Facultad de Informática
Universidad Complutense de Madrid

EXAMEN DE SISTEMAS EMPOTRADOS

CURSO 2023-24, FINAL, 19 DE ENERO DE 2024

1. **(1 punto)** Un sistema empotrado dispone de un sistema de memoria central constituido por una memoria principal Mp de 16 Mpalabras y una cache Mc de 4 Kpalabras asociativa por conjuntos, con 4 bloques por conjunto y 16 palabras por bloque, con política de reemplazamiento RLU. Se ejecuta 5 veces el programa que referencia las siguientes direcciones: 1024, 1025, …, 5134, 5173, 5174,…, 5189.

   Si el tiempo de acceso a Mp es 20 veces superior al de Mc se pide:

   a). Interpretación de los bits de la dirección física del sistema de memoria para Mc. (0.2)

   b). Evolución de los conjuntos de bloques durante la ejecución del programa. (0.4)

   c). Tiempo medio de acceso a memoria. (0.4)

2. **(0.5 puntos)** Optimización del consumo de sistemas empotrados a nivel de aplicación/sistema.

3. **(0.50 puntos)** Bus USB.

4. **(0.25 puntos)** Protocolo Bluetooth

5. **(0.75 punto)** Diseño del controlador de los leds de colores de la placa de expansión que utilizamos en el laboratorio.

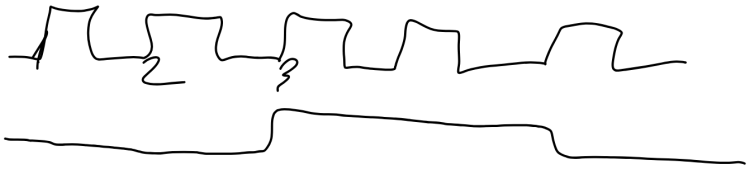
6. **(2 puntos)** En la página siguiente se dispone del código *user*_logic.*vhd*, generado por la herramienta EDK utilizada en los laboratorios, para añadir un periférico mapeado en memoria, que es un emisor de infrarrojos, accesible a través de un registro reg0. El puerto de salida del emisor de infrarrojos se llama *code_infrared* . Los bits 31-24 del *reg0* se corresponden con el código a enviar por *code_infrared*, y el bit 0 indica si el código se envía de forma continua o solo una vez. Se pide realizar varias modificaciones sobre el propio código. Puede utilizarse papel anexo, pero debe quedar bien claro en qué parte del código inicial se realizan las modificaciones.

- a). Modificar el código para generar un Divisor de frecuencias que obtiene en *clk_aux1* una señal de reloj de 38 KHz (frecuencia portadora)
  (0.5 puntos)

b.) Implementar el código para enviar, a una frecuencia de 38KHz, el valor almacenado en *reg0* (31 downto 24) (0.5 puntos).

c) Añadir el código necesario para que el valor se esté enviando continuamente si *reg0(0)* =1 o solo una vez si *reg0(0)* =0. (0.5 puntos)

d.) Realizar las modificaciones del código necesarias para añadir un registro de estado (*reg1*) que cargue en el bit 31 un 0 cada vez que se escribe en el registro de control (*reg0*), y un 1 la primera vez que se envía el valor almacenado en *reg0.* Este registro lo puede leer *microblaze* y solo lo escribe el controlador. (0.5 puntos)



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

library proc_common_v2_00_a;
use proc_common_v2_00_a.proc_common_pkg.all;

entity user_logic is
  generic
  (
    C_SLV_DWIDTH                : integer              := 32;
    C_NUM_REG                   : integer              := 12
  port
  (
    code_infrared               : OUT STD_LOGIC;
    Bus2IP_Clk                  : in  std_logic;
    Bus2IP_Reset                : in  std_logic;
    Bus2IP_Data                 : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
    Bus2IP_BE                   : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                 : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE                 : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data                 : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck                : out std_logic;
    IP2Bus_WrAck                : out std_logic;
    IP2Bus_Error                : out std_logic
  );

end entity user_logic;

