FACULTAD DE INGENIERÍA

Departamento de Electrónica

Tesis de Ingeniería Electrónica

Diseño de un Circuito Integrado CMOS para Identificación por Radiofrecuencia basado en el Estándar ISO-14443

Tesista
Fabricio P. Alcalde Bessia
Padrón №86296
f@lcald.com.ar

Director
Dr. Ing. José Lipovetzky
jlipove@fi.uba.ar

Co-Director
Ing. Octavio Alpago
oalpago@fi.uba.ar

Agosto, 2014

Agradecimientos

Agradezco a José por haberme dado la oportunidad de realizar este trabajo, que surgió de un interés personal.

También agradezco Diego M. por haber puesto en marcha el servidor con las herramientas que cómodamente pude usar desde mi casa y por haberme ayudado desde su experiencia con el trabajo.

Debo agradecer también a Allegro Microsystems, en especial a Patricio P. Preiti y Julio Raiponeri, por haberme dejado utilizar los elementos del laboratorio. También a MOSIS, Mentor Graphics y Synopsys, por sus respectivos programas estudiantiles que hicieron posible la realización de este trabajo.

Finalmente agradezco a mis compañeros y amigos que me ayudaron e hicieron más amenos todos estos años de estudio y sobretodo agradezco a mi familia por haber hecho de soporte todo este tiempo.

Resumen

En el presente trabajo se comenzará realizando una breve introducción a los sistemas de identificación por radiofrecuencia. Luego se analizará detalladamente el estándar ISO/IEC 14443, enfocando el estudio a la interfaz de comunicación tipo A. A continuación se presentará el diseño de un circuito integrado que cumplirá el rol de *transponder* y que será implementado en un proceso CMOS estándar de 0,5 µm.

Para el diseño del circuito integrado se comenzará por analizar en profundidad el vínculo existente entre lector y transponder, lo que permitirá entender el proceso de traspaso de energía e información y se verán las distintas implementaciones posibles. Luego se desarrollará un modelo basado en la extracción de parámetros de la estructura física de las antenas, que permitirá verificar los resultados analíticos y realizar simulaciones mediante SPICE de los circuitos, estando éstos conectados a un modelo realista de la antena.

El circuito integrado contará con diseño analógico y digital, este último sintetizado a partir de código RTL. Se tratará entonces de un dispositivo de señal mixta por lo que se deberán compatibilizar ambos dominios. Se mostrará el diseño digital junto con su verificación funcional a nivel de compuerta y el diseño analógico con las simulaciones realizadas.

Finalmente se cerrará el trabajo con los detalles de la implementación del dispositivo en el proceso de fabricación CMOS y la verificación de su funcionamiento.

Índice general

1. Introducción a RFID **1**
- 1.1. Identificación por radiofrecuencia . . . 1
- 1.2. Clasificación de los sistemas de RFID . . . 2
- 1.3. Estandarización de los sistemas de RFID . . . 4
  - 1.3.1. ISO/IEC 14443 – Parte 1: Características físicas . . . 5
  - 1.3.2. ISO/IEC 14443 – Parte 2: Interfaz de radiofrecuencia para señal y energía . . . 5
  - 1.3.3. ISO/IEC 14443 – Parte 3: Inicialización y anticolisión 10
- 1.4. Resumen del capítulo . . . 15

2. Diseño del transponder de RFID **17**
- 2.1. Objetivos del diseño . . . 17
- 2.2. Descripción general del funcionamiento . . . 18
- 2.3. Implementación . . . 20

3. Acoplamiento Inductivo **23**
- 3.1. Transmisión de la energía . . . 23
- 3.2. Transmisión por modulación de carga . . . 29
- 3.3. Modelo de SPICE del arreglo de antenas . . . 32
- 3.4. Resultados del análisis . . . 33

4. Diseño digital **35**
- 4.1. Arquitectura . . . 36
- 4.2. Recepción de datos . . . 37
  - 4.2.1. Módulo `Bit Decoder` . . . 38
  - 4.2.2. Módulo `Frame Receiver` . . . 40
- 4.3. Transmisión de Datos . . . 42
  - 4.3.1. Módulo `Frame Sender` . . . 44
  - 4.3.2. Módulo `Bit Coder` . . . 46
- 4.4. Verificación funcional . . . 47
  - 4.4.1. Eco de un byte . . . 48
- 4.5. Implementación: Síntesis y *Place&Route* . . . 49

5. Diseño Analógico **53**
5.1. Acondicionamiento y uso de la energía 53
5.1.1. Regulador/Limitador de tensión 54
5.1.2. Rectificador + Filtro 60
5.2. Transmisión y recepción de datos 63
5.2.1. Detector de Pausas 63
5.2.2. Modulador 66
5.3. Generador de reloj 67
5.4. *Power-On Reset* (POR) 69

6. Implementación, Evaluación y Resultados **73**
6.1. *Layout* Completo del Circuito Integrado 73
6.2. Método de Verificación 76
6.3. Resultados 78
6.3.1. Recepción de datos 79
6.3.2. Transmisión de Datos 82
6.3.3. Regulador/Limitador de tensión 83
6.3.4. Verificación del funcionamiento completo 84

7. Conclusiones **87**

Bibliografía **91**

Capítulo 4

Diseño digital

El chip cuenta con un bloque digital que realiza un procesamiento mínimo de los datos recibidos para presentarlos en forma cruda —decodificados y desencapsulados— en la interfaz digital de salida, donde serán utilizados por otros sistemas. Por otro lado también realiza el encapsulamiento y codificación de los datos a enviar a través del canal de RF, generando para ello la señal de mando del bloque «Modulador». Todo el procesamiento, desde las entradas hasta las salidas del bloque digital, se realiza utilizando lógica CMOS estática y es sincrónico con la señal portadora de 13,56 MHz recibida en la antena. Sin embargo, dado que existen pausas en la portadora, el bloque digital también cuenta con un pequeño módulo asincrónico que ayuda a decodificar los datos recibidos.

El sistema digital fue diseñado utilizando el lenguaje descriptor de hardware *Verilog* [13], con el que se escribieron y probaron cada uno de los bloques por separado y luego todos en conjunto. La simulación del funcionamiento se realizó utilizando el software compilador/sintetizador *Icarus Verilog* [6] junto con *GTKwave* [4] para visualizar los resultados. La implementación del sistema digital fue realizada utilizando las herramientas de *Synopsys Inc.* para síntesis (*Desing Compiler* [1]) y place&route (*IC compiler* [5]), junto con la biblioteca de compuertas digitales de la Oklahoma State University (OSU) [10].

Para la verificación del funcionamiento a nivel lógico de cada uno de los bloques se desarrolló una estructura de pruebas que permitió enviar todas las combinaciones posibles de un byte de datos y comprobar su correcta recepción y transmisión. La estructura de pruebas consistió en varios *tasks* de Verilog parametrizados de forma tal de poder indicar los datos a enviar.

En este capítulo se describirá la arquitectura del sistema digital y su funcionamiento. Se analizarán los módulos que conforman el bloque digital, su interconexión y se detallará el funcionamiento de cada uno de los ellos junto con las simulaciones realizadas. Por último se darán detalles acerca de la implementación con las herramientas de *Synopsys*.

