Cyclic Redundancy Checks

Will Hasenplaugh
Intel Corporation

October 16, 2006
Outline

• Introduction to Error Control Systems
• Properties and Examples
• Cyclic Redundancy Checks
• Homework
  – Break
• Linear Time CRC Circuit
• Logarithmic Time CRC Circuit
Errors

- Errors occur due to noise or interference on a communication channel.
- Error detection and correction codes are used to increase the \textit{effective utility} of the channel.
  - forward error correction uses redundant bits to \textit{fix} errors
  - error detection exploits \textit{retransmission (ARQ)} to improve communication fidelity
- Usually, error correction codes are used for bit errors and ARQ is used for entire packets.
Channel Coding

- **Error Detecting Codes**
  - **Parity**
    - Most widely used error control mechanism
    - Detects all odd weight errors
  - **Cyclic Redundancy Check (CRC)**
    - Robust error detection mechanism used ubiquitously w/ ARQ

- **Error Correcting Codes**
  - **Block Codes**
  - \( m(x) \) (k input bits) -> \( c(x) \) (n output bits); an “(n,k) code”
    - memoryless process, simple mapping
    - code rate \( R = k/n \) [typical values are \( .25 < R < .875 \)]
    - Hamming Codes, Golay Codes, Low-Density Parity-Check codes etc.
  - **Convolutional Codes**
    - Viterbi decoding
    - Turbo Coding achieves close to Shannon limit via a pair of interleaved convolutional codes, effectively an iterative Bayesian update algorithm.
Where is Error Correction / Detection used?

• **Error Detection**
  - Communication networks
    - Wireless networks
    - Internet etc.
  - Anywhere retransmission is possible and the probability of bit error is relatively low.

• **Error Correction**
  - In storage media (ram, disk, tape, CDs etc.)
  - In memories in processors to combat ‘dead’ memory cells to improve processor yield.
  - Anywhere a retransmission is either inconvenient or impossible.
Outline

• Introduction to Error Control Systems
• Properties and Examples
• Cyclic Redundancy Checks
• Homework
  – Break
• Linear Time CRC Circuit
• Logarithmic Time CRC Circuit
Basics

• Hamming weight is the number of non-zero coefficients in a polynomial over GF(2)

• Hamming distance between two words is the Hamming weight of their sum

• The minimum distance, $d_{min}$, of a code, $C$, is the smallest Hamming distance between any two codewords in $C$.

• A pattern of $t$ or fewer errors can be detected if:
  • $d_{min} \geq t+1$ (transmission is not a valid codeword -> error)

• A pattern of $t$ or fewer errors can be detected and corrected if:
  • $d_{min} \geq 2t+1$ (use closest codeword)
Parity

• Starting with $n-1$ message bits, construct the $n$th bit such that the Hamming weight of the codeword is even.
• Will only detect an odd number of bit errors
• Does not correct any bit errors
• Relies on ARQ for correct transmission

$n$ bit packet – $c(x)$

message – $m(x)$
$k$ bits

parity
1 bit

xor
Linear Block Codes

• Syndrome Decoding
  • Transmit
    – $m$ is a $k$ bit message and $c$ is a $n$ bit codeword
    – $c = mG$, where $G$ is a $k$ by $n$ bit generator matrix
    – $G = [ Z^T \mid I ]$, $Z$ is a matrix that defines the linear block code.
  • Receive
    – We receive a codeword plus an $n$ bit error vector, $e$
    – $H = [ I \mid Z ]$, is the $n-k$ by $n$ bit Parity Check Matrix
    – $s = (c+e)H^T$, is the $n-k$ bit syndrome
    – If $s$ is non-zero, then an error is detected
    – Thus, codewords are in the null space of $H$
Three Properties of Linear Block Codes

• Property One:
  - The linear combination of any set of codewords is a codeword. So, linear block codes always contain the zero vector.

• Property Two:
  - $d_{\text{min}}$ is equal to the weight of the lowest weight non-zero codeword.

