SISTEMAS DE COMUNICACIONES DIGITALES

CODIFICACIÓN Y DETECCIÓN DE ERRORES

Métodos de detección de errores

- Las redes deben ser capaces de transferir datos de un dispositivo a otro con total exactitud, si los datos recibidos no son idénticos, el sistema de comunicación es inútil.

- Sin embargo, siempre que se transmiten de un origen a un destino, se puede corromper en el camino.

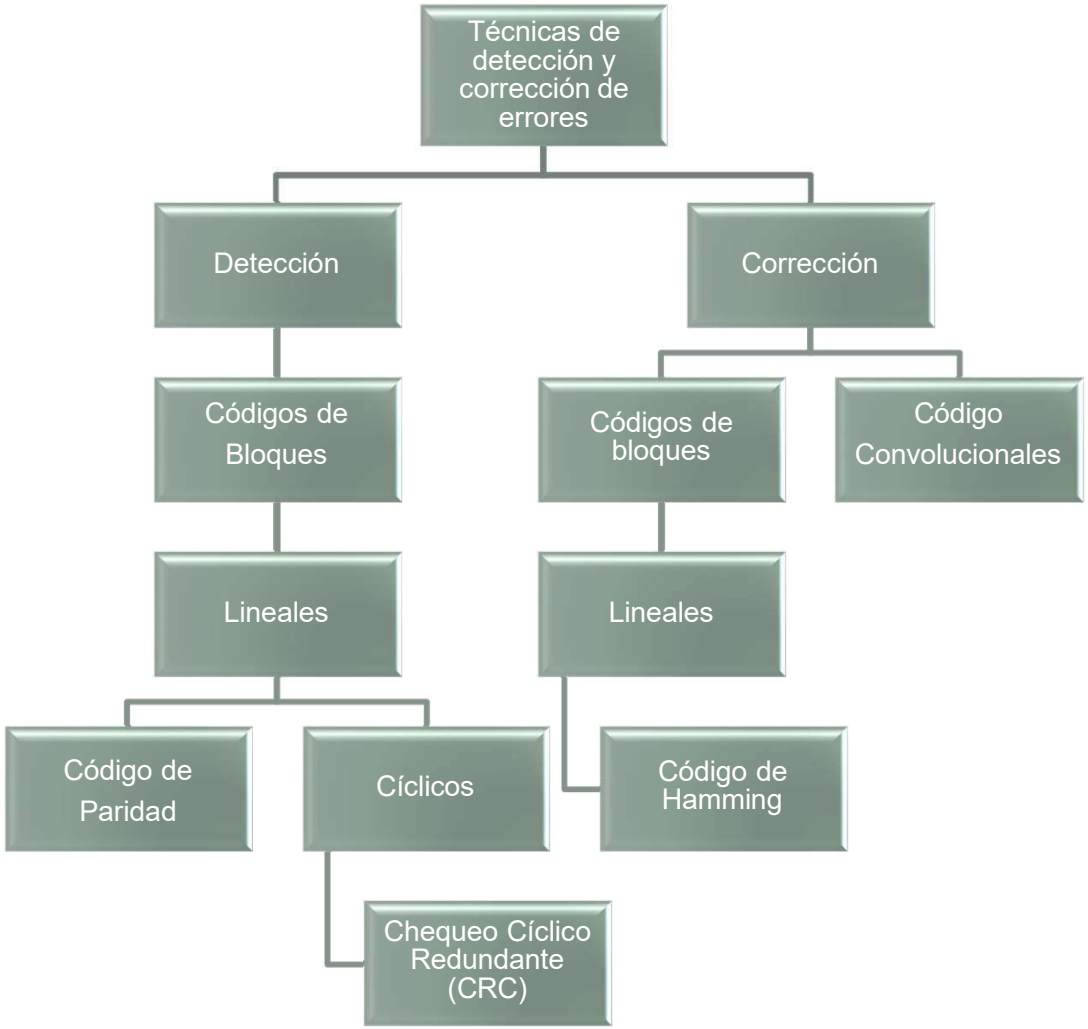
- Consiste en monitorear la información recibida y a través de técnicas implementadas en el Codificador de Canal, éstas técnicas permiten determinar si un carácter o grupo de bits presentan errores.

