UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Automatización del proceso de instalación de software de Synopsys e implementación de mejoras utilizando contenedores

Trabajo de graduación presentado por Paola Alejandra Mendizábal Batres para optar al grado académico de Licenciada en Ingeniería Electrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

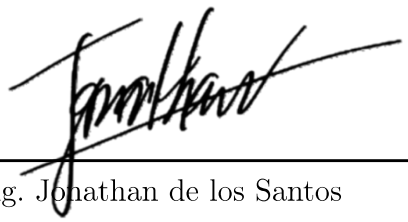
Automatización del proceso de instalación de software de Synopsys e implementación de mejoras utilizando contenedores

Trabajo de graduación presentado por Paola Alejandra Mendizábal Batres para optar al grado académico de Licenciada en Ingeniería Electrónica

Guatemala,

2024

Vo.Bo.:

(f) 

Ing. Jonathan de los Santos

Tribunal Examinador:

(f) 

Ing. Jonathan de los Santos

(f) 

MSc. Carlos Esquit

(f) 

Ing. Kurt Kellner

Fecha de aprobación: Guatemala, 06 de enero de 2023.

Prefacio

Cuando yo era pequeña pensaba que mi trabajo de graduación -fuera lo que fuera- iba a cambiar el mundo. Uno de niño es muy idealista. Conforme uno crece se da cuenta que ese cambiar el mundo se vuelve más bien en cambiar un mundito: cambiar una parte del mundo pequeña pero valiosa.

Me siento contenta con los resultados de mi trabajo. Espero que les haya facilitado el trabajo a mis compañeros/as y próximos colegas que planean continuar con el diseño del chip. Es un proyecto ambicioso que le abrirá las puertas de la nanotecnología, materia que usualmente solo se ve en países del llamado primer mundo, a muchos guatemaltecos.

Quiero agradecer especialmente al Ingeniero Esquit, quien se ha enfocado en hacer esto posible y ha concentrado sus energías en regresar los conocimientos que ha adquirido afuera, a Guatemala.

También quiero agradecer a mi asesor, Jonathan de los Santos, quien me compartió herramientas de gran utilidad y me ayudó a confiar en mi capacidad para realizar un buen trabajo. De igual manera, a Julio Shin, quien repetidas veces me apoyó resolviendo dudas sobre contenedores y otras herramientas útiles a mi trabajo.

Finalmente a mi familia, quienes siempre confiaron en mi capacidad, me acompañaron en mis subidas y bajadas, y escucharon cada avance de este trabajo. No hubiera podido haberlo hecho sin ellos.

Índice

Lista de figuras

Lista de cuadros

Resumen

En los últimos años, gran parte de los estudiantes de Ingeniería Electrónica de la Universidad del Valle han enfocado sus trabajos de graduación en el diseño de un circuito integrado utilizando software Synopsys para lograrlo. El proceso de descarga e instalación de estas aplicaciones toma un tiempo considerable y con frecuencia se cometen errores en el camino que afectan el desempeño de la aplicación y por consiguiente del diseño del chip.

El objetivo principal de este trabajo fue facilitar y automatizar este proceso. Así como implementar tecnología de contenedores para instalar y ejecutar las aplicaciones en un ambiente aislado y con todo lo necesario para su correcto funcionamiento.

La descarga se realizó por medio de comandos utilizando el protocolo SFTP. Por otra parte, la instalación se automatizó con Bash Script. Se hizo una interfaz donde el usuario puede escoger el producto y la versión a instalar de tal manera que todo el proceso de descarga, instalación y actualización de variables de entorno se hace automáticamente.

Se lograron descargar e instalar de manera automática un total de 18 aplicaciones de Synopsys de manera exitosa. Se verificó la descarga con un algoritmo que compara el checksum real con el valor teórico dado por el proveedor. Por otro lado, se verificó la Instalación siguiendo las recomendaciones de las guías de Instalación de Synopsys para cada aplicación.

Abstract

In recent years, most Electronic Engineering students at Universidad del Valle has dedicaded their graduation project on the design of an integrated circuit using Synopsys Software. The process of downloading and installing these applications takes considerable time and mistakes are often made along the way that affect the performance of the application and consequently the design of the chip.

The main objective of this work was to facilitate and automate this process. Container technology was also implemented for some of the applications. This was done through Singularity Software.

The downloading process was done by commands using the SFTP protocol. On the other hand, the installation of the apps was automated using Bash Script. An interface was made where the user can choose the product and version to install in such a way that the entire process of downloading, installing and updating environment variables is done automatically.

A total of 18 Synopsys applications were successfully downloaded and installed automatically. The download was verified by performing an algorithm that compares the actual checksum with the theoretical value given by the provider. On the other hand, the Installation was verified following the recommendations of the Synopsys Installation guides for each Application.

CAPÍTULO 1

Introducción

En el año 2020 inició la primera crisis por escasez de semiconductores. Empresas en todo el mundo aún tienen problemas fabricando sus productos pues les hace falta un componente de suma importancia: el microchip. Es indiscutible que no hay vuelta atrás para la necesidad de los microchips. Estos se encuentran en casi todos los dispositivos que utilizamos en el día a día. La escasez hizo darnos cuenta de la vital importancia de este pequeño componente.

Para suplir la demanda, las empresas de semiconductores están buscando maneras de aumentar su producción y acelerar el proceso de fabricación. Intel, una de las empresas más grandes en esta industria, ya ha expresado su interés en expandirse hacia Latinoamérica, donde ya cuenta con un centro de Diseño e Investigación en Costa Rica. La presencia de Intel en Costa Rica, se tradujo a un aumento en la inversión externa y en la oferta de trabajo, mejorando el desarrollo general del país [1].

El Ingeniero Carlos Esquit, actual director del departamento de Electrónica, Mecatrónica y Biomédica, trabajó en la compañía Intel al finalizar su carrera y pudo darse cuenta de los grandes beneficios de traer este tipo de empresas a Guatemala. Con este objetivo, comenzó a impartir la clase *Introducción al diseño de sistemas VLSI* y posteriormente las clases de *Nanoelectrónica I y II* en la Universidad del Valle.

Así nació el proyecto Gran Jaguar en el año 2019, con el objetivo de mandar a fabricar el primer chip diseñado completamente por guatemaltecos. En los últimos años, casi todos los estudiantes de ingeniería Electrónica han enfocado sus trabajos de graduación en finalizar este proyecto utilizando las aplicaciones de la empresa Synopsys, la cual es una empresa que se especializa en el desarrollo de software para el diseño de circuitos integrados.

Uno de los obstáculos con los que se encontraron fue realizar la descarga e instalación de este software tan especializado. Este proceso toma una cantidad considerable de tiempo y hay que realizarlo cada año para mantener las aplicaciones actualizadas. Este proyecto nació con el objetivo de ahorrar tiempo a los diseñadores del chip y evitar posibles errores en el proceso.

CAPÍTULO 2

Antecedentes

Actualmente, la Universidad del Valle es la única casa de estudios en Guatemala que imparte cursos de Nanoelectrónica. Esto inició en el año 2013, cuando se agregó el curso *Introducción al diseño de sistemas VLSI* al mapa curricular de Ingeniería Electrónica. Al año siguiente se obtuvo un acuerdo con la empresa Synopsys para que los estudiantes tuvieran acceso a su software. Ese mismo año apareció la primera tesis relacionada al tema: *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*, encontrada en [2].

En el Marco Teórico de este documento, De los Santos explica el proceso de instalación de algunas aplicaciones de Synopsys y del sistema operativo donde estas se ejecutarán (CentOS). Esta fue la primera vez que se realizó este proceso en las computadoras de la Universidad del Valle por lo que este documento marca la pauta para futuras generaciones. La descarga de las aplicaciones se realizó vía HTTPS puesto que el servidor FTP tenía problemas de conexión. Se descargó el *installer* de Synopsys versión 3.1. Los archivos de instalación se guardaron en la siguiente ubicación: `/usr/synopsys/`, adentro de un folder con el nombre de cada aplicación. Este es el listado de las aplicaciones descargadas en 2015:

1. Galaxy Custom Designer
2. IC Validator
3. HSPICE
4. StarRC
5. Custom Explorer
6. PyCell Studio

La mayoría de estas aplicaciones fueron instaladas utilizando la interfaz gráfica de *Synopsys Installer*. Únicamente PyCell requirió de un proceso un tanto diferente. Aparte de

las aplicaciones, se instalaron los iPDK de 32/28 nm, esta vez utilizando el servidor FTP. Luego de la instalación se editó el archivo `$HOME/.bashrc` para agregar ciertas variables de entorno y la licencia del software. Estas fueron algunas de las líneas agregadas al archivo en ese entonces:

```
$ export PATH=$PATH:/usr/synopsys/installer
LM_LICENSE_FILE=27020@scluvg; export LM_LICENSE_FILE
# Variable para iPDK de Synopsys
export SAED32_28_PDK=/usr/synopsys/iPDK/SAED32_28_iPDK/
# Variable para IC Validator
export ICV_HOME_DIR=/usr/synopsys/IC_Validator
# Directorio de IC Validator
PATH=/usr/synopsys/IC_Validator/bin/AMD.64:$PATH
#Generación de variables para PyCell Studio
source /usr/synopsys/PyCell/quickstart/bashrc
#Directorio del servidor de licencias
PATH=/usr/synopsys/11.8/linux/bin:$PATH
#Directorio de Hspice
PATH=/usr/synopsys/HSPICE/I-2013.12-SP2-1/hspice/bin/:$PATH
#Directorio de StarRC
PATH=/usr/synopsys/StarRC/bin/:$PATH
# Directorio de CustomDesigner
PATH=/usr/synopsys/CustomDesigner/bin/:$PATH
#Directorio de CustomExplorer
PATH=/usr/synopsys/CustomExplorer/bin/:$PATH
```

Ya instaladas las aplicaciones, estas fueron utilizadas para diseñar un sumador de 32 bits por medio de un circuito *Ripple Carry* y los resultados de las pruebas realizadas fueron satisfactorias.

En el año 2015, se sustituyó el curso de *Introducción al diseño de sistemas VLSI* por *Nanoelectrónica 1 y 2*, donde se enseña a los estudiantes a utilizar el software de Synopsys, instalado anteriormente y la teoría detrás de estos procesos. No fue hasta el año 2019 que Steven Rubio [3] y Luis Najera [4] realizaron nuevos trabajos de graduación relacionados con el tema. Esta vez con el objetivo de crear el primer circuito integrado diseñado en Guatemala. En ese momento se definió la función del chip, la cual se mantiene hasta la actualidad: imprimir un mensaje con el nombre de sus diseñadores.

Para poder realizar esto, era crucial obtener las librerías de diseño de TSMC, pues las que se tenían no tenían permisos de fabricación. La empresa dio acceso a librerías con tecnología de 180nm. Debido a que ya habían pasado varios años después de la primera instalación de las aplicaciones, el proceso se volvió a realizar con las versiones más actualizadas. También se agregaron otras aplicaciones útiles para el proceso que no se habían agregado en el pasado. Este es el listado de las aplicaciones descargadas en 2019:

1. Design Vision
2. IC Compiler

3. Custom Compiler
4. IC Validator
5. StarRC
6. Formality
7. VCS
8. Verdi

Con estas aplicaciones se realizó la primera iteración del proceso de diseño y se llevaron a cabo varias verificaciones; todas exitosas. Al siguiente año, se obtuvo el financiamiento para la fabricación del chip gracias a el programa HORIZON 2020, a través de IMEC.

Desde el año 2019 en adelante, la mayoría de estudiantes de Ingeniería Electrónica de la Universidad del Valle ha dedicado sus trabajos de graduación a finalizar el diseño del circuito. Gracias a que las generaciones anteriores realizaron el flujo de diseño de forma general, ahora cada estudiante podía enfocar su estudio en una sola fase y así conocer a mayor profundidad cada herramienta, asegurando el funcionamiento del circuito integrado. En los últimos años se ha aumentado la cantidad de verificaciones, así como las aplicaciones a utilizar. Este es el listado de las aplicaciones descargadas en 2021:

1. HSpice (PrimeSim)
2. Verdi
3. VCS
4. IC Compiler I y II
5. IC Validator
6. Cosmos Scope
7. Custom Waveview
8. Design Vision
9. DVE
10. Formality
11. StarRC

Actualmente se cuenta con un código en python para generar un archivo verilog cambiando únicamente el texto que se desea desplegar. Este archivo se sintetiza lógicamente, se realizan las verificaciones necesarias y se procede a realizar la síntesis física. Todas las verificaciones realizadas han salido exitosas a excepción de DRC en donde se obtienen seis errores de densidad que aún están por arreglarse. En la Figura 1 se muestra el Layout Final del circuito.

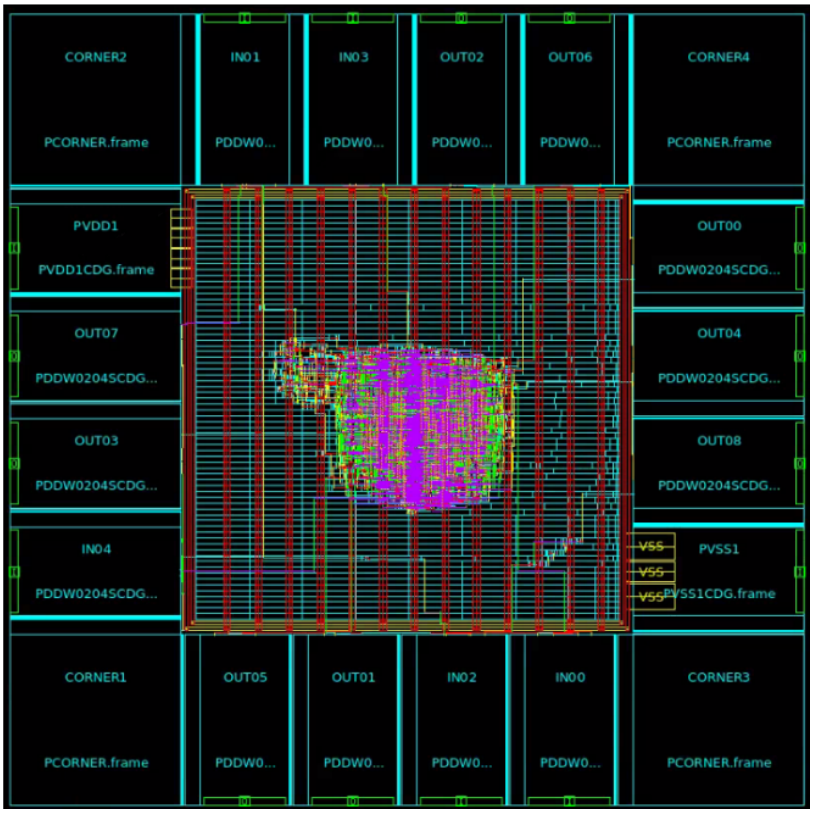


Figura 1: Layout obtenido en 2021

En el año 2021 también se comenzó el primer trabajo de graduación, realizado por Joel González, relacionado a la automatización del proceso de diseño utilizando Bash. Este trabajo se ha enfocado en la automatización y unión de cada fase de diseño, sin embargo aún se encuentra en proceso. La automatización de la descarga e instalación de las aplicaciones es algo que aún no se ha realizado anteriormente.

CAPÍTULO 3

Justificación

Desde el siglo pasado, la electrónica ha sido de vital importancia para el avance de la industria y la globalización. Hoy en día es imposible pensar en un mundo sin chips. Se encuentran en los dispositivos electrónicos que utilizamos diariamente, en medios de transporte, equipo médico, etc. El transistor es la base de esta revolución. Desde su aparición, se han encontrado métodos para hacerlo cada vez más pequeño. Haciendo que se reduzca el costo y potencia de un chip, mientras aumenta su eficiencia. Es evidente que la habilidad de diseñar y fabricar chips es de suma importancia para la humanidad y lo será por mucho tiempo.

Actualmente, la industria de fabricación de chips se encuentra mayoritariamente en países del primer mundo: Estados Unidos, China, Alemania, entre otros. Consecuentemente, el aprendizaje de nanoelectrónica en casas de estudio de nivel superior se ha centralizado en estos países. Muy pocos países Latinoamericanos tienen acceso a este conocimiento y mucho menos los recursos para llevar a cabo prácticas o proyectos. Adquirir el conocimiento de diseño de chips y divulgarlo en Guatemala, abriría nuevas oportunidades laborales y la posibilidad de traer la industria del semiconductor aquí.

El primer trabajo de graduación relacionado al tema realizado en la Universidad del Valle, enfocó gran parte de su tesis en la descarga e instalación de las aplicaciones de Synopsys. Este es un procedimiento de suma importancia pues de realizarse de manera incorrecta afectaría por completo los resultados. Además, son procedimientos tardados y que consumen grandes cantidades de recursos en nuestras máquinas. Debido a que Synopsys lanza nuevas versiones del software con frecuencia, cada año el equipo de trabajo del chip debió de repetir el proceso de descarga e instalación. Gracias a los esfuerzos de cada generación, el procedimiento se fue volviendo más sencillo. Sin embargo, siempre ha tomado una cantidad considerable de tiempo, que bien pudo haberse utilizado para mejorar la arquitectura o comprender a mayor profundidad las herramientas utilizadas.

A lo largo de todos estos años, la descarga e instalación de las aplicaciones la ha realizado cada estudiante en la computadora que se le asignó. Esto ha tenido como consecuencia resultados diferentes a la hora de correr las aplicaciones, pues las librerías y dependencias

se encuentran en diferentes ubicaciones y las probabilidades de cometer errores son mayores. Utilizar contenedores nos ayudará a instalar y correr las aplicaciones de una manera uniforme, ordenada y modular. Además, la utilización de contenedores permite utilizar los recursos de una manera eficiente, liberando espacio cuando estos no son necesitados.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Crear una interfaz por la cual se puedan instalar, de forma automática, las aplicaciones de Synopsys con versiones más recientes que sean necesarias para las fases de diseño de un circuito integrado. Se busca que esta interfaz sea intuitiva y permita al usuario escoger las aplicaciones a instalar, así como nombres y ubicaciones de archivos. Además, se busca que las aplicaciones se ejecuten en contenedores utilizando el Software de Singularity.

