UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO Y CONTRUCCIÓN DEL CONTROL DE UN ANDROIDE UTILIZANDO UN CPLD Y LENGUAJE VHDL

QUE PARA OBTENER EL TÍTULO DE
INGENIERA ELÉCTRICA-ELECTRONICA

PRESENTAN:
GUEVARA JUAREZ JOSÉ IVAN
LEYVA FERNÁNDEZ MARCIAL ROBERTO

DIRECTOR DE TESIS:
M.I. NORMA ELVA CHÁVEZ RODRÍGUEZ

MÉXICO D.F. MAYO 2009

ÍNDICE página

CAPÍTULO V “Diseño y Construcción del Control para el Androide”

CAPÍTULO VI “Resultados Finales y Conclusiones”

ANEXOS

CAPÍTULO I

Introducción a la Lógica Programable

1.1 Definición de dispositivo lógico programable

Un dispositivo lógico programable, o PLD (Programmable Logic Device), es un dispositivo cuyas características pueden ser modificadas y almacenadas mediante programación. El dispositivo programable más común es el PAL (Programmable Array Logic), cuyo circuito interno consiste de una matriz de conexiones, un arreglo de compuertas "AND" y un arreglo de compuertas "OR".

Una matriz de conexiones es una red de conductores distribuidos en renglones y columnas, con un interruptor en cada punto de intersección, mediante la cual se seleccionan cuales entradas del dispositivo serán conectadas al arreglo "AND". Las salidas de las compuertas "AND', son conectadas al arreglo de compuertas "OR" para, de esta manera, obtener una función lógica en forma de suma de productos. Las matrices pueden ser fijas o programables. Con estos recursos se implementan las funciones lógicas deseadas mediante un software especial y un programador de dispositivos. El tipo más sencillo de matriz programable data de los años 60 y era una matriz de diodos con un interruptor en cada punto de intersección de la misma. En la figura 1.1 se muestran los circuitos básicos para la mayoría de los PLD's.

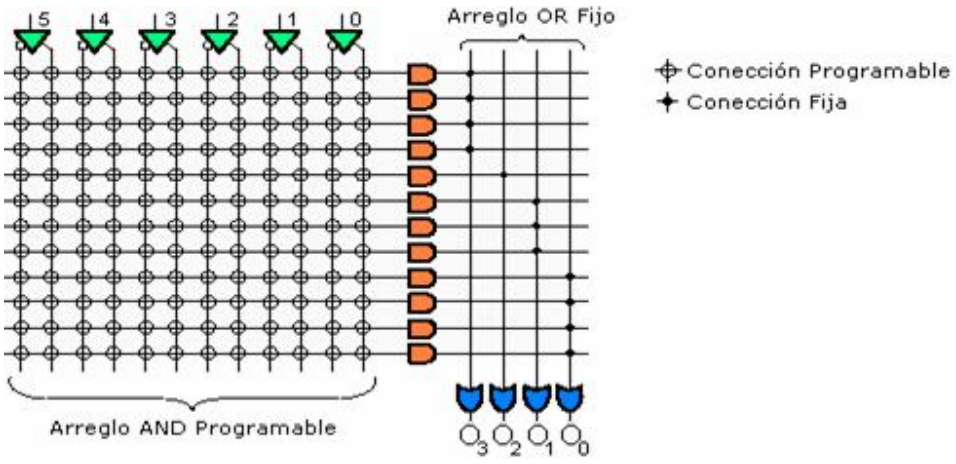


Figura 1.1 Estructura comúnmente utilizada para la mayoría de los PLD's [2]

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

1.2 Conceptos fundamentales

Para el diseño y fabricación de los dispositivos lógicos programables se requiere del análisis de los conceptos siguientes:

- Funcionalidad completa

  La cual se fundamenta en el hecho de que cualquier función lógica se puede realizar mediante una suma de productos.

- Celdas de funciones universales

  También denominadas generadores de funciones, son bloques lógicos configurados para procesar cualquier función lógica, los cuales tienen un funcionamiento similar al de una memoria. En estas celdas se almacenan los datos de salida del circuito combinacional en vez de realizar físicamente la ecuación booleana.

Existe una gran variedad de dispositivos lógicos programables, a continuación se hace mención de algunos de ellos:

- ASIC (Application Specific Integrated Circuit)

  Significa circuito integrado de aplicación específica e identifica a un dispositivo definido por el usuario. Los ASICs, al contrario de otros dispositivos, pueden contener funciones analógicas, digitales y combinaciones de ambas. En general, son programables mediante máscara y no programables por el usuario. Esto significa que los fabricantes configuran el dispositivo según las especificaciones del usuario. Se usan para combinar una gran cantidad de funciones lógicas en un dispositivo; sin embargo, estos dispositivos tienen un costo inicial alto.

- EPROM (Erase Programmable Read Only Memory)

  Significa memoria de sólo lectura, programable y borrable. Identifica un sistema cuya arquitectura puede ser dividida en dos arreglos, un arreglo no programable de compuertas “AND’s” conectado a un arreglo programable de compuertas “OR’s” de forma tal que se puede realizar cualquier función booleana.

- PAL (Programmable Array Logic)

  Significa lógica de arreglo programable y su arquitectura puede ser dividida en dos arreglos, uno, programable, de compuertas “AND’s” conectado a otro, no programable, de compuertas “OR’s” de forma tal que se puede realizar cualquier función booleana.

- PLA (Programmable Logic Array)

  Significa arreglo lógico programable y su arquitectura puede ser dividida en dos arreglos, un arreglo programable de compuertas "AND's" conectado a un arreglo programable de compuertas "OR's" de forma tal que se puede realizar cualquier función booleana.

- CPLD (Complex Programmable Logic Device)

  Significa dispositivo lógico programable complejo y extiende el concepto de un PLD a un mayor nivel de integración, permitiendo realizar sistemas más eficientes que utilizan menos espacio, mejoran la confiabilidad y reducen costos. Un CPLD se forma con múltiples bloques lógicos, cada uno similar a un PLD, que se comunican entre sí utilizando una matriz programable de interconexiones, lo cual hace más eficiente el uso del silicio y conduce a un mejor desempeño.

- FPGA (Field Programmable Gate Array)

  Significa arreglo de compuertas programables en el campo y consiste en arreglos de varias celdas lógicas que se comunican unas con otras mediante canales de conexiones verticales y horizontales.

1.3 Antecedentes

Los dispositivos programables han pasado por una larga evolución para llegar a la complejidad que tienen hoy en día. En las secciones siguientes se abordan de manera cronológica la discusión de estos dispositivos, desde los menos complejos hasta los más complejos.

- Memoria Programable de Sólo Lectura (PROMs)

  Las memorias programables de sólo lectura o PROMs son memorias poco costosas que pueden ser programadas por el usuario y que contienen un patrón específico. Este patrón puede ser usado para representar el programa de un microprocesador, un simple algoritmo o una máquina de estados. Algunas PROMs se pueden programar una sola vez mientras que otro tipo de PROMs, tales como las memorias EPROM o EEPROM, pueden ser borradas y programadas varias veces.

  Las PROMs son excelentes para la aplicación de cualquier tipo de lógica combinatoria con un número limitado de entradas y salidas. Para lógica secuencial, debe ser agregado un reloj exterior como con los microprocesadores. También las

PROMs tienden a ser extremadamente lentas, por lo que no son útiles para aplicaciones en las que la velocidad es un problema.

- Arreglos Lógicos Programables (PLAs)

Los Arreglos Lógicos Programables (PLAS) fueron una solución a la velocidad de entrada y limitaciones en la programación. Los PLAs constan de un gran número de entradas conectadas a un plano de ANDs, donde diferentes señales pueden ser combinadas lógicamente, mediante compuertas AND de acuerdo a como se programa el dispositivo. Los productos del plano AND se conectan a un plano de OR´s, donde los términos son OR-n juntos en diferentes combinaciones de productos.

Estos dispositivos pueden aplicar un gran número de funciones combinacionales, aunque no todas las posibles combinaciones como uno puede ser implementadas, tal como en la PROM. Sin embargo, por lo general tienen muchas más entradas y son mucho más rápidos. En la figura 1.2 se muestra la estructura básica de un PLA.

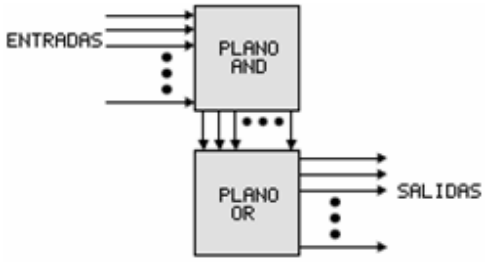


Figura 1.2 Estructura básica de un PLA [2]

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

- Matriz Lógica Programable (PALs)

  La matriz de lógica programable (PAL) es una variación de la PLA. Al igual que el PLA, tiene un plano AND programable para cada combinación de las entradas. Sin embargo, el plano OR es fijo, limitando el número de términos que pueden obtenerse. Otros dispositivos de lógica básica, tales como multiplexor, EXOR, y LATCH se añaden a las entradas y salidas. Lo que es más importante, los elementos síncronos, por lo general los flip-flop están incluidos. Estos dispositivos ya son capaces de aplicar un gran número de funciones, incluida la lógica secuencial sincronizada necesaria para realizar una máquina de estados. En la figura 1.3 se muestra la estructura básica de una PAL.

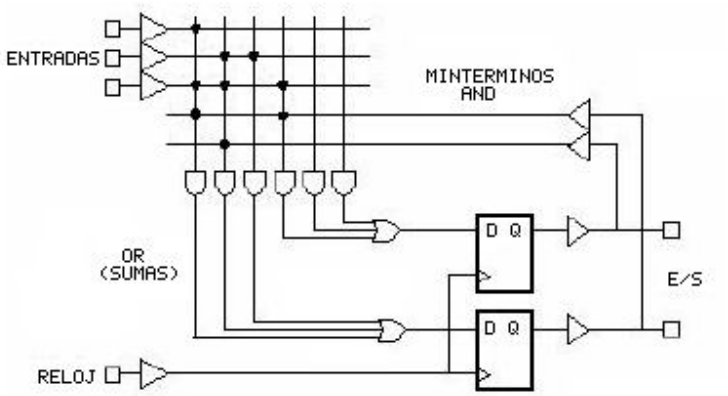


Figura 1.3 Estructura Básica de una PAL [2]

1.4 Arquitectura básica de un CPLD

Los dispositivos lógicos programables (PLDs), en su gran mayoría, se utilizan en aplicaciones para reemplazar a los circuitos SSI (Small Scale Integration) y MSI (Medium Scale Integration), ya que ahorran espacio y reducen el número y costo de los dispositivos empleados en el diseño. Sin embargo, estos dispositivos son poco útiles cuando se incorporan en el diseño de funciones lógicas muy complejas; en este caso, lo más apropiado es utilizar dispositivos con tecnología VLSI (Very Large Scale Integration), de muy alta escala de integración. Un dispositivo VLSI es un circuito que incorpora desde miles hasta millones de compuertas lógicas dentro de un sólo chip. Un ejemplo de los dispositivos VLSI programables son los CPLDs, que por lo general se basan en la tecnología EEPROM (Electrical Erase Programmable Read Only Memory) y tienen la estructura general mostrada en la figura 1.4.

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

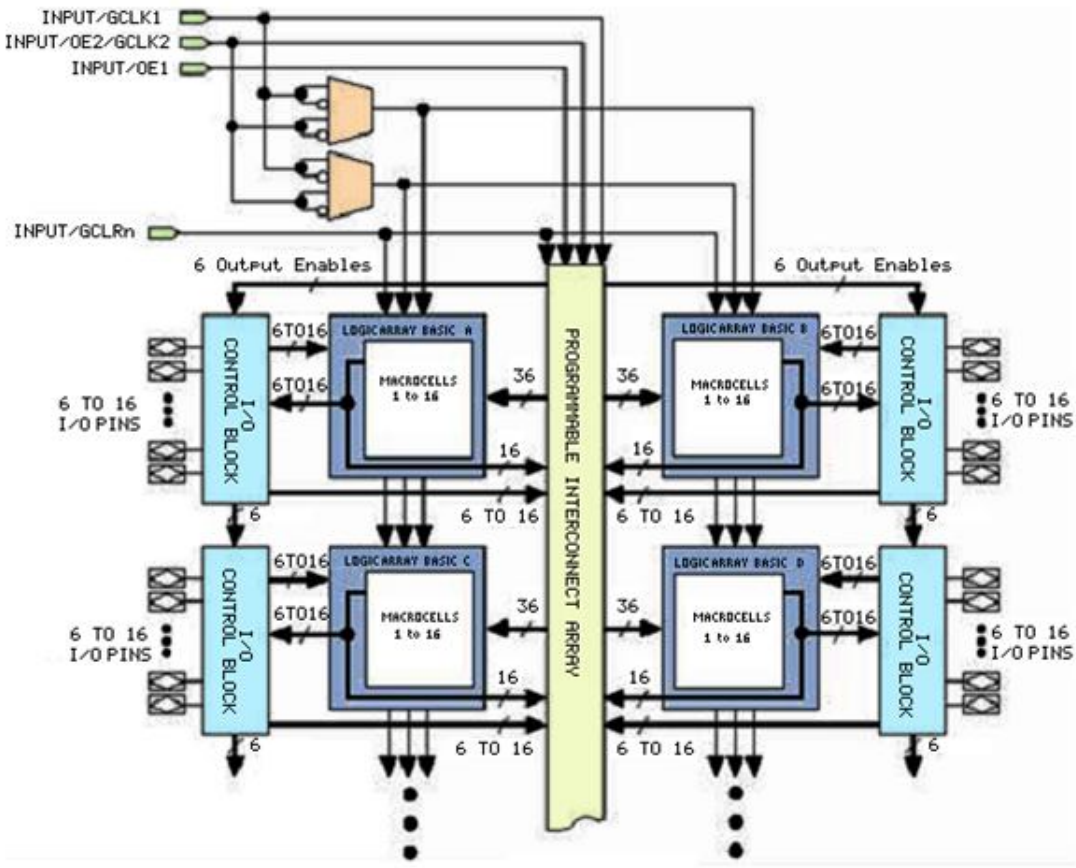


Figura 1.4 Estructura general de un CPLD [2]

Al utilizar el CPLD la tecnología EEPROM para almacenar la información de la programación, ésta permanecerá en el circuito hasta que sea programado nuevamente. La falta de energía en el circuito no implica pérdidas de información.

Como se observa en la figura 1.4, el CPLD está constituido por varios componentes que conviene describir brevemente, tal como se hace a continuación.

- LAB (Logic Array Basic)

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

La estructura de un CPLD está compuesta de varios arreglos lógicos básicos llamados LABs. Las salidas de los LABs se conectan a una matriz de interconexiones programables denominada PIA (Programmable Interconnect Array). A esta matriz de interconexiones también se conectan las salidas de los bloques de control de entrada/salida, permitiendo la comunicación del circuito con el mundo exterior. La matriz de interconexiones puede re-programarse para conectar las entradas del circuito con los bloques lógicos internos, según las necesidades del diseño.

- Macrocelda

Cada arreglo lógico básico contiene macroceldas, que se componen de bloques funcionales como son la matriz lógica, la matriz de selección de términos producto y el registro programable, los cuales se observan en la figura 1.5

La matriz lógica se utiliza para realizar funciones combinacionales, generando los términos producto de la macrocelda. La matriz de selección de términos producto asigna estos términos a las entradas de la lógica primaria o a las entradas de la lógica secundaria.

La lógica primaria, compuesta por las compuertas "OR" y "XOR", sirve para realizar funciones combinacionales, mientras que la lógica secundaria controla las señales de clear, preset, clock y clock enable del registro de la macrocelda.

La macrocelda también cuenta con dos tipos de extensiones: las compartidas y las paralelas. Las extensiones compartidas invierten los términos producto de la macrocelda y los realimenta a la matriz lógica. Las extensiones paralelas toman prestados los términos producto de macroceldas adyacentes y los alimentan a la matriz de selección. Gracias a las extensiones paralelas, una macrocelda puede emplear compuertas "AND" de macroceldas vecinas.

El flip-flop de la macrocelda puede programarse como flip-flop tipo "D", "T", "JK" ó "SR", con señal de reloj y cualquier combinación de señales preset, clear y clock enable. Estas señales son controladas por la lógica secundaria.

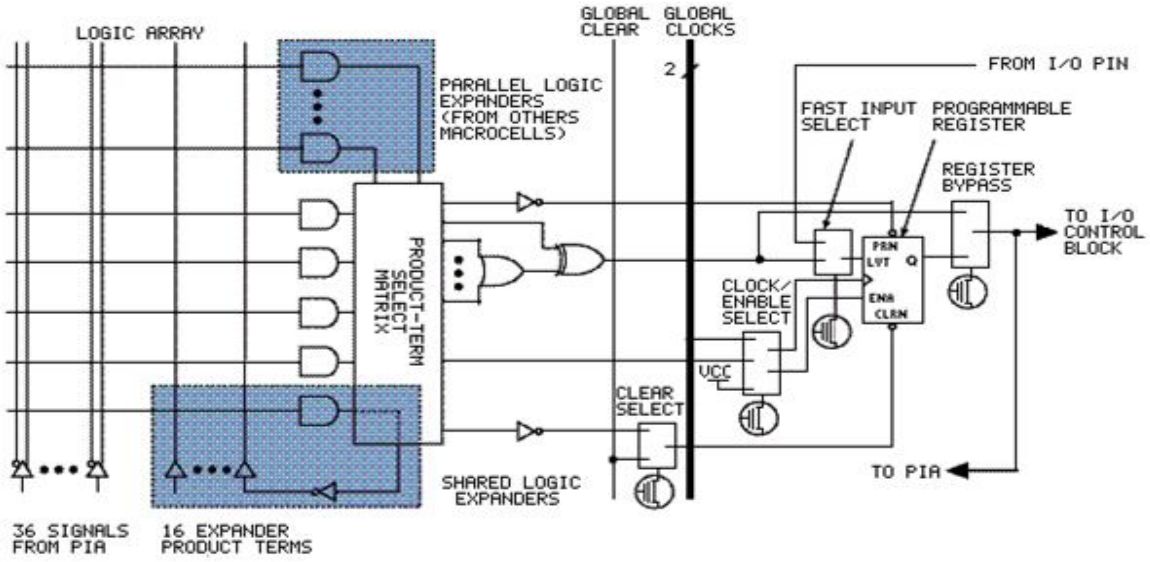


Figura 1.5 Estructura de una macrocelda [2]

- Celda de entrada/salida

  Con respecto a la entrada y salida, cada macrocelda de un LAB está conectada a los bloques de control de entrada/salida. Este bloque está compuesto de un pin bidireccional controlado por un búfer de tres estados, de manera que un sólo pin sirve como canal de entrada y/o de salida. El control del flujo de datos sobre estos pines lo ejecuta la lógica de las macroceldas. La figura 1.6 muestra el bloque de control de entrada/salida de las macroceldas.

