GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES GSProg

JOSE LUIS SALAZAR MONTES

GABRIEL ALEJANDRO TOVAR TORRES

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ, D.C.

2010

GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES GSProg

JOSE LUIS SALAZAR MONTES

GABRIEL ALEJANDRO TOVAR TORRES

Trabajo de grado presentado para optar al título de Ingeniero Electrónico.

Director:

JOSÉ LUÍS URIBE APONTE MSc.
Ingeniero Electrónico

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ, D.C.

2010

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNÍFICO: R.P. JOAQUIN EMILIO SANCHEZ GARCÍA S.J

DECANO ACADÉMICO: Ing. FRANCISCO JAVIER REBOLLEDO MUÑOZ

DECANO DEL MEDIO UNIVERSITARIO: R.P SERGIO BERNAL RESTREPO S.J

DIRECTOR DE CARRERA: Ing. JUAN MANUEL CRUZ BOHÓRQUEZ M, Ed.

DIRECTOR DEL PROYECTO: Ing. JOSÉ LUÍS URIBE APONTE MSc.

NOTA DE ADVERTENCIA

"La Universidad no se hace responsable de los conceptos emitidos por algunos de sus alumnos en los proyectos de grado. Solo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ello el anhelo de buscar la verdad y la justicia."

Artículo 23 de la Resolución No. 13, del 6 de julio de 1946, por la cual se reglamenta lo concerniente a Tesis y Exámenes de Grado en la Pontificia Universidad Javeriana.

AGRADECIMIENTOS

Agradecemos a Dios y a nuestras familias por el apoyo incondicional a lo largo de todos nuestros estudios para la formación como ingenieros.

Al ingeniero José Luís Uribe Aponte por su apoyo incondicional, sus grandes conocimientos, sus consejos, su amistad, su tiempo y su orientación dada para el excelente desarrollo del presente trabajo de grado.

A los Ingenieros Santiago Valencia Convers y Nicolás Uribe por su colaboración, tiempo y sugerencias.

Al grupo de trabajo del laboratorio de electrónica, quienes siempre nos brindaron el mejor servicio.

DEDICATORIAS

Dedico este trabajo de grado a mi padre por su apoyo incondicional y sus concejos en los momentos más difíciles en el trascurso de mis estudios y en especial a mi madre recientemente fallecida a quien amo profundamente y quien estaría muy orgullosa por la culminación de mis estudios universitarios.

GABRIEL ALEJANDRO TOVAR TORRES

Dedico este trabajo de grado a mi madre y a mi padre, ya que gracias a su esfuerzo, dedicación y tiempo, han influido en la culminación de esta etapa importante en mi vida.

JOSE LUIS SALAZAR MONTES

Tabla de contenido

FIGURAS

1. Introducción

En el diseño e implementación de sistemas digitales existe la necesidad de poseer un instrumento que esté en la capacidad de entregar las señales eléctricas adecuadas que con frecuencia son más utilizadas como entradas en estos sistemas. Una vez identificadas las señales que se utilizan más frecuentemente para el diseño e implementación de sistemas digitales, se desarrolló el ***GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES (GSProg)*** como un sistema digital capaz de generar palabras digitales representadas en señales binarias asíncronas que sirvan como entradas a los diversos diseños digitales realizados.

La implementación del generador se hizo mediante una interfaz gráfica y un hardware en un dispositivo programable FPGA. La interfaz gráfica desarrollada en un lenguaje de programación es el medio por el cual se permite al usuario ingresar desde un computador de forma ordenada y sencilla los parámetros de las palabras digitales que serán entregadas como entradas al sistema digital bajo prueba. Estos parámetros son: Tipo de representación de la palabra digital (serial o paralela), número de palabras digitales, número de bits que componen cada palabra digital, contenido de la o las palabras digitales (ingresado por el usuario o generado aleatoriamente por el generador) y envío de éstas, (una vez o múltiples veces).

Una vez son ingresados por medio de la interfaz los anteriores parámetros, un hardware implementado en un dispositivo FPGA, ACEX 1k100 de ALTERA, tiene la función de recibir, clasificar y procesar ésta información para posteriormente generar las palabras digitales, representadas en señales binarias asíncronas, con sus respectivas señales de sincronismo e inicio de nueva palabra. Este generador de manipulación sencilla y práctica permitirá validar el funcionamiento de diferentes sistemas digitales.

En éste documento se muestra y explica el desarrollo teórico que se empleó para la realización de éste generador. Los capítulos en los que está dividido este documento son:

- **Sección 2; Objetivos.** En ésta sección se encuentra el objetivo general y los objetivos específicos de este trabajo de grado.

- **Sección 3; Marco teórico**. En el que se plantan conceptos como comunicación serial asíncrona, protocolo de comunicación RS 232 y método de detección de errores en una transmisión mediante redundancia de paquete.

- **Sección 4; Especificaciones**. En ésta sección se encuentra todas las especificaciones alcanzadas por el ***GSProg*** así como sus especificaciones eléctricas.

- **Sección 5; Desarrollos**. En ésta sección se encuentra el análisis y descripción de todos los bloques, sub-bloques y señales que conforman todo el sistema diseñado.

- **Sección 6**; **Pruebas y análisis de resultados.** En ésta sección se muestra las pruebas realizadas y resultados obtenidos del sistema digital diseñado.

- **Sección 7**; **Conclusiones.** En esta sección se presenta las conclusiones a las que se llegó durante el desarrollo y la culminación de este trabajo de grado.

- **Sección 8**; **Referencias bibliográficas.** Esta sección presenta la bibliografía utilizada para el correcto diseño del sistema.

- **Sección 9**; **Anexos.** Esta sección presenta diseños y cálculos de circuitos combinatorios y secuenciales, descripción AHPL y código de programación de la interfaz gráfica.

2. Objetivos

OBJETIVO GENERAL.

Diseñar e implementar un generador de secuencias de entrada que pueda ser operado mediante un programa de computador para verificar sistemas digitales.

OBJETIVOS ESPECÍFICOS.

1. Determinar los tipos de señales y secuencias más comunes, usadas para verificar sistemas digitales.
2. Especificar los parámetros variables para los diferentes tipos de señales y secuencias usadas para verificar sistemas digitales.
3. Diseñar e implementar un generador de secuencias en un FPGA.
4. Diseñar una interfaz (programa de computador) que permita seleccionar el tipo de salida del generador de secuencias.

3. Marco teórico

3.1. Protocolo de comunicación RS232

Para el desarrollo de este trabajo de grado se utilizó el protocolo de comunicación RS 232 el cual permite transmitir los datos procedentes de la interfaz gráfica a la FPGA a través del puerto serial del computador. Dicho protocolo es una norma o estándar mundial que rige los parámetros de uno de los modos de comunicación serial. Por medio de este protocolo se estandarizan las velocidades de transferencia de datos, la forma de control que utiliza dicha transferencia, los niveles de voltajes utilizados, el tipo de cable permitido, las distancias entre equipos y los conectores utilizados. Además de las líneas de transmisión (Tx) y recepción (Rx), las comunicaciones seriales poseen otras líneas de control de flujo (Hands-hake), donde su uso es opcional dependiendo del dispositivo a conectar.

A nivel de software, la configuración principal que se debe dar a una conexión a través de puertos seriales RS-232 es básicamente la selección de la velocidad en baudios (1200, 2400, 4800, etc.), la verificación de datos o paridad (parida par o paridad impar o sin paridad), los bits de parada luego de cada dato (1 ó 2), y la cantidad de bits por dato (7 ó 8), que se utiliza para cada símbolo o carácter enviado. La norma RS-232 fue definida para conectar un ordenador a un modem. La transmisión de información por medio del

protocolo de comunicación RS232 se realiza en forma serial asíncrono donde la duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos.

Normalmente cuando no se realiza ninguna transferencia de datos, la línea del transmisor se encuentra en estado alto. Para iniciar la transmisión de datos, el transmisor coloca esta línea en bajo durante determinado tiempo, lo cual se le conoce como bit de arranque (Start bit) y a continuación empieza a transmitir con un intervalo de tiempo los bits correspondientes al dato, empezando siempre por el bit menos significativo (LSB), y terminando con el bit más significativo (MSB). Si el receptor no está sincronizado con el transmisor, este desconoce cuándo se van a recibir los datos. Los niveles de tensiones empleados en la norma RS232 están comprendidos entre +15/-15 voltios donde normalmente un 1 lógico es representado por una tensión de 12 V y un 0 lógico es representado por una tensión de -12V. [10]

La siguiente figura muestra la estructura de un carácter que se trasmite siguiendo la norma RS232.

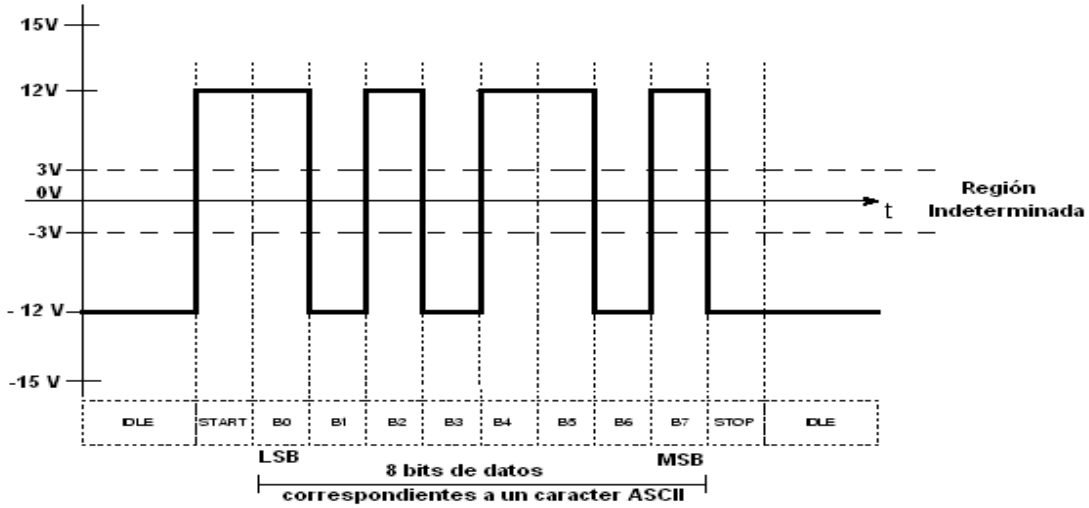


Figura 1: Transmisión de datos siguiendo el protocolo de comunicación RS232.

La importancia de conocer esta norma radica en que los niveles de voltaje que maneja el puerto serial del computador son diferentes a los que utilizan los microcontroladores, FPGAS y los demás circuitos digitales integrados. Por lo tanto se necesita de un dispositivo que haga posible la conversión de niveles de voltaje RS 232 a niveles de voltaje TTL y CMOS donde se asume que un 0 lógico es igual a cero Voltios y un 1 lógico es igual a cinco Voltios, lo cual se realiza mediante el CI MAX 232.

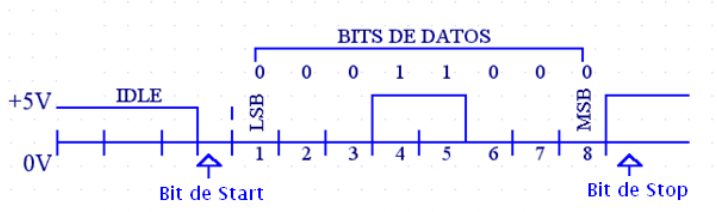


Figura 2: Transmisión de datos en forma serial asíncrona con los niveles de voltaje adecuados que utiliza un FPGA. [8]

3.2. Conector RS232D (Conector DB9 de 9 pines)

El conector RS232D consiste en un conector tipo DB-9 de 9 pines. Cada pin puede ser de entrada o de salida y tiene una función específica. Las más importantes son los que se observan en la figura 3.

SIGNAL	PIN No.
Carrier Detect	1
Receive Data	2
Transmit Data	3
Data Terminal Ready	4
Signal Ground	5
Data Set Ready	6
Request To Send	7
Clear To Send	8
Ring Indicator	9

Figura 3: Disposición de pines del conector RS232D de tipo DB9. [9]

En este trabajo de grado se utilizaron los pines 2,3 y 5 del conector DB9 para la transmisión de datos desde el computador a la FPGA.

- Transmit Data (TD): Es la línea por donde se transmiten los datos bit a bit.
- Receive Data (RD): Es la línea por donde se reciben los datos bit a bit.
- Signal Ground (SG): Es la línea de referencia a tierra, 0 V.

3.3. Método de detección de errores por redundancia de paquete

En éste trabajo de grado el método de detección de errores utilizado fue el de redundancia de paquete que consiste en un esquema de detección de errores sustentado en la transmisión completa de dos copias del mismo paquete de información. Si las dos copias son idénticas cuando lleguen al receptor, es muy probable que la información esté correcta. Si no son iguales, un error fue introducido en una de las copias (o en ambas) y deben ser descartadas o transmitidas nuevamente.

4. Especificaciones

El GSProg es un sistema digital implementado en un FPGA, ACEX 1k100 de ALTERA, encargado de generar una o múltiples veces conjuntos de 1 a 8 palabras digitales programadas o aleatorias entre 1 a 16 bits seriales o paralelas asíncronas con sus respectivas señales de *Sincronismo* e *Inicio de Trama*, donde cada bit que conforma cada una de las palabras digitales a entregar por el generador junto con las señales de *Sincronismo* e *Inicio de Trama* deben cumplir con las siguientes especificaciones.

Para los bits pertenecientes a las palabras digitales seriales y paralelas, se tiene que:

- Duración de cada bit que conforma la palabra digital: Tiempo 3µs.
- Descanso entre bit y bit perteneciente a cada palabra digital en formato serial: Tiempo 2µs.

- Descanso entre palabra digital y palabra digital en formato paralelo: Tiempo 2µs.

Para la señal de *Sincronismo*, se tiene que:

- Duración de la señal de *Sincronismo*: Tiempo 1µs.
- Descanso entre la señales de *Sincronismo*: Tiempo 4µs.
- Ubicación de la señal de sincronismo: En la mitad del tiempo de la señal de dato o bit que conforma la palabra digital.
- El Sincronismo es inmediato o simultaneo; mientras esta el dato, la señal de Sincronismo esta activa.

Para la señal *Inicio de Trama*, se tiene que:

- La señal de *Inicio de Trama* es generada 1 µs antes del inicio de una nueva palabra digital serial o paralela, tiene una duración de 1 µs.

A continuación se ilustran éstas especificaciones:

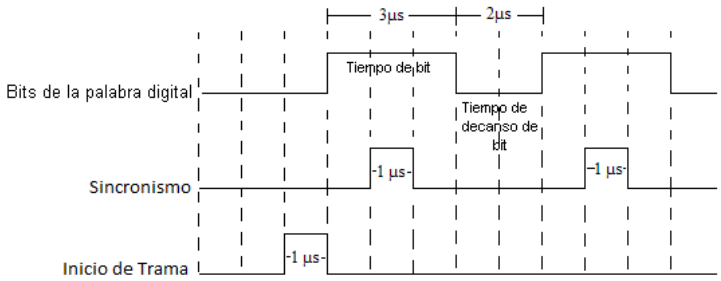


Figura 4: Representación de señales entregadas por el generador con sus especificaciones.

Las anteriores especificaciones que definen a los bits pertenecientes a la palabra digital y a las señales de *Sincronismo* e *Inicio de Trama* están definidas por el diseñador del sistema y no son modificables por parte del usuario.

En las figuras 5 y 6 se ejemplifican 2 palabras digitales de 3 bits en formato serial y paralelo entregadas por *GSProg* con sus respectivos tiempos de bit, *Sincronismo* e *Inicio de Trama*.

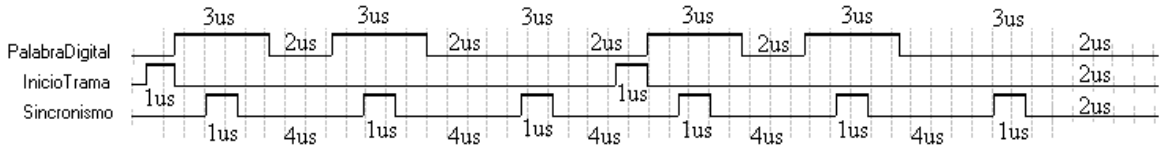


Figura 5: 2 palabras digitales de 3 bits en formato serial entregadas por GSProg correspondientes a 110 y 110.

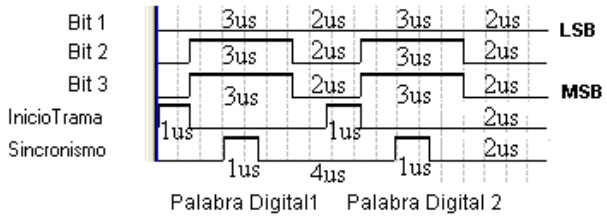


Figura 6: 2 palabras digitales de 3 bits en formato paralelo entregadas por GSProg correspondientes a 110 y 110.

La interfaz gráfica implementada en VISUAL BASIC 6.0 permite al usuario caracterizar desde un computador las palabras digitales que desea obtener a la salida del generador con el fin de que éste tenga la capacidad de:

- Entregar de 1 a 8 palabras digitales seriales o paralelas asíncronas con sus respectivas señales de *Sincronismo* e *Inicio de Trama*, donde todas las palabras generadas son del mismo número de bits.
- Que cada palabra digital generada pueda estar comprendida de 1 a 16 bits según desee el usuario.
- Que los bits que conforman cada palabra digital serial o paralela puedan ser introducidos por el usuario o generados aleatoriamente.
- Que las palabras digitales sean generadas una sola vez, Envío *Único*, o múltiples veces, Envío *Continuo*, según desee el usuario del generador.

En la figura 7 se esquematiza el *GSProg* el cual está conformado por la interfaz gráfica en el computador y el generador en el FPGA. La comunicación entre el computador y el FPGA se realiza a través del protocolo de comunicación serial RS232.

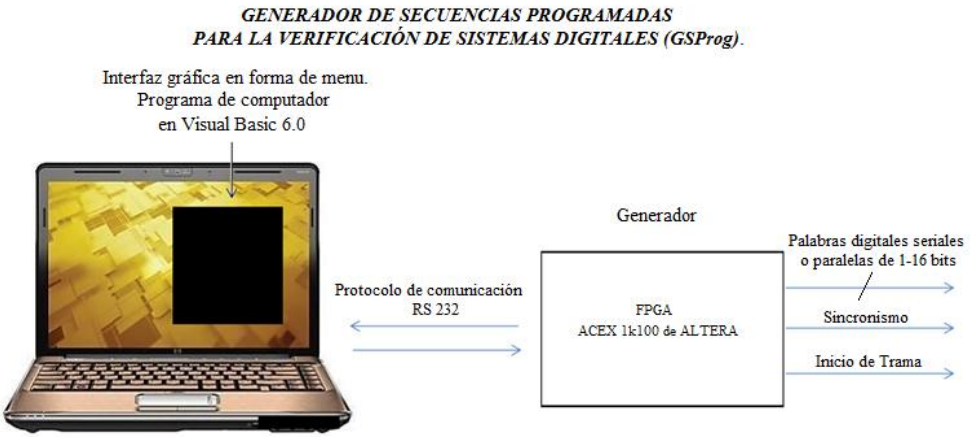


Figura 7: Generador de secuencias programadas para la verificación de sistemas digitales (GSProg).

4.1. Entradas y salidas del sistema hardware

Entradas

- *Rx*: Señal conformada por una trama de 10 bits seriales provenientes del puerto serial del computador siguiendo el protocolo RS232. Esta señal llega a la FPGA con los niveles de voltaje adecuados para que la FPGA la pueda procesar después de pasar por un tranceiver. De los diez bits solo 8 poseen información para ser utilizada, ya que el primer bit es de inicio y el último bit es de parada (Norma RS232). Cada bit perteneciente a *Rx* tiene una duración de 104 us debido a que la velocidad de transmisión de datos del puerto serial se dispuso de 9600 kbps, por tanto $\frac{1}{9600\ \text{kbps}} = 104\ \text{us}$. Esta señal *Rx* es un conjunto o conjuntos de paquetes de 10 bits como se observa a continuación.

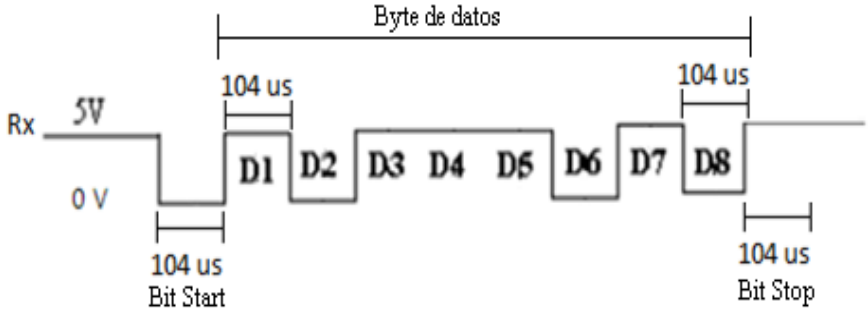


Figura 8: Representación de la señal digital Rx. [7]

- *Clock*: Señal de reloj de 8 MHz con el que opera el (*GSProg*).

- *Reset*: Señal asíncrona externa que reinicia al (*GSProg*).

- *EntradaAlta*: Señal de 5 voltios correspondiente a un 1 lógico que sirve para cargar los registros *Llega Byte*, *Llega Bien* y *Llega Mal* pertenecientes al **Sistema de Comunicación** con los bits que deben ir en 1 para formar los códigos ASCII de los caracteres Ç, M, o B que deben ser enviados al computador para corroborar que llegó un byte (Ç), que un conjunto de bytes han llegado bien (B) o que un conjunto de bytes han llegado mal (M).

- *Ground*: Señal de 0 voltios correspondiente a un 0 lógico que sirve para cargar los registros *Llega Byte*, *Llega Bien* y *Llega Mal* pertenecientes al **Sistema de Comunicación** con los bits que deben ir en 0 para formar los códigos ASCII de los caracteres Ç, M, o B que deben ser enviados al computador para corroborar que llegó un byte (Ç), que un conjunto de bytes han llegado bien (B) o que un conjunto de bytes han llegado mal (M).

Salidas

- *Palabra Digital*: Salida que representa las palabras digitales seriales o paralelas, está conformada por un bus de 16 bits de datos donde cada uno de ellos pertenece a la palabra digital paralela. *Palabra Digital (1)* corresponde al bit LSB mientras que *Palabra Digital (16)* corresponde al bit MSB de la palabra digital paralela.

- Para el caso de las señales seriales, la salida *Palabra Digital (1)* perteneciente al bus de 16 bits está dispuesta para que salga la secuencia de bits que conforman la palabra digital serial.

- *Sincronismo*: Señal auxiliar que indicará:

  ✓ Para el caso de señales paralelas, que la palabra digital está lista para la captura por parte del usuario.

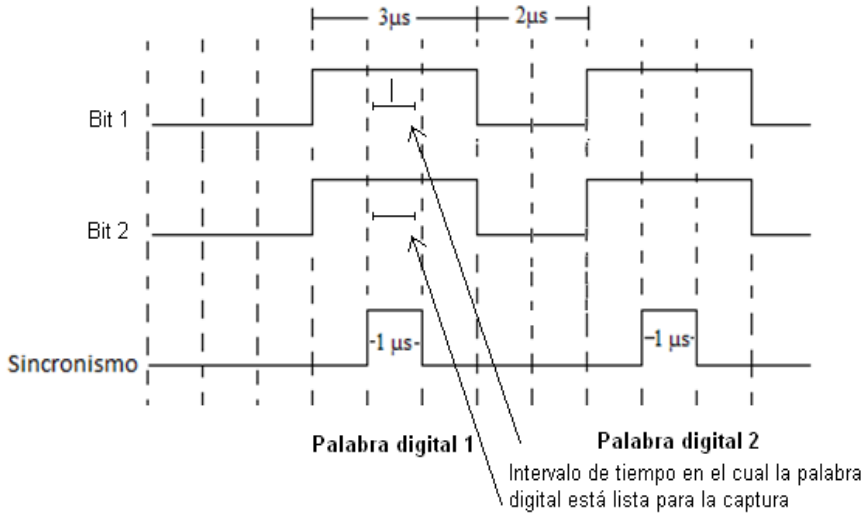


Figura 9: Función de la señal de sincronismo para palabras digitales en formato paralelo.

  ✓ Para el caso de señales seriales, que el bit que conforma la palabra digital está listo para la captura por parte del usuario. Nótese que para el caso de las señales seriales se debe hacer la captura bit a bit de la palabra digital.

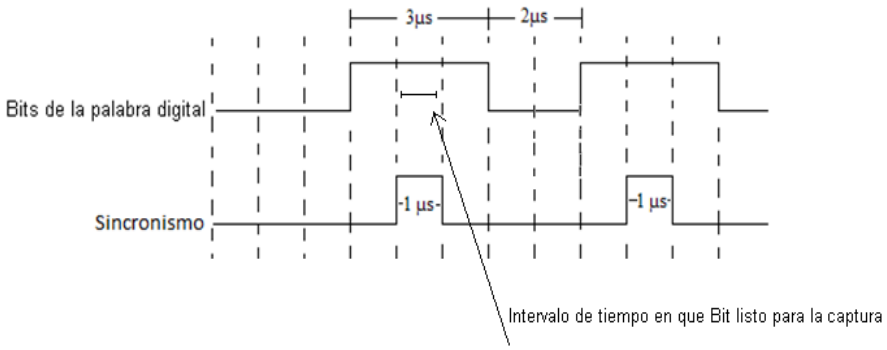


Figura 10: Función de la señal de sincronismo para palabras digitales en formato serial.

