UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESIS

SIMULAR EN VHDL LA IMPLEMENTACIÓN DE ALGORITMOS FEC EN DISPOSITIVOS PROGRAMABLES RECONFIGURABLES

QUE PARA OBTENER EL TÍTULO DE
INGENIERO **ELÉCTRICO ELECTRÓNICO**

PRESENTA:

FERNANDO SERRALDE MEDINA

DIRECTOR DE TESIS
JUAN FERNANDO SOLÓRZANO PALOMARES

CIUDAD UNIVERSITARIA AGOSTO/2013.

"Cada instante es una oportunidad de rehacer nuestras vidas"

Agradecimientos

A mis padres que son la razón y motivos más importantes de mi vida, mis hermanos siempre han formado parte de mí.

Mi alma máter la UNAM, todos los grandes ingenieros y profesores que hicieron posible directa o indirectamente la realización de este trabajo.

Ing. Carolina Garrido, una gran profesora y amiga.

A mis amigos y todos aquellos que han sido parte de mi vida en una forma especial.

Índice general

1. Introducción

En todo sistema digital, ya sea de comunicaciones, almacenamiento de información, etc. la información se ve afectada por diversos fenómenos, generando una gran área de trabajo que busca obtener mejores resultados en la transmisión de la información, para lo cual se han desarrollado toda una teoría de códigos, los cuales en general, se han aplicado primordialmente con tres grandes objetivos:

1. Comprimir mensajes.
2. Detectar y corregir posibles errores
3. Garantizar privacidad

En otras palabras, hacer la transmisión rápida, fiable y segura. La teoría aborda por separado cada uno de estos objetivos, porque de hecho son independientes. Por tanto, tenemos tres tipos diferentes de códigos:

1. Códigos compresores
2. Códigos correctores de errores
3. Códigos criptográficos.

Definir y delimitar cada uno, nos permite trabajarlos en forma totalmente independiente. Para el caso de la fiabilidad en la transmisión de la información, se han desarrollado los códigos correctores de error bajo dos vertientes;

1. Solo detección de errores
2. Detección y corrección

La primera se realiza por requerimiento de repetición (Automatic Repeat reQuest, ARQ), mientras que la segunda se hace por corrección de los errores en el lado del receptor (Forward Error Correction FEC). La corrección de los errores en el lado del receptor (FEC) se compone de dos partes, la aplicación de un cierto algoritmo FEC en el emisor y, la detección y corrección de errores producidos por el canal, mediante la comprobación del algoritmo recibido en el lado del receptor.

Por otro lado, la creciente área de aplicación de los diseños en VHDL, comienza a abarcar otras áreas en general, por ejemplo, el área de corrección de errores y procesamiento digital de señales, así como criptografía, donde por razones de seguridad y desempeño, resulta preferible realizar su implementación a nivel hardware sobre dispositivos reconfigurables, implicando esto la comprensión de los correspondientes

algoritmos, desarrollo metodológico de los componentes y el establecimiento del modelo correspondiente, así como su respectiva validación mediante algún simulador HDL.

Los algoritmos de codificación pueden ser implementados en plataformas de software y hardware, sin embargo, los métodos implementados en hardware ofrecen soluciones veloces para aplicaciones donde el tráfico de datos es más intenso y requieren de procesamiento en tiempo real. Los circuitos VLSI (Very Large Scale Integration), y los dispositivos FPGAs (Field Programmable Gate Arrays) son dos alternativas para implementar estos algoritmos en hardware, teniendo cada uno sus propias características, los FPGAs ofrecen un balance adecuado entre el espacio requerido por los circuitos y la rapidez con que se pueden realizar las operaciones de codificación y decodificación, aunado a la ventaja de la relación costo/beneficio.

Es posible simular los algoritmos de detección y corrección de errores hacia adelante (FEC) en lenguajes de descripción por hardware (HDL), que permiten visualizar el funcionamiento de dichos algoritmos a fin de sintetizar circuitos a implementar, por lo que proponemos: *realizar en VHDL, que resulta ser una herramienta ampliamente usada a nivel industrial, la simulación de aquellos algoritmos FEC, cuyo desarrollo metodológico, o partes de él, sirven de fundamento para conformar otros algoritmos.* Presentando las ventajas del método modular, así como del diseño descendente Top-Down, no solo dentro de la estructura del diseño algorítmico, sino también en la contextualización y análisis del problema.

Mediante una metodología descendente, partiendo de la importancia de los sistemas digitales, áreas de aplicación de los códigos detectores de error, contextualización de aspectos referentes al proceso de codificación, hacia la codificación de canal y los fenómenos que la afectan. Consecuentemente trabajar en la teoría de codificación, que servirá de argumento en el proceso de diseño, y como método de comprobación de la simulación a modo que podamos simular y comprobar la aplicación de los códigos FEC, sin necesidad de implementar el proceso de decodificación.

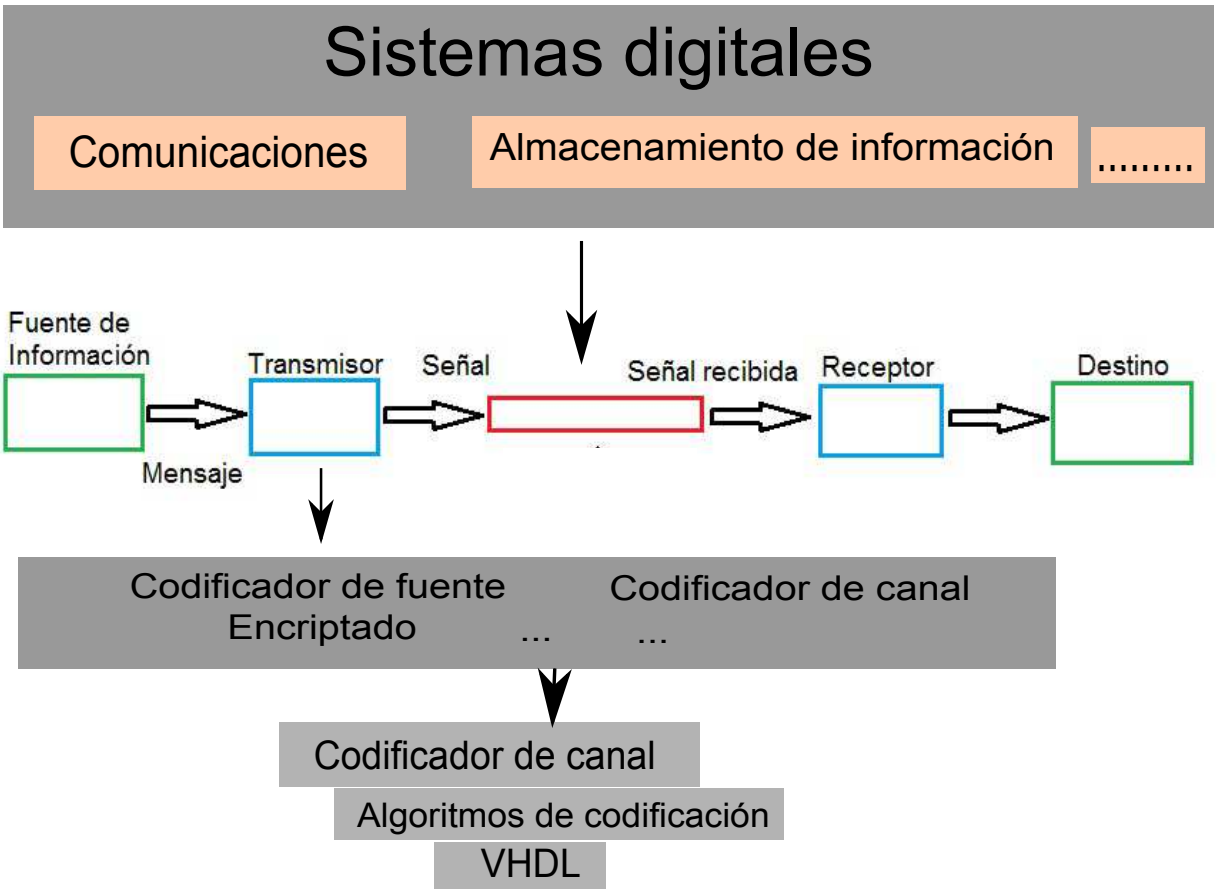


Figura 1.1.: Esquema Descendente/Modular

Por qué es importante la propuesta:

- Diseño multiplataforma.
  - Uso de un lenguaje HDL de uso libre e industrial, Verilog o VHDL
- No usará ningún IP-core.
- No usarán bibliotecas o ficheros de diseño predeterminadas.
- Implementación jerárquica y modular.
- Diseño con potencial de implementación en hardware.
- Reconocer y proponer el diseño de bloques o sistemas funcionales, como principio y fundamento del diseño de librerías y en su caso, de cores.
- Selección de estructuras y niveles en VHDL, adecuados para implementar algoritmos FEC.

2. Generalidades

Actualmente existe una fuerte tendencia hacia las comunicaciones digitales, esto debido a sus ventajas frente a las analógicas, claramente cada una con sus respectivas atribuciones. Las cuales han hecho imprescindible trabajar arduamente en éstas, de modo que él uso de herramientas programables reconfigurables (por ejemplo CPLD'S, FPGA's, etc.), resultan de excelente ayuda en el diseño de sistemas de comunicaciones.

Ventajas de las comunicaciones digitales:

- Inmunidad al ruido
- Resistentes a las distorsiones lineales y no lineales
- Detección y corrección de errores
- Compatible con multiplexaje
- Procesamiento digital de señales
- Flexibilidad de implementar hardware digital

Desventajas de las comunicaciones digitales:

- Sincronía
- El ancho de banda requerido; para una información requiere 10 o más veces que si fuera una señal analógica
- Introducen ruido regenerativo llamado "jitter" o fluctuaciones de fase, en los regeneradores

Las características de un dispositivo reprogramable reconfigurable (ejemplo un FPGA), son su flexibilidad, capacidad de procesado en paralelo y velocidad[1], lo cual les convierte en dispositivos idóneos para:

- Simulación y depuración en el diseño de microprocesadores.

- Simulación y depuración en el diseño de ASICs.

- Procesamiento de señal digital, por ejemplo vídeo.

- Sistemas aeronáuticos y militares.

[1]Prestaciones del dispositivo.

2.1. Antecedentes y contexto general

En sus publicaciones Claude E. Shannon, considerado como el padre de la teoría de la información, presentó un modelo para representar un sistema de comunicaciones, fundamentando en estas publicaciones importantes conceptos para la teoría de la comunicación como; ancho de banda, velocidad de transmisión, propiedades "ideales" de los sistemas, etc.

En su primer modelo de sistema de comunicaciones Shannon definió que este se componía esencialmente por cinco partes:

1. Fuente de información
2. Transmisor
3. Canal de comunicación
4. Receptor
5. Destinatario

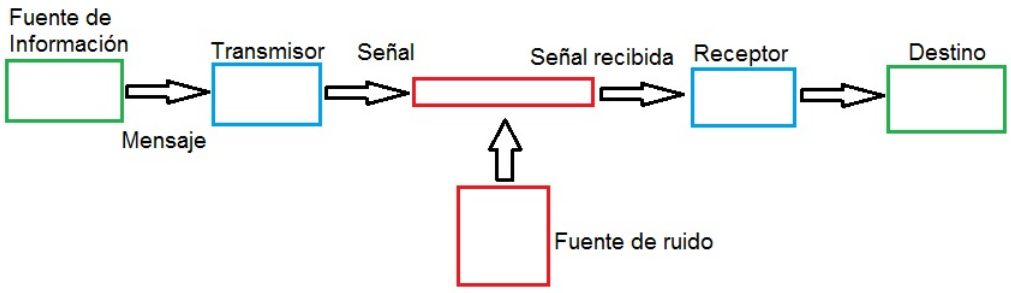


Figura 2.1.: Modelo de Shannon

Este modelo sin embargo, ha sufrido diversos cambios, y/o especificaciones debido a diversas circunstancias, como el agregado de compresión, *watermarker*, etc. Ahora bien, es importante reconocer que estos varían de acuerdo con el sistema de comunicaciones, sus medios y fines.

El esfuerzo principal de este trabajo se concentra en los sistemas digitales de comunicaciones, por lo que el enfoque de descripción, y modelado tendrá de igual manera esta prioridad. A continuación se ejemplifican diversas formas de modelar un sistema de comunicaciones.

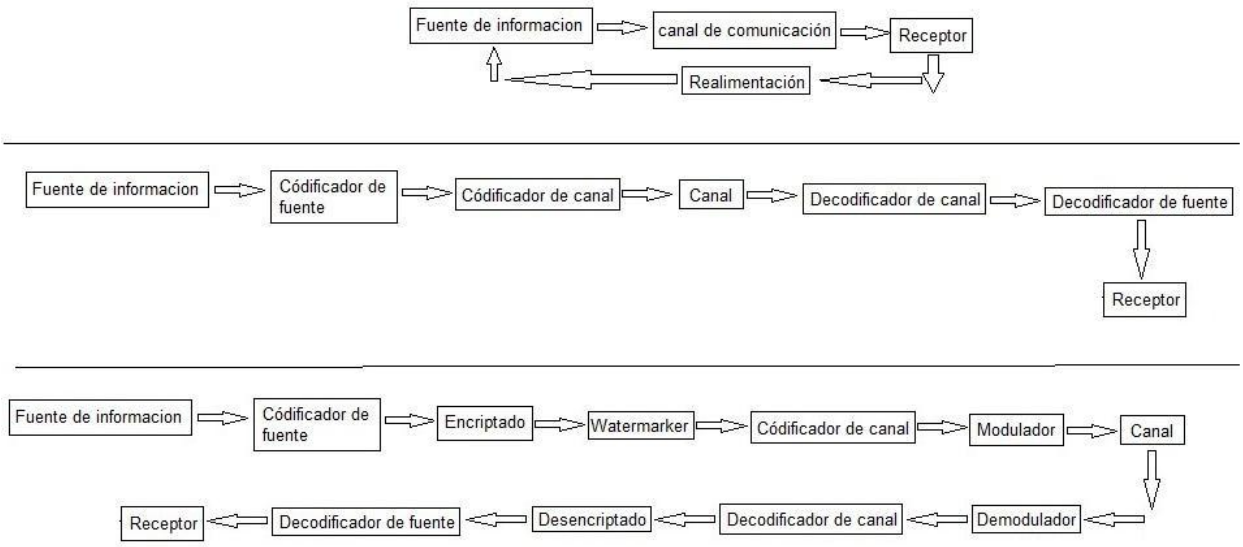


Figura 2.2.: Modelos de comunicaciones digitales

Con base en el esquema anterior se simulan los fundamentos FEC usados en un canal digital binario con auxilio de VHDL, para lo cual se estructura el procedimiento desde la conceptualización y profundidad a la teoría relativa, los algoritmos de codificación de canal, para finalmente a través de una metodología descendente y/o modular, obtener la simulación de los fundamentos correspondientes a los algoritmos de codificación de canal.

2.1.0.1. Términos y Definiciones

Sistema de comunicaciones. Es un sistema que tiene como primordial objetivo, el envió de información entre dos o más puntos en forma de señales analógicas o digitales.

Fuente de información. La fuente selecciona un mensaje a enviar dentro de varios posibles mensajes a transmitir a la terminal de recepción. el mensaje puede ser de diversas naturalezas, por ejemplo una secuencia de números o letras como en telegrafía, o una función de tiempo continuo f(t), como en radio o telefonía.

Transmisor. Este opera en el mensaje de diversas formas y produce una señal adecuada para la transmisión, a través del canal.

Canal. Es meramente el medio para transmitir la señal del punto de transmisión al de recepción, este puede ser un par de cables, cable coaxial, una banda o radio de frecuencias, etc. Durante la transmisión o en la terminal de recepción, la señal puede ser perturbada por ruido o distorsiones. El ruido y la distorsión pueden ser diferenciadas con base en que la distorsión es una operación fija

aplicada a la señal, mientras que el ruido implica perturbaciones impredecibles. La distorsión puede en un principio ser corregida aplicando la operación inversa, mientras que la perturbación debida al ruido no puede siempre eliminarse, ya que las señales no siempre presentan los mismos cambios durante la transmisión.

Receptor. Este opera al recibir la señal e intenta reproducir el mensaje original. Comúnmente realizará las operaciones matemáticas inversas del transmisor, pudiendo alterar algunas con el fin de mejorar el diseño para combatir el ruido.

Destinó. Es la persona u objeto a quien está intencionalmente dirigido el mensaje.

Codificador de fuente. Cambia la forma natural de la información, a una más comprensible para el sistema de comunicaciones.

Codificador de canal. Manipula los datos con la finalidad de darle mayor fortaleza a la información frente al ruido, etc.,(una forma común es introduciendo bits redundantes), y darle la posibilidad al receptor de detectar y hasta corregir errores que puedan existir en el proceso de la transmisión.

Entropía. Es la diferencia entre los elementos ordenados y desordenados de la información.

Capacidad. Es la habilidad del canal para transmitir información $\frac{Unidad.de.información}{segundo}$.

Encriptado. Previene que usuarios no autorizados capten o inyecten información al sistema.

Watermarker. Técnicas de manipulación que buscan proteger los derechos de autor y/o reproducción,

Teorema de codificación de canal. La tasa de errores de datos transmitidos en un canal confinado y con ruido puede reducirse a una cantidad arbitrariamente pequeña, si la velocidad de transmisión es menor que la capacidad del canal [Claude E. Shannon]

$C = W \cdot log[1 + \frac{S}{N}]$

Emisor y codificador de fuente El sistema de comunicaciones tiene como finalidad el trasladar información entre dos entes, de acuerdo con el modelo de canal de comunicaciones el origen de la información a transmitir es la fuente (emisor), la cual existe con diversas naturalezas, ya sea voz, vídeo, etc, y la información que arroja debe ser tratada para poder ser interpretada por el canal de comunicaciones, éste es el proceso del que se encarga el codificador de fuente.

2.2. Canal

Como ya se ha dicho el canal de comunicación es meramente el medio para transmitir la señal del punto de transmisión al de recepción, teniendo este, distintas naturalezas,

por tanto diferentes particularidades (efectos del canal) a determinar, durante la transmisión; la señal puede ser perturbada por ruido o distorsiones, las cuales afectan de formas diversas a la señal y varían de acuerdo con diversos factores, por ejemplo en redes inalámbricas, no se pueden modelar estas alteraciones de la señal, con solo agregar ruido gaussiano a la señal, sino que resulta fundamental agregar el desvanecimiento de la señal causado por las características de propagación, así como las pequeñas fluctuaciones causadas por la dispersión de la señal, (fenómeno conocido como propagación por trayectorias múltiples), entre otros fenómenos, para poder así realizar un efectivo modelado, para los fines de este trabajo, se menciona el canal perfecto y el binario simétrico, los cuales se podrían considerar como los límites extremos del modelado del canal, cuando para el binario simétrico la probabilidad de error es del 100 %.

Canal perfecto o sin ruido Un canal ideal es aquel que no presenta fenómeno alguno que afecte la señal transmitida (ruido, atenuación, etc.) por tanto la señal en el receptor es completamente igual a la señal enviada por el transmisor, de modo que no resulta necesaria la codificación de canal, este canal ideal solo existe como una referencia.

Canal Binario simetrico (BSC) Este canal es el más usual de los canales discretos sin memoria (DMC) y es definido por una probabilidad p, de que se produzca error para cada bit (0 ó 1) enviado.

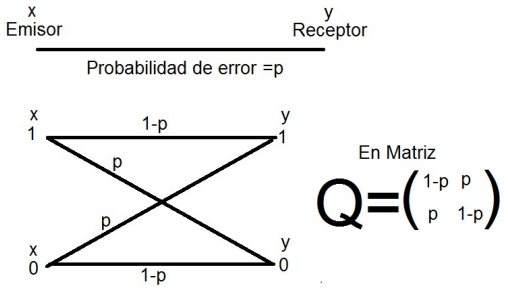


Figura 2.3.: Modelo de canal binario simétrico

Este canal es simétrico por ser iguales las probabilidades de recibir un 0 al enviar un 1, y viceversa. La probabilidad de que haya error es la suma de los términos de un renglón cualquiera es igual a la unidad, en la matriz del canal Q.

El valor de la probabilidad p definida para el BSC, equivale a la tasa de error en el bit (BER) del canal digital binario, pero para el cálculo de esta BER se emplean las probabilidades de aparición de cada uno de los símbolos de la constelación. A pesar de las discordancias entre los modelos BSC y un canal digital binario, se suele aceptar en la práctica la equivalencia entre ambos, asignando a p el valor de la BER, del canal digital.

2.2.1. Capacidad de canal

Para verificar si un sistema de comunicación es ideal o perfecto, pueden utilizarse muchos criterios. Para los sistemas digitales, el sistema óptimo es aquel que minimiza la probabilidad de error de bit a la salida del sistema sujeto a las restricciones de la energía transmitida y del ancho de banda del canal. Por lo tanto, el error de bit y el ancho de banda de la señal son de primordial importancia, por lo que Shannon planteó que es posible tener un sistema sin error de bit a la salida aun cuando se introduce ruido en el canal, pero solamente bajo ciertas circunstancias. Shannon demostró que (para el caso de una señal con ruido blanco gausiano añadido) se puede calcular una capacidad de canal C (bits/s) tal que si la velocidad de información R (bits/s) era menor a C, entonces la probabilidad de error se aproximará a cero, y donde la ecuación para C, está dada por la expresión;

$C = B \cdot log_2(1 + \frac{S}{N})$

Donde B es el ancho de banda del canal en hertz (Hz) y $\frac{S}{N}$ es la relación de potencia de señal a ruido a la entrada del receptor digital. Shannon no describe cómo construir tal sistema, pero sí demuestra que tener tal sistema es prácticamente posible. Por tanto, Shannon aporta un límite de rendimiento teórico que puede alcanzarse con sistemas de comunicación prácticos. Los sistemas que se aproximan a este límite a menudo incorporan codificación para la correción de errores.

El sistema óptimo en los sistemas analógicos es aquel que puede alcanzar la relación más grande de señal a ruido a la salida del receptor, sujeta a las restricciones de diseño tales como el ancho de banda del canal y la potencia transmitida.[Claude E. Shannon]

Otros límites fundamentales para la señalización digital fueron descubiertos por Nyquist en 1924 y Hartley en 1928. Nyquist demostró que si un pulso representa un bit de datos, se pueden enviar pulsos no interferentes a través de un canal a una velocidad no mayor a 2B pulsos, donde B es el ancho de banda del canal en hertz. Esto se conoce ahora como el teorema de dimensionalidad.[Leon W. Couch]

2.2.2. Codificación de canal

En el proceso de cualquier comunicación, resulta vital, poder obtener en el receptor la información del emisor lo más fielmente posible, sin embargo, durante el proceso de transmisión pueden producirse alteraciones en la información, debido a la presencia de ruido en el canal (esto debido a los mecanismos de propagación, la estabilidad del medio, interferencias, etc.) las no linealidades del transmisor, el ruido de cuantificación por el proceso de codificación, etc. Estas alteraciones causan que al reconstruir la señal en el receptor, ésta no corresponda fielmente a la enviada por el transmisor, estas alteraciones son llamadas errores. Buscando proteger la información de estos errores, se realiza el proceso de codificación de canal, el cual primordialmente protege la información ante las degradaciones del canal, así como detectar y corregir

errores producidos en el mismo, mediante *el agregado de redundancia y/o alguna secuencia estructurada*, principalmente. Resulta prudente enfatizar que la codificación de canal no tiene que ver con la codificación de fuente. El codificador de canal tiene como entrada una señal digital procedente del codificador de fuente. El codificador de canal "desconoce" si se trata de audio, datos, vídeo o de otro tipo, para él solo son una secuencia de bits cuya integridad se debe proteger de alguna forma para que puedan ser recuperados con la mayor fidelidad posible en el receptor.

Los sistemas de comunicaciones que trabajan con medios de transmisión muy estables como, fibra óptica y/o cable, tienen como principal problema la degradación de la señal, lo cual es muy predecible y por tanto relativamente fácil de compensar, mientras que los sistemas que trabajan con medios menos estables presentan problemas más complejos.

La codificación de canal puede tener dos vertientes; ya sea introduciendo redundancia controlada (secuencias estructuradas), o mediante la codificación de la forma de onda. La primera permite al receptor detectar y estimar la corrección de los errores en la información recibida, a diferencia de la codificación de fuente que tiene por función reducir al máximo el caudal binario preservando el contenido de información, el codificador de canal agrega información al código de fuente para posibilitar la detección y corrección de errores. En la segunda vertiente se tiene por finalidad convertir un conjunto de pulsos en otro conjunto mejorado, de modo que cada una de las formas de onda así codificadas sea lo menos parecida posible a cualquier otra del conjunto. Esta forma está asociada con la modulación digital.

El proceso de codificación en general suele componerse en dos partes principalmente, una codificación externa que se enfoca principalmente a los errores de ráfaga que afectan a varios bytes y, una codificación interna que se enfoca a la distorsión de los bits. La codificación llamada Exterior se emplea en todos los estándares DVB y se complementa con la llamada Interior en el caso de los estándares de transmisión vía satélite y terrestre.

Estratégicamente la corrección de errores a partir de lo antes mencionado tiene dos principales formas:

- ARQ (Automatic Repeat Request) solicitud automática de re-envío
- FEC (Forward Error Correction) corrige a partir de la información recibida.

Éstas se detallarán más adelante

El principio del problema de codificación es determinar el código óptimo[2] para que un canal dado aproxime la tasa binaria efectiva transmitida, al límite de la capacidad de Shannon, [3] lo cual está condicionado por: características del canal y recursos

[2]Decodificador óptimo: se conceptualiza como el mejor decodificador, aquel que elije como palabra código, aquella que al ser transmitida, difiere en menor número de bits en la secuencia recibida.
[3]artículo de Shannon "Communication in the presence of noise" reimpresión IEEE 1998

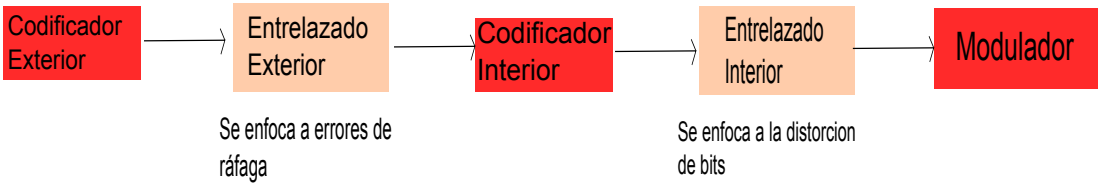


Figura 2.4.: Esquema de codificación

disponibles (límites en la velocidad de transmisión), para esto entonces definimos los siguientes conceptos:

k: longitud de los mensajes de entrada al codificador

n: longitud de las palabras de salida del codificador

Tasa de codificación (R): mide la proporción de cada palabra codificada que 'transporta' información.