• Property Three:
  - Undetectable error patterns for a linear code are independent of the transmitted code word and are themselves codewords. Thus, a linear block code can detect any error pattern of weight less than $d_{\text{min}}$. 
Redundancy - The Hamming Bound

$r = n-k$ is the number of redundant bits in an $n$ bit codeword. A $t$-error correcting code of length $n$ must have redundancy $r$ that satisfies:

$$r \geq \log_2 V(n,t)$$

$$V(n,t) = \sum_{i=0}^{t} \binom{n}{i}$$

**Proof:** Each of the $M$ codewords in $C$ is associated with a Hamming sphere of radius $t$. The spheres do not overlap, and the total volume of the spheres $C$ cannot exceed the total number of vectors in the space of all possible received words. It follows that:

$$MV(n,t) \leq 2^n$$

$$r = n - \log_2 M \geq \log_2 V(n,t)$$
Redundancy – The Gilbert Bound

There exists a $t$-error correcting code of length $n$ and redundancy $r$ that satisfies:

$$r \leq V(n,2t)$$

**proof:** A $t$-error correcting code $C$ is to be constructed by randomly selecting vectors one at a time from the vector space of binary $n$-tuples. When a code word is selected, all surrounding vectors that are Hamming distance $2t$ away from the selected code word are removed from further consideration. This ensures that the resulting code has $d_{min} = 2t+1$ and is thus $t$-error correcting. The selection of each code word results in the deletion of at most $V(n,2t)$ vectors from the space of $n$-tuples. It follows that at least $M$ codewords are selected, where:

$$M = \left\lceil \frac{2^n}{V(n,2t)} \right\rceil \geq \frac{2^n}{V(n,2t)}$$
Redundancy - Single Error Correction

\[ R = \frac{n-r}{n} \quad n \xrightarrow{\infty} 1 \]

Where:
- \( R \) is the code rate.
- \( n \) is the total number of symbols.
- \( r \) is the number of redundant symbols.

The graph shows the Hamming Bound and Gilbert Bound for different values of \( n \) and \( r \).
Hamming Codes

- Special linear block code with $d_{\text{min}} = 3$
  - $d_{\text{min}} \geq 2t+1$, $t=1$
  - Single error correction
  - $(n,k)=(2^m-1, 2^m-1-m)$, $m>2$

- Syndrome decoding
  - Hamming codes have a convenient decoding property:
    - If the syndrome, $s$, is non-zero, find the column of $H$ that equals $s$ and the index of that column corresponds to the position of the bit error.

- **Perfect** codes meet the Hamming Bound with equality.
  - Only Golay, Hamming and Repetition codes are perfect.
  - Perfect codes are not necessarily the most powerful, they merely define non-overlapping Hamming spheres that ‘perfectly’ fill an $n$-dimensional vector space.
Outline

• Introduction to Error Control Systems
• Properties and Examples
• **Cyclic Redundancy Checks**
• Homework
  - **Break**
• Linear Time CRC Circuit
• Logarithmic Time CRC Circuit
Cyclic Redundancy Check (CRC) Encoding

$m(x)$ is a $k$ bit message
$g(x)$ is an $n-k$ bit generator
$d(x)=m(x)x^{n-k} \mod g(x)$
= is an $n-k$ bit residue
$c(x)=m(x)x^{n-k} + d(x)$
= is an $n$ bit codeword
thus, $g(x) | c(x)$
$e(x)$ is an $n$ bit error pattern
we receive $c(x)+e(x)$
$g(x) | c(x)+e(x) \rightarrow$ undetected error
Properties of $g(x)$ – Odd Weight Errors

In order for an error to go undetected, the $n$ bit error pattern, $e(x)$, must be divisible by $g(x)$.

If $g(x)$ has more than one non-zero term, it cannot evenly divide $x^i$ (for some $i<n$). All non-trivial CRC polynomials do, thus all one bit errors are detectable.

If $g(x)$ has $(x+1)$ as a factor, then all codewords have even parity and odd weight error patterns are thus detectable.
Properties of $g(x)$ – Two Bit Errors

If $g(x)$ does not have $x$ as a factor, and does not evenly divide $(x^i+1)$, it can detect all two bit errors.

It is common practice to select $g(x) = (x+1)b(x)$ where $b(x)$ is primitive (ie - $b(x)$ is irreducible and does not divide $(x^i+1)$ for $i < 2^{|b(x)|-1}$).

If $b(x)$ is irreducible, it clearly does not have $x$ as a factor.

If, additionally, $b(x)$ is primitive, it does not divide $(x^i+1)$, provided $n < 2^{|b(x)|-1}$.
Outline

- Introduction to Error Control Systems
- Properties and Examples
- Cyclic Redundancy Checks

**Homework**
- *Break*

- Linear Time CRC Circuit
- Logarithmic Time CRC Circuit
Let $g(x) = (x+1)b(x)$ ($b(x)$ is primitive) be an $r+1$ bit generator for a CRC code, which encodes $k=2^{r-1}-r-1$ bit messages.

- Show that $g(x)$ generates a single error correcting code.
- Show that it is not a two bit error correcting code.
- Show that two bit errors and one bit errors are distinguishable.
Homework #2

For simplicity assume that $g(x)$ can detect all odd bit errors and all two bit errors; All other errors can be detected with probability $2^{-r}$.

