Universidad Pontificia Comillas

Escuela Técnica Superior de Ingeniería ICAI

Proyecto de Fin de Grado

Emulación de VHDL mediante microsimulaciones

Autor:
Juan Pérez Vilanova

Supervisores:
Fermín Zabalegui Sanz y José Daniel Muñoz Frías

Julio, 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
'Emulación de VHDL mediante microsimulaciones'
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020/21 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.
El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Autor del Proyecto

Juan Pérez Vilanova

Fdo.: ..... Fecha: 21 / 07 / 2021

Director del Proyecto

Fermín Zabalegui Sanz

ZABALEGUI SANZ FERMIN - 50202045Z
Firmado digitalmente por ZABALEGUI SANZ FERMIN - 50202045Z
Fecha: 2021.07.21 13:26:54 +02'00'

Fdo.: ..... Fecha: ..... / ..... / .....

Codirector del Proyecto

José Daniel Muñoz Frías

Firmado por MUÑOZ FRIAS JOSE DANIEL - 52573841G el día 21/07/2021 con un certificado

Fdo.: Fecha: ..... / ..... / .....

A Fermín, por ofrecerme
este proyecto y confiar en mí.
Y a mi familia y amigos, gracias por todo.

'Logic is the foundation of the
certainty of all the knowledge we acquire.'
Leonhard Euler

'If people do not believe that mathematics is simple,
it is only because they do not realize how complicated life is.'
John von Neumann

Resumen

Emulación de VHDL mediante microsimulaciones
Autor: *Pérez Vilanova, Juan*
Director: *Zabalegui Sanz, Fermín*
Codirector: *Muñoz Frías, José Daniel*
Entidad Colaboradora: *ICAI – Universidad Pontificia Comillas*

1. Introducción

Actualmente, existen diversos campos de desarrollo dentro del ámbito tecnológico: robótica, inteligencia artificial, redes neuronales, tratamiento de datos... Todas ramas de investigación operativas, marcando el avance de las nuevas generaciones. En un entorno globalizado, es natural una transición entre nuevas implantaciones más sostenibles cuyo producto da como resultado una transformación de la tecnología.

Dentro del sector de la electrónica digital, con el constante avance de los dispositivos, la renovación es la alternativa al reciclaje de dichas piezas caídas en un mayor desuso. Este es el caso de las FPGA, abreviatura de «*Field Programmable Gate Array*», herramientas de uso comercial focalizadas al desarrollo industrial, cuyo interés se ha mantenido reducido por un periodo prolongado por décadas debido a diversos factores.

Con este proyecto, se busca abordar dicha problemática, mediante la creación de un software de emulación ejecutado para lenguaje VHDL sobre Python por medio de micro simulaciones.

2. Definición del proyecto

El objetivo de este documento consiste en la recopilación de información y desarrollo de herramientas disponibles hacia los usuarios recién introducidos al software basado en lenguaje VHDL [acrónimo combinación de dos términos: VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language)] [1], utilizado para la descripción de circuitos digitales y la automatización de diseño electrónico.

Figura 1: Licencia Creative Commons

3. Descripción de la herramienta

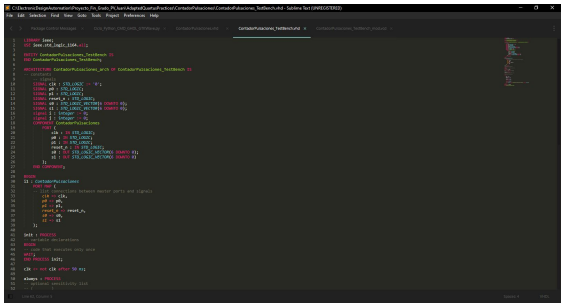
El método de diseño empleado para la creación del programa desarrollado está basado en la lectura y escritura de ficheros *.vcd* y *.vhd* respectivamente. Para lograrlo, serán necesarios el uso de determinadas herramientas *'open-source'*, tales como:

- **GHDL**, una herramienta de uso libre destinada al análisis, compilación, simulación y síntesis (de forma experimental) de lenguaje VHDL; sin embargo, no es un intérprete. Su función principal consiste en la elaboración y análisis de fuentes para generar código de máquina a partir de diseño, únicamente permitiendo la ejecución de simulación en alta velocidad por medio de programas nativos. [2]
- **GTKWave**, un visualizador de ondas disponible para los sistemas operativos basados en Unix, Win32, y Mac OSX, cuyos formatos de archivo permite archivos LXT, LXT2, VZT, FST, y GHW, al igual que archivos Verilog VCD/EVCD
- **Sublime Text**, un editor de texto sofisticado para código, margen y prosa que permite la escritura en los principales lenguajes de programación y especificación: Python, JavaScript, Java, C#, C, C++, Go, R, PHP,... Además que permite la inclusión de numerosos lenguajes, junto a un ambiente de personalización completa al alcance de cada usuario.

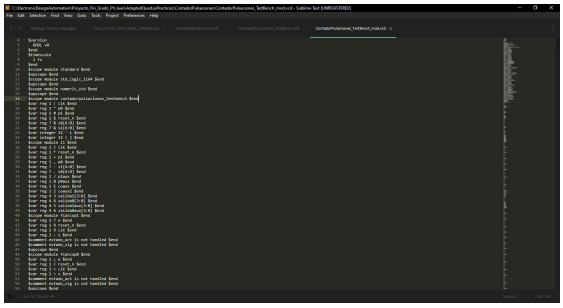
Una vez se dispone del software necesario, la programación recoge el testigo y toma el liderazgo, adaptando la estructuración del proyecto:

- Creación en lenguaje VHDL de 'test script': escritura de diversos programas (según la dependencia del factor tiempo, asíncronos o síncronos)
- Creación en lenguaje VHDL de un 'testbench' o banco de pruebas blanco: para cada script, se relacionará con un marco de trabajo vacío donde el usuario introducirá los valores de las entradas deseados
- Creación en Python de un 'script' canal de comunicación con la CMD [GHDL y GTKWave]: el núcleo del proyecto, un programa capaz de manipular la consola de comandos de forma simultanea con los ficheros *.vhd* y *.vcd* deseados

En términos generales, utilizando Python como punto común, GHDL lee el archivo *.vhd* a emular, simulando el resultado de las señales de entrada captadas por el programa, provocadas por el usuario, y genera un archivo *.vcd* donde se descargan los resultados obtenidos; donde por medio del mismo programa o GTKWave, se pueden interpretar los resultados.



(a) Banco de pruebas - Archivo *.vhd*

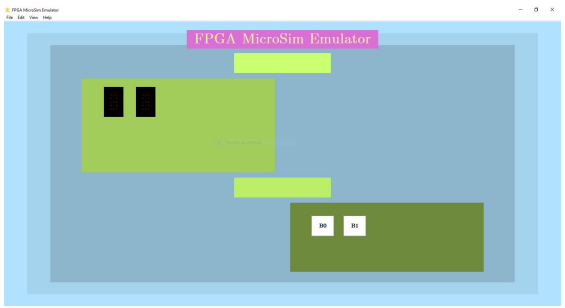


(b) Banco de pruebas - Archivo *.vcd*

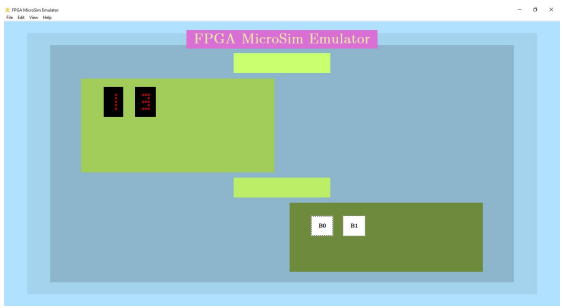
Figura 2: Capturas de pantalla del banco de pruebas 'ContadorPulsaciones'

4. Resultados

Una vez se han creado los diferentes programas y finalizado el proyecto (todos los códigos detallados en el Capítulo 8) se muestran los resultados de 'ContadorPulsaciones'. Se ejecuta una prueba donde el pulsador B1, se pulsará 13 veces; y tras pulsar B0, se mostrará el resultado del registro de pulsaciones en los 7 segmentos representados encima (Figura 3):



(a) Situación inicial de 'ContadorPulsaciones'



(b) Situación 13 pulsos 'ContadorPulsaciones'

Figura 3: Diseño del programa 'FPGA MicroSim Emulator'

Evaluando los resultados obtenidos, el programa creado en Python responde de manera adecuada a los estimulos previstos, en comunión con el simulador GHDL y el visualizador GTKWave cumplen las expectativas previstas.

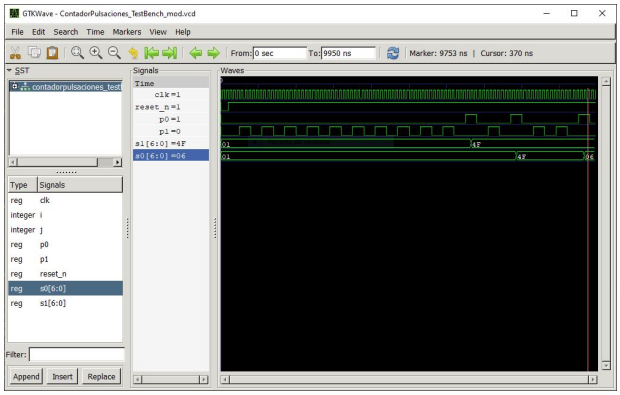


Figura 4: Visualización GTKWave - 13 pulsos 'ContadorPulsaciones

Como se observa en la figura 4, el pulsador 'p1' asociado a 'B1' ha sido pulsado un total de 13 veces; mientras que el pulsador 'p0' ha sido pulsado de manera intercalada tras 10, 11 y 13 pulsos. Dichos cambios aparecen reflejados en los 'displays' 's1' y 's0', obteniendo un resultado favorable.

5. **Conclusiones**

   Gracias al programa desarrollado, a través de comunicación directa con la consola de comandos, se obtiene un emulador orientado a la ejecución de aplicaciones programadas en VHDL.

   Se logra simplificar en fracciones de segundos simular la situación actual de la FPGA virtual, transferir los resultados a la estructura del archivo *.vcd* generado y leer los resultados para mostrarlos por pantalla. Igualmente, se ha considerado la posibilidad de modificar el tipo de fichero para ahorro de tiempo, donde la máxima sería la reducción de los tiempos de carga, aunque con las centésimas de segundos son adecuadas para los resultados esperados.

Referencias

[1] Intel Corporation. *VHDL Definition*. 2017. URL: https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_vhdl.htm#:~:text=The%20Very%20High%20Speed%20Integrated,is%20defined%20by%20IEEE%20standards. (visitado 21-07-2021).

[2] Tristan Gingold. *GHDL*. 2017. URL: http://ghdl.free.fr/ (visitado 21-07-2021).

Abstract

VHDL emulation via microsimulations
Author: *Pérez Vilanova, Juan*
Supervisor: *Zabalegui Sanz, Fermín*
CoSupervisor: *Muñoz Frías, José Daniel*
Collaborating Entity: *ICAI – Universidad Pontificia Comillas*

1. **Introduction**

   Currently, there are several fields of development within the technological field: robotics, artificial intelligence, neural networks, data processing... All branches of operational research, marking the advance of the new generations. In a globalized environment, it is natural to transition between new, more sustainable deployments whose output results in a transformation of technology.

   Within the digital electronics sector, with the constant advancement of devices, renewal is the alternative to recycling such fallen parts into further disuse. This is the case of FPGA, short for ¡¡*Field Programmable Gate Array¿¿*, tools for commercial use focused on industrial development, whose interest has been reduced for a period of decades due to various factors.

   This project seeks to address this problem, by creating an emulation software executed for VHDL language on Python by means of micro simulations.

2. **Project definition**

   The objective of this document is to gather information and develop tools available to newly introduced users of VHDL-based software [acronym combination of two terms: VHSIC (Very High Speed Integrated Circuit) and HDL (Hardware Description Language)] [1], used for digital circuit description and electronic design automation.

Figure 5: Creative Commons License

3. Tool Description

The design method used for the creation of the developed program is based on the reading and writing of *.vcd* and *.vhd* respectively. To achieve this, the use of certain *'open-source'* tools, such as:

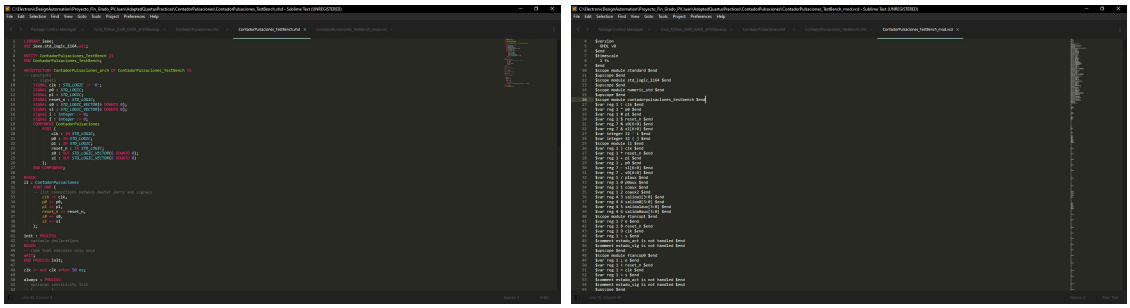
- **GHDL**, a free-use tool for the analysis, compilation, simulation and synthesis (experimentally) of VHDL language; however, it is not an interpreter. Its main function is the elaboration and analysis of sources to generate machine code from design, only allowing the execution of high-speed simulation through native programs. [2]
- **GTKWave**, a wave viewer available for Unix, Win32, and Mac OSX-based operating systems, whose file formats allow LXT, LXT2, VZT, FST, and GHW files, as well as Verilog VCD/EVCD files
- **Sublime Text**, a sophisticated text editor for code, margin and prose that allows writing in the main programming and specification languages: Python, JavaScript, Java, C#, C, C++, Go, R, PHP,... It also allows the inclusion of numerous languages, along with a complete customization environment at the reach of each user.

Once the necessary software is available, the programming picks up the baton and takes the lead, adapting the structure of the project:

- Creation in VHDL language of 'test script': writing of different programs (depending on the time factor, asynchronous or synchronous)
- VHDL language creation of a white testbench: for each script, it will relate to an empty framework where the user will enter the values of the desired entries
- Python creation of a 'script' communication channel with the CMD [GHDL and GTKWave]: the kernel of the project, a program capable of manipulating the command console simultaneously with the *.vhd* and *.vcd* desired

Generally speaking, using Python as a common point, GHDL reads the *.vhd* file to emulate, simulating the result of the input signals captured by the

program, caused by the user, and generates a *.vcd* file where the obtained results are downloaded; where by means of the same program or GTKWave, the results can be interpreted.

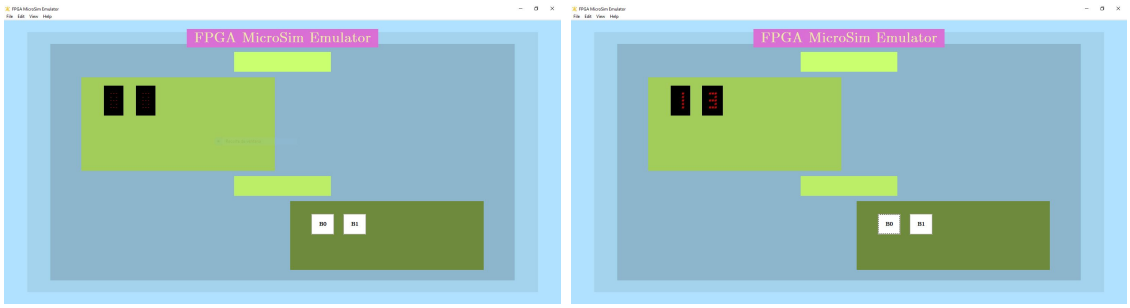


(a) Test bench - *.vhd* file (b) Test bench - *.vcd* file

Figure 6: Screenshots of the test bench 'ContadorPulsations'

4. Results

Once the different programs have been created and the project is completed (all the codes detailed in the chapter 8) the results of 'ContadorPulsations' are shown. A test is executed where the B1 button is pressed 13 times; and after pressing B0, the result of the recording of pulses in the 7 segments represented above will be displayed (Figure 7):



(a) Initial status of 'ContadorPulsations' (b) Situation 13 pulses 'ControlPulsations'

Figure 7: Program design 'FPGA MicroSim Emulator'

Evaluating the results obtained, the program created in Python responds adequately to the expected stimuli, in communion with the simulator GHDL and the viewer GTKWave meet the expected expectations.

As can be seen in figure 8, the 'p1' button associated with 'B1' has been pressed a total of 13 times, while the 'p0' button has been pressed inter-

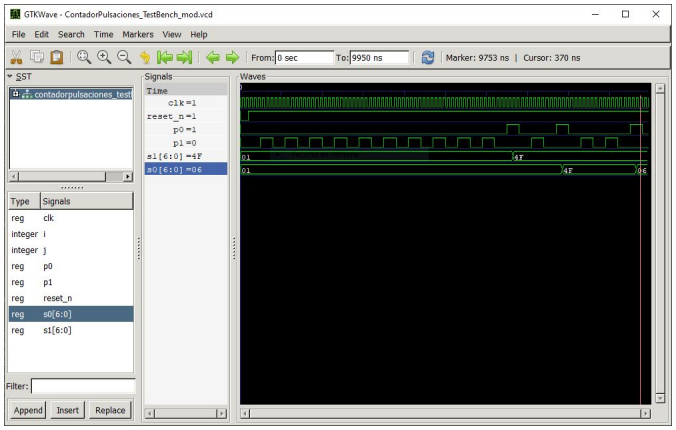


Figure 8: Display GTKWave - 13 pulses 'ControlPulsations'

changeably after 10, 11 and 13 pulses. These changes are reflected in the 'displays' 7 segments 's1' and 's0', obtaining a favorable result.

5. **Conclusions**

   Thanks to the developed program, through direct communication with the command console, you get an emulator oriented to the execution of applications programmed in VHDL.

   It is possible to simplify in fractions of seconds simulate the current situation of the virtual FPGA, transfer the results to the structure of the *.vcd* file generated and read the results to display them on screen. Likewise, the possibility of modifying the file type for saving time has been considered, where the maximum would be the reduction of loading times, although with the hundredths of seconds are suitable for the expected results.

References

[1] Intel Corporation. *VHDL Definition*. 2017. URL: `https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_vhdl.htm#:~:text=The%20Very%20High%20Speed%20Integrated,is%20defined%20by%20IEEE%20standards.` (visited on 07/21/2021).

[2] Tristan Gingold. *GHDL*. 2017. URL: `http://ghdl.free.fr/` (visited on 07/21/2021).

Índice general

1. Introducción **1**
1.1. Historia del lenguaje VHDL 1
1.2. Desarrollo de FPGAs 3
1.3. Motivación del proyecto 6

2. Descripción de las tecnologías **7**
2.1. GHDL 7
2.1.1. Instalación del software 7
2.1.2. Utilización del software 9
2.2. GTKWave 11
2.2.1. Instalación del software 11
2.2.2. Utilización del software 15
2.3. Sublime Text 17
2.3.1. Instalación del software 17
2.3.2. Utilización del software 19
2.4. Python 21
2.4.1. Instalación del software 21
2.4.2. Utilización del software 22

3. Estado de la cuestión **23**
3.1. Sector industrial 23
3.2. Sector académico 24

4. Definición del trabajo **25**
4.1. Justificación 25
4.2. Objetivos 26
4.3. Metodología 26
4.3.1. Metodología STEAM 26
4.3.2. Objetivos de Desarrollo Sostenible 27
4.4. Planificación y estimación económica 28

5. Modelo desarrollado **29**
5.1. Análisis del lenguaje 29
5.1.1. VHDL: archivos .vhd y .vcd 29
5.1.2. Python: Comunicación CMD - GHDL/GTKWave 32
5.2. Diseño 33
5.2.1. Elección de Inputs/Outputs 33
5.2.2. Asociación de variables con periféricos 34
5.2.3. Emulación dinámica de FPGA 35
5.3. Implementación 36

6. Análisis de resultados **37**

7. Conclusiones y trabajos futuros **39**
7.1. Conclusiones del proyecto 39
7.2. Implementaciones futuras 40

8. Anexo: Código del programa **41**

Bibliografía **93**

Índice de figuras

1. Licencia Creative Commons viii
2. Capturas de pantalla del banco de pruebas 'ContadorPulsaciones' . ix
3. Diseño del programa 'FPGA MicroSim Emulator' ix
4. Visualización GTKWave - 13 pulsos 'ContadorPulsaciones x
5. Creative Commons License xii
6. Screenshots of the test bench 'ContadorPulsations' xiii
7. Program design 'FPGA MicroSim Emulator' xiii
8. Display GTKWave - 13 pulses 'ControlPulsations' xiv

1.1. Norma IEEE 1076-1987 1
1.2. Diagrama de Gajski-Kuhn 3
1.3. Xilinx XC2064: Primer modelo FPGA comercial 3
1.4. Topología FPGA 4
1.5. Vendedores FPGA 4
1.6. IceZero Lattice iCE40 5
1.7. Alhambra y Icestudio: FPGA de código abierto 6

2.1. Página de inicio GHDL 8
2.2. Versión v0.37 GHDL 9
2.3. Comandos GHDL en CMD 11
2.4. Página de inicio GTKWave 11
2.5. Página de descarga GTKWave - Versión v3.3.100 destinada a Windows 12
2.6. Configuración de Windows 13
2.7. Propiedades del sistema - Opciones avanzadas - Variables del entorno 14
2.8. Edición de variables del entorno 14
2.9. Ventana principal de GTKWave 16
2.10. Página de descarga Sublime Text 3 17
2.11. Sublime Text 3 18
2.12. Página de descarga Python 21
2.13. Logo Python 22

3.1. Proyecto en ActiveHDL 24

3.2. Simulación en ActiveHDL 24

5.1. Estructura VHDL 29
5.2. Flujo de trabajo de una FPGA 30
5.3. Ciclo de comunicación Python con CMD - GHDL/GTKWave 32
5.4. Diagrama GUI 33
5.5. Ventana de elección - Entradas y salidas - FPGA Emulator 33
5.6. Ventana de asociación - Nombres de variables - FPGA Emulator . . 35
5.7. Ventana de emulación - Situación inicial - FPGA Emulator 35

6.1. Ventana de elección - 'ContadorPulsaciones.vhd' 37
6.2. Ventana de asociación - FPGA Emulator 37
6.3. Ventana de emulación - 'ContadorPulsaciones.vhd' 38
6.4. GTKWave - 'ContadorPulsaciones.vhd' 38

Listado de códigos

2.1. ha.vhd 19
2.2. ha_tb.vhd 19
2.3. Librerías del proyecto 22
5.1. Librerías y paquetes estándar VHDL 31
8.1. ha_tb.vcd 41
8.2. ContadorPulsaciones.vhd 43
8.3. BinA7Seg.vhd 46
8.4. ContadorAscMod10.vhd 47
8.5. DetectorFlancoSubida.vhd 48
8.6. Despliegue.vhd 50
8.7. ContadorPulsaciones_TestBench.vhd 52
8.8. python_cmd.py 54
8.9. def CMD_Python_Variable_Selection() 57
8.10. def CMD_Python_Button_Execution_B(enter_value) 60
8.11. Proyecto_Fin_Grado_PV Juan.py 63

Capítulo 1

Introducción

En este primer capítulo, se dará un breve repaso por el pasado de esta tecnología, sus orígenes y su influencia posterior, unido a su evolución durante estos años. De igual forma, se abordará cual es la principal motivación del proyecto, derivado de su situación contemporánea.

1.1. Historia del lenguaje VHDL

Año 1983. El inicio del uso de gráficos por computadora en la televisión, el lanzamiento de Windows 1.0 por parte de Microsoft, el descubrimiento de los bosones W y Z en la Organización Europea para la Investigación Nuclear o CERN, y el lanzamiento de Nintendo de Mario Bros. Un año con grandes avances tecnológicos, entre los que se incluye un movimiento poco reconocido, pero clave para el desarrollo posterior de la electrónica digital.

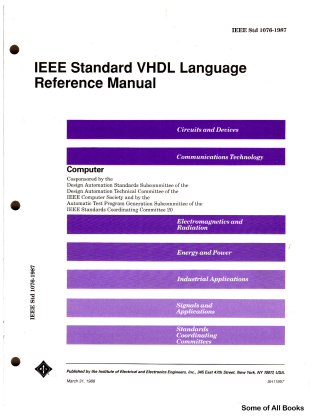


Figura 1.1: Norma IEEE 1076-1987

Este mismo año, el gobierno de los Estados Unidos de América cede a las empresas *Intermetrics* (en la actualidad conocida como AverStar, fundada en 1969 por veteranos del MIT pertenecientes al Programa Apollo de la NASA), *IBM* y *Texas Electronics* la concesión de derechos sobre el desarrollo de VHDL, un lenguaje de descripción de hardware (de sus siglas en inglés HDL) creado por el Departamento de Defensa de EEUU (DoD) entre los años 1980 y 1981 como una ramificación del proyecto DARPA.

Concretamente, VHDL es un acrónimo de VHSIC-HDL, siendo VHSIC (Very High Speed Integrated Circuits) uno de los proyectos DARPA destinados a uso militar en plena '*Segunda Guerra Fría*'. El desarrollo de este lenguaje, tanto en términos sintácticos como conceptuales, proviene en parte considerable de Ada: un lenguaje de programación diseñado por Jean Ichbiah en 1980 como un intento de unificación de la innovación tecnológica de la época.

Aunque en agosto de 1985, la versión 7.2 accedió a dominio público, no fue hasta **diciembre de 1987** cuando el *Instituto de Ingenieros Eléctricos y Electrónicos*, conocido por sus siglas inglesas IEEE, ratificó VHDL como su estándar con *norma 1076-1987*. Sería un año más tarde cuando ANSI, el Instituto de Estándares Nacionales Americanos, recogería el testigo, formalizando dicho conocimiento en la comunidad de la Electrónica Digital. [1]

Para el año **1993**, IEEE revisó su normativa para la inclusión de nuevos estándares, entre los que se destaca al ISO-8859-1, con una impresión de caracteres más consistente y una sintaxis más flexible. Las posteriores actualizaciones en los años 2000 y 2002 trajeron cambios menores para la mejora de las funcionalidades de VHDL y los dispositivos que utilizaban este lenguaje. Con el año 2006, la versión VHDL Draft 3.0 consiguió elaborar un sistema capaz de mantener la completa compatibilidad entre las funcionalidades de aparatos longevos con los nuevos sistemas mediante la introducción de numerosas extensiones, facilitando el manejo de código y potenciando su respectiva síntesis. [2]

En **2008** y con la versión 4.0 Draft, IEEE realizó una incorporación masiva de cambios dentro del estándar principal 1076, siendo la última modificación significativa con un único cambio posterior a la norma en diciembre de 2019.

Después de este paso por la historia, es natural cuestionarse que es un lenguaje de descripción de hardware, sus dispositivos de aplicación, y el atractivo generado en el ámbito académico, doméstico y/o industrial.

