

INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2026

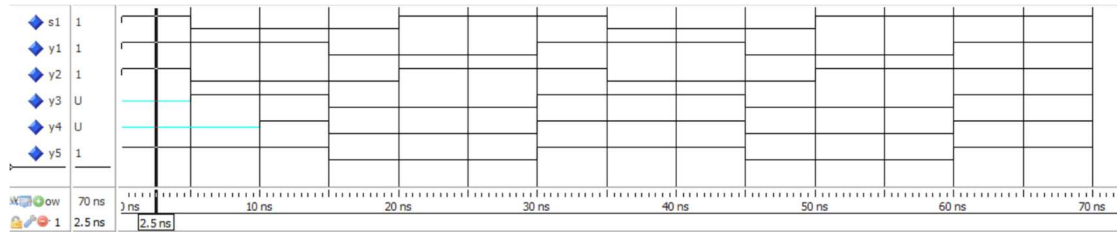
PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales y1, y2, y3, y4 e y5 entre los instantes 0 y 70 ns. Indique el valor inicial de cada una de las señales, así como el instante en que cambian de valor y el nuevo valor.

```
library IEEE;
library IEEE;
use IEEE.std_logic_1164.all;
entity cronol is
end entity cronol;
architecture comportamiento of cronol is
    signal s1, y1, y2, y3, y4 : std_logic;
    signal y5 : std_logic := '1';
begin
    process (s1, y5)
        variable aux : std_logic;
    begin
        aux := s1;
        y1 <= y5;
        y2 <= aux;
        y3 <= y2;
    end process;
    s1 <= '1', '0' after 5 ns, '1' after 20 ns,
        '0' after 35 ns, '1' after 50 ns;
    y4 <= transport s1 after 10 ns;
    y5 <= s1 after 10 ns;
end architecture comportamiento;
```

Solución a la Pregunta 1

En la siguiente figura se muestra el cronograma de evolución de las señales.

**PREGUNTA 2 (3 puntos)**

Se desea diseñar, mediante VHDL, un sistema secuencial síncrono que controle el funcionamiento de una puerta automática de garaje. El sistema debe operar de forma síncrona en el flanco de subida de la señal de reloj.

La puerta puede encontrarse en uno de los siguientes cuatro estados: CERRADA, mientras está completamente bajada; ABRIENDO, mientras el motor está elevando la puerta; ABIERTA, mientras está completamente subida; y CERRANDO, mientras el motor la está bajando.

El circuito dispone como entradas de una señal de reloj (`clk`), una señal de reinicio síncrono (`reset`) activa a nivel alto que pone al sistema en el estado CERRADA, una señal de pulsador (`boton`), y dos sensores de posición (`fin_abierta` y `fin_cerrada`). Como salidas, el sistema tiene las señales `motor_sube` y `motor_baja`. La señal `boton` tiene el valor '1' cuando el usuario acciona el botón, y tiene valor '0' en caso contrario. La señal `fin_abierta` tiene valor '1' mientras el sensor detecta que la puerta está totalmente abierta. La señal `fin_cerrada` tiene valor '1' mientras el sensor detecta que la puerta está totalmente cerrada. El sistema deberá activar el motor en sentido de subida dando valor '1' a la señal `motor_sube` y deberá activar el motor en sentido de bajada dando valor '1' a la señal `motor_baja`.

Cuando la puerta está en estado CERRADA, la pulsación del botón provoca que el sistema pase al estado ABRIENDO. Una vez que la puerta alcanza el final de carrera superior (señal `fin_abierta` tiene valor '1'), el sistema pasa al estado ABIERTA. Desde este estado, una nueva pulsación del botón hace que la puerta pase al estado CERRANDO, hasta alcanzar el final de carrera inferior (`fin_cerrada` tiene valor

'1'), volviendo así al estado CERRADA. Además, por motivos de seguridad, si la puerta se encuentra en movimiento (estados ABRIENDO o CERRANDO) y se pulsa el botón, el sistema debe invertir el sentido del movimiento, pasando inmediatamente al estado opuesto.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. La **entity** del circuito a diseñar es la siguiente:

```
entity garaje is
    port( motor_sube : out std_logic;
          motor_baja : out std_logic;
          clk : in std_logic;
          reset : in std_logic;
          boton : in std_logic;
          fin_abierta : in std_logic;
          fin_cerrada : in std_logic);
end entity garaje;
```

