UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de arquitectura para un circuito con tecnología nanométrica y validación de aspectos y especificaciones técnicas de un proceso de fabricación industrial en silicio

Trabajo de graduación presentado por Pablo Alejandro Ortiz Barillas
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de arquitectura para un circuito con tecnología nanométrica y validación de aspectos y especificaciones técnicas de un proceso de fabricación industrial en silicio

Trabajo de graduación presentado por Pablo Alejandro Ortiz Barillas
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

Vo.Bo.:

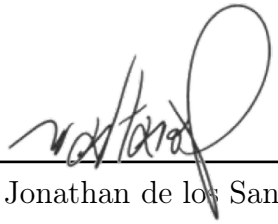
(f) _____

MSc. Carlos Esquit

Tribunal Examinador:

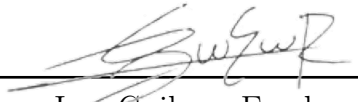
(f) _____

MSc. Carlos Esquit



(f) _____

Ing. Jonathan de los Santos



(f) _____

Ing. Guilmar Escobar

Fecha de aprobación: Guatemala, 05 de diciembre de 2019.

Índice

Lista de figuras x

Lista de cuadros xi

Resumen xiii

Abstract xv

1. Introducción 1

2. Antecedentes 3

3. Justificación 7

4. Objetivos 9
4.1. Objetivo general 9
4.2. Objetivos específicos 9

5. Marco teórico 11
5.1. ¿Qué son los chips? 11
5.2. Método de fabricación 12
5.3. Mutiple project wafers (MPW) 15
5.4. Tecnología de fabricación 15
5.5. Reglas de diseño 16
5.6. Flujo de diseño 17
5.6.1. Especificaciones y diseño de la estructura en RTL 20
5.6.2. Síntesis 21
5.6.3. Simulación del diseño lógico o validación de la arquitectura . . 22
5.6.4. Diseño físico o Layout 22
5.6.5. Revisión de reglas de diseño y comparación del diseño físico contra el esquemático 22
5.6.6. Caracterización del diseño físico 22

5.6.7. Validación del diseño físico . . . 23
5.7. Registros . . . 23
5.7.1. Flip-Flops . . . 23
5.7.2. Transferencia de información dentro y fuera de los registros . . 23
5.8. Máquina de estados finitos . . . 24
5.9. Conversor paralelo a USB . . . 26
5.9.1. Protocolos de transmisión serial . . . 26
5.9.2. Convertidor paralelo a serial . . . 27
5.9.3. Convertidor UART a USB . . . 27
5.10. Python . . . 28
5.10.1. Programación orientada objetos y derivados . . . 28
5.10.2. Librerías . . . 28
5.10.3. Comunicación serial . . . 29
5.10.4. Text to Speech . . . 29

6. Empresas fabricantes de semiconductores 31
6.1. Búsqueda de fabricantes . . . 31
6.2. Fabricantes de semiconductores . . . 32
6.2.1. Empresas intermediarias . . . 32
6.3. Fabricación con MOSIS . . . 33
6.3.1. Proceso de fabricación . . . 33
6.3.2. Reglas de de diseño . . . 34
6.3.3. Eventualidades . . . 34
6.4. Fabricación con Europractice . . . 35
6.4.1. Proceso de fabricación . . . 36
6.5. Reglas de de diseño . . . 37

7. Descripción del sistema implementado 39
7.1. Análisis de los requerimientos funcionales . . . 39
7.1.1. Organización temporal de los datos . . . 39
7.1.2. Diagrama de bloques del sistema . . . 40

8. Diseño de la arquitectura 43
8.1. Descripción de la máquina de Estados Finitos . . . 43
8.2. Diseño de la máquina de estados finitos . . . 44
8.2.1. Selección del modelo de la FSM . . . 44
8.2.2. Calculo de la cantidad de registros . . . 45
8.2.3. Función de transiciones de estado . . . 45
8.2.4. Función de salidas . . . 48
8.3. Implementación en Verilog . . . 51
8.4. Validación arquitectura . . . 52

9. Diseño del sistema que implementa el IC - Prototipo 1 53
9.1. Topología en alto Nivel . . . 53
9.1.1. Arduino . . . 54
9.1.2. Programa de lectura de texto . . . 54
9.2. Pruebas y resultados . . . 55

9.3. Obstáculos 57

10. Diseño del sistema que implementa el IC - Prototipo 2 **59**
10.1. Cambios con respecto al Prototipo 1 59
10.1.1. Circuito generador de señal 59
10.1.2. Conversión paralelo-serial 60
10.2. Topología en alto nivel - Prototipo 2 60
10.2.1. Señal de reloj 60
10.2.2. Divisor de frecuencia 61
10.2.3. FSM 61
10.2.4. Conversor paralelo a serial 61
10.2.5. Modulo UART-USB 62
10.3. Pruebas y resultados 63
10.4. Obstáculos 64

11. Conclusiones **67**

12. Recomendaciones **69**

13. Bibliografía **71**

14. Anexos **73**
14.1. Ecuaciones de la FSM 73
14.1.1. Ecuaciones de transición de estados 73
14.1.2. Ecuaciones de la función de salida 74
14.2. Código de la FSM simulada con Arduino para el Prototipo 2 77

Lista de figuras

1. Aplicación de capa foto-resistente 13
2. Empaquetado del procesador 14
3. Localización de los diseños en un MPW 15
4. Simplificación de las reglas de diseño basadas en lambda 17
5. Evolución de flujos de diseño en función de la tecnología 18
6. Diagrama del flujo de diseño en alto nivel para la fabricación de un Chip 20
7. Diagrama del flujo de diseño para la fabricación de un chip 21
8. Pasos para síntesis lógica 21
9. Modelo en alto nivel de la FSM de Mealy y Moore 25
10. Diagrama de estados y su correspondiente tabla de transiciones de estados 25
11. Diagrama de conexión entre interfaces UART 27
12. Trama de 7-bits que representa la letra E en codificación ASCII . . . 27
13. Conversor paralelo a serial 28

14. Logo MOSIS 33
15. Design Kit para el proceso de 28nm por TSMC 34
16. Logo Europractice 35
17. Empresas y tecnologías disponibles a través de *Mini@sic* 36
18. Precios (en euros) y tecnologías disponibles con TMSC 37

19. Ejemplo de organización de tramas 40
20. Diagrama de bloques del sistema 40

21. Diagrama de estados de la FSM 44
22. Fragmento de la implementación de la FSM diseñada 51
23. Resultados obtenidos de las pruebas a la arquitectura 52

24. Diagrama de bloques del Prototipo 1 53
25. Código del Arduino utilizado para el prototipo 1 54
26. Programa lector de texto para el Prototipo 1 56
27. Archivo tipo mp3 generado 57
28. Mensaje recibido a través del puerto serial 57

29. Diagrama de bloques del Prototipo 2 60
30. Diagrama convertidor paralelo-serial 62
31. Modulo conversor UART-USB 63
32. Implementación del Protitopo 2 en un Protobard 63
33. Mensaje formado por los caracteres generados por la FSM simulada . 64
34. Velocidad de transmisión de datos 65
35. Trama de datos enviada 65

Lista de cuadros

1. Evolución de la productividad del diseño 19

2. Características de la máquina de estados finitos 43
3. Fragmento 1 Tabla de estados 45
4. Fragmento 2 Tabla de estados 46
5. Fragmento 3 Tabla de estados 47
6. Fragmento 1 Tabla de salidas 48
7. Fragmento 2 Tabla de salidas 49
8. Fragmento 3 Tabla de salidas 50

Resumen

Este trabajo tiene como objetivo general diseñar la arquitectura del circuito integrado y documentar tanto el proceso de diseño como el proceso de comunicación y validar los parámetros y aspectos técnicos con el fabricante de semiconductores. Este objetivo se traza en el contexto del diseño y fabricación del primer circuito integrado diseñado por estudiantes de ingeniería en el país. Por ende, el diseño del circuito integrado será puramente digital, esto pues el trabajo se basará principalmente el seguimiento de los pasos del flujo de diseño para llevar acabo la fabricación de un circuito integrado con tecnología nanométrica.

La primer parte del trabajo consistió en la investigación de las empresas encargadas de fabricar circuitos integrados y la selección de la misma en base a ciertas características, aspectos y requerimientos tanto funcionales como técnicos.

El diseño de la arquitectura del circuito integrado consistió en una máquina de estados finitos (FSM) de 345 estados, la cual tendrá como función mostrar en sus 8 salidas el valor en código ASCII de caracteres que formarán un mensaje conmemorativo de la realización del proyecto. Además del diseño de esta arquitectura se realizó su validación con la ayuda de herramientas de diseño de circuitos integrados con tecnología nanométrica. Esta validación se realizó por medió del *software VCS*.

Finalmente se procedió a realizar el diseño del sistema que implementará al circuito integrado, este consistió de un circuito físico capaz de convertir los valores de los caracteres recibidos en su entrada paralela a una serie de bits organizados por tramas temporales y enviarlos mediante mediante un protocolo serial al puerto USB de la computadora para posteriormente procesar dichos caracteres y articular el mensaje con la ayuda de un programa realizado en el lenguaje Python.

Abstract

The purpose of this work is to design the integrated circuit architecture and document both the design process and the communication process and validate the parameters and technical aspects with the semiconductor manufacturer. This objective is outlined in the context of the design and manufacture of the first integrated circuit designed by engineering students in the country. Therefore, the design of the integrated circuit will be purely digital, this is because the work will mainly be based on the follow-up of the design flow steps to carry out the manufacture of an integrated circuit with nanometric technology.

The first part of the work consists in the investigation of the companies in charge of manufacturing integrated circuits and the selection of the same based on certain characteristics, aspects and functional and technical requirements.

The design of the integrated circuit architecture consists of a finite state machine (FSM) of 345 states, which will have the function of displaying in its 8 outputs the value in ASCII code of characters that will form a commemorative message of the completion of the project. In addition to the design of this architecture, it was validated with the help of integrated circuit design tools with nanometric technology. This validation was done through the *VCS software*.

Finally, we proceeded to carry out the design of the system that will implement the integrated circuit, this consists of a physical circuit capable of converting the values of the characters received in a parallel input to a series of bits organized by time frames and send them through a serial protocol to the USB port of the computer for later processing these characters and articulate the message with the help of a program made in the Python language.

CAPÍTULO 1

Introducción

Desde el momento de su invención hasta ahora los chips o circuitos integrados han sido y son parte de nuestras vidas, no hay dispositivo electrónico que no lo posea. Se han vuelto tan comunes que incluso ignoramos como funciona, como se fabrica llegando a caer en términos como: "No sé cómo funciona, pero parece mágico debido a todo lo que hace". La fabricación de chips puede parecer fácil, ya que hoy en día la producción de estos se ha convertido en un proceso de fabricación en masa. Sin embargo, el diseño de la arquitectura y lógica de los chips ha ido evolucionando con el paso del tiempo, ejemplo de esto es uno de los primeros procesadores Intel, los cuales realizaban operaciones de 4 bits en la década de los 70's, ahora el procesador comercial más potente de Intel, el core i9, realiza operaciones de 64 bits a velocidades antes inconcebibles. Diversas cantidades de chips se han desarrollado desde entonces, circuitos integrados, micro-procesadores, procesadores, etc. Para ayudarnos y simplificar las labores humanas cotidianas.

Cada sistema electrónico digital que hemos usado posee un chip, la computadora en la que se escribe el presente trabajo funciona gracias a los miles de millones de cálculos que el procesador realiza, este procesa toda la información en la computadora, controla y sabe que teclas están siendo presionadas o si el mouse se está moviendo. Lleva la cuenta numérica de las operaciones realizadas, hace los cálculos de una hoja de datos de Excel, controla la música que suena en el reproductor, etc. En pocas palabras, el procesador, este circuito integrado o chip controla todo el sistema de la computadora.

Todas esas tareas mencionadas y más realiza el procesador el cual es el ejemplo más claro de lo que es y las tareas que realiza un circuito integrado.

Un circuito integrado (IC) o chip es un dispositivo de pequeñas dimensiones fabricado principalmente de material semiconductor, el cual normalmente es silicio. El área abarcada por un circuito integrado es de pocos milímetros cuadrados. El propósito de estos es la minimización tanto de costos económicos como el tamaño de los

dispositivos electrónicos en diseño.

Un IC es un elemento, el cuál físicamente puede modelarse como una caja negra; ya qué, tiene una cantidad variante de pines de entrada (según sea su aplicación) y así mismo una cantidad variante de pines de salida. Internamente se encuentran diferentes dispositivos, significativamente pequeños, los cuales en su mayoría son transistores interconectados de una manera en específica para cumplir una tarea.

El diseño de estos IC se elabora por medio de distintos *"software"* especializados para cumplir con cada paso del proceso (*design flow*). Estos *software* van desde la descripción de *hardware* a utilizar, es decir el diseño y prueba de la arquitectura, hasta la obtención de los archivos necesarios para la fabricación de las máscaras del chip diseñado. Dichas máscaras son análogos a los moldes para la fabricación y maquinado de una pieza mecánica, por ejemplo: la fabricación de un tornillo específico.

Uno de los entornos más utilizados en la actualidad por la industria de fabricación de semiconductores, es el entorno de *Synopys*. Este provee las herramientas especializadas (EDA) previamente mencionadas, no solo para el diseño sino también la fabricación y testeo del chip. Estas herramientas pueden, hasta cierto punto, automatizar todo el proceso de diseño del circuito integrado.

El proceso de diseño puede tomar diferentes caminos según sean los requerimientos y decisiones que tomen los responsables del área. En el entorno de *Synopys* el proceso de diseño puede tomar 2 caminos: *Custom*, el cuál es iniciar el diseño en silicio totalmente desde cero; o el Automatizado, el cuál toma la descripción de la arquitectura en *Verilog* y mediante distintas herramientas de compilación y síntesis se obtiene el diseño en silicio mapeado según la arquitectura del circuito.

Si bien este último es un proceso automático, se requiere un trabajo significativo en la investigación del funcionamiento de las herramientas para lograr el objetivo final. El trabajo planteado busca elaborar el flujo para el diseño de un circuito nanométrico y obtener los archivos (GDSII) necesarios para su fabricación física, documentar el proceso que va desde la selección del fabricante de semiconductores, intercambio de información con este, obtención y manejo de librerías o modelos de componentes necesarios para la fabricación del circuito integrado hasta el proceso de instalación y utilización de herramientas. Esta documentación busca que el flujo de diseño y fabricación de sea reproducible por futuros estudiantes que necesiten fabricar un chip en silicio y que este cumpla una tarea específica.

En el siguiente trabajo se presentan los distintos métodos de fabricación de los circuitos integrados, así como las empresas que se dedican a la fabricación de estos dispositivos y los procesos de diseño litográfico que estas manejan en la actualidad. Así mismo, se enumeran las reglas de diseño para el proceso litográfico seleccionado, las herramientas necesarias para cumplir el flujo de diseño establecido en función de su aplicación y la presentación del diseño de la arquitectura realizada para el circuito a fabricar.

CAPÍTULO 2

Antecedentes

En la actualidad la mayor parte de los sectores económicos se ven beneficiados por el desarrollo de la tecnología, desde laboratorios de ingeniería aeroespacial hasta el sector agrícola han sido afectados, para bien, por este desarrollo. De manera directa, estos avances tecnológicos son afectados por la constante creación de dispositivos electrónicos, los cuales a su vez aumentan su rendimiento con la mejora de sus 'computadoras internas', es decir los circuitos integrados que poseen dentro.

El desarrollo de estos circuitos integrados son realizados por grupos y empresas dedicadas a la constantes mejoras de las capacidades de estos. Sin embargo, el proceso de diseño y fabricación de estos, a grandes rasgos, es general y replicable para las organizaciones que posean las herramientas adecuadas.

Es por eso que Universidades alrededor del mundo se dedican a la investigación de la micro y nanoelectrónica; diseñan y desarrollan sus propios circuitos integrados y a través del seguimiento del flujo adecuado, generan los archivos necesarios para fabricar sus diseños a través de distintas empresas. Estos casos son más frecuentes en universidades especializadas en el área. Sin embargo, en el caso de universidades latinas existen casos en los cuales los proyectos de diseño e investigación se han enfocado en replicar el flujo de diseño para la fabricación de circuitos integrados, esto con el fin de causar cierto impacto en un área poco conocida en estos países.

El trabajo realizado en [1] es muestra de lo anterior, pues se realiza, partiendo desde el diseño lógico y funcional, un controlador USB 2.0 especificando cada paso tanto del diseño lógico hasta la obtención de los archivos GDSII los cuales son necesarios para su implementación física. Este trabajo realizado en Chile, tiene como objetivo principal exponer el flujo de diseño de un circuito integrado digital y en consecuencia motivar a explorar este campo en el país, partiendo desde el entorno más cercano el cual es el departamento de la universidad.

En [1] se explora la ejecución de todo el flujo de diseño, el cual a través del entorno

de herramientas especializadas proporcionada por *Synopsys*, se obtiene como resultado final los archivos necesarios para la fabricación de un circuito integrado funcional. Aunque la aplicación de este diseño en la actualidad sea considerado como trivial, pues es un controlador USB 2.0, es de suma importancia el aporte de este, ya que, muestra detalladamente la aplicación del flujo mediante un entorno de herramientas comercialmente usado. En [1] se concluye que: 'El diseño de circuitos integrados, sin considerar la fabricación, es totalmente posible de realizar en el país, considerando la existencia de herramientas y personal capacitado'.

El flujo de diseño para la fabricación de un circuito integrado también es aplicado en [2], este trabajo muestra no solo una variante del flujo, pues acá se muestra un flujo de diseño digital, sino también es el resultado de una investigación para volver más seguros a los circuitos integrados, por ejemplo, chips en tarjetas de crédito. El autor explica que los circuitos integrados son vulnerables a los llamados *"side-channel attacks"* en los cuales los atacantes pueden ganar información monitoreando el consumo de potencia, tiempos de ejecución, radiación electromagnética, etc; causado por el comportamiento del switcheo de las compuertas *CMOS.* Este trabajo presenta un flujo de diseño digital para crear circuitos integrados resistentes al análisis de potencia.

La investigación realizada en [2] muestra como a través de una variación al flujo de diseño original se puede crear ciertos chips con capacidades diferentes, los cuales abren una nueva posibilidad de mercado para estos dispositivos. Este trabajo es importante pues ejemplifica a detalle el flujo de diseño utilizado y muestra, a diferencia de [1], pruebas de caracterización al circuito integrado fabricado.

En [3] se aplica el flujo de diseño tradicional para la obtención de los archivos necesarios para fabricación de un microprocesador *RISC* de 16 bits utilizando una herramienta no comercial, el cuál es el caso de *Alliance CAD System.* Al igual que en los trabajos mencionados anteriormente, en [3] se ejemplifican de manera bastante específica el proceso de diseño de este microprocesador. Es importante pues muestra una alternativa para el proceso de diseño, en cuanto a herramientas utilizadas. Las empresas más grandes en desarrollar dichas herramientas son Synopsys y Cadence, sin embargo, no son las únicas alternativas. En este trabajo se trata de comprobar las capacidades de esta herramienta para la obtención de los archivos necesarios para la fabricación de las máscaras a partir de un conjunto de descripciones de hardware dadas en *VHDL.* [3] concluye la posible implementación de circuitos *CMOS VLSI* con *Alliance* así como la caracterización del microprocesador diseñado de dimensiones físicas, cantidad de transistores y frecuencia de operación.

Un ejemplo más de la ejecución del flujo de diseño de un circuito integrado es el de [4]. Este trabajo es realizado en una universidad de Valparaíso, Chile, en colaboración con *Synopsys* y *MOSIS.* En este trabajo se diseñó un circuito integrado de aplicación específica *(ASIC).* En este se especifican los pasos realizados para llegar al producto final, el circuito integrado, totalmente funcional. Se hace énfasis en la colaboración con *MOSIS,* pues a través de sus programas universitarios fue posible la fabricación del circuito integrado el cual consistió en un timer.

Es importante tomar en cuenta los casos mencionados con anterioridad pues, en su mayoría representan trabajos en universidades sin abundante experiencia en el área de fabricación de circuitos integrados.

CAPÍTULO 3

Justificación

Los circuitos integrados se encuentran en la mayoría de los dispositivos electrónicos utilizados hoy en día: teléfonos inteligentes, computadoras, electrodomésticos, etcétera; son prueba de la importancia y el crecimiento que ha tenido esta industria. La labor de investigación y experimentación previa ha permitido que varios dispositivos que antes tenían dimensiones físicas y consumo energético elevados ahora sean significativamente más pequeños, que sean energéticamente más eficientes y computacionalmente más potentes.

Por ende, este es uno de los campos más explotados y desarrollados de la ingeniería electrónica. Sin embargo, es un campo con poco conocimiento en el país. Por lo cual, este trabajo es de importancia debido al impacto que puede generar para que más instituciones nacionales puedan iniciar incursiones en este campo y generar otras alternativas y soluciones para los problemas que acontecen hoy en día.

