UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un circuito integrado con tecnología de 180nm usando la librería de diseño de TSMC. Verificación avanzada de la síntesis lógica y simulación avanzada del deck final que incluye resistencias y capacitancias parásitas

Protocolo de trabajo de graduación presentado por Gerardo Enrique Cardoza Marroquín , estudiante de Ingeniería en Electrónica

Guatemala,

2021

Resumen

Este proyecto consta en la simulación y verificación de los archivos obtenido del diseño lógico y la extracción de parásitos de los componentes del circuito realizado. En la parte del diseño lógico se enfoca mas en lo que es la depuración y optimización en el flujo de diseño con tecnología de 180nm, en donde esto consta de un flujo de diseño para poder fabricar el chip. Esta parte se va a enfocar en optimizar para la prueba y simulación de archivos HDL y esquemáticos que se pueden llegar a generar las herramientas de synopsys, en esta parte del diseño lógico se utilizara tanto como VCS y Verdi. En esta parte se realizaran distintas pruebas, para que en la parte final, ya se pueda validar cada uno de los flujos de diseño.

En la parte de parásitos se va a encargar de la simulación en HSpice, que nos permite realizar una extracción exitosa para realizar el chip y lograr ver si cumple con las especificaciones. Se pretende mostrar lo que realmente hace el software, las librerías y los comandos que se vana realizar en este proyecto, para que las personas en un futuro puedan agarrar el flujo y realizar sus propios proyectos.

Antecedentes

Este proyecto se puedo realizar gracias al Ingeniero Carlos Esquit, que fue asignado en el 2009 como el director de carrera del departamento de Ingeniería de Electrónica y Mecatrónica. Esta persona fue que comenzó con esta iniciativa por el 2013.

Todo comenzó en el 2013 cuando comenzaron a dar el curso de VLSI en la Universidad del Valle de Guatemala, en el que paso a ser Nanoelectronica 1, en donde se aprendieron todos los conceptos necesarios para realizar esta parte del proyecto.

En el 2020 y el 2021 no se tuvo que parar en el proceso de la investigación sobre como realizar el flujo de diseño para realizar un chip a nanoescala. Con Splashtop se pudo ingresar a las computadoras de la Universidad, en las cuales se podría acceder al equipo que puede ser utilizado para el flujo de diseño y gracias a synopsys se pudieron instalar los software necesarios para la realización del flujo de diseño.

Este flujo de diseño ha ido evolucionando conforme a los cursos de Nanoelectronica 1 y 2, que dieron lugar en la reforma curricular realizada en el 2015, en la Universidad del Valle por Carlos Esquit. Con esto se pudo ir evaluando conceptos claves que nos servirían en un futuro, para la realización del flujo de diseño.

En el 2019 un grupo de estudiantes de la Universidad logro comenzar a realizar un proyecto de un chip a escala nanométrica, desde que se comenzó con esta iniciativa otro grupo de estudiantes en el 2020 comenzaron a realizar retoques en este proceso de poder realizar un chip a nanoescala. Los estudiantes que comenzaron con esta iniciativa fueron Luis Najera y Steven Rubio, estos individuos comenzaron a realizar un estructura de lo que es el diseño de flujo, el cual es un proceso que se divide en 2 categorías: En diseño lógico y el diseño físico. Mas que todo dejaron un esquema planteado para lograr elaborar ambas partes del diseño, dejando las herramientas necesarias para la producción del mismo que se tienen que tomar de synopsys, generaron guías de instalación y un uso poco ambiguo del uso de las herramientas de synopsys. Para el trabajo que se realizo en el 2020 fue un proceso de varios alumnos que realizaron varias etapas de diseño con el fin de dejar una plantilla casi terminada, para poder correr cualquier programa y que pasara el flujo de diseño sin ningún inconveniente, lastimosamente la pandemia que realizo el COVID-19, se pudo realizar solamente un circuito pequeño, pero sin comprobar cosas del diseño físico y sin el conocimiento completo de cada para del flujo del diseño, pero con una gran parte de la finalización del proyecto teniendo cada parte definida y que herramientas de software son las necesarias para realizar cualquier proceso del flujo de diseño.

