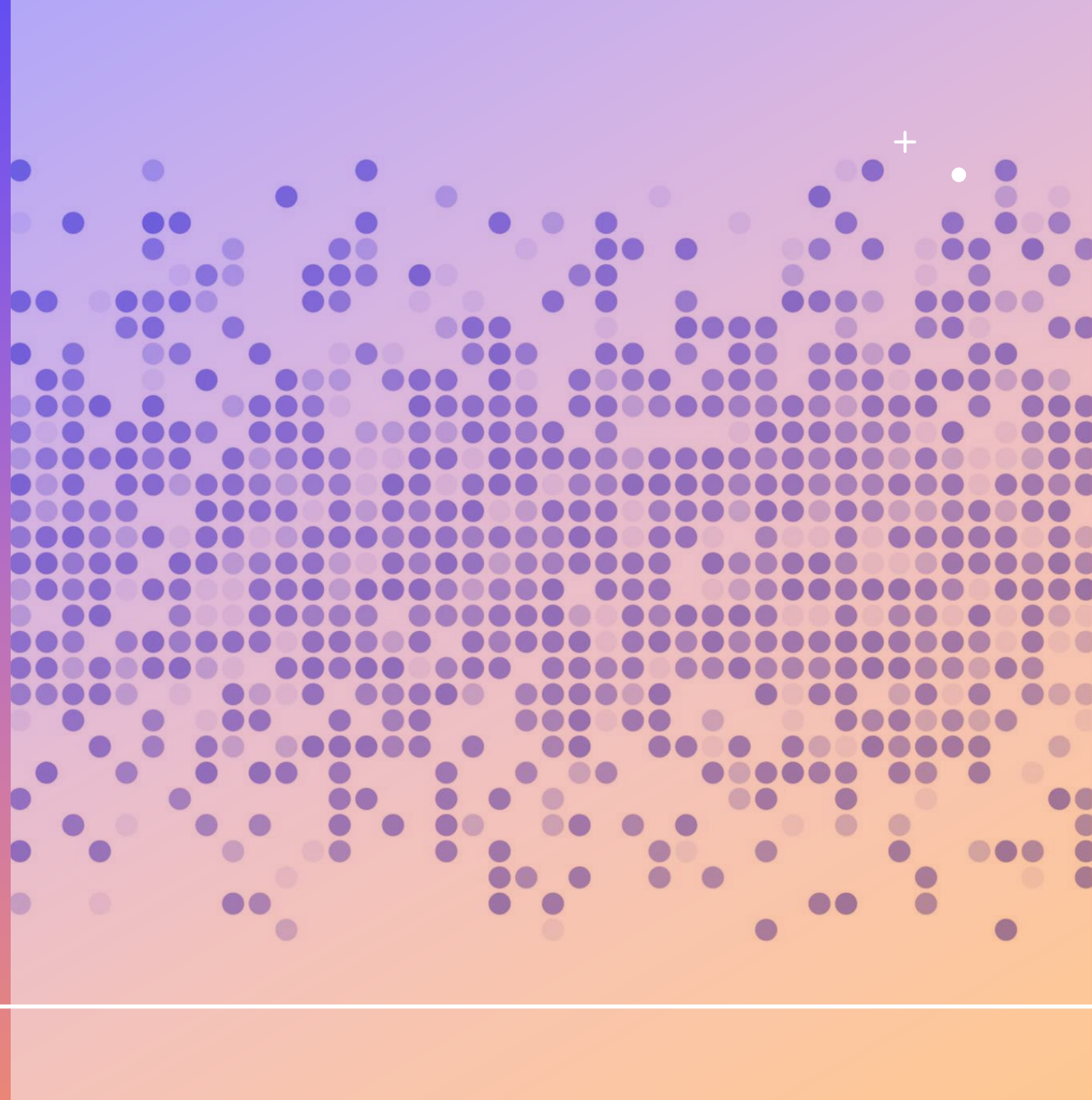


FUNDAMENTAL VHDL UNITS

Dr. Fatma Elfouly



WHAT IS VHDL?

VHSIC

Hardware
Description
Language

VHSIC – Very High Speed Integrated Circuit



+

o

DIFFERENCE BETWEEN HARDWARE DESCRIPTION LANGUAGE AND SOFTWARE LANGUAGE

Hardware description language :

It describes hardware in textual form.