$R = \frac{k}{n}$

Peso de palabra: número de veces en que un "1" aparece en la palabra ejemplo; sea la palabra (0110010), el peso de palabra es 3

Distancia entre dos palabras: número de dígitos en los que dos palabras no coinciden, ejemplo; sean dos palabras cualesquiera d(01100110,01101101) la distancia entre estas palabras código es entoces; d(01100110+01101101)=3.

Distancia mínima de código: Es la distancia Hamming entre las dos palabras código más cercanas: $d_{mín} = min[d_{mín}(c_1, c_2)]$ donde $c_1, c_2 \in C$.[Leon W. Couch][4]

Un tema digno de dedicarle unos segundos de análisis es la asociación de los códigos con la *tasa del código* $R = \frac{k}{n}$, *que mide el nivel de redundancia empleado en la codificación*, indicando el porcentaje de bits codificados que contienen información de mensaje. Este concepto tiene relación con el ancho de banda empleado en la transmisión cuando se utiliza codificación. Por ejemplo, al emplear una codificación en bloques de tasa $\frac{k}{n} = \frac{2}{3}$ entonces se debe tener en cuenta que en el mismo tiempo T en que originalmente y sin codificar se alojan las dos señales que representan a los dos bits, se tiene ahora que poder alojar a tres señales con la misma ocupación temporal, de manera que el tiempo de duración de cada señal pasa a ser de $T = 2$ a $T = 3$, y la ocupación espectral resulta por tanto mayor. Del mismo modo en un concepto equivalente en el caso del almacenamiento de datos digitales es que

[4]Cuanto más distintas sean las palabras del código, mejor es la capacidad de discriminación del código, es decir de detección y corrección de errores. Esta diferencia se mide como una distancia de Hamming, que es el número de bits distintos entre dos palabras del código V_i y V_j, si se define al "peso" de v_k, $w(v_k)$, como el número de 1's de v_k, entonces la distancia mínima $d_{mín}$de un código es el mínimo peso del código (excluyendo la palabra 0), y es una cota mínima para la capacidad de detección y corrección de errores. De igual modo, sea C un código lineal cuya matriz de comprobación de paridad es H . El peso mínimo (o la mínima distancia) de C es igual al menor número de columnas de H que suman 0.

la información codificada requeriría de mayor espacio físico para ser almacenada. Por estas razones será entonces conveniente mantener la tasa del código en niveles razonables, a pesar de que este objetivo va a estar en compromiso con la capacidad de corrección del código. Dado que los $2k$ mensajes son transformados en palabras de n bits, se puede interpretar este procedimiento como la aplicación de una expansión del espacio vectorial de dimensión $2k$ a otro de dimensión mayor $2n$, del cual se eligen de forma conveniente solo $2k$ vectores.

Por otro lado, de forma ideal lo que se busca es tener un decodificador que proporcione el mismo resultado del decodificador óptimo, pero con una carga computacional que crezca linealmente de acuerdo con el número de bits de información k. En los últimos años se han encontrado muchos esquemas de codificación con una carga computacional lineal pero no óptima. Una excepción son los códigos bloque perfectos y los códigos convolucionales, donde el algoritmo de decodificación proporciona una efectiva y óptima decodificación. Si cada k bits de información vienen codificados de formas independientes, se trata de códigos bloque, sin embargo esto no siempre es así, y por ejemplo en los códigos convolucionales la codificación se realiza sobre un flujo continuo de bits de mensaje. Un código se dice sistemático si los bits de mensaje son parte de la palabra código, separados de los bits de redundancia.

Para verificar si un sistema de comunicación es ideal o perfecto, pueden utilizarse muchos criterios. En los sistemas digitales, el sistema óptimo es aquel que minimiza la probabilidad de error de bit a la salida del sistema sujeto a las restricciones de la energía transmitida y del ancho de banda del canal. Por tanto se realizará una conceptualización general de estos.

2.3. Códigos de detección y corrección de errores *vs* códigos de línea

Como se ha mencionado con anterioridad, la codificación de canal puede tener dos vertientes; ya sea introduciendo redundancia controlada (secuencias estructuradas), o mediante la codificación de la forma de onda (codificación binaria de línea). Se detallará un poco más, la diferencia esencial entre ambas, con el fin de evitar confusiones, así como delimitar el campo a abordar, excluyendo del trabajo la codificación binaria de línea.

Mientras que un código de detección y corrección de errores trabaja esencialmente mediante el agregado de secuencias estructuradas, las cuales pueden componerse, por redundancia controlada, agregado de indicadores de paridad, etc., los Códigos de línea trabajan directamente sobre la forma de la onda a transmitir.

Podemos ejemplificar la codificación de detección y corrección de errores:

- Códigos simples
- Códigos convolucionales

- Códigos Hamming
- Códigos lineales de bloque
- Turbo códigos
- Etc.

Dado el hecho que la codificación de detección y corrección de errores, es el eje central del presente trabajo, la explicación detallada se realizará de manera más completa en un capítulo posterior.

La codificación de línea trabaja sobre la forma de la onda, bajo varios formatos de señalización de bit-serial, asignando niveles de voltaje, y/o fases de acuerdo con una regla particular.

Los unos y ceros binarios, como aquellos en la señalización PCM, pueden representarse en varios formatos de señalización serial de bit llamados códigos de línea. Existen dos principales categorías: con retorno a cero (RZ) y sin retorno a cero (NRZ). En la codificación RZ la forma de onda regresa a un nivel de cero volts para una porción, generalmente una mitad, del intervalo de bit. Las formas de onda para los códigos de línea pueden clasificarse aún más de acuerdo con la regla empleada para asignar niveles de voltaje para representar los datos binarios, algunos ejemplos son:

Señalización unipolar. En la señalización unipolar de lógica positiva el 1 binario está representado para un alto nivel (+ A volts) y el 0 binario por un nivel de cero. Este tipo de señalización también se conoce como modulación de encendido-apagado.

Señalización polar. Los unos y ceros binarios están representados por niveles positivos y negativos iguales.

Señalización bipolar (seudoternaria). Los unos binarios están representados por valores alternativamente positivos y negativos. El cero binario está representado por un nivel de cero. El término seudoternario se refiere al uso de tres niveles de señal codificados para representar datos de dos niveles (binarios). Esto se conoce como señalización de inversión alterna de marca (AMI, por sus siglas en inglés).

Señalización de Manchester. Cada uno binario está representado por un medio periodo de pulso de bit positivo seguido de uno negativo. De la misma manera, un 0 binario está representado por un medio periodo de pulso de bit negativo seguido de uno positivo. A este tipo de señalización también se le conoce como codificación por fase dividida.[Leon W. Couch]

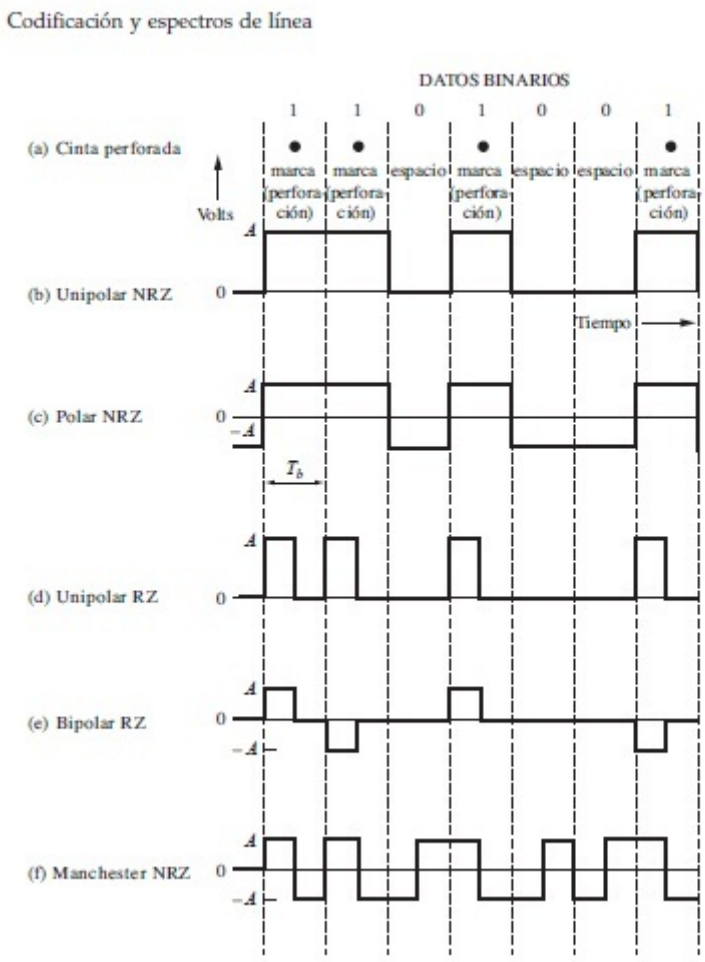


Figura 2.5.: Codificación Binaria de línea [Leon W. Couch]

2.4. Errores

2.4.1. Error de bit

Se denomina asi donde ocurre un error únicamente en un bit de una determinada cadena de bits, ya sea que este cambie de uno a cero o viceversa.

El cambio de un bit, es decir, la ocurrencia de un error altera el significado del dato. Son el tipo de error menos probable en una transmisión de datos serie, puesto que el intervalo de bit es muy breve $\frac{1}{f}$, por lo que el ruido o interferencia que lo causase tendría que tener una duración muy breve. Sin embargo, sí puede ocurrir en una transmisión paralela, en que un cable puede sufrir una perturbación y alterar a un bit de cada byte.

1	1	0	1	0	1	1
correcto		error	correcto			
1	1	1	1	0	1	1

2.4.2. Error de ráfaga

Éste es la ocurrencia de dos o más errores en una determinada cadena de bits, esto no significa que necesariamente los errores ocurran de forma consecutiva, algunos bits intermedios podrían ser correctos.[Leon W. Couch]

Este tipo de errores es común en sistemas seriales, dado que el ruido normalmente es mayor que la duración de un bit, por lo que afectará a un conjunto de bits. El número de bits afectados del código dependerá de la duración del ruido, estos errores se pueden producir por irregularidades físicas o estructurales en los medios de grabación.

Existen códigos especiales para corregir este tipo de errores, entre los principales los códigos cíclicos.

1	1	0	1	1	1	0	0	1	1	1	0
correctos		error	correcto	error					correctos		
1	1	1	1	0	0	0	0	0	1	1	0

2.5. BER

En los sistemas digitales, la medición del deterioro a menudo se toma como la probabilidad de error en el bit (P_e), también conocida como la tasa de error en el bit (BER).

Es un índice del número de bits o bloques recibidos incorrectamente, con referencia al total de bits o bloques de bits enviados durante un periodo de tiempo.

Al ser éste un índice, es posible interpretar;

$BER = 0$ la transmisión ha sido perfecta y sin tipo alguno de error.

$BER = 1$ se puede interpretar como una pérdida total de la información.

Este parámetro se encuentra en cierta forma relacionado con la relación señal a ruido.[5]

2.6. Ruido, origen y clasificación

Como parte de los factores que afectan la transmisión de la información en un sistema de comunicaciones, encontramos al ruido, por lo que resulta necesario, definirlo, así

[5]Para más detalles referentes a la BER, como demostraciones, etc, se recomienda visitar entre otros [Leon W. Couch] capítulo 7, donde se encontrará más información del tema.

como nombrar sus principales causas y efectos que se relacionan con el tema central del presente trabajo, sin embargo, no hay necesidad de profundizar en este tema, debido a que no constituye un eje central del trabajo.

Ruido Es el factor de mayor importancia en la limitación de las prestaciones de un sistema de comunicaciones, pues una señal se compone de: señal deseada, y señal no deseada (ruido), éste puede componerse de señales eléctricas indeseables externas o internas que degradan el rendimiento de un canal de comunicaciones.

Origen del ruido

- El movimiento de electrones u otros portadores de carga eléctrica en una corriente. Éste aumenta con la temperatura, la irradiación de los cuerpos negros [todos los objetos del universo], dependiendo de su temperatura, emiten energía en forma de ondas electromagnéticas
- Contactos defectuosos
- Artefactos eléctricos, alumbrado flourecente
- El ruido errático producido por fenómenos naturales como tormentas eléctricas, eclipses, manchas solares.

Clasificacion

- Ruido blanco o gaussiano
- Ruido de impulsos
- Ruido de intermodulación
- Ruido de amplitud
- Ruido rosa
- Ruido térmico

Ruido blanco o gaussiano Es una señal aleatoria con densidad espectral de potencia plana [su valor es constante en Hz en ancho de banda] es decir; tiene igual potencia sobre el espectro de frecuencias. No tiene correlación, es decir, su valor en dos momentos diferentes no están relacionados, es llamado ruido blanco porque es una mezcla de todos los colores.

Ruido de impulsos Es un ruido de ráfaga que dependiendo de su duración y la tasa de transferencia puede cambiar de un bit hasta decenas de bits, se causa por: falsos contactos, cambios de voltaje en líneas adyacentes, arcos eléctricos, interruptores o relevadores en telefonía antigua.

Ruido de intermodulación Se produce cuando las señales de 2 líneas se intermodulan y forman un producto que cae dentro de la banda de una tercera señal, pero fuera de ambas entradas.

Ruido de amplitud Comprende un cambio repentino en el nivel de potencia, causado por amplificadores defectuosos.

Ruido rosa Tiene espectro de frecuencias tal que su densidad espectral de potencia es proporcional al recíproco de su frecuencia, resulta en un predominio de frecuencias bajas.

Ruido térmico Agitación térmica de los electrones, no se puede eliminar e impone un límite en el desempeño de los sistemas de comunicaciones, se mide en un ancho de banda de 1[Hz], y se encuentra expresado; $N_0 = kT\ [\frac{W}{Hz}]$, donde;

$k = 1.3803x10^{-23}$ constante de Boltzmann

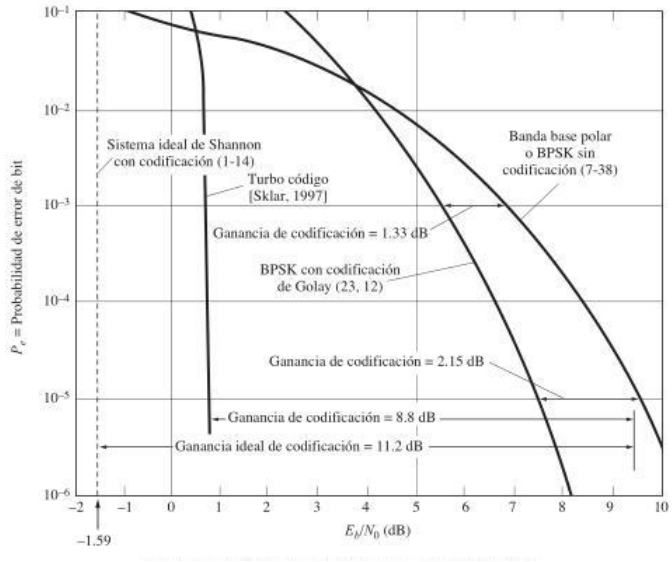
$T =$ Temperatura termodinámica expresada en grados Kelvin

$N_0 =$ Densidad de potencia de ruido

2.7. Rendimiento de código

En la figura 2.6, se ilustra la mejora en el rendimiento de un sistema de comunicación digital que puede alcanzarse utilizando la codificación. Se asume que una señal digital con ruido añadido en el canal está presente a la entrada del receptor. Se muestra también el rendimiento de un sistema que utiliza una señalización a través de la modulación por desplazamiento de fase binaria (BPSK), cuando se utiliza la codificación, así como cuando está no está existente.

La ganancia del código se define como la reducción en $\frac{E_b}{N_0}$ (en decibeles) que se alcanza cuando se utiliza codificación, en comparación con la $\frac{E_b}{N_0}$, requerida para el caso sin codificación a ningún nivel específico de P_e (probabilidad de error de bit). Por ejemplo, como se ve en la figura, se puede alcanzar una ganancia de codificación de 1.3 dB para una BER de 10^{-3}, pero la ganancia aumenta si BER es más pequeña, tal que se alcanza una ganancia de 2.15 dB cuando $P_e = 10^{-5}$. Ésta mejora es significativa en las aplicaciones de comunicaciones espaciales, donde cada decibel de mejora es invaluable. También podemos ver el umbral de codificación en el sentido de que el sistema codificado en realidad suministra un rendimiento más pobre que el de un sistema sin codificar cuando $\frac{E_b}{N_0}$ es menor que el valor de umbral, recordando que existe un umbral de codificación en todos los sistemas codificados.



Para el caso sin codificación, se emplea el circuito de detección óptimo en el receptor, para el caso con codificación se usa un código Golay.

Figura 2.6.: Rendimiento de sistemas digitales con o sin codificación.[Leon W. Couch]

El teorema de Shannon provee la $\frac{E_b}{N_0}$ requerida para una codificación óptima. Esto es, si la velocidad de la fuente está por debajo de la capacidad del canal, la codificación óptima permitirá la decodificación de la información fuente en el receptor con una $P_e = \longrightarrow 0$, aun cuando exista algún ruido en el canal.

3. Teoría de codificación y algoritmos

La diversidad de formas de codificación se basa en la necesidad de afrontar las situaciones diversas, como las mencionadas anteriormente, la capacidad del canal, el tipo de datos a transmitir, el ruido y efectos indeseables del canal, etc.

Las familias de códigos son clasificadas de acuerdo con la forma en que interactúan con los datos a transmitir, pero primero se dividen en métodos de detección y corrección, ARQ y FEC.

3.1. Detección y corrección de errores

Los métodos de detección de error suelen ser más simples que los de corrección de estos, sin embargo la realización y aplicación de estos métodos siempre depende de las características del sistema en donde se les deseé aplicar. Cuando el sistema tiene la posibilidad de la comunicación dúplex, es decir cuando la transmisión puede realizarse en ambos sentidos (por ejemplo la línea telefónica), los códigos pueden ser diseñados solo para la detección de errores pues la corrección de los mismos se realiza por requerimiento de repetición (Automatic Repeat reQuest ARQ). En todo sistema ARQ existe la posibilidad de la retransmisión, ahora bien existen sistemas con capacidad de transmitir en un solo sentido en los cuales no existe posibilidad de retransmisión, para estos sistemas es necesario utilizar una codificación capaz de permitir la corrección de los errores en el lado del receptor (Forward Error Correction FEC).

En la primera, el receptor al detectar problemas con la información recibida y "solicitar" un re-envío, de la información respectiva; mientras que en la segunda, a partir de la información recibida y la estructura de codificación "intenta" recuperar la información originalmente enviada.

La utilización de ARQ o FEC depende principalmente de la aplicación en cuestión, donde por ejemplo ARQ es comúnmente usado en sistemas de comunicación por computadora, debido a que su implementación es relativamente barata y generalmente existe un canal dúplex (bidireccional), que permite al receptor transmitir un acuse de recibido ACK para datos correctamente recibidos o NAC datos incorrectamente recibidos.

Por tanto en presencia de un canal símplex unidireccional, donde el envío del indicador ACK o NAC resulta poco factible y los retrasos de transmisión resultan largos, son preferidas las técnicas FEC, pues si se usara la técnica de ARQ, la velocidad efectiva de datos resultaría pequeña, causando por tanto largos periodos de inactividad del transmisor mientras espera el indicador ACK/NAC.

El trabajo presente se enfocará principalmente en los FEC, con finalidad primordial de desarrollo de un método de FEC para aplicación en VHDL.

Si bien una predicción originada por el segundo teorema de Shannon es una sofisticada técnica de codificación que puede llevar la transmisión sobre un canal ruidoso a operar como si se tratara de una transmisión sobre un canal libre de ruido. El teorema de Shannon muestra que la transmisión puede ser libre de errores utilizando una técnica de codificación de naturaleza aleatoria. En este proceso, las palabras de mensaje constituidas típicamente por bloques de bits, son asignadas de forma aleatoria, y que permiten decodificar unívocamente cada mensaje, bajo ciertas condiciones. Esta codificación propuesta por Shannon es esencialmente una codificación en bloques. Sin embargo, lo que no queda totalmente definido por el teorema es algún método constructivo para diseñar la sofisticada técnica de codificación. Para esto existen dos técnicas de codificación que difieren en la mecánica de generación de redundancia, estas dos técnicas básicas son la codificación en bloques y la convolucional.

Las etapas en el codificador se disponen en cascada y cada etapa es independiente de la anterior. El orden de las etapas es predispuesta de acuerdo con cada estándar, junto con su correspondiente configuración.

3.1.1. ARQ *vs* FEC

Resumiendo

Detección de error y retransmisión (ARQ):

- Utiliza los bits de paridad (bits redundantes añadidos a los de datos) para detectar la presencia de errores.
- El terminal receptor no intenta corregir los errores y simplemente pide al transmisor una retransmisión de los datos.
- Se requiere por tanto un enlace doble (ida y vuelta) para establecer este diálogo entre transmisor y receptor.

Corrección adelantada de errores (FEC):

- Necesita sólo un enlace de ida
- Los bits de paridad se diseñan tanto para detectar la presencia de errores como para corregirlos.
- No es posible corregir todos los posibles errores.

Diferencias y ventajas, ARQ y FEC ARQ funciona mejor que los FEC simples en presencia de errores por ráfaga.

FEC funciona mejor que ARQ en presencia de errores independientes.

Cuando es posible preveer la naturaleza de los errores, es posible diseñarse un sistema FEC ad-hoc más eficiente que ARQ.

-aquí la importancia de presentar los tipos de error y los efectos del canal sobre el sistema.

FEC es muy sensible a la degradación del canal (interferencia, ruido impulsivo, atenuación, etc.)

ARQ requiere grandes buffers de memoria e introduce retardo, mientras que en los sistemas FEC el rendimiento se mantiene constante.

Por tanto podemos decir que ARQ tiene ventaja sobre FEC en el sentido de que éste es mucho más simple y requiere el uso de menos redundancia en los códigos, sin embargo debe usarse FEC cuando la conexión es simplex, cuando los retardos con ARQ son excesivos o el número esperado de errores sin corrección implique un gran número de retransmisiones.

Para finalizar, se muestra en la figura 3.1. la función de los sistemas ARQ, que esencialmente pueden ser interpretados y asociados:

- ARQ de parada y espera (stop-and-wait - half duplex)
  - El transmisor envía un bloque de información y espera del receptor una señal de reconocimiento (ACK), que le indica que la información se ha recibido sin errores, antes de que proceda al envío de más información.
  - Si se detectan errores, el receptor envía una señal de no reconocimiento (NAK) y el transmisor retransmite la información.
  - Sólo requiere una conexión semiduplex.
- ARQ continua con vuelta atrás (pullback - full duplex)
  - El transmisor envía continuamente paquetes de información, con un número de identificación, y el receptor envía los correspondientes ACK o NAK con la identificación del paquete. – Cuando se recibe un NAK el transmisor vuelve a retransmitir desde el bloque erróneo en adelante.
  - Se requiere una conexión dúplex.
- ARQ con repetición selectiva (selective repeat - full duplex)
  - Parecido al anterior con la diferencia de que sólo se retransmiten los mensajes recibidos con error.

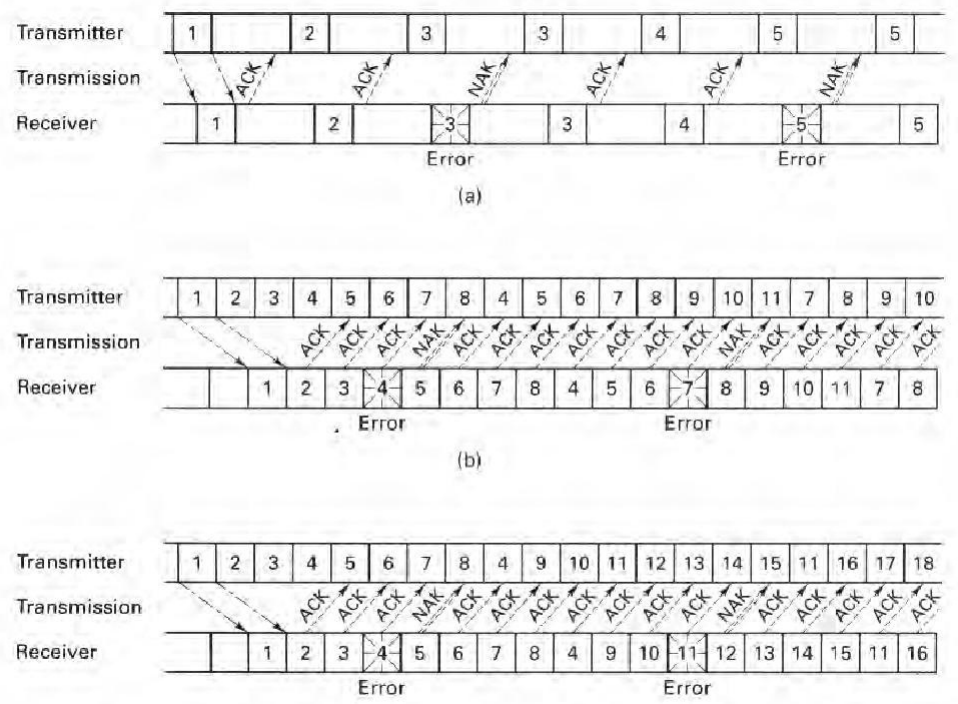


Figura 3.1.: Automatic repeat request (ARQ) (a) Stop and wait ARQ (Half duplex) (b) Continuos ARQ with pullback (full duplex) (c) Continous ARQ with selective repeat (full duplex) [Sklar]

FEC Existe una enorme variedad de los algoritmos FEC, sin embargo y bajo el objetivo de encontrar los fundamentos de la codificación FEC resulta necesario *remitirse a la codificación digital binaria*, excluyendo con esto los códigos que no cumplan con alguna de estas condiciones, (aun aquellos que trabajan los bits como símbolos y no como bits individuales) como los Reed Solomon, Reed Moler entre otros, de modo que, se hallaran los fundamentos de la codificación digital binaria.

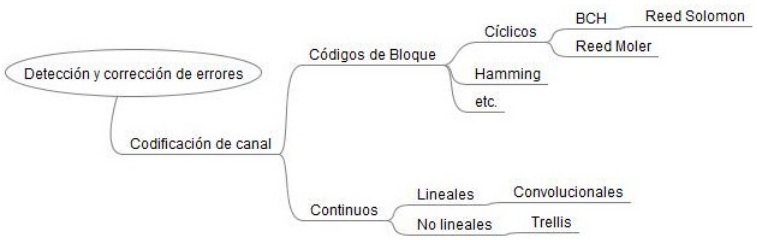


Figura 3.2.: Algoritmos FEC

3.1.2. Nomenclatura de codificación

Distintos autores describen los elementos de la salida del codificador en una gran diversidad de formas: “bits de código”, “bits de canal”, “símbolos de código” y “símbolo de canal”, teniendo todos exactamente el mismo significado. Los términos “bits de código” y “bits de canal” resultan más descriptivos para el caso de códigos binarios, mientras que “símbolos de código” y “símbolos de canal” son frecuentemente preferidos porque pueden ser usados para describir igualmente códigos binarios como no - binarios. Es necesario no confundir “símbolos de código” con el grupo de bits que conforman la transmisión de símbolos.

