, 10

CALL wait

LOAD s0, ascii\_a ;ASCII "a".

OUTPUT s0, uart\_data\_tx

store s0, 11

CALL wait

LOAD s0, ascii\_c ;ASCII "c".

OUTPUT s0, uart\_data\_tx

STORE s0, 12

CALL wait

LOAD s0, ascii\_k ;ASCII "k".

OUTPUT s0, uart\_data\_tx

STORE s0, 13

CALL wait

LOAD s0, ascii\_SPACE ;ASCII " ".

OUTPUT s0, uart\_data\_tx

store s0, 14

CALL wait

LOAD s0, ascii\_p ;ASCII "p".

OUTPUT s0, uart\_data\_tx

store s0, 15

CALL wait

LOAD s0, ascii\_r ;ASCII "r".

OUTPUT s0, uart\_data\_tx

store s0, 16

CALL wait

LOAD s0, ascii\_o ;ASCII "o".

OUTPUT s0, uart\_data\_tx

store s0, 17

CALL wait

LOAD s0, ascii\_g ;ASCII "g".

OUTPUT s0, uart\_data\_tx

store s0, 18

CALL wait

LOAD s0, ascii\_r ;ASCII "r".

OUTPUT s0, uart\_data\_tx

store s0, 19

CALL wait

LOAD s0, ascii\_a ;ASCII "a".

OUTPUT s0, uart\_data\_tx

store s0, 1A

CALL wait

LOAD s0, ascii\_m ;ASCII "m".

OUTPUT s0, uart\_data\_tx

store s0, 1B

CALL wait

LOAD s0, ascii\_EXCLAIM ;ASCII "!".

OUTPUT s0, uart\_data\_tx

store s0, 1C

2. Briefly explain design details. Include assembly code of **led\_echo (Code 2).** *Use as many pages as needed.*

For this code we needed to read the switch states and then rewrite it inverted to the LEDS . We loaded S0, disable interrupts and XOR s0 with FF to invert the bits. Then we would enable interrupts and output the bits to leds then jump into the next code.

led\_echo: LOAD s0, s0 ; (nop)

; Code 2

; Write code to read the switch state and

; write it, inverted, to the LED output port.

INPUT s0, switches

DISABLE INTERRUPT

XOR s0, FF

OUTPUT s0, leds

ENABLE INTERRUPT

JUMP rs232\_echo

3. Briefly explain design details. Include assembly code of **rs232\_echo (Code 3).** *Use as many pages as needed.*

This code is to check if a byte has been received in the uart and if so it would write it back to the uart transmit port then it would return back to led\_echo.

rs232\_echo: LOAD s0, s0 ; (nop)

; Code 3

; Write code to check if a byte has been

; received by the uart. If so, write it

; back to the uart transmit port. Then...

NAMEREG sF, dr

NAMEREG sE, Data

BEGIN1:

INPUT dr, data\_present

TEST dr, 01

CALL C, SEND

JUMP BEGIN1

SEND:

INPUT Data, uart\_data\_rx

COMPARE s0, buffer\_full

OUTPUT Data, uart\_data\_tx

JUMP led\_echo ; endless loop

4. Pin Mapping file and synthesis report. *Use as many pages as needed.*

PIN MAPPING FILE

**Graphical user interface, application, table

Description automatically generated**

SYNTHESIS REPORT

It will be in the submission file as synthesis.txt