  Todos los CPLDs tienen canales de entrada de alta velocidad hacia el registro de la macrocelda. Estos canales están conectados a los pines de entrada/salida del dispositivo, de manera que el dato pasa directamente a la entrada D del flip-flop evitando pasar por la PIA ó por la lógica combinacional. El dato de una entrada configurada de esta forma tarda 2.5 nanosegundos en llegar al flip-flop.

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

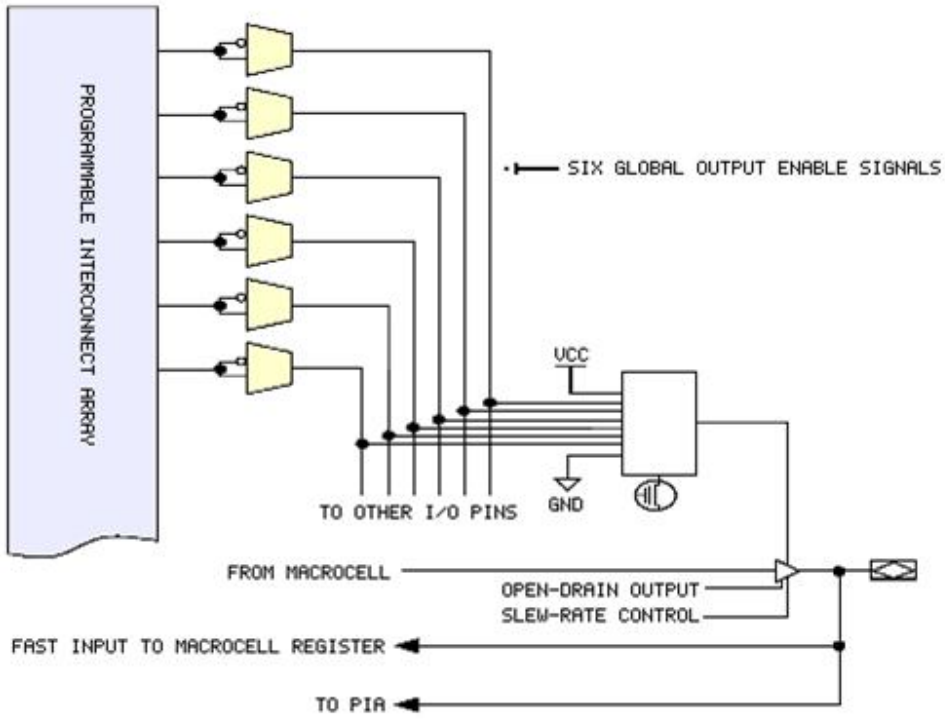


Figura 1.6 Bloque de control de entrada/salida [2]

- EAB (Embedded Array Block)

  Un EAB consiste de un bloque de memoria RAM con registros conectados a sus puertos de entrada y salida. Los EABs permiten realizar funciones lógicas utilizando para ello una tabla LUT, creada durante la configuración del dispositivo.

  En esta tabla son almacenados los resultados de la función combinacional, de manera que ya no son calculados sino consultados. La figura 1.7 muestra la estructura de un EAB.

  Los EABs pueden realizar memorias RAM síncronas, las cuales son mucho más fáciles de utilizar que las RAM asíncronas. Esto se debe a que las RAM síncronas generan automáticamente la señal de habilitación de escritura en la memoria, en cambio, en las RAM asíncronas esta señal debe generarse vía programación. Cada EAB puede configurarse para crear memorias de las siguientes capacidades: 256x8, 512x4, 1024x2 y 2048x1. Bloques de RAM de mayor capacidad pueden crearse combinando múltiples EABs; por ejemplo, dos bloques de 256x8 pueden combinarse en un bloque de 256x16, o dos bloques de 512x4 en un bloque de 512x8.

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

Si es necesario, todos los EABs en el dispositivo pueden conectarse en cascada para formar un sólo bloque de RAM. Este bloque de RAM puede estar compuesto de hasta 2048 palabras sin que los tiempos de acceso a la memoria se vean afectados. Cada EAB es alimentado por un renglón de interconexiones y puede dirigir su salida hacia un renglón o una columna de interconexiones.

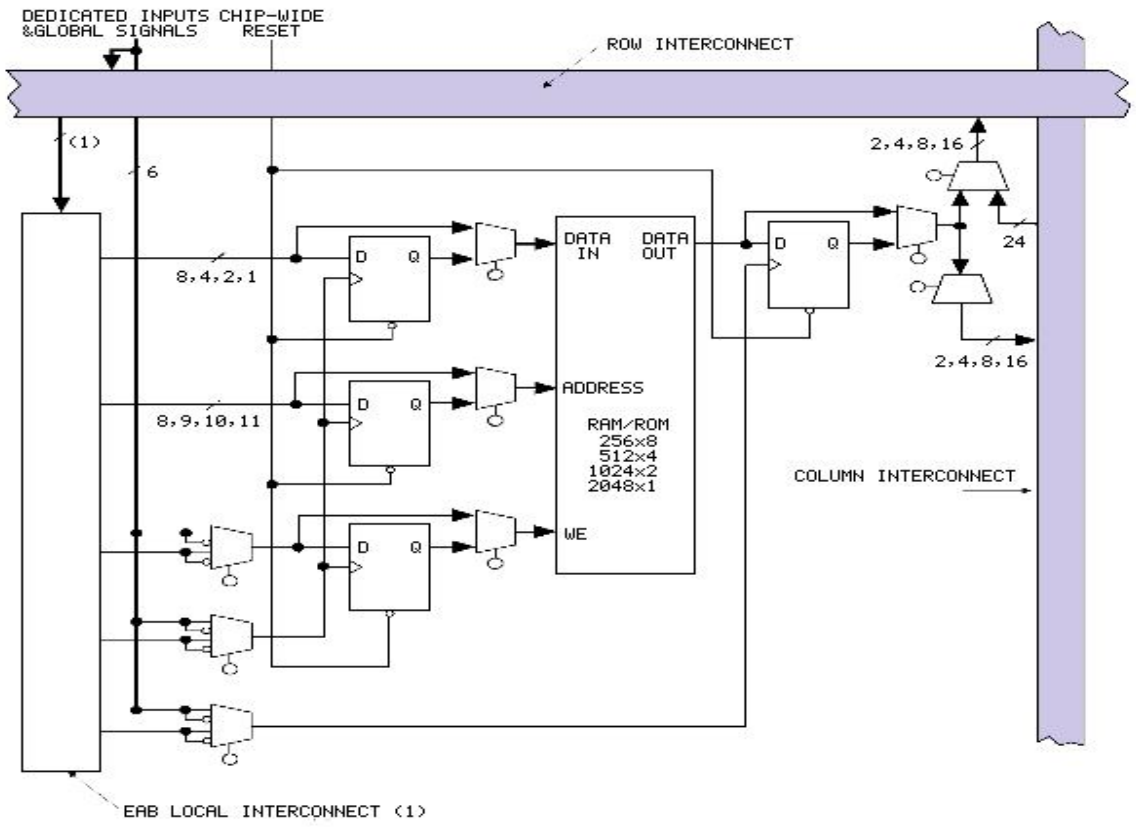


Figura 1.7 Estructura de un EAB [2]

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

CAPÍTULO II

Lenguaje de Descripción de Hardware VHDL

2.1 Generalidades de los lenguajes HDL

Un lenguaje HDL (Hardware Description Language) es cualquier lenguaje para la descripción formal de circuitos electrónicos. Puede describir tanto la operación del circuito como su diseño y organización.

A diferencia de los lenguajes de programación de software, los HDLs incluyen notación explícita para expresar tiempo y concurrencia, que son los principales atributos del hardware.

Los HDLs son usados para escribir especificaciones ejecutables de una parte de hardware. Junto con un programa de simulación, se provee al diseñador la capacidad de verificar en tiempo real el comportamiento de un diseño previo a ser implementado físicamente.

2.2 Herramientas EDA para la síntesis y el análisis en lenguaje VHDL

Las herramientas de diseño electrónico automatizado EDA (Electronic Design Automation) son todas aquellas que sirven para el diseño y síntesis de un sistema electrónico.

Las herramientas EDA surgen de la necesidad de reducir el costo y el tiempo en el ciclo de diseño de un circuito electrónico. Mediante una herramienta EDA se tiene la posibilidad de simular los sistemas diseñados, evitando así la fabricación de diversos prototipos, con lo que se logra un ciclo de diseño más eficiente y mucho menos costoso.

Existen varias herramientas EDA para la síntesis, implementación y simulación de circuitos diseñados en lenguaje VHDL. Algunas de estas herramientas son ofrecidas como parte de los programas del fabricante del dispositivo lógico programable. Un ejemplo es la plataforma de diseño Quartus II de la compañía Altera, la cual permite la síntesis de códigos en lenguajes VHDL, VERILOG y AHDL en dispositivos lógicos programables CPLDs y FPGAs fabricados por la misma compañía.

El flujo de diseño para estos dispositivos se muestra en la figura 2.1

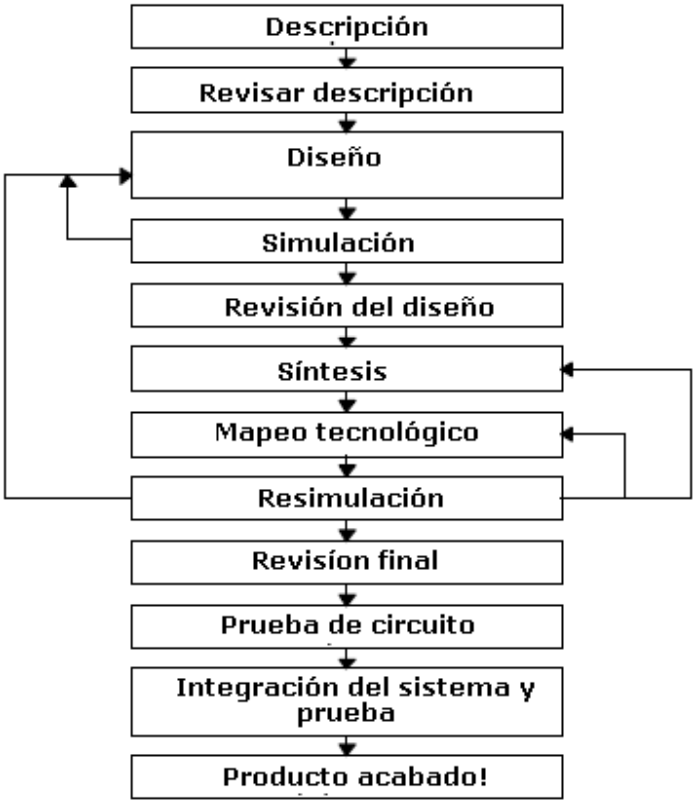


Figura 2.1 Flujo de diseño [3]

Este proyecto de tesis fue diseñado y simulado mediante la plataforma de desarrollo Quartus II e implementado en un CPLD de la compañía Altera.

2.3 Fundamentos del lenguaje VHDL

VHDL (Very high speed Hardware Description Language), por sus siglas en inglés, es un lenguaje de descripción de hardware que como cualquier HDL describe el comportamiento de circuitos o sistemas electrónicos para que luego éstos puedan ser implementados físicamente.

[3] Bob Zeidman, *Introduction to CPLD and FPGA design*, The Chalkboard Network

VHDL esta diseñado para síntesis de circuitos, sin embargo no todas las declaraciones pertenecientes a VHDL pueden ser implementadas físicamente en un circuito.

Un aspecto muy relevante del lenguaje VHDL es que a diferencia de los programas de computadora, los cuales son ejecutados en forma secuencial, en VHDL las instrucciones pueden ser ejecutadas de forma paralela. Por tal motivo VHDL es referido más como un código que como un programa.

Dentro de VHDL, sólo declaraciones puestas dentro de un "PROCESS", "FUNCTION" o "PROCEDURE" son ejecutadas de forma secuencial.

Un código en VHDL esta formado por 3 unidades principales, las cuales son: las bibliotecas, una entidad y una arquitectura. A continuación son descritas.

- Biblioteca

  Una biblioteca es una colección de partes de código usadas comúnmente. Poner dichas partes dentro de una librería les permiten ser reutilizadas o compartidas por otros diseños.

- Entidad

  Una entidad es la lista con la especificación de todas las entradas y salidas del circuito

- Arquitectura

  Una arquitectura es la descripción de cómo el circuito debe comportarse. La arquitectura esta formada por dos partes, una parte declarativa opcional, donde señales y constantes son declaradas y la parte de código. Dentro de un programa en VHDL el nombre de la arquitectura puede ser cualquiera, incluyendo el nombre de la entidad.

La figura 2.2 muestra la forma en que se estructuran las unidades fundamentales del lenguaje VHDL.

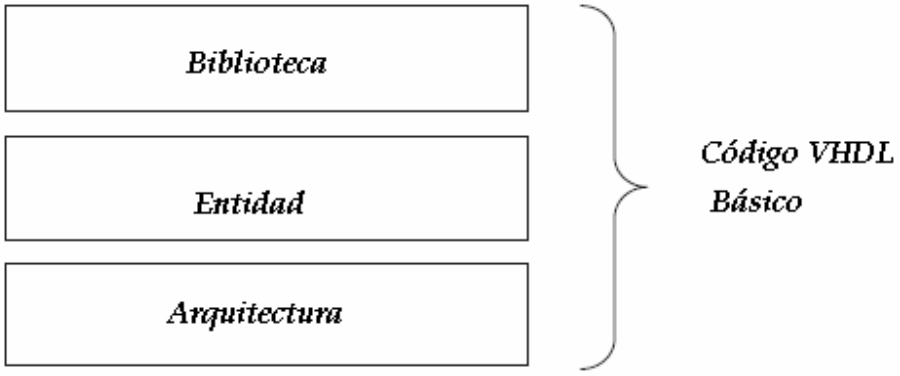


Figura 2.2 Unidades fundamentales de VHDL [1]

2.4 Tipos de datos permitidos en lenguaje VHDL

Al escribir un código VHDL es esencial conocer que tipo de datos son permitidos y como especificarlos en el diseño.

- Tipo de datos predefinidos

VHDL contiene una serie de datos predefinidos especificados en los estándares IEEE 1076 e IEEE 1164. Más específicamente, las definiciones de este tipo de datos son encontradas en las siguientes librerías:
Paquete estándar de la biblioteca std: define BIT, BOOLEAN, INTEGER y REAL
Paquete STD_LOGIC_1164 de la biblioteca IEEE: Define STD_LOGIC y STD_ULOGIC.
Entre otras bibliotecas.
Algunos de los tipos de datos más importantes son descritos a continuación:
BIT y BIT_VECTOR cuyo nivel solo puede ser "1" o "0"
STD_LOGIC (y STD_LOGIC_VECTOR): Sistema de 8 valores lógicos introducidos en el estándar IEEE 1164.

'X' Desconocido forzado
'0' Bajo forzado
'1' Alto forzado
'Z' alta impedancia
'W' Desconocido débil
'L' Bajo débil
'H' Alto débil

[1] Pedroni, Volnei A, *Circuit design with VHDL*, Massachusetts Institute of Technology, The MIT Press; illustrated edition (August, 2004)

‘–’ No importa
_ BOOLEAN: Verdadero o falso.
_ INTEGER: entero de 32-bit (desde -2,147,483,647 hasta +2,147,483,647).
_ NATURAL: Entero no negativo (desde 0 hasta +2,147,483,647).
_ REAL: Número real desde -1.0E38 hasta +1.0E38. No sintetizable.

- Tipo de datos definidos por el usuario

VHDL también permite al usuario definir sus propios tipos de datos. Existen dos categorías de tipo de datos definidos por el usuario: entero y enumerado.

Datos definidos por el usuario del tipo entero:
TYPE integer IS RANGE -2147483647 TO +2147483647;

-- El anterior es de hecho el tipo de dato predefinido para un entero
TYPE natural IS RANGE 0 TO +2147483647;

-- También es el tipo de dato predefinido para un entero sin signo.
TYPE my_integer IS RANGE -32 TO 32;
--Un tipo de dato definido que está conformado por un subconjunto de enteros
TYPE student_grade IS RANGE 0 TO 100;
--Un tipo de dato definido que es un subconjunto de enteros naturales.

- Datos definidos por el usuario del tipo enumerados:

TYPE my_logic IS ('0', '1', 'Z');
--Un tipo de datos definido por el usuario que es un subconjunto del tipo de dato STD_LOGIC
TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF BIT;
-- Este es el tipo de dato predefinido para el tipo de dato BIT_VECTOR
-- NATURAL RANGE <>, indica que la única restricción es que el rango debe de caer dentro del rango NATURAL

2.5 Señales, variables y constantes

Las constantes y las señales pueden ser globales, esto es, que pueden ser visibles para todo el código completo. Por otro lado una variable es únicamente local ya que solo es usada dentro de una pieza de código secuencial y además nunca es sacada directamente.

- Constantes

  Una constante sirve para establecer valores fijos. Su sintaxis se muestra a continuación:
  CONSTANT nombre_de_la_constante: tipo:=valor;
  Una constante puede ser declarada dentro de un "PACKAGE","ENTITY", o "ARCHITECTURE". Cuando es declarada dentro de un "PACKAGE" se convierte totalmente global, ya que un paquete puede ser usado por diferentes entidades.
  Si la constante es declarada dentro de una "ENTITY", entonces es global para todas las arquitecturas que siguen de la entidad. Finalmente si es declarada dentro de una arquitectura, entonces es visible únicamente dentro del código de dicha arquitectura.

- Señales

  Una señal sirve para asignar valores dentro y fuera del circuito así como dentro de unidades internas del circuito. Visto de otra manera las señales representan conexiones dentro del circuito.
  La declaración de una señal puede ser hecha en los mismos lugares que la declaración de una constante, tal como ya se ha descrito.

  Un aspecto relevante acerca de las señales es que, cuando éstas son declaradas dentro de una sección de código secuencial, se actualizará hasta que termine de ejecutar la sección de código donde se encuentra.

