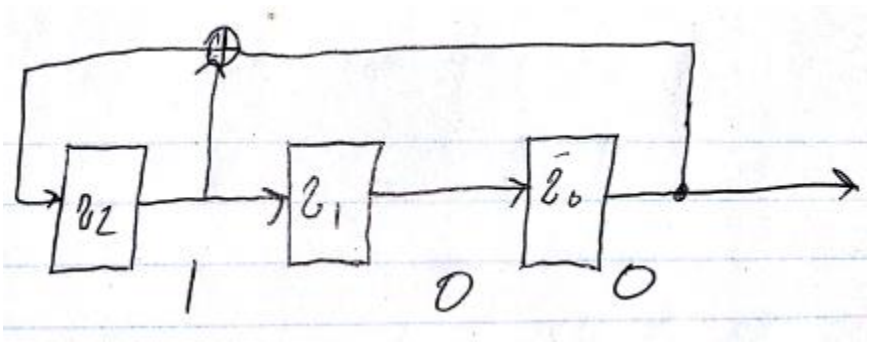
Jeremy Barthélemy
ECE 746
Hw. 6

1.



Case 1

z2	z1	z0
1	0	0
1	1	0
1	1	1
0	1	1
1	0	1
0	1	0
0	0	1
1	0	0

The sequence generated is: 00111010011101001110100 11….

Case 2

z2	z1	z0
0	1	1
1	0	1
0	1	0
0	0	1
1	0	0
1	1	0
1	1	1
0	1	1

The sequence generated is: 1101001110100111010 01….

The two sequences are the same but they start at a different initialization (i.e. they are shifted).

2.

a.

Case 1

1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
1	0	0	0

$x^4 + x + 1$ is a primitive polynomial, and we can see that it generates a maximum length sequence of:
000100110101111

b.

Case 1

1	0	0	0
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
0	0	0	1
1	0	0	0

Case 2

0	1	1	1
0	0	1	1
1	0	0	1
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1

Case 3

0	1	1	0
1	0	1	1
1	1	0	1
0	1	1	0

The lengths of the sequences generated are indeed equal to $2^m-1 = 2^4-1 = 15$.
For this case, our LSFR does not generate a maximum-length sequence, and the sequence length depends upon the initialization values for each register, thus our LSFR $x^4 + x^2 + 1$ is a reducible polynomial.

c.

Case 1

1	1	1	1
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0
1	1	1	1

Case 2

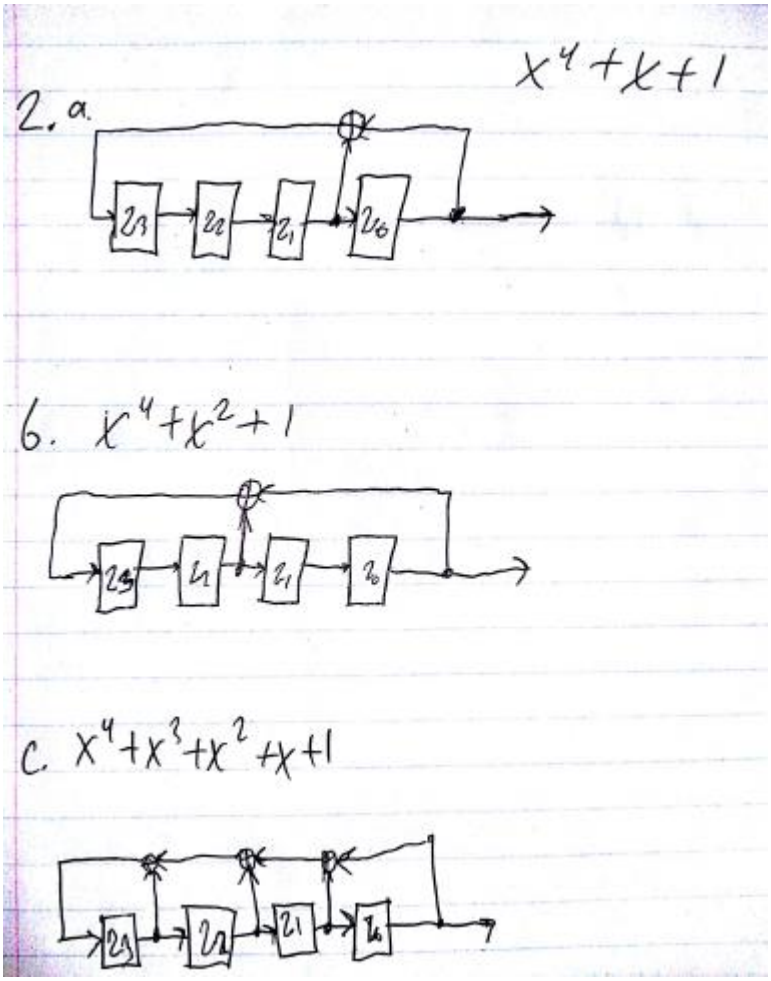
0	0	0	1
1	0	0	0
1	1	0	0
0	1	1	0
0	0	1	1
0	0	0	1

