Assignment 1

Welcome to PyRTL[1]

PyRTL Installation:

Installation of PyRTL is very easy. The following is from the manual written by one of the authors and pioneers of this fantastic project, Dr. Tim Sherwood, UC Santa Barbara.

To install this package you can issue this command in terminal in all three major operating systems Windows, Mac and Linux:

pip install pyrtl

PyRTL is listed in PyPI and can be installed with **pip** or **pip3**. If the above command fails due to insufficient permissions, you may need to do **sudo pip install pyrtl** (to install as superuser) or **pip install --user pyrtl** (to install as a normal user). PyRTL is tested to work with Python 3.8+.

Using the Template (skeleton file)

You will develop your project on the skeleton file which has the necessary parts for your project to work and you build up on that. The files exist in the repository named **lfsr_combinatorial.py** and **lfsr_behavioral.py**.

A Few Words About LSFR

LFSR or Linear Feedback Shift Register is a digital circuit that is used for generation of pseude-random numbers. The generated numbers are not completely random but psuedo-random as this circuit repeats the generated numbers after a very long time. These two links give good information on LFSR:
From Wikipedia.org

https://en.wikipedia.org/wiki/Linear-feedback_shift_register

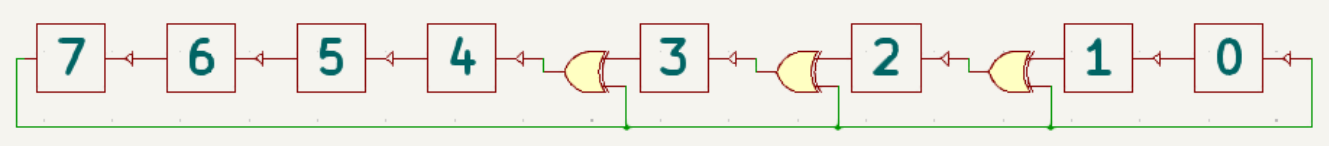
From eetimes.com

https://www.eetimes.com/tutorial-linear-feedback-shift-registers-lfsrs-part-1/

1 This assignment is inspired by CSE x25 - Assignment 1 used in UC Santa Cruz

Assignment 1

Below is a diagram of an LFSR. Note that the boxes with numbers inside them are sub-modules that you have to implement the way that the whole LFSR can be initialized to a predetermined number upon Initialization and keeps on generating pseudo-random numbers in the output of each box. The arrows indicate the data flow. The numbers inside the boxes are the bit position for this 8 bit LFSR.



PART 1

LSFR implementation in Combinatorial Logic

In this part, you will implement the LFSR in a combinatorial way. Although it's not a practical way to do it in PyRTL, because PyRTL is made to use abstraction. You will come out of this part of the lab appreciating the abstraction and see how abstraction makes designing digital circuits much easier.

Create the necessary modules separately and then wire them up together. Your circuit should have means by which the LSFR can be initialized to a predetermined number so that it starts generating pseudo-random numbers starting with that number. The bits of the generated 8-bit numbers are tapped from the output of the boxes. Use the skeleton file **combinatorial.py** and build up your LFSR using that.

PART 2

LFSR implementation in Behavioral Logic

In this part, you will develop the LSFR in a behavioral way. Use the skeleton file **behavioral.py** and build up your LSFR using that.

Helpful hint to test your circuit even without a test-bench

Once you are done with the implementation of both combinatorial and behavioral LFSRs, submit your files along with the lab write up and please include a snapshot of PyRTL terminal waveform viewer which is initialized by the number "A5", in which case the second generated number should be "57". In order to check if your circuit works, it's very common to develop a test bench for the design and this part is a Must, because you want to make sure your circuit is working properly. But

Assignment 1

here is an easy way to double check the proper working on your circuit.

Once you initialize your circuit with the number A5, your circuit should generate the following number sequence until re-initializing the LFSR by asserting the “init” input while initializing with a number in the sequence and the circuit should generate the numbers to the right of the initializing number in the sequence. Here is the set of outputs results that should be generated by your LFSR:

{A5, 57, AE, 41, 82, 01, 32, 64, C8, 8D, 07, …}

So for example if you initialize the circuit with the number 41 in this sequence, you should have the following number as the first number generated by the circuit:

{82, 01, 32, 64, C8, 8D, 07, …}

This is the reason why LFSR is called a Pseudo-random number generator, because the generated numbers are not truly random and they follow a pattern but since this circuit repeats itself after a long time, it seems that the generated numbers are random. You can generate this sequence yourself for further testing and learning about LFSR. First pick any 8-bit number you like as initializing number, then follow the logic of the circuit as the bits are getting shifted and Xor’ed and you get the numbers that are expected to be generated in next cycle. You will see the circuit should generate the numbers you came up with, after initializing the circuit with your particular number.