Como indica su nombre, un **HDL** describe mediante su lenguaje: el modelo, la estructura, el diseño y las operaciones de circuitos electrónicos. Previa su inclusión, los integrados se diseñaban y desarrollaban de forma manual, y con el aumento de circuitos más complejos, su *coste de reposición* aumentaba debido a la falta de documentación. Con el uso de un lenguaje común estandarizado, el tiempo para diseñar nuevos circuitos se redujo a una descripción de la utilidad del mismo, y con ello, se consolidaban las primera herramientas en el campo de la automatización del diseño electrónico (EDA en inglés).

Con el nacimiento de estas herramientas, la comunidad pudo enfocar los proyectos, su concepción, y producción desde el diseño digital abarcando desde el proyecto de circuitos integrados hasta el desarrollo de placas impresas.

Estos lenguajes de descripción se fragmentaron en varias vertientes dependientes del tipo de tecnología hardware a diseñar. Entre sus ramas principales, se

encuentran: **Verilog** para los circuitos integrados para aplicaciones específicas, más conocidos como *ASICs*, y **VHDL**, el lenguaje utilizado para este proyecto, focalizado en las matrices de puertas lógicas programables en campo, de sus siglas *FPGAs*.

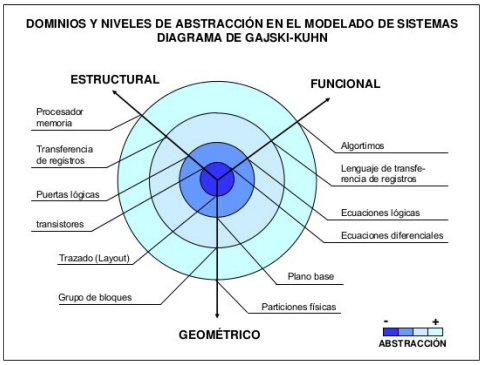


Figura 1.2: Diagrama de Gajski-Kuhn

Fijándose en la figura 1.2, se puede observar como el lenguaje VHDL abarca desde la *circuitería lógica* (puertas y flip-flops con ecuaciones booleanas en tiempos de conmutación continuos) hasta el *chip del procesador* (subsistemas con algoritmos), pasando por la *transferencia entre registros* (ALU, regs., multiplexores...) [3]. Esta abanico de posibilidades, se traduce en la práctica a simulaciones de proyectos para comprobar su correcta funcionalidad como la sintetización para crear un sistema equivalente al modelo descrito.

1.2. Desarrollo de FPGAs

En un breve resumen del apartado previo, VHDL en un inicio sirvió como registro del comportamiento y las especificaciones de los dispositivos ASICs de la época, evolucionando en simuladores de dicho dispositivos por medio de archivos VHDL; culminando en la síntesis de las descripciones generadas y la obtención de una salida apta para su implantación en ASIC, FPGA y CPLD. [4]

Figura 1.3: Xilinx XC2064: Primer modelo FPGA comercial

Una **matriz de puertas lógicas programable en campo** consiste en un circuito lógico cuya funcionalidad puede ser descrita por el usuario. Esta característica tiene su origen en 1978 con la *Matriz Lógica Programable* (PAL), seguido de los *Dispositivos de Lógica Programable Compleja* (CPLD), una colección de PALs interconectados mediante una matriz de conectores programables.

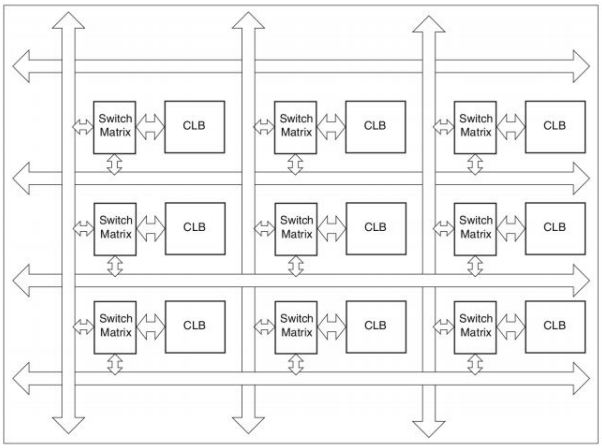


Figura 1.4: Topología FPGA

La topología de una FPGA hace uso de bloques lógicos elementales interconectados, que suelen incluir algunos con funcionalidad fija como multiplexores, funciones lógicas AND, OR, NOT y XOR; o memoria (flip-flop tipo D).

Su memoria puede ser *volátil* (basada en RAM; tras dejar de alimentar el dispositivo, su estructura volverá a la situación predeterminada, sin guardar ningún registro previo de organización) o *no volátil* (basada en ROM; pueden ser reprogramables, o no reprogramables).

Las ventajas de la lógica programable frente a la discreta son varias: la flexibilidad de los elementos reconfigurables, la reducción de espacio en la placa impresa, la rapidez del circuito y la reducción de costes energéticos y de inversión.

Entre los **campos de aplicación** pertenecientes a este dispositivo, destacan: comunicaciones radio definida por software (SDR) [42 %], prototipos de ASICs, aplicaciones bio-informáticas y emulación de hardware de ordenador [18 %], Data Center, criptografía, sistemas de imágenes médicas [18 %] procesamiento digital de señales, reconocimiento de voz [13 %], y diversos sistemas (automovilísticos, aeroespaciales, defensa/militar) [9 %]... [5]

En el mercado actual, existen dos compañías encargadas de la creación, distribución y venta mayoritaria de esta tecnología hardware: *Xilinx* y *Altera*. Ambas son propiedad de dos compañías reconocidas como son **Intel**, la mayor empresa de circuitos integrados a nivel mundial, y **AMD**, segundo proovedor de microprocesadores.

Concretamente, en octubre de 2020, AMD anunció el acuerdo para la compra de Xilinx por un precio estimado de 35.000 millones de dólares, y hasta abril de 2021, los accionistas no habían aprobado la compra.

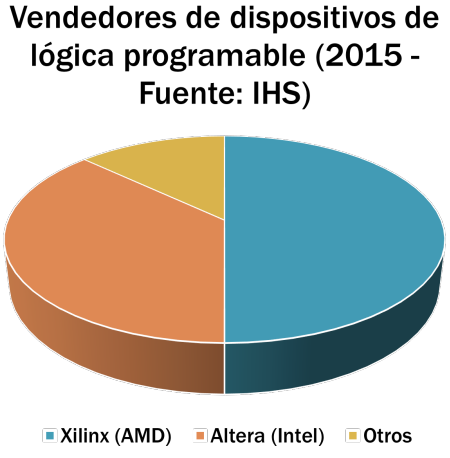


Figura 1.5: Vendedores FPGA

Cada compañía dispone no únicamente de una arquitectura de hardware diferente. También de su propio software orientado hacia uso industrial, como Quartus II con ModelSim para Intel y XSIM/ISE con LogicSim para dispositivos Xilinx.

Como se puede observar en la Figura 1.5, el porcentaje de ventas asociado a fabricantes de CPLDs, FPGAs y ASICs es conformado en un **85 %** por las empresas arriba mencionadas: Xilinx con un 50 % y Altera con un 35 %. El 15 % restante corresponde a empresas como Lattice Semiconductor o QuickLogic, con un abanico mayor de ventajas para poder competir con el duopolio presente, como la especialización en tecnología no volátil Flash o basados en anti-fusibles de programación única.

Un modelo de mercado inalterado por décadas, donde la utilización de las diferentes utilidades de diseño requería de su licencia específica, resultando en dispositivos cerrados e incompatibles. La síntesis, el emplazado y enrutado del proceso de 'bitstream'de la etapa de diseño eran privadas, la configuración de las FPGA era cerrada y las opciones de código abierto eran bastante limitadas. Las novedades vienen acompañadas por proyectos de la comunidad informática para la obtención de hardware y software libre.

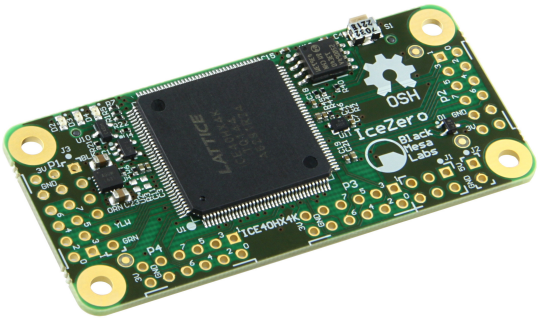


Figura 1.6: IceZero Lattice iCE40

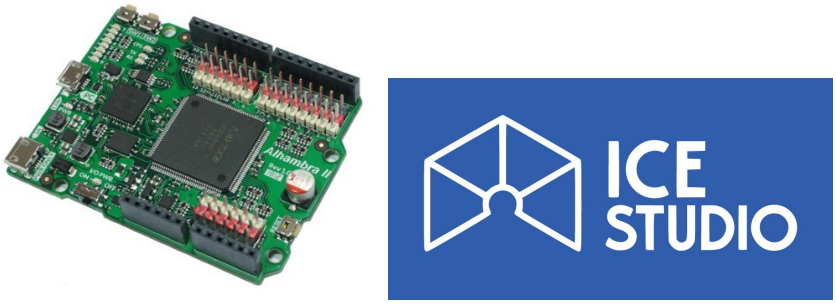
- **Hardware libre**. Entre todos los avances, destaca el proyecto hito subido a la red en marzo de 2015 por Clifford Wolf, con nombre de usuario «Claire Wolf». Tras una investigación de tres años, el *proyecto IceStorm* [6] había documentado el formato del 'bitstream' para una familia de FPGAs (iCE40 de Lattice, figura 1.6), obtenido mediante ingeniería inversa.

  Con este descubrimiento, muchas propuestas de la comunidad informática comenzaron a florecer. Entre ellas, en el campo de las FPGA de hardware libre, se destaca el trabajo de diseño de los españoles Juan González-Gómez y Eladio Delgado, con la comercialización de la placa *Icezum Alhambra*. En la actualidad, el proyecto ha evolucionado a una nueva placa conocida como *Alhambra II V1.0*, con mejoras frente al modelo original. [7]

- **Software libre**. En paralelo al desarrollo de la placa Alhambra, Jesús Arroyo Torrens hace uso del estudio de Wolf, y por medio de la creación de una nueva plataforma, el ecosistema APIO, nace *IceStudio*: un HDL basado en una herramienta visual; utilizada para los recién iniciados en la electrónica

digital. Con el uso de bloques definidos, el usuario puede describir el diseño digital, para posteriormente traducir dichos bloques en lenguaje HDL, que será sintetizado y posteriormente, utilizado en una FPGA de código abierto.

Como curiosidad, estos tres ingenieros forman parte del grupo 'FPGAwars', una comunidad educativa dirigida por ingenieros expertos hacia la promoción del conocimiento de estas herramientas mediante talleres, tutoriales y como se ha mencionado previamente, hardware y software de código abierto. [8]



(a) Alhambra II (b) Icestudio

Figura 1.7: Alhambra y Icestudio: FPGA de código abierto

1.3. Motivación del proyecto

La noticia de la compra de Xilinx es reflejo del impacto actual de esta tecnología, su vigencia y relevancia en el panorama actual tras medio siglo de existencia. El proyecto IceStorm sirvió de inspiración, no solo a Alhambra o Icestudio, sino a la comunidad completa de la electrónica digital, y demostró el apoyo hacia esta rama de la informática.

El desconocimiento por parte del colectivo de esta tecnología, el costo de los dispositivos físicos aún muy elevado, las licencias de programas de utilidad, simulación y síntesis asociadas a un determinado hardware, una escasez (cada vez menor) de herramientas no comerciales...

A nivel industrial, todos los puntos expuestos previamente no suponen un gran impacto, pero en el ámbito académico o doméstico, dichas implicaciones suponen grandes limitaciones en términos generales, y con elementos de código abierto, dicha brecha entre educación y mercado se ve reducida.

Con este proyecto, y alineado con los ideales educativos y divulgativos propios de un entorno académico, se busca la aportación de una nueva herramienta más de código libre para su uso y disfrute, sin necesariamente estar enfocado en una adaptación industrial.

Capítulo 2

Descripción de las tecnologías

En estos apartados posteriores se mostrará la correcta forma de instalación, utilización y actualización de las distintas herramientas de software destinadas al proyecto mencionadas en el resumen.

Todos los elementos presentados en este documento están revisados y actualizados a fecha de julio de 2021. Para cualquier duda, consulte los enlaces facilitados en el documento, al autor y/o supervisores del proyecto.

2.1. GHDL

2.1.1. Instalación del software

Para comenzar con la instalación del compilador, se buscará la página principal del proyecto. Dentro de la misma, se explican las cualidades de GHDL en la actualidad, junto a todas las características necesarias para su correcta utilización. Dichas cumplimentaciones, junto a su funcionamiento, serán posteriormente explicadas en profundidad durante la Subsección 2.1.2

Dentro de la página web descrita en la figura 2.1, se observan varias pestañas:

- Download, enlace directo a los últimos lanzamientos y actualizaciones del programa. A fecha de entrega, este proyecto utiliza la última versión disponible dentro del repositorio para Windows 'ghdl-0.37-mingw32-mcode.zip'
- User Guide, un resumen recopilador con las últimas actualizaciones y noticias agregadas a los ficheros del proyecto
- GitHub, la mayor plataforma web dedicada al alojamiento de código para la colaboración entre archivos. El directorio principal donde se encuentra todo el proyecto subido y en desarrollo, junto a todas las versiones previas y anotaciones de la comunidad

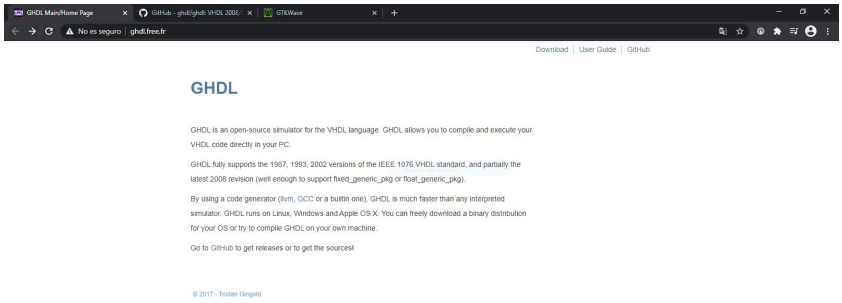


Figura 2.1: Página de inicio GHDL

Para la descarga del programa, se entrará dentro de 'Download' y se buscará el apartado titulado 'v0.37', lanzado a la red por el usuario «tgingold», a fecha de 28 de febrero de 2020.

Dentro del apartado, se descargará la carpeta comprimida nombrada como 'ghdl-0.37-mingw32-mcode.zip'. Dicho archivo ha sido seleccionado para mayor facilidad del usuario promedio, tratándose de archivos pre-compilados binarios orientados a Windows, facilitando dicha tarea.

En caso de una búsqueda para actualizar el software, manualmente se pueden descargar los archivos posteriormente lanzados a la plataforma y modificar los existentes en sus respectivas carpetas, atendiendo a las orientaciones indicadas en dicha versión.

Una vez completada la descarga, se procede a la descompresión del fichero descargado, y se recomienda modificar el directorio de descarga a una carpeta residente. Para el desarrollo de este proyecto, se ha creado una nueva carpeta en la ruta principal «C:\ElectronicDesignAutomation», destinada a albergar todos los ficheros utilizados para el proyecto, y se renombrará la carpeta descomprimida para evitar posibles errores posteriores.

Como se puede observar en la captura de pantalla representada en la figura 2.2, existen diferentes versiones del programa para distintos sistemas operativos (MAC OS, Linux Ubuntu, Fedora...), abreviados como SO.

Para una correcta instalación en diferentes SO mencionados previamente, es recomendable la lectura completa del directorio web facilitado previamente, unido a instalaciones grabadas por otros usuarios de la red, subidas a las diferentes plataformas de visualización de vídeos.

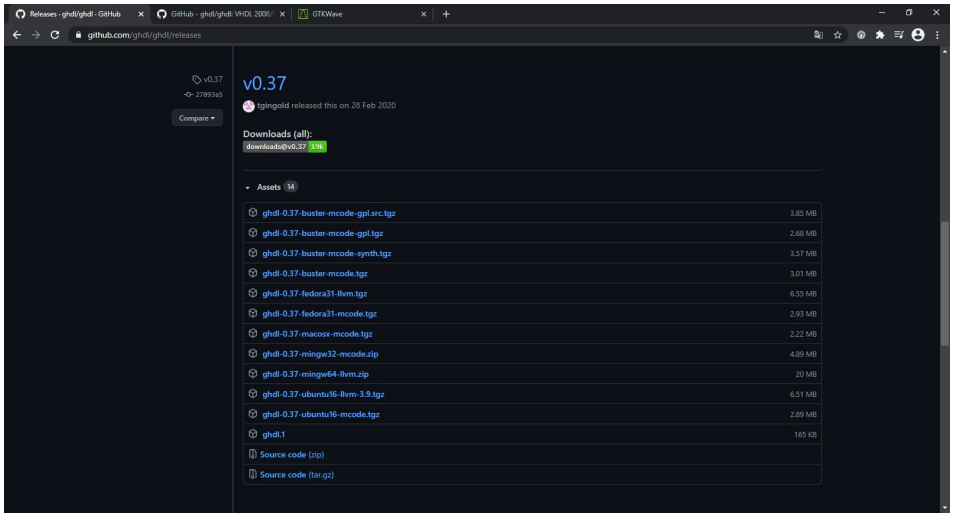


Figura 2.2: Versión v0.37 GHDL

2.1.2. Utilización del software

Atendiendo a la guía provista, GHDL es un simulador que puede ejecutar casi cualquier programa escrito en VHDL. Al contrario que otras herramientas, GHDL no es una herramienta de síntesis, es decir, no crea una descripción de las conexiones de un circuito electrónico. [9]

Por otro lado, GHDL no es únicamente un simulador de circuitería electrónica, sino un compilador con base GNU Compiler Collection (GCC), un compilador de uso libre y gratuito para varios lenguajes distribuido por el proyecto GNU. Y a diferencia de otros programas, GHDL realiza una traducción directa del archivo VHDL a código máquina, sin utilizar un lenguaje intermedio como C o C++.

Esta carencia de una lista de conexiones, unido al añadido de ser un compilador de traducción directa, hace el código procesado más rápido, evitando tiempos de carga intermedios, y será la característica que permita el uso de Python como medio de comunicación con los distintos programas para el desarrollo del proyecto.

Con el archivo VHDL creado en el editor de texto [para mayor detalle, consulte la Subsección 2.3.2], existen tres comandos principales de construcción:

- Comando de análisis: compilación de uno o varios ficheros, con la sucesiva creación de un archivo objeto por cada original:

  ```
  $ ghdl -a [options] files
  ```

- Comando de elaboración: creación de un ejecutable que incluya: el código fuente de los archivos previamente analizados, las instrucciones del procedi-

miento a seguir y el código para ejecutar posteriormente una simulación con su jerarquía pre-diseñada.

Para ello, la unidad primaria debe contener el nombre de una configuración previamente cargada, una entidad que puede venir seguida de una arquitectura, como se indica posteriormente en la línea de comando adjunta.

GHDL re-analiza todas las entradas previamente mencionadas junto a declaraciones de paquetes, y crea una configuración y uniones de indicaciones estandarizadas en base a las reglas del Modelo de Referencia de Librerías:

```
$ ghdl -e [options] primary_unit [secondary_unit]
```

- Comando de ejecución: inicio de la simulación del diseño previo:

```
$ ghdl -r [options] primary_unit ...
...[secondary_unit] [simulation_options]
```

Estos suponen los principales comandos para la interacción con GHDL desde la CMD, aunque existe diversas opciones para modificar parámetros a elección del usuario: cambiar el nombre de la librería lógica de trabajo, el directorio de trabajo, el estándar del lenguaje... Para ver todas las opciones, consultar la sección 'GHDL Options' dentro del manual de usuario.

También existen otros comandos alternativos que surgen como versiones de análisis simple o combinación de los mencionados previamente:

- Comando de elaboración y ejecución:

```
$ ghdl --elab-run [elab_options] primary_unit ...
...[secondary_unit] [run_options]
```

- Comando de comprobación de sintaxis:

```
$ ghdl -s [options] files
```

- Comando de análisis y elaboración:

```
$ ghdl -c [options] file... -r primary_unit [secondary_unit]
```

En materia de advertencias, GHDL contiene una variedad de mensajes emitidos en función de las construcciones analizadas y/o elaboradas. Aunque algunas no sean erróneas, contienen secciones de códigos con opción de interpretación dubitativa, suficiente para lanzar diagnósticos a través de la CMD.

Por último, para proyectos con demasiados archivos, librerías, generar referencias cruzadas, manipulación de ficheros y demás comandos en desuso, consultar el manual de usuario publicado por «Tristan Gingold». [9]

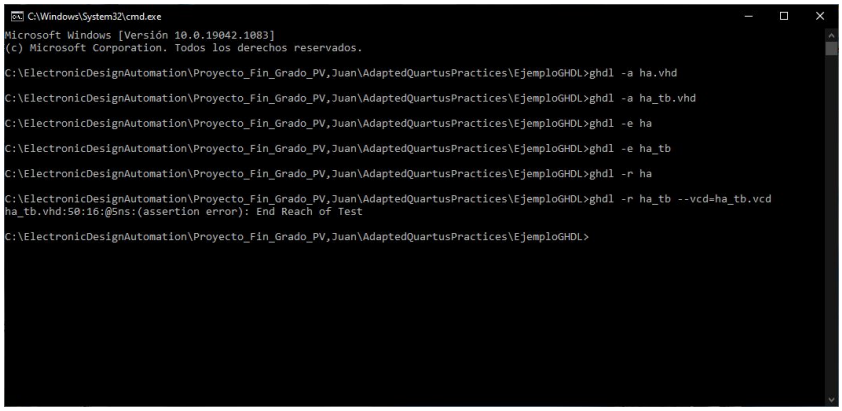


Figura 2.3: Comandos GHDL en CMD

2.2. GTKWave

2.2.1. Instalación del software

Para comenzar con la instalación del visualizador de ondas, se buscará la página principal del proyecto. Al igual que en el caso previo, dentro de la página web se explica brevemente las cualidades de la herramienta, junto a la documentación orientada para su instalación y caracterización.

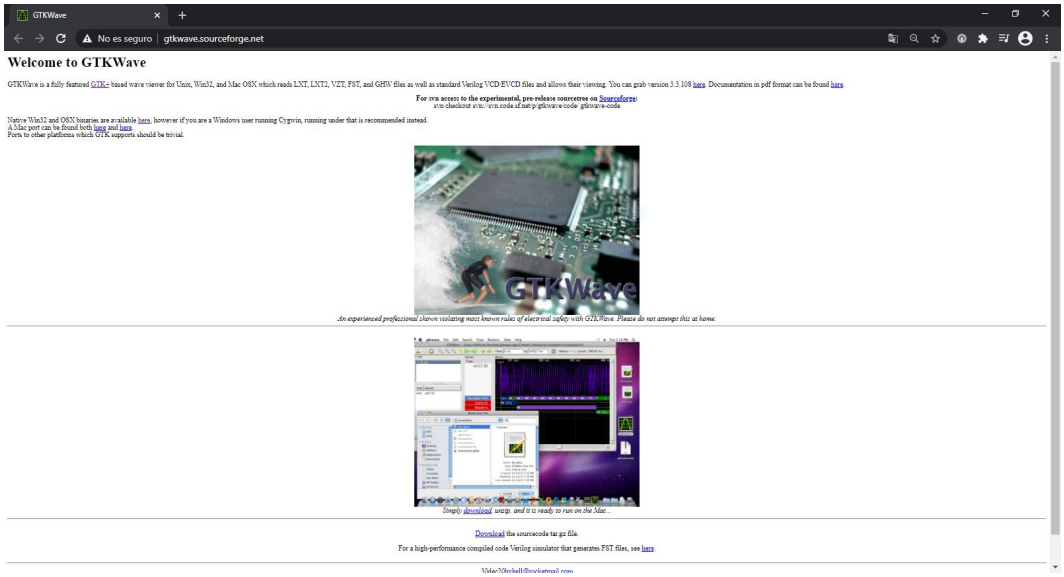


Figura 2.4: Página de inicio GTKWave

Como es habitual para software de uso libre, se encuentra disponible para los principales OS del mercado. Una vez se introduzca en el enlace facilitado, el directorio web se trasladará a *SourceForge*, un servidor web especializado en el alojamiento y la subida de archivos en la nube para su posterior descarga por parte de los usuarios con acceso a dicho contenido.

Una vez dentro, un listado con todas las actualizaciones del programa irán apareciendo. Para su instalación, busque el archivo nombrado como 'gtkwave-3.3.100-bin-win32.zip', usualmente siendo el más descargado de los paquetes disponibles, como muestra la figura 2.5.

Al igual que sucedía con GHDL, para este proyecto se ha optado por las versiones pre-compiladas de los programas debido a su fácil interpretación. En caso de posteriores actualizaciones, se recomienda consultar los enlaces facilitado previamente junto a la lectura adjunta a estos.

Como con GHDL, GTKWave se descomprimirá y modificará de carpeta a «C:\ElectronicDesignAutomation», siendo albergado en la misma común para todo el proyecto.

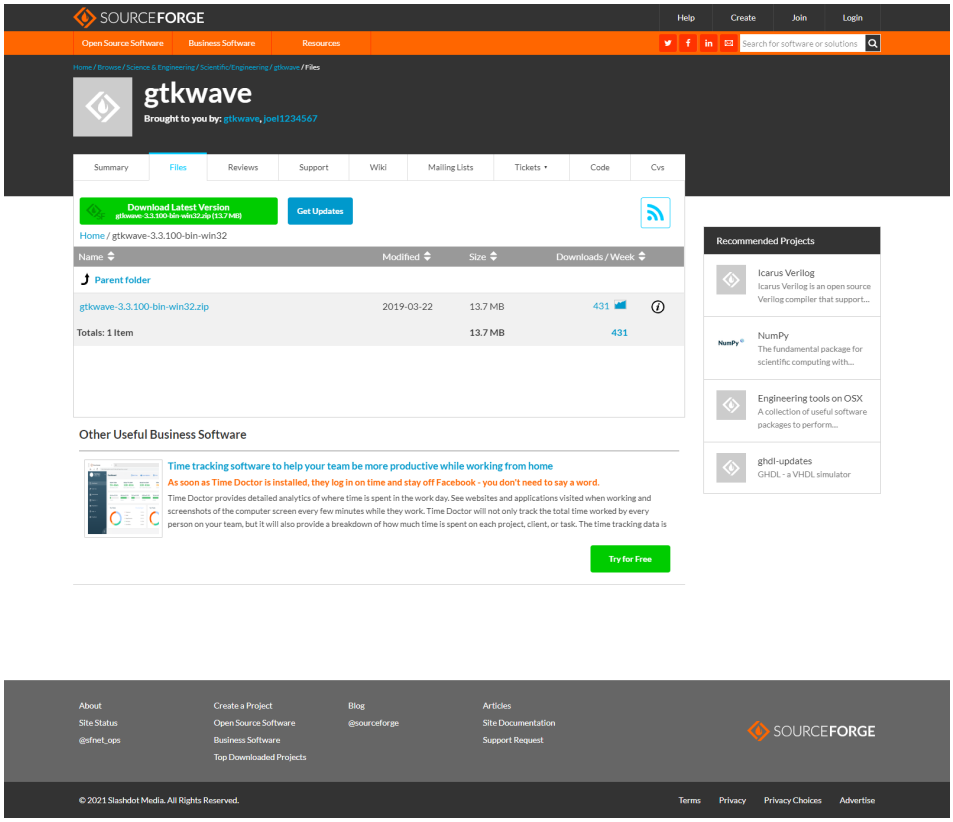


Figura 2.5: Página de descarga GTKWave - Versión v3.3.100 destinada a Windows

Con ambas descargadas, se procederá a la inclusión de ambos directorios en las variables de entorno del sistema. Este método permite la ejecución de las distintas funcionalidades del programa haciendo uso de la consola de comandos [CMD] para establecer un canal de comunicación directo entre la CMD del SO y los argumentos propios de ambos programas.

