

# CDM

## Feedback Shift Registers

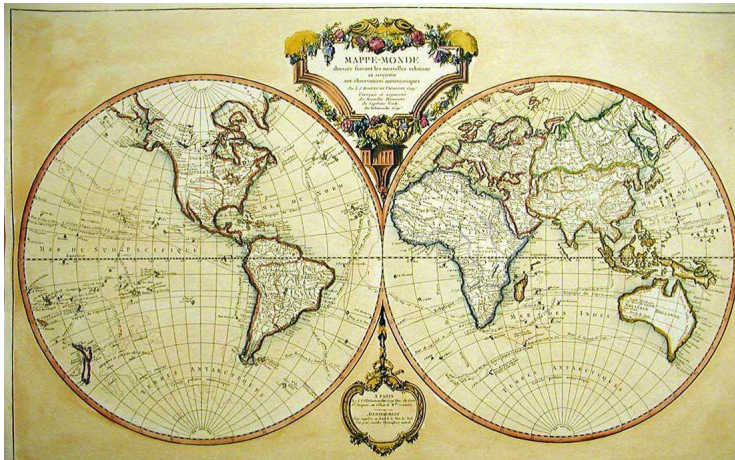
Klaus Sutner  
Carnegie Mellon University

Fall 2011

## Outline

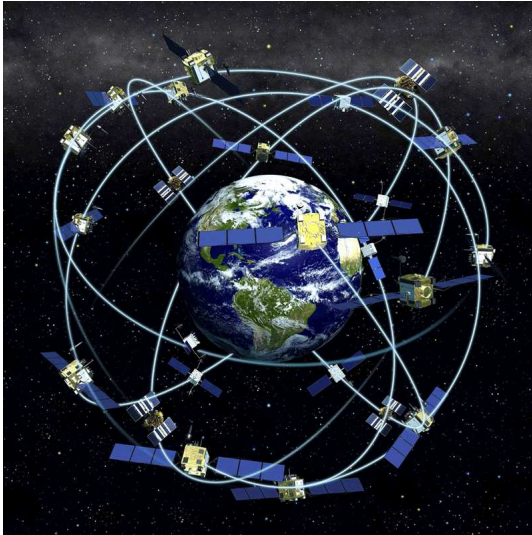
- 1 The GPS Challenge
- 2 Feedback-Shift-Registers
- 3 Generating Functions
- 4 Fibonacci FSRs
- 5 Fibonacci versus Galois

## Getting Around



Athena told Odysseus to “keep the Great Bear on his left”.

## Global Positioning System



## Details

24 satellites, about 20000 km above ground, moving at some 5 km/sec.

Each has 4 highly accurate atomic clocks, everything centrally controlled.

Relativistic slow-down: about 7  $\mu$ sec per day.

**The Problem:** A simple, cheap GPS receiver cannot determine direction of satellites for triangulation.

But: it can measure distance to the satellites by comparing signals (easy with synchronized clocks, more work in real life). Once distance has been measured only have to compute intersections of spheres.

Suppose for simplicity we have synchronized clocks.

So how does one measure the time needed by the signal to travel to the receiver?

## Measuring Delay, on the Cheap

### Main Idea:

Send a stream of bits such that  $k$  consecutive bits suffice to determine the position in the stream.

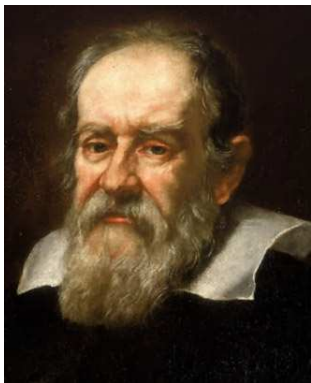
Whole stream is periodic, but the period is quite long.

- Standard Positioning System, C/A signal: period 1023, sent once every millisecond. Accuracy: 100 m, 340 nanoseconds.
- Precision Positioning System, P and Y signals: military use, period of 267 days. Accuracy: 21 m, 200 nanoseconds.

To ascertain the delay, the GPS receiver generates the same sequence.

The receiver can halt its own sequence until the two match up perfectly.

## Galileo Positioning System



The commercial part will have accuracy of 1m, with ground station better than 10cm.

Read Wolfowitz's letter on Wikipedia and despair.

## The Challenge

Fix some positive integer  $k$ , say,  $k = 50$ .

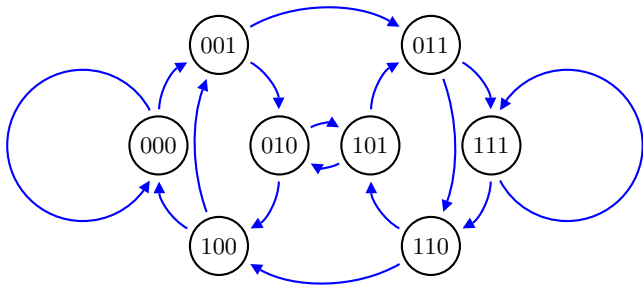
*We want to construct a long bit sequence  $S$  such that someone reading  $k$  consecutive bits in the sequence will know where in the sequence the  $k$  bits are located.*

Of course, for this to work at all  $S$  can contain each  $k$ -bit word at most once. The most ambitious approach is to try to build  $S$  so that every  $k$ -bit word appears exactly once, leading to a length of  $2^k + k - 1$  bits: this is called a **de Bruijn sequence** of order  $k$ .

But we will also settle for length approximately  $2^k$ .

## The Perils of Education

Every CS major will have an immediate knee-jerk response: Just trace a Hamiltonian cycle in a de Bruijn graph. Here is  $\mathbb{B}(3)$ :



This is essentially the minimal DFA for the “third-symbol-from-the-end” language.

## A Computational Obstruction

There are perfectly good linear time graph algorithms that construct Hamiltonian cycles in de Bruijn graphs.

Alas, we need to remember whether we have already touched a node.

This requires  $\Omega(n)$  bits of storage for a size  $n$  graph even when this graph is highly regular and has a nice succinct representation like a de Bruijn graph.

