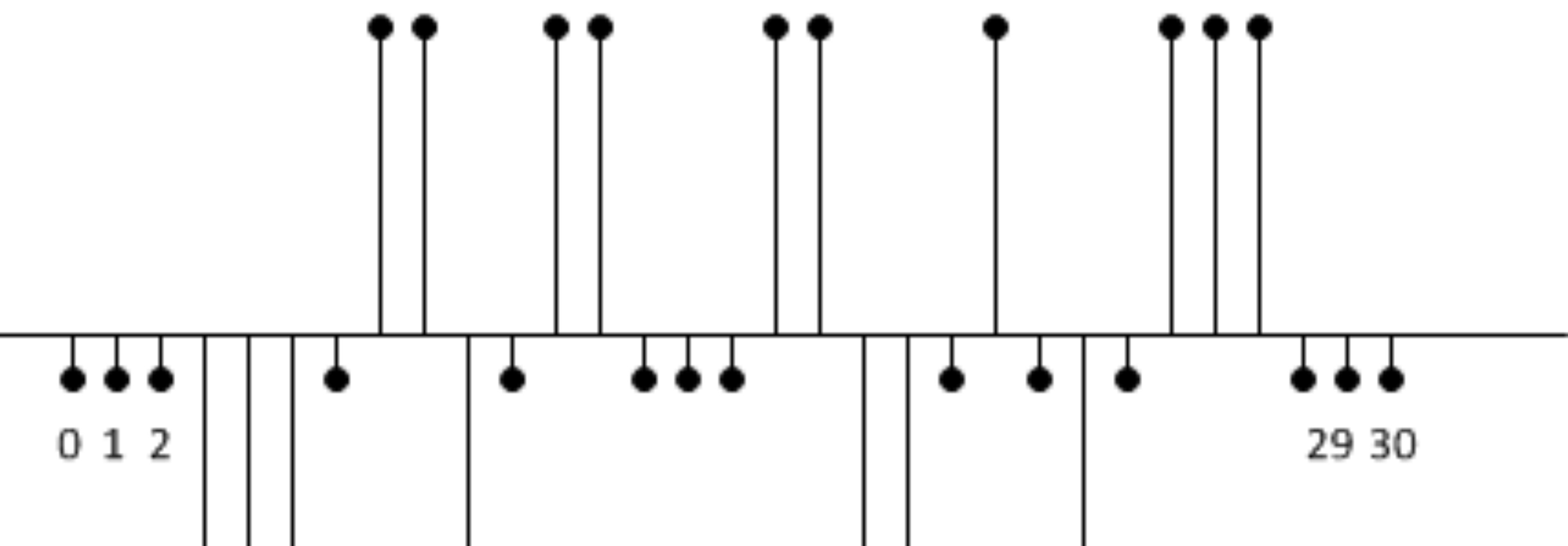
E11 – Autonomous Vehicles

Gold Codes



Outline

- Gold Code Overview
- Gold Code Generation
- Gold Code Detection
- Application

Overview

- Gold Codes are sequences of 0's and 1's
  - Invented by Dr. Robert Gold in 1967
  - Easy to generate in hardware or software
  - Have characteristics resembling random noise
  - Minimally jam other Gold codes transmitted by other sources
  - Commonly used in communications systems
    - Notably GPS and cell phones

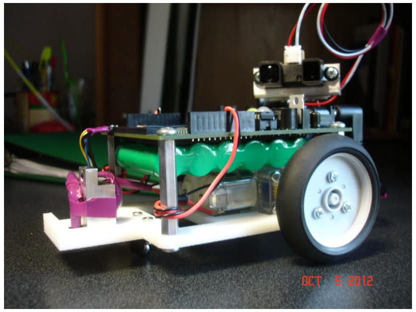
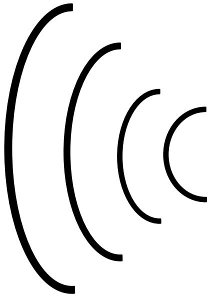
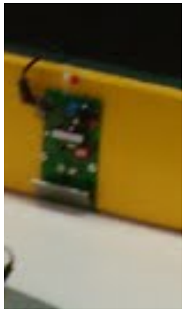
Applications