Los términos “bit de paridad” y “símbolo de paridad” es usado para identificar solamente esos elementos del código que representan los componentes redundantes agregados a los datos originales.

Todo codificador se caracteriza básicamente por el tipo y por la tasa de codificación.

Códigos simples El código de repetición es una forma simple de realizar una codificación, en el cual simplemente se repite el símbolo transmitido una determinada cantidad de veces. En la transmisión de símbolos binarios el bit ’1’ se representa por una secuencia de ’1’ s , mientras el símbolo ’0’ se representa por una secuencia de ’0’s. Para el caso de un código de repetición por ejemplo, las palabras pertenecientes al código son (111) y (000). La primera típicamente representa al uno, mientras que la segunda representa al cero. Existirá detección de error cuando se reciba cualquier otra palabra de las ocho posibles que se tienen que no sean las del código. Así se sabrá que hay errores si se recibe por ejemplo la palaba (110). El proceso de codificar es básicamente ampliar la dimensión del sistema en que se trabaja para elegir en un espacio de vectores mayor, ciertas palabras validas, mientras otras no pertenecen al código. Los otros seis patrones posibles de ser recibidos corresponden a patrones de error. Se dice entonces que el sistema es capaz de detectar uno o dos errores. Así, si consideramos que la posibilidad de un error es mayor que la de dos errores, los patrones (110), (101) y (011) se considerarán secuencias de tres “1” con un error, de igual manera los patrones de (001), (010) y (100) se considerarán secuencias de tres ceros con un error. La corrección establecerá que el vector transmitido fue el de tres ’0’s por lo que el bit transmitido es ’0’, por tanto este sistema solo es capaz de corregir un error.

Ahora recordando que la eficiencia del código se mide por medio de la tasa de transmición (recordando; $R = \frac{k}{n}$, donde k son los bits de información[1] y n la palabra entera que se transmite[2]), entonces los códigos de repetición tienen una baja eficiencia en cuanto a velocidad, a pesar de su buena capacidad de corrección de errores.[Arnone Ing.]

Todo codificador se caracteriza básicamente por el tipo y por la tasa de codificación.

[1] bits de entrada

[2] bits de salida.

Código sistemático El término sistemático se refiere a aquellos códigos en los que una de sus ramas generadoras de los símbolos de código emite justamente los bits de datos originales. Es decir que en la salida aparece explícitamente la entrada. Así que cada uno de los puertos de entrada al codificador debe estar conectado directamente con un puerto de salida. Un codificador es sistemático si se puede identificar la matriz identidad entre los elementos de G(x). Por tanto si no se cumplen las propiedades anteriores, el codificador es no sistemático. Si k es igual a uno, entonces el codificador es no sistemático cuando hay al menos un puerto de entrada que no está conectado directamente con una salida.

Código no recursivo o recursivo En un codificador no recursivo, no hay ninguna realimentación en ninguna etapa de los registros de desplazamiento, mientras que en un recursivo hay realimentación, los bits de salida se realimentan a la entrada, o alguna etapa de algún registro de desplazamiento se realimenta a una etapa anterior del registro de desplazamiento. [3][Viñé Viñuelas]

3.1.3. Códigos lineales

Un código lineal es aquel en que las palabras código se calculan mediante transformaciones lineales. Todo código lineal debe contener la palabra "cero" (todos los digitos de la palabra son cero), Esto es:

- Si C_i,C_j son palabras código, entonces: $C_i + C_j$ tambien debe ser una palabra código
- La palabra código "cero" C_1ésta en el código

éste tipo de código es uno de los más usados debido a las facilidades que nos presenta para resolver los problemas de codificación.

3.1.4. Códigos Hamming

Una clase de códigos de bloques ampliamente utilizada es la de los códigos de Hamming.

Sea para cualquier entero positivo $d \geq 3$ existe un código de Hamming con ciertas caracteristicas particulares, las cuales son:

- longitud $n = 2^m - 1$ En este codificador la información presentada es segmentada en bloques de K bits, que son los denominados bits de mensaje, que en conjunto constituyen 2^K posibiles mensajes. El códificador transforma cada

[3]La forma de realimentar los bits de salida a alguna etapa del registro de desplazamiento, es también usado en el diseño de los bloques de código pseudo aleatorio.

bloque de datos en un bloque más largo, de $n > K$ bits, agregando los denominados bits bits de redundancia o de control de paridad, para formar una palabra código:

- Número de bits de mensaje codificado $n = 2^{k_c} - k_c - 1$
- Número de bits de mensaje $K = n - k_c$
- Número de bits de control de paridad $k_c = n - K$
- Capacidad de correccion de errores $t = 1(d_{min} = 3)$

La matriz de paridad de estos códigos H se forma con las columnas de k_c bits no nulas que pude ser implementada de forma:

$$H = [I_{k_{cc}} Q]$$

donde la submatriz identidad es de mxn y la submatriz Q consiste de $2^m - m - 1$ columnas formadas con vectores de peso 2 o mayor. La matriz generadora se obtiene de acuerdo a la expresión, para la forma sistemática del código: $G = [Q^T I_{2^{k_c}-1}]$

En la matriz H la suma de tres columnas dan como resultado el vector nulo, con lo cual la distancia mínima de estos códigos es $d_{min} = 3$ de forma que pueden ser utilizados para corregir patrones de error de un bit o detectar cualquier patrón de error de dos bits[Arnone Ing.].

Criterio de Hamming Sea una secuencia de información de K bits. Una decodificación eficiente resulta si, al añadir K_c bits de codificación, se cumple que:

$2^k > K + K_c + 1$

Donde; K_c es el valor de Hamming.[García Álvarez]

Ejemplificando el algoritmo simple de Hamming; Usando el algoritmo simple de Hamming, para codificar con capacidad de corrección de un error simple

Sea la palabra a codificar:

$X_k = 1001101$

De acuerdo al criterio de Hammin $K_c = 4$[4], por tanto son necesarios 4 bits de redundancia, los cuales ocupan todas las posiciones de la nueva secuencia que son potencias de 2. Este código se denominara (N,K)=(11,7)

Posición K_c	11	10	9	8	7	6	5	4	3	2	1
X_{kc}	1	0	0	X	1	1	0	X	1	X	X

[4]De acuerdo al criterio de Hamming para este ejemplo, si $K_c = 4$ & $K = 7 \rightarrow 2^4 > 7 + 4 + 1$ Por tanto $K_c = 4$

X es la posición de los bits redundantes dentro de la secuencia K_c, los cuales resultan de la suma en modulo 2, de los números que corresponden a las posiciones que tienen bit=1.

11	1011
7	0111
6	0110
3	0011

Operando en modulo 2

$11 \oplus 7$	$1011 \oplus 0111 = 1100$
$(11 \oplus 7) \oplus 6$	$1100 \oplus 0110 = 1010$
$((11 \oplus 7) \oplus 6) \oplus 3$	$1010 \oplus 0011 = 1001$

Por tanto $1001 = K_c$

Colocando K_c en las posiciones potencia de 2, los cuales anteriormente marcamos con X

Posiciones K_c	11	10	9	8	7	6	5	4	3	2	1
X_{kc}	1	0	0	1	1	1	0	0	1	0	1

Obtenemos la palabra codificada, por tanto $X_k = 10011100101$

Bajo esta estructura es posible detectar un error simple, en el decodificador los números correspondientes a las posiciones con bit=1, incluyendo a los de paridad, se suman en modulo 2.

En caso de que la secuencia enviada fuese se recibiera sin error

Posición

11	1011
8	1000
7	0111
6	0110
3	0011
1	0001

Operando

$11 \oplus 8$	$1011 \oplus 1000 = 0011$
$(11 \oplus 8) \oplus 7$	$0011 \oplus 0111 = 0100$
$((11 \oplus 8) \oplus 7) \oplus 6$	$0100 \oplus 0110 = 0010$
$(((11 \oplus 8) \oplus 7) \oplus 6) \oplus 3$	$0010 \oplus 0011 = 0001$
$((((11 \oplus 8) \oplus 7) \oplus 6) \oplus 3) \oplus 1$	$0001 \oplus 0001 = 0000$

Como el resultado es una secuencia de ceros, podemos asegurar que no ha habido error.

Este tipo de codificación tiene capacidad de detectar un solo error simple y se realiza mediante detectar en el receptor una secuencia distinta de ceros, la cual nos indica

la posición del error, por ejemplo, si la suma binaria en el receptor igual a 1011, entonces el error se encuentra en la posición 11, de la palabra código.

Esto solo es válido para la distribución donde los bits de control son posiciones 2^{k_c}, sin embargo existen otras formas de distribuir las posiciones de control, o de paridad para esas otras formas el receptor debe tener en en su memoria las posiciones donde se insertan dichos bits. De éstas y otras variaciones de distintos parámetros es que nacen nuevas formas del código de Hamming.

3.1.4.1. Hamming (7,4)

Hoy en dia el código (7,4), hace referencia al introducido por Hamming en los 50's, el cual agrega 3 bits de comprobacion, por cada 4 de mensaje.

Éste algoritmo puede corregir cualquier error de un solo bit, pero cuando hay errores en más de un bit, la palabra transmitida se confunde con otra con error en un sólo bit, siendo corregida, pero de forma incorrecta, es decir que la palabra que se corrige es otra distinta a la original, y el mensaje final será incorrecto sin saberlo. Para poder detectar (aunque sin corregirlos) errores de dos bits, se debe añadir un bit más, y el código se llama Hamming extendido. El procedimiento para esto se explica al final.

1. Todos los bits cuya posición es potencia de dos se utilizan como bits de paridad (posiciones 1, 2, 4, 8, 16, 32, 64, etc.).
2. Los bits del resto de posiciones son utilizados como bits de datos (posiciones 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.).
3. Cada bit de paridad se obtiene calculando la paridad de alguno de los bits de datos.

La posición del bit de paridad determina la secuencia de los bits que alternativamente comprueba y salta, a partir de éste, de modo que tenemos:

Posición 1: salta 0, comprueba 1, salta 1, comprueba 1, etc.

Posición 2: salta 1, comprueba 2, salta 2, comprueba 2, etc.

Posición 4: salta 3, comprueba 4, salta 4, comprueba 4, etc.

Posición 8: salta 7, comprueba 8, salta 8, comprueba 8, etc.

Posición 16: salta 15, comprueba 16, salta 16, comprueba 16, etc.

Por tanto para la posición n es: salta $n-1$ bits, comprueba n bits, salta n bits, comprueba n bits.[Dapaena]

3.1.4.2. CODIGOS HAMMING EXTENDIDOS.

Estos códigos se obtienen añadiendo un símbolo adicional que computa todos los anteriores n símbolos de la palabra código. Tienen $d_{min} = 4$, por lo que detectan

todos los errores dobles y a la vez corrigen todos los individuales. La decodificación se realiza así:

Si el último dígito del síndrome es 2, entonces el número de errores debe ser impar. La corrección se realizaría de la manera habitual.

Si el último dígito del síndrome es 0, pero el síndrome no es todo ceros, no hay corrección posible, porque se ha producido más de un error, pero los errores dobles son detectados.

CÓDIGOS DUALES. Se dice que dos códigos son duales, cuando la matriz de comprobación de paridad H de uno, es la matriz generadora del otro.

CÓDIGOS MAXIMAL-LENGTH. Son los duales de los códigos HAMMING, por lo que la matriz de comprobación H de un código Hamming es la matriz generadora de uno maximal-length.

CÓDIGOS REED-MULLER Son una familia de códigos que cubre un amplio rango de tasas y distancias mínimas. Para cualquier valor de m, y fijando un $r < m$, hay un código Reed-Muller con $n = 2^m$.

3.1.5. Códigos cíclicos

Estos códigos (BCH, cyclic error-correcting codes) son una subclase de los códigos de bloque lineales, estos son fácilmente implementados mediante registros de desplazamiento realimentados. (figuras 3.3 y 3.4), De igual forma, el cálculo del síndrome resulta sencillo mediante registros de desplazamiento. [Lathi]

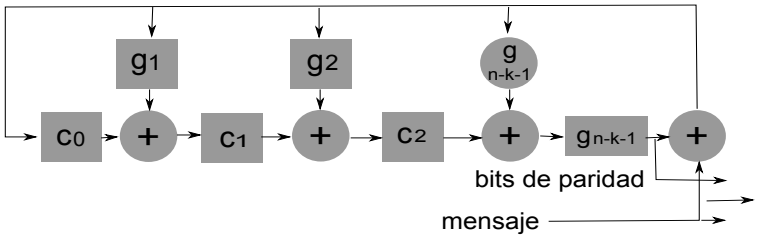


Figura 3.3.: Registro corrimiento en el codificador

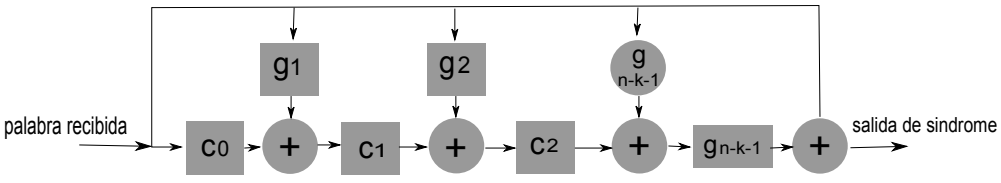


Figura 3.4.: Registro de corrimiento en el decodificador

Un código (n, k) se dice que es un código cíclico, si cumple la propiedad:

Si u-upla $u = (u_0, u_1, u_2, ..., u_{n-1})$ es un vector del código en el subespacio S, entonces $u(1) = (u_{n-1}, u_0, u_1, u_2, ..., u_{n-2})$ obteniendo mediante un desplazamiento; también es un vector del código en el subespacio S.

Los componentes del vector del código $u = (u_0, u_1, u_2, ..., u_{n-1})$ pueden ser tratados como los coeficientes de un polinomio $u(X) = u_0 + u_1X + u_2X^2 + ... + u_{n-1}X^{n-1}$ donde la presencia o ausencia de cada termino en el polinomio indica la presencia de un 1 ó 0 en la correspondiente localización de la n-upla. En este caso un desplazamiento i es indicado como $u^{(i)}(X)$ y es obtenido mediante la operación:

$u^{(i)}(X) = X^i u(X) mod(X^n + 1)$

mod es la operación módulo. En un código cíclico (n, k), cada polinomio de la palabra de código $u(X)$ en él subespacio S, y puede ser expresado como:

$u(X) = m(x) \cdot g(X)$

siendo $m(x)$ él polinomio del mensaje, él cual queda escrito en el registro de corrimiento en el decodificador como:

$m(X) = m_0 + m_1X + m_2X^2 + ... + m_{k-1}X^{k-1}$

y $g(X)$ es el polinomio generador, él cual tiene la forma: $g(X) = g_0 + g_1X + g_2X^2 + ... + g_{n-k}X^{n-k}$

donde g_0 y g_{n-k} deben ser igual a 1.

A continuación se describe una forma sistemática para obtener a partir del vector de mensaje $m(X) = m_0 + m_1 + m_2 + ... + m_{k-1}$ el vector del código $u = (r_0, r_1, ..., r_{n-k-1}, m_0, m_1, ..., m_{k-1})$ donde al principio del vector se encuentran los $(n - k)$ bits de paridad y al final los k bits del mensaje. Al vector u le corresponde el polinomio de código $u(X)$ que entonces puede ser escrito como:

$u(X) = r_0 + r_1X + ... + r_{n-k}X^{n-k-1} + m_0X^{n-k} + m_1X^{m-k+1} + ... + m_{k-1}X^{n-1}$

ésta puede ser representada;

$u(X) = r(X) + X^{n-k}m(X)$

De las ecuaciones anteriores podemos lograr una forma sistemática a partir de códigos cíclicos colocando los dígitos del mensaje en las k etapas más hacia la derecha del registro de palabra de código y ubicando los dígitos de paridad en las $(n - k)$ etapas más hacia la izquierda, donde se tiene el polinomio generador $r(X)$ definido por la operación:

$r(X) = X^{n-k}m(X) mod(g(X))$.

El cálculo de los bits de paridad es el resultado de obtener $X^{n-k}m(X) mod(g(X))$, en otras palabras, es la división del polinomio de mensaje desplazado hacia la derecha por el polinomio generador $g(X)$. La codificación comienza inicializando con ceros los registros r_0a r_{n-k-1}, cerrando el sistema de realimentación y colocando la llave de

salida en la posición que permite que los bits del mensaje $m(X)$pasen a la salida. Los k bits del mensaje son desplazados dentro del registro r y simultáneamente liberados hacia la salida. Luego de desplazar k veces, los registros r_0 a r_{n-k-1} contienen los bits de paridad. Entonces se abre la llave de realimentación, y la llave de salida se coloca en la posición que permite que los bits de paridad lleguen a la salida.

El vector de código transmitido, puede estar perturbado por ruido, por lo tanto, el vector recibido puede no coincidir con el transmitido. Si se supone que se transmite la palabra de código u(X) representada por:$u(X) = m(X)g(X)$ El polinomio $z(X)$ recibido estará formado por:

$z(X) = u(X) + e(X)$ donde;

$e(X)$ es el polinomio error

él síndrome $s(X)$ es el resto de dividir $z(X)$ entre $g(X)$, quedando estos expresados de la siguiente forma: $z(X) = q(X)g(X) + s(X)$, donde por tanto obtenemos:

$e(X) = [m(X) + q(X)]g(X) + s(X)$

De observar que al dividir $z(X)$ entre $g(X)$ obtenemos como el resto al síndrome, de igual forma que dividiendo $e(X)$ entre $g(X)$. Así de la palabra de código recibida $z(X)$ contiene la información necesaria para la corrección de los errores producidos. El cálculo del síndrome se realiza mediante circuitos similares a los usados en el codificador.[Sklar]

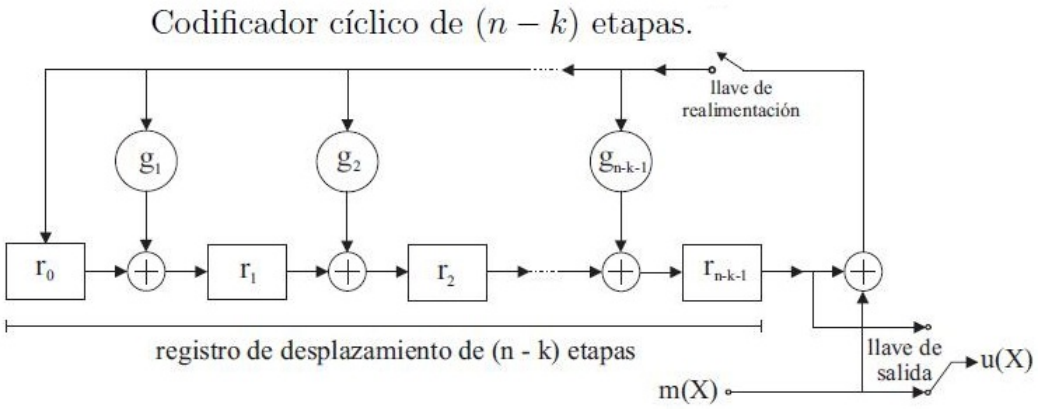


Figura 3.5.: Esquema de código cíclico [Leon W. Couch]

En general podemos resumir de los códigos cíclicos:

- Fácilidad de implementación en base a registros de desplazamiento
  - La operación de multiplicación en base a un cierto operador genera un desplazamientos cíclico en el código transmitido.
- Gran capacidad de detección y corrección de errores

- Linealidad; la suma en modulo 2 de 2 palabras del código resulta otra palabra del código.
- De acuerdo a su principio de operación: el mensaje de información de k bits es multiplicado por un operador adecuado en el extremo transmisor, generando el código de n bits ($> k$). En el extremo receptor el código recibido, con posibles errores, se divide por el operador para obtener el mensaje, si el resto es cero se valida, según el resto se corrige (FEC) o se rechaza y pide retransmisión (ARQ).

3.1.6. Códigos cíclicos sistemáticos

Sea la polinomio de la palabra código c(x) correspondiente al polinomio de datos $d(x)$

donde;

$c(x) = x^{n-k}d(x) + p(x)$

$d(x) = x^{n-k}d(x) + p(x)$

si;

$p(x) = Residuo\frac{x^{n-k}d(x)}{g(x)}$

donde

$\frac{x^{n-k}d(x)}{g(x)} = q(x) + \frac{p(x)}{g(x)}$ ó bien $q(x)g(x) = x^{n-k}d(x) + p(x)$

$q(x)$ es del orden $k-1$o menor, $q(x)g(x)$, es una palabra de código.

por tanto observamos $x^{n-k}d(x)$ representa $d(x)$ desplazada hacia la izquierda n-k dígitos, por tanto los primeros dígitos serán precisamente d, y los últimos serán los correspondientes a p(x) , los cuales son los dígitos de control de paridad .[Lathi]

Ejemplificando Sea el código cíclico (7,4) utilizando el polinomio generador; $g(x) = x^3 + x^2 + 1$

y el vector de datos $d(x) = (1010) = x^3 + x$

Por tanto $x^{n-k} = x^3$

$x^{n-k}d(x) = x^6 + x^4$

Se realiza la division polinomica, para obtener $p(x) \quad y/ó \quad q(x)$

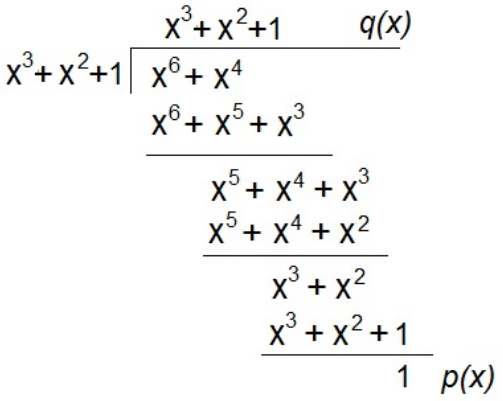


Figura 3.6.: División polinómica

Por tanto $c(x) = q(x)g(x) = x^{n-k}d(x) + p(x)$

$c(x) = (x^3 + x^2 + 1)(x^3 + x^2 + 1) = x^3(x^3 + x) + 1$

$c(x) = x^6 + x^4 + 1$

Por tanto $c(x) = (1010001)$

de esta misma forma obtenemos los demás elementos pertenecientes al código

d	c
1111	1111111
1110	1110010
1101	1101000
1100	1100101
1011	1011100
1010	1010001
1001	1001011
1000	1000110
0111	0111001
0110	0110100
0101	0101110
0100	0100011
0011	0011010
0010	0010111
0001	0001101
0000	0000000

3.1.6.1. Algoritmo para codificación

Obtener n-k, del ejemplo anterior (7,4)

n-k=3

realizar $x^{n-k}d(x) = x^3(x^3 + x) = x^6 + x^4$

Dividir para obtener q(x) y/o p(x), para poder aplicar

Por tanto $c(x) = q(x)g(x) = x^{n-k}d(x) + p(x)$

La forma que presentaremos la división, es mediante desplazamientos y el uso de or exclusiva (XOR), debido a que ésta es la forma en que se suele implementar la división polinómica, en hardware.

sea:

$d(x) = x^3 + x = (1010)$

$g(x) = x^3 + x^2 + 1 = (1101)$

$x^{n-k}d(x) = x^6 + x^4 = (1010000)$

Dividir $x^{n-k}d(x) \div g(x) = (x^6 + x^4) \div (x^3 + x^2 + 1) = (1010000) \div (1101)$

La multiplicación se realiza con un registro de corrimiento de n-k, de derecha a izquierda, colocando 0's a las nuevas posiciones agregadas.

$d(x) = x^3 + x = (1010)$	palabra original
$n - k = 3$	bits a agregar
$x^{n-k}d(x) = x^6 + x^4 = (1010000)$	resultante

para la división, se sigue el método;

1. Cuando el primer bit de la palabra a dividir es 1 se realiza la OR-exclusiva, pero si el primer digito es 0 no se afecta la palabra.
2. Se rota la palabra que ésta siendo dividida, de izquierda a derecha
3. Repetir punto 1, hasta que el número de rotaciones sea igual a n-k
4. Rotar inversamente n-k.

Así, sea $x^{n-k}d(x)$ dividido entre g(x);

XOR	1 1	0 1	1 0	0 1	0	0	0	g(x)
	0	1	1	1	0	0	0	1er resultado

rotando

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

Se repite el procedimiento hasta n-k.

1 1	0 1	1 0	0 1	0	0	0	XOR
0 1	1 1	1 1	1 0	0 0	0 0	0 0	Rotar
1 1	1 1	1 0	0 1	0	0	0	XOR
0 0	0 1	1 1	1 0	0 0	0 0	0 0	Rotar
0 1	1 1	1 0	0 1	0	0	0	XOR
0 1	1 1	1 0	0 0	0 0	0 0	0 0	Rotar
1 1	1 1	0 0	0 1	0	0	0	XOR
0	0	0	1	0	0	0	Residuo
0	0	0	0	0	0	1	Rotacion inversa n-k

Por tanto el residuo es $p(x) = (0000001)$

Modo simplificado, haciendo corrimiento del polinomio generador, en lugar de la palabra;

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

vemos que el residuo es el mismo $p(x) = (0000001)$

obtenemos la palabra codificada mediante;

$c(x) = q(x)g(x) = x^{n-k}d(x) + p(x)$

por tanto;

$c(x) = x^6 + x^4 + 1$

lo que es igual a $c(x) = (1010001)$

Comprobamos por tanto la validez del método comparándolo con la ejemplificación anterior.

3.1.6.2. Códigos de redundancia cíclica (CRC)

Códigos de redundancia cíclica son eficientes para la detección de errores, se encuentran estandarizados y son muy usados en la práctica. El extenso uso de estos códigos es debido a su facilidad de implementación en hardware.

Estos códigos se basan en el uso de un polinomio generador $G(X)$ de grado$r$, y en el principio de que n bits de datos binarios se pueden considerar como los coeficientes de un polinomio de orden $n-1$.

CRC-12	$g(x) = x^{24} + x^{11} + x^3 + x^2 + x + 1$
CRC-16	$g(x) = x^{16} + x^{15} + x^5 + 1$
CRC-ITU ó CRC-CCITT	$g(x) = x^{16} + x^{12} + x^5 + 1$
UMTS 3G	$g(x) = x^{24} + x^{23} + x^6 + x^5 + x + 1$
	$g(x) = x^{16} + x^{12} + x^5 + 1$
	$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$
	$g(x) = x^8 + x^7 + x^4 + x^2 + x + 1$

Cuadro 3.1.: Estandarización, códigos cíclicos

Uno de los polinomios generadores que más se suelen utilizar es el estándar CCITT:

$x^{16} + x^{12} + x^5 + 1$.

Este polinomio permite la detección de:

- 100 % de errores simples.
- 100 % de errores dobles.
- 100 % de errores de un número impar de bits.
- 100 % de errores en ráfagas (en una serie sucesiva de bits) de 16 o menos bits.
- 99.99 % de errores en ráfagas de 18 o más bits.