In our case  $n = 2^{50}$ , so any linear space algorithm is out.

- Are there other ways to generate a Hamiltonian cycle in  $\mathbb{B}(k)$ ?
- Or at least a very long cycle, something of length nearly  $2^k$ ?
- We would like to generate the  $i$ th bit in time **and** space  $O(1)$ .

- The GPS Challenge

## 2 Feedback-Shift-Registers

- Generating Functions
- Fibonacci FSRs
- Fibonacci versus Galois

## Long Bit-Sequences

Well-behaved long bit-sequences have several computational applications

- Global positioning system
- Pseudo-random number generation
- Stream ciphers

The problem is that we need a computationally cheap ways to generate bit-sequences with very long periods.

Computationally cheap here means in particular that the memory requirement should be a small constant.

Hence we can only preserve a little bit of state, say,  $k$  bits worth of state. We need a “device” that can update the state and an “output” operation that reads off the next bit.

The output operation will be simple, say,  $g(b_1, b_2, \dots, b_k) = b_1$  or  $g(b_1, b_2, \dots, b_k) = b_1 \wedge b_5$  or some such.

## The Device

But how about the update operation? We need some easily computable function

$$F : \mathbf{2}^k \rightarrow \mathbf{2}^k$$

Then we can iterate away to our heart's content.

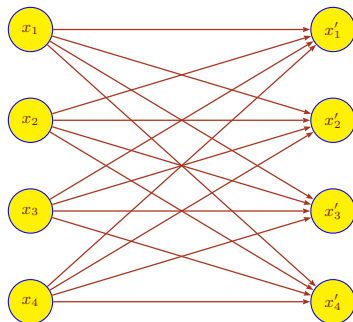
- Pick an initial  $k$ -bit block  $\mathbf{x}_0$ .
- Iterate to obtain a sequence  $\mathbf{x}_{i+1} = F(\mathbf{x}_i)$  of  $k$ -bit blocks.
- Output the bit-sequence  $g(\mathbf{x}_i)$ .

So the problem is to select the right function for  $F$  among the  $(2^k)^{2^k}$  possible choices. Right means the orbit is very long and, of course,  $F$  must not be too complicated.

## A Circuit

Think of this a circuit design problem: we have  $k$  one-bit registers and want to update their contents by some simple circuitry.

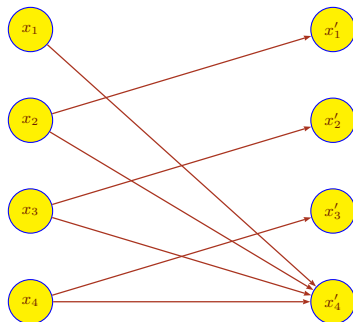
We could make every new bit depend on every old bit:



## Simplify

... but that's too complicated.

How about making only one new bit depend on all old bits, and all the other new bits depend on just a single old bit?



## Shift and Compute

Circuits of this type can be described by specifying a single Boolean function

$$f : \mathbf{2}^k \rightarrow \mathbf{2}$$

together with a shift operation:

$$\begin{aligned}x'_i &= x_{i+1} & i < k \\x'_k &= f(\mathbf{x})\end{aligned}$$

Now the only question is: what is the right choice for  $f$ ?

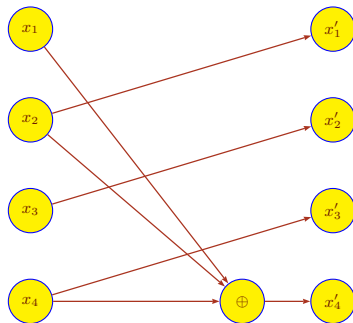
There are  $2^{2^k}$  possibilities.

## Xor Functions

To keep the hardware requirements down, we could use *xor* based rules:

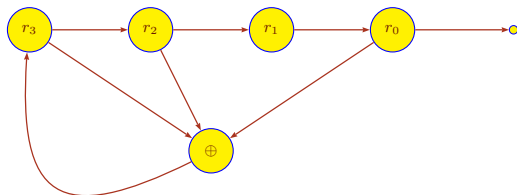
$$f(\mathbf{x}) = x_{p_1} + x_{p_2} + \dots + x_{p_r} \bmod 2$$

The positions  $p_i$  are the so-called **taps**.



## Feedback Shift-Registers

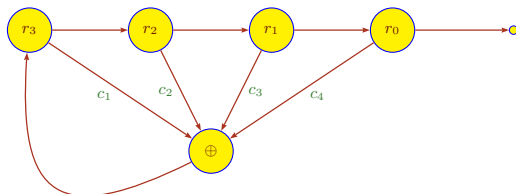
The last step is to redraw the picture slightly, and to renumber the registers (which will make the analysis easier later on).



We can think of generating one output bit at register  $r_0$  during each clock cycle.

## Alternative Description

Another way to think about FSRs is to assume that there is a tap at every register, but they have weights  $c_i \in \mathbf{2}$  which determine whether the tap is on or off.



So the feedback value placed into the last register is

$$c_1 r_{k-1} + c_2 r_{k-2} + \dots + c_k r_0$$

Note that we may safely assume that there is a tap at  $r_0$ : otherwise we are just inflating  $k$  and shifting the output bit a number of times before releasing it into the light. Accordingly,  $k$  is called the **span** of the FSR.

## Generating Bit-Sequences

To generate a bit-sequence we need to choose  $k$  initial values for the registers.

We write  $\mathbf{a} = (a_{k-1}, a_{k-2}, \dots, a_1, a_0)$  for these values.

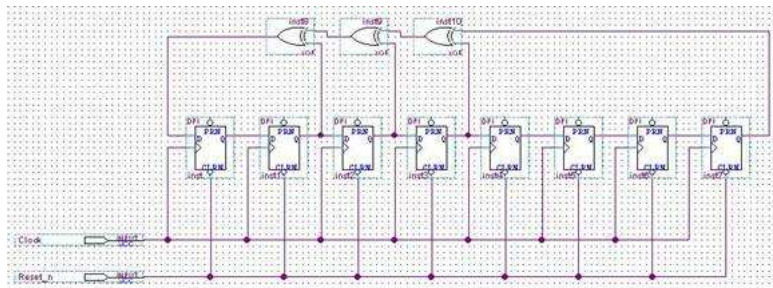
Also, as already indicated, we need to agree on a register where we read off the bits. Which one matters little, since the values are just shifted over, but let's agree on register  $r_0$ .

