

# **MICROELECTRÓNICA**

**GB-17**



## **INFORME PROYECTO**

### **"TRADUCTOR DE LENGUAJE DE SEÑAS EN FPGA"**

 **GRUPO 03**

#### **ALUMNOS :**

- Mamani Huanca Jhoel René
- Mendoza Cari Iván Marcos
- Vilcapaza Goyzueta Denilson
- Yanapa Ccoarite Rony Miguel

#### **DOCENTE:**

PhD: Alexander B. Hilario Tacuri



*Agosto del 2020*

# Contenido

---

<b>1. Introducción</b>	<b>1</b>
1.1. Resumen . . . . .	1
1.2. Objetivos . . . . .	2
1.3. ¿Por qué este proyecto? . . . . .	2
<b>2. Marco teórico - referencial</b>	<b>3</b>
2.1. Sensores Flex . . . . .	3
2.2. Conversor de señal analógica a digital . . . . .	5
2.2.1. Convertidor ADC . . . . .	6
2.3. Comunicación I2C . . . . .	6
2.3.1. Desarrollo I2C . . . . .	6
2.3.2. Descripción de puertos . . . . .	7
<b>3. Diseño y análisis de diseño</b>	<b>8</b>
3.1. Enfoque del proyecto - sumario . . . . .	9
3.2. Diseño Traductor de lenguaje de señas . . . . .	10
3.3. Selección del sensor a utilizar . . . . .	11
3.4. Acondicionamiento de la señal del sensor flexible . . . . .	12
3.5. Generación de señales analógicas en Vivado . . . . .	14
3.6. Interpretación ADC . . . . .	14
3.7. Interpretación I2C . . . . .	17
3.8. Módulo - Mapeo (Señales y letras) . . . . .	18
3.8.1. Explicación del código . . . . .	18
3.9. Modulo - Controlador VGA . . . . .	23
3.9.1. Características generales VGA . . . . .	24
3.9.2. Los valores de la pantalla de VGA . . . . .	25
3.10. Módulo de sincronización de VGA . . . . .	26
<b>4. Implementación de diseño</b>	<b>30</b>
4.1. Productos y materiales a utilizar . . . . .	30
<b>5. Resultados</b>	<b>32</b>
<b>6. Conclusiones y Recomendaciones</b>	<b>32</b>

# Traductor de lenguaje de señas en FPGA

## Proyecto final

### 1. Introducción

---

En nuestro medio las personas privadas de hablar y escuchar aprenden la comunicación manual (mímica dactilológica) y/o la lectura labio facial para poder comunicarse con la sociedad, sin embargo la gran mayoría de personas con la capacidad de hablar no entienden el lenguaje que tales personas utilizan para comunicarse con los demás.

Es muy común que las personas con esta discapacidad sean de bajos recursos económicos, debido a su dificultad de comunicación, la que no les permite acceder a un trabajo bien remunerado, ya que toda actividad exige de una buena comunicación para poder realizar las distintas tareas que se asignen.

El presente trabajo muestra el diseño y construcción de un guante prototipo que capture el movimiento de los dedos de la mano de una persona sordomuda para luego traducir el lenguaje de señas sordomudo al lenguaje alfabético y con su respectiva traducción auditiva, debido a que existen personas que no son totalmente sordas y les sirve de mucho el relacionar la letra o palabra con el audio proporcionado.

#### 1.1 Resumen

Hoy en día, con el desarrollo de la tecnología, se han logrado grandes avances en tecnología médica. Como resultado, se desarrollan varias soluciones tecnológicas para personas con discapacidad. Este proyecto tiene como objetivo facilitar la comunicación entre las personas con discapacidad auditiva y las personas sanas.

Aprender un lenguaje de señas es un proceso exigente. Por lo tanto, este proceso no es bien conocido por muchas personas sanas. Con la ayuda de este guante inteligente, esas personas sanas podrán comprender a quienes usan el lenguaje de señas. El lenguaje en este proyecto se basa en ASL. El guante inteligente también se puede usar como un simulador para personas sanas que desean aprender el lenguaje de señas.

En conclusión, el proyecto tiene como objetivo escribir los gestos con las manos realizados con guantes inteligentes en el monitor como letras.

## 1.2 Objetivos

El propósito de este proyecto es construir un guante inteligente que traduzca una letra de gesto de la mano usando un FPGA. La letra traducida se realizará en un monitor VGA después de que las señales provenientes del guante sean convertidas por FPGA. En el monitor solo habrá un carácter. Dado que se interpreta un personaje, no será difícil interpretar todas las palabras u oraciones. Sin embargo, para este proyecto actual, solo se realiza un solo carácter en el medio de la pantalla para mostrar claramente el resultado y mantener el código comprensible. También se realizan algunas de las 26 letras. Para estas algunas letras, el guante inteligente diseñado no es capaz de realizarlas, ya que este no es el propósito principal. También se pretende descubrir cómo se utiliza un protocolo de comunicación en una FPGA utilizando VHDL. Para este proyecto se utiliza el protocolo de comunicación I2C con PMOD AD2 (en este proyecto el fpga adc) para poder comunicarse entre sensores y FPGA. En este proyecto se utiliza Artix 7 FPGA. El ADC tiene 4 canales y una resolución de 12 bits que es suficiente para definir esas letras.

## 1.3 ¿Por qué este proyecto?

El proyecto se diseña para resolver problemas de comunicación a personas que usan el lenguaje de señas. Avanzar con el desarrollo de la comunicación que es una barrera para estas personas, así permitirá comunicarse fácilmente en la vida diaria y además puede ser utilizado para aprender el lenguaje de señas.

Al comienzo del proyecto, la idea era trabajar en FPGA para aprender y descubrir sus comportamientos y especificaciones. Para ello decidimos trabajar en una FPGA y preparar un proyecto para el Concurso de Diseño Digilent. Primero comenzamos con otro FPGA hasta que obtenemos el Artix 7 FPGA, pero no se menciona en esta etapa, ya que aplicamos los mismos pasos en Artix 7 FPGA y también las partes más importantes se implementan en Artix 7 FPGA. Los temas que queríamos estudiar son: ¿Cómo podemos comunicar un sensor con una FPGA? ¿Cómo podemos obtener una salida de la FPGA para hacer visibles las señales y los datos significativos como siete segmentos, impulsando un motor de CC o un monitor, etc.? Queríamos que las entradas y salidas fueran comprensibles para las personas que en realidad no saben nada sobre FPGA o incluso no saben nada sobre electrónica. Con la idea de llegar a estos temas, decidimos construir un sistema usando FPGA para convertir los gestos de las manos en caracteres legibles para poder hacer entender a las personas que no conocen el lenguaje de señas. Esta es la principal motivación para nosotros. Sin embargo, esta idea se desarrolló utilizando microcontroladores. Pero aplicando esta idea en una FPGA, también podremos ver las diferencias y ventajas / desventajas en comparación con un microcontrolador.

## 2. Marco teórico - referencial

### 2.1 Sensores Flex

Los sensores piezoelectrómicos tienen la particularidad que al ser flexionados ocurre un cambio en la resistencia eléctrica entre sus terminales.

Son transductores pasivos, es decir necesitan alguna excitación o polarización para poder convertir un tipo de energía en otra.

En un lado del sensor se imprime con una tinta de polímero que tiene partículas conductoras embebidas en él. Cuando el sensor esta recto, las partículas de la tinta dan una resistencia de aproximadamente  $10K\Omega$ . Cuando el sensor esta doblado lejos de la tinta, las partículas conductoras se encuentran más separadas, aumentando la resistencia (a alrededor de  $40 K\Omega$  ohmios cuando el sensor está doblado a  $90^\circ$ , como en el diagrama de la Figura 1.). Cuando el sensor se endereza de nuevo, la resistencia vuelve al valor original. Mediante la medición de la resistencia, se puede determinar hasta qué punto el sensor está siendo sometido.

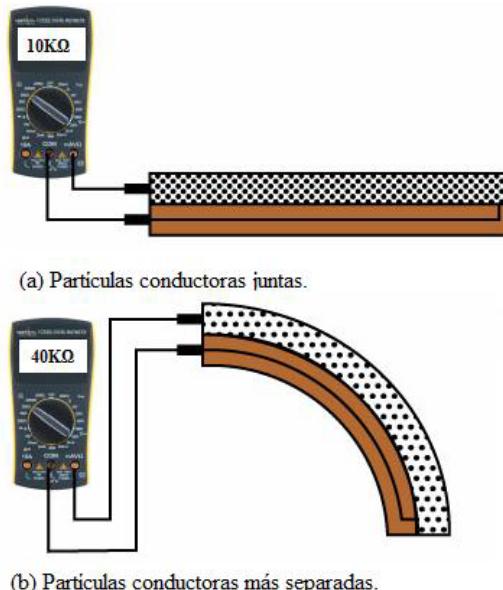


Figura 1: Comportamiento interno del sensor flexible. (a) Partículas conductoras juntas, (b) Partículas conductoras separadas.

Este sensor cambia su resistencia dependiendo de la cantidad de curva que experimenta. Como se dijo anteriormente su variación en curvatura es convertida a resistencia eléctrica, cuanto más es la curva, más es el valor de la resistencia, en la Figura 2 se aprecia las dimensiones de este tipo de sensor.



Figura 2: Dimensiones sensor Flexible

Es importante detallar que la variación de la resistencia de este sensor es en un solo sentido. En la actualidad estos sensores se los encuentra en longitudes de 2.2” y 4.5” (pulgadas), ambos tienen las mismas características, solo varía en sus distancias longitudinales. En la figura 3 se aprecia el rango de variación de ángulos a la cual el sensor flexible reacciona, cuando este se encuentra lineal tiene el mínimo valor de resistencia, cuando se encuentra a 90 grados tiene el máximo valor resistancial.

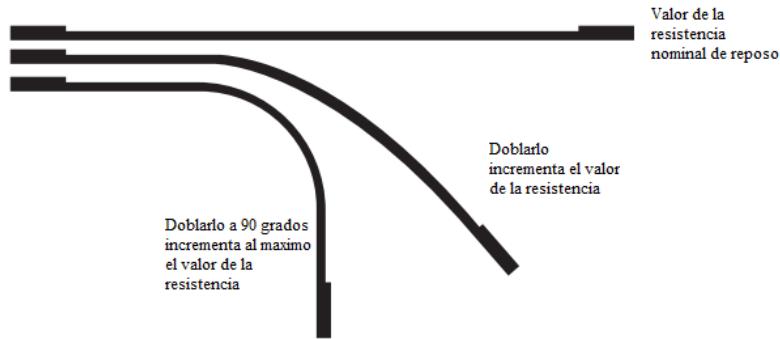


Figura 3: Rango de ángulos de reacción del Flex sensor.

Se puede apreciar el esquema físico del sensor flexible con las medidas indicadas anteriormente. (Figura 4)

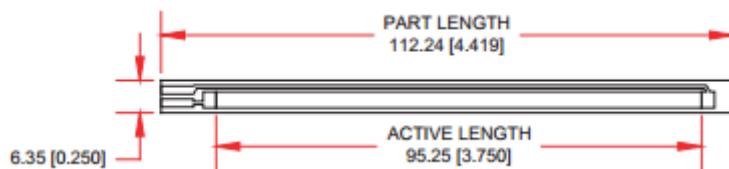


Figura 4: Esquema físico y medidas del sensor flexible de 4.5”.

