EJERCICIOS DE SISTEMAS ELECTRÓNICOS DIGITALES: HOJA 2
2º CURSO DE INGENIERÍA TÉCNICA INDUSTRIAL. ESPECIALIDAD EN ELECTRÓNICA INDUSTRIAL

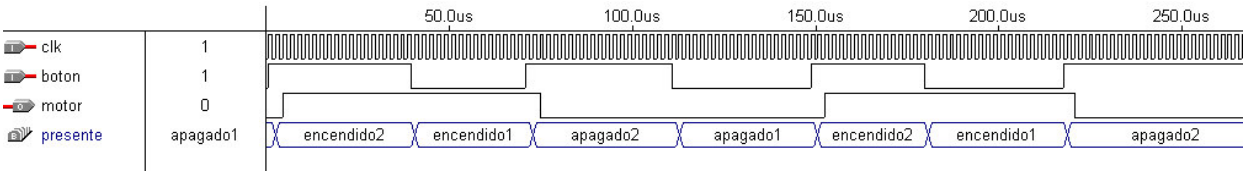
LENGUAJES DE ALTO NIVEL

1) Realiza en RTL un comparador de dos buses de 16 bits *a* y *b* de forma que indique a su salida si *a<b, a=b* ó *a>b*.

```
ENTITY comp IS
PORT(a,b: IN bit_vector(15 DOWNTO 0);
        amayb,aeqb,amenb: OUT bit);
END comp;

ARCHITECTURE flujo OF comp IS
BEGIN
        amayb<='1' WHEN a>b ELSE '0';
        aeqb <='1' WHEN a=b ELSE '0';
        amenb<='1' WHEN a<b ELSE '0';
END flujo;
```

2) Un motor eléctrico viene controlado por un único botón. Cuando se pulsa el motor pasa de encendido a apagado. Si se vuelve a pulsar el motor pasará otra vez al estado apagado. Sintetizar en VHDL el circuito de control del motor.

```
 ENTITY conmutador IS
        PORT (
                boton: IN bit;
                clk:        IN bit;
                motor: OUT bit);
END conmutador;
ARCHITECTURE moore OF conmutador IS
TYPE estado IS (apagado1,apagado2,encendido1,encendido2);
SIGNAL presente: estado:=apagado1;
BEGIN
PROCESS(clk)
BEGIN
        IF (clk'EVENT AND clk='1') THEN
                CASE presente IS
                        WHEN apagado1 =>
                                motor<='0';
                                IF boton='1' THEN
                                        presente<=encendido2;
                                END IF;
                        WHEN encendido2 =>
                                motor<='1';
                                IF boton='0' THEN
                                        presente<=encendido1;
                                END IF;
                        WHEN encendido1 =>
                                motor<='1';
                                IF boton='1' THEN
                                        presente<=apagado2;
                                END IF;
                        WHEN apagado2 =>
                                motor<='0';
                                IF boton='0' THEN
                                        presente<=apagado1;
                        END IF;
                END CASE;
        END IF;
END PROCESS;
END moore;
```

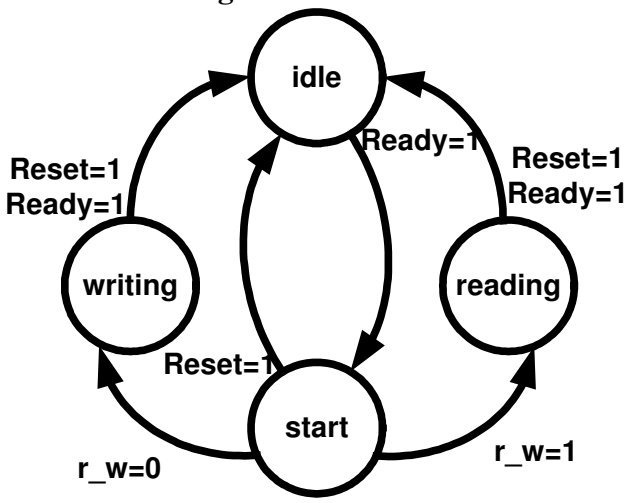


3) Implementa en VHDL una máquina de estados finitos (FSM) que controle un bloque de memoria. La FSM recibe dos entradas, *ready* que indica cuando la memoria esta preparada, *read/write* (r_w) que indica si se desea realizar una lectura o escritura y una señal de *reset*. La FSM genera dos variables, *oe* y *we* que se aplican al "output enable" y al "write enable" del bloque de memoria. El diagrama de transición de estados y la tabla de variables de salida en función del estado se indican a continuación.

