UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys.

Trabajo de graduación presentado por:
Jonathan Alberto de los Santos Chonay
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,
2014

Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys.

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño de un sumador/restador completo de 32 bits con tecnología CMOS en un proceso de 28 nanómetros usando aplicaciones de diseño de la empresa Synopsys.

Trabajo de graduación presentado por:
Jonathan Alberto de los Santos Chonay
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,
2014

Vo. Bo.:

(f)_____
MSc. Carlos Alberto Esquit Hernández

Tribunal examinador:

(f)_____
MSc. Carlos Alberto Esquit Hernández

(f)_____
MSc. Roberto Saravia

(f)_____
Ing. Julio Vásquez

Fecha de aprobación: Guatemala, diciembre 2, de 2014

PREFACIO

Es impresionante ver cómo en pocos años la forma de vivir de los humanos ha cambiado drásticamente, podemos ver atrás y darnos cuenta que hace unos 40 años no se disponía de las comodidades tecnológicas que ahora tenemos, todo era más arcaico, más mecánico. El impacto que ha tenido la electrónica en nuestro mundo es impresionante, nadie puede dudarlo. Muchas investigaciones se han realizado en el transcurso de varios años, experimentos incontables donde se demuestra la teoría que los respalda, desarrollo de aplicaciones, desarrollo de más teorías, implementación de máquinas que facilitan el estudio y la investigación, todos estos esfuerzos fueron necesarios para tener lo que ahora tenemos: un mundo tecnológicamente moderno.

Son diversos los campos en donde se han realizado mejorías significativas: en automatización, en telecomunicaciones, biotecnología, almacenamiento de datos, procesamiento de datos a alta velocidad, procesamiento multimedia etc. Todos estos campos son mejorados año con año, pero todo es debido a la mejora absoluta de la fabricación de los circuitos integrados que componen cada uno de los equipos electrónicos que se usan para realizar alguna tarea. Desde la creación del transistor en 1947 la historia de la humanidad cambió, todo fue mejorándose exponencialmente, integrando más y más componentes en espacios cada vez más pequeños. *Very Large Scale Integration* (VLSI) es el término que se usa en la actualidad para definir esta integración a muy gran escala de componentes en una oblea de silicio que es considerablemente pequeña.

VLSI es el tema de este documento, se analizarán aplicaciones de diseño que han permitido la realización de circuitos integrados tan pequeños y cada vez más eficientes en su labor. Esta rama de la electrónica nunca ha sido tratada en Guatemala por lo que este trabajo puede ser el gran comienzo del desarrollo de Guatemala en el mundo de la nanoelectrónica. En el transcurso de la realización de este trabajo se hará la implementación del servidor de licencias de las aplicaciones que serán nuestro arsenal para el desarrollo de un sumador/restador de 32 bits en un proceso de manufacturación de 28 nanómetros. El fin es proveer al laboratorio de electrónica de aplicaciones profesionales desarrolladas por Synopsys para el diseño VLSI, también hacer menos complicada para los futuros diseñadores su labor, realizando manuales de usuario y tutoriales sobre el uso de las aplicaciones.

Es importante mencionar que las licencias de las aplicaciones de Synopsys fueron obtenidas gracias a la institución Universidad del Valle de Guatemala que atendió a la idea del director del departamento de Ingeniería Electrónica de la misma institución, el Ingeniero Carlos Esquit Hernández que es el asesor de este trabajo de graduación. Agradezco en gran manera a la Universidad del Valle de Guatemala y al ingeniero Carlos Esquit por estar siempre en busca de la excelencia en la educación del país y permitirme investigar y aventurarme en este nano pero inmenso universo.

Para Tony y Augusto…

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

La nanoelectrónica es el estudio de los procesos que se requieren para el diseño e implementación de un circuito con una integración a muy gran escala (VLSI) de sus componentes. Una ciencia innovadora ya que con ella se ha logrado un avance considerable de los dispositivos que se usan en las grandes industrias como en nuestro diario vivir. Este trabajo universitario introduce a la institución educativa Universidad del Valle de Guatemala en el diseño de aplicaciones a muy gran escala, con tecnología de vanguardia y software profesional del proveedor Synopsys, una de las grandes empresas de desarrollo en este ámbito.

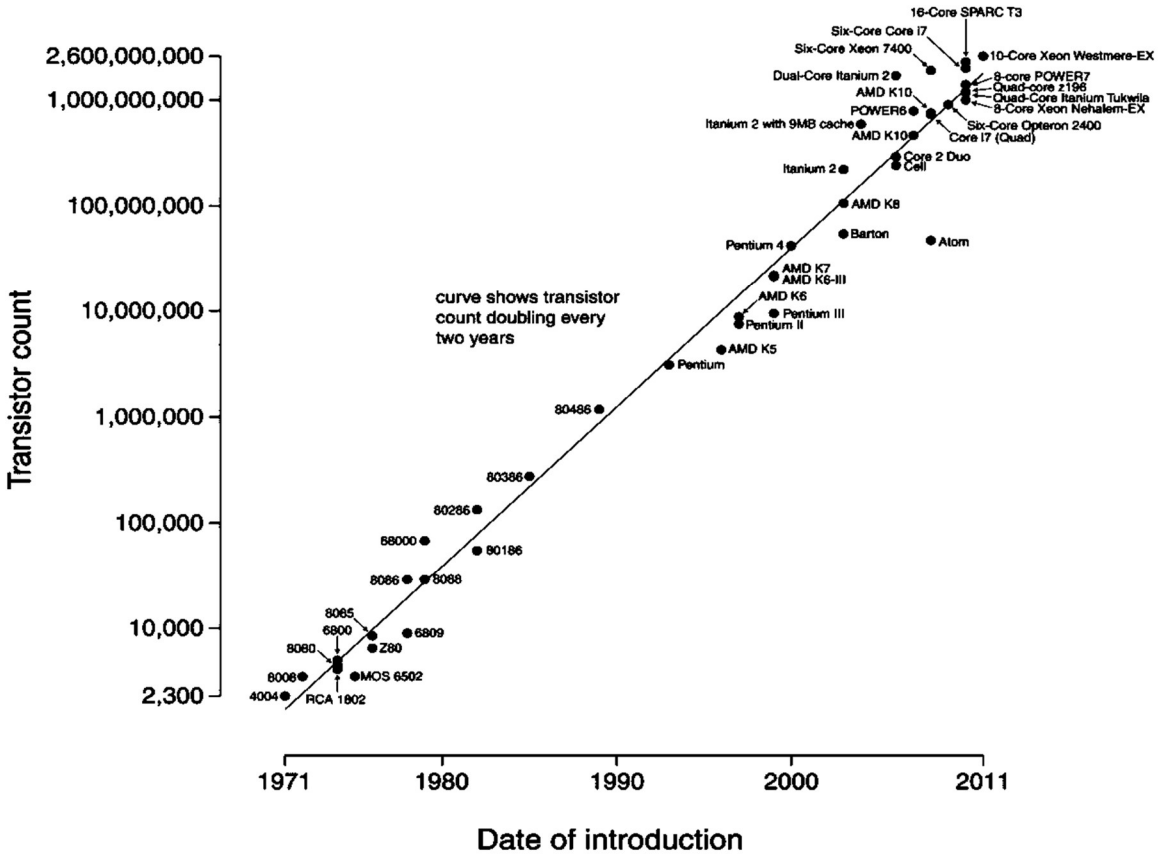
El trabajo consiste en empezar de cero la instalación de las aplicaciones en el laboratorio de electrónica de la universidad, para esto fue necesaria la configuración de un servidor de licencias y de los clientes. Usando como plataforma una distribución de Linux llamada CentOS versión 6.4 que es recomendada para estos fines. Se siguieron las instrucciones dadas por Synopsys para la instalación de todo el entorno y posteriormente, cuando las aplicaciones estaban completamente funcionales se usaron los manuales y ayuda de cada aplicación para el uso correcto. Durante el proceso de aprendizaje se realizaron guías rápidas para el uso de las aplicaciones, esto beneficiará la curva de aprendizaje de nuevos diseñadores, centrándose en realizar un buen diseño y no invertir mucho tiempo revisando extensos manuales. Estas guías de uso de las aplicaciones constituyen gran parte del marco teórico de este trabajo.

Synopsys tiene un programa universitario para el aprendizaje de esta rama de la electrónica y por esta razón provee una librería muy completa para el diseño eficiente, la experimentación, y la investigación. Esta librería se usó como parte final del trabajo de graduación para realizar un proyecto de dificultad media, un sumador/restador de 32 dígitos binarios. Esta librería contiene todo lo necesario para el desarrollo con un proceso de 32-28 nanómetros y está avalada por la gran empresa *foundry* TSMC, por lo que es ideal para la enseñanza universitaria. El proceso de diseño fue exitoso como se mostrará en la sección de resultados donde también se muestran parámetros propios del diseño como retraso de propagación, potencia, energía y la densidad de transistores en el diseño final. (TSMC, 2014)

I. INTRODUCCIÓN

Las aplicaciones de diseño de circuitos integrados son de gran utilidad, estos circuitos pueden asemejarse a cualquier otro en cuanto a funcionamiento final, pero durante el proceso de diseño es necesario tomar muchos factores en cuenta. El proceso de fabricación de estos circuitos es bastante complejo, se usan máquinas muy precisas para lograr la manufacturación de componentes tan pequeños. Para que el lector se haga una idea de lo impresionante que puede llegar a ser este mundo se deja una gráfica, donde se muestran algunos datos de cómo ha mejorado este proceso a lo largo del tiempo. (Farrow, 2011)

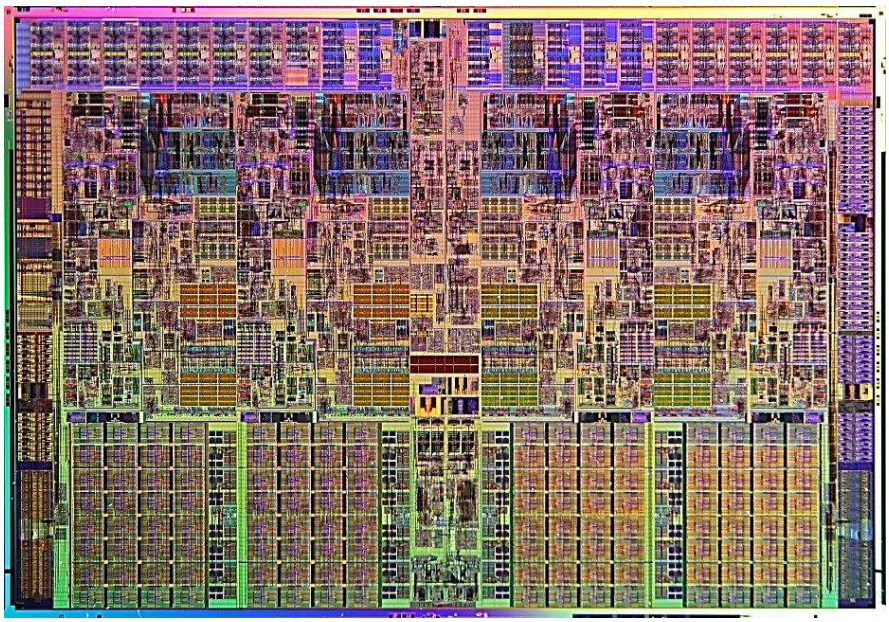
Figura 1: Cantidad de transistores en distintos procesadores



La cantidad de transistores que se ha logrado integrar en una pequeña pieza de silicio en los tiempos actuales es impresionante, tomando en cuenta que en 1971 el procesador 4004 tenía aproximadamente 2300 transistores y uno de los procesadores de unos 43 años después tiene más de 2,600,000,000 transistores, denota un crecimiento y mejora impresionante.

Normalmente se hará referencia a la palabra *die*, que es un término usado en esta ciencia para referirse a la parte del circuito integrado que forma todos sus transistores y todas las capas de metal que se usan para interconectar a cada uno. Por ejemplo la siguiente figura muestra un *die*. (Xedoloh, 2011)

Figura 2: Die de procesador i7



Con el pasar del tiempo no sólo las máquinas que implementan los *die's* se han mejorado, sino que también las aplicaciones de diseño de estos mismos han mejorado considerablemente, la combinación de estas grandes mejoras ha dado como resultado la elaboración de circuitos muy complejos como el procesador mencionado con anterioridad. Este trabajo se centra en la manipulación de estas aplicaciones de diseño, el proyecto aquí diseñado no será implementado.

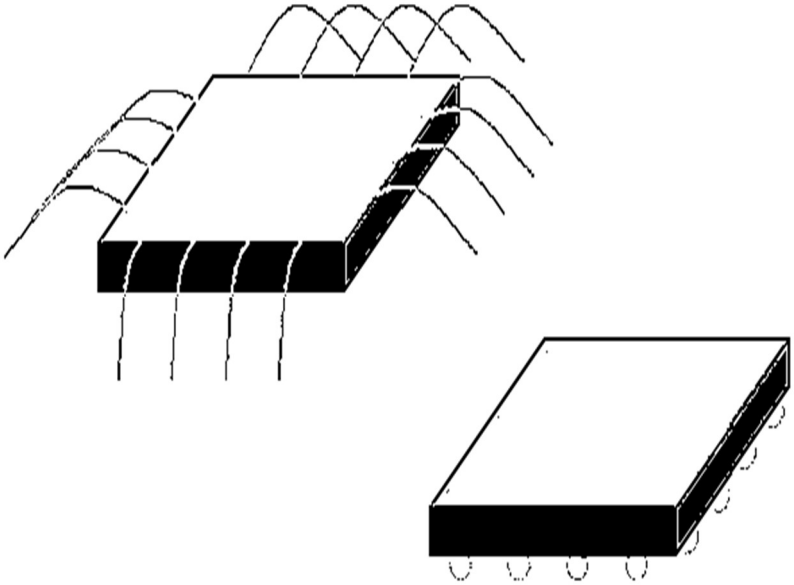
En la actualidad en Guatemala no se presenta ninguna institución educativa o empresa comercial que se dedique a la investigación de esta clase de temas, del diseño con integración a muy gran escala. El principal fin de este trabajo es dejar en el laboratorio de electrónica de la Universidad del Valle de Guatemala un conjunto de aplicaciones de diseño VLSI, que permita a estudiantes interesados en el tema hacer investigación, experimentación y diseño eficiente y profesional de circuitos a nanoescala. En el transcurso del montaje físico y la instalación de las aplicaciones se aprenderán a usar algunas elaborando un proyecto básico, pero que es una parte vital de un procesador: un sumador y restador de número binarios de 32 bits. Como es de suponerse, en el transcurso de la elaboración del proyecto se aprenderá a usar las aplicaciones, siendo la primera aplicación desarrollada con estas aplicaciones en el país se toma la obligación de realizar guías de uso rápido de las aplicaciones que se usen en el proceso de diseño. Esto con el fin de hacer más rápido el aprendizaje de nuevos ingenieros diseñadores.

Como ya se ha mencionado, el proyecto se limita al diseño y nunca se ha pensado en implementar el proyecto por cuestiones de tiempo y costo, también se limita al uso de las librerías que Synopsys provee, estas librerías son de gran utilidad para el diseño eficiente ya que consisten circuitos conocidos que ya están implementados, por ejemplo una compuerta NAND que ya trae archivos que la caracterizan y la definen como su *netlist* que es la definición del circuito en texto plano, modelos de Hspice, *layout* de la compuerta (*die*) entre algunas otras utilidades que ayudan a acelerar el proceso de diseño. La idea del uso de librerías es usada alrededor del mundo de una u otra forma, si no fuera así tardarían cientos de años en la realización de un procesador. Imagine poner 2,600,000,000 transistores uno por uno en el proceso de diseño.

La metodología empleada es simple y puede resumirse en algunos pasos. 1.) Levantar el servidor de licencias 2.) Instalación correcta de las aplicaciones que se vayan a utilizar 3.) Aprender el uso de las aplicaciones 4.) Seguir el flujo de diseño que Synopsys recomienda para la realización del proyecto 5.) Implementación de *pad ring*. Se espera tener el *die* del sumador restador de 32 bits completamente funcional y que cumpla con las reglas de diseño de la librería de Synopsys que como ya se ha dicho son aprobadas por la TSMC para la investigación y experimentación. Las conclusiones importantes de este trabajo son el tiempo de retraso por propagación del circuito final, la densidad de transistores lograda en el *die*, la energía

y la potencia disipada por el circuito. También se toma en cuenta la realización del denominado *pad ring* que se usan para conectar el *die* al empaquetado, por la aplicación se decidió usar *pads* tipo *wire bond*. Este tipo de conexión al exterior tiene los pines externos alrededor del empaquetado y no debajo (*flip chip*) como es usual en los microprocesadores de Intel por ejemplo. (Journal, 2003)

Figura 3: Wire bond (izquierda) Flip Chip (derecha)



Conforme el ingeniero desarrollador avanza y su conocimiento crece, se da cuenta de lo mucho que queda por recorrer, la cantidad de información encontrada en la nanoelectrónica puede representar una vida entera de estudio y dedicación, este trabajo presenta el inicio de ese recorrido. A continuación se presentan las secciones donde se muestra la investigación realizada, en el marco teórico se encontrarán las guías rápidas de diseño e información teórica adicional, además si gusta puede visitar la sección de anexos donde se presenta un espacio en línea donde se ha posteado un pequeño video tutorial que puede hacer el aprendizaje más eficiente. Tome en cuenta que tanto las guías que se presentan en este trabajo y el video tutorial es básico y para operaciones más específicas no hay mejor fuente de información que los manuales oficiales de las aplicaciones que las provee Synopsys en su portal.

II. OBJETIVOS

A. Generales

Montar un servidor de licencias que permita el uso correcto de las aplicaciones profesionales para el diseño VLSI de empresa Synopsys y realizar la implementación de una aplicación con acabado profesional usando estas aplicaciones.

B. Específicos

1. Proveer al laboratorio de electrónica de aplicaciones profesionales para el diseño VLSI.
2. Realizar una guía rápida para el uso correcto de las diferentes aplicaciones que se vayan a utilizar.
3. Diseñar un sumador/restador completo de 32 bits usando las aplicaciones de Synopsys instaladas en un proceso de fabricación de 28 nanómetros.
4. Realizar un diseño eficiente del sumador/restador aplicando el método de *Logical Effort*.
5. Diseñar el sumador/restador tomando en cuenta el efecto de los retrasos ocasionados por el cableado del *layout* final usando *parasitic extraction*.
6. Implementar el *pad ring* apropiado para conectar el *die* a un empaquetado.
7. Obtener el diseño del sumador/restador de 32 bits listo para ser fabricado, con las reglas de diseño de la empresa *Taiwan Semiconductor Manufacturing Company, Limited*. (Synopsys, TSMC - Synopsys Collaboration, 2014)

II. JUSTIFICACIÓN

Desde hace muchos años la electrónica ha ido revolucionando la industria, es la culpable de muchas herramientas que usamos en diario vivir que sin duda hacen más fácil nuestras vidas, optimizaciones de procesos, procesamiento de datos masivos a gran velocidad, telecomunicaciones y un sinfín de aplicaciones. La famosa Ley de Moore se ha cumplido con mucha exactitud, aunque con la mejora en los procesos de manufacturación de las empresas dedicadas a la fabricación de semiconductores esta llamada ley podría peligrar (Pepone, 2014). Esto no hubiera sido posible sin el desarrollo y constante mejora de los circuitos integrados logrando tener una integración a muy grande escala comúnmente llamada VLSI. Este campo de estudio, sin tan siquiera ser explotado en nuestro país Guatemala, aparte de presentar un mundo de oportunidades añade una gran cantidad de conocimiento al estudiante de verdadera importancia en su vida profesional.

Como se ha mencionado ninguna institución de nivel superior en el país y de hecho muy pocos en Latinoamérica (Argentina, México, Brasil) (Anónimo, 2002) cubre este importante tema y es necesario empezar a impartirlo. Lo que se busca en este trabajo es instalar en el laboratorio de ingeniería electrónica una serie de aplicaciones profesionales dedicadas al diseño VLSI, también se aprenderán a utilizar estas aplicaciones y se realizará un sumador/restador de 32 bits que es una parte vital de un microprocesador y se dejará listo para su fabricación o para ser implementado en otras aplicaciones, este es otro de los fines del trabajo. El aporte es significativo en la universidad ya que se está rompiendo una barrera importante en la educación superior ya que se dará paso a una parte de la electrónica que mundialmente necesita ser explorada y explotada, con muchos temas de investigación y desarrollo.

III. MARCO TEÓRICO

En este capítulo se presenta toda la teoría que se necesitó para poder desarrollar todo el proyecto, básicamente se trata de una síntesis de los procesos descritos en los manuales que provee Synopsys, es importante recalcar que se menciona solamente lo necesario ya que es imposible poder recabar toda la información de estos manuales en este texto, además, hay temas que van fuera del alcance del mismo como texto introductorio. La universidad del Valle en el departamento de electrónica ha adquirido 87 aplicaciones dedicadas para muchas funciones específicas en el mundo de la nanoelectrónica, en este trabajo se usarán algunas aplicaciones que se usan para algún proceso estándar de diseño e implementación y no se usan aplicaciones muy específicas.

Respecto al sistema operativo utilizado se eligió el sistema operativo CentOS, este sistema operativo podría decirse que es un clon de Red Hat ya que la comunidad desarrolladora utiliza el *kernel* de Red Hat que ha sido publicado por la empresa para la creación del mismo. La gran ventaja es que CentOS, aparte de ser completamente gratuito es 100% compatible con las aplicaciones de Synopsys. Las siguientes secciones presentan de forma ordenada guías rápidas para la instalación y el uso de las aplicaciones.

A. Instalación de CentOS

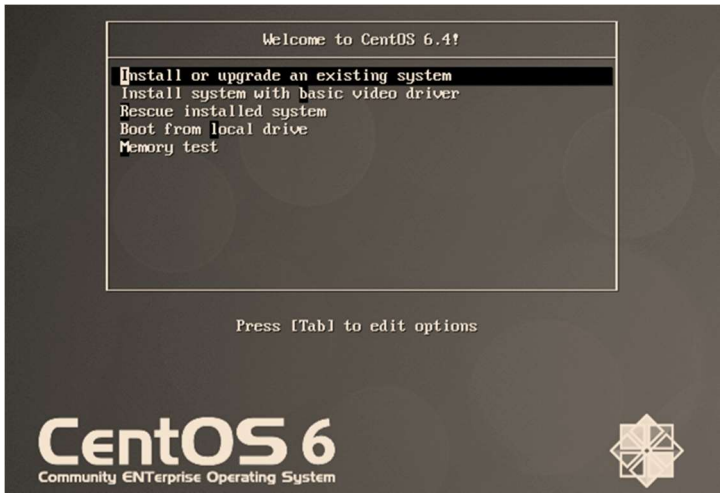
1. Prerrequisitos. Para la instalación de CentOS se requieren que la computadora cumpla con los siguientes requisitos:

a. Procesador x86_64 reciente.
b. Múltiples CPUs o núcleos (un CPU con 4 o más núcleos es recomendado)
c. CPU de 2GHz o superior.
d. 4 GB de RAM.
e. Tarjeta Ethernet de 2 Gbps. (Synopsys, Synopsys Licensing QuickStart Guide, 2014)

2. Proceso de instalación. El proceso de instalación de CentOS es básico, únicamente se sigue el asistente de instalación que trae la versión 6.4 del mismo, este asistente es muy gráfico y no hay forma de perderse en el proceso si se hace una instalación básica, la única complicación podría surgir cuando se requiera modificar las particiones donde se instalará, esto en el caso de tener un sistema operativo extra en el disco duro y pretender tener un *booteo* dual en la máquina. Si se planifica tener un servidor completamente en Linux y dedicar la Workstation para este fin se tiene un disco duro vacío y la instalación es un proceso simple. (Mamani, 2010)

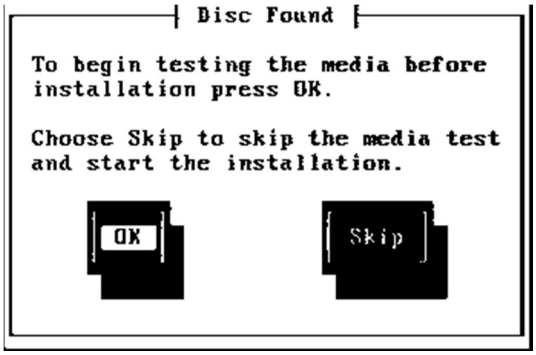
a. Elegir *Install or upgrade an existing system* en la pantalla de *booteo*.

Figura 4: Menú de booteo de CentOS



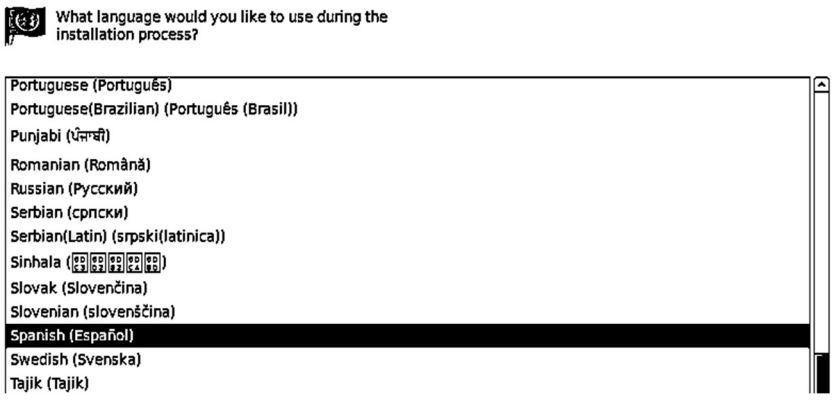
b. Comprobar que el disco de instalación no tenga errores.

Figura 5: Comprobación de errores en disco



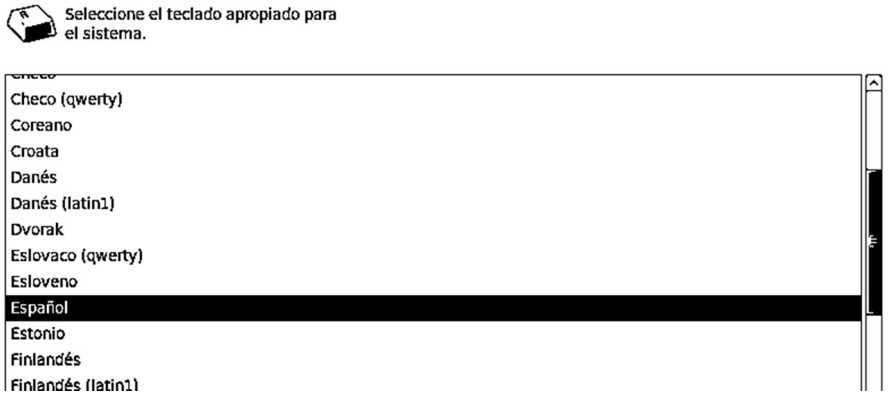
c. Elegir idioma del sistema operativo (algunos idiomas necesitarán la descarga de paquetes adicionales).

Figura 6: Elección de idioma



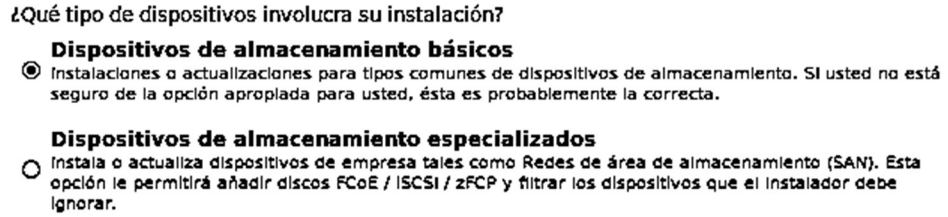
d. Elegir la distribución del teclado de acuerdo al hardware instalado en la PC.

Figura 7: Distribución de teclado



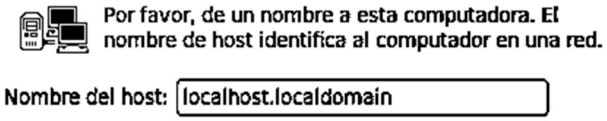
e. Elegir el tipo de almacenamiento que se tiene instalado en la PC.

Figura 8: Tipo de almacenamiento



f. Asignar el nombre a la computadora. Este es el *hostname* de la PC, servirá después en la configuración del servidor de licencias Synopsys.

Figura 9: Nombre de computadora

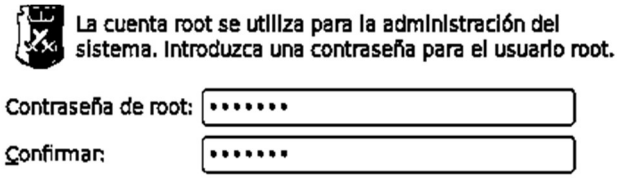


g. Elegir un huso horario.

Figura 10: Huso horario

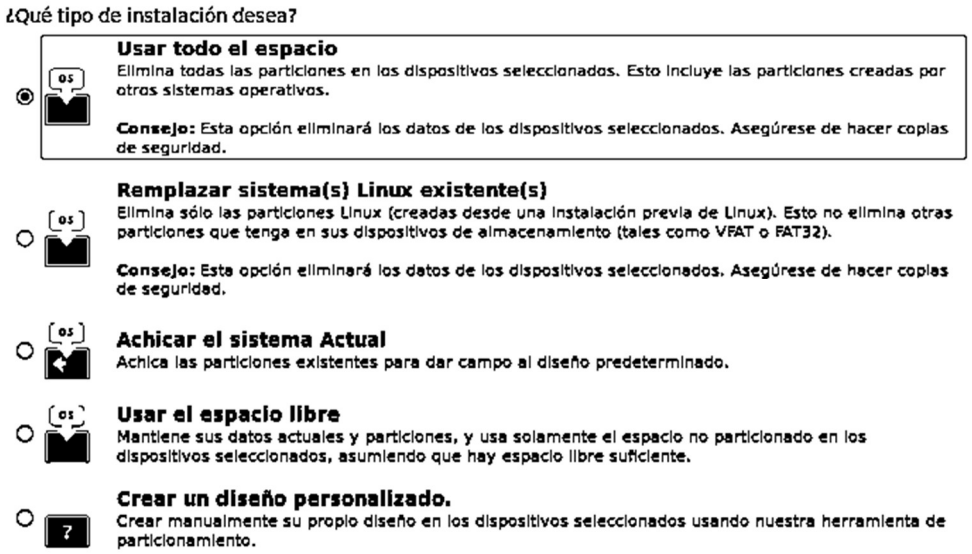
h. Contraseña para usuario *root*.
Nota: No olvidar esta contraseña ya que es la del usuario (*root*) con completo dominio del sistema.

Figura 11: Contraseña de superusuario



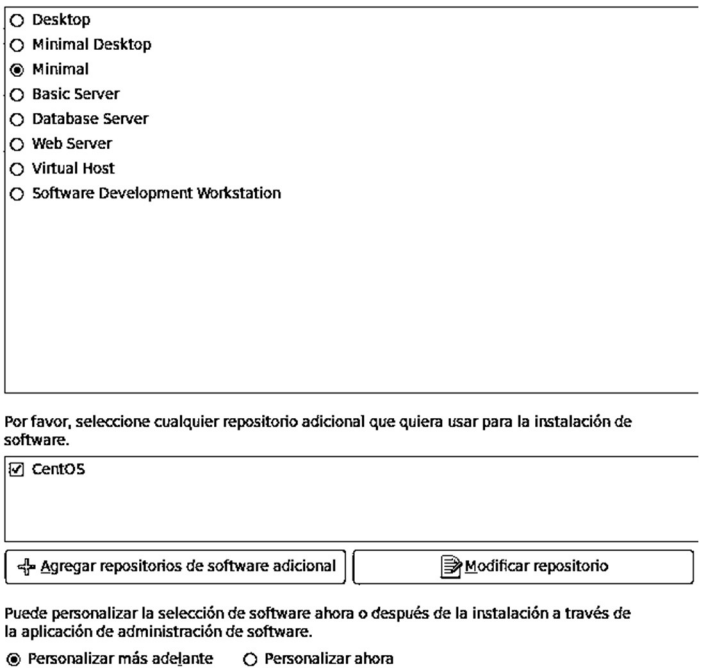
i. Elegir el tipo de particionamiento del disco duro. Se recomienda usar un disco duro en blanco y usar todo el espacio del mismo, si no es posible lo recomendable es crear un diseño personalizado de instalación de particionamiento de disco duro.

Figura 12: Particionamiento



j. Personalizar la instalación. En este paso elegir la opción *Basic Server* y la de "Personalizar ahora", en la personalización se agregan los programas que se desean instalar automáticamente en el proceso de instalación del sistema operativo. Se recomienda agregar un entorno gráfico para usar el servidor como cliente al mismo tiempo y aprovechar sus capacidades de hardware en proyectos muy grandes. (Mamani, 2010)

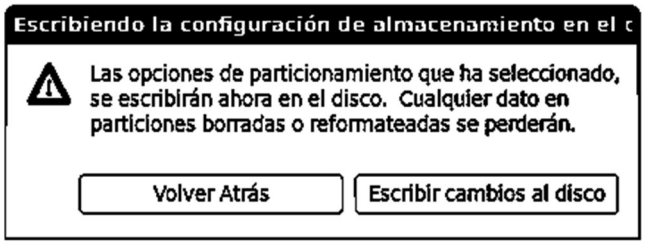
Figura 13: Personalización de instalación



k. Guardar cambios en disco duro.

Nota: Todos los datos en las particiones a formatear serán destruidos al hacer click en "Escribir cambios al disco".

Figura 14: Escribir datos en disco



l. Esperar a la copia de los paquetes.

Figura 15: Instalación en proceso

m. Instalación terminada. Click en reiniciar para completar la instalación.

Figura 16: Instalación completada exitosamente

Felicitaciones, la instalación de su CentOS está completa.

Por favor, reinicie para usar el sistema instalado. Note que las actualizaciones pueden estar disponibles para asegurar el funcionamiento apropiado de su sistema y su instalación es recomendada luego de reiniciar.

Atrás Reiniciar

n. Ejecución de CentOS con entrono gráfico instalado (gnome).

