

**Escuela Superior Politécnica del Litoral**  
**Facultad de Ingeniería Eléctrica y Computación**  
**SISTEMAS DIGITALES II**



**SISTEMAS DIGITALES II**

*Máquina de Helados*

**Integrantes:**

- ☐ Bustamante Apolo Jonathan.
- ☐ Lucio Lara Víctor.

**Paralelo Teórico:**

1

**Paralelo Practico:**

101

## INDICE

DESCRIPCION DEL PROBLEMA.....	4
PROPUESTA 1 .....	5
PROPUESTA 2 .....	6
ANALISIS DE PROPUESTAS DE SOLUCION.....	7
DIAGRAMA DE BLOQUES DEL SISTEMA DIGITAL .....	8
DIAGRAMA ASM .....	9
EXPLICACION DIAGRAMA ASM .....	10
VDHL DEL CONTROLADOR .....	11
EXPLICACION DEL VHDL DE LA MSS .....	14
ASIGNACION DE PINES.....	15
PROGRAMACION DE LA FPGA.....	16
DIAGRAMA DE TIEMPO DE CONTROLADOR .....	17
EXPLICACION DEL DIAGRAMA DE TIEMPO DEL CONTROLADOR.....	17
EXPLICACION DE LOS BLOQUES .....	18
SIMULACIÓN DEL SISTEMA DIGITAL.....	20
EXPLICACION DEL SISTEMA DIGITAL .....	21
ESQUEMA ALTIUM .....	22
EXPLICACIÓN ESQUEMA ALTIUM .....	23
ESQUEMA ELECTRICO PROTEUS.....	24
EXPLICACION DEL ESQUEMA ELECTRICO .....	24
DIAGRAMA ESQUEMATICO EN FISICO.....	25
ANEXOS .....	26
VHDL ANTIREBOTE .....	26
VHDL DECODER_JALEAS .....	26

VHDL PUERTA_OR .....	27
VHDL DECODER_HELADOS .....	27
VHDL REGISTRO_SOSTENIMIENTO .....	28
VHDL CLOCK_DIV .....	29
VHDL DECODER_FRUTAS .....	31
<i>VHDL DECODER_DULCES</i> .....	32
VHDL DIRECCION .....	34
VHDL RAM.....	35
VHDL MUX.....	37
<i>VHDL BINtoBCD</i> .....	37
VHDL SUMADOR.....	39
VHDL 7SEGMENTOS .....	40
VHDL 20 CONSTANTE .....	41
Vínculo de GitHub .....	41
<a href="https://github.com/jeba1797/Proyecto-Helader-a">https://github.com/jeba1797/Proyecto</a> -Helader-a .....	41
Link de Youtube .....	41

## DESCRIPCION DEL PROBLEMA

Se requiere implementar en Plaza Lagos un Sistema Digital que funcione como una máquina expendedora de Helados, donde el usuario podrá elegir los ingredientes que tendrá su helado, y al final se mostrará el precio total a pagar.

Para esto se ha decidido requerir los servicios de la empresa HELADOS S.A., donde el jefe del área técnica de la empresa le pide a usted como técnico diseñar un Sistema Digital, que funcione como una Máquina Expendedora de Helados.

Inicialmente se debe presionar y soltar el botón START, para que el sistema quede listo para operar en un estado de activación.

Al helado se le puede agregar diferentes ingredientes, cada uno su respectivo precio,

- 3 diferentes tipos helados de yogurt: Frutilla (\$1), Chocolate (\$1) y Vainilla (\$0.75).
- Jaleas: Frutilla (0.40), Manjar (\$0.75) y Chocolate (\$0.80).
- Frutas: Frutilla (\$0.25), Cereza (\$0.30), Higo (\$0.40), Durazno (\$0.50), Uva (\$0.20).
- Dulces: Grageas de colores (\$0.15), Oreo (\$0.25), Gomitas (\$0.10) y Granola (\$0.40).

Para los helados y las jaleas, usted contará con botoneras para implementarlo, y para las frutas y Dulces podrá implementarlo con switch. Al comienzo usted seleccionará un tipo de helado y luego un tipo de jalea. Luego procederá a seleccionar las frutas que tendrá su helado y utilizará la botonera Validar para guardar las frutas, y por último validará los dulces que tendrá su helado. En cada selección se mostrarán indicadores como visualización en el estado en el cual se encuentra.

Una vez que valida los dulces, al final se presentará el valor a pagar en display de 7 segmentos, que permanecerá encendido por un tiempo de 5 segundos y luego volverá al estado de activación para un nuevo helado.

El sistema deberá tener un historial de las últimas 20 ventas realizadas. Para esto contará con una botonera que se la podrá presionar y soltar para visualizar el helado vendido junto con el costo total en intervalos de 3 segundos. La consulta se la realizará siempre y cuando no se esté comprando un helado, es decir, en el estado de activación.

El sistema debe tener validaciones para las botoneras (presionar y soltar), y funcionar para clock\_automático y clock\_manual.

El proyecto debe ser descrito COMPLETAMENTE en VHDL, y debe utilizar descripción Estructural o RTL.

## PROPUESTA 1

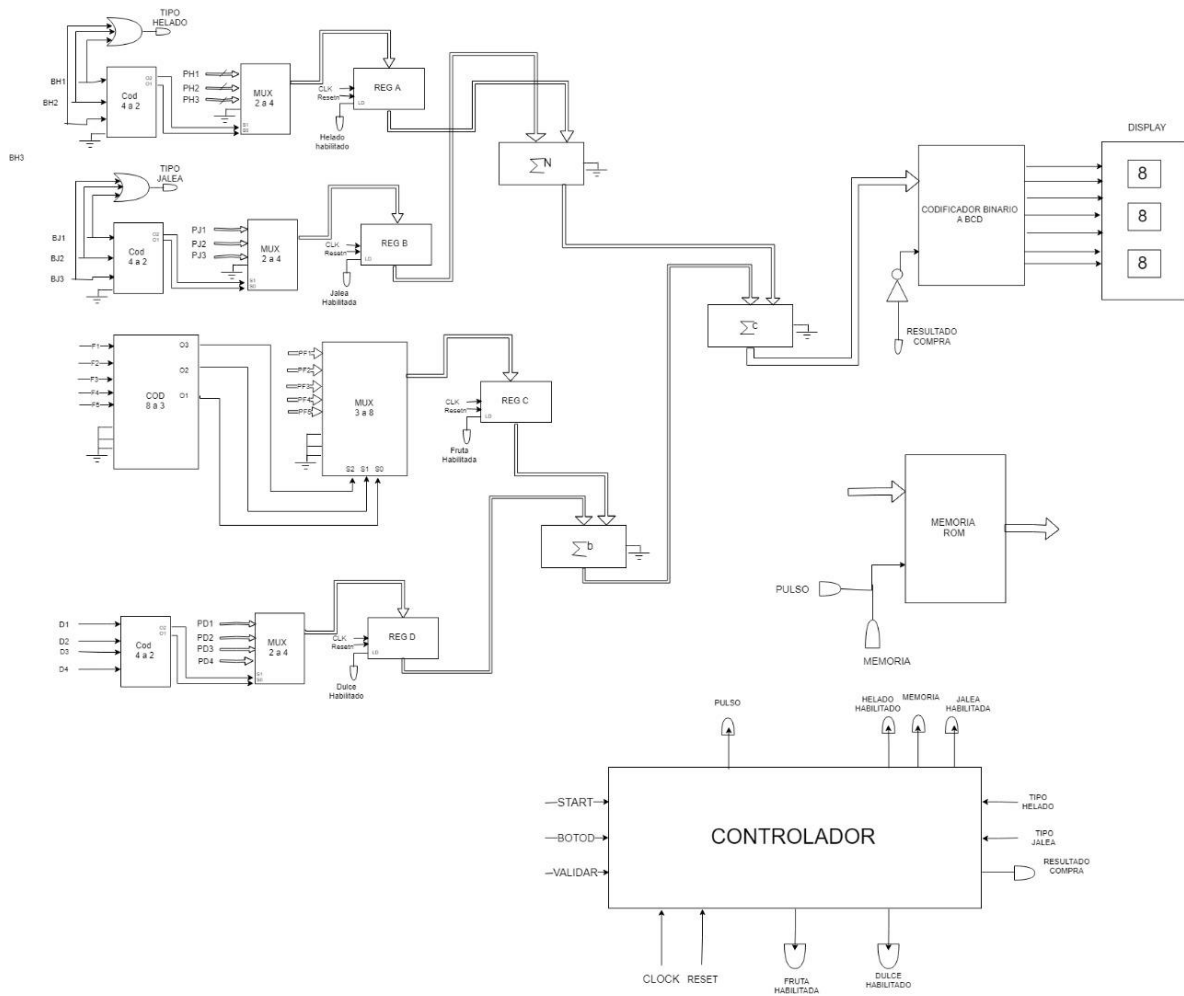


Figura 1 Propuesta 1 para la realización del proyecto.