Códigos Cíclicos de máxima longitud Los Código del tipo $(n, k) = (2^m + 1, m)$ tasa de código relativa baja $R = \frac{m}{(2^n-1)}$ donde m es el largo del registro de desplazamiento que son los bits de mensaje, cargados secuencialmente en el registro. para esto, se carga el registro con los m bits, y se hace circular $n = 2^m - 1$ desplazamientos para generar los n bits de código de salida, excluyendo la palabra de m 0's. Consiguiendo con ésto;

$d_{mín} = 2^m - 1$

detecta $2^m - 2$ errores

corrige $t = [(d_{mín} - 1)/2] = 2^{m-1} - 1$ errores

3.1.7. Codificación a bloques $C_b(n, k)$

Funciona mediante un mapeo de símbolos binarios de entrada k (bits de mensaje, que en conjunto constituyen 2^kposibles mensajes) a símbolos binarios de salida n. Por consiguiente, el codificador de bloque es un dispositivo sin memoria, debido a que $n > k$, el código puede seleccionarse para proveer redundancia, tal como los bits de paridad, los cules son utilizados por el decodificador para proporcionar alguna detección y corrección de errores, operando estos bits en el proceso de redundancia sobre la información del mensaje, de forma que aplicando la operación inversa en el receptor sea posible recuperar el mensaje original. Por tanto durante el proceso de decodificación, los bits de redundancia son descartados, ya que no contienen información de mensaje. Estos códigos, su tasa de código está dada por $R = \frac{k}{n}$. Los valores prácticos para R varían en el rango de $\frac{1}{4}$ y $\frac{7}{8}$ y los valores de k varían en el rango de 3 a varios cientos[Clark y Cain]. Este concepto tiene relación con el ancho de banda empleado en la transmisión cuando se utiliza codificación. Por ejemplo si se empleara una tasa $\frac{n}{k} = \frac{2}{3}$ entonces se debe tener en cuenta que en el mismo tiempo T en que originalmente y sin codificar se alojan dos señales que representan a los dos bits, se tiene ahora que poder alojar a tres señales con la misma ocupación temporal, de manera que el tiempo de duración de cada señal pasaa a ser de $T/2$ a $T/3$, y la ocupación espectral es en consecuencia mayor. tambien podemos ver que en el caso de alamacenamiento de datos digitales la información codificada requerirá de mayor espacio físico para ser almacenada. Por estas razones será entonces conveniente mantener la tasa del código en niveles razonables, a pesar de que este objetivo compromete la capacidad de correción del código.

Dado que los 2^k mensajes son transformados en palabras de n bits, se puede interpretar este procedimiento como la aplicación de una expansión del espacio vectorial de dimensión 2^k a otro de dimensión mayor 2^n, del cual se eligen de forma conveniente solo 2^k vectores.[Arnone Ing.]

Éste codificador tiene una capacidad más amplia de detección de errores que la corrección de estos, dado que para esto se necesita conocer la posición y la magnitud del error.

3.1.7.1. Códigos lineales de bloque

Es una clase de códigos de chequeo de paridad que puede ser caracterizado por (n, k), el códificador un bloque k digitos de mensaje dentro de un bloque de n palabra código, construido desde un alfabeto dado de elementos, donde el alfabeto consiste en dos elementos (0 & 1), el código es un código binario.

Sea una secuencia binaria, segmentada en bloques de u con una longitud fija de k-bits, esta es codificada en bloques$v$ de n-bits, donde $n > k$. Por tanto tenemos que:

Existen 2^k distintas palabras código validas a enviar, éste es el “código bloque”

Existen 2^n distintas palabras código posibles a recibir

Caracteristicas de un código lineal

- Cumple con la suma módulo 2.
- En un código lineal (n,k) es posible encontrar k palabras de código linealmente independientes, $g_0, g_1, ...g_{k-1}$ tal que cada palabra código v es una combinación lineal de esas k palabras código:
  - $V = u_0 g_0 + u_1 g_1 + ... + u_{k-1} g_{k-1}$
  - con $u_i = 0$ ó 1
  - $i \geqq 0$ e $i < k$
  - $\vec{V} = \vec{u} \bullet G$
  - Donde tenemos a la matriz Generadora G binaria de (k, n), definida como

$$G = \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & . & . & . & g_{0,n-1} \\ g_{10} & g_{11} & . & . & . & g_{1,n-1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ g_{k0} & . & . & . & . & g_{k,n-1} \end{bmatrix}$$

Si $u = (u_0, u_1, ...u_{k-1})$ es el mensaje a codificar, entonces podemos generar el correspondiente código de la siguiente forma:

$$v = uG = u \bullet \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ g_{k-1} \end{bmatrix} = u_0 g_0 + u_1 g_1 + ... + u_{k-1} g_{k-1}$$

De manera más general se puede considerar a ésta codificación como una asignación biyectiva entre los 2^k vectores del espacio vectorial de mensaje y 2^n vectores del espacio vectorial codificado. Cuando los valores de k y n son pequeños, la asignación puede realizarse a travéz de una tabla, pero cuando la magnitud de éstas cantidades es grande, es necesario encontrar un método o mecanismo de generación de código. En este sentido la linealidad de las operaciones de este mecanismo simplifica grandemente el proceso de codificación. Se dice que un código de bloques de longitud n y 2^k palabras de mensaje es un código de bloques lineal $C_b(n, k)$, si las 2^kplabras de código forman un subespacio vectorial de dimensión k, del espacio vectorial V_n de todos los vectores y componentes de longitud n.

La elección de los vectores de n bits debe hacerse empleando la menor redundancia, y maximizando la distancia o separación entre las palabras. En un código de bloques lineal, el conjunto de 2^k palabras constituye un subespacio vectorial del conjunto de vectores de n palabras, y por tanto se pude asegurar que la suma de dos palabras cuales quiera del código será también otra palabra o vector del código.

3.1.7.2. Matriz Generadora

Si k es grande, la tabla de implementación del codificador llega a ser contraproducente, para un código (127,92) habrían 2^{92} o aproximadamente $5x10^{27}$ vectores código lo cual requeriria una cantidad de memoria enorme para contener tales palabas código. Afortunadamente es posible reducir la complejidad generando las palabras código conforme son necesarias, en lugar de almacenarlas.

De un conjunto de palabras código que forman el bloque lineal de código de dimensión k, en el sub espacio del vector binario dimension n, de modo (k<n), para el cual siempre es posible encontrar un conjunto de n-tuples, menos que 2^k que pueden generar todas las 2^k palabras código del sub espacio.

Matriz generadora de código de bloques G Dado que un código lineal de bloques $C_b(n,k)$ es un subespacio vectorial del espacio vectorial V_n, será posible encontrar k vectores linealmente independientes que son a su vez palabras del código $g_0, g_1, ..., g_{k-1}$. Estos vectores linealmente independientes se organizan en una matriz generadora que mantiene la forma:

$G = [g_0, g_1, ..., g_{k-1}]^T$

Si el vector de mensaje se expresa como $m = (m_0, m_1, ..., m_{k-1})$, entonces la palabra o vector de código se obtiene como: $c = m \circ G$

De ésta forma se establece un mecanismo matricial para la generación de las palabras del código.

Forma sistemática de un código de bloques En esta forma, los vectores de código aparecen constituidos por $(n-k)$ bits de paridad seguidos por los k bits del vector mensaje. Esta forma de organizar la palabra codificada podría ser hecha al reves es decir ubicando los bits de mensaje al principio de la palabra, y los de paridad al final. La manera en que esto sea hecho no modifica las propiedades de los códigos de bloques, aunque algunas expresiones matemáticas de las operaciones de codificación adoptan lógicamente una forma diferente en cada caso. Un código lineal sistemático de bloques (n,k) está especificado unívocamente por la matriz generadora de la forma: $G = [PI_k]$, donde I_k es la matriz identidad de dimensión kxk y P es la matriz paridad de dimensión $kx(n-k)$.

La matriz generadora G contiene k vectores fila linealmente independientes, que generan el subespacio vectorial S que pertenece al espacio vectorail V_n. También por las filas de una matriz H. A ésta matriz H se le denomina: $H = [I_{n-k}P^T]$ donde P^T es la traspuesta de la submatriz de paridad P.Cada vector del espacio fila de la matriz G es ortogonal a las filas de la matriz H y viceversa.

La matriz H está construida de manera que el producto interno entre un vector fila g_i de G y un vector fila h_j de H sean ortogonales, es decir $g_i \circ H^T = 0$, por tanto tenemos de $c = m \circ G$ que:

así $c \circ H^T = m \circ G \circ H^T = 0$

3.1.8. Codificador Convolucional

A diferencia de los códigos bloques, los Códificadores Convolucionales no están restringidos a codificar los mensajes en bloque, es decir puede ser una codificación continua de largo indefinido. Están diseñados para corrección, no detección, esto debido a su forma estructural. Este es un codificador continuo en el que la secuencia de bits depende de los bits previos. La codificación se basa en registros de desplazamiento de K etapas de k bits (K se denomina constraint length; longitud de restricción, y es básicamente un parametro de medicion de la forma en que los bits previos afectan a los que se éstan generando en la salida) y n funciones algebraicas generadoras. Los bits de mensaje entran al registro en paquetes de k bits, y salen codificados en paquetes de n bits, los valores típicos para k y n varían en el rango de 1 a 8[Leon W. Couch], la tasa de código es, $R = \frac{k}{n}$ y su rango se encuentra entre $\frac{1}{4}$ y $\frac{7}{8}$[Clark y Cain]. K da origen a una memoria de $Kxk - 1$ bits que generan la secuencia codificada, lo que le da el nombre de convolucion. de modo que éste codificador, quedando especificado por:

- n número de salidas
- k número de entrada
- m memoria del código
- $R = \frac{k}{n}$ tasa del código

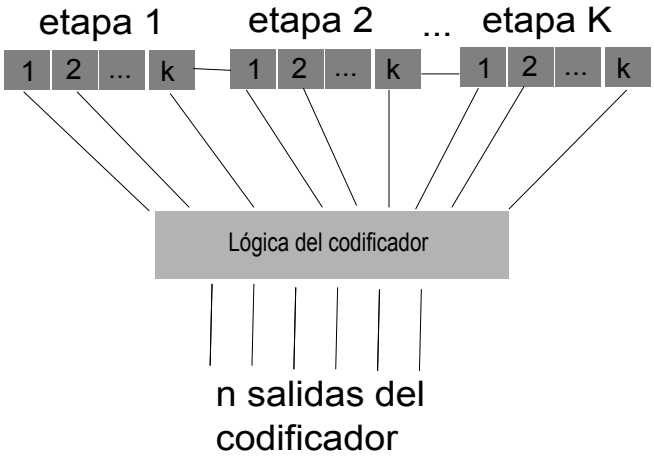


Figura 3.7.: Codificador convolucional

Donde el nivel de memoria K debe hacerse grande para aumentar la capacidad de detección de error del código. En un esquema general, de un decodificador convolucional, emplea kK registros que operan en el campo de Galois GF(2) (suma y multiplicación binaria en modulo 2).

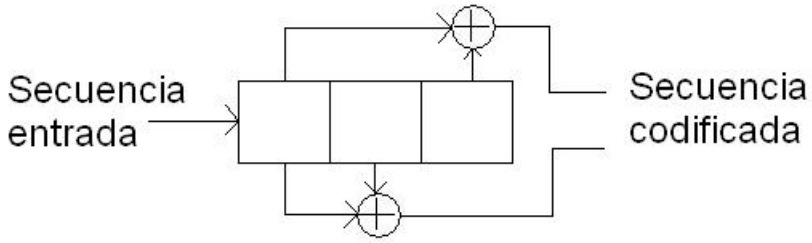


Figura 3.8.: Codificador convolucional no sistematico

De entre las ventajas que tiene el codificador a convolucional:

- Simplicidad en el decodificador
- Un buen rendimiento corrector de errores se consiguen ganancias de codificación alta, sobre todo en canales modelados con ruido AWGN (Additive White Gaussian Noise).
- Los códigos convolucionales tienen un excelente rendimiento cuando los comparamos con los códigos de bloque de complejidad codificador/decodificador equivalente. Además, fueron los primeros códigos en los que se implementaron algoritmos efectivos de decisión blanda, (soft decision).
- Trabajan con un flujo continuo de símbolos que no están divididos en bloques finitos de mensajes. Aunque es importante indicar que también funcionan si la secuencia de símbolos de entrada no es continua, sino que está dividida en tramas.
- El codificador convolucional, varia sus configuraciones de aplicación, por lo cual podremos encontrar recursivos, no recursivos, sistemáticos, no sistemáticos y de diversos índices de codificación, como se ejemplifica en la figura 3.9

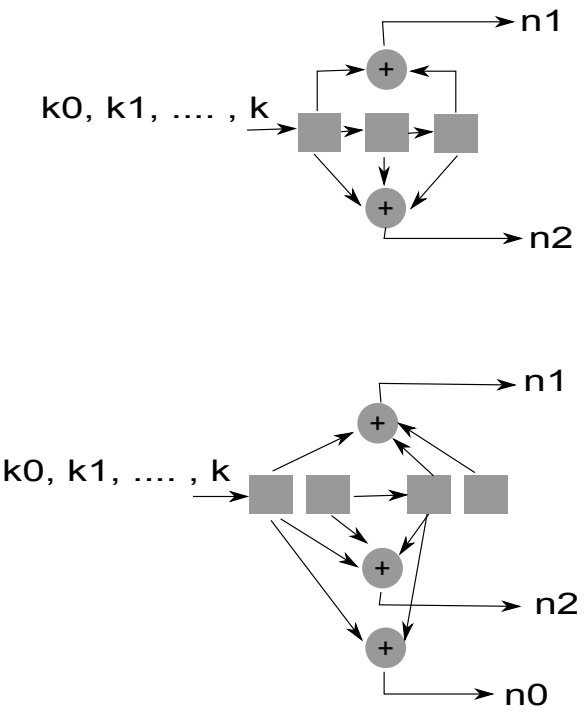


Figura 3.9.: Ejemplo codificación convolucional [BHARGAVA]

Otra forma de representar un codificador convolucional es mediante un diagrama de estados, el cual es posible programar en VHDL, en la figura siguiente se ve el esquema general del diagrama de estados para un codificador convolucional de 4 estados.

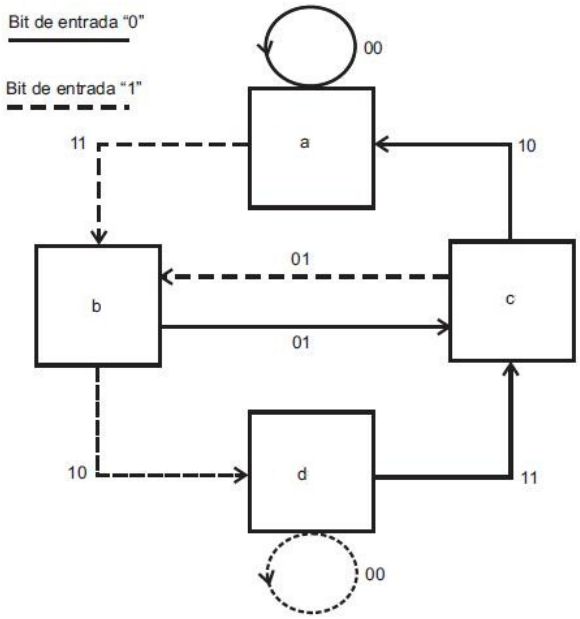


Figura 3.10.: convolucional diagrama estados

Los códigos convolucionales pueden utilizarse en varias estructuras de codificación diferentes:

1. Un codificador convolucional en solitario.

2. Un codificador de bloque y uno convolucional concatenados.

3. Dos, o más, codificadores convolucionales concatenados y separados por un entrelazador. Esto constituye un turbo código.

Código Convolucional sistemático El termino sistemático se refiere a aquellos códigos en los que una de sus ramas generadoras de los símbolos de código emite justamente los bits de datos originales.

Así con un codificador sistemático tendrá sus etapas conectadas solamente a $r-1$ sumadores, y la r-esima se reemplaza por una conexión directa desde la etapa correspondiente al conmutador. La siguiente figura muestra un codificador sistemático con $R = \frac{1}{2}$ y $M = 3$.[Viñé Viñuelas]

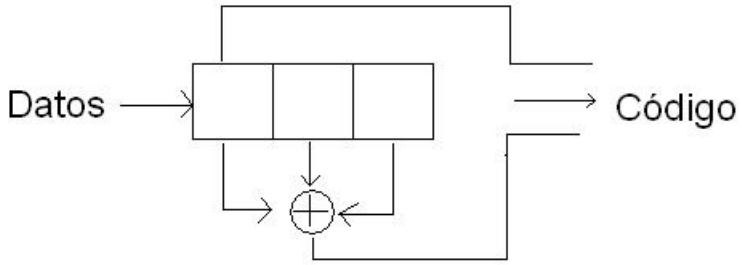


Figura 3.11.: Código Convolucional sistemático

Código RA Como se ha mencionado existe una enorme variedad de códigos, los cuales mantienen los principios de codificación, ajustándolos de diversas formas.

Aquí presento el código RA, el cual se verá, aplica los principios de un código simple, un mezclador y un código convolucional realimentado.

Estos códigos consisten en tres pasos de codificación simples:

1. Primer bloque: Código de repetición; es el más simple código de corrección, con buenas propriedad de distancias pero baja tasa de codificación.
2. Un paso de permutación (conocida).
3. Un código convolucional con generador $G(x) = \frac{1}{1+x}$. Esto seria como "acumular", es decir va sumando los bits en entrada (todo este proceso se realiza en modulo 2).

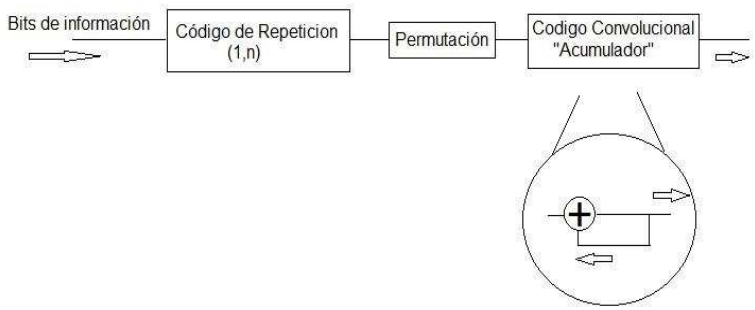


Figura 3.12.: Esquema código RA

3.1.9. Turbo codificación

Este esquema fue presentado originalmente por Berrou, Blavieux y Thitimajshima en 1993, consiguiendo revolucionar los la codificación proponiendo un mecanismo

iterativo que proporciona ganancias de código óptimo, las cuales llevan al sistema a trabajar cerca del límite de Shannon.

La estructura inicial son codificadores convolucionales ordenados en configuración concatenada en paralelo involucrando también un permutador de datos pseudo aleatorio, de manera que cada dato de entrada es codificado dos veces con la ayuda de un permutador aleatorio, consiguiendo con esto independizar estadísticamente las dos secuencias de datos generadas. Por esto se le denomina permutador al entrelazador de datos, que es utilizado clásicamente en un código turbo.

La estructura más usual de un decodificador turbo está basado en el uso de codificadores convolucionales, los cuales son construidos con máquinas secuenciales de estados infinitos de respuesta impulsiva infinita, conocidos como codificadores convolucionales sistemáticos recursivos (CCSR), que generalmente tienen tasas de $k = n = 1 = 2$, frecuentemente se emplea la técnica de eliminación selectiva de salidas, donde alguno de los datos de redundancia es omitido de acuerdo a una ley conocida que permite entonces mejorar la tasa del código.

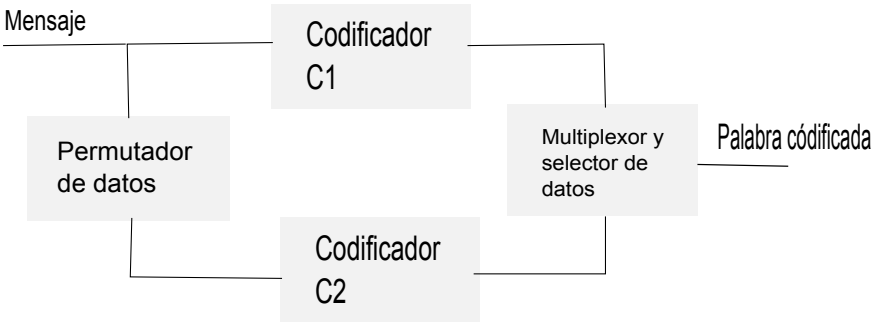


Figura 3.13.: Esquema del codificador turbo

Codificador Reed-Solomon Los códigos digitales binarios han sido fundamentales en la creación de nuevas formas de codificación llegando a trascender su estructura y/o lógica a nuevas formas de codificación, las cuales integran bajo un solo código, diversidad de principios y argumentos. Por ejemplo, en el caso de los códigos FEC se han desarrollado nuevos códigos como el Reed-Solomon que es un código cíclico y una subclase de códigos de bloque, no es un código digital binario, por lo que existen diferencias que lo excluyen de los objetivos de este trabajo, aun que sea posible su implementación en VHDL, y solo para resaltar las similitudes presentes con los códigos digitales binarios, se presentara en forma general al codificador Reed-Solomon.

Los códigos Reed-Solomon se basan en una área especializada de la matemática llamada campos de Galois o campos finitos, estos tienen la propiedad de que las operaciones aritméticas sobre elementos del campo siempre tienen un resultado en el campo, las propiedades de estos campos son la base de varios algoritmos en el área de corrección de errores y procesamiento digital de señales. Sin embargo al ser

tan amplia y extensa ésta área, quedan los detalles particulares fuera de la intensión de este trabajo, de modo que se acotara la explicación respectiva, refiriéndose a los algoritmos de un modo más general, dejando bibliografía anexa para él lector interesado en el tema.

Usado ampliamente en la detección y corrección de errores, tanto en transmisión de información, como en dispositivos de almacenamiento, el codificador Reed-Solomon es indicado como RS(n,k), con símbolos de s bits, significando que el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra de código de n símbolos, por tanto tendremos n-k símbolos de paridad s bits cada uno. Permitiendo al decodificador corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t=n-k.

Resumiendo:

s número de bits por simbolo

n número de bits despues de códificacion

k número de bites de datos antes de la códificacion

t capacidad de corrección de errrores

Las propiedades del código Reed-Solomon lo hacen adecuado para aplicaciones en donde ocurren errores en ráfaga, el esquema de corrección de errores es trabajar primero en la construcción de un polinomio de los símbolos de los datos a transmitir y envía una sobre muestra del polinomio, en lugar de los datos originales. La capacidad de corrección de errores de un RS es determinada por (n-k), la medida de la redundancia en el bloque

En el siguiente esquema se muestra una típica palabra de código Reed-Solomon:

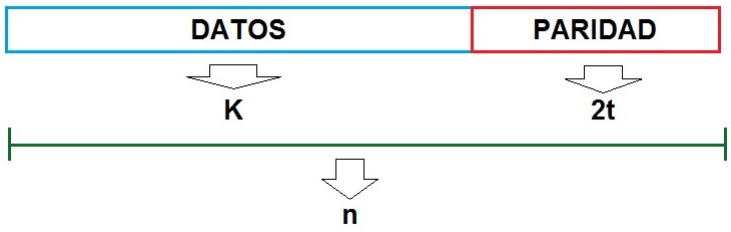


Figura 3.14.: Esquema de palabra bajo algoritmo Reed-Solomon

Descripción genérica de los algoritmos Reed-Solomon

Una palabra código Reed-Solomon es generada usando un polinomio especial. Todas la palabras código válidas son divisibles exactamente por un polinomio llamado polinomio generado;

$g(x) = (x - \alpha^i)(x - \alpha^{i+1})(x - \alpha^{i+2j})$

Una palabra código se genera de:

$c(x) = g(x) * i(x)$

Donde:

$g(x)$ polinomio generador

$i(x)$ bloque de información

$c(x)$ palabra de código valida

El polinomio generador de código es usado para calcular la paridad de los simbolos

3.2. Entrelazado de código (inter- leaving)

Sabiendo que si no se utiliza codificación, entonces el ruido en el canal causará errores de bit aleatorios en la salida del receptor que estarán más o menos aislados (esto es, no adyacentes). Cuando se añadió la codificación, la redundancia en el código permitió al decodificador receptor la corrección de errores en forma tal que la salida decodificada estará casi libre de errores. Sin embargo, en algunas aplicaciones ocurren pulsos anchos y largos de ruido en el canal. Si se utilizan las técnicas de codificación comunes en dichas situaciones ocurrirán ráfagas de errores a la salida del decodificador debido a que las ráfagas de ruido son más anchas que el "tiempo de redundancia" del código. Esta situación puede mejorarse utilizando el entrelazado de código. Los datos codificados se entrelazan en el lado transmisor al mezclar al azar los bits codificados, sobre un intervalo de tiempo con una longitud equivalente a varios bloques (para códigos de bloque) o varias longitudes de restricción (para códigos convolucionales). La longitud requerida del intervalo es de varias veces la duración de la ráfaga de ruido. En el lado transmisor, y antes de la decodificación, los datos con ráfagas de errores se entrelazan para producir datos codificados con errores aislados. Éstos son a su vez corregidos al pasar los datos codificados a través del decodificador. Ello produce una salida casi libre de errores, aun cuando ocurran ráfagas de ruido en la entrada del receptor. Existen dos clases de entrelazadores: de bloque y convolucionales.[Sklar]

3.3. FEC y las comunicaciones

La codificación de canal pretende darle robustez a la señal de manera que los efectos de las imperfecciones del canal sean mínimos, buscando contribuir a mejorar la calidad de la señal, mejorando por tanto, el desempeño general del sistema. Ésta se realiza en la capa física y tiene como entrada el flujo de bits definidos por la capa de enlace, sin embargo, es bueno aclarar que el término "codificación de canal" no siempre se utiliza para describir los mismo procesos, pues estos, pueden variar de acuerdo a cada estándar en particular, donde por ejemplo, algunas veces se refiere

a todo el proceso desde el Randomizer hasta el modulador de canal, mientras que otros hacen referencia únicamente a uno de estos procesos[5].[Marcano] El presente trabajo se centrara únicamente en el bloque que indicado como el proceso de forward error correction (FEC), figura 3.15 realizando las correspondientes descripciones y delimitaciones.