Figura 17: Ambiente gráfico CentOS 6.4

B. Instalación y ejecución del instalador de aplicaciones Synopsys

1. Registrarse en la página https://solvnet.synopsys.com usando el correo electrónico de la universidad o sitio. Tomar en cuenta que la autorización para el acceso lleva aproximadamente 3 días. (Synopsys, Synopsys Licensing QuickStart Guide, 2014)
2. Ir a la página https://solvnet.synopsys.com/DownloadCenter si es necesario ingresar su usuario y contraseña.
3. En *My Product Releases* elegir *Synopsys Installer* y descargar la versión 3.1 vía HTTPS usando el botón *Download here*. El servidor FTP da varios problemas de conexión, por lo que no se recomienda usarlo.
4. Leer y aceptar los términos de licencia.
5. Descargar el archivo `installer_v3.1.tar.Z`.
6. Crear una carpeta donde estarán todas las aplicaciones de Synopsys.

```
# mkdir /usr/synopsys
```

7. Es recomendable no realizar el proceso de instalación como superusuario, para no borrar accidentalmente archivos o carpetas del S.O. Sin embargo, por la ubicación de la carpeta, sí se requieren permisos de superusuario para crearla, como se puede observar en el comando, debido al prompt #, pero se cambiará de propietario con el siguiente comando. (Synopsys, Synopsys Licensing QuickStart Guide, 2014)

```
# chown Electronica /usr/synopsys
```

   Donde "Electronica", es el nombre de usuario que será el nuevo propietario de la carpeta. Después de esto se podrá leer y escribir en la carpeta como un usuario normal, con menos riesgos de cometer algún error que pueda dañar parte, o completamente la plataforma.
8. Crear una carpeta para alojar los archivos del Synopsys Installer. (notar el cambio de `#` a `$`)

```
$ mkdir /usr/synopsys/installer
```

9. Mover el archivo `installer_v3.1.tar.Z` a `/usr/synopsys/installer`

```
$ mv /directorio_archivo/installer_v3.1.tar.Z
/usr/synopsys/installer/installer_v3.1.tar.Z
```

10. Descomprimir archivo `installer_v3.1.tar.Z`.

```
$ tar xvzf installer_v3.1.tar.Z
```

11. Dar permisos de ejecución a los archivos del *Installer*.

```
$ chmod -R 755 /usr/synopsys/installer
```

12. Agregar el directorio del *Installer* al PATH de Linux.

```
$ export PATH=$PATH:/usr/synopsys/installer
```

13. Agregar esta última línea al archivo `$HOME/.bashrc` con el fin de que los comandos del *Installer* estén disponibles en la CLI aun si es reiniciado el servidor. Tomar en cuenta que esto sólo va a afectar al usuario que estemos usando, en este ejemplo sería al usuario "Electronica".
14. Iniciar el *Installer* con el comando `setup.sh`. (No olvidar la extensión sh). Se debe iniciar el entorno gráfico del Synopsys Installer.

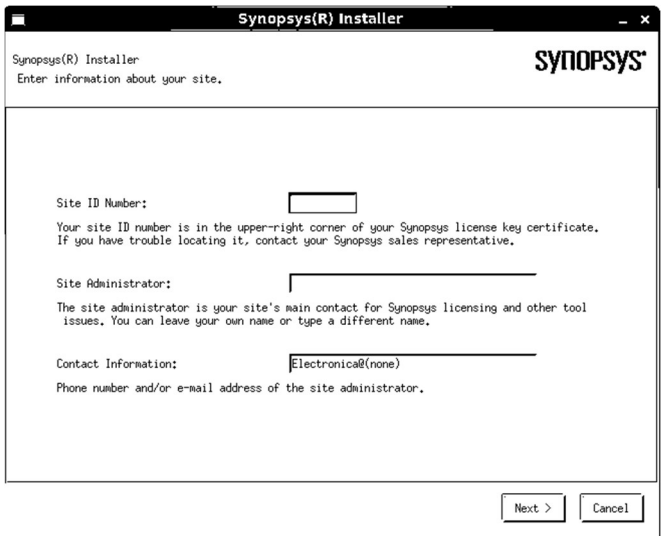
Figura 18: Pantalla de bienvenida a Synopsys Installer

C. Cómo instalar aplicaciones con el Installer

Nota: Todas las aplicaciones de Synopsys usan el *installer* para ser instaladas. (Synopsys, Synopsys Licensing QuickStart Guide, 2014)

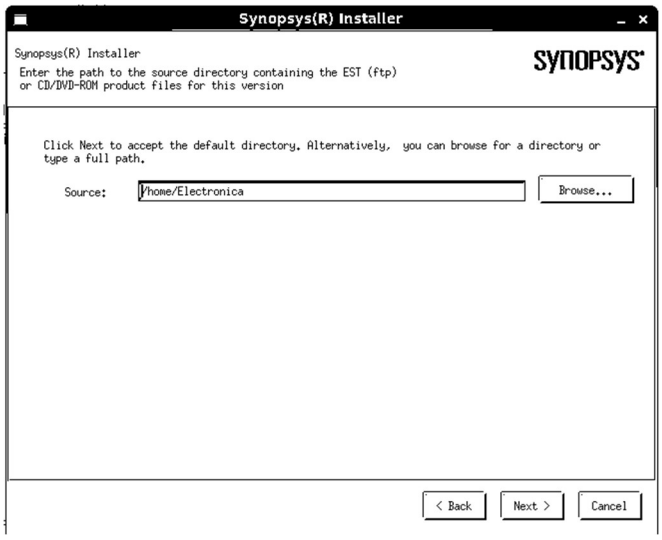
1. En la pantalla introductoria dar click en *Start*. Ver Figura 18.
2. Llenar los datos que se solicitan por el asistente, estos datos los posee la persona que representa al sitio en Synopsys, es decir, la persona que adquirió la licencia.

Figura 19: Información acerca del sitio



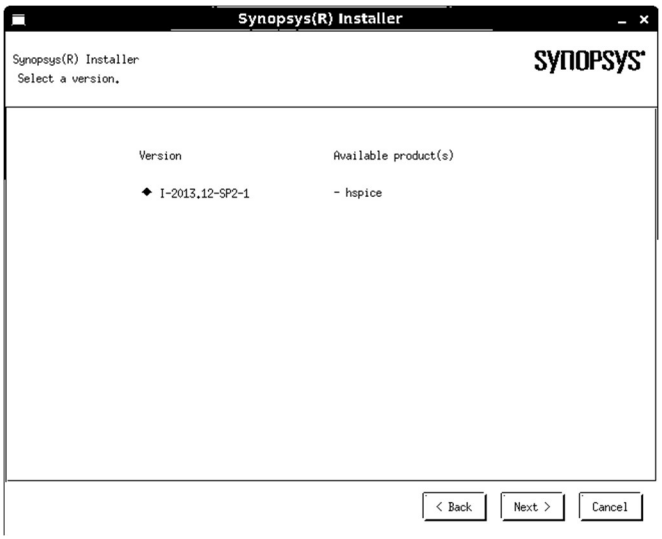
3. Elegir la carpeta temporal donde están los archivos de instalación *common* y de la plataforma de Linux. Estos archivos se descargan en la página de SolvNet. En ocasiones no hay un paquete de instalación de la plataforma Linux, pero la denominada AMD también es compatible con el CentOS instalado.

Figura 20: Directorio con archivos de instalación



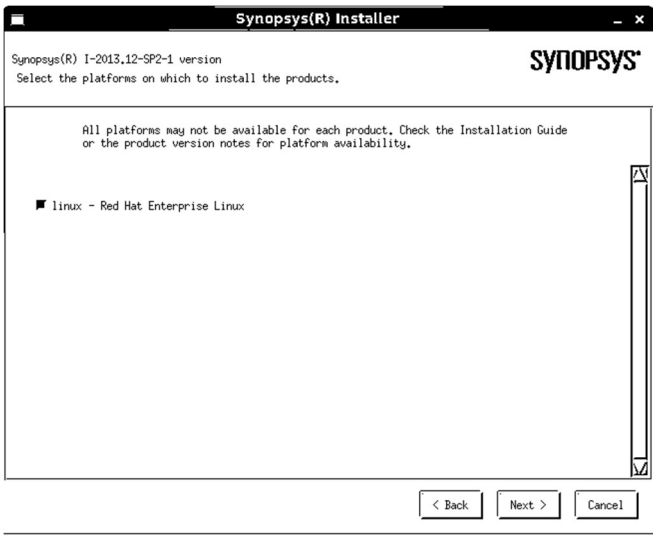
4. Se debe reconocer la aplicación a instalar como se muestra en la siguiente figura. Algunas aplicaciones después de esta pantalla muestran alguna otra información como otras aplicaciones que se instalarán como dependencia, únicamente es necesario dar click en *next* si fuera el caso.

Figura 21: Confirmación de la aplicación a instalar



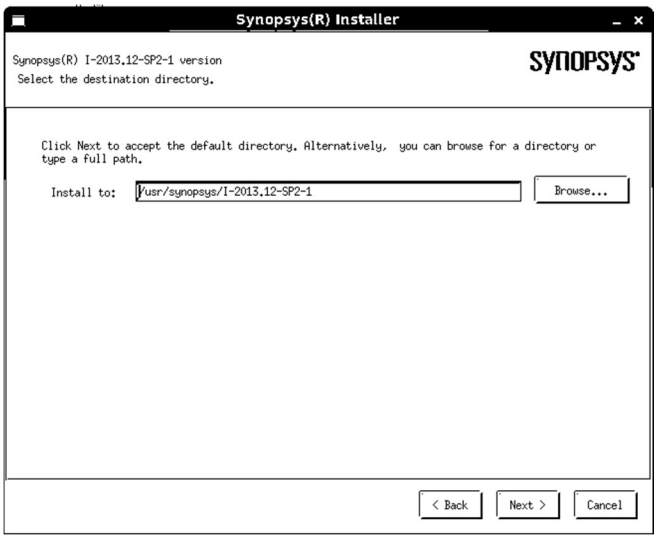
5. Elegir la plataforma donde se instalará la aplicación, recordar que CentOS es un "clon" de Red Hat.

Figura 22: Confirmación de la plataforma



6. Elegir la carpeta de destino, en esta carpeta estarán todos los archivos que la aplicación a instalar necesita para su ejecución, esta carpeta es la misma donde se instaló el *Installer*, por ejemplo: `/usr/synopsys/nombre_app`. Las aplicaciones pueden necesitar de alguna librería propia del S.O como dependencia, si se tiene problema con estas dependencias el *Installer* no mostrará algún error, el error se mostrará cuando se trate de iniciar la aplicación, para resolver este tipo de problemas use el comando `yum whatprovides` que mostrará el paquete que provee la librería faltante, posteriormente instale este paquete con el comando `yum install "paquete"`.

Figura 23: Directorio de instalación

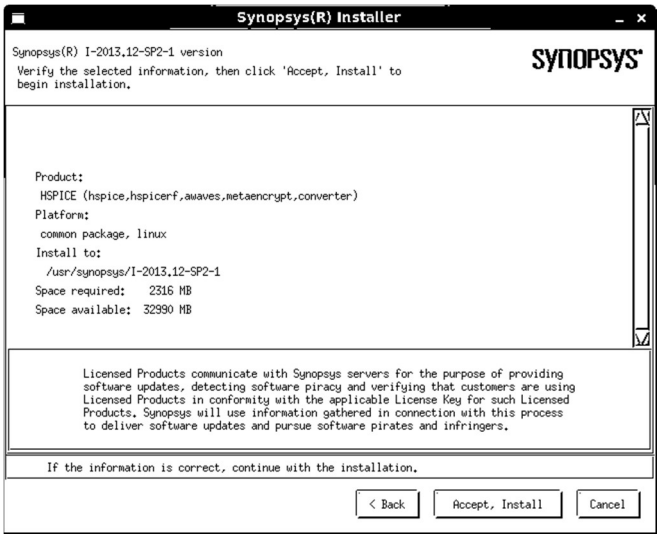


7. Si la carpeta de destino no existe se mostrará un mensaje para aprobar la creación del directorio, presionar *yes*.

Figura 24: Crear directorio de instalación

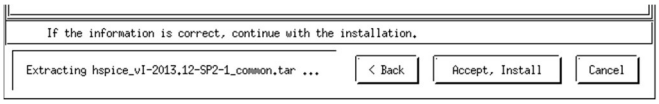
8. Verificar que la información se la correcta, si hay algún error se puede retroceder y corregirlo. Cuando todo esté correcto dar click en *Accept, Install*.

Figura 25: Verificación preinstalación



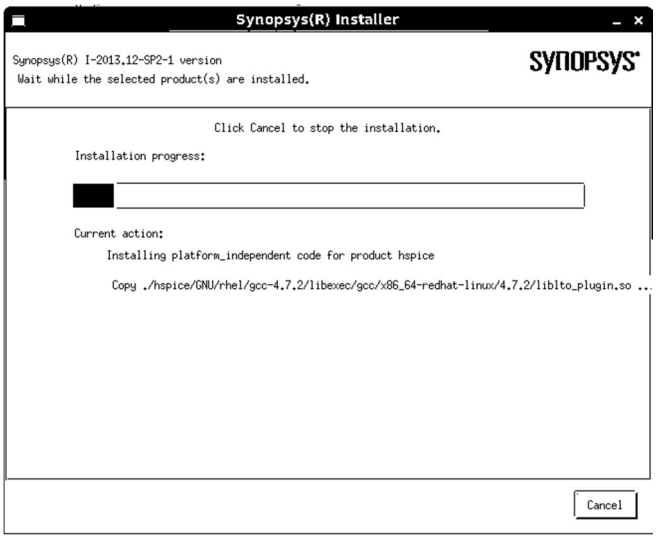
9. Los paquetes se comenzarán a extraer.

Figura 26: Extracción de paquetes para instalación



10. Esperar a que el proceso de instalación termine.

Figura 27: Proceso de instalación



11. Se muestran algunas notas de post-instalación, verificar que la instalación haya sido instalada con éxito.

Figura 28: Notas post-instalación

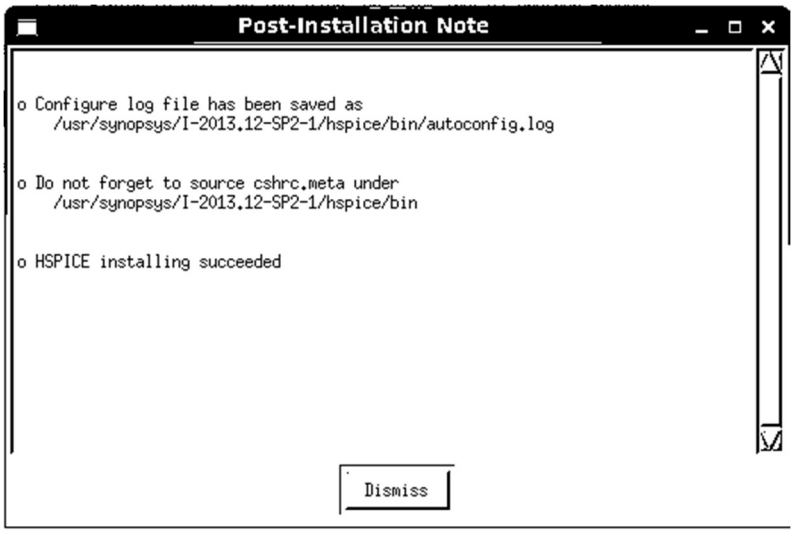
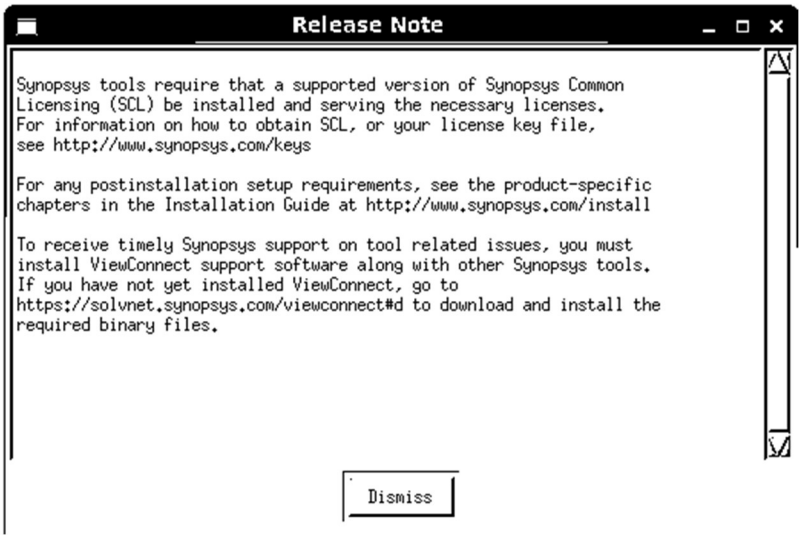
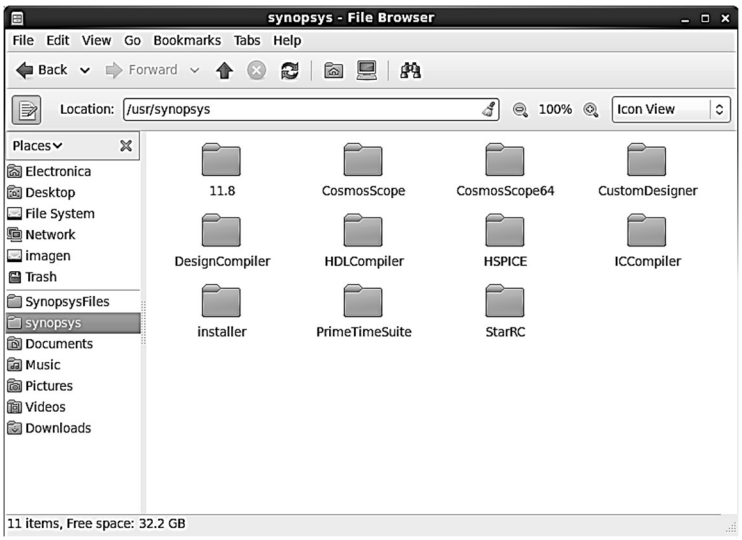


Figura 29: Notas de lanzamiento



12. Muestra de la carpeta donde se alojan las aplicaciones de Synopsys.

Figura 30: Muestra de carpeta de instalación de aplicaciones



D. Instalación y ejecución del servicio de licencia Synopsys Common Licensing v 11.8

1. Determinar el *HostID* que es la dirección MAC de la tarjeta Giga Ethernet conectada a lared donde se usará el servicio de licencia, esta dirección fue notificada a la empresa Synopsys al momento de adquirir la licencia por lo que no se puede cambiar, la forma de visualizar esta dirección MAC en Linux es mediante el comando: (Synopsys, Synopsys Licensing QuickStart Guide, 2014)

```
$ ifconfig -a | grep HWaddr
eth0        Link encap:Ethernet  HWaddr XX:YY:RR:UU:WW:ZZ
```

Entonces el *HostID* = `XXYYRRUUWWZZ` (sin los ":")

2. Ir a la página https://solvnet.synopsys.com/DownloadCenter si es necesario ingresar su usuario y contraseña.
3. En *My Product Releases* elegir Synopsys Common Licensing y descargar la versión 11.8 vía HTTPS dando click en el botón *download now*.
4. Leer y aceptar los términos de licencia.
5. Descargar el archivo *common* y el archivo de la plataforma específica a la distribución de Linux. Elegir un directorio temporal para la descarga.
6. Instalar el Synopsys Common Licensing usando el Synopsys Installer como se explica en la sección anterior.
7. Descargar el archivo de licencia mediante la siguiente dirección https://solvnet.synopsys.com/smartkeys/smartkeys.cgi si fuera necesario ingrese su nombre de usuario y contraseña.
8. Click en *Key Retrieval* y seleccione el ID de sitio correcto.
9. Click en *continue*, en un momento llega el archivo de licencia vía correo electrónico.
10. Personalizar el archivo de licencia recibido.
    a. El archivo debe verse como lo siguiente, añadiendo muchas líneas más.

Cuadro 1: Ejemplo de archivo de licencia

```
   #-------------------------------------------------------
#Server Config: 0000052142
#-------------------------------------------------------
SERVER hostname1 0128D07E321F 27020
VENDOR snpslmd /path/to/snpslmd
USE_SERVER
INCREMENT SSS snpslmd 1.0 10-dec-2014 1 3EE593F14EED \
VENDOR_STRING="382be a5896 de613 1618c 27e26 40bb1 40cec 13318 8da63
128" \
NOTICE="Licensed to ABC Corporation [PLEASE DO NOT DELETE THIS SSS
KEY]" \
SN=RK:0:0:802047 ISSUER="Synopsys Inc." \
START=03-oct-2005
INCREMENT hspice snpslmd 2013.12 10-dec-2014 8 VENDOR_STRING=^1+S \
SUPERSEDE ISSUED=06-dec-2013 ck=115 SN=RK:6719-0:733381:117716 \
START=03-oct-2005 AUTH={ snpslmd=( LK=6D8411D4365F7D1C39DB) \
avantd=( LK=CDE4E1546CD26F379ACB) }
INCREMENT metawaves snpslmd 2013.12 10-dec-2014 8 VENDOR_STRING=^1+S \
SUPERSEDE ISSUED=06-dec-2013 ck=39 SN=RK:6719-0:733381:117716 \
START=03-oct-2005 AUTH={ snpslmd=( LK=6D3441449638F796655F) \
avantd=( LK=DD44A164467F8C415694) }
```

    b. Mover el archivo de licencia a la carpeta recomendada `/usr/synopsys/SCL_Folder/admin/license/license.lic`
    c. No borrar la línea `USE_SERVER`, esta debe estar antes de cualquier línea `INCREMENT`.
    d. Abrir el archivo de licencia y verificar que haya solo una línea `VENDOR snpslmd /usr/synopsys/SCL_Folder/linux/bin/snpslmd`, si existe alguna otra línea `VENDOR` o `DAEMON` es necesario borrarla. La dirección de snpslmd está en la carpeta de instalación de SCL.

e. Todas las líneas deben empezar con `SERVER,VENDOR,USE_SERVER, PACKAGE,INCREMENT, o #.`
f. Hay líneas que comienzan con el carácter `(" \")` estas líneas son continuación de una línea anterior.
g. No deben aparecer líneas en blanco ya que pueden causar problemas.
h. Borrar caracteres extraños como > o >> que en ocasiones son añadidos en los correos electrónicos.
i. Obtener el hostname de la PC.

```
$ hostname
```

j. Modificar la línea SERVER.

```
SERVER hostname hostid tcp_port_number
```

En `hostname` se debe poner el resultado del inciso i. de este paso, en `hostid` la MAC address de la tarjeta Giga Ethernet como se mostró en el paso 1 de esta sección, Es muy importante que el hostid coincida con el verdadero hostid de otra manera se invalidará el archivo de licencia. Por último en `tcp_port_number`, el puerto TCP que se usará, el puerto por defecto es el 27020, por cuestiones de desempeño, los puertos 27000-27009, no son recomendados por Synopsys. A continuación se muestra un ejemplo.

```
SERVER uvgserver XXYYRRUUWWZZ 27020
```

k. Guardar los cambios en el archivo de licencia, debe ser un archivo en texto plano, en el proceso se tuvieron inconvenientes con el editor de texto de CentOS, se modificó el archivo de licencia con notepad de MS Windows.
l. Verificar la exactitud del archivo de licencia.

```
/usr/synopsys/SCL_Folder/Linux/bin/sssverify
/usr/synopsys/SCL_Folder/ admin/license/license.lic
```

Asegurarse de que la verificación sea exitosa, de no ser así existen problemas en el formato del archivo de licencia.

11. Verificar que no haya ningún servicio levantado que use nuestro archivo de licencia.

```
lmdown -c /usr/synopsys/11.8/admin/license/license.lic
```

12. Iniciar el servidor de licencias SCL.

```
lmgrd -c /usr/synopsys/11.8/admin/license/license.lic
```

El log obtenido debe ser algo parecido al siguiente. (Se han omitido algunas líneas por brevedad)

-

Cuadro 2: Log de inicialización de servidor de licencias

```
11:18:10 (lmgrd) ---------------------------------------
11:18:10 (lmgrd)   Please Note:
11:18:10 (lmgrd)
11:18:10 (lmgrd)   This log is intended for debug purposes only.
11:18:10 (lmgrd)   In order to capture accurate license
11:18:10 (lmgrd)   usage data into an organized repository,
11:18:10 (lmgrd)   please enable report logging. Use Flexera Software LLC's
11:18:10 (lmgrd)   software license administration  solution,
11:18:10 (lmgrd)   FlexNet Manager, to  readily gain visibility
11:18:10 (lmgrd)   into license usage data and to create
11:18:10 (lmgrd)   insightful reports on critical information like
11:18:10 (lmgrd)   license availability and usage. FlexNet Manager
11:18:10 (lmgrd)   can be fully automated to run these reports on
11:18:10 (lmgrd)   schedule and can be used to track license
11:18:10 (lmgrd)   servers and usage across a heterogeneous
11:18:10 (lmgrd)   network of servers including Windows NT, Linux
11:18:10 (lmgrd)   and UNIX. Contact Flexera Software LLC at
11:18:10 (lmgrd)   www.flexerasoftware.com for more details on how to
11:18:10 (lmgrd)   obtain an evaluation copy of FlexNet Manager
11:18:10 (lmgrd)   for your enterprise.
11:18:10 (lmgrd)
11:18:10 (lmgrd) ---------------------------------------------
11:18:10 (lmgrd)
11:18:10 (lmgrd)
11:18:10 (lmgrd) Server's System Date and Time: Wed Aug 20 2014 11:18:10 CST
11:18:10 (lmgrd) SLOG: Summary LOG statistics is enabled.
[Electronica@scluvg license]$ 11:18:10 (lmgrd) FlexNet Licensing (v11.12.1.0 build
146690 i86_lsb) started on scluvg (linux) (8/20/2014)
11:18:10 (lmgrd) Copyright (c) 1988-2014 Flexera Software LLC. All Rights Reserved.
11:18:10 (lmgrd) World Wide Web:  http://www.flexerasoftware.com
11:18:10                  (lmgrd)                  License                  file(s):
/usr/synopsys/11.8/admin/license/Synopsys_Key_Site_31759_Server_249913_snpslmd.txt
11:18:10 (lmgrd) lmgrd tcp-port 27020
11:18:10 (lmgrd) (@lmgrd-SLOG@) ===============================================
11:18:10 (lmgrd) (@lmgrd-SLOG@) === LMGRD ===
11:18:10 (lmgrd) (@lmgrd-SLOG@) Start-Date: Wed Aug 20 2014 11:18:10 CST
11:18:10 (lmgrd) (@lmgrd-SLOG@) PID: 3355
11:18:10 (lmgrd) (@lmgrd-SLOG@) LMGRD Version: v11.12.1.0 build 146690 i86_lsb (
build 146690 (ipv6))
11:18:10 (lmgrd) (@lmgrd-SLOG@)
11:18:10 (lmgrd) (@lmgrd-SLOG@) === Network Info ===
11:18:10 (lmgrd) (@lmgrd-SLOG@) Socket interface: IPV6
11:18:10 (lmgrd) (@lmgrd-SLOG@) Listening port: 27020
11:18:10 (lmgrd) (@lmgrd-SLOG@)
11:18:10 (lmgrd) (@lmgrd-SLOG@) === Startup Info ===
11:18:10 (lmgrd) (@lmgrd-SLOG@) Server Configuration: Single Server
11:18:10 (lmgrd) (@lmgrd-SLOG@) Command-line options used at LS startup: -c
/usr/synopsys/11.8/admin/license/Synopsys_Key_Site_31759_Server_249913_snpslmd.txt
11:18:10           (lmgrd)           (@lmgrd-SLOG@)           License           file(s)
used:   /usr/synopsys/11.8/admin/license/Synopsys_Key_Site_31759_Server_249913_snpslmd
.txt
11:18:10 (lmgrd) (@lmgrd-SLOG@) ===============================================
11:18:10 (lmgrd) Starting vendor daemons ...
11:18:10 (lmgrd) Started snpslmd (internet tcp_port 36769 pid 3356)
11:18:10 (snpslmd) FlexNet Licensing version v11.12.1.0 build 146690 i86_lsb
08/20/2014 11:18:10 (snpslmd) Synopsys Corporate Licensing (SCL) Release: version
SCL_11.8
11:18:33 (snpslmd) SLOG: Summary LOG statistics is enabled.
11:18:33 (snpslmd) SLOG: FNPLS-INTERNAL-CKPT1
11:18:33 (snpslmd) SLOG: VM Status: 0
11:18:33 (snpslmd) SLOG: FNPLS-INTERNAL-CKPT2
11:18:33 (snpslmd) Server started on scluvg for:    SSS
```

Continuación de Cuadro 2.

```
11:18:33 (snpslmd) ACS         ADP_OA         AN-Impl3D_all
11:18:33 (snpslmd) AdvTelecomLib_simulation BASIC_ANALYSES    BATCH_MEASURE
11:18:33 (snpslmd) BATT_TOOL    BLHandler_sbl_amba2 BLHandler_sbl_amba3_axi
11:18:33 (snpslmd) COSMOS_SCOPE    COSMOS_VO    CPI-100
11:18:33 (snpslmd) HDL-Compiler    HERCULES-CRYPT_XREF_DATA HERCULES-DP_MT
11:18:33   (snpslmd)   HERCULES-EXPLORER_DRC   HERCULES-EXPLORER_FILTERS   HERCULES-
   EXPLORER_LVS
11:18:33 (snpslmd) HERCULES-NETLIST HERCULES-RUN_TRAN HERCULES_DEBUGGER
11:18:33  (snpslmd)  HSPICE_MODEL_LIBRARY  ICValidator2-CompareEngine  ICValidator2-
   GeometryEngine
11:18:33 (snpslmd) VHDL-Analyzer    VHDL-Compiler    VHDL-Cycle-Sim
11:18:33 (snpslmd) VHDL-Elaborator VHDL-Event-Sim    VHDL-Tools
11:18:33 (snpslmd) VHDL-VirSim    VIEWLOGIC_FRAMEWAY VIEWLOGIC_NET
11:18:33 (snpslmd) VIP-LIBRARY-SVT VIP-PROTOCOL-ANALYZER VSDK.primary
11:18:33 (snpslmd) VSDK.simulation VT_Assertions    VT_AssertionsRuntime
11:18:33 (snpslmd) VT_Coverage    VT_CoverageRuntime VT_CoverageURG
11:18:33 (snpslmd) VT_CoverageURG    VT_Testbench    VT_TestbenchRuntime
11:18:33 (snpslmd) VT_VCS_NATIVE_LP VT_VCS_NLP_SIGNALS VT_Visual
11:18:33 (snpslmd) Verdi       VerdiCoverage    WF_API
11:18:33 (snpslmd) WF_API_HSPICE    WF_API_STARSIM    XVCSDebugger
11:18:33 (snpslmd) hspice      hspice3des    hspicecmidev
11:18:33 (snpslmd) hspicecmirt    hspicerf    identdebugger_cert
11:18:33 (snpslmd) sx_cdslink_ext    sx_daiclink    sx_data
11:18:33 (snpslmd) sx_jedatlink    synopsys_designware_tlm_lib synphony
11:18:33 (snpslmd) synphony_batch    synphony_cout    synphony_coutsl
11:18:33 (snpslmd) synphony_ipgen_asic synphony_ipgen_fpga synphony_msynth
11:18:33 (snpslmd) synphony_sacout synplifydsp_asic synplifydspsl
11:18:33 (snpslmd) synplifypremierdp vera_comp    vera_debug
11:18:33 (snpslmd) vera_rtime    xsim
11:18:33 (snpslmd)
11:18:33 (snpslmd) Licenses are case sensitive for TE_CATS
11:18:33 (snpslmd)
11:18:33 (snpslmd) EXTERNAL FILTERS are OFF
11:18:33 (lmgrd) snpslmd using TCP-port 36769
11:18:33 (snpslmd) Serving features for the following vendor names:
snpslmd    ACAD    adalmd    anagram    arcd    archprod    avantd    CADABRA    chrysa-
   lisd    cowared    EPIC    eved    everest    extremed    hscd    innologd    ISE-
   TCADd  knights  la_dmon  leda  magma  metasoftd  mwflexd  nassd  nsysnvs  numer-
   itchd     pdld     perflmd     riod     saber_dmn     sandwork     sclmgrd     sig-
   macd   slat   snpsOEM1   snpsOEM2   snpsOEM3   snpsOEM4   snslmgrd   ssilmd   syn-
   forad  synopsysd  synplctyd  TAVEREN  TE_CATS  tmald  vastlmd  vcsd
08/20/2014 11:18:34 (snpslmd) ----------------------------------------------------------
   -----------
08/20/2014 11:18:34 (snpslmd) Checking the integrity of the license file...
08/20/2014 11:18:34 (snpslmd) Valid SSS feature found.
08/20/2014 11:18:34 (snpslmd) The feature is needed to enable the other keys in your
   license file.
08/20/2014 11:18:34 (snpslmd) Licensed to Universidad del Valle de Guatemala
08/20/2014 11:18:34 (snpslmd) Siteid: 31759, Server Hostid: 180373237BDF, Issued on:
   7/16/2014
08/20/2014 11:18:34 (snpslmd) ----------------------------------------------------------
   -----------
11:18:34 (snpslmd) SLOG: Statistics Log Frequency is 240 minute(s).
11:18:34 (snpslmd) SLOG: TS update poll interval is not set.
11:18:34 (snpslmd) SLOG: Activation borrow reclaim percentage is 0.
11:18:34 (snpslmd) (@snpslmd-SLOG@) ===============================================
11:18:34 (snpslmd) (@snpslmd-SLOG@) === Vendor Daemon ===
11:18:34 (snpslmd) (@snpslmd-SLOG@) Vendor daemon: snpslmd
11:18:34 (snpslmd) (@snpslmd-SLOG@) Start-Date: Wed Aug 20 2014 11:18:34 CST
11:18:34 (snpslmd) (@snpslmd-SLOG@) PID: 3356
11:18:34  (snpslmd)  (@snpslmd-SLOG@)  VD  Version:  v11.12.1.0  build  146690  i86_lsb  (
   build 146690 (ipv6))
11:18:34 (snpslmd) (@snpslmd-SLOG@)
11:18:34 (snpslmd) (@snpslmd-SLOG@) === Startup/Restart Info ===
11:18:34 (snpslmd) (@snpslmd-SLOG@) Options file used: None
11:18:34 (snpslmd) (@snpslmd-SLOG@) Is vendor daemon a CVD: Yes
11:18:34 (snpslmd) (@snpslmd-SLOG@) Is TS accessed: No
11:18:34 (snpslmd) (@snpslmd-SLOG@) TS accessed for feature load: -NA-
11:18:34 (snpslmd) (@snpslmd-SLOG@) Number of VD restarts since LS startup: 0
```

Continuación de Cuadro 2.

```
+11:18:34 (snpslmd) (@snpslmd-SLOG@)
11:18:34 (snpslmd) (@snpslmd-SLOG@) === Network Info ===
11:18:34 (snpslmd) (@snpslmd-SLOG@) Socket interface: IPV6
11:18:34 (snpslmd) (@snpslmd-SLOG@) Listening port: 36769
11:18:34 (snpslmd) (@snpslmd-SLOG@) Daemon select timeout (in seconds): 1
11:18:34 (snpslmd) (@snpslmd-SLOG@)
11:18:34 (snpslmd) (@snpslmd-SLOG@) === Host Info ===
11:18:34 (snpslmd) (@snpslmd-SLOG@) Host used in license file: scluvg
11:18:34 (snpslmd) (@snpslmd-SLOG@) Running on Hypervisor: None (Physical)
11:18:34 (snpslmd) (@snpslmd-SLOG@) LMBIND needed: No
11:18:34 (snpslmd) (@snpslmd-SLOG@) LMBIND port: -NA-
11:18:34 (snpslmd) (@snpslmd-SLOG@) ================================================
```

E. Agregar la dirección del servidor de licencias a un usuario final

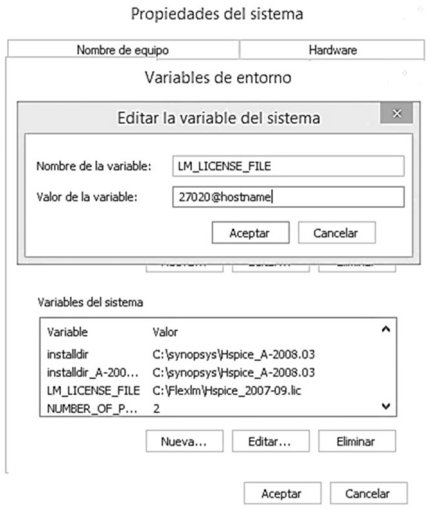
1. ***En CentOS.*** Para agregar la dirección del servidor de liciencias en CentOS, en el archivo `$HOME/.bashrc` agregar la siguiente línea.

```
LM_LICENSE_FILE=tcp_port@hostname; export LM_LICENSE_FILE
```

Con *tcp_port* y *hostname* el puerto TCP y el nombre de host que se configuró en el archivo de licencia con anterioridad. Si hay algún problema resolviendo el nombre del host del servidor se puede remplazar con la dirección IP del mismo. (Synopsys, Synopsys Licensing QuickStart Guide, 2014)

2. ***En MS Windows.*** Agregar la variable de entorno como se muestra en la siguiente figura:

Figura 31: Variable de entorno a servidor de licencias



Si hay algún problema resolviendo el nombre del host del servidor se puede remplazar con la dirección IP del mismo.

