Applications of LFSRs

- Can be used as a random number generator:
  - Sequence is a pseudo-random sequence
  - Numbers appear in 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?

bit i is XOR of ith bit of each byte

Example: TCP Checksum

- 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

- 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) / G(x)
  - r(x) is of degree n
- What is (M(x) x^n – r(x) / G(x) ?
  - n-bit of zero at the end
- So I send you M(x) x^n – r(x)
  - m+1 degree polynomial
  - You divide by G(x) to check
  - M(x) is just the m most significant coefficients, r(x) the lower m
- n-bit Message is viewed as coefficients of n-degree polynomial over binary numbers

Announcements

- Reading
  - KELBY IEEE 802.3 Cyclic Redundancy Check (pages 1-3)
- 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+1 = 7
      - 1+1 = 7
**Galois Fields - The theory behind LFSRs**

- Consider polynomials whose coefficients come from GF(2).
- Each term of the form $x^i$ is either present or absent.
- Examples: $0,1,x,x^2$, and $x^2 + x^3 + 1$.
  
  \[ x^2 + x^3 + 1 = x^2 + x + 1 + x^2 + x^3 + 1 - x^2 \]

- With addition and multiplication these form a field:
  - "Add": XOR each element individually with no carry:
    \[ x^2 + x^3 + 1 + x^2 + 1 = x^2 + x + 1 \]
    
  - "Multiply": multiplying by $x^i$ is like shifting to the left.
    \[ x^2 + x + 1 \]

**So what about division (mod)**

\[ x^4 + x^2 \equiv x^4 + x \quad \text{with remainder 0} \]

\[ x^4 + x^2 + 1 \equiv x^4 + x^2 \quad \text{with remainder 1} \]

\[ x + 1 \equiv x^4 + 0x + 0 \]

\[ x + 1 \equiv x^4 + 0x + 1 \]

\[ x^4 + x^2 \equiv x^3 + 1 \]

\[ x^4 + x^2 \equiv x^3 + x \]

**Polynomial division**

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

**CRC encoding**

- 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 $p(x)$.
  - 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, $\alpha$, such that all non-zero elements of the field can be expressed as a power of $\alpha$. By raising $\alpha$ 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, $x^4 + x$ is a primitive element and successive powers of $\alpha$ will generate all non-zero elements of $GF(16)$. Example on next slide.
Galois Fields – Primitives

\[
\begin{align*}
\alpha^0 &= 1 \\
\alpha^1 &= \alpha \\
\alpha^2 &= \alpha^2 \\
\alpha^3 &= \alpha^3 \\
\ldots \\
\alpha^k &= 1
\end{align*}
\]

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

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

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 \( x^n + 1 \).
- The \( x^n + 1 \) term corresponds to connecting the feedback directly to the \( D \) input of FF1.
- Each term of the form \( x^i \) corresponds to connecting an xor between FF = and \( x^i \).
- 4-bit example, uses \( x^2 + x + 1 \)
  - \( x^2 \) as \( FF \) of FF output
  - \( x \) as xor between FF1 and FF2
  - \( x^2 \) as FF1’s D input
- To build an \( n \)-bit LFSR, use the primitive polynomial \( x^n + x^i + x^j + x + 1 \) and connect xors between FF2 and FF3, FF3 and FF4, and FF4 and FF5.

\[
\text{Generating Polynomials}
\]

- CR-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: \( G(x) = x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^4 + x + 1 \)
  - Used in ethernet
  - Also 32 bits of 1 added on front of the message
  - Initialize the LFSR to all 1s

Motivation

Why should a digital designer care about power consumption?

- Portable devices:
  - handhelds, laptops, phones, MP3 players, cameras, … all need to run for extended periods on small batteries without recharging
  - Devices that need regular recharging or large heavy batteries will lose out to those that don’t.
- Power consumption important even in “tethered” devices.
  - System cost tracks power consumption:
    - power supplies, distribution, heat removal
    - power conservation, environmental concerns
- In a span of 10 years we have gone from designing without concern for power consumption to (in many cases) designing with power consumption as the primary design constraint!
Battery Technology

- Battery technology has moved very slowly
  - Moore’s law does not seem to apply
- Li-Ion and NiMh still the dominate technologies
- Batteries still contribute significant to the weight of mobile devices

Basics

- **Warning!** In everyday language, the term “power” is used incorrectly in place of “energy.”
- Power is *not* energy.
- Power is *not* something you can run out of.
- Power can *not* be lost or used up.
- It is *not* a thing, it is merely a rate.
- It can *not* be put into a battery any more than velocity can be put in the gas tank of a car.