La construcción de la membrana que conforma este tipo de sensores es como era de esperarse flexible y además algo duradera, puede ser utilizado dentro de un rango de temperatura de  $-35^{\circ}\text{ C}$  hasta  $+80^{\circ}\text{ C}$ , para un nivel de vida operativa de más de 1 millón de movimientos si el sensor está fijado correctamente. En la figura 5 se puede apreciar el esquema físico y las medidas del sensor flexible de 2.2”.

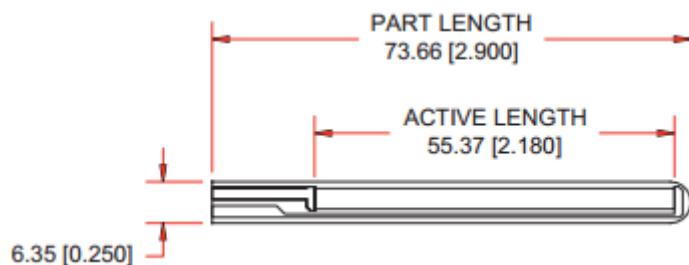


Figura 5: Esquema físico y medidas del sensor flexible de 2.2”.

Existen 3 fabricantes de este tipo de sensores.

- Spectra Symbol Flex sensors.
- Gentile Abrams sensor, disponibles por Jameco Electronics.

- Gizmo Music.

No son más que resistencias. Trabajan como divisores variables de tensión analógicos. En el interior del sensor de flexión tiene elementos resistivos de carbono dentro de un sustrato flexible delgado. Más carbono significa menos resistencia. Cuando el sustrato se dobla el sensor produce una resistencia de salida con respecto al radio de curvatura.

### Características eléctricas:

- **Tamaño:**  
Aproximadamente 0.28" ancho y 1"/3/5" de longitud.
- **Rango de Resistencia:**  
1,5 – 40K $\Omega$  dependiendo del sensor.
- **Tiempo de vida:**  
Más de aproximadamente 1 millón de usos.
- **Rango de temperatura:**  
-35° C hasta +80° C.
- **Histéresis:**  
7%
- **Voltagje:**  
5 a 12 V.

## 2.2 Conversor de señal analógica a digital

Un conversor o convertidor de señal analógica a digital (Conversor Analógico Digital, CAD; Analog-to-Digital Converter, ADC) es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos, en este caso, cada valor numérico binario hace corresponder a un solo valor de tensión o corriente.

En la cuantificación de la señal se produce pérdida de la información que no puede ser recuperada en el proceso inverso, es decir, en la conversión de señal digital a analógica y esto es debido a que se truncan los valores entre 2 niveles de cuantificación, mientras mayor cantidad de bits mayor resolución y por lo tanto menor información perdida.

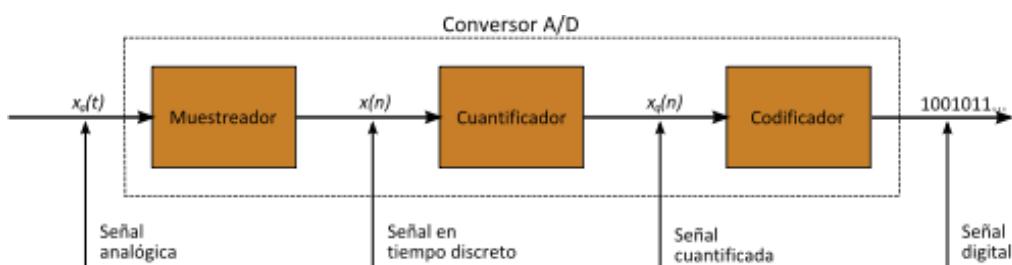


Figura 6: Procesos de la conversión A/D

### 2.2.1 Convertidor ADC

Las FPGA tienen muchas características potentes que incluyen un convertidor analógico a digital (ADC) integrado llamado "XADC". Es un ADC dual de 12 bits y 1 mega muestra por segundo (MSPS) que se utiliza para acomodar el muestreo de hasta 17 señales auxiliares, así como también incluye sensores en chip para monitoreo de temperatura y potencia.

El XADC admite una amplia gama de modos de funcionamiento, como un solo canal, canal secuenciado, secuencia simultánea de duelo ADC o muestreo de canal de secuencia de una pasada; operación de muestreo dirigida por eventos o continua; muestreo de señal unipolar, diferencial o bipolar; etc ... El XADC también se puede configurar para apagar sus ADC individuales, reorganizar el orden de muestreo cuando se ejecuta en una operación de canal secuenciado, permitir el promedio de muestras para reducir el ruido, establecer alarmas para violaciones de tolerancia de temperatura y voltaje definidas por el usuario para el chip sensores, etc.

La imagen a continuación muestra el diagrama de bloques del XADC que se encuentra en la hoja de datos del dispositivo tal como lo proporciona Xilinx. .

## 2.3 Comunicación I2C

El bus de comunicaciones I2C es un protocolo que se efectúa por medio de DOS hilos. A través de estos dos hilos pueden conectarse diferentes dispositivos donde algunos de ellos serán MAESTROS en cuanto muchos otros dispositivos serán ESCLAVOS.

### 2.3.1 Desarrollo I2C

Para poder reconocer cada uno de los dispositivos conectados a los DOS hilos del bus I2C, a cada dispositivo se le asigna una dirección.

Así en este tipo de comunicaciones el MAESTRO es el que tiene la iniciativa en la transferencia y este es quien decide con quien se quiere conectar para enviar y recibir datos y también decide cuando finalizar la comunicación.

Los DOS hilos del BUS interfaz de comunicación I2C son líneas de colector abierto donde una de las líneas lleva la señal de reloj y es conocida como (SCL), y la otra linea lleva los datos y es conocida como (SDA).

Los Pines SDA y SCL I2C se encuentran especificados en todos los componentes que usan este tipo de protocolo de comunicación.

Para que la comunicación funcione se deben utilizar unas resistencias PULL UP (resistencias conectadas a positivo) para asegurar un nivel alto cuando NO hay dispositivos conectados al BUS I2C.

La conexión I2C entre un maestro y varios esclavos se muestra a continuación:

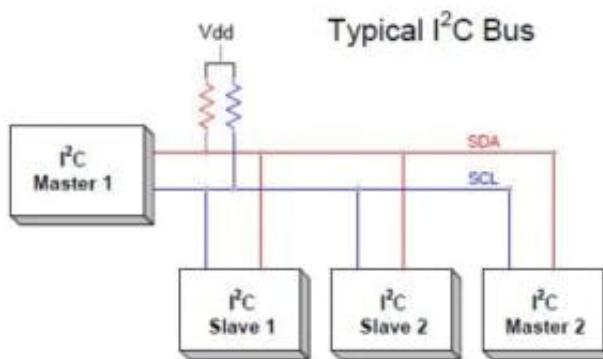


Figura 7: Maestro y varios esclavos configuración

### 2.3.2 Descripción de puertos

Para este trabajo, el componente lee y escribe en la lógica del usuario a través de una interfaz paralela. Fue diseñado usando 2020.1.

Los requisitos de recursos dependen de la implementación. La Figura 9 ilustra un ejemplo típico del maestro I2C integrado en un sistema.

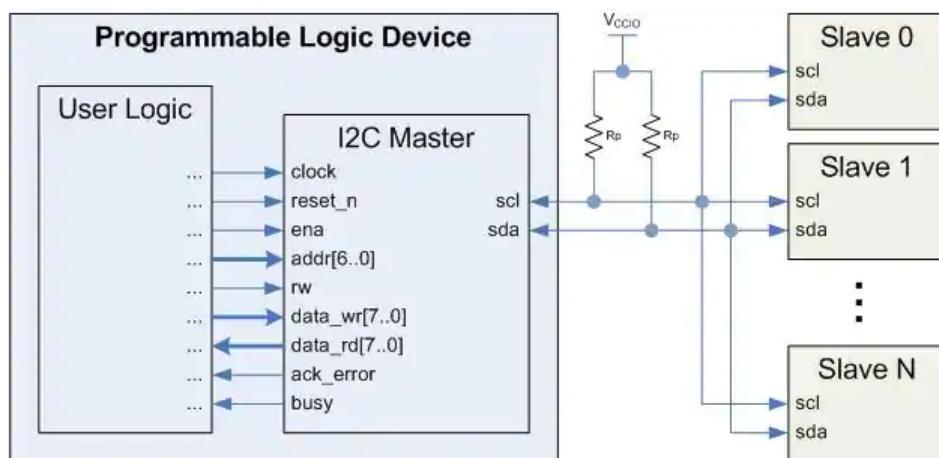


Figura 8: Implementación de ejemplo

La siguiente tabla describe los puertos del maestro I2C.

Puerto	Anchura	Modo	Tipo de datos	Interfaz
clk	1	in	standard logic	user logic
reset_n	1	in	standard logic	user logic
ena	1	in	standard logic	user logic
addr	7	in	standard logic vector	user logic
rw	1	in	standard logic	user logic
data_wr	8	in	standard logic vector	user logic
data_rd	8	out	standard logic vector	user logic
busy	1	out	standard logic	user logic
ack_error	1	buffer	standard logic	user logic
sda	1	inout	standard logic	slave device(s)
scl	1	inout	standard logic	slave device(s)

*Puerto clk.*

Reloj del sistema.

*Puerto reset.*

Restablecimiento bajo activo asíncrono.

*Puerto ena.*

0: no se inicia ninguna transacción.

1: se bloquea en addr, rw y data\_wr para iniciar una transacción. Si ena es alta al final de una transacción (es decir, cuando el nivel de ocupación baja), entonces una nueva dirección, comando de lectura / escritura y datos se bloquean para continuar la transacción.

*Puerto addr.*

Dirección del esclavo objetivo.

*Puerto rw.*

0: comando de escritura.

1: comando de lectura.

*Puerto data\_wr*

Datos a transmitir si rw = 0 (escritura).

*Puerto data\_rd*

Datos recibidos si rw = 1 (lectura).

*Puerto busy.*

0: el maestro I2C está inactivo y los últimos datos leídos están disponibles en data\_rd.

1: el comando se ha bloqueado y la transacción está en curso.

*Puerto ack.*

0: sin reconocer errores.

1: se produjo al menos un error de reconocimiento durante la transacción. ack\_error se borra al principio de cada transacción.

*Puerto sda.*

Línea de datos serie del bus I2C.

*Puerto scl.*

Línea de reloj serie del bus I2C.

---

### 3. Diseño y análisis de diseño

En esta parte del trabajo se explica los procedimientos, parámetros y pruebas realizadas para la construcción del guante traductor así como también su diseño final con las partes que lo componen, también las diferentes mediciones de los sensores flexibles para análisis y propuestas de mejora en la colocación de los mismos según el símbolo que presenta más movimiento para la realización de las señas, y sus circuitos de acondicionamiento.

Antes de realizar el diseño de cada parte y proceso del proyecto, es necesario tener una referencia de descripción estructural - funcional de este trabajo. A continuación se muestra el resumen del proyecto.

