Transmisor de mensajes por código Morse

Adrián Dittel-Retana, Gabriel O. González-Rodríguez, Emmanuel Naranjo-Blanco, Jose Fabio Navarro-Naranjo, David Rodríguez-Camacho
adriandittel19@estudiantec.cr gabrielgr01@estudiantec.cr naranjo760emm@estudiantec.cr
josefabio1127@estudiantec.cr davo4006@estudiantec.cr
Área Académica de Ingeniería Mecatrónica
Instituto Tecnológico de Costa Rica

***Resumen*—En su momento, el código morse impactó los métodos de comunicación a largas distancias. Este se trata de un sistema que codifica información a base de rayas y puntos, en el que se representan mensajes incluyendo letras, números y caracteres. En el presente proyecto de laboratorio se presentará una solución propuesta al diseño de un circuito digital que traduce las letras del abecedario y los dígitos numéricos a código morse. Como parte del problema por resolver se utilizó la teoría de lógica combinacional y lógica secuencial con el fin de cumplir con el objetivo principal, comprender las estructuras lógicas y secuenciales en la comprobación práctica de un problema mediante HDL. La propuesta de solución se basó fundamentalmente en la partición por módulos, cada uno cumpliendo una finalidad característica, entre los cuales destaca el uso de registros en paralelo, parte esencial en la propuesta de diseño. Una vez finalizado el experimento, se llegó a la conclusión principal del proyecto: Mediante el uso de lógica combinacional y secuencial, fue posible implementar un traductor de carácteres a código morse. Además, se comprobó el experimento en la placa de desarrollo BASYS 3, programada en Verilog.**

***Palabras clave*—Lógica secuencial, Lógica combinacional, Código morse, Diseño modular, Circuitos digitales.**

I. INTRODUCCIÓN

Una parte esencial en la electrónica digital consiste en la lógica secuencial, que en conjunto con la lógica combinacional, facilitan la operación de múltiples sistemas tecnológicos actuales. En otras palabras, gran parte de los elementos digitales consisten tanto de circuitos combinacionales como de elementos de memoria. De este modo, la idea fundamental de este proyecto consiste en aplicar los conocimientos teóricos de los circuitos digitales en un sistema práctico totalmente funcional.

La esencia detrás de un circuito secuencial es que sus salidas dependen de sus entradas actuales y también de la secuencia pasada de sus entradas. De acuerdo con Tocci, una de las funciones principales de las configuraciones secuenciales consiste en que según la operación combinacional sobre las entradas, se producen varias salidas de las cuales algunas se utilizan para determinar los valores que se almacenan en memoria [1].

El elemento de memoria más importante es el flip-flop, que según su configuración interna de compuertas lógicas permiten jugar con la información. Existen varios tipos de flip-flop, entre ellos tipo D, SR, T, JK; entre otros [1]. Por su parte, en términos de mayor complejidad, se habla de circuitos integrados con arreglos de flip-flops internos que determinan funciones, como puede ser un contador, registros de desplazamiento o memorias. Aspectos que se tratarán en el diseño del proyecto.

Asimismo, dichos sistemas digitales son gobernados por una señal que controla los cambios de estados de las salidas en función de sus entradas. Esta se trata de una señal de reloj que oscila en tiempos específicos para determinar las transiciones. A esto se le denomina operación síncrona, la cual puede ocurrir en los flancos positivos o negativos de la señal de reloj (CLK). Por su parte, también se encuentran operaciones asíncronas, es decir, que tienen total libertad y son independientes de los flancos de reloj, por ejemplo, el uso de una señal de reset que reinicia el sistema (RST) [1].

La velocidad a la que opera un sistema digital síncrono depende de la frecuencia con la que ocurren los ciclos de reloj. En el presente proyecto se trabajará con una placa de desarrollo FPGA - Basys3 que va a operar a una frecuencia de 100 MHz. Además, se aplicará el concepto de divisor de frecuencia para definir unidades de tiempo en la respuesta del sistema propuesto, con el fin de que las salidas sean de fácil comprensión para el ojo humano, por ejemplo, periodos de 2 segundos.

En relación con el tema en cuestión, años atrás, el código Morse representó una forma revolucionaria hacia la comunicación, que tuvo relevancia en transmisión de información a larga distancia y actualmente tiene uso en áreas marinas, militares y de aviación. Se trata de una especie de lenguaje o diccionario que describe letras, números y caracteres únicamente con puntos y guiones. Cada uno tiene su secuencia propia que la caracteriza, y en este caso, se trabajará exclusivamente con las letras mayúsculas del alfabeto latino y los números del sistema decimal, indicados en la figura 1.