Clearly, this kind of gadget is very easy to realize in circuitry. Requires only  $k$  one-bit registers, some xor circuits, and a clock (the whole shift-register must be synchronized).

Truth in advertising: xor gates usually have two inputs, so we may need to build a little tree to get the desired feedback bit.

## Engineering Angle

Engineers love these devices: they are easy to implement and lightning fast.



## Global Map

So suppose we have a feedback function  $f(\mathbf{x}) = \sum x_{p_i}$  of span  $k$ , determined by a vector  $\mathbf{p}$  of taps.

According to our convention, the feedback function  $f$  induces a global function

$$F : \mathbf{2}^k \rightarrow \mathbf{2}^k$$
$$F(\mathbf{x}) = (f(\mathbf{x}), x_1, x_2, \dots, x_{k-1})$$

The hope is that a suitable choice of the local map  $f$  (and of the initial conditions  $\mathbf{a}$ ) will produce a long orbit under  $F$ .

## Additivity

### Proposition

*The global map  $F$  is additive:*

$$F(\mathbf{x} + \mathbf{y}) = F(\mathbf{x}) + F(\mathbf{y})$$

*where all addition is mod 2.*

In other words, if we think of  $\mathbf{2}^k$  as an  $\mathbb{F}_2$ -vector space then the map  $F$  is linear. It follows that we can determine the value of  $F(\mathbf{x})$  by just adding the images of the canonical basis vectors:

$$F(\mathbf{e}_i) = F(0, \dots, 0, 1, 0, \dots, 0)$$

### Exercise

*Prove that the global map is indeed additive.*

## Reversibility

### Proposition

*The global map  $F$  based on linear feedback function  $f$  is injective provided that there is a tap at the last register.*

*Proof.*

Since  $c_k = 1$  we can recover  $x_k$  from  $F(\mathbf{x}) = (f(\mathbf{x}), x_1, x_2, \dots, x_{k-1})$ :

$$x_k = f(\mathbf{x}) + \sum_{i < k} c_i x_i.$$

As always, since we are in characteristic 2 the minus is a plus. □

## Reversible Computation

So every FSR of span  $k$  produces a reversible global map  $F : \mathbf{2}^k \rightarrow \mathbf{2}^k$ .

So, we are dealing with reversible gates and, at least from the thermodynamical point of view, our computation is potentially quite cheap.

It follows from reversibility that the functional digraph of  $F$  is just a collection of disjoint cycles, there are no transients.

We need to find  $f$  so that there is one, very long cycle.

## Tracing Cycles in a de Bruijn Graph

Another way of thinking about this is to trace the action of the global map in the de Bruijn graph  $\mathbb{B}(k)$ .

Actually, we are tracing a path backwards.

$$(f(\mathbf{x}), x_1, x_2, \dots, x_{k-1}) \longrightarrow (x_1, x_2, \dots, x_{k-1}, x_k)$$

Since  $F$  is injective that path is a (hopefully long) cycle, anchored at the initial point  $\mathbf{a}$ .

Of course, some starting points are no good:  $\mathbf{0}$  is always a fixed point of  $F$ .

## Full Disclosure

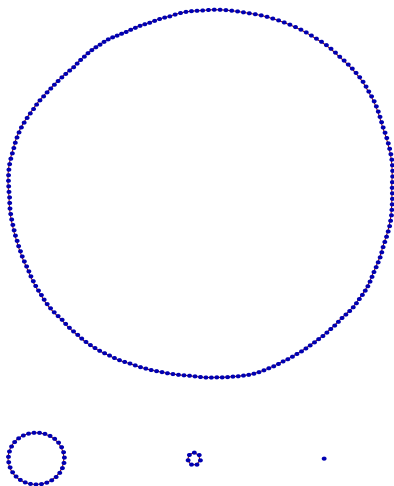
You may recall from the finite fields lectures that multiplication by  $x$  in  $\mathbb{F}[x]/(f)$  can be expressed as a kind of feedback operation.

True, but that is a different type of feedback register that we will discuss later (Fibonacci versus Galois).

For the time being we are only dealing with the FSRs just defined.

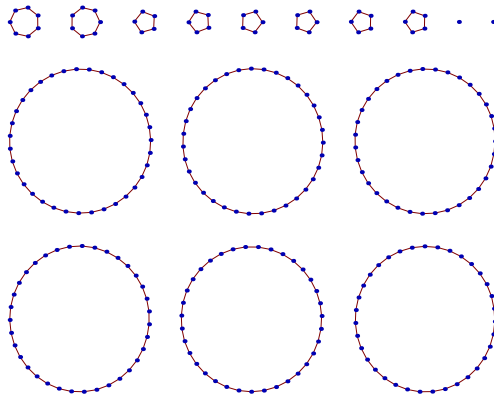
## Phasespace

Here are the cycles obtained from taps 11101011.



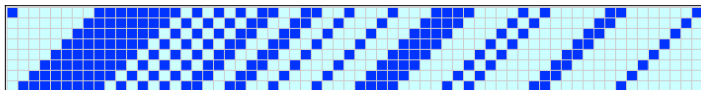
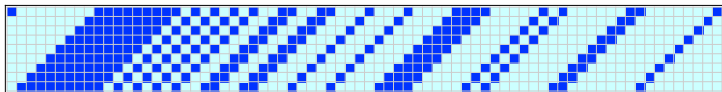
## Changing one Tap

Flip the first tap, so we are now dealing with 01101011.



## Some Orbits

Some pictures of individual orbits for two-tap  $k$ -bit FSRs where  $\mathbf{p} = (0, k - 1)$  and  $k = 6, 8, 9$ . The initial configuration in each case is  $100 \dots 00$ . The periods are 63, 63, 73, respectively.



