UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

Ingeniería de Telecomunicación

Tutorial Interactivo Android sobre Mapeado en FPGAs

Adrián Moreno Villalón

Abril 2016

Tutorial Interactivo Android sobre Mapeado en FPGAs

AUTOR: Adrián Moreno Villalón
TUTOR: Eduardo Boemo Scalvinoni

TEC
Depto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Abril de 2016

Resumen

En la era en la que los dispositivos inteligentes están llamados a superar cualquier otra forma de computación inteligente, este proyecto trata de aprovechar estos medios para ofrecer a los estudiantes de ingeniería una plataforma multimedia con fines educativos dando lugar a una aplicación Android. Dicha aplicación es posible ejecutarla tanto sobre una tableta o un móvil (preferiblemente de pantalla de 4.7'' o superior) con sistema operativo Android. En ella se puede visualizar una maqueta de una FPGA básica sobre la que poder realizar los ejercicios oportunos para conseguir entender la metodología final. La aplicación consta de 9 ejercicios tipo que el usuario puede manipular y mapear en la pantalla base de FPGA dando una visión básica de los conceptos principales de una arquitectura FPGA. Del mismo modo, durante la resolución de los ejercicios, el usuario puede guardar el ejercicio para seguir con el más adelante o enviarlo por correo para que sea corregido o le surge alguna duda.

Palabras clave

Dispositivos inteligentes, aplicación Android, FPGA, Interconection Box, Logic Element, multiplexor, puertas lógicas.

Abstract

At the present time, smart devices are called to overcome any other form of intelligent computing, this project try to take advantage of these ways to give engineering students a multimedia platform for educational purposes and the final result is this Android application. The application can run either on a tablet or a mobile (preferably screen 4.7 " or higher) with Android operating system. It can display a model of a basic FPGA on which to perform the appropriate exercises for the right understanding of the main ideas of the subject. The application consists of 9 type exercises that the user can manipulate and map at the base FPGA screen giving a basic overview of the main concepts of an FPGA architecture. In the same way, during the resolution of the exercises, the user can save the exercise to continue the later or mail to be corrected or concerns you experience.

Key Words

Smart devices, Android, FPGA, Interconnection Box, Logic Element, multiplexer, logic gates.

INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES

1 Introducción

1.1 Motivación

Parándose a pensar y analizando cómo avanza la tecnología en la sociedad donde cada vez es más necesario y está más difundido el tener un dispositivo inteligente con el fin de ayudarse de ellos para hacernos la vida más fácil, poniéndonos a nuestro alcance múltiples opciones sin tener que movernos de nuestro lugar, como puede ser administrar nuestros gastos con aplicaciones de bancos o diferentes aplicaciones que agrupan gastos, para realizar compras de un modo cómodo y sencillo o simplemente por ocio donde multitud de juegos de diferentes categorías copan el mercado de aplicaciones móviles, la educación no se queda atrás en este avance tecnológico y tiene la posibilidad de ofrecer una amplia gama de aplicaciones que permitan al usuario aprender de una manera amena y entretenida e incluso personalizada en muchos casos. Cientos de aplicaciones para aprender idiomas, para realizar ejercicio o para aprender a leer y escribir en el caso de los niños, están en las tiendas de aplicaciones de los principales sistemas operativos. Por ello es cada vez más usual ver como los centros se dotan de tabletas para dar el temario o de portales internet para interactuar entre alumnos/padres/tutores.

Dado que hoy en día se ha cambiado el cuaderno y el papel por los archivos en PDF, las transparencias y los libros digitales, se intenta dar un paso más y aprovechar el mercado de las tabletas para ayudar a entender la constitución de una FPGA de manera sencilla en forma de aplicación.

Además el interactuar con la aplicación nos permitirá agregar consejos para la resolución de ejercicios, guías de ejercicios, tutoriales para entender cómo funciona una FPGA, posibilidad de enviar desde dicha aplicación cualquier duda al profesor…

Por otro lado existen otras ventajas al utilizar un mercado como el de las tabletas para la expansión de correcciones y modificaciones tanto de la guía de ejercicios, como de su solución, los distintos tutoriales o incluso notas informativas acerca del mercado de las FPGAS. Ya que, de esta manera cualquier estudiante interesado puede en cualquier parte del mundo tener acceso a esta información gracias a la distribución por Google Play y poder realizar ejercicios en cualquier parte con un fácil acceso y fácil manejo sin la necesidad de cargar con libros ni apuntes.

1.2 Objetivos

El objetivo principal de esta aplicación es la comprensión y la aclaración de los conceptos básicos por parte del usuario de una arquitectura FPGA mediante la práctica de ejercicios de una manera amena y sobretodo visual, acorde con las tecnologías del momento, así como la posibilidad por parte del profesor de crear nuevos ejercicios, publicar aclaraciones, consejos, opiniones, y mejorar la interactuación con el alumno mediante el envío de ejercicios con las dudas pertinente además de dar la posibilidad de evaluar de manera continua al alumnado mediante el envío de ciertos ejercicios y la posterior corrección.

Para ello se ha desarrollado una aplicación en la cual el usuario puede realizar distintos ejercicios elegidos de manera concreta por el profesor, donde se destacan los principales aspectos y funciones de una FPGA para el correcto entendimiento del mismo. Para la resolución de ejercicios puede hacer uso de los siguientes componentes:

- Circuito simplificado y esquematizado de una FPGA.
- Mapeo de circuitos combinacionales con multiplexores.
- Conexión de las distintas pistas por medio de las *Interconection Box*.
- Mapeo de circuitos secuenciales sencillos.
- Utilización de los *Logic Element*.
- Anexo de una ayuda para la óptima utilización de la aplicación.
- Guardar y cargar el ejercicio cada vez que el usuario desee.
- Enviar el ejercicio resuelto al profesor para su corrección.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:** Se resume el objetivo del proyecto.

- **Estado del arte:** Se establece el marco teórico en el que se encuentra este PFC dentro del mundo de las aplicaciones educativas que hay disponibles en el mercado para los diferentes ámbitos.

- **Diseño y Desarrollo de la aplicación:** Se explica la especificación del proyecto y su diseño así como su implementación. Se detalla los distintos módulos implementados, las principales funciones utilizadas con extractos de código que forman parte de la aplicación además de los distintos parámetros utilizados.

- **Conclusiones:** Se detalla la finalidad del proyecto, donde y como será usará la aplicación. Además se proporcionara el plan de pruebas utilizado para el diseño de la aplicación y los principales valores aprendidos durante el diseño y la implementación de dicha aplicación.

- **Apéndice:**

  - **Manual de Usuario.** Se define el manual del usuario para que este pueda ser autosuficiente y sacar todo el partido a la aplicación.

  - **Ejemplos.** Se realizan pruebas con los un ejercicio ejemplo que servirá de guía para el usuario.

  - **Trabajo futuro.** En el trabajo futuro se deja la puerta abierta a las posibles mejoras para de la aplicación dando opiniones e ideas para el desarrollo de la aplicación que no han podido llevarse a cabo ya sea por falta de tiempo o por la incompatibilidad con otras partes del proyecto

- **Anexos:** Serie de anexos referentes a información complementaria acerca de los capítulos del proyecto.

2 Estado del arte

2.1 Introducción

Hoy en día, el alcance de los Smartphone es mayor que nunca, y su continua adopción está condicionando tanto el comportamiento de los consumidores como las estrategias de negocio. Estos dispositivos inteligentes son omnipresentes y, de hecho, podemos comprobar como simplemente en nuestro país hay más dispositivos móviles que personas.

La industria móvil atraviesa un momento de gran impulso y escala, en mayor parte debido a la multitud de tareas que puede realizar. Este motivo ha llevado a una revolución en nuestra manera de comunicarnos, haciendo casi imprescindible tener un dispositivo móvil en nuestras vidas. El uso del mismo ha cambiado hábitos como el de conexión a Internet o el uso de las aplicaciones que nos facilitan la vida. Nos conectamos desde cualquier lugar a través del dispositivo que podemos transportar fácilmente en nuestros bolsillos.

Tanto es así que todos los campos de negocio se están abriendo al mundo de las aplicaciones y comercio online para llegar a más usuarios. En el ámbito educacional también se está aprovechando las facilidades didácticas que presentas los dispositivos móviles para sacar el mayor provecho, y es por ello que este proyecto intenta tomar ventaja de esa demanda de aplicaciones que presenta la sociedad para facilitar el estudio y comprensión de la asignatura.

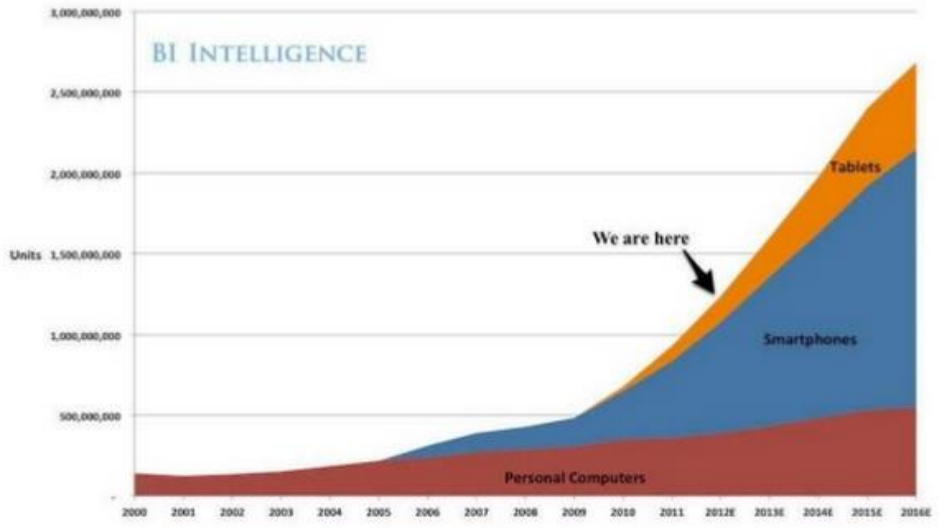

Figura 2.1 Evolución de los dispositivos inteligentes [5]

2.1.1 Aplicaciones

Desde la existencia de Android ha habido una gran evolución de todo lo que rodea al sistema operativo de Google, son muchísimos los desarrolladores que han ido añadiendo más y más aplicaciones a través de la tienda oficial de Google Play.

Para poder desarrollar una aplicación Android necesitamos elegir y prepara un entorno de desarrollo. En este aspecto existen dos alternativas, o programar una aplicación nativa o hacer uso de los diferentes frameworks o aplicaciones gratuitas que existen hoy en día que permiten hacer aplicaciones móviles sin apenas conocimientos de programación.

Basándose en el desarrollo de una aplicación sin el uso de framework ni aplicaciones, hay que tener en cuenta que Android también ofrece una completa interfaz de usuario, aplicaciones, bibliotecas de código, estructuras para aplicaciones y compatibilidad multimedia para su uso.

Además, aunque los componentes del sistema operativo se escriban en lenguajes de programación como C o C++, las aplicaciones para Android se diseñan en Java. De esta manera, como Java es también Open Source al igual que Android, podemos desarrollar lo que se quiera sin costes asociados y de una manera más fácil, aunque también se pueden utilizar otros lenguajes de programación.

Para ello lo primero será configurar un entorno de desarrollo que se basará en utilizar un IDE o entorno de desarrollo integrado (como en nuestro caso Eclipse) en el que se deberán instalar la JDK (Java Development Kit) o kit de desarrollo para Java correspondiente, el SDK (Software Development Kit) y ADT (Android Development Tools) de Android para poder hacer uso de las herramientas y librerías propias de Android.

A partir de ahí, se aconseja configurar un emulador de Android dentro del entorno para no tener que instalar las distintas versiones de la aplicación en el propio terminal cada vez que se quiera realizar una prueba. También se puede tener conectado el móvil al PC a través del USB en modo "Depuración USB" y por lo tanto cada vez que se lance la aplicación se podrá elegir que se instale también la aplicación en el móvil, pudiendo hacer las pruebas correspondientes sobre el mismo terminal.

Una vez se haya desarrollado la aplicación, si queremos instalarla en cualquier terminal se hará a través del archivo comprimido “APK”, el cual se instala en el dispositivo una vez se ejecute.

Si además se quiere vender la aplicación, se deberá dar de alta como desarrollador en la tienda de Google Play y subir la aplicación para que sea revisada y aprobada para su publicación. A partir de ahí Google establecerá un contrato con unas condiciones para su comercialización y se podrá ver como la aplicación puede ser descargada por miles de usuarios.

2.1.2 Aplicaciones educativas sobre electrónica

Digital Electronics 101

Aplicación donde se exponen preguntas de opción múltiple con soluciones y teoría. Con ello se pretende revisar los fundamentos de la electrónica digital, tratar las preguntas y cálculos y poder comprobar la solución. Adaptarse a los estudiantes de electrónica digital, técnicos automotrices, técnicos electrónicos, ingenieros eléctricos, ingenieros electrónicos y los estudiantes de física.

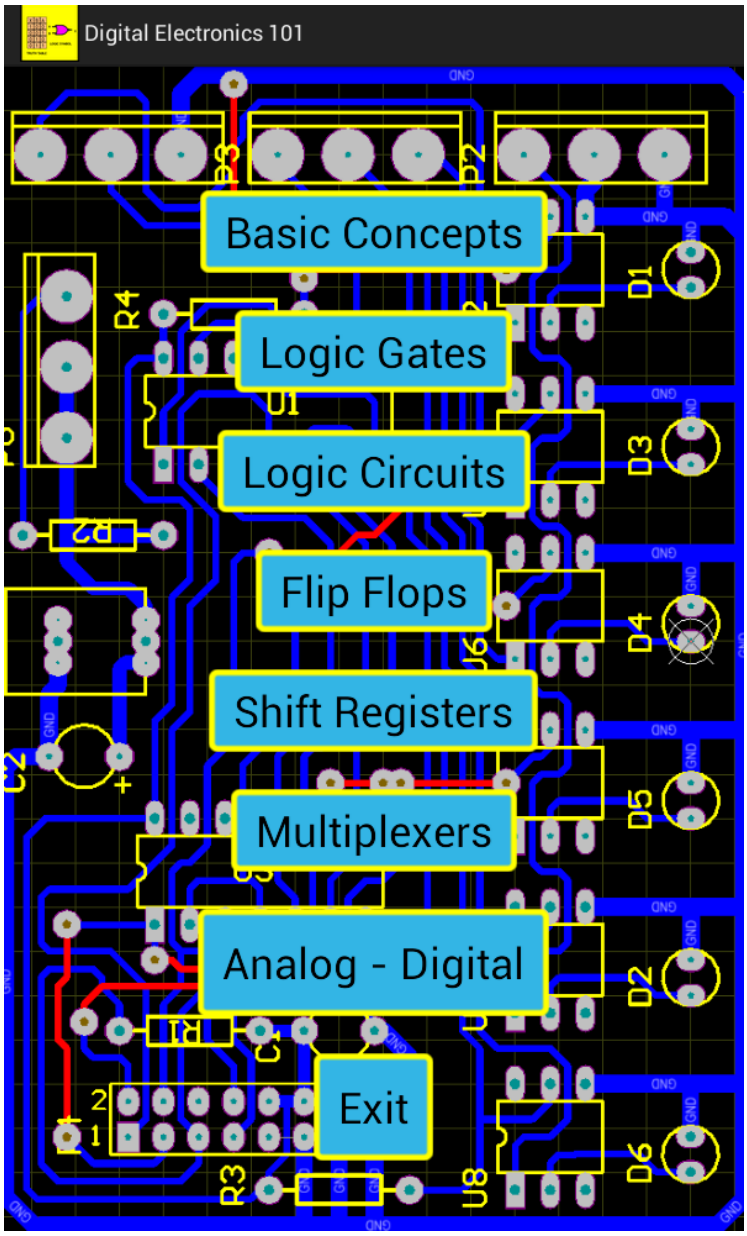

Figura 2.2 Pantalla principal de la aplicación Digital Electronics 101

En la pantalla principal nos exponen los temas de los que podemos ponernos a prueba y evaluar nuestros conocimientos. Según pulsemos un tema y otro, la aplicación nos propondrá una veinte preguntas con cuatro posibles respuestas cada una.

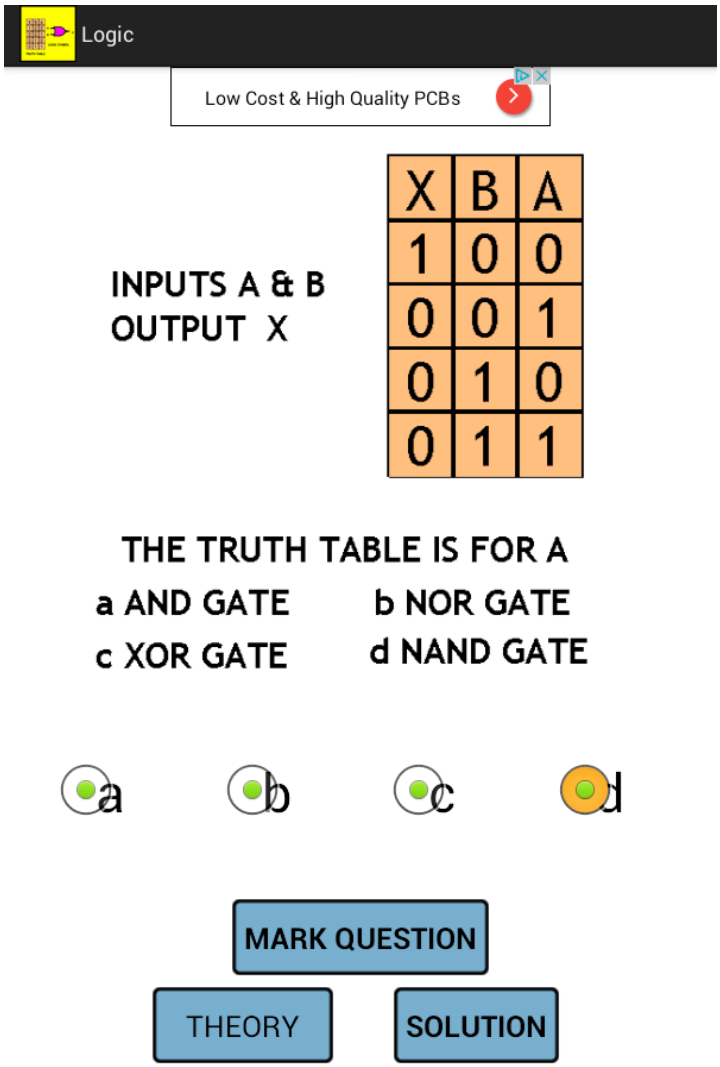

Figura 2.3 Pregunta sobre puertas lógicas de la aplicación Digital Electronics 101

En la misma pregunta, la aplicación te ofrece la posibilidad de revisar la teoría correspondiente a esa pregunta dando una explicación de la respuesta de la misma.

Aspectos positivos y negativos de esta aplicación:

- ✓ Disponible Off-Line.
- ✓ Posibilidad de comprobar conocimientos mediante preguntas.
- ✓ Explicación teórica de cada pregunta.

- × No permite rotación de pantalla.
- × Diseño poco atractivo.
- × No ofrece la posibilidad de interactuar.
- × Teoría demasiado escasa.

Electronic Components

Esta aplicaciones es muy eficiente para conocer el mundo electrónico a través de algunos pequeños componentes, como por ejemplo: Ic, Diodo, Registro, Transformador, Chip, etc...Se ha tratado de mostrar el componente en sí y explicar su funcionamiento.

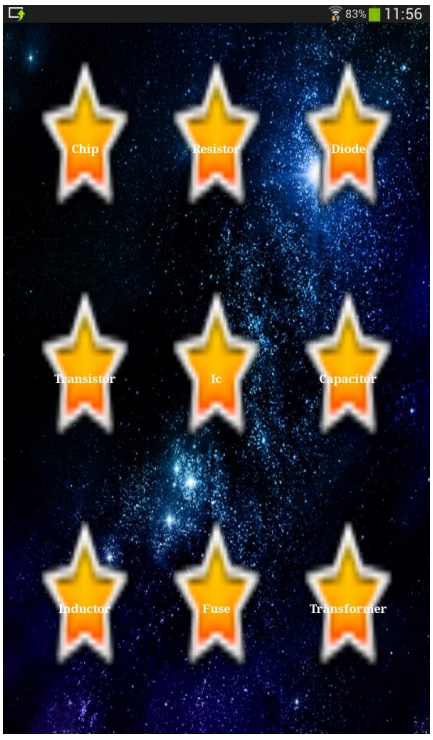

Figura 2.4 Pantalla principal de la aplicación Electronic Components

En la pantalla principal nos aparecen los distintos componentes que ofrece la aplicación, estos son: chip, resistencias, diodos, transistores, Ics, condensadores, bobinas, fusibles y transformadores.