F. Aplicaciones a instalar para el correcto uso del ambiente de diseño.

1. Galaxy Custom Designer (instalar con Synopsys Installer).
2. iPDKs y librerías. (Ver la sección 4.8).
3. IC Validator. (instalar con Synopsys Installer).
4. Hspice. (instalar con Synopsys Installer).
5. StarRC. (instalar con Synopsys Installer).
6. Custom Explorer. (instalar con Synopsys Installer).
7. PyCell Studio. (Ver el siguiente cuadro). (Synopsys, PyCell Studio, 2014)

Cuadro 3: Instalación de PyCell Studio

a. Ir a la página: https://www.synopsys.com/TOOLS/IMPLEMENTATION/CUSTOMIMPLEMENTATION/Pages/pycell-studio.aspx y hacer click en *download now.*
b. Llenar los campos solicitados.
c. Mediante el correo electrónico registrado llegará un link de descarga, un usuario y una contraseña. Descargar el paquete.
d. Descomprimir el paquete en algún directorio temporal.
e. En la línea de comando navegar hasta el directorio temporal y ejecutar el comando:
```
./installer
```
f. Elegir un directorio para la instalación, por ejemplo: `/usr/synopsys/PyCell`
g. Agregar las variables de entorno necesarias con el comando.
```
$ source /usr/synopsys/PyCell/quickstart/bashrc
```
h. Agregar el comando anterior al archive $HOME/.bashrc.
i. Verificar la instalación.
```
$ cd $CNI_ROOT/quickstart
$ pyros
```
j. En el programa abierto
   - `Ir al menu: File > Open Design`
   - `Library: IPL_cni130`
   - `Cell: FastNmos`
   - `View: layout`
   - `Click Ok.`
k. En el panel de parámetros cambiar wf a 2.0 y hacer click en aplicar, se debe observar un cambio en la celda, esto nos asegura una instalación de PyCell Studio correcta.

G. Edición del archivo $HOME/.bashrc

Este es un archivo muy importante para la configuración inicial de nuestro ambiente de trabajo, en este archivo se agregan comandos importantes para que se agreguen las correctas variables de entorno al sistema operativo al inicio del mismo, estas variables son de extrema importancia para que las distintas aplicaciones se comuniquen entre sí, como se verá más adelante nuestra aplicación principal es Galaxy Custom Designer que es una plataforma unificada y necesita estas variables para poder funcionar correctamente. A continuación se presenta el archivo `$HOME/.bashrc` base, con el cual se han probado las aplicaciones, antes de usar las aplicaciones debe asegurarse que su archivo `.bashrc` esté en óptimas condiciones, el que se presenta es una base, este archivo se usa para muchas configuraciones, acá se muestra la configuración para el fin de este trabajo.