## Periods

Here is a small table of the periods of these two-tap FSRs where  $\mathbf{p} = (0, k - 1)$ , again for the initial configuration  $100 \dots 00$ .

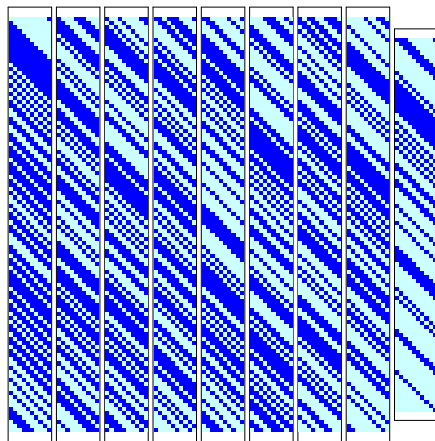
$k$	per
3	7
4	15
5	21
6	63
7	127
8	63
9	73

Does not look particularly spectacular. But some easy conjectures pop up:

### Conjecture

*The period is  $2^{2t} - 1$  for  $k = 2^t$  bits.*

## A Surprise



But for  $k = 10$  the period jumps to 889, surprisingly close to the upper bound 1024.

## More Periods

A slightly bigger table confirms that at or near powers of 2 things are easy, but overall the period lengths are rather complicated.

$k$	per	$k$	per
1	—	11	1533
2	3	12	3255
3	7	13	7905
4	15	14	11811
5	21	15	32767
6	63	16	255
7	127	17	273
8	63	18	253921
9	73	19	413385
10	889	20	761763

So for  $k = 20$  the length of the cycle is  $761763/1048576 \approx 73\%$  of the maximal value. For  $k = 18$  it's even 97%.

## Impulse-Response Sequences

Additivity has an important side-effect when it comes to periods: we can obtain maximal period by selecting as the starting configuration the unit vector  $e_1$ .

### Lemma

*The period of any configuration  $a$  divides the period of  $e_1$ .*

### Definition

An **impulse-response** sequence for  $F$  is an orbit obtained from a basis vector  $e_1$ .

These cycles won't be Hamiltonian, but at least for some choices of  $k$  and the feedback function they are quite long. Moreover, the next bit can indeed be computed in constant time and space.

## The Companion Matrix

We need a little more machinery (which is also independently useful) for the proof of the lemma.

### Definition

Let  $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$  be linear recurrence of order  $k$  over some ring  $R$ . The **companion matrix**  $C$  of the recurrence is a  $k \times k$  matrix over  $R$  defined by

$$C(i, j) = \begin{cases} c_j & \text{if } i = 1, \\ 1 & \text{if } i = j + 1, \\ 0 & \text{otherwise.} \end{cases}$$

For example, for  $k = 5$  the companion matrix looks like so:

$$C = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

## Matrix Multiplication and Iteration

By multiplying the companion matrix with a vector representing the current bit-pattern in the registers we can compute the next bit-pattern.

### Proposition

$C^t \mathbf{a}$  is the content of the registers at time  $t$  where  $\mathbf{a} = (a_{k-1}, \dots, a_0)$  is the initial configuration.

Computationally this produces the following speed-up: using a standard matrix multiplication algorithm and fast exponentiation it would take us  $O(k^3 \lg t)$  steps to compute the state of the system at time  $t$ .

This is an example of predictability or computational compressibility: it does not take  $\Omega(t)$  steps to find the configuration at time  $t$ .

## Proof of Lemma

Let  $q$  be the period of the impulse-response sequence generated by  $\mathbf{e}_1$ ,  $C$  the companion matrix.

Then  $C^q \mathbf{e}_1 = \mathbf{e}_1$  and hence for any  $i \geq 0$  we have

$$C^{q+i} \mathbf{e}_1 = C^i \mathbf{e}_1$$

and therefore

$$(C^q - I) \cdot (C^i \mathbf{e}_1) = 0.$$

But  $C^i \mathbf{e}_1$  runs through the standard basis of  $\mathbf{2}^k$ , so we must have  $C^q = I$ .

It follows that  $q$  is a period of any configuration, and the least period must divide  $q$ .

□

- The GPS Challenge
- Feedback-Shift-Registers

### 3 Generating Functions

- Fibonacci FSRs
- Fibonacci versus Galois

## Predicting Periods

The key question now is: Where should the taps go?

Note that we actually have a problem if we really succeed: If the cycles obtained from an impulse-response sequence are indeed very long then we cannot really find out by simulation, at least not for large  $k$ .

Hence, we need a computational shortcut, some way of computing periods from the tap positions without brute-force simulation.

### Exercise

*For a reasonably small value of  $k$ , say,  $k = 10$ , determine the complete cycle structure of  $2^k$  for all possible feedback shift-registers of order  $k$ .*

## Words of Wisdom



*There is nothing more practical than a good theory.*

*Kurt Lewin*

## Generating Functions

Consider the sequence  $(a_n)$  of bits in the rightmost register  $r_0$ .

Write the original configuration as  $a_{k-1}, a_{k-2}, \dots, a_1, a_0$ .

Clearly we have a linear recurrence

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

where the coefficients  $c_i$  are determined by the taps as above,  $n \geq k$ .

Can we get a good description for the generating function

$$G(x) = \sum a_n x^n?$$

## Example Generating Function

Consider a span 4 FSR with taps  $c_1 = c_4 = 1$ ,  $c_2 = c_3 = 0$ , and initial conditions  $a_0 = 1$ ,  $a_1 = a_2 = a_3 = 0$ . So, we have a generalized Fibonacci recurrence  $a_n = a_{n-1} + a_{n-4}$  for  $n \geq 4$ .

Then for  $G(x) = \sum a_n x^n$  we have

$$\begin{aligned}G(x) &= 1 + \sum_{n \geq 4} a_n x^n \\&= 1 + x \sum_{n \geq 3} a_n x^n + x^4 \sum_{n \geq 0} a_n x^n \\&= 1 - x + x \sum_{n \geq 0} a_n x^n + x^4 \sum_{n \geq 0} a_n x^n \\&= 1 - x + (x + x^4)G(x)\end{aligned}$$

so that

$$G(x) = \frac{1 - x}{1 - x - x^4}$$

## Comments

We have used minus signs to emphasize the arithmetic, in characteristic 2 we could just as well have written plus (but then you have to make changes for the characteristic  $p > 2$  case).