- GPS
  - Multiple satellites transmit information simultaneously at the same frequency
  - Receiver can pick out the signals from the individual satellites because each has a unique Gold code

www.gamin.com

Applications

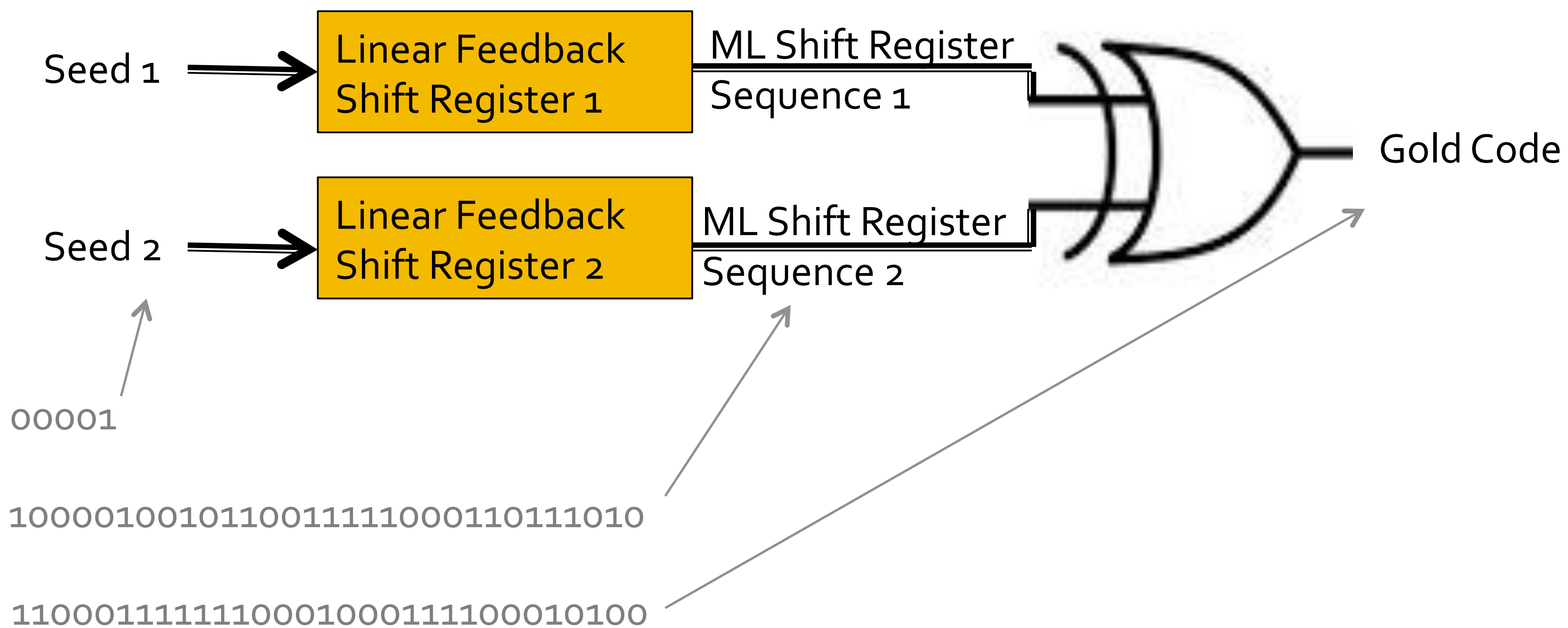
- E11 Bots
  - Your bot will seek beacons transmitting different Gold codes
  - Identify the desired beacon by recognizing its code
  - Progressively more difficult problem sets:
    - PS3: Gold Code Generation
    - PS4: Gold Code Correlation
    - PS6: Gold Code Detection

Outline

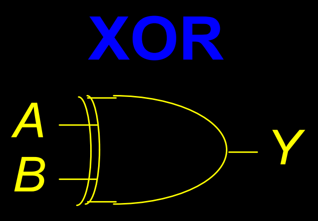
- Gold Code Overview
- Gold Code Generation
  - XOR gates
  - Shift Registers
  - Constructing MLSRS
  - Generating the GC
- Gold Code Detection
- Application

GC Generation



LFSRs

- XOR Gate
  - XOR of 2 inputs is TRUE if exactly one input is TRUE
  - XOR of many inputs is TRUE if an ODD # of inputs are TRUE