Case 3

0	0	1	0
1	0	0	1
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0

This is an irreducible polynomial that is not primitive, thus why we have sequences of size 5 (5+5+5) = $2^m-1 = 15$, as expected.

Below are the LFSR schematics for the respective polynomials:



3.

Taking the plaintext and the stream obtained from the channel, we obtain the following:

1001 0010 0110 1101 1001 0010 0110

<u>1011 1100 0011 0001 0010 1011 0001</u>

0010 1110 0101 1100 1011 1001 0111

The sequence generated by the LFSR is: 0010111, with a period of 7 (2^3-1 = 7), thus we have m = 3.

a.

The degree m of the stream generator is 3.

b.

The initialization vector is 001.

c. Z_{i+m} = sum(c_jZ_{i+j} mod2 for j = 0 to m - 1),

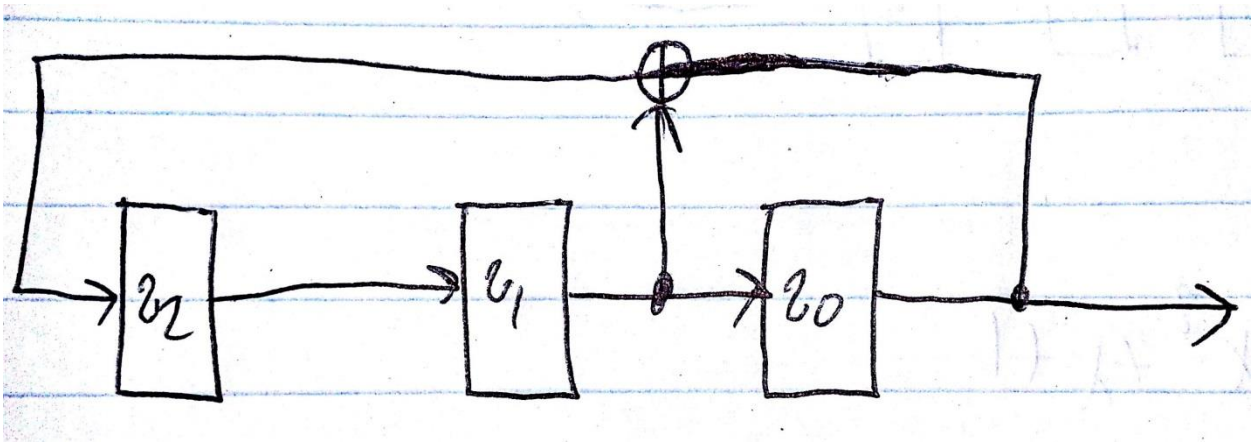
i = 0 → Zm = ($c_0z_0 + c_1z_1 + c_2z_2$) mod 2

