REPORTE DE PRACTICA DEL CPU DE 4 INSTRUCCIONES

el programa aqui descrito es un cpu capaz de realizar las siguientes operaciones:

1. cargar datos en cualquier registro.
2. mover un dato de un registro a otro.
3. Sumar dos valores.
4. Restar dos valores.

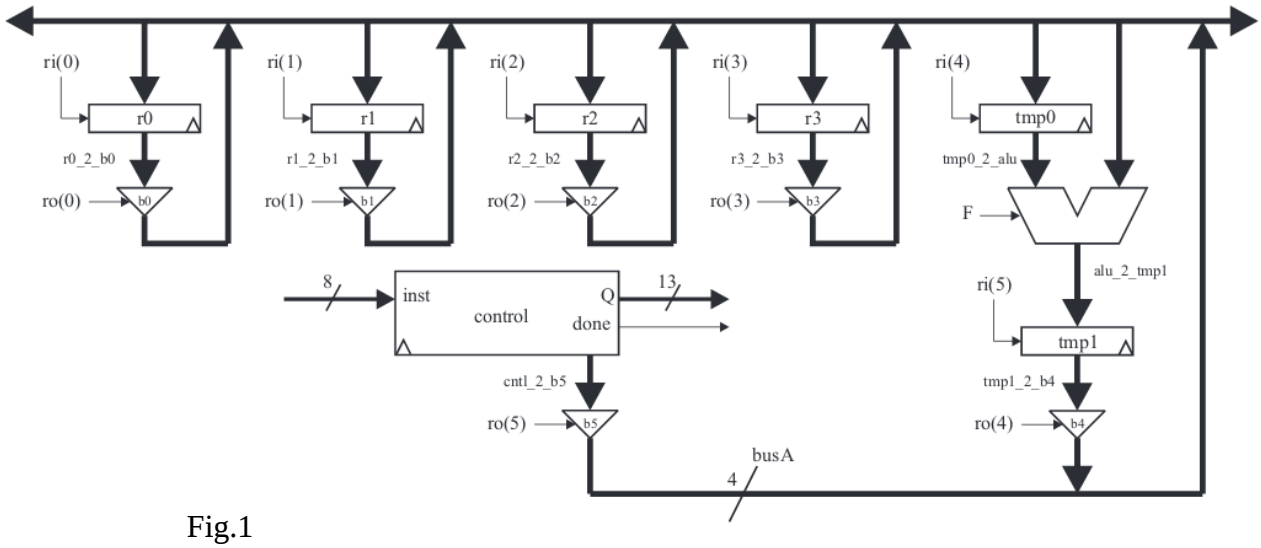


Fig.1

En la fig.1 se muestra el diagrama estructural de nuestro cpu de 4 instrucciones, el cual esta conformado por:

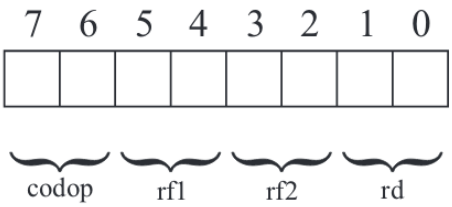
1. unidad de control.
2. Buffer de 3 estados.
3. Registros de carga y almacenamiento.
4. Registros temporales.
5. Alu (sumadora y restadora).

De este modo comenzaremos a descibrir la funcionalidad de los componentes y su alambrado general.

1. Unidad de control

1. recibe una señal de 8 bits seran las instrucciones del cpu.
2. Una salida Q de 13 bits, el habilitador de buffers y registros.
3. Una salida done de tipo bit que avisa cada que una instruccion fue llevada a cabo con exito.
4. La señal de reloj, por ser un circuito secuencial.

- Cuando la instrucción tiene tres operandos



- Cuando la instrucción tiene dos operandos

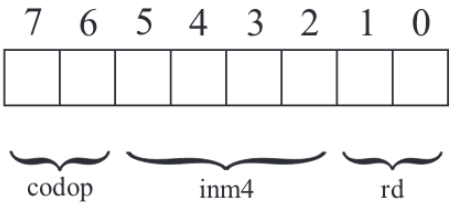


Fig.2

codop	Instrucción
00	load
01	move
10	add
11	sub

la fig.2 muestra la distribucion de los bits en un vector, que es la entrada inst de la unidad de control, los bits 7 y 6 son el codigo de operacion(codop), la instruccion relacionada con el codigo correspondiente se encuentra en la tabla de lado izquierdo.

Los bits 5 y 6 es el numero de registro que vamos a operar(rf1) y el 3 y 4 es el otro operador(rf2), en el caso de la suma el orden no tiene mayor relevancia, en el caso de la resta rf1 es el minuendo y rf2 el sustraendo.
Los bits 1 y 2, son el registro direccion, es decir, sera donde alojaremos el resultado de cada instruccion.
En el caso de la instruccion load, en los bits 2 a 5, indicaremos el valor a cargar y para load el numero de registro que queremos desplazar.

Descripcion en VHDL de la unidad de control.

```
--Edgar Humberto Perez Martinez
--Ing Electronica
--descripcion de la unidad de control para el cpu de 4 instrucciones
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity control is
   port(

          inst:   in std_logic_vector(7 downto 0);
           clk:   in std_logic;
             q:   out std_logic_vector(12 downto 0);
          done:   out std_logic

       );
end entity control;
```

Arquitectura.

Para poder describir el comporamiento de la unidad de control, lo mejor es trabajar con una MEF(maquina de estados finitos), para llevar a cabo la MEF haremos uso de un diagrama de estados para no perder la secuencia de las instrucciones, para ello nos apoyaremos en el siguiente diagrama.

