GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ADAPTACIÓN DE ICESTUDIO PARA TRABAJAR CON VHDL

Autor: Fernando Guijarro Bueno

Directores: Fermín Zabalegui Sanz y José Daniel Muñoz Frias

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Adaptación de Icestudio para trabajar con VHDL 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.

Fdo.: Fernando Guijarro Bueno Fecha: 21/ 07/ 2021

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

ZABALEGUI SANZ FERMIN - 50202045Z

Firmado digitalmente por ZABALEGUI SANZ FERMIN - 50202045Z
Fecha: 2021.07.21 13:24:45 +02'00'

Fdo.: Fermín Zabalegui Sanz Fecha: 21/ 07/ 2021

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

Fdo.: José Daniel Muñoz Frias Fecha: 21/ 07/ 2021

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ADAPTACIÓN DE ICESTUDIO PARA TRABAJAR CON VHDL

Autor: Fernando Guijarro Bueno

Directores: Fermín Zabalegui Sanz y José Daniel Muñoz Frias

Madrid

ADAPTACIÓN DE ICESTUDIO PARA TRABAJAR CON VHDL

Autor: Guijarro Bueno, Fernando.
Directores: Zabalegui Sanz, Fermín y Muñoz Frias, José Daniel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El campo de las FPGAs está en constante crecimiento, haciendo que las utilidades y posibilidades de uso de las FPGAs vayan en aumento. La herramienta de Icestudio permite el uso de estas de manera sencilla y visual. Icestudio permite describir en Verilog, lo que se busca en este proyecto es el aceptar la descripción de hardware en VHDL también, permitiendo así un mayor número de funciones y usuarios para este programa. Permitiendo a diseñadores la descripción hardware en el lenguaje que más les interese a ellos.

Palabras clave: Icestudio, VHDL, Verilog.

1. Introducción

El programa ICESTUDIO, usa las características de descripción por bloques, es decir, tiene pequeños componentes ya descritos, como puertas lógicas, bits, entradas, salidas, etc. Estos se pueden juntar pudiendo hacer un circuito complicado sin tener que describir gran cosa. Además, siempre se pueden incorporar códigos parciales para conseguir nuevos componentes o acciones no incluidas en el programa.

Estos códigos en el programa actual se han de diseñar en el lenguaje de Verilog. Y lo que se busca en este trabajo es darle una alternativa a este programa, haciendo que pueda llegar a mayor número de personas.

Al hacer esta aplicación accesible con VHDL se obtendrá un mayor ecosistema de aplicaciones de fácil uso de carácter académico. Intentando conseguir e igualar los logros y usos ya obtenidos con Arduino, pero con la función de realizar circuitos lógicos, lo cual a día de hoy está poco desarrollado. Esta aplicación dará a alumnos de gran diversidad de edades la posibilidad del manejo de FPGAs de una forma sencilla, pudiendo ser usado así por colegios y universidades para la docencia. Además, esta actualización de programa hará que un mayor número de profesionales puedan usarlo, facilitando y reduciendo los tiempos de cada trabajo, pudiendo eliminar la necesidad de uso de un diseñador muy experimentado.

2. Definición del proyecto

Este trabajo final de grado ha servido para dar los primeros pasos para la modificación completa de este programa, buscando hacerlo compatible tanto con la decripción en VHDL como la descripción en Verilog. Para ello se basará en el programa de Icestudio base, fácilmente descargable para cualquier plataforma, para posteriormente modificarlo permitiendo el uso de VHDL dentro de este, manteniendo bastante similar la estética de este.

El objetivo principal es generar un programa accesible e intuitivo para poder ser usado tanto a nivel educativo como a nivel empresarial, buscando dar las mayores facilidades a la hora de la descripción de código de las FPGAs para cualquier público. Los objetivos secundarios son los siguientes:

- Estudiar la generación de código de Icestudio y adaptar los componentes a lenguaje VHDL.
- Analizar la generación de circuitos sintetizados mediante Icestudio (y su dependencia Icestorm) y su adaptación a VHDL.
- Valorar y seleccionar las herramientas de código libre de compilación de VHDL para el proyecto.
- Seleccionar un traductor de VHDL a Verilog de código libre.
- Desarrollar la herramienta de diseño VHDL completa para Icestudio.

3. Descripción de la herramienta

La herramienta desarrollada ha sido Icestudio. Se ha partido del programa base que se ha descargado de la página oficial de este y se ha procedido a modificarlo. Además, se han utilizado programas de traducción de VHDL a Verilog como vhd2vl y compiladores como Mingw o Ghdl. A partir de ahí se han creado nuevos apartados en la herramienta para el fácil uso de los nuevos componentes. Haciendo así posible que los diseñadores de hardware que usen el programa puedan elegir el lenguaje de sus componentes de manera sencilla.

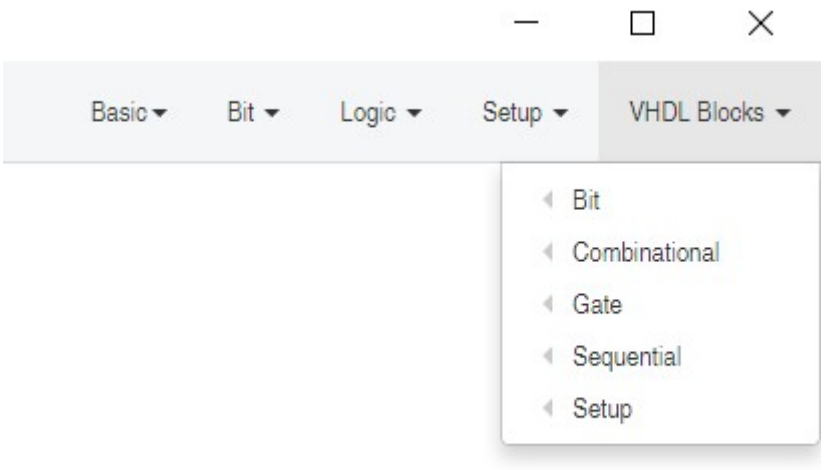

Ilustración 1 - Cambios visuales en la parte superior derecha de la pantalla del programa

Como se puede comprobar en la imagen, cuando se procede a seleccionar el bloque que se va a usar se puede ver fácilmente los añadidos para las mejoras que se pretenden incorporar. Todo esto sin modificar los bloques que el programa ya incluía originalmente, permitiendo tanto combinarlos como elegir el lenguaje de descripción preferido.

4. Resultados

Los resultados finales de este trabajo final de grado es el poner las bases para la modificación completa de la herramienta Icestudio para aceptar ambos lenguajes de descripción de hardware. Para ello se han diseñado ya todos los bloques, además de la elección de traductor entre ambos lenguajes de código libre. Permitiendo dar los primeros pasos en la obtención de un Icestudio que permita la creación de circuitos para FPGAs usando ambos lenguajes.

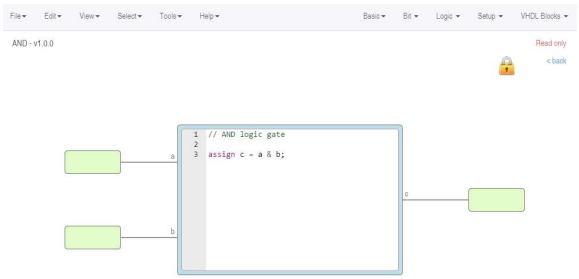

Ilustración 2 - Código AND Verilog

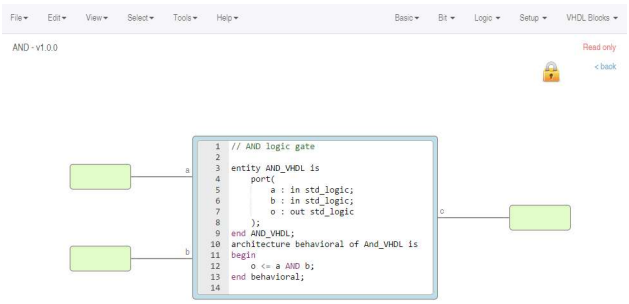

Ilustración 3 - Codigo AND VHDL

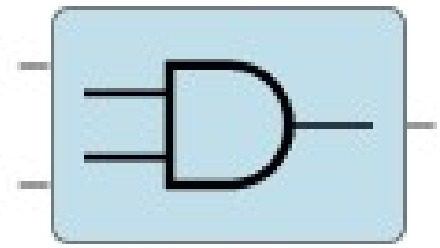

Ilustración 4 - Simbolo de la puerta AND, tanto para el código en VHDL como en Verilog

5. Conclusiones

Estas modificaciones del programa son fácilmente identificables además de ser sencillas de uso, manteniendo el carácter intuitivo del programa, permitiendo a estudiantes de todas las edades usarlo, pudiendo proceder a ser un programa muy útil en el sector educativo, haciendo así posible el aumento de uso y conocimiento de las FPGAs.

Este trabajo son los primeros pasos para hacer completamente accesible el mudo de las FPGAs, pudiendo compararse con la facilidad de desarrollo de Arduino y proponiendo las FPGAs como un sistema de fácil acceso y uso con gran cantidad de posibilidades.

6. Referencias

[1] Codrops, C., 2021, Icestudio, A real gamechanger in the world of Open Source FPGAs. Disponible en: https://icestudio.io [Último Acceso 7 de Julio de 2021]

[2] GitHub. 2021. GitHub - FPGAwars/icestudio: Visual editor for open FPGA boards. [online] Disponible en: https://github.com/FPGAwars/icestudio#installation [Último Acceso 7 de Julio de 2021]

[3] Sánchez-Élez, M., 2021. [online] Eprints.ucm.es. Disponible en: https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf [Último Acceso 7 de Julio de 2021]

ADAPTING ICESTUDIO TO BE COMPATIBLE WITH VHDL DESCRIPTION

Author: Guijarro Bueno, Fernando.
Supervisors: Zabalegui Sanz, Fermín y Muñoz Frias, José Daniel.
Collaborating Entity: Universidad Pontificia Comillas

ABSTRACT

The field of FPGAs is constantly growing, making FPGAs' utilities and possibilities of use increase. The Icestudio tool allows the use of these in a simple and visual way. Allowing the Verilog and VHDL language to be used within the program allows this application's use to be further opened. Allowing programmers to interact with the FPGAs in the language that interests them the most.

Keywords: Icestudio, VHDL, Verilog.

1. Introduction

The ICESTUDIO program uses the block programming characteristics, meaning by this that it has small components already programmed that can be connected with other blocks, a few examples of these blocks are logic gates, bits, inputs, outputs, etc ... Connecting these small blocks can make a complicated circuit without having to program much, understanding as programming the typical way of writing code. In addition, partial codes can always be incorporated to get new components or actions not included in the program.

These codes in the current program have to be programmed in the Verilog language. And what is sought in this final project is to give an alternative to this program, making it possible to reach a greater number of people.

By making this application accessible with VHDL, a greater ecosystem of easy-to-use academic applications will be obtained. Trying to achieve or match the achievements and uses already obtained with microcontrollers such as Arduino, but with the function of making logic circuits, which today is underdeveloped. This application will give students of a great diversity of ages the possibility of handling FPGAs in a simple way. In addition, this program update will allow a greater number of professionals to use it, facilitating and reducing the times of each programming job, thus eliminating the need for a very experienced programmer.

2. Project definition

This final degree project has served to take the first steps for the complete modification of this program, seeking to make it compatible with both VHDL and Verilog programming. In order to accomplish this project, the changes will be based on the Icestudio original program, easily downloadable for any platform, to later modify it allowing the use of VHDL within it, keeping its aesthetics quite similar.

The main objective is to generate an accessible and intuitive program to be used at an educational level as well as at a business level, seeking to give the greatest facilities when programming FPGAs for any audience. The secondary objectives are as follows:

- Study the generation of Icestudio code and adapt the components to the VHDL language.
- Analyze the generation of synthesized using Icestudio (within its dependency Icestorm Project) and its adaptation to VHDL.
- Evaluate and select the VHDL compilation open-source tools for the project.
- Select an open source VHDL to Verilog translator.
- Develop the complete VHDL design tool for Icestudio.

3. Tool description

The tool developed has been Icestudio. This project has started from the basic program that has been downloaded from its official website and have proceeded to modify it. In addition, VHDL to Verilog translation programs such as VHD2VL and compilers such as Mingw or Ghdl have been used. From there, new sections have been created in the tool for easy use of the new components. Making it possible for programmers who use the program to choose the language of its components in a simple way.

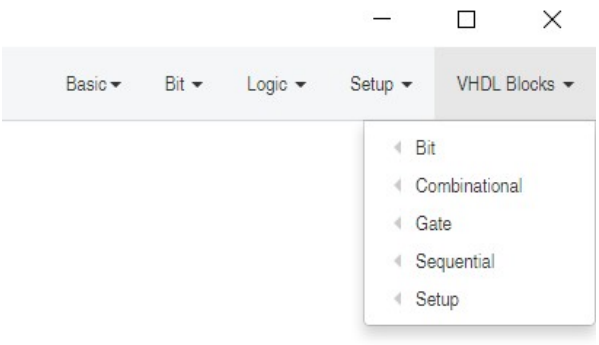

Ilustración 5 - Visual changes in the upper right of the program screen

As it can be seen in the image of above, when the user proceeds to select the block to be used, can easily see the additions for the improvements to be incorporated. All this without modifying the blocks that the program included originally, allowing both to be combine and to choose the preferred programming language.

4. Results

The final results of this final degree project are to lay the foundations for the complete modification of the Icestudio tool to accept both programming languages. For this, all the blocks have already been programmed, in addition to the choice of translator between both open-source languages. Allowing to take the first steps in obtaining an Icestudio that allows the creation of programs for FPGAs in both languages.

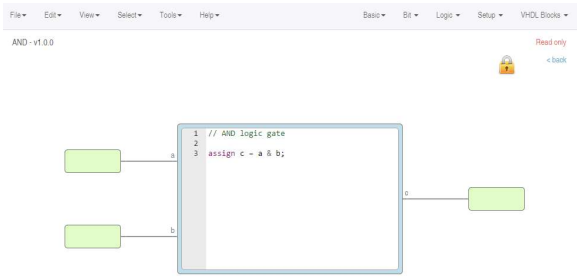

Ilustración 6 – AND Verilog code

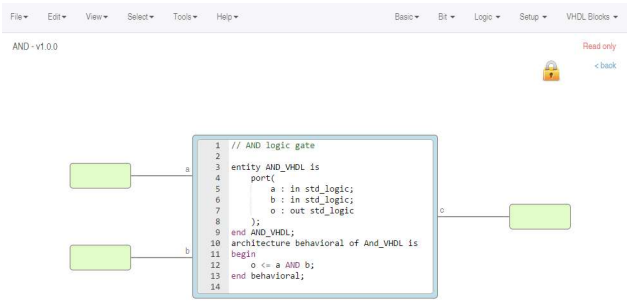

Ilustración 7 – AND VHD code

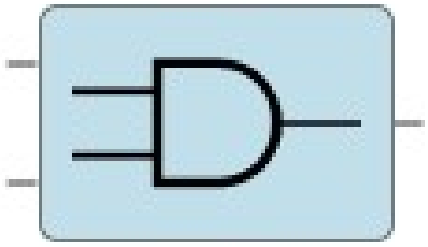

Ilustración 8 - AND gate symbol for both, VHDL and Verilog code

5. Conclusions

These modifications of the program are easily identifiable as well as being simple to use, maintaining the intuitive nature of the program, allowing students of all ages to use it, being able to proceed to be a very useful program in the educational sector, thus making it possible to increase its use. and knowledge of FPGAs.

This work is the first few steps to make the world of FPGAs completely accessible, being able to compare it with the ease of development of Arduino and proposing FPGAs as an easy-access and use processor with a large number of possibilities.

6. References

[1] Codrops, C., 2021, Icestudio, A real gamechanger in the world of Open Source FPGAs. Available at: https://icestudio.io [Accessed 7 July 2021]

[2] GitHub. 2021. GitHub - FPGAwars/icestudio: Visual editor for open FPGA boards. [online] Available at: https://github.com/FPGAwars/icestudio#installation [Accessed 7 July 2021]

[3] Sánchez-Élez, M., 2021. [online] Eprints.ucm.es. Available at: https://eprints.ucm.es/id/eprint/26200/1/intro_VHDL.pdf [Accessed 7 July 2021]

Índice de la memoria

Capítulo 1. Introducción 6
1.1 Motivación del proyecto 6
1.2 Icestudio, orígenes y descripción 7
Capítulo 2. Descripción de las Tecnologías 14
2.1 GITHUB 14
2.2 Programas de edición de texto 16
2.2.1 Microsoft Visual Studio Code 16
2.2.2 Sublime Text 3 17
2.3 Icestorm 17
2.4 Yosys 18
2.5 Python, JavaScript y HTML 19
2.6 Google Chrome 20
Capítulo 3. Estado de la Cuestión 21
Capítulo 4. Definición del Trabajo 32
4.1 Justificación 32
4.2 Objetivos 33
4.3 Metodología 33
4.4 Planificación y Estimación Económica 34
Capítulo 5. Sistema/Modelo Desarrollado 36
5.1 Análisis de Icestudio 36
5.2 Funciones Añadidas en VHDL 41
5.3 Diseño del Menú y Accesos 42
5.4 Traductor VHDL A Verilog 43
5.4.1 GHDL 44
5.4.2 Flex 44
5.4.3 GNU Bison 45
5.5 Análisis de Resultados 46
Capítulo 6. Conclusiones y Trabajos Futuros 48

Capítulo 7. Bibliografía ***50***

ANEXO I – Código de Compilado ***52***

ANEXO II – Código de la sección Tools ***74***

ANEXO III – Código de los Bloques en VHDL ***77***

ANEXO IV – Código del Menú ***85***

ANEXO V – Objetivos de Desarrollo Sostenible ***95***

Índice de figuras

Figura 1 - Proceso de funcionamiento de Icestudio 8
Figura 2 - Bloques dentro del grupo de Basic 9
Figura 3 - Bloques dentro del grupo de Logic 9
Figura 4 - Ejemplo 2D de la FPGA Alhambra 11
Figura 5 - Archivos que Icestudio permite exportar 13
Figura 6 - Parte del código sin formato del main.js, sacado de la descarga de Icestudio 15
Figura 7 - Parte del código con formato del main.js, sacado del Github de Icestudio 15
Figura 8 - Porcentajes de los lenguajes usados en el desarrollo de Icestudio 20
Figura 9 - Cambios realizados en la FPGA con bitstream 23
Figura 10 - Proceso de programación de una FPGA 23
Figura 11 - Programa ejemplo en Icestudio 25
Figura 12 - Sección de Tools Icestudio 41
Figura 13 - Bloque AND 42
Figura 14 - Acceso a los Bloques desde la barra superior 43
Figura 15 - Acceso a los Bloques desde File 43
Figura 16 - Esquema del funcionamiento de Icestudio modificado 45
Figura 17 - Programa de Icestudio Final 47
Figura 18 - Modificación de Icestudio para la compresión de VHDL y Verilog sin traductorees 49
Figura 19 - Bloque AND 77
Figura 20 - Bloque NAND 77
Figura 21 - Bloque OR 78
Figura 22 - Bloque NOR 78
Figura 23 - Bloque NOT 79
Figura 24 - Bloque XOR 79
Figura 25 - Bloque XNOR 80

Figura 26 - Bloque Multiplexor 80
Figura 27 - Bloque BIT 0 81
Figura 28 - Bloque BIT 1 81
Figura 29 - Bloque Debouncer 82
Figura 30 - Bloque Flip-Flop tipo D 83
Figura 31 Bloque Flip-Flop tipo T 83
Figura 32 - Interior de Flip-Flop tipo T 84
Figura 33 - Bloque de Preescalado con N genérico 84

Índice de tablas

Tabla 1 - Costes de Materiales 34
Tabla 2 - Costes por Tarea de Ingeniería..... 35

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

Este trabajo fin de grado tiene dos motivaciones principales, las cuales están bastante ligadas entre sí. Una de estas motivaciones principales es el influir de manera positiva en el desarrollo tecnológico, dando así más posibilidades de uso a las FPGAs, y por lo tanto darle más oportunidades de uso a distintas personas. Lógicamente con el aumento de uso de las FPGAs estas se desarrollarán más, ya sea por las marcas distribuidoras de estas, como Xilinx y Altera, o los consumidores, que al poseerlas y facilitar su uso les encontrarán nuevas funciones o desarrollos, como ya se ha visto con los microcontroladores del estilo de Arduino, que gracias a su sencillez de uso se han desarrollado bastante, consiguiendo hacer proyectos complejos en estos.

