

**UNIVERSIDAD POLITÉCNICA DE MADRID**  
**ESCUELA TÉCNICA SUPERIOR DE**  
**INGENIEROS DE TELECOMUNICACIONES**



**MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS  
ELECTRÓNICOS**

**TRABAJO FIN DE MÁSTER**

**DISEÑO E IMPLEMENTACIÓN DE UN  
OSCILOSCOPIO MODULAR BASADO EN  
FPGA.**

**MARTÍN NOVOA PELLO  
JULIO 2021**



# MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS

## TRABAJO FIN DE MÁSTER

Titulo: DISEÑO E IMPLEMENTACIÓN DE UN OSCILOSCOPIO MODULAR BASADO EN FPGA.

Autor: MARTÍN NOVOA PELLO

Tutor: AMADEO DE GRACIA HERRANZ

Ponente:

Departamento: DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

## COMITÉ DE EVALUACIÓN

Presidente: .....

Vocal: .....

Secretario/a: .....

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de:

.....

Madrid, a ..... de ..... de 2021



UNIVERSIDAD POLITÉCNICA DE MADRID  
ESCUELA TÉCNICA SUPERIOR DE  
INGENIEROS DE TELECOMUNICACIONES



MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS  
ELECTRÓNICOS

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UN  
OSCILOSCOPIO MODULAR BASADO EN  
FPGA.

AUTOR: MARTÍN NOVOA PELLO

TUTOR: AMADEO DE GRACIA HERRANZ

JULIO 2021



## Resumen

Un osciloscopio es un elemento fundamental a la hora de trabajar con señales eléctricas, pudiendo medir un gran número de fenómenos físicos si se cuenta con el sensor adecuado. Este dispositivo tiene un gran peso en el campo de la electrónica, donde se emplea tanto en el diseño como en la reparación de todo tipo de sistemas electrónicos.

Este Trabajo Fin de Máster tiene como objetivo desarrollar un osciloscopio digital capaz de ser implementado de manera sencilla, teniendo como único requisito disponer de una FPGA y un monitor en el que ver las señales muestreadas. Además, se desarrollan funcionalidades básicas de un osciloscopio como tener dos canales, *trigger* con varios modos de funcionamiento y representación numérica de las diferentes magnitudes de las señales. Finalmente, se representan los datos en un monitor conectado a la FPGA por el puerto VGA, por lo que será necesario realizar el diseño de dicha interfaz gráfica. Para conseguir todo esto, se diseñaran todos los elementos necesarios para la implementación de un osciloscopio, desde la adquisición de datos hasta la visualización en las señales en una pantalla. Esto se realizará mediante el lenguaje de descripción *hardware* VHDL, con el fin de implementar el sistema en una FPGA.

## Abstract

Oscilloscopes are essential devices when working with electrical signals, being able to measure a large number of physical phenomena when the correct sensor is available. This device has a big importance in the field of electronics, where it is used both in the design and in the repairing of all kinds of electronic systems.

The aim of this Master's Thesis is to develop a digital oscilloscope capable of being implemented in a simple way, with the only requirement of having an FPGA and a monitor in which the signals will be shown. Besides, basic functionalities of an oscilloscope are developed, such as having two channels, trigger with several operating modes and numerical representation of the signal magnitudes. Lastly, the data are represented on a monitor connected to the FPGA through the VGA port. To achieve this, all the necessary elements needed for the implementation of an oscilloscope will be designed, for the acquisition of data for the visualization of the signals on the screen. The description language use to do this is VHDL, in order to implement the system in an FPGA.



# Índice

Resumen . . . . .	7
Abstract . . . . .	7
Contenidos del TFM . . . . .	9
Listado de figuras . . . . .	11
Listado de tablas . . . . .	13
<b>I MEMORIA . . . . .</b>	<b>15</b>
<b>1 INTRODUCCIÓN . . . . .</b>	<b>17</b>
1.1 Introducción . . . . .	17
1.2 Motivación . . . . .	17
1.3 Objetivo . . . . .	18
1.4 Metodología . . . . .	18
<b>2 DEFINICIONES Y ABREVIATURAS . . . . .</b>	<b>21</b>
<b>3 ANTECEDENTES . . . . .</b>	<b>23</b>
3.1 Osciloscopio . . . . .	23
3.1.1 Osciloscopios analógicos . . . . .	23
3.1.2 Osciloscopios digitales . . . . .	23
3.1.3 Principales funcionalidades de los osciloscopios . . . . .	25
3.2 Programación de las FPGAs . . . . .	26
<b>4 ANÁLISIS DE LAS SOLUCIONES . . . . .</b>	<b>27</b>
4.1 Selección de la FPGA . . . . .	27
4.2 Diagrama general . . . . .	28
4.3 Utilización de los switches incorporados en la Nexys 4 DDR . . . . .	30
4.4 Diseño de la arquitectura . . . . .	31
4.4.1 Arquitectura basada en buses . . . . .	32
4.4.2 Arquitectura basada en elementos de interconexión . . . . .	32
4.5 Implementación genérica . . . . .	33
4.6 Diseño de la adquisición de datos . . . . .	33
4.6.1 Convertidor analógico-digital . . . . .	33
4.6.2 Memoria FIFO . . . . .	38
4.6.3 Trigger . . . . .	39
4.6.4 Memorias de los canales . . . . .	42

4.7	Diseño de la modificación de parámetros . . . . .	47
4.7.1	Antirrebotes . . . . .	47
4.7.2	Memoria opciones . . . . .	48
4.7.3	Controlador de los botones . . . . .	48
4.7.4	Display de 7 segmentos . . . . .	50
4.8	Diseño del cálculo de magnitudes . . . . .	53
4.8.1	Tensión máxima . . . . .	54
4.8.2	Tensión mínima . . . . .	54
4.8.3	Tensión pico a pico . . . . .	54
4.8.4	Tensión media . . . . .	55
4.9	Diseño del VGA . . . . .	56
4.9.1	El estándar VGA . . . . .	56
4.9.2	Representación del <i>plot</i> . . . . .	58
4.9.3	Representación de los parámetros . . . . .	63
4.9.4	Representación de símbolos . . . . .	66
4.9.5	Representación del indicador de trigger automático . . . . .	68
4.9.6	Representación de las mediciones . . . . .	69
4.10	Simulación del sistema . . . . .	71
5	RESULTADOS . . . . .	75
6	CONCLUSIONES Y LÍNEAS FUTURAS . . . . .	77
6.1	Conclusiones . . . . .	77
6.2	Líneas futuras . . . . .	77
7	Bibliografía . . . . .	79
<b>II</b>	<b>ANEXOS . . . . .</b>	<b>81</b>
A	Mapas de memoria . . . . .	83
A.1	Mapa de la memoria de caracteres . . . . .	83
A.2	Mapa de la memoria de símbolos . . . . .	84
B	Presupuesto . . . . .	85
B.1	Recursos amortizables . . . . .	85
B.2	Recursos no amortizables . . . . .	85
B.3	Mano de obra . . . . .	85
B.4	Coste total . . . . .	86
C	Impacto Ético, Social, Económico y Medioambiental . . . . .	87
C.1	Impacto Ético . . . . .	87
C.2	Impacto Económico . . . . .	87
C.3	Impacto Social . . . . .	87
C.4	Impacto Medioambiental . . . . .	87

# Listado de figuras

3.1	Esquema de funcionamiento de un osciloscopio analógico . . . . .	23
3.2	Osciloscopio digital . . . . .	24
3.3	Digital Storage Oscilloscope . . . . .	24
3.4	Digital Phosphor Oscilloscope . . . . .	25
3.5	Interpolación . . . . .	26
4.1	Placa de desarrollo Nexys 4 DDR . . . . .	27
4.2	Diagrama de bloques general . . . . .	28
4.3	Asociación de las señales de entrada a los <i>switches</i> presentes en la placa . . . . .	31
4.4	Arquitectura Buses . . . . .	32
4.5	Diagrama simplificado de la adquisición de datos . . . . .	33
4.6	Puerto PMOD . . . . .	34
4.7	Diagrama ADC . . . . .	34
4.8	Registros de estado del ADC . . . . .	35
4.9	Formato dato del ADC . . . . .	36
4.10	Funciones de transferencia del ADC . . . . .	37
4.11	Máquina de estados para el control del ADC . . . . .	37
4.12	Estímulos para la simulación del ADC . . . . .	38
4.13	Simulación de la lectura del ADC . . . . .	38
4.14	Funcionamiento de una memoria FIFO . . . . .	39
4.15	Diagrama detector de flanco en la señal de entrada . . . . .	40
4.16	Máquina de estados trigger . . . . .	41
4.17	Simulación del disparo del trigger . . . . .	41
4.18	Señal original del upsampling . . . . .	42
4.19	Señal con upsample 2 . . . . .	42
4.20	Señal original del downsampling . . . . .	43
4.21	Señal con downsample 2 . . . . .	43
4.22	Máquina de estados para el control de la memoria de muestras . . . . .	44
4.23	Simulación de la lectura de la memoria FIFO - parte 1 . . . . .	45
4.24	Simulacion de la lectura de la memoria FIFO - parte 2 . . . . .	46
4.25	Simulación del downsampling . . . . .	46
4.26	Diagrama interfaz de usuario . . . . .	47
4.27	Diagrama antirrebotes . . . . .	48
4.28	Diagrama del controlador de los botones . . . . .	49

4.29	Máquina de estados del control de los botones . . . . .	50
4.30	Diagrama de funcionamiento del <i>display</i> de la Nexys 4 DDR . . . . .	51
4.31	Diagrama de la representación en el <i>display</i> de 7 segmentos . . . . .	53
4.32	Diagrama del cálculo de la tensión máxima . . . . .	54
4.33	Diagrama del cálculo de la tensión mínima . . . . .	54
4.34	Diagrama del cálculo de la tensión de pico . . . . .	55
4.35	Cuantificación del calculo de tensión media . . . . .	56
4.36	Conector <i>D-sub</i> de 15 contactos . . . . .	56
4.37	Diagrama temporal del estándar VGA . . . . .	57
4.38	Diagrama de la implementación del VGA . . . . .	58
4.39	Diagrama del generador de la rejilla . . . . .	59
4.40	Diagrama de la implementación del <i>Buffer</i> . . . . .	62
4.41	Maquina de estados del <i>Buffer</i> . . . . .	63
4.42	Ejemplo mapa de bits de los caracteres . . . . .	63
4.43	Diagrama de la representación de los voltios por división . . . . .	65
4.44	Diagrama de la representación del tiempo por división . . . . .	66
4.45	Ejemplo mapa de bits de los símbolos . . . . .	66
4.46	Diagrama de la representación de la flecha del trigger . . . . .	67
4.47	Diagrama de la representación del flanco del trigger . . . . .	68
4.48	Diagrama de la representación del indicador de <i>trigger</i> automático . . . . .	69
4.49	Diagrama de la representación de la tensión máxima . . . . .	70
4.50	Formato del fichero para la simulación de la interfaz VGA . . . . .	72
4.51	Figura generada por el VGA simulator . . . . .	72
4.52	Figura generada por el VGA simulator 2 . . . . .	73
5.1	Resultado de la interfaz gráfica . . . . .	75
5.2	Representación VPD en la placa . . . . .	76
5.3	Representación del trigger en la placa . . . . .	76
5.4	Utilización y consumo de la FPGA . . . . .	76

# Listado de tablas

4.1 Valores de <i>downsample</i> para diferentes escalas horizontales . . . . .	43
4.2 Mapa de la memoria de opciones . . . . .	48
4.3 Mapas de las memorias de VPD, TPD y decodificador de 7 segmentos . . . . .	51
4.4 Temporizaciones generales del VGA . . . . .	57
4.5 Temporización horizontal del VGA . . . . .	58
4.6 Temporización vertical del VGA . . . . .	58
4.7 Mapa de la memoria para el factor de conversión . . . . .	61
A.1 Mapa de la memoria de caracteres . . . . .	83
A.2 Mapa de la memoria de símbolos . . . . .	84
B.1 Coste de los recursos amortizables . . . . .	85
B.2 Coste de los recursos amortizables . . . . .	85
B.3 Coste de los recursos amortizables . . . . .	86
B.4 Coste de los recursos amortizables . . . . .	86



# **MEMORIA**

MARTÍN NOVOA PELLO

JULIO 2021



# 1 INTRODUCCIÓN

## 1.1. Introducción

El primer osciloscopio, tal y como los conocemos hoy en día, se creó en el año 1897 por el científico Karl Ferdinand Braun, que consistía en una adaptación del tubo de rayos catódicos para generar representaciones visuales de señales eléctricas. Posteriormente, se le incorpora un *trigger* para obtener sincronismo horizontal, lo que supone un gran avance en el desarrollo de los osciloscopios. Con el avance de la electrónica se desarrollan los osciloscopios digitales, que suponen un gran paso a la hora de trabajar con señales ya que, además de permitir su visualización, permite realizar mediciones muy precisas mediante los controles que incorpora, como pueden ser valores de tensión de pico o medio, frecuencia u otras mediciones más avanzadas en función de la gama del osciloscopio.

Por todo esto, los osciloscopios se han convertido en una herramienta imprescindible cuando trabajamos, no solo con circuitos electrónicos, sino con cualquier tipo de señal ya que, disponiendo del sensor adecuado, es posible medir un gran numero de fenómenos físicos, como pueden ser las ondas de sonido o las señales eléctricas del corazón, entre otros.

Todas aquellas personas acostumbradas a trabajar con un osciloscopio conocen cuales son las funcionalidades de estos aparatos. Sin embargo, puede ser muy interesante adentrarse en el funcionamiento interno, realizando la implementación de un osciloscopio en una placa de desarrollo de una FPGA.

## 1.2. Motivación

La motivación de este proyecto surge de un interés por el *hardware* y el *software* libre. Desde que me comencé a adentrar en el mundo de la tecnología me he dado cuenta de como ciertas tecnologías y programas son enormemente costosos. Debido a que la creación de *software* es mucho más accesible al público general, se forman grandes comunidades que alimentan ciertos programas, haciendo que cada día sean mejores y que merezcan mucho la pena, como puede ser el caso de KiCad. Este programa es un paquete de *software* libre para el diseño de circuitos impresos, que cuenta con una gran comunidad que alimenta sus librerías de componentes.

Sin embargo, debido a que el desarrollo de *hardware* requiere de mayores inversiones de tiempo y dinero, las comunidades que se forman en este ámbito son muy reducidas. Todo esto, unido a un interés por la instrumentación electrónica me hace plantearme un proyecto en el que se diseñe en *hardware* un osciloscopio que sea funcional, que se pueda implementar fácilmente y con un coste reducido y que permita realizar las mejoras iterativas que se realizan en las comunidades de *software* libre.

Este es un proyecto que ya ha sido realizado por otras personas, como puede ser el proyec-

to de Andrei Purcarus, que implemento un osciloscopio digital en una placa DE1-SoC Cyclone V y cuyo proyecto está disponible en el repositorio de GitHub: <https://github.com/Gripnook/digital-storage-oscilloscope>. A pesar de ello, creo que es un proyecto interesante, con el que se puede aprender mucho sobre diseño *hardware* y sobre osciloscopios.

### 1.3. Objetivo

El objetivo de este proyecto es diseñar un osciloscopio digital que pueda ser implementado de manera sencilla en diferentes FPGAs (*Field Progammable Gate Array*). Para ello, el diseño debe ser lo más genérico posible, evitando el uso de IP cores (núcleo de propiedad intelectual) y creando un fichero de constantes en el que se definan todos los parámetros principales en los que se basa la configuración, de manera que en este archivo se pueda configurar el mayor número de opciones posibles.

Este osciloscopio debe contar con las funcionalidades básicas de los osciloscopios digitales como son: ajuste vertical y horizontal, 2 canales con posibilidad de ser habilitados/deshabilitados, *trigger* con flanco ascendente/descendente y posibilidad de activar un modo de *trigger* automático. En cuanto al procesado de las señales, se realizarán cálculos de los valores máximos, mínimos, medios y pico a pico de las señales de entrada; que podrán ser habilitados/-deshabilitados por el usuario, aunque sería interesante realizar cálculos más complejos como el de la frecuencia o realizar la transformada de Fourier.

Por último, se deberá diseñar la interfaz de usuario. El osciloscopio deberá mostrar la señal, los resultados de las mediciones y los parámetros de configuración en vivo en un monitor que funcione con el estándar VGA (*Video Graphics Array*). Los parámetros de configuración en vivo son aquellos que debe modificar el usuario en función de sus necesidades, como los voltios o el tiempo por división. Por otro lado, la comunicación del usuario con el osciloscopio se realizará mediante botones y *switches*, con los que se ajustaran los parámetros de configuración en vivo que se han comentado anteriormente. Para que la modificación se realice de manera óptima se debe mostrar el valor instantáneo del parámetro que se esta modificando.

### 1.4. Metodología

Para la realización del proyecto se seguirán las siguientes fases de desarrollo:

1. Estudio y lectura de documentación acerca de tipos de osciloscopios digitales y principales funcionalidades.
2. Selección del tipo de osciloscopio a implementar en la FPGA.
3. Diseño de la arquitectura.
4. Diseño de los bloques que componen el proyecto.
5. Implementación en la placa de desarrollo Nexys 4 DDR.

- Adquisición de las señales mediante el ADC presente en la FPGA Artix 7.
- Procesamiento de las señales: cálculo de magnitudes.
- VGA: generación de la interfaz gráfica y comunicación con el monitor.
- Interacción con el usuario.



## 2 DEFINICIONES Y ABREVIATURAS

- ADC: *Analogic to Digital Converter.*
- ASIC: *Application-Specific Integrated Circuit*
- AVG: *Average.*
- Ca2: Complemento a 2.
- DSP: *Digital Signal Processor.*
- FIFO: *First In, First Out.*
- FPGA: *Field Programmable Gate Arrays.*
- LUT: *LookUp Table.*
- PLD: *Programmable Logic Device*
- PLL: *Phase-Locked Loop*
- RAM: *Random Access Memory.*
- TPD: Tiempo por División.
- VGA: *Video Graphics Array.*
- VPD: Voltios por División.



### 3 ANTECEDENTES

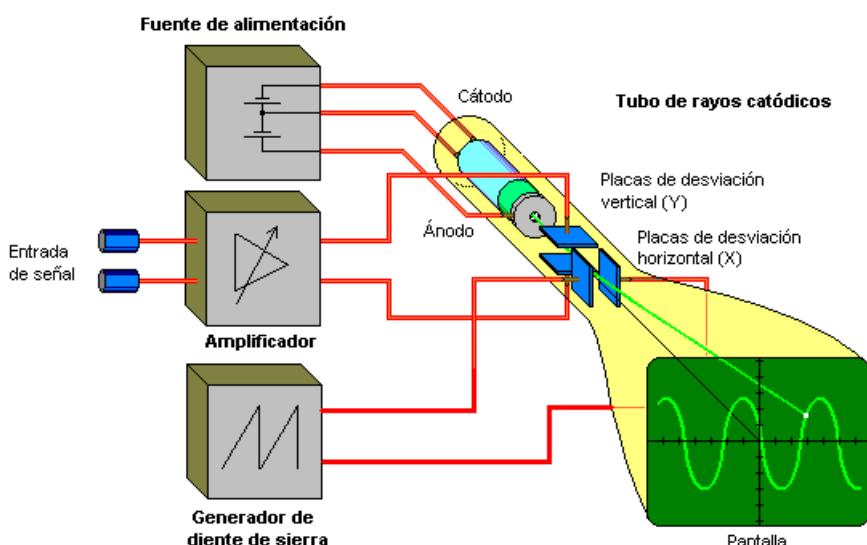
#### 3.1. Osciloscopio

Actualmente existen una gran variedad de osciloscopios, que se pueden dividir en dos grandes grupos: osciloscopios analógicos y osciloscopios digitales. En este capítulo se va a hacer una pequeña introducción a estos grandes grupos contando sus características principales.

##### 3.1.1. Osciloscopios analógicos

Los osciloscopios analógicos son aquellos que miden una señal eléctrica por medio de las desviaciones que provoca a los electrones que son disparados en un tubo de rayos catódicos.

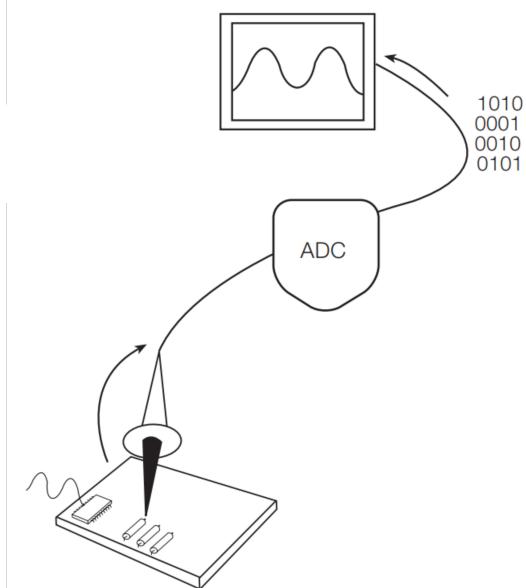
La desviación vertical de los electrones es provocada por la señal de entada y la desviación horizontal por una señal de tipo diente de sierra. La visualización de la imagen se produce por el brillo que provoca el impacto del electrón sobre una capa fluorescente que se encuentra en el interior de la pantalla. Esto supone ciertas limitaciones, como que las señales que queramos medir deben ser periódicas. Por lo tanto, no será posible medir eventos transitorios. En la figura 3.1 se puede ver un esquema simplificado del funcionamiento de este tipo de osciloscopio.



**Figura 3.1 – Esquema de funcionamiento de un osciloscopio analógico**  
(fuente: <https://es.wikipedia.org/wiki/Osciloscopio>)

##### 3.1.2. Osciloscopios digitales

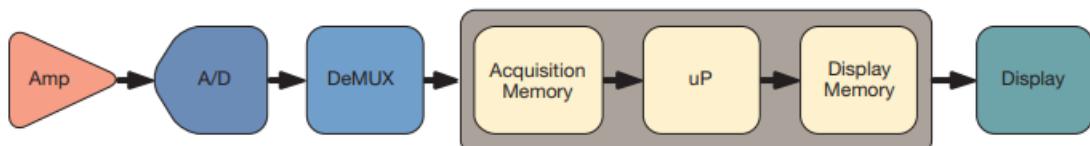
Los osciloscopios digitales adquieren la señal tomando muestras, que son almacenadas en una memoria para ser mostradas en una pantalla, como se puede ver en la figura 3.2. De este modo, es posible realizar diferentes tipos de procesados digitales de la señal y se elimina la limitación que existía en los osciloscopios analógicos para medir transitorios.



**Figura 3.2 – Osciloscopio digital (fuente: [3])**

Los osciloscopios digitales se pueden dividir en tres tipos principales:

- *Digital Storage Oscilloscope (DSO)*: este es el osciloscopio digital más extendido, consta de una etapa de acondicionamiento de la señal, que permite aumentar el rango dinámico de entrada y posteriormente se muestrea esta señal para ser almacenada, procesada y mostrada en pantalla. En la figura 3.3 se puede ver un diagrama de bloques de este tipo de osciloscopio.



**Figura 3.3 – Digital Storage Oscilloscope (fuente: [3])**

- *Digital Sampling Oscilloscope*: la señal de entrada se muestrea directamente, es decir, no se realiza ningún tipo de acondicionamiento de la señal de entrada. Esto provoca que el rango dinámico se reduzca, normalmente a 1 V pico a pico, pero el ancho de banda es mucho mayor ya que la etapa de acondicionamiento lo limitaría.
- *Digital Phosphor Oscilloscope (DPO)*: este tipo de osciloscopios cuenta con una frecuencia de muestreo muy alta y realiza un procesamiento de los datos en paralelo a la adquisición. Este procesamiento se centra mostrar los *glitches* que sufre la señal asociando un valor de intensidad a cada píxel. En la figura 3.4 se puede ver una interfaz de un osciloscopio de este tipo, en el que las zonas por las que pasa repetidamente la señal se muestran con colores cálidos, mientras que las por las que pasa con menos frecuencia, se muestran con colores fríos.