- Variables

  A diferencia de una constante o una señal, una variable representa únicamente información local. Solo puede ser usada dentro de secciones de código secuencial y su valor no puede ser sacado de dicho código directamente. En contraste con las señales, las variables son actualizadas inmediatamente, de tal forma que pueden ser usadas en la siguiente línea de código después de haberles asignado un valor.

  La sintaxis para declarar una variable es la siguiente:
  VARIABLE nombre_de_la_variable : tipo [rango][:= valor_inicial]
  Cabe mencionar que una variable solo puede ser declarada dentro de la sección declarativa de un "PROCESS", "PROCEDURE" o "FUNCTION".

2.6 Ejecución concurrente y ejecución secuencial

Existen dos formas distintas en la ejecución de un código en VHDL. Para poder entender de manera adecuada la diferencia entre un código concurrente y uno secuencial es importante recordar la diferencia entre la lógica combinacional y la lógica secuencial.

La lógica combinacional es aquella en que la salida depende únicamente de las entradas y por tanto no requiere de la información pasada y puede ser implementada únicamente con compuertas convencionales. En contraste, la lógica secuencial es aquella en que las salidas dependen de entradas anteriores, por lo se requieren que elementos de almacenamiento, los cuales están a su vez conectados a la lógica combinacional a través de una realimentación.

- Ejecución concurrente

  Las declaraciones concurrentes en VHDL son "WHEN" y "GENERATE", a lado de ellas las operaciones AND, NOT, etc pueden ser usadas también para generar código concurrente.

  Un código VHDL es inherentemente concurrente. Únicamente declaraciones puestas dentro de un "PROCESS, "FUNCTION" o "PROCEDURE" son llevadas a cabo de manera secuencial. Aún cuando las instrucciones llevadas a cabo dentro de ellos son secuenciales, cada uno de los "PROCESS", "FUNCTION" o "PROCEDURE" son ejecutados de forma concurrente entre sí.

  De forma resumida, cuando se emplea código concurrente, los siguientes operadores pueden ser empleados:

  _ WHEN
  _ GENERATE
  _ BLOCK

- Ejecución secuencial

  En VHDL, los "PROCESS", "FUNCTION" y "PROCEDURE" forman secciones de código que son ejecutadas de forma secuencial. Un aspecto importante de un código secuencial es que éste no se limita a contener únicamente lógica secuencial. Con un código de este tipo se pueden construir circuitos secuenciales así como circuitos combinacionales.
  Cabe mencionar que las siguientes declaraciones, todas secuenciales, son permitidas únicamente dentro de los "PROCESS", "FUNCTION" o "PROCEDURE": IF, WAIT, CASE, y LOOP.

Las variables también están restringidas a ser usadas únicamente dentro de un código secuencial. Así, contrario a una señal, una variable nunca puede ser global.

CAPÍTULO III

Transductores

3.1 Definición de transductor

Un transductor es un dispositivo, por lo general electrónico, eléctrico, electro-mecánicos, electromagnéticos, fotónicos, o fotovoltaico que convierte un tipo de energía o atributo físico a otro para diversos fines, entre ellos la medición o la transferencia de información.

3.2 Transductor de distancia

Un transductor de distancia, en esencia, es un dispositivo que transforma la distancia existente entre éste y un objeto en un voltaje proporcional a esa distancia.
A continuación en este capitulo se describe el funcionamiento de un transductor de este tipo con el que se equipó al androide para éste sea capaz de reconocer un obstáculo.

3.3 Sensor de distancia SRF-05

Se trata de un sensor para la medición de distancia compuesto de una combinación integrada de un modulo ultrasónico y un circuito de procesamiento de señales.

Basado en el eco que produce una señal de baja frecuencia (40kHz) que es emitida en un tren de pulsos de 8 ciclos, en el cono de aceptación, el dispositivo requiere entonces trabajar con dos señales, una de transmisión y otra de recepción. La señal de trasmisión que activa ese tren de pulsos es un nivel alto de duración mínima de 10µs. La señal de recepción se lleva a un circuito con un procesamiento digital de señales que la transforma a un pulso de duración proporcional a la distancia del eco. Por lo tanto, este sensor se puede utilizar también como un sensor de proximidad.

La figura 3.1 muestra el esquema de tiempo del dispositivo. Cabe mencionar que este dispositivo, dada la característica de la señal emitida requiere de un retardo considerable, durante el cual se procesa el eco. Esto traducido a tiempo correspondiente a un retardo de aproximadamente 50ms, entre la señal de transmisión y de recepción.

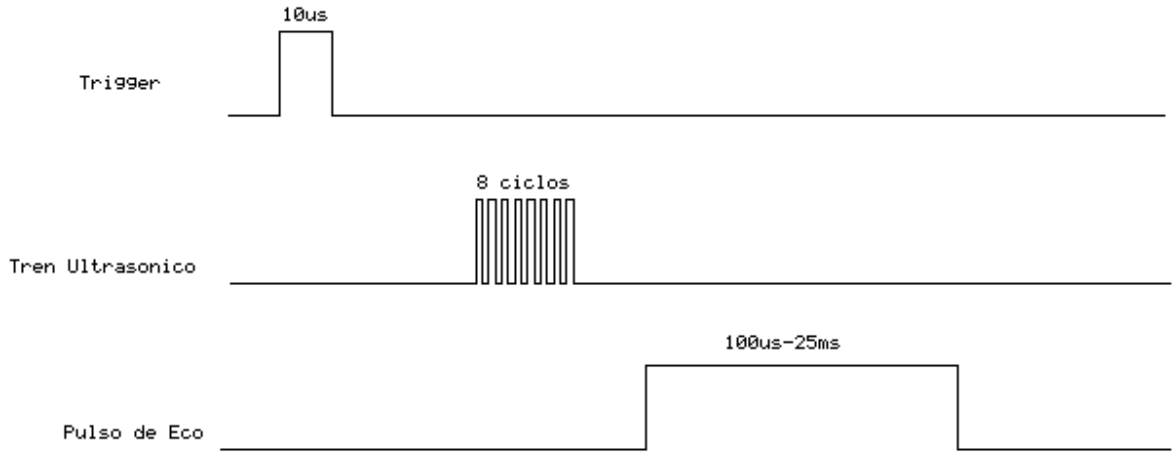


Figura 3.1 Esquema de tiempos del sensor ultrasónico †

La figura 3.2 muestra el patrón de radiación del dispositivo, típicamente puede detectar un objeto a una distancia máxima de 4m.

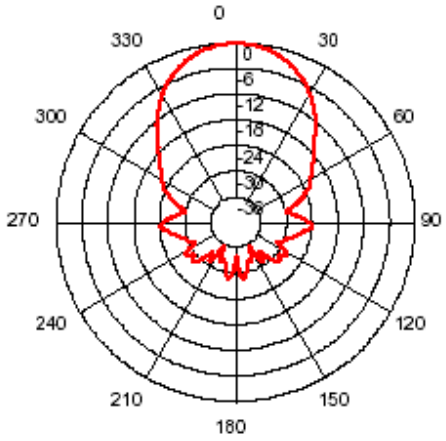


Figura 3.2 Patrón de radiación †

En la figura 3.3 se muestra el dispositivo SRF-05 empleado para el reconocimiento del obstáculo.

† Para mas información consultar hoja de especificaciones del dispositivo SRF-05

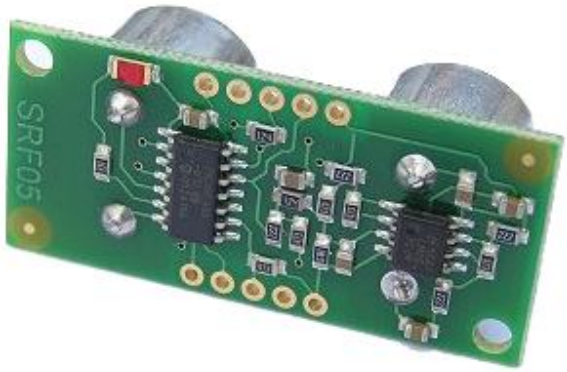


Figura 3.3 Sensor de distancia SRF-05

CAPÍTULO IV

Flujo de Diseño

En esta sección se analiza el flujo de diseño para cualquier dispositivo lógico programable llámese ASIC, CPLD o FPGA. Éste es el proceso que permite diseñar un dispositivo que trabaje de manera adecuada dentro de un sistema. A continuación se describen los diferentes pasos dentro del flujo de diseño.

4.1 Especificaciones del diseño

La especificación del diseño es muy importante, ya que es una guía para elegir de forma adecuada la tecnología que cumpla las necesidades planteadas.
La especificación del diseño debe de incluir lo siguiente:

- Un diagrama de bloques externo que indique como el circuito integrado trabaja dentro del sistema
- Un diagrama de bloques interno mostrando las principales partes del diseño dentro del circuito integrado
- Una descripción de cada una de las terminales de entrada y salida incluyendo capacidad en el manejo de corriente de cada una de ellas
- Tiempo de propagación para las terminales de salida
- Frecuencia del reloj principal
- Uso estimado de elementos lógicos o macroceldas
- Consumo de potencia del circuito
- Costo del dispositivo seleccionado
- Procedimientos de prueba del diseño final

4.2 Elección de un dispositivo capaz de contener el diseño realizado

Una vez que se tiene una descripción del diseño, ésta puede ser usada para encontrar el mejor dispositivo que se adapte a las necesidades del proyecto.

4.3 Elección del método de captura del diseño

En este punto se debe decidir el método que se va a elegir para capturar el diseño dentro del dispositivo lógico programable. Para diseños pequeños se utiliza generalmente un editor esquemático, sin embargo en diseños mucho más grandes como el realizado en este

proyecto de tesis se utiliza comúnmente un lenguaje de descripción de hardware como VHDL, el cual se ha descrito previamente en el capítulo II.

Cuando se emplea algún lenguaje de alto nivel, como lo es VHDL, siempre es necesario un software que sintetice el diseño realizado, lo que significa que el software debe generar el diseño al nivel mas bajo dentro del dispositivo, partiendo del código descrito en lenguaje de alto nivel.

4.4 Metodologías de diseño

A continuación se describen los dos tipos de metodologías que existen y que pueden ser usados en cualquier diseño con éste tipo de dispositivos.

- Metodología de diseño BOTTOM-UP

  La metodología de diseño de abajo hacia arriba consiste en primero describir componentes pequeños de un sistema los cuales más tarde se pretenden agrupar hasta formar un solo sistema.

  En general esta forma de diseñar no es muy buena ya que es un método de diseño bastante ineficiente. Para diseños muy grandes es muy difícil unir una cantidad muy grande de componentes a un bajo nivel y esperar que en conjunto funcionen de manera adecuada. En el caso de un diseño con un dispositivo lógico programable empleando este tipo de metodología es más difícil detectar fallas o anomalías dentro del diseño. Asimismo la probabilidad de cometer errores al diseñar cada uno de los elementos que conforman al sistema es mucho mayor que con otro tipo de metodología de diseño. La figura 4.1 muestra el esquema de éste tipo de metodología de diseño.

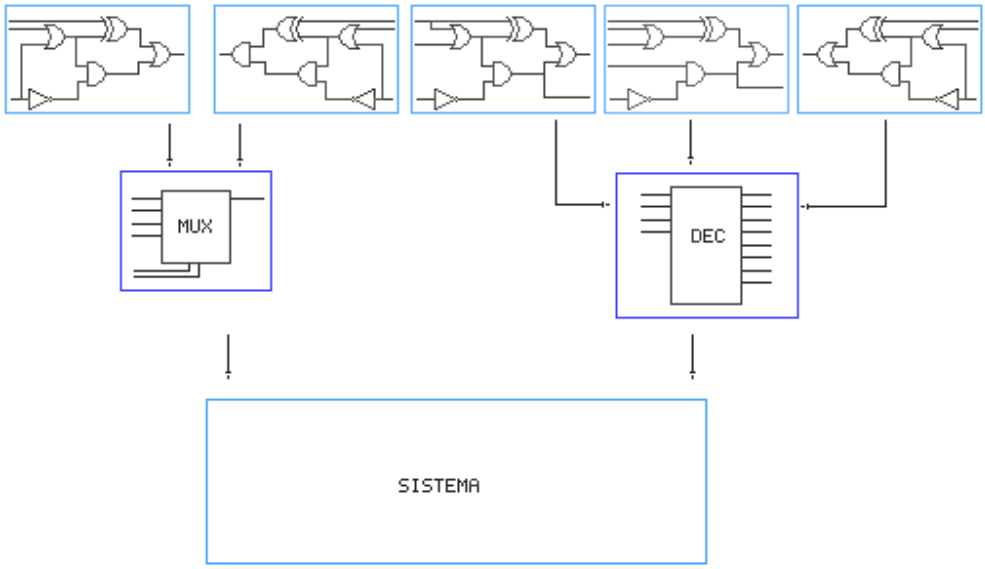


Figura 4.1 Metodología de diseño Bottom-Up [3]

- Metodología de diseño TOP-DOWN

  La metodología de diseño TOP-DOWN consiste esencialmente en dividir las partes que componen un sistema con la finalidad de tener una mejor comprensión en cada una de dichas partes. Con ésta metodología de diseño primero se formula una idea general del sistema y las principales partes que lo van a conformar pero sin detallar cómo cada una de éstas deben funcionar. Posteriormente cada parte del sistema se describe a detalle hasta que el diseño completo halla sido descrito.

  Un modelo descrito con este tipo de metodología a menudo tiene sus diferentes partes descritas mediante cajas negras con la finalidad de manipular más fácilmente el diseño.

4.5 Metodología de diseño TOP-DOWN utilizando VHDL

La metodología de diseño Top-down es un método de diseño en donde primero son definidas funciones de alto nivel y posteriormente se detallan implementaciones de bajo nivel. Un esquema de éste tipo de diseño puede ser visto como un árbol jerárquico. El bloque mas alto representa el CHIP completo.

[3] Bob Zeidman, *Introduction to CPLD and FPGA design*, The Chalkboard Network

Los primeros bloques por debajo del mas alto representan mayores funciones del chip. Bloques intermedios contienen cada vez funcionalidad más pequeña incluso a nivel de lógica con compuertas. El nivel más bajo contiene únicamente lógica a nivel de compuertas y macro funciones que pueden ser proporcionadas por el fabricante.

Afortunadamente los lenguajes HDL's permiten la descripción de diseños con esta metodología. La figura 4.2 muestra el esquema de éste tipo de metodología de diseño.

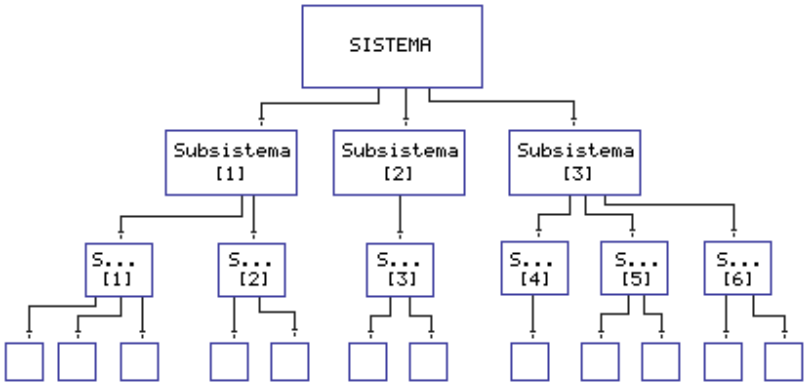


Figura 4.2 Metodología de diseño Top-Down [3]

Es importante decir que éste tipo de metodología es preferida para el diseño de circuitos integrados digitales por varias razones. Primero, este tipo de chips incorpora con bastante frecuencia una gran cantidad de compuertas para lograr un nivel muy alto de funcionalidad. Ésta metodología simplifica la tarea de diseñar y permite la participación de más de un ingeniero en el diseño del circuito. Segundo, se tiene una mayor flexibilidad en el diseño. Secciones del diseño pueden ser removidas y remplazadas con diseños que ofrezcan un mayor desempeño sin afectar otras secciones.

Otro aspecto importante es que la simulación se simplifica bastante empleando éste tipo de metodología.

La simulación es un aspecto muy importante en el diseño de un chip, ya que el circuito una vez fabricado no puede ser modificado. Por tal motivo, la simulación debe ser llevada a cabo extensivamente en un chip que se planea producir en serie.

La metodología de diseño Top-down permite la simulación independiente de cada uno de los módulos que conforman al diseño. Lo anterior es una ventaja sobre todo en diseños muy complejos donde la simulación completa puede tardar, días inclusive semanas.

[3] Bob Zeidman, *Introduction to CPLD and FPGA design*, The Chalkboard Network

CAPÍTULO V

Diseño y Construcción del Control para el Androide

En este capítulo se describe el diseño en lenguaje VHDL utilizado para el manejo del CPLD que controla los movimientos del androide y los componentes de hardware que interactúan con él.

5.0 El androide

Para el proyecto se utilizo un androide ROBONOVA-I, el cual cuenta con 16 servomotores digitales que pueden trabajar en diversos tipos de operación. Para este proyecto de tesis se empleara el modo de control PWM unidireccional, que más adelante será descrito detalladamente. En la figura 5.0 se muestra el androide utilizado.

Figura 5.0 ROBONOVA-I

5.1 Especificaciones del control del androide

Se pretende controlar mediante un CPLD un androide que cuenta con 16 servomotores digitales a los cuales se les controla su posición mediante modulación por ancho de pulsos. Asimismo se pretende contar con un sensor de distancia el cual permita que el androide de la vuelta y regrese al encontrarse con un obstáculo que le impida el paso.

A continuación se muestra una comparación sobre dos tarjetas de desarrollo de dos fabricantes distintos (Altera y Xilinx) las cuales se presentaron como opciones para desarrollar el control del androide y se explica porque se eligió la tarjeta de desarrollo de Altera.

Dos opciones que se presentaron para el desarrollo de éste control fueron las tarjetas de desarrollo "CoolRunner II Starter KIT" de la compañía Xilinx y la tarjeta "MAX II micro board de Altera". En la tabla 5.1 se realiza una comparación sobre la capacidad del CPLD con que cuentan cada una de ellas.

Fabricante	Macroceldas disponibles	Pines de entrada/salida	Frecuencia de operación máxima
Xilinx XC2C256	256	118	256 MHz
Altera EPM2210	1700	204	304 MHz

Tabla 5.1 Comparación de CPLD's de Xilinx y Altera

De la tabla anterior es posible observar que la capacidad del dispositivo lógico programable complejo con el que cuenta la tarjeta de altera es muy superior al de su competidor. En lo que se refiere al costo de ambas tarjetas de desarrollo la de Xilinx tiene un costo aproximado de $39 USD mientras que la tarjeta de Altera de $69 USD.

