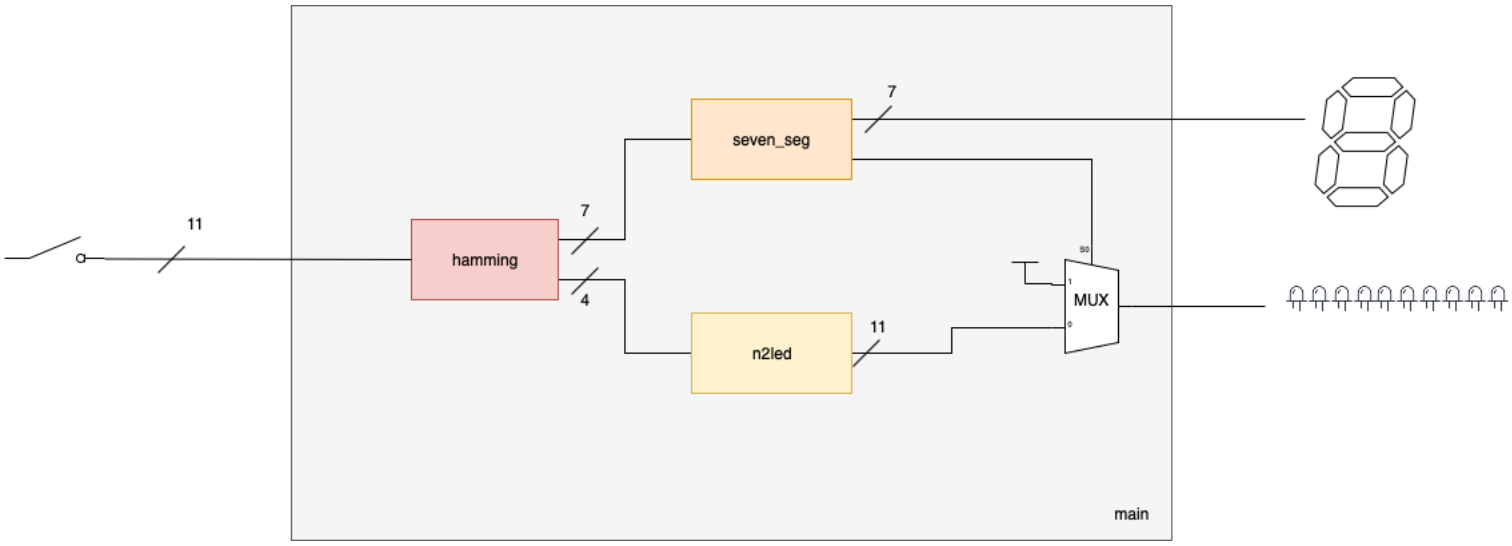
Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2713 - Sistemas Digitales

# Pauta Tarea Nº2

15 de Mayo de 2019

## 1. Diagrama de bloques




## 2. Descripción de cada módulo

- **Hamming:** Módulo que comprueba la codificación y arregla el bit de error. Este módulo tiene como salida el código codificado y el número de la posición del error.
- **Seven_seg:** Módulo decodificador que toma como entrada el código *ASCII* de una letra y entrega los segmentos que deben encenderse. Este módulo también entrega un bit que indica si la entrada es invalida.
- **Num2led:** Módulo decodificador que toma el número del led que debe encenderse y retorna los bits de salida a los leds.

