Implementar memorias en VHDL

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2025/01/22/implementar-memorias-en-vhdl/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 13/02/2025

Para implementar memorias en VHDL lo único que necesitamos es hacer uso de la palabra reservada *array.*

Esta palabra permite crear matrices de datos escalables. Por ejemplo, si queremos crear una matriz de datos de 100 x 12, solo necesitamos crear una matriz de 100 datos *std_logic_vector* de 12 bits.

```
type qtyp is array (0 to 99) of std_logic_vector(11 downto 0);
```

Entonces, la definición anterior nos crea un *type*, que es lo que luego nos permite crear una señal o una constante que contenga los datos que queremos almacenar como una memoria.

Por ejemplo si queremos almacenar 100 datos de 12 bits cada uno, podemos hacerlo así.

```
signal T_s : q_i_type;

constant T : q_i_type := (
                    "000000000000", "000000010000", "000001000000",
"000010001111",
                    "000011111101", "000110000111", "001000101010",
"001011100110",
                    "001110110110", "010010010111", "010110000110",
"011001111111",
                    "011101111110", "100010000000", "100101111111",
"101001111000",
                    "101101100111", "110001001000", "110100011000",
"110111010100",
                    "111001110111", "111100000001", "111101101111",
"111110111110",
                    "111111101110", "111111111111", "111111101110",
"111110111110",
                    "111101101111", "111100000001", "111001110111",
"110111010100",
                    "110100011000", "110001001000", "101101100111",
"101001111000",
                    "100101111111", "100010000000", "011101111110",
"011001111111",
                    "010110000110", "010010010111", "001110110110",
"001011100110",
                    "001000101010", "000110000111", "000011111101",
"000010001111",
                    "000001000000", "000000010000", "000000000000",
"000000010000",
                    "000001000000", "000010001111", "000011111101",
"000110000111",
                    "001000101010", "001011100110", "001110110110",
"010010010111",
                    "010110000110", "011001111111", "011101111110",
"100010000000",
                    "100101111111", "101001111000", "101101100111",
"110001001000",
                    "110100011000", "110111010100", "111001110111",
"111100000001",
                    "111101101111", "111110111110", "111111101110",
"111111111111",
```

```
                        "111111101110", "111110111110", "111101101111",
"111100000001",
                        "111001110111", "110111010100", "110100011000",
"110001001000",
                        "101101100111", "101001111000", "100101111111",
"100010000000",
                        "011101111110", "011001111111", "010110000110",
"010010010111",
                        "001110110110", "001011100110", "001000101010",
"000110000111",
                        "000011111101", "000010001111", "000001000000",
"000000010000"
                );
```

Esto se puede aplicar en FIFO, memorias ROM, RAM, etc.

Nota final

Esta misma estructura se puede utilizar para trabajar con buses, haciendo que cada bus provenga de un sitio distinto y unificando el manejo de señales desde un solo sitio.