La idea fundamental del proyecto es definir un trasmisor de mensajes de caracteres en código ASCII a código morse, que tenga tres entradas y una salida. Las entradas se tratan de un mensaje enviado desde una transmisión de datos por puerto serial RX, un reset RST que reinicia el sistema con una señal activa en alto y una señal de inicio START que permite comenzar la traducción; y la salida Y representa el mensaje codificado.

De forma conceptual, el protocolo de comunicación UART, Transmisor Receptor Asíncrono Universal por sus siglas, es una especie de protocolo de comunicación en serie que se utiliza para el intercambio de datos a corta distancia, baja velocidad y bajo costo entre la computadora y los periféricos. La comunicación UART básica solo necesita dos líneas de

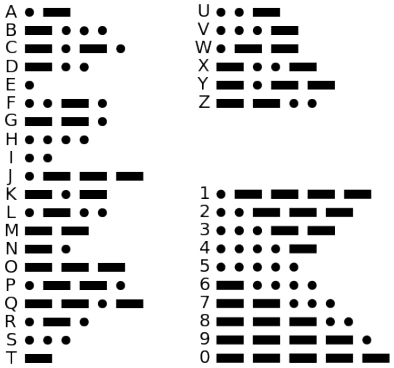

Figura 1. Representación de letras y números mediante el código Morse internacional.

señales (RXD, TXD) como se observa en la figura 2 [2].

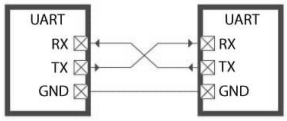

Figura 2. Esquemático de comunicación serial UART.

Una vez descrita cada operación del sistema, el proyecto se requiere comprobar de dos formas prácticas, una mediante el uso de lenguaje de descripción de hardware y otra a partir de la transmisión serial hacia una FPGA BASYS 3 Artix-7 (xc7a35tcpg236-2L) en Verilog.

Por último, para el desarrollo del proyecto se aplicaron estrategias de diseño modular, que acoplan un gran sistema mediante circuitos más simples, cuya representación se da bajo diagramas de primer, segundo y tercer nivel; cada uno cumpliendo los estándares de documentación en el diseño, los cuales se abordarán detalladamente más adelante.

Sírvase los conceptos expuestos como base de partida en el desarrollo del presente proyecto, que tiene como objetivo general desarrollar el diseño para implementar un transmisor de código morse aplicando circuitos digitales, del cual se parte para definir los objetivos específicos del proyecto explicados seguidamente.

El primero de ellos es fraccionar el problema en diversos módulos con la finalidad de implementar un diseño modular que aplique los conocimientos y conceptos estudiados sobre circuitos secuenciales y combinacionales. Una vez alcanzado el objetivo anterior, se plantean el siguiente objetivo, el cual consiste en verificar la funcionalidad del diseño propuesto mediante el lenguaje de descripción de hardware (HDL), para luego, verificar el funcionamiento del circuito transmisor de código morse en la FPGA Basys 3 Artix 7. Finalmente, el último objetivo planteado, luego de la implementación de la solución propuesta, es analizar el diseño propuesto en términos de calidad, eficiencia, integración de conceptos combinacionales y secuenciales, y funcionalidad del circuito tanto en HDL como en la FPGA.

En las siguientes secciones se detallará el proceso de diseño junto con los diagramas modulares y su respectiva descripción, además se detallará el proceso de creación del circuito en el texto de documentación anexado en las referencias del presente, al cual se invita a leer. Posteriormente se analizarán los resultados obtenidos tanto a nivel físico como simulado mediante el software de Verilog.

II. DESCRIPCIÓN DE SOLUCIÓN

II-A. PRIMER NIVEL

En la figura 3 se muestra el diagrama de primer nivel para el circuito transmisor de código morse.

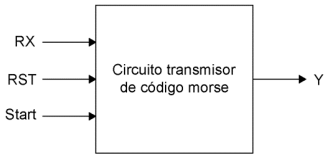

Figura 3. Diagrama de primer nivel para el transmisor de código morse.

II-B. SEGUNDO NIVEL

En la figura 6 de los anexos se muestra con detalle el diagrama de segundo nivel para el diseño del circuito transmisor de código morse.