Otro objetivo o motivación del proyecto es la búsqueda de la ayuda a la enseñanza. Haciendo que programas como Icestudio ofrezcan distintos lenguajes de especificación, como VHDL y Verilog, permite tanto al cuerpo docente como a alumnos un rápido aprendizaje y uso de estos. Pudiendo potenciar lo aprendido en las clases y abriendo las posibilidades a un mayor número de gente. Los lenguajes de descripción de hardware, a día de hoy, más usados son VHDL y Verilog. Creando programas compatibles con ambos se abre mucho el abanico de uso de estos y a la vez aumenta el número de diseñadores que puedan usar FPGAs.

Además, se busca realizar un programa simple e intuitivo para permitir a diseñadores de todos los niveles usarlos. La incorporación de ejemplos en el programa junto con la facilidad de uso de este, permiten a diseñadores principiantes una toma de contacto con las FPGAs, haciendo posible que estos vayan mejorando sus conocimientos con un uso progresivo del programa.

1.2 ICESTUDIO, ORÍGENES Y DESCRIPCIÓN

Icestudio es un proyecto que comenzó en 2016 con el trabajo de Juan González Gómez, Jesús Arroyo Torrens y Carlos Venegas Arrabé, los cuales comenzaron con el proyecto de desarrollo tanto de FPGAs accesibles como un programa accesible y sencillo para el uso de estas. Buscando continuar los pasos de Icestorm, siendo este el primer programa para FPGA de código libre.

Como se comentará más a fondo posteriormente, Icestorm es un sistema que permite generar archivos de bitstream, es decir, archivos de configuración de FPGAs, a partir de verilog, usando herramientas libres como YOSYS y Arachne.pnr. Para el correcto funcionamiento de Icestorm, es necesario partir de un archivo de Verilog y de un archivo de configuración.

Icestudio es un programa disponible para Linux, Mac y Windows, siendo así accesible por mayor número de usuarios, que junto a la facilidad de uso que tiene, permite poder usar de forma sencilla las distintas FPGAs disponibles.

Dado a ser un proyecto de código libre, está sustentado por una gran comunidad de programadores grandes y pequeños que se ayudan mutuamente a la resolución de problemas surgidos en su uso. Además de estar ya traducida en tres idiomas casi al completo y más de diez por encima del setenta por ciento.

El funcionamiento de Icestudio es muy sencillo. Tras instalarse en el ordenador de igual manera que cualquier otro programa y abrirlo, aparecerá una página en blanco en la que ya se podrá proceder a programar. Y el proceso de sintetizado y compilado de esta página es el mostrado en la Figura 1.

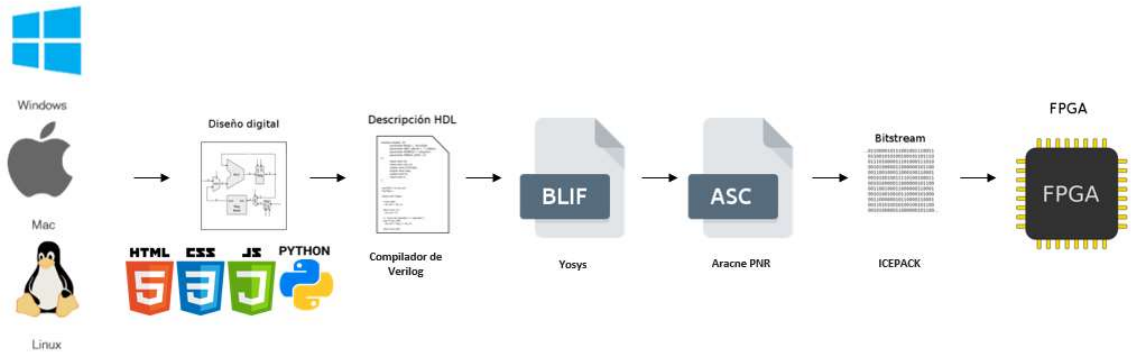

Figura 1 - Proceso de funcionamiento de Icestudio

Los pasos mostrados en la Figura 1 son los siguientes, primero se muestra en que tres sistemas operativos es descargable la aplicación de Icestudio, seguido de la parte visual, es decir el Frontend de la aplicación, siendo este diseñado en HTML, CSS, JS y Python, permitiendo así al usuario la realización del diseño digital, con este diseño se genera el código de Verilog, siendo este el que se va a usar para la creación del archivo de bitstream. Con un proceso basado en el Icestorm se consigue este archivo bitstream el cual se procede a cargar en la FPGA.

Icestudio tiene cuatro grandes grupos de bloques a poder usar, los cuales son los siguientes:

- Basic/Básicos: en los que se incluyen las entradas y salidas de la placa, de la FPGA, pulsadores, interruptores o LEDs. Además, se incluye la opción de poner cuadros de información o de código, por si no está predefinido el bloque que se desea usar. Como se puede ver en la Figura 2 a continuación.

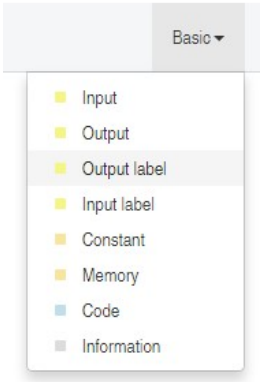

Figura 2 - Bloques dentro del grupo de Basic

- Bit: en esta opción se puede agregar un bloque con un 0 o 1 lógico, siendo estos los únicos bloques que se encuentran aquí, siendo muy útiles por ejemplo en el caso de querer usar un comparador.

- Logic/Lógica: incluyendo en estos, otros tres subgrupos los cuales engloban puertas lógicas, multiplexores, debouncers, siendo este un bloque elimina las perturbaciones generadas al comienzo de pulsar un botón, flip-flops y pre-escalados. Estos están dentro de los subgrupos de combinacional, puertas y secuencial. Estos subgrupos se pueden observar de forma más clara en la Figura 3.

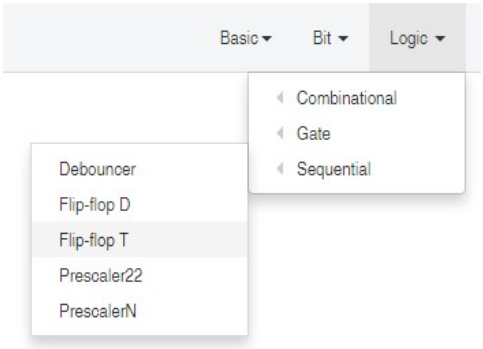

Figura 3 - Bloques dentro del grupo de Logic

- Setup: incluyen un bloque de pull-up y de triestado, bloques comunes en circuitos electrónicos, pero más complicados de diseño.

Además, para el correcto funcionamiento del programa se ha de seleccionar con que FPGA se va a conectar, ya que la configuración de los pines de entrada y salida ya están descritos en archivos HTML a los que accederá la aplicación. De ahí que sea importante seleccionar la FPGA a usar antes de escribir nada. Todas las placas cargadas tienen también un modelo cargado en el programa para poder ver correctamente los pines, como un gráfico 2D de la FPGA seleccionada por el usuario. En la Figura 4 se puede observar lo expuesto anteriormente.

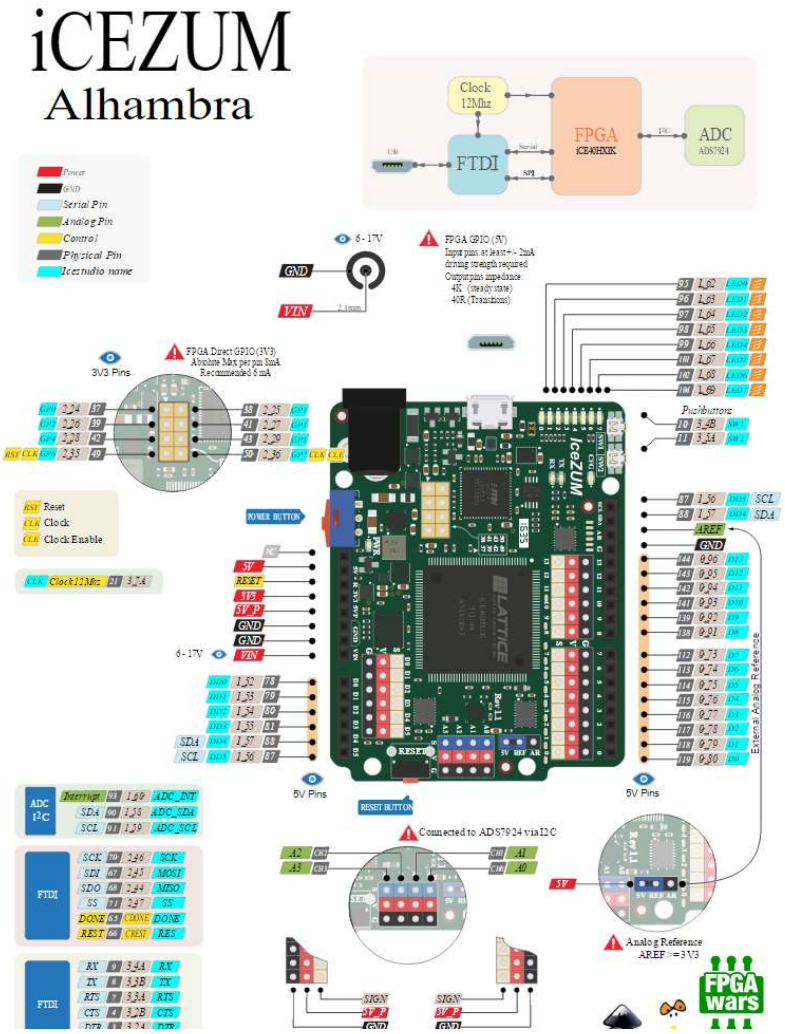

Figura 4 - Ejemplo 2D de la FPGA Alhambra

Como se puede observar en la Figura 4, se pueden ver todos los pines de entrada y salida, así como el nombre que tienen y por tanto como se llamarán dentro del programa Icestudio, esto es bastante útil de cara a cuando el usuario cambia de FPGA, o cuando el usuario no está muy familiarizado con esta. Con este gráfico 2D se puede elegir fácilmente entre todas las opciones que se ofrecen a la hora de elegir las entradas y salidas del programa.

Icestudio además permite incluir colecciones creadas por el usuario o por otros diseñadores, permitiendo que la cantidad de bloques a usar aumente y, por lo tanto, la descripción clásica de código por parte del usuario se reduzca. A pesar de poder añadir colecciones el usuario seguirá teniendo la posibilidad de escribir código si hay algún componente que le falte o

quiera modificar. Todos los componentes o bloques incluidos por el programa se pueden modificar fácilmente, permitiendo al usuario cambiarlos a su antojo.

Como se ha expuesto antes, Icestudio es un "hijo" de Icestorm, es decir, que se ha desarrollado a partir de este. Icestorm funciona a partir de dos archivos iniciales, siendo estos, un archivo .v, es decir, un archivo de Verilog, con código en este lenguaje, de que quiere el usuario que haga el circuito y de un archivo .pcf que indica a que pines se va a conectar cada señal, es decir, un configurador de la FPGA. A partir del archivo de Verilog, Icestorm crea un archivo de tipo BLIF, y con este más el archivo configurador, ya convertido a un archivo de texto se crea el archivo de tipo BIN, que es el que la placa entiende. Para ello se usan programas como Yosys o Arachne.pnr.

El funcionamiento de Icestudio es realmente similar, con las pequeñas diferencias de que Icestudio no crea los archivos de texto, sino solo los archivos de Verilog, PCF, BLIF y BIN. Todos estos archivos los genera en una carpeta temporal de donde se procederá a cargarlo a la placa. La razón de guardarlo en la carpeta temporal es debido a que para no guardar archivos de carga de manera indefinida y evitar la saturación del ordenador del usuario.

Por último, de cara al conocimiento a fondo de Icestuio, este permite la posibilidad de exportar algunos de los archivos comentados anteriormente, en caso de que se quieran compilar con otro programa, escribir sobre estos o usarlos en algún otro dispositivo que precise de esos archivos y no del mismo final de Icestudio. Además, esto es útil también en el caso de que el usuario posea otra FPGA que no esté cargada en la base de datos de Icestudio, permitiendo así con otro compilador y pequeñas modificaciones en la elección de los pines el poder usar el mismo código para esta y así evitarse tener que escribir la mayoría de código. Los distintos archivos que Icestudio genera para que el usuario pueda usarlos se pueden observar en la Figura 5.

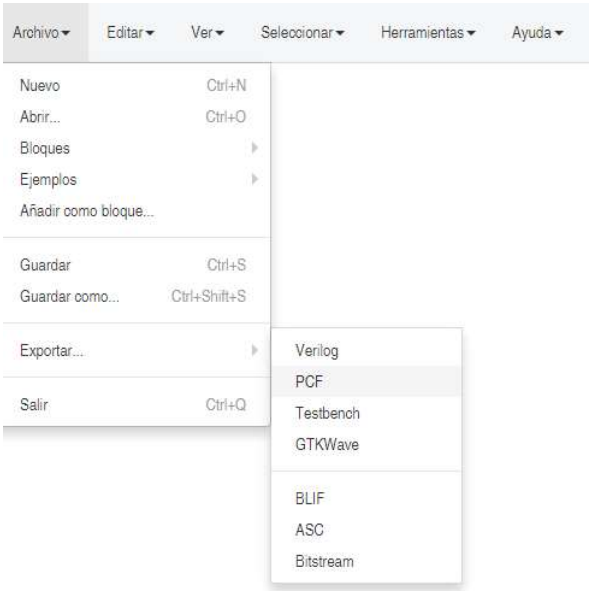

Figura 5 - Archivos que Icestudio permite exportar

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 GITHUB

Github es una página web donde muchos programas y proyectos de código libre se desarrollan y se descargan. En el caso de Icestudio a pesar de tener su propia página web para la descarga y algunas versiones, otras se aconsejan descargarlas desde Icestudio, ya que las versiones más modernas que aún están probando se añaden primero a esta página web.

A parte de para descargar archivos, Github es como una gran comunidad de usuarios que se ayudan entre sí. Debido a esto cada programa tiene una sección de "Issues", en el cual cualquier usuario registrado puede hacer una pregunta y le es respondida en poco tiempo, generalmente, por otros usuarios más experimentados en el tema o incluso por los desarrolladores.

Además, como usuario registrado puede ayudar a los desarrolles originales haciendo modificaciones por tu cuenta y agregarlas directamente a la página de Github del programa en cuestión, obviamente tiene un control de cambios en el que solo los administradores de ese proyecto pueden aceptar o rechazar los cambios, para evitar los posibles cambios, ya sean por error o por alguna otra causa.

Para el desarrollo de este trabajo también ha sido muy útil ya que todas las partes del programa Icestudio programado en JS, java script, estaban correctamente separadas y escritas, facilitando su lectura y comprensión. Cuando te descargas el programa hay un archivo de JS, llamado main.js en el que está desarrollado todo el código del programa para su correcto funcionamiento, pero este archivo está sin formato y generado solo de carácter funcional, en el caso del Github, este código está dividido en numerosos archivos más pequeños y de fácil lectura, donde también se ha cuidado el estilo facilitando la lectura de

este. En las Figura 6 y Figura 7 se pueden observar las diferencias entre ambos formatos, pudiendo verse claramente la facilidad de lectura de uno y la dificultad de lectura del otro

Figura 6 - Parte del código sin formato del main.js, sacado de la descarga de Icestudio

```
angular.module('icestudio')
  .service('compiler', function (common,
    utils,
    nodeSha1,
    _package) {

    this.generate = function (target, project, opt) {
      var content = '';
      var files = [];
      switch (target) {
        case 'verilog':
          content += header('//', opt);
          content += '`default_nettype none\n\n';
          content += verilogCompiler('main', project, opt);
          files.push({
            name: 'main.v',
            content: content
          });
          break;
        case 'pcf':
          content += header('#', opt);
          content += pcfCompiler(project, opt);
          files.push({
            name: 'main.pcf',
            content: content
          });
          break;
        case 'lpf':
          content += header('#', opt);
          content += lpfCompiler(project, opt);
          files.push({
            name: 'main.lpf',
            content: content
          });
          break;
```

Figura 7 - Parte del código con formato del main.js, sacado del Github de Icestudio

Como se puede observar las diferencias son notables, debido a que en el código del main.js se muestra el código compilado de la aplicación Icestudio, mientras que en Github se permite visualizar el código fuente. La ventaja que tiene Git, es que se permite el desarrollo en

paralelo, es decir cada programador puede escribir sus funciones y código por separado para posteriormente unirlo, marcándose las diferencias de manera fácil y visual. Además, Git permite el control de cambios, haciendo así que si hay algún fallo en versiones posteriores se puede retroceder de manera sencilla.

2.2 PROGRAMAS DE EDICIÓN DE TEXTO

En este trabajo final de grado se han usado mayormente dos programas de escritura y lectura de código, los cuales son los comentados a continuación.

2.2.1 MICROSOFT VISUAL STUDIO CODE

Microsoft Visual Studio Code es un editor de código fuente que permite la escritura y lectura de código. El usuario puede elegir entre un gran número de lenguajes, incluyendo así ayudas a la hora de escribir y leer, gracias a los distintos patrones de colores y a las predicciones de escritura que permiten mayor rapidez para ello.

Este programa es gratis para descargarse y tiene una amplia gama de lenguajes para programar en los cuales te proporciona ayudas, y aunque existen extensiones de VHDL y Verilog se decidió usar Sublime para estos archivos.

Otras ventajas que tiene este editor es que es personalizable, permitiendo a los distintos usuarios añadir sus propios atajos de teclado, cambiar los temas y colores del editor.

Los lenguajes principales que soporta y permite el resaltado de código son HTML, CSS, JS, C y C++, entre otros. De estos tanto HTML como JS serán muy útiles, ya que el desarrollo completo del programa Icestudio está programado mayormente en estos lenguajes.

Además de ser un programa gratis, en 2015 fue publicado en Github, con la característica de ser un programa de código libre y permitiendo a los usuarios de Github la creación de extensiones y colecciones, haciéndolo más personalizable a los usuarios de este.