## PROPUESTA 2

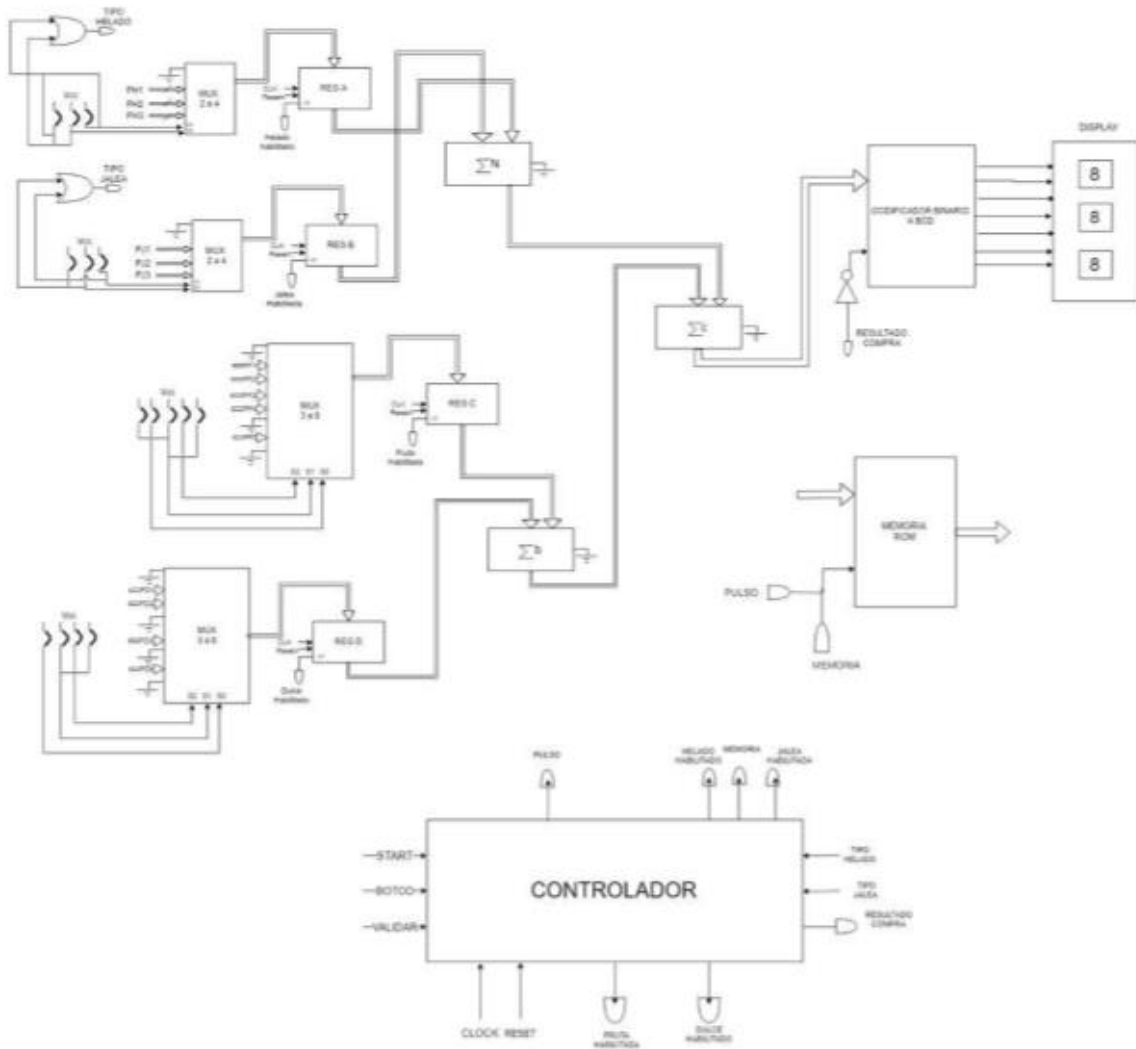


Figura 2 Propuesta 2 para la realización del proyecto.

## ANALISIS DE PROPUESTAS DE SOLUCION

Se ha decidido optar en gran parte por tomar la propuesta numero 1 como método de operación para el sistema digital, debido a que posee ventajas por encima de la segunda propuesta. Por ejemplo, se puede observar la simplicidad en cuanto al funcionamiento y toma de datos respecta. Es recomendable optar por propuestas que impliquen simplicidad y facilidad de operación, de esta manera el proceso de datos se realiza de una manera más ordenada y es fácil de comprender añadiendo el hecho que existe un menor consumo de recursos y elementos lógicos en la tarjeta a utilizar.

Cabe mencionar que hemos decido emplear partes de la segunda propuesta para la presentación de las 20 ventas de helados de la memoria, ya que posee mejoras sobre el 1 método para la presentación de datos. Finalmente se emplearán los leds que sirven para mostrar el estado en el que nos encontramos, para que operen en el momento de presentar las 20 ventas; con ello logramos aprovechar componentes y ahorrar espacio en esquema eléctrico.

## DIAGRAMA DE BLOQUES DEL SISTEMA DIGITAL

En la siguiente ilustración se puede observar el diagrama de bloques que se empleara para la realización de nuestro sistema digital.

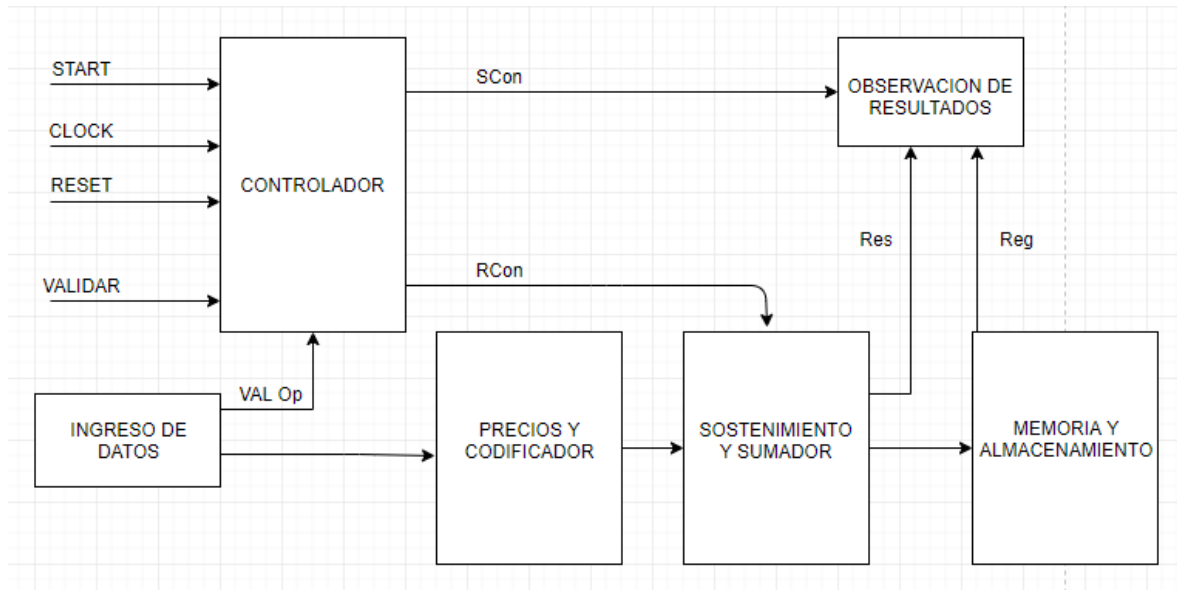


Figura 2 Diagrama de bloques del sistema digital.

El diagrama de bloques presentado permite entender cómo será la operación del sistema digital para la maquina vendedora de helados realizado, continuación se presenta la explicación de la misma:

Una vez el usuario haya tomado las decisiones de qué tipo de helado, fruta, dulces y jaleas desea en el bloque de **INGRESO DE DATOS**; estos son enviados hasta el bloque de **PRECIOS** en donde por medio de codificadores se tomarán los valores respectivos a las opciones tomadas.

El siguiente bloque es el de **SOSTENIMIENTO Y SUMA**, en donde se realizará el sumatorio total de las distintas opciones que fueron tomadas, de esta forma se obtiene el precio final del helado. A la vez el precio es guardado en un banco de **MEMORIA**.

El bloque **CONTROLADOR** cumple la función de presentar los estados ante la validación de una opción tomada por el usuario, este está vinculado con el bloque de **OBSERVACION DE RESULTADOS** ya que es necesario ilustrar el valor final del helado en el display (que también se encuentra vinculado a la **MEMORIA**).



## DIAGRAMA ASM

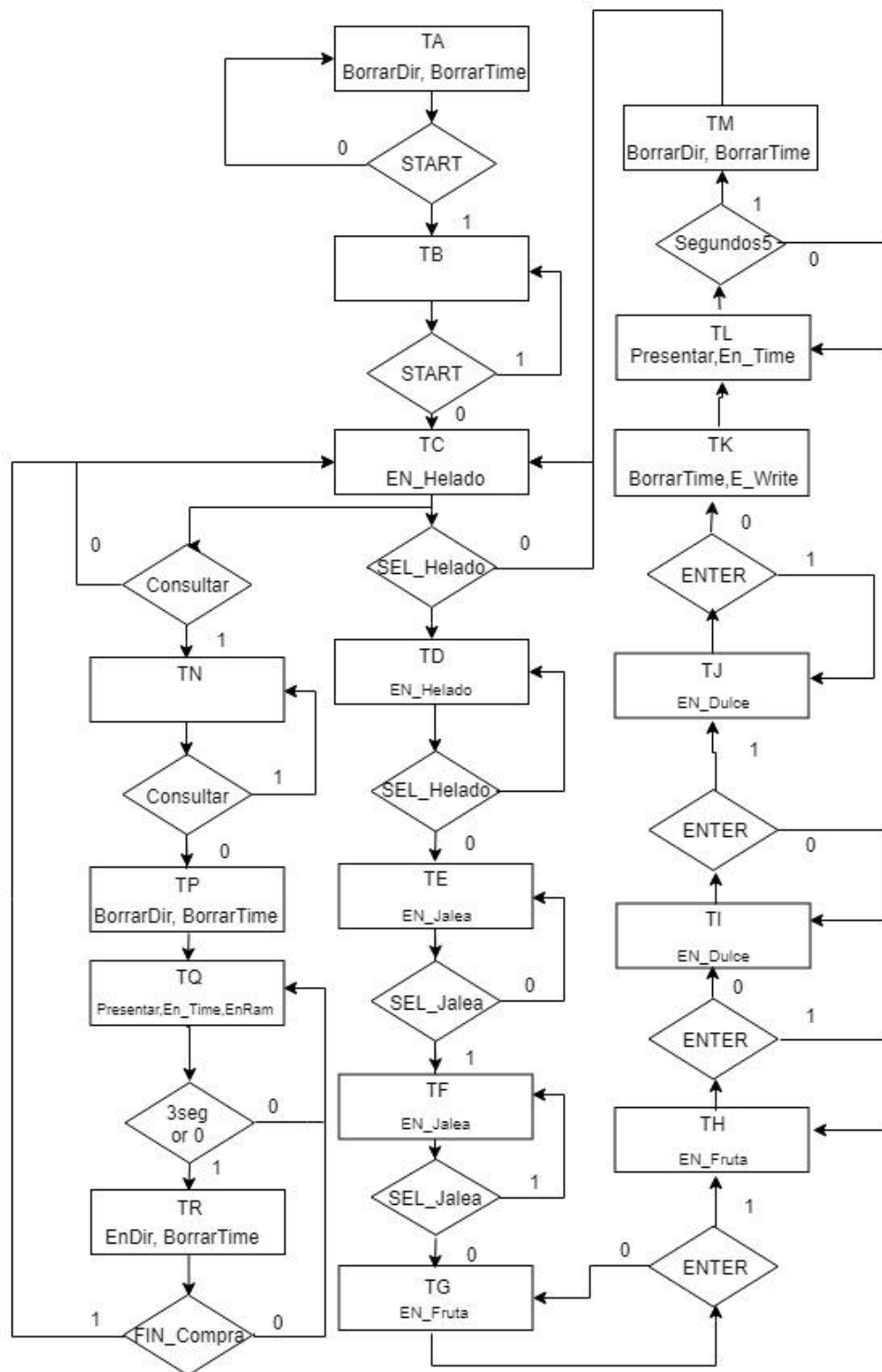


Figura 3 diagrama ASM del controlador.