### 3.1 Enfoque del proyecto - sumario

Los principios de funcionamiento de los proyectos se pueden dividir en partes simplemente así:

- 1) Un guante único que tiene 4 sensores Flex. Excepto el dedo meñique, otros 4 dedos tienen sensores. Los sensores detectan qué letras están funcionando y luego envían una señal analógica. Estas señales serán simuladas en software Vivado Design Suite.
- 2) Vivado toma estas señales analógicas y luego se procesa para asimilar tendencias de comportamientos en los sensores; así, se envía estas señales analógicas hacia vhdl FPGA (otro programa). Donde se realizará la conversión analógico digital.
- 3) Se realiza la conversión, para luego simular a un conversor como si fuese PMOD ADC.
- 4) Después de convertir el proceso analógico a digital, se realiza la comparación en FPGA vhdl creando asignaciones para las respuestas.
- 5) El FPGA decide qué letra se realizó y envía señales digitales a sus pines. Estos pines están conectados al cable VGA y, por lo tanto, al monitor. El FPGA tiene un algoritmo que deriva un monitor VGA.
- 6) Por último, la letra realizada se visualiza en el medio de la pantalla.

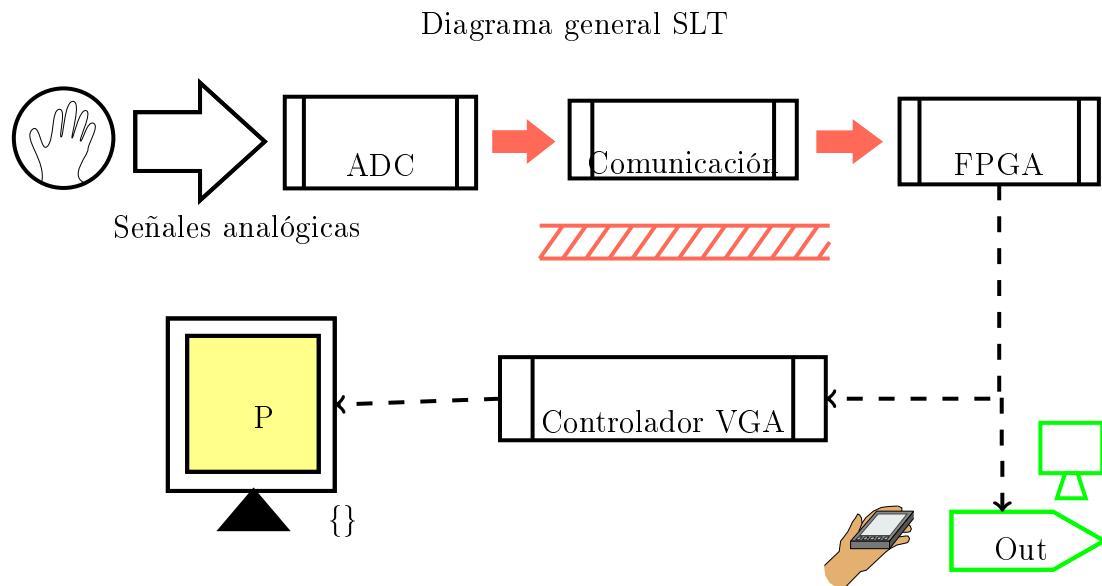
En este proyecto, todos los algoritmos se implementan en VHDL. Las letras del lenguaje de señas son diferentes en cada país. El estándar de lenguaje de señas americano (ASL) ha sido elegido para este proyecto. ASL es también el lenguaje de señas más utilizado en el mundo. Un usuario puede traducir 16 letras de ellas usando este sistema. Las letras se enumeran a continuación en la figura. (a,b,c,d,e,f,g,h,k,l,o,p,s,t,x,y)



Figura 9: Lenguaje de signos

### 3.2 Diseño Traductor de lenguaje de señas

Un diseño general de configuración para este proyecto, se muestra en el diagrama general STL (Sign Language Translator).



A continuación describiremos este diagrama de funcionamiento.

Se realiza el análisis de las señales analógicas en Vivado, realizando generación de señales que asimilarán las señales de sensor flex. Luego, En vhdl se realiza la conversión analógico digital. Entonces se simulará 4 canales que nos permiten conectar 4 sensores (señales analógicas generadas) con una resolución de 12 bits. La comunicación se dà a través de protocolo I2C. El algoritmo se ha escrito con VHDL.

Cada proceso está clasificado para cada gesto, que dará un valor de señales analógica que debería obtener de cada sensor. Después de recopilar todos los datos necesarios, el algoritmo similar se implementa en “el FPGA”.

Para poder hacer que estas señales digitales sean visibles y legibles, se utiliza el monitor VGA. La razón por la que se usa un monitor VGA es que queremos aprender a manejar un monitor VGA / HDMI. Se puede decir que VGA es una tecnología un poco antigua; sin embargo, nos permitiría comprender los conceptos básicos de conducir un monitor, por lo que esta es la razón particular por la que se elige. Un televisor CRT simple funciona con VGA, pero también la mayoría de los monitores LCD usan la misma manera. Luego, el algoritmo de conducción VGA se implementa en “el FPGA”. Un cable VGA necesita 5 señales esenciales: rojo, verde, azul, H-Sync y V-Sync. Usando estas señales, cualquier personaje ASCII o personajes de diseño propio, dibujos.

Finalmente, en el monitor, cada 16 letras se pueden realizar mediante gestos con las manos por separado.

#### Recepción señales sensores flex.

El comienzo del diseño comienza con la decisión de qué tipo de lenguaje de señas se elegirá. En este proyecto, se usa ASL como se puede ver en la figura 1. 16 letras de esas 26

letras se usan como mayúsculas: A, B, C, D, E, F, G, H, K, L, O, P, S, T, X e Y.

Los sensores flexibles se colocan en el guante como se puede ver en la figura 4. Cada sensor se coloca en un tubo retráctil para que sean flexibles. Para obtener señales analógicas significativas de los sensores, se utiliza un divisor de voltaje simple. Cuando no está doblado, la resistencia de este sensor es de aproximadamente  $10\text{ k}\Omega$ , y cuando está doblado, su resistencia puede alcanzar hasta  $20\text{k}\Omega$ . Usando el divisor de voltaje en la figura 5, el rango de salida de voltaje analógico está entre 0.75 V y 1.25 V.



Figura 10: Guante con sensores flex acoplados

### 3.3 Selección del sensor a utilizar

Para realizar la selección del sensor se ha contrastado las características de los sensores y verificando que cumpla las exigencias que se necesitan, esto es, un dispositivo maleable, de tamaño pequeño para cuidar la estética del proyecto, de fácil manipulación, ya que va destinado a niños, con el objetivo de que puedan mover el prototipo sin dificultad alguna, liviano y con características técnicas estables para evitar descalibraciones cuando se encuentre en funcionamiento. Con las consideraciones anteriores se opta por utilizar el *sensor flexible* (flex sensor) debido a que reúne en amplio rango las características más apropiadas para la elaboración del guante traductor.

#### Justificación del sensor a utilizar

Es un sensor bastante estable con un óptimo funcionamiento de acuerdo a sus características principales de diseño. Los siguientes son los parámetros más necesarios que motivaron a la elección del sensor flexible.

##### *Rangos óhmicos máximos y mínimos*

Necesarios para la lectura de datos en cada dedo, pues se considera útil que el sensor indique un valor de resistencia mínima en una posición extrema (0 grados) y un valor máximo en la otra posición extrema (90 grados).

Considerando que los dedos de la mano de algunas personas al estar totalmente abiertos, tienden a inclinarse, la selección de este sensor se vuelve indispensable, porque al ser doblado el sensor hacia el otro costado (estando totalmente recto) no va a medir un valor menor, se mantendrá el valor mínimo del sensor mas no variará a un menor valor. Con esto se evita confusiones en la programación para la detección de la mano abierta y mano

cerrada.

#### *Flexibilidad*

Estos sensores son sumamente flexibles, por lo que, en el momento de mover la mano con el guante colocado, no se tendrá ninguna dificultad en el movimiento de los dedos.

#### *Precio del sensor*

Sensores con precios bastante accesibles más aun al comprarlos al por mayor e importarlos del extranjero (como se hizo para esta tesis).

#### *Rango de voltajes*

En vista de que se va a trabajar con una comunicación USB (5V) el sensor flexible está apto para desempeñarse en un rango de voltajes entre 5V – 12V, con esto se evita la realización de fuentes externas para alimentación de los sensores, por tanto con el voltaje que se obtiene de la salida del puerto USB del computador se logra alimentar todo el circuito incluyendo los sensores.

#### *Usos*

La mayor utilidad que se le da a este sensor es en los guantes para detectar el movimiento del dedo, para controlar automóviles, equipos de gimnasio, aparatos de medición, tecnología de asistencia, instrumentos musicales, Smartphones, joysticks, juguetes que detectan diferentes grados de flexión, robots, control de máquinas, dispositivos médicos, pues posee además una alta estabilidad entre otros.

De acuerdo a todas las características técnicas y físicas encontradas en este sensor, las mismas que satisfacen muchos de los intereses solicitados, se convierte en elemento clave para continuar con el desarrollo de esta tesis de manera positiva.

### **3.4 Acondicionamiento de la señal del sensor flexible**

El fabricante recomienda trabajar para las diferentes aplicaciones del Flex Sensor con las configuraciones de circuitos que son: Circuito comparador para accionamiento, Amplificador inversor.

Se ha tomado la configuración divisor de voltaje con el fin de obtener una variación exacta para cada sensor. Es decir, se necesita los mismos valores de voltaje para todos los sensores, debido a que se va a trabajar con los voltajes de referencia que proporciona el Artix 7 FPGA.

A continuación se presenta la formulación necesaria para este tipo de método de acondicionamiento.

#### *Desarrollo*

Una vez seleccionado el método adecuado (Figura 12) para el correcto funcionamiento del circuito, se procede a realizar los cálculos respectivos de las resistencias, de esta forma se obtiene a la salida de cada divisor, el voltaje deseado para posteriormente transmitir esa señal al Artix 7 FPGA.

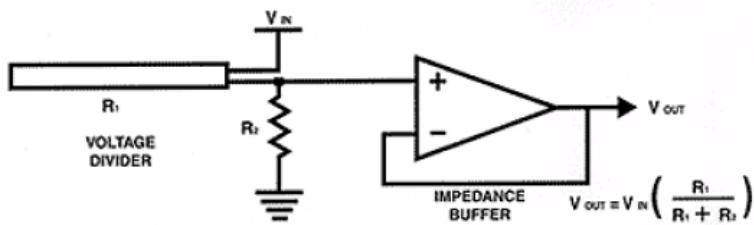
**BASIC FLEX SENSOR CIRCUIT:**

Figura 11: Seguidor Tensión.

La fórmula siguiente es la que permite realizar el cálculo de la resistencia que se necesita para obtener en la salida del amplificador el voltaje adecuado:

$$V_{out} = V_{in} * \frac{R_1}{R_1 + R_2}$$

Con esta fórmula se puede obtener los valores de resistencias imponiéndose el voltaje de salida deseado ( $V_{out}$ ) máximo y mínimo que se quiere obtener en la salida del divisor de tensión.