4.2. Objetivos específicos

- Realizar la descarga e instalación de la última versión de las aplicaciones necesarias para llevar a cabo el diseño, de forma automática.
- Generar variables de entorno para cada aplicación y actualizarlas en el .bashrc del usuario de forma automática.
- Actualizar la variable de entorno PATH del usuario local.
- Definir el orden y la ubicación de los archivos a descargar.
- Agregar una lógica de depuración en el caso de que algo falle en el proceso.
- Unir todas las fases de automatización en un solo script.
- Comprobar que el resultado obtenido por el proceso automático sea equivalente a las iteraciones pasadas.
- Hacer pruebas para comprobar la utilidad de los contenedores.

CAPÍTULO 5

Alcance

El alcance de este proyecto es poder definir un algoritmo y generar una plataforma de descarga e instalación de ciertas aplicaciones de Synopsys necesarias para la realización del proyecto **Gran Jaguar**, además de los archivos PDKs y EDKs. En total, serán 18 aplicaciones, incluyendo *Synopsys Installer* y *Synopsys Container*. Para esto se proponen dos métodos: uno por línea de comandos y la otra por interfaz gráfica. Estas se realizarán utilizando *Bash Script* conjunto a la herramienta *Whiptail*.

Estas plataformas serán capaces de descargar desde el servidor de Synopsys los productos, verificar su checksum, instalar las aplicaciones utilizando *Synopsys Installer* y cambiar las variables de entorno para que las aplicaciones se puedan ejecutar desde cualquier parte. Los programas deberán tomar en cuenta las diferencias entre cada aplicación, pues el proceso de instalación no es el mismo para todas. También deben avisar al usuario si algo salió mal en alguna de las etapas o si el valor ingresado por el usuario es incorrecto.

Además de esto, se realizará la configuración del Contenedor de Synopsys en las computadores del laboratorio de Nanoelectrónica y este será utilizado para algunas de las aplicaciones.

CAPÍTULO 6

Marco teórico

6.1. Protocolos de Transferencia de Archivos

6.1.1. File Transfer Protocol (FTP)

Es un protocolo de comunicación utilizado para la transferencia de archivos entre un cliente y un servidor, mediante conexiones TCP/IP. Esto lo hace un protocolo de la capa de Aplicación.

Tanto HTTP como FTP trabajan sobre el protocolo TCP y ambos son utilizados para la transferencia de archivos, sin embargo tienen algunas diferencias fundamentales. A diferencia de HTTP, FTP usa dos conexiones paralelas para la transferencia de archivos: una para control y otra para datos. Por medio de la conexión de control se mandan todos los comandos necesarios y la conexión de datos se utiliza solamente para realizar la transferencia de archivos, como se observa en la figura 2 [5].

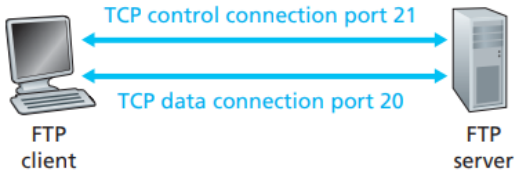


Figura 2: Puertos de control y datos FTP

FTPS o FTP/SSL

IMEC es un medio inseguro pues la información no se transporta cifrada. Por esa razón surgió FTPS como una extensión de FTP. Este corre sobre SSL (Secure Sockets Layer) o

TLS (Transport Layer Security) para cifrar los canales de control y datos. Utiliza el puerto 989 para control y el 990 para datos. Debido a que es una extensión de FTP, FTPS utiliza los mismos comandos que FTP.

Una de las desventajas de FTPS es que puede tener problemas funcionando con firewalls de alta seguridad. Esto es debido a que FTPS abre una nueva sesión en un puerto diferente cada vez que necesita hacer una transferencia de datos, lo cual puede ser riesgoso para el sistema si el firewall no se configura correctamente [6].

6.1.2. SSH File Transfer Protocol (SFTP)

SFTP tiene las mismas funcionalidades que FTP y FTPS pero corre sobre el protocolo SSH como un subsistema. Es decir, SFTP utiliza como base el protocolo SSH y le agrega funciones de transferencia de archivos. Esto lo hace mucho más seguro por las propiedades de autenticación de SSH. Por esta razón ha reemplazado por completo a FTP y FTPS. Además, la configuración para este protocolo es mucho más sencilla debido a que solamente utiliza un puerto.

SFTP protege contra ataques como *password sniffing* y *man-in-the-middle*. Protege los datos mediante cifrado y utiliza métodos de autenticación tanto en el servidor como en el usuario.

El servidor puede procesar solicitudes de parte del cliente de forma asíncrona y responder desordenadamente. Para mejorar el rendimiento, los clientes normalmente envían varias solicitudes antes de esperar las respuestas [7].

Parámetros	**FTPS**	**SFTP**
Puertos	990 para control y 989 para datos	22
Encriptación	SSL/TLS	SSH
Configuración Firewalls	Compleja	Sencilla
Lista de Comandos	Limitado	Amplia
Compatibilidad	No todos los servidores FTP soportan SSL.	Casi todo servidor lo soporta.

Cuadro 1: Comparación entre SFTP y FTPS

6.2. Bash Shell

Normalmente, se utiliza la interfaz gráfica del Sistema Operativo para interactuar de manera efectiva con los archivos y llevar a cabo una amplia variedad de acciones. Esta interfaz gráfica proporciona un entorno visual intuitivo que facilita la navegación y la manipulación de los archivos. Sin embargo, existe otra alternativa que permite realizar estas mismas interacciones mediante la línea de comandos.

Para llevar a cabo esta tarea, se emplea lo que se conoce como *Shell*. Esta se encarga de enviar los comandos que hemos ingresado a través de la terminal directamente a nuestro sistema operativo, para que sean ejecutados.

Este método nos brinda la capacidad de realizar acciones de manera significativamente más rápida y eficiente en comparación con el uso de la interfaz gráfica.

Es importante señalar que muchos servidores, por su naturaleza y configuración, no poseen una interfaz gráfica. Por lo tanto, la utilización de una *Shell* se vuelve no solo útil, sino absolutamente indispensable en estos casos.

En el entorno de Linux, existe una variedad de intérpretes de comandos que funcionan sobre la *Shell*. Entre todos ellos, los más conocidos y utilizados son, sin lugar a dudas, el *Bourne Shell* y su popular extensión, que es el *Bourne-Again Shell*, comúnmente abreviado como *bash* [8].

6.2.1. Bash Script

Es posible realizar scripts de Bash para ejecutar comandos de Linux de forma automática. Estos scripts son especialmente útiles para simplificar tareas repetitivas o complejas. Bash ejecuta los comandos línea por línea. Normalmente el script se guarda con la extensión `.sh`, que es una convención comúnmente aceptada, pero estrictamente necesaria.

Para que bash identifique de manera correcta al interprete de comandos, es necesario incluir en la primera línea de texto lo que se conoce como el *shebang*. Este consiste en una combinación de un *(#)* y un *(!)*, seguido del path de *Bash* o *Bourne Shell*. Un ejemplo típico de este sería: `#!/bin/bash`.

Si se desea usar *Bourne Shell* en lugar de *Bash*, el *shebang* quedaría de la siguiente manera: `#!/bin/sh`. Para conocer el path del interprete se puede utilizar el comando `which bash` o `which sh`, según corresponda.

Para ejecutar el script se puede hacer de dos maneras:
`./nombre_script.sh` o `bash nombre_script.sh`

Para ejecutarlo de la primera forma, se le deben de conceder permisos de ejecución con el comando `chmod +x <script>` [8].

6.2.2. Comandos de Linux

Se puede realizar una gran cantidad de operaciones en la *shell* de Linux. Desde crear, copiar y mover archivos hasta automatizar procesos complejos repetitivos. En la siguiente tabla se muestran algunos comandos de gran utilidad [9]:

Comando	Descripción
`mkdir`	Crear un directorio.
`lftp`	Es utilizado para acceder a protocolos de transferencia de archivos. Soporta FTP FTPS, SFTP, HTTP, HTTPS etc. Cada operación en LFTP es confiable, pues de haber un error en el proceso, LFTP automáticamente lo reinicia hasta cumplir lo requerido.
`sed`	Puede utilizarse para buscar cadenas de texto y realizar cambios en ellas.
`awk`	Permite realizar una amplia variedad de acciones encontrando patrones en la data.
`pwd`	Imprime la ruta del directorio actual, iniciando desde el root.
`cat`	Concatena archivos y los imprime de manera secuencial en la terminal.
`locate`	Encuentra la ubicación de archivos dandole como entrada el nombre o parte de él.
`find`	Encuentra la ubicación de archivos que cumplan con ciertos criterios (nombre, tamaño, tipo de archivo, etc).
`df`	Despliega el espacio utilizado y disponible en los sistemas de archivos escogidos.
`du`	Reporta el espacio en disco para algún directorio especifico.
`diff`	Compara dos archivos y despliega las diferencias entre ellos.
`chmod`	Cambia los permisos de acceso de un archivo.
`jobs`	Despliega los procesos que están corriendo actualmente y su porcentaje de uso de CPU.
`bind`	Asocia teclas con funciones o variables, de tal manera que sean de fácil ejecución.
`cksum`	Obtiene la verificación de redundancia cíclica (CRC) de algún archivo en específico.
`grep`	Busca e imprime en algún archivo un patrón de texto específico. También se puede usar `egrep` para expresiones extendidas.
`env`	Imprime, crea o elimina alguna variable de entorno específica.
`gawk`	Encuentra y reemplaza cadenas de texto.

Cuadro 2: Algunos comandos de Linux

6.2.3. Técnicas de scripting

Expresiones regulares (RegEx)

Las expresiones regulares, RegEx o RegExp por sus siglas en inglés, son un conjunto de caracteres usados principalmente para encontrar patrones en textos y manipular cadenas. La mayoría de veces son utilizadas junto a los comandos `grep`, `sed`, `awk`, entre otros.

Expresión	Descripción
`.`	Encuentra todos los matches del texto, haciendo el punto un comodín.
`^`	Encuentra un match al inicio de la cadena.
`$`	Encuentra un match al final de la cadena.
`*`	Encuentra todas las cadenas que contengan el texto que le preceda. En este caso, el asterisco puede referirse a varios caracteres.
`\`	Deshabilita la función RegExp del carácter que le sigue.
`()`	Busca el conjunto de expresiones que se encuentre dentro de los paréntesis.
`?`	Encuentra matches con cualquiera de los caracteres que le precedan. Esto quiere decir, que no necesariamente se deben encontrar juntos.
`\<...\>`	Marca límites en las cadenas.

Cuadro 3: Expresiones frecuentes

Una expresión puede ser una combinación de varias; por lo que existen infinidad de opciones.

Globbing

Es importante recalcar que RegEx solamente funciona con cadenas de texto. Si se desea encontrar patrones en el nombre de cualquier archivo o directorio se utiliza Globbing, también conocido como *Path Name Expansion*. Esta técnica utiliza patrones *wildcard* para lograrlo. Un patrón *wildcard* es cualquier cadena que contenga los símbolos '?', '*' o '['.

Wildcard	Descripción
`'?'`	Hace referencia a cualquier carácter. Por ejemplo, `ls <nombre>?` imprimiría todos los archivos cuyo nombre contenga <nombre> seguido de un carácter cualquiera.
`'*'`	Hace referencia a cualquier cadena, incluyendo las que están vacías. Por ejemplo, `ls <nombre>*` imprimiría todos los archivos cuyo nombre contenga <nombre>.
`'[]'`	Busca el primer archivo que contenga lo indicado adentro de los corchetes. Si el primer carácter es '!', esto quiere decir que se está excluyendo el carácter que le siga.

Cuadro 4: Wildcards para Globbing

Se pueden realizar combinaciones de todas las *wildcards* para obtener resultados más completos. Para identificar que se está haciendo Globbing en lugar expresiones regulares, es recomendable encerrar las *wildcards* en comillas; aunque no es estrictamente obligatorio.

Redireccionamiento

Al correr un script, existen tres archivos por default: los mensajes de entrada que vienen de nuestro teclado (`stdin`), la salida del script en la pantalla (`stdout`) y los mensajes de

error (`stderr`). Estos archivos pueden ser redireccionados entre sí. Es decir, se puede tomar la salida de un programa e ingresarlo como una entrada en otro.

Cada uno de estos archivos tiene asignado un número, siendo el 0 para `stdin`, 1 para `stdout` y 2 para `stderr`. Si se quisiera utilizar otro tipo de archivos, se puede escoger un número del 3 al 9 [10].

Operador	**Función**
<	Realiza una redireccionamiento de entrada solamente de lectura.
<<	Es utilizado para los bloques de código *Here*.
>	Es típicamente usado para redireccionar la salida de un comando o archivo hacia otro sobreescribiéndolo.
>>	Hace lo mismo que el operados >, solamente que en lugar de sobreescribirlo lo adjunta a lo ya obtenido.
|	También conocido como 'pipe'. Es utilizado para mandar la salida del primer comando como la entrada del segundo.
&	Se usa para redireccionar tanto el `stdout` como el `stdin`
< >	Abre un archivo para lectura y escritura.

Cuadro 5: Operadores de redireccionamiento

Process Substitution

El redireccionamiento nos permite mandar la salida de un comando o programa y mandarlo a otro. Si se quisiera mandar la salida de múltiples comandos se necesita usar *Process Substitution*. Su estructura es la siguiente:

```
comando ..<(lista de comandos)     o
comando..>(lista de comandos)
```

Los comandos en la lista son separados únicamente con un espacio [11].

Documentos Here

Es un bloque de código utilizado para mandar una lista de comandos a un comando interactivo como ftp, lftp o cat, aunque también puede ser utilizado para no interactivos. Esto se realiza por medio de un tipo de redireccionamiento. Su estructura es la siguiente:

```
programa << limite
       comando 1
       comando 2
       ....
limite
```

Donde `programa` llama al programa interactivo que se estará utilizando y `limite` indica el inicio y final de la lista de comandos a mandarle.

También existen las Cadenas Here. Su estructura es la siguiente:

```
COMANDO << $STRING
```

En este caso, lo que se encuentre en la variable `STRING` será la entrada (`stdin`) de `COMANDO`.

6.3. Contenedores

Un Sistema Operativo está formado por dos componentes principales: el espacio de Kernel y el espacio de usuario. En el espacio de usuario es donde se ejecutan todas las aplicaciones, librerías y servicios. Debido a que la configuración del espacio de usuario es diferente para cada Sistema Operativo, no todas las aplicaciones pueden ejecutarse en ellos. Por ejemplo, si se desea correr una aplicación empaquetada para Ubuntu en CentOS no se podrá hacerlo porque habrán problemas de compatibilidad.

Los contenedores son la solución a este problema. Su función principal es empaquetar y aislar todo lo necesario para ejecutar una aplicación (dependencias, runsets, etc.), de tal manera que sea ejecutable sin importar el ambiente en el que se encuentre. Es decir, se puede utilizar el mismo contenedor para correr cierta aplicación en cualquier versión de Windows, CentOS, Ubuntu o Fedora, sin ningún problema. Varios contenedores pueden estar corriendo en la misma máquina compartiendo espacio en el Kernel, pero corriendo procesos aislados en el espacio de usuario [12].

Las máquinas virtuales tienen un funcionamiento similar al de un contenedor. Ambos buscan aislar y empaquetar procesos, pero cada uno lo hace de forma distinta. Los contenedores se miden en Megabytes, pues únicamente necesitan empaquetar una sola aplicación, mientras que las máquinas virtuales se miden en Gigabytes, pues estas contienen su propio sistema operativo que les permite realizar diversas funciones al mismo tiempo, con una gran cantidad de recursos. Estas diferencias hacen que los contenedores sean pequeños, rápidos, modulares y eficientes, utilizando solamente los recursos que necesita.

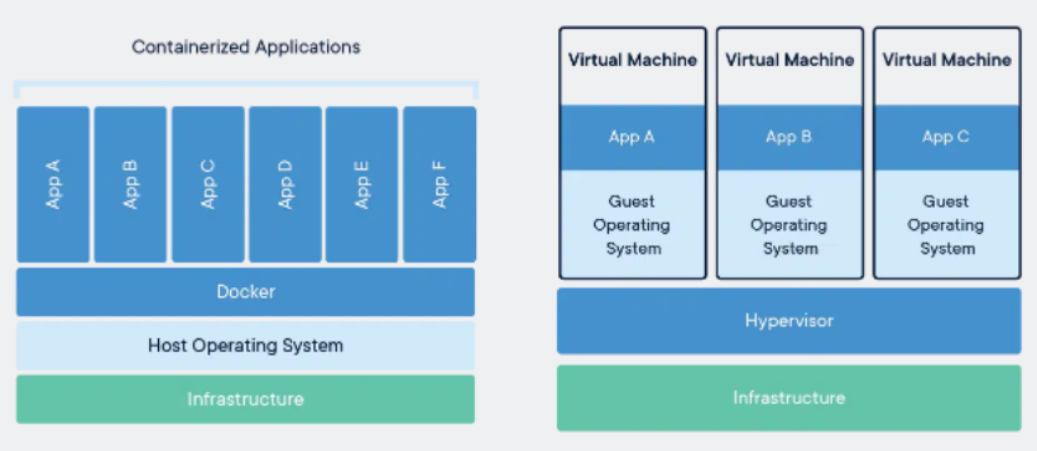


Figura 3: Comparación entre un contenedor y una VM

Actualmente el software más utilizado para crear y manejar contenedores es Docker. Frecuentemente es utilizado junto con la plataforma Kubernetes, que ayuda a orquestar la ejecución de los contenedores. Entre otras aplicaciones útiles para el manejo de contenedores están: Red Hat OpenShift, AWS Fargate y Singularity [13].

6.3.1. Singularity

Es una plataforma open-source en el que se puede crear y correr contenedores. Singularity se creó con el fin de ejecutar aplicaciones complejas de forma sencilla, portátil y reproducible. Ofrece soluciones para grupos científicos con una carga de datos u aplicaciones extensa.

