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

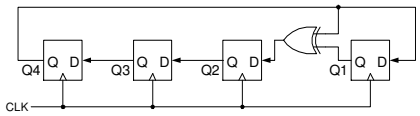
- Concept of error coding
  - Add a few extra bits (enlarges the space of values) that carry information about all the bits
  - Detect: Simple function to check of entire data+check received correctly
    - » Small subset of the space of possible values
  - Correct: Algorithm for locating nearest valid 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

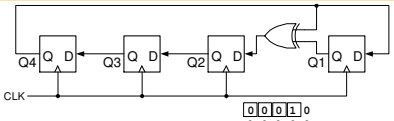
- Introduce LFSR as fancy counter
- Practice of Cyclic Redundancy Checks
  - Burst errors in networks, disks, etc.
- Theory of LFSRs
- Power

Linear Feedback Shift Registers (LFSRs)

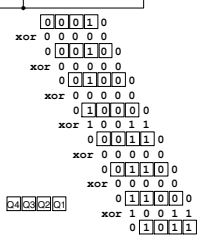
- These are n-bit counters exhibiting *pseudo-random* behavior.
- Built from simple shift-registers with a small number of xor gates.
- Used for:
  - random number generation
  - counters
  - error checking and correction
- Advantages:
  - very little hardware
  - high speed operation
- Example 4-bit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



4-bit LFSR



- 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.)



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

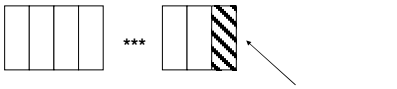
Applications of LFSRs

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

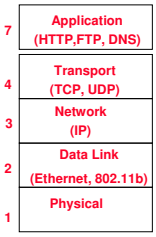
    $T > T_{2\text{-input-xor}}$ + clock overhead
  - Very little latency, and independent of n!
- This can be used as a fast counter, if the particular sequence of count values is not important.
  - Example: micro-code micro-pc
- Can be used as a random number generator.
  - Sequence is a pseudo-random 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

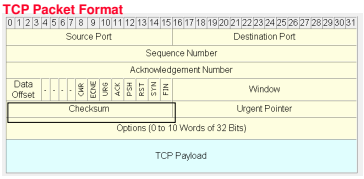
- Send a message M and a "check" word C
- Simple function on <M,C> to determine if both received correctly (with high probability)
- Example: XOR all the bytes in M and append the "checksum" byte, C, at the end
  - Receiver XORs <M,C>
  - What should result be?
  - What errors are caught?



Example: TCP Checksum

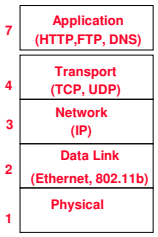


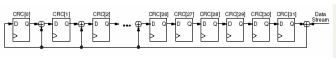
TCP Packet Format



- 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





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

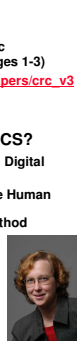
- So I send you M(x) x^n – r(x)
  - m+n degree polynomial
  - You divide by G(x) to check
  - M(x) is just the m most signficant coefficients, r(x) the lower m

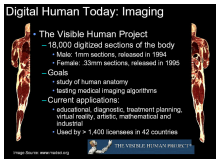
tack on n bits of remainder
Instead of the zeros

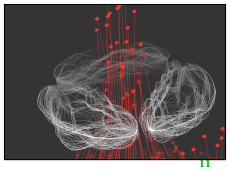
- n-bit Message is viewed as coefficients of n-degree polynomial over binary numbers

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.

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

13

So what about division (mod)

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

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

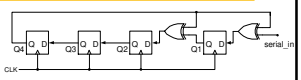
$$\begin{array}{r|l} & x^3 + x^2 + 0x + 0 \\ \hline x + 1 & x^4 + 0x^3 + x^2 + 0x + 1 \\ & x^4 + x^3 \\ \hline & x^3 + x^2 \\ & x^3 + x^2 \\ \hline & 0x^2 + 0x \\ & 0x + 1 \end{array}$$

Remainder 1

14

Polynomial division

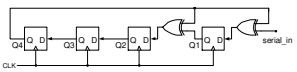
$$\begin{array}{r} 0\,0\,0\,0\,1\,0\,1 \\ 10011 \;\overline{)\; 1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0} \\ 1\,0\,0\,1\,1 \\ \hline 0\,0\,1\,0\,1 \\ 0\,1\,0\,1\,0 \\ 1\,0\,1\,0\,1 \\ \hline 1\,0\,0\,1\,1 \\ 0\,0\,1\,0\,0 \end{array}$$



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

15

CRC encoding



1 0 1 1 0 0 1 0 0 0 0

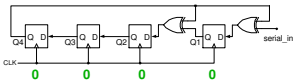
0	0	0	0	
0	0	0	1	0 1 1 0 0 1 0 0 0 0
0	0	1	0	1 1 0 0 1 0 0 0 0
0	1	0	1	1 0 0 1 0 0 0 0
1	0	1	1	0 0 1 0 0 0 0
0	1	0	1	0 1 0 0 0 0
1	0	1	0	1 0 0 0 0
0	1	1	0	0 0 0 0
1	1	0	0	0 0 0
1	0	1	1	0 0
0	1	0	1	0
1	0	1	0	

Message sent:

1 0 1 1 0 0 1 1 0 1 0

16

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	

17

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.*

18

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:

19

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

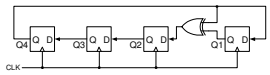
Taking the result mod $p(x)$ ⇔ XOR-ing with the coefficients of $p(x)$ when the most significant coefficient is 1.

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.