Cuadro 4: Archivo $HOME/.bashrc

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi

# User specific aliases and functions
# Se agrega la dirección del servidor de licencias

LM_LICENSE_FILE=27020@scluvg; export LM_LICENSE_FILE

# Variable para iPDK de Synopsys
export SAED32_28_PDK=/usr/synopsys/iPDK/SAED32_28_iPDK/
# Variable para IC Validator
export ICV_HOME_DIR=/usr/synopsys/IC_Validator
# Directorio de IC Validator
PATH=/usr/synopsys/IC_Validator/bin/AMD.64:$PATH
export PATH
#Generación de variables para PyCell Studio
source /usr/synopsys/PyCell/quickstart/bashrc
#Directorio del servidor de licencias
PATH=/usr/synopsys/11.8/linux/bin:$PATH
export PATH
#Directorio de Hspice
PATH=/usr/synopsys/HSPICE/I-2013.12-SP2-1/hspice/bin/:$PATH
export PATH
#Directorio de StarRC
PATH=/usr/synopsys/StarRC/bin/:$PATH
export PATH
# Directorio de CustomDesigner
PATH=/usr/synopsys/CustomDesigner/bin/:$PATH
export PATH
#Directorio de CustomExplorer
PATH=/usr/synopsys/CustomExplorer/bin/:$PATH
export PATH
```

H. Galaxy Custom Designer

Es una aplicación de Synopsys que provee una solución unificada para el diseño personalizado o basado en celdas de los *layouts* de un circuito en VLSI, Custom Designer provee la capacidad de simulación, análisis, extracción de capacitancias parásitas y la verificación física. Es importante mencionar que soporta el uso de kits de diseño de proceso interoperable (iPDK) que son paquetes que proveen las empresas dedicadas a hacer el proceso de fundición (*foundry*) donde están sus reglas de diseño, mecanismos para comparar el *layout* final con el diagrama esquemático y su forma de extraer parásitos. Synopsys, en su plan de capacitación universitaria provee un muy útil PDK, que puede ser usado para investigación, experimentación y capacitación, aunque no recomiendan usarlo para la fabricación final. Este PDK está avalado por la TSMC que le da un peso significante a la librería para usarla como medio de aprendizaje. Estas características aumentan la productividad en cualquier ámbito en el que se use Galaxy Custom Designer. (Synopsys, Custom Designer, 2014)

1. Instalación del iPDK

a. Ir a la página http://www.synopsys.com/apps/protected/university/members.html que ofrece información a las instituciones educativas asociadas con Synopsys, también material de estudio y experimentación, entre ellos las librerías que se usarán para el diseño del sumador restador. (Synopsys, 32/28nm Interoperable PDK, 2014)
b. En la sección de *Libraries* descargar las librerías que se quieran usar. En este caso se usaron las de 32 a 28 nanómetros usando el enlace *32/28nm Generic Library Download*.
c. Después de llenar una pequeña encuesta para estadísticas de Synopsys y aceptar los términos de licencia se tiene acceso a la descarga de los paquetes.
d. Conforme a la experiencia se recomienda descargar los paquetes desde el servidor de licencias usando la línea de comandos como lo muestran las instrucciones en la siguiente tabla, esto debido a que usando la interfaz web se tiende a rechazar la conexión al servidor FTP o a terminar las descargar antes de ser completadas.
e. Descargar todos los archivos de la lista.

Cuadro 5: Descarga desde servidor FTP

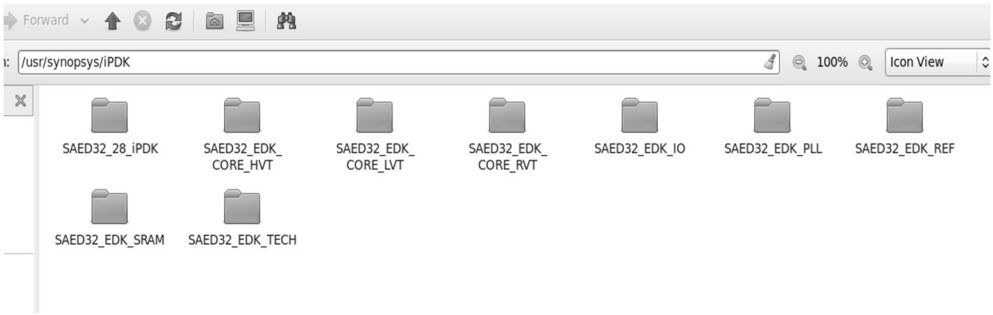
1) Iniciar session en "ftp.synopsys.com" eg.
   $ ftp ftp.synopsys.com
2) Ingresar usuario y contraseña de SolvNet.
3) Si no está en modo pasivo ingrese usando el commando "passive".
4) Ingrese "binary" en el ftp prompt para establecer el modo de transferencia a binario. ftp> binary
5) ftp> hash (opcional, muestra # en proceso de descarga).
6) ftp> cd cafe
7) ftp> cd 32-28nm_EDK_07292013
8) ftp> Escriba "ls" o "dir" para ver la lista de archivos.
9) ftp> get + nombre de archivo.

Si no es posible conectarse con el servidor ftp puede usar algunos de los alternativos según la ubicación geográfica.

- US - marley.synopsys.com (198.182.44.204)
- Europe - tosh.synopsys.com (149.117.30.1) o fireball.synopsys.com(198.182.37.221)
- Asia - hu01estftp.synopsys.com (84.2.38.58)

f. Descargar el iPDK en la sección “iPDKs” de la misma forma.
g. Descomprimir todos los paquetes obtenidos en alguna carpeta, por ejemplo en `/usr/synopsys/iPDK` la siguiente figura mostrará el resultado final.

Figura 32: Directorio de iPDK de Synopsys



La carpeta iPDK contiene todos los archivos sobre la tecnología de las librerías, modelos de Hspice, archivos de mapeo, reglas de diseño y archivos para la evaluación física del diseño y también para extracción de parásitos de un *layout*. Para que Custom Designer reconozca esta librería es necesario instalarla en nuestro SO.

h. Agregar la variable de entorno

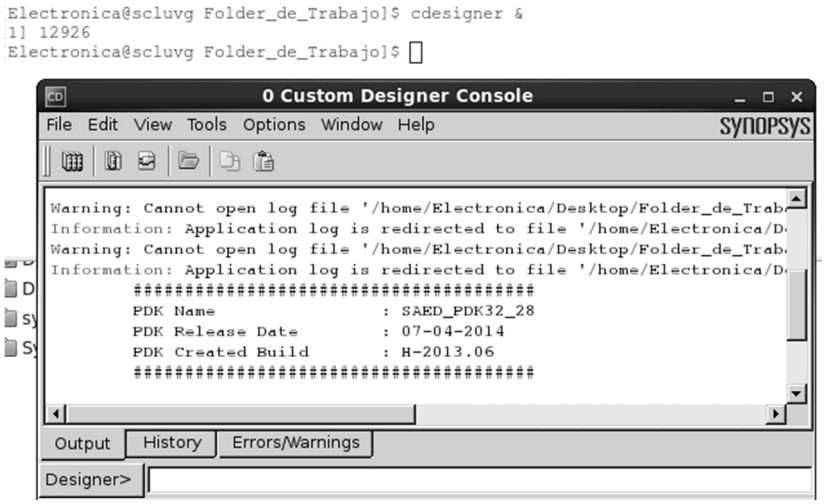
```
export SAED32_28_PDK=/usr/synopsys/iPDK/SAED32_28_iPDK/
```

Nota: La dirección `$SAED32_28_PDK` debe ser una dirección válida.

i. Copiar los archivos de `/usr/synopsys/iPDK/SAED32_28_iPDK/install` a la `$HOME`.
j. En este punto ya se puede iniciar el Custom Designer en algún folder que se use como “folder de trabajo”, puede iniciarse desde cualquier ubicación, sin embargo, se recomienda usar un sólo folder ya que la cantidad de archivos generados es grande y es mejor tenerlos en una única ubicación. Navegar desde la línea de comandos hasta el folder de trabajo y ejecutar el siguiente comando el cual mostrará una ventana (consola de Custom Desgner) similar a la de la Figura 33.

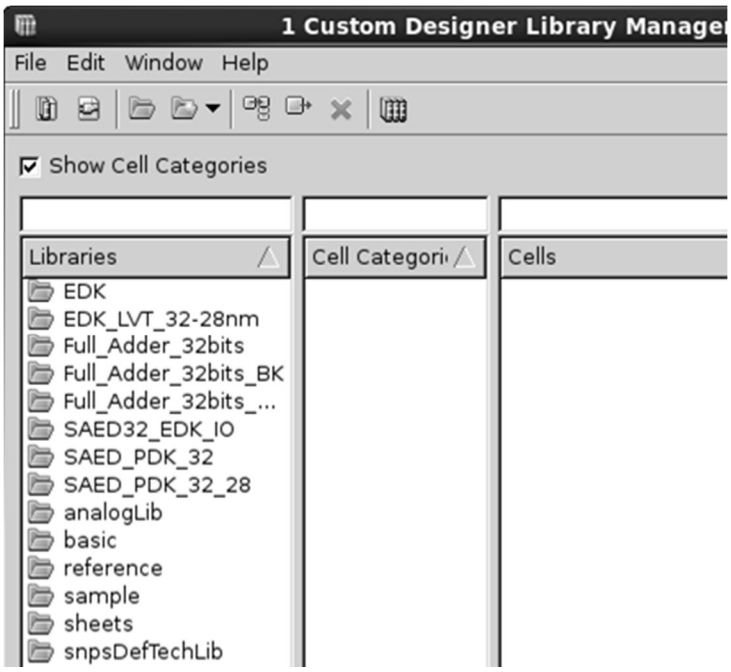
```
$ cdesigner &
```

Figura 33: Custom designer console



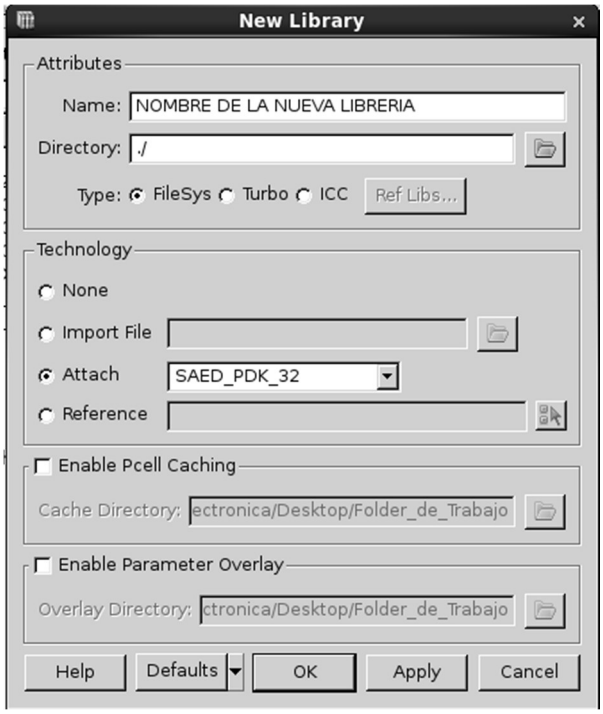
Y al abrir el *Library manager* deben aparecer nuestras librerías, en este punto sólo aparecerá la *SAED_PDK_32_28*. (Synopsys, 32/28nm Interoperable PDK, 2014)

Figura 34: Custom designer library manager



k. Para agregar las librerías descargadas primero se crea una librería donde se alojarán los datos nuevos, agregando a esta nueva librería la librería *SAED_PDK_32_28*.

Figura 35: Agregar una librería nueva



l. En la consola de Custom Designer
File>Import>{stream | LEF | Netlist}
- *Stream* es para agregar *layouts*, mediante un archivo extensión *gds*.
- LEF para agregar un *abstract* que es una parte del layout, mediante un archivo extensión *lef*.
- *Netlists* para agregar archivos de texto con la definición de *decks.*
  - CDL agrega *decks.*
  - HSPICE agrega *decks* con extracción de parásitos.
  - VERILOG agrega un archivo que describe algún circuito.

m. La librería de destino será la librería que se creó en el paso 11. Hasta este punto tenemos todo configurado para empezar con las fases de diseño de nuestro circuito.

2. Fases de diseño

a. Realización de circuito esquemático. El diseño comienza al nivel de conexión de los diferentes componentes en un diagrama esquemático, cada componente debe tener sus propiedades bien editadas para que las siguientes fases muestren resultados fiables. Un diagrama esquemático es la conexión de los componentes con sus propiedades físicas correctas. Después de tener un diseño esquemático terminado se puede crear un *netlist* para la simulación y evaluación del desempeño, cuando el desempeño es correcto se puede proceder a la realización del *layout*. (Synopsys, Custom Designer, 2014)

El editor esquemático es muy pa recido a los programas que se usan para la simulación de circuitos, tiene las características de orientación (o transformación) que gira o refleja los componentes para hacer más entendible el diagrama. También está el modo *Infix*, que facilita la edición de los elementos cuando se usa algún comando, cuando se usa este modo se realiza un click implícito que su función depende del comando que se use. Por ejemplo puede ser un punto de referencia para dibujar un rectángulo o un segmento de línea, o el eje de rotación de algún elemento.

Lo primero que se debe hacer, al momento de empezar a usar el editor esquemático es elegir nuestras preferencias en el programa, qué se quiere visualizar y qué no, establecer las opciones de diseño que son puntos como espaciado entre elementos, grosores de cables, prefijos de los nombres de las rutas etc. Todo esto se hace desde el menú **Options>{Display | Design | General}**. Algo que es muy recomendable es editar la visualización de las etiquetas que se mostrarán, esto se hace en **Edit>properties>Manage Labels**. Después de poner a punto el programa donde se estará trabajando se puede proceder a la creación del esquemático. Algunos conceptos se explican a continuación.

Una instancia es una representación de un diseño o de un objeto dentro de otro diseño, se pueden tener varias instancias de un mismo diseño u objeto, pero deben tener diferentes nombres. Para agregar una instancia vaya a **Add>Instance** y elija las opciones y los parámetros de la instancia, luego en el área de trabajo elija el lugar donde colocar la instancia. Después de estas acciones la instancia estará colocada en el lienzo de trabajo, tome encuentra que el comando sigue activo para agregar una instancia igual, puede salir del comando con la tecla *Esc*.

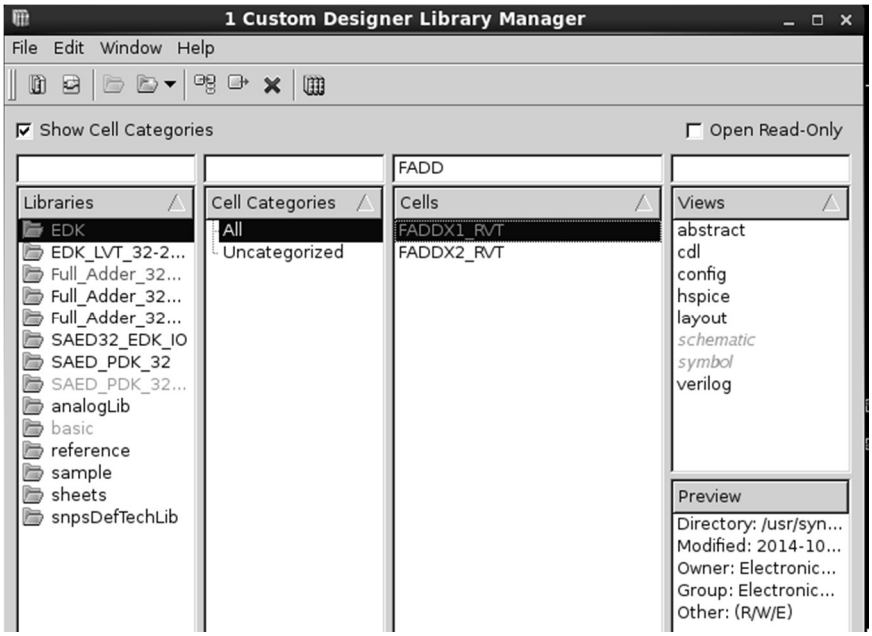
El agregar conexiones físicas (alambres) entre los dispositivos es una tarea trivial, sin embargo, hay que tomar en cuenta que para Custom Designer todas las rutas tienen un nombre para ser identificadas, será una tarea específica cuando se cambie este nombre, por lo general se deja el nombre agregado por defecto.

Para agregar un alambre **Add>Wire**, seguidamente especificar la opciones de la conexión y elegir el punto de inicio y fin.

Para un diseño más ordenado y más profesional se recomienda realizar bloques, al estilo de las conocidas cajas negras, donde se tengan entradas y salidas, conocer la función que los relaciona pero no forzosamente tener que saber qué hay dentro, esta es una buena técnica para reciclar trabajos antiguos. Las entradas y salidas se hacen mediante pines que se agregan en **Add>Pin**, luego especificar las opciones y parámetros del pin y agregarlo en alguna parte del lienzo de trabajo.

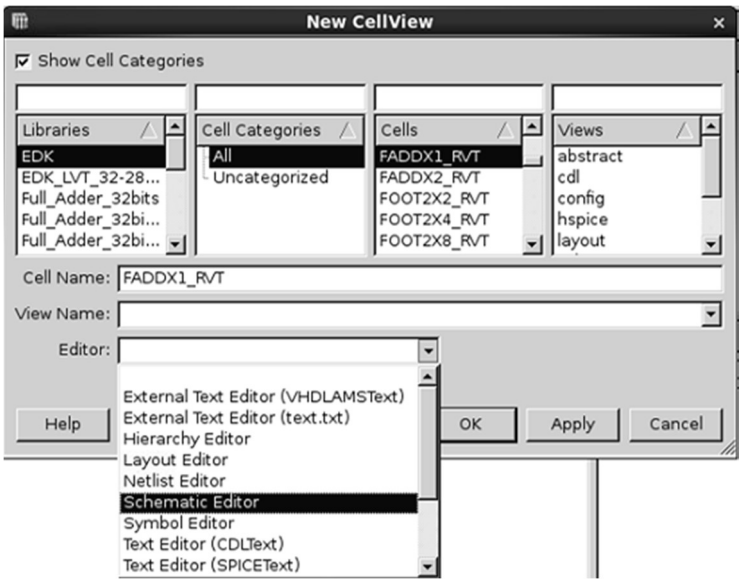
Cuando se han agregado las librerías a Custom Designer, se muestra cada una de ellas en el Library Manager, la iPDK instalada posee una gran cantidad de compuertas y componentes comunes como sumadores, sumadores completos, flip-flops, multiplexores, etc. Cada uno viene con un *deck* para simulaciones (CDL), un *deck* para simulaciones tomando en cuenta parásitos (Hspice), *layout* y un archivo de descripción de hardware en *Verilog*. Algo que se omite es la agregación de un esquemático, aunque no es crítico tenerlo ya que con los *decks* se puede simular y caracterizar al circuito, se mostrará a fin de ejemplo como se desarrolla una caja negra para un sumador completo de 1 bit a nivel de diagrama esquemático. Esta caja negra es una copia idéntica del circuito descrito en el *netlist* CDL. (Synopsys, Custom Designer, 2014)

Figura 36: iPDK agregado a Custom Designer



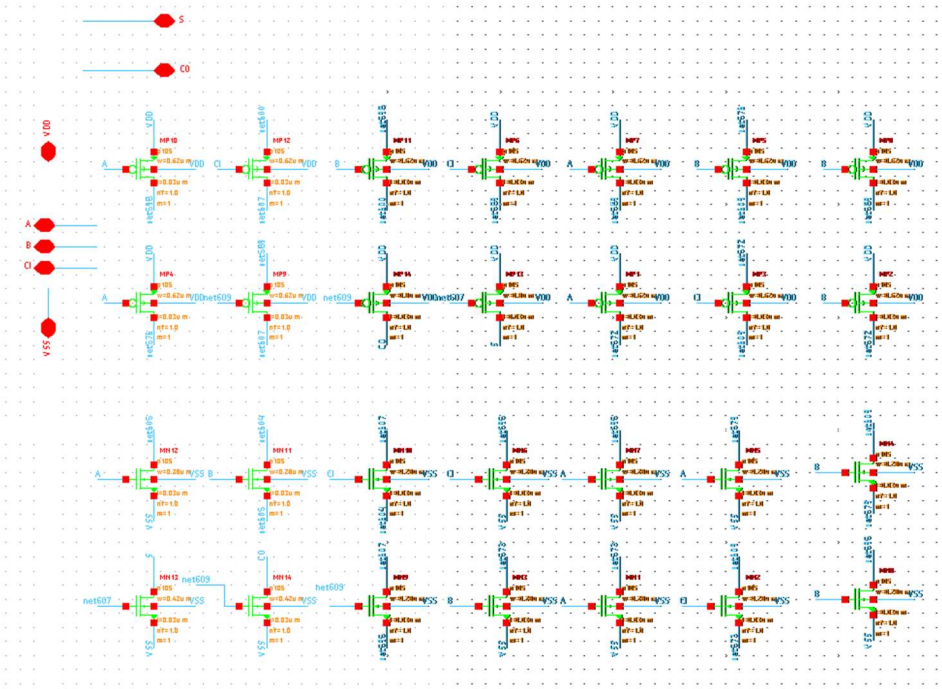
Lo primero que se hace es ubicar y seleccionar la librería en el *Library Manager*, seleccionar la categoría a la que pertenece nuestra celda, seleccionar la celda. Hacer click derecho en el espacio de *views* y seleccionar nuevo. Aparecerá la siguiente pantalla.

Figura 37: Elaboración de la vista Schematic



En editor elegir *Schematic Editor*, asegurarse que la librería, la categoría y que la celda sean las correctas y hacer click en OK. Inmediatamente se abrirá el editor esquemático en el cual se deben agregar los transistores que se indican en el CDL y conectarlos entre sí. En circuitos con muchos transistores es recomendable usar etiquetas y no conectar mediante *wires* cada una de las instancias, se podría volver muy complejo.

Figura 38: Interconexión de instancias

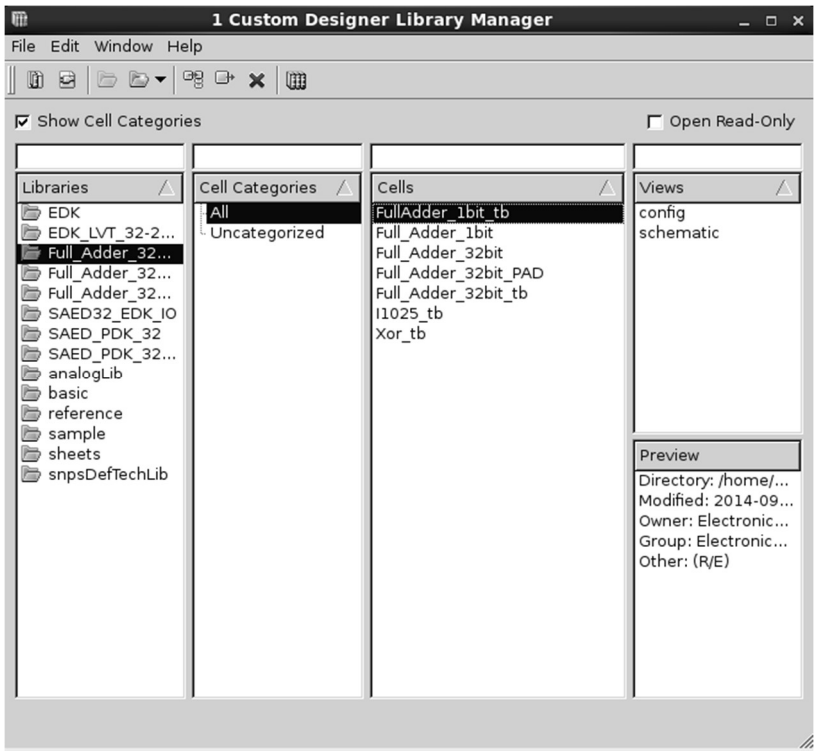


Seguidamente se crea un símbolo, una caja negra que representará al sumador en cualquier otro lienzo de trabajo en el que se agregue su instancia. Para el efecto se agrega una nueva vista, de la misma forma que se agregó la vista del esquemático con la diferencia que se selecciona *Symbol Editor*. Automáticamente se abre el editor correspondiente y una proposición de símbolo por parte de Custom Designer, se puede cambiar de tamaño, cambiar el lugar de los pines al gusto del diseñador.

b. Simulación. Una vez que ya se tiene el esquemático y el símbolo agregado a la vista de nuestra celda es necesario saber si funciona correctamente. Recordar que el símbolo hace referencia al conjunto de transistores que componen el circuito y es una forma compacta de verlo, si hay algo no esperado en los resultados de simulación el problema estará en los transistores y no en el símbolo a menos que sea un problema que falte algún pin, se debe actualizar el símbolo. (Synopsys, Custom Designer, 2014)

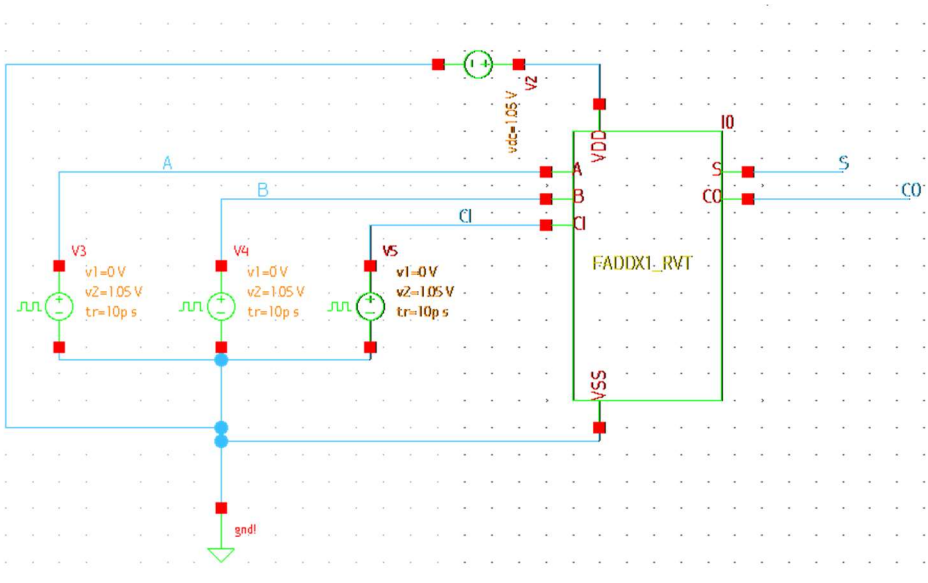
La forma correcta de hacer una simulación es hacer otra celda para el *testbench*, esto porque no se puede instanciar una celda dentro de sí misma. Se muestra un ejemplo en la siguiente figura.

Figura 39: Celda para testbench



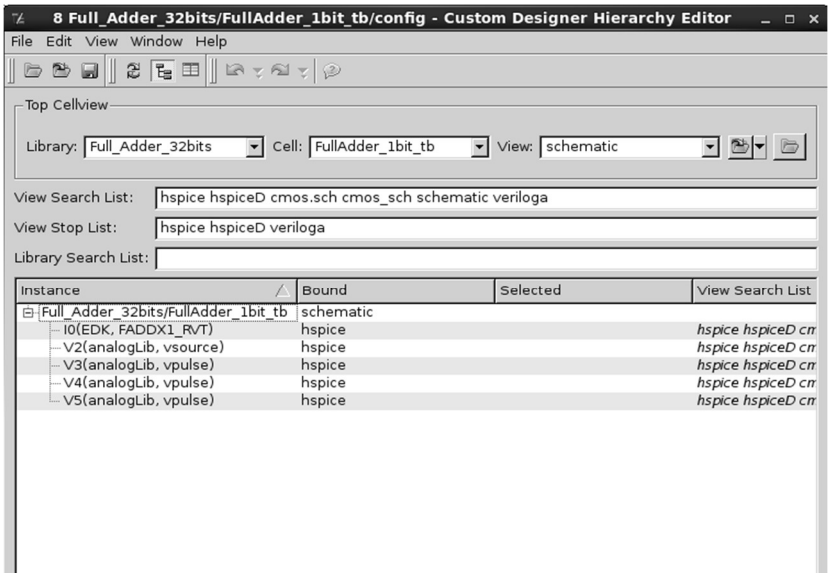
Resta explicar qué contendrá entonces las vistas *schematic* y *config* de esta celda para *testbench*. En la vista de esquemático se agrega el símbolo anterior con fuentes de alimentación, fuentes de pulsos, resistencias y todo lo demás que se necesite para poder inyectar al circuito las señales necesarias para comprobar su funcionamiento.

Figura 40: Esquemático para realización de pruebas



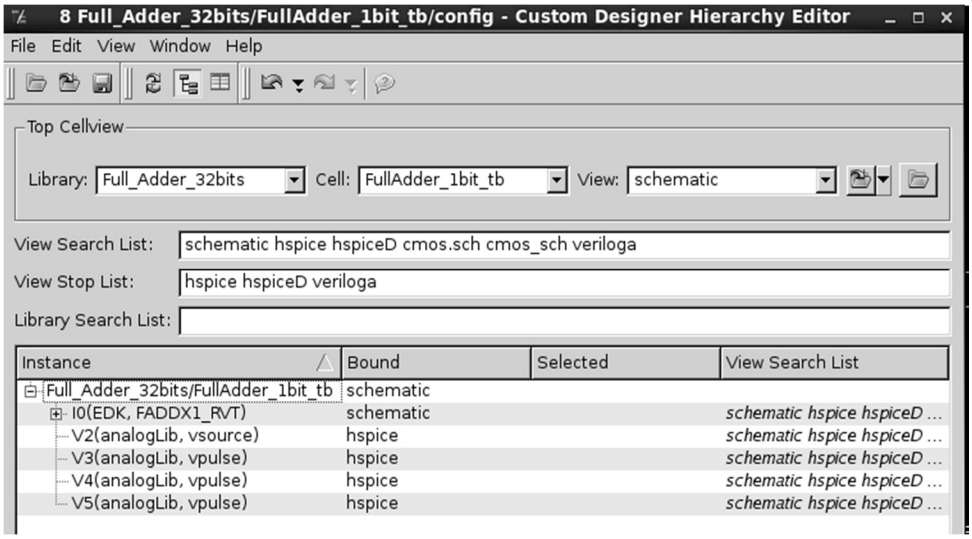
El siguiente paso es ir a **Tools>Hierarchy Editor** se abrirá la ventana del editor de jerarquía que es un orden el cual tomará el simulador en cuanto a la vista que tomará para hacer la simulación. Como primer verificación se asegurará que la librería, la celda y la vista sean las que se quieren configurar. (Synopsys, Custom Designer, 2014)

Figura 41: Configuración de la búsqueda jerárquica entre vistas 1



En el apartado *View Search List* se deben agregar las listas en el orden que el simulador las buscará para hacer la labor. En la figura anterior es **<hspice hspiceD cmos.sch cmos_sch schematic veriloga>** y en *View Stop List* que es donde se parará de buscar **<hspice hspiceD veriloga>**. Se pueden ver los resultados de las vistas que el simulador usará en la parte de abajo, en la columna *Bound* se puede ver en este ejemplo que el simulador usará el *netlist* con parásitos de Hspice, si se desea usar la vista del esquemático basta con colocar la palabra *Schematic* de primer lugar en la jerarquía de búsqueda.

Figura 42: Configuración de la búsqueda jerárquica entre vistas 2

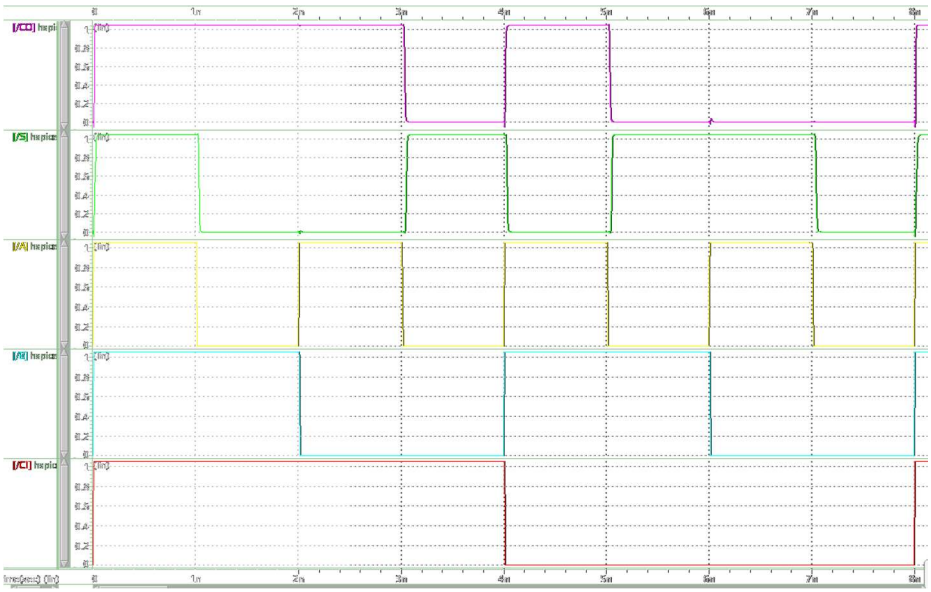


La figura muestra el cambio mencionado, notar que las fuentes de alimentación y excitación del circuito siguen usando a Hspice para hacer la simulación, esto es así normalmente para las fuentes de tensión ya que no hay un esquemático en las librerías instaladas que definan su funcionamiento. Al guardar estos cambios se crea la vista *config* que es básicamente eso, configuración del *netlist* que se usará para la simulación. Se procede a abrir el esquemático del testbench, ir a: **Tools>SAE**. *(Simulation Analysis Enviroment).* En la ventada de SAE lo primero que se hace es configurar la simulación en el menú Setup.

Cuadro 6: Configuración de SAE

Menú	Configuraciones
Setup>Design	Verificar la librería, celda y vista (*config*) que se desea simular.
Setup>Simulator	Se elige el motor de simulación que se usará. Normalmente se elige Hspice.
Setup>Enviroment Option	Se configura el entorno en que se hará la simulación, normalmente no hay que modificar nada, pero si se tiene algún problema con que no se encuentra el simulador elegido intentar deseleccionar la opción de simulación en 64-bits.
Setup>Analyses	Seleccionar los tipos de análisis que se desean realizar durante la simulación, por ejemplo: Análisis en el tiempo, puntos de operación, DC etc. Se puede elegir uno o más a la vez.
Setup>Model Files	Es quizá la más importante de las configuraciones. En esta sección se agregan los archivos que contienen los modelos para la simulación. Sin estos modelos la simulación no se puede llevar a cabo. El archivo de modelos de la tecnología usada se encuentra en el directorio de instalación de la librería. En este caso se encuentra en: `/usr/synopsys/iPDK/SAED32_28_iPDK/hspice`. Primero se debe agregar el directorio en *Include Path* y luego el archivo en sí en la sección *Model File* también es necesario especificar la sección del modelo que se va usar, un archivo de modelos normalmente las secciones TT, FF etc. Que sirven para especificar de mejor forma el transistor que se va a usar.
Output>Select in Design	Se abre el esquemático del *testbench* y mediante un click se selecciona la señal que se desea analizar. Cuando se termine de seleccionar las señales deseadas presionar la tecla *Esc*.
Simulation>Netlist and run	Se crea el *deck* y empieza Hspice (o el simulador elegido) a adquirir los datos para mostrar los resultados, cuando termina se abre WaveView y se muestran las gráficas, esto en caso de haber selecciona una análisis transiente (que es el tipo de análisis más común).

Figura 43: Resultados de simulación

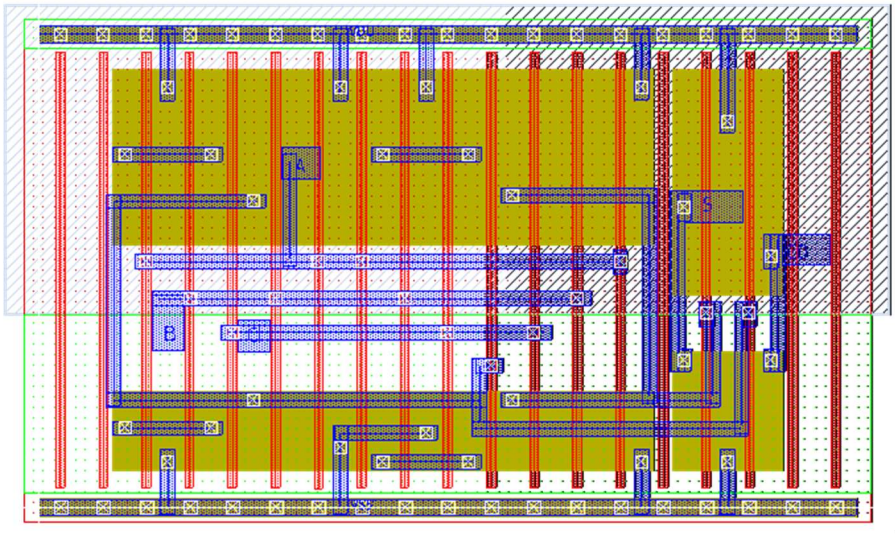