Se tiende a ver los contenedores como una forma diferente y más eficiente de máquina virtual, pues son mayormente utilizados para contener aplicaciones y ejecutar aplicaciones en medios aislados como es el caso para las máquinas virtuales. Sin embargo, los beneficios que nos ofrecen los contenedores pueden ser aprovechados de diferentes maneras. Muchos científicos podrían beneficiarse enormemente del uso de la tecnología de contenedores, pero necesitan un conjunto de funciones que difiera un poco del que está disponible con la tecnología de contenedores actual. Singularity se enfoca en suplir esta necesidad [14].

6.4. Software de Synopsys

6.4.1. Descarga de productos

Synopsys tiene cuatros servidores EFT: uno en los Estados Unidos y tres en Europa. Al acceder al servidor eft.synopsys.com, se está accediendo al servidor que se encuentra más cercano al usuario. Si llegara a dar problemas, también se puede acceder específicamente a cada servidor [15]:

Ubicación	**Servidor**
Estados Unidos	marley.synopsys.com (198.182.44.204)
Europa	tosh.synopsys.com (149.117.30.1) fireball.synopsys.com (198.182.37.221) hu01estftp.synopsys.com (84.2.38.58)

Cuadro 6: Servidores de Synopsys

Synopsys ofrece cinco opciones para la descarga de sus productos:

EFTStream

Es uno de los métodos de descarga más rápidos. Se basa en el protocolo FASP (Aspera Fast Adaptive and Secure Protocol). Para utilizarlo es necesario instalar la extensión IBM Aspera Connect en el navegador.

HTTPS

Es el método más intuitivo. Para descargar archivos solamente se debe ir al Download Center de Solvnet, escoger el producto y la versión y hacer click en descargar. Este es el método que se ha utilizado en los años anteriores para instalar las aplicaciones en las computadoras del laboratorio de Nanoelectrónica.

FileZilla

Es una interfaz de terceros para descargar archivos utilizando el protocolo FTP. Para utilizarla hay que descargar la aplicación FileZilla Client e ingresar la información correspondiente para poder acceder al servidor de synopsys.

SFTP

Anteriormente Synopsys soportaba el uso del protocolo FTP, pero este fue deshabilitado por no ser seguro. Este fue reemplazado por el uso del protocolo SFTP. Para acceder al servidor EFT por medio de SFTP se utiliza la siguiente linea de comandos, junto con la contraseña correspondiente:

sftp <SolvNetUsername>@eft.synopsys.com

FTPS

Se puede acceder al servidor FTP de synopsys por medio del protoclo FTPS con la siguiente línea de comandos, junto con la contraseña correspondiente:

lftp -u <SolvNetUsername>@eft.synopsys.com

6.4.2. Synopsys Installer

Se utiliza para instalar las aplicaciones de Synopsys con formato de archivo *.spf*. No es posible instalar productos basados en Windows o con otro tipo de archivos con el Installer de Synopsys.

Los archivos que el instalador soporta tienen la siguiente convención de nombres:

```
productname_vproductversion_common.spf
productname_vproductversion_platform.spf
productname_vproductversion_platform.spf.partnn
```

Cada aplicación descargada tendrá un archivo *common* y al menos uno *platform*.

Instalación

Primero se debe verificar que el archivo .run del Instalador sea ejecutable, de no serlo ejecutar el comando:

```
chmod 755 SynopsysInstaller_vlatest_version.run
```

Luego instalar el programa utilizando el siguiente comando e ingresar la dirección donde se desea realizar la instalación cuando sea pedida.

```
./SynopsysInstaller_vlatest_version.run
```

Uso en la instalación de aplicaciones

Al día de hoy, la última versión de este instalador es la 5.5. Es de suma importancia tener la versión más reciente si se desea instalar la última versión de las aplicaciones. La instalación del software se puede realizar ya sea con la interfaz gráfica, por línea de comandos o por lotes.

Interfaz gráfica

Se invoca la interfaz gráfica por medio del comando `installer -gui`.

Al presionar 'Start' nos pedirá el siteID, el usuario y correo del administrador de la licencia. Luego nos pedirá colocar la dirección donde se encuentran los archivos descargados *.spf* que se deseen instalar. Si no encontrara alguno de estos archivos el programa tirará un error.

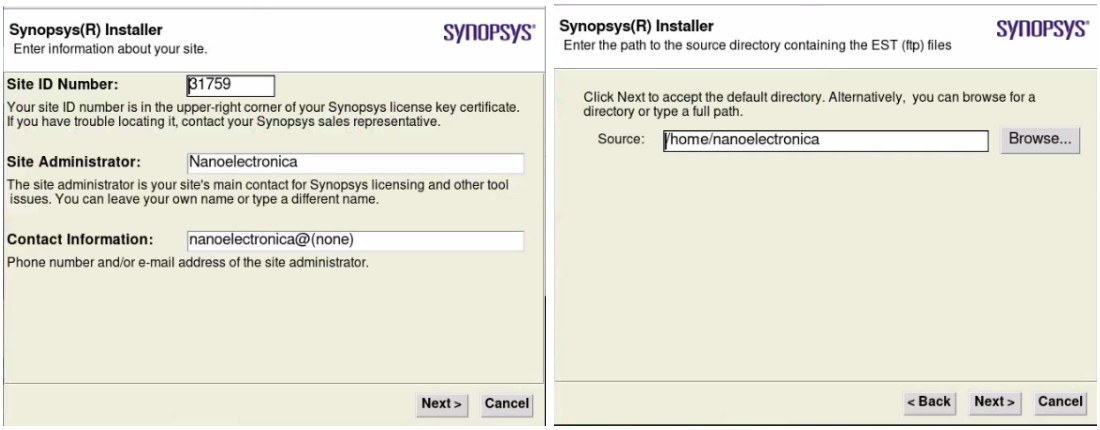


Figura 4: Segunda y tercera imagen del Instalador

Inmediatamente después de esto nos pedirá colocar la dirección (top-level) donde queremos que se realice la instalación. Luego nos mostrará el listado de productos que encontró para que el usuario seleccione cuáles desea instalar.

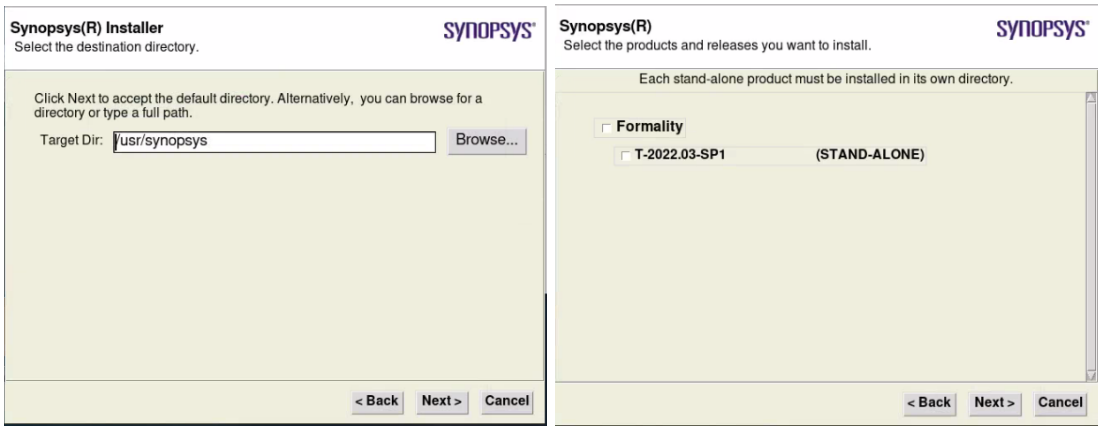


Figura 5: Cuarta y quinta imagen del Instalador

En la siguiente pantalla nos permitirá, si se desea, cambiar el fólder de instalación. Finalmente, mostrará información sobre el producto a instalar. Luego de verificarla se puede proceder a instalar.

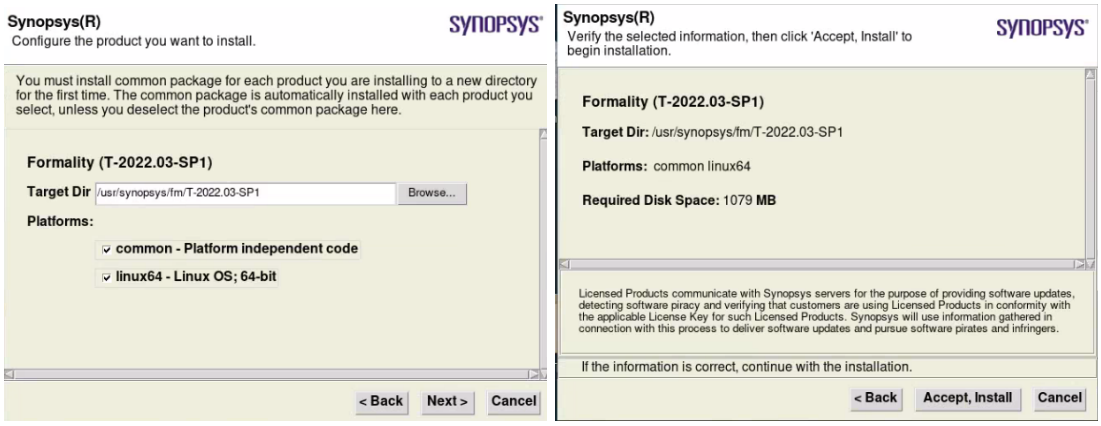


Figura 6: Sexta y séptima imagen del Instalador

Línea de comandos

Para iniciar el instalador en modo linea de comandos únicamente se debe ingresar `installer`. Al realizar esto, el instalador pedirá la siguiente información al usuario.

1. Dirección donde se encuentran los archivos descargados *.spf* (source).

2. Dirección donde se desea realizar la instalación (target).

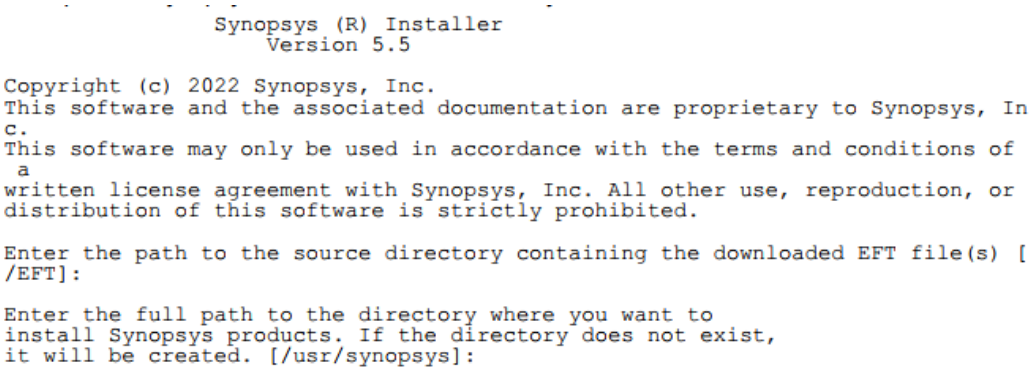


Figura 7: Texto desplegado al utilizar el Instalador en línea de comandos

Luego de ingresar esta información, se desplegará información del producto y esperará por una verificación por parte del usuario. Al recibirla iniciará inmediatamente la instalación de los productos.

Este mismo procedimiento también puede llevarse a cabo con un solo comando de la siguiente manera:

```
installer -source <dirección> -target <dirección>
```

El Instalador instalará todos los productos que encuentre en el source. Si se desea tener la opción de seleccionar solamente algunos, se debe utilizar la interfaz gráfica o usarlo en modo legacy con el comando `installer -legacy`.

Por Lotes
En este modo el instalador acepta un archivo de configuración donde se especifica el source, target, productos y versiones para poder descargar un conjunto de productos. Para utilizar este archivo de configuración se utiliza el comando
`batch_installer -config <archivo>`.

6.4.3. Synopsys Container

Se basa en la tecnología de contenedores de Singularity. Es una herramienta completamente opcional pero que se vuelve especialmente útil en instalaciones en la nube o en conjuntos de servidores con entornos heterogéneos.

No todas las aplicaciones de Synopsys están habilitadas para ser ejecutadas en contenedores. La lista de aplicaciones que sí lo están se encuentran en el Cuadro 8, donde también se especifica desde qué versiones están habilitadas.

Instalación

Actualmente la versión más reciente de Synopsys Container es la versión 1.7. Los archivos necesarios para realizar la instalación son los siguientes:

Comando	Descripción
`-gui`	Iniciar la interfaz gráfica del Installer.
`-install_as_root`	Realizar alguna instalación desde el root. Esto es útil para tatatata
`-legacy`	Correr el instalador en modo legacy. El instalador correrá de tal manera que soporte versiones viejas de los productos.
`-batch_installer`	Es utilizado cuando se desea instalar varias aplicaciones que se encuentran en un mismo directorio.
`-source <product dir>`	Se especifica el directorio donde se encuentran los archivos descargados necesarios para la instalación.
`-config <config file>`	Se especifica la configuración de la instalación por batches.
`-target <destination dir>`	Directorio en el que se instalará la aplicación.
`-product <products>`	Listado de productos a instalar. Si no se especifica, se descargan todos los productos encontrados.
`-release <version>`	Listado de versiones del producto a instalar. Si no se especifica, se descargan todas las versiones encontradas.
`-platform <platform>`	Listado de plataformas a instalar. Si no se especifica, se descargan todas las plataformas.
`-site_info <site info file>`	Archivo donde se encuentra la información del sitio de descarga.
`-container`	Habilitar configuración de contenedores.
`-container_config <config file>`	Archivo de configuración del contenedor.

Cuadro 7: Comandos del Instalador de Synopsys

- snps_container_vversion_common.spf.part00
- snps_container_vversion_common.spf.part01
- singularity-3.6.1.tar.gz

La instalación se realiza utilizando Synopsys Installer, realizando el procedimiento descrito en la sección anterior. Además, se necesita instalar el software *Singularity*, el cual viene incluido en estos archivos.

Instalación de Singularity

Synopsys soporta e incluye la versión 3.6.1 de Singularity en sus archivos de descarga del Contenedor. Para realizar la instalación, es necesario ejecutar los siguientes comandos con privilegios de *root*.

```
mkdir -p /usr/synopsys/downloads/singularity
```

Producto	Primera versión
CustomSim	P-2019.06
Custom Compiler	P-2019.06-SP1
Design Compiler (Synthesis)	P-2019.03-SP4
FineSim	P-2019.06
Formality	P-2019.03-SP4
Fusion Compiler	P-2019.03-SP2
HSPICE	P-2019.06
IC Compiler II	P-2019.03-SP2
IC Validator	P-2019.06
Library Compiler	P-2019.03-SP4
Lynx Design System	P-2019.03-SP2
NanoTime	P-2019.03-SP2-1
PrimePower	P-2019.03-SP1
PowerReplay	R-2020.12-SP1
PrimeTime	P-2019.03-SP1
S-Litho	S-2021.06
S-Litho PWA Ultra	S-2021.06
SiliconSmart	P-2019.06
StarRC	P-2019.03-SP1
TestMAX	P-2019.03-SP3
TestMAX CustomFault	P-2019.06

Cuadro 8: Productos y primeras versiones habilitadas para contenedores

```
mv singularity-3.6.1.tar.gz /usr/synopsys/downloads/singularity
cd /usr/synopsys/downloads/singularity
tar xvfz singularity-3.6.1.tar.gz
cd singularity-3.6.1
```

Si se desea realizar una instalación NFS correr el comando:

```
sudo ./singularity_install \ --prefix=/path/to/nfs/share/singularity-3.6.1
```

Si, en cambio, se desea instalar localmente:

```
sudo ./singularity_install \ --prefix=/opt/singularity-3.6.1
```

Configuración de Container

Para configurar el contenedor de Synopsys, primero se debe crear el archivo `.snps_container` de la siguiente manera:

```
cd installer_root_dir/container_setup
./container_setup.sh config \
        -source snps_container_installation_dir \
        -exe singularity_binary_path
```

Esto crea el archivo mencionado anteriormente, usa la imagen por default de cent Os (`centos7.simg`) y habilita el montaje automático del contenedor. Si se tiene la versión 6 de CentOs se debe cambiar esta imagen con el siguiente comando:

```
./container_setup.sh config -image centos6simg
```

Para verificar que el contenedor se haya configurado correctamente, se deben correr los siguientes comandos como usuario `su` en el directorio `root`.

1. Ejecutar el archivo `snps_container`:

```
/usr/synopsys/installer/5.3/container_setup/config/snps_container sh
```

Si todo está correctamente configurado, el prompt de Singularity debe aparecer: `Singularity>`.

2. Ahí mismo cambiar el directorio al home: `Singularity> cd $HOME`

3. Verificar que otro directorios home han sido montados encima del contenedor con el comando `ls ..`

4. Verificar que todos los puntos de montaje se encuentren disponible con el comando `df`. Por default, todos deberían de estar disponibles.

Desplegar el contenedor

Finalmente se debe desplegar la configuración del contenedor en las aplicaciones de Synopsys instaladas:

```
cd <installer_root_dir>/container_setup
./container_setup.sh deploy -target <synopsys_tools_path>
```

Existen dos opciones para iniciar aplicaciones accediendo a los contenedores:

1. Cambiar la variable de entorno `SNPS_CONTAINER = 1` para que se acceda automáticamente al contenedor.

2. Al llamar a la aplicación, invocar también al contenedor. De la siguiente manera: `<aplicacion> -container`.

CAPÍTULO 7

Servidor EFT

7.1. Conexión al servidor

El servidor de Synopsys al que se estará accediendo tiene la dirección `eft.synopsys.com`. Este nos direccionará a uno de los cuatro servidores de Synopsys: el que se encuentre más cercano. Para acceder al servidor se decidió utilizar el protocolo SFTP debido a la seguridad en la transferencia de los archivos y su facilidad de uso en la linea de comandos. Se realizó la conexión SFTP sobre el comando `lftp`, el cual es un programa para la transferencia de archivos que soporta FTP, HTTP, FTPS, entre otros. Se tomó esta decisión porque `lftp` nos permite realizar la conexión de forma interactiva. Es decir, el usuario, contraseña y servidor son especificados en una sola línea; lo cual facilita la ejecución del código.