2.2.2 SUBLIME TEXT 3

El programa Sublime Text 3 es otro editor de texto, que en este caso se usará mayormente para la descripción y desarrollo de los bloques y circuitos en VHDL. Tiene características similares al Microsoft Visual Studio Code con la diferencia que este programa no es de código libre ni gratuito, es necesario una licencia, pero es posible usar la versión de prueba, la cual permite usar todo el potencial el editor.

Las ventajas de este editor de código con respecto otros editores que también tienen ayudas con VHDL, es que este editor permite la multi selección y el multi cursor, que permite tanto escribir como borrar en varios sitios a la vez. Permitiendo así la modificación rápida de nombres de variables o de nombres de estructuras. Visual Studio Code también tiene estas funcionalidades.

Con este editor se ha procedido a desarrollar todos los bloques incluidos en el programa base de Icestudio en el lenguaje de VHDL. Para el correcto desarrollo de estos hay que tener en cuenta las diferencias entre ambos lenguajes, teniendo en cuenta que en VHDL hay que declarar las entradas y salidas, pudiendo hacer esto tanto en un programa de compilado previo o en los mismos bloques. En los bloques escritos en Verilog no declaran dichas entradas o salidas ya que cuando se diseñó el programa de Icestudio se decidió a omitir esto en la parte visual e incluirlo en el archivo de compilado.

2.3 ICESTORM

Icestorm es la primera herramienta de código libre para descripción de hardware para FPGAs. Previamente a este todas las herramientas para describir las FPGAs eran

proporcionadas por los desarrolladores de estas placas o no era posible acceder al código fuente de estas, haciendo que no hubiera herramientas de código libre.

De ahí surgió la necesidad de crear Icestorm, capaz de describir FPGAs, y siendo de código libre y gratuito. Icestorm permite a partir de un archivo de Verilog con el código a programar en la FPGA y un archivo PCF de la FPGA, crear un archivo BIN y cargarlo en la FPGA, permitiendo así finalmente programar la placa. Esto se consigue gracias a los programas Yosys y Arachne.pnr.

Para la creación del archivo BIN usa el paquete de Icestorm llamado Icepack y para finalmente cargarlo se usa el paquete Iceprog. Hay que tener en cuenta que Icestorm es un programa que no tiene interfaz visual para el usuario, por lo tanto, se accederá a él desde Command Prompt o Símbolo del Sistema, en español. A través de comandos en la pestaña de Command Prompt se podrá usar el programa Icestorm y permitiendo así cargar el diseño de hardware deseado en la FPGA.

2.4 YOSYS

Yosys es un programa para la síntesis de código en Verilog RTL, con otras palabras, un compilador de código de Verilog. A día de hoy da mucho soporte a Verilog 2005 y proporciona algoritmos básicos para la síntesis de este para su aplicación en distintos ámbitos. En el caso de este trabajo fin de grado será usado en la síntesis de código de Verilog para la creación de archivos BLIF tanto en Icestudio como en Icestorm.

A parte de generar archivos de tipo BLIF también es capaz de genera archivos EDIF o BTOR o SMT-LIB. Al ser de código libre Yosys se puede adaptar para compilar a otro tipo de archivos, combinando los algoritmos de compilado y cambiando ligeramente, añadiendo pasos de compilado al código en C++ que define a este. Debido a una extensión de GHDL, Yosys es capaz de sintetizar ahora diseños en VHDL, estando esta extensión aún en periodo de prueba.

2.5 PYTHON, JAVASCRIPT Y HTML

Estos lenguajes son en los que se basa el desarrollo del programa de Icestudio. Son tres de los lenguajes más usados actualmente en el desarrollo de apps, así como el desarrollo Backend o el desarrollo web.

Python, en particular, es un lenguaje de programación que se basa en la legibilidad del código escrito, facilitando así la lectura por los futuros programadores, haciéndolo fácil de leer y entender, a pesar de no tener un nivel muy elevado en este. En este caso, Python se está usando para desarrollar una futura versión en la que se permita estar en un entorno de Python y por lo tanto programar en este.

JavaScript es el lenguaje más usado en la redacción de todo el código y funcionamiento del programa. JavaScript es un lenguaje de programación de alto nivel que permite hacer funciones complejas dentro del desarrollo del código de HTML y CSS, los cuales están más dedicados al desarrollo de la aplicación de carácter visual y no tanto al funcionamiento de esta.

JavaScript permite la creación de programas dinámicos y mejora la interfaz del usuario, permitiendo así funciones como la unión de bloques o el movimiento de estos por la pantalla, además de la creación de nuevos bloques. Básicamente permite la interacción en tiempo real del usuario con los distintos componentes del programa.

Por último, queda la descripción del lenguaje HTML, que junto a la extensión CSS, permite generar un programa más visual. Tanto el lenguaje HTML como CSS se encargan de la parte del desarrollo Frontend más simple, ya sean las figuras o colores del display del programa, permitiendo mostrar una interfaz más amigable y simple para el usuario.

Los porcentajes del desarrollo de la aplicación de Icestudio son los mostrados en la Figura 8, siendo estos los lenguajes mayoritarios usados, pero no los únicos.

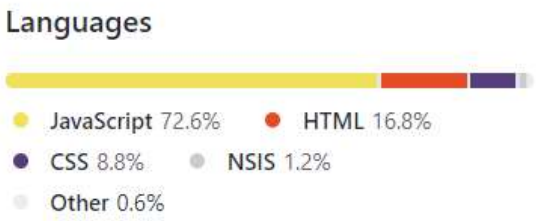

Figura 8 - Porcentajes de los lenguajes usados en el desarrollo de Icestudio

2.6 GOOGLE CHROME

Hay que hacer una especial mención a Google Chrome, que a pesar de ser conocido y la mayoría de los usuarios conocen y usan, se debe recalcar el uso intensivo de este para descargar los programas, así como toda la información necesaria para el correcto desarrollo de este trabajo final de grado, siendo este el principal motor de búsquedas y de obtención de información. Además, también se ha usado para comprobar en tiempo real los cambios ejecutados en los archivos de HTML del programa, ya que se podía modificarlos estos y actualizar la página para poder observar de manera rápida estos cambios y así conseguir ver los bloques y apartados añadidos en la aplicación antes de incluirlos a esta. Permitiendo la rápida modificación del menú.

Capítulo 3. ESTADO DE LA CUESTIÓN

Las FPGAs son una versión de los circuitos integrados de aplicación especifica o ASIC, siendo estas sus siglas en inglés, que permite a estos ser configurables. La FPGA siendo inventada en 1984 y teniendo aplicaciones similares a los ASIC, tienen distintas ventajas y desventajas respecto a su predecesor.

Las principales ventajas y desventajas son las siguientes:

- Son más lentas que un ASIC, siendo este un problema en el caso de la necesidad de un procesador rápido ya sea para el uso de procesadores en nuevas tecnologías en vehículos, poniendo un ejemplo, en la comunicación entre vehículos para funciones de seguridad.

- Al ser un dispositivo programable compuesto por bloques lógicos, debido a la sobrecarga del sistema de configuración, para un mismo tamaño la FPGA tiene menos puertas lógicas útiles que un ASIC.

- A pesar de esto, cando se buscan FPGAs o ASICs de menor tamaño, es más sencillo y barato de desarrollar, con menor tiempo de desarrollo también, y construir la FPGA con respecto al ASIC

- La mayor ventaja de las FPGAs con respecto a las placas ASICs, es la posibilidad de ser reprogramables fácilmente en cualquier sitio con una inversión muy pequeña, permitiendo gran flexibilidad en diseño.

Como se puede observar no es mejor el ASIC que una FPGA, ni viceversa, lo importante es el uso que va a dar el usuario. Buscando siempre sacar el máximo provecho al componente y, por lo tanto, el usuario debe tener claro el uso y necesidades, tanto actuales como futuras.

En el caso de este trabajo de fin de grado, se buscará el centrarse en la mayor ventaja de las FPGAs, siendo esta la capacidad de reprogramación. El chip es capaz de programarse debido a que está compuesto por gran cantidad de componentes lógicos, pudiendo variar su valor y por lo tanto activarse o desactivarse para el correcto funcionamiento del diseño. Estos permiten la actuación como puertas lógicas (AND, OR, NOT...) o como sistemas más complejos como debouncers o flip-flops. Las conexiones de las puertas que componen el chip son activadas debido a que la memoria RAM les asigna un valor. El archivo bitstream tiene guardado los valores que se quiere asignar a cada celda de RAM.

Para comprender correctamente todas las tecnologías y proyectos que hay activos en el presente o ya han sido acabados, hay que entender cómo funciona una FPGA y como ha cambiado a lo largo de los años.

Mucha de la información obtenida ha sido leída y sintetizada de artículos escritos por Juan González Gómez, uno de los creadores de Icestudio y con numerosos artículos en la materia.[8][9]

Las FPGAs son chips que contienen numerosas puertas o componentes lógicos impresos en este y conectados entre sí como si de una malla de componentes se tratara. Programas como Icestudio lo que consiguen es generar un código de bitstream que indica que cables se activan y cuales se genera un abierto, como se puede observar en la Figura 9, haciendo así que la FPGA cumpla con los parámetros desarrollados por el diseñador.

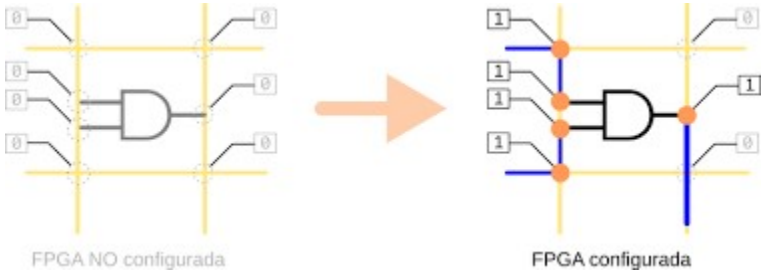

Figura 9 - Cambios realizados en la FPGA con bitstream

Como se puede observar en la Figura 10 el proceso de programación es el diseño digital, la descripción en lenguajes tales como Verilog o VHDL, la síntesis, la generación del configurador, bitstream comúnmente, y por último la carga del archivo en la FPGA.

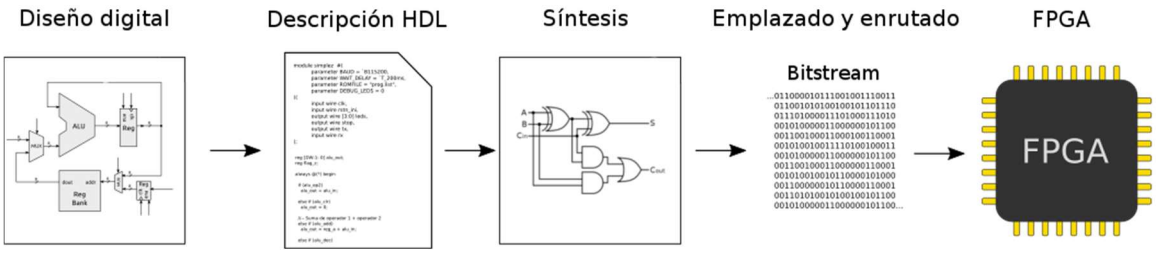

Figura 10 - Proceso de programación de una FPGA

Tras esta breve descripción cabe destacar que las FPGAs a pesar de ser reprogramables hasta 2015, cuando se desarrolló la FPGA iCE40 de Lattice, las marcas desarrolladoras de estas no permitían una descripción abierta, lo que lo hacia una tecnología muy cerrada, ya que estas solo permitían a los diseñadores usar la FPGA en las condiciones indicadas cuando se compró, sin permitir el desarrollo de programas propios por estos o con grandes variantes a los proporcionados por los fabricantes. Esto hizo que a pesar de estar desarrollada la tecnología por más de treinta años no se hubiera desarrollado mucho por diseñadores particulares, sino únicamente por los fabricantes.

En 2015, debido al desarrollo del primer software de código libre aparecieron las primeras FPGAs denominadas libres. Este programa era Icestorm, que como se ha explicado anteriormente permitía por un proceso ya no tan complejo la programación de las FPGAs.

Icestorm, como lo explicado en esta memoria anteriormente, funciona junto con Yosys y Arachne.pnr en la creación de un programa compatible con las FPGAs. A partir de un archivo de Verilog, Yosys lo sintetiza y genera un archivo de netlist en formato BLIF, para que el programa Arachne.pnr genere una representación del bitstream en formato de texto. Y por último las funciones internas de Icestorm convierten con Icepack el archivo de texto a un archivo Binario, el cual define el bitstream en un lenguaje comprensible por la FPGA, y la función Iceprog carga este archivo en la FPGA.

Icestorm es el primer proyecto de código libre de descripción de hardware para FPGAs, en los que se basan muchos programas posteriores de similar índole.

Icestudio, es uno de los programas más usados y conocidos actualmente para la programación de FPGAs, el cual basándose en Icestorm permite la programación de FPGAs con una interfaz más amigable para el usuario y permitiendo también la creación de los archivos de Verilog con descripción gráfica.

La mayor diferencia entre Icestudio y Icestorm es la interfaz. Para poder usar Icestorm el programador ha tenido que crear primero un diseño con la extensión de Verilog y en este lenguaje.

En Icestudio el diseñador no tiene por qué escribir código, puede realizar un diseño en Verilog con la tecnología de bloques, pero permitiendo añadir código de manera manual como ya se hace en Icestorm.

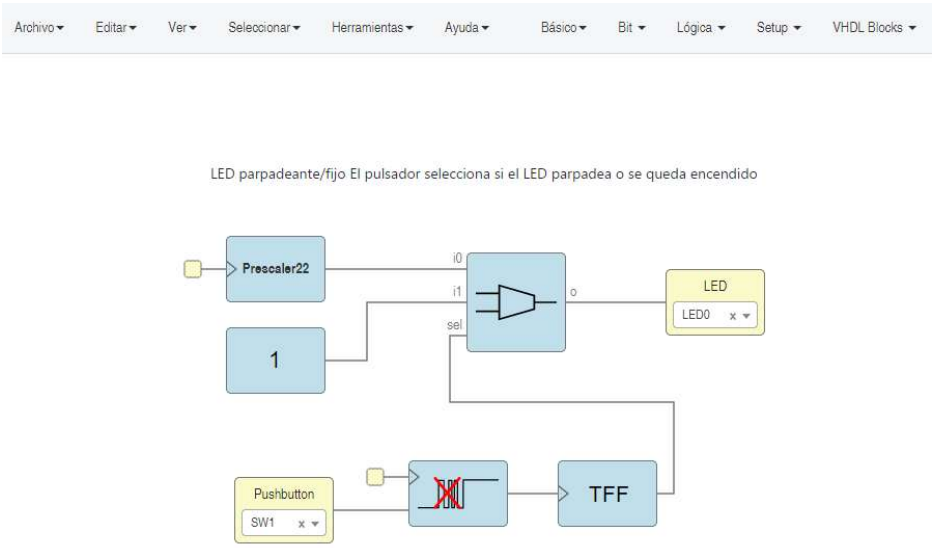

Figura 11 - Programa ejemplo en Icestudio

Como se puede observar en la Figura 11, se ha podido definir una descripción de hardware completa en Verilog con bloques, haciendo con estos que un Led pueda tener la función de encendido fijo o parpadeante según un reloj. Al ser un ejemplo sencillo no ha sido necesario la creación de código extra, pero en casos más complejos se podría añadir bloques de especificación más concretos y no incluidos en las colecciones del programa.

```
// Code generated by Icestudio 0.5.0

`default_nettype none

module main (
 input vf1bd8c,
 input vclk,
 output v090b60,
 output [0:6] vinit
);
 wire w0;
 wire w1;
 wire w2;
 wire w3;
 wire w4;
 wire w5;
 wire w6;
 wire w7;
 assign v090b60 = w1;
 assign w2 = vf1bd8c;
 assign w6 = vclk;
```

```
 assign w7 = vclk;
 assign w7 = w6;
 vadcbe7 vce87cc (
  .v2efea4(w3),
  .v0daa9e(w6)
 );
 vddd2fa v008358 (
  .v9a9c84(w0),
  .v741632(w1),
  .v765971(w3),
  .v675419(w5)
 );
 v3e6c24 ve5eac8 (
  .v608bd9(w0)
 );
 v10d933 v521b17 (
  .v6a82dd(w2),
  .vd4e5d7(w4),
  .v444878(w7)
 );
 v6c8610 vd07a9d (
  .vef4cea(w4),
  .vc24d9f(w5)
 );
 assign vinit = 7'b0000000;
endmodule

module vadcbe7 #(
 parameter v245bef = 22
) (
 input v0daa9e,
 output v2efea4
);
 localparam p1 = v245bef;
 wire w0;
 wire w2;
 assign w0 = v0daa9e;
 assign v2efea4 = w2;
 v435b29 #(
  .v100e1b(p1)
 ) v81886f (
  .v0daa9e(w0),
  .v2efea4(w2)
 );
endmodule

module v435b29 #(
 parameter v100e1b = 22
) (
 input v0daa9e,
 output v2efea4
);
 localparam p2 = v100e1b;
```

```
 wire w0;
 wire w1;
 assign v2efea4 = w0;
 assign w1 = v0daa9e;
 v435b29_vac7386 #(
  .N(p2)
 ) vac7386 (
  .clk_out(w0),
  .clk_in(w1)
 );
endmodule

module v435b29_vac7386 #(
 parameter N = 0
) (
 input clk_in,
 output clk_out
);
 //-- Number of bits of the prescaler
 //parameter N = 22;

 //-- divisor register
 reg [N-1:0] divcounter;

 //-- N bit counter
 always @(posedge clk_in)
   divcounter <= divcounter + 1;

 //-- Use the most significant bit as output
 assign clk_out = divcounter[N-1];
endmodule

module vddd2fa (
 input v765971,
 input v9a9c84,
 input v675419,
 output v741632
);
 wire w0;
 wire w1;
 wire w2;
 wire w3;
 assign v741632 = w0;
 assign w1 = v765971;
 assign w2 = v9a9c84;
 assign w3 = v675419;
 vddd2fa_v7f68b8 v7f68b8 (
  .o(w0),
  .in0(w1),
  .in1(w2),
  .sel0(w3)
 );
endmodule
```

```
module vddd2fa_v7f68b8 (
 input in0,
 input in1,
 input sel0,
 output o
);
 reg _o;

 always @(*) begin
     case(sel0)
         0: _o = in0;
         1: _o = in1;
         default: _o = in0;
     endcase
 end

 assign o = _o;
endmodule

module v3e6c24 (
 output v608bd9
);
 wire w0;
 assign v608bd9 = w0;
 v3e6c24_v68c173 v68c173 (
  .v(w0)
 );
endmodule

module v3e6c24_v68c173 (
 output v
);
 // Bit 1

 assign v = 1'b1;
endmodule

module v10d933 (
 input v444878,
 input v6a82dd,
 output vd4e5d7
);
 wire w0;
 wire w1;
 wire w2;
 assign vd4e5d7 = w0;
 assign w1 = v444878;
 assign w2 = v6a82dd;
 v10d933_va7041c va7041c (
  .out(w0),
  .clk(w1),
  .in(w2)
```

```
 );
endmodule

module v10d933_va7041c (
 input clk,
 input in,
 output out
);
 //-- Debouncer Circuit
 //-- It produces a stable output when the
 //-- input signal is bouncing

 reg btn_prev = 0;
 reg btn_out_r = 0;

 reg [16:0] counter = 0;


 always @(posedge clk) begin

   //-- If btn_prev and btn_in are differents
   if (btn_prev ^ in == 1'b1) begin

       //-- Reset the counter
       counter <= 0;

       //-- Capture the button status
       btn_prev <= in;
   end

   //-- If no timeout, increase the counter
   else if (counter[16] == 1'b0)
       counter <= counter + 1;

   else
     //-- Set the output to the stable value
     btn_out_r <= btn_prev;

 end

 assign out = btn_out_r;

endmodule

module v6c8610 (
 input vef4cea,
 output vc24d9f
);
 wire w0;
 wire w1;
 wire w2;
 wire w3;
 assign vc24d9f = w0;
```

```
 assign w2 = vef4cea;
 assign w1 = w0;
 v70ff7f v082a97 (
  .vc24d9f(w0),
  .vb55943(w2),
  .vef4cea(w3)
 );
 v32200d v3c00a7 (
  .v0e28cb(w1),
  .vcbab45(w3)
 );
endmodule

module v70ff7f (
 input vb55943,
 input vef4cea,
 output vc24d9f
);
 wire w0;
 wire w1;
 wire w2;
 assign w0 = vef4cea;
 assign w1 = vb55943;
 assign vc24d9f = w2;
 v70ff7f_v526aa2 v526aa2 (
  .d(w0),
  .clk(w1),
  .q(w2)
 );
endmodule

module v70ff7f_v526aa2 (
 input clk,
 input d,
 output q
);
 // D flip-flop

 reg q = 1'b0;

 always @(posedge clk)
 begin
   q <= d;
 end


endmodule

module v32200d (
 input v0e28cb,
 output vcbab45
);
 wire w0;
```

```
 wire w1;
 assign w0 = v0e28cb;
 assign vcbab45 = w1;
 v32200d_vd54ca1 vd54ca1 (
  .a(w0),
  .c(w1)
 );
endmodule

module v32200d_vd54ca1 (
 input a,
 output c
);
 // NOT logic gate

 assign c = ~ a;
endmodule
```

El código mostrado anteriormente es el que automáticamente genera Icestudio, evitando al diseñador tener que escribirlo. Este código se guarda en un archivo de Verilog que podría ser usado en Icestorm y pudiendo seguir los pasos para la carga en la FPGA. Icestudio al tener incorporado Icestorm dentro de su desarrollo hace esto automáticamente, permitiendo cargar el archivo sin tener que hacer pasos extras.