i = 1 → z_{m+1} = ($c_0z_1 + c_1z_2 + c_2z_3$) mod 2

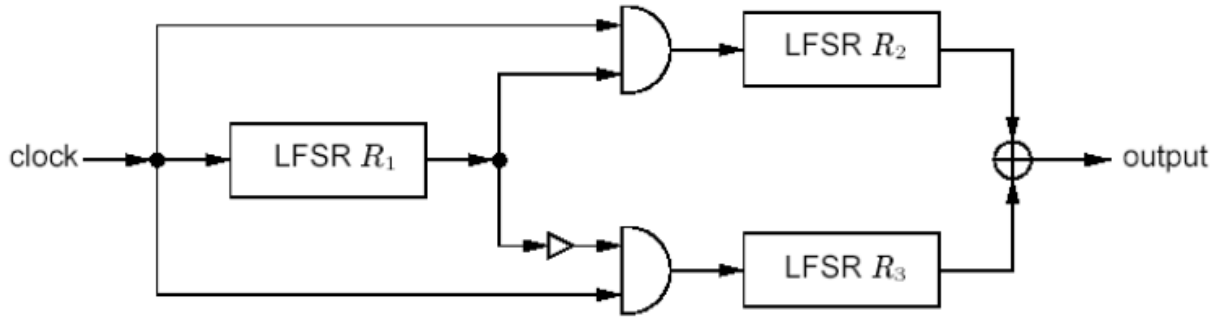
i = 2 → z_{m+2} = ($c_0z_2 + c_1z_3 + c_2z_4$) mod 2

$z_m = (c_0*0 + c_1*0 + c_2*1) \bmod 2 = c_2 \bmod 2 = 0$
$z_{m+1} = (c_0*0 + c_1*1 + c_2*0) \bmod 2 = c_1 \bmod 2 = 1$
$z_{m+2} = (c_0*1 + c_1*0 + c_2*1) \bmod 2 = c_0 + c_2 \bmod 2 = 1$
$c_2 = 0$, $c_1 = 1$, $c_0 = 1$

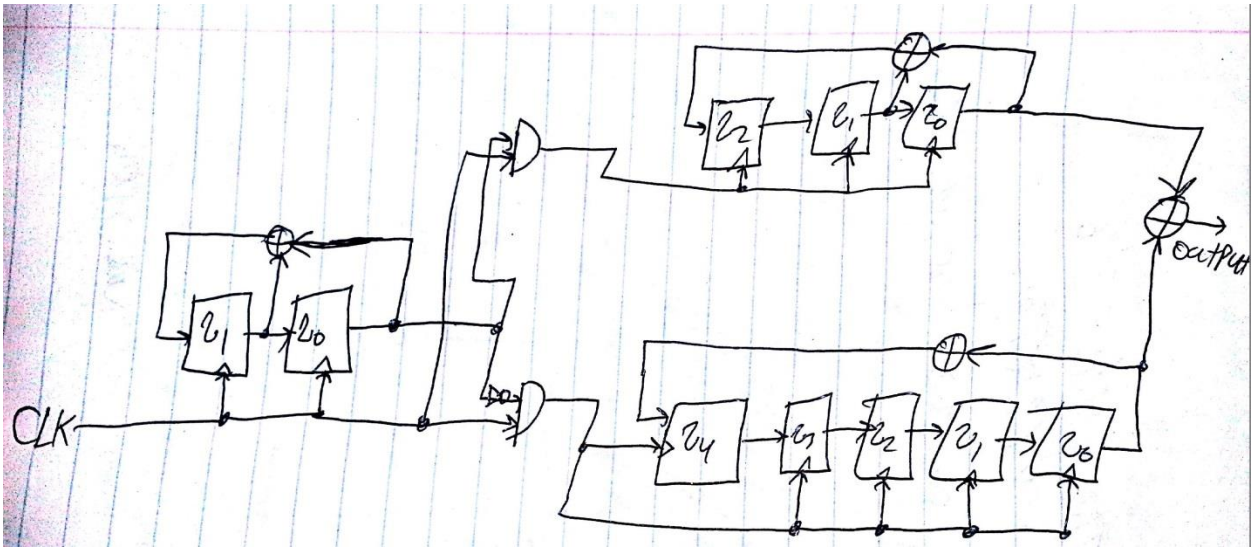
d.