II-C. TERCER NIVEL

II-C1. MÓDULO RECEPTOR: Con el fin de manejar de forma correcta los datos recibidos a través de comunicación serial se utilizó el módulo receptor. El cual distribuye los datos recibidos en serie de forma paralela hacia el resto del circuito. En otras palabras, al módulo le ingresa la señal RX con los datos en serie y envía a la salida 10 bits en paralelo, correspondiendo el primero al bit de ”start”, los ocho del medio al código ASCII del carácter ingresado, y el último al bit de ”stop”.

Para este módulo se utilizó la teoría planteada en las técnicas de comunicación por medio de puerto serial, y se tomó como referencia el código de Verilog del proyecto de alexwonglik [3].

Adicional a esto se agregó otra salida llamada Çhar_Detected”que envía una señal en alto al módulo contador de caracteres cada vez que ingresa un carácter nuevo.

II-C2. MÓDULO TRADUCTOR: La finalidad del módulo traductor es ser el primer paso hacia la traducción a código morse del dato recibido. La entrada recibe los 7 bits del caracter en código ASCII proveniente del módulo receptor y tiene dos salidas, una para el tamaño del mensaje en código Morse y la otra para la codificación en Morse del caracter. Estas dos salidas se componen de dos multplexores que trabajan como una memoria, uno recibe el ASCII a traducir y devuelve su homólogo en morse y, el segundo indica el tamaño de palabra del caracter.

El homólogo en morse tiene la siguiente composición: un 1 para el punto, tres 1 para para cada raya, un 0 corresponde a los espacios en un solo caracter y el espacio entre caracteres se compone de tres ceros. Además de esto se tiene una entrada destinada al espacio entre palabras, el cual corresponde a cuatro ceros, ya que al combinarse con los otros tres ceros del espacio entre caracteres del caracter anterior, se obtiene un espacio entre palabras de siete ceros. Esto se tomando en cuenta el código morse de la figura 1 de derecha a izquierda.

II-C3. MÓDULO DE MEMORIA: Este módulo se divide internamente el 2 etapas en secciones, las cuales están integradas por registros en paralelo, multiplexores, contadores, un comparador, un decodificador y un FF-JK. La primera etapa, a partir de las señales recibidas de los módulos anteriores a este, se encarga de almacenar en los registros los caracteres y la información necesaria para su salida. Luego, la segunda etapa utiliza la señal de Start para desencadenar la salida de los datos.

Primeramente, en la etapa de entrada (la cual está formada por un decodificador de 4 bits a 16 bits, y los 12 registros en paralelo de 27 bits) se reciben las dos entradas provenientes del módulo traductor y la salida del contador de 4 bits, que mediante la señal Char_Detected derivada del módulo receptor, se encarga de contar la cantidad de caracteres que han sido recibidos como parte del mensaje. Con las entradas anteriores, las cuales son de 22, 5 y 4 bits respectivamente, dicho circuito se encarga de guardar los datos del mensaje.

Primero, las entradas del mensaje codificado, y la longitud de dicho mensaje, se colocan en las entradas de todos los registros, a la espera de un pulso de reloj para ser almacenadas. Mientras esto sucede, en los 4 bits de entrada del decodificador, se coloca la salida del contador de caracteres (el cual empieza a contar en 0), por lo que, cuando el contador aumenta en 1 su valor, la salida 1 del decodificador se activa y envía un pulso únicamente al primer registro, debido a la conexión que se observa en la Figura 9 de los anexos, lo cual provoca que se guarden los datos del primer caracter en este registro, y luego de esto repite este mismo procedimiento para almacenar todos los datos caracteres ingresados por el usuario, uno a uno en los 11 registros restantes.

Luego de que el mensaje está guardado en los 12 registros, se activa la señal de “Start”, la cual provoca que los contadores de 4 y 5 bits presentes en la etapa de salida se activen y comiencen con la migración de datos hacia la salida. La señal de “Start”provoca que el FF JK tome un valor de 0 en su salida, y por esto, deje de activar las entradas de “RST”de los contadores.

Asimismo, hablando meramente de la salida de los datos, se puede observar, en la figura 9 de los anexos, que las salidas de los registros se dividen en 2 buses de datos, los cuales se conectan a 2 multiplexores de 16 a 1 (esto debido a que provienen señales de 12 registros distintos), específicamente, la salida de la codificación en Morse se conecta a un multiplexor que transporta 22 bits, y la salida de la longitud del mensaje se conecta a un multiplexor que transporta 5 bits de información. Es importante mencionar que dichos multiplexores funcionan con el mismo selector, el cual proviene de un contador de 4 bits e indica a cuál registro se le van a sacar los datos.