En las primeras pruebas se realizaba la conexión con el siguiente comando:

```
lftp -f comandos_lftp.txt
```

Este comando ejecuta, línea por línea, los comandos guardados en el archivo de texto llamado *comandos_lftp.txt* . Sin embargo, esto era muy complicado pues para automatizarlo se tuvo que realizar un python que modificara el archivo .txt a conveniencia del usuario. De esta manera, se necesitaban tres archivos diferentes solamente para conectarse al servidor. Se solucionó esto con el uso de un HEREDOC adentro del bash script. Estos permiten mandar varios comandos a programas interactivos. Los comandos están delimitados por una variable, que en este caso fue nombrada EOF (End Of File). Adentro del HEREDOC se encontrarán los comandos para acceder a las carpetas correspondientes y descargar los archivos de interés. La estructura es la siguiente:

```
echo "Conectando al servidor de Synopsys ..."
```

```
lftp sftp://$USER:$PASSWORD@$SERVER << EOF
echo "Conexión exitosa ..."
...comando1...
...comando2...
...comando3...
EOF
```

7.2. Directorios

Los directorios en el servidor de Synopsys están organizados de la siguiente manera:

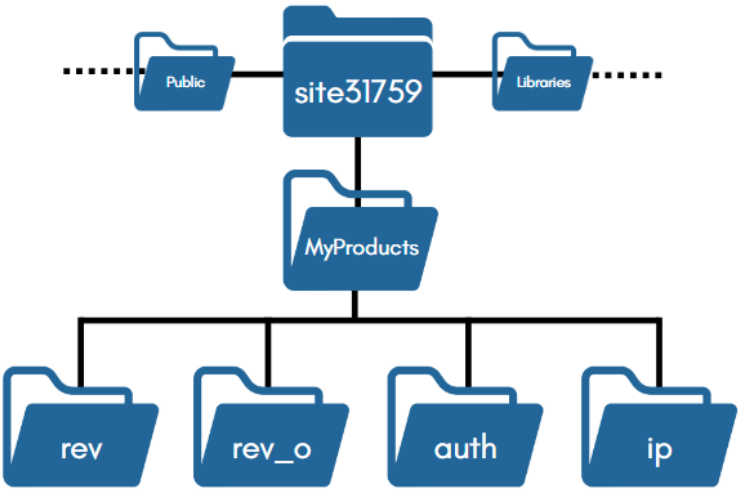


Figura 8: Jerarquía de directorios

El site31759 es el número que fue dado con la licencia obtenida. Los archivos de descarga de cada aplicación pueden encontrarse en tres carpetas: rev, rev_o y auth. En la siguiente tabla se muestran las especificaciones de cada carpeta:

Directorio	**Contenido**
/rev	Revenue. Contiene el último lanzamiento de los productos.
/rev_o	Archived Revenue. Contiene las versiones viejas de todos los productos.
/auth	Patch Releases. Contiene productos a los que se le hayan pequeñas modificaciones.

Cuadro 9: Descripción de directorios

En la carpeta *Libraries* se encuentran todas las librerías proveídas por Synopsys. En nuestro caso, se encuentran las librerías de 90, 32 y 14nm; así como los runsets de ICV.

7.3. Aplicaciones

En el Cuadro 10 se muestra el listado de aplicaciones que serán incluidas en la plataforma. Los archivos de descarga de cada una de ellas se encuentran guardados en carpetas con el siguiente nombre: `<nombre>_v<versión>`. El nombre de la carpeta (segunda columna del Cuadro 10) donde se encuentran guardados los archivos de cada aplicación en el servidor también fue guardada en una lista en el script, pues estos son utilizados en la búsqueda automática de los productos.

Nombre	Carpeta	Comando
Synopsys Installer	installer	`installer`
VCS	vcs_all	`vcs`
*DVE	dve	`dve`
*Verdi	verdi	`verdi`
ICC	ic_compiler	`icc_shell`
ICC2	icc2	`icc2_shell`
*IC Validator	icvalidator	`icv`
*Cosmos Scope	cosmosscope	`cscope`
*Custom Compiler	cust_compiler	`custom_compiler`
Custom Waveview	custom_wv_adv	`wv`
Formality	fm	`fm_shell`
Star RC	starrc	`StarXtract`
PrimeWave	p_wave	`pw_shell` `primewave`
IC Validator Workbench	icv_wb	`icvwb`
PrimeTime	pt	`pt_shell` `primetime`
Design Vision	syn	`design_vision`
Design Compiler	syn	`dc_shell`
TestMAX DFT (Synthesis)	syn	`de_shell`
TestMAX ATPG	tx	`tmax2`
PrimeSim	p_sim	`primesim`
PrimeSim XA	customsim	`xa`
*PrimeSim HSPICE	hspice	`hspice`

Cuadro 10: Aplicaciones necesarias

Es importante mencionar que algunas de las aplicaciones se pueden encontrar en la misma carpeta. Tal es el caso de *Design Vision*, *Design Compiler* y *TestMAX DFT (Synthesis)*, herramientas designadas a la síntesis y cuyo nombre de carpeta es *syn*. De igual manera, la carpeta de *PrimeTime* (pt) contiene tres otras herramientas: *PrimeECO*, *PrimePowr* y *PrimeShield*. En estos casos, se decidió instalar todas las herramientas que se encontrara en la carpeta descargada.

Por otro lado, hay algunas aplicaciones que dependen de otras para poder ser instaladas. Tal es el caso de *DVE*, la cual es una extensión para *VCS*. De igual manera, las últimas versiones de *Custom Compiler* necesitan de *Primewave* para funcionar; de hecho los archivos

de instalación de *Primewave* van incluídos en los de *Custom Compiler*. En estos casos, el algoritmo se realizó de tal manera que se instalara la herramienta independiente primero y luego la dependiente.

La mayoría de aplicaciones tiene el mismo o similar algoritmo de instalación. Normalmente se utiliza *Synopsys Installer* para instalar el producto, referenciando la carpeta donde se descargaron los productos e indicando dónde se instalará la app. Luego se agrega la variable de entorno, cuya estructura es, generalmente, la siguiente:

```
PATH=/usr/synopsys/<nombre>/<version>/bin:$PATH
```

Obteniendo la variable <nombre> y <version> del archivo *common* que se haya descargado. Sin embargo, para algunas aplicaciones esta estructura cambia un poco. Las aplicaciones con un asterisco (*) que se muestran en el Cuadro 10 presentan cambios en su proceso de instalación.

7.3.1. VCS y DVE

Estas aplicaciones deben descargarse juntas y con las mismas versiones. Se debe instalar antes VCS para que la instalación de DVE sea posible, pues esta es una extensión de VCS. La variable de entorno de VCS sí sigue la estructura mostrada anteriormente, sin embargo para DVE, por ser una extensión, no se agrega una variable de entorno. Para VCS, es necesario exportar la variable de entorno `VCS_HOME` en el archivo .bashrc con la dirección del directorio de instalación de VCS.

7.3.2. Verdi y IC Validator

Para estas dos aplicaciones el proceso de instalación es la misma que se describió anteriormente. La única diferencia es que debe de exportarse una variable de entorno adicional: `VERDI_HOME` para verdi y `ICV_HOME_DIR` para IC Validator, con la dirección del directorio de instalación de la aplicación que le corresponda.

7.3.3. HSPICE y Cosmos Scope

Ambas aplicaciones son instaladas utilizando *Synopsys Installer*, sin embargo al exportar la variable de entorno `PATH`, la estructura cambia levemente. Para *HSPICE* la estructura es la siguiente:

```
PATH=/usr/synopsys/<nombre>/<version>/<nombre>/bin:$PATH
```

Para *Cosmos Scope*, la estructura es:

```
PATH=/usr/synopsys/<nombre>/<version>/ai_bin:$PATH
```

CAPÍTULO 8

Descarga e instalación por línea de comandos

Se crearon 5 bash scripts independientes entre sí para la descarga e instalación de librerías y aplicaciones de Synopsys. El script principal y más completo tiene el nombre de `SNPS_APPS.sh`. En este capítulo se estará explicando el funcionamiento y las secciones de código de este script.

Los otros 4 scripts se crearon para llevar a cabo funciones aisladas y están compuestos de las mismas funciones o versiones más simplificadas de las mismas de las utilizadas en el script principal. En el Cuadro 11 se describen los nombres de los scripts creados y su función principal.

Script	Función
SNPS_APPS.sh	Descargar e instalar librerías, aplicaciones y el instalador de Synopsys.
SNPS_DWNLD.sh	Descarga de las aplicaciones.
SNPS_INST.sh	Instalación de las aplicaciones descargadas.
SNPS_INSTALLER.sh	Instalación o actualización de Synopsys Installer.
SNPS_LIBS.sh	Descarga e instalación de EDKs y PDKs, según su tecnología.

Cuadro 11: Bash Scripts creados

La ejecución de los scripts secundarios es especialmente útil si el usuario desea realizar alguna descarga o instalación rápida. También es útil si se necesitan descargar e instalar todas las aplicaciones. Puesto que el proceso de descarga es el que lleva mayor tiempo se puede dejar corriendo el script `SNPS_DWNLD.sh` durante la noche y luego ejecutar `SNPS_INST.sh` para instalar lo descargado, pues el proceso de instalación sí necesita de confirmación del usuario. Solamente se debe de verificar que las versiones escogidas en ambos scripts sean las mismas.

8.1. Funcionamiento

Lo primero que sucede al ejecutar el script es pedirle al ususario el usuario y contraseña de su cuenta de Solvnet Synopsys. Si cualquiera de estos datos es incorrecto se desplegará el error `Login failed: Login incorrect` y se deberá parar la ejecución.

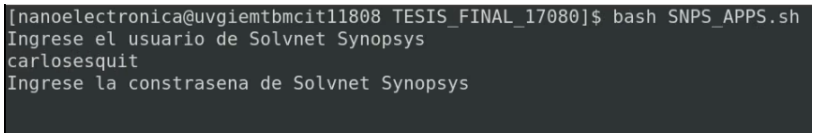


Figura 9: Ejecución de SNPS_APPS.sh

Si el usuario y contraseña son correctos se actualizará el listado de versiones de todas las aplicaciones y librerías. Esto lo hace con la función `ls_products`, la cual se describirá más adelante. Posterior a esto, se desplegará el menú principal (Figura 10) el cual da la opción de descarga de EDKs, PDKs o Aplicaciones. Para salir de la aplicación, se debe de enviar cualquier valor que no esté dentro del rango aceptado (0-2).

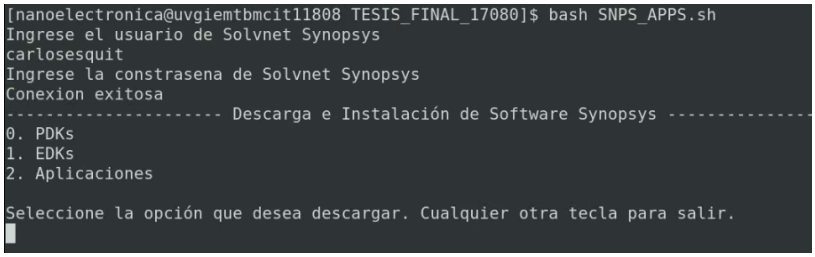


Figura 10: Menú principal de SNPS_APPS.sh

Si se escoge la opción 0, se desplegará los PDKs con sus tecnologías disponibles. En este caso: 90, 32 y 14nm. Como se puede observar en la Figura 11, la aplicación permite navegar entre los menús utilizando la tecla `b` para regresar al menú anterior.

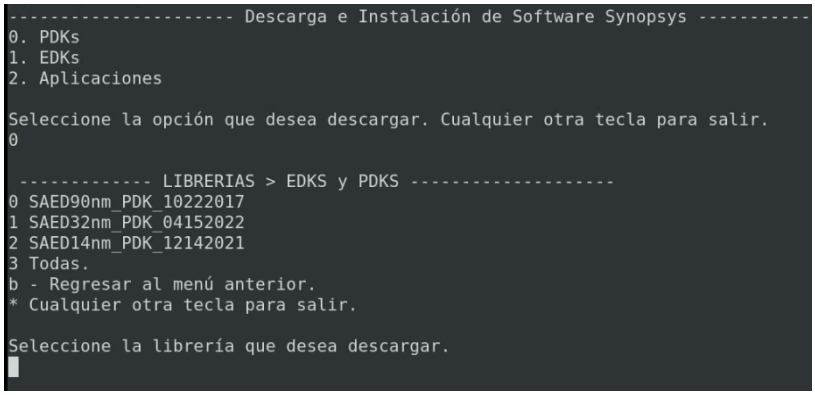


Figura 11: Menú de PDKs de SNPS_APPS.sh

Si se escoge la opción 1, se desplegará los EDKs con sus tecnologías disponibles. En este

caso: 90, 32nm. Es importante recalcar que sí se tiene acceso a las librerías EDK de 14nm, pero estas se encuentran contenidas en el los archivos PDKs. Por lo que si se desea descargar los EDKs, se deben de descargar los PDKs de 14nm y luego trasladas los archivos necesarios a la carpeta de Folder de Trabajo.

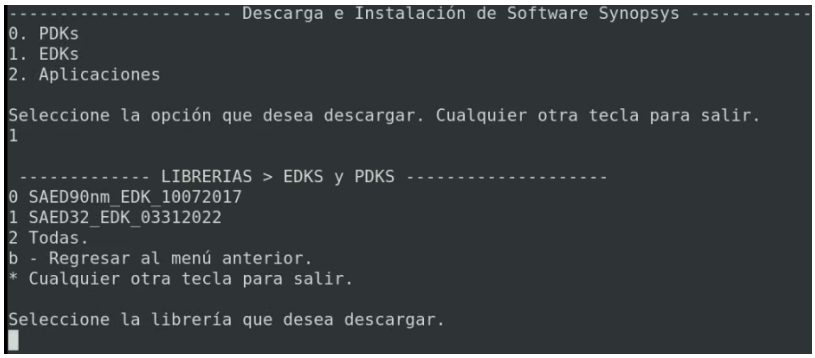


Figura 12: Menú EDKs de SNPS_APPS.sh

Finalmente, si se escoge la opción 2, lo primero que hará el código será verificar si el usuario tiene *Synopsys Installer* instalado, pues este es fundamental para la instalación de las demás aplicaciones. Como se puede observar en la Figura 13, se despliega la versión que se tiene instalada y se pregunta si se desea actualizarla. En caso positivo se despliegan las versiones disponibles.

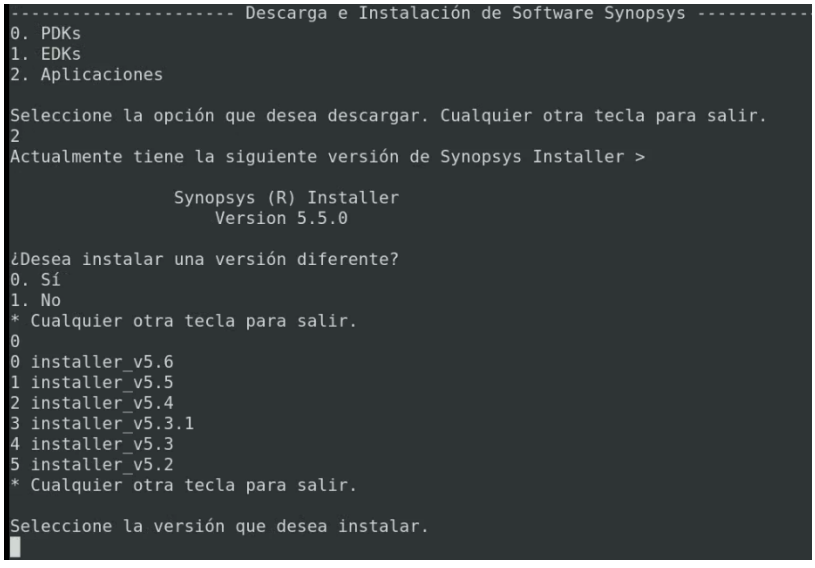


Figura 13: Verificación de Synopsys Installer en SNPS_APPS.sh

Posterior a la verificación de *Synopsys Installer* se procede a desplegar el listado de aplicaciones que este código es capaz de instalar. Al escoger la aplicación, se despliegan las versiones disponibles como se observa en la Figura 14.

Figura 14: Menú de aplicaciones de SNPS_APPS.sh

8.1.1. Requisitos de ejecución

Se creó un archivo `README.txt` donde se especifica los pasos a seguir para asegurar una ejecución del script exitosa. Estos pasos solamente deben de realizarse una vez. Este archivo se puede encontrar en el anexo 13.1.

En términos de dependencias solamente es necesario instalar el programa `lftp`. Esto se hace ejecutando el siguiente comando como root:

```
# yum install lftp
```

También es importante verificar que los directorios y variables de usuario existan y tengan el formato especificado en el archivo README.txt.

8.1.2. Variables

En el Cuadro 12 se listan algunas de las variables de uso más general en el script. Las primeras cuatro son variables de usuario, lo cual quiere decir que el usuario puede cambiar el valor de las mismas según sus necesidades. Estas se pueden modificar en las primeras líneas del script. Se hará referencia a este listado en la explicación de las funciones utilizadas.