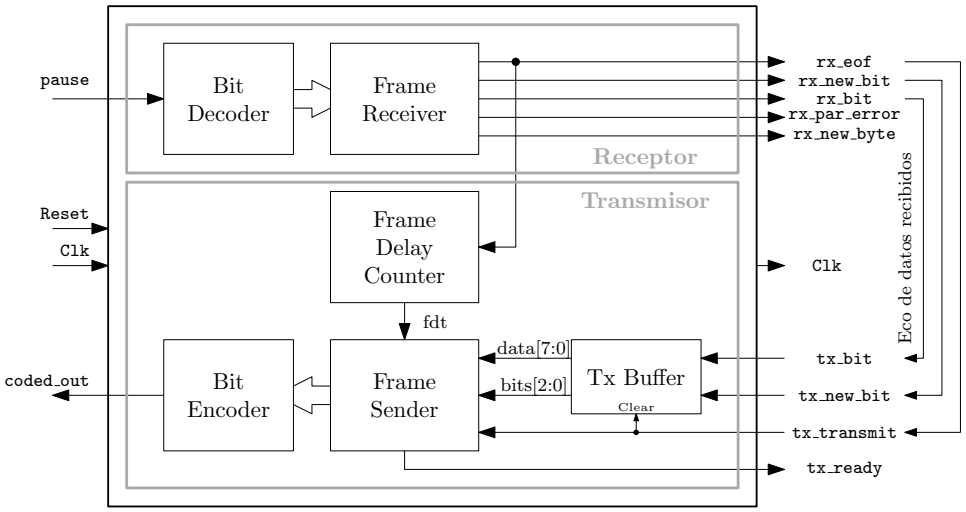

Figura 4.1: Diagrama en bloques del sistema digital.

4.1. Arquitectura

En la figura 4.1 se muestra un diagrama en bloques del sistema digital. El mismo puede dividirse en dos grandes partes que funcionan de forma independiente. Por un lado se tienen los módulos que conforman el bloque receptor de datos, compuesto por `Bit Decoder` y `Frame Receiver`, donde el primero identifica el comienzo de una trama y decodifica los bits recibidos, mientras que el segundo toma los bits decodificados y realiza la comprobación de las tramas, esto es, comprueba los bits de «Inicio», «Fin» y «Paridad». La carga útil dentro del frame es transmitida en serie a través de la salida `rx_bit`, con `rx_new_bit` señalizando el momento en que el dato es válido.

La recepción de datos se realiza a través de la entrada `pause`, que es la señal de salida del detector de envolvente y que se muestrea a una velocidad adecuada como para identificar cada bit.

Por otro lado se tiene el bloque transmisor de datos, formado por `Tx Buffer`, `Frame Sender`, `Frame Delay Counter` y `Bit Encoder`. Los datos a transmitir a través de la interfaz de RF se ingresan en serie al buffer `Tx Buffer` utilizando las entradas `tx_bit` y `tx_new_bit`. El bloque transmisor permite ingresar de uno a ocho bits, que serán transmitidos encapsulados en una trama estándar luego de cumplido el tiempo de demora FDT. El módulo `Frame Sender` es el encargado de agregar los bits de «Inicio», «Paridad» y «Fin» correspondientes a la trama e iniciar la transmisión cuando recibe la señal del módulo `Frame Delay Counter`. Los bits a transmitir son codificados en formato *Manchester* por el módulo `Bit Encoder` cuya función es también realizar la modulación OOK (ASK 100 %) con la señal sub-portadora de frecuencia $f_c/16$. La salida `coded_out` es la señal codificada y modulada que se conecta directamente al circuito modulador de carga.

La transmisión a través de la interfaz de RF debe comenzar luego de transcurrida una cantidad exacta de ciclos de la señal portadora, comenzando a contar a partir de la recepción del símbolo de «Fin» de trama, según fue definido por el estándar y se vio en la figura 1.5. El módulo `Frame Delay Counter` se encarga de contar la cantidad de ciclos transcurridos a partir de que se activa la señal `rx_eof`, la cual señaliza el fin de trama (*end of frame*), y al cumplirse el tiempo FDT activa la señal `fdt` que habilita al bloque `Frame Sender` a iniciar la transmisión.

La señal portadora fue utilizada como reloj para el sistema digital, lo que tiene la ventaja de evitar el uso de un generador de reloj interno a la vez que sincroniza el transponder con el lector. Sin embargo, también tiene la desventaja de que la señal se interrumpe debido al esquema de codificación y modulación definido por el estándar (sección 1.3.2). Entonces surge el inconveniente de que la señal `pause` debe ser muestreada para decodificar los bits justo en el instante en que no hay reloj. Para superar este inconveniente se diseñó un pequeño circuito asincrónico que forma parte del módulo `Bit Decoder` y cuya función es retener el estado de la señal `pause` hasta que retorne el reloj y pueda ser muestreada.

4.2. Recepción de datos

La recepción de datos requiere de tres pasos: primero, con el detector de envolvente analógico se demodula la señal recibida para quitar la portadora y obtener la señal `pause`, que es una representación digital de la envolvente. Luego se identifican los bits realizando un muestreo de `pause` y finalmente se procesan los bits de control de la trama y se presenta la carga útil en la salida.

En la figura 4.2 se observa un ejemplo del proceso de recepción. Allí se observa como `pause` cambia a estado '0' cada vez que ocurre una pausa en la entrada de RF y vuelve al estado lógico '1' cuando retorna la señal. La forma de `pause` es similar a la de las secuencias X, Y y Z definidas por el estándar (figura 1.3). Entonces basta con identificar cada una de esas secuencias y traducirlas a bits para obtener la información decodificada. Esta decodificación es la que se realiza el módulo `Bit Decoder` mediante el muestreo de la señal `pause`.

Según la norma, la duración de cada bit es de 128 ciclos de portadora, mientras que la duración de las pausas es de 32 ciclos. Tomando cuatro muestras por bit centradas dentro del intervalo, es decir, en los ciclos 16, 48, 80 y 112, es posible identificar las tres secuencias unívocamente. La secuencia X representa los '1' lógicos e Y y Z representan los '0' lógicos.

Una vez decodificados los bits una máquina de estados se encarga de desencapsular la información contenida en la trama analizando para ello cada bit recibido. Todas las tramas —ya sean cortas, estándar o anti-colisión

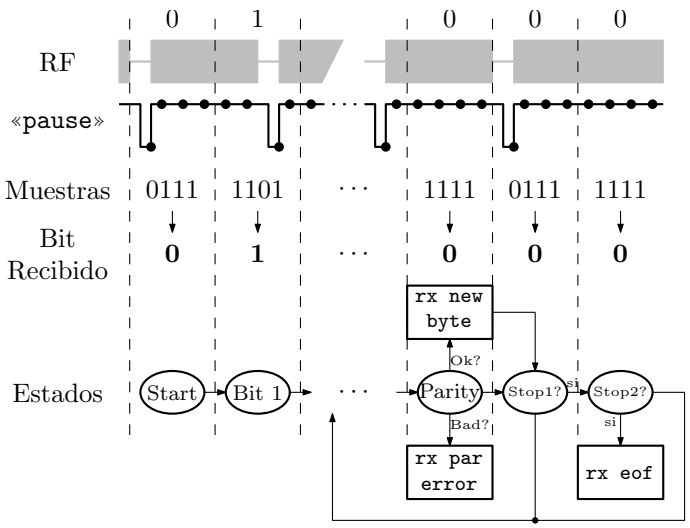

Figura 4.2: Esquema del proceso de decodificación de los bits y procesamiento de la trama recibida.

(figura 1.6)— comienzan con un bit de «Inicio» de comunicación, que se representa mediante un '0' y utilizando una secuencia Z. Esta secuencia tiene la característica de que la pausa se encuentra al inicio del tiempo del bit y por lo tanto al inicio de la comunicación. Entonces es posible utilizar esta primer pausa para sincronizar el traspaso de bits entre el lector y el transponder.