[1] [2]

Justificación

La fabricación de un chip a nanoescala representa grandes avances para la Universidad de Guatemala y para el mismo país, ya que esta sera la primera vez que una Universidad de Guatemala pueda ser capaz del diseño de un chip con una tecnología de 180nm. Con el flujo de diseño se pretende diseñar y lograr mandar a fabricar chips que sean diseñados por medio de un programa de descriptor de hardware y que pueda ser utilizado para un proyecto de mayor escala que revolucione la universidad y que sea un incentivo para el desarrollo en Guatemala sobre la investigación en nanoelectrónica.

Este trabajo, es un parte muy importante en el flujo de diseño. Este trabajo es importante para la continuación de las partes posteriores, ya que se sintetizara por medio de las herramientas de synopsys y con la simulación y la comparación de los diseños digitales que serán el original y el sintetizado jugaran un importante papel en las otras fases de diseño, por la complejidad que pueda representar el circuito en las otras fases. A la hora de sintetizar un circuito se espera que se logre comportar de la manera deseada y este puede ayudar a la parte anterior para ver si se encuentra un error en la parte del diseño del mismo circuito.

En la otra fase de diseño enfocada en la parte de simulación de la extracción de parásitos, es una parte que depende de todas las fases anteriores, ya que este mirara las simulaciones, en donde mostrara las respuestas mas reales posibles por las herramientas de synopsys del circuito que se pretende realizar, por lo tanto esta etapa sera la que indicara el correcto funcionamiento del chip antes de ser enviado a fabricar. En si, lo que consta el trabajo es realizar las simulaciones de la primera etapa de diseño y la ultima de la extracción de parásitos, para poder corroborar que el chip que se mande a fabricar sea un chip, que no tenga problemas al probarlo de forma física.

Objetivos

Objetivo General

Realizar las comprobaciones necesarias de la síntesis lógica y la extracción de parásitos del modelado que se piensa implementar en el chip. Con el fin de poder comprobar la parte de la síntesis lógica, en donde se realizara simulaciones y verificaciones de rigor para el correcto funcionamiento del mismo y poder comprobar que el circuito no se vea afectado por los parásitos. Poder tener una comunicación eficiente con los demás grupos de trabajo y poder lograr como grupo solventar los errores de forma rapida.

Objetivos Específicos

- Tener una comunicaciones efectiva con los demás grupos de trabajo para la solución rapida de errores.
- Proporcionar el circuito, ya sintetizado para que no haya errores del diseño lógico y que pase al diseño físico.
- Poder realizar una verificación de los parásitos y poder emular el comportamiento de los parásitos dentro del circuito con gráficos.
- Lograr ver el correcto funcionamiento del circuito para mandarlo a fabricar.
- Brindar los archivos necesarios para que puedan pasar a otras fases de diseño.
- Verificar que los análisis de la extracción de parásitos sea correcta con las herramientas de synopsys.
- Lograr el entendimiento básico de las otras fases de diseño, para poder apoyar en las otras fases de diseño.

Marco teórico

Diseño VLSI

En este proceso se detalla el diseño y la fabricación de los CMOS, con el fin de poder fabricar un chip. Este proceso requiere de los transistores de canal n (nMOS) y canal p (pMOS). Estos sistemas basados en VLSI contienen grandes ventajas en el mundo de la electrónica, ya que permiten hacer chip con muchas funcionalidades a un tamaño muy pequeño, La velocidad también se considera un factor importante, ya que las capacitancias parásitas se logran reducir a un nivel considerable y por ultimo la potencia se logra reducir, en donde se puede ver beneficiado respecto al costo, alimentación mas pequeña, enfriamiento, etc..

Desde que William Schockley, John Bardeen y Walter Brattain logra inventar el transistor de puntas de contacto, comenzaron a revolucionar este mundo de los transistores y al proponer el transistor bipolar en 1948, es una de las grandes bases que se utiliza en la actualidad, para realizar el proceso de un chip a nanoescala. Luego de los años se descubrió la estructura de los MOSFET'S que llegaron a reemplazar a los JFET'S, esto fue gracias a Ian Munro Ross. La idea de los MOSFET'S era mas antigua pero el logro que se hiciera viable los efectos de campo. El VLSI se conoce como *Very Large Scale Integration*, en donde el numero de componentes es de 10,000 a 100,000 y el numero de compuertas es de 1000 a 10,000.