Como se verá hacia el final de esta tesis el diseño realizado para el control del androide demando un total de 1792 elementos lógicos lo que equivale a 1378 macroceldas. Por tal motivo, a pesar de que la tarjeta de Altera es mas costosa no hubo mas opción que optar por ella debido a su gran capacidad en comparación con la tarjeta de Xilinx la cual hubiera sido insuficiente para contener el diseño del control del androide.

En la figura 5.1 se muestra la tarjeta de desarrollo elegida de la cual es posible observar que se cuenta con dos CPLD´s. Uno de ellos contiene un programa que permite la programación mediante el puerto USB, y el en el otro se albergan los diseños realizados por el usuario.

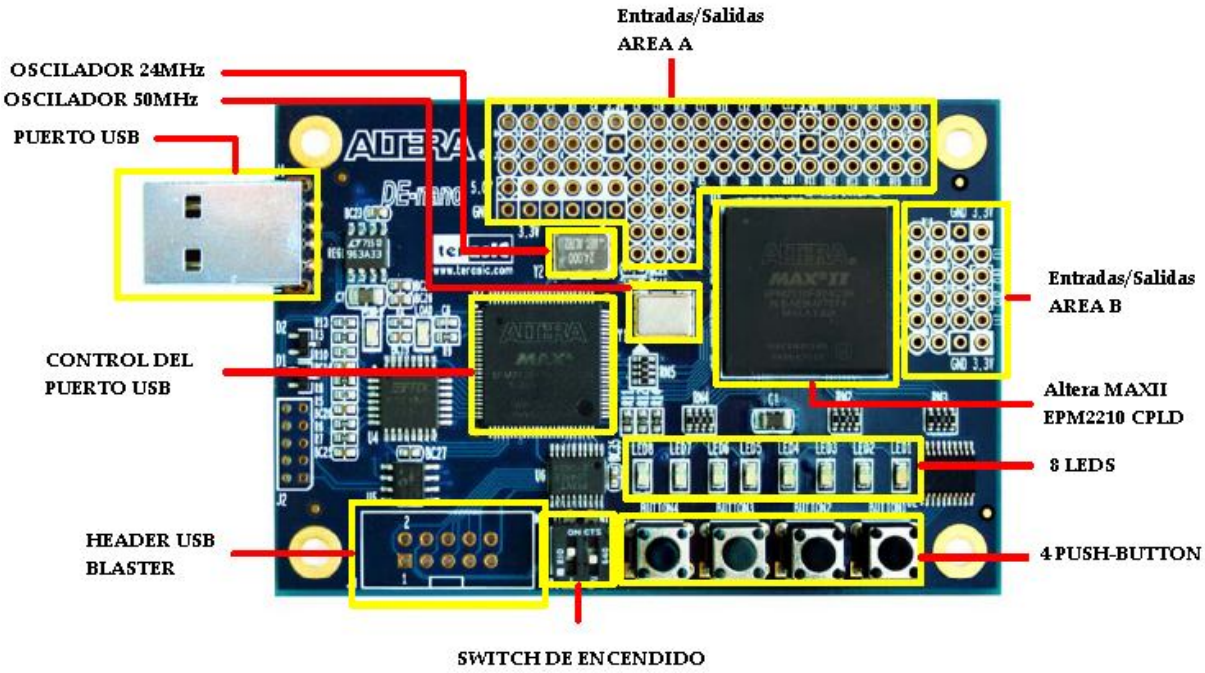


Figura 5.1 Tarjeta MAX II micro board

5.2 Diseño del control del androide y de los distintos circuitos que interactúan con él

Para lograr el diseño adecuado de este control se realizaron dos diagramas de bloques uno externo y otro interno.

- Diagrama de bloques externo

  En el diagrama de bloques externo se representan los diferentes elementos físicos que forman parte del diseño del control del androide en el cual se utilizo como entradas del sistema un modulo de radio frecuencia, un sensor de distancia y un oscilador, cada uno de estos bloques se describen a continuación en el capitulo así como los tres bloques de salida del sistema conformados por los servomotores y los leds. La figura 5.2 muestra el diagrama de bloques externo del control.

Figura 5.2 Diagrama de Bloques Externo

- Diagrama de bloques interno

Para empezar a escribir código en lenguaje VHDL es necesario reconocer ciertos elementos con los que debe contar el circuito digital encargado de controlar los 16 servomotores que mueven al esqueleto del robot y uno adicional para el control de la cabeza. Uno de los principales elementos a considerar dentro del CPLD es un divisor de frecuencia para el manejo interno del reloj principal externo en la tarjeta el cual oscila a una frecuencia de 50MHz. Dado que a los servomotores digitales con los que cuenta el robot se les controla con pulsos que pueden variar entre los 600 y 2400 microsegundos de duración en alto, se optó por dividir el reloj principal de 50MHz en uno de 200kHz.
Como es evidente una señal de reloj de 200kHz tiene un período de 5μs lo que significa que se puede dividir el intervalo de movimiento de cada uno de los servomotores que va de -95° a 95° en aproximadamente 360 posiciones diferentes.

El mismo reloj de 200kHz se empleó para los diferentes elementos del circuito programado ya que para efectos de éste control se requieren contadores que realicen retardos en diferentes secuencias del orden de los segundos. La figura 5.3 muestra los elementos requeridos en el diagrama de bloques interno. Respecto a los elementos que forman parte de este diagrama su funcionamiento se explica a lo largo de este capítulo.

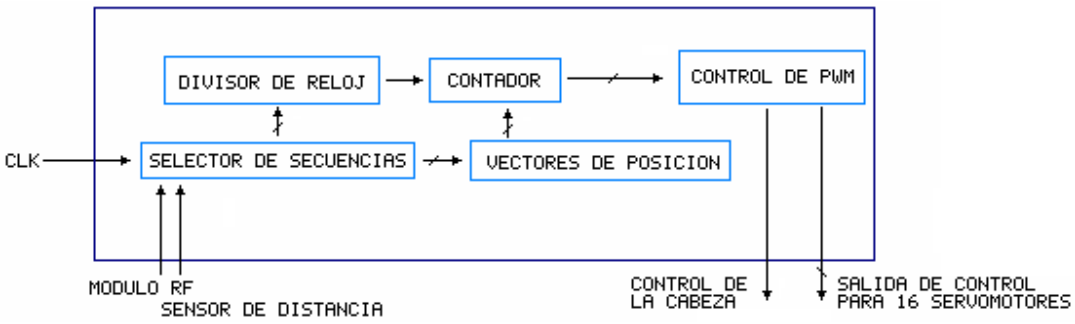


Figura 5.3 Diagrama de bloques interno

5.3 Modulación por ancho de pulsos PWM

La modulación de ancho de pulsos es una técnica en la que se modifica el ciclo de trabajo de una señal periódica.

En señales digitales, el ciclo de trabajo de una señal periódica se define como la relación que guarda el tiempo activo de la señal (un nivel alto) con el periodo de la misma; esto se muestra en la ecuación 5.1.

$$D = \frac{t}{T} \qquad \text{…… Ecuación número 5.1}$$

Donde:
D: ciclo de trabajo
t: tiempo activo de la señal
T: periodo de la señal

Ejemplos con esta técnica de control sobre las señales periódicas, son un canal de comunicación con el que se puede establecer un control sobre el tiempo en la lógica de transmisión y la cantidad de energía que se entrega a un sistema. En la figura 5.4 se muestra el control del ciclo de trabajo de una señal.

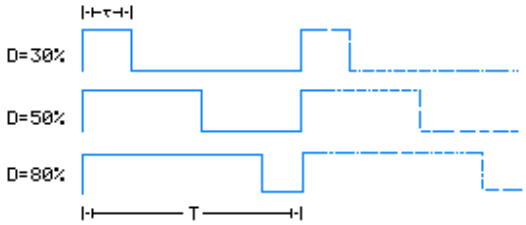


Figura 5.4 Control sobre el ciclo de trabajo

El tener control sobre el ciclo de trabajo, puede ser un parámetro para evaluar si contabilizamos el tiempo que la señal tiene un nivel alto respecto a su nivel bajo o su periodo.

Las acciones entonces pueden ser determinadas por la comparación de la duración de la cuenta de manera muy precisa, pero se debe de considerar que puede existir falta de exactitud en el dispositivo, como la frecuencia de su reloj interno, el ruido en las señales, etc. Lo cual obliga entonces a establecer un intervalo de valores para los contadores, a fin de garantizar que señales que se tienen siempre sean las deseadas.

5.4 Teoría básica de control de servomotores mediante PWM

Un servomotor es básicamente un sistema de control realimentado. Consta de un motor de de DC con engranes, una etapa de potencia y un circuito electrónico que se encarga del control de la dirección de rotación y de la posición del motor.
El control del motor emplea señales moduladas con técnica PWM la cual ha sido descrita previamente.
A diferencia de un motor de DC convencional en un servomotor la señal de control de PWM no se utiliza para controlar la velocidad del mismo sino para controlar la dirección de rotación y la posición.
Dado que se trata de un sistema con realimentación dentro del servomotor se cuenta con un codificador que informa al control del servo (usualmente un microcontrolador) la posición actual del motor de DC. Posteriormente se comprara esta señal con la señal PWM de entrada al servo y el control del servo envía los pulsos al driver del motor para que éste ultimo lo mueva a la posición deseada.

En la figura 5.5 se muestra un diagrama básico de los diferentes elementos que conforman un servomotor.

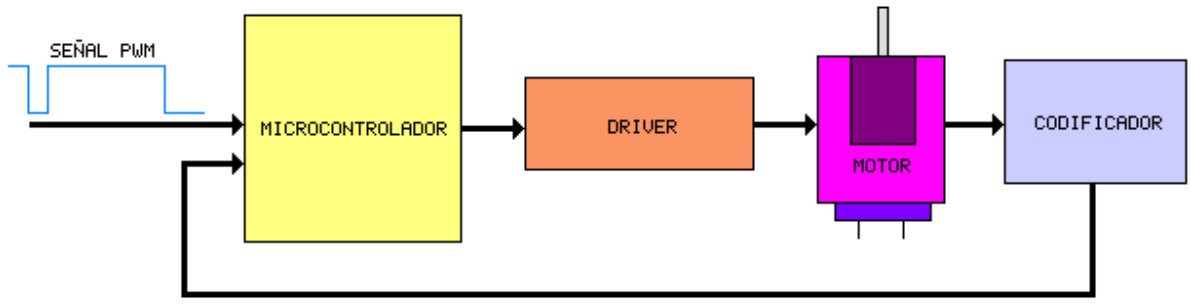


Figura 5.5 Elementos básicos de un servomotor

Cada uno de los servomotores con los que cuenta el androide Robonova-I tienen como circuito de control un microprocesador Atmel ATmega8 el cual se encarga de recibir la señal de control enviada por el usuario y generar los pulsos que son enviados a un driver MOSFET en configuración puente que se encarga de mover el motor a la posición deseada.

Debido a que los servos del androide ya cuentan con un driver y un circuito de control en el diseño del control del androide solo nos preocuparemos por generar las señales de control de PWM que nos permitan, de acuerdo a las hojas de especificaciones de los servomotores empleados, moverlos a las posiciones y velocidades deseadas y las cuales además pueden ser enviadas directamente del CPLD a cada uno de los servos. La generación de dichas señales de control se detalla a continuación.

5.5 Control por modulación de ancho de pulso PWM

A continuación se describe el control de los 16 servomotores, éstos pueden ser controlados todos a la vez o de manera individual. Para lograr el control de la posición de cada uno de los servos se envían 16 pulsos (uno a cada servo) de ancho variable. Cada servomotor tomará una posición distinta dependiendo del ancho de pulso que se le mande. En la figura 5.6 se muestran las posiciones que pueden tomar cada uno de los servos dependiendo de la duración del pulso que se les envíe.

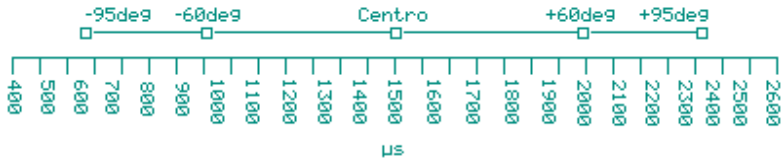


Figura 5.6 Posiciones para los servomotores dadas por la duración del pulso

Se requiere un código que envíe pulsos de distinta duración y que estos pulsos vayan cambiando de duración en el tiempo, además el código debe ser capas de incrementar o decrementar la duración de esos pulsos a distintas velocidades (no todos los servomotores se mueven a la misma velocidad). En la figura 5.7 se muestra de manera grafica el control sobre cada uno de los motores en la línea de tiempo.

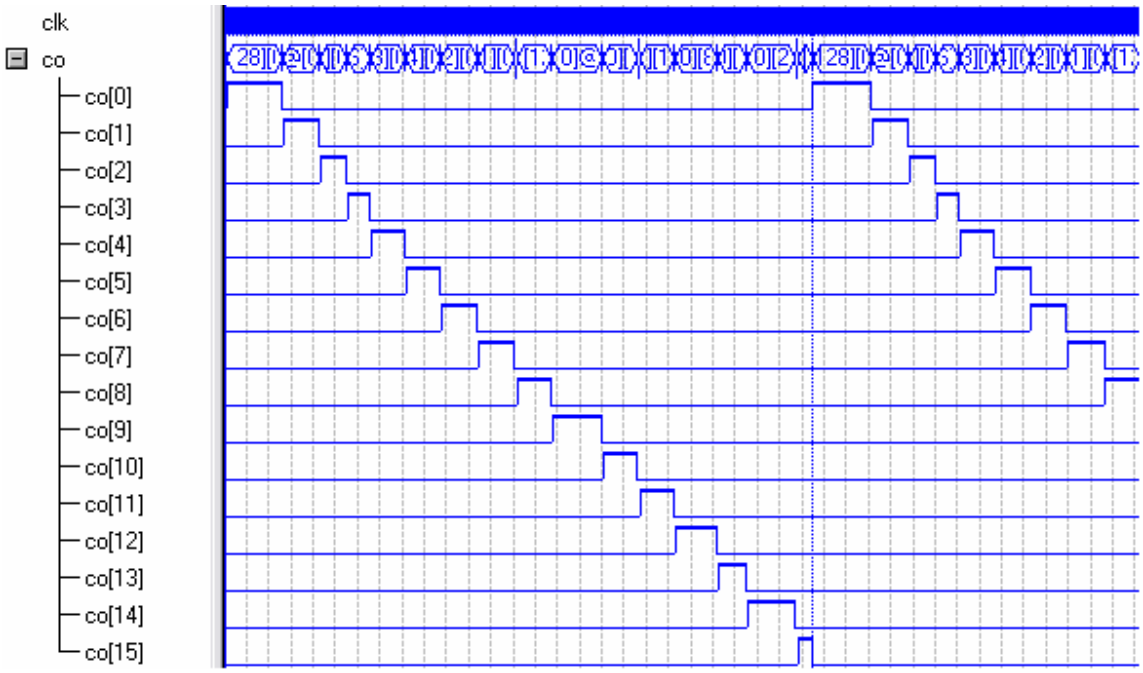


Figura 5.7 Control sobre los servomotores

5.6 Envío de las señales moduladas por PWM a cada uno de los 16 servomotores

Para lograr enviar 16 pulsos se hace un vector de 16 salidas. Los pulsos no fueron enviados de manera simultánea a cada uno de los elementos del vector de salida dado que de esta forma los servomotores en conjunto demandarían una corriente eléctrica excesiva. En ves de ello se envían los pulsos por separado, esto es, un pulso ala vez para cada uno de los elementos del vector de salida.
Para lograr lo anterior se empleo un contador, que durante el intervalo de su cuenta, una salida tenga un nivel alto que sea la duración del pulso. Un vector guarda los valores de los contadores que debe tener para que durante esa cuenta la n-esima salida se mantenga en alto.

Se requiere del manejo de tres vectores, uno para la posición inicial, otro para la posición final y por último otro para el manejo de velocidad, con lo cual se tiene el control absoluto sobre los movimientos de los servomotores. La diferencia entre la posición final y la inicial se logra cuando los incrementos o decrementos del vector de velocidad hace que esos dos vectores sean iguales y se pase a otro estado de movimiento. En la figura 5.8 se ilustra una secuencia de movimientos, donde los incrementos de los contadores son de 50µs.

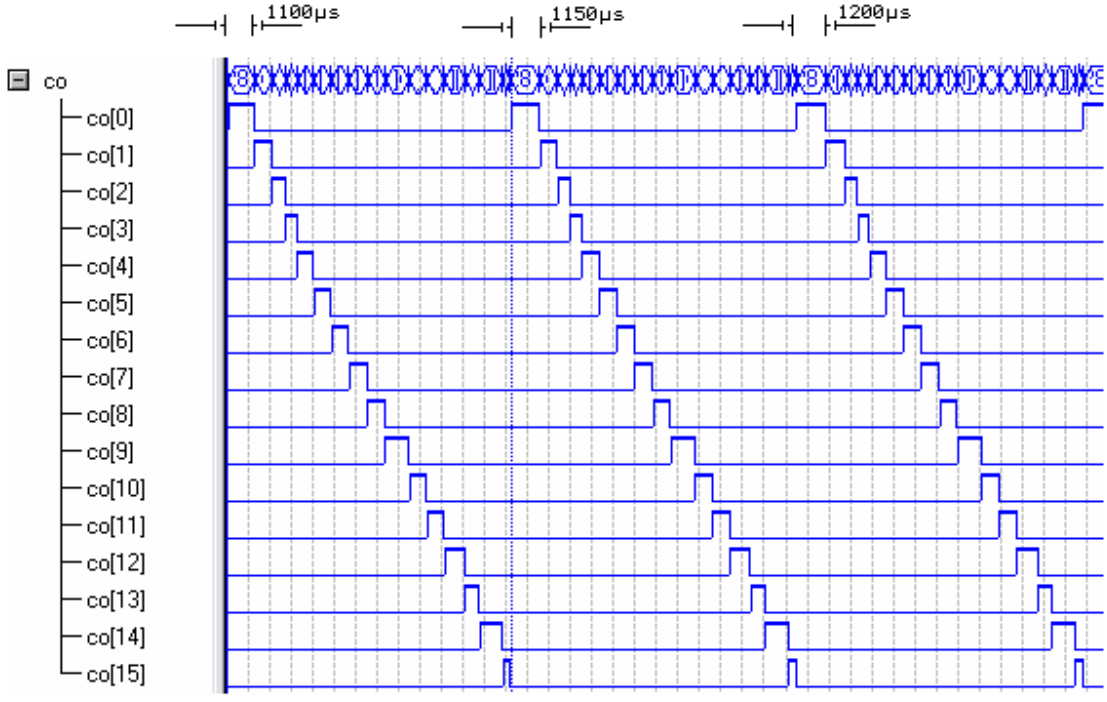


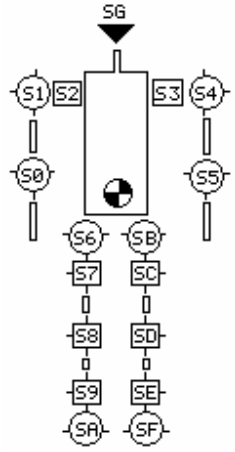
Figura 5.8 Secuencia de movimientos

5.7 Movimiento y estabilización del androide

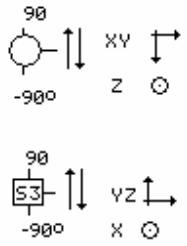
Una vez que se tiene el control sobre la posición y velocidad de rotación de cada uno de los servomotores con los que cuenta el androide es necesario que todos ellos se muevan de forma coordinada para lograr que el androide realice de forma adecuada los movimientos deseados.