c. Análisis. La etapa de análisis es una etapa donde se debe verificar que los datos obtenidos son los esperados, en este punto es necesario estar seguros que todo funciona correctamente, que los niveles de voltaje en salidas y entradas son correctos etc. No es necesario en esta etapa analizar tanto en aspectos como potencia, energía, tiempo o alguna otra variable muy específica ya que en este punto no se tiene un circuito final porque faltan los parásitos que generará el Layout. (Synopsys, Custom WaveView™, 2014).

Existen varias aplicaciones para analizar los datos obtenidos como Custom WaveView, Custom Explorer, Custom Explorer Ultra y CosmosScope. En la última fase de diseño se aprenderá a hacer un análisis más extenso para tener datos específicos del circuito, ahora como ya se ha mencionado, lo importante es que el circuito funcione correctamente. (Synopsys, Custom WaveView™, 2014)

d. Schematic-Driven Layout. Esta fase de diseño es una fase crucial en el proceso, en esta fase se deja de tener un esquemático que puede que sea muy común para muchos. En esta fase de diseño se realiza la implementación del *layout* que será el corazón del proyecto que se realiza, un buen diseño del *layout* garantiza menos potencia, menos energía, tiempo de respuesta menor. En el mundo de VLSI el diseño puede tener miles o hasta millones de transistores, no es nada eficiente realizar el diseño transistor por transistor, esta opción podría llegar a ser lo mejor en cuanto a desempeño pero llevará varios días, meses, años, dependiendo del proyecto, es por eso que se usa un diseño en base a celdas, a librerías de circuitos comunes ya realizadas que se colocan estratégicamente para lograr una mayor densidad de transistores por unidad de área, luego de colocar las celdas se procede al que une a cada una de las celdas y a los puertos de entrada y salida. A continuación se muestra la celda del sumador completo que se ha estado usando como ejemplo. Estos *layout* se agregan cuando se importa el archivo *gds*, que se explicó en la instalación de la iPDK. (Synopsys, Custom Designer, 2014)

Figura 44: Celda estándar de diseño



El proceso de generar un *layout* a partir de un diagrama esquemático se denomina "Schematic-Driven Layout" y se inicia abriendo el esquemático en el menú **Tools>SDL** se abre en seguida una ventana donde se indica que se generarán dos vistas adicionales, una denominada *layout* y otra denominada *layout.config* la primera contendrá el *layout* y la segunda la configuración del mismo, verificar que la librería y la celda de destino sean correctas y hacer click en OK. Se abre el editor de *layout* vacío. Centrándose en el editor de *layout* se tomarán en cuenta las siguientes configuraciones o revisiones.

Cuadro 7: Configuración de Layout Editor

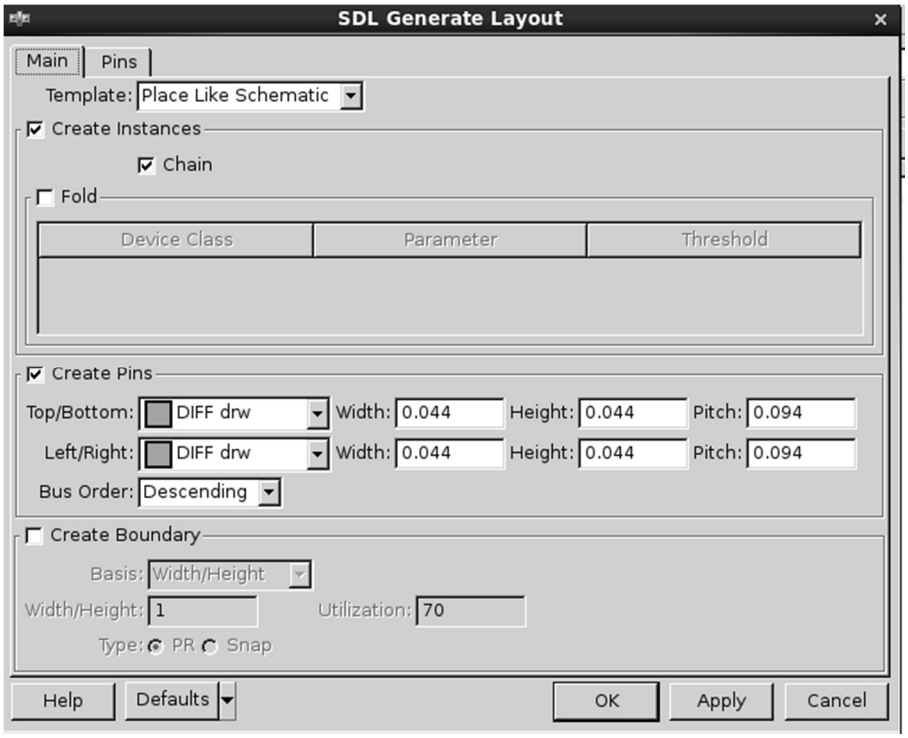
Menú	Configuraciones
Tools>Constraint Manager	Se encuentran todas las restricciones que se tomarán en cuenta al momento de hacer algún ruteo, sea automático o sea manual. En otras palabras, son todas las reglas de diseño que se deben cumplir. Las distancias o cualquier regla se pueden cambiar, sin embargo, esto no es recomendable si estamos usando un iPDK ya que están elaborados por las empresas *foundry* y se debe cumplir para una manufacturación correcta.
Conectivity>Define Terminals	Al terminar un *Layout* aquí deben aparecer todos los pines/puertos que se agregaron al esquemático, así como la dirección del mismo y la *net* a la que pertenecen.
Options>display	Si se quiere cambiar alguna configuración de visualización.
Options>Design	Contiene todas las opciones que se toman en cuenta para realizar el diseño, como ángulos de pistas, nivel de restricciones, opciones de espaciado entre las instancias en la rejilla cuando se selecciona automáticamente el posicionado etc.
SDL>Options	En la pestaña *pins*, habilitar que se agreguen las etiquetas de las terminales, agregarlas en la capa MxPIN drw, donde x es la capa donde se agregarán los pines. Es importante para la iPDK que Synopsys provee etiquetar los pines para que el proceso de LVS sea exitoso.

El proceso continúa con la colocación de las instancias en el lienzo del editor de *layout*, esto se puede hacer de forma automática **SDL>Generate Layout** o de forma manual **SDL>Pick and Place**.

La forma de decidir entre qué método es mejor depende de muchos factores como las configuraciones hechas en el programa y la complejidad del proyecto, se puede tener la mejor de las configuraciones y un complejo proyecto que el programa tardaría demasiado en poder posicionar cada instancia sin olvidar que luego es necesario realizar el ruteo. Lo recomendable es seguir aquel dicho que usan los programadores: "divide y vencerás" no es nada recomendable tomar un diseño de miles de miles de transistores o celdas y esperar algún milagro del programa diseño. Se recomienda hacer una división en módulos del diseño y realizar una combinación estratégica del *Generate Layout* y el *Pick and Place*. (Synopsys, Custom Designer, 2014)

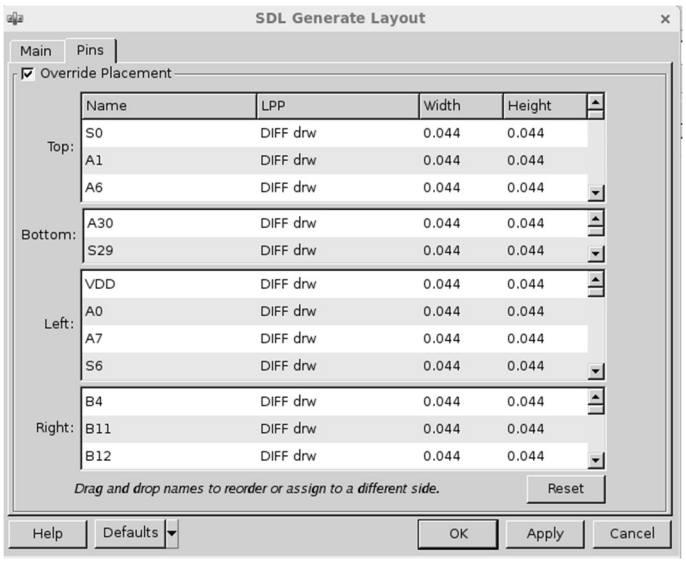
Usando **SDL>Generate Layout** se abre una ventana donde se puede decidir que se posicionará y de qué forma, en qué lugar y qué elementos simplemente se ignoran.

Figura 45: Generación de layout



Si la opción *Create instances* está seleccionada la posición de las instancias se modificará o se agregarán las que no se encuentren en el *layout editor*, si no está habilitada no se modifica la posición o la agregación de las instancias. La opción *Create Pins* añadirá las terminales correspondientes al diseño, en la capa, con las dimensiones y con el espaciado *(Pitch)* seleccionado. La opción *Create Boundary* agrega un rectángulo que envuelve a todo el *die*, y sobre éste se agregan las terminales en el orden y posición especificada en la pestaña *Pins* para el efecto la opción *Override Placement* debe estar seleccionada (Ver la siguiente imagen).

Figura 46: Ubicación de terminales



La opción **SDL>Pick and Place** es una opción interactiva de tomar mediante un click la instancia del editor de esquemático y posicionarla en el editor de *layout*.

Figura 47: Modo Pick and Place

La barra aparece cuando se está en el modo *Pick and Place* y muestra algunas opciones básicas para la agregación de las instancias o pines. Para configuraciones más detalladas se usa el botón *Eject* de esta barra.

En el momento en que estén todas las instancias y terminales agregadas se procede a rutear todas las *nets* en **Tools>Galaxy Custom Router**. En el menú **GCR** aparecen las opciones asociadas a este proceso, se pueden agregar *Shields* a alguna ruta, esto es agregar alguna ruta de VSS o VDD para evitar el *crosstalk* que puede ocasionar la conmutación de otra ruta. Para iniciar el modo de autoruteo ir a **GCR>Auto Route**

Figura 48: Modo Auto Route

Seleccionar rutas específicas o todas las rutas y hacer click en el botón *Route*, el proceso de auto ruteo se iniciará y la duración dependerá de la complejidad del proyecto.

Figura 49: Proceso de ruteo

Al momento de terminar de rutear se tendrá un *layout* donde únicamente faltará realizar las verificaciones correspondientes en cuanto a reglas de diseño y comparación con el esquemático para su completa finalización. Esto se hará en la siguiente fase de diseño.

e. DRC y LVS. En esta fase de diseño se realiza la verificación de las reglas de diseño *(Design Rules Checking)* y la verificación física del diseño *(Layout Vs. Schematic)*. Esta fase es crucial para lograr un diseño que pueda ser fabricado, es necesario cumplir con las reglas de diseño para que la empresa de manufactura pueda implementar lo requerido y es necesario que el esquemático, que ya se sabe que funciona correctamente por la fase de simulación y análisis, sea idéntico a lo implementado en el *layout*. Ambos procesos son bastante sencillos pero en ocasiones podrá requerir de mucha paciencia para solucionar cualquier violación encontrada. (Synopsys, IC Validator, 2014)

Para el proceso DRC vaya al menú **Verification>DRC>Setup and Run** en el siguiente cuadro se explican las configuraciones y los aspectos que se deben tomar en cuenta.

Cuadro 8: Configuración de DRC

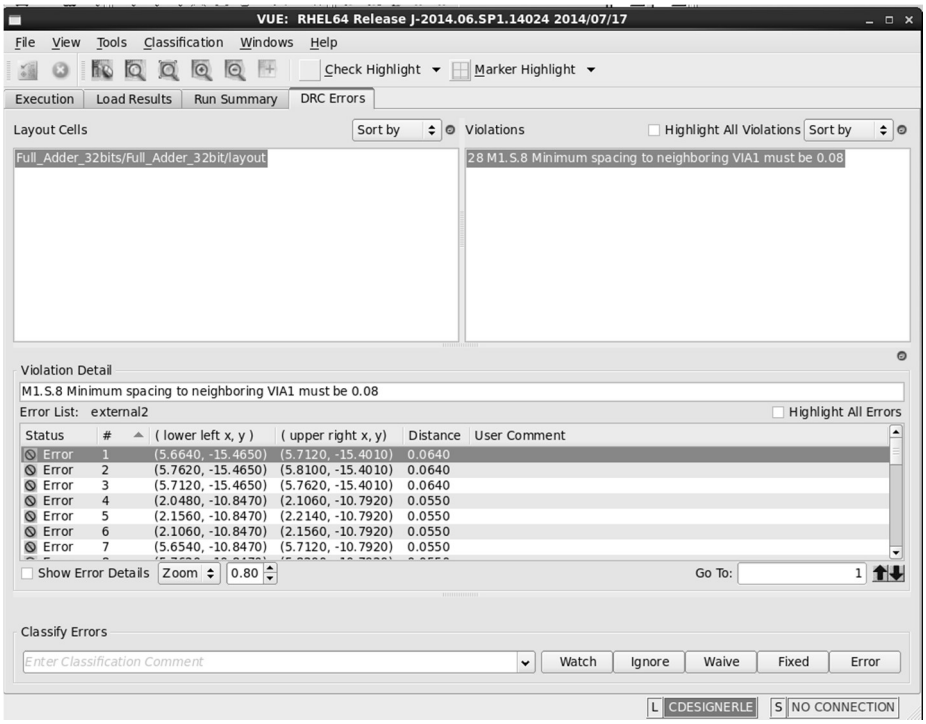
Sección	Explicación
Pestaña Main	
Run Dir.	Un directorio donde se guardarán todos los resultados, logs y archivos necesarios para el proceso DRC.
Format.	Galaxy custom Layout trabaja con el formato OpenAccess, normalmente se elige éste, pero no debiera haber ningún inconveniente en realizar el proceso con formato Stream.
Library, Cell y View	Verificar que la librería, la celda y la vista sean las correctas para ejecutar el proceso.
Tool	La herramienta que se usará para realizar el proceso, actualmente se usa con frecuencia IC Validator pero existen otras herramientas como Hercules.
Runset	Es el archivo que contiene las reglas de diseño y cómo estas se manejan, normalmente cuando se configura un iPDK las rutas a estos archivos se agregan por defecto. Si existiera algún problema con la agregación, el archivo se ubica en: `/usr/synopsys/iPDK/SAED32_28_iPDK//icv/drc/saed32nm_1p9m_drc_rules.rs` Que puede variar según el directorio donde se hayan instalado los archivos de las librerías agregadas.

Continuación de Cuadro 8.

View Output	Al finalizar el proceso se abre una ventana donde se muestran los *logs* del proceso y errores encontrados.
Pestaña Custom Option	
Layer Map	Es el archivo de la tecnología que se usa, en él se especifican aspectos importantes sobre las distintas capas de metal. Se ubica en: `/usr/synopsys/iPDK/SAED32_28_iPDK//techfiles/saed32nm_1p9m_gdsout.map`

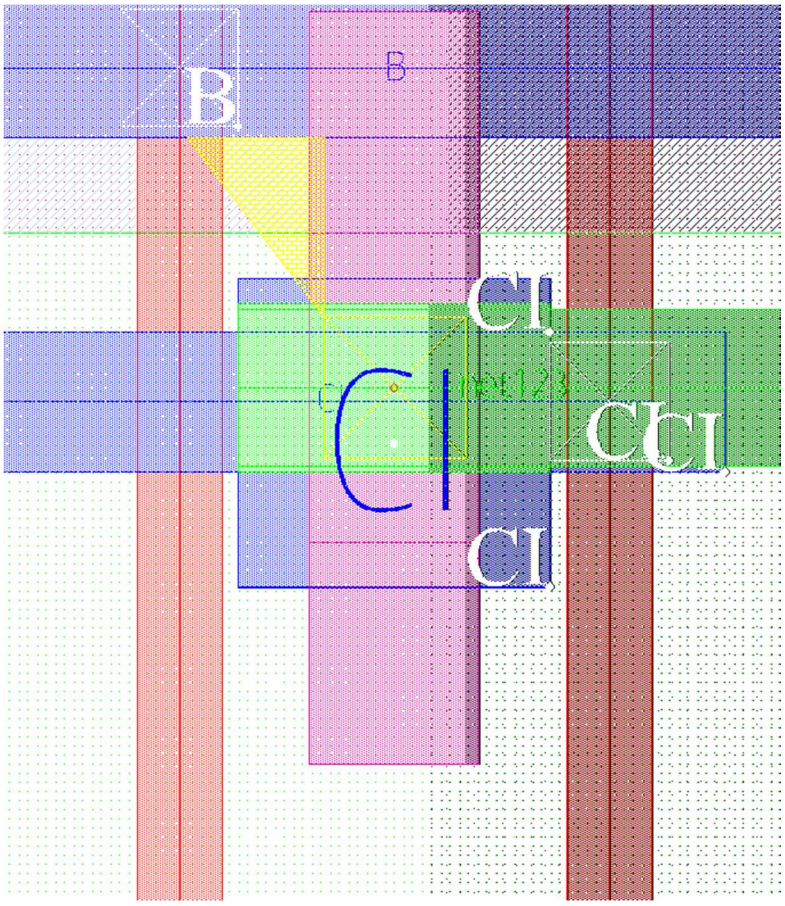
Cuando la configuración se ha terminado se procede a dar click en OK. El proceso comenzará y luego mediante una ventana de la aplicación que llevó a cabo el proceso (IC Validator, Hercules) se notificará las violaciones encontradas. (Synopsys, IC Validator, 2014)

Figura 50: Verificaciones DRC



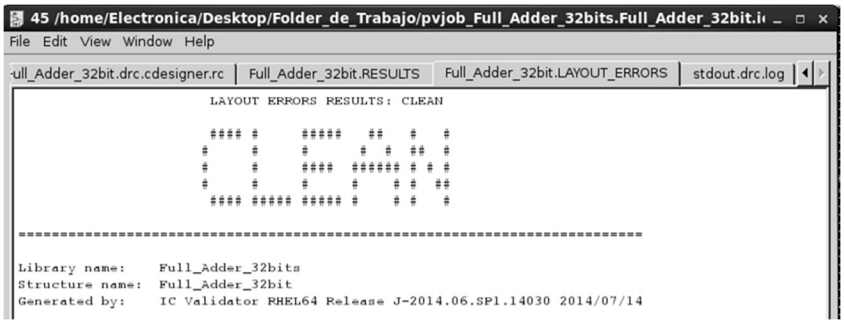
Las violaciones encontradas son fáciles de ubicar y reparar manualmente debido a la interacción que se tiene mediante el sistema operativo, IC Validator o Hercules y Custom Designer. Cada vez que se resalta un error el editor de *layout* se magnifica en las coordenadas donde se encontró la violación. Por ejemplo la siguiente violación encontrada, mostrada en la siguiente figura, muestra una violación en cuanto a la distancia que debe haber en una VIA de capa 1 a capa 2 de metal hacia una pista de metal capa 1.

Figura 51: Error identificado en proceso DRC



La marca amarilla muestra el lugar exacto de la violación, en este caso una violación de distancias como ya se ha explicado. El número de tipos de errores que puede ocurrir es muy extenso, sin embargo, por la interactividad de las aplicaciones será fácil ubicarlos y corregirlos. El resultado satisfactorio debe lucir algo como:

Figura 52: DRC exitoso



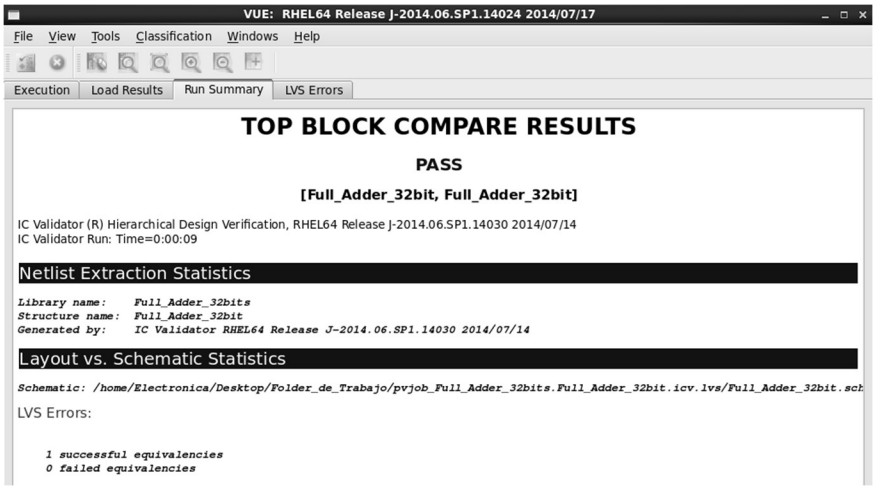
Para la ejecución de LVS vaya a **Verification>LVS>Run and Setup**, el proceso de configuración es muy similar al de DRC. En el siguiente cuadro se especifican algunos detalles.

Cuadro 9: Configuración de LVS

Sección	Explicación
Pestaña Main	
Run dir	El directorio donde se ejecutará el proceso y se guardarán todos los resultados y *logs* del mismo.
Layout y Schematic	Especificar las librerías, las celdas y las vistas tanto del *layout* y del esquemático que se van a comparar.
Tool	La aplicación que se usará para ejecutar LVS.
View Output	Para ver los *logs* y demás resultados al terminar el proceso.
Netlisting Option	
Schematic Netlist	Dirección del *deck* que se extraerá del esquemático, normalmente es una ubicación dentro del Run dir.
Netlister	Formato de salida del *deck*.
View Search List y View Stop List.	Configuración de la búsqueda jerárquica en las vistas para la generación del *deck*.
Pestaña Custom Option	
Layer Map:	Al igual que el DRC, especifica la ubicación del archivo de tecnología de las diferentes capas de metal. Ubicado en: `/usr/synopsys/iPDK/SAED32_28_iPDK//techfiles/saed32nm_1p9m_gdsout.map`

El sistema interactivo también hará fácil el poder ubicar un error y corregirlo, de la misma forma que con el DRC. Un proceso exitoso si errores se mostrará con el siguiente resultado.

Figura 53: LVS Exitoso



Cuando se logren ambas verificaciones exitosas se tendrá un *layout* que puede ser fabricado por el *foundry* que proveyó el iPDK que se usó para el diseño. Las siguientes fases mostrarán resultados básicamente de cómo se comportará nuestro circuito en vida real, tomando en cuenta las capacitancias parásitas a lo largo de cada net, si los resultados en las siguientes fases no son los deseados, se debe hacer modificaciones de optimización del diseño (puede requerir el uso de aplicaciones avanzadas) y volver a ejecutar todas las fases de diseño. (Synopsys, IC Validator, 2014)

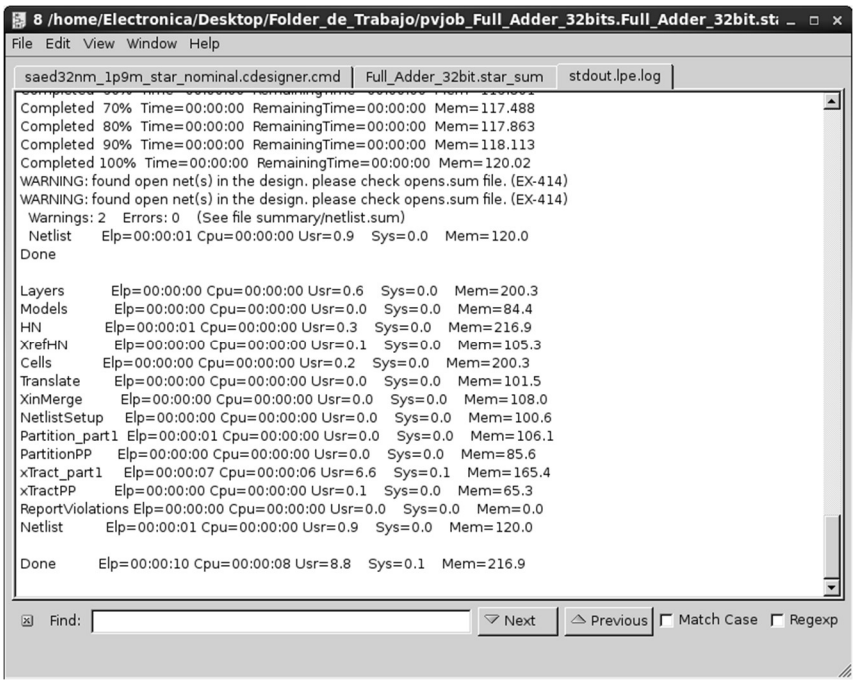
f. Extracción. Esta sección corresponde a la obtención de un *deck* que se asemeje más a la realidad en cuanto a simulación, esto es debido a que se ejecuta un proceso donde se calculan de manera muy acertada todas las capacitancias, inductancias y resistencias parásitas que se generan con el diseño de *layout* que hasta este paso ya se tiene. El proceso es sencillo y de alta utilidad, la aplicación que se integra al entorno de Galaxy Custom Designer es StarRC y como requisito ya se debió haber realizado el LVS ya que se necesitan algunos archivos generados en este proceso. (Synopsys, Extraction & Layout Analysis, 2014). Para poner en marcha el proceso diríjase a **Verification>LPE>Setup and Run**, se especifican las configuraciones básicas en el siguiente cuadro.

Cuadro 10: Configuración de LPE

Sección	Explicación
Pestaña Main	
Run Dir	Al igual que en los casos anteriores, el directorio donde se guardan los archivos generados en el proceso.
Library, Cell y View	Verificar que la librería, la celda y la vista sean las correctas.
Tool	Elegir StarRC, a menos que se prefiera alguna otra disponible.
Runset	Especifica la ubicación del archivo que rige cómo se hará la extracción de parásitos, normalmente se agrega por defecto y se ubica en: `/usr/synopsys/iPDK/SAED32_28_iPDK//starrc/icv_cmd/saed32nm_1p9m_star_nominal.cmd`
Pestaña Extraction Options	
LVS Tool	Es la aplicación que se usó para el proceso LVS.
Runset report File	La ubicación del archivo `pex_runset_report` este se encuentra en el Run Dir del proceso LVS.
TCAD GRD File	Archivo de tecnología que se usa para la extracción de parásitos. `/usr/synopsys/iPDK/SAED32_28_iPDK//starrc/nominal/saed32nm_1p9m_nominal.nxtgrd`
Extraction Type	Es el tipo de extracción que se realizará: R, C, RC, RKC. Esto se refiere a si se tomarán en cuenta sólo las resistencias parásitas, las capacitancias, capacitancias y resistencias o capacitancias, resistencias e inductancias respectivamente.
Pestaña Output Options	
Format	El formato del archivo de salida, es el formato del *deck* final con parásitos. Se recomienda usar el formato SPF que es en texto plano para poder usarlo en otras aplicaciones más fácilmente.
Netlist File	Dirección del *deck* final.
Netlist Port Order File	Si se desea usar el mismo orden de las terminales del LVS, se debe especificar el *netlist* generado en el proceso LVS normalmente con terminación de nombre `*.cdesgner.sp`.

Cuando la configuración ya esté realizada dar click en OK y el proceso iniciará.

Figura 54: Extracción de parásitos



Al finalizar el proceso se presentará una ventana similar a la anterior, el *deck* final está ahora en la dirección que se eligió en el proceso de configuración. (Synopsys, Extraction & Layout Analysis, 2014)

g. Verificación Post-Layout. Esta fase de diseño es una fase extremadamente importante ya que consiste en poder caracterizar nuestro circuito. Caracterizar se refiere a conocer aspectos del circuito que son cruciales en su funcionamiento, como la potencia disipada, la energía necesaria, y el tiempo máximo que se tarda para realizar una operación. Esta última característica resulta un poco difícil de encontrar, ya que es necesario analizar cada una de las rutas que se usan para lograr un resultado, con todas las posibles combinaciones en los vectores de entrada. En ocasiones la ruta más larga puede ser obvia, o si no lo es puede ser fácil realizar iteraciones con las combinaciones en los vectores de entrada que produzcan un cambio en la salida, pero en proyectos muy complicados no es ni obvio ni de fácil iteración. Para esto se usan aplicaciones específicas como PrimeTime y NanoTime, la primera de ellas a nivel de compuertas, cuando se tenga el circuito descrito con un archivo HDL de Verilog y la segunda es a nivel de transistores.

Cuando la ruta más larga sea una combinación de vectores de entrada obvia, el análisis se puede realizar usando Hspice, la herramienta por excelencia de simulación de circuitos que es el motor del SAE que se usó con anterioridad. Los manuales de usuario de Hspice (Synopsys, HSPICE, 2014) como tal son demasiado extensos, podría tomar varios años en poner en práctica todo lo que se puede hacer con esta aplicación. Un comando muy usado para obtener las características del circuito es el `.measure` que puede ser usado de muchas formas.

A continuación se muestran algunas líneas con las que se puede obtener el retraso de propagación de una ruta crítica simple, la medición de potencia promedio, potencia máxima y operaciones matemáticas con resultados de estas mediciones.