Así también, este trabajo proveerá de material didáctico para que futuros estudiantes pueden hacer uso de él y así puedan enfocarse totalmente en el diseño del circuito integrado, eficiencia energética y rendimiento, más no en todo el proceso que se requiera para su fabricación.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar la arquitectura del circuito integrado y documentar tanto el proceso de diseño como el proceso de comunicación y validar los parámetros y aspectos técnicos con el fabricante de semiconductores

4.2. Objetivos específicos

- Investigación de las distintas empresas que se dedican a manufacturar circuitos integrados..
- Conocer los distintos procesos litográficos de fabricación que manejan las empresas que manufacturan circuitos integrados.
- Seleccionar el proceso de fabricación que mejor se adapte a los recursos que se poseen.
- Investigar las reglas de diseño del proceso de fabricación seleccionado.
- Investigar las herramientas necesarias para iniciar el flujo de diseño que permita la elaboración del circuito integrado.
- Diseñar la arquitectura del circuito integrado a fabricar.
- Investigar y aplicar las reglas de diseño del proceso de fabricación seleccionado.
- Realizar las pruebas correspondientes a lo largo del desarrollo del chip para asegurar el funcionamiento correcto de este.

CAPÍTULO 5

Marco teórico

Debido a que la mira central de este trabajo estará puesta en la ejecución y desarrollo de un flujo de diseño para la fabricación de un circuito integrado con tecnología nanométrica será necesario plantear algunos conceptos y parámetros que funcionen como ejes sobre los cuales se pueda apoyar la investigación. Además es importe recalcar, que a lo largo del flujo de diseño fue necesario la aplicación de varios conceptos de diseño de circuitos digitales, los mismos serán presentados más adelante en esta sección.

5.1. ¿Qué son los chips?

Para empezar, ¿qué es un chip? Un chip o circuito integrado, IC por sus siglas en inglés, es un elemento en el que se agrupan diversos componentes que forman algún circuito en específico, como su nombre lo indica integra distintos elementos para realizar alguna tarea específica. En otras palabras, los "Chips" o "Circuitos Integrados (IC)", "son redes eléctricas formadas sobre un substrato o dentro del él" [5].

En los chips, cada parte en él cumple con una función importante dentro del dispositivo, por ejemplo el substrato es usado como material semiconductor o aislante mientras que los otros elementos como: diodos, transistores, resistencias y condensadores o capacitores cumplen con papeles específicos que permiten al chip funcionar de una manera óptima. Podemos definir un circuito integrado o chip como una combinación de elementos de circuito interconectados asociados inseparablemente [5].

En la actualidad los chips utilizados en la práctica están fabricados casi en su totalidad con materiales semiconductores, los cuales dependiendo a las propiedades que a las que son sometidos pueden o no conducir una corriente eléctrica. El silicio en su forma cristalina tiene las propiedades semiconductoras necesarias para ser el

material a preferido y más común para la fabricación de los chips.

Desde la invención del transistor sustituyendo a los tubos al vacío, los ingenieros en electrónica en los años 50's vieron la posibilidad más grande para minimizar tanto costos como tamaño de los dispositivos que diseñaban. Jack Kilby fue uno de los pioneros en el diseño de chips. Como menciona [6]: "Su idea fundó una nueva industria y es el elemento clave que yace atrás de nuestra sociedad computarizada".

Desde entonces, el diseño del chip creado por Jack Kilby ha sufrido dramáticos cambios desde el tamaño hasta la cantidad de componentes que los chips poseen. Hoy en día los chips más avanzados contienen cientos de millones de elementos en un área muy pequeña. Esta drástica disminución de tamaño y maximización de la eficiencia y la cantidad de componentes ha sido posible gracias a las nuevas técnicas de fabricación de circuitos que hoy se manejan.

5.2. Método de fabricación

Uno de los fabricantes de semiconductores más grande en el mundo es la empresa Intel, pues ellos fabrican sus propios procesadores para computadora. En [7] se explica el proceso de fabricación: Todo comienza en los grandes depósitos donde se obtienen toneladas de arena muy especial. Aproximadamente el 25 % de la masa de esta arena especial es silicio, el cual es el segundo elemento químico más frecuente en la corteza terrestre. Esta arena especial es principalmente cuarzo, la cual contiene altos porcentajes de dióxido de silicio (SiO_2).

Luego de la extracción el silicio se purifica en múltiples etapas hasta alcanzar la calidad conocida como "Silicio de calidad electrónica". La impureza permitida para esta calidad de silicio es un átomo extraño por cada mil millones, es decir, el $10-9\,\%$ de impureza permitido. Luego de estar purificado el silicio se procede a derretir el mismo hasta conseguir un mono-cristal resultante conocido como lingote. El lingote de "silicio de calidad electrónica" pesa acerca de 100Kg y tiene una pureza de 99.9999 %

El lingote se envía al área de corte, estos son en forma de discos sólidos de silicio individuales. Los discos son conocidos como obleas o *wafers* en la industria. Luego del corte, las obleas son pulidas hasta eliminar todos sus defectos. La compañía Intel compra estas obleas ya listas para la fabricación a otras empresas. El proceso altamente avanzado de High-K/Metal gate de 45nm de Intel usa obleas o wafers con un diámetro de 300 milímetros. Con el paso del tiempo estás obleas han empezado a variar su tamaño, pasando de obleas de 50mm a 300mm lo cual disminuye el costo por fabricación de chip.

El proceso de impresión del chip en la oblea es llamado fotolitografía. Este proceso inicia con la aplicación de una capa foto-resistente (Figura 1), se derrama un líquido sobre la oblea mientras esta gira. Esta aplicación debe ser de manera muy sutil y homogénea. Después de la aplicación de la capa foto-resistente se procede a la exposición, esta se hace a través de la irradiación de una luz ultravioleta (UV). Este

proceso activa una reacción química similar a la que ocurre a la película de una cámara cuando se presiona el botón obturador. El acabado de la capa foto-resistente aplicado anteriormente se vuelve soluble. La exposición se realiza usando foto-máscaras, las cuales actúan como plantillas en este proceso. En las "plantillas" creadas, se imprimen varios patrones de circuito en cada capa del microprocesador o chip, mientras que un lente reduce la imagen de la foto-máscara. El lente reduce aproximadamente cuatro veces el patrón que hay en la foto-máscara. En la oblea mencionada con anterioridad se construyen cientos de chips en un único disco. En los chips actuales hay miles de millones de transistores en un solo circuito integrado; los investigadores, desarrolladores y diseñadores de Intel desarrollaron transistores muy pequeños, tanto que sería posible poner aproximadamente 30 millones de estos en la cabeza de un alfiler. Después de la exposición, el proceso continua lavando la capa foto-resistente por medio de un disolvente lo cual revela un patrón de la capa foto-resistente trazado por la foto-máscara. El material revelado es atacado con productos químicos para eliminar la capa foto-resistente y así, finalmente tener visible la forma deseada.

Continúa el proceso con la implantación de iones. Se inicia con la aplicación de una nueva foto-resistente la cual protege el material que no debe recibir la implantación de iones. La implantación de iones no es más que una forma de realizar el tan conocido dopaje de material las áreas expuestas de la oblea de silicio se les aplica varias impurezas químicas llamadas iones. Estos se implantan en la oblea de silicio para alterar el modo como el silicio conduce la electricidad en esas áreas. Los iones o impurezas se disparan sobre la superficie no cubierta de la oblea a una velocidad muy grande. Un campo eléctrico acelera esas partículas a una velocidad de más de 300,000 km/h.

Después de dopar el material, el proceso de deposición de metal continúa. Para este punto el transistor está casi listo. Se realizan unas perforaciones que se rellenan con cobre, estas hacen las conexiones con otros transistores. La oblea se coloca en una solución de sulfato de cobre. Estos iones se depositan en el transistor por un proceso llamando galvanización. Después de este proceso, la superficie de la oblea está cubierta por una delgada capa de cobre. Se pule el exceso de material y se crean múltiples capas de metal para la interconexión entre varios transistores. Los equi-

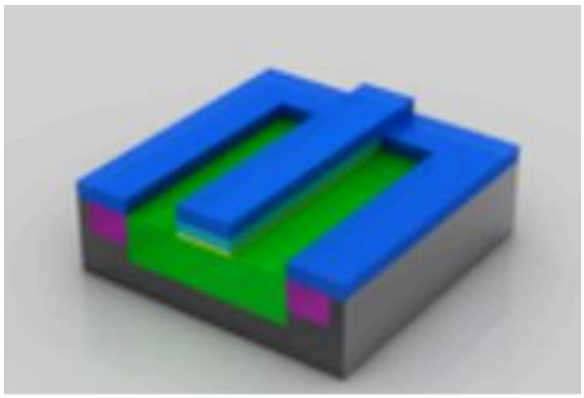


Figura 1: Aplicación de capa foto-resistente

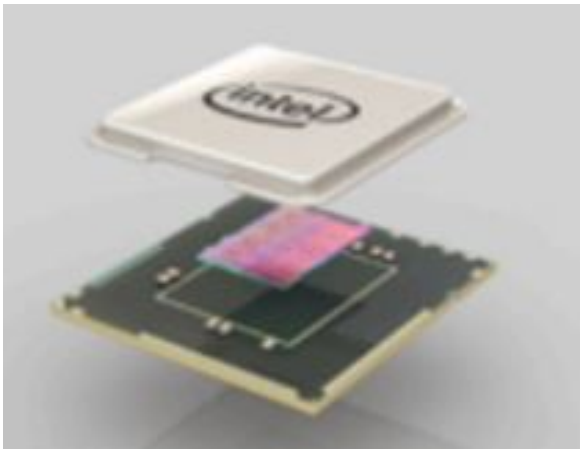


Figura 2: Empaquetado del procesador

pos de arquitectura que desarrollan la funcionalidad de los respectivos procesadores determinan cómo esas conexiones necesitan cablearse. Aunque los chips de las computadoras parezcan extremadamente planos, pueden tener, más de 20 capas para formar un circuito complejo. Como forma de ejemplificación, la imagen ampliada de un chip se ve como una red o sistema de carreteras de múltiples capas.

Luego del cableado se realizan pruebas por cada fracción de la oblea. En esta etapa los patrones de prueba se colocan en cada chip individualmente; luego se monitoriza la respuesta del chip y se compara con la "respuesta correcta". Después de que las pruebas hayan sido satisfactorias cada oblea se vuelve a cortar en nuevos pedazos, a estos se les llama pastillas; cada pastilla es vuelta a examinar esto con la intención de descartar las que están con fallos. Las pastillas que hayan pasado la segunda revisión pasarán a la próxima etapa, el empaquetado.

La pastilla individual que se cortó en la etapa anterior es empaquetada. El sustrato, la pastilla y el disipador de calo se colocan juntos para formar un chip completo. El sustrato o material que se le coloca como empaquetado forma la interfase eléctrica y mecánica para que el chip interactúe con el resto del sistema al que será conectado (Figura 2). Cada chip Intel después del empaquetado es ahora un microprocesador, este tiene un disipador de calor plateado. El último paso antes de que el procesador esté listo es la prueba de clase, durante esta prueba final se prueban las características clave de los procesadores, por ejemplo: la disipación de calor y la frecuencia máxima a la que estos trabajan.

Este es el proceso, que generalmente es usado para la fabricación de todos los chips en la actualidad. Cada fabricante tiene sus peculiaridades, lo cual es la diferencia entre un chip u otro. Sin embargo, este proceso es general para la fabricación de la mayor parte de chips o circuitos integrados que se encuentran en el mercado no importando cual se la aplicación que a este se le vaya a dar.

Las variaciones que ha tenido este método a lo largo de los años han sido principalmente mejoras en cuanto al proceso litográfico y en el diseño de la circuitería dentro

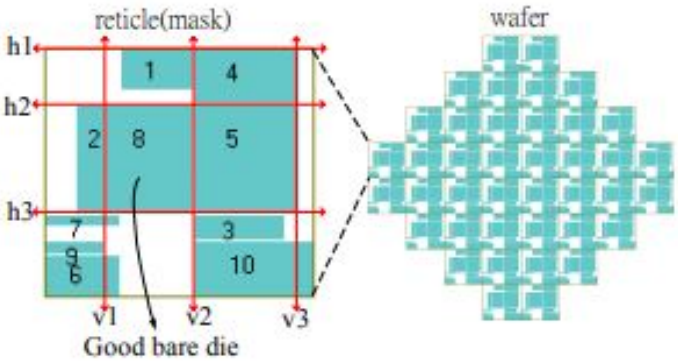


Figura 3: Localización de los diseños en un MPW

de un microcontrolador, la reducción de tamaño ha sido realmente significativa, pues desde los años 70's que se inició con la manufacturación de los circuitos integrados impresos o chips, las reducciones han logrado que los chips puedan maximizar su velocidad y capacidad de procesamiento de datos.

5.3. Mutiple project wafers (MPW)

Según [8], Multi-project wafer (MPW) es comúnmente utilizado para producción de circuitos integrados en pequeñas masas y para la fabricación de chips de programas educacionales o con fines de investigación. Fabricando bajo esta modalidad, los altos costos de las máscaras para fabricación pueden ser compartidos entre los chips.

Las compañías de servicio de diseño y fábricas recolectan el diseño de muchos clientes y estos diseño pueden ser localizados como se observa en la Figura 3. A este proceso se le conoce como *reticle floorplanning*.

5.4. Tecnología de fabricación

La tecnología de fabricación, también conocido como proceso de fabricación o litografía, se refiere a las dimensiones de las principales características físicas de los componentes dentro del circuito integrado. Estas dimensiones varían dependiendo el proceso utilizado. Actualmente estas se encuentran en el orden de los nanómetros. Su función es establecer un estándar de tamaño en los componentes, facilitar el proceso de diseño y garantizar que las características físicas de estos permanezcan acorde a los modelos teóricos y así garantizar un correcto funcionamiento del circuito integrado.

Como se menciona en [9] las librerías de tecnología contienen información acerca de las características y funciones de cada celda provista en la librería de componentes semiconductores del fabricante.

Las características de las celdas incluyen información como nombre de las celdas, nombre de pines, área, *'delays'* y carga en los pines. La librería de tecnología además define las condiciones que se deben cumplir para un diseño funcional, por ejemplo, el tiempo máximo de transición en los nodos. Estas condiciones son llamadas restricciones de reglas de diseño, o simplemente reglas de diseño; estas se explican detalladamente en la sección 5.5.

Las librerías de tecnología se utilizan en varios pasos del flujo de diseño (Sección 5.6), además estas implementan una representación de las compuertas utilizadas en el diseño, es decir, *"definen las características temporales y físicas de las compuertas de cierta tecnología"* [1].

Las dimensiones de los componentes se generalizan en función a un parámetro, usualmente llamado lambda [10] para facilitar la aplicación de las reglas de diseño, estas se explican a continuación.

5.5. Reglas de diseño

El mejor atributo de los procesos de fabricación modernos en que los modelos o diseños son independientes. Esto quiere decir que hay una clara separación entre el proceso de la fabricación en la oblea y el diseño que crea los modelos o características a ser implementadas. Para lograr esta independencia entre el diseñador y el fabricante se requieren una serie de definiciones precisas de las capacidades de la línea de producción. Estas especificaciones usualmente son un conjunto de geometrías permitidas que pueden ser usadas por el diseñador siempre y cuando no violen las características físicas de los dispositivos para un correcto funcionamiento. Cuando son reducidas a su forma más simple, estas restricciones geométricas son llamadas reglas de diseño [10].

Las reglas de diseño son de suma importancia para la fabricación del circuito integrado pues dictan aspectos como:

- Separaciones entre componentes, capas o diseños
- Ancho de las vías y contactos
- Ancho del polisilicio en la compuerta del transistor
- Tamaño de las difusiones
- Separaciones, extensiones y superposiciones de objetos geométricos

La aplicación de estas reglas varía dependiendo la tecnología de fabricación utilizada y el fabricante del circuito integrado. Estas reglas son de utilidad pues evitan fenómenos físicos como capacitancias parásitas, así como restringir el diseño pues los equipos utilizados por los fabricantes poseen limitantes.

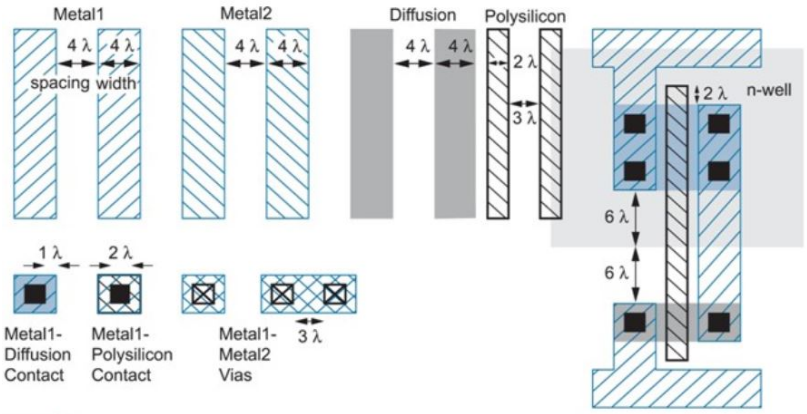


Figura 4: Simplificación de las reglas de diseño basadas en lambda

Uno de los avances más importantes ha sido la evolución de reglas de diseño basadas en lambda. Este es un parámetro adimensional que es igual a resolución fundamental del proceso en sí [10]. Esta es la distancia a la cual una característica geométrica, en cualquier capa, puede estar separada de otra característica geométrica en la misma u otra capa. Todas las dimensiones están dadas en términos de esta unidad de distancia elemental.

En la Figura 4 extraída de [11] se observa las reglas más generales y simplificadas que aplican a las geometrías usualmente diseñadas.

5.6. Flujo de diseño

El flujo de diseño ha sufrido cambios significativos en los últimos 25 años. El escalamiento continuo de las tecnólogas CMOS ha cambiado el objetivo de varios pasos de diseño.

Actualmente se maneja el concepto de de la aproximación de un flujo de diseño totalmente integrado en lugar de una serie de pasos independientes. En la Figura 5 extraída de [12] se observa la evolución de los flujos de diseño a media que el escalamiento de las tecnologías CMOS ha cambiado.

El ritmo en el desarrollo tecnológico guía a un excesivo incremento en el número de transistores en un sistema. Sin embargo, hay una brecha entre lo que es posible diseñar acorde a lo que permite el desarrollo tecnológico y lo que es posible diseñar en términos de costos, estos incluyen los siguientes rubros [13]:

- Tiempo
- Dinero
- Mano de obra

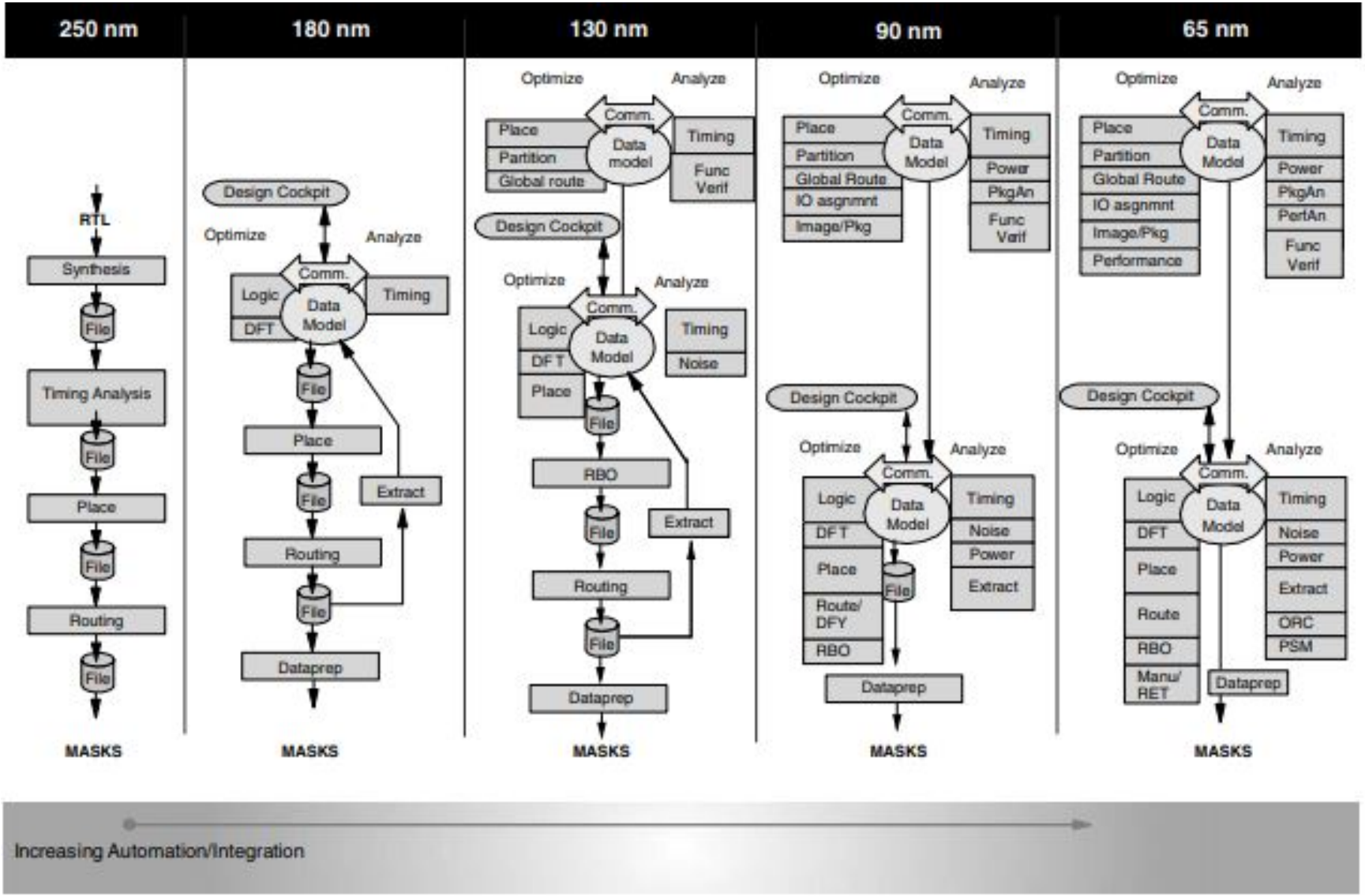


Figura 5: Evolución de flujos de diseño en función de la tecnología

A esta brecha se le conoce como: *"Productivity gap"* o en español Brecha de productividad. Esta es la diferencia entre lo que es posible diseñar utilizando la tecnología disponible y lo que es razonable cumplir en un tiempo realista de diseño. La productividad es definida en [14] como:

$$Productivity = \frac{Total\ Compuertas}{Tiempo\ de\ Desarrollo \times Numero\ de\ Disenadores} \tag{1}$$

En la ecuación 1, se aprecia que si la productividad se mantiene constante, el tiempo de diseño incrementará, pues el número total de compuertas también lo hace. Una alternativa para evitar que el tiempo de diseño crezca es aumentar el número de diseñadores, sin embargo, esto es inviable pues aumentan drásticamente los costos. Por ende, la evolución de la automatización del diseño electrónico (EDA) por sus siglas en inglés, provee de mejoras en la productividad. Estas mejoras se logran pues las evolución de las herramientas permiten un mejor manejo y control de diseños más complejos a los usuarios a través de un modelo inicial de sistema más simple o con un nivel de abstracción más alto.