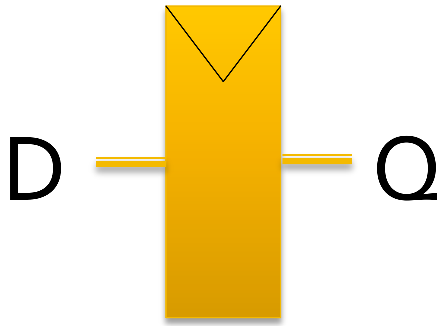


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

LFSRs

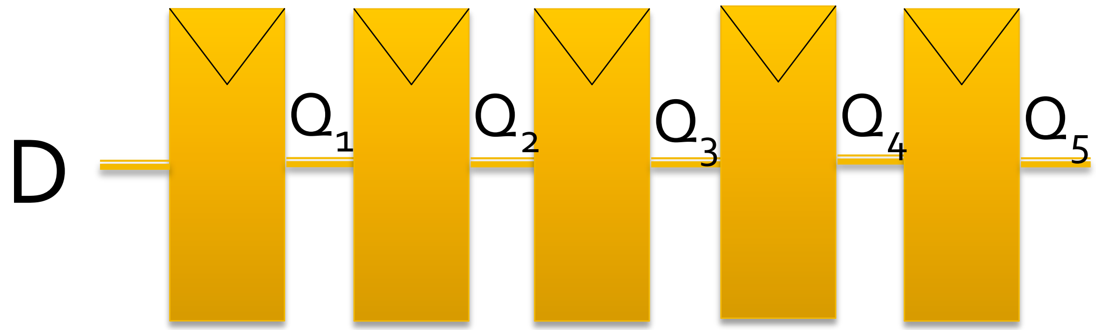
- Registers
  - A **register** copies its input D to its output Q on each time step

$$Q_t = D_{t-1}$$



LFSRs

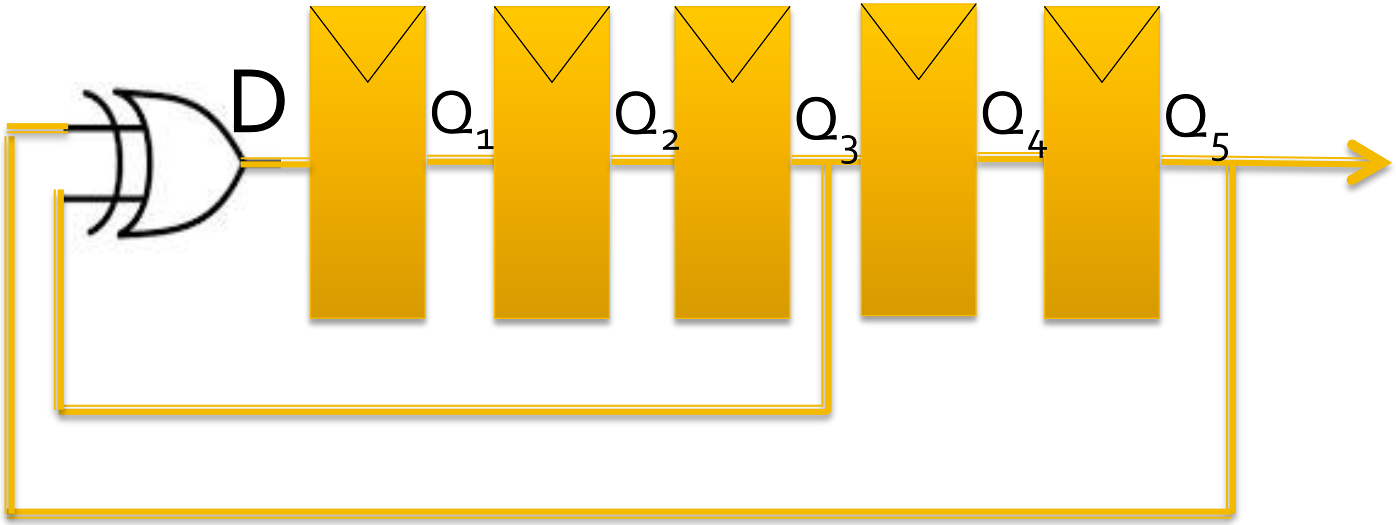
- Shift Registers
  - A **shift register** shifts all of its bits right each step



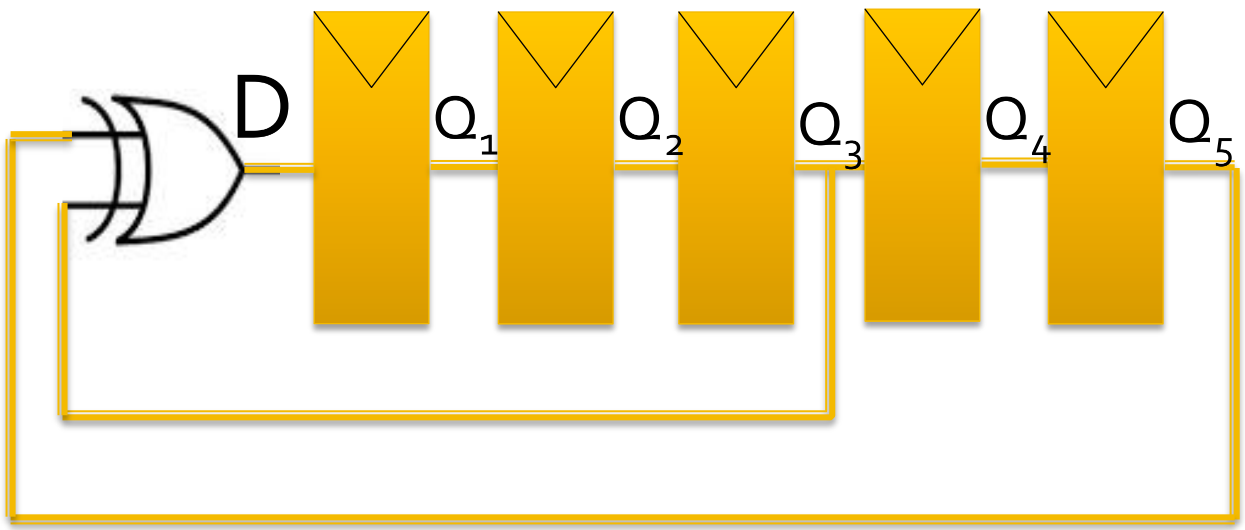
Step	Q1	Q2	Q3	Q4	Q5	D
0	0	0	0	1	0	1
1		?				1
2		?				1