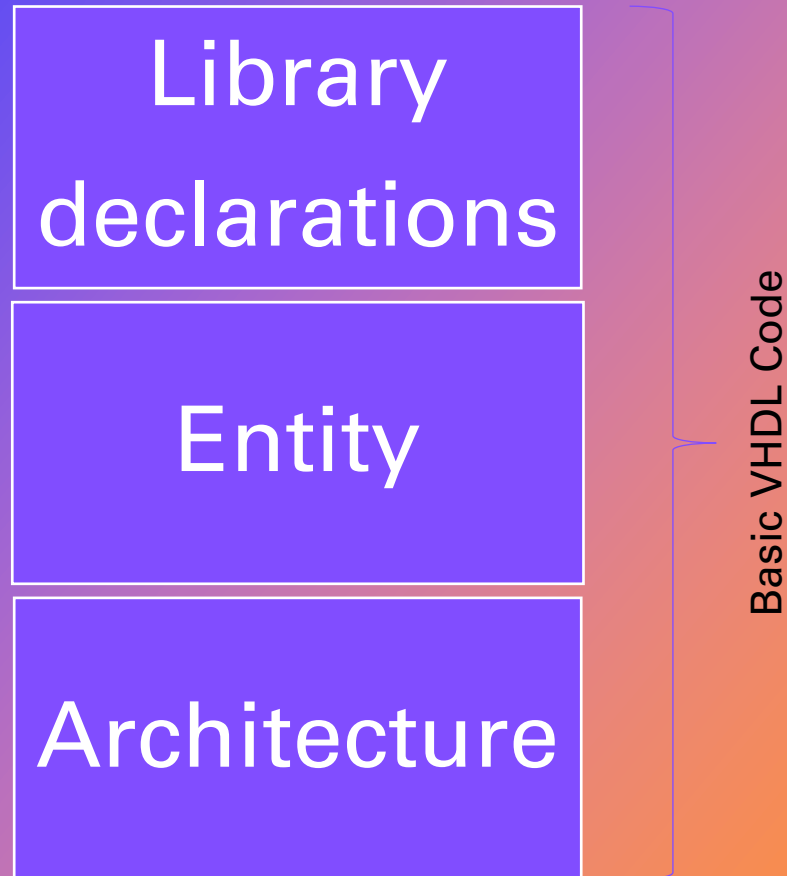
It describes hardware behavior and their structure.

Software language :

It is a [programming language](#) that allow a software designer to executable software applications that will operate on a suitable processor.



BASIC VHDL CODE



❖ **LIBRARY declarations:** Contains a list of all libraries to be used in the design.

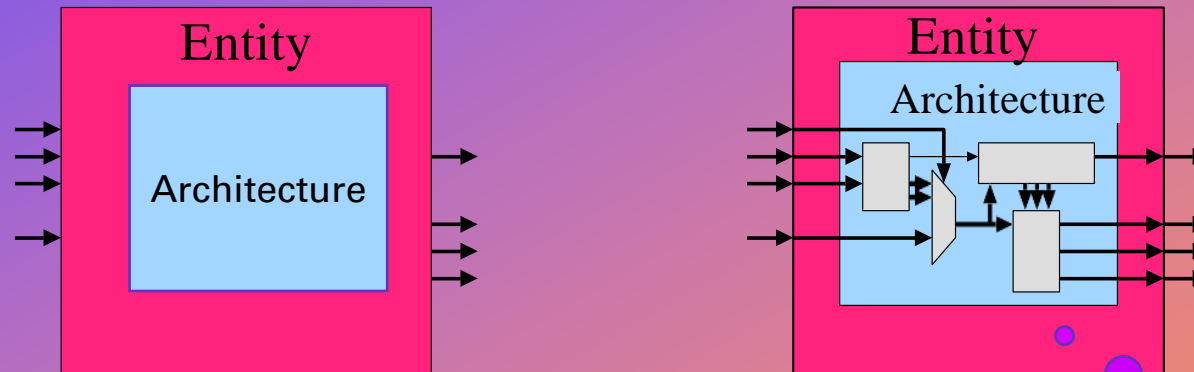
For example: `ieee`, `std`, `work`, etc.

❖ **ENTITY:** Specifies the I/O pins of the circuit.

❖ **ARCHITECTURE:** Contains the VHDL code proper, which describes how the

❖ circuit should behave (function).