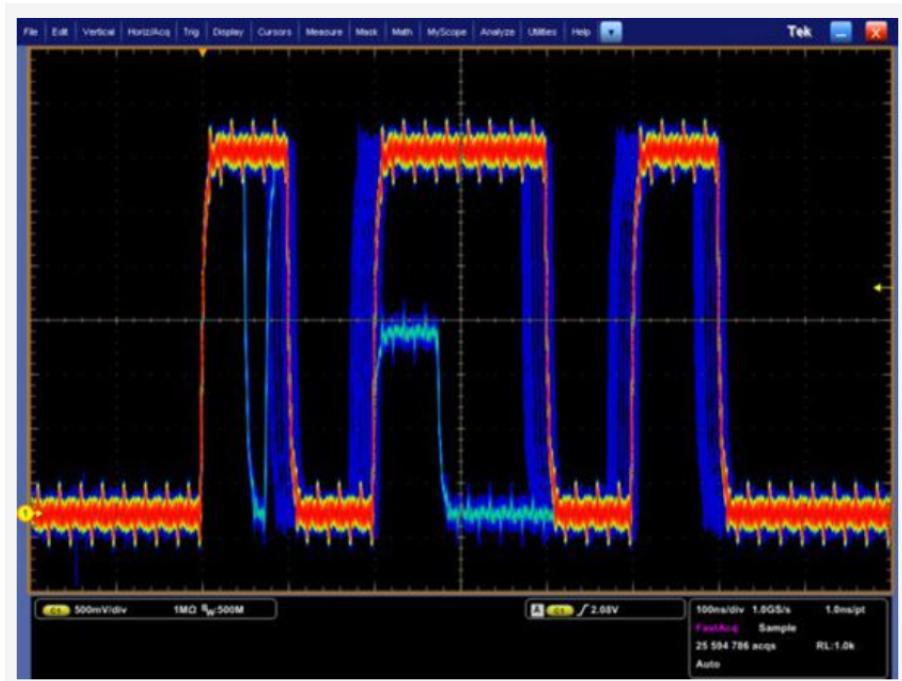


Figura 3.4 – Digital Phosphor Oscilloscope (fuente: [3])

### 3.1.3. Principales funcionalidades de los osciloscopios

- Ajuste vertical: se trata de la atenuación de la señal de entrada. Este control, establece los voltios que se corresponden a cada una de las divisiones verticales que dividen la pantalla, permitiendo así ajustar el tamaño de la señal de entrada. En los osciloscopios del tipo *Storage Oscilloscope* este mando se corresponde con la atenuación o la amplificación de la señal de entrada. Sin embargo, en los *Sampling Oscilloscope* se corresponderá únicamente con el tamaño de la señal en pantalla. Se emplean las unidades V/div.
- Ajuste horizontal: se trata de la base de tiempos. Permite establecer la cantidad de tiempo que se representa en pantalla mediante la asignación de un tiempo determinado a cada división horizontal de la pantalla. Se emplean las unidades s/div.
- *Trigger*: es el encargado de la sincronización horizontal de la señal, haciendo que las señales periódicas sean estables. Existen numerosos tipos de *trigger*, siendo el más común el *trigger* por flanco, aunque existen otros más avanzados como puede ser el *trigger* por tiempo de respuesta, por *glitch* o por ancho del pulso. El *trigger* por flanco puede activarse en los flancos ascendentes o descendentes y la señal de *trigger* será activada por un comparador entre el valor de la señal de entrada y el valor del *threshold*.
- Interpolación: se procesan las muestras para representar una señal continua. Es útil cuando la frecuencia de muestreo es entre 3 y 5 veces el ancho de banda. Existen dos tipos, cuyas representaciones se pueden ver en la figura 3.5:
  - Interpolación lineal: une las muestras con líneas rectas.

- Interpolación seno x/x: une las muestras con curvas, por lo que se consigue una representación bastante exacta.

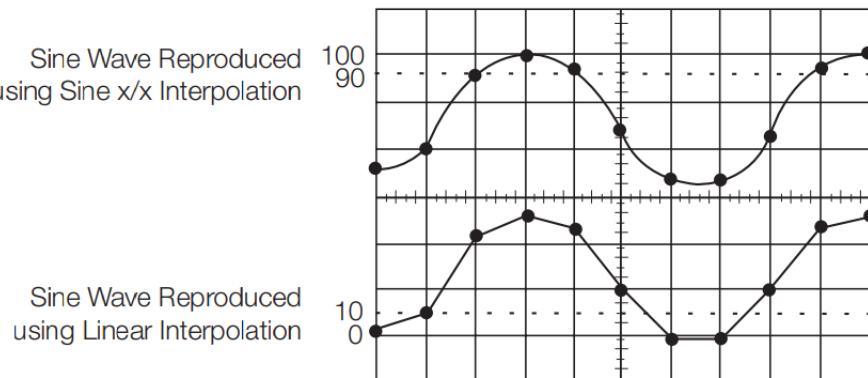


Figura 3.5 – Interpolación (fuente: [3])

### 3.2. Programación de las FPGAs

Las FPGAs se emplean para la implementación de circuitos digitales descritos mediante los denominados Lenguajes de Descripción de Hardware (HDL, por sus siglas en inglés). Para ello, se describe el funcionamiento del hardware que deseamos implementar y posteriormente, las tareas de síntesis e implementación crearán el archivo denominado *bitstream*, que configurará en la FPGA los circuitos que hemos descrito. Los principales lenguajes HDL son:

- VHDL(*Very High Speed Integrated Circuit Hardware Description Language*): es un lenguaje de descripción de hardware que se desarrolla en la década de los 80 como herramienta para la simulación de circuitos digitales complejos. Poco después se empiezan a desarrollar herramientas de síntesis e implementación a partir de los ficheros generados. Este lenguaje es estandarizado por el IEEE (*Institute of Electrical and Electronics Engineers*) y actualmente se emplea para implementar diseños en PLDs (*Programmable Logic Device*), FPGAs y ASICs (*Application-Specific Integrated Circuit*).
- Verilog: es un lenguaje de descripción hardware que soporta el diseño de circuitos analógicos, digitales y de señal mixta. Es un lenguaje que está basado en el lenguaje de programación C, para que la aceptación por parte de los ingenieros sea rápida. Inicialmente era un lenguaje privado hasta que, debido a la popularización de VHDL, se decide convertir en un lenguaje abierto y el IEEE lo estandariza. Del mismo modo que con VHDL, este lenguaje se emplea para programar PLDs, FPGAs y ASICs.

## 4 ANÁLISIS DE LAS SOLUCIONES

### 4.1. Selección de la FPGA

Para la realización de este proyecto es necesario muestrear las señales de entrada mediante un convertidor analógico-digital (ADC, por sus siglas en inglés). Esto se puede realizar mediante un ADC externo con comunicación SPI o I<sub>2</sub>C o con una FPGA con un ADC integrado. La utilización de un ADC externo puede ser interesante cuando tenemos un sistema en el que se necesita una FPGA con pocas unidades funcionales. De este modo, se reduce el coste de la FPGA al mínimo y se puede comprar un ADC de mejores características. Por otra parte, debido a que en este proyecto se pretende crear una interfaz gráfica que será visualizada en un monitor, es necesario que esta placa tenga un puerto VGA.

Contar con un ADC y un puerto VGA son requisitos mínimos a la hora de llevar a cabo este proyecto. Sin embargo, también sería interesante utilizar una placa con una buena cantidad de pulsadores y *switches*, con los que el usuario interactuaría con el osciloscopio. Actualmente, se dispone de la placa Nexys 4 DDR, como la que se puede ver en la figura 4.1. Esta es una placa de desarrollo de la FPGA Artix 7, es decir, una plataforma completa y lista para ser usada en el desarrollo de circuitos digitales basados en la Artix 7. Esta placa cuenta con puerto VGA, 5 pulsadores y 16 *switches*. Además, cuenta con 8 *displays* de 7 segmentos con los que se podría representar algún dato de interés. Por otra parte, la FPGA Artix 7 cuenta con un ADC de 12 bits y 1 MSPS. Por lo tanto, debido a que cumple todos los requisitos mínimos para la realización de este proyecto, esta será la placa sobre la que se implementará el sistema.

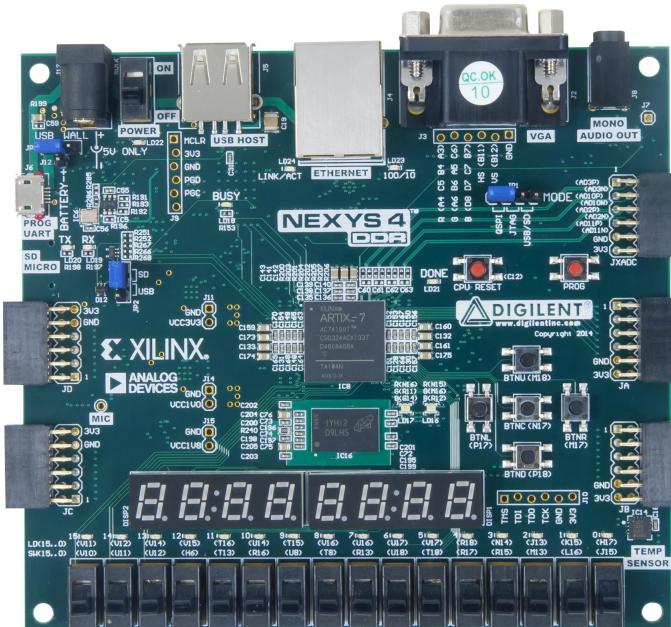


Figura 4.1 – Placa de desarrollo Nexys 4 DDR (fuente: [1])

## 4.2. Diagrama general

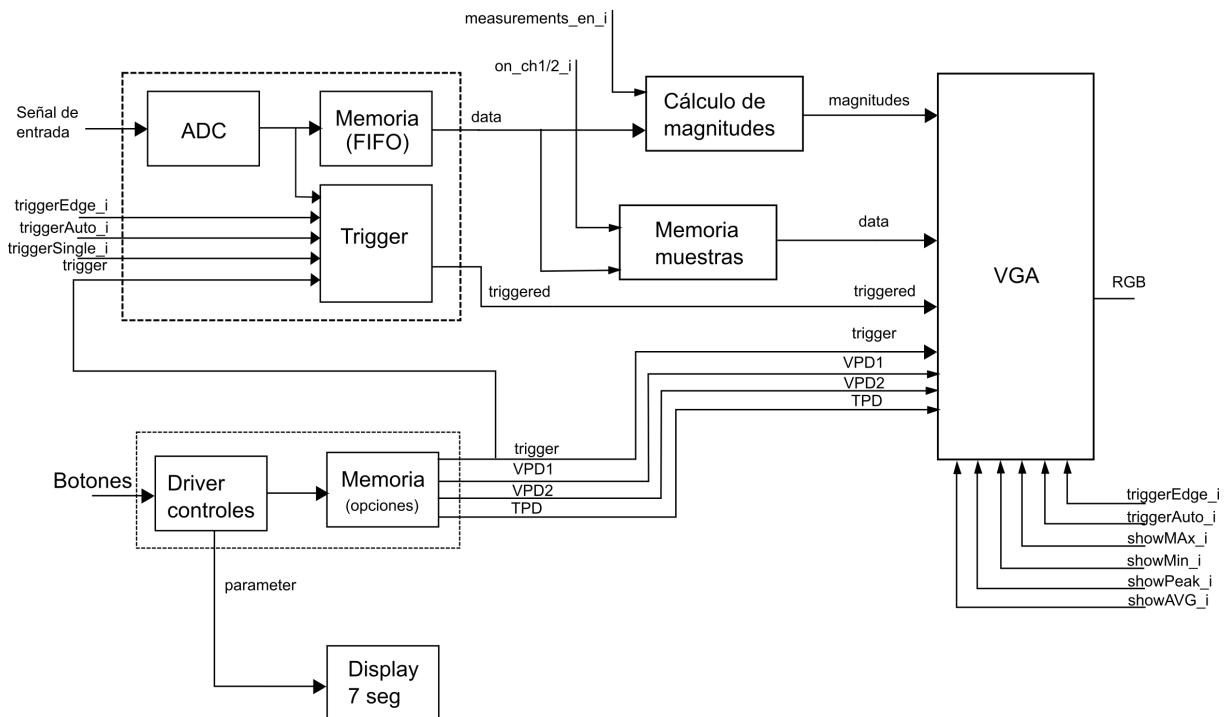


Figura 4.2 – Diagrama de bloques general

Antes de profundizar en el análisis de las soluciones, es necesario hacer una descripción general del sistema y de los diferentes bloques en los que se ha decidido dividir el proyecto. Estos bloques se realizan teniendo en cuenta las principales funciones que se deben realizar. Así, se consiguen módulos que realizan tareas concretas y uniendo todos estos módulos se conforma el sistema completo. La descripción del sistema se realizará en el lenguaje VHDL, cuyo código estará disponible en el repositorio de GitHub <https://github.com/martinpp12/ProyectoOsciloscopio>. En la figura 4.2 se puede ver un diagrama de todos los elementos que conforman el sistema y sus entradas y salidas. Estos módulos son:

- Adquisición de datos: es el módulo encargado de convertir las señales analógicas en valores digitales, dispuestos para ser leídos por los siguientes elementos. Además, es el encargado de realizar la función del *trigger* para la sincronización horizontal. Sus principales entradas y salidas son:
  - Entradas:
    - Señal de entrada: señal analógica que debe ser muestreada.
    - *Trigger*: valor digital del *threshold* para el mecanismo del *trigger*.
    - *triggerAuto\_i*: señal de habilitación del modo automático del *trigger*.
    - *triggerSingle\_i*: señal de habilitación del modo *single* del *trigger*.
  - Salidas:
    - *measurements\_en\_i*
    - *on\_ch1/2\_i*
    - *triggered*
    - *data*

- Data: Valor digital de la señal de entrada convertida.
  - *Triggered*: señal que indica que se ha producido un disparo.
- Controlador botones: es el módulo encargado de leer los 5 pulsadores presentes en la placa y modificar los registros donde se almacenan los ajustes del osciloscopio. Sus principales entradas y salidas son:
- Entradas:
    - Botones: son los 5 pulsadores presentes en la Nexys 4 DDR.
  - Salidas:
    - VPD1: Valor de los voltios por división del canal 1.
    - VPD2: Valor de los voltios por división del canal 2.
    - TPD: valor del tiempo por división.
    - *Trigger*: valor digital del *trigger*.
    - *Parameter*: valor del parámetro que se está modificando y que debe ser representado en los *displays* de 7 segmentos
- Memoria muestras: es la memoria encargada de almacenar todas las muestras de un canal. Sus principales entradas y salidas son:
- Entrada:
    - on\_ch1/ch2\_i: señales de habilitación de cada uno de los canales.
    - *Address*: es la dirección de memoria que se quiere leer/escribir.
    - *Ram\_read*: señal de habilitación de lectura.
  - Entrada/Salida:
    - *Data*: valor de la muestra que se quiere leer/escribir
- Cálculo de magnitudes: es el módulo encargado de calcular la tensión máxima, mínima, pico a pico y media de cada uno de los canales. Su entrada y salida son:
- Entrada:
    - measurements\_en\_i: señal de habilitación del cálculo de las magnitudes de las señales.
    - Data: valor de la muestra del canal correspondiente.
  - Salida:
    - Magnitudes: valor de las magnitudes que se han calculado.
- VGA: es el módulo encargado de realizar la interfaz gráfica del osciloscopio en el monitor. Sus entradas y salidas son:
- Entradas:

- Magnitudes: valores de las magnitudes calculadas por el módulo “Cálculo de magnitudes”.
  - Data: valor de las muestras del canal correspondiente.
  - *Triggered*: señal que indica que se ha producido un disparo.
  - VPD1: valor de los voltios por división del canal 1.
  - VPD2: valor de los voltios por división del canal 2.
  - TPD: valor del tiempo por división.
  - *Trigger*: valor digital del *trigger*.
  - triggerEdge.i: cuando esta señal esté a nivel alto, aparecerá en pantalla el símbolo del flanco ascendente, en caso contrario aparecerá el de flanco descendente.
  - triggerAuto.i: señal de habilitación de la representación del indicador de modo automático.
  - showMax.i: señal de habilitación de la representación la tensión máxima de los canales 1 y 2.
  - showMin.i: señal de habilitación de la representación la tensión mínima de los canales 1 y 2.
  - showPeak.i: señal de habilitación de la representación la tensión pico a pico de los canales 1 y 2.
  - showAVG.i: señal de habilitación de la representación la tensión media de los canales 1 y 2.
- Salidas:
  - RGB: valor de los tres componentes que conforman la señal de color del VGA.

### 4.3. Utilización de los switches incorporados en la Nexys 4 DDR

A la hora utilizar un osciloscopio no es suficiente con 5 pulsadores. A lo largo del proyecto se utilizan ciertas señales de entrada utilizadas como señal de habilitación de ciertas funcionalidades del osciloscopio. Estas señales están asociadas a los *switches* presentes en la placa Nexys 4 DDR, tal y como se ve en la figura 4.3. Las funciones de estas señales de entrada son las siguientes:

- reset.i: reinicia por completo el sistema, poniendo todas las señales y registros a 0. Activo a nivel bajo.
- on\_ch1.i: señal de habilitación del canal 1. La señal de este canal únicamente se muestra en pantalla cuando esta entrada está a nivel alto.
- on\_ch2.i: señal de habilitación del canal 2. La señal de este canal únicamente se muestra en pantalla cuando esta entrada está a nivel alto.

- triggerEdge\_i: cuando esta señal se encuentra a nivel alto el *trigger* es de flanco ascendente. En caso contrario, será de flanco descendente.
- triggerAuto\_i: cuando esta entrada está a nivel alto se activa el modo automático del *trigger*. Si este modo está activo, el *trigger* se dispara cuando pasa un determinado tiempo desde el último disparo.
- triggerSingle\_i: cuando esta señal está a 1, el *trigger* solo se puede disparar una vez. Para que pueda volver a dispararse es necesario desactivar este modo.
- showPeak\_i: cuando está a nivel alto se muestra la tensión de pico a pico de los canales 1 y 2.
- showMax\_i: cuando está a nivel alto se muestra la tensión máxima de los canales 1 y 2.
- showMin\_i: cuando está a nivel alto se muestra la tensión mínima de los canales 1 y 2.
- showAVG\_i: cuando está a nivel alto se muestra la tensión media de los canales 1 y 2.

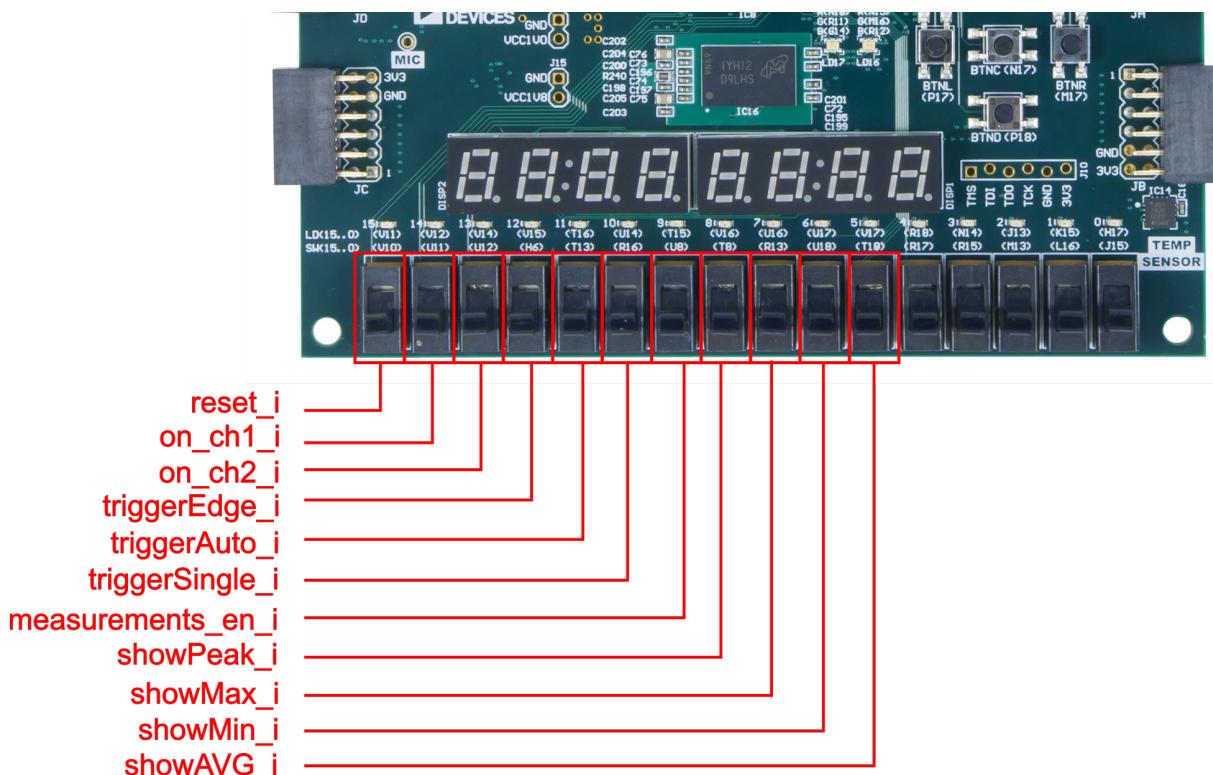


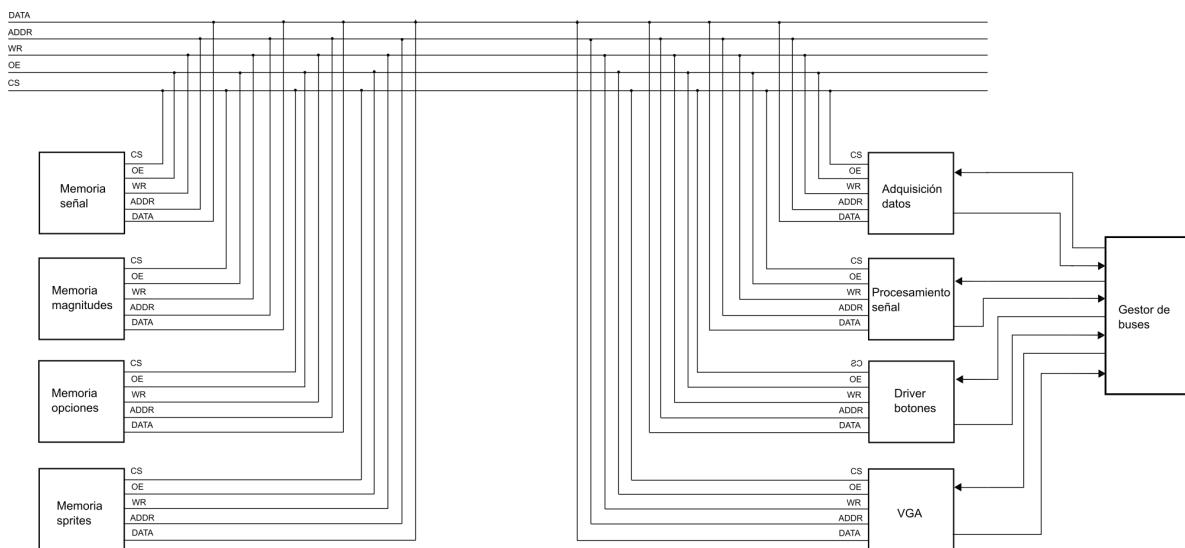
Figura 4.3 – Asociación de las señales de entrada a los *switches* presentes en la placa

## 4.4. Diseño de la arquitectura

A la hora de diseñar la arquitectura se puede recurrir a dos metodologías: la arquitectura basada en buses y la arquitectura basada en elementos de interconexión.

#### 4.4.1. Arquitectura basada en buses

Esta arquitectura consiste en conectar todos los módulos por medio de buses de datos controlados por un gestor de buses, encargado de conceder el bus al módulo que le corresponda, como podemos ver en la figura 4.4



**Figura 4.4 – Arquitectura Buses**

El principal beneficio de esta arquitectura es la escalabilidad. Cualquier módulo que se desee añadir al sistema irá conectado directamente al bus y únicamente habrá que reconfigurar el gestor de buses. Sin embargo, el número de módulos que necesitan comunicarse a través de un mismo bus puede suponer que se forme un cuello de botella en el mismo, aumentando considerablemente el tiempo necesario para transferir los datos.

#### 4.4.2. Arquitectura basada en elementos de interconexión

Esta arquitectura consiste en que cada módulo se conecta directamente con aquellos que sea necesario. Si un módulo necesita comunicarse con más de un elemento, se seleccionará mediante multiplexores el camino que recorrerán los datos. Esta arquitectura se encuentra representada en la figura 4.2

El principal beneficio de esta arquitectura es la velocidad de transferencia de los datos al no formarse el cuello de botella en el bus de comunicaciones. Sin embargo, el área necesaria para esta arquitectura será superior.

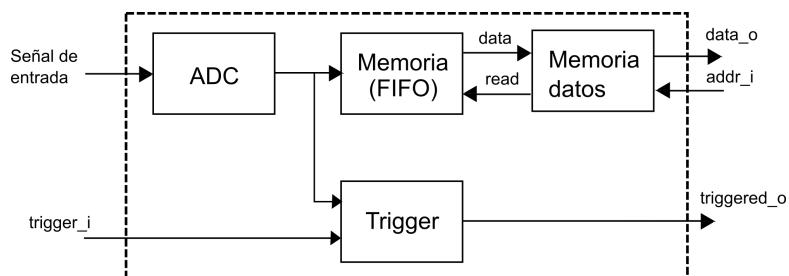
Como el incremento de área para un diseño de esta complejidad no supone un problema, se seleccionará esta arquitectura para evitar problemas con las latencias en la transferencia de los datos.

## 4.5. Implementación genérica

Con el fin de conseguir una implementación lo más genérica posible se crea un paquete de constantes, denominado “oscilloscope\_pkg” en el que se establecen todos los parámetros de configuración, como pueden ser la resolución del ADC o la resolución del monitor que se va a usar y sus especificaciones temporales. De este modo, se hace mucho más sencilla la implementación del sistema en FPGAs con otras características como puede ser el número de bits del ADC, que es un parámetro del que depende todo el proyecto y cuyo reajuste supondría un gran esfuerzo si no se realizara de esta manera.

## 4.6. Diseño de la adquisición de datos

El bloque de la adquisición, representado en la figura 4.5, está conformado por 3 elementos principales: el ADC integrado en la FPGA, el módulo del *trigger* que pondrá a nivel alto su salida cuando se produzca un flanco y una memoria FIFO (*First In, First Out*) para cada canal, que funcionarán como *buffer* de las memorias de almacenamiento del canal que corresponda, almacenando muestras mientras que estas memorias están siendo leídas por otro módulo.



**Figura 4.5 – Diagrama simplificado de la adquisición de datos**

### 4.6.1. Convertidor analógico-digital

En el apartado 3.1.2 se han analizado los tipos de osciloscopios digitales que existen, entre los que se encontraba el *Digital Sampling Oscilloscope*. Debido a que este proyecto se centra en una implementación de un osciloscopio sobre una placa de desarrollo de una FPGA, este es el tipo de osciloscopio que se considera hacer, al no disponer de etapa de acondicionamiento. Por lo tanto, la señal de entrada accede directamente al ADC.

