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 de la simulación y verificación de archivos obtenidos del diseño lógico y la extracción de parásitos de los componentes de un circuito. La fase de simulación del diseño lógico se enfoca en la depuración y optimización. Esto se realiza con una tecnología de 180nm, la misma se llevará a cabo usando las librerías de TSMC, con el fin de realizar un chip a escala nanométrica. Esta parte busca concentrar su atención en optimizar el circuito descrito en Verilog, en simular archivos HDL esquemáticos, que pueden llegar a generar las herramientas de Synopsys, y para corroborar la síntesis lógica se utilizará VCS y Verdi. La parte de la simulación de parásitos se realizará en HSPICE, esto permite encontrar la cantidad de parásitos que están dentro del circuito y comprobar si este cumple con los requisitos de funcionalidad, para luego proceder a la fabricación el chip. Con este proceso se pretende comprender lo que HSPICE ofrece, las librerías y los comandos que se utilizan en este proyecto, para que las personas en un futuro puedan utilizar el flujo de diseño y realizar sus propios proyectos desde un lenguaje descriptor de hardware. Resumiendo, el trabajo muestra las simulaciones y verificaciones para garantizar que la síntesis lógica se realice de la forma correcta y poder obtener resultados congruentes con lo planteado en Verilog. Además, se busca que sea evidente la extracción de parásitos, para que al ejecutar la simulación en HSPICE se corrobore si el circuito funciona y se conozcan las características finales del circuito (Potencia, Frecuencia, Análisis de tiempos, etc..).

Antecedentes

El siguiente proyecto se llevó a cabo gracias al Ing.Carlos Esquit, quien en 2009 fue nombrado director de carrera del departamento de Ingeniería Electrónica y Mecatrónica en la Universidad Del Valle de Guatemala. En 2013 Impartió el curso VLSI posteriormente nanoelectrónica 1 y gracias a este se aprendieron los conceptos básicos para comenzar a realizar este proyecto.

El diseño ha ido evolucionando conforme a los cursos de nanoelectrónica 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 completar el flujo de diseño.

En el 2019 un grupo de estudiantes de la Universidad inicio un proyecto de un chip a escala nanométrica, y posteriormente otro grupo de estudiantes en el 2020 realizaron mejoras sobre el mismo, conservando siempre la nano escala.

Los pioneros anteriormente mencionados son los Luis Nájera y Steven Rubio, quienes iniciaron con una estructura del flujo de diseño, La cual consta de proceso que se divide en 2 categorías: En diseño lógico y el diseño físico. Este esquema planteado, tenía el fin elaborar ambas partes, dejando los instrumentos necesarios para la ejecución del flujo utilizando siempre herramientas de Synopsys, ya que sobre esta se plantearon las guías de instalación. Aunque al final el uso de las herramientas resultó ambiguo.

Durante los años 2020 y 2021 se continuo con el proceso de investigación sobre cómo realizar el flujo de diseño para realizar un chip a nano escala. Luego fue gracias a Synopsys el poder instalar el software Splashtop, el cual hizo posible el ingreso a las computadoras de la Universidad y de manera remota poder completar el diseño.

En el año 2020 varios alumnos fueron encargados de las distintas etapas de diseño, pues esto tenía como fin dejar una plantilla casi terminada, para poder correr cualquier programa y que el flujo de diseño pasara sin ningún inconveniente, lastimosamente por el COVID 19 (2020) se completó solamente un circuito pequeño, y no fue comprobado en su totalidad el diseño físico. Incluso no se tenía el conocimiento completo de cada parte del flujo del diseño. Sin embargo, se llegó a tener una gran parte del proyecto completada, y fue posible definir que herramientas de software son las necesarias para realizar cualquier proceso del flujo de diseño.

