

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



PROYECTO FIN DE CARRERA

Aplicación Android para simular una FPGA educativa

Ángel Guerra Martín

Septiembre 2013

Aplicación Android para simular una FPGA educativa

AUTOR: Ángel Guerra Martín

TUTOR: Eduardo Boemo Scalvinoni

DSLab

Dpto. de Tecnología Electrónica y Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Septiembre de 2013

Agradecimientos

Me gustaría que este apartado no solo fuera dedicado a agradecer a todas las personas relacionadas conmigo y la universidad, sino a todas y cada una de las personas que me he ido encontrando en mi vida, me han hecho llegar hasta donde estoy, me han acompañado y me han hecho ser un poco mejor.

En primer lugar me gustaría dar las gracias a mi tutor, Eduardo Boemo, por darme la posibilidad de llevar a cabo este proyecto. También doy las gracias a cada uno de los profesores con los que he tenido el placer de dar clases.

A todos los compañeros y amigos de la universidad, que han tenido que sufrir tantas horas de estudios, de laboratorios, de agobios, muchas gracias por estar ahí, por apoyarme, por hacer que todo fuera mucho más ameno, por los muses y por las noches de fiesta merecidas que nos hemos dado todos juntos. Espero que por muchos años que pasen, siempre haya un rato para seguir viéndonos.

Henar se merece una mención especial, una compañera y una amiga, primero en el colegio y luego en la universidad, siempre juntos. Muchas gracias por estar ahí en todo momento, por ser como un pepito grillo cuando lo he necesitado y por todo lo que nos queda.

A mis dos mejores amigos, Pablo y Carlos, poco puedo decir que no sepan ya, muchas gracias por cada risa que nos echamos juntos, por apoyarme en cualquier circunstancia y por demostrarme que siempre que os necesite vais a estar ahí.

A mi novia y amiga, Cristina. Agradecerte todo lo que has hecho por mí, por escucharme, por acompañarme, por apoyarme, por hacerme reír, por ser como eres, por cada momento que hemos pasado y por toda la vida que nos queda por estar juntos.

Y por último, pero no menos importante, tengo que dar las gracias a mi familia, ya que son las personas que realmente han llevado el peso y la carga de aguantarme en casa todos los días, son los que me han animado a seguir cuando he tenido dudas, han confiado en mí en todo momento, me han apoyado, me han hecho ser como soy, me han regalado la oportunidad de hacer esta carrera y me han dado todo lo que tengo. Muchas gracias Mamá, Papá y Natalia.

Muchas gracias a todos.

Ángel Guerra Martín

Septiembre 2013

Resumen

En los últimos años se ha producido un uso masivo de dispositivos inteligentes. Este proyecto trata de aprovechar el actual auge de estos dispositivos ofreciendo a los estudiantes de ingeniería una plataforma multimedia ideal para realizar acciones educativas.

En este proyecto se ha desarrollado una aplicación Android. Esta aplicación se ejecuta sobre una tableta con sistema operativo Android utilizando una maqueta de una FPGA básica educativa. La arquitectura de la FPGA está formada por LUTs, FF, pistas, puntos programables de interconexión y E/S.

En la aplicación se da la posibilidad de elegir entre 11 tipos de ejercicios pudiendo mapearlos en la maqueta de la FPGA. Con la resolución de los diferentes ejercicios, los estudiantes pueden comprender conceptos de Arquitectura de FPGAs: multiplexores como tablas de look-up, puntos de interconexiones programables, congestión de pistas, memoria de configuración y temas de algoritmos EDA de mapeado.

Durante la resolución de los ejercicios, el alumno puede comprobar si la solución de su ejercicio cumple con los requisitos para ser correcta, observar el estado de las posiciones de memoria que forman la FPGA y guardar el ejercicio para posteriormente continuar con él o para enviarlo al correo del profesor. Una vez que el profesor ha recibido el ejercicio puede importarlo, cargándolo en la aplicación de su tableta para revisarlo y poder puntuarlo.

Palabras clave

Dispositivos inteligentes, aplicación Android, FPGA, multiplexores, puntos de interconexiones programables, congestión de pistas, memoria de configuración.

Abstract

In the last years, a massive use has been produced in smart devices. This project seeks to take advantage of current growth of these devices offering a great multimedia platform for educational activities to engineering students.

In this work an Android application has been developed. This application has been run on an Android tablet using a basic educational FPGA model. FPGA architecture includes LUTs, FF, tracks, programmable interconnect points and I / O.

The application gives the possibility to choose between 11 different exercises which can be mapped on the FPGA model. By solving the different exercises, students can understand concepts of FPGAs architecture: multiplexers and look-up tables, programmable interconnect points, tracks congestion, configuration memory and EDA issues mapping algorithms.

During the exercises resolution, students can check if the solution meets the requirements of the correct form of solving them, see the state of the memory locations that form the FPGA and save the exercise and then continue with it or send it to the teacher's e-mail. Once the exercise has been received by the teacher, he can import it, charging in the application of its tablet to review and rate it later.

Key words

Intelligent devices, Android application, FPGA, multiplexers, programmable interconnect points, tracks congestion, configuration memory.

Índice de contenidos

Índice de figuras	XII
1 INTRODUCCIÓN.....	1
1.1 Motivación	1
1.2 Objetivos	2
1.3 Organización de la memoria	3
2 ESTADO DEL ARTE	5
2.1 Introducción	5
2.1 Tablets	5
2.1.1 Utilidades.....	6
2.1.2 Sistema Operativo Tablets	7
2.2.2.1 Sistemas Operativos basados en los de un ordenador tradicional.....	7
2.2.2.2 Sistemas Operativos post-PC	7
2.1.3 Sistema Operativo Android	8
2.1.3.1 Características	8
2.1.3.2 Arquitectura	9
2.1.3.3 Nombres de las versiones	9
2.1.3.4 Aplicaciones.....	10
2.1.3.5 Lenguaje de programación.....	10
2.1.3.6 Componentes de una aplicación Android	10
2.1.4 Comparativa entre SO	12
2.2 FPGAs	13
2.2.1 Programación	13
2.2.2 Aplicación Logical Gate Simulator	13
2.2.3 Robei FPGA Simulation.....	14
3 DISEÑO	16
3.1 Introducción	16
3.2 La enseñanza	16
3.2.1 El avance de las tecnologías y la enseñanza	16
3.3 Elección del proyecto	18
3.4 Diseño del sistema	18

3.4.1	Herramientas necesarias.....	18
3.4.2	Pasos previos.....	18
3.5	Requisitos.....	19
3.5.1	Requisitos del proyecto.....	19
3.5.2	Requisitos del entorno para el uso de la App	19
4	DESARROLLO.....	22
4.1	Introducción	22
4.2	Descripción de la aplicación	22
4.2.1	Realización de un ejercicio	23
4.2.2	Importación de un ejercicio	28
4.3	Desarrollo de la aplicación	29
4.3.1	Layout 1.....	29
4.3.2	Layout 2.....	29
4.3.3	Layout 3.....	30
4.3.4	Layout 4.....	30
4.3.5	Layout 5.....	31
4.3.6	Layout 6.....	32
4.3.7	Layout 7.....	32
4.3.8	Layout 8.....	33
4.3.9	Layout 9.....	33
4.3.10	Layout 10.....	34
4.4	Desarrollos futuros.....	35
5	MANUAL DE USUARIO.....	37
5.1	Introducción	37
6	MANUAL DEL PROFESOR	49
6.1	Introducción	49
7	PRUEBAS Y EJEMPLOS	55
7.1	Introducción	55
7.2	Ejercicio #1	56
7.3	Ejercicio #2	58
7.4	Ejercicio #3	60
7.5	Ejercicio #4	62
7.6	Ejercicio #5	65
7.7	Ejercicio #6	68

7.8	Ejercicio #7	71
7.9	Ejercicio #8	75
7.10	Ejercicio #9	78
7.11	Ejercicio #10	82
7.12	Ejercicio #11	84
8	REFERENCIAS	86
9	CONCLUSIONES.....	89
	ANEXOS.....	91
10	MANUAL DEL PROGRAMADOR	91
10.1	NewExercise	91
10.2	NewExercise1	96
10.3	Clase SendEmailActivity	100
10.4	Clase ContinueExercise	101
10.5	Clase GuardarEjercicio	102
11	PRESUPUESTO.....	104
12	PLIEGO DE CONDICIONES.....	105

Índice de figuras

Figura 2-1: Tablets [45]	5
Figura 2-2: Gráfico SO PC tradicional [46].....	7
Figura 2-3: Gráfico SO post-PC [46].....	7
Figura 2-4: Ventas y estimaciones de ventas según SO [48].....	12
Figura 2-5: Logic Gate Simulator[49].....	13
Figura 4-1: Menú inicial.....	23
Figura 4-2: Listado ejercicios.....	23
Figura 4-3: Enunciado.....	24
Figura 4-4: Layout principal.....	24
Figura 4-5: Layout dibujo	25
Figura 4-6: Guardando ejercicio.....	25
Figura 4-7: Seleccionando ejercicio a enviar.....	26
Figura 4-8: Enviando ejercicio	26
Figura 4-9: Memoria.....	27
Figura 4-10: Importación ejercicio guardado.....	28
Figura 5-1: Icono app.....	37
Figura 5-2: Menú principal	38
Figura 5-3: Listado ejercicios.....	39
Figura 5-4: Importación ejercicio	39
Figura 5-5: Ejemplo enunciado.....	41
Figura 5-6: Layout principal.....	41
Figura 5-7: Ejemplo interconexión	42
Figura 5-8: Dibujando líneas.....	44
Figura 5-9: Guardar ejercicio.....	45
Figura 5-10: Datos correo.....	46
Figura 5-11: Memoria.....	47
Figura 6-1: Icono app.....	49
Figura 6-2: Menú principal	50
Figura 6-3: Importar ejercicio.....	51
Figura 6-4: Ejercicio cargado	51
Figura 6-5: Memoria.....	53
Figura 7-1: Layout principal.....	56
Figura 7-2: CLB31.....	56
Figura 7-3: Enunciado.....	58
Figura 7-4: Layout principal.....	58
Figura 7-5: CLB31.....	59
Figura 7-6: Enunciado.....	60
Figura 7-7: Layout principal.....	60
Figura 7-8: CLB31.....	61
Figura 7-9: Layout principal.....	62
Figura 7-10: CLB31.....	62
Figura 7-11: CLB33.....	63

Figura 7-12: Enunciado.....	65
Figura 7-13: Layout principal.....	65
Figura 7-14: CLB31, CLB11 y CLB13	66
Figura 7-15: CLB33.....	66
Figura 7-16: Enunciado.....	68
Figura 7-17: Layout principal.....	68
Figura 7-18: CLB31 y CLB11	69
Figura 7-19: CLB33.....	69
Figura 7-20: Enunciado.....	71
Figura 7-21: Layout principal.....	71
Figura 7-22: CLB31 y CLB11	72
Figura 7-23: CLB22.....	72
Figura 7-24: CLB13.....	73
Figura 7-25: Layout principal.....	75
Figura 7-26: CLB31.....	76
Figura 7-27: CLB33.....	76
Figura 7-28: CLB11.....	77
Figura 7-29: Enunciado.....	78
Figura 7-30: Layout principal.....	78
Figura 7-31: CLB31.....	79
Figura 7-32: CLB11.....	79
Figura 7-33: CLB22.....	80
Figura 7-34: CLB33.....	80
Figura 7-35: Enunciado.....	82
Figura 7-36: CLB31.....	82
Figura 7-37: CLB31.....	83

1 INTRODUCCIÓN

1.1 Motivación

Dado que las tecnologías están abaratándose a pasos agigantados, el uso de las tabletas y teléfonos inteligentes se ha hecho común en la sociedad y mucho más entre los estudiantes. La enseñanza no puede quedarse a la cola; por eso mismo, el desarrollo de este proyecto se enmarca en estas nuevas circunstancias.

Hoy en día, cualquier estudiante tiene una tableta o teléfono inteligente, y de no ser así, sabe manejarlo correctamente. Por esto mismo, el uso de estos aparatos en la vida diaria de la universidad adquiere cada vez más importancia, hasta el nivel de desarrollar una aplicación en la que el alumno y el profesor puedan interactuar, facilitando así la captación de conocimientos por parte de los primeros a la vez que hace más cómodo al profesor impartir las clases y comprobar los conocimientos adquiridos por parte de sus alumnos.

Por todo ello, la implementación de esta aplicación ayudará a la sociedad universitaria y será una buena base para el comienzo del uso de aplicaciones destinadas a la interacción entre alumnos y profesores.

De entre la inmensidad de temas que se podría haber optado para realizar la plantilla sobre la que posteriormente se ejecutan los ejercicios, se ha elegido el tema de las FPGAs, puesto que es una herramienta que es muy dinámica, permitiendo incluir gran cantidad de ejercicios de varios tipos.

1.2 Objetivos

Los objetivos principales de este proyecto son, principalmente, mejorar la adquisición de conocimientos por parte del alumno, facilitar la evaluación de los alumnos por parte del profesor sobre los temas impartidos en clase y por último optimizar la interacción entre el personal docente con sus alumnos.

Todo ello se lleva a cabo desarrollando una aplicación en la que el alumno podrá observar los enunciados de varios ejercicios, para posteriormente comenzar a resolverlo usando todas las funciones que se han creado para ello, como pueden ser:

- Mapeo de circuitos combinacionales con multiplexores.
- Conexión a GND de los elementos que no se utilizan.
- Mapeo de circuitos secuenciales sencillos.
- Comprobación del estado del ejercicio.
- Visualización de las posiciones de memoria usadas.
- Dibujo de las pistas usadas para la interconexión.

Tras esto, el alumno podrá guardar el ejercicio, añadiendo el nombre que él desee.

El alumno también tendrá la opción de enviarle el ejercicio al correo del profesor o a cualquier otro correo.

Otra de las funcionalidades creadas para esta aplicación, es la importación de cualquier ejercicio guardado o recibido al correo (siempre que éste se guarde en la tarjeta de memoria) para su posterior carga en la aplicación. Esta carga de datos será de las posiciones de memoria, y de los dibujos de las pistas, para que se pueda continuar con el ejercicio o llevar a cabo su evaluación por parte del profesor.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:** Se explica el objetivo del proyecto.
- **Estado del arte:** Se establecerá el marco teórico en el que se encuentra este PFC
- **Diseño:** Este capítulo se centrará en la implementación y diseño del sistema.
- **Desarrollo de la aplicación y trabajo futuro:** Se detallará cada módulo implementado y las funciones y parámetros utilizados. También se añade una parte en la que se indicarán posibles mejoras para la aplicación.
- **Manual de Usuario:** En este capítulo se definirá el manual para que el alumno pueda usar la aplicación y así sea autónomo al 100%.
- **Manual del Profesor:** En este capítulo se definirá el manual para que el profesor pueda usar la aplicación y así sean autónomo al 100%.
- **Pruebas y ejemplos:** Ya que se ha usado una plantilla común para todos los ejercicios, se realizarán pruebas con los distintos ejercicios que servirán de ejemplo y guía para los alumnos.
- **Anexos:** Serie de anexos referentes a información complementaria acerca de los capítulos del proyecto.

2 ESTADO DEL ARTE

2.1 Introducción

Desde que el mercado de los Smartphone comenzó a funcionar, ha evolucionado a pasos agigantados, llegando incluso a hacerse indispensable para mucha gente tener uno de estos equipos en su vida diaria.

Por esto mismo, aprovechando esta enorme demanda sobre este tipo de dispositivos por parte de la sociedad, se realiza el proyecto en el que estamos inmersos. Ya que mucha gente hace uso de estos dispositivos inteligentes, se quiere facilitar el desarrollo de acciones cotidianas, como podría ser el estudio de una asignatura.

2.1 Tablets

La evolución de la tecnología desde sus comienzos hasta la actualidad, ha hecho que los tamaños de los dispositivos que anteriormente se usaban, se hayan visto reducidos considerablemente. Hoy en día los dispositivos electrónicos son portátiles, ya que se pueden llevar a donde se desee, realizando siempre o casi siempre, las mismas funciones que un ordenador normal y corriente.



Figura 2-1: Tablets [45]

Según [44], estos dispositivos son:

“Una tableta es una computadora portátil de mayor tamaño que un teléfono inteligente o una PDA, integrado en una pantalla táctil con la que se interactúa primariamente con los dedos o una pluma stylus (pasiva o activa), sin necesidad de teclado físico ni ratón. Estos últimos se ven reemplazados por un teclado virtual. “

Este tipo de dispositivos se presenta en varios formatos:

- Estándar (pizarra/slate), que no posee teclado.
- Convertible, dispone de un teclado anclado al terminal, pudiendo moverle para poder usarlo u ocultarlo.
- Híbrido, en el que el teclado existente se puede separar del dispositivo.
- Booklets. Estas tablets disponen de dos pantallas, siendo al menos una de ellas táctil.

Actualmente todos o casi todos los fabricantes de equipos electrónicos se han introducido en la producción de tablets y esto ha hecho que exista una gran diversidad de estos dispositivos en cuanto al tamaño, precio, sistemas operativos (SO), etc.

2.1.1 Utilidades

- Lectura de libros electrónicos
- Consulta y edición de documentos
- Navegación web (mediante Wi-Fi, USB o 3G Interno)
- Llamadas telefónicas
- GPS
- Reproducción de música
- Visualización de vídeos y películas
- Cámara fotográfica y de vídeo HD
- Videoconferencia
- Etc.

2.1.2 Sistema Operativo Tablets

Existen dos tipos de sistemas operativos con los que las tablets pueden funcionar:

2.2.2.1 Sistemas Operativos basados en los de un ordenador tradicional.

Los más conocidos y extendidos entre los consumidores de Tablets, son el Windows de Microsoft y varios SO de sistemas de Linux.

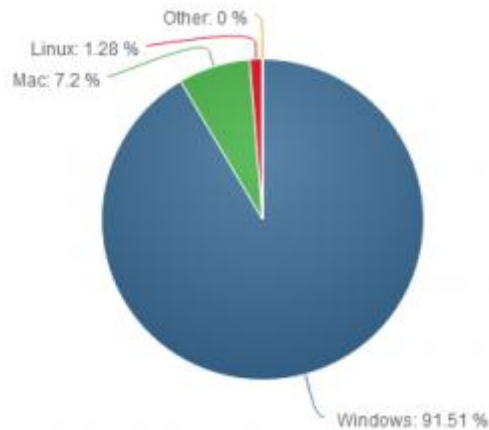


Figura 2-2: Gráfico SO PC tradicional [46]

2.2.2.2 Sistemas Operativos post-PC (similares a los SO de los dispositivos móviles inteligentes).

En este grupo se encuentran los SO más populares para las tablets, como el iOS de Apple y el Android de Google. Se están empezando a encontrar otros SO, como el Chrome OS de Google, Symbian, Windows Phone, etc.

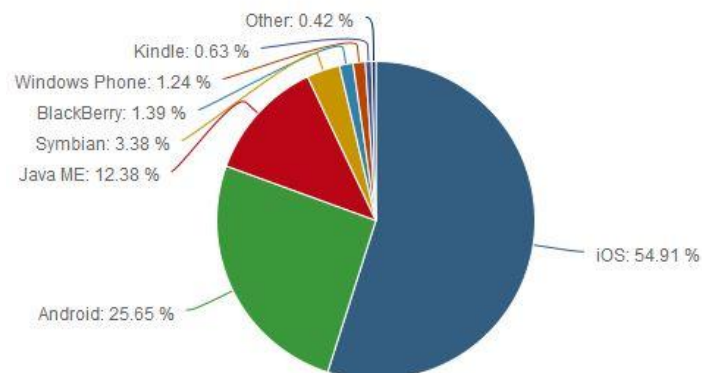


Figura 2-3: Gráfico SO post-PC [46]

La siguiente es una lista de algunos sistemas operativos disponibles para Tablets:

- Android
- iOS
- Ubuntu Touch
- Chrome OS
- BlackBerry Tablet OS
- Windows CE
- Windows Phone
- Windows RT
- Windows 8

2.1.3 Sistema Operativo Android

Android es un SO basado en el sistema Linux, que inicialmente fue creado para dispositivos móviles con pantalla táctil como los Smartphone, aunque con el avance del tiempo y por el aumento de tipos de dispositivos, se decidió usar en las tablets también.

El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

2.1.3.1 Características (Reproducido de Wikipedia)

Estas son las características principales del SO de Android:

- Diseño de dispositivo
- Almacenamiento
- Conectividad
- Mensajería
- Navegador web
- Soporte de Java
- Soporte multimedia
- Soporte para streaming
- Soporte para hardware adicional
- Entorno de desarrollo
- Google Play
- Multi-táctil
- BluetoothVideollamada
- Multitarea
- Características basadas en voz
- Tethering

2.1.3.2 Arquitectura

Estos son los componentes principales del sistema operativo de Android:

- **Aplicaciones:** Este sistema operativo tiene entre las aplicaciones principales un cliente de correo electrónico, calendario, un navegador, cámara de fotos, reproductor de música, mapas, contactos, etc. El lenguaje usado para programar estas aplicaciones es Java.
- **Marco de trabajo de aplicaciones:** Los APIs del framework de las apps principales están disponibles para los desarrolladores. La arquitectura está diseñada para simplificar la reutilización de componentes.
- **Bibliotecas:** Las bibliotecas también son accesibles para los desarrolladores y son usadas por ciertas partes del sistema Android.
- **Runtime de Android:** Ya que para programar en Android se usa el lenguaje Java, en Android se incluye un conjunto de bibliotecas en las cuales se encuentran la mayoría de las funciones que aportan las bibliotecas principales de Java.
- **Núcleo Linux:** El núcleo de Android es Linux, y de él dependen los servicios principales del sistema por ejemplo la seguridad.

2.1.3.3 Nombres de las versiones (Reproducido de Wikipedia)

Estos son los nombres que tienen las diferentes versiones de Android:

- A: Apple Pie (v1.0)
- B: Banana Bread (v1.1)
- C: Cupcake (v1.5)
- D: Donut (v1.6)
- E: Éclair (v2.0/v2.1)
- F: Froyo (v2.2), (Abreviatura de «frozen yogurt»)
- G: Gingerbread (v2.3)
- H: Honeycomb (v3.0/v3.1/v3.2)
- I: Ice Cream Sandwich (v4.0)
- J: Jelly Bean (v4.1/v4.2/v4.3)
- K: Key Lime Pie (v5.0)

2.1.3.4 Aplicaciones

Hay diversas formas de desarrollar aplicaciones para Android, pero la principal y más común es usando Java con el SDK (Software Development Kit) de Android. Con eso y con ciertos conocimientos de Java se podrán llevar a cabo.

Como decía, existen más formas de desarrollar aplicaciones, puede ser un ejemplo la aplicación Google App Inventor.

Una vez las aplicaciones están desarrolladas, para instalarlas en el terminal, se hará con el archivo comprimido “APK”, que tan solo ejecutándolo en el dispositivo se instalará y se podrá hacer uso de dichas apps.

2.1.3.5 Lenguaje de programación

He comentado que para desarrollar aplicaciones en Android se usa el lenguaje Java, pero esto no es del todo cierto, puesto que sí que se utiliza el lenguaje como tal, la sintaxis y semántica de Java, pero existen ciertas bibliotecas de clases de Java y APIs que no emplea.

2.1.3.6 Componentes de una aplicación Android

Como se puede observar en [47] los distintos tipos de componentes de software con los que podremos construir una aplicación Android son:

“Activity

Las actividades (activities) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual.

View

Las vistas (view) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análoga por ejemplo a los controles de Java o .NET. De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Service

Los servicios (service) son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario.

Content Provider

Un proveedor de contenidos (content provider) es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

Broadcast Receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada", ...) o por otras aplicaciones (cualquier aplicación puede generar mensajes (intents, en terminología Android) broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

Widget

Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

Intent

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc."

2.1.4 Comparativa entre SO

Ya se han mencionado varios sistemas operativos, pero llega el momento de hacer hincapié en los más extendidos entre los usuarios de tablets.

1. Android

Es el sistema operativo más extendido entre las tablets, es decir, entre los diferentes modelos existentes es la más común entre todos. Se desarrolla por una compañía que es propiedad de Google. La mayoría de las tabletas funcionan actualmente con la versión 4.0, o Ice Cream Sandwich.

2. iOS

Es el sistema operativo que tiene el iPad, y éste tiene al menos un 60% del total de ventas de tablets, por lo que es el sistema operativo que más utiliza el usuario de las tablets. Este sistema operativo es desarrollado por Apple. Actualmente va por la versión iOS 6, aunque en breves se lanzará la versión iOS 7.

3. Blackberry Tablets OS

Este sistema operativo se desarrolló para los terminales Smartphone de Blackberry, pero en términos de ventas, no ha obtenido resultados notables.

4. Windows RT

Este último sistema operativo que se menciona en la lista de los más utilizados por los usuarios, es propiedad de la multinacional Microsoft. Se ha desarrollado específicamente para las tablets.

A continuación se puede observar una tabla en la que se muestra unas estadísticas sobre las ventas y previsiones de ventas de dispositivos móviles en base a su sistema operativo.

Por ejemplo, en 2012 se vendieron el 49.8% de dispositivos móviles con SO Android y un 18.3% de dispositivos con iOS.

Global Smartphone UB by OS : % of Total	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Android	0.0%	0.3%	2.2%	13.0%	33.7%	49.8%	56.4%	57.5%	56.3%	54.2%	51.9%
Apple iOS	1.5%	5.1%	8.9%	11.8%	15.1%	18.3%	20.5%	21.1%	21.1%	20.8%	20.4%
Bada	0.0%	0.0%	0.0%	0.6%	1.4%	1.5%	1.4%	1.4%	1.4%	1.3%	1.3%
Blackberry OS	9.0%	11.2%	13.9%	13.7%	10.8%	6.8%	4.9%	4.2%	4.0%	3.9%	3.8%
Firefox OS	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%	0.6%	1.0%	1.4%	1.7%
LiMo	0.0%	2.2%	2.5%	1.9%	1.1%	0.5%	0.2%	0.1%	0.1%	0.0%	0.0%
MeeGo	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%	0.1%	0.0%	0.0%	0.0%	0.0%
Microsoft	12.7%	12.7%	10.9%	7.2%	4.0%	3.9%	5.2%	7.8%	10.3%	12.5%	14.2%
Palm OS / webOS	1.6%	1.5%	1.1%	0.7%	0.2%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
Symbian	64.8%	59.3%	54.7%	47.8%	32.0%	18.2%	10.2%	5.9%	3.6%	2.3%	1.5%
Tizen	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.3%	1.0%	1.7%	2.4%	2.9%
Others	10.5%	7.8%	5.7%	3.3%	1.5%	0.7%	0.4%	0.4%	0.7%	1.2%	2.3%
Total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Figura 2-4: Ventas y estimaciones de ventas según SO [48]

2.2 FPGAs

Según la definición que se encuentra en la Wikipedia[52]:

“Una FPGA (del inglés Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada 'in situ' mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.”

2.2.1 Programación

En este proyecto se quiere realizar una aplicación en el que el estudiante de la asignatura DIE trabaje de forma similar a como lo haría un programa configurador de FPGA. Como en las FPGAs reales, la implementada tendrá celdas que se configurarán con una función específica, dependiendo de los ejercicios. El bloque básico de la FPGA contendrá un multiplexor y una función lógica de 3 entradas. Por esto mismo, la tarea del usuario de la aplicación, es saber configurar la función lógica que realizará cada uno de los bloques lógicos, seleccionar los IOB (Input Output Blocks) que apliquen e interconectarlos entre sí usando las múltiples pistas.

2.2.2 Aplicación Logical Gate Simulator

“Logic Gate Simulator” es una herramienta de código abierto destinada a experimentar y a aprender sobre las puertas lógicas.

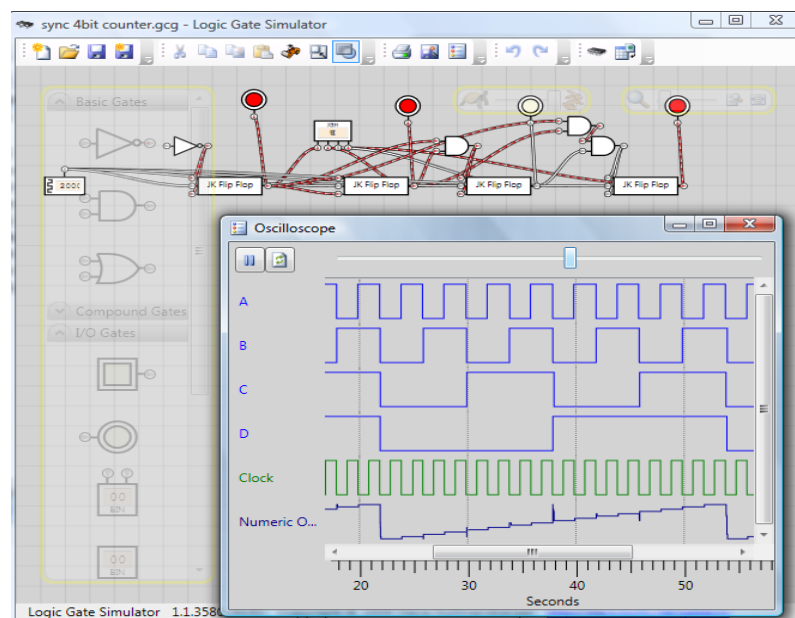


Figura 2-5: Logic Gate Simulator[49]

2.2.3 Robei FPGA Simulation

“Robei es una herramienta de diseño de chips de nueva generación, que tiene como objetivo simplificar la interfaz de usuario y el aumento de la visibilidad. Se introduce un nuevo método de diseño intuitivo para el prototipado rápido mediante la combinación de las ventajas de ambos gráficos y codificación.