Métodos de detección de errores

- Los sistemas de comunicación deben tener mecanismos para detectar y corregir errores que alteren los datos recibidos debido a múltiples factores de la transmisión

- La detección y corrección de errores se implementa bien en el nivel de enlace de datos.

Métodos de detección de errores



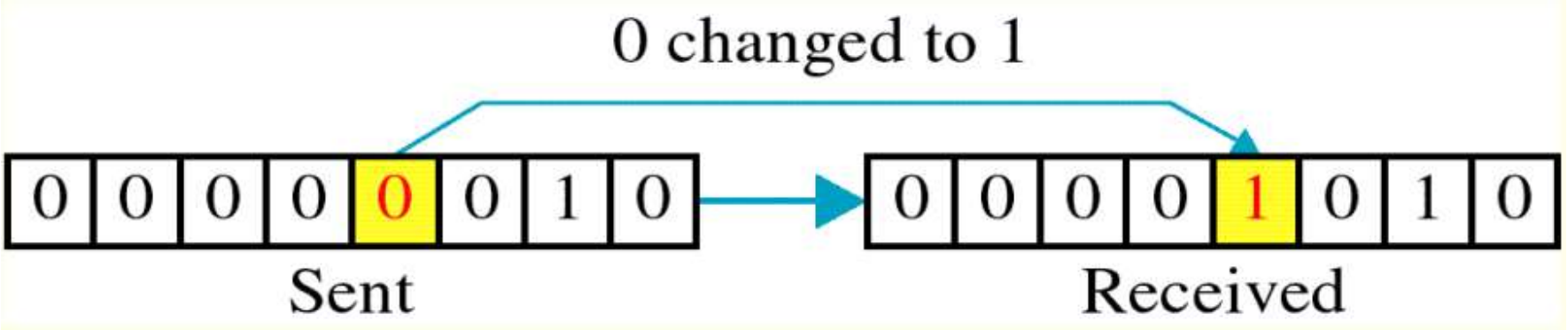
Tipos de errores

- Interferencias, calor, magnetismo, etc., influyen en una señal electromagnética, esos factores pueden alterar la forma o temporalidad de una señal. Si la señal transporta datos digitales, los cambios pueden modificar el significado de los datos. Los errores posibles son:

Error de bit

- Únicamente un bit de una unidad de datos determinada cambia de 1 a 0 o viceversa.

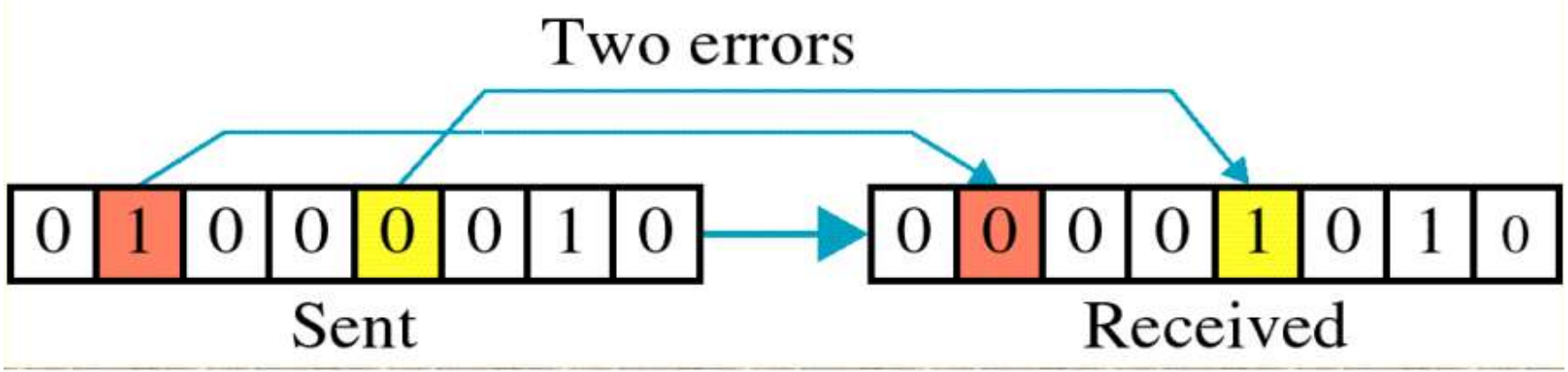
- Un error de bit altera el significado del dato.