Es importante recalcar en esta parte que los valores dados en la teoría no son los mismos que en la práctica, es decir, el valor mínimo que nos da el sensor flexible es de  $8K\Omega$ , mientras cuando se lo flexiona al ángulo que necesitamos, nos da un máximo total de  $18K\Omega$ , en sensor de  $4.5''$  pero en sensor de  $2.2''$  el valor promedio es  $20K\Omega$  como valor mínimo sin flexionar y  $30K\Omega$  en su valor máximo, cuando se lo flexiona [5].

**Mediciones.-** Se realizaron medidas de los sensores flexibles de  $4.5''$  con el guante. Los valores de resistencias de los sensores para este guante fueron las que se aprecian en la tabla.

Guante	Valor mínimo ( $K\Omega$ )	Valor máximo( $K\Omega$ )
Dedo pulgar	8.6	12.8
Dedo índice	7.8	17
Dedo medio	6.66	9.5
Dedo anular	6.6	12.3

Calculando la resistencia para cada sensor a través del método de divisor de voltaje (Ecuación última) se obtiene los siguientes valores de resistencias para cada sensor, siendo  $R_1$  el valor de resistencia a encontrar y considerando que el voltaje mínimo es de  $2V$  se obtiene lo siguiente.

Valor de Resistencia	( $K\Omega$ )
R1.pulgar	12,6
R1.índice	12,225
R1.medio	10,125
R1.anular	10,125

Posteriormente se hacen las medidas con el guante, en donde se obtienen los valores reales de voltajes mínimos y máximos cuando se abre y cierra la mano respectivamente, además se observa la diferencia que da como resultado de la resta de los voltajes máximos y mínimos más esta diferencia de voltajes convertida a un valor analógico a digital(ADC). **Valores máximos y mínimos de voltaje de los sensores de 4.5”.**

Valores medidos con el guante	V. minimo (V)	V. máximo(V)	Dif. voltajes
Dedo pulgar	2.5	3.45	0.95
Dedo índice	2.48	3.76	1.28
Dedo medio	2.3	3.2	0.9
Dedo anular	2.3	3.5	1.2

### 3.5 Generación de señales analógicas en Vivado

Se generan 4 señales para los 4 canales analógicos de entrada.

*El código vhdl se encuentra junto al ADC (ver ADC\_12bit)*

La simulación de estos canales en diagrama de tiempos es el siguiente:

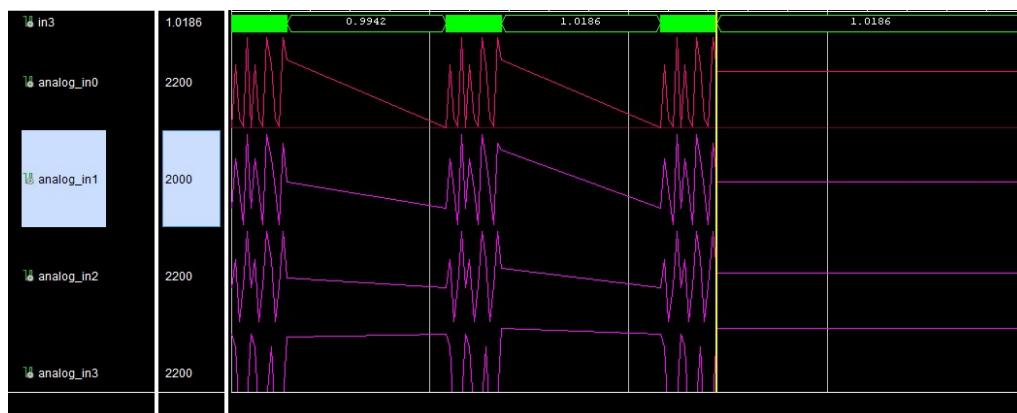


Figura 12: Generación de señales analógicas

### 3.6 Interpretación ADC

Para este trabajo, se implementa un ADC para las 4 señales analógicas que recibe el FPGA.

Estas señales analógicas varían de respuesta para la obtención de caracteres. Debemos de convertir la señal analógica en señal digital, se convierte a señal digital 12bits.

**Nota:** En este trabajo se diseña una asimilación de funcionamiento, para la implemetación podríamos conectar las entradas simuladas en el sensor FLEX (configuración ya visto en la parte teórica).

También, otra alternativa es utilizar PMOD ADC; el cual nos serviría para la conversión analógica y también se puede integrar la comunicación I2C.

La nota anterior describe el diagrama siguiente:

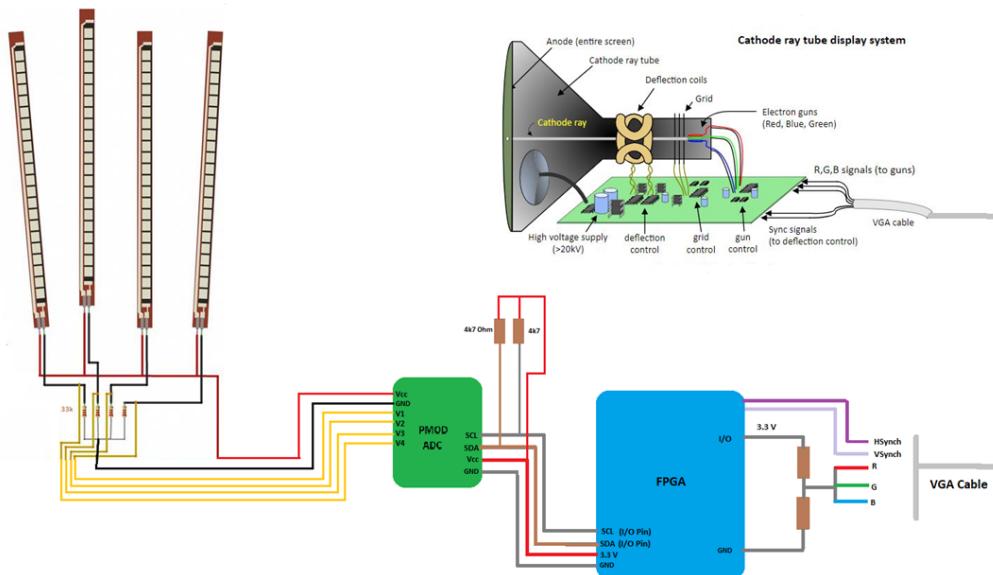
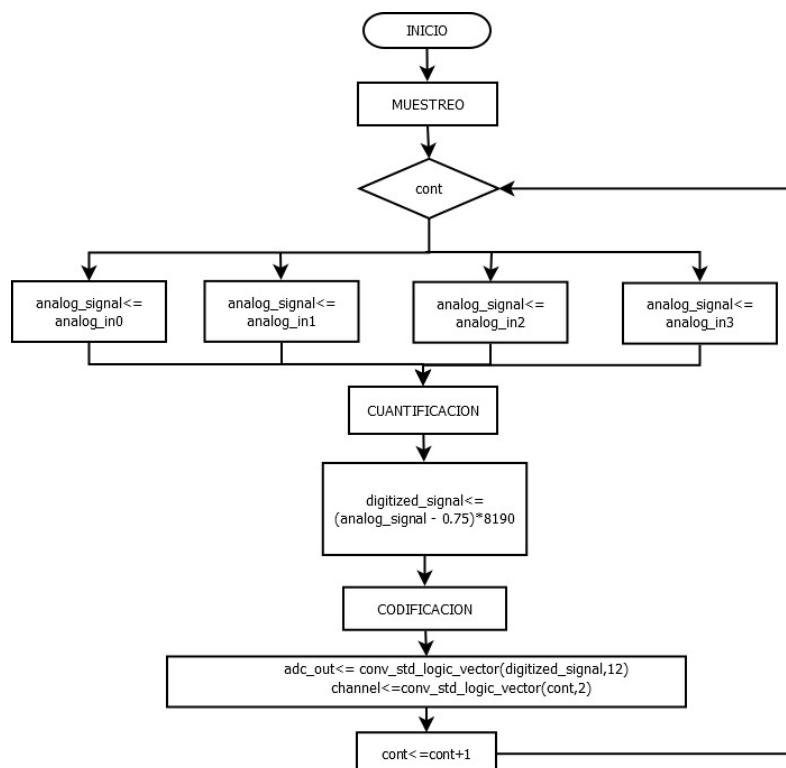


Figura 13: Diagrama con PMOD ADC

Explicado el caso, nosotros utilizamos el diagrama general SLT.

Tenemos que implementar un ADC de 12 bits para 4 canales digitales en el FPGA; entonces se realiza la descripción en VHDL del programa.

#### *Diagrama ADC*



**ADC\_12bit.vhd**

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;
USE ieee.std_logic_arith.all;

entity ADC_12_bit is
  port (analog_in0,analog_in1,analog_in2,analog_in3 : in real range 0.75 to 1.25;
        analog_integer0,analog_integer1,analog_integer2,analog_integer3: out integer
        range 0 to 4095;
        clk:in std_logic;
        digital_out : out std_logic_vector(11 downto 0);
        channel: out std_logic_vector(1 downto 0)
      );
end entity;
architecture Behavioral of ADC_12_bit is

  constant conversion_time: time := 25 ns;
  signal instantly_digitized_signal : integer range 0 to 4095;
  signal analog_signal0, analog_signal1, analog_signal2, analog_signal3: real range 0.75
    to 1.25;
  signal digitized_signal0,digitized_signal1,digitized_signal2,digitized_signal3: integer
    range 0 to 4095;
  signal adc_out: std_logic_vector(11 downto 0);
  CONSTANT max: INTEGER := 3;
  SIGNAL CONT : INTEGER RANGE 0 TO max:=0;
begin
analog_signal0 <= analog_in0;
analog_signal1 <= analog_in1;
analog_signal2 <= analog_in2;
analog_signal3 <= analog_in3;

process(analog_signal0, analog_signal1, analog_signal2, analog_signal3)
begin
  digitized_signal0 <= integer((analog_signal0 -0.75) * 8190.0);
  digitized_signal1 <= integer((analog_signal1 -0.75) * 8190.0);
  digitized_signal2 <= integer((analog_signal2 -0.75) * 8190.0);
  digitized_signal3 <= integer((analog_signal3 -0.75) * 8190.0);

end process;
process(clk,digitized_signal0,digitized_signal1,digitized_signal2,digitized_signal3)
begin
IF RISING_EDGE(clk) THEN
  case CONT is
    when 0 => instantly_digitized_signal <= digitized_signal0;
    when 1 => instantly_digitized_signal <= digitized_signal1;
    when 2 => instantly_digitized_signal <= digitized_signal2;
    when 3 => instantly_digitized_signal <= digitized_signal3;
    when others => null;
  end case;
  channel <= conv_std_logic_vector(cont,2);
  IF CONT = max THEN
    CONT <= 0;
  else
    CONT <= CONT +1;
  END IF;