Variable	Descripción	Tipo
SRC_DIR	Directorio en el que se guardarán los archivos de descarga. Se recomienda usar $HOME/Downloads .	String
TRGT_DIR	Directorio en el que se guardarán los archivos e instalación. Se recomiendo usar usr/synopsys .	String
VERSION_ALL	Versión seleccionada al descargar e instalar todas las aplicaciones. La versión más reciente: VERSION_ALL=0.	Int
NUMB_VERSIONS	Cantidad de versiones a desplegar.	Int
SFTP_USER	Usuario de Solvnet Synopsys.	String
SFTP_PSW	Contraseña de la cuenta de Solvnet Synopsys.	String
SFTP_SERVER	Dirección del servidor de Synopsys. En este caso: eft.synopsys.com.	String
HERE	Directorio desde donde se ejecuta el script.	String
apps_formal	Listado de aplicaciones que es posible descargar.	Array
apps_real	Listado del nombre de las carpetas donde se encuentran las aplicaciones a descargar en el servidor.	Array
contador	Contador utilizado para enumerar el listado de aplicaciones, librerías o versiones.	Int
option	Selección del usuario para el menú principal. 0 - PDKs, 1 - EDKs, 2 - Aplicaciones, * - Salir.	String
numapp	Selección del usuario para el listado de aplicaciones.	String
up_lim		Int
num_ver	Selección del usuario para el listado de versiones de la aplicación escogida.	String
ver_app	Listado de versiones disponibles para la aplicación seleccionada.	Array
ver_app2	Listado de versiones disponibles para DVE.	Array
var_sep	Utilizada para guardar una cadena que se haya separado.	String

Cuadro 12: Variables generales

8.1.3. Obtención de versiones

Para obtener el listado de las aplicaciones y versiones más actualizado se debe llamar a la función `ls_products()`. La estructura de esta función es la siguiente:

```
ls_products() {
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexión exitosa"
cd Libraries
ls > libsyn.txt
cd ..
cd site31759/MyProducts/rev
ls > myproductsrev.txt
cd ..
cd auth
ls > myproductsauth.txt
```

```
cd ..
cd rev_o
ls > myproductsrevold.txt
bye
EOF
awk '{print $9, "rev"}' myproductsrev.txt > tmp_file && mv tmp_file
↪  myproductsrev.txt
awk '{print $9, "rev_o"}' myproductsrevold.txt > tmp_file && mv tmp_file
↪  myproductsrevold.txt
awk '{print $9, "auth"}' myproductsauth.txt > tmp_file && mv tmp_file
↪  myproductsauth.txt
awk '{print $9}' libsyn.txt > tmp_file && mv tmp_file libsyn.txt
}
```

Esta función se ubica en cada una de las carpetas en el servidor de Synopsys donde podrían existir productos y redirecciona la salida del comando `ls` a un archivo .txt en el sistema local. Luego se eliminan los datos no importantes de estos archivos con el comando `awk`, dejando únicamente el nombre y la versión del producto, y el nombre de la carpeta padre (rev, rev_o o auth).

Luego de obtener el listado completo de las versiones, se debe de filtrar esta lista a la aplicación que haya escogido el usuario, guardada en la variable `apps_real[$numapp]`. Para esto se usa la función `obtener_versiones()` cuya estructura es la siguiente:

```
obtener_versiones() {
echo "Obteniendo el listado de versiones de > ${apps_formal[$numapp]}..."

if (($numapp==0)); then
    read -a var_sep <<< "${apps_real[$numapp]}"
    ver_app=($(awk 'match($0, '/${var_sep[0]}/') {print $1}'
    ↪  myproductsrev.txt myproductsrevold.txt myproductsauth.txt | sort -r
    ↪  | head -$NUMB_VERSIONS))
    ver_app2=($(awk 'match($0, '/${var_sep[1]}/') {print $1}'
    ↪  myproductsrev.txt myproductsrevold.txt myproductsauth.txt | sort -r
    ↪  | head -$NUMB_VERSIONS))
    var_empty $ver_app2
else
    ver_app=($(awk 'match($0, '/${apps_real[$numapp]}/') {print $1}'
    ↪  myproductsrev.txt myproductsrevold.txt myproductsauth.txt | sort -r
    ↪  | head -$NUMB_VERSIONS))
fi

var_empty $ver_app
}
```

Si la aplicación escogida es VCS y DVE, es decir `$numapp`=0, se debe de separar la variable para poder encontrar las versiones tanto de vcs como de dve. El filtrado se hace

puramente con el comando `awk` con la opción de búsqueda de patrones. Luego se ordenó de mayor a menor y se guardó la cantidad de patrones encontrados indicada por la variable `$NUMB_VERSIONS`, en la variable `$ver_app`.

8.1.4. Funciones de protección

El programa se diseñó de tal manera que si el usuario ingresa cualquier valor que no esté dentro del rango aceptado, se saldrá de la aplicación. Para asegurar esto se crearon dos funciones: `verif_input()` y `verif_input2()`. La primera verifica que lo ingresado por el usuario no esté vacío, es decir, no sea un espacio en blanco o un enter, por medio de la opción -z. El segundo *if* de la función verifica que la entrada sea un número. Esto lo hace con la ayuda de una expresión regular definida al prinicipo del script (`re='^[0-9]+$'`).

```
verif_input() {
if [[ -z $1 ]] ; then
    echo "Variable vacía. ¡Adiós!"
    exit 0
fi

if [[ ! $1 =~ $re && "$1" != "b" ]] ; then
    echo "La variable no es un número. ¡Adiós!"
    exit 0
fi
}
```

Si la entrada del usuario no cumple con las condiciones definidas en la función anterior, entonces la variable es un número. Únicamente queda verificar que este número se encuentre dentro del rango de la lista desplegada. En todos los casos, el parámetro `$2` es la variable `contador`, pues este será el último número de la lista desplegada.

```
verif_input2() {
if (($1 < 0 || $1 >= $2)); then
echo "Fuera de rango. ¡Adiós! "
exit 0
fi
}
```

También se crearon funciones para verificar que directorios o archivos de suma importancia para el código existan. La función `verif_folder()` es especialmente útil para verificar que los directorios de instalación o descarga existan en la computadora del usuario y la función `verif_file()` para verificar que los archivos `checksum` existan; sin embargo también son utilizados en diversas partes del código en modo de protección.

```
verif_folder() {
if [ ! -d $1 ]; then
```

```
        echo "ERROR > El directorio $1 no existe. "
        exit 0
fi
}
```

```
verif_file() {
if [ ! -f $1 ]; then
        echo "ERROR > El archivo $1 no existe. "
        exit 0
fi
}
```

Al realizar algunas pruebas con las primeras versiones del script, se tuvieron problemas cuando algunas variables se mantenía vacías por algún error en el algoritmo. Aunque esos errores se han arreglado, se decidió mantener una medida de protección secundaria pues los comandos `rm` usados frecuentemente en el script pueden ser perjudiciales al dispositivo si se usan de manera incorrecta. Para esto, se creó la función `var_empty` que hace uso nuevamente de la opción -z.

```
var_empty() {
var=$1
if [[ -z ${var// } ]] ; then
        echo "ERROR > Variable vacía."
        exit 0
fi
}
```

8.2. Synopsys Installer

8.2.1. Algoritmo

Luego de actualizar el listado de versiones, se revisará si el usuario tiene instalada alguna versión de Synopsys Installer. Si sí, desplegará la versión que tiene y preguntará si desea instalar una versión diferente. Si no, le preguntará al usuario si desea instalarlo pues de lo contrario no podrá instalar otras aplicaciones de Synopsys.

Para desplegar la versión de `Synopsys Installer` se utiliza un HEREDOC para llamar al instalador y se redirecciona la salida de este comando a un archivo de texto. Luego se leen las primeras líneas de este archivo, donde se especifica la versión de `Synopsys Installer`.

En la Figura 15 se muestra el algoritmo de Synopsys Installer en conjunto a algunas de las variables y funciones que utiliza.

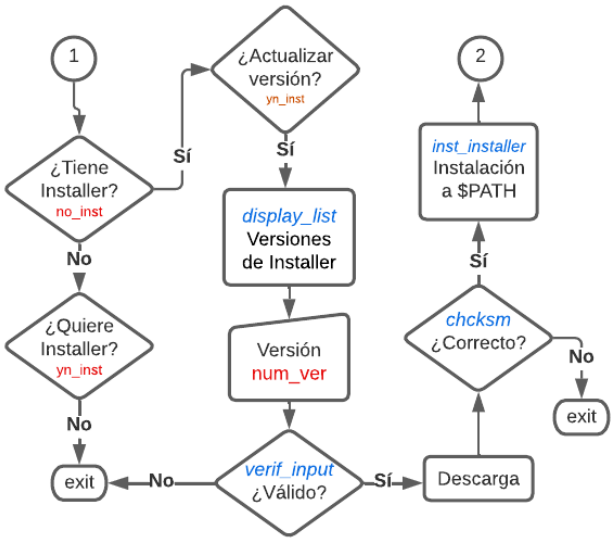


Figura 15: Algoritmo de instalación de Synopsys Installer

8.2.2. Variables

En el Cuadro 13 se describen las variables que se utilizan mayormente en el proceso de instalación de Synopsys Installer. Se harán referencia a estas en la explicación de cada función.

Variable	Descripción	Tipo
yn_inst	Indica si el usuario desea instalar Synopsys Installer. 0 - Sí, 1 - No, * - Salir de la aplicación.	String
no_inst	Indica si Synopsys Installer está instalado en la computadora del usuario. 0 - Sí, 1 - No.	Int
ver_installer	Listado de versiones disponibles de Synopsys Installer.	Array
num_ver	Número de versión seleccionada por el usuario.	Int

Cuadro 13: Variables para el algoritmo de Synopsys Installer

8.2.3. Funciones

Para instalar Synopsys Installer se utiliza la función `inst_installer()` cuya estructura es la siguiente:

```
1 inst_installer() {
2 cd "$SRC_DIR/${ver_installer[$num_ver]}"
3
4 chmod 755 *.run
5 ./*.run << EOF
```

```
$TRGT_DIR/installer
EOF

verif_folder "$TRGT_DIR/installer"

echo "Actualizando variables de entorno.."
if ! grep -q PATH=$TRGT_DIR/installer ~/.bashrc; then
echo 'PATH='$TRGT_DIR'/installer:$PATH' >> ~/.bashrc
fi
cd $HERE
}
```

Primero se le da permisos de ejecución al archivo, luego se corre dentro de un HEREDOC para poder instanciar el directorio de instalación dentro del script. Finalmente se exporta el `$PATH` del comando para poder ejecutarlo desde cualquier carpeta.

8.3. Descarga

8.3.1. Algoritmo

En la Figura 16, se muestra el diagrama de flujo para el proceso de descarga de archivos. Esto incluye la elección de la aplicación y versión del producto, la conexión al servidor por medio de SFTP, la eliminación carpetas de descarga viejas como método de protección y la verificación del checksum.

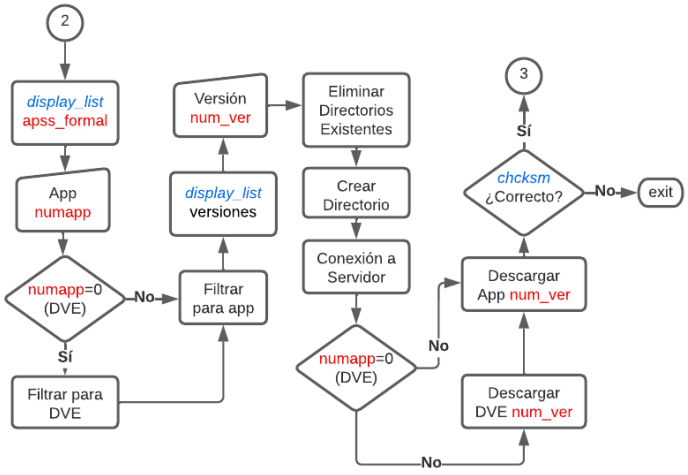


Figura 16: Algoritmo de descarga

8.3.2. Funciones

Como se mencionó anteriormente, la conexión al servidor de Synopsys se hace por medio del protocolo SFTP y con la ayuda de un HEREDOC. La estructura básica es la siguiente:

```
echo "Conectando al servidor de Synopsys ..."
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexion exitosa"
cd site31759/MyProducts
cd <dir_product>
cd <name_version>
mget * -O "$SRC_DIR/<name_version>}"
bye
EOF
```

Donde `<dir_product>` puede ser alguna de las tres carpetas donde se encuentran las aplicaciones: rev, rev_o o auth; y `<name_version>` es la carpeta donde se encuentra la aplicación y versión escogida por el usuario. Los archivos se descargarán en una carpeta con el mismo nombre en el directorio local.

La conexión por SFTP es casi siempre rápida, sin embargo si llegara a fallar, el comando lftp intenta volver a conectar luego de unos segundos. La función usada en el script para realizar la descarga de las aplicaciones o el installer se llama `descarga_apps()` y el de las librerías `descarga_libs()`

Para verificar el checksum primero se accede a la carpeta donde se descargaron todos los archivos (`<name_ver>`) y se crea un archivo .txt donde se guardarán los valores de los checksum obtenidos de cada archivo. Esto se realiza con el comando cksum excluyendo los archivos de checksum y el log del installer. Luego se comparan los dos archivos .txt del checksum. Si son exactamente iguales, significa que sí se ha descargado exitosamente.

```
1  chcksm() {
2  echo "Verificando Checksum de $1..."
3  cd "$SRC_DIR/$1"
4  > checksum_real.txt
5  if [ ! -f checksum_info.txt ]; then
6          echo "El archivo checksum_info.txt no existe. Verifique la descarga
           ↪  de archivos."
7          exit 0
8  fi
9  sed -n '1,2p' checksum_info.txt >> checksum_real.txt
10 cksum !(installer.log|*checksum*) >> checksum_real.txt
11 if [[ $(diff -w checksum_info.txt checksum_real.txt) = "" ]]; then
12     echo "Checksum Correcto. Descarga de $1 exitosa."
13 else
14     echo "Los archivos de $1 no se han descargado correctamente. Intentelo
       ↪  de nuevo."
```

```
#     exit 0
fi
if ls *.gz &>/dev/null; then
        gzip -d *.gz
fi
cd "$HERE"
}
```

Adelantándonos al proceso de instalación, se incluyó un *if* que busca archivos comprimidos y los descomprime para que no hayan problemas con la instalación.

8.4. Instalación

8.4.1. Variables

En el Cuadro 14 se describen las variables que se utilizan mayormente en el proceso de instalación de las aplicaciones, aunque algunas también son utilizadas en el proceso de descarga. Se harán referencia a estas en la explicación de cada función.

Variable	Descripción	Tipo
name_cmn	Nombres de los archivos common encontrados en los archivos descargados.	Array
string	Indica si Synopsys Installer está instalado en la computadora del usuario. 0 - Sí, 1 - No.	Array
delimiter	Patrón de caracteres que se usará para separar una string. En este caso: "_v".	String
ver_arr	Número de versión seleccionada por el usuario.	Array
name_app	Nombres de la app, obtenidos de la separación de la variable name_cmn.	Array
version	Versiones de las apps, obtenidos de la separación de la variable name_c,m.	String
dir_inst	Directorio en el servidor de Synopsys donde se encuentran los archivos de la aplicación seleccionada.	String

Cuadro 14: Variables para el algoritmo descarga e instalación

8.4.2. Funciones

Los archivos de instalación se guardan dentro del `TRGT_DIR` adentro de las carpetas <name>/<version> de la aplicación. Se notó que el nombre de estos directorios era el mismo del archivo *common*. Por esa razón, se realizó una función (`sep_cmn`) para separar el nombre y la versión de ese archivo common. Estas variables las usaremos al definir las variables de entorno y al navegar entre los directorio de instalación.

```
sep_cmn() {
echo "Obteniendo nombre y versión de archivos common..."
arr=()
name_app=()
version=()
delimiter="_v"
ver_arr=()
contador=0

verif_folder "$SRC_DIR/$1"
cd "$SRC_DIR/$1"

name_cmn=($(ls *common*))
var_empty $name_cmn

for item in "${name_cmn[@]}"; do
    string=$item$delimiter
    while [[ $string ]]; do
        if [[ $contador -eq 0 ]]; then
            name_app+=( "${string%%"$delimiter"*}" )
            string=${string#*"$delimiter"}
            let contador++
        else
            ver_arr+=( "${string%%"$delimiter"*}" )
            string=${string#*"$delimiter"}
            let contador--
        fi
    done
done
for value in ${ver_arr[@]}; do
    IFS_OLD=$IFS
    IFS='_'
    read -a var_sep <<< "$value"
    version+=( ${var_sep[0]} )
    IFS=$IFS_OLD
done
var_empty $name_app
var_empty $version
cd "$HERE"
}
```

El resultado de esta función son dos arrays: `name_app`, el cual contiene los nombres y `version` el cual contiene las versiones de los archivos common encontrados.

También se creó una función para realizar la instalación con `Synopsys Installer` llamada `instalacion()`. Lo primero que hace esa función es eliminar todos los archivos de instalación que pudieran existir de versiones diferentes a la que se desea instalar para evitar cualquier tipo de conflicto. En esta función, los parámetros `$1` y `$2` son las variables

`name_app` y `version` obtenidos con la función `sep_cmn()`. Finalmente, se llama al instalador indicando el directorio fuente y el destino.

```
instalacion() {
echo "Eliminando archivos de instalación de versiones diferentes..."
if [ -d "$TRGT_DIR/$1" ]; then
cd "$TRGT_DIR/$1"
rm -rf !($2)
fi
# Llamar al installer
echo "Instalando ${ver_app[$num_ver]}.."
#installer -source "$SRC_DIR/${ver_app[$num_ver]}" -target "$TRGT_DIR"
cd "$HERE"
}
```

Para que los comandos de cada herramienta puedan ser ejecutados sobre cualquier directorio, es necesario agregar las direcciones al `$PATH` de usuario. Se desea que solamente exista un `$PATH` por aplicación, por lo que al instalar versiones nuevas es necesario borrar las direcciones de versiones viejas. Esto lo hacemos con el comando `sed -i` que buscará el patrón indicado en la variable `name_app` y eliminará la línea completa del bashrc. Luego de esto, ya se pueden agregar las variables de entorno según corresponda a cada aplicación.

```
set_env() {
contador=0
echo "Actualizando las variables de entorno de ${ver_app[$num_ver]}..."
for y in "${name_app[@]}"; do
    sed -i '/\/'$y'\//d' ~/.bashrc
    echo "Nombre: $y"
    echo "Versión: ${version[contador]}"

    # Variables de entorno para HSPICE
    if (($numapp==17)); then
    echo 'PATH='$TRGT_DIR'/'$y'/'${version[contador]}'/'$y'/bin:$PATH' >>
    ↪  ~/.bashrc
    # Variables de entorno para CSCOPE
    elif (($numapp==5)); then
    echo 'PATH='$TRGT_DIR'/'$y'/'${version[contador]}'/ai_bin:$PATH' >>
    ↪  ~/.bashrc
    else
    # Variables de Entorno para las demás Apps
    echo 'PATH='$TRGT_DIR'/'$y'/'${version[contador]}'/bin:$PATH' >>
    ↪  ~/.bashrc
    fi

    # Instalación y variables de entorno extra para DVE
    if (($numapp==0)); then
```

```
22      echo "Instalando ${ver_app2[$num_ver]}"
23      installer -source "$SRC_DIR/${ver_app2[$num_ver]}" -target "$TRGT_DIR"
24      echo "Actualizando las variables de entorno de ${ver_app2[$num_ver]}..."
25      sed -i '/VCS_HOME/d' ~/.bashrc
26      echo 'export VCS_HOME='$TRGT_DIR'/'$y'' >> ~/.bashrc
27      elif (($numapp==4)); then
28      # Instalación y variables de entorno extra para IC Validator
29      sed -i '/ICV_HOME_DIR/d' ~/.bashrctest
30      echo 'export ICV_HOME_DIR='$TRGT_DIR'/'$y'' >> ~/.bashrc
31      elif (($numapp==1)); then
32      # Instalación y variables de entorno extra para IC Validator
33      sed -i '/VERDI_HOME/d' ~/.bashrc
34      echo 'export VERDI_HOME='$TRGT_DIR'/'$y'' >> ~/.bashrc
35      fi
36
37      let contador++
38  done
39  sed -i '/export PATH/d' ~/.bashrc
40  echo 'export PATH' >> ~/.bashrc
41  }
```

En la función anterior se exportan las variables de entorno según la elección de la aplicación y finalmente se exporta el PATH.