ENTITIES AND ARCHITECTURE



Entity: interface
– names, modes (**in / out**),
types of externally visible signals of circuit

Architecture: **internals**
– **structure and behaviour of module**

ENTITY

Signal mode: is one of the reserved words to indicate the signal direction:

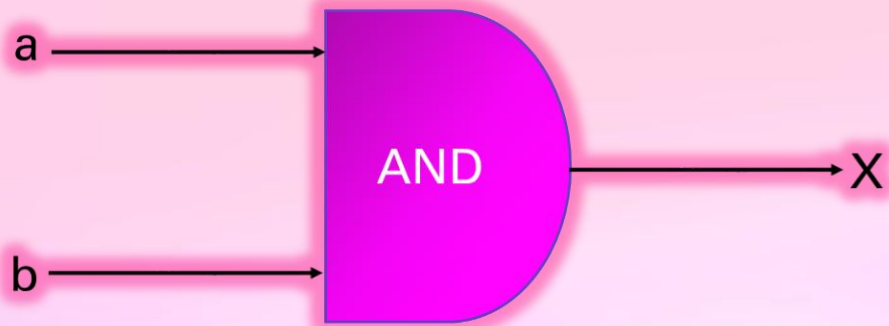
- **in** – indicates that the signal is an input
- **out** – indicates that the signal is an output of the entity whose value can only be read by other entities that use it.
- **buffer** – indicates that the signal is an output of the entity whose value can be read inside the entity's architecture
- **inout** – the signal can be an input or an output.

Signal_type:

a built-in or user-defined signal type. Examples of types are bit, **bit_vector**, Boolean, **character**, **std_logic**, etc.

```
ENTITY entity_name IS
  PORT (
    port_name : signal_mode signal_type;
    port_name : signal_mode signal_type;
    ... );
END entity_name;
```

ENTITY



STD_LOGIC is defined in the library **std_logic_1164**. This is a nine valued logic system. It has 9 values: 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H' and '-'.
- = dont care

BIT has 2 values: '0' and '1'.

```
entity and_gate is  
  port (a, b: in bit;  
        x: out bit);  
end entity and_gate;
```

```
entity and_gate is  
  port (a, b: in std_logic;  
        x: out std_logic);  
end entity and_gate;
```

U = uninitialized
X = unknown
0 = logic 0
1 = logic 1
Z = high impedance (tri state)
W = weak unknown
L = weak "0"
H = weak "1"
- = dont care

ARCHITECTURE

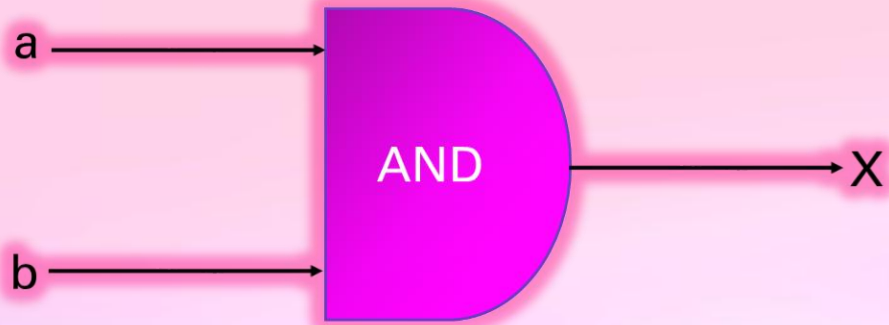
- The architecture body specifies how the circuit operates and how it is implemented.
- Architecture is described using behavioral code or structural code

```
ARCHITECTURE architecture_name OF entity_name IS  
    [declarations]  
BEGIN  
    [code]  
END architecture_name;
```


