Lab 02
Arquitectura de Computadores
Sección 2

Joaquín Ramírez

Mayo 04, 2020

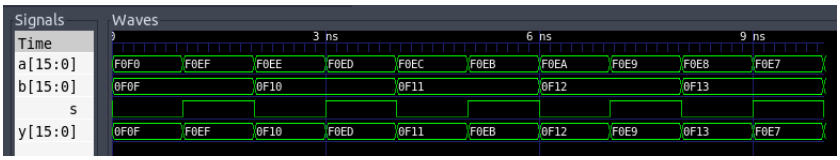
1. Será implementado en el ejercicio 2.a

2. (a) Implementación de un 2-to-1 MUX de 16-bits.La implementación es igual que un MUX 2x1 de 1 bi. La complejidad no requirió separar los módulos en *small* y *top*. Se probaron 10 tiempos, en los que cada un segundo a disminuye en 1, mientras que b aumenta en 1 cada dos segundos. Se observa que cuando el *select* s está "low", el valor de y es el de b, y cuando está "high", y toma el valor de a. El diagrama en $GTKWave$ comprueba la tabla de verdad.

```
module mux16_2x1(a, b, s, y);
        input   [15:0] a;
        input   [15:0] b;
        input   s;

        output [15:0]y;

        assign y = s? a : b;
endmodule
```

```
`timescale 1ns/1ns
module mux16_2x1_tb;
        reg s;
        reg [15:0] a,b;
        wire  [15:0]y;
        mux16_2x1 g (a,b,s,y);
        initial begin
        $display("time\t\ta\t\t\tb\t\ts\t\ty");
                a = 16'b1111000011110000;
                b = 16'b0000111100001111;
                s = 16'b00;
                #10 $finish;
        end
        initial begin
                $monitor("%2d:\t%b\t%b\t%b\t%b",$time,a,b,s,y);
        end
        always #1 a = a - 1;
        always #1 s = !s;
        always #2 b = b + 1;

        initial begin
                $dumpfile("mux16_2x1.vcd");
                $dumpvars;
        end
endmodule
```

```
time            a                       b               s               y
VCD info: dumpfile mux16_2x1.vcd opened for output.
 0:     1111000011110000        0000111100001111        0       0000111100001111
 1:     1111000011101111        0000111100001111        1       1111000011101111
 2:     1111000011101110        0000111100010000        0       0000111100010000
 3:     1111000011101101        0000111100010000        1       1111000011101101
 4:     1111000011101100        0000111100010001        0       0000111100010001
 5:     1111000011101011        0000111100010001        1       1111000011101011
 6:     1111000011101010        0000111100010010        0       0000111100010010
 7:     1111000011101001        0000111100010010        1       1111000011101001
 8:     1111000011101000        0000111100010011        0       0000111100010011
 9:     1111000011100111        0000111100010011        1       1111000011100111
10:     1111000011100110        0000111100010100        0       0000111100010100
```



(b) Implementación de un 8-to-1 MUX de 16-bits. Se crearon por separado módulos de OR, AND y NOT gate (small), y se llamaron en el top. En el test bench se puede apreciar que para cada combinación entre selects, el output es un input único. La tabla de verdad coincide con la representación en *GTKWave*.

```
module mux16_8x1_or(l, m, n, o, p, q, r, s, t);
        output [15:0]l;
        input [15:0]m, n, o, p, q, r, s, t;
assign l = m | n | o | p | q | r | s | t;
endmodule
```

(c) Implementación de un 16-to-1 MUX de 16-bits. El mismo concepto previo, pero ahora hay que aumentar 8 inputs, por lo que

```
module mux16_8x1_and(a, b, c, d, e);
        output [15:0]a;
        input [15:0]b;
        input c,d,e;
        wire validacion;
        assign validacion = (c & d & e);
assign a = validacion? b: 16'b00;
endmodule
```