En el Cuadro 1 extraída de [14], se resumen las cantidades típicas de la productividad de diseño por persona en un tiempo determinado utilizando el paquete de herramientas disponible en la generación EDA.

Conociendo la importancia de la evolución de las herramientas EDA, a partir de esta automatización se ha podido generalizar un flujo de diseño sistemático el cual es aplicado para la construcción de un circuito integrado. Una primera aproximación a un flujo de diseño para la fabricación de circuitos integrados es modelar el sistema inicial con un nivel de abstracción alto.

En [13] se menciona que, con herramientas que permitan realizar una síntesis semi-automática y pasos de optimización es posible transformar el diseño desde las especificaciones funcionales hasta el circuito final (Figura 6). Modelar el sistema con un alto nivel de abstracción significa que se especifican menos detalles de implementación comparado si se modela el sistema a un nivel de abstracción más bajo (Figura 7).

En otras palabras, el flujo de diseño consiste en: a partir de una idea generada por demanda del mercado o razones investigativas, es necesario determinar las especificaciones funcionales del sistema; caracterizar el circuito utilizando un lenguaje de descripción de *hardware*, como Verilog; traducir o sintetizar dicha descripción a una lista de compuertas lógicas físicas para finalmente, utilizando la representación física

Generación EDA	**Compuertas/Diseñadores*Tiempo de desarrollo**
I	500
II	8,000
III	50,000

Cuadro 1: Evolución de la productividad del diseño

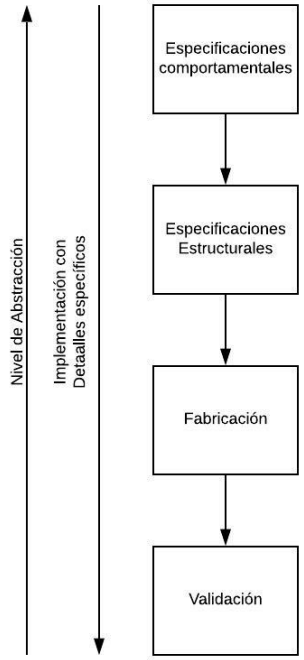


Figura 6: Diagrama del flujo de diseño en alto nivel para la fabricación de un Chip

de cada compuerta y conexiones, producir un plano físico del circuito [1].

En la Figura 7 se encuentra el flujo mencionado anteriormente. Cada uno de estos pasos serán descritos a continuación en las subsecciones 5.6.1 a 5.6.7.

5.6.1. Especificaciones y diseño de la estructura en RTL

El diseño del circuito integrado inicia con la concepción de una idea, la cual es plasmada en un lenguaje gráfico en donde las especificaciones funcionales, de arquitectura y eléctricas son explicitas.

El siguiente paso es la implementación de estas especificaciones. Esto se realiza describiendo el circuito en un lenguaje de descripción de *hardware*, como Verilog. Esta descripción se puede realizar en tres niveles: comportamental, RTL y estructural.

Hoy en día RTL o *Register Transfer Level*, en español nivel de transferencia de registros, es la forma más popular de especificar las funcionalidades del diseño en alto nivel. Esta representación describe al sistema en términos de transformación y transferencia lógica de un registro a otro. Los valores lógicos son almacenados en los registros en donde son evaluados a través de una combinación lógica y posteriormente son guardados, de nuevo, en el siguiente registro [15]. Esta fase funciona como un puente entre *software y hardware*.

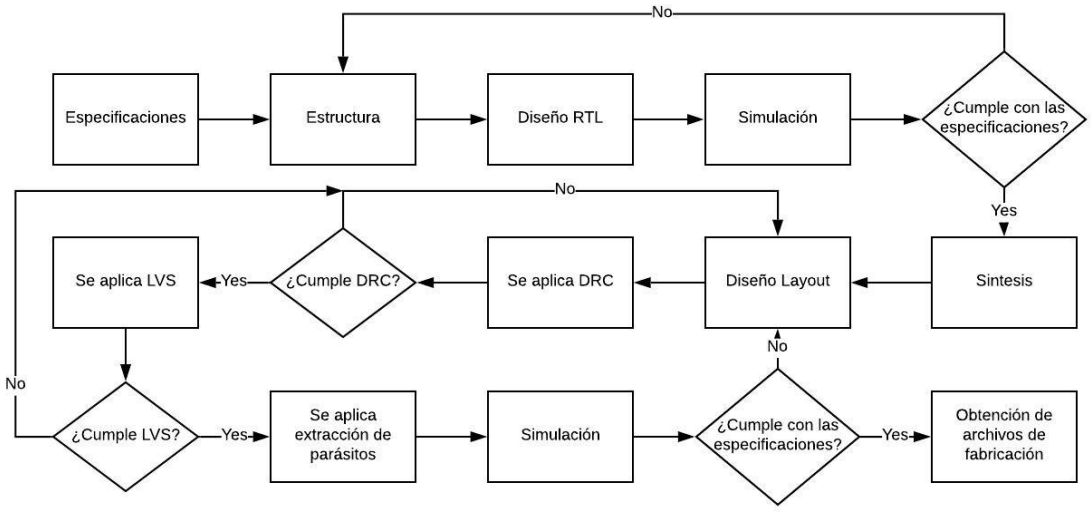


Figura 7: Diagrama del flujo de diseño para la fabricación de un chip

5.6.2. Síntesis

Este paso se realizó por mucho tiempo manualmente. Los diseñadores traducían la descripción de la arquitectura a esquemáticos para luego producir las interconexiones de los componentes a nivel compuertas. Este procedimiento es ahora obsoleto pues la herramientas de síntesis toman la descripción de la arquitectura realizada y automáticamente la transforman en un circuito a nivel de celdas de compuertas [15].

Este proceso es conocido como síntesis lógica y para ser llevada a cabo es necesario las librerías de la tecnología a mapear (provistas por el fabricante), esta es conocida como tecnología o librería objetivo 5.4, la descripción de la arquitectura o diseño RTL y las restricciones de tiempo, área y potencia del circuito [1]. En la Figura 8 se observa las entradas necesarias para generar el *netlist* de compuertas.

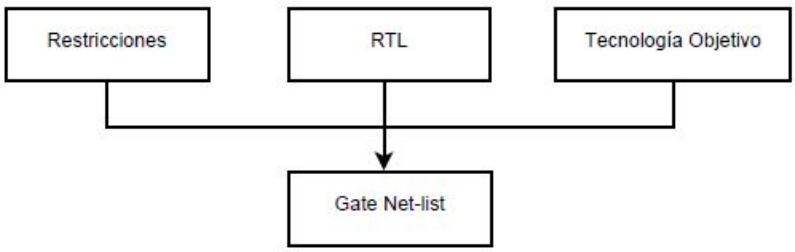


Figura 8: Pasos para síntesis lógica

5.6.3. Simulación del diseño lógico o validación de la arquitectura

El siguiente paso la revisión de la funcionalidad del diseño mediante la simulación del código RTL. Esta simulación se realiza a la arquitectura a nivel Verilog, sin embargo, algunos simuladores son capaces de simular también el diseño mapeado a nivel celdas de compuertas. Es importante tomar en cuenta la cobertura de prueba del diseño es totalmente dependiente del número de pruebas realizadas y la calidad del test bench utilizado.

5.6.4. Diseño físico o Layout

En este paso se realiza el proceso de localización de las celdas y el ruteo para interconectar estas. La primer etapa de este paso se refiere a la localización de las celdas en el plano, también conocido como *foorplan*. La localización de las celdas es un paso crítico pues al ser colocadas eficientemente pueden influir significativamente en los resultados de tiempo y potencia de las pruebas que se realizarán en siguientes pasos. Además, de que facilita el proceso de ruteo [15].

Como se menciona en [1], previo al proceso de ruteo se realiza la inserción del árbol del reloj el cual consiste en el ruteo óptimo del reloj a manera de evitar un desfase de este, también conocido como *clock skew*. Finalmente se realiza el proceso del ruteo entre las celdas.

5.6.5. Revisión de reglas de diseño y comparación del diseño físico contra el esquemático

Las reglas de diseño fueron explicadas en la sección 5.5 y el proceso de revisión de estas es conocido como DRC *(Design Rule Check*. Las condiciones del DRC son basadas en el proceso de tecnología del fabricante y no deben ser violadas. Los atributos del DRC definen las condiciones en las cuales las celdas de la librería operan de una forma segura [15].

El siguiente paso de revisión es la comparación entre el layout final contra el esquemático, el LVS. Esta verificación principalmente revisa que las interconexiones entre las celdas coincidan con las interconexiones entre las compuertas del esquemático.

5.6.6. Caracterización del diseño físico

En esta etapa se realiza la extracción estimada de capacitancias parásitas y delays de las interconexiones (RC delays) del diseño global ruteado. Este paso provee un método más rápido de acercarse a los valores reales de delay del circuito a fabricar.

5.6.7. Validación del diseño físico

Finalmente, posterior a la caracterización del diseño físico, lo cual se refiere a la extracción de capacitancias parásitas, se genera un archivo para simulación el cual contiene la información del circuito integrado. Este archivo es conocido como *Deck*, pues contiene tanto la información de las interconexiones entre las celdas, los delay, capacitancias e inductancias parásitas y desfase del reloj según el diseño físico obtenido. Si la simulación del archivo obtenido es exitosa se procede a la obtención de los archivos necesarios para la fabricación del chip. Estos archivos son los GDS-II.

5.7. Registros

En sistemas electrónicos, los registros son usualmente utilizados para almacenamiento temporal de datos. En computación los registros de este tipo son comúnmente grandes, teniendo al menos 32 entradas [16]. Los registros usualmente son una serie de *Fliop-flops* (ver subsección 5.7.1) conectados en cadena. Un circuito que almacena uno o más bits de información es llamado registro. Frecuentemente un registro es utilizado para hacer más que almacenar datos (subsección 5.7.2).

5.7.1. Flip-Flops

Como se menciona en [16], los circuitos con lógica secuencial son aquellos que siguen un orden específico. Para obtener un sistema secuencial, se debe conocer que ha pasado en el pasado, valga la redundancia. Por lo tanto, es necesario disponer de dispositivos de almacenamiento para retener la información hasta que estemos listos para utilizarla. La unidad básica para este almacenamiento es el *Flip-Flop* o FF. Estos existen en varios tipos:

- FF Set-Reset
- FF tipo D
- FF tipo T
- FF tipo JK

5.7.2. Transferencia de información dentro y fuera de los registros

De los mayores usos que se le da al FF en circuitos digitales secuenciales es para la construcción de registros que almacenan información. La información puede ser transferida dentro o fuera de los registros con n FF, ya sea conectados en serie (1 bit por periodo de reloj) o en paralelo (n bits en un periodo de reloj). Existen 4 maneras posibles de transferir información dentro o fuera de un registro:

- Entrada serial, salida serial
- Entrada paralelo, salida serial
- Entrada serial, salida paralelo
- Entrada paralelo, salida paralelo

5.8. Máquina de estados finitos

Las máquinas de estados finitos son circuitos secuenciales compuestos por varios componentes. Teniendo una aproximación a un nivel superficial, todas las máquinas de estados finitos se componen de los siguientes módulos o estructuras:

- Señal de reloj
- Registros de estado o memoria
- Circuito combinacional lógico para el estado siguiente
- Circuito combinacional lógico para las salidas de la máquina.

Las máquinas de estados finitos o FSM (Finite State Machine), por sus siglas en inglés, según menciona [17] son: *'circuitos secuenciales que tienen entradas, salidas y estados internos. Se acostumbra distinguir entre dos modelos de circuitos secuenciales: el modelo Mealy y el modelo Moore. Difieren en la forma en que se genera la salida. En el modelo Mealy, la salida es función tanto del estado actual como de la entrada. En el modelo Moore, la salida sólo es función del estado actual. Al tratar los dos modelos, algunos libros y otras fuentes técnicas ven el circuito secuencial como una máquina de estados finitos (FSM, finite state machine)'.*

En la Figura 9 se observa los modelos de las máquinas de estados finitos a nivel de bloques.

Acorde a [18]: El modelo matemático de un circuito secuencial es una máquina de estados finitos, la cual es una tupla o lista de 5 elementos:$[\mathbb{X}, \mathbb{Y}, \mathbb{S}, \delta, \lambda]$

Donde:

- $\mathbb{X}$: *Es un número finito de entradas*
- $\mathbb{Y}$: *Es un número finito de salidas*
- $\mathbb{S}$: *Es un número finito de estados*
- δ : *Es la función de transición de estados*
- λ : *Es la función de salidas*

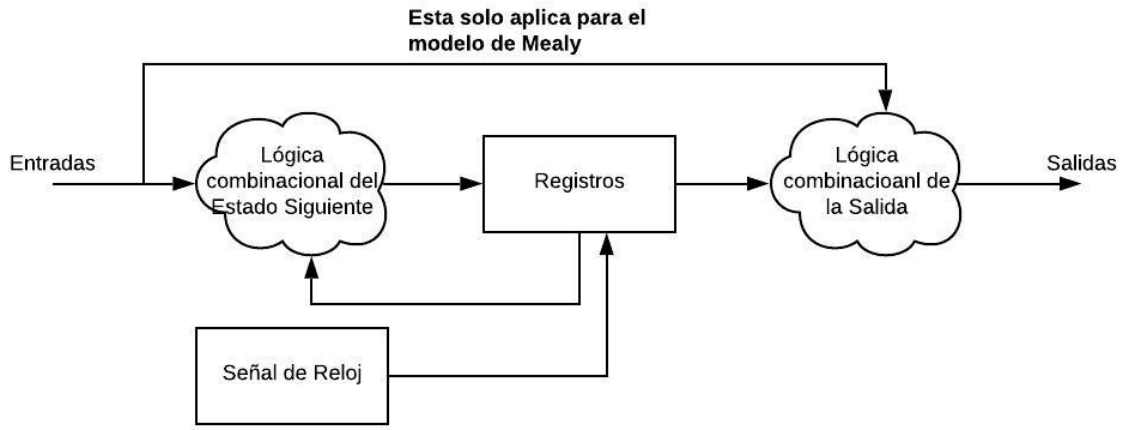


Figura 9: Modelo en alto nivel de la FSM de Mealy y Moore

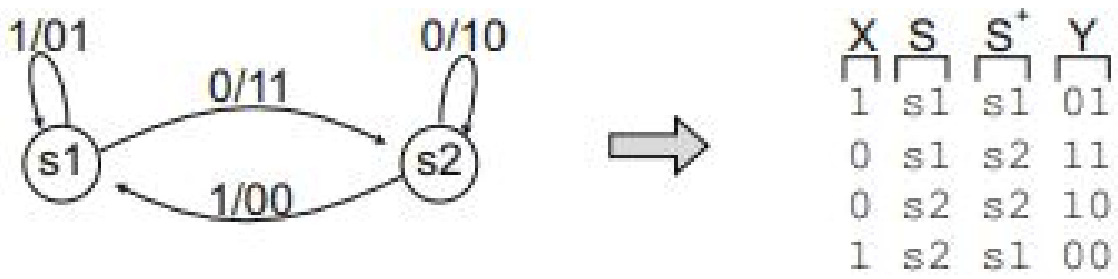


Figura 10: Diagrama de estados y su correspondiente tabla de transiciones de estados

La función de transición de una FSM determina el estado siguiente del autómata, así como la función de salida es asociada con cada transición para el modelo de Mealy o para cada estado según el modelo de Moore. El principal objetivo del estado asignado es asignar a cada estado una representación en binario.

El número de registros necesarios está dado en base el número de estados [19]:

$$Numero\,de\,registros = \frac{\log_{10}(Numero\,de\,estados)}{\log_{10}(2)} \tag{2}$$

Según [18], la máquina de estados finitos puede ser representada por una tabla de transiciones de estados en donde cada fila de la tabla corresponde a la transición entre dos estados de la máquina. En la Figura 10 se observa un diagrama de estados con su respectiva tabla de transiciones.

5.9. Conversor paralelo a USB

En la sección 5.7.2, se explicaron las distintas maneras de transmisión de información. En [20] se menciona que dentro de un sistema computacional, los microprocesadores se comunican en un formato paralelo con sus chips de apoyo como la ROM, RAM, etc. Mientras que la comunicación con los dispositivos externos, como modems e impresoras se hace mediante formatos de comunicación serial. Por ello es que la conversión entre ambos formatos debe ser efectuada.

5.9.1. Protocolos de transmisión serial

Diferentes protocolos de comunicación serial son utilizados para la transferencia de información entre diferentes sistemas digitales. Los protocolos de comunicación serial pueden ser síncronos o asíncronos y estos se diferencian pues en la comunicación síncrona existe una señal de reloj común entre los dispositivos que se están comunicando, mientras que en la asíncrona no existe esa conexión común.

Como se menciona en [21], las interfaces de comunicación serial transfieren un bit a la vez. Los términos utilizados para referirse a las tasas de transmisión son *bit rate* o *baud rate*. Existen varios tipos de protocolos de comunicación serial, los más comunes se enlistan a continuación:

- Universal Asynchronous Receiver/Transmitter (UART)
- Serial Peripheral Interface (SPI) or Synchronous Serial Interface (SSI)
- Inter-Integrated Circuit (I^2C) Bus
- Controller Area Network (CAN) Bus
- Unversal Serial Bus
- Ethernet

Para este trabajo únicamente se entrará en detalle con el protocolo UART, pues fue el utilizado en el prototipo del proyecto.

La interfaz UART es una de las más simples y es uno de los protocolos más utilizados en la comunicación serial. Específicamente, el protocolo UART puede estar basado en los estándares RS232, RS422 y RS485 [21]. Este es un protocolo asíncrono y Full Dúplex, pues utiliza un canal físico de subida y un canal físico de bajada.

La línea de comunicación siempre está en un uno lógico, es decir en *HIGH*, el receptor identifica el inicio de la trama pues el paquete de datos es encabezado por un bit de inicio o *Start Bit*, el cual es un cero lógico. Seguido de este bit se encuentra la trama de información enviada en orden ascendente, es decir después del bit de

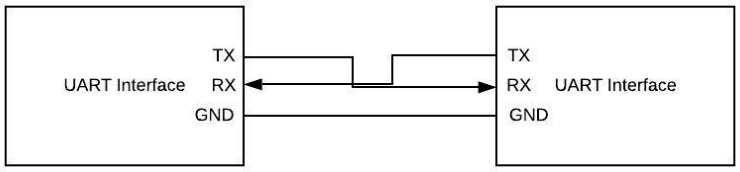


Figura 11: Diagrama de conexión entre interfaces UART

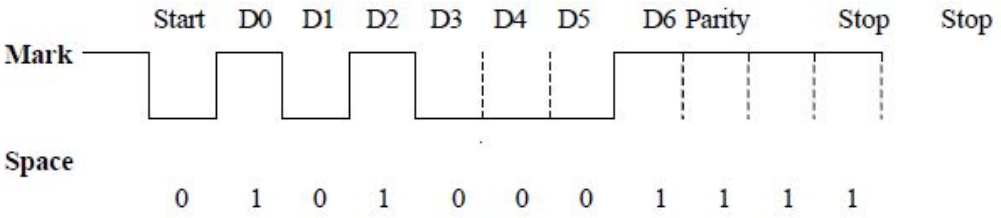


Figura 12: Trama de 7-bits que representa la letra E en codificación ASCII

inicio se encuentra el bit menos significativo de la trama, con valor de 2^0, seguido el que representa 2^1; y así sucesivamente hasta terminar la trama de información. Como método de comprobación de error de transmisión en la trama pueden enviarse bits de paridad. El paquete finaliza con el bit de paro o *Stop bit* que tiene un valor lógico de 1. Un ejemplo de la trama se extrajo de [20] y se muestra ene la figura 12.