STRUCTURAL AND BEHAVIORAL

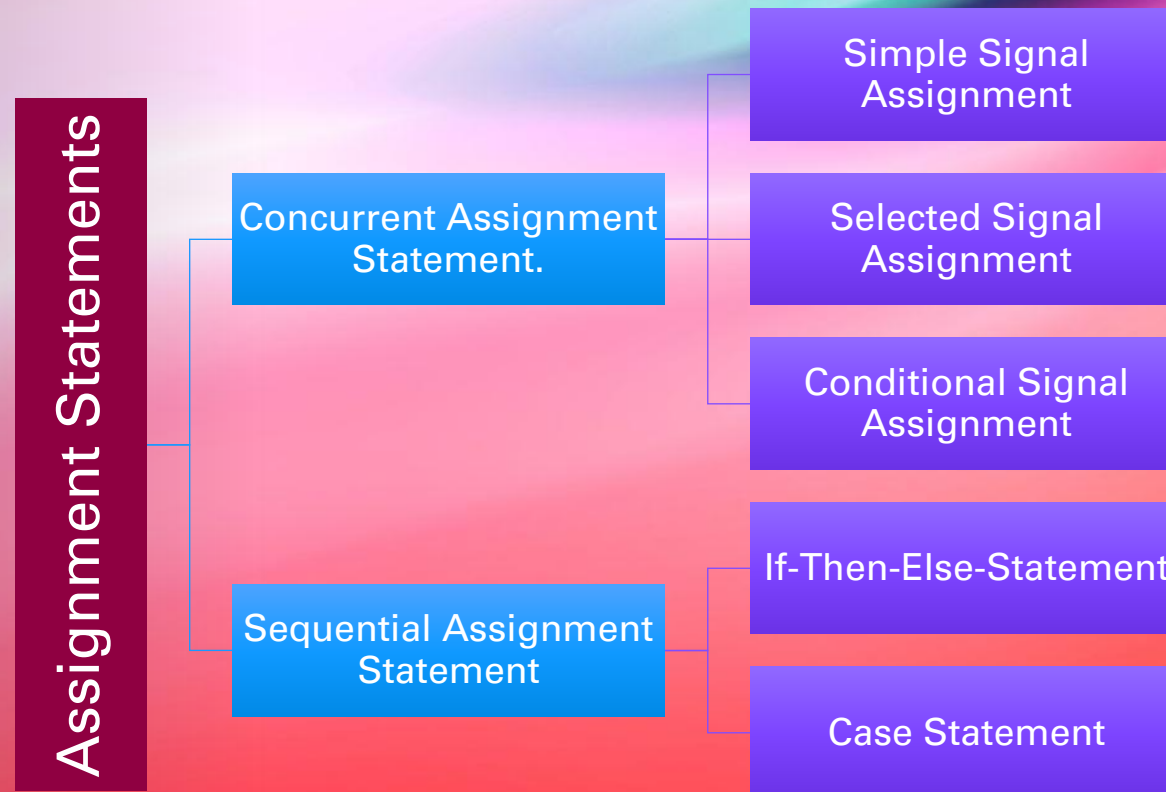
- ❖ An entity or circuit can be specified in a variety of ways, such as **behavioral**, **structural** (interconnected components), or a **combination of the above**.
- ❖ The **behavioral** level that describes a system in terms of what it does (or how it behaves). A behavioral description specifies the relationship between the input and output signals.
- ❖ The **structural** level, on the other hand, describes a system as a collection of gates and components that are interconnected to perform a desired function.

ARCHITECTURE



```
ARCHITECTURE RTL OF and_gate IS  
BEGIN  
  X <= a AND b;  
END and_gate;
```

ASSIGNMENT STATEMENTS



CONCURRENT ASSIGNMENT STATEMENT.



SIMPLE SIGNAL ASSIGNMENT

❖ It used for a logic or arithmetic expression.

The general form is:-

```
Signal_name <= expression;
```

Example:

```
y <= a AND b;
```

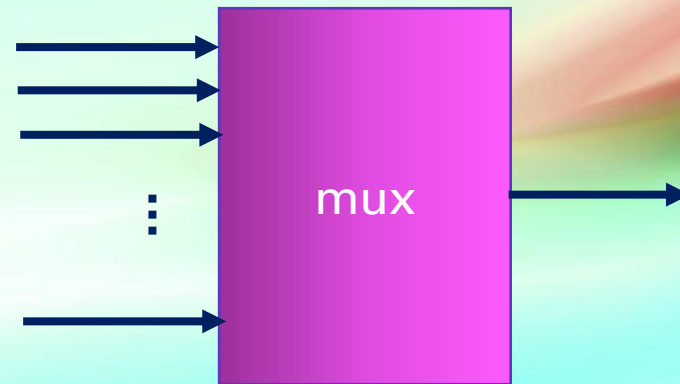
SELECTED SIGNAL ASSIGNMENT

- It used to assign one of several values based on selection criterion used with keyword.