## EXPLICACION DIAGRAMA ASM

El diagrama ASM realizado permite comprender el funcionamiento y operación del sistema digital creado para la máquina de helados, que se presenta de forma detallada a continuación:

- En el punto de partida el usuario debe presionar y soltar la botonera START para proceder a seleccionar las características del helado que desea comprar
- Seguidamente debe escoger el tipo de helado que desea, ya sea chocolate, frutilla o vainilla, luego el usuario debe validar la opción escogida mediante la botonera (presionando y soltando como forma de validación)
- Seguidamente debe escoger el tipo de jalea que desea, ya sea chocolate, frutilla o manjar, luego el usuario debe validar la opción escogida mediante la botonera (presionando y soltando como forma de validación)
- Una vez haya escogido el tipo de helado y jalea, se procede a escoger la fruta que llevará su helado; cabe mencionar que para este punto será necesaria el uso de la botonera VALIDAR como forma de confirmación de la opción tomada.
- Una vez haya escogida la fruta del helado, se procede a escoger el dulce que llevará su helado; cabe mencionar que para este punto será necesaria el uso de la botonera VALIDAR como forma de confirmación de la opción tomada.
- Finalmente se procede a presentar el precio del helado escogido mediante el display de 7 segmentos durante un tiempo de 5 segundos, para luego empezar en el estado inicial.
- El usuario además podrá observar cuales fueron las ultimas 20 ventas de helados realizadas mediante un banco de memoria que mostrara los datos durante 3 segundos cada uno, para luego regresar en el estado inicial.

## VDHL DEL CONTROLADOR

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity controlador is
    port(
        CLOCK: in std_logic;
        Reset: in std_logic;
        START: in std_logic;
        ENTER: in std_logic;
        CONSULTAR: in std_logic;
        PRESIONO_HELADO: in std_logic;
        PRESIONO_JALEA: in std_logic;
        GUARDADO: in std_logic;
        SEG5: in std_logic;
        SEG3: in std_logic;
        FIN_COMPRA: in std_logic;
        cero: in std_logic;
        ESTADO_HELADO: OUT std_logic;
        ESTADO_JALEA: OUT std_logic;
        ESTADO_FRUTAS: OUT std_logic;
        ESTADO_DULCES: OUT std_logic;
        EN_DIRECCION: OUT std_logic;
        BORRAR_DIRECCION: OUT std_logic;
        EN_RAM: OUT std_logic;
        WRITE_EN: OUT std_logic;
        EN_TIEMPO: OUT std_logic;
        MOSTRAR: OUT std_logic_VECTOR(1 DOWNTO 0);
        BORRAR_TIEMPO : OUT std_logic
    );
end controlador;
architecture COMPORTAMIENTO of controlador is
    TYPE estado IS (Ta,Tb,Tc,Td,Te,Tf,Tg,Th,Ti,Tj,Tk,Tl,Tm,Tn,Tp,Tq,Tr,Ts);
    SIGNAL y : estado;
begin
    PROCESS (Reset,CLOCK)
    begin
        if Reset='0' then y<=Ta;
        elsif (CLOCK'event and Clock='1') then
            case y is
                when Ta=> if (START='1') then y<=Tb;

```

```
else y<=Ta;
end if;
when Tb=> if (START='0') then y<=Tc;
else y<=Tb;
end if;
when Tc=> if (PRESIONO_HELADO='1') then y<=Td;
elseif (CONSULTAR='1') then y<=Tn;
else y<=Tc;
end if;
when Td=> if (PRESIONO_HELADO='0') then y<=Te;
else y<=Td;
end if;
when Te=> if (PRESIONO_JALEA='1') then y<=Tf;
else y<=Te;
end if;
when Tf=> if (PRESIONO_JALEA='0') then y<=Tg;
else y<=Tf;
end if;
when Tg=> if (ENTER='1') then y<=Th;
else y<=Tg;
end if;
when Th=> if (ENTER='0') then y<=Ti;
else y<=Th;
end if;
when Ti=> if (ENTER='1') then y<=Tj;
else y<=Ti;
end if;
when Tj=> if (ENTER='0') then y<=Tk;
else y<=Tj;
end if;
when Tk=> y<=Tl;
when Tl=> if (seg5='1') then y<=Tm;
else y<=Tl;
end if;
when Tm=> y<=Tc;
when Tn=> if (CONSULTAR='0') then y<=Tp;
else y<=Tn;
end if;
when Tp=> y<=Tq;
when Tq=> if (seg3='1') or (cero='0') then y<=Tr;
else y<=Tq;
end if;
when Tr=> if (FIN_COMPRA='1') then y<=Tc;
else y<=Ts;
```