Como se puede ver Icestudio es prácticamente igual que Icestorm pero mejorando la experiencia del usuario en su uso. Eso ha permitido que con la herramienta de Icestudio cada día se realicen proyectos nuevos y experimentos con FPGAs nuevos.

En el caso de este trabajo final de grado se espera poder conseguir el usar Icestudio con el lenguaje de descripción VHDL, nunca hecho anteriormente. Solo se han intentado hacer pequeños proyectos como el encender un LED con VHDL, pero nunca una conversión completa del programa que permita usar tanto VHDL como Verilog.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Este trabajo final de grado se ha desarrollado para dar más oportunidades a la hora del desarrollo de usos y opciones para el uso de FPGAs. La idea de realizar este trabajo final de grado surgió desde un punto de vista docente. Además del punto de vista docente también se pensó para añadir una ayuda más a los profesionales o aficionados interesados en el uso de FPGAs.

Se podría pensar que la no realización de proyectos similares hasta la fecha es por la falta de necesidad, pero la realidad dista bastante de ello. La razón principal por la que un proyecto como este a gran escala no ha sido realizado antes ha sido simplemente por la falta de recursos.

No hay gran número de traductores entre ambos lenguajes, Verilog y VHDL, esto es porque es complicado de hacer ya que tienen distintos procesos y funciones, siendo el Verilog basado en el lenguaje C mientras que el VHDL está basado en Ada y Pascal.[14]

Debido a esto no hay proyectos anteriores a este pero la necesidad existe ya que ambos lenguajes son válidos para la descripción de sistemas electrónicos digitales, y, por lo tanto, hay diseñadores que prefieren el uso de un lenguaje u otro.

Además de esto, el poder usar ambos lenguajes en una misma aplicación ayudaría mucho a los estudiantes de ambos lenguajes permitiéndoles estudiarlos y usarlos en paralelo, facilitando el aprendizaje de VHDL, ya que este se desarrollaría de una manera más visual, en comparación con las maneras de aprendizaje actuales. Icestudio proporciona la facilidad de aprender y usar un lenguaje de una forma más visual, mediante el desarrollo de código con bloques predefinidos, pudiendo modificarlos o añadir nuevos bloques.

4.2 OBJETIVOS

Los objetivos de este trabajo final de grado se han ido comentando a lo largo de todo el documento, habiendo distintos objetivos que se enumerarán a continuación:

- La modificación del programa Icestudio para permitir un mayor uso de carácter docente para el aprendizaje de ambos lenguajes, VHDL y Verilog, siendo estos los dos más importantes dentro de los lenguajes de descripción.

- La modificación del programa Icestudio para ser más accesible a mayor número de usuarios, permitiendo tanto profesionales como principiantes el poder usarlo y proponiendo una mayor variedad de lenguajes para que el usuario use el que le permita lograr el diseño deseado de la forma más sencilla posible.

- El hacer que las modificaciones no supongan una dificultad añadida a la sencillez a la hora de comprender y usar el programa de Icestudio.

- Analizar la generación de sintetizados mediante Icestudio (y su dependencia Icestorm) y su adaptación a VHDL.

- Valorar y seleccionar las herramientas de código libre de compilación de VHDL para el proyecto.

4.3 METODOLOGÍA

Para llevar a cabo este proyecto se deberá dimensionar la cantidad de archivos a reprogramar. Posteriormente se deberá proceder a su programación y diseño en VHDL de la aplicación para las distintas FPGAs.

Se procederá a reprogramar los bloques en VHDL, además de añadirlos a la barra del menú en una pestaña aparte que permita la fácil selección de estos. Posteriormente se tendrá que hacer un estudio de compiladores y traductores de VHDL a Verilog, eligiendo el más indicado para la correcta realización del programa a incorporar en la FPGA. Siguiendo esto se intentará incorporar al programa y por lo tanto probarlo e intentar que todo el programa funcione correctamente.

Tras esto se llevará a cabo un estudio de costes para poder llevarlo a cabo con el análisis de diversas alternativas. Finalmente se deberá procederá realizar un estudio acerca de la aplicación de este programa en distintos ordenadores.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

A continuación, se procederá a realizar unos costes aproximados de respecto a la realización del proyecto.

Costes Materiales:

Material	Cantidad	Precio (aproximado)
Ordenador	1	700€
FPGA Alhambra II	1	60€
Cable tipo Micro USB	2	5€
Total	4	765€

Tabla 1 - Costes de Materiales

Costes por tareas de ingeniería:

Tareas	Horas	Precio (aproximado)
Búsqueda de Información	10	200€
Aprendizaje de Icestudio	40	800€
Reprogramación de Bloques	40	800€
Estudio del Funcionamiento Interno de Icestudio	150	3000€
Búsqueda de Traductor VHDL-Verilog	10	200€
Desarrollo del compilador	20	400€
Total	270	5400€

Tabla 2 - Costes por Tarea de Ingeniería

El coste de un ingeniero se ha estimado que sea de 20€/hora. Esto significa que el coste del proyecto tal y cómo se presenta en este documento sería de 6165€. Teniendo en cuenta que todos estos costes son aproximados.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

Para el desarrollo de este trabajo fin de grado se ha contado con la ayuda de un experto, siendo este el que ha ayudado a establecer la correcta dirección y abordamiento de este trabajo.

Se planteó abordarlo de distintas maneras, las cuales se explicarán y desarrollarán a lo largo de este capítulo, ya que son importantes para la decisión final de desarrollo y a pesar de no haber sido elegidas eran perfectamente validas y podrían ser interesantes de desarrollo también.

Este capítulo se va a dividir en varias secciones en las que se va a ir relatando cuál ha sido el procedimiento para desarrollar la modificación del programa Icestudio.

Primero se tratará del estudio y aprendizaje del programa, procediendo así a desarrollar la estrategia de trabajo. Debido a que hay muchos códigos se adjuntarán en el Anexo A debidamente indicados. Como gran parte del trabajo requiere comprensión del código, no se extenderá demasiado la redacción de estos apartados.

A continuación, se procederá a la descripción de bloques, y por lo tanto su debido desarrollo en VHDL, el cual se ubicará en el Anexo A. Y por último se procederá a la explicación del traductor elegido para usar.

5.1 ANÁLISIS DE ICESTUDIO

En este apartado se procederá ha estudiar las “tripas” del programa Icestudio, procediendo a explicar ciertos trozos del código desarrollado. Se adjuntarán en esta parte solo partes del

dicho código, pudiendo verlo más desarrollado en los Anexos. A pesar de esto, no se incluirá el código completo de Icestudio ya que este ocuparía demasiadas páginas.

Para generar el archivo de Verilog y el archivo de configuración PCF, se usa la siguiente parte de código:

```
this.generate = function (target, project, opt) {
      var content = '';
      var files = [];
      switch (target) {
        case 'verilog':
          content += header('//', opt);
          content += '`default_nettype none\n\n';
          content += verilogCompiler('main', project, opt);
          files.push({
            name: 'main.v',
            content: content
          });
          break;
        case 'pcf':
          content += header('#', opt);
          content += pcfCompiler(project, opt);
          files.push({
            name: 'main.pcf',
            content: content
          });
          break;
        case 'lpf':
          content += header('#', opt);
          content += lpfCompiler(project, opt);
          files.push({
            name: 'main.lpf',
            content: content
          });
          break;

        case 'list':
          files = listCompiler(project);
          break;
        case 'testbench':
          content += header('//', opt);
          content += testbenchCompiler(project);
          files.push({
            name: 'main_tb.v',
            content: content
          });
          break;
```

```
        case 'gtkwave':
          content += header('[*]', opt);
          content += gtkwaveCompiler(project);
          files.push({
            name: 'main_tb.gtkw',
            content: content
          });
          break;
        default:
          break;
      }
      return files;
    };
```

El código muestra un switch-case declarado en JS que decide que tipo de archivo se genera y con que proceso dependiendo de lo llamado en código principal. El tener las opciones de GTKwave y Testbench es debido a que Icestudio permite exportar versiones del programa escrito por el usuario en ambos para poder ser usados en otros entornos.

Si nos fijamos en la función verilogCompiler la cual se adjuntará en el ANEXO I – Código de Compilado, junto con todas las líneas necesarias para el compilado podemos destacar las siguientes partes de la función. Primero se procede a la carga de los bloques y dependencias, seguido de la inicialización de los puertos de entrada, seguido de la búsqueda en el diseño por bloques desarrollado por el usuario de los bloques de entrada y, por lo tanto, la creación de cables imaginarios para la conexión con el resto de los bloques. Después se inicializa de forma análoga los puertos de salida y los bloques de salida con sus cables. Por último, se cargan los bloques intermedios con una sentencia For, gracias a las dependencias ya cargadas anteriormente en el programa, a excepción de los bloques de código que se cargan por separado ya que se tienen que cargar los parámetros y puertos, pudiendo ser estos diferentes para cada diseño de bloques hecho por el usuario.

Otras funciones importantes dentro del ANEXO I – Código de Compilado son:

- GetContent: Para poder compilar hay que crear tanto Labels, nombres, como Wires, cables, virtuales y reordenar las especificaciones internas para poder compilarlo. Primero se identifican las fuentes y objetivos y se organizan en una tabla para una

búsqueda de estos más fácil. Después se crean los cables virtuales. Con las especificaciones típicas, longitud, salida y llegada (fuente y objetivo). Se procede después a guardar correctamente tanto las entradas como salidas con cables temporales. Luego se hacen las conexiones de los cables. Se borran los cables temporales y se vuelve al grafico inicial.

- Module: La función module tiene distintos apartados. Es para comprobar y guardar todos los datos de una lista o file que se le envía. Se comprueba que el dato tenga valores y tras esto se pasa a guardar los parámetros. La función insteadof solo te devuelve un verdadero o falso de lo que le envíes. Tras esto se pasará a guardar los puertos, haciéndose de forma similar, siendo la única diferencia que se guardan los puertos de entrada y salida en dos subapartados de la variable ports. Si tanto la longitud de los parámetros como la de los puertos es estrictamente igual a 0 se pone un salto de línea. Se procede a guardar ahora el contenido. Se guarda el contenido con la función 'foreach' declarada en app/resources/plugins/serial-term/js/xterm/lib/xterm.js (siendo este un enlace al código de dicha función en github) . Por último, se escribe el pie de página o footer.

Siendo estas dos las más complicadas de comprender, habiendo también muchas otras funciones más sencillas las cuales siendo tanto un nombre como el desarrollo del código bastante intuitivo.

Continuando con las funciones para generar código en los lenguajes anteriores también hay que añadir la siguiente, que conjuntamente con las anteriores son capaces de generar el código en las distintas versiones y lenguajes deseados:

```
function generateCode(cmd) {
              return new Promise(function(resolve) {
                     project.snapshot(), project.update();
                     var opt = {
                            datetime: !1,
                            boardRules: profile.get("boardRules")
                     };
                     opt.boardRules && (opt.initPorts =
compiler.getInitPorts(project.get()), opt.initPins =
compiler.getInitPins(project.get()));
```

```
                    var verilogFile = compiler.generate("verilog", project.get(),
opt)[0];
                    if (nodeFs.writeFileSync(nodePath.join(common.BUILD_DIR,
verilogFile.name), verilogFile.content, "utf8"), cmd.indexOf("verify") > -1 &&
=== cmd.length);
                    else {
                          var pcfFile = compiler.generate("pcf", project.get(),
opt)[0];
                          nodeFs.writeFileSync(nodePath.join(common.BUILD_DIR,
pcfFile.name), pcfFile.content, "utf8")
                    }
                    var listFiles = compiler.generate("list", project.get());
                    for (var i in listFiles) {
                          var listFile = listFiles[i];
                          nodeFs.writeFileSync(nodePath.join(common.BUILD_DIR,
listFile.name), listFile.content, "utf8")
                    }
                    project.restoreSnapshot(), resolve({
                          code: verilogFile.content,
                          internalResources: listFiles.map(function(res) {
                                return res.name
                          })
                    })
              })
      }
```

Se mostrarán en el ANEXO II – Código de las funciones necesarias para que función correctamente los siguientes apartados de Icestudio de la sección de TOOLS.

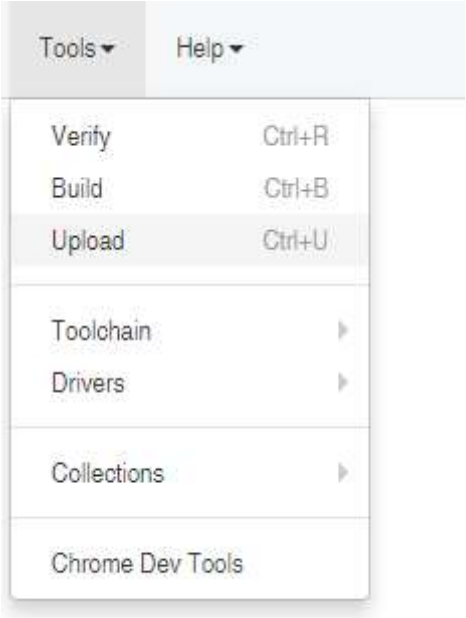

Figura 12 - Sección de Tools Icestudio

5.2 FUNCIONES AÑADIDAS EN VHDL

Debido a que los bloques de la sección Basic están implementados para cada FPGA de manera especifica y se espera poder mezclar tanto bloques en VHDL como en Verilog estos no se habrán modificado. Al igual que estos, los bloques de la sección de Setup tampoco se modificarán debido a que estos no tienen tanto que ver con la descripción por bloques, que es lo que se plantea enseñar. Por lo tanto, se habrán descrito todos los bloques de la sección Bit y Logic, agrupándolos ahora en una sección de Bloques VHDL como ya se ha mostrado en capítulos anteriores.

A continuación, se mostrará tanto el código como el símbolo que aparecerá en el diseño del bloque AND. Todos los códigos y símbolos se encuentran en el ANEXO III – Código de los Bloques en VHDL.

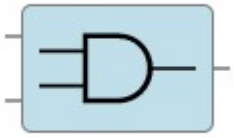

Figura 13 - Bloque AND

```
// AND logic gate

entity AND_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end AND_VHDL;
architecture behavioral of And_VHDL is
begin
	c <= a AND b;
end behavioral;
```

Se ha añadido la definición de entradas y salidas dentro del bloque pero se podría hacer como los bloques en Verilog que no se definen dentro del bloque ya que se hace esta definición dentro del código completo del programa. Esto es por mayor sencillez de los bloques, de manera visual. Debido a que esta modificación está orientada a que diseñadores noveles puedan usarlo y aprender de ello, se ha decidido mostrar las declaraciones para que puedan usarse como base de aprendizaje de cara al desarrollo de códigos más complicados en un futuro.

5.3 DISEÑO DEL MENÚ Y ACCESOS

Para el diseño del menú se planteó seguir la misma línea que ya tenía, tan solo añadiendo una pestaña más donde se pudiera acceder a todas las opciones nueva, enumeradas en el apartado anterior e incluidas en el ANEXO III – Código de los Bloques en VHDL.

Para ello se creó el subapartado llamado “VHDL Blocks”, al cual se puede acceder desde la barra superior junto a los otros bloques existentes o desde el apartado de file. Donde

pinchando el apartado de Blocks se podría acceder a este. Ambas ubicaciones se muestran en las Figura 14 y Figura 15.

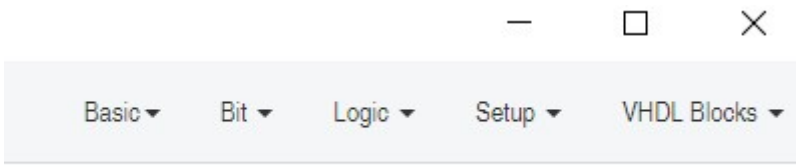

Figura 14 - Acceso a los Bloques desde la barra superior

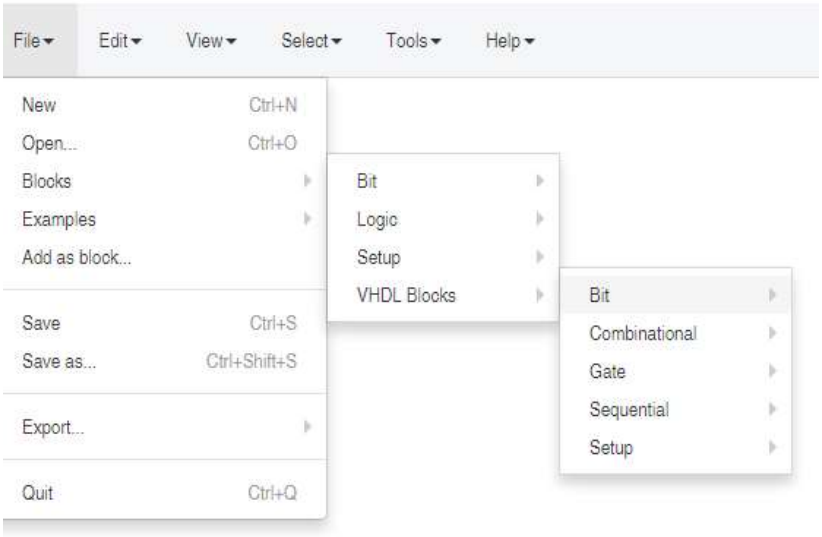

Figura 15 - Acceso a los Bloques desde File

Como se puede observar en ambas figuras el diseño original se mantiene, con el único cambio el que añade los bloques en VHDL, todo esto sin cambiar el formato ya impuesto por los creadores. El código completo de este menú está descrito mayormente en HTML y este se puede revisar en el ANEXO IV – Código del Menú.