Figura 3.15.: Modelo general del proceso de codificación en la capa física [ETSI]

El proceso de codificación de canal resulta de especial importancia dentro de la industria de las comunicaciones pues ayuda a reducir la tasa de errores con código predefinidos en los nodos receptores de las redes de datos. Todas estas técnicas se suele usar en las redes de comunicaciones, por ejemplo en las inalámbricas, y sistemas de comunicación móvil que son vulnerables a factores como el exceso de tráfico de datos, las interferencias de transmisión máximas, y la pérdida de datos debido al ruido en el canal. La selección de una técnica determinada sólo depende del estándar de comunicación sobre el que se hace la transmisión de datos[6], por ejemplo:

- La CODIFICACIÓN CONVOLUCIONAL es la técnica de codificación de canal más usada en los sistemas de comunicación globales, y tiene muchas subcategorías que son estándares para distintos mecanismos de comunicación inalámbrica. Esta técnica requiere la codificación de un número determinado de bits, que es la unidad más pequeña de datos. Estos bits contienen información sobre los valores de datos transmitidos recientemente o actualmente dentro de la transmisión. Estos valores de datos informan al receptor sobre el tamaño y características de los datos transmitidos. La información sobre los valores de datos no sólo ayuda a los receptores a comprobar que no haya pérdida o corrupción de la ruta en los bits recibidos, sino que también ayuda a asegurarte de que los datos se reciben como se enviaron. Algunos estándares de comunicaciones móviles populares que usan la codificación de canal convolucional son GSM (Global System of Mobile Communications) y WCDMA (Wideband Code Division Multiple Access).
- La CODIFICACIÓN DE BLOQUE es otra técnica de codificación de canal habitual en las comunicaciones móviles, basada principalmente en la codificación de un

[5]Por ejemplo, en el estándar de LTE ETSI TS 136 212 V8.6.0 (2009-04), ChannelCodingse refiere sólo a los métodos usados para FEC, y de manera separada se incluye el Interleaver y la modulación.

[6]En los sistemas de comunicaciones móviles existe un esquema de codificación de canal para el canal de bajada o DL, y otro para el de subida o UL.

número determinado de bits de código dentro de los valores de datos transmitidos, para protección de datos y corrección de errores. Más específicamente, el número de bits de código codificados dentro de los datos transmitidos no sólo permanece fijo con esta técnica, sino que se clasifica de forma distinta según el número de bits transmitidos por segundo desde el emisor. Este tipo de codificación de canal se usa mucho en combinación con la codificación convolucional en las redes *GSM de tercera generación*, incluyendo los sistemas *GPRS* (General Packet Radio System), *EDGE* (Enhanced Data-rates for GSM Evolution) y *UTRAN* (Universal Terrestrial Radio Access Network).

- La CODIFICACIÓN DE CANAL TURBO requiere codificar los bits de datos en forma de símbolos de código intercalados en puntos de transmisión, que sirven como valores de comprobación de error en el receptor. En operaciones, la codificación de canal turbo usa de forma aleatoria bits de control de error de división entre dos codificadores colocados en paralelo, lo que genera códigos de comprobación de error distintos unidos a bits de datos transmitidos. Este procedimiento no sólo acelera el proceso de codificación de datos en el emisor, sino que hace posible la decodificación rápida y la reducción de errores mejorada en el receptor. Las técnicas de codificación de canal turbo suelen usarse en las redes de comunicación de largo alcance como las redes por satélite, pero algunos sistemas de comunicación móvil como *CDMA2000* (Code Division Multiple Access 2000) y *UMTS* (Universal Mobile Telecommunication System) los usan en combinación con la codificación de canal convolucional para las operaciones de transmisión de datos de largo alcance.
- La técnica LDPC (LOW-DENSITY PARITY-CHECK) de codificación de canal se usa específicamente para proporcionar protección contra errores en los datos desde distintas fuentes de ruido en la ruta de transmisión. Este tipo de mecanismo de codificación de canal para corregir errores usa los principios de escoger un límite máximo de interferencias por ruido dentro de las señales de datos en unas condiciones de transmisión normales. LDPC crea un valor máximo para la inclusión del ruido dentro de un conjunto de bits de datos a través de definir los niveles de umbral de ruido en los valores de los datos transmitidos a través de bits de paridad codificados. Este tipo de codificación de canal se ha usado como técnica de codificación de canal principal para el método de transmisión OFDM (Orthogonal Frequency Division Multiplexing), que es la técnica de transmisión de las redes móviles de cuarta generación como *LTE* (Long Term Evolution) Advanced y *E-UTRA* (evolved-UTRAN).

3.4. Estandarización en codificacion de canal

En este punto presentaremos en forma general algún, estándar y/o norma referente a los algoritmos de detección y corrección de error (FEC), a fin de connotar la importancia de éstos.

Es importante conocer la existencia de diversos procesos de codificación en un mismo sistema, como por ejemplo en los sistemas de comunicaciones móviles, los cuales mantienen un esquema de codificación de canal para el canal de bajada, y otro para el de subida, Así mismo la tasa de codificación en cada caso varían en función de la modulación y también si se trata de señalización o de datos de usuario.

Siguiendo el ejemplo de las comunicaciones móviles, existen para éstas, dos grupos importantes de canales, los de control y los de datos, cada grupo puede usar tasas de codificación diferentes. En el receptor, una vez demodulados y decodificados los bits entran al decodificador de canal a fin de corregir los posibles errores, si es el caso. El decodificador está diseñado para corregir una cantidad limitada de errores, la cual depende de la redundancia incorporada por el transmisor.

- A modo de ejemplo podemos ver distintos índices de codificación en el canal de transporte en el estándar LTE[ETSI];

TrCH	Coding scheme	Coding rate
UL-SCH	Turbo coding	$\frac{1}{3}$
DL-SCH		
PCH		
MCH		
BCH	Tail biting convolutional coding	$\frac{1}{3}$

US-SCH Physical Uplink Shared channel

DS-SCH Physical Downlink Shared channel

PCH Paging channel

MCH Physical Multicast

BCH Physical Broadcast channel

- Mientras que en el canal de control en LTE

Control Information	Coding Scheme	Coding Rate
DCI	Tail biting convolutional coding	$\frac{1}{3}$
DFI	Block code	$\frac{1}{6}$
HI	Reperirion code	$\frac{1}{3}$
UCI	Block code	variable
	Tail biting convolutional	$\frac{1}{3}$

.

DCI (Downlink Control Information); Describir la señalización (ACK/ACK y CQI)

UCI (Uplink Control Information); Describir la señalización (ACK/ACK y CQI)

La información se transmite por el PUCCH.

HI (HARQ Indicator): Indicador llevado por el PHICCH, 0 ACK positivo, 1 ACK negativo.

HARQ indicator channel

PCFICH Physical Control Format Indicator channel

A continuación mencionare algunos puntos referenciales a los estandares usados en la codificación de canal en algunos sistemas.

- LTE usa Códigos Convolucionales y Códigos Turbo, y al igual que WiMAX
- G.S.M. usa Códigos de Canal Convolucionales;

  la codificación de canal en el sistema digital de telefonía móvil utiliza un código continuo lineal (convolucional) con parámetros;

  L=5

  k=l

  n=2[**?**]

- ETSI define el estándar de codificación de canal para LTE con la longitud de restricción de 7 y tasa de codificación de $\frac{1}{3}$. El valor inicial del registro de desplazamiento del codificador se establecen en los valores correspondientes a los bits de información últimos 6 en la corriente de entrada de manera que los estados inicial y final del registro de desplazamiento son las mismas.[ETSI]

  $K = 7$

  $R = \frac{1}{3}$

Codificador convolucional, estándar ETSI, para LTE.

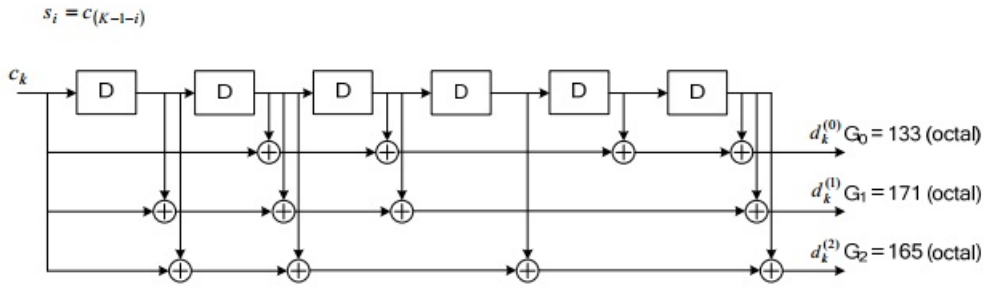


Figura 3.16.: Estándar de codificación de canal LTE. [ETSI]

Codificador Turbo, estándar ETSI, en LTE.

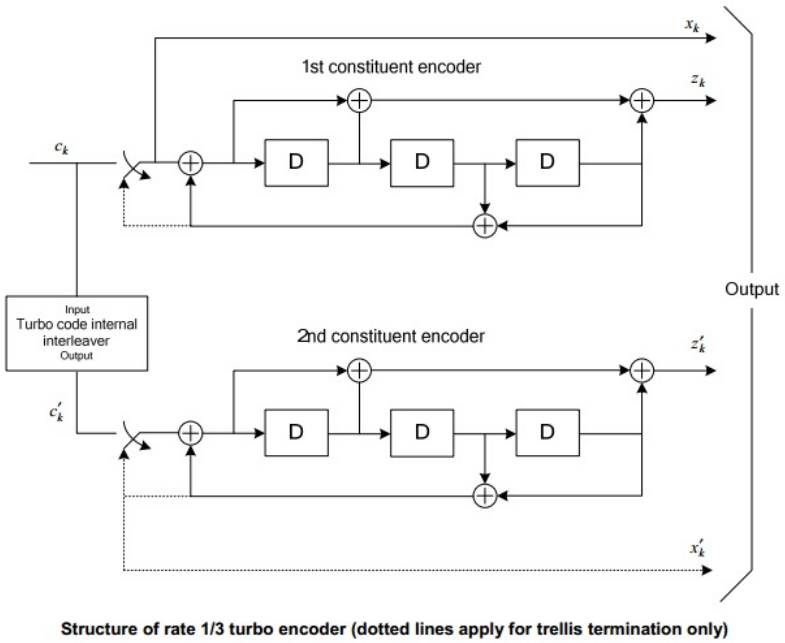


Figura 3.17.: Estándar de codificación de canal LTE [ETSI]

Entre otros tantos estándares, también encontramos el mencionado en la sección 7, en Códigos cíclicos eficientes para la detección de errores, de uso estándar.[7]

CRC-12	$g(x) = x^{24} + x^{11} + x^3 + x^2 + x + 1$
CRC-16	$g(x) = x^{16} + x^{15} + x^5 + 1$
CRC-ITU ó CRC-CCITT	$g(x) = x^{16} + x^{12} + x^5 + 1$
UMTS 3G	$g(x) = x^{24} + x^{23} + x^6 + x^5 + x + 1$
	$g(x) = x^{16} + x^{12} + x^5 + 1$
	$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$
	$g(x) = x^8 + x^7 + x^4 + x^2 + x + 1$

Cuadro 3.2.: Estandarización, códigos cíclicos

Uno de los polinomios generadores más utilizado es el estándar CCITT:
$x^{16} + x^{12} + x^5 + 1$.

[7]Detalles en la sección 7, en Códigos cíclicos.

4. Métodos y modelos de diseño

Actualmente el diseño asistido por computador (CAD) significa un proceso que emplea complejas técnicas de ordenador apoyadas en paquetes de software. La trascendencia de estas herramientas de CAD sobre el proceso de diseño de circuitos y sistemas procesadores, resulta fundamental, no solo por la adición de interfaces gráficas, que facilitan la descripción de esquemas, sino por la inclusión de herramientas, como los simuladores, que facilitan el proceso de diseño y la conclusión exitosa de los proyectos.

Automatización del diseño electrónico (EDA), es el nombre de las herramientas (software como hardware) para la ayuda al diseño asistido de sistemas electrónicos. Dentro del EDA, las herramientas CAD tienen un importante papel, sin embargo, no solo el software es importante, Las estaciones de trabajo son cada vez más veloces, elementos de entrada de diseño cada vez más sofisticados, etc. son también elementos que ayudan a facilitar el diseño de circuitos electrónicos.

El diseño de hardware tiene un problema fundamental, que no existe, por ejemplo en la producción del software. Este problema es el alto coste del ciclo de diseño - prototipacion - testeo - vuelve a empezar, ya que el coste del prototipo suele ser en general bastante elevado. Se impone la necesidad de reducir este ciclo de diseño no para incluir la fase de prototipacion hasta el final del proceso, evitando con esto la repetición de varios prototipos que encarecen el ciclo. Para ello se introduce la fase de simulación y comprobación de circuitos utilizando herramientas de CAD, de forma que no resulta necesario realizar físicamente un prototipo para comprobar el funcionamiento del circuito, economizando así el ciclo del diseño.

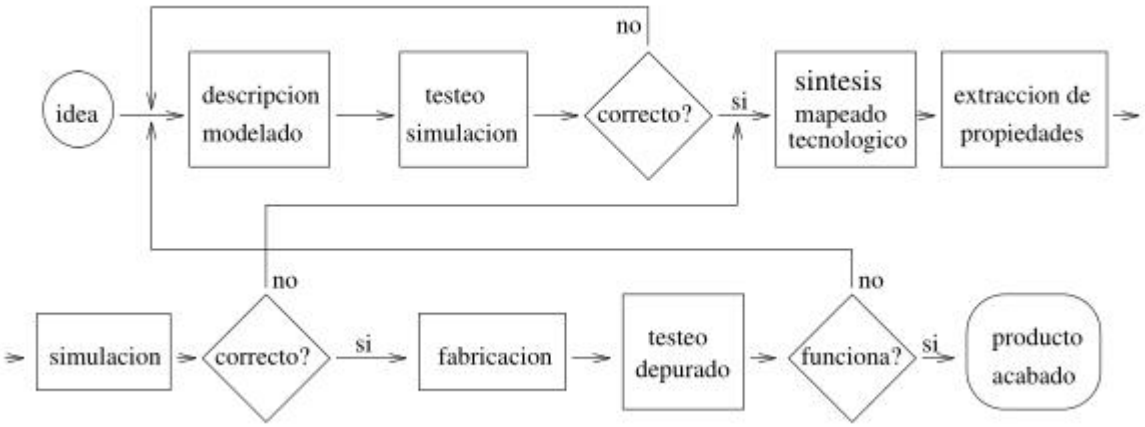


Figura 4.1.: Flujo de diseño para sistemas electrónicos y digitales. [Pardo Carpio]

Herramientas CAD para el diseño de hardware:

- Lenguaje de descripción de circuitos: Son lenguajes mediante los cuales es posible describir un circuito eléctrico o digital. La descripción puede ser de bloques, donde se muestra la arquitectura del diseño, o de comportamiento, donde se describe el comportamiento del circuito en vez de los elementos de los que está compuesto.
- Captura de esquemas: Es la forma clásica de describir un diseño electrónico y la más extendida ya que era la única usada antes de la aparición de las herramientas de CAD. La descripción está basada en un diagrama donde se muestran los diferentes componentes de un circuito.
- Grafos y diagramas de flujo: Es posible describir un circuito o sistema mediante diagramas de flujo, redes de Petri, máquinas de estados, etc. En este caso será una descripción grafica pero, al contrario que la captura de esquemas, la descripción será comportamental en vez de una descripción de componentes.
- Simulación de sistemas: Estas herramientas se usan sobre todo para la simulación de sistemas. Los componentes de la simulación son elementos de alto nivel como discos duros, buses de comunicaciones, etc. Se aplica la teoría de colas para la simulación.
- Simulación funcional: Bajando al nivel de circuitos digitales se puede realizar una simulación funcional. Este tipo de simulación comprueba el funcionamiento de circuitos digitales de forma funcional, es decir, a partir del comportamiento lógico de sus elementos (sin tener en cuenta problemas eléctricos como retrasos, etc.) se genera el comportamiento del circuito frente a unos estímulos dados.
- Simulación digital: Esta simulación, también exclusiva de los circuitos digitales, es como la anterior con la diferencia de que se tienen en cuenta retrasos en

la propagación de las señales digitales. Es una simulación muy cercana al comportamiento real del circuito y prácticamente garantiza el funcionamiento correcto del circuito a realizar.

- Simulación eléctrica: Es la simulación demás bajo nivel donde las respuestas se elaboran a nivel del transistor. Sirven tanto para circuitos analógicos como digitales y su respuesta es prácticamente idéntica a la realidad.
- Realización de PCBs: Con estas herramientas es posible realizar el trazado de pistas para la posterior fabricación de una placa de circuito impreso.
- Realización de circuitos integrados: Son herramientas de CAD que sirven para la realización de circuitos integrados. Las capacidades graficas de estas herramientas permiten la realización de las diferentes mascaras que intervienen en la realización de circuitos integrados.
- Realización de dispositivos reprogramables: Con estas herramientas se facilita la programación de este tipo de dispositivos, desde las simples PALs (ProgrammableAndLogic) hasta las más complejas FPGAs (FieldProgrammableGateArrays), pasando por las PLDs (ProgrammableLogicDevices).[Pardo Carpio]

4.1. Por qué un dispositivo reprogramable

Un dispositivo reprogramable, es una matriz de celdas que contienen una lógica configurable y elementos de memoria. Los FPGA's y CPLD's, por ejemplo, son reprogramables, y por tanto, pueden ser modificados muy fácilmente. Esto provee gran flexibilidad y reduce riesgos en comparación con los ASICs (application specific integrated circuits), que no son reprogramables. Por tanto los FPGA's combinan la velocidad hardware con la flexibilidad software, unido a una relación precio/rendimiento mucho más favorable, que por ejemplo, los ASIC.[1]

Hoy en día VHDL se considera como un estándar para la descripción, modelado y síntesis de circuitos digitales y sistemas complejos. Este lenguaje presenta diversas características que lo hacen uno de los HDL más utilizados en la actualidad.

Ventajas del VHDL

- Al estar basado en un estándar (IEEE Std 1076 - 1987) los ingenieros de toda industria de diseño pueden usar este lenguaje para minimizar errores de comunicación y problemas de compatibilidad.[Pardo Carpio]

[1] Por ésta razon los diseñadores de ASIC digitales ahora también usan lenguajes descriptores de hardware, como Verilog o VHDL, para describir la funcionalidad de estos dispositivos.

- Como en todas las normas IEEE se somete a revisiones periódicas en 2000, 2002 y 2008. En estas revisiones se hacen algunos añadidos respecto a la norma anterior, pero la norma estándar de referencia sigue siendo IEEE 1076-1993.[Pardo Carpio]
- VHDL es un estándar no sometido a ninguna patente o marca registrada. No obstante es mantenido y documentado por el IEEE, por lo que existe una garantía de estabilidad y soporte.
- VHDL permite diseño Top-Down, esto es, permite describir (modelado) el comportamiento de los bloques de alto nivel, analizandolos (simulacion), y refinar la funcionalidad de alto nivel requerido antes de llegar a niveles mas bajos de abstracción de la implementacion del diseño.
- El uso de HDL permite la descripción de sistemas electrónicos digitales a cualquier nivel de abstracción
- Modularidad: VHDL permite dividir o descomponer un diseño hardware y su descripción VHDL en unidades más pequeñas.
- Al modelar al sistema independientemente de la tecnología, puede ser sintetizado y realizado mediante diferentes soluciones con muy poco esfuerzo. Esto también implica que el diseño sea reutilizable.
- El HDL proporciona un interfaz común entre las diferentes personas involucradas en el proceso de diseño.
- El uso de herramientas de síntesis automáticas permite al diseñador concentrarse más en los aspectos de la arquitectura de la estructura.
- La probabilidad de error se reduce considerablemente así como el tiempo de diseño y el esfuerzo dedicado. Todo redunda en una mejora de la calidad y tiempo de puesta en el mercado.[Arnone Ing.]
- Notación formal. Los circuitos integrados VHDL cuentan con una notación que permite su uso en cualquier diseño electrónico.
- Independencia tecnológica de diseño. VHDL se diseñó para soportar diversas tecnologías de diseño (PLD, FPGA, ASIC, etc.) con distinta funcionalidad (circuitos combinacionales, secuenciales, síncronos y asincronos), a fin de satisfacer las distintas necesidades de diseño.
- Independencia de la tecnología y proceso de fabricación. VHDL se creó para que fuera independiente de la tecnología y del proceso de fabricación del circuito o del sistema electrónico. El lenguaje funciona de igual manera en circuitos diseñados con tecnología MOS, bipolares, BICMOS, etc., sin necesidad de incluir en el diseño información Estado actual de la lógica programable concreta de la tecnología utilizada o de sus características (retardos, consumos, temperatura, etc.), aunque esto puede hacerse de manera opcional.[G. Maxines]
- Independencia de los proveedores. Debido a que VHDL es un lenguaje estándar, permite que las descripciones o modelos generados en un sitio sean

accesibles desde cualquier otro, sean cuales sean las herramientas de diseño utilizadas.

- Facilitación de la participación en proyectos internacionales. En la actualidad VHDL constituye el lenguaje estándar de referencia a nivel internacional. Impulsado en sus inicios por el Departamento de Defensa de Estados Unidos, cualquier programa lanzado por alguna de las dependencias oficiales de ese país vuelve obligatorio su uso para el modelado de los sistemas y la documentación del proceso de diseño. Este hecho ha motivado que diversas empresas y universidades adopten a VHDL como su lenguaje de diseño.[G. Maxines]

Desventajas del VHDL

- Como el diseñador pierde información precisa sobre la estructura del sistema, éste se hace más difícil de depurar, especialmente en aspectos críticos (tiempo, área y consumo).
- Algunas de las primitivas del lenguaje son difíciles de sintetizar o no tienen correspondencia clara con el hardware.
- Son difíciles de implementar circuitos que realizan operaciones matemáticas complejas. Por ejemplo, los decodificadores de baja densidad de comprobación de paridad (LDPC) y turbo códigos emplean una gran cantidad de operaciones de suma, resta, multiplicación y división de números en punto flotante.
- Los vendedores de herramientas de síntesis han definido subconjuntos para sus sintetizadores. Distintos sintetizadores admiten distintos subconjuntos y por ello se pierde en gran medida la ventaja de lenguaje estandarizado. [Arnone Ing.]

4.2. Métodos de diseño

Ascendente o Bottom - Up Consiste en diseñar un sistema o dispositivos partiendo desde el más bajo nivel de abstracción [semiconductor].

Descendente Top - Down Consiste en diseñar partiendo desde el más alto nivel de abstracción, hasta llegar a un nivel de utilización de las modernas herramientas de diseño, para poder sintetizar el producto final, a este nivel no es necesario llegar al nivel de semiconductor.

4.2.1. Modelos

MEALY; Asíncrona En éste modelo, la salida depende del estado presente, y una entrada, por lo que en su diagrama de estados se incluyen ambas para cada línea,

permitiendo mapear a las salidas.

MOORE; Síncrono En éste modelo la salida depende únicamente del estado actual, y su correspondiente diagrama de estados incluye una señal de salida para cada estado

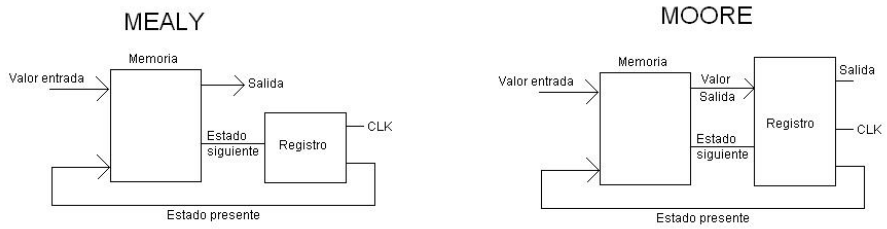


Figura 4.2.: Máquinas de estados, asíncronos y síncronos

Podemos notar la implicita presencia de éstos modelos a lo largo del sistema de comunicaciones, ya sea por ejemplo el proceso de decodificación, como un sistema secuencial sincrono, etc...[2]

4.2.2. Ingeniería concurrente

La ingeniería concurrente permite utilizar datos de un paso en el proceso de diseño antes de que el paso previo haya sido completado. Esto implica la existencia de monitores dentro del diseño para comunicar adecuadamente la actividad de diseño haciendo todos los pasos del proceso. La forma sencilla de obtener un sistema concurrente es que todos los pasos del proceso de diseño, compartan los mismos datos. Un cambio realizado con una herramienta tiene efectos inmediatos sobre la ejecución de la otra herramienta. Por tanto podemos decir que el elemento más importante en un sistema EDA que permita la ingeniería concurrente, es la base de datos. En esta base de datos cada elemento es común a todas las herramientas que componen el sistema. Las diferencias entre una herramienta y otras vendrán de lo que la herramienta ve del elemento. Así cada elemento de la base de datos estará compuesto por distintas vistas, cada una asociada generalmente a una herramienta del sistema.

En una herramienta CAD, donde se incluyan diferentes fases del proceso de diseño como captura de esquemas, simulación, etc. existe siempre la operación por la cual las herramientas posteriores del flujo de diseño (como la simulación de diseño de

[2]Mealy y Moore, en realidad son variaciones del modelo general de la maquina secuencial, clase "A" y "B" respectivamente, por lo cual se afirma la existencia de una maqina de Moore equivalente para cada maquina de Mealy. Éstas no son las únicas variaciones, también existen la clases "C" y "D".

PCBs) conocen los resultados de los pasos previos (como la captura de esquemas). A esta operación se le conoce con el nombre de preanotacion o forwardannotation y consiste en que las herramientas anteriores dentro del flujo de diseño, informan a las herramientas posteriores de los cambios realizados en el diseño. En el caso de herramientas con capacidad para ingeniería concurrente debe permitir una operación adicional, la cual es muy importante dentro de la ingeniería concurrente, es la retro anotación o backannotatio. Uno de los objetivos de la ingeniería concurrente es la posibilidad de disponer de una base de datos única, sino también, disponer de los mecanismos necesarios para que, herramientas asociadas a fases anteriores del proceso de diseño, puedan saber de los cambios realizados por herramientas posteriores incorporándolos a sus visiones especial del diseño. Para esto existe el mecanismo de backannotation que simplemente sirve para que herramientas pertenecientes a sus fases finales del proceso de diseño. Para esto existe el mecanismo de backannotation que simplemente sirve para que herramientas pertenecientes a fases finales del proceso de diseño puedan anotar cambios a la fase del diseño. Por ejemplo, en un esquema podemos especificar el encapsulado de un chip, pero puede que en la fase del diseño no se sepa todavía. Es posible que el proceso de diseño de las pistas de un circuito impreso, que será una fase posterior, ya sea conozca dicho encapsulado. En este caso la herramienta de diseño de las pistas de un circuito impreso, que será una fase posterior, ya se conozca dicho encapsulado. En este proceso de diseño de las pistas de un circuito impreso, que será una fase posterior, ya se conozca dicho encapsulado. En este la herramienta que realiza el diseño del circuito impreso puede backannotar la información del encapsulado a la herramienta de captura de esquemas.[Pardo Carpio]

5. Simulaciónes e implementación en VHDL

Existen diversas compañías internacionales que fabrican o distribuyen dispositivos lógicos programables. Algunas ofrecen productos con características generales y otras introducen innovaciones a sus dispositivos, por mencionar algunas, tenemos a Altera Corporation. Es una de las compañías más importantes de producción de dispositivos lógicos programables y también es la que más familias ofrece, ya que tiene en el mercado ocho familias: APEX™20K, FLEX®10K, FLEX 8000, FLEX 6000, MAX® 9000, MAX7000, MAX5000, y Classic™. La capacidad de integración en cada familia varía desde 300 hasta 1 000 000 compuertas utilizables por dispositivo, además de que todas tienen la capacidad de integrar sistemas complejos.[G. Maxines]

La importancia de Altera, ha llegado más allá de la tecnología que ofrece, es un miembro del grupo NASDAQ-100 y del S&P500. E incluso compite con empresas enfocadas a la producción de procesadores embebidos y ASICs estructuradas.