El ingeniero Gordon Moore, logro observar en 1965 que la cantidad de transistores iba aumentando según los años que pasaban por un factor de 2, es decir que por cada año la cantidad de transistores en un microprocesador aumentaba cada año. En el 2007, el mismo Moore produjo que su ley dejara de existir dentro de 10 o 15 años, dependiendo de lo que se realice en estos años. [1] [2]

Flujo de diseño

El flujo de diseño nos muestra los pasos requeridos para poder realizar una serie de pasos para el diseño y la fabricación de lo que es un circuito integrado. Este flujo de diseño se logra dividir en 2 partes (Front End y Back End). Lo que se busca con este flujo es encontrar la manera que a la hora de fabricar un chip, este no tenga errores y que sea funcional respecto a las necesidades requeridas.

Front End Este se encarga de resolver un problema pasándolo a un lenguaje descriptivo, en este caso se usa los lenguajes de Verilog y VHDL. En esta etapa se encuentran los pasos necesarios para la arquitectura de diseño. A continuación se logran ver los pasos necesarios para que se pueda elaborar el diseño front end.

Primero: Se debe realizar el circuito, por medio del lenguaje descriptivo de hardware como Verilog, el cual presenta la solución del problema establecido.

Segundo: Se denomina la síntesis lógica, este es un proceso en donde se toma el diseño RTL, que fue descrito en Verilog o VHDL. En esta también se producen comprobaciones y simulaciones, con el fin de comprobar el modelo RTL propuesto.

Tercero: El netlist, este es un circuito generado, después de la síntesis lógica en donde se logra observar por medio de librerías.

Cuarto: El Ultimo paso, se utilizan varios software's con el fin de verificar la funcionalidad de la síntesis del circuito.

[3]

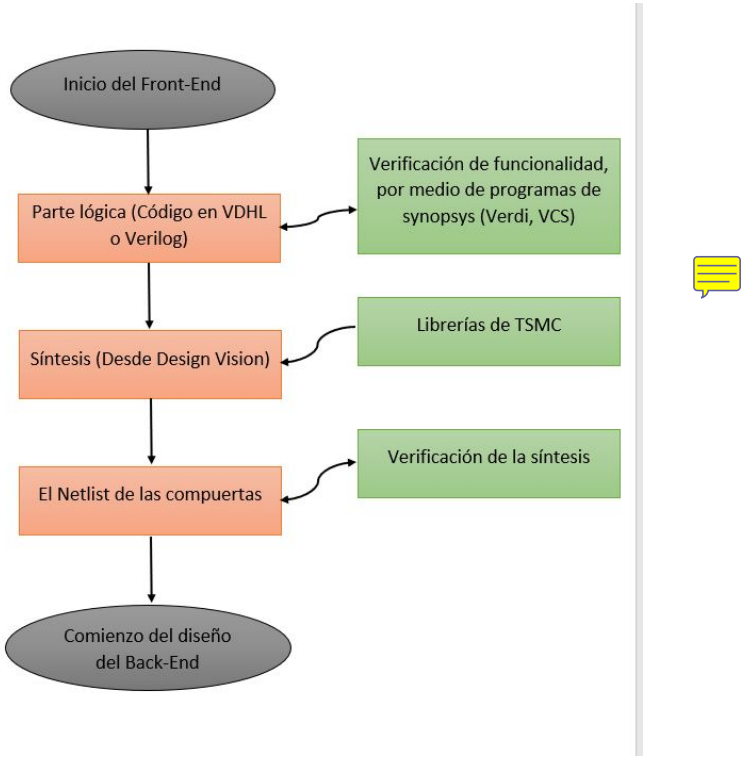

Figura 1: Diagrama del Front-End

Back End Esta parte va mas inclinada a la implementación del circuito en físico. Esta toma la parte del lenguaje descriptivo, en forma de un diseño en físico, el cual se convierte en un layout, en donde podemos ver a mas detalle como esta compuesto el circuito.