Error doble de bit

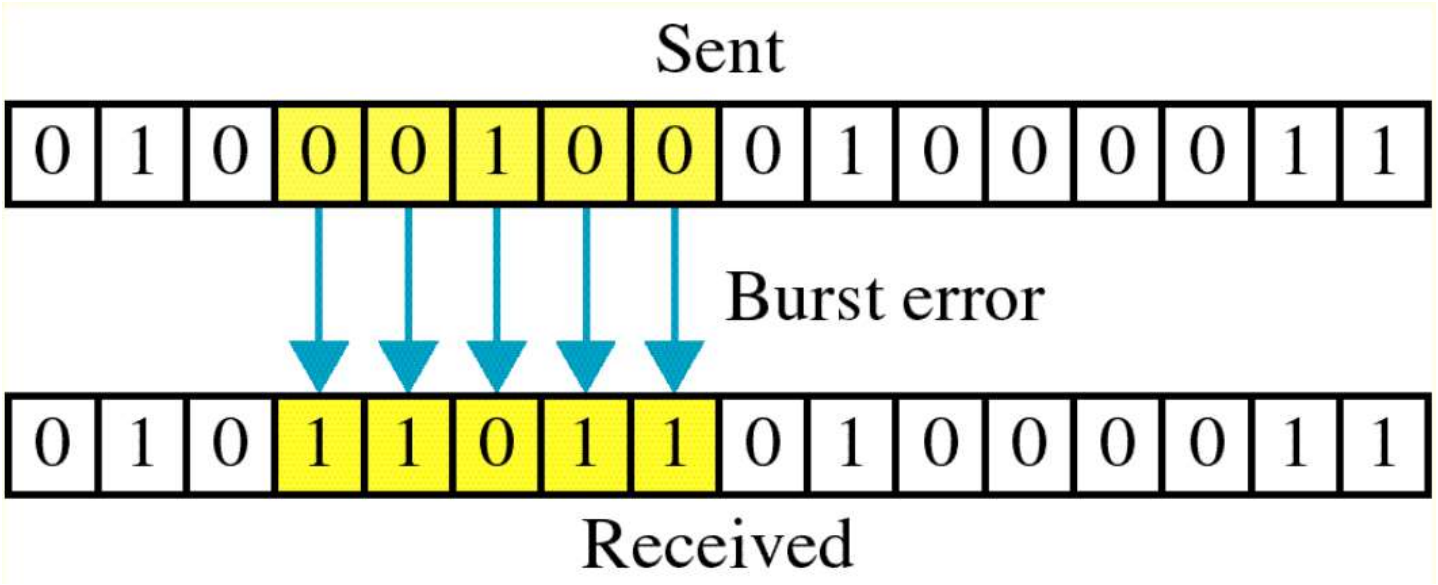
- Cuando dos bits de una unidad de datos determinada cambia de 1 a 0 o viceversa.

- Más de un error de bit altera el significado del dato.



Error de Ráfaga

El error de ráfaga significa que dos o más bits de la unidad de datos han cambiado. Los errores de ráfaga no significa necesariamente que los errores se produzcan en bits consecutivos.



Detección de errores

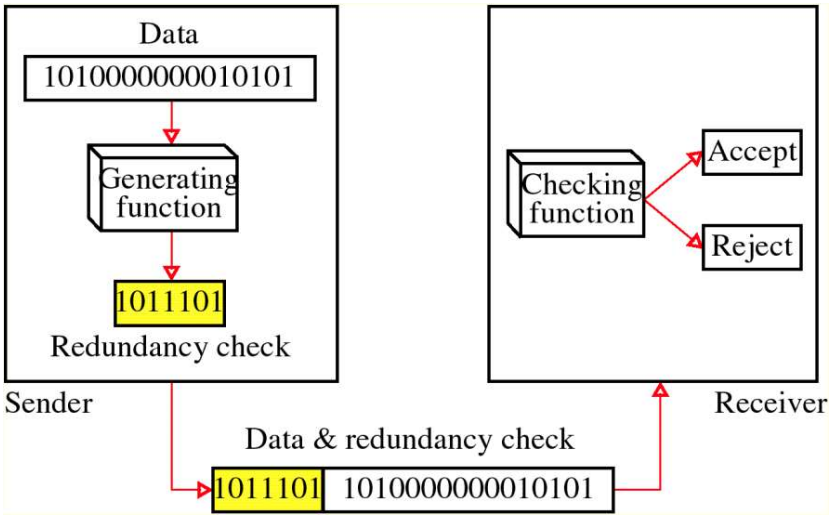
- Se conocen el tipo de errores que pueden existir, el problema es ser capaz de reconocerlos, dado que no se puede comparar el dato recibido con el original, sólo se podría saber que ha habido un error cuando se descodifique todo el mensaje y se vea que no tiene sentido.