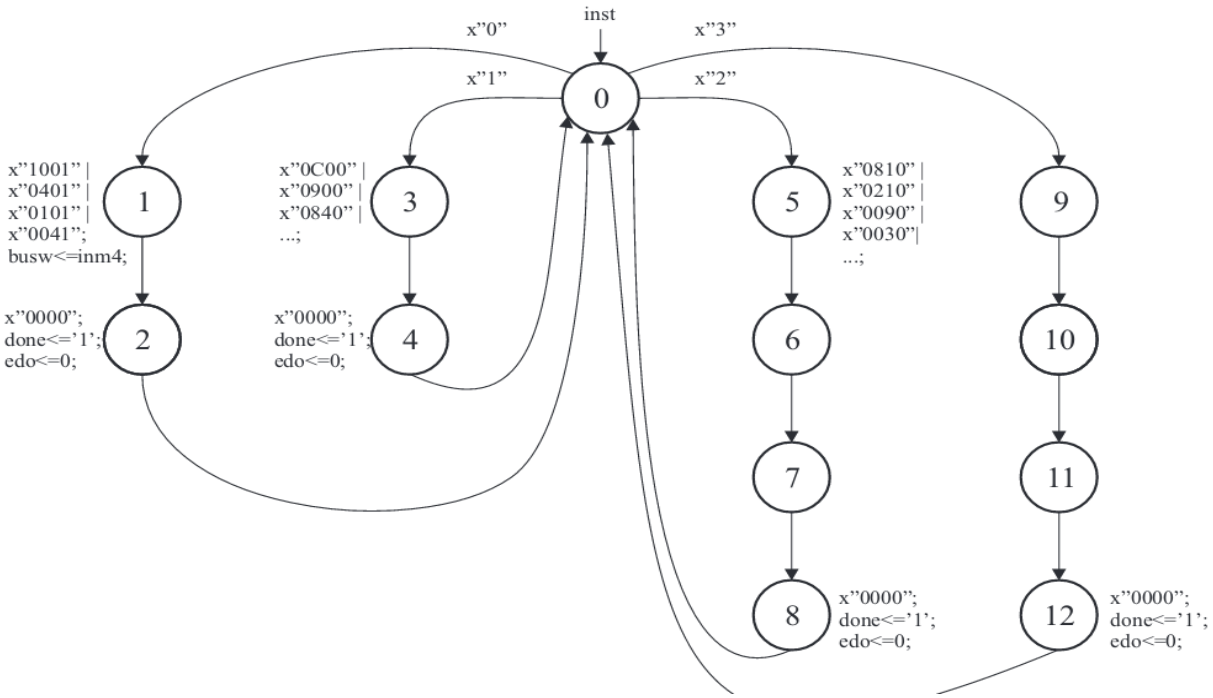


Fig.4

en primer instancia declaramos la arquitectura, las señales que ocuparemos, y asignamos los valores de la instruccion a su debido codigo, el cual se acordo en la figura 2.

```
architecture beh of control is

signal inm4,ep,es:std_logic_vector(3 downto 0):="0000";
signal codop,rd,rf1,rf2:std_logic_vector(1 downto 0):="00";
begin

codop<=inst(7 downto 6);--siguiendo el acuerdo de la fig.2
rd<=inst(1 downto 0);
rf1<=inst(5 downto 4);
rf2<=inst(3 downto 2);
inm4<=inst(5 downto 2);
```

debido a que es una MEF tarbajaremos con estado presente y estado siguiente.

```
secuencial:process(clk)--proceso del circuito secuencial
        begin
        if clk'event and clk = '1' then
                ep<=es;
        end if;
end process secuencial;
```

Aqui describimos los flip flops como un proceso.

```
combinacional:process(ep)--proceso del circuito combinacional
        begin

if ep = "0000" then --estado 0
        case codop is
        when "00"=>
        es<="0001";
        when "01"=>
        es<="0011";
        when "10"=>
        es<="0101";
        when "11"=>
        es<="1001";
        when others=>
        es<="0000";
        end case;
        done<='0';
```

Comenzamos a describir el estado 0:

1. si el codigo de operacion es 00 significa carga, asi que vamos al estado 1.
2. si el codigo de operacion es 01 significa mover, asi que vamos al estado 3.
3. si el codigo de operacion es 10 significa suma, asi que vamos al estado 5.
4. si el codigo de operacion es 11 significa resta, asi que vamos al estado 9.
5. para cualquier otra combinacion nos quedamos en el estado 0.

```
elsif ep = "0001" then -- estado 1
        case rd is
        when "00"=>
        q<="1000000000001";
        busw<=inm4;
        when "01"=>
        q<="0010000000001";
        busw<=inm4;
        when "10"=>
        q<="0000100000001";
        busw<=inm4;
        when "11"=>
        q<="0000001000001";
        busw<=inm4;
        when others=>
        q<="0000000000000";
        busw<=inm4;
        end case;
```

el estado 1 corresponde a la secuencia de carga, asi que en el siguiente ciclo de reloj, seleccionamos el registro destino de carga:

1. si rd es 00 habilitamos la lectura del r0 y que control deje pasar el conjunto de 4 bits (inm4) a traves del bus 5 con permiso de escritura b5 .
2. si rd es 01 habilitamos la lectura del r1 y que control deje pasar el conjunto de 4 bits (inm4) a traves del bus 5 con permiso de escritura b5 .
3. si rd es 10 habilitamos la lectura del r2 y que control deje pasar el conjunto de 4 bits (inm4) a traves del bus 5 con permiso de escritura b5 .
4. si rd es 11 habilitamos la lectura del r3 y que control deje pasar el conjunto de 4 bits (inm4) a traves del bus 5 con permiso de escritura b5
5. para cualquier otra combinacion, deshabilitamos todos los registros.

```
elsif ep = "0010" then -- estado 2
        q<="0000000000000";
        done<='1';
        es<="0000";
```

En el estado 2 finalizamos la instruccion de carga y regresamos al estado 00 para hacer una nueva seleccion, avisamos que la instruccion se realizo con done=’1’ y deshabilitamos todos los registros.

```
elsif ep = "0011" then -- estado 3
        if rf1 = "00" then --r0 a mover
                if rd = "01" then--posible r1
                        q<="0110000000000";
                elsif rd = "10" then--posible r2
                        q<="0100100000000";
                elsif rd = "11" then--posible r3
                        q<="0100001000000";
                else
                        q<="0000000000000";
                end if;
        elsif rf1 = "01" then --r1 a mover
                if rd = "00" then--posible a r0
                        q<="1001000000000";
                elsif rd = "10" then--posible a r2
                        q<="0001100000000";
                elsif rd = "11" then--posible a r3
                        q<="0001001000000";
                else
                        q<="0000000000000";
                end if;
        else
                        q<="0000000000000";
        end if;
        done<='0';

                end if;
        elsif rf1 = "10" then --r2 a mover
                if rd = "00" then--posible a r0
                        q<="1000010000000";
                elsif rd = "10" then--posible a r1
                        q<="0010010000000";
                elsif rd = "11" then--posible a r3
                        q<="0000011000000";
                else
                        q<="0000000000000";
                end if;
        elsif rf1 = "11" then --r3 a mover
                if rd = "00" then--posible a r0
                        q<="1000000100000";
                elsif rd = "10" then--posible a r1
                        q<="0010000100000";
                elsif rd = "11" then--posible a r2
                        q<="0000100100000";
                else
                        q<="0000000000000";
                end if;
        else
                        q<="0000000000000";
        end if;
        done<='0';
```