Después de la simulación, se puede generar el código Verilog estándar que se puede utilizar con otras herramientas EDA. Robei no sólo es una gran herramienta de educación para los estudiantes y profesores, también es una gran herramienta de desarrollo para la entrada a los ingenieros de nivel medio”

Fuente: [50]

ElectroDroid

“ElectroDroid es una aplicación Android para los que se dedican a la electrónica como aficionados y también para los que hacen de ella una profesión.

ElectroDroid es una colección de herramientas electrónicas y de referencia que incluye:

- *Decodificador del código de color de Inductores;*
- *Calculadora de la Ley de Ohm;*
- *Calculadora de Reactancia;*
- *Divisor de Voltaje;*
- *Razón de Resistores;*
- *Cálculo de Carga de Capacitador;*
- *Amplificador Operacional;*
- *Calculadora de Resistor LED;*
- *Calculadora LM317;*
- *Disipación de Calor;*
- *Calculadora de Baterías; “*

Fuente: [51]

3 DISEÑO

3.1 Introducción

Previamente se ha realizado el estado del arte, y así se ha podido analizar en qué estado se encuentran algunas de las aplicaciones de tutoriales para dispositivos móviles. En este análisis también se han estudiado los diferentes sistemas operativos que se encuentran en el mercado para las distintas Tablets y Smartphones. Tras todo esto, se pasa a detallar el diseño de la aplicación, porqué se ha elegido realizar este proyecto y todo lo que es necesario para ello.

3.2 La enseñanza

En este proyecto se va a realizar una aplicación para así enseñar a los alumnos de la asignatura DIE (EPS UAM), a completar ejercicios y aprender los conceptos necesarios. Para ello se hará una mención a la relación existente entre el avance de la tecnología y la enseñanza.

3.2.1 El avance de las tecnologías y la enseñanza

Como se viene diciendo en capítulos anteriores, la tecnología está avanzando a pasos agigantados, sin dar tregua a nadie y modificando cada día aspectos de nuestra vida diaria, por lo que la enseñanza y sus métodos no iban a ser ajenos a estos cambios y mejoras.

Para enseñar a los alumnos, antes se usaban sólo los libros y pizarras, pero poco a poco se han ido utilizando dispositivos electrónicos que ayudaban a impartir las clases, como podría ser un proyector con el que se mostraban ciertas diapositivas, ordenadores, ya fueran portátiles o sobremesas, altavoces, etc.

Hace unos años, los ordenadores se comienzan a usar en ciertas asignaturas o lecciones de asignaturas para que todos los alumnos de una clase tengan acceso a ciertas herramientas o material, teórico o práctico, y así optimizar aún más la metodología de trabajo, puesto que todos los alumnos pueden acceder a la vez a este material, pero individualmente, es algo que sin el avance de la tecnología no se habría podido conseguir. Un ejemplo de esto podría ser que en la asignatura de “Tecnología” hay que simular un circuito electrónico, gracias a los ordenadores se tiene la aplicación necesaria para hacerlo y así todos los alumnos podrían acceder a dicha aplicación y realizar el ejercicio individualmente.

Pero los ordenadores en las aulas tienen un problema, si son sobremesa, no puedes llevarlos a ningún sitio fácilmente y se necesita un aula especial para poder tenerlos todos. Por lo que con los portátiles quizás se solventa dicho problema de movilidad, pero no son económicamente tan accesibles como podrían serlo las Tablets.

Al aparecer las Tablets en el mercado se aporta al usuario una facilidad de movimiento enorme, ya que puedes tener un “ordenador”, con menor rendimiento, pero te lo puedes

llevar a cualquier sitio, pudiendo realizar casi la misma operativa que se haría con un ordenador portátil o sobremesa.

Por esto mismo, si de alguna manera se pudieran extender el uso de las Tablet para enseñar conceptos de asignaturas, o incluso usarlo de forma habitual en las clases, aportaría mucho valor, tanto para los usuarios, como para la enseñanza en general. Simplemente hay que adaptar algunos de los programas que se usan en los ordenadores y crear nuevas aplicaciones y herramientas que se ajusten a la metodología de enseñanza.

3.3 Elección del proyecto

Tras haber relacionado el avance de la tecnología con la enseñanza, la justificación de la elección del proyecto es muy sencilla, se quiere extender el uso de las Tablets en las aulas y así facilitar la enseñanza tanto al alumnado como al profesorado. Se quiere desarrollar una aplicación en la que el alumno pueda realizar ejercicios prácticos para así aplicar los conceptos teóricos previamente adquiridos por las lecciones del profesor.

3.4 Diseño del sistema

En este apartado se detallan las directrices a seguir para poder desarrollar la aplicación.

3.4.1 Herramientas necesarias

Para poder desarrollar la aplicación son necesarias ciertas herramientas o dispositivos que paso a listar a continuación:

- Ordenador portátil o sobremesa
- Eclipse
- SDK de Android
- Simulador ADT de Android
- Tablet con sistema operativo de Android

3.4.2 Pasos previos

Previo al comienzo del desarrollo de la aplicación, es necesario tener ciertos conocimientos de lenguajes de POO, y más concretamente de Java, ya que Android utiliza la misma semántica que usa Java.

También, para poder desenvolverse al ir desarrollando la aplicación, se debería conocer el entorno en el que se va a realizar la implementación del código, que en este caso se hará usando la herramienta Eclipse con el SDK de Android y el simulador ADT de Android. Para ello se aconseja que antes de comenzar con el código de la aplicación del proyecto, se realicen ciertos ejercicios o ejemplos de aplicaciones más sencillas, por ejemplo, siguiendo tutoriales.

En el caso de este proyecto será necesario realizar ejercicios previos en los que se tenga que usar la clase Canvas, ya que será de suma importancia poder realizar ciertos dibujos en la aplicación.

3.5 Requisitos

Tanto los requisitos del proyecto en general, como los necesarios en el entorno para poder usar la aplicación, se definen a continuación y se han consensuado con el tutor a la hora de decidir cómo se iba a realizar el proyecto.

3.5.1 Requisitos del proyecto

En esta aplicación los usuarios podrán realizar las siguientes funciones:

- Mapeo de circuitos combinacionales con multiplexores.
- Conexión a GND de los elementos que no se utilizan (antigua opción TIE en Xilinx).
- Mapeo de circuitos secuenciales sencillos.
- Comprobación del estado del ejercicio.
- Visualización de las posiciones de memoria usadas.
- Dibujo de las pistas usadas para la interconexión.
- Se podrá guardar el ejercicio, añadiendo el nombre que él desee.
- El alumno también tendrá la opción de enviarle el ejercicio al correo del profesor o a cualquier otro correo.
- Otra de las funcionalidades creadas para esta aplicación, es la importación de cualquier ejercicio guardado o recibido al correo (siempre que éste se guarde en la tarjeta de memoria, en la carpeta "Downloads") para su posterior carga en la aplicación. Esta carga de datos será de las posiciones de memoria, y de los dibujos de las pistas, para que se pueda continuar con el ejercicio o llevar a cabo su evaluación por parte del profesor.

3.5.2 Requisitos del entorno para el uso de la App

Los requisitos del entorno para poder usar la aplicación sin problemas son:

- Tablet con sistema operativo Android
- Versión de Android 4.1
- No se podrá usar en dispositivos con pantallas menores a 10", ya que se perderían la correcta disposición de los componentes.
- Uso del cliente de correo electrónico del sistema operativo de Android, no otro.

- No se utilizarán los botones de la Tablet, ni los propios del SO, mientras se interactúa con la aplicación, ya que se ha realizado el proyecto teniendo en cuenta que el usuario solo interactuará con la aplicación y sus opciones.
- Debe existir una tarjeta de memoria SD en el dispositivo, o en cuyo caso, una parte de la memoria que actúe como tal y así la aplicación, en el caso de que no exista, pueda localizar esa parte de memoria al utilizar esta llamada:
“*Environment.getExternalStorageDirectory()*” en la que se obtiene el path de la tarjeta de memoria SD.
- Los documentos descargados desde el correo, se guardarán en la carpeta “Downloads” de la tarjeta SD o en la parte de memoria que actúa como tarjeta SD.

4 DESARROLLO

4.1 Introducción

Para desarrollar la aplicación, una vez que se han realizado algunos de los pasos previos mencionados en el apartado de diseño, como tener el entorno de desarrollo y haber hecho algunos ejercicios básicos para tener cierta práctica, se continúa con el desarrollo de la aplicación propiamente dicha. Siempre se tendrán en cuenta todos los requisitos con los que tiene que contar el proyecto.

4.2 Descripción de la aplicación

En este apartado se detallan las funcionalidades principales de la aplicación de cara al usuario de la misma, es decir, se explicará todo lo que puede realizar tanto el alumno como el profesor con la aplicación.

Pese a que se trata de una aplicación que intenta asemejarse a un entorno gráfico de configuración de FPGAs, por lo que podría parecer muy complejo, se ha intentado que para el usuario sea lo más sencillo posible, y así pueda interactuar con la misma sin muchas complicaciones.

Las funciones principales de la aplicación son dos: la realización de ejercicios y la importación de los mismos.

4.2.1 Realización de un ejercicio

Inicialmente, en el menú principal se podrá elegir entre tres opciones, salir de la aplicación, importar un ejercicio o elegir uno entre una lista previamente definida.

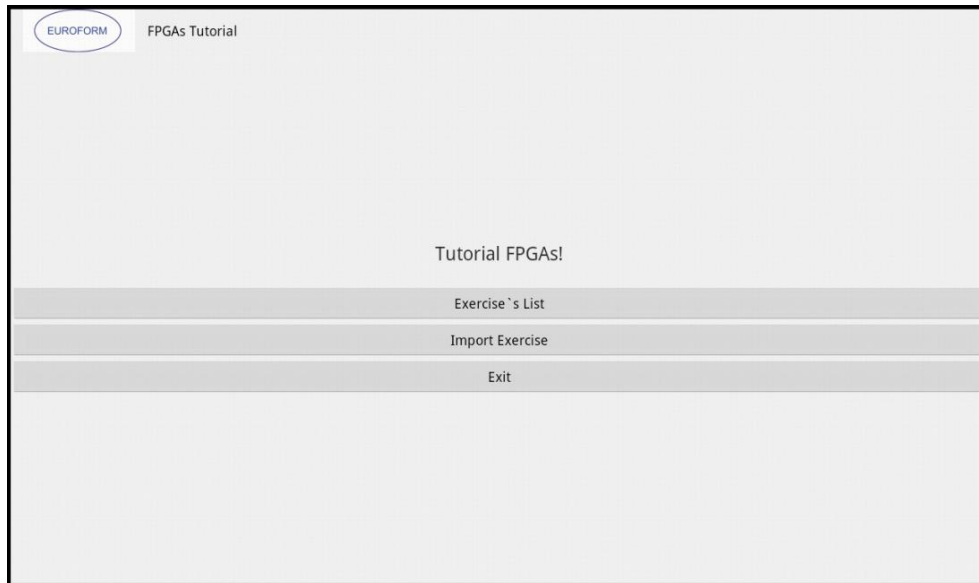


Figura 4-1: Menú inicial

1. En este caso se elige la opción "Exercise's List", es decir, se selecciona el botón en el que se pasa a elegir un ejercicio de la lista predefinida.
2. Tras esto se da la opción de escoger uno de los ejercicios de la lista. Esta lista se ha definido con el tutor de la asignatura y en ella hay 10 enunciados de ejercicios y un ejercicio sin enunciado. Se explicarán más adelante los ejercicios y en que consiste cada uno de ellos.

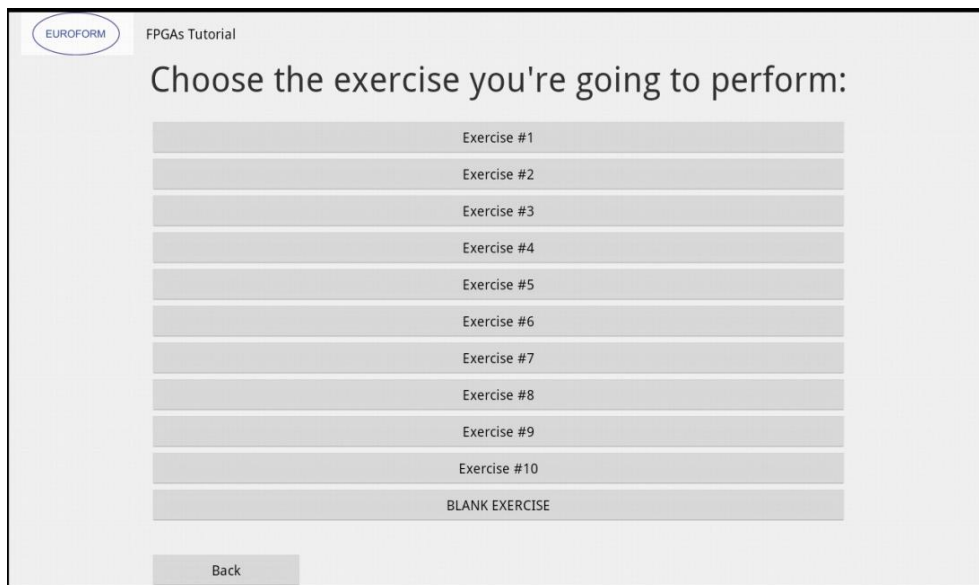


Figura 4-2: Listado ejercicios

- Una vez se elige uno de los ejercicios de la lista (el ejercicio "Blank Exercise" funcionará de forma distinta y se explicará más adelante), se muestra el enunciado del mismo, en el que se detallan los requisitos para realizar correctamente el ejercicio.

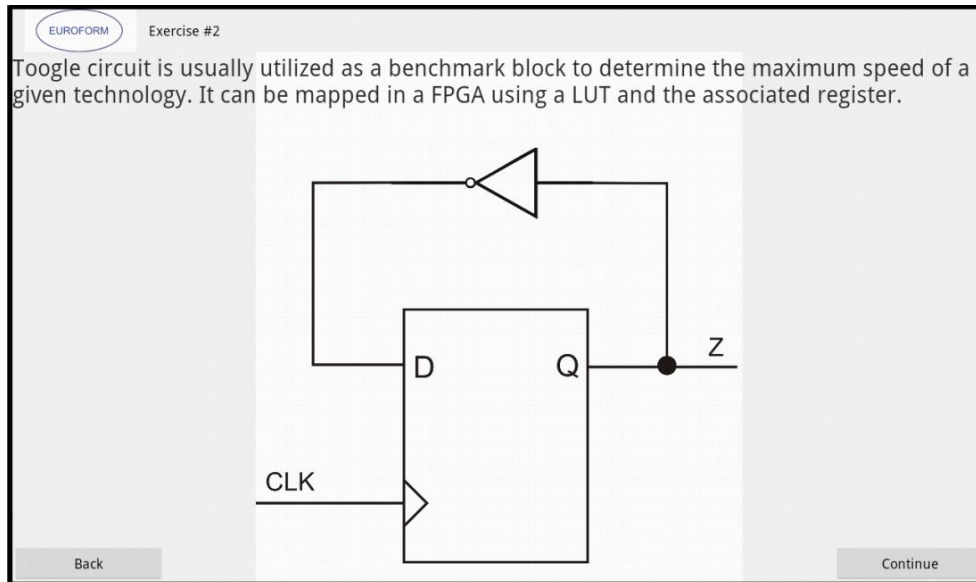


Figura 4-3: Enunciado

- En el siguiente paso, aparece el layout principal del ejercicio. En este layout se tendrá que mapear la interconexión de pistas, seleccionar las entradas y salidas de la FPGA, si se han de mapear puntos de la FPGA con VCC o GND del chip y si alguno de los CLB de la FPGA requiere del uso del CLK también se deberá seleccionar en dicho layout.

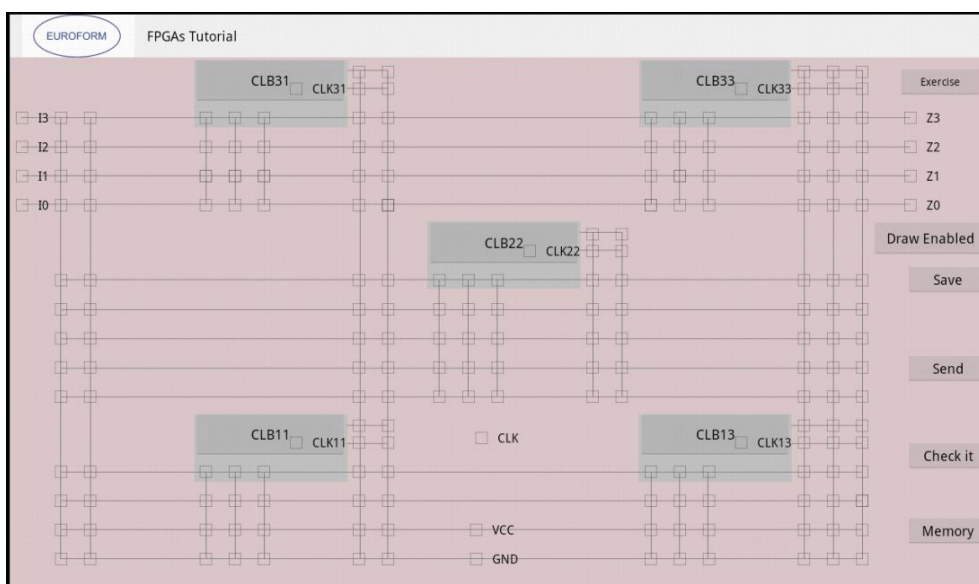


Figura 4-4: Layout principal

En este layout también aparecen varios botones que dan acceso a ciertas funciones de la aplicación:

- a. El primero con el que nos encontramos es el botón “Exercise”, con el que tras pulsarlo, se vuelve al enunciado del ejercicio que se seleccionó previamente.
- b. El botón “Draw Enabled” da acceso a otra funcionalidad del ejercicio, en el que se permitirá dibujar el camino elegido cuando se mapea el circuito, con el fin de que el usuario facilite la corrección del ejercicio al tutor.

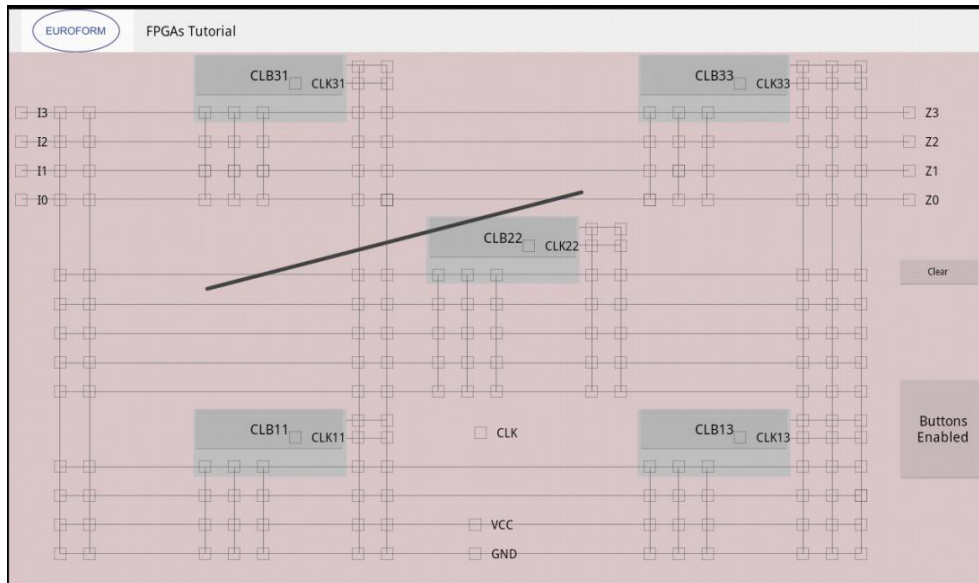


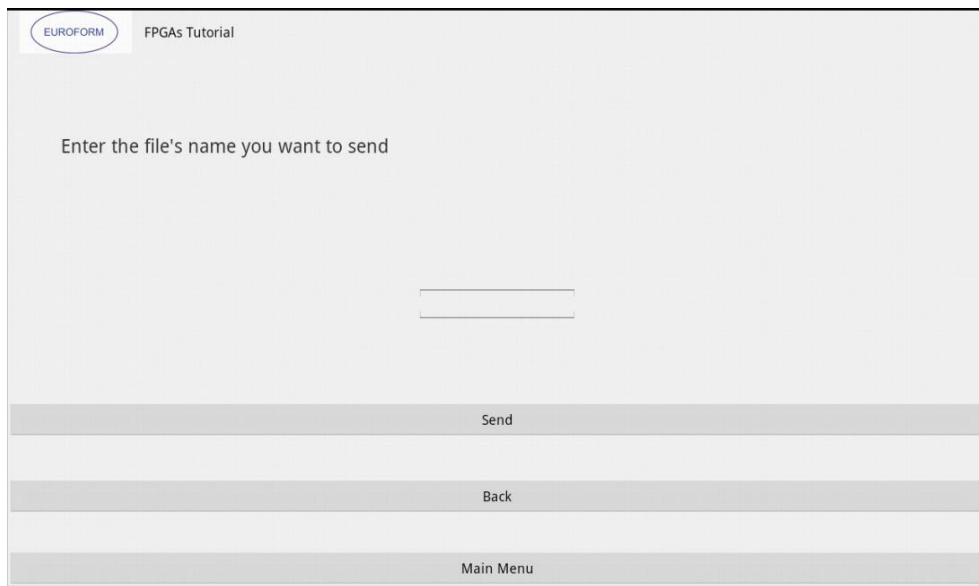
Figura 4-5: Layout dibujo

- c. Con el botón “Save” el usuario podrá guardar el ejercicio que ha estado resolviendo, tras haber escrito el nombre que el usuario desee, o el que el tutor necesite para poder corregirlo una vez se lo ha enviado el alumno.

Figura 4-6: Guardando ejercicio

- d. El siguiente botón con el que el usuario de la aplicación se encuentra, es el botón "Send". Con este botón, tras volver a escribir el nombre del archivo que se quiere enviar, se continúa con la creación del correo, en el que se escribirá la dirección de correo de la persona a la que se quiere enviar el ejercicio, el asunto, se podrá adjuntar o no los archivos y también se da la posibilidad de escribir un texto.

Una vez se ha realizado el punto anterior, se procede a enviar el correo eligiendo el cliente de correos, en el que se cargará toda la información que se ha rellenado.



The screenshot shows a user interface for sending a file. At the top left, there is a logo for "EUROFORM" and the text "FPGAs Tutorial". The main area contains the instruction "Enter the file's name you want to send" above a text input field. Below the input field are three buttons: "Send", "Back", and "Main Menu".

Figura 4-7: Seleccionando ejercicio a enviar



The screenshot shows the email composition screen. At the top left, there is a logo for "EUROFORM" and the text "FPGAs Tutorial". The form includes fields for "To:", "Subject:", and "Message:". Below the "Message:" field is a checkbox labeled "Send attachment". At the bottom of the form are three buttons: "Send", "Back", and "Main Menu".

Figura 4-8: Enviando ejercicio

- e. Con el botón "Check it" se podrá comprobar el estado de la realización del ejercicio, ya que con esta función se chequean individualmente, para cada ejercicio, las selecciones de varios de las entradas, salidas, CLKs, el VCC, el GND, etcétera, por lo que tras hacer la comprobación se dará un mensaje informativo de este estado.

- f. Por último se encuentra el botón "Memory", con el que se podrá observar tanto las posiciones de memoria como el interior de los CLB, y que valor tienen.

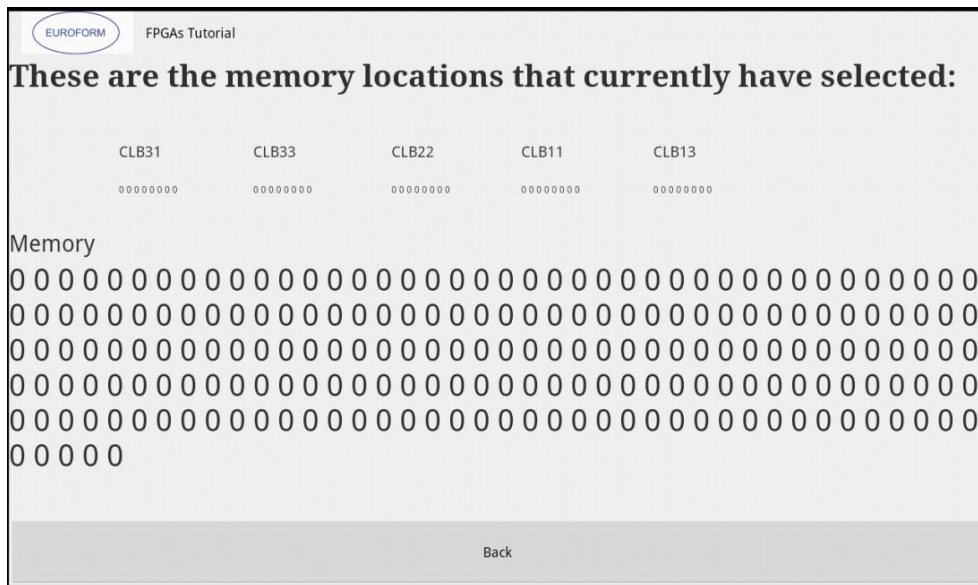
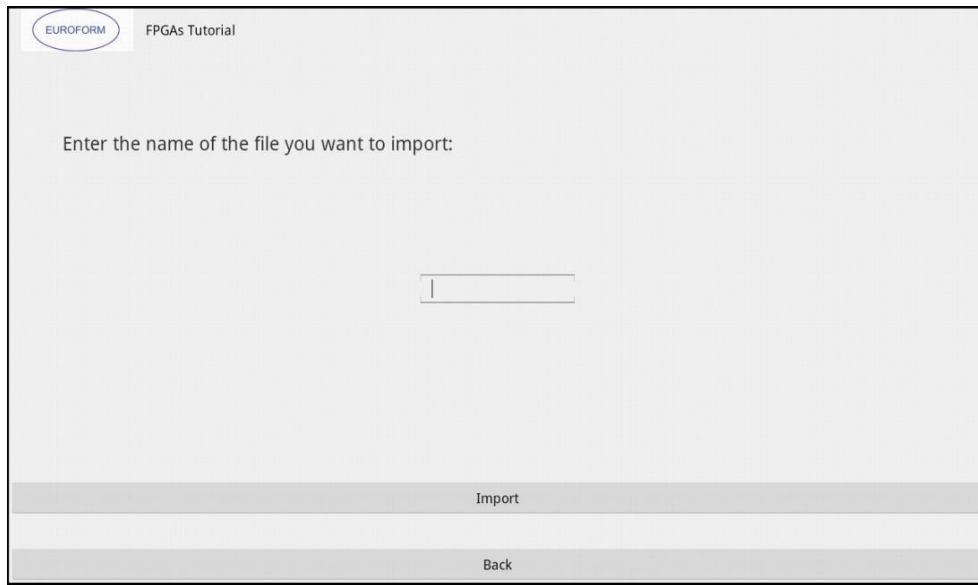


Figura 4-9: Memoria

4.2.2 Importación de un ejercicio

Recordemos, como se mencionó anteriormente, que en la página principal se podría elegir entre tres opciones, salir de la aplicación, importar un ejercicio o elegir uno entre una lista previamente definida.

1. En este caso se elige la opción “Import Exercise”, es decir, se procede a escribir el nombre del ejercicio que se quiere importar para corregir.



The screenshot shows a web interface for 'EUROFORM FPGAs Tutorial'. The main content area contains the instruction 'Enter the name of the file you want to import:' and a single-line text input field. Below the input field, there is a horizontal bar with two buttons: 'Import' and 'Back'.

Figura 4-10: Importación ejercicio guardado

2. Al estar importando un ejercicio que estaba ya completado, se cargará toda la información que el alumno había hecho, tanto los checkbox (las entradas, salidas, GCC, VCC, CLK, etc.) como el interior de los CLBs. También se cargarán los posibles dibujos que hubiera hecho el alumno.
3. Al realizar la importación de un ejercicio no se podrá volver atrás para comprobar el enunciado del ejercicio ni tampoco se podrá chequear el estado del mismo.
4. El tutor hará la corrección del ejercicio con sus conocimientos, no con los medios de la aplicación.

4.3 Desarrollo de la aplicación

En este punto de la memoria se describirán tanto las funciones del proyecto basándose en las clases que se han ido creando para cumplir los distintos requisitos establecidos, como la manera en la que se ha ido desarrollando.