[1] [2] [3]

Justificación

La fabricación de un chip a nano escala representa un gran avance para la Universidad del Valle de Guatemala y para el país mismo, ya que esta será la primera vez que una Universidad de Guatemala pueda ser capaz de realizar el diseño de un chip con una tecnología de 180nm. Con el flujo de diseño se pretende diseñar y posteriormente enviar a fabricar chips que sean diseñados por medio de un programa de descriptor de hardware, además que puedan ser utilizados en proyectos de mayor escala. Esto no solo representa una revolución en la universidad, sino que incluso representa un incentivo para el desarrollo en Guatemala sobre la investigación en nanoelectrónica. Este trabajo, es un parte importante dentro el flujo de diseño, ya que es base para partes posteriores. Esto porque plantea que se sintetizará 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. Estos jugarán un papel importante pues dentro del esquema ayudan a resolver la complejidad que pueda representar el circuito en las otras fases. Teniendo en cuenta esto se espera que, con el esquema claro a la hora de sintetizar un circuito este logre comportar de la manera deseada y de ser así este pueda ayudar a la parte del diseño lógico para ver si se encuentra un error en el circuito. En fase de diseño enfocada a la simulación en el 'deck' generado de la extracción de parásitos, es una parte que depende de todas las fases anteriores, ya que esta enseñará las simulaciones, en donde podremos ver las respuestas más reales gracias a las herramientas de Synopsys del circuito que se pretende realizar, por lo tanto, esta etapa será la que indique el correcto funcionamiento del chip antes de ser enviado a fabricar.

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 obtener el correcto funcionamiento del circuito propuesto y poder obtener la caracterización del diseño en silicio del proyecto completo.

Objetivos Específicos

- Establecer una forma de validar los archivos generados en la síntesis lógica.
- Realizar una verificación de los parásitos, con el fin de ver si llega a afectar el circuito.
- Simular el circuito tomando en cuenta los parásitos y observar que cumpla con los requisitos de fabricación mostrando la caracterización final del chip.
- Obtener la extracción de parásitos para corroborar el correcto funcionamiento del circuito.
- Verificar que los análisis de la extracción de parásitos sean correctos con las herramientas de Synopsys.
- Brindar archivos y la información necesaria, con el fin de automatizar el proceso para las partes posteriores de la verificación de la síntesis lógica.
- Tener una comunicación efectiva con los demás grupos de trabajo para la solución rápida de errores.

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 un 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 último la potencia se logra reducir, y se puede ver respecto al costo un beneficio en la alimentación más pequeña, enfriamiento, etc. Desde que William Schockley, John Bardeen y Walter Brattain logran inventar el transistor de puntas de contacto, comenzaron a revolucionar este mundo de los transistores y al proponer el transistor bipolar en 1948, se convierte en una de las bases que se utiliza en la actualidad, para realizar el proceso de un chip a nano escala. 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 más antigua, pero él logró que se hiciera viable los efectos de campo. El VLSI se conoce como *Very Large Scale Integration*, en donde el número de componentes es de 10,000 a 100,000 y el número de compuertas es de 1000 a 10,000. El ingeniero Gordon Moore, logró 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 el esquema requerido para poder realizar una serie de pasos para el diseño y la fabricación 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 de fabricar un chip 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 Ultimó paso, se utilizan varios softwares con el fin de verificar la funcionalidad de la síntesis del circuito.

[4]

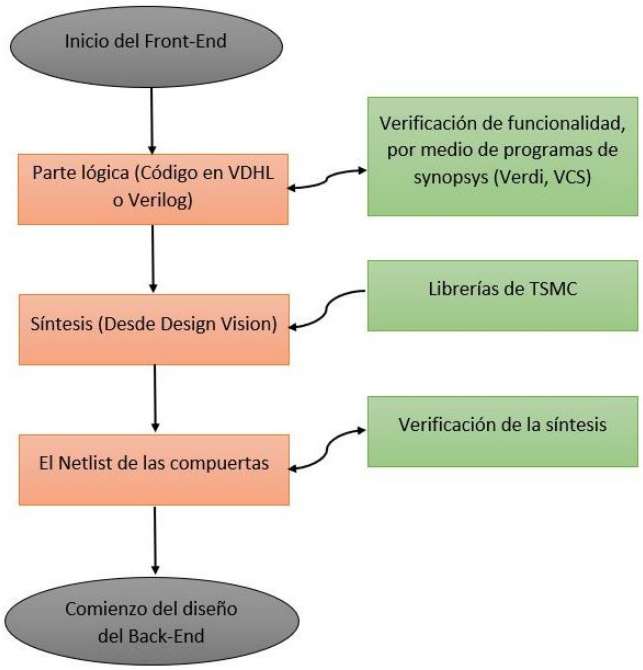

Figura 1: Diagrama del Front-End

Back End Esta parte va más 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 más detalle como está 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 más 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 más 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.