En el estado 3 podemos mover un valor a otro registro, claro esta que sera diferente del actual, en las anteriores figuras se muestran todas las posibilidades de movimiento de valor de un registro.

```
elsif ep = "0100" then -- estado 4
        q<="0000000000000";
        done<='1';
        es<="0000";
```

El estado 4 finaliza la instruccion de move, regresamos al estado 00 para hacer una nueva seleccion, avisamos que la instruccion se realizo con done='1' y deshabilitamos todos los registros.

```
elsif ep = "0101" then -- estado 5
        if rf1 = "00" then --r a sumar
                        q<="0100000010000";
        elsif rf1 = "01" then
                        q<="0001000010000";
        elsif rf1 = "10" then
                        q<="0000010010000";
        elsif rf1 = "11" then
                        q<="0000000110000";
        else
                        q<="0000000000000";
        end if;
```

Con el estado 5 indicamos el registro a sumar dejamos que el registro que esta seleccionado con rf1 tenga permiso de escritura y el registro temporal 0 lo tenga de lectura.

```
elsif ep = "0110" then -- estado 6
        if rf1 = "00" then --r0 a sumar
                if rf2 = "01" then--posible con r1
                        q<="0001000001000";
                elsif rf2 = "10" then--posible con r2
                        q<="0000010001000";
                elsif rf2 = "11" then--posible con r3
                        q<="0000000101000";
                else
                        q<="0000000000000";
                end if;

        elsif rf1 = "01" then --r1 a sumar
                if rf2 = "00" then--posible con r|
                        q<="0100000001000";
                elsif rf2 = "10" then--posible con r2
                        q<="0000010001000";
                elsif rf2 = "11" then--posible con r3
                        q<="0000000101000";
                else
                        q<="0000000000000";
                end if;
        elsif rf1 = "10" then --r2 a sumar
                if rf2 = "00" then--posible con r0
                        q<="0100000001000";
                elsif rf2 = "01" then--posible con r1
                        q<="0001000001000";
                elsif rf2 = "11" then--posible con r3
                        q<="0000000101000";
                else
                        q<="0000000000000";
                end if;

        elsif rf1 = "11" then --r3 a sumar
                if rf2 = "00" then--posible con r0
                        q<="0100000001000";
                elsif rf2 = "01" then--posible con r1
                        q<="0001010001000";
                elsif rf2 = "10" then--posible con r2
                        q<="0000010001000";
                else
                        q<="0000000000000";
                end if;
        end if;
```

En el estado 6 una vez seleccionado uno de los sumandos, instanciamos todas las posibilidades con las que podemos sumar el primer registro, claro esta, sin sumarse a si mismo.

```
elsif ep = "0111" then -- estado 7
        if rf1 = "00" then --r0
                if rf2 = "01" then--r0 +r1
                        if rd = "10" then--guarda en r2
                        q<="0000100000100";
                        elsif rd = "11" then--guarda en r3
                        q<="0000001000100";
                        else
                        q<="0000000000000";
                        end if;
                elsif rf2 = "10" then--r0 + r2
                        if rd = "01" then--guarda en r1
                        q<="0010000000100";
                        elsif rd = "11" then--guarda en r3
                        q<="0000001000100";
                        else
                        q<="0000000000000";
                        end if;
                elsif rf2 = "11" then--r0 + r3
                        if rd = "01" then--guarda en r1
                        q<="0010000000100";
                        elsif rd = "10" then--guarda en r2
                        q<="0000100000100";
                        else
                        q<="0000000000000";
                        end if;
                else
                        q<="0000000000000";
                end if;
```

En el estado 7 listamos las posibillidades de suma con el registro 0 y lo guardamos en cualquier registro que no este ocupado previamente, se repite la misma lista en el caso del registro 1,2 y 3.

```
elsif rf1 = "01" then --r1 a sumar
        if rf2 = "00" then--r1 + r0
                if rd = "10" then--guarda en r2
                q<="0000100000100";
                elsif rd = "11" then--guarda en r3
                q<="0000001000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "10" then--r1 + r2
                if rd = "00" then--guarda en r0
                q<="1000000000100";
                elsif rd = "11" then--guarda en r3
                q<="0000001000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "11" then--r1 + r3
                if rd = "00" then--guarda em r0
                q<="1000000000100";
                elsif rd = "10" then--guarda en r2
                q<="0000100000100";
                else
                q<="0000000000000";
                end if;
        else
                q<="0000000000000";
        end if;
```

```
elsif rf1 = "10" then --r2 a sumar
        if rf2 = "01" then--r2 + r1
                if rd = "00" then--guarda en r0
                q<="1000000000100";
                elsif rd = "11" then--guarda en r3
                q<="0000001000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "00" then--r2 + r0
                if rd = "01" then--guarda en r1
                q<="0010000000100";
                elsif rd = "11" then--guarda en r3
                q<="0000001000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "11" then--r2 + r3
                if rd = "00" then--guarda en r0
                q<="1000000000100";
                elsif rd = "01" then--guarda en r1
elsif rf1 = "11" then --r3 a sumar
        if rf2 = "00" then--r3 + r0
                if rd = "01" then--guarda en r1
                q<="0010000000100";
                elsif rd = "10" then--guarda en r2
                q<="0000100000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "10" then--r3 + r2
                if rd = "00" then--guarda en r0
                q<="1000000000100";
                elsif rd = "01" then--guarda en r1
                q<="0010000000100";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "01" then--r3 + r1
                if rd = "00" then--guarda en r0
                q<="1000000000100";
                elsif rd = "10" then--guarda en r2
                q<="0000100000100";
                else
                q<="0000000000000";
                end if;
        else
                q<="0000000000000";
        end if;
end if;--elsif de rf1
done<='0';
```

con el anterior codigo mostramos todas las posibilidades de suma sin que se repitan los registros entre si.

```
elsif ep = "1000" then -- estado 8
        q<="0000000000000";
        done<='1';
        es<="0000";
```

En el estado 8 finalizamos la suma de los registros,regresamos al estado 00 para hacer una nueva seleccion, avisamos que la instruccion se realizo con done=’1’ y deshabilitamos todos los registros.

Para el caso de los estados 9 al 12, es similar, a diferencia que habilitaremos h del alu, ya que en esta ocasion sera una resta, de cualquier modo se mostraran los respectivos codigos.