```

                                end if;
                        when Ts=> y<=Tq;
                end case;
end if;
END PROCESS;
PROCESS (y)
begin
        ESTADO_HELADO<='0';
        ESTADO_JALEA<='0';
        ESTADO_FRUTAS<='0';
        ESTADO_DULCES<='0';
        EN_DIRECCION<='0';
        BORRAR_DIRECCION<='0';
        EN_RAM<='0';
        WRITE_EN<='0';
        EN_TIEMPO<='0';
        BORRAR_TIEMPO <='0';
        MOSTRAR<="00";
        case y is
                when Ta=> BORRAR_DIRECCION<='1'; BORRAR_TIEMPO
<='1';
                        when Tb=>
                                when Tc=> ESTADO_HELADO<='1';
                                when Td=> ESTADO_HELADO<='1';
                                when Te=> ESTADO_JALEA<='1';
                                when Tf=> ESTADO_JALEA<='1';
                                when Tg=> ESTADO_FRUTAS<='1';
                                when Th=> ESTADO_FRUTAS<='1';
                                when Ti=> ESTADO_DULCES<='1';
                                when Tj=> ESTADO_DULCES<='1';
                                when Tk=> BORRAR_TIEMPO <='1'; WRITE_EN<='1';
                                when Tl=> EN_TIEMPO<='1'; MOSTRAR<="01";
                                when Tm=> BORRAR_TIEMPO
<='1';BORRAR_DIRECCION<='1';
                                        when Tn=>
                                                when Tp=> BORRAR_TIEMPO <='1';
BORRAR_DIRECCION<='1';
                                                when Tq=> EN_TIEMPO<='1'; MOSTRAR<="11";
EN_RAM<='1';
                                                when Tr=>
                                                when Ts=> EN_DIRECCION<='1'; BORRAR_TIEMPO <='1';
                                end case;
        END PROCESS;
END COMPORTAMIENTO;

```

## EXPLICACION DEL VHDL DE LA MSS

Al inicializar la máquina expendedora de helados el usuario se encontrará en el estado Ta, en el mismo habilitará los reset para los contadores de la misma y para la memoria que almacena los datos de venta. Para pasar al siguiente es necesario que se presione y suelte la tecla START. En Tc preguntara si ha sido seleccionado el tipo de helado, en tal caso habilita los registros del mismo caso contrario se mantiene en el estado Tc a menos que se presione la botonera CONSULTAR en caso de querer observar las 20 últimas ventas. En Te preguntara si ha sido seleccionado el tipo de jalea, en tal caso habilita los registros del mismo caso contrario se mantiene en el estado Te. En Tg preguntara si han sido seleccionado las frutas por medio de la botonera ENTER, en tal caso habilita los registros del mismo caso contrario se mantiene en el estado Tg. En Ti preguntara si han sido seleccionado los dulces por medio de la botonera ENTER, en tal caso habilita los registros del mismo caso contrario se mantiene en el estado Ti.

En el estado Tk resetea los tiempos del contador, para seguidamente pasar al estado TI donde mostrará el valor a pagar durante 5 segundos; seguidamente transcurre al estado tm, para luego regresar estado de activación, borrando el contador anticipadamente. En el estado Tp reinicia el contador de dirección para proceder a mostrar las 20 compras, en el estado Tq valida que se muestren los precios de venta durante 3 segundos; mientras que en Tr valida que se hayan presentado todas las 20 últimas compras.

## PIN PLANNER

### Top View - Wire Bond Cyclone IV E - EP4CE22F17C6

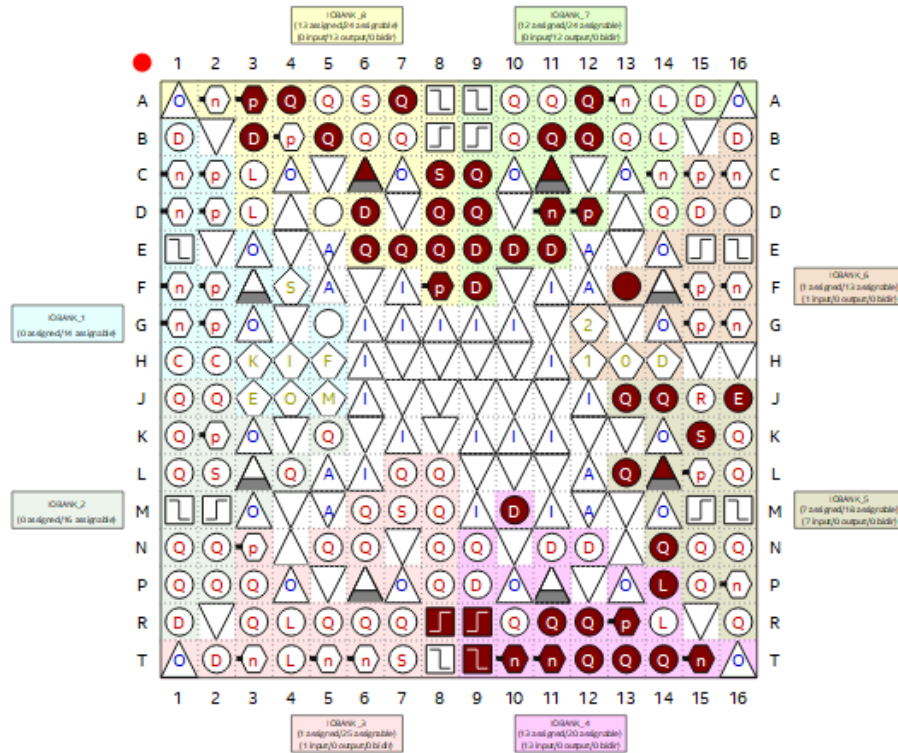


Figura 4 Bloques empleados de la tarjeta Deo-Nano.

## ASIGNACION DE PINES

in	CHOCOLATE	Input	PIN_T15	4	B4_NO	PIN_T15	2.5 V	8mA (default)	
in	CHOCOLATE_J	Input	PIN_T12	4	B4_NO	PIN_T12	2.5 V	8mA (default)	
in	CLOCK	Input	PIN_R8	3	B3_NO	PIN_R8	2.5 V	8mA (default)	
in	CONSULTAR	Input	PIN_T11	4	B4_NO	PIN_T11	2.5 V	8mA (default)	
out	DISA[1]	Output	PIN_B12	7	B7_NO	PIN_B12	2.5 V	8mA (default)	2 (default)
out	DISA[2]	Output	PIN_D12	7	B7_NO	PIN_D12	2.5 V	8mA (default)	2 (default)
out	DISA[3]	Output	PIN_D11	7	B7_NO	PIN_D11	2.5 V	8mA (default)	2 (default)
out	DISA[4]	Output	PIN_A12	7	B7_NO	PIN_A12	2.5 V	8mA (default)	2 (default)
out	DISA[5]	Output	PIN_B11	7	B7_NO	PIN_B11	2.5 V	8mA (default)	2 (default)
out	DISA[6]	Output	PIN_C11	7	B7_NO	PIN_C11	2.5 V	8mA (default)	2 (default)
out	DISA[7]	Output	PIN_E10	7	B7_NO	PIN_E10	2.5 V	8mA (default)	2 (default)
out	DISB[1]	Output	PIN_E11	7	B7_NO	PIN_E11	2.5 V	8mA (default)	2 (default)
out	DISB[2]	Output	PIN_D9	7	B7_NO	PIN_D9	2.5 V	8mA (default)	2 (default)
out	DISB[3]	Output	PIN_C9	7	B7_NO	PIN_C9	2.5 V	8mA (default)	2 (default)
out	DISB[4]	Output	PIN_E9	7	B7_NO	PIN_E9	2.5 V	8mA (default)	2 (default)
out	DISB[5]	Output	PIN_F9	7	B7_NO	PIN_F9	2.5 V	8mA (default)	2 (default)
out	DISB[6]	Output	PIN_F8	8	B8_NO	PIN_F8	2.5 V	8mA (default)	2 (default)
out	DISB[7]	Output	PIN_E8	8	B8_NO	PIN_E8	2.5 V	8mA (default)	2 (default)

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
DISB[4]	Output	PIN_E9	7	B7_NO	PIN_E9	2.5 V		8mA (default)	2 (default)
DISB[5]	Output	PIN_F9	7	B7_NO	PIN_F9	2.5 V		8mA (default)	2 (default)
DISB[6]	Output	PIN_F8	8	B8_NO	PIN_F8	2.5 V		8mA (default)	2 (default)
DISB[7]	Output	PIN_E8	8	B8_NO	PIN_E8	2.5 V		8mA (default)	2 (default)
DISC[1]	Output	PIN_D8	8	B8_NO	PIN_D8	2.5 V		8mA (default)	2 (default)
DISC[2]	Output	PIN_E7	8	B8_NO	PIN_E7	2.5 V		8mA (default)	2 (default)
DISC[3]	Output	PIN_E6	8	B8_NO	PIN_E6	2.5 V		8mA (default)	2 (default)
DISC[4]	Output	PIN_C8	8	B8_NO	PIN_C8	2.5 V		8mA (default)	2 (default)
DISC[5]	Output	PIN_C6	8	B8_NO	PIN_C6	2.5 V		8mA (default)	2 (default)
DISC[6]	Output	PIN_A7	8	B8_NO	PIN_A7	2.5 V		8mA (default)	2 (default)
DISC[7]	Output	PIN_D6	8	B8_NO	PIN_D6	2.5 V		8mA (default)	2 (default)
ESTADO_DULCES	Output	PIN_B3	8	B8_NO	PIN_B3	2.5 V		8mA (default)	2 (default)
ESTADO_FRUTAS	Output	PIN_A3	8	B8_NO	PIN_A3	2.5 V		8mA (default)	2 (default)