From the example, one can see that the denominator of the rational generating function will be

$$1 - c_1x - c_2x^2 - \dots - c_kx^k.$$

The numerator is slightly more complicated to write down.

## The General Case

### Theorem

*The generating function for our feedback shift-registers is*

$$G(x) = \frac{-\sum_{i=0}^{k-1} \left( \sum_{j=0}^i c_j a_{i-j} \right) x^i}{1 - \sum_i c_i x^i}$$

where  $c_0 = -1$ .

*Proof.*

$$\begin{aligned} G(x) &= \sum_n \left( \sum_i c_i a_{n-i} \right) x^n \\ &= \sum_i c_i x^i \sum_n a_{n-i} x^{n-i} \\ &= \sum_i c_i x^i \left( a_{-i} x^{-i} + \dots + a_{-1} x^1 + \sum_n a_n x^n \right) \end{aligned}$$

□

## Feedback Polynomial

### Definition

The denominator of this rational function is called the **feedback polynomial** or the **connection polynomial** of the sequence  $(a_n)$ .

Note that we can solve the equation  $-\sum_{i=0}^{k-1} \left( \sum_{j=0}^i c_j a_{i-j} \right) x^i = 1$  for the  $a_i$ .

The solution yields the initial conditions for a sequence whose generating function simplifies to the reciprocal of the feedback polynomial:

$$G(x) = \frac{1}{1 - \sum_i c_i x^i}$$

In general, however, the numerator is some polynomial of degree less than  $k$ .

## Example

For taps  $\mathbf{p} = (0, 4)$  and the impulse-response sequence we get

$$G(x) = \frac{1 - x}{1 - x - x^5}$$

By Taylor expansion, we can compute a few terms of this series:

$$1 + x^5 + x^6 + x^7 + x^8 + x^9 + 2x^{10} + 3x^{11} + 4x^{12} + 5x^{13} + 6x^{14} + 8x^{15} + 11x^{16} \dots$$

Note that we get integer coefficients, which is not quite right.

## Wrong Field

We are computing in characteristic 0, but we should be using characteristic 2.

Then the expansion looks like

$$1 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{11} + x^{13} + x^{16} + x^{17} + \dots$$

and that is the right answer.

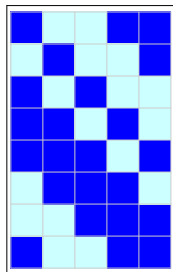
Of course, computing Taylor expansions is not really an answer at all: we need to calculate the period, and the first few Taylor coefficients tell us nothing about that.

## Changing Initial Conditions

Suppose we change the initial conditions to  $(1, 1, 0, 0, 1)$ . Then

$$\begin{aligned}G(x) &= \frac{1 - x^2 + x^4}{1 - x - x^5} \\ &= 1 + x + x^4 + x^6 + x^7 + x^8 + x^{11} + x^{13} + x^{14} + \dots\end{aligned}$$

and the orbit looks like so:



### Exercise

*Match the series from above with the picture.*

## Period and Polynomial Divisors

### Theorem

Let  $(a_n)$  be a sequence with generating function  $G(x) = 1/g(x)$ . Then the period of  $(a_n)$  is the least  $p > 0$  such that  $g(x)$  divides  $1 - x^p$ .

*Proof.*

If the period is  $p$  then

$$\begin{aligned} 1/g(x) &= (a_0 + a_1x + \dots + a_{p-1}x^{p-1}) (1 + x^p + x^{2p} + \dots) \\ &= (a_0 + a_1x + \dots + a_{p-1}x^{p-1}) / (1 - x^p) \end{aligned}$$

So  $g(x)$  divides  $1 - x^p$ , as required.

## And Back

On the other hand, if  $1 - x^p = g(x)(b_0 + b_1x + \dots + b_{p-1}x^{p-1})$  then

$$\begin{aligned} 1/g(x) &= (b_0 + b_1x + \dots + b_{p-1}x^{p-1}) / (1 - x^p) \\ &= (b_0 + b_1x + \dots + b_{p-1}x^{p-1}) (1 + x^p + x^{2p} + \dots) \end{aligned}$$

Comparing coefficients we get  $a_n = b_{n \bmod p}$ , so the period must of the sequence must divide  $p$ .

Since  $p$  is minimal, they must agree.



## Characteristic 2

Note that the same result holds in any field of characteristic 2, we have not used anything but the standard, axiomatic field properties (plus a touch of formal power series).

This is precisely the reason why it is a good idea to argue from axioms rather than concrete examples: the latter don't carry over to other domains.

Of course, concrete examples are very valuable when it comes to forming the right conjecture in the first place, but once one knows exactly what one would like to prove, abstraction rules supreme.

Again, we have kept the minus sign to keep the argument general, but note that in characteristic 2 it turns into a plus sign.

## Numerators

What happens if the generating function is a general rational function

$$G(x) = h(x)/g(x)?$$

We may assume that  $h$  and  $g$  are coprime, otherwise we can simply factor out.

Accordingly, the sequence will be eventually periodic in the general case, but strictly periodic as long as the degree of  $h$  is less than the degree of  $g$  – exactly the situation that we are in.

The denominator controls period length, the numerator expresses the initial conditions given by the first  $k$  bits in the registers.

## Long Cycles and Primality

### Definition

The least  $p > 0$  such that  $g(x)$  divides  $1 - x^p$  in  $\mathbb{F}[x]$  is called the **exponent** of  $g(x)$ .

It is not clear how to compute exponents efficiently, but one can show the following.

### Theorem

*If a shift-register sequence of span  $k$  has maximum length  $2^k - 1$  then the corresponding polynomial  $g(x)$  must be irreducible.*

Unfortunately, this condition is not sufficient.