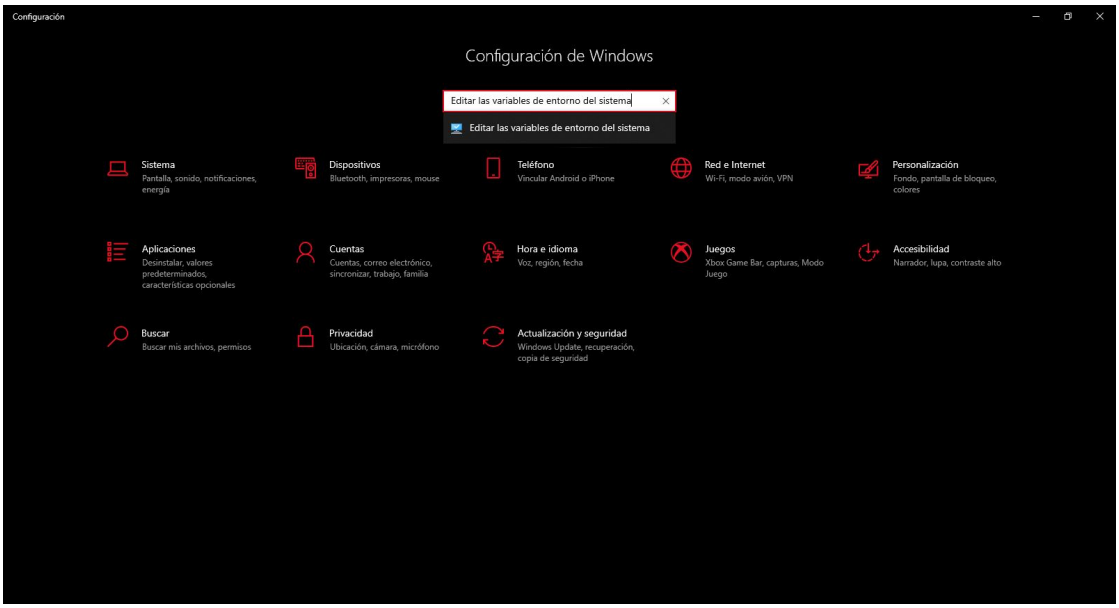


Figura 2.6: Configuración de Windows

Para ello, se introducirá en Configuración/Panel de Control (dependiendo de la versión de SO), y en la barra de búsquedas, escribir «*Editar las variables de entorno del sistema*», al igual que en la figura 2.6. Dentro de la ventana de 'Propiedades del sistema', haga click en la pestaña 'Opciones avanzadas' y busque el botón «*Variables de entorno...*».

Una vez dentro, en la sección de Variables del sistema, busque la casilla nombrada como 'Path' [Figura 2.7], y una vez dentro, debe añadir la carpeta 'bin' dentro de ambos softwares entre los directorios mostrados. Para ello, dispone de las herramientas al costado de la nueva ventana para incluir ambas rutas [Figura 2.8].

Una vez logrado, reinicie el sistema y habrá terminado por completar la primera fase del proceso. Únicamente falta la instalación del editor de texto destinado a código, y se podrá comenzar a utilizar todas las funcionalidades adquiridas con los distintos programas, explicadas con detalle en la Capítulo 2

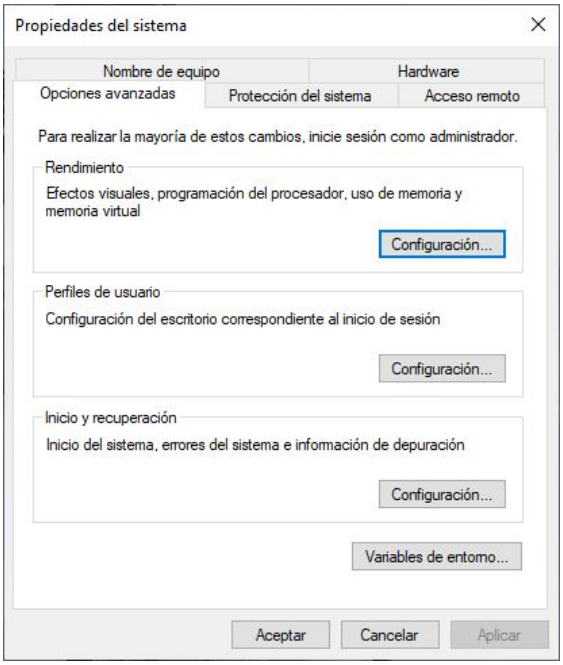


Figura 2.7: Propiedades del sistema - Opciones avanzadas - Variables del entorno

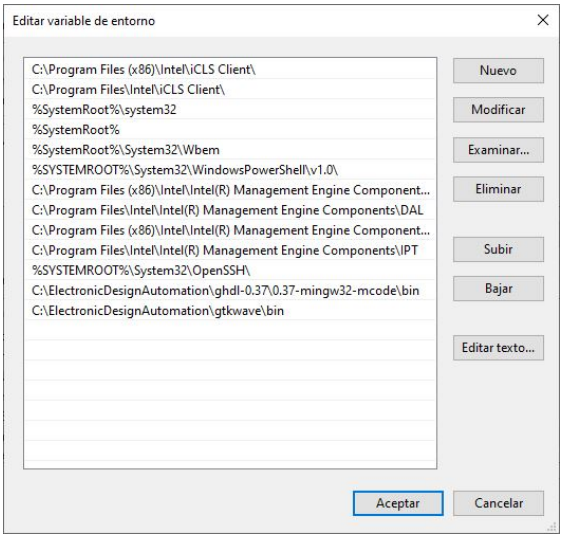


Figura 2.8: Edición de variables del entorno

2.2.2. Utilización del software

Como se ha indicado en la Sección 2.2, GTKWave es una herramienta de análisis, concretamente es una colección de archivos binarios conformados principalmente por dos aplicaciones:

- «***gtkwave***»*:* visualización de ondas. Permite la observación de resultados de simulaciones de datos obtenidos en un entorno analógico y/o digital, búsqueda y manipulación temporal, operaciones de salvaguardado parcial en formatos como *PostScript* o *FrameMaker*

- «***rtlbrowse***»*:* visión y navegación de código fuente a nivel registro-transferencia (en inglés, Register-Transfer Level, RTL) de un diseño del circuito digital simulado. Tras ser analizado y procesado por una aplicación auxiliar, GTKWave permite ver RTL tanto a nivel de archivo como módulo, lo que habilita la anotación en el propio código fuente.

Dichas herramientas facilitan el trabajo del usuario en la interpretación de los datos obtenidos tras el análisis, elaboración y ejecución del diseño circuital.

Como sucede con GHDL, GTKWave hace uso de la consola de comandos para interactuar con los archivos por medio del uso de argumentos. Aunque a continuación se citen los más relevantes para el desarrollo del proyecto, todos estos, junto con las opciones de personalización de los comandos, vienen detalladas en el manual de usuario:

- «*gtkwave:*» Herramienta de visualización para archivos .vcd, .lxt y .vzt, siendo VCD el estándar industrial para el manejo de datos de simulación:

```
gtkwave [option]... [DUMPFILE] [SAVEFILE] [RCFILE]
```

- *Comandos de conversión entre formatos*: aunque .vcd es un formato estandarizado, las alternativas propuestas por GTKWave tienen su origen en el ahorro de memoria como .lxt o la facilidad de interpretación como .tim

```
fst2vcd [option]... [FSTFILE]
vcd2fst [option]... [VCDFILE] [FSTFILE]
evcd2vcd [option]... [EVCDFILE]
lxt2vcd <filename>
vcd2lxt [VCDFILE] [LXTFILE] [option]...
vcd2lxt2 [option]... [VCDFILE] [LXTFILE]
vcd2vzt [option]... [VCDFILE] [VZTFILE]
vzt2vcd <filename>
```

- «*Interactive VCD:*» Herramienta de visualización para archivos .vcd dinámicos, permitiendo la simulación simultánea con la observación de los resultados depositados en el fichero:

```
mkfifo outfile.vcd
cver myverilog.v &
shmidcat outfile.vcd | gtkwave -v -I myverilog.sav
```

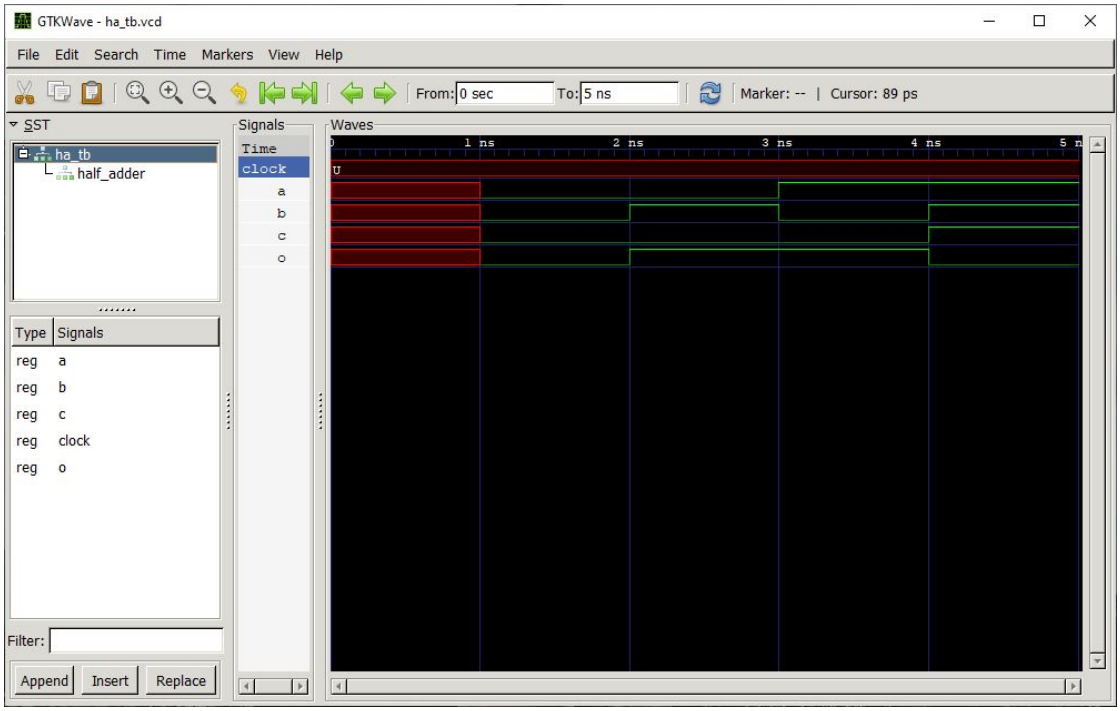


Figura 2.9: Ventana principal de GTKWave

Para este proyecto, se ha optado por el formato de fichero *.vcd*, no solo por ser el estándar industrial, sino por su equilibrio entre memoria e interpretación. Al análisis de cada módulo correspondiente a las entidades del código, serán asignados a cada variable de entrada y salida un carácter.

Por cada marca temporal, si existe un cambio en dicha variable, se reflejará este con el nuevo valor adquirido. Tomando como referencia la entidad principal del proyecto (el programa convierte automáticamente los caracteres a sus equivalentes en minúscula) *[$scope module ha_tb $end]*, a continuación se detallan las variables, junto con el tipo, número de bits necesarios para su representación, carácter equivalente en el fichero .vcd y la variable *[$var reg 1 " a $end]*

Se puede observar el archivo «***ha_tb.vcd***», referencia en el listado 8.1

2.3. Sublime Text

2.3.1. Instalación del software

De entre todas las herramientas utilizadas para este proyecto, el editor de texto orientado hacia código «Text Sublime 3», popularmente conocido como «Sublime», es la herramienta más abierta a modificaciones de las presentes. La elección frente a otros editores como puede ser «VS Code» es la versatilidad de su manejo y la cómoda personalización del mismo.

La inclusión de múltiples lenguajes de programación de forma directa, junto a las extensiones añadidas posteriormente hacen del programa una experiencia cómoda para un entorno de trabajo, facilitando su posterior ejecución. La personalización no se limita exclusivamente al entorno del código, sino también al diseño gráfico.

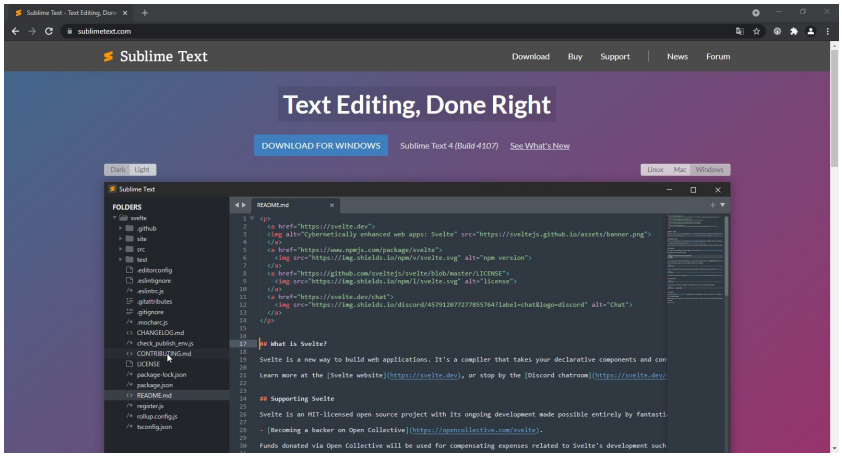


Figura 2.10: Página de descarga Sublime Text 3

Para la obtención del máximo partido para la aplicación, se darán una serie de consejos para optimizar el uso de las herramientas al alcance de los usuarios:

- ***Paquetes de recursos***: manteniendo «*Ctrl Izq. + Shift Izq. + P*», se abre un desplegable con la lista de comandos de Sublime, entre los que se incluye el instalador de paquetes «*Package Control: Install Package*», con diversas opciones, haciendo uso del repositorio y plugins subidos por la comunidad.

- ***Lenguaje VHDL***: Entre las posibles extensiones disponibles para el programa, se recomienda *Sublime Text 'Smart' VHDL Package*, cuyo objetivo es proveer características similares a un entorno de desarrollo integrado para

Verilog HDL. El desplazamiento a lo largo del código, la muestra por pantalla de los diferentes símbolos, las sangrías adaptables... son algunos ejemplos de los añadidos incluidos con este paquete

- ***Colores***: con *Material Theme*, y con el lenguaje de programación debidamente marcado, la comodidad durante el transcurso de la jornada laboral se traduce en un ahorro de tiempo en el transcurso de la manipulación del código. Disponible en quince colores predeterminados, permite la creación de un entorno único y adaptado a las necesidades de cada usuario.

- ***Atajos de comandos rápidos***: posiblemente, uno de los consejos generales más relevantes a la hora de programar. Agilizar los procesos repetitivos por medio del uso de comandos predefinidos supone un alivio en la carga de trabajo para altos tiempos de dedicación frente al monitor. Para evitar una saturación a la exposición de dichos elementos, es muy recomendable hacer uso de estos atajos para aligerar la carga del proyecto.

- ***División de pantalla***: poder trabajar en dos proyectos simultáneamente como punto de partida no es recomendable, ya que la des centralización del proyecto puede repercutir en retrasos posteriores. Superada la fase inicial, no solo supone una mejora con respecto a trabajar en un único módulo del proyecto, sino que la segmentación del código permite ir verificando de manera periódica las diferentes versiones, evitando errores posteriores derivados de código ya escrito.

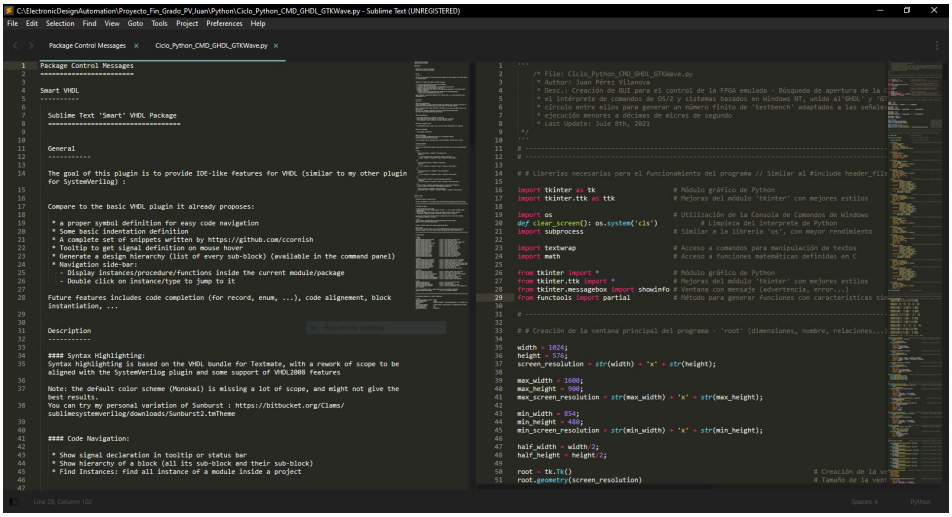


Figura 2.11: Sublime Text 3

Oficialmente, la versión descargada de Sublime es una prueba indefinida. Se recomienda, si la experiencia es agradable, colaborar con dicho proyecto mediante el registro de una licencia.

2.3.2. Utilización del software

Por último, se analiza el funcionamiento del editor de texto. Para comenzar a programar, se crea un nuevo archivo y en la pestaña inferior derecha, indique el lenguaje de programación a utilizar (por defecto, el texto no contiene significado de inicio en ningún lenguaje, es texto plano). Para la comprobación inicial del proyecto, se ha desarrollado un pequeño código de ejemplo, que viene referenciado en las figuras 2.11 y 2.9.

La funcionalidad de este sencillo código es un sumador de 1 bit con acarreo. Siendo *a* y *b* los valores de entrada, *o* el resultado de la suma correspondiente a la función lógica XOR y *c* el acarreo, producto de ambas entradas. También se dispone de un banco de pruebas donde se prueban todas las combinaciones posibles para comprobar como, efectivamente, la funcionalidad del código se ajusta a los resultados esperados.

Listing 2.1: ha.vhd

```
library ieee; --Libreria imprescindible para escritura VHDL
use ieee.std_logic_1164.all; --Estandar de descripcion logica
use ieee.numeric_std.all; --Funciones aritmeticas con signo

entity ha is --Definicion de entidad con entradas y salidas
port (a: in std_ulogic; --Entrada logica
b: in std_ulogic; --Entrada logica
o: out std_ulogic; --Salida logica
c: out std_ulogic); --Salida logica
end ha;

architecture ha_arc of ha is --Arquitectura de la entidad
begin
o <= a xor b; --Resultado de la suma de entradas
c <= a and b; --Acarreo de la suma de entradas
end  ha_arc;
```

Listing 2.2: ha_tb.vhd

```
library ieee; --Libreria imprescindible para escritura VHDL
use ieee.std_logic_1164.all; --Estandar de descripcion logica
use ieee.numeric_std.all; --Funciones aritmeticas con signo
```

```
entity ha_tb is --Definicion de entidad con entradas y salidas
port (clock : in std_logic);
end ha_tb;

architecture test of ha_tb is --Arquitectura de la entidad
component ha --Introduccion del componente 'ha.vhd'
port (a: in std_logic; --Entrada logica
b: in std_logic; --Entrada logica
o: out std_logic; --Salida logica
c: out std_logic); --Salida logica
end component;
signal a, b, o, c : std_ulogic;

begin
half_adder: ha port map (a => a, b => b, o => o, c => c);
process begin

a <= 'X'; -- Indeterminacion de la entrada a
b <= 'X'; -- Indeterminacion de la entrada a
wait for 1 ns;

a <= '0'; -- Entrada a sin corriente
b <= '0'; -- Entrada b sin corriente
wait for 1 ns;

a <= '0'; -- Entrada a sin corriente
b <= '1'; -- Entrada b con corriente
wait for 1 ns;

a <= '1'; -- Entrada a con corriente
b <= '0'; -- Entrada b sin corriente
wait for 1 ns;

a <= '1'; -- Entrada a con corriente
b <= '1'; -- Entrada b con corriente
wait for 1 ns;

assert false report "End Reach of Test";
wait;
end process;
end test;
```

2.4. Python

Por último, y más importante para el desarrollo del proyecto, se procede a la instalación del lenguaje de programación a utilizar. A julio de 2021, la versión última disponible para Windows estable es v3.9.6.

2.4.1. Instalación del software

La instalación de la última versión de Python comienza en su página web, concretamente en la sección de descargas. De manera automática, la página web recomendará el fichero comprimido más adecuado al sistema operativo y la arquitectura del procesador. En caso negativo, descendiendo en la misma página, podrá encontrar el fichero necesario.

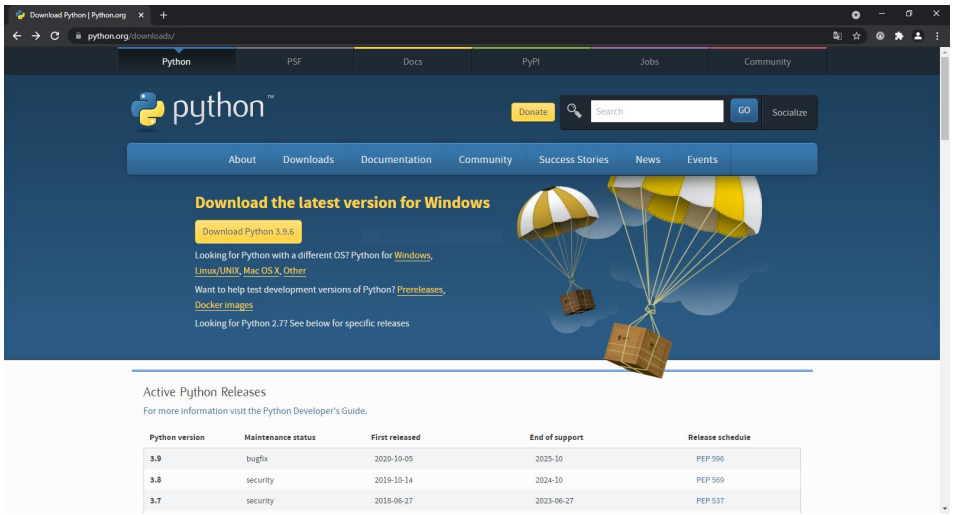


Figura 2.12: Página de descarga Python

Una vez abierto el ejecutable, se recomienda marcar las casillas para la instalación del lenguaje para todos los usuarios y la adición de la versión a las rutas dentro de las variables de entorno (como previamente se ha realizado manualmente en las figuras 2.6, 2.7 y 2.8)

Una vez terminada la instalación, existe una opción para deshabilitar el número máximo de caracteres de una ruta dentro las variables de estado. Se recomienda evitar manipular dicha limitación para mayor estabilidad del sistema operativo.

2.4.2. Utilización del software

Figura 2.13: Logo Python

La fortaleza en la programación de Python reside en la legibilidad de su código, y su multifuncionalidad lo hace una herramienta perfecta para este proyecto.

Python posee capacidad para soportar orientación a objetos, programación imperativa y funcional. Dicha adaptabilidad permite modificaciones futuras para programadores sin necesidad de adoptar un estilo concreto. [10]

Consiste en un lenguaje interpretado (capaz de analizar y ejecutar otros programas instrucción por instrucción), dinámico (las variables pueden adquirir valores de distinto tipo) y multiplataforma. El funcionamiento del proyecto y su descripción detallada en Python será indicada a lo largo de la Sección 5.2.

Listing 2.3: Librerías del proyecto

```
import tkinter as tk
# Modulo grafico de Python
import tkinter.ttk as ttk
# Mejoras del modulo 'tkinter' con mejores estilos

import os
# Utilizacion de la Consola de Comandos de Windows
def clear_screen(): os.system('cls')
# Limpieza del interprete de Python
import subprocess
# Similar a la libreria 'os', con mayor rendimiento

import textwrap
# Acceso a comandos para manipulacion de textos
import math
# Acceso a funciones matematicas definidas en C

from tkinter import *
# Modulo grafico de Python
from tkinter.ttk import *
# Mejoras del modulo 'tkinter' con mejores estilos
from tkinter.messagebox import showinfo
# Ventana con mensaje (advertencia, error...)
from functools import partial
# Metodo para generar funciones con caracteristicas similares
from pathlib import Path
# Comprobacion de existencia de ficheros/directorios indicados
```

Capítulo 3

Estado de la cuestión

Para la industria, el surgimiento de proyectos de código abierto no posee un impacto significativo. Sin embargo, en el ámbito académico o doméstico, la inclusión de programas basados en código abierto facilita una introducción sencilla y llamativa para los nuevos usuarios.

Para este capítulo, se buscarán alternativas en el mercado o en algún trabajo de investigación al campo de desarrollo del proyecto. Cabe destacar la diferenciación entre simulador y emulador previa su búsqueda en la red:

- **Simulador**: sistema software o hardware con ajuste de modificación del realismo, cuyo objetivo es la imitación de otro sistema complejo.
- **Emulador**: imitación de forma precisa del hardware informático, buscando la funcionalidad equivalente al aparato original.

3.1. Sector industrial

Dentro de los simuladores industrializados se incluyen software como Quartus II con ModelSim para Intel y XSIM con ISE para AMD, siendo ambos muy utilizados en sus respectivos productos. Ambas empresas no disponen de un elemento que imite en tiempo real las acciones sobre los dispositivos a implantar la descripción del usuario.

El único software distribuido de manera comercial encontrado por el autor de este proyecto es **Active-HDL**, un producto perteneciente a la empresa **ALDEC: The Design Verification Company.**

Active-HDL es un programa de *diseño, creación y simulación de FPGA* destinado para entornos de trabajo. Se encuentra exclusivamente disponible para Windows, e incluye un IDE con herramientas de diseño gráfico y un simulador de lenguaje mixto para una mayor rapidez de velocidad. [11]

La compatibilidad del programa se extiende con los dispositivos FPGA de las principales empresas del sector: Intel, Lattice, Microsemi (Actel), QuickLogico, Xilinx... [12]

Dentro del programa, ALDEC provee de más de 200 herramientas orientadas para la automatización de diseño electrónico, simulación, síntesis e implantación en FPGA a sus usuarios de manera simultánea sin necesidad de salir de la plataforma.

