a E = 2^S

viernes, 16 de mayo de 2025 13:04

5. Explique qué es y cómo funcionan un multiplexor y un demultiplexor.¿Para qué sirve? De ejemplos de aplicación.
   a. Diseñe, simule e impacte en la FPGA un circuito multiplexor de 4 entradas;

Si tiene 4 entradas → tiene 2 selectores

$E = 2^S$

$4 = 2^S$

$\log_2(4) = S$

$S = 2$

⇒

| $S_0$ | $S_1$ | f |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

$f = \bar{S}_0 \bar{S}_1 I_0 + \bar{S}_0 S_1 I_1 + S_0 \bar{S}_1 I_2 + S_0 S_1 I_3$

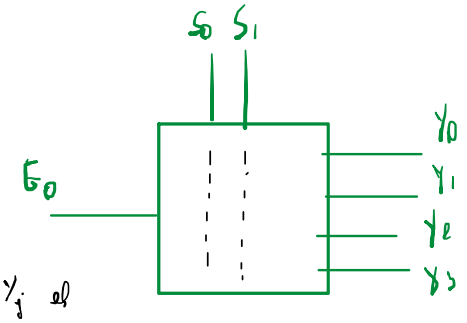
en Verilog ⇒ $f = (\bar{S}_0 \& \bar{S}_1 \& I_0) \mid (\bar{S}_0 \& S_1 \& I_1) \mid (S_0 \& \bar{S}_1 \& I_2) \mid (S_0 \& S_1 \& I_3)$

# b

viernes, 16 de mayo de 2025 14:27

b. Diseñe, simule e impacte en la FPGA un circuito demultiplexor de 4 salidas;

**1 entrada 4 salidas, selectores 2**




Para que salga por el canal $Y_j$ el valor debe ser $S_0 S_1$ en binario:

Ej: si quiero que salga por $Y_1$

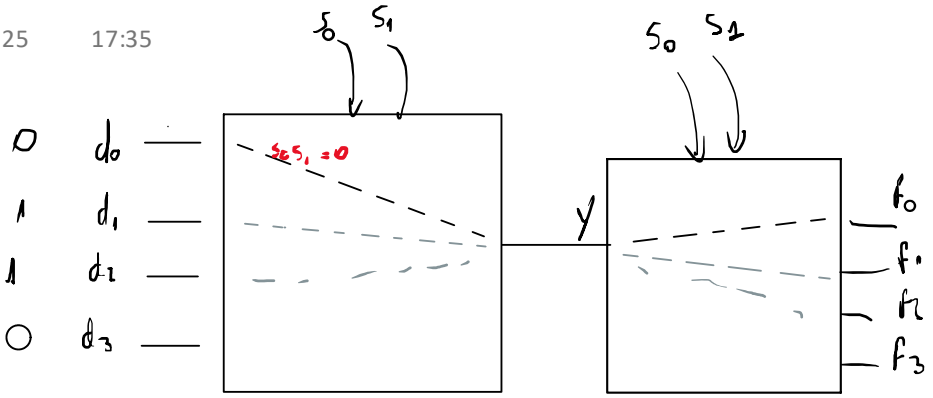
$S_0 = 0$ $S_1 = 1 \rightarrow Y_1 = 1$

| $S_0$ | $S_1$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

$f = \bar{S_0}\bar{S_1}Y_0 + \bar{S_0}S_1Y_1 + S_0\bar{S_1}Y_2 + S_0S_1Y_3$

d

viernes, 16 de mayo de 2025 17:35



# 6

viernes, 16 de mayo de 2025 19:51

6. Diseñe e implemente en Verilog un circuito que genere los bits de paridad de Hamming para un dato de 8 bits. Realice lo mismo para el circuito corrector correspondiente. Analice y compruebe el correcto funcionamiento de cada circuito mediante testbench.