Las características generales más significativas de los dispositivos Altera son:

- Frecuencia de operación del circuito superior a los 175 Mhz y retardos pin a pin de menos de 5 ns.
- La implementación de bloques de arreglos integrados (EAB), que se usan para realizar circuitos que incluyan funciones aritméticas como multiplicadores, ALU, y secuenciadores. También se aplican en microprocesadores, microcontroladores y funciones complejas con DSP (procesadores digitales de señales).
- La programación en sistema (ISP), que permite programar los dispositivos montados en la tarjeta.
- Más de cuarenta tipos y tamaños de encapsulados, incluyendo el TQFP, el cual es un dispositivo delgado, de forma cuadrangular y plano, que permite ahorrar un espacio considerable en la tarjeta.
- Operación multivoltaje, entre los 5 y 3.3 volts, para máximo funcionamiento y 2.5 V en sistemas híbridos.
- Potentes herramientas de software como MAX + PLUS II y Quartus, que soportan todas las familias de dispositivos de Altera, así como el software estándar compatible con VHDL.
- Algunas familias disponen de memoria embebida, procesadores embebidos, y transceptores de alta velocidad.

Por otro lado encontramos a Xilinx, una de las compañías líder en soluciones de lógica programable, incluyendo circuitos integrados avanzados, herramientas en software para diseño, funciones predefinidas y soporte de ingeniería. Xilinx fue la compañía que inventó los FPGA y en la actualidad sus dispositivos ocupan más de la mitad del mercado mundial de los dispositivos lógicos programables.[G. Maxines]

Tanto Xilinx como Altera ofrecen una conversión del FPGA a un ASIC listo para producción comercial, llamado HardCopy ASIC para el caso de Altera y EasyPath para Xilinx. En grandes volúmenes de producción se consigue bajar mucho los costos e incrementar la seguridad del diseño. La arquitectura de este FPGA-ASIC es similar a la del FPGA, pero las celdas programables, tanto las de ruteo, como las de lógica, ya no son más celdas SRAM o FLASH, ni son programables, son conexiones fijas como en un ASIC. Pero a diferencia de un ASIC, la lógica del sistema que se ha diseñado ya ha sido verificada en hardware, en el FPGA, por lo que el riesgo de tener que re-hacer al ASIC es prácticamente nulo.

Como ya se ha mencionado, realizar un diseño en VHDL nos permite tener un diseño multiplatarforma capaz de implementarse en cualquier dispositivo reprogramable, sin embargo resulta necesario visualizar ventajas y/o caracteristicas que tiene cada dispositivo para seleccionar el más adecuado para cada diseño. En este caso, y por lo anterior, decidimos realizar a la simulacion en dispositivos de Altera

Codificación en VHDL Se realizó el diseño y simulación de un algoritmo de Hamming, Cíclico, Convolucional y Convolucional por maquina de estados, debido a que estos presentan fundamentos usados en otros tipos de codificadores, por ejemplo:

Los códigos de Hamming presentaron originalmente los parámetros de peso y distancia usados “técnicamente” en toda forma de codificación.

Entrelazar un par de códigos, principalmente los convolucionales, forman los turbo códigos.

Los códigos cíclicos sistemáticos, codificación cíclica, redundancia cíclica y codificadores de baja densidad de paridad (LDPC), se basan en realización de operaciones de división y multiplicación de polinomios requiriendo para esto demasiados recursos, además de que estos métodos han trascendido principalmente en algoritmos que trabajan las palabras código validas como símbolos de código, y no por análisis de bit, por lo que solamente mostraremos algunos de estos procesos.

Para conseguir que un diseño sea multiplataforma, el código debe estar desarrollado completamente con VHDL o Verilog. De ésta forma el hardware que se describa con este código, podrá implementarse con cualquier herramienta de síntesis y en cualquier soporte hardware que acepte VHDL o Verilog, estos dos lenguajes HDL son estándares en la industria, de manera que la gran mayoría de fabricantes y de dispositivos lógicos programables los aceptan. El diseño hardware deja de ser multiplataforma cuando se utiliza un core proporcionado por algún fabricante.[1]

[1]Un core es un módulo implementado por el fabricante utilizando su propia tecnología. Está

El trabajo se basa en la implementación de los algoritmos de codificación en un lenguaje VHDL, para lo cual necesitamos un simulador, para lo cual se ha seleccionado Quartus II 9.1, proporcionado por Altera, debido a su manejo de VHDL asi como su compatibilidad con todas las familias de dispositivos de Altera. Al no ser prioritario el analisis del dispositivo, sino la implementación como tal, él método que nos llevó a ésta y los algoritmos correspondientes, se usara la auto selección de las familias MAX3000A, MAX7000, MAX7000S Y MAX5000, dadas las ventajas antes mencionadas.

Un punto notorio es, que *la entrada y salida del bloque codificador puede ser tanto serie como paralelo*, dado que dependerá del sistema al cual pertenezca, por lo que *esto resulta indiferente para dicho bloque*, haciendo factible realizar el diseño con entradas y salidas a conveniencia, *recordando la existencia de los registros electrónicos*[2] que nos permiten realizar la transferencia de uno a otro, en caso necesario. [3]

La forma de observar y validar los resultados de la implementación se realizan mediante el Wave Editor Form, el cual es una herramienta del mismo simulador, que nos permite observar las señales de la entidad, así como las señales en la arquitectura del circuito a implementar, bajo las condiciones del dispositivo, *un apoyo más visual en algunos casos*, *puede obtenerse mediante el tools RTL del Quartus*, donde se tendrá en forma grafica el flujo de los datos atraves del diseño, lo cual no implica que el diseño este realizado necesariamente en esta forma, sino que el simulador nos permite visualizar el flujo de los datos mediante la herramienta de view RTL del simulador. Asimismo Quartus ofrece la posibilidad de realizar la implementación del diseño de hardware mediante la captura de esquemas lo cual, a pesar de no haberse realizado, nos permitiría diseñar manualmente todo el circuito y realizar su correspondiente implementación mediante un diseño propio de hardware.

Antes de comenzar con la simulación de la codificación se mostrara, la estructura general de un registro electrónico serial input - parallel output (SIPO) en VHDL, mostrando con esto la irrelevancia del tipo de entrada de datos al sistema, así mismo en consecuentes simulaciones, este registro en particular funge en ocaciones como generador de bloques o cadenas de bits.

optimizado para usarse en determinados dispositivos de su propio catálogo de productos, y con determinadas herramientas de síntesis.

[2]Los registros eléctronicos, así como las maquinas secuenciales sincronas utilizan como hardware un dispositivo denominado flip-flop como memoria, los cuales son sincronizados por una señal de reloj en forma de onda cuadrada.

Un nombre más correcto para un registro eléctronico es módulo lógico secuencial.

[3]SISO, SIPO, PISO, PIPO

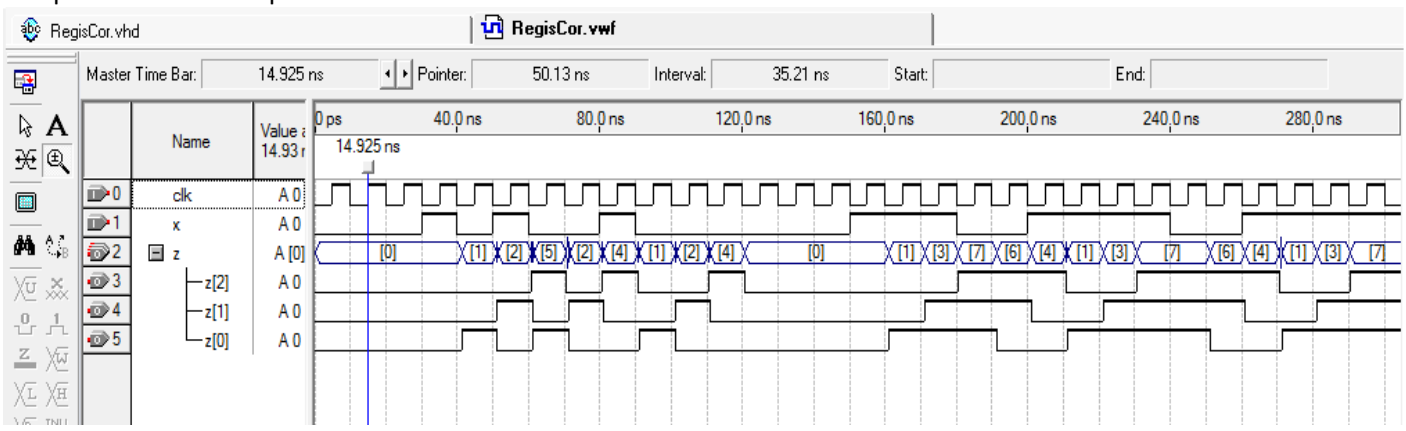
Conversión serie paralelo, paralelo-serie

La simulación e implementación de éstos registros electrónicos, se realiza comúnmente mediante la implementación de flip-flops tipo D.

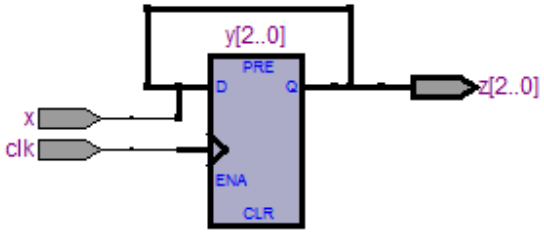
Código en VHDL:

```
entity RegisCor is
port(x,clk: in bit;
                    z: out bit_vector (2 downto 0));
end RegisCor;

architecture corr of RegisCor is
signal y: bit_vector(2 downto 0);
begin
process(clk)
begin
if clk'event and clk='1' then
        y(0) <= x;
        y(1) <= y(0);
        y(2) <= y(1);
end if;
end process;
z <= y;
end corr;
```

Comprobación de la implementación.



Visualización del flujo de datos en el registro serie-paralelo



5.1. Hamming (7,4)

Sea Hamming (7,4), un codificador que agrega 3 bits de paridad por 4 de datos, y conforme a la demostración en 3.1.4, se desarrolla el correspondiente en VHDL, mediante desarrollo modular

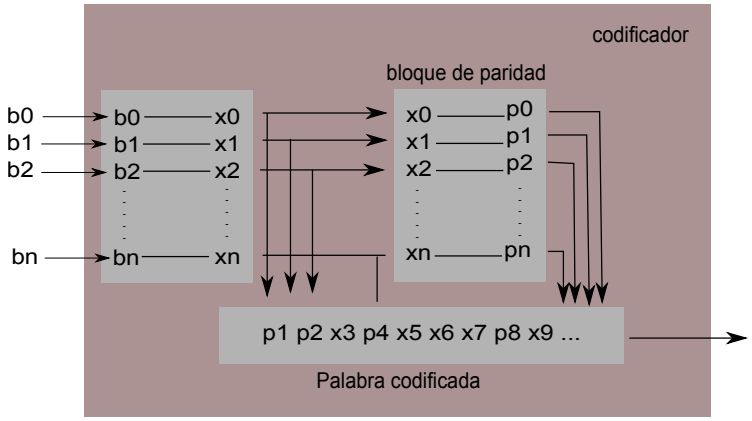


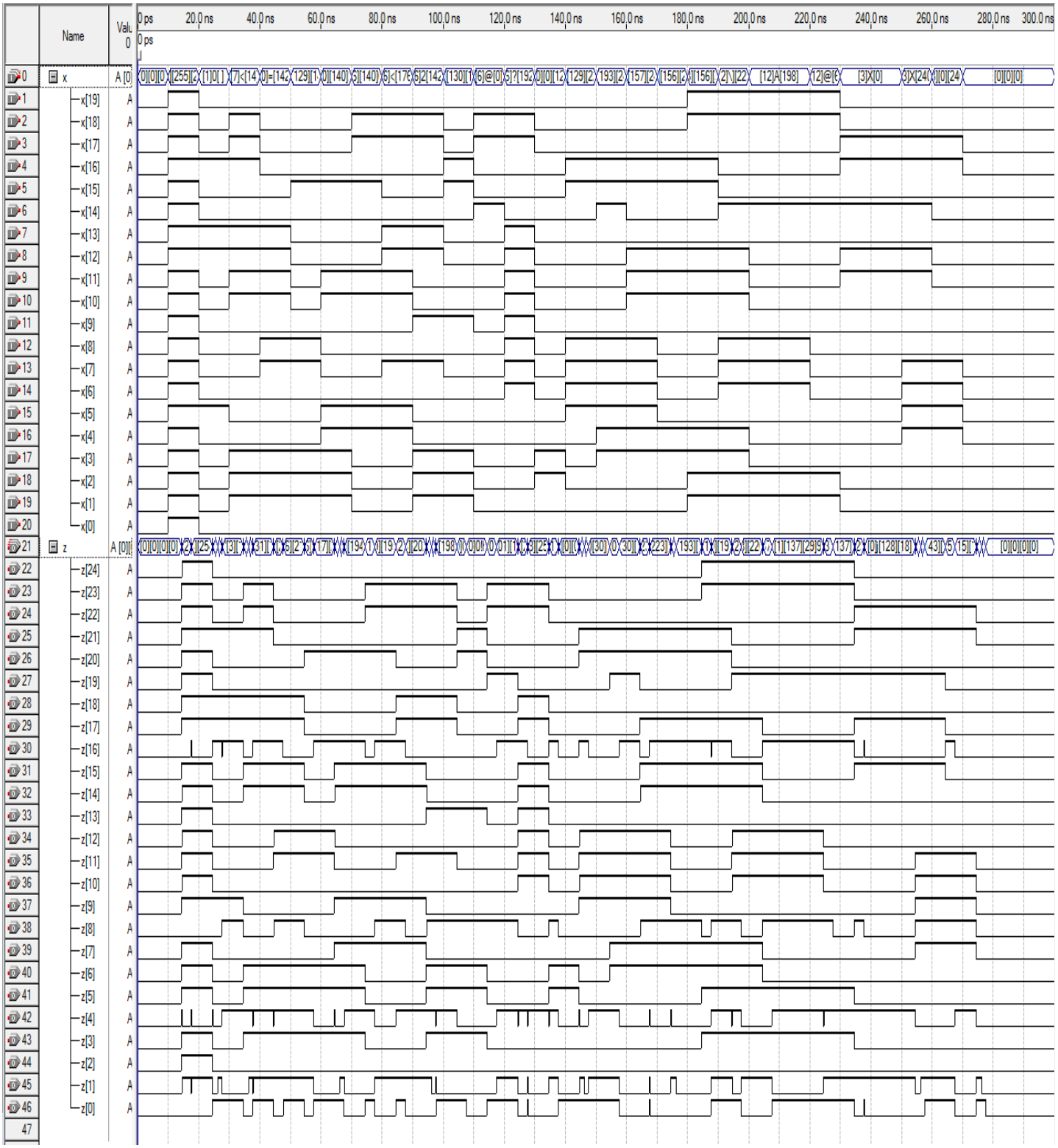
Figura 5.1.: Modulos generales a implementar Hamming

De acuerdo al método modular, se ha comenzado colocar en bloques, cada parte funcional del codificador, para poder implementar cada bloque en un lenguaje VHDL.

Código en VHDL

```
entity hamming is
port(x: in bit_vector(19 downto 0);
        z: out bit_vector(24 downto 0));
end entity;

architecture hamming7a4 of hamming is
signal y: bit_vector(24 downto 0);
signal a: bit_vector(24 downto 0);
signal p: bit_vector(4 downto 0);

begin
process(x)
begin
a(0) <= '0';
a(1) <= '0';
a(2) <= x(0);
a(3) <= x(1);
a(4) <= '0';
a(5) <= x(2);
a(6) <= x(3);
a(7) <= x(4);
a(8) <= '0';
a(9) <= x(5);
a(10) <= x(6);
a(11) <= x(7);
a(12) <= x(8);
a(13) <= x(9);
a(14) <= x(10);
a(15) <= x(11);
a(16) <= '0';
a(17) <= x(12);
a(18) <= x(13);
a(19) <= x(14);
a(20) <= x(15);
a(21) <= x(16);
a(22) <= x(17);
a(23) <= x(18);
a(24) <= x(19);
end process;

p(0) <= a(2) xor a(4) xor a(6) xor a(8) xor a(10) xor a(12) xor a(14) xor a(15) xor a(18) xor a(20) xor a(22) xor a(24);
p(1) <= a(2) xor a(5) xor a(6) xor a(9) xor a(10) xor a(13) xor a(14) xor a(17) xor a(18) xor a(21) xor a(22);
p(2) <= a(4) xor a(5) xor a(6) xor a(11) xor a(12) xor a(13) xor a(14) xor a(17) xor a(18) xor a(21) xor a(22);
p(3) <= a(8) xor a(9) xor a(10) xor a(11) xor a(12) xor a(13) xor a(14) xor a(23) xor a(24);
p(4) <= a(16) xor a(17) xor a(18) xor a(19) xor a(20) xor a(21) xor a(22) xor a(23) xor a(24);

process(p,a)
begin
y(0) <= p(0);
y(1) <= p(1);
y(2) <= a(2);
y(3) <= a(3);
y(4) <= p(2);
y(5) <= a(5);
y(6) <= a(6);
y(7) <= a(7);
y(8) <= p(3);
y(9) <= a(9);
y(10) <= a(10);
y(11) <= a(11);
y(12) <= a(12);
y(13) <= a(13);
y(14) <= a(14);
y(15) <= a(15);
y(16) <= p(4);
y(17) <= a(17);
y(18) <= a(18);
y(19) <= a(19);
y(20) <= a(20);
y(21) <= a(21);
y(22) <= a(22);
y(23) <= a(23);
y(24) <= a(24);
end process;
z <= y;
end architecture;
```

Resultados de la simulación



Podemos visualizar el flujo de datos mediante herramientas del Simulador, como se aprecia en la figura 5.2

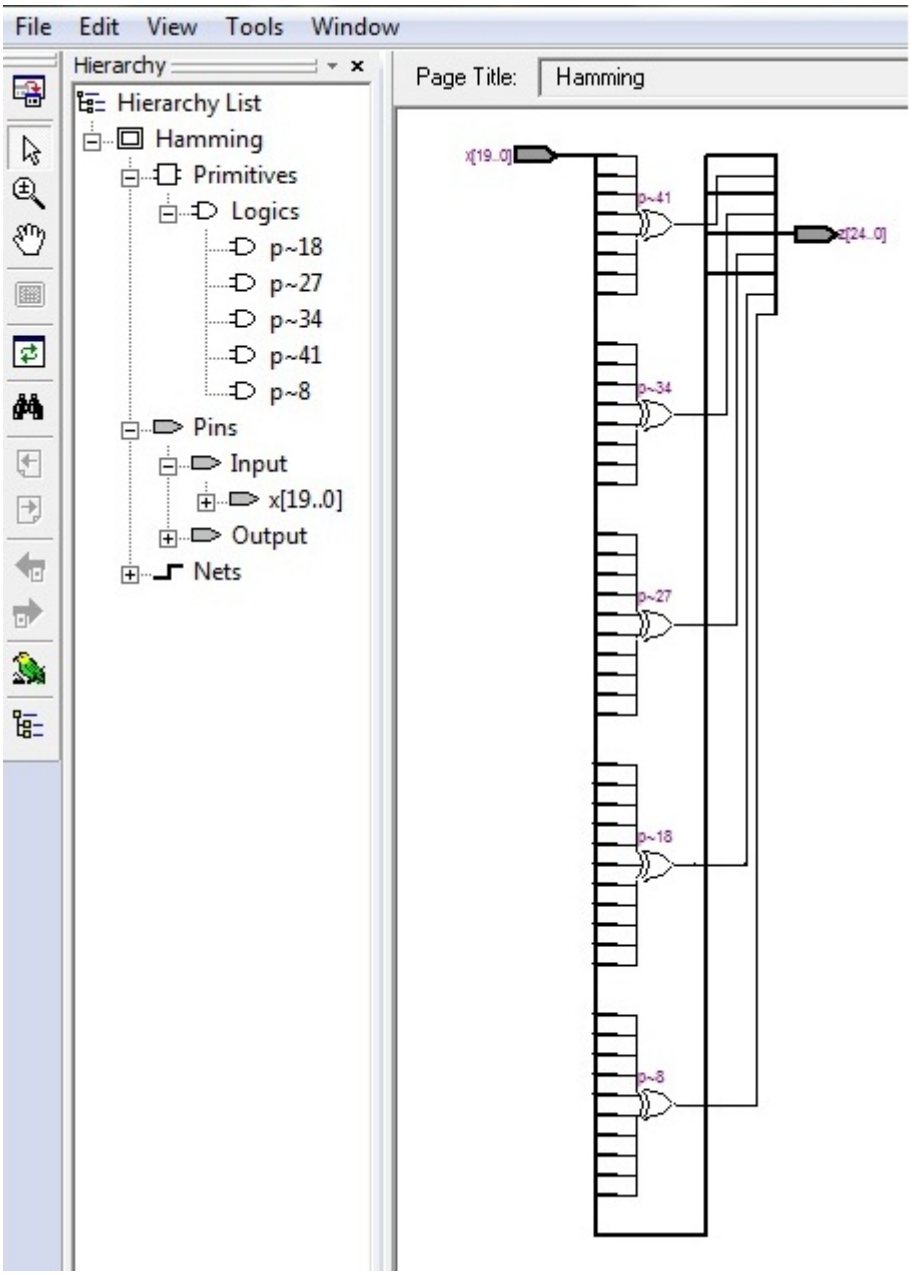


Figura 5.2.: Visualización del flujo de datos del codificador de Hamming

5.2. Codificador cíclico

Es posible utilizar el diseño comportamental, del proceso de codificación, mediante la descripción de su funcionalidad, sin necesidad de conocer la estructura interna de

un circuito.[4]

Para éste caso en particular, se realizara una descripción de flujo de datos, para lo cual retomaremos el desarrollo del código cíclico usado en 3.1.4, donde ya se ha determinado tanto su teoría, desarrollo y comprobación tenemos.

Sea el codigo (7,4) utilizando, como polinomio generador; $g(x) = x^3 + x^2 + 1$

Las palabras código validas están dadas por 2^k, por tanto, para este caso 16 palabras código, mostradas en la siguiente tabla donde d es la palabra código sin codificar, y c son las palabras codificadas.

d	c
1111	1111111
1110	1110010
1101	1101000
1100	1100101
1011	1011100
1010	1010001
1001	1001011
1000	1000110
0111	0111001
0110	0110100
0101	0101110
0100	0100011
0011	0011010
0010	0010111
0001	0001101
0000	0000000

Implementando en VHDL:

```
entity CodCiclicoTabla is
port(a: in bit_vector (3 downto 0);
z: out bit_vector (6 downto 0));
end entity;
architecture tab of CodCiclicoTabla is
signal sal: bit_vector(6 downto 0);
begin
```

[4] Aveces la descripción se divide en dos, dependiendo del nivel de abstracción, y del modo en que se ejecutan las instrucciones. Estas dos formas comportamentales de describir son la de flujo de datos y la algorítmica.[Pardo Carpio]

```
process(a)
begin
case a is
when "0000" => sal <= "0000000";
when "0001" => sal <= "0001101";
when "0010" => sal <= "0010111";
when "0011" => sal <= "0011010";
when "0100" => sal <= "0100011";
when "0101" => sal <= "0101110";
when "0110" => sal <= "0110100";
when "0111" => sal <= "0111001";
when "1000" => sal <= "1000110";
when "1001" => sal <= "1001011";
when "1010" => sal <= "1010001";
when "1011" => sal <= "1011100";
when "1100" => sal <= "1100101";
when "1101" => sal <= "1101000";
when "1110" => sal <= "1110010";
when "1111" => sal <= "1111111";
end case;
end process;
z <= sal;
end tab;
```

Durante la simulación del proceso de codificación se evaluaron todas las palabras código, para verificar la validez de la simulación, esto es posible a qué cantidad de palabras código validas [5] para este caso, son una cantidad aceptable, pues un aumento en éstas nos llevaría a un uso excesivo de la memoria del dispositivo.

[5] La cantidad de palabras código, ésta dada por 2^k

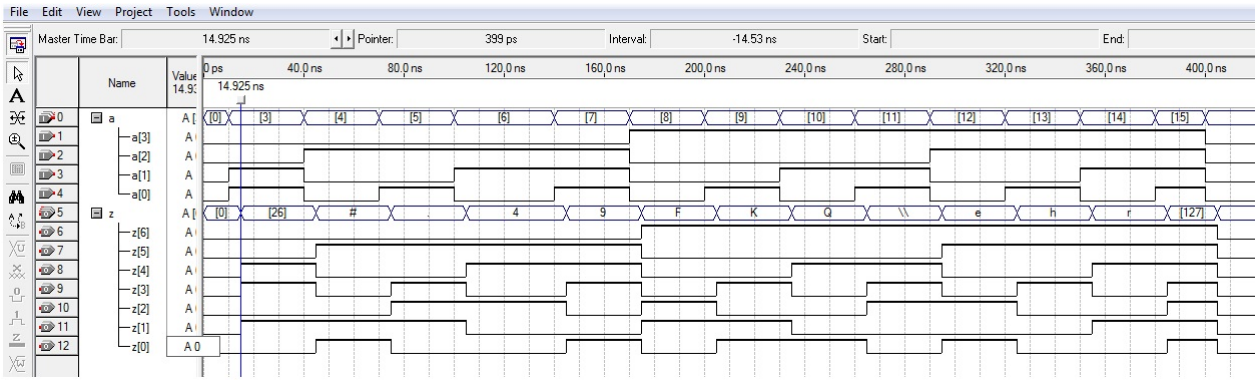


Figura 5.3.: Simulación del proceso de codificación

Visualización del flujo de datos en el codificador medinte el tools del simulador.

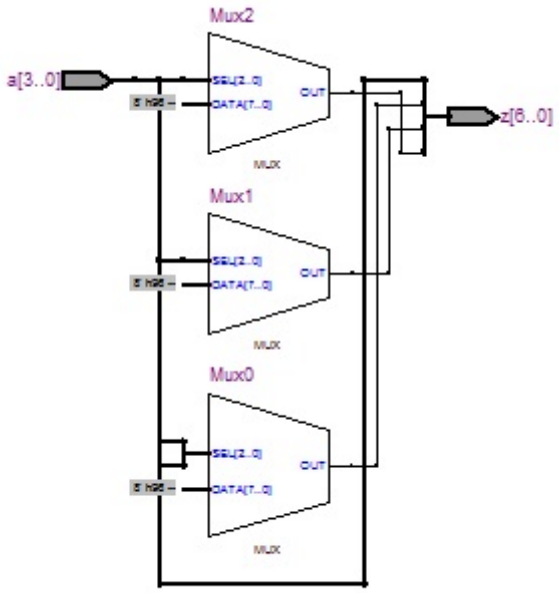


Figura 5.4.: Visualización del flujo de datos en el codificador

5.3. Codificador convolucional

Sea un Codificador convolucional de indice $R = \frac{1}{3}$, para el siguiente caso particular, se inicio por obtener las ecuaciones que describirían al sistema de codificación ilustrado en la siguiente figura.