#### 4.6.1.1. Canales del ADC

La Artix 7 cuenta con un ADC de 12 bits, 1MSPS y 16 canales. En la placa Nexys 4 DDR las entradas de los canales van rutadas directamente al puerto PMOD JXADC. Los canales rutados son el 3, 10, 2 y 11; que están conectados en los pines 1, 2, 3 y 4 respectivamente (figura 4.6).

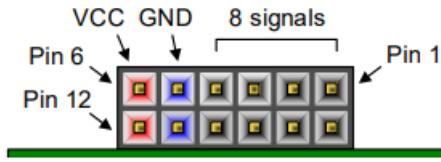


Figura 4.6 – Puerto PMOD (fuente: [1])

Estos canales se reparten en dos ADCs por medio de multiplexores, llevando las señales de los canales 0 al 7 al multiplexor A y las señales de los canales 8 al 15 al multiplexor B, como se puede ver en la figura 4.7

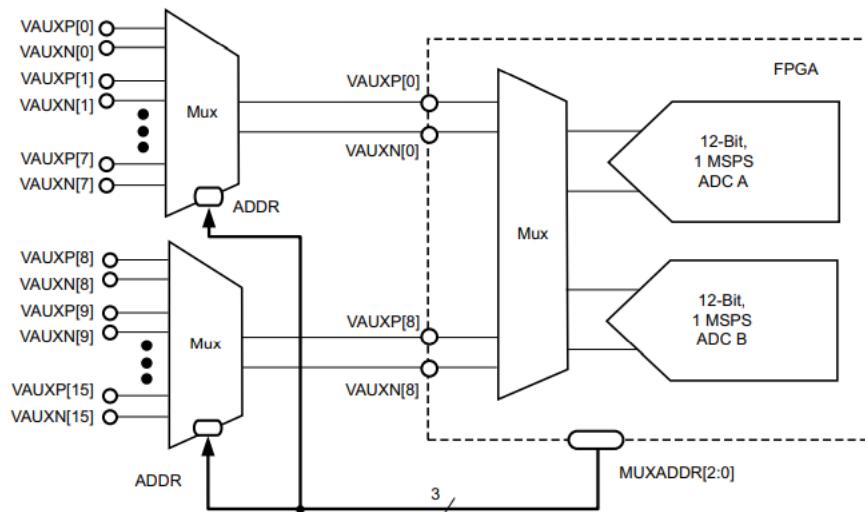


Figura 4.7 – Diagrama del ADC (fuente: [2])

Como se ha especificado en el apartado 1.3, el osciloscopio debe tener 2 canales. Con el fin de que estos canales puedan tener la máxima frecuencia de muestreo, lo más recomendable es seleccionar canales que estén conectados a ADCs diferentes. Es decir, que el canal 1 este conectado al convertidor A y el canal 2 al convertidor B. Por lo tanto, se decide emplear los canales 2 y 10, ya que son canales que están rutados al puerto de la placa y cumplen estas características.

#### 4.6.1.2. Modos de funcionamiento

En el apartado anterior se ha seleccionado los canales a utilizar con el fin de que se pueda tener la máxima frecuencia de muestreo. Sin embargo, es necesario configurar el modo de captura del ADC. Este ADC dispone de los siguientes modos de captura:

- *Single Channel Mode*: se muestrea un solo canal. Se puede cambiar el canal que se está muestreando con los registros de control, por lo que si hay que muestrear muchos canales, puede haber una sobrecarga en el control.
- *Automatic Channel Sequencer*: el secuenciador del ADC selecciona automáticamente el próximo canal que va a ser muestreado y ajusta todos los parámetros necesarios.

- *Channel Sequencer*: en este modo se muestran varios canales de uno en uno secuencialmente.
- *Simultaneous Selection*: utiliza los dos ADCs disponibles (A y B) para muestrear dos señales simultáneamente. El XADC va seleccionando automáticamente que pares va muestreando. Es útil cuando se quiere preservar la relación de fase entre dos señales.
- *Independent ADC*: en este caso, el ADC A se emplea para medir parámetros de la FPGA (temperatura, tensión de alimentación, etc.) para la generación de alarmas. Por otra parte, el ADC B se emplea para medir señales analógicas externas.

Como en el osciloscopio se desea poder ver simultáneamente los dos canales, es necesario que se mantenga la relación de fase entre las señales. Por lo tanto, es necesario emplear el modo de selección simultánea. Sin embargo, este modo de funcionamiento requiere un control adicional para la lectura de los datos.

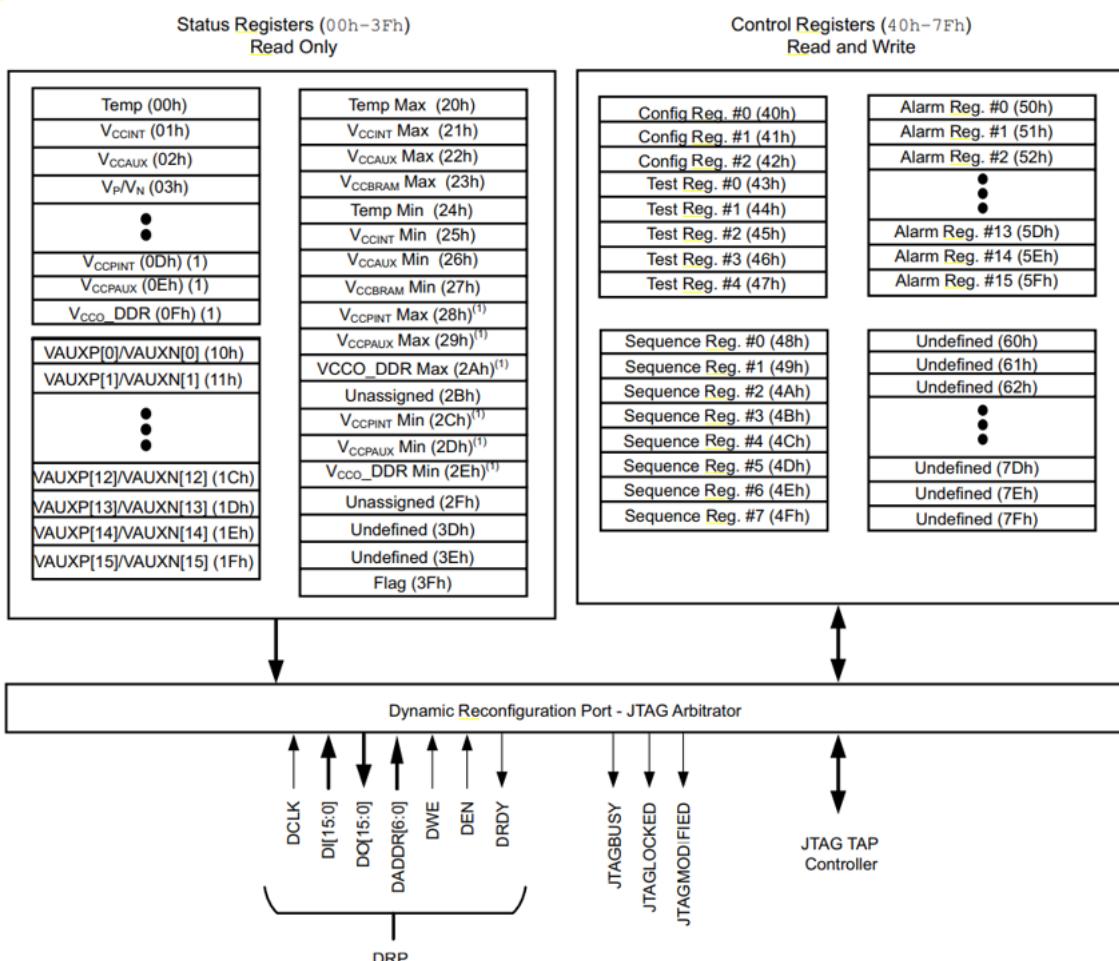
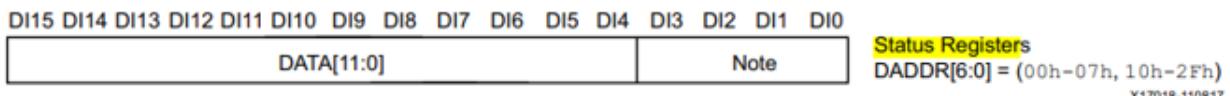


Figura 4.8 – Registros de estado del ADC (fuente: [2])

El ADC cuenta con un registro interno en el que se almacena la última medición de cada uno de los canales. Este registro se puede ver en la figura 4.8 y en ella se puede ver que las direcciones de los canales, que se corresponden con los registros del “VAUXP[0]VAUXN[0]”

al “VAUXP[15]VAUXN[15]”, van desde la  $10_{16}$  hasta la  $1F_{16}$ . Como en el apartado 4.6.1.1 se decide utilizar los canales 2 y 10, las direcciones que se deberán utilizar son la  $12_{16}$  y la  $1A_{16}$ , respectivamente. La lectura del dato se realiza mediante los pines DRP que se pueden ver en la figura 4.8 del siguiente modo:

1. Se introduce la dirección  $12_{16}$  (canal 2) en el puerto “daddr\_in”.
2. Se pone “den” a nivel alto, indicando que queremos leer la dirección del puerto “daddr\_in”.
3. Se espera hasta que la salida “drdy” este a nivel alto, indicando que ya se puede leer el dato.
4. Se lee el dato del puerto “do\_out”.
5. Se repite el proceso con la dirección  $1A_{16}$  (canal 10).



**Figura 4.9 – Formato del dato del ADC (fuente: [2])**

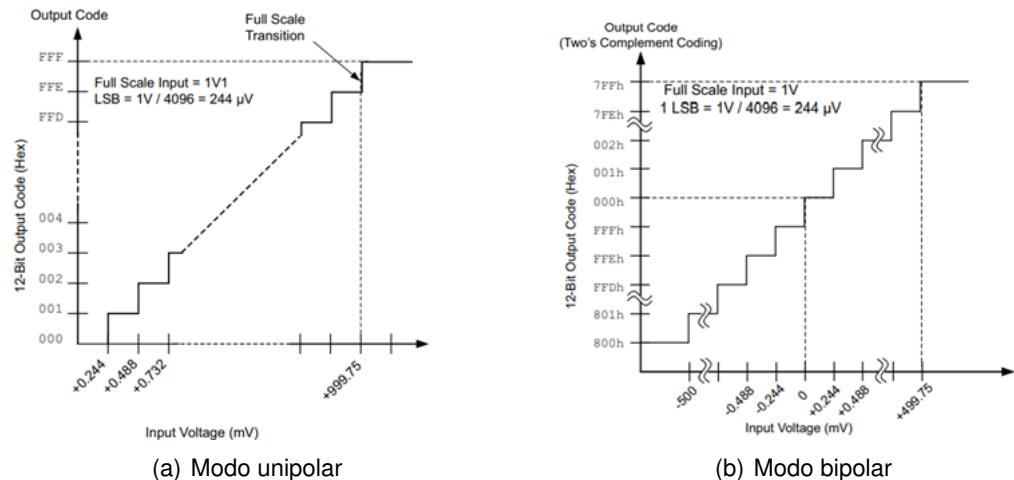
En la figura 4.9 se puede ver el formato del dato leído del ADC. El resultado de la conversión siempre produce 16 bits, de los que habrá que desechar los 4 bits menos significativos. El formato de este resultado depende de la configuración de entrada del ADC.

#### 4.6.1.3. Configuración del tipo de entrada

Existen dos modos de interpretación de la señal de entrada:

- Entrada unipolar: en este modo de funcionamiento se miden señales positivas entre 0 y 1 V. Debido a que el ADC es de 12 bits, el resultado de la conversión tomará valores entre 0 para 0 V y 4095 para 1 V o lo que es lo mismo, entre  $000_{16}$  y  $FFF_{16}$ , tal y como aparece representado en la figura 4.10(a).
- Entrada bipolar: en este modo la conversión resultante está en complemento a 2. Se sigue manteniendo el rango de 0 a 1 V en la entrada, pero se establece el 0 del ADC en 0.5 V, como se puede ver en la imagen 4.10(b). Por lo tanto, será necesario introducirle un offset de 0.5 V a las señales de entrada.

La selección de cualquiera de estos modos no supone ninguna modificación de la señal de entrada, sino en el resultado de la conversión. Sin embargo, como para la realización de este proyecto es interesante la medición de señales positivas y negativas, disponer de una conversión en Ca2 con la que poder realizar operaciones con signo aumenta la simplicidad del tratamiento de los datos. Si se utilizase la entrada unipolar habría convertir los valores digitales, resultantes de la conversión, de un valor sin signo a un valor con signo para operar con ellos de manera eficiente.

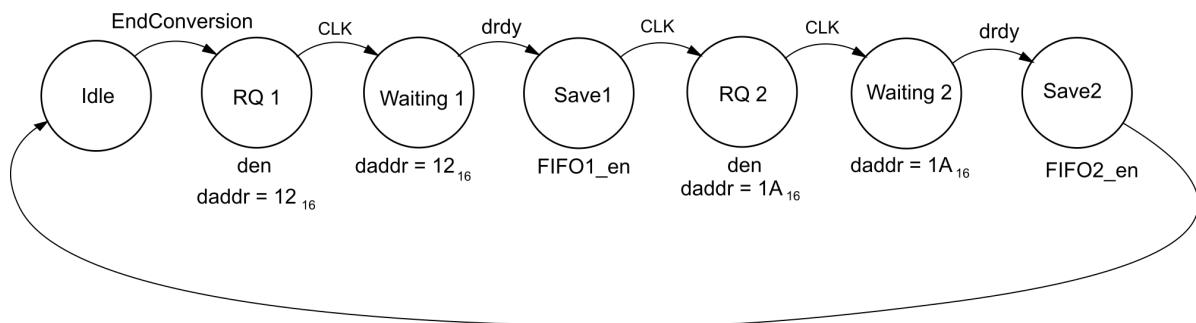


**Figura 4.10 – Funciones de transferencia del ADC (fuente: [2])**

#### 4.6.1.4. Control del ADC

Finalmente, se diseña la máquina de estados de la figura 4.11, que se encarga de la lectura de ambos canales del ADC:

En primer lugar, se permanece en un estado de *Idle* en el que se espera a que se realice una conversión. Cuando se ha terminado la conversión se pone la dirección  $12_{16}$ , correspondiente al canal 2, en el puerto “daddr” y se pone a nivel alto la señal “den”, que le indica al ADC que debe devolver el valor del registro cuya dirección se ha introducido. A continuación, es necesario esperar a que la señal “drdy” se ponga a nivel alto, indicando que el dato está disponible para ser leído. En ese momento, se habilita la FIFO del canal 1 para que almacene el dato. Una vez guardado el dato en la FIFO podemos pasar a realizar el mismo proceso para leer el dato del canal 10, introduciendo la dirección  $1A_{16}$ , y guardarla en la FIFO de su canal. Una vez hecho todo el proceso, se vuelve al estado de *Idle* hasta que se realice una nueva conversión.



**Figura 4.11 – Máquina de estados para el control del ADC**

#### 4.6.1.5. Simulación de la lectura del ADC

Una vez diseñado el módulo de control del ADC es necesario comprobar su correcto funcionamiento mediante una simulación. Con esto se pretende comprobar, tanto que el ADC

realiza la conversión correctamente, como que la máquina de estados encargada del control interactúa correctamente con el ADC, de manera que se lean los datos correctamente. Para realizar la simulación del ADC es necesario crear un fichero, denominado “design”, en la carpeta de simulación del proyecto, en el que se introducirán los estímulos. Estos estímulos se deben corresponder con los valores de la tensión de entrada para los diferentes instantes de tiempo. En la figura 4.12 se puede ver un ejemplo del formato de este fichero.

TIME	TEMP	VCCINT	VCCBRAM	VCCAUX	VP	VN	VAUXP[2]	VAUXN[2]	VAUXP[10]	VAUXN[10]
0	63.0	1.02	1.02	1.8	0.5	0.0	0.50	0.5	0.50	0.5
1000	63.0	1.02	1.02	1.8	0.5	0.0	0.51	0.5	0.52	0.5
2000	63.0	1.02	1.02	1.8	0.5	0.0	0.52	0.5	0.53	0.5
3000	63.0	1.02	1.02	1.8	0.5	0.0	0.53	0.5	0.55	0.5
4000	63.0	1.02	1.02	1.8	0.5	0.0	0.54	0.5	0.56	0.5
5000	63.0	1.02	1.02	1.8	0.5	0.0	0.55	0.5	0.58	0.5

Figura 4.12 – Estímulos para la simulación del ADC

Una vez realizado este fichero se puede realizar la simulación. En la figura 4.13 se puede apreciar como, empleando la máquina de estados descrita en el apartado anterior, el ADC devuelve ambos canales.

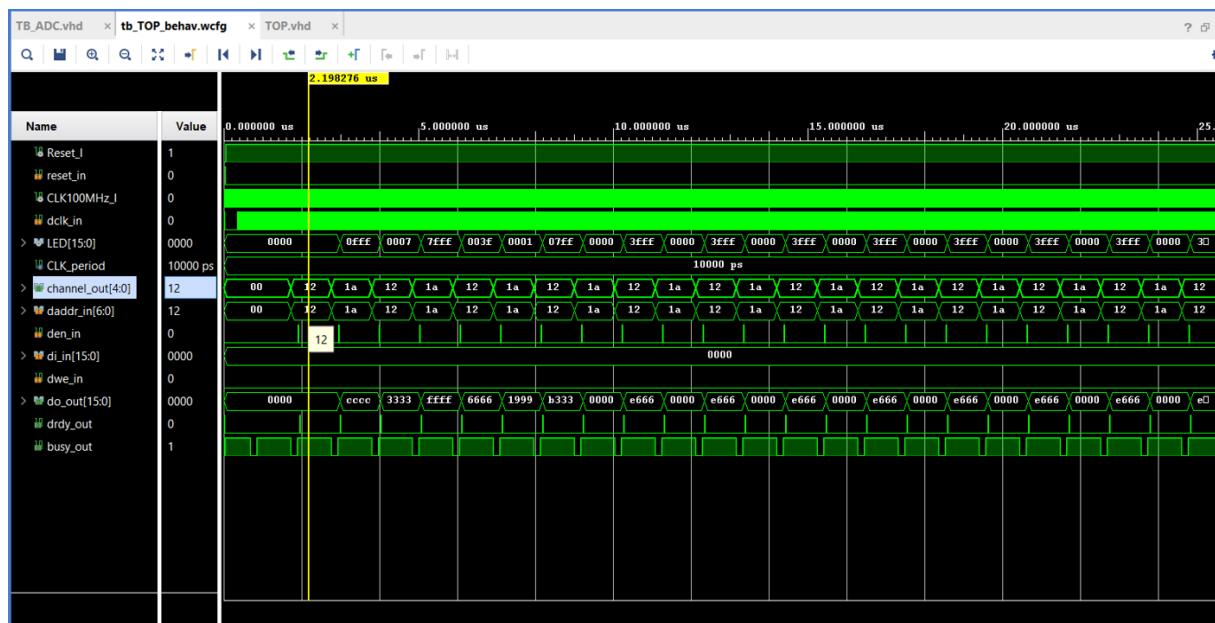


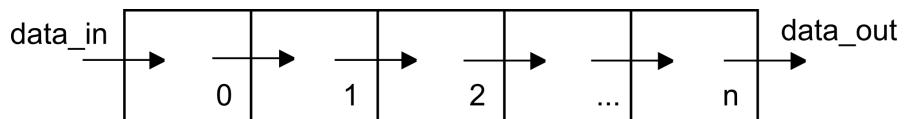
Figura 4.13 – Simulación de la lectura del ADC

#### 4.6.2. Memoria FIFO

Debido a que las memorias de las señales están siendo leídas por varios elementos, es necesario incorporar una memoria que almacene estas muestras hasta que puedan ser leídas por la memoria principal. El funcionamiento de esta memoria debe ser del tipo FIFO, ya que es necesario que cuando se solicite un dato se devuelva el dato más antiguo.

Las memorias FIFO son un tipo de memorias en las que el dato introducido ocupa la posición vacía con el número más alto posible, según el diagrama de la figura 4.14. Cuando se le solicite un dato, se pondrá en la salida el que ocupa la posición n, a la vez que todos los elementos de la FIFO se desplazan una posición a la derecha, ocupando la posición que acaba

de quedar vacía. Además, las memorias FIFO incorporan pines de memoria vacía y/o llena, útiles a la hora de leer o escribir en este tipo de memorias.



**Figura 4.14** – Funcionamiento de una memoria FIFO

### 4.6.3. Trigger

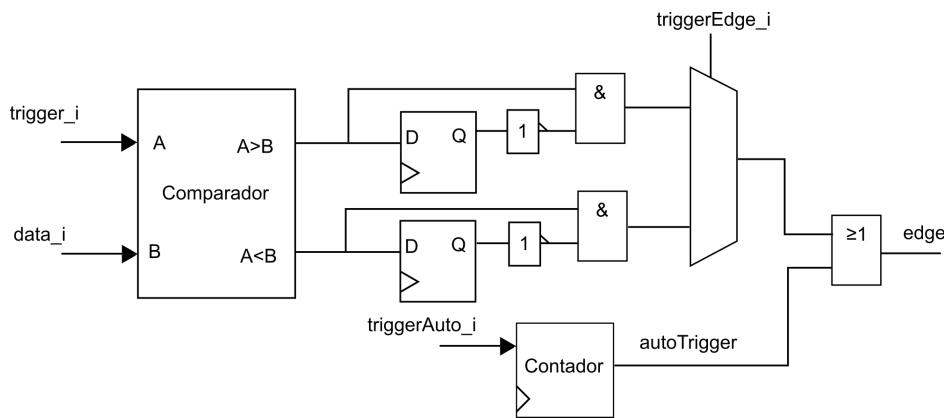
Como se ha comentado en el apartado 3.1.3 existen diferentes tipos de *triggers*, siendo el más común el *trigger* por flanco.

En primer lugar, es necesario seleccionar el canal que se va a tener en cuenta a la hora de activar la señal de *trigger*. En algunos osciloscopios se permite seleccionar cualquiera de los canales de los que se dispone. Sin embargo, en este caso y por simplicidad se establecerá el canal 1 para esta función.

#### 4.6.3.1. Mecanismo de disparo

Este tipo de trigger está formado, principalmente por un comparador entre el valor de *threshold* y el valor actual de la señal de entrada. Este valor de *threshold* será introducido por el usuario, como se recoge en el apartado 4.7.2 y la comparación se realizará en función de una entrada que indique el tipo de flanco deseado. En el momento en el que la señal de entrada sea superior o inferior al *threshold*, en función del tipo de flanco deseado, la señal de *trigger* debe activarse. Para saber cuando ocurre esto, se incorpora un detector de flanco en el resultado de la comparación. Además, debido a que la señal en pantalla se actualizará únicamente cuando se active el *trigger*, es necesario incorporar un mecanismo de *trigger* automático que refresque la señal cuando pase un tiempo determinado desde el último *trigger*. El ajuste del comportamiento del *trigger* mediante las señales “*triggerEdge\_i*” y “*triggerAuto\_i*” se consigue asociando estas entradas a los *switches* presentes en la placa.

Con todo esto, se obtiene el diagrama que se puede ver en la figura 4.15, en la que la activación del *trigger* se indica mediante la señal “*edge*”.



**Figura 4.15 – Diagrama detector de flanco en la señal de entrada**

#### 4.6.3.2. Dirección en la que se produce el disparo

Hasta este momento, se ha diseñado la generación de la señal que indica que se ha superado el *threshold*. Este instante de tiempo se corresponde con una dirección de las memorias de las señales, en la que se almacena la muestra correspondiente.

Con el fin de poder ver que es lo que sucede antes y después del momento de activación, es interesante que este instante se encuentre en el centro de la pantalla. El ancho del *plot* se define en la constante "*PLOT\_WIDTH*", contenida en el paquete de constantes comentado en el apartado 4.5. Por lo tanto, al módulo encargado de mostrar la señal en pantalla (véase el apartado 4.9.2.2) se le debe enviar la dirección de memoria asociada al instante de activación menos *PLOT\_WIDTH* / 2. De este modo, se representarán las muestras desde el disparo menos *PLOT\_WIDTH* / 2 hasta el disparo más *PLOT\_WIDTH* / 2.

#### 4.6.3.3. Máquina de estados

Finalmente, con el fin de controlar este mecanismo, se desarrolla la máquina de estados de la figura 4.16 en la que se incorpora la posibilidad de emplear un modo de funcionamiento "*single*", en el que una vez se ha disparado el *trigger* es necesario desactivar este modo para que pueda volver a producirse un disparo. Además, se incorpora un tiempo de espera, durante el que se van a tomar *PLOT\_WIDTH*/2 muestras, ya que de otro modo la mitad del *plot* estaría desactualizado. Para ello, será necesario un contador que cuente el número de muestras emitidas por el ADC.

En un inicio, la máquina se encuentra en el estado *Waiting*, en el que está esperando a que se produzca un disparo, ya sea producido por la señal o por el *trigger* automático. En cuanto se produce el *trigger* se pasa al estado *Triggered* en el que se almacena la dirección en la que se ha producido el disparo. A continuación, se pasa al estado *Sampling* en el que se permanece hasta que se han tomado muestras suficientes para completar el *plot*, momento en el que se va al estado de *Finished*. En este punto, si el modo *single* está habilitado se pasa al estado *singleShot* en el que se permanece hasta que se desactive este modo. En caso contrario, se vuelve directamente al estado *Waiting*.