But quite a bit is known about the exponents of irreducible polynomials.

## Exploiting Mersenne

Here is a nice trick exploiting primality.

Any irreducible polynomial  $g(x)$  of degree  $k$  divides  $1 - x^{2^k - 1}$ . Hence the exponent of  $g(x)$  must divide  $2^k - 1$ .

So if  $2^k - 1$  happens to be prime (a so-called Mersenne prime), then the exponent must be equal to  $2^k - 1$ .

It is an open problem whether infinitely many Mersenne primes exist but for the following values of  $k$  we do get a Mersenne prime  $2^k - 1$ :

$$2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127$$

These are all appropriate  $k$ 's less than 256.

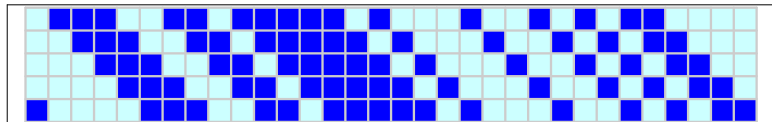
BTW, the largest known Mersenne prime currently is  $2^{43,112,609} - 1$ .

## Example Mersenne Prime

31 is a Mersenne prime,  $k = 5$ .

There are exactly 6 irreducible polynomials of degree 5 in  $\mathbb{F}_2[x]$ , the orbit for the last is plotted below.

$$1 + x^2 + x^5, 1 + x^3 + x^5, 1 + x + x^2 + x^3 + x^5, \\ 1 + x + x^2 + x^4 + x^5, 1 + x + x^3 + x^4 + x^5, 1 + x^2 + x^3 + x^4 + x^5$$



## Alas ...

Unfortunately, in general not every irreducible polynomial produces a shift-register sequence of maximum length, but for a “good” irreducible polynomial we can choose the taps according to its coefficients, and obtain a sequence of maximum period.

This is essentially the same problem that we encountered in the discussion of finite fields: any irreducible polynomial  $f$  will give us a field  $\mathbb{F}[x]/(f)$  but  $\alpha = x/(f)$  is not always a generator of the multiplicative subgroup.

## Primitive Field Elements

To hammer this home:

We know that the multiplicative subgroup of every finite field is cyclic.

More precisely, if we have  $K = \mathbb{F}[x]/(f)$  where  $f$  is irreducible **and** primitive then we can choose  $\alpha = x/(f)$  as a generator.

But the order of  $\alpha$  is  $2^k - 1$ , and likewise for  $\alpha^{-1}$ .

Hence if we choose the taps of our FSR according to the coefficients of  $f$  we obtain maximal period.

## The Field Angle

Here is a wild & woolly idea: maybe shift register sequences can be explained completely in terms of finite fields, something along the lines of

$$a_n = \text{blah blah } \alpha^n \text{ blah blah}$$

Since FSRs are connected to multiplication by  $\alpha = x/(f) \in \mathbb{F}_p/(f)$  this may not be so unreasonable.

The good news: this actually works.

The bad news: it involves a bit of pain.

## Traces

Suppose  $f(x) \in \mathbb{F}_2[x]$  is irreducible of degree  $k$ .

Let  $\alpha$  be a root of  $f$  over the splitting field  $\mathbb{F}_{2^k}$ .

### Definition

The **trace** function of  $\mathbb{F}_{2^k}$  over  $\mathbb{F}_2$  is defined by

$$\text{Tr} : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_2 \qquad \text{Tr}(z) = \sum_{i < k} z^{2^i}$$

### Proposition

*Tr is linear and its range is indeed  $\mathbb{F}_2$ .*

### Exercise

*Prove the last proposition.*

## Trace Representation

### Theorem

Let  $\beta$  be any element  $\mathbb{F}_{2^k}$  and initialize the registers of a FSR with

$$\text{Tr}(\beta), \text{Tr}(\alpha^{-1}\beta), \dots, \text{Tr}(\alpha^{2-k}\beta), \text{Tr}(\alpha^{1-k}\beta).$$

Then the sequence generated by the FSR is  $a_n = \text{Tr}(\alpha^{-n}\beta)$ .

*Proof.*

To see this first note that  $\alpha^{-k} = \sum c_i \alpha^{i-k}$ .

But then

$$\begin{aligned} c_1 \text{Tr}(\alpha^{1-k}\beta) + c_2 \text{Tr}(\alpha^{2-k}\beta) + \dots + c_k \text{Tr}(\beta) &= \\ \text{Tr}(\beta(c_1 \alpha^{1-k} + c_2 \alpha^{2-k} + \dots + c_k)) &= \\ \text{Tr}(\alpha^{-k}\beta) \end{aligned}$$

Done by induction. □

## Counting Initial Conditions

The question arises how many of the  $2^k$  possible initial vectors are of the special form

$$\mathrm{Tr}(\beta), \mathrm{Tr}(\alpha^{-1}\beta), \dots, \mathrm{Tr}(\alpha^{2-k}\beta), \mathrm{Tr}(\alpha^{1-k}\beta).$$

Since  $\beta$  ranges over the whole extension field, there are  $2^k$  choices.

We claim that these choices must all produce distinct vectors.

For suppose two vectors agree; then by linearity there is a  $\beta$  such that  $\mathrm{Tr}(\alpha^i\beta) = 0$  for all  $i < k$ . We can think of the condition  $\mathrm{Tr}(\alpha^i\beta) = 0$  for all  $i < k$ , as a system of linear equations.

## Vandermonde Matrices

The matrix of this system has a special form: it's (the transpose of) a Vandermonde matrix.

$$V(\gamma_1, \dots, \gamma_m) = \begin{pmatrix} 1 & \gamma_1 & \gamma_1^2 & \cdots & \gamma_1^{n-1} \\ 1 & \gamma_2 & \gamma_2^2 & \cdots & \gamma_2^{n-1} \\ 1 & \gamma_3 & \gamma_3^2 & \cdots & \gamma_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_m & \gamma_m^2 & \cdots & \gamma_m^{n-1} \end{pmatrix}$$

The determinant of  $V$  is  $\prod_{i < j} (\gamma_j - \gamma_i)$ .