[4]

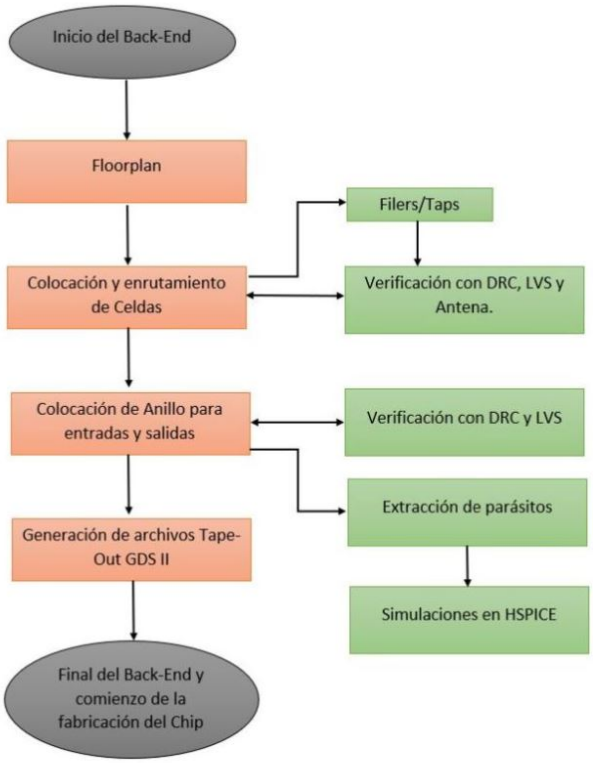

Figura 2: Diagrama del Back-End

Lenguaje Descriptor de Hardware

Estos lenguajes son utilizados con el fin de describir una arquitectura y lograr visualizar el comportamiento de un sistema electrónico. La forma más fácil de representar un circuito es mediante la utilización de diagramas o esquemas estos hacen una evidencia gráfica de lo que se pretende realizar. Ya que con el paso del tiempo comenzaron a salir circuitos más complejos que no pueden ser representados de una manera visual, nació la idea de integrar las herramientas de descripción de síntesis, simulación y realización. Al principio se logró 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, lograron 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 se hacía más 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 también para la síntesis de circuitos.

[5]

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 sí, 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 más 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.

[5]

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, 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.

[6]

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 logró 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.

[7]

Verdi

Esta herramienta de Synopsys se considera un sistema de depuración automatizado y 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 más relevante es que ayuda a optimizar los procesos difíciles y tediosos. En esta herramienta pretende reducir el tiempo de depuración en más 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.[8]

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 más 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.

[7]

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 rápida 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 información 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, después con HSPICE se busca el archivo generado con parásitos para ver cómo 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 van 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 las frecuencias, más 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. [4]

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 más 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. Más que todo se especializa en optimizar el proceso reduciendo las capacitancias parásitas que puedan llegar a afectar el comportamiento deseado de un circuito.

[9]

HSPICE

Hspice es una 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 más potentes, ya que muestra gran precisión en los datos y la mayoría de las 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.

[4]

Trabajo realizado

En los años anteriores se realizaron trabajos de investigación para poder utilizar y lograr entender los softwares como Verdi, Formality, VCS y HSPICE.

Se realizaron diferentes tipos de flujos según la complejidad requerida, en donde se lograron diseñar distintos flujos en la herramienta VCS, con el objetivo de verificar el comportamiento del circuito sintetizado. En este se realizan 2 simulaciones una para el circuito original y uno para el circuito sintetizado que ya hace referencia a las librerías de TSMC de 180 nm. Entonces el trabajo realizado por Jefferson Ruano, ya nos dice las librerías utilizadas para la simulación. En el trabajo del año 2020, también se logra ver el flujo de simulación con detección de condiciones de carrera, las cuales puede mostrar inconsistencias en las simulaciones ejecutadas. También logra mostrar el flujo para la evaluación de desempeño, en donde VCS cuenta con una herramienta conocida como generación de perfiles de simulación unificado, esta herramienta nos permite ver la cantidad de tiempo de CPU y memoria usada por nuestro diseño.