```
.measure tran tpdr trig V(cin)  val='0.5*1.05' rise=1
+ targ=V(cout) val='0.5*1.05' rise=1
```

Esto se puede leer como: medir el tiempo desde que el primer flanco de subida en el nodo `cin` tenga el valor de 0.5*1.05 V hasta el primer flanco de subida en el nodo `cout` cuando tenga 0.5*1.05 V. (Synopsys, HSPICE, 2014)

```
.measure tran POTavg AVG P(V1) from=20n to=22.3n
```

La potencia promedio del elemento V1 en el intervalo de tiempo 20 ns a 22.3 ns.

```
.measure tran POTmax MAX P(V1) from=20n to=22.3n
```

La potencia máxima del elemento V1 en el intervalo de tiempo 20 ns a 22.3 ns.

```
.measure energy param='POTavg*(22.3n-20n)'
```

Cálculo de la energía mediante la potencia media calculada.

Hspice es una herramienta magníficamente poderosa y de extrema confiabilidad, todo su potencial puede aprovecharse para obtener datos del circuito que pueden ayudar al diseñador a mejorar su producto final. Ya se ha mencionado que toda la teoría sobre el uso de Hspice es muy extensa, sin embargo, con el SAE, que ya se ha configurado y utilizado se puede aprovechar de una manera fácil una mayor parte de Hspice.

IV. ANTECEDENTES

Como ya se ha mencionado esta rama de la electrónica es recientemente explotada en la región, con anterioridad se carecía del software profesional de Synopsys para el diseño en VLSI, sin embargo, se usaban aplicaciones gratuitas para introducir al estudiante en el tema. Se realizó un sumador completo de 32 bits usando estas aplicaciones gratuitas.

Figura 55: Esquemático primera versión de sumador de 32 bits

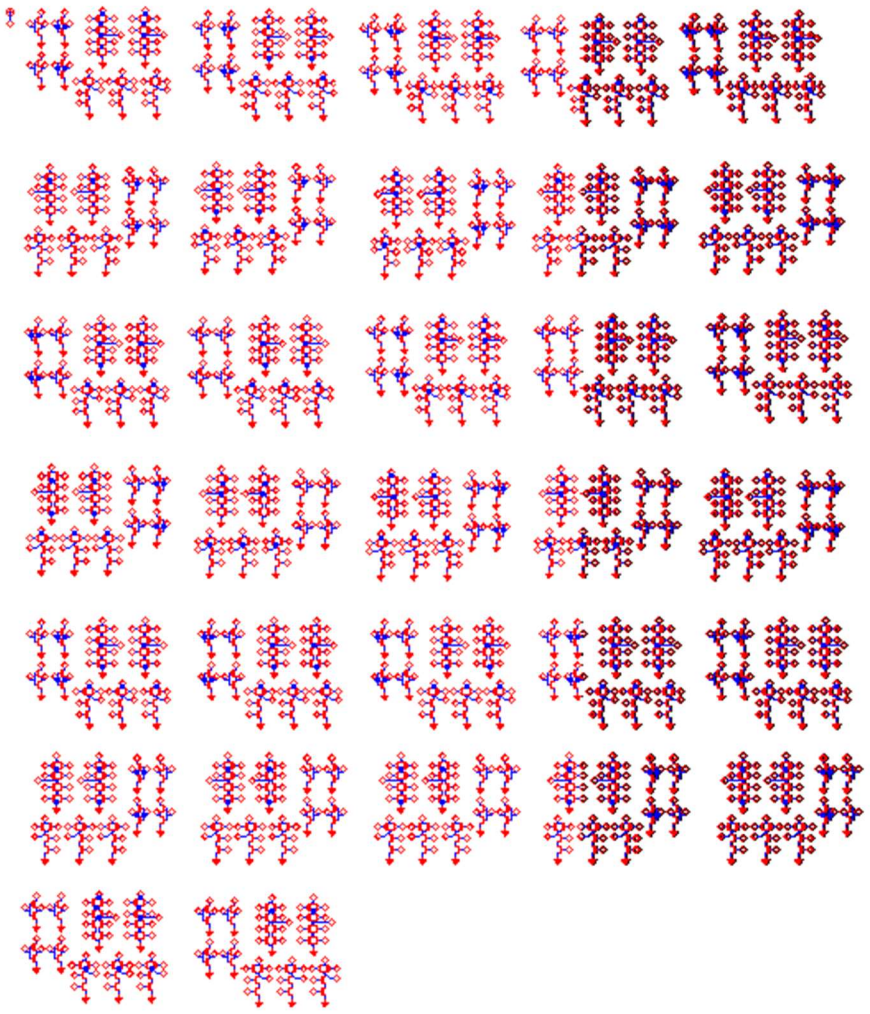
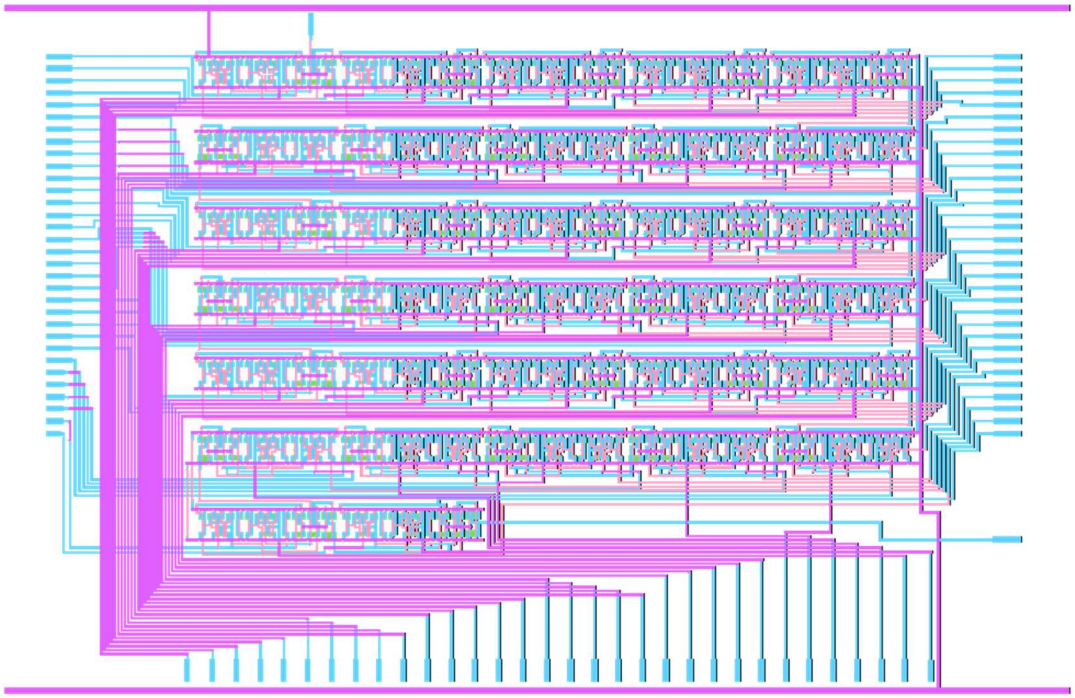


Figura 56: Layout de primera versión de sumador de 32 bits



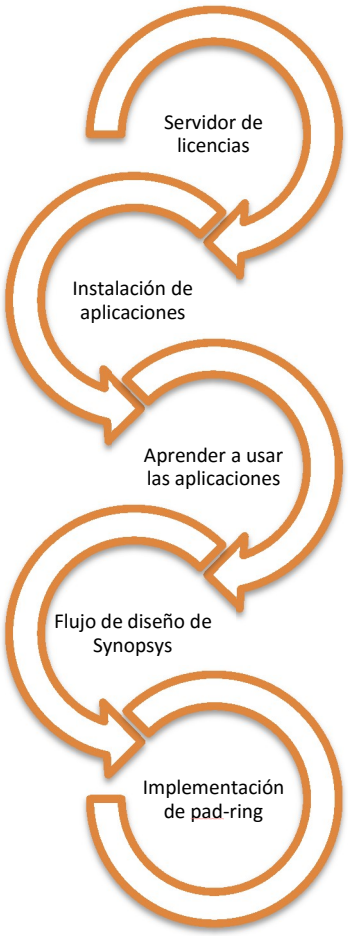
Un resumen de los resultados obtenidos en el diseño de esta primera versión se muestra en el siguiente cuadro.

Cuadro 11: Resultados obtenidos en primera versión de sumador de 32 bits

Tecnología	CMOS de 130 nm
Área de silicio	94 x 80 $(\mu m)^2$
Retraso de propagación (sin extracción de parásitos)	1.5 ns
Frecuencia máxima de operación	667 MHz
Potencia promedio	9 µW
Densidad	15 MT/cm^2

Es importante mencionar que esta primera versión carece de una verificación de reglas de diseño y de una extracción de parásitos lo cual puede afectar drásticamente los resultados que se presentan en el cuadro anterior. Sin embargo, se aprovecharán estos datos para discutir los resultados obtenidos en el diseño realizado en este trabajo en las secciones posteriores. (Universidad del Valle de Guatemala, 2013)

V. METODOLOGÍA



- Descargar de archivos
- Instalación de aplicaciones
- Prueba y experimentación de las aplicaciones instaladas

- Descarga de archivos
- Instalación de CentOS
- Instalación de SCL
- Implementación de SCL

- Instalación de librerías
- Elaboración de guías de uso

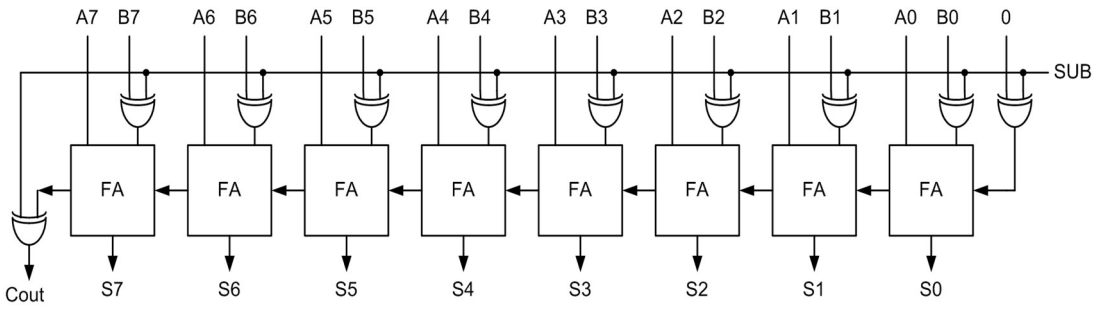
- Esquemático
- Simulación
- Análisis
- Schematic-DrivenLayout
- DRC y LVS
- Parasitic extraction
- Verificación post-Layout

- Investigación sobre librerías de pads
- Implementación de pad-ring
- Pruebas y análisis usando pads

VI. RESULTADOS

El circuito que se implementó consiste en un *Ripple Carry Adder*, este circuito consta de sumadores completos de 1 bits realizando un conexión en cascada. Debido a que no sólo se necesitaba un sumador sino que también, de alguna forma se quería restar lasr entradas se agregó compuertas XOR a la entrada de uno de los dígitos a operar, esto para que con una señal de control pueda invertirse y sumársele 1 mediante el Carry In para que se convierta la entrada a complemento a 2. La siguiente figura muestra el circuito implementado. (Kem, 2014)

Figura 57: Diagrama esquemático de un sumador/restador



El diseño final obtenido entonces muestra la siguiente tabla de verdad, con la que se puede resumir su funcionamiento.

Cuadro 12: Tabla de verdad del diseño final

SUB	CIN	S[ ]
0	0	A[ ] + B[ ]
0	1	A[ ] + B[ ] + 1
1	0	A[ ] – B[ ]
1	1	Estado no permitido

Luego de pasar por las diferentes fases de diseño se obtuvieron resultados satisfactorios en cuanto a los procesos de verificación DRC y LVS como en los diferentes análisis de simulación. A continuación se presentan los resultados de forma ordenada respecto al proceso de diseño.

Figura 58: Diagrama esquemático final

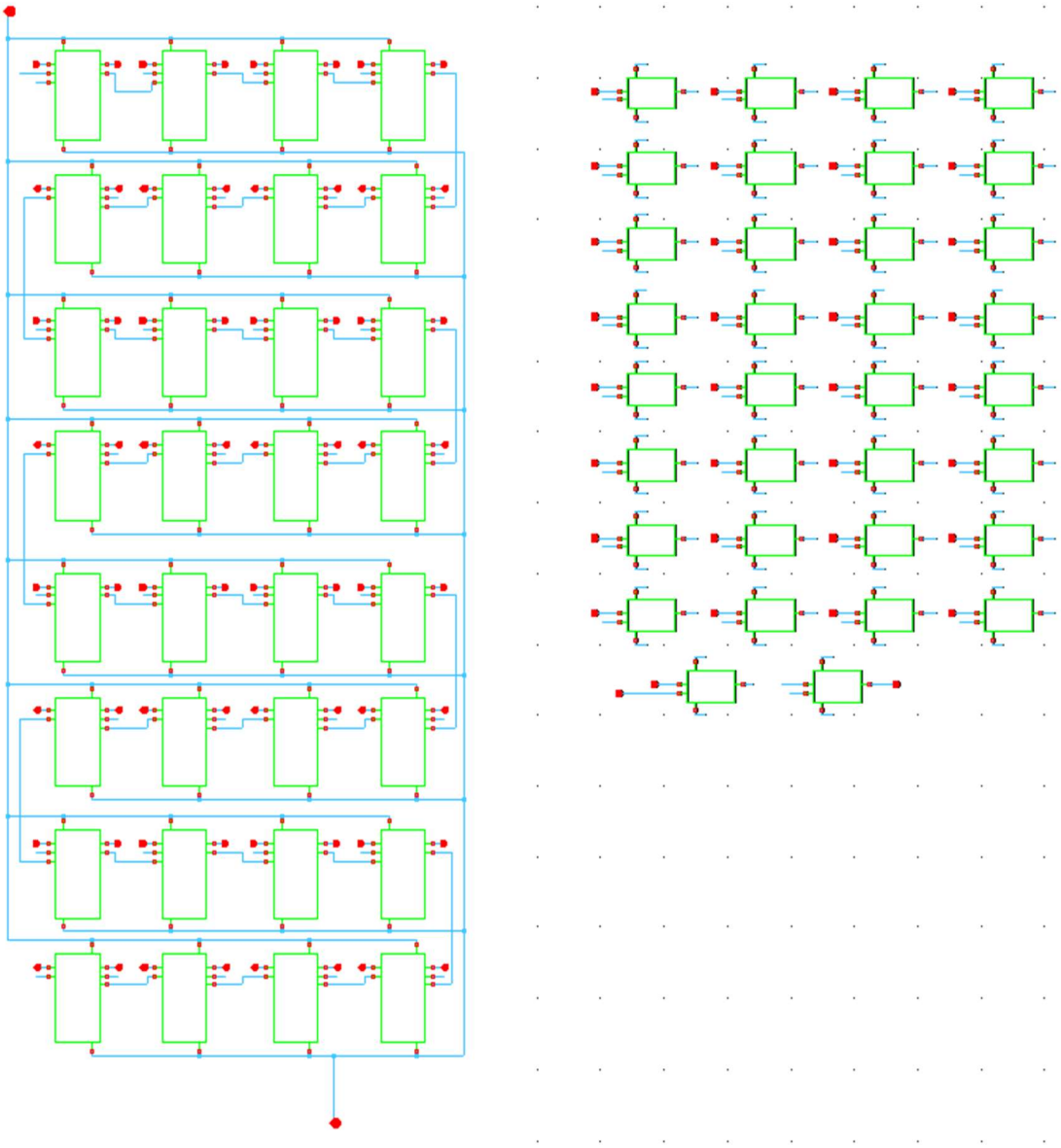


Figura 59: Esquemático para testbench

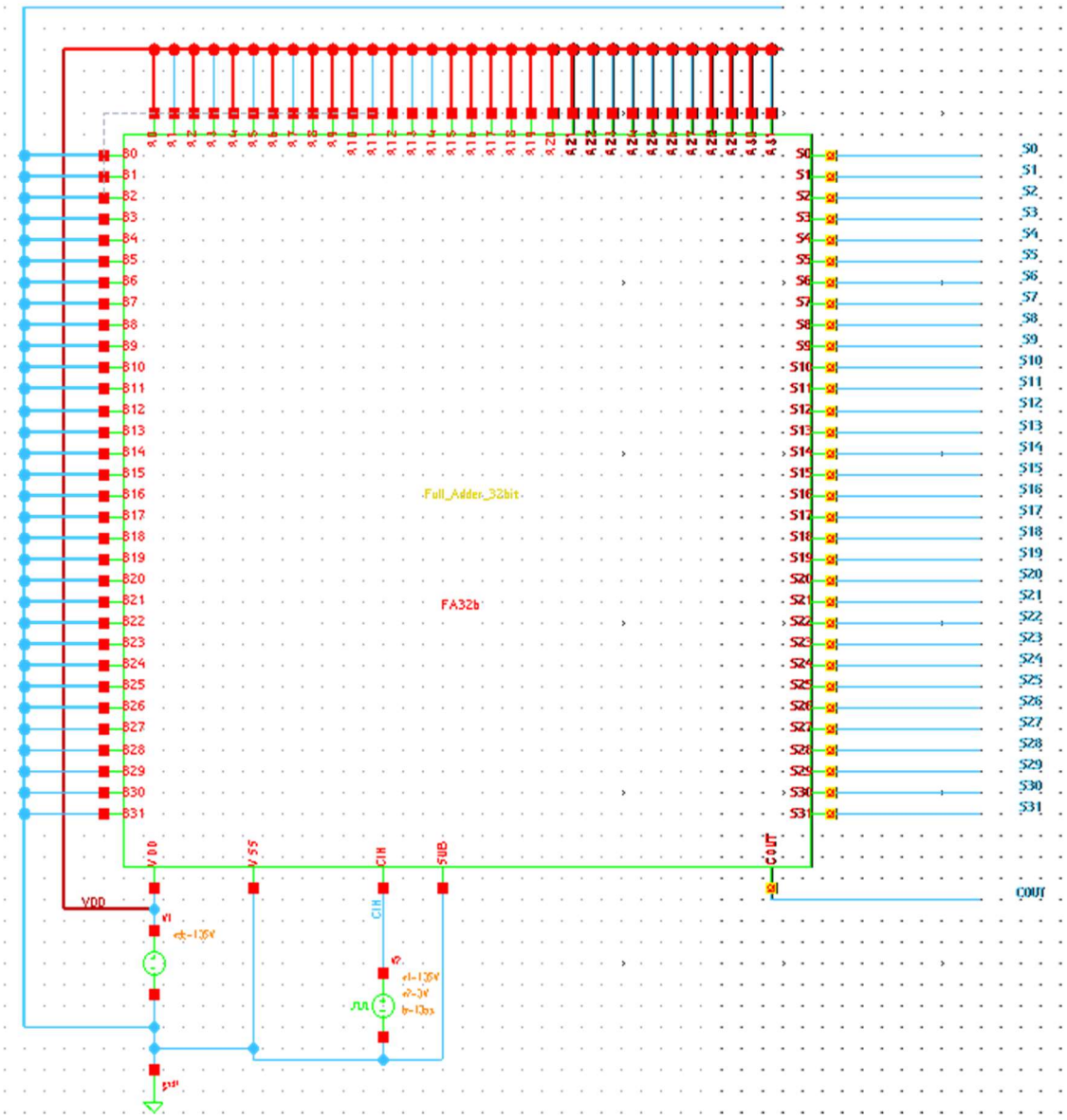


Figura 60: Simulación, suma de prueba en todos los estados del sistema

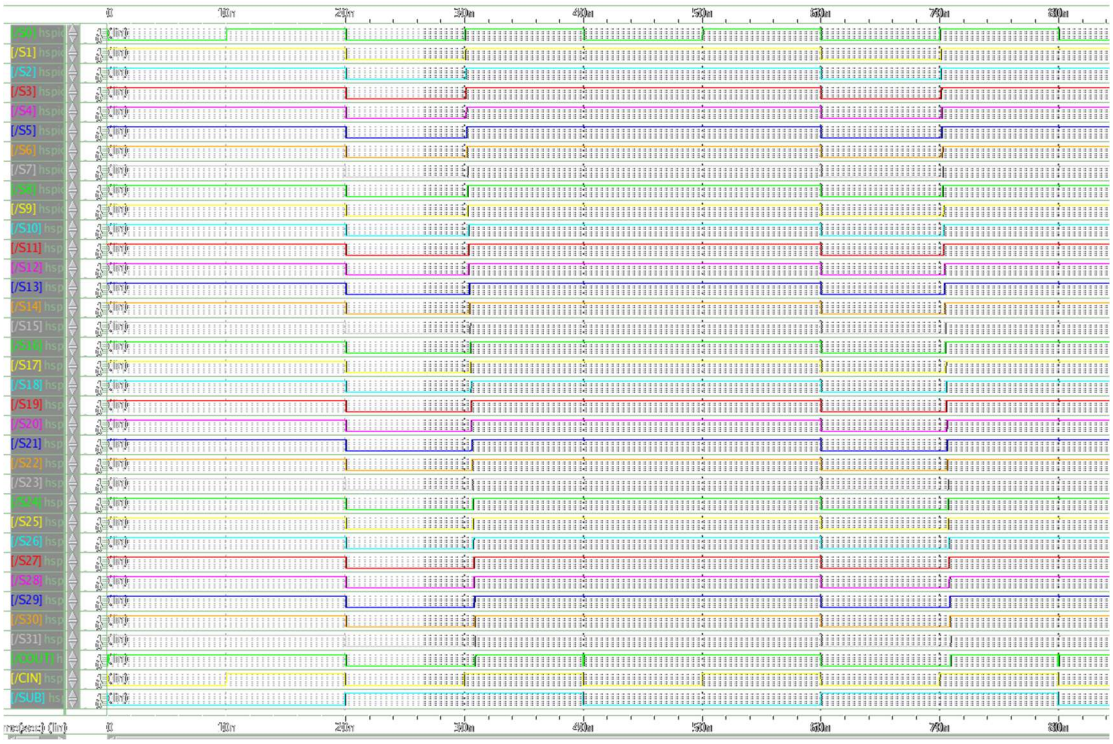


Figura 61: Transición H-L de todas las señales en ruta crítica

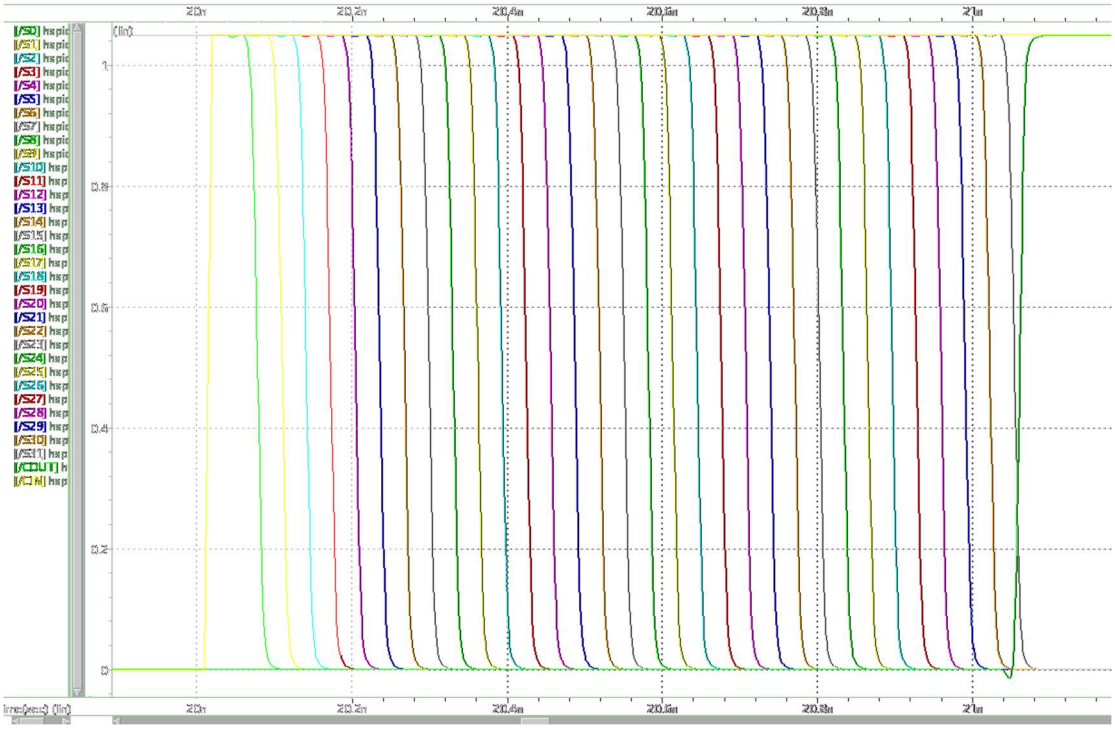


Figura 62: Transición L-H de todas las señales en ruta crítica

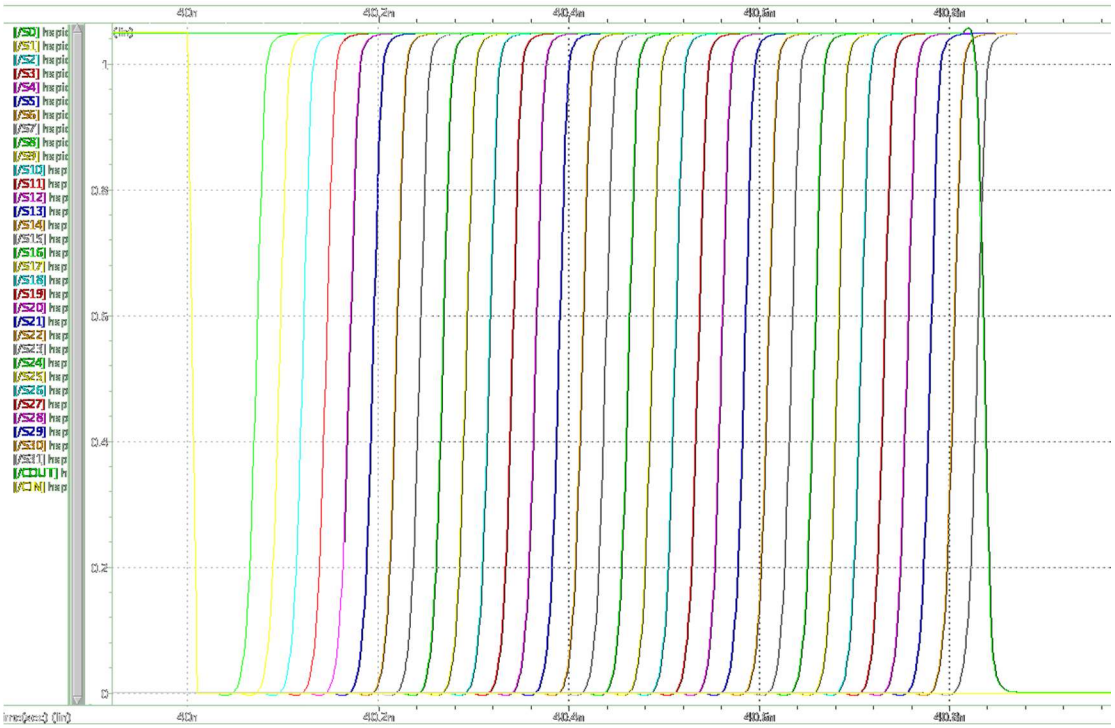


Figura 63: Ruta crítica CIN (rising) Δt = 1.05 ns

Figura 64: Ruta crítica CIN (falling) Δt = 832 ps

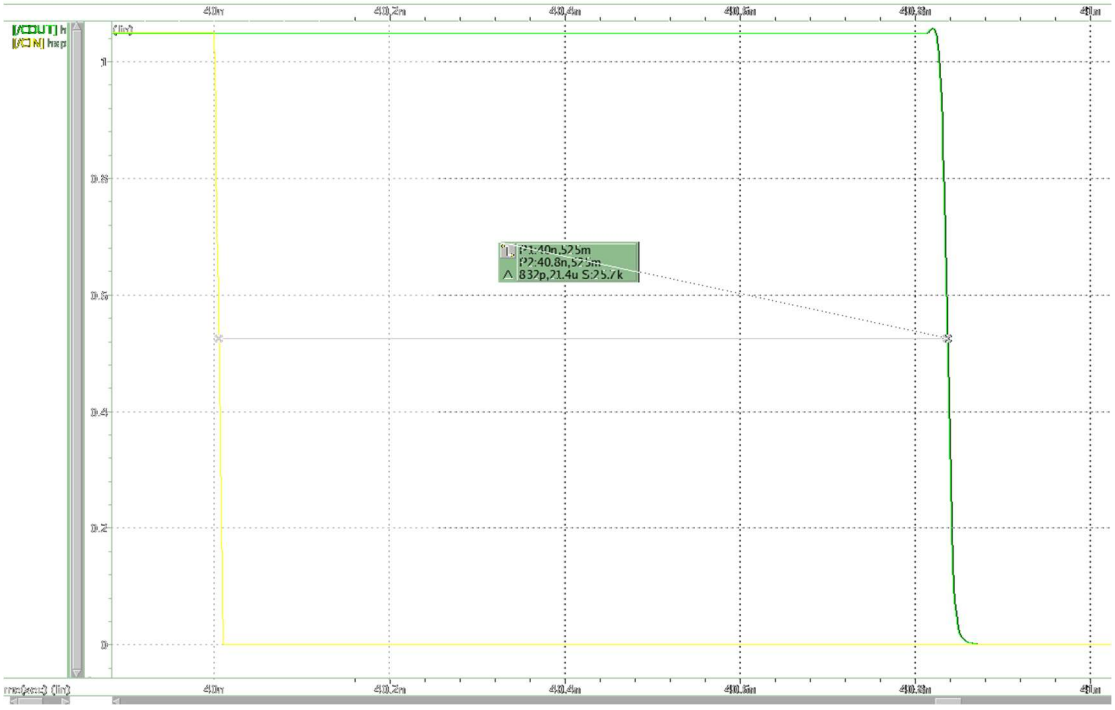


Figura 65: Ruta crítica B0 (rising) Δt = 1.05 ns

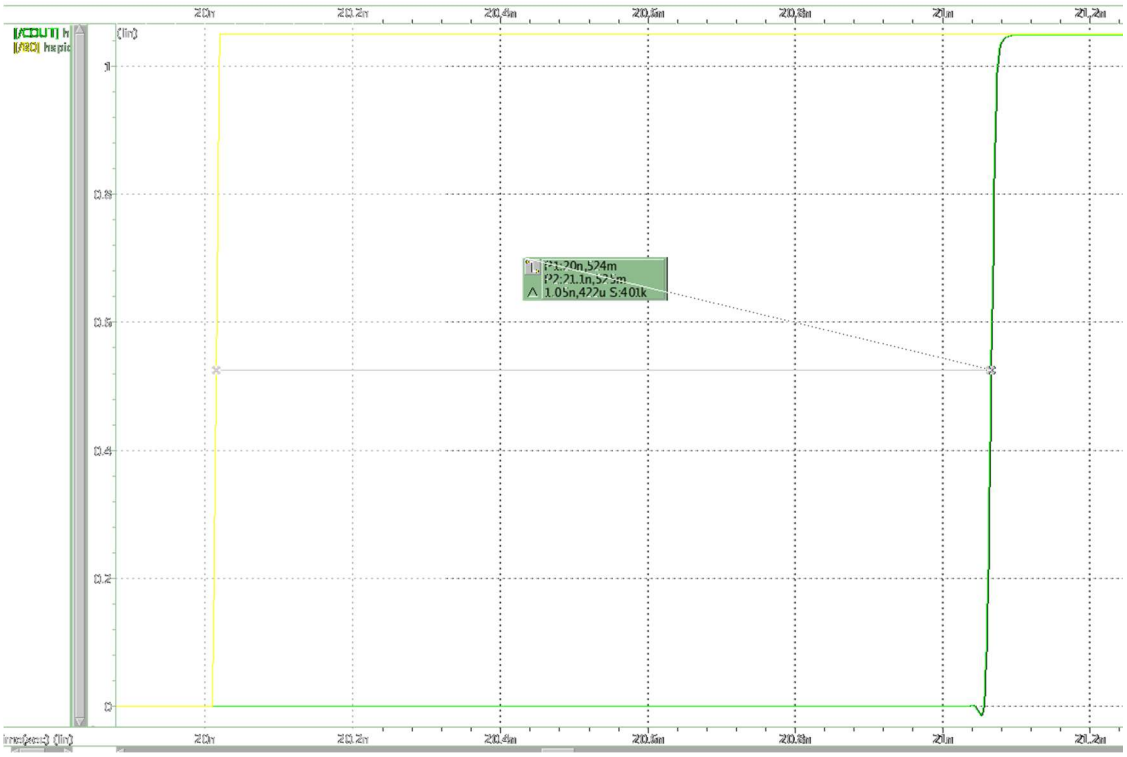


Figura 66: Ruta crítica B0 (falling) Δt = 838 ps

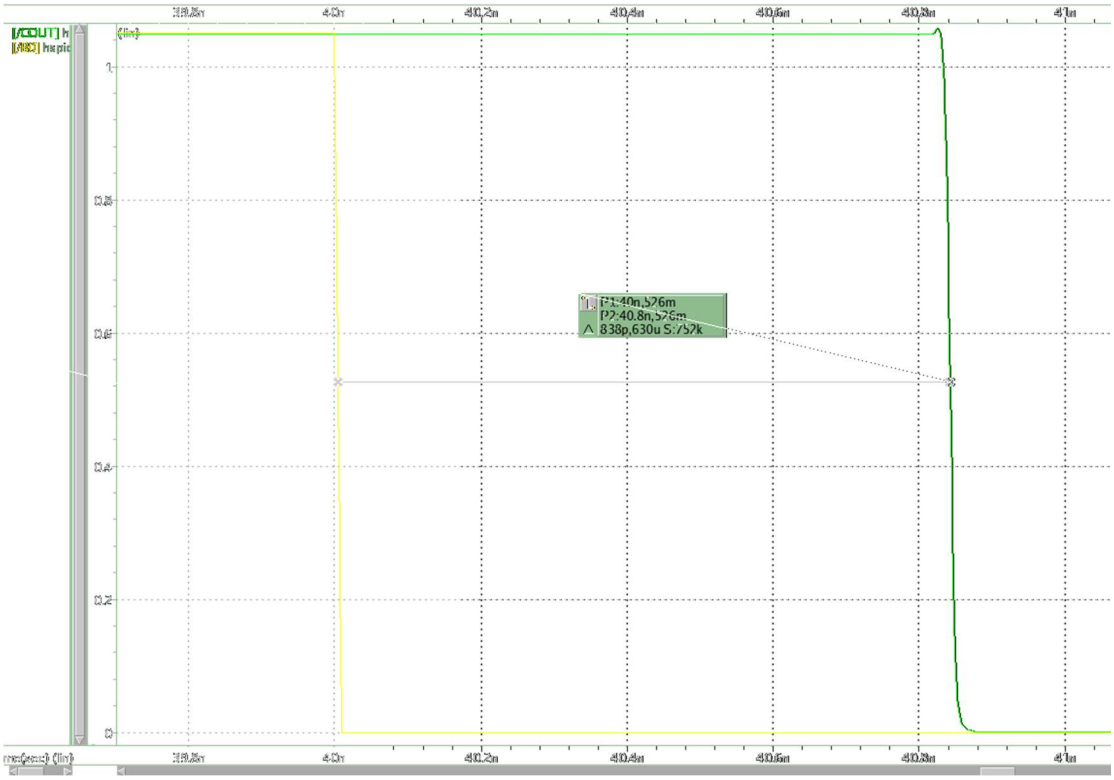


Figura 67: Ruta crítica A0 (rising) Δt = 1.03 ns

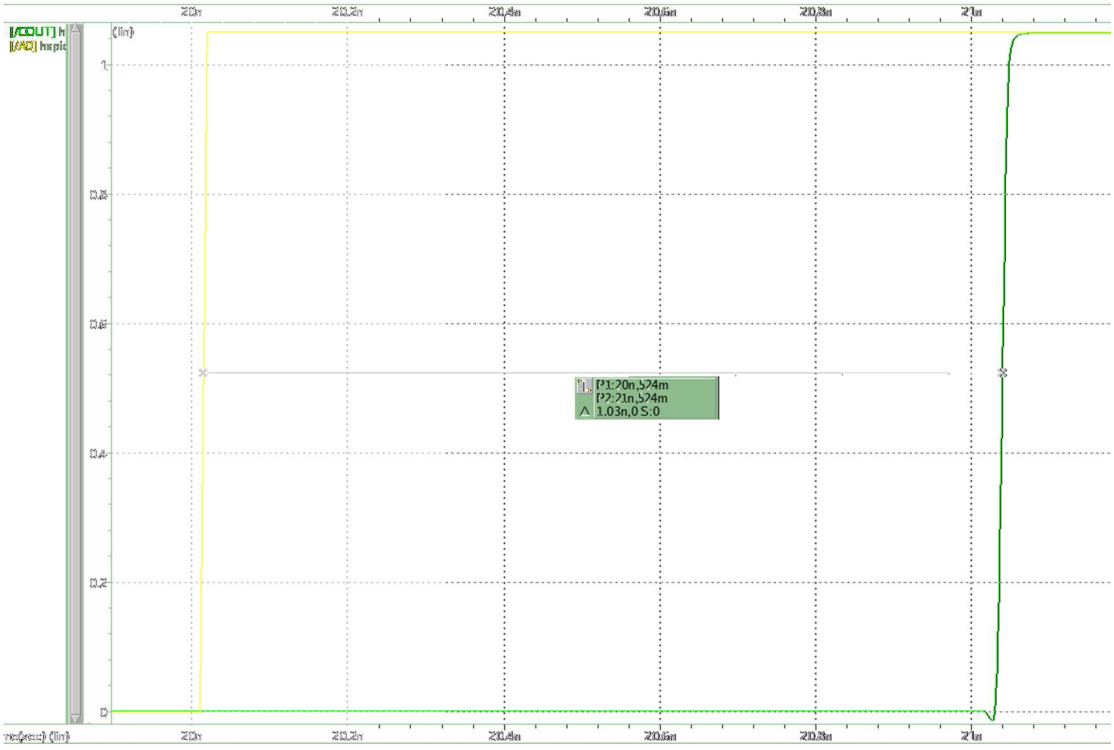


Figura 68: Ruta crítica A0 (falling) Δt = 805 ps

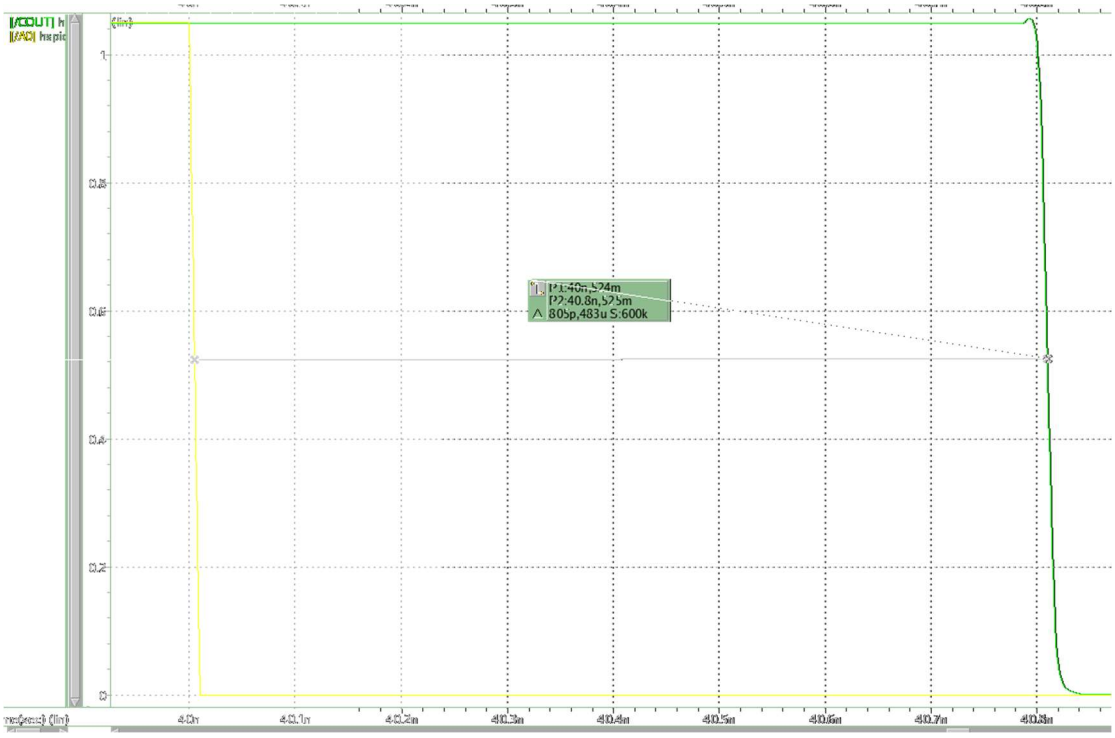


Figura 69: Layout final (die)

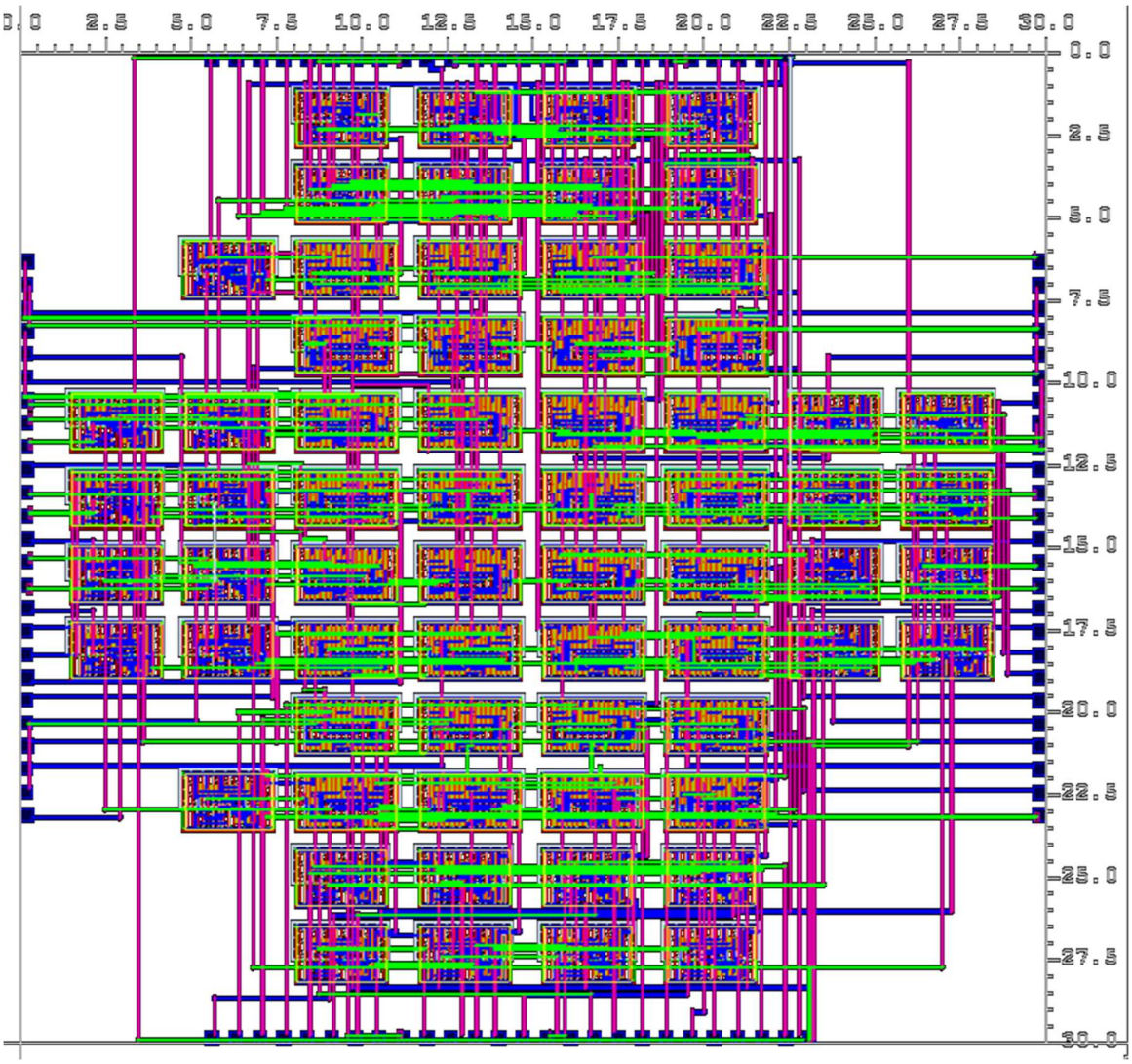


Figura 70: Verificación de reglas de diseño exitosa

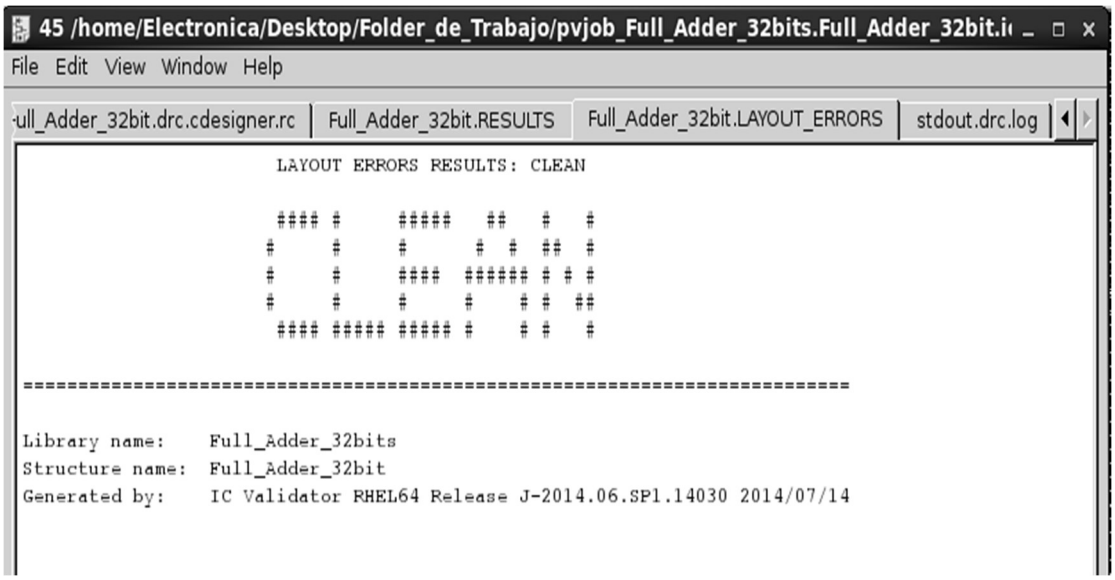


Figura 71: Comparación de esquemático contra layout exitosa

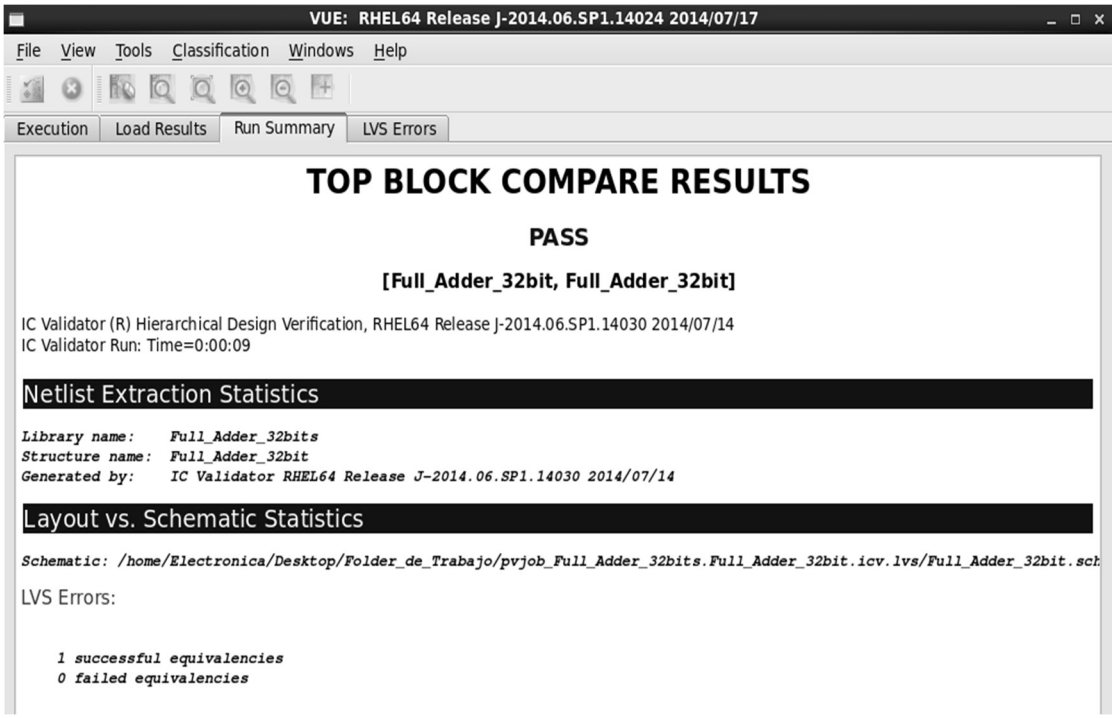


Figura 72: Deck con parásitos (muestra)