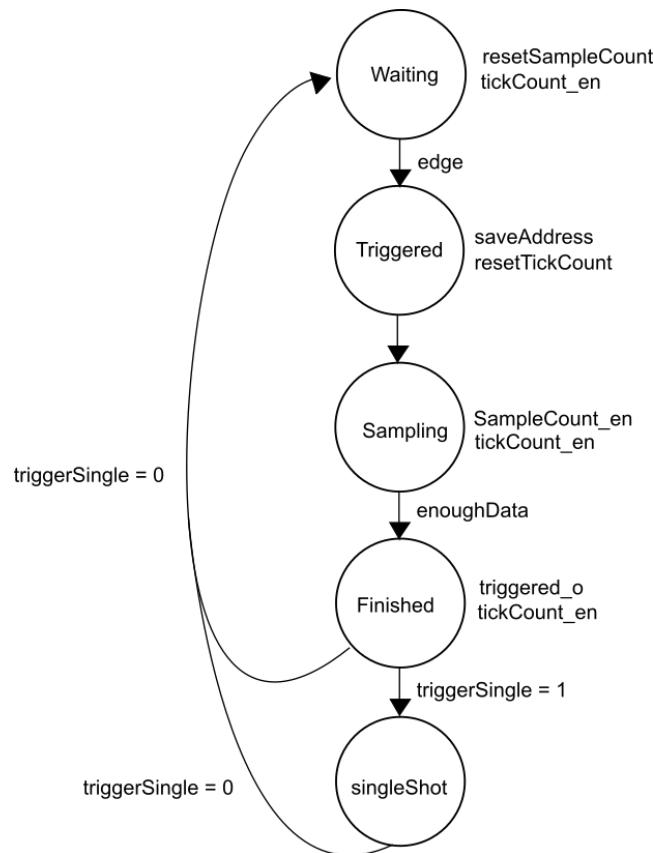


Figura 4.16 – Máquina de estados trigger

#### 4.6.3.4. Simulación

Como se puede ver en la figura 4.17, cuando la señal de entrada supera el valor del *trigger* se produce un pulso en la señal *edge*. En ese momento, se comienza a tomar muestras hasta llenar el plot. Una vez se ha llenado el plot, se pone a nivel alto la señal *enoughData*, haciendo que se registre el trigger en la señal “*reg\_triggered*” y la dirección de memoria en “*reg\_address*”.

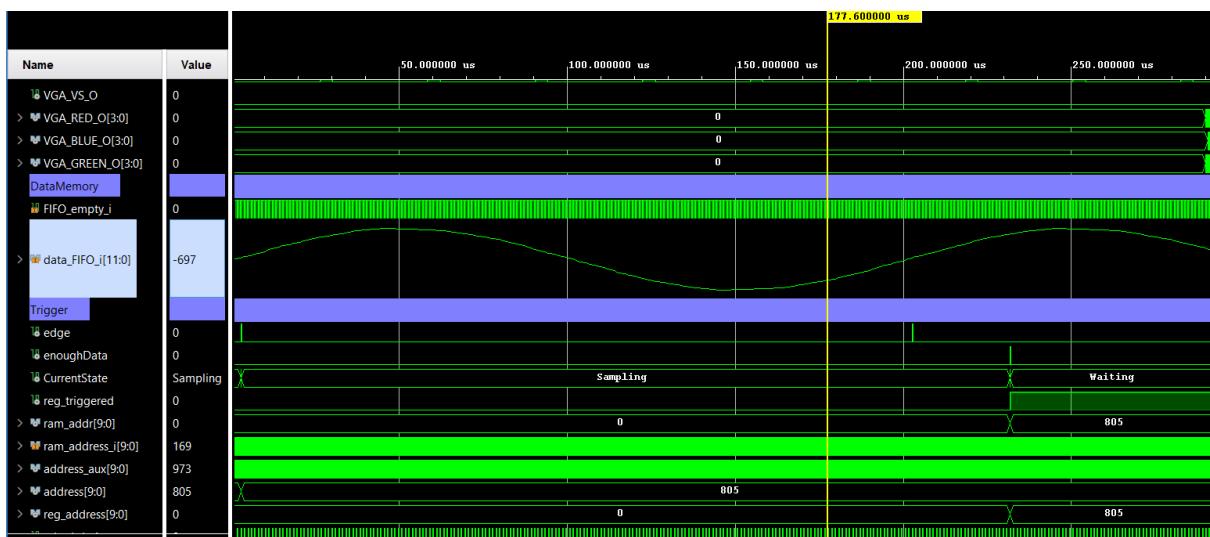


Figura 4.17 – Simulación del disparo del trigger

#### 4.6.4. Memorias de los canales

Este módulo consiste en una memoria RAM (*Random Access Memory*) que contiene las memorias en las que se almacenan las muestras de los canales y sus elementos de control. Cada una de los canales debe disponer de una memoria independiente, ya que es la manera más eficiente de poder encender los canales por separado.

##### 4.6.4.1. Tamaño de la memoria

A la hora de diseñar la memoria es necesario saber el número de muestras que se van a mostrar en pantalla, ya que el tamaño de la memoria deberá ser necesariamente superior a este. Como se ha comentado anteriormente, en el paquete de constates se define *PLOT\_WIDTH*, que establece el ancho del *plot*. El valor de esta constante es de 440 píxeles. El número de bits necesarios para direccionar 440 posiciones es 9, que se corresponde con 512 direcciones. Sin embargo, como el ancho de la pantalla es de 640 píxeles, se debería dar la posibilidad de poner un *plot* de ese ancho. Para ello, sería necesario disponer de una memoria de, como mínimo, 640 posiciones. Esto significa que habría que añadir un bit más al bus de direcciones, teniendo así 10 bits. Por lo tanto, el tamaño de la memoria será finalmente de  $2^{10} = 1024$  direcciones.

##### 4.6.4.2. Control del número de muestras

Todos los osciloscopios, sean analógicos o digitales disponen de un ajuste horizontal con el que se puede ampliar o alejar la señal con el fin de ver más o menos ciclos. Cuando se desee ver menos ciclos de la señal habrá que aumentar la frecuencia de muestreo, de manera que se alcancen las *PLOT\_WIDTH* muestras necesarias para llenar el *plot* en menos tiempo. Sin embargo, cuando se desee ver muchos ciclos habrá que reducirla, de manera que se tarde más tiempo en llenar el *plot*. Esto se consigue mediante los siguientes métodos:

- *Upsampling*: consiste en añadir muestras con valor cero entre dos muestras reales. Se realiza cuando se requiere una frecuencia de muestreo mayor a la máxima del ADC. Empleando algoritmos como la interpolación es posible obtener el valor que tendría una muestra en esa posición. Cuanto mayor es el *upsample*, más ceros se insertan entre muestras.



Figura 4.18 – Señal original

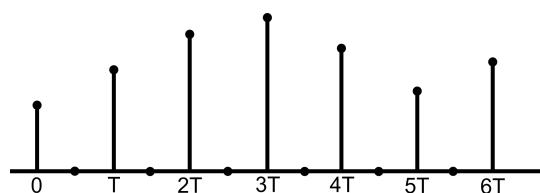
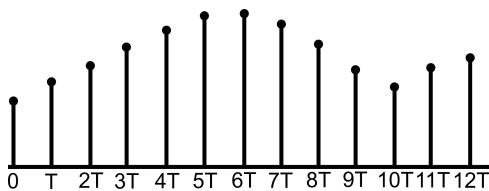


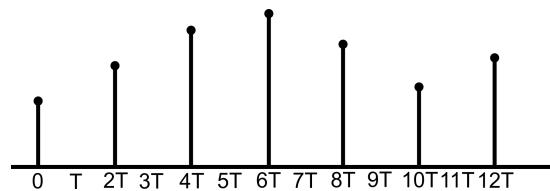
Figura 4.19 – Señal con upsample 2

- *Downsampling*: consiste en descartar un determinado número de muestras en función del número de *downsample* que se quiera realizar. De este modo, podemos reducir la

frecuencia de muestreo sin modificar la frecuencia del ADC. El número de muestras que es necesario descartar es el valor del *downsample* menos 1.



**Figura 4.20** – Señal original



**Figura 4.21** – Señal con *downsample* 2

Como se ha especificado anteriormente, el ADC tiene una frecuencia de muestreo de 1 MHz y el *plot* tiene un ancho de 440 píxeles. Teniendo en cuenta que, como se establece en el apartado 4.9.2.1, el *plot* se divide en 10 divisiones horizontales y sabiendo que:

$$f_s = \frac{PLOT\_TIME}{PLOT\_WIDTH} \quad (4.1)$$

$$downsample = \frac{f_s}{f_{ADC}} \quad (4.2)$$

Si  $f_s$  es la frecuencia de muestreo necesaria y  $f_{ADC}$  la frecuencia de muestreo del ADC. Se obtienen los siguientes valores de *downsample* para los diferentes TPD:

**Tabla 4.1** – Valores de *downsample* para diferentes escalas horizontales

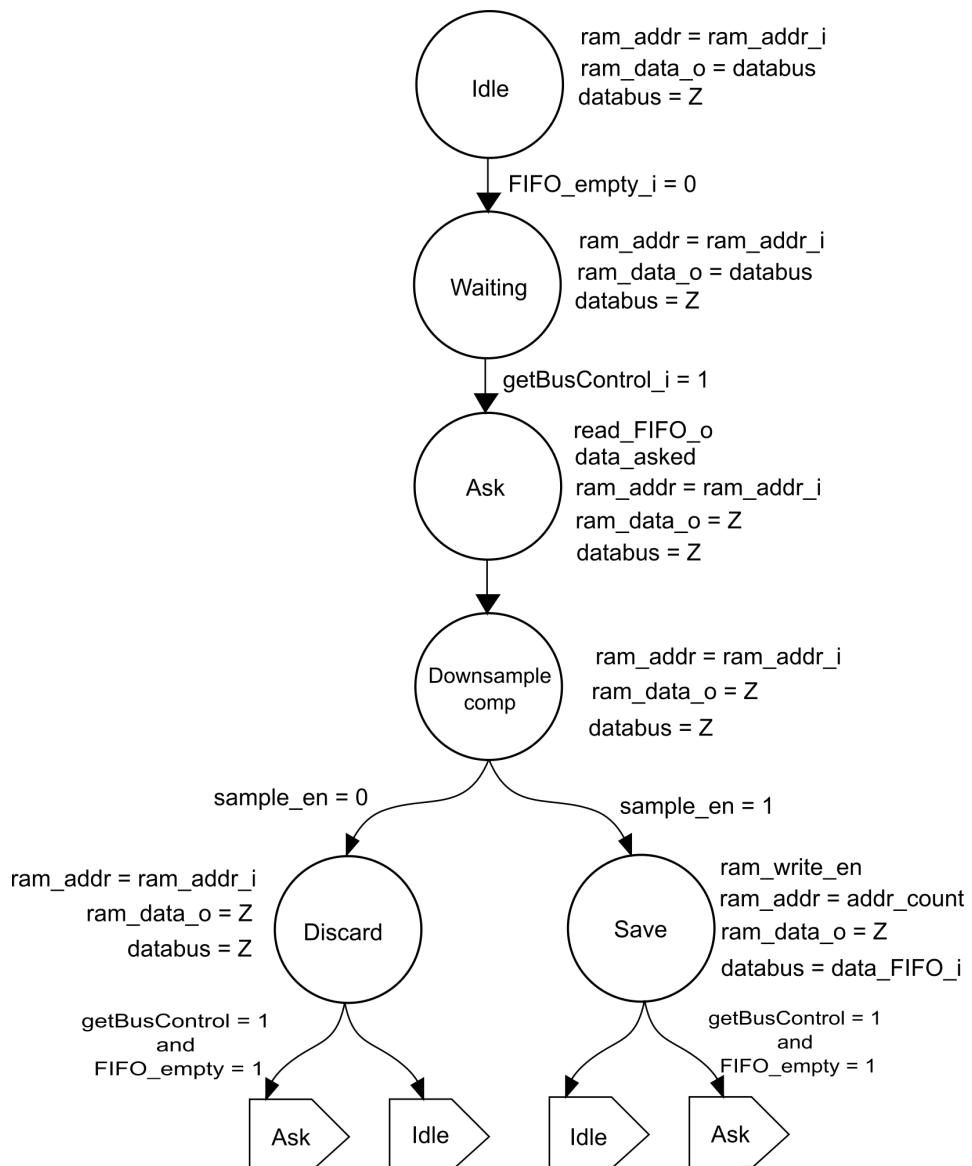
TPD	Plot time	muestras/s	downsample	downsample int
2 $\mu$ s	20 us	$2,2 * 10^7$		
5 $\mu$ s	50 us	$8,8 * 10^6$		
10 $\mu$ s	100 us	$4,4 * 10^6$		
20 $\mu$ s	200 us	$2,2 * 10^6$		
50 $\mu$ s	500 us	$8,8 * 10^5$	1.14	1
100 $\mu$ s	1 ms	$4,4 * 10^5$	2.27	2
200 $\mu$ s	2 ms	$2,2 * 10^5$	4.55	4
500 $\mu$ s	5 ms	$8,8 * 10^4$	11.36	11
1 ms	10 ms	$4,4 * 10^4$	22.73	22
2 ms	20 ms	$2,2 * 10^4$	45.45	45
5 ms	50 ms	$8,8 * 10^3$	113.64	113
10 ms	100 ms	$4,4 * 10^3$	227.27	227
20 ms	200 ms	$2,2 * 10^3$	454.55	454
50 ms	500 ms	$8,8 * 10^2$	1136.36	1136
100 ms	1 s	$4,4 * 10^2$	2272.73	2272
200 ms	2 s	$2,2 * 10^2$	4545.45	4545

En la tabla 4.1 se puede ver que para TPDs menores a 50  $\mu$ s la frecuencia de muestreo es superior a la frecuencia máxima de muestreo del ADC, momento en el que es necesario

comenzar a realizar *upsampling*. Sin embargo, debido a que la interpolación de la señal se sale del alcance del proyecto se decide no realizar el *upsampling*, ya que produciría distorsión en la señal.

#### 4.6.4.3. Control de la memoria

Para realizar el control de la memoria es necesario tener en cuenta el estado de la memoria FIFO y de los buses de datos. Como se ha comentado en el apartado 4.6.2, las memorias FIFO cuentan con pines de *empty* que indican cuando la memoria está vacía. De este modo, es posible saber cuando hay un dato disponible para ser leído. Por otra parte, los buses de datos son ocupados por el módulo del VGA. Este es el encargado de leer la señal de la memoria y mostrarla en pantalla, por lo que debe tener prioridad sobre el bus, respecto a la FIFO. Por ello, se deben comunicar ambos módulos, de manera que el VGA le indique cuando no está ocupando el bus.



**Figura 4.22 – Máquina de estados para el control de la memoria de muestras**

Sabiendo todo esto, se diseña la máquina de estados de la figura 4.22 que en un primer momento se encuentra en el estado de *Idle*, en el que la memoria está siendo leída por el módulo VGA. Cuando el *flag* de *empty* de la FIFO se pone a 0, indicando que contiene un dato, se pasa al estado de *Waiting*, en el que se espera a que el módulo del VGA ceda los buses, poniendo "getBusControl\_i" a nivel alto. En ese momento, se pasa al estado *Ask*, en el que se le solicita el dato a la memoria FIFO. A continuación se pasa al estado *Downsample comp* y se comprueba si se debe almacenar la muestra, en función del valor del *downsample*, tal y como se especifica en el apartado 4.6.4.2. En este momento, la máquina puede tomar dos caminos. Por un lado, se va al estado *Save* en el que se guarda la muestra en memoria y por otro lado se va a *Discard* en el que no se guarda. En ambos casos si se dispone de los buses y la FIFO contiene alguna muestra, se regresa al estado *Ask*. En caso contrario, se va a *Idle*.

#### 4.6.4.4. Simulación de la lectura de la FIFO

Una vez diseñada la interacción de la memoria FIFO y la memoria de las muestras, es necesario realizar la simulación de la comunicación para comprobar que las muestras se leen correctamente de la FIFO y se almacenan en la memoria de las muestras. En la figura 4.23 se puede ver como cada vez que se da un flanco de bajada se produce un pulso en la señal "readFIFO\_o" y a continuación, la memoria emite el dato. En la figura 4.24 se puede ver como se almacena correctamente el dato en la memoria RAM.



Figura 4.23 – Simulación de la lectura de la memoria FIFO - parte 1

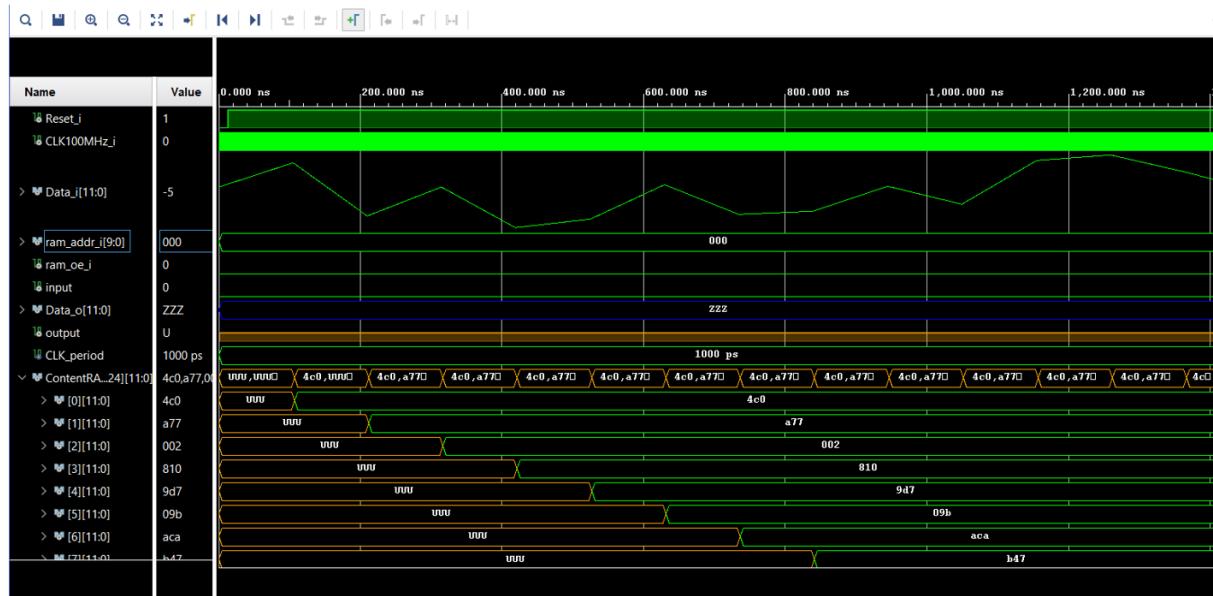


Figura 4.24 – Simulación de la lectura de la memoria FIFO - parte 2

#### 4.6.4.5. Simulación del downsampling

En las señales “ram\_write\_en” y “downsample\_i” de la simulación de la figura 4.25 se puede apreciar el correcto funcionamiento de este sistema. Como se puede ver, cuando el downsampling es 2 se desecha una muestra por cada muestra que se guarda, cuando es 3 se desechan 2 muestras por cada una que se guardan y cuando es 10 se desechan 9 muestras por cada una que se guarda.

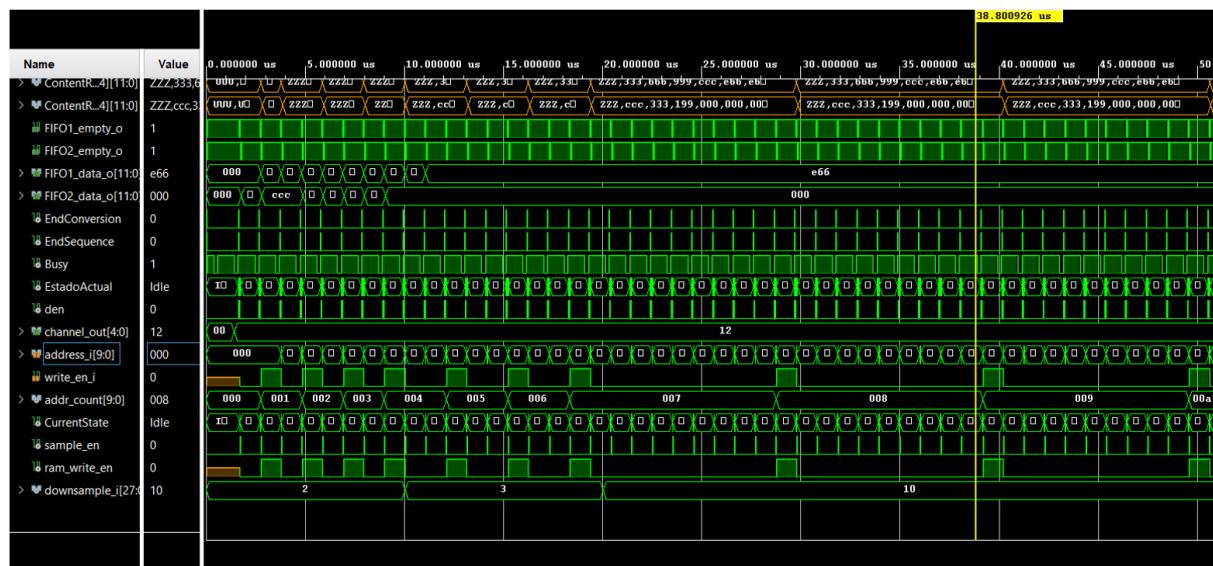


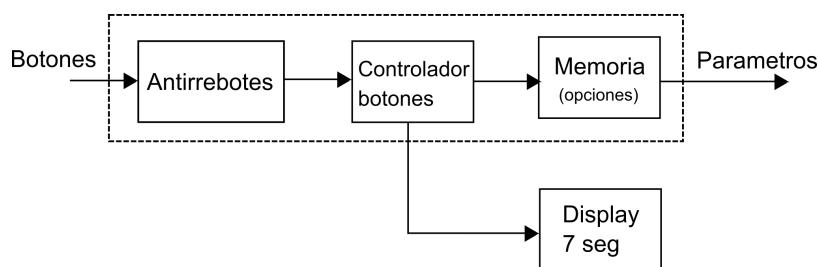
Figura 4.25 – Simulación del downsampling

## 4.7. Diseño de la modificación de parámetros

En este módulo, que se representa en la figura 4.26, se realiza la comunicación del usuario con el osciloscopio, permitiéndole modificar los siguientes parámetros:

- Voltios por división de ambos canales (VPD).
- Tiempo por división (TPD).
- *Threshold* del trigger.

El usuario modificará estos parámetros utilizando los 5 pulsadores que incorpora la Nexys 4 DDR y podrá ver los valores de estos parámetros tanto en la pantalla (véase el apartado 4.9.1) como en los *displays* de 7 segmentos, presentes en la placa.



**Figura 4.26 – Diagrama interfaz de usuario**

### 4.7.1. Antirrebotes

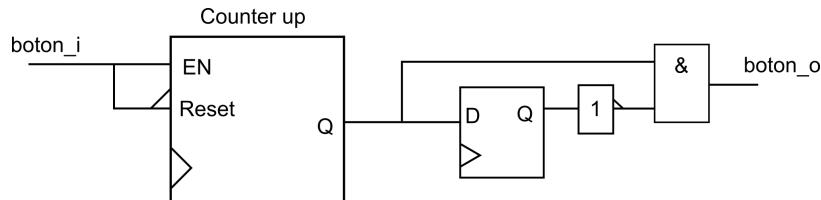
Cuando se acciona un pulsador o interruptor, la señal que se produce no es instantánea, si no que aparecen una serie de pulsos debido a los rebotes del contacto del accionamiento. Esto provoca un periodo de inestabilidad que es recomendable evitar si se quiere un correcto funcionamiento del sistema.

Los circuitos antirrebotes pretenden eliminar el ruido generado en la pulsación. Los métodos empleados pueden ser analógicos, en los que con un condensador es posible filtrar ese ruido, o digitales, que se pueden dividir en dos tipos:

- **Antirrebotes software:** existen diferentes métodos como el uso de interrupciones que se deshabilitan durante un periodo de tiempo una vez se ha pulsado el botón o establecer una rutina en la que se comprueba el tiempo que la señal es estable.
- **Antirrebotes hardware:** se establece el número de ciclos durante el que la señal debe ser estable. Una vez pasado este tiempo se considera que la señal es estable y se pone a 1 la salida.

En este caso, como se va a implementar en una FPGA se empleará el método del antirrebotes *hardware*. Para ello se decide implementar mediante un contador, cuya señal de habilitación es el botón en cuestión. De este modo, se cuenta el número de ciclos que la señal es estable y una vez alcanzado el límite del contador, se pone a nivel alto la salida. Además,

debido a que los botones se van a emplear para controlar registros de desplazamiento (véase el apartado 4.7.3) es recomendable que la salida del antirrebotes solo dure un ciclo de reloj. Esto se consigue incorporando un detector de flanco ascendente en la salida. De este modo, únicamente se produce un desplazamiento. En la figura 4.27 se recoge un diagrama de la implementación de este sistema.



**Figura 4.27 – Diagrama antirrebotes**

#### 4.7.2. Memoria opciones

Esta memoria es la encargada de almacenar los parámetros de configuración en vivo del osciloscopio. Existen varios métodos para implementar una memoria de estas características. Sin embargo, debido a que es una memoria de tamaño muy reducido que está siendo leída y modificada por diferentes módulos lo más óptimo es realizar un banco de registros. De este modo, se evitan los problemas de acceso simultaneo y la sensación de respuesta en tiempo real del usuario es mucho mayor.

