EECS 150 - Components and Design Techniques for Digital Systems

Lec 26 – CRCs, LFSRs (and a little power)

David Culler
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~culler
http://www-inst.eecs.berkeley.edu/~cs150

Review

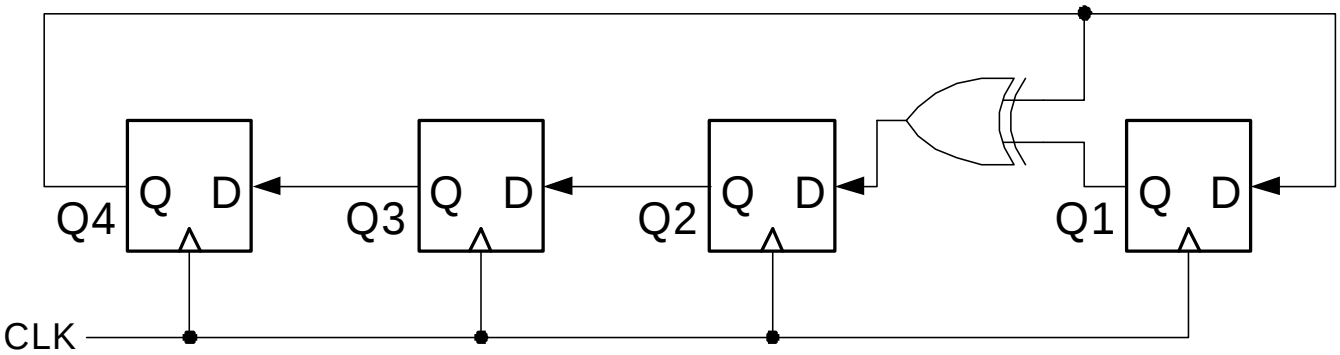
- **Concept of error coding**
  - **Add a few extra bits (enlarges the space of values) that carry information about all the bits**
  - **Detect: Simple function to check of entire data+check received correctly**
    - » **Small subset of the space of possible values**
  - **Correct: Algorithm for locating nearest valid symbol**
- **Hamming codes**
  - **Selective use of parity functions**
  - **Distance + # bit flips**
  - **Parity: XOR of the bits => single error detection**
  - **SECDED**
    - » **databits+p+1 < 2^p**

Outline

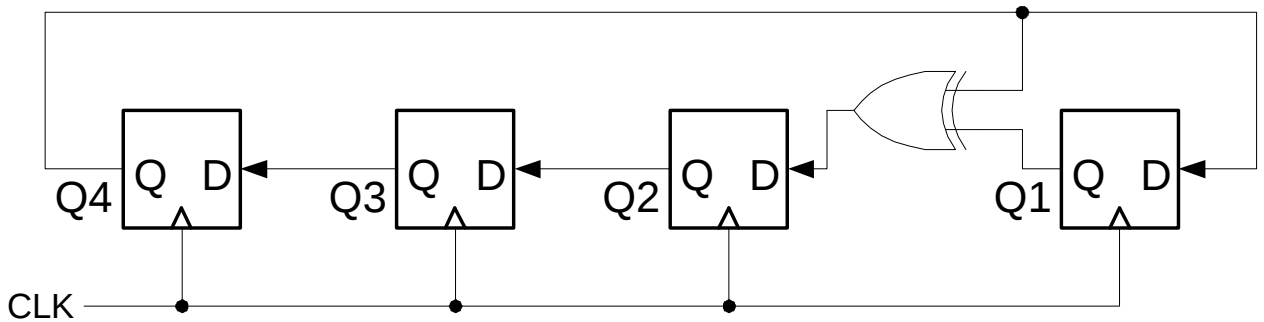
- Introduce LFSR as fancy counter
- Practice of Cyclic Redundancy Checks
  - Burst errors in networks, disks, etc.
- Theory of LFSRs
- Power