A continuación se reciben como máximo ocho bits de datos antes de recibir el bit de paridad. La trama puede finalizar antes con la recepción de un símbolo de «Fin» de comunicación (E: End) y en ese caso se tratará de una trama anticolisión, o de una trama corta si la cantidad de bits recibidos fue siete y son los primeros y únicos bits recibidos [1].

Una vez recibidos los ocho bits y comprobado el bit de paridad se señaliza el estado a través de las salidas `rx_new_byte` y `rx_par_error`. Si el bit de paridad fue correcto se procede a recibir otro byte a continuación, a menos que los dos bits siguientes sean el símbolo de «Fin», en cuyo caso se activa la salida `rx_eof` y el dispositivo entra en estado de reposo a la espera de un nuevo bit de «Inicio».

4.2.1. Módulo `Bit Decoder`

En la figura 4.3 se muestra un diagrama en bloques del módulo `Bit Decoder`. El mismo está compuesto por un registro de desplazamiento de 4 bits en donde se toman las muestras de la señal `pause`, un contador que

[1] Esta condición es debido a que podría darse el caso de que en una trama anti-colisión se divida un byte en el séptimo bit y que esos siete bits recibidos al final de la trama sean equivalente a algún comando válido (REQA, WUPA, etc).

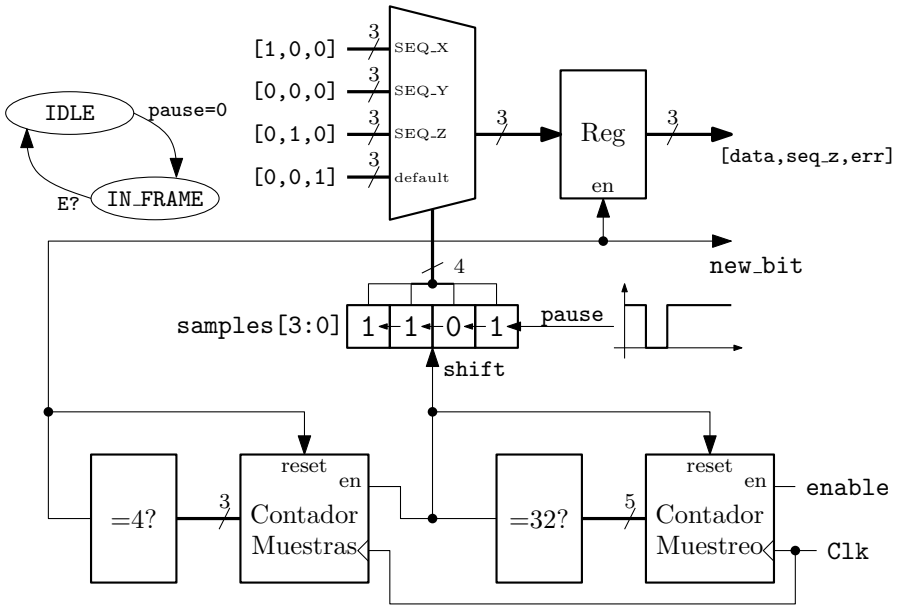

Figura 4.3: Diagrama en bloques de «Bit Decoder».

realiza el muestreo, otro que cuenta la cantidad de muestras tomadas, un multiplexor y una pequeña máquina de estados.

El módulo comienza en el estado de reposo `IDLE` en el que espera a recibir la primer pausa. En este estado los contadores se mantienen en cero y no se toman muestras. Cuando se detecta que la señal `pause` es igual a '0' —esto es al final de la pausa, que es cuando retorna la señal de reloj— se toma la primera muestra y el circuito pasa al estado `IN_FRAME`.

En el estado `IN_FRAME` se desplazan los bits del registro de desplazamiento cada 32 pulsos de reloj, entrando las nuevas muestras por el bit menos significativo. El contenido del registro de desplazamiento direcciona el multiplexor de forma tal de elegir `data` y `seq_z` en base a la secuencia recibida. Cuando el segundo contador cuenta cuatro muestras, se activa la señal `new_bit` y se habilita el registro de salida.

El módulo `Bit Decoder` no reconoce el fin de las tramas, ya que no es su función interpretar los datos recibidos, y por lo tanto queda en el estado `IN_FRAME`, convirtiendo secuencias en bits, hasta que recibe la señal de fin de trama del módulo `Frame Receiver`.

Por otro lado, existe el problema de que durante las pausas no se puede realizar ningún procesamiento, ya que el sistema digital se encuentra congelado en algún estado a falta de señal de reloj. Para superar este inconveniente, la detección de las pausas se realiza con la ayuda del circuito de la figura 4.4. Se trata de un circuito asincrónico que utiliza el flanco descendente de la señal `env`, que es la envolvente de la portadora, para retener las pausas y de esta forma generar la señal `pause`.

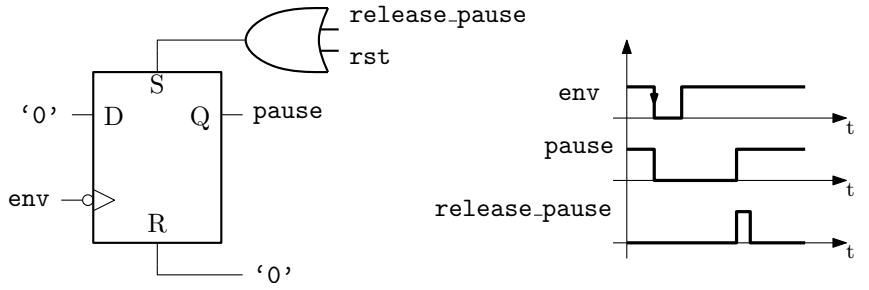

Figura 4.4: Circuito asincrónico de retención de pausas.

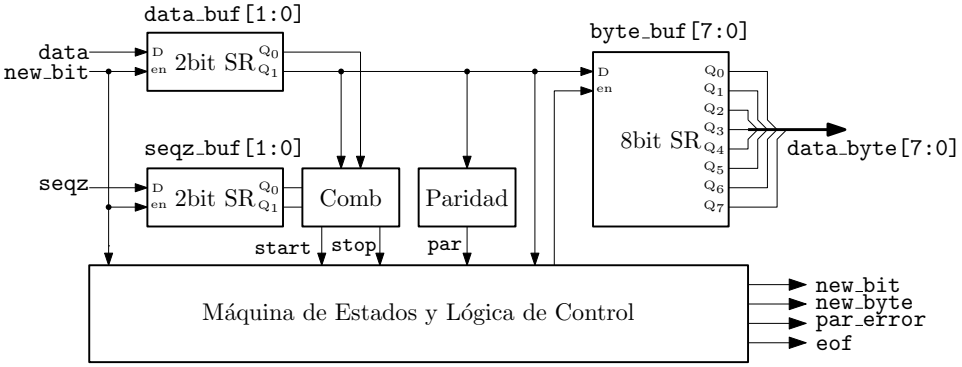

Figura 4.5: Diagrama en bloques del módulo `Frame Receiver`.

El circuito comienza con la salida `pause` en '1'. Al producirse el primer flanco descendente en `env` se dispara el flip-flop, la señal `pause` pasa a '0' y queda retenida durante todo el tiempo en que no se tiene señal de reloj. En el primer flanco ascendente del reloj, la señal `pause` es muestreada y luego se activa la señal `release_pause`, que libera la pausa forzando el estado del flip-flop a '1'. De esta forma es posible retener la pausas en la envolvente hasta que son interpretadas por el sistema digital.