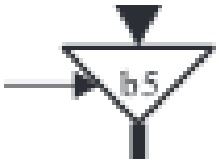
```
elsif ep = "1001" then -- estado 9
        if rf1 = "00" then --r a restar
                        q<="0100000010010";
        elsif rf1 = "01" then
                        q<="0001000010010";
        elsif rf1 = "10" then
                        q<="0000010010010";
        elsif rf1 = "11" then
                        q<="0000000110010";
        else
                        q<="0000000000010";
        end if;
 elsif ep = "1010" then -- estado 10
         if rf1 = "00" then --r a restar
                if rf2 = "01" then
                         q<="0001000001010";
                elsif rf2 = "10" then
                         q<="0000010001010";
                elsif rf2 = "11" then
                         q<="0000000101010";
                else
                         q<="0000000000010";
                end if;

         elsif rf1 = "01" then --minuendo
                if rf2 = "00" then--sustraendo
                         q<="0100000001010";
                elsif rf2 = "10" then
                         q<="0000010001010";
                elsif rf2 = "11" then
                         q<="0000000101010";
                else
                         q<="0000000000010";
                end if;
         elsif rf1 = "10" then
                if rf2 = "00" then
                         q<="0100000001010";
                elsif rf2 = "01" then
                         q<="0001000001010";
                elsif rf2 = "11" then
                         q<="0000000101010";
                else
                         q<="0000000000010";
                end if;

          elsif rf1 = "11" then
                  if rf2 = "00" then
                           q<="0100000001010";
                  elsif rf2 = "01" then
                           q<="0001010001010";
                  elsif rf2 = "10" then
                           q<="0000010001010";
                  else
                           q<="0000000000010";
                  end if;
          end if;
```

```
elsif ep = "1011" then -- estado 11
        if rf1 = "00" then
                if rf2 = "01" then
                        if rd = "10" then
                        q<="0000100000110";
                        elsif rd = "11" then
                        q<="0000001000110";
                        else
                        q<="0000000000000";
                        end if;
                elsif rf2 = "10" then
                        if rd = "01" then
                        q<="0010000000110";
                        elsif rd = "11" then
                        q<="0000001000110";
                        else
                        q<="0000000000000";
                        end if;
                elsif rf2 = "11" then
                        if rd = "01" then
                        q<="0010000000110";
                        elsif rd = "10" then
                        q<="0000100000110";
                        else
                        q<="0000000000000";
                        end if;
                else
                        q<="0000000000000";
                end if;

          elsif rf1 = "01" then
                  if rf2 = "00" then
                          if rd = "10" then
                          q<="0000100000110";
                          elsif rd = "11" then
                          q<="0000001000110";
                          else
                          q<="0000000000000";
                          end if;
                  elsif rf2 = "10" then
                          if rd = "00" then
                          q<="1000000000110";
                          elsif rd = "11" then
                          q<="0000001000110";
                          else
                          q<="0000000000000";
                          end if;
                  elsif rf2 = "11" then
                          if rd = "00" then
                          q<="1000000000110";
                          elsif rd = "10" then
                          q<="0000100000110";
                          else
                          q<="0000000000000";
                          end if;
                  else
                          q<="0000000000000";
                  end if;
```

```
elsif rf1 = "10" then
        if rf2 = "01" then
                if rd = "00" then
                q<="1000000000110";
                elsif rd = "11" then
                q<="0000001000110";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "00" then
                if rd = "01" then
                q<="0010000000110";
                elsif rd = "11" then
                q<="0000001000110";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "11" then
                if rd = "00" then
                q<="1000000000110";
                elsif rd = "01" then
                q<="0010000000110";
                else
                q<="0000000000000";
                end if;
        else
                q<="0000000000000";
        end if;

elsif rf1 = "11" then
        if rf2 = "00" then
                if rd = "01" then
                q<="0010000000110";
                elsif rd = "10" then
                q<="0000100000110";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "10" then
                if rd = "00" then
                q<="1000000000110";
                elsif rd = "01" then
                q<="0010000000110";
                else
                q<="0000000000000";
                end if;
        elsif rf2 = "01" then
                if rd = "00" then
                q<="1000000000110";
                elsif rd = "10" then
                q<="0000100000110";
                else
                q<="0000000000000";
                end if;
        else
                q<="0000000000000";
        end if;
end if;--elsif de rf1
```

```
        elsif ep = "1100" then -- estado 12
                q<="0000000000000";
                done<='1';
                es<="0000";


        end if;
        end process combinacional;
d architecture beh;
```

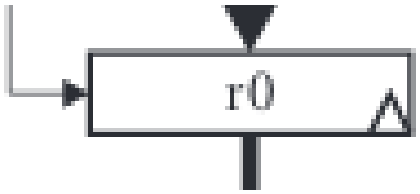
2.Descripcion del buffer de 3 estados.




```
--Edgar Humberto Perez Martinez
--Ing Electronica
--Descripcion de un buffer de 3 estados
library ieee;
use ieee.std_logic_1164.all;

entity buffer3e is
   port(

          ent:  in std_logic_vector(3 downto 0);
          h:    in std_logic;
          sal:  out std_logic_vector(3 downto 0)
       );
end entity buffer3e;

architecture beh of buffer3e is
begin
        --proceso
        process(ent,h)
                begin

                if h = '1' then
                        sal<=ent;
                else
                        sal<=(others=>'Z');
                end if;

        end process;
end architecture beh;
```

Como podemos observar el buffer tiene una entrada de 4 bits, que en este caso es el bus de transmision de datos, el habilitador h, y la salida (sal), el funcinamiento del buffer es sencillo; si h es 1 permite el paso de los datos, si es 0 desconecta el bus.

3.Registros de carga y almacenamiento.




```
--Edgar Humberto Perez Martinez
--Ing Electronica
--Descripcion de un registro de carga y almacenamiento
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity registro is
   port(
          ent:in   std_logic_vector(3 downto 0);
            h:in   std_logic;
          clk:in   std_logic;
          sal:out  std_logic_vector(3 downto 0)
       );
end entity registro;

architecture beh of registro is

signal interna:std_logic_vector(3 downto 0);
        begin
                sal<=interna;
        process(clk)

                begin
                        if clk'event and clk='1' then
                                if h ='1' then
                                interna <= ent;
                        end if;
                end if;
        end process;

end architecture beh;
```

4.ALU(Unidad Aritmetica-Logica)

```vhdl
--Edgar Humberto Perez Martinez
--Ing Electronica
--Descripcion de un alu de carga y almacenamiento
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity alu is
   port(
          x:in   std_logic_vector(3 downto 0);
          y:in   std_logic_vector(3 downto 0);
          h:in   std_logic;
          f:out  std_logic_vector(3 downto 0)
       );
end entity alu;

architecture beh of alu is
        begin
        process(x,y,h)
                begin
                        case h is
                                when '0'=>
                                f<=std_logic_vector(signed(x)+signed(y));
                                when '1'=>
                                f<=std_logic_vector(signed(x)-signed(y));
                                when others=>
                                f<="ZZZZ";
                        end case;
        end process;

end architecture beh;
```