LFSRs

- Linear Feedback Shift Register (LFSR)
  - An **LFSR** Feeds XOR of certain bits back to input D
  - The initial value of the bits (i.e. Q_i) is called the **seed**
  - Over time the output (e.g. Q_5) will be a bit **sequence**



LFSRs



Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1
1					
2					
3					
4					

LFSRs

- Seeds
  - The initial values of the LFSR are called the **seed**
  - Ex: 00001

Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	1	0	0	1	0
5	0	1	0	0	1

  - If the seed is all 0's, the state will never change!

LFSRs

- A Complete Sequence

Step	Q1	Q2	Q3	Q4	Q5	Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1	16	0	0	1	1	1
1	1	0	0	0	0	17	0	0	0	1	1
2	0	1	0	0	0	18	1	0	0	0	1
3	0	0	1	0	0	19	1	1	0	0	0
4	1	0	0	1	0	20	0	1	1	0	0
5	0	1	0	0	1	21	1	0	1	1	0
6	1	0	1	0	0	22	1	1	0	1	1
7	1	1	0	1	0	23	1	1	1	0	1
8	0	1	1	0	1	24	0	1	1	1	0
9	0	0	1	1	0	25	1	0	1	1	1
10	1	0	0	1	1	26	0	1	0	1	1
11	1	1	0	0	1	27	1	0	1	0	1
12	1	1	1	0	0	28	0	1	0	1	0
13	1	1	1	1	0	29	0	0	1	0	1
14	1	1	1	1	1	30	0	0	0	1	0
15	0	1	1	1	1	repeat	0	0	0	0	1

LFSRs

- Different seeds give **shifted** versions of the sequence

Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	1	0	0	1	0
5	0	1	0	0	1
6	1	0	1	0	0
7	1	1	0	1	0
8	0	1	1	0	1
9	0	0	1	1	0
10	1	0	0	1	1
11	1	1	0	0	1
12	1	1	1	0	0
13	1	1	1	1	0
14	1	1	1	1	1
15	0	1	1	1	1

Step	Q1	Q2	Q3	Q4	Q5
16	0	0	1	1	1
17	0	0	0	1	1
18	1	0	0	0	1
19	1	1	0	0	0
20	0	1	1	0	0
21	1	0	1	1	0
22	1	1	0	1	1
23	1	1	1	0	1
24	0	1	1	1	0
25	1	0	1	1	1
26	0	1	0	1	1
27	1	0	1	0	1
28	0	1	0	1	0
29	0	0	1	0	1
30	0	0	0	1	0 ← Seed

Seed 00010: Sequence 0100001001011001111100011011101

MLSRSs

- Shift Register Sequence
  - The output of an LFSR, i.e. the rightmost bit, is a **shift register sequence**
- Maximal Length Shift Register Sequence
  - An *N*-bit a **maximal length shift register sequence** (MLSRS) is one that repeats after 2^N-1 steps
  - E.g. our example repeats after 31 = 2^5-1 steps

MLSRSs

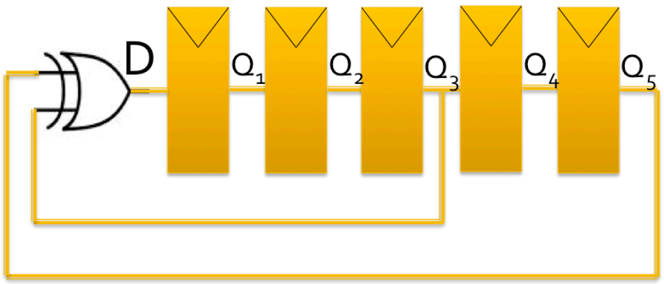
- The **shift register sequence** in our example is Q5