4.2.2. Módulo `Frame Receiver`

El módulo `Frame Receiver` se encarga de interpretar los bits de la trama y dejar la información útil disponible a la salida. Para ello recibe los bits de salida del módulo `Bit Decoder` que corresponden a las secuencias decodificadas, junto con las señales que informan del estado, como son `new_bit`, `err`, etc.

En la figura 4.5 se presenta un diagrama en bloques del módulo `Frame Receiver`. Los bits decodificados por el módulo `Bit Decoder` ingresan en dos registros de desplazamiento de dos bits cada uno. En el primero, `data_buf`, se almacenan los bits decodificados y en `seqz_buf` se almacenan unos lógicos sólo si los bits fueron representados mediante una secuencia Z.

El objetivo de estos pequeños buffers es poder detectar el símbolo de fin

de comunicación que, como dice el estándar (sección 1.3.2), está formado por un cero lógico seguido de una secuencia Y. En un principio se podría simplificar la definición diciendo que el fin de comunicación está representado por dos bits '0', ya que la secuencia Y también representa al cero lógico. Sin embargo se debe notar que la combinación de secuencias dada por el estándar no puede darse nunca durante la transmisión de datos, ya que cualquier cero lógico tiene que estar representado por una secuencia Y o bien Z en caso de que el bit anterior haya sido un '0'. Entonces, un cero lógico seguido de una secuencia Y puede darse sólo en dos casos:

- El último bit transmitido fue un '0', en cuyo caso el cero lógico queda representado por una secuencia Z y entonces «Fin» = ZY.
- El último bit transmitido fue un '1', y por lo tanto el cero lógico es representado por una secuencia Y y entonces «Fin» = YY.

En ambos casos la secuencia de bits representados es '00', sin embargo, de tratarse de dos ceros dentro de la trama, con información, las secuencias utilizadas serían ZZ en el primer caso, e YZ en el segundo.

Teniendo en cuenta las diferencias en el orden de las secuencias Z en uno y otro caso se desarrolló el sistema de detección del símbolo «Fin». El circuito combinacional de la figura 4.5 activa la señal de `stop`, que indica que fue encontrado el símbolo de fin de comunicación, sólo cuando el buffer de datos contiene la secuencia '00' y `seqz_buf` contiene '10' o '00'.

De forma similar se detecta el bit de «Inicio» de comunicación, que está representado por una secuencia Z. La señal `start` se activa cuando `data_buf[0]=1b'0` y `seqz_buf[0]=1b'1`.

La salida del buffer de datos está conectada directamente al buffer de salida `byte_buf`, donde se almacena la carga útil de la trama. La lógica de control habilita de forma selectiva el buffer de salida de forma tal de descartar los bits de «Inicio», «Fin» y «Paridad», y almacenar allí solo los datos contenidos en la trama. Los bits de `byte_buf` se desplazan de MSB a LSB, por lo tanto, en caso de recibir una trama corta que contiene sólo comandos de 7 bits (REQA, WUPA,...), éstos quedarán almacenados en los bits más significativos de la salida, `data_byte[7:1]`.

Cada nuevo bit que se almacena en el buffer de salida se señaliza con la salida `new_bit`. Del mismo modo, cuando se completan los ocho bits de `data_byte`, se activa la señal `new_byte`; en caso de producirse un error de paridad se activa `par_error`; y por último, al detectar el símbolo «Fin» se activa la señal `eof`. Todas estas señales son pulsos de un ciclo de reloj de duración y además forman parte de la salida de datos del circuito integrado.

La señal `new_bit` es útil en el caso de recibir una trama anti-colisión, que puede estar dividida en cualquier bit. En este caso el sistema externo al chip puede recibir los bits de a uno, leyendo el dato de `data_byte[7]` cada vez que `new_bit` es igual a '1'; o puede recibir los datos de a bytes, y en el

b_n	P_n	P_{n+1}	Paridad
0	0	0	Par
0	1	1	Impar
1	0	1	Impar
1	1	0	Par

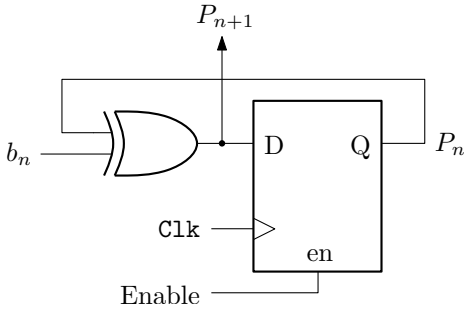

Figura 4.6: Circuito generador de bit del bit de paridad.

byte dividido contar la cantidad de bits con `new_bit` y leer sólo los datos válidos de `data_byte`.

En una trama estándar, luego de recibidos los ocho bits de datos se debe chequear el bit de paridad. El estándar dice que el bit de paridad debe ser tal que la cantidad de unos lógicos en la dupla `[data,P]` debe ser impar. Para corroborar que esto se cumpla en las tramas recibidas, se implementó un circuito generador de paridad como el de la figura 4.6. Por cada bit b_n que es recibido, se realiza la operación $P_{n+1} = b_n \oplus P_n$, donde P_n es el resultado de la operación anterior y su valor inicial es '0'. Observando la tabla de verdad, el resultado de esta operación es '1' si la cantidad de bits recibidos en estado '1' es impar. Por ejemplo, si $P_n = 0$ y se recibe un bit en '1', el resultado de la XOR es '1', indicando que la cantidad de unos recibidos hasta el momento es impar. Si se recibe otro '1', entonces $1 \oplus 1 = 0$, lo que indica que se tiene una cantidad par de unos lógicos.

Este procesamiento se realiza con los bits de datos recibidos, dejando afuera los bits de «Inicio» y «Fin». Al recibir el bit de paridad se compara con el calculado hasta el momento por el circuito generador de paridad. Si ambos bits coinciden, los datos fueron recibidos correctamente y se sigue adelante con el procesamiento. Si los bits no coinciden se activa la señal `par_error`, informando de esta manera al sistema externo que utiliza los datos.

4.3. Transmisión de Datos

Para transmitir datos hacia el lector, primero se carga el buffer de transmisión `Tx buffer` de a un bit por vez utilizando las entradas `tx bit` y `tx new bit` de la figura 4.1. El módulo `Tx buffer` no es más que un registro de desplazamiento con entrada serie y salida paralelo, que se habilita con la señal `tx new bit`, más un contador de 3 bits que cuenta los N bits que ingresan al registro. Por cada flanco ascendente de reloj, si la señal `tx new bit` está en alto se desplaza un bit y se incrementa la cuenta.

Los datos y la cuenta del buffer ingresan al módulo `Frame Sender` y este

se encarga de encapsular la información en la trama, que luego es enviada a través de la interfaz de RF. Cuando se recibe la señal `tx transmit`, que indica que se debe transmitir el contenido del buffer, los datos y la cuenta son pasados a registros internos de `Frame Sender` y se reinicia el contador, dejando el buffer listo para una nueva carga. Esto permite que el dispositivo externo que controla al circuito integrado pueda cargar el buffer con el siguiente bloque de datos aunque exista una transmisión en curso. De esta forma se agiliza la transmisión sin la utilización de un buffer de mayor tamaño.

La transmisión de datos se inicia cuando que se cumple el tiempo de demora entre tramas FDT (*Frame Delay Time*). El FDT es controlado por el módulo `Frame Delay Counter`, que cuenta la cantidad de ciclos de reloj transcurridos desde que se recibe el símbolo de «Fin» de comunicación, señalizado por `rx eof`.

