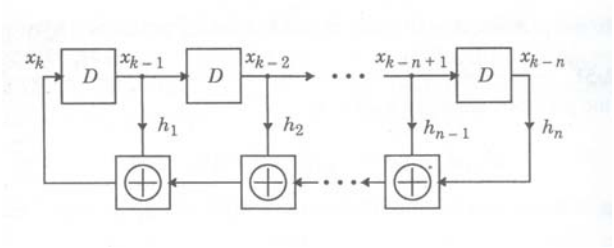
EEL319 Digital Signal Processing
Experiment 3: PN Sequence Generation

Theory: Pseudo-Noise (PN) sequences are commonly used to generate noise that is approximately "white". It has applications in scrambling, cryptography, and spread-spectrum communications. It is also commonly referred to as the Pseudo-Random Binary Sequence (PRBS). These are very widely used in communication standards these days. The qualifier "pseudo" implies that the sequence is not truly random. Actually, it is periodic with a (possibly large) period, and exhibits some characteristics of a random white sequence within that period. PN sequences are generated by Linear Feedback Shift Registers (LFSR), as shown in the following figure:



In the figure, the output x_k is binary (0 or 1), and so are the constants h_j, j=0,1,…n, and $\oplus$ denotes the XOR operation. This means that x_k is given by:

$$x_k = h_1x_{k-1}\oplus\ldots\ldots\oplus h_nx_{k-n}$$

Since $x_k\oplus x_k=0$, it follows from the above that:

$$x_k \oplus h_1x_{k-1}\oplus\ldots\ldots\oplus h_nx_{k-n} =0$$

or

$$h(D)x(D) = 0$$

where $h(D) = 1 \oplus h_1D\oplus\ldots\ldots\oplus h_nD^n$ and D denotes a unit delay ($x_kD^j=x_{k-j}$ for any j). Note that the "1" in the polynomial does not correspond to a tap.

It is not difficult to see that the output x_k will be periodic. However, the dependence of the length of the period (which we would like to be as large as possible) on the constants h_j, j=1,2,…n is not obvious. We can see that the "state" (x_{k-1} …. x_{k-n}) can assume at most 2^n values. We note the following:

- If the state of the shift register is all zero at any time, it remains so for all time. We need to ensure that this never happens (we start with a non-zero value).
- If the state ever remains unchanged from one clock cycle to the next, it remains the same forever.
- The sequence must be periodic (since there are at most 2^n -1 states).
- Since all the all zero state is not allowed, the period of the output sequence can be at most 2^n-1. A feed-back shift register that generates a sequence of this period is said to be of *maximal length.*

How do we design the feed-back shift register (i.e., h_j) so that it is maximal length, keeping the hardware (XORs) minimum? The answer to this question involves the divisibility of $1\oplus D^m$ by h(D) for $m<2^{n-1}$, and need not concern us here. The following table lists the tap-points (points where h_j is 1) for registers of various sizes. The first column lists the order n of the register, and the second column lists the tap points or h(D) in octal notation. For example, for a register of size 12, the table lists 10123 in octal or 1000001010011 in binary which means that $h(D) = 1\oplus D\oplus D^4\oplus D^6\oplus D^{12}$ (this means that the first, fourth, sixth, and the 12^{th} points in the shift register are tapped for XORing).

n=2	7	14	42103	26	400000107
3	13	15	100003	27	1000000047
4	23	16	210013	28	2000000011
5	45	17	400011	29	4000000005
6	103	18	1000201	30	10040000007
7	211	19	2000047	31	20000000011
8	435	20	4000011	32	40020000007
9	1021	21	10000005	33	-
10	2011	22	20000003	34	-
11	4005	23	40000041	35	-
12	10123	24	100000207	36	-
13	20033	25	200000011		

Minimal weight polynomials of order 2 to 32 are listed in the above table. Each entry in the table is an octal number for each n in the first column, which when converted to binary specifies the coefficients of h(D). The most significant (leftmost) bit h_n is 1, and so is the least-significant (right-most) bit h_0. . You can find the tap points for larger length shift registers in various sites on the internet (use this hyperlink for example)

If you look at any n-length segment of the output, you will find all possible sequences with the exception of the all zero sequence. However, when you look at a smaller segment, you will come across all possible sequences. Intuitively therefore, when the register length is large, the sequence is approximately "white". Consider for example the case when n=2 and $h_1=h_2=1$ implying that both points are tapped so that $x_k=x_{k-1}\oplus x_{k-2}$. Starting with $x_{k-1}=0$ and $x_{k-2}=1$ gives the following sequence of shift register contents: (0,1),(1,0),(1,1),(0,1),… Notice that periodicity is three since $2^2-1=3$ and this happens to be the tapping to get a maximal length sequence. Note that the shift-register contents are shifted versions of each other and it makes no difference which register output is considered the output.

The output of the PN sequence generator is purely deterministic – given the state of the generator, the output is uniquely determined for all time. With the zero level mapped to a "-1" to make it an antipodal sequences, the autocorrelation of the maximal-length PN sequence is periodic, and its value in one period is -1/M except at one location where it is 1 (M is 2^n-1). This sequence can be filtered to generate bandlimited Gaussian-like noise.

Experiment: You are asked to design a PN sequence generator of length *n*. The length *n* is stored in a memory location. . So are the tap points h_j (for example, 1 4 6 and 12 may be written in sequential memory locations for the example considered above). You can store h_j and *n* in a different format if you like. Try to make the program as general as possible, so that it can work for large n. Program should work for n≤32 at least, though I would recommend that you try to make the shift register length larger. How many clock cycles does your program take to generate one bit of the output?

Hints: Note that you can rotate/shift through carry bit. Go through the ROR and ROL instructions. Note that you can test for the carry bit in many ways. It may not be a bad idea to use BCNT instruction. The reason is that the number of tap-points is relatively small compared to the register length, and it may be wasteful to shift/rotate so many times. Should your counter be in the LSBs or in the MSBs of the accumulator? Note that you are not doing arithmetic, but doing logical operations. For this reason, you might want to turn off sign extension (BCLR SXM).