Entidad del CPU de 4 instrucciones.

```vhdl
--Edgar Humberto Perez Martinez
--Ing Electronica
--descripcion de la unidad de cpu4inst para el cpu de 4 instrucciones
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cpu4inst is
   port(

          inst:   in std_logic_vector(7 downto 0);
           clk:   in std_logic;
          done:   out std_logic

       );
end entity cpu4inst;
```

Señales de acuerdo al diagrama de la fig.1

```
architecture beh of cpu4inst is

signal a:std_logic_vector(12 downto 0):="0000000000000";
signal busx:std_logic_vector(3 downto 0):="0000";
signal cntl_2_b5:std_logic_vector(3 downto 0):="0000";
signal tmp1_2_b4:std_logic_vector(3 downto 0):="0000";
signal alu_2_tmp1:std_logic_vector(3 downto 0):="0000";
signal tmp0_2_alu:std_logic_vector(3 downto 0):="0000";
signal r3_2_b3:std_logic_vector(3 downto 0):="0000";
signal r2_2_b2:std_logic_vector(3 downto 0):="0000";
signal r1_2_b1:std_logic_vector(3 downto 0):="0000";
signal r0_2_b0:std_logic_vector(3 downto 0):="0000";
```

a continuacion instanciamos los componentes.

```
component buffer3e is
   port(
          ent:  in std_logic_vector(3 downto 0);
          h:    in std_logic;
          sal:  out std_logic_vector(3 downto 0)
       );
end component buffer3e;

component registro is
   port(
          ent:  in std_logic_vector(3 downto 0);
          h:    in std_logic;
          clk:  in std_logic;
          sal:  out std_logic_vector(3 downto 0)
       );
end component registro;

component control is
   port(
          inst:   in std_logic_vector(7 downto 0);
           clk:   in std_logic;
             q:   out std_logic_vector(12 downto 0);
          busw:   out std_logic_vector(3 downto 0);
          done:   out std_logic
       );
end component control;
component alu is
   port(
          x:  in std_logic_vector(3 downto 0);
          y:  in std_logic_vector(3 downto 0);
          h:    in std_logic;
          f:  out std_logic_vector(3 downto 0)
       );
end component alu;
```

Aqui comenzaremos a alambrar los componentes

```
begin
        u0:control
                port map(
                        inst=>inst,
                        q=>a,
                        clk=>clk,
                        busw=>cntl_2_b5,
                        done=>done
                );
        u1:alu
                port map(
                        x=>tmp0_2_alu,
                        y=>busx,
                        h=>a(1),
                        f=>alu_2_tmp1
                );
        r0:registro
                port map(
                        ent=>busx,
                        clk=>clk,
                        h=>a(12),
                        sal=>r0_2_b0
                );
        r1:registro
                port map(
                        ent=>busx,
                        clk=>clk,
                        h=>a(10),
                        sal=>r1_2_b1
                );


        r2:registro
                port map(
                        ent=>busx,
                        clk=>clk,
                        h=>a(8),
                        sal=>r2_2_b2
                );
        r3:registro
                port map(
                        ent=>busx,
                        clk=>clk,
                        h=>a(6),
                        sal=>r3_2_b3
                );
        t0:registro
                port map(
                        ent=>busx,
                        clk=>clk,
                        h=>a(4),
                        sal=>tmp0_2_alu
                );
        t1:registro
                port map(
                        ent=>alu_2_tmp1,
                        clk=>clk,
                        h=>a(3),
                        sal=>tmp1_2_b4
                );
```

```
b0:buffer3e
        port map(
                ent=>r0_2_b0,
                h=>a(11),
                sal=>busx
        );
b1:buffer3e
        port map(
                ent=>r1_2_b1,
                h=>a(9),
                sal=>busx
        );
b2:buffer3e
        port map(
                ent=>r2_2_b2,
                h=>a(7),
                sal=>busx
        );
b3:buffer3e
        port map(
                ent=>r3_2_b3,
                h=>a(5),
                sal=>busx
        );
b4:buffer3e
        port map(
                ent=>tmp1_2_b4,
                h=>a(2),
                sal=>busx
        );
 b5:buffer3e
          port map(
                  ent=>cntl_2_b5,
                  h=>a(0),
                  sal=>busx
          );
```

```
--Edgar Humberto Perez Martinez
--Ing Electronica
--Descripcion de estimulos para sumador de 2 bits
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_cpu4inst is
end entity tb_cpu4inst;

architecture beh of tb_cpu4inst is
    component cpu4inst
    port(
          inst:   in std_logic_vector(7 downto 0);
           clk:   in std_logic;
          done:   out std_logic
        );
    end component cpu4inst;

    signal inst:std_logic_vector(7 downto 0):="00000000" ;
    signal clk:std_logic:='0' ;
    signal done:std_logic ;

begin
        u0:cpu4inst
   port map(
          inst=> inst,
          clk => clk,
          done => done
           );
```