5.4 TRADUCTOR VHDL A VERILOG

A la hora de la elección de traductor de VHDL a Verilog se ha pensado en distintos traductores, llegando a dos opciones, consideradas las mejores. Estas opciones son GHDL o VHD2VL.

5.4.1 GHDL

GHDL es una abreviatura de G Hardware Design Language. Es un analizador, compilador, simulador y sintetizador de VHDL que puede procesar gran cantidad de diseños de VHDL. GHDL compila directamente de VHDL. Para ello en el diseño de GHDL lo que se hizo fue modificar el programa Yosys ya comentado anteriormente. Este programa solo acepta código en Verilog y genera un archivo BLIF que luego junto Aracne PNR, se consigue llegar al archivo Bitstream necesario para programar la FPGA. GHDL funciona de similar forma, con la pequeña diferencia que el programa Yosys el cual solo entiende Verilog, ha sido modificado para que entienda VHDL.

Como se puede observar con la herramienta GHDL, que incluye una versión experimental de Yosys, para aceptar código en VHDL y así compilarlo. Debido a esto, con unas pequeñas modificaciones se podría lograr que Yosys entienda y compile en ambos lenguajes indistintamente, estando así un paso más cerca de que Icestudio permita el diseño en ambos lenguajes.

Otra opción es VHD2VL, el cual es un traductor más clásico que permite el pasar un código completo de VHDL a Verilog. Para el uso de este habría que hacerlo de forma externa o programando la aplicación para que cuando vaya a compilar lo llame antes y así generar un único archivo de Verilog para poder proseguir así con los pasos indicados en el funcionamiento de Icestorm Project.

Para la ejecución de VHD2VL es necesario tener descargado en el ordenador tanto Flex como GNU Bison.

5.4.2 FLEX

Flex es un “fast lexical analyzer generator” de código libre, que sirve como alternativa a Lex. En un programa que genera scanners o lexers. Es un programa descrito en C que lo que hace y consigue es convertir un conjunto de caracteres, como los que hay en un programa informático o una página web, a una secuencia de tokens, siendo estos una cadena con un significado asignado y por lo tanto identificado. Estos programas realizan análisis sintáctico

y tokenización de caracteres mediante el uso de un tipo de máquina de estados, llamada Deterministic Finite Automata, o en español, autómata finito determinista (DFA).

Es una herramienta para generar programas que realizan coincidencia de patrones en el texto. Flex tiene una gran variedad de utilidades en su uso, incluida la escritura de compiladores junto con GNU Bison. Flex es una implementación gratuita del conocido programa Lex.

En este caso Flex tomará el uso junto a Bison de crear un compilador para el código generado con el VHD2VL, permitiendo así su uso y la conversión correcta de VHDL a Verilog.

5.4.3 GNU BISON

GNU Bison, es un programa generador de analizadores sintácticos que forma parte del Proyecto GNU, disponible para la mayoría de los sistemas operativos. Bison convierte la descripción formal de un lenguaje, escrita como una gramática libre de contexto LALR, en un programa en C, C++, o Java que realiza análisis sintáctico. Lee las secuencias de tokens y decide si la secuencia se ajusta a la sintaxis especificada por la gramática. Los analizadores generados son portátiles: no requieren ningún compilador específico. Bison genera por defecto LALR parsers, siendo estos unos analizadores look-ahead de LR, pero también puede generar analizadores LR canónico, IELR (1) y GLR.

Como se puede observar en la Figura 16 el proceso de funcionamiento de Icestudio con estos cambios sería el siguiente.

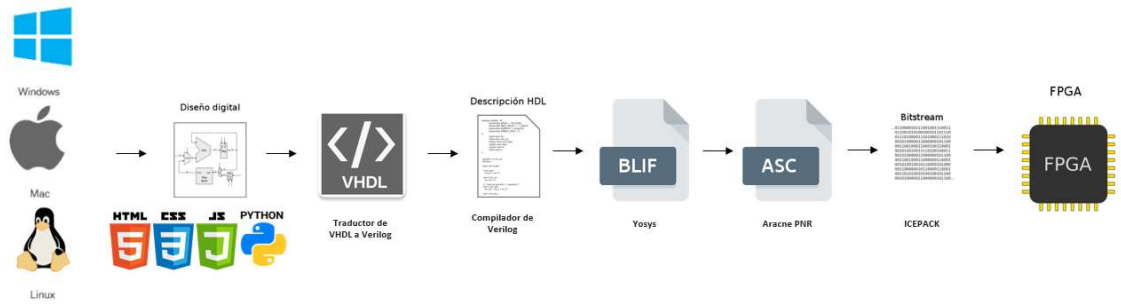

Figura 16 - Esquema del funcionamiento de Icestudio modificado

.

5.5 ANÁLISIS DE RESULTADOS

Como se puede observar en los apartados anteriores, el desarrollo de este trabajo fin de grado, ha tenido buenos resultados, pudiendo avanzar enormemente en la compatibilización de la aplicación Icestudio con VHDL.

La descripción de los bloques en VHDL es bastante correcta y ha sido probada y depurada, comprobando el correcto funcionamiento de estos. Para ello se usó las herramientas Sublime3 y Quartus II, las cuales permiten la descripción de código. Además Quartus permite la compilación, pudiendo así comprobar el correcto funcionamiento de los bloques.

Además, hay que tener en cuenta que dichos bloques se han podido incluir en el programa, de una forma sutil y de fácil acceso para el usuario, manteniendo la estética del programa. Consiguiendo así una clara implementación de estos.

Por último, cabe destacar, el uso de VHD2VL, que permite de forma sencilla, traducir los códigos en VHDL a Verilog, pudiendo así dar un paso importante en la obtención del programa final.

Obteniendo el resultado final del programa el diseño que muestra Figura 17 a continuación. En dicha figura se puede observar como la estética es similar añadiendo el desplegable en el que se permite seleccionar bloques de VHDL. El diseño mostrado es tan solo un ejemplo genérico de como quedaría un diseño en la modificación de la ventana.

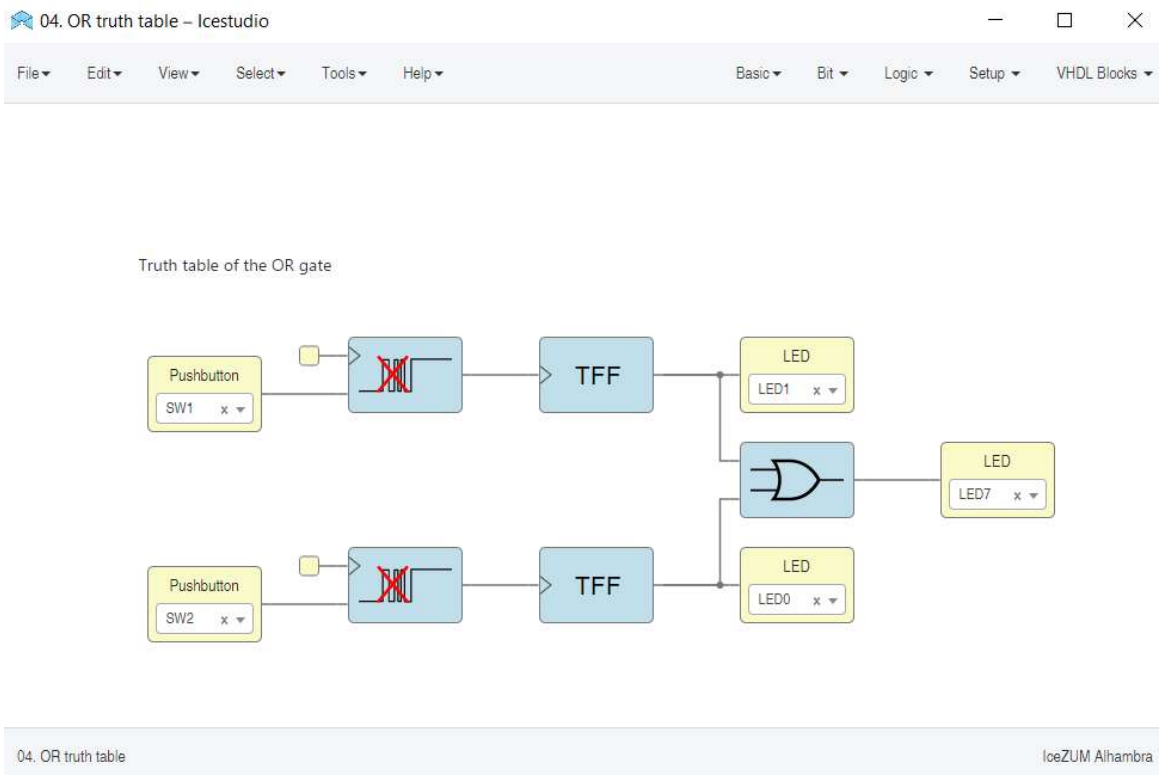

Figura 17 - Programa de Icestudio Final

Capítulo 6. CONCLUSIONES Y TRABAJOS FUTUROS

Para concluir este trabajo final de grado, se van a resaltar primero los objetivos iniciales, si se han cumplido estos y por último los futuros trabajos.

El principal objetivo era la modificación de la herramienta Icestudio para el posible uso de dos lenguajes de descripción. Estos lenguajes son VHDL y Verilog, usados para la descripción de FPGAs.

Dicha modificación se buscaba para dar más herramientas, a los docentes para la explicación por parte de estos y aprendizaje por parte de los alumnos de ambos lenguajes. Además, permitiendo a usuarios autodidactas usarlo sin tener complicaciones y con gran facilidad de aprendizaje, manteniendo siempre el ser un programa muy intuitivo de usar.

Como objetivo secundario era dar más opciones a los diseñadores más expertos, ya que como se ha explicado, ambos lenguajes tienen sus diferencias y por lo tanto también pueden tener distintos usos. Estos diseñadores más experimentados podrían acceder a realizar códigos o diseños más complejos mezclando ambos lenguajes.

Según se ha descrito en los apartados anteriores se ha llegado a cumplir dichos objetivos en gran medida. Completando con éxito la descripción de todos los bloques del programa en ambos lenguajes, desarrollando estos con anotaciones y también códigos completos y fáciles de leer, para que los usuarios más inexpertos los puedan usar como apoyo para aprender.

Así como también se han podido implementar en la barra superior y en el apartado de file, los bloques nuevos desarrollados en VHDL. Manteniendo la accesibilidad y diseño original del programa.

Por último, se ha incluido el desarrollo y explicación de la herramienta de traducción VHDTOVL y de GHDL, que permiten traducir las partes de código en VHDL a Verilog, permitiendo así el acercamiento al desarrollo completo de la aplicación en ambos lenguajes, y por lo tanto, el cumplimiento de los objetivos marcados al inicio del proyecto.

Este trabajo final de grado ha abierto las posibilidades a infinidad de futuros proyectos, así como la unificación de todas las partes enumeradas y por lo tanto el completo logro de los objetivos de este, como el trabajo y desarrollo de las FPGAs como procesadores que pueden ser descritos con programas de código libre y por lo tanto para el uso de cualquier usuario. Además, también permite la opción de en lugar de añadir programas externos para la traducción de VHDL a Verilog, también hay la opción de modificar el código de compilación de Icestudio permitiendo la unión de ambos antes de llegar a la etapa en la que actúa Yosys. Esta idea se puede observar en el esquema mostrado en la Figura 18.

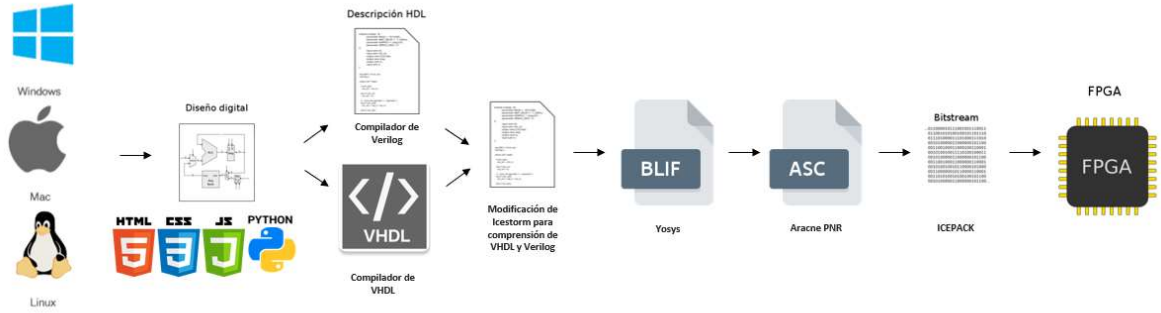

Figura 18 - Modificación de Icestudio para la compresión de VHDL y Verilog sin traductorees

Con estas posibilidades se plantea y espera la obtención de más proyectos con FPGAs, así como su desarrollo. Intentando llegar al nivel de desarrollo de los microcontroladores como Arduino. Permitiendo al usuario tener un mayor número de herramientas para usar en el desarrollo de sus proyectos personales como profesionales.

Capítulo 7. BIBLIOGRAFÍA

[1] GitHub. 2021. FPGAwars/icestudio. [online] Disponible en: <https://github.com/FPGAwars/icestudio> [Último acceso 25 de Junio de 2021].

[2] Alhambra Bits. Icestudio IDE. [online] Disponible en: <https://alhambrabits.com/software/#:~:text=Icestudio%20is%20a%20graphic%20IDE,into%20the%20FPGA%20hardware%20board.> [Último acceso 25 de Junio de 2021].

[3] Codrops, C., 2021. Icestudio. [online] Icestudio.io. Disponible en: <https://icestudio.io/> [Último acceso 25 de Junio de 2021].

[4] Clifford.at. 2021. Project IceStorm. [online] Disponible en: <http://www.clifford.at/icestorm/#:~:text=Project%20IceStorm%20aims%20at%20documenting,Bitstream%20flow%20for%20iCE40%20FPGAs.> [Último Acceso 25 de Junio de 2021].

[5] Es.wikipedia.org. 2021. Verilog. [online] Disponible en: <https://es.wikipedia.org/wiki/Verilog> [Último acceso 25 de Junio de 2021].

[6] Daroch Montoya, D. and Antiñire Monje, S., [online] Repositorio.usm.cl. Disponible en: <https://repositorio.usm.cl/bitstream/handle/11673/46840/3560901543783UTFSM.pdf?sequence=1&isAllowed=y> [Último acceso 25 de Junio de 2021].

[7] Es.wikipedia.org. 2021. Field-programmable gate array. [online] Disponible en: <https://es.wikipedia.org/wiki/Field-programmable_gate_array#Aplicaciones> [Último acceso 25 de Junio de 2021].

[8] Fpgawars.github.io. 2021. FPGAwars. [online] Disponible en: <http://fpgawars.github.io/> [Último acceso 25 de Junio de 2021].

[9] Obijuan.github.io. 2021. FPGAwars. [online] Disponible en: <https://obijuan.github.io/intro-fpga.html> [Último acceso 25 de Junio de 2021].

[10] Planeta ChatBot. 2021. Qué es una FPGA y por qué jugarán un papel clave en el futuro. [online] Disponible en: <https://planetachatbot.com/que-es-fpga-y-por-que-jugaran-papel-clave-en-futuro/> [Último acceso 25 de Junio de 2021].

[11] strephonsays. 2021. ¿Cuál es la diferencia entre Verilog y VHDL?. [online] Disponible en: <https://es.strephonsays.com/what-is-the-difference-between-verilog-and-vhdl#menu-3> [Último acceso 25 de Junio de 2021].

[12] En.wikipedia.org. 2021. GNU Bison. [online] Disponible en: <https://en.wikipedia.org/wiki/GNU_Bison> [Último acceso 25 de Junio de 2021].

[13] Ríos Santillán, I., 2021. Diseño Electrónico en FPGA con herramientas de Software Libre. [online] Ebuah.uah.es. Disponible en: <https://ebuah.uah.es/xmlui/bitstream/handle/10017/39326/TFG_Rios_Santillan_2019.pdf?sequence=1&isAllowed=y> [Último Acceso 25 de Junio de 2021].

[14] David R. Coelho, 1989. The VHDL Handbook. [online] Disponible en: <https://books.google.es/books?id=TssFCAAAQBAJ> [Último Acceso 25 de Junio de 2021].