La función del multiplexor de los datos de longitud es enviar la longitud del mensaje a un comparador, al cual le ingresa también la señal de un contador de 5 bits, de modo que el comparador va a emitir una señal que indique cuando el mensaje salió por completo y de este modo, el contador que sirve de selector para los multiplexores mencionados anteriormente cambia al estado siguiente, y se procede a la migración de los datos del siguiente registro.

Asimismo, la señal del contador de 5 bits sirve como selector para otro multiplexor colocado en la salida final del módulo, y del diseño. Este multiplexor recibe el bus de los 22 bits que salen del multiplexor anterior, y que representa el mensaje codificado en Morse. Estos 22 bits se conectan a las primeras 22 entradas del multiplexor, de modo que al ir aumentando el contador de 5 bits, va permitiendo que salga bit a bit el mensaje codificado en Morse como se esperaba. Y así, este proceso se va a repetir hasta que se complete la salida de los 12 caracteres almacenados en los registros.

Finalmente es importante destacar algunos aspectos importantes en el diseño de este módulo. Primero, en la figura 9 de los anexos, se pueden observar 2 compuertas lógicas del tipo OR, las cuales sirven como lógica de control de los contadores, para poder activar el reset de los mismos mediante diversas señales. Segundo, el reloj que se utiliza para operar la etapa de salida de este módulo tiene un periodo de 2 segundos, ya que proviene de un divisor de frecuencia, esto debido a que se necesita que el mensaje pueda ser captado por los sentidos de una persona.

II-C4. MÓDULOS ESTANDARIZADOS: Para el diseño de este proyecto se utilizaron diversos módulos estandarizados, como registros, contadores, comparadores y divisores de frecuencia. Primero, los registros utilizados corresponden a una serie de flip-flops tipo D que permiten el almacenamiento y transporte de datos de modo paralelo. Luego, los contadores implementados son contadores normales, cuya cantidad de bits se muestran en los diagramas de los anexos. Dichos contadores avanzan de forma continua, y en caso de necesitar una inicialización, el diseño permite aplicar una señal de reset a los mismos, la cual puede provenir de diferentes partes. Los comparadores comprenden una lógica completamente combinacional que envía un pulso de salida cuando encuentra igualdad en las 2 señales de entrada. Y por último, el divisor de frecuencia, que se muestra en la Figura 8 de los anexos, tiene una estructura formada por un contador, un comparador y un flip-flop T de modo de que envía una señal a la salida una vez que se cumplen la cantidad de ciclos en los que se desea dividir la frecuencia original.

Finalmente de manera general, para la simulación en el lenguaje de descripción de hardware, se omitió la comunicación serial y se añadieron dos módulos nuevos, un comparador que opera de manera inversa, es decir, emite un pulso en alto cuando los valores de entrada son diferentes, esto para simular la acción del modulo receptor de emitir la señal Char_Detected. Y también, se utilizó un registro de 8 bits en paralelo, como el descrito anteriormente para complementar con el comparador anterior y poder detectar de manera correcta el ingreso de un nuevo caracter.

III. IMPLEMENTACIÓN Y ANÁLISIS DE RESULTADOS

En la figura 4 podemos visualizar el diagrama de tiempos de la simulación del circuito en Vivado. Los bits de RxData corresponden al código ASCII de los caracteres utilizados a la entrada del circuito y la señal Y corresponde al código morse de todo el mensaje. No todos los caracteres se llegan a visualizar en la salida debido al tiempo de simulación, pero se comprueba, con los que sí, que el diseño está funcionando correctamente.

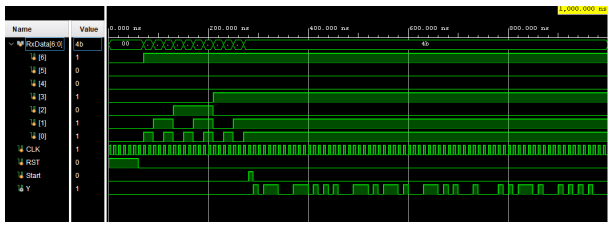

Figura 4. Diagrama de tiempos de la simulación en el software Vivado.

Por otra parte, para realizar la implementación del diseño planteado, se conectó la FPGA para cargar el código generado en Verilog. Para esto se utilizaron dos botones, uno para la señal de RST y otro para la de Start, mientras que la salida Y se dio por medio de un LED. Por medio del software PuTTY se incorporó la comunicación serial entre la computadora y la FPGA, de forma que al ingresar un carácter (al presionar una tecla del teclado) esta sería transmitida, y recibida por la FPGA a la entrada RX.