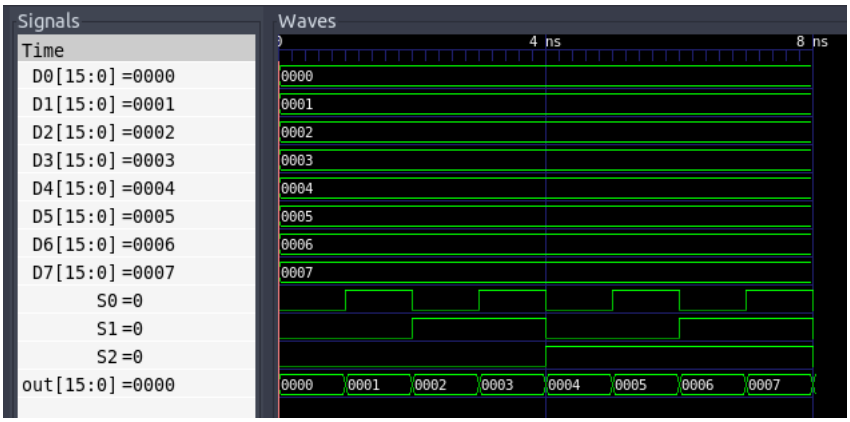
```
module mux16_8x1_not(f,h);
        output f;
        input h;
assign f = ~h;
endmodule
```

```
module mux16_8x1(D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2, out);
output [15:0]out;
input [15:0] D0,D1,D2, D3,D4,D5,D6,D7;
input S0,S1,S2;
input NS0, NS1, NS2;
wire [15:0]T1, T2, T3, T4, T5, T6, T7, T8;

mux16_8x1_not a1(NS0, S0);
mux16_8x1_not a2(NS1, S1);
mux16_8x1_not a3(NS2, S2);

mux16_8x1_and u1(T1, D0, NS0, NS1, NS2);
mux16_8x1_and u2(T2, D1, S0, NS1, NS2);
mux16_8x1_and u3(T3, D2, NS0, S1, NS2);
mux16_8x1_and u4(T4, D3, S0, S1, NS2);
mux16_8x1_and u5(T5, D4, NS0, NS1, S2);
mux16_8x1_and u6(T6, D5, S0, NS1, S2);
mux16_8x1_and u7(T7, D6, NS0, S1, S2);
mux16_8x1_and u8(T8, D7, S0, S1, S2);
mux16_8x1_or u12(out, T1, T2, T3, T4, T5, T6, T7, T8);
endmodule
```

```
`timescale 1ns/1ns
module mux16_8x1_tb;
    wire[15:0]out;
        reg  [15:0]D0, D1, D2, D3, D4, D5, D6, D7;
        reg S0, S1, S2;

    mux16_8x1 g (D0, D1, D2, D3, D4, D5, D6, D7, S0, S1, S2, out);
    initial begin
                D0 = 16'b000;
                D1 = 16'b001;
                D2 = 16'b010;
                D3 = 16'b011;
                D4 = 16'b100;
                D5 = 16'b101;
                D6 = 16'b110;
                D7 = 16'b111;
                S0 = 0;
                S1 = 0;
                S2 = 0;

$display("time\t\tD0\t\t\tD1\t\t\tD2\t\t\tD3\t\t\tD4\t\t\tD5\t\t\tD6\t\t\tD7\t\tS2\tS1\tS0\t\tY");
                $monitor("%2d:\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b",$time,D0, D1, D2, D3, D4,
D5, D6, D7, S2, S1, S0, out);
                    #8 $finish;
    end
always #1 S0 = !S0;
always #2 S1 = !S1;
always #4 S2 = !S2;
    initial begin
        $dumpfile("mux16_8x1.vcd");
        $dumpvars;
    end
endmodule
```

```
time            D0                  D1                  D2                  D3                  D4
VCD info: dumpfile mux16_8x1.vcd opened for output.
 0:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 1:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 2:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 3:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 4:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 5:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 6:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 7:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 8:     0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
```

```
        D5                  D6                  D7                  S2    S1    S0          Y

0000000000000101    0000000000000110    0000000000000111    0     0     0     0000000000000000
0000000000000101    0000000000000110    0000000000000111    0     0     1     0000000000000001
0000000000000101    0000000000000110    0000000000000111    0     1     0     0000000000000010
0000000000000101    0000000000000110    0000000000000111    0     1     1     0000000000000011
0000000000000101    0000000000000110    0000000000000111    1     0     0     0000000000000100
0000000000000101    0000000000000110    0000000000000111    1     0     1     0000000000000101
0000000000000101    0000000000000110    0000000000000111    1     1     0     0000000000000110
0000000000000101    0000000000000110    0000000000000111    1     1     1     0000000000000111
0000000000000101    0000000000000110    0000000000000111    0     0     0     0000000000000000
```



un *select* más es necesario,

```
module mux16_16x1_or(l, m, n, o, p, q, r, s, t, a, b, c, d, e, f, g, h);
        output [15:0]l;
        input [15:0]m, n, o, p, q, r, s, t, a, b, c, d, e, f, g, h;
assign l = m | n | o | p | q | r | s | t| a | b | c | d | e | f | g | h;
endmodule
```

3.

4. .

5. .