architecture IMP of user_logic is

  signal slv_reg0                 : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg_write_sel        : std_logic_vector(0 to 0);
  signal slv_reg_read_sel         : std_logic_vector(0 to 0);
  signal slv_ip2bus_data          : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_read_ack             : std_logic;
  signal slv_write_ack            : std_logic;
  SIGNAL clk_aux1                 : STD_LOGIC;
  SIGNAL cuenta                   : STD_LOGIC_VECTOR(0 TO 11) := '000000000000';
  signal sending                  : std_logic := '1'; -- Indica si se está enviando
  signal bit_pos_to_send          : std_logic := '31; -- Señal para ver qué posición del bit mandar
  signal bit_to_send              : std_logic := 0; -- Señal que manda el bit
begin signal value_sent           : STD_LOGIC <= '0';

  slv_reg_write_sel <= Bus2IP_WrCE(0 to 0);
  slv_reg_read_sel  <= Bus2IP_RdCE(0 to 0);
  slv_write_ack     <= Bus2IP_WrCE(0);
  slv_read_ack      <= Bus2IP_RdCE(0);
```

Nombre del Alumno:
Apellidos: DNI:

```
SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
begin

  if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
    if Bus2IP_Reset = '1' then
      slv_reg0 <= (others => '0'); slv_reg1 <= (OTHERS => '0');
    else
     case slv_reg_write_sel is
       when "1" =>
                    slv_reg0 <= Bus2IP_Data; slv_reg1(31) <= '0';
          end if;
        end loop;

       when others => null;
      end case;
    end if;
  end if;
end process SLAVE_REG_WRITE_PROC;

                                                              slv_reg1
SLAVE_REG_READ_PROC : process( slv_reg_read_sel, slv_reg0 ) is
begin

  case slv_reg_read_sel is
    when "1" => slv_ip2bus_data <= slv_reg0;
   WHEN "10" => slv_ip2bus_data <= slv_reg1;
    when others => slv_ip2bus_data <= (others => '0');
  end case;

end process SLAVE_REG_READ_PROC;

  IP2Bus_Data  <= slv_ip2bus_data when slv_read_ack = '1' else
                  (others => '0');

  IP2Bus_WrAck <= slv_write_ack;
  IP2Bus_RdAck <= slv_read_ack;
  IP2Bus_Error <= '0';
  IP2Bus_clk1 <= clk_aux1;
  code_infrared <= bit_to_send;
end IMP;
```

```
IF value_sent = '0' THEN
 slv_reg1(31) <= '1';
 value_sent <= '1';
END IF;
```

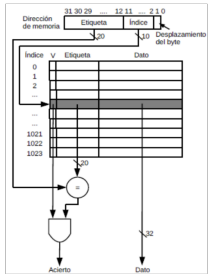
```
CONTADOR : PROCESS (Bus2IP_Clk) is
begin
   IF rising_edge(Bus2IP_Clk) AND Bus2IP_Clk = '1' THEN
      IF Bus2IP_Reset = '1' THEN
         cuenta <= (OTHERS => 0);
         clk_aux <= '0';
      ELSE
         IF cuenta = '101010000111' THEN
            cuenta <= (OTHERS => 0);
            clk_aux1 <= NOT clk_aux1;
         ELSE
            cuenta <= cuenta + '1';
         END IF;
      END IF;
   END IF;
END PROCESS CONTADOR;
```