Una vez probado el circuito en la FPGA se observa que el diseño realizado funciona de forma casi correcta, ya que se observa el código morse correcto, para cualquier carácter ingresado, sin embargo, la FPGA solo muestra el resultado para el primer carácter ingresado, a diferencia de la simulación en Verilog, en donde se podían observar hasta un máximo de 12 caracteres, en caso de ajustar el tiempo de simulación.

Los resultados del análisis post-implementación se presentan en la figura 5, en donde se observa que la utilización, en términos del LUT, fue del 1 %, lo que significa que fue baja. Además, con respecto a la energía, se tiene que el circuito consume un total de $2{,}192W$. En términos generales, se puede decir que la implementación del diseño planteado es eficiente en términos de energía y área.

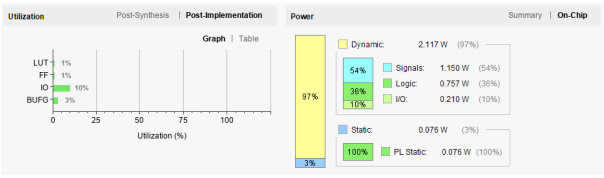

Figura 5. Resultados post-implementación de energía y área.

IV. CONCLUSIONES

A partir de este presente trabajo, se logró concluir lo siguiente.

- Se realizó un diseño a partir de lógica combinacional y secuencial, mediante la aplicación de los conceptos de diseño modular, el cual se implementó en distintos módulos que en conjunto cumplen con el objetivo de lograr una traducción de caracteres en código ASCII a código Morse de manera satisfactoria.
- Se verificó de manera exitosa el funcionamiento del diseño propuesto mediante HDL a partir de un testbench en Verilog. En este se realizó distintas pruebas con diversos mensajes arrojando resultados satisfactorios.
- Se logró implementar el circuito diseñado en la placa de desarrollo BASYS 3, de la cual se aplicó el protocolo de comunicación serial UART, del cual fue capaz la transmisión de caracteres desde el computador hacia la FPGA, traduciendo de binario a morse correctamente a partir de la iluminación de un LED.
- En términos de calidad, eficiencia y funcionalidad del circuito propuesto, se determinó que se planteó un circuito con área de uso de FPGA de un 1 % y un consumo de $2{,}192W$, lo cual es aceptable. No obstante, en términos de funcionalidad, existe cabida hacia futuras mejoras, específicamente en la transmisión del mensaje completo en el circuito físico.

V. RECOMENDACIONES

En función a mejorar los resultados obtenidos en el proyecto se proponen una serie de recomendaciones que se describen a continuación.

- Hacer una correcta división del proyecto en más módulos, para poder identificar los problemas de un módulo o de comunicación entre módulos de manera más sencilla, y así, determinar la fuentes de error que se pueden generar al pasar de la implementación en el lenguaje de descripción de hardware (mediante una simulación) a la hora de implementar la solución propuesta en FPGA utilizada.
- Generar test bench's para los distintos módulos en Verilog esto para cerciorarse del correcto funcionamiento del módulo, o en dado caso, para identificar de manera más sencilla los posibles errores que se puedan generar en la solución al estudiar uno a uno cada módulo.
- Realizar la mayor cantidad de pruebas y casos para el diseño planteado, esto para corroborar que lo que se planteó como solución realmente funciona para las diferentes longitudes del mensaje.

REFERENCIAS

[1] R. Tocci. *Sistemas digitales: Principios y aplicaciones*. Décima edición. Pearson Education. 2007.

[2] Ms. Neha R. Laddha. Prof. A.P. Thakare. *Implementation of serial communication using UART with configurable baud rate*, 1(4), 263-268. International Journal on Recent and Innovation Trends in Computing and Communication. 2013. http://www.ijritcc.org

[3] Alexwonglik. 2021. *UART Communication on Basys 3, FPGA Dev Board Powered by Xilinx Artix 7 Part II.* Recuperado de https://www.instructables.com/UART-Communication-on-Basys-3-FPGA-Dev-Board-Power-1/

VI. ANEXOS

En esta sección se presentan los diversos diagramas de segundo y tercer nivel que representan todos los aspectos del diseño propuesto. Se incluyen el diagrama general de la solución y los diagramas para todas las operaciones lógicas como secuenciales.