8.5. Librerías

8.5.1. Variables

En el Cuadro 15 se describen las variables utilizadas para el proceso de descarga de las librerías. Las primeras dos son variables de usuario que pueden ser cambiadas según las necesidades del usuario.

8.5.2. Funciones

Luego de realizar la descarga de la librería escogida, se debe de actualizar las variables de entorno. Esto es necesario solamente para los PDKs. Dentro de los archivos PDK, synopsys provee un script llamado `INSTALL.sh` el cual genera la variable de entorno y la guarda en un archivo de texto llamado `SOURCEME`.

La función `set_env_libs()` busca este script y actualiza el archivo bashrc. Si no se encuentra el script, el usuario deberá hacerlo manualmente.

```
1   set_env_libs() {
2   if (($option==0)); then
```

Variable	Descripción	Tipo
EDK_DIR	Directorio en el que se guardarán las librerías EDKs. Se recomiendo usar $HOME/Desktop/Folder_de_Trabajo .	String
PDK_DIR	Directorio en el que se guardarán las librerías PDKs. Se recomiendo usar usr/synopsys/iPDK .	String
tech_EDK	Tecnologías disponibles para EDKs (14nm, 32nm y 90nm).	Array
tech_PDK	Tecnologías disponibles para PDKs (14nm, 32nm y 90nm).	Array
tech_LIBS	Es reemplazada por tech_PDK si la variable option=0 o por tech_EDK si se escoge option=1.	Array
LIBS_DIR	Es reemplazada por PDK_DIR si la variable option=0 o por EDK_DIR si se escoge option=1.	String
num_lib	Opción escogida por el usuario para seleccionar la tecnología a descargar.	String
PDK_DIR_ENV	Directorio en el que se encuentran los PDKs a llamar con la variable de entorno.	String

Cuadro 15: Variables para el algoritmo de obtención de librerías

```
3
4  echo "Actualizando variables de entorno de ${tech_PDK[$num_lib]}..."
5  PDK_DIR_ENV=$(find $PDK_DIR/${tech_PDK[$num_lib]} -iname INSTALL.sh)
6
7  if [[ -z ${PDK_DIR_ENV// } ]] ; then
8          echo "WARNING > No se encontro el archivo INSTALL.sh. Se deben
           ↪  actualizar las variables de entorno manualmente."
9  else
10         cd ${PDK_DIR_ENV:0:-10}
11         bash INSTALL.sh
12         read -r line < SOURCEME
13         echo "Actualizando bashrc: $line"
14         echo ${line:7:6}
15         sed -i '/'${line:7:6}'/d' ~/.bashrc
16         echo $line >> ~/.bashrc
17 fi
18 fi
19 cd $HERE
20 }
```

8.6. Pruebas

El Script se corrió en una de las computadoras del laboratorio de Nanoelectrónica y se instalaron todas las últimas versiones de las aplicaciones. Primero se actualizó *Synopsys Installer* a la versión 5.5 y con él se instalaron las demás apps. Para verificar que las aplicaciones se instalaron correctamente, se siguieron las instrucciones de las guías de instalación de cada aplicación proveídas por Synopsys.

La mayoría de ellas solamente consistía en ejecutar, de distintas formas, el comando de cada herramienta. Estos están listados en la tercera columna del Cuadro 10.

Por ejemplo, para VCS se ejecutó el comando `vcs -id -full64` sobre distintos directorios, para verificar que la variable de entorno está cumpliendo su función, como se muestra en la Figura 17. Este comando solamente muestra la versión que está instalada en el dispositivo, la cual es justamente la que se pretendía instalar.

```
[nanoelectronica@uvgiemtbmcit11808 ~]$ vcs -id -full64
vcs script version : T-2022.06
machine name = uvgiemtbmcit11808
machine type = linux64
machine os = Linux 3.10.0-1160.76.1.el7.x86_64
The FLEXlm host ID of this machine is "a4bb6d53d8ae"
Compiler version = VCS T-2022.06-SP1_Full64
VCS Build Date = Aug 28 2022 20:17:55
If you need to renew your license, please email this information to your
Account Manager. To obtain contact information for your Account Manager
please email pass@synopsys.com
[nanoelectronica@uvgiemtbmcit11808 ~]$
```

Figura 17: Prueba VCS

En la Figura 18 se muestra la prueba para *Formality*. Como se puede observar en las primeras líneas, al escribir parte del comando y luego presionar *tab*, el comando se autocompleta o muestra los comandos posibles; entre ellos el comando `fm_shell` el cual abre la shell para trabajar con *Formality*. Como se puede observar, también muestra la versión más actualizada.

```
[nanoelectronica@uvgiemtbmcit11808 ~]$ fm_
fm_auto_eco_wrapper.pl  fm_eco_to_svf          fm_eco_to_svf.pl        fm_mk_script
         fm_olh_invoke.sh        fm_shell
[nanoelectronica@uvgiemtbmcit11808 ~]$ fm_shell

                              Formality (R)

              Version T-2022.03-SP4 for linux64 - Aug 29, 2022

                    Copyright (c) 1988 - 2022 Synopsys, Inc.
   This software and the associated documentation are proprietary to Synopsys,
 Inc. This software may only be used in accordance with the terms and conditions
 of a written license agreement with Synopsys, Inc. All other use, reproduction,
   or distribution of this software is strictly prohibited.  Licensed Products
     communicate with Synopsys servers for the purpose of providing software
    updates, detecting software piracy and verifying that customers are using
    Licensed Products in conformity with the applicable License Key for such
  Licensed Products. Synopsys will use information gathered in connection with
    this process to deliver software updates and pursue software pirates and
                                  infringers.

 Inclusivity & Diversity - Visit SolvNetPlus to read the "Synopsys Statement on
            Inclusivity and Diversity" (Refer to article 000036315 at
                        https://solvnetplus.synopsys.com)

Build: 7777987
Hostname: uvgiemtbmcit11808
Current time: Fri Sep 30 21:40:33 2022

Loading db file '/usr/synopsys/fm/T-2022.03-SP4/libraries/syn/gtech.db'
fm_shell (setup)>
```

Figura 18: Prueba Formality

En otros casos, se desplegó la versión de la aplicación y también se abrió la interfaz

gráfica para verificar que no existiera ningún error con la plataforma. Tal fue el caso de TestMAX ATPG como se puede observar en la Figura 19.

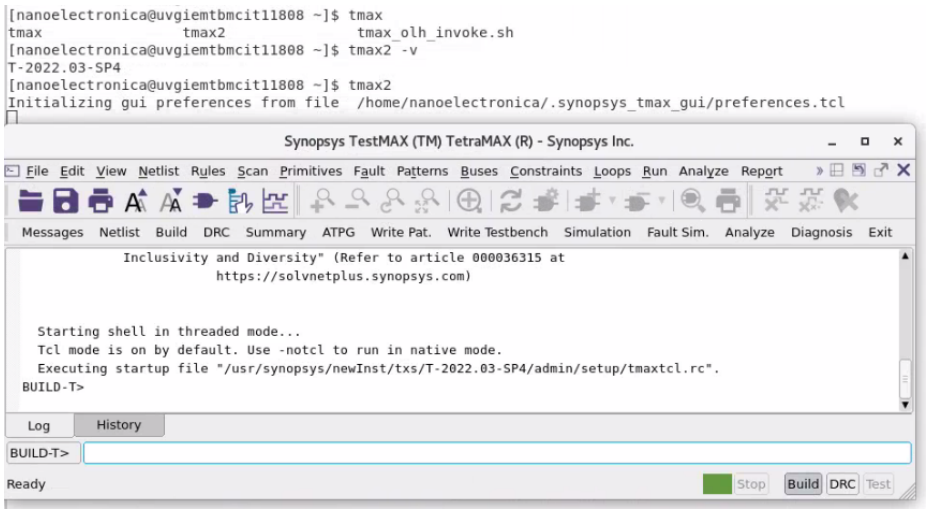


Figura 19: Prueba TestMAX

En la Figura 20 se muestra la interfaz de *Cosmos Scope* al correr el comando `cscope` el cual funcionaba sin importar desde donde se ejecutara.

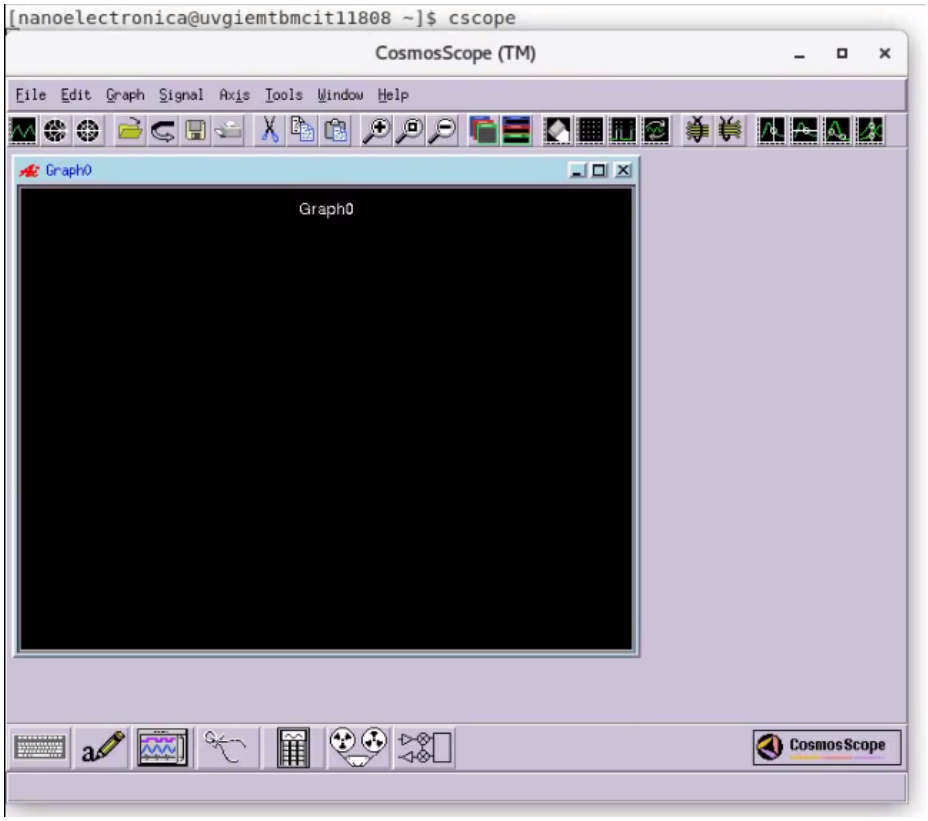


Figura 20: Prueba Cosmos Scope

Este mismo proceso se repitió con todas las Aplicaciones y todas funcionaron de la manera deseada. La única aplicación con un proceso de verificación distinto es *PrimeSim (HSPICE)*, pues esta necesita correr una simulación de un archivo .sp . Dentro de sus archivos de Instalación, contiene un archivo para realizar la demostración llamado *demo.sp*. Se corrió la simulación con el comando: `hspice demo.sp -o demo.lis`. Como se puede observar en la Figura 21, hspice pudo realizar la simulación exitosamente.

```
[nanoelectronica@uvgiemtbmcit11808 bench]$ hspice demo.sp -o demo.lis
Using: /usr/synopsys/newInst/hspice/T-2022.06-SP1/hspice/linux64/hspice  'demo.sp' -o demo.lis

>info:         ***** hspice job concluded
[nanoelectronica@uvgiemtbmcit11808 bench]$
```

Figura 21: Prueba HSPICE demo

Para visualizar los resultados de la simulación se ejecutó el comando `vi demo.lis`. Como se puede observar en la Figura 22, la simulación fue exitosa pues se despliega el texto *job concluded* junto con información del circuito analizado.

```
        ***** job concluded
******
*file power.sp  test of the power supplies

****** job statistics summary tnom=  25.000 temp=  25.000 ******

******  Machine Information  ******
CPU:
model name     : Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
cpu MHz        : 4599.926
CPU(s)         : 16

OS:
Linux version 3.10.0-1160.76.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org)

System loadavg : 1.07 0.88 0.42 2/621 2429


******  PrimeSim HSPICE Threads Information  ******

Command Line Threads Count :     1
Available CPU Count        :    16
Actual Threads Count       :     1


******  Circuit Statistics  ******
# nodes       =       9 # elements   =      13
# resistors   =       1 # capacitors =       4 # inductors   =       4
# mutual_inds =       0 # vccs       =       0 # vcvs        =       0
# cccs        =       0 # ccvs       =       0 # volt_srcs   =       4
# curr_srcs   =       0 # diodes     =       0 # bjts        =       0
# jfets       =       0 # mosfets    =       0 # U elements  =       0
# T elements  =       0 # W elements =       0 # B elements  =       0
# S elements  =       0 # P elements =       0 # va device   =       0
# vector_srcs =       0 # N elements =       0
```

Figura 22: Prueba HSPICE visualización

CAPÍTULO 9

Contenedor de Synopsys

El contenedor de Synopsys proporciona todas las librerías y dependencias necesitadas por las aplicaciones de Synopsys listadas en el Cuadro 8. Este contenedor se puede descargar en el servidor de Synopsys y se basa en la tecnología de Singularity. El contenedor solamente se puede instalar en sistemas Linux. La última versión del contenedor es la 1.8 y es la que se descargó e instaló en esta ocasión, utilizando *Synopsys Installer* versión 5.6.

9.1. Synopsys Installer

Para instalar el contenedor se debe usar *Synopsys Installer* versión 5.1 o más reciente. Se puede hacer este proceso por la interfaz gráfica del installer o por línea de comandos, de la misma manera que con las aplicaciones.

En el directorio root de Synopsys Installer se encuentra un bash script que es usado para instalar, configurar y desplegar el contenedor de Synospys. El nombre de este archivo es `container_setup.sh` y es el único archivo que se encuentra en el directorio `/usr/synopsys/installer/container_setup/` antes de hacer la instalación del contenedor. Este script puede ejecutarse en tres modalidades:

```
./container_setup install -<opciones>
./container_setup config -<opciones>
./container_setup deploy -<opciones>
```

Instalación

Esta opción instala el contenedor en el directorio definido en el parámetro target y crea el archivo de configuración (`.snps_container.txt`) por default en el directorio `/usr/synopsys/installer/container setup/config/`.

Parámetro	Descripción
-target <root dir container>	Directorio donde se instalará el contenedor. En este caso `/usr/synopsys/`.
-source <snps_container dir>	Directorio en el que se encuentran los archivos common del contenedor. En este caso `$HOME/Descargas/container_v1.8/`

Cuadro 16: Parámetros para la opción de instalación

Configuración

El archivo de configuración creado por default puede ser modificado con la modalidad de configuración. Esto es especialmente útil si se necesita cambiar la imagen escogida por default o montar directorios de nuestra computadora al contenedor. Los parámetros posibles para la configuración se muestran en el Cuadro 17.

Parámetro	Descripción
-target <output dir>	Donde se guardará el archivo de configuración del contenedor. El default es `/usr/synopsys/installer/container_setup/config`.
-source <container root dir>	Directorio donde se instaló el contenedor.
-exe <singularity path>	Directorio donde se encuentran el ejecutable de Singularity. En este caso `/opt/singularity-3.10.2`
-image <container img>	La imagen que se usa por default es centos7.simg, pero puede ser cambiada a centos6.simg si fuera necesario.
-bind <mount point>	Con esta opción se pueden agregar paths que no se hayan montado automáticamente al contenedor.
-noautomount	Desactiva el montaje automático de ciertos directorios.

Cuadro 17: Parámetros para la opción de configuración

Despliegue

Esta opción copia y actualiza el archivo de configuración del contenedor en las carpetas de instalación de las aplicaciones habilitadas para el mismo, es decir, las aplicaciones listadas en el Cuadro 8. Para esto es necesario indicarle el directorio donde se encuentran estas aplicaciones con el parámetro `target`. El parámetro `source` es opcional.