ESTADO_HELADO	Output	PIN_B5	8	B8_NO	PIN_B5	2.5 V		8mA (default)	2 (default)
ESTADO_JALEA	Output	PIN_A4	8	B8_NO	PIN_A4	2.5 V		8mA (default)	2 (default)
FRU_CEREZA	Input	PIN_J13	5	B5_NO	PIN_J13	2.5 V		8mA (default)	
FRU_DURAZNO	Input	PIN_J16	5	B5_NO	PIN_J16	2.5 V		8mA (default)	
FRU_FRUTILLA	Input	PIN_J14	5	B5_NO	PIN_J14	2.5 V		8mA (default)	

Activar  
Ve a Conf

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength
FRU_CEREZA	Input	PIN_J13	5	B5_NO	PIN_J13	2.5 V		8mA (default)
FRU_DURAZNO	Input	PIN_J16	5	B5_NO	PIN_J16	2.5 V		8mA (default)
FRU_FRUTILLA	Input	PIN_J14	5	B5_NO	PIN_J14	2.5 V		8mA (default)
FRU_HIGO	Input	PIN_K15	5	B5_NO	PIN_K15	2.5 V		8mA (default)
FRU_UVA	Input	PIN_L13	5	B5_NO	PIN_L13	2.5 V		8mA (default)
FRUTILLA	Input	PIN_R9	4	B4_NO	PIN_R9	2.5 V		8mA (default)
FRUTILLA_J	Input	PIN_T13	4	B4_NO	PIN_T13	2.5 V		8mA (default)
GOMITAS	Input	PIN_L14	5	B5_NO	PIN_L14	2.5 V		8mA (default)
GRAJEAS	Input	PIN_M10	4	B4_NO	PIN_M10	2.5 V		8mA (default)
GRANOLA	Input	PIN_P14	4	B4_NO	PIN_P14	2.5 V		8mA (default)
MANJAR_J	Input	PIN_R13	4	B4_NO	PIN_R13	2.5 V		8mA (default)
MANUAL	Input	PIN_R11	4	B4_NO	PIN_R11	2.5 V		8mA (default)
OREO	Input	PIN_N14	5	B5_NO	PIN_N14	2.5 V		8mA (default)
RESET	Input	PIN_T9	4	B4_NO	PIN_T9	2.5 V		8mA (default)
SELECTOR	Input	PIN_T10	4	B4_NO	PIN_T10	2.5 V		8mA (default)
START	Input	PIN_F13	6	B6_NO	PIN_F13	2.5 V		8mA (default)
VAINILLA	Input	PIN_T14	4	B4_NO	PIN_T14	2.5 V		8mA (default)
VALIDAR	Input	PIN_R12	4	B4_NO	PIN_R12	2.5 V		8mA (default)

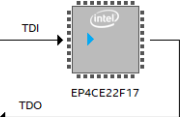
## PROGRAMACION DE LA FPGA

Hardware Setup...
USB-Blaster [USB-0]
Mode: JTAG
Progress: 100% (Successful)

☐ Enable real-time ISP to allow background programming when available

Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down

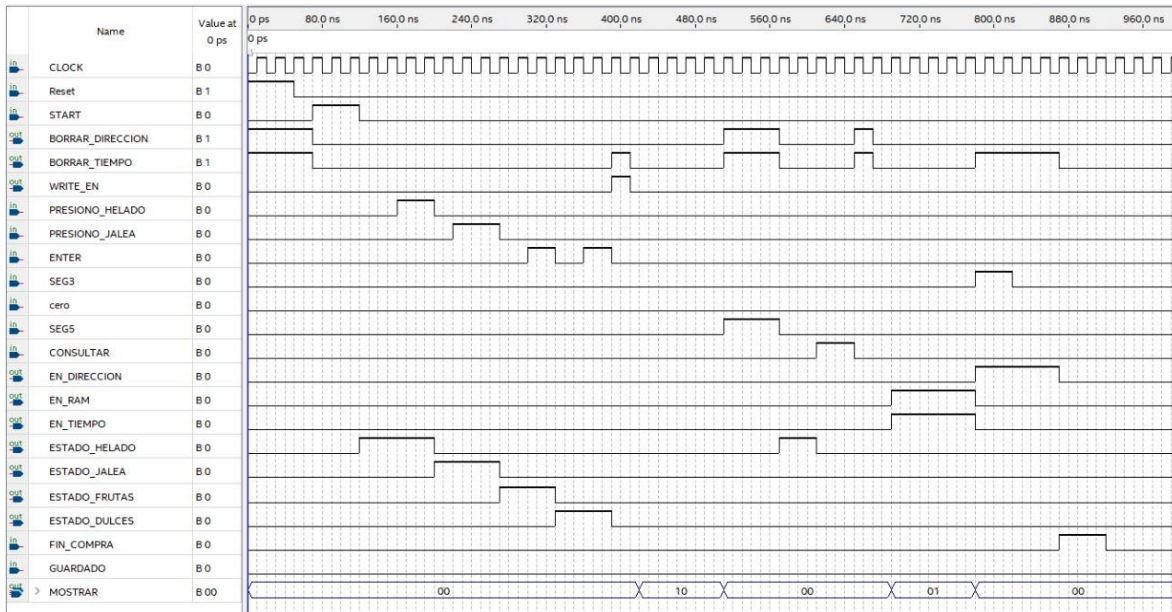
File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/HELAIDOS...	EP4CE22F17	001E2BAA	001E2BAA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>





## DIAGRAMA DE TIEMPO DE CONTROLADOR

Compra del Helado y consulta.



## EXPLICACION DEL DIAGRAMA DE TIEMPO DEL CONTROLADOR

En la ilustración podemos observar la simulación del controlador en el momento de realizar la compra del helado, en primer lugar, el usuario debe presionar la tecla Start para darle inicio a la compra, seguidamente debe elegir el tipo de helado que el desee, es decir: frutilla, chocolate o vainilla, lo cual activa la señal PRESIONO\_HELADO. Seguidamente debe elegir el tipo de jalea, lo cual deberá mostrar a PRECIONO\_JALEA como un alto. Finalmente debe seleccionar los tipos de frutas y dulces que requiere, para luego presionar la tecla validar activando la señal ENTER, de esta forma finalizando la compra.

A cada paso de selección debe mostrarse en alto los indicadores de estado de Helado, Fruta, Jalea y Dulce respectivamente. Una vez finalizada la compra, debe activarle la señal de presentar hasta que se hayan transcurrido los 5 segundos de presentación del precio.

Por otra parte, para el proceso de presentación de las ultimas 20 compras, el usuario debe accionar la entrada Botón, de esta forma el sistema accede a la memoria y presenta en el display de 7 segmentos las 20 compras realizadas al final; lo cual se logra por medio de un contador. Cabe mencionar que tales valores se deben presentar por medio de la salida denominada memoria, cada vez que la entrada 3 segundos se active.

## **EXPLICACION DE LOS BLOQUES**

### **Control**

El control cumple la función de ser el controlador del proyecto, recepta las salidas de los bloques anteriores y con respecto a tales responde de una determinada manera, de esta forma reinicia el sistema o presenta los estados de venta de los helados.

### **Tiempo**

El tiempo permite que nuestro código de forma manual procesa al siguiente estado de forma continua, sin que se requiera una determinada señal de reloj. De esta forma el sistema procede al siguiente estado de venta, a partir del momento de inicio.

### **Antirebote**

El antirebote sirve que las tomas de datos desde las botoneras de los parámetros de venta sean realizadas de forma correcta, al momento que son activadas mediante su entrada.

### **Sumador**

El bloque sumador sirva para realizar la suma total de los parámetros de venta como el tipo de helado, fruta, jalea y dulces respectivamente; posteriormente procederá a presentar al precio por medio del bloque BDCtoSEVEN.

### **Registro Sostenimiento**

El registro sirve para almacenar el valor del parámetro entrante respectivamente de cada variable. De esta manera es posible el empleo de dicha variable para el próximo estado de programa, por ejemplo, para realizar el sumatorio total.

### **JaleasDEC**

El jaleasDEC cumple la función de presentar el precio del tipo de la jalea seleccionada en base a la opción que fue tomada por el usuario, es decir entregara el precio que posteriormente pasara por el registro.

### **HeladosDEC**

El HeladosDEC cumple la función de presentar el precio del tipo de Helado seleccionado en base a la opción que fue tomada por el usuario, es decir entregara el precio que posteriormente pasara por el registro.

### **7 Segmentos**

El bloque BCDtoSEVEN permite que se muestre el precio total del helado escogido por el usuario, por medio de 3 displays de 7 segmentos.

### **Dec\_Dul**

El Dec-Dul cumple la función de presentar el precio del tipo de Dulce seleccionado en base a la opción que fue tomada por el usuario, es decir entregara el precio que posteriormente pasara por el registro.

### **Bin\_to\_BCD**

El bloque Bin\_to\_BCD cumple la función de presentar el valor de la entrada que se encuentra en binario, a BCD para realizar la siguiente operación respectiva, y posteriormente mostrar la suma total.