- Using the ARQ (retransmission) protocol (ie – retransmit upon detecting an error), what is the undetected error probability and code rate, $R$, over the Binary Symmetric Channel with crossover probability $p$ (ie – each bit is independent and toggles with probability $p$)?

- Find the undetected error probability and code rate, $R$, if one were to correct all detected one bit errors and retransmit all other errors?

- Find the undetected error probability and code rate, $R$, for Parity with ARQ?
Outline

• Introduction to Error Control Systems
• Properties and Examples
• Cyclic Redundancy Checks
• Homework
  – Break
• Linear Time CRC Circuit
• Logarithmic Time CRC Circuit
Binary Polynomial Reduction in Silicon

It is convenient for a CRC encoder / decoder to operate on data words of size $2^L$, for some $L$. Because the residue, $d(x)$, is linearly composable, we can operate on conveniently sized words and assemble the results as we go. Here is a conceptual diagram of a typical circuit:
Linear Time CRC Circuit

Ex: 9-bit CRC polynomial, reducing 1 bit of the message

Adds over GF(2) are XORs, Multiplications in GF(2) become ANDs.

We assume that $g(x)$ has the top bit set. So, if the top bit of the message is set, we want to add $g(x)$, if it isn’t we add zero.

Notice that every input bit to this circuit travels through one AND gate and one XOR gate.
Numerical Example

\[ g(x) = x^8 + x^7 + x^6 + x^4 + x^2 + 1 \]
\[ m(x) = x^7 + x^3 + x \]
\[ d(x) = m(x)x^8 \mod g(x) \]

The grey box represents the intermediate value of \( d(x) \) after reducing the message by one bit. A programmable CRC circuit may not assume anything about \( g(x) \), other than that the top bit is set.
Parallel CRC Circuit

Each slice successively reduces the ‘current’ MSB by the generator, $g(x)$, reducing the span of the message by one bit.

The inputs to each slice are dependent on the outputs of the slice above it, thus the total gate delay is linear in the number of slices.
Outline

• Introduction to Error Control Systems
• Properties and Examples
• Cyclic Redundancy Checks
• Homework
  – Break
• Linear Time CRC Circuit
• Logarithmic Time CRC Circuit
Some New Polynomials

We begin by deriving some additional polynomials from the \( r+1 \) bit generator, \( g(x) \).

\[
g_i(x) = x^{r+i} + [x^{r+i} \mod g(x)]
\]

\[
g_0(x) = g(x)
\]

Each \( g_i(x) \) has the form: one followed by \( i \) zeroes followed by an \( r \) bit non-zero remainder. Also, notice that \( g(x) \) divides \( g_i(x) \).

Example:

\[
\begin{align*}
g_8 &= 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \\
g_4 &= 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \\
g_2 &= 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \\
g_1 &= 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\
g_0 &= 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1
\end{align*}
\]
A Convenient Recurrence

We define a recurrence, where each at each stage we successively reduce $d_0(x)$ by a smaller degree polynomial, $g_i(x)$, which is a multiple of the generator, $g(x)$. So, each $d_i(x)$ is congruent to $d_0(x) \mod g(x)$ and $d_{L+1}(x)$ is clearly equal to $d(x)$.

$$d_0(x) = m(x) \cdot x^r$$
$$d_i(x) = [d_{i-1}(x) \mod g_{2^{L-i}}(x)] \quad 0 < i \leq L$$
$$d_{L+1}(x) = [d_L(x) \mod g(x)]$$
Logarithmic Time Circuit

We can exploit the form of the derived polynomials, $g_i(x)$, to make the linear time circuit run faster.

Due to the fact that the $i$ zeroes that follow the leading one (in bold) will have no effect on the outputs, we can eliminate those AND and XOR gates.
We can aggregate the bit slices from the last slide to make one stage that runs in logarithmic time.

The conceptual picture above illustrates how each selective addition of $g_4(x)$ can be summed concurrently. Thus, the input operands to each XOR operation are available at the clock edge, allowing us to collect the XORs in a binary tree, as shown to the left.
• Ultimately, we cascade stages like the one in the previous slide, where each stage reduces half of the remaining bits in parallel.
• The critical paths for each stage are the highest order bits, which feed into the AND gates.
• Fortunately, those inputs are available from the previous stage comparatively early.
Gate Delays for $2^L = 256$, $r = 64$
Performance

The overall delay of the Fast CRC Circuit is:
\[ \sim = 4.5^L - 15 \]

The overall delay of the Slow CRC Circuit is:
\[ \sim = 2^L \]

As communication rates increase, the benefit of this fast circuit becomes more pronounced.
Questions?