Step	Q1	Q2	Q3	Q4	Q5	Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1	16	0	0	1	1	1
1	1	0	0	0	0	17	0	0	0	1	1
2	0	1	0	0	0	18	1	0	0	0	1
3	0	0	1	0	0	19	1	1	0	0	0
4	1	0	0	1	0	20	0	1	1	0	0
5	0	1	0	0	1	21	1	0	1	1	0
6	1	0	1	0	0	22	1	1	0	1	1
7	1	1	0	1	0	23	1	1	1	0	1
8	0	1	1	0	1	24	0	1	1	1	0
9	0	0	1	1	0	25	1	0	1	1	1
10	1	0	0	1	1	26	0	1	0	1	1
11	1	1	0	0	1	27	1	0	1	0	1
12	1	1	1	0	0	28	0	1	0	1	0
13	1	1	1	1	0	29	0	0	1	0	1
14	1	1	1	1	1	30	0	0	0	1	0
15	0	1	1	1	1						

Sequence: 1000010010110011111000110111010

MLSRSs

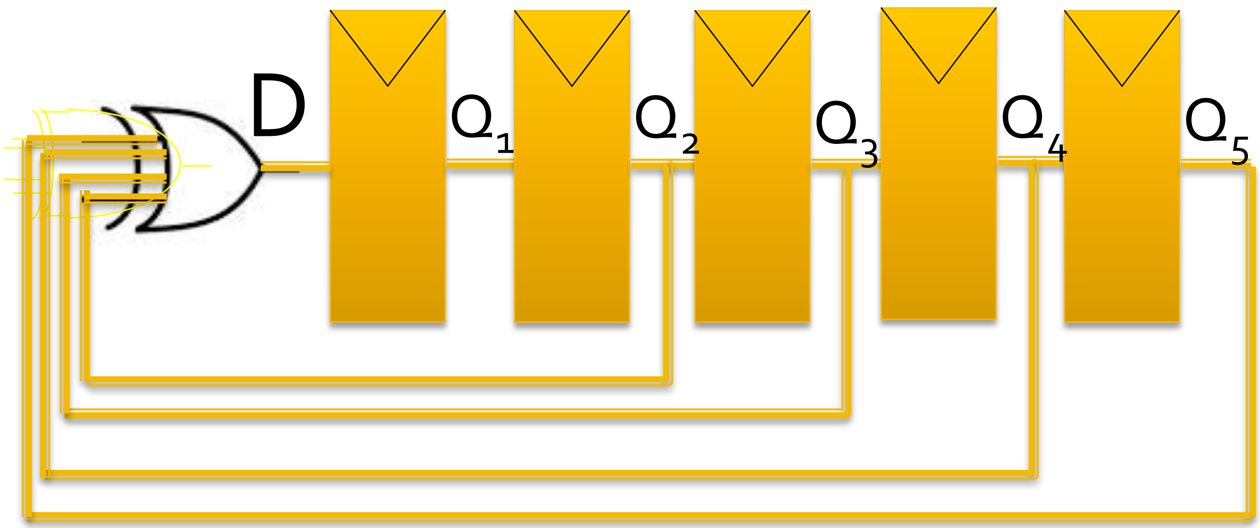
- Taps
  - The bits fed back
- LFSR taps
  - Described by a *characteristic polynomial*
  - E.g. $1 + x^3 + x^5$
    - Taps in columns 3 and 5
    - 1 is not a tap but corresponds to the input to the first bit x^0