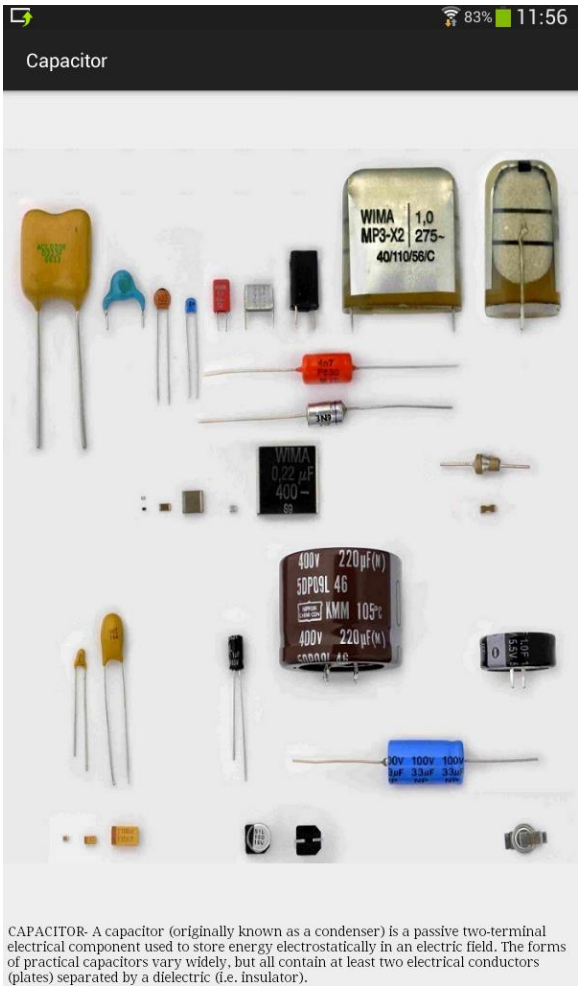

Figura 2.5 Ejemplo y explicación que muestra la aplicación sobre condensadores.

Según vayamos pulsando sobre cada componente nos aparece una imagen ilustrativa sobre los distintos tipos de ese componente junto con una breve descripción.

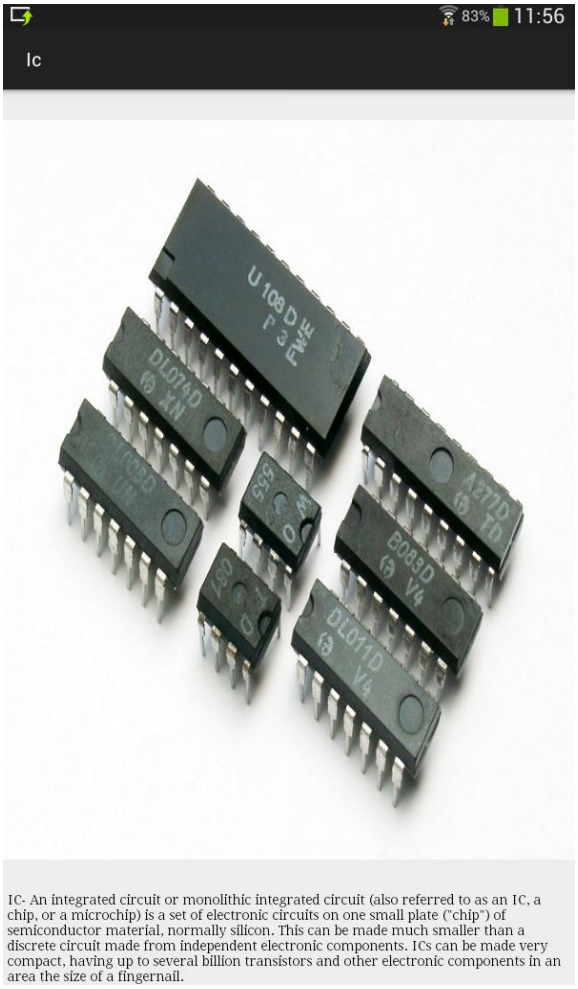

Figura 2.6 Ejemplo y explicación que muestra la aplicación sobre condensadores.

Aspectos positivos y negativos de esta aplicación:

- ✓ Disponible Off-Line.
- ✓ Resumen teórico de pequeños componentes.
- ✓ Permite rotación de pantalla.

- ✗ Demasiado simple.
- ✗ Diseño poco atractivo.
- ✗ No ofrece la posibilidad de interactuar.

DiCiDe: Circuitos digitales

Esta aplicación te permite resolver sistemas combinacionales de hasta 8 variables y 8 salidas por Quine-McCluskey de una manera totalmente interactiva.

La aplicación te permite pasar desde las tablas de la verdad hasta el circuito simulado pasando por el mapa de Karnaugh.

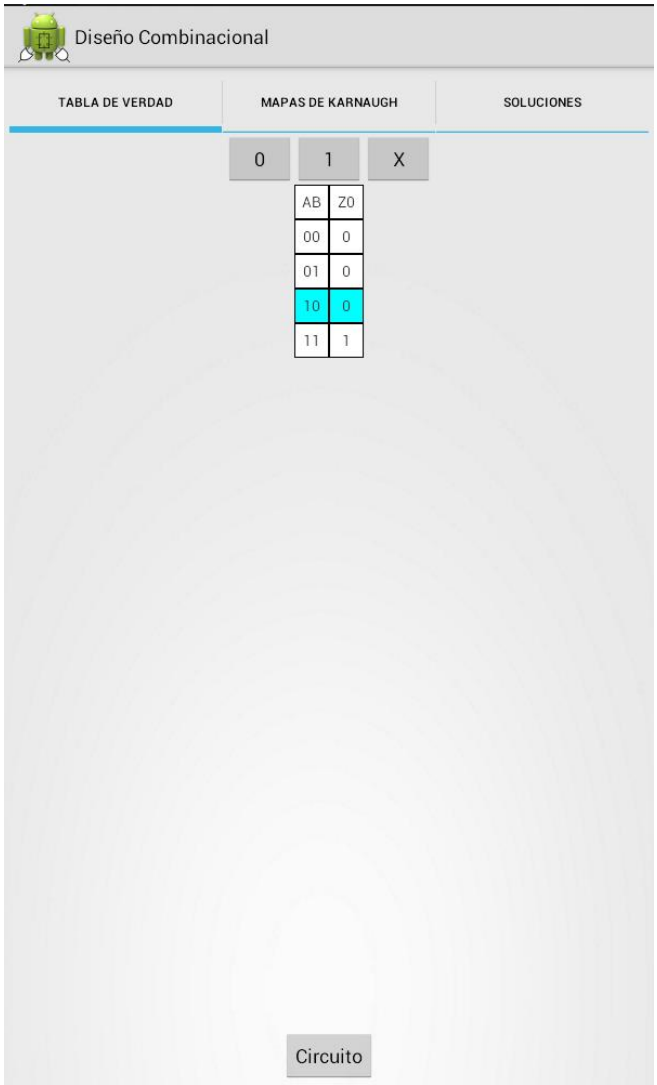

Figura 2.7 Pantalla principal de la aplicación DiCiDe.

Además muestra los lazos en mapas de Karnaugh, muestra las soluciones en forma algebraica booleana e incluso permite hacer un dibujo esquemático.

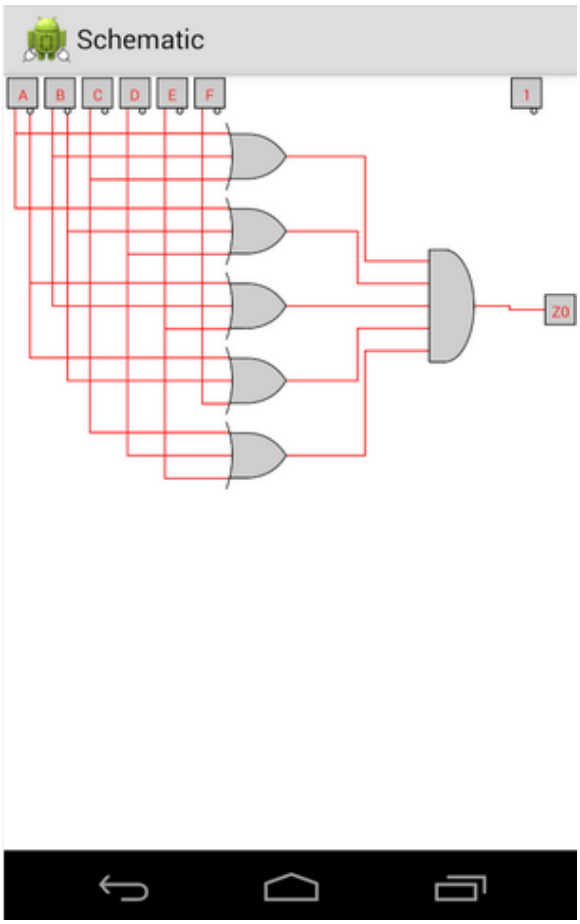

Figura 2.8 Aplicación DiCiDe

También permite diseñar sistemas de hasta 5 entradas y 5 salidas: máquinas de Moore y Mealy, diagrama de estados, tablas de codificación de estados y excitación de biestables

Aspectos positivos y negativos de esta aplicación:

- ✓ Permite realizar circuitos sin exceso de cálculos.
- ✓ Permite simular el circuito creado.
- ✓ Totalmente interactuable.
- ✓ Intuitiva y de fácil comprensión.

- ✗ Falta de teoría.

Curso de electrónica

El curso de electrónica gratis explica con detalle el funcionamiento y aplicación de los principales componentes y dispositivos electrónicos, así como las técnicas empleadas para verificar su operatividad. Aprender los principios de la electrónica analógica conociendo los componentes básicos de los circuitos electrónicos y sus parámetros.

En la pantalla principal nos muestra todos los temas disponibles en la aplicación.

También es en esta pantalla donde se nos ofrece la posibilidad de cambiar de idioma, a elegir entre: inglés, francés, alemán, italiano, portugués, ruso o español.

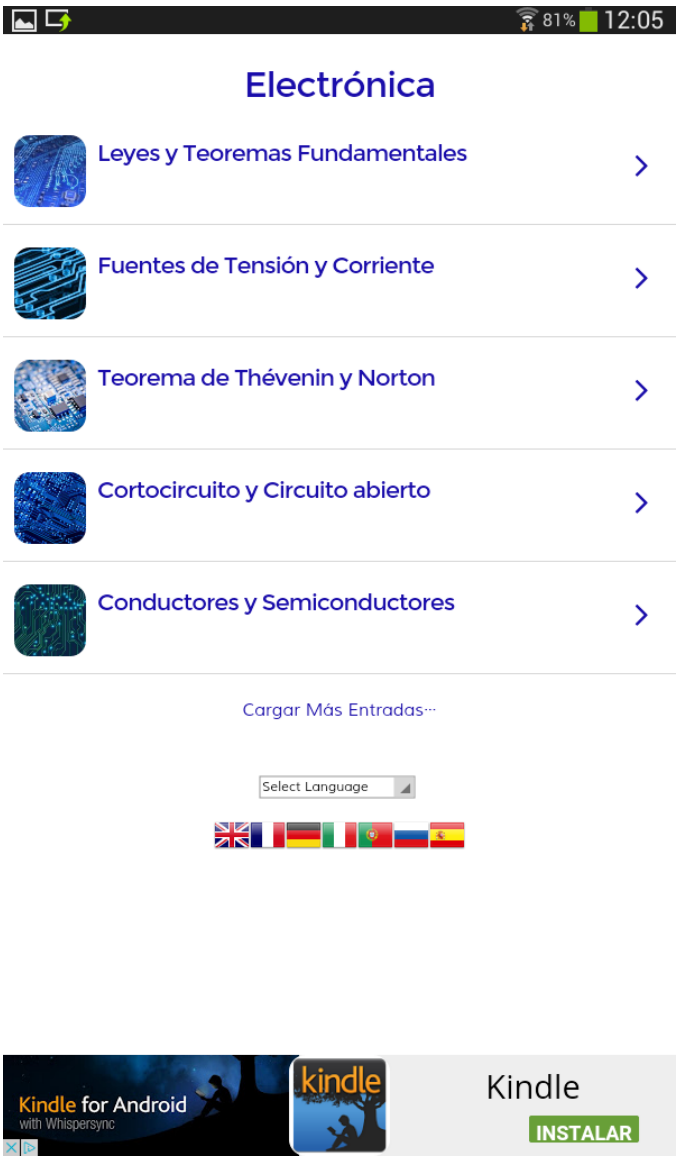

Figura 2.9 Pantalla principal de la aplicación Curso de electrónica.

Cuando hemos elegido el tema por el que queremos interesarnos o estudiar, la aplicación nos lleva a una nueva pantalla dónde se nos explica detalladamente el tema a tratar. Si es necesario la aplicación se ayuda de ejercicios ejemplo o de dibujos para entender mejor la explicación.

Figura 2.10 Explicación teórica sobre los teoremas fundamentales de la electrónica.

Aspectos positivos y negativos de esta aplicación:

- ✓ Disponible Off-Line.
- ✓ Disponible en varios idiomas.
- ✓ Permite rotación de pantalla.
- ✓ Imágenes explicativas sobre cada tema.
- ✓ Ejercicios ejemplo.

- × No ofrece la posibilidad de interactuar.
- × Conocimientos demasiado básicos.
- × No ofrece ejercicios para realizar.

Logic Maker

LogicMaker es una aplicación de diseño de circuitos digitales educativos para smartphones y tablets Android. Se puede utilizar para construir circuitos lógicos de puertas, registros, víboras y otros bloques de construcción. Esta aplicación permite construir circuitos combinacionales simples y CPUs completas.

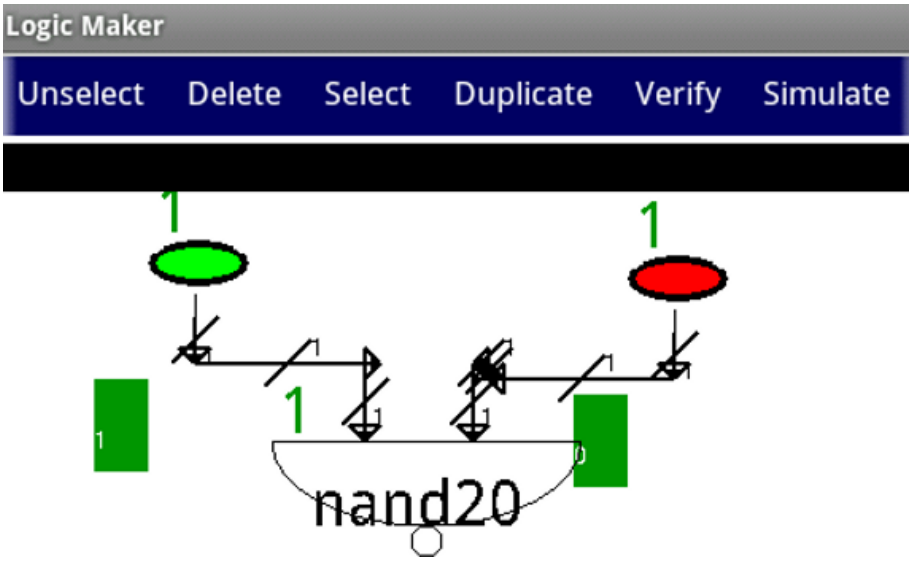

Figura 2.11 Aplicación Logic Maker

Aspectos positivos y negativos de esta aplicación:

- ✓ Permite simular el circuito creado.
- ✓ Totalmente interactuable.

- × Poco intuitiva.
- × Falta de tutorial.
- × Bastante lenta debido a su gran capacidad.

Logic Simulator Pro

Esta aplicación te ofrece la posibilidad de construir circuitos con puertas lógicas y varios otros componentes para simular sus funciones. Con su interfaz simple y fácil de usar que no toma mucho tiempo para aprender cómo usarlo, ideal para estudiantes que está aprendiendo cómo funciona el circuito lógico.

Lógica simulador Pro contiene varias características útiles, tales como:

- Las puertas lógicas (AND, OR, NAND, NOT, NOR, XOR, XNOR)
- Botones, Demostradores, Lámparas, relojes.
- Multiplexores.

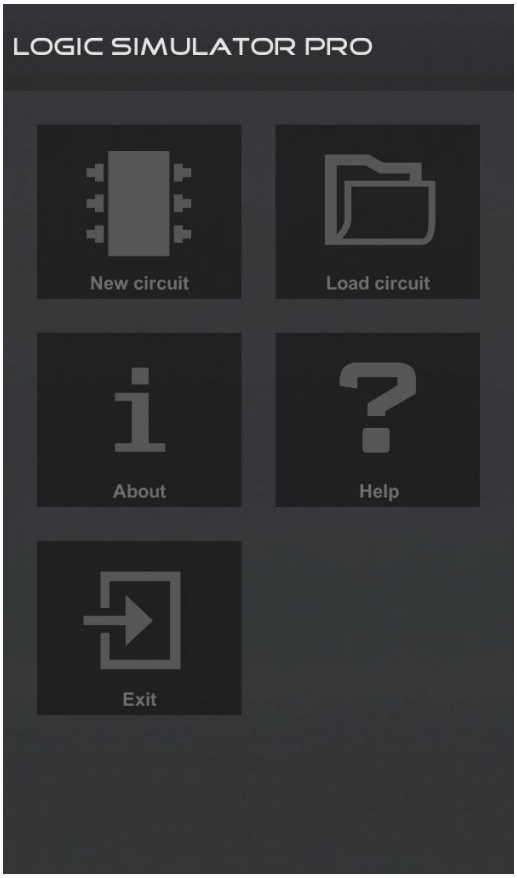

Figura 2.12 Pantalla principal de la aplicación Logic Simulator Pro.

La interfaz de la aplicación es simple e intuitiva, de manera que solo tienes que pulsar para realizar un nuevo circuito o cargar uno ya existente. Esta idea ha sido muy útil a la hora de diseñar el proyecto ya que ha ayudado a simplificar ideas.

Una vez elegimos diseñar un nuevo circuito, debemos asignarle un nombre que cumpla una serie de características para poder guardarlo.

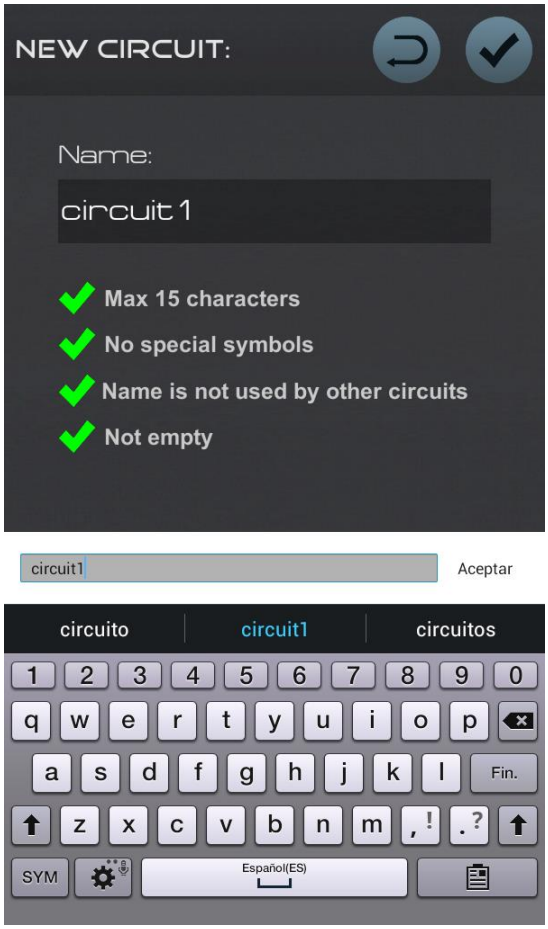

Figura 2.13 Pantalla de selección de nombre al circuito.

Simplemente debemos pulsar el botón de añadir componente para que nos aparezca un menú con todos los componentes disponibles clasificados por tipos. De todos ellos elegimos los que nos sean necesarios y así podremos formar el circuito deseado.

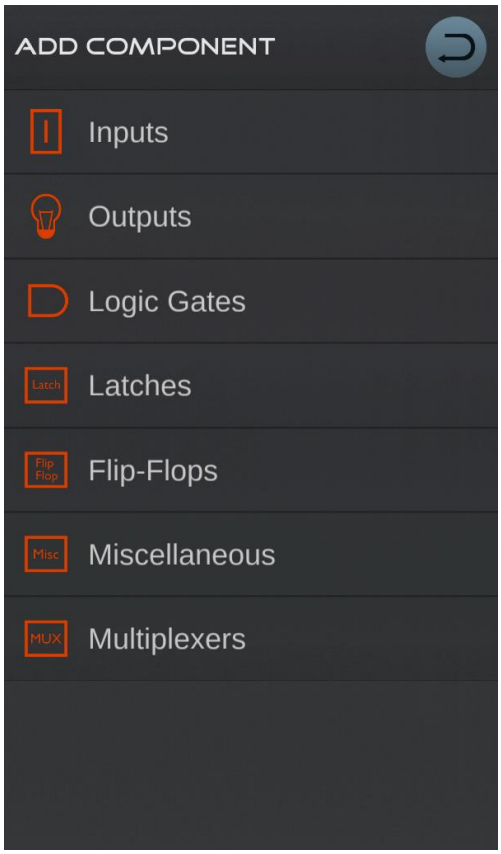

Figura 2.14 Pantalla para añadir un nuevo componente al circuito.