- *Inicio de Trama*: Señal que indica el comienzo de una nueva palabra digital serial o paralela.

- *Tx*: Señal de 10 bits en formato RS232 que informa al computador si los datos provenientes de éste han llegado de forma correcta al Hardware.

En la figura 11 se aprecia la disposición de las entradas/salidas del *GSProg*.

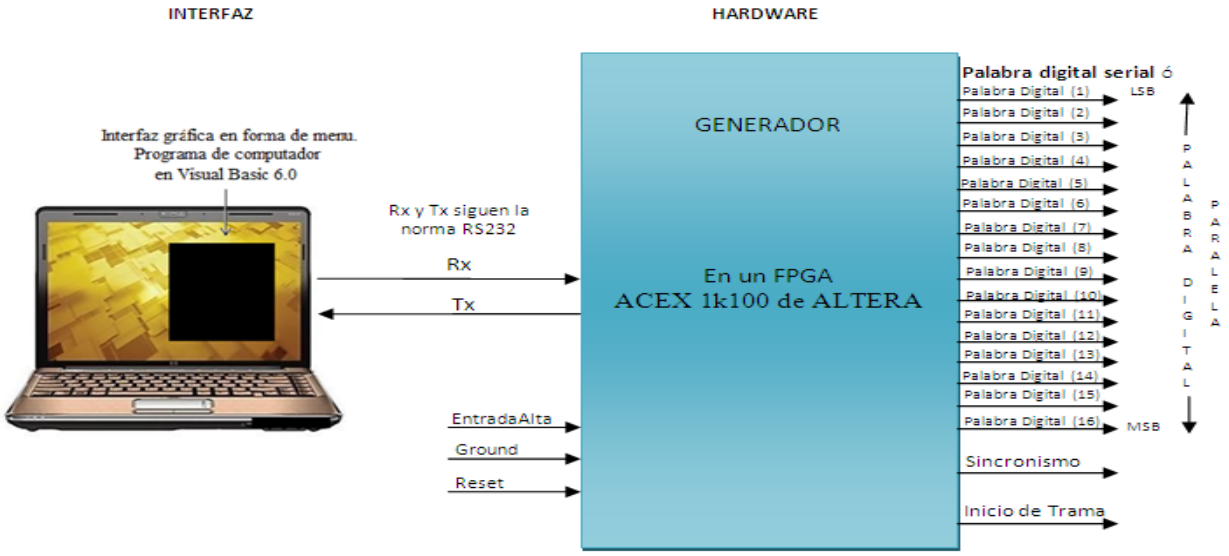


Figura 11: Entradas y salidas del sistema (GSProg).

4.2 Especificaciones técnicas y eléctricas del GSProg

4.2.1 Datos técnicos del GSProg

- Entrega a su salida entre 1 y 16 señales binarias asíncronas.
- Cada señal binaria asíncrona está comprendida entre 1 y 8 bits.
- Entrega 1 señal de *Sincronismo* y 1 señal de *Inicio de Trama*.
- Tiempo de bit de cada señal digital entregada; 1us.
- Duración de la señal de *Sincronismo*; 1 us.
- Duración de la señal de *Inicio de Trama*; 1 us.
- Tiempos de bit, duración de señales de *Sincronismo* e *Inicio de Trama* son fijos y por consiguiente no modificables por el usuario.
- Posee una interfaz gráfica en forma de menú para caracterizar las señales binarias asíncronas que se desean obtener en las salidas del generador.
- La interfaz gráfica debe ser ejecutada en una versión de Visual Basic 6.0.

4.2.2 Especificaciones eléctricas del GSProg

Especificaciones electricas GSProg	
Tensión	
Nivel de tensión de linea:	110 V AC, 60 Hz
Nivel de tensió de entrada:	5 V DC
Nivel de tensión bajo (0 Lógico):	0 a 2.5 V DC.
Nivel de tensión alto (1 Lógico):	3.3 a 5 V DC.
Corriente	
Corriente de salida :	0 a 25 mA
Potencia	
P entrada:	35 mW
Impedancia	
Z in:	0.7 MOhms
Velocidades de Tx y Rx entre computador y FPGA	
Tx:	9600 Kbps, Norma RS232
Rx:	9600 Kbps, Norma RS232
Frecuencia de reloj	
Reloj:	8 MHz

Figura 12: Especificaciones eléctricas del (GSProg).

4.3 Recursos utilizados del dispositivo FPGA ACEX 1K100 de Altera

A continuación se observa el análisis y síntesis de recursos utilizados por el dispositivo FPGA utilizado para realizar la implementación en hardware del GSProg.

Analysis & Synthesis Resource Usage Summary		
	Resource	Usage
1	Total logic elements	796
2	⊟ Total combinational functions	789
3	-- Total 4-input functions	275
4	-- Total 3-input functions	159
5	-- Total 2-input functions	352
6	-- Total 1-input functions	1
7	-- Total 0-input functions	2
8	Total registers	409
9	I/O pins	33
10	Maximum fan-out node	Clock
11	Maximum fan-out	409
12	Total fan-out	3587
13	Average fan-out	4.33

Figura 13: Análisis de recursos utilizados por el FPGA ACEX 1K100 de Altera.

Es importante mencionar que el uso de recursos del GSProg en el FPGA es del 26%

5. Desarrollo teórico

Después de haber determinado y especificado los parámetros de los tipos de señales digitales más comunes y utilizadas para verificar sistemas digitales, se procedió a plantear las especificaciones que debía tener el generador a diseñar. En esta sección se procede a mostrar el desarrollo teórico que permitió que el *GSProg* diseñado funcionara correctamente y cumpliera con dichas especificaciones.

5.1. Arquitectura detallada del sistema

En esta sección se encuentra la explicación del diagrama en bloques general del sistema, y la explicación detallada que muestra el funcionamiento de cada parte que compone el ***GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES*** (*GSProg*). El (*GSProg*) está comprendido por 2 sub-sistemas, **INTERFAZ** y **HARDWARE**, y el protocolo de comunicación RS232 que une a ambos.

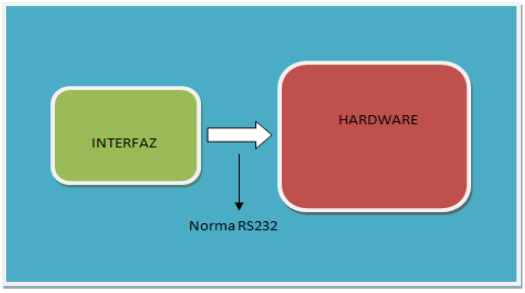


Figura 14: Sub-sistemas del GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES (GSProg).

5.1.1. La interfaz

La interfaz es un programa de computador desarrollado en Visual Basic 6.0 el cual posee un menú grafico con el fin de que el usuario del sistema introduzca de una manera fácil y ordenada los parámetros de las palabras digitales que desea obtener a la salida del generador. Por medio de este programa el usuario verá en el computador un menú con opciones para escoger. Estas opciones son:

- Tipo de Palabra digital (serial o paralela).
- Número de palabras digitales, entre 1 a 8.
- Número de Bits de la o las palabra digitales, entre 1 a 16.
- Posibilidad de ingresar o que el programa genere los bits que conforman las palabras digitales.
- Tipo de envío de las palabras digitales (único ó continuo).

El menú implementado en el software es el siguiente:

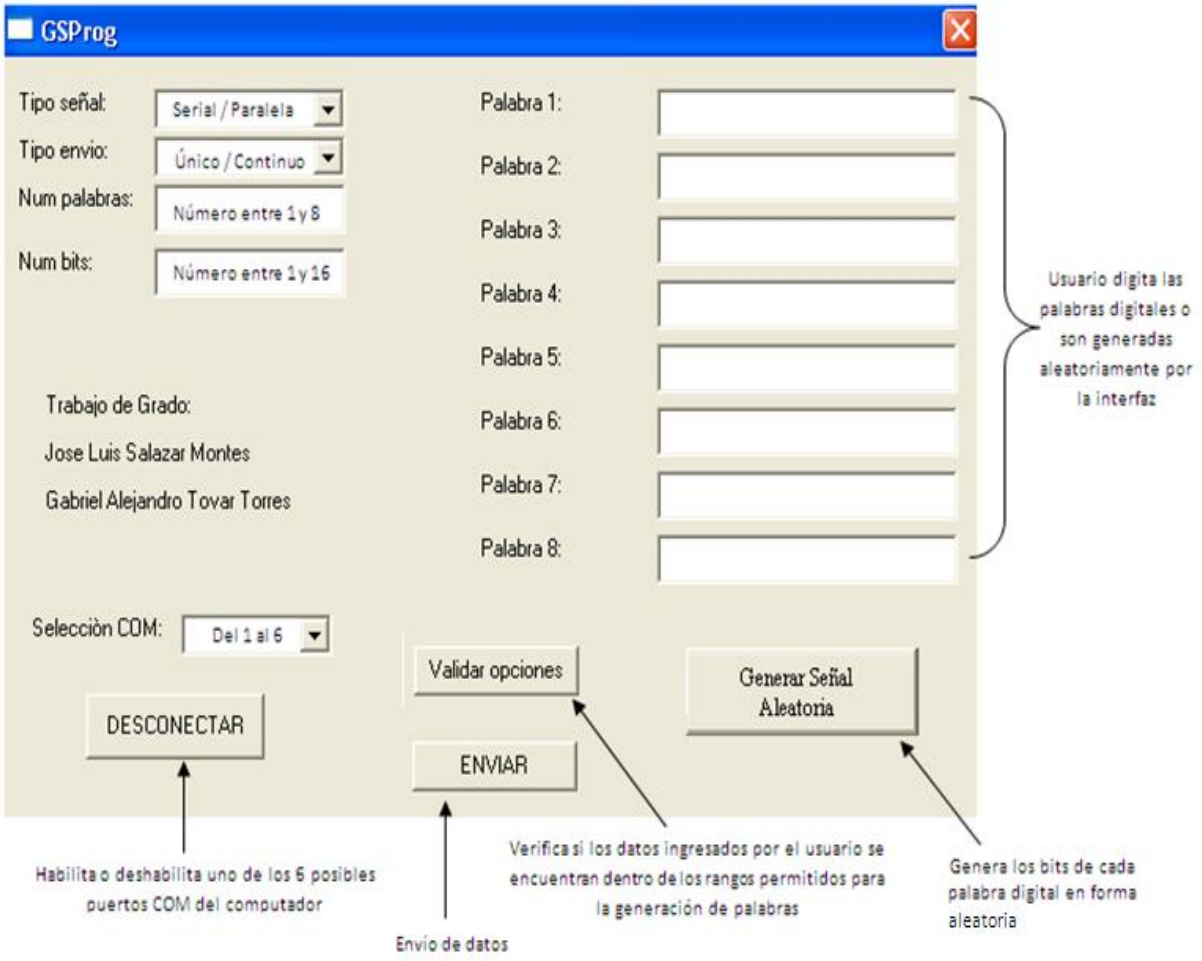


Figura 15: Interfaz gráfica que permite al usuario caracterizar las palabras digitales que se obtendrán a la salida del generador.

Una vez son escogidos e introducidos los datos que caracterizan las palabras digitales a generar por el sistema, son enviados a través del puerto serial del computador mediante el protocolo serial de comunicación RS 232. El envío de estos datos desde el computador al hardware se realiza por byte, donde va la información referente a las palabras digitales y su correspondiente caracterización, más un bit de start y uno de stop.

Como el generador puede entregar como máximo 8 palabras de 16 bits, equivalentes a un total de 128 bits de información, el computador debe enviar éstos bits a el FPGA en 16 envíos de byte de información; 128 bits / 8bits = 16 bytes de información correspondientes a los contenidos de cada palabra digital y a las características con las cuales deben ser entregadas por el generador. Las características que contienen la información que caracterizan a las palabras digitales son dispuestas en 9 bits de opciones.

- La opción enmarcada del bit 1 al 3 corresponde al número de palabras digitales que entregará el generador.

- La opción enmarcada del bit 4 al 7 corresponde al número de bits que contendrán las palabras digitales.

- La opción enmarcada en el bit 8 corresponde a la representación de las palabras digitales (seriales o paralelas).

- La opción enmarcada en el bit 9 corresponde al tipo de envío de las palabras digitales (Unico o Continuo).

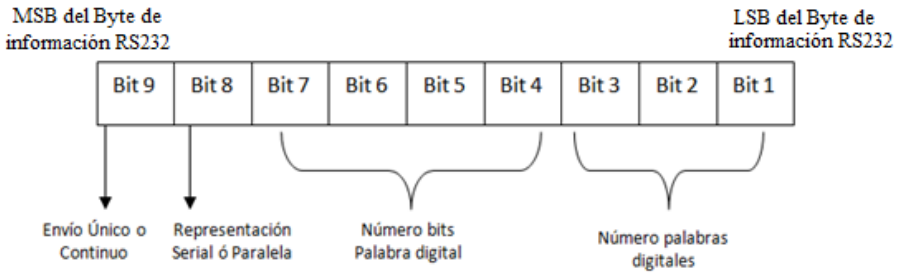


Figura 16: Disposición de los bits que permiten caracterizar las palabras digitales a entregar por parte del generador.

Como la norma RS232 contiene un byte (8 bits) de información y las opciones que caracterizan a las palabras digitales contienen 9 bits, el envío de estas opciones se debe realizar en 2 bytes mas, donde el segundo byte está compuesto por el Bit 9 en la posición LSB, y el resto de bits que completan el byte son 0 lógico.

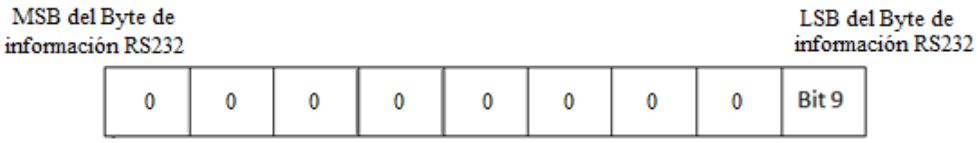


Figura 17. Segundo byte de información correspondiente a las opciones que caracterizan a las palabras digitales.

Así se tiene un total de 18 envíos de byte con sus respetivos bits de inicio y parada presentes en la norma RS232. Estos 18 envíos contienen toda la información para la correcta generación de las palabras digitales a entregar por el sistema. Los datos enviados desde el puerto serial del computador viajan a una velocidad de 9600 bps (tiempo de bit: 1/9600 = 104 us) paquetes de 8 bits, sin bit de paridad, con 1 bit de parada y sin control de flujo. Como método de detección de errores se utilizó el método de *redundancia de paquete* que consiste en enviar dos veces las tramas de 18 bytes, para un total 36 bytes enviados, con el fin de que se comparen éstos dos conjuntos de envíos y se determine si hubo o no error.

En cuanto a los niveles de voltaje que maneja el puerto serial del computador (usando el estándar RS-232) son diferentes a los que utiliza la FPGA y los demás circuitos integrados digitales, por lo tanto se utilizó un elemento que hizo posible la conversión de niveles de voltaje RS232 a los estándares manejados por los CI TTL, esto se hace a través del circuito integrado MAX 232 que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa.

Una vez son enviados los datos por parte de la interfaz gráfica a través del protocolo de comunicación RS232, éstos son recibidos por el **Hardware** el cual se encarga de clasificar, procesar y generar posteriormente las palabras digitales con sus respectivas señales de *Sincronismo* e *Inicio de Trama*. Éste **Hardware** se denomina **Generador**. El **Generador** está compuesto a su vez por 3 subsistemas cuyo funcionamiento conjunto permite que opere de forma correcta.

- Un bloque denominado **Sistema de Comunicación** encargado de recibir clasificar y procesar los datos provenientes del puerto serial del computador.

- Un bloque denominado **Generador de secuencia** encargado de recibir los datos provenientes del **Sistema de Comunicación** para procesarlos con el fin de generar las palabras digitales seriales o paralelas asíncronas con sus respectivas señales de *Sincronismo* e *Inicio de Trama*.

- Un bloque denominado **Unidad de Control** que permite que los bloques **Sistema de Comunicación** y **Generador de secuencia** interactúen entre sí de forma correcta en pulsos específicos de reloj.

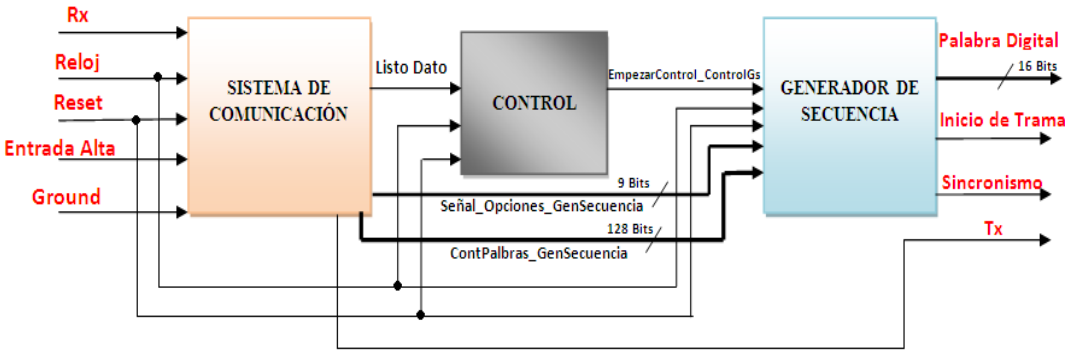


Figura 18: Bloques internos del Hardware (Generador).

Esta división en subsistemas se realizó con el fin de sectorizar las funciones del Generador para realizar un diseño ordenado, entendible y congruente con las especificaciones que éste debe cumplir. A continuación se describe de forma detallada el funcionamiento y diseño de cada uno de los su-bloques que lo conforman.

5.2. Sistema de comunicación

Este bloque recibe, clasifica y procesa los datos provenientes del puerto serial del computador, que contiene las instrucciones y palabras digitales que el usuario ingresa en la interfaz (señal *Rx*). Informa con una señal de voltaje alto, *Listo Dato*, 1 lógico, que su labor ha terminado y que los datos están listos para su utilización. En la figura 19 se muestra las entradas y salidas del Sistema de Comunicación, las cuales son explicadas a continuación.

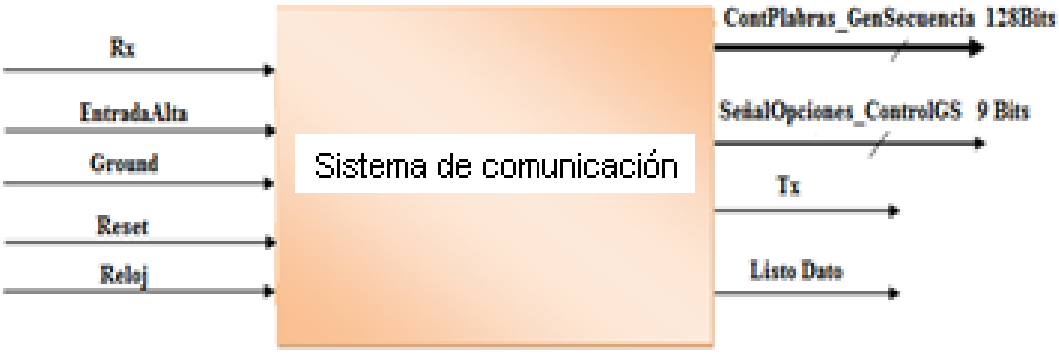


Figura 19: Entradas y salidas del Sistema de Comunicación.

5.2.1. Entradas y salidas del sistema de comunicación

Entradas

Rx, *EntradaAlta*, *Ground*, *Reset* y *Reloj*. Son las mismas entradas del *GSProg*. Ver sección 3.1, entradas del *GSProg.*

Salidas

- *ListoDato*: Señal de 1 bit que informa si el **Sistema de Comunicación** ha terminado de de recibir, clasificar y procesar los datos provenientes del puerto serial del computador. 1 lógico indica que los datos se encuentran listos a su salida para ser utilizados por la **Unidad de Control** perteneciente al **Generador**.

- *Tx:* Señal de 10 bits que informa al computador mediante un caracter Ç si cada envío de información proveniente del puerto serial del computador a través del protocolo RS232 ha sido recibida por el **Sistema de Comunicación**. También informa al programa de computador sobre el cual se implementó la interfaz si los 36 envíos de información provenientes del puerto serial han llegado de manera correcta, envío de un caracter B, o incorrecta, envío de un carácter M.

- *ContPalabras_GenSecuancia:* Bus de 128 bits que contiene la información de las palabras digitales provenientes del PC, estos bits corresponden a los bits que conforman las palabras digitales que son entregadas por el sistema. Son 128 bits debido a que lo máximo que puede entregar el **Generador** son 8 palabras de 16 bits, $8 \times 16\ bits = 128\ bits$. Así La primera palabra digital está conformada del bit 1 al 16, la segunda palabra del bit 16 al 32, la tercera palabra del bit 32 al 48, la cuarta palabra del bit 48 al 64, la quinta palabra del bit 64 al 80, la sexta palabra del bit 80 al 96, la séptima palabra del bit 96 al 112 y la octava del bit 112 al 128. Para la generación de menos palabras digitales de menor número de bits, los bits que conforman cada trama perteneciente a una palabra digital están en estado de 0 lógico.

- *SeñalOpciones_GenSecuancia:* Señal de 9 bits que contiene los parámetros que caracterizan las palabras digitales. Estos 9 bits representan los siguientes parámetros.

  - ✓ *Número de palabras digitales a enviar*: Como el sistema puede entregar entre 1 y 8 palabras digitales, ésta opción viene representada por tres bits que determinan el número de palabras digitales a generar.

  - ✓ *Numero de bits de las palabras digitales*: Como el sistema puede entregar entre 1 y 16 bits pertenecientes a cada palabra digital, ésta opción viene representada por cuatro bits que determinan el número de bits de cada palabra digital.

  - ✓ *Representación de las palabras digitales a la salida del* **Generador de Secuencia**: Esta opción viene representada por un número binario de un bit el cual permite saber si las palabras digitales a la salida del sistema estarán en formato serial o paralelo. Si el bit es 0 las palabras digitales a generar serán seriales, si el bit es 1 serán paralelas.

- ✓ *Tipo de envío:* Esta opción está representada por un bit el cual permite saber si el envío de las palabras digitales que debe entregar el generador a su salida es *Único o Continuo.* Si el bit es 0 el envío es *Único,* si el bit es 1 el envío es *Continuo.*

Estos bits contenidos en la señal *SeñalOpciones_GenSecuancia* se distribuyen de la siguiente manera:

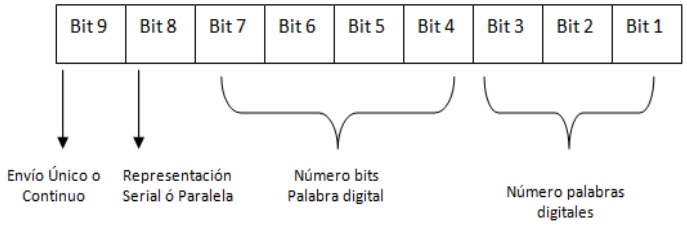


Figura 20: Distribución de bits contenidos en la señal SeñalOpciones_GenSecuencia.

5.2.2. Descripción de funcionamiento del sistema de comunicación

El **Sistema de Comunicación** está conformado por dos registros A y B, que a su vez están conformados cada uno por varios sub-registros, un bloque de **Comparar** y un bloque de control, **Control Siscom**. Debido a que la información llega desde el computador por un medio físico empleando la norma RS232, a través de un cable, se necesita estar seguro que la información llega correctamente. Para eso desde el computador se envían dos "ráfagas" de información en paquetes de 10 bits con las opciones y las palabras digitales.

El **Registro A** se encarga de almacenar el byte de información proveniente de *Rx* en el primer sub-registro de este registro. Luego de almacenados los bits que conforman el byte de datos se realiza un corrimiento de éstos entre los sub-registros del **Registro A** hasta llegar al último sub-registro del **Registro B**. Al finalizar el proceso, la primera "ráfaga" de información queda almacenada en los sub-registros del registro B y la segunda "ráfaga" de información queda almacenada en los sub-registros del registro A.

Almacenados los datos en los registros A y B, se procede a comparar la información de los sub-registros del **Registro A** con la información de los sub-registros del **Registro B** por medio del bloque de lógica combinatoria **COMPARAR** para determinar si la información procedente del computador llego correctamente.

A continuación se esquematiza el diagrama de bloques de la anterior descripción.