The general form is:-

With expression **select**

Signal_name <= expression **when** constant_value;



MULTIPLEXER

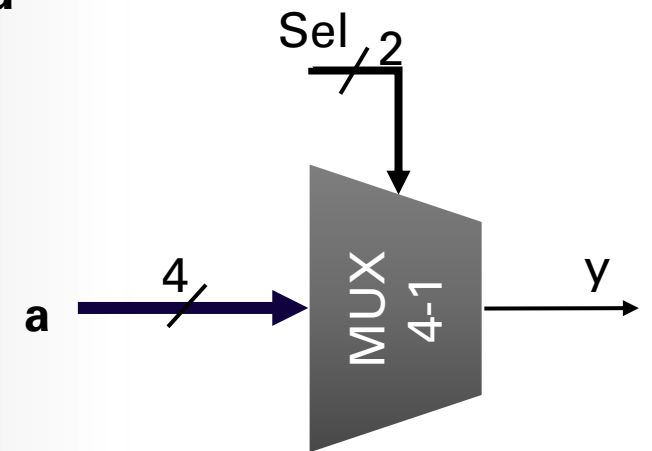
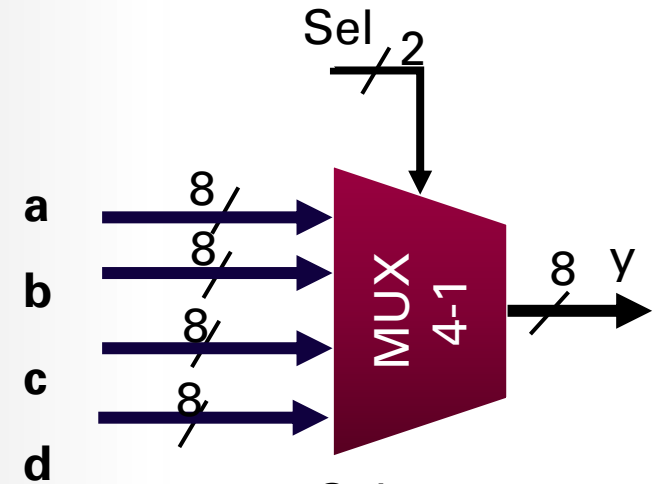
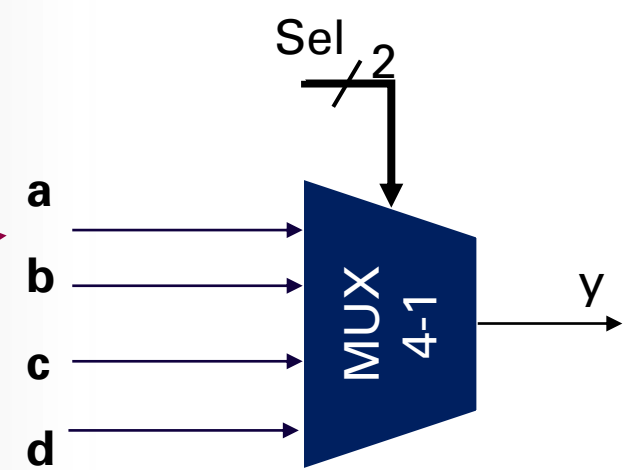
Multiplexer is a device that selects between several input signals and forwards it to a single output line.

MULTIPLEXER

4-1

Sel		y
0	0	a
0	1	b
1	0	c
1	1	d

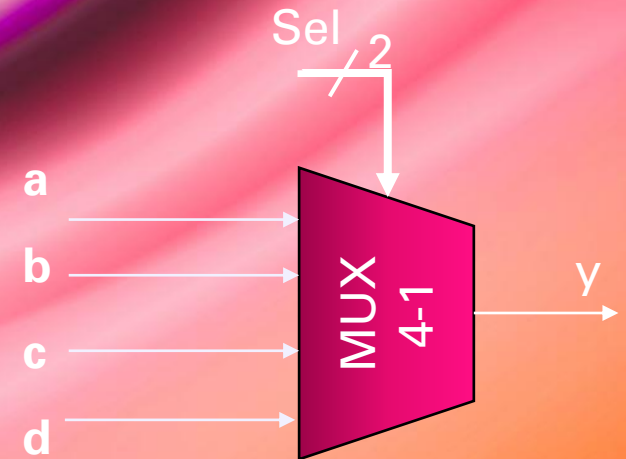
Sel		y
0	0	a(0)
0	1	a(1)
1	0	a(2)
1	1	a(3)