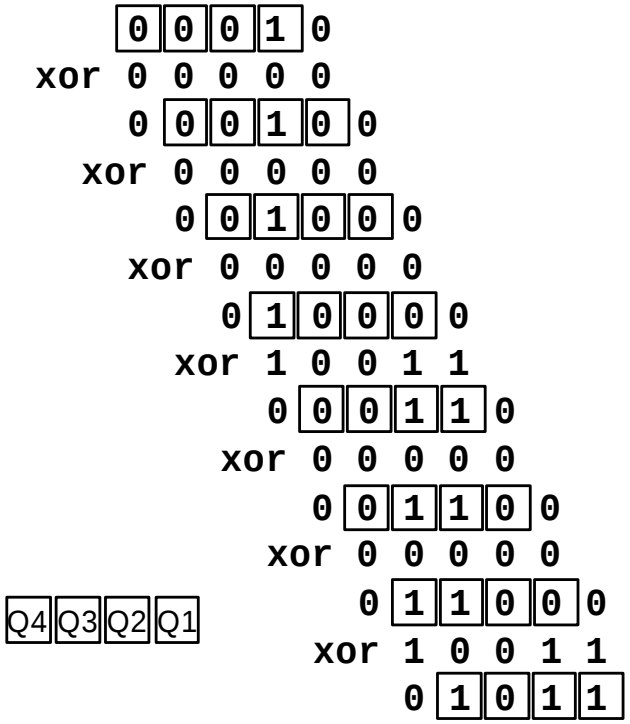
Linear Feedback Shift Registers (LFSRs)

- These are n-bit counters exhibiting *pseudo-random* behavior.
- Built from simple shift-registers with a small number of xor gates.
- Used for:
  - random number generation
  - counters
  - error checking and correction
- Advantages:
  - very little hardware
  - high speed operation
- Example 4-bit LFSR:



4-bit LFSR





0001
0010
0100
1000
0011
0110
1100
1011
0101
1010
0111
1110
1111
1101
1001
0001

- **Circuit counts through 2^4-1 different non-zero bit patterns.**
- **Left most bit determines shiftl or more complex operation**
- **Can build a similar circuit with any number of FFs, may need more xor gates.**
- **In general, with n flip-flops, 2^n-1 different non-zero bit patterns.**
- **(Intuitively, this is a counter that *wraps around* many times and in a strange way.)**

Applications of LFSRs

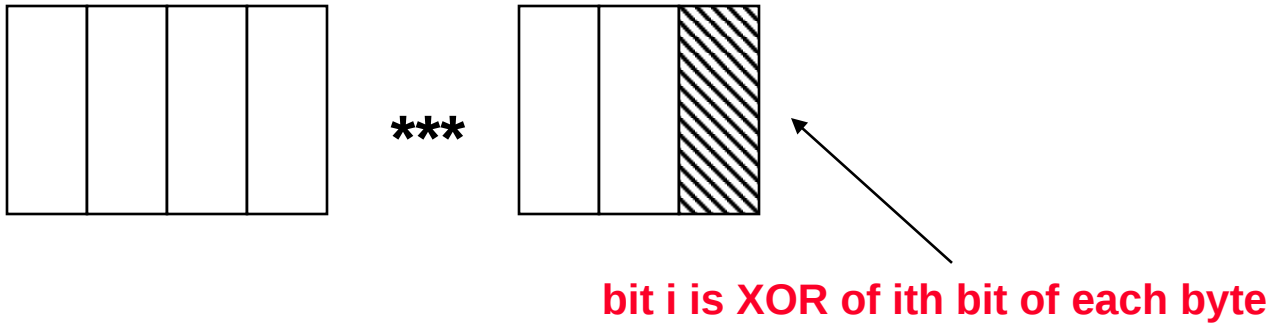
- **Performance:**
  - In general, xors are only ever 2-input and never connect in series.
  - Therefore the minimum clock period for these circuits is:

    $$T > T_{2\text{-input-xor}} + \text{clock overhead}$$

  - Very little latency, and independent of n!
- **This can be used as a fast counter, if the particular sequence of count values is not important.**
  - Example: micro-code micro-pc
- **Can be used as a random number generator.**
  - Sequence is a pseudo-random sequence:
    - » numbers appear in a random sequence
    - » repeats every 2^n-1 patterns
  - Random numbers useful in:
    - » computer graphics
    - » cryptography
    - » automatic testing
- **Used for error detection and correction**
    - » CRC (cyclic redundancy codes)
    - » ethernet uses them