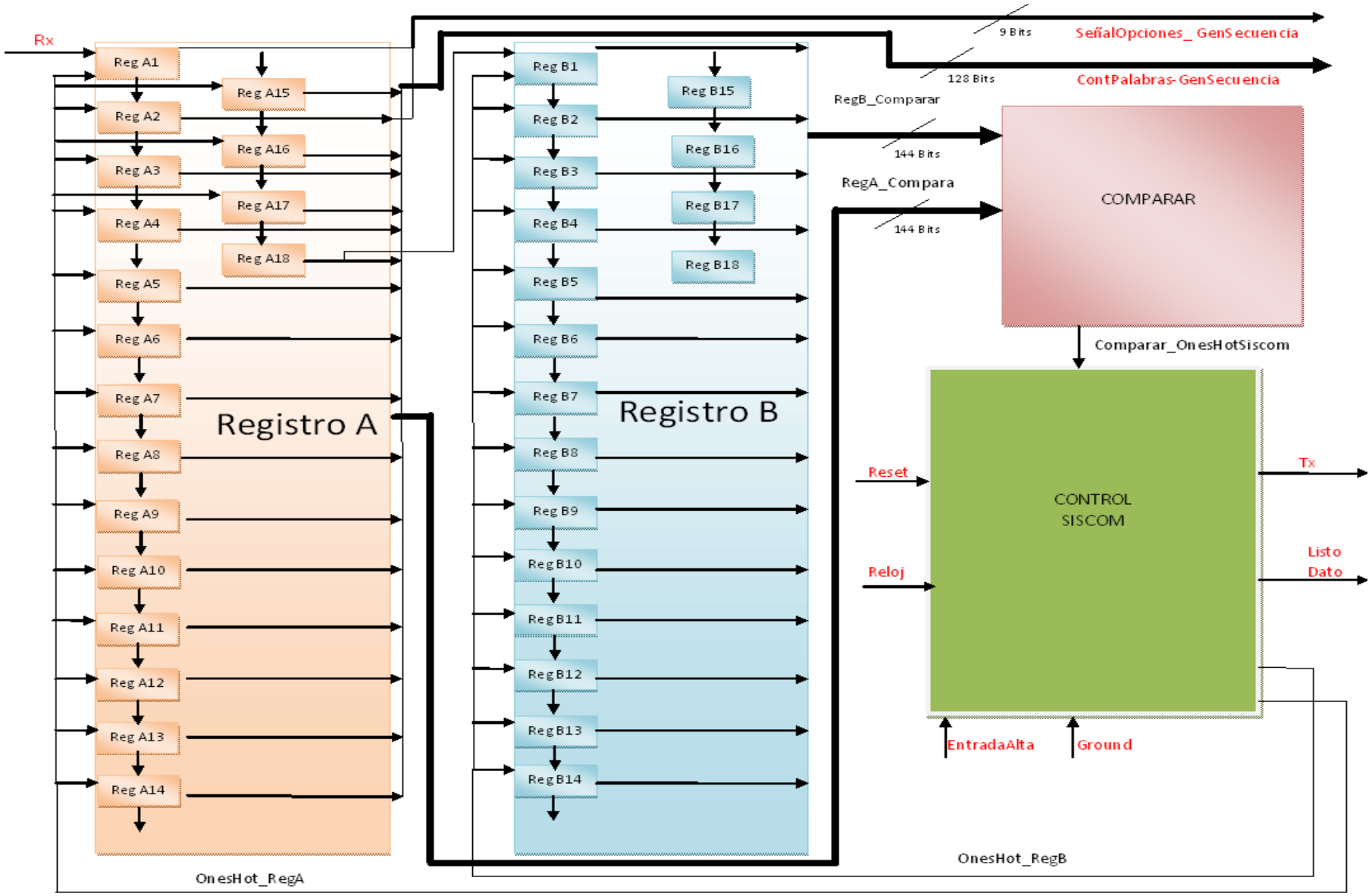


Figura 21: Diagrama en bloques del Sistema de comunicación ilustrando la composición interna del Registro A y del Registro B y el bloque de Control Siscom de manera general.

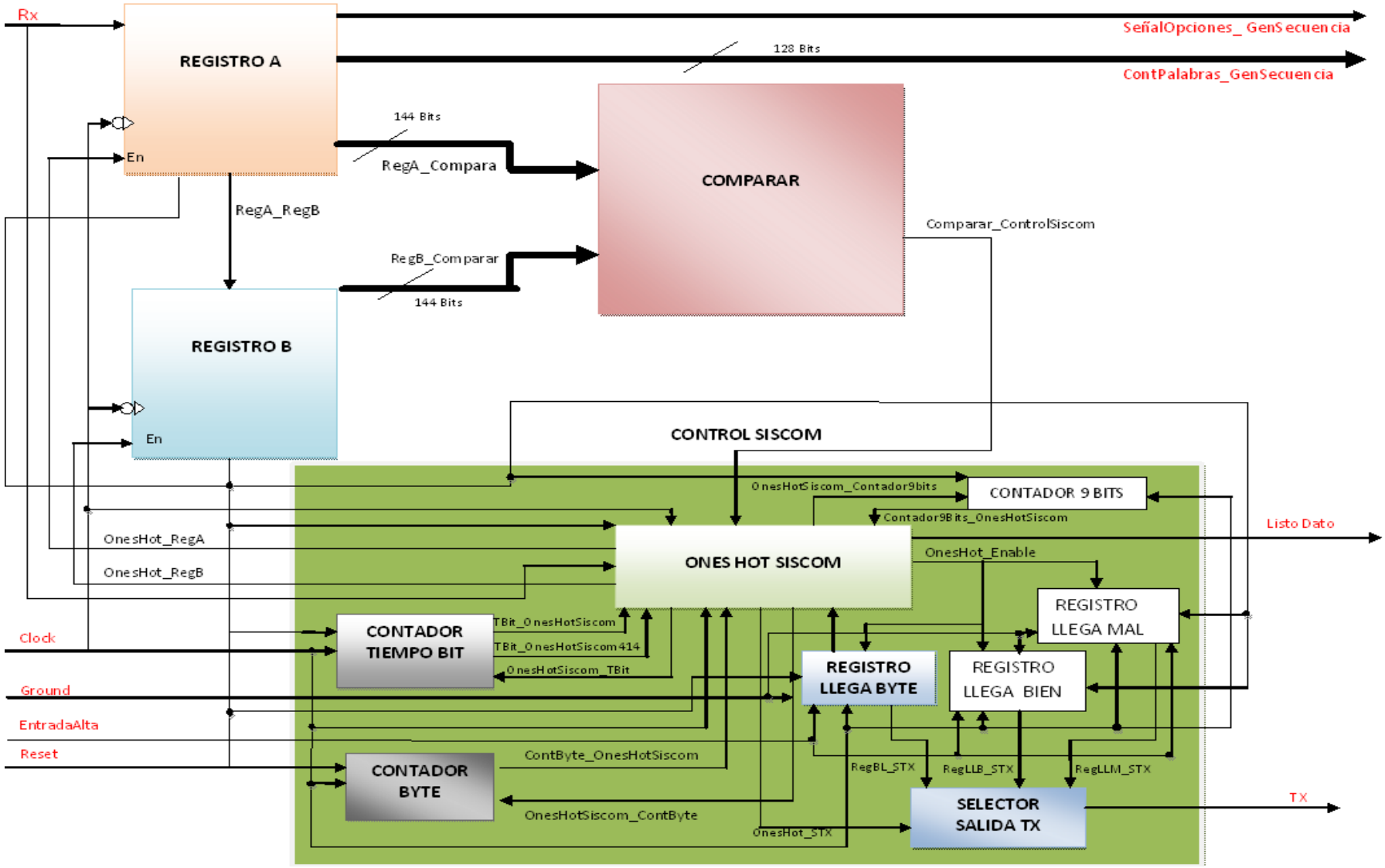


Figura 22: Diagrama en bloques del Sistema de Comunicación.

5.2.3. Descripción del diagrama de bloques del sistema de comunicación

A continuación se describen los bloques que componen este sistema.

- **Registro A**: Unidad de almacenamiento de datos conformado por 18 sub-registros de 8 posiciones cada uno.

  Señales de entrada:
  Rx, Clock, Reset y *OnesHot_RegA*.

  Señales de Salida: SeñalOpciones_GenSecuencia, ContPalabras_GenSecuencia, RegA_Comparar y *RegA_RegB.*

- **Registro B**: Unidad de almacenamiento de datos. Al igual que el **Registro A**, está conformado por 18 sub-registros de 8 posiciones cada uno.

  Señales de entrada: RegA_RegB, Clock Reset y OnesHot_RegB.
  Señales de Salida: RegB_Comparar.

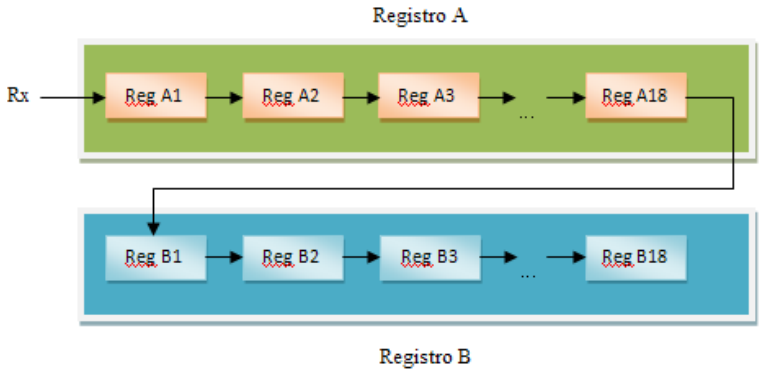


Figura 23: Esquematización de los registros A y B.

- **Comparar:** Bloque de lógica combinatoria encargado de comparar los datos almacenados en el **Registro A** y en el **Registro B**.

  Señales de entrada:
  RegA_Comparar y *RegB_Comparar*.

  Señal de Salida:
  Comparar_ControlSiscom.

- **Control Siscom:** Unidad de control encargada de controlar y dar las instrucciones a los bloques que componen el **Sistema de Comunicación** para su correcta operación.

  Señales de entrada:
  Rx, Clock y *Reset Comparar_ControlSiscom*.

  Señales de Salida:
  ListoDato y *Tx*.

Control Siscom está conformado por 8 sub-bloques como se describe a continuación:

a) **Ones Hot Siscom**: Máquina de estados de la unidad de control encargada de controlar los demas bloques pertenecientes a ésta unidad y a los de la unidad de datos del **Sistema de Comunicación**.

Señales de entrada:
Rx, Comparar, *ControlSiscom*, *Clock*, *Reset*, *TBit_OnesHotSiscom*, *ContByte_OnesHotSiscom* y *Contador9Bits_OnesHotSiscom*.

Señales de Salida:
OnesHot_RegASeñal, *OnesHot_RegB*, *OnesHotSiscom_TBit*, OnesHotSiscom_ContByte, *OnesHot_STX*, *OnesHot_Enable* y *OnesHotSiscom_Contador9Bits*.

b) **Contador Tiempo Bit**: Maquina de estados encargada de contar 832 pulsos de reloj para controlar el tiempo de bit de la señal *Rx* debido a que el periodo de la señal de reloj ($\frac{1}{\text{fclock}} = \frac{1}{8}\text{Mhz} = 0.125$ us) es menor al tiempo de bit de la señal *Rx* ($\frac{1}{9600\ \text{bps}} = 104$ us), por tanto 104us/0.125us = 832 pulsos de reloj.

Señales de entrada:
Clock, *Reset* y *OnesHotSiscom_Tbit*.

Señales de salida:
TBit_OnesHotSiscom414 y *TBit_OnesHotSiscom*.

c) **Contador Byte**: Bloque encargado de contar los 36 paquetes de 8 bits de la señal *Rx*, 36 envíos provenientes del puerto serial del computador, con el fin de determinar si las dos ráfagas de información, cada una de 18 envíos, han llegado al **Sistema de Comunicación**.

Señales de entrada:
Clock, *Reset* y OnesHotSiscom_ContByte.

Señales de salida:
ContByte_OnesHotSiscom.

d) **Selector Tx**: Bloque encargado de escoger la trama de bits correspondiente a:

- Si llego el byte de la señal *Rx*. Escoge trama correspondiente a 010000000 = caracter Ç.
- Si la ráfaga de información llego bien. Escoge trama correspondiente a 001000010 = caracter ASCII B.
- Si la ráfaga de información llego mal. Escoge trama correspondiente a 001001101 = caracter ASCII M.
- Si no está enviando Ç, M, o B debe enviar "Marca" o 1 lógico.
- El primer bit MSB dispuesto en 0 en cada trama de bits que representa los caracteres Ç, B y M y que corresponde al bit de Start de norma RS232, esto con el fin de que estos caracteres puedan ser enviados posteriormente al computador.

Con respecto a los cuatro últimos componentes que conforman la unidad **Control Siscom**, se explica a continuación el por qué se hicieron y diseñaron. Tres Sub-bloques, son registros de 9 bits (9 bits por que el bit menos significativo es el bit de Start y los 8 restantes corresponden al byte de información), que tienen información previa almacenada y que sirven para generar la trama correcta que se utiliza para formar el paquete de bits que conforma la norma RS232, así el contenido de la salida *Tx* depende del registro escogido y de la información que este contenga. Cada bit almacenado en los registros tiene una duración de 832 pulsos de reloj al momento de salir por la señal *Tx*. Esto debido a que 832 pulsos de reloj equivalen a 104 us, tiempo de bit de la transmisión serial RS 232 a 9600bps.

El último componente es un contador cíclico de 9 conteos que informa a la unidad **ONES HOT SISCOM** que los bits que conforman la norma RS232 para *Tx* se han completado.

e) **Registro Llega Bien:** Registro de 9 bits que contiene la información que se utiliza para generar la trama o paquete de bits que requiere la norma RS232 para transmitir e informar al computador que las ráfagas de información llagaron bien desde éste. Esta trama está conformada por el siguiente arreglo de bits 010000100, el cero LSB de este arreglo es usado como bit de Start y el byte (8 bits) 01000010 corresponde al carácter ASCII B, leído de derecha a izquierda.

Señales de entrada:
Clock, *Reset*, *EntradaAlta*, *Ground* y *OnesHot_Enable*.

Señales de salida:
RegLLB_STX.

f) **Registro Llega Mal:** Registro de 9 bits que contiene la información que se utiliza para generar la trama o paquete de bits que requiere la norma RS232 para transmitir e informar al computador que las ráfagas de información llagaron mal desde éste. Esta trama está conformada por el siguiente arreglo de bits 010011010, el cero LSB de este arreglo es usado como bit de Start y el byte (8 bits) 01001101 corresponde al carácter ASCII M, leído de derecha a izquierda.

Señales de entrada:
Clock, *Reset*, *EntradaAlta*, *Ground* y *OnesHot_Enable*.

Señales de salida:
RegLLM_STX.

g) **Registro Llega Byte**: Registro de 9 bits que contiene la información usada para generar la trama o paquete de bits usando la norma RS232 informando que el byte o paquete de información enviado desde el computador ha llegado al **Sistema de Comunicación**. Esta trama está conformada por el siguiente arreglo de bits 100000000, el cero LSB de este arreglo es usado como bit de Start y el byte (8 bits) 10000000 corresponde al carácter ASCII Ç.

Señales de entrada:
Clock, *Reset* , *EntradaAlta*, *Ground* y *OnesHot_Enable*.

Señales de salida: RegBL_STX.

h) Contador 9 Bits: Es un contador o maquina de 9 estados cíclica. Se encarga de avisar que los nueve bits almacenados en los anteriores registros (**Llega Byte**, **Llega Mal** y **Llega BIEN**) han sido utilizados como señal de salida *Tx*.

Señal de entrada:
Clock, Reset y *OnesHotSiscom_Contador9Bits*

Señale de salida:
Contador9Bits_OnesHotSiscom.

5.2.4. Descripción de las señales del sistema de comunicación

- ***Clock***: Señal de Reloj del ***GSProg*.**
- ***Reset***: Señal que reinicia el sistema.
- ***OnesHot_RegA***: Señal que activa los sub-registros que pertenecen al **Registro A** y que proviene de la unidad de control.
- ***RegA_Comparar***: Bus de 148 bits que contiene los datos almacenados en los sub-registros del **Registro A.**
- ***RegA_RegB***: Bus de 8 bits que contiene la información proveniente del último sub-registro del **Registro A, Reg A18.**
- ***OnesHot_RegB***: Señal que activa los sub-registros que pertenecen al **Registro B** y que proviene de la unidad de control.
- ***RegB_Comparar***: Bus de 148 bits que contiene los datos almacenados en los sub-registros del **Registro B**.
- ***Comparar_ControlSiscom***: Señal de un bit que informa a **Control Siscom** si la información contenida en los **Registros A** y **B** es igual o diferente. 1 lógico indica que es igual, 0 lógico indica que es diferente.
- ***TBit_OnesHotSiscom***: Señal de un bit proveniente del bloque **Contador Tiempo Bit** que indica que el contador ya conto los 832 pulsos de reloj.
- ***TBit_OnesHotSiscom414***: Señal de un bit proveniente del bloque **Contador Tiempo Bit** que indica que el contador ya conto los 414 pulsos de reloj. Esto con el fin de garantizar que los datos provenientes del puerto serial del computador que viajan a una velocidad de 9600 kbps sean leídos de manera correcta por el **Sistema de Comunicación** ya que cada dato es adquirido a los 51.75 us ($414 \times 0.125\ us = 51.75\ us$) de su duración total equivalente a 104 us.
- ***ContByte_OnesHotSiscom***: Señal de un bit proveniente del bloque **Contador Byte** que indica que ya han pasado los 36 envíos, 2 ráfagas de 18 envíos, de información provenientes del puerto serial del computador**.**
- ***OnesHot_RegASeñal***: Señal que habilita o deshabilita el **Registro A**.
- ***OnesHot_RegB***: Señal que habilita o deshabilita el **Registro B.**
- ***OnesHotSiscom_TBit***: Señal que habilita o deshabilita el **Contador Tiempo Bit**.
- **OnesHotSiscom_ContByte**: Señal que habilita o deshabilita **Contador Byte.**

- ***OnesHot_Enable***: Señal que habilita o deshabilita **Registro Llega Mal** y **Registro Llega Bien.**
- ***RegBL_STX***: Señal de un bit proveniente del bloque **Registro Llega Byte**.
- **RegLLB_STX**: Señal de un bit proveniente del bloque **Registro Llega Bien.**
- ***RegLLM_STX***: Señal de un bit proveniente del bloque **Registro Llega Mal**.
- ***OnesHot_STX***: bus de 2 bits proveniente del bloque **Ones Hot Siscom.**
- ***OnesHotSiscom_Contador9Bits:*** Señal que habilita o deshabilita **Contador 9 Bits.**
- ***Contador9Bits_OnesHotSiscom***: Señal de un bit proveniente del bloque **Contador 9 Bits** y que informa que el contador ha contado hasta 9.

5.3. Generador de secuencias

Este bloque está encargado de generar las palabras digitales seriales o paralelas, caracterizadas por el usuario, con sus correspondientes señales de Sincronismo e Inicio de Trama siguiendo las normativas especificadas en cuanto a los tiempos de bit de cada palabra y la duración de cada una de estas señales, (Ver especificaciones del *GSProg*, Sección 3).

En la figura 24 se muestra las entradas y salidas del Generador de Secuencia, las cuales son explicadas a continuación.

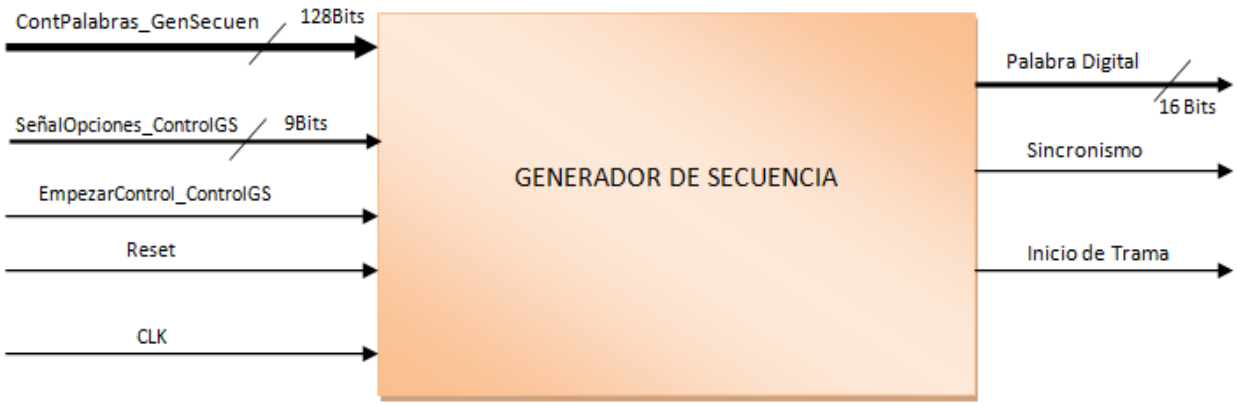


Figura 24: Entradas y salidas del sistema Generador de Secuencia.

5.3.1. Entradas y salidas del generador de secuencias

Entradas

ContPalabras_GenSecuencia: Señal de 128 bits proveniente del bloque **Sistema de Comunicación** y descrita en las salidas de éste. ***Ver sección 4.2.1, salidas del Sistema de Comunicación***.

SeñalOpciones_ControlGS: Señal de 9 bits proveniente del bloque **Sistema de Comunicación** que contiene la información de las opciones digitadas por el usuario en el programa de computador (*Tipo de palabra digital;* serial o paralelo, *número de palabras digitales*, *tamaño de la palabra digital*, *tipo de envío*; único o continuo). Ver distribución de sus bits en la ***figura 18.***

EmpezarControl_ControlGS: Señal proveniente del bloque de **Control** que indica si los datos presentes a la entrada del **Generador de Secuencia** están listos para ser procesados.

Reset: Señal de un bit que se encarga de reiniciar el sistema o volverlo a su condición inicial.

Reloj: Señal de Reloj de 8 MHz del **Generador**.

Salidas

Palabra Digital, Sincronismo e Inicio de Trama. Ver sección 3.1, salidas del GSProg.

5.3.2. Descripción de funcionamiento del generador de secuencias

En la unidad de datos se encuentran los bloques **Selector Representación Serial** y **Selector Representación Paralelo** encargados de seleccionar los bits que conformarán las palabras digitales seriales o paralelas provenientes del **Sistema de Comunicación** en la señal ***ContPalabras_GenSecuencia*** de forma correcta y ordenada. Para logar tal propósito estos selectores son controlados por **Maquina de Estados Selector Serial** y **Maquina de Estados Selector Paralelo** contenidas en la unidad de control, **Control Gs.** La secuencia de los estados internos de la **Maquina de Estados Selector Serial** está condicionada por la salida del bloque **Número de Bits** el cual se encarga de informar a dicha maquina cuando han acabo los bits de una palabra digital para que ésta salte a un estado no consecutivo y proceda a hacer la selección de los bits de la siguiente palabra digital en el caso serial. Para el caso paralelo, **Maquina de Estados Selector Paralelo,** es una maquina cuyos estados son secuenciales y cíclicos ya que los bits de las palabras digitales paralelas son representados uno a uno en un bus de 16 bits.

Una vez el sistema digital ha seleccionado de forma correcta y ordenada los bits de las palabras digitales éste procede a ajustar los tiempos de bit (3us), los tiempos de descanso entre un bit y el siguiente (2us) y posteriormente a generar las señales de ***Inicio de Trama*** y ***Sincronismo*** con sus respectivos tiempos de bit (1us).

El ajuste de los tiempos de bit se realiza mediante el bloque **Contador 8 pulsos 1us**, bloque controlado por la **Maquina de Estados Ones Hot** y encargado de contar cíclicamente 8 pulsos de reloj equivalentes a un tiempo de 1us, $\frac{1}{8\,MHz} \times 8 = 1\,us$. En el primer conteo **Maquina de Estados Ones Hot** genera la señal de ***Inicio de Trama*** con duración de 1us. En los siguientes 3 conteos se determina el tiempo de bit (3us), donde simultáneamente al estar presente el segundo conteo **Maquina de Estados Ones Hot** genera la señal de ***Sincronismo*** de bit (para palabras digitales seriales) o de palabra digital (para palabras digitales paralelas).

Una vez transcurridos estos 3 us **Maquina de Estados Ones Hot** activa los bloques **Dato Descanso Serial** o **Dato Descanso Paralelo** dependiendo si la palabra digital a entregar por el sistema es serial o

paralela. Estos bloques se encargan de dejar pasar el bit durante 3us, donde una vez cumplido este tiempo **Maquina de Estados Ones Hot** les ordena generar el descanso de bit (0 lógico) durante 2 conteos de **Contador 8 pulsos 1us**, (2us).

El bloque **Salida** es dispuesto en el sistema para permitir poner en el primer bus de salida del **Generador de Secuencia**, *Palabra Digital (1)*, las palabras digitales en formato serial ó un bit perteneciente a una palabra digital en formato paralelo. **Salida** es controlado por **Maquina de Estados Ones Hot** gracias a que éste bloque perteneciente a la unidad de control sabe si la palabra digital a entregar es serial o paralela ya que procesa la información contenida a su entrada en ***SeñalOpciones_ControlGS***, señal que posee las características de las señales digitales a generar y que permite a **Maquina de Estados Ones Hot** tomar decisiones para el control de **Salida**.

El bloque **Número de Palabras** presente en la unidad de control informa a **Maquina de Estados Ones Hot** que todas las palabras digitales deseadas por el usuario se han entregado a la salida para que ésta deje de seguir dando instrucciones a todos los bloques que controla para finalizar el proceso.

A continuación se esquematiza el diagrama de bloques de la anterior descripción.