- Sin embargo existen determinadas técnicas sencillas y objetivas para detectar los errores producidos en la transmisión:

Redundancia

- La redundancia consiste en enviar dos veces cada unidad de datos, de forma que el dispositivo receptor puede hacer una comparación bit a bit entre ambos datos y detectar si ha habido errores, para corregirlos con el mecanismo apropiado. Esta técnica es muy exacta pero enlentece la transmisión.

- Sin embargo el concepto es aplicable añadiendo al flujo de datos un grupo pequeño de bits al final de cada unidad, siendo estos bits redundantes con una parte de la información, esos bits redundantes se descartan una vez comprobada la integridad de la transmisión.



Redundancia

En las comunicaciones de datos se usan estos tipos de comprobación de redundancia:

- Verificación de Redundancia Vertical: (VRC, Vertical Redundancy Check) conocida como verificación de paridad.
- Verificación de Redundancia Longitudinal: (LRC longitudinal Redundancy Check).
- Verificación de Redundancia Cíclica (CRC Cyclic Redundancy Check) y
- Calculo de suma de comprobación (Checksum)

Bit de paridad

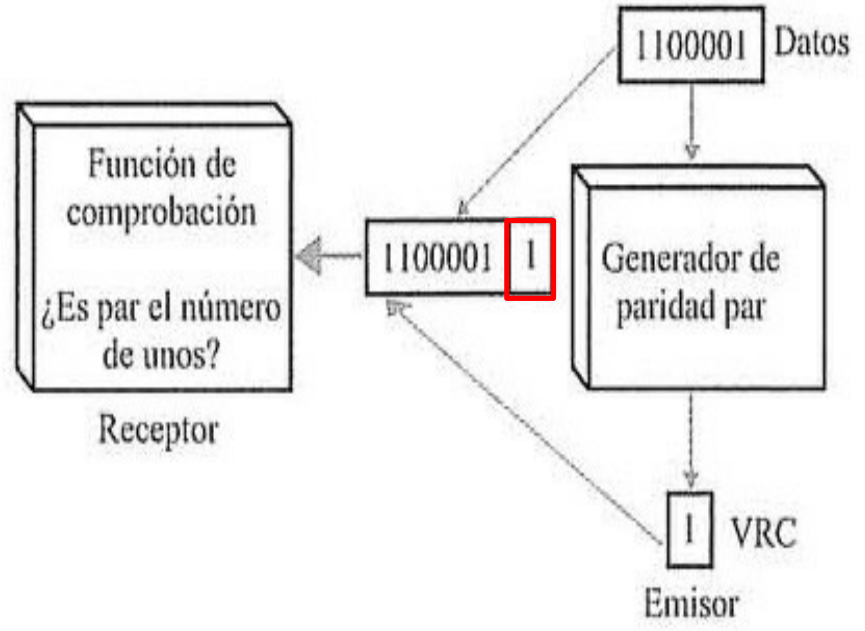
- Una de las formas más elementales de detección de errores consiste en incorporar un bit de paridad al dato transmitido.
    - El bit de paridad es un bit que toma el valor 0 ó 1 con el objeto de satisfacer una cierta restricción sobre la paridad de un determinado patrón de bits.
- Es posible implementar un esquema de paridad par o impar, a saber:
    - Un patrón de bits se dice tener paridad par o impar si, y sólo si, tiene un peso par o impar respectivamente.