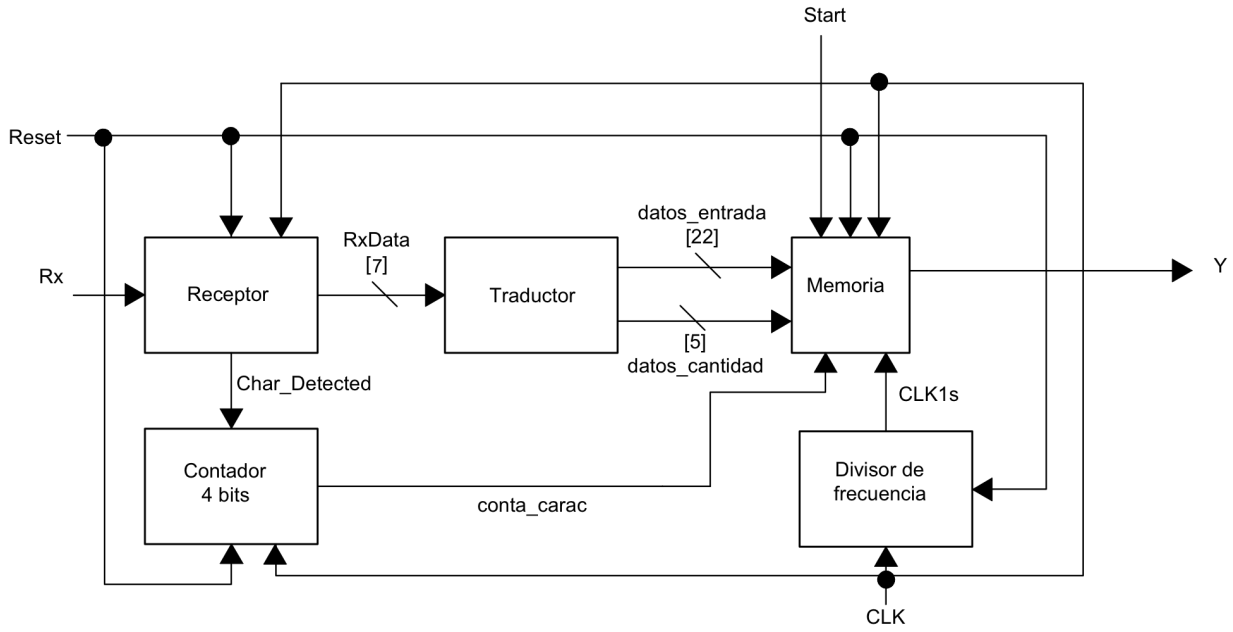

Figura 6. Diagrama de segundo nivel para el transmisor de código morse.

