Segundo Trabalho de ELD 2019

Construção de um verificador de CRC de 32 *bits*

Introdução

Os CRCs (*Cyclic Redundancy Check*) são baseados na teoria dos códigos cíclicos de correção de erros. O uso de códigos cíclicos sistemáticos, que codificam mensagens adicionando um valor de verificação de comprimento fixo, para fins de detecção de erros em redes de comunicação, foi proposto pela primeira vez por W. Wesley Peterson em 1961. Os códigos cíclicos não são apenas simples de implementar, mas têm o benefício de serem particularmente adequados para a detecção de erros de intermitência: sequências contíguas de símbolos de dados errados nas mensagens. Isso é importante porque os erros de *burst* são erros de transmissão comuns em muitos canais de comunicação, incluindo dispositivos de armazenamento magnético e óptico. Normalmente, um CRC de *n bits* aplicado a um bloco de dados de comprimento arbitrário detectará qualquer erro de rajada não superior a *n bits*, e a fração de todos os erros maiores que ele detectará é $(1 - 2^{-n})$.

A especificação de um código CRC requer a definição do chamado polinômio gerador. Esse polinômio se torna o divisor em uma divisão longa polinomial, que recebe a mensagem como dividendo e na qual o quociente é descartado e o restante se torna o resultado. A ressalva importante é que os coeficientes polinomiais são calculados de acordo com a aritmética de um campo finito, para que a operação de adição sempre possa ser executada em paralelo *bit* a *bit* (não há transporte entre dígitos).

Um CRC é chamado de CRC de *n bits* quando seu valor de verificação é de *n bits*. Para um dado *n*, são possíveis vários CRCs, cada um com um polinômio diferente. Esse polinômio possui o grau mais alto *n*, o que significa que possui *n* + 1 termos. Em outras palavras, o polinômio tem um comprimento de *n* + 1; sua codificação requer *n* + 1 *bits*. Observe que a maioria das especificações polinomiais descarta o MSB ou LSB, pois sempre são 1. O CRC e o polinômio associado geralmente têm um nome no formato CRC-n-XXX.

O sistema mais simples de detecção de erros, o *bit* de paridade, é de fato um CRC de 1 *bit*: ele usa o polinômio gerador *x* + 1 (dois termos) e tem o nome CRC-1.

Um dispositivo habilitado para CRC calcula uma sequência binária curta e de comprimento fixo, conhecida como valor de verificação ou CRC, para cada bloco de dados a ser enviado ou armazenado e o anexa aos dados, formando uma palavra de código.

Quando uma palavra de código é recebida ou lida, o dispositivo compara seu valor de verificação com um recém-calculado a partir do bloco de dados ou, equivalentemente, executa um CRC em toda a palavra de código e compara o valor de verificação resultante com uma constante de resíduo esperada.

Se os valores CRC não coincidirem, o bloco conterá um erro de dados. O dispositivo pode tomar medidas corretivas, como reler o bloco ou solicitar que ele seja enviado novamente. Caso contrário, presume-se que os dados estão livres de erros (embora, com alguma pequena probabilidade, possam conter erros não detectados; isso é inerente à natureza da verificação de erros)

Cálculo do CRC

Para calcular um CRC binário de *n bits*, alinhe os *bits* que representam a entrada em uma linha e posicione o padrão de *bits* (*n* + 1) que representa o divisor do CRC (chamado de "polinômio") embaixo do lado esquerdo da linha .

Neste exemplo, codificaremos 14 bits de mensagem com um CRC de 3 *bits*, com um polinômio $x^3 + x + 1$. O polinômio é escrito em binário como coeficientes; um polinômio de terceiro grau possui 4 coeficientes ($1x^3 + 0x^2 + 1x + 1$). Nesse caso, os coeficientes são 1, 0, 1 e 1. O resultado do cálculo é de 3 *bits*.