Concept: Redundant Check

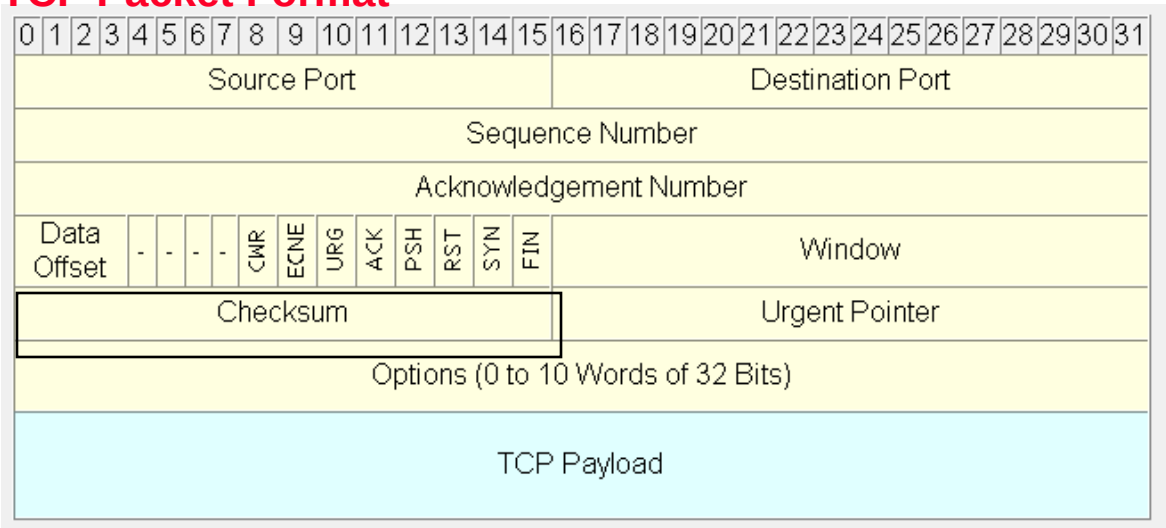
- Send a message M and a “check” word C
- Simple function on <M,C> to determine if both received correctly (with high probability)
- Example: XOR all the bytes in M and append the “checksum” byte, C, at the end
  - Receiver XORs <M,C>
  - What should result be?
  - What errors are caught?



Example: TCP Checksum

7	Application (HTTP,FTP, DNS)
4	Transport (TCP, UDP)
3	Network (IP)
2	Data Link (Ethernet, 802.11b)
1	Physical

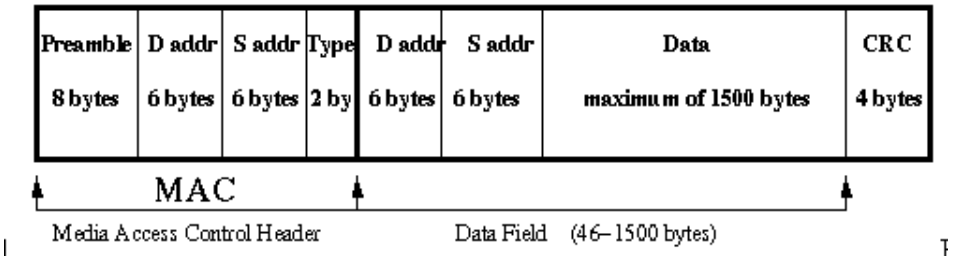
TCP Packet Format

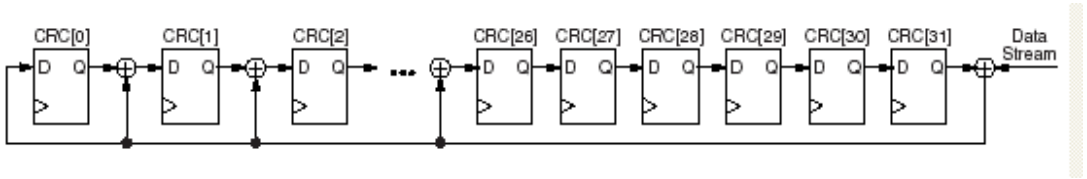


- **TCP Checksum a 16-bit checksum, consisting of the one's complement of the one's complement sum of the contents of the TCP segment header and data, is computed by a sender, and included in a segment transmission. (note end-around carry)**
- **Summing all the words, including the checksum word, should yield zero**