5.9.2. Convertidor paralelo a serial

La conversión paralelo a serial, como su nombre lo indica, toma un dato binario presentado en la línea de datos paralelos y los convierte en un patrón de bits serial en una solo línea [16]. Las aplicaciones típicas de este conversor es la transmisión de información desde computadoras a dispositivos periféricos.

Para convertir grupos de entrada paralela a una salida serial, la velocidad de la salida del reloj debe ser N veces más rápida que la velocidad aplicada a las subsecciones de entradas paralelas. Según [16], esta es la diferencia de velocidad mínima que permite que todos los datos sean transmitidos fuera del conversor antes que nuevos datos sean aplicados. La representación en alto nivel de este conversor se observa en la Figura 5.9.2

5.9.3. Convertidor UART a USB

El convertidor UART a USB, permite la comunicación entre dos interfaces seriales, una asíncrona como lo es USB y la otra asíncrona, UART. Este convertidor es un módulo que permite conectar dispositivos TTL (Transistor-Transistor Logic) mediante USB a sistemas que se comuniquen por este protocolo, por ejemplo las computadoras.

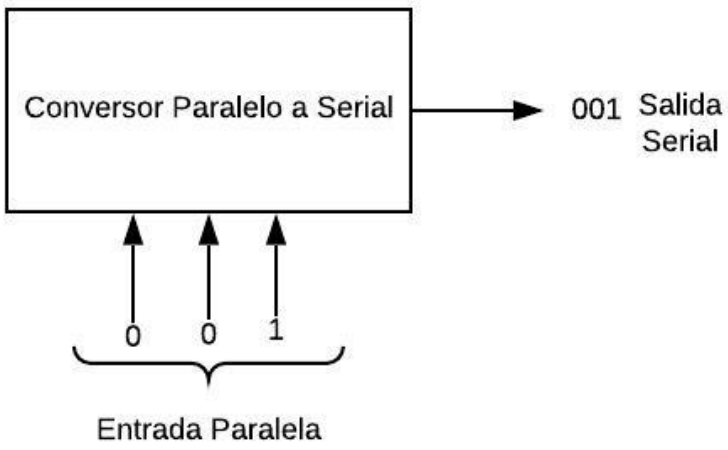


Figura 13: Conversor paralelo a serial

5.10. Python

Python es un lenguaje de programación de alto nivel, libre y gratuito. Según [22], este es un *"lenguaje de programación interpretado, multiparadigma con un tipado dinámico fuerte, dotado de una gestión automática de los recursos, de un alto grado de introspección y de un sistema de gestión de excepciones."*. La sintaxis es simple y explícita, cercana al lenguaje natural.

5.10.1. Programación orientada objetos y derivados

Según [22], se distinguen dos grandes variantes: el paradigma orientado a objetos mediante clases y el paradigma orientado a objetos mediante prototipos. La diferencia principal entre ambos es que la primera, una clase permite tener instancias que pueden, en función de la implementación del lenguaje, tener más o menos afinidad con las clases. Mientras que, en la programación mediante prototipo no existe la noción de instanciación. Los objetos no son más que contenedores de métodos estáticos.

Python permite trabajar con ambos estilos de programación orientada a objetos. Así mismo Python también permite implementar una programación orientada a componentes y también una programación orientada a eventos.

5.10.2. Librerías

En Python un módulo o librería es un archivo que contiene código del lenguaje almacenados con la extensión .py en donde se almacenan la declaración de variables, funciones, objetos, etc [23]. Existen tres tipos de módulos o librerías según [22]: La

primera son las propias, las cuáles consisten en todos los archivos que nosotros creamos de manera modular; la segunda es la librería estándar y la tercera son todas las librerías de terceros. A continuación en las secciones 5.10.2 y 5.10.2 se detallan estos últimos.

Librería estándar

Python provee una librería estándar que permite realizar prácticamente cualquier operación corriente, e incluso más. La documentación de esta librería se encuentra en:

https://docs.python.org/3/library/

Librerías de terceros

Esta librerías son desarrolladas por empresas y desarrolladores independientes. Existe una gran cantidad de este tipo. Python permite empaquetar estas librerías e instalarlas de forma extremadamente sencilla.

A continuación se describen las dos librerías de terceros utilizadas en este trabajo.

5.10.3. Comunicación serial

En la sección 5.9.1 se mencionó los principales protocolos de comunicación serial. La comunicación serial en Python se realiza mediante la librería *pyserial*. Este módulo encapsula el acceso al puerto serial [24].

5.10.4. Text to Speech

Acorde a [25], esta es una librería de Python y una interfaz de línea de comandos (CLI), la cual interactúa con el API del traductor de texto a voz de Google. Escribe los datos de voz en un archivo-objeto mp3 para posterior manipulación de audio.

CAPÍTULO 6

Empresas fabricantes de semiconductores

El proceso de fabricación de un circuito integrado se describió en la sección 5.2, este proceso es ajeno al diseñador pues la empresa que fabrica el semiconductor es la que se encarga de esta parte. Sin embargo, hay varios factores en los que el diseñador puede intervenir; estos van desde selección del fabricante hasta el proceso de fabricación a utilizar.

6.1. Búsqueda de fabricantes

La primera etapa de la metodología consistió en la búsqueda de fabricantes. Esta búsqueda se dividió en 2 categorías:

- Fabricantes a gran escala (producción en masa)
- Empresas intermediarias

La razón de la categorización se expresa en la secciones 6.2 y 6.2.1. La búsqueda se realizó a través del sitio *AnySilicon*, el cual es un motor de búsqueda enfocado en encontrar soluciones en la industria de los semiconductores. Debido a la popularidad de los fabricantes a gran escala se opto únicamente por la búsqueda de empresas intermediarias que ofrecen el servicio de MPW *(Multi Project Wafer)*, este concepto se menciona en la sección 5.3.

6.2. Fabricantes de semiconductores

El campo de la fabricación y diseño de semiconductores es de los más grandes en el mundo, comercialmente hablando. En la actualidad este mercado puede generar movimientos de billones de dólares. Por esta razón las empresas que fabrican los dispositivos físicos se restringen a mantener relaciones comerciales con clientes que generan órdenes sobre el orden los miles, inclusive millones de dólares. Entre estas empresas podemos encontrar:

- Taiwan Semiconductor Manufacturing Company (TSMC)
- On Semiconductor
- Texas Instruments
- Samsung
- AMD
- Intel
- ARM

Debido a las características de estas empresas millonarias, para la fabricación a pequeños volúmenes se opta por seleccionar empresas que ofrecen el servicio llamado *Multi-Project Wafer*, antes mencionado.

6.2.1. Empresas intermediarias

Debido a las características que ofrecen estas empresas, la investigación para la selección de un fabricante se centró en las de este tipo. Las empresas intermediarias en el mercado de los semiconductores son las que prestan su servicios a los potenciales clientes que no están interesados en la producción de circuitos integrados en masa, pues estos lo hacen mayormente con fines investigativos.

Entre las diferentes características que ofrecen estas empresas se encuentran:

- Multi-Project Wafer (costos compartidos)
- Producción a baja escala
- Acceso a documentos de diseño como librerías y modelos de componentes
- Encapsulamiento
- Fabricación de soluciones para IP

La búsqueda de empresas que prestan estos servicios se realizó a través del sitio AnySillicon, mencionado anteriormente. Según [9], una de las primeras cosas a hacer antes de fabricar un chip es seleccionar el fabricante y la tecnología que se va a utilizar. Las características a tomar en cuenta para el proceso de selección son las siguientes:

- Máxima frecuencia de operación
- Restricciones físicas
- Restricciones de potencia
- Restricciones de empaquetado
- Implementación del árbol del reloj
- Floorplanning
- Soporte
- Soporte de diseño para librerías, mega celdas y RAMs
- Núcleos disponibles
- Servicios de testeo y estilo de escaneo disponibles

Tomando en cuenta los criterios de selección enumerados anteriormente y las características ofrecidas por cada intermediario, se seleccionó a MOSIS (Figura 14) como empresa encargada de la fabricación del circuito integrado a diseñar.

6.3. Fabricación con MOSIS

6.3.1. Proceso de fabricación

El acceso que se posee en la universidad a las herramientas de Synopsys para diseño de circuitos integrados incluye las librerías de los componentes con tecnología de 32nm. Las herramientas específicas para realizar el proceso de la validación de reglas de diseño y comparación entre esquemático y *layout* también tienen en cuenta estas librerías y modelos. Por ende, se deseaba fabricar el circuito integrado con una tecnología similar, es decir cercana a los 32nm.

A continuación se enumeran los procesos de fabricación con los que MOSIS cuenta:

Figura 14: Logo MOSIS

- TSMC: 16nm, 28nm, 40/45nm, 65nm, 130nm, 180nm, 250nm
- GlobalFoundries: 12nm FinFET, 22FDX, 45RFSOI, 55nm, 180nm, 90nm, 130nm.
- ON Semiconductor: 500nm
- AIM: Silicon Photonics Full.

Se seleccionó el proceso de 28nm, a través de la empresa TSMC. El proceso de fabricación también llamado 'tecnología' se explica más a fondo en 5.4

6.3.2. Reglas de de diseño

Las reglas de diseño son de suma importancia estas se mencionan a fondo en la sección 5.5

Uno de los aspectos por los que se seleccionó MOSIS fue por el acceso a varios documentos y archivos necesarios para fabricación del chip. A estos archivos se les conoce como *Design Kits*, los cuales son manejados por herramientas específicas según su aplicación. En la Figura 15 se observan las herramientas necesarias para realizar cada paso del flujo de diseño para la fabricación del circuito integrado a través de MOSIS con TSMC utilizando el proceso de fabricación de 28nm.

6.3.3. Eventualidades

Durante el proceso de llenado de documentos legales para trabajar con MOSIS, se encontró nueva información acerca de las cuotas mínimas para fabricación con esta empresa. Pues al poseer una cuenta comercial, términos y condiciones en cuanto a cuotas de fabricación se refiere no son los mismos que con cuentas académicas. La cantidad mínima de fabricación sobrepasó el presupuestos establecido y por ende se decidió buscar nuevas opciones que puedan proveer el servicio.

Se buscó nuevamente a través del motor de búsqueda del sitio *AnySilicon* y se cotejaron 5 nuevas opciones de empresas intermediarias para el proceso de fabricación del circuito integrado. Los criterios utilizados para la selección de este fueron:

Process	Design Kit	Simulation	DRC, LVS, Extraction	Libraries	Other
TSMC028	TSMC iPDK Mentor PDK[1]	Spectre, HSpice, Eldo	Calibre: DRC/LVS/xRC PVS: DRC/LVS Star: RCXT	TSMC, Synopsys	

Figura 15: Design Kit para el proceso de 28nm por TSMC

- Trabajo previo con instituciones en países latinoamericanos
- Fabricantes asociados (a gran escala)
- PDK compatibles con Synopsys
- Procesos de fabricación adecuados para los recursos que se poseen
- Precio de fabricación en función del área y tecnología seleccionados
- Acceso a los archivos necesarios para la fabricación: modelos y librerías para el diseño físico del IC

Estos criterios se sumaron a la lista enumerada anteriormente en la sección 6.2.1. Finalmente, la empresa intermediara seleccionada fue *Europractice* (Figura 16).

6.4. Fabricación con Europractice

Europractice es una empresa lanzada por la comisión europea en 1995 y durante los últimos 25 años ha provisto, tanto en la industria como en instituciones académicas, una plataforma para desarrollar sistemas integrados inteligentes que van desde el diseño avanzado de prototipos hasta la producción en volumen.

Como uno de sus principales objetivos propuestos en 2019 es ampliar el alcance para trabajar no solo con países de la unión europea si no alrededor del mundo. Europractice cuenta con socios los cuales son los encargados de brindar la interfaz necesaria entre clientes y proveedores de tecnología. Entre estos se encuentran:

- Imec
- Science and Technology Facilities Council
- Fraunhofer
- CMP
- Tyndall

Figura 16: Logo Europractice

ams	selected MPW runs
GLOBALFOUNDRIES	selected MPW runs for 130nm, 55nm and 22nm nodes
IHP	all MPW runs
On Semiconductor	all MPW runs, except On Semi 0,18 µm
TSMC	selected MPW runs for 0.18µm, 65nm, 40nm and 28nm CMOS mixed signal RF
UMC	selected MPW runs for 0.18µm, 0.13µm and 65nm CMOS mixed signal RF
X-FAB	selected MPW runs for XH018 and XT018

Figura 17: Empresas y tecnologías disponibles a través de *Mini@sic*

Una de las características principales por la que seleccionó trabajar con Europractice, como se menciona en la sección 6.3.3 fue los fabricantes asociados y los procesos de fabricación o tecnología que se poseen. Una de las ventajas de esta empresa con respecto a las cotejadas con anterioridad reside en una de sus modalidades dentro de su fabricación en *MPW*, esto pues posee una opción de fabricación llamada *Mini@asic* (Mini - Application Specific Integrated Circuits por sus siglas en inglés), la cual se adecua a nuestro requerimientos y necesidades debido a que la carga mínima de diseño con la opción regular de MPW es más grande de lo que necesitamos. Además, el servicio *Mini@sic* proporciona un medio de fabricación con las empresas y tecnologías mostradas en la figura 17

6.4.1. Proceso de fabricación

Con el cambio de fabricante, el presupuesto disponible y el *design kit* compatible con Synopsys se ampliaron los parámetros de selección del proceso de fabricación. Desde un inicio, el enfoque fue fabricar con TSMC, por lo que se revisaron las tecnologías de fabricación y precios disponibles con Europractice, los mismos se muestran en la Figura 18

Con estos nuevos parámetros, finalmente se seleccionó el proceso de fabricación de 180nm de propósito general. Posteriormente se tradujo, para firma del rector de la Universidad, el documento legal correspondiente para obtener el acceso a los archivos de diseño adecuados para continuar con el flujo de diseño del circuito integrado.

6.5. Reglas de de diseño

Las reglas de diseño como se mencionan en la sección 5.5, son de suma importancia pues dictan aspectos tanto físicos como funcionales que debe cumplir el diseño en silicio. Este set de reglas será proporcionado por Europractice al momento de completar las formalidades del acuerdo. Esta información al ser exclusiva no se encuentra disponible en la página, pues es incluida dentro de los archivos del *Design Kit* que se hace entrega cuando el NDA (non-disclosure agreement, por sus siglas en inglés) es entregado firmado por el cliente al fabricante y este es aceptado.

TSMC	STANDARD Price/block	DISCOUNTED Price/block
TSMC 0.18 CMOS Logic or Mixed-Signal/RF, General Purpose	3610 [9]	3140 [9]
TSMC 0.18 CMOS High Voltage BCD Gen 2	5090 [16]	4470 [16]
TSMC 65nm CMOS LP MS RF	12890 [10]	12130 [10]
TSMC 40nm CMOS LP MS RF	17320 [8]	16300 [8]
TSMC 28nm CMOS HPC RF	19960 [15]	17510 [15]
TSMC 28nm CMOS HPC RF – Micro-block	11805 [17]	9600 [17]

Figura 18: Precios (en euros) y tecnologías disponibles con TMSC

CAPÍTULO 7

Descripción del sistema implementado

En esta sección se presenta de manera detallada los requerimientos y características del sistema a implementar. El sistema consistirá en un bloque que dará como salida el valor ASCII de distintos caracteres, para luego a través de una interfaz entre el circuito físico y la computadora, este mensaje pueda ser recibido por esta y finalmente mediante un programa sea posible la articulación del mensaje enviado. El mensaje que se desea leer es un texto conmemorativo que representa el logro de diseñar el primer circuito integrado en el país. En otras palabras, este capítulo corresponde al proceso previo a la descripción RTL del circuito integrado a fabricar y el diseño del sistema que lo implementará.

7.1. Análisis de los requerimientos funcionales

El primer paso antes de iniciar con el diseño tanto del bloque que genera los caracteres ASCII y del sistema que implementará al circuito integrado a fabricar es la descripción de los requerimientos funcionales del sistema integrado. Es decir, es necesario describir el modelo y las especificaciones de diseño.

7.1.1. Organización temporal de los datos

La intención principal es transmitir los caracteres generados en código ASCII hacia el puerto serial del computador para luego poder ser procesado por un programa. Para ello se utilizará el protocolo UART mencionado en la sección 5.9.1. Como se explica en dicha sección cada trama de datos consta de 10 bits, pues que para la representación

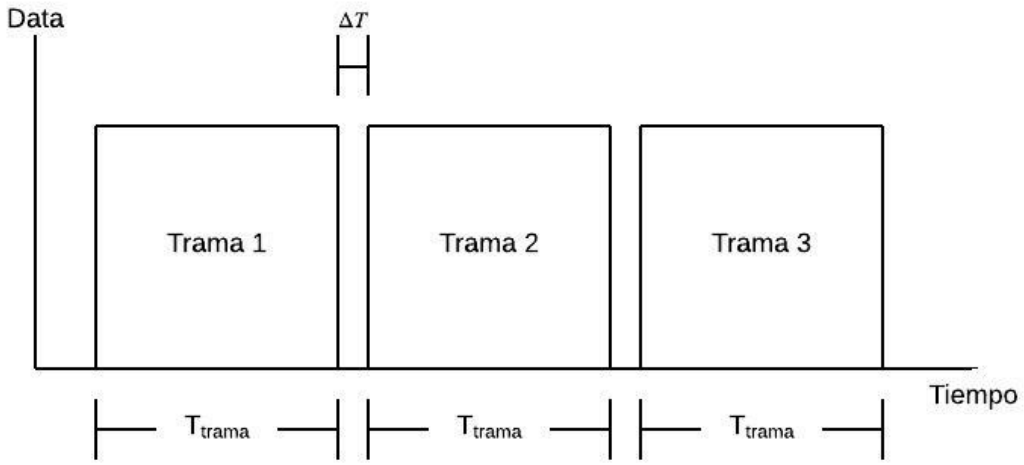


Figura 19: Ejemplo de organización de tramas

de cada carácter en código ASCII se requieren 8 bits además de los 2 bits de control requeridos por el protocolo (Start bit y Stop bit). Por lo tanto, cada trama enviada ocupará un espacio temporal el cual se calcula en base a la ecuación 3. En la Figura 19 se ejemplifica como las diferentes tramas se organizan en el tiempo. El T_{trama} depende directamente de la frecuencia de transmisión de datos, mientras que el ΔT es un intervalo de tiempo entre cada trama de datos.

$$T_{trama} = 10 \cdot Frecuencia\,de\,transmision\,de\,datos \tag{3}$$

7.1.2. Diagrama de bloques del sistema

En la Figura 20 se observa la implementación en alto nivel del sistema. Cada bloque de este diagrama presenta una parte importante del sistema en conjunto. Es importante la descripción de la funcionalidad, características y requerimientos necesarios de cada bloque. Estos bloques serán descritos en las subsecciones posteriores con excepción del bloque que estará representado por la computadora.

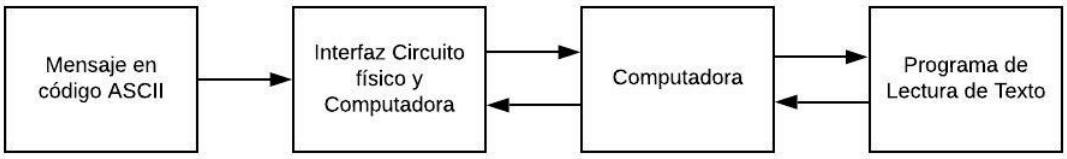


Figura 20: Diagrama de bloques del sistema

Mensaje en código ASCII

Este bloque tiene como función generar el código ASCII de cada carácter para completar el mensaje conmemorativo. Entre los requerimientos funcionales es que este pueda presentar en su salida el valor de cada carácter, debido a la naturaleza y la intención de este trabajo, el bloque representará a una máquina de estados finitos (FSM) (ver sección 5.8) y tendrá 8 salidas para representar el valor de cada carácter en ellas. Este valor cambiará cada vez que una señal externa (reloj) tenga un flanco positivo. Uno de los principales requerimientos es que esta FSM pueda funcionar al menos a una frecuencia de 9.6KHz, la cual es la velocidad mínima de transmisión serial.

Debido a que este bloque está representado por un circuito totalmente digital, se decidió aplicar el flujo de diseño (5.6) a este. Por consiguiente, el circuito integrado a fabricar consistirá de una máquina de estados finitos (FSM).

Interfaz circuito físico y computadora

Este bloque tiene como función establecer el puente entre el bloque que generará los caracteres que formarán el mensaje y la computadora. Debido a que en la subsección 7.1.2, se estableció que este bloque estará representado por una FSM, la interfaz de comunicación debe ser capaz de poder tomar código ASCII de cada carácter y enviarlo a la computadora a través del puerto USB de la misma. Por ende, este bloque debe ser capaz de funcionar al menos a una frecuencia de 9.6KHz además de poder transmitir el código que está en su entrada paralela a su salida serial mediante y luego poder enviar este al puerto USB de la computadora. Por consiguiente, esta interfaz será un circuito conversor Paralelo a Serie, mediante el protocolo UART.

Programa de lectura de texto

El bloque final del sistema debe ser capaz de poder recibir el mensaje a través del puerto USB, interpretarlo, procesarlo y finalmente leerlo. Este bloque estará representado por un programa realizado en Python (ver sección 5.10) pues como ahí se menciona la sintaxis con la que funciona es bastante, simple y natural además de ser en alto nivel pues facilita la codificación del mismo. Otra de las razones importantes para la selección de este *software* es la cantidad de documentación y librerías que posee.