Primero: En este se encuentra la síntesis física, el cual se basa en el netlist. Al pasar a esta etapa se encuentra el layout, en donde tenemos mas información acerca del circuito y además podemos manipularlo, con el fin que cumpla las reglas de diseño especificadas por la empresa que va a realizar el chip.

Segundo: Obtenemos lo que es el Placement, en este caso nosotros decidimos donde se van a poner las celdas con el fin de que llegue a ser lo mas óptimo posible y sencillo de arreglar en un futuro.

Tercero: Al tener las celdas puestas en la mejor posición posible se procede a interconectar las salidas y las entradas de cada parte del circuito descrito, este parte se conoce como routing.

Cuarto: Este se denomina *Design Rule Check*, este consiste en verificar que el layout no

contenga errores o reglas de diseño que son impuestas dependiendo de la tecnología.

Quinto: Este se denomina *layout versus schematic*, este consiste en verificar la integridad del diseño. Este compara el layout con el netlist (Síntesis lógica).

Sexto: La extracción de parásitos, aquí es donde se simula la parte del layout con la cantidad de parásitos que pueda tener el layout, y se logra ver que tanto llega a afectar al final del proceso.

[3]

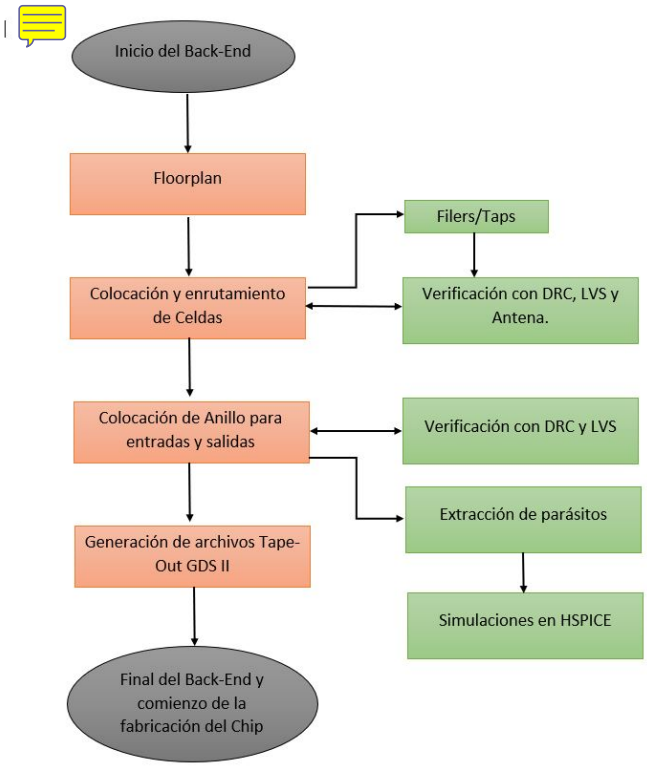

Figura 2: Diagrama del Back-End

Lenguaje Descriptor de Hardware

Estos lenguajes son los que son utilizados con el fin de describir una arquitectura y lograr visualizar el comportamiento de un sistema electrónico. La forma mas fácil de representar un circuito es mediante la utilización de diagramas o esquemas estos hacen una representación gráfica de lo que se pretende realizar. Con el tiempo comenzaron a salir circuitos mas complejos que ya no podían ser representadas de una manera visual, por lo que nació una idea de integrar las herramientas de descripción de síntesis, simulación y realización. Al principio se logro utilizar un lenguaje de descripción que solo permitía secuencias simples, a este se le nombro Netlist. Luego de estos programas simples que ya eran reconocidos como lenguajes de descripción de hardware, se logro ver el gran impacto que podían tener a la hora de describir circuitos complejos.