Los bits a enviar son codificados en código *Manchester* por el módulo `Bit Coder`, quién también agrega la subportadora y maneja las llaves que actúan en la modulación de carga (Estas llaves se verán más adelante en la sección 5.2.2). El módulo `Bit Coder` se encarga además de controlar la duración de cada bit, que es de 128 ciclos de portadora, y una vez finalizada la transmisión de un bit informa a `Frame Sender` que se encuentra disponible para enviar el siguiente bit. La salida `coded_out` se conecta directamente al bloque Modulador y la información codificada es transmitida hacia el lector.

Las tramas enviadas comienzan con el bit de «Inicio» de comunicación, luego se transmiten los N bits que fueron cargados en el buffer y por último el bit de paridad. Si bien las tramas estándar (figura 1.6b) contienen bloques de 8 bits más el bit de paridad, es imprescindible contar con la posibilidad de enviar bloques de menor tamaño para poder contestar a las tramas anti-colisión (figura 1.6c) de las que en principio no se sabe en que punto pueden ser divididas.

Un ciclo de reloj antes de que finalice la transmisión del bit de paridad, `Frame Sender` activa la señal `tx ready` que informa al dispositivo controlador que el circuito está listo para recibir una nueva orden de transmisión. Si no se recibe la orden, la trama finaliza con el símbolo de «Fin» y `Frame Sender` vuelve al estado de reposo. Si la orden es recibida en el instante en que termina de enviarse el último bit, la transmisión previa continúa y no se envía un nuevo bit de «Inicio» sino que se continúa con la trama anterior.

Esto es útil para las tramas estándar o anti-colisión, que contienen más de un byte, donde los bits deben transmitirse de forma continua para no dividir la trama, lo que podría ocasionar errores en la comunicación. El mecanismo de transmisión implementado maneja de forma autónoma el inicio y fin de las tramas, decidiendo automáticamente si debe agregar o no el bit de «Inicio» de comunicación basado en el instante en que recibe la señal `tx transmit`.

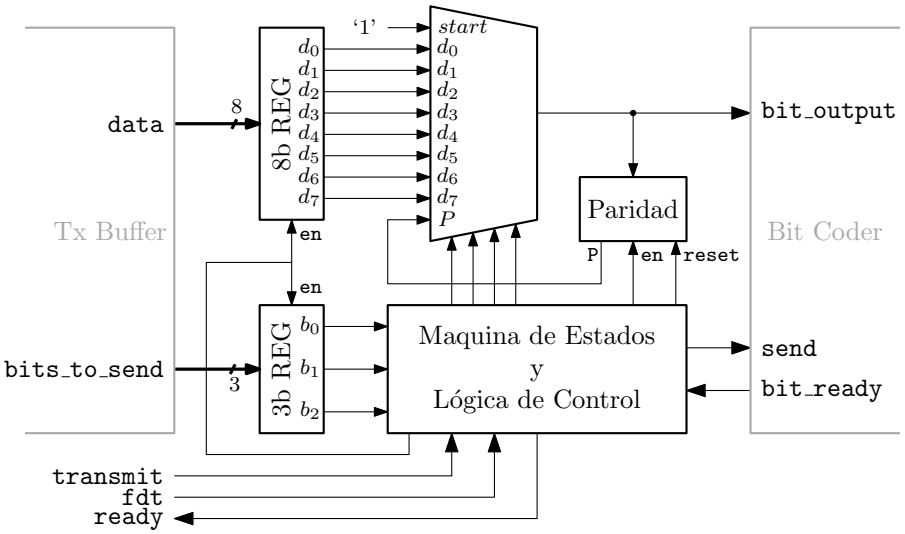

Figura 4.7: Diagrama del módulo `Frame Sender`.

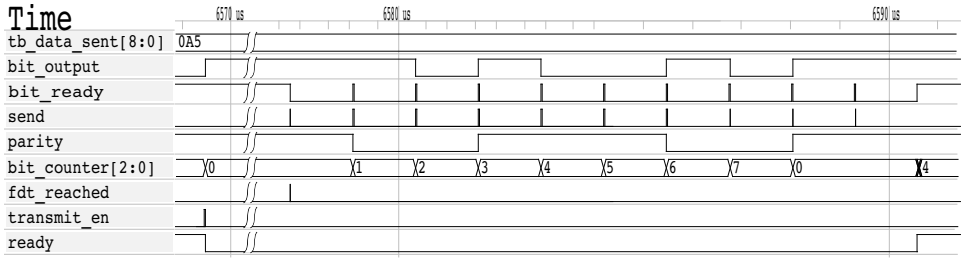

Figura 4.8: Diagrama de tiempos del módulo `Frame Sender`.

4.3.1. Módulo `Frame Sender`

En la figura 4.7 se muestra un diagrama del módulo `Frame Sender`. La maquina de estados controla un par de registros —donde se almacenan los datos a transmitir y la cantidad de bits—, un multiplexor y un generador de paridad como el de la figura 4.6.

Al recibir la señal `transmit`, la máquina de estados carga los buffers de entrada con los datos del módulo `Tx Buffer` e inicializa un contador interno que lleva la cuenta de la cantidad de bits enviados hasta el momento. Luego espera a recibir la señal `fdt`, que le indica que el tiempo FDT se ha cumplido, y que puede iniciar la transmisión. En el diagrama de tiempos de la figura 4.8 se muestra un ejemplo de transmisión. Allí pueden seguirse los pasos realizados por `Frame Sender` junto con el diagrama de estados de la figura 4.9.

Una vez recibida la señal `fdt` comienza la transmisión direccionando el primer bit con el multiplexor, que siempre es el bit de «Inicio» de comunicación, y enviando un pulso a través de `send` al módulo `Bit Coder`. Este

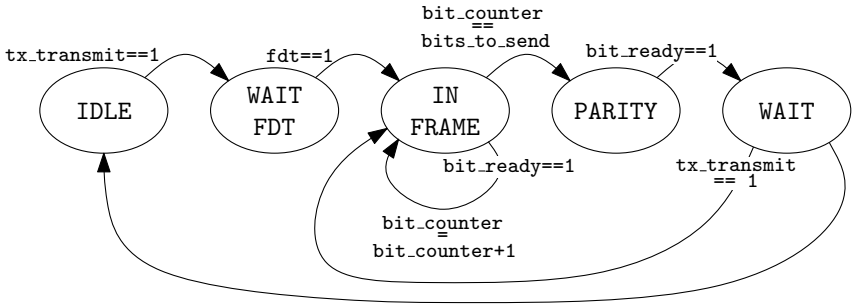

Figura 4.9: Diagrama de estados del módulo `Frame Sender`.

módulo se encarga de codificar el bit y de generar la señal de manejo del modulador de carga.

Cuando `Bit Coder` finaliza la transmisión, activa la señal `bit_ready`, informando que se encuentra disponible para enviar otro bit. Al recibir esta señal la máquina de estados direcciona el bit siguiente, d_0, nuevamente envía un pulso a través de `send`, y luego espera a que se active la señal `bit_ready` antes de continuar. Cuando finaliza el envío de d_0 se continua con d_1, luego d_2 y así sucesivamente.

Como la cantidad de bits en el registro de entrada puede ser menor a ocho, el proceso se repite hasta que la cantidad de bits enviados, o lo que es lo mismo, la dirección del multiplexor, sea igual a la cantidad de bits cargados en `Tx Buffer`. Cuando esto se cumple se pasa automáticamente a la dirección del bit de paridad P. En caso contrario, se continúa incrementando la dirección hasta enviar todos los bits y llegar a P, lo que sucederá cada vez que se envíe una trama estándar con ocho bits de datos.