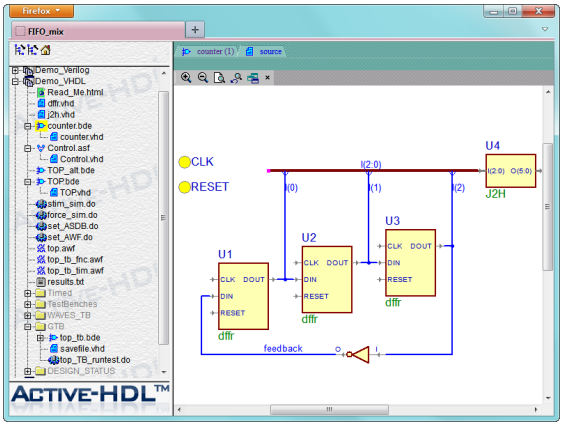


Figura 3.1: Proyecto en ActiveHDL

Como se puede observar en la figura 3.1, siempre está visible el directorio de trabajo donde se muestran los ficheros, entidades y arquitecturas

Una vez definidas, se indican las entradas y salidas con sus respectivos tipos, y el programa crea una plantilla con los elementos definidos. Tras indicar la arquitectura de la entidad, se puede dar inicio a la simulación desde la consola interna, donde el usuario cambiará las entradas y se definirá el tiempo total

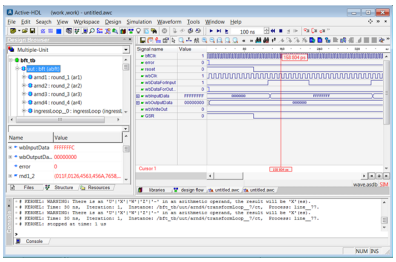


Figura 3.2: Simulación en ActiveHDL

Observando las figuras 3.1 y 3.2, y atendiendo a la documentación oficial, se proceden a listar las características principales del programa:

- Dirección del proyecto desde equipos conectados de manera local o en remoto con protección de encriptación estándar interoperable
- Entradas de diseño por medio del uso de texto, esquemáticos y/o máquinas de estado
- Simulación y compilación interactiva gráficamente de kernel común de lenguaje mixto que admite VHDL, Verilog, SystemVerilog y SystemC mediante herramientas de calidad

3.2. Sector académico

En términos académicos, haciendo una búsqueda exhaustiva en Google Scholar, no he encontrado literatura universitaria española destinada de manera específica a la emulación de VHDL, y de lectura extranjera, es interesante la transcripción del artículo *'Emulator environment based on an FPGA prototyping board'* [13].

Capítulo 4

Definición del trabajo

4.1. Justificación

A la vista del capítulo anterior, se hace evidente la escasez de soluciones ante la misma problemática.

La versión comercial de **Active-HDL** es muy completa, pero únicamente permanece accesible durante un *periodo anual* por medio del uso de licencias entre 2k$ y 2.5k$, dependiendo de la capacidad de modificación entre dispositivos.

Y aunque para estudiantes universitarios, siempre y cuando estén cursando estudios superiores en una universidad autorizada, **ALDEC** contiene una versión '*load and go*' preparada para su funcionamiento tras la instalación del software; para aquellos interesados en el mundo del diseño digital de circuitos, ve su capacidad de actuación muy limitada.

Para concluir, en ambas versiones es necesaria la cumplimentación de un formulario con datos personales. Un movimiento comprensible para mantener la integridad de su producto, pero en conflicto directo con la mentalidad informática del código abierto de compartir conocimiento desde una perspectiva altruista para el desarrollo de la ciencia.

Por otro lado, el artículo publicado por los profesores de la Universidad Nacional de Seúl resulta en un trabajo específico intermedio para una problemática particular a resolver, sin salir del marco teórico.

La FPGA utilizada por los autores corresponde a una **PCI9050**, tanto física como digitalmente, únicamente se hace uso de ficheros con formato VHDL, requiere de **manipulación manual** tras el inicio del emulador...

Por medio del uso de Python como programa de comunicación entre el simulador, visualizador y los componentes del emulador, existe una *automatización completa* desde la ejecución del programa desarrollado hasta su salida. Unido a la interpretación de resultados por medio del fichero .vcd generado en la simulación.

Con este proyecto, se busca la obtención de las funcionalidades principales de un emulador, abierto a la interpretación y modificación de los nuevos usuarios para adaptarse a sus necesidades.

El impacto en la comunidad de diseño electrónico para un aprendizaje desde los ensayos, sin necesidad de licencias ni especificaciones técnicas, supone un gran atractivo para los interesados en este campo de la informática.

4.2. Objetivos

Para el desarrollo del proyecto, se han establecido una serie de objetivos puntuales en forma de hitos intermedios para llevar un registro de los avances obtenidos.

Dichos parámetros de control han sido previamente especificados en la memoria descriptiva del proyecto:

1. **Síntesis** y simulación de diseños en VHDL usando únicamente herramientas de código abierto, accesibles al público

2. **Documentación** de los protocolos de diseño, síntesis y simulación con ejemplos, adaptados a prácticas de Electrónica Digital

3. **Evaluación** de los tiempos y formatos de salida de las simulaciones para la búsqueda de la automatización de pequeñas simulaciones sucesivas sobre un mismo diseño.

4. Conseguir una **emulación** del hardware en "falso tiempo real" mediante micro simulaciones, que permitan recrear la interacción de una placa física

5. Como opcional, la creación de un **entorno visual** para la gestión del emulador supondría un gran acabado final al proyecto

4.3. Metodología

4.3.1. Metodología STEAM

La **metodología STEAM** (Science, Technology, Engineering, Arts & Mathematics), antiguamente conocida como STEM, consiste en la *enseñanza-aprendizaje* de manera integrada de las distintas áreas del conocimiento indicadas. La transferencia de contenidos permite un *desarrollo integral* de los individuos formados para introducirse en la sociedad, gracias al aprendizaje significativo y contextualizado que implica el desarrollo dichas de competencia.

Utilizar como método de diseño un pilar de la enseñanza moderna basado en proyectos, permite adquirir una serie de beneficios derivados de este enfoque disciplinario:

- Fomento del *interés* y adopción temprana de la tecnología: con una motivación impulsada, el aumento de la exploración por parte de los recién ingresados se verá reforzado
- Generación de *experiencias* prácticas: una participación activa estimula el desarrollo cognitivo, nivelando el campo de habilidades del usuario
- Desarrollo del *pensamiento* crítico: análisis sintético potenciado por los trabajos desde diferentes enfoques
- Enseñanza de *valores*: para la programación/descripción, es necesaria una mente creativa con una visión amplia y flexible
- Exposición a procesos *creativos* diferentes: los conocimientos complementarios entre lenguajes derivan en preguntas reflexivas
- *Motivación* con trabajo en equipo: cualquier campo de desarrollo involucra la relación con compañeros, donde el diálogo y la colaboración juegan un papel clave
- *Preparación* para los trabajos del futuro: evitar el temor a la experimentación y tener amplia adaptabilidad a circunstancias novedosas
- *Construcción* de una mentalidad fuerte: ante la incertidumbre de resultados, inculcar la motivación a probar diseños para conocer el compromiso y entrega con el trabajo desarrollado

4.3.2. Objetivos de Desarrollo Sostenible

El principal promotor del proyecto, tal y como indica el cuarto objetivo de la agenda del Desarrollo Sostenible para el año 2030 organizado por la Organización de las Naciones Unidas, es la garantía de una educación de calidad.

Independiente de su origen o de su condición, el nivel presente durante la formación técnica, profesional y superior de calidad del alumnado debe ser la máxima de cualquier proyecto. No únicamente el desarrollo e innovación científico en el área electrónica, sino un impulso por el progreso humano. Solo con medidas cuyo origen radique en la entrega, se podrá construir las bases para un mañana mejor.

En segundo plano, la evolución natural al desarrollo en la investigación busca una generación de infraestructuras resilientes con elementos, cuyo impacto ecológico, se limite al tipo de energía consumida y la tecnología vigente.

La promoción hacia una industrialización sostenible a través del fomento, la innovación. Por medio del desarrollo de este proyecto, se busca aumentar el acceso a los servicios informáticos, orientados a la lógica programable, y su integración en las pequeñas industrias y emprendedores, particularmente en el ámbito académico

4.4. Planificación y estimación económica

Planificación del proyecto						
Tareas – Tiempo disponible	Febrero	Marzo	Abril	Mayo	Junio	Julio
Síntesis y simulación de diseños						
Documentación de protocolos						
Evaluación de tiempos y salidas						
Emulación con micro simulaciones						
Entorno gráfico						
Revisión de errores						
Desarrollo de la memoria						

Cuadro 4.1: Planificación del proyecto

Como se ha indicado previamente, un pilar fundamental del proyecto es la creación de una herramienta basada en código abierto y de uso libre. Derivados de este, los criterios de selección han buscado cumplir dicha norma, por ello, *no existen costes directos del proyecto* (desestimando la eficiencia del consumo energético del dispositivo utilizado, y la energía eléctrica utilizada para alimentar la batería).

Aunque cabe mencionar que bajo las condiciones de uso de las distintas tecnologías, se encuentran una numerosa cantidad de licencias para su uso:

- **GHDL**: el paquete front-end std.textio, y la librería de tiempo de ejecución GRT están cedidas bajo GNU GPLv2, y la documentación se encuentra bajo licencia Creative Commons CC-BY-SA.
- **GTKWave**: el programa se encuentra bajo GNU GPLv2
- **Sublime Text**: los términos y condiciones aprobados en el Acuerdo de Licencia Final de Usuario
- **Python**: aprobado bajo la Iniciativa Open-Source, se permite su uso, distribución y uso comercial

Capítulo 5

Modelo desarrollado

En este capítulo, se procede a detallar el proceso de obtención, así como el resultado final del emulador: la comunicación de Python con la consola de comandos, la lectura y escritura de los archivos .vcd y .vhd respectivamente, las funciones de control encargadas del proyecto, las ventanas de la interfaz gráfica...

5.1. Análisis del lenguaje

En esta sección se definen los procesos internos del programa, así como los razonamientos implicados en las funciones pertenecientes al proyecto.

5.1.1. VHDL: archivos .vhd y .vcd

La compresión, no solo de la escritura del lenguaje, sino el proceso de funcionamiento del simulador GHDL resultan fundamentales para cumplir con los objetivos establecidos en el marco de tiempo presentado.

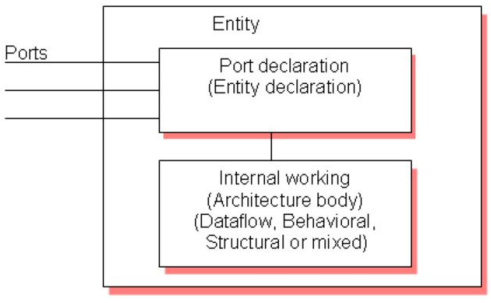


Figura 5.1: Estructura VHDL

La estructura de una archivo VHDL consta de: entidad, elemento que define el componente con su declaración de entradas y salidas; y arquitectura, elemento que define el comportamiento interno del bloque.

Una entidad puede contener varias arquitecturas, así como estas pueden declarar componentes y bloques. En todos ellos, existe la declaración de entradas y salidas con sus respectivos tipos incluidos en la norma IEEE 1076-2019.

Dichas señales pueden ser de diferentes tipos:

- *character*: carácter codificado en ASCII
- *integer*: entero con signo de 32 bits
- *time*: retardos en los bancos de prueba
- *boolean*: resultado lógico (true/false)
- *severity_level*: avisos de simulación
- *string*: cadena de caracteres
- *real*: numero en coma flotante
- *bit*: valores 0 y 1 (desuso)
- *bit_vector*: vector de bits (desuso)

Aunque por utilidad, los más usados son **std_logic** y **std_logic_vector**. Con el uso de este tipo de datos, se puede describir el estado de una señal con 9 valores: sin inicializar U, desconocido X, 0 y 1 lógico, alta impedancia Z, desconocido débil W, L cero débil y H uno débil, y no importa.

Unido al uso de puertas lógicas (NOT, AND, OR y XOR), multiplexores/demultiplexores, codificadores/decodificadores y biestables (flip-flops tipo J-K) en forma de operadores aritméticos, lógicos y relacionales; se puede comenzar a describir hardware con VHDL.

Para una comprensión extensa del lenguaje de descripción VHDL, es muy recomendable la lectura de 'Introducción a los sistemas digitales', obra de José Daniel Muñoz Frías, co-director de este proyecto. [14]

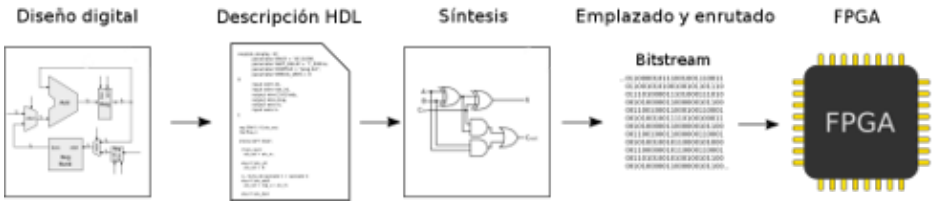


Figura 5.2: Flujo de trabajo de una FPGA

Como se puede observar en la figura 5.2, a la hora de configurar una FPGA (para este proyecto, el emulador), se sigue un flujo de trabajo. Dicho proceso comienza con el diseño del circuito, siendo muy recomendable tener pre-visualizada la imagen de la arquitectura mediante un diagrama de la aplicación, un boceto para generar un esquema de trabajo.

A continuación, se procede a la descripción por medio de VHDL en un fichero .vhd con las características previamente ideadas. Resulta de vital importancia la importación de las librerías reflejadas en el código Cód. 5.1

Listing 5.1: Librerías y paquetes estándar VHDL

```
library IEEE; --Libreria con las definiciones de logica programable
use IEEE.std_logic_1164.all; --Uso del tipo std_logic
use IEEE.std_logic_unsigned.all; --Op. mat. con std_logic
use IEEE.numeric_std.all; --Conversiones entre numeros
use IEEE.math_real.all; --Operaciones matematicas avanzadas
```

Una vez se ha realizado el diseño de la jerarquía y la codificación de los bloques, se procede a la compilación, simulación funcional y síntesis del archivo *.vhd*. Para este proyecto, esta labor recae en GHDL.

Atendiendo a su naturaleza, está estructurado por un controlador para llamar al compilador, ensamblador o enlazador en caso de ser necesario, una librería de tiempo de ejecución para ayudar a mantener la precisión de los ciclos, un front-end que analiza VHDL y back-end que generan código o una lista de conexiones.

Una vez ha sido correctamente sintetizado, en caso de no haber fallos, si existiese una FPGA, la configuración descrita sería volcada mediante 'bitstream' tras un proceso de Place&Route y una simulación temporal, e implantada físicamente. Para este proyecto, una vez se termina de sintetizar, en vez de realizar la descripción en la placa, se busca la generación de un archivo .vcd, almacén de los resultados temporales generados por las instrucciones del banco de pruebas.

Un 'testbench' o banco de pruebas es un fichero .vhd que incluye en un escala temporal valores de señales de entrada. Con ello, el simulador sintetiza dichos valores y los resultados los vuelca en un fichero .vcd

El formato .vcd, de sus siglas en inglés 'Value Change Dump', es el estándar definido desde la norma IEEE 1364-1995 para archivos volcados por herramientas de simulación lógica EDA. El contenido de estos archivos, como se puede comprobar en Cód. 8.1, se estructura en:

1. Encabezado. Información general como fecha, versión del simulador y unidad de la escala temporal

2. Definición de las variables. En este apartado, se muestra la información contenida en cada uno de las librerías, entidades, arquitecturas y componentes, junto a sus señales asociadas, donde cada variable será asociada con un identificador equivalente a un carácter ASCII

3. Situación inicial y cambios de valor. Con marcas temporales indicadas con un '#' al inicio, en una primera instancia se muestran los valores iniciales

que adoptan las señales, y por cada cambio producido, se generará una nueva marca temporal y se reflejarán los cambios. Para escalares, se asigna 0 o 1 directamente; para vectores, previo al valor se designa una 'b' y para reales, una 'r'

5.1.2. Python: Comunicación CMD - GHDL/GTKWave

Una vez comprendido y explicado el funcionamiento y estructura de los ficheros .vhd y .vcd, y haciendo uso de los comandos descritos en la Subsección 2.1.2 y Subsección 2.2.2, se desarrolla un canal de comunicación por medio de la CMD.

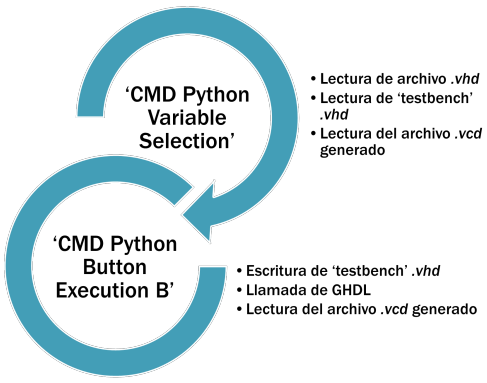


Figura 5.3: Ciclo de comunicación Python con CMD - GHDL/GTKWave

Como se muestra en el código listado 8.8, dicho 'script' contiene la esencia de actuación planteada en el ciclo reflejado en la figura 5.3, y aunque está modificado en el proyecto final, su concepto es el mismo. Se procede a realizar un breve análisis de la interacción entre archivos, directorios y programas:

- Se realiza una lectura inicial del archivo VHDL para la recogida de información esencial. Llamando a la función asociada al código Cód. 8.9, se obtiene el punto donde se debe sobrescribir el banco de pruebas, los caracteres asociados a cada una de las señales y la posición del último cambio en el .vcd.
- Una vez iniciada la emulación, cada vez que Python detecte algún cambio en el canal de una señal, llamará a la función asociada al código Cód. 8.10, generando un nuevo .vcd en centésimas de milisegundos y obteniéndose de nuevo los valores de las distintas señales, la última marca temporal registrada y registrando dichos cambios en la emulación

En resumen, tras una llamada a CMD_Python_Variable_Selection, se obtienen los elementos necesarios para comenzar el ciclo de emulación con la función CMD_Python_Button_Execution_B siendo llamada en aquellas ocasiones necesarias, para aumentar la velocidad de ejecución, resolución y evitar errores.

Como se puede extraer del código y la figura representativa del ciclo de comunicación, existen una serie de variables globales necesarias para la ejecución del programa. Entre estas se encuentran el nombre del fichero .vhd, el banco de pruebas vacío .vhd, el directorio de ambos ficheros y el nombre de la entidad global del banco de pruebas, cuyas señales son las medibles directamente por el programa.

5.2. Diseño

Una vez el tratamiento interno de datos ha sido analizado desde la perspectiva del lenguaje de la descripción y la programación, se dispone a definir la interfaz gráfica del emulador con sus respectivas características, considerada pertinente para la mayor rapidez de resolución del usuario.

Dicha GUI (Graphic User Interface) tendrá como objetivo materializar las acciones del usuario de manera simultanea con la manipulación de los archivos .vhd y .vcd, buscando la mayor flexibilidad de configuración.

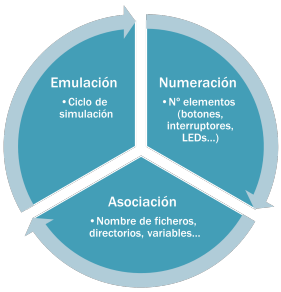


Figura 5.4: Diagrama GUI

5.2.1. Elección de Inputs/Outputs

Figura 5.5: Ventana de elección - Entradas y salidas - FPGA Emulator

Con el fin de aumentar las posibles integraciones futuras dentro de la estructura del programa, se van a utilizar un subconjunto de elementos electrónicos sencillos.

Para la primera sección, se busca la elección del número de elementos de cada tipo. En base a distintas referencias, observando la gama media de FPGAs comerciales y de código abierto, se ha optado por la inclusión de los siguiente elementos:

- **Pulsadores**: también conocidos como botones, permiten la circulación de corriente eléctrica por medio de presión. En la versión actual del proyecto, no incluyen la posibilidad de mantener su presión de manera indefinida, constituyendo un pulso fijo dentro del ciclo interno del reloj.
- **Interruptores**: al igual que los pulsadores, pueden bloquear el paso de electrones, teniendo dos posibles estados. En la versión actual del proyecto, supone la alternativa a botones mantenidos con presión, con posibilidad de almacenar el valor actual.
- **Visualizador 7 Segmentos**: elemento utilizado para representar caracteres, concretamente números, en equipos electrónicos. Cada combinación es diferente y evita entrar en conflicto con otras.
- **LEDs verdes y rojos**: diodo formados por un ánodo y un cátodo, estos elementos direccionales se presentan en varios formatos de color, entre los que destacan el verde y el rojo.

Como el comentario de texto presente en la figura 5.5 indica, únicamente se pueden tener hasta diez elementos de cada tipo. Para posteriores modificaciones, se planea eliminar dicha limitacion e incluir botones para resetear las casillas numericas a los valores predefinidos.

Para hacer uso de estos, marque aquellas casillas numéricas para volver al estado inicial y el programa llamará a la función «*name_display_func*», devolviendo a su situación predeterminada.

Para registrar todos los valores, siempre que el número de entradas y salidas totales sean distinto de cero, haciendo click en el botón 'Naming Vars.', se llamará a la función «*name_asociation*» avanzará a la siguiente ventana del programa, descrita en la Figura 5.6.

5.2.2. Asociación de variables con periféricos

En la siguiente ventana, representada en la figura 5.6, el programa contiene diversos espacios de escritura donde se debe introducir cada uno de los requisitos solicitados, incluidas el nombre de las variables del archivo VHDL.

La aparición de cuadros de texto del último punto marcado se generarán de forma procedimental, de acuerdo a las especificaciones previamente indicadas en la ventana de elección del programa.

Figura 5.6: Ventana de asociación - Nombres de variables - FPGA Emulator

Una vez se haya escrito todos los elementos correctamente, el programa dará inicio a la emulación por medio de la función «*FPGA_initiation*», iniciando el emulador junto al ciclo de comunicación, tal y como se describe en la Subsección 5.1.2

5.2.3. Emulación dinámica de FPGA

Con la emulación ya cargada, se puede ejecutar la descripción de la placa en tiempo real. Tras diversas comprobaciones y mediciones, la media estimada de reacción son centésimas de segundos, siendo una gran ventaja unida al éxito de la funcionalidad de la aplicación.

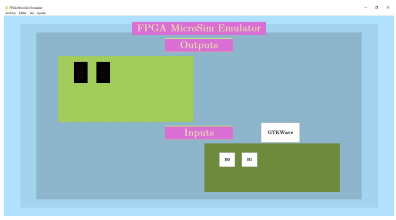


Figura 5.7: Ventana de emulación - Situación inicial - FPGA Emulator

5.3. Implementación

Previa la creación de un archivo VHDL, destacar ciertas características del programa: ventana de tamaño ajustable con dimensiones mínimas, barra de menú con funcionalidades como cambio entre ventanas (numeración, asociación y emulación), modificación del número de elementos, ventana de ayuda y contacto; y visualización del historial de estímulos con GTKWave.

Para demostrar la funcionalidad del proyecto, se ha optado por la implantación de un programa ejemplo nombrado 'ContadorPulsaciones.vhd', cuyo código completo se encuentra en el Capítulo 8 Cód. 8.2.

Este configuración busca la recreación de un registro con dos entradas y una salida. Mediante el uso de dos botones, con uno se mantiene el registro del número de veces pulsado, 'p1' y aumenta por cada nueva pulsación. El otro muestra en los displays 7 segmentos 's1' y's0' el número de pulsaciones guardado, 'p0'.

La resultados obtenidos de la emulación serán documentados en el Capítulo 6

Capítulo 6

Análisis de resultados

Figura 6.1: Ventana de elección - 'ContadorPulsaciones.vhd'

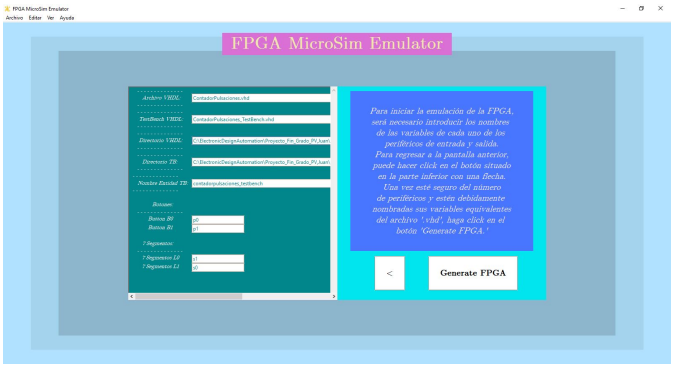


Figura 6.2: Ventana de asociación - FPGA Emulator

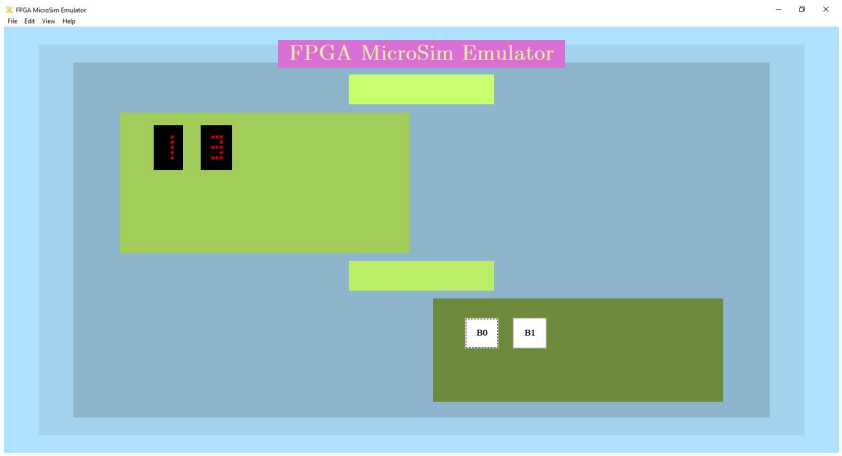


Figura 6.3: Ventana de emulación - 'ContadorPulsaciones.vhd'

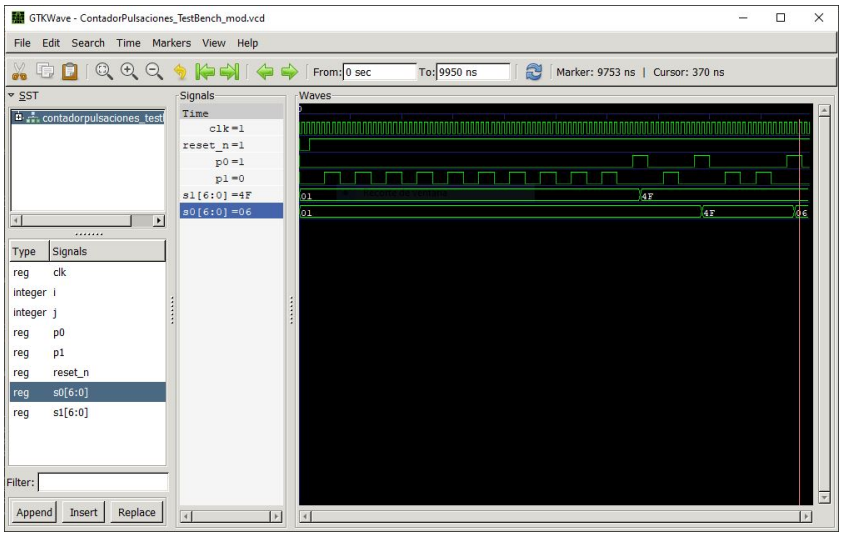


Figura 6.4: GTKWave - 'ContadorPulsaciones.vhd'

Observando las capturas de pantalla realizadas a las distintas ventanas del emulador y del visualizador, se puede observar no solo la salida por pantalla del registro total. En conjunción con GTKWave, se puede verificar los estímulos introducidos por el usuario, haciendo una experiencia de prototipado más completa.