ANEXO I – CÓDIGO DE COMPILADO

```
'use strict';

angular.module('icestudio')
  .service('compiler', function (common,
    utils,
    nodeSha1,
    _package) {

    this.generate = function (target, project, opt) {
      var content = '';
      var files = [];
      switch (target) {
        case 'verilog':
          content += header('//', opt);
          content += '`default_nettype none\n\n';
          content += verilogCompiler('main', project, opt);
          files.push({
            name: 'main.v',
            content: content
          });
          break;
        case 'pcf':
          content += header('#', opt);
          content += pcfCompiler(project, opt);
          files.push({
            name: 'main.pcf',
            content: content
          });
          break;
        case 'lpf':
          content += header('#', opt);
          content += lpfCompiler(project, opt);
          files.push({
            name: 'main.lpf',
            content: content
          });
          break;

        case 'list':
          files = listCompiler(project);
          break;
        case 'testbench':
          content += header('//', opt);
          content += testbenchCompiler(project);
          files.push({
            name: 'main_tb.v',
            content: content
```

```
          });
          break;
        case 'gtkwave':
          content += header('[*]', opt);
          content += gtkwaveCompiler(project);
          files.push({
            name: 'main_tb.gtkw',
            content: content
          });
          break;
        default:
          break;
      }
      return files;
    };

    function header(comment, opt) {
      var header = '';
      var date = new Date();
      opt = opt || {};
      if (opt.header !== false) {
        header += comment + ' Code generated by Icestudio ' + _package.version +
'\n';
        if (opt.datetime !== false) {
          header += comment + ' ' + date.toUTCString() + '\n';
        }
        header += '\n';
      }
      return header;
    }

    function module(data) {
      var code = '';
      if (data && data.name && data.ports) {

        // Header
        code += '\nmodule ' + data.name;

        //-- Parameters

        var params = [];
        for (var p in data.params) {
          if (data.params[p] instanceof Object) {
            params.push(' parameter ' + data.params[p].name + ' = ' +
(data.params[p].value ? data.params[p].value : '0'));
          }
        }

        if (params.length > 0) {
          code += ' #(\n';
          code += params.join(',\n');
          code += '\n)';
        }
```

```
        //-- Ports

        var ports = [];
        for (var i in data.ports.in) {
          var _in = data.ports.in[i];
          ports.push(' input ' + (_in.range ? (_in.range + ' ') : '') +
_in.name);
        }
        for (var o in data.ports.out) {
          var _out = data.ports.out[o];
          ports.push(' output ' + (_out.range ? (_out.range + ' ') : '') +
_out.name);
        }

        if (ports.length > 0) {
          code += ' (\n';
          code += ports.join(',\n');
          code += '\n)';
        }

        if (params.length === 0 && ports.length === 0) {
          code += '\n';
        }

        code += ';\n';

        // Content

        if (data.content) {

          var content = data.content.split('\n');

          content.forEach(function (element, index, array) {
            array[index] = ' ' + element;
          });

          code += content.join('\n');
        }

        // Footer

        code += '\nendmodule\n';
      }

      return code;
    }

    function digestNameBlock(block){
        let id=block;

       if(typeof block.id !== 'undefined'){
          id=utils.digestId(block.id);
```

```
    if(typeof block.data !== 'undefined' &&
       typeof block.data.name !== 'undefined' &&
       block.data.name.trim() !== ''){
         id=`${id}__${block.data.name}`;
       }
     }
     return id.replace(/(-)|(\s)/g,'');
}

function getParams(project) {
  var params = [];
  var graph = project.design.graph;

  for (var i in graph.blocks) {
    var block = graph.blocks[i];

    if (block.type === 'basic.constant') {
      params.push({
        name: utils.digestId(block.id),
        value: block.data.value
      });
    } else if (block.type === 'basic.memory') {
      let name =  utils.digestId(block.id);

      params.push({
        name: name,
        value: '"' + name + '.list"'
      });
    }
  }

  return params;
}

function getPorts(project) {
  var ports = {
    in: [],
    out: []
  };
  var graph = project.design.graph;

  for (var i in graph.blocks) {
    var block = graph.blocks[i];
    if (block.type === 'basic.input') {

      ports.in.push({
        name: utils.digestId(block.id),
        range: block.data.range ? block.data.range : ''
      });
    } else if (block.type === 'basic.output') {
      ports.out.push({
        name: utils.digestId(block.id),
```

```
            range: block.data.range ? block.data.range : ''
          });
        }
      }

      return ports;
    }


    function getContent(name, project) {
      var i, j, w;
      var content = [];
      var graph = project.design.graph;
      var connections = {
        localparam: [],
        wire: [],
        assign: []
      };
      // We need to rearrange internal design specification to compile it
      // Convert virtual labels to wire and some stuff .

      var vwiresLut = {};
      var lidx, widx, lin, vw;
      var twire;

      //Create virtual wires

      //First identify sources and targets and create a look up table to work
easy with it
      if (typeof graph !== 'undefined' &&
        graph.blocks.length > 0 &&
        graph.wires.length > 0) {

        for (lidx in graph.blocks) {
          lin = graph.blocks[lidx];
          if (lin.type === 'basic.inputLabel') {
            for (widx in graph.wires) {
              vw = graph.wires[widx];
              if (vw.target.block === lin.id) {
                if (typeof vwiresLut[lin.data.name] === 'undefined') {
                  vwiresLut[lin.data.name] = {
                    source: [],
                    target: []
                  };
                }
                twire = vw.source;
                twire.size = vw.size;
                vwiresLut[lin.data.name].source.push(twire);
              }
            }
          }
          if (lin.type === 'basic.outputLabel') {
            for (widx in graph.wires) {
```

```
            vw = graph.wires[widx];
            if (vw.source.block === lin.id) {
              if (typeof vwiresLut[lin.data.name] === 'undefined') {
                vwiresLut[lin.data.name] = {
                  source: [],
                  target: []
                };
              }

              twire = vw.target;
              twire.size = vw.size;
              vwiresLut[lin.data.name].target.push(twire);
            }
          }
        }
      } //for lin
    } // if typeof....

    //Create virtual wires
    for (widx in vwiresLut) {
      vw = vwiresLut[widx];
      if (vw.source.length > 0 && vw.target.length > 0) {
        for (var vi = 0; vi < vw.source.length; vi++) {
          for (var vj = 0; vj < vw.target.length; vj++) {
            graph.wires.push({
              tcTodelete: true,
              size: vw.size,
              source: vw.source[vi],
              target: vw.target[vj],
              vertices: undefined
            });
          }
        }
      }
    }

    // Remove virtual blocks
    // Save temporal wires and delete it

    graph.wiresVirtual = [];
    var wtemp = [];
    var iwtemp;
    var wi;
    for (wi = 0; wi < graph.wires.length; wi++) {

      if (graph.wires[wi].source.port === 'outlabel' ||
        graph.wires[wi].target.port === 'outlabel' ||
        graph.wires[wi].source.port === 'inlabel' ||
        graph.wires[wi].target.port === 'inlabel') {

        graph.wiresVirtual.push(graph.wires[wi]);

      } else {
```

```
          iwtemp = graph.wires[wi];
          if (typeof iwtemp.source.size !== 'undefined') {
            iwtemp.size = iwtemp.source.size;
          }
          wtemp.push(iwtemp);
        }
      }
      graph.wires = utils.clone(wtemp);
      // End of rearrange design connections for compilation


      for (w in graph.wires) {
        var wire = graph.wires[w];
        if (wire.source.port === 'constant-out' ||
          wire.source.port === 'memory-out') {
          // Local Parameters
          var constantBlock = findBlock(wire.source.block, graph);
          var paramValue = utils.digestId(constantBlock.id);
          if (paramValue) {
            connections.localparam.push('localparam p' + w + ' = ' + paramValue +
';');
          }
        } else {
          // Wires
          var range = wire.size ? ' [0:' + (wire.size - 1) + '] ' : ' ';
          connections.wire.push('wire' + range + 'w' + w + ';');
        }
        // Assignations
        for (i in graph.blocks) {
          var block = graph.blocks[i];
          if (block.type === 'basic.input') {
            if (wire.source.block === block.id) {
              connections.assign.push('assign w' + w + ' = ' +
utils.digestId(block.id)+ ';');
            }
          } else if (block.type === 'basic.output') {
            if (wire.target.block === block.id) {
              if (wire.source.port === 'constant-out' ||
                wire.source.port === 'memory-out') {
                // connections.assign.push('assign ' + digestId(block.id) + ' =
p' + w + ';');
              } else {
                connections.assign.push('assign ' + utils.digestId(block.id) + '
= w' + w + ';');
              }
            }
          }
        }
      }

      content = content.concat(connections.localparam);
      content = content.concat(connections.wire);
      content = content.concat(connections.assign);
```

```
// Wires Connections

var numWires = graph.wires.length;
var gwi, gwj;
for (i = 1; i < numWires; i++) {
  for (j = 0; j < i; j++) {
    gwi = graph.wires[i];
    gwj = graph.wires[j];
    if (gwi.source.block === gwj.source.block &&
      gwi.source.port === gwj.source.port &&
      gwi.source.port !== 'constant-out' &&
      gwi.source.port !== 'memory-out') {
      content.push('assign w' + i + ' = w' + j + ';');
    }
  }
}

// Block instances

content = content.concat(getInstances(name, project.design.graph));

// Restore original graph
// delete temporal wires
//

wtemp = [];
var wn = 0;
for (wi = 0, wn = graph.wiresVirtual.length; wi < wn; wi++) {
  if (typeof graph.wiresVirtual[wi].tcTodelete !== 'undefined' &&
    graph.wiresVirtual[wi].tcTodelete === true) {
    //Nothing for now, only remove
  } else {
    wtemp.push(graph.wiresVirtual[wi]);
  }

}

for (wi = 0, wn = graph.wires.length; wi < wn; wi++) {
  if (typeof graph.wires[wi].tcTodelete !== 'undefined' &&
    graph.wires[wi].tcTodelete === true) {
    //Nothing for now, only remove
  } else {
    wtemp.push(graph.wires[wi]);
  }

}

graph.wires = wtemp;

delete graph.wiresVirtual;
//END ONWORK
```

```
    return content.join('\n');
  }

  function getInstances(name, graph) {
    var w, wire;
    var instances = [];
    var blocks = graph.blocks;

    for (var b in blocks) {
      var block = blocks[b];

      if (block.type !== 'basic.input' &&
        block.type !== 'basic.output' &&
        block.type !== 'basic.constant' &&
        block.type !== 'basic.memory' &&
        block.type !== 'basic.info' &&
        block.type !== 'basic.inputLabel' &&
        block.type !== 'basic.outputLabel') {

        // Header
        var instance;
        if (block.type === 'basic.code') {
          instance = name + '_' + utils.digestId(block.id);
        } else {
          instance = utils.digestId(block.type);

        }

        //-- Parameters

        var params = [];
        for (w in graph.wires) {
          wire = graph.wires[w];
          if ((block.id === wire.target.block) &&
            (wire.source.port === 'constant-out' ||
              wire.source.port === 'memory-out')) {
            var paramName = wire.target.port;
            if (block.type !== 'basic.code') {
              paramName =  utils.digestId(paramName);
            }
            var param = '';
            param += ' .' + paramName;
            param += '(p' + w + ')';
            params.push(param);
          }
        }

        if (params.length > 0) {
          instance += ' #(\n' + params.join(',\n') + '\n)';
        }

        //-- Instance name
```

```
        instance += ' ' +  utils.digestId(block.id) ;

        //-- Ports

        var ports = [];
        var portsNames = [];
        for (w in graph.wires) {
          wire = graph.wires[w];
          if (block.id === wire.source.block) {
            connectPort(wire.source.port, portsNames, ports, block);
          }
          if (block.id === wire.target.block) {
            if (wire.source.port !== 'constant-out' &&
              wire.source.port !== 'memory-out') {
              connectPort(wire.target.port, portsNames, ports, block);
            }
          }
        }

        instance += ' (\n' + ports.join(',\n') + '\n);';

        if (instance) {
          instances.push(instance);
        }
      }
    }

    function connectPort(portName, portsNames, ports, block) {
      if (portName) {
        if (block.type !== 'basic.code') {
          portName =  utils.digestId(portName);
        }
        if (portsNames.indexOf(portName) === -1) {
          portsNames.push(portName);
          var port = '';
          port += ' .' + portName;
          port += '(w' + w + ')';
          ports.push(port);
        }
      }
    }

    return instances;
  }

  function findBlock(id, graph) {
    for (var b in graph.blocks) {
      if (graph.blocks[b].id === id) {
        return graph.blocks[b];
      }
    }
    return null;
  }
```

```
this.getInitPorts = getInitPorts;

function getInitPorts(project) {
  // Find not connected input wire ports to initialize

  var i, j;
  var initPorts = [];
  var blocks = project.design.graph.blocks;

  // Find all not connected input ports:
  // - Code blocks
  // - Generic blocks
  for (i in blocks) {
    var block = blocks[i];
    if (block) {
      if (block.type === 'basic.code' || !block.type.startsWith('basic.')) {
        // Code block or Generic block
        for (j in block.data.ports.in) {
          var inPort = block.data.ports.in[j];
          if (inPort.default && inPort.default.apply) {
            initPorts.push({
              block: block.id,
              port: inPort.name,
              name: inPort.default.port,
              pin: inPort.default.pin
            });
          }
        }
      }
    }
  }

  return initPorts;
}

this.getInitPins = getInitPins;

function getInitPins(project) {
  // Find not used output pins to initialize

  var i;
  var initPins = [];
  var usedPins = [];
  var blocks = project.design.graph.blocks;

  // Find all set output pins
  for (i in blocks) {
    var block = blocks[i];
    if (block.type === 'basic.output') {
      for (var p in block.data.pins) {
        usedPins.push(block.data.virtual ? '' : block.data.pins[p].value);
      }
```

```
      }
    }

    // Filter pins defined in rules
    var allInitPins = common.selectedBoard.rules.output;
    for (i in allInitPins) {
      if (usedPins.indexOf(allInitPins[i].pin) === -1) {
        initPins.push(allInitPins[i]);
      }
    }

    return initPins;
  }

  function verilogCompiler(name, project, opt) {
    var i, data, block, code = '';
    opt = opt || {};

    if (project &&
      project.design &&
      project.design.graph) {

      var blocks = project.design.graph.blocks;
      var dependencies = project.dependencies;

      // Main module

      if (name) {

        // Initialize input ports

        if (name === 'main' && opt.boardRules) {

          var initPorts = opt.initPorts || getInitPorts(project);
          for (i in initPorts) {
            var initPort = initPorts[i];

            // Find existing input block with the initPort value
            var found = false;
            var source = {
              block: initPort.name,
              port: 'out'
            };
            for (i in blocks) {
              block = blocks[i];
              if (block.type === 'basic.input' &&
                !block.data.range &&
                !block.data.virtual &&
                initPort.pin === block.data.pins[0].value) {
                found = true;
                source.block = block.id;
                break;
              }
```

```
            }

            if (!found) {
              // Add imaginary input block with the initPort value
              project.design.graph.blocks.push({
                id: initPort.name,
                type: 'basic.input',
                data: {
                  name: initPort.name,
                  pins: [{
                    index: '0',
                    value: initPort.pin
                  }],
                  virtual: false
                }
              });
            }

            // Add imaginary wire between the input block and the initPort
            project.design.graph.wires.push({
              source: {
                block: source.block,
                port: source.port
              },
              target: {
                block: initPort.block,
                port: initPort.port
              }
            });
          }
        }

        var params = getParams(project);

        //-- Future improvement: After reading the ports, get the names defined
in the .pcf or .lpf file
        //-- these are better names to use in the top module, instead of the
current not-for-humans pin names
        var ports = getPorts(project);

        var content = getContent(name, project);

        // Initialize output pins

        if (name === 'main' && opt.boardRules) {

          var initPins = opt.initPins || getInitPins(project);
          var n = initPins.length;

          if (n > 0) {
            // Declare m port
            ports.out.push({
              name: 'vinit',
```

```
                range: '[0:' + (n - 1) + ']'
              });
              // Generate port value
              var value = n.toString() + '\'b';
              for (i in initPins) {
                value += initPins[i].bit;
              }
              // Assign m port
              content += '\nassign vinit = ' + value + ';';
            }
          }

          data = {
            name: name,
            params: params,
            ports: ports,
            content: content
          };

          code += '//---- Top entity';
          code += module(data);
        }

        // Dependencies modules
        if(typeof project.package !== 'undefined'){

          code+='\n/*-------------------------------------------------*/\n';
          code+='/*-- '+project.package.name+'  */\n';
          code+='/*-- - - - - - - - - - - - - - - - - - - - - - - - --*/\n';
          code+='/*-- '+project.package.description+'\n';
          code+='/*-------------------------------------------------*/\n';


        }
        for (var d in dependencies) {
          code += verilogCompiler( utils.digestId(d) , dependencies[d]);
        }

        // Code modules

        for (i in blocks) {
          block = blocks[i];
          if (block) {
            if (block.type === 'basic.code') {
              data = {
                name: name + '_' +  utils.digestId(block.id) ,
                params: block.data.params,
                ports: block.data.ports,
                content: block.data.code //.replace(/\n+/g, '\n').replace(/\n$/g,
''')
              };
              code += module(data);
            }
```

```
        }
      }
    }

    return code;
  }

  function pcfCompiler(project, opt) {
    var i, j, block, pin, value, code = '';
    var blocks = project.design.graph.blocks;
    opt = opt || {};

    for (i in blocks) {
      block = blocks[i];
      if (block.type === 'basic.input' ||
        block.type === 'basic.output') {

        if (block.data.pins.length > 1) {
          for (var p in block.data.pins) {
            pin = block.data.pins[p];
            value = block.data.virtual ? '' : pin.value;
            code += 'set_io ';
            code +=  utils.digestId(block.id) ;
            code += '[' + pin.index + '] ';
            code += value;
            code += '\n';
          }
        } else if (block.data.pins.length > 0) {
          pin = block.data.pins[0];
          value = block.data.virtual ? '' : pin.value;
          code += 'set_io ';
          code +=  utils.digestId(block.id) ;
          code += ' ';
          code += value;
          code += '\n';
        }
      }
    }

    if (opt.boardRules) {
      // Declare init input ports

      var used = [];
      var initPorts = opt.initPorts || getInitPorts(project);
      for (i in initPorts) {
        var initPort = initPorts[i];
        if (used.indexOf(initPort.pin) !== -1) {
          break;
        }
        used.push(initPort.pin);

        // Find existing input block with the initPort value
        var found = false;
```

```
        for (j in blocks) {
          block = blocks[j];
          if (block.type === 'basic.input' &&
            !block.data.range &&
            !block.data.virtual &&
            initPort.pin === block.data.pins[0].value) {
            found = true;
            used.push(initPort.pin);
            break;
          }
        }

        if (!found) {
          code += 'set_io v';
          code += initPorts[i].name;
          code += ' ';
          code += initPorts[i].pin;
          code += '\n';
        }
      }

      // Declare init output pins

      var initPins = opt.initPins || getInitPins(project);
      if (initPins.length > 1) {
        for (i in initPins) {
          code += 'set_io vinit[' + i + '] ';
          code += initPins[i].pin;
          code += '\n';
        }
      } else if (initPins.length > 0) {
        code += 'set_io vinit ';
        code += initPins[0].pin;
        code += '\n';
      }
    }

    return code;
  }

  function lpfCompiler(project, opt) {
    var i, block, pin, value, code = '';
    var blocks = project.design.graph.blocks;
    opt = opt || {};

    code += '# -- Board: ';
    code += common.selectedBoard.name;
    code += '\n\n';

    for (i in blocks) {
      block = blocks[i];
      if (block.type === 'basic.input' ||
        block.type === 'basic.output') {
```

```
            //-- Future improvement: Both cases: 1-pin or multiple pins in an
array
            //-- could be refactorized instead of repeating code
            //-- (i usually name this as plural and singular cases)

          if (block.data.pins.length > 1) {
            for (var p in block.data.pins) {
              pin = block.data.pins[p];
              value = block.data.virtual ? '' : pin.value;
              code += 'LOCATE COMP "';
              code +=  utils.digestId(block.id) ;  //-- Future improvement: use
pin.name. It should also be changed in the main module
              code += '[' + pin.index + ']" SITE "';
              code += value;
              code += '";\n';

              code += 'IOBUF  PORT "';
              code +=  utils.digestId(block.id) ;
              code += '[' + pin.index + ']" ';

              //-- Get the pullmode property of the physical pin (its id is
pin.value)
              let pullmode = common.selectedBoard.pinout.find(x => x.value ===
value).pullmode;
              pullmode = (typeof pullmode === 'undefined') ? 'NONE' : pullmode;

              code += 'PULLMODE=' + pullmode;
              code += ' IO_TYPE=LVCMOS33 DRIVE=4;\n\n';
            }
          } else if (block.data.pins.length > 0) {
            pin = block.data.pins[0];
            value = block.data.virtual ? '' : pin.value;
            code += 'LOCATE COMP "';
            code +=  utils.digestId(block.id) ;  //-- Future improvement: use
pin.name. It should also be changed in the main module
            code += '" SITE "';
            code += value;
            code += '";\n';

            code += 'IOBUF  PORT "';
            code +=  utils.digestId(block.id) ;
            code += '" ';

            //-- Get the pullmode property of the physical pin (its id is
pin.value)
            let pullmode = common.selectedBoard.pinout.find(x => x.value ===
value).pullmode;
            pullmode = (typeof pullmode === 'undefined') ? 'NONE' : pullmode;

            code += 'PULLMODE=' + pullmode;
            code += ' IO_TYPE=LVCMOS33 DRIVE=4;\n\n';
          }
```

```
      }
    }

    return code;
  }

  function listCompiler(project) {
    var i;
    var listFiles = [];

    if (project &&
      project.design &&
      project.design.graph) {

      var blocks = project.design.graph.blocks;
      var dependencies = project.dependencies;

      // Find in blocks

      for (i in blocks) {
        var block = blocks[i];
        if (block.type === 'basic.memory') {
          listFiles.push({
            name: utils.digestId(block.id)  + '.list',
            content: block.data.list
          });
        }
      }

      // Find in dependencies

      for (i in dependencies) {
        var dependency = dependencies[i];
        listFiles = listFiles.concat(listCompiler(dependency));
      }
    }

    return listFiles;
  }

  function testbenchCompiler(project) {
    var i, o, p;
    var code = '';

    code += '// Testbench template\n\n';

    code += '`default_nettype none\n';
    code += '`define DUMPSTR(x) `"x.vcd`"\n';
    code += '`timescale 10 ns / 1 ns\n\n';

    var ports = {
      in: [],
      out: []
```

```
      };
      var content = '\n';

      content += '// Simulation time: 100ns (10 * 10ns)\n';
      content += 'parameter DURATION = 10;\n';

      // Parameters
      var _params = [];
      var params = mainParams(project);
      if (params.length > 0) {
        content += '\n// TODO: edit the module parameters here\n';
        content += '// e.g. localparam constant_value = 1;\n';
        for (p in params) {
          content += 'localparam ' + params[p].name + ' = ' + params[p].value +
';\n';
          _params.push(' .' + params[p].id + '(' + params[p].name + ')');
        }
      }

      // Input/Output
      var io = mainIO(project);
      var input = io.input;
      var output = io.output;
      content += '\n// Input/Output\n';
      var _ports = [];
      for (i in input) {
        content += 'reg ' + (input[i].range ? input[i].range + ' ' : '') +
input[i].name + ';\n';
        _ports.push(' .' + input[i].id + '(' + input[i].name + ')');
      }
      for (o in output) {
        content += 'wire ' + (output[o].range ? output[o].range + ' ' : '') +
output[o].name + ';\n';
        _ports.push(' .' + output[o].id + '(' + output[o].name + ')');
      }

      // Module instance
      content += '\n// Module instance\n';
      content += 'main';

      //-- Parameters
      if (_params.length > 0) {
        content += ' #(\n';
        content += _params.join(',\n');
        content += '\n)';
      }

      content += ' MAIN';

      //-- Ports
      if (_ports.length > 0) {
        content += ' (\n';
        content += _ports.join(',\n');
```

```
        content += '\n)';
      }

      content += ';\n';

      // Clock signal
      var hasClk = false;
      for (i in input) {
        if (input[i].name.toLowerCase() === 'clk') {
          hasClk = true;
          break;
        }
      }
      if (hasClk) {
        content += '\n// Clock signal\n';
        content += 'always #0.5 clk = ~clk;\n';
      }

      content += '\ninitial begin\n';
      content += ' // File were to store the simulation results\n';
      content += ' $dumpfile(`DUMPSTR(`VCD_OUTPUT));\n';
      content += ' $dumpvars(0, main_tb);\n\n';
      content += ' // TODO: initialize the registers here\n';
      content += ' // e.g. value = 1;\n';
      content += ' // e.g. #2 value = 0;\n';
      for (i in input) {
        content += ' ' + input[i].name + ' = 0;\n';
      }
      content += '\n';
      content += ' #(DURATION) $display("End of simulation");\n';
      content += ' $finish;\n';
      content += 'end\n';

      var data = {
        name: 'main_tb',
        ports: ports,
        content: content
      };
      code += module(data);

      return code;
    }

    function gtkwaveCompiler(project) {
      var code = '';

      var io = mainIO(project);
      var input = io.input;
      var output = io.output;

      for (var i in input) {
        code += 'main_tb.' + input[i].name + (input[i].range ? input[i].range :
'') + '\n';
```

```
      }
      for (var o in output) {
        code += 'main_tb.' + output[o].name + (output[o].range ? output[o].range
: '') + '\n';
      }

      return code;
    }

    function mainIO(project) {
      var input = [];
      var output = [];
      var inputUnnamed = 0;
      var outputUnnamed = 0;
      var graph = project.design.graph;
      for (var i in graph.blocks) {
        var block = graph.blocks[i];
        if (block.type === 'basic.input') {
          if (block.data.name) {
            input.push({
              id: utils.digestId(block.id) ,
              name: block.data.name.replace(/ /g, '_'),
              range: block.data.range
            });
          } else {
            input.push({
              id:  utils.digestId(block.id) ,
              name: inputUnnamed.toString(),
            });
            inputUnnamed += 1;
          }
        } else if (block.type === 'basic.output') {
          if (block.data.name) {
            output.push({
              id:  utils.digestId(block.id) ,
              name: block.data.name.replace(/ /g, '_'),
              range: block.data.range
            });
          } else {
            output.push({
              id:  utils.digestId(block.id) ,
              name: outputUnnamed.toString()
            });
            outputUnnamed += 1;
          }
        }
      }

      return {
        input: input,
        output: output
      };
    }
```

```
  function mainParams(project) {
    var params = [];
    var paramsUnnamed = 0;
    var graph = project.design.graph;
    for (var i in graph.blocks) {
      var block = graph.blocks[i];
      if (block.type === 'basic.constant') {
        if (!block.data.local) {
          if (block.data.name) {
            params.push({
              id: utils.digestId(block.id),
              name: 'constant_' + block.data.name.replace(/ /g, '_'),
              value: block.data.value
            });
          } else {
            params.push({
              id: utils.digestId(block.id),
              name: 'constant_' + paramsUnnamed.toString(),
              value: block.data.value
            });
            paramsUnnamed += 1;
          }
        }
      }
    }

    return params;
  }

});
```