Solución a la Pregunta 2

El Código VHDL 1.1 es una posible solución a la Pregunta 2.

```
1
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4 architecture comportamiento of garaje is
5
6     type estado is (CERRADA, ABRIENDO, ABIERTA, CERRANDO);
7     signal estado_actual, estado_siguiete : estado;
8
9 begin
10 process(clk)
11 begin
12     if rising_edge(clk) then
13         if reset = '1' then
14             estado_actual <= CERRADA;
15         else
16             estado_actual <= estado_siguiete;
17         end if;
18     end if;
19 end process;
20 -- Lógica de transición de estados
21 process(estado_actual, boton, fin_abierta, fin_cerrada)
22 begin
23     case estado_actual is
24         when CERRADA =>
25             if boton = '1' then
```

```

26         estado_siguiete <= ABRIENDO;
27     else
28         estado_siguiete <= CERRADA;
29     end if;
30 when ABRIENDO =>
31     if boton = '1' then
32         estado_siguiete <= CERRANDO;    -- inversión de sentido
33     elsif fin_abierta = '1' then
34         estado_siguiete <= ABIERTA;
35     else
36         estado_siguiete <= ABRIENDO;
37     end if;
38 when ABIERTA =>
39     if boton = '1' then
40         estado_siguiete <= CERRANDO;
41     else
42         estado_siguiete <= ABIERTA;
43     end if;
44 when CERRANDO =>
45     if boton = '1' then
46         estado_siguiete <= ABRIENDO;    -- inversión de sentido
47     elsif fin_cerrada = '1' then
48         estado_siguiete <= CERRADA;
49     else
50         estado_siguiete <= CERRANDO;
51     end if;
52 end case;
53 end process;
54 -- Lógica de salida (tipo Moore)
55 process (estado_actual)
56 begin
57     case estado_actual is
58         when CERRADA =>
59             motor_sube <= '0';
60             motor_baja <= '0';
61         when ABRIENDO =>
62             motor_sube <= '1';
63             motor_baja <= '0';
64         when ABIERTA =>
65             motor_sube <= '0';
66             motor_baja <= '0';
67         when CERRANDO =>
68             motor_sube <= '0';
69             motor_baja <= '1';
70     end case;
71 end process;
72 end architecture comportamiento;

```

Código VHDL 1.1: Circuito controlador de la puerta del garaje.

PREGUNTA 3 (3 puntos)

Se desea desarrollar un banco de pruebas en VHDL para verificar parcialmente el comportamiento del circuito diseñado al contestar la Pregunta 2. El banco de pruebas deberá cumplir estrictamente las siguientes especificaciones:

- La señal de reloj `clk` deberá ser periódica, con un periodo de 20 ns y ciclo de trabajo del 50 %. Es decir, permanecerá a '0' durante 10 ns y a '1' durante los siguientes 10 ns, repitiéndose continuamente durante toda la simulación.
- La señal `reset` será activa a nivel alto durante los primeros 40 ns de simulación y después permanecerá a '0'.
- La puerta se considera inicialmente cerrada, por lo que al inicio la señal `fin_cerrada` tiene valor '1' y la señal `fin_abierta` tiene valor '0'.
- Se generará un pulso en `boton` de duración exactamente igual a un periodo de reloj para iniciar la apertura.
- Tras 60 ns desde la activación del botón, se activará la señal `fin_abierta`.
- Posteriormente, se aplicará un nuevo pulso al botón para iniciar el cierre, manteniendo las mismas condiciones temporales.
- Durante el cierre, se aplicará un pulso adicional en `boton` antes de que se active la señal `fin_cerrada`, forzando la inversión del movimiento.

El banco de pruebas deberá verificar automáticamente el comportamiento del sistema utilizando sentencias `assert`. En particular:

- Se comprobará que la señal `motor_sube` se activa tras el primer pulso del botón.
- Se comprobará que `motor_baja` se activa tras el segundo pulso.
- Se comprobará que, al producirse la inversión de movimiento, cambia la señal de control del motor adecuadamente.

Cada comprobación deberá generar un mensaje de error claro y específico mediante la instrucción `assert report` en caso de fallo. Finalmente, una vez ejecutadas todas las pruebas, el banco de pruebas deberá finalizar explícitamente la simulación, evitando la ejecución indefinida del proceso de estímulos.

Solución a la Pregunta 3

El código VHDL 1.2 describe el comportamiento del circuito.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity bp_garaje is
5  end entity;
6
7  architecture test of bp_garaje is
8
9  -- Señales del banoo de pruebas
10 signal clk : std_logic := '0';
11 signal reset : std_logic := '1';
12 signal boton : std_logic := '0';
13 signal fin_abierta : std_logic := '0';
14 signal fin_cerrada : std_logic := '1';
15 signal motor_sube : std_logic;
16 signal motor_baja : std_logic;
17 begin
18 -- Instancia del DUT
19 UUT: entity work.garaje
20 port map (
21     motor_sube => motor_sube,
22     motor_baja => motor_baja,
23     clk => clk,
24     reset => reset,
25     boton => boton,
26     fin_abierta => fin_abierta,
27     fin_cerrada => fin_cerrada );
28 -- Generación de reloj con asignación concurrente
29 clk <= not clk after 10 ns;
30 -- Estímulos
31 stimulus_process : process
32 begin
33     -- Reset activo durante 40 ns
34     wait for 40 ns;
35     reset <= '0';
36     -----
37     -- 1. Pulso para iniciar apertura
38     -----
39     boton <= '1';
40     wait for 20 ns;  -- 1 periodo de reloj
41     boton <= '0';
42     wait for 20 ns;
43     assert motor_sube = '1'
44         report "ERROR: motor_sube no activado al iniciar apertura"
45         severity error;
46     -- Activar fin_abierta tras 60 ns
47     wait for 60 ns;
48     fin_abierta <= '1';
49     fin_cerrada <= '0';
50     wait for 20 ns;
51     assert motor_sube = '0' and motor_baja = '0'

```