It follows that our matrix is invertible. But then  $\beta = 0$ , done. Hence all  $2^k$  initial conditions can be generated by the right choice of the multiplier  $\beta$ .

- The GPS Challenge
- Feedback-Shift-Registers
- Generating Functions
- ④ Fibonacci FSRs
  - Fibonacci versus Galois

## Fibonacci Feedback Shift-Registers

The feedback shift-registers we have considered so far generate linear recurrent sequences according to

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

Since this generalizes the classical Fibonacci sequence, these FSR are also called **Fibonacci feedback shift-registers (FFSR)**. Of course, the classical Fibonacci sequence  $k = 2$ ,  $c_1 = c_2 = 1$ , is defined over  $\mathbb{Z}$ , we are dealing with  $\mathbb{Z}_2$ .

Perhaps one can use the machinery for recurrence equations to study FSR sequences?

## Characteristic Polynomials

### Definition

The **characteristic polynomial** of a square matrix  $M$  is defined by

$$\gamma(x) = |M - xI|.$$

For a companion matrix  $C$  it is easy to see that the characteristic polynomial has the form

$$\gamma(x) = x^k - c_1x^{k-1} - c_2x^{k-2} - \dots - c_k$$

Compare this to the connection polynomial

$$g(x) = -1 + c_1x^1 + c_2x^2 + \dots + c_kx^k$$

So they are “mirror images” of each other:  $-x^k \gamma(1/x) = g(x)$ .

Can we exploit this fact?

## Reciprocal Polynomials

### Definition

The **reciprocal** of a polynomial  $f(x)$  of degree  $k$  is the polynomial  $f^*(x) = x^k f(1/x)$ .

The map  $f \mapsto f^*$  is not very well behaved (it is not a homomorphism), but we have the following properties.

### Proposition

*Let  $f$  and  $g$  be two polynomials.*

- *If  $f(0) \neq 0$  then  $(f^*)^* = f$ .*
- *$(f \cdot g)^* = f^* \cdot g^*$ .*

### Exercise

*Prove the last proposition.*

## Connection and Characteristic

By the last proposition,  $f$  is irreducible if, and only if,  $f^*$  is so irreducible.

Moreover, in characteristic 2 we have  $(1 + x^d)^* = 1 + x^d$ , so the exponents of an irreducible polynomial and its reciprocal are the same.

Hence, as far as irreducibility and exponent are concerned, there is no difference between the connection polynomial  $g(x)$  of a FFSR and the characteristic polynomial  $\gamma(x)$  of its companion matrix.

In fact, some authors use  $\gamma(x)$  instead of  $g(x)$  (but refer to it as the connection polynomial).

## Example Degree 10

For degree 10 there are 99 irreducible polynomials in  $\mathbb{F}_2[x]$ .

The frequencies for their exponents are

exp.	count
11	1
33	2
93	6
341	30
1023	60

So almost  $2/3$  of the irreducible polynomials would produce maximum length FSR sequences.

Note that the map  $f \mapsto f^*$  moves polynomials only within each of the groups.

## Counting Irreducibles

How many irreducible polynomials of degree  $k$  are there in  $\mathbb{F}_2[x]$ ?

Let's write  $I_k$  for this number, so trivially  $0 \leq I_k < 2^k$ .

### Proposition

$$\frac{1}{1-2z} = \prod_{m \geq 1} \left( \frac{1}{1-z^m} \right)^{I_m}$$

This comes down to the simple observation that every polynomial is a product of powers of irreducible ones, and this decomposition is essentially unique: if we order the factors in some canonical way then the decomposition is unique.

### Exercise

*Prove the proposition using the hint above.*

## Möbius Inversion

A bit more fumbling shows that

### Lemma

Let  $I_m$  be the number of irreducible polynomials in  $\mathbb{F}_2[x]$  of degree  $m$ . Then

$$2^m = \sum_{d|m} d I_d.$$

We can apply Möbius inversion to this expression to find:

$$I_m = \frac{1}{m} \sum_{d|m} 2^d \mu(m/d).$$

where  $\mu$  is the Möbius function.

## A Table

The last expression is fairly elegant, but it's not clear what one should expect numerically.

$m$	$I_m$
10	99
11	186
12	335
13	630
14	1161
15	2182
16	4080
17	7710
18	14532
19	27594
20	52377

Incidentally,  $I_{50} = 22517997465744$ .

- The GPS Challenge
- Feedback-Shift-Registers
- Generating Functions
- Fibonacci FSRs
- ⑤ Fibonacci versus Galois

## A Small Gripe

From the point of view of the circuitry needed to implement a FSR, the Fibonacci type has one small drawback: it involves the addition of all the tapped registers.

In particular when the FSR is programmable (i.e., when the taps can be set arbitrarily), this incurs a bit of overhead.

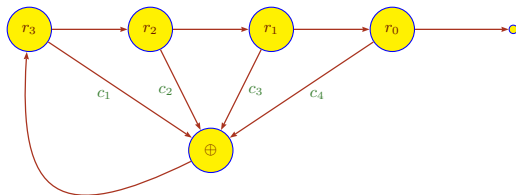
Could we modify the device slightly so that both the shift and the feedback operations can be performed in “one step” by the circuitry?

Another crazy idea: How about reversing all the arrows in an FFSR?

In other words, feed the content of the last register back to all the tapped registers.

## Recall: Fibonacci FSR

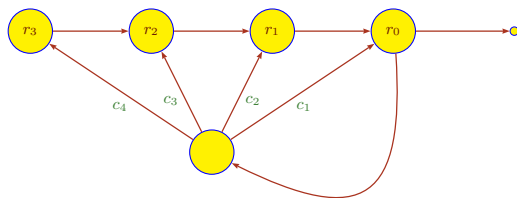
Our Fibonacci FSRs look like so:



The taps sample the registers and send the sum back to the last register.

## Reversing Arrows

Keeping our shift direction to go from left to right, the new device looks like so:



After shifting, the bits in the tapped registers are flipped, provided the bit in  $r_0$  was a 1.