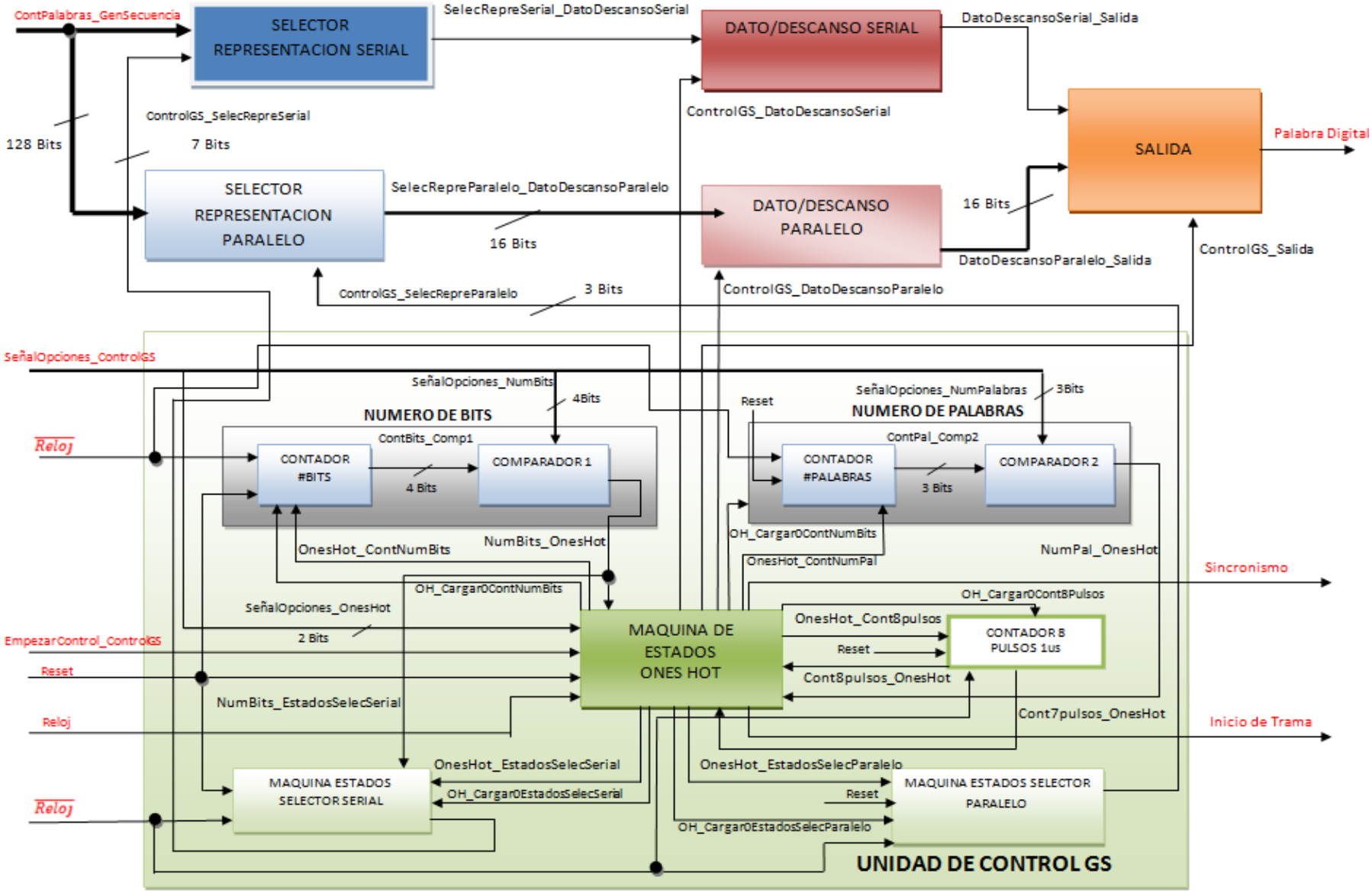


Figura 25: Diagrama de bloques del Generador de Secuencia.

5.3.3. Descripción del diagrama de bloques del generador de secuencias

A continuación se describen los bloques que componen este sistema.

- **Selector Representación Serial**: Bloque encargado de seleccionar los bits presentes en la señal ***ContPalabras_GenSecuencia*** para entregar a su salida de forma ordenada y consecutiva los bits que conformarán las palabras digitales seriales.

  Señales de entrada:
  ContPalabras_GenSecuencia y ***ControlGS_SelecRepreSerial.***

  Señales de salida:
  SelecRepreSerial_DatoDescansoSerial.

- **Selector Representación Paralelo**: Bloque encargado de seleccionar los bits presentes en la señal ***ContPalabras_GenSecuencia*** para entregar a su salida de forma ordenada y consecutiva los bits que conformarán las palabras digitales paralelas.

  Señales de entrada:
  ContPalabras_GenSecuencia y ***ControlGS_SelecRepreParalelo.***

  Señales de salida:
  SelecRepreParalelo_DatoDescansoParalelo.

- **Dato/Descanso Serial:** Bloque encargado de mantener a su salida cada bit perteneciente a las palabras digitales seriales durante 3us. Una vez ha transcurrido este tiempo el bloque genera el descanso de bit durante 2us.

  Señales de entrada:
  SelecRepreSerial_DatoDescansoSerial y ***ControlGS_DatoDescansoSerial .***

  Señales de salida: ***DatoDescansoSerial_Salida.***

- **Dato/Descanso Paralelo:** Bloque encargado de mantener a su salida cada bit perteneciente a las palabras digitales paralelas durante 3us. Una vez ha transcurrido este tiempo el bloque genera el descanso de bit durante 2us.

  Señales de entrada:
  SelecRepreParalelo_DatoDescansoParalelo y ***ControlGS_DatoDescansoParalelo .***

  Señales de salida: ***DatoDescansoParalelo_Salida.***

- **Salida:** Bloque encargado de poner en el primer bus de ***Palabra Digital***, ***Palabra Digital (1)***, las palabras digitales seriales ó el bit LSB correspondiente a una palabra digital paralela.
  Señales de entrada: ***DatoDescansoSerial_Salida***, ***DatoDescansoParalelo_Salida*** y ***ControlGS_Salida***.
  Señales de salida: ***Palabra Digital***.

- **Control Gs:** Unidad de control del **Generador de Secuencia**. Se encarga de controlar y dar instrucciones a los demás bloques pertenecientes a éste sistema.

Señales de entrada:
Reloj, ***Reset***, ***SeñalOpciones_ControlGS*** y ***EmpezarCOntrol_COntrolGS***.

Señales de Salida: ***Sincronismo*** e ***Inicio Trama***.

Control Gs está conformado por 6 sub-bloques como se describe a continuación:

a) Contador 8 Pulsos 1us: Este bloque es una máquina de 8 estados cíclica que se encarga de contar 8 pulsos del reloj que equivalen a un tiempo de 1us (El reloj del sistema es de 8 MHz, $\frac{1}{8}$ MHz $=$ 0.125 us, 0.125 us X $8 = 1$ us). Este bloque es de suma importancia ya que gracias a él es posible definir los tiempos de bit con su descanso, y los tiempos de las señales de ***Sincronismo*** e ***Inicio de Trama***.

Señales de entrada:
OnesHot_Cont8Pulsos, ***OH_Cargar0Cont8Pulsos***, ***Reset*** *y* ***Reloj***.

Señales de salida:
Cont8Pulsos_ OnesHot *y* ***Cont7Pulsos_ OnesHot***.

b) Numero de Bits: Bloque encargado de informar a los bloques **Maquina de Estados Ones Hot** y **Maquina Estados Selector Serial** que el número de bits de las palabras digitales especificado por el usuario se ha completado. Este bloque está compuesto por 2 bloques internos, **Contador Número Bits** y **Comparador 1. Contador Número Bits** es un contador de 4 bits representado en una maquina de 16 estados y encargado de hacer el conteo del número de bits de la palabra digital hasta el número de bits especificado por el usuario, información presente en ***SeñalOpciones_ControlGS [4:7]***. La salida de este contador va al bloque **Comparador 1** el cual se encarga de comparar el estado presente del contador, ***ContBits_Comp1***, con **SeñalOpciones_ControlGS** ***[4:7]***. Una vez hecha la comparación éste comparador entrega a su salida un 0 lógico si la comparación no es igual ó un 1 lógico si la comparación es igual.

Señales de entrada:
SeñalOpciones_NumBits, ***Reloj***, ***Reset***, ***OH_Cargar0ContNumBits***, y ***OnesHot_ContNumBits***.

Señales de salida: ***NumBits_ OnesHot*** y ***NumBits_EstadosSelecSerial***.

c) Numero de Palabras: Bloque encargado de informar a **Maquina de Estados Ones Hot** que el número de palabras digitales especificado por el usuario se ha completado, esto con el fin de informarle a todo el sistema completo que ya se han entregado todas las palabras digitales requeridas por el usuario. Este bloque está compuesto por 2 bloques internos, **Contador Numero Palabras** y **Comparador 2. Contador Numero Palabras** es un contador de 3 bits representado en una maquina de 8 estados y encargado de hacer el conteo del número de palabras digitales hasta el número de palabras digitales especificado por el usuario, información presente en ***SeñalOpciones_ControlGS [1:3]***. La salida de este contador va al bloque **Comparador 2** el cual se encarga de comparar el estado presente del contador, ***ContPal_Comp2,*** con ***SeñalOpciones_ControlGS [1:3]***. Una vez hecha la comparación éste comparador entrega a su salida un 0 lógico si la comparación no es igual ó un 1 lógico si la comparación es igual.

Señales de entrada:
SeñalOpciones_NumPalabras, ***Reloj***, ***Reset***, ***OH_Cargar0ContNumPal*** y ***OnesHot_ContNumPal***.

Señales de salida:

NumPal_ OnesHot.

d) Maquina Estados Selector Serial: Maquina de estados de 7 bits y 128 estados encargada de manejar los selectores de **Selector Representación Serial** para asegurar así la efectiva escogencia ordenada de los bits que conforman las palabras digitales seriales presentes en ***ContPalabras_GenSecuencia***. En esta máquina de estados el pasar de un estado presente a uno siguiente depende de la entrada ***NumBits_EstadosSelecSerial***, ya que esta entrada indica cuando acabaron los bits de una palabra digital para así poder pasar a la escogencia de los bits de la siguiente palabra digital. Esto se ve reflejado en que la maquina salta de un estado presente a otro no consecutivo lo cual representa que el selector escoja el primer bit de la siguiente palabra digital, esto en el caso de que el sistema deba entregar más de 1 palabra digital a su salida.
Cada vez que ***NumBits_EstadosSelecSerial*** = 1 se hacen saltos de 16 estados, por consiguiente la maquina pasa por los estados 1, 16, 32, 64, 80, 96, 112 y 128. Cabe resaltar que cada uno de estos estados corresponde a la escogencia del primer bit de una nueva palabra digital, por tanto como son 8 palabras digitales se observa que son 8 estados no consecutivos a los que se salta.

Señales de entrada:
NumBits_EstadosSelecSerial, ***OnesHot_EstadosSelecSerial***, ***Reloj***, ***Reset***, y ***OH_Cargar0EstadosSelecSerial***.

Señales de salida:
ControlGS_SelecRepreSerial.

e) Maquina Estados Selector Paralelo: Maquina de estados de 3 bits y 8 estados encargada de manejar los selectores de **Selector Representación Paralelo** para asegurar así la efectiva escogencia ordenada de los bits que conforman las palabras digitales en formato paralelo. Cada estado de esta máquina controla los selectores de **Maquina Estados Selector Paralelo** para que éste permita pasar una palabra digital de 1-16 bits. La maquina es de 8 estados debido a que el número de palabras digitales en formato paralelo máximo que el sistema puede generar son 8.

Señales de entrada:
OnesHot_EstadosSelecParalelo, ***Reloj***, ***OH_Cargar0EstadosSelecParalelo*** y ***Reset***.

Señales de salida:
ControlGS_SelecRepreParalelo.

f) Maquina de Estados Ones Hot: Bloque de gran importancia encargado de dar instrucciones para activar y desactivar en pulsos de reloj específicos los distintos bloques pertenecientes a la unidad de control y datos para la correcta funcionabilidad del sistema digital, es el encargado de informar al sistema si las palabras digitales a la salida del generador deben ser seriales o paralelas, y si éstas deben ser entregadas una o múltiples veces por el generador, envío *Unico* y *Continuo*, esto gracias a la información provista a su entrada en *SeñalOpciones_ControlGS[8]* y *SeñalOpciones_ControlGS [9]*. Finalmente también se encarga de generar las salidas de ***Sincronismo*** e ***Inicio de trama*** de las palabras digitales presentes a la salida del **Generador**.

Señales de entrada:
Reset, ***Reloj***, ***EmpezarControl_ControlGS***, ***NumBits_OnesHot***, ***NumPal_OnesHot***, ***Cont8Pulsos_OnesHot*** y ***Cont7Pulsos_OnesHot***.

Señales de salida:
OnesHot_EstadosSelecSerial,***OnesHot_EstadosSelecParalelo***,***OnesHot_NumPal***,***OnesHot_NumBits***,***OnesHot_Contador8Pulsos***,***ControlGs_DatoDescansoSerial***,***ControlGs_DatoDescansoParalelo***, ***ControlGs_Salida***,***OH_Cargar0EstadosSelecParalelo***,***OH_Cargar0EstadosSelecSerial***,***OH_Cargar***

0ContNumPal*,*OH_Cargar0ContNumBits*, *OH_Cargar0Cont8Pulsos*, *Sincronismo e ***Inicio de Trama.***

5.3.4. Descripción de las señales del generador de secuencias

- ***SelecRepreSerial_DatoDescansoSerial:*** Señal que contiene los bits que conforman la palabra digital serial, y que fueron escogidos previamente por **Selector Representación Serial,** aclarando que éstos bits están sin los tiempos de bit y descanso establecidos por el sistema y ya explicados anteriormente.
- ***SelecRepreParalelo_DatoDescansoParalelo:*** Señal de 16 bits que representa una palabra digital en forma paralela pero sin los tiempos de bit y descanso establecidos por el sistema y ya explicados anteriormente.
- ***DatoDescansoSerial_Salida*:** Señal que contiene los bits que conforman la palabra digital serial con los tiempos de bit y descanso establecidos por el sistema y ya explicados anteriormente.
- ***DatoDescansoParalelo_Salida:*** Señal de 16 bits que representa una palabra digital en forma paralela pero con los tiempos de bit y descanso establecidos por el sistema y ya explicados anteriormente.
- ***NumBits_OnesHot:*** Señal de un bit que informa a **Maquina de Estados Ones Hot** que el número de bits de la o las palabras digitales ha sido completado. Un 0 lógico indica que aun no se ha completado el número de bits, mientras que un 1 lógico indica que ya se ha completado el número de bits de la palabra digital.
- ***NumBits_EstadosSelecSerial:*** Señal de un bit que informa a **Maquina de Estados Selector Serial** que el número de bits de la o las palabras digitales ha sido completado. Un 0 lógico indica que aun no se ha completado el número de bits, mientras que un 1 lógico indica que ya se ha completado el número de bits de la palabra digital. Esto con el fin de que **Maquina de Estados Selector Serial** sepa cuando saltar de un estado a otro no consecutivo, el salto se efectúa si ***NumBits_EstadosSelecSerial*** =1.
- ***NumPal_OnesHot:*** Señal de un bit que informa a **Maquina de Estados Ones Hot** que el número de palabras digitales ha sido completado. Un 0 lógico indica que aun no se ha completado y un 1 lógico indica que ya se han completado las palabras digitales.
- ***Cont8Pulsos_OnesHot:*** Señal encargada de informar a **Maquina de Estados Ones Hot** que el **Contador 8 Pulsos 1us** ha realizado el conteo de 8 pulsos de reloj equivalentes a 1us.
- ***Cont7Pulsos_OnesHot:*** Señal encargada de informar a **Maquina de Estados Ones Hot** que el **Contador 8 Pulsos 1us** ha realizado el conteo de 7 pulsos de reloj.
- ***OnesHot_Cont8Pulsos:*** Señal de control encargada de activar el **Contador 8 Pulsos 1us** para que éste empiece a contar.
- ***OnesHot_ContNumBits:*** Señal de control encargada de activar el **Contador Número Bits** para que éste empiece a contar.
- ***OnesHot_ContNumPal:*** Señal de control encargada de activar el **Contador Número Palabras** para que éste empiece a contar.
- ***OnesHot_EstadosSelecSerial:*** Señal de control encargada de activar o desactivar **Máquina Estados Selector Serial.**
- ***OnesHot_EstadosSelecParalelo:*** Señal de control encargada de activar o desactivar **Máquina Estados Selector Paralelo.**
- ***SeñalOpciones_NumBits:*** Señal de 4 bits, ***SeñalOpciones_ControlGS [4:7]***, que contiene el número de bits de las palabras digitales a entregar por el sistema.

- ***SeñalOpciones_NumPalabras:*** Señal de 3 bits, ***SeñalOpciones_ControlGS [1:3]***, que contiene el número de las palabras digitales a entregar por el sistema.
- ***ContBits_Comp1:*** Señal de 4 Bits que contiene el estado presente de **Contador Número Bits.**
- ***ContPal_Comp2:*** Señal de 3 Bits que contiene el estado presente de **Contador Número Palabras.**
- ***ControlGs_DatoDescansoSerial:*** Señal de control de 1 bit encargada de controlar el bloque **Dato/Descanso Serial.** Si es igual a 1 lógico, el bloque controlado permite dejar pasar durante 3 us un bit perteneciente a una palabra digital serial, si por el contrario es igual a 0 lógico, el bloque genera el tiempo de descanso de bit durante 2 us.
- ***ControlGs_DatoDescansoParalelo:*** Señal de control de 1 bit encargada de controlar el bloque **Dato/Descanso Paralelo.** Si es igual a 1 lógico, el bloque controlado permite dejar pasar durante 3 us los bits pertenecientes a una palabra digital Paralela, si por el contrario es igual a 0 lógico, el bloque genera el tiempo de descanso de cada bit durante 2 us.
- ***ControlGs_Salida:*** Señal de control que permite al bloque **Salida** poner en uno de sus buses de salida las palabras digitales seriales (si ésta señal es 0 lógico) ó un bit correspondiente a una palabra digital paralela (si esta señal es un 1 lógico).
- ***ControlGs_SelecRepreSerial:*** Señal de 7 bits encargada de controlar los selectores de **Selector Representación Serial**.
- ***ControlGs_SelecRepreParalelo:*** Señal de 3 bits encargada de controlar los selectores de **Selector Representación Paralelo**.
- ***OH_Cargar0ContNumBits:*** Señal proveniente de **Máquina de Estados Ones Hot** encargada de cargar con 0 el **Contador Número Bits**.
- ***OH_Cargar0ContNumPal:*** Señal proveniente de **Máquina de Estados Ones Hot** encargada de cargar con 0 el **Contador Número Palabras**.
- ***OH_Cargar0Cont8Pulsos:*** Señal proveniente de **Máquina de Estados Ones Hot** encargada de cargar con 0 el **Contador 8 Pulsos 1us**.
- ***OH_Cargar0EstadosSelecSerial:*** Señal proveniente de **Máquina de Estados Ones Hot** encargada de cargar con 0 la **Máquina Estados Selector Serial.**
  OH_Cargar0EstadosSelecParalelo: Señal proveniente de **Máquina de Estados Ones Hot** encargada de cargar con 0 la **Máquina Estados Selector Paralelo.**

5.4. Unidad de control

Unidad encargada de recibir la señal de ***Listo Dato*** proveniente del bloque **Sistema de Comunicación,** para generar la señal de ***EmpezarControl_ControlGS*** que informa que los datos están listos y de la forma adecuada para que el bloque **Generador de Secuencia** pueda comenzar a procesarlos para generación de las palabras digitales.

En la figura 26 se muestra las entradas y salidas del **Control** las cuales son explicadas a continuación.

Figura 26: Entradas y salidas del sistema de Control.

5.4.1. Entradas y salidas de control

Entradas

- ***Listo Dato:*** Señal que informa a **Control** que **Sistema de Comunicación** ha finalizado su función.
- **Reloj:** Señal de reloj del sistema.
- ***Reset:*** Señal asíncrona externa que reinicia el sistema.

Salida

EmpezarControl_ControlGS: Señal de un bit que informa al bloque **Generador de Secuencia**, que puede comenzar a operar.

6. Pruebas y análisis de resultados

Para la verificación del correcto funcionamiento del sistema ***GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACIÓN DE SISTEMAS DIGITALES (GSProg)***, se contó con la herramienta de software QUARTUS II, utilizada para el análisis y síntesis de diseños realizados en **HDL** (*Hardware Description Language),* y el analizador de estados lógicos LA5000-40 con el fin de visualizar , analizar y posteriormente verificar si las señales entregadas por el generador cumplen con las especificaciones descritas en el ***numeral 3, Especificaciones.***

El análisis de resultados del ***GSProg*** se dividió así:

- Recepción y transmisión de datos entre el computador y el **Sistema de Comunicación** perteneciente al **Generador** implementado en la FPGA.
- Pruebas con palabras digitales seriales en modo *Único*.
- Pruebas con palabras digitales seriales en modo *Continuo*.
- Pruebas con palabras digitales paralelas en modo *Único*.
- Pruebas con palabras digitales paralelas en modo *Continuo*.

6.1. Recepción y transmisión de datos entre el computador y el sistema de comunicación perteneciente al generador implementado en la FPGA

Las propiedades del puerto serial del computador para la transmisión y recepción de datos con la FPGA, mediante el protocolo de comunicación RS232, fueron configuradas como se muestra a continuación.

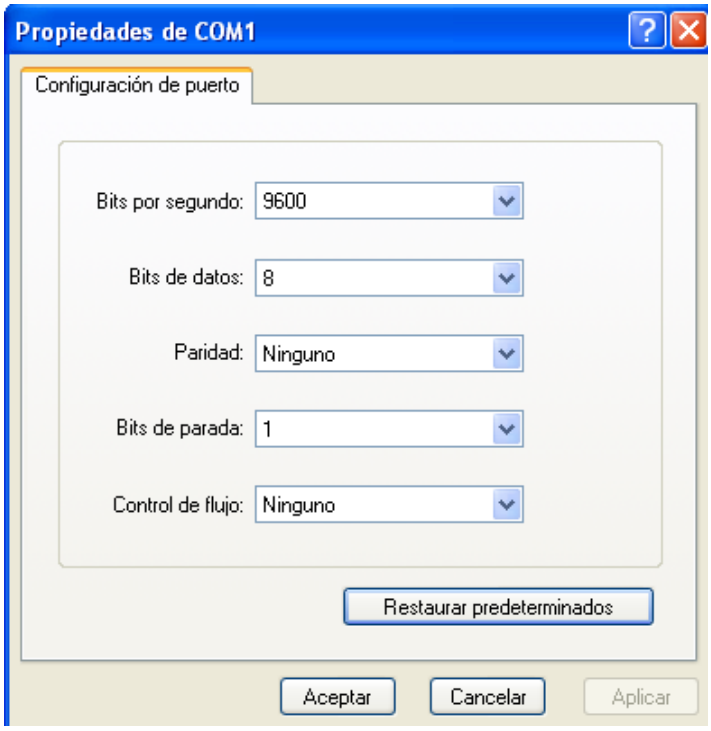


Figura 27: Configuración de parámetros para la transmisión y recepción de datos entre el computador y la FPGA mediante el protocolo de comunicación RS 232.

Con base a las propiedades de la figura anterior fue que se realizaron todos los cálculos con los cuales el **Sistema de Comunicación** del ***GSProg*** pudo recibir clasificar y procesar los datos provenientes del puerto serial del computador.

Esta prueba se realizó de 2 formas.

- A través del programa *HyPerterminal* donde una de sus aplicaciones es la comunicación serial RS232 entre el puerto serial del computador y otros dispositivos electrónicos.

- Por medio de la interfaz gráfica realizada en Visual Basic 6.0.

6.1.1. Conexión con Hyperterminal

El programa *HyPerterminal* permitió visualizar de forma clara la transmisión y recepción de datos entre el computador y el FPGA como sigue.

- Al enviarse un caracter desde el computador, el **Sistema de Comunicación** implementado en el *FPGA* debe informar a éste que el byte correspondiente al caracter enviado lo ha recibido. Esto lo hace mediante el envío del byte 10000000 correspondiente al código ASCII del caracter Ç como se muestra a continuación.

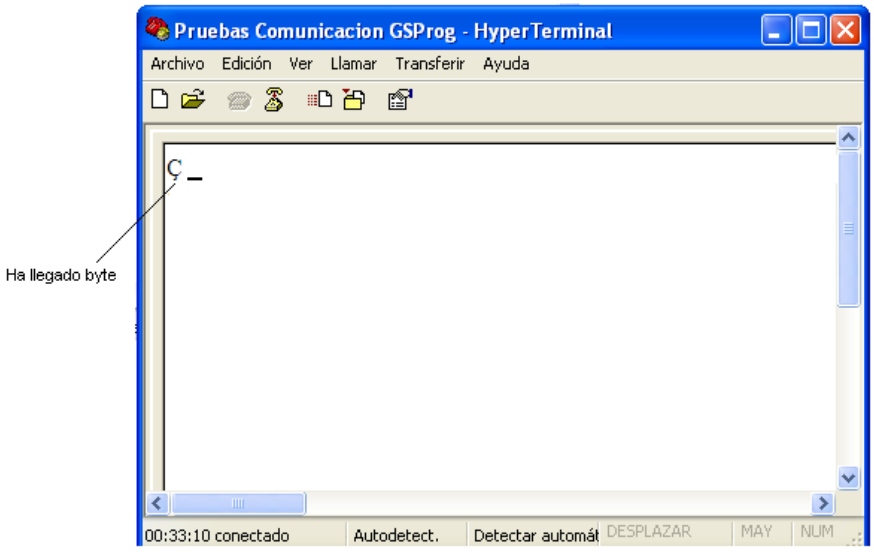


Figura 28: Envío del carácter Ç por parte del Sistema de Comunicación para informar que el carácter enviado por el computador ha llegado.

- Al realizarse el envío de las 2 tramas de 18 bytes de información, contenido de toda la información para la correcta generación de las palabras digitales, mediante el uso de carateres desde el computador, el Sistema de Comunicación implementado en el FPGA informa si los bytes han llegado de forma correcta (si las 2 tramas de 18 Bytes son iguales) mediante el envío del byte 01000010 correspondiente al código ASCII del carácter B o incorrecta (si las 2 tramas de 18 Bytes son diferentes) mediante el envío del byte 01001101 correspondiente al código ASCII del carácter M.

En la figura 29 se aprecia que el 6to byte perteneciente a la segunda trama de 18 envíos difiere del 6to byte perteneciente a la primera trama de 18 envíos, por lo cual el **Sistema de Comunicación** informa que ambas tramas son diferentes mediante el envío del carácter M.