- En primer lugar se decidió completar los menús y los layouts de la aplicación, en base a la distribución que más adelante se daría al escribir el código. Se fue desarrollando en el orden en el que el usuario irá encontrándose los distintos layouts, es decir:
 - Layout 1: Menú inicial
 - Layout 2: Listado con los ejercicios
 - Layout 3: Layout de importación del ejercicio y otro layout que es la réplica del layout principal de la FPGA con diferentes funciones extra.
 - Layout 4: Los diferentes enunciados de todos los ejercicios
 - Layout 5: El layout principal de la FPGA, donde se encuentran las pistas y CLBs
 - Layout 6: Layout del contenido de los distintos CLBs
 - Layout 7: El layout en el que el usuario podrá dibujar
 - Layout 8: Los layout usados al pulsar sobre “Save” y “Send”, en donde se escribirá un nombre de archivo
 - Layout 9: La interfaz gráfica en la que se pueden observar los valores de las posiciones de memoria y de los CLBs
 - Layout 10: Y por último el layout en el que el usuario escribe los datos del correo electrónico
- Una vez creada la interfaz gráfica se desarrolla la lógica de la aplicación:

En todas las clases, antes que cualquier otro método, se lanza el método onCreate, en el que se carga el layout correspondiente.

4.3.1 Layout 1

Se ha creado una clase (MainActivity) en la que se asocia para cada botón del layout una llamada a la actividad correspondiente y así lanzarla cuando el usuario pulse dicho botón. Hay 3 métodos en su interior, uno para cada botón del layout.

4.3.2 Layout 2

A este layout está ligada la clase “ListadoEjercicios”, en la que se asocia para cada botón del layout una llamada a la actividad correspondiente y así lanzarla cuando el usuario pulse dicho botón. Hay 12 métodos en su interior, uno para cada botón del layout y son 11 botones para los distintos ejercicios y uno para volver atrás.

4.3.3 Layout 3

A la hora de importar un ejercicio, tras pulsar el botón que corresponde en el layout del menú inicial, se lanza la clase `ContinueExercise`, se carga el layout en el que existen dos opciones, volver al menú principal tras pulsar el botón o se podrá escribir un nombre de un archivo que se quiera importar. Una vez se haya escrito y se haya pulsado el botón correspondiente, se lanza el método “recuperar”, con el que se accede a la carpeta “Downloads” de la tarjeta SD para buscar el archivo con el nombre que ha escrito el usuario. Una vez que ha encontrado el archivo, se abre y se lee todo lo que tiene en su interior para posteriormente lanzar una actividad y pasar dicha información. La actividad que se lanza es “ContinuandoExercise”.

En la clase “ContinuandoExercise”, que está dentro de la actividad con el mismo nombre, se carga un layout semejante al layout principal de los ejercicios en el que aparece reflejada la FPGA. Pero este layout, pese a que tiene la misma interfaz gráfica que el principal, pierde ciertas funciones, ya que a la hora de importar un archivo no se puede saber a qué ejercicio corresponde. Las funciones que pierde son, poder realizar el “check it” del ejercicio y leer el enunciado del ejercicio. Todas las demás funciones sí que se realizan.

Inicialmente, al tratarse de un ejercicio ya resuelto, el archivo que se lee en “ContinueExercise” viene con información, dicha información se pasa a “ContinuandoExercise” y estos datos se tratan en esta última clase, en la que se rellenarán los checkbox, los togglebuttons de los CLB y se pintarán las líneas que indiquen esos datos del archivo importado. Ésta también es una diferencia con la clase que corresponde al layout principal donde está implementada la interfaz de la FPGA, ya que en la clase que carga dicho layout no se importan ejercicios resueltos, por lo que no hay información previa que cargar. Las demás funciones, comunes a ambas se detallarán cuando se explique el “Layout 5”.

4.3.4 Layout 4

Tras elegir cualquiera de los ejercicios en el “Layout 2” pulsando el botón del ejercicio que se desee, se lanza la actividad del ejercicio que corresponda. Todas las actividades de todos los ejercicios funcionan exactamente igual, se carga una imagen del enunciado del ejercicio y el usuario tiene dos opciones para interactuar con la aplicación, pulsar el botón con el que se irá de nuevo a “ListadoEjercicios” y el otro botón, el botón de continuar, con el que se lanzará la actividad principal de la aplicación, “NewExercise”. A esta actividad se le envían ciertos datos, como por ejemplo el parámetro que identifica que ejercicio ha elegido el usuario.

4.3.5 Layout 5

Este layout se carga tras lanzarse la clase “NewExercise”, la cual se podría decir que es la clase principal de la aplicación, la común para todos los ejercicios, la que replica el funcionamiento de la FPGA. Esta réplica de FPGA se ha realizado con numerosos checkbox que harán de puntos de interconexión de pistas, de entradas, de salidas, de puntos de VCC, de GND y de activación de CLK, tanto de la placa como de los CLK de los CLB.

Siempre que se vuelva a esta actividad desde otras, no en la primera llamada, pero si en las sucesivas, se hará una comprobación del valor de los diferentes checkbox y de los dibujos que ha realizado el usuario y se cargarán los valores en los checkbox, poniéndose activos si corresponde, y también se dibujarán las líneas si el usuario las ha dibujado (con el método “pintarlíneas”).

Además de esto, en esta clase se crean los métodos que se lanzan al pulsar los distintos botones que existen en este layout. Se pasa a describir los métodos en base a los botones del layout:

1. Métodos que lanzan las actividades de los distintos CLBs. Las actividades se definirán en el punto del “Layout 6”.
2. El método “enunciado” se lanza al pulsar sobre el botón de “Exercise”. Este método usa el parámetro que identifica el ejercicio previamente seleccionado para así lanzar la actividad del ejercicio que corresponde y que, de esta forma, el usuario pueda visualizar el enunciado del mismo.
3. El método “draw” se ejecuta al pulsar sobre el botón “Draw enabled”. Este método simplemente lanza la actividad “NewExercise1” en el que se permite dibujar sobre la pantalla.
4. Seguidamente se encuentran los botones “Save” y “Send”, que al pulsar sobre ellos se ejecutarán los métodos “guardar” y “enviar” respectivamente. Al ejecutarse estos métodos se lanzan las actividades en las que se guardarán y se enviarán los ejercicios, pasándose los datos necesarios para su posterior carga de datos.
5. Al pulsar sobre el botón “Check it”, se lanza el método “resultado”, con el que el usuario puede comprobar cómo se encuentra su ejercicio. Esta comprobación se hace en base al identificativo del ejercicio que se ha ido pasando entre actividades y así cada uno de los ejercicios tendrá una resolución diferente. Dará un mensaje indicando si se está haciendo correctamente o en caso contrario, no se está solventando como debería.

6. Por último está el botón “Memory”, con el que se ejecuta el método “memoria”, el que lanzará la actividad que mostrará los valores de los checkbox.

4.3.6 Layout 6

Al lanzarse las actividades tras pulsarse los botones de los CLBs, se carga el layout que simula un multiplexor con tres entradas y dos salidas, una de ellas controlada por el CLK y la otra no.

En este layout existen 8 togglebuttons, en los que el usuario tendrá que poner “1” o “0”, según lo crea conveniente para así resolver el ejercicio. Siempre que se lance una actividad de los CLBs inicialmente, estarán todos los togglebuttons a “0”, pero una vez se hayan modificado, siempre se guardará su estado, pudiéndose modificar cuando se desee.

En este layout también aparece un botón de “Ok”, con el que se volverá a lanzar la actividad “NewExercise”, enviándose toda la información de los togglebuttons de estas clases.

Hay un botón para cada uno de los 5 CLBs, y cada uno de ellos lanza una actividad diferente e individual.

4.3.7 Layout 7

El layout en el que el usuario puede realizar los dibujos, se lanza desde el botón “Draw enabled” del “Layout 5”. Este layout de aspecto es muy parecido al “Layout 5”, pero con funcionalidades diferentes.

Las funcionalidades que varían son:

- No se puede interactuar con ninguno de los botones anteriores, ni checkbox, ni CLBs.
- Se quitan todos los botones del otro layout y aparecen dos nuevos, “Clear all” y “Buttons Enabled”. Con el primer botón se borran todos los dibujos que haya realizado el usuario y con el segundo botón se volverá a la actividad “New Exercise”.
- El usuario podrá dibujar líneas y todo ello se hará en el método “onTouch”. Este método responderá ante las acciones del usuario cuando toca la pantalla, cuando presiona se localizan las coordenadas iniciales de la línea y cuando levanta el dedo localiza las coordenadas finales de la misma.

- Antes de lanzarse la actividad “NewExercise” al pulsar el botón “Buttons Enabled”, se guardará en un archivo las coordenadas de cada una de las líneas que se hayan dibujado. El método que ejecuta estas acciones es el método “enable”.

4.3.8 Layout 8

Estos layouts, el de “Save” y el de “Send” son iguales en apariencia, salvo la frase que indica la función que se realizará en la interfaz. En ambos se tiene que escribir un nombre de un archivo.

En la clase “EnviarEjercicio”, la clase que carga el layout de “Send”, se deberá escribir el nombre del archivo que se enviará como adjunto en el correo, este archivo ha tenido que ser guardado previamente. En este layout también aparecen tres botones, el botón “Send” con el que se confirmará el nombre del archivo y se lanzará la actividad “SendEmailActivity” en la que se escribirán los datos para enviar el correo. Los otros dos botones son “Back”, con el que se volverá a la actividad anterior y el botón “Main Menu”, con el que se lanzará la actividad inicial de la aplicación.

Y en la clase “GuardarEjercicio”, que es la clase que carga el layout de “Save”, se escribirá el nombre con el que se quiere guardar el archivo que contiene los datos de las posiciones de memoria y el contenido de los CLBs. Una vez que se escribe el nombre y se pulsa el botón “Save” (uno de los tres botones del layout), se guardará el archivo usando el nombre que ha escrito el usuario. Se guarda el archivo en la tarjeta SD. Los otros dos botones son “Back”, con el que se volverá a la actividad anterior y el botón “Main Menu”, con el que se lanzará la actividad inicial de la aplicación.

4.3.9 Layout 9

Este layout tiene un solo botón y al pulsarlo se volverá a la actividad “NewExercise”. Este layout es cargado por la clase “MemoriaMostrada”, en la que se mostrarán todos los valores (“0” o “1”) de la interconexión de pistas, que realmente son los checkbox. También se muestran los valores del contenido de los distintos CLBs, separando para cada CLB su propia información.

4.3.10 Layout 10

El layout en el que el usuario escribe los datos del correo electrónico se carga en la clase "SendEmailActivity". En este layout se escribirá el destinatario del correo, el asunto, un mensaje de texto, también se podrá elegir adjuntar o no los archivos de los ejercicios y por último aparecen tres botones.

Los archivos de los ejercicios que se adjuntan son el archivo en el que están las coordenadas de las líneas dibujadas y el otro es el archivo en el que están los valores de los checkbox y el contenido de los CLBs

Al pulsar el botón "Send" se cargará toda esa información en un cliente de correo que se seleccione. Los otros dos botones son "Back", con el que se volverá a la actividad anterior y el botón "Main Menu", con el que se lanzará la actividad inicial de la aplicación.

4.4 Desarrollos futuros

Mientras se ha ido realizando el proyecto, han ido surgiendo posibles mejoras, algunas de ellas se pudieron implementar y otras, por el contrario, al ser más complejas y por no disponer del tiempo necesario para ello, no se pudieron desarrollar.

Estas mejoras son:

- Uso de la base de datos SQLite para que en lugar de pasar toda la información entre actividades cada vez que se lance una nueva actividad, se pudiera acceder a dicha base de datos a consultar estos datos.
- Realizar de una forma más dinámica la carga de ejercicios en la aplicación, de forma que si el profesor de la asignatura desea proponer ejercicios nuevos, no haya que implementar uno a uno con código.
- En lugar de dibujar líneas para definir mejor el resultado del ejercicio, se había pensado resaltar en color las líneas entre los puntos de interconexión que se marquen, ya sea al activar los checkbox o al pulsar sobre esas líneas.
- La comprobación del resultado de los ejercicios se realiza de una manera muy básica, ya que puede haber infinidad de opciones para resolver correctamente cualquiera de los ejercicios, por lo que si se pudiera concretar más los resultados, se optimizaría mucho más la aplicación.
- Los correos con los archivos adjuntos solo se envían si se usa el cliente de correos electrónicos propio de Android, ya que por ejemplo, el Gmail no adjunta ambos archivos. Una mejora podría ser averiguar porque esto no funciona correctamente y resolver el problema.
- Otra posible mejora sería incorporar teoría en la aplicación, añadiendo nuevas opciones para acceder a ella, ya sea mediante enlaces a páginas web o incorporando dicha teoría a la aplicación, lo que no sería muy complicado y en cambio sería muy práctico.
- Una vez se ha importado un archivo, no se puede ni realizar la comprobación del estado del ejercicio, ni visualizar el enunciado correspondiente, por lo que otra mejora podría ser guardar un identificador en el archivo, y al ser leído al importarlo poder realizar estas dos acciones, visualizar el enunciado y realizar la comprobación, si el usuario pulsa los botones que corresponden.
- Y por último, se podría mejorar la aplicación en cuanto al uso de recursos que hace la misma.

5 MANUAL DE USUARIO

5.1 Introducción

En este apartado se pasa a detallar todo lo que el usuario debe realizar para que la aplicación funcione correctamente. Este manual de usuario es para el alumno, ya que más adelante se encontrará el manual del profesor, puesto que ambos usuarios no trabajarán con la aplicación del mismo modo.

5.1.1 Paso 1

Antes de nada, el usuario debe estar en posesión de la aplicación y para ello tendrá que tener el archivo con extensión “.apk”, es decir, “MainActivity.apk”. Para tener este archivo, normalmente se descargará del Play Store, pero dado que esta aplicación es una primera versión y necesita ser optimizada en modo funcional, se ha decidido no subirla aún al Play Store. Así que el profesor de la asignatura deberá enviar o pasar el archivo a los alumnos para poder instalarlo.

5.1.2 Paso 2

Una vez se está en posesión del archivo “MainActivity.apk”, el alumno simplemente tiene que ejecutarlo en el dispositivo y éste instalará la aplicación automáticamente, creando así el icono correspondiente para dar acceso a la aplicación al usuario.

5.1.3 Paso 3

Cuando la aplicación ha sido instalada ya se puede comenzar a interactuar con ella. Lo primero que hay que hacer es entrar a ella presionando el icono de la app.



Figura 5-1: Icono app

5.1.4 Paso 4: Menú inicial

Una vez el usuario está dentro de la aplicación se encuentra con el menú principal, en el que podrá elegir entre “Exercise’s List”, “Import Exercise” y “Exit”. Si elige la primera opción, a continuación observará la lista con todos los ejercicios con los que consta la app. Si elige la segunda opción, podrá importar un archivo que previamente haya guardado o descargado en su dispositivo. Y con la última opción, se saldrá de la app tras presionar el botón.

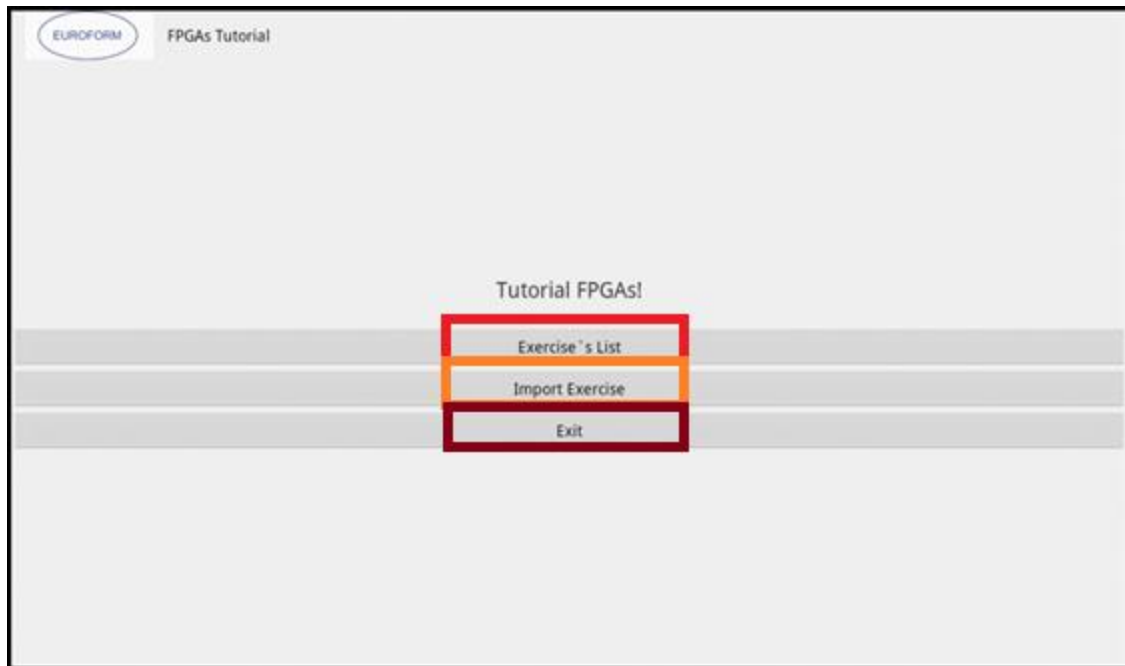


Figura 5-2: Menú principal

5.1.5 Paso 5

En este punto el usuario habrá podido presionar tres botones, el de “Exit” sale de la aplicación por lo que no tiene mayor importancia, en cambio las otras dos opciones sí que tienen que seguir siendo explicadas:

- a. “Exercise’s List”: El usuario ante el menú con el que se encuentra en este momento podrá elegir uno de los 11 ejercicios que se muestran. De los 11 ejercicios, los 10 primeros tienen enunciado en el que se explica lo que el usuario debe realizar, pero el último, “Blank Exercise”, es un ejercicio “comodín”, y será el que el usuario deba usar para resolver los ejercicios que el profesor mande y no estén desarrollados en la aplicación.



Figura 5-3: Listado ejercicios

- b. “Import Exercise”: Si el usuario elige esta opción verá el siguiente layout:

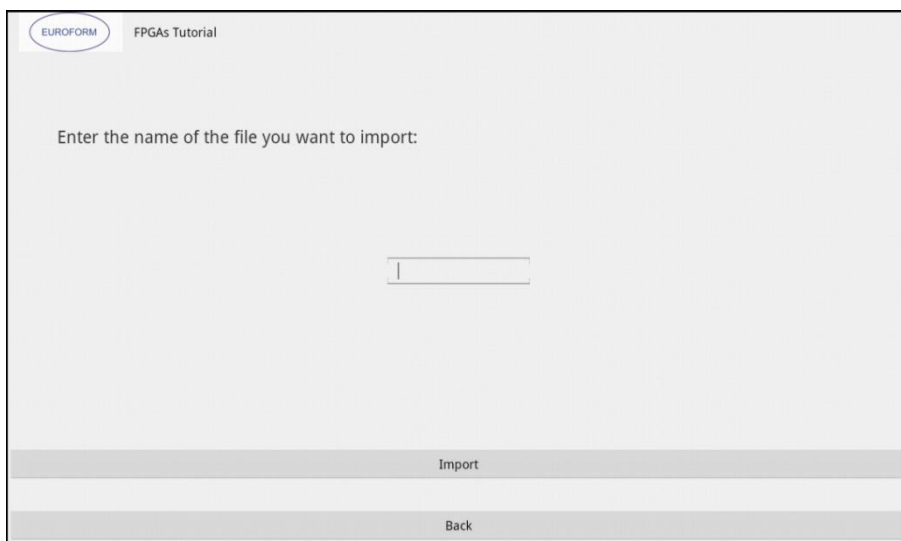


Figura 5-4: Importación ejercicio

Se encontrará ante un layout en el que deberá escribir un nombre de un ejercicio que previamente se haya guardado o descargado y esté en la Tablet. Si no está, la aplicación mostrará un mensaje indicando este problema, y si está, al pulsar sobre el botón “Import”, la aplicación cargará en el layout principal el ejercicio importado. A continuación se detallan los pasos que deberá seguir el usuario si ha elegido importar un archivo guardado:

- i. Si el usuario ya ha pulsado el botón “Import”, se cargará el archivo y se mostrará en el layout principal de la aplicación todos los checkbox, todo el contenido de los CLBs (al entrar en ellos) y los dibujos tal y como lo haya resuelto el usuario previamente a guardarlo/descargarlo.
- ii. Todas las funciones que pueda tener la aplicación en este punto respecto a lo que tendría si no hubiera importado el archivo serán exactamente las mismas, salvo que no podrá consultar el enunciado del ejercicio, ni podrá comprobar el resultado del mismo. Podrá modificar los dibujos, podrá volverlo a enviar, podrá volverlo a guardar y podrá visualizar el contenido de la memoria.

**** Importante: Cada vez que se importe un archivo, hay que asegurarse de que este está en la carpeta “Downloads” de la tarjeta SD del dispositivo, y que sólo existe un archivo “graficos.txt” y éste es el último. Esto es así porque todos los archivos que contienen las coordenadas de los dibujos se llaman igual.***

5.1.6 Paso 6: Enunciado

Si el usuario elige uno de los 10 primeros ejercicios, verá el enunciado del ejercicio que haya seleccionado. En este layout podrá volver atrás para escoger otro ejercicio, o continuar hacia el layout en el que comenzará a resolver dicho ejercicio.

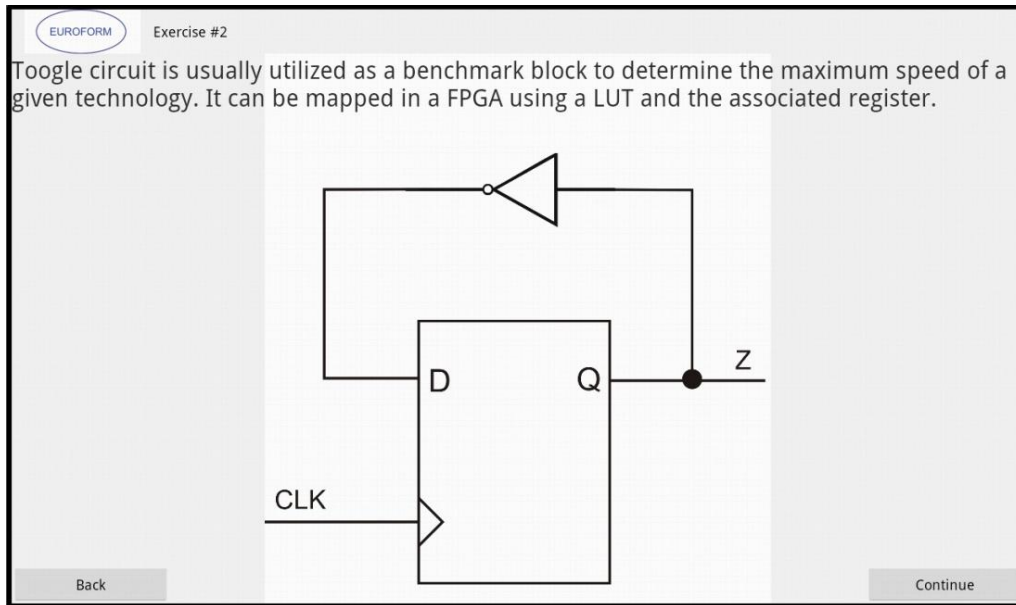


Figura 5-5: Ejemplo enunciado

5.1.7 Paso 7: Layout principal

En este punto el usuario se encuentra en el layout principal de la aplicación, en el que deberá resolver el ejercicio que haya seleccionado.

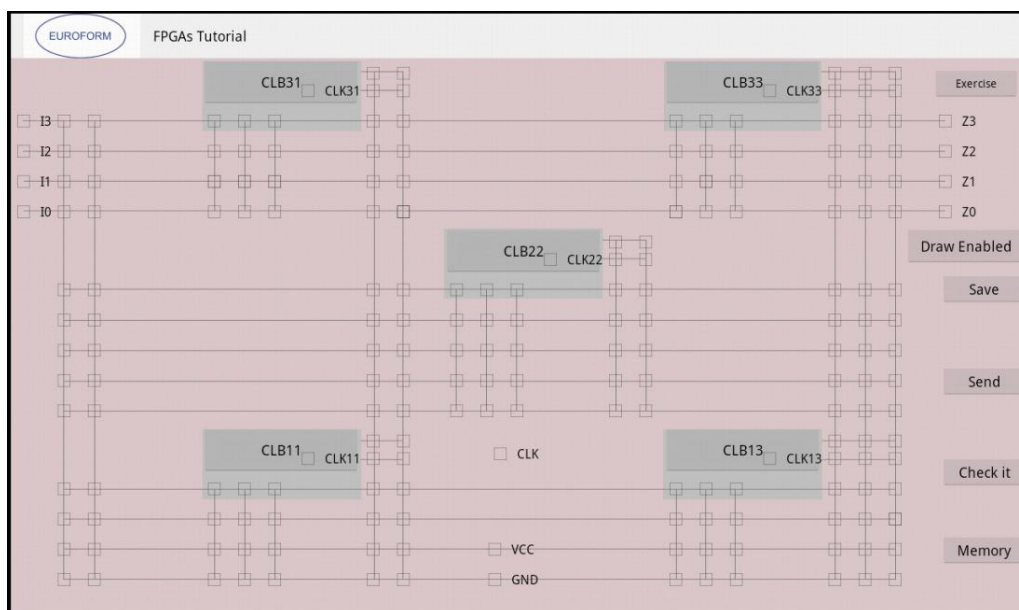


Figura 5-6: Layout principal

Para resolver los ejercicios tendrá que basarse en sus conocimientos de la asignatura y así podrá plasmar en la interfaz de la aplicación la resolución del ejercicio. Para poder plasmarlo, en pantalla se encuentran varios tipos de elementos:

- Los **checkbox**, con los que al marcarlos, el usuario dará a entender que, selecciona esas entradas (I1, I2, I3, I4), las salidas (Z1, Z2, Z3, Z4), CLK, VCC, GND o activa el CLK del CLB seleccionado con el checkbox. Por otra parte, los checkbox que no tienen nombre a su lado, funcionarán como puntos de interconexión de pistas, con los que el usuario deberá realizar la supuesta unión de pistas para completar correctamente el ejercicio. Se muestra una imagen a modo de ejemplo.

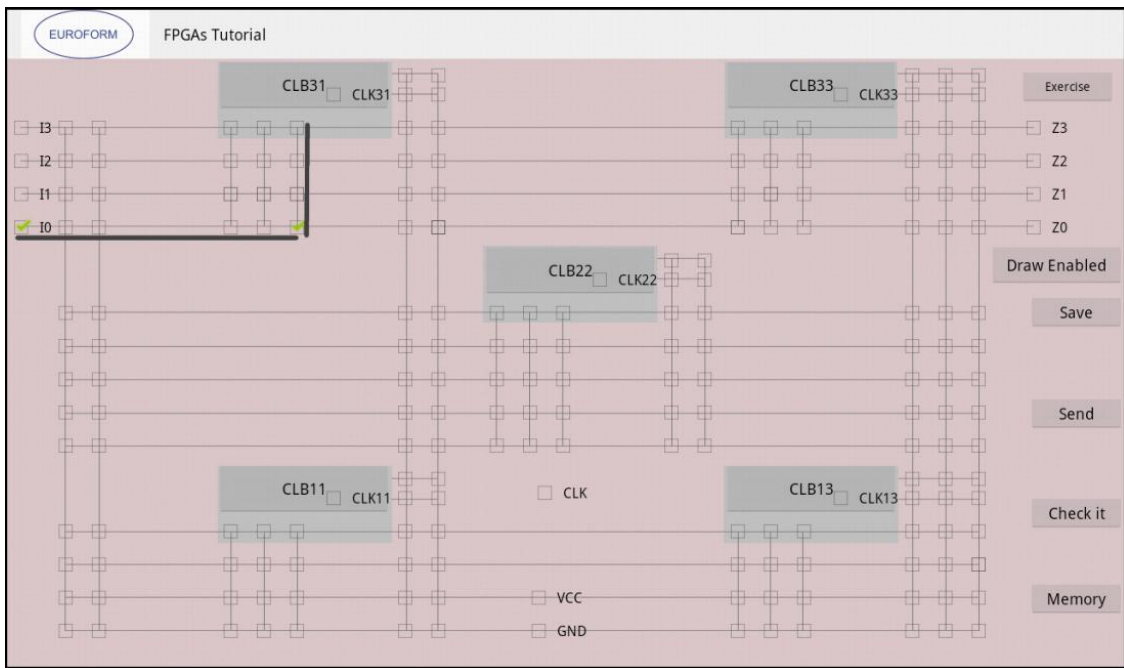


Figura 5-7: Ejemplo interconexión

Se selecciona la entrada I0 y si se desea que esa entrada pase a ser la entrada S0 del multiplexor del CLB31, se debe marcar el checkbox que está pulsado en la imagen. Se han dibujado dos líneas para que sea más fácil visualizar las conexiones realizadas y el camino que seguirían dichas conexiones.

- Los **botones CLB** en el layout simulan un chip, pero que al pulsarlos dan acceso a otro layout en el que se deberá completar con el fin de resolver los ejercicios. En esta pantalla que se accede tras pulsar cualquiera de los 5 botones de los CLB, aparecen 8 togglebuttons, con valor "1" o "0", que habrá que presionar en base al ejercicio que se esté resolviendo. Cuando se haya completado para volver a la pantalla principal se presionará el botón "Ok" y así se guardará la configuración del CLB.