Começamos com a mensagem a ser codificada (apenas os coeficientes do polinômio):

```
11010011101100
```

A mensagem é preenchida com zeros à direita correspondentes ao comprimento do CRC. Isso é feito para que a palavra-código resultante esteja na forma sistemática. Aqui está o primeiro cálculo para obterr um CRC de 3 *bits*:

```
11010011101100 000 <--- entrada preenchida com 3 bits
1011               <--- divisor (4 bits) = x³ + x + 1
------------------
01100011101100 000 <--- resultado
```

O algoritmo atua nos *bits* diretamente acima do divisor em cada etapa. O resultado dessa iteração é o XOR *bit* a *bit* do divisor polinomial com os *bits* acima dele. Os *bits* que não estão acima do divisor são simplesmente copiados diretamente abaixo para essa etapa. O divisor é então deslocado um pouco para a direita e o processo é repetido até que o divisor atinja a extremidade direita da linha de entrada. Aqui está o cálculo inteiro:

```
11010011101100 000 <--- entrada preenchida com 3 bits
1011               <--- divisor (4 bits) = x³ + x + 1
01100011101100 000 <--- resultado
                        (note os primeiros 4 bits são o XOR com o
                        divisor, o restante dos bits ficam
                        inalterados)
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000 <--- note que o divisor salta para se alinhar
                        com o próximo 1 do dividendo (desde que
                        o quociente para aquele passo seja zero)
       1011             (em outras palavras, não necessariamente
                        se move um bit por iteração)
00000000110100 000
        1011
```

```
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
00000000000000 100 <--- resto (3 bits).

Algoritmo de Divisão para quando o dividendo é igual a zero.
```

Esses *n bits* à direita são o restante da etapa de divisão e também serão o valor da função CRC.A validade de uma mensagem recebida pode ser facilmente verificada executando o cálculo acima novamente, desta vez com o valor de verificação adicionado em vez de zeros. O restante deve ser igual a zero se não houver erros detectáveis.

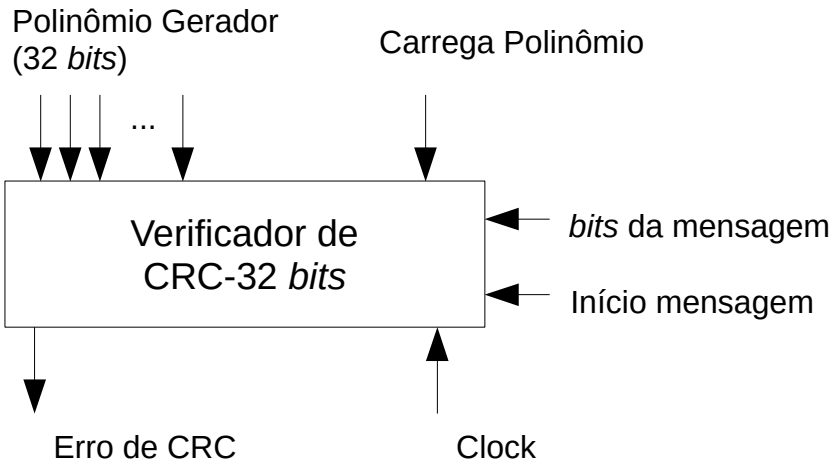
```
11010011101100 100 <--- entrada com CRC
1011               <--- divisor
01100011101100 100 <--- resultado
 1011              <--- divisor ...
00111011101100 100

......

00000000001110 100
          1011
00000000000101 100
           101 1
------------------
00000000000000 000 <--- resto
```

Objetivo

A turma se dividirá em grupos de 3 alunos para construir no **Logisim**, um circuito sequencial para a verificação de um valor de CRC de 32 *bits* para mensagens original de 500 *bits* de comprimento. O **polinômio gerador** é carregado de forma paralela em um registrador, através de um sinal específico de carregamento, a mensagem chegará de forma serial, um *bit* a cada pulso do relógio e será ativado um sinal de **Início de mensagem**, junto com o primeiro *bit* de mensagem recebido. O circuito deverá emitir um sinal de **erro de CRC**, caso encontre algum erro detectável.



Data de Entrega

O trabalho deverá ser entregue impreterivelmente até o dia 04/12/2019 pelo Classroom. O relatório deverá vir em formato PDF, contando os desenhos dos circuitos e todo o desenvolvimento, além de uma descrição do funcionamento. Os arquivos do projeto deverão vir completos e comprimidos em formato ZIP.