```
71784 XMXI132|MMN6 XI132|MMN6:DRN XI132|MMN6:GATE VSS VSS n105 SCA=72.6261 SCB=0.0449695 SCC=0.00974379 ad=0.02754p
      as=0.02754p l=0.03u m=1 nf=1 nrd=0.377778 nrs=0.377778 pd=0.744u ps=0.744u w=0.27u
71785 XMXI132|MMN7 XI132|MMN7:DRN XI132|MMN7:GATE VSS VSS n105 SCA=95.3561 SCB=0.0439188 SCC=0.0101565 ad=0.04284p
      as=0.04284p l=0.03u m=1 nf=1 nrd=0.242857 nrs=0.242857 pd=1.044u ps=1.044u w=0.42u
71786 XMXI132|MMP1 XI132|MMP1:DRN XI132|MMP1:GATE VDD VDD p105 SCA=59.1158 SCB=0.0550513 SCC=0.00733071 ad=0.03672p
      as=0.03672p l=0.03u m=1 nf=1 nrd=0.283333 nrs=0.283333 pd=0.924u ps=0.924u w=0.36u
71787 XMXI132|MMP2 XI132|MMP2:DRN XI132|MMP2:GATE VDD VDD p105 SCA=60.2967 SCB=0.0623035 SCC=0.00738467 ad=0.03366p
      as=0.03366p l=0.03u m=1 nf=1 nrd=0.309091 nrs=0.309091 pd=0.864u ps=0.864u w=0.33u
71788 XMXI132|MMP3 XI132|MMP3:DRN XI132|MMP3:GATE VDD VDD p105 SCA=56.237 SCB=0.0550496 SCC=0.00691423 ad=0.03366p
      as=0.03366p l=0.03u m=1 nf=1 nrd=0.309091 nrs=0.309091 pd=0.864u ps=0.864u w=0.33u
71789 XMXI132|MMP4 XI132|MMP4:DRN XI132|MMP4:GATE XI132|MMP4:SRC VDD p105 SCA=55.2788 SCB=0.0530149 SCC=0.00686298
      ad=0.03366p as=0.03366p l=0.03u m=1 nf=1 nrd=0.309091 nrs=0.309091 pd=0.864u ps=0.864u w=0.33u
71790 XMXI132|MMP5 XI132|MMP5:DRN XI132|MMP5:GATE VDD VDD p105 SCA=77.9536 SCB=0.0763946 SCC=0.01096 ad=0.0408p as=0.0408p
      l=0.03u m=1 nf=1 nrd=0.255 nrs=0.255 pd=1.004u ps=1.004u w=0.4u
71791 XMXI132|MMP6 XI132|MMP6:DRN XI132|MMP6:GATE XI132|MMP6:SRC VDD p105 SCA=62.5933 SCB=0.0623076 SCC=0.00771724
      ad=0.0408p as=0.0408p l=0.03u m=1 nf=1 nrd=0.255 nrs=0.255 pd=1.004u ps=1.004u w=0.4u
71792 XMXI132|MMP7 XI132|MMP7:DRN XI132|MMP7:GATE VDD VDD p105 SCA=115.307 SCB=0.0754836 SCC=0.0136763 ad=0.0765p as=0.0765p
      l=0.03u m=1 nf=1 nrd=0.136 nrs=0.136 pd=1.704u ps=1.704u w=0.75u
```

En las siguientes imágenes se muestran los análisis realizados tomando en cuenta la extracción de parásitos de resistencias y capacitancias.

Figura 73: Ruta crítica CIN (rising) Δt = 2.11 ns

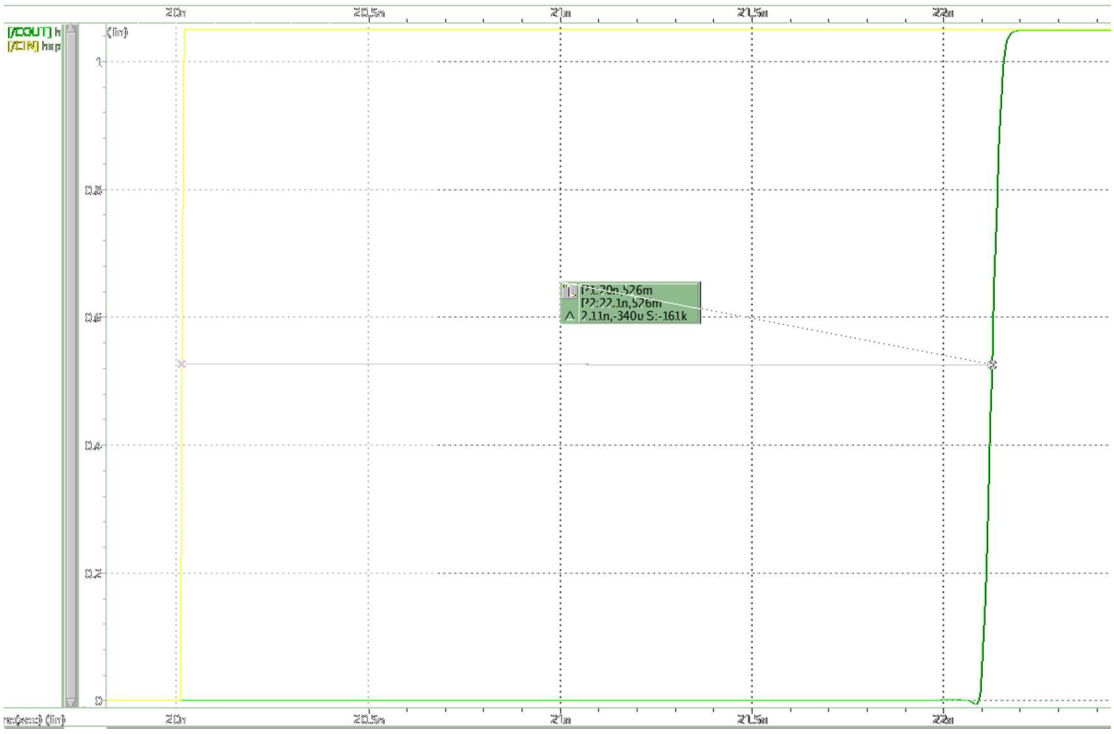


Figura 74: Ruta crítica CIN (falling) Δt = 1.7 ns

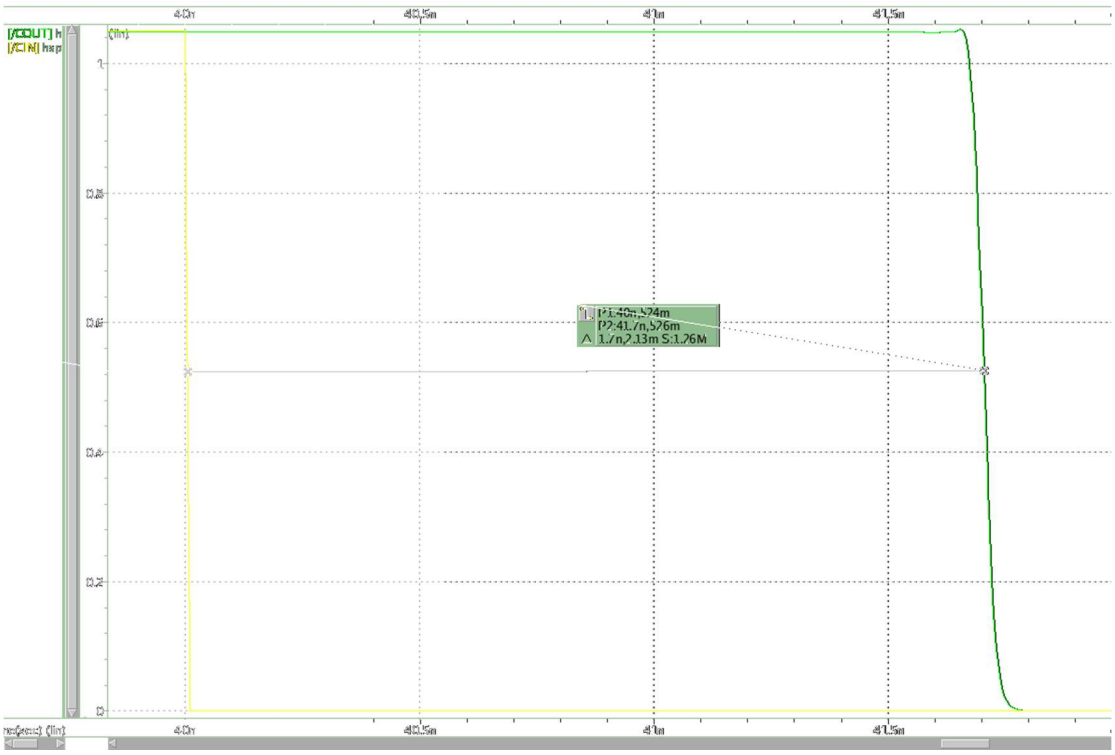


Figura 75: Ruta crítica B0 (rising) Δt = 2.12 ns

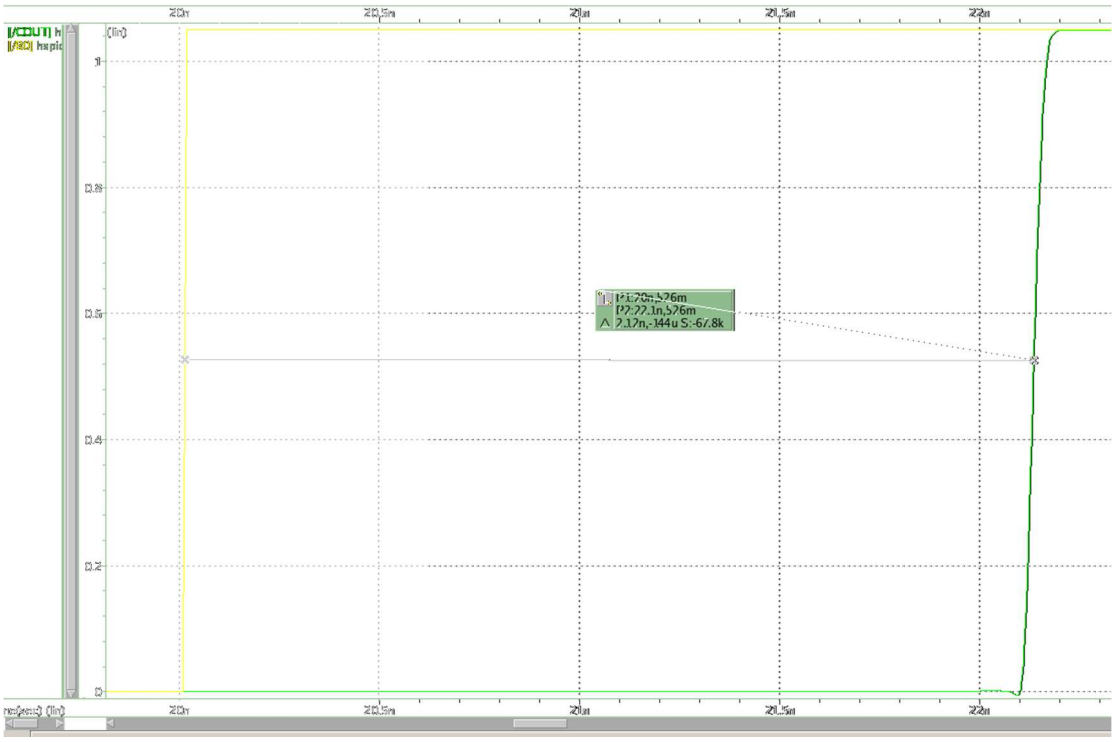


Figura 76: Ruta crítica B0 (falling) Δt = 1.71 ns

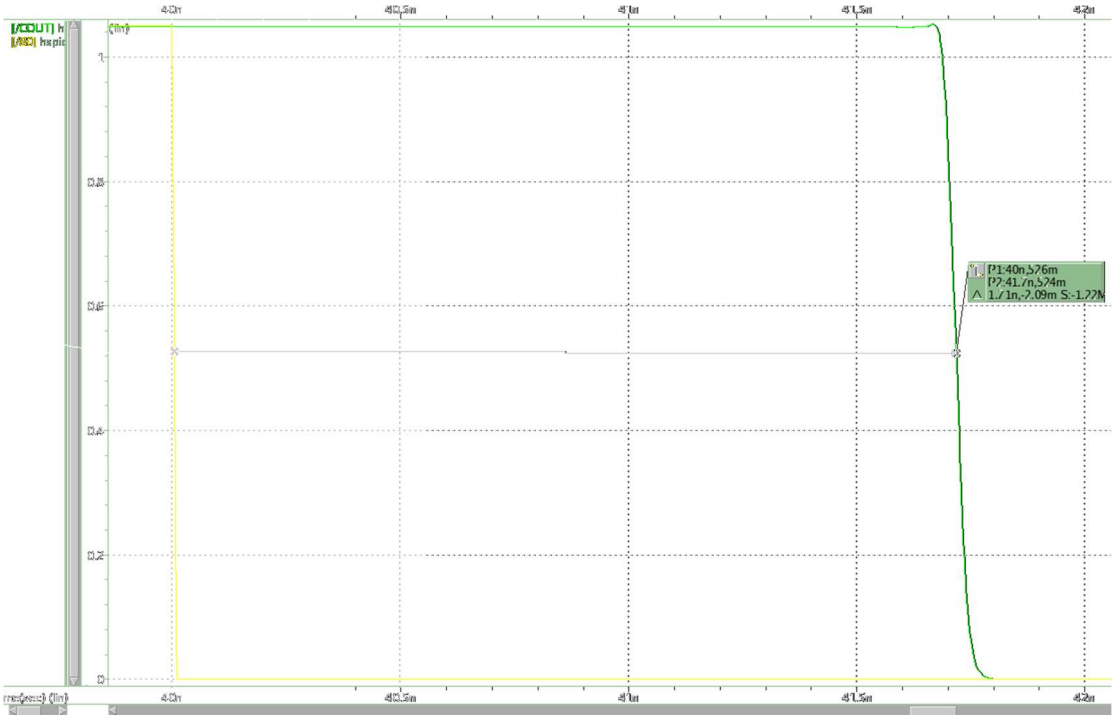


Figura 77: Ruta crítica A0 (rising) Δt = 2.06 ns

Figura 78: Ruta crítica A0 (falling) Δt = 1.64 ns

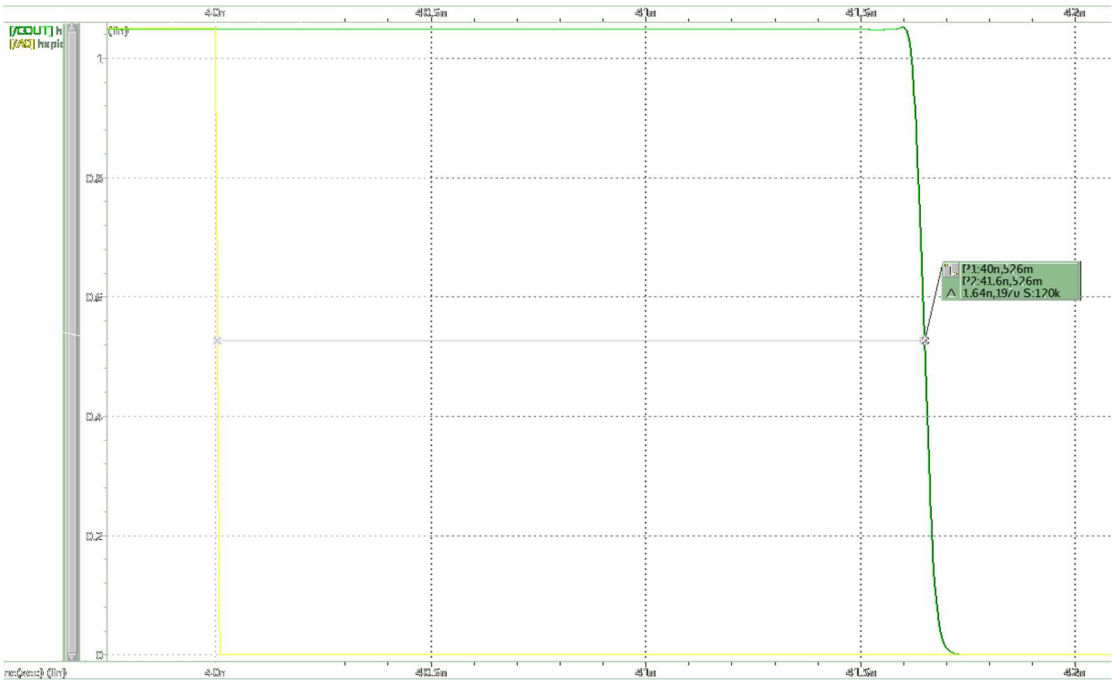


Figura 79: Transición H-L de las señales en ruta crítica

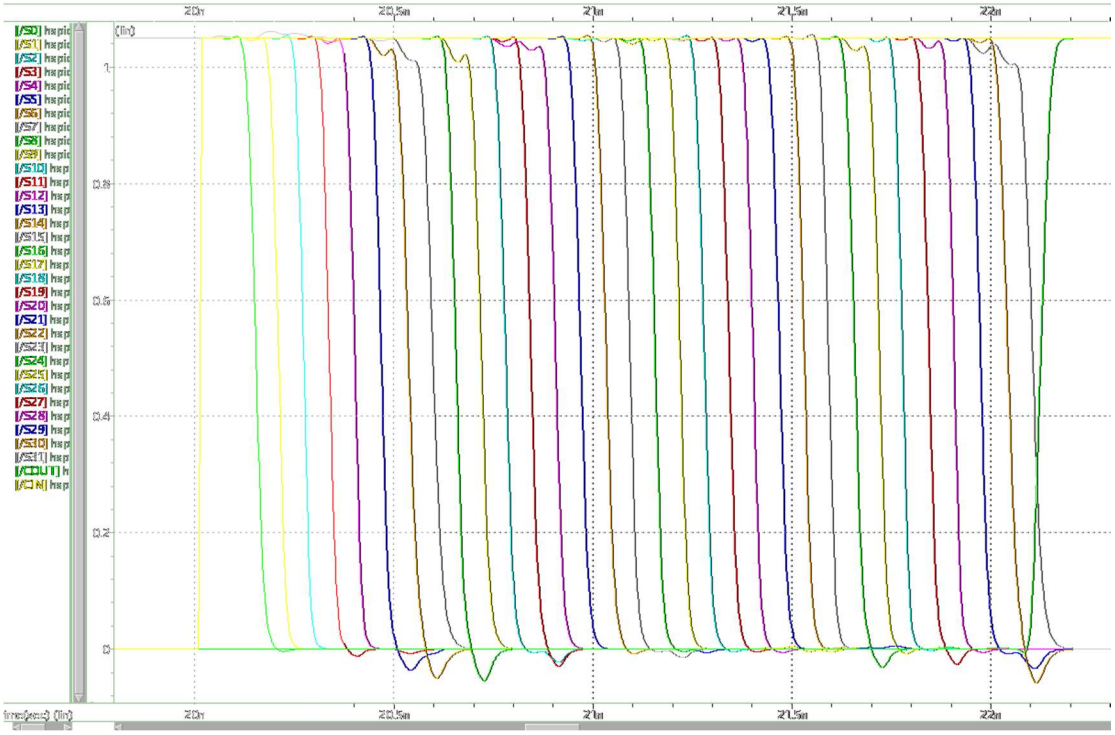
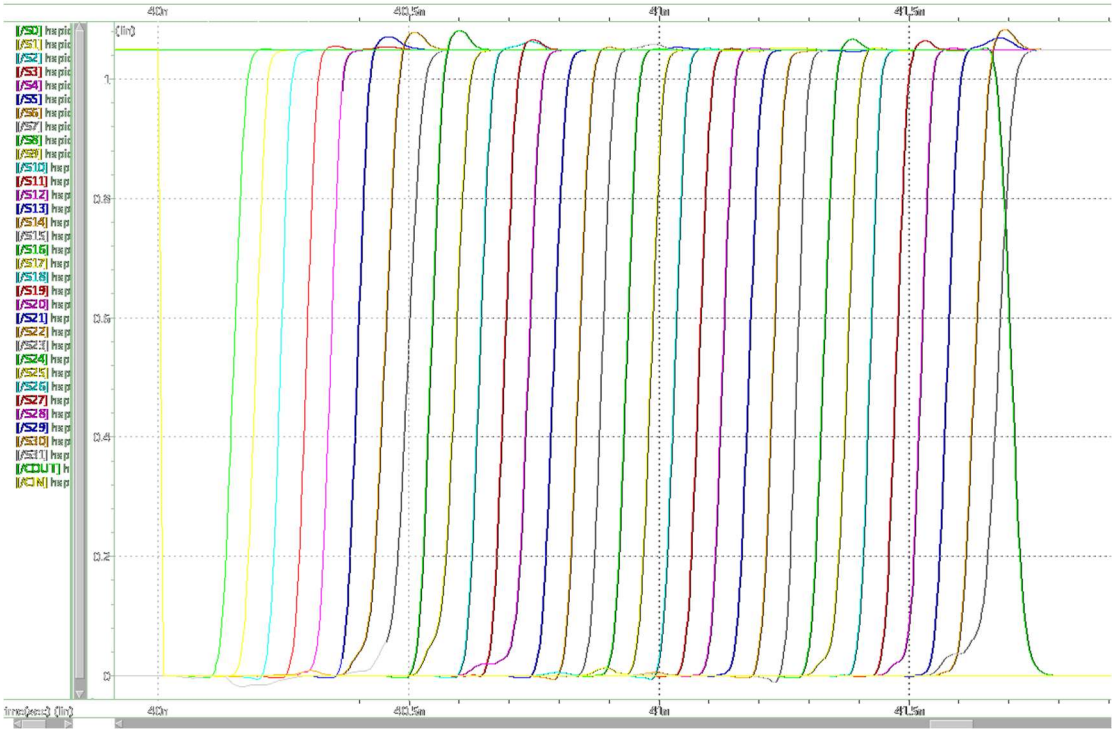


Figura 80: Transición L-H de las señales en ruta crítica



Cuadro 13: Resumen de resultados obtenidos

Tecnología	CMOS de 28 nm
Área de silicio (*die*)	30 x 30 $(\mu m)^2$
Retraso de propagación	2.12 ns.
Frecuencia máxima de operación	471.7 MHz
Potencia promedio	-112.7054 µW
Energía	-238.0362 fJ
Número de transistores	1372
Densidad	152.4 MT/$(cm)^2$

Las gráficas que muestran a continuación presentan los resultados obtenidos agregando al *die* el *pad ring*.

Figura 81: Diseño final

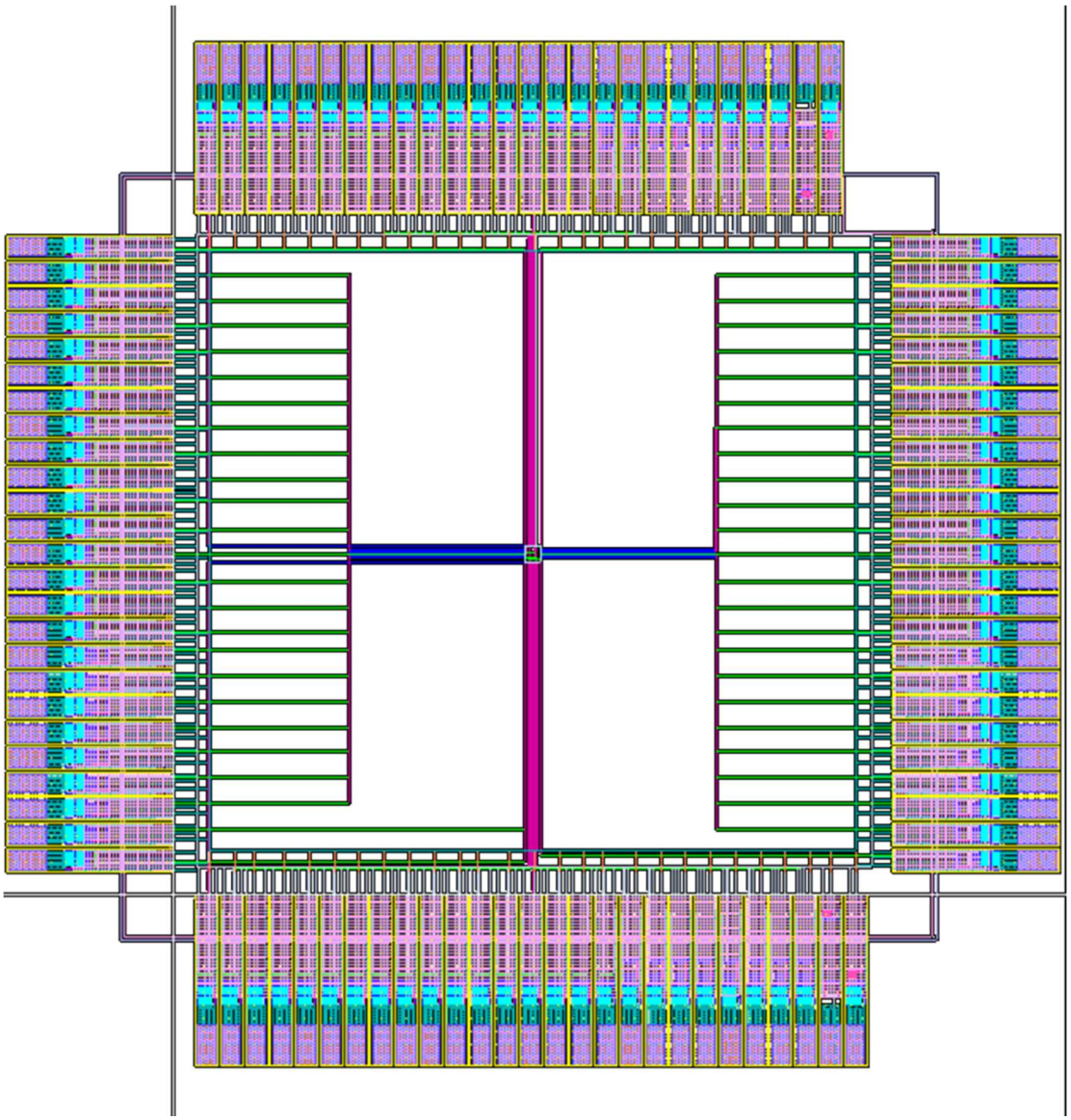


Figura 82: Ruta crítica con pad's B0 (rising) Δt = 4.56 ns

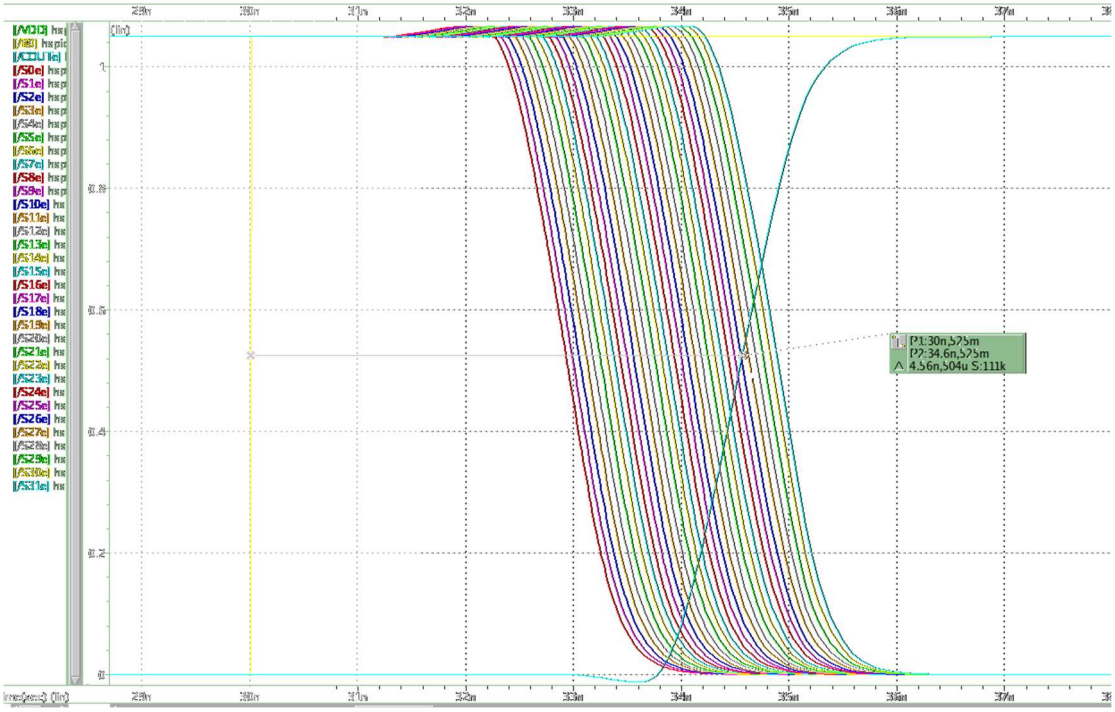
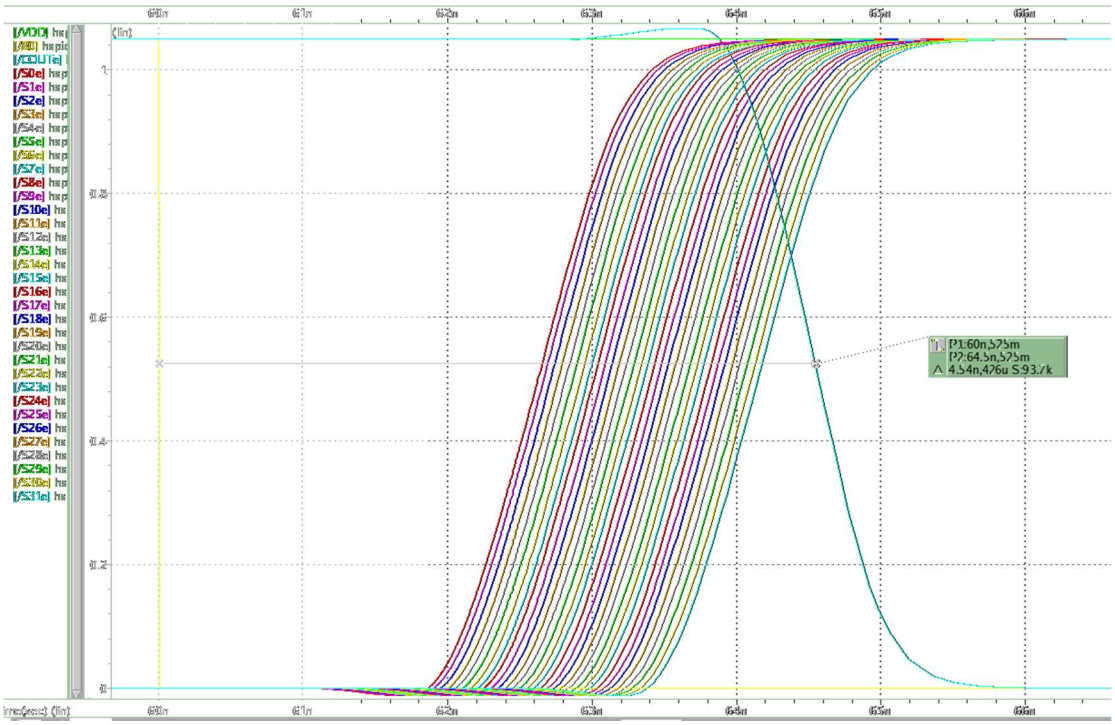


Figura 83: Ruta crítica con pad's B0 (falling) Δt = 4.54 ns



VII. ANÁLISIS DE RESULTADOS

El capítulo anterior muestra un proceso de diseño con resultados satisfactorios, estos resultados fueron obtenidos después de muchas pruebas y correcciones tanto en la plataforma de trabajo como en el diseño en sí. Los problemas en la plataforma de trabajo resultaron de una instalación correcta pero de una configuración en el sistema operativo CentOS incompleta. Por ejemplo, cuando se necesitaba realizar el proceso de extracción de parásitos no se podía comunicar el Galaxy Custom Designer con StarRC, el *log* de la ejecución mostraba un error que indicaba que uno de los archivos necesarios faltaba, este problema similar a muchos otros ocurridos con las demás aplicaciones surgía debido a la falta de alguna variable de entorno o la que existía apuntaba a un directorio incorrecto. Las variables de entorno son de vital importancia para que las aplicaciones se puedan comunicar entre sí y también con el sistema operativo en general. Desde un principio se sabía que se iban a tener problemas de este tipo por la experiencia que se tiene en el uso de diversas distribuciones de Linux; aunque los manuales de instalación se siguieron al pie de la letra algunos de ellos no especificaban el nombre de la variable de entorno o la dirección que se sugería era incorrecta, quizá por cambios en las nuevas versiones. Como primera aplicación desarrollada en este ambiente de trabajo se pretendía superar todos estos problemas con las aplicaciones, dejar todas las configuraciones en un estado óptimo para permitir que en el futuro se dedique únicamente al diseño sin tener problemas con el entorno de trabajo. Un archivo de vital importancia es `$HOME/.bashrc` este archivo es de personalización del usuario hacia el sistema operativo, en el entorno Linux se puede usar para diversas cosas, en este trabajo se ha usado para especificar las variables de entorno necesarias para el correcto funcionamiento de las aplicaciones de diseño. El Cuadro 4 muestra una referencia de cómo este archivo debería editarse para obtener los resultados deseados, también es importante notar que no es posible especificar las variables de entorno para todas las aplicaciones que Synopsys provee por lo que es recomendable, al instalar una aplicación leer el manual de instalación, específicamente la sección de configuración con el entorno. Las aplicaciones quedaron apropiadamente instaladas dando al laboratorio de electrónica de la Universidad del Valle de Guatemala la suite de herramientas profesionales para diseño de circuitos a muy gran escala. El único inconveniente que se tuvo con los objetivos planteados es el de la implementación del método de *logical effort* ya que no se pudo llevar a cabo por la utilización de celdas estándares de diseño. Para poder aplicar *logical effort* es necesario hacer el diseño transistor por transistor debido a que los transistores de las diferentes rutas tienen diferentes tamaños, este tipo de diseño queda entonces fuera del alcance de este documento.

Synopsys recomienda el diseño en 7 fases que tienen un orden lógico y hacen muy eficiente la obtención de un resultado final. A continuación se irán detallando cada una de las fases haciendo referencia a los resultados obtenidos y mencionando los hallazgos encontrados. La primera fase es la realización del diagrama esquemático, la Figura 58 muestra el diseño esquemático final como primera fase de diseño. El circuito consta de varios sumadores completos de 1 bit colocados en cascada como lo muestra la figura 57 y compuertas XOR que con la adición de la entrada CIN convierten uno de los vectores de entrada en complemento a 2. Es importante mencionar que cada una de estas pequeñas cajas negras está construida a nivel transistor en su interior.

Siguiendo el proceso de diseño se observa en la Figura 59 que se creó una caja negra, que es el sumador/restador completo con todos los pines de entrada y salida que tendría ya implementado fuera del empaquetado final, este esquemático para pruebas (*testbench*) se realizó con dos objetivos, el primero fue para verificar el correcto funcionamiento del diseño, que es la segunda fase de diseño, y el segundo dejar una instancia que pueda ser usada en otros diseños que requieran de un sumador/restador de 32 bits, esta es una buena práctica para hacer más simples los procesos de diseño e implementación. Al circuito de la Figura 59 se le inyectaron señales para poder proceder con las simulaciones respectivas y analizar el resultado de las mismas. La versatilidad del graficador que ofrece la aplicación Custom Explorer y la integración con Custom Designer hizo verdaderamente simple esta labor. La Figura 60 muestra el resultado de las simulaciones graficando cada señal de salida con respecto al tiempo. Las señales cambian debido al cambio que se hizo intencionalmente en las entradas CIN y SUB para analizar tanto la suma de los números binarios como la resta de los mismos. El análisis de los resultados de la simulación es la 3ra etapa de diseño, note que en esta fase debemos obtener los resultados que deseamos, no tanto en aspectos como potencia o tiempos de opera-

ción pero sí en los niveles de voltaje que se esperan en la salida. Por ejemplo, en el caso de este proyecto, en esta fase ya debe ser posible realizar sumas y restas, las pruebas se realizaron a una frecuencia adecuada para saber que el circuito sí funciona. Varias veces fueron las que no se obtuvieron resultados satisfactorios, al principio de creyó que era por la velocidad en la que se realizaban las transiciones de los estados por lo que se disminuyó la frecuencia de operación, y seguía sin funcionar. Todo indicaba que era una mala conexión en la circuitería que conforma el esquemático y así fue. Puede ser una tarea realmente difícil encontrar este tipo de errores, como el lector seguramente sabe por simulaciones realizadas en otros entornos, por lo que se recomienda tener mucho cuidado en la primera fase de diseño. Como es lógico, sin tener los resultados que se esperan no se puede pasar a las siguientes etapas de diseño.

La cuarta etapa de diseño es la generación del *layout*, una fase que probablemente todos los interesados en la nanoelectrónica están ansiosos en llegar. La maravilla que ahora son las aplicaciones de diseño como las que Synopsys provee hacen de esta tarea, una labor relativamente simple. El proceso se denomina Schematic Driven Layout (SDL) como ya se ha explicado en las guías rápidas de uso de las aplicaciones que conforman el marco teórico de este documento. En esta fase se usó una combinación estratégica de posicionamiento de las instancias en el editor de *layout* del generador automático y del *pick and place* esto para tener un diseño más ordenado y probablemente más eficiente en cuanto a área. Luego de tener las instancias debidamente alineadas y debidamente ubicadas se procede a realizar el proceso de auto ruteo, esta función normalmente la logra hacer en su totalidad Galaxy Custom Designer pero suele fallar cuando se le indica que rutee todas las pistas de golpe. Lo que se hizo fue rutear todas las rutas y al terminar el proceso en la Custom Designer Console se imprime un sumario de cuáles rutas fueron completadas y cuáles no, de las que no fueron ruteadas se repite el proceso indicando que rutee sólo éstas faltantes y así hasta completar el 100% del ruteo. Si en algún caso no fuera posible por parte del programa hacer el ruteo puede rutearse a mano, estos errores se encontrarán en las siguientes fases de diseño. La Figura 69 muestra el *die* final obtenido.

La quita etapa es la de la verificación física del diseño, los errores obtenidos en el proceso de diseño fueron detecciones de violaciones en las reglas de diseño y errores en la comparación del esquemático con el *layout* generado. Los problemas de DRC aunque en general fue un número grande de violaciones encontradas fueron fáciles de solucionar debido a que el programa IC Validator como ya se ha mencionado se comunica con el Layout Editor de Custom designer y ubica el error marcándolo con una malla amarilla (que puede ser modificada desde IC Validator) como lo muestra la Figura 51. En el proceso LVS también se encontraron algunas desigualdades, LVS por ser un proceso de IC Validator también se comunica con Custom Designer e identifica y ubica el error de la misma forma que DRC. Los problemas que se encuentren durante la ejecución de LVS son producto de un mal ruteo en las distintas capas de metal, y un mal ruteo puede originarse por alguna deficiencia en la configuración del Auto Route o porque simplemente el software no fue capaz de hacer la conexión con las restricciones impuestas. Normalmente el Auto Route realiza la mayoría de conexiones sin ningún problema, en el proceso del diseño del sumador/restador fueron 2 conexiones las que no se conectaron automáticamente las cuales fueron detectadas fácilmente con los resultados del LVS y fueron realizadas las conexiones manualmente que también se ofrece una forma muy intuitiva de realizarlo. Se puede ver en las figuras 70 y 71 el resultado satisfactorio de las verificaciones DRC y LVS respectivamente.

Hasta este punto ya se tiene un *layout* completamente terminado, pero es necesario analizar cómo se comportará el circuito con este diseño de *layout*, las simulaciones realizadas con anterioridad muestran el correcto funcionamiento a nivel esquemático, pero no se toman en cuenta los efectos que el *layout* agrega al circuito. Para poder analizar estos efectos es necesario ejecutar un proceso llamado LPE (Layout Parasitic Extaction), que realiza una extracción de capacitancias, inductancias y resistencias parásitas y las interconecta en un *deck* con el resto del circuito a nivel de transistores, esta es la sexta etapa de diseño. Este análisis es crucial para la verificación de potencia disipada y tiempo de retraso de propagación y aspectos mucho más específicos que no se cubren en este documento, recordemos que este es un campo de investigación muy amplio. La figura 72 muestra una parte del *deck* generado por este proceso, este *deck* se agrega como una vista Hspice a la celda del proyecto y se procede a verificar con el mismo *testbench* realizado con anterioridad, especificando en la jerarquía que se use el Hspice en lugar del Schematic.

Con el *deck* obtenido se hicieron varias simulaciones, este análisis es la séptima y última etapa de diseño, para la verificación de que el sistema seguía funcionando de forma correcta, el "cableado" que se hace en el *layout* puede ser tan impactante que la aplicación deje de funcionar. Las simulaciones fueron exitosas, sin embargo, se ve la clara presencia de capacitores interconectados en el circuito ya que las salidas presentan la forma de carga y descarga de un capacitor mucho más definida, es decir, una carga o descarga que requiere más tiempo. En la sección de resultados varias son las figuras que muestran el retraso de propagación medido en el circuito, en este caso la ruta crítica es obvia y se obtiene cuando los valores de un vector de entradas son todos 1, y las del otro son todas 0 y el CIN cambia de 0 a 1, o bien el CIN permanece en 0 y el bit menos significativo de un vector, cambia de 0 a 1 mientras que el bit 0 del otro vector está en 1. En cualquiera los dos casos el acarreo de salida de cada sumador completo de 1 bit conmuta de 0 a 1, haciendo que en el siguiente sumador se realice la misma conmutación y así hasta que COUT realiza un flanco de subida, esta es la ruta crítica y el tiempo para realizar esa operación será el tiempo máximo de operación y la potencia disipada por la fuente al realizar esa operación será la potencia promedio máxima por operación y por lo tanto la energía máxima que se requiere para hacer una operación. En el Cuadro 13 se muestran los resultados obtenidos de estas mediciones. Se observa que el retraso de propagación es de 2.12 ns, es un número de vital importancia ya que marcará la frecuencia a la que puede operar el dispositivo que es de 471.7 MHz. Si se compara con los resultados obtenidos en la primera versión del circuito presentada en el capítulo 5 vemos que los resultados de esta versión pueden ser peores, sin embargo esto no es así ya que en la primera versión no se realizó una extracción de parásitos. Para poder comparar de mejor forma las mejoras se hizo el análisis del retraso de propagación sin tomar en cuenta las capacitancias parásitas, esto se muestra en las figuras 63 a 68, se nota un retraso de propagación de 1.05 ns, que produce una frecuencia de operación de 952.38 MHz demostrando así la mejora en el proceso.

Otro aspecto a discutir es la energía usada para realizar la operación más crítica, se observa que la magnitud es de aproximadamente 238 fJ un número bastante pequeño, haciendo un análisis para que el lector pueda comprender un poco mejor lo que esto significa se recuerda la ecuación de energía de un capacitor $E_c = \frac{1}{2}CV^2$ si se sustituye E_c por 238 fJ, se supone un capacitor de 238µF y se despeja para V se obtiene un resultado de 44.72µV, esto quiere decir que se puede realizar la operación crítica del circuito con un capacitor de 238µF cargado a 44.72µV. Sin duda es una cantidad de energía bastante pequeña y este es uno de los objetivos de la mejora continua de los circuitos integrados y de la electrónica en general.

La última parte de este proyecto consta de la implementación del *ring pad,* el *die* ya analizado es el corazón del proyecto pero quedaría incompleto sin poder usarlo en el exterior por lo que es necesario amplificar la corriente de sus salidas y protegerlo de cualquier pico de voltaje en las entradas y de ruido ya sea en la entrada o en la salida, las salidas también poseen protección ESD. Synopsys en sus librerías provee de *pads* estándares que fueron los que se usaron, estos *pads* constan de un circuito de protección contra descargas electroestáticas, un circuito que restablece la señal a niveles de voltaje aceptables y un habilitador / deshabilitador de la entrada o de la salida. Este habilitador no fue utilizado en el proyecto. Las Figuras 82 y 83 muestran las salidas usando los *pads*, se observa que el circuito sigue funcionando correctamente, que la señal de salida tiene un forma más suave sin muchas perturbaciones como la salida del *die*, pero que el retraso de propagación es más del doble, por lo que la nueva frecuencia de operación es de 219.30 MHz variando según la carga que se conecte en el exterior del empaquetado. La Figura 81 muestra el proyecto completamente terminado y funcional. Es importante agregar que en la última parte del proyecto se encontraron algunas librerías con errores de DRC y LVS, estas librerías son D4I1025_NS, IOVSS_NS, IOVDD_NS, VSS_NS, VDD_NS y I1025_NS es muy recomendable solucionar estos problemas antes de proceder con la agregación de *pad ring*, esto aislará y solucionará muchos errores en el resultado final. El resultado final mencionado con *pad ring* posee el problema de muchos errores de DRC y LVS y esto se debe a los problemas mencionados con las librerías de los *pads* que requieren mucho tiempo para su solución, específicamente, estas librerías usan transistores grandes denominados n25 para amplificación de corriente, la implementación de estos transistores en el *layout* es la que presenta los problemas. Quedará como una mejora propuesta la solución del circuito de *pad ring*, aunque el circuito es completamente funcional, la parte de verificación física es muy importante.

VIII. CONCLUSIONES Y RECOMENDACIONES

H. Conclusiones

1. El laboratorio de electrónica de la Universidad del Valle de Guatemala cuenta con aplicaciones de diseño profesionales de circuitos a nanoescala.
2. Se ha generado exitosamente el primer circuito a muy gran escala usando las aplicaciones profesionales proveídas por Synopsys.
3. Se obtuvo un retraso de propagación final de 2.12 ns en el die y de 4.56 ns con la agregación de *pads*.
4. Se comprueba que la mejora en la tecnología usada para la implementación de circuitos integrados provee ventajas energéticas considerables.
5. Algunas librerías de diseño contienen errores de conexión.
6. No se realizó el método de *logical effort* debido a la utilización de librerías estándares de diseño.

I. Recomendaciones

1. Familiarizarse con el sistema operativo CentOS antes de empezar algún diseño.
2. Conocer los directorios importantes de instalación de las aplicaciones y ubicación de archivos de las librerías que se pueden agregar a Custom Designer.
3. Seguir las fases de diseño sin obviar ninguna, al terminar de realizar alguna fase es una buena práctica realizar copias de seguridad de lo obtenido.
4. Usar la documentación de las aplicaciones en caso se requiera una tarea muy específica.
5. Usar algún programa de conexión remota para trabajar fuera del campus de la Universidad.
6. Verificar cada una de las librerías de forma individual para descartar errores en las mismas.

IX. BIBLIOGRAFÍA

Anónimo. (21 de agosto de 2002). *Inauguran el Laboratorio de Microelectrónica y Diseño VLSI del ITC*. Recuperado el 5 de julio de 2014, de http://noticias.universia.com.ar/en-portada/noticia/2002/08/21/383857/inauguran-laboratorio-microelectronica-diseno-vlsi-itc.html

Farrow, M. (2011). *Microprocessor Transistor Counts*. Recuperado el 13 de 10 de 2014, de http://mikefarrow.co.uk/work/dissertation

Journal, M. (2003). *Gold Stud Bumps in Flip-chip Applications*. Recuperado el 5 de noviembre de 2014, de http://www.microwavejournal.com/articles/print/3622-gold-stud-bumps-in-flip-chip-applications

Kem, K. (2014). *Design of a 1-bit adder-subtractor with additional carry/borrow input*. Recuperado el 1 de 8 de 2014, de http://electronics.stackexchange.com/questions/101882/design-of-a-1-bit-adder-subtractor-with-additional-carry-borrow-input

Mamani, G. (2010). *Instalar GNU/Linux CentOS 6 en VMware 10*. Recuperado el 7 de julio de 2014, de http://isyskernel.blogspot.com/2014/01/instalar-gnulinux-centos-65-en-vmware-10.html

Pepone, D. X. (11 de febrero de 2014). *Las complicaciones de la Ley de Moore en la actualidad*. Recuperado el 5 de julio de 2014, de http://www.mundopepone.com/2014/02/11/las-complicaciones-de-la-ley-de-moore-en-la-actualidad/

Synopsys. (2014). *32/28nm Interoperable PDK*. Recuperado el 1 de agosto de 2014, de http://www.synopsys.com/Community/UniversityProgram/Pages/32-28nm-ipdk.aspx

Synopsys. (2014). *Custom Designer*. Recuperado el 7 de agosto de 2014, de https://solvnet.synopsys.com/dow_retrieve/latest/ni/cd.html

Synopsys. (2014). *Custom WaveView™*. Recuperado el 15 de 8 de 2014, de https://solvnet.synopsys.com/dow_retrieve/latest/ni/wv.html

Synopsys. (2014). *Extraction & Layout Analysis*. Recuperado el 16 de septiembre de 2014, de https://solvnet.synopsys.com/dow_retrieve/latest/ni/extract.html#StarRC

Synopsys. (2014). *HSPICE*. Recuperado el 8 de 8 de 2014, de https://solvnet.synopsys.com/dow_retrieve/latest/ni/hspice.html#HSPICE

Synopsys. (2014). *IC Validator*. Recuperado el 11 de 8 de 2014, de https://solvnet.synopsys.com/dow_retrieve/latest/ni/icv.html

Synopsys. (2014). *PyCell Studio*. Recuperado el 28 de 8 de 2014, de https://www.synopsys.com/TOOLS/IMPLEMENTATION/CUSTOMIMPLEMENTATION/Pages/pycell-studio.aspx

Synopsys. (2014). *Synopsys Licensing QuickStart Guide*. Recuperado el 3 de julio de 2014, de http://www.synopsys.com/Support/LI/Licensing/Pages/default.aspx

Synopsys. (2014). *TSMC - Synopsys Collaboration*. Recuperado el 5 de julio de 2014, de https://www.synopsys.com/COMMUNITY/PARTNERS/TSMC/Pages/default.aspx

TSMC. (2014). *20nm Technology*. Recuperado el 5 de julio de 2014, de http://www.tsmc.com/english/dedicatedFoundry/technology/20nm.htm

Universidad del Valle de Guatemala. (2013). *Sumador de 32 bits con tecnología CMOS 130 nm.* Guatemala.

Weste, N. H., & Money Harris, D. (2011). *CMOS VLSI Design. A Circuits and Systems Perspective* (4ta ed.). United States of America: Addison-Wesley, PEARSON.

Xedoloh, H. (2011). *Processor Basics*. Recuperado el 13 de octubre de 2014, de http://hewo.xedoloh.com/2011/10/processor-basics/

X. GLOSARIO

B

bits
En este contexto se refiere a dígitos binarios. · 6, vi, 8, 10, 11, 42, 54, 55, 57

C

CDL
Circuit Design Language · 36, 37, 38
CentOS
Sistema operativo base Linux, usa el kernel liberado por Red Hat. · vi, 12, 13, 17, 18, 20, 21, 27, 30
Custom Designer
Plataforma de diseño VLSI · 30, 31, 32, 33, 34, 36, 37, 38, 39, 48, 51, 74, 75, 78, 79

D

deck
Archivo que contiene a un circuito descrito en texto plano. · 37, 42, 50, 51, 52, 75, 76
die
Bloque de material semiconductor donde se ha realizado un circuito funcional. · 7, 8, 10
DRC
Proceso de verificación de reglas de diseño. · 28, 47, 48, 49, 50, 57

G

gnome
Ambiente gráfico de Linux · 17

H

HDL
Hardware description language · 28, 53
host
Dispositivo de usuario final en una red. · 30

I

iPDK
Interoperable Process Design Kits · 31, 32, 33, 37, 38, 42, 43, 44, 47, 50, 51, 52

L

layout
Parte física del circuito donde se definen todas las interconexiones en diferentes capas de metal. · 8, 10, 31, 32, 33, 36, 37, 43, 44, 45, 46, 47, 48, 50, 51, 66, 75
Ley de Moore
Sugiere que cada 2 años se duplica el número de transistores en un circuito integrado. · 11
LPE
Proceso de extracción de capacitancias, inductancias y resistencias parásitas. · 51, 52
LVS
Proceso de verificación de circuito esquemático contra su respectivo Layout. · 28, 44, 47, 50, 51, 52, 57

N

nanoelectrónica
es la rama de la electrónica referente a los circuitos electrónicos miniaturizados integrados en chips semiconductores, siendo su elemento de base el transistor. · vi, 12
nanoescala
Una medida en el orden de los nanómetros. · 8
netlist
Archivo de texto que contiene la especificación de un circuito. · 36, 37, 41, 52
Netlist
Definición de un circuito en texto plano. · 8, 36, 42, 50, 52

P

pad
Circuito de amplificación y protección para conectar un die a un empaquetado. · 9, 76, 78

R

S

T

V

X

XI. ANEXOS

A. Video tutorial

https://www.youtube.com/watch?v=PuYQJpAKRSE

B. Artículo científico

En las siguientes páginas se encontrará un artículo científico extraído de este trabajo donde se tocan los puntos más importantes de esta investigación. El artículo contiene principalmente el proceso de diseño del Sumador / Restador de 32 bits.

Diseño de un sumador/restador completo de 32 bits con tecnología de 28 nanómetros (Noviembre 2014)

Jonathan de los Santos (09375), *Ingeniería electrónica, UVG*