Los parámetros que se deben almacenar tienen tamaños diferentes. Por un lado tenemos los valores de las escalas, que se establecen en 4 bits, mientras que el valor del *trigger* debe tener el mismo número de bits que el resultado de la conversión del ADC. Por lo tanto, se realiza el mapa de memoria recogido en la tabla 4.2.

**Tabla 4.2 – Mapa de la memoria de opciones**

Dirección ( <i>one-hot</i> )	Parámetro	Tamaño
0001	VPD1	4 bits
0010	VPD2	4 bits
0100	TPD	4 bits
1000	<i>Trigger</i>	12 bits

En la tabla se especifica una dirección en formato *one-hot*. Al tratarse de registros, cada uno con una entrada de habilitación, es recomendable utilizar este tipo de codificación, evitando tener que utilizar un decodificador.

#### 4.7.3. Controlador de los botones

Este módulo permite al usuario seleccionar y modificar los parámetros del osciloscopio almacenados en la memoria descrita en el apartado 4.7.2. Como se ha especificado, el direccionamiento se realiza con la codificación *one-hot*. Este direccionamiento se puede implementar

de manera sencilla con un registro de desplazamiento de 4 posiciones en el que haya precargado un registro a 1 y el resto a 0. De este modo, desplazando el 1 en ambos sentidos, pulsando los botones arriba o abajo, se puede seleccionar que registro va a ser modificado. Además, con el fin de que el usuario sepa cual es el registro que está seleccionando se emplean los LEDs integrados en la placa. Gracias a la codificación utilizada se enciende directamente el LED correspondiente al registro seleccionado.

Para la implementación de este módulo se realiza el diagrama de la figura 4.28. En primer lugar, tenemos el contador encargado de modificar los registros de los datos. Este contador dispone de una entrada a través de la que se cargará el valor que tiene el registro que se desea modificar a través de un multiplexor. Debido a que el contador es de 12 bits, necesarios para el *trigger*, y los valores de las escalas son de 4 bits, es necesario concatenar ceros hasta completar el tamaño del puerto. El *trigger* es un caso especial ya que se divide entre 64, por el motivo que se verá más adelante. Estos valores se cargan o se borran del contador cuando los pines de *Load* o *Reset* están a nivel alto, respectivamente. Además, cuenta con las típicas entradas para incrementar o decrementar una unidad y entradas para incrementar o decrementar 3 unidades, con el fin de poder hacer más cómoda la configuración.

En la parte derecha del diagrama tenemos los registros de los parámetros, habilitados por las variables “*parameterSelection*”, cuyo valor es el del registro de desplazamiento que se ha comentado anteriormente y “*writeEN*”, que se pondrá a nivel alto mientras se esté modificando un parámetro. A las entradas de estos registros se conecta el valor de la cuenta. En el caso de las escalas es necesario seleccionar únicamente los 4 bits menos significativos para ajustarse al tamaño del registro. En cuanto al *trigger*, su registro tiene el mismo tamaño que el valor de la cuenta. Sin embargo, es necesario multiplicar su valor por 64, ya que sino modificarlo mediante los botones sería demasiado tedioso, al tener que pulsar el botón hasta 2047 veces. Por este motivo, es necesario dividir este valor entre 64 al introducirlo en el contador.

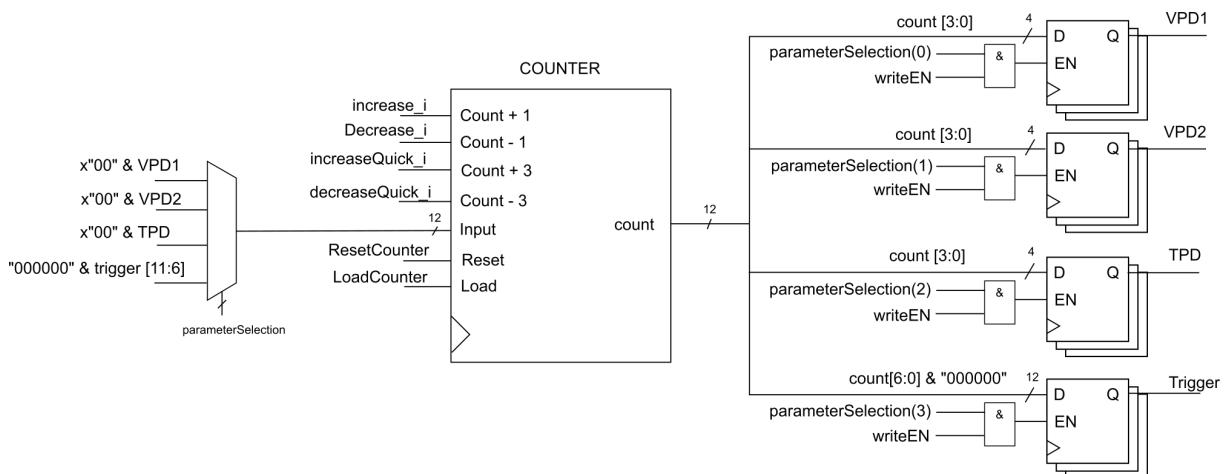
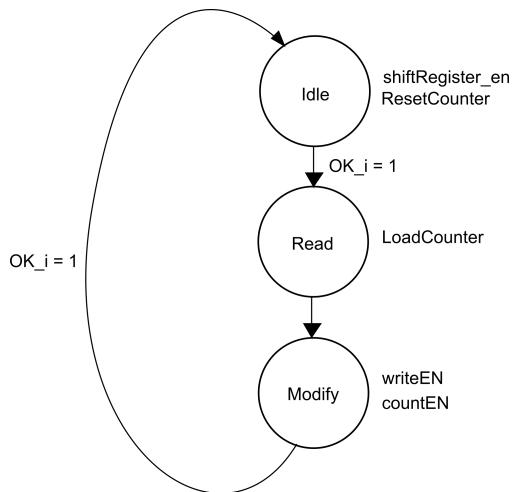


Figura 4.28 – Diagrama del controlador de los botones

Finalmente, para el control de este modulo se desarrolla la máquina de estados de la figura 4.29. Esta máquina tiene un estado inicial de *Idle*, en el que permanece mientras se están controlando los registros de desplazamiento y el contador está apagado y reseteado. Una vez

se pulsa el botón central, que se corresponde con la entrada “OK\_i” se pasa al estado *Read* en el que se carga el contador con el valor del parámetro seleccionado e inmediatamente se pasa al estado *Modify*, en el que se habilita el contador y los registros de los parámetros.



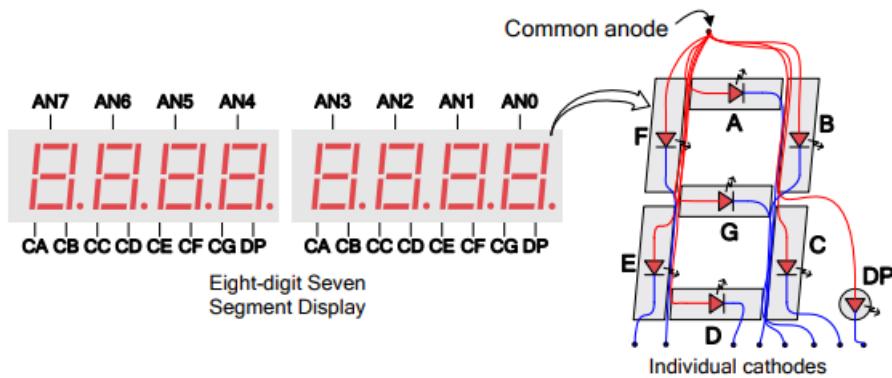
**Figura 4.29 – Máquina de estados del control de los botones**

#### 4.7.4. Display de 7 segmentos

Cuando el usuario está modificando el parámetro es necesario que pueda visualizar el valor actual. Esto se podría hacer en el monitor, ya que los parámetros están presentes en pantalla. Sin embargo, la Nexys 4 DDR dispone de *displays* de 7 segmentos en los que se pueden representar estos valores.

##### 4.7.4.1. Funcionamiento del display de 7 segmentos

La Nexys 4 DDR dispone de 8 *displays* de 7 segmentos de anodo común. Es decir, todos los segmentos de un mismo *display* tienen todos sus bornes positivos conectados a un mismo punto. Sin embargo, los cátodos están conectados a pines denominados de la A a la G y que son compartidos por todos los *displays*, como se puede ver en la figura 4.30. Esto hace que no se puedan encender varios *displays* simultáneamente, sino que habrá que hacer un barrido por todos ellos, poniendo cada uno de los ánodos a nivel alto y los cátodos que sea necesario para representar el dígito que se desee a nivel bajo. Este barrido debe realizarse a una frecuencia suficientemente alta como para que el ojo humano no perciba el parpadeo.



**Figura 4.30 – Diagrama de funcionamiento del display de la Nexys 4 DDR (fuente: [1])**

#### 4.7.4.2. Representación de la escala vertical

Como se ha especificado en el apartado 4.7.2, la escala vertical es un número de 4 bits. Es decir, tiene un valor entre 0 y 15 que se corresponden con los valores de la escala. Para obtener estos valores, se recurre a una memoria, cuyo mapa se recoge en la columna “VPD” de la tabla 4.3, en la que se almacena el valor en V/div para cada valor del contador.

**Tabla 4.3 – Mapas de las memorias de VPD, TPD y decodificador de 7 segmentos**

Dirección	VPD	TPD	7 segmentos
0	$D106_{16}$	$DD26_{16}$	0
1	$D206_{16}$	$D256_{16}$	1
2	$D506_{16}$	$D106_{16}$	2
3	$1006_{16}$	$D206_{16}$	3
4	$2006_{16}$	$D506_{16}$	4
5	$5006_{16}$	$1006_{16}$	5
6	$DD13_{16}$	$2006_{16}$	6
7	$DD23_{16}$	$5006_{16}$	7
8	$DD53_{16}$	$DD13_{16}$	8
9	$D103_{16}$	$DD23_{16}$	9
10	$D203_{16}$	$DD53_{16}$	-
11	$D503_{16}$	$D103_{16}$	U
12	$1003_{16}$	$D203_{16}$	S
13	$2003_{16}$	$D503_{16}$	off
14	$5003_{16}$	$1003_{16}$	E
15	$DD10_{16}$	$2003_{16}$	F

Como se puede ver en la tabla, hay dígitos que aparentemente no deberían estar ahí. Esto se debe a que a la hora de representar el número en los displays de 7 segmentos es necesario crear nuevas asociaciones para los dígitos que no se suelen emplear en los decodificadores de 7 segmentos. Por ello se diseña el decodificador recogido en la columna “7 segmentos” de

la tabla 4.3. De este modo es posible representar la escala de manera sencilla, en notación científica, pudiendo apagar los dígitos o añadiendo el signo menos al exponente según sea necesario.

#### 4.7.4.3. Representación de la escala horizontal

La representación de esta escala horizontal se realiza de manera análoga a la escala vertical, con la diferencia de que la escala es diferente y por lo tanto la decodificación del parámetro se realiza con la memoria cuyo mapa se recoge en la columna “TPD” de la tabla 4.3. Del mismo modo que en el apartado anterior, la decodificación a 7 segmentos se realiza con la columna “7 segmentos” de la tabla 4.3, representando el parámetro en notación científica en el *display*.

#### 4.7.4.4. Representación del trigger

Para la representación del *trigger* se puede recurrir a la representación del valor en voltios o la representación del valor digital. Debido a que, en la pantalla se muestra la posición del *trigger* en el *plot*; se decide representar únicamente el valor digital en el *display* de 7 segmentos al ser una transformación menos costosa y tratarse de una representación complementaria. De este modo, se realiza únicamente una conversión BCD (*Binary-Coded Decimal*) del valor digital del contador y mediante el decodificador de 7 segmentos, se representa el valor digital del *display*.

#### 4.7.4.5. Diagrama de la implementación

Teniendo en cuenta lo establecido en los apartados anteriores se realiza el diagrama de la figura 4.31. En primer lugar, en la parte izquierda del diagrama, se puede observar que los VPD de los canales 1 y 2 se multiplexan para poder compartir el mismo decodificador de VPD, ya que nunca se van a utilizar de manera simultanea. Sin embargo, el decodificador de TPD debe ser único, como se ha especificado en el apartado 4.7.4.3. Además, el *trigger* se convierte de binario a BCD. Estos datos ya convertidos, se multiplexan de nuevo en función del parámetro seleccionado. Este parámetro se registra para evitar sincronías. Finalmente, existen dos caminos que pueden tomar los datos, dependiendo de si es necesario representar el *trigger* o el valor de alguna escala:

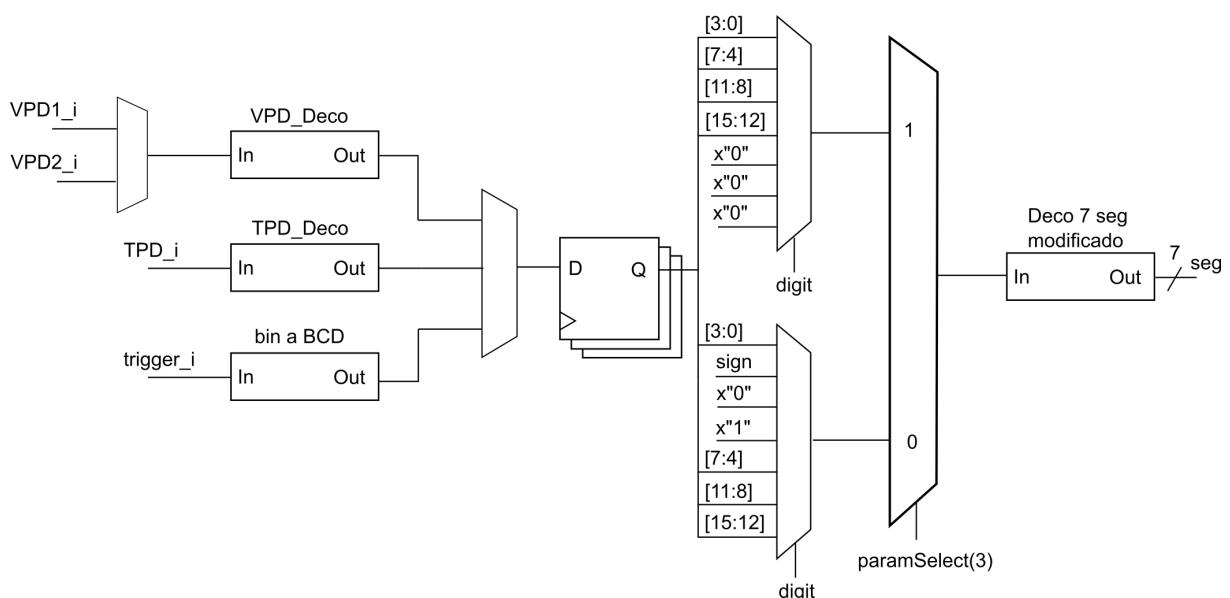
- Representación de una escala: en este caso la representación se realiza en notación científica, siguiendo el siguiente formato:
  - Dígito 1: es la cifra de las centenas. Se corresponde con los bits del 15 al 12.
  - Dígito 2: es la cifra de las decenas. Se corresponde con los bits del 11 al 8.
  - Dígito 3: es la cifra de las unidades. Se corresponde con los bits del 7 al 4.
  - Dígito 4: es el 1 de la base de la potencia de base 10.
  - Dígito 5: es el 0 de la base de la potencia 10.
  - Dígito 6: es el signo de la notación científica, necesaria cuando el valor es menor que 1.

- Dígito 7: es el exponente de la potencia de base 10. Se corresponde con los bits del 3 al 0. Permanece presente todo el tiempo, ya que el exponente más alto es el 0.

Debido a que tenemos caracteres como el signo menos, es necesario utilizar el decodificador de 7 segmentos recogido en la tabla 4.3.

- Representación del *trigger*: en este caso la representación se realiza en BCD, por lo que se puede emplear directamente un decodificador de 7 segmentos, multiplexando cada uno de los dígitos en función de *display* que se vaya a representar en ese momento. Sin embargo, se decide reutilizar el decodificador modificado, ya que nunca va a ser necesario representar números mayores de 9, por lo que es un decodificador válido.

Además se realiza el barrido de los ánodos para encender cada uno de los *displays* a una frecuencia de aproximadamente 244 Hz, suficiente para que el ojo humano lo detecte como que todos están encendidos estáticamente.



**Figura 4.31 – Diagrama de la representación en el *display* de 7 segmentos**

## 4.8. Diseño del cálculo de magnitudes

A la hora de utilizar un osciloscopio es especialmente útil poder visualizar ciertas magnitudes en pantalla sin necesidad de medirlas contando las divisiones del *plot*. Estas mediciones es interesante hacerlas sobre los datos que se van a representar en pantalla. Por ello, un buen método para conseguir esto es conectar este módulo directamente al bus de la señal, leyendo los datos cuando son mandados al modulo VGA y procesarlos según se requiera. Los cálculos más interesantes a realizar pueden ser:

#### 4.8.1. Tensión máxima

El cálculo de esta magnitud se realiza comparando el valor del dato nuevo con el valor de tensión máxima registrado. Si este dato es mayor, se almacena en el registro. Además, para que esta magnitud pueda ser representada en el monitor (véase el apartado 4.9.6) es necesario que se convierta el dato a BCD. En primer lugar, se convierte el dato de Ca2 a BCD, restándole 1 y negando el resultado cuando sea negativo. Después, se separa el signo de la magnitud y se convierte esta última a BCD. Finalmente, se concatena el signo y la magnitud en BCD y se envía al módulo del VGA para que lo represente en el monitor. En la figura 4.32 se puede ver un diagrama de la implementación de este cálculo.

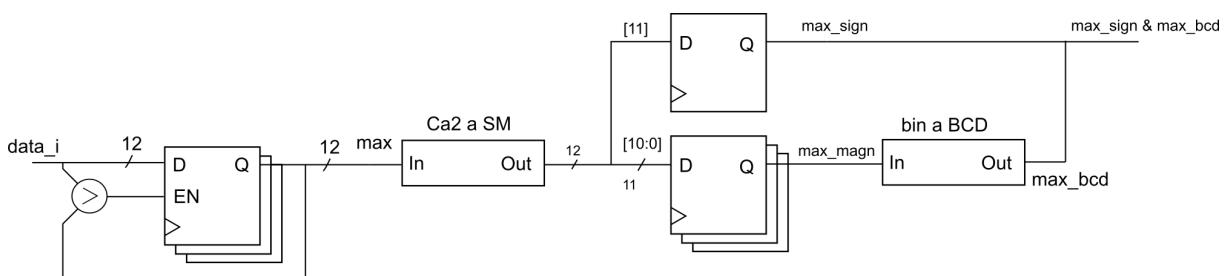


Figura 4.32 – Diagrama del cálculo de la tensión máxima

#### 4.8.2. Tensión mínima

Este cálculo se hace de manera análoga al de la tensión máxima, comparando que el dato sea menor que el valor mínimo registrado, como se puede ver en el diagrama de la figura 4.33. Finalmente se convierte de el dato a BCD, del mismo modo que se ha hecho con la tensión máxima.

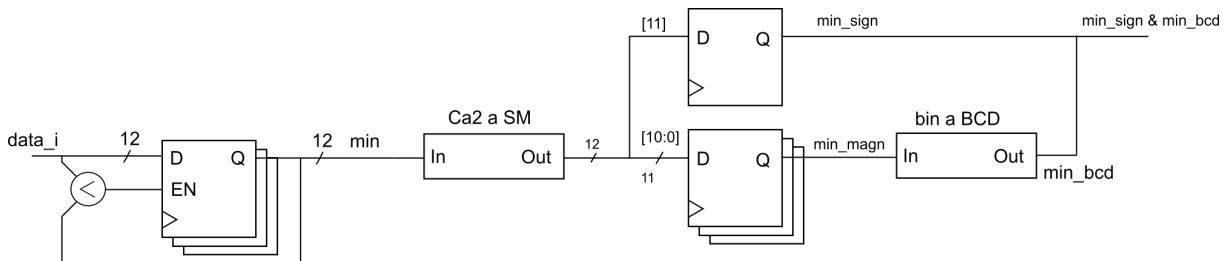
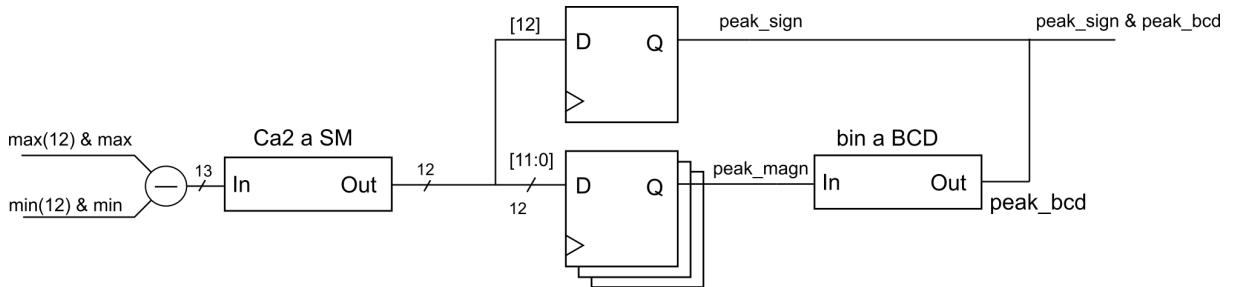


Figura 4.33 – Diagrama del cálculo de la tensión mínima

#### 4.8.3. Tensión pico a pico

Como hemos visto en el apartado 4.6.1.3 los datos con los que estamos trabajando están en complemento a 2. Por lo tanto, el cálculo de este parámetro consiste en realizar la resta con signo entre la tensión máxima y la mínima, calculadas en los apartados anteriores. El resultado de esta operación puede dar como resultado el doble del valor máximo que podemos representar con 12 bits. Por lo que es necesario añadir un bit más, realizando una extensión de signo de las entradas. Finalmente, con el fin de que esta magnitud sea representada en el

monitor, se convierte a BCD del mismo modo que los apartados anteriores. En la figura 4.34 se puede ver un diagrama de la implementación de este cálculo.



**Figura 4.34 – Diagrama del cálculo de la tensión de pico**

#### 4.8.4. Tensión media

El cálculo de este parámetro se puede realizar con la siguiente formula:

$$V_{avg} = \frac{\sum V_i}{PLOT\_WIDTH} \quad (4.3)$$

Para ello, será necesario realizar la división en punto fijo, multiplicando y dividiendo por una potencia de 2. Esto se realiza del siguiente modo.

$$V_{avg} = \frac{\sum V_i * floor(\frac{2^n}{PLOT\_WIDTH})}{2^n} \quad (4.4)$$

En la ecuación 4.5 la variable n es la cuantificación de la operación, por lo que cuanto mayor sea su valor, mayor será la precisión del resultado obtenido. En la figura 4.35 se puede ver como a medida que aumenta el numero de bits de la cuantificación, se reduce el error entre el valor real y el resultado obtenido por este método. A partir de 24 bit las lineas se superponen, al no haber un error muy pequeño para la resolución disponible. Por lo tanto, se empleará n = 24.

Con el fin de que esta operación no se deba reajustar en función del ancho del plot, se incorpora en el paquete de constantes el cálculo del resultado de la fracción que se encuentra en el numerador de la ecuación 4.4, a la que llamamos “operador”. Esto se realiza del siguiente modo:

$$QUANTIFICATION = 24$$

$$OPERATOR = \frac{2^{QUANTIFICATION}}{PLOT\_WIDTH}$$

De este modo, la operación que será necesario implementar será:

$$V_{avg} = (\sum V_i * OPERATOR) >> QUANTIFICATION \quad (4.5)$$

Donde “>> QUANTIFICATION” indica un desplazamiento de 24 posiciones a la dere-

cha, en este caso.

Por último, estos parámetros deberán ser convertidos a BCD para poder ser representados en pantalla por el módulo del VGA.

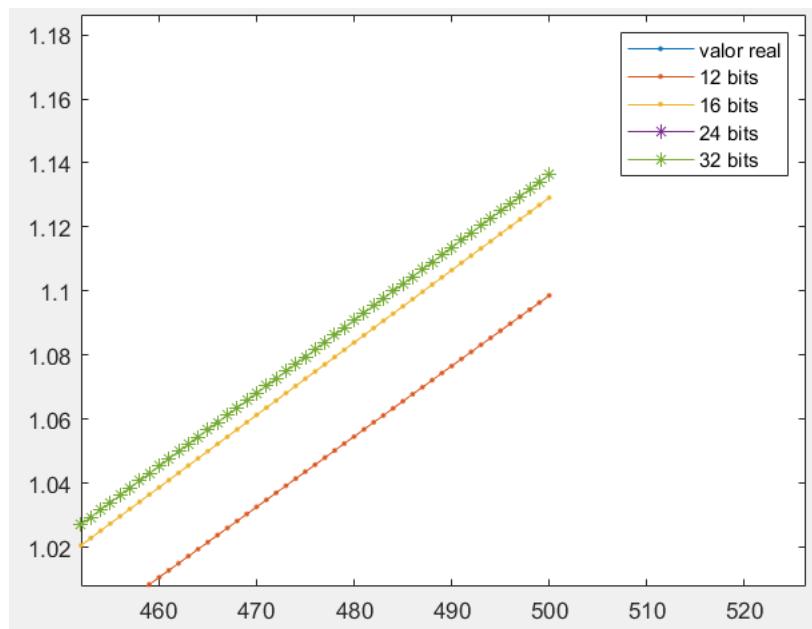


Figura 4.35 – Cuantificación del cálculo de tensión media

## 4.9. Diseño del VGA

### 4.9.1. El estándar VGA

El VGA diseñado por IBM para trabajar con pantallas de ordenador analógicas con una resolución de 640 x 480 píxeles y una frecuencia de refresco de 60 Hz. La comunicación con el monitor se realiza mediante el conector de tipo *D-sub* de 15 contactos, como el que se puede ver en la figura 4.36.