Example: Ethernet CRC-32

7	Application (HTTP,FTP, DNS)
4	Transport (TCP, UDP)
3	Network (IP)
2	Data Link (Ethernet, 802.11b)
1	Physical





CRC concept

- I have a msg polynomial M(x) of degree m
- We both have a generator poly G(x) of degree m
- Let r(x) = remainder of M(x) x^n / G(x)
  - M(x) x^n = G(x)p(x) + r(x)
  - r(x) is of degree n
- What is (M(x) x^n – r(x)) / G(x) ?

n bits of zero at the end

- So I send you M(x) x^n – r(x)
  - m+n degree polynomial
  - You divide by G(x) to check
  - M(x) is just the m most signficant coefficients, r(x) the lower m

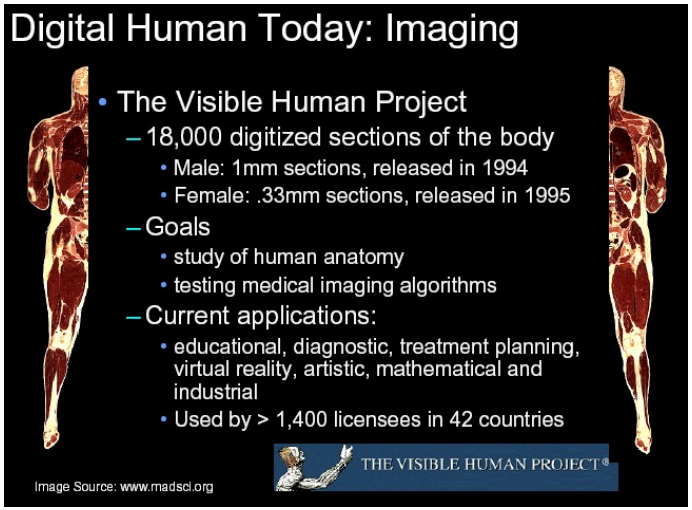
tack on n bits of remainder

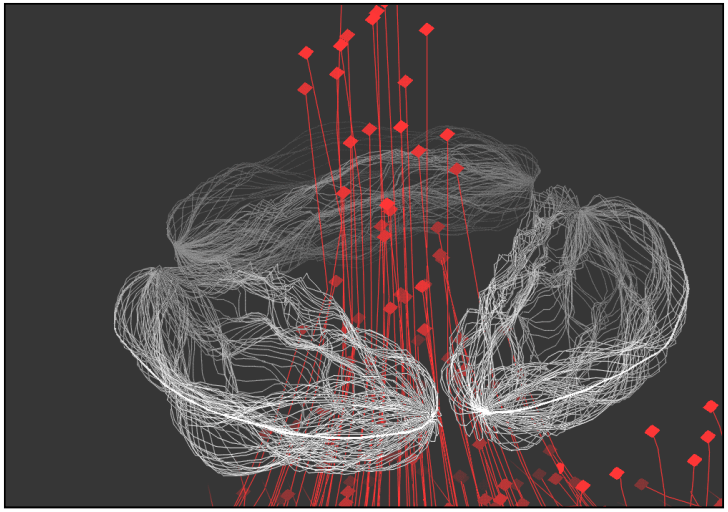
Instead of the zeros

- n-bit Message is viewed as coefficients of n-degree polynomial over binary numbers

Announcements

- **Reading**
  - **XILINX IEEE 802.3 Cyclic Redundancy Check (pages 1-3)**
  - **ftp://ftp.rocksoft.com/papers/crc_v3.txt**
- **Final on 12/15**
- **What's Going on in EECS?**
  - **Towards simulation of a Digital Human**
  - **Yelick: Simulation of the Human Heart Using the Immersed Boundary Method on Parallel Machines**