Tabla de Salida

Estado	oe	we
idle	0	0
start	0	0
writing	0	1
reading	1	0

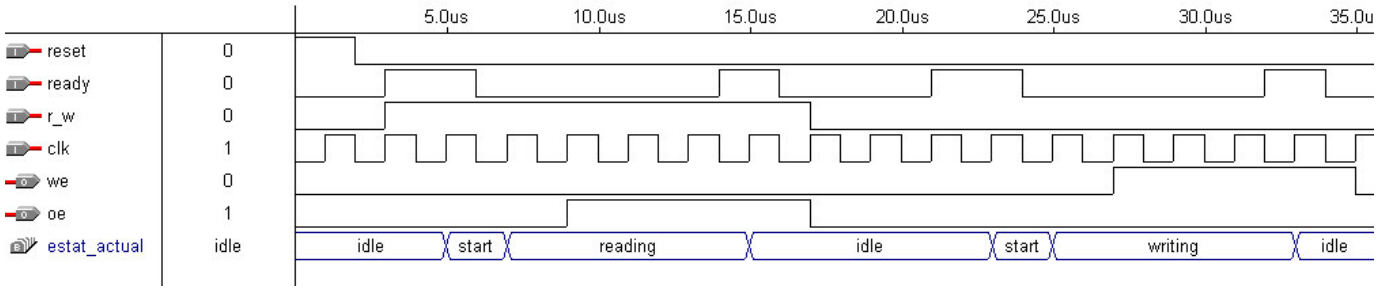
Diagrama de estados:



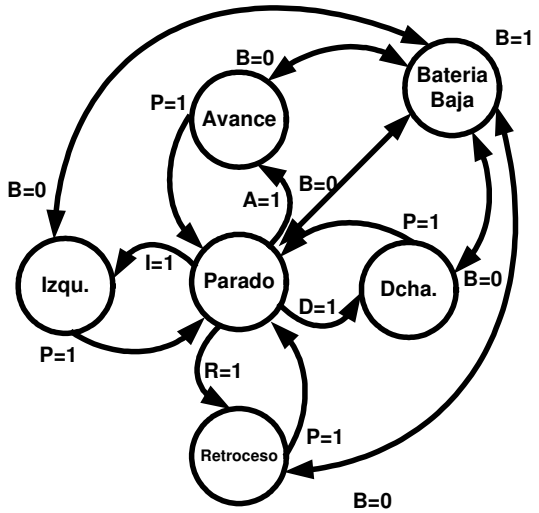
Solución:

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

entity exemple_FSM is
        port(   r_w, ready: in std_logic;
                reset, clk: in std_logic;
                oe, we: out std_logic);
end exemple_FSM;


architecture implementacio_FSM of exemple_FSM is
type estats is (idle, start, writing, reading);
signal estat_actual: estats;
```

```
begin
        procediment: process (r_w, ready, reset, clk) begin
                if reset= '1' then
                        estat_actual <= idle;
                elsif (clk'event and clk = '1') then
                        case estat_actual is
                                when idle => oe <= '0'; we <= '0';
                                        if ready = '1' then
                                                estat_actual <= start;
                                        else
                                                estat_actual <= idle;
                                        end if;
                                when start => oe<= '0'; we <= '0';
                                        if r_w = '1' then
                                                estat_actual <= reading;
                                        else
                                                estat_actual <= writing;
                                        end if;
                                when reading => oe <= '1'; we <= '0';
                                        if ready = '1' then
                                                estat_actual <= idle;
                                        else
                                                estat_actual <= reading;
                                        end if;
                                when writing => oe <='0'; we <= '1';
                                        if ready = '1' then
                                                estat_actual <= idle;
                                        else
                                                estat_actual <= writing;
                                        end if;
                        end case;
                end if;
        end process procediment;
end implementacio_FSM;
```

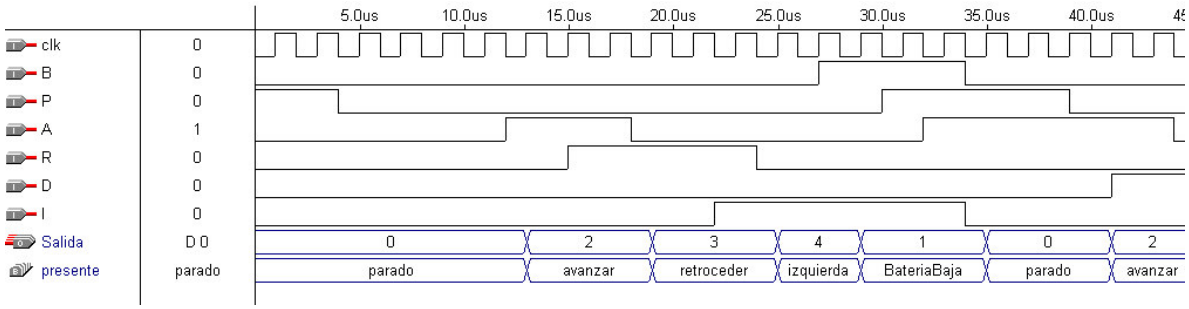


4) Se desea implementar un bloque de control del movimiento de un ROBOT. Este bloque tendrá seis señales de entrada D, I, A, R, P, B (indicando respectivamente si se quiere girar a la Derecha o la Izquierda, Avanzar, Retroceder, Pararse o indicar que las baterias están bajas). En total tendremos seis estados posibles. Implementa un diagrama de estados que realice este comportamiento además de su implementación en VHDL.