CAPÍTULO 8

Diseño de la arquitectura

8.1. Descripción de la máquina de Estados Finitos

La definición de una máquina de estados finito se mencionó en la sección 5.8. Ahora es importante definir las características y la función de esta. La intención de este trabajo es mostrar la aplicación de del flujo de diseño para un circuito integrado mediante la fabricación de un chip con una función trivial, es decir una función como la de esta FSM. Esta máquina de estados finitos fue diseñada para que la salida de cada estado mostrara el valor ASCII de una serie de caracteres, las cuales al unirse pudieran mostrar un mensaje en conmemoración a este proyecto. El mensaje a desplegar es el siguiente: *"Soy el primer chip con tecnología nanométrica 100 por ciento diseñado por un programa de licenciatura de una escuela de ingeniería en la historia de Centroamérica. Mis creadores son: Carlos Esquit, Diego Soler Castañeda, Steven Hiram Rubio, Luis Arturo Nájera y Pablo Ortiz Barillas. Departamento de ingeniería electrónica. Universidad del Valle de Guatemala, 2019."*

Para la simplificación del diseño, el número de estados de la FSM será igual al total de caracteres que posee el mensaje a reproducir y la salida de cada estado será representada por el valor ASCII en 8bits que representa a cada caracter. Las características generales de la FSM se presentan en el Cuadro 2.

Características de la FSM		
Número de estados	Número de salidas	Numero de entradas (No se incluye el clock)
345	8	0

Cuadro 2: Características de la máquina de estados finitos

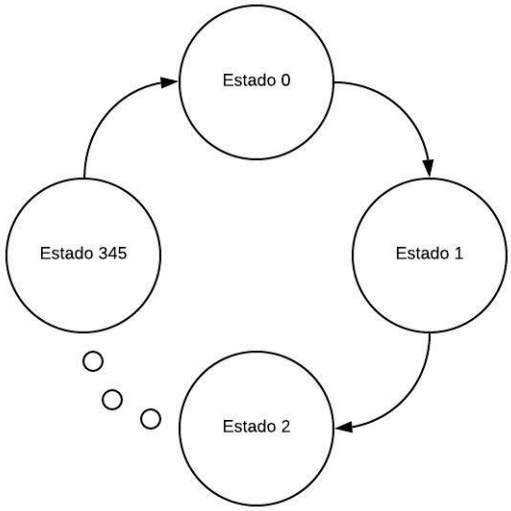


Figura 21: Diagrama de estados de la FSM

Con las características generales de la máquina de estados finitos definidas se procedió con el diseño. Este se explica a fondo en la sección 8.2

8.2. Diseño de la máquina de estados finitos

Luego de definir las características y requisitos funcionales de la FSM, se inició con el proceso de diseño. Como se explica en la sección 5.8, una FSM consta de varias módulos: la lógica combinacional del estado siguiente, el módulo de registros y la lógica combinacional de la salida. Para iniciar el proceso de diseño, se estableció que el funcionamiento de la FSM debía de ser cíclico por lo que la transición del último estado llevaría al primero.

La cantidad de estados fue definida con base en la cantidad de caracteres en el texto, si el conteo de estos fue de 345 entonces la cantidad de estados se definió de igual número. Por lo que las transiciones serían ordenadas y un estado únicamente podría ir otro. Es decir no hay dos transiciones que lleguen a un mismo estado. A modo de ejemplo ver Figura 21.

8.2.1. Selección del modelo de la FSM

La selección del modelo con el que se diseñará la FSM está establecida por la cantidad de entradas, salidas y la mejor representación de la transición de estados. Por ende, se decidió aplicar el modelo de Moore, pues al no tener entradas lógicas la función combinacional de la salida de la FSM únicamente dependería del estado actual de la máquina.

8.2.2. Calculo de la cantidad de registros

Si se conoce la cantidad de estados que posee la FSM se puede calcular la cantidad de registros necesarios para implementarla. Aplicando la ecuación 2 se obtiene que la cantidad de flip-flops necesaria es de 8.

8.2.3. Función de transiciones de estado

Luego de definir como serían las transiciones de estados se procedió a la construcción de las tablas de estos. Las tablas están compuestas por los cuadros descritos a continuación:

- **Cuadro 3:** Contiene la letra o caracter que corresponde a cada estado

Estado actual			Estado siguiente		
Letra	Decimal	Binario	Letra	Decimal	Binario
S	0	000000000	O	1	000000001
O	1	000000001	Y	2	000000010
Y	2	000000010	ESPACIO	3	000000011
ESPACIO	3	000000011	E	4	000000100
E	4	000000100	L	5	000000101
L	5	000000101	ESPACIO	6	000000110
ESPACIO	6	000000110	P	7	000000111
P	7	000000111	R	8	000001000
R	8	000001000	I	9	000001001
I	9	000001001	M	10	000001010
M	10	000001010	E	11	000001011
E	11	000001011	R	12	000001100
R	12	000001100	ESPACIO	13	000001101
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
A	336	101010000	,	337	101010001
,	337	101010001	ESPACIO	338	101010010
ESPACIO	338	101010010	2	339	101010011
2	339	101010011	0	340	101010100
0	340	101010100	1	341	101010101
1	341	101010101	9	342	101010110
9	342	101010110	.	343	101010111
.	343	101010111	ESPACIO	344	101011000
ESPACIO	344	101011000	ESPACIO	345	101011001
ESPACIO	345	101011001	S	0	000000000

Cuadro 3: Fragmento 1 Tabla de estados

- **Cuadro 4:** Contiene el valor en binario del estado actual

Estado actual								
Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	1
0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	0	0	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	0	1	0	0	1	0	0	1
1	0	1	0	0	1	0	1	0
1	0	1	0	0	1	0	1	1
1	0	1	0	0	1	1	0	0
1	0	1	0	0	1	1	0	1
1	0	1	0	0	1	1	1	0
1	0	1	0	0	1	1	1	1
1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0
1	0	1	0	1	0	0	1	1
1	0	1	0	1	0	1	0	0
1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0	1

Cuadro 4: Fragmento 2 Tabla de estados

- **Cuadro 5:** Contiene el valor en binario del estado siguiente

Estado siguiente								
D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	1
0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	1
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	0	1	0	0	1	0	1	0
1	0	1	0	0	1	0	1	1
1	0	1	0	0	1	1	0	0
1	0	1	0	0	1	1	0	1
1	0	1	0	0	1	1	1	0
1	0	1	0	0	1	1	1	1
1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0
1	0	1	0	1	0	0	1	1
1	0	1	0	1	0	1	0	0
1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0	1
0	0	0	0	0	0	0	0	0

Cuadro 5: Fragmento 3 Tabla de estados

De estas se obtuvieron las ecuaciones para la función de transición de estados (Anexo 14.1.1).

8.2.4. Función de salidas

Se construyó la tabla de salidas, la cual indica cual debe ser el valor de la salida en función de cada uno de los estados de la máquina de estados finitos. La tabla se compone de los siguientes cuadros:

- **Cuadro 6:** Indica el valor ASCII de cada uno de los caracteres

Caracteres		
Letra	Decimal	Valor ASCII
S	0	83
O	1	79
Y	2	89
ESPACIO	3	32
E	4	69
L	5	76
ESPACIO	6	32
P	7	80
R	8	82
I	9	73
M	10	77
E	11	69
R	12	82
.	.	.
.	.	.
.	.	.
M	333	77
A	334	65
L	335	76
A	336	65
,	337	44
ESPACIO	338	32
2	339	50
0	340	48
1	341	49
9	342	57
.	343	46
ESPACIO	344	32
ESPACIO	345	32

Cuadro 6: Fragmento 1 Tabla de salidas

- **Cuadro 7:** Indica el valor binario de cada uno de los estados

Salida registros								
Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	1
0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	0	0	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	0	1	0	0	1	0	0	1
1	0	1	0	0	1	0	1	0
1	0	1	0	0	1	0	1	1
1	0	1	0	0	1	1	0	0
1	0	1	0	0	1	1	0	1
1	0	1	0	0	1	1	1	0
1	0	1	0	0	1	1	1	1
1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0
1	0	1	0	1	0	0	1	1
1	0	1	0	1	0	1	0	0
1	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0	1

Cuadro 7: Fragmento 2 Tabla de salidas

- **Cuadro 8:** Indica el valor binario de cada salida en función del estado actual

Salida lógica							
S8	S7	S6	S5	S4	S3	S2	S1
0	1	0	1	0	0	1	1
0	1	0	0	1	1	1	1
0	1	0	1	1	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	0	1	0	1
0	1	0	0	1	1	0	0
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	0	1	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	0	1	1	0	1
0	1	0	0	0	1	0	1
0	1	0	1	0	0	1	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	1
0	1	0	0	1	0	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	0	1	1	0	1
0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0
0	1	0	0	0	0	0	1
0	0	1	0	1	1	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	1	0
0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	1
0	0	1	1	1	0	0	1
0	0	1	0	1	1	1	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0

Cuadro 8: Fragmento 3 Tabla de salidas

Con los cuadros anteriormente descritos, se precedió a la obtención de las ecuaciones de la función de salida mostradas en el anexo 14.1.2. Así mismo, se procedió a la implementación en Verilog 8.3 y se iniciaron las pruebas del diseño.

8.3. Implementación en Verilog

Se implementó la máquina de estados finitos descrita en la sección 8.2. La implementación de la esta se realizó a nivel *behavioural* para realizar las pruebas de funcionalidad y posteriormente la síntesis a nivel celdas.

No fue necesaria la implementación a nivel *structural* de la máquina de estados finitos, pues para realizar el proceso de síntesis la herramienta requiere que el archivo de entrada este descrito con instrucciones en alto nivel.

En la Figura 22 se presenta un fragmento de la implementación de la FSM en el lenguaje descriptivo de Verilog.

```
module chip(output [7:0] q, input reset, input clk);

    wire [8:0] contador;
    reg [7:0] q;

    Cont cont1(contador, reset, clk);

    always @ (posedge clk)

    if(contador == 'd0)
    begin
        q=8'b01010011;
    end
    else if(contador == 'd1)
    begin
        q=8'b01001111;
    end
    else if(contador == 'd2)
    begin
        q=8'b01011001;
    end
    else if(contador == 'd3)
    begin
        q=8'b00100000;
    end
    else if(contador == 'd4)
    begin
        q=8'b01000101;
    end
```

Figura 22: Fragmento de la implementación de la FSM diseñada

8.4. Validación arquitectura

El último paso para completar el diseño de la arquitectura, según el flujo mostrado en la sección 5.6, es la validación de esta. Esta validación se realizó a través de la herramienta VCS guiándose con el documento [26]. La idea de esta verificación es comprobar que con cada pulso de reloj se obtenían la representación de los caracteres del mensaje en la salida de la FSM.

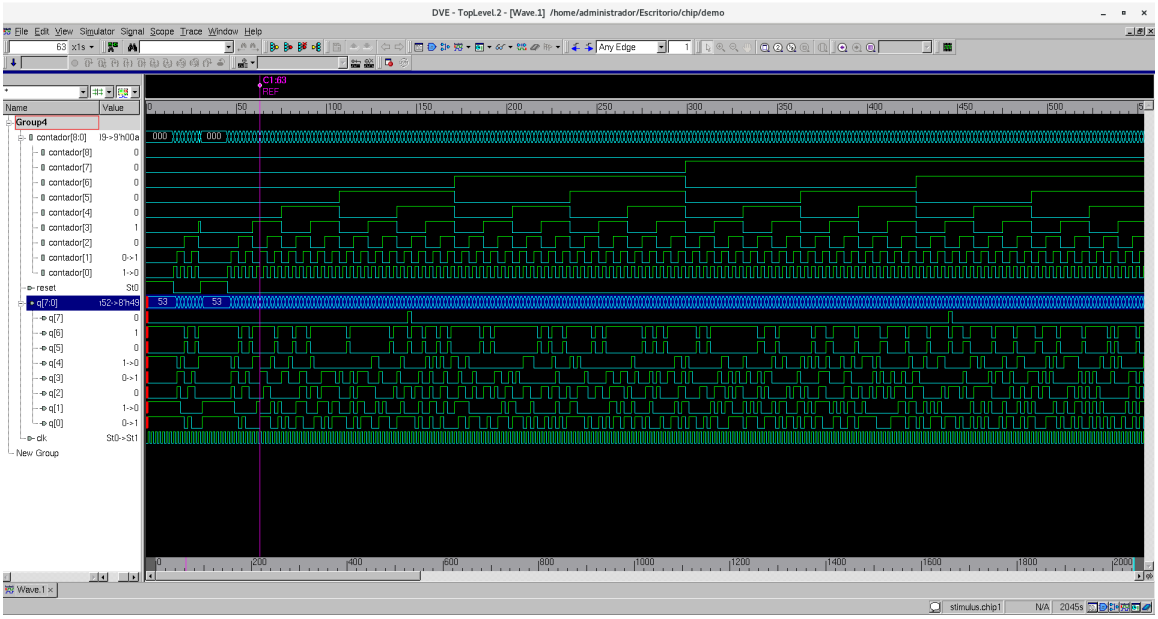


Figura 23: Resultados obtenidos de las pruebas a la arquitectura

Con los resultados obtenidos en la Figura 23 se comprobó el correcto funcionamiento de la arquitectura y por ende se precedió a realizar la síntesis del circuito.

CAPÍTULO 9

Diseño del sistema que implementa el IC - Prototipo 1

El objetivo principal del diseño y la construcción de este prototipo era conocer y estudiar el funcionamiento de la librería de Python gTTS, explicada en la sección 5.10.4. Por ello únicamente se utilizó como dispositivo físico un Arduino. El diagrama de bloques se muestra en la sección 9.1.

9.1. Topología en alto Nivel

En la Figura 24 se presenta el diagrama de bloques correspondiente con el prototipo 1 descrito anteriormente. Se observa que este consta de tres módulos los cuales serán descritos posteriormente. Como se mencionó anteriormente, la intención de este prototipo consistió tanto en la prueba del método de lectura de texto utilizando la librería gTTS, como en la prueba del algoritmo utilizado para la concatenación de caracteres provenientes del puerto serial de la computadora.

Figura 24: Diagrama de bloques del Prototipo 1

9.1.1. Arduino

La función de este bloque correspondió a la simulación tanto de la FSM que generará el código ASCII de cada carácter del mensaje como también la función de ser la interfaz que comunica este mensaje hacia la computadora. Para el funcionamiento del programa (Figura 25) se utilizó la librería serial del Arduino y se envió a través de este los caracteres que conformarán el mensaje deseado.

9.1.2. Programa de lectura de texto

La función de este programa es la lectura de los caracteres recibidos a través del puerto serial. Por ende, la función de este bloque se puede dividir en una secuencia de pasos: El primer paso consiste en inicializar la configuración para establecer una comunicación serial; posteriormente se procede a leer el carácter localizado en el buffer del puerto serial, únicamente se lee un byte; luego de leer el carácter se procede a su almacenamiento en un arreglo para luego concatenar ese último carácter en una variable tipo String; finalmente cuando se envía el carácter con el valor ASCII decimal de 10, se cierra el puerto serial y se procede a enviar como parámetro la variable String concatenada al método de lectura de la librería gTTS, esta devuelve un objeto tipo *.mp3* que puede ser almacenado para su posterior reproducción. En la Figura 26 se observa el código utilizado para realizar la descripción anterior.

Clase serial

Para inicializar la comunicación serial a través de Python es necesario crear un objeto de tipo serial, esto se realiza instanciando la clase serial.Serial que se encuentra en la librería serial de Python (ver sección 5.10.3). Para instanciar esta clase es necesario enviarle los parámetros necesarios y así la inicialización del puerto serial pueda llevarse acabo, estos se describen a continuación:

```
char *mensaje = {"Soy el primer chip con tecnologia nanometrica 100 por ciento disenado por

void setup() {
  Serial.begin(9600);
}

void loop() {
  /*for (int i = 0; i < 6; i++) {
    Serial.println(myStrings[i]);
    delay(500);
    */

    Serial.println(mensaje);
    delay(500);
}
```

Figura 25: Código del Arduino utilizado para el prototipo 1

- **Port:** Este parámetro debe contener la información del puerto serial que se estará abriendo.
- **Baudrate:** Este parámetro debe contener el valor de la velocidad de transmisión de datos que se efectuará. Para este caso el valor a utilizar será 9600.
- **Parity:** A través de este parámetro se envía la información sobre la paridad de bits en la trama que se transmite, es decir, si esta contendrá bits de paridad para corroborar la transmisión correcta. Para efectos de este prototipo no se utilizarán bits de paridad.
- **Stop bits:** En este parámetro se establecen la cantidad de bits de paro que contendrá la trama de información.
- **Byte Size:** Este parámetro debe contener la cantidad de bits de datos que contendrá la trama transmitida.

De igual forma fue necesario el uso de dos métodos de la clase serial. Estos fueron: El método *read()*, el cual lee la cantidad de bytes especificada (si no se hace el valor por defecto es un byte) del puerto serial; y el método *close()*, el cual es utilizado para cerrar el puerto inmediatamente.

Clase gTTS

Esta clase únicamente posee un método el cual es llamado también gTTS (*Google Text-to-Speech*). El objeto devuelto por el método puede variar según los parámetros enviados, para la aplicación del prototipo uno se utilizaron tres parámetros los cuales fueron:

- **Text:** Este parámetro debe contener el texto que se leerá. El tipo de dato aceptado por el método es String.
- **Lang:** Este parámetro debe contener el lenguaje de lectura del texto. Para efectos de este proyecto se seleccionó como lenguaje el español.
- **Slow:** En este parámetro se debe especificar la velocidad de lectura del mensaje enviado. El método acepta datos tipo *booleans*, por lo que si el valor del parámetro es *False* el mensaje será leído a una velocidad natural, mientras que si es *True* el mensaje será leído a una velocidad lenta.

9.2. Pruebas y resultados

Al implementar el sistema explicado en la sección 9.1 se obtuvieron los resultados esperados, pues al utilizar el Arduino como circuito físico se simplificó el problema

```python
# Import the required module for text
# to speech conversion
from gtts import gTTS
import serial

ser = serial.Serial(
    port='COM3',\
    baudrate=9600,\
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
        timeout=0)
print("connected to: " + ser.portstr)
#this will store the line
seq = []
count = 1
contador =0
while count<2:
    for c in ser.read():
        seq.append(chr(c)) #convert from ANSII
        joined_seq = ''.join(str(v) for v in seq) #Make a string from array
        letra = chr(c)
        print (letra)

        if chr(c) == '\n':
            print("Line " + str(count) + ': ' + joined_seq)
            seq = []
            count += 1
            break
ser.close()
# The text that you want to convert to audio
mytext = joined_seq
# Language in which you want to convert
language = 'es'
# Passing the text and language to the engine, here we have marked slow=False. Which tells
# the module that the converted audio should
# have a high speed
myobj = gTTS(text=mytext, lang=language, slow=False)
# Saving the converted audio in a mp3 file named
myobj.save("prueba2.mp3")
```

Figura 26: Programa lector de texto para el Prototipo 1

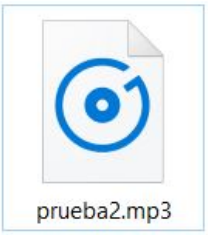


Figura 27: Archivo tipo mp3 generado

```
Line 1: Soy el primer chip con tecnologia nanometrica 100 por ciento disenado por un
programa de lincenciatura de una escuela de ingenieria en la historia de centroamerica.
Mis cradores son: Carlos Esquit, Diego Soler Castaneda, Steven Hiram Rubio, Luis Arturo
Najera, Pablo Ortiz Barillas. Departamento de Ingenieria Electronica. Universidad del
Valle de Guatemala. 2019.
```

Figura 28: Mensaje recibido a través del puerto serial

de *hardware* y se resumió a uno de *software*, lo que representó un menor reto. Como resultado se obtuvo un archivo .mp3 (Figura 27) el cual contenía la voz del traductor de Google articulando el mensaje recibido (ver Figura 28).

9.3. Obstáculos

Como se mencionó en la sección 9.2, al eliminar la implementación de un circuito físico se simplificó significativamente el problema pues el reto se simplificó a un problema de *software*. Sin embargo, existieron puntos importantes los cuales son analizados a continuación:

- **Diferencias librería pySerial v3.0 y v2.5:** La primer dificultad enfrentada fue con respecto al tipo de datos que devuelve el método *read()* de pySerial v3.0 con respecto a sus versiones anteriores. La versión 3.0 devuelve un dato tipo byte, es decir un número en este formato que representa el carácter recibido. Mientras que, el mismo método para la versión utilizada con anterioridad (v2.5) devuelve un dato tipo String por lo que no era necesaria su conversión para previa manipulación.
- **Documentación librería gTTS:** Otra reto importante presentado fue la investigación del funcionamiento de los métodos de la librería gTTS, pues era necesario conocer cuál era el método que devuelve la lectura del mensaje como archivo .mp3 además de los parámetros necesarios que este requería. Como se observa en la Figura 26, el único método utilizado es el que lleva el mismo nombre de la librería (gTTS) y este recibe 3 parámetros que fueron descritos en la sub-sección 9.1.2.
- **Algoritmo de concatenación:** El último reto significativo presentado consistió en la elaboración del proceso de concatenación de caracteres recibidos. Para

ello se hizo uso de las propiedades de los arreglos de Python y finalmente a través del método *join()* de la clase String se agregaba a la secuencia el último elemento del arreglo que contiene los caracteres recibidos via serial.