Galois Fields - the theory behind LFSRs

- **LFSR circuits performs multiplication on a *field*.**
- **A field is defined as a *set* with the following:**
  - **two operations defined on it:**
    - » **"addition" and "multiplication"**
  - **closed under these operations**
  - **associative and distributive laws hold**
  - **additive and multiplicative identity elements**
  - **additive inverse for every element**
  - **multiplicative inverse for every non-zero element**
- **Example fields:**
  - **set of rational numbers**
  - **set of real numbers**
  - **set of integers is *not* a field (why?)**
- **Finite fields are called *Galois* fields.**
- **Example:**
  - **Binary numbers 0,1 with XOR as "addition" and AND as "multiplication".**
  - **Called GF(2).**
  - **0+1 = 1**
  - **1+1 = 0**
  - **0-1 = ?**
  - **1-1 = ?**

Galois Fields - The theory behind LFSRs

- **Consider *polynomials* whose coefficients come from GF(2).**
- **Each term of the form x^n is either present or absent.**
- **Examples: 0, 1, x, x^2, and $x^7 + x^6 + 1$**

  $= 1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$
- **With addition and multiplication these form a field:**
- **"Add": XOR each element individually with no carry:**

$$\begin{array}{r} x^4 + x^3 + \quad + x + 1 \\ + \quad x^4 + \quad + x^2 + x \\ \hline x^3 + x^2 \quad + 1 \end{array}$$

- **"Multiply": multiplying by x^n is like shifting to the left.**

$$\begin{array}{r} x^2 + x + 1 \\ \times \qquad x + 1 \\ \hline x^2 + x + 1 \\ x^3 + x^2 + x \\ \hline x^3 \qquad + 1 \end{array}$$

So what about division (mod)

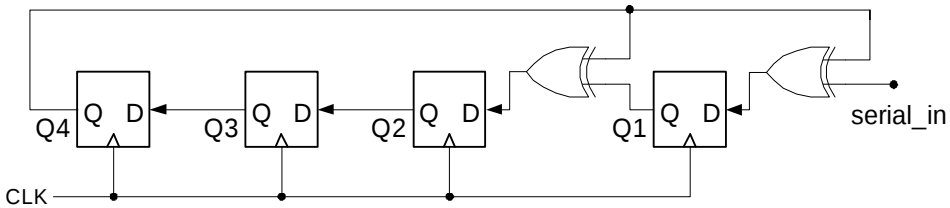
$$\frac{x^4 + x^2}{x} = x^3 + x \quad \text{with remainder 0}$$

$$\frac{x^4 + x^2 + 1}{X + 1} = x^3 + x^2 \quad \text{with remainder 1}$$

$$
\begin{array}{rl}
 & x^3 + x^2 + 0x + 0 \\
X + 1 \;\big|\; & x^4 + 0x^3 + x^2 + 0x + 1 \\
 & x^4 + x^3 \\
\hline
 & x^3 + x^2 \\
 & x^3 + x^2 \\
\hline
 & 0x^2 + 0x \\
 & 0x + 1 \\
\hline
 & \text{Remainder } 1
\end{array}
$$

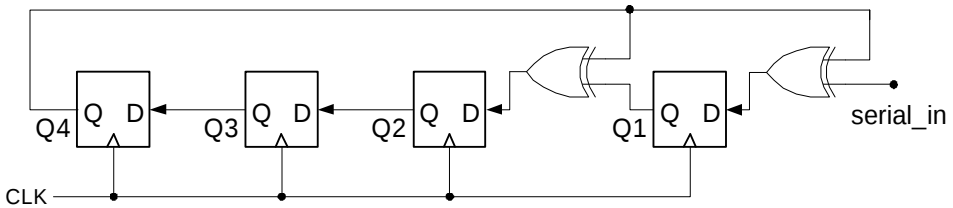
Polynomial division

```
                0 0 0 0 1 0 1
              ┌──────────────
10011         │ 1 0 1 1 0 0 1 0 0 0 0
                1 0 0 1 1
                ---------
                0 0 1 0 1
                  0 1 0 1 0
                    1 0 1 0 1
                    ---------
                    1 0 0 1 1
                    0 0 1 0 0
```



- When MSB is zero, just shift left, bringing in next bit
- When MSB is 1, XOR with divisor and shiftl