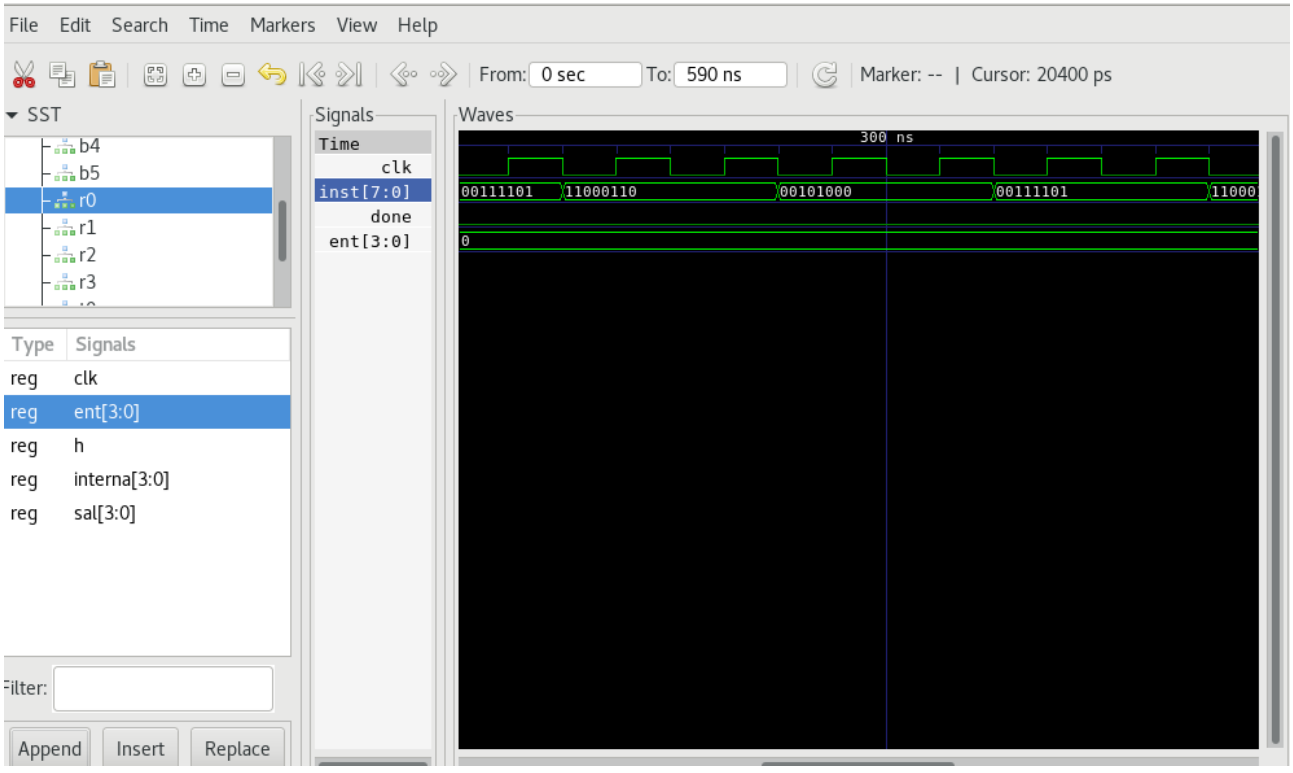
Ambiente de pruebas (Test Bench).

```
 process
   begin
      wait for 10 ns;
      clk <= not clk;
   end process;

   process
   begin
      wait for 40 ns;
      inst <= "00101000";
      wait for 40 ns;
      inst <= "00111101";
      wait for 40 ns;
      inst <= "11000110";
   end process;

end architecture beh;
```

Makefile.

```
#Edgar Humberto Perez Martinez
#Archivo que describe la compilacion con GHDL
#y la visualización con GTKWave, en Linux

all:tb_cpu4inst

tb_cpu4inst:cpu4inst.o buffer3e.o registro.o control.o  alu.o tb_cpu4inst.o
	ghdl -e tb_cpu4inst
	./tb_cpu4inst --stop-time=600ns --vcd=tb_cpu4inst.vcd
	gtkwave tb_cpu4inst.vcd ondas.gtkw

cpu4inst.o:cpu4inst.vhdl
	ghdl -a cpu4inst.vhdl

alu.o:alu.vhdl
	ghdl -a alu.vhdl

registro.o:registro.vhdl
	ghdl -a registro.vhdl

buffer3e.o:buffer3e.vhdl
	ghdl -a buffer3e.vhdl

control.o:control.vhdl
	ghdl -a control.vhdl

tb_cpu4inst.o:tb_cpu4inst.vhdl
	ghdl -a tb_cpu4inst.vhdl

clean:
	rm *.o *.vcd tb_cpu4inst *.cf
```

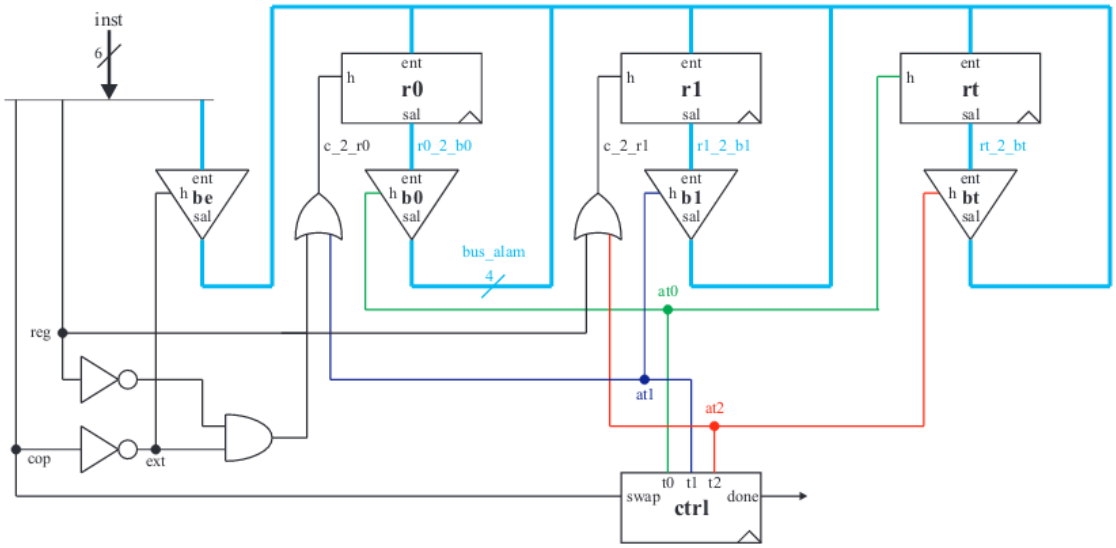
Gtkwave.



La conclusion al observar el simulador es que no hubo problemas de compilacion, sin embargo el registro 0 no recibe ningun valor en ningun flanco, de modo que tenemos que revisar el motivo que impde el envio de las señales.

REPORTE DE PRACTICA DEL CPU DE INTERCAMBIO.

Para la dscripcion del cpu de intercambio, haremos uso de algunos componentes ya descritos anteriormente en este archivo, sera el caso de los buffer de 3 estados. Registros de carga y almacenamiento, y una unidad de control.



Aqui se describe la forma estructural de nuestro circuito, tambien haremos uso de 2 compuertas NOT, 2 compuertas OR y una compuerta AND.
Para describir el funcionamiento del circuito, nos apoyaremos con los siguientes diagramas.

Formatos de instrucción

- Instrucción con dato inmediato

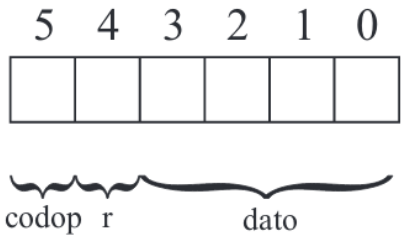


Fig.1a

al igual que con el cpu de 4 instrucciones, necesitamos llegar a un formato de instruccion, nos apoyamos en la fig 1a el bit 5 sera el codigo de operacion; carga o intercambio. El bit 4 sera el registro destino en el caso de carga, y los bits de 0 a 3, hacen referencia al valor a cargar en el registro de destino.