CAPÍTULO 10

Diseño del sistema que implementa el IC - Prototipo 2

En este capítulo se describe el diseño del prototipo 2 que implementa el circuito integrado a fabricar. El objetivo principal de este prototipo es poder establecer una comunicación exitosa entre el circuito integrado a fabricar el cual generará los caracteres que formarán el mensaje conmemorativo y el programa en Python, descrito en la sección 9.1.2, que leerá dicho mensaje recibido. Para ello se realizaron cambios con respecto al prototipo 1 los cuales se describen en la sección 10.1 y se diseño la topología que se describe en la sección 10.2.

10.1. Cambios con respecto al Prototipo 1

Los cambios realizados no fueron por problemas de funcionamiento del prototipo descrito en el capítulo 9, sino en lograr un sistema más realista que implementara una FSM simulada por un microcontrolador (en este caso nuevamente un Arduino). Los principales cambios se describen en las sub-secciones posteriores.

10.1.1. Circuito generador de señal

El primer cambio realizado con respeto al prototipo 1 fue el diseño de un circuito generador de señal cuadrada. Esto pues con la inclusión de *hardware* entre FSM y programa de lectura, este necesita esta señal para poder funcionar. Este cambio se explica más a fondo en la sección 10.2.1. En el prototipo 1 no fue necesaria pues toda la parte de comunicación serial se realizó a través del Arduino.

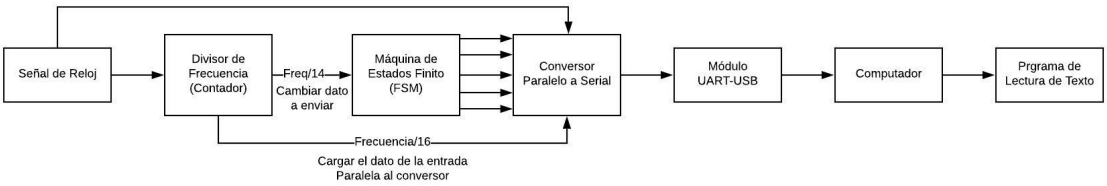


Figura 29: Diagrama de bloques del Prototipo 2

10.1.2. Conversión paralelo-serial

Otro de los cambios realizados para este prototipo consistió en el diseño de un circuito capaz de convertir el valor en código ASCII de cada carécter mostrado en las salidas de la FSM, a una serie de bits capaz de poder ser transmitido a la computadora por medio del puerto USB. De igual forma este cambio se describe de forma más específica en secciones posteriores dentro de este capítulo.

10.2. Topología en alto nivel - Prototipo 2

El diagrama de bloques que implementa la topología del sistema se presenta en la Figura 29. El funcionamiento de esta topología inicia con la señal de reloj, esta se conecta al conversor paralelo a serial, el cual a la frecuencia de esta señal, transporta los bits en su entrada paralela a una serie de datos para posteriormente ser enviado al módulo conversor UART-USB y este puede comunicarse con la computadora y enviarle los datos generados por la FSM. La FSM muestra el valor del siguiente carácter en sus salidas cuando el contador, que en este caso es utilizado para dividir la frecuencia del reloj, muestra en su salida el valor 14. El valor del carácter es cargado a la entrada paralela del conversor paralelo-serial cuando el contador llega a un valor de 16, es necesario hacer este proceso después debido a que cada trama de datos debe estar separada por un ΔT, esto es descrito en la sección 7.1.1. Después de ser cargados los datos en las entradas del conversor, el proceso se inicia de nuevo hasta que el carácter con valor de 10 es enviado y el programa termina la comunicación serial con el circuito. En las siguientes sub-secciones se describe cada uno de los bloques de ese diagrama.

10.2.1. Señal de reloj

La señal de reloj es una parte vital de todos los circuitos digitales, pues marca el ritmo al que el sistema debe funcionar. Generalmente es una onda cuadrada periódica con ciclo de trabajo simétrico, es decir mismo tiempo en *HIGH* y en *LOW*. En el sistema a implementar, este bloque estará formado por un circuito generador de señales a una frecuencia de 9.6KHz, la cual es una de las frecuencias estándar de

transmisión serial que se utilizan.

10.2.2. Divisor de frecuencia

El divisor de frecuencia en este prototipo fue construido mediante 1 contador de 4 bits, el circuito integrado TTL que lo implementa es el 74LS193. Este contador tiene conectado a su entrada de reloj ascendente la señal del reloj descrita en 10.2.1. Para lograr ambas salidas de frecuencia se realizó un AND en los pines del salida del contador que representan el número 14 en decimal cuando están en uno. La salida de este AND estará conectada a la señal de reloj de la FSM. Mientras que, la otra frecuencia es obtenida del acarreo que ocurre cada vez que el contador llega a 15. Este pin es conectado al pin de carga de la entrada paralela del conversor paralelo-serial descrito en la sección 10.2.4.

Es necesario utilizar 2 frecuencias más pues para que la conversión funcione de manera correcta el cambio de bits que genera la FSM así como la carga de los mismos al conversor paralelo-serial se debe realizar a una frecuencia menor a la de transmisión de datos, esto pues se debe asegurar que el corrimiento de bits realizados por parte del conversor se haya completado antes de cargar un nuevo valor entrada paralela.

10.2.3. FSM

Para este prototipo se simuló la FSM con un Arduino. Este recibiría una señal, el reloj del sistema, que le indica cuando cambiar el valor de la representación del carácter en sus salidas, las cuales estarán conectadas a la entrada paralela del conversor paralelo-serial descrito en 10.2.4.

El código de Arduino utilizado se muestra en las sección 14.2 de los anexos.

Parte de los cambios realizados en el código que simula la FSM con respecto al prototipo 1, es que debido a la utilización del hardware de conversión paralelo-serial. Para ello se representó el valor ASCII de cada carácter a enviar en 8 diferentes pines del Arduino, estos fueron los siguientes: 9,8,7,6,5,4,3,2. Done el pin 9 representa el bit más significativo y el 2 el menos significativo. La forma de mapeo de cada valor se presenta en la última parte del código en donde se utiliza un AND lógico y luego a través de un ciclo for se escribe el valor en los pines de salida.

10.2.4. Conversor paralelo a serial

Para poder transmitir el mensaje de la FSM al computador fue necesaria la implementación de una etapa de conversión paralelo-USB, pues las computadoras actualmente se comunican con dispositivos externos, también llamados periféricos, a través de este último protocolo el USB. Este protocolo es descrito a detalle en la sección

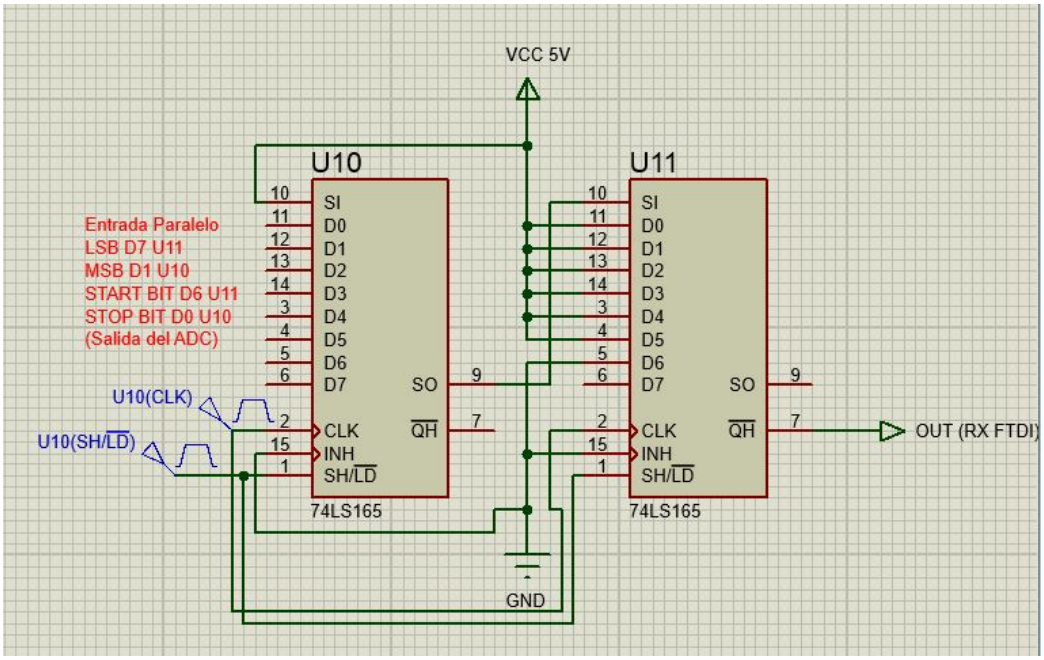


Figura 30: Diagrama convertidor paralelo-serial

5.9.1.

Como se mencionó en la sección anterior (8.1), la máquina de estados finitos diseñada tiene 8 salidas, las cuales son necesarias para la representación de los caracteres en código ASCII, por lo que el conversor serial requerido debe de ser de 8 bits. Para esto se buscó los circuitos integrados que permiten la conversión de un sistema a otro, estos son los *'shift registers'* los cuales no son más que una combinación entre *flip-flops* y compuertas lógicas que permite dicha conversión. Entre las opciones cotejadas se seleccionó el circuito integrado *74LS165* pues permite la conversión paralelo a serial de un dato de 8 bits.

Además de la tranmisión de los 8 bits de información el protocolo UART requiere 2 bits mas de control, como se mencionó en la sección 5.9.1, al ser un protocolo asíncrono, es necesario indicar cuando inicia y cuando finaliza la trama de información. Para agregar estos bits de control fue necesario la implementación de un segundo IC 74LS165 en serie. Los bits no utilizados se encuentran en *HIGH*. El diagrama de conexión se muestra en la Figura 30

10.2.5. Modulo UART-USB

La transmisión de información de forma serial se hace siguiendo las pautas del protocolo de transmisión seleccionado. Sin embargo, como se mencionó anteriormente las computadores actuales solamente se comunican a través del protocolo USB, por ende fue necesario utilizar un conversor UART-USB (Figura 31), quien fue el encargado de ser la interfaz entre el circuito y la computadora.

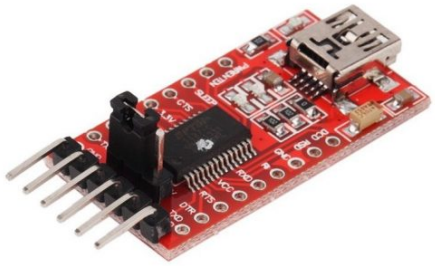


Figura 31: Modulo conversor UART-USB

10.3. Pruebas y resultados

Al implementar el sistema descrito, se pude comprobar el funcionamiento del hardware diseñado y construido en un *protoboard*, en la Figura 32 se observa la construcción del hardware.

Para comprobar el correcto funcionamiento de este se realizó el mismo proceso que en el prototipo 1 y se procedió a enviar un mensaje que luego sería leído por medio del programa en Python. El resultado de los caracteres enviados hacia el computador y concatenados por el programa lector del texto se muestra en la Figura 33.

Así mismo utilizando un osciloscopio se pudo comprobar tanto la frecuencia de transmisión, la cual es igual al ancho de un bit enviado hacia el computador (Figura 34), como el valor de una de las tramas enviadas (Figura 35). En esta última figura la señal del canal 2 (gráfica amarilla) representa el pulso para la carga de los datos en la entrada paralela del conversor. Se observa que después de cada pulso, la señal de datos representado con la gráfica azul permanece por un tiempo en *HIGH* y luego se

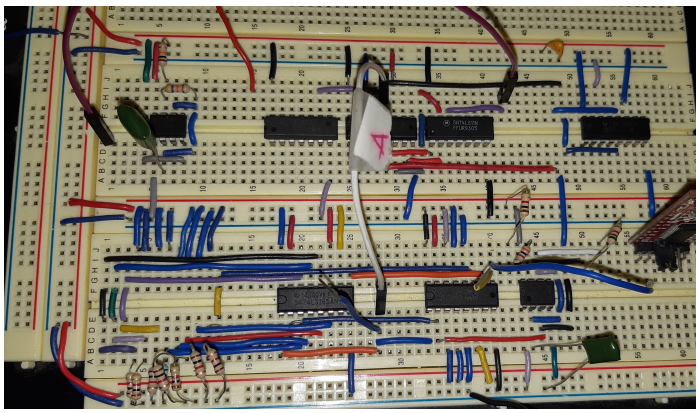


Figura 32: Implementación del Protitopo 2 en un Protobard

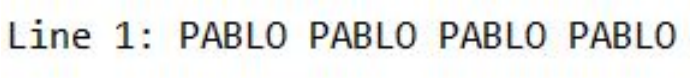


Figura 33: Mensaje formado por los caracteres generados por la FSM simulada

observa el *Start Bit*, este retraso de envío de la trama se debe que al inicio se envían los 6 bits sin utilizar del circuito integrado 74LS165 (IC U11 en la Figura 13) y luego la trama de datos.

10.4. Obstáculos

Para la implementación de este prototipo, los retos más significativos fueron principalmente de diseño de *hardware*. Los mismo se describen a continuación:

- **Circuito divisor de frecuencia:** El principal reto que presentó este módulo fue la selección del circuito que simplificara la implementación y el cableado de este, pues existen varias opciones entre las cuales se pueden mencionar, los divisores utilizando flip-flops. Finalmente se optó por la utilización de un contador. Seguidamente, otro reto fue la selección del factor de división de frecuencia, pues primero se debía calcular el tiempo que demora la trama de datos en salir del conversor.
- **Obtención de la frcuencia de reloj:** Otro reto presentado consistió en la obtención de la frecuencia de reloj lo más exacta posible y que al mismo tiempo la tolerancia de error con respecto a la frecuencia de transmisión teórica no afectara el envío de datos.
- **Lectura de la señal de reloj en Arduino:** El reto presentado en este item, se refiere a que debido a la forma de lectura del estado de pines configurados como entradas digitales en Arduino es en base al estado de los mismos ($HIGH$, LOW), fue necesario calcular previamente el ancho del pulso de esta señal. Este cálculo se realizó como se muestra en la ecuación 4. Con esto en mente únicamente se debía garantizar que el delay realizadod en Arduino fuera mayor que este valor, por ende se utilizó uno de $20\mu S$.

$$T_{pulso} = \frac{1}{9600KHz} = 104.17\mu S \tag{4}$$

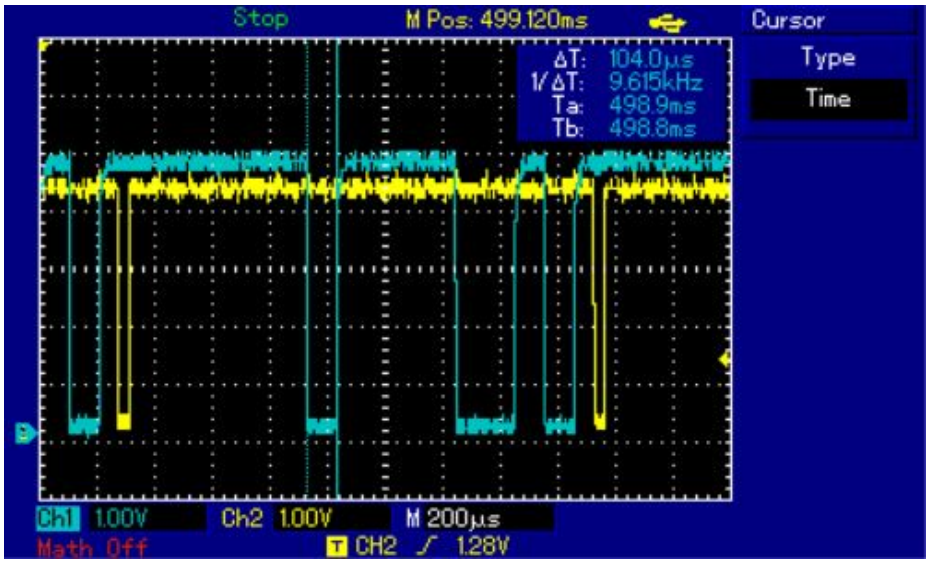


Figura 34: Velocidad de transmisión de datos

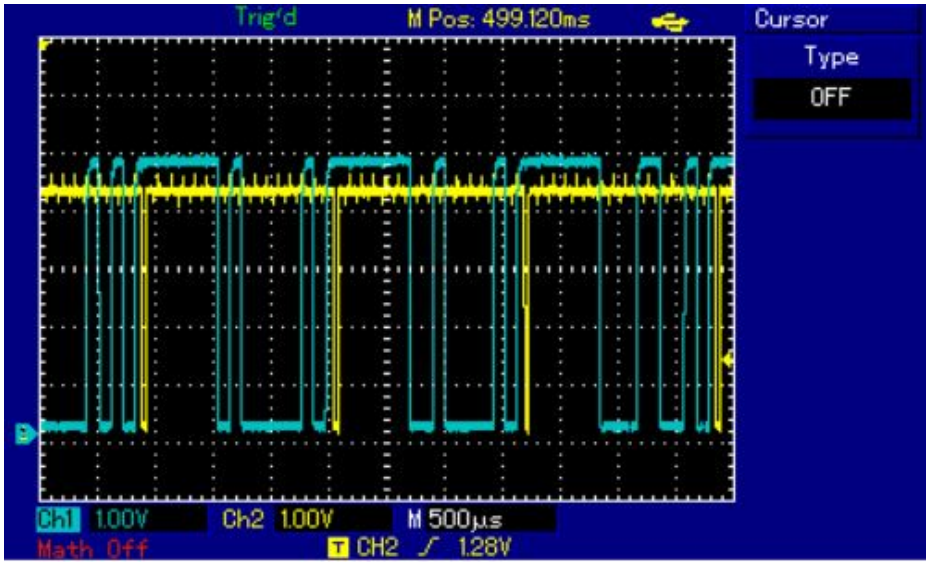


Figura 35: Trama de datos enviada

CAPÍTULO 11

Conclusiones

- De acuerdo con los resultados de la herramienta VCS mostrados en la sección 8.4 se pudo observar que las salaidas de la FSM fueron las esperadas en cada flanco de reloj, lo mismo verifica el correcto funcionamiento de la arquitectura del circuito integrado a fabricar.
- De acuerdo con los resultados mostrados en las secciones 9.2 y 10.3 se verifica el correcto funcionamiento del diseño del sistema que implementará el circuito integrado a fabricar, pues fue posible el envío y procesamiento de los caracteres generados por la FSM simulada además de la articulación del mensaje formado con dichos caracteres.
- A través del motor de búsqueda *Any Silicon* se encontraron las distintas empresas las cuales son las encargadas de intermediar con los fabricantes a gran escala de semiconductores. La principal función de dichas empresas se menciona en la sección 6.2.1.
- Las características principales características para selección del fabricante con el que se trabajará son:
  - Servicios MPW
  - Acceso a documentos de diseño y librerías de componentes
  - Servicio de encapsulamiento
  - Acceso a diseños IP como memorias, procesadores, etc.
- Las características principales para seleccionar un proceso de fabricación o tecnología con un fabricante se relacionan a los requerimientos y especificaciones de funcionamiento y diseño, dichas características son:
  - Máxima frecuencia de operación requerida
  - Restricciones físicas (dimensiones del circuito integrado a fabricar)

    - Restricciones de disipación de potencia activa y estática
    - Implementación del árbol del reloj

- En base a las dos conclusiones anteriores y a la descripción realizada en la sección 6.2.1 se seleccinó un proceso de fabricación de 180nm ofrecido por TSMC mediante la empresa intermediaria *Europractice*

- La validación de aspectos y especificaciones técnicas del proceso de fabricación, en otras palabras, la aplicación de las reglas de diseño no se ha completado pues el proceso de comunicación con la empresa intermediaria aún se encuentra en fase legal, pues no ha sido aprobado por su parte el *NDA* firmado por los integrantes del grupo de trabajo. Por ende, no se ha hecho entrega de los documentos y librerías de diseño.

CAPÍTULO 12

Recomendaciones

- Según las características mencionadas en la sección 6.2.1 y a las eventualidades descritas en la sección 6.3.3 se recomienda considerar dichos aspectos al momento de selección de fabricante y selección del proceso de fabricación a utilizar.
- Uno de los principales factores de las eventualidades descritas en la sección 6.3.3 fue debido al cambio de los requisitos de la empresa intermediaria con sus clientes, por lo mismo, se recomienda a futuros diseñadores mantener una comunicación eficiente con la empresa seleccionada además de clarificar todas las dudas en cuanto a requerimientos se refiere con la persona de contacto.
- Debido a que la fase legal de la comunicación con los fabricantes requiere un tiempo de espera significativo, se recomienda que durante ese tiempo de espera buscar nuevas opciones de fabricantes en caso de alguna eventualidad.
- Se recomienda realizar el proceso de verificación de la arquitectura del circuito integrado diseñado en un FPGA, pues la validación esta en una plataforma física evita futuros problemas de implementación cuando la arquitectura diseñada es significativamente compleja.
- Debido a que la forma de validación del funcionamiento del circuito integrado fabricado se realizará a través de la lectura de un mensaje conmemorativo y debido a la naturaleza del proyecto, se recomienda la implementación de un entorno más amigable para el el usuario. Por ejemplo, la implementación de una interfaz gráfica.
- Se recomienda que el sistema que implementa a la FSM se diseñe con una configuración en lazo cerrado, es decir que el programa pueda enviar al sistema cuando sea requerido el envío de los caracteres para la formación del mensaje. Esto evitará un consumo de energía y recursos innecesarios.