Con VCS en el trabajo realizado en el 2019 se realizo un flujo de simulación básico, el cual se utilizo para la simulación de Verdi del circuito original y el sintetizado. En este flujo básico se utilizaron ciertos comandos para llegar a realizar esto:

vcs-Muddate -RPP -v /yourPath/chip.v/yourPath/testBench.v -o demo -full 64 -debug_-all

vcs -V -R /yourPath/saed90nm.v /yourPath/chip_syn.v /yourPath/testBench_s.v -o yourDesignName -full64 -debug

./yourDesignName -gui [7] [3]

Luego con el tiempo se fue desarrollando métodos mas complejos para el desarrollo de circuitos mas complejos, en donde ya se realizaron flujos con detecciones de carrera, en esta se utilizaron detecciones de carrera dinámica y detecciones de carrera estática.

Para la dinámica se llega a generar lo que es un archivo *race.out* y *race.unique.out*, en donde una contiene una linea para las condiciones encontradas en la simulación y el otro contiene las lineas para las condiciones de carrera que son únicas respectivamente. Se mostrará a continuación el flujo propuesto por Jefferson con detección de carrera dinámica:

VCS_-HOME=/usr/synopsys/vcs-mx

export VCS_HOME

PATH=$VCSHOME/bin$:PATH

export PATH

PATH=usr/synopsys/verdi/bin:$PATH

export PATH

% vcs -V -R tcb018gbwp7t.v tpd018nv.v yourDesign.v yourTestbench.v -o racesim -race -full64 -debug_all ./racesim -gui

Para la herramienta de detección estática, se utilizan los *loops*, ya que estos son considerados como condiciones de carrera y esto suele ocurrir cuando eventos de control se encuentran después de un *always*. En esta parte del flujo se llega a proponer dos pruebas del diseño evaluado, una para evaluar los *loops* y otra para evaluar las condiciones de carrera que puedan existir en el reloj y los datos. El flujo que fue propuesto es el siguiente:

VCS_HOME=/usr/synopsys/vcs-mx

export VCS_HOME

PATH=$VCS HOME/bin:$PATH

export PATH

PATH=usr/synopsys/verdi/bin:$PATH

export PATH

% vcs yorDesign.v yourLibrary.v -hsopt=racedetect

% ./simv

% cat hsRaceInfo.db

Como ultima parte se habla del flujo para la evaluación de desempeño. VCS cuenta con una función que se le conoce como generador de perfiles de simulación unificado y con esta herramienta podemos ver la caracterización del circuito. Al utilizar esta herramienta requiere del siguiente flujo: VCS_HOME=/usr/synopsys/vcs-mx

export VCS_HOME

PATH=$VCS HOME/bin:$PATH

export PATH

PATH=usr/synopsys/verdi/bin:$PATH

export PATH

% vcs tcb018gbwp7t.v tpd018nv.v yourDesign.v yourTestbench.v -simprofile=time ./simv -simprofile mem (or time)

profrpt simprofile_dir -view time_summary -format text -outout NOTperformance -filter 0.0001

El trabajo realizado con Formality, se logró aprender que es una alternativa a la validación en las herramientas de VCS y Verdi. Este llega a generar un reporte más practico, ya que logra generar un reporte más simple con las coincidencias e inconsistencias al ver los circuitos.

En el trabajo realizado por Charlie Ayenci, en donde logro simular el diseño con parásitos por medio HSPICE, hubo ciertos errores al correr la simulación en donde, la empresa TSMC solamente logra proveer un kit académico, por lo que solo se encontraran las *black boxes*, que solo contienen el diseño de silicio del componente y con esto el netlist no logra poseer

los puertos de entrada, salida y alimentación.

[3] [4] [7]

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 consultó 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 también aparecen los programas como: Verdi, StarRc y HSpice, estas serán necesarias al realizar las simulaciones y las verificaciones necesarias, en donde SolvNet, será 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 más aspectos a tomar en cuenta.