**Si en un ejercicio no se entra en los CLB, se tomarán los valores de su interior como "0".*

**Toda la información que se configure en este layout estará siempre disponible y se cargará debidamente, salvo si el usuario sale, o al menú inicial, o al menú con el listado de los ejercicios.*

- A continuación se nombran ciertos botones que dan acceso a otras funciones de la aplicación que se explicarán en puntos sucesivos:
 - Exercise
 - Draw Enabled
 - Save
 - Send
 - Check it
 - Memory

Se seguirán explicando las distintas opciones que aparecen en el layout anterior en el orden en el que se muestran.

5.1.8 Paso 8: Botón "Exercise"

Si el usuario pulsa este botón, se ha configurado para que vuelva al enunciado del ejercicio que eligió para desarrollar, así podrá volver a leer el enunciado y aclarar cómo realizar el ejercicio.

En esa pantalla se vuelven a tener dos opciones, continuar con el ejercicio o volver atrás, si se volviera atrás al listado donde están los ejercicios, se perdería toda la información del ejercicio que se estaba completando.

5.1.9 Paso 9: Botón "Draw Enabled"

Al pulsar el botón "Draw Enabled" se accede a otra pantalla de la aplicación, en la que el usuario podrá dibujar líneas para así simular las pistas usadas para resolver el ejercicio.

Para pintar las líneas simplemente hay que presionar la pantalla en la zona en la que se quiere comenzar a pintar y se presionará hasta el punto en el que se quiere que finalice la línea. Cuando se quiere que se pare de dibujar la línea habrá que levantar el dedo. La línea no se verá dibujada en pantalla hasta que se levante el dedo de la misma.

Las líneas serán líneas rectas, y éstas cogerán las coordenadas iniciales donde se presionó la pantalla y cogerán las coordenadas finales donde se levante el dedo, uniendo estos dos puntos y así se creará la línea recta que se dibujará. Aunque se hagan caminos que no sean líneas rectas en pantalla, no se mostrarán, ya que como se ha explicado sólo reconoce el punto inicial donde se presionó la pantalla y el punto final donde se levantó el dedo.

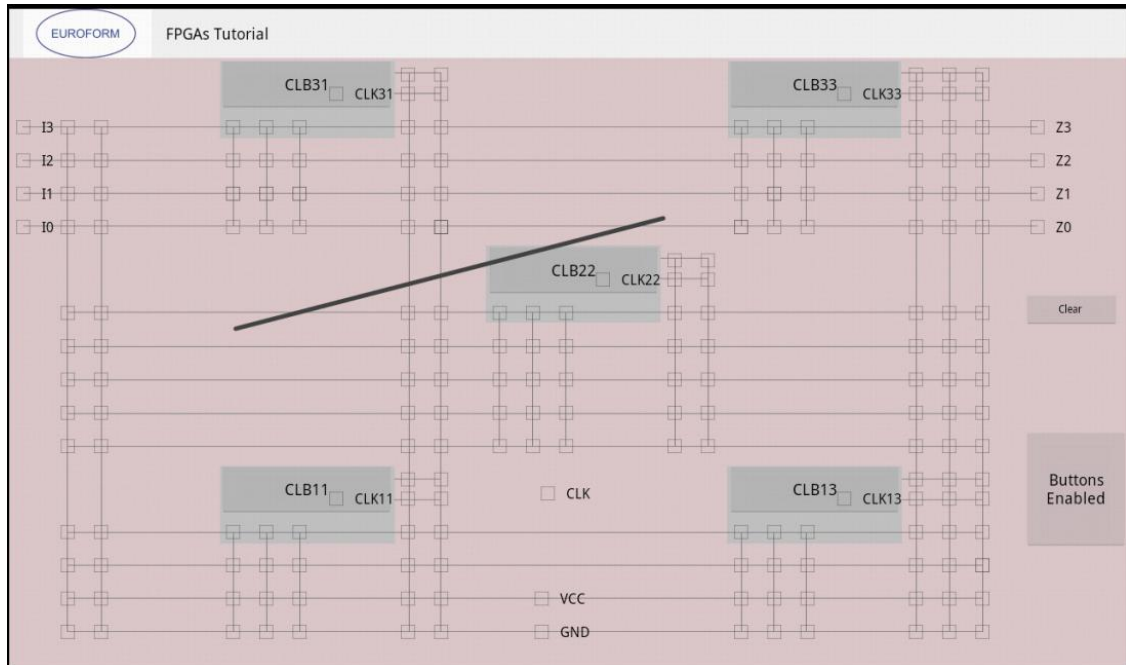


Figura 5-8: Dibujando líneas

5.1.10 Paso 10: Botón "Save"

Al pulsar este botón el usuario accederá a una pantalla en la que podrá guardar el ejercicio. Para guardar el ejercicio simplemente deberá escribir el nombre usando el teclado virtual que aparece en pantalla. Tras tenerlo escrito habrá que pulsar el botón "Save" de esa pantalla y se procederá a guardar el archivo correspondiente a ese ejercicio en la tarjeta SD del dispositivo.

En esta pantalla también aparecen dos botones más, un botón con el que se volverá a la pantalla en la que se resuelve el ejercicio y otro botón con el que se accede al menú principal, pero con este acceso al menú principal se pierden los datos del ejercicio en pantalla, no así el archivo, si previamente ha sido guardado.



Figura 5-9: Guardar ejercicio

5.1.11 Paso 11: Botón "Send"

Al pulsar este botón, se accede a una pantalla similar a la que se accede al pulsar el botón "Save", pero la función en este caso es diferente, ya que tras escribir el nombre del ejercicio y al pulsar el botón de "Send", se accede a otra pantalla, similar a un cliente de correo electrónico.

El archivo que se desee enviar debe estar previamente guardado en el dispositivo, para ello hay que realizar el **"Paso 10: Botón "Save"**.

En esta pantalla también aparecen dos botones más, un botón con el que se volverá a la pantalla en la que se resuelve el ejercicio y otro botón con el que se accede al menú principal, pero con este acceso al menú principal se pierden los datos del ejercicio en pantalla.

Al pulsar el botón “Send” de la pantalla en la que se escribe el nombre del archivo a enviar, se accede a otra pantalla, similar a un cliente de correo electrónico.



Figura 5-10: Datos correo

En esta pantalla habrá que escribir varios datos:

- i. Destinatario del correo
- ii. Asunto del correo
- iii. Mensaje de texto

Una vez se haya escrito toda esta información, se marcará el **checkbox** “Send attachment” con el fin de adjuntar los dos archivos correspondiente al ejercicio que se está resolviendo. Estos dos archivos son, el que contiene la información del contenido de los CLBs y de los checkbox, que tendrá el nombre que el usuario haya escrito en la pantalla de “Save” y el otro archivo será el que contenga la información de las coordenadas de las líneas dibujadas por el usuario. Este último archivo se llamará siempre “graficos.txt”.

Tras tener completa toda la información, se pulsará el botón “Send” de esta pantalla y se dará a elegir entre varios clientes de correos electrónicos. Una vez se elija uno, se le pasarán todos los datos que se han completado y se rellenarán donde corresponda.

**** El cliente de correo electrónico que ha de elegirse para que funcione correctamente, tiene que ser el del sistema operativo de Android, ya que con otros, los archivos a enviar no se adjuntan correctamente.***

Paso 12: Botón “Check it”

Pulsando este botón se realiza la comprobación del estado del ejercicio que se esté resolviendo. Para cada ejercicio habrá una comprobación individual, ya que cada ejercicio se desarrolla por separado.

El chequeo del ejercicio es en base a los checkbox de las entradas (I0, I1, I2 e I3), a los checkbox de las salidas (Z0, Z1, Z2 y Z3), a los de VCC, GND, CLK, CLK31, CLK33, CLK22, CLK11 y CLK13. No se puede realizar la comprobación del contenido de los CLB ni habrá una única solución, ya que el usuario podrá resolver los ejercicios de múltiples maneras.

Cuando se pulse el botón, saldrá un mensaje indicando si es o no correcto, en base al valor de los checkbox que se han indicado, pero con este mensaje no quiere decir que esté implementado correcto totalmente.

Paso 13: Botón “Memory”

Tras pulsar este último botón se accede a una pantalla en la que se muestra la información del contenido de los CLB, en los que hay 8 dígitos, que podrán tener como valor “1” o “0”. En la parte media de la pantalla se muestran las posiciones de memoria de la FPGA simulada. También podrán tener como valor “1” o “0”.

Al pulsar el botón Back, se volverá a la pantalla anterior y se cargará toda la información que había previamente.

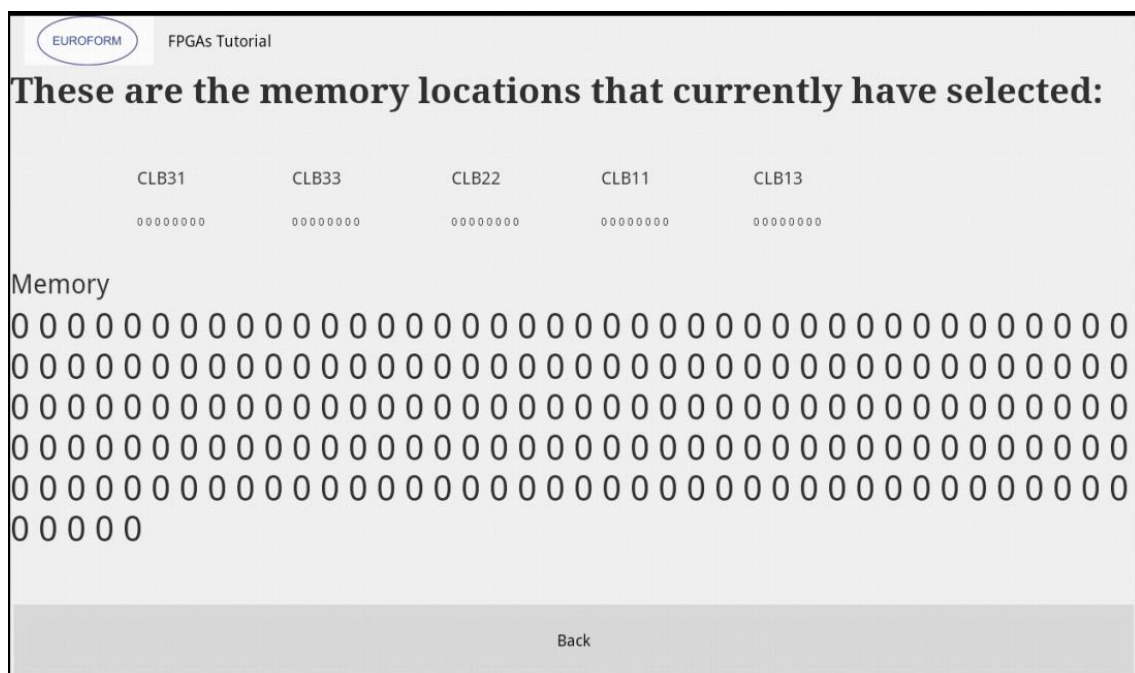


Figura 5-11: Memoria

6 MANUAL DEL PROFESOR

6.1 Introducción

En este apartado se describirán los pasos a seguir por el profesor para que la aplicación funcione correctamente.

6.1.1 Paso 1

Antes de nada, el usuario debe estar en posesión de la aplicación y para ello tendrá que tener el archivo con extensión “.apk”, es decir, “MainActivity.apk”. Para tener este archivo, normalmente se descargará del Play Store, pero dado que esta aplicación es una primera versión y necesita ser optimizada en modo funcional, se ha decidido no subirla aún al Play Store. En este caso, al ser el profesor el destinatario del manual, será el mismo el que posea el archivo.

6.1.2 Paso 2

Una vez se está en posesión del archivo “MainActivity.apk”, el alumno simplemente tiene que ejecutarlo en el dispositivo y éste instalará la aplicación automáticamente, creando así el icono correspondiente para dar acceso a la aplicación al usuario.

6.1.3 Paso 3

Cuando la aplicación ha sido instalada ya se puede comenzar a interactuar con ella. Lo primero que hay que hacer es entrar a ella presionando el icono de la app.



Figura 6-1: Icono app

6.1.4 Paso 4

Una vez el usuario está dentro de la aplicación se encuentra con el menú principal, en el que podrá elegir entre “Exercise’s List”, “Import Exercise” y “Exit”. Si elige la primera opción, a continuación observará la lista con todos los ejercicios con los que consta la app. Si elige la segunda opción, podrá importar un archivo que previamente haya guardado o descargado en su dispositivo. Y con la última opción, se saldrá de la app tras presionar el botón.



Figura 6-2: Menú principal

6.1.5 Paso 5

En este punto el usuario habrá podido presionar tres botones, el de “Exit” sale de la aplicación por lo que no tiene mayor importancia, en cambio las otras dos opciones sí que tienen que seguir siendo explicadas:

- c. “Exercise’s List”: Esta opción el profesor no la tendrá que usar para la resolución de los ejercicios de los alumnos.
- d. “Import Exercise”: Si el profesor elige esta opción, se encontrará ante un layout en el que deberá escribir un nombre de un ejercicio que previamente se haya descargado del correo electrónico. Si no está o no ha escrito debidamente el nombre, la aplicación mostrará un mensaje indicando este problema, y si está, al pulsar sobre el botón “Import”, la aplicación cargará en el layout principal el ejercicio importado. El ejercicio que se ha descargado el profesor será de un alumno que le ha enviado previamente al correo electrónico y estará ubicado en la carpeta

“Downloads” de la tarjeta SD. De no ser así, la aplicación no cargará el ejercicio importado.

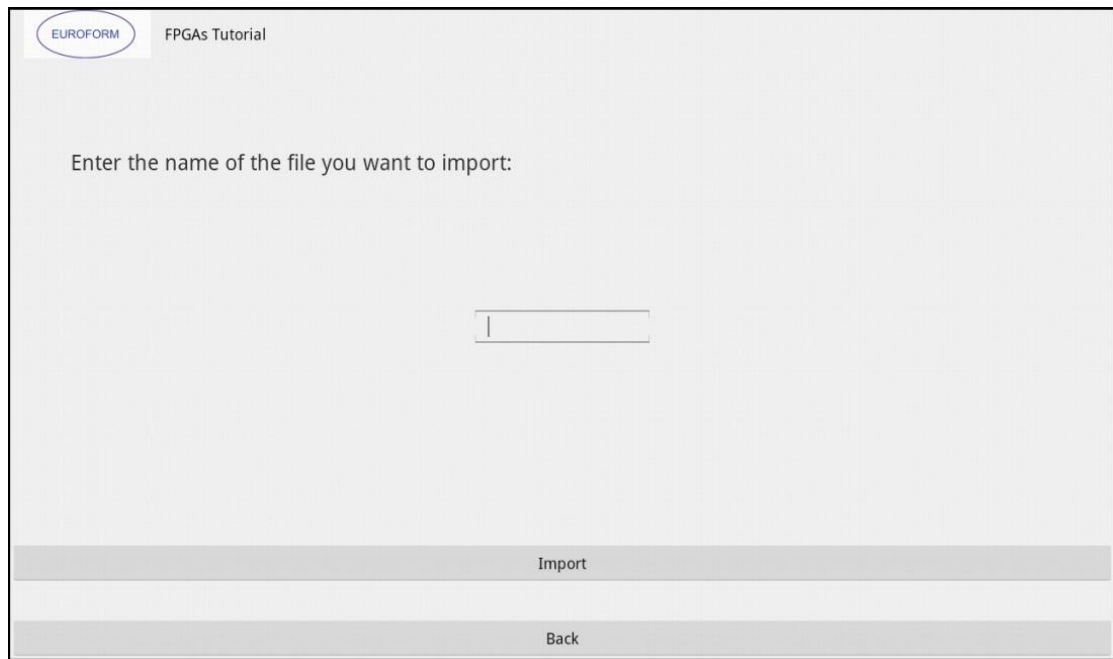


Figura 6-3: Importar ejercicio

6.1.6 Paso 6

Si el usuario ya ha pulsado el botón “Import”, se cargará el archivo y se mostrará en el layout principal de la aplicación todos los checkbox, todo el contenido de los CLBs (al entrar en ellos) y los dibujos tal y como lo haya resuelto el alumno previamente a enviarlo al profesor.

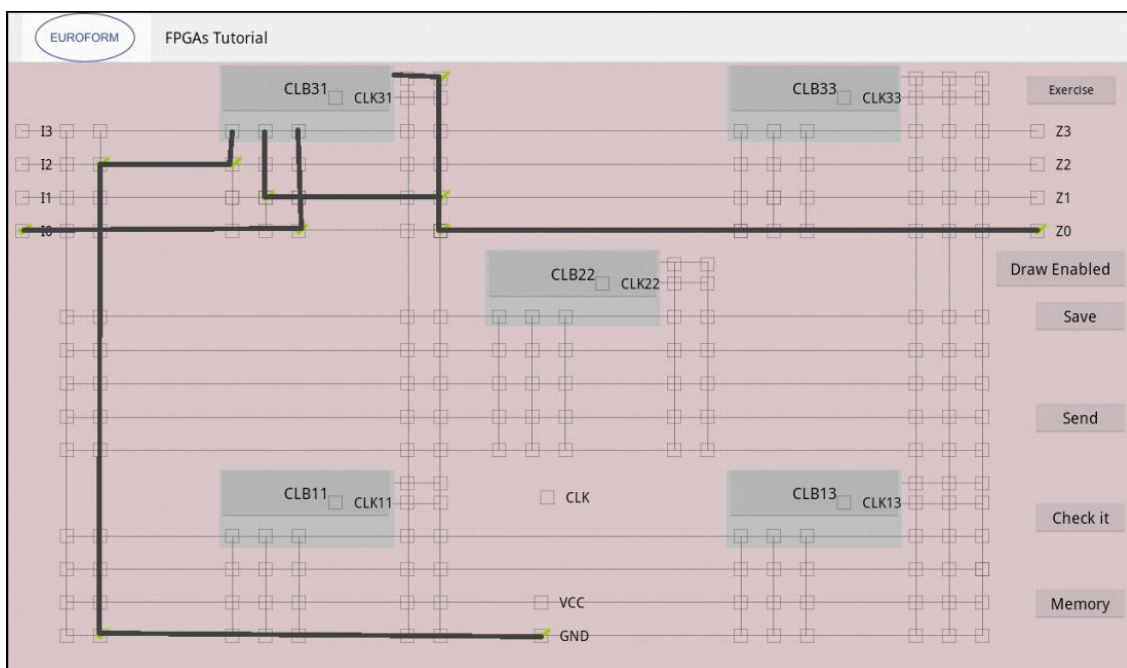


Figura 6-4: Ejercicio cargado

**** Importante: Cada vez que se importe un archivo, hay que asegurarse de que este está en la carpeta "Downloads" de la tarjeta SD del dispositivo, y que sólo existe un archivo "graficos.txt" y éste es el último. Esto es así porque todos los archivos que contienen las coordenadas de los dibujos se llaman igual.***

A continuación se nombran ciertos botones que dan acceso a otras funciones de la aplicación que se explicarán en puntos sucesivos:

- Exercise
- Draw Enabled
- Save
- Send
- Check it
- Memory

Se seguirán explicando las distintas opciones que aparecen en el layout anterior en el orden en el que se muestran.

1. Botón "Exercise"

El profesor no hará uso de esta función.

2. Botón "Draw Enabled"

El profesor no hará uso de esta función.

3. Botón "Save"

El profesor no hará uso de esta función.

4. Botón "Send"

El profesor no hará uso de esta función.

5. Botón "Check it"

El profesor no hará uso de esta función.

6.1.7 Paso 7: Botón “Memory”

Tras pulsar este último botón se accede a una pantalla en la que se muestra la información del contenido de los CLB, en los que hay 8 dígitos, que podrán tener como valor “1” o “0”. En la parte media de la pantalla se muestran las posiciones de memoria de la FPGA simulada. También podrán tener como valor “1” o “0”.

Al pulsar el botón Back, se volverá a la pantalla anterior y se cargará toda la información que había previamente.

Esta función la podrá usar el profesor para corregir el ejercicio visualizando directamente el valor de las “posiciones de memoria” y de los CLBs. Para facilitar la corrección debería disponer de una plantilla en la que se muestre la solución para cada ejercicio.

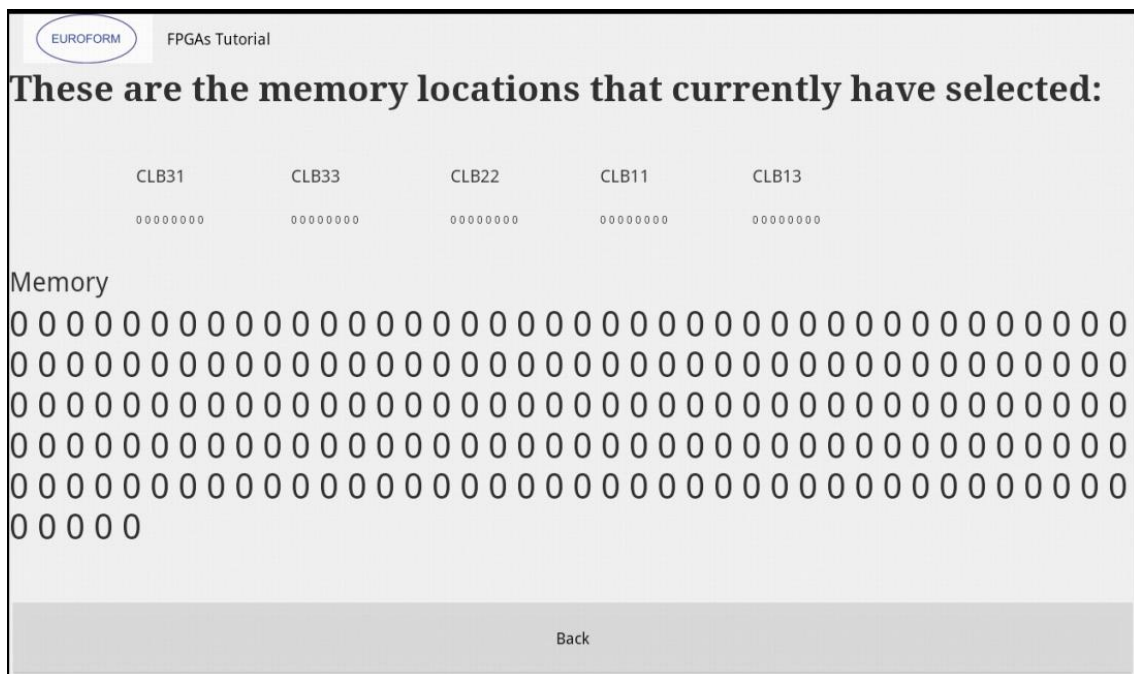


Figura 6-5: Memoria

*** Una vez se han detallado todas las opciones que existen en la aplicación para el profesor, éste ya podrá evaluar los ejercicios de los alumnos, basándose en toda la información que ha visualizado en la aplicación.**

7 PRUEBAS Y EJEMPLOS

7.1 Introducción

En este apartado se tratará en detalle cada ejercicio de la aplicación, haciendo especial interés en las especificaciones necesarias para solventar cada uno de ellos.

Para todos los ejercicios se ha usado el mismo layout, ya que se ha decidido que era mejor que existiera uno común y así el alumno tenga que pensar cual es la mejor forma de resolver cada ejercicio en este layout único, usando el número de interconexiones, entradas, salidas y CLBs necesarios según su criterio.

El esquema a seguir para exponer cada ejercicio en esta memoria será de la siguiente forma:

- Número del ejercicio
- Enunciado del ejercicio
- Explicación del ejercicio
- Imagen de la resolución del ejercicio en el layout
- Datos de interés propios del ejercicio (si los tuviera)

7.2 Ejercicio #1

I. Número del ejercicio:

#1

II. Enunciado del ejercicio:

“Map a 2-input AND gate”

III. Imagen de la resolución del ejercicio:

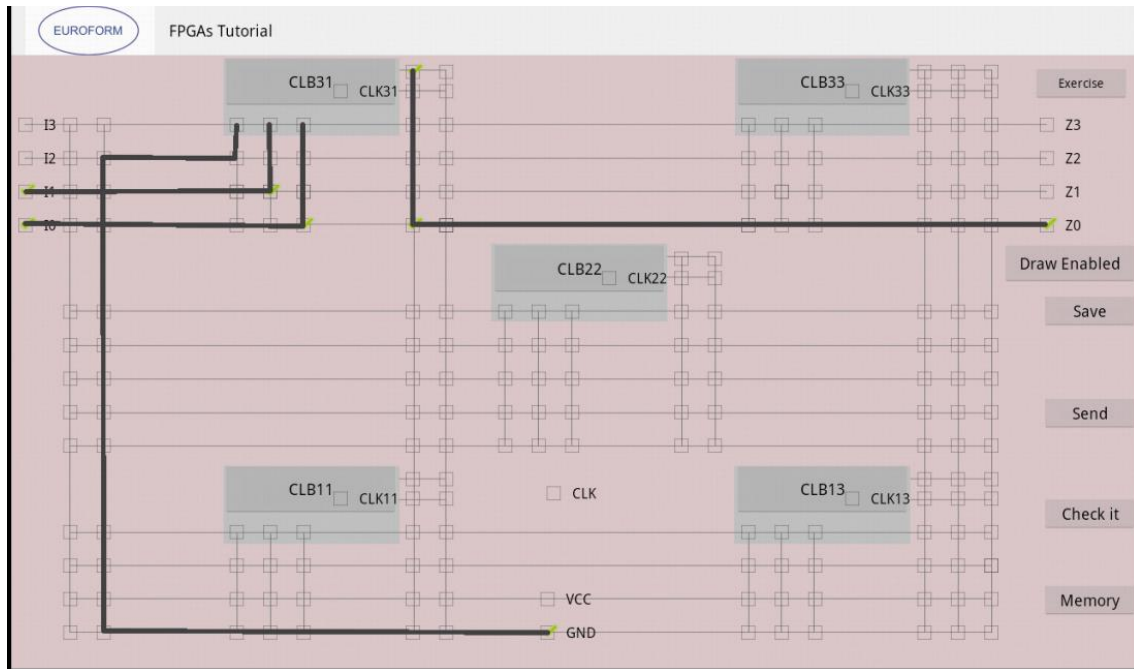


Figura 7-1: Layout principal

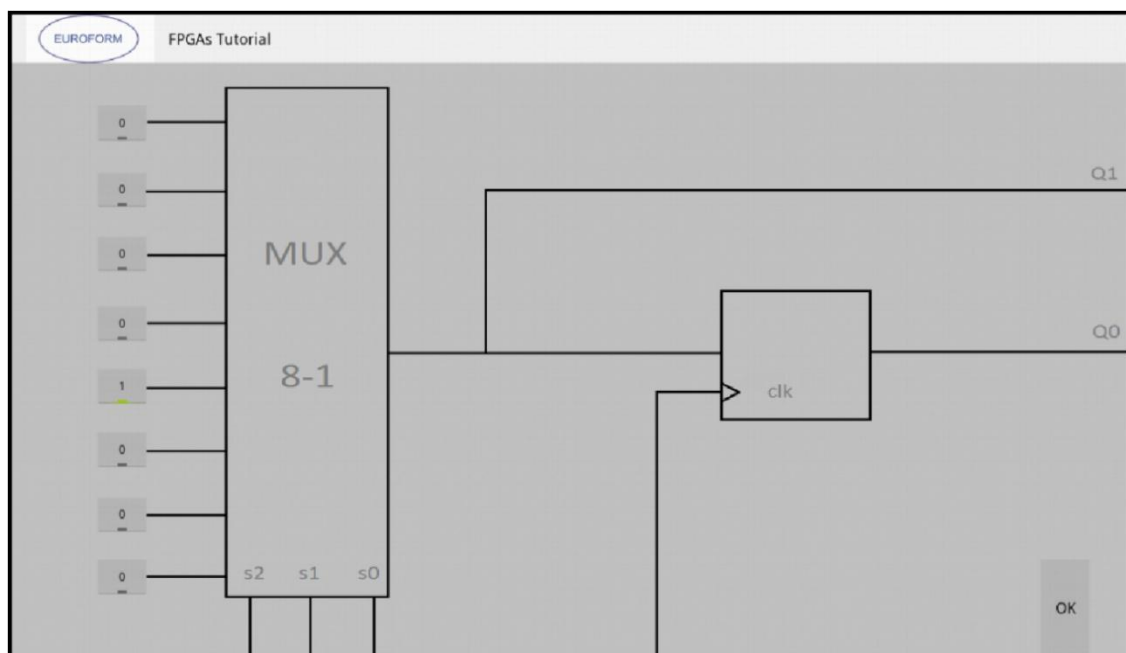


Figura 7-2: CLB31

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear una puerta lógica tipo AND de dos entradas, usando el layout genérico.

Para resolverlo se escogen las entradas I0 y I1 y se interconexionan con las entradas S1 y S0 del CLB31 (que a su vez son las del Mux 8-1). La salida elegida es la Z0, y a ésta llega la salida del CLB que no está controlada por el CLK. A la entrada S2 del CLB31 se le conecta a GND, ya que no se usan las 4 primeras entradas del MUX.

En la segunda imagen se observa el interior del CLB31, y ahí simplemente hay que crear la puerta AND, teniendo en cuenta que las 4 primeras entradas del MUX 8-1 no se usan, por lo que la entrada al MUX es "1000".

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I1, I0 activos
- GND activo
- Z0 activo

7.3 Ejercicio #2

I. Número del ejercicio:

#2

II. Enunciado del ejercicio:

“Toggle circuit is usually utilized as a benchmark block to determine the maximum speed of a given technology. It can be mapped in a FPGA using a LUT and the associated register.”