Atendiendo al objetivo final de este proyecto, se puede corroborar con los resultados obtenidos la obtención de un emulador funcional mediante el uso de un simulador open-source, con solidez experimental y reducidos tiempos de ejecución.

Capítulo 7

Conclusiones y trabajos futuros

En este último capítulo, se hará una reflexión del proyecto, comparando los objetivos planteados con los resultados obtenidos, y se abrirán añadidos futuros para solventar el decaimiento del programa en determinados aspectos.

7.1. Conclusiones del proyecto

El objetivo principal del proyecto ha sido cumplido satisfactoriamente en los plazos de entrega previstos con un riguroso seguimiento quincenal hasta mitad del proyecto, y mensual en la segunda mitad.

El programa permite la síntesis y simulación de diseños en VHDL, consiguiendo una emulación de hardware en "falso tiempo real" mediante micro-simulaciones llamadas por el usuario, que recrear la interacción con una placa física.

Todo ello respetando la normativa vigente, con una profunda lectura previa, añadiendo el objetivo opcional de utilizar una interfaz gráfica para mayor agilidad de trabajo en los proyectos a describir.

Como objetivo no prioritario de forma activa, pero con beneficios notables para el proyecto es la velocidad de síntesis, lectura y escritura en conjunto, dando resultados más que favorables, reduciendo los tiempos de espera

Por medio de este soporte a nivel informático, se abre la posibilidad de realizar las prácticas de forma telemática, permitiendo obtener resultados similares a aquellos que se obtendrían dentro del formato presencial en una época de pademia global con dificultades para ejecutar clases en laboratorios y aulas.

La finalización de este proyecto ha culminado en la creación de una herramienta libre y de código abierto para la comunidad de la electrónica digital.

7.2. Implementaciones futuras

En relación al software descargado, existen determinadas mejoras para facilitar el uso de las mismas, tanto para este proyecto como para futuros desarrollos:

- **GHDL** - Soporte completo de la revisión estándar del IEEE 1076 del año 2008, versión pre compilada de archivos binarios v1.0.0, desarrollo completo de 'pyGHDL' (interfaz de Python incluida en la librería *'libghdl'* mediante la implantación de LSP, actualmente para desarrollo avanzado)...
- **GTKWave** - Corrección de configuración predeterminada en la escala temporal de los ficheros, utilización de manera predeterminada de archivos *.lxt* en formato de ahorro (*.vcd* en determinados proyectos de numerosos archivos, la memoria ocupada puede superar el volumen máximo para agilidad y lectura del mismo en tiempos reducidos), inclusión de herramientas...

En relación al software desarrollado para este proyecto, existen complementaciones futuras para las posteriores mejoras en la utilización del software:

- **Ventana de elección** - Botones de reset (mayor rapidez en la ventana de selección, el tiempo antes de la simulación no era clave), eliminación de la obligatoriedad en incluir al menos una entrada y salida para la versión actual del proyecto, (suponiendo una limitación para casos de estudio síncronos sin entradas).
- **Ventana de asociación** - Cuadro dinámico más fluido (inclusión del alcance de la rueda del ratón a dicha región completa), mejor transición entre ventanas (tras la generación del emulador, dicho tiempo es excluido; la lectura inicial en proyectos de varios archivos deriva en un retardo temporal)
- **Ventana de emulación** - Distribución no uniforme de los elementos (ordenación con mayor grado de libertad de las entradas y salidas), posible reducción en los tiempos de carga (esta característica, aunque no era el objetivo del proyecto, se han conseguido tiempos de carga muy óptimos; su reducción podría venir derivada del uso de un mejor equipo)
- **Características generales** - Aumento de funcionalidades en la barra de menú (edición de determinado números sin necesidad de repetir el proceso completo, reducción de 'glitches' tras el cambio de ventanas, visión modificable a escala de la salida por pantalla...)

Capítulo 8

Anexo: Código del programa

Listing 8.1: ha_tb.vcd

```
$date
Mon Jul 19 16:30:00 2021
$end
$version
GHDL v0
$end
$timescale
1 fs
$end
$scope module standard $end
$upscope $end
$scope module std_logic_1164 $end
$upscope $end
$scope module numeric_std $end
$upscope $end
$scope module ha_tb $end
$var reg 1 ! clock $end
$var reg 1 " a $end
$var reg 1 # b $end
$var reg 1 $ o $end
$var reg 1 % c $end
$scope module half_adder $end
$var reg 1 & a $end
$var reg 1 ' b $end
$var reg 1 ( o $end
$var reg 1 ) c $end
$upscope $end
$upscope $end
```

```
$enddefinitions $end
#0
U!
X"
X#
X$
X%
X&
X'
X(
X)
#1000000
0"
0#
0$
0%
0&
0'
0(
0)
#2000000
1#
1$
1'
1(
#3000000
1"
0#
1&
0'
#4000000
1#
0$
1%
1'
0(
1)
#5000000
```

Listing 8.2: ContadorPulsaciones.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ContadorPulsaciones is

port (
clk             : in std_logic;
reset_n         : in std_logic;

p1                      : in std_logic;
p0                      : in std_logic;

s1              : out std_logic_vector(6 downto 0);
s0              : out std_logic_vector(6 downto 0));
end ContadorPulsaciones;

architecture ContadorPulsaciones_arch of ContadorPulsaciones is

component DetectorFlancoSubida
port (
e       : in std_logic;
reset_n : in std_logic;
clk     : in std_logic;
s       : out std_logic);
end component;

component BinA7Seg
port (
A_Comp   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
A_D_Comp : OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
end component;

component Despliegue
port (
clk             : in std_logic;
reset_n         : in std_logic;

p0                      : in std_logic;

s1aux   : in std_logic_vector(3 downto 0);
s0aux   : in std_logic_vector(3 downto 0);
```

```
s1              : out std_logic_vector(3 downto 0);
s0              : out std_logic_vector(3 downto 0));
end component;

component ContadorAscMod10
port (
clk             : in std_logic;
reset_n         : in std_logic;
enable  : in std_logic;
co              : out std_logic;
s                       : out std_logic_vector(3 downto 0));
end component;

signal p1aux : std_logic;
signal p0aux : std_logic;

signal coaux    : std_logic;
signal coaux2   : std_logic;

signal salida1 : std_logic_vector(3 downto 0);
signal salida0 : std_logic_vector(3 downto 0);

signal salida1aux       : std_logic_vector(3 downto 0);
signal salida0aux       : std_logic_vector(3 downto 0);

begin

FlancoP1 : DetectorFlancoSubida
port map(
e       => p1,
reset_n => reset_n,
clk     => clk,
s       => p1aux);

FlancoP0 : DetectorFlancoSubida
port map(
e       => p0,
reset_n => reset_n,
clk     => clk,
s       => p0aux);

ContadorDecenas : ContadorAscMod10
```

```
port map(
clk              => clk,
reset_n          => reset_n,
enable  => coaux,
co               => coaux2,
s                        => salida1aux);

ContadorUnidades : ContadorAscMod10
port map(
clk              => clk,
reset_n          => reset_n,
enable  => p1aux,
co               => coaux,
s                        => salida0aux);

DespliegueBin : Despliegue
port map (
clk              => clk,
reset_n          => reset_n,
p0                       => p0aux,
s1aux   => salida1aux,
s0aux   => salida0aux,
s1               => salida1,
s0               => salida0);

Decimas : BinA7Seg
port map(
A_Comp  => salida1,
A_D_Comp => s1);

Unidades : BinA7Seg
port map(
A_Comp  => salida0,
A_D_Comp => s0);

end ContadorPulsaciones_arch;
```

Listing 8.3: BinA7Seg.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY BinA7Seg IS
port (
A_Comp : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
A_D_Comp : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
);
END BinA7Seg;

ARCHITECTURE BinA7Seg_Arch of BinA7Seg IS

begin

with A_Comp select
A_D_Comp <=

"0000001" when "0000",
"1001111" when "0001",
"0010010" when "0010",
"0000110" when "0011",
"1001100" when "0100",
"0100100" when "0101",
"0100000" when "0110",
"0001111" when "0111",
"0000000" when "1000",
"0000100" when "1001",
"0001000" when "1010",
"1100000" when "1011",
"0110001" when "1100",
"1000010" when "1101",
"0110000" when "1110",
"0111000" when "1111",
"1111111" when others;

end BinA7Seg_Arch;
```

Listing 8.4: ContadorAscMod10.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ContadorAscMod10 is

port (
clk             : in std_logic;
reset_n         : in std_logic;
enable  : in std_logic;
co              : out std_logic;
s                       : out std_logic_vector(3 downto 0));
end ContadorAscMod10;

architecture ContadorAscMod10_arch of ContadorAscMod10 is

signal contador : unsigned(3 downto 0);

begin

process(clk, reset_n)

begin
if reset_n = '0' then
contador <= (others => '0');
elsif clk'event and clk = '1' then
if enable = '1' then
if contador = "1001" then
contador <= (others => '0');
else
contador <= contador + 1;
end if;
end if;
end if;
end process;

co <= '1' when contador = "1001" and enable = '1'
else '0';

s <= std_logic_vector(contador);

end ContadorAscMod10_arch;
```

Listing 8.5: DetectorFlancoSubida.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity DetectorFlancoSubida is
port (
e       : in std_logic;
reset_n : in std_logic;
clk     : in std_logic;
s       : out std_logic
);
end DetectorFlancoSubida;

architecture DetectorFlancoSubida_arc of DetectorFlancoSubida is

type t_estados is (Esp0 , Pulso , Esp1);
signal estado_act , estado_sig : t_estados;

begin

VarEstado : process (clk, reset_n)
begin
if reset_n = '0' then
estado_act <= Esp0;
elsif (clk'event and clk = '1') then
estado_act <= estado_sig;
end if;
end process VarEstado;

TransicionEstados : process (estado_act, e)
begin
estado_sig <= estado_act;
case(estado_act) is
when Esp0 =>
if e = '1' then
estado_sig <= Pulso;
end if;
when Pulso =>
if e = '1' then
estado_sig <= Esp1;
else
estado_sig <= Esp0;
```

```
end if;
when Esp1 =>
if e = '0' then
estado_sig <= Esp0;
end if;
when others =>
estado_sig <= Esp0;
end case;
end process TransicionEstados;

Salidas : process (estado_act)
begin
s <= '0';
case estado_act is
when Esp0   => null;
when Pulso  => s <= '1';
when Esp1   => null;
when others => null;
end case;
end process Salidas;

end  DetectorFlancoSubida_arc;
```

Listing 8.6: Despliegue.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Despliegue is

port (
clk             : in std_logic;
reset_n         : in std_logic;

p0                      : in std_logic;

s1aux   : in std_logic_vector(3 downto 0);
s0aux   : in std_logic_vector(3 downto 0);

s1              : out std_logic_vector(3 downto 0);
s0              : out std_logic_vector(3 downto 0));
end Despliegue;

architecture Despliegue_arch of Despliegue is

type t_estados is (Reposo, Mostrar, Contar);

signal estado_act , estado_sig : t_estados;

signal s1mem : std_logic_vector(3 downto 0) := "0000";
signal s0mem : std_logic_vector(3 downto 0) := "0000";

begin

VarEstado : process (clk, reset_n, estado_sig)
begin
if reset_n = '0' then
estado_act <= Reposo;
elsif (clk'event and clk = '1') then
estado_act <= estado_sig;
end if;
end process VarEstado;

TransicionEstados : process (estado_act, p0, s1aux, s0aux)
begin
estado_sig <= estado_act;
```

```
case(estado_act) is

when Reposo =>
if p0 = '1' then
estado_sig <= Mostrar;
else
estado_sig <= Contar;
end if;

when Mostrar =>
s1mem <= s1aux;
s0mem <= s0aux;
estado_sig <= Contar;

when Contar =>
if p0 = '1' then
estado_sig <= Mostrar;
end if;

when others =>
estado_sig <= Reposo;
end case;

end process TransicionEstados;

with estado_act select
s1 <= "0000" when Reposo,
s1aux when Mostrar,
s1mem when Contar,
"0000" when others;

with estado_act select
s0 <= "0000" when Reposo,
s0aux when Mostrar,
s0mem when Contar,
"0000" when others;

end Despliegue_arch;
```

Listing 8.7: ContadorPulsaciones_TestBench.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ContadorPulsaciones_TestBench IS
END ContadorPulsaciones_TestBench;

ARCHITECTURE ContadorPulsaciones_arch OF ContadorPulsaciones_TestBench IS
-- constants
-- signals
SIGNAL clk : STD_LOGIC := '0';
SIGNAL p0 : STD_LOGIC;
SIGNAL p1 : STD_LOGIC;
SIGNAL reset_n : STD_LOGIC;
SIGNAL s0 : STD_LOGIC_VECTOR(6 DOWNTO 0);
SIGNAL s1 : STD_LOGIC_VECTOR(6 DOWNTO 0);
signal i : integer := 0;
signal j : integer := 0;
COMPONENT ContadorPulsaciones
PORT (
clk : IN STD_LOGIC;
p0 : IN STD_LOGIC;
p1 : IN STD_LOGIC;
reset_n : IN STD_LOGIC;
s0 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
s1 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
);
END COMPONENT;

BEGIN
i1 : ContadorPulsaciones
PORT MAP (
-- list connections between master ports and signals
clk => clk,
p0 => p0,
p1 => p1,
reset_n => reset_n,
s0 => s0,
s1 => s1
);

init : PROCESS
-- variable declarations
```

```
BEGIN
-- code that executes only once
WAIT;
END PROCESS init;

clk <= not clk after 50 ns;

always : PROCESS
-- optional sensitivity list
-- (        )
-- variable declarations
BEGIN

p0 <= '0';
p1 <= '0';
reset_n <= '0';

wait for 200 ns;
reset_n <= '1';

wait for 200 ns;
--std.env.stop;
assert false report "End Reach of Test: All correct" severity failure;

WAIT;
END PROCESS always;
END ContadorPulsaciones_arch;
```

Listing 8.8: python_cmd.py

```
import tkinter as tk
import tkinter.ttk as ttk
import os
import subprocess
import textwrap
import math
from tkinter import *
from tkinter.ttk import *
from tkinter.messagebox import showinfo

file_name_vhd = 'ha.vhd';
testbench_name_vhd = 'ha_tb.vhd';
file_cd = 'C:\ElectronicDesignAutomation\
Proyecto_Fin_Grado_PV,Juan\PracticasED';
testbench_cd = 'C:\ElectronicDesignAutomation\
Proyecto_Fin_Grado_PV,Juan\PracticasED';

entity_testbench_name = 'ha_tb';
list_inputs = ['clock','a','b','o','c'];
list_outputs = ['o','c'];

print(file_name_vhd, testbench_name_vhd,
file_cd, testbench_cd);

def CMD_Python_Communication(file_name_vhd,
testbench_name_vhd, file_cd, testbench_cd,
entity_testbench_name, list_inputs, list_outputs):

file_name = file_name_vhd.rsplit('.')[0]
testbench_name = testbench_name_vhd.rsplit('.')[0]

file_vhd_cd = file_cd + '\\' + file_name_vhd;
file_vcd_cd = file_cd + '\\' + file_name + '.vcd';

testbench_vhd_cd = testbench_cd + '\\' + testbench_name_vhd;
testbench_vcd_cd = testbench_cd + '\\' + testbench_name + '.vcd';

ghdl_file_scan = 'ghdl -s ' + file_name_vhd;
ghdl_file_analize = 'ghdl -a ' + file_name_vhd;

ghdl_testbench_scan = 'ghdl -s ' + testbench_name_vhd;
ghdl_testbench_analize = 'ghdl -a ' + testbench_name_vhd;
```

```
ghdl_file_execute = 'ghdl -e ' + file_name;
ghdl_testbench_execute = 'ghdl -e ' + testbench_name;

ghdl_testbench_run = 'ghdl -r ' + testbench_name +
 ' --vcd=' + testbench_name + '.vcd';

gtkwave_testbench_run = 'gtkwave ' + testbench_name + '.vcd';

initial_commands_vhdl = [
ghdl_file_scan,
ghdl_file_analize,
ghdl_file_execute]

initial_commands_tb = [
ghdl_testbench_scan,
ghdl_testbench_analize,
ghdl_testbench_execute,
ghdl_testbench_run,
gtkwave_testbench_run]

for command in initial_commands_vhdl:
subprocess.run(command, shell = True, cwd = file_cd)

for command in initial_commands_tb:
subprocess.run(command, shell = True, cwd = testbench_cd)

# Busqueda de las variables a interpretar

entity_text = '$scope module ' + entity_testbench_name + ' $end';

line_number = 0; entity_number = 0; read_value = 0; check = 0;

list_variables_txt = []
list_variables = []

list_inputs_var = []
list_outputs_var = []
list_outputs_var_value = []

with open(testbench_vcd_cd,'rt') as vcd:
for line in vcd:
line_number += 1
```

```
if entity_text in line:
entity_number = line_number

if (('$scope␣module' in line) and (line_number > entity_number > 0)
and (check == 0)):
check = 1

if (('$var␣reg' in line) and (line_number > entity_number)
and (check == 0)):
list_variables_txt.append(line.rstrip())
list_variables.append(list_variables_txt[line_number -
entity_number - 1].rsplit('␣')[3])

for i in range(len(list_inputs)):
if list_variables_txt[line_number -
entity_number - 1].rsplit('␣')[4] == list_inputs[i]:
list_inputs_var.append(list_variables_txt[line_number -
entity_number - 1].rsplit('␣')[3])

for j in range(len(list_outputs)):
if list_variables_txt[line_number -
entity_number - 1].rsplit('␣')[4] == list_outputs[j]:
list_outputs_var.append(list_variables_txt[line_number -
entity_number - 1].rsplit('␣')[3])

if '$enddefinitions␣$end' in line:
read_value = line_number

print(list_variables_txt)
print(list_variables)

print(list_inputs_var)
print(list_outputs_var)

CMD_Python_Communication(file_name_vhd, testbench_name_vhd,
file_cd, testbench_cd, entity_testbench_name, list_inputs,
list_outputs)
```

Listing 8.9: def CMD_Python_Variable_Selection()

```
## Funcion de comunicacion inicial para la extraccion de datos
## necesarios para la emulacion. Llamada una unica vez en la
## transicion de la ventana de asociacion y emulacion

file_name = (file_name_vhd.get()).rsplit('.')[0]
# Nombre del archivo VHDL sin extension
testbench_name = (testbench_name_vhd.get()).rsplit('.')[0]
# Nombre del archivo TestBench sin extension
testbench_vcd_mod = testbench_name + '_mod.vcd'
# Nombre del futuro archivo .vcd con extension
file_vhd_cd = (file_cd.get()) + '\\' + (file_name_vhd.get());
# Directorio del archivo VHDL incluido
testbench_vhd_cd = (testbench_cd.get()) + '\\' +
(testbench_name_vhd.get());
# Directorio del archivo TestBench incluido
testbench_vcd_mod_cd = (testbench_cd.get()) + '\\' +
testbench_vcd_mod;
# Directorio del archivo .vcd incluido

ghdl_file_scan = 'ghdl -s ' + (file_name_vhd.get());
ghdl_file_analize = 'ghdl -a ' + (file_name_vhd.get());
ghdl_file_execute = 'ghdl -e ' + file_name;
ghdl_testbench_scan = 'ghdl -s ' + (testbench_name_vhd.get());
ghdl_testbench_analize = 'ghdl -a ' + (testbench_name_vhd.get());
ghdl_testbench_execute = 'ghdl -e ' + testbench_name;
ghdl_testbench_run = 'ghdl -r ' + testbench_name + ' --vcd=' +
testbench_vcd_mod;

initial_file_commands = [
ghdl_file_scan,
ghdl_file_analize,
ghdl_file_execute]
# Analisis, elaboracion y sintesis del archivo VHDL

initial_testbench_commands = [
ghdl_testbench_scan,
ghdl_testbench_analize,
ghdl_testbench_execute,
ghdl_testbench_run]
# Analisis, elaboracion y sintesis del archivo TestBench

for command in initial_file_commands:
```

```
subprocess.run(command, shell = True, cwd = (file_cd.get()))

for command in initial_testbench_commands:
subprocess.run(command, shell = True, cwd = (testbench_cd.get()))

# Busqueda de las variables a interpretar dentro del archivo .vcd

entity_text = '$scope module ' + (entity_testbench_name.get()) +
 ' $end';

line_number = 0; entity_number = 0; read_value = 0; check = 0;
list_variables_txt = []; list_variables = [];

with open(testbench_vhd_cd,'rt') as vhd:
vhd_txt = vhd.readlines()

with open(testbench_vhd_cd,'rt') as vhd:
for line in vhd:
line_number += 1
if "reset_n <= '1';" in line:
reset_line.set(line_number);
last_line_entry_vhd.set(line_number);

line_number = 0; entity_number = 0; read_value = 0; check = 0;
list_variables_txt = []; list_variables = [];

with open(testbench_vcd_mod_cd,'rt') as vcd:
for line in vcd:
line_number += 1

if entity_text in line:
entity_number = line_number

if (('$scope module' in line) and (line_number > entity_number > 0)
and (check == 0)):
check = 1

if (('$var reg' in line) and (line_number > entity_number)
and (check == 0)):

list_variables_txt.append(line.rstrip())
list_variables.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])
```

```
if len(list_buttons_vcd) != 0:
for i in range(len(list_buttons_vcd)):
if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4]
== list_buttons_vcd[i]:
list_buttons_var_vcd.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

if len(list_switches_vcd) != 0:
for i in range(len(list_switches_vcd)):
if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4]
== list_switches_vcd[i]:
list_switches_var_vcd.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

if len(list_labels_vcd) != 0:
for j in range(len(list_labels_vcd)):
if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4]
== list_labels_vcd[j]:
list_labels_var_vcd.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

if len(list_greenleds_vcd) != 0:
for j in range(len(list_greenleds_vcd)):
if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4]
== list_greenleds_vcd[j]:
list_greenleds_var_vcd.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

if len(list_redleds_vcd) != 0:
for j in range(len(list_redleds_vcd)):
if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4]
== list_redleds_vcd[j]:
list_redleds_var_vcd.append(
list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

if '$enddefinitions $end' in line:
read_value = line_number
last_line_entry_vcd.set(read_value)

print(last_line_entry_vcd.get())

clear_screen()
```

Listing 8.10: def CMD_Python_Button_Execution_B(enter_value)

```
## Funcion de comunicacion presente durante la emulacion,
## siendo llamada cuando detecta un cambio en algun boton

file_name = (file_name_vhd.get()).rsplit('.')[0]
testbench_name = (testbench_name_vhd.get()).rsplit('.')[0]

testbench_vcd_mod = testbench_name + '_mod.vcd'

file_vhd_cd = (file_cd.get()) + '\\' + (file_name_vhd.get());
file_vcd_cd = (file_cd.get()) + '\\' + file_name + '.vcd';

testbench_vhd_cd = (testbench_cd.get()) + '\\' + (testbench_name_vhd.get())
testbench_vcd_cd = (testbench_cd.get()) + '\\' + testbench_name + '.vcd';
testbench_vcd_mod_cd = (testbench_cd.get()) + '\\' + testbench_vcd_mod;

line_number = 0; entity_number = 0; read_value = 0; check = 0;
list_variables_txt = []; list_variables = [];

with open(testbench_vhd_cd,'rt') as vhd:
vhd_txt = vhd.readlines()

with open(testbench_vhd_cd,'wt') as vhd:
vhd.writelines( vhd_txt[ 0 : last_line_entry_vhd.get() ] )
vhd.writelines( [ "    \n", "    wait for 300 ns;\n" ,
"    " + list_buttons_vcd[enter_value] + " <= '1';\n" ,
"    wait for 300 ns;\n" ,
"    " + list_buttons_vcd[enter_value] + " <= '0';\n", "    \n" ] )
vhd.writelines(
vhd_txt[ (last_line_entry_vhd.get() + 1) : len(vhd_txt) ] )

last_line_entry_vhd.set(last_line_entry_vhd.get() + 5)

'''if n_switches_value.get() != 0:
pass;'''

# Ejecucion del nuevo codigo, con las entradas
# debidamente indicadas

ghdl_file_scan = 'ghdl -s ' + (file_name_vhd.get());
ghdl_file_analize = 'ghdl -a ' + (file_name_vhd.get());
ghdl_file_execute = 'ghdl -e ' + file_name;
```

```
ghdl_testbench_scan = 'ghdl -s ' + (testbench_name_vhd.get());
ghdl_testbench_analize = 'ghdl -a '+(testbench_name_vhd.get());
ghdl_testbench_execute = 'ghdl -e ' + testbench_name;
ghdl_testbench_run = 'ghdl -r ' + testbench_name + ' --vcd='
+ testbench_vcd_mod;

initial_file_commands = [
ghdl_file_scan,
ghdl_file_analize,
ghdl_file_execute]

initial_testbench_commands = [
ghdl_testbench_scan,
ghdl_testbench_analize,
ghdl_testbench_execute,
ghdl_testbench_run]

for command in initial_file_commands:
subprocess.run(command, shell = True, cwd = (file_cd.get()))

for command in initial_testbench_commands:
subprocess.run(command, shell = True,cwd=(testbench_cd.get()))

line_number = 0; list_variables_index = 0;
list_variables_txt = []; list_variables = [];

# Busqueda de las variables
# a interpretar dentro del archivo .vcd

print(last_line_entry_vcd.get())

with open(testbench_vcd_mod_cd,'rt') as vcd:

for line in vcd:

line_number += 1

if (line_number > last_line_entry_vcd.get()):
if len(list_labels_var_vcd) != 0:
for i in range(len(list_labels_var_vcd)):
if list_labels_var_vcd[i] in line:
if (((list_labels_var_vcd[i] == '#') and (
bool( list_variables_txt[ list_variables_index ]
```

```
.rsplit('#')[0] != '') == True))
or (list_labels_var_vcd[i] != '#')):
list_variables_txt.append(line.rstrip())
list_variables.append(
(list_variables_txt[ list_variables_index ]
.rsplit(str( '␣' + list_labels_var_vcd[i] ))[0]).rsplit('b')[1] )
globals()[label_text_var[i]]
.set( "\n".join(binA7seg_representations[
list_variables[ list_variables_index ] ]) )
list_variables_index += 1

if len(list_greenleds_var_vcd) != 0:
for i in range(len(list_greenleds_var_vcd)):
if list_greenleds_var_vcd[i] in line:
if (((list_greenleds_var_vcd[i] == '#') and (
bool( list_variables_txt[ list_variables_index ].rsplit('#')[0] != '')
 == True)) or (list_greenleds_var_vcd[i] != '#')):
list_variables_txt.append(line.rstrip())
list_variables.append(
(list_variables_txt[ list_variables_index ]
.rsplit(str( list_greenleds_var_vcd[i] )[0]) ))
greenLEDConfiguration( globals()[greenled_name_var[i]],
bool(list_variables[ list_variables_index ]) )
list_variables_index += 1

if len(list_redleds_var_vcd) != 0:
for i in range(len(list_redleds_var_vcd)):
if list_redleds_var_vcd[i] in line:
if (((list_redleds_var_vcd[i] == '#') and (
bool( list_variables_txt[ list_variables_index ].rsplit('#')[0] != '')
== True)) or (list_redleds_var_vcd[i] != '#')):
list_variables_txt.append(line.rstrip())
list_variables.append(
(list_variables_txt[ list_variables_index ]
.rsplit( list_greenleds_var_vcd[i] ))[0])
redLEDConfiguration( globals()[redled_name_var[i]],
bool(list_variables[ list_variables_index ]) )
list_variables_index += 1

last_line_entry_vcd.set(line_number)

clear_screen()
```