END IF;

end process;

analog_integer0 <= digitized_signal0;
analog_integer1 <= digitized_signal1;
analog_integer2 <= digitized_signal2;
analog_integer3 <= digitized_signal3;

adc_out <= conv_std_logic_vector(instantly_digitized_signal,12);
  digital_out <= adc_out;
end Behavioral;

```

La simulación de este apartado ADC es el siguiente:

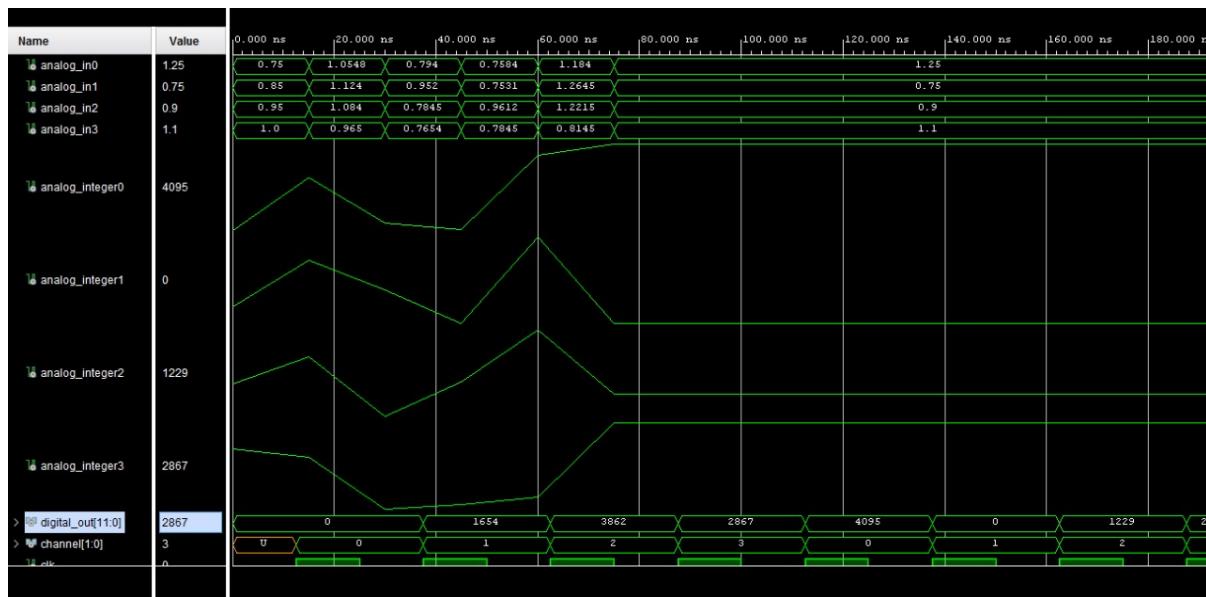


Figura 14: ADC Simulación diagrama de tiempos

### 3.7 Interpretación I2C

El maestro I2C utiliza la máquina de estado que se muestra en la Figura 2 para implementar el protocolo de bus I2C. Tras la puesta en marcha, el componente entra inmediatamente en el estado listo . Espera en este estado hasta que la señal ena se engancha en un comando. El estado de inicio genera la condición de inicio en el bus I2C, y el estado de comando comunica la dirección y el comando rw al bus. El estado slv\_ack1 luego captura y verifica el reconocimiento del esclavo. Dependiendo del comando rw, el componente procede a escribir datos en el esclavo ( estado wr ) o recibir datos del esclavo ( estado rd ).

Una vez completado, el maestro captura y verifica la respuesta del esclavo (slv\_ack2 state) si escribe o emite su propia respuesta ( estado mstr\_ack ) si lee. Si los ena pestillos de señal en otro comando, el maestro continúa inmediatamente con otra escritura ( WR estado ) o leer ( rd estado ) si el comando es el mismo que el comando anterior. Si es diferente del comando anterior (es decir, una lectura después de una escritura o una escritura después de una lectura o una nueva dirección de esclavo), el maestro emite un inicio repetido ( estado de inicio ) según la especificación I2C. Una vez que el maestro completa una lectura o escritura y la señal ena no se engancha en un nuevo comando, el maestro genera la condición de parada ( estado de parada ) y vuelve al estado listo .

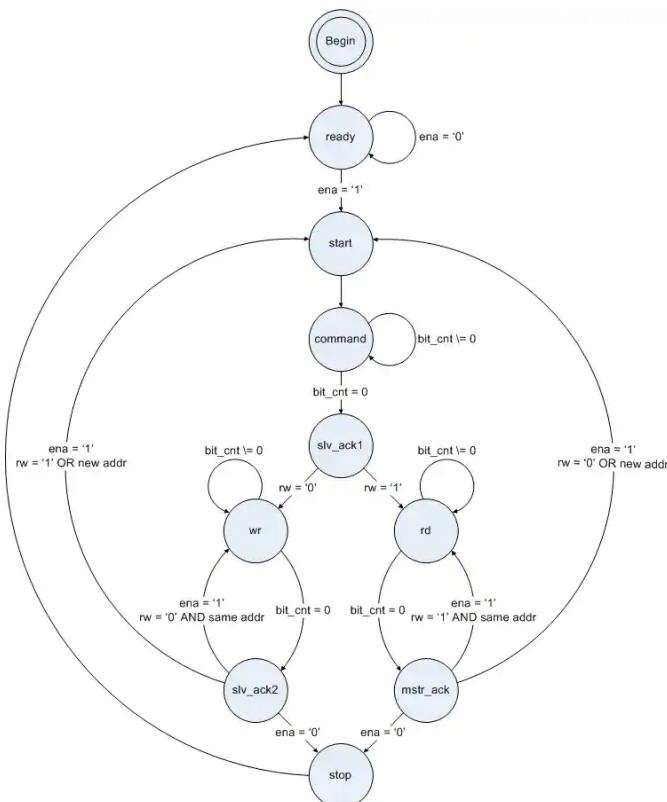
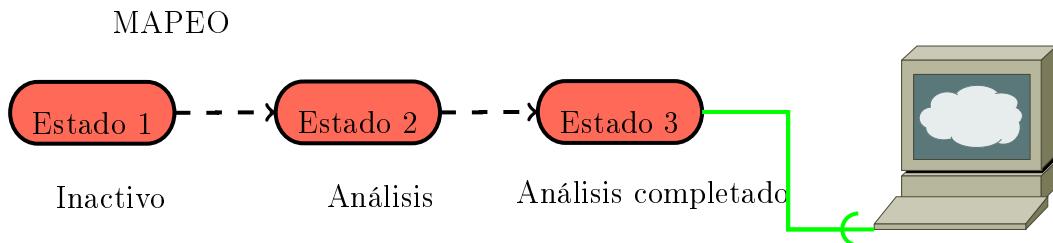


Figura 15: Máquina de estados maestro

### 3.8 Módulo - Mapeo (Señales y letras)

#### 3.8.1 Explicación del código

Primero designamos los puertos y las señales que nos ayudaran a realizar el procedimiento de la escritura y análisis de datos de lectura, que obtendremos del ADC para llevarlo al módulo del VGA mapeando las señales de cada letra.



**Puertos del módulo mapeo.**

Nombre	Long	Direcc	Descripción
clk	1	in	El reloj principal
digital_in	12	in	Línea de entrada del bloque ADC
channel	2	in	Indica el canal que se está analizando en los estados
Enable	1	in	Habilitador para iniciar el proceso de análisis de las señales
led_de_control	4	out	Salida de leds que designan la señal a representarse digitalmente
carta_entrante_vga	5	out	Valor enviado al VGA para ser mostrado en la pantalla

### Descripción de los estados:

*S1- inactivo.-*

Básicamente es el estado donde esta deshabilitado nuestro módulo, inicializamos los valores de la salida led\_de\_control y carta\_entrante\_vga. Para pasar al siguiente estado “enable” debe estar en 1 (habilitado)

*S2- análisis.-*

Estado donde se toma los valores de las señales del ADC, los cuales previamente fueron convertidos y tomados por los enteros de “p1, p2, p3, p4”. Solo cuando el contador “cont\_aux” alcance el valor de 125000 que esta designado por la constate “estable”, es decir cuando las señales que manda el ADC se mantengan estables durante 20mseg. Estas señales estables se guardaran en los enteros de catch1, catch2, catch3, catch4. Para después pasar al siguiente estado.

*S3- inactivo.-*

En este estado se evalúa los valores tomados de catch1, catch2, catch3, catch4 con los parámetros que representan cada letra, los cuales tienen un rango establecido en números enteros. Comparando los valores, según sea a la letra que representan se guardara en las salidas de led\_de\_control el cual se representara en una salida de leds y carta\_entrante\_vga que se enviara al MODULO VGA para ser mostrado en la pantalla.

El diagrama ASM que representa este modulo mapeo es el siguiente:

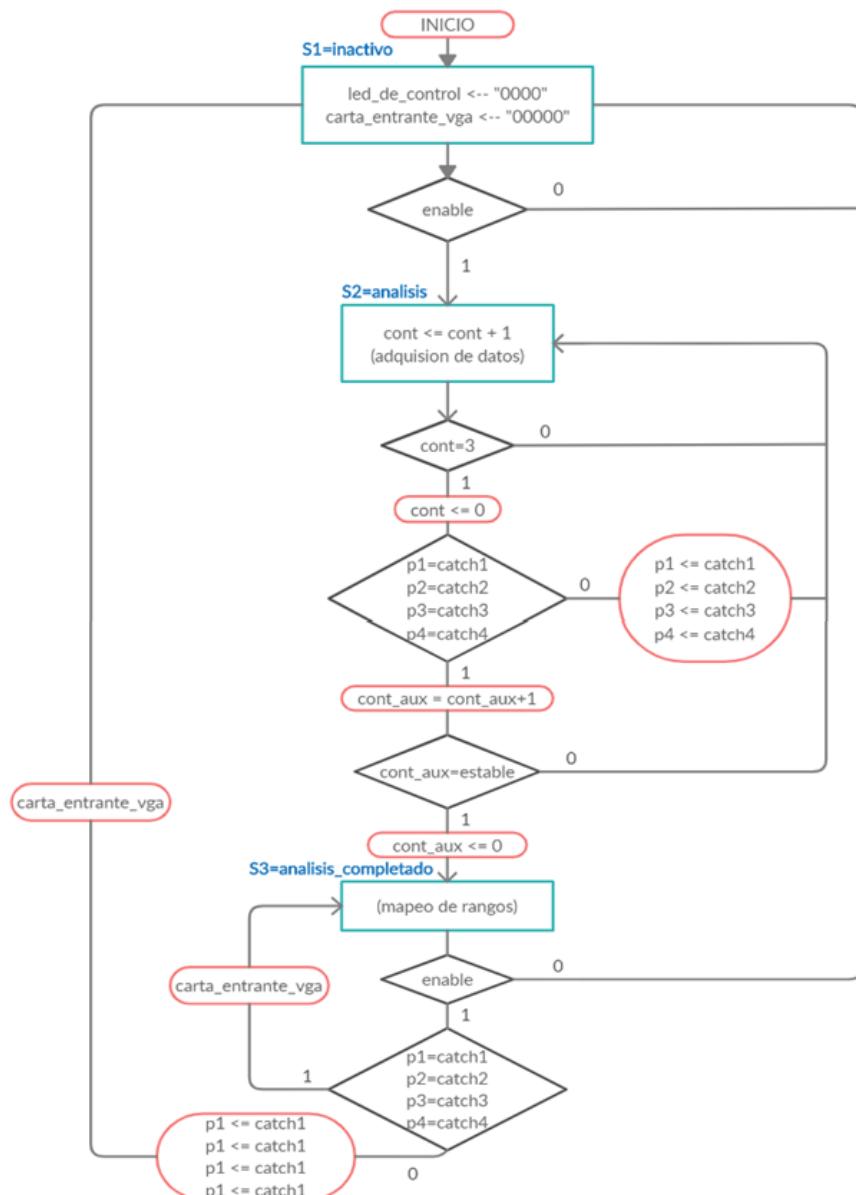


Figura 16: asm mapeo

**Mapeo.vhd**