ANEXO II – CÓDIGO DE LA SECCIÓN TOOLS

```
this.toolchain = toolchain, nodeFse.removeSync(common.OLD_BUILD_DIR),
this.verifyCode = function(startMessage, endMessage) {
            return apioRun(["verify"], startMessage, endMessage)
      }, this.buildCode = function(startMessage, endMessage) {
            return apioRun(["build", "--board", common.selectedBoard.name],
startMessage, endMessage)
      }, this.uploadCode = function(startMessage, endMessage) {
            return apioRun(["upload", "--board", common.selectedBoard.name],
startMessage, endMessage)
      }, this.checkToolchain = checkToolchain,
$rootScope.$on("installToolchain", function() {
            this.installToolchain()
      }.bind(this)), this.installToolchain = function() {
            resultAlert && resultAlert.dismiss(!1),
utils.checkDefaultToolchain() ? (utils.removeToolchain(),
installDefaultToolchain()) : alertify.confirm(gettextCatalog.getString("Default
toolchain not found. Toolchain will be downloaded. This operation requires
Internet connection. Do you want to continue?"), function() {
                  utils.removeToolchain(), installOnlineToolchain()
            })
      }, this.updateToolchain = function() {
            resultAlert && resultAlert.dismiss(!1),
alertify.confirm(gettextCatalog.getString("The toolchain will be updated. This
operation requires Internet connection. Do you want to continue?"), function() {
                  installOnlineToolchain()
            })
      }, this.resetToolchain = function() {
            resultAlert && resultAlert.dismiss(!1),
utils.checkDefaultToolchain() ? alertify.confirm(gettextCatalog.getString("The
toolchain will be restored to default. Do you want to continue?"), function() {
                  utils.removeToolchain(), installDefaultToolchain()
            }) : alertify.alert(gettextCatalog.getString("Error: default
toolchain not found in '{{dir}}'", {
                  dir: common.TOOLCHAIN_DIR
            }))
      }, this.removeToolchain = function() {
            resultAlert && resultAlert.dismiss(!1),
alertify.confirm(gettextCatalog.getString("The toolchain will be removed. Do you
want to continue?"), function() {
                  utils.removeToolchain(), toolchain.apio = "",
toolchain.installed = !1, alertify.success(gettextCatalog.getString("Toolchain
removed"))
            })
      }, $rootScope.$on("enableDrivers", function() {
            this.enableDrivers()
      }.bind(this)), this.enableDrivers = function() {
            checkToolchain(function() {
```

```
                        toolchain.installed && drivers.enable()
                })
        }, this.disableDrivers = function() {
                checkToolchain(function() {
                        toolchain.installed && drivers.disable()
                })
        }, this.addCollections = function(filepaths) {
                async.eachSeries(filepaths, function(filepath, nextzip) {
                        var zipData = nodeAdmZip(filepath),
                                _collections = getCollections(zipData);
                        async.eachSeries(_collections, function(collection, next) {
                                setTimeout(function() {
                                        collection.package && (collection.blocks ||
collection.examples) ? alertify.prompt(gettextCatalog.getString("Edit the
collection name"), collection.origName, function(evt, name) {
                                                if (!name) return !1;
                                                collection.name = name;
                                                var destPath =
nodePath.join(common.INTERNAL_COLLECTIONS_DIR, name);
                                                nodeFs.existsSync(destPath) ?
alertify.confirm(gettextCatalog.getString("The collection {{name}} already
exists.", {
                                                        name: utils.bold(name)
                                                }) + "<br>" + gettextCatalog.getString("Do
you want to replace it?"), function() {

        utils.deleteFolderRecursive(destPath), installCollection(collection,
zipData), alertify.success(gettextCatalog.getString("Collection {{name}}
replaced", {
                                                                name: utils.bold(name)
                                                        })), next(name)
                                                }, function() {

        alertify.warning(gettextCatalog.getString("Collection {{name}} not
replaced", {
                                                                name: utils.bold(name)
                                                        })), next(name)
                                                }) : (installCollection(collection,
zipData), alertify.success(gettextCatalog.getString("Collection {{name}} added",
{
                                                        name: utils.bold(name)
                                                })), next(name))
                                        }) :
alertify.warning(gettextCatalog.getString("Invalid collection {{name}}", {
                                                name: utils.bold(name)
                                        }))
                                }, 0)
                        }, function(name) {
                                collections.loadInternalCollections(),
common.selectedCollection.name === name && collections.selectCollection(name),
utils.rootScopeSafeApply(), nextzip()
                        })
                })
```

```
      }, this.removeCollection = function(collection) {
            utils.deleteFolderRecursive(collection.path),
collections.loadInternalCollections(),
alertify.success(gettextCatalog.getString("Collection {{name}} removed", {
                  name: utils.bold(collection.name)
            }))
      }, this.removeAllCollections = function() {
            utils.removeCollections(), collections.loadInternalCollections(),
alertify.success(gettextCatalog.getString("All collections removed"))
      }, this.checkForNewVersion = function() {
            void 0 !== _package.updatecheck && $.getJSON(_package.updatecheck +
"?_tsi=" + (new Date).getTime(), function(result) {
                  var hasNewVersion = !1;
                  if (!1 !== result && (void 0 !== result.version &&
_package.version < result.version && (hasNewVersion = "stable"), void 0 !==
result.nightly && _package.version < result.nightly && (hasNewVersion =
"nightly"), !1 !== hasNewVersion)) {
                        var msg = "";
                        msg = "stable" === hasNewVersion ? '<div class="new-
version-notifier-box"><div class="new-version-notifier-box--icon"><img
src="resources/images/confetti.svg"></div>
<div class="new-version-notifier-box--text">' + gettextCatalog.getString("There
is a new stable version available") + '<br/><a class="action-open-url-external-
browser" href="https://icestudio.io" target="_blank">Click here to download
it.</a></div></div>' : '<div class="new-version-notifier-box"><div class="new-
version-notifier-box--icon"><img src="resources/images/confetti.svg"></div>
<div class="new-version-notifier-box--text">' + gettextCatalog.getString("There
is a new nightly version available") + '<br/><a class="action-open-url-external-
browser" href="https://icestudio.io" target="_blank">Click here to download
it.</a></div></div>', alertify.notify(msg, "notify", 30)
                  }
            })
      }, this.ifDevelopmentMode = function() {
            void 0 !== _package.development && void 0 !==
_package.development.mode && !0 === _package.development.mode &&
utils.openDevToolsUI()
      }
```

ANEXO III – CÓDIGO DE LOS BLOQUES EN VHDL

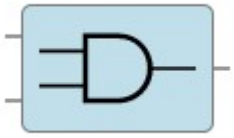

Figura 19 - Bloque AND

```
// AND logic gate

entity AND_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end AND_VHDL;
architecture behavioral of AND_VHDL is
begin
	c <= a AND b;
end behavioral;
```

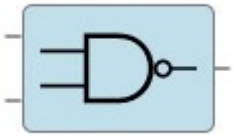

Figura 20 - Bloque NAND

```
// NAND logic gate

entity NAND_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end NAND_VHDL;
architecture behavioral of NAND_VHDL is
begin
	c <= a NAND b;
end behavioral;
```

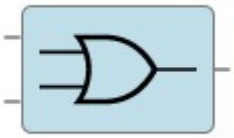

Figura 21 - Bloque OR

```
// OR logic gate

entity OR_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end OR_VHDL;
architecture behavioral of OR_VHDL is
begin
	c <= a OR b;
end behavioral;
```

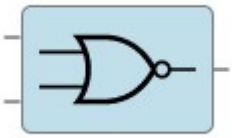

Figura 22 - Bloque NOR

```
// NOR logic gate

entity NOR_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end NOR_VHDL;
architecture behavioral of NOR_VHDL is
begin
	c <= a NOR b;
end behavioral;
```

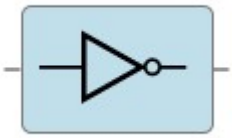

Figura 23 - Bloque NOT

```
// NOT logic gate

entity NOT_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end NOT_VHDL;
architecture behavioral of NOT_VHDL is
begin
	c <= a NOT b;
end behavioral;
```

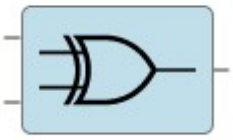

Figura 24 - Bloque XOR

```
// XOR logic gate

entity XOR_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end XOR_VHDL;
architecture behavioral of XOR_VHDL is
begin
	c <= a XOR b;
end behavioral;
```

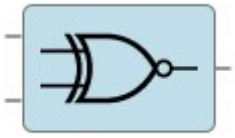

Figura 25 - Bloque XNOR

```
// XNOR logic gate

entity XNOR_VHDL is
	port(
		a : in std_logic;
		b : in std_logic;
		c : out std_logic
	);
end XNOR_VHDL;
architecture behavioral of XNOR_VHDL is
begin
	c <= a XNOR b;
end behavioral;
```

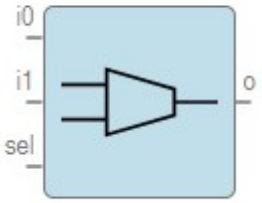

Figura 26 - Bloque Multiplexor

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

 ENTITY mux IS
    PORT(in0      : IN std_logic;
         in1      : IN std_logic;
         sel0     : IN std_logic;
         o        : OUT std_logic);
 END mux;

 ARCHITECTURE behavioral OF mux IS
 BEGIN

   PROCESS (sel0, in0, in1, o) IS
   BEGIN
```

```
    CASE sel0 IS
      WHEN "0" => o <= in0;
      WHEN "1" => o <= in1;
      WHEN OTHERS => o <= in0;
    END CASE;
  END PROCESS;
END behavioral;
```

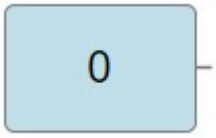

Figura 27 - Bloque BIT 0

```
// Bit 0

entity BIT_0 is
      port(
            v : out std_logic
      );
end BIT_0;
architecture behavioral of BIT_0 is
begin
      v <= '0';
end behavioral;
```

Figura 28 - Bloque BIT 1

```
// Bit 1

entity BIT_1 is
      port(
            v : out std_logic
      );
end BIT_1;
architecture behavioral of BIT_1 is
begin
      v <= '1';
end behavioral;
```

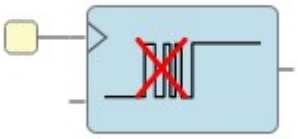

Figura 29 - Bloque Debouncer

```
/-- Debouncer Circuit
//-- It produces a stable output when the
//-- input signal is bouncing

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity debouncer_top is
    port (
    clk           : in std_logic;
      in     : in std_logic;
      out           : out std_logic);
end debouncer_top;

architecture behavioral of debouncer is
    constant counter_size   : integer := 16;
    signal btn_prev         : std_logic := '0';
    signal btn_out_r        : std_logic := '0';
    signal counter          : std_logic_vector(counter_size downto 0) := (others
=> '0');

begin
    process(clk)
    begin
      if (clk'event and clk='1') then
          //-- If btn_prev and btn_in are differents
            if (btn_prev xor in) = '1' then
                  //-- Reset the counter

                  counter <= (others => '0');
                  //-- Capture the button status

                  btn_prev <= in;
            elsif (counter(counter_size) = '0') then
              //-- If no timeout, increase the counter

                  counter <= counter + 1;
            else
            //-- Set the output to the stable value

                  btn_out_r <= btn_prev;
```

```
            end if;
      end if;
    end process;
    out <= btn_out_r;
end behavioral;
```

Figura 30 - Bloque Flip-Flop tipo D

```
entity flipflopD is
      port(
            clk : in std_logic;
            d   : in std_logic;
            q   : out std_logic
      );
end flipflopD;
architecture behavioral of flipflopD is
begin
      process(clk)--tanto un cambio en clk como reset disparan el proceso
      begin
            if (clk'event and clk='1') then-- si no existe un reset y el cambio
de clk=1
                  q<=d;-- funcionamiento normal del ffd
            end if;
      end process;
end behavioral;
```

Figura 31 Bloque Flip-Flop tipo T

En el caso del flip-flop tipo T, está compuesto por un flip-flop tipo D y una puerta lógica NOT, como se mostrará en la Figura 32. Como todos estos bloques ya están explicados y definidos no se procederá a reescribir el código.

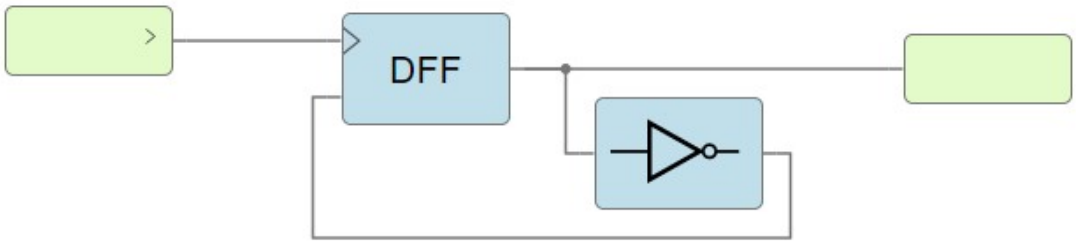

Figura 32 - Interior de Flip-Flop tipo T

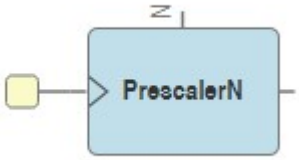

Figura 33 - Bloque de Preescalado con N genérico

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Preescaler is
    generic (N: integer);
    Port (
        clk_in  : in  STD_LOGIC;
        clk_out : out STD_LOGIC
    );
end Preescaler;

architecture behavioral of Preescaler is
    signal divcounter: integer range 0 to N-1 := 0;
begin
    divisor_frecuencia: process (clk_in) begin
        if rising_edge(clk_in) then
            divcounter <= divcounter+1;
        end if;
    end process;

    clk_out <= divcounter[N-1];
end behavioral;
```

En el caso del bloque con preescalado 22, se usará N=22 pero el código se mantendrá igual.