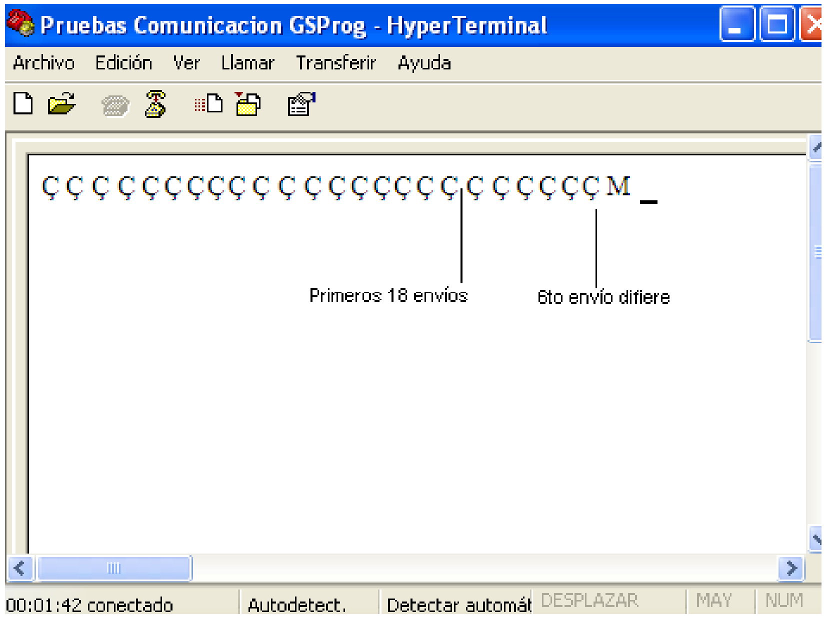


Figura 29: Envío del caracter B por parte del Sistema de Comunicación debido a que las 2 tramas de 18 bytes enviadas por el computador son iguales.

En la figura 30 se aprecia que ambas tramas de 18 bytes son iguales, por lo cual el Sistema de Comunicación informa que ambas tramas han llegado correctas mediante el envío del carácter B.

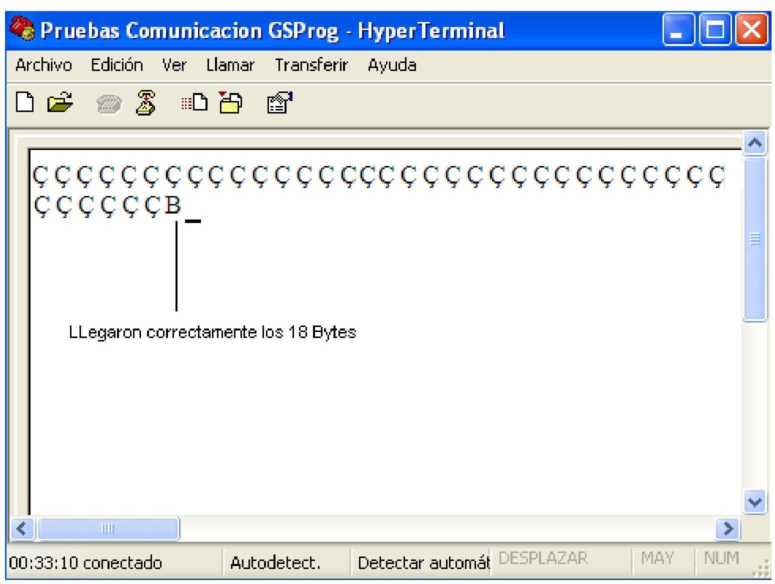


Figura 30: Envío del caracter B por parte del Sistema de Comunicación debido a que las 2 tramas de 18 bytes enviadas por el computador son iguales.

6.1.2. Conexión con interfaz realizada en Visual Basic 6.0

Esta interfaz permite saber si los datos provenientes del computador han llegado de forma correcta o incorrecta mediante la aparición de un mensaje de alerta como se muestra a continuación.

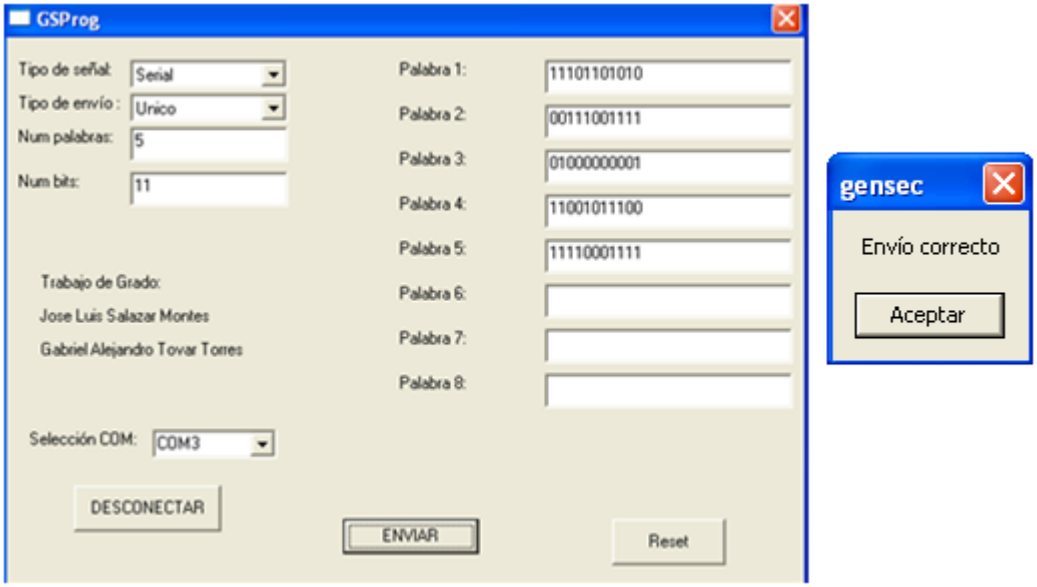


Figura 31: Mensaje de alerta entregado por la interfaz para informar que los datos han llegado de forma correcta.

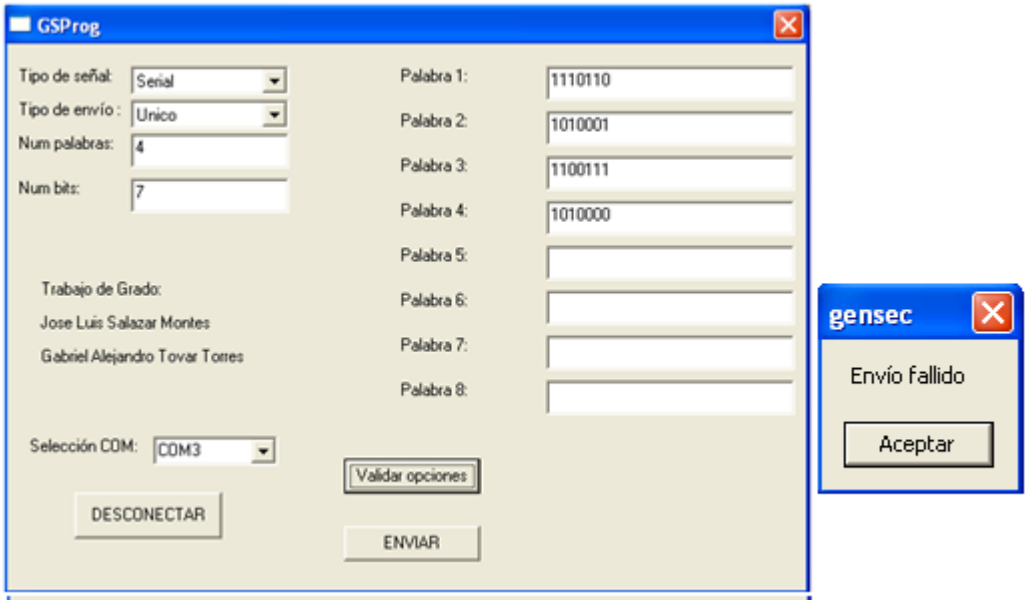


Figura 32: Mensaje de alerta entregado por la interfaz para informar que los datos han llegado de forma incorrecta.

6.2. Pruebas con palabras digitales seriales en modo *Único*

En la figura 33 se observa la generación de 2 palabras digitales seriales de 4 bits enviadas en modo *Único* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*

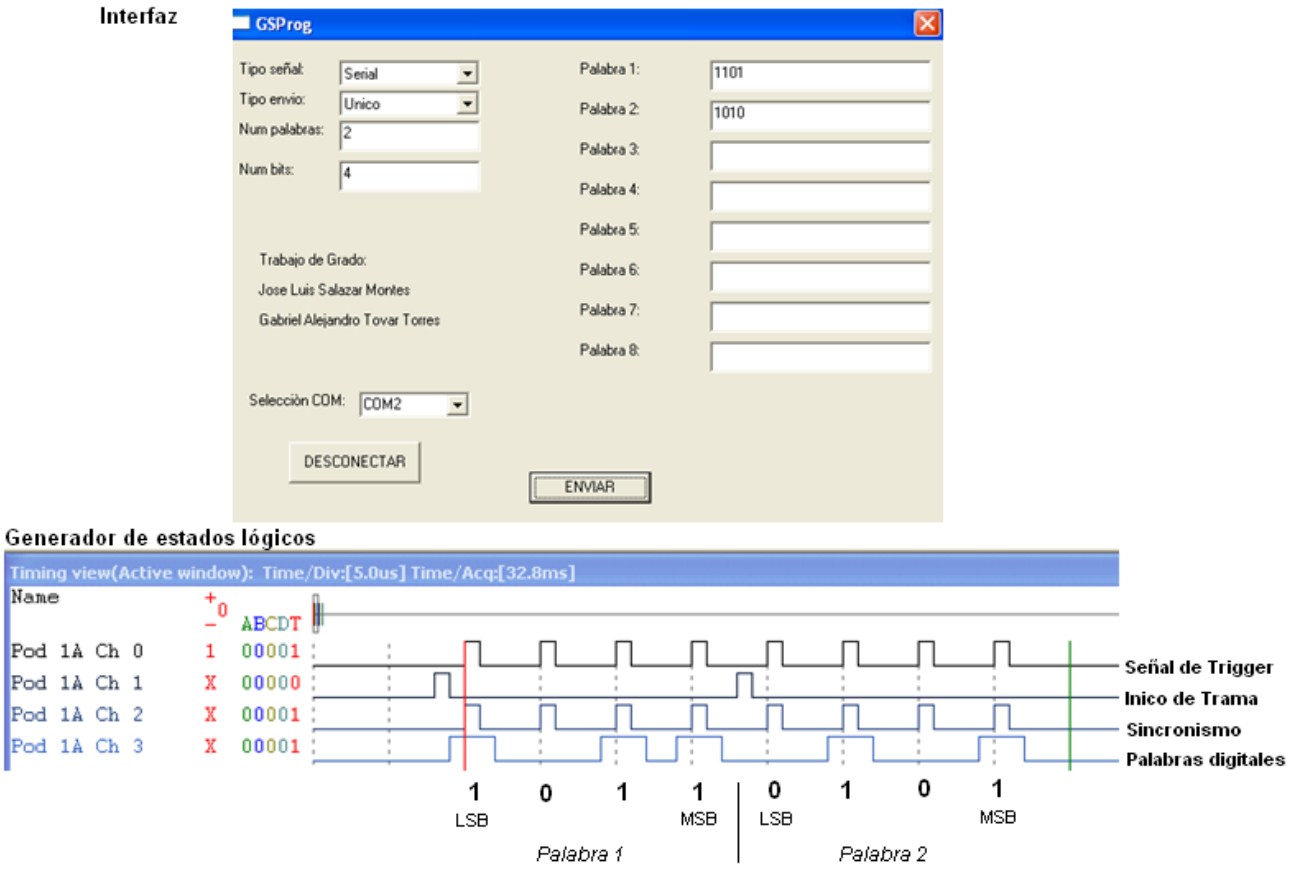


Figura 33: Envío de 2 palabras digitales seriales de 4 bits en modo Único.

En la figura 32 se observa la generación de 7 palabras digitales seriales de 5 bits enviadas en modo *Único* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*

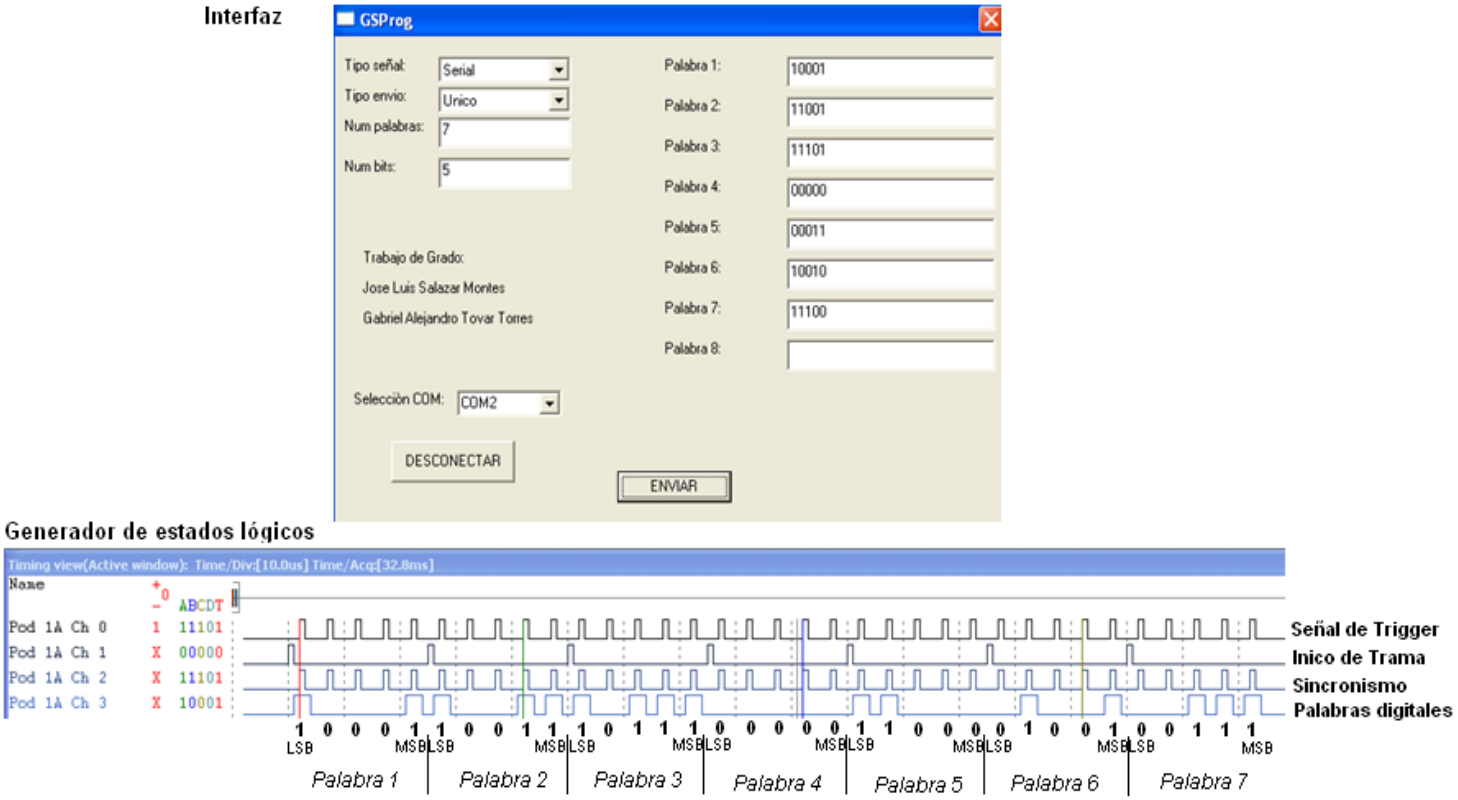


Figura 34: Envío de 7 palabras digitales seriales de 5 bits en modo Único.

Palabra 1: 10001
Palabra 2: 11001
Palabra 3: 11101
Palabra 4: 00000
Palabra 5: 00011
Palabra 6: 10010
Palabra 7: 11100

De las figuras 33 y 34 efectivamente se observa que la señal de *Inicio de Trama* se activa cada vez que se genera una nueva palabra digital y que la señal de *Sincronismo* se activa por 1us en el 2do us de cada tiempo de bit.

6.3. Pruebas con palabras digitales seriales en modo *Continuo*

- En la figura 35 se observa la generación de 2 palabras digitales seriales de 3 bits enviadas en modo *Continuo* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*

  Palabra 1: 101
  Palabra 2: 110

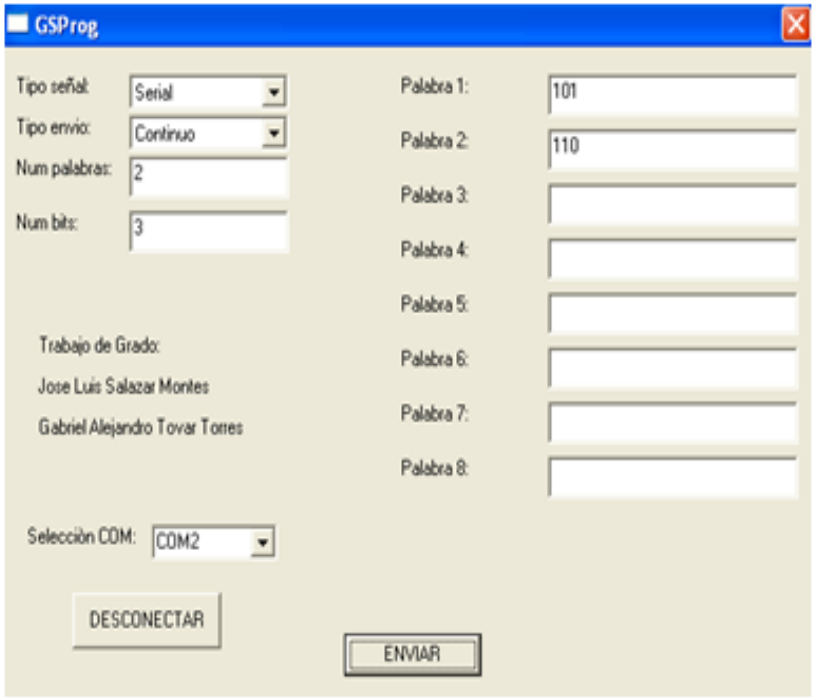
  Se observa que las 2 palabras digitales seriales son entregadas múltiples veces por el generador de manera ciclica.

- En la figura 36 se observa la generación de 8 palabras digitales seriales de 3 bits enviadas en modo *Continuo* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*

  Palabra 1: 000
  Palabra 2: 111
  Palabra 3: 001
  Palabra 4: 111
  Palabra 5: 010
  Palabra 6: 000
  Palabra 7: 000
  Palabra 8: 001

  De la figura se observa que las 8 palabras digitales seriales son entregadas múltiples veces por el generador una y otra vez.

Interfaz



Generador de estados lógicos

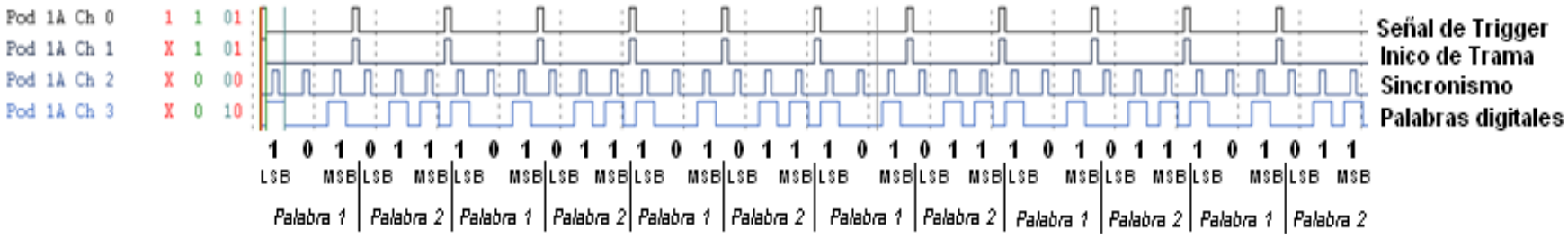


Figura 35: Envío de 2 palabras digitales seriales de 3 bits en modo Continuo.

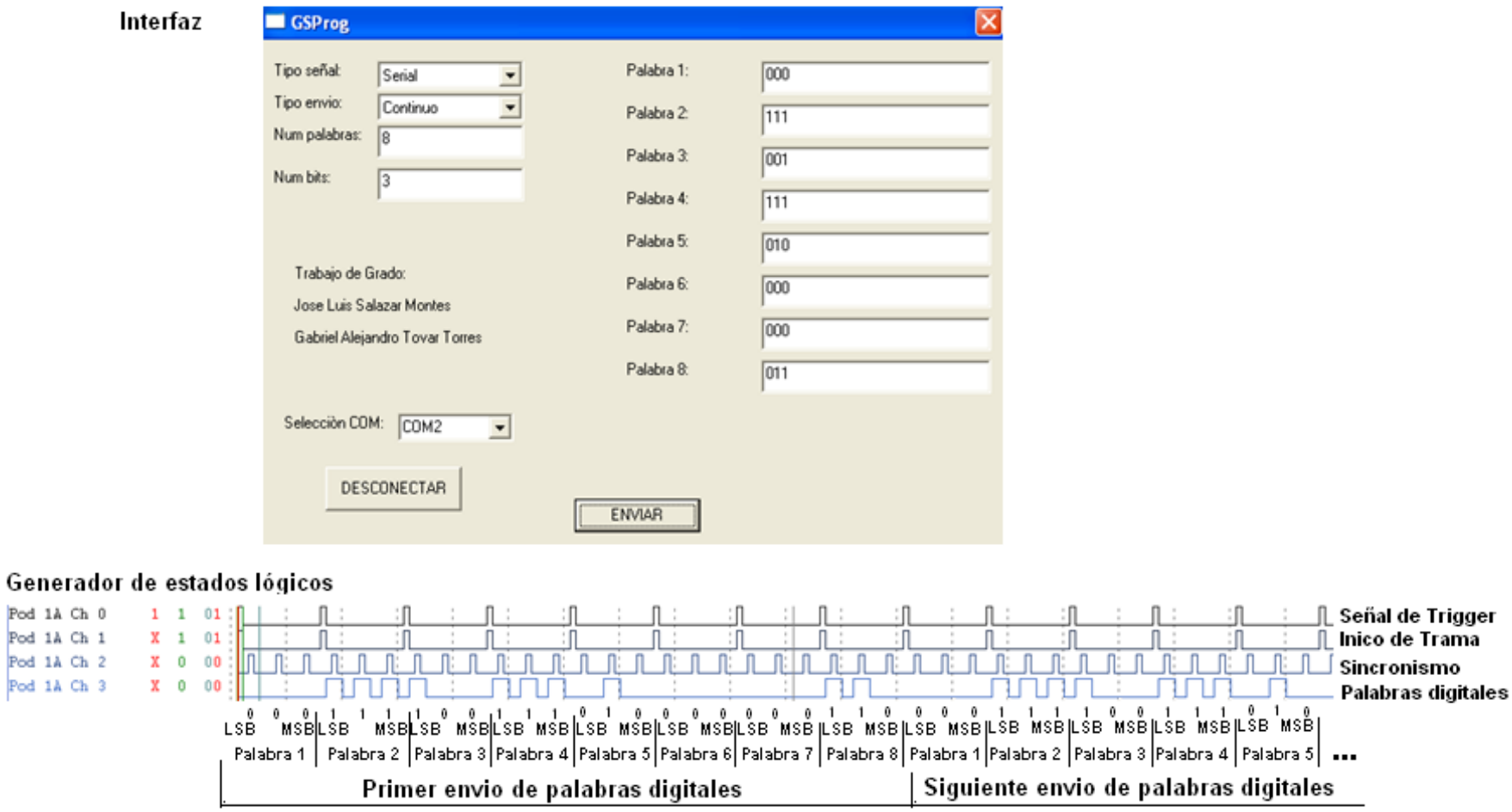


Figura 36: Envío de 8 palabras digitales serial de 3 bits en modo Continuo

6.4. Pruebas con palabras digitales paralelas en modo *Único*

En la figura 37 se observa la generación de 8 palabras digitales de 16 bits en formato paralelo enviadas en modo *Único* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*.