```

library ieee;
use ieee.std_logic_1164.all;
use IEEE.Numeric_Std.all;
use ieee.math_real.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mapeo is
    Port (
        clk:in std_logic;
    
```

```

digital_in : in std_logic_vector(11 downto 0);
channel: in std_logic_vector(1 downto 0);
enable: in std_logic;
led_de_control: out std_logic_vector(3 downto 0);
carta_entrante_vga: out std_logic_vector(4 downto 0)
);
end mapeo;

architecture Behavioral of mapeo is
type nombre_estado is ( inactivo,analisis ,analisis_completado );
—— los estados que se usan para la separacion y el analisis
signal estado : nombre_estado := inactivo; —los estados que se usan para la
separacion y el analisis
—— signal part1: std_logic_vector(14 downto 0);
—— signal part2: std_logic_vector(14 downto 0);
—— signal part3: std_logic_vector(14 downto 0);
—— signal part4: std_logic_vector(14 downto 0);
signal clk_aux: std_logic:='0';
signal p1: integer range 0 to 4095;
signal p2: integer range 0 to 4095;
signal p3: integer range 0 to 4095;
signal p4: integer range 0 to 4095;
—— signal p1salida: integer range 0 to 4095;
—— signal p2salida: integer range 0 to 4095;
—— signal p3salida: integer range 0 to 4095;
—— signal p4salida: integer range 0 to 4095;
signal catch1: integer range 0 to 4095;
signal catch2: integer range 0 to 4095;
signal catch3: integer range 0 to 4095;
signal catch4: integer range 0 to 4095;
signal cont:integer range 0 to 3 :=0;
signal cont_aux:integer :=0;
signal estable:integer :=125000;—division de 20ms/4*clk simulacion, en la vida
real debe ser 1seg
signal pantallazo:integer :=416750; —16.67ms/40 ns
signal cont_pantallazo:integer :=0;
begin

p1<=to_integer(unsigned(digital_in))when channel = "00";
p2<=to_integer(unsigned(digital_in))when channel = "01";
p3<=to_integer(unsigned(digital_in))when channel = "10";
p4<=to_integer(unsigned(digital_in))when channel = "11";

process (clk,enable)
begin
if (enable = '0') then
estado <= inactivo;
cont_aux <= 0;
end if;

case estado is
when inactivo =>
led_de_control<="0000";
carta_entrante_vga<="00000";
if (enable = '0') then
estado <= inactivo;
else
estado <= analisis;
end if;

when analisis =>
if(rising_edge(clk)) then
cont <= cont +1;
if cont=3 then
cont <= 0;
clk_aux <= not clk_aux;

if cont_aux=estable then
cont_aux <= 0;
estado <= analisis_completado;
else
if p1=catch1 and p2=catch2 and p3=catch3 and p4=catch4 then

```

```

cont_aux <= cont_aux +1;
else
cont_aux <= 0;
catch1<=p1;
catch2<=p2;
catch3<=p3;
catch4<=p4;
estado <= analisis;
end if;
end if;
end if;

end if;

when analisis_completado =>
if (((catch1>2600)and(catch1<2720)) and ((catch2>1500)and(catch2<2100)) and ((catch3>1500)
) and(catch3<2100)) and ((catch4>1500)and(catch4<2150))) then
--A
    led_de_control<= "0001";
    carta_entrante_vga<= "00001";
    elsif(((catch1>2150)and(catch1<2550)) and ((catch2>2650)and(
catch2<4000)) and ((catch3>2700)and(catch3<4000)) and ((catch4>2800)and(catch4<4000)))
) then
--B
    led_de_control<= "0010";
    carta_entrante_vga<= "00010";
    elsif(((catch1>2450)and(catch1<2750)) and ((catch2>2400)and(
catch2<2700)) and ((catch3>2450)and(catch3<2750)) and ((catch4>2550)and(catch4<2800)))
) then
--C
    led_de_control<= "0011";
    carta_entrante_vga<= "00011";
    elsif(((catch1>2270)and(catch1<2550)) and ((catch2>2750)and(
catch2<4000)) and ((catch3>2200)and(catch3<2400)) and ((catch4>2200)and(catch4<2450)))
) then
--D
    led_de_control<= "0100";
    carta_entrante_vga<= "00100";
    elsif(((catch1>2100)and(catch1<2400)) and ((catch2>1900)and(
catch2<2340)) and ((catch3>2150)and(catch3<2420)) and ((catch4>2000)and(catch4<2300)))
) then
--E
    led_de_control<= "0101";
    carta_entrante_vga<= "00101";
    elsif(((catch1>2250)and(catch1<2600)) and ((catch2>1850)and(
catch2<2250)) and ((catch3>2650)and(catch3<4000)) and ((catch4>2850)and(catch4<4000)))
) then
--F
    led_de_control<= "0110";
    carta_entrante_vga<= "00110";
    elsif(((catch1>2520)and(catch1<2700)) and ((catch2>2700)and(
catch2<4000)) and ((catch3>2000)and(catch3<2250)) and ((catch4>2000)and(catch4<2250)))
) then
--G
    led_de_control<= "0111";
    carta_entrante_vga<= "00111";
    elsif(((catch1>2400)and(catch1<2700)) and ((catch2>2680)and(
catch2<4000)) and ((catch3>2650)and(catch3<4000)) and ((catch4>2000)and(catch4<2340)))
) then
--H
    led_de_control<= "1000";
    carta_entrante_vga<= "01000";
    elsif(((catch1>2700)and(catch1<2855)) and ((catch2>2660)and(
catch2<4000)) and ((catch3>2600)and(catch3<4000)) and ((catch4>2080)and(catch4<2315)))
) then
--K
    led_de_control<= "1001";
    carta_entrante_vga<= "01001";
    elsif(((catch1>2700)and(catch1<4000)) and ((catch2>2650)and(
catch2<4000)) and ((catch3>1900)and(catch3<2200)) and ((catch4>1900)and(catch4<2200)))
)
```