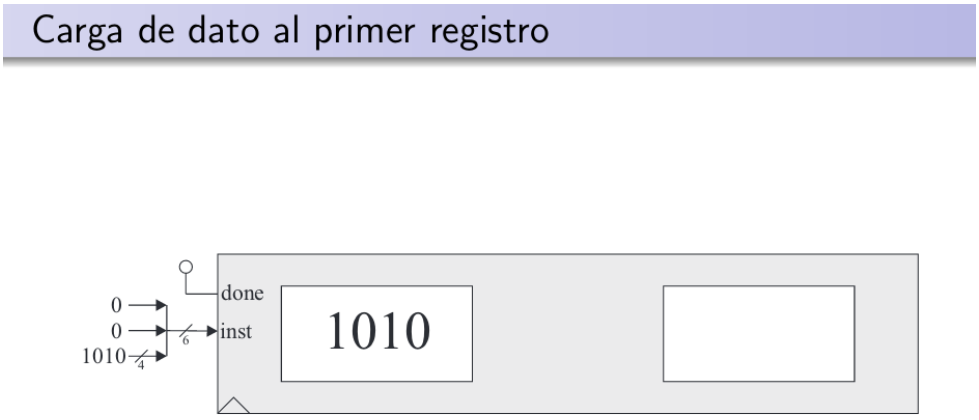
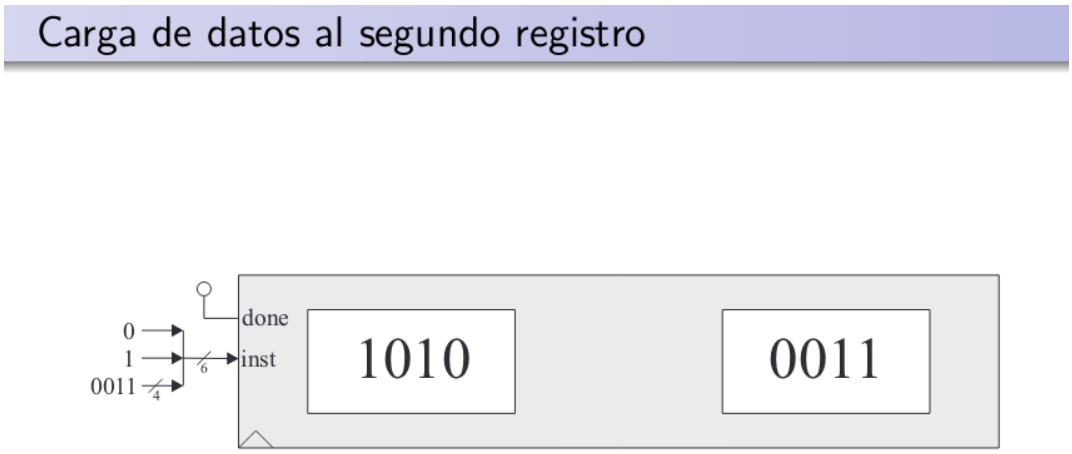
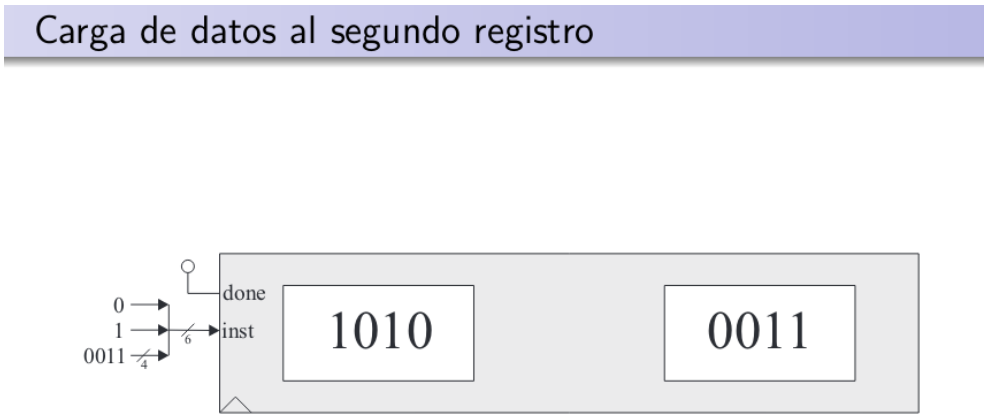


Fig 2a

con la fig.1a nos damos una idea del funcionamiento del cpu, en primer lugar, seleccionamos el codigo de operacion como 0, ya que sera carga de datos, en 0 r1 ya que alojaremos el dato en el registro 0, y los siguientes cuatro bits indican que cargaremos el vector 1010.





mediante los anteriores diagramas nos podemos dar una simple idea de la operacion del cpu de intercambio.

Ahora pues aqui describo la entidad en vhdl.

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

entity cpuswap is
   port(

          clk:   in std_logic;
          inst:  in std_logic_vector(5 downto 0);--instrucciones
          done:  out std_logic

       );
end entity cpuswap;
```

Las señales que usaremos, siguiendo el diagrama estructural. A diferencia que las señales at0,at1 y at2 en mi descripcion en vhdl estan agrupadas en un std_logic_vector nombrado a, asi como las señales a1,a2 y a3 son las conexiones entre compuertas.

```
architecture beh of cpuswap is
signal a:std_logic_vector(2 downto 0);
signal reg,at1,at0,at2,a1,a2:std_logic;
signal cop:std_logic;
signal ext:std_logic;
signal c_2_r0:std_logic;
signal c_2_r1:std_logic;
signal rt_2_bt:std_logic_vector(3 downto 0);
signal r1_2_b1:std_logic_vector(3 downto 0);
signal r0_2_b0:std_logic_vector(3 downto 0);
signal busw:std_logic_vector(3 downto 0);
```

Y listamos los componentes que usaremos:

```
    component registro is
       port(
              ent:  in std_logic_vector(3 downto 0);
              h:    in std_logic;
              clk:  in std_logic;
              sal:  out std_logic_vector(3 downto 0)
           );

    end component registro;

    component control is
       port(

              clk:   in std_logic;
              swap:  in std_logic;
              t:    out std_logic_vector(2 downto 0);
              done:  out std_logic

           );
    end component control;

    component compand2 is
       port(

              x:   in std_logic;
              y:  in std_logic;
              f:    out std_logic

           );
    end component compand2;
```

```
component compor2 is
   port(

          x:   in std_logic;
          y:  in std_logic;
          f:    out std_logic

       );
end component compor2;

component compnot1 is
   port(

          x:   in std_logic;
          f:    out std_logic

       );
end component compnot1;
```

posteriormente comenzamos a alambrar los componentes.