El bit de paridad se calcula del mismo modo que en el módulo `Frame Receiver`, sólo que en este caso se utilizan para el cálculo los bits de salida del multiplexor. Al finalizar la transmisión del bit de «Inicio», el bloque generador de paridad es reiniciado por la máquina de estados. Luego, a medida que los bits de datos son seleccionados, se hacen ingresar al generador hasta que finalmente, cuando se selecciona P, se toma su salida y se envía en la posición del bit de paridad. De esta forma se cumple siempre el requerimiento del estándar de enviar los datos con paridad impar.

Por último, cuando `Bit Coder` informa que finalizó la transmisión del último bit, la máquina de estados activa la señal `ready` que informa al circuito o dispositivo externo que finalizó la transmisión. Esto sucede en realidad un ciclo de reloj antes de que el último bit termine y permite que, en caso de recibir una nueva orden de transmisión durante el ciclo de reloj siguiente a la activación de `ready`, se continúe la trama enviando un nuevo bloque de datos más paridad sin bit de «Inicio». Cuando esto ocurre, se cargan los registros de entrada con los datos de `Tx Buffer`, se direcciona directamente el bit d_0 y se repite el proceso de envío.

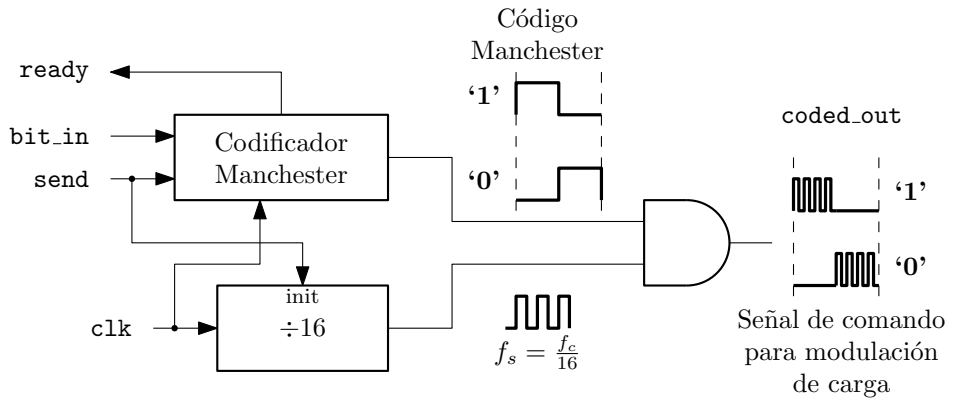

Figura 4.10: Diagrama del módulo `Bit Coder`.

Si `transmit` no se activa durante el ciclo de reloj siguiente al flanco ascendente de `ready`, la trama se da por terminada y la máquina de estados vuelve al estado de reposo.

4.3.2. Módulo `Bit Coder`

El módulo `Bit Coder` se encarga de codificar los bits que deben ser enviados a través de la interfaz de RF como especifica el estándar (ver tabla 1.3) y generar la señal de comando del modulador de carga.

En la figura 4.10 se muestra un diagrama de los componentes internos del módulo. Allí se observa que el mismo está compuesto por un bloque codificador *Manchester*, un divisor de frecuencia y una compuerta AND. La salida `coded_out` es la señal de comando de la llave que conecta la carga en el bloque modulador analógico. Cuando `coded_out` es igual a '1' la llave se cierra y por lo tanto se dice que la portadora está *cargada*. Por otro lado, cuando `coded_out` es '0' la llave está abierta y no se carga a la señal de RF.

El Codificador Manchester se encarga de generar el código a partir de los bits de entrada. En el estado de reposo su salida se mantiene en '0' y por lo tanto la salida del módulo `coded_out` también. Al recibir un pulso en la entrada `send` produce en su salida el código Manchester correspondiente al estado de `bit_in`. Cada bit codificado tiene la duración especificada por el estándar, es decir, 128 ciclos de reloj. Al finalizar con el código activa la señal `ready` y vuelve al estado de reposo.

Por otro lado, el divisor de frecuencia divide la señal de reloj de 13,56 MHz por 16 para generar la señal sub-portadora de 848 kHz. El estándar especifica que la modulación de carga al inicio de cada bit debe tener una fase definida, es por ello que se incluyó en el divisor de frecuencia la entrada `init`. Según la norma, los bits deben comenzar con el estado *cargado* de la señal de RF y entonces la entrada `init` sirve para forzar el estado de la salida del divisor de frecuencia a '1' cada vez que comienza la transmisión de un bit.

Finalmente, la operación AND de los bits codificados en código Manchester con la señal sub-portadora da como resultado las secuencias definidas en la tabla 1.3.

Un detalle importante es que `Bit Coder` informa que finalizó la transmisión de un bit un ciclo antes de que la transmisión realmente termine. Esto es debido a que se debe permitir continuar la con la trama con un nuevo pulso en `send` sin interrumpirla ni dejar espacios entre bits.

4.4. Verificación funcional

La verificación del funcionamiento fue realizada módulo por módulo y luego del sistema completo utilizando para ello el conexionado que realiza el eco de un byte.

Los *testbenchs*[2] desarrollados permitieron generar las señales de entrada del sistema digital y enviar datos con tan solo llamar a un *task* de *Verilog*, pasando como argumento el valor a enviar.

Los *tasks* de más bajo nivel generan el patrón de envolvente de la portadora correspondiente a las secuencias X, Y o Z. El patrón de envolvente se utiliza para generar la señal `pause` y a la vez, a través de una operación AND lógica, se detiene el reloj del sistema. De esta forma se simula lo que sucede realmente al arribar una pausa en la portadora. También se codificaron *tasks* para generar las secuencias de «Inicio» y «Fin» de comunicación y luego se escribió un *task* de más alto nivel que utiliza a los anteriores para enviar cualquier bytes de datos pasado como parámetro.

Con los *tasks* desarrollados se pudieron enviar las tramas de entrada al sistema digital y verificar la respuesta del mismo. Para probar el sistema de recepción de información se desarrollo un *testbench* que envía los bytes de 8'h00 a 8'hFF, de a uno por vez y armando la trama completa, es decir, «Inicio» + datos + «Paridad» + «Fin» y comprueba automáticamente que a la salida de `Frame Receiver`, en el registro `byte_buf`, quede almacenado el valor correcto cuando se genera el pulso en `new_byte`. Las pruebas realizadas de esta forma permitieron depurar el diseño hasta obtener un sistema de recepción de información libre de errores.

Por otro lado, para el sistema de transmisión de información el *testbench* desarrollado constó del envío de algunos bytes típicos, que fueron cargados manualmente en el buffer de transmisión, y la verificación se hizo mediante el análisis del diagrama de tiempos del sistema. En este caso no se desarrolló un test automatizado debido a la dificultad de leer y verificar la señal `coded_out`, que está formada por el código *Manchester* modulado con la sub-portadora, mediante código en *Verilog*. Sin embargo, también se hizo la prueba exhaustiva de enviar los 256 bytes de datos posibles utilizando para ello el conexionado de la interfaz de recepción con la de transmisión, como se verá a continuación.

[2]Bancos de prueba codificados en lenguaje *Verilog*

Figura 4.11: Simulación del bloque digital cuando el dispositivo es utilizado en modo *eco*.