```

) then
    --L
        led_de_control<= "1010";
        carta_entrante_vga<= "01010";
        elsif((catch1>2400)and(catch1<2650)) and ((catch2>2010)and(
        catch2<2390)) and ((catch3>2220)and(catch3<2440)) and ((catch4>2280)and(catch4<2520))
    ) then
        --O
            led_de_control<= "1011";
            carta_entrante_vga<= "01011";
            elsif((catch1>2600)and(catch1<2750)) and ((catch2>2735)and(
            catch2<4000)) and ((catch3>2350)and(catch3<2500)) and ((catch4>2100)and(catch4<2340))
        ) then
            --P
                led_de_control<= "1100";
                carta_entrante_vga<= "01100";
                elsif(((catch1>2000)and(catch1<2500)) and ((catch2>1500)and(
                catch2<2040)) and ((catch3>1800)and(catch3<2100)) and ((catch4>1900)and(catch4<2240)))
            ) then
                --S
                    led_de_control<= "1101";
                    carta_entrante_vga<= "01101";
                    elsif(((catch1>2400)and(catch1<2690)) and ((catch2>2050)and(
                    catch2<2335)) and ((catch3>2000)and(catch3<2275)) and ((catch4>2000)and(catch4<2265)))
                ) then
                    --T
                        led_de_control<= "1110";
                        carta_entrante_vga<= "01110";
                        elsif(((catch1>1700)and(catch1<2240)) and ((catch2>2230)and(
                        catch2<2600)) and ((catch3>2140)and(catch3<2340)) and ((catch4>2140)and(catch4<2400)))
                ) then
                    --X
                        led_de_control<= "1111";
                        carta_entrante_vga<= "01111";
                        elsif(((catch1>2710)and(catch1<4000)) and ((catch2>1500)and(
                        catch2<2215)) and ((catch3>1500)and(catch3<2260)) and ((catch4>1500)and(catch4<2390)))
                ) then
                    --Y
                        led_de_control<= "0001";
                        carta_entrante_vga<= "10000";
                    else
                        led_de_control<= "0000";
                        carta_entrante_vga<= "00000";
                    end if;
                if rising_edge(clk) then
                    if cont_pantallazo = pantallazo then
                        estado <= analisis;
                        cont_pantallazo<=0;
                    else
                        cont_pantallazo<=cont_pantallazo+1;
                    end if;
                end if;
            end case;
        --
    end process;
end Behavioral;

```

### 3.9 Modulo - Controlador VGA

El comienzo del diseño comienza tomar las letras de vocabulario y esas 26 letras se usan como mayúsculas: A, B, C, D, E, F, G, H, K, L, O, P, S, T, X e Y para mostrarlas con una simulación de TEST BENCH cada señal proveniente de los módulos de conversor analógico- digital Y la etapa de mapeo de las señales se mostrarán a través de simulación en VIVADO.

Se usará una resolución de 480x640 de arreglo de gráficos de video, de esta manera se

podrá visualizar las letras del lenguaje de señas por medio de una simulación de tiempos. Se crearán módulos de SINCRONIZACION y MODULOS DE GRAFICAS VGA en el diseño estructural de cada módulo.

### 3.9.1 Características generales VGA

Para el control del VGA, la información de cada píxel de la pantalla se va enviando consecutivamente al ir barriendo toda la superficie por líneas horizontales y verticales. Cada píxel se compone de tres colores: rojo, verde y azul.

El envío de los valores de los píxeles se hace fila a fila empezando por el píxel de arriba a la izquierda y terminando con el píxel de abajo al derecho. Existen distintas resoluciones de pantalla, por ejemplo, de 640 columnas x 480 filas (640x480) y otras más habituales como 800x600 ó 1024x768 nosotros usaremos la resolución de 640 columnas x 480 filas (640x480).

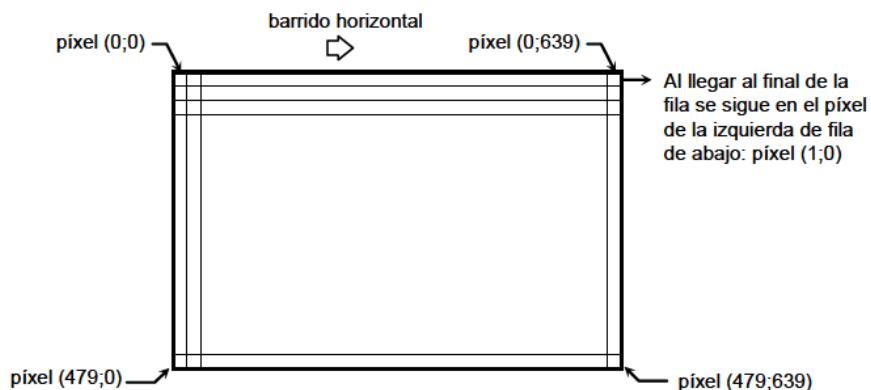


Figura 17: vga1

Para indicar el cambio de fila y el cambio de pantalla existen dos señales de sincronismo: sincronismo horizontal (hsynch) y vertical (vsynch). El sincronismo horizontal marca el final de una línea y el comienzo de una nueva. El sincronismo vertical marca el fin de una pantalla y el comienzo de una nueva. La frecuencia de refresco viene dada por la frecuencia del sincronismo vertical.

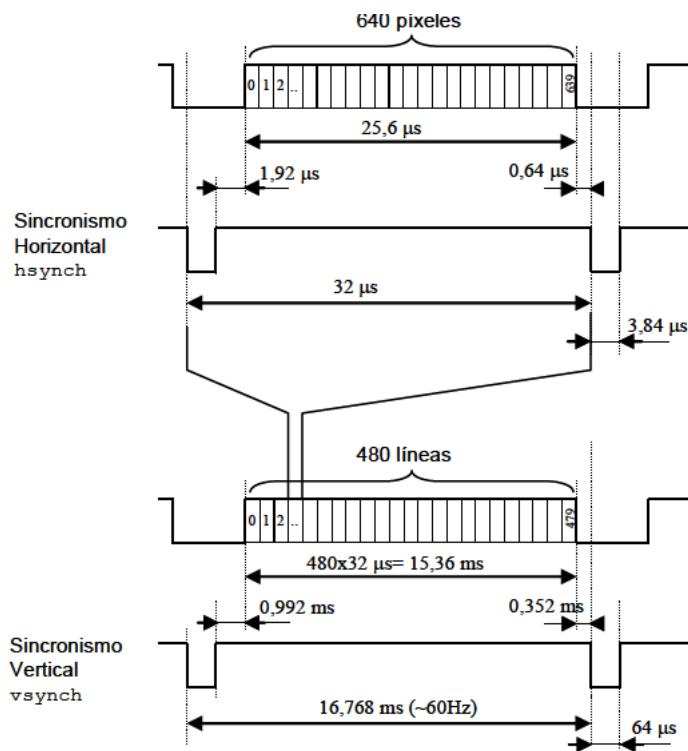


Figura 18: vga2

Usaremos una frecuencia de REFRESCO de 60Hz.

Los valores de los tiempos son aproximados. Existe un estándar que define diversas frecuencias a las que normalmente se adaptan los monitores, aunque como ya se ha dicho que varían según el fabricante. Observa que hay intervalos de tiempo de la información de los píxeles, a estos se les llama porches sincronismo y del porche delantero y trasero. Se muestran el porche trasero y delantero (back porch y front porch).

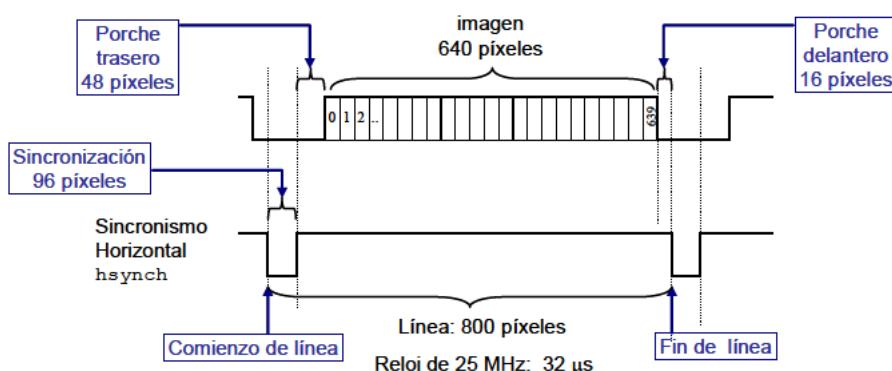


Figura 19: vga3

A veces, en el diseño digital las especificaciones no se dan en tiempos, sino que se da la frecuencia a la que salen los píxeles. Con esta frecuencia, se especifica el número de píxeles de los porches y las señales de sincronización par Horizontal y Vertical.

### 3.9.2 Los valores de la pantalla de VGA

Usaremos los valores estándar según el IC NEXYS 4DD.

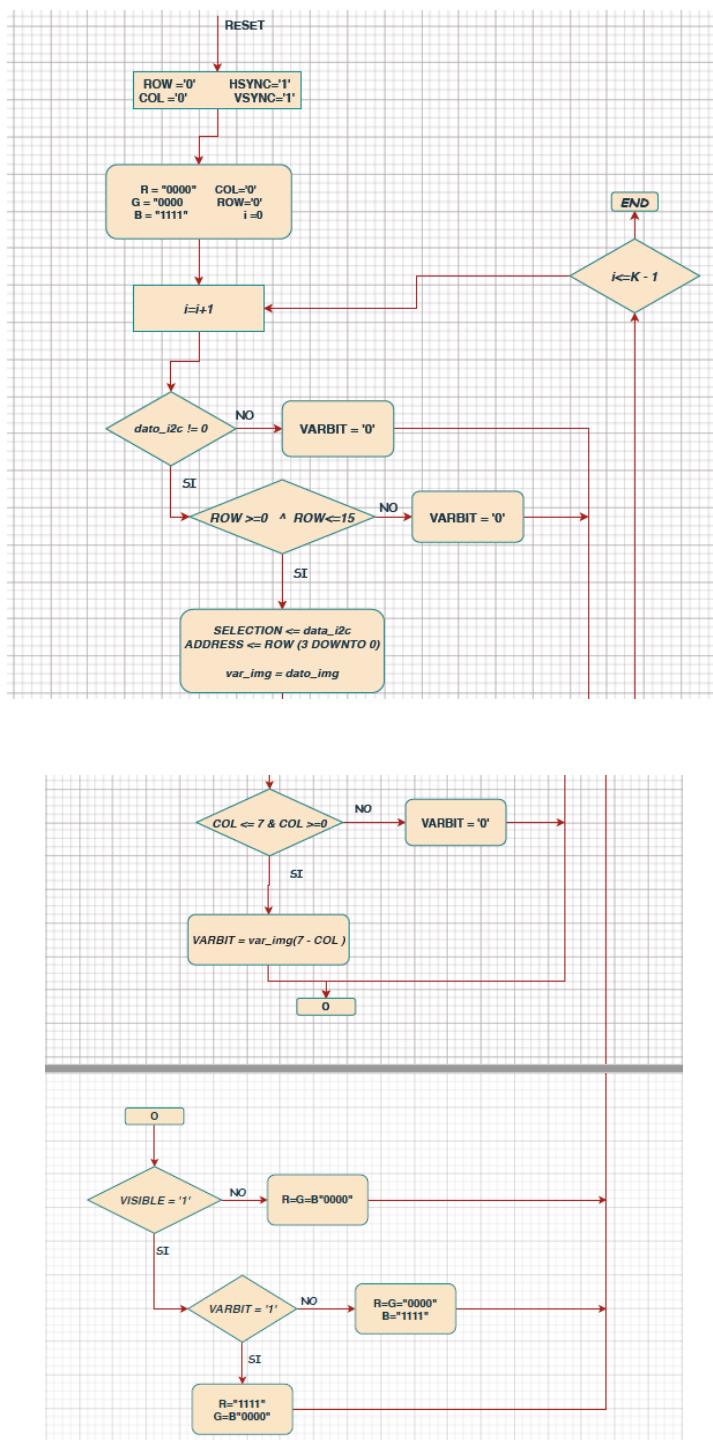
Formato	Reloj MHz	Horizontal (en píxeles)					Vertical (en líneas)				
		Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total	Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total
640x480@60Hz	25	640	16	96	48	800	480	9	2	29	520

Figura 20: vga4

### 3.10 Módulo de sincronización de VGA

En sincronización.

La descripción está según el diagrama siguiente.



En resumen la configuración es la siguiente:

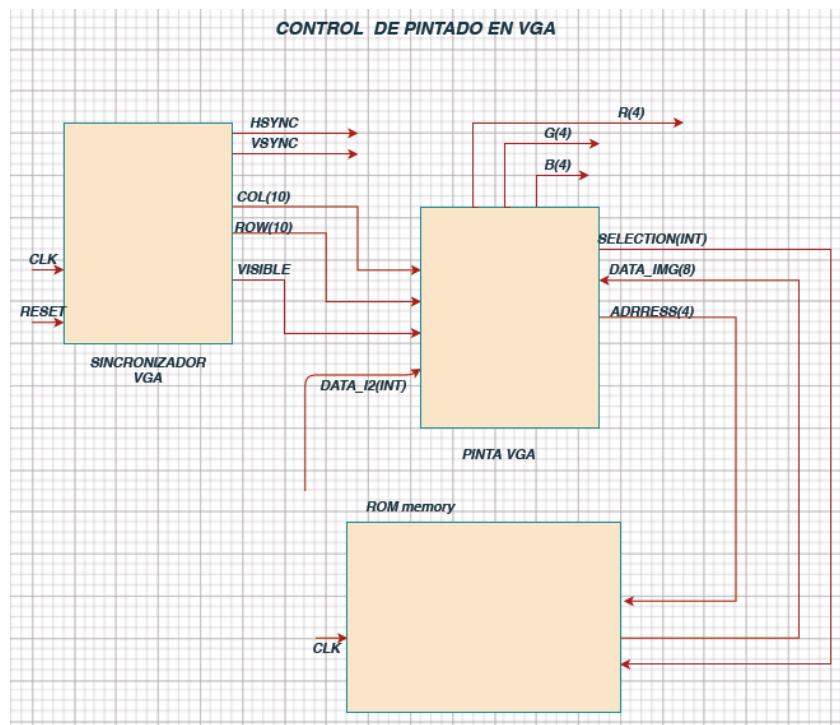


Figura 21: Control de pintado en VGA

Se presenta la ROM donde se almacenan los datos de cada patrón.

## SINCRONIZADOR (TOP MODULE)

### MODULO HYSYNC (SUBMODULE)

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD.LOGIC_UNSIGNED.ALL; — INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES
ARITMETICAS EN STD_LOGIC TIPO DE DATOS Y OPERADORES MATEMATICOS
use ieee.numeric_std.all ; — concvbersiobnd edatos TO_INTEGER

— Uncomment the following library declaration if using
— arithmetic functions with Signed or Unsigned values
—use IEEE.NUMERIC_STD.ALL;

— Uncomment the following library declaration if instantiating
— any Xilinx leaf cells in this code.
—library UNISIM;
—use UNISIM.VComponents.all ;

entity PINTA_VGA is
  Port (
    COL_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0); —ESQUINA SUPERIOR
    IZQUIERDA
    ROW_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0); — PARA DIRECCION DE
    MEMORIA DE ROM
    VISIBLE_P: IN STD_LOGIC; — PORCH
    LA DIRTERCCION DE MEMORIA
    data_img: IN STD_LOGIC_VECTOR(7 DOWNTO 0); — VECTOR DEVUELVE
    bestia
    data_i2c: IN INTEGER RANGE 0 TO 50; — corazon de la
    CARACTER DE LA ROM
    SELECTION: OUT INTEGER RANGE 0 TO 50; — PARA ELEGIR EL
  );
end entity;
  