CAPÍTULO 13

Bibliografía

[1] D. Diaz, "Flujo de diseño de circuitos integrados digitales aplicado al desarrollo de un controlador USB 2.0".

[2] K. Tiri e I. Verbauwhede, "A Digital Design Flow for Secure Integrated Circuits", *IEE Transaction on computer-aided design of integrated circuits and systems*, vol. 25, n.º 7, pág. 12, 2006.

[3] J. Parra, F. Andrés y B. Rafael, "Diseño de circuitos VLSI a partir de VHDL", *Visión Electrónica*, vol. 1, n.º 1, pág. 8, 2008.

[4] e. MacDonald Eric, "Enseñando a diseñar circuitos integrados digitales usando herramientas Synopsys", 2012.

[5] F. Fitchen, "Circuitos integrados y sistemas", en *Circuitos integrados y sistemas*, Reverte, 1975, pág. 461.

[6] N. Prize. (2014). The History of the Integrated Circuit, dirección: `http://www.nobelprize.org/educational/physics/integrated_circuit/history/index.html` (visitado 10-03-2019).

[7] IntelCorporation, "De la arena al silicio "La fabricación de un Chip" Historia ilustrada", Intel, 2009.

[8] M. Wu y R.-B. Lin, "Multiple Project Wafers for Medium-Volume IC production", en *IEEE International symposium on Circuits and Systems*, IEEE, 2005, págs. 4725-4728.

[9] I. Synopsys, "Synopsys Design Compilere User Guide", en *Design Compiler User Guide*, Synopsys, 2005, págs. 103-118.

[10] C. Mead y L. Conway, "Integrated System Fabrication", en *Introduction to VLSI Systems*, Addison Wesley, 1980, págs. 60-65.

[11] P. Kogge, "VLSI Design Rules", en *Introduction to CMOS VLSI Design*, University of Notre Dame, 2015, págs. 1-17.

[12] L. Scheffer L. Lavagno y G. Martin, "RTL to GDS-II, or Synthesis, Place and Route", en *EDA for IC Implementation, Circuit Design and Process Technology*, Taylor y Francis Group, 2006, págs. 20-26.

[13] E. Larsson, "Design Flow", en *Introduction to Advanced System-On-Chip Test Design and Optimization*, Springer, 2005, págs. 5-6.

[14] D. e. a. Jansen, "The Concept of Electronic Design Automation", en *The Electronic Design Automation Handbook*, Kluwer Academic Publishers, 2003, págs. 46-48.

[15] H. Bhatnagar, "ASIC Design Methodology", en *Advanced ASIC Chip Synthesis Using Synopsys Design Compiler, Physical Compiler and PrimeTime*, Kluwer Academic Publishers, 2002, págs. 1-17.

[16] S. Bali, "Registers and Counters", en *2000 Solved Problems in Digital Electronics*, Tata McGraw-Hill, 2005, pág. 84.

[17] M. Mano, "Análisis de Circuitos Secuenciales Con Reloj", en *Diseño Digital*, Pearson Education, 2002, págs. 189-190.

[18] R. Czerwinski y D. Kania, "Finite State Machine Logic Synthesis for Complex Programmable Logic Devices", en *Lecture Notes in Electrical Engineering 231*, Springer, 2013, págs. 9-11.

[19] P. Minns e I. Elliot, "Introduction to FSM and State Diagramas for the Design of Electronic Circuits and Systems", en *FSM-based Digital Design using Verilog HDL*, John Wiley y Sons, Ltd., 2008, pág. 407.

[20] M. Rabiee, "Parallel-to-Serial and Serial-to-Parallel Converters", en *American Society for Engineering Education Annual Conference and Exposition*, American Society for Engineering Education, 2002, pág. 17.

[21] J. Tahir, "Serial Communication", en *Arm Microprocessor Systems Cortex-m Architecture Programming and Interfacing*, American Society for Engineering Education, 2017, págs. 400-484.

[22] S. Chazallet, "Presentación de Python", en *Python 3 Los fundamentos del lenguaje*, Ediciones Eni, 2016, pág. 904.

[23] G. Rossum, "Módulos", en *El tutorial de Python*, Python Sotware Foundation, 2009, pág. 116.

[24] Python. (2015). pySerial API, dirección: `https://pythonhosted.org/pyserial/pyserial_api.html` (visitado 10-08-2019).

[25] ——, (2018). gTTS, dirección: `https://pypi.org/project/gTTS/` (visitado 10-08-2019).

[26] I. Synopsys, "VCS User Guide", en *VCS User Guide*, Synopsys, 2019, pág. 2106.

CAPÍTULO 14

Anexos

14.1. Ecuaciones de la FSM

14.1.1. Ecuaciones de transición de estados

A continuación se muestran las ecuaciones para la función de transición de estados.

```
D8 = Q8 Q7' Q6' + Q8 Q7' Q5' Q4'  +  Q8 Q7' Q5' Q3'  +  Q8 Q7' Q5' Q2'
Q1' Q0' + Q8' Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0;

D7 = Q8' Q7 Q6'  + Q8' Q7 Q5'  + Q8' Q7 Q4'  + Q8' Q7 Q3'  + Q8' Q7 Q2'
+ Q8' Q7 Q1'  + Q8' Q7 Q0' + Q8' Q7' Q6 Q5 Q4 Q3 Q2 Q1 Q0;

D6 = Q8' Q6 Q5'  + Q8' Q6 Q4'  + Q8' Q6 Q3'  + Q8' Q6 Q2'  + Q8' Q6 Q1'
+ Q8' Q6 Q0' + Q7' Q6 Q5' Q4'  + Q7' Q6 Q5' Q3'  + Q7' Q6 Q5' Q2' Q1' Q0'
+ Q8' Q6' Q5 Q4 Q3 Q2 Q1 Q0 + Q7' Q6' Q5 Q4 Q3 Q2 Q1 Q0;

D5 = Q8' Q5 Q4'  + Q8' Q5 Q3'  + Q8' Q5 Q2'  + Q8' Q5 Q1'  + Q8' Q5 Q0' +
Q7' Q6' Q5 Q4'  + Q7' Q6' Q5 Q3'  + Q7' Q6' Q5 Q2'  + Q7' Q6' Q5 Q1'  +
Q7' Q6' Q5 Q0' + Q8' Q5' Q4 Q3 Q2 Q1 Q0 + Q7' Q6' Q5' Q4 Q3 Q2 Q1 Q0;

D4 = Q8' Q4 Q3'  + Q8' Q4 Q2'  + Q8' Q4 Q1'  + Q8' Q4 Q0' + Q7' Q6' Q4
Q3'  + Q7' Q5' Q4 Q3'  + Q7' Q6' Q4 Q2'  + Q7' Q6' Q4 Q1'  + Q7' Q6' Q4
Q0' + Q8' Q4' Q3 Q2 Q1 Q0 + Q7' Q5' Q4 Q2' Q1' Q0' + Q7' Q6' Q4' Q3 Q2 Q1
Q0 + Q7' Q5' Q4' Q3 Q2 Q1 Q0;

D3 = Q8' Q3 Q2'  + Q8' Q3 Q1'  + Q8' Q3 Q0' + Q7' Q6' Q3 Q2'  + Q7' Q6'
Q3 Q1'  + Q7' Q6' Q3 Q0' + Q7' Q5' Q4' Q3 Q2'  + Q7' Q5' Q4' Q3 Q1'  +
```

```
Q8’ Q3’ Q2 Q1 Q0 + Q7’ Q5’ Q4’ Q3 Q0’ + Q7’ Q6’ Q3’ Q2 Q1 Q0 + Q7’ Q5’
Q3’ Q2 Q1 Q0 + Q7’ Q5’ Q3 Q2’ Q1’ Q0’;

D2 = Q8’ Q2 Q1’  + Q8’ Q2 Q0’ + Q7’ Q6’ Q2 Q1’  + Q8’ Q2’ Q1 Q0 + Q7’ Q6’
Q2 Q0’ + Q7’ Q5’ Q4’ Q2 Q1’  + Q7’ Q5’ Q3’ Q2 Q1’  + Q7’ Q6’ Q2’ Q1 Q0 +
Q7’ Q5’ Q4’ Q2 Q0’ + Q7’ Q5’ Q3’ Q2 Q0’ + Q7’ Q5’ Q4’ Q2’ Q1 Q0 + Q7’ Q5’
Q3’ Q2’ Q1 Q0;

D1 = Q8’ Q1’ Q0 + Q8’ Q1 Q0’ + Q7’ Q6’ Q1’ Q0 + Q7’ Q6’ Q1 Q0’ + Q7’ Q5’
Q4’ Q1’ Q0 + Q7’ Q5’ Q3’ Q1’ Q0 + Q7’ Q5’ Q4’ Q1 Q0’ + Q7’ Q5’ Q3’ Q1
Q0’;

D0 = Q8’ Q0’ + Q7’ Q6’ Q0’ + Q7’ Q5’ Q4’ Q0’ + Q7’ Q5’ Q3’ Q0’ + Q7’ Q5’
Q2’ Q1’ Q0’
```