Pruebas

Las pruebas van a ser realizadas por medio de los programas anteriormente mencionados, se documentara las pruebas y los cambios que se realizaron en el transcurso del proyecto con el fin de visualizar los cambios antes de la simulación con los resultados obtenidos después, para lograr ver una mejora y poder interpretar los datos, en donde no solo actualizaremos los conocimientos de las herramientas, sino que también para entender mejor el funcionamiento de las mismas. Se buscará visualizar el correcto funcionamiento del circuito, ya con la implementación del chip en físico, para no enviar a fabricar un chip que realmente no funcione.

Cronograma de actividades

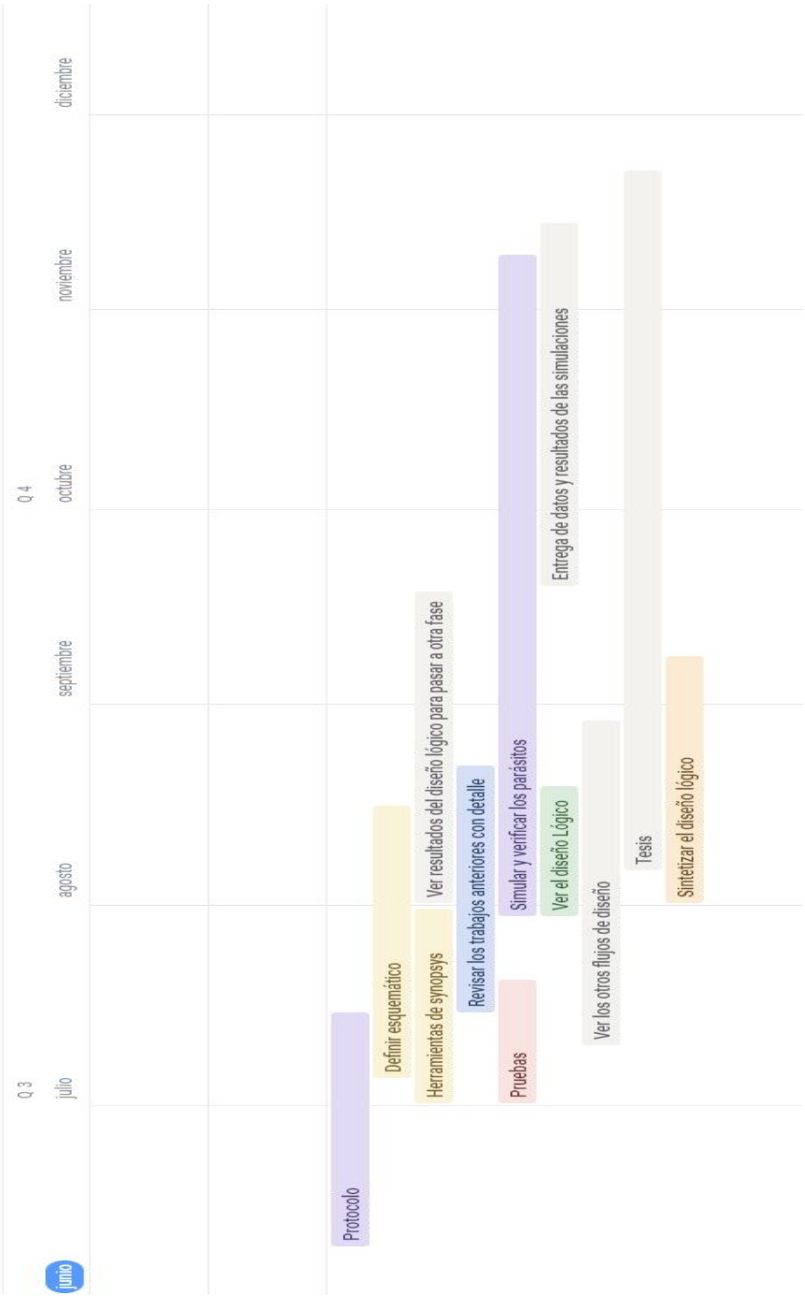

Figura 3: Cronograma de actividades

Índice preliminar

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] S. R. Vasquez, "Definición del flujo de diseño para fabricacion de un chip con tecnologia VLSI CMOS," *Facultad de Ingeniería Electrónica*, 2020.

[4] 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.

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

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

[7] 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.

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

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