Mientras que todo iba evolucionando, la posibilidad de desarrollar circuitos digitales mediante dispositivos programables era la forma mas viable, ya que los circuitos, ya representaban una cantidad considerable de componentes. Este tipo de lenguajes estaban orientados para la realización de simulaciones, por lo que no importaba mucho que el nivel de abstracción fuera tan alto. Al momento de la aparición de técnicas para la síntesis de circuitos a partir de lenguajes que se consideraban de alto nivel de abstracción, se comenzaron a utilizar para la síntesis de circuitos. [4]

Verilog y VHDL

Verilog es un lenguaje que se utiliza para describir hardware, en el cual podemos diseñar y simular circuitos electrónicos (Digitales). Este lenguaje es una derivación de la programación en C, con el fin que los ingenieros puedan entenderlo. VHDL también se encuentra englobado dentro de los lenguajes de descripción de Hardware. En si, un HDL es un tipo de lenguaje especializado que define la estructura, diseño, operación de circuitos electrónicos y circuitos digitales. Este tipo de lenguaje es una forma de representación formal de un circuito electrónico, con la posibilidad de hacer un proceso mas simple.

EL lenguaje VHDL se considera un estándar, este no depende de ningún fabricante o dispositivo, es independiente. El fin de este lenguaje es que se puedan reutilizar los diseños y tiene la ventaja de ser un diseño jerárquico. Verilog este puede llegar a soportar lo que son circuitos analógicos, pruebas y señales mixtas a distintos niveles de abstracción. [4]

VCS

Es una de las soluciones de la verificación funcional que se utiliza en la mayoría de las empresas de semiconductores en el mundo. Este es un programa que proporciona características innovadoras para lograr un mayor rendimiento y permite los flujos de verificación de desplazamiento al principio del ciclo de diseño. Este ofrece como solución con Native Testbench, compatibilidad con Verilog,a análisis y cobertura e integración con Verdi. VCS satisface las necesidades de los diseñadores para poder verificar los resultados obtenidos de la prueba de un circuito elaborado en un programa compatible de descripción de hardware.

Esta herramienta cuenta con un flujo de dos pasos para evaluar el funcionamiento del circuito descrito. [5]

Flujos de VCS

Este es un flujo que se utiliza para poder comprobar si el diseño en Verilog HDL y SystemVerilog, son funcionales para la fabricación del Chip.

Primero: El primer paso es lograr compilar el diseño, en esta etapa VCS logra construir una instancia de jerarquía y logra generar un archivo .simv que se utiliza para poder realizar la simulación.

Segundo:En esta etapa se simula el diseño con el archivo generado en la primera parte (.simv). En esta etapa se logra ver si se logro correr la simulación.

Existe otro flujo de dos pasos, este permite diseños en Verilog, VHDL y mixedHDL. Este consta de 3 pasos.

Primero: Analizar el diseño, en esta etapa VCS logra dar los ejecutables, los cuales son vhdlan y clogan, con el fin de analizar el diseño y lograr almacenar archivos en la carpeta de trabajo.

Segundo: Elaborar el diseño, en esta parte se adquiere un archivo .vcs, el cual se compila y logra elaborar el diseño utilizando los archivos generados en la librería de trabajo. Luego da un archivo .simv.

Tercero: Simular el diseño, este se logra simular con el archivo, el cual es un ejecutable binario que se describió en el paso anterior. [6]

Verdi

Esta herramienta de synopsys se considera un sistema de depuración automatizado y esta herramienta permite la depuración integral de todos los flujos de diseño y verificación. Este sistema tiene una tecnología poderosa que le ayuda a comprender el comportamiento del diseño definido, y lo mas relevante es que ayuda a optimizar los procesos difíciles y tediosos.

En esta herramienta se reducir el tiempo de depuración en mas del 50 por ciento, esto es posible porque automatiza el comportamiento, por medio de un seguimiento con una tecnología única de análisis. También logra extraer, aislar y mostrar la lógica pertinente en los diseños. Logra revelar el funcionamiento y la interacción con el diseño propuesto. El sistema de depuración por completo que tiene verdi, utiliza la tecnología y las capacidades de un sistema de depuración, cabe resaltar que este utiliza funciones avanzadas de depuración con soporte para una gran cantidad de lenguajes y metodologías. [7] **Funciones principales**

- Seguimiento rápido de la actividad de muchos ciclos de reloj con una potente tecnología de análisis.
- Vistas de flujo temporal que proporciona una visualización de tiempo y estructura para comprender las relaciones de causa y efecto.