```

```

>ADDRESS           address: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- relacion ROW-
      — COLOR
      R_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      G_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
      B_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
end PINTA_VGA;

architecture Behavioral of PINTA_VGA is

— PINTADO: FONDO --> AZUL
— — LETRA --> ROJO


begin
PROCESS(data_i2c,data_img,COL_P,ROW_P,VISIBLE_P) — LISTA DE SENSIBILIDAD
VARIABLE var_bit: STD_LOGIC:='0'; — PARA ALMACENRA EL bit BIFURCADO
VARIABLE var_img: STD_LOGIC_VECTOR(7 DOWNTO 0); — PARA ALMACENRA el vector

BEGIN

— PART1: verificacion de DATA
  IF(data_i2c /= 0) THEN
    IF(ROW_P>=0 AND ROW_P<=15 )THEN — UBICACION DE MI FILA
      SELECTION <=data_i2c; — union para elegir la LETRA
      address<= ROW_P(3 DOWNTO 0); — LOS MENOS SIGNIFICATIVOS DE
      — almacenamiento del dto
      var_img:=data_img; — 8 bits , vectores de la Imagen
    ELSE
      var_bit:=var_img(7- TO_INTEGER(UNSIGNED(COL_P))); — DAATOS DESDEel :
      MSB --> LSB
      ELSE
        var_bit:='0'; — FUERA DEL RANGO DE COL_P solo azul
      END IF;
    ELSE
      var_bit:='0';
    END IF;

  ELSE — asegureamos que se pinta sea AZUL
    var_bit:='0';
  END IF;

— PART3: PINTADO
  IF(VISIBLE_P = '1')THEN
    IF(var_bit='1')THEN
      R_P<="1111";
      G_P<="0000";
      B_P<="0000";
    ELSE
      R_P<="0000";
      G_P<="0000";
      B_P<="1111";
    END IF;

  ELSE — zona no visible
    R_P<="0000";
    G_P<="0000";
    B_P<="0000";
  END IF;

END PROCESS;

end Behavioral;

```

Luego de la simulación de la descripción vhdl se tiene el diagrama de tiempos:



Figura 22: Diagrama de simulación en el tiempo VGA project

## 4. Implementación de diseño

---

### 4.1 Productos y materiales a utilizar

**Nota:** En este trabajo se diseña una asimilación de funcionamiento, ya que solamente se utilizará el software *Vivado 2020.1*.

Está planificado para un funcionamiento real experimental, por consiguiente, se muestra un listado que permitirá implementar el desarrollo de este proyecto.

- FPGA
- PMOD AD2 – (En este trabajo se reemplaza por FPGA Artix 7)
- Sensores flexibles de 4,5 pulgadas (4 piezas) (En este trabajo se asimila entradas analógicas directas al FPGA)
- Resistencias
- Puentes
- Soldador
- Multímetro
- Esposas de plástico
- Tubo termocontraíble
- Cable VGA
- Monitor VGA
- Cables
- Silicona caliente
- Guante
- Microsoft Windows 10
- Vivado 2020.1

- Generar el archivo bitstream (Generate Bitstream)

Se generó el archivo parquimetro.bit

Con el cual tenemos toda la configuración del programa y podremos cargarlo en nuestra tarjeta para la implementación.

- Descargar el bitstream en el FPGA. (Open Hardware Manager)

## BITSTREAM FPGA

## 5. Resultados

---

Uno de los aspectos más importantes de este proyecto es la correcta clasificación de la entrada recibida, que es reenviada a un mapeador y luego a la pantalla VGA, produciendo el resultado esperado que será traducido por la descripción FPGA vhdl en texto. Al principio, los datos deben enviarse desde el la generación de señales en vivado. Tiene que recibir completamente la señal, tiene que asegurar una conectividad rápida y confiable; luego, la comparación ocurre en el proceso de backend después de obtener la entrada reenviada en cuestión de poco tiempo y esto mejora la funcionalidad del proyecto. También es claro y está bien diseñado para que el usuario no se pierda mientras lo usa. La salida de VGA puede entenderse claramente y el usuario puede cambiar la señal dependiendo del movimiento.

A continuación se muestra la simulación general del proyecto.

## 6. Conclusiones y Recomendaciones

---

- El sistema Sign Language Translator (hardware y software) se describe en este documento. El objetivo de este sistema es detectar cualquier cambio en el gesto de la mano que proporcione datos o información suficientes para permitir que el sistema traduzca el signo en letra. Se descubrió que este sistema es lo suficientemente preciso, confiable y asequible, lo que brinda la oportunidad de que todas las personas que sufren de discapacidad auditiva lo utilicen en cualquier momento que necesiten comunicarse con otras personas. Los sistemas, similares a este diseño que emplea tecnologías de comunicación y sensores avanzados, abren nuevos horizontes para mejorar nuestros estilos de vida. Se están estableciendo planes futuros para mejorar el sistema propuesto. Primero, se implementará una mayor calidad de sensores flexibles o giroscopios y / o acelerómetros en este sistema para mejorar su desempeño y reducir errores. En segundo lugar, el sistema se volverá inalámbrico para transmitir datos a un teléfono inteligente, esto debería eliminar el uso de una pantalla VGA y aprovechará las aplicaciones de texto a voz disponibles en los teléfonos inteligentes. Así mencionamos la implementación de tecnología watson de IBM, con el cual precisariamos de señales procesadas para que se pueda emitir sonido desde un solo movimiento de la mano.
- Lo que nos hace más humanos, más serviciales, y sentir mejor, es el poder aplicar los conocimientos adquiridos en la universidad al servicio de la comunidad, buscando que por medio de la tecnología se pueda eliminar estas diferencias debido a las capacidades diferentes de otras personas y hacernos uno solo, trabajando por un

bien común consiguiendo que estos detalles de comunicación sean desapercibidos y todos nos podamos comunicar sin problemas.

- El prototipo es muy fácil de usar, proporciona una gran ayuda para la enseñanza-aprendizaje de lenguaje, el diseño es práctico fácil de conectar y manipular.
- La información existente para realizar este proyecto de investigación, sirvió de mucho para poder realizar el propio, analizando los diferentes métodos y aplicaciones de los sensores existentes, los mismos que sirvieron para poder aportar en nuestra sociedad con un prototipo útil para el destinatario final.
- Con este prototipo se lanza una gran idea para que poco a poco vaya creciendo y perfeccionarla de tal manera que se convierta en un diseño pequeño, portable y la gente con discapacidad de hablar, se pueda comunicar fácilmente con personas que desconocen del lenguaje sordomudo.

---

## Referencias

---

- [1] Kam, David, "The Importance of Communication,"Marketing Deviant - Marketing Business Strategies, June, 2009, <http://marketingdeviant.com/the-importance-of-communication>
- [2] World Federation of the Deaf (WFD), <http://wfdeaf.org>
- [3] Su-Jing Wang; De-Cai Zhang; Cheng-Cheng Jia; Na Zhang; Chun- Guang Zhou; Li-Biao Zhang, .^ Sign Language Recognition Based on Tensor,"Second International Conference on Multimedia and Information Technology (MMIT), vol.2, pp.192-195, April 2010.
- [4] Allen, J.M.; Asselin, P.K.; Foulds, R., .^merican Sign Language finger spelling recognition system,"2003 IEEE 29th Annual Proceedings of Bioengineering Conference, pp.285-286, 2003
- [5] P. Espinosa, H.Pogo. "Diseño y construccion de un guante prototipo electrónico capaz de traducir el lenguaje de señas de una persona sordomuda al lenguaje de letras". UNIVERSIDAD POLITÉCNICA SALESIANA. Ecuador 2013.
- [6] R. Achkar, G. Abou Haidar, Sayah and Fadi Joubran. "Sign Language Translator using the Back Propagation Algorithm of an MLP". Department of Computer and Communications Engineering. American University of Science and Technology, AUST Beirut, Lebanon

## ANEXOS



## FLEX SENSOR FS

### Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
  - Simple Construction
  - Low Profile

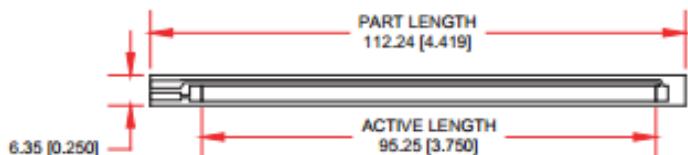
### Mechanical Specifications

- Life Cycle: >1 million
- Height: ≤0.43mm (0.017")
- Temperature Range: -35°C to +80°C

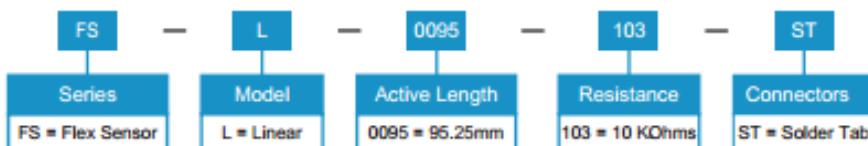
### Electrical Specifications

- Flat Resistance: 10K Ohms
- Resistance Tolerance: ±30%
- Bend Resistance Range: 60K to 110K Ohms
- Power Rating : 0.50 Watts continuous. 1 Watt Peak

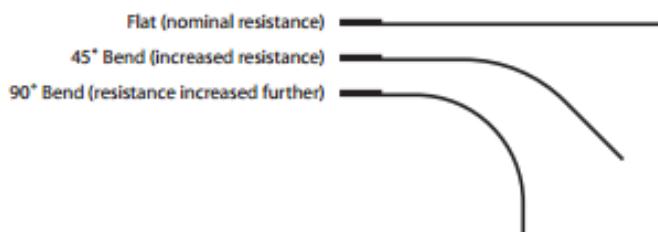
### Dimensional Diagram - Stock Flex Sensor



### How to Order - Stock Flex Sensor



### How It Works





## FLEX SENSOR FS

*Special Edition Length*

### Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
  - Simple Construction
  - Low Profile

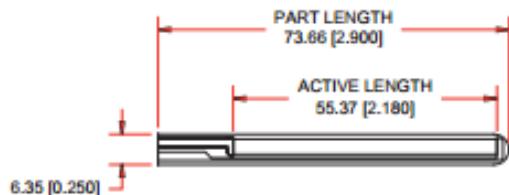
### Mechanical Specifications

- Life Cycle: >1 million
- Height: ≤0.43mm (0.017")
- Temperature Range: -35°C to +80°C

### Electrical Specifications

- Flat Resistance: 25K Ohms
- Resistance Tolerance: ±30%
- Bend Resistance Range: 45K to 125K Ohms (depending on bend radius)
- Power Rating : 0.50 Watts continuous. 1 Watt Peak

### Dimensional Diagram - Stock Flex Sensor



### How to Order - Stock Flex Sensor

FS	-	L	-	0055	-	253	-	ST
Series		Model		Active Length		Resistance		Connectors
FS = Flex Sensor		L = Linear		0055 = 55.37mm		253 = 25K Ohms		ST = Solder Tab

### How It Works