Listing 8.11: Proyecto_Fin_Grado_PV Juan.py

```
'''
/* File: Proyecto_Fin_Grado_PV,Juan.py
* Author: Juan Perez Vilanova
* Desc.: Creacion de GUI para el control de la FPGA emulada -
* Busqueda de apertura de la Consola de Comandos de Windows,
* el interprete de comandos de OS/2 y sistemas basados en Windows
* NT, unido al 'GHDL' y 'GTKWave', donde se establecera un
* circulo entre ellos para generar un numero finito de 'testbench'
* adaptados a las senales de entrada, con intervalos de
* ejecucion menores a decimas de micras de segundo
* Last Update: Jule 19th, 2021
*/
'''
# ------------------------------------------------------------------------
# ------------------------------------------------------------------------

# # Librerias necesarias para el funcionamiento del programa //
# Similar al #include header_file en C/C++

import tkinter as tk
# Modulo grafico de Python
import tkinter.ttk as ttk
# Mejoras del modulo 'tkinter' con mejores estilos

import os
# Utilizacion de la Consola de Comandos de Windows
def clear_screen(): os.system('cls')
# Limpieza del interprete de Python
import subprocess
# Similar a la libreria 'os', con mayor rendimiento

import textwrap
# Acceso a comandos para manipulacion de textos
import math
# Acceso a funciones matematicas definidas en C

from tkinter import *
# Modulo grafico de Python
from tkinter.ttk import *
# Mejoras del modulo 'tkinter' con mejores estilos
from tkinter.messagebox import showinfo
# Ventana con mensaje (advertencia, error...)
from functools import partial
# Metodo para generar funciones con caracteristicas similares
from pathlib import Path
# Comprobacion de existencia de ficheros/directorios indicados

# ------------------------------------------------------------------

# # Creacion de la ventana principal del programa - 'root'
# (dimensiones, nombre, relaciones...)

width = 1024;
height = 576;
screen_resolution = str(width) + 'x' + str(height);

max_width = 1600;
max_height = 900;
max_screen_resolution = str(max_width) + 'x' + str(max_height);

min_width = 854;
min_height = 480;
min_screen_resolution = str(min_width) + 'x' + str(min_height);

half_width = width/2;
half_height = height/2;

root=tk.Tk()
# Creacion de la ventana
root.geometry(screen_resolution)
# Tamano de la ventana
root.minsize(min_width,min_height)
# Tamano minimo de la ventana
#root.maxsize(max_width,max_height)
# Tamano maximo de la ventana
root.resizable(True,True)
# Ajuste del tamano de la ventana
# -> True = Movil // False = Fijo
root.title('FPGA_MicroSim_Emulator')
# Cambio de nombre de la ventana
root.iconphoto(True,
```

```
tk.PhotoImage(file = 'LogoIconComillasColor.png'))
# Icono de la ventana

# ------------------------------------------------------------
# ------------------------------------------------------------

# # Creacion del estilo de los widgets del programa
# (visualizacion de la interfaz grafica)

s = ttk.Style()
# Creacion del estilo

def style_definition(s):
        # Resumen - Definicion del estilo

        s.theme_use('clam')
        # Tema del estilo por defecto

        s.configure(
                # Definicion del estilo del titulo
                # (Configuracion predeterminada)
                'label_Title.Label',
                anchor = CENTER,
                background = 'Orchid',
                foreground = 'PaleGoldenrod',
                font=('CMU_Serif_Roman', 20, 'bold'),
                width = 25)

        s.configure(
                # Definicion del estilo de la caja de texto numerica
                # (Configuracion estatica)
                'Election.TSpinbox',
                arrowsize = 15,
                selectbackground = 'gray35',
                selectforeground = 'azure',
                foreground = 'black')

        s.configure(
                # Definicion del estilo del texto explicativo en la seccion
                # de eleccion inicial (Configuracion predeterminada)
                'label_Explain_text.Label',
                anchor = CENTER,
                font=('CMU_Serif_Roman', 10, 'italic'),
                background = 'RoyalBlue1',
                foreground = 'lavender_blush',
                justify = CENTER)

        s.configure(
                # Definicion del estilo de los botones de eleccion iniciales
                # (Configuracion estatica)
                'Election.TButton',
                background = 'white',
                font = ('CMU_Serif_Roman', 12, 'bold'),
                foreground = 'black',
                highlightthickness = '20',
                padding = 16,
                width = 3.5)
        s.map(
                # Definicion del estilo de los botones de eleccion iniciales
                # (Configuracion dinamica)
                'Election.TButton',
                foreground = [('disabled', 'gold'),
                ('pressed', 'red2'),
                ('active', 'DeepSkyBlue2')],
                background = [('disabled', 'magenta'),
                ('pressed', '!focus', 'cyan2'),
                ('active', 'OliveDrab1')],
                highlightcolor = [('focus', 'OliveDrab1'),
                ('!focus', 'red2')],
                relief = [('pressed', 'sunken'),
                ('!pressed', 'raised')])

        s.configure(
                # Definicion del estilo de las casillas de eleccion iniciales
                # (Configuracion estatica)
                'Election.TCheckbutton',
                background = 'DeepSkyBlue3',
                font = ('CMU_Serif_Roman', 12, 'bold'),
                foreground = 'white',
                padding = 5)
        s.map(
                # Definicion del estilo de las casillas de eleccion iniciales
                # (Configuracion dinamica)
```

```
        'Election.TCheckbutton',
        foreground = [('disabled', 'gold'),
        ('pressed', 'red2'),
        ('active', 'DeepSkyBlue2')],
        background = [('disabled', 'magenta'),
        ('pressed', '!focus', 'cyan2'),
        ('active', 'OliveDrab1')],
        highlightcolor = [('focus', 'OliveDrab1'),
        ('!focus', 'red2')],
        relief = [('pressed', 'sunken'),
        ('!pressed', 'raised')])

s.configure(
        # Definicion del estilo del titulo
        # (Configuracion predeterminada)
        'asociation_txt.Label',
        anchor = CENTER,
        background = 'turquoise4',
        foreground = 'white',
        font=('CMU_Serif_Roman', 10, 'italic'),
        width = 25)

s.configure(
        # Definicion del estilo del titulo
        # (Configuracion predeterminada)
        'asociation_txt.TEntry',
        anchor = CENTER,
        background = 'white',
        foreground = 'turquoise4',
        font=('CMU_Serif_Roman', 10, 'italic'))

s.configure(
        # Definicion del estilo de los botones
        # (Configuracion estatica)
        'Emulator.TButton',
        background = 'white',
        font = ('CMU_Serif_Roman', 12, 'bold'),
        foreground = 'black',
        highlightthickness = '20',
        padding = 16,
        width = 3.5)
s.map(
        # Definicion del estilo de los botones
        # (Configuracion dinamica)
        'Emulator.TButton',
        foreground = [('disabled', 'gold'),
        ('pressed', 'red2'),
        ('active', 'DeepSkyBlue2')],
        background = [('disabled', 'magenta'),
        ('pressed', '!focus', 'cyan2'),
        ('active', 'OliveDrab1')],
        highlightcolor = [('focus', 'OliveDrab1'),
        ('!focus', 'red2')],
        relief = [('pressed', 'sunken'),
        ('!pressed', 'raised')])

s.configure(
        # Definicion del estilo de las casillas
        # de eleccion (Configuracion estatica)
        'Emulator.TCheckbutton',
        background = 'DeepSkyBlue3',
        font = ('CMU_Serif_Roman', 12, 'bold'),
        foreground = 'white',
        padding = 16)
s.map(
        # Definicion del estilo de las casillas
        # de eleccion (Configuracion dinamica)
        'Emulator.TCheckbutton',
        foreground = [('disabled', 'gold'),
        ('pressed', 'red2'),
        ('active', 'DeepSkyBlue2')],
        background = [('disabled', 'magenta'),
        ('pressed', '!focus', 'cyan2'),
        ('active', 'OliveDrab1')],
        highlightcolor = [('focus', 'OliveDrab1'),
        ('!focus', 'red2')],
        relief = [('pressed', 'sunken'),
        ('!pressed', 'raised')])

s.configure(
        # Definicion del estilo de los conversores
        # binario a 7 segmentos (Configuracion estatica)
        'Emulator.TLabel',
```

```
                background = 'black',
                font = ('CMU Serif Roman', 6, 'bold'),
                foreground = 'red',
                padding = 16)

style_definition(s);
# Definicion del estilo

# ------------------------------------------------------------
# ------------------------------------------------------------

# # Creacion de los marcos y ventanas de trabajo donde se
# proceden a colocar los distintos elementos ('widgets')

binA7seg_representations = {
# Asignacion con diccionario de los conversores 7 segmentos
        '1111110': ('###', '#  #', '#  #', '#  #', '###'),
        '0000001': ('###', '#  #', '#  #', '#  #', '###'),

        '1001111': ('  #', '  #', '  #', '  #', '  #'),
        '0110000': ('  #', '  #', '  #', '  #', '  #'),

        '0010010': ('###', '    #', '###', '#   ', '###'),
        '1101101': ('###', '    #', '###', '#   ', '###'),

        '0000110': ('###', '    #', '###', '    #', '###'),
        '1111001': ('###', '    #', '###', '    #', '###'),

        '1001100': ('#  #', '#  #', '###', '    #', '    #'),
        '0110011': ('#  #', '#  #', '###', '    #', '    #'),

        '0100100': ('###', '#   ', '###', '    #', '###'),
        '1011011': ('###', '#   ', '###', '    #', '###'),

        '0100000': ('###', '#   ', '###', '#  #', '###'),
        '1011111': ('###', '#   ', '###', '#  #', '###'),

        '0001111': ('###', '    #', '    #', '    #', '    #'),
        '1110000': ('###', '    #', '    #', '    #', '    #'),

        '0000000': ('###', '#  #', '###', '#  #', '###'),
        '1111111': ('###', '#  #', '###', '#  #', '###'),

        '0000100': ('###', '#  #', '###', '    #', '###'),
        '1111011': ('###', '#  #', '###', '    #', '###'),

        'default': ('- - -', '-   -', '- - -', '-   -', '- - -')}

def background_windows():
        # Resumen - Decoracion de fondo
        # ['CanvasGUI', 'CanvasGUI2', 'CanvasGUI3']
        globals()['canvasGUI'] = tk.Frame(
                # Creacion de la venta de fondo
                root,
                bg = 'LightSkyBlue1')
        globals()['canvasGUI'].place(
                # Colocacion de la venta de fondo
                relx = 0.5,
                rely = 0.5,
                relwidth = (width)/(width),
                relheight = (height)/(height),
                anchor = CENTER)

        globals()['canvasGUI2'] = tk.Frame(
                # Creacion de otra venta de fondo
                root,
                bg = 'LightSkyBlue2')
        globals()['canvasGUI2'].place(
                # Colocacion de otra venta de fondo
                relx = 0.5,
                rely = 0.5,
                relwidth = (width - (width - min_width)/2)/(width),
                relheight = (height - (height - min_height)/2)/(height),
                anchor = CENTER)

        globals()['canvasGUI3'] = tk.Frame(
                # Creacion de la venta donde se situaran
                #los widgets de emulacion
                root,
                bg = 'LightSkyBlue3')
        globals()['canvasGUI3'].place(
                # Colocacion de la venta donde se situaran
                # los widgets de emulacion
```

```
                relx = 0.5,
                rely = 0.5,
                relwidth = (min_width)/(width),
                relheight = (min_height)/(height),
                anchor = CENTER)

def bg_lift_windows():
        # Resumen - Elevacion de las ventanas del fondo a primer plano
        globals()['canvasGUI'].lift()
        globals()['canvasGUI2'].lift()
        globals()['canvasGUI3'].lift()
        globals()['label_Title'].lift()

# ------------------------------------------------------------

def asociation_window_display():
        # Resumen - Ventana de asociacion de nombres de variables con su
        # numero ['canvasGUI_Asociation', 'canvasGUI_Asociation_Canvas']
        globals()['canvasGUI_Asociation'] = tk.Frame(
                # Creacion de la venta donde se asocian los widgets de la
                # emulacion con sus variables
                root,
                bg = 'turquoise2')
        globals()['canvasGUI_Asociation'].place(
                # Colocacion de la venta donde se asocian los widgets de la
                # emulacion con sus variables
                in_ = root,
                relx = 0.500,
                rely = 0.500,
                relwidth = (min_width * 0.750)/(width),
                relheight = (min_height * 0.750)/(height),
                anchor = CENTER)

        globals()['canvasGUI_Asociation_Canvas'] = tk.Canvas(
                # Creacion de la venta donde se asocian los widgets de la
                # emulacion con sus variables
                canvasGUI_Asociation,
                bg = 'turquoise4', #'turquoise4',
                borderwidth = 0)
        globals()['canvasGUI_Asociation_Canvas'].place(
                # Colocacion de la venta donde se asocian los widgets de la
                # emulacion con sus variables
                in_ = canvasGUI_Asociation,
                relx = 0.250,
                rely = 0.500,
                relheight = 1.000,
                relwidth = 0.500,
                anchor = CENTER)

def election_window_display():
        # Resumen - Ventana de eleccion del numero de perfifericos
        # a utilizar ['canvasGUI_Election']
        globals()['canvasGUI_Election'] = tk.Frame(
                # Creacion de la venta donde se eligen los widgets de la
                # emulacion
                root,
                bg = 'turquoise3')
        globals()['canvasGUI_Election'].place(
                # Colocacion de la venta donde se eligen los widgets
                # de la emulacion
                relx = 0.5,
                rely = 0.5,
                relwidth = (min_width * 0.75)/(width),
                relheight = (min_height * 0.75)/(height),
                anchor = CENTER)

def title_window():
        # Resumen - Titulo del programa desarrollado ['label_Title']
        globals()['label_Title'] = ttk.Label(
                # Creacion del titulo del programa
                root,
                text='FPGA_MicroSim_Emulator',
                style = 'label_Title.Label')
        globals()['label_Title'].place(
                # Creacion del titulo del programa
                in_= canvasGUI,
                relx = (half_width)/(width),
                rely = (36)/(height),
                anchor = CENTER)

# ------------------------------------------------------------

def FPGA_widgets_frame():
```

```
# Resumen - Fondo de los perifericos de entrada y
# salida de la FPGA emulada

        globals()['canvasGUI_Outputs_Title'] = tk.Frame(
                # Creacion de la venta donde se situaran el titulo de
                # los widgets con senales de salida de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen1')
        globals()['Outputs_Title'] = ttk.Label(
                # Creacion del titulo del programa
                globals()['canvasGUI_Outputs_Title'],
                text = 'Outputs',
                style = 'label_Title.Label')
        globals()['canvasGUI_Outputs_Label'] = tk.Frame(
                # Creacion de la venta donde se situaran los conversores
                # 7 segmentos como senales de salida de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen3')
        globals()['canvasGUI_Outputs_GreenLed'] = tk.Frame(
                # Creacion de la venta donde se situaran los LEDs verdes
                # como senales de salida de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen3')
        globals()['canvasGUI_Outputs_RedLed'] = tk.Frame(
                # Creacion de la venta donde se situaran los LEDs rojos
                # como senales de salida de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen3')

        globals()['canvasGUI_Inputs_Title'] = tk.Frame(
                # Creacion de la venta donde se situaran el titulo
                # de los widgets con senales de entrada de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen2')
        globals()['Inputs_Title'] = ttk.Label(
                # Creacion del titulo del programa
                globals()['canvasGUI_Inputs_Title'],
                text = 'Inputs',
                style = 'label_Title.Label')
        globals()['canvasGUI_Inputs_Button'] = tk.Frame(
                # Creacion de la venta donde se situaran los botones
                # como senales de entrada de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen4')
        globals()['canvasGUI_Inputs_Switch'] = tk.Frame(
                # Creacion de la venta donde se situaran los interruptores
                # como senales de entrada de la emulacion
                canvasGUI3,
                bg = 'DarkOliveGreen4')

def FPGA_widgets_frame_display(
        # Resumen - Colocacion del fondo de los perifericos de
        # entrada y salida de la FPGA emulada
        number_buttons,
        number_switches,
        number_label,
        number_greenled,
        number_redled):

        globals()['canvasGUI_Outputs_Title'].place(
                # Colocacion de la venta donde se situaran el titulo de
                # los widgets con senales de salida de la emulacion
                relx = 0.500,
                rely = 0.075,
                relwidth = (min_width * 0.25)/(width),
                relheight = (min_height * 0.10)/(height),
                anchor = CENTER)
        globals()['Outputs_Title'].place(
                # Creacion del titulo del programa
                in_= globals()['canvasGUI_Outputs_Title'],
                relx = (half_width)/(width),
                rely = (half_width)/(width),
                anchor = CENTER)

        if number_label != 0:
                globals()['canvasGUI_Outputs_Label'].place(
                        # Colocacion de la venta donde se situaran los conversores
                        # 7 segmentos como senales de salida de la emulacion
                        relx = 0.275,
                        rely = 0.3375,
                        relwidth = (min_width)/(width)/2,
                        relheight = (min_height * 0.475)/(height),
                        anchor = CENTER)
```

```
        if number_greenled != 0:
                globals()['canvasGUI_Outputs_GreenLed'].place(
                        # Colocacion de la venta donde se situaran los LEDs verdes
                        # como senales de salida de la emulacion
                        relx = 0.725,
                        rely = 0.3375 + 1/9.25,
                        relwidth = (min_width)/(width)/2,
                        relheight = (min_height * 0.475)/(height)/2.25,
                        anchor = CENTER)

        if number_redled != 0:
                globals()['canvasGUI_Outputs_RedLed'].place(
                        # Colocacion de la venta donde se situaran los LEDs rojos
                        # como senales de salida de la emulacion
                        relx = 0.725,
                        rely = 0.3375 - 1/9.25,
                        relwidth = (min_width)/(width)/2,
                        relheight = (min_height * 0.475)/(height)/2.25,
                        anchor = CENTER)

        globals()['canvasGUI_Inputs_Title'].place(
                # Colocacion de la venta donde se situaran el titulo
                # de los widgets con senales de entrada de la emulacion
                relx = 0.500,
                rely = 0.600,
                relwidth = (min_width * 0.25)/(width),
                relheight = (min_height * 0.10)/(height),
                anchor = CENTER)
        globals()['Inputs_Title'].place(
                # Creacion del titulo del programa
                in_= globals()['canvasGUI_Inputs_Title'],
                relx = (half_width)/(width),
                rely = (half_width)/(width),
                anchor = CENTER)

        if number_buttons != 0:
                globals()['canvasGUI_Inputs_Button'].place(
                        # Colocacion de la venta donde se situaran los botones
                        # como senales de entrada de la emulacion
                        relx = 0.725,
                        rely = 0.810,
                        relwidth = (min_width)/(width)/2,
                        relheight = (min_height * 0.35)/(height),
                        anchor = CENTER)

        if number_switches != 0:
                globals()['canvasGUI_Inputs_Switch'].place(
                        # Colocacion de la venta donde se situaran los
                        # interruptores como senales de entrada de la emulacion
                        relx = 0.275,
                        rely = 0.810,
                        relwidth = (min_width)/(width)/2,
                        relheight = (min_height * 0.35)/(height),
                        anchor = CENTER)

        globals()['gtkwave_visual'] = ttk.Button (
                text = 'GTKWave',
                style = 'Election.TButton',
                command = lambda : CMD_Python_GTKWave_Init() )
        globals()['gtkwave_visual'].place(
                in_= canvasGUI3,
                relx = 0.750,
                rely = 0.600,
                anchor = CENTER)

def FPGA_widgets_frame_destroy():
        # Resumen - Eliminacion del fondo de los perifericos
        # de entrada y salida de la FPGA emulada
        globals()['canvasGUI_Outputs_Title'].destroy()

        if n_label_value.get() != 0:
                globals()['canvasGUI_Outputs_Label'].destroy()

        if n_greenled_value.get() != 0:
                globals()['canvasGUI_Outputs_GreenLed'].destroy()

        if n_redled_value.get() != 0:
                globals()['canvasGUI_Outputs_RedLed'].destroy()

        globals()['canvasGUI_Inputs_Title'].destroy()

        if n_buttons_value.get() != 0:
```

```
                globals()['canvasGUI_Inputs_Button'].destroy()

        if n_switches_value.get() != 0:
                globals()['canvasGUI_Inputs_Switch'].destroy()

        globals()['gtkwave_visual'].destroy()


# -----------------------------------------------------------

def initial_windows():
        # Resumen - Union de las funciones definidas previamente,
        # unicamente se muestran por pantalla las necesarias
        background_windows();
        FPGA_widgets_frame();
        election_window_display();
        title_window();

        initial_windows();

# -----------------------------------------------------------
# -----------------------------------------------------------

# # Creacion del menu del programa (gestion de archivos, edicion
# del programa, visualizacion de este, ventana de ayuda...)

def filemenu_definition(menu):
        filemenu = Menu( menu, tearoff = 0 )
        menu.add_cascade( label = 'Archivo', menu = filemenu)
        filemenu.add_command( label = 'Ventana_de_eleccion',
                command = lambda : election_widgets() )
        filemenu.add_command( label = 'Ventana_de_asociacion',
                command = lambda : name_variables() )
        filemenu.add_separator()
        filemenu.add_command( label = 'Exit...',
                command = root.destroy )

def editmenu_definition(menu):
        editmenu = Menu(menu, tearoff = 0)
        menu.add_cascade( label = 'Editar', menu=editmenu)
        editmenu.add_command( label = 'Numero_de_botones',
                command = lambda : election_widgets() )
        editmenu.add_command( label = 'Numero_de_interruptores',
                command = lambda : election_widgets() )
        editmenu.add_command( label = 'Numero_de_displays',
                command = lambda : election_widgets() )
        editmenu.add_command( label = 'Numero_de_LEDs_verdes',
                command = lambda : election_widgets() )
        editmenu.add_command( label = 'Numero_de_LEDs_rojos',
                command = lambda : election_widgets() )

def viewmenu_definition(menu):
        viewmenu = Menu(menu, tearoff = 0)
        menu.add_cascade(label = 'Ver', menu = viewmenu)
        viewmenu.add_command( label = 'GTKWave',
                command = lambda : CMD_Python_GTKWave_Init () )

# -----------------------------------------------------------

def contact_window():
        top = Toplevel(takefocus = True)

        top.title('Contact_Me')
        # Titulo de la ventana modal
        top.geometry('320x200')
        # Ajuste del tamano de la ventana modal
        top.resizable(False,False)
        # Reajuste del tamano de la ventana modal 'modal window'
        # -> True = Movil // False = Fijo

        contact_me_text = textwrap.dedent('''
                Para cualquier problema derivado del
                programa (bugs, glitches...), puede
                ponerse en contacto con el siguiente
                correo electronico:

                201706346@alu.comillas.edu

                Su peticion sera revisada con la
                mayor diligencia posible.
                Muchas gracias por su atencion!
                ''')
```

```
        label = tk.Label(
                top,
                anchor = CENTER,
                font=('CMU_Serif_Roman', 10),
                justify = CENTER,
                text = contact_me_text)

        label.place(
                in_= top,
                relx = 0.5,
                rely = 0.5,
                anchor = CENTER)

        top.lift(root)
        # Coloca la ventana modal al frente de la ventana principal
        top.transient(root)
        # Coloca la ventana modal al frente de la ventana principal
        # [Principal]
        top.grab_set()
        # Piratear los comandos de la ventana maestra
        # -> Clicks en la ventana principal son ignorados
        root.wait_window(top)
        # Pausa cualquier actividad en la ventana principal hasta el
        # cierre de esta

        top.mainloop()
        # Bucle activo de la ventana modal

def about_window():
        top = Toplevel(takefocus = True)

        top.title('About_FPGA_MicroSim_Emulator')
        # Titulo de la ventana modal
        top.geometry('384x240')
        # Ajuste del tamano de la ventana modal
        top.resizable(False,False)
        # Ajuste del tamano de la ventana modal 'modal window'
        # -> True = Movil // False = Fijo

        about_me_text = textwrap.dedent('''
                Emulador dinamico de VHDL mediante
                microsimulaciones.
                Bases del programa:
                - GHDL (compilador de lenguaje VHDL)
                - GTKWave (Visualizador de ondas)
                - Python (nexo e interfaz grafica)
                Proyecto de Fin de Grado, cuya autoria
                corresponde a D. Juan Perez Vilanova,
                dirigido por D. Fermin Zabalegui Sanz y
                D. Jose Daniel Munoz Frias
                Este obra esta bajo una licencia de Creative
                Commons Reconocimiento - CompartirIgual
                4.0 Internacional.
                ''')

        label = tk.Label(
                top,
                anchor = CENTER,
                font=('CMU_Serif_Roman', 10),
                justify = CENTER,
                text = about_me_text)

        label.place(
                in_= top,
                relx = 0.5,
                rely = 0.5,
                anchor = CENTER)

        top.lift(root)
        # Coloca la ventana modal al frente de la ventana principal
        top.transient(root)
        # Coloca la ventana modal al frente de la ventana principal
        # [Principal]
        top.grab_set()
        # Piratear los comandos de la ventana maestra
        # -> Clicks en la ventana principal son ignorados
        root.wait_window(top)
        # Pausa cualquier actividad en la ventana principal
        # hasta el cierre de esta

        top.mainloop()
        # Bucle activo de la ventana modal
```

```
def helpmenu_definition(menu):
        helpmenu = Menu(menu, tearoff = 0)
        menu.add_cascade(label = 'Ayuda', menu = helpmenu)
        helpmenu.add_command(label = 'Contact_Me',
                command = contact_window)
        helpmenu.add_command(label = 'About_FPGA_MicroSim_Emulator',
                command = about_window)

# ------------------------------------------------------

def menu_definition():
        menu_root = tk.Menu(root)
        root.config(menu = menu_root)
        filemenu_definition(menu_root)
        editmenu_definition(menu_root)
        viewmenu_definition(menu_root)
        helpmenu_definition(menu_root)

menu_definition()

# ------------------------------------------------------
# ------------------------------------------------------

# # Creacion de la clase generadora de N variables con nombre
# iterativo, y elementos de generacion de los perifericos

class Type_Widgets_Config:
        # Creacion de clase para el control de las
        # variables a modificar dentro de la emulacion
        def __init__(self, name_element, number_element):
                self.name_element = name_element
                self.number_element = number_element

        def name_iteration(self):
                type_string = [None] * (self.number_element)

        for i in range(self.number_element):
                if (i != self.number_element - 1):
                        type_string[self.number_element - (i + 1)] =
                                self.name_element + str(self.number_element - (i + 1))
                else:
                        type_string[0] = self.name_element + str(0)
                return type_string

def display_N_buttons(actual_frame, number_buttons):
        # Configuracion de botones

        buttons_name = Type_Widgets_Config("buttonB", number_buttons)
        globals()['buttons_name_var'] = buttons_name.name_iteration()

        buttons_value = Type_Widgets_Config("value_buttonB",
        number_buttons)
        globals()['buttons_value_var'] = buttons_value.name_iteration()

        buttons_text = Type_Widgets_Config("B", number_buttons)
        globals()['buttons_text_var'] = buttons_text.name_iteration()

        for i in range(number_buttons):

                globals()[buttons_name_var[i]] = StringVar()
                globals()[buttons_value_var[i]] = IntVar()

                if i < 5:
                        globals()[buttons_name_var[i]] = ttk.Button (
                                root,
                                text = buttons_text_var[i],
                                style = 'Emulator.TButton',
                                command = lambda : CMD_Python_Button_Execution_B(i) )
                        globals()[buttons_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i + 1),
                                rely = 1/3,
                                anchor = CENTER)
                else:
                        globals()[buttons_name_var[i]] = ttk.Button (
                                root,
                                text = buttons_text_var[i],
                                style = 'Emulator.TButton',
                                command = lambda : CMD_Python_Button_Execution_B(i) )
                        globals()[buttons_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i - 4),
                                rely = 2/3,
```

```
                                anchor = CENTER)

def display_N_switches(actual_frame, number_switches):
        # Configuracion de interruptores

        switches_name = Type_Widgets_Config("switchS", number_switches)
        globals()['switches_name_var'] = switches_name.name_iteration()

        switches_label = Type_Widgets_Config("S", number_switches)
        globals()['switches_label_var'] =
        switches_label.name_iteration()

        switches_value =
        Type_Widgets_Config("value_switchS", number_switches)
        globals()['switches_value_var'] =
        switches_value.name_iteration()

        for i in range(number_switches):

                globals()[switches_name_var[i]] = IntVar()
                globals()[switches_label_var[i]] = IntVar()
                globals()[switches_value_var[i]] = IntVar()

                if i < 5:
                        globals()[switches_name_var[i]] = ttk.Checkbutton (
                                root,
                                text = switches_label_var[i],
                                style = 'Emulator.TCheckbutton',
                                command = lambda : CMD_Python_Switch_Execution_S(i),
                                variable = globals()[switches_value_var[i]])
                        globals()[switches_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i + 1),
                                rely = 1/3,
                                anchor = CENTER)

                else:
                        globals()[switches_name_var[i]] = ttk.Checkbutton (
                                root,
                                text = switches_label_var[i],
                                style = 'Emulator.TCheckbutton',
                                command = lambda : CMD_Python_Switch_Execution_S(i),
                                variable = globals()[switches_value_var[i]])
                        globals()[switches_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i - 4),
                                rely = 2/3,
                                anchor = CENTER)

def display_N_label(actual_frame, number_label):
        # Configuracion de Etiquetas - Acompanamiento de interruptores

        label_name = Type_Widgets_Config("labelL", number_label)
        globals()['label_name_var'] = label_name.name_iteration()

        label_text = Type_Widgets_Config("L", number_label)
        globals()['label_text_var'] = label_text.name_iteration()

        for i in range(number_label):

                globals()[label_name_var[i]] = StringVar()
                globals()[label_text_var[i]] = StringVar()

                if i < 5:
                        globals()[label_name_var[i]] = ttk.Label (
                                root,
                                textvariable = globals()[label_text_var[i]],
                                style = 'Emulator.TLabel')
                        globals()[label_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i + 1),
                                rely = 1/4,
                                anchor = CENTER)

                else:
                        globals()[label_name_var[i]] = ttk.Label (
                                root,
                                textvariable = globals()[label_text_var[i]],
                                style = 'Emulator.TLabel')
                        globals()[label_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i - 4),
                                rely = 3/4,
```

```
                                anchor = CENTER)

                        globals()[label_text_var[i]]
                                .set("\n".join(binA7seg_representations['default']))

def display_N_greenled(actual_frame, number_greenled):
        # Configuracion LEDs verdes

        greenled_name =
        Type_Widgets_Config("greenledGL", number_greenled)
        globals()['greenled_name_var'] = greenled_name.name_iteration()

        for i in range(number_greenled):

        globals()[greenled_name_var[i]] = StringVar()

        if i < 5:
                globals()[greenled_name_var[i]] = tk.Canvas(
                        root,
                        width = 20,
                        height = 20,
                        bg = 'black')
                globals()[greenled_name_var[i]].place(
                        in_= actual_frame,
                        relx = 1/6 * (i + 1),
                        rely = 1/3,
                        anchor = CENTER)

        else:
                globals()[greenled_name_var[i]] = tk.Canvas(
                        root,
                        width = 20,
                        height = 20,
                        bg = 'black')
                globals()[greenled_name_var[i]].place(
                        in_= actual_frame,
                        relx = 1/6 * (i - 4),
                        rely = 2/3,
                        anchor = CENTER)
def greenLEDConfiguration(name_greenled, value):
        # Establecimiento del color -> LEDs verdes ON/OFF
        if (value == True):
                name_greenled.configure(bg = 'green')
        else:
                name_greenled.configure(bg = 'black')

def display_N_redled(actual_frame, number_redled):
        # Configuracion LEDs rojos

        redled_name = Type_Widgets_Config("redledRL", number_redled)
        globals()['redled_name_var'] = redled_name.name_iteration()

        for i in range(number_redled):

                globals()[redled_name_var[i]] = StringVar()

                if i < 5:
                        globals()[redled_name_var[i]] = tk.Canvas(
                                root,
                                width = 20,
                                height = 20,
                                bg = 'black')
                        globals()[redled_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i + 1),
                                rely = 1/3,
                                anchor = CENTER)

                else:
                        globals()[redled_name_var[i]] = tk.Canvas(
                                root,
                                width = 20,
                                height = 20,
                                bg = 'black')
                        globals()[redled_name_var[i]].place(
                                in_= actual_frame,
                                relx = 1/6 * (i - 4),
                                rely = 2/3,
                                anchor = CENTER)
def redLEDConfiguration(name_redled, value):
        # Establecimiento del color -> LEDs rojos ON/OFF
        if (value == True):
                name_redled.configure(bg = 'red')
```

```
        else:
                name_redled.configure(bg = 'black')

def FPGA_Emulator_Generator(
        # Creacion de funcion encargada de la creacion del emulador con
        # las caracteristicas seleccionadas
        number_buttons,
        number_switches,
        number_label,
        number_greenled,
        number_redled):

        if (number_buttons != 0):
                display_N_buttons(canvasGUI_Inputs_Button, number_buttons)

        if (number_switches != 0):
                display_N_switches(canvasGUI_Inputs_Switch, number_switches)

        if (number_label != 0):
                display_N_label(canvasGUI_Outputs_Label, number_label)

        if (number_greenled != 0):
                display_N_greenled(canvasGUI_Outputs_GreenLed, number_greenled)

        if (number_redled != 0):
                display_N_redled(canvasGUI_Outputs_RedLed, number_redled)

# ------------------------------------------------------------
# ------------------------------------------------------------

# # Lista de variables globales a utilizar de forma independiente

# Posicion de ventanas de eleccion, asociacion y emulacion
# -> Variables booleanas
# Numero de perifericos de entrada y salida (botones,
# interruptores, 7 segmentos, LEDs verdes, LEDs rojos...)
# Reset de perifericos de entrada y salida en la ventana de
# edicion -> Version predeterminada de casillas numericas

# Nombre de los archivos .vhd (script del programa y banco
# de pruebas), junto con sus directorios (sin los archivos)
# Lista con el nombre de los perifericos [entradas y salidas]
# -> Comparacion con el archivo .vcd generado
# Lista con el nombre de las variables del archivo .vcd
# -> Comparacion con entradas y salidas de la lista

bool_election_window = BooleanVar();
bool_asociation_window = BooleanVar();
bool_emulation_window = BooleanVar();

n_buttons_value = IntVar();
n_switches_value = IntVar();
n_label_value = IntVar();
n_greenled_value = IntVar();
n_redled_value = IntVar();

sw_buttons_reset = IntVar();
sw_switches_reset = IntVar();
sw_label_reset = IntVar();
sw_greenled_reset = IntVar();
sw_redled_reset = IntVar();

file_name_vhd = StringVar();
testbench_name_vhd = StringVar();
file_cd = StringVar();
testbench_cd = StringVar();
entity_testbench_name = StringVar();

list_buttons_vcd = [];
list_switches_vcd = [];
list_labels_vcd = [];
list_greenleds_vcd = [];
list_redleds_vcd = [];
last_line_entry_vhd = IntVar(value = 0);
reset_line = IntVar(value = 0);

list_buttons_var_vcd = [];
list_switches_var_vcd = [];
list_labels_var_vcd = [];
list_greenleds_var_vcd = [];
list_redleds_var_vcd = [];
last_line_entry_vcd = IntVar(value = 0);
```

```
def bool_mapped_windows(bool_election_window,
        bool_asociation_window, bool_emulation_window):
                try:
                        bool_election_window
                                .set(bool(canvasGUI_Election.winfo_ismapped()))
                except:
                        bool_election_window.set(False);

                try:
                        bool_asociation_window
                                .set(bool(canvasGUI_Asociation.winfo_ismapped()))
                except:
                        bool_asociation_window.set(False);

                try:
                        bool_emulation_window
                                .set(bool(canvasGUI_Outputs_Title.winfo_ismapped()))
                except:
                        bool_emulation_window.set(False);

bool_mapped_windows(bool_election_window,
        bool_asociation_window, bool_emulation_window)

# ------------------------------------------------

def FPGA_initiation(
        # Cambio de ventana entre eleccion de numero
        # de perifericos y asociacion de nombres
        buttons_list_var,
        switches_list_var,
        label_list_var,
        greenled_list_var,
        redled_list_var):

        error_value = BooleanVar(value = False);
        read_error = BooleanVar(value = False);

        if len(file_name_vhd.get()) == 0:
                error_value.set(True)

        if len(testbench_name_vhd.get()) == 0:
                error_value.set(True)

        if len(file_cd.get()) == 0:
                error_value.set(True)

        if len(testbench_cd.get()) == 0:
                error_value.set(True)

        if n_buttons_value.get() != 0:
                for i in range(n_redled_value.get()):
                        if len(globals()[buttons_list_var[i]].get()) == 0:
                                error_value.set(True)

        if n_switches_value.get() != 0:
                for i in range(n_switches_value.get()):
                        if len(globals()[switches_list_var[i]].get()) == 0:
                                error_value.set(True)

        if n_label_value.get() != 0:
                for i in range(n_label_value.get()):
                        if len(globals()[label_list_var[i]].get()) == 0:
                                error_value.set(True)

        if n_greenled_value.get() != 0:
                for i in range(n_greenled_value.get()):
                        if len(globals()[greenled_list_var[i]].get()) == 0:
                                error_value.set(True)

        if n_redled_value.get() != 0:
                for i in range(n_redled_value.get()):
                        if len(globals()[redled_list_var[i]].get()) == 0:
                                error_value.set(True)

# ----------------------------------------

        if (error_value.get() == False):

                if (Path(file_cd.get()).is_dir() == False):
                        read_error.set(True)

                if (Path(testbench_cd.get()).is_dir() == False):
                        read_error.set(True)
```

```
cd_with_file = file_cd.get() + '\\' + file_name_vhd.get();
cd_with_tb = testbench_cd.get() + '\\' + testbench_name_vhd.get();

if (Path(cd_with_file).is_file() == False):
        read_error.set(True)

if (Path(cd_with_tb).is_file() == False):
        read_error.set(True)

if (read_error.get() == False):

        bool_mapped_windows(bool_election_window,
                bool_asociation_window, bool_emulation_window);

        if (bool_election_window.get() == True):
                canvasGUI_Election.destroy()

        if (bool_asociation_window.get() == True):
                canvasGUI_Asociation.destroy()

        bool_mapped_windows(bool_election_window,
                bool_asociation_window, bool_emulation_window);

        bg_lift_windows();

        if n_buttons_value.get() != 0:
                for i in range(n_buttons_value.get()):