Figura 4.36 – Conector *D-sub* de 15 contactos

En la figura 4.37 se puede ver las diferentes zonas en las que se divide el VGA. En primer lugar esta la zona del *display* en la que se van representar todos los elementos y que se corresponde con la resolución de la pantalla. La otra zona que se puede ver está formada por el *front porch*, el *back porch* y el *sync pulse*. Estas zonas no se encuentran dentro de la

pantalla y cumplen una tarea de sincronización de la imagen. En cada refresco de la imagen es necesario emitir los pulsos de sincronización vertical y horizontal, respetando los tiempos establecidos para la resolución que se esté empleando.

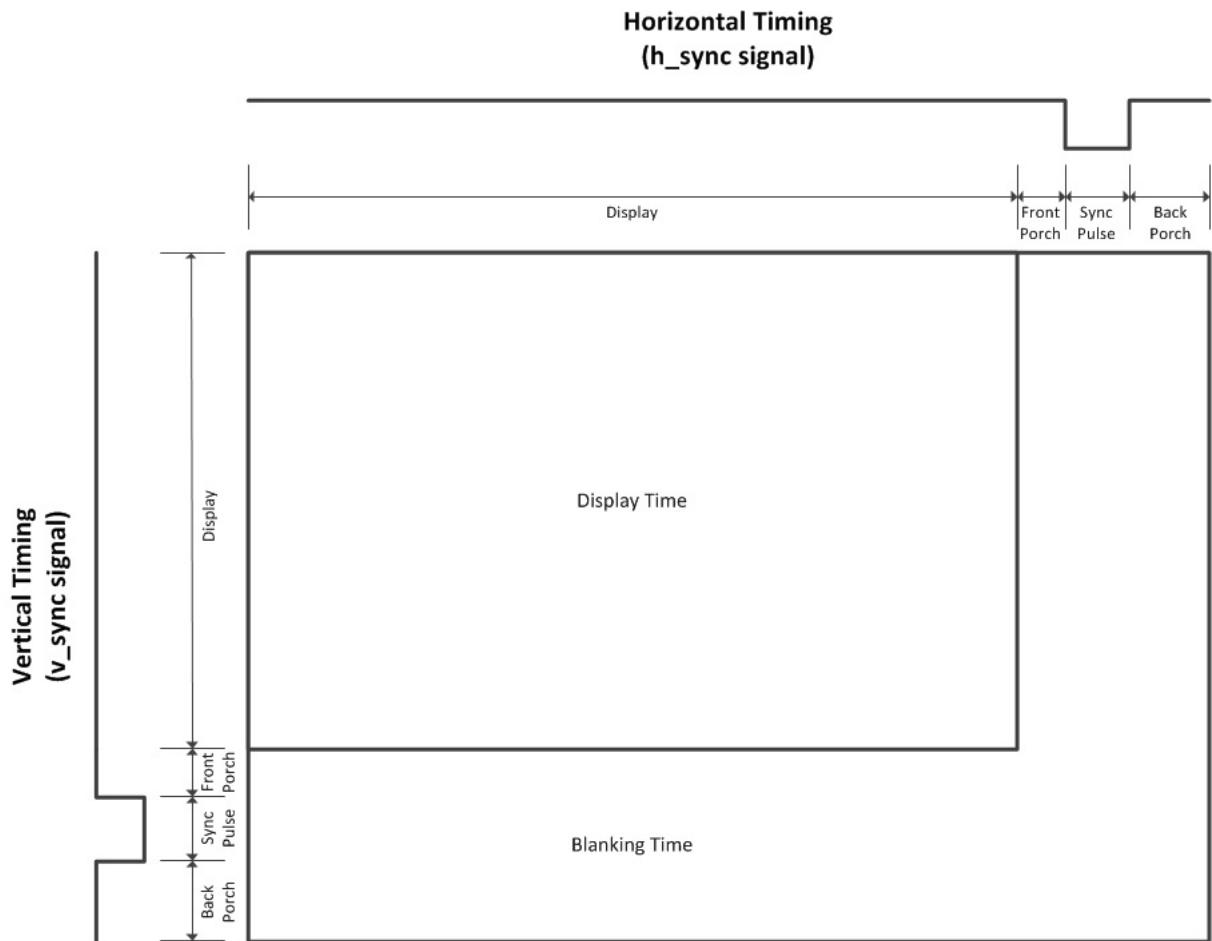


Figura 4.37 – Diagrama temporal del estándar VGA

En la actualidad, hay gran variedad de resoluciones y frecuencias de refresco que se pueden utilizar con este estándar. Sin embargo, en este caso se utilizará la resolución y frecuencia clásicas. Es decir, 640 x 480 píxeles y 60 Hz. Los parámetros temporales de esta configuración son:

Tabla 4.4 – Temporizaciones generales del VGA

#### Temporización general

Frecuencia de refresco	60 Hz
Refresco vertical	31.469 kHz
Frecuencia del píxel	25.175 MHz

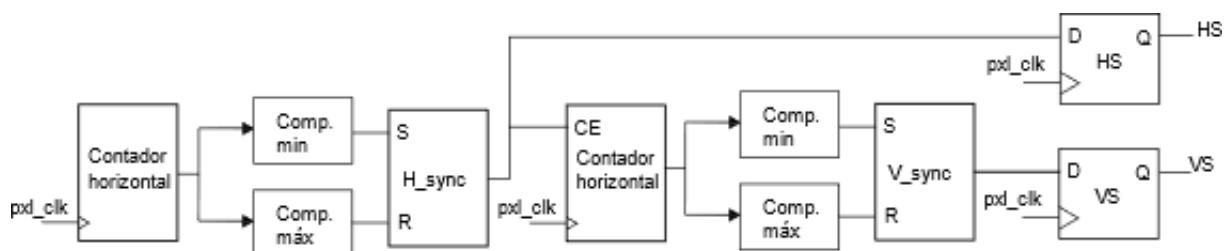
**Tabla 4.5 – Temporización horizontal del VGA**

<b>Temporización horizontal</b>	
Zona	Píxeles
<i>Visible area</i>	640
<i>Front porch</i>	16
<i>Sync pulse</i>	96
<i>Back porch</i>	48
<i>Whole line</i>	800

**Tabla 4.6 – Temporización vertical del VGA**

<b>Temporización vertical</b>	
Zona	Píxeles
<i>Visible area</i>	480
<i>Front porch</i>	10
<i>Sync pulse</i>	2
<i>Back porch</i>	33
<i>Whole line</i>	525

Con todo esto, es posible implementar el controlador VGA mediante el diagrama de la figura 4.38. En este diagrama se especifica como se generan las señales de sincronización horizontal y vertical, comparando la posición de los contadores con los valore entre los que se tiene que generar los pulsos. Además es necesario incorporar un divisor de frecuencia para conseguir que los contadores funcionen a la frecuencia del píxel de 25.175 MHz, como se especificada en la tabla 4.4. Esta señal será utilizada como una señal de habilitación de los registros y contadores, de manera que la señal de reloj siga siendo la del sistema global, asegurando el sincronismo.

**Figura 4.38 – Diagrama de la implementación del VGA**

Una vez hecho esto ya se puede comenzar a generar las imágenes que se van a representar en pantalla.

#### 4.9.2. Representación del *plot*

Este es el bloque encargado de generar el marco y la rejilla del *plot*, así como de representar las señales de ambos canales. Con el fin de evitar el uso excesivo de elementos de

memoria y aumentar el generalidad del código, se decide evitar la precarga del fondo de la interfaz en una memoria.

Para la representación del *plot* y su rejilla se establecen, en el paquete “oscilloscope\_pkg” (véase el apartado 4.5), unas dimensiones de 352 x 440 píxeles con 10 divisiones en el eje horizontal y 8 en el vertical. De este modo, se forman unas rejillas cuadradas de 44 x 44 píxeles.

#### 4.9.2.1. Generador de la rejilla

Como se ha comentado, con el fin de que el código sea lo más genérico posible se debe generar la rejilla en vivo en función de constantes de configuración que deben estar establecidas en un paquete de constates. Por lo tanto, el modo más correcto será implementar contadores de los ejes x e y, cuyos valores se compararán con las posiciones que deben tener las diferentes barras que componen el *plot*. Esta comparación dará como resultado el valor que debe tomar ese píxel.

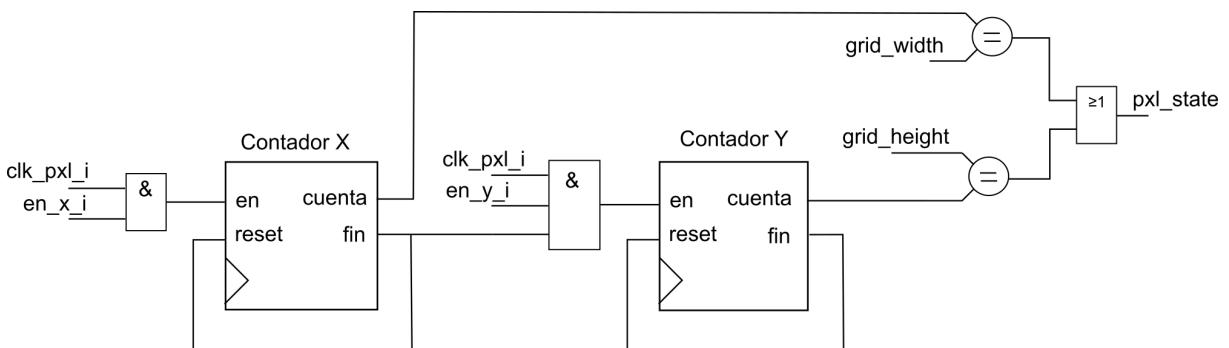


Figura 4.39 – Diagrama del generador de la rejilla

En la figura 4.39 se puede ver un diagrama de la implementación de la rejilla, en la que se establecen las entradas de habilitación “en\_x\_i” y “en\_y\_i”, que se encuentran a nivel alto cuando los contadores del VGA están en las posiciones que le corresponden al *plot*. Por otra parte las constantes “grid\_width” y “grid\_height” son las encargadas de establecer el ancho y alto de la rejilla. En este caso, se deciden calcular directamente introduciendo como constante el número de divisiones. De este modo, sabiendo el ancho y alto del *plot* y el número de divisiones, se puede calcular el ancho y alto de la rejilla. En este caso se establecen 10 divisiones horizontales y 8 verticales.

#### 4.9.2.2. Buffer de la señal

Con el fin de que la memoria del las señales pueda estar el mayor tiempo posible disponible para el ADC, se decide implementar un *buffer* que almacene las muestras que se van a representar en pantalla cada vez que se ha disparado el *trigger*. Como se ha comentado en el apartado 4.6.3, el *trigger* es el mecanismo encargado de la sincronización horizontal, para lo que envía una señal a nivel alto y una dirección de memoria cuando se dispara. Estas señales es necesario registrarlas, ya que de lo contrario cada disparo del *trigger* se modificaría la di-

rección de memoria, perdiéndose el punto del primer disparo. Una vez, se ha representado la señal correspondiente en el *plot* se resetean los registros del trigger y la dirección de memoria.

En el apartado 4.9.1 hemos visto que los contadores del VGA se extienden fuera de la pantalla para poder realizar la sincronización. En esos instantes de tiempo en los que se hace la sincronización se puede asegurar con certeza que la modificación del *buffer* es segura, ya que no puede estar siendo leído. Por lo tanto, es un buen momento para realizar la copia de los datos. Debido a que el bus que se emplea es compartido es necesario decidir que módulo es el “dueño” del bus. Teniendo en cuenta que la escritura de la memoria de la señal se hace por medio de una FIFO que puede almacenar datos mientras el bus está ocupado y que el tiempo del que se dispone para hacer la copia es limitado, se decide que es este módulo el que debe controlar el bus. Por lo tanto, este módulo tomará posesión de los buses de datos para actualizar el *buffer* y durante el resto del tiempo cederá el bus para la escritura de la memoria de la adquisición de datos.

En cuanto a la copia de los datos, esta no se puede realizar directamente, ya que no existe una asociación directa con el valor digital de la señal con los píxeles que se deben encender y que dependen de la escala vertical que se haya seleccionado. Por lo tanto, es necesario procesar los datos leídos de la memoria para poder almacenar en el *buffer* el pixel que debe ser encendido para representar la señal. Para ello, se realiza la siguiente operación:

$$pixelValue = GND - data_i * \frac{HEIGHT\_MAX}{2047} \quad (4.6)$$

Siendo “HEIGHT\_MAX” la amplitud en píxeles máxima que tendría una señal de 0.5 V para el VPD establecido y GND la coordenada Y del eje abscisas del *plot*. Es necesario restarle el dato a GND, ya que el valor del eje Y aumenta hacia abajo.

Anteriormente se ha comentado, que en el *plot* se le realizan 8 divisiones verticales. Sabiendo que el *plot* tiene una altura de 352 píxeles, se obtiene que la rejilla tiene 44 píxeles de alto. Esos 44 píxeles se corresponden con 0.5 V cuando tenemos un VPD de 0.5V. Por lo tanto, se puede hacer la siguiente conversión:

$$factorConversion = \frac{GRID\_HEIGHT * 0,5}{VPD} \quad (4.7)$$

Para simplificar el cálculo del factor de conversión se realiza el cálculo en función de los valores de VPD recogidos en la tabla 4.3 y se introduce los resultados en una memoria cuyas direcciones coincidan con las direcciones de la memoria del decodificador de VPD. De este modo, se obtiene la memoria de la tabla 4.7

**Tabla 4.7 – Mapa de la memoria para el factor de conversión**

VPD	Factor	Factor (Hexa)
0	$2,2 * 10^6$	2191C0
1	$1,1 * 10^6$	10C8E0
2	440000	6B6C0
3	220000	35B60
4	110000	1ADB0
5	44000	ABE0
6	22000	55F0
7	11000	2AF8
8	4400	1130
9	2200	898
10	1100	44C
11	440	1B8
12	220	DC
13	110	6E
14	44	2C
15	22	16

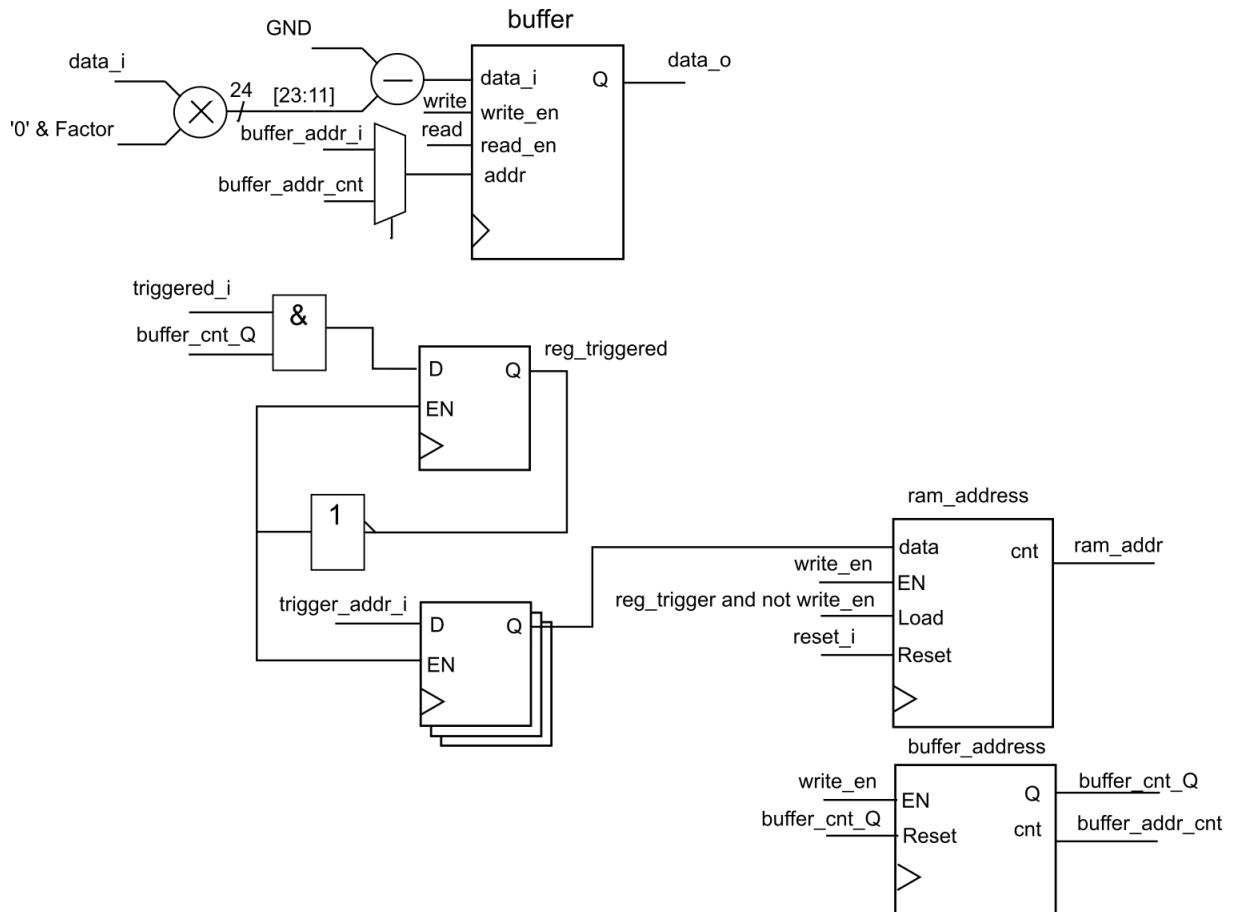
En cuanto al valor de GND, este sera igual a:

$$GND = PLOT\_START\_Y + 4 * GRID\_HEIGHT \quad (4.8)$$

Finalmente, la expresión con la que produce la conversión será:

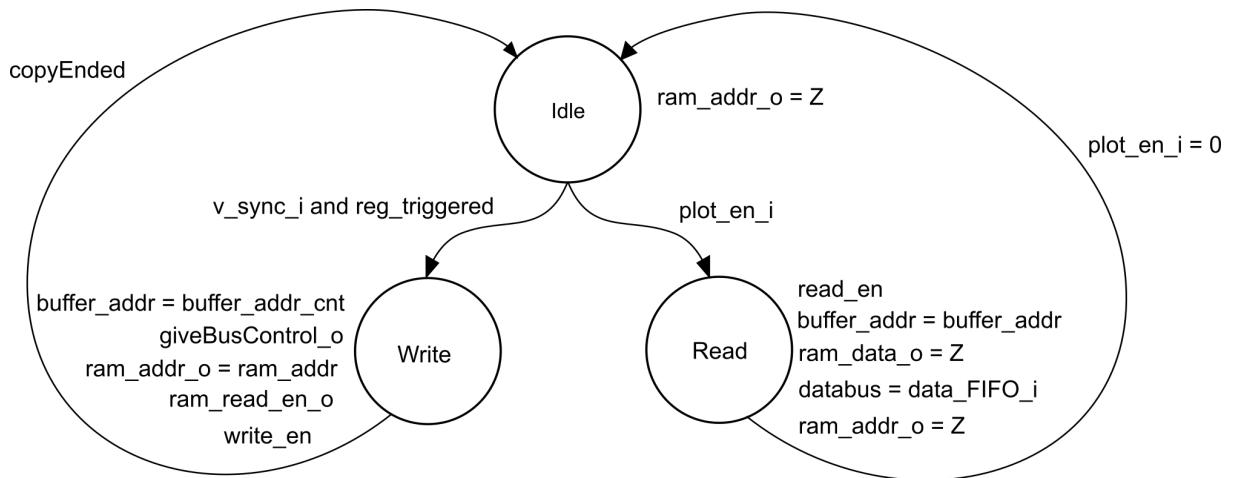
$$pixelValue = GND - \frac{data\_i * Factor}{2047} \quad (4.9)$$

En la figura 4.40 se puede ver un diagrama de la implementación del *buffer*, en la que, como se ha comentado, se registran las señales de *trigger* y la dirección en la que se produce y en base a eso se hace el direccionamiento del *buffer* y la memoria.



**Figura 4.40 – Diagrama de la implementación del Buffer**

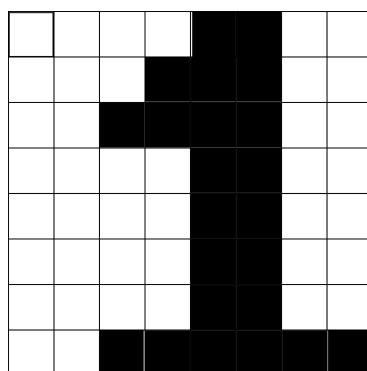
Para el control de este proceso se diseña la máquina de estados de la figura 4.41. En un inicio la máquina se encuentra en el estado de *Idle*, en el que no se producen ni lecturas ni escrituras. En este momento, pueden darse dos situaciones. Por un lado, cuando se entra en las coordenadas del *plot* se pasa al estado *Read*, en el que se produce la lectura del *buffer* y la representación de los datos en el monitor. Por otro lado, si se ha producido un *trigger* y se produce el pulso de sincronización vertical, indicando que las coordenadas están fuera de la pantalla, se pasa al estado *Write*, en el que se lee la memoria de las señales para sobrescribir el *buffer*.

**Figura 4.41 – Maquina de estados del Buffer**

#### 4.9.3. Representación de los parámetros

La representación de los parámetros se realiza de manera similar a la representación en el *display* de 7 segmentos del apartado 4.7.4. En primer lugar, es necesario convertir el valor proporcionado por el controlador de los botones (véase el apartado 4.7.3), mediante las memorias recogidas en las tablas 4.3. Una vez se tiene el parámetro transformado es necesario generar un mapa de bits de cada uno de los dígitos que se van a utilizar. Estos se almacenan en una memoria. Este mapa tiene unas dimensiones de 8 x 8 píxeles, que se corresponden con 8 direcciones de memoria y 8 bits por palabra, como el que se puede ver en la figura 4.42. Sin embargo, se consideran los números de 10 píxeles de ancho, de modo que la separación entre los dígitos sea mayor.

Con este mapa se genera una memoria en la que las casillas en blanco son ceros y las casillas en negro son unos y cuyo mapa se recoge en la tabla A.1 que se encuentra en el Anexo. Las posiciones de esta memoria están asociadas a los diferentes dígitos, y conociendo las coordenadas de la pantalla en la que se encuentre se leerá el valor del píxel.

**Figura 4.42 – Ejemplo mapa de bits de los caracteres**

#### 4.9.3.1. Representación de los voltios por división

En el diagrama de la figura 4.43 se puede ver la implementación del módulo que representa los voltios por división en pantalla. Este modulo está formado por 3 contadores. En primer lugar, tenemos el contador en el eje X, encargado de contar las columnas de la representación. En segundo lugar, tenemos el contador del dígito, directamente relacionado con el contador del eje X. Cuando este contador llega a 10, el contador del dígito, que controla el multiplexor encargado de enviar la dirección de memoria, aumenta una unidad. El orden de los datos representados es:

- Dígito 1: es el canal que se va a representar.
- Dígito 2: se incorporan dos puntos para separar el canal de los VPD.
- Dígito 3: es la cifra de las centenas. Se corresponde con los bits del 15 al 12.
- Dígito 4: es la cifra de las decenas. Se corresponde con los bits del 11 al 8.
- Dígito 5: es la cifra de las unidades. Se corresponde con los bits del 7 al 4.
- Dígito 6: es el prefijo del sistema internacional.
- Dígito 7: es el símbolo de los voltios (V).
- Dígito 8: es la barra inclinada, que indica que el valor representado se corresponde con cada una de las divisiones.

Finalmente, el contador del eje Y es el encargado de recorrer las filas de cada dígito. En la tabla A.1, correspondiente a la memoria de caracteres, se puede ver que para cada número de carácter se le asocian 8 direcciones. Esto hace que la primera dirección del carácter sea 8 veces el número que tiene asignado. Además, es necesario que por cada línea que se aumente, se aumente también una unidad la dirección de memoria. Por lo tanto, habría que realizar la siguiente operación:

$$mem\_addr = addr * 8 + cnt\_y \quad (4.10)$$

Una manera sencilla de realizar esta operación es concatenando las señales “addr” y “cnt\_y” como se puede ver en la figura 4.43. De este modo, como “cnt\_y” tiene 3 bits, se realiza la multiplicación por 8 y la suma de su valor simultáneamente.