ANEXO IV – CÓDIGO DEL MENÚ

```
<!DOCTYPE html>
<div ng-controller="MenuCtrl">

  <div id="menu" ondragstart="return false;" ondrop="return false;">

    <input id="input-open-project" type="file" accept=".ice" class="hidden"/>
    <input id="input-save-project" type="file" accept=".ice" class="hidden"
nwsaveas="{{ workingdir + project.name }}.ice"/>
    <input id="input-save-snapshot" type="file" accept=".png" class="hidden"
nwsaveas="{{ (snapshotdir ? snapshotdir : workingdir) + project.name }}.png"/>

    <input id="input-add-as-block" type="file" accept=".ice,.iceb" class="hidden"
multiple/>

    <input id="input-export-verilog" type="file" accept=".v" class="hidden"
nwsaveas="{{ workingdir + project.name }}.v"/>
    <input id="input-export-pcf" type="file" accept=".pcf" class="hidden"
nwsaveas="{{ workingdir + project.name }}.pcf"/>
    <input id="input-export-testbench" type="file" accept=".v" class="hidden"
nwsaveas="{{ workingdir + project.name }}_tb.v"/>
    <input id="input-export-gtkwave" type="file" accept=".gtkw" class="hidden"
nwsaveas="{{ workingdir + project.name }}_tb.gtkw"/>
    <input id="input-export-blif" type="file" accept=".blif" class="hidden"
nwsaveas="{{ workingdir + project.name }}.blif"/>
    <input id="input-export-asc" type="file" accept=".asc" class="hidden"
nwsaveas="{{ workingdir + project.name }}.asc"/>
    <input id="input-export-bin" type="file" accept=".bin" class="hidden"
nwsaveas="{{ workingdir + project.name }}.bin"/>

    <input id="input-add-collection" type="file" accept=".zip" class="hidden"
multiple/>

    <nav class="navbar navbar-default navbar-static-top ice-bar"
role="navigation">

      <div class="collapse navbar-collapse">

        <ul class="nav navbar-nav">

          <li uib-dropdown ng-mouseover="showMenu('file')" ng-
mouseleave="hideMenu('file')" ng-click="fixMenu('file')" is-open="status.file">
            <a href uib-dropdown-toggle>{{ 'File' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu style="min-width: 230px">
              <li>
                <a href ng-click="newProject()">{{ 'New' | translate }}<span
class="shortcut">{{ 'newProject' | shortcut }}</span></a>
```

```
              </li>
              <li>
                <a href ng-click="openProjectDialog()">{{ 'Open' | translate
}}...<span class="shortcut">{{ 'openProject' | shortcut }}</span></a>
              </li>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Blocks' | translate }}</a>
                <menutree data="common.selectedCollection.content.blocks"
callback="openProject(path)"></menutree>
              </li>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Examples' | translate }}</a>
                <menutree data="common.selectedCollection.content.examples"
callback="openProject(path)"></menutree>
              </li>
              <li>
                <a href ng-click="addAsBlock()">{{ 'Add as block' | translate
}}...</a>
              </li>
              <li class="divider"></li>

              <li>
                <a href ng-click="saveProject()">{{ 'Save' | translate }}<span
class="shortcut">{{ 'saveProject' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="saveProjectAs()">{{ 'Save as' | translate
}}...<span class="shortcut">{{ 'saveProjectAs' | shortcut }}</span></a>
              </li>
              <li class="divider"></li>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Export' | translate }}...</a>
                <ul uib-dropdown-menu>
                  <li>
                    <a href ng-click="exportVerilog()">{{ 'Verilog' }}</a>
                  </li>
                  <li>
                    <a href ng-click="exportPCF()">{{ 'PCF' }}</a>
                  </li>
                  <li>
                    <a href ng-click="exportTestbench()">{{ 'Testbench' |
translate }}</a>
                  </li>
                  <li>
                    <a href ng-click="exportGTKwave()">{{ 'GTKWave' }}</a>
                  </li>
                  <li class="divider"></li>
                  <li>
                    <a href ng-click="exportBLIF()">{{ 'BLIF' }}</a>
                  </li>
                  <li>
                    <a href ng-click="exportASC()">{{ 'ASC' }}</a>
                  </li>
```

```
                  <li>
                    <a href ng-click="exportBitstream()">{{ 'Bitstream' }}</a>
                  </li>
                </ul>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="quit()">{{ 'Quit' | translate }}<span
class="shortcut">{{ 'quit' | shortcut }}</span></a>
              </li>
            </ul>
          </li>

          <li uib-dropdown ng-mouseover="showMenu('edit')" ng-
mouseleave="hideMenu()" ng-click="fixMenu('edit')" is-open="status.edit">
            <a href uib-dropdown-toggle>{{ 'Edit' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu style="min-width: 200px">
              <li>
                <a href ng-click="undoGraph()">{{ 'Undo' | translate }}<span
class="shortcut">{{ 'undoGraph' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="redoGraph()">{{ 'Redo' | translate }}<span
class="shortcut">{{ 'redoGraph' | shortcut }}</span></a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="cutSelected()">{{ 'Cut' | translate }}<span
class="shortcut">{{ 'cutSelected' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="copySelected()">{{ 'Copy' | translate }}<span
class="shortcut">{{ 'copySelected' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="pasteSelected()">{{ 'Paste' | translate }}<span
class="shortcut">{{ 'pasteSelected' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="pasteAndCloneSelected()">{{ 'Clone' | translate
}}<span class="shortcut">{{ 'pasteAndCloneSelected' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="selectAll()">{{ 'Select all' | translate
}}<span class="shortcut">{{ 'selectAll' | shortcut }}</span></a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="fitContent()">{{ 'Fit content' | translate
}}<span class="shortcut">{{ 'fitContent' | shortcut }}</span></a>
              </li>
              <li class="divider"></li>
```

```
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Preferences' | translate }}</a>
                <ul uib-dropdown-menu>
                  <li class="dropdown-submenu" uib-dropdown>
                    <a href uib-dropdown-toggle>{{ 'Language' | translate }}</a>
                    <ul uib-dropdown-menu ng-
include="'views/languages.html'"></ul>
                  </li>
                  <li class="divider"></li>
                  <li>
                    <a href ng-click="toggleBoardRules()">
                      {{ 'Board rules' | translate }} 
                      <span ng-show="profile.data.boardRules" class="glyphicon
glyphicon-ok-circle"></span>
                    </a>
                  </li>
                  <li>
                    <a href ng-click="setExternalCollections()">{{ 'External
collections' | translate }}</a>
                  </li>
                  <li>
                    <a href ng-click="setRemoteHostname()">
                      {{ 'Remote hostname' | translate }}
                      <span ng-show="profile.data.remoteHostname"
class="glyphicon glyphicon-ok-circle"></span>
                    </a>
                  </li>
                </ul>
              </li>
              <li>
                <a href ng-click="setProjectInformation()">{{ 'Project
information' | translate }}</a>
              </li>
            </ul>
          </li>

          <li uib-dropdown ng-mouseover="showMenu('view')" ng-
mouseleave="hideMenu()" ng-click="fixMenu('view')" is-open="status.view">
            <a href uib-dropdown-toggle>{{ 'View' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu>
              <li>
                <a href ng-click="showPCF()">{{ 'PCF' }}</a>
              </li>
              <li>
                <a href ng-click="showPinout()">{{ 'Pinout' }}</a>
              </li>
              <li>
                <a href ng-click="showDatasheet()">{{ 'Datasheet' | translate
}}</a>
              </li>
              <li>
```

```
                <a href ng-click="showBoardRules()">{{ 'Board rules' | translate
}}</a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="showCollectionData()">{{ 'Collection info' |
translate }}</a>
              </li>
              <li>
                <a href ng-click="showCommandOutput()">{{ 'Command output' |
translate }}</a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="toggleFPGAResources()">
                  {{ 'FPGA resources' | translate }} 
                  <span ng-show="profile.data.showFPGAResources" class="glyphicon
glyphicon-ok-circle"></span>
                </a>
              </li>
            </ul>
          </li>

          <li uib-dropdown ng-mouseover="showMenu('select')" ng-
mouseleave="hideMenu()" ng-click="fixMenu('select')" is-open="status.select">
            <a href uib-dropdown-toggle>{{ 'Select' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Board' | translate }}</a>
                <ul uib-dropdown-menu>
                   <li><a><b>HX1K</b></a></li>
                   <li ng-repeat="board in common.boards">
                     <menuboard type="'HX1K'" board="board"></menuboard>
                   </li>
                   <li><a><b>HX8K</b></a></li>
                   <li ng-repeat="board in common.boards">
                     <menuboard type="'HX8K'" board="board"></menuboard>
                   </li>
                   <li><a><b>LP8K</b></a></li>
                   <li ng-repeat="board in common.boards">
                     <menuboard type="'LP8K'" board="board"></menuboard>
                   </li>
                   <li><a><b>UP5K</b></a></li>
                   <li ng-repeat="board in common.boards">
                     <menuboard type="'UP5K'" board="board"></menuboard>
                   </li>
                 </ul>
              </li>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Collection' | translate }}</a>
                <ul uib-dropdown-menu>
                  <data ng-init="collection = common.defaultCollection"/>
```

```
                  <li>
                    <a href ng-click="selectCollection(collection)" ng-
show="collection.name == ''">
                      {{ 'Default' | translate }} 
                      <span ng-show="common.selectedCollection.name == ''"
class="glyphicon glyphicon-ok-circle"></span>
                    </a>
                  </li>
                  <li class="divider" ng-show="common.internalCollections.length
> 0"></li>
                  <li ng-repeat="collection in common.internalCollections">
                    <a href ng-click="selectCollection(collection)" ng-
hide="collection.name == ''">
                      {{ collection.name }} 
                      <span ng-show="common.selectedCollection.path ==
collection.path" class="glyphicon glyphicon-ok-circle"></span>
                    </a>
                  </li>
                  <li class="divider" ng-show="common.externalCollections.length
> 0"></li>
                  <li ng-repeat="collection in common.externalCollections">
                    <a href ng-click="selectCollection(collection)" ng-
hide="collection.name == ''">
                      {{ collection.name }} 
                      <span ng-show="common.selectedCollection.path ==
collection.path" class="glyphicon glyphicon-ok-circle"></span>
                    </a>
                  </li>
                </ul>
              </li>
            </ul>
          </li>

          <li uib-dropdown ng-mouseover="showMenu('tools')" ng-
mouseleave="hideMenu()" ng-click="fixMenu('tools')" is-open="status.tools">
            <a href uib-dropdown-toggle>{{ 'Tools' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu style="min-width: 180px">
              <li>
                <a href ng-click="verifyCode()">{{ 'Verify' | translate }}<span
class="shortcut">{{ 'verifyCode' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="buildCode()">{{ 'Build' | translate }}<span
class="shortcut">{{ 'buildCode' | shortcut }}</span></a>
              </li>
              <li>
                <a href ng-click="uploadCode()">{{ 'Upload' | translate }}<span
class="shortcut">{{ 'uploadCode' | shortcut }}</span></a>
              </li>
              <li class="divider" ng-if="common.showToolchain()"></li>
              <li class="dropdown-submenu" uib-dropdown ng-
if="common.showToolchain()">
```

```
                <a href uib-dropdown-toggle>{{ 'Toolchain' | translate }}</a>
                <ul uib-dropdown-menu>
                  <li ng-class="toolchain.disabled ? 'disabled' : ''">
                    <a href ng-hide="toolchain.installed" ng-
click="tools.installToolchain()">
                      {{ 'Install' | translate }}
                    </a>
                    <a href ng-show="toolchain.installed" ng-
click="tools.updateToolchain()">
                      {{ 'Update' | translate }}
                    </a>
                  </li>
                  <li ng-class="(toolchain.disabled || !toolchain.installed) ?
'disabled' : ''">
                    <a href ng-click="(toolchain.disabled ||
!toolchain.installed) ? '' : tools.removeToolchain()">
                      {{ 'Remove' | translate }}
                    </a>
                  </li>
                  <li ng-class="(toolchain.disabled || !toolchain.installed) ?
'disabled' : ''">
                    <a href ng-click="(toolchain.disabled ||
!toolchain.installed) ? '' : tools.resetToolchain()">
                      {{ 'Reset default' | translate }}
                    </a>
                  </li>
                  <li class="divider" ng-show="toolchain.apio"></li>
                  <li class="disabled" ng-show="toolchain.apio">
                    <a href>{{ 'Apio ' + toolchain.apio }}</a>
                  </li>
                </ul>
              </li>
              <li class="dropdown-submenu" uib-dropdown ng-
if="common.showDrivers()">
                <a href uib-dropdown-toggle>{{ 'Drivers' | translate }}</a>
                <ul uib-dropdown-menu>
                  <li>
                    <a href ng-click="tools.enableDrivers()">
                      {{ 'Enable' | translate }}
                    </a>
                  </li>
                  <li>
                    <a href ng-click="tools.disableDrivers()">
                      {{ 'Disable' | translate }}
                    </a>
                  </li>
                </ul>
              </li>
              <li class="divider"></li>
              <li class="dropdown-submenu" uib-dropdown>
                <a href uib-dropdown-toggle>{{ 'Collections' | translate }}</a>
                <ul uib-dropdown-menu>
                  <li>
```

```
                    <a href ng-click="addCollections()">
                      {{ 'Add' | translate }}
                    </a>
                  </li>
                  <li>
                    <a href ng-click="reloadCollections()">
                      {{ 'Reload' | translate }}
                    </a>
                  </li>
                  <li class="dropdown-submenu" uib-dropdown>
                    <a href uib-dropdown-toggle>{{ 'Remove' | translate }}</a>
                    <ul uib-dropdown-menu
                        ng-show="common.internalCollections.length > 0">
                      <li ng-repeat="collection in common.internalCollections">
                        <a href ng-click="removeCollection(collection)" ng-
if="collection.name != ''">
                          {{ collection.name | translate }}
                        </a>
                      </li>
                    </ul>
                  </li>
                  <li>
                    <a href ng-click="removeAllCollections()">
                      {{ 'Remove all' | translate }}
                    </a>
                  </li>
                </ul>
              </li>
               <li class="divider"></li>
              <li>
                <a href ng-click="showChromeDevTools()">{{ 'Chrome Dev Tools' |
translate }}</a>
              </li>

            </ul>
          </li>

          <li uib-dropdown ng-mouseover="showMenu('help')" ng-
mouseleave="hideMenu()" ng-click="fixMenu('help')" is-open="status.help">
            <a href uib-dropdown-toggle>{{ 'Help' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu>
              <li disabled>
                <a href ng-click="openUrl('https://www.gnu.org/licenses/old-
licenses/gpl-2.0.html')">{{ 'View license' | translate }}</a>
              </li>
              <li>
                <a href ng-click="openVersionInfoWindow()">{{ 'Version notes' |
translate }}</a>
              </li>
              <li class="disabled">
                <a href>{{ 'Version' | translate }} {{ version }}</a>
              </li>
```

```
              <li class="divider"></li>
              <li>
                <a href ng-click="openUrl('http://icestudio.readthedocs.io')">{{
'Documentation' | translate }}</a>
              </li>
              <li>
                <a href ng-
click="openUrl('https://github.com/FPGAwars/icestudio')">{{ 'Source code' |
translate }}</a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-
click="openUrl('https://groups.google.com/forum/#!forum/fpga-wars-explorando-el-
lado-libre')">{{ 'Community forum' | translate }}</a>
              </li>
              <li class="divider"></li>
              <li>
                <a href ng-click="about()">{{ 'About Icestudio' | translate
}}</a>
              </li>
            </ul>
          </li>

        </ul>

        <ul class="nav navbar-nav navbar-right">

          <li uib-dropdown
              ng-mouseover="showMenu('basic')" ng-mouseleave="hideMenu()" ng-
click="fixMenu('basic')" is-open="status.basic">
            <a href uib-dropdown-toggle>{{ 'Basic' | translate }}<span
class="caret"></span></a>
            <ul uib-dropdown-menu>
              <li>
                <a href ng-click="project.addBasicBlock('basic.input')">
                  <div class="marker marker-yellow"></div>{{ 'Input' | translate
}}
                </a>
                <a href ng-click="project.addBasicBlock('basic.output')">
                  <div class="marker marker-yellow"></div>{{ 'Output' | translate
}}
                </a>
                <a href ng-click="project.addBasicBlock('basic.outputLabel')">
                  <div class="marker marker-yellow"></div>{{ 'Output label' |
translate }}
                </a>
                <a href ng-click="project.addBasicBlock('basic.inputLabel')">
                  <div class="marker marker-yellow"></div>{{ 'Input label' |
translate }}
                </a>
                <a href ng-click="project.addBasicBlock('basic.constant')">
```

```
                  <div class="marker marker-orange"></div>{{ 'Constant' |
translate }}
                </a>
                <a href ng-click="project.addBasicBlock('basic.memory')">
                  <div class="marker marker-orange"></div>{{ 'Memory' | translate
}}
                </a>
                <a href ng-click="project.addBasicBlock('basic.code')">
                  <div class="marker marker-blue"></div>{{ 'Code' | translate }}
                </a>
                <a href ng-click="project.addBasicBlock('basic.info')">
                  <div class="marker marker-gray"></div>{{ 'Information' |
translate }}
                </a>
              </li>
            </ul>
          </li>

          <li uib-dropdown
              ng-repeat="block in common.selectedCollection.content.blocks" ng-
if="block.children"
              ng-mouseover="showMenu(block.name)" ng-mouseleave="hideMenu()" ng-
click="fixMenu(block.name)" is-open="status[block.name]">
            <a href uib-dropdown-toggle>{{ block.name | translate }} <span
class="caret"></span></a>
            <menutree data="block.children" callback="project.addBlockFile(path)"
right="true"></menutree>
          </li>

        </ul>

      </div>

    </nav>

  </div>

<div ng-include="'views/version.html'"></div>
</div>
```

ANEXO V – OBJETIVOS DE DESARROLLO SOSTENIBLE

Este proyecto se alineará principalmente con los siguientes objetivos de desarrollo sostenible.

- Educación de calidad (ODS 4): al hacer una aplicación de código libre que permite usar VHDL para programar FPGAs, permite a un mayor número de alumnos familiarizarse con el uso de estas de una manera mucho más sencilla que hasta ahora, con menor necesidad de conocimientos avanzados de diseño de hardware, permitiendo el aprendizaje secuencial e impulsando el número de proyectos con FPGAs, intentado igualar los avances que ya ha habido con otro tipo de controladores como Arduino.

- Industria, innovación e infraestructuras (ODS 9): haciendo esta tecnología más accesible para todos los profesionales y jóvenes estudiantes, se fomentará la innovación en la industria haciendo que se automaticen más sistemas con estos microcontroladores, fácilmente programables. Permitiendo así el desarrollo de la industria, para ser más eficientes permitiendo así un desarrollo más sostenible.