                        list_buttons_vcd.insert(i,
                                globals()[buttons_list_var[i]].get())

        if n_switches_value.get() != 0:
                for i in range(n_switches_value.get()):
                        list_switches_vcd.insert(i,
                                globals()[switches_list_var[i]].get())

        if n_label_value.get() != 0:
                for i in range(n_label_value.get()):
                        list_labels_vcd.insert(i,
                                globals()[label_list_var[i]].get() + '[6:0]')

        if n_greenled_value.get() != 0:
                for i in range(n_greenled_value.get()):
                        list_greenleds_vcd.insert(i,
                                globals()[greenled_list_var[i]].get())

        if n_redled_value.get() != 0:
                for i in range(n_redled_value.get()):
                        list_redleds_vcd.insert(i,
                                globals()[redled_list_var[i]].get())

        clear_screen();

        CMD_Python_Variable_Selection();

        FPGA_widgets_frame_display(
                n_buttons_value.get(),
                n_switches_value.get(),
                n_label_value.get(),
                n_greenled_value.get(),
                n_redled_value.get());

        FPGA_Emulator_Generator(
                n_buttons_value.get(),
                n_switches_value.get(),
                n_label_value.get(),
                n_greenled_value.get(),
                n_redled_value.get())

else:
        print('\n Compruebe que esten correctas las siguientes casillas:')

        if (Path(file_cd.get()).is_dir() == False):
                print('    Directorio del archivo VHDL')

        if (Path(testbench_cd.get()).is_dir() == False):
                print('    Directorio del archivo del banco de pruebas original')

        if (Path(cd_with_file).is_file() == False):
                print('    Nombre del archivo VHDL')

        if (Path(cd_with_tb).is_file() == False):
                print('    Nombre del archivo del banco de pruebas original')
```

```
        else:

                print('\n Debe rellenar las siguientes casillas:')

                if len(file_name_vhd.get()) == 0:
                        print('    Nombre del archivo VHDL')

                if len(testbench_name_vhd.get()) == 0:
                        print('    Nombre del archivo del banco de pruebas original')

                if len(file_cd.get()) == 0:
                        print('    Directorio del archivo VHDL')

                if len(testbench_cd.get()) == 0:
                        print('    Directorio del archivo del banco de pruebas original')

                if n_buttons_value.get() != 0:
                        for i in range(n_buttons_value.get()):
                                if len(globals()[buttons_list_var[i]].get()) == 0:
                                        print('    Nombre del boton B' + str(i))

                if n_switches_value.get() != 0:
                        for i in range(n_switches_value.get()):
                                if len(globals()[switches_list_var[i]].get()) == 0:
                                        print('    Nombre del interruptor S' + str(i))

                if n_label_value.get() != 0:
                        for i in range(n_label_value.get()):
                                if len(globals()[label_list_var[i]].get()) == 0:
                                        print('    Nombre del conversor 7 segmentos L' + str(i))

                if n_greenled_value.get() != 0:
                        for i in range(n_greenled_value.get()):
                                if len(globals()[greenled_list_var[i]].get()) == 0:
                                        print('    Nombre del LED verde GL' + str(i))

                if n_redled_value.get() != 0:
                        for i in range(n_redled_value.get()):
                                if len(globals()[redled_list_var[i]].get()) == 0:
                                        print('    Nombre del LED rojo RL' + str(i))

# ------------------------------------------------------------

def name_asociation():
        # Cambio de ventana entre eleccion de numero de
        # perifericos y asociacion de nombres

        bool_mapped_windows(bool_election_window,
                bool_asociation_window, bool_emulation_window);

        if bool_election_window.get() == True:
                canvasGUI_Election.destroy()
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        if bool_asociation_window.get() == False:
                asociation_window_display()
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        if bool_emulation_window.get() == True:
                FPGA_widgets_frame_destroy();
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        # ------------------------------------------------

        canvasGUI_Asociation_Canvas.update()

        globals()['canvasGUI_AsociationHalf'] = tk.Frame(
                # Creacion de la venta donde se asocian los widgets de
                # la emulacion con sus variables
                canvasGUI_Asociation_Canvas,
                bg = 'turquoise4',
                height = canvasGUI_Asociation_Canvas.winfo_height() - 4,
                width = canvasGUI_Asociation_Canvas.winfo_width() - 4)
        globals()['canvasGUI_Asociation_Canvas'].create_window(
                (2,2),
                anchor = 'nw',
                window = canvasGUI_AsociationHalf)

        globals()['scrollbar_frame_vert'] = tk.Scrollbar(
```

```
        canvasGUI_Asociation_Canvas,
        orient = VERTICAL,
        command = canvasGUI_Asociation_Canvas.yview)
globals()['canvasGUI_Asociation_Canvas'].config(
        yscrollcommand = scrollbar_frame_vert.set )
globals()['scrollbar_frame_vert'].place(
        in_ = canvasGUI_Asociation_Canvas,
        relx = 1,
        rely = 0,
        relheight = 1,
        anchor = 'ne')

globals()['scrollbar_frame_horz'] = tk.Scrollbar(
        canvasGUI_Asociation_Canvas,
        orient = HORIZONTAL,
        command = canvasGUI_Asociation_Canvas.xview)
globals()['canvasGUI_Asociation_Canvas'].config(
        xscrollcommand = scrollbar_frame_horz.set )
globals()['scrollbar_frame_horz'].place(
        in_ = canvasGUI_Asociation_Canvas,
        relx = 0,
        rely = 1,
        relwidth = 1,
        anchor = 'sw')

# ------------------------------------------------

vhdl_file_label_txt =
textwrap.dedent('''  - - - - - - - - - - - - - - - -
Archivo VHDL:
- - - - - - - - - - - - - - - - ''')

textbench_file_label_txt =
textwrap.dedent('''  - - - - - - - - - - - - - - - -
TestBench VHDL:
- - - - - - - - - - - - - - - - ''')

vhdl_cd_label_txt =
textwrap.dedent('''  - - - - - - - - - - - - - - - -
Directorio VHDL:
- - - - - - - - - - - - - - - - ''')

textbench_cd_label_txt =
textwrap.dedent('''  - - - - - - - - - - - - - - - -
Directorio TB:
- - - - - - - - - - - - - - - - ''')

entity_testbench_label_txt =
textwrap.dedent('''  - - - - - - - - - - - - - - - -
Nombre Entidad TB:
- - - - - - - - - - - - - - - - ''')

vhdl_file_label = ttk.Label(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.Label',
        text = vhdl_file_label_txt)
vhdl_file_label.grid(row = 0, column = 0, sticky = '_')
vhdl_file_entry = ttk.Entry(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.TEntry',
        textvariable = file_name_vhd,
        width = 100)
vhdl_file_entry.grid(row = 0, column = 1, sticky = '_')

textbench_file_label = ttk.Label(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.Label',
        text = textbench_file_label_txt)
textbench_file_label.grid(row = 1, column = 0, sticky = '_')
textbench_file_entry = ttk.Entry(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.TEntry',
        textvariable = testbench_name_vhd,
        width = 100)
textbench_file_entry.grid(row = 1, column = 1, sticky = '_')

vhdl_cd_label = ttk.Label(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.Label',
        text = vhdl_cd_label_txt)
vhdl_cd_label.grid(row = 2, column = 0, sticky = '_')
vhdl_cd_entry = ttk.Entry(
        canvasGUI_AsociationHalf,
```

```
        style = 'asociation_txt.TEntry',
        textvariable = file_cd ,
        width = 100)
vhdl_cd_entry.grid(row = 2, column = 1, sticky = '␣')

textbench_cd_label = ttk.Label(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.Label',
        text = textbench_cd_label_txt)
textbench_cd_label.grid(row = 3, column = 0, sticky = '␣')
textbench_cd_entry = ttk.Entry(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.TEntry',
        textvariable = testbench_cd ,
        width = 100)
textbench_cd_entry.grid(row = 3, column = 1, sticky = '␣')

entity_testbench_label = ttk.Label(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.Label',
        text = entity_testbench_label_txt)
entity_testbench_label.grid(row = 4, column = 0, sticky = '␣')
entity_testbench_entry = ttk.Entry(
        canvasGUI_AsociationHalf,
        style = 'asociation_txt.TEntry',
        textvariable = entity_testbench_name ,
        width = 100)
entity_testbench_entry.grid(row = 4, column = 1, sticky = '␣')

# ----------------------------------------------------

carry = IntVar(); carry.set(0);
elements = IntVar(); elements.set(0);

buttons_list = Type_Widgets_Config(
        "Variable_Button_B",
        n_buttons_value.get())
buttons_list_var = buttons_list.name_iteration()

switches_list = Type_Widgets_Config(
        "Variable_Switch_S",
        n_switches_value.get())
switches_list_var = switches_list.name_iteration()

label_list = Type_Widgets_Config(
        "Variable_7Segmentos_L",
        n_label_value.get())
label_list_var = label_list.name_iteration()

greenled_list = Type_Widgets_Config(
        "Variable_GreenLed_GL",
        n_greenled_value.get())
greenled_list_var = greenled_list.name_iteration()

redled_list = Type_Widgets_Config(
        "Variable_RedLed_RL",
        n_redled_value.get())
redled_list_var = redled_list.name_iteration()

if n_buttons_value.get() != 0:
        carry.set(int(carry.get() + 1))
        bottons_label_txt = textwrap.dedent('''
        Botones:
        - - - - - - - - - - - - - - ''')
        bottons_label = ttk.Label(
                canvasGUI_AsociationHalf,
                style = 'asociation_txt.Label',
                text = bottons_label_txt)
        bottons_label.grid(row = 4 + carry.get() + elements.get(),
        column = 0, sticky = 'ne')

        buttons_name = Type_Widgets_Config(
                "Button_B",
                n_buttons_value.get())
        buttons_name_var = buttons_name.name_iteration()

        for i in range(n_buttons_value.get()):
                globals()[buttons_list_var[i]] = StringVar();
                buttons_var_label = ttk.Label(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.Label',
                        text = buttons_name_var[i])
                buttons_var_label.grid(row = 4 + carry.get() + elements.get() + 1 + i,
```

```
                        column = 0, sticky = 'ne')
                buttons_var_entry = ttk.Entry(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.TEntry',
                        textvariable =  globals()[buttons_list_var[i]])
                buttons_var_entry.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                         column = 1, sticky = 'nw')

        elements.set(int(elements.get() + n_buttons_value.get()))

if n_switches_value.get() != 0:
        carry.set(int(carry.get() + 1))
        switches_label_txt = textwrap.dedent('''
        Interruptores:
        - - - - - - - - - - - - - - - ''')
        switches_label = ttk.Label(
                canvasGUI_AsociationHalf,
                style = 'asociation_txt.Label',
                text = switches_label_txt)
        switches_label.grid(row = 4 + carry.get() + elements.get(),
                column = 0, sticky = 'ne')

        switches_name = Type_Widgets_Config(
                "Switch_S",
                n_switches_value.get())
        switches_name_var = switches_name.name_iteration()

        for i in range(n_switches_value.get()):
        globals()[switches_list_var[i]] = StringVar();
                switches_var_label = ttk.Label(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.Label',
                        text = switches_name_var[i])
                switches_var_label.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                        column = 0, sticky = 'ne')
                switches_var_entry = ttk.Entry(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.TEntry',
                        textvariable = globals()[switches_list_var[i]])
                switches_var_entry.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                        column = 1, sticky = 'nw')

        elements.set(int(elements.get() + n_switches_value.get()))

if n_label_value.get() != 0:
        carry.set(int(carry.get() + 1))
        label_label_txt = textwrap.dedent('''
        7 Segmentos:
        - - - - - - - - - - - - - - - ''')
        label_label = ttk.Label(
                canvasGUI_AsociationHalf,
                style = 'asociation_txt.Label',
                text = label_label_txt)
        label_label.grid(row = 4 + carry.get() + elements.get(),
                column = 0, sticky = 'ne')

        label_name = Type_Widgets_Config(
                "7_Segmentos_L",
                n_label_value.get())
        label_name_var = label_name.name_iteration()

        for i in range(n_label_value.get()):
                globals()[label_list_var[i]] = StringVar();
                label_var_label = ttk.Label(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.Label',
                        text = label_name_var[i])
                label_var_label.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                        column = 0, sticky = 'ne')
                label_var_entry = ttk.Entry(
                        canvasGUI_AsociationHalf,
                        style = 'asociation_txt.TEntry',
                        textvariable = globals()[label_list_var[i]])
                label_var_entry.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                        column = 1, sticky = 'nw')

        elements.set(int(elements.get() + n_label_value.get()))

if n_greenled_value.get() != 0:
        carry.set(int(carry.get() + 1))
        greenled_label_txt = textwrap.dedent('''
        LEDs verdes:
        - - - - - - - - - - - - - - - ''')
```

```
            greenled_label = ttk.Label(
                    canvasGUI_AsociationHalf,
                    style = 'asociation_txt.Label',
                    text = greenled_label_txt)
            greenled_label.grid(row = 4 + carry.get() + elements.get(),
                    column = 0, sticky = 'ne')

            greenled_name = Type_Widgets_Config(
                    "GreenLed_GL",
                    n_greenled_value.get())
            greenled_name_var = greenled_name.name_iteration()

            for i in range(n_greenled_value.get()):
                    globals()[greenled_list_var[i]] = StringVar();
                    greenled_var_label = ttk.Label(
                            canvasGUI_AsociationHalf,
                            style = 'asociation_txt.Label',
                            text = greenled_name_var[i])
                    greenled_var_label.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                            column = 0, sticky = 'ne')
                    greenled_var_entry = ttk.Entry(
                            canvasGUI_AsociationHalf,
                            style = 'asociation_txt.TEntry',
                            textvariable = globals()[greenled_list_var[i]])
                    greenled_var_entry.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                            column = 1, sticky = 'nw')

                    elements.set(int(elements.get() + n_greenled_value.get()))

    if n_redled_value.get() != 0:
            carry.set(int(carry.get() + 1))
            redled_label_txt = textwrap.dedent('''
            LEDs rojos:
            _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ''')
            redled_label = ttk.Label(
            canvasGUI_AsociationHalf,
            style = 'asociation_txt.Label',
            text = redled_label_txt)
            redled_label.grid(row = 4 + carry.get() + elements.get(),
            column = 0, sticky = 'ne')

            redled_name = Type_Widgets_Config("RedLed_RL",
            n_redled_value.get())
            redled_name_var = redled_name.name_iteration()

            for i in range(n_redled_value.get()):
                    globals()[redled_list_var[i]] = StringVar();
                    redled_var_label = ttk.Label(
                            canvasGUI_AsociationHalf,
                            style = 'asociation_txt.Label',
                            text = redled_name_var[i])
                    redled_var_label.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                            column = 0, sticky = 'ne')
                    redled_var_entry = ttk.Entry(
                            canvasGUI_AsociationHalf,
                            style = 'asociation_txt.TEntry',
                            textvariable = globals()[redled_list_var[i]])
                    redled_var_entry.grid(row = 4 + carry.get() + elements.get() + 1 + i,
                            column = 1, sticky = 'nw')

            elements.set(int(elements.get() + n_redled_value.get()))

    horz_margin = ttk.Label(
            canvasGUI_AsociationHalf,
            style = 'asociation_txt.Label',
            text = '\n ')
    horz_margin.grid(row = 4 + carry.get() + elements.get() + 1,
            column = 0, sticky = ' ')

    ver_margin = ttk.Label(
            canvasGUI_AsociationHalf,
            style = 'asociation_txt.Label',
            text = ' ')
    ver_margin.grid(row = 0, column = 2, sticky = 'nw')

    # ------------------------------------------------------------

    explain_asociation_text = textwrap.dedent('''
    Para iniciar la emulacion de la FPGA,
    sera necesario introducir los nombres
    de las variables de cada uno de los
    perifericos de entrada y salida.
    Para regresar a la pantalla anterior,
```

```
        puede hacer click en el boton situado
        en la parte inferior con una flecha.
        Una vez este seguro del numero
        de perifericos y esten debidamente
        nombradas sus variables equivalentes
        del archivo '.vhd', haga click en el
        boton 'Generate FPGA.'
        ''')

        explain_asociation = ttk.Label(
                style = 'label_Explain_text.Label',
                text = explain_asociation_text)
        explain_asociation.place(
                in_= canvasGUI_Asociation,
                relx = 0.750,
                rely = 19/48,
                relwidth = 7/16,
                relheight = 3/4,
                anchor = CENTER)

        # ------------------------------------------------------------

        back_election = ttk.Button (
                text = '<',
                style = 'Election.TButton',
                command = lambda : election_widgets() )
        back_election.place(
                in_= canvasGUI_Asociation,
                relx = 0.625,
                rely = 0.875,
                anchor = CENTER)

        # ------------------------------------------------------------

        FPGA_generate = ttk.Button (
                text = 'Generate_FPGA',
                style = 'Election.TButton',
                command = lambda : FPGA_initiation(
                buttons_list_var,
                switches_list_var,
                label_list_var,
                greenled_list_var,
                redled_list_var) )
        FPGA_generate.place(
                in_= canvasGUI_Asociation,
                relx = 0.825,
                rely = 0.875,
                anchor = CENTER)

        # ------------------------------------------------------------
        # ------------------------------------------------------------

def name_variables():
        # Cambio de ventana entre eleccion de
        # numero de perifericos y asociacion de nombres

        error_value = IntVar(value = 0);

        try:
                n_buttons_value.get()
        except:
                print('Debe_introducir_un_numero_de_botones\n');
                error_value.set(1);

        try:
                n_switches_value.get()
        except:
                print('Debe_introducir_un_numero_de_interruptores\n')
                error_value.set(1);

        try:
                n_label_value.get()
        except:
                print('Debe_introducir_un_numero_de_conversores_de_7_segmentos\n')
                error_value.set(1);

        try:
                n_greenled_value.get()
        except:
                print('Debe_introducir_un_numero_de_LEDs_verdes\n')
                error_value.set(1);

        try:
```

```
                n_redled_value.get()
        except:
                print('Debe introducir un numero de LEDs rojos\n')
                error_value.set(1);

        if (error_value.get() == False):

                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

                if bool_election_window.get() == True:
                canvasGUI_Election.destroy()
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        clear_screen();

        name_asociation();

        # ------------------------------------------------------------
        # ------------------------------------------------------------

def name_display_func(
        # Version predeterminada de las casillas numericas
        # por medio de las casillas de eleccion
        n_buttons,
        n_switches,
        n_label,
        n_greenled,
        n_redled):

        if (sw_buttons_reset.get() == True):
        # Si la marca de reset al lado de la casilla de 'N de botones',
        # se encuentra marcada en el momento de llamar a la funcion
                n_buttons.set('N de botones')
        # 'name_display_func', la casilla de 'N de botones' vuelve al
        # estado predeterminado. Se hace igual con el resto de perifericos

        if (sw_switches_reset.get() == True):
                n_switches.set('N de interruptores');

        if (sw_label_reset.get() == True):
                n_label.set('N de 7 segmentos');

        if (sw_greenled_reset.get() == True):
                n_greenled.set('N de LEDs verdes');

        if (sw_redled_reset.get() == True):
                n_redled.set('N de LEDs rojos');

# ------------------------------------------------------------
# ------------------------------------------------------------

def election_widgets():
        # Cambio de ventana entre eleccion de numero de
        # perifericos y asociacion de nombres

        bool_mapped_windows(bool_election_window,
                bool_asociation_window, bool_emulation_window);

        if bool_emulation_window.get() == True:
                FPGA_widgets_frame_destroy();
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        if bool_asociation_window.get() == True:
                canvasGUI_Asociation.destroy()
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        if bool_election_window.get() == False:
                election_window_display()
                bool_mapped_windows(bool_election_window,
                        bool_asociation_window, bool_emulation_window);

        n_buttons = ttk.Spinbox(
                from_ = 0,
                to = 10,
                increment = 1,
                state="readonly",
                style = 'Election.TSpinbox',
                textvariable = n_buttons_value)
        n_buttons.set('N de botones')
```

```
n_buttons.place(
        in_= canvasGUI_Election,
        relx = 0.275,
        rely = 0.150,
        anchor = CENTER)

n_switches = ttk.Spinbox(
        from_ = 0,
        to = 10,
        increment = 1,
        state="readonly",
        style = 'Election.TSpinbox',
        textvariable = n_switches_value);
n_switches.set('N_de_interruptores');
n_switches.place(
        in_= canvasGUI_Election,
        relx = 0.275,
        rely = 0.325,
        anchor = CENTER)

n_label = ttk.Spinbox(
        from_ = 0,
        to = 10,
        increment = 1,
        state="readonly",
        style = 'Election.TSpinbox',
        textvariable = n_label_value);
n_label.set('N_de_7_segmentos');
n_label.place(
        in_= canvasGUI_Election,
        relx = 0.275,
        rely = 0.500,
        anchor = CENTER)

n_greenled = ttk.Spinbox(
        from_ = 0,
        to = 10,
        increment = 1,
        state="readonly",
        style = 'Election.TSpinbox',
        textvariable = n_greenled_value);
n_greenled.set('N_de_LEDs_verdes');
n_greenled.place(
        in_= canvasGUI_Election,
        relx = 0.275,
        rely = 0.675,
        anchor = CENTER)

n_redled = ttk.Spinbox(
        from_ = 0,
        to = 10,
        increment = 1,
        state="readonly",
        style = 'Election.TSpinbox',
        textvariable = n_redled_value);
n_redled.set('N_de_LEDs_rojos');
n_redled.place(
        in_= canvasGUI_Election,
        relx = 0.275,
        rely = 0.850,
        anchor = CENTER)

# ------------------------------------------------------

sw_buttons = ttk.Checkbutton(
        state="readonly",
        style = 'Election.TCheckbutton',
        text = 'Reset',
        variable = sw_buttons_reset)
sw_buttons.place(
        in_= canvasGUI_Election,
        relx = 1/(math.sqrt(2)*10),
        rely = 0.150,
        anchor = CENTER)

sw_switches = ttk.Checkbutton(
        state="readonly",
        style = 'Election.TCheckbutton',
        text = 'Reset',
        variable = sw_switches_reset)
sw_switches.place(
        in_= canvasGUI_Election,
        relx = 1/(math.sqrt(2)*10),
```

```
        rely = 0.325,
        anchor = CENTER)

sw_label = ttk.Checkbutton(
        state="readonly",
        style = 'Election.TCheckbutton',
        text = 'Reset',
        variable = sw_label_reset)
sw_label.place(
        in_= canvasGUI_Election,
        relx = 1/(math.sqrt(2)*10),
        rely = 0.500,
        anchor = CENTER)

sw_greenled = ttk.Checkbutton(
        state="readonly",
        style = 'Election.TCheckbutton',
        text = 'Reset',
        variable = sw_greenled_reset)
sw_greenled.place(
        in_= canvasGUI_Election,
        relx = 1/(math.sqrt(2)*10),
        rely = 0.675,
        anchor = CENTER)

sw_redled = ttk.Checkbutton(
        state="readonly",
        style = 'Election.TCheckbutton',
        text = 'Reset',
        variable = sw_redled_reset)
sw_redled.place(
        in_= canvasGUI_Election,
        relx = 1/(math.sqrt(2)*10),
        rely = 0.850,
        anchor = CENTER)

# ------------------------------------------------------------

explain_election_text = textwrap.dedent('''
Para iniciar la emulacion de la FPGA, sera necesario
introducir los valores de cada uno de los perifericos
de entrada y salida.
Se pueden utilizar hasta 10 unidades de cada periferico.
Para resetear los valores de entrada, debe marcar las
casillas de los elementos a restablecer y posteriormente,
hacer click en el boton 'Reset Spinboxes'. Una vez haya
seleccionado todos los elementos, haga click en el boton
'Naming vars.'
''')
explain_election = ttk.Label(
        canvasGUI_Election,
        style = 'label_Explain_text.Label',
        text = explain_election_text)
explain_election.place(
        in_= canvasGUI_Election,
        relx = 0.700,
        rely = 1/3 + 1/16,
        relwidth = 1/2 + 1/32,
        relheight = 9/16,
        anchor = CENTER)

# ------------------------------------------------------------

name_display = ttk.Button (
        text = 'Reset_Spinboxes',
        style = 'Election.TButton',
        command = lambda : name_display_func(
        n_buttons,
        n_switches,
        n_label,
        n_greenled,
        n_redled))
name_display.place(
        in_= canvasGUI_Election,
        relx = 0.575,
        rely = 0.750 + 1/16,
        anchor = CENTER)

FPGA_generate = ttk.Button (
        text = 'Naming_vars.',
        style = 'Election.TButton',
        command = lambda : name_variables())
FPGA_generate.place(
```

```
                in_= canvasGUI_Election,
                relx = 0.825,
                rely = 0.750 + 1/16,
                anchor = CENTER)

# ------------------------------------------------------

election_widgets();

# ------------------------------------------------------
# ------------------------------------------------------

def resize(event):
        s.configure(
                # Ajuste del tamano del titulo
                'label_Title.Label',
                font=('CMU_Serif_Roman',
                int(20 * (root.winfo_width())/(width)), 'bold'))
        s.configure(
                # Ajuste del tamano de la explicacion
                'label_Explain_text.Label',
                font = ('CMU_Serif_Roman',
                int(10 * (root.winfo_width())/(width)), 'italic'))
        s.configure(
                # Ajuste del tamano de los botones iniciales
                'Election.TButton',
                font = ('CMU_Serif_Roman',
                int(10 * (root.winfo_width())/(width)), 'bold'),
                padding = int(16 * (root.winfo_width())/(width)))
        s.configure(
                # Ajuste del tamano de las casillas de eleccion numericas
                'Election.TSpinbox',
                arrowsize = int(15 * (root.winfo_width())/(width)))
        # Ajuste del tamano de las casillas de texto numericas
        s.configure(
                # Ajuste del tamano de las casillas de eleccion
                iniciales para resetear
                'Election.TCheckbutton',
                background = 'DeepSkyBlue3',
                font = ('CMU_Serif_Roman',
                int(12 * (root.winfo_width())/(width)), 'bold'),
                foreground = 'white',
                padding = int(5 * (root.winfo_width())/(width)))
        s.configure(
        # Ajuste del tamano de los botones
                'Emulator.TButton',
                background = 'white',
                font = ('CMU_Serif_Roman', int(12 * (root.winfo_width())/(max_width)), 'bold'),
                foreground = 'black',
                highlightthickness = '20',
                padding = int(16 * (root.winfo_width())/(max_width)),
                width = int(3.5 * (root.winfo_width())/(max_width)))
        s.configure(
                # Ajuste del tamano de los interruptores
                'Emulator.TCheckbutton',
                background = 'DeepSkyBlue3',
                font = ('CMU_Serif_Roman', int(12 * (root.winfo_width())/(max_width)), 'bold'),
                foreground = 'white',
                padding = int(16 * (root.winfo_width())/(max_width)))
        if (bool_asociation_window.get() == True):
                globals()['canvasGUI_AsociationHalf'].config(
                height = (canvasGUI_Asociation_Canvas.winfo_height() - 4),
                width = (canvasGUI_Asociation_Canvas.winfo_width() - 4))

        try:
                canvasGUI_Asociation_Canvas.configure(
                        scrollregion = canvasGUI_Asociation_Canvas.bbox('all') )
        except:
                pass

# ------------------------------------------------------
# ------------------------------------------------------

def CMD_Python_Variable_Selection ():
# # Funcion de comunicacion inicial para la extraccion de datos
# # necesarios para la emulacion. Llamada una unica vez en la
# # transicion de la ventana de asociacion y emulacion

        file_name = (file_name_vhd.get()).rsplit('.')[0]
        # Nombre del archivo VHDL sin extension
        testbench_name = (testbench_name_vhd.get()).rsplit('.')[0]
        # Nombre del archivo TestBench sin extension
        testbench_vcd_mod = testbench_name + '_mod.vcd'
```

```
# Nombre del futuro archivo .vcd con extension
file_vhd_cd = (file_cd.get()) + '\\' + (file_name_vhd.get());
# Directorio del archivo VHDL incluido
testbench_vhd_cd = (testbench_cd.get()) + '\\' +
(testbench_name_vhd.get());
# Directorio del archivo TestBench incluido
testbench_vcd_mod_cd = (testbench_cd.get()) + '\\' +
testbench_vcd_mod;
# Directorio del archivo .vcd incluido

ghdl_file_scan = 'ghdl -s ' + (file_name_vhd.get());
ghdl_file_analize = 'ghdl -a ' + (file_name_vhd.get());
ghdl_file_execute = 'ghdl -e ' + file_name;
ghdl_testbench_scan = 'ghdl -s ' + (testbench_name_vhd.get());
ghdl_testbench_analize = 'ghdl -a ' + (testbench_name_vhd.get());
ghdl_testbench_execute = 'ghdl -e ' + testbench_name;
ghdl_testbench_run = 'ghdl -r ' + testbench_name + ' --vcd=' +
testbench_vcd_mod;

initial_file_commands = [
        ghdl_file_scan,
        ghdl_file_analize,
        ghdl_file_execute]
        # Analisis, elaboracion y sintesis del archivo VHDL

initial_testbench_commands = [
        ghdl_testbench_scan,
        ghdl_testbench_analize,
        ghdl_testbench_execute,
        ghdl_testbench_run]
        # Analisis, elaboracion y sintesis del archivo TestBench

for command in initial_file_commands:
        subprocess.run(command, shell = True,
                cwd = (file_cd.get()))

for command in initial_testbench_commands:
        subprocess.run(command, shell = True,
                cwd = (testbench_cd.get()))

# Busqueda de las variables a interpretar dentro del archivo .vcd

entity_text = '$scope module ' + (entity_testbench_name.get()) + ' $end';

line_number = 0; entity_number = 0; read_value = 0; check = 0;
list_variables_txt = []; list_variables = [];

with open(testbench_vhd_cd,'rt') as vhd:
        vhd_txt = vhd.readlines()

with open(testbench_vhd_cd,'rt') as vhd:
        for line in vhd:
                line_number += 1
                if "reset_n <= '1';" in line:
                        reset_line.set(line_number);
                        last_line_entry_vhd.set(line_number);

line_number = 0; entity_number = 0; read_value = 0; check = 0;
list_variables_txt = []; list_variables = [];

with open(testbench_vcd_mod_cd,'rt') as vcd:
        for line in vcd:
        line_number += 1

        if entity_text in line:
        entity_number = line_number

        if (('$scope module' in line) and (line_number >entity_number > 0) and (check == 0)):
                check = 1

        if (('$var reg' in line) and (line_number > entity_number)
        and (check == 0)):

                list_variables_txt.append(line.rstrip())
                list_variables.append(
                list_variables_txt[line_number - entity_number - 1].rsplit(' ')[3])

                if len(list_buttons_vcd) != 0:
                        for i in range(len(list_buttons_vcd)):
                                if list_variables_txt[line_number - entity_number - 1].rsplit(' ')[4] ==
                                        list_buttons_vcd[i]:
                                        list_buttons_var_vcd.append(
                                                list_variables_txt[
```

```
                                                        line_number - entity_number - 1].rsplit('_')[3])

                        if len(list_switches_vcd) != 0:
                                for i in range(len(list_switches_vcd)):
                                        if list_variables_txt[line_number - entity_number - 1].rsplit('_')[4] ==
                                                list_switches_vcd[i]:
                                                list_switches_var_vcd.append(
                                                        list_variables_txt[
                                                                line_number - entity_number - 1].rsplit('_')[3])

                        if len(list_labels_vcd) != 0:
                                for j in range(len(list_labels_vcd)):
                                        if list_variables_txt[line_number - entity_number - 1].rsplit('_')[4]==
                                                list_labels_vcd[j]:
                                                list_labels_var_vcd.append(
                                                        list_variables_txt[
                                                                line_number - entity_number - 1].rsplit('_')[3])

                        if len(list_greenleds_vcd) != 0:
                                for j in range(len(list_greenleds_vcd)):
                                        if list_variables_txt[line_number - entity_number - 1].rsplit('_')[4] ==
                                                list_greenleds_vcd[j]:
                                                list_greenleds_var_vcd.append(
                                                        list_variables_txt[
                                                                line_number - entity_number - 1].rsplit('_')[3])

                        if len(list_redleds_vcd) != 0:
                                for j in range(len(list_redleds_vcd)):
                                        if list_variables_txt[line_number - entity_number - 1].rsplit('_')[4] ==
                                                 list_redleds_vcd[j]:
                                                list_redleds_var_vcd.append(
                                                        list_variables_txt[
                                                                line_number - entity_number - 1].rsplit('_')[3])

                        if '$enddefinitions_$end' in line:
                                read_value = line_number
                                last_line_entry_vcd.set(read_value)

        print(last_line_entry_vcd.get())

        clear_screen()

# ------------------------------------------------------------
# ------------------------------------------------------------

def CMD_Python_Button_Execution_B(enter_value):
        ## Funcion de comunicacion presente durante la emulacion,
        ## siendo llamada cuando detecta un cambio en algun boton

        file_name = (file_name_vhd.get()).rsplit('.')[0]
        testbench_name = (testbench_name_vhd.get()).rsplit('.')[0]

        testbench_vcd_mod = testbench_name + '_mod.vcd'

        file_vhd_cd = (file_cd.get()) + '\\' + (file_name_vhd.get());
        file_vcd_cd = (file_cd.get()) + '\\' + file_name + '.vcd';

        testbench_vhd_cd = (testbench_cd.get()) + '\\' + (testbench_name_vhd.get());
        testbench_vcd_cd = (testbench_cd.get()) + '\\' + testbench_name + '.vcd';
        testbench_vcd_mod_cd = (testbench_cd.get()) + '\\' + testbench_vcd_mod;

        line_number = 0; entity_number = 0; read_value = 0; check = 0;
        list_variables_txt = []; list_variables = [];

        with open(testbench_vhd_cd,'rt') as vhd:
                vhd_txt = vhd.readlines()

        with open(testbench_vhd_cd,'wt') as vhd:
                vhd.writelines( vhd_txt[ 0 : last_line_entry_vhd.get() ] )
                vhd.writelines( [ "____\n", "____wait_for_300_ns;\n" ,
                "____" + list_buttons_vcd[enter_value] + "_<=_'1';\n" ,
                "____wait_for_300_ns;\n" ,
                "____" + list_buttons_vcd[enter_value] + "_<=_'0';\n", "____\n" ] )
                vhd.writelines(
                vhd_txt[ (last_line_entry_vhd.get() + 1) : len(vhd_txt) ] )

        last_line_entry_vhd.set(last_line_entry_vhd.get() + 5)

        # Ejecucion del nuevo codigo, con las entradas
        # debidamente indicadas

        ghdl_file_scan = 'ghdl_-s_' + (file_name_vhd.get());
        ghdl_file_analize = 'ghdl_-a_' + (file_name_vhd.get());
```

```
ghdl_file_execute = 'ghdl -e ' + file_name;

ghdl_testbench_scan = 'ghdl -s ' + (testbench_name_vhd.get());
ghdl_testbench_analize = 'ghdl -a '+(testbench_name_vhd.get());
ghdl_testbench_execute = 'ghdl -e ' + testbench_name;
ghdl_testbench_run = 'ghdl -r ' + testbench_name + ' --vcd='
+ testbench_vcd_mod;

initial_file_commands = [
        ghdl_file_scan,
        ghdl_file_analize,
        ghdl_file_execute]

initial_testbench_commands = [
        ghdl_testbench_scan,
        ghdl_testbench_analize,
        ghdl_testbench_execute,
        ghdl_testbench_run]

for command in initial_file_commands:
        subprocess.run(command, shell = True, cwd = (file_cd.get()))

for command in initial_testbench_commands:
        subprocess.run(command, shell = True,cwd=(testbench_cd.get()))

line_number = 0; list_variables_index = 0;
list_variables_txt = []; list_variables = [];

# Busqueda de las variables
# a interpretar dentro del archivo .vcd

print(last_line_entry_vcd.get())

with open(testbench_vcd_mod_cd,'rt') as vcd:

for line in vcd:

line_number += 1

        if (line_number > last_line_entry_vcd.get()):
                if len(list_labels_var_vcd) != 0:
                        for i in range(len(list_labels_var_vcd)):
                                if list_labels_var_vcd[i] in line:
                                        if (((list_labels_var_vcd[i] == '#') and (
                ****bool( list_variables_txt[ list_variables_index ]
                                        .rsplit('#')[0] != '') == True))
                                        or (list_labels_var_vcd[i] != '#')):
                                                list_variables_txt.append(line.rstrip())
                                                list_variables.append(
                                        *(list_variables_txt[ list_variables_index ].rsplit(
                                        *str( '_' + list_labels_var_vcd[i] ))[0]).rsplit('b')[1] )
                                                globals()[label_text_var[i]].set(
                                                        "\n".join(binA7seg_representations[
                                                **list_variables[ list_variables_index ] ]) )
                                                list_variables_index += 1

                if len(list_greenleds_var_vcd) != 0:
                        for i in range(len(list_greenleds_var_vcd)):
                                if list_greenleds_var_vcd[i] in line:
                                        if (((list_greenleds_var_vcd[i] == '#') and (
                ****bool( list_variables_txt[ list_variables_index ].rsplit('#')[0]
                                        != '') == True)) or (list_greenleds_var_vcd[i] != '#')):
                                                list_variables_txt.append(line.rstrip())
                                                list_variables.append(
                                        *(list_variables_txt[ list_variables_index ].rsplit(
                                        *list_greenleds_var_vcd[i] )[0]) )
                                        *greenLEDConfiguration( globals()[greenled_name_var[i]],
                                                *bool(list_variables[ list_variables_index ]) )
                                        *list_variables_index += 1

                if len(list_redleds_var_vcd) != 0:
                        for i in range(len(list_redleds_var_vcd)):
                                if list_redleds_var_vcd[i] in line:
                                        if (((list_redleds_var_vcd[i] == '#') and (
                ****bool( list_variables_txt[ list_variables_index].rsplit('#')[0]
                                        != '') == True)) or (list_redleds_var_vcd[i] != '#')):
                                                list_variables_txt.append(line.rstrip())
                                                list_variables.append(
                                        *(list_variables_txt[ list_variables_index ].rsplit(
                                        *list_greenleds_var_vcd[i] ))[0])
                                        *redLEDConfiguration( globals()[redled_name_var[i]],
                                                *bool(list_variables[ list_variables_index ]) )
                                        *list_variables_index += 1
```

```
        last_line_entry_vcd.set(line_number)

        clear_screen()

# ------------------------------------------------------------
# ------------------------------------------------------------

def CMD_Python_GTKWave_Init ():

        testbench_name =
        ( testbench_name_vhd.get() ).rsplit('.')[0]
        # Nombre del archivo TestBench sin extension

        testbench_vcd_mod = testbench_name + '_mod.vcd'
        # Cambio de ventana entre eleccion de numero de
        # perifericos y asociacion de nombres

        gtkwave_run = 'gtkwave_' + testbench_vcd_mod;
        # Cambio de ventana entre eleccion de numero de
        # perifericos y asociacion de nombres

        subprocess.run( gtkwave_run, shell = True,
                cwd = (testbench_cd.get()) )

        clear_screen()

# ------------------------------------------

root.bind('<Configure>', resize)

root.mainloop()
```

