Facultad de Informática
Universidad Complutense de Madrid

EXAMEN DE SISTEMAS EMPOTRADOS

CURSO 2015-16, FINAL, 15 DE SEPTIEMBRE DE 2016

1. (1 punto) Un sistema empotrado dispone de un sistema de memoria central constituido por una memoria principal Mp y una cache Mc. Mp tiene una dimensión de 4 M palabras y está estructurada como un conjunto de módulos de 256 K palabras con entrelazado de orden inferior. Mc tiene un tamaño de 64 K palabras organizada en 512 conjuntos. Se pide:

a. Calcular el tamaño de línea para minimizar el tiempo de transferencia de bloques entre Mp y Mc. Calcular para dicho tamaño de línea el grado de asociatividad. (**0. 3 puntos**)
b. Para el tamaño de línea elegido anteriormente mostrar la interpretación de los bits de la dirección física del sistema de memoria para Mp y Mc (**0.4 puntos**)
c. En un momento determinado de la ejecución de un programa se determina que el contenido del campo etiqueta de 4 posiciones de la cache es el siguiente (todos los datos están en hexadecimal):

Conjunto 0x031 etiqueta 0x020 Conjunto 0x049 etiqueta 0x177

Conjunto 0x068 etiqueta 0x118 Conjunto 0x1F8 etiqueta 0x1FF

Determinar el rango de direcciones físicas almacenadas en dichas posiciones. (**0.3 puntos**)

2. (0.5 punto) Optimización del consumo de sistemas empotrados a nivel de sistema

3. (1 punto) Bus USB

4. (0.5 punto) Explica brevemente cómo funciona un sensor de luz y cómo lo conectarías a un sistema empotrado.

5. (2 puntos) El siguiente código, correspondiente a un generador de sonido, permite generar una onda cuadrada (*sonido*) cuyo semiperiodo se almacena en el registro *semi:*

```
-------------------------
-- GENERADOR DEL SONIDO --
-------------------------
altavoz_p: PROCESS(Bus2IP_Clk, Bus2IP_Reset)
BEGIN
  IF(Bus2IP_Reset='1')THEN
    q <= '0';
      cs <= (OTHERS=>'0');
  ELSIF(Bus2IP_Clk'event and Bus2IP_Clk='1') THEN
       IF (slv_write_ack = '1') THEN
          q <= '0';
          cs <= (OTHERS=>'0');
       ELSIF (semi = cs) THEN
          cs <= (OTHERS=>'0');
          q <= not q;
       ELSE
          cs <= cs + 1;
```

```
            q <= q;
        END IF;
    END IF;
END PROCESS;

sonido <= q;
```

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 con un único registro de L/E. 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 user_logic.vhd para que se comporte como un controlador de altavoz de forma que el semiperiodo se pase a través del registro **reg0** y la salida del generador de sonido se saque por el puerto externo **sal0**. Añadir las entradas, salidas y señales necesarias **(0.5) punto**

b. Añadir una segunda instancia del generador de sonido cuyo semiperiodo se pase en el registro **reg1**. La salida será por **sal1**. **(0. 5) punto**

c. Añadir un registro de control ***control***, de sólo escritura, de forma que

    a. si *control*=0x00000000 no se produce sonido,
    b. si *control*=0x00000001 la salida del primer generador es por **sal0** y la del segundo por **sal1**,
    c. *si control*=0x00000002 la salida del primer generador es por **sal1** y la del segundo por **sal0**,
    d. *si control*=0x00000003 la salida del primer generador va por **sal0** y **sal**1.

   Modificar el código donde sea necesario. **(0.5) puntos**

d. Explicar cómo se podría añadir al sistema un led que se iluminara cuando se produce sonido. Modificar el código donde sea necesario. **(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              := 1
  port
  (


    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;



begin

  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);
```

```
 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');
   else
    case slv_reg_write_sel is
       when "1" =>
             slv_reg0 <= Bus2IP_Data;
         end if;
        end loop;

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

 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 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';

end IMP;
```

① Mp 4M palabras → 2^{22}

↳ módulos de 256 K palabras → 2^{18}

Mc 64 K palabras → 2^{16}

↳ 512 conjuntos → 2^9

a) Tam. línea = nº módulos Mp

$$\text{nº módulos} = \frac{2^{22}}{2^{18}} = 16 \text{ módulos}$$

Tam. línea = 16 palabras / línea.

$$\text{Gr. asoc.} = \frac{\text{nº líneas}}{\text{nº cjtos}} = \frac{\frac{2^{16}}{16}}{512} = 8.$$

b) Interpretación de bits:

Mp → 2^{22}

nº modulos = 16

$\log_2 16 = 4$ bits para el modulo

$\log_2 2^{18} = 18$ bits para la palabra.

Mc → 2^{16}

$\log_2 512 = 9$ bits para indice cjto.

$\log_2 16 = 4$ bits para la palabra

22 − 9 − 4 = 9 bits para la etiqueta.

c) Conjunto 0x031 → 00011 0001 = 49

Etiqueta 0x020 → 00010 0000 = 32

$\frac{4}{1000000}\ \frac{0}{00011}\ \frac{3}{0001}\ \frac{1}{0000}\ \frac{0}{1111}$

0x40310 − 0x4031F

C: 0x049 → 00100 1001 = 73

E: 0x177 → 1 0111 0111 = 247

$\frac{2}{1}\ \frac{E}{0111}\ \frac{E}{01110}\ \frac{8}{0100}\ \frac{9}{1001}$

⋮