4.
60 seconds * 60 minutes * 24 hours = # seconds in a day = 86400
With R = 155Mbps = $155*2^{20}$ bps = 162529280 bps
Given 24 hours, we have 86400*162529280 bits transferred.
The minimum degree must then be such that we repeat >= 86400*162529280
2^m-1 >= 86400*162529280
We plug in m until we have a positive value from 2^m-1 – 86400*162529280, and find that the minimum degree for our LFSR under these conditions is m = 44!
5.
a.
Taking directly from the slides, a high-level view of our circuit will look like this:



With a more detailed view based upon the polynomial description of each LFSR, our circuit will look like this:



b.

The sequences generated by the three LFSRs are as follows:

LFSR1

z1	z0
0	1
1	0
1	1
0	1

LFSR2

z2	z1	z0
0	0	1
1	0	0
0	1	0
1	0	1
1	1	0
1	1	1
0	1	1
0	0	1

LFSR3

z4	z3	z2	z1	z0
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
1	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0	0	1	1	0
1	0	0	1	1
1	1	0	0	1
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
1	0	0	0	1
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1
0	1	1	1	0
1	0	1	1	1
0	1	0	1	1

1	0	1	0	1
0	1	0	1	0
0	0	1	0	1
0	0	0	1	0
0	0	0	0	1

Given these values, we can feed in respective bits from LFSR1 to produce the final output, where LFSR1 bits are used with the clock to trigger the clock inputs to the other LFSRs:

	LFSR1		LFSR2			LFSR3					Output
Cycle	z1	z0	z2	z1	z0	z4	z3	z2	z1	z0	(XOR)
0	0	1	0	0	1	0	0	0	0	1	0
1	1	0	1	0	0	0	0	0	0	1	1
2	1	1	1	0	0	1	0	0	0	0	0
3	0	1	0	1	0	1	0	0	0	0	0
4	1	0	1	0	1	1	0	0	0	0	1
5	1	1	1	0	1	0	1	0	0	0	1
6	0	1	1	1	0	0	1	0	0	0	0
7	1	0	1	1	1	0	1	0	0	0	1
8	1	1	1	1	1	0	0	1	0	0	1

The output is thus: 0, 1, 0, 0, 1, 1, 0, 1, 1 for the first 8 bits (including the initial reset cycle).

c. The sequence lengths of the respective LFSRs is 3, 7, and 31.
gcd(3, 7) = gcd(3, 31) = gcd(7,31) = 1, thus the condition is satisfied.
The total output sequence generated would be 3*7*31 = 651 bits long.

6. For a one-time pad with a repeating key, the perfect secrecy of this crypto scheme is compromised. For example, if we are able to determine the 1000 bit key, we will then be able to read all of the associated plaintext! With a repeating key, we can perform frequency analysis or pattern matching (known plaintext could aid in this, or if we do plaintext-ciphertext known pairs) to discern the key, and then take these 1000 bits to be able to read all plaintext that is sent afterward.

7.

```
#Read in SN, determine n

S_array = [0, 0, 1, 1, 0, 1, 1, 1, 0]

n = len(S_array)

#Initializations

var('D')
```

```
CD = 1;L = 0;m = -1;BD = 1;N = 0;

ci_array = 9*[0]

SN = 0; TD = 0;

print "sN, d, T(D), C(D), L, m, B(D), N";

while (N < n):

   print(S_array[N],  d,   TD,  CD,  L,  m,  BD,  N+1);

   #d = (S_array[N] + sum(ci_array[i]*S_array[N-i],i,1,L)) % 2

   #d = S_array[-1]

   #d = (S_array[N]) % 2


   d = (S_array[N+1] + sum([ci_array[i]*S_array[N-i] for i in range(1,L)])) % 2


   if (d==1):

      TD = CD;

      CD = CD + BD*(D^(N-m+1));

      if(L <= (N/2)):

         L = N + 1 - L;

         m = N;

         BD = TD;

   N = N + 1;
   #print(S_array[N],  d,   TD,  CD,  L,  m,  BD,  N);
```