CRC encoding



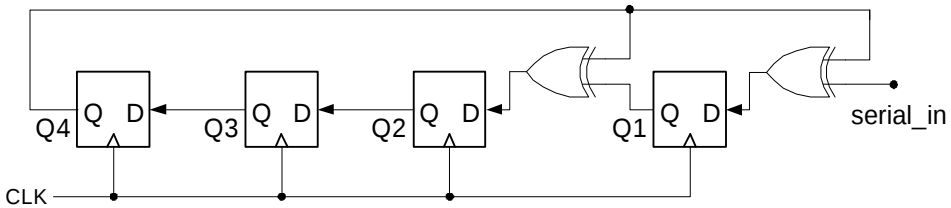
1 0 1 1 0 0 1 0 0 0 0

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

Message sent:

1 0 1 1 0 0 1 1 0 1 0

CRC decoding



1 0 1 1 0 0 1 1 0 1 0

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

Galois Fields - The theory behind LFSRs

- These polynomials form a *Galois (finite)* field if we take the results of this multiplication modulo a prime polynomial $p(x)$.
  - A prime polynomial is one that cannot be written as the product of two non-trivial polynomials $q(x)r(x)$
  - Perform modulo operation by subtracting a (polynomial) multiple of $p(x)$ from the result. If the multiple is 1, this corresponds to XOR-ing the result with $p(x)$.
- For any degree, there exists at least one prime polynomial.
- With it we can form $GF(2^n)$
- Additionally, …
- Every Galois field has a primitive element, α, such that all non-zero elements of the field can be expressed as a power of α. By raising α to powers (modulo $p(x)$), all non-zero field elements can be formed.
- Certain choices of $p(x)$ make the simple polynomial x the primitive element. These polynomials are called *primitive*, and one exists for every degree.
- For example, $x^4 + x + 1$ is primitive. So $\alpha = x$ is a primitive element and successive powers of α will generate all non-zero elements of GF(16). *Example on next slide.*

Galois Fields – Primitives

$\alpha^0 = 1$

$\alpha^1 = x$

$\alpha^2 = x^2$

$\alpha^3 = x^3$

$\alpha^4 = x + 1$

$\alpha^5 = x^2 + x$

$\alpha^6 = x^3 + x^2$

$\alpha^7 = x^3 + x + 1$

$\alpha^8 = x^2 + 1$

$\alpha^9 = x^3 + x$

$\alpha^{10} = x^2 + x + 1$

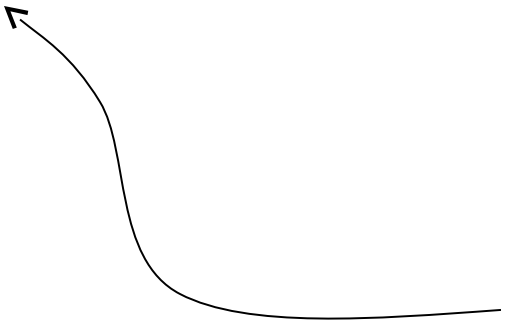
$\alpha^{11} = x^3 + x^2 + x$

$\alpha^{12} = x^3 + x^2 + x + 1$

$\alpha^{13} = x^3 + x^2 + 1$

$\alpha^{14} = x^3 + 1$

$\alpha^{15} = 1$



- **Note this pattern of coefficients matches the bits from our 4-bit LFSR example.**

$$\alpha^4 = x^4 \bmod x^4 + x + 1$$
$$= x^4 \text{ xor } x^4 + x + 1$$
$$= x + 1$$

- **In general finding primitive polynomials is difficult. Most people just look them up in a table, such as:**