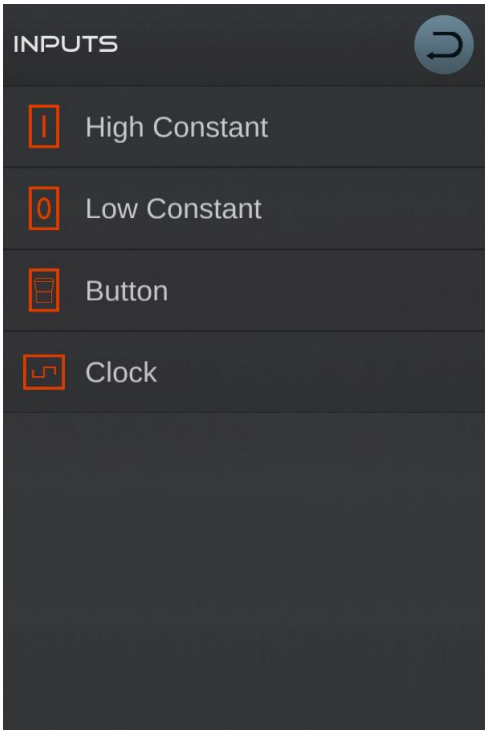

Figura 2.15 Ejemplos de entradas a añadir al circuito.

Como resultado final te muestra cómo va quedando el diseño del circuito.

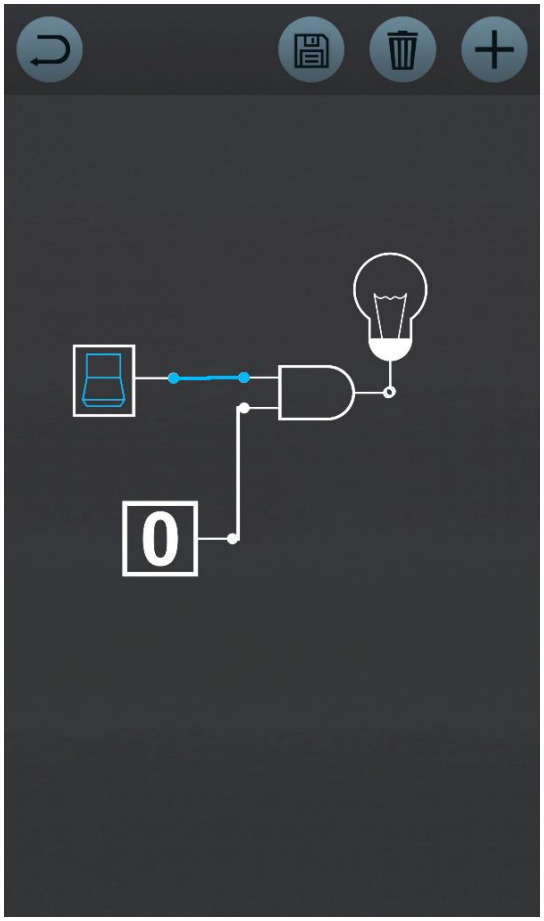

Figura 2.16Ejemplo de circuito.

Aspectos positivos y negativos de esta aplicación:

- ✓ Disponible Off-Line.
- ✓ Bien organizado cada tipo de componente.
- ✓ Permite la rotación de pantalla.
- ✓ Ofrece la posibilidad de guardar y cargar cada circuito realizado.

- × No permite la rotación de la pantalla.
- × No ofrece ejercicios para probar el circuito.

3 Diseño

3.1 Introducción

Tras un análisis del estado del arte donde se ha estudiado el estado actual del mercado de las aplicaciones móviles y sus sistemas operativos se procede a detallar el diseño de la aplicación, el por qué se ha elegido el sistema operativo de Android y todo lo que ha sido necesario para llevarlo a cabo.

3.2 Enseñanza avanzando al lado de la tecnología

Este proyecto surgió de la idea de ayudar a entender de una manera más sencilla conceptos básicos de la asignatura DIE (EPS UAM) por medio de la realización de ejercicios apoyándose para ello de la tecnología.

Debido a que la tecnología avanza cada día con más fuerza en todos los ámbitos de nuestra sociedad, ayudando a campos tan importantes como la medicina o la ciencia, se ha decidido apoyar el crecimiento de la tecnología en otro campo de interés como es la enseñanza. Desde la sociedad se pretende de alguna manera el cambio del papel, cuadernos y pizarras por dispositivos inteligentes. Poco a poco la tecnología ha ido copando las aulas, ya sea mediante el uso de diapositivas, proyectores o ayudándonos en casa con el uso del ordenador o tablets para recolectar información, acceder al temario de la asignatura y exámenes de años anteriores o incluso la realización de algunos exámenes. De esta manera se optimiza la metodología de trabajo en las aulas, pues es una manera mucho más cómoda, tanto para el alumno a la hora de tener acceso a los temas sin tener grandes libros de manera física que trasladar de un lugar a otro e incluso realizar el examen de manera interactiva sin papeles y poder saber en algunos casos la nota de manera automática, como también para el profesor que puede actualizar el temario del curso en cualquier momento, interactuar con su alumnado de una manera mucho más continua o corregir exámenes de una forma más sencilla.

Este hecho de poder compartir el temario de cualquier asignatura por medio de cualquier plataforma se ha ido extendiendo de manera satisfactoria de modo que cualquier alumno puede disponer del material necesario de la asignatura en cualquier parte si tiene acceso a un ordenador. Esto, unido al avance de la tecnología dónde, por comodidad, se ha ido sustituyendo el ordenador de sobremesa personal por el ordenador portátil personal y, hoy en día, por las tablets gracias a que ofrecen la posibilidad de trabajar a nivel de usuario como si fuera un ordenador personal, hacen que sea interesante el aportar aplicaciones y facilidades para facilitar el aprendizaje a través de esta herramienta al alcance de los alumnos, manejable y de fácil traslado.

3.3 Elección del proyecto

Como se ha explicado anteriormente, se pretende aprovechar la expansión de la tecnología de las tablets en todos los campos, incluida la educación, para facilitar el aprendizaje de conceptos básicos a través de la realización de ejercicios en una aplicación y asentar las ideas clave expuestas en las clases teóricas.

3.4 Diseño del sistema

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

3.4.1 Herramientas necesarias

Las herramientas necesarias para poder llevar a cabo la realización del proyecto son:

- Ordenador (sobremesa o portátil).
- Plataforma Eclipse.
- SDK de Android.
- Simulador ADT de Android.
- Teléfono inteligente con sistema operativo Android.
- Tablet con sistema operativo de Android.

3.4.2 Pasos previos

Antes de iniciar el desarrollo de la aplicación es necesario tener unos conocimientos básicos sobre el lenguaje que utiliza la herramienta de desarrollo de Android, Java. Para ello se ha realizado una documentación sobre dicho lenguaje y se han realizado varios tutoriales con ejemplos básicos para el conocimiento de la herramienta a utilizar, en este caso Eclipse con SDK y el simulador ADT de Android. Una vez adquiridos los conocimientos necesarios, se procede a la realización de un diseño previo de cómo se va a estructurar la aplicación y que clases se van a utilizar. En este caso se decide, por usabilidad, utilizar un mismo layout para todos los ejercicios y que dicho layout se pinte por medio de la clase Canvas. Primero para que la aplicación no tenga que estar cargando continuamente una imagen con los recursos que implica, y segundo, para facilitar el posterior dibujo sobre las vías de conexión.

3.5 Requisitos

Todos los requisitos que se muestran a continuación han sido consensuados con el tutor a la hora de decidir cómo se iba a realizar el proyecto.

3.5.1 Requisitos del proyecto

El objetivo principal del proyecto es realizar una aplicación útil y sencilla de manejar, que permita a los estudiantes realizar diferentes ejercicios sobre FPGA, desde la comodidad de su dispositivo móvil, ya sea tableta o "smartphone". Se va a desarrollar en Android, sistema operativo móvil elegido para este proyecto.

En principio la aplicación está pensada para los alumnos de la asignatura DIES (Dispositivos Integrados Especializados) impartida en el primer cuatrimestre del tercer curso en el Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación en la Universidad Autónoma de Madrid, pero es válida para cualquier usuario que quiera comprobar sus conocimientos en el nivel y la temática que se ofrece.

En la aplicación desarrollada los usuarios podrán hacer uso de los siguientes recursos:

- 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 de las conexiones realizadas.
- Visualización de la lógica utilizada en los Logic Element.
- Guardar el ejercicio en un fichero con el nombre que desee para su realización posterior.
- 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 previamente.

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 o teléfono inteligente con sistema operativo Android
- Versión mínima de Android 4.1
- No se podrá usar en dispositivos con pantallas menores a 7”, debido a dificultad de captar la resolución entre pistas de la pantalla principal.
- Uso del cliente de correo electrónico del sistema operativo de Android.

3.6 Descripción de la aplicación

La aplicación cuenta con un esquema básico y visual para la optimización del aprendizaje sin dejar de lado el uso principal de la arquitectura FPGA. Cabe destacar los principales módulos y funciones con las que el usuario puede interactuar a fin de lograr los objetivos de aprendizaje del proyecto.

3.6.1 Realización de un ejercicio

Al arrancar la aplicación desde el panel de aplicaciones de nuestro dispositivo móvil se nos muestra el menú principal dónde se elegirá la realización de un ejercicio predeterminado, cargar un ejercicio ya iniciado y guardado o salir de la aplicación. También se cuenta con la funcionalidad de volver a la pantalla principal con el botón propio de *return* del dispositivo móvil correspondiente.

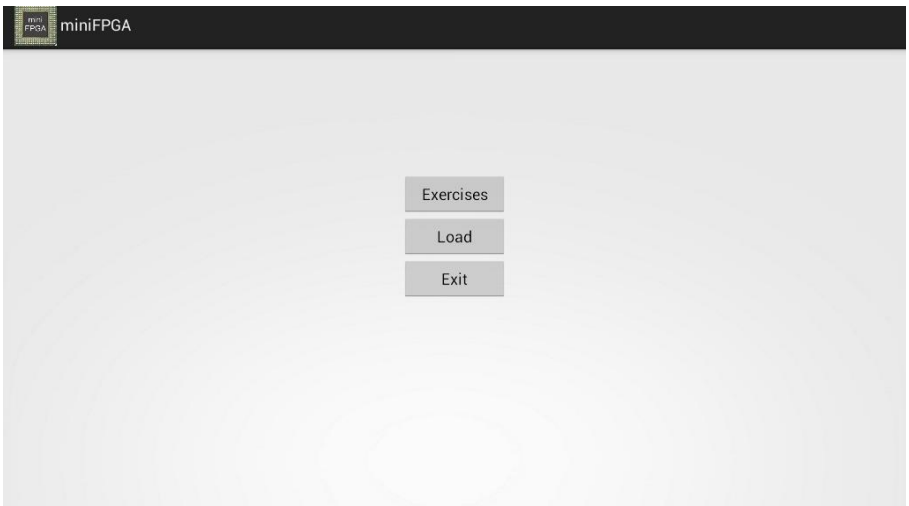

Figura 3.1 Pantalla principal.

Se elige la opción ‘Exercises’ para realizar uno de los ejercicios que se nos mostrarán en la siguiente pantalla. Esta pantalla, como todas las de la aplicación cuenta con la funcionalidad de volver a la pantalla principal con el botón propio de *return* del dispositivo móvil correspondiente.

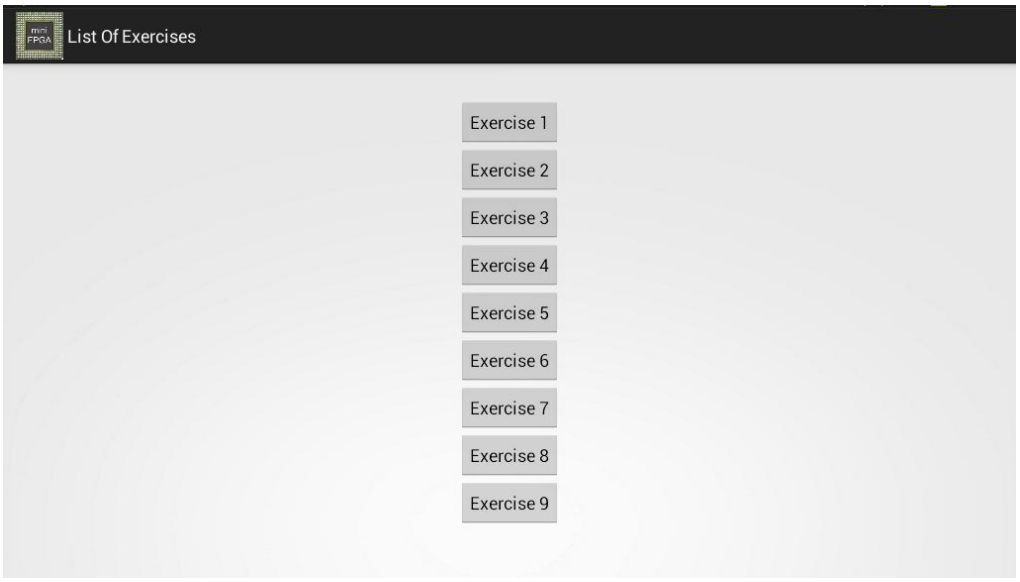

Figura 3.2 Lista de ejercicios.

Ahora se deberá elegir un ejercicio a realizar de los mostrados en la lista. Esta serie de ejercicios han sido escogidos con la aprobación del tutor de la asignatura dónde se considera que se muestran las principales características de la arquitectura FPGA.

Al pulsar y seleccionar un ejercicio de la lista, se mostrará el enunciado con el objetivo y los requisitos del mismo. Una vez se ha entendido el ejercicio se pulsará el botón *OK*.

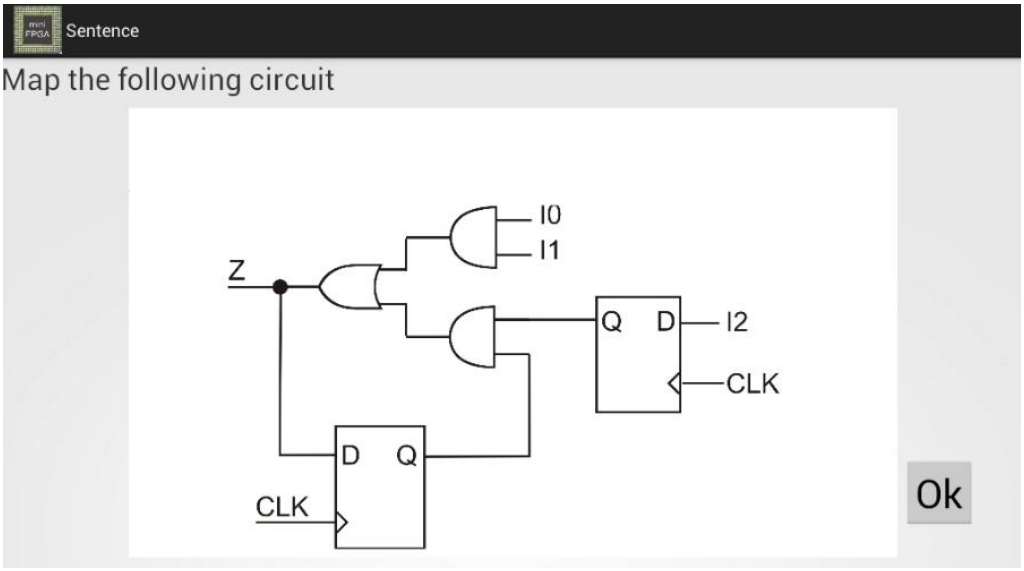

Figura 3.3 Enunciado ejercicio.

Se mostrará el layout principal y simplificado de una FPGA donde se tendrá que desarrollar el ejercicio seleccionado. Para ello se mapearan las interconexiones entre pistas, se seleccionaran las entradas y salidas así como si se ha de mapear puntos con VCC o GND o si algunos de los CLB de la FPGA requiere la sincronización por CLK.

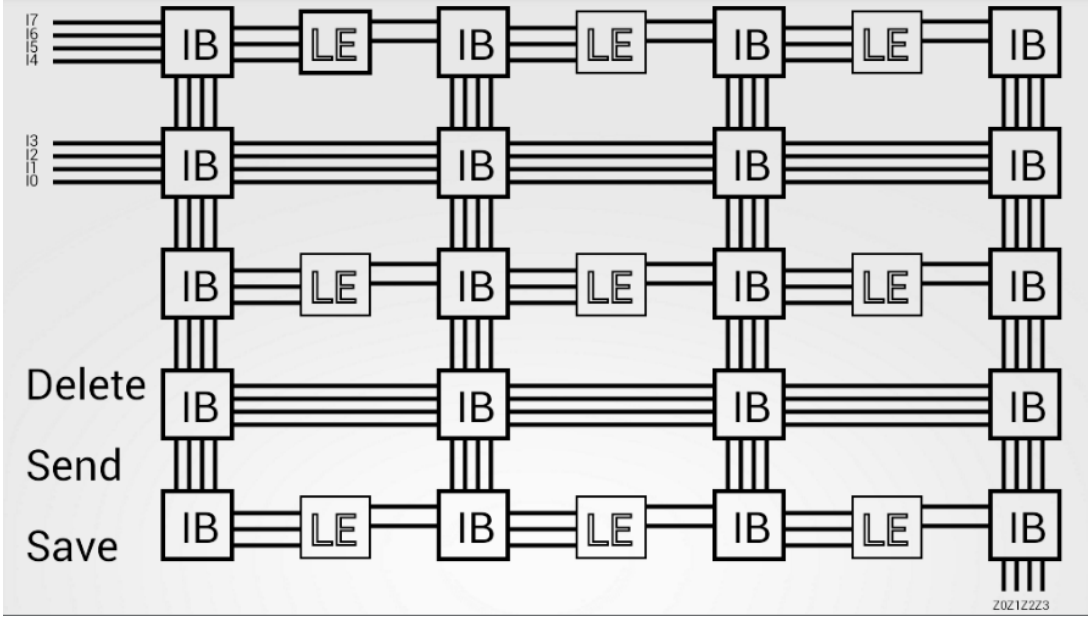

Figura 3.4 Layout principal.

En un inicio, hubo problemas en el diseño de FPGA. Se debía prestar atención al diseño básico de una FPGA teniendo en cuenta el espacio limitado de la pantalla del dispositivo. Por ello, hubo que estudiar detenidamente la situación de las entradas a la LE, el número de pistas para interconectar, como unir estas pistas,… Se decidió solucionar este problema con la creación de más cajas negras a modo de hacer zoom en las partes más importantes de la FPGA.

Para facilitar el mapeo y la interconexión de pistas se dispone de un layout donde existen dos tipos de áreas *clickables*:

- Interconection Box: para facilitar la interconexión entre pistas.
- Logic Element: para programar la lógica de la FPGA.

En esta misma pantalla se ofrece la posibilidad de pulsar sobre tres áreas *clickables* con distinta funcionalidad cada una de ellas:

Delete donde se ofrece la posibilidad al usuario de borrar los datos del ejercicio que está realizando y empezar de cero.

El área *Send* ofrece la posibilidad de adjuntar el ejercicio a un correo para poder mandarlo siempre y cuando se haya guardado anteriormente a la dirección que se desee con un mensaje de texto como cuerpo del mensaje.

Save da la posibilidad al usuario de poder guardar el ejercicio que está resolviendo en ese instante. Se ofrecerá la posibilidad de ponerle un nombre al fichero para posteriormente cuando el usuario así lo estime recuperar los datos y poder finalizar el ejercicio.

3.6.2 Importar un ejercicio

Para poder importar un ejercicio se debe pulsar sobre el botón Load en el menú principal. Una vez pulsado, aparecerá una pantalla donde insertar el nombre del fichero guardado. Al escribirlo y pulsar el botón Load se cargará automáticamente todos los datos del ejercicio para poder continuar con el desarrollo del mismo. Al realizar la importación de un ejercicio no se podrá volver atrás para comprobar el enunciado del ejercicio.