- Analizar diseños en niveles mas altos de abstracción.
- Depuración basada en aserciones con soporte integrado.

Depuración de SystemVerilog Testbench

- Soporte de código fuente para SystemVerilog Testbench, incluyendo la metodología de verificación universal (UVM).
- Vistas especializadas que ayudan a comprender el código, viendo la navegación y exploración de jerarquías basadas en declaraciones.
- Capacidad de registro de transacciones automatizadas, que brindan una imagen completa de la actividad del banco de pruebas en el entorno de verificación después de la simulación.
- Control de simulación interactivo que permite correr el código completo.
- Las vistas de depuración compatibles con UVM permite explorar los resultados de aspectos específicos.
- Vistas de depuración a nivel de transacción y logran admitir el registro de datos.

[6]

Mas acerca de Verdi

Este sistema de depuración automatizado tiene compiladores e interfaces. Los compiladores que son utilizados en la parte de diseño mayoritariamente son Verilog, VHDL Y System Verilog. La interfaz que tiene verdi implementa los estándares industriales de datos VCD y SDF. Normalmente los resultados por la herramienta son guardados en la Fast Signal Database. Verdi también permite la interoperabilidad con lo que se conocen los simuladores lógicos, las herramientas de verificación y los análisis de tiempo.

Las bases de datos que verdi maneja son las de conocimiento que se pueden llamar KDB y la base de datos de señal rapida conocida como FSDB. En KDB normalmente se utiliza para almacenar información lógica y funcional para el diseño implementado. La base de datos FSDB, logra almacenar los resultados de la simulación.

Al utilizar la informacion que proporcionan estas bases de datos, verdi tiene unas herramientas de análisis que resultan ser muy útiles dependiendo de la aplicación que se quiera dar, estas son: Análisis de estructura, comportamiento, evaluación y Mensajes.

Extracción de parásitos

La extracción de parásitos se puede realizar después de haber hecho las pruebas pertinentes en el flujo del diseño, este pertenece a la parte del Back-End y se puede realizar correctamente después de realizar el lvs y el drc, que son parte del diseño del layout después con HSPICE se busca el archivo generado con parásitos para ver como se comporta la

simulación, en donde también se utiliza el starRc. Todo esto es con el fin de garantizar que el diseño VLSI es factible y que las capacitancias parásitas no va a afectar el circuito.

Esto sucede al jugar con los transistores que generan lo que se llaman capacitancias parásitas que generan un efecto adicional en los conductores que funcionan como placas entre el dieléctrico, y este tipo de parásitos va a ir aumentando conforme la frecuencias, mas que todo en este caso al meter muchos transistores las capacitancias parásitas van a ir aumentando. En la implementación de circuitos nanométricas las interconexiones entre cables pueden llegar a representar problemas por los parásitos que se pueden generar, al igual que puede pasar con las difusiones de los componentes que pueda proveer las librerías de TSMC.

El fin de la extracción de parásitos es la representación de los efectos electromagnéticos que los cables y las difusiones puedan generar al momento de correr la simulación, en componentes como la capacitancia, resistencia e inductancia. [3]

StarRC

Esta herramienta se considera la solución para la extracción de parásitos. Esta herramienta amplia los beneficios de rendimiento , en donde se basa en mejorar la arquitectura y hace que el proceso sea mas efectivo. Este programa logra eliminar la necesidad de lo que es la escritura parásita en un netlist y este mismo logra ahorrar el espacio del disco. Mas que todo se especializa en optimizar el proceso reduciendo las capacitancias parásitas que puedan llegar a afectar el comportamiento deseado de un circuito.