```

52     report "ERROR: la puerta no se detiene al estar abierta";
53     -----
54     -- 2. Pulso para iniciar cierre
55     -----
56     boton <= '1';
57     wait for 20 ns;
58     boton <= '0';
59     wait for 20 ns;
60     assert motor_baja = '1'
61         report "ERROR: motor_baja no activado al iniciar cierre";
62     -----
63     -- 3. Inversión de sentido durante cierre
64     -----
65     wait for 20 ns;
66     boton <= '1';
67     wait for 20 ns;
68     boton <= '0';
69     wait for 20 ns;
70     assert motor_sube = '1'
71         report "ERROR: no se invierte el movimiento durante el cierre";
72     -----
73     -- Fin de simulación
74     -----
75     wait;
76 end process;
77 end architecture;

```

Código VHDL 1.2: Banco de pruebas.

PREGUNTA 4 (2 puntos)

Se desea diseñar en VHDL un circuito secuencial síncrono que detecte la aparición de una secuencia específica de bits en una entrada serie. El circuito operará con una señal de reloj y evaluará la entrada en el flanco de subida de dicha señal.

La secuencia a detectar es "11010". Cada vez que esta secuencia aparezca en la entrada x, el circuito deberá activar la salida z a '1' durante un ciclo de reloj. En el resto de los casos, la salida permanecerá a '0'. Se permitirá la detección de secuencias solapadas; es decir, el sistema deberá ser capaz de detectar una nueva secuencia incluso si comparte parte de los bits con una detección anterior.

El circuito dispondrá de las siguientes señales de entrada y salida:

- Entrada clk: señal de reloj.
- Entrada reset: señal de reinicio síncrono, activa a nivel alto.
- Entrada x: entrada serie de datos.

- Salida z: señal de detección de la secuencia.

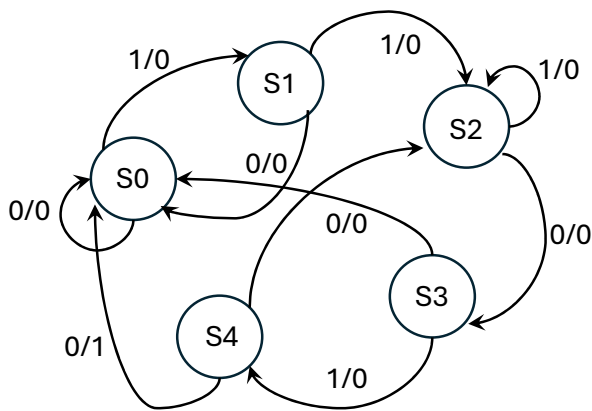
La **entity** del circuito a diseñar es la siguiente:

```
entity detector is
  port( z      : out std_logic;
        clk    : in  std_logic;
        reset  : in  std_logic;
        x      : in  std_logic);
end entity detector;
```

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Mealy. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

Solución a la Pregunta 4

El diagrama de estados del circuito detector se muestra en la siguiente figura.



El Código VHDL 1.3 es el diseño del detector de secuencias.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 architecture comportamiento of detector is
5     type estado is (S0, S1, S2, S3, S4);
6     signal estado_actual, estado_siguiete : estado;
7 begin
8     process(clk) -- Registro de estado (síncrono)
9     begin
```

```

10     if rising_edge(clk) then
11         if reset = '1' then
12             estado_actual <= S0;
13         else
14             estado_actual <= estado_siguiete;
15         end if;
16     end if;
17 end process;
18 -- Lógica de transición y salida (Mealy)
19 process(estado_actual, x)
20 begin
21     z <= '0'; -- valor por defecto
22     case estado_actual is
23         when S0 =>
24             if x = '1' then
25                 estado_siguiete <= S1;
26             else
27                 estado_siguiete <= S0;
28             end if;
29         when S1 => -- detectado "1"
30             if x = '1' then
31                 estado_siguiete <= S2; -- "11"
32             else
33                 estado_siguiete <= S0;
34             end if;
35         when S2 => -- detectado "11"
36             if x = '0' then
37                 estado_siguiete <= S3; -- "110"
38             else
39                 estado_siguiete <= S2; -- "111" ? mantiene "11"
40             end if;
41         when S3 => -- detectado "110"
42             if x = '1' then
43                 estado_siguiete <= S4; -- "1101"
44             else
45                 estado_siguiete <= S0;
46             end if;
47         when S4 => -- detectado "1101"
48             if x = '0' then
49                 estado_siguiete <= S0;
50                 z <= '1'; -- Secuencia completa "11010"
51             else
52                 estado_siguiete <= S2; -- "11011" ? conserva "11"
53             end if;
54     end case;
55 end process;
56 end architecture comportamiento;

```

Código VHDL 1.3: Circuito detector de secuencias.