Bit de paridad

- A partir de la paridad par o impar es posible definir sendos códigos de detección de errores:
  - Código de paridad par: a partir de un cierto dato se incorpora un bit adicional el cual adoptará el valor necesario para que el patrón dato + bit de paridad satisfaga una paridad par.
  - Código de paridad impar: definido de manera análoga, con la salvedad de que el patrón dato + bit de paridad ahora debe satisfacer una paridad impar.
- Supongamos que nuestro dato original es p = 0111001.
  - Paridad par 0111001 0
  - Paridad Impar 0111001 1

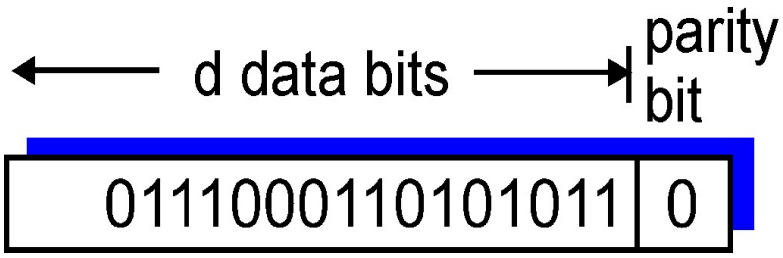
Verificación de redundancia vertical (VRC)

- Es el mecanismo más frecuente y barato, la VRC se denomina a menudo verificación de paridad, o también conocido como código TRC (Transverse Redundancy Check) y se basa en añadir un bit de redundancia, denominado bit de paridad, al final de cada unidad de datos, de forma que el número total de unos en la unidad (incluyendo el bit de paridad) sea par o impar.



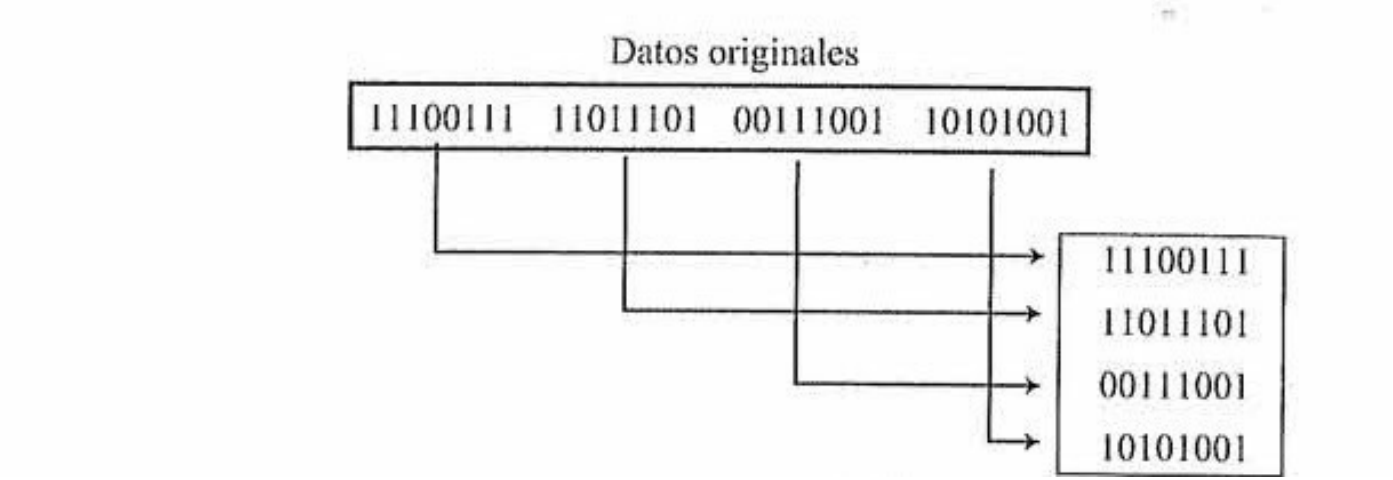
Verificación de redundancia vertical (VRC)

- Esta técnica permite reconocer un error de un único bit, y también de ráfaga siempre que el número total de bits cambiados sea impar. La función de paridad (par o impar) suma el dato y devuelve la cantidad de unos que tiene el dato, comparando la paridad real (par o impar) con la esperada (par o impar)