[8]

HSPICE

Hspice es un herramienta de synopsys que permite realizar las simulaciones de un programa generado (.spf), en donde viene de la mano con Custom Wave View. El fin de este programa es ver una simulación y lograr ver el comportamiento del circuito. Este se considera uno de los simuladores mas potentes, ya que muestra gran precisión en los datos y la mayoría de empresas de semiconductores dependen de esta herramienta. Además se pueden realizar análisis de los circuitos eléctricos en estado estacionario, en el dominio de frecuencia y el transitorio.

[3]

Metodología

Para la metodología de la parte del diseño lógico, este consta del diseño de varios flujos utilizando la herramienta de VCS con el fin de testear y luego ser validados por el mismo programa.

Análisis

VCS es una herramienta que proporciona un parte importante en el flujo de diseño del proyecto, en donde para poder realizar una simulación exitosa se consulto con la documentación de la herramienta proporcionada por synopsys. La documentación de este viene en un parte de synopsys llamada SolvNet, la cual presenta una documentación extensa, en donde se puede aprender sobre la herramienta y detallada, en la cual Cual también aparecen los programas como: Verdi, StarRc y HSpice, estas serán necesarias al realizar las simulaciones y las verificaciones necesarias, en donde SolvNet, sera de gran ayuda para poder realizar las simulaciones y verificaciones necesarias. Estas herramientas se van actualizando cada año, en donde las mismas presentan mejoras a su sistema para poder evaluar con mayor detenimiento las simulaciones, en donde cada vez van a ver mas aspectos a tomar en cuenta.

Pruebas

Las pruebas van a ser realizadas por medio de los programas anteriormente mencionados, en donde cada uno de estos va a jugar un papel importante y con los resultados de las pruebas, se documentara para poder visualizar los cambios antes de la simulación con los resultados obtenidos después, para lograr ver una mejora y poder interpretar los datos de una mejor manera, en donde no solo actualizaremos los conocimientos de las herramientas, sino que también para entender mejor el funcionamiento de las mismas. Se buscara visualizar como funcionaria el circuito, ya con la implementación del chip en físico, para no enviar a fabricar un chip que realmente no funcione.