SELECTED SIGNAL ASSIGNMENT

► Multiplexer

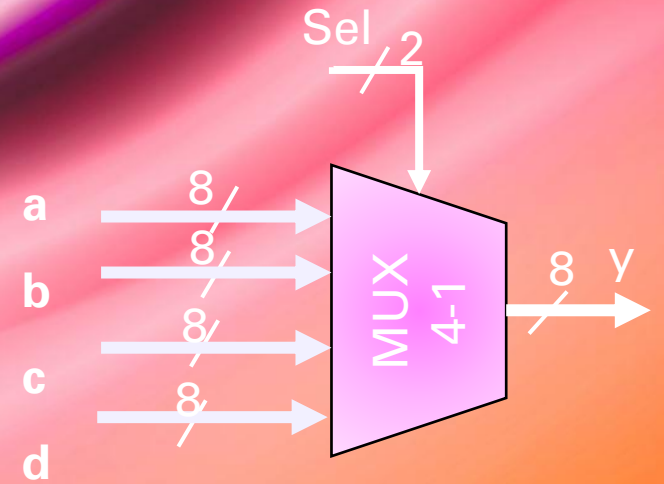
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
with sel select
y <= a when "00",
b when "01",
c when "10",
d when "11",
a when others;
```



SELECTED SIGNAL ASSIGNMENT

► Multiplexer

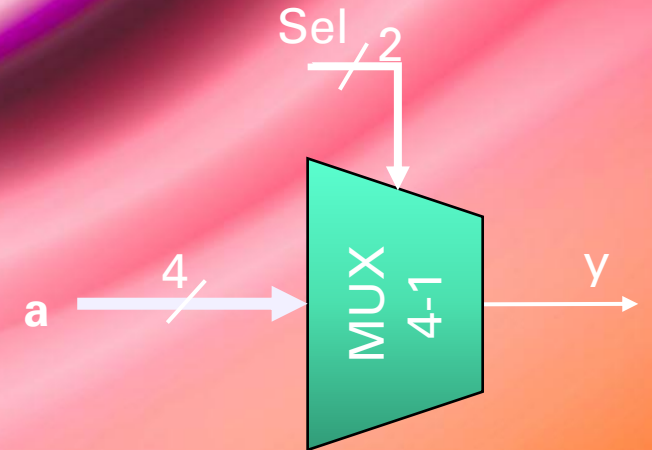
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic_vector(7 downto 0);
y: out std_logic_vector(7 downto 0));
end entity mux4_1;
architecture rtl of mux4_1 is
begin
with sel select
y <= a when "00",
b when "01",
c when "10",
d when "11",
a when others;
```



SELECTED SIGNAL ASSIGNMENT

► Multiplexer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a: in std_logic_vector(3 downto 0);
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
with sel select
y <= a(0) when "00",
a(1) when "01",
a(2) when "10",
a(3) when others;
```



CONDITIONAL SIGNAL ASSIGNMENT

It used to set a signal to one of several values.

The general form is:-

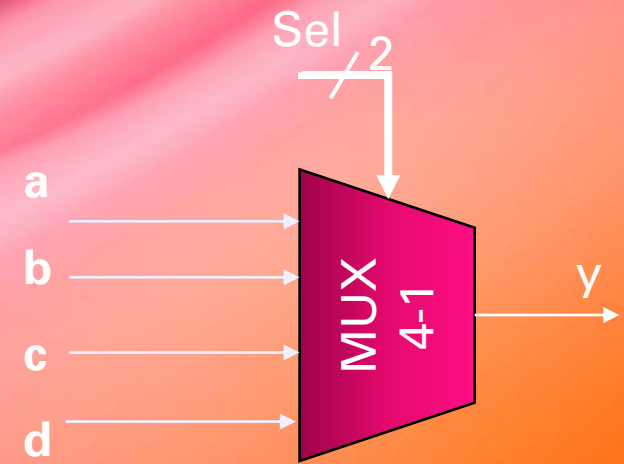
Signal_name \leftarrow expression when logic_expression else expression ;



CONDITIONAL SIGNAL ASSIGNMENT

➤ Multiplexer

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
entity mux4_1 is  
port (sel: in std_logic_vector (1 downto 0);  
a, b, c, d: in std_logic;  
y: out std_logic);  
end entity mux4_1;  
architecture rtl of mux4_1 is  
begin  
y <= a when sel = "00" else  
b when sel = "01" else  
c when sel = "10" else  
d;  
end architecture rtl;
```



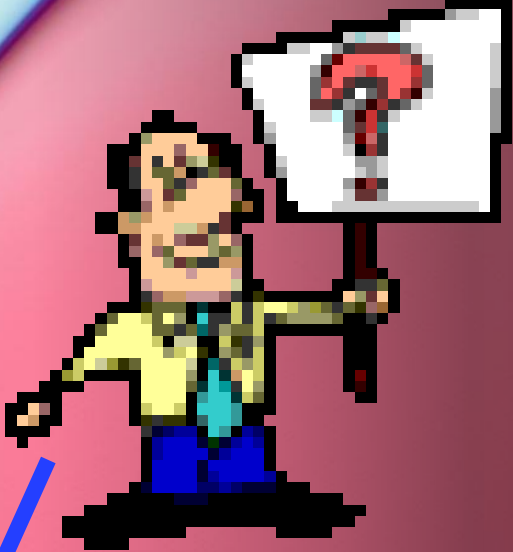
SEQUENTIAL ASSIGNMENT STATEMENT

- ❖ **Case Statement**

- ❖ **If-Then-Else-Statement**

A process statement is the main construct that allows you to use sequential statements to describe the behavior of a system over time.