Bibliografía

[1] Universidad Nebrija. *Introducción a VHDL*. 2020. URL: https://www.cartagena99.com/recursos/alumnos/apuntes/IntroduccionVHDL.pdf (visitado 21-07-2021).

[2] HardwareBee. *The History of VHDL*. 2018. URL: https://hardwarebee.com/the-history-of-vhdl (visitado 21-07-2021).

[3] A.G.O. - Universidad de Alicante. *VHDL. Lenguaje de descripción hardware - Introducción e historia*. 2007. URL: https://rua.ua.es/dspace/bitstream/10045/3833/1/S2_1_VHDL_INTRODUCCION_HISTORIA.pdf (visitado 21-07-2021).

[4] Sergio Noriega - Universidad Nacional de La Plata. *Introducción al diseño lógico con VHDL*. 2012. URL: https://catedra.ing.unlp.edu.ar/electrotecnia/islyd/Tema%2012b%20Logica%20Programable%20VHDL%202012.pdf (visitado 21-07-2021).

[5] Genera Tecnologías. *Ingeniería FPGA: Aplicaciones de las FPGA*. 2020. URL: https://www.generatecnologias.es/aplicaciones_fpga.html (visitado 21-07-2021).

[6] Clifford Wolf. *Project IceStorm*. 2017. URL: http://www.clifford.at/icestorm/ (visitado 21-07-2021).

[7] Juan González Gómez Eladio Delgado Mingorance. *Alhambra board*. 2015. URL: https://alhambrabits.com/alhambra/ (visitado 21-07-2021).

[8] Carlos Venegas Arrabé Juan Gonzalez-Gomez Jesús Arroyo Torrens. *FPGA-Wars*. 2015. URL: https://github.com/fpgawars (visitado 21-07-2021).

[9] Tristan Gingold. *GHDL User Guide*. 2010. URL: http://ghdl.free.fr/site/pmwiki.php?n=Main.UserGuide (visitado 21-07-2021).

[10] Isabel Gracia Luengo Andrés Marzal Varó. *Introducción a la programación con Python*. Castellón de la Plana: Universidad Jaume I, 2009. ISBN: 978-84-692-5869-9. URL: http://www1.herrera.unt.edu.ar/biblcet/wp-content/uploads/2014/12/ippython.pdf.

[11] Inc. Aldec. *Active-HDL™ — FPGA Design and Simulation*. 2021. URL: `https://www.aldec.com/files/products/Active-HDL_Datasheet.pdf` (visitado 21-07-2021).

[12] Inc. Aldec. *FPGA Design Creation and FPGA Simulation*. 2021. URL: `https://www.aldec.com/en/products/fpga_simulation/active-hdl` (visitado 21-07-2021).

[13] Soo-Ik Chae Kyung-Soo Oh Sang-Yong Yoon. *Emulator environment based on an FPGA prototyping board*. Korea: IEEE, 2000. ISBN: 0-7695-0668-2. URL: `https://ieeexplore.ieee.org/abstract/document/855197/authors#authors`.

[14] José Daniel Muñoz Frías. *Introducción a los sistemas digitales*. Madrid: ICAI – Universidad Pontificia Comillas, 2018.