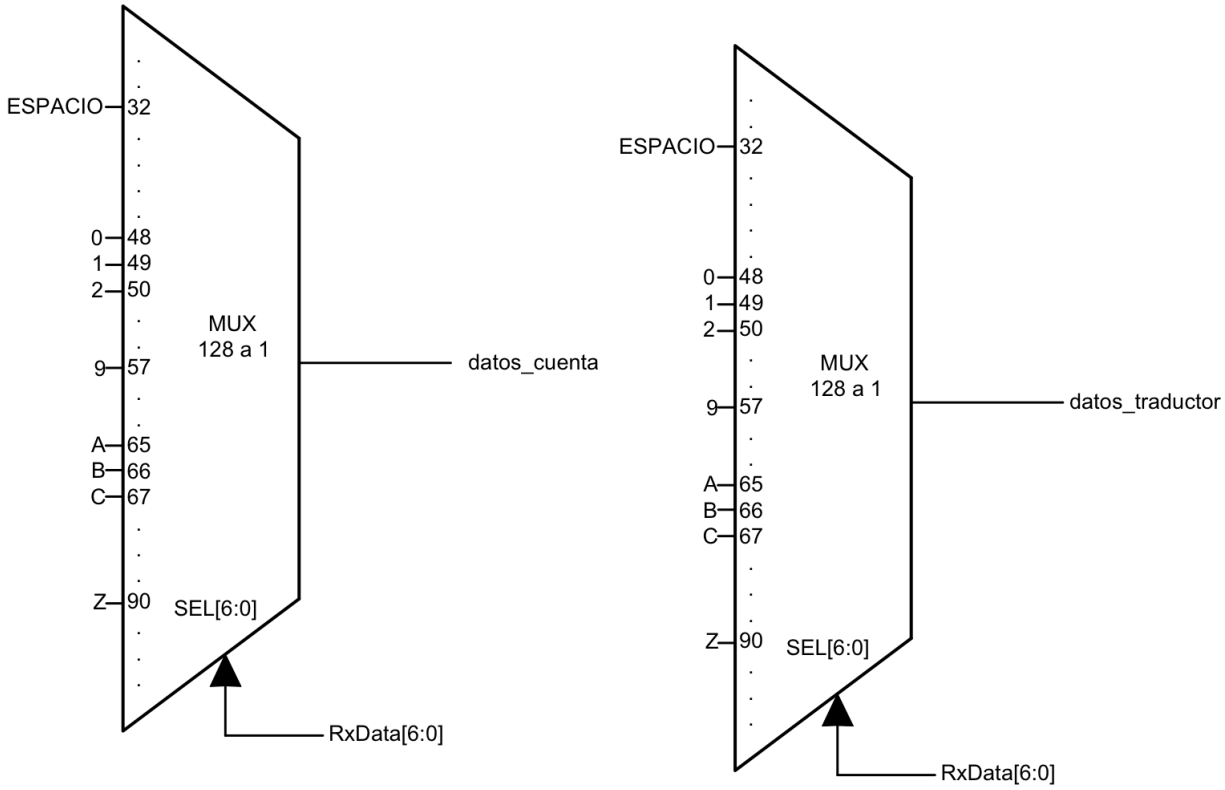

Figura 7. Diagrama de tercer nivel para el módulo traductor.

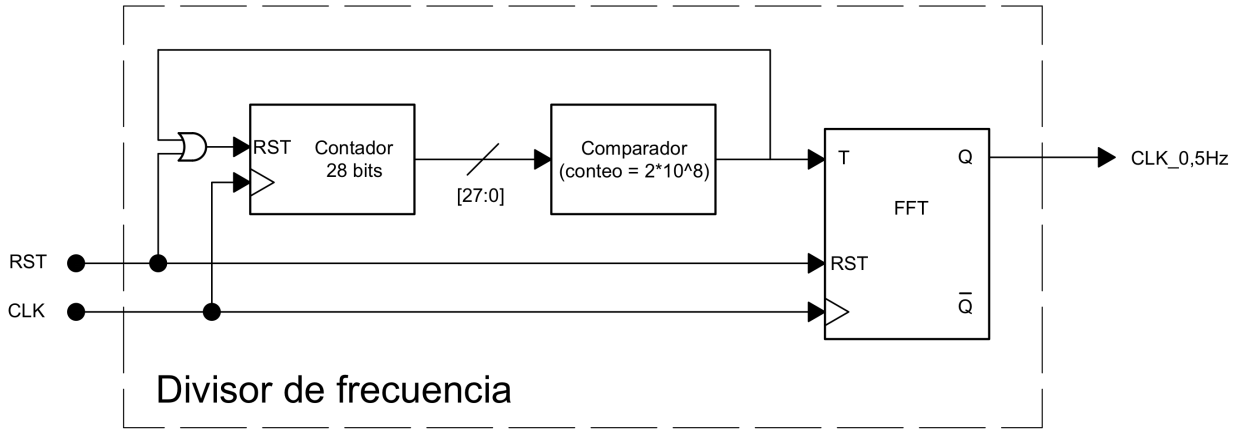

Figura 8. Diagrama de segundo nivel para el divisor de frecuencia.