Las salidas se encuentran dadas por:

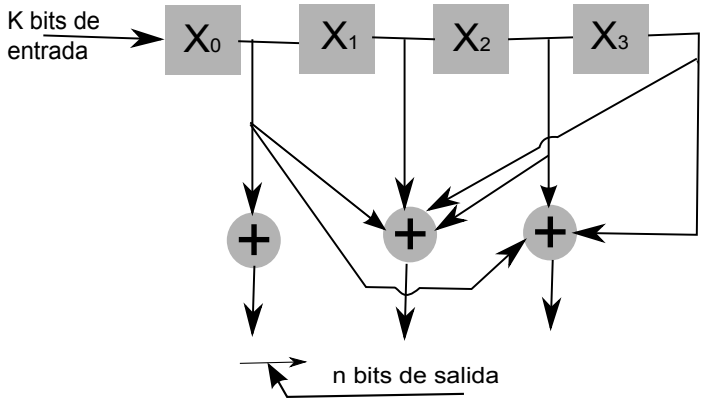


Figura 5.5.: Codificador convolucional

$V_0 = X_0$

$V_1 = X_0 + X_1 + X_2 + X_3$

$V_2 = X_0 + X_2 + X_3$

por tanto el estado del codificador dado por $X = (X_0, X_1, X_2, X_3)$

Al ser un codificador convolucional, éste depende de la secuencia de entrada, para validar la simulación, iniciando el codificador en cero, se propone una secuencia de entrada (10110), para obtener su correspondiente salida, analizando a detalle la entrada de cada bit, tenemos:

$X = (1000)$

$V_0 = 1$

$V_1 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$V_2 = 1 \oplus 0 \oplus 0 = 1$

Código de salida $V = (111)$

Si $X = (0100)$

$V_0 = 0$

$V_1 = 0 \oplus 1 \oplus 0 \oplus 0 = 1$

$V_2 = 0 \oplus 0 \oplus 0 = 0$

Código de salida $V = (010)$

Si $X = (1010)$

$V_0 = 1$

$V_1 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$

$V_2 = 1 \oplus 1 \oplus 0 = 0$

Código de salida $V = (100)$

Si $X = (1101)$

$V_0 = 1$

$V_1 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$

$V_2 = 1 \oplus 0 \oplus 1 = 0$

Código de salida $V = (110)$

Si $X = (0110)$

$V_0 = 0$

$V_1 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

$V_2 = 0 \oplus 1 \oplus 1 = 0$

Código de salida $V = (000)$

La implementación de éste codificador convolucional en VHDL, se realizó incluyendo el bloque generador de etapas k, mediante un registro electrónico de entrada serie y salida paralelo (SIPO), con el fin de visualizar la generación y aplicación directa de dicho bloque al bloque de lógica convolucional.

Código VHDL, para implementación de codificador convolucional, con tasa de codificación 1/3:

```
entity Convol1a3 is
port(x,clk: in bit;
                z: out bit_vector (2 downto 0));
end Convol1a3;

architecture conv of Convol1a3 is
signal q: bit_vector (3 downto 0);
--vector de salida
signal v: bit_vector (2 downto 0);


begin
--Generador de etapa k
process(clk)
begin
--Registro con entrada serie y salida paralelo
if clk'event and clk='1' then
        q(0) <= x;
        q(1) <= q(0);
        q(2) <= q(1);
        q(3) <= q(2);
end if;
end process;

--bloque de lógica convolucional
v(0) <= q(0);
v(1) <= q(0) XOR q(1) XOR q(2) XOR q(3);
v(2) <= q(0) XOR q(2) XOR q(3);

--asignación de salida
        z <= v;
end conv;
```

Verificamos la validez del código mediante el Wave Editor Form, con la secuencia de entrada(10110), para la cual ya se han calculado los vectores de salida, por cada entrada de bit;

$1 \rightarrow V = (111)$

$0 \rightarrow V = (010)$

$1 \rightarrow V = (100)$

$1 \rightarrow V = (110)$

$0 \rightarrow V = (000)$

Así mismo es posible observar en el mismo Wave Editor, la transición del registro SIPO, mediante el vector q.

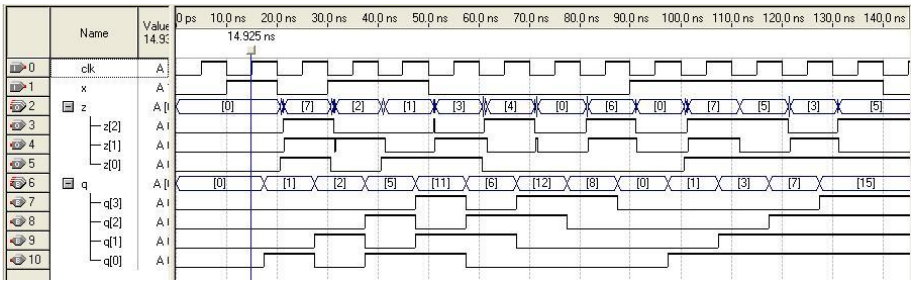


Figura 5.6.: Convolucional 1 a 3

Visualización del flujo de datos en el codificador medinte el tools del simulador.

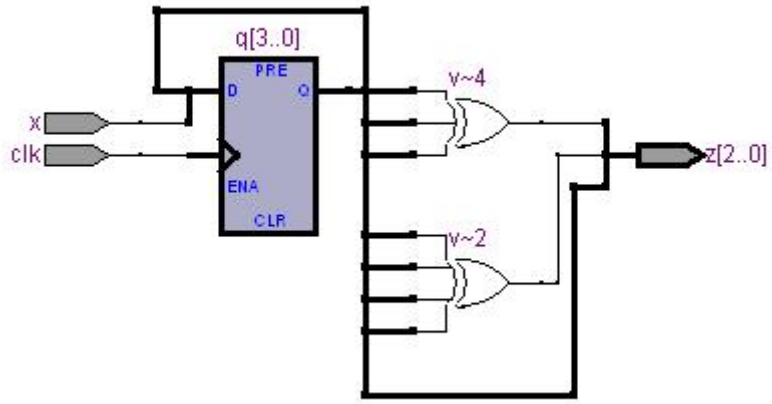


Figura 5.7.: Visualización del flujo de datos del convolucional

5.3.1. Convolucional por máquina de estados

El uso de diagramas de estados en la lógica programable facilita de manera significativa la descripción de un diseño secuencial, ya que no es necesario seguir la metodología tradicional de diseño. En VHDL se puede utilizar un modelo funcional en que sólo se indica la transición que siguen los estados y las condiciones que controlarán el proceso.

Existen una variedad de casos donde tendremos distintas combinaciones de partes síncronas con partes combinacionales, así mismo puramente combinacionales. De modo que en ocasiones resulta conveniente establecer cada parte en distintos procesos.

En VHDL es posible declarar tipos de señales propias, para el caso del diseño de una maquina de estados[6], puede hacerse a partir de establecer dos procesos:

1. Donde se describe únicamente la transición de estados.
2. Donde se describe únicamente la asignación de valores de salida.

En algunos diseños de máquina de estado, donde se dé como entrada la señal de reloj, resulta conveniente hacer la máquina de estados síncrona con éste reloj, para evitar problemas de metaestabilidad [7] con las entradas, además de que algunas herramientas de síntesis entregaran resultados más óptimos al poder interpretar mejor la máquina de estados.[Pardo Carpio]

Sea el caso de un codificador convolucional $R = \frac{1}{2}$, dado por:

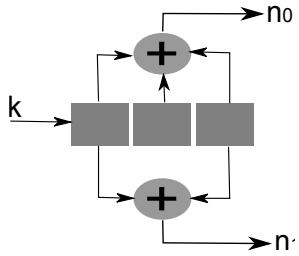


Figura 5.8.: Convolucional, no sistematico, no recursivo

En la tabla de palabras código válidas, para éste codificador apreciamos un comportamiento repetitivo y una cantidad limitada de combinaciones, que nos posibilita usar el método de una maquina de estados.

[6]Del inglés; Algotithmic State Machine, ASM
[7]Estado débilmente estable.

			Salida	
x0	x1	x2	n0	n1
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

En éste caso se empleara el diseño de maquinas de estados, ya explicado en la parte 4.2.1

Estado actual		Estado siguiente, si la entrada es 0	
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	0

Estado actual		Estado siguiente, si la entrada es 1	
0	0	1	1
0	1	0	1
1	0	0	0
1	1	1	0

Nombrando a los estados

a=00

b=01

c=10

d=11

entonces tenemos

Estado actual	Estado siguiente, si la entrada es 0
a	a
b	c
c	a
d	c

Estado actual	Estado siguiente, si la entrada es 1
a	d
b	b
c	a
d	c

Podemos observar como la maquina depende del estado actual y de la entrada, por lo cual podemos realizar la descripción en VHDL, de la siguiente forma.

```
library ieee;
use ieee.std_logic_1164.all;
entity ConvEdos is
port(x,clk: in bit;
        z: out bit_vector(1 downto 0));
end entity;

architecture ConvolXestados of ConvEdos is
type estado is (a,b,c,d);
signal y: bit_vector(1 downto 0);
signal presente: estado;
begin
process(clk)
begin
if clk'event and clk='1' then
case presente is
when a => if x='1' then presente <= b;
                        else presente <= a;
                        end if;
when b => if x='1' then presente <= d;
                        else presente <= c;
                        end if;
when c => if x='1' then presente <= b;
                        else presente <= a;
                        end if;
when d => if x='0' then presente <= c;
                        else presente <= d;
                        end if;
end case;
end if;
end process;

process(presente)
begin
case presente is
when a => y(1)<='0'; y(0)<='0';
when b => y(1)<='0'; y(0)<='1';
when c => y(1)<='1'; y(0)<='0';
when d => y(1)<='1'; y(0)<='1';
end case;
end process;
z<=y;
end architecture;
```

Mediante el simulador de onda de altera verificamos el resultado de la implementación, de la máquina de estados.

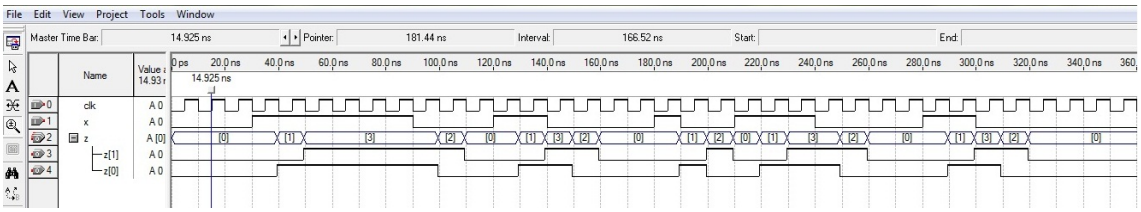


Figura 5.9.: Simulación del código

Dentro de las herramientas que nos ofrece el simulador de altera, tenemos la opción de visualizar nuestro modelo de la máquina de estados, facilitando la comprensión de éste.

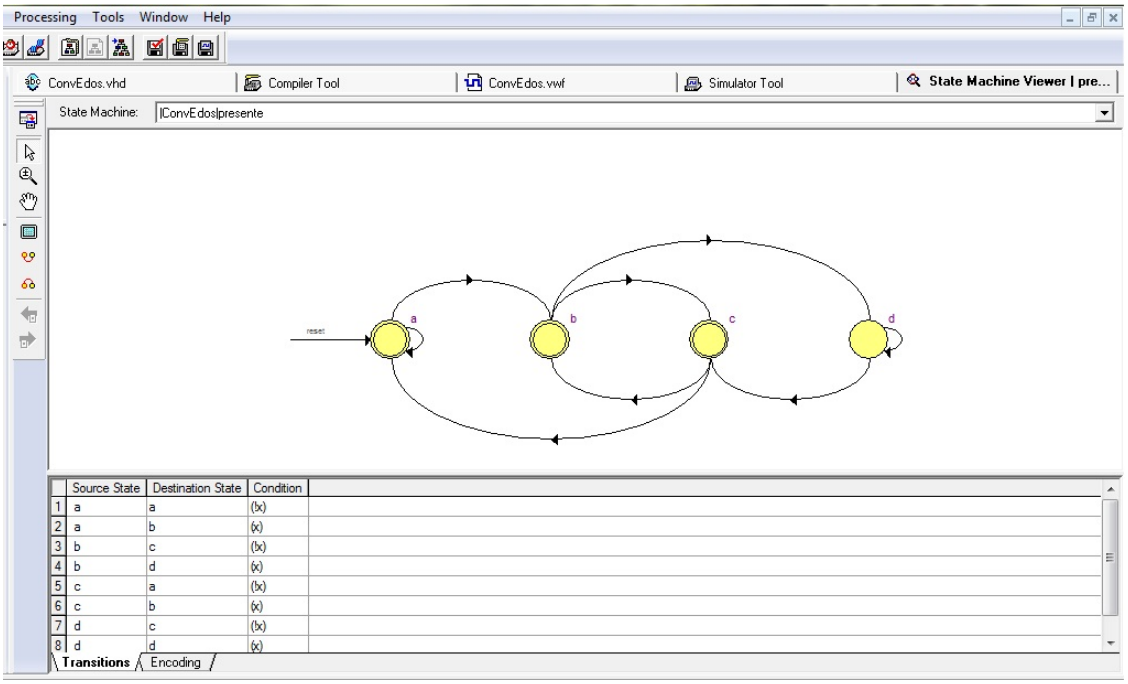


Figura 5.10.: Convolucional, maquina de estados, simulador de altera

La tabla en la parte inferior de la imagen nos permite observar las condiciones de la máquina de estados, el origen y fin de la maquina combinacional de acuerdo a la condición de entrada, las cuales para este caso cumplen con las condiciones de la tabla de verdad inicialmente presentada, verificando de ésta otra forma la correcta implementación.

6. Conclusiones

Podemos resumir la metodología descendente y modular que se conformó esta tesis, observando el desarrollo que en ésta se llevó a cabo;

Se analizó el problema en forma descendente desde los sistemas digitales, sus implicaciones y efectos, hacia los modelos correspondientes para posteriormente segmentarlos en los bloques, seleccionando de entre estos bloques al codificador de canal, ubicado en el emisor, donde se fundamentaron los algoritmos de codificación de canal, extrayendo las bases y fundamentos de tales algoritmos, con lo cual finalmente se implementaron estos fundamentos en VHDL a un bajo nivel, apreciando no solo la conformación del algoritmo, sino también la estructura modular que algunos presentaban.

De la parte teórica:

Al presentar los algoritmos se extrajeron principios y argumentos que comparten entre sí, e incluso notar algunas ventajas de algunos algoritmos sobre otros, como el codificador convolucional que acepta secuencias continuas de bits, a diferencia de los codificadores a bloques, lo que resulta generalmente en una secuencia de bits codificados a una tasa más alta.

Para realizar la correspondiente simulación, se eligieron aquellos algoritmos cuyo argumento o partes de él, presentaran fundamentos en la conformación de terceros códigos, para lo cual se realizaron las siguientes observaciones:

- Los parámetros de peso y bits de paridad son fácilmente observados en los códigos de Hamming, aunque todo codificador necesita medir los bits de control o paridad que lo conforman, así como el peso de las palabras código válidas.
- El fundamento de los turbo códigos se basa en entrelazar codificadores, principalmente se usan los convolucionales.
- Las formaciones más complejas de codificación, como la codificación cíclica, redundancia cíclica y codificadores de baja densidad de paridad LDPC, se basan en operaciones de división y multiplicación de polinomios, que son usados en el algoritmo de codificador cíclico sistemático.

Las formas de trascender de los fundamentos y argumentos de los algoritmos codificadores de canal algunas veces resultan fácilmente apreciables, como los ejemplos anteriores donde se ve directamente la aplicación de estos fundamentos, sin embargo existen algoritmos que usan estos fundamentos mediante su combinación, complicando la visualización de éstos, como se vio por ejemplo; algunos algoritmos al igual

que el convolucional, se basan en la concatenación y/o entrelazado de codificadores, e incluso con el agregado de procesos de pseudoaleatorización, como el codificador RA. En algunos códigos se aprecia que solo son complementados, o usan argumentos de comprobación, como argumentos de codificación, por ejemplo; los códigos extendidos, como hamming extendido, agregan un bit adicional que comprueba la paridad de los bits de comprobación, y los códigos duales usan la matriz de comprobación de paridad H de uno codificador, como su propia matriz generadora.

De la implementación:

Los IP-Core son módulos completamente funcionales, de un circuito electrónico, generalmente una porción de un proceso completo u otro circuito integrado muy complejo, que al ser adquiridos debe realizarse mediante el pago de licenciamiento de propiedad intelectual o bien pueden ser usados libremente mediante el propio desarrollo del bloque en un lenguaje de uso libre, en este caso al realizarse un completo análisis y llevado a implementación en VHDL, pueden usarse estos bloques desarrollados en la presente tesis, como bibliotecas para uso dentro de un sistema o fundamentos de un IP-Core.

Los módulos programados permiten ser usados en aplicaciones de sistemas de comunicación, ofreciendo un alto nivel de paralelismo en el procesamiento a través de hardware, significando reducción del tiempo de procesamiento. Los diseños realizados al ser programados en VHDL resultaron multiplataforma, implicando que no dependen del tipo de dispositivo o la tecnología de fabricación de estos, comprobado al obtener resultados muy similares implementando los mismos diseños en distintos dispositivos de Altera. por lo que a pesar de haberse sintetizados e implementados en un simulador y dispositivos de Altera, en realidad pueden ser implementados en cualquier dispositivo reprogramable reconfigurable, ya sea un CPLD, FPGA, etc. a reserva del tiempo de respuesta de cada dispositivo.

Se observó el diseño modular en la descripción de los componentes, donde los algoritmos pueden por si mismos comprender solo módulos de funciones matemáticas, lo cual resulta apropiado para aplicar códigos correctores de errores simples como concatenados, dando posibilidad de ampliar y optimizar cada parte de un algoritmo, mientras este pueda por si mismo funcionar como un módulo independiente.

El codificador de canal trabaja en darle robustez a una cadena de bits, por lo que este bloque observa únicamente una secuencia de bits de entrada sin importar la fuente o el tipo de información que estos bits conllevan, por tanto cualquier cadena de bits puede ser mejorada para su envió a través del canal, de modo que un codificador puede encontrarse en cascada con otro ya sea de forma directa o entrelazadas. Esta forma nos permite interconectar directa o indirectamente distintos módulos de codificadores en un solo diseño, es decir realizar esquemas concatenados mediante VHDL, lo que es entrelazar esquemas de codificación bajo un mismo dispositivo reprogramable reconfigurable, el cual al ser sintetizado en un CPLD o FPGA, permite diseños “sobre Chip” donde múltiples módulos son combinados en un solo circuito integrado, como pudiera ser el codificador externo e interno. Esta forma de integrar

distintos procesos de un proceso completo u otro circuito integrado muy complejo, es una actual tendencia en diseños en VHDL, por lo que incluso se pueden incluir otros procesos del emisor dentro del mismo diseño, tales como el bloque de encriptamiento que también trabaja con vectores de bits.

Dispositivos y herramientas

También se observó la potencia de algunas herramientas dentro del simulador que permiten realizar la conversión del VHDL a diagramas esquemáticos, visualizar el flujo de datos que ocurre en el diseño, e incluso modelar una maquina de estados.

Por otra parte se observaron las ventajas del uso de diseño por hardware, de entre las cuales destacaron:

- Mayores prestaciones del los FPGA's frente a otros dispositivos
  - Relación de velocidad de procesamiento.
- Uso estándar, no patentado, independiente de la tecnología de implementación y también de proveedores

Parte I.

Anexos

A. Álgebra moderna

Para la comprensión adecuada del trabajo es necesario tener un buen conocimiento del algebra moderna, en especial de Algebra Booleana[1], Se considera el lector tiene los conocimientos adecuados para ello, por lo que solo se anexaran las generalidades, sin necesidad de su profundización.

- Operacion binaria: Es una regla que asigna a cada par ordenado de elementos de un conjunto S, un único elemento de dicho conjunto.

Propiedades

Cerradura: Una operación binaria definida en un conjunto S, asigna siempre como resultado un elemento de S.

sea $\forall a, b \exists S$

entonces $a * b \exists S$

$\therefore$ El conjunto S es cerrado bajo la operación $*$

Subconjuntos: Sea $*$ una operacion binaria en S. *Si T es un subconjunto de S,* se dice que T es cerrado respecto a la operación $*$ si;

$\forall a, b \in T$

$\therefore a * b \in T$

Es decir, si al aplicar dicha operación binaria a dos elementos cualquiera de T, se obtiene como resultado otro elemento de T.

A.1. Espacios Vectoriales

Sea μ_n espacio vectorial[Nakos, George], V es un conjunto de objetos, llamados vectores, junto con dos operaciones llamadas suma y multiplicación por un escalar, que satisfacen las siguientes diez axiomas;

Si $\underset{u}{\overline{\ }}$, $\underset{v}{\overline{\ }}$, $\underset{w}{\overline{\ }}$ se encuentran en V, $\alpha, \beta \in k$ donde $(k, +, \cdot)$ es un campo

1. $\underset{u}{\overline{\ }} + \underset{v}{\overline{\ }} = \underset{v}{\overline{\ }} + \underset{u}{\overline{\ }}$ ésta en V cerrado bajo la adicción

[1]Por lo que acerca de ésta no se hará mención, ni aun en estos anexos, pero puede consultarse el libro: Diseño Digital de M. Morris Mano

2. $\bar{u} + \bar{v} = \bar{v} + \bar{u}$ Conmutatividad
3. $\left(\bar{u} + \bar{v}\right) + \bar{w} = \bar{v} + \left(\bar{u} + \bar{w}\right)$ Asocciatividad
4. $\exists$ un elemento $\bar{0} \in$N denominado Vector cero, tal que $\bar{u} + \bar{0} = \bar{u}$, $\forall u \in V$
5. Para cada $\bar{u} \in V$, $\exists$ un elemento $\bar{u} \in V$ tal que $\bar{u} + \left(\bar{-u}\right) = \bar{0}$
6. Para cada $\alpha\ \bar{u}$ ésta en V, cerradura bajo la multiplicación escalar
7. $\alpha\left(\bar{u} + \bar{v}\right) = \alpha\ \bar{v} + \alpha\ \bar{u}$ distriutiva
8. $\alpha\left(\beta\ \bar{u}\right) = (\alpha\beta)\ \bar{u}$
9. $(\alpha + \beta)\ \bar{u} = \alpha\ \bar{u} + \alpha\ \bar{u}$
10. $1 \cdot \bar{u} = \bar{u}$

NOTA: Los escalares α, β pueden ser elegidos de cualquier sistema númerico con estructura de campo.

A.2. Subespacio vectorial

Un subconjunto w de un espacio vectorial[Nakos, George] se denomina; subespacio de V. Si w mismo es un espacio vectorial con los mismo escalares, adición y multiplicación por escalares que V.

Téorema

Un subconjunto no vacio W, de un espacio vectorial de V es un subespacio de V si se cumplen las 2 reglas de cerradura:

1. $\bar{u} + \bar{v} \in W$

   $a)$ $\bar{u}$, $\bar{v} \in W$

2. $\alpha\ \bar{u} \in W$

Los axiomas para espacio vectorial se cumplen.

A.2.0.1. Vectores subespaciales

Un subconjunto S de un vector espacial V_n es llamado un sub espacio si cumple con las siguientes condiciones:

1. Todos los ceros del vector éstan en S
2. La suma de cualesquiera dos vectores en S, debe tener un resultado también en S (propiedad de cerradura)

Éstas propiedades son fundamentales para la caracterización de un código de bloque lineal. Suponiendo que V_i y V_j son dos palabras códigoso ó dos vectores códigos en un (n,k) código de bloque lineal, si y solo si $V_i \oplus V_j$ es también un vector del código.

A.3. Campos de Galois o campos finitos

Estos pertenecen a una área especializada de la matemática y tienen la propiedad de que las operaciones aritméticas sobre elementos del campo siempre tienen un resultado en el campo, las propiedades de estos campos son la base de varios algoritmos en el área de corrección de errores y procesamiento digital de señales. Sin embargo al ser tan amplia y extensa ésta área, quedan los detalles particulares fuera de la intensión de este trabajo, de modo que se acotara la explicación respectiva, refiriéndose a estos algoritmos de un modo más general, definiendo unicamente los correspondientes a los campos binarios con sus correspondientes operaciones.

Campo 1

+	0	1
0	0	1
1	1	0

Campo 2

•	0	1
0	0	0
1	0	1

Podemos observar como estos resultan ser cuerpos conmutativos.

Para hacer referencia a estos campos de galois binarios se suele usar la nomenclatura GF(2).

De forma silimilar por ejemplo el campo GF(5) está dado por:

+	1	2	3	4	5
1	0	1	2	3	4
2	1	2	3	4	0
3	2	3	4	0	1
4	3	4	0	1	2
5	4	0	1	2	3

•	1	2	3	4	5
1	0	0	0	0	0
2	0	1	2	3	4
3	0	2	4	1	3
4	0	3	1	4	2
5	0	4	3	2	1

De forma general se pueden construir campos finitos en aritmetrica modulo-q[Proakis, John].
[2]

[2]Para profundizar en el tema, es necesario realizar una revisión profunda de aritmética modular, la cual no es tratada en ninguna forma en el presente trabajo.

B. Indicadores

B.1. Interferencia intersimbólica

Ésta hace referencia a la interferencia entre símbolos que podrían ser;

- En un sistema de comunicaciones digitales, la distorsion de la señal recibida, donde el receptor no puede distinguir correctamente entre los cambios de estado de la señal.
- Interferencia energetica entre señales en los intervalos de modulación.

La idea del trabajo presente no referencia como tal a este tipo de errores, debido a que para trabajarlo, existen otro tipo de métodos, (como la reducción de tasa de Baudios, etc.) los cuales no se abordan en el presente.

B.2. $\frac{E_b}{N_0}$

Mejor conocida como "relació señal a ruido (SNR) por bit", es una medida de la relació señal a ruido normalizada, utilizada al comparar el Indice de error por bit (BER) de distintos esquemas digitales sin tener en cuenta el ancho de banda.

Ésta eficiencia espectral es la tasa binaria "bruta" dividida por el ancho de banda y se mide en $(\frac{\frac{bits}{s}}{hz})$. la tasa binaria "bruta" hace referencia a la cantidad de bits transmitidos, incluyendo la redundancia para la corrección de errores FEC y las cabeceras de protocolos. La Eb/N0 se usa habitualmente cuando se desea ahorrar la máxima potencia pero se dispone de una cantidad de ancho de banda arbitrariamente elevada, como por ejemplo en técnicas de espectro ensanchado. Está optimizada para utilizar grandes anchos de banda respecto a la tasa binaria.

Los factores fundamentales que controlan el índice y la calidad de la transmisión de información son el ancho de banda B y la potencia S de la señal

En la Relación Señal a Ruido (S/N) La potencia S de la señal desempeña un papel dual en la transmisión de información. Primero, S esta relacionada con la calidad de la transmisión. Al incrementarse S, la potencia de la señal, se reduce el efecto del ruido de canal, y la información se recibe con mayor exactitud, o con menos incertidumbre. Una mayor relación de señal a ruido S/N permite también la transmisión a través de una distancia mayor. En cualquier caso, una cierta S/N mínima es necesaria para la comunicación.