Parámetro	Descripción
-target <synopsys tools root>	Directorio donde se encuentran las aplicaciones de Synopsys. En este caso `/usr/synopsys/`.
-source <conf file>	Directorio en el que se encuentran el archivo de configuración del contenedor. En este caso `/usr/synopsys/installer/container_setup/config/`

Cuadro 18: Parámetros para la opción de despliegue

9.2. Archivos de descarga

Al igual que las aplicaciones, el contenedor se puede descargar desde la página `https://solvnet.synopsys.com/DownloadCenter` por medio de HTTP o por linea de comandos por SFTP. En este caso se descargó por medio de SFTP accediendo al directorio `/site31759/MyProducts/rev/container_v1.8/` del servidor de Synopsys y descargando todos los archivos encontrados en esa carpeta. La lista de archivos descargados es la siguiente:

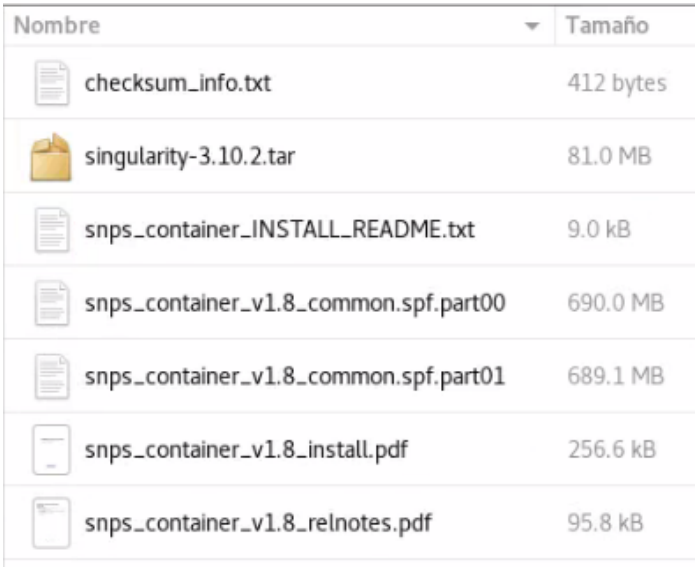


Figura 23: Archivos descargados para Synopsys Container

9.3. Instalación de Singularity

El contenedor de Synopsys está creado sobre la tecnología de Singularity, por lo que es necesario instalarlo en nuestra computadora. Para la versión 1.8 del contenedor, Synopsys recomienda instalar Singularity versión 3.10.2, la cual se encuentra dentro de los archivos de descarga. Para instalarlo se ejecutaron los siguientes comandos en la terminal del directorio de descarga.

```
tar xvfz singularity-3.10.2.tar.gz
```

```
cd singularity-3.10.2
./singularity_install --prefix=/opt/singularity-3.10.2
```

Este último comando se debe de ejecutar como root. El parámetro prefix indica el directorio donde se instalarán los subdirectorios de Singularity.

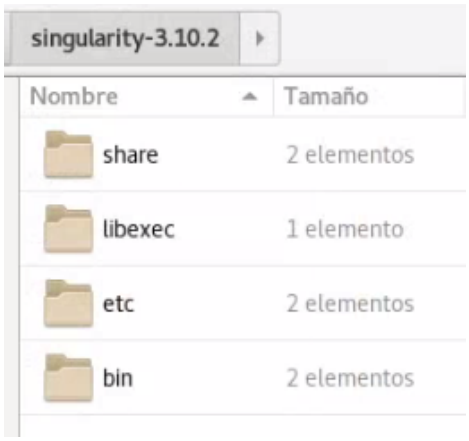


Figura 24: Subdirectorios de Singularity

En la Figura 24 se muestran los subdirectorios de Singularity luego de la instalación. En la carpeta `bin` se encuentra el binario de singularity

9.4. Instalación, configuración y uso

Para realizar la instalación del contenedor por medio de *Synopsys Installer* se ejecutó el siguiente comando:

```
installer -source $HOME/Downloads/container_v1.8 \
-target /usr/synopsys/
```

El cual desplegó la información de la plataforma a instalar como se muestra en la figura 25.

```
Site ID number [000]: 31759
################################################
############### Installation Summary ###############
Product:    Synopsys Singularity CentOS Container Images (snps_container)
Release:    1.8
Platforms:   common
Disk Space: 1375 MB
Target Dir: /usr/synopsys/snps_container/1.8
```

Figura 25: Archivos descargados para Synopsys Container

En los subdirectorios de instalación del contenedor se encuentran tres carpetas: `admin`, la cual contiene la información del sitio, `data`, la cual contiene las imágenes (centos6.simg

y centos7.simg) y `bin` donde se encuentran los shell scripts: `container_conf.sh` y `snps_container.sh` .

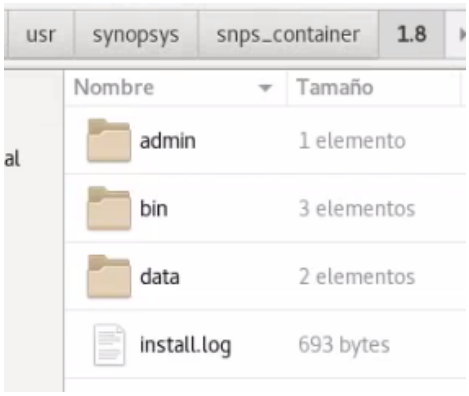


Figura 26: Subdirectorios de Synopsys Container

Para crear el archivo de configuración que se estará usando para correr el contenedor se ejecuta el siguiente comando, especificando la dirección de instalación del contenedor y el binario de Singularity:

```
% pwd
/usr/synopsys/sontainer_setup
% ./container_setup config -source /usr/synopsys/snps_container/1.8 \
-exe /opt/singularity-3.10.2/bin/singularity
```

Esto genera una nueva carpeta `/usr/synopsys/installer/container_setup/config` donde se encuentran los archivos mostrados en la Figura 27.

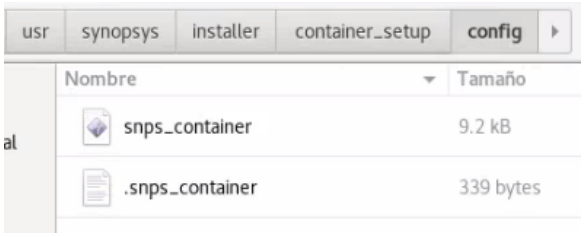


Figura 27: Archivos generados al configurar el contenedor

El archivo `.snps_container` es el archivo de configuración y en la figura tal se muestra su contenido. Por otro lado, el archivo `snps_container` es utilizado para abrir una terminal dentro del contenedor en pasos posteriores.

```
1 #container_root /usr/synopsys/snps_container/1.8
2 #image_centos6 /usr/synopsys/snps_container/1.8/data/centos6.simg
3 #image_centos7 /usr/synopsys/snps_container/1.8/data/centos7.simg
4 path /opt/singularity-3.10.2/bin/singularity
5 image /usr/synopsys/snps_container/1.8/data/centos7.simg
6 automount 1
7 bind
8 banner
9 global_options
10 exec_options
```

Figura 28: Archivo de configuración .snps_container

Luego de crear y modificar el archivo de configuración del contenedor, se debe desplegar el mismo para que pueda ser usado por las aplicaciones de nuestro interés. Para esto, usamos la modalidad `deploy` del archivo `container_setup.sh` indicando la ubicación de instalación de las aplicaciones con el parámetro `target`, como se ve en la Figura 29.

```
[nanoelectronica2021@uvgiemtbmj31308 container_setup]$ ./container_setup.sh deploy -target /usr/synopsys
INFO: Update /usr/synopsys/fm/R-2020.09-SP3/.snps_container
INFO: Update /usr/synopsys/starrc/R-2020.09-SP3/.snps_container
INFO: Skip /usr/synopsys/verdi/P-2019.06-SP2-11, no container enabled tools installed
INFO: Update /usr/synopsys/customcompiler/R-2020.12-2/.snps_container
INFO: Skip /usr/synopsys/snps_container/1.8, no container enabled tools installed
INFO: Skip /usr/synopsys/vcs/P-2019.06-SP2-11, no container enabled tools installed
INFO: Update /usr/synopsys/hspice/R-2020.12-2/.snps_container
INFO: Update /usr/synopsys/nt/R-2020.09-SP3/.snps_container
INFO: Update /usr/synopsys/syn/Q-2019.12-SP5-4/.snps_container
```

Figura 29: Actualización de archivo de configuración

```
[nanoelectronica2021@uvgiemtbmj31308 container_setup]$ ./container_setup.sh config -bind /run,/boot,
/boot/efi,/sys/fs/cgroup,/run/user/1001,/run/user/1002
Install /usr/synopsys/installer/container_setup/config/.snps_container ...
Done
```

Figura 30: Montaje de nuevos directorios

El manual de Synopsys recomienda montar todos los directorios de nuestra computadora que no se hayan montado automáticamente en el contenedor con la opción `bind`, como se observa en la Figura 33. Cada path debe estar separado por una coma.

```
#container_root /usr/synopsys/snps_container/1.8
#image_centos6 /usr/synopsys/snps_container/1.8/data/centos6.simg
#image_centos7 /usr/synopsys/snps_container/1.8/data/centos7.simg
path /opt/singularity-3.10.2/bin/singularity
image /usr/synopsys/snps_container/1.8/data/centos7.simg
automount 1
bind /run,/boot,/boot/efi,/sys/fs/cgroup,/run/user/1001,/run/user/1002
banner
global_options
exec_options
```

Figura 31: Archivo de configuración actualizado

Esto actualizará el archivo `.snps_container`. Luego de hacer cambios en el archivo de configuración, es necesario volver a desplegar el contenedor para que las aplicaciones se actualicen.

Luego de desplegar el contenedor, este ya puede ser usado por las aplicaciones. Es importante recalcar que la aplicación se ejecutará afuera del contenedor pero usando recursos

que se encuentran adentro del mismo. Para utilizarlo, solamente se corre el comando de la aplicación junto con la opción `-container`. Por ejemplo, `cdesigner -container`.

Otra opción es establecer la variable de entorno `SNPS_CONTAINER=1` en el archivo `bashrc`. Esta variable permitirá que siempre se ejecuten las aplicaciones usando recursos del contenedor, sin tener que definir la opción `-container`.

9.5. Explorando el contenedor

Se siguieron los pasos recomendados en el manual de *Synopsys Container* para verificar que el contenedor se haya configurado correctamente. Desde el directorio root se ejecutó `snps_container` de la siguiente manera:

```
% pwd
/
% /usr/synopsys/installer/container_setup/config/snps_container
```

Es importante ejecutarlo en el directorio `root` para que se monten los directorios correctos de forma automática. Al ejecutar este script se abrirá una terminal de Singularity. En esta terminal podemos explorar el contenido del contenedor. En la Figura 32 se observa los directorios `$HOME` del contenedor (imagen izquierda) y nuestra computadora (imagen derecha).

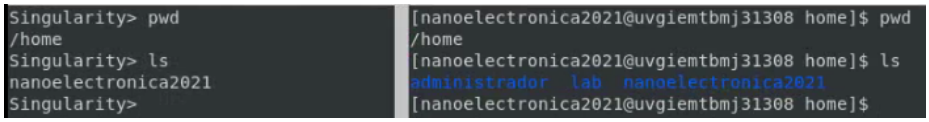


Figura 32: Comparación de directorio $HOME. Izquierda contenedor, derecha computadora local.

Luego se compararon los puntos de montaje entre el contenedor y el sistema local con el montaje automático en el archivo de configuración con el comando `df`. Es decir, sin especificar ningún directorio con la opción `bind`.

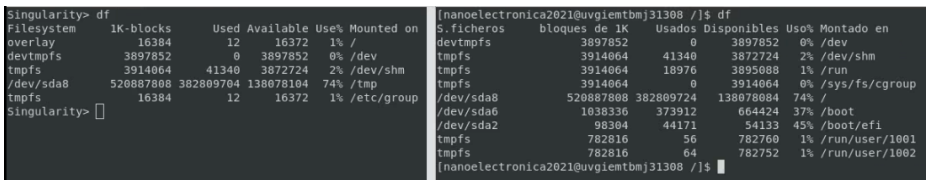


Figura 33: Comparación de puntos de montaje. Izquierda contenedor, derecha computadora local.

Los contenidos del directorio `root` del contenedor se pueden observar en la Figura 34. De especial importancia parecen ser los los enlaces simbólicos `environment` y `singularity` los cuales se direccionan al directorio oculto `.singularity.d/`.

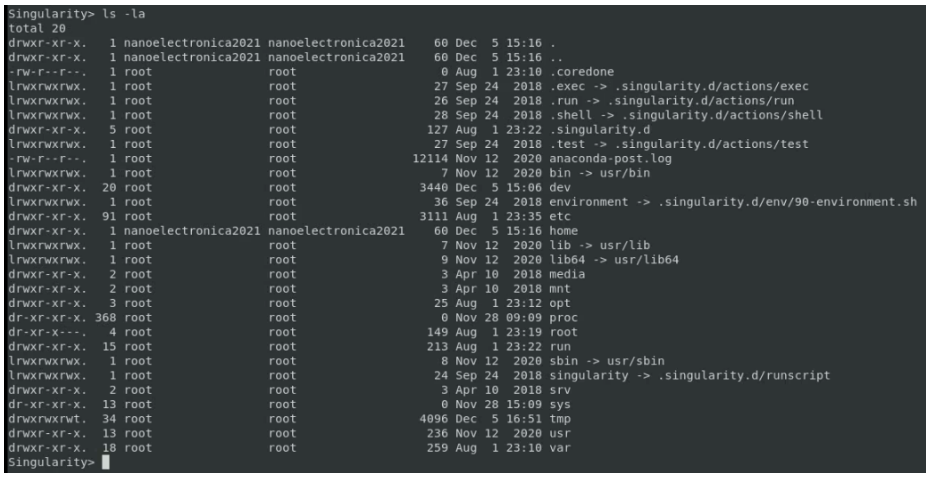


Figura 34: Listado de archivos en el directorio root del contenedor

Dentro de estas carpetas se encuentra una serie de scripts que utiliza el contenedor para obtener los recursos que necesita y ejecutar las aplicaciones.

CAPÍTULO 10

Conclusiones

- Se descargaron e instalaron exitosamente y de manera automática, un total de 18 aplicaciones con las últimas versiones disponibles utilizando el script creado.
- Las aplicaciones instaladas pueden ser ejecutadas desde cualquier directorio lo cual verifica que la variable de entorno PATH fue actualizada exitosamente.
- En el caso que exista una falla durante alguno de los procesos, el script avisará al usuario y terminará la aplicación para evitar posibles conflictos. Esto ocurre cuando hay fallas en la descarga, el usuario ingresa valores no soportados o algunas variables importantes no son actualizadas de manera correcta.
- Actualmente cada vez que se desea ingresar a SolvNet se necesita verificar el usuario por medio del envío de un código al correo del mismo. Esto no sucede al acceder al servidor por medio de SFTP lo cual ahorra tiempo al usuario.

CAPÍTULO 11

Recomendaciones

- La estructura actual del script depende mucho de la estructura del servidor de Synopsys y de los archivos de instalación. Si Synopsys deseara cambiar esto, gran parte del código tendría que modificarse. Se recomienda encontrar maneras dinámicas y automáticas que se acoplen a las nuevas estructuras del Servidor.
- Mejorar el algoritmo de checksum, de tal manera que no tenga que borrarse el directorio de descarga y así aprovechar cuando los archivos ya fueron descargados en el pasado. También es recomendable agregar un warning antes de la eliminación de los archivos para evitar errores de usuario.
- Explorar todas las funciones que provee *Synopsys Installer* para facilitar la instalación de las herramientas. Especialmente el modo de instalación por lotes.
- Agregar una función para verificar que la licencia de synopsys esté agregada en el archivo $HOME/.bashrc.
- Crear una carpeta compartida con todos los estudiantes donde se guarden los archivos de descarga, para no ocupar tanto espacio en cada máquina e implementar esto dentro del script.

CAPÍTULO 12

Bibliografía

[1] Intel. “Intel in Costa Rica. ”dirección: https://www.intel.la/content/www/xl/es/corporate-responsibility/intel-in-costa-rica.html.

[2] J. de los Santos, en *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*, Tesis de Licenciatura en Ingeniería Elctrónica, Universidad del Valle, 2014.

[3] S. H. R. Vásquez, en *Definición del Flujo de Diseño para Fabricación de un Chip con Tecnologia VLSI CMOS*, Tesis de Licenciatura en Ingeniería Elctrónica, Universidad del Valle, 2019.

[4] L. A. Najera, en *Implementación de circuitos sintetizados a nivel netlist a partir de un diseño en lenguaje descriptivo de hardware como primer paso en el flujo de diseño de un circuito integrado*, Tesis de Licenciatura en Ingeniería Elctrónica, Universidad del Valle, 2019.

[5] J. K. y Keith Ross, en *Computer Networking: A Top-Down Approach*, Pearson, 2013, págs. 116-118.

[6] precisely. “FTPS. ”dirección: https://www.precisely.com/glossary/ftps.

[7] S. Academy. “SFTP File Transfer Protocol. ”dirección: https://www.ssh.com/academy/ssh/sftp.

[8] freeCodeCamp. “Shell Scripting for Beginners. ”dirección: https://www.freecodecamp.org/news/shell-scripting-crash-course-how-to-write-bash-scripts-in-linux/.

[9] SS64. “An A-Z Index of the Linux command line: bash + utilities. ”dirección: https://ss64.com/bash/.

[10] M. Cooper. “Advanced Bash-Scripting Guide. ”dirección: https://tldp.org/LDP/abs/html/index.html.

[11] A. Prakash. “Process Substitution: An Uncommon but Advanced Way for Input/Output Redirection in Linux. ”dirección: https://linuxhandbook.com/bash-process-substitution/.

[12] R. Hat. “El concepto de los contenedores Linux. ”dirección: https://www.redhat.com/es/topics/containers.

[13] IBM. “Containers. ”dirección: https://www.ibm.com/cloud/learn/containers.

[14] S. Inc. “Introduction to Singularity. ”dirección: https://sylabs.io/guides/3.5/user-guide/introduction.html.

[15] Synopsys. “Download and Upload Help. ”dirección: https://solvnet.synopsys.com/efthelp#download_filezilla.