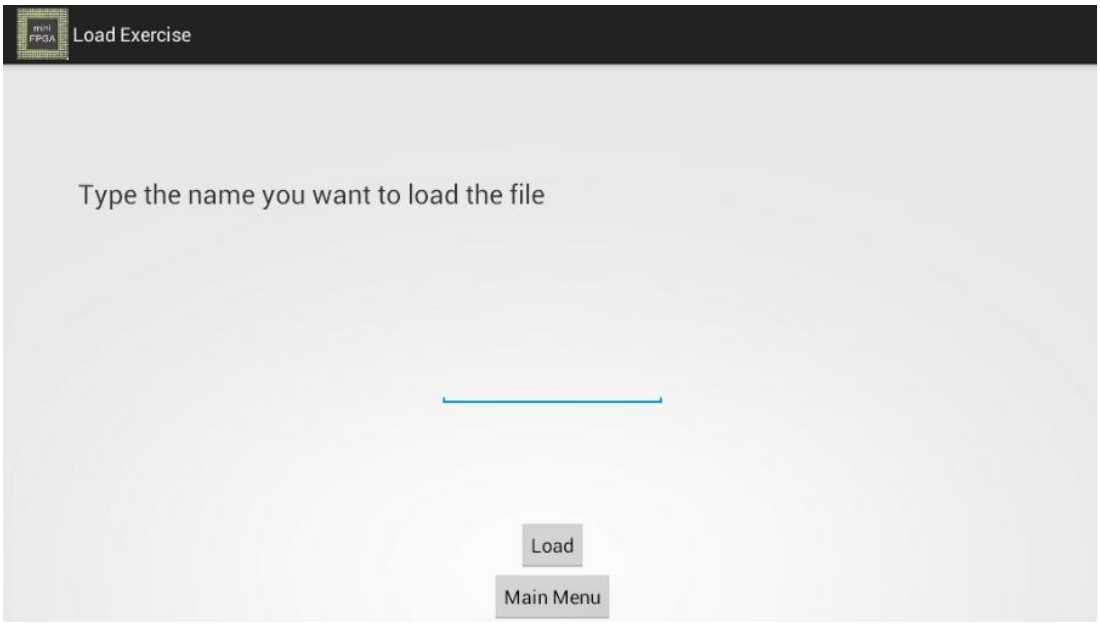

Figura 3.5 Importar ejercicio.

3.7 Desarrollo de la aplicación

Una vez se han adquirido los conocimientos mínimos de Java para introducirse en el mundo del desarrollo de las aplicaciones móviles, así como conocer también el entorno en el que se va a desarrollar el proyecto, en este caso Eclipse y haber realizado algunos ejercicios tutoriales se comienza con el desarrollo de la aplicación contado siempre con los objetivos y requisitos del proyecto.

En este apartado 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.

Tras valorarse distintas soluciones, se decidió pintar todo el layout en lugar de tener una imagen de fondo de tal manera que la aplicación ocupe lo menos posible ya que no tiene que cargar la imagen cada vez que cargue la pantalla principal (dibujando pixel a pixel incluso los pixeles blancos), sino que dibuja sobre un fondo por defecto solo los pixeles seleccionados. Además, esta opción nos permite en todo momento mejorar la calidad del detalle de pulsación sobre la pantalla, ya que sabemos en cada momento desde que pixel a que pixel va cada línea.

Otra de las razones por las que se ha decidido pintar la plantilla del dibujo en lugar de poner un fondo de pantalla ha sido la comodidad a la hora de ajustar el dibujo a las distintas pantallas, ya que, una vez dibujamos el layout principal tomando como variables la altura y el ancho de la pantalla para pintar las línea, podemos saber las limitaciones de cada área clickable.

Por tanto se ha decidido pintar usando *Canvas* la pantalla principal (el layout que emula la FPGA), pues es la misma para todos los ejercicios y de esta manera se podrá ir marcando cada unión que realice el usuario en rojo, dejando más claro el ejercicio. Para distinguir ejercicios se utilizarán identificativos que cargarán archivos de preferencias en donde se irán almacenando los distintos datos de cada ejercicio.

Las pantallas generales como el menú, enunciados o cargar y guardar el ejercicio se desarrollaron por medio de diferentes layout.

3.7.1 Clases Java

Se ha realizado la creación de distintas clases java para poder desarrollar las distintas pantallas de la aplicación. Algunas de ellas llevan asociado un layout que se carga para implementar de una manera más sencilla la pantalla correspondiente. De esta manera, el UML de la aplicación quedaría de la siguiente manera:

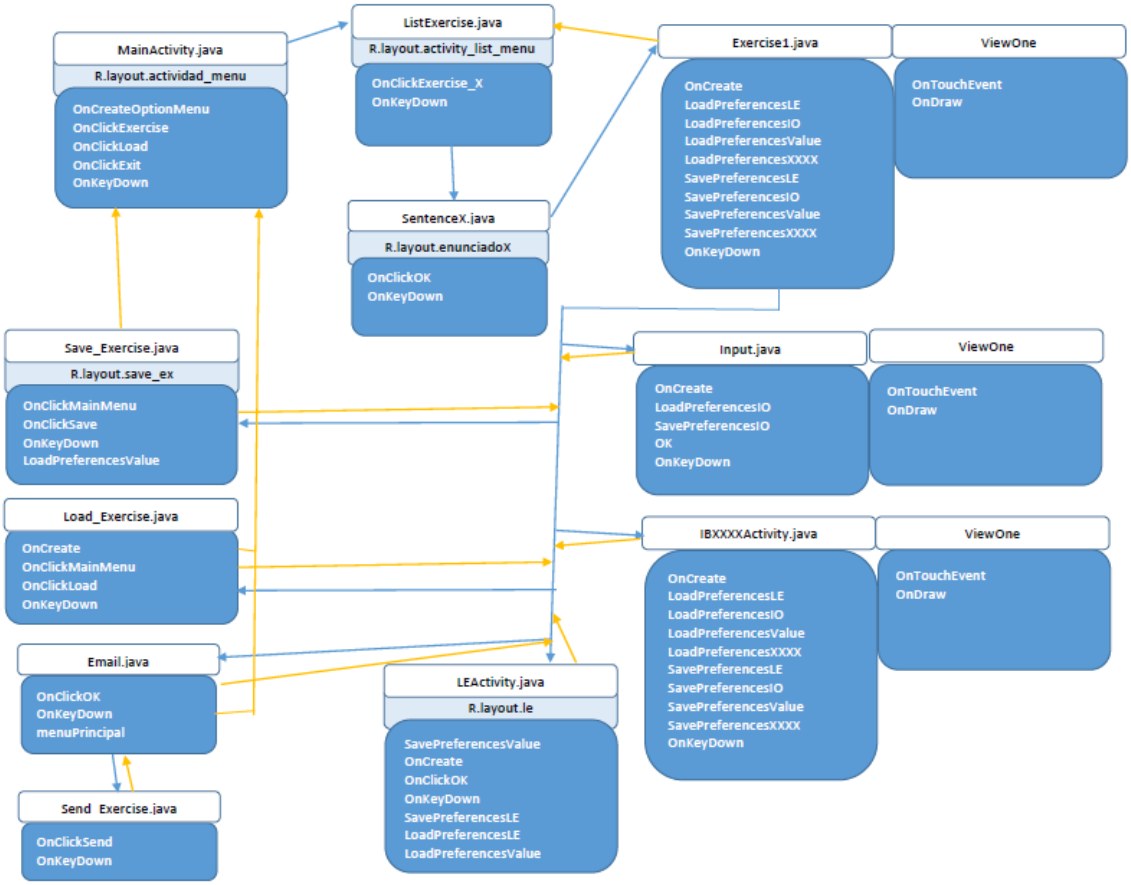

Figura 3.6 UML de la aplicación.

Menú inicial

En esta clase se asocia un layout donde se crean los tres botones principales (lista de ejercicios, cargar ejercicio y salir de la aplicación). Cada uno de los botones llama a la actividad y lanza el método correspondiente para continuar con la aplicación.

Listado con los ejercicios

Esta clase está asociada a un layout donde se crean los botones correspondientes a cada ejercicio. Nuevamente cada botón lanzará un método y llamara a la actividad correspondiente enviándole un identificativo a la siguiente actividad, puesto que todos los ejercicios comparten pantalla principal salvo que los datos se cargan en diferentes archivos de preferencias.

Importación del ejercicio

También tiene asociado un layout (load_ex) en el cual se define una frase para que el usuario escriba el nombre del archivo que quiere cargar en un texto editable previamente definido. Una vez se haya escrito y se haya pulsado el botón correspondiente, se lanza el método para buscar en la carpeta correspondiente el archivo guardado con anterioridad y se recoge el fichero de preferencias que es el que contiene todos los valores de cada conexión del ejercicio guardado. Finalmente se declara un botón para volver al menú principal.

Los diferentes enunciados de todos los ejercicios

Cada enunciado tiene asociado un layout donde se escribe el enunciado del ejercicio y se declara un botón 'OK' para acceder a la realización del mismo. A esta actividad se le envía el número de ejercicio para poder guardar las preferencias correspondientes que posteriormente será enviado a la actividad principal.

Simulación FPGA

Esta es la clase principal Exercise1.java, no tiene asociado ningún layout puesto que se decidió pintar sobre el fondo blanco todas las líneas de la FPGA, haciendo *clickables* las zonas que nos interesan (Interconection Box, Logic Element, Save, Send, Delete). Cada vez que accedamos a esta actividad se cargará el archivo de preferencias del ejercicio correspondiente y cada vez que se efectúe alguna conexión o cambio en los Logic Element también quedará registrado en dicho archivo.

Las áreas clickables de los Logic Element e Interconection Box están numeradas de tal manera que al pulsar sobre una de ellas se le pasará el número correspondiente a la siguiente actividad para que al hacer cambios en los elementos edite el archivo de preferencias del ejercicio en el espacio correcto.

También tiene variables compartidas con las Interconection Box de tal manera que cuando accedemos a una de ellas para realizar alguna conexión de pistas y volvemos a la actividad principal, se pintaran en rojo las conexiones realizadas.

Input/Output

Se ha habilitado una clase (Input.java) donde el usuario puede habilitar/deshabilitar tanto entradas como salidas. De esta manera se facilita al usuario dicha función debido a que por motivos de espacio en pantalla resulta difícil hacerlo en la pantalla principal.

Figura 3.7 Habilitar/deshabilitar entradas/salidas.

Interconection Box

Se dispone de distintas clases de Interconection Box, según el número de pistas con posibilidad de ser conectadas entre sí pero todas ellas siguen un mismo sistema. Reciben de la actividad principal dos argumentos: uno de ellos con el número de ejercicio correspondiente para saber que archivo de preferencias debe modificar y otro con el número de Interconection Box que sele es asignado para saber que parte del archivo de preferencias debe modificar.

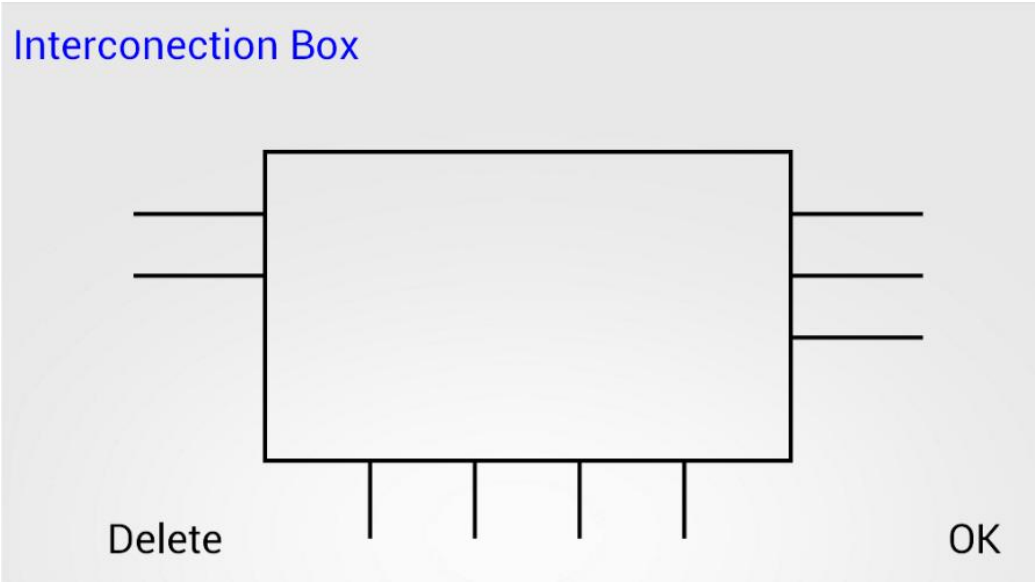

Figura 3.8 Interconection Box.

Así mismo, tras realizar las conexiones que se necesiten, le devolverá al programa principal las variables necesarias para realziar dichas conexiones en la FPGA.

Logic Element

Tiene asociado el layout le.xml donde se reproduce un Logic Element (multiplexor + Flip Flop). Se añaden botones de cambio de estado (1/0) para poder configurar cada Logic Element. Se da la opción de sacar la salida síncrona o asíncronamente.

Al igual que pasa con las Interconection Box, cada Logic Element recibe de la actividad principal dos variables: el número de ejercicio para editar el archivo de preferencias correspondiente y el número de Logic Element asignado según se haya pulsado uno u otro, para saber qué espacio del fichero debe editar.

Save

En esta clase se carga el layout sabe_ex y se escribirá el nombre con el que se quiere guardar el archivo de preferencias que contiene los datos del ejercicio. Se declaran dos botones: el primero de ellos para confirmar y poder guardar el ejercicio, y el segundo para volver al menú principal.

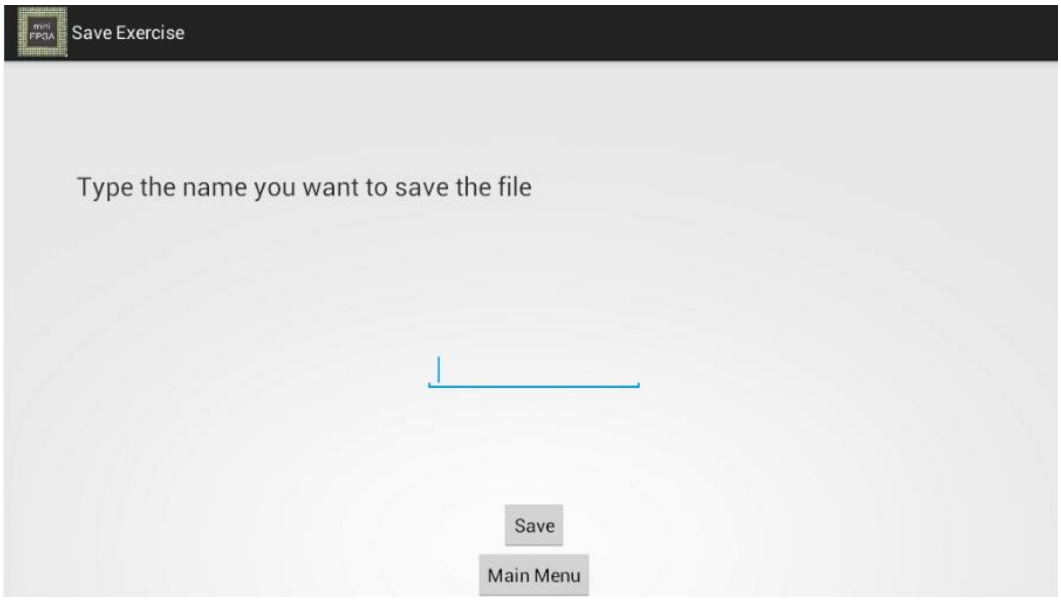

Figura 3.9 Guardar ejercicio.

Send_Exercise

En esta clase se carga el layout activity_send_exercise donde se muestra una frase al usuario para que escriba, en el texto editable declarado, el nombre con el que desea guardar el ejercicio. Además se declaran dos botones: el primero para confirmar el deseo de enviar el ejercicio y proceder a la pantalla de escribir el mail, y el botón para volver al menú principal.

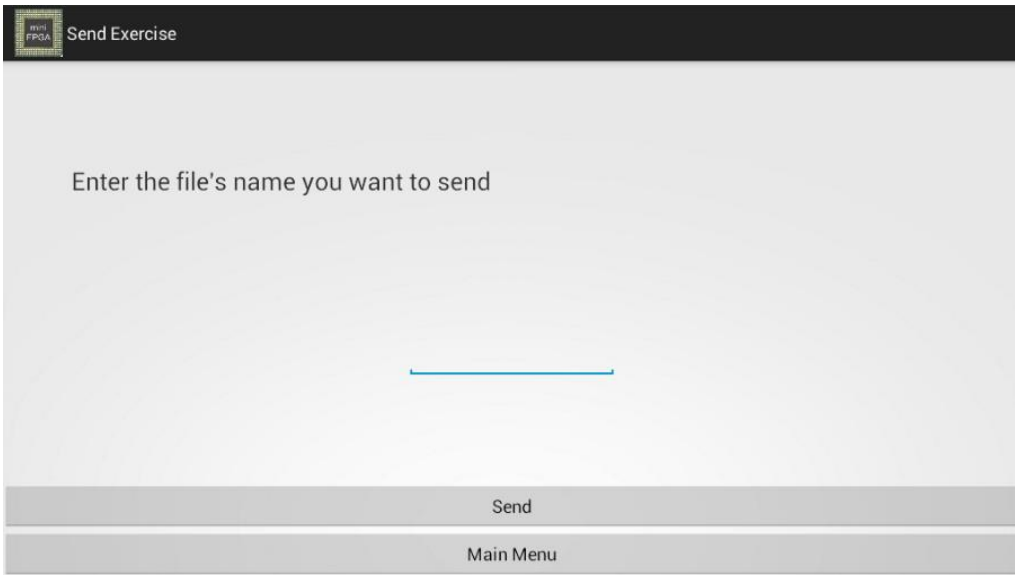

Figura 3.10 Enviar ejercicio.

Mail

En el layout de esta clase (activity_send_exercise) se escriben los datos correspondientes al correo que se va a enviar, tanto el destinatario como el asunto y si se quiere un mensaje de texto. El archivo quedará adjuntado en el correo. Se declaran también dos botones para enviar definitivamente el correo y para volver al menú principal.

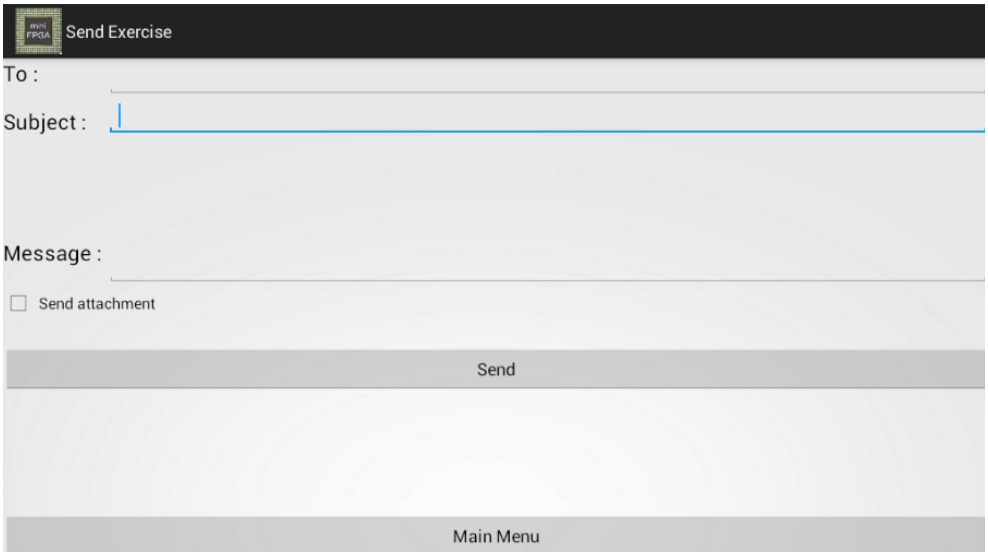

Figura 3.11 Enviar correo.

3.8 Ejemplos

En este capítulo de la memoria se resolverá un ejercicio de la aplicación a modo de ejemplo. El mecanismo es el mismo para todos los ejercicios ya que se ha usado el mismo layout para todos ellos de forma que se simplificará la aplicación y sea el usuario el que interprete el layout y sea capaz de realizar las interconexiones necesarias, elegir las entradas y salidas que se utilizan y el que programe la lógica de los Logic Element.