4.4.1. Eco de un byte

En la figura 4.1 se muestra en líneas punteadas como puede conectarse directamente la salida del bloque receptor con la entrada del bloque transmisor para obtener un *tag* autónomo que realiza el eco de un byte, es decir, devuelve al lector el byte de datos recibido. Al conectar el dispositivo de esta forma, los bits recibidos en `rx_bit` ingresan directamente al buffer de transmisión. Al finalizar la trama y detectarse el símbolo de «Fin» se activa la señal `rx_eof`, que a su vez sirve de señal de inicio de transmisión a través de `tx_transmit`.

El diseño puede retransmitir sólo un byte de datos, ya que el buffer de transmisión tiene el tamaño de un byte y se sobreescribe si se recibe una cantidad de datos mayor.

En la figura 4.11 se muestra una simulación a nivel lógico del bloque digital con la conexión que produce el eco. Allí se observa como el dispositivo recibe el byte de datos, 8h'53 en este caso, a la vez que lo carga en el buffer e inicia la transmisión luego de cumplido el FDT. Las señales `tb_rx_bit` y `tx_bit` son los bits recibidos y a cargar en el buffer, mientras que `tb_rx_new_bit` indica que el bit es un nuevo bit válido para la lectura y `tx_new_bit` es la señal que habilita la carga del buffer de transmisión en el siguiente flanco ascendente de reloj.

Se debe notar que los bits se reciben en orden *LSB First*, es decir, se recibe primero el bit menos significativo. El buffer de transmisión fue diseñado para ser cargado también en ese orden, lo que evita el uso de componentes externos.

En el *testbench* desarrollado para verificar la funcionalidad del modo *eco* se enviaron los 256 bytes de datos posibles al sistema digital y todos fueron reenviados a través de `coded_out` correctamente.

4.5. Implementación: Síntesis y *Place&Route*

La descripción del hardware en lenguaje *Verilog* fue sintetizada utilizando el software *Synopsys Design Compiler* [1] junto con la biblioteca de celdas digitales de la *Oklahoma State University* (OSU), celdas que fueron diseñadas para el proceso de fabricación C5N de *ON Semiconductor*.

La biblioteca incluye compuertas lógicas NAND, NOR, XOR, inversores, AOI, OAI, etc., junto con buffers especiales para el árbol de distribución de reloj, multiplexores y flip-flops tipo D disparados por flanco ascendente/descendente. Todas las celdas están caracterizadas en cuanto a tiempos de propagación para todos los caminos posibles entre entradas y salidas, cuentan con características dinámicas de las salidas, como son los tiempos de crecimiento/decrecimiento y consumo de potencia dinámica según la capacidad de carga.

La herramienta de síntesis utiliza la caracterización de las compuertas junto con restricciones impuestas al diseño para optimizar los tiempos de retardo y el área ocupada por el circuito. La frecuencia de operación es la primer restricción de todo circuito sincrónico y en este caso es de 13,56 MHz. Sin embargo, para contar con un margen de seguridad en caso de que el retardo de las compuertas fuese mayor al esperado por variaciones del proceso, se impuso como restricción una frecuencia de operación de 15 MHz.

Por otro lado, para reducir el consumo de potencia se activó la opción de la herramienta de síntesis que hace uso de la técnica *clock gating*. Esta técnica agrega una pequeña lógica en la entrada de reloj de los flip-flops que permite activar o desactivar esa señal. Cuando un FF debe mantener el estado de su salida durante varios ciclos de reloj, como por ejemplo los FF siguientes al primero de un contador binario, se desactiva la señal de reloj en su entrada para evitar el consumo de potencia dinámica debido a la transición del reloj de un estado a otro.

El trazado físico del diseño (*Place&Route*) se realizó con el software *Synopsys IC Compiler* [5]. En este caso se le debe indicar al programa la forma o tamaño deseado. Como el área ocupada en general se desea que sea la mínima posible, se suele definir entonces la relación de aspecto alto/ancho del trazado deseado. También se le deben indicar las posiciones de las entradas y salidas del bloque, que luego serán conectadas con el resto del los circuitos.

El programa ubica las compuertas optimizando las posiciones según el conexionado. Luego genera un árbol de reloj y finalmente realiza el conexionado de todas las compuertas. El *layout* final del bloque digital puede verse en la figura 4.12.

Las herramientas generan una serie de reportes del área ocupada, el tiempo de retardo del camino crítico y los registros en que se utilizó la técnica de *clock gating*. En la tabla 4.1 se presenta un resumen del área ocupada por cada uno de los módulos del diseño luego de la síntesis lógica y sin tener en cuenta el árbol de distribución de reloj ni el área necesaria

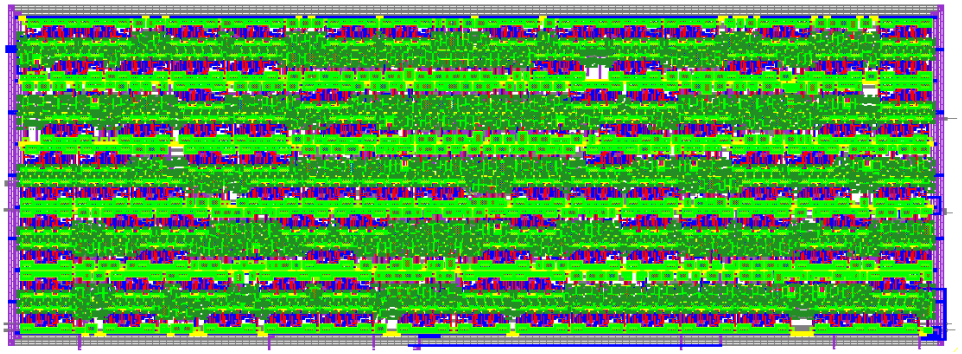

Figura 4.12: *Layout* final del bloque digital. Sus dimensiones son de 900 µm × 330 µm.

Módulo	Area $[\mu m^2]$
`bit_coder`	33669 (15,1 %)
`bit_decoder`	37791 (16,9 %)
`frame_delay_counter`	28656 (12,8 %)
`frame_receiver`	36972 (16,6 %)
`frame_sender`	57447 (25,8 %)
`pause_latch`	1800 (0,1 %)
`rst_sync`	3600 (0,2 %)
`tx_buffer`	22563 (10,1 %)
Total 8 Módulos	222498

Tabla 4.1: Área utilizada por cada uno de los módulos luego de la síntesis.

para el conexionado de las compuertas. El módulo `frame_sender` es el de mayor tamaño, mientras que los demás son de tamaños similares, salvo `pause_latch` y `rst_sync`[3], que al contener un registro el primero y dos el segundo son de tamaño despreciable.

En la tabla 4.2 puede verse un resumen del reporte de la herramienta de síntesis luego de aplicar la técnica de *clock gating*. De los 95 registros del diseño, a 70 se les pudo aplicar esta técnica para reducir el consumo. El software también estima la potencia consumida por las compuertas trabajando a la frecuencia establecida. Luego de aplicar la técnica de *clock gating* la potencia dinámica estimada por el sintetizador se redujo un 30 %, de 3,43 mW a 2,39 mW.

Sin embargo, el agregado de las celdas de *clock gating*, cuyo esquemático puede verse en la figura 4.13, tiene el inconveniente de que disminuye el tiempo disponible para el arribo de la señal proveniente de la lógica. Esto es debido a que el latch del esquemático actúa cuando el reloj está en nivel bajo,

[3]Este módulo es el circuito sincronizador de la señal de reset.