MLSRSs

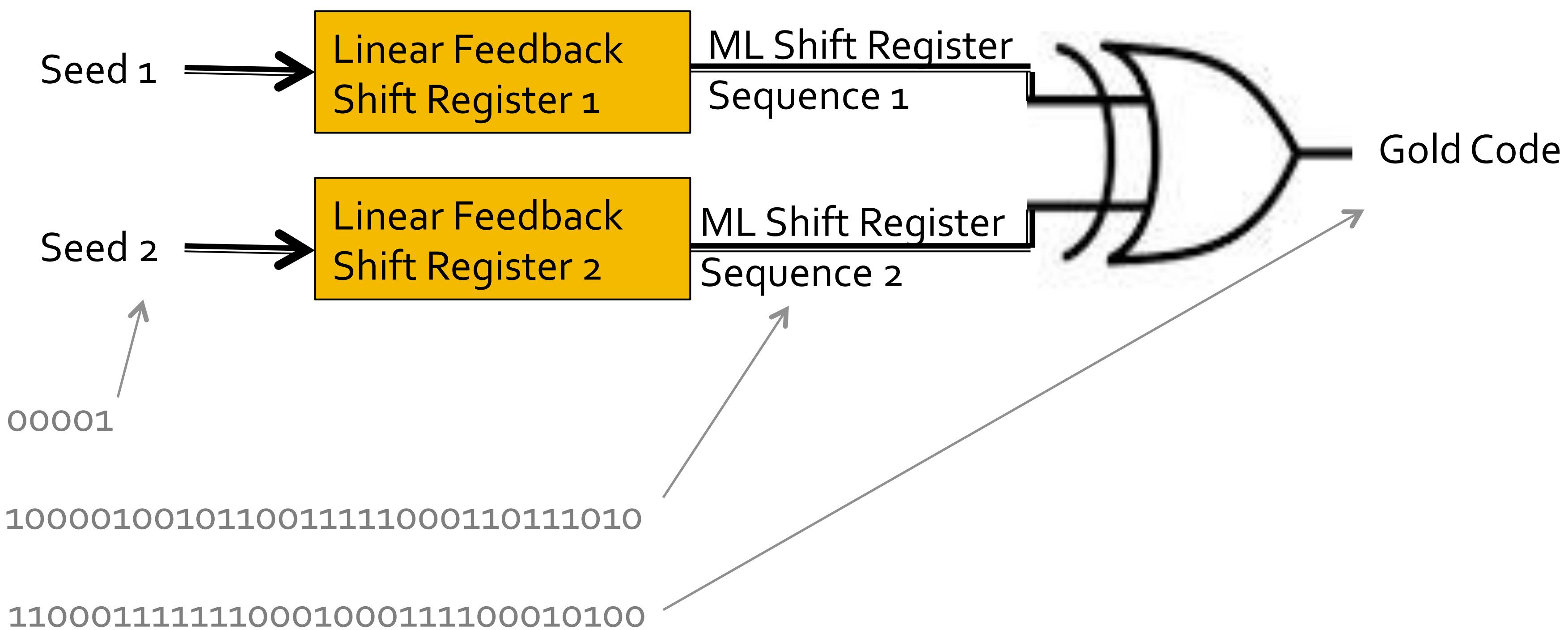
- $1+x^2+x^3+x^4+x^5$ generates a different MLSRS:

1000010110101000111011111001001



Step	Q1	Q2	Q3	Q4	Q5
0	0	0	0	0	1
1	1	0	0	0	0
2	0	1	0	0	0
3	1	0	1	0	0
4	1	1	0	1	0

GC Generation



GC Generation

- To uniquely define a Gold code:
  - State characteristic polynomial for the two LFSRs
  - State seed for the second LFSR
  - Always use a seed of 00…001 for the first LFSR
- Example: $GC(1+x^2+x^3+x^4+x^5,\ 1+x^3+x^5,\ 00011)$
- There are 2^N-1 Gold codes in a family
  - Defined by the different possible seeds (except 00…000)

GC Generation

- Examples
  - GC($1+x^2+x^3+x^4+x^5$, $1+x^3+x^5$, 00001)

    1000010110101000111011111001001 ($1+x^2+x^3+x^4+x^5$ seed 00001)

    xor 1000010010110011111000110111010 ($1+x^3+x^5$ seed 00001)

    0000000100011011000011001110011

  - GC($1+x^2+x^3+x^4+x^5$, $1+x^3+x^5$, 00010)

    1000010110101000111011111001001 ($1+x^2+x^3+x^4+x^5$ seed 00001)

    xor 0100001001011001111100011011101 ($1+x^3+x^5$ seed 00010)