B.3. Información y entropía

Intuitivamente podemos decir que un evento poco frecuente contiene mucha información y un evento frecuente, poca. Para evitar la subjetividad que entraña una definición de información como esta, es necesario recurrir a la teoría de la probabilidad y definir una medida objetiva de la información sobre una variable aleatoria, esta medida es la entropía.

B.4. Probabilidad de error

Para que un sistema sea razonablemente confiable, la probabilidad de que ocurra un error debe ser pequeña: $P_{1e} = 10^{-6}$

y la probabilidad de un error doble: $P_{1e} = (10^{-6})^2$

En general la probabilidad de error puede calcularse mediante: $P_e = \frac{n!}{m!(n-m)!}$

Donde:

m es la longitud del vector código

n son los bits de redundancia

C. Otros procesos

C.1. Modulación

Una vez codificada la señal, ésta, entra a un "bloque modulador", él cual procesa la señal para ser transmitida sobre el canal de comunicación correspondiente.

La modulación es en si la variación de uno o más parámetros de una señal "portadora", debido a otra señal "moduladora".

En el caso de las señales digitales la forma de representar estas modulaciones es mediante un plano complejo donde puedan ser visualizadas, a este plano se le conoce como diagrama de constelación, donde los ejes real e imaginario suelen ser llamados I (In-phase) y Q (cuadrature), los puntos en la constelación representan símbolos de modulación las "palabras" que podrán usarse en un intercambio de información

PSK (Phase Shift Keying): La modulación por desplazamiento de fase consiste en hacer variar la fase de la portadora entre un número de valores discretos.

QAM (Quadrature Amplitude Modulation): La modulación de amplitud en cuadratura combina la modulación por desplazamiendo de fase con la modulación por variación de amplitud, favorece el aprovechamiento del ancho de banda disponible.

C.2. Pucturing

El proceso de pucturing es una eliminación sistemática de bits de la secuencia de salida de un codificador de baja velocidad con el fin de reducir la cantidad de datos a ser transmitidos, dándole de este modo mayor velocidad al código. Los bits son borrados de acuerdo a una matriz en la cual "zero" significa descartar bit.

D. Dispositivos y VHDL

D.1. VHDL

HDL (Hardware Description Language) es un lenguaje para modelar sistemas electrónicos mientras que VHDL (Very High Speed Integrated Circuit) representa la combinación de VHSIC y HDL donde VHSIC (Very High Speed Integrated Circuit) es la implementación de Circuitos Integrado de Muy Alta Velocidad.

Este tipo de lenguajes difiere de los lenguajes de programación convencionales, pues la ejecución de las sentencias no es estrictamente lineal, algunos casos como los diseños en VHDL y Verilog pueden consistir en una jerarquía de módulos, los cuales son definidos con conjuntos de puertos de entrada, salida y bidireccionale. Las sentencias concurrentes y secuenciales definen el comportamiento del módulo, describiendo las relaciones entre los puertos, cables y registros. Así mismo un módulo puede contener una o más instancias de otro módulo para definir un sub-comportamiento.

VHDL describe estructura El VHDL puede ser usado como lenguaje de Netlist normal y corriente donde se especifica por un lado los componentes del sistema y por otro sus interconexiones.

VHDL describe comportamiento Es posible utilizar el diseño comportamental, del proceso de codificación, mediante la descripción de su funcionalidad, sin necesidad de conocer la estructura interna de un circuito

VHDL, Flujo de Datos Algunas veces resulta importante describir el circuito en forma más cercana a una posible realización física del mismo, este estilo de descripción se encuentra entre una descripción estructural y una descripción completamente abstracta, esta descripción todas las instrucciones son de asignación, de ahí el nombre de flujo de datos.

D.1.1. Herramientas

Altera es una compañia que pone a disposicion varias herramientas para el diseño de FPGAs y CPLDs, las cuales son;

- Quartus II Web Edition: Es una de las herramientas más nuevas de Altera para el diseño de FPGAs. El objetivo de esta herramienta es hacer frente a los dispositivos más complejos que han ido apareciendo en los últimos años. Estas FPGAs pueden incluir tal cantidad de componentes que es necesario disponer de una herramienta específica para ellos. Esta edición web es una versión de Quartus II con algunas limitaciones.
- SOPC Builder: Es una herramienta integrada en Quartus para el desarrollo de sistemas completos en el chip. Incluye compiladores C/C++ para los micros Nios y Excalibur.
- DSP Builder: Versión de evaluación del interface entre Matlab y Simulink de MathWorks, y Altera. Permite sintetizar descripciones en Matlab. Esta versión permite simular las descripciones pero no la síntesis.
- MAX+PLUS II Baseline: Herramienta completa de diseño de circuitos de lógica programable (PLDs) que incluyen varias entradas en diferentes lenguajes (entre ellos VHDL), captura de esquemas, diferentes tipos de simulación, etc. Es la primera herramienta que produjo Altera para sus dispositivos. En la actualidad es la herramienta más aconsejada para dispositivos CPLD de 5V; para dispositivos más complejos y familias de FPGAs se aconseja el Quartus II. Dependiendo del tipo de licencia que se descargue. Se aconseja la licencia de estudiantes pues es la única gratuita que permite la entrada de diseño en VHDL.

Altera no es la única compañía que distribuye software para síntesis de estos dispositivos, algunas otras ofrecen software de síntesis completo o en evaluación, por ejemplo[Pardo Carpio]:

Página Web	Compañia	Alcance
http://www.altera.com	ALTERA	PLDs y CPLDs de Altera
http://cypress.com	Cypress Semiconductors	PLDs y FPGAs de Cypress
http://www.xilinx.com/	Xilinx Inc.	FPGAs y PLD de Xilinx
http://www.actel.com	Actel	FPGAs y PLD de Actel
http://www.synopsys.com	Synopsys	Varios dispositivos
http://model.com	Model Technology	Simulación y modelado

D.2. Dispositivos

Los dispositivos lógicos programables (o PLD, por sus siglas en inglés) favorecen la integración de aplicaciones y desarrollos lógicos mediante el empaquetamiento de soluciones en un circuito integrado. El resultado es la reducción de espacio físico dentro de la aplicación; es decir, se trata de dispositivos fabricados y revisados que se pueden personalizar desde el exterior mediante diversas técnicas de programación. El diseño se basa en bibliotecas y mecanismos específicos de mapeado de funciones,

mientras que su implementación solamente requiere una fase de programación del dispositivo que el diseñador puede realizar en poco tiempo.

El hablar de dispositivos reprogramables reconfigurables, generalmente se usa para describir cualquier circuito principalmente FPGA (Field Programmable Gate Array), PLD (Programable Logic Device - Dispositivo Lógico Programable), y ASIC.

D.2.1. FPGA's

Cada chip de FPGA está hecho de un número limitado de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable y bloques de E/S para permitir que los circuitos tengan acceso al mundo exterior.

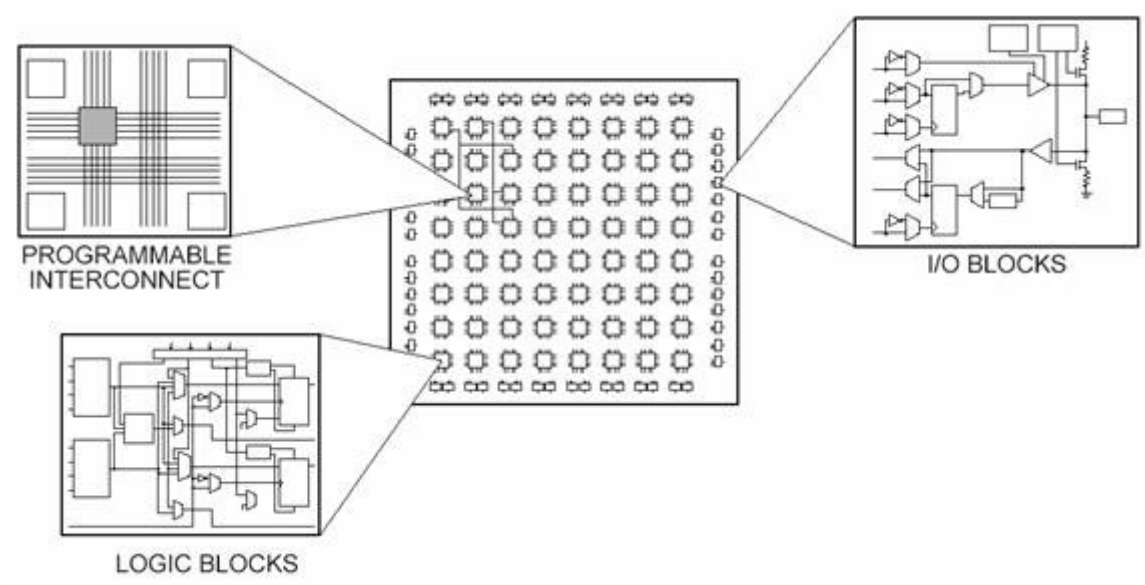


Figura D.1.: Estructura de un FPGA

Las especificaciones de recursos de FPGA a menudo incluyen el número de bloques de lógica configurables, número de bloques de lógica de función fijos como multiplicadores y el tamaño de los recursos de memoria como RAM en bloques embebidos. De las muchas partes del chip FPGA, estos son generalmente los más importantes cuando se seleccionan y comparan FPGAs para una aplicación en particular. Los bloques de lógica configurables (CLBs) son la unidad de lógica básica de un FPGA. Algunas veces referido como segmentos o células de lógica, los CLBs están hechos de dos componentes básicos: flip-flops y tablas de consulta (LUTs). Es importante tomar esto en cuenta porque distintas familias de FPGAs se diferencian en la manera en que los flip-flops y las LUTs están empacados.[1]

[1]Para consulta más detallada consultar la referencia bibliografica de [Pardo Carpio] o bien una

Los terminales de entrada y salida del FPGA usan celdas especiales de E/S que son diferentes de las celdas de elementos lógicos. Además tienen un esquema de interconexión programable que permite la conexión entre las celdas de elementos lógicos entre sí, y con las celdas de E/S. La programación de las interconexiones y de los elementos lógicos puede o no ser permanente, eso depende de la tecnología de programación usada.

Si bien para la configuración de un FPGA particular se usa un software especifico del fabricante del FPGA, la tendencia actual es tratar de realizar el diseño digital, en un diagrama esquemático o en Lenguaje de Descripción de Hardware (Hardware Description Language, HDL), con la máxima abstracción del FPGA a usar. De modo que si por una razón u otra es necesario cambiar de fabricante de FPGA, sea posible una transición lo más fácil posible. Esto se le llama diseño transportable.

Los FPGAs son dispositivos orientados a satisfacer una muy amplia gama de aplicaciones, desde simple lógica combinacional hasta sistemas con microprocesador embebido, transmisión tipo Ethernet, transmisión de datos series a 3.5Gb/s, todo con el mismo dispositivo. Por ello los FPGAs tienen características diversas, pero se podría decir que las principales son las siguientes:

- Gran cantidad de terminales de E/S.
- Desde 100 hasta unos 1400 terminales de E/S
- Buffers de E/S programables: control de sesgo, control de corriente, configuración del estándar de E/S , pull-up y pull-down configurables
- Gran cantidad de Flips-Flops, los dispositivos mas grandes tienen unos 40.000 FFs
- Gran cantidad de Tablas de Búsqueda (Look-Up Tables), ~100.000
- Bloques de Memoria (BRAM) de doble puerto, puerto simple, de hasta 18Mbits, configurables como RAM, ROM, FIFO y otras configuraciones
- Bloques dedicados de Multiplicación Transceptores para transmisión serie de muy alta velocidad , entre 1.5 a- 10.0Gb/s
- Procesador en hardware embebido, tal como el Power-PC, ARM9
- Procesadores descriptos en software, HDL, tales como el 8051, ARM3
- Controladores de reloj tipo Delay Lock Loop (DLL) y Phase Lock Loop (PLLs) de hasta 550MHz. De 2 a 8 controladores por dispositivo
- Control de impedancia programable por cada terminal de E/S
- Interface DDR/DDR2 SDRAM soportando interfaces de hasta 800 Mb/s
- Interfaz con estándares de E/S tipo diferencial tales como LVDS, SSTL diferencial, etc.

sintesis general del dispositivo, aplicaciones y herramientas, proporcionada por National Instruments en: http://www.ni.com/white-paper/6983/es/

D.3. Prestaciones de un dispositivo:

Se puede definir como la velocidad de funcionamiento.

Es posible ver que los dispositivos reprogramables reconfigurables tienen excelentes prestaciones. Por ejemplo, la arquitectura de los CPLDs hace posible implementar funciones lógicas complejas con pocos niveles de realimentación y, por tanto, pocos retrasos. La arquitectura de las CPLDs hace que se pueda predecir los retrasos de manera sencilla, haciendo fácil la prever las prestaciones del cada diseño, aun antes de implementarlo.

D.4. Arquitecturas híbridas

El problema de la perdida de presentaciones con las FPGAs al implementar sistemas complejos ha sido abordado por los distintos fabricantes de PLDs. La nueva filosofía de diseño consiste en realizar arquitecturas híbridas que intentan combinar los beneficios de ambas arquitecturas. Se procura conservar la velocidad de las CPLDs mediante una nueva arquitectura de conexionado, agrupando las celdas elementales en estructuras más grandes. Por otra parte, se intenta mantener la granularidad de las FPGAs difundiendo un gran número de estos bloques más grandes entre las líneas de interconexión.

D.5. Mega-estructuras

A partir de las arquitecturas híbridas, y como consecuencia de la alta escala de integración que se está consiguiendo con la tecnología SRAM, se están desarrollando dispositivos programables más complejos.

Un ejemplo son las mega-estructuras desarrolladas por Altera con su familia APEX20K que son un paso más allá de la complejidad de los dispositivos programables. La alta escala de integración permite agrupar los bloques lógicos de las arquitecturas híbridas en un nivel jerárquico superior llamado mega bloque. La interconexión de los bloques se realiza por pistas rápidas de conexión, de la misma manera que la interconexión, de la misma manera que la interconexión entre mega bloques.

Esta agrupación de bloques lógicos (16 en la familia APEX) con pistas rápidas permite la aparición de buses con altas prestaciones. La alta escala de integración permite en estas familias llegar a:

- 200 megabloques
- más de 2 millones de puertas equivalentes
- más de 40000 elementos lógicos

- medio millón de bits de RAM
- encapsulado de más de 700 pines de entrada/salida

Ademas estos dispositivos son extremadamente flexibles, ya que permiten configurar sus elementos lógicos de diversas maneras, en función de si van a utilizar como bits de RAM o como celdas lógicas.

E. Glosario:

Código: Asociación biunivoca de elementos de un conjunto, ésta asociación se hace asugnando un grupo de simbolos para representar elementos discretos de información, cada tipo de información es una combinación única de simbolos.

Código: Conjunto de todas las posibles palabras código.

Código Binario: Es la asignación de patrones de bits a elementos de información. Los patrones de bits pueden constar de cualquier longitud dependiendo del tamaño de la palabra de código asignada.

Palabra código: vector legal de un código.

Código lineal: Un código es lineal si y solo si para cualquier par de palabras código del mismo, la combinación lineal de las mismas produce otra palabra código. Como corolario, para cualquier código lineal, la palabra código CERO (000.....00) es siempre una palabra código válida.

Código bloque: Un código bloque (n, k) toma "k" bits de la fuente y produce un bloque de tamaño "n". Cada bloque se codifica de forma independiente, sin memoria. Un código bloque lineal se puede representar siempre como la siguiente operación matricial:

$c = b * G$

Donde "c" es la palabra código generada, "b" es el bloque de datos inicial y "G" es la matriz generadora del código.

Código sistemático: Código bloque en el que los primeros "k" bits son idénticos a los de la fuente. Para un código bloque lineal, su matriz generadora "G" se puede dividir en dos componentes $G(I \mid P)$: una matriz identidad "I" y una matriz adicional "P", que representa el código en sí.

Sistema: Entidad que manipula una o más señales para llevar a cabo una funcion, produciendo de este modo nuevas señales.[Haykin]

Señal: Función de una o más variables, que transportan información acerca de la naturaleza de un fenómeno físico.[Haykin]

Sistema combinacional: Sistema cuya salidao salidas, esta en función del estado que guardan las variables de entrada asociadas sin importar el orden en que éstas alcanzaron dichos estados.

Sistema Secuencia: Sistema cuya salida o salidas, está en función del estado que guardan las variables de estado asociadas y el orden en que éstas alcanzaron dichos estados.

Decibel: Es una medida logarítmica de base 10 para relaciones o razones de potencia:

$$dB = 10log\left(\frac{potencia.promedio.de.salida}{potencia.promedio.de.entrada}\right) = 10log\left(\frac{P_{salida}}{P_{entrada}}\right)$$

Distancia Hamming: Se define como el número de bits que cambian entre una palabra código y la otra. La distancia Hamming de un código es la mínima distancia Hamming entre cualesquiera palabras código del mismo. Si el código es lineal, basta con calcular la distancia de todas las palabras código respecto a la palabra código CERO (000...00).

Código perfecto: Aquel para el que toda posible palabra recibida está a distancia "t" o menos de una y solo una palabra código. $t = truncar(\frac{DistanciaHammingmínima-1}{2})$

Matriz de comprobación de paridad: Sea un código bloque lineal. Sabemos que $c = b * G$, con $G = (Ik|P)$. La matriz de comprobación de paridad "H" es $(Pt|In-k)$.

Síndrome: Suponiendo una transmisión ruidosa; $c' = b * G + e$. El síndrome es $S = c' * Ht = b * G * H^t + e * Ht^t$. Operando en módulo 2 (binario), obtenemos $S = e * H^t$. Si no hay errores, el síndrome es el vector CERO. Teniendo una tabla de errores más probables para cada síndrome posible, obtenemos un mecanismo de corrección de errores.

Capacidad detectora; es la capacidad de un código lineal binario de distancia Hamming "d" es capaz de detectar "$d-1$" errores. Esta métrica se usa en la decodificación "dura".

Capacidad correctora Un código lineal binario de distancia Hamming "d" es capaz de corregir "t" errores, con $t = truncar(\frac{DistanciaHammingmínima-1}{2})$. Esta métrica se usa en la decodificación "dura".

Borrado: A veces se pueden identificar datos "borrados" (no presentes, por ejemplo, o con incertidumbre alta). Si somos capaces de identificar borrados, un código puede corregirlos de forma equivalente a su capacidad detectora, que puede ser muy superior a su capacidad correctora.

Código prefijo: Se trata de un código que no tiene un tamaño en bits determinados, sino en el que cada palabra código puede tener un tamaño variable, y en el que es posible determinar la palabra código recibida en cuanto se recibe su último símbolo. En otras palabras, ninguna palabra código es prefijo de otra.

Entropía de una fuente: Es la cantidad de información generada por una fuente. Trabajando en base 2 (binario), la entropía nos indica el número de bits, en media (en momentos puntuales podemos estar por encima o por debajo de ese valor "medio"), que necesitamos para transmitir fielmente los datos

generados por una fuente. Matemáticamente, la entropía de una fuente que genera mensajes $X[i]$ con una probabilidad $P[i]$ independiente y sin memoria, se calcula como:

$entropia = -suma(P[i] * log(P[i]))$La base del logaritmo empleado depende de las unidades que nos interese. Si trabajamos en binarios, usaremos logaritmos en base dos.

Decodificación "dura": La decodificación se realiza bit a bit, de forma independiente.

Decodificación "blanda": La decodificación se realiza a nivel de toda la palabra código de forma simultanea. Esta decodificación proporciona resultados muy superiores, pero no siempre es factible.

Ganancia de codificación: Este valor, típicamente en decibelios (dB), nos indica el margen que hemos ganado con la codificación, para una tasa de error determinada. Es decir, nos indica los decibelios adicionales de ruido que podemos tolerar para obtener la misma tasa de error o, alternativamente, cuánto podemos reducir la potencia de la señal para obtener la misma tasa de error. Para decodificación "blanda" y tasas de error bajas, la ganancia de codificación viene determinada por

$dB = 10 * log10(distanciaHammingmínima * k/n)$

Un código de paridad simple tiene una distancia de Hamming de 2, y $n = k + 1$. A medida que crece el tamaño del bloque, la ganancia de codificación tiende a 3dB (suponiendo ruido gausiano, etc).

Un código Hamming (7,4) tiene una distancia Hamming de 3 y una ganancia de codificación de 2'34dB, si se usa decodificación "blanda". Si usamos decodificación "dura", y la probabilidad de error es 10-5, la ganancia es de solo 0'63dB.

ASIC: Circuito integrado de aplicación específica, o a la medida .

ETSI: European Telecommunications Standards Institute o Instituto Europeo de Normas de Telecomunicaciones.

Metaestabilidad: Propiedad de un sistema con varios estados de equilibrio, donde puede presentarse un estado de equilibrio débilmente estable, durante un considerable espacio de tiempo.

Paridad Par: El bit de paridad se escoge de forma tal que el número de unos en la palabra código incluyendo al bit de paridad sea par.

Paridad Impar: El bit de paridad se escoge de forma tal que el número de unos en la palabra código incluyendo al bit de paridad sea impar.

OFDMA: Orthogonal Frecuency-Division Multiple Access.

IP-CORE: Intellectual Property Core, Bloque funcional complejo de un circuito electrónico, su uso es licenciado a otras compañías por el diseñador original.

éste es generalmente una porción de un proceso completo u otro circuito integrado muy complejo.

Nasdaq 100: Índice bursátil de Estados Unidos que recoge a los 100 valores de las compañías más importantes del sector de la industria las telecomunicaciones, incluyendo empresas de hardware y de software.

S&P 500: Es el índice bursátil más importantes de Estados Unidos, se le considera el índice más representativo de la situación real del mercado.

Registro de corrimiento: Es un modulo lógico que controla las posiciones de bits de datos binarios recorriendo los bits de izquierda a derecha o viceversa.

UL-SCH: Canal de transporte compartido de enlace descendente.

DL-SCH: Canal de transporte compartido de enlace ascendente.

PCH: Canal de transporte de busqueda.

MCH: Canal de transporte de multidifusión.

BCH: Canal de transporte de difusión.

DCI: Información de control de enlace descendente.

HI: Indicador HARQ

UCI: Información de control de enlace ascendente.

SISO: Registros de entrada serie y salida serie.

SIPO: Registros de entrada serie y salida paralelo.

PISO: Registros de entrada paralelo y salida serie.

PIPO: Registro de entrada y salida en paralelo.

Bibliografía

[Leon W. Couch] Leon W. Couch II, "Sistemas de comunicación digital", ed Pearson, 2008.

[Nakos, George] George Nakos, "Algebra lineal con aplicaciones", ed. Thomson.

[Clark y Cain] Clark, G. C. y J. B. Cain, "Error-Correction Coding for Digital Communications", Plenum Publishing Corporation, Nueva York, 1981.

[BHARGAVA] Bhargava, V. K., D. Haccoun, R. Matyas y P. P. Nuspl, "Digital Communications by Satellite", Wiley-Interscience, Nueva York, 1981.

[Proakis, John] John G. Proakis "Digital Communications". 4ta edición.

[Pardo Carpio] Pardo Carpio, Fernando. "VHDL Lenguaje para descripción y modelado de circuitos". Universidad de Valencia, 1997.

[G. Maxines] G. Maxines, David y Alcalá Jessica. "VHDL El arte de programar sistemas digitales". Ed. Continental, México 2002.

[Brown y Vranesic] Brown, Stephen y Vranesic, Zvonko. "Fundamentals of digital logic with VHDL design". Second edition, Ed. Mc Graw Hill.

[Haykin] Simon Haykin. "Señales y sistemas". Ed. Limusa Wiley, México, 2001.

[Arnone Ing.] Ing. Arnone, Leonardo José. Tesis Doctoral en ingeniería. "Transmisión segura en comunicaciones inalámbricas de corto alcance", Universidad del Mar del plata, Facultad de Ingeniería. 2008.

[Monroy Escabosa] Monroy Escabosa, Eduardo. "Introducción a la codificación de canal". Ramas de estudiantes del IEEE. http://upcommons.upc.edu/revistes/bitstream/2099/9512/1/Article007.pdf

[Marcano] Diógenes, Marcano. "Canal Móvil". Pontificia Universidad Católica del Perú, Unidad de posgrado, Consultado Febrero 2013. Web; http://departamento.pucp.edu.pe/ingenieria/images/documentos/seccion_telec

[Granado, José] Granado, José M. "Hardware vs software: el algoritmo criptográfico IDEA implementado mediante FPGAs". Universidad

de Extremadura. Departamento de Informática. Escuela poítecnica, España, 2004, pag. 85.

[ETSI] ETSI, European Telecommunications Standards Institute. "Evolved Universal Terrestrial Radio Access". European Telecommunications Standards Institute, 2008. Web; http://webapp.etsi.org/key/queryform.asp

[García Álvarez] Garcia Álvarez, Julio Cesar. "Códigos de corrección de errores". Universidad Nacional de Colombia, Sede Manizales, 2009. capítulos ii & ii. web; http://www.virtual.unal.edu.co/cursos/sedes/manizales/4040051/html/capit

[Sklar] Sklar, Bernard. "Digital Communications Fundamentals and Applications", second edition . Ed. Prentice Hall, California, 1998.

[Claude E. Shannon] Claude E. Shannon. "Communication in the presence of noise", reimpresión IEEE 1998.

[Freznel] Freznel, Luis E. "Sistemas de comunicaciones", Ed. Alfaomega, Austin Community College.

[Lathi] B. P. Lathi. "Sistemas de comunicaciones". Ed. Mc Graw Hill, California state university, sacramento, 1986.

[Dapaena] Dapaena Janeiro, Adriana "Codificación de canal". Facultad de informática, Universidad de Coruña, Campus de Elviña s/n 15071, A. Coruña.

[Viñé Viñuelas] Viñé Viñuelas, Miguel. "Implementación en VHDL de un decodificador de Viterbi y su integración en un prototipo de un sistema WIMAX ". Escuela Politécnica Superior, Departamento de Tecnología Electrónica, Leganés, junio de 2012.

Mesografia

Monroy Escabosa, Eduardo. “Introducción a la codificación de canal”. Ramas de estudiantes del IEEE. http://upcommons.upc.edu/revistes/bitstream/2099/9512/1/Article007.pdf

Garcia Álvarez, Julio Cesar. “Códigos de corrección de errores”. Universidad Nacional de Colombia, Sede Manizales, 2009. capítulos ii & ii. web;

http://www.virtual.unal.edu.co/cursos/sedes/manizales/4040051/html/capitulos/

http://webapp.etsi.org/key/queryform.asp

http://www.virtual.unal.edu.co/cursos/sedes/manizales/4040051/html/capitulos/

http://departamento.pucp.edu.pe/ingenieria/images/documentos/seccion_telecomunicaciones/Capitulo%201%20Canal%20Movil.pdf