Cronograma de actividades

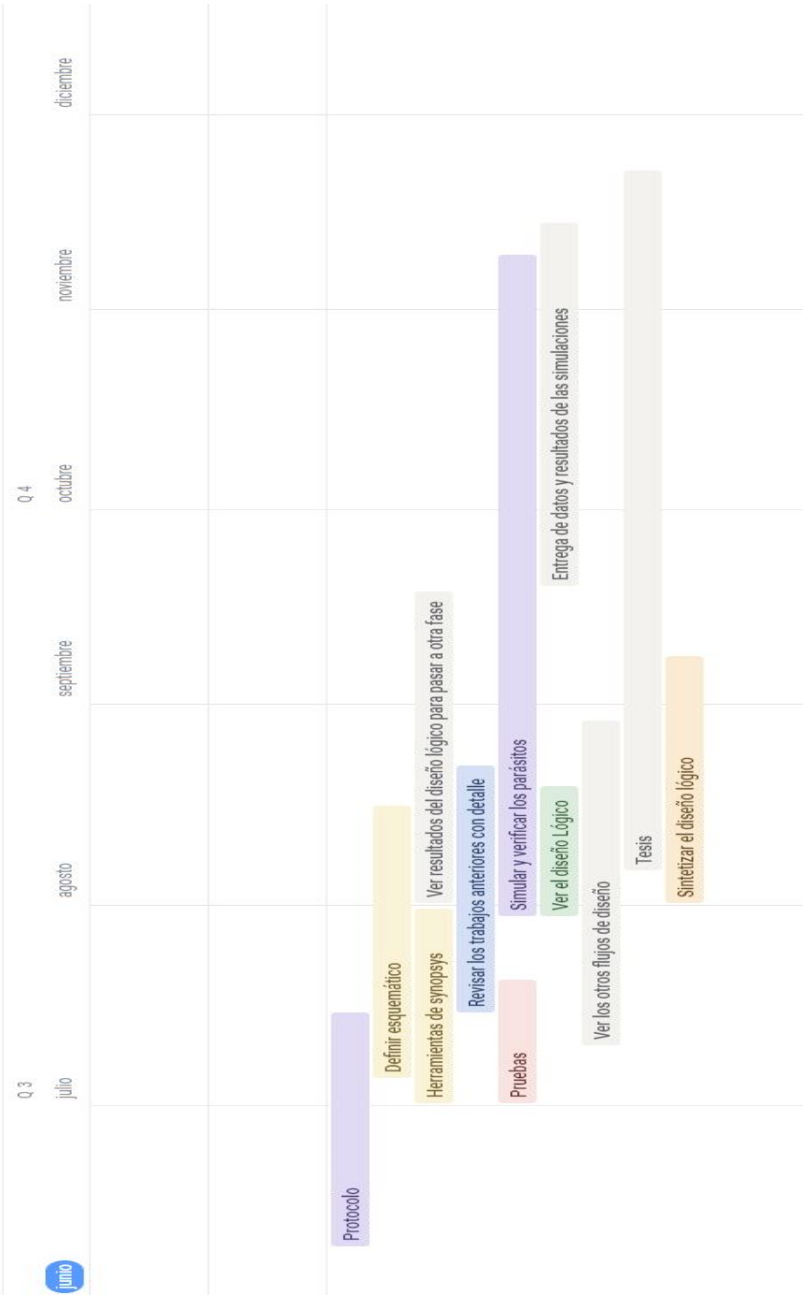

Figura 3: Cronograma de actividades

Índice preliminar

Índice general

0.1. Prefacio

0.2. Lista de figuras

0.3. Resumen

0.4. Abstract

0.5. Introducción

0.6. Antecedentes

0.7. Justificación

0.8. Objetivos

0.8.1. Objetivo General

0.8.2. Objetivos específicos

0.9. Alcance

0.10. Marco teórico

0.10.1. Diseño de VLSI

Flujo de diseño

Front-End Back-End

Lenguaje Descriptor de Hardware

0.10.2. Verilog y VHDL

0.10.3. VCS

Flujos de VCS

0.10.4. Verdi

Funciones principales

Depuración de SystemVerilog Testbench

Mas acerca de Verdi

0.10.5. Extracción de parásitos

0.10.6. StarRC

0.10.7. HSPICE

0.11. Metodología

0.12. Herramientas utilizadas

0.13. Análisis de Resultados

0.14. Verificación de resultados

0.15. Proceso para la verificación y para las simulaciones

0.16. Conclusiones

0.17. Recomendaciones

0.18. Bibliografía

0.19. Anexos

Referencias

[1] J. F. M. Lenddech, "Circuitos integrados de pequeña, mediana y gran escala," en *Licenciatura en ingeniería en computación*, Universidad Autónoma del Estado de México, N/A, págs. 1-32.

[2] S. Kang e Y. Leblebici, "CMOS FABRICATION TECHNOLOGY AND DESIGN RULES," *Chapter 2 (Fabrication of MOSFETs) of the book CMOS Digital Integrated Circuit Design*, 2003.

[3] C. C. Girón, "Ejecución y utilización de un flujo de diseño para el desarrollo de un chip con tecnología nanométrica: Extracción de componentes parásitos y simulaciones en HSPICE," *Facultad de Ingeniería Electrónica*, 2020.

[4] F. torres del Valle, "LENGUAJES DE DESCRIPCIÓN DE HARDWARE," *xdoc.mx*, págs. 1-8, https://xdoc.mx/preview/lenguajes-de-descripcion-de-hardware-5c2d1aa29da5a.

[5] Synopsys, "VCS," *https://www.synopsys.com/verification/simulation/vcs.html*, N/A, 2021.

[6] J. R. Orellana, "Definición del flujo en la herramienta VCS para la simulación de HDLs en la Fabricación de un Chip con Tecnología Nanométrica CMOS," *Facultad de Ingeniería Electrónica*, 2020.

[7] Synopsys, "Verdi," *https://www.synopsys.com/verification/debug/verdi.html*, N/A, 2021.

[8] ——, "StarRC," *https://www.synopsys.com/implementation-and-signoff/signoff/starrc.html*, N/A, 2021.