    1100011111110001000111100010100

GC Generation

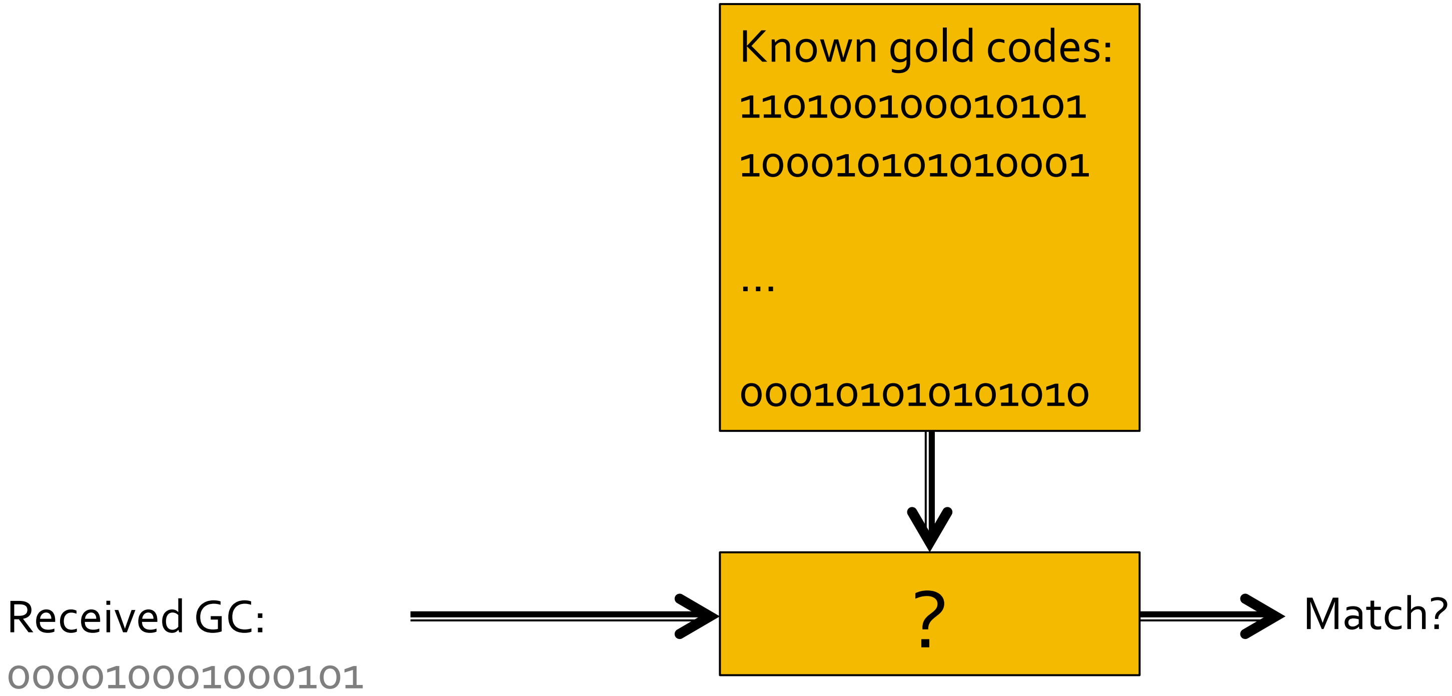
- Gold Codes are a class of 2^N-1 sequences of length 2^N-1
  - Formed by XORing MLSRSs generated by different taps
  - Each seed gives a different Gold code
  - Each code is quite different than the others

Outline

- Gold Code Overview
- Gold Code Generation
- Gold Code Detection
- Application

GC Detection

How do we **detect** gold codes?



How do we **compare** gold codes?

GC Detection

- Compare detected sequence with known Gold Codes
  - Use **correlation**: all possible dot products
  - Highest **correlation** indicates detected Gold Code

GC Detection

- Dot Product
  - The **dot product** of two binary sequences is

    # of positions where bits match
    \- # of positions where bits mismatch

  - Ex: 110010 • 101010

    1 1 0 0 1 0
    1 0 1 0 1 0
    ____

-> dot product is

GC Detection

- Dot product measures similarity of two sequences
  - Large positive dot product indicates strong similarity
  - Large negative dot product indicates nearly all bits differ
  - Dot product near 0 indicates two sequences are uncorrelated
  - Dot product of l-bit sequence with itself is l

GC Detection

- Example of dot product of 2 GCs

1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 (1 + x^3 + x^5 seed 00001)

dot 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 (1 + x^3 + x^5 seed 00010)

= -1 -1 1 1 1 -1 -1 1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 -1

15 matches - 16 mismatches

Dot product is -1

GC Detection

- Cross - Correlation
  - Measures of the similarity of two sequences (e.g. Gold Codes) when one is shifted by varying amounts.
  - Take the dot product of one sequence with each shifted version of the other
- Autocorrelation
  - Cross-correlation of a sequence with itself.

GC Detection

- Lets autocorrelate 110010

110010 • 110010 = 6	(shift by 0)
110010 • 011001 = -2	(shift by 1)
110010 • 101100 = -2	(shift by 2)
110010 • 010110 = 2	(shift by 3)
110010 • 001011 = -2	(shift by 4)
110010 • 100101 = -2	(shift by 5)

Autocorrelation:6, -2, -2, 2, -2, -2

GC Detection

- A Gold Code has
  - A high correlation of 2^N-1 with itself
  - A low correlation with other codes in the family, i.e. the maximum cross-correlation is $2^{(N+1)/2} + 1$

- Robust to noise!
  - Even in the face of noise that flip some of the bits, it is easy to detect a match.