The syntax for a process statement is:

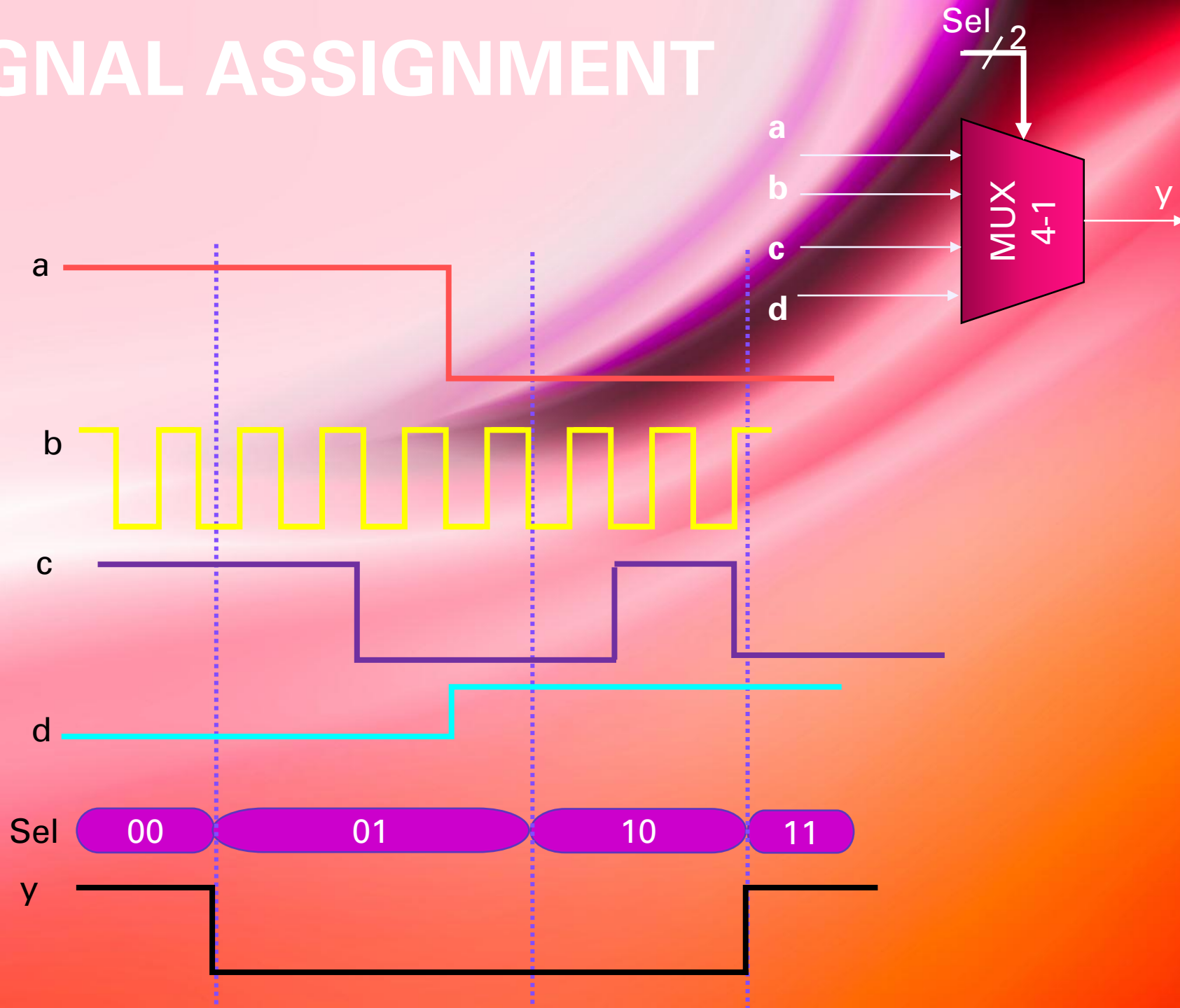


```
[process_label:] process (sensitivity_list)  
Variable declarations  
Begin  
[if-then-else-statement]  
[case-statement]  
End process;
```