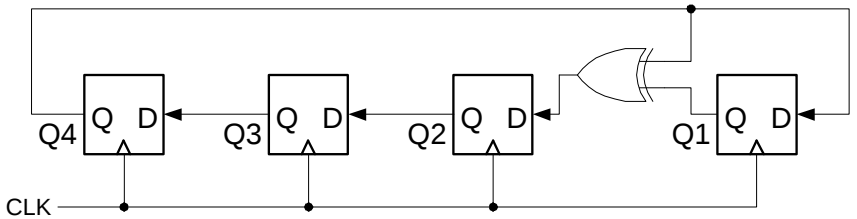
Primitive Polynomials

$x^2 + x + 1$
$x^3 + x + 1$
$x^4 + x + 1$
$x^5 + x^2 + 1$
$x^6 + x + 1$
$x^7 + x^3 + 1$
$x^8 + x^4 + x^3 + x^2 + 1$
$x^9 + x^4 + 1$
$x^{10} + x^3 + 1$
$x^{11} + x^2 + 1$
$x^{12} + x^6 + x^4 + x + 1$
$x^{13} + x^4 + x^3 + x + 1$
$x^{14} + x^{10} + x^6 + x + 1$
$x^{15} + x + 1$
$x^{16} + x^{12} + x^3 + x + 1$
$x^{17} + x^3 + 1$
$x^{18} + x^7 + 1$
$x^{19} + x^5 + x^2 + x + 1$
$x^{20} + x^3 + 1$
$x^{21} + x^2 + 1$
$x^{22} + x + 1$
$x^{23} + x^5 + 1$
$x^{24} + x^7 + x^2 + x + 1$
$x^{25} + x^3 + 1$
$x^{26} + x^6 + x^2 + x + 1$
$x^{27} + x^5 + x^2 + x + 1$
$x^{28} + x^3 + 1$
$x^{29} + x + 1$
$x^{30} + x^6 + x^4 + x + 1$
$x^{31} + x^3 + 1$
$x^{32} + x^7 + x^6 + x^2 + 1$

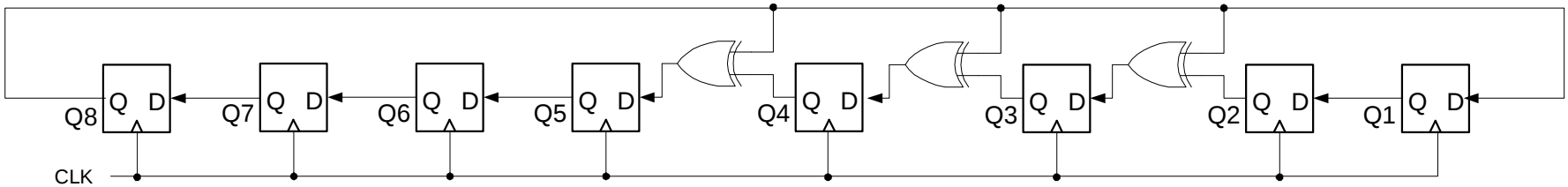
Galois Field		Hardware
Multiplication by x	**⇔**	**shift left**
Taking the result mod $p(x)$	**⇔**	**XOR-ing with the coefficients of $p(x)$ when the most significant coefficient is 1.**
Obtaining all 2^n-1 non-zero elements by evaluating x^k for $k = 1, \ldots, 2^n-1$	**⇔**	**Shifting and XOR-ing 2^n-1 times.**

Building an LFSR from a Primitive Poly

- For *k-bit* LFSR number the flip-flops with FF1 on the right.
- The feedback path comes from the Q output of the leftmost FF.
- Find the primitive polynomial of the form $x^k + \ldots + 1$.
- The $x^0 = 1$ term corresponds to connecting the feedback directly to the D input of FF 1.
- Each term of the form x^n corresponds to connecting an xor between FF n and $n+1$.
- 4-bit example, uses $x^4 + x + 1$
  - $x^4 \Leftrightarrow$ FF4's Q output
  - $x \Leftrightarrow$ xor between FF1 and FF2
  - $1 \Leftrightarrow$ FF1's D input



- To build an 8-bit LFSR, use the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$ and connect xors between FF2 and FF3, FF3 and FF4, and FF4 and FF5.



Generating Polynomials

- **CRC-16: $G(x) = x^{16} + x^{15} + x^2 + 1$**
  - **detects single and double bit errors**
  - **All errors with an odd number of bits**
  - **Burst errors of length 16 or less**
  - **Most errors for longer bursts**
- **CRC-32: $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$**
  - **Used in ethernet**
  - **Also 32 bits of 1 added on front of the message**
    - » **Initialize the LFSR to all 1s**