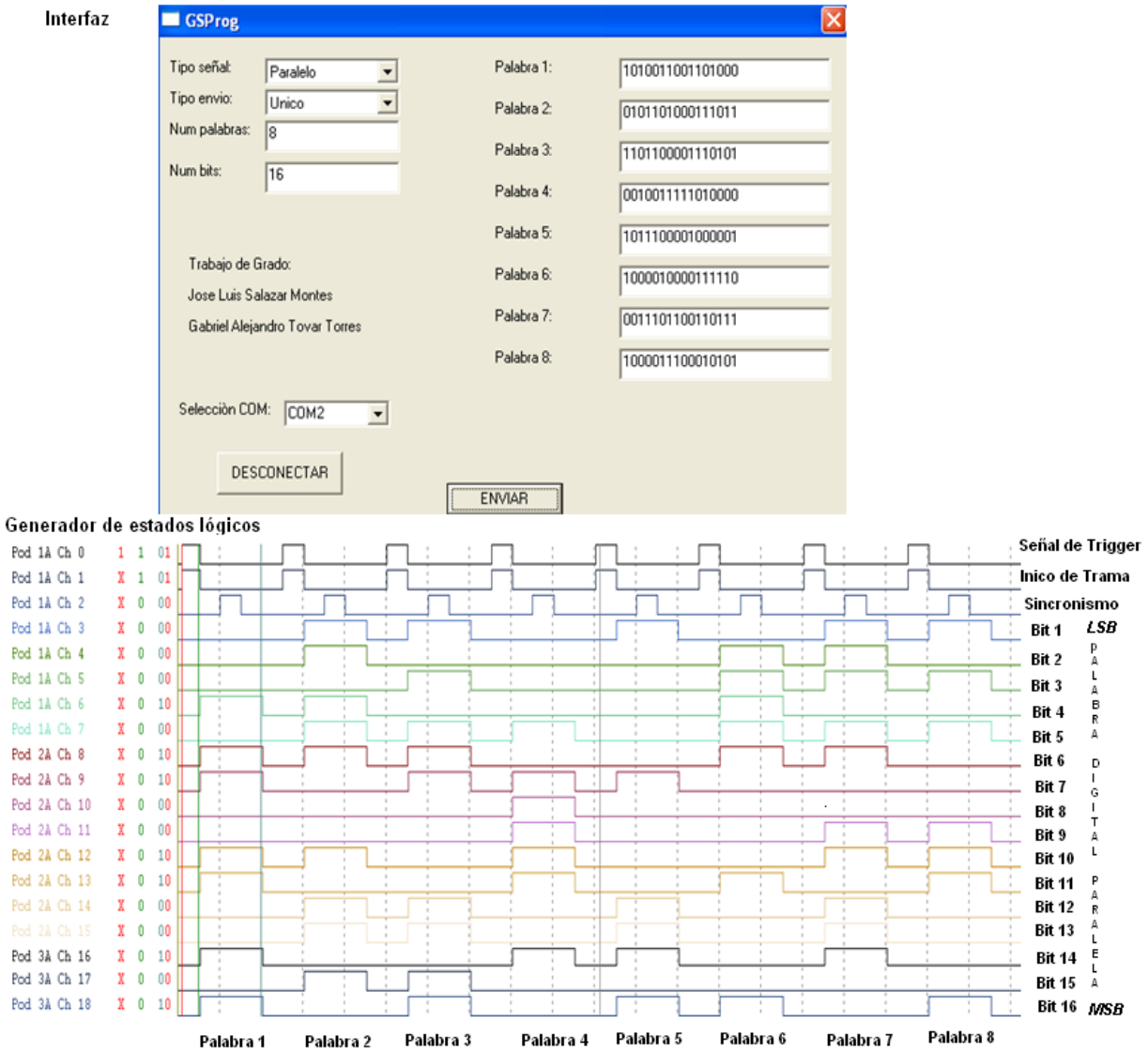


Figura 37: Envío de 8 palabras digitales en formato paralelo de 16 bits en modo Único.

En la figura 38 se observa la generación de 5 palabras digitales de 12 bits en formato paralelo enviadas en modo *Único* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*.

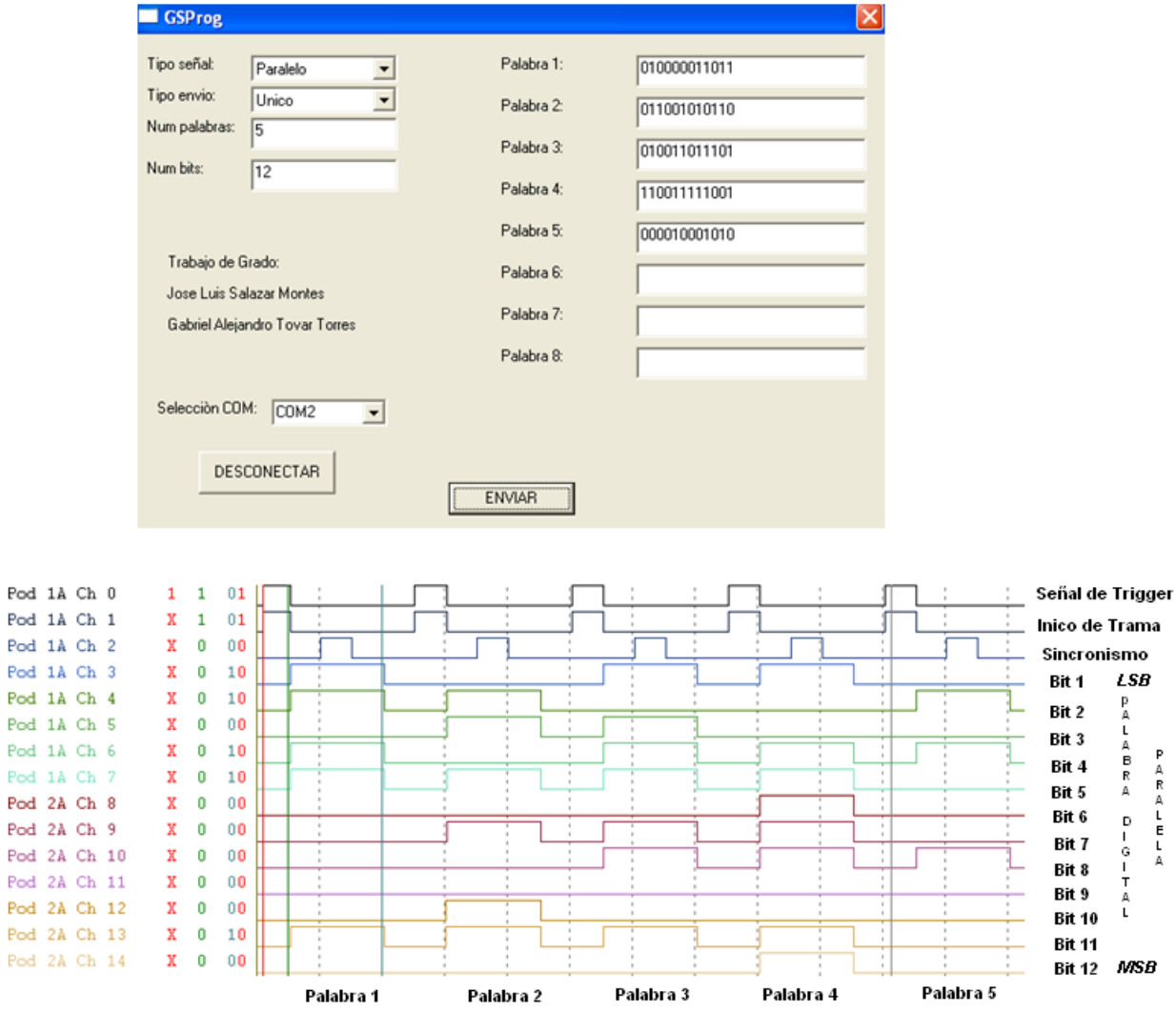
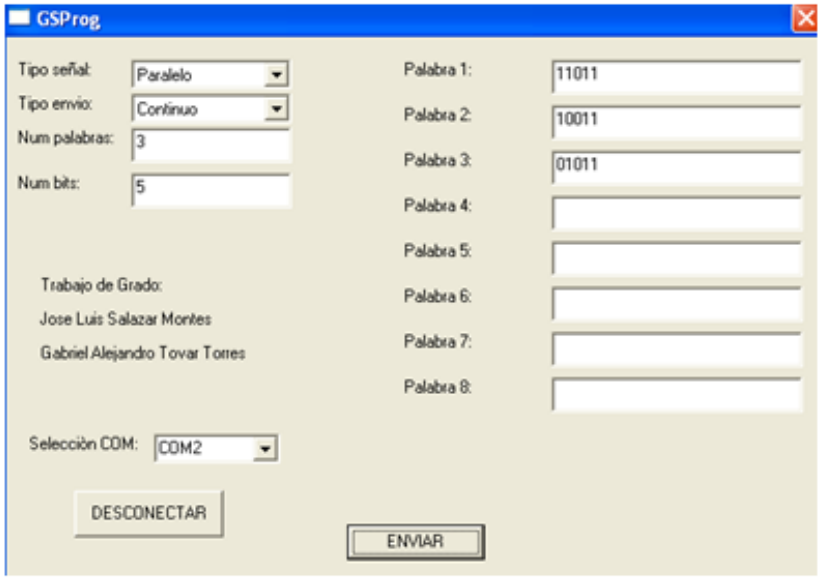


Figura 38: Envío de 5 palabras digitales en formato paralelo de 12 bits en modo Único.

En las figuras 37 y 38 efectivamente se observa que la señal de *Inicio de Trama* se activa cada vez que se genera una nueva palabra digital y que la señal de *Sincronismo* se activa por 1us en el 2do us de cada tiempo de palabra digital paralela.

6.5. Pruebas con palabras digitales paralelas en modo *Continuo*

En la figura 39 se observa la generación de 3 palabras digitales de 5 bits en formato paralelo enviadas en modo *Continuo* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*



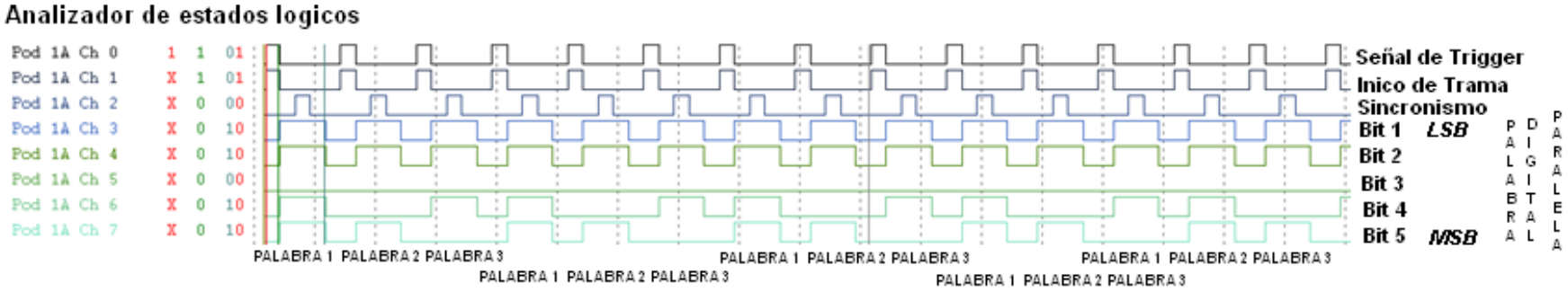


Figura 39: Envío de 3 palabras digitales en formato paralelo de 5 bits en modo Continuo.

En La figura 40 se observa la generación de 1 palabra digital de 16 bits en formato paralelo enviadas en modo *Continuo* con sus respectivas señales de *Sincronismo* e *Inicio de Trama*

Figura 40: Envío de 1 palabra digital en formato paralelo de 16 bits en modo Continuo.

Las anteriores pruebas muestran que el generador funciona conforme a las especificaciones planteadas en el **numeral 3.**

7. Conclusiones

- El ***GENERADOR DE SECUENCIAS PROGRAMADAS PARA LA VERIFICACION DE SISTEMAS DIGITALES***, ***GSprog***, es una buena opción para ser utilizado como la herramienta para probar diseños de sistemas digitales que requieran entradas de señales asíncronas, ya sean de datos o de control. En este orden de ideas, el ***GSprog*** puede proveer satisfactoriamente las diferentes señales de entrada de uso frecuente y necesarias para probar los diseños digitales realizados e implementados por los estudiantes.

- La existencia de la interfaz como medio para seleccionar lo que el usuario desea a la entrada de su sistema digital facilita la programación de de las palabras digitales que entregará el generador de forma sencilla y ordenada.

- Una vez identificadas las señales digitales más utilizadas como entradas de los diversos diseños digitales que se realizan por los estudiantes de la universidad; señales seriales entre 1 y 16 bits seriales o paralelas (buses de tamaño entre 1 y 16 bits), señales de listo dato y sincronismo se procedió a diseñar el generador desarrollado en éste trabajo de grado como un instrumento que genere éste tipo de señales.

- Al contar con un número de ocho (8) palabras digitales de dieciséis (16) bits cada una y con la posibilidad de ingresar según la necesidad del usuario el contenido de las diferentes palabras digitales requeridas por éste, el ***GSprog*** brinda una cantidad de posibilidades en cuanto a la generación de señales binarias asíncronas que satisfagan la verificación de diseños digitales que requieran entradas asíncronas al sistema.

- La existencia de las señales de *Inicio Trama* y *Sincronismo* ayuda de manera eficaz en la correcta captación de las señales binarias o datos necesarios, además el tiempo de duración de las señales de dato al ser mayor al de las señales de *Inicio Trama* y *Sincronismo*, permite una captación adecuada del valor real del dato.

- Los envíos *Único* y *Continuo* del generador permiten una secuencia o periodicidad de las señales a su salida, lo que hace posible confirmar el dato a la entrada del diseño a verificar, o determinar si la señal es captada correctamente con solo un envió.

- La interfaz desarrollada en Visual Basic 6.0 y gracias a la versión portable que se anexa permite que con solo una FPGA (a una frecuencia de 8MHz) y un computador con puerto serial se pueda hacer uso del ***GSprog***.

- El desarrollo del sistema ***GSprog***, posee una correspondencia uno a uno entre el diseño y la implementación. Es de importancia mencionar que la implementación física del diseño como tal es igual al diseño elaborado, lo que permite modificaciones del diseño original con el fin de hacer una expansión o reducción al número de palabras, al número de bits que conforman la palabra

digital o incluso a la duración del tiempo de bit de la palabra digital y a las señales de *Sincronismo* e *Inicio de Trama* si se desea.

- La implementación del ***GSprog*** en un microcontrolador u otro dispositivo es una opción que puede tenerse en cuenta al igual que el desarrollo de la interfaz grafica en otro lenguaje de programación.

- Con los respectivos ajustes al sistema de comunicación, se puede logar reemplazar el protocolo de comunicación RS232 por un USB o por un puerto paralelo.

8. Aplicaciones del GSProg.

El ***GSProg*** puede ser el instrumento que sirva como entrada a diseños digitales como:

- ✓ Sumador Full Adder de entrada paralela.
- ✓ Conversor Gray de entrada serial.
- ✓ Generador de secuencia serial de 10 bits y de entrada paralela de 16 bits, interconectado con un conversor en notación polaca.
- ✓ Negador de entrada serial.
- ✓ Inversor de entrada paralela.

A continuación se explican 2 arquitecturas de sistemas digitales que pueden recibir a sus entradas las señales provenientes del ***GSProg.***

Negador de entrada serial.

Es un sistema digital encargado de entregar a su salida la señal serial presente a su entrada en forma negada.

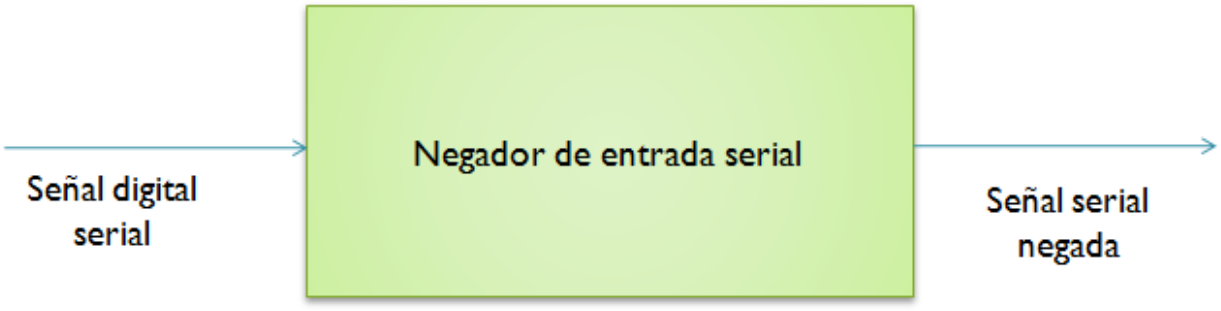


Figura 41: Diagrama general del Negador de entrada serial.

Entradas y salidas del sistema:

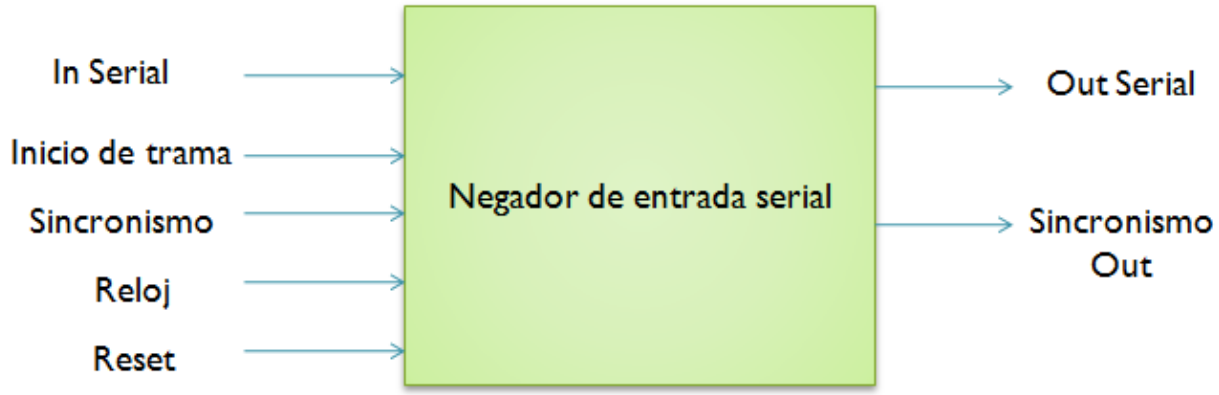


Figura 42: Entradas salidas del Negador de entrada serial.

Arquitectura del negador de entrada serial:

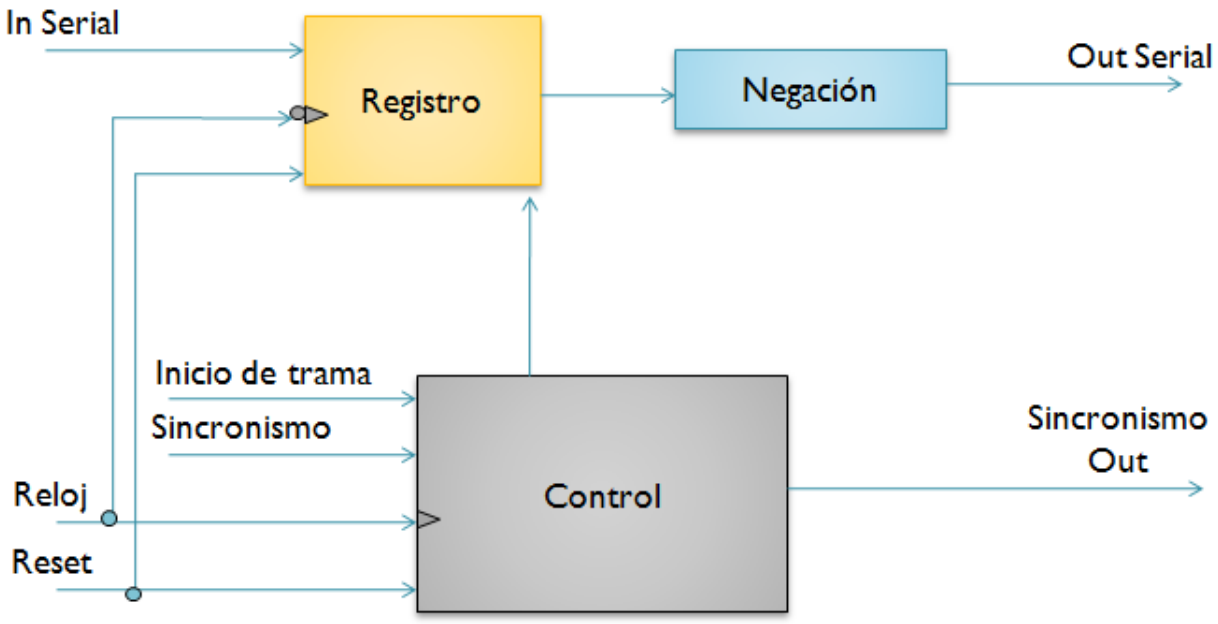


Figura 43: Arquitectura del Negador de entrada serial.

AHPL ilustrativo del negador de entrada serial que utiliza a su entrada las señales digitales asíncronas entregadas por el GSProg.

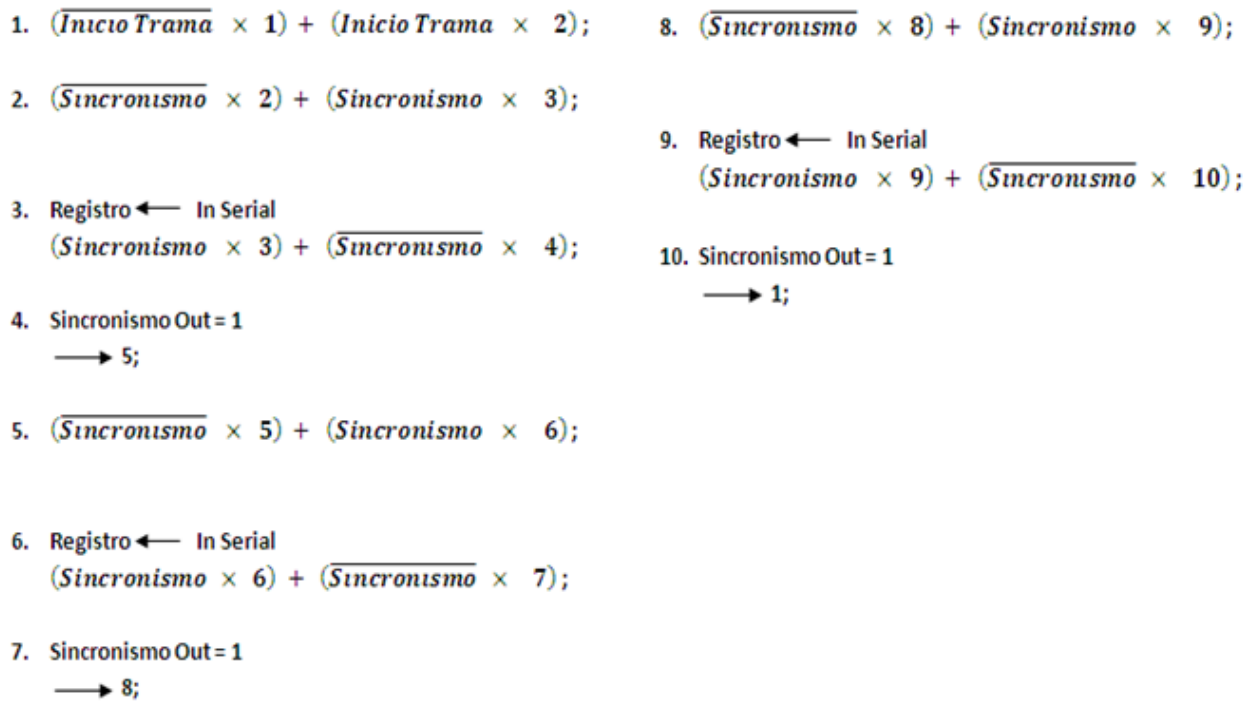


Figura 44: AHPL Negador de entrada serial.

En la figura anterior se observa que en los pasos 1 y 2 del AHPL se pregunta por las señales de *Sincronismo* e *Inicio de Trama* para la captación de los datos seriales provistos por el GSProg y recibidos por el negador de entrada serial.

A continuación se observa cómo debe ser la conexión entre el ***GSProg*** y el negador de entrada serial

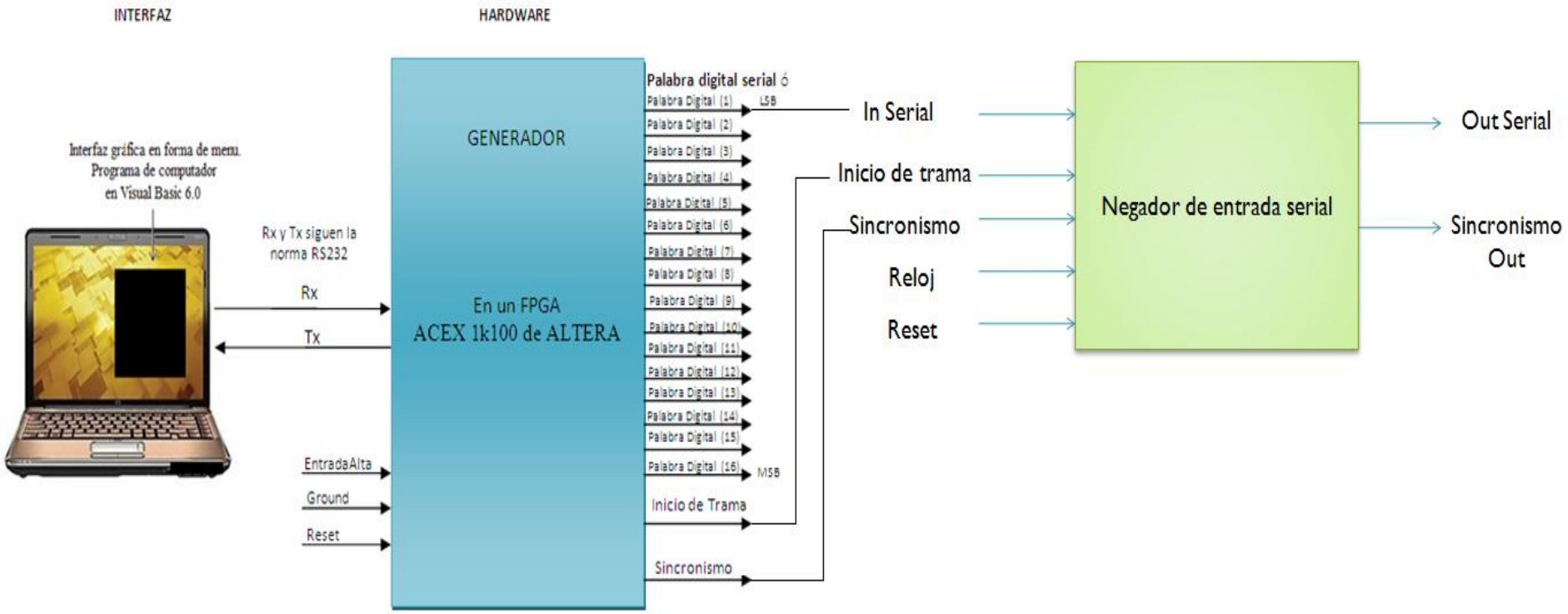


Figura 45: Conexión entre el GSProg y el Negador de entrada serial.