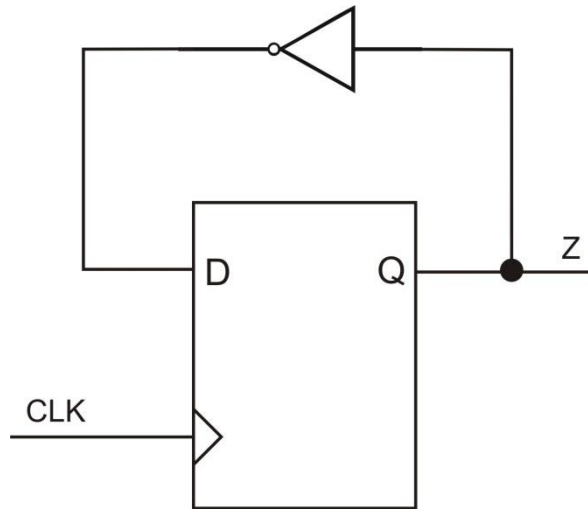


Figura 7-3: Enunciado

III. Imagen de la resolución del ejercicio:

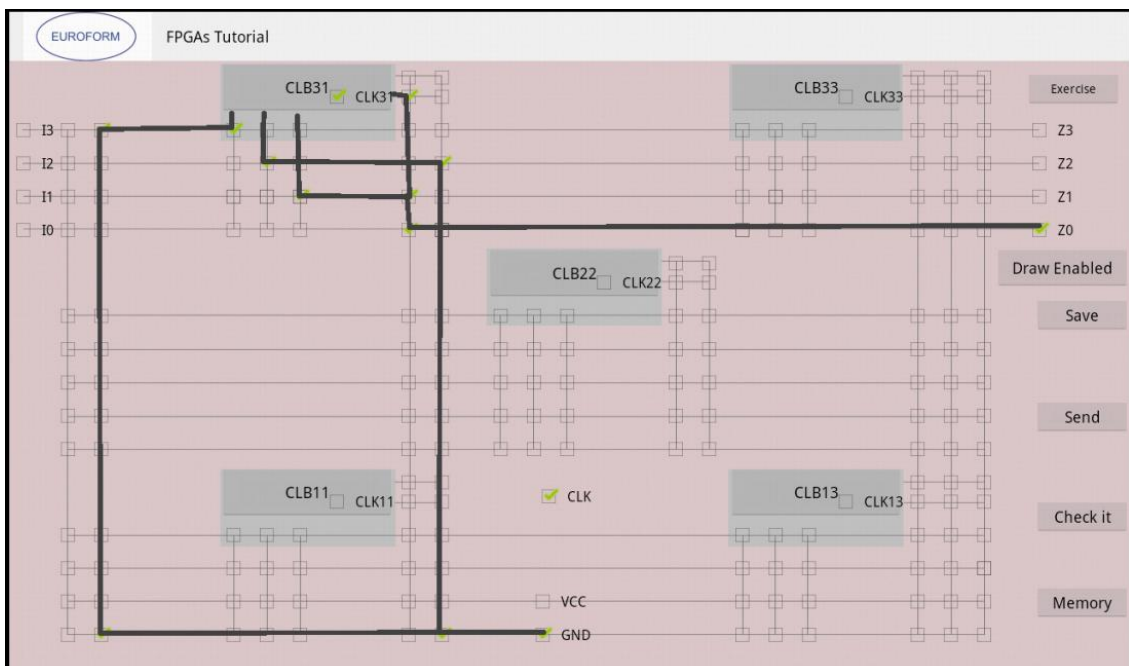


Figura 7-4: Layout principal

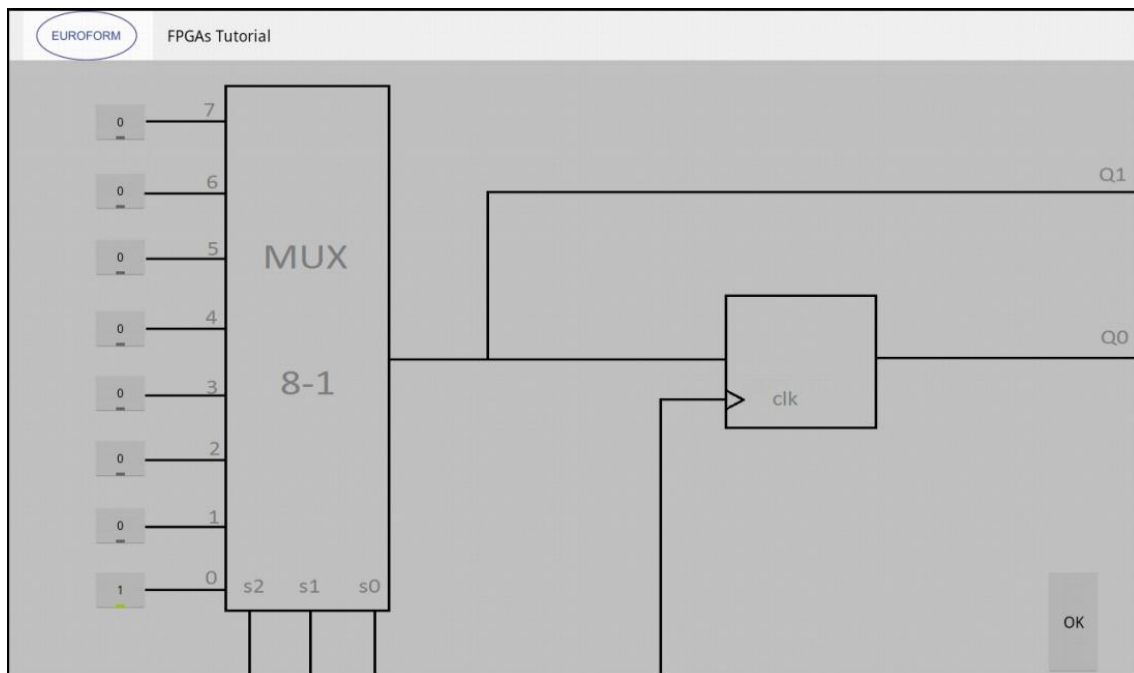


Figura 7-5: CLB31

IV. Explicación del ejercicio:

En este ejercicio se ha de realizar un elemento Toggle mediante la interconexión de pistas de la FPGA.

Para resolverlo en este caso no tiene que usarse ninguna de las entradas disponibles en la FPGA, ya que la entrada al CLB31 ha de conectarse a la salida del mismo y a su vez a la salida Z0. En este caso hay que marcar tanto el CLK del chip como el CLK31, activando así el Flip Flop del CLB. A las entradas S1 y S2 del CLB31 se les conecta a GND, ya que no se usan las 6 primeras entradas del MUX.

En la segunda imagen se observa el interior del CLB31, en el que hay que dar valor a las entradas del MUX. Teniendo en cuenta que las 6 primeras entradas no se usan, la entrada es "01".

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- GND activo
- Z0 activo
- CLK y CLK31 activos

7.4 Ejercicio #3

I. Número del ejercicio:

#3

II. Enunciado del ejercicio:

“Map the following circuit based on a 2-input XOR. What is the value of the output Z if the signal I0 pass from 0 to 1.”

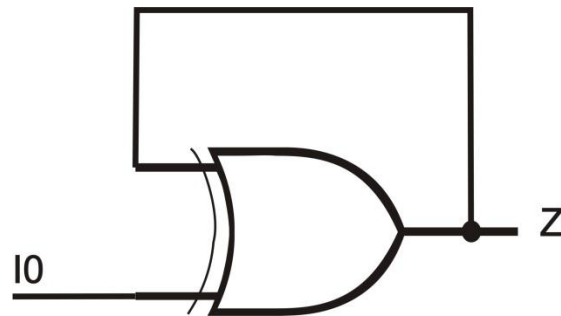


Figura 7-6: Enunciado

III. Imagen de la resolución del ejercicio:

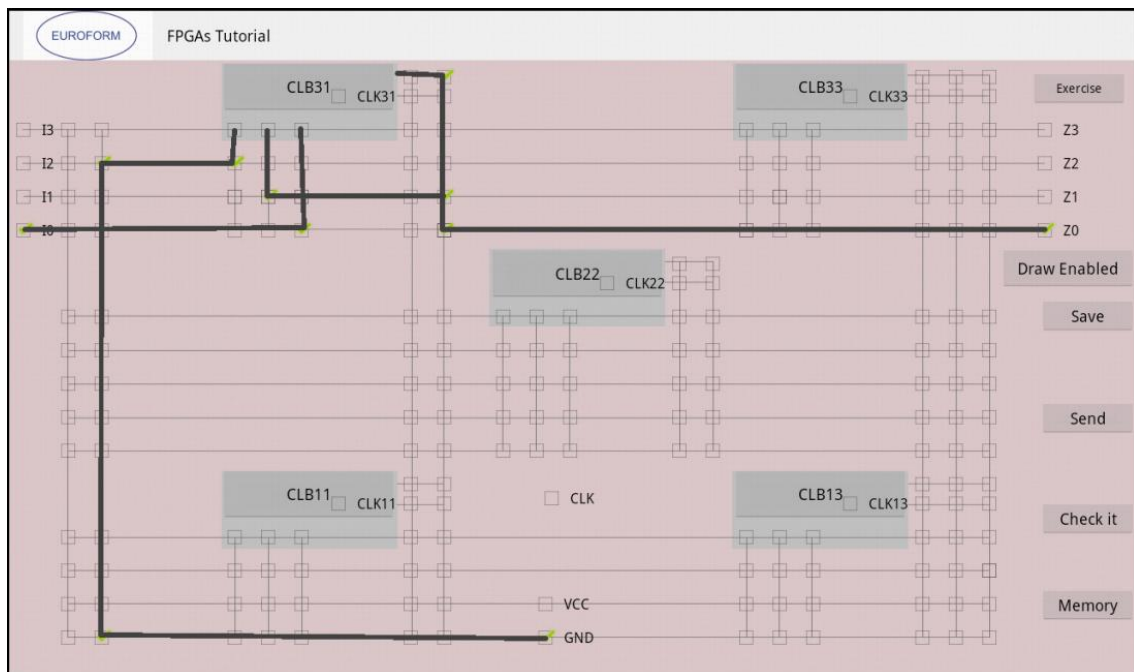


Figura 7-7: Layout principal

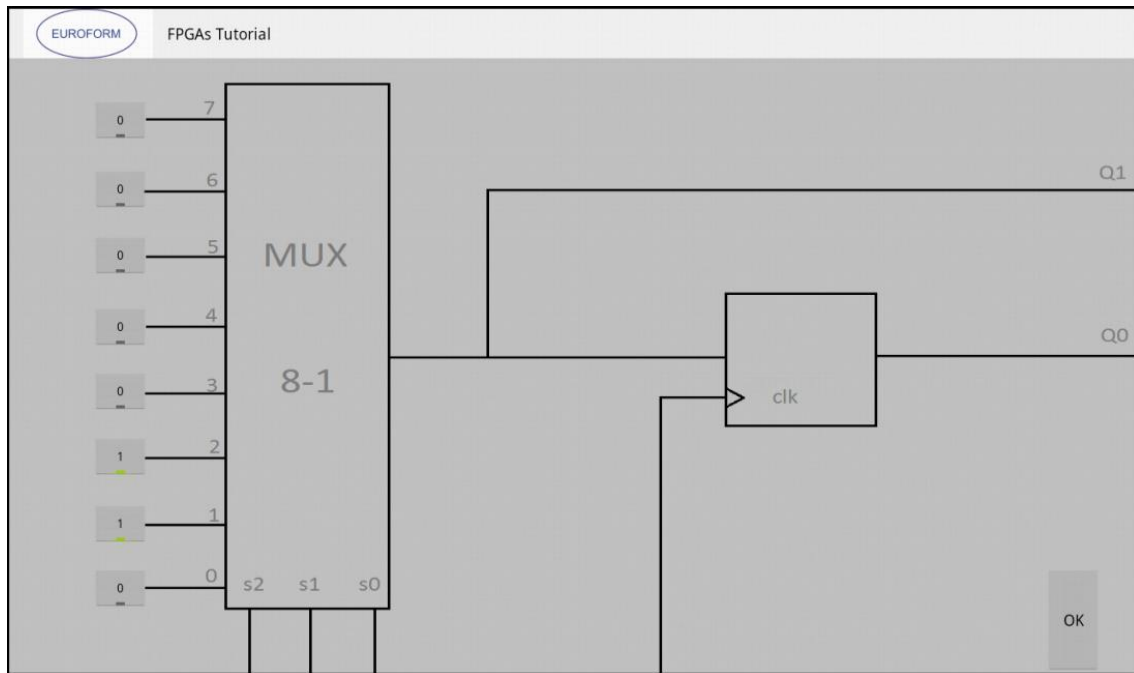


Figura 7-8: CLB31

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear una puerta lógica tipo XOR de dos entradas, usando el layout genérico.

Para resolverlo se escoge únicamente la entrada IO ya que la segunda entrada a la XOR es la salida de la misma. Por lo que la entrada IO se conecta a la entrada S0 del MUX y la salida del CLB31 se conecta a la salida Z0 y a la entrada S1 del MUX. A la entrada S2 del CLB31 se le conecta a GND, ya que no se usan las 4 primeras entradas del MUX.

En la segunda imagen se observa el interior del CLB31, y ahí simplemente hay que crear la puerta XOR, teniendo en cuenta que las 4 primeras entradas del MUX 8-1 no se usan, por lo que la entrada al MUX es "0110".

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- IO activo
- GND activo
- Z0 activo

7.5 Ejercicio #4

I. Número del ejercicio:

#4

II. Enunciado del ejercicio:

“Map a 4-input AND gate”

III. Imagen de la resolución del ejercicio:

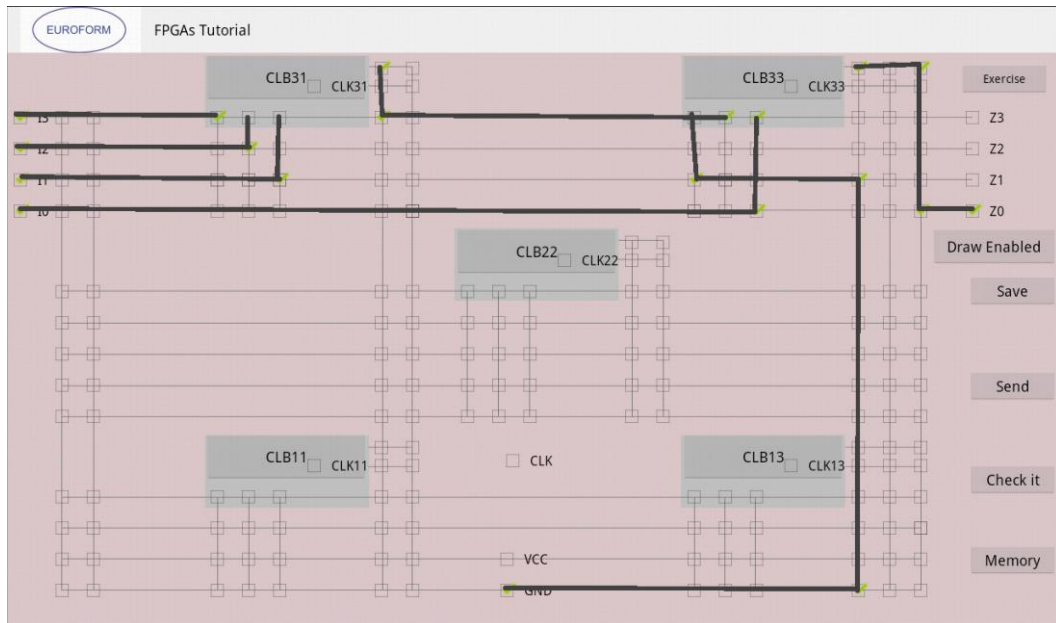


Figura 7-9: Layout principal

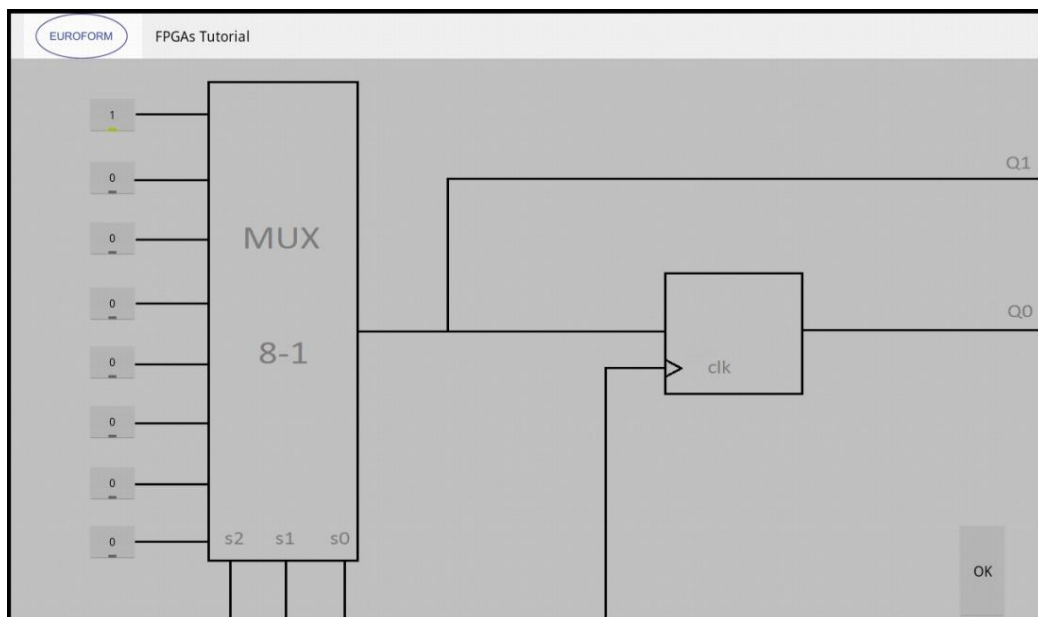


Figura 7-10: CLB31

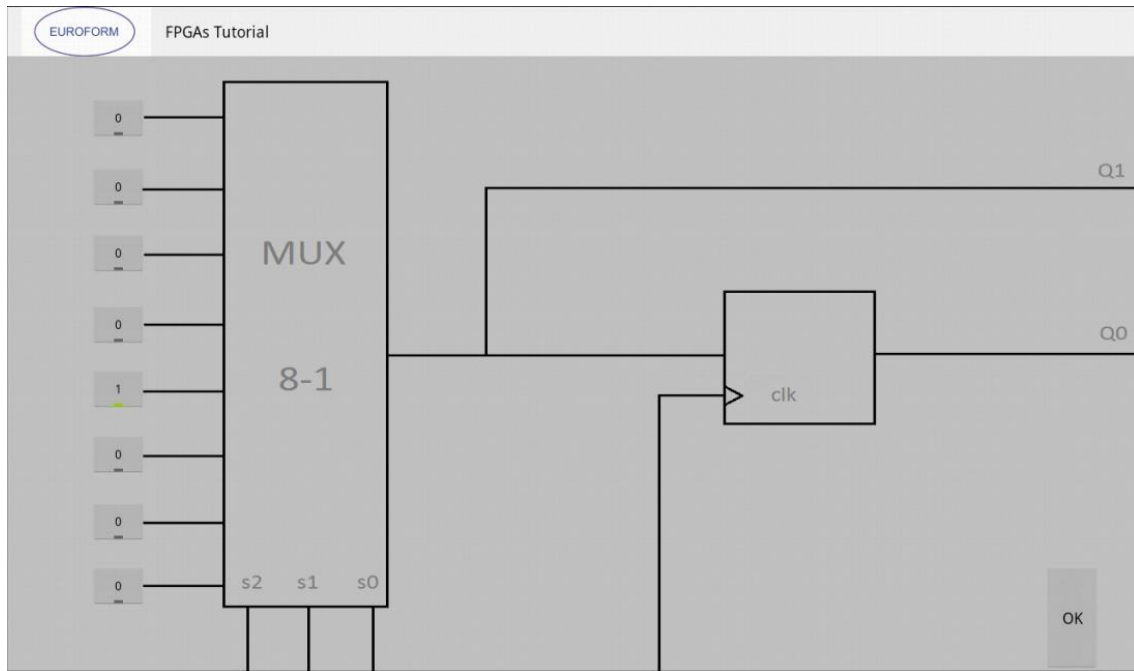


Figura 7-11: CLB33

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear una puerta lógica tipo AND de cuatro entradas, usando el layout genérico.

Para resolverlo se escogen las entradas I0, I1, I2 e I3. I1, I2 e I3 se interconexionan con las entradas S2, S1 y S0 del CLB31 (que a su vez son las del Mux 8-1). La salida Q1 del CLB31 se conecta a la entrada S1 del CLB33 y la entrada I0 al S0 también del CLB33. La salida de este CLB, la que no tiene que ver con el CLK, se conecta a la salida Z0 de la FPGA. A la entrada S2 del CLB33 se le conecta a GND, ya que no se usan las 4 primeras entradas del MUX.

En la segunda imagen se observa el interior del CLB31, y ahí simplemente hay que crear la puerta AND de tres entradas, por lo que la entrada al MUX es "10000000".

La tercera imagen es la del CLB33, teniendo en cuenta que las 4 primeras entradas del MUX 8-1 no se usan, la entrada al MUX es "1000".

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I3, I2, I1, I0 activos
- GND activo
- Z0 activo

7.6 Ejercicio #5

I. Número del ejercicio:

#5

II. Enunciado del ejercicio:

"Map the following pipeline stage."

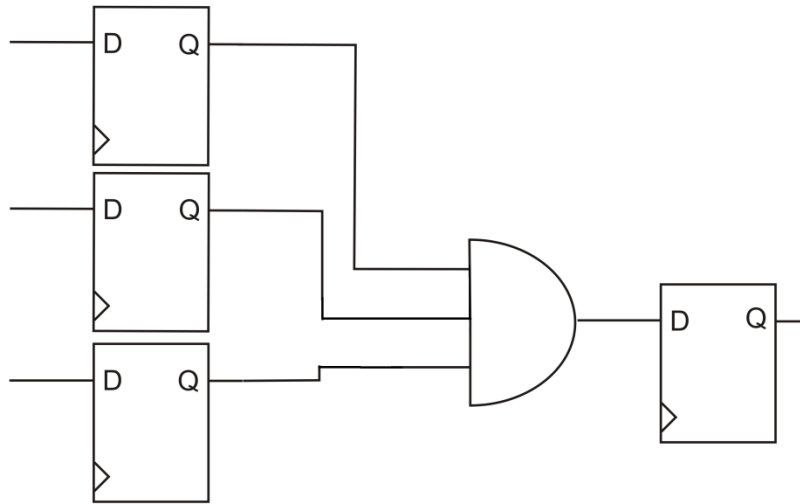


Figura 7-12: Enunciado

III. Imagen de la resolución del ejercicio:

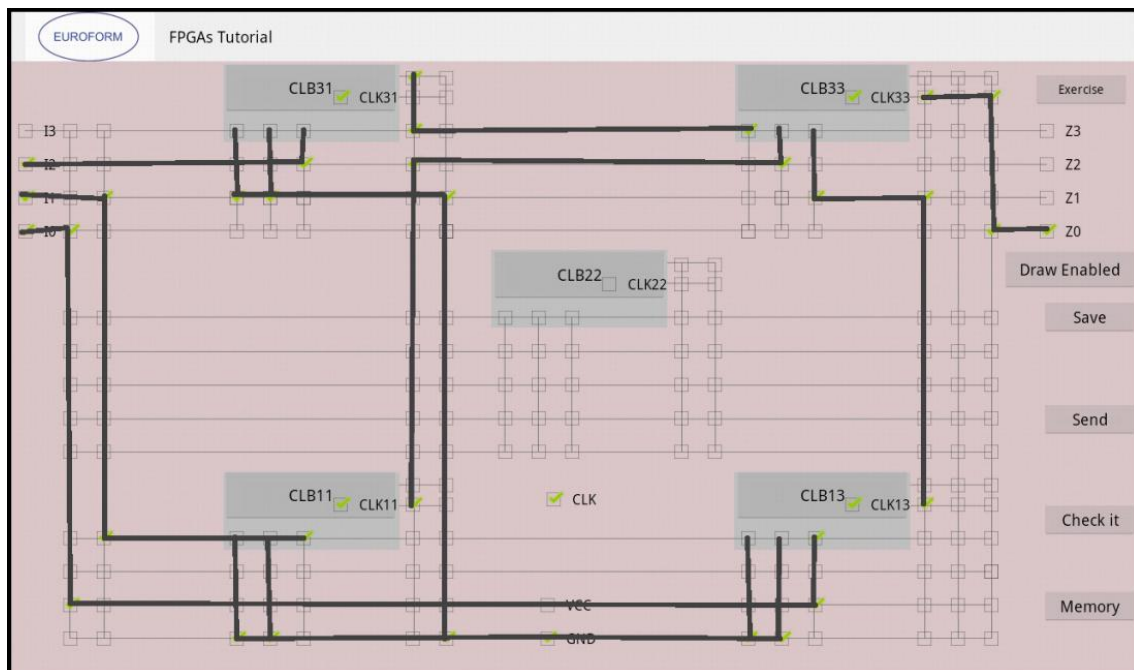


Figura 7-13: Layout principal

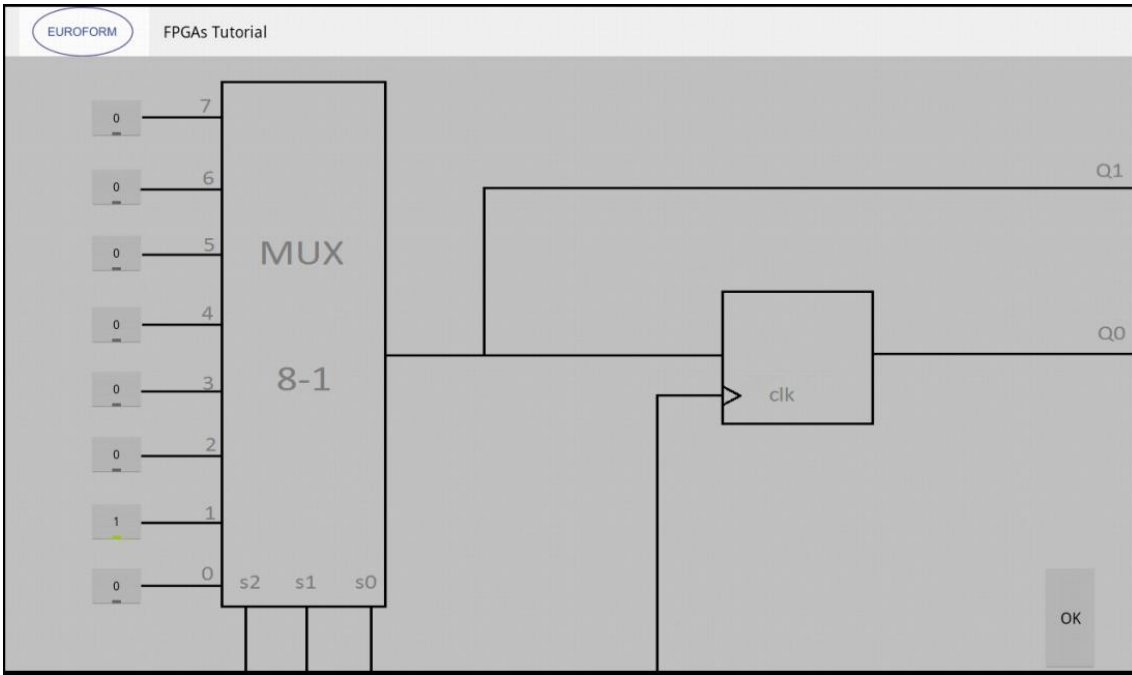


Figura 7-14: CLB31, CLB11 y CLB13

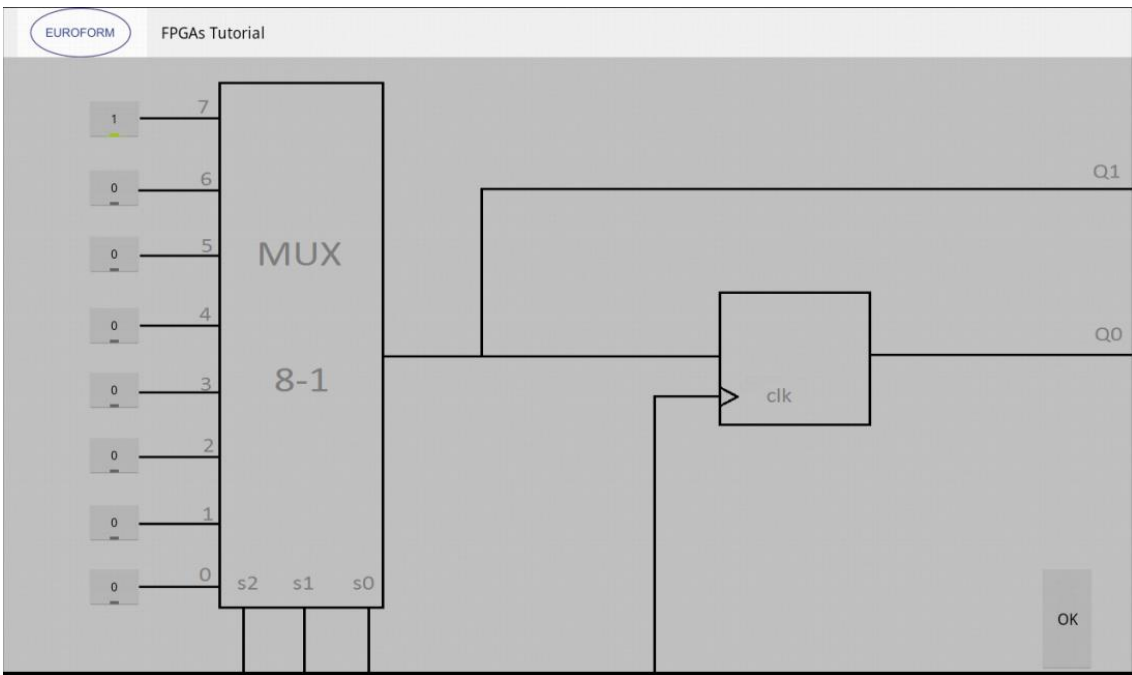


Figura 7-15: CLB33

IV. Explicación del ejercicio:

Para resolverlo se escogen las puertas I0, I1 e I2. Cada una de estas entradas va a un CLB diferente, ya que por el enunciado se observa que cada entrada va conectada a un Flip Flop. Una vez que se ha conectado cada entrada con un CLB, las salidas de cada uno de estos CLB se conecta a otro CLB, en este caso el CLB33, que actuará como una puerta AND de 3 entradas, cuya salida estará controlada por el Flip Flop, para así ahorrar el uso de otro CLB. Esta salida del CLB33, se conectará a la salida Z0. Las entradas que no se usan de los CLB31, CLB11 y CLB13, se conectan a GND.