```
SEND_REG0_DATA : PROCESS(IP2Bus_clk1) is
   begin
      IF rising_edge(IP2Bus_clk1) AND IP2Bus_clk1 = '1'
THEN
         IF sending = '1' THEN
            bit_to_send <= slv_reg0(bit_pos_to_send);
            bit_pos_to_send <= bit_pos_to_send - 1;
            IF bit_pos_to_send = 23 THEN
               bit_pos_to_send <= 31;
               sending <= slv_reg0(0);
            END IF;
         ELSE
            sending <= slv_reg0(0);
         END IF;
      END IF;
   END PROCESS SEND_REG0_DATA;
```

1. Mp = 16 Mpalabras
   Mc = 4 Kpalabras
   ↳ 4 bloques por cjto
   y 16 palabras por bloque
   política RLU (Least Recently Used)

Dirección de memoria

Etiqueta	Indice	Desplaz. del byte



a) Interpretación de los bits de la dirección física:

1. Bits de la palabra dentro de un bloque: para cada bloque hay 16 palabras. Entonces → $\log_2 16 = 4$ bits

2. Bits del índice del cjto: Hay cuatro bloques por cjto. y la caché tiene capacidad de 4K palabras. → $2^2 \cdot 2^{10}$

Calculamos cantidad de conjuntos: $\text{Total conjuntos} = \frac{\text{Tam. caché (palabras)}}{\text{Palabras por bloque} \cdot \text{Bloques por cjto.}} = \frac{2^{12}}{16 \cdot 4} = 64$ conjuntos

Por lo que necesitamos $\log_2 64 = 6$ bits para el índice del conjunto.

3. Bits de la etiqueta: La memoria principal tiene 16×2^{20} palabras, lo que requiere $\log_2(16 \cdot 2^{20}) = 24$ bits para direccionar cada palabra. De estos, se han usado $4 + 6 = 10$ bits, por lo que quedan $24 - 10 = 14$ bits para la etiqueta.

Dir. física → E: 14 bits, Cjto: 6 bits, Palabra bloque: 4 bits.

b) Evolución de los conjuntos de bloques:

- Caché tiene 4K palabras, 16 palabras por bloque, 4 bloques por cjto, 64 conjuntos.
- Rango de dir. accedidas: 1024 a 5134 y 5173 a 5189
- Calcular índice cjto.: $\text{Índice} = \left(\frac{\text{Dirección}}{16}\right) \bmod 64$ ↳ palabras/bloque
- $\text{Nº bloque} = \frac{\text{Dirección}}{\text{Palabras por bloque}}$
- Calcular palabra dentro del bloque: $\text{Palabra} = \text{Dirección} \bmod 16$ ↳ palabras/bloque

Evolución:

Primera dirección: 1024
- Bloque → $\frac{1024}{16} = 64$ (nº bloque)
- Índice → $64 \bmod 64 = 0$
- Palabra → $1024 \bmod 16 = 0$

El bloque 64 se almacena en el cjto. 0 de la caché. Si estaba vacío, no hay reemplazo.

Segunda dirección: 1025
- Bloque → $\frac{1025}{16} = 64$
- Índice → $64 \bmod 64 = 0$
- Palabra → $1025 \bmod 16 = 1$

El bloque ya está en el cjto. 0, por LRU, se actualiza como más reciente usado, no hay reemplazo.

⋮

Se agrupan las direcciones en grupos de 16

Bloques distintos = $\frac{5134 - 1024 + 1}{16} = 257$ → Como hay 64 conjuntos, y cada uno con 4 bloques, hay 256 bloques distintos.
- Las direcciones 1024 a 5119 han llenado los bloques 64 a 319 (256 bloques).
- En la dir. 5120, se accede al bloque 320, asignado al cjto. 0, como los 4 bloques de este conjunto ya están llenos, el bloque 320 reemplazará al bloque menos reciente utilizado (LRU), es decir, el bloque 64. *

Dir. 5173 a 5189:

5173
- Bloque = $\frac{5173}{16} = 323$
- Índice = $323 \bmod 64 = 3$
- Palabra = $5173 \bmod 16 = 5$

→ Bloque 323: [5173, 5183]
Bloque 324: [5184, 5189]
→ reemplazamos 16 direcciones de los conjuntos 3 y 4.

* Conjunto 0:
- Bloque 64 → Bloque 320
- Bloque 128
- Bloque 192
- Bloque 256

Conjunto 1:
- Bloque 65
- ⋮
- 129
- ⋮
- 193
- ⋮
- 257

c) Tiempo medio de acceso a memoria:

1024
⋮
5134
5173
⋮
5189
} 5 veces

3134 − 1024 + 1 = 4111 direcciones
5189 − 5173 + 1 = 17 direcciones

Total accesos = (4111 + 17) · 5 = 20640 accesos

1ª vuelta → $\frac{4111}{16} \simeq 256 + 3 = 259$ fallos
↳ nº palabras por bloque

Los 3 fallos son por el acceso a 3 conjuntos distintos.

El resto de fallos son: 4 · 3 · 5 = 60 fallos
- 4 → nº de fallos en cada fila
- 3 → nº de fallos de tags distintos
- 5 → nº de vueltas que quedan

Total fallos = 259 + 60 = 319 → tasa fallos → $\frac{319}{20640} = 0'0155$

Total aciertos = 20640 − 319 = 20321 → tasa aciertos → $\frac{20321}{20640} = 0'9845$

$T_{medio} = (\text{Tasa aciertos} \cdot T_c) + (\text{Tasa fallos} \cdot T_p) = 0'9845 \cdot T_c + 0'0155 \cdot 20 \cdot T_c = 1'2945\ T_c$