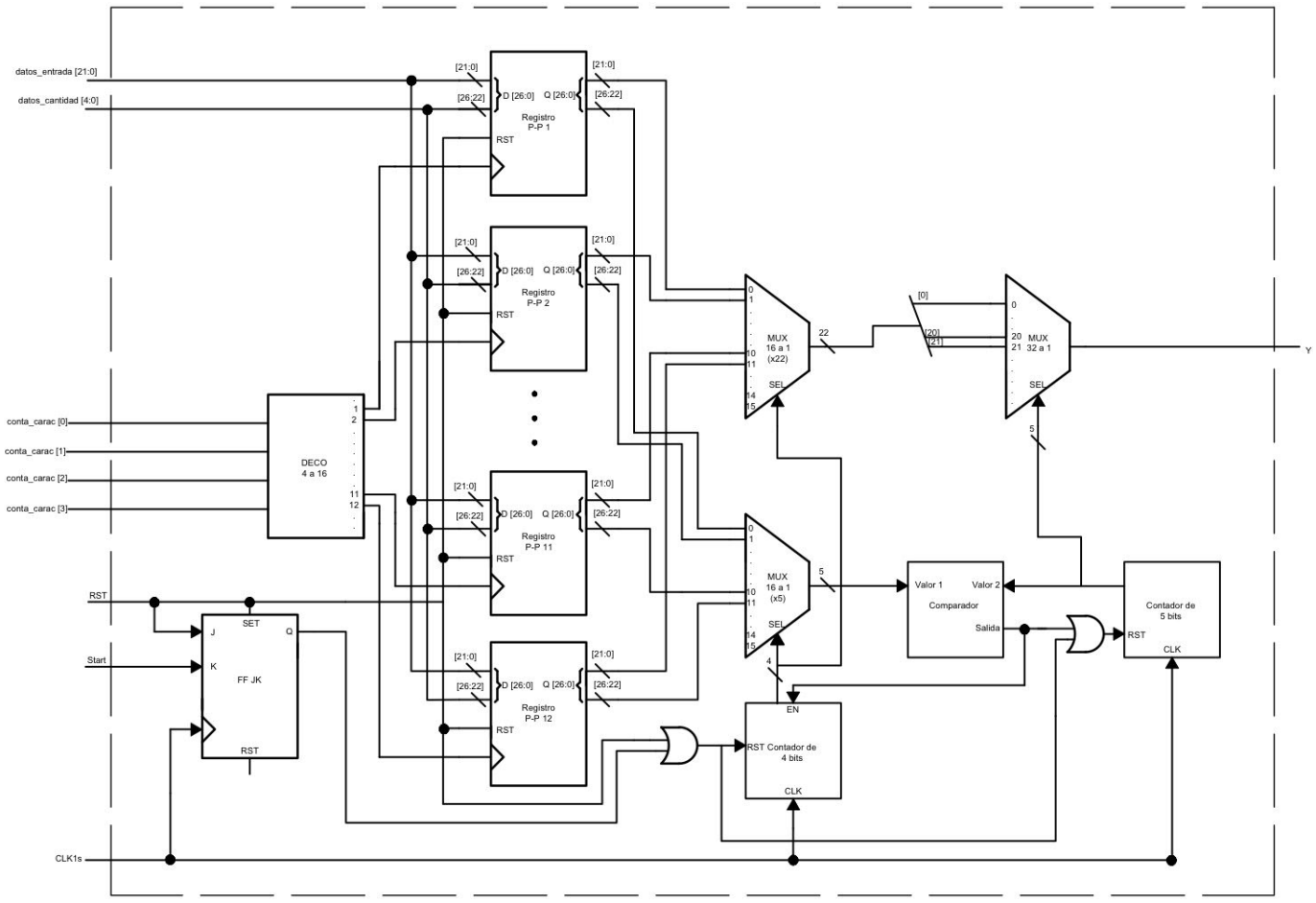

Figura 9. Diagrama de tercer nivel para el módulo de memoria.

SEL en ASCII	Salida codificada	Cantidad iniciando en 0	SEL en ASCII	Salida codificada	Cantidad iniciando en 0
ESPACIO	22'b0	3	**A**	22'b0000000000000000011101;	7
			B	22'b0000000000000101010111;	11
0	22'b0001110111011101110111;	21	**C**	22'b0000000000010111010111;	13
1	22'b0000011101110111011101;	19	**D**	22'b0000000000000001010111;	9
2	22'b0000000111011101110101;	17	**E**	22'b0000000000000000000001;	3
3	22'b0000000001110111010101;	15	**F**	22'b0000000000000101110101;	11
4	22'b0000000000011101010101;	13	**G**	22'b0000000000000101110111;	11
5	22'b0000000000000101010101;	11	**H**	22'b0000000000000001010101;	9
6	22'b0000000000010101010111;	13	**I**	22'b0000000000000000000101;	5
7	22'b0000000001010101110111;	15	**J**	22'b0000000001110111011101;	15
8	22'b0000000101011101110111;	17	**K**	22'b0000000000000111010111;	11
9	22'b0000010111011101110111;	19	**L**	22'b0000000000000101011101;	11
			M	22'b0000000000000001110111;	9
			N	22'b0000000000000000010111;	7
			O	22'b0000000000011101110111;	13
			P	22'b0000000000010111011101;	13
			Q	22'b0000000001110101110111;	13
			R	22'b0000000000000001011101;	9
			S	22'b0000000000000000010101;	7
			T	22'b0000000000000000000111;	5
			U	22'b0000000000000001110101;	9
			V	22'b0000000000000111010101;	11
			W	22'b0000000000000111011101;	11
			X	22'b0000000000011101010111;	13
			Y	22'b0000000001110111010111;	15
			Z	22'b0000000000010101110111;	13

Figura 10. Uso de multiplexores para la etapa del módulo traductor según el selector.