En la segunda imagen se observa el interior de los CLB31, CLB11 y CLB13, ya que para los 3 el valor de sus entradas es el mismo, siendo este "10", puesto que simplemente deja pasar el valor que se venga en el MUX.

En la tercera imagen se observa el interior del CLB33, en el que se introduce como valor una AND de tres entradas, es decir "10000000".

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso de han elegido los CLB que se ven en las imágenes, pero podría haberse usado cualquiera de los 5 para conectar cualquiera de las entradas iniciales y cualquiera de ellos podría haber funcionado como el CLB que recreara la AND de tres entradas.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I2, I1, I0 activos
- GND activo
- Z0 activo
- CLK
- Se contabilizan 4 checkbox activos de CLK, de los 5 que hay para los CLB.

7.7 Ejercicio #6

I. Número del ejercicio:

#6

II. Enunciado del ejercicio:

“Map the Johnson counter of the figure. Write the output sequence if it is initialized at $Q_2Q_1Q_0 = 000$.”

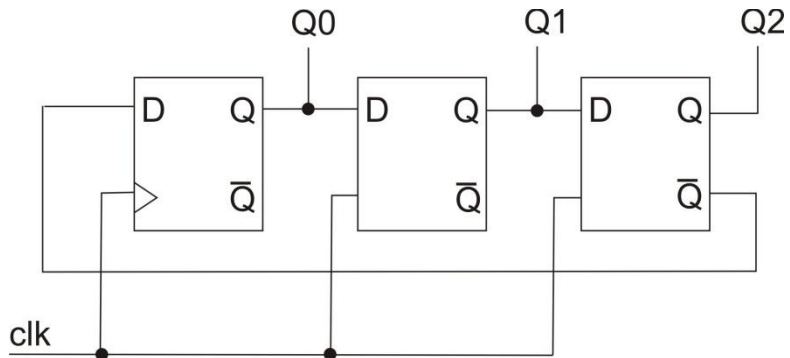


Figura 7-16: Enunciado

III. Imagen de la resolución del ejercicio:

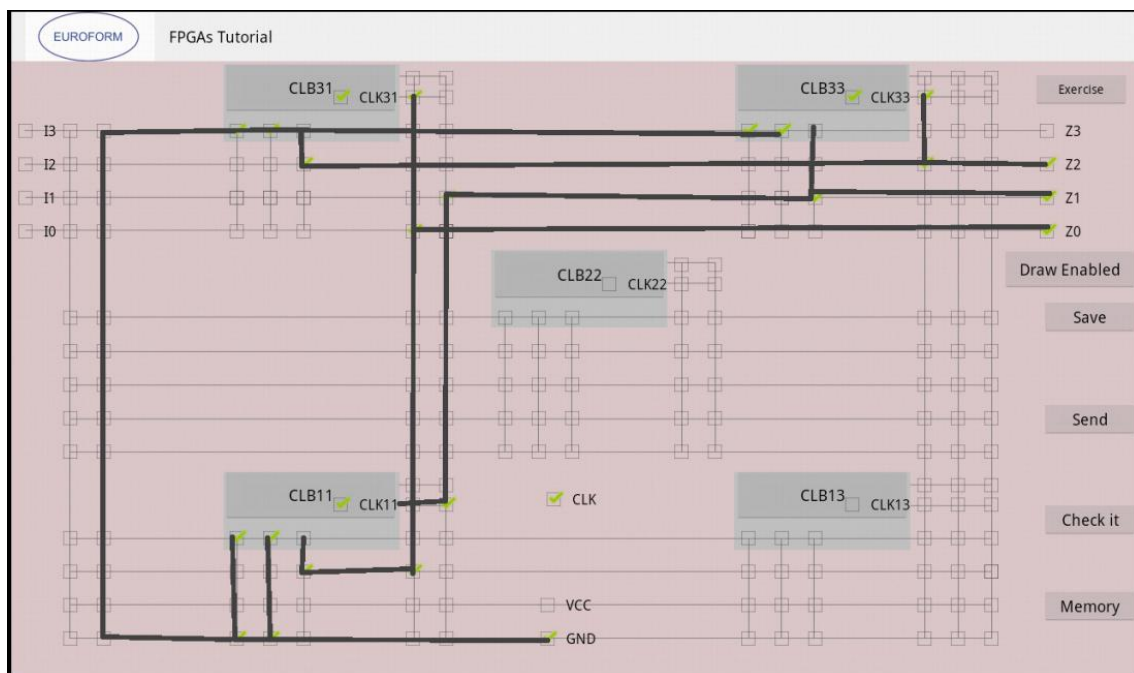


Figura 7-17: Layout principal

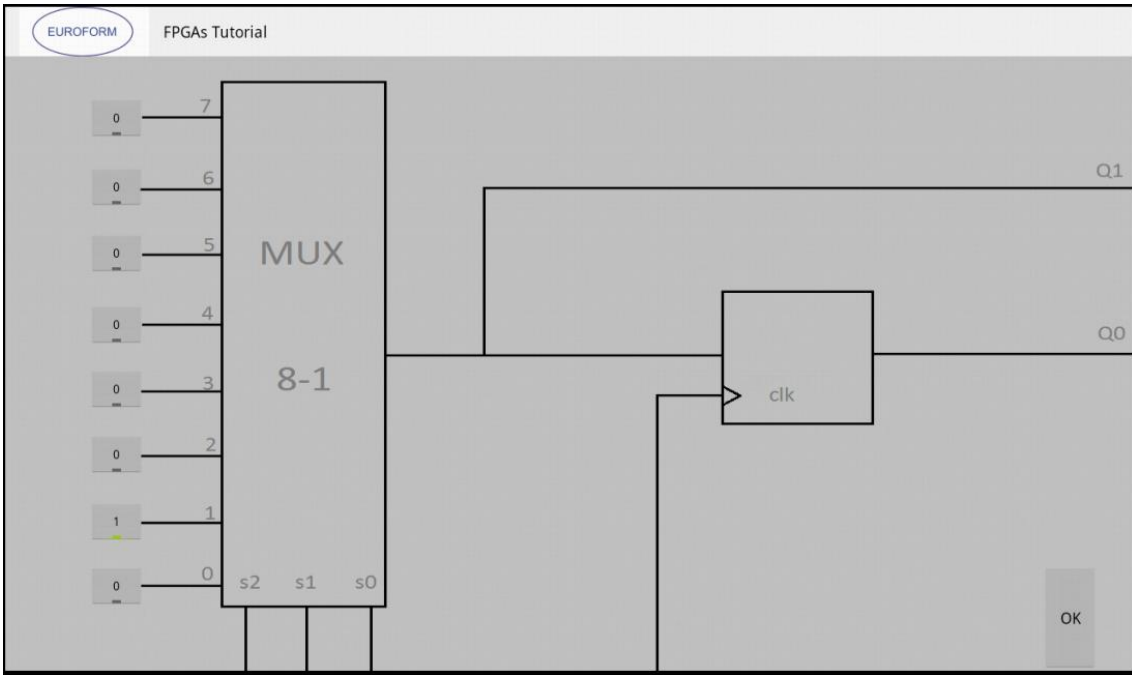


Figura 7-18: CLB31 y CLB11

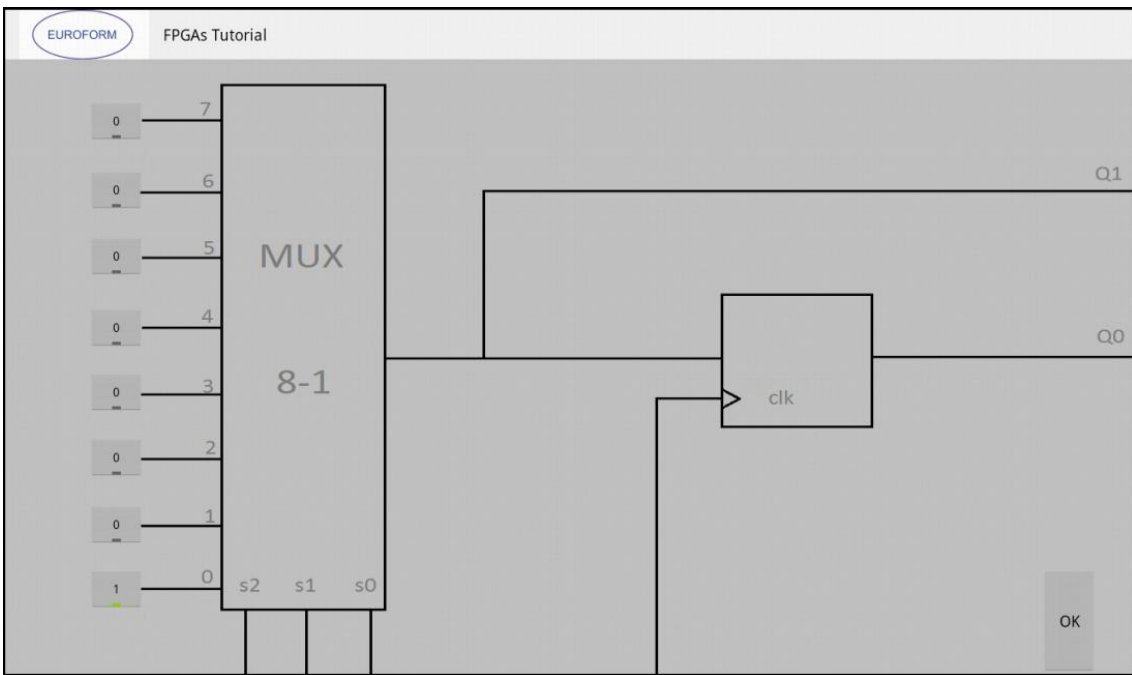


Figura 7-19: CLB33

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear contador de Johnson, usando el layout genérico.

En este ejercicio no hay que utilizar las entradas de la FPGA.

Se usan los CLB33, CLB31 y CLB11. El CLB31 actúa como el primer Flip Flop de la imagen, es decir, su salida va a Z0 y a la entrada del CLB11. A su vez la salida de este CLB va a la salida Z1 y a la entrada del CLB33. Por último, la salida del CLB33, se conecta a la salida Z2 y a la entrada del CLB31. Esto se observa en la primera imagen.

En la segunda imagen se puede observar el contenido de los CLB31 y CLB11, siendo los valores en sus entradas de "10", ya que dejan pasar el valor que entra en el MUX. EN el CLB33, ocurre el caso contrario, el valor que debe salir del mismo es el inverso, es decir, "01", ya que en el enunciado se puede ver claramente que la entrada del primer Flip Flop es la salida del último, pero invertida.

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- GND activo
- Z2, Z1, Z0 activo
- CLK activo
- 3 CLKs de los CLBS activos de los 5 que existen

7.8 Ejercicio #7

I. Número del ejercicio:

#7

II. Enunciado del ejercicio:

"Map the following circuit."

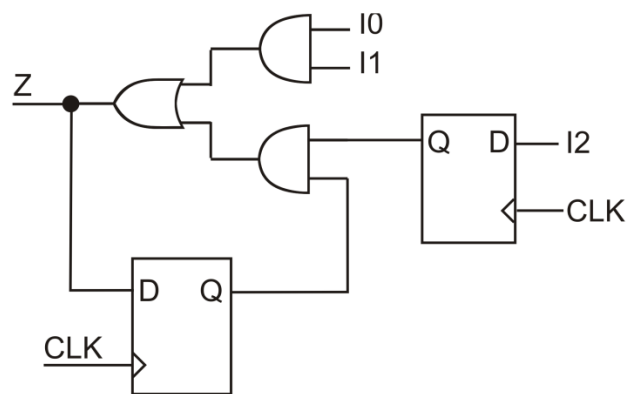


Figura 7-20: Enunciado

III. Imagen de la resolución del ejercicio:

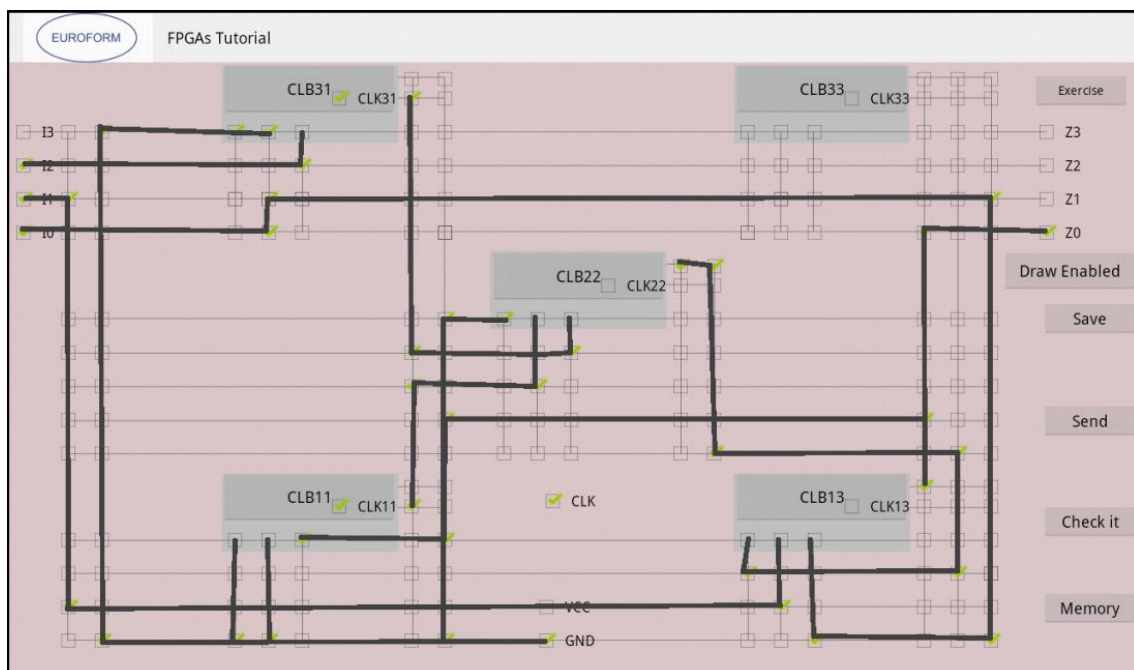


Figura 7-21: Layout principal

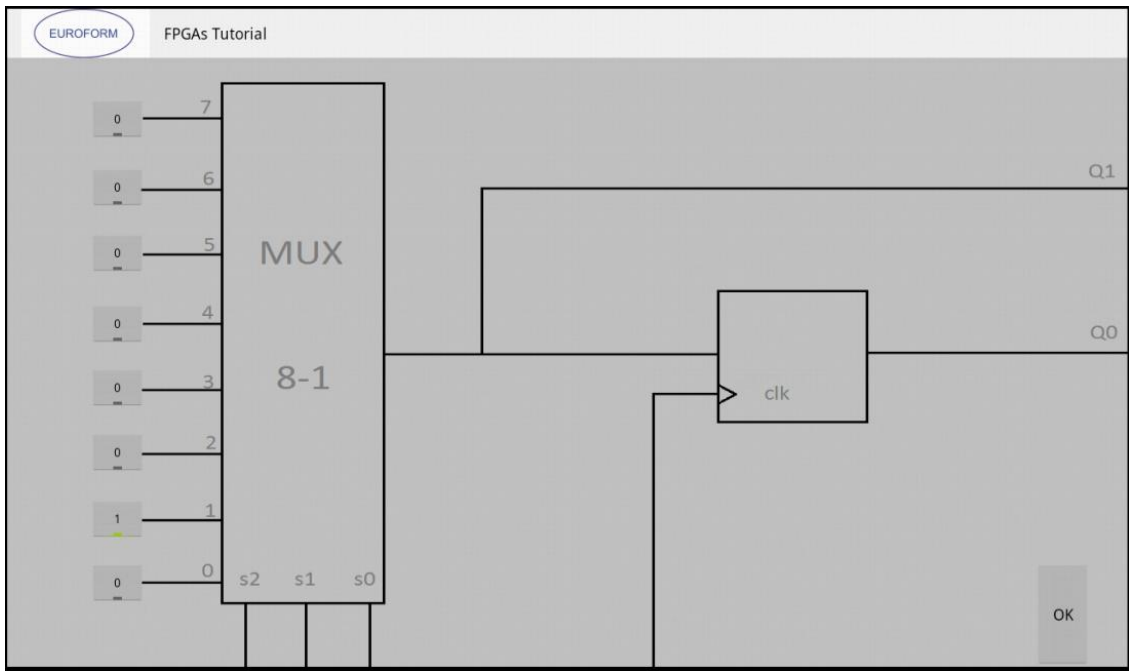


Figura 7-22: CLB31 y CLB11

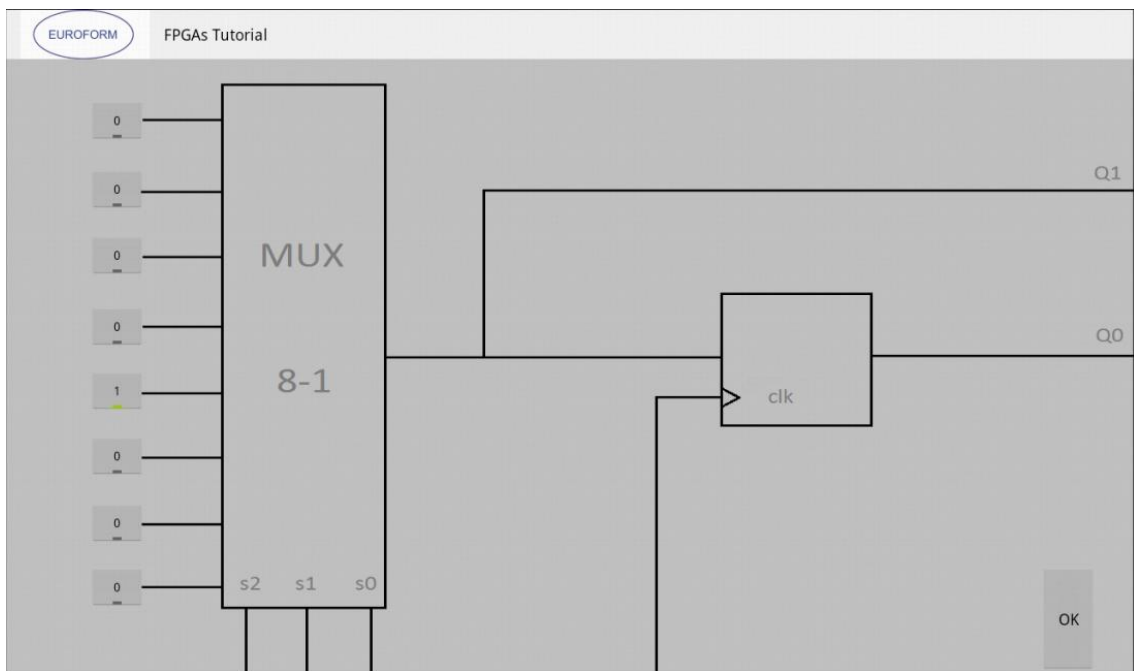


Figura 7-23: CLB22

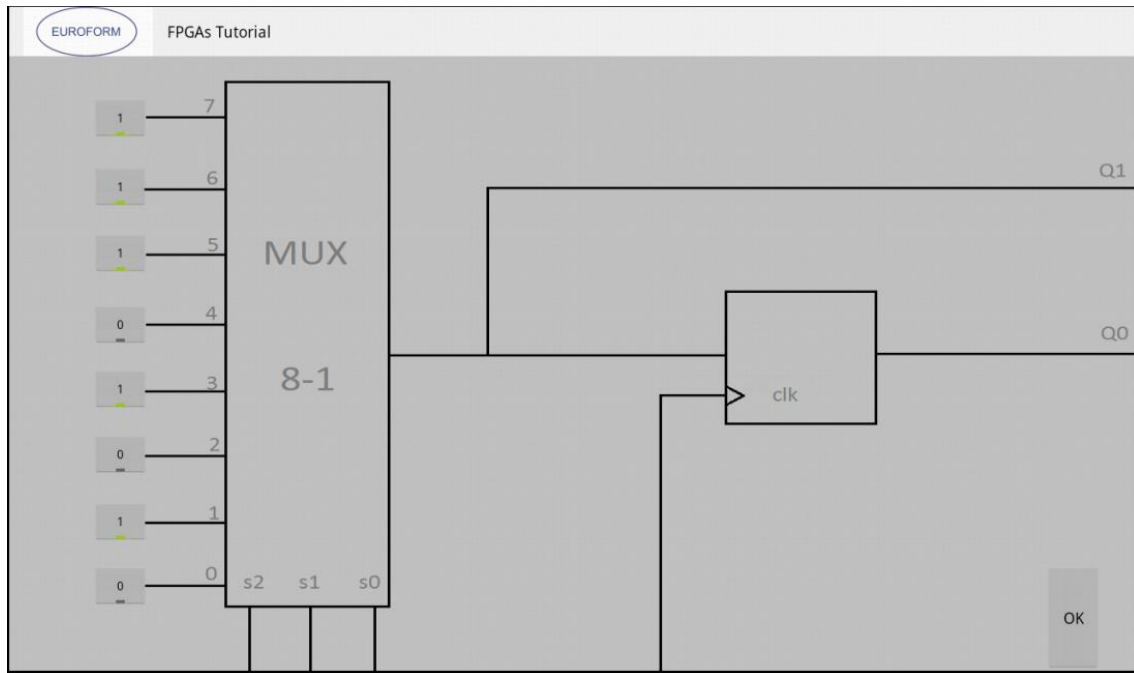


Figura 7-24: CLB13

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear el circuito lógico del enunciado usando el layout genérico. Para ello se ha seguido el dibujo del enunciado, realizando las conexiones como se observa en el mismo. No se entra en detalle de las conexiones realizadas. Esta es la solución más óptima respecto a las demás.

Se han usado 4 LUTs. En el CLB11 y en el CLB31, el interior de las LUTs tiene el valor que se introduce al MUX, en este caso “10” en ambos.

En el interior del CLB22, se ha realizado una puerta lógica AND de dos entradas, “1000”:

En el interior del CLB13, se realiza la función lógica que corresponde, que es una función OR de una AND de I0 e I1 y la salida de otra AND entre la salida Z0 y la entrada I2.

V. Datos de interés propios del ejercicio:

En este caso se han elegido esas LUTs y los contenidos de los mismos, pero podría resolverse introduciendo correctamente la misma información de forma diferente entre los LUTs.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I2,I1, I0 activos
- GND activo
- Z0 activo
- CLK activo
- 2 CLK de los CLB activos de los 5 existentes

7.9 Ejercicio #8

I. Número del ejercicio:

#8

II. Enunciado del ejercicio:

“Using Shannon Decomposition map the logic function $Z(DCBA) = \overline{D}CBA + \overline{D}C\overline{B}A + D\overline{C}BA + D\overline{C}\overline{B}A$. The sign $\overline{}$ indicates negation.”