***Abstract*—** Nanoelectronics is a branch of electronics that specializes in the design of integrated circuits, these ICs have a large number of components inside. The process of designing an integrated circuit is complex because of numerous rules that must be met so that the layout can be implemented by a company dedicated to manufacturing as TSMC. This paper describes the design process of an adder / subtractor 32 bit using 28-nanometer technology design applications company Synopsys described. Design phases suggests that Synopsys are made and then some data characterizing the final design are measured.

***Resumen*—** La nanoelectrónica es una rama de la electrónica que se dedica al diseño de circuitos integrados, estos circuitos integrados poseen un gran número de componentes en su interior. El proceso de diseño de un circuito integrado es complejo debido a numerosas reglas que se deben cumplir para que el layout pueda ser implementado por alguna empresa dedicada a la manufacturación como la TSMC. En este artículo se describe el proceso de diseño de un sumador/restador de 32 bits con tecnología de 28 nanómetros usando aplicaciones de diseño profesional de la empresa Synopsys. Se cumplen con las fases de diseño que Synopsys sugiere y luego se miden algunos datos que caracterizan el diseño final.

***Palabras clave*— VLSI, DRC, LVS, LPE, Layout, transistores CMOS, Synopsys.**

I. Introducción

Las aplicaciones de diseño de circuitos integrados son de gran utilidad, estos circuitos pueden asemejarse a cualquier otro en cuanto a funcionamiento final, pero durante el proceso de diseño es necesario tomar muchos factores en cuenta. El proceso de fabricación de estos circuitos es bastante complejo, se usan máquinas muy precisas para lograr la manufacturación de componentes tan pequeños. La cantidad de transistores que se ha logrado integrar en una pequeña pieza de silicio en los tiempos actuales es impresionante, tomando en cuenta que en 1971 el procesador 4004 tenía aproximadamente 2300 transistores y uno de los procesadores de unos 43 años después tiene más de 2,600,000,000 transistores, denota un crecimiento y mejora impresionante. [1]

El proyecto que se presenta en este artículo se trata del diseño de un sumador/restador de 32 bits usando aplicaciones profesionales de la empresa Synopsys, el proyecto se limita al diseño y no se implementará por razones de costo y tiempo. [2]

II. Fases de diseño

Galaxy Custom Designer es un ambiente de diseño que permite la realización de circuitos a nanoescala de una manera eficiente usando celdas de diseño estándar. Esto quiere decir que existen circuitos conocidos a los que se le ha implementado un layout, un esquemático y se han caracterizado para poder usarlos de forma confiable en diseños más complejos. Según la documentación de Galaxy Custom Designer son 7 las fases de diseño, las cuales se presentan a continuación. [2]

A. Circuito esquemático.

El diseño comienza al nivel de conexión de los diferentes componentes en un diagrama esquemático, cada componente debe tener sus propiedades bien editadas para que las siguientes fases muestren resultados fiables. Un diagrama esquemático es la conexión de los componentes con sus propiedades físicas correctas. El editor esquemático es muy parecido a los programas que se usan para la simulación de circuitos, tiene las características de orientación (o transformación) que gira o refleja los componentes para hacer más entendible el diagrama. Para un diseño más ordena y más profesional se recomienda realizar bloques, al estilo de las conocidas cajas negras, donde se tengan entradas y salidas, conocer la función que los relaciona pero no forzosamente tener que saber qué hay dentro, esta es una buena técnica para reciclar trabajos antiguos.

B. Circuito simulaciones.

Una vez que ya se tiene el esquemático y el símbolo agregado a la vista de nuestra celda es necesario saber si funciona correctamente. Recordar que el símbolo hace referencia al conjunto de transistores que componen el circuito y es una forma compacta de verlo, si hay algo no esperado en los resultados de simulación el problema estará en los transistores y no en el símbolo. La forma correcta de hacer una simulación es hacer otra celda para el *testbench*, esto porque no se puede instanciar una celda dentro de sí misma. Dentro de esta celda para *testbench* se instancia el símbolo del componente que se quiere probar y usando fuentes de tensión se le inyectan señales varias para analizar su funcionamiento. [4]

C. Análisis

La etapa de análisis es una etapa donde se debe verificar que los datos obtenidos son los esperados, en este punto es necesario estar seguros que todo funciona correctamente, que los niveles de voltaje en salidas y entradas son correctos, etc. No es necesario, en esta

etapa analizar tanto en aspectos como potencia, energía, tiempo o alguna otra variable muy específica ya que en este punto no se tiene un circuito final porque faltan los parásitos que generará el Layout. Existen varias aplicaciones para analizar los datos obtenidos como Custom WaveView, Custom Explorer, Custom Explorer Ultra y CosmosScope.

D. Schematic-Driven Layout

En esta fase de diseño se realiza la implementación del layout que será el corazón del proyecto que se realiza, un buen diseño del layout garantiza menos potencia, menos energía, tiempo de respuesta menor. En el mundo de VLSI el diseño puede tener miles o hasta millones de transistores, no es nada eficiente realizar el diseño transistor por transistor, esta opción podría llegar a ser lo mejor en cuanto a desempeño pero llevará varios días, meses o años, dependiendo del proyecto, es por eso que se usa un diseño en base a celdas, a librerías de circuitos comunes ya realizadas que se colocan estratégicamente para lograr una mayor densidad de transistores por unidad de área, luego de colocar las celdas se procede al proceso de ruteo que une a cada una de las celdas y a los puertos de entrada y salida. Se recomienda hacer una división en módulos del diseño y realizar una combinación estratégica del *Generate Layout* y el *Pick and Place*. [3]

E. DRC y LVS

En esta fase de diseño se realiza la verificación de las reglas de diseño (*Design Rules Checking*) y la verificación física del diseño (*Layout Vs. Schematic*). Esta fase es crucial para lograr un diseño que pueda ser fabricado, es necesario cumplir con las reglas de diseño para que la empresa de manufactura pueda implementar lo requerido y es necesario que el esquemático, que ya se sabe que funciona correctamente por la fase de simulación y análisis, sea idéntico a lo implementado en el Layout. Las violaciones encontradas son fáciles de ubicar y reparar manualmente debido a la interacción que se tiene mediante el sistema operativo, IC Validator o Hercules y Custom Designer. Cada vez que se resalta un error el editor de layout se magnifica en las coordenadas donde se encontró la violación la cual se marca con una malla amarilla. [5]

F. LPE

Esta sección corresponde a la obtención de un deck que se asemeje más a la realidad en cuanto a simulación, esto es debido a que se ejecuta un proceso donde se calculan de manera muy acertada todas las capacitancias, inductancias y resistencias parásitas que se generan con el diseño de layout que hasta este paso ya se tiene. El proceso es sencillo y de alta utilidad, la aplicación que se integra al entorno de Galaxy Custom Designer es StarRC y como requisito ya se debió haber realizado el LVS ya que se necesitan algunos archivos generados en este proceso. Como resultado, al terminar el proceso, se obtiene un deck con todo el circuito a nivel transistor más capacitores y resistencias (extracción RC) interconectados entre sí que simulan las capacitancias y resistencias generadas entre las distintas capas de metal del layout. [6]

G. Verificación Post-Layout.

Esta fase de diseño es una fase extremadamente importante ya que consiste en poder caracterizar nuestro circuito. Caracterizar se refiere a conocer aspectos del circuito que son cruciales en su funcionamiento, como la potencia disipada, la energía necesaria, y el tiempo máximo que se tarda para realizar una operación. Esta última característica resulta un poco difícil de encontrar, ya que es necesario analizar cada una de las rutas que se usan para lograr un resultado, con todas las posibles combinaciones en los vectores de entrada. En ocasiones la ruta más larga puede ser obvia, o si no lo es puede ser fácil realizar iteraciones con las combinaciones en los vectores de entrada que produzcan un cambio en la salida, pero en proyectos muy complicados no es ni obvio ni de fácil iteración. Para esto se usan aplicaciones específicas como PrimeTime y NanoTime, la primera de ellas a nivel de compuertas, cuando se tenga el circuito descrito con un archivo HDL de Verilog y la segunda es a nivel de transistores. El uso de las dos aplicaciones mencionadas con anterioridad va más allá del alcance de este artículo. [7]

III. Diseño Experimental

El circuito que se realizó es el siguiente con la diferencia que fue extendido a 32 bits. [3]

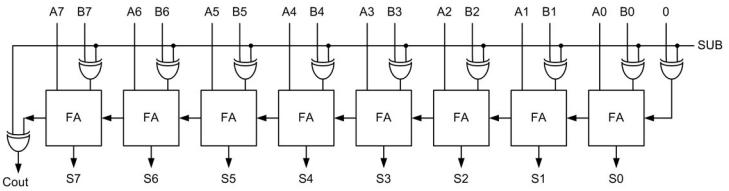


Figura 1: Circuito a implementar

Sobre este circuito se deben realizar las 7 fases de diseño usando la iPDK de que Synopsys provee.

IV. RESULTADOS

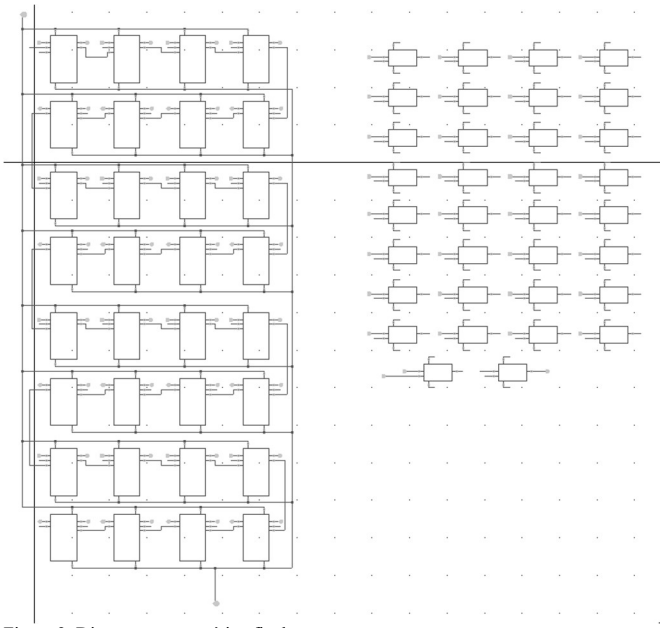


Figura 2: Diagrama esquemático final

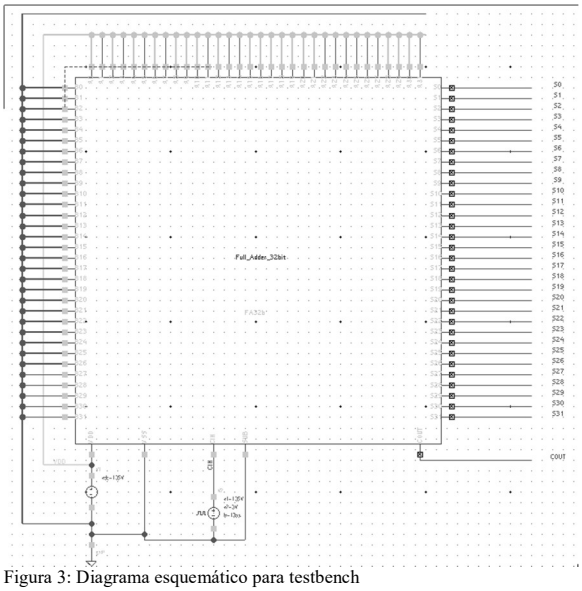


Figura 3: Diagrama esquemático para testbench

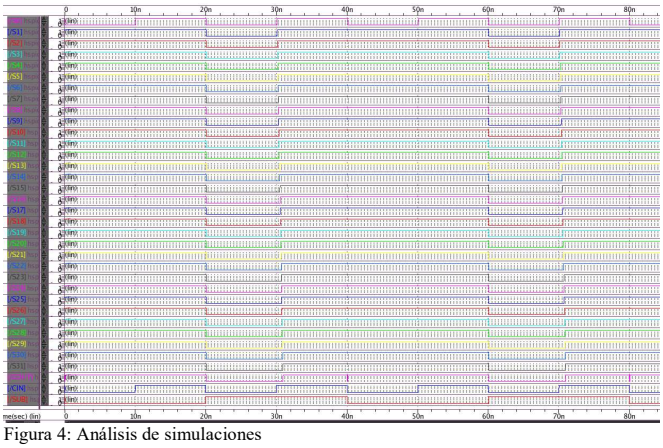


Figura 4: Análisis de simulaciones

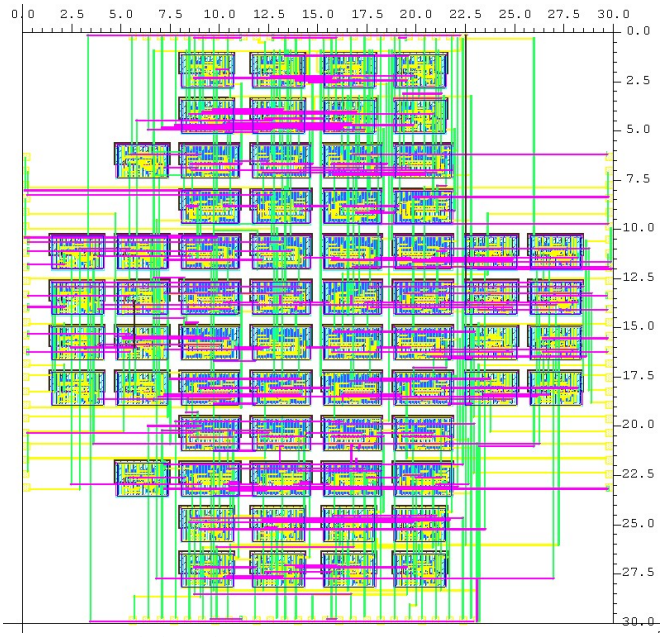


Figura 5: Layout de die final

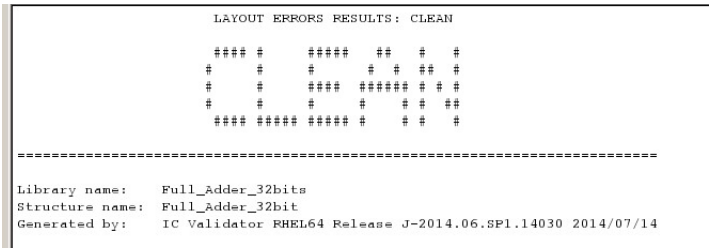


Figura 6: DRC Exitoso.

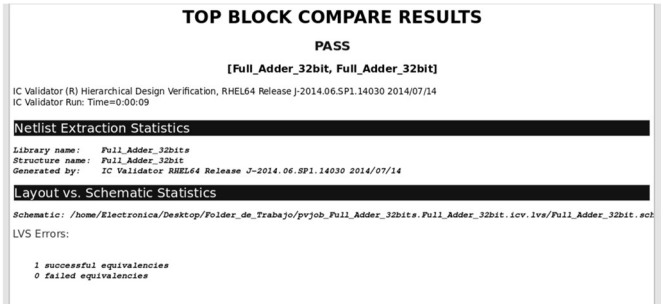


Figura 7: LVS Exitoso.

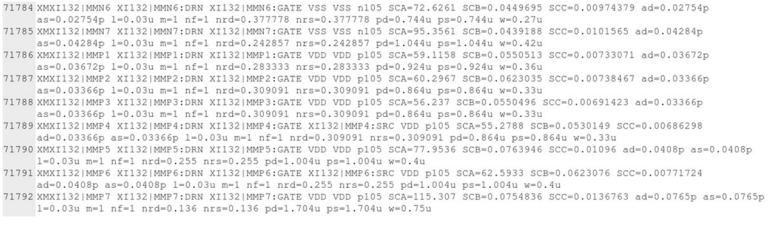


Figura 8: Deck con parásitos (muestra)

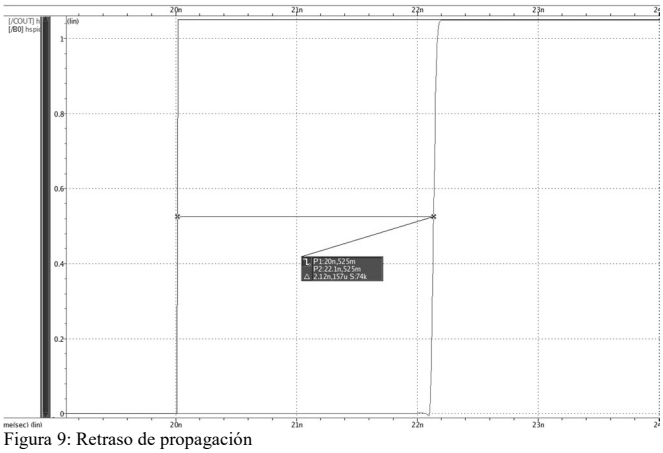


Figura 9: Retraso de propagación

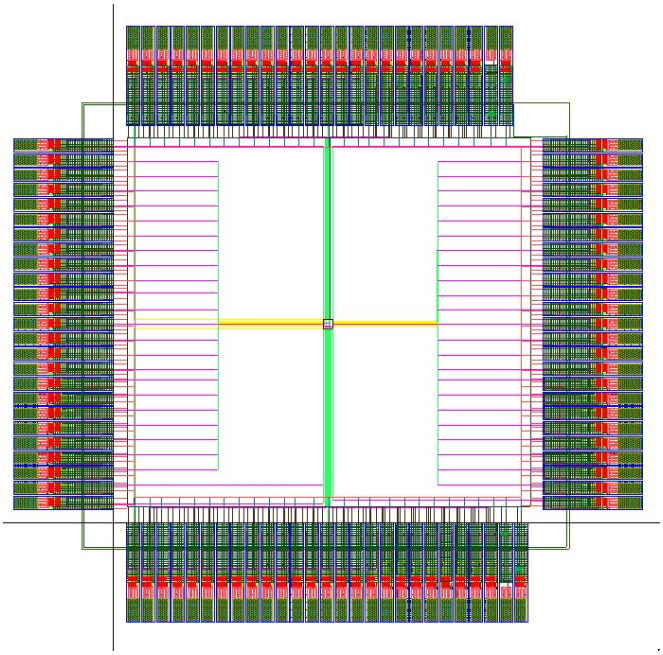


Figura 10: Layout final con agregación de pads.

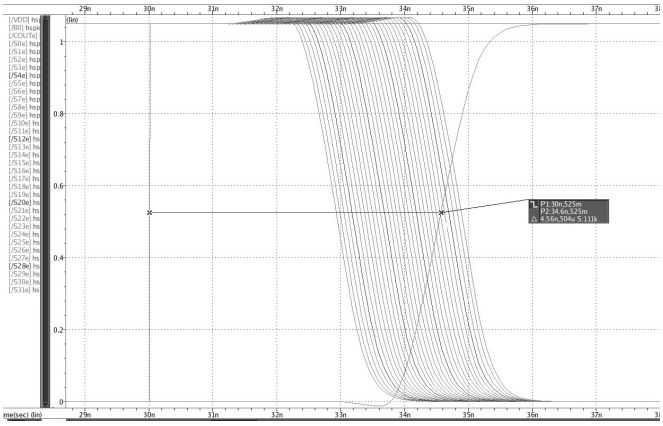


Figura 11**:** Retraso de propagación con pads.

TABLA I

CARACTERIZACIÓN DEL CIRCUITO

Aspecto	**Especificación**
Tecnología	CMOS de 28 nm
Área de silicio (*die*)	30 x 30 $(\mu m)^2$
Retraso de propagación	2.12 ns.
Frecuencia máxima de operación	471.7 MHz
Potencia promedio	-112.7054 µW
Energía	-238.0362 fJ
Número de transistores	1372
Densidad	152.4 MT/(cm)2
Retraso de propagación (*pads*)	4.56 ns

V. DISCUSIÓN DE RESULTADOS

La sección anterior muestra un proceso de diseño con resultados satisfactorios, estos resultados fueron obtenidos después de muchas pruebas y correcciones tanto en la plataforma de trabajo como en el diseño en sí. Los problemas en la plataforma de trabajo resultaron de una instalación correcta pero de una configuración en el sistema operativo CentOS incompleta. Por ejemplo, cuando se necesitaba realizar el proceso de extracción de parásitos no se podía comunicar el Galaxy Custom Designer con StarRC, el log de la ejecución mostraba un error que indicaba que uno de los archivos necesarios faltaba, este problema similar a muchos otros ocurridos con las demás aplicaciones surgía debido a la falta de alguna variable de entorno o la que existía apuntaba a un directorio incorrecto. [7] Las variables de entorno son de vital importancia para que las aplicaciones se puedan comunicar entre sí y también con el sistema operativo en general. Desde un principio se sabía que se iban a tener problemas de este tipo por la experiencia que se tiene en el uso de diversas distribuciones de Linux; aunque los manuales de instalación se siguieron al pie de la letra algunos de ellos no especificaban el nombre de la variable de entorno o la dirección que se sugería era incorrecta, quizá por cambios en las nuevas versiones. Como primera aplicación desarrollada en este ambiente de trabajo se pretendía superar todos estos problemas con las aplicaciones, dejar todas las configuraciones en un estado óptimo para permitir que en el futuro se dedique únicamente al diseño sin tener problemas con el entorno de trabajo. Un archivo de vital importancia es `$HOME/.bashrc` este archivo es un archivo de personalización del usuario hacia el sistema operativo, en el entorno Linux se puede usar para diversas cosas, en este trabajo se ha usado para especificar las variables de entorno necesarias para el correcto funcionamiento de las aplicaciones de diseño. Es importante notar que no es posible especificar las variables de entorno para todas las aplicaciones que Synopsys provee por lo que es recomendable, al instalar una aplicación leer el manual de instalación, específicamente la sección de configuración con el entorno. Las aplicaciones quedaron apropiadamente instaladas dando al laboratorio de electrónica de la Universidad del Valle de Guatemala la suite de herramientas profesionales para diseño de circuitos a muy gran escala.

Synopsys recomienda el diseño en 7 fases que tienen un orden lógico y hacen muy eficiente la obtención de un resultado final. A continuación se irán detallando cada una de las fases haciendo refe-

rencia a los resultados obtenidos y mencionando los hallazgos encontrados. La primera fase es la realización del diagrama esquemático, la figura 2 muestra el diseño esquemático final como primera fase de diseño. El circuito consta de varios sumadores completos de 1 bit puestos en cascada como lo muestra la figura 1 y compuertas XOR que con la adición de la entrada CIN convierten uno de los vectores de entrada en complemento a 2. Es importante mencionar que cada una de estas pequeñas cajas negras está construida a nivel transistor en su interior.

Siguiendo el proceso de diseño se observa en la figura 3 que se creó una caja negra, que es el sumador/restador completo con todos los pines de entrada y salida que tendría ya implementado fuera del empaquetado final, este esquemático para pruebas (*testbench*) se realizó con dos objetivos, el primero fue para verificar el correcto funcionamiento del diseño, que es la segunda fase de diseño, y el segundo dejar una instancia que pueda ser usada en otros diseños que requieran de un sumador/restador de 32 bits, esta es una buena práctica para hacer más simples los procesos de diseño e implementación. Al circuito de la figura 3 se le inyectaron señales para poder proceder con las simulaciones respectivas y analizar el resultado de las mismas. La versatilidad del graficador que ofrece la aplicación Custom Explorer y la integración con Custom Designer hizo verdaderamente simple esta labor. La figura 4 muestra el resultado de las simulaciones graficando cada señal de salida con respecto al tiempo. Las señales cambian debido al cambio que se hizo intencionalmente en las entradas CIN y SUB para analizar tanto la suma de los números binarios como la resta de los mismos. El análisis de los resultados de la simulación es la 3ra etapa de diseño, note que en esta fase debemos obtener los resultados que deseamos, no tanto en aspectos como potencia o tiempos de operación pero sí en los niveles de voltaje que se esperan en la salida. Por ejemplo, en el caso del de este proyecto, en esta fase ya debe ser posible realizar sumas y restas, las pruebas se realizaron a una frecuencia adecuada para saber que el circuito sí funciona. Varias veces fueron las que no se obtuvieron resultados satisfactorios, al principio de creyó que era por la velocidad en la que se realizaban las transiciones de los estados por lo que se disminuyó la frecuencia de operación, y seguía sin funcionar. Todo indicaba que era una mala conexión en la circuitería que conforma el esquemático y así fue. Puede ser una tarea realmente difícil encontrar este tipo de errores, como el lector seguramente sabe por simulaciones realizadas en otros entornos, por lo que se recomienda tener mucho cuidado en la primera fase de diseño. Como es lógico, sin tener los resultados que se esperan no se puede pasar a las siguientes etapas de diseño.

La cuarta etapa de diseño es la generación del Layout, una fase que probablemente todos los interesados en la nanoelectrónica están interesados en llegar. La maravilla que ahora son las aplicaciones de diseño como las que Synopsys provee hacen de esta tarea, una labor relativamente simple. El proceso se denomina Schematic Driven Layout (SDL) como ya se ha explicado en las guías rápidas de uso de las aplicaciones que conforman el marco teórico de este documento. En esta fase se usó una combinación estratégica de posicionamiento de las instancias en el editor de *layout* del generador automático y del *pick and place* esto para tener un diseño más ordenado y probablemente más eficiente en cuanto a área. Luego de tener las instancias debidamente alineadas y debidamente ubicadas se procede a realizar el proceso de auto ruteo, esta función normalmente la logra hacer en su totalidad Galaxy Custom Designer pero suele fallar cuando se le indica que rutee todas las pistas de golpe. Lo que se hizo fue rutear todas las rutas y al terminar el proceso en la "Custom Designer Console" se imprime un sumario de cuáles rutas fueron completadas y cuáles no, de las que no fueron ruteadas se repite el proceso indicando que rutee sólo éstas faltantes y así hasta completar el 100% del ruteo. Si en algún caso no fuera posible por parte del programa hacer el ruteo puede rutearse a mano, estos errores se encontrarán en las siguientes fases de diseño. La figura 5 muestra el *die* final obtenido.

La quita etapa es la de la verificación física del diseño, los errores obtenidos en el proceso de diseño fueron detecciones de violaciones en las reglas de diseño y errores en la comparación del esquemático con el layout generado. Los problemas de DRC aunque en general fue un número grande de violaciones encontradas fueron fáciles de solucionar debido a que el programa IC Validator como ya se ha mencionado se comunica con el Layout Editor de Custom Designer y ubica el error marcándolo con una malla amarilla (que puede ser modificada desde IC Validator). En el proceso LVS también se encontraron algunas desigualdades, LVS por ser un proceso de IC Validator también se comunica con Custom Designer e identifica y ubica el error de la misma forma que DRC. Los problemas que se encuentren durante la ejecución de LVS son producto de un mal ruteo en las distintas capas de metal, y un mal ruteo puede originarse por alguna deficiencia en la configuración del Auto Route o porque simplemente el software no fue capaz de hacer la conexión con las restricciones impuestas. Normalmente el Auto Route realiza la mayoría de conexiones sin ningún problema, en el proceso del diseño del sumador/restador fueron 2 conexiones las que no se conectaron automáticamente las cuales fueron detectadas fácilmente con los resultados del LVS y fueron realizadas las conexiones manualmente que también se ofrece una forma muy intuitiva de realizarlo. Se puede ver en las figuras 6 y 7 el resultado satisfactorio de las verificaciones DRC y LVS respectivamente.

Hasta este punto ya se tiene un layout completamente terminado, pero es necesario analizar cómo se comportará el circuito con este diseño de layout, las simulaciones realizadas con anterioridad muestran el correcto funcionamiento a nivel esquemático, pero no se toman en cuenta los efectos que el layout agrega al circuito. Para poder analizar estos efectos es necesario ejecutar un proceso llamado LPE (Layout Parasitic Extaction), que realiza una extracción de capacitancias, inductancias y resistencias parásitas y las interconecta en un *deck* con el resto del circuito a nivel de transistores, esta es la sexta etapa de diseño. Este análisis es crucial para la verificación de potencia disipada y tiempo de retraso de propagación y aspectos mucho más específicos que no se cubren en este documento, recordemos que este es un campo de investigación muy amplio. La figura 8 muestra una parte del *deck* generado por este proceso, este *deck* se agrega como una vista Hspice a la celda del proyecto y se procede a verificar con el mismo *testbench* realizado con anterioridad, especificando en la jerarquía que se use el Hspice en lugar del Schematic.

Con el *deck* obtenido se hicieron varias simulaciones, este análisis es la séptima y última etapa de diseño, para la verificación de que el sistema seguía funcionando de forma correcta, el "cableado" que se hace en el layout puede ser tan impactante que la aplicación deje de funcionar. Las simulaciones fueron exitosas sin embargo se ve la clara presencia de capacitores interconectados en el circuito ya que las salidas presentan la forma de carga y descarga de un capacitor

mucho más definida, es decir, una carga o descarga que requiere más tiempo. En la sección de resultados varias son las figuras que muestran el retraso de propagación medido en el circuito, en este caso la ruta crítica es obvia y se obtiene cuando los valores de un vector de entradas son todos 1, y las del otro son todas 0 y el CIN cambia de 0 a 1, o bien el CIN permanece en 0 y el bit menos significativo de un vector, cambia de 0 a 1 mientras que el bit 0 del otro vector está en 1. En cualquiera los dos casos el acarreo de salida de cada sumador completo de 1 bit conmuta de 0 a 1, haciendo que en el siguiente sumador se realice la misma conmutación y así hasta que COUT realiza un flanco de subida, esta es la ruta crítica y el tiempo para realizar esa operación será el tiempo máximo de operación y la potencia disipada por la fuente al realizar esa operación será la potencia promedio máxima por operación y por lo tanto la energía máxima que se requiere para hacer una operación. En la tabla 1 se muestran los resultados obtenidos de estas mediciones. Se observa que el retraso de propagación es de 2.12 ns, es un número de vital importancia ya que marcará la frecuencia a la que puede operar el dispositivo que es de 471.7 MHz.

Otro aspecto a discutir es la energía usada para realizar la operación más crítica, se observa que la magnitud es de aproximadamente 238 fJ un número bastante pequeño, haciendo un análisis para que el lector pueda comprender un poco mejor lo que esto significa se recuerda la ecuación de energía de un capacitor $E_c = \frac{1}{2}CV^2$ si se sustituye E_c por 238 fJ, se supone un capacitor de 238µF y se despeja para V se obtiene un resultado de 44.72µV, esto quiere decir que se puede realizar la operación crítica del circuito con un capacitor de 238µF cargado a 44.72µV. Sin duda es una cantidad de energía bastante pequeña y este es uno de los objetivos de la mejora continua de los circuitos integrados y de la electrónica en general. [10]

La última parte de este proyecto consta de la implementación del *pad ring* el *die* ya analizado es el corazón del proyecto pero quedaría incompleto sin poder usarlo en el exterior por lo que es necesario amplificar la corriente de sus salidas y protegerlo de cualquier pico de voltaje en las entradas y de ruido ya sea en la entrada o en la salida, las salidas también poseen protección ESD. Synopsys en sus librerías provee de *pads* estándares que fueron los que se usaron, estos *pads* constan de un circuito de protección contra descargas electroestáticas, un circuito que restablece la señal a niveles de voltaje aceptables y un habilitador / deshabilitador de la entrada o de la salida. Este habilitador no fue utilizado en el proyecto. La figuras 9 muestra las salidas usando los *pads*, se observa que el circuito sigue funcionando correctamente, que la señal de salida tiene un forma más suave sin muchas perturbaciones como la salida del *die*, pero que el retraso de propagación es más del doble, por lo que la nueva frecuencia de operación es de 219.30 MHz variado según la carga que se conecte en el exterior del empaquetado. La figura 9 muestra el proyecto completamente terminado y funcional.

VI. Conclusiones y Recomendaciones

A. Conclusiones

1. El laboratorio de electrónica de la Universidad del Valle de Guatemala cuenta con aplicaciones de diseño profesionales de circuitos a nanoescala.
2. Se ha generado exitosamente el primer circuito a muy gran escala usando las aplicaciones profesionales proveídas por Synopsys.
3. Se obtuvo un retraso de propagación final de 2.12 ns.
4. Se comprueba que la mejor en la tecnología usada para la implementación de circuitos integrados provee ventajas energéticas considerables.

B. Recomendaciones

1. Familiarizarse con el sistema operativo CentOS antes de empezar algún diseño.
2. Conocer los directorios importantes de instalación de las aplicaciones y ubicación de archivos de las librerías que se pueden agregar a Custom Designer.
3. Seguir las fases de diseño sin obviar ninguna, al terminar de realizar alguna fase es una buena práctica realizar copias de seguridad de lo obtenido.
4. Usar la documentación de las aplicaciones en caso se requiera una tarea muy específica.
5. Usar algún programa de conexión remota para trabajar fuera del campus de la Universidad.

VII. Bibliografía

[1] M. Farrow, «Microprocessor Transistor Counts,» 2011. [En línea]. Available: http://mikefarrow.co.uk/work/dissertation. [Último acceso: 13 de octubre de 2014].

[2] TSMC, «20nm Technology,» 2014. [En línea]. Available: http://www.tsmc.com/english/dedicatedFoundry/technology/20nm.htm. [Último acceso: 5 de julio de 2014].

[3] Synopsys, «Custom Designer,» 2014. [En línea]. Available: https://solvnet.synopsys.com/dow_retrieve/latest/ni/cd.html. [Último acceso: 7 de agosto de 2014].

[4] Synopsys, «Custom WaveView™,» 2014. [En línea]. Available: https://solvnet.synopsys.com/dow_retrieve/latest/ni/wv.html. [Último acceso: 15 de agosto de 2014].

[5] Synopsys, «IC Validator,» 2014. [En línea]. Available: https://solvnet.synopsys.com/dow_retrieve/latest/ni/icv.html. [Último acceso: 11 de agosto de 2014].

[6] Synopsys, «Extraction & Layout Analysis,» 2014. [En línea]. Available: https://solvnet.synopsys.com/dow_retrieve/latest/ni/extract.html#StarRC. [Último acceso: 16 de septiembre de 2014].

[7] Synopsys, «HSPICE,» 2014. [En línea]. Available: https://solvnet.synopsys.com/dow_retrieve/latest/ni/hspice.html#HSPICE. [Último acceso: 8 de agosto de 2014].

[8] K. Kem, «Design of a 1-bit adder-subtractor with additional carry/borrow input,» 2014. [En línea]. Available: http://electronics.stackexchange.com/questions/101882/design-of-a-1-bit-adder-subtractor-with-additional-carry-borrow-input. [Último acceso: 1 de agosto de 2014].

[9] Synopsys, «Synopsys Licensing QuickStart Guide,» 2014. [En línea]. Available: http://www.synopsys.com/Support/LI/Licensing/Pages/default.aspx. [Último acceso: 3 de julio de 2014].

[10] N. . H. Weste y D. Money Harris, CMOS VLSI Design. A Circuits and Systems Perspective, 4ta ed., United States of America: Addison-Wesley, PEARSON, 2011.