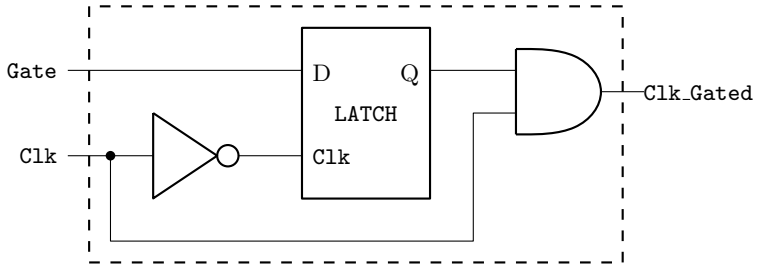

Figura 4.13: Celda de *clock gating* construida por el sintetizador.

Resumen de *Clock Gating*	
Cantidad total de registros	95
Cantidad de celdas de *clock gating*	13
Cantidad de registros con *clock gating*	70 (73,68 %)
Cantidad de registros sin *clock gating*	25 (26,32 %)

Tabla 4.2: Resultado arrojado por la herramienta de síntesis luego de aplicar la técnica de *clock gating* al diseño.

mientras que los flip–flops del resto del diseño trabajan por flanco ascendente. Entonces, luego de un flanco ascendente de reloj, la entrada `Gate` tiene sólo medio período para establecerse antes de que actúe el latch en el siguiente nivel bajo. Esto afectará el camino crítico como se verá a continuación.

Finalmente, utilizando las características de retardos y capacidad de entradas y salidas, la herramienta calcula el retardo de las señales entre registros para ver que se cumpla con la frecuencia de operación requerida. En caso de que el retardo sea mayor al período de reloj el software optimiza las dimensiones de las compuertas o agrega buffers, incrementando el área ocupada, para intentar cumplir con la frecuencia de operación indicada. El parámetro que se toma como referencia para saber si se cumple o no es el *slack time*, que es el tiempo entre que arriba la señal de datos a la entrada de un registro hasta que arriba el flanco de reloj. Si el *slack time* fuese negativo el circuito no podría funcionar. Luego de la síntesis se genera un informe con los tiempos de retardo de cada uno de los caminos posibles entre registros. Al camino con menor *slack time* se lo denomina *camino crítico* y es el primero que debe optimizarse.

En la tabla 4.3 se muestra el reporte del camino crítico para el diseño. El camino parte de la entrada `tx_transmit` y termina en un FF del módulo `Frame Sender`. Una de las restricciones impuestas para forzar un peor caso es que todas las entradas tengan un retardo de 10 ns y por este motivo es que en la tabla el *input delay* comienza con ese valor. El software suma el retardo de las compuertas hasta llegar al registro al final del camino. El *data arrival time* es el resultado de la suma, que se compara con el *data required*

Punto	Retardo [ns]
Input external delay	10.00
frsen/U13/Y (INVX2)	10.07
frsen/U65/Y (NOR2X1)	10.28
frsen/U74/Y (NAND2X1)	10.70
frsen/U81/Y (OAI22X1)	10.82
Data arrival time	10.82
Clock clk' (rise edge)	33.00
Clock network delay (ideal)	33.00
Data required time	33.00
Data arrival time	-10.82
Slack (MET)	22.18

Tabla 4.3: Retardos del camino crítico: Desde `tx_transmit` hasta `frsen/clk_gate_bits_to_send_ret_reg/latch`.

time generalmente dado por la frecuencia de trabajo. Debido a la utilización de las celdas de *clock gating*, que como se dijo reducen el tiempo disponible para el establecimiento de la lógica, el *data required time* fue de la mitad del período de reloj. Finalmente, el *slack* calculado para el camino crítico fue de 22,18 ns y no fue necesario intervenir en la síntesis.

Luego, mediante una simulación a nivel de transistor se obtuvo la corriente promedio de consumo del bloque digital, que fue de 325 µA a 3 V, lo que da una potencia ligeramente menor a la estimada por el sintetizador.

Bibliografía

[1] *Design Compiler Graphical.* http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCGraphical/Pages/default.aspx.

[2] *DIP40 from Kyocera.* http://www.mosis.com/pages/Technical/Packaging/Ceramic/menu-pkg-ceramic.

[3] *GNU Octave Web Page.* https://www.gnu.org/software/octave/.

[4] *GTKwave Web Page.* http://gtkwave.sourceforge.net/.

[5] *IC Compiler Place and Route System.* http://www.synopsys.com/Tools/Implementation/PhysicalImplementation/Pages/ICCompiler.aspx.

[6] *Icarus Verilog Web Page.* http://iverilog.icarus.com/.

[7] *LTSpice.* http://en.wikipedia.org/wiki/LTspice.

[8] *Mentor Graphics IC Design.* http://www.mentor.com/products/ic_nanometer_design/custom-ic-design/.

[9] *MOSIS Integrated Circuit Fabrication Service.* http://www.mosis.com/.

[10] *Oklahoma State University System on Chip Design Flows.* http://vlsiarch.ecen.okstate.edu/?page_id=12.

[11] *Identification cards — Test methods — Part 6: Proximity cards*, 2000.

[12] *Identification cards — Physical characteristics*, 2003.

[13] *IEEE Standard for Verilog Hardware Description Language.* IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001), páginas 1–560, 2006.

[14] *Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 1: Physical characteristics*, 2007.

[15] *Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 4: Transmission protocol*, 2007.

[16] *Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 3: Initialization and anticollision*, 2008.

[17] *Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 2: Radio frequency power and signal interface*, 2009.

[18] Alcalde Bessia, Fabricio, Diego Fanego y Guillermo Makar: *Diseño de un TAG RFID integrado en un proceso CMOS de* 0,5μm. En *CASE 2012, Congreso Argentino de Sistemas Embebidos*, página 120, August 2012, ISBN 978-987-9374-82-5. `http://www.sase.com.ar/2012/congreso-argentino-de-sistemas-embebidos-case-2012/`.

[19] Baker, Jacob: *CMOS Circuit design, layout and simulation.* Wiley-Interscience, 2005.

[20] Finkenzeller, Klaus: *RFID Handbook.* Wiley, 3ª edición, 2010.

[21] Gray, P., P. Hurst, S. Lewis y R. Mayer: *Analysis and design of analog integrated circuits.* John Wiley, 2001.

[22] Gudnason, Gunnar y Erik Bruun: *CMOS circuit design for RF sensors.* Kluwer Academic Publishers, 2002.

[23] Hastings, Alan: *The art of analog layout.* Prenctice Hall, 2001.

[24] Himanshu, Bhatnagar: *ADVANCED ASIC CHIP SYNTHESIS Using Synopsys® Design Compiler™ Physical Compiler™ and PrimeTime®.* KLUWER ACADEMIC PUBLISHERS, 2002.

[25] Kamon, M., M.J. Tsuk y J.K. White: *FASTHENRY: a multipole-accelerated 3-D inductance extraction program.* Microwave Theory and Techniques, IEEE Transactions on, 42(9):1750–1758, Sept 1994, ISSN 0018-9480.

[26] Mandolesi, P., G. San Martín y Julián P.: *RFID Front-End in 0.5um Standard CMOS process: Experimental results.* En *Proceedings of the Argentine School of Micro-Nanoelectronics, Technology and Applications 2008*, 2008.

[27] Marechal, Catherine y Dominique Paret: *Optimization of the law of variation of shunt regulator impedance for Proximity Contactless Smart Card Applications to reduce the loading effect.* Informe técnico, Laboratoire LRIT – ESIGETEL.

[28] Zhu, Zheng, Ben Jamali y Peter H. Cole: *Brief Comparison of Different Rectifier Structures for RFID Transponders.* Informe técnico, Auto-ID lab at University of Adelaide.