### **MUX**

El comparador contrasta dos señales, en el caso del proyecto comparara 5 con el valor de los segundos transcurridos a partir de la última selección, de esta forma entrega un alto cuando se hayan cumplido los 5 segundos para presentar el precio.

### **Dec\_Fru**

El Dec-Fru cumple la función de presentar el precio del tipo de la fruta seleccionada en base a la opción que fue tomada por el usuario, es decir entregara el precio que posteriormente pasara por el registro.

### **CLOCK\_DIV\_50**

El CLOCK\_DIV\_50 cumple la función de presentar distintas frecuencias de operación debido a que en distintos puntos del sistema se requiere operar con valores de frecuencia distintos respectivamente.

### **MUX\_2to1**

El bloque MUX\_2to1 permite la selección de una de dos opciones en función de un parámetro de entrada, es decir permite representar un selector de entradas.

### **RAM**

El bloque RAM sirve para almacenar los datos de venta del helado, incluyendo que helado se vendió y el precio final del mismo; que posteriormente se presentara. El bloque consta con la programación automática para el desplazamiento de datos al ser guardados.

### **Puerta Or**

Bloque diseñado que toma las señales de entrada para validar que se está seleccionando una opción de compra y realiza la suma lógica de estas y la muestra a su salida como habilitador de un registro.

### **Dirección**

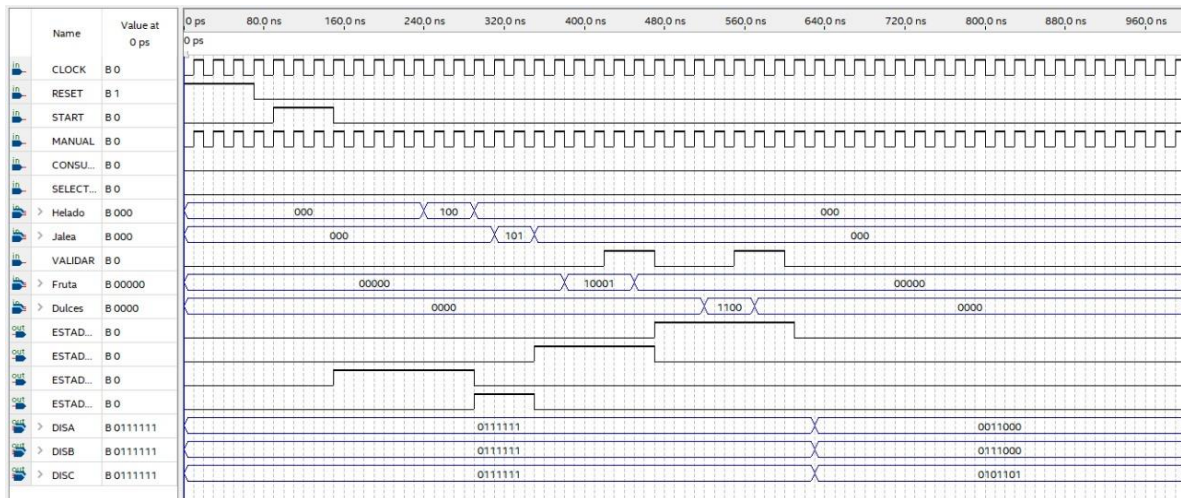
Bloque encargado de definir la entrada de direccionamiento de la RAM, para almacenar el dato correspondiente después de una compra.

## Veinte\_Constante

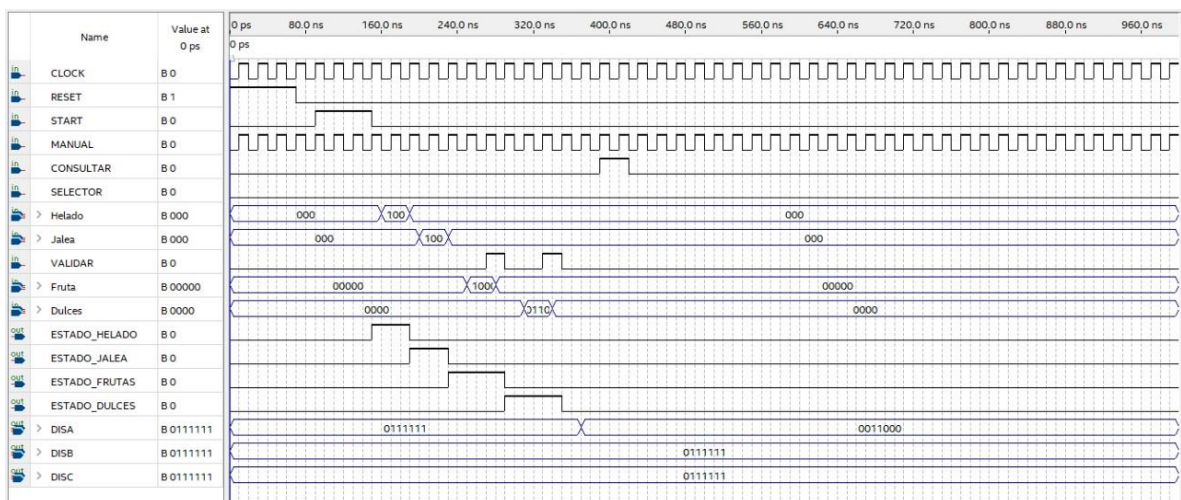
Corresponde a la referencia de 20, cuya funcionalidad es a de establecer que se hayan presentado las 20 últimas compras realizadas de helados.

## SIMULACIÓN DEL SISTEMA DIGITAL

### Venta de Helado



### Consulta de Ventas



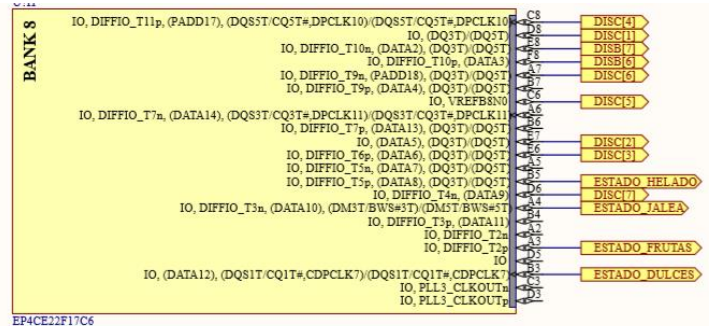
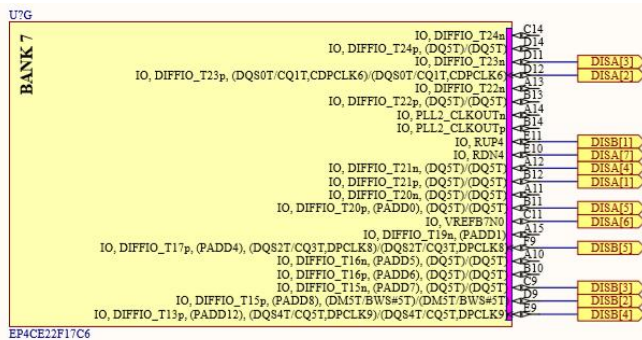
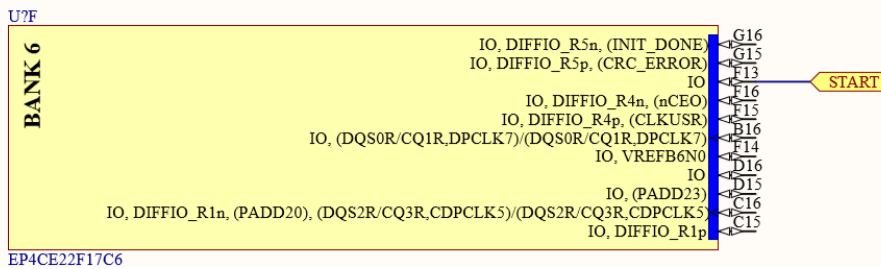
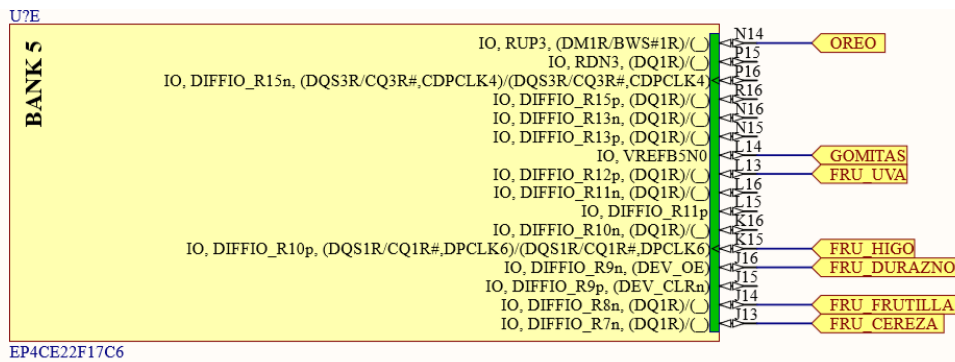
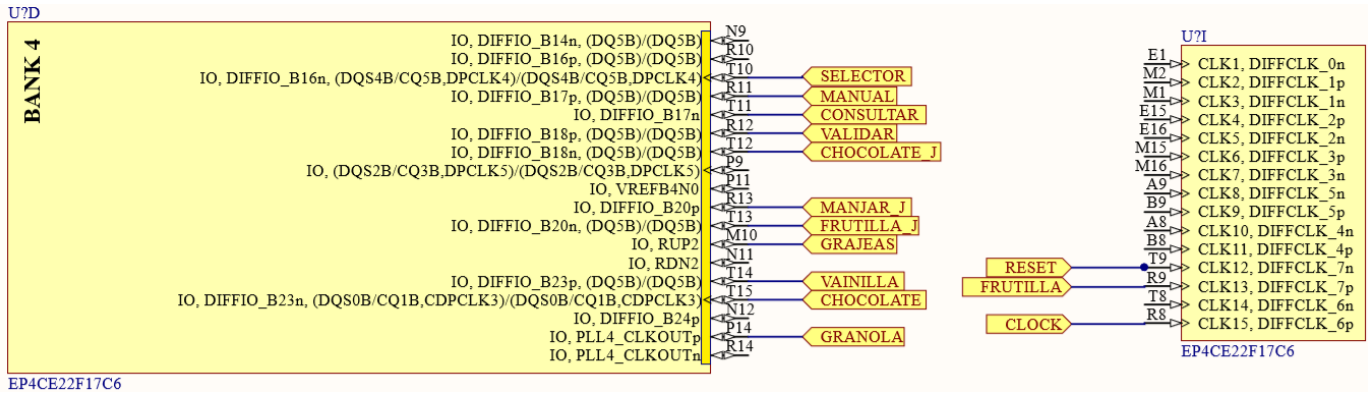
## EXPLICACION DEL SISTEMA DIGITAL

En el sistema digital se observan únicamente las entradas y salidas del sistema total, sin mostrar ninguna de las señales internas. En la compra del helado, se observa la señal reset está en activación, luego se procede a activar la tecla start, donde una vez soltada la maquina queda en el estado de activación. Como se observa se habilita señal de estado de helado encendiendo el led respectivo. Cuando se presiona un helado y una jalea se activa la respectiva salida de estado indicando la etapa en la que se encuentra el usuario, funcionando de igual manera cuando se presiona y suelta la señal validar en la selección de frutas y dulces. Una vez terminado el estado de dulces se activan los displays mostrando el resultado de la compra tal como se observa al final del waveform.

Para realizar la consulta nos debemos encontrar en el estado de compra de helado seleccionando el sabor del mismo, aquí se activa la señal consultar la cual mostrara las ventas realizadas en los displays durante tres segundos recorriendo las memorias del sistema.



## ESQUEMA ALTUM



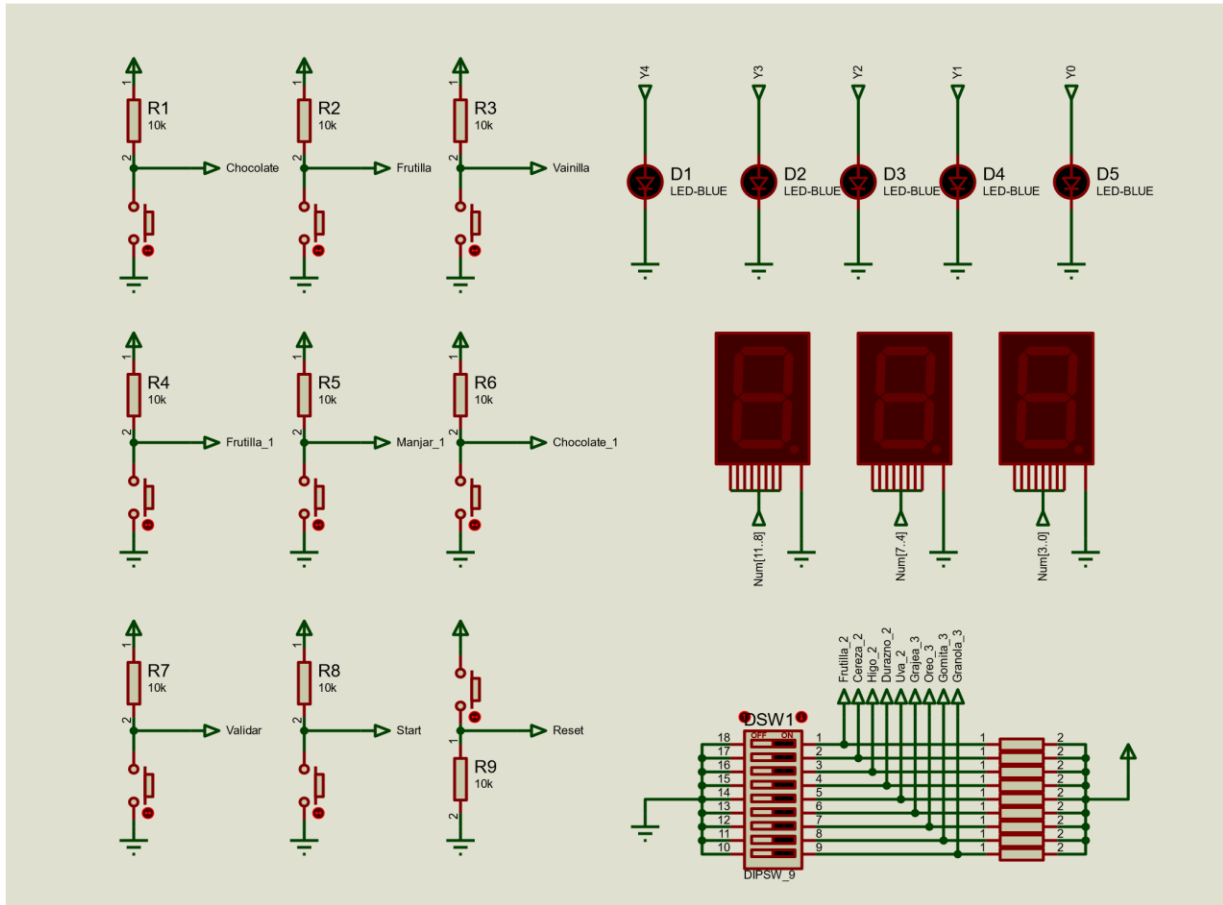
## EXPLICACIÓN ESQUEMA ALTIIUM

En el esquema se puede observar los bloques representativos de la tarjeta Deo-Nano, donde se encuentran ubicados los diferentes pines que dispone la tarjeta. Se observa que solo se emplearon los bancos 4, 5, 6, 7 y 8 de la tarjeta.

En los bancos 7 y 8 se ubicaron las salidas del sistema digital completo, en donde se incluye el control de los displays de 7 segmentos y los leds indicadores de estado.

En los bancos 4, 5, y 6 se ubicaron las entradas del sistema digital, se conectaron las señales de sabores de helado, sabores de jaleas, frutas y dulces que tendrá el helado. Recibe también la señal de la botonera validar y memoria; así como las señales de funcionamiento básico como reset, start, clock manual y selección del clock del sistema.

## ESQUEMA ELECTRICO PROTEUS



## EXPLICACION DEL ESQUEMA ELECTRICO

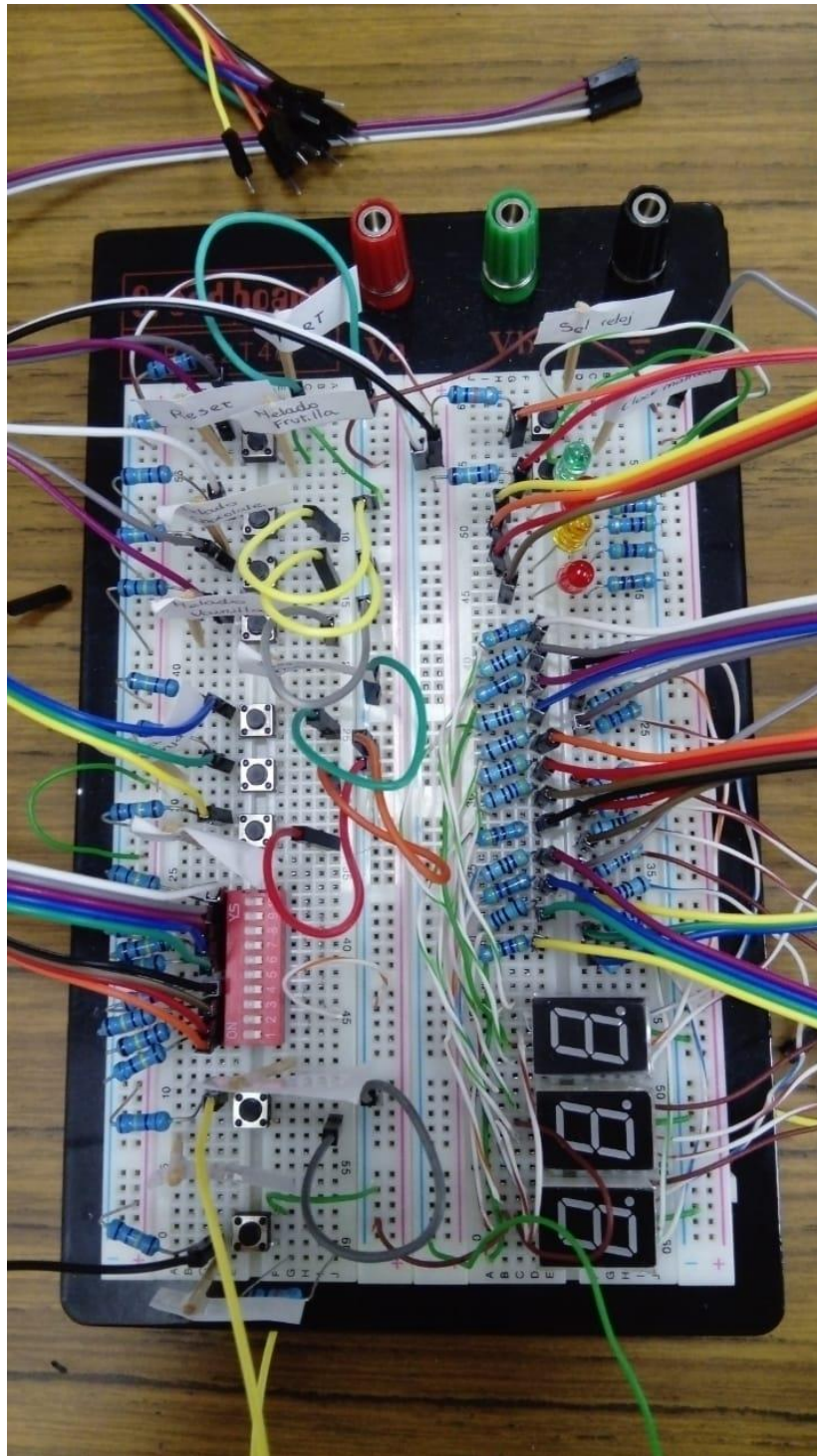
En el esquemático se observan tres botoneras las cuales se emplean para la selección del helado, también se cuentan con tres botoneras para la selección de jaleas. Se emplea un switch para seleccionar tanto las frutas como dulces que tendrá en helado los cuales se validan con la botonera validar.

El sistema se inicializa con la pulsación de la botonera reset y luego start, habilitante para la compra del helado. Se cuenta también con la presencia de la botonera de memoria para seleccionar que se desea presentar las últimas 20 compras realizadas. De igual manera existen dos botoneras, que permiten la selección del reloj manual o automático y la generación del clock manual como tal

Se presentan 3 display de 7 segmentos para la muestra de precios de los helados, así como leds indicadores de estado



### DIAGRAMA ESQUEMATICO EN FISICO



## ANEXOS

### VHDL ANTIREBOTE

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

-- Debounce Pushbutton: Filters out mechanical switch bounce for around
40Ms.
ENTITY ANTIREBOTE IS
    PORT(PB_N, CLOCK_100Hz    : IN  STD_LOGIC;
          PB_SIN_REBOTE       : OUT STD_LOGIC);
END ANTIREBOTE;

ARCHITECTURE a OF ANTIREBOTE IS
    SIGNAL SHIFT_PB           : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN

    -- Debounce clock should be approximately 10ms or 100Hz
    PROCESS
    BEGIN
        WAIT UNTIL (clock_100Hz'EVENT) AND (clock_100Hz = '1');
        -- Use a shift register to filter switch contact bounce
        SHIFT_PB(2 DOWNTO 0) <= SHIFT_PB(3 DOWNTO 1);
        SHIFT_PB(3) <= NOT PB_N;
        IF SHIFT_PB(3 DOWNTO 0) = "0000" THEN
            PB_SIN_REBOTE <= '0';
        ELSE
            PB_SIN_REBOTE <= '1';
        END IF;
    END PROCESS;
END a;