Inversor de entrada paralela

Es un sistema digital encargado de entregar a su salida la señal paralela de 8 bits presente a su entrada en forma invertida.

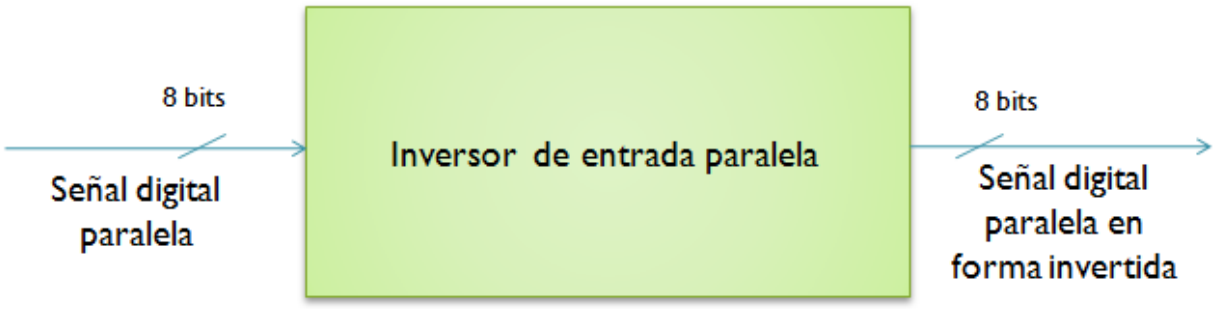


Figura 46: Diagrama general del Inversor de entrada paralela.

Entradas y salidas del sistema:

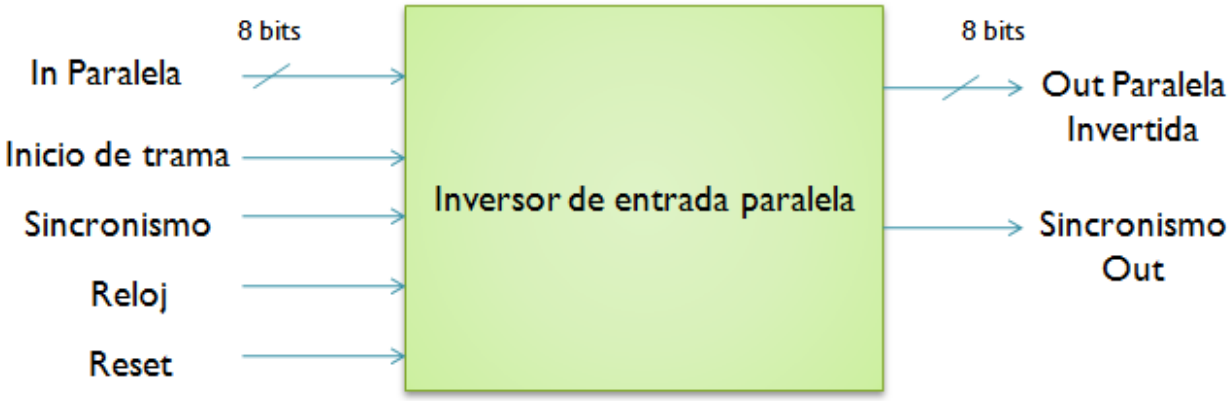


Figura 47: Entradas salidas del Inversor de entrada paralela.

Arquitectura del Inversor de entrada paralela:

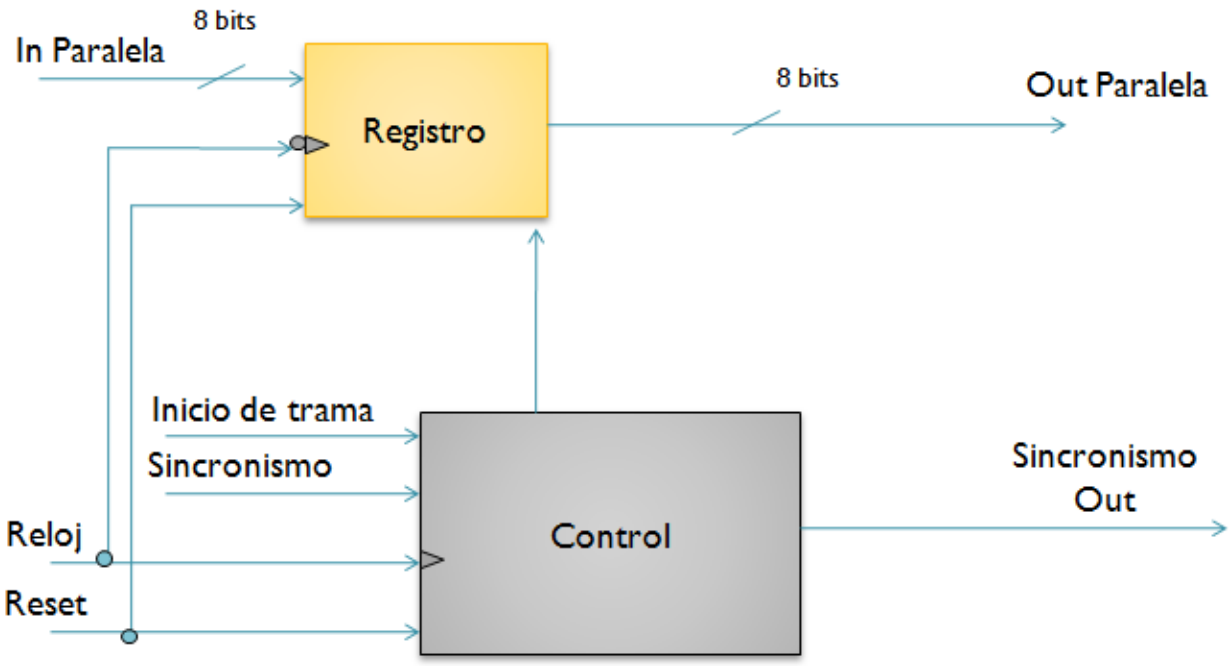


Figura 48: Arquitectura del Inversor de entrada paralela.

AHPL ilustrativo del inversor de entrada paralela que utiliza a su entrada las señales digitales asíncronas entregadas por el GSProg.

1. $(\overline{Inicio\ Trama} \times 1) + (Inicio\ Trama \times 2);$

2. $(\overline{Sincronismo} \times 2) + (Sincronismo \times 3);$

3. Registro ← In Paralelo
   $(Sincronismo \times 3) + (\overline{Sincronismo} \times 4);$

4. Sincronismo Out = 1
   ⟶ 1;

Figura 49: AHPL Inversor de entrada paralela.

En la figura anterior se observa que en los pasos 1 y 2 del AHPL se pregunta por las señales de *Sincronismo* e *Inicio de Trama* para la captación de los datos seriales provistos por el GSProg y recibidos por el inversor de entrada paralela.

A continuación se observa cómo debe ser la conexión entre el ***GSProg*** y el inversor de entrada paralela.

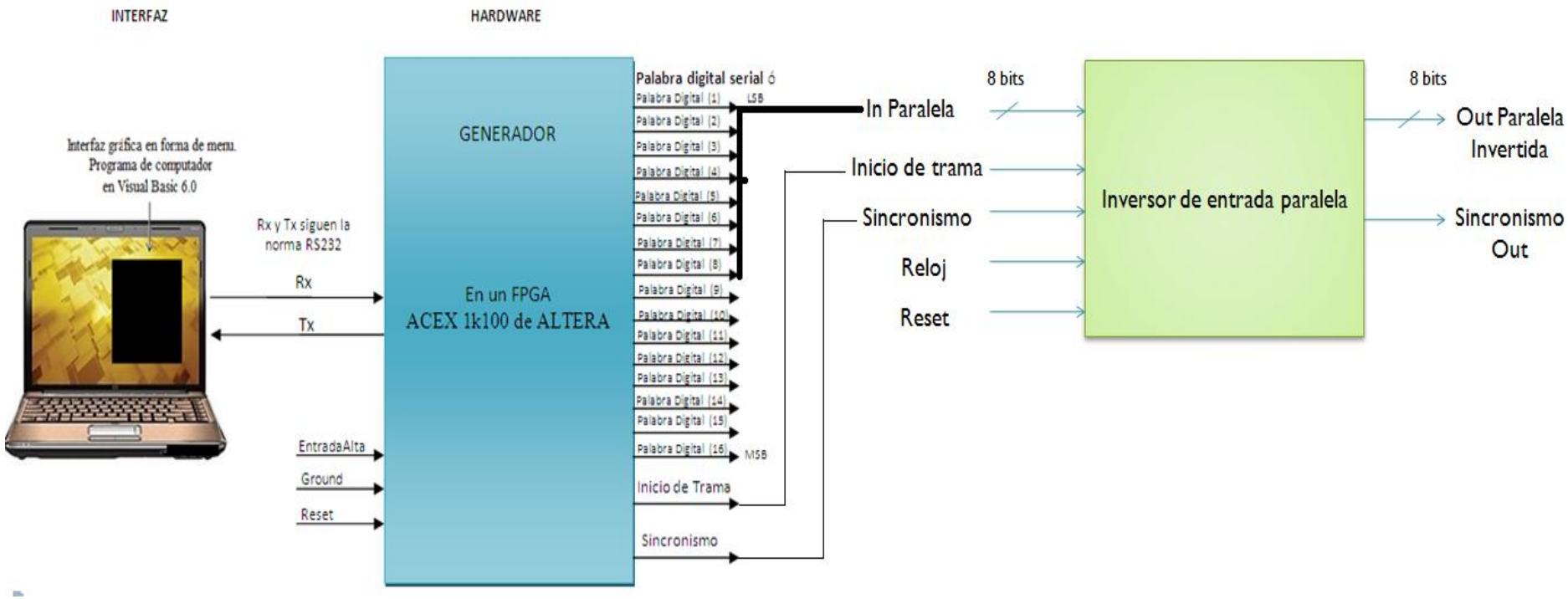


Figura 50: Conexión entre el GSProg y el Inversor de entrada paralela.

9. Bibliografía y fuentes de información

[1] MORRIS, Mano. Diseño Digital. Prentice Hall. Tercera Edición.

[2] Wakerly. Diseño Digital Principios y Prácticas. Prentice Hall México. Sexta Edición.

[3] Floyd. T.L. Fundamentos de Sistemas Digitales. Ed. Prentice Hall.Novena Edición.

[4] Hall & Peterson. Arquitectura de Computadores. Prentice Hall.

[5] Saenz y Torres. Electrónica Digital, teoría y problemas.

[6] Uribe Nicolás. Trabajo De Grado Panel Electrónico Programable. Pontificia Universidad Javeriana, 2009.

[7] Solano Johana María y Solano Rodriguez Diana Carolina. Módulos hardware multipropósito para la implementación de sistemas digitales: DIGIMOD V1.0. Pontificia Universidad Javeriana, 2009.

[8] Imagen de Transmisión de datos en forma serial asíncrona con los niveles de voltaje adecuados que utiliza un FPGA. Disponible en internet en la dirección:

http://www.rootshell.be/~wcruzy/cd/tutorialserialrs232.pdf

[9] Imagen de Disposición de pines del conector RS232D de tipo DB9. Disponible en internet en la dirección:

http://www.euskalnet.net/shizuka/rs232.htm

[10] Norma RS232. Disponible en internet en la dirección:

http://juandeg.tripod.com/rs232.htm

10. ANEXOS

8.1 Anexo 1 AHPL Sistema de Comunicación

C1: ContadortiempoBit<-0000000000;
ContadorByte<-000000;
Contador 9 Bits <- 0000;
RegistroLLegaMal<-010011010; (M) //*Contadores y registros en su estado inicial*//
RegistroLLegaBien<-010000100; (B)
RegistroLLegaByte<-100000000; (Ç)
RegistroA<-"0";
RegistroB<-"0";

(Rx X C1) + ($\overline{Rx}$ X C31)

C2: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Bit de start*//
(TBit_OnesHotSiscom X C2) + (TBit_OnesHotSiscom X C3)

C3: RegA1[1]<-Rx

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C4

C4: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //* Primer tiempo de bit*//
(TBit_OnesHotSiscom X C4) + (TBit_OnesHotSiscom X C5)

C5: RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C6

C6: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //* Segundo tiempo de bit*//
(TBit_OnesHotSiscom X C6) + (TBit_OnesHotSiscom X C7)

C7: RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C8

C8: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Tercer tiempo de bit*//
(TBit_OnesHotSiscom X C8) + (TBit_OnesHotSiscom X C9)

C9: RegA1[4]<-RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C10

C10: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Cuarto tiempo de bit*//
(TBit_OnesHotSiscom X C10) + (TBit_OnesHotSiscom X C11)

C11: RegA1[5]<-RegA1[4]<-RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C12

C12: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Quinto tiempo de bit*//
(TBit_OnesHotSiscom X C12) + (TBit_OnesHotSiscom X C13)

C13: RegA1[6]<-RegA1[5]<-RegA1[4]<-RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C14

C14: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Sexto tiempo de bit*//
(TBit_OnesHotSiscom X C14) + (TBit_OnesHotSiscom X C15)

C15: RegA1[7]<-RegA1[6]<-RegA1[5]<-RegA1[4]<-RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C16

C16: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //* Séptimo tiempo de bit*//
(TBit_OnesHotSiscom X C16) + (TBit_OnesHotSiscom X C17)

C17: RegA1[8]<-RegA1[7]<-RegA1[6]<-RegA1[5]<-RegA1[4]<-RegA1[3]<-RegA1[2]<-RegA1[1]<-Rx;

Contador Tiempo Bit <- Contador Tiempo Bit + 1;
→ C18

C18: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //* Octavo tiempo de bit*//
(TBit_OnesHotSiscom X C18) + (TBit_OnesHotSiscom X C19)

C19: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //*Bit de parada*//
(TBit_OnesHotSiscom X C19) + (TBit_OnesHotSiscom Λ ContByte_OnesHotSiscom X C20) + (TBit_OnesHotSiscom Λ ContByte_OnesHotSiscom X C21)

C20: RegB18<-RegB17<-RegB16<-RegB15<- RegB14<-RegB13<-RegB12<-RegB11<-RegB10<-RegB9<-

RegB8<-RegB7<-RegB6<-RegB5<-RegB4<- RegB3<-RegB2<-RegB1<-RegA18<- RegA17<-
RegA16<-
Rega15<-RegA14<- RegA13<- RegA12<- RegA11<-RegA10<-RegA9<-RegA8<- RegA7<-
RegA6<-
RegA5<- RegA4<- RegA3<- RegA2<- RegA1;

ContadorByte<-ContadorByte + 1; //*Desplazo lo que tengo en el RegA1 a los otros registros*//
Contador tiempo bit <- 0000000000;
→ C23

C21 //*Esperar señal para decidir qué hacer*//
$\overline{\text{(Comparar\_ControlSiscom}}$ X C24) + (Comparar_ControlSiscom X C25)

C22 OnesHot_EnableRBL=1; //*habilito el registros para hacer el corrimiento de dato y espero señal*//
OnesHot_STX[0]=0
OnesHot_STX[1]=1 //* para el caso llegò byte*//

Contador 9 Bits <- Contador 9 Bit +1; //*Incremento el contador de los 9 bits
Contador Tiempo Bit <- Contador Tiempo bit +1; y el del tiempo de bit*//

→ C23

C23: OnesHot_STX[0]=0
OnesHot_STX[1]=1 //*Salida de Tx para llegó byte*//

Contador Tiempo Bit <- Contador Tiempo Bit + 1;

$\overline{\text{(TBIt\_OnesHotSiscom}}$ X C23) + (TBit_OnesHotSiscom Λ $\overline{\text{Contador9Bits\_OnesHotSiscom}}$ X C22)
(TBit_OnesHotSiscom Λ contador9Bits_OnesHotSiscom X C28)

C24: OnesHot_STX[0]=1
OnesHot_STX[1]=1 //*Salida de Tx para Llegó mal la ráfaga de Bytes*//

Contador Tiempo Bit <- Contador Tiempo Bit + 1;

$\overline{\text{(TBIt\_OnesHotSiscom}}$ X C24) + (TBit_OnesHotSiscom Λ $\overline{\text{Contador9Bits\_OnesHotSiscom}}$ X C29)
+ (TBit_OnesHotSiscom Λ Contador9Bits_OnesHotSiscom X C1)

C25: OnesHot_STX[0]=1
OnesHot_STX[1]=0 //*Salida de Tx para llegó bien la ráfaga de bytes*//

Contador Tiempo Bit <- Contador Tiempo Bit + 1;

$\overline{\text{(TBIt\_OnesHotSiscom}}$ X C25) + (TBit_OnesHotSiscom $\overline{\text{Λ Contador9Bits\_OnesHotSiscom}}$ X C25)
+ (TBit_OnesHotSiscom Λ Contador9Bits_OnesHotSiscom X C26)

C26: ListoDato=1;

→ C27

C27: NULL

C28: ContadortiempoBit<-0000000000;
Contador 9 Bits <- 0000;
RegistroLLegaByte<-100000000; (Ç) //*Espero por la llegada del otro byte*//

(Rx X C28) + ($\overline{Rx}$ X 31)

C29: OnesHot_EnableRLLM=1;
OnesHot_STX[0]=1 //* para el caso llego mal*//
OnesHot_STX[1]=1

Contador 9 Bits <- Contador 9 Bit +1; //*Incremento el contador de los 9 bits
Contador Tiempo Bit <- Contador Tiempo bit +1; y el del tiempo de bit*//

→ C24

C30: OnesHot_EnableRLLB=1;
OnesHot_STX[0]=1 //* Para el caso llego mal*//
OnesHot_STX[1]=0

Contador 9 Bits <- Contador 9 Bit +1; //*Incremento el contador de los 9 bits
Contador Tiempo Bit <- Contador Tiempo bit +1; y el del tiempo de bit*//

→ C25

C31: Contador Tiempo Bit <- Contador Tiempo Bit + 1; //* Se cuenta 414 pulsos de reloj con el fin de garantizar la captura correcta del dato*//

$\overline{\text{TBit\_OnesHotSiscom414}}$ X C31) + (TBit_OnesHotSiscom414 X C32)

C32: ContadortiempoBit<-0000000000; //*Se coloca el contador en su estado inicial*//

→ C2

8.2 Anexo 2 AHPL Unidad de Control

C1:
$\overline{\text{Listo dato}}$ X C1) + (Listo Dato X C2)

C2: EmpezarControl_ControlGS=1;
→ C3

C3: NULL

8.3 Anexo 3 AHPL Generador de Secuencia

C1: Contador #Bits<- 0000;
Contador #Palabras<- 000; //* Inicializando contadores y esperando la señal de inicio*//
Contador8pulsos1us<- 000;
Maquina estados selector serial<- 0000000;
Maquina estados selector paralelo<-000;

($\overline{\text{Empezar\_ControlGS}}$ X C1) + (Empezar_ControlGS Λ $\overline{\text{SeñalOpciones\_OnesHot}}$[8] X C2) +
(Empezar_ControlGS Λ SeñalOpciones_OnesHot[8] X C13)

//*SE EMPIEZA EL PROCESO PARA CUANDO LAS PALABRAS DIGITALES SON SERIALES*//

C2: Iniciotrama=1;
Sincronismo=0; //*Generando el primer inicio trama*//
ControlGS_Salida=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;

($\overline{\text{Contador8pulso\_OnesHot}}$ X C2) + (Contador8pulsos_OnesHot X C3)

C3: Iniciotrama=0;
Sincronismo=0; //*1er us del bit de la palabra*//
ControlGS_Salida=0;
ControlGS_DatoDescansoSerial=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

($\overline{\text{Contador8pulso\_OnesHot}}$ X C3) + (Contador8pulsos_OnesHot X C4)

C4: Iniciotrama=0;
Sincronismo=1; //*2do us del bit de la palabra y sincronismo*//
ControlGS_Salida=0;
ControlGS_DatoDescansoSerial=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

($\overline{\text{Contador8pulso\_OnesHot}}$ X C4) + (Contador8pulsos_OnesHot X C5)

C5: Iniciotrama=0;
Sincronismo=0; //*3er us del bit de la palabra*//
ControlGS_Salida=0;
ControlGS_DatoDescansoSerial=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulso\_OnesHot}}$ X C5) + (Contador8pulsos_OnesHot X C6)

C6: Iniciotrama=0;
Sincronismo=0; //*1er us del tiempo de descanso *//
ControlGS_Salida=0;
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulso\_OnesHot}}$ X C6) + (Contador8pulso_OnesHot Λ $\overline{\text{NumBits\_OnesHot}}$ X C9) +
(Contador8pulso_OnesHot Λ NumBits_OnesHot Λ $\overline{\text{NumPal\_OnesHot}}$ X C7) +
(Contador8pulso_OnesHot Λ NumBits_OnesHot Λ NumPal_$\overline{\text{OnesHot}}$ Λ $\overline{\text{SeñalOpciones\_Oneshot}}$[9] X C11) +
(Contador8pulso_OnesHot Λ NumBits_OnesHot Λ NumPal_OnesHot Λ SeñalOpciones_Oneshot[9] X C7)

C7: Iniciotrama=1;
Sincronismo=0; //*2do us del tiempo de descanso con inicio trama. Se acaban bits de
ControlGS_Salida=0; la palabra digital*//
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador7pulsos\_OnesHot}}$ X C7) + (Contador7pulsos_OnesHot Λ $\overline{\text{NumPal\_OnesHOt}}$ X C8) +
(Contador7pulsos_OnesHot Λ NumPal_OnesHOt X C12)

C8: Iniciotrama=1;
Sincronismo=0; //*Completando el 2do us del tiempo de descanso con inicio trama.
ControlGS_Salida=0; Se acaban los bits de la palabra digital*//
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;
Maquina Estados Selector Serial<- INC (Maquina Estados Selector Serial);
Contador #Bits<- 0000;
Contador #Palabras<- Contador #Palabras + 1;

→ C3

C9: Iniciotrama=0;
Sincronismo=0; //* 2do us del tiempo de descanso con inicio trama. No se ha acabado
ControlGS_Salida=0; los bits que conforman la palabra digital*//
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{(Contador7pulsos\_OnesHot}}$ X C9) + (Contador7pulsos_OnesHot X C10)

C10: Iniciotrama=0;
Sincronismo=0; //* Completando 2do us del tiempo de descanso. No se ha acabado los
ControlGS_Salida=0; bits que conforman la palabra digital*//
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;
Maquina Estados Selector Serial<- INC (Maquina Estados Selector Serial);
Contador #Bits<- Contador #Bits +1;

⟶ C3

C11: Iniciotrama=0;
Sincronismo=0; //* 2do us del tiempo de descanso. Se Acabaron palabras digitales*//
ControlGS_Salida=0;
ControlGS_DatoDescansoSerial=0;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{(Contador8pulsos\_OnesHot}}$ X C11) + (Contador8pulsos_OnesHot X C22)

C12: Iniciotrama=1;
Sincronismo=0; //*Finalizo el tiempo de bit. Se Completa palabras digitales y se ponen
ControlGS_Salida=0; contadores a cero. Se repite todo otra vez porque es continuo*//
ControlGS_DatoDescansoSerial=0;

Contador8pilsos1us<- Contador8pulsos1us + 1;
Maquina Estados Selector Serial<-0000000;
Contador #Bits<- 0000;
Contador #Palabras<- 000;

⟶ C3;

//*SE FINALIZA PROCESO DE LAS PALABRAS DIGITALES SERIALES*//

//*SE EMPIEZA PROCESO PARA CUANDO LAS PALABRAS DIGITALES SON PARALELAS*//

C13: Iniciotrama=1;
Sincronismo=0; //*Generar el primer Inicio trama*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{(Contador8pulsos\_OnesHot}}$ X C13) + (Contador8pulsos_OnesHot X C14)

C14: ControlGS_Dato DescansoParalelo=1;
Iniciotrama=0;
Sincronismo=0; //*1er us de dato*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulsos\_OnesHot}}$ X C14) + (Contador8pulsos_OnesHot X C15)

C15: ControlGS_Dato DescansoParalelo=1;
Iniciotrama=0;
Sincronismo=1; //*2do us de dato y sincronismo*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulsos\_OnesHot}}$ X C15) + (Contador8pulsos_OnesHot X C16)

C16: ControlGS_Dato DescansoParalelo=1;
Iniciotrama=0;
Sincronismo=0; //*3er us de dato*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulsos\_OnesHot}}$ X C16) + (Contador8pulsos_OnesHot X C17)

C17: ControlGS_Dato DescansoParalelo=0;
Iniciotrama=0;
Sincronismo=0; //*1er us de tiempo de descanso*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulsos\_OnesHot}}$ X C17) + (Contador8pulsos_OnesHot Λ $\overline{\text{NumPal\_OnesHot}}$ X C18) +
(Contador8pulsos_OnesHot Λ NumPal_OnesHot Λ $\overline{\text{SeñalOpciones\_OnesHot}}$[9] X C20)+
(Contador8pulsos_OnesHot Λ NumPal_OnesHot Λ SeñalOpciones_OnesHot[9] X C18)

C18: ControlGS_Dato DescansoParalelo=0;
Iniciotrama=1;
Sincronismo=0; //*2do us de tiempo de descanso con inicio trama*//
ControlGS_Salida=1;

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador7pulsos\_OnesHot}}$ X C18) + (Contador7pulsos_OnesHot Λ $\overline{\text{NumPal\_Ones Hot}}$ X C19) +
(Contador7pulsos_OnesHot Λ NumPal_Ones Hot X C21)