Debido a que no se cuenta con algún tipo de dispositivo que permita estabilizar al androide cuando éste se encuentra de pie o cuando éste camina fue necesario observar la forma en que cada uno de los servomotores debían de moverse a fin de realizar los movimientos de forma segura y no caer al suelo.

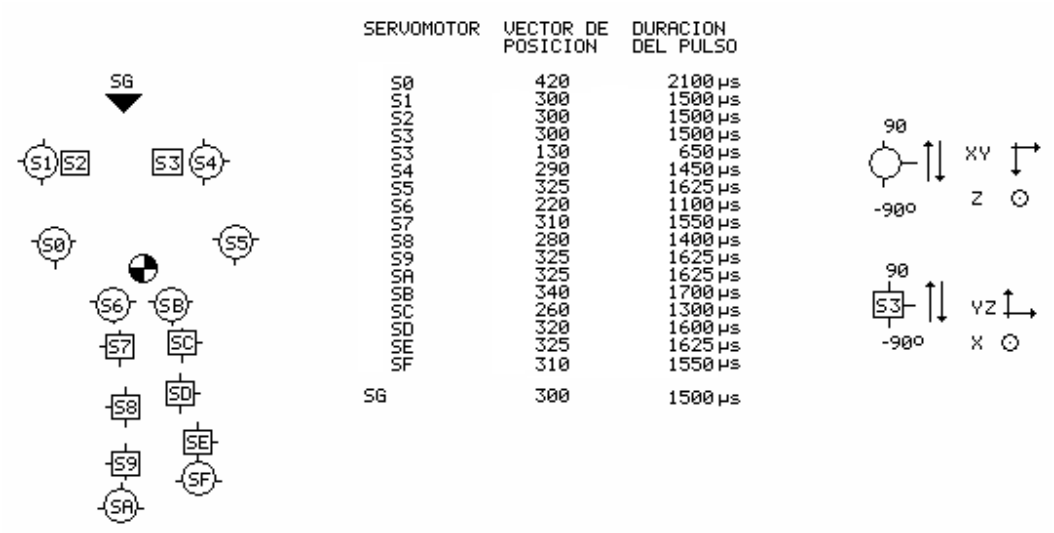
En la figura 5.9a se muestra una tabla con la duración de los pulsos que se deben enviar a cada uno de los 16 servomotores de forma continua para que éste se mantenga de pie en equilibrio. En la figura 5.9b se tiene otra tabla donde se observan los cambios necesarios para que el androide dé el primer paso en su secuencia de caminata.



SERVOMOTOR	VECTOR DE POSICION	DURACION DEL PULSO
S0	420	2100 µs
S1	360	1800 µs
S2	300	1500 µs
S3	300	1500 µs
S3	280	1400 µs
S4	240	1200 µs
S5	305	1525 µs
S6	270	1350 µs
S7	340	1700 µs
S8	420	2100 µs
S9	280	1400 µs
SA	300	1500 µs
SB	295	1475 µs
SC	320	1600 µs
SD	260	1300 µs
SE	320	1600 µs
SF	300	1500 µs
SG	300	1500 µs



a)



b)

Figura 5.9 Mapeo de valores y posiciones para cada uno de los 16 servomotores

5.8 Control sobre el sensor de distancia

Se logra que el sensor tenga como respuesta una señal llamada "ECHO" proporcional a la distancia detectada mediante el envío de una señal de disparo llamada "TRIGGER" la cual debe tener una duración en alto mayor a 10 µs para que se genere una secuencia de pulsos llamada "ULTRASONIC BURST"; una vez enviado el pulso se debe de esperar un tiempo aproximado de 50 ms para recibir un pulso proporcional a la distancia, resulta necesario entonces un contador que asigne un entero dependiendo de la duración del pulso de ECHO, y también generar el pulso del TRIGGER cada 50 ms. Dado que un ciclo de 50 ms es mucho mayor que un ciclo de 20 ns (periodo del reloj de 50 MHz) es necesario dividir varias veces las señales de reloj (CLK1,CLK2,CLK3) para emitir el pulso de 10 µs. En la figura 5.10 se muestra el pulso de disparo de 10 µs con un periodo de 50 ms.

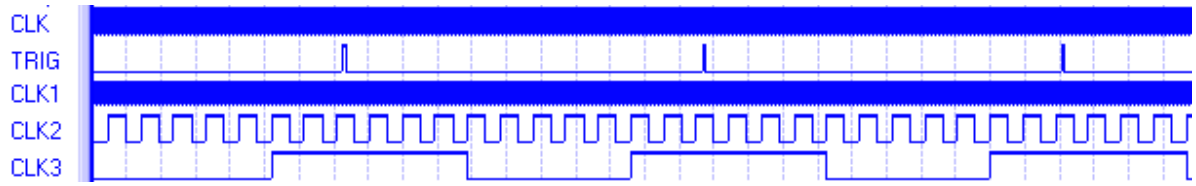


Figura 5.10 Señal de disparo

Una vez que se tiene respuesta de ECHO se inicializa el contador para determinar la longitud del pulso recibido. En la figura 5.11 se muestra la simulación de una señal cualquiera recibida en el CPLD por el sensor de distancia.

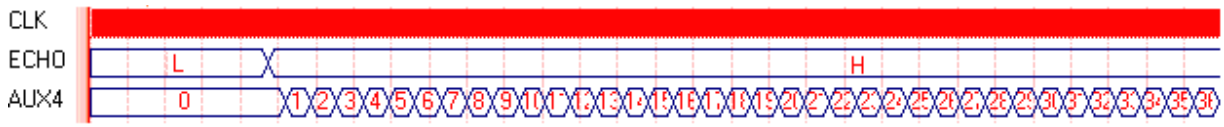


Figura 5.11 Procesamiento de la señal ECHO

Para que sea una señal útil y determinar la distancia del obstáculo se evalúa la longitud del entero del contador llamada AUX4 y si es menor de un cierto valor, se asigna un "1" o un "0" a otra señal.

5.9 Módulo de radio frecuencia

Con el objetivo de poder controlar al androide de forma remota y además con la finalidad de mostrar la capacidad del CPLD de manejar distintos procesos ejecutándose de forma independiente y paralela, se optó por integrar en el diseño del control del androide un módulo de recepción de radio frecuencia. Asimismo se adaptó un joystick a la parte transmisora del módulo de radio frecuencia mencionado con el que se comandan las instrucciones que debe realizar el androide.

Los dos bloques funcionales de este sistema de transmisión y recepción se describen a continuación.

- Módulo transmisor de radio frecuencia:

  Se emplea un joystick que genera una señal de entrada a un CPLD, la forma del manejo de esta señal para su transmisión se describe posteriormente.
  Del CPLD se obtiene una señal que es enviada al transmisor de radio frecuencia como un tren de pulsos para ser transmitida.

- Módulo receptor de radio frecuencia:

  El módulo receptor de radio frecuencia con el que se equipó al androide permite la recepción digital serial de señales provenientes del módulo transmisor.

En la figura 5.12 se muestra el módulo de RF empleado.

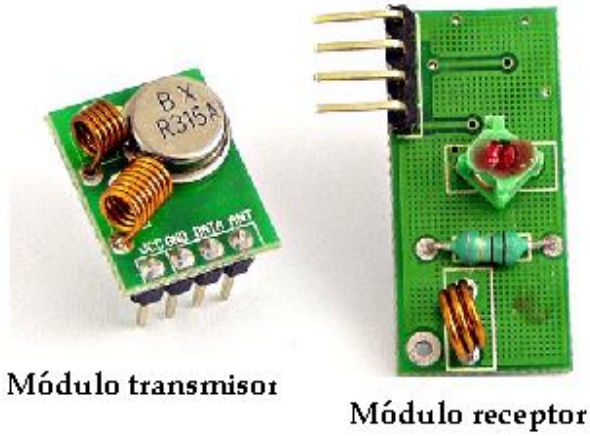


Figura 5.12 Módulo RF empleado

5.10 Descripción del modulo de radio frecuencia empleado

El modulo de radio frecuencia que se empleó cuenta con un solo canal y utiliza modulación ASK para el envío de información digital del transmisor hacia el receptor. En el transmisor y receptor no se cuenta con algún circuito que establezca alguna interfaz o protocolo para el envío de la información por lo que si se aplica un nivel alto como entrada al transmisor, éste aparece sin alteración en el receptor y lo mismo ocurre con un nivel bajo.
La señal portadora de este modulo de radio frecuencia se encuentra a una frecuencia de 315MHz y la máxima frecuencia para la transferencia de información es de 10kHz.

5.11 Modulación ASK

La modulación digital de amplitud o ASK consiste en cambiar la amplitud de una sinusoide entre dos valores posibles. Si uno de los dos valores anteriores es cero la modulación se denomina OOK.
La modulación OOK (On-Off keying) es un caso especial de la modulación ASK en donde no existe señal portadora durante la transmisión de un cero.

En este tipo de modulación el modulador consiste en un simple multiplicador de los datos binarios por una portadora. En la figura 5.13 se muestra un ejemplo de un mensaje en banda base y el resultado de modular en ASK(OOK).

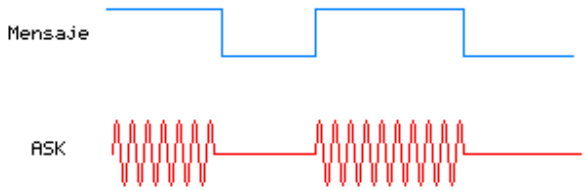


Figura 5.13 Modulación ASK (OOK)

De la figura anterior es posible observar que solo cuando se transmite un nivel alto existe una señal moduladora.

La modulación OOK tiene la ventaja de ser muy sencilla y de fácil implementación, también se tiene la ventaja de que cuando la señal moduladora adquiere el valor de cero no se transmite señal alguna por lo que es posible tener transmisores que conserven energía durante la transmisión de un cero lógico. La principal desventaja de este tipo de modulación es la presencia de una señal indeseada la cual puede llegar a ser interpretada como un nivel alto por el receptor.

5.12 Protocolo empleado en la transmisión de la información

Debido a que el modulo de RF antes descrito no cuenta con alguna interfaz para la transmisión o recepción de palabras que puedan contener información alguna se tuvo la necesidad de programar dentro del CPLD en lenguaje VHDL un protocolo propio para la transmisión y recepción de palabras de 10 bits.

A continuación se describe de forma breve los principios que se utilizaron para la realización de un protocolo propio que permite la transmisión y la recepción de una palabra de 10 bits.
En la transmisión de radio frecuencia se codificaron 5 instrucciones diferentes que se envían en palabras de 10 bits. La elección de 10 bits para codificar cada una de las instrucciones se debió al hecho de asegurar que no existiera interferencia en la transmisión.
Al momento de transmitir una palabra se envían los primeros 5 bits en alto lo que el receptor debe interpretar como inicio de la transmisión. Finalmente después de los 5 bits de inicio se comienza a transmitir la palabra de 10 bits que contiene la instrucción que debe realizar el androide.

En la figura 5.14 se observa el diagrama de tiempo del transmisor enviando una palabra.

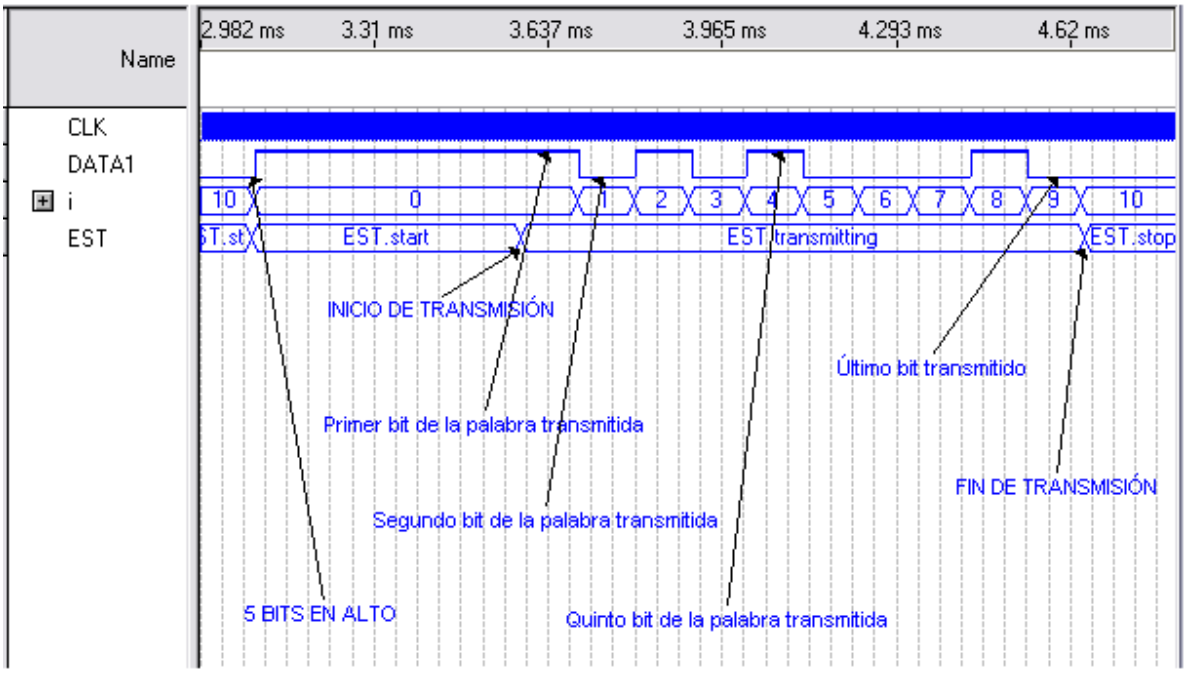


Figura 5.14 Transmisión de una palabra

De la figura anterior es posible observar que la palabra transmitida fue “1010100010” la cual es antecedida por 5 bits en nivel alto.

Debido a que es necesario enviar diferentes instrucciones al androide para que realice distintas acciones, se programó una máquina de estados en el circuito transmisor la cual es mostrada en la figura 5.15 en donde se observa que solo cuando se presiona un botón en el circuito transmisor se transmite una palabra, de otra forma el transmisor espera una instrucción sin transmitir información alguna. De la misma figura también es posible apreciar que si al final de la transmisión se continúa oprimiendo el mismo botón entonces se vuelve a transmitir una vez más la palabra deseada.

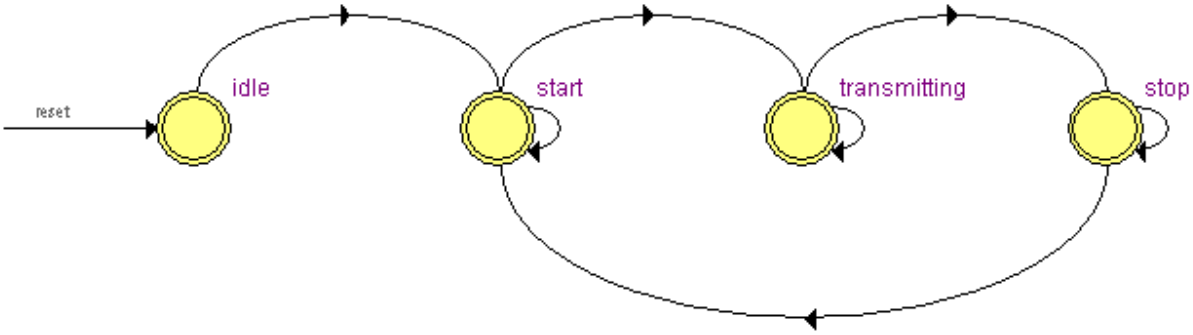


Figura 5.15 Máquina de estados en el CPLD transmisor

Se codificaron 5 diferentes instrucciones cada una con una palabra específica, las cuales deben de estar grabadas en la parte del código que se encuentra en el control del robot para que éste las reconozca y realice una acción específica. En caso de que el programa que se encarga de la recepción reciba alguna otra palabra diferente a las 5 antes mencionadas no se realiza ninguna acción.

El circuito transmisor se implemento físicamente con el modulo de transmisión antes mencionado junto con una tarjeta MAX II micro board idéntica con la que cuenta el androide.

5.13 Recepción digital serial asíncrona

En el módulo receptor de RF que se encuentra en el androide para la recepción de instrucciones no se cuenta con la señal de reloj usada para la transmisión de la palabra de 10 bits que contiene la información. Por tal motivo es necesario implementar un código en VHDL que permita la recepción serial asíncrona.

Para lograr la recepción de forma correcta de acuerdo al protocolo de comunicación antes expuesto es necesario identificar los 5 bits en alto que indican el inicio en la transmisión de una palabra y posteriormente muestrear cada uno de los 10 bits de la palabra recibida.

En la figura 5.16 es posible observar que cuando el circuito receptor identifica un nivel alto se inicia un contador que determina si se tienen 5 bits en alto. De ser así entonces se inicia otro contador en el circuito receptor que toma una muestra de cada uno de los 10 bits de la palabra en aproximadamente el centro de cada uno de ellos.