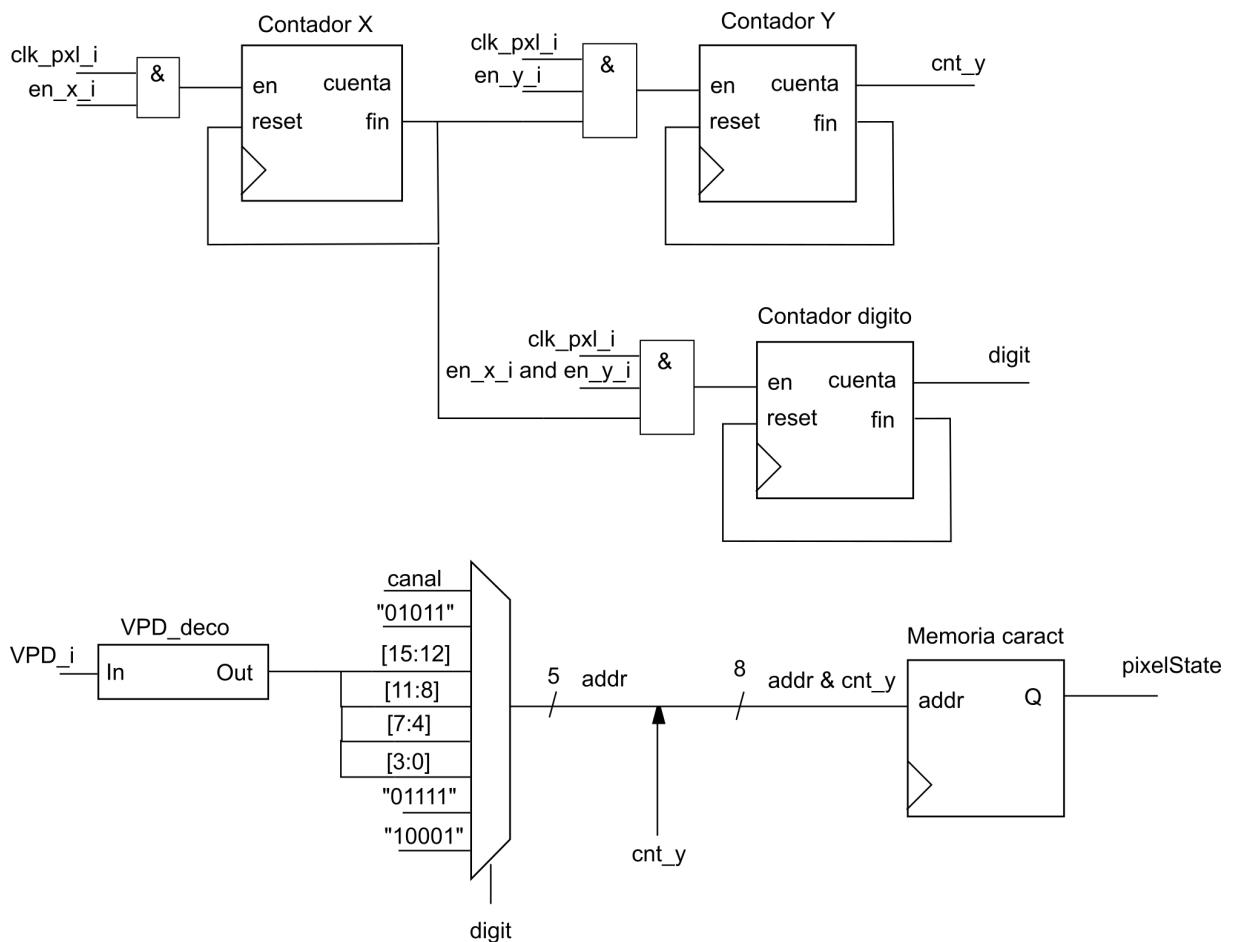


Figura 4.43 – Diagrama de la representación de los voltios por división

#### 4.9.3.2. Representación del tiempo por división

Como se puede ver en la figura 4.44, esta representación se realiza del mismo modo que VPD, salvo por la asignación de ciertas direcciones de memoria, ya que en este caso se representan los siguientes dígitos:

- Dígito 1: espacio en blanco. Se deja un espacio para poder reutilizar el bloque de VPD.
- Dígito 2: espacio en blanco. Se deja un espacio para poder reutilizar el bloque de VPD.
- Dígito 3: es la cifra de las centenas. Se corresponde con los bits del 15 al 12.
- Dígito 4: es la cifra de las decenas. Se corresponde con los bits del 11 al 8.
- Dígito 5: es la cifra de las unidades. Se corresponde con los bits del 7 al 4.
- Dígito 6: es el prefijo del sistema internacional.
- Dígito 7: es el símbolo de los segundos (s).
- Dígito 8: es la barra inclinada, que indica que el valor representado se corresponde con cada una de las divisiones.

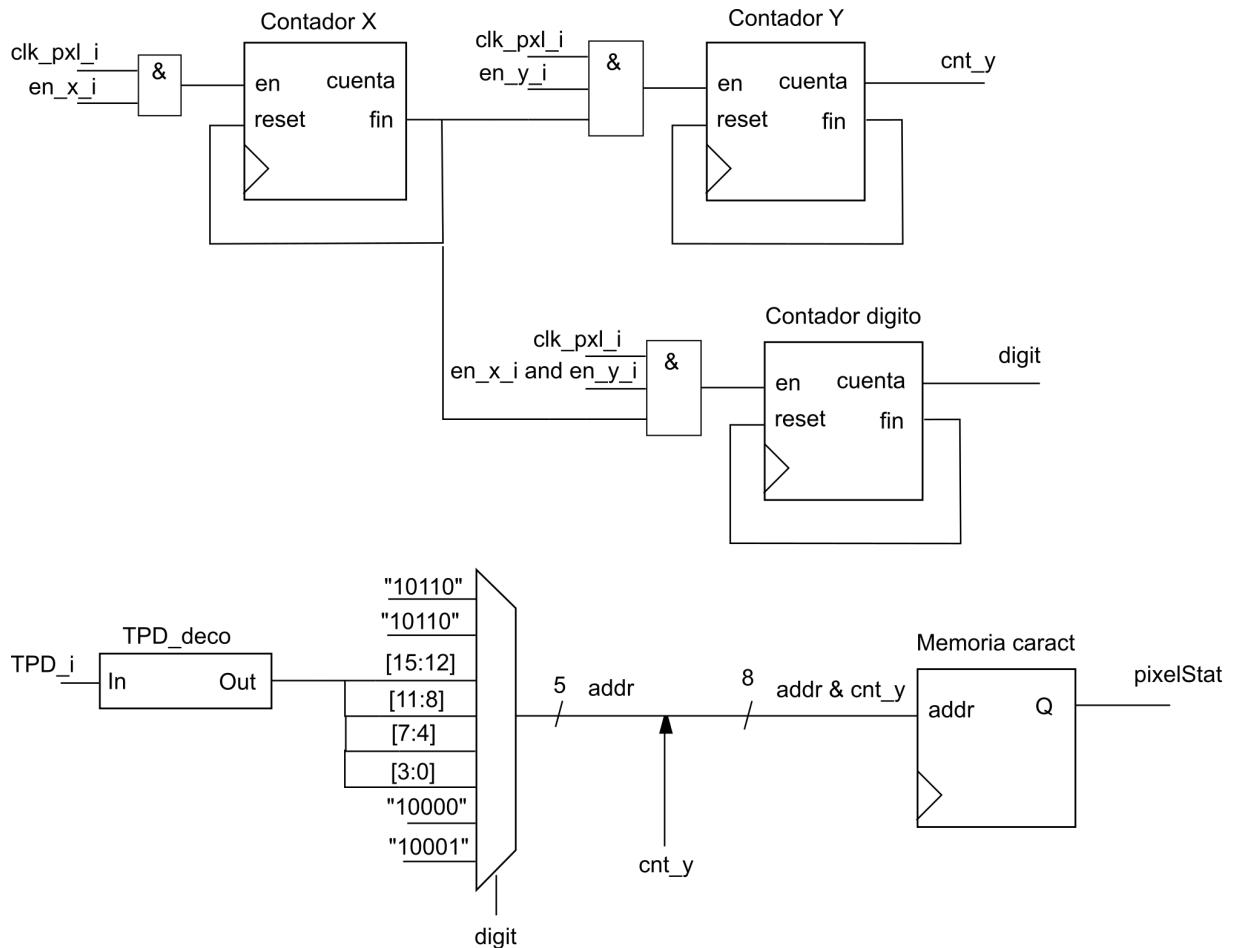


Figura 4.44 – Diagrama de la representación del tiempo por división

#### 4.9.4. Representación de símbolos

La representación de los símbolos se realiza de manera similar a las representaciones de los caracteres, pero utilizando la memoria de símbolos, cuyo mapa de memoria se recoge en la tabla A.2 recogida en el anexo. En este caso el mapa de bits generado tiene unas dimensiones de 9 x 9 píxeles, como el ejemplo que se puede ver en la figura 4.45.

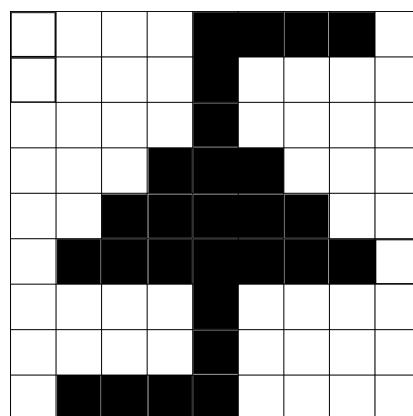


Figura 4.45 – Ejemplo mapa de bits de los símbolos

En este caso el número de símbolos a representar es reducido, contando únicamente con 3 símbolos:

- Símbolo de flanco ascendente: este símbolo esta presente cuando se selecciona el *trigger* por flanco ascendente poniendo el switch correspondiente a .
- Símbolo de flanco descendente: este símbolo esta presente cuando se selecciona el *trigger* por flanco ascendente mediante el switch indicado.
- Símbolo de la flecha del *trigger*:

#### 4.9.4.1. Representación de la flecha del trigger

Este símbolo indica cual es el valor del *trigger*. Para eso se posiciona al lado izquierdo del *plot* y en función del valor del *trigger* la flecha subira o bajara. Además, es necesario que la flecha se corresponda al valor de tensión correspondiente para todos los valores de ajuste vertical. Esto se consigue utilizando la ecuación 4.6, del mismo modo que se hizo en el apartado 4.9.2.2.

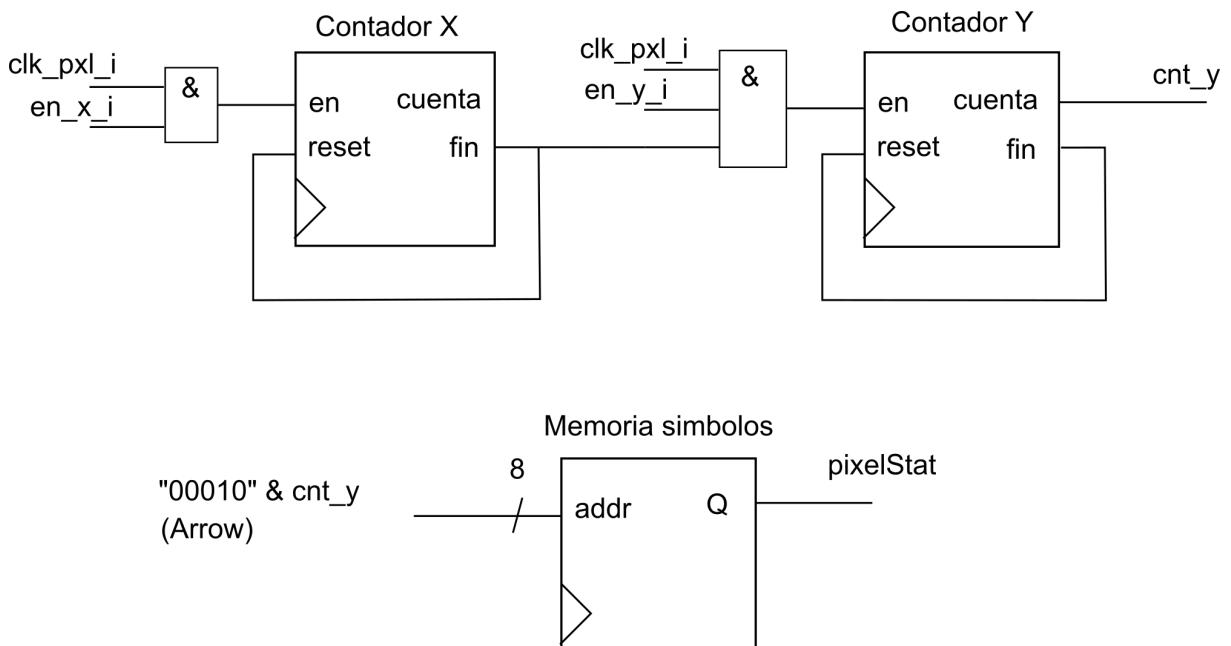
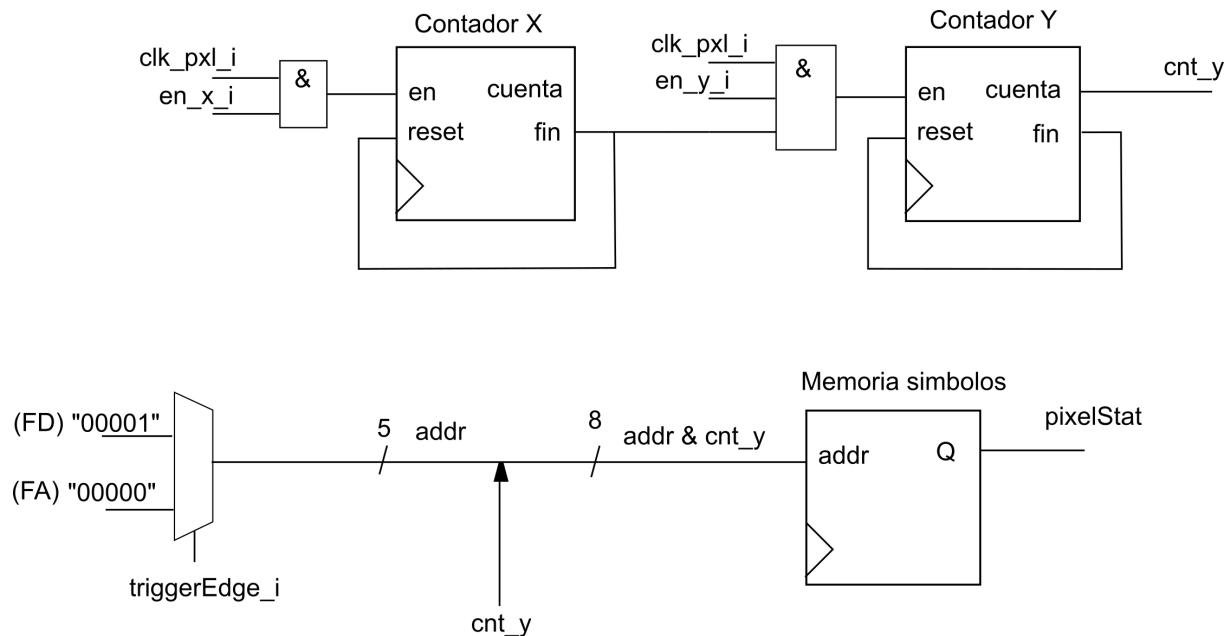


Figura 4.46 – Diagrama de la representación de la flecha del trigger

#### 4.9.4.2. Representación de la flanco del trigger

El flanco del *trigger* es controlado por el *switch* que tiene asociada la entrada “triggerEdge.i”, tal y como se especifica en el apartado . Cuando esta entrada está a nivel alto, indica que el flanco es ascendente y por lo tanto este es el que debe aparecer en pantalla. Sin embargo, cuando está a nivel bajo, es el flanco descendente el que debe aparecer. Esto se consigue mediante un multiplexor que selecciona la dirección correspondiente a uno u otro símbolo en función del valor de esta entrada, como se puede ver en la figura 4.47. Por último, y tal como

se hizo en el resto de representaciones se concatena el número del símbolo con el valor del contador Y para obtener la dirección que corresponde

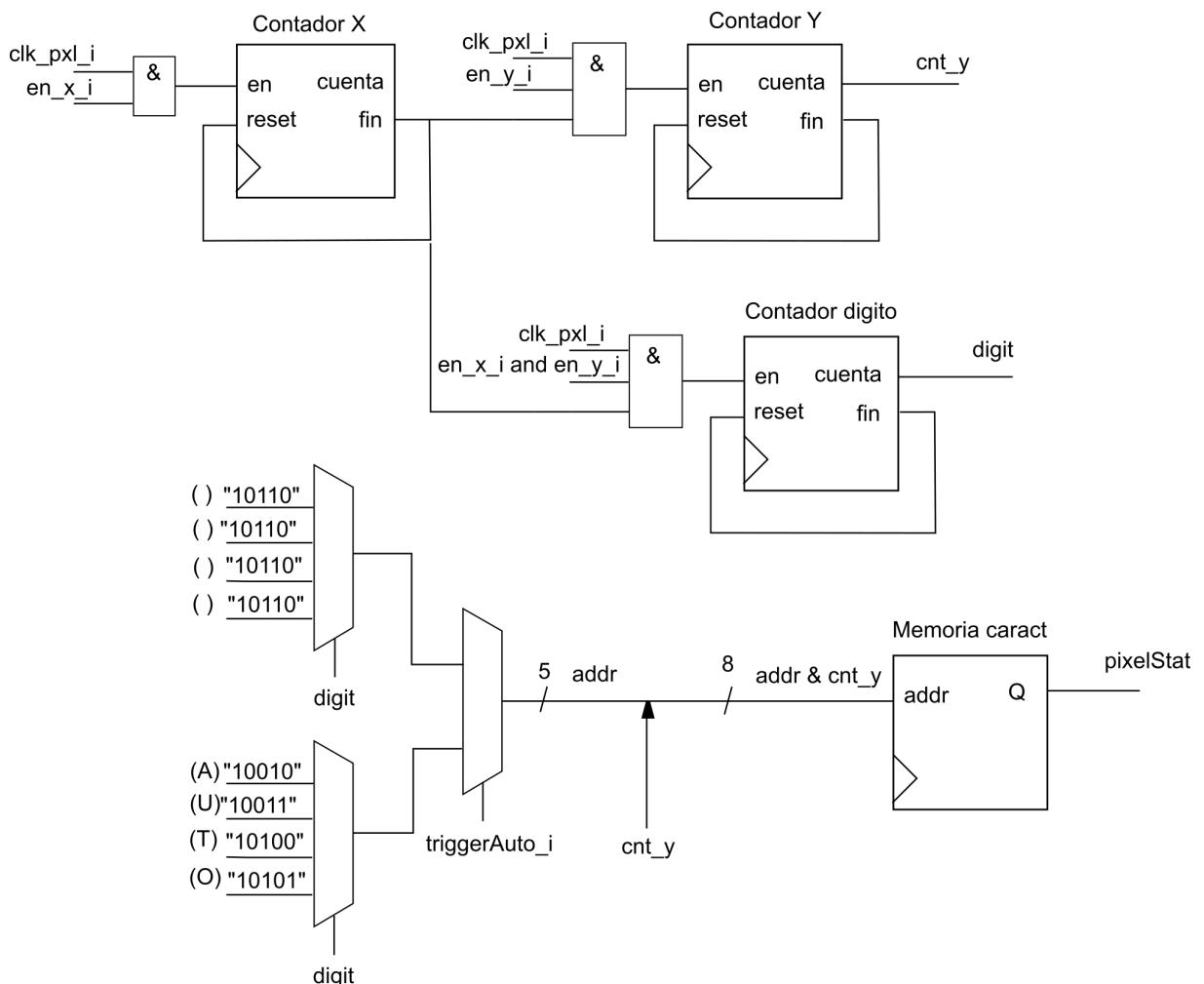


**Figura 4.47 – Diagrama de la representación del flanco del trigger**

#### 4.9.5. Representación del indicador de trigger automático

Como se puede ver en la figura 4.48, esta representación se realiza de manera similar a como se ha explicado en el apartado 4.9.3.1. Sin embargo en este caso, la visualización del parámetro depende de la señal “triggerAuto\_i” que, como se ha dicho en el apartado 4.9.4.2 está asociada a un *switch* de la placa Nexys. Por lo tanto, cuando esta señal esté a nivel alto, se representará la palabra “AUTO” en el monitor. Esto se realiza mediante el multiplexor controlado por esta señal, representándose los dígitos de la siguiente manera:

- Dígito 1: letra A cuando triggerAuto = 1, en caso contrario se coloca un espacio en blanco.
- Dígito 2: letra U cuando triggerAuto = 1, en caso contrario se coloca un espacio en blanco.
- Dígito 3: letra T cuando triggerAuto = 1, en caso contrario se coloca un espacio en blanco.
- Dígito 4: letra O cuando triggerAuto = 1, en caso contrario se coloca un espacio en blanco.



**Figura 4.48 – Diagrama de la representación del indicador de *trigger* automático**

#### 4.9.6. Representación de las mediciones

La representación de las mediciones se realiza de manera similar a la representación de los parámetros descrita en el apartado 4.9.3.1. La diferencia de esta es que, en este caso, la representación es de 13 dígitos. En cuanto a las diferencias entre las representaciones de las mediciones, estas son únicamente el nombre de la magnitud, como se verá en los apartados siguientes.

##### 4.9.6.1. Tensión máxima

En esta representación se representan las tensiones máximas de ambos canales con el formato “Vmax = XXX.X mV”. La selección de la dirección de memoria se realiza mediante el multiplexor controlado por el número de dígito que se necesite representar, como se puede ver en el diagrama de la figura 4.49. En este caso las direcciones de la 0 a la 3 que se pueden ver en la figura serán:

- Dir0: se corresponde con la V, cuya dirección es 01111.
- Dir1: se corresponde con la m, cuya dirección es 01110.

- Dir2: se corresponde con la a, cuya dirección es 11100.
- Dir3: se corresponde con la x, cuya dirección es 10111.

En cuanto al signo, este se corresponde con el bit 16 de la magnitud, cuando es 1 aparece el signo negativo, cuya dirección es “11010”, y cuando es 0 se pone un espacio en blanco, cuya dirección es “10110”. Para el resto de parámetros se emplean las direcciones que se pueden ver en la figura 4.49, ya que son constantes en la representación de todas las magnitudes.

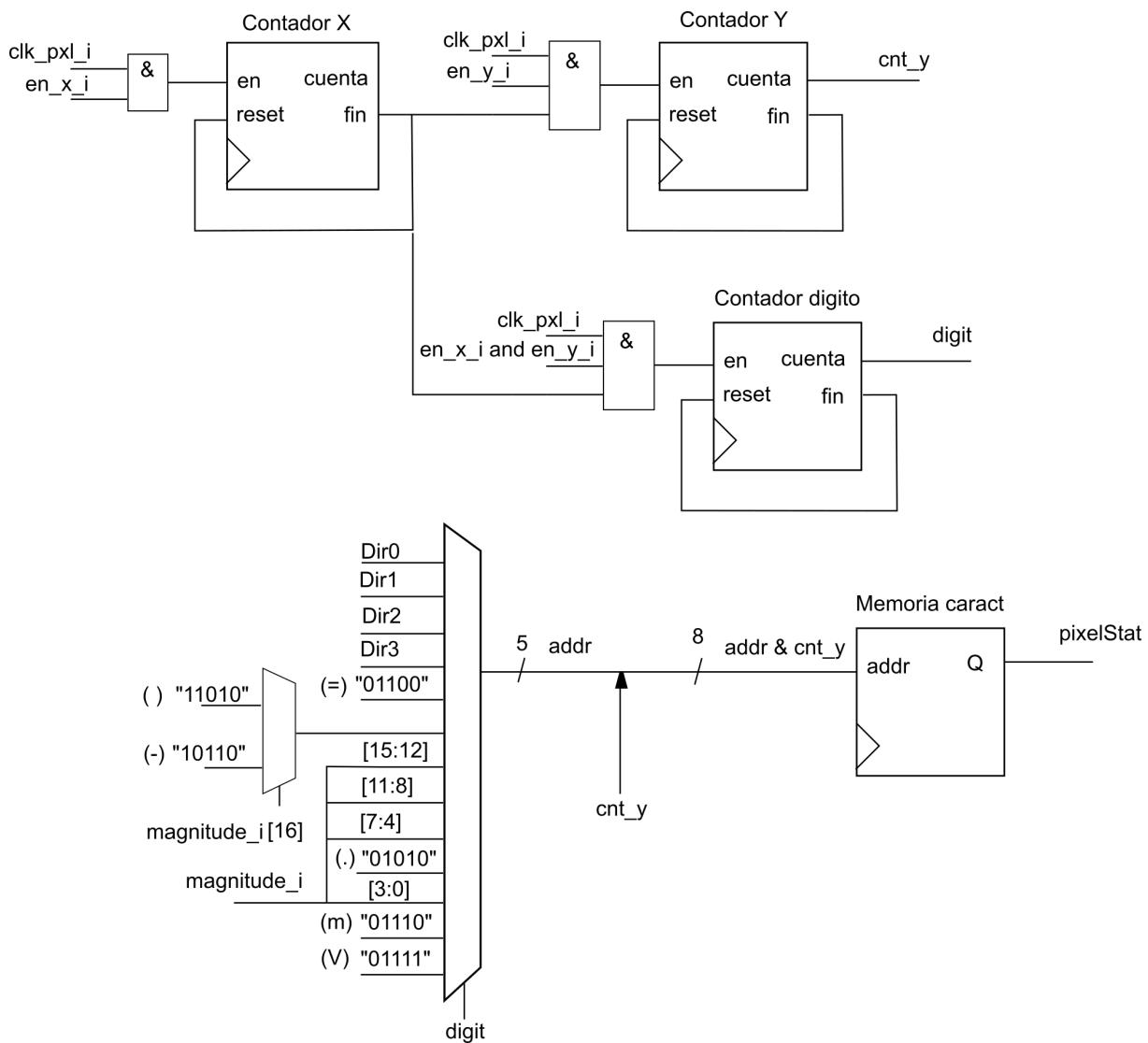


Figura 4.49 – Diagrama de la representación de la tensión máxima

#### 4.9.6.2. Tensión mínima

La representación de este parámetro se hace del mismo modo que en el apartado 4.9.6.1, pero modificando las primeras direcciones para que se represente el nombre “Vmin”, quedando las siguientes direcciones del 0 al 3:

- Dir0: se corresponde con la V, cuya dirección es 01111.