```

### VHDL DECODER\_JALEAS

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DECODER_JALEAS is

port (
    a : in std_logic;
    b : in std_logic;
    c : in std_logic;
    salida : out std_logic_vector(8 downto 0)

```

```
);  
end DECODER_JALEAS;  
  
architecture sol of DECODER_JALEAS is  
  
begin  
  
salida <= "00000" when (a='1') ELSE  
           "00000" when (b='1') ELSE  
           "00000" when (c='1' ) ELSE  
           "00000";  
  
end sol;
```

### **VHDL PUERTA\_OR**

```
library ieee;  
use ieee.std_logic_1164.all;
```

```
Entity pUERTA_OR is  
  Port(  
    a : in std_logic;  
    b : in std_logic;  
    c : in std_logic;  
    salida: out std_logic);  
end pUERTA_OR;
```

Architecture SOLUTION of pUERTA\_OR is

```
Begin  
  salida<= a or b or c;  
end SOLUTION;
```

### **VHDL DECODER\_HELADOS**

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

entity DECODER\_HELADOS is

```
port (  
  a : in std_logic;  
  b : in std_logic;  
  c : in std_logic;  
  
  salida : out std_logic_vector(8 downto 0)
```

```

);
end DECODER_HELADOS;

architecture sol of DECODER_HELADOS is

begin

salida <= "00000" when (a='1') ELSE
           "00000" when (b='1') ELSE
           "00000" when (c='1') ELSE
           "00000";

end sol;
```

## **VHDL REGISTRO\_SOSTENIMIENTO**

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY registro_sostenimiento IS
    PORT(clock,reset,enable: IN STD_LOGIC;
          Ent : IN STD_LOGIC_VECTOR(8 downto 0);
          Q : OUT STD_LOGIC_VECTOR (8 downto 0));
END registro_sostenimiento;

ARCHITECTURE sol OF registro_sostenimiento IS
    SIGNAL temp: STD_LOGIC_VECTOR(8 downto 0);
BEGIN
    PROCESS(clock,reset)
    BEGIN
        if reset='0' then temp<="000000000";
        elsif (clock'event and clock='1') then
            if(enable='1') then
                temp<=Ent;
            end if;
        end if;
    end process;
    Q<=temp;
END sol;
```

## VHDL CLOCK\_DIV

```
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;
```

```
ENTITY CLOCK_DIV IS
```

```
  PORT
```

```
    ( CLOCK_50MHz  :IN  STD_LOGIC;
      CLOCK_1MHz   :OUT STD_LOGIC;
      CLOCK_100KHz :OUT STD_LOGIC;
      CLOCK_10KHz  :OUT STD_LOGIC;
      CLOCK_1KHz   :OUT STD_LOGIC;
      CLOCK_100Hz  :OUT STD_LOGIC;
      CLOCK_10Hz   :OUT STD_LOGIC;
      CLOCK_1Hz    :OUT STD_LOGIC);
```

```
END CLOCK_DIV;
```

```
ARCHITECTURE a OF CLOCK_DIV IS
```

```
  SIGNAL count_1Mhz: STD_LOGIC_VECTOR(5 DOWNTO 0);
  SIGNAL count_100KHz, count_10KHz, count_1KHz: STD_LOGIC_VECTOR(2
DOWNTO 0);
  SIGNAL count_100hz, count_10hz, count_1hz: STD_LOGIC_VECTOR(2
DOWNTO 0);
  SIGNAL clock_1Mhz_int, clock_100KHz_int, clock_10KHz_int, clock_1KHz_int:
STD_LOGIC;
  SIGNAL clock_100hz_int, clock_10hz_int, clock_1hz_int: STD_LOGIC;
```

```
BEGIN
```

```
  PROCESS
```

```
  BEGIN
```

```
-- Divide by 24
```

```
  WAIT UNTIL clock_50Mhz'EVENT and clock_50Mhz = '1'; -- 24 Mhz
```

```
  IF count_1Mhz < 49 THEN
```

```
    count_1Mhz <= count_1Mhz + 1;
```

```
  ELSE
```

```
    count_1Mhz <= "000000";
```

```
  END IF;
```

```
  IF count_1Mhz < 4 THEN
```

```
    clock_1Mhz_int <= '0';
```

```
  ELSE
```

```
    clock_1Mhz_int <= '1';
```

```
  END IF;
```

```
-- Ripple clocks are used in this code to save prescalar hardware
```

```
-- Sync all clock prescalar outputs back to master clock signal
```

```
  clock_1Mhz <= clock_1Mhz_int;
```

```
  clock_100KHz <= clock_100KHz_int;
```

```
  clock_10KHz <= clock_10KHz_int;
```

```

clock_1Khz <= clock_1Khz_int;
clock_100hz <= clock_100hz_int;
clock_10hz <= clock_10hz_int;
clock_1hz <= clock_1hz_int;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_1Mhz_int'EVENT and clock_1Mhz_int = '1';
    IF count_100Khz /= 4 THEN
        count_100Khz <= count_100Khz + 1;
    ELSE
        count_100Khz <= "000";
        clock_100Khz_int <= NOT clock_100Khz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_100Khz_int'EVENT and clock_100Khz_int = '1';
    IF count_10Khz /= 4 THEN
        count_10Khz <= count_10Khz + 1;
    ELSE
        count_10Khz <= "000";
        clock_10Khz_int <= NOT clock_10Khz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_10Khz_int'EVENT and clock_10Khz_int = '1';
    IF count_1Khz /= 4 THEN
        count_1Khz <= count_1Khz + 1;
    ELSE
        count_1Khz <= "000";
        clock_1Khz_int <= NOT clock_1Khz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_1Khz_int'EVENT and clock_1Khz_int = '1';
    IF count_100hz /= 4 THEN
        count_100hz <= count_100hz + 1;
    ELSE
        count_100hz <= "000";
        clock_100hz_int <= NOT clock_100hz_int;

```

```

        END IF;
    END PROCESS;
    -- Divide by 10
    PROCESS
    BEGIN
        WAIT UNTIL clock_100hz_int'EVENT and clock_100hz_int = '1';
        IF count_10hz /= 4 THEN
            count_10hz <= count_10hz + 1;
        ELSE
            count_10hz <= "000";
            clock_10hz_int <= NOT clock_10hz_int;
        END IF;
    END PROCESS;
    -- Divide by 10
    PROCESS
    BEGIN
        WAIT UNTIL clock_10hz_int'EVENT and clock_10hz_int = '1';
        IF count_1hz /= 4 THEN
            count_1hz <= count_1hz + 1;
        ELSE
            count_1hz <= "000";
            clock_1hz_int <= NOT clock_1hz_int;
        END IF;
    END PROCESS;
END a;
```