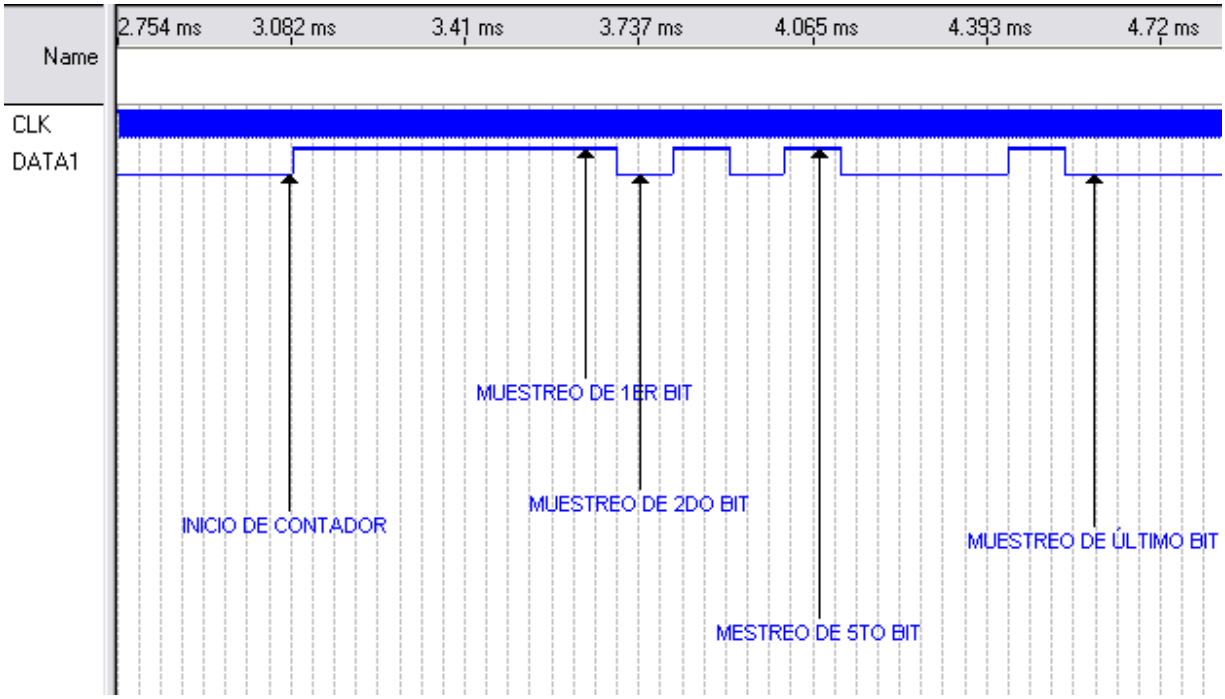


Figura 5.16 Recepción digital serial asíncrona

Cabe mencionar que en una recepción asíncrona de este tipo es necesario conocer la frecuencia de la señal moduladora, lo que permite la elaboración de un circuito que muestre cada uno de los bits de la palabra de información. De otra forma sería imposible tomar una muestra de los bits de la palabra que contiene la información ya que no se conocería la duración de cada uno de ellos.

5.14 Máquina de estados

El control de los movimientos que se tendrá sobre el androide se lleva a cabo mediante una máquina de estados, cada vez que un vector de posición inicial se iguala con un vector posición final en una línea de código, es la indicación de que se puede pasar a otro estado en donde el vector inicial de posición es igual a el vector final de posición del estado anterior. En el caso de caminar se tendrán una serie de estados finitos que se van repitiendo según sea el caso.

En la figura 5.17 se muestra el trayecto y movimientos del androide; se inicia desde una posición inicial (meta) y conforme el sensor indique que no se tiene un obstáculo el androide caminará hasta encontrar uno, en ese momento seguirá una secuencia de movimientos que le permita dar un giro de 180° y regresar hacia la meta, en donde se detendrá una vez que la detecte.

Figura 5.17 Trayectoria a seguir para el androide

En este proyecto de tesis, solo se lleva cabo una vez el trayecto a la pared, no es un ciclo y por lo tanto el androide no deberá volver a adquirir la posición inicial (viendo hacia la pared) en algún otro estado, esto implicaría otro giro de 180°, únicamente llegará a la meta y se detendrá.

5.15 Diagrama de flujo

En el diagrama de flujo se lleva todo a una forma a la cual se establecen los estados y las condiciones para la lógica secuencial que debe ser implementada. La figura 5.18 muestra el diagrama de flujo correspondiente a la figura en la cual se muestra los pasos a seguir en la estructura de diseño.

De una manera descrita a un lenguaje de nivel mayor, el diagrama de flujo es visto como: una señal activa el ciclo caminar hasta que un obstáculo active el ciclo de giro sobre si mismo, una vez terminado el ciclo, nuevamente se ingresa al ciclo caminar hasta que llegue a la meta y se detenga y se quede en la posición inicial.

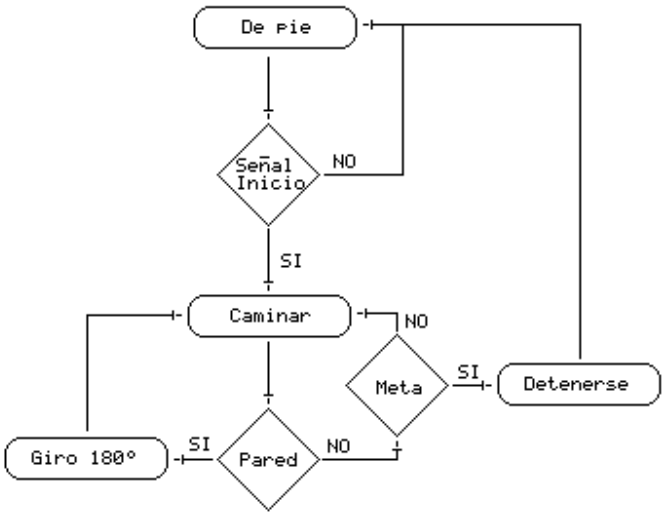


Figura 5.18 Diagrama de flujo

El más alto lenguaje de descripción, pone en orden los pasos a seguir de la lógica que debe activar las matrices de movimiento en cada uno de los estados necesarios para que el androide realice esta trayectoria.

5.16 Consumo de energía del androide

Toda la energía requerida por el androide y sus diferentes dispositivos provienen de 6 baterías recargables níquel metal hidruro (NI-MH) de 1.2V cada una. En conjunto forman una batería de 7.2 V la cual provee al androide de una autonomía de funcionamiento de al menos 25 minutos.

En la tabla 5.2 se muestra el consumo de potencia del androide al realizar algunas de las acciones programadas.

Acción realizada	Potencia disipada
Mantenerse de pie	6.14W
Caminar	9.25W
Girar para cambiar de dirección	11.24W
Mover dos extremidades bruscamente	14.13W

Tabla 5.2 Consumo de energía del androide

CAPÍTULO VI

Resultados Finales y Conclusiones

6.1 Resultados finales

A continuación se muestran algunos de los resultados reportados por la herramienta de desarrollo Quartus II sobre el código final para el control del androide.
Cuando se compiló el código completo que se encarga de controlar todas las funciones del androide se obtiene el reporte mostrado en la figura 6.1

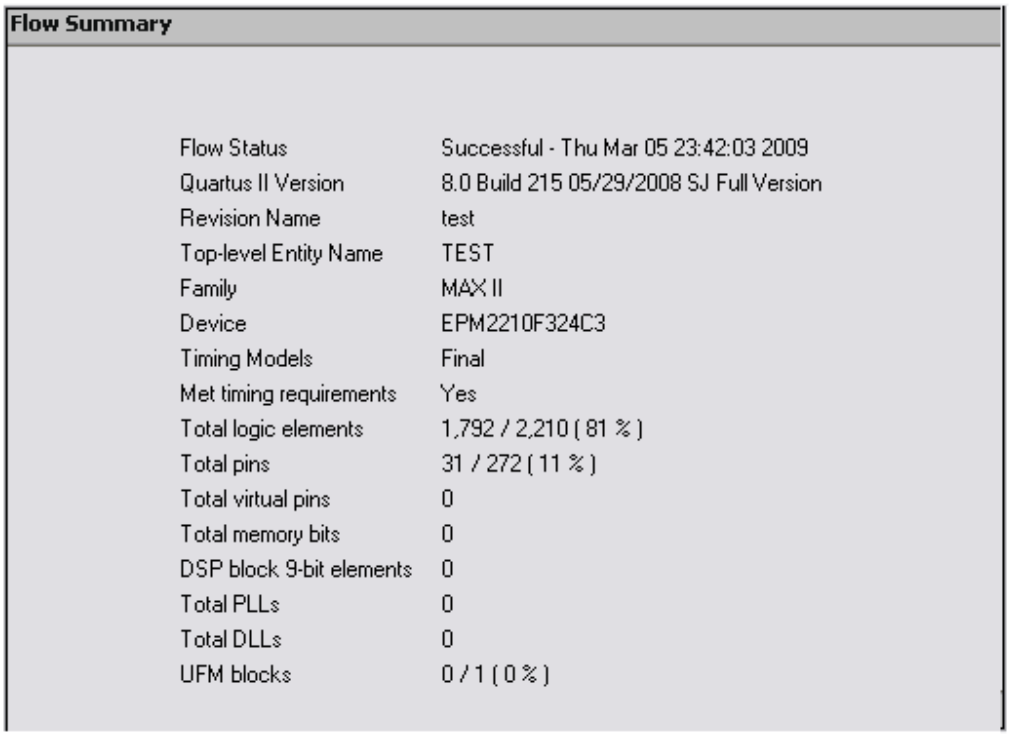


Figura 6.1 Resumen de resultados

De la figura anterior se observa que se utilizaron un total de 1792 elementos lógicos del dispositivo EPM2210 que es con el que cuenta la tarjeta MAX II micro board empleada en el androide. El número de elementos lógicos anterior representa el 81% de la capacidad total del integrado EPM2210.
De la figura anterior es posible observar también que se están empleando 31 pines de un total de 272 con los que cuenta el dispositivo EPM2210.

6.2 Diagramas de conexiones

Fue necesario diseñar y construir dos circuitos impresos: Uno que lleva el androide y otro que es el que tiene el control transmisor.

- Diagrama de conexiones para el androide.

  El circuito impreso que lleva el androide realiza todas las conexiones necesarias entre la tarjeta y los demás dispositivos que forman parte del sistema. Dentro del mismo circuito impreso también fue necesario considerar los voltajes de polarización de cada uno de los dispositivos, ya que no todos funcionan con el mismo voltaje de polarización. Toda la potencia requerida por el conjunto de dispositivos y el androide provienen de una batería de 7.2 V ubicadas dentro del robot androide. En la figura 6.2 se muestra el diagrama de conexiones existentes entre la tarjeta MAX II micro board, el módulo de radio frecuencia, el sensor de distancia y los 16 servomotores.

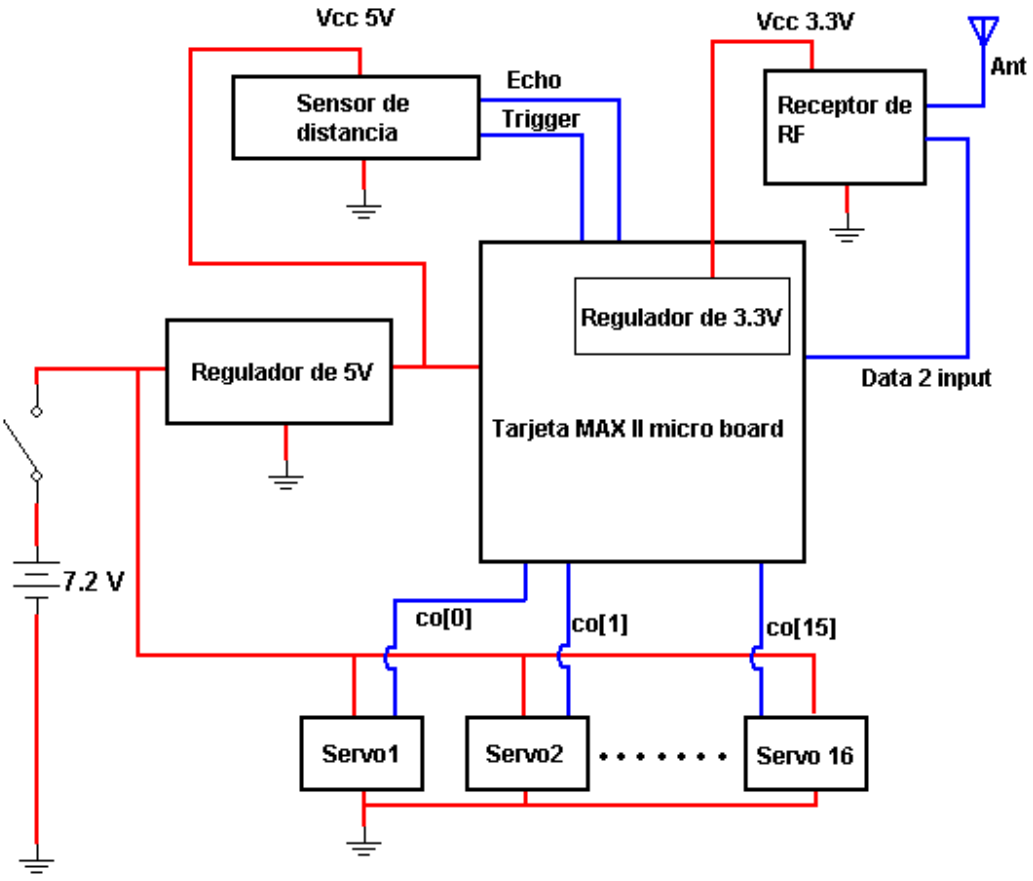


Figura 6.2 Diagrama de conexiones para el androide

De la figura anterior es importante notar que la señal de control enviada a los servomotores tiene una amplitud de 3.3V, que es el voltaje de salida manejado por el CPLD de la tarjeta. Sin embargo, los servomotores se polarizaron con el voltaje de 7.2V de las baterías, con la finalidad de que éstos hagan uso de toda la potencia que pueden suministrar las baterías.

- Diagrama de conexiones del transmisor de radio frecuencia.

  En la figura 6.3 se observa el diagrama de conexiones para el transmisor, de donde se ve que se utilizó una batería de 9.6V para polarizar el transmisor de RF y de esta manera tener un mayor alcance de la señal transmitida. De forma similar al diagrama de conexiones de la figura 6.2, se utilizó un regulador de 5V para polarizar a la tarjeta MAXII micro board que se encarga de generar el tren de bits a transmitirse.

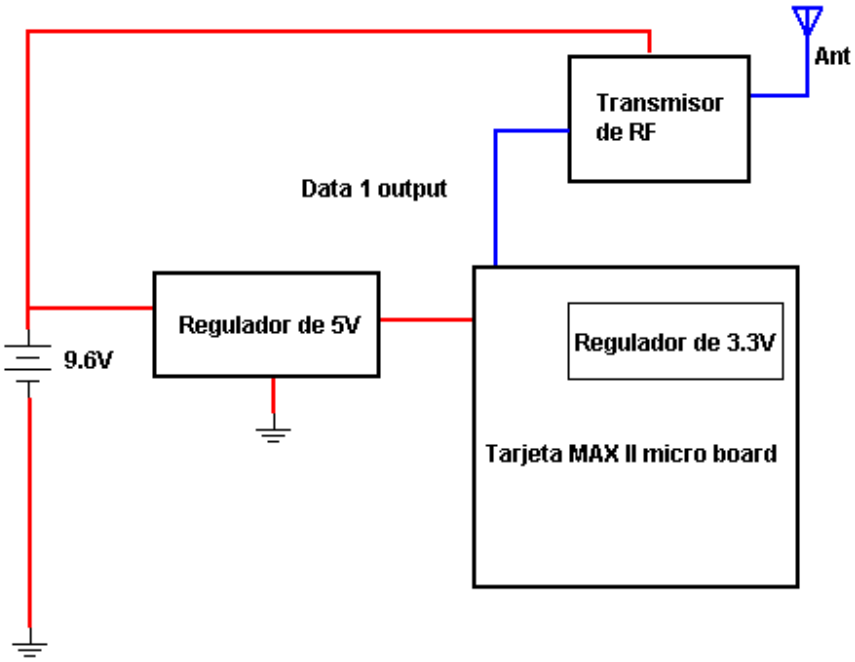


Figura 6.3 Diagrama de conexiones del transmisor de RF

6.3 Conclusiones

Se cumplió con el propósito de esta tesis que era lograr el control absoluto sobre los movimientos del androide, tales como caminar, dar vuelta, mover los brazos, girar la cabeza para detectar un obstáculo, etc. Consecuentemente se pone al descubierto la efectividad de la lógica programable, las herramientas del diseño y los fundamentos antes expuestos en esta tesis.

Esta tesis ha mostrado una manera muy eficiente de programación para aplicaciones de hardware. El control realizado y las ideas antes expuestas pueden ser retomadas para posibles aplicaciones futuras en diseños incluso más complejos.

El lenguaje de programación VHDL es muy intuitivo y de fácil abstracción y, a diferencia de los lenguajes ensamblador comúnmente utilizados con los microcontroladores, se tiene la capacidad de migrar de un dispositivo a otro sin importar quien lo halla el fabricado ya que VHDL es un lenguaje totalmente portátil y reusable.

Entonces, se abre paso toda una amplia gama de aplicaciones basada en la programación en VHDL, lo cual permitirá implementar desde transmisiones digitales, control por radio frecuencia y obtención de datos hasta el control sobre muchos dispositivos a la vez, en un tiempo de implementación relativamente corto.

A diferencia de los microprocesadores, con un CPLD se tiene la capacidad de ejecutar múltiples procesos de manera simultánea con lo que se dejan atrás problemas como retardos generados por las instrucciones y revisiones de banderas, implementación de preescaladores, overflow en el manejo de datos y operaciones con enteros en los acumuladores, entre otros. Puede decirse entonces, que con un dispositivo lógico se tiene la capacidad total de implementar una arquitectura propia que se adapte por completo a una aplicación específica, con lo que también se logra un modo de programación mas eficiente y una utilización del dispositivo mucho más óptima.