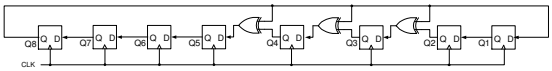
20

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 ⇔ FF4's Q output
  - x ⇔ xor between FF1 and FF2
  - 1 ⇔ 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.



21

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

22

POWER

23

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!***

24

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

Nokia 61xx - 33%

Handspring PDA - 10%

Toshiba Portege 3110 laptop - 20%

Basics

- Power supply provides energy for charging and discharging wires and transistor gates. The energy supplied is stored and dissipated as heat.

$P \equiv dw/dt$ *Power: Rate of work being done w.r.t time. Rate of energy being used.*

Units: $P = E/\Delta t$ *Watts = Joules/seconds*

- If a differential amount of charge ***dq*** is given a differential increase in energy ***dw***, the potential of the charge is increased by: $V = dw/dq$
- By definition of current: $I = dq/dt$

$$dw/dt = \frac{dw}{dq} \times \frac{dq}{dt} = \boxed{P = V \times I}$$ A very practical formulation!

$$w = \int_{-\infty}^{t} P dt \quad \textit{total energy}$$ If we would like to know total energy

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.

Metrics

How do we measure and compare power consumption?

- One popular metric for microprocessors is: MIPS/watt
  - MIPS, millions of instructions per second.
    - » Typical modern value?
  - Watt, standard unit of power consumption.
    - » Typical value for modern processor?
  - MIPS/watt is reflective of the tradeoff between performance and power. Increasing performance requires increasing power.
  - Problem with "MIPS/watt"
    - » MIPS/watt values are typically not independent of MIPS
      - techniques exist to achieve very high MIPS/watt values, but at very low absolute MIPS (used in watches)
    - » Metric only relevant for comparing processors with a similar performance.
  - One solution, $MIPS^2/watt$. Puts more weight on performance.

Metrics

- How does MIPS/watt relate to *energy*?
- Average power consumption = energy / time

MIPS/watt = instructions/sec / joules/sec = instructions/joule

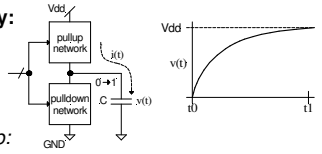
  - therefore an equivalent metric (reciprocal) is energy per operation (E/op)

- E/op is more general - applies to more than processors
  - also, usually more relevant, as batteries life is limited by total energy draw.
  - This metric gives us a measure to use to compare two alternative implementations of a particular function.

Power in CMOS

Switching Energy:

energy used to switch a node



Calculate energy dissipated in pullup:

$$E_{sw} = \int_{t_0}^{t_1} P(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t)dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c\,(dv/dt)\,dt =$$

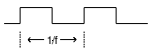
$$= cV_{dd} \int_{t_0}^{t_1} dv - c\int_{t_0}^{t_1} v \cdot dv = cV_{dd}^{\ 2} - 1/2\,cV_{dd}^{\ 2} = \boxed{1/2\,cV_{dd}^{\ 2}}$$

Energy supplied *Energy stored* *Energy dissipated*

An equal amount of energy is dissipated on pulldown.

Switching Power

- **Gate power consumption:**
  - **Assume a gate output is switching its output at a rate of:**



$\alpha \cdot f$

activity factor — *clock rate*

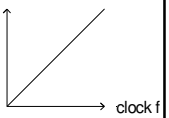
(probability of switching on any particular clock period)

$$P_{avg} = E/\Delta t = switching\ rate \cdot E_{sw}$$

Therefore:

$$P_{avg} = \alpha \cdot f \cdot 1/2\, cV_{dd}^{\ 2}$$

Pavg



- **Chip/circuit power consumption:**

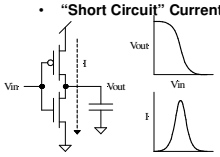
$$P_{avg} = n \cdot \alpha_{avg} \cdot f \cdot 1/2\, c_{avg} V_{dd}^{\ 2}$$

number of nodes (or gates)

31

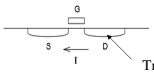
Other Sources of Energy Consumption

- **"Short Circuit" Current:**



10-20% of total chip power

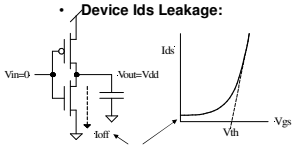
- **Junction Diode Leakage:**



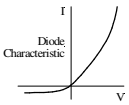
~1nWatt/gate
few mWatts/chip

Transistor drain regions "leak" charge to substrate.

- **Device Ids Leakage:**



Transistor s/d conductance never turns off all the way. ~3pWatts/transistor. ~1mWatt/chip *Low voltage* processes much worse.



32

Controlling Energy Consumption

What control do you have as a designer?

- **Largest contributing component to CMOS power consumption is switching power:**

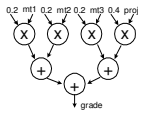
$$P_{avg} = n \cdot \alpha_{avg} \cdot f \cdot 1/2\, c_{avg} V_{dd}^{\ 2}$$

  - **Factors influencing power consumption:**
    - **n: total number of nodes in circuit**
    - **α: 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 effect the total Energy?**

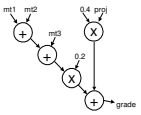
33

Power / Cost / Performance

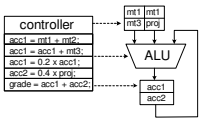
- **Parallelism to trade cost for performance. As we trade cost for performance what happens to energy?**



$4\ E_{MUL} + 3\ E_{ADD} + E_{WIRES}$



$2\ E_{MUL} + 3\ E_{ADD} + E_{WIRES}$



$2\ E_{MUL} + 3\ E_{ADD} + E_{MUXES} + E_{CNTL} + E_{WIRES}$

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

34