***Ayuda***: Piense en cada una de las funciones (generador y corrector) como dos circuitos por separado.

Si $n=8 \longrightarrow 8 + p \leq 2^p - 1$

$p = 4$

$\Rightarrow 2^3\ 2^2\ 2^1\ 2^0$

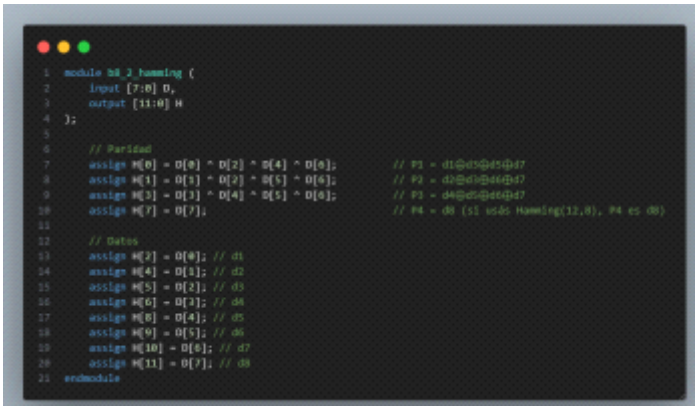
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_1$ | $p_2$ | $d_1$ | $p_3$ | $d_2$ | $d_3$ | $d_4$ | $p_4$ | $d_5$ | $d_6$ | $d_7$ | $d_8$ |

$P_1 =$ 3, 5, 7, 9 → $p_1 = d_1 \oplus d_3 \oplus d_7 \oplus d_5$

$P_2 =$ 3, 6, 7, 10 → $p_2 = d_1 \oplus d_3 \oplus d_4 \oplus d_6$

$P_3 =$ 5, 6, 7 → $p_3 = d_2 \oplus d_3 \oplus d_4$

$P_4 =$ 9, 10 → $p_4 = d_5 \oplus d_6$



```
module h8_2_hamming (
    input [7:0] D,
    output [11:0] H
);

    // Paridad
    assign H[0] = D[0] ^ D[2] ^ D[4] ^ D[6];        // P1 = d1⊕d3⊕d5⊕d7
    assign H[1] = D[1] ^ D[2] ^ D[5] ^ D[6];        // P2 = d2⊕d3⊕d6⊕d7
    assign H[3] = D[1] ^ D[4] ^ D[5] ^ D[6];        // P3 = d4⊕d5⊕d6⊕d7
    assign H[7] = D[7];                             // P4 = d8 (si usás Hamming(12,8), P4 es d8)

    // Datos
    assign H[2] = D[0]; // d1
    assign H[4] = D[1]; // d2
    assign H[5] = D[2]; // d3
    assign H[6] = D[3]; // d4
    assign H[8] = D[4]; // d5
    assign H[9] = D[5]; // d6
    assign H[10] = D[6]; // d7
    assign H[11] = D[7]; // d8
endmodule
```

---

Como funciona el corrector:

| | $d_0$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ | $d_7$ | $d_8$ | $d_9$ | $d_{10}$ | $d_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | | | | | | | | | | | | |
| $P_2$ | | | | | | | | | | | | |
| $P_3$ | | | | | | | | | | | | |
| $P_4$ | | | | | | | | | | | | |

chequeo cada bit de paridad aplicado ⊕ entre todos los bits $d_i$

$P_4 = d_2 \oplus d_8 + d_9 + d_{10} + d_{11} = D_4$

$P_3 = d_3 \oplus d_7 \oplus d_5 + d_6 = D_3$

$P_2 = d_2 \oplus d_5 \oplus d_6 + d_9 + d_{10} + d_{11} = D_2$

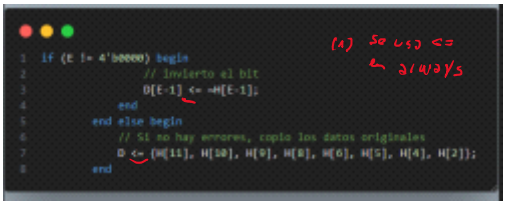
$P_1 = d_1 \oplus d_5 \oplus d_6 \oplus d_7 = D_1$

} en esa pos está el error (siempre que $D \neq 0000$)

En código

```
// Calculo los bits de paridad
wire P1, P2, P3, P4;
assign P1 = H[0] ^ H[2] ^ H[4] ^ H[6] ^ H[8] ^ H[10];
assign P2 = H[1] ^ H[2] ^ H[5] ^ H[6] ^ H[9] ^ H[10];
assign P3 = H[3] ^ H[4] ^ H[5] ^ H[6] ^ H[11];
assign P4 = H[7] ^ H[8] ^ H[9] ^ H[10] ^ H[11];
```

Si $D \neq 0000 \Rightarrow$ en la pos donde está el error lo invierto



```
if (E != 4'b0000) begin
            // Invierto el bit
            D[E-1] <= ~H[E-1];
        end
    end else begin
        // Si no hay errores, copio los datos originales
        D <= {H[11], H[10], H[9], H[8], H[6], H[5], H[4], H[2]};
    end
```

(1) Se usa <= en always