ANEXOS

A continuación se anexa el código VHDL generado para el control del androide.

```
--////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////---
----------------------CONTROL DEL ANDROIDE EN VHDL ----------------------------------------------
--////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////---

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
PACKAGE MY_DATA_TYPES IS
            TYPE TITIVECTOR IS ARRAY (NATURAL RANGE <>) OF BIT;
END MY_DATA_TYPES;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE WORK.MY_DATA_TYPES.ALL;
ENTITY  TEST  IS
            PORT(CLK: IN STD_LOGIC;ECHO:IN STD_LOGIC;TRIG:OUT BIT;
                    SEL:IN INTEGER RANGE 0 TO 15;
                DATA2:IN STD_LOGIC;
        LED1:OUT STD_LOGIC;
        LED2:OUT STD_LOGIC;
        LED3:OUT STD_LOGIC;
        LED4:OUT STD_LOGIC;
        LED5:OUT STD_LOGIC;
        LED6:OUT STD_LOGIC;
                    co:OUT TITIVECTOR(0 TO 15);
                NECK:OUT STD_LOGIC);
END ENTITY;

ARCHITECTURE ALGOR OF TEST IS
            SIGNAL AUX1:INTEGER RANGE 0 TO 250;SIGNAL AUX2:INTEGER RANGE 0 TO 256;
            SIGNAL i,j:INTEGER RANGE 0 TO 16;
            SIGNAL AUX3:INTEGER RANGE 0 TO 5000;
            SIGNAL HEAD:INTEGER RANGE 0 TO 450;
  SIGNAL CONT1:INTEGER RANGE 0 TO 400000;
            TYPE EE2 IS(WAITING, START,RECIEVING,COMPARING);
  SIGNAL EST2:EE2:=WAITING;
  SIGNAL PALABRA:STD_LOGIC_VECTOR(0 TO 9);
  SIGNAL CONT11:INTEGER RANGE 0 TO 20;
  SIGNAL CONT44:INTEGER RANGE 0 TO 40;
  SIGNAL TEMP:INTEGER RANGE 0 TO 5;
  SIGNAL CONTWAIT:INTEGER RANGE 0 TO 100;
  SIGNAL CONT22:INTEGER RANGE 0 TO 120;
  SIGNAL ii:INTEGER RANGE 0 TO 10;
  SIGNAL AUTO:STD_LOGIC;
  SIGNAL ALTO:STD_LOGIC;
    ------SRF05
            SIGNAL AUX5,AUX6,AUX7:INTEGER RANGE 0 TO 1000;
            SIGNAL TEMPS,LEN:INTEGER RANGE 0 TO 300;
            SIGNAL AUX8:INTEGER RANGE 0 TO 10;
            SIGNAL CLK5,CLK6,CLK7,NEAR:BIT;
            TYPE vector IS ARRAY (15 DOWNTO 0) OF INTEGER RANGE 0 TO 256;SIGNAL x,y,z:vector;
            TYPE svector IS ARRAY (15 DOWNTO 0) OF INTEGER RANGE 0 TO 200;SIGNAL w:svector;
            SIGNAL CLK1:STD_LOGIC;
            TYPE st IS(a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,
            a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,
            a41,a42,a43,a44,a45,a46,a47,a48,a49,a50,a51,a52,a53,a54,a55,a56,a57);
            SIGNAL STATE:st:=a1;
  TYPE BB IS(b1); SIGNAL BB30:BB:=b1;

            BEGIN

            PROCESS(CLK,CLK1)
            BEGIN
------------------------------DIVISOR DE RELOJ DE 50MHz a 200kHz--------------------------
                        IF CLK'EVENT AND CLK='1' THEN AUX1<=AUX1+1;
                                IF 125>AUX1 THEN CLK1<='0';END IF;
                                IF AUX1>=125 AND 250>AUX1 THEN CLK1<='1';END IF;
                                IF 250=AUX1 THEN AUX1<=0;END IF;

-----------------------------------------------------------------------------------------------------------
```

```
----------COMPARACIÓN DE IGUALDAD DE VECTOR FINAL CON VECTOR VARIABLE
--cuando los 16 elementos del vector son iguales se puede pasar a otro estado--
                                    IF z(j)=y(j) THEN j<=j+1;END IF;
                                    IF j=16 THEN j<=0;END IF;
                            END IF;
-------------------------------------------------------------------------------------------------------------------------------
------------ENVIO DE TREN DE PULSOS A LOS 16 SERVOMOTRES-----------------------------------
                            IF CLK1'EVENT AND CLK1='1' THEN AUX2<=AUX2+1;
                                    IF z(i)=0 THEN z(i)<=x(i);END IF;
                                    IF z(i)>AUX2 THEN co(i)<='1';END IF;
                                    IF AUX2>z(i) THEN co(i)<='0';END IF;
                                    IF z(i)+1=AUX2 THEN
                                            IF z(i)<y(i) THEN z(i)<=z(i)+w(i);END IF;
                                            IF z(i)>y(i) THEN z(i)<=z(i)-w(i);END IF;
                                            AUX2<=0;i<=i+1;
                                    END IF;IF i=16 THEN i<=0;END IF;
                            END IF;
                            END PROCESS;
-------------------------------------------------------------------------------------------------------------------------------

-------CONTADOR PARA RETARDOS DE TIEMPO ENTRE ESTADOS----------------------
            PROCESS(CLK1,STATE,BB30,j)
            BEGIN
            IF STATE=a9 OR STATE=a10 OR STATE=a11 THEN ---retardo de 2 segundos para el estado a9 a10 y a11
             IF CLK1'EVENT AND CLK1='1' THEN IF CONT1=400000 THEN CONT1<=0;
             ELSE CONT1<=CONT1+1; END IF; END IF;END IF;
             END PROCESS;
-----------------------------------------CONTROL PARA  EL SERVO DE LA CABEZA-----------
             PROCESS(CLK1)
            BEGIN
                    IF CLK1'EVENT AND CLK1='1' THEN
                    IF AUX3=1000000 THEN AUX3<=0; ELSE
                    AUX3<=AUX3+1;
                    IF HEAD>AUX3 THEN NECK<='1';END IF;
                    IF HEAD<AUX3 THEN NECK<='0';END IF;
                    END IF;
                    END IF;
            END PROCESS;
----------------------------------PROCESOC PARA EL RECPETOR----------------
            PROCESS (EST2,CLK1)
BEGIN
 IF EST2=START OR EST2=RECIEVING  THEN
  IF CLK1'EVENT AND CLK1='1' THEN
  IF CONT11=20 THEN CONT11<=0;
  ELSE CONT11<=CONT11+1; END IF;END IF;
  ELSE CONT11<=0;
  END IF;

  IF EST2=COMPARING THEN
  IF CLK1'EVENT AND CLK1='1' THEN
  IF CONT22=120 THEN CONT22<=0;
  ELSE CONT22<=CONT22+1;END IF;END IF;
  ELSE CONT22<=0;
  END IF;
END PROCESS;

PROCESS(CLK1,DATA2,ii,TEMP,AUTO)
 BEGIN
IF CLK1'EVENT AND CLK1='1' THEN
CASE EST2 IS
WHEN WAITING=> IF CLK1'EVENT AND CLK1='1' THEN IF
                 DATA2='1' THEN
                 CONTWAIT<=CONTWAIT+1; ELSE CONTWAIT<=0;
                 END IF; END IF;

                IF CONTWAIT=90 THEN
                 EST2<=START;
                 CONTWAIT<=0; ELSE
                 EST2<=WAITING; END IF;
                  ii<=0;
```

```
WHEN START=> IF CONT11=20 THEN
                    EST2<=RECIEVING;
                    END IF;

 WHEN RECIEVING=>
                          IF CONT11=0 THEN
                          PALABRA(ii)<=DATA2;END IF;
                          IF CONT11=20 THEN
                          ii<=ii+1;
                          END IF;
                          IF ii=9  THEN
                          EST2<=COMPARING;
                          ii<=0;
                          END IF;

 WHEN COMPARING=> IF CONT22=120 THEN
                             EST2<=WAITING;ELSE
                             EST2<=COMPARING; END IF;

                            IF CONT22>60 THEN
                             IF PALABRA="0101010101"THEN
                             TEMP<=1;
                             AUTO<='0'; END IF; --prende rojo
                             IF PALABRA="1010100010" THEN
                             TEMP<=2;
                             AUTO<='0'; END IF; -- prende naranja
                             IF PALABRA="0110010101" THEN
                             TEMP<=3;
                             AUTO<='0'; END IF; -- prende verde
                             IF PALABRA="0010101001" THEN
                             TEMP<=4;
                             AUTO<='0';  END IF;
                             IF PALABRA="1101010110"THEN
                             TEMP<=5;
                             END IF;--prENDe azul
                             IF PALABRA="1100110011"THEN
                             AUTO<='1';END IF;
                             END IF;

 END CASE;
 END IF;

 IF TEMP=1 THEN ---izquierda
 LED1<='0'; ELSE LED1<='1';END IF;
 IF TEMP=2 THEN --derecha
 LED2<='0';ELSE LED2<='1';END IF;
 IF TEMP=3 THEN ---abajo
 LED3<='0';ELSE LED3<='1';END IF;
 IF TEMP=4 THEN --arriba
 LED4<='0';ELSE LED4<='1';END IF;
 IF TEMP=5 THEN--centro
 LED1<='0'; LED2<='0'; LED3<='0'; LED4<='0'; END IF;
 IF AUTO='1' THEN
 LED6<='0'; ELSE LED6<='1'; END IF;

 END PROCESS;


-------------------------------------------------------------------------------------------------------------
             PROCESS(STATE,CLK,CLK1)
             VARIABLE CONT2,CONT4,CONT5,CONT6:INTEGER RANGE 0 TO 40;

             BEGIN
--CONTADORES PARA DETERMINAR EL NUMERO DE VECES QUE DEBE DE
--LLEVARSE A CABO ALGUNO DE LOS ESTADOS
                IF CLK'EVENT AND CLK='1' THEN
                            IF STATE=a12 AND j=16 THEN CONT2:=CONT2+1;END IF;
                            IF STATE=a14 THEN CONT2:=0;END IF;
                            IF STATE=a25 AND j=16 THEN CONT4:=CONT4+1;END IF;
       IF STATE=a14 AND j=16 THEN CONT4:=0;END IF;
       IF STATE=a21 AND j=16 AND ALTO='0' THEN CONT5:=CONT5+1;END IF;
       IF STATE=a21 AND j=16 AND ALTO='1' THEN CONT5:=CONT5-1;END IF;
       IF STATE=a43 AND j=16 THEN CONT5:=0;END IF;
       IF STATE=a49 AND j=16 THEN CONT6:=CONT6+1;END IF;
```

```
      IF STATE=a45 AND j=16 THEN CONT6:=0;END IF;
--MAQUINA DE ESTADOS CON LOS DIFERENTES MOVIMIENTOS Y
--CONDICIONES DE CAMBIO ENTRE UNO Y OTRO
CASE STATE IS
WHEN a1=> IF j=16 AND TEMP=4 THEN STATE<=a2;ELSIF SEL=14 THEN STATE<=a14;ELSIF SEL=13 THEN
STATE<=a45;END IF;
WHEN a2=> IF j=16  THEN STATE<=a3;ELSIF SEL=14 THEN STATE<=a14; ELSIF SEL=7 THEN
STATE<=a31; END IF;
WHEN a3=> IF j=16   THEN STATE<=a4;ELSIF SEL=14 THEN STATE<=a14; END IF;
WHEN a4=> IF j=16   THEN STATE<=a5;ELSIF SEL=14 THEN STATE<=a14; END IF;
WHEN a5=> IF j=16   THEN STATE<=a6;ELSIF SEL=14 THEN STATE<=a14;END IF;
WHEN a6=> IF j=16    THEN STATE<=a7;ELSIF SEL=14 THEN STATE<=a14;END IF;
WHEN a7=> IF j=16    THEN STATE<=a8;ELSIF SEL=14 THEN STATE<=a14;END IF;
WHEN a8=> IF j=16    THEN STATE<=a9;ELSIF SEL=14 THEN STATE<=a14;END IF;
WHEN a9=> IF j=16     AND CONT1=400000 THEN STATE<=a10;ELSIF SEL=14 THEN
STATE<=a14;END IF;
WHEN a10=> IF j=16  AND CONT1=400000 THEN STATE<=a11;ELSIF SEL=14 THEN
STATE<=a14;END IF;
WHEN a11=> IF j=16  AND CONT1=400000 THEN STATE<=a12;ELSIF SEL=14 THEN
STATE<=a14;END IF;
WHEN a12=> IF j=16  THEN STATE<=a13;ELSIF SEL=14 THEN STATE<=a14;END IF;
WHEN a13=> IF j=16  THEN STATE<=a12;ELSIF SEL=14 THEN STATE<=a14;
ELSIF CONT2=20 THEN
STATE<=a14;END IF;ALTO<='0';

WHEN a14=> IF j=16  AND TEMP=4 THEN STATE<=a15;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;
           ELSIF TEMP=1 THEN STATE<=a31;ELSIF TEMP=3 THEN STATE<=a25; ELSIF TEMP=2 THEN STATE<=a37;
           END IF; IF j=16 AND AUTO='1' THEN STATE<=a15;END IF;
WHEN a15=> IF j=16  AND TEMP=4 THEN STATE<=a16;ELSIF TEMP=5 AND AUTO='0'  THEN STATE<=a14;
   END IF;IF j=16 AND AUTO='1' THEN STATE<=a16;END IF;  IF NEAR='0' AND AUTO='1'AND ALTO='0'THEN
            STATE<=a25; ALTO<='1'; END IF;
WHEN a16=> IF j=16  AND TEMP=4 THEN STATE<=a17;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;
   END IF;IF j=16 AND AUTO='1' THEN STATE<=a17;END IF; IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN
  STATE<=a25; ALTO<='1';END IF;
WHEN a17=> IF j=16   AND TEMP=4 THEN STATE<=a18;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;
  END IF; IF j=16 AND AUTO='1' THEN STATE<=a18;END IF;  IF NEAR='0'AND AUTO='1' AND ALTO='0' THEN
  STATE<=a25; ALTO<='1';END IF;
WHEN a18=> IF j=16  AND TEMP=4 THEN STATE<=a19;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;
  END IF; IF j=16 AND AUTO='1' THEN STATE<=a19;END IF;  IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN
  STATE<=a25; ALTO<='1';END IF;

WHEN a19=> IF j=16  AND TEMP=4 THEN STATE<=a20;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14; END IF;
IF j=16 AND AUTO='1' THEN STATE<=a20;END IF;  IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1'; END IF;
WHEN a20=> IF j=16  AND TEMP=4 THEN STATE<=a21;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14; END IF;
IF j=16 AND AUTO='1' THEN STATE<=a21;END IF;  IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1'; END IF;
WHEN a21=> IF j=16   AND TEMP=4 THEN STATE<=a22;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;END IF;
IF j=16 AND AUTO='1' THEN STATE<=a22;END IF; IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1';END IF;  ---donde puede pararse
WHEN a22=> IF j=16  AND TEMP=4 THEN STATE<=a23;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14; END IF;
IF j=16 AND AUTO='1' THEN STATE<=a23;END IF; IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1';END IF;
WHEN a23=> IF j=16  AND TEMP=4 THEN STATE<=a24;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14; END IF;
IF j=16 AND AUTO='1' THEN STATE<=a24;END IF; IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1';END IF;
WHEN a24=> IF j=16  AND TEMP=4 THEN STATE<=a19;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14; END IF;
IF j=16 AND AUTO='1' THEN STATE<=a19;END IF; IF NEAR='0' AND AUTO='1' AND ALTO='0'THEN STATE<=a25;
ALTO<='1';END IF;  IF j=16 AND CONT5=0 THEN STATE<=a43;END IF;----donde puede pararse

WHEN a25=> IF j=16  AND TEMP=3 THEN STATE<=a26;ELSIF TEMP=5 AND AUTO='0' THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a26;END IF;
WHEN a26=> IF j=16  AND TEMP=3  THEN STATE<=a27;ELSIF TEMP=5 AND AUTO='0'THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a27;END IF;
WHEN a27=> IF j=16  AND TEMP=3  THEN STATE<=a28;ELSIF TEMP=5 AND AUTO='0'THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a28;END IF;
WHEN a28=> IF j=16  AND TEMP=3  THEN STATE<=a29;ELSIF TEMP=5 AND AUTO='0'THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a29;END IF;
WHEN a29=> IF j=16  AND TEMP=3  THEN STATE<=a30;ELSIF TEMP=5 AND AUTO='0'THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a30;END IF;
WHEN a30=> IF j=16  AND TEMP=3  THEN STATE<=a25;ELSIF TEMP=5 AND AUTO='0'THEN STATE<=a14;
                END IF; IF j=16 AND AUTO='1' THEN STATE<=a25;END IF; IF CONT4=9 THEN STATE<=a14;END IF;

WHEN a31=> IF j=16   AND TEMP=1 THEN STATE<=a32;  ELSIF TEMP=5 THEN STATE<=a14;END IF;
```

```
WHEN a32=> IF j=16   AND TEMP=1 THEN STATE<=a33;  ELSIF  TEMP=5 THEN STATE<=a14;END IF;
WHEN a33=> IF j=16    AND TEMP=1 THEN STATE<=a34;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a34=> IF j=16   AND TEMP=1 THEN STATE<=a35;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a35=> IF j=16    AND TEMP=1 THEN STATE<=a36;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a36=> IF j=16    AND TEMP=1 THEN STATE<=a32;  ELSIF TEMP=5  THEN STATE<=a14;END IF;

WHEN a37=> IF j=16   AND TEMP=2 THEN STATE<=a38;  ELSIF TEMP=5 THEN STATE<=a14;END IF;
WHEN a38=> IF j=16   AND TEMP=2 THEN STATE<=a39;  ELSIF  TEMP=5 THEN STATE<=a14;END IF;
WHEN a39=> IF j=16    AND TEMP=2 THEN STATE<=a40;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a40=> IF j=16   AND TEMP=2 THEN STATE<=a41;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a41=> IF j=16    AND TEMP=2 THEN STATE<=a42;  ELSIF TEMP=5  THEN STATE<=a14;END IF;
WHEN a42=> IF j=16    AND TEMP=2 THEN STATE<=a38;  ELSIF TEMP=5  THEN STATE<=a14;END IF;

WHEN a43=> IF j=16 AND TEMP/=5 THEN STATE<=a14;ALTO<='0';ELSIF j=16 THEN STATE<=a44;END IF;
WHEN a44=> IF j=16  THEN STATE<=a45;END IF;
WHEN a45=> IF j=16  AND TEMP/=5 THEN STATE<=a46;END IF;
WHEN a46=> IF j=16   THEN STATE<=a47;END IF;
WHEN a47=> IF j=16   THEN STATE<=a48;END IF;
WHEN a48=> IF j=16   THEN STATE<=a49;END IF;
WHEN a49=> IF j=16   THEN STATE<=a50;END IF;
WHEN a50=> IF j=16   THEN STATE<=a49;ELSIF CONT6=10 THEN STATE<=a51;END IF;
WHEN a51=> IF j=16   THEN STATE<=a52;END IF;
WHEN a52=> IF j=16   THEN STATE<=a53;END IF;
WHEN a53=> IF j=16   THEN STATE<=a54;END IF;
WHEN a54=> IF j=16   THEN STATE<=a55;END IF;
WHEN a55=> IF j=16   THEN STATE<=a56;END IF;
WHEN a56=> IF j=16   THEN STATE<=a57;END IF;
WHEN a57=> IF j=16   THEN STATE<=a57;END IF;
-------------------------------------------------------------------------------------------------------------------------------------------------------
                        END CASE;
                        END IF;
                        CASE STATE IS
--VECTORES DE POSICIÓN
--X=vector de posicón inicial
--Y=vector d posición final
--W=velocidad con la que cada elemento de X llega a su posicion final Y
--Los signos por encima de cada elemento de los vectores indica si se debe de incrementar
--o decrementar el valor del elemento del vecotor para hacer que el servo al que controla se mueva hacia la
--derecha o hacia arriba siempre.
--Los elementos de todos los vectores representan los diferentes servos del androide en el siguiente orden:
--(empeine derecho, antebrazo derecho, rodilla derecha, codo derecho, tobillo derecho, Cadera derecha, hombro derecho, muslo derecho,
--rodilla izquierda, tobillo izquierdo, hombro izquierdo, Cadera izquierda, antebrazo izquierdo, codo izquierdo, empeine izquierdo,muslo --izquierdo)

----------POSICIÓN SENTADO Y LUEGO SE PARA-----------------------------------------------------------------
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a1=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --Primera posición sentado
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a2=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,320,240,170,360); --sube brazo izquierdo
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,002,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a3=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,320,240,170,360);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --baja brazo izquierdo
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,002,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a4=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,280,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --sube brazo derecho
            w<=(001,002,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a5=>x<=(430,280,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --baja brazo derecho
            w<=(001,002,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
```

```
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a6=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(280,420,300,360,300,305,300,230,300,300,300,295,180,240,320,360); --se levanta 1
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);

                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a7=>x<=(280,420,300,360,300,305,300,230,300,300,300,295,180,240,320,360);
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --se levanta 2
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
-------------------------------------------------------------------------------------------------------------------------