3.8.1 Ejercicio 1

El enunciado de dicho ejercicio es el siguiente:

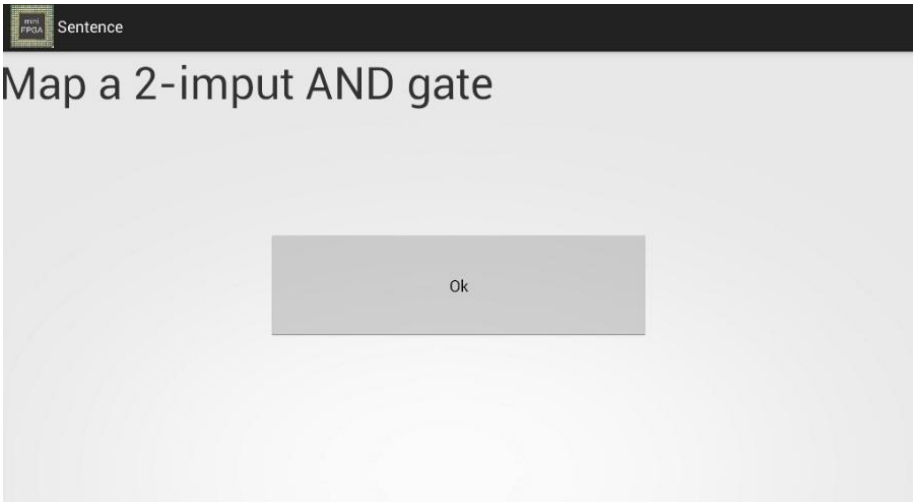

Figura 3.12 Enunciado ejercicio 1.

El propósito de este ejercicio es mapear una puerta lógica tipo AND de dos entradas. Para ello se habilitan dos de las ocho entradas de las que se dispone en la aplicación y una salida. También será necesario conectar la primera entrada del Logic Element a GND ya que no se usan las cuatro primeras entradas del mutiplexor. Podremos elegir el Logic Element que queramos y las Interconection Box que creamos oportunas, de tal manera que queden conectadas las entradas habilitadas al Logic Element y la salida de este a la salida de la FPGA. Tendremos también que crear la lógica del Logic Element para simular la puerta AND, teniendo en cuenta que las cuatro primeras entradas no se usan.

Lo primero que haremos será seleccionar las entradas que vamos a utilizar, para ello pulsamos sobre las entradas y seleccionamos dos de ellas.

Figura 3.13 Habilitar entradas/salidas.

Conectamos las líneas de entrada a la entrada del Logic Element, ponemos la primera entrada del LE a GND y mapeamos acorde a lo pedido en el ejercicio (en este caso se mapeará el primer LE, pero en este caso, se puede realizar en cualquiera de los otros ocho).

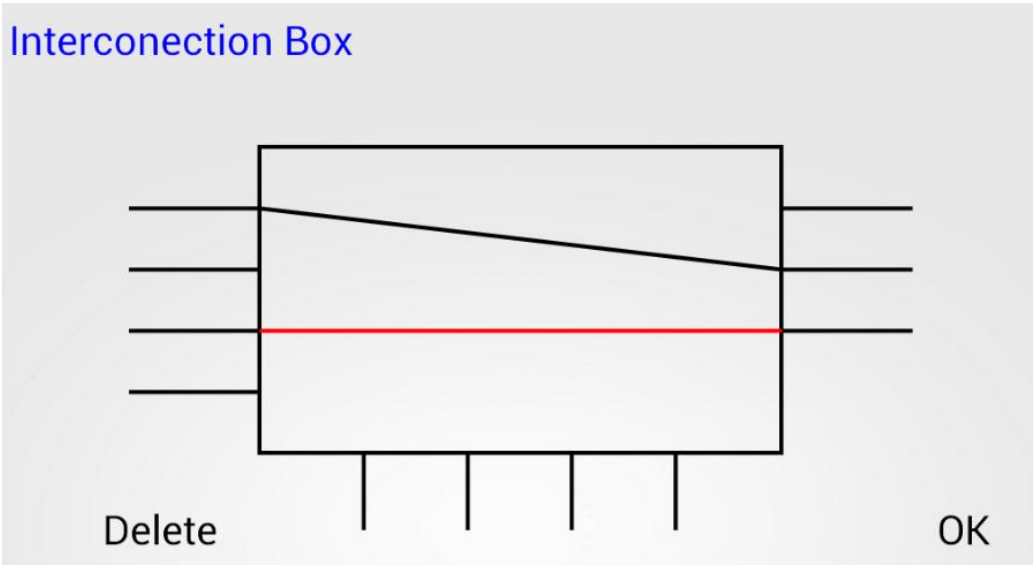

Figura 3.14 Realizar conexiones en las IB.

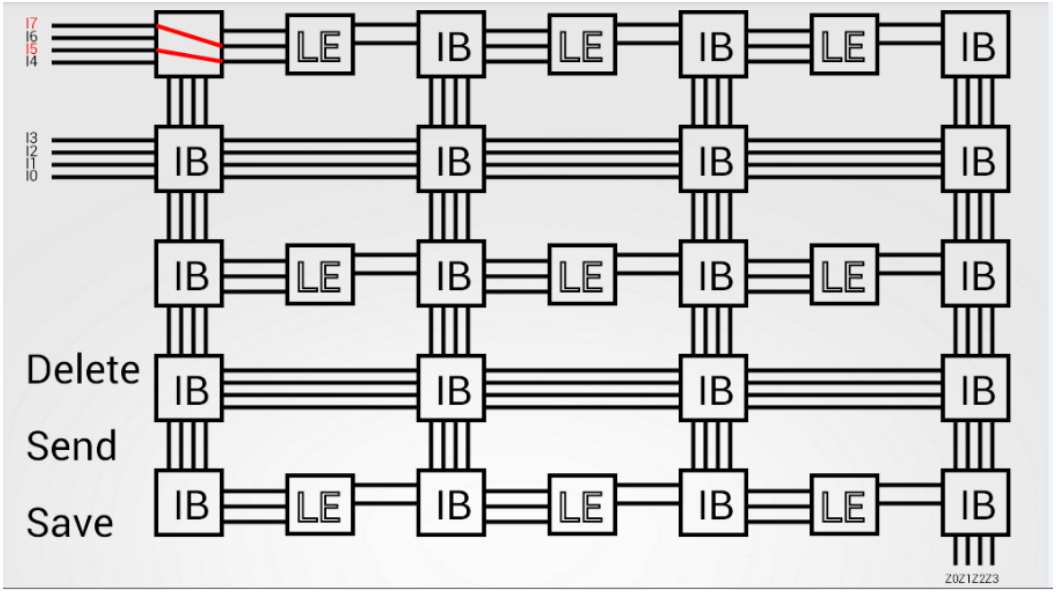

Figura 3.15 Vista de la FPGA tras conexiones.

Finamente, conectamos las vías necesarias para llegar a la salida y activamos la salida necesaria para completar el ejercicio.

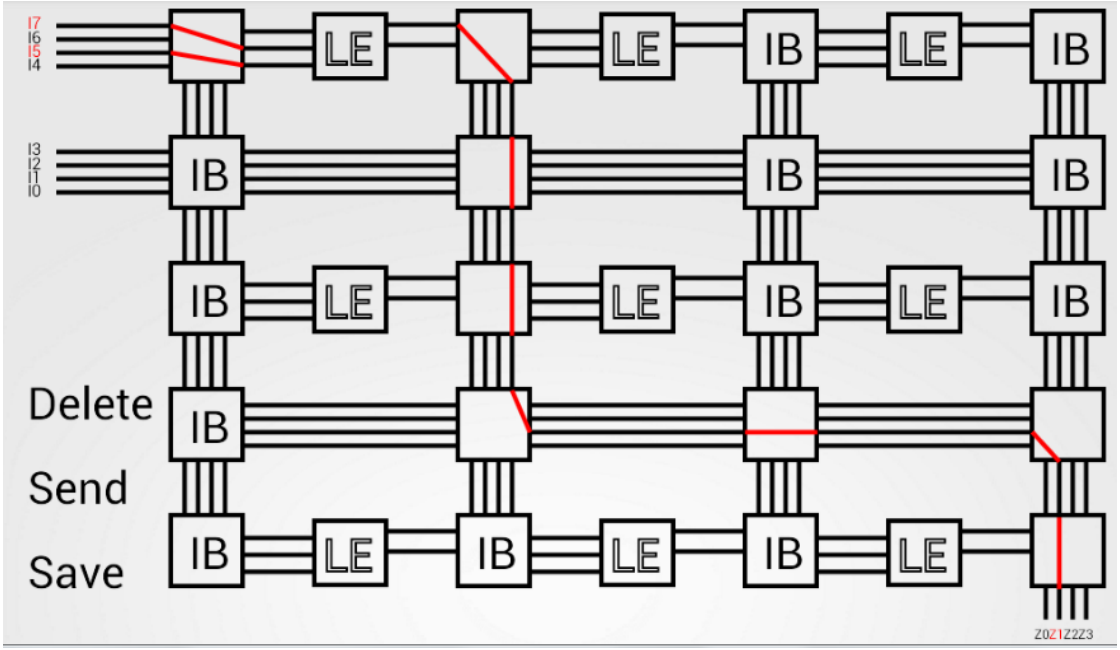

Figura 3.16 Vista de ejercicio FPGA.

3.8.2 Ejercicio 7

El enunciado de este ejercicio es el siguiente:

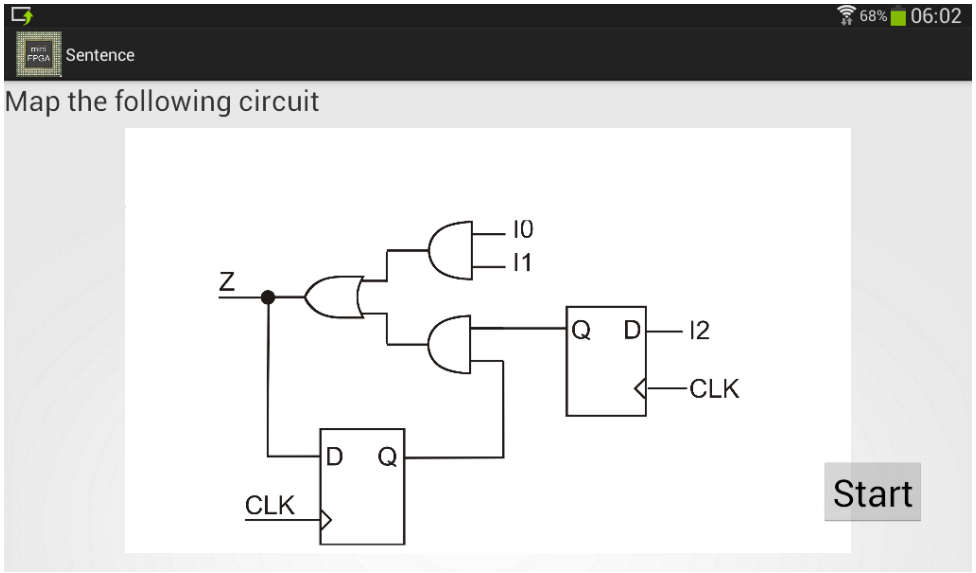

Figura 3.17 Enunciado ejercicio 7.

El propósito de este ejercicio es el de mapear el circuito lógico del enunciado usando el layout genérico para todos los ejercicios.

Para ello se ha seguido el dibujo del enunciado, realizando las conexiones como se observa en el mismo. Puesto que este ejercicio se podría realizar de múltiples manera aquí solo se ofrecerá una de ellas.

En este ejercicio tendremos que habilitar tres salidas, las cuales estarán conectadas con la salida de una caja lógica debidamente configurada. La salida de estas cajas lógicas además de ser salida, servirán de entrada a la siguiente caja lógica cerrando el circuito.

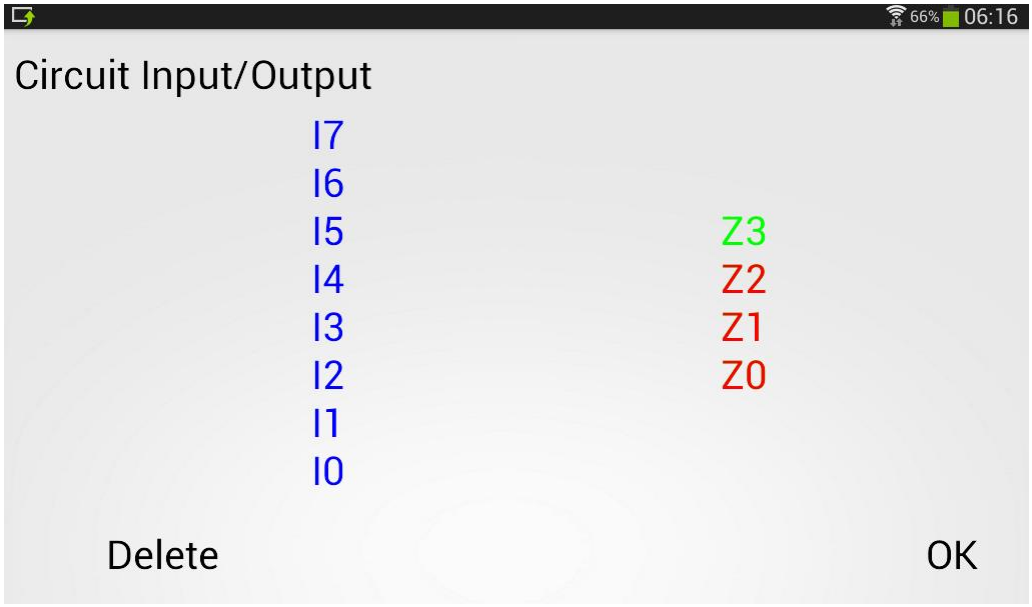

Figura 3.18 Elección entradas/salidas del circuito.

Habrá que configurar los Logic Element para que nos permitan realizar el circuito del enunciado. Se conectarán las dos entradas más significativas a tierra para sacar el valor correcto.

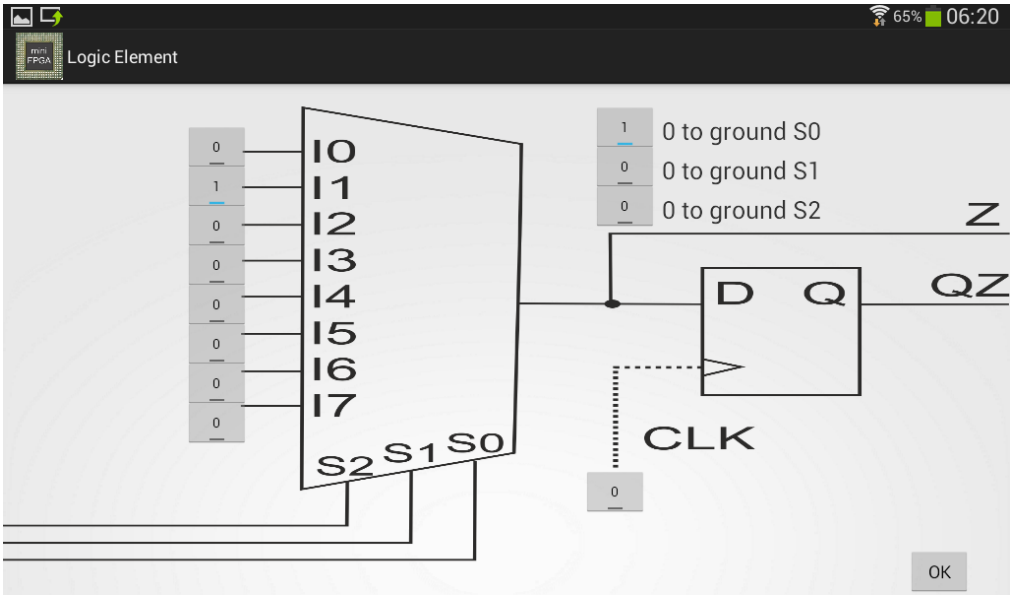

Figura 3.19 Configuración LE del circuito.

Además se activará la salida síncrona activando el botón de CLK.

Finalmente habrá que realizar las conexiones oportunas, quedando por lo tanto el circuito de la siguiente manera:

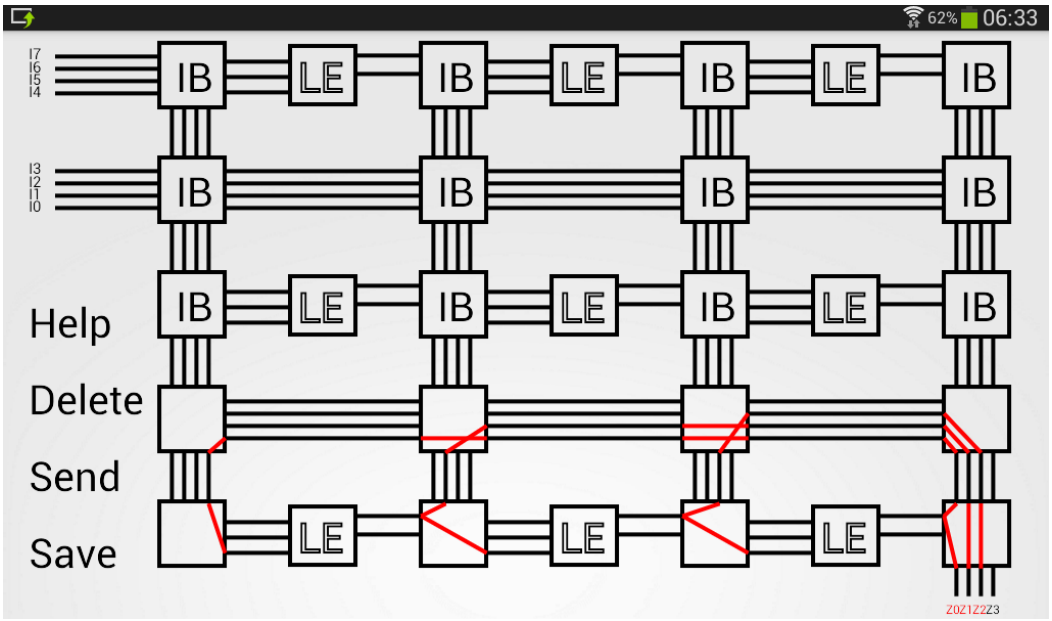

Figura 3.20 Resultado final del circuito.

4 Conclusiones y pruebas de diseño

4.1 Conclusiones

El principal objetivo en el inicio del proyecto era el desarrollo de una aplicación educativa prototipo para la plataforma Android, valido tanto para tablets como móviles cuyo tamaño de pantalla fuera lo suficientemente grande, que permitiera a los usuarios (principalmente estudiantes) estudiar y realizar una serie de ejercicios tipo con los que comprender el funcionamiento de una FPGA.

Este objetivo se ha cumplido obteniendo una aplicación Android enfocada a la asignatura DIES (Dispositivos Integrados Especializados) impartida en el primer cuatrimestre del tercer curso en el Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación en la Universidad Autónoma de Madrid. Los alumnos se descargarán la aplicación al inicio del estudio de la tecnología FPGA en clase, de tal manera que puedan ir realizando ejercicios mandados por el profesor y vayan comprendiendo y completando los conocimientos adquiridos en la clase teórica.

Los principales puntos que repasa esta aplicación son:

- Teorema de Shannon para mapear en LUTs
- Mapeo de un pipeline
- Partición óptima en LUT
- LUT transparente
- LUT como inversor

Además se han conseguido otra serie de funcionalidades:

- Insertar una guía en modo ayuda en modo sólo lectura para la realización de ejercicios.
- Conseguir adaptar ejercicios de electrónica a dispositivos inteligentes Android.
- Posibilidad de guardar y cargar ejercicios en cualquier momento del ejercicio.
- Posibilidad de envío del ejercicio por correo electrónico.
- Adaptación de las imágenes a los distintos tipos de pantalla.

La aplicación permite dar un paso más en la educación dejando a un lado los papeles y los cuadernos, dando la opción de poder practicar y repasar en cualquier parte sin necesidad de depender de material, simplemente desde el dispositivo móvil.

La opción de autocorrección, pese a ser parte del objetivo se decidió descartarla debido al gran número de posibilidades distintas que ofrece un esquema como el diseñado de realizar un ejercicio de la manera correcta. Por ello, se limita este hecho al envío del ejercicio al profesor para su evaluación.

A nivel didáctico, se han aprendido diferentes herramientas y conceptos de interés:

- Programación JAVA y XML.
- Conocer el entorno de desarrollo Eclipse y la arquitectura Android.
- Organización y planificación de un proyecto

4.2 Pruebas de diseño

Según se ha ido diseñando y desarrollando la aplicación, han ido surgiendo distintos problemas, dudas y/o distintas opciones sobre la realización del proyecto.

Problema	Solución
Inicialmente surgió el problema de cuántos Logic Element se podrían implementar de tal manera que diera el suficiente margen para la realización de algunos ejercicios complejos y pudieran caber en la pantalla sin ningún problema.	Se decidió realizar un diseño en forma de cuadricula o matriz para poder optimizar el número de LE a la pantalla. Para el problema de la visualización y la facilidad a la hora de pulsar e interactuar con la aplicación se barajaron dos opciones: • Realización de zoom al tocar y ampliar con los dedos la pantalla. • La realización de zoom utilizando una nueva pantalla mostrando únicamente el elemento clicado. Se decidió realizar esta opción debido para clarificar el Logic Element y su configuración.
Cómo mostrar la interconexión de vías debido al gran número que se entrecruzaban para unir los LE.	Este problema se solucionó realizando 'cajas negras' de interconexión. Pulsando sobre estas Interconection Box se activa otra pantalla en forma de zoom donde ver de forma clara las conexiones posibles a realizar y poder hacerlo y llevarlo al ejercicio principal.
Cómo conseguir la calidad suficiente en el click para diferenciar pulsaciones con tan mínima distancia en diferentes pantallas.	Para solucionar este problema se optó por pintar directamente todo el circuito ajustándolo a cada pantalla. De esta manera ya se sabe en qué pixel empieza cada acción y resulta más fácil ajustar los clickables.