```
ENTITY controlRobot IS
PORT(
        clk,D,I,A,R,P,B: IN bit;
        Salida: OUT INTEGER RANGE 0 TO 5);
END controlRobot;
ARCHITECTURE flujo OF controlRobot IS
TYPE estado IS (parado, retroceder, avanzar, izquierda, derecha, BateriaBaja);
SIGNAL presente: estado:=parado;
BEGIN
proceso:PROCESS (clk)
        BEGIN
                IF (clk'EVENT AND clk='1') THEN
                        CASE presente IS
                                WHEN parado =>
                                        IF B='1' THEN
                                                presente<=BateriaBaja;
                                        ELSIF P='1' THEN
                                                presente<=parado;
                                        ELSIF A='1' THEN
                                                presente<=avanzar;
                                        ELSIF R='1' THEN
                                                presente<=retroceder;
                                        ELSIF D='1' THEN
                                                presente<=derecha;
                                        ELSIF I='1' THEN
                                                presente<=izquierda;
                                        ELSE
                                                presente<=parado;
                                        END IF;
                                WHEN retroceder =>
                                        IF B='1' THEN
                                                presente<=BateriaBaja;
                                        ELSIF P='1' THEN
                                                presente<=parado;
                                        ELSIF R='1' THEN
                                                presente<=retroceder;
                                        ELSIF A='1' THEN
                                                presente<=avanzar;
                                        ELSIF D='1' THEN
                                                presente<=derecha;
                                        ELSIF I='1' THEN
```

```
                                        presente<=izquierda;
                                ELSE
                                        presente<=retroceder;
                                END IF;
                        WHEN avanzar =>
                                IF B='1' THEN
                                        presente<=BateriaBaja;
                                ELSIF P='1' THEN
                                        presente<=parado;
                                ELSIF A='1' THEN
                                        presente<=avanzar;
                                ELSIF R='1' THEN
                                        presente<=retroceder;
                                ELSIF D='1' THEN
                                        presente<=derecha;
                                ELSIF I='1' THEN
                                        presente<=izquierda;
                                ELSE
                                        presente<=avanzar;
                                END IF;
                        WHEN derecha =>
                                IF B='1' THEN
                                        presente<=BateriaBaja;
                                ELSIF P='1' THEN
                                        presente<=parado;
                                ELSIF D='1' THEN
                                        presente<=derecha;
                                ELSIF A='1' THEN
                                        presente<=avanzar;
                                ELSIF R='1' THEN
                                        presente<=retroceder;
                                ELSIF I='1' THEN
                                        presente<=izquierda;
                                ELSE
                                        presente<=derecha;
                                END IF;
                        WHEN izquierda =>
                                IF B='1' THEN
                                        presente<=BateriaBaja;
                                ELSIF P='1' THEN
                                        presente<=parado;
                                ELSIF I='1' THEN
                                        presente<=izquierda;
                                ELSIF A='1' THEN
                                        presente<=avanzar;
                                ELSIF R='1' THEN
                                        presente<=retroceder;
                                ELSIF D='1' THEN
                                        presente<=derecha;
                                ELSE
                                        presente<=izquierda;
                                END IF;
                        WHEN BateriaBaja =>
                                IF B='0' THEN
                                        presente<=parado;
                                ELSE
                                        presente<=BateriaBaja;
                                END IF;
                        END CASE;
                END IF;
END PROCESS proceso;
Salida <=  0 WHEN presente=parado                ELSE
           1 WHEN presente=BateriaBaja ELSE
           2 WHEN presente=avanzar         ELSE
           3 WHEN presente=retroceder     ELSE
           4 WHEN presente=izquierda      ELSE
           5 ;
END flujo;
```


5.0us
10.0us
15.0us
20.0us
25.0us
30.0us
35.0us
40.0us
clk
0
B
0
P
0
A
1
R
0
D
0
I
0
Salida
D 0
0
2
3
4
1
0
2
presente
parado
parado
avanzar
retroceder
izquierda
BateriaBaja
parado
avanzar