EL 6463: Advanced Hardware Design 2021

Final EXAM (20%) DUE: December 24th 2021, 12PM NOON ET

Instructions

- Read and complete **all** questions carefully.
- This exam is to be completed **individually.** Collaboration/plagiarism will not be tolerated.
- Please sign the code of conduct (located at the bottom of this exam) and include it in your submission.
- Clarification questions can be posted on Slack in #finalexam, but do not put any specifics about your own answers in the channel.
- During the exam period, you will not be allowed to receive assistance from the teaching staff regarding the use of the tools or implementation/simulation.

Overview: In this examination you will design, implement, and simulate digital hardware with sequential and combinational logic to implement a *pseudo-random number generator* algorithm. This algorithm will be implemented via hardware known as a *linear feedback shift register (LFSR).* These have applications in generating simple randomness in gaming, cryptography, audio and signal processing.

An example of a 16-bit LFSR is shown below:

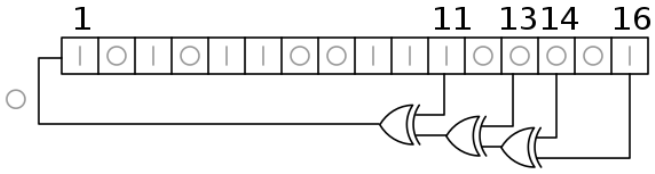

Image courtesy: CC0 https://en.wikipedia.org/wiki/File:LFSR-F16.svg

The depicted LFSR has XOR "taps" at positions [16, 14, 13, 11] (or zero-indexed: 15, 13, 12, 10).

In each clock cycle, the rightmost tap (which is known as the output bit) is XOR'd sequentially with the next right-most taps and the result is "shifted in" to the left-most bit (the most significant bit (MSB)).

In this final exam you will implement an LFSR with the following specifications (essential features).
Note: this is the same base specification as what was used in the mid-term exam.

1. It will be 32 bits long (from position 0 (MSB) to 31 (LSB)), with taps at positions as determined:
    a. Each of the unique adjacent 2-digit numbers in your N number modulo 32.
    b. The 32 bit (position 31, the LSB).
    c. For example, if your N number is 01123497, you will have taps at positions
        i. [01, 11, 12, 23, (34 % 32 =) 2, (49 % 32 =) 17, ~~(97 % 32 =) 1~~] *("1" value is not unique)*
        ii. = [1, 2, 11, 12, 17, 23, 31] *(31 comes from point 1b)* - remember these are zero-indexed
2. Top I/O:
    a. clk: input
    b. rstn: active-low reset signal
    c. data_out: 8 bit output
    d. get_random: input 1 bit control signal
    e. 1-bit input "load_seed"

f. 8-bit input “data_in”.

3. Unless otherwise specified, the LFSR will update (shift-in the XOR result) upon every clock cycle. The bits shift into position 0 (the MSB).
4. If “output” is requested by setting “get_random” to 1, the least significant 8 bits will be output to “data_out” in the next clock cycle, the next 8 in the next clock cycle, the next 8 in the next cycle, and the most significant 8 bits will be output in the final (fourth) cycle. No LFSR shifts will occur during this output sequence. At all other times the “data_out” will be all zeros.
    a. **NOTE:**
        i. **If the get_random signal is held high for longer than 4 cycles then the LFSR would presumably immediately begin exporting the contents again, and so it wouldn't perform any shifts, however there might be one clock cycle where the internals could shift. This part of the spec is upto to you.**
        ii. **Similarly if get_random is high for less than 4 cycles, it should continue with the output until it is completed.**
5. Reset signal “rstn” will reset the LFSR contents to 0x02468ACD
    a. **(Reset signal “rstn” should always take precedence over every other signal)**
6. If “load_seed” is set to 1, the lower 8 bits will be loaded into the LFSR from “data_in” in the first clock cycle, the next 8 in the next, the next 8 in the next, and the upper 8 bits will be loaded in the final (fourth) cycle. No LFSR shifts will occur during loading. After loading, shifts will resume as normal.

Questions

Problem A: Complete the following: [6 points total]

(1) **Draw** your intended high-level FSM + Datapath of your LFSR. **[1 point]**

(a) hint: You may re-use your midterm diagram if you think it is good enough…

(2) **C simulation, synthesize, and C/RTL co-simulation (functional)** your design in Vivado HLS (C language). Write C code for the LFSR and the testbench. Comment well. **[5 points]**

(a) Your design may be architectured in any reasonable way. The C codes should be synthesizable in Vivado. **[1 point]**

(b) **Write** a thorough C testbench to demonstrate the capabilities of your C specification and to do C/RTL Co-simulation. Use at least 1000 random test vectors. Test **for meaningful corner cases**. Justify the corner cases. **[2 points]**

(c) When reporting timing and area results use the same part number that you used in the mid term. This way, you can compare the results of HLS-based design (Final Exam) and Gate-level design (Mid term). If your Midterm part number is not available in Vivado HLS, use the part #xc7a200tfbv676-2L **[2 points]**

Problem B: Complete the following: [2 points total]

(1) Choose two of the parameters/pragmas that you were taught in class, and experiment with them in your code. Record the effects. Create a short 1 page document which discusses each parameter

you chose, the effect it is supposed to have, and the effects it does have when applied to your code. **[1 point per parameter discussion]**

Problem C: Supplementary Report [2 points total] (2 page text maximum, figures may add more pages)

(1) Describe your design process. Justify your code and relate it to your desired FSM + datapath design. **[0.5 point]**

   (a) Include your diagram from A(1)

      (i) The diagram is worth the additional/separate 1 mark from A(1).

(2) Present the resource utilization of your HLS design, including the number of total register bits. Identify the largest consuming component(s). **[0.25 points]**

(3) Discuss the difference in implemented design and resource consumption between your original mid-term and the HLS equivalent. You may wish to include diagrams. **[0.5 points]**

(4) Present and discuss your HLS C simulation. Justify your testbench (where did test values come from?), and describe how it covers all possible execution scenarios. **[0.5 points]**

(5) Present and discuss your C/RTL Co-simulation. Include screenshots. **[0.25 points]**

Deliverables:
Submit a zip file with your VHDL or Verilog files and Xilinx Project and a PDF document containing your answers to Problems B and C. **Include screenshots of your RTL-C co-simulation.**

Student Exam Code of Conduct

I certify and affirm that,

1. I am aware of the NYU Code of Conduct.
2. Work presented is 100% my own. I have not collaborated with anyone on the test.
3. I did not misrepresent someone else's work as my own.
4. I did not discuss, nor in any way divulge the content of this test or my answers.
5. I understand that my failure to abide by this code of conduct will result in consequences outlined in the NYU Code of Conduct.

Name: _____

Signature: _____

Date: _____