Conseguir enviar los valores del ejercicio de una pantalla a otra diferenciando el ejercicio al que pertenecen utilizando una sola clase común para mostrar la plantilla de ejercicio la cuál es común para todos.	Mediante la utilización de archivos de preferencias se consigue separar los valores de cada uno de los ejercicios, teniendo una organización clara de a qué valor del ejercicio corresponde cada posición dentro de dicho archivo y creando un archivo para cada ejercicio.
El diseño de las pantallas ofrecía siempre la misma duda: realizar el dibujo mediante el método 'OnDraw' o poner un fondo con el dibujo directamente.	Se optó por la opción de dibujar las pantallas en las cuales el usuario interactúa con el ejercicio para que, de esta manera, se pudieran agregar funciones como las de marcar en rojo la última conexión realizada.

	Descripción prueba	Resultado esperado	OK/NOK
1	Pintar la plantilla principal sobre la que se realizará el ejercicio con todas las 'cajas negras' y vías de conexión necesarias.	Se espera que el dibujo se ajuste correctamente a todos los tamaños de pantalla, sin salirse ninguna línea por los márgenes y con el espacio suficiente para que la visión de la plantilla sea clara y para poder hacer cada área clickable independiente	OK
2	Comprobar que todas las áreas clickables de la plantilla del ejercicio funcionan. (Entradas, Interconection Box, Logic Element, Help, Save, Send, Delete)	Cada área clickable tiene que hacer su función, esto es, ir a la pantalla correspondiente o borrar todos los datos del ejercicio en el caso del botón Delete.	OK
3	Comprobar dentro de Interconection Box que todas las conexiones funcionan y las lleva correctamente a la pantalla principal.	Cada una de las posibles conexiones de cada una de las Interconection Box debe dibujarse de manera correcta, así como quedarse en rojo la última conexión realizada, cambiando de nuevo a negro cuando haya algún nuevo cambio. Todas las conexiones realizadas tienen que enviarse y dibujarse posteriormente en la pantalla principal.	OK

4	Comprobar que la aplicación coge bien los valores de los botones toggle de los Logic Element y realiza correctamente la lógica aplicada.	Al entrar en los Logic Element y programar la lógica necesaria, el valor a la salida debe ser el esperado realizando correctamente la función del multiplexor.	OK
5	Las imágenes importadas para utilizar como fondo de pantalla deben ajustarse correctamente a cada tamaño.	Se comprueba con distintos dispositivos que las imágenes se ajustan correctamente.	OK
6	Las funciones de guardar, cargar y enviar ejercicio funcionan correctamente.	Se comprueba que el ejercicio se puede guardar, que se carga correctamente y que se puede enviar por correo haciendo uso de la aplicación propia de cada dispositivo para mandar correos.	OK

4.3 Desarrollos futuros

La utilización de dispositivos móviles en el mundo de la educación tiene enormes ventajas. Esta aplicación ha sido desarrollada mejorando aspectos de una primera versión. Aun así, la idea principal es que se continúe mejorando en futuros proyectos de fin de carrera. En el desarrollo del proyecto se han ido visualizando mejoras, algunas de ellas han hecho que el proyecto se vea modificado para mejorar su funcionamiento y otras de ellas no ha sido posible llevarlas a la practica en esta versión, bien por ser complejas o la necesidad de reconstruir el proyecto desde cero con el tiempo que ello implique.

- Actualmente, todos los layouts son cargados desde la aplicación. Se puede optimizar el espacio de la misma si estas imágenes estuvieran en una base de datos de la cual nos las trajéramos siempre que nos fueran de utilidad, dejando espacio libre cuando no fueran necesarias. Se podrías utilizar de un servidor donde almacenar los datos que más ocupan, principalmente imágenes. Incluso crear una base de datos en un servidor, en los que aparte de las imágenes, se puedan añadir enunciados, soluciones de ejercicios, consejos y ayudas, y estos puedan ser cargados en la aplicación o en el conjunto de aplicaciones de DSLab.

- Crear un foro en dicho servidor donde los estudiantes puedan compartir sus dudas y debatir sobre la tecnología FPGA.

- Al cargar un ejercicio, no se permite el interactuar entre actividades como por ejemplo volver a ver el enunciado, siendo solo posible la finalización del mismo. Esto sería mejorable con una arquitectura más compleja dónde a partir de algún identificativo se nos permitiera movernos por la aplicación con total libertad sin perder los datos del ejercicio.

- Realizar algún algoritmo fiable para la corrección de ejercicios de tal manera que se pudiera llegar a realizar exámenes directamente sobre el dispositivo. Para ello sería necesario agregar una palabra clave que sólo permita acceder a la misma durante el horario de examen, limitar el envío por correo y agregar un temporizador tipo "watchdog" que cierra la aplicación a las 3 horas.

- Según está diseñado el proyecto, el usuario puede conectar y programar todas las pistas de la FPGA, pero en ningún momento puede hacer una demostración de la FPGA. En vistas a una posible nueva versión sería interesante que se recibiese una información visual de cada pista según este o no activa.

- Añadir opciones de idioma a la aplicación para ayudar a estudiantes erasmus.

- Actualmente no se ha considerado apropiado la implementación para la visualización del mapa de memoria de la FPGA por falta de espacio en pantalla para mostrarla. Habría alguna forma de poder realizar dicho mapeo, por ejemplo que guardando los valores en una base de datos o sitio web fácilmente accesible cuando se desee mostrarlos y con una visualización mejor que la de la propia aplicación.

- Incorporación de un cronómetro y sistema de puntuaciones en los ejercicios.

- Se propone reservar un espacio en la aplicación que pueda servir de guía a la hora de resolver un ejercicio. De tal manera que si el usuario se ve perdido en la actividad, reciba algún tipo de consejo o incluso enlaces de interés o material de la asignatura dónde revisar sus pasos.

- Como siempre, esta aplicación se puede optimizar en cuanto al uso de recursos y espacio a la hora de lanzarla al mercado.

5 Referencias

[1] **cincodias.com**
[2] **https://es.wikipedia.org/wiki/Tel%C3%A9fono_inteligente**
[3] **https://es.wikipedia.org/wiki/Tableta_(computadora)**
[4] **https://www.wayerless.com/2012/05/para-el-2016-las-tabletas-habran-suplantado-a-las-notebooks/**
[5] **http://es.slideshare.net/Marcoviaweb/app-inventor-flisol-2015**
[6] **https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/**
[7] **http://www.configurarequipos.com/doc1107.html**
[8] **http://quees.la/mercado-smartphone-2013/**
[9] **http://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte_ES_TMT_Consumo-movil-espana-2014-def.pdf**
[10] **www.enriquedans.com**
hipertextual.com
[11] **http://www.edukanda.es/mediatecaweb/data/zip/1179/page_04.htm**
[12] **http://cincodias.com/cincodias/2015/01/26/lifestyle/1422287483_090165.html**
[13] **http://www.areatecnologia.com/informatica/sistemas-operativos-moviles.html**

Apéndice I

Manual de Usuario

Se procede a detallar en este apartado el funcionamiento de la aplicación. Para poder entender y hacer un uso óptimo de la misma.

Instalación de la aplicación

Para poder hacer uso de la aplicación el usuario deberá bajarse la aplicación de la tienda de aplicaciones de Google, Play Store (de manera gratuita) e instalársela en su dispositivo o bien tener el archivo con extensión ".apk" que permita instalar correctamente la aplicación. Si se decide utilizar este segundo método, será el profesor de la asignatura el comparta con los alumnos el archivo junto con el resto de material del curso. El alumno simplemente deberá ejecutar dicho archivo para que la aplicación se instale correctamente en su dispositivo.

Acceso a la aplicación

Una vez la aplicación ha sido instalada, pulsar sobre el icono para entrar en ella y poder empezar a interactuar y a resolver ejercicios.

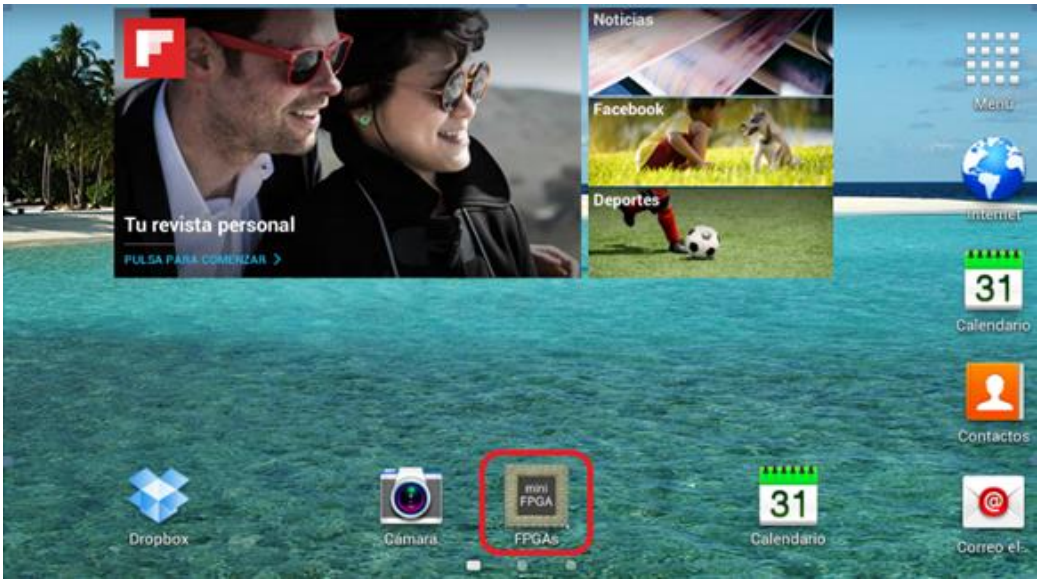

Figura 5.1 Acceso aplicación.

Menú principal

Al entrar en la aplicación aparecerá el menú principal donde el usuario tiene la opción de elegir pulsando uno de los tres botones que aparecen, “Exercise List”, “Load” o “Exit”. Para realizar un ejercicio desde el inicio, pulsar el botón ‘Exercise’ accediendo así al listado de ejercicios. Si lo que se quiere es importar un ejercicio guardado previamente para su finalización se deberá pulsar el segundo botón, “Load”, donde accederá a una pantalla para escribir el nombre del ejercicio que queremos cargar. El tercer botón sirve al usuario para salir de la aplicación, de igual modo también existe la posibilidad de salir de la aplicación pulsando el botón de atrás propio de cada dispositivo.

Figura 5.2 Menú principal.

Listado de ejercicios

Si se decide hacer un ejercicio desde su inicio deberá pulsar el botón ‘Exercise’. Una vez lo haya pulsado se mostrará un listado con los ejercicios a realizar. Para acceder al enunciado de cada uno de ellos se deberá pulsar sobre el botón del ejercicio al que queremos acceder. Así mismo, si queremos volver al menú principal, se deberá pulsar el botón de su dispositivo que realiza dicha función.

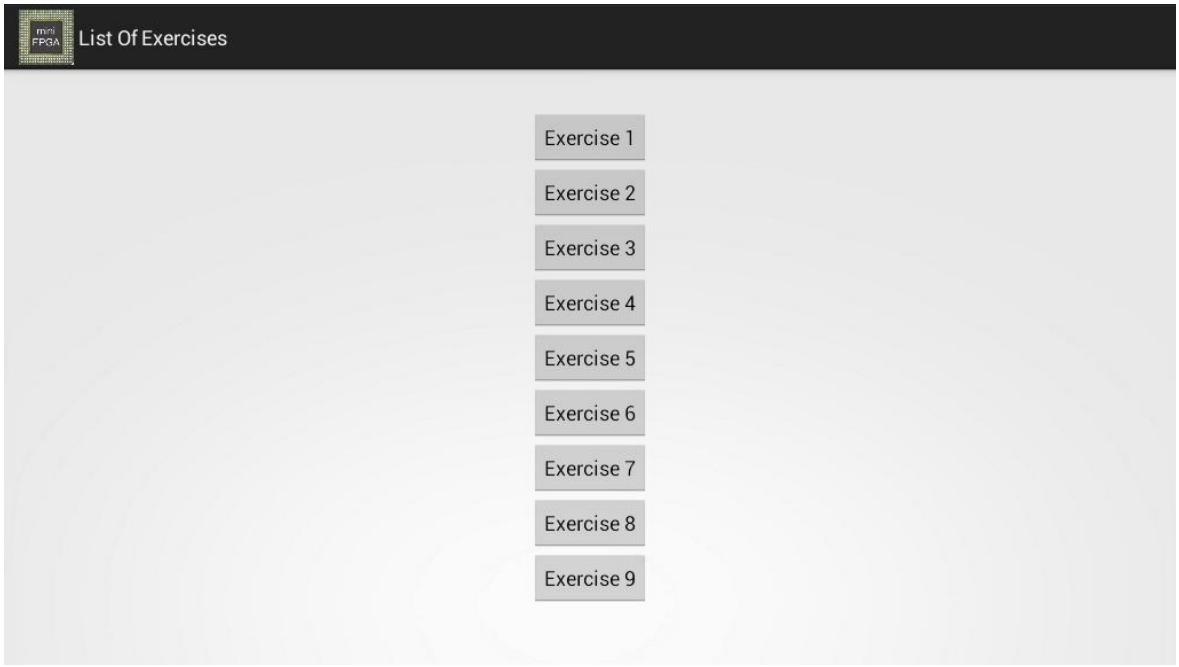

Figura 5.3 Listado de ejercicios

Enunciado

Al pulsar sobre el botón de cualquier ejercicio de la lista, se accede al enunciado del mismo. El usuario podrá leerlo y pulsar el botón 'Start' para acceder a la realización del ejercicio o pulsar el botón 'atrás' de su dispositivo para volver al listado de ejercicios.

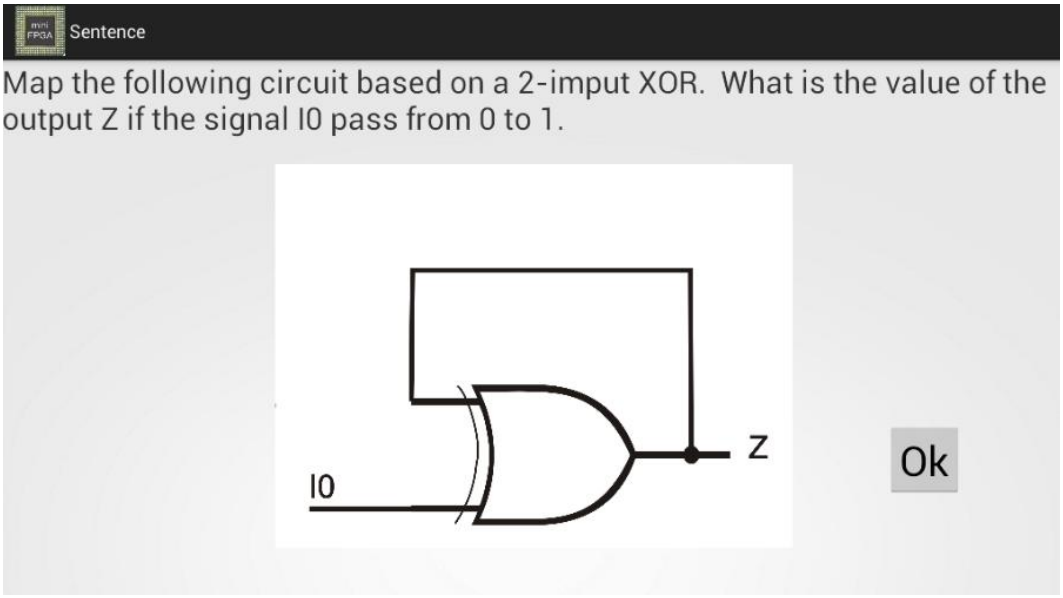

Figura 5.4 Enunciado.

Ejercicio

Una vez se ha leído el enunciado y se ha decidido realizar el ejercicio, se mostrará el layout principal de la FPGA donde deberá resolver el ejercicio mediante sus conocimientos de la asignatura.

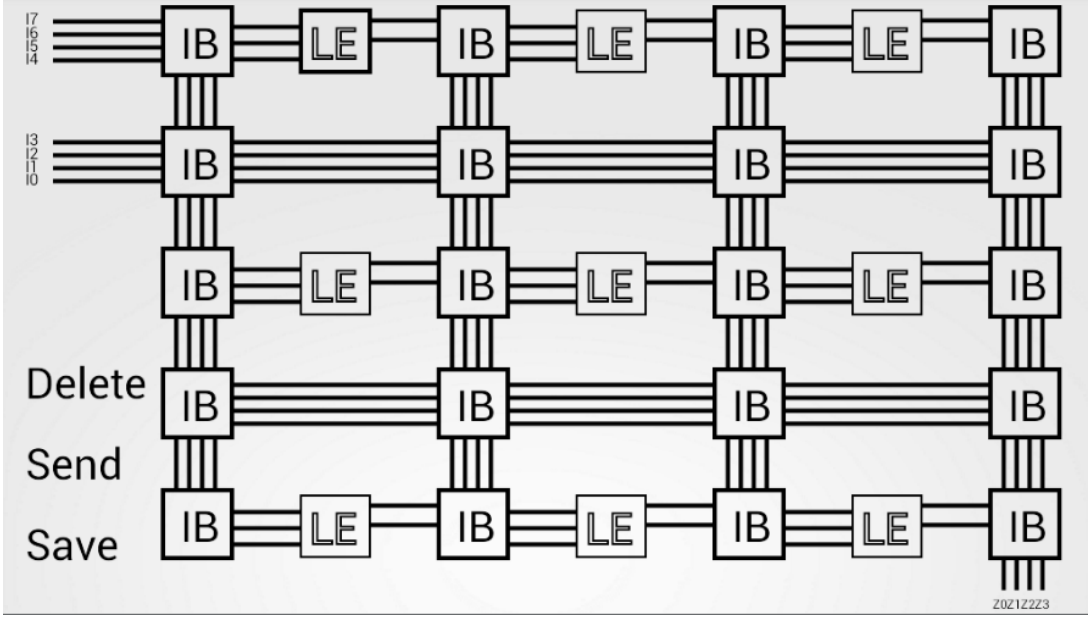

Figura 5.5 FPGA.

También tendrá la posibilidad de marcar tanto las entradas como las salidas que desea habilitar para la realización correcta del ejercicio, para ello deberá pulsar sobre las entradas situadas en la parte superior de la pantalla a la izquierda y configurarlas como desee.

Si desea volver a ver el enunciado del ejercicio, bastará con pulsar el botón ‘atrás’ del dispositivo.

Interconection Box

Si pulsa sobre una de estas áreas accederá a una pantalla donde se mostrará de forma clara las conexiones posibles para esa área de conexión entre pistas.

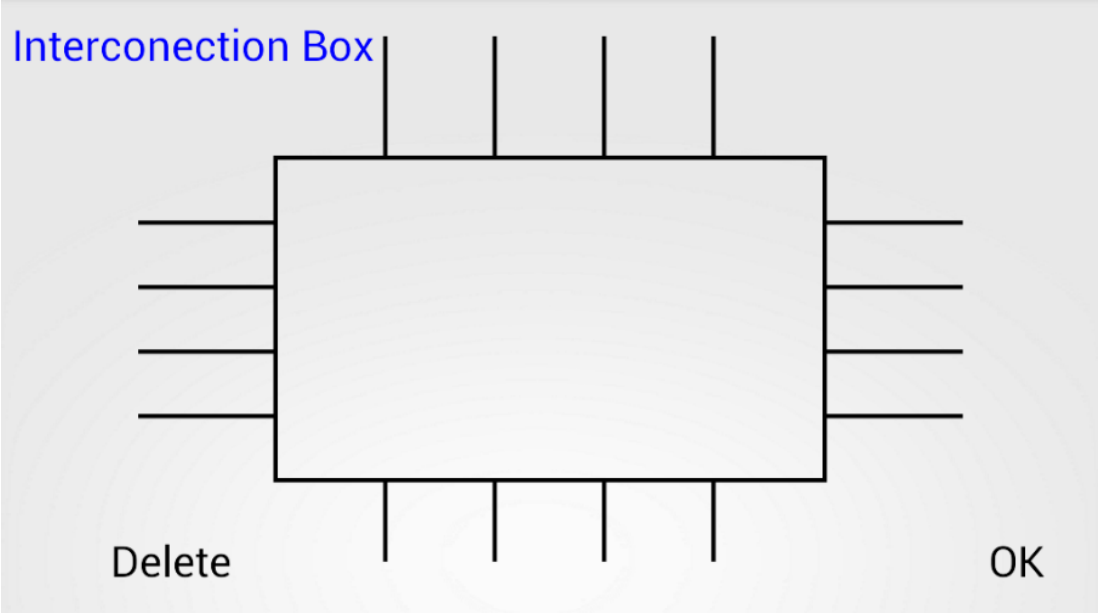

Figura 5.6 Interconection Box.