CAPÍTULO 13

Anexos

13.1. README.txt

```
Autora > Paola Alejandra Mendizabal Batres
Trabajo de Graduación > Automatización del proceso de instalación
de software de Synopsys e implementación
de mejoras utilizando contenedores.
Fecha > Noviembre de 2022

Este archivo contiene información importante sobre la ejecución
de scripts para descargar aplicaciones y
librerías de Synopsys. Para mayor detalle sobre el código referirse
al documento del trabajo de graduación.

---------------------------------------------------------------
    SNPS_APPS.sh
---------------------------------------------------------------

Descripción General
===============================================================

Este script desplegará un menú principal dando a escoger si se
desea descargar las librerías EDKs, PDKs
o aplicaciones que serán utiles tanto para el curso de Nanoelectrónica
como para el proyecto Gran Jaguar.
También otorga la opción de instalar o actualizar la versión de
Synopsys Installer.

Si se necesita realizar funciones aisladas como descargar los archivos
de la aplicación, instalar aplicaciones,
```

```
instalar Synopsys Installer o descargar las librerías, se pueden
ejecutar los scripts llamados SNPS_DWNLD.sh,
SNPS_INST.sh, SNPS_INSTALLER.sh y SNPS_LIBS.sh, respectivamente.



Pasos para ejecutarlo por primera vez
=============================================================

1. Instalar la librería lftp, corriendo el siguiente comando
como root:

 # yum install lftp

2. Asegurarse que los siguientes directorios existan en su
computadora o crearlos. Se pueden modificar
estas variables en el script a preferencia del usuario, pero
estos directorios son los recomendados.

 # Directorio donde se guardarán los archivos descargados.
 SRC_DIR=$HOME/Downloads
 # Directorio donde se guardarán los archivos instalados.
 TRGT_DIR=/usr/synopsys

 # Directorio donde se guardarán las respectivas librerías.
 EDK_DIR=$HOME/Desktop/Folder_de_Trabajo
 PDK_DIR=/usr/synopsys/iPDK

*Estas variables las puede encontrar al principio del Script en
la sección "VARIABLES DE USUARIO".
IMPORTANTE >> Los nombres de los directorios no deben de contener
espacios.

3*. Si no cuenta con el directorio /usr/synopsys debe de correr
los siguientes comandos como root:

 # mkdir /usr/synopsys
 # chown <usuario> /usr/synopsys (Actualmente el usuario es
 nanoelectronica)

4*. Si se desea, cambiar las siguientes variables:

 NUMB_VERSIONS > Cantidad de versiones a desplegar (por default 6)
 VERSION_ALL > Número de versión a instalar al escoger la opción
 "Todas las aplicaciones"
 (por default 0, la cual representa la versión más reciente).

*Estas variables las puede encontrar al principio del Script en la
```

sección "VARIABLES DE USUARIO".

5. Asegurarse de tener acceso a internet.

6. Correr el código con:

```
bash SNPS_APPS.sh
```

IMPORTANTE >> El nombre del directorio donde se ejecute tampoco debe de contener espacios.

13.2. SNPS_APPS.sh

```
#! /bin/bash
# Autor > Paola Alejandra Mendizabal Batrees
# Objetivo > Descarga e Instalación de Aplicaciones de Synopsys para el
Laboratorio de Nanoelectrónica y proyecto GranJaguar.
# Fecha > Septiembre de 2022

# ++++++++++++++++++++++++++++ VARIALBES ++++++++++++++++++++++++++++++

echo "Ingrese el usuario de Solvnet Synopsys"
read SFTP_USER
echo "Ingrese la constraseña de Solvnet Synopsys"
read -s SFTP_PSW

SFTP_SERVER=eft.synopsys.com
SRC_DIR=/home/nanoelectronica/Downloads
TRGT_DIR=/usr/synopsys/newInst
ACT_DIR=$(pwd)

apps_formal=('VCS y DVE' 'Verdi' 'ICC' 'ICC2' 'IC Validator' 'Cosmos Scope'
'Custom Compiler' 'Custom Waveview' 'Formality' 'STAR RC' 'PrimeWave'
'IC Validator Workbench' 'Primetime' 'NanoTime' 'Design Vision,
Design Compiler y TestMAX DFT (Synthesis)' 'TestMax (Spyglass)'
'TestMax ATPG' 'TestMax Manager' 'TestMAX VTRAN' 'PrimeSim' 'PrimeSim XA'
'PrimeSim HSPICE')

apps_real=('vcs_all dve' 'verdi' 'ic_compiler' 'icc2' 'icvalidator'
'cosmosscope' 'cust_compiler' 'custom_wv_adv' 'fm' 'starrc' 'p_wave'
'icv_wb' 'pt' 'nanotime' 'syn' 'spyglass' 'tx' 'testmax' 'vtran' 'p_sim'
'customsim' 'hspice')

re='^[0-9]+$'
shopt -s extglob
contador=0
```

```
# ++++++++++++++++++++++++++ BANDERAS ++++++++++++++++++++++++++
is_inst=0           # Se activa si el installer no está instalado.
wnt_inst=0           # Se activa si se desea instalar el installer

# ++++++++++++++++++++++++++ FUNCIONES ++++++++++++++++++++++++++

ls_products() {
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexion exitosa"
cd site31759/MyProducts/rev
ls > myproductsrev.txt
cd ..
cd auth
ls > myproductsauth.txt
cd ..
cd rev_o
ls > myproductsrevold.txt
bye
EOF
}

verif_input() {
if [[ -z $1 ]] ; then
    echo "Adios!"
    exit 0
fi

if [[ ! $1 =~ $re ]] ; then
    echo "Adios!"
    exit 0
fi
}

verif_input2() {
if (($1 < 0 || $1 >= $2)); then
echo "Fuera de rango. Adios! "
exit 0
fi
}

display_list() {
contador=0
arr=("$@")
for x in "${arr[@]}"; do
        echo "$contador $x"
        let contador++
done
```

```
}

var_empty() {
var=$1
if [[ -z ${var// } ]] ; then
        echo "Error. Variable vacia."
        exit 0
fi
}

chcksm() {
cd $SRC_DIR/$1
> checksum_real.txt
sed -n '1,2p' checksum_info.txt >> checksum_real.txt
cksum !(installer.log|*checksum*) >> checksum_real.txt

if [[ $(diff -w checksum_info.txt checksum_real.txt) = "" ]]; then
    echo "Checksum Correcto. Descarga de $1 exitosa."
pwd
else
    echo "Los archivos de $1 no se han descargado correctamente.
    Inténtelo de nuevo."
    exit 0
fi

# Descomprimir archivos para la instalcion
if ls *.gz &>/dev/null; then
        gzip -d *.gz
fi
cd $ACT_DIR
}

sep_cmn() {
cd $SRC_DIR/$1
pwd
IFS_OLD=$IFS
name_cmn=$(ls *common* | head -1)
echo $name_cmn
IFS='_'
read -a var_sep <<< "$name_cmn"
name_app=${var_sep[0]}
version=${var_sep[1]:1}
IFS=$IFS_OLD
cd $ACT_DIR
}

instalacion() {
if [ -d $TRGT_DIR/$1 ]; then
```

```
cd $TRGT_DIR/$1
rm -rf !($2)
fi
# Llamar al installer
echo "Instalando ${ver_app[$num_ver]}"
installer -source "$SRC_DIR/${ver_app[$num_ver]}" -target $TRGT_DIR
cd $ACT_DIR
}

# ++++++++++++++++++++++++ ALGORITMO INSTALLER ++++++++++++++++++++++

echo "---- Descarga e Instalación de Software Synopsys --------------"

# ------ Actualizar lista de versiones y eliminar datos no importantes -----
echo "¿Desea actualizar el listado de versiones de las aplicaciones?"
read yn_act
while [ 1 ]
do
    case $yn_act in
    [Yy]* )
    echo "Obteniendo versiones de las aplicaciones ..."
    ls_products
    awk '{print $9, "rev"}' myproductsrev.txt > tmp_file &&
    mv tmp_file myproductsrev.txt
    awk '{print $9, "rev_o"}' myproductsrevold.txt > tmp_file
    && mv tmp_file myproductsrevold.txt
    awk '{print $9, "auth"}' myproductsauth.txt > tmp_file
    && mv tmp_file myproductsauth.txt
    break;;
    [Nn]* )
    echo "Continuando.."
    break;;
    * ) echo "Inválido. Vuelva a intentarlo."
    read yn_act;;
    esac
done

# --------- Tiene el Installer? -------------------------
if command -v installer &>/dev/null; then
        echo "Actualmente tiene la siguiente versión de
        Synopsys Installer >"
installer > inst.txt << EOF
EOF
head -4 inst.txt
        echo "¿Desea instalar una versión diferente? (y/n)"
        read yn_inst
else
        echo "Synopsys Installer no se encuentra instalado. Sin él
```

```
        no podrá instalar otras aplicaciones de Synopsys."
        echo "¿Desea instalarlo? (y/n)"
        is_inst=1
        read yn_inst
fi
# --------- Quiere el Installer? -------------------------
while [ 1 ]
do
    case $yn_inst in
    [Yy]* )
    wnt_inst=1
    break;;
    [Nn]* )
    if [[ $is_inst == 1 ]]; then
        echo "Sin el Installer no se pueden instalar las demás
        aplicaciones. Saliendo de la aplicación.."
        exit 0
    fi
    break;;
    * ) echo "Inválido. Vuelva a intentarlo."
    read yn_inst;;
    esac
done


# ** INICIO INST **

if [[ $wnt_inst == 1 ]]; then

# -------- Buscar en la lista las versiones del Installer --------

ver_installer=($(awk 'match($0, /installer/) {print $1}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt | sort -r))

# --------- Desplegar versiones y dar a escoger -------------------
contador=0
display_list "${ver_installer[@]}"

echo "Seleccione la versión que desea instalar. Cualquier otra tecla
para salir."
read num_ver

# -------- Entradas no deseadas (Protección)------------------

verif_input $num_ver
verif_input2 $num_ver $contador

# ------- Creación de Directorio de Descarga -----------
```

```
echo "Ha escogido> ${ver_installer[$num_ver]}"

rm -rf $SRC_DIR/${ver_installer[$num_ver]}
mkdir $SRC_DIR/${ver_installer[$num_ver]}

# -------- Obtencion de nombre de directorio padre -----
dir_inst=($(awk 'match($0, '/${ver_installer[$num_ver]}/') {print $2}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt))

# -------------- Descarga -----------
echo "Conectando al servidor de Synopsys ..."
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexión exitosa"
cd site31759/MyProducts
cd $dir_inst
cd ${ver_installer[$num_ver]}
mget * -O "$SRC_DIR/${ver_installer[$num_ver]}"
bye
EOF

# ----------- Verificar Checksum ---------------
echo "Descarga completa. Verificando checksum.."
chcksm ${ver_installer[$num_ver]}

# ------------ Instalación --------------
echo "Iniciando proceso de instalación."
pwd
chmod 755 *.run
./*.run << EOF
/usr/synopsys/installer
EOF

echo "Actualizando variables de entorno.."
if ! grep -q PATH=$TRGT_DIR/installer ~/.bashrctest; then
echo 'PATH='$TRGT_DIR'/installer:$PATH' >> ~/.bashrctest
fi

# ** FIN INST **************************
fi

# +++++++++++++++++++ ALGORITMO APPS ++++++++++++++++++++
# ** INICIO APPS ******
while [ 1 ]; do
# ---- Desplegar Versiones ---------
display_list "${apps_formal[@]}"

echo "Que aplicación desea descargar e instalar? Cualquier otra
```

```
tecla para salir."
read numapp

# ----- Entradas no deseadas (Protección) ---------

verif_input $numapp
verif_input2 $numapp $contador

# ------ Obtener y desplegar las versiones ----------

# Algoritmo para VCS y DVE
if (($numapp==0)); then
echo ${apps_real[$numapp]}
read -a var_sep <<< "${apps_real[$numapp]}"
cd $ACT_DIR
ver_app=($(awk 'match($0, '/${var_sep[0]}/') {print $1}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt
| sort -r | head -6))
ver_app2=($(awk 'match($0, '/${var_sep[1]}/') {print $1}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt
| sort -r | head -6))
echo $ver_app
echo $ver_app2

contador=0
for x in "${ver_app[@]}"; do
echo "$contador $x ---- ${ver_app2[$contador]}"
let contador++
done

# Algoritmo para las demás Apps
else
ver_app=($(awk 'match($0, '/${apps_real[$numapp]}/') {print $1}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt
| sort -r | head -6))
echo $ver_app
display_list "${ver_app[@]}"
fi

echo "Seleccione la versión que desea instalar. Cualquier otra tecla
para salir."
read num_ver

# ------------- Entradas no deseadas (Protección) ----------

verif_input $num_ver
verif_input2 $num_ver $contador
```

```
# -------------- Crear Directorios si no existen -------------------------
var_empty ${ver_app[$num_ver]}
rm -rf $SRC_DIR/${ver_app[$num_ver]}
mkdir $SRC_DIR/${ver_app[$num_ver]}

# Algoritmo para VCS y DVE

if (($numapp==0)); then
    var_empty ${ver_app2[$num_ver]}
    rm -rf $SRC_DIR/${ver_app2[$num_ver]}
    mkdir $SRC_DIR/${ver_app2[$num_ver]}
fi

dir_inst=($(awk 'match($0, '/${ver_app[$num_ver]}/') {print $2}'
myproductsrev.txt myproductsrevold.txt myproductsauth.txt))

# ----------------- Descarga --------------------------------------

if (($numapp==0)); then
echo "Conectando al servidor de Synopsys ..."
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexion exitosa"
cd site31759/MyProducts/$dir_inst
cd ${ver_app[$num_ver]}
mget * -O "$SRC_DIR/${ver_app[$num_ver]}"
cd ..
cd ${ver_app2[$num_ver]}
mget * -O "$SRC_DIR/${ver_app2[$num_ver]}"
bye
EOF
# Para VCS y DVE
else
echo "Conectando al servidor de Synopsys ..."
lftp sftp://$SFTP_USER:$SFTP_PSW@$SFTP_SERVER << EOF
echo "Conexion exitosa"
cd site31759/MyProducts/$dir_inst
cd ${ver_app[$num_ver]}
ls
mget * -O "$SRC_DIR/${ver_app[$num_ver]}"
bye
EOF
fi


# -------------- Checksum -----------------------
echo "Descarga completa. Verificando checksum.."

# Para VCS y DVE
```

```
if (($numapp==0)); then
chcksm ${ver_app2[$num_ver]}
fi

# Para las demás apps
chcksm ${ver_app[$num_ver]}

# -------------- Instalación -------------------
# Separar nombre y versión del common
sep_cmn ${ver_app[$num_ver]}

# Borrar carpetas de instalacion viejas
var_empty ${ver_app[$num_ver]}
var_empty $name_app
var_empty $version

if (($numapp==0)); then
var_empty ${ver_app2[$num_ver]}
fi

instalacion $name_app $version

echo "Actualizando las variables de entorno de ${ver_app[$num_ver]}..."
sed -i '/\/'$name_app'\//d' ~/.bashrc

# Variables de entorno para HSPICE
if (($numapp==21)); then
echo 'PATH='$TRGT_DIR'/'$name_app'/'$version'/'$name_app'/bin:$PATH'
>> ~/.bashrc

# Variables de entorno para CSCOPE
elif (($numapp==5)); then
echo 'PATH='$TRGT_DIR'/'$name_app'/'$version'/ai_bin:$PATH' >> ~/.bashrc
else
# Variables de Entorno para las demás Apps
echo 'PATH='$TRGT_DIR'/'$name_app'/'$version'/bin:$PATH' >> ~/.bashrc
fi

# Instalación y variables de entorno extra para DVE
if (($numapp==0)); then

echo "Instalando ${ver_app2[$num_ver]}"
installer -source "$SRC_DIR/${ver_app2[$num_ver]}" -target $TRGT_DIR
echo "Actualizando las variables de entorno de ${ver_app2[$num_ver]}..."

sed -i '/VCS_HOME/d' ~/.bashrc
echo 'export VCS_HOME='$TRGT_DIR'/'$name_app'/'$version'' >> ~/.bashrc
```

```
fi

# Instalación y variables de entorno extra para IC Validator
if (($numapp==4)); then
sed -i '/ICV_HOME_DIR/d' ~/.bashrc
echo 'export ICV_HOME_DIR='$TRGT_DIR'/'$name_app'/'$version'' >> ~/.bashrc

fi

sed -i '/export PATH/d' ~/.bashrc
echo 'export PATH' >> ~/.bashrc

done
```

13.3. Archivo bashrc

```
    # .bashrc

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

LM_LICENSE_FILE=27020@192.168.6.252; export LM_LICENSE_FILE



# Uncomment the following line if you don't like systemctl's
auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
PATH=/usr/synopsys/fm/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/icc2/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/installer:$PATH
PATH=/usr/synopsys/wv/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/syn/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/newInst/icvalidator/T-2022.03-SP3/bin:$PATH
export ICV_HOME_DIR=/usr/synopsys/newInst/icvalidator/T-2022.03-SP3
PATH=/usr/synopsys/newInst/cscope64/P-2019.06/ai_bin:$PATH
PATH=/usr/synopsys/newInst/customcompiler/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/newInst/icc/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/newInst/starrc/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/newInst/hspice/T-2022.06-SP1/hspice/bin:$PATH
PATH=/usr/synopsys/newInst/primewave/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/newInst/nt/T-2022.03-SP4/bin:$PATH
```

```
PATH=/usr/synopsys/newInst/prime/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/newInst/vcs/T-2022.06-SP1/bin:$PATH
export VCS_HOME=/usr/synopsys/newInst/vcs/T-2022.06-SP1
PATH=/usr/synopsys/newInst/verdi/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/newInst/xa/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/newInst/primesim/T-2022.06-SP1/bin:$PATH
PATH=/usr/synopsys/newInst/txs/T-2022.03-SP4/bin:$PATH
PATH=/usr/synopsys/newInst/vtran/T-2022.03-SP4/bin:$PATH
export PATH
```

CAPÍTULO 14

Glosario

IMEC Por sus siglas en inglés: Interuniversity Microelectronics Centre. Es un centro de desarrollo e investigación que se enfoca en temas de nanoelectrónica y soluciones digitales. 5

iPDK Por sus siglas en inglés: Interoperable Process Design Kit. Es un conjunto de archivos proveídos por Synopsys para facilitar el uso de las aplicaciones. Puede incluir archivos de verificación, modelos de HSPICE, archivos de extracción de parásitos, etc. 4

VLSI Por sus siglas en inglés: Very Large-Scale Integration. Proceso de incorporación de millones de transistores en un solo chip. 1