Gold Code Cross-Correlation

- **Hamming Distance**
  - The minimum difference between the correlation with itself and other GC's in the same family

$$(2^N-1) \quad - \quad (2^{(N+1)/2} + 1)$$

  - For our 5-bit code, correlation is 31 with itself
  - The **Hamming distance** is 31-9 = 22

Gold Code Correlation

- Correlation: Gold Code 1, Gold Code 2

GC 1: 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1 1

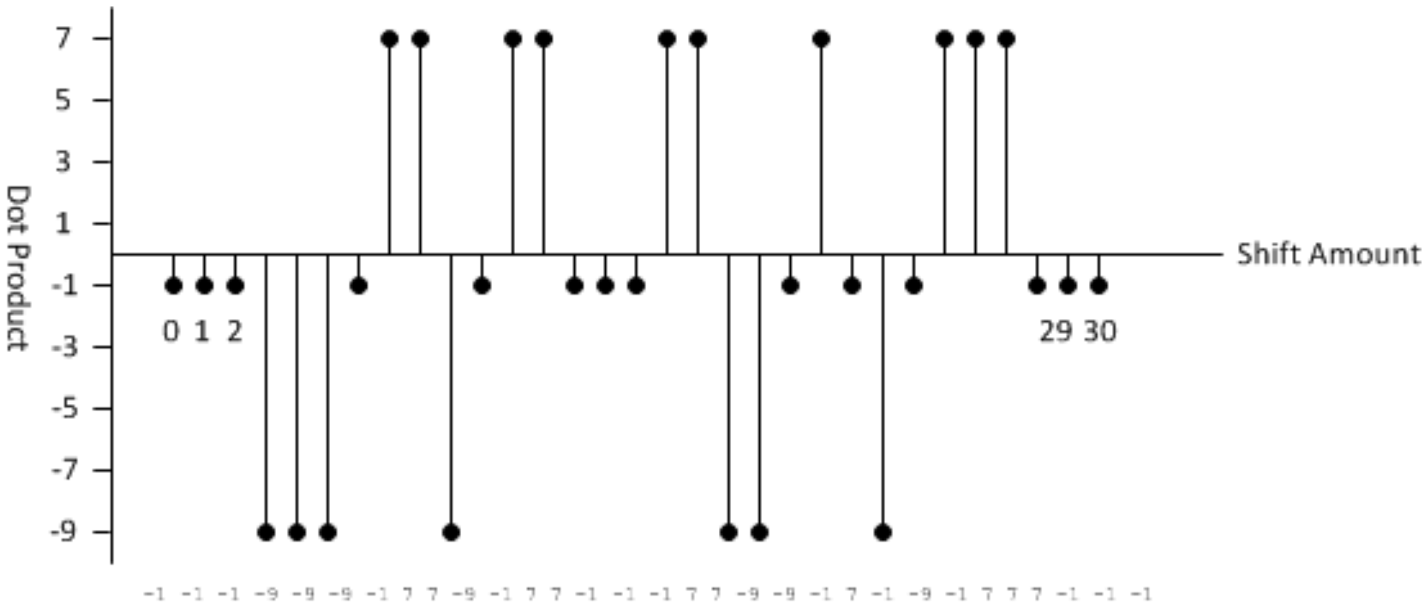
GC 2: 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0

-1 -1 1 1 1 -1 -1 1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 -1

shift =0, dot product = -1

Cross-Correlation

- Cross-correlation of
  - GC($1+x^2+x^3+x^4+x^5$, $1+x^3+x^5$, 00001)
  - GC($1+x^2+x^3+x^4+x^5$, $1+x^3+x^5$, 00010)



Outline

- Gold Code Overview
- Gold Code Generation
- Gold Code Detection
- Application
  - E11 Beacons

Application: E11 Beacons

- LED beacons on the E11 playing field
  - Beacon *b* (*b* = 1…8) flashes GC($1+x^2+x^3+x^4+x^5$, $1+x^3+x^5$, *b*)
  - 4 KHz data rate (250 microseconds / bit)
  - Sequence is inverted depending on team (white vs. green)
- Detect beacons using a phototransistor on your bot
  - Produces a voltage related to the light intensity
  - Principles of operation to be described later

Application: E11 Beacons

1. Read 31 phototransistor samples at 4 KHz
2. Compute average value
3. Convert readings to binary by comparing to average
4. Correlate against each of 31 offsets for each of 8 beacons
5. If correlation exceeds a threshold, report beacon found
6. Improve accuracy by taking more than 31 samples

GC Detection

- Good Luck!!!

GC Detection

- MLSRS is also called a **pseudo-random bit sequence** (PRBS)
  - About half the bits are 0's and half 1's
  - Run length distribution consistent with randomness
  - But sequence is deterministic and easy to generate with XOR

GC Detection

- Autocorrelation with an MLSRS
  - Has a value of 2^N-1 for an offset of 0
  - Has a value of -1 at all other offsets