## **VHDL DECODER\_FRUTAS**

```

library ieee;
use ieee.std_logic_1164.all;
```

```

Entity Dec_Fru is
    PORT( FRUTILLA: IN std_logic;
          CEREZA: IN std_logic;
          HIGO: IN std_logic;
          DURAZNO: IN std_logic;
          UVA: IN std_logic;
          Q: OUT std_logic_vector(8 downto 0));
end Dec_Fru;
```

```

Architecture Funcionamiento of Dec_Fru is
    signal temp: std_logic_vector(4 downto 0);
    Begin
        temp<=FRUTILLA&CEREZA&HIGO&DURAZNO&UVA;
```

```

Q <= "000010100" when temp = "00001" ELSE
    "000110010" when temp = "00010" ELSE
    "001000110" when temp = "00011" ELSE
    "000101000" when temp = "00100" ELSE
    "000111100" when temp = "00101" ELSE
    "001011010" when temp = "00110" ELSE
    "001101110" when temp = "00111" ELSE
    "000011110" when temp = "01000" ELSE
    "000110010" when temp = "01001" ELSE
    "001010000" when temp = "01010" ELSE
    "001100100" when temp = "01011" ELSE
    "001000110" when temp = "01100" ELSE
    "001011010" when temp = "01101" ELSE
    "001111000" when temp = "01110" ELSE
    "010001100" when temp = "01111" ELSE
    "000011001" when temp = "10000" ELSE
    "000101101" when temp = "10001" ELSE
    "001001011" when temp = "10010" ELSE
    "001011111" when temp = "10011" ELSE
    "001000001" when temp = "10100" ELSE
    "001010101" when temp = "10101" ELSE
    "001110011" when temp = "10110" ELSE
    "010000111" when temp = "10111" ELSE
    "000110111" when temp = "11000" ELSE
    "001001011" when temp = "11001" ELSE
    "001101001" when temp = "11010" ELSE
    "001111101" when temp = "11011" ELSE
    "001011111" when temp = "11100" ELSE
    "001110011" when temp = "11101" ELSE
    "010010001" when temp = "11110" ELSE
    "010100101" when temp = "11111" ELSE
    "000000000" ;

```

end Funcionamiento;

### *VHDL DECODER\_DULCES*

```

library ieee;
use ieee.std_logic_1164.all;

Entity Dec_Dul is
PORT( GRAJEAS: IN std_logic;
      OREO: IN std_logic;
      GOMITAS: IN std_logic;

```



```

        GRANOLA: IN
std_logic;
        Q: OUT
std_logic_vector(8 downto 0));
end Dec_Dul;

```

Architecture Funcionamiento of  
Dec\_Dul is  
signal temp: std\_logic\_vector(3  
downto 0);  
Begin

```

        temp<=GRAJEAS&OREO&G
OMITAS&GRANOLA;
        Q <= "000101000" when temp
= "0001" ELSE
            "000001010" when
temp = "0010" ELSE
            "000110010" when
temp = "0011" ELSE
            "000011001" when temp =
"0100" ELSE
            "001000001" when
temp = "0101" ELSE
            "000100011" when
temp = "0110" ELSE
            "001001011" when
temp = "0111" ELSE
            "000001111" when
temp = "1000" ELSE
            "000110111" when
temp = "1001" ELSE
            "000011001" when
temp = "1010" ELSE
            "001000001" when
temp = "1011" ELSE
            "000101000" when
temp = "1100" ELSE
            "001010000" when
temp = "1101" ELSE
            "000110010" when
temp = "1110" ELSE
            "001011010" when
temp = "1111" ELSE

```

```

        "000000000" ;
end Funcionamiento;

```

## VHDL DIRECCION

```

entity DIRECCION is
    port( Enable: in std_logic;
          clear: in std_logic;
          Clock: in std_logic;
          Reset: in std_logic;
          Output: out
          std_logic_vector(4 downto 0));
end DIRECCION;

architecture solve of DIRECCION
is
    signal temp: std_logic_vector(4
downto 0);
begin
    process(Clock,Reset,temp,clear)
    begin
        if Reset='0' then
            temp <= "000000";
            Output <= temp;
        elsif clear='1' then

            temp<="000000";
            Output <=
temp;
        elsif(Clock'event and
Clock='1') then
            if Enable='1' then
                if
temp="10100" then

                    temp<="000000";
                    Output <=
temp;

                    else
                        temp <=
temp + 1;
                        Output <=
temp;
                    end if;
                end if;
            end if;
        end if;
    end process;
end architecture solve;

```

```

        end if;
    end process;
end solve;

```

## VHDL RAM

```

LIBRARY IEEE;
USE
IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

entity RAM is
Generic (
    DATA_WIDTH          :
integer := 9;
    ADDRESS_WIDTH: integer :=
5
);
Port (
    Clock, Enable        : in
STD_LOGIC;
    Reset : in STD_LOGIC;
    DataIn      : in
STD_LOGIC_VECTOR
(DATA_WIDTH - 1 downto 0);
    Address      : in
STD_LOGIC_VECTOR
(ADDRESS_WIDTH - 1 downto
0);
    WriteEn      : in
STD_LOGIC;
    DataOut      : out
STD_LOGIC_VECTOR
(DATA_WIDTH - 1 downto 0);
    Loaded      : out STD_LOGIC
);
end RAM;

architecture Behavioral of RAM is
type Memory_Array is array (0 to
(2 ** ADDRESS_WIDTH) - 1) of
STD_LOGIC_VECTOR
(DATA_WIDTH - 1 downto 0);
signal Memory : Memory_Array;
begin

```

```

-- Read process
process (Clock)
begin
    if rising_edge(Clock) then
        if Reset = '1' then
            -- Clear DataOut
on Reset
                DataOut <=
(others => '0');
            elsif Enable = '0' then
                DataOut <=
Memory(to_integer(unsigned(Add
ress)));
            end if;
        end if;
    end process;

-- Write process
process (Clock)
begin
    if rising_edge(Clock) then
        loaded <='0';
        if Reset = '0' then
            -- Clear Memory
on Reset
                for i in
Memory'Range loop
                    Memory(i)
<= (others => '0');
                end loop;
            elsif WriteEn = '1' then
                for i in
Memory'Range loop

                    if(Memory(i)="000000000")
then

                        Memory(i) <= DataIn;

                        loaded <='1';exit when TRUE;
                    end if;
                end loop;
            end if;
        end if;
    end process;

```

end Behavioral;

## **VHDL MUX**

```
library IEEE;
use
IEEE.STD_LOGIC_1164.ALL;

entity mux_mostrar is

port (
    datoA : in std_logic_vector(8
downto 0);
    datoB : in std_logic_vector(8
downto 0);
    sel: in std_logic;
    SalidaA : out std_logic_vector(8
downto 0)

);
end mux_mostrar;

architecture sol of mux_mostrar is
begin

process(datoA,datoB)
begin
case sel is
when '0' =>
SalidaA <= datoA;
when others =>
SalidaA <= datoB;
end case;

end process;
end sol;
```

## ***VDHL BINtoBCD***

## ***VHDL BINtoBCD***

```
library IEEE;
use
IEEE.STD_LOGIC_1164.ALL;
```

```
use
IEEE.STD_LOGIC_ARITH.ALL;
use
IEEE.STD_LOGIC_UNSIGNED.A
LL;
```

```
entity Conv_Bin_BCD is
  Port ( Bin : in
STD_LOGIC_VECTOR (7
downto 0);
        Cen : out
STD_LOGIC_VECTOR (3
downto 0);
        Dec : out
STD_LOGIC_VECTOR (3
downto 0);
        Uni : out
STD_LOGIC_VECTOR (3
downto 0));
end Conv_Bin_BCD;
```

```
architecture Behavioral of
Conv_Bin_BCD is
```

```
begin
```

```
  Process(Bin)
  variable Z:
STD_LOGIC_VECTOR (19
downto 0);
  begin
```

```
    for i in 0 to 19 loop
      Z(i) := '0';
    end loop;
```

```
    Z(10 downto 3) := Bin;
```

```
    for i in 0 to 4 loop
```

```
      if Z(11 downto 8) > 4 then
        Z(11 downto 8) := Z(11 downto
8) + 3;
      end if;
```

```
if Z(15 downto 12) > 4 then
  Z(15 downto 12) := Z(15 downto
12) + 3;
  end if;
```

```
Z(17 downto 1) := Z(16 downto
0);
  end loop;
```

```
Cen <= Z(19 downto 16);
Dec <= Z(15 downto 12);
Uni <= Z(11 downto 8);
end Process;
end Behavioral
```

## VHDL SUMADOR

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
```

```
ENTITY SUMADOR IS
  PORT (    a : IN
std_logic_vector(8 DOWNT0 0);
          b : IN
std_logic_vector(8 DOWNT0 0);
          c : IN
std_logic_vector(8 DOWNT0 0);
          d : IN
std_logic_vector(8 DOWNT0 0);
          salida : OUT
std_logic_vector(8 DOWNT0 0));
  END SUMADOR;
```

```
  ARCHITECTURE SOLUTION
OF SUMADOR IS
  BEGIN
```

```
    PROCESS (a,b,c,d) IS
    BEGIN
      salida <=
std_logic_vector(UNSIGNED(a) +
UNSIGNED(b) + UNSIGNED(c) +
UNSIGNED(d));
```

```

END PROCESS;
END SOLUTION;

```

## VHDL 7SEGMENTOS

```

library IEEE;
use
IEEE.STD_LOGIC_1164.ALL;
use
IEEE.STD_LOGIC_ARITH.ALL;
use
IEEE.STD_LOGIC_UNSIGNED.A
LL;

entity
decoder_de_bcd_a_7segmentos
is
    port( BCD: in std_logic_vector(3
downto 0);
          SEG7: out
std_logic_vector(1 to 7));
end
decoder_de_bcd_a_7segmentos;

architecture solve of
decoder_de_bcd_a_7segmentos
is
    begin
    SEG7 <= "1111110"   when BCD
= "0000" ELSE
                "0110000"
    when BCD = "0001" ELSE
                "1101101"
    when BCD = "0010" ELSE
                "1111001"
    when BCD = "0011" ELSE
                "0110011"
    when BCD = "0100" ELSE
                "1011011"
    when BCD = "0101" ELSE
                "1011111"
    when BCD = "0110" ELSE
                "1110000"
    when BCD = "0111" ELSE

```



```
        "1111111"
when BCD = "1000" ELSE
        "1111011"
when BCD = "1001" ELSE
        "0000000" ;
end solve;

;
```

## **VHDL 20 CONSTANTE**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use
IEEE.STD_LOGIC_UNSIGNED.ALL;

entity VEINTE_CONSTANTE is
    port(      SALIDA: out
std_logic_vector(4 DOWNT0 0));

end VEINTE_CONSTANTE;

architecture solve of
VEINTE_CONSTANTE is
begin
    SALIDA <="10100";
end solve;
```

## **Vínculo de GitHub**

<https://github.com/jeba1797/Proyecto-Helader-a>

## **Link de Youtube**

<https://www.youtube.com/watch?v=vDXJo8Lljbs&t=252s>