6. .

```
module mux16_16x1_and(a, b, c, d, e, w);
        output [15:0]a;
        input [15:0]b;
        input c,d,e,w;
        wire validacion;
        assign validacion = (c&d&e&w);
assign a = validacion? b : 16'b00;
endmodule
```

```
module mux16_16x1_not(f,h);
        output f;
        input h;
assign f = ~h;
endmodule
```

```
module mux16_16x1(D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, S0, S1, S2,S3,out);
output [15:0]out;
input [15:0] D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15;
input S0,S1,S2,S3;
input NS0, NS1, NS2,NS3;
wire [15:0]T1, T2, T3, T4, T5, T6, T7, T8,T9,T10,T11,T12,T13,T14,T15,T16;

mux16_16x1_not a1(NS0, S0);
mux16_16x1_not a2(NS1, S1);
mux16_16x1_not a3(NS2, S2);
mux16_16x1_not a4(NS3, S3);

mux16_16x1_and u1(T1, D0, NS0, NS1, NS2, NS3);
mux16_16x1_and u2(T2, D1, S0, NS1, NS2, NS3);
mux16_16x1_and u3(T3, D2, NS0, S1, NS2, NS3);
mux16_16x1_and u4(T4, D3, S0, S1, NS2, NS3);
mux16_16x1_and u5(T5, D4, NS0, NS1, S2, NS3);
mux16_16x1_and u6(T6, D5, S0, NS1, S2, NS3);
mux16_16x1_and u7(T7, D6, NS0, S1, S2, NS3);
mux16_16x1_and u8(T8, D7, S0, S1, S2, NS3);
mux16_16x1_and u9(T9, D8, NS0, NS1, NS2, S3);
mux16_16x1_and u10(T10, D9, S0, NS1, NS2, S3);
mux16_16x1_and u11(T11, D10, NS0, S1, NS2, S3);
mux16_16x1_and u12(T12, D11, S0, S1, NS2, S3);
mux16_16x1_and u13(T13, D12, NS0, NS1, S2, S3);
mux16_16x1_and u14(T14, D13, S0, NS1, S2, S3);
mux16_16x1_and u15(T15, D14, NS0, S1, S2, S3);
mux16_16x1_and u16(T16, D15, S0, S1, S2, S3);
mux16_16x1_or u17(out, T1, T2, T3, T4, T5, T6, T7, T8,T9,T10,T11,T12,T13,T14,T15,T16);
endmodule
```

```verilog
`timescale 1ns/1ns
module mux16_16x1_tb;
    wire[15:0]out;
        reg  [15:0]D0, D1, D2, D3, D4, D5, D6, D7,D8, D9, D10, D11, D12, D13, D14, D15;
reg S0, S1, S2, S3;

    mux16_16x1 g (D0, D1, D2, D3, D4, D5, D6, D7,D8, D9, D10, D11, D12, D13, D14, D15,S0, S1, S2,
S3,out);
    initial begin
                D0 = 16'b0000;
                D1 = 16'b0001;
                D2 = 16'b0010;
                D3 = 16'b0011;
                D4 = 16'b0100;
                D5 = 16'b0101;
                D6 = 16'b0110;
                D7 = 16'b0111;
                D8 = 16'b1000;
                D9 = 16'b1001;
                D10 = 16'b1010;
                D11 = 16'b1011;
                D12 = 16'b1100;
                D13 = 16'b1101;
                D14 = 16'b1110;
                D15 = 16'b1111;
                S0 = 0;
                S1 = 0;
                S2 = 0;
                S3 = 0;

$display("time\t\tD0\t\t\tD1\t\t\tD2\t\t\tD3\t\t\tD4\t\t\tD5\t\t\tD6\t\t\tD7\t\t\tD8\t\t\tD9\t\t\t
D10\t\t\tD11\t\t\tD12\t\t\tD13\t\t\tD14\t\t\tD15\t\t\tS3\tS2\tS1\tS0\t\tY");
$monitor("%2d:\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b",
$time,D0, D1, D2, D3, D4, D5, D6, D7,D8, D9, D10, D11, D12, D13, D14, D15,S3, S2, S1, S0,out);
#16 $finish;
    end
always #1 S0 = !S0;
always #2 S1 = !S1;
always #4 S2 = !S2;
always #8 S3 = !S3;
    initial begin
        $dumpfile("mux16_16x1.vcd");
        $dumpvars;
    end
endmodule
```

```
time      D0                  D1                  D2                  D3                  D4
VCD info: dumpfile mux16_16x1.vcd opened for output.
 0:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 1:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 2:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 3:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 4:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 5:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 6:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 7:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 8:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
 9:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
10:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
11:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
12:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
13:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
14:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
15:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
16:      0000000000000000    0000000000000001    0000000000000010    0000000000000011    0000000000000100
```

D5	D6	D7	D8	D9
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001
0000000000000101	0000000000000110	0000000000000111	0000000000001000	0000000000001001

D10	D11	D12	D13	D14
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110
0000000000001010	0000000000001011	0000000000001100	0000000000001101	0000000000001110

S3	S2	S1	S0	Y
0	0	0	0	0000000000000000
0	0	0	1	0000000000000001
0	0	1	0	0000000000000010
0	0	1	1	0000000000000011
0	1	0	0	0000000000000100
0	1	0	1	0000000000000101
0	1	1	0	0000000000000110
0	1	1	1	0000000000000111
1	0	0	0	0000000000001000
1	0	0	1	0000000000001001
1	0	1	0	0000000000001010
1	0	1	1	0000000000001011
1	1	0	0	0000000000001100
1	1	0	1	0000000000001101
1	1	1	0	0000000000001110
1	1	1	1	0000000000001111
0	0	0	0	0000000000000000