Note that we have changed the numbering system on the tap coefficients, this will make the algebra a bit easier in what follows.

## Galois Feedback Shift-Registers

These devices are called **Galois feedback shift-registers (GFSR)**. The global function now looks like so:

$$\begin{aligned} F : \mathbf{2}^k &\rightarrow \mathbf{2}^k \\ F(\mathbf{x}) &= (0, x_1, x_2, \dots, x_{k-1}) + x_k \mathbf{c} \\ &= (x_k c_k, x_1 + x_k c_{k-1}, \dots, x_{k-1} + x_k c_1) \end{aligned}$$

Thus if  $x_k = 0$  we simply shift; otherwise we shift and then flip all bits in certain positions.

This is really what we did in the discussion of finite fields.

## Galois versus Fibonacci

This looks quite similar to our original Fibonacci design, so one might expect the bit sequences obtained from a GFSR to be quite similar to the FFSR sequences.

First off, let us agree that the bit sequence  $(b_n)$  generated by a GFSR is read off at register  $r_0$ .

Note that given initial values  $a_{k-1}, \dots, a_0$  it is in general not the case that  $b_i = a_i$  for  $i < k$  (for FFSR there is no problem).

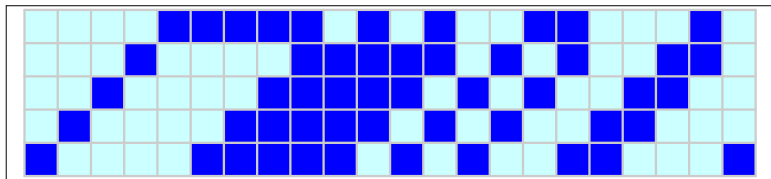
We have  $b_0 = a_0$ , but the other bits may differ.

Second, for a GFSR it is not so easy to write down a recurrence equation: the distributed nature of the changes speeds up the circuits, but makes the math more complicated.

## Experiment

Time to stop worrying and start computing.

Here is a picture for a two-tap GFSR with  $\mathbf{c} = (1, 0, 0, 0, 1)$ .



The first 4 steps a pure shifts, and at step 5 the masking takes effect.

At any rate, this looks just like the Fibonacci case, so we should expect similar results.



## Exploiting Periodicity

Again we may assume that  $c_k = 1$ : otherwise we are simply inflating  $k$ .

But then the induced map  $F : \mathbf{2}^k \rightarrow \mathbf{2}^k$  is injective, just as in the Fibonacci case.

Hence the corresponding bit-sequence  $(b_n)$  obtained at register  $r_0$  is strictly periodic and its generating function must be rational:

$$B(x) = \frac{P(x)}{Q(x)}$$

where  $P$  is some polynomial of degree less than the degree of  $Q$ .

- What are the polynomials in question?
- How do they compare to the polynomials for a Fibonacci SR?

## Generating Functions for Galois FSR

### Theorem

Let  $B(x)$  be the generating function for the sequence obtained at register  $r_0$  in a Galois FSR. Then

$$B(x) = \frac{-\sum_{i=0}^{k-1} a_i x^i}{-1 + \sum_{i=1}^k c_i x^i}$$

*Proof.*

For simplicity we consider only  $b_n$  for  $n \geq k$ , the first few values require slightly more fumbling.

Since the registers are shifted at each step,  $b_n$  depends only on  $b_{n-k}, \dots, b_{n-1}$  but none of the earlier values. Moreover, because of the masking whenever the output bit is 1 we have

$$b_n = \sum_{i=1}^k c_i b_{n-i}.$$

Done by comparing coefficients. □

## Déjà vu all over again

So the connection polynomials are the same – but recall that we are numbering the  $c$  coefficients backwards. In other words, you have to take the mirror image of the connection vector in the Fibonacci case.

The numerators are harmless: they comprise all polynomials of degree at most  $k - 1$ , but to get the same polynomial we need different initial conditions.

Thus, we get exactly the same bit-sequences, but we have to adjust the taps and the initial conditions.

This is really surprising, it's intuitively far from clear that GFSRs and FFSRs are equivalent in this sense.

Example:  $\mathbb{F}_{16}$ 

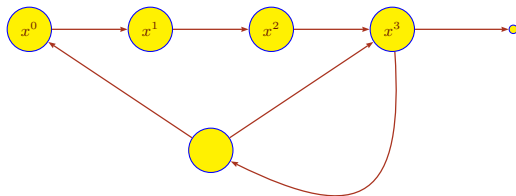
Consider the irreducible polynomial  $f(x) = 1 + x^3 + x^4 \in \mathbb{F}_2[x]$ .

In this case  $\alpha = x/(f)$  is primitive and generates the whole multiplicative subgroup.

$\alpha^0$	1	$\alpha^8$	$x + x^2 + x^3$
$\alpha$	$x$	$\alpha^9$	$1 + x^2$
$\alpha^2$	$x^2$	$\alpha^{10}$	$x + x^3$
$\alpha^3$	$x^3$	$\alpha^{11}$	$1 + x^2 + x^3$
$\alpha^4$	$1 + x^3$	$\alpha^{12}$	$1 + x$
$\alpha^5$	$1 + x + x^3$	$\alpha^{13}$	$x + x^2$
$\alpha^6$	$1 + x + x^2 + x^3$	$\alpha^{14}$	$x^2 + x^3$
$\alpha^7$	$1 + x + x^2$	$\alpha^{15}$	1

## The Corresponding GFSR

Here is FSR that multiplies elements in  $\mathbb{F}_{16}$  by  $\alpha$ .



The shift corresponds to the actual multiplication by  $x$ , and the potential flip to reduction according to  $x^4 = 1 + x^3$ .

## Summary

- (Near) Hamiltonian cycles play an important role in GPS.
- Also very important in algebraic coding theory.
- The standard graph theory argument for their construction is inapplicable in resource bounded computation.
- Feedback shift-registers provide an excellent, cheap way to construct long cycles.
- Come in two flavors: Fibonacci and Galois.
- Analysis and design requires finite field theory; in particular primitive elements.