III. Imagen de la resolución del ejercicio:

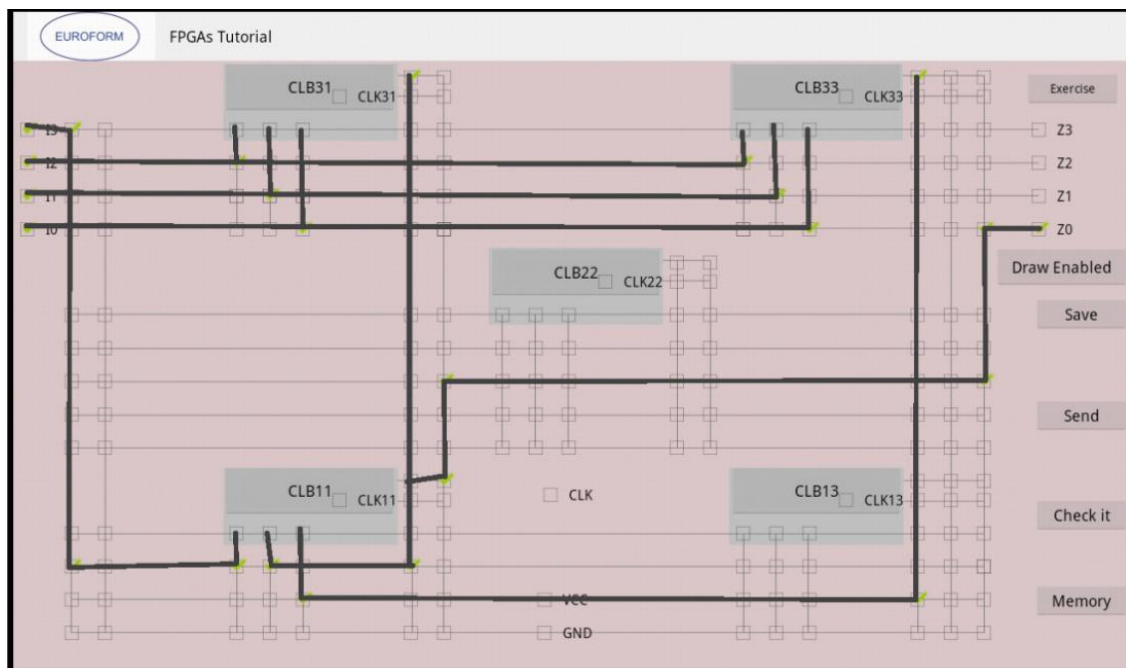


Figura 7-25: Layout principal

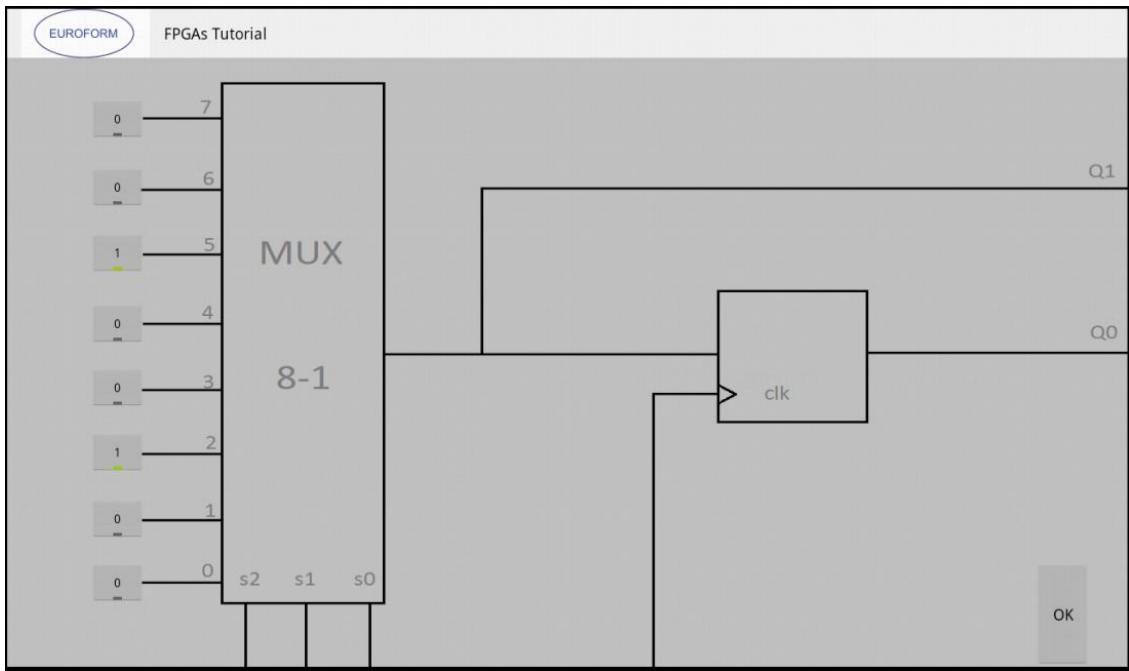


Figura 7-26: CLB31

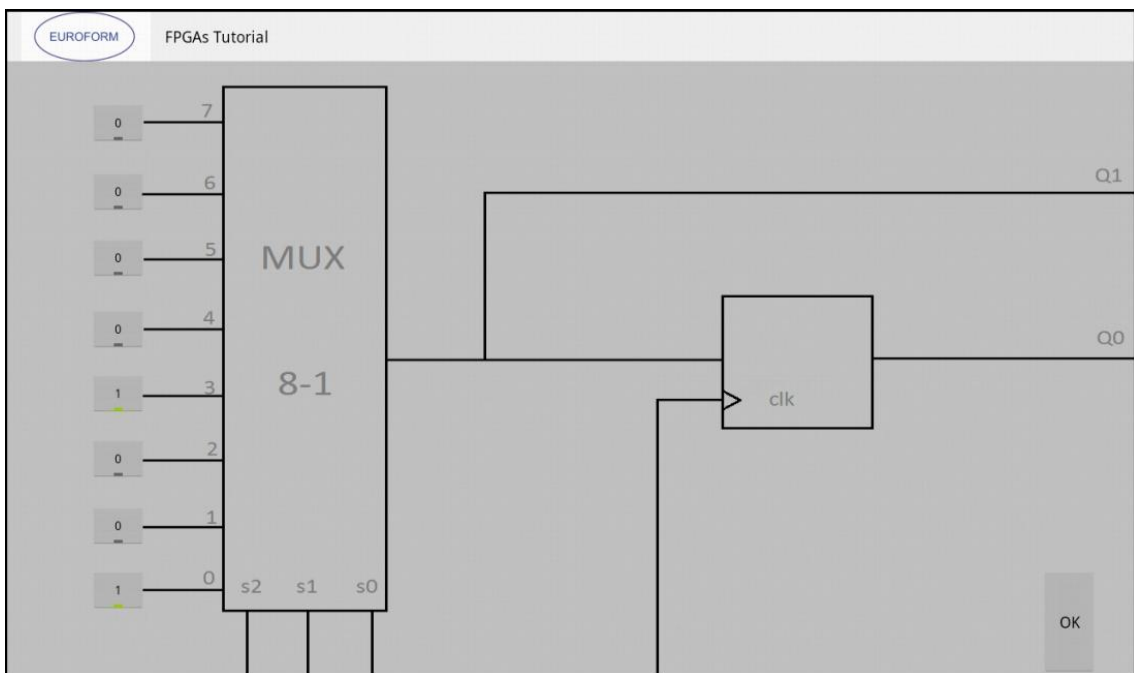


Figura 7-27: CLB33

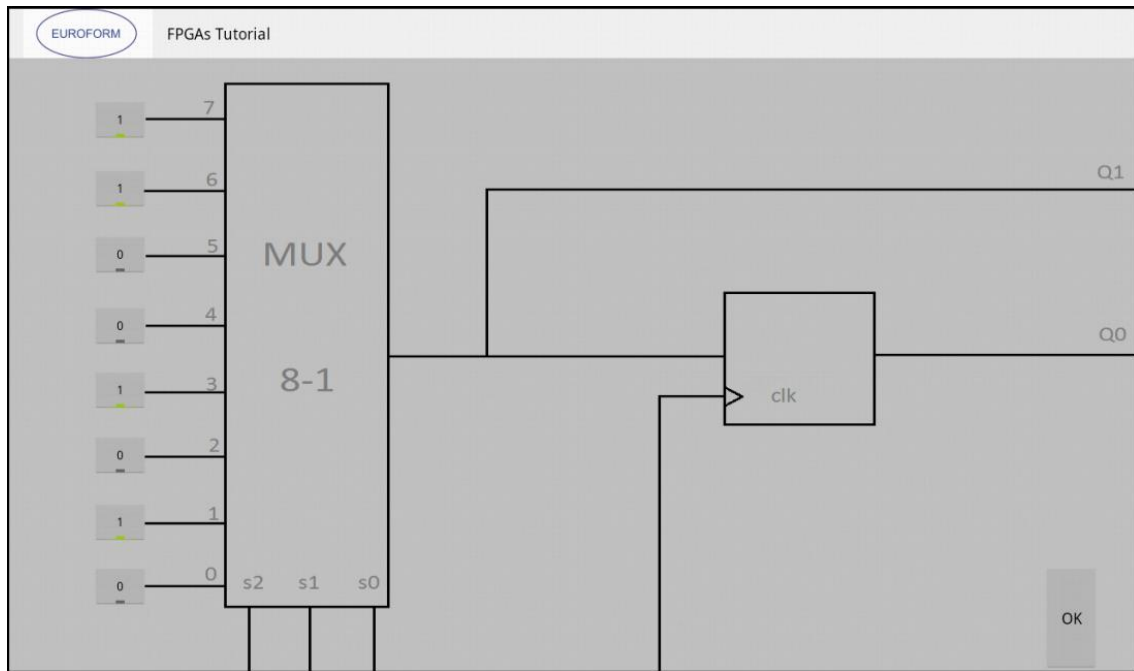


Figura 7-28: CLB11

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear una función resuelta mediante Shannon, usando el layout genérico.

Para terminar el ejercicio con éxito, primero se resuelve de forma manual la función. Una vez que se ha desglosado la función inicial y se tienen partes más pequeñas y practicables, se comienza a llevarlo a la aplicación.

I0, I1 e I2, se conectan a CLB31 y CLB33, las 3 a ambas LUTs. Sus salidas se conectan a CLB11 junto con I3, y la salida de esta LUT es la salida de la FPGA y se conectará a Z0.

Los valores de las LUTs se han obtenido resolviendo la función mediante el teorema de Shannon.

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I3, I2, I1, I0 activos
- Z0 activo

7.10 Ejercicio #9

I. Número del ejercicio:

#9

II. Enunciado del ejercicio:

"Map the following combinational circuit."

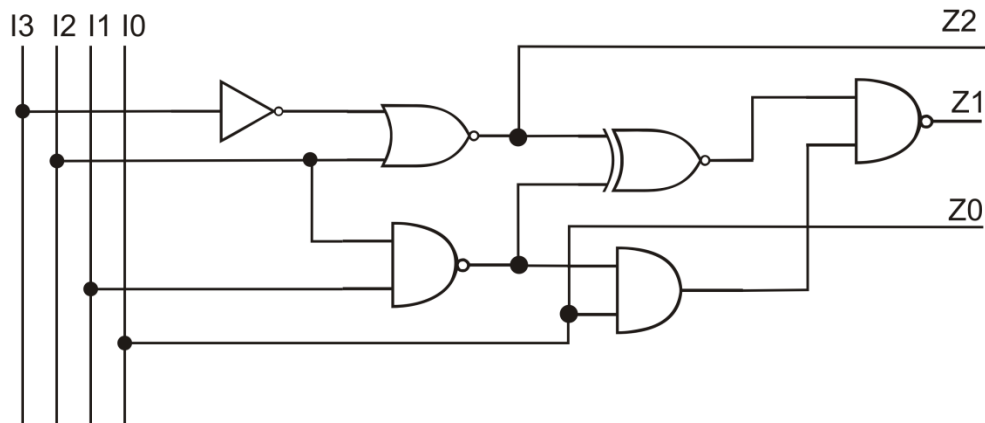


Figura 7-29: Enunciado

III. Imagen de la resolución del ejercicio:

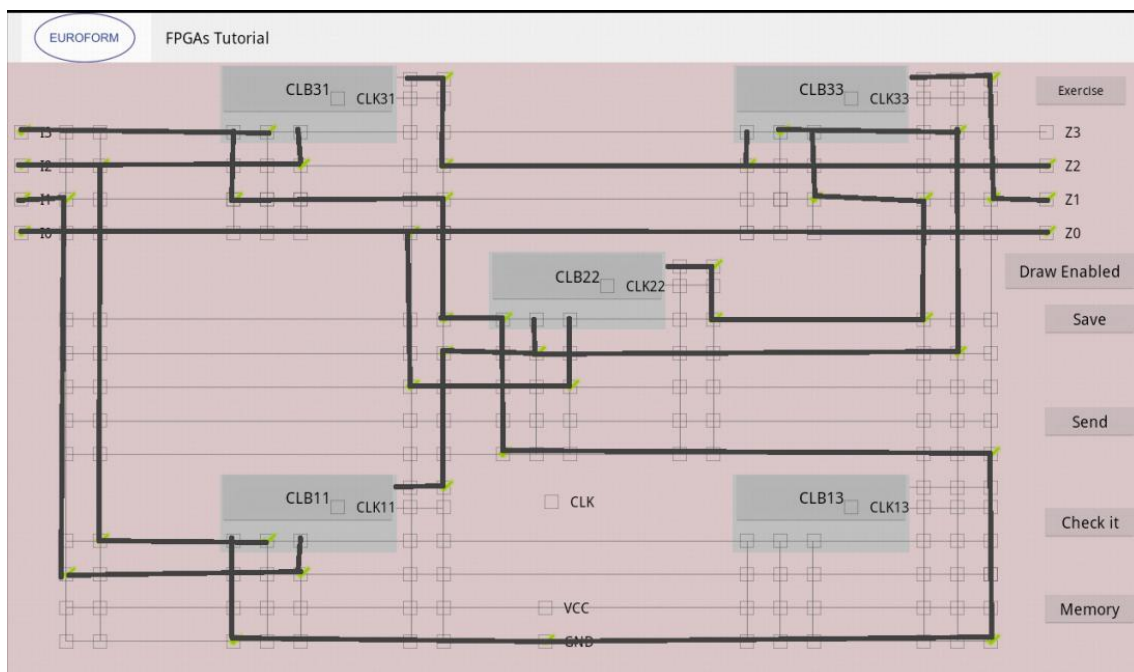


Figura 7-30: Layout principal

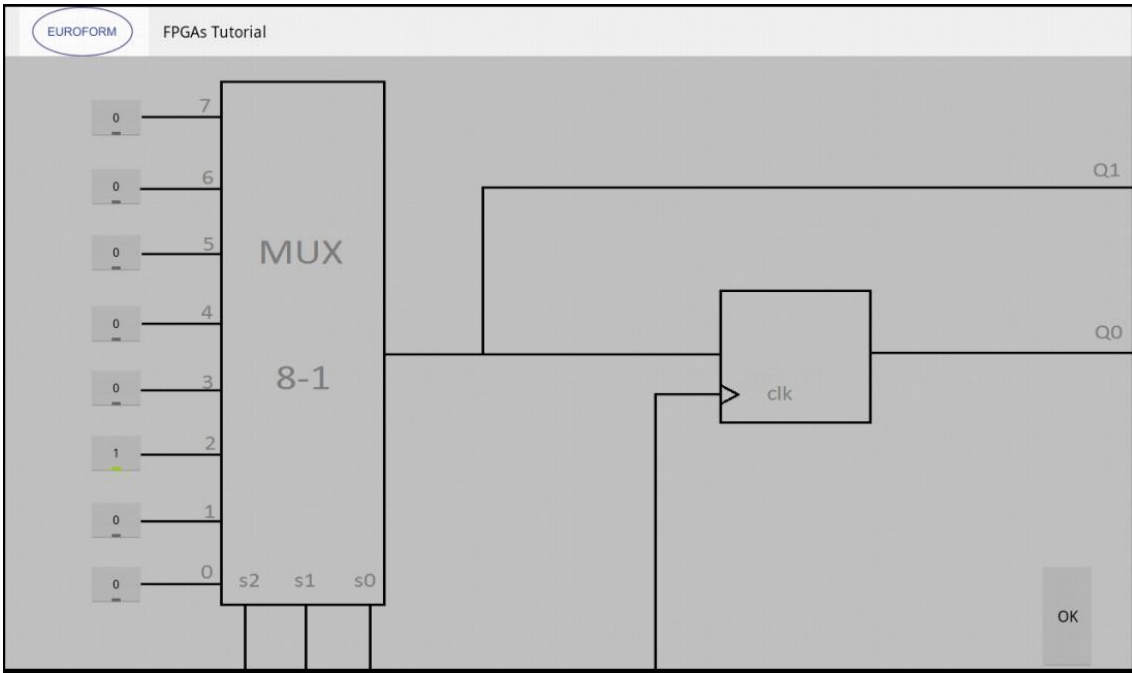


Figura 7-31: CLB31

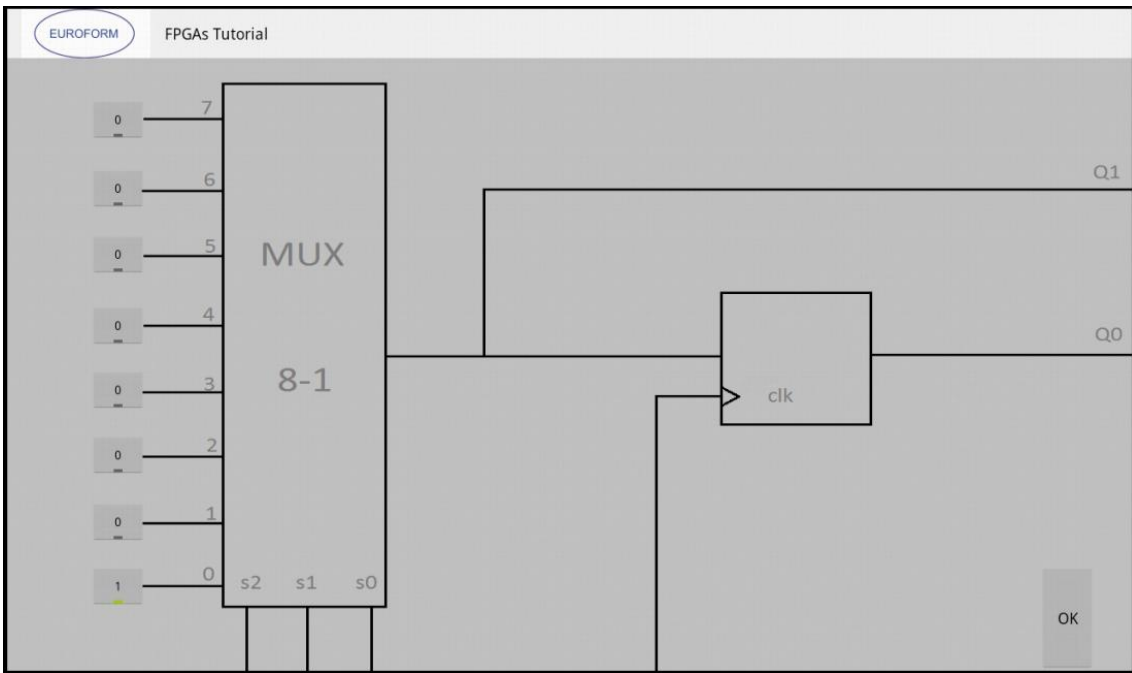


Figura 7-32: CLB11

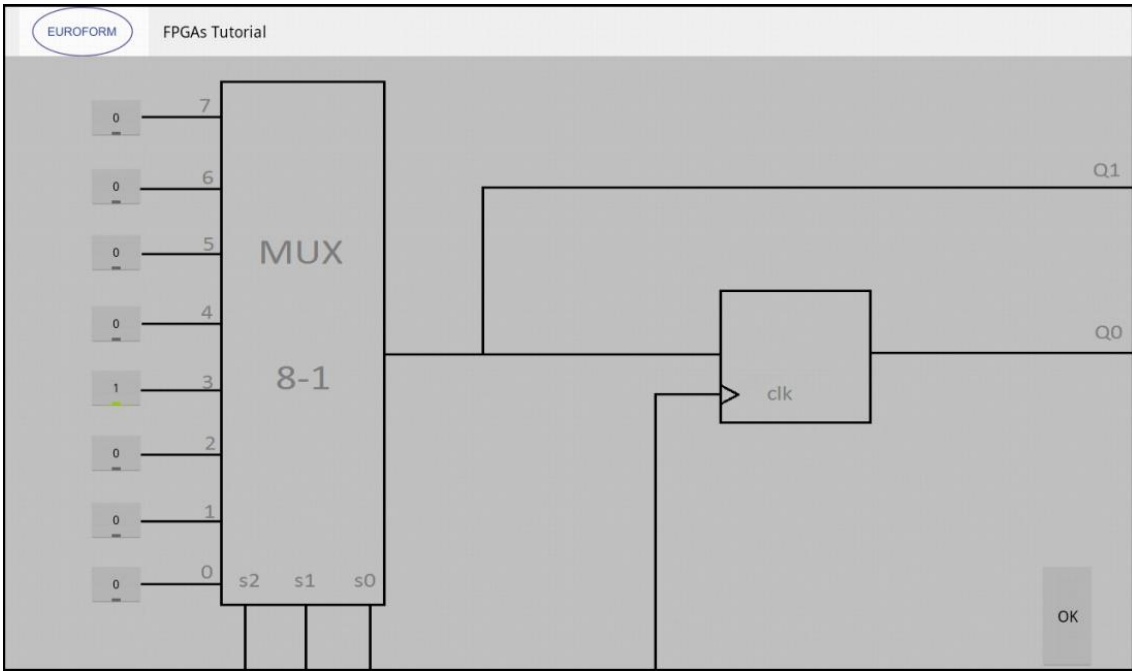


Figura 7-33: CLB22

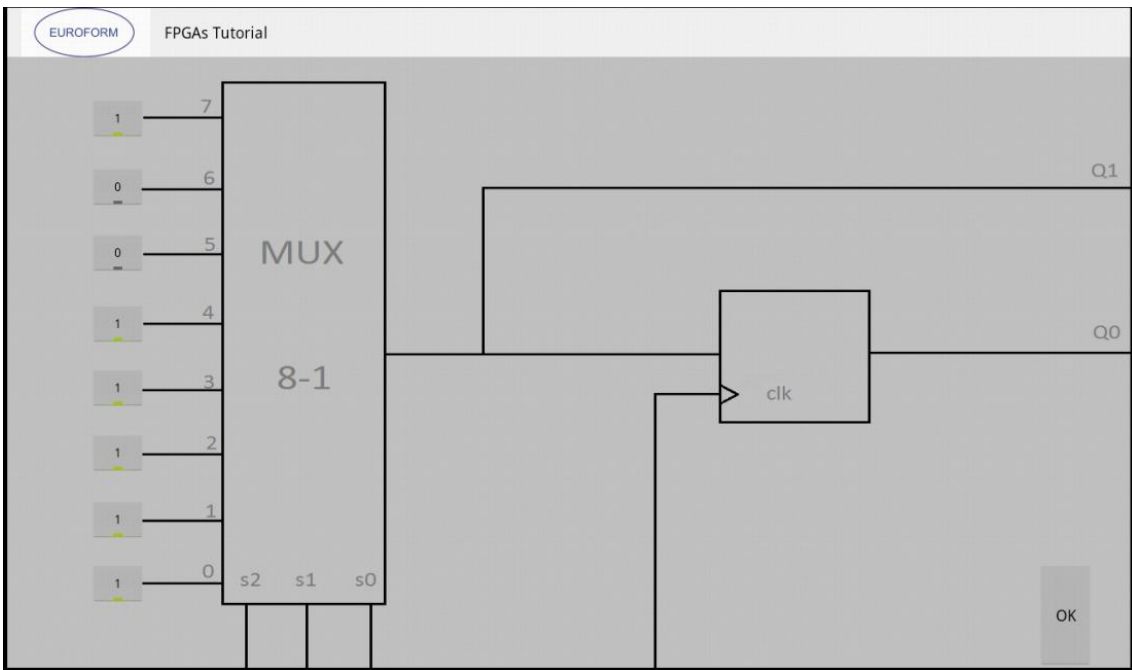


Figura 7-34: CLB33

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear el circuito lógico del enunciado usando el layout genérico.

Se han unido las entradas a los CLBs correspondiente, tal y como se observa en el enunciado. El único paso peculiar en este ejercicio, es que en el CLB33, se ha definido en su interior la información correspondiente al resultado de la función que queda ante la unión de las puertas lógicas XOR invertida y a la NAND de la derecha.

En cada imagen correspondiente a los CLB, se puede observar el valor que se debe poner en su interior. Este valor se haya en base a la puerta lógica del enunciado que se decide colocar en ese CLB.

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

Para implementar la resolución del ejercicio se ha elegido hacerlo con esos CLBs, pero en lugar de esta combinación de 4, podría haberse usado otra combinación de 4 LUTs cualquiera siempre y cuando las conexiones y la información de su interior sea correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I3, I2, I1, I0 activos
- GND activo
- Z0, Z1, Z2 activos

7.11 Ejercicio #10

I. Número del ejercicio:

#10

II. Enunciado del ejercicio:

“Map a XOR of 3 input signals using the minimum number of LUTs.”

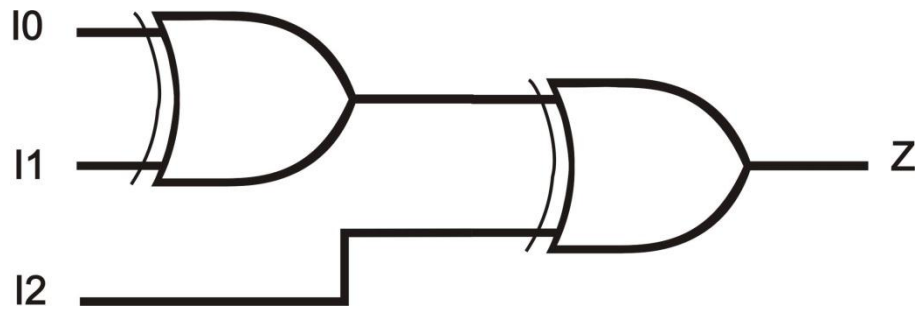


Figura 7-35: Enunciado

III. Imagen de la resolución del ejercicio:

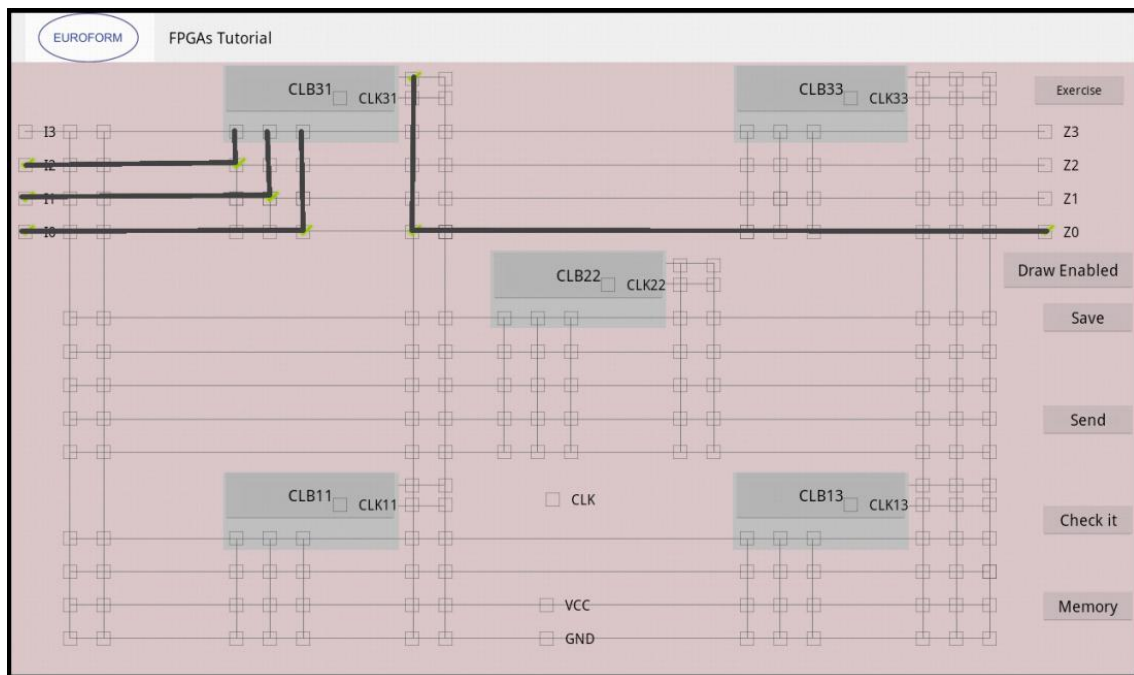


Figura 7-36: CLB31

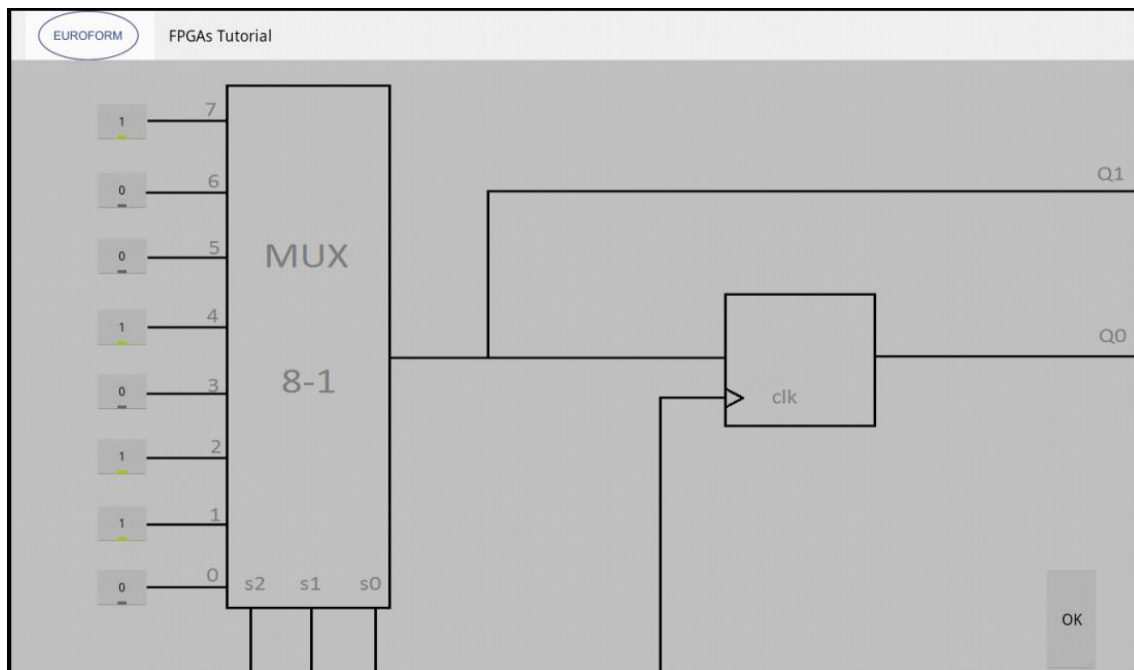


Figura 7-37: CLB31

IV. Explicación del ejercicio:

En este ejercicio se ha de mapear una puerta lógica tipo XOR de 3 entradas, usando el layout genérico.

Para resolverlo se escogen las entradas I0, I1 e I2 y se interconexionan respectivamente con las entradas S0, S1 y S2 del CLB31 (que a su vez son las del Mux 8-1). La salida elegida es la Z0, y a ésta llega la salida del CLB31 que no está controlada por el CLK.

En la segunda imagen se observa el interior del CLB31, y ahí simplemente hay que crear la puerta XOR de tres entradas, que es “10010110”

Una vez se ha resuelto, se pasa a dibujar las líneas para resaltar las interconexiones creadas.

V. Datos de interés propios del ejercicio:

En este caso se ha elegido el CLB31, pero se podría haber elegido cualquiera de los 5, por lo que cualquier solución eligiendo otro CLB también sería correcta.

En este caso, para realizar el chequeo de este ejercicio se ha realizado la comprobación de los siguientes checkbox:

- I2, I1, I0 activos
- Z0 activo

7.12 Ejercicio #11

I. Número del ejercicio:

#11

II. Enunciado del ejercicio:

"BLANK EXERCISE"

III. Imagen de la resolución del ejercicio:

No tiene una resolución fija

IV. Explicación del ejercicio:

Este ejercicio será usado como plantilla que se usará por parte del alumno para solucionar los ejercicios que el tutor mande. Estos ejercicios serán creados por cuenta del tutor, sin estar desarrollados en la aplicación.

V. Datos de interés propios del ejercicio:

En este ejercicio no se puede realizar comprobación alguna, ya que es la plantilla que se ha de usar para la resolución de los ejercicios que el tutor manda según su criterio.