Power in CMOS

- **Switching Energy:**
  - energy used to switch a node

  ![Switching Energy Diagram]

  **Calculate energy dissipated in pullup:**

  \[
  E_{pu} = \int_{t_i}^{t_f} P(t)\,dt = \int_{V_{dd}}^{(V_{dd} - v)}(V_{dd} - v)\,i(t)\,dt = \int_{V_{dd}}^{V_{dd} - v} (V_{dd} - v) \cdot c (dv/dt)\,dt =
  
  = c V_{dd} \int_{V_f}^{V_i} dv - c \int_{V_f}^{V_i} v \cdot dV = c V_{dd}^2 - 1/2 c V_{dd}^2 \left[1/2 V_{dd}^{-2}\right]
  
  \]

  - Energy supplied
  - Energy stored
  - Energy dissipated

  An equal amount of energy is dissipated on pullup.
### Switching Power

- **Gate power consumption:**
  - Assume a gate output is switching its output at a rate of: 
  
  \[
  P_{\text{avg}} = \frac{E}{\Delta t} = \text{switching rate} \cdot E_{sw} 
  \]
  
  Therefore:
  
  \[
  P_{\text{avg}} = \alpha \cdot f \cdot \frac{1}{2} cV_{dd}^2 
  \]

- **Chip/circuit power consumption:**
  
  \[
  P_{\text{avg}} = n \cdot \alpha_{\text{avg}} \cdot f \cdot \frac{1}{2} cV_{dd}^2 
  \]

  *number of nodes (or gates)*

### Other Sources of Energy Consumption

- **“Short Circuit” Current:**
  
  \[
  I_{\text{ds}}(V_{gs}) = \frac{1}{2} \beta V_{gs}^2 
  \]

- **Device Ids Leakage:**
  
  - Junction Diode Leakage:
    - \( V_{out} = V_{dd} \)
    - \( V_{out} \approx V_{dd} \)
    - \( I_{ds} \) (drain) current
    - \( I_{ds} \) (source) current
    - \( V_{gs} \) (gate-source voltage)
    - \( V_{gs} \) (source-gate voltage)
    - \( N \) (number of nodes)
    - \( K \) (transistor size)
    - \( f \) (clock frequency)
    - \( V_{dd} \) (power supply voltage)
    - \( n \) (number of nodes)
    - \( c \) (channel length)
    - \( \alpha \) (activity factor)
    - \( \beta \) (transconductance)
    - \( \psi \) (channel width)
    - \( \mu \) (mobility)
    - \( \rho \) (resistivity)

### Controlling Energy Consumption

**What control do you have as a designer?**

- **Largest contributing component to CMOS power consumption is switching power:**
  
  \[
  P_{\text{avg}} = n \cdot \alpha_{\text{avg}} \cdot f \cdot \frac{1}{2} cV_{dd}^2 
  \]

- **Factors influencing power consumption:**
  
  - \( n \): total number of nodes in circuit
  - \( \alpha \): activity factor (probability of each node switching)
  - \( f \): clock frequency (does this effect energy consumption?)
  - \( V_{dd} \): power supply voltage
  - What control do you have over each factor?
  - How does each affect the total Energy?

### Power / Cost / Performance

- **Parallelism to trade cost for performance.**
  
  - As we trade cost for performance what happens to energy?
  
  \[
  E_{\text{total}} = E_{\text{gate}} + E_{\text{buf}} + E_{\text{inv}} + E_{\text{add}} + E_{\text{mul}} + E_{\text{mux}} 
  \]

- **The lowest energy consumer is the solution that minimizes cost without time multiplexing operations.**