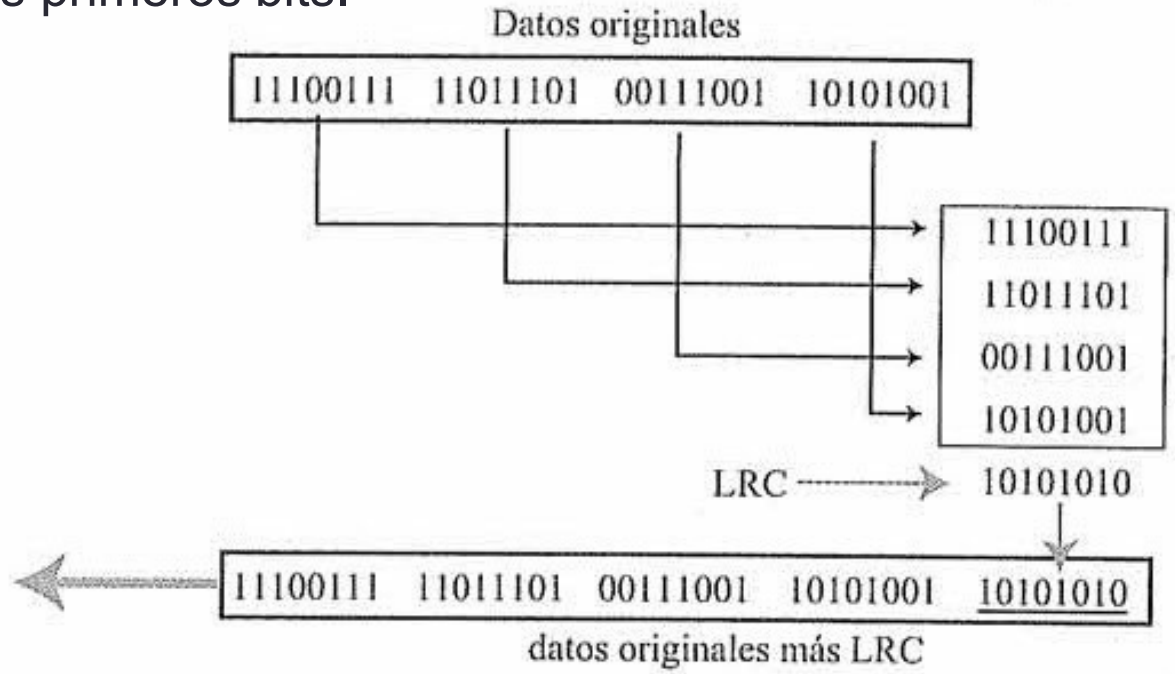
Verificación de redundancia longitudinal (LRC)

- En esta técnica, los bloques de bits se organizan en forma de tabla (filas y columnas), a continuación se calcula un bit de paridad para cada columna y se crea una nueva fila de bits, que serán los bits de paridad de todo el bloque, a continuación se añaden los bits de paridad al dato y se envían al receptor.



Verificación de redundancia longitudinal (LRC)

- Típicamente los datos se agrupa en unidades de múltiplos de 8 -1 byte- (8, 16, 24, 32 bits) la función coloca los octetos uno debajo de otro y calcula la paridad de los bits primeros, de los segundos, etc, generando otro octeto cuyo primer bit es el de paridad de todos los primeros bits.



Verificación de redundancia longitudinal (LRC)

- Esta técnica incrementa la probabilidad de detectar errores de ráfaga, ya que una LRC de n bits (n bits de paridad) puede detectar una ráfaga de más de n bits, sin embargo un patrón de ráfaga que dañe algunos bits de una unidad de datos y otros bits de otra unidad exactamente en la misma posición, el comprobador de LRC no detectará un error.