8 REFERENCIAS

- [1] Jerome (J.F.) Di Marzio, "Android, A Programmer's Guide" Mc Graw Hill
- [2] Rick Rogers, John Lombardo, Zigurd Mednieks & Blake Meike, "Android Application Development", Oreilly
- [3] Chris Haseman, "Android Essentials", firstPress
- [4] Nicolas Gramlich, "Android Programming", andbook.anddev.org
- [5] Bruce Eckel, "Piensa en Java", Prentice Hall
- [6] Javier García de Jalón, José Ignacio Rodríguez, Íñigo Mingo, Aitor Imaz, Alfonso Brazález, Alberto Larzabal, Jesús Calleja, Jon García, "Aprende Java como si estuvieras en primero"
- [7] Jesús Tomás Gironés, "El gran libro de Android"
- [8] Zechner, Mario, "Beginning Android games"
- [9] Gargenta, Marko "Learning Android"
- [10] Mednieks, Zigurd, "Programming Android"
- [11] Murphy, Mark L., "Beginning android"
- [12] Lee, Wei-Meng, "Beginning Android application development"
- [13] www.sgoliver.net
- [14] <http://www.androidcurso.com>
- [15] Curso Miriadax.net Android: Programación de aplicaciones
- [16] <http://www.elandroidelibre.com/2013/02/graba-la-pantalla-de-tu-android-con-screencast-video-recorder.html>
- [17] <http://andbook.anddev.org/files/andbook.pdf>
- [18] <http://www.edu4java.com/index.html>
- [19] <http://blog.vidasconcurrentes.com/android/creando-una-aplicacion-de-android-el-juego-y-la-logica-parte-2/>
- [20] <http://android-ui-utils.googlecode.com/hg/asset-studio/dist/icons-launcher.html#foreground.type=image&foreground.space.trim=0&foreground.space.pad=0&foreColor=33b5e5%2C0&crop=0&backgroundShape=circle&backColor=000%2C100>
- [21] <http://www.terminalesandroid.com/graficos-2d-y-aplicaciones-multimedia>
- [22] <http://www.foro.lospillaos.es/viewtopic.php?p=28784>
- [23] <http://insanitydesign.com/wp/projects/nehe-android-ports/>
- [24] <http://bestsiteinthemultiverse.com/2008/11/android-graphics-example/>
- [25] <http://www.maestrosdelweb.com/editorial/curso-android-enviar-emails/>
- [26] <http://www.android-spa.com/viewtopic.php?t=1845>
- [27] <http://www.nosinmiubuntu.com/2011/11/como-guardar-datos-en-android-bases-de.html>
- [28] <http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=144&inicio=>
- [29] <https://polimedia.upv.es/visor/?id=9793556b-6331-8049-a316-ba7a3f5dd5cb>
- [30] <http://elbauldelprogramador.com/opensource/programacion-android-recursos-usando/>

- [31]http://elbauldelprogramador.com/opensource/programacion-android-interfaz-grafica_18/
- [32]<http://francho.org/2010/04/23/truco-android-textos-largos-en-textview/>
- [33]<http://www.tutorialforandroid.com/2009/06/drawing-with-canvas-in-android.html>
- [34]<http://stackoverflow.com/questions/9951329/android-draw-with-finger-over-webview>
- [35]<http://android-coding.blogspot.mx/2012/12/draw-path-on-surfaceviews-canvas.html>
- [36]<http://stackoverflow.com/questions/5248583/how-to-get-a-color-from-hexadecimal-color-string>
- [37]<http://developer.android.com/reference/android/graphics/Path.html>
- [38]<http://stackoverflow.com/questions/4675750/lock-screen-orientation-android>
- [39]<http://stackoverflow.com/questions/2067426/android-force-horizontal-layout>
- [40]<http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/string.htm>
- [41]<http://www.novanebula.net/blog/archives/142-Convertir-String-a-int-en-Java-y-viceversa.html>
- [42]http://www.tutorialspoint.com/java/java_string_split.htm
- [43]<http://stackoverflow.com/questions/3470042/intent-uri-to-launch-gmail-app>
- [44][http://es.wikipedia.org/wiki/Tableta_\(computadora\)](http://es.wikipedia.org/wiki/Tableta_(computadora))
- [45]<http://www.huntmads.com/the-continuous-growth-of-tablets.html?lang=es>
- [46]<http://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados/>
- [47]<http://www.sgoliver.net/blog/?p=1295>
- [48]<http://blogthinkbig.com/tendencias-dispositivos-moviles-2013/>
- [49]http://www.kolls.net/gatesim/gatesim_osc.png
- [50]<http://robei.com/>
- [51]<http://es.kioskea.net/download/descargar-17369-electrodroid>
- [52]http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array

9 CONCLUSIONES

1. Desde el punto de vista educativo, se ha creado una herramienta que permite practicar los siguientes conceptos:
 - Teorema de Shannon para mapear en LUTs
 - Mapeo de un pipeline
 - Partición óptima en LUT
 - LUT transparente
 - LUT como inversor
2. La aplicación reemplaza ventajosamente a la guía de ejercicios en papel del tema “Arquitectura de FPGAs” del curso “Dispositivos E. Especializados”, pues permite:
 - Trabajar de manera móvil, en cualquier parte
 - No utilizar lápiz y papel, por ejemplo, en el Metro
 - Compartir resultados por correo electrónico
 - Realizar verificaciones parciales de los resultados
3. Desde el punto de vista de programación Android, se han resuelto los siguientes problemas relacionados con herramientas de educación:
 - Se permite llevar una plantilla única, en la que se pueden desarrollar varios tipos de ejercicios.
 - Hay un problema en el que se puede resolver cualquier ejercicio planteado por el profesor, usando el mismo layout que para los demás ejercicios.
 - Al poder ver las posiciones de memoria y el contenido de los CLB permite tener una visión global del ejercicio.
4. Una aplicación de este tipo requiere 600 horas de programación y 100 de aprendizaje de Java aplicado en Android.
5. Se han realizado pruebas de funcionamiento de la aplicación y ésta se incorporará a la docencia del curso DIE con 2013.

ANEXOS

10 MANUAL DEL PROGRAMADOR

En esta parte de la memoria se van a incluir algunas de las clases que se han implementado para el correcto funcionamiento de la aplicación.

10.1 NewExercise

En esta clase se ha desarrollado toda la lógica necesaria para que el usuario pueda interactuar con el layout de la FPGA.

Se declaran las variables globales de la clase. Estas son referentes a la clase y método que se usará para que se logre pintar las líneas que el usuario ha dibujado en NewExercise1

```
Paint mPaint;
Canvas mCanvas;
Path mPath;
ArrayList<Path> _graphics = new ArrayList<Path>();
```

Se declaran 400 posibles puntos para 400 posibles rectas.

```
int[] x1 = new int[400];
int[] y1 = new int[400];
int[] x = new int[400];
int[] y = new int[400];
```

a. Clase Mens

Con esta clase se manda pintar en el layout lo que hay en el array `_graphics`, que son las coordenadas de las líneas dibujadas por el usuario.

```
public class Mens extends View {

    public Mens(Context context) {
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas) {

        for (Path mPath : _graphics) {
            canvas.drawPath(mPath, mPaint);
        }

        invalidate();
    }
}
```

b. Método sonidito

Método usado para provocar un sonido al pulsar ciertos botones.

```
public void sonidito() {
    ToneGenerator sound = new ToneGenerator(AudioManager.STREAM_SYSTEM, 100);
    sound.startTone(ToneGenerator.TONE_PROP_BEEP);
}
```

c. Método resultado

Este método se usa para realizar la comprobación parcial de los ejercicios. En base al ejercicio que sea, se pasa un parámetro que sumado al "switch", realiza la comprobación de los checkbox correspondientes.

```
@SuppressWarnings("unused")
public void resultado(View button3) {

    Bundle figuraresultado = getIntent().getExtras();

    int[] arraymemoria = funcionlogica();
    int[] comprobacion1 = figuraresultado.getIntArray("comprobacion1");
    int[] comprobacion2 = figuraresultado.getIntArray("comprobacion2");
    int[] comprobacion3 = figuraresultado.getIntArray("comprobacion3");
    int[] comprobacion4 = figuraresultado.getIntArray("comprobacion4");
    int[] comprobacion5 = figuraresultado.getIntArray("comprobacion5");
    int param = figuraresultado.getInt("param");
    int contadorlineas = figuraresultado.getInt("contadorlineas");
    int identificativo = figuraresultado.getInt("identificativo");

    int[] cero = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };

    if (figuraresultado.getIntArray("comprobacion1") == null) {
        comprobacion1 = cero;
    }
    if (figuraresultado.getIntArray("comprobacion2") == null) {
        comprobacion2 = cero;
    }
    if (figuraresultado.getIntArray("comprobacion3") == null) {
        comprobacion3 = cero;
    }
    if (figuraresultado.getIntArray("comprobacion4") == null) {
        comprobacion4 = cero;
    }
    if (figuraresultado.getIntArray("comprobacion5") == null) {
        comprobacion5 = cero;
    }

    int contclk = 0;

    int primero = 0, segundo = 0, tercero = 0, cuarto = 0, quinto = 0;

    primero = arraymemoria[200];
    segundo = arraymemoria[201];
    tercero = arraymemoria[202];
    cuarto = arraymemoria[203];
    quinto = arraymemoria[204];

    contclk = primero + segundo + tercero + cuarto + quinto;

    switch (identificativo) {

        case 1:

            if (arraymemoria[199] == 0 && arraymemoria[182] == 1
                && arraymemoria[170] == 1 && contclk == 0
                && arraymemoria[185] == 1 && arraymemoria[183] == 1
                && arraymemoria[0] == 0 && arraymemoria[171] == 0
                && arraymemoria[186] == 0 && arraymemoria[187] == 0
                && arraymemoria[198] == 0 && arraymemoria[184] == 0) {

                Toast.makeText(this,
                    "You are going on the right way!!Continue!!",
                    Toast.LENGTH_SHORT).show();

            } else {
                Toast.makeText(this, "Sorry, but something is wrong",
                    Toast.LENGTH_SHORT).show();
            }
            break;

        case 2:
```

```

        if (arraymemoria[171] == 0 && arraymemoria[0] == 0
            && arraymemoria[170] == 0 && arraymemoria[182] == 0
            && arraymemoria[183] == 1 && arraymemoria[184] == 0
            && arraymemoria[199] == 1 && arraymemoria[185] == 1
            && arraymemoria[186] == 0 && arraymemoria[187] == 0
            && arraymemoria[198] == 0 && contclk == 1) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }
        break;

    case 3:

        if (arraymemoria[171] == 0 && arraymemoria[0] == 0
            && arraymemoria[170] == 0 && arraymemoria[182] == 1
            && arraymemoria[183] == 1 && arraymemoria[184] == 0
            && arraymemoria[199] == 0 && arraymemoria[185] == 1
            && arraymemoria[186] == 0 && arraymemoria[187] == 0
            && arraymemoria[198] == 0 && contclk == 0) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }
        break;

    case 4:

        if (arraymemoria[171] == 1 && arraymemoria[0] == 1
            && arraymemoria[170] == 1 && arraymemoria[182] == 1
            && arraymemoria[183] == 1 && arraymemoria[184] == 0
            && arraymemoria[199] == 0 && arraymemoria[185] == 1
            && arraymemoria[186] == 0 && arraymemoria[187] == 0
            && arraymemoria[198] == 0 && contclk == 0) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }
        break;

    case 5:

        if (arraymemoria[171] == 0 && arraymemoria[0] == 1
            && arraymemoria[170] == 1 && arraymemoria[182] == 1
            && arraymemoria[183] == 1 && arraymemoria[184] == 0
            && arraymemoria[199] == 1 && arraymemoria[185] == 1
            && arraymemoria[186] == 0 && arraymemoria[187] == 0
            && arraymemoria[198] == 0 && contclk == 4) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }

        break;

    case 6:

        if (arraymemoria[171] == 0 && arraymemoria[0] == 0
            && arraymemoria[170] == 0 && arraymemoria[182] == 0

```

```

        && arraymemoria[183] == 1 && arraymemoria[184] == 0
        && arraymemoria[199] == 1 && arraymemoria[185] == 1
        && arraymemoria[186] == 1 && arraymemoria[187] == 1
        && arraymemoria[198] == 0 && (contclk == 4) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }
        break;
case 7:

    if (arraymemoria[171] == 0 && arraymemoria[0] == 1
        && arraymemoria[170] == 1 && arraymemoria[182] == 1
        && arraymemoria[183] == 1 && arraymemoria[184] == 0
        && arraymemoria[199] == 1 && arraymemoria[185] == 1
        && arraymemoria[186] == 0 && arraymemoria[187] == 0
        && arraymemoria[198] == 0 && contclk == 2) {

        Toast.makeText(this,
            "You are going on the right way!!Continue!!",
            Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "Sorry, but something is wrong",
            Toast.LENGTH_SHORT).show();
    }
    break;
case 8:

    if (arraymemoria[171] == 1 && arraymemoria[0] == 1
        && arraymemoria[170] == 1 && arraymemoria[182] == 1
        && arraymemoria[183] == 0 && arraymemoria[184] == 0
        && arraymemoria[199] == 0 && arraymemoria[185] == 1
        && arraymemoria[186] == 0 && arraymemoria[187] == 0
        && arraymemoria[198] == 0 && contclk == 0) {

        Toast.makeText(this,
            "You are going on the right way!!Continue!!",
            Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "Sorry, but something is wrong",
            Toast.LENGTH_SHORT).show();
    }
    break;
case 9:

    if (arraymemoria[171] == 1 && arraymemoria[0] == 1
        && arraymemoria[170] == 1 && arraymemoria[182] == 1
        && arraymemoria[183] == 1 && arraymemoria[184] == 0
        && arraymemoria[199] == 0 && arraymemoria[185] == 1
        && arraymemoria[186] == 1 && arraymemoria[187] == 1
        && arraymemoria[198] == 0 && contclk == 0) {

        Toast.makeText(this,
            "You are going on the right way!!Continue!!",
            Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "Sorry, but something is wrong",
            Toast.LENGTH_SHORT).show();
    }
    break;
case 10:

    if (arraymemoria[171] == 0 && arraymemoria[0] == 1
        && arraymemoria[170] == 1 && arraymemoria[182] == 1
        && arraymemoria[183] == 0 && arraymemoria[184] == 0
        && arraymemoria[199] == 0 && arraymemoria[185] == 1

```

```

        && arraymemoria[186] == 0 && arraymemoria[187] == 0
        && arraymemoria[198] == 0 && contclk == 0) {

            Toast.makeText(this,
                "You are going on the right way!!Continue!!",
                Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "Sorry, but something is wrong",
                Toast.LENGTH_SHORT).show();
        }
        break;
default:
    Toast.makeText(this, "Sorry, but can not correct the exercise",
        Toast.LENGTH_LONG).show();
    break;
}
}
}

```

d. Método pintarlineas

En este último método que se explica de esta clase, se realiza la lectura del archivo que contiene las coordenadas de las líneas dibujadas por el usuario, para ir añadiendo estas al array que posteriormente será pintado en el layout.

```

public void pintarlineas() {

    mCanvas = new Canvas();
    mPath = new Path();
    mPaint = new Paint();
    mPaint.setAntiAlias(true);
    mPaint.setDither(true);
    mPaint.setColor(Color.DKGRAY);
    mPaint.setStyle(Paint.Style.STROKE);
    mPaint.setStrokeJoin(Paint.Join.ROUND);
    mPaint.setStrokeCap(Paint.Cap.ROUND);
    mPaint.setStrokeWidth(5);

    RelativeLayout layMain = (RelativeLayout) this
        .findViewById(R.id.LayMain1);
    layMain.addView(new Mens(this));

    Bundle pintarlinea = getIntent().getExtras();
    int contadorlineas = pintarlinea.getInt("contadorlineas");

    File tarjeta = Environment.getExternalStorageDirectory();
    File file = new File(tarjeta.getAbsolutePath(), "graficos");
    try {
        FileInputStream fIn = new FileInputStream(file);
        InputStreamReader archivo = new InputStreamReader(fIn);
        BufferedReader br = new BufferedReader(archivo);
        String linea = br.readLine();
        br.close();
        archivo.close();

        int cantidaddecomas = 0;
        for (int i = 0; i < linea.length(); i++) {
            if (linea.charAt(i) == ',') {
                cantidaddecomas++;
            }
        }

        int[] cantidaddefloats = new int[cantidaddecomas + 1];

        int z = 0;
        for (String retval : linea.split(",")) {
            cantidaddefloats[z] = Integer.parseInt(retval);
            z++;
        }
    }
}

```

```

        int j = 0, cont = 0;

        for (j = 0; j < contadorlineas; j++) {
            x[j] = cantidaddefloats[cont];
            cont++;
        }

        for (j = 0; j < contadorlineas; j++) {
            y[j] = cantidaddefloats[cont];
            cont++;
        }

        for (j = 0; j < contadorlineas; j++) {
            x1[j] = cantidaddefloats[cont];
            cont++;
        }

        for (j = 0; j < contadorlineas; j++) {
            y1[j] = cantidaddefloats[cont];
            cont++;
        }

        mPath = new Path();
        for (i = 0; i < contadorlineas; i++) {
            mPath.moveTo(x[i], y[i]);
            mPath.lineTo(x1[i], y1[i]);
            _graphics.add(mPath);
        }
    } catch (IOException e) {
        Log.e("Ficheros", "Error al leer fichero desde memoria interna");
        Toast.makeText(this, "Sorry, but there is no file with graphics",
            Toast.LENGTH_LONG).show();
    }
    return;
}
}}

```

10.2 NewExercise1

En esta clase, además de desarrollarse algunos de los métodos de la clase NewExercise, se han implementado algunos más que se pasan a mostrar a continuación.

En esta clase, se implementa onTouchListener, ya que el usuario va a interactuar con el layout correspondiente y la actividad debe estar “escuchando” a los eventos de “touch” del usuario.

```
public class NewExercise1 extends Activity implements onTouchListener {
```

...

Se define el layout correspondiente y se le asocia el evento onTouchListener, también el elemento Canvas que se usará para pintar.

```

LinearLayout layMain = (LinearLayout) this.findViewById(R.id.layMain);
layMain.setOnTouchListener((onTouchListener) this);
layMain.addView(new Mens(this));

```

...

a. Método onTouch

En este método, que escucha los eventos del usuario al interactuar con la pantalla del dispositivo, se recogen las coordenadas de donde toca el usuario y se van añadiendo al elemento mPath, para posteriormente añadirlo al array que contendrá todas las coordenadas. Se diferencian las coordenadas del evento de presionar, con las coordenadas del evento levantar el dedo de la pantalla.

```
public boolean onTouch(View v, MotionEvent event) {

    if (parametrogeneral < 200 && parametrogeneral > 0) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                mPath = new Path();
                x[parametrogeneral] = (int) event.getX();
                y[parametrogeneral] = (int) event.getY();
                mPath.moveTo(x[parametrogeneral], y[parametrogeneral]);
                break;
            case MotionEvent.ACTION_UP:
                x1[parametrogeneral] = (int) event.getX();
                y1[parametrogeneral] = (int) event.getY();
                mPath.lineTo(x1[parametrogeneral], y1[parametrogeneral]);
                _graphics.add(mPath);
                parametrogeneral++;
                contadorlineas = parametrogeneral;
                break;
        }
    } else {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                mPath = new Path();
                x[contadorlineas] = (int) event.getX();
                y[contadorlineas] = (int) event.getY();
                mPath.moveTo(x[contadorlineas], y[contadorlineas]);
                break;
            case MotionEvent.ACTION_UP:
                x1[contadorlineas] = (int) event.getX();
                y1[contadorlineas] = (int) event.getY();
                mPath.lineTo(x1[contadorlineas], y1[contadorlineas]);
                _graphics.add(mPath);
                contadorlineas++;
                break;
        }
    }

    return true;
}
```

b. Clase Mens

Ya se explicó anteriormente.

```
public class Mens extends View {

    public Mens(Context context) {
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        for (Path mPath : _graphics) {
            canvas.drawPath(mPath, mPaint);
        }

        invalidate();
    }
}
```

c. Método enable

En este método, al presionar sobre el botón que lanza la actividad NewExercise, se realizan las funciones necesarias para guardar el archivo que contiene todas las coordenadas de todas las líneas que el usuario dibujó.

```
public void enable(View button2) {

    Bundle enablebutton = getIntent().getExtras();

    int[] arraymemoria = enablebutton.getIntArray("arraymemoria");
    int[] comprobacion1 = enablebutton.getIntArray("comprobacion1");
    int[] comprobacion2 = enablebutton.getIntArray("comprobacion2");
    int[] comprobacion3 = enablebutton.getIntArray("comprobacion3");
    int[] comprobacion4 = enablebutton.getIntArray("comprobacion4");
    int[] comprobacion5 = enablebutton.getIntArray("comprobacion5");
    int contadorlineas1 = enablebutton.getInt("contadorlineas1");
    int identificativo = enablebutton.getInt("identificativo");

    String temp = Arrays.toString(x).replace(", ", ",");
    String[] probando = new String[200];
    int l = 0;
    for (String retval : temp.split(",0,")) {
        probando[l] = retval;
        l++;
    }
    String encoded = probando[0].substring(1, probando[0].length());

    String temp1 = Arrays.toString(y).replace(", ", ",");
    l = 0;
    for (String retval : temp1.split(",0,")) {
        probando[l] = retval;
        l++;
    }
    String encoded1 = probando[0].substring(1, probando[0].length());

    String temp2 = Arrays.toString(x1).replace(", ", ",");
    l = 0;
    for (String retval : temp2.split(",0,")) {
        probando[l] = retval;
        l++;
    }
    String encoded2 = probando[0].substring(1, probando[0].length());

    String temp3 = Arrays.toString(y1).replace(", ", ",");
    l = 0;
    for (String retval : temp3.split(",0,")) {
        probando[l] = retval;
        l++;
    }
    String encoded3 = probando[0].substring(1, probando[0].length());
    try {

        File tarjeta = Environment.getExternalStorageDirectory();
        File file = new File(tarjeta.getAbsolutePath(), "graficos");
        OutputStreamWriter osw = new OutputStreamWriter(
            new FileOutputStream(file));

        String concatenacionencoded = (encoded + "," + encoded1 + ","
            + encoded2 + "," + encoded3);

        osw.write(concatenacionencoded);
        osw.flush();
        osw.close();
        Toast.makeText(this, "The file has been saved correctly",
            Toast.LENGTH_SHORT).show();

    } catch (Exception ex) {
        Log.e("Ficheros",
            "Error al escribir fichero en memoria interna");
        Toast.makeText(
```

```

        this,
        "Sorry but is not possible save the file",
        Toast.LENGTH_LONG).show();

    }

    int parametro = 1;
    Intent intent = new Intent();
    intent.setComponent(new ComponentName(this, NewExercise.class));
    intent.putExtra("arraymemoria", arraymemoria);
    intent.putExtra("comprobacion1", comprobacion1);
    intent.putExtra("comprobacion2", comprobacion2);
    intent.putExtra("comprobacion3", comprobacion3);
    intent.putExtra("comprobacion4", comprobacion4);
    intent.putExtra("comprobacion5", comprobacion5);
    intent.putExtra("param", parametro);
    intent.putExtra("identificativo", identificativo);

    if (contadorlineas1 < 200 && contadorlineas1 > 0 && contadorlineas == 0) {
        intent.putExtra("contadorlineas", parametrogeneral);
    } else
        intent.putExtra("contadorlineas", contadorlineas);

    finish();
    startActivity(intent);
}

```

d. Método borrar

En este método, al presionar sobre el botón de “Clear all”, se dan valor nulo a todas las coordenadas y así se borran todas las líneas que habían sido dibujadas.

```

public void borrar(View button07) {
    _graphics.clear();
    contadorlineas = 0;
    parametrogeneral = 0;
    int r=0;
    for (r=0;r<x.length;r++)
        x[r] = 0;
    for (r=0;r<y.length;r++)
        y[r] = 0;
    for (r=0;r<x1.length;r++)
        x1[r] = 0;
    for (r=0;r<y1.length;r++)
        y1[r] = 0;

}
}

```

10.3 Clase SendEmailActivity

En esta clase se implementa la parte del código que hace que sea posible enviar los archivos por correo electrónico.

```
public class SendEmailActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_enviar_mail);

        final Bundle extras = getIntent().getExtras();

        Button btnSend = (Button) findViewById(R.id.buttonSend);
        btnSend.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {

                CheckBox chkAttachment = (CheckBox)
                findViewById(R.id.chkAttachment);

                Intent email = new Intent(
                    android.content.Intent.ACTION_SEND_MULTIPLE);

                email.putExtra(android.content.Intent.EXTRA_EMAIL,
                    new String[] { "angel.guerra.martin87@gmail.com"
                });

                email.putExtra(android.content.Intent.EXTRA_SUBJECT,
                    "Exam");
                email.putExtra(android.content.Intent.EXTRA_TEXT,
                    "Here's my exam");

                if (chkAttachment.isChecked()) {

                    String nomarchivo = extras.getString("nomarchivo");
                    String nomarchivo1 = "graficos";

                    String externalStoragePathStr = Environment
                    .getExternalStorageDirectory().toString()
                    + File.separatorChar;
                    String filePaths[] = { externalStoragePathStr +
                    nomarchivo,
                    externalStoragePathStr + nomarchivo1 };

                    ArrayList<Uri> uris = new ArrayList<Uri>();
                    for (String file : filePaths) {
                        File fileIn = new File(file);
                        Uri u = Uri.fromFile(fileIn);
                        uris.add(u);
                    }
                    email.putParcelableArrayListExtra(
                        Intent.EXTRA_STREAM, uris);
                    email.setType("*/*");
                    email.setClassName("com.google.android.gm",
                    "com.google.android.gm.ComposeActivityGmail");
                }
                startActivity(Intent.createChooser(email,
                    "Choose an Email client :"));
            }
        });
    }
}
```

10.4 Clase ContinueExercise

En esta clase se desarrolla un método con el que se lee de la tarjeta SD, carpeta “Downloads”, el archivo con el nombre que el usuario escriba en pantalla, siempre que éste exista. Una vez que lo ha leído, la información obtenida se guarda en un parámetro que posteriormente será pasado a otra clase para que sea tratado y pueda incorporar dicha información al layout de la FPGA, se pinten los dibujos y se añada el contenido de los CLBs a los CLBs correspondientes.

```
public class ContinueExercise extends Activity {
    EditText et1;
```

a. Método recuperar

```
public void recuperar(View button1) {

    String nomarchivo = et1.getText().toString();
    File tarjeta = Environment.getExternalStorageDirectory();
    File file = new File(tarjeta.getAbsolutePath() + "/" + "Download", nomarchivo);
    try {
        FileInputStream fIn = new FileInputStream(file);
        InputStreamReader archivo = new InputStreamReader(fIn);
        BufferedReader br = new BufferedReader(archivo);
        String linea = br.readLine();

        br.close();
        archivo.close();
        Intent intent = new Intent();
        intent.setComponent(new ComponentName(this,
            ContinuandoExercise.class));
        intent.putExtra("todo", linea);
        finish();

        startActivity(intent);

    } catch (IOException e) {
        Log.e("Ficheros", "Error al leer fichero desde memoria interna");
        Toast.makeText(
            this,
            "Sorry, but there is no file with that name, you must
write one file's name that exists",
            Toast.LENGTH_LONG).show();

        return;
    }
}
}
```

10.5 Clase GuardarEjercicio

```
public class GuardarEjercicio extends Activity {  
...
```

a. Método grabar

En este método, toda la información que se ha ido añadiendo a la aplicación, ya sea la habilitación de los checkbox o el contenido de los CLBs, se guardará en un archivo con el nombre que el usuario le dé, en la tarjeta SD.

```
public void grabar(View v) {  
    nomarchivo = et1.getText().toString();  
    Bundle extras = getIntent().getExtras();  
    int[] arraymemoria = extras.getIntArray("arraymemoria");  
    int[] comprobacion = extras.getIntArray("comprobacion");  
  
    String temp = Arrays.toString(arraymemoria).replace(", ", " ");  
    String encoded = temp.substring(1, temp.length() - 1);  
  
    String temp1 = Arrays.toString(comprobacion).replace(", ", " ");  
    String encoded1 = temp1.substring(1, temp1.length() - 1);  
  
    String concatenacionencoded = (encoded + " " + encoded1);  
  
    try {  
        File tarjeta = Environment.getExternalStorageDirectory();  
        File file = new File(tarjeta.getAbsolutePath(), nomarchivo);  
        OutputStreamWriter osw = new OutputStreamWriter(  
            new FileOutputStream(file));  
  
        osw.write(concatenacionencoded);  
        osw.flush();  
        osw.close();  
        Toast.makeText(this, "The file is saved correctly",  
            Toast.LENGTH_SHORT).show();  
        et1.setText("");  
    } catch (Exception ex) {  
        Log.e("Ficheros", "Error al escribir fichero en memoria interna");  
        Toast.makeText(  
            this,  
            "Sorry, but you must write a name",  
            Toast.LENGTH_LONG).show();  
    }  
    guardarfigurita=1;  
    return;  
}
```


11 PRESUPUESTO

1) Ejecución Material

▪ Compra de ordenador personal (Software incluido)	1.500 €
▪ Dispositivo Tablet para test	500 €
▪ Material de oficina	50 €
▪ Total de ejecución material	2.050 €

2) Honorarios Proyecto

▪ 900 horas a 15 € / hora	13.500 €
---------------------------	----------

3) Material fungible

▪ Gastos de impresión	150 €
▪ Encuadernación	50 €

4) Subtotal del presupuesto

▪ Subtotal Presupuesto	15.750 €
------------------------	----------

5) I.V.A. aplicable

▪ 21 % Subtotal Presupuesto	3.307,5 €
-----------------------------	-----------

6) Total Presupuesto

▪ Total Presupuesto	19.057,5 €
---------------------	------------

Madrid, Septiembre 2013

El Ingeniero Jefe de Proyecto

Fdo.: Ángel Guerra Martín

Ingeniero Superior de Telecomunicación

12 PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un "Aplicación Android para simular una FPGA educativa". En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales.

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.
2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.
3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.
4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.
5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.
6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.
8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.
9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.
10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.
11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.
12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.
13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.
15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.
16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.
17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.
18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.
19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.
20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.
21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.
22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares.

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.
2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.
3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.
4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.
5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.
6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.
7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.
8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.
9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.
11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.
12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.