# 3. Códigos

```
module hamming(input logic [10:0]p_in,
               output logic [3:0] ns,
               output logic [6:0]l_out);

   logic p1;
   logic p2;
   logic p3;
   logic p4;

   assign p1 = p_in[10] ^ p_in[8] ^ p_in[6] ^ p_in[4] ^ p_in[2] ^ p_in[0];
   assign p2 = p_in[9] ^ p_in[8] ^ p_in[5] ^ p_in[4] ^ p_in[1] ^ p_in[0];
   assign p3 = p_in[7] ^ p_in[6] ^ p_in[5] ^ p_in[4];
   assign p4 = p_in[3] ^ p_in[2] ^ p_in[1] ^ p_in[0];

   logic [3:0]n;
   assign n = {p4, p3, p2, p1};

   logic [10:0] cod;

   assign cod[10] = (n == 1) ? ~l_out[10]:l_out[10];
   assign cod[9] = (n == 2) ? ~l_out[9]:l_out[9];
   assign cod[8] = (n == 3) ? ~l_out[8]:l_out[8];
   assign cod[7] = (n == 4) ? ~l_out[7]:l_out[7];
   assign cod[6] = (n == 5) ? ~l_out[6]:l_out[6];
   assign cod[5] = (n == 6) ? ~l_out[5]:l_out[5];
   assign cod[4] = (n == 7) ? ~l_out[4]:l_out[4];
   assign cod[3] = (n == 8) ? ~l_out[3]:l_out[3];
   assign cod[2] = (n == 9) ? ~l_out[2]:l_out[2];
   assign cod[1] = (n == 10) ? ~l_out[1]:l_out[1];
   assign cod[0] = (n == 11) ? ~l_out[0]:l_out[0];


   assign l_out = {cod[8], cod[6], cod[5], cod[4], cod[2], cod[1], cod[0]};
   assign ns = n;

endmodule // hamming
```

```
module seven_seg(input logic [6:0] letra,
                 output logic e;
                 output logic [6:0] seg);

   always @(*)
   begin
       case(letra)
           7'h41:
               begin
                   seg = 7'b0001000;    // A
                   e = 0;
                   end
           7'h42:
               begin
                   seg = 7'b0000000;    // B
                   e = 0;
                   end
           7'h43:
               begin
                   seg = 7'b0110001;    // C
                   e = 0;
                   end
           7'h44:
               begin
                   seg = 7'b0000001;    // D
                   e = 0;
                   end
           7'h45:
               begin
                   seg = 7'b0110000;    // E
                   e = 0;
                   end
           7'h46:
               begin
                   seg = 7'b0111000;    // F
                   e = 0;
                   end
           7'h47:
               begin
                   seg = 7'b0100001;    // G
                   e = 0;
                   end
           7'h48:
               begin
                   seg = 7'b1001000;    // H
                   e = 0;
                   end
           7'h49:
               begin
                   seg = 7'b1001111;    // I
                   e = 0;
                   end
           7'h4A:
               begin
```

```
            seg = 7'b1000111;    // J
            e = 0;
            end
        7'h4B:
            begin
            seg = 7'b0101000;    // K
            e = 0;
            end
        7'h4C:
            begin
            seg = 7'b1110001;    // L
            e = 0;
            end
        7'h4D:
            begin
            seg = 7'b0000110;    // M
            e = 0;
            end
        7'h4E:
            begin
            seg = 7'b1101010;    // N
            e = 0;
            end
        7'h4F:
            begin
            seg = 7'b1100010;    // O
            e = 0;
            end
        7'h50:
            begin
            seg = 7'b0011000;    // P
            e = 0;
            end
        7'h51:
            begin
            seg = 7'b0001100;    // Q
            e = 0;
            end
        7'h52:
            begin
            seg = 7'b1111010;    // R
            e = 0;
            end
        7'h53:
            begin
            seg = 7'b0100100;    // S
            e = 0;
            end
        7'h54:
            begin
            seg = 7'b1110000;    // T
            e = 0;
            end
        7'h55:
            begin
```

```
                    seg = 7'b1000001;    // U
                    e = 0;
                    end
            7'h56:
                begin
                    seg = 7'b1100011;    // V
                    e = 0;
                    end
            7'h57:
                begin
                    seg = 7'b0100011;    // W
                    e = 0;
                    end
            7'h58:
                begin
                    seg = 7'b0110110;    // X
                    e = 0;
                    end
            7'h59:
                begin
                    seg = 7'b1011000;    // Y
                    e = 0;
                    end
            7'h5A:
                begin
                    seg = 7'b1011010;    // Z
                    e = 0;
                    end
            default:
                begin
                    seg = 7'b1111111;
                    e = 1;
                end
        endcase
    end
endmodule // seven_seg
```

```
module n2led(input logic [3:0]n,
             output logic [10:0]leds);
    always @(*)
    begin
        case(n)
            0: leds = 11'b00000000000;
            1: leds = 11'b10000000000;
            2: leds = 11'b01000000000;
            3: leds = 11'b00100000000;
            4: leds = 11'b00010000000;
            5: leds = 11'b00001000000;
            6: leds = 11'b00000100000;
            7: leds = 11'b00000010000;
            8: leds = 11'b00000001000;
            9: leds = 11'b00000000100;
            10: leds = 11'b00000000010;
            11: leds = 11'b00000000001;
            default: leds = 11'b00000000000;
        endcase
    end
endmodule // n2led
```

```
module main(input logic [15:5]sw,
            output logic [15:5]LED,
            output logic [6:0]seg,
            output logic [3:0]an);

    assign an = 4'b1110;

    logic [3:0] n;
    logic [6:0] l;
    hamming h1(sw, n, l);

    logic e;
    seven_seg ss(l, e, seg);

    logic [10:0] ld;
    n2led nl(n, ld);

    assign LED = e ? 11'b11111111111:ld;

endmodule // main
```

## Notas

La sintaxis de los módulos es *System* Verilog, en *Verilog* seria casi idéntica, hay que cambiar los tipos de datos de las datos que se modifican en las sentencias *always* por tipo *reg* y los demas datos a *wire*.