```
begin

       u0:control
               port map(
                       swap=>cop,
                       t=>a,
                       clk=>clk,
                       done=>done
               );

       r0:registro
               port map(
                       ent=>busw,
                       sal=>r0_2_b0,
                       clk=>clk,
                       h=>c_2_r0
               );

       r1:registro
               port map(
                       ent=>busw,
                       sal=>r1_2_b1,
                       clk=>clk,
                       h=>c_2_r1
               );
       r2:registro
               port map(
                       ent=>busw,
                       sal=>rt_2_bt,
                       clk=>clk,
                       h=>at0
               );
```

```
        b0:buffer3e
                port map(
                        ent=>inst(3 downto 0),
                        h=>ext,
                        sal=>busw
                );
        b1:buffer3e
                port map(
                        ent=>r0_2_b0,
                        h=>a(0),
                        sal=>busw
                );
        b2:buffer3e
                port map(
                        ent=>r1_2_b1,
                        h=>a(1),
                        sal=>busw
                );
        b3:buffer3e
                port map(
                        ent=>rt_2_bt,
                        h=>a(2),
                        sal=>busw
                );

        c0:compnot1
                port map(
                        x=>reg,
                        f=>a1
                );
        c1:compnot1
                port map(
                        x=>cop,
                        f=>ext
                );
                );
       c2:compand2
                port map(
                         x=>a1,
                         y=>ext,
                         f=>a2
                );

       c3:compor2
                port map(
                         x=>a2,
                         y=>a(1),
                         f=>c_2_r0
                );
       c4:compor2
                port map(
                         x=>reg,
                         y=>a(2),
                         f=>c_2_r1
                );


end architecture beh;
```

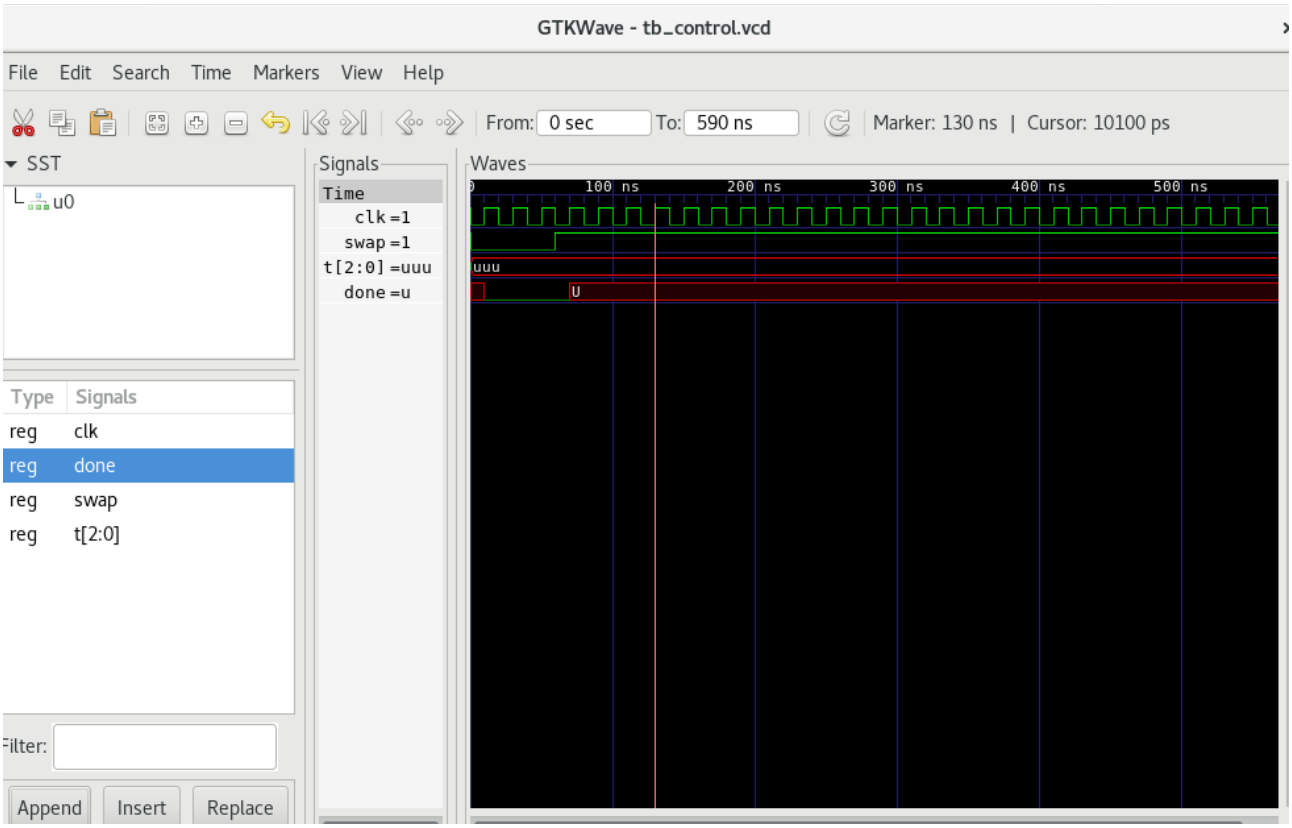
El lenguaje no detecto errores de modo que en la compilacion y ejecucion del makefile y el gtkwave no hubo problemas sin embargo todo el problema radico en la descripcion del control dado que no inserta unos en el desplazamiento.
A continuacion muestro el codigo de la unidad de control:

```
--Edgar Humberto Perez Martinez
--Ing Electronica
--Descripcion de un buffer de 3 estados
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity control is
   port(

          clk:   in std_logic;
          swap:  in std_logic;
          t:    out std_logic_vector(2 downto 0);
          done:  out std_logic

       );
end entity control;

architecture beh of control is

signal interna:std_logic_vector(3 downto 0):="0000";
signal sap:std_logic:='1';
```

podemos observar la entidad de la unidad de control, y las señales de las que haremos uso.

```
begin
interna<=swap&interna(2 downto 0);
t<=interna(2 downto 0);

        process(clk)
                begin

                if clk'event and clk = '1' then
                        if swap ='1' then
                                --interna<=swap&interna(2 downto 0);
                                for i in 0 to 2 loop
                                interna(i)<=interna(i+1);
                                end loop;
                                interna(3)<='0';
                                done<=interna(0);
                                t<=interna(2 downto 0);
                                --interna<='0' & interna(3 downto 1);

                        else
                                t<=interna(2 downto 0);
                                done<='0';
                        end if;
                end if;
        end process;
end architecture beh;
```

En esta parte describo de manera funcional la unidad de control.



Esto es lo que reporto la simulacion.