- Dir1: se corresponde con la m, cuya dirección es 01110.
- Dir2: se corresponde con la i, cuya dirección es 11000.
- Dir3: se corresponde con la n, cuya dirección es 11001.

#### 4.9.6.3. Tensión pico a pico

La representación de este parámetro se hace del mismo modo que en el apartado 4.9.6.1, pero modificando las primeras direcciones para que se represente el nombre “Vpp”, quedando las siguientes direcciones del 0 al 3:

- Dir0: se corresponde con la V, cuya dirección es 01111.
- Dir1: se corresponde con la p, cuya dirección es 11011.
- Dir2: se corresponde con la p, cuya dirección es 11011.
- Dir3: se corresponde con un espacio, cuya dirección es 10110.

#### 4.9.6.4. Tensión media

La representación de este parámetro se hace del mismo modo que en el apartado 4.9.6.1, pero modificando las primeras direcciones para que se represente el nombre “Vavg”, quedando las siguientes direcciones del 0 al 3:

- Dir0: se corresponde con la V, cuya dirección es 01111.
- Dir1: se corresponde con la a, cuya dirección es 11100.
- Dir2: se corresponde con la v, cuya dirección es 11101.
- Dir3: se corresponde con la g, cuya dirección es 01100.

### 4.10. Simulación del sistema

Para comprobar el sistema antes de cargarlo en la placa, se emplea el simulador de VGA disponible en la web <https://ericeastwood.com/lab/vga-simulator/>. En esta página se introduce un fichero de texto que contiene todas las señales necesarias del estándar VGA. Este fichero se realiza por medio del *testbench*, utilizando la librería “std.textio”, y debe tener el formato que se puede ver en la figura 4.50, donde las columnas de izquierda a derecha se corresponden con el tiempo, la sincronización horizontal, la sincronización horizontal, el rojo, el verde y el azul.

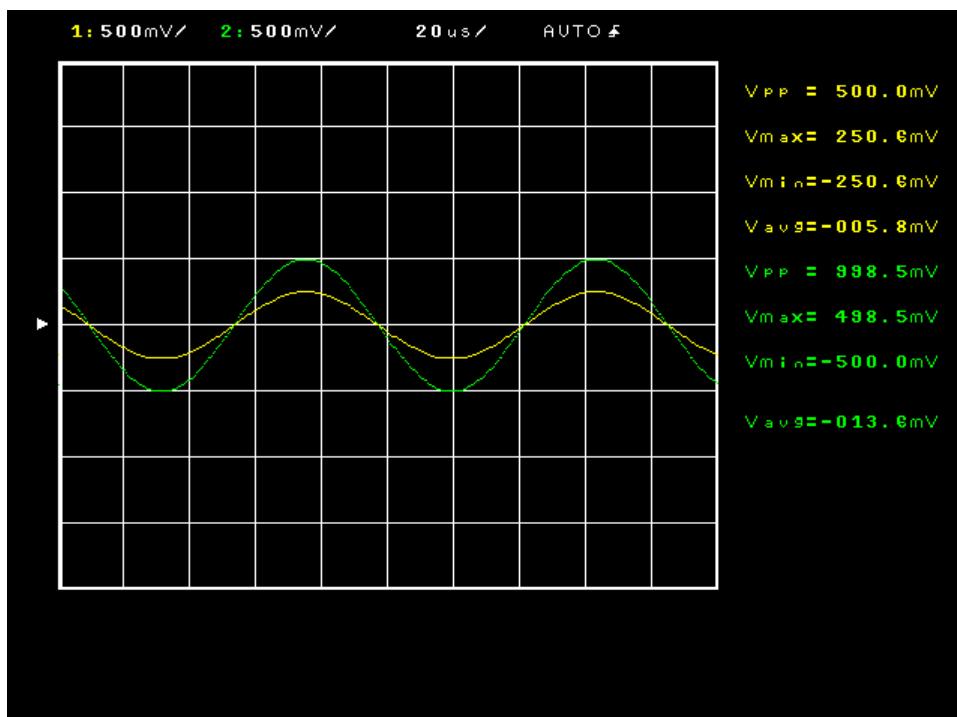
```

40 ns: 1 1 000 000 000
80 ns: 1 1 000 000 000
120 ns: 1 1 000 000 000
160 ns: 1 1 000 000 000
200 ns: 1 1 000 000 000
240 ns: 1 1 000 000 000
280 ns: 1 1 000 000 000
320 ns: 1 1 000 000 000
360 ns: 1 1 000 000 000
400 ns: 1 1 000 000 000
440 ns: 1 1 000 000 000
480 ns: 1 1 000 000 000
520 ns: 1 1 000 000 000
560 ns: 1 1 000 000 000
600 ns: 1 1 000 000 000
640 ns: 1 1 000 000 000
680 ns: 1 1 000 000 000
720 ns: 1 1 000 000 000
760 ns: 1 1 000 000 000
800 ns: 1 1 000 000 000
840 ns: 1 1 000 000 000

```

**Figura 4.50 – Formato del fichero para la simulación de la interfaz VGA**

Realizando la simulación, introduciendo el fichero de estímulos en el ADC, como se indicó en el apartado [4.12](#), con una señal de 250 mV de amplitud en el canal 1 y una de 500 mV en el 2, se obtiene el siguiente resultado:



**Figura 4.51 – Figura generada por el VGA simulator**

Como se puede ver en la figura, tanto el marco del *plot* como la rejilla se representan correctamente y las señales se representan con la amplitud que deben tener, teniendo en cuenta que, como se puede ver arriba, ambos canales están configurados con 500 mV/div. En cuanto a los caracteres y símbolos todos se representan correctamente, con un buen tamaño que

permite que se entiendan sin ningún tipo de problema.

Por último, el cálculo de las magnitudes se realiza de manera bastante precisa. Como se puede ver en la figura 4.52 si se le introduce un offset a la señal del canal 2, la tensión media aumenta.

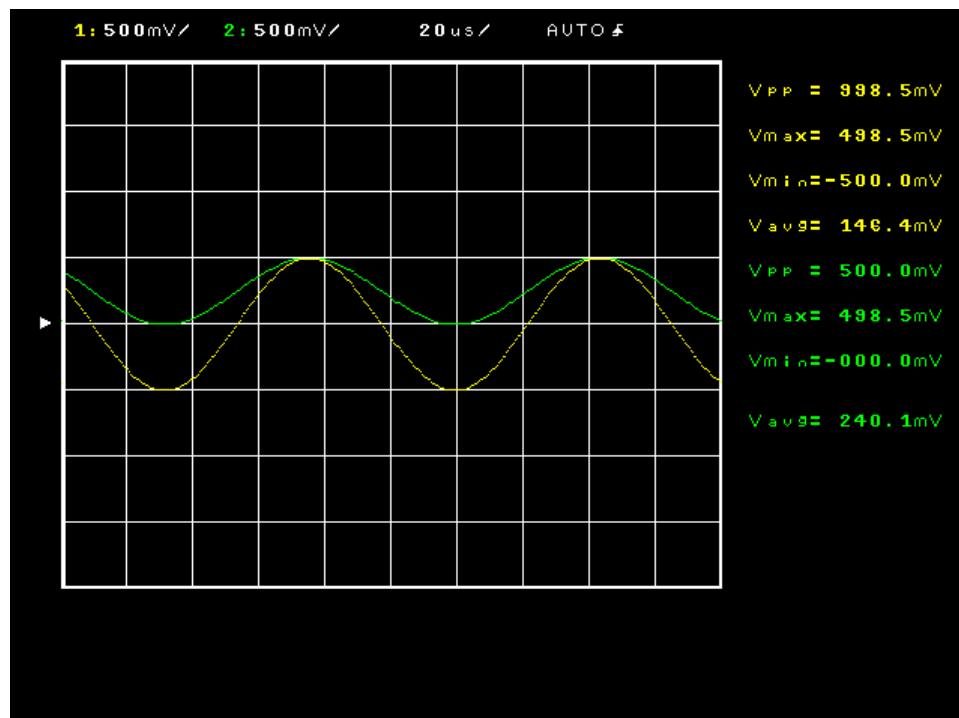
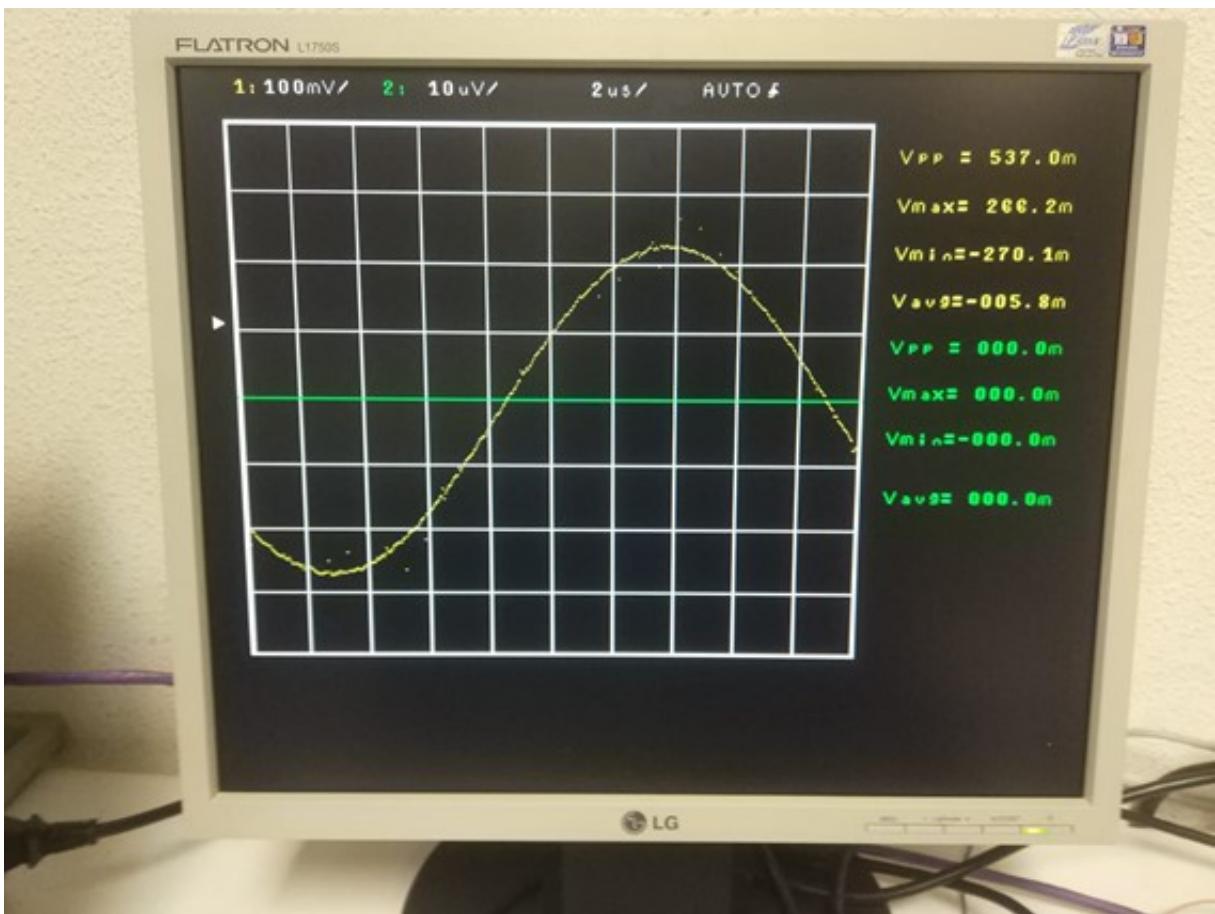


Figura 4.52 – Figura generada por el VGA simulator 2



## 5 RESULTADOS

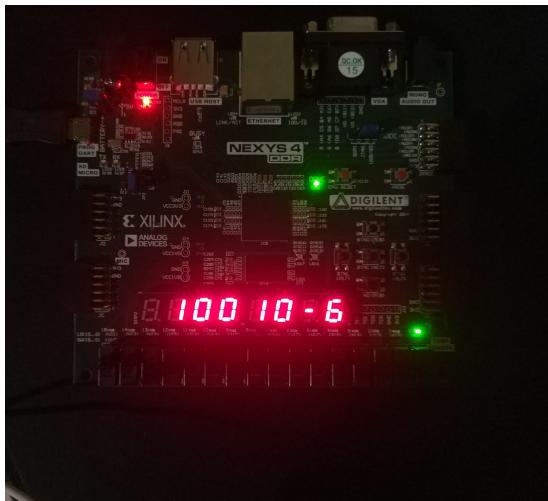
Una vez diseñado y simulado todo el proyecto, llega la hora de implementarlo en la placa. Para ello, se genera el *bitstream* con el que se programa la FPGA. Esto se hace empleando el código recogido en el repositorio <https://github.com/martinnp12/ProyectoOsciloscopio>. Una vez programada es necesario introducir una señal en el canal 2 del puerto PMOD, que se corresponde con el canal 1 del osciloscopio, presente en la placa. Para ello se emplea un generador de funciones con una onda sinusoidal. Para ver la interfaz gráfica se emplea un monitor VGA.



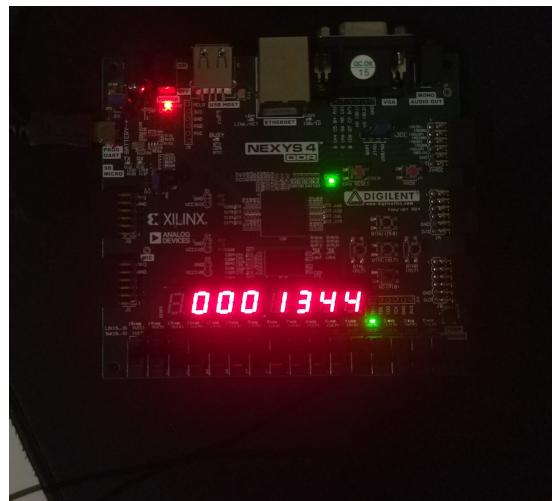
**Figura 5.1 – Resultado de la interfaz gráfica**

En la figura 5.1 se puede ver como la interfaz se genera exactamente igual que en la simulación. Sin embargo en este caso, se puede comprobar como el centro de la gráfica se corresponde con el punto donde se produce el *trigger* con flanco ascendente. Sin embargo, en la señal de entrada se produce un cierto ruido, que provoca pequeñas perturbaciones en el cálculo de las diferentes magnitudes.

Por otro lado, tenemos la representación de los valores de VPD y TPD en la Nexys 4 DDR. Como se puede apreciar en las figuras 5.2 y 5.3, tanto la representación en notación científica, para el VPD y TPD, como la representación en BCD del *trigger* se realizan de manera correcta.

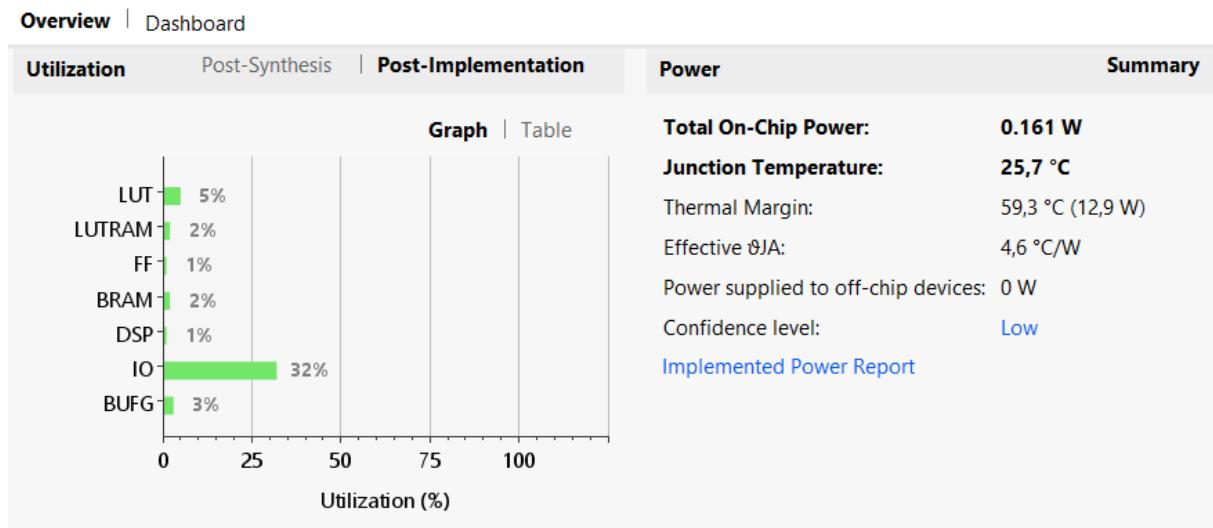


**Figura 5.2 – Representación VPD en la placa**



**Figura 5.3 – Representación trigger en la placa**

Finalmente, analizamos las estadística de uso y de consumo de nuestro sistema, recogidos en la figura 5.4. Como se puede apreciar, el numero de unidades funcionales utilizadas es muy reducido, siendo únicamente las entradas y salidas los elementos con un uso superior al 5%. En cuanto al consumo, este se estima en 161 mW. Sin embargo, es necesario tener en cuenta que este consumo es únicamente el de la FPGA y habría que sumarle el consumo del monitor que se utilice.



**Figura 5.4 – Utilización y consumo de la FPGA**

## 6 CONCLUSIONES Y LÍNEAS FUTURAS

### 6.1. Conclusiones

Los principales objetivos de este proyecto eran el diseño de un osciloscopio compuesto de módulos genéricos, diseñados para ser fácilmente configurables, y que realizase el cálculo de magnitudes básicas. Estos objetivos se han cumplido, consiguiendo un osciloscopio con funcionalidades básicas. Sin embargo, no se han podido incorporar funciones que habrían sido muy interesantes como el cálculo de la frecuencia o la interpolación, por falta de tiempo. A pesar de ello, queda claro que las FPGAs son dispositivos muy versátiles con los que es perfectamente viable desarrollar un osciloscopio con las funcionalidades necesarias para trabajar con sistemas electrónicos y con un coste y consumo energético reducidos.

### 6.2. Líneas futuras

A pesar de que en este proyecto se ha conseguir un osciloscopio funcional, hay ciertas funcionalidades que mejorarían mucho el proyecto:

- Interpolación: interpolar las señales haría que el uso del osciloscopio sea mucho más cómodo para el usuario, al poder ver las señales representadas por líneas continuas y no por puntos. Además, la interpolación permite aumentar el ancho de banda del osciloscopio.
- Calculo de la frecuencia de las señales: conocer la frecuencia de la señal es un cálculo más complejo que los que se han realizado en este proyecto al ser necesario realizar divisiones. Sin embargo, sería muy interesante poder disponer de este cálculo.
- Transformada de Fourier: esta operación permite descomponer la señal en la suma de senos que la conforman, mejorando mucho el análisis del ruido en determinadas señales.



## 7 Bibliografía

- [1] Digilent. *Nexys4 DDR™ FPGA Board Reference Manual*, Abril de 2016.
- [2] Xilinx. *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter. User Guide*, Julio de 2018.
- [3] Tektronix. *XYZs of Oscilloscopes*, 2016.
- [4] Andrei Purcarus, Ze Yu Yang. *An FPGA Implementation of a Digital Storage Oscilloscope*. McGill University, 2017.



# **ANEXOS**

MARTÍN NOVOA PELLO

JULIO 2021



## A Mapas de memoria

En este anexo se recogen las direcciones de memoria asociadas a cada uno de los caracteres y símbolos.

### A.1. Mapa de la memoria de caracteres

Las dimensiones de los caracteres son de 8 x 8 píxeles. Como es necesario almacenar 32 caracteres, se crea una memoria de 256 posiciones de memoria de 1 byte cada una y se recoge en la tabla A.1.

**Tabla A.1 – Mapa de la memoria de caracteres**

Número	Posición	Contenido
0	0 - 7	Filas del 0
1	8 - 15	Filas del 1
2	16 - 23	Filas del 2
3	24 - 31	Filas del 3
4	32 - 39	Filas del 4
5	40 - 47	Filas del 5
6	48 - 55	Filas del 6
7	56 - 63	Filas del 7
8	64 - 71	Filas del 8
9	72 - 79	Filas del 9
10	80 - 87	Filas del punto
11	88 - 95	Filas dos puntos
12	96 - 103	Filas del igual
13	104 - 111	Filas de la u
14	112 - 119	Filas de la m
15	120 - 127	Filas de la V
16	128 - 135	Filas de la s
17	136 - 143	Filas del slash
18	144 - 151	Filas de la A
19	152 - 159	Filas de la U
20	160 - 167	Filas de la T
21	168 - 175	Filas de la O
22	176 - 183	Filas del espacio
23	184 - 191	Filas de la x
24	192 - 199	Filas de la i
25	200 - 207	Filas de la n
26	208 - 215	Filas del -
27	216 - 223	Filas de la p
28	224 - 231	Filas de la a
29	232 - 239	Filas de la v
30	240 - 247	Filas de la g
31	248 - 255	Filas de la k

## A.2. Mapa de la memoria de símbolos

Las dimensiones de los símbolos son de 9 x 9 píxeles. Como es necesario almacenar 3 caracteres, se crea una memoria de 27 posiciones de memoria de 9 bits cada una y se recoge en la tabla A.2.

**Tabla A.2 – Mapa de la memoria de símbolos**

Número	Posición	Contenido
0	0 - 8	Filas del flanco ascendente
1	9 - 18	Filas del flanco descendente
2	19 - 26	Filas de la flecha del trigger

## B Presupuesto

En este apéndice se expone el coste del desarrollo del proyecto en el que se tendrá en cuenta los recursos necesarios, tanto *hardware* como *software*, la mano de obra y los impuestos asociados.

### B.1. Recursos amortizables

En la tabla B.1 se recoge el coste asociado al proyecto de todos los recursos que han sido necesarios durante la realización del mismo.

	Dispositivo	Precio (€)	Cantidad	Vida útil (meses)	Tiempo de uso (meses)	Coste (€)
Hardware	Ordenador portátil	1292	1	48	5	134.58
Software	Vivado	2995	1	12	5	1247
	Microsoft Office	69	1	12	5	28.75
						<b>TOTAL:</b> 1410.33

Tabla B.1 – Coste de los recursos amortizables

### B.2. Recursos no amortizables

En la tabla B.2 se recogen el coste de aquellos dispositivos en los que se implementa el proyecto y que se considera que serán utilizados durante toda su vida útil.

Dispositivo	Cantidad	Coste unitario (€)	Coste total (€)
Nexys 4 DDR	1	224.46	224.46
Monitor VGA	1	80	80
Cable VGA	1	0.51	0.51
			<b>TOTAL:</b> 304.97

Tabla B.2 – Coste de los recursos amortizables

### B.3. Mano de obra

En la tabla B.3 se recoge coste de la mano de obra para la realización de este proyecto por un graduado en Ingeniería Electrónica Industrial y Automática.

Tipo de mano de obra	Unidades (h)	Honorarios (€/h)	Total (€)
Graduado en Ingeniería Electrónica Industrial y Automática	450	40	18 000

**Tabla B.3 – Coste de los recursos amortizables**

## B.4. Coste total

Finalmente, se recoge en la tabla B.4 el coste total del proyecto, incluyendo los impuestos asociados.

Concepto	Coste(€)
Recursos amortizables	1 410.33
Recursos no amortizables	304.97
Mano de obra	18 000
<b>SUBTOTAL:</b>	19 715.3
IVA (21 %)	4 140.21
<b>TOTAL:</b>	23 855.5

**Tabla B.4 – Coste de los recursos amortizables**

## C Impacto Ético, Social, Económico y Medioambiental

En este anexo se trata el posible impacto que puede tener este proyecto en diferentes campos.

### C.1. Impacto Ético

Este es un proyecto en el que se desarrolla un osciloscopio de *hardware* libre. El desarrollo tanto de *hardware* como de *software* libre hace posible que ciertas tecnologías lleguen a lugares en los que no se la podrían permitir de otro modo, debido a sus bajos recursos. Esto hace que en estos lugares se facilite la formación en un gran abanico de campos, no solo en el de la tecnología. Dispositivos como la Raspberry Pi, que consiste en un ordenador de *hardware* y *software* libre permiten acceder a una gran cantidad de información a un coste mucho más reducido.

### C.2. Impacto Económico

Como se ha comentado en el Impacto Ético, el desarrollo de tecnología libre fomenta el acceso a aquellas personas que no se pueden permitir de otro modo. Esto hace que aquellas personas se puedan formar en determinados campos, como puede ser el de la ingeniería, acelerando el crecimiento económico de países en vías de desarrollo.

### C.3. Impacto Social

En el ámbito social este tipo de tecnologías también tiene un peso relevante. Gracias al acceso al conocimiento a un coste mucho más reducido se puede favorecer la alfabetización, consiguiendo eliminar en cierta medida las desigualdades sociales que ocasiona esto. Además, gracias a la aceleración del crecimiento económico, comentada en el Impacto Económico, se fomentaría también un aumento de la calidad de vida de las personas y su bienestar en general.

### C.4. Impacto Medioambiental

El impacto medioambiental de los circuitos electrónicos es bastante elevado. La producción de componentes electrónicos supone un gran impacto en el terreno, al tener que realizar minería para obtener ciertos minerales necesarios para la fabricación. Este impacto es inherente a todo dispositivo electrónico. Por otra parte, es necesario tener en cuenta el impacto medioambiental debido al consumo energético. Contribuyendo a efectos como el calentamiento global, debido a la generación de energía eléctrica mediante combustibles fósiles, o a la alteración del entorno, con la generación de energías renovables, como puede ser la eólica.

Sin embargo, la utilización de dispositivos como FPGAs permite reducir el consumo energético y consecuentemente el impacto ambiental debido a este.