El usuario marcará las conexiones que considere oportunas, quedando dibujada en rojo su última conexión realizada.

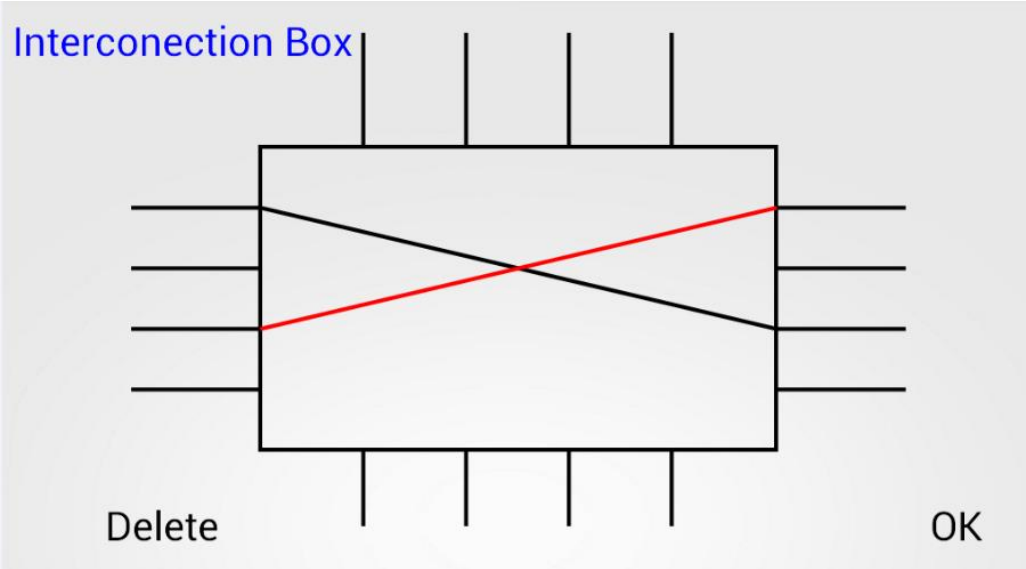

Figura 5.7 Conexiones en la Interconection Box.

Si comete algún error y desea corregirlo basta con pulsar ‘Delete’ para borrar todas las conexiones realizadas en esa área de conexión y volver a realizarlas.

Una vez se han realizado las conexiones oportunas se pulsará el botón ‘OK’ para trasladar los cambios realizados al layout principal.

Si en lugar de pulsar ‘OK’ se pulsa sobre el botón ‘atrás’ del dispositivo se volverá al layout principal pero sin guardar los cambios en las conexiones realizadas.

Logic Element

Se diseña al igual que con las Interconection Box, unas áreas para la implementación de la lógica necesaria de cara a realizar correctamente el ejercicio. Al pulsar sobre uno de estos ‘Logic Element’ se accede a una pantalla donde se muestra el layout que deberá ser completado con el fin de resolver los ejercicios. 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. De igual manera si pulsamos el botón ‘atrás’ del dispositivo se volverá al layout principal pero los cambios no serán guardados.

Así mismo, en el layout principal el usuario tiene tres botones con los que poder interactuar:

Delete

Este botón borrara todo lo realizado sobre el layout, dejándolo limpio para la realización del ejercicio. El usuario podrá pulsarlo en caso de error para poder empezar de nuevo el ejercicio.

Save

Si desea guardar el ejercicio para poder continuarlo más adelante, deberá pulsar sobre el botón ‘Save’ mostrado a la izquierda de la pantalla. Se mostrará una pantalla donde deberá escribir el nombre con que desea guardar el ejercicio y pulsar de nuevo el botón ‘Save’ para guardar el fichero o el botón ‘atrás’ del dispositivo si finalmente el usuario decide no guardar el ejercicio.

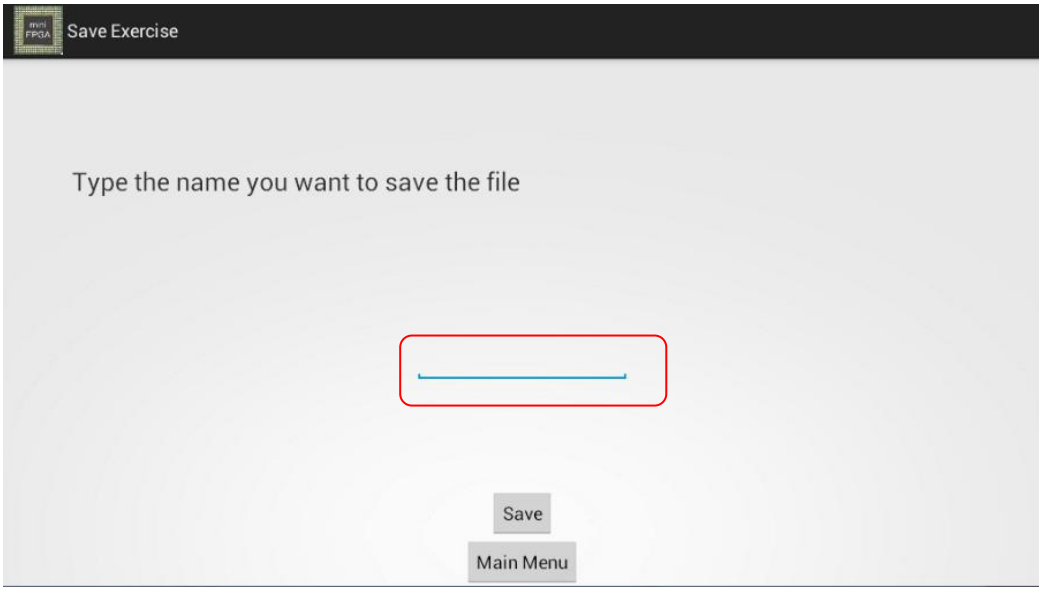

Figura 5.8 Guardar.

Send

También en la parte izquierda de la pantalla, dispone de otro botón para enviar el ejercicio por correo (Send). Una vez pulsado este botón, se mostrará una pantalla donde el usuario debe escribir el correo destinatario al que quiere enviar el ejercicio, el asunto del mensaje y si quiere acompañar el ejercicio con algún tipo de texto.

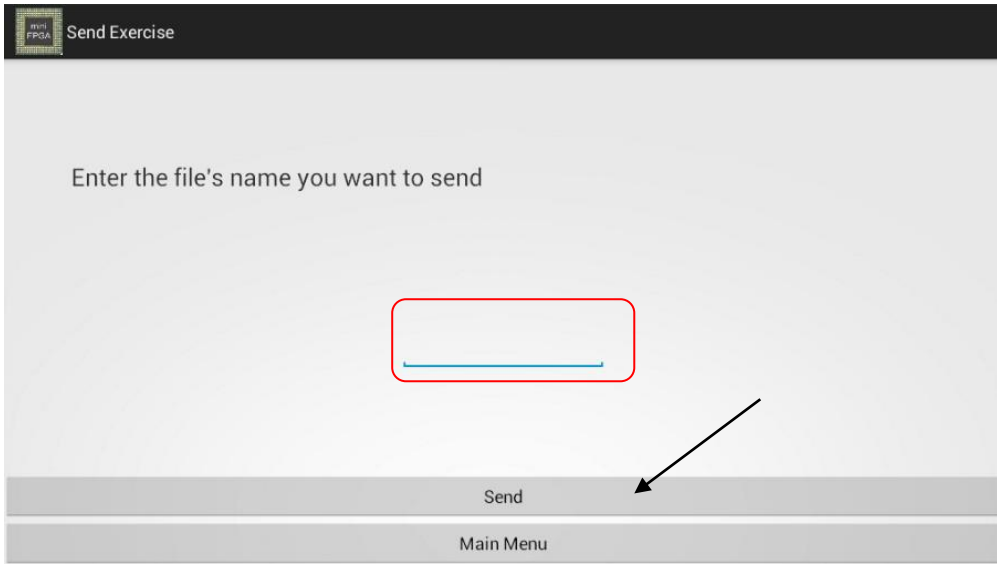

Figura 5.9 Enviar.

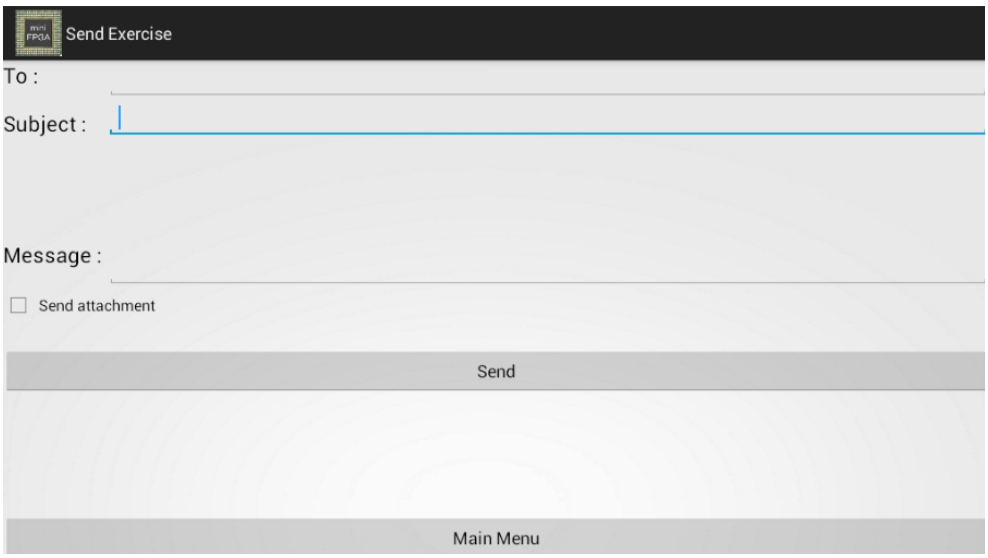

Figura 5.10 Correo.

Una vez completado el formulario, se deberá pulsar de nuevo ‘Send’ para enviar el ejercicio o el botón ‘atrás’ del dispositivo si finalmente decide no enviarlo y volver al layout principal.

* 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.

Load

Si el usuario quiere importar un ejercicio guardado previamente pulsará sobre este botón.
Se encontrará ante un layout en el que deberá escribir un nombre de un ejercicio que previamente se haya guardado en el mismo dispositivo. 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.

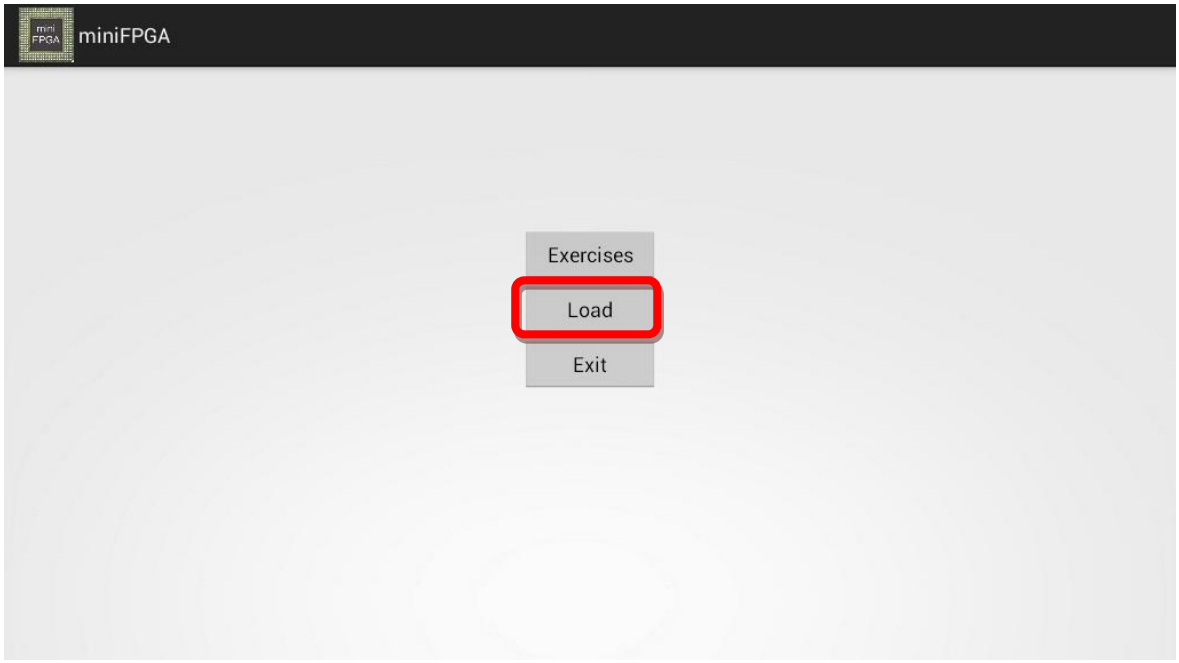

Figura 5.11 Cargar.

Una vez se haya importado el archivo, el usuario podrá continuar con la resolución del ejercicio con todos los contenidos que tenía anteriormente. Podrá utilizar todas las funciones de la aplicación como si fuera un ejercicio nuevo (modificarlo, guardarlo, enviarlo, acceder a las Interconection Box y Logic Element y modificarlas) excepto la de poder volver a ver el enunciado del ejercicio que está realizando.

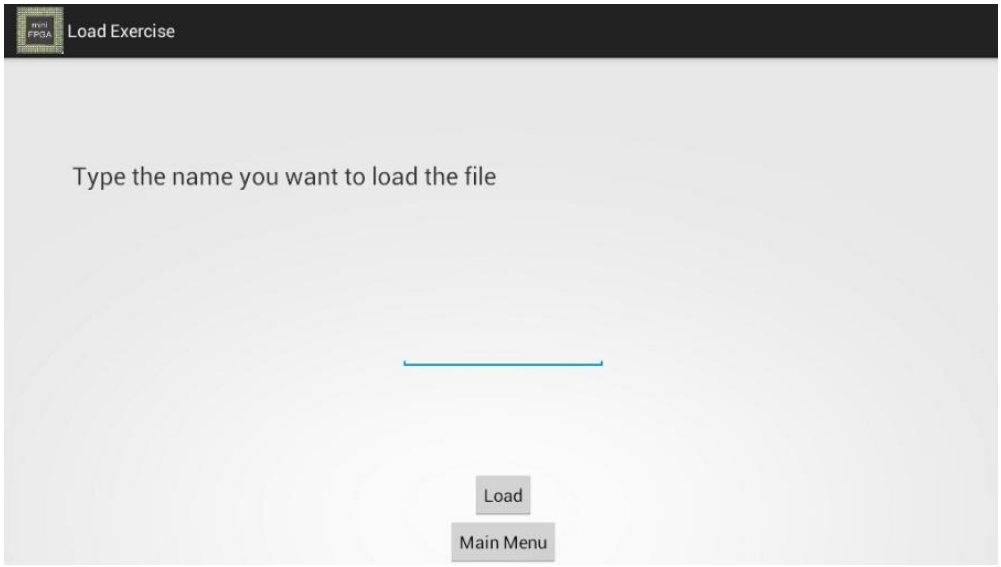

Figura 5.12 Cargar archivo.

Exit

Si lo que desea es salir de la aplicación deberá pulsar este botón en el menú principal.

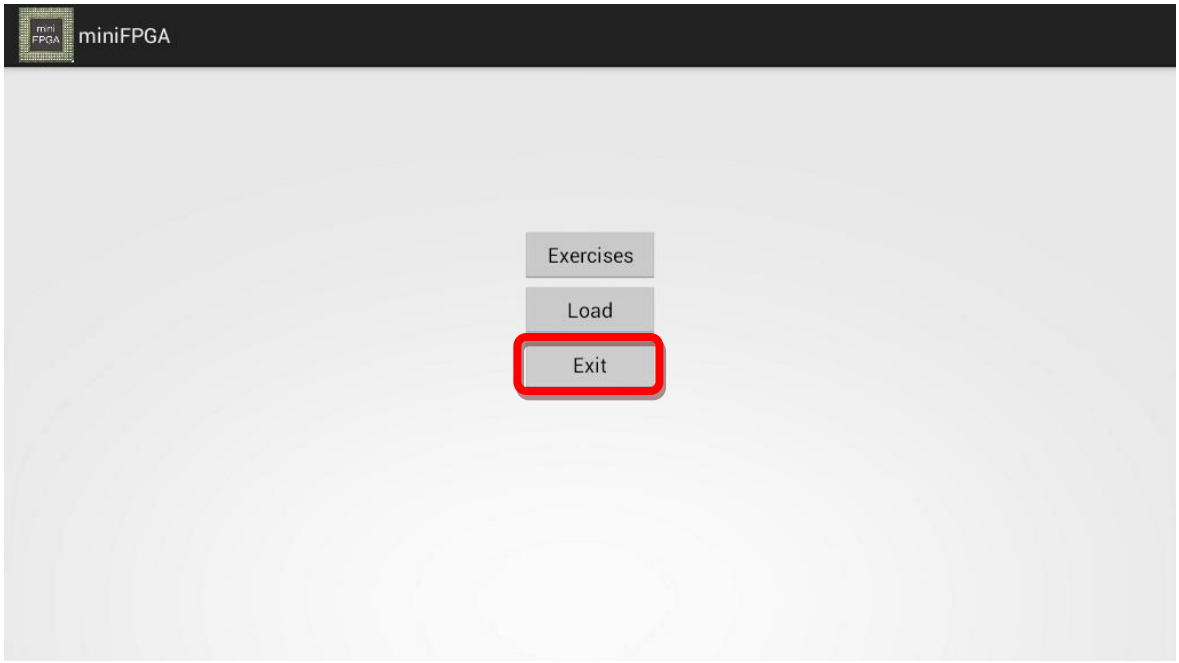

Figura 5.13 Salir.

Apéndice II

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.

Exercise 1

Este .java es el que se utiliza para la realización de los ejercicios. De forma estándar se pinta el layout de una FPGA haciendo clickables las áreas necesarias. Al pulsar sobre la realización de un ejercicio en la pantalla de listado de ejercicios, se crea un fichero específico para ese ejercicio, de manera que todos los cambios realizados en él, quedarán escritos en ese fichero para poder guardarlo, enviarlo o cambiar de ejercicio.