14.1.2. Ecuaciones de la función de salida

A continuación se muestran las ecuaciones para la función de salidas de la FSM.

```
S8 = Q8’ Q7’ Q6’ Q5 Q4 Q3’ Q2’ Q1 Q0’ + Q8’ Q7 Q6 Q5’ Q4’ Q3’ Q2’ Q1 Q0’;

S7 = Q7’ Q6’ Q1’ Q0’ + Q8’ Q7 Q5 Q2’ Q1  + Q8’ Q6 Q5 Q4 Q1’ Q0 + Q7’ Q5’
Q3’ Q2’ Q1’ Q0’ + Q8 Q7’ Q6 Q5’ Q4’ Q1 Q0’ + Q7’ Q6’ Q5’ Q4 Q3  + Q8 Q7’
Q5’ Q4’ Q3  + Q8 Q7’ Q5’ Q4’ Q2’  + Q8’ Q6’ Q5’ Q3 Q2’  + Q7’ Q6’ Q4’ Q3
Q2’  + Q8’ Q7’ Q5’ Q3 Q1  + Q8 Q7’ Q6’ Q4 Q1’  + Q8’ Q6’ Q3’ Q2 Q1’  +
Q8’ Q5’ Q3’ Q2’ Q1’  + Q8’ Q7 Q4 Q2 Q0 + Q8’ Q6’ Q4 Q2’ Q0 + Q8’ Q5 Q3
Q2’ Q0 + Q7’ Q6’ Q5 Q1 Q0 + Q7’ Q6’ Q2 Q1 Q0 + Q8’ Q5’ Q3’ Q1’ Q0 + Q8’
Q7’ Q5’ Q1’ Q0’ + Q8 Q7’ Q6’ Q5 Q3’ Q2  + Q8’ Q7 Q6 Q4’ Q3 Q2’  + Q8’ Q6
Q5’ Q4 Q3’ Q2’  + Q8’ Q7 Q6’ Q4’ Q3’ Q1  + Q8’ Q7 Q5’ Q4 Q2 Q1  + Q8’ Q7
Q4 Q3 Q2 Q1  + Q8’ Q7 Q6 Q5’ Q3 Q1’  + Q8’ Q6 Q4’ Q3’ Q2’ Q0 + Q8’ Q7 Q6’
Q5’ Q1 Q0 + Q8’ Q6 Q4’ Q3 Q1 Q0 + Q8’ Q7’ Q3’ Q2 Q1 Q0 + Q8’ Q7 Q5 Q2 Q1’
Q0 + Q8 Q7’ Q6’ Q3’ Q2’ Q0’ + Q8’ Q7’ Q4’ Q3’ Q2’ Q0’ + Q8’ Q7’ Q6 Q5 Q1
Q0’ + Q8’ Q5 Q4’ Q3’ Q1 Q0’ + Q8’ Q7 Q4’ Q2 Q1’ Q0’ + Q8’ Q5 Q3’ Q2 Q1’
Q0’ + Q8’ Q4 Q3 Q2’ Q1’ Q0’ + Q7’ Q6 Q5’ Q4’ Q2 Q1’ Q0 + Q8’ Q6’ Q5 Q3 Q2
Q1 Q0’;

S6 = Q8 Q7’ Q6 Q5’ Q4 Q3’ Q1  + Q7’ Q6’ Q5’ Q3’ Q2 Q1 Q0’ + Q8’ Q7 Q5 Q3’
Q2’ Q1’ Q0’ + Q8’ Q7 Q6 Q5’ Q4 Q3 Q2’ Q1  + Q8 Q7’ Q6 Q5’ Q4 Q3 Q2’ Q1’
+ Q8 Q7’ Q6 Q5’ Q3’ Q2 Q1 Q0 + Q8’ Q7 Q6 Q4’ Q3’ Q2 Q1 Q0 + Q8 Q7’ Q5’ Q4
Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6 Q5’ Q4 Q3 Q1’ Q0 + Q8’ Q7’ Q6 Q5 Q4’ Q2 Q1’ Q0
+ Q8’ Q7’ Q6’ Q5 Q3 Q2 Q1’ Q0 + Q7’ Q6’ Q5 Q4’ Q3 Q2 Q1’ Q0 + Q8’ Q6’ Q5’
Q4’ Q3 Q2 Q1’ Q0 + Q8’ Q7’ Q6 Q5’ Q3 Q2’ Q1’ Q0 + Q7’ Q6’ Q5 Q4’ Q3’ Q2’
Q1’ Q0 + Q8’ Q7 Q6 Q5’ Q4’ Q3’ Q1 Q0’ + Q8 Q7’ Q6’ Q5 Q3 Q2 Q1 Q0’ + Q8’
Q7 Q6 Q4’ Q3 Q2 Q1 Q0’ + Q8’ Q7 Q5’ Q4’ Q3 Q2 Q1 Q0’ + Q8’ Q7 Q5 Q4 Q3’
Q2 Q1 Q0’ + Q8’ Q6’ Q5’ Q4 Q3’ Q2’ Q1 Q0’ + Q8’ Q7’ Q6 Q5 Q4’ Q3 Q1’ Q0’
```

```
+ Q8’ Q7 Q6’ Q4 Q3 Q2 Q1’ Q0’ + Q8 Q7’ Q6 Q5’ Q3’ Q2 Q1’ Q0’ + Q8’ Q7 Q6’
Q5 Q4’ Q2’ Q1’ Q0’ + Q8’ Q6 Q5 Q4 Q3’ Q2’ Q1’ Q0’ + Q8’ Q7’ Q6 Q5 Q4 Q3
Q2 Q1 Q0 + Q8’ Q7 Q6’ Q5 Q4’ Q3 Q2 Q1 Q0 + Q8’ Q7’ Q6 Q5 Q4 Q3’ Q2’ Q1 Q0
+ Q8’ Q7’ Q6’ Q5’ Q4’ Q3’ Q2’ Q1 Q0 + Q8 Q7’ Q6’ Q5’ Q4’ Q3’ Q2 Q1’ Q0 +
Q8’ Q7 Q6 Q5’ Q4 Q3’ Q2 Q1’ Q0’ + Q8 Q7’ Q6 Q5’ Q4 Q3’ Q0 + Q8’ Q7’ Q6’
Q4 Q3’ Q1 Q0’ + Q8’ Q7’ Q5’ Q3’ Q2 Q1 Q0’ + Q8’ Q6’ Q5 Q4’ Q3’ Q2’ Q1’ Q0
+ Q7’ Q6’ Q5 Q4 Q3 Q2’ Q1 Q0’ + Q8’ Q6 Q5 Q4 Q3 Q2 Q1’ Q0’;

S5 = Q8 Q7’ Q5’ Q4’ Q3 Q2’ Q0 + Q8’ Q7 Q5 Q4’ Q2 Q1’ Q0’ + Q8’ Q7’ Q6’ Q5
Q4 Q3 Q2 Q1  + Q8’ Q7’ Q6 Q5 Q4 Q3’ Q2 Q1  + Q8 Q7’ Q6’ Q5 Q4’ Q3’ Q2 Q1
+ Q8’ Q7 Q6 Q5 Q4’ Q3’ Q2’ Q1  + Q8 Q7’ Q6’ Q5 Q4 Q3’ Q2 Q1’  + Q8 Q7’ Q6
Q5’ Q4 Q3’ Q2 Q1’  + Q8’ Q7 Q6 Q5 Q4’ Q3’ Q2 Q1’  + Q8’ Q7 Q6 Q5 Q4 Q3
Q2’ Q1’  + Q8’ Q7 Q6 Q5’ Q4’ Q3 Q2’ Q0 + Q8’ Q7 Q6 Q5 Q3 Q2 Q1 Q0 + Q8’
Q6 Q5’ Q4 Q3 Q2 Q1 Q0 + Q8’ Q7 Q6’ Q5 Q3’ Q2 Q1 Q0 + Q8’ Q7’ Q6’ Q5’ Q3’
Q2 Q1 Q0 + Q7’ Q6 Q5’ Q4 Q3’ Q2’ Q1 Q0 + Q7’ Q6’ Q5’ Q4 Q3’ Q2’ Q1’ Q0 +
Q8 Q7’ Q6 Q5’ Q4 Q3’ Q2 Q0’ + Q8’ Q7’ Q6 Q5’ Q4 Q2’ Q1 Q0’ + Q8’ Q7 Q6’
Q5 Q4’ Q3’ Q1’ Q0’ + Q8 Q7’ Q6’ Q5’ Q4’ Q2 Q1’ Q0’ + Q8’ Q7 Q6’ Q5’ Q3’
Q2 Q1’ Q0’ + Q7’ Q6 Q5’ Q4 Q3’ Q2 Q1’ Q0’ + Q8’ Q7 Q6 Q5’ Q4’ Q2’ Q1’ Q0’
+ Q8’ Q7 Q6’ Q5’ Q4’ Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6 Q5 Q4’ Q3’ Q2’ Q1’ Q0 +
Q8 Q7’ Q6’ Q5’ Q4 Q3 Q2 Q1 Q0’ + Q8’ Q7 Q6 Q5’ Q4 Q3’ Q2 Q1 Q0’ + Q8 Q7’
Q6’ Q5 Q4 Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6’ Q5 Q4 Q3’ Q2’ Q1’ Q0 + Q8’ Q7 Q5
Q4 Q3 Q1 Q0 + Q8’ Q6’ Q5 Q4 Q2 Q1 Q0 + Q8’ Q7’ Q6’ Q5 Q4’ Q3 Q2’ Q1’  +
Q8’ Q7’ Q6’ Q5 Q4 Q3 Q2’ Q0 + Q8 Q7’ Q6’ Q5 Q3 Q2 Q1 Q0 + Q8’ Q7 Q6’ Q5
Q3 Q2’ Q1 Q0 + Q8’ Q7 Q6’ Q4 Q3 Q2’ Q1 Q0 + Q8’ Q7 Q6 Q5’ Q4 Q2 Q1’ Q0 +
Q8’ Q7 Q5’ Q4 Q3 Q2 Q1’ Q0 + Q8’ Q7 Q6’ Q5’ Q3 Q2’ Q1’ Q0 + Q8’ Q7’ Q5 Q4
Q3 Q2’ Q1’ Q0 + Q8’ Q7 Q6 Q4 Q3’ Q2’ Q1’ Q0 + Q8’ Q7 Q5’ Q4 Q3’ Q2’ Q1’
Q0 + Q8’ Q7 Q6’ Q5 Q4’ Q3 Q1 Q0’ + Q8’ Q7’ Q5’ Q4’ Q3’ Q2’ Q1 Q0’ + Q8’
Q7’ Q6’ Q5’ Q4’ Q3 Q1’ Q0’ + Q8’ Q7’ Q6’ Q5’ Q4’ Q2’ Q1’ Q0’;

S4 = Q8’ Q6 Q5 Q4 Q3’ Q2 Q1’  + Q8’ Q7 Q6’ Q5’ Q4’ Q2 Q0 + Q8’ Q7 Q5’ Q4’
Q3 Q2 Q0 + Q7’ Q6’ Q5’ Q4 Q3 Q1 Q0 + Q8’ Q5 Q4’ Q3 Q2 Q1 Q0 + Q7’ Q6’ Q5’
Q4’ Q3’ Q1’ Q0 + Q8’ Q7 Q6 Q5’ Q4 Q3 Q0’ + Q8 Q7’ Q6’ Q5’ Q3’ Q2’ Q0’ +
Q8’ Q6’ Q5’ Q4’ Q2’ Q1 Q0’ + Q8 Q7’ Q6 Q5’ Q4 Q3’ Q2 Q1  + Q8’ Q7 Q6 Q5
Q4 Q3 Q2’ Q1  + Q8’ Q7 Q6’ Q5 Q4 Q3 Q2’ Q1’  + Q8’ Q7 Q6 Q5’ Q4 Q3 Q2’
Q1’  + Q8 Q7’ Q6’ Q5 Q4 Q3’ Q2’ Q1’  + Q8 Q7’ Q6 Q5’ Q4’ Q3 Q2 Q0 + Q8’
Q7 Q6 Q5 Q4 Q3’ Q2 Q0 + Q8’ Q7’ Q6 Q5 Q4 Q3’ Q1’ Q0 + Q8 Q7’ Q6’ Q5 Q3 Q2
Q1’ Q0 + Q8’ Q7’ Q6’ Q5’ Q4’ Q2’ Q1’ Q0 + Q8 Q7’ Q6 Q5’ Q3’ Q2’ Q1’ Q0 +
Q8’ Q7 Q6 Q5 Q4’ Q3 Q2’ Q0’ + Q8’ Q7’ Q6 Q5’ Q4’ Q3 Q1 Q0’ + Q8’ Q7 Q6
Q4’ Q3’ Q2 Q1 Q0’ + Q8 Q7’ Q5’ Q4’ Q3’ Q2’ Q1 Q0’ + Q8’ Q7’ Q6 Q5’ Q4’
Q3’ Q1’ Q0’ + Q8’ Q7 Q6 Q5’ Q4 Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6 Q5 Q4’ Q3’ Q2’
Q1 Q0 + Q8 Q7’ Q6’ Q5 Q4’ Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6 Q5 Q4’ Q3 Q2’ Q1’ Q0
+ Q8 Q7’ Q6’ Q5’ Q4’ Q3 Q2 Q1 Q0’ + Q8 Q7’ Q6’ Q5 Q4 Q3’ Q2 Q1 Q0’ + Q8’
Q7 Q6’ Q5 Q4 Q3’ Q2’ Q1 Q0’ + Q8’ Q7’ Q6’ Q5 Q4’ Q3 Q2 Q1’ Q0’ + Q8’ Q7
Q6’ Q5 Q4’ Q3’ Q2’ Q1’ Q0’ + Q8’ Q7’ Q6’ Q5’ Q4 Q2 Q1’  + Q8’ Q6’ Q5’ Q3
Q2 Q1 Q0 + Q8’ Q7 Q6’ Q4’ Q2 Q1’ Q0 + Q8’ Q6’ Q5 Q3’ Q2 Q1’ Q0 + Q8’ Q7’
Q6’ Q4’ Q2’ Q1 Q0’ + Q8’ Q6 Q5 Q3 Q2’ Q1 Q0’ + Q8’ Q7 Q5’ Q4 Q2’ Q1’ Q0’
```

```
+ Q8’ Q5’ Q4 Q3’ Q2’ Q1’ Q0’ + Q8’ Q7 Q6’ Q5’ Q4 Q3 Q2 Q1  + Q8 Q7’ Q6’
Q5 Q4’ Q3 Q2’ Q1’  + Q8 Q7’ Q5’ Q4 Q3’ Q2 Q1 Q0 + Q8’ Q7’ Q6 Q5’ Q3 Q2’
Q1 Q0 + Q8’ Q7’ Q6’ Q5 Q4’ Q3’ Q2 Q0’ + Q8’ Q7’ Q6 Q5 Q4 Q3 Q2’ Q0’ + Q8’
Q6’ Q5 Q4’ Q3’ Q2 Q1 Q0’ + Q8’ Q7’ Q6’ Q5’ Q3 Q2’ Q1 Q0’ + Q7’ Q6’ Q5 Q4’
Q3 Q2’ Q1 Q0’ + Q8’ Q7’ Q6’ Q5 Q4 Q3 Q1’ Q0’ + Q8 Q7’ Q6’ Q5’ Q4 Q3 Q1’
Q0’;

S3 = Q8’ Q7’ Q6 Q5 Q4’ Q1  + Q8’ Q6’ Q3’ Q2 Q1’ Q0 + Q8’ Q7’ Q6 Q5’ Q4
Q2’ Q1  + Q8’ Q7’ Q6’ Q5’ Q3 Q2’ Q1  + Q8 Q7’ Q6’ Q5 Q3’ Q2’ Q1  + Q8’ Q7
Q5’ Q4’ Q3’ Q2’ Q1  + Q8’ Q7 Q6 Q5’ Q3 Q2 Q1’  + Q8 Q7’ Q6’ Q5 Q4 Q1 Q0 +
Q8’ Q7’ Q6 Q3’ Q2 Q1 Q0 + Q7’ Q6’ Q4 Q3 Q2’ Q1 Q0 + Q8 Q7’ Q5’ Q3’ Q2’
Q1’ Q0 + Q7’ Q6’ Q5 Q4’ Q3’ Q2 Q0’ + Q8’ Q7 Q6’ Q4 Q3 Q2’ Q0’ + Q8 Q7’
Q6’ Q4 Q3’ Q2’ Q0’ + Q8’ Q7’ Q6 Q3 Q2 Q1 Q0’ + Q7’ Q6’ Q5 Q3’ Q2’ Q1 Q0’
+ Q8’ Q7’ Q4 Q3 Q2’ Q1’ Q0’ + Q8’ Q7 Q6’ Q5 Q4 Q3’ Q2’ Q0 + Q7’ Q6 Q5’ Q4
Q3’ Q2 Q1 Q0 + Q8’ Q7 Q6’ Q5 Q3 Q2’ Q1’ Q0 + Q8’ Q7’ Q6 Q5 Q4’ Q3 Q2 Q0’
+ Q8’ Q7 Q6’ Q5 Q4’ Q3 Q2 Q0’ + Q8 Q7’ Q6 Q5’ Q4’ Q3’ Q1 Q0’ + Q8’ Q7 Q6
Q5’ Q3’ Q2 Q1 Q0’ + Q8’ Q7 Q6’ Q5’ Q3 Q2’ Q1’ Q0’ + Q8’ Q7’ Q6 Q5 Q4 Q3
Q2 Q1’ Q0 + Q8’ Q7 Q6 Q5 Q4’ Q3 Q2’ Q1’ Q0’ + Q8’ Q7’ Q6’ Q3’ Q2 Q1’  +
Q8 Q7’ Q6’ Q5’ Q4 Q3 Q2’  + Q8’ Q7’ Q5 Q4’ Q3’ Q2 Q1  + Q8’ Q7 Q5 Q4 Q3’
Q2 Q1’  + Q7’ Q6’ Q5’ Q4 Q3’ Q2 Q1’  + Q8’ Q7 Q6’ Q5’ Q4’ Q2 Q0 + Q8’ Q6’
Q5’ Q4 Q3’ Q2 Q0 + Q8 Q7’ Q6 Q5’ Q4’ Q2’ Q0 + Q8’ Q7 Q6 Q4’ Q2’ Q1 Q0 +
Q8 Q7’ Q6’ Q5 Q3 Q1’ Q0 + Q8 Q7’ Q5’ Q4’ Q3’ Q1’ Q0 + Q8’ Q6’ Q5’ Q4’ Q3’
Q1’ Q0 + Q7’ Q6’ Q5’ Q4 Q2 Q1’ Q0 + Q8’ Q7’ Q6’ Q5’ Q4 Q3 Q0’ + Q8’ Q7 Q6
Q4’ Q3’ Q2 Q0’ + Q8’ Q6’ Q5 Q4’ Q2 Q1 Q0’ + Q8’ Q6 Q5’ Q3 Q2’ Q1 Q0’ + Q8
Q7’ Q5’ Q4’ Q3 Q1’ Q0’ + Q8’ Q6 Q5’ Q4’ Q3’ Q1’ Q0’ + Q8’ Q7 Q6’ Q5’ Q4
Q3 Q2 Q1  + Q8’ Q7 Q6 Q5 Q4 Q3’ Q2 Q0 + Q8’ Q7 Q6 Q5’ Q4’ Q3 Q2’ Q0 + Q8
Q7’ Q6’ Q5’ Q4’ Q2 Q1 Q0 + Q8’ Q7 Q6’ Q4’ Q3 Q2 Q1 Q0 + Q8 Q7’ Q6’ Q4’ Q3
Q2 Q1 Q0 + Q8 Q7’ Q5’ Q4’ Q3 Q2 Q1 Q0 + Q8’ Q6 Q5’ Q4 Q3’ Q2’ Q1 Q0 + Q7’
Q6 Q5’ Q4’ Q3 Q2 Q1’ Q0 + Q8’ Q7 Q6 Q4 Q3 Q2’ Q1’ Q0 + Q8’ Q7’ Q5 Q4 Q3’
Q2’ Q1’ Q0 + Q8’ Q7’ Q6 Q4’ Q3’ Q2’ Q1’ Q0 + Q8 Q7’ Q6’ Q5’ Q4’ Q3 Q2 Q0’
+ Q8 Q7’ Q5’ Q4’ Q3’ Q2’ Q1 Q0’ + Q7’ Q6’ Q5 Q4 Q3 Q2 Q1’ Q0’ + Q8’ Q7’
Q6 Q5’ Q4’ Q2’ Q1’ Q0’ + Q7’ Q6’ Q5 Q4’ Q3 Q2’ Q1’ Q0’;

S2 = Q8’ Q7 Q5 Q4 Q2 Q0 + Q8’ Q7’ Q5 Q4 Q3 Q2’ Q1’  + Q8’ Q7’ Q5’ Q4’
Q3’Q2’ Q1’  + Q8’ Q7 Q6 Q5 Q4 Q1 Q0 + Q8’ Q6’ Q5’ Q4 Q2’ Q1 Q0 + Q8’ Q7
Q5’ Q4 Q2’ Q1’ Q0 + Q8 Q7’ Q6’ Q4 Q3’ Q2’ Q0’ + Q8’ Q7 Q6’ Q4’ Q3’ Q2’
Q0’ + Q8’ Q7 Q5 Q4’ Q3’ Q1 Q0’ + Q8’ Q7 Q5 Q4’ Q2’ Q1 Q0’ + Q8 Q7’ Q6’
Q5’ Q4’ Q3’ Q2 Q1’  + Q8 Q7’ Q6’ Q5 Q4’ Q3 Q2’ Q1’  + Q8 Q7’ Q6’ Q5 Q4’
Q3’ Q2 Q0 + Q8 Q7’ Q6 Q5’ Q4 Q3’ Q1 Q0 + Q7’ Q6’ Q5 Q4 Q3 Q2 Q1 Q0 + Q8’
Q7 Q5’ Q4’ Q3 Q2’ Q1 Q0 + Q8’ Q7 Q5’ Q4’ Q3 Q2 Q1’ Q0 + Q8’ Q7 Q6 Q5 Q3’
Q2 Q1’ Q0 + Q8’ Q7’ Q6 Q5’ Q4’ Q3’ Q2’ Q0’ + Q7’ Q6’ Q5’ Q4 Q3 Q2 Q1 Q0’
+ Q8’ Q6’ Q5 Q4’ Q3’ Q2’ Q1 Q0’ + Q8’ Q7’ Q6’ Q5 Q4 Q3 Q1’ Q0’ + Q8’ Q7’
Q5’ Q4 Q3’ Q2 Q1’ Q0’ + Q8’ Q7 Q6 Q4’ Q3 Q2’ Q1’ Q0’ + Q8’ Q7’ Q6 Q4’ Q3’
Q2’ Q1’ Q0’ + Q8’ Q7’ Q6 Q5 Q4 Q3’ Q2 Q1 Q0’ + Q8 Q7’ Q6 Q5’ Q4’ Q3 Q2’
Q1’ Q0’ + Q8’ Q7 Q6 Q4 Q3’ Q2 Q0 + Q7’ Q6’ Q5’ Q4 Q3 Q2’ Q0 + Q8’ Q7 Q6’
Q4 Q3 Q1 Q0 + Q8’ Q7 Q6’ Q5’ Q3 Q1’ Q0 + Q8’ Q6’ Q5’ Q4 Q3 Q1’ Q0 + Q8’
```

```
Q7’ Q5’ Q4’ Q3 Q2 Q0’ + Q8’ Q7 Q6’ Q3’ Q2 Q1’ Q0’ + Q8’ Q5 Q4 Q3 Q2’ Q1’
Q0’ + Q8’ Q7 Q6’ Q5’ Q4 Q3 Q2 Q1  + Q8’ Q7’ Q6 Q5 Q4’ Q3 Q2’ Q1  + Q8 Q7’
Q6’ Q5 Q4 Q3’ Q2 Q1’  + Q8’ Q7’ Q6’ Q5 Q4’ Q3’ Q2 Q1’  + Q8’ Q7 Q6’ Q5’
Q4’ Q3’ Q2 Q1’  + Q8 Q7’ Q6’ Q5’ Q4 Q3 Q2’ Q1’  + Q8’ Q7’ Q6 Q5’ Q4 Q3 Q1
Q0 + Q8’ Q6 Q5’ Q4 Q3 Q2 Q1 Q0 + Q8’ Q7’ Q6 Q4’ Q3 Q2 Q1 Q0 + Q8’ Q7 Q6’
Q5 Q3’ Q2 Q1 Q0 + Q8 Q7’ Q6’ Q5’ Q3 Q2’ Q1 Q0 + Q8 Q7’ Q6’ Q5 Q4’ Q2 Q1’
Q0 + Q8’ Q7’ Q6’ Q4 Q3’ Q2 Q1’ Q0 + Q8’ Q7’ Q6’ Q5’ Q4 Q3 Q1 Q0’ + Q8’
Q7’ Q6’ Q5’ Q4’ Q3 Q1’ Q0’ + Q8’ Q6 Q5 Q4’ Q3 Q2 Q1’ Q0’ + Q8’ Q7 Q6’ Q4
Q3 Q2’ Q1’ Q0’ + Q7’ Q6’ Q5 Q4 Q3’ Q2’ Q1’ Q0’;

S1 = Q8’ Q6’ Q5 Q4’ Q3’ Q2 Q1  + Q8 Q7’ Q6 Q5’ Q4’ Q3 Q1’  + Q8’ Q7 Q6 Q5
Q4’ Q3 Q0 + Q8 Q7’ Q6’ Q5’ Q4 Q2 Q0 + Q8’ Q7 Q5 Q4 Q3’ Q2 Q0 + Q7’ Q6’ Q5
Q4 Q3’ Q2’ Q0 + Q7’ Q6’ Q5’ Q4 Q3 Q1’ Q0 + Q8 Q7’ Q6’ Q5 Q4’ Q2’ Q0’ +
Q8’ Q7’ Q6 Q5 Q4 Q1 Q0’ + Q8’ Q7’ Q6’ Q5’ Q3’ Q1’ Q0’ + Q7’ Q6 Q5’ Q3’
Q2’ Q1’ Q0’ + Q8’ Q7 Q6’ Q5 Q4 Q3’ Q2 Q1’  + Q8’ Q7’ Q5 Q4’ Q3’ Q2 Q1 Q0
+ Q8 Q7’ Q5’ Q4’ Q3’ Q2’ Q1 Q0 + Q8’ Q7’ Q6 Q5 Q3 Q2 Q1 Q0’ + Q8 Q7’ Q6
Q5’ Q3’ Q2 Q1 Q0’ + Q8 Q7’ Q6’ Q5’ Q4 Q2’ Q1 Q0’ + Q8 Q7’ Q6’ Q5 Q4 Q3’
Q2 Q1 Q0’ + Q8’ Q6’ Q4’ Q3 Q2’ Q1  + Q8’ Q5’ Q4 Q2’ Q1’ Q0’ + Q8’ Q7’ Q5’
Q4 Q3 Q2 Q1  + Q8’ Q6 Q5’ Q4 Q3 Q2 Q1  + Q8’ Q7 Q6’ Q5 Q3’ Q2’ Q1  + Q8’
Q7 Q6 Q5’ Q3’ Q2’ Q1  + Q8’ Q7’ Q6 Q5’ Q4’ Q2 Q1’  + Q8’ Q7 Q6’ Q5’ Q3’
Q2’ Q1’  + Q8’ Q7 Q6’ Q5 Q4 Q2 Q0 + Q8 Q7’ Q6’ Q5’ Q3 Q2 Q0 + Q8’ Q7’ Q6
Q5’ Q3’ Q2’ Q0 + Q7’ Q6’ Q5 Q4’ Q3 Q1 Q0 + Q8’ Q6’ Q5’ Q4 Q2’ Q1 Q0 + Q8’
Q7’ Q5 Q3 Q2’ Q1 Q0 + Q8’ Q6’ Q5 Q3’ Q2’ Q1 Q0 + Q8’ Q7 Q5 Q4’ Q3 Q1’ Q0
+ Q8’ Q7 Q5’ Q4’ Q3’ Q1’ Q0 + Q8’ Q7 Q6’ Q4 Q2 Q1’ Q0 + Q7’ Q6’ Q5 Q3’ Q2
Q1’ Q0 + Q8’ Q6 Q5 Q4’ Q2’ Q1’ Q0 + Q8’ Q7’ Q6 Q4 Q3 Q2’ Q0’ + Q8’ Q7 Q6’
Q4 Q3 Q2’ Q0’ + Q8’ Q7 Q5’ Q4’ Q3 Q2’ Q0’ + Q8 Q7’ Q5’ Q4’ Q3 Q2’ Q0’ +
Q8’ Q7 Q5 Q4 Q3 Q1 Q0’ + Q8 Q7’ Q5’ Q4’ Q3 Q1 Q0’ + Q8’ Q7 Q5’ Q4 Q2 Q1
Q0’ + Q8’ Q5 Q4 Q3’ Q2’ Q1 Q0’ + Q8’ Q7’ Q6 Q5’ Q3 Q1’ Q0’ + Q8’ Q7 Q5’
Q4’ Q3 Q1’ Q0’ + Q8’ Q7’ Q6 Q4’ Q3’ Q1’ Q0’ + Q8 Q7’ Q6’ Q4’ Q3’ Q1’ Q0’
+ Q8’ Q7’ Q6’ Q3’ Q2’ Q1’ Q0’ + Q8’ Q7’ Q6 Q5’ Q4’ Q3 Q1 Q0 + Q8’ Q7 Q6’
Q5’ Q4’ Q2 Q1 Q0 + Q8’ Q7 Q6 Q5’ Q4 Q3 Q1’ Q0 + Q8’ Q7’ Q6 Q4 Q3’ Q2 Q1’
Q0 + Q7’ Q6 Q5’ Q4 Q3’ Q2 Q1’ Q0 + Q8’ Q7’ Q6’ Q5’ Q4’ Q2’ Q1’ Q0 + Q8’
Q7 Q6 Q5 Q4’ Q3’ Q2 Q0’ + Q8’ Q7’ Q6 Q5 Q4’ Q3’ Q2’ Q0’ + Q8’ Q7’ Q6’ Q5’
Q3 Q2 Q1 Q0’ + Q8’ Q7 Q6’ Q5’ Q3’ Q2 Q1 Q0’ + Q8’ Q7’ Q6’ Q5’ Q4’ Q2’ Q1
Q0’ + Q8 Q7’ Q6’ Q4 Q3 Q2 Q1’ Q0’ + Q8’ Q6’ Q5 Q4’ Q3 Q2 Q1’ Q0’ + Q7’
Q6’ Q5 Q4’ Q3 Q2 Q1’ Q0’ + Q7’ Q6’ Q5 Q4 Q3 Q2’ Q1’ Q0’;
```

14.2. Código de la FSM simulada con Arduino para el Prototipo 2

```
int outputValue = 0;
int estadoBoton = 0;
int contador = 0;
```

```
//Vectores para utilización de pines
int pines[8] = {9,8,7,6,5,4,3,2};
int numeros[8] = {0, 0, 0, 0, 0, 0, 0, 0};

void setup() {
  //Serial.begin(9600);
  //Se configura el pin 10 como entrada
  pinMode(10, INPUT);

  //Se configuran los pines como salida
  for (int thisPin = 0; thisPin < 8; thisPin++) {
    pinMode(pines[thisPin], OUTPUT);
  }
}

void loop() {
  /*for (int i = 0; i < 6; i++) {
    Serial.println(myStrings[i]);
    delay(500);
    */
/*
    Serial.println(mensaje2);
    delay(500);*/

    //Serial.println(outputValue);
    estadoBoton = digitalRead(10);

    if (estadoBoton== HIGH){
       delayMicroseconds(200);
          contador = contador+1;
    }

    if (contador >24){
      contador =1;
      }

    switch(contador){
      case 1:
        outputValue = 80;
        break;
      case 2:
        outputValue = 65;
        break;
      case 3:
        outputValue = 66;
```

```
  break;
case 4:
  outputValue = 76;
  break;
case 5:
  outputValue = 79;
  break;
case 6:
  outputValue = 32;
  break;
case 7:
  outputValue = 80;
  break;
case 8:
  outputValue = 65;
  break;
case 9:
  outputValue = 66;
  break;
case 10:
  outputValue = 76;
  break;
case 11:
  outputValue = 79;
  break;
case 12:
  outputValue = 32;
  break;
case 13:
  outputValue = 80;
  break;
case 14:
  outputValue = 65;
  break;
case 15:
  outputValue = 66;
  break;
case 16:
  outputValue = 76;
  break;
case 17:
  outputValue = 79;
  break;
case 18:
  outputValue = 32;
  break;
```

```
      case 19:
        outputValue = 80;
        break;
      case 20:
        outputValue = 65;
        break;
      case 21:
        outputValue = 66;
        break;
      case 22:
        outputValue = 76;
        break;
      case 23:
        outputValue = 79;
        break;
      case 24:
        outputValue = 10;
        break;

      }

    //Se mapea el valor de la salida a los pines
    numeros[0] = outputValue & B00000001;
    numeros[1] = outputValue & B00000010;
    numeros[2] = outputValue & B00000100;
    numeros[3] = outputValue & B00001000;
    numeros[4] = outputValue & B00010000;
    numeros[5] = outputValue & B00100000;
    numeros[6] = outputValue & B01000000;
    numeros[7] = outputValue & B10000000;

    for (int i = 0; i<8; i++){
      digitalWrite (pines[7-i],numeros[i]);
    }
}
```