Ejemplo (LRC)

- Supongamos que se desea transmitir usando el código LRC, paridad par, al mensaje HOLA el cual está codificado en ASCII:

Ejemplo (LRC)

- Supongamos que se desea transmitir usando el código LRC, paridad par, al mensaje HOLA el cual está codificado en ASCII extendido:

H: 1 0 0 1 0 0 0
O: 1 0 0 1 1 1 1
L: 1 0 0 1 1 0 0
A: 1 0 0 0 0 0 1
0 0 0 1 0 1 0

Tx: 1 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0

Ejemplo (VRC y LRC)

- Supongamos que se desea transmitir usando el código VRC y LRC, paridad par, al mensaje HOLA el cual está codificado en ASCII:

Ejemplo (LRC)

- Supongamos que se desea transmitir usando el código VRC y LRC, paridad par, al mensaje HOLA el cual está codificado en ASCII:

H: 1 0 0 1 0 0 0 0
O: 1 0 0 1 1 1 1 1
L: 1 0 0 1 1 0 0 1
A: 1 0 0 0 0 0 1 0
0 0 0 1 0 1 0 0

Tx: 1 0 0 1 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0

Verificación de redundancia cíclica (CRC)

A diferencia de las técnicas VRC y LRC, que se basan en la suma (para calcular la paridad), la técnica CRC se basa en la división binaria.

Mecanismo: agregar al patrón de bits que compone el mensaje a ser enviado un conjunto de bits adicionales, de manera que el patrón resultante (al ser considerado como un polinomio binario), resulte divisible de manera exacta por un cierto polinomio denominado polinomio generador.

Verificación de redundancia cíclica (CRC)

Para poder llevar adelante el cociente entre polinomios, se debe interpretar el mensaje original como si se tratara de un polinomio.

Por ejemplo:
Los datos 010011 denota al polinomio $x^4 + x + 1$,
Mientras que 110110 denota al polinomio $x^5 + x^4 + x^2 + x$.

Verificación de redundancia cíclica (CRC)

- Entonces, sea M(x) el polinomio binario que representa el mensaje original, y sea G(x) el polinomio generador de grado n.

- En este contexto, el mensaje a ser transmitido T(x) = M(x) + r(x), donde r(x) es el residuo de dividir M(x) + redundancia (# de 0s = n) por G(x).

Si T(x) resulta divisible de manera exacta por G(x) el mensaje no tiene errores.

Verificación de redundancia cíclica (CRC)

- Pasos para calcular los bits que deben ser agregados a un cierto mensaje M(x):
  - Primero se añaden n bits en 0 a la derecha de M(x) (esto es, se añaden tantos ceros como grado tenga el polinomio generador).
  - Luego se divide el polinomio obtenido por el polinomio generador. Esta división se realiza en módulo dos, que es igual que la división binaria, con dos excepciones:
    - no hay carries ni borrows.
  - Finalmente, para obtener T(x) se suma el resto r(x) al polinomio original M(x) (desplazado en n bits en 0)

Verificación de redundancia cíclica (CRC)

- Supongamos que el mensaje que se desea transmitir es M(x) = 11010110111 y que el polinomio generador que se está usando es G(x) = 10011.

Verificación de redundancia cíclica (CRC)

- Supongamos que el mensaje que se desea transmitir es M(x) = 11010110111 y que el polinomio generador que se está usando es G(x) = 10011.

M(x) = 11010110111**0000** | 10011

10011
010011
010011
0000010111
10011
0010000
10011
0001**1100** = r(x)

T(x) = 11010110111**1100** → M(x) + r(x)

Verificación de redundancia cíclica (CRC)

- El algoritmo CRC cumple dos roles:
  - Por un lado, permite determinar los bits que se deben agregar al mensaje original.
  - A su vez, también permiten verificar si el mensaje recibido contiene o no errores.
- Nótese que el receptor puede ir dividiendo el mensaje a medida que va recibiendo los bits.
  - Es decir, no hace falta recibirlo en su totalidad para recién ahí comenzar a verificar el CRC.
  - Este código es muy simple de implementar en HW.

Verificación de redundancia cíclica (CRC)

- Supongamos que el mensaje que se recibe es R(x) = 1101011011111100 y que el polinomio generador que se está usando es G(x) = 10011.

R(x) = 1101011011111100 | 10011

```
10011
010011
010011
 0000010111
      10011
      0010011
        10011
        0000000 = residuo → Sin errores
```

Verificación de redundancia cíclica (CRC)

- Supongamos que se altera un par de bits del mensaje transmitido T(x) = 1101011011111100 → R(x)=1001111011111100. En este contexto verifiquemos el nuevo residuo.

```
R(x) = 1001111011111100 | 10011
        10011
       0000011011
           10011
          010001
            10011
           00010110
              10011
               0001010 = residuo → Con errores
```