Se declaran variables para pintar de forma automática el layout y se cargan los datos de los archivos preferencias (si los hubiera) debido al posible cambio continuo de pantalla (entre LE o IB)

```
protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            paint = new Paint();
            //paint.setColor(Color.RED);
            paint.setStrokeWidth(6);
            paint.setTextSize(20);
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(new ViewOne(this));

            LineValue= new int[145];

            InitialPointsX = new int[4];
            InitialPointsY = new int[4];
            FinalPointsX = new int[4];
            FinalPointsY = new int[4];

            InitialIndex2004 = new int[2];
            FinalIndex2004 = new int[2];

            InitialIndex0434 = new int[4];
            FinalIndex0434 = new int[4];

            InitialIndex4034 = new int[4];
            FinalIndex4034 = new int[4];

            InitialIndex0430 = new int[3];
            FinalIndex0430 = new int[3];
```

```
            EnableText= new int[5];
            EnableText2034 = new int[2];
            EnableText2404 = new int[2];

SharedPreferences io = getSharedPreferences("PreferencesFileIO",
Context.MODE_PRIVATE);

LoadPreferencesIO(io);

SharedPreferences multiplexor = getSharedPreferences("PreferencesFile",
Context.MODE_PRIVATE);

LoadPreferences(multiplexor);

SharedPreferences multiplexor4404 = getSharedPreferences("PreferencesFile4404",
Context.MODE_PRIVATE);

LoadPreferences4404(multiplexor4404);

SharedPreferences multiplexor2434 = getSharedPreferences("PreferencesFile2434",
Context.MODE_PRIVATE);

LoadPreferences2434(multiplexor2434);

SharedPreferences multiplexor2034 = getSharedPreferences("PreferencesFile2034",
Context.MODE_PRIVATE);

LoadPreferences2034(multiplexor2034);

SharedPreferences multiplexor2430 = getSharedPreferences("PreferencesFile2430",
Context.MODE_PRIVATE);

LoadPreferences2430(multiplexor2430);

SharedPreferences multiplexor2404 = getSharedPreferences("PreferencesFile2404",
Context.MODE_PRIVATE);

LoadPreferences2404(multiplexor2404);

SharedPreferences multiplexor2004 = getSharedPreferences("PreferencesFile2004",
Context.MODE_PRIVATE);

LoadPreferences2004(multiplexor2004);

SharedPreferences multiplexor0434 = getSharedPreferences("PreferencesFile0434",
Context.MODE_PRIVATE);

LoadPreferences0434(multiplexor0434);

SharedPreferences multiplexor0444 = getSharedPreferences("PreferencesFile0444",
Context.MODE_PRIVATE);

LoadPreferences0444(multiplexor0444);

SharedPreferences multiplexor0430 = getSharedPreferences("PreferencesFile0430",
Context.MODE_PRIVATE);

LoadPreferences0430(multiplexor0430);
```

```
SharedPreferences multiplexor4034 = getSharedPreferences("PreferencesFile4034",
Context.MODE_PRIVATE);

LoadPreferences4034(multiplexor4034);

SharedPreferences LE = getSharedPreferences("PreferencesLE",
Context.MODE_PRIVATE);

LoadPreferencesLE(LE);

}
```

Se utiliza el método onTouchEvent para detector la pulsación sobre áreas clickables, guardando en tal caso banderas en el archivo de preferencias correspondiente. Como ya sabemos dónde hemos pintado las líneas, utilizamos las mismas variables para hacer clickables dichas áreas.

```
public boolean onTouchEvent(MotionEvent event)
            {

      Display display = getWindowManager().getDefaultDisplay();

      DisplayMetrics metrics = new DisplayMetrics();
      getWindowManager().getDefaultDisplay().getMetrics(metrics);

      int w = metrics.widthPixels;
      int h = metrics.heightPixels;

      int x1= (int) (w*0.15);//cada 250
      int dc = (int) (w/16);

      int yi1=(int) (0.5*h/24);
      int yf1=     (int) (3*h/24);

        if(event.getX() > x1 && event.getX() < x1+dc &&  event.getY() > yi1 &&
                              event.getY() < yf1)
{


SharedPreferences multiplexor = getSharedPreferences("PreferencesFile4034",
Context.MODE_PRIVATE);

SavePreferences4034(multiplexor);

SharedPreferences value = getSharedPreferences("PreferencesFileValue",
Context.MODE_PRIVATE);

SavePreferencesValue(value);

Intent intent = new Intent(Exercise1.this, IB4034Activity.class);
finish();
startActivity(intent);


}
```

En el método onDraw se pinta las líneas y textos correspondientes según los archivos preferencias.

```
public void onDraw(Canvas canvas) {

	Display display = getWindowManager().getDefaultDisplay();

	DisplayMetrics metrics = new DisplayMetrics();
	getWindowManager().getDefaultDisplay().getMetrics(metrics);

	int w = metrics.widthPixels;
	int h = metrics.heightPixels;

	IndiceIBY = 0;

	paint.setColor(Color.BLACK);
	paint.setStyle(Style.FILL);
	paint.setTextSize(40);


	int x_1=(int) (w*0.025);
	int y_1 = (int) (h/2 + (h/8));
	canvas.drawText("Delete", x_1, y_1, paint);

	int x_2=(int) (w*0.025);
	int y_2 = (int) ((h/2) + (h/4));
	canvas.drawText("Send", x_2, y_2, paint);

	int x_3=(int) (w*0.025);
	int y_3 = (int) ((h/2) + (3*h/8));
	canvas.drawText("Save", x_3, y_3, paint);

	paint.setStrokeWidth(4);
	paint.setStyle(Style.STROKE);

	////////////////////////////////////
	/////                          /////
	////         FPGA             /////
	///                           /////
	////////////////////////////////////


	//INPUTS
	for(int i=0,j=0;i<(3.5*h/48);i=(int) (i+(h/48)),j++)
	{
		int x=(int) (w*0.05);
		int x2=      (int) (w*0.15);
		int in = (int) (x-(w*0.025));
		int y = (int) (h/24);
		int y2 = (int) ((3*h/16)+(h/24));
		int o = (int) (0.002*h);

		int yi = (int) ((int) (h/24)+(0.005*h));
		int yi2 = (int) ((3*h/16)+(h/24) +(0.005*h));
		canvas.drawLine(x, y+i, x2, y+i, paint);
		canvas.drawLine(x, y2+i, x2, y2+i, paint);
```

```
paint.setTextSize(15);
paint.setStrokeWidth(1);
paint.setStyle(Style.FILL);
paint.setColor(Color.BLACK);

if (j==0)
{
	if(inputoutput[7] == 1)
	{
		paint.setColor(Color.RED);
		canvas.drawText("I7",in , yi, paint);//Input
	}
	if(inputoutput[7] == 0)
	{
		paint.setColor(Color.BLACK);
		canvas.drawText("I7",in , yi, paint);//Input
	}
}

if (j==1)
{
	if(inputoutput[6] == 1)
	{
		paint.setColor(Color.RED);
		canvas.drawText("I6",in , yi+i+o, paint);//Input
	}
	if(inputoutput[6] == 0)
	{
		paint.setColor(Color.BLACK);
		canvas.drawText("I6",in , yi+i+o, paint);//Input
	}
}

if (j==2)
{
	if(inputoutput[5] == 1)
	{
		paint.setColor(Color.RED);
		canvas.drawText("I5",in , yi+i+o, paint);//Input
	}
	if(inputoutput[5] == 0)
	{
		paint.setColor(Color.BLACK);
		canvas.drawText("I5",in , yi+i+o, paint);//Input
	}
}

if (j==3)
{
	if(inputoutput[4] == 1)
	{
		paint.setColor(Color.RED);
		canvas.drawText("I4",in , yi+i+o, paint);//Input
	}
	if(inputoutput[4] == 0)
	{
		paint.setColor(Color.BLACK);
		canvas.drawText("I4",in , yi+i+o, paint);//Input
```

```
        }
    }

    //UNION ENTRE COLUMNAS 1-3-5-7
    for(int i=0;i<(3.5*w/80);i=i+(w/80))
    {
        for(int j=0;j<(11*h/16);j=j+(int) (3*h/16))
        {
            int dc = (int) (w/16);
            int x1= (int) (w*0.15)+(dc/5);
            int x2= (int) (w*0.400)+(dc/5);
            int x3= (int) (w*0.65)+(dc/5);
            int x4= (int) (w*0.900)+(dc/5);

            int y1 = (3*h/24);
            int y2 =(int) ((int) (3*h/16)+(0.5*h/24));

            paint.setStrokeWidth(4);
            canvas.drawLine(x1+i, y1+j, x1+i, y2+j, paint);
            canvas.drawLine(x2+i, y1+j, x2+i, y2+j, paint);
            canvas.drawLine(x3+i, y1+j, x3+i, y2+j, paint);
            canvas.drawLine(x4+i, y1+j, x4+i, y2+j, paint);

        }
    }

    //UNION 1-2 3-4 5-6
    for(int i=0;i<(6*h/96);i= (int) (i+(2.5*h/96)))
    {
        for(int j=0;j<(3*w/4);j=j+(w/4))
        {
            int x1= (int) (w*0.15);
            int x2=(int) (w*0.275);
            int dc = (int) (w/16);

            int y1 = (int) (0.5*h/24)+(h/32);
            int y2 = (int) ((int) (0.5*h/24)+(6*h/16)+(h/32));
            int y3 = (int) (0.5*h/24)+(12*h/16)+(h/32);

            paint.setStrokeWidth(4);
            canvas.drawLine(x1+dc+j, y1+i, x2+j, y1+i, paint);
            canvas.drawLine(x1+dc+j, y2+i, x2+j, y2+i, paint);
            canvas.drawLine(x1+dc+j, y3+i, x2+j, y3+i, paint);
        }
    }

    //COLUMAS 2-4-6
    for(int i=0;i<(13*h/16);i=i+(6*h/16))
    {
        for(int j=0;j<(0.75*w);j=(int) (j + (w*0.25)))
        {
            int x=(int) (w*0.275);
            int dc = (int) (w/16);

            int yi=(int) ((int) (0.875*h/16)-(0.65*h/24));
            int yf=      (int) ((int) (2.375*h/16)-(0.65*h/24));
            int y3 = (int) (0.5*h/16)+(9*h/128);

            canvas.drawRect(x+j, yi+i, x+dc+j, yf+i, paint);
```

```
                    paint.setStrokeWidth(2);
                    paint.setTextSize(40);
                    canvas.drawText("LE", (x+(dc/8))+j, y3+i, paint);
                    paint.setStrokeWidth(4);

              }

       }
```

IB

Cada clase de las Interconection Box funciona de la misma manera con la única diferencia de las entradas o salidas que tenga por cada lado. Para ello se declaran las vías que debe tener la IB por cada lado, se pintan y se declaran clickables. Posteriormente se declara el método para capturar cada pulsación sobre la pantalla y se guarda el valor para pintar las conexiones realizadas.

```
public class IB2430Activity extends Activity{


       int LineValue[];
       //Para un array unidimensional (un vector)
       int vector_area_initial[];
       int vector_area_final[];

       int c=0,d=0;
       int marca=0;
       int IB2430Number;
       int CompleteLine = 0, EnableText2430[];


       public void OK(){

              Intent intent = new Intent(IB2430Activity.this, Exercise1.class);
              finish();
              startActivity(intent);
              return;
       }


       public void SavePreferences2430(SharedPreferences multiplexor2430)
       {
              SharedPreferences.Editor editor = multiplexor2430.edit();

       for(int i=0;i<8;i++)
       {
              editor.putInt("InitialIndex_"+i,InitialCompleteLineIndex[i]);
              editor.putInt("FinalIndex_"+i,FinalCompleteLineIndex[i]);
              editor.commit();

       }


              for(int n=0; n<2; n++){
```

```
                    editor.putInt("EnableText_"+n,EnableText2430[n]);
                    editor.commit();
             }

             OK();

      }

      public void SavePreferencesValue(SharedPreferences value)
      {
             SharedPreferences.Editor editor = value.edit();


             for(int i=0;i<145;i++)
             {
                    editor.putInt("LineValue_"+i,LineValue[i]);
                    editor.commit();

             }

SharedPreferences multiplexor2430 = getSharedPreferences("PreferencesFile2430",
MODE_PRIVATE);

SavePreferences2430(multiplexor2430);
      }


      @Override
      protected void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);


             paint = new Paint();
             paint.setStrokeWidth(6);
             paint.setTextSize(20);
             requestWindowFeature(Window.FEATURE_NO_TITLE);
             setContentView(new ViewOne(this));
             Display display = getWindowManager().getDefaultDisplay();

             DisplayMetrics metrics = new DisplayMetrics();
             getWindowManager().getDefaultDisplay().getMetrics(metrics);

             int w = metrics.widthPixels;
             int h = metrics.heightPixels;

             int x1=(int) (w*0.25);
             int dc = (int) (w*0.5);
             int yi1=(int) (h/4)+(3*h/40);
             int yf1=(int) (h/4)+(5*h/40);
             int x2=(int) (w*0.75);
             int xf2 = (int) (w*0.875);

             int xi1=(int) ((int) (w/4)+(3*w/40));
             int xf1=(int) ((int) (w/4)+(5*w/40));


            int valor;

             vector_area_initial = new int[3];
```

```
for(int i=0,j=0; i<(2.5*h/10) ;i=(int) (i+(h/10)),j++)
{
	vector_area_initial[j] = yi1+i;
}


vector_area_final = new int[3];
for(int i=0,j=0; i<(2.5*h/10) ;i=(int) (i+(h/10)),j++)
{
vector_area_final[j] = yf1+i;
}

vector_area_initial_X = new int[4];
for(int i=0,j=0; i<(3.5*w/10) ;i=(int) (i+(w/10)),j++)
{
vector_area_initial_X[j] = xi1+i;
}

vector_area_final_X = new int[4];
for(int i=0,j=0; i<(3.5*w/10) ;i=(int) (i+(w/10)),j++)
{
vector_area_final_X[j] = xf1+i;
}


cixy = new int[4];
ciyx = new int[2];
cfyx = new int[3];
EnableText2430 = new int[2];


InitialWriteEnableX = new int[4];
InitialWriteEnableY = new int[2];
FinalWriteEnableY = new int[3];

InitialCompleteLineX = new int[4];
FinalCompleteLineX = new int[4];
InitialCompleteLineY = new int[4];
FinalCompleteLineY = new int[4];

FinalCompleteLineIndex = new int[8];
InitialCompleteLineIndex = new int[8];

LineValue = new int[145];

for(int n=0; n<2; n++){
	ciyx[n] = 0;
	InitialWriteEnableY[n] = 1;
}
for(int n=0; n<3; n++){
	cfyx[n] = 0;
	FinalWriteEnableY[n] = 1;
}

for(int n=0; n<4; n++){
	cixy[n] = 0;

	InitialWriteEnableX[n] = 1;
```

```
            }

            for(int n=0; n<4; n++){

                  InitialCompleteLineX[n] = 0;
                  FinalCompleteLineX[n] = 0;
                  InitialCompleteLineY[n] = 0;
                  FinalCompleteLineY[n] = 0;
            }


SharedPreferences multiplexor2430 = getSharedPreferences("PreferencesFile2430",
Context.MODE_PRIVATE);

LoadPreferences2430(multiplexor2430);

SharedPreferences value = getSharedPreferences("PreferencesFileValue",
Context.MODE_PRIVATE);

LoadPreferencesValue(value);


            EnableText2430[IB2430Number-1]= 0;



      }


      public boolean onKeyDown(int keyCode, KeyEvent event) {

            Intent intent = new Intent(this, Exercise1.class);
            finish();
            startActivity(intent);
            return super.onKeyDown(keyCode, event);
      }

      public class ViewOne extends View
      {
            @SuppressLint("NewApi")
            public ViewOne (Context context) {
                  super(context);
                  Display display = getWindowManager().getDefaultDisplay();
                  Point size = new Point();
                  display.getSize(size);
            }






@Override
public boolean onTouchEvent(MotionEvent event)
{

      Display display = getWindowManager().getDefaultDisplay();
```

```
      DisplayMetrics metrics = new DisplayMetrics();
      getWindowManager().getDefaultDisplay().getMetrics(metrics);

      int w = metrics.widthPixels;
      int h = metrics.heightPixels;
      int x1=(int) (w*0.25);
      int xi1=(int) (w*0.125);
      int dc = (int) (w*0.125);
      int y1=(int) (h/4)+(h/10);
      int x2=(int) (w*0.75);
      int xf2 = (int) (w*0.875);



      int yi1=(int) (h/16);
      int yf1 = (int) (h/4);

      int yi2=(int) (h*0.75);
      int yf2 = (int) (h*0.875);

      int s=0;

      for(int i=0; i<4; i++ ){

if(event.getX() > vector_area_initial_X[i] && event.getX() <
vector_area_final_X[i] &&  event.getY() > yi1 &&  event.getY() < yf1 )
{
      if(InitialWriteEnableX[i]==1)
      {
             bandera++;

            cixy[i] = vector_area_initial_X[i]+(w/40);
            ciy = yf1;


            if(bandera == 1)
            {
                  if(IB2430Number == 1)
                  {
                        for(c=0, marca = 0 ; c<4;c++)
                        {
                        if(InitialCompleteLineIndex[c] ==0 && marca == 0)
                        {
                              InitialCompleteLineIndex[c]=i+9;
                              marca = 1;

                        }
                  }
            }


      if(IB2430Number == 2)
      {
            for(c=4, marca = 0 ; c<8;c++)
            {
                  if(InitialCompleteLineIndex[c] ==0 && marca == 0)
                  {
```

```
                        InitialCompleteLineIndex[c]=i+9;
                        marca = 1;
                    }
                }
            }

}


        if(bandera == 2)
        {
            CompleteLine = 1;
            if(IB2430Number == 1)
            {
                for(c=0, marca = 0 ; c<4;c++)
                {
                    if(FinalCompleteLineIndex[c] ==0 && marca == 0)
                    {
                        FinalCompleteLineIndex[c]=i+9;
                        marca = 1;
                    }
                }
        }

        if(IB2430Number == 2)
        {
            for(c=4, marca = 0 ; c<8;c
            {
                if(FinalCompleteLineIndex[c] ==0 && marca == 0)
                {
                    FinalCompleteLineIndex[c]=i+9;
                    marca = 1;
                }
            }
        }

}









            //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//
            //     DELETE BUTTON           //
            //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//

            int xdel=(int) (w*0.1);
            int ydel = (int) (h*0.9);
```

```
            if(event.getX() > xdel && event.getX() < xdel+dc &&  event.getY() >
(ydel-(h/10)) &&  event.getY() < ydel){


                  for(int n=0; n<2; n++){
                        ciyx[n] = 0;
                        InitialWriteEnableY[n] = 1;
                  }
                  for(int n=0; n<3; n++){
                        cfyx[n] = 0;
                        FinalWriteEnableY[n] = 1;
                  }

                  for(int n=0; n<4; n++){
                        cixy[n] = 0;

                        InitialWriteEnableX[n] = 1;

                  }

                  for(int n=0; n<4; n++){

                        InitialCompleteLineX[n] = 0;
                        FinalCompleteLineX[n] = 0;
                        InitialCompleteLineY[n] = 0;
                        FinalCompleteLineY[n] = 0;

                  }

                  cix = 0;
                  cfx = 0;
                  ciy = 0;
                  bandera = 0;
                  CompleteLine=0;
            }


            }// termina bucle for*/



      invalidate();// Llama a OnDraw

      return super.onTouchEvent(event);
}







      public void onDraw(Canvas canvas) {

      Display display = getWindowManager().getDefaultDisplay();
      DisplayMetrics metrics = new DisplayMetrics();
      getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

```
        int w = metrics.widthPixels;
        int h = metrics.heightPixels;


        paint.setColor(Color.BLACK);
        paint.setStyle(Style.FILL);


        paint.setStrokeWidth(3);
        paint.setTextSize(40);
        int xdel=(int) (w*0.1);
        int ydel = (int) (h*0.9);
        canvas.drawText("Delete", xdel, ydel, paint);

        int xok=(int) (w*0.9);
        int yok=(int) (0.9*h);
        canvas.drawText("OK", xok, yok, paint);

        paint.setColor(Color.BLUE);
        int xIB=(int) (w*0.01);
        int yIB=(int) (0.1*h);
        canvas.drawText("Interconection Box", xIB, yIB, paint);
        paint.setColor(Color.BLACK);

        paint.setStrokeWidth(6);
        paint.setTextSize(20);
        paint.setStyle(Style.STROKE);

        int x1=(int) (w*0.25);
        int dc = (int) (w*0.5);
        int yi = (h/4);
        int yf = (3*h/4);

        //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//
        //%%      INTERCONECTION BOX            %%//
        //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%//

        paint.setStrokeWidth(4);
        canvas.drawRect(x1, yi, x1+dc, yf, paint);

        for(int i=0; i<(1.5*h/10) ;i=(int) (i+(h/10)))
        {
               int xi1=(int) (w*0.125);
               int xf1 = (int) (w*0.25);
               int y1=(int) (h/4)+(h/10);

               canvas.drawLine(xi1, y1+i, xf1, y1+i, paint);
        }




        for(int i=0; i<(2.5*h/10) ;i=(int) (i+(h/10)))
        {
               int x2=(int) (w*0.75);
               int xf2 = (int) (w*0.875);

               int y1=(int) (h/4)+(h/10);
```

```
            canvas.drawLine(x2, y1+i, xf2, y1+i, paint);
      }

      for(int i=0; i<(3.5*w/10) ;i=(int) (i+(w/10)))
      {
            int xi1=(int) ((int) (w/4)+(w/10));

            int y1=(int) (h/16);
            int yf1 = (int) (h/4);


            canvas.drawLine(xi1+i, y1, xi1+i, yf1, paint);
      }
```

PRESUPUESTO

1) Ejecución Material

- **Uso de ordenador personal (Software incluido)**
  (1.500 € / 36) * 25 meses = 1042 €
- **Dispositivo móvil para la realización de test 300 €**
- **Material de oficina 50 €**
- **Total de ejecución material 1.392 €**

2) Gastos generales

- **16% sobre Ejecución Material 223 €**

3) Beneficio Industrial

- **6% sobre Ejecución Material 84 €**

4) Honorarios Proyecto

- **800 horas a 15 € / hora 12.000 €**

5) Material fungible

- **Gastos de impresión 60 €**
- **Encuadernación 200 €**

6) Subtotal del presupuesto

- **Subtotal Presupuesto 13.959 €**

7) I.V.A. aplicable

- **21% Subtotal Presupuesto 2.931.4 €**

8) Total presupuesto

- **Total Presupuesto 16.890,4 €**

Madrid, 04 de Abril de 2016

El Ingeniero Jefe de Proyecto

Fdo.: Adrián Moreno Villalón
Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un Tutorial Interactivo Android sobre Mapeado en FPGAs. 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.