

# Trabajo Práctico N° 1

---

## Electrónica III - 2019

Grupo 1:

Farall, Facundo

Gaytan, Joaquín

Kammann, Lucas

Maselli, Carlos

18 de agosto de 2019

## 1. EJERCICIO 3

De los dos módulos pedidos para implementar en Verilog se tomó la decisión, de forma completamente arbitraria, de realizar uno mediante las compuertas lógicas que lo componen, y otro a través de la descripción de su comportamiento (behavioural). La razón detrás de esta decisión fue pura y exclusivamente para hacer uso de las variantes provistas por Verilog, e interiorizarnos en su estilo de programación.

### 1.1. DECODER DE 4 ENTRADAS

Recibe una entrada de 2 bits, que determinan cual de las cuatro salidas accionar. Su implementación se realizó a través de lógica de compuertas, donde únicamente la combinación correcta de los dos bits de entrada, ponen a la salida con ese número en 1 lógico. Las relaciones lógicas son relativamente sencillas y *straight-forward*, por lo cual no se considera necesario demostrarlas mediante una tabla de verdad o mapa de Karnaugh. A continuación se presenta el código:

---

```
module decoder4out(coded, y0, y1, y2, y3);
    input [1:0] coded;
    output y0, y1, y2, y3;

    assign y0 = ~coded[1] & ~coded[0];
    assign y1 = ~coded[1] & coded[0];
    assign y2 = coded[1] & ~coded[0];
    assign y3 = coded[1] & coded[0];

endmodule
```

---

### 1.2. MUX DE 4 ENTRADAS

Recibe una entrada de 4 bits con las 4 "fuentes", otra entrada de 2 bits que hace las veces de selector, y cuenta con una salida de 1 bit. La salida copiará el valor de la entrada determinada por el selector. El módulo es logró mediante una descripción del comportamiento del mismo, en el cual se le especificó qué debía realizar ante cambios en alguna de sus entradas. A continuación se presenta el código:

---

```
module mux4in (x, sel, y);
    input [3:0] x;
    input [1:0] sel;           // sel selects the exit (x[3], x[2], x[1], x[0]).
    output reg y;

    always @(sel or x) begin
        if (sel == 0)
            assign y = x[0];
        else if (sel == 1)
            assign y = x[1];
        else if (sel == 2)
            assign y = x[2];
        else if (sel == 3)
            assign y = x[3];
    end

end
```

---

endmodule

---

### 1.3. TEST-BENCH

Se sometió a los dos módulos a un testeo de su respuesta a cada una de las posibles entradas, y los resultados fueron los esperados. Los mismos pueden ser replicados ejecutando los comandos:

---

```
user@computer: path/to/EJ_3/folder$ make
user@computer: path/to/EJ_3/folder$ ./run
```

---