C19: ControlGS_Dato DescansoParalelo=0;
Iniciotrama=1;
Sincronismo=0; //*Completando 2do us de tiempo de descanso con inicio trama.
ControlGS_Salida=1; No se han acabado las palabras digitales*//

Contador8pulsos1us<- Contador8pulsos1us + 1;
Maquina Estados Selector Serial <- Maquina Estados Selector Serial + 1;
Contador #Palabras <- Contador #Palabras +1;

→ C14

C20: ControlGS_Dato DescansoParalelo=0;
Iniciotrama=0;
Sincronismo=0; //*2do us de tiempo de descanso y se terminaron palabras
ControlGS_Salida=1; digitales paralelas*//

Contador8pulsos1us<- Contador8pulsos1us + 1;

$\overline{\text{Contador8pulsos\_OnesHot}}$ X C20) + (Contador8pulsos_OnesHot X C22)

C21: ControlGS_Dato DescansoParalelo=0;
Iniciotrama=1; //*Completando 2do us de tiempo de descanso con inicio trama. Se
Sincronismo=0; acabaron palabras digitales pero como es continuo, se inicializan
ControlGS_Salida=1; Contadores y reinicio el proceso*//

Contador8pulsos1us<- Contador8pulsos1us + 1;
Maquina Estados Selector Paralelo <- 000;
Contador #Palabras <- 000;

→ C14

//* SE FINALIZA PROCESO DE LAS PALABRAS DIGITALES PARALELAS*//

C22: NULL

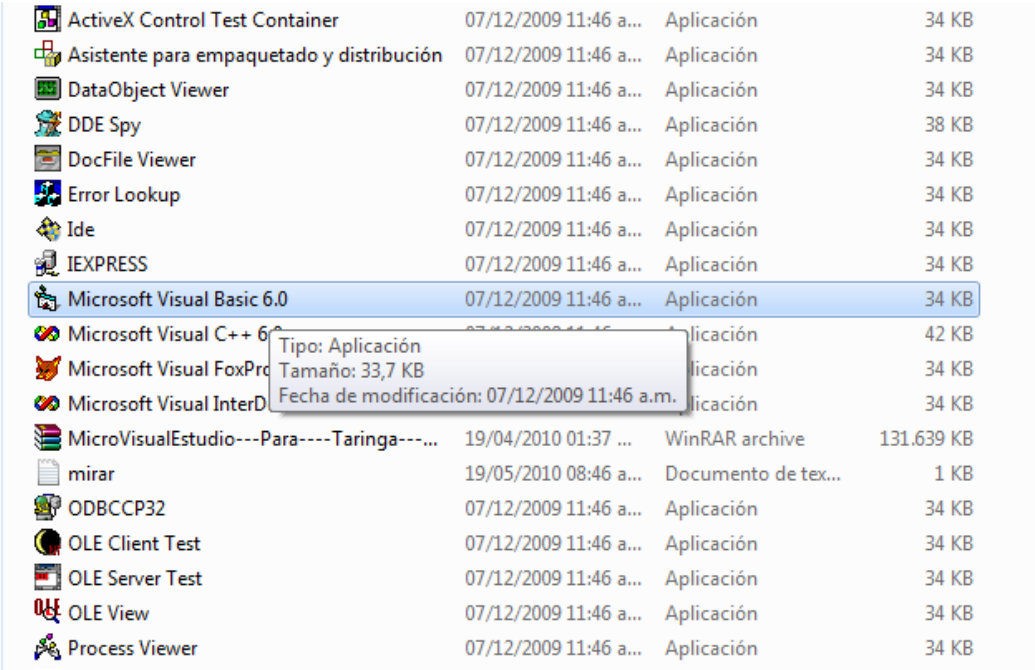
8.4 Anexo 4 Manual de usuario del GSProg

MANUAL DEL USUARIO

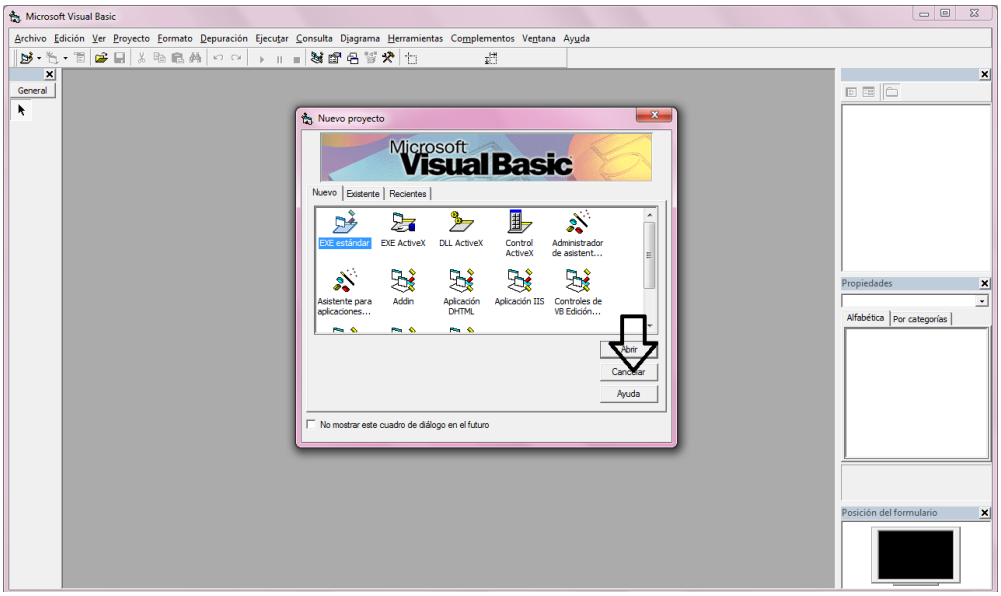
1. Programar en el FPGA el generador de secuencia.
2. Abrir el Visual Basic 6.0 o su versión portable.
3. Conectar el FPGA con el computador, usando el conector DB9.
4. Abrir el proyecto que contiene la interfaz grafica (identificado con el nombre de “interfaz visual”).
5. Escoja las opciones que desee en la interfaz.
6. Antes de enviar reinicie el generador en la FPGA, usando el botón de “reset”
7. En caso de que salga un aviso diferente al de “Envío correcto”, antes de volver a enviar es recomendable reiniciar el generador (usando el botón de “reset”) y la interfaz grafica. Si no se hace y ha cambiado alguna opción previa en la interfaz lo que se envía al generador no va a coincidir con lo que se escoge.
8. Después de cada envío desde la interfaz grafica (fallido o correcto) y si desea realizar un nuevo envío, ya sea con las mismas opciones o con opciones diferentes, DEBE REINICIAR EL GENERADOR (usando el botón de “reset”)
9. Siempre revise la conexión, un error en el envío puede deberse a mala conexión entre el FPGA y el computador

FORMA DE MANEJAR LA INTERFAZ GRAFICA

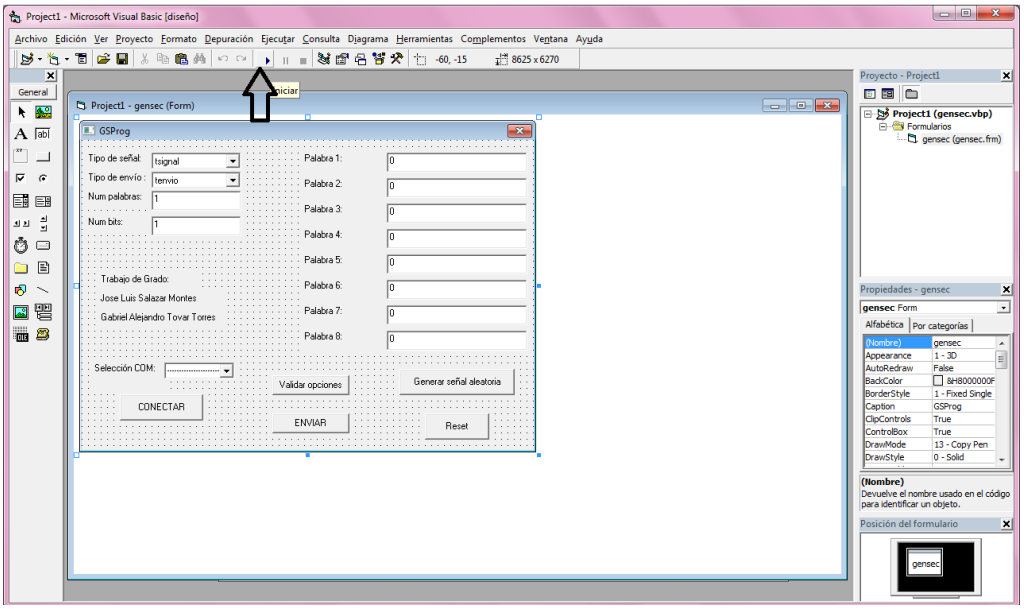
Abrir el Visual Basic 6.0



Al abrir el programa escogemos la opción de cancelar.

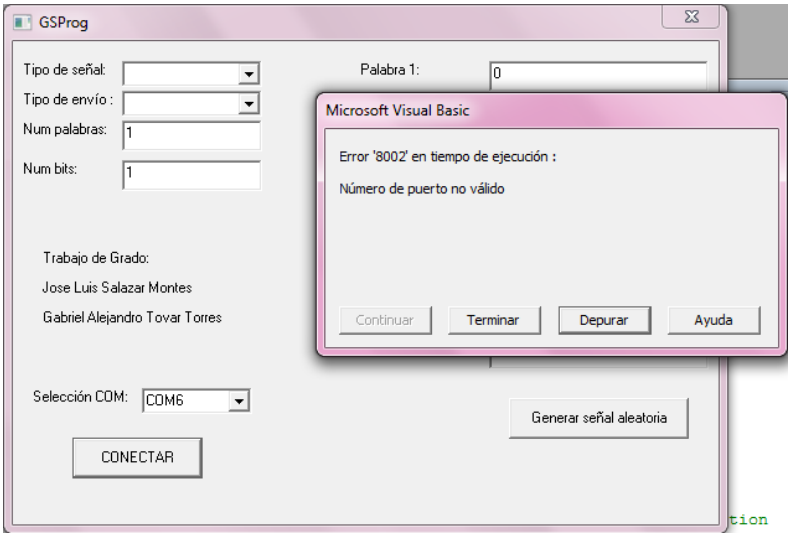


Le damos archivo, abrir proyecto y buscamos la ubicación de "interfaz visual". Una vez abierto el proyecto le damos la opción de iniciar.

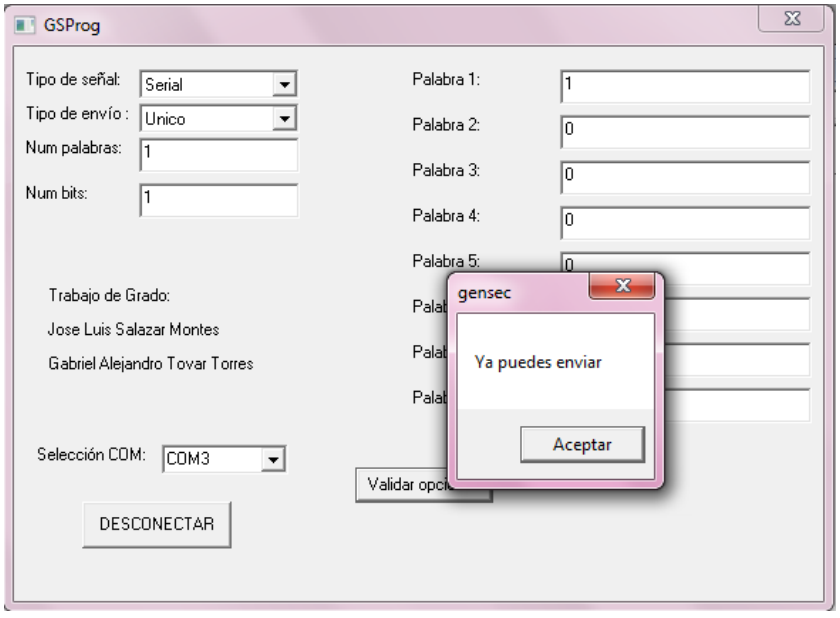


Procedemos a escoger las opciones que deseamos a la salida del generador. Es muy importante saber porque puerto del computador vamos a hacer la conexión con el FPGA.

La interfaz está diseñada para poder escoger entre 6 puertos seriales del computador aunque no existan en el computador. Si por alguna razón se escoge un puerto que no posee el computador, el programa genera un error, al cual le damos terminar.



Una vez escogida las opciones y seleccionado al puerto correcto, procedemos a enviar. Al momento de conectarnos, debemos validar las opciones con el fin de determinar si hemos o no cometido algún error en la escogencia de estás. Una vez validadas las opciones aparecerá un letrero que nos avisara que se puede enviar los datos al generador.



Luego de envíar, usando la opción de "ENVIAR" que aparece después de validar los datos debemos esperar a que la interfaz nos informe si los datos en llegado (Envío correcto), no han llegado (Envío fallido) o si no hay conexión o no se ha confirmado el dato.

En caso de no aparecer el mensaje de "Envío correcto", aparecerá un botón con el nombre de "RESET" el cual es recomendable usar para así garantizar que las opciones en la interfaz se reinicien. En caso de no hacerlo, es recomendable entonces usar el botón de "DESCONECTAR", el cual no reinicia las opciones y garantiza que al momento de enviar los datos, la ráfaga de información que viajara desde la interfaz grafica al generador será la previamente escogida.

Cabe resaltar que una vez aparezca el mensaje de “Envío correcto” en la pantalla, si deseamos hacer un nuevo envío y con las mismas opciones solo debemos usar nuevamente el botón de “ENVIAR”. Por el contrario si deseamos opciones diferentes, desconectamos y escogemos las nuevas opciones.

NOTA: después de realizar un envío satisfactorio o no des la interfaz a la FPGA se DEBE REINICIAR EL GENERADOR con el botón de “reset”.

8.5 Anexo 5 Código de programación de la interfaz gráfica desarrollada en Visual Basic 6.0

```
Dim OKRecibido As Boolean
Dim InBuff As String

Private Sub Command1_Click()

  Dim aparecer As String

  ' Validacion numero de 1 a 8
  If IsNumeric(npal.Text) = True Then
     If Val(npal.Text) >= 1 And Val(npal.Text) <= 8 Then
       ' MsgBox "OK"
       aparecer = 1
     Else
        MsgBox "Debe ingresar un numero de 1 a 8 en el Num pal"
        Exit Sub
     End If
  Else
          MsgBox "Debe ingresar un numero en Num pal"
          Exit Sub
```

```
End If



' Validacion numero de bits de 1 a 16
If IsNumeric(nbits.Text) = True Then
   If Val(nbits.Text) >= 1 And Val(nbits.Text) <= 16 Then
      ' MsgBox "OK"
      aparecer = aparecer + 1
   Else
       MsgBox "Debe ingresar un numero de 1 a 16 en el Num bits"
       Exit Sub
   End If
Else
         MsgBox "Debe ingresar un numero en Num bits"
         Exit Sub
End If

'validacion tipo señal
If Val(tsignal.ListIndex + 1) = "0" Then
     MsgBox ("Digite el tipo de señal")
     Exit Sub
     End If

'validacion tipo envio
   If Val(tenvio.ListIndex + 1) = "0" Then
     MsgBox ("Digite el tipo de envio")
     Exit Sub
     End If

 'para ver el boton de enviar
```

```
        If aparecer = 2 Then
        MsgBox "Ya puedes enviar"
        enviar.Visible = True



' protocolo
' bit mas significativo es cero...
' bitslargo.Text = "0"
' tipo de señal

 'si es serial o paralelo
If tsignal.Text = "Serial" Then
    bitslargo.Text = bitslargo.Text & "0"
Else
    bitslargo.Text = bitslargo.Text & "1"
End If

' numero de bits
bitslargo.Text = bitslargo.Text & ajuste(Dec2Bin(nbits - 1), 4)

 ' numero de palabras
bitslargo.Text = bitslargo.Text & ajuste(Dec2Bin(npal - 1), 3)

' tipo de envio
If tenvio.Text = "Continuo" Then
    bitslargo.Text = bitslargo.Text & ajuste("1", 8)
Else
    bitslargo.Text = bitslargo.Text & ajuste("0", 8)
```

```
    End If

    ' agregar 8 palabras...
        For I = 1 To 8  ' Val(npal.Text)
            'paldigital = ajustepal(pal(I - 1).Text, 16)
            bitslargo.Text = bitslargo.Text & palabrabien(ajuste(pal(I - 1).Text, 16))
        Next

End If

End Sub

Private Sub Command2_Click()
    If IsNumeric(npal.Text) Then
        If Val(npal.Text) >= 1 And Val(npal.Text) <= 8 Then
            Dim I, j As Integer
            For I = 1 To Val(npal.Text)
                pal(I - 1).Text = ""
                For j = 1 To Val(nbits.Text)
                    pal(I - 1).Text = pal(I - 1).Text & Trim(Str(Int(Rnd(1) * 10) Mod 2))
                Next j
            Next
        End If
    End If
End Sub

Private Sub comunicacion_OnComm()


        Select Case comunicacion.CommEvent
```

```
' Handle each event or error by placing
' code below each case statement.

' This template is found in the Example
' section of the OnComm event Help topic
' in VB Help.

' Errors
   Case comEventBreak   ' A Break was received.
   Case comEventCDTO    ' CD (RLSD) Timeout.
   Case comEventCTSTO   ' CTS Timeout.
   Case comEventDSRTO   ' DSR Timeout.
   Case comEventFrame   ' Framing Error.
   Case comEventOverrun ' Data Lost.
   Case comEventRxOver  ' Receive buffer overflow.
   Case comEventRxParity   ' Parity Error.
   Case comEventTxFull  ' Transmit buffer full.
   Case comEventDCB     ' Unexpected error retrieving DCB]

' Events
   Case comEvCD   ' Change in the CD line.
   Case comEvCTS  ' Change in the CTS line.
   Case comEvDSR  ' Change in the DSR line.
   Case comEvRing ' Change in the Ring Indicator.
   Case comEvReceive ' Received RThreshold # of chars.

      InBuff = comunicacion.Input
      OKRecibido = True
   Case comEvSend ' There are SThreshold number of
                  ' characters in the transmit buffer.
```

```
        Case comEvEOF  ' An EOF character was found in the
                       ' input stream.
      End Select
End Sub

Private Sub conexion_Click()
' un unico boton para conectar o desconectar
  If conexion.Caption = "CONECTAR" And Val(compc.ListIndex + 1) <> "0" Then
    'seleccion del COMX
    comunicacion.CommPort = Val(compc.ListIndex + 1)
    ' abrir puerto
    comunicacion.PortOpen = True
    'enviar y recibir datos
    Command1.Visible = True
    Command2.Visible = False
    'Timer1.Enabled = True
    'para desconectar
    conexion.Caption = "DESCONECTAR"
  Else
    'MsgBox ("No ha selecionado el puerto COM de salida")
    'Exit Sub
    'End If

  If conexion.Caption = "DESCONECTAR" Then
  'desactivamos en orden:
  'Timer1.Enabled = False
  ' para no hacer visible el enviar
  Command1.Visible = False
  enviar.Visible = False
  Command2.Visible = True
```

```
   comunicacion.PortOpen = False
   bitslargo.Text = ""
   reset.Visible = False
   compc.Text = "----------"

   ' para volver a conectar
   conexion.Caption = "CONECTAR"
   End If

End If
End Sub

Private Sub enviar_Click()

Command1.Visible = False

Dim caracter As String
Dim l As Integer
Dim timeout As Long
l = Len(bitslargo.Text)

'Primera rafaga
For I = 1 To l Step 8
    caracter = Bin2Ascii(Mid(bitslargo.Text, I, 8))
    InBuff = ""
    OKRecibido = False
    timeout = 9999999
    comunicacion.Output = caracter
    'No hacer nada mientras espera le responde...
    Do
```

```
    timeout = timeout - 1
    Loop While (timeout > 0)
    InBuff = comunicacion.Input

    If InBuff <> Chr(128) Then
        MsgBox ("El dispositivo no pudo confirmar el dato o no hay conexión con la FPGA")
        MsgBox "Vuelva a enviar los datos o reinicie opciones y vuelva a enviar"
        reset.Visible = True
        Exit Sub
    End If
    'If InBuff = "M" Then
     '   MsgBox "Envío fallido"
    'Else
      ' If InBuff = "B" Then
       '     MsgBox "Envío correcto"
     '   End If
    'End If
Next

'Segunda rafaga
Dim bytes As Integer
bytes = 0
For j = 1 To 136 Step 8
    caracter = Bin2Ascii(Mid(bitslargo.Text, j, 8))
    InBuff = ""
    OKRecibido = False
    timeout = 9999999
    comunicacion.Output = caracter
    'No hacer nada mientras espera le responde...
    Do
```

```
    timeout = timeout - 1
    Loop While (timeout > 0)
    InBuff = comunicacion.Input

    If InBuff <> Chr(128) Then
        MsgBox ("El dispositivo no pudo confirmar el dato o no hay conexión con l aFPGA")
        MsgBox "Vuelva a enviar los datos o reinicie opciones y vuelva a enviar"
        reset.Visible = True
        Exit Sub
    End If
    'If InBuff = "M" Then
     '  MsgBox "Envío fallido"
    'Else
     ' If InBuff = "B" Then
         ' MsgBox "Envío correcto"
     ' End If
    'End If
Next

'último byte
For k = 137 To 144 Step 8
    caracter = Bin2Ascii(Mid(bitslargo.Text, k, 8))
    InBuff = ""
    OKRecibido = False
    timeout = 9999999
    comunicacion.Output = caracter
    'No hacer nada mientras espera le responde...
    Do
    timeout = timeout - 1
    Loop While (timeout > 0)
```

```
    InBuff = comunicacion.Input
    'If InBuff <> Chr(128) Then
     '   MsgBox ("El dispositivo no pudo confirmar el dato")
      '  MsgBox "Vuelva a enviar los datos o reinicie opciones y vuelva a enviar"
       ' reset.Visible = True
        'Exit Sub
    'End If
    If InBuff = "M" Then
       MsgBox "Envío fallido"
    Else
      If InBuff = "B" Then
         MsgBox "Envío correcto"
      End If
    End If
Next

End Sub



Private Sub npal_Change()
    If IsNumeric(npal.Text) Then
       If Val(npal.Text) >= 1 And Val(npal.Text) <= 8 Then
          Dim I As Integer
          For I = 1 To Val(npal.Text)
             pal(I - 1).Text = 0
             pal(I - 1).Enabled = True
          Next
          For I = Val(npal.Text) + 1 To 8
             pal(I - 1).Text = ""
             pal(I - 1).Enabled = False
```

```
            Next
        End If
    End If
End Sub

Private Sub pal_KeyPress(Index As Integer, KeyAscii As Integer)
' solo permite escribir el ascii 48 y 49 ... osea '0' y '1'...
    If KeyAscii = 8 Then ' no valide si es borrar
        Exit Sub
    End If
    If Not (KeyAscii = 48 Or KeyAscii = 49) Then
        KeyAscii = 0
    End If
    If Len(pal(Index).Text) + 1 > Val(nbits.Text) Then
        KeyAscii = 0
    End If
End Sub
Function ajuste(ByVal x As String, ByVal bdeseados As Integer) As String
    Do While Len(x) < bdeseados
        x = "0" & x
    Loop
    ajuste = x
End Function

Function ajustepal(ByVal x As String, ByVal bdeseados As Integer) As String
    Do While Len(x) < bdeseados
        x = x & "0"
    Loop
    ajustepal = x
End Function
```

```
Private Function InvertirCadena(CadenaOriginal As String) As String
   Dim CadenaInvertida As String, I As Integer
   CadenaInvertida = ""
   For I = Len(CadenaOriginal) To 1 Step -1
      CadenaInvertida = CadenaInvertida & Mid(CadenaOriginal, I, 1)
   Next
   InvertirCadena = CadenaInvertida
End Function

Private Function palabrabien(Cadena As String) As String
   pal2 = ""
   For I = 1 To 8
      pal2 = pal2 & Mid(Cadena, I, 1)
   Next
   pal1 = ""
   For j = 9 To 16
      pal1 = pal1 & Mid(Cadena, j, 1)
   Next
   palabrabien = pal1 & pal2
End Function

Function Dec2Bin(ByVal n As Long) As String
Do Until n = 0
   If (n Mod 2) Then Dec2Bin = "1" & Dec2Bin Else Dec2Bin = "0" & Dec2Bin
   n = n \ 2
Loop
End Function

Function Bin2Dec(Num As String) As Long
```

```
  Dim n As Integer
    n = Len(Num) - 1
    a = n
    Do While n > -1
      x = Mid(Num, ((a + 1) - n), 1)
      Bin2Dec = IIf((x = "1"), Bin2Dec + (2 ^ (n)), Bin2Dec)
      n = n - 1
    Loop
End Function

Function Bin2Ascii(Binario As String) As String
   Dim I, longitud As Integer
   Dim retorno As String
   retorno = ""
   retorno = Chr(Val(Bin2Dec(Binario)))
   Bin2Ascii = retorno
End Function

Private Sub reset_Click()

tsignal.ListIndex = "0"
tenvio.ListIndex = "0"
npal.Text = ""
nbits.Text = ""
bitslargo.Text = ""
pal(0).Text = ""
pal(1).Text = ""
pal(2).Text = ""
pal(3).Text = ""
pal(4).Text = ""
```

```
pal(5).Text = ""
pal(6).Text = ""
pal(7).Text = ""

Command1.Visible = False
enviar.Visible = False
Command2.Visible = True
comunicacion.PortOpen = False
conexion.Caption = "CONECTAR"
compc.Text = "----------"
reset.Visible = False
End Sub
```

8.6 Anexo 6 Layout GSProg