----------MUEVE BRAZOS RAPIDAMENTE CUANDO ESTA PARADO----------
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a8=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --empieza a mover
             y<=(280,420,340,360,300,305,300,270,260,300,400,295,180,240,320,320);  --hombro izquierdo
           w<=(001,001,001,001,001,001,001,001,001,001,005,001,001,001,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a9=>x<=(280,420,340,360,300,305,300,270,260,300,400,295,180,240,320,320); --mueve brazos
             y<=(280,300,340,300,300,305,180,270,260,300,480,295,110,130,320,320); --a la derecha
            w<=(001,120,001,060,001,001,120,001,001,001,080,001,070,110,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a10=>x<=(280,300,340,300,300,305,180,270,260,300,480,295,110,130,320,320); --mueve brazos
             y<=(280,470,340,470,300,305,120,270,260,300,420,295,300,300,320,320); --a la izquierda
            w<=(001,170,001,170,001,001,060,001,001,001,060,001,190,170,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a11=>x<=(280,470,340,470,300,305,120,270,260,300,420,295,300,300,320,320); --baja brazos a posición inicial
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320);
            w<=(001,050,001,110,001,001,180,001,001,001,120,001,120,060,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a12=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --levanta tobillos
             y<=(280,400,340,380,300,305,300,270,260,300,300,295,200,220,320,320);
            w<=(001,010,001,010,001,001,001,001,001,001,001,001,010,010,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a13=>x<=(280,400,340,380,300,305,300,270,260,300,300,295,200,220,320,320); --baja tobillos
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320);
            w<=(001,010,001,010,001,001,001,001,001,001,001,001,010,010,001,001);
-------------------------------------------------------------------------------------------------------------------------
----------SECUENCIA DE CAMINATA----------
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a14=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --no hace nada y primera posición
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --antes de caminar
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a15=>x<=(280,420,340,360,300,305,300,250,260,300,300,295,180,240,320,340);--inclinación a la izquierda
             y<=(280,420,340,300,326,321,300,250,260,326,300,321,130,290,320,340);
            w<=(001,001,001,005,002,002,001,001,001,002,001,002,005,005,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a16=>x<=(280,420,340,300,326,321,300,250,260,326,300,321,130,290,320,340);
             y<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,340); -- levanta pie derecho
            w<=(001,001,030,001,001,001,001,030,001,001,001,001,001,001,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a17=>x<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,340);
             y<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,340); --inclinación hacia el frente
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
              --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
  --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a18=>x<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,340);
             y<=(220,420,380,300,325,325,350,224,260,325,350,320,130,290,320,340); --patada derecha
            w<=(002,001,002,001,001,001,005,002,001,001,005,001,001,001,001,001);
```

```
----------SECUENCIA QUE SE REPITE CUANDO YA ESTA CAMINANDO----------
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a19=>x<=(218,420,382,300,325,325,325,224,260,325,325,320,130,290,320,340);
             y<=(235,470,375,300,277,277,325,214,260,277,325,276,180,290,320,340);--inclinación a la derecha
            w<=(001,005,001,001,002,002,001,001,001,002,001,002,005,005,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a20=>x<=(235,470,375,300,277,277,325,214,260,277,325,276,180,290,320,340);
             y<=(235,470,375,300,275,275,325,214,305,275,325,269,180,290,320,385); --levanta pie izquierdo
            w<=(001,001,001,001,001,001,001,001,045,001,001,001,001,001,001,045);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a21=>x<=(235,470,375,300,275,275,325,214,305,275,325,269,180,290,320,385);
             y<=(281,470,339,300,275,275,275,246,229,275,275,269,180,290,380,369); --patada izquierda!
            w<=(002,001,002,001,001,001,005,002,004,001,005,001,001,001,004,002);                                                                --
(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a22=>x<=(281,470,339,300,275,275,275,246,229,275,275,269,180,290,380,369);
             y<=(281,420,339,300,325,325,275,230,229,325,275,321,130,290,380,385);--segunda inclinación a la izquierda
            w<=(001,005,001,001,002,002,001,002,001,002,001,002,005,001,001,002);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a23=>x<=(281,420,339,300,325,325,275,230,229,325,275,321,130,290,380,385);
             y<=(270,420,290,300,325,325,275,189,228,325,275,321,130,290,380,386); --levanta pie derecho
            w<=(011,001,049,001,001,001,001,041,001,001,001,001,001,001,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a24=>x<=(270,420,290,300,325,325,275,189,228,325,275,321,130,290,380,386);
             y<=(218,420,382,300,325,325,325,225,260,325,325,320,130,290,330,340);--2da patada derecha
            w<=(004,001,004,001,001,001,005,002,002,001,005,001,001,001,002,002);


----------SECUENCIA PARA GIRAR A LA IZQUIERDA----------

               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a25=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --primera posición parado
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320);
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a26=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --inlcinación a la izquierda
             y<=(280,420,340,300,326,321,300,270,260,326,300,321,130,290,320,320);
            w<=(001,001,001,005,002,002,001,001,001,002,001,002,005,005,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a27=>x<=(280,420,340,300,326,321,300,270,260,326,300,321,130,290,320,320);
             y<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,320); -- levanta pie derecho
            w<=(001,001,030,001,001,001,001,010,001,001,001,001,001,001,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a28=>x<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,320);
             y<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,320); --no hace nada
            w<=(001,001,001,001,001,001,001,020,001,001,001,001,001,001,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a29=>x<=(280,420,310,300,325,325,300,220,260,325,300,320,130,290,320,320);
             y<=(210,420,380,300,326,325,300,200,280,326,300,321,130,290,300,320); --patada derecha
            w<=(002,001,002,001,001,001,005,002,001,001,005,001,001,001,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a30=>x<=(210,420,380,300,326,325,300,200,280,326,300,321,130,290,300,320);
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --endereza a posición inicial
            w<=(002,001,002,002,002,002,001,002,002,002,001,002,001,002,001,002);
-------------------------------------------------------------------------------------------------------------------------------------------------------
---SECUENCIA PARA MOVERSE DE LADO A LA IZQUIERDA

               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)  (+) (-) (+)  (+) (-)  (+)
WHEN a31=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --no hace nada y primera posición
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --antes de caminar
            w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
```

```
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a32=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --inclinación a la derecha
             y<=(280,420,340,360,274,275,300,270,260,274,300,275,220,240,320,320);
             w<=(001,001,001,001,002,002,001,001,001,002,001,002,002,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a33=>x<=(280,420,340,360,274,275,300,270,260,274,300,275,220,240,320,320); --alza pie izquierdo
             y<=(280,420,340,360,274,275,300,270,290,274,300,275,220,240,290,320);
             w<=(001,001,001,001,001,001,001,001,002,001,001,002,001,001,002,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a34=>x<=(280,420,340,360,274,275,300,270,290,274,300,275,220,240,290,320); --empuja a la izquierda
             y<=(280,420,340,360,320,321,300,270,290,274,300,275,220,240,290,320);
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a35=>x<=(280,420,340,360,320,321,300,270,290,274,300,275,220,240,290,320);--se inlcina a la izquierda
             y<=(280,420,340,360,326,321,300,270,260,326,300,321,220,240,320,320);
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a36=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --no hace nada y primera posición
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320);
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);

-----SECUENCIA PARA MOVERSE DE LADO A LA DERECHA

                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a37=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --no hace nada y primera posición
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --antes de caminar
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a38=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --inclinación a la izquierda
             y<=(280,380,340,360,326,321,300,270,260,326,300,325,180,240,320,320);
             w<=(001,002,001,001,002,002,001,001,001,002,001,002,002,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a39=>x<=(280,380,340,360,326,321,300,270,260,326,300,325,180,240,320,320); --alza pie derecho
             y<=(310,380,310,360,326,321,300,270,260,326,300,325,180,240,320,320);
             w<=(002,001,002,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a40=>x<=(310,380,310,360,326,321,300,270,260,326,300,325,180,240,320,320); --empuja a la derecha
             y<=(310,380,310,360,326,335,300,270,260,280,300,279,180,240,320,320);
             w<=(001,001,001,001,001,002,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a41=>x<=(310,380,310,360,326,335,300,270,260,280,300,279,180,240,320,320); --se inclina a la derecha
             y<=(300,380,310,360,280,290,300,270,260,280,300,285,180,240,320,320);
             w<=(001,002,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a42=>x<=(280,426,310,360,280,290,248,270,260,280,300,285,180,240,320,320); --no hace nada
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --y primera posición
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
-------------------------------------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------------------------------------
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a43=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --ALTO DESPUES DE HABER
             y<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320); --DETECTADO PARED
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001); --Y HABER REGRESADO
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)   (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a44=>x<=(280,420,340,360,300,305,300,270,260,300,300,295,180,240,320,320);
             y<=(280,420,300,360,300,305,300,230,300,300,300,295,180,240,320,360); --SE SIENTA 1
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
```

```
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a45=>x<=(280,420,300,360,300,305,300,230,300,300,300,295,180,240,320,360);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --SE SIENTA 2
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a46=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,420,150,360,300,305,130,170,450,300,470,295,180,240,170,420); --SE INLCINA HACIA ADELANTE
             w<=(001,001,001,001,001,001,005,001,001,001,005,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a47=>x<=(430,420,150,360,300,305,130,170,450,300,470,295,180,240,170,420);
             y<=(430,420,210,360,300,305,130,170,390,300,470,295,180,240,170,420);--SE INLCINA HACIA ADELANTE2
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a48=>x<=(430,420,210,360,300,305,130,170,390,300,470,295,180,240,170,420);
              y<=(280,420,340,360,300,305,130,270,260,300,470,295,180,240,320,320);--ESTIRA LAS PIERNAS
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a49=>x<=(280,420,340,360,300,305,130,270,260,300,470,295,180,240,320,320);
             y<=(280,318,340,462,300,305,130,270,260,300,470,295,282,138,320,320); --HACE LAGARTIJAS
             w<=(001,003,001,003,001,001,001,001,001,001,001,001,003,003,001,001);
               --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
      --              (+) (-)  (+)  (-)  (-)  (+)  (-)  (-)   (-)  (+) (+)  (-)   (+)  (+)  (-)  (+)
WHEN a50=>x<=(280,318,340,462,300,305,130,270,260,300,470,295,282,138,320,320);
             y<=(280,420,340,360,300,305,130,270,260,300,470,295,180,240,320,320); --HACE LAGARTIJAS
             w<=(001,003,001,003,001,001,001,001,001,001,001,001,003,003,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
      --              (+) (-)  (+)  (-)  (-)  (+)  (-)  (-)   (-)  (+) (+)  (-)   (+)  (+)  (-)  (+)
WHEN a51=>x<=(280,420,340,360,300,305,130,270,260,300,470,295,180,240,320,320);
             y<=(430,420,150,360,300,305,130,230,450,300,470,295,180,240,170,360);--EMPIEZA A PARARSE1
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a52=>x<=(430,420,150,360,300,305,130,230,450,300,470,295,180,240,170,360);
             y<=(430,470,150,310,300,305,180,160,450,300,420,295,130,290,170,430); --CONTINUA PARANDOSE
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a53=>x<=(430,470,150,310,300,305,180,160,450,300,420,295,130,290,170,430);
             y<=(430,470,195,310,300,305,180,160,405,300,420,295,130,290,170,430); --CONTINUA PARANDOSE2
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a54=>x<=(430,470,195,310,300,305,180,160,405,300,420,295,130,290,170,430);
             y<=(430,470,195,310,360,415,180,160,405,240,420,185,130,290,170,430); --CONTINUA PARANDOSE3
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a55=>x<=(430,470,195,310,360,415,180,160,405,240,420,185,130,290,170,430);
             y<=(430,470,150,310,360,370,180,160,450,240,420,230,130,290,170,430); --CONTINUA PARANDOSE4
             w<=(001,001,001,001,001,001,001,001,001,001,001,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a56=>x<=(430,470,150,310,360,370,180,160,450,240,420,230,130,290,170,430);
             y<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360); --TERMINA DE PARARSE
             w<=(001,001,001,001,001,001,002,001,001,001,002,001,001,001,001,001);
                --(ede,ade,rde,cde,tde,Cde,hde,mde,riz,tiz,hiz,Ciz,aiz,ciz,eiz,miz) rango (120 a 480)
   --              (+) (-)  (+)  (-)  (-)  (+)  (-) (-) (-)  (+)   (+) (-) (+)  (+) (-)  (+)
WHEN a57=>x<=(430,420,150,360,300,305,300,230,450,300,300,295,180,240,170,360);
             y<=(430,360,150,480,300,305,300,230,450,300,300,295,240,120,170,360); --MANOS EN LA CINTURA
             w<=(001,005,001,005,001,001,001,001,001,001,001,001,005,005,001,001);

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

                    END CASE;
            END PROCESS;
```

```
PROCESS(CLK,STATE)
            BEGIN
            IF CLK'EVENT AND CLK='1' THEN
            CASE BB30 IS
            WHEN b1=>
                        IF STATE=a9 THEN
                        HEAD<=150;
                        ELSIF STATE=a10 THEN
                        HEAD<=450;
                        ELSIF STATE=a11 THEN
                       HEAD<=300;
                      ELSIF STATE=a14 THEN
                HEAD<=300;
                ELSE HEAD<=300; END IF;
                        BB30<=b1;
END CASE;
END IF;
END PROCESS;
-----------SENSOR DE DISTANCIA-----------------------------------
 PROCESS(CLK,AUX6,CLK7)
            BEGIN
                        IF CLK'EVENT AND CLK='1' THEN
                        IF CLK7='1'AND AUX6=250 AND AUX8=8 THEN
                        TRIG<='0';ELSE TRIG<='1';END IF;
                        END IF;
            END PROCESS;

            PROCESS(CLK5,ECHO)
            BEGIN
                        IF CLK5'EVENT AND CLK5='1' THEN
                        IF ECHO='1' THEN AUX7<=AUX7+1;TEMPS<=AUX7;END IF;
                        IF ECHO='0' THEN AUX7<=0;LEN<=TEMPS;END IF;
                        END IF;
            END PROCESS;

            PROCESS(CLK,LEN)
            BEGIN
                        IF CLK'EVENT AND CLK='1' THEN
                        IF LEN<80 THEN NEAR<='0';LED5<='0';ELSE NEAR<='1';LED5<='1';
                        END IF;
                        END IF;

            END PROCESS;

            PROCESS(CLK,CLK5,CLK6,CLK7)
            BEGIN
                        IF CLK'EVENT AND CLK='1' THEN AUX5<=AUX5+1;
                                    IF 500>AUX5 THEN CLK5<='0';END IF;
                                    IF AUX5>=500 AND 1000>AUX5 THEN CLK5<='1';END IF;
                                    IF 1000=AUX5 THEN AUX5<=0;END IF;
                        END IF;
                        IF CLK5'EVENT AND CLK5='1' THEN AUX6<=AUX6+1;
                                    IF 500>AUX6 THEN CLK6<='0';END IF;
                                    IF AUX6>=500 AND 1000>AUX6 THEN CLK6<='1';END IF;
                                    IF 1000=AUX6 THEN AUX6<=0;END IF;
                        END IF;
                        IF CLK6'EVENT AND CLK6='1' THEN AUX8<=AUX8+1;
                                    IF 5>AUX8 THEN CLK7<='0';END IF;
                                    IF AUX8>=5 AND 10>AUX8 THEN CLK7<='1';END IF;
                                    IF 10=AUX8 THEN AUX8<=0;END IF;
                        END IF;
            END PROCESS;

END ALGOR;
```

BIBLIOGRAFÍA

[1] Pedroni, Volnei A, *Circuit design with VHDL*, Massachusetts Institute of Technology, The MIT Press; illustrated edition (August, 2004)

[2] Norma Elva Chávez Rodríguez; Jorge Valeriano Assem; Luis Arturo Haro Ruiz; Pablo Roberto Pérez Alcázar; Carlos Rivera Rivera, Jojutla Pacheco Arteaga; Ana María Vázquez Vargas, *Dispositivos Lógicos Programables VLSI*, Facultad de Ingeniería, UNAM

[3] Bob Zeidman, *Introduction to CPLD and FPGA design*, The Chalkboard Network