SEQUENTIAL SIGNAL ASSIGNMENT

► Multiplexer

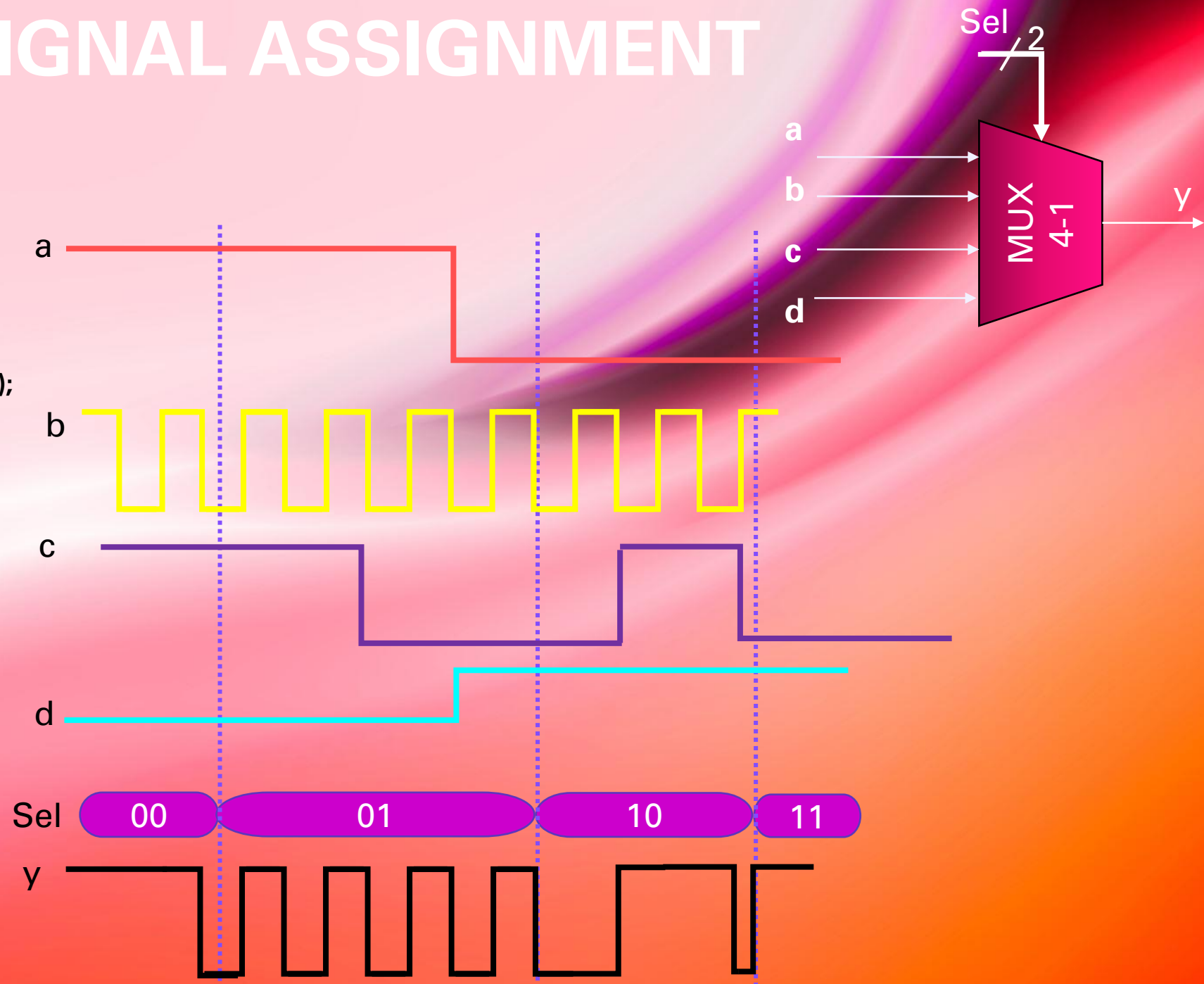
```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
entity mux4_1 is  
  port (sel: in std_logic_vector (1 downto 0);  
        a, b, c, d: in std_logic;  
        y: out std_logic);  
end entity mux4_1;  
architecture rtl of mux4_1 is  
  begin  
    process (sel)  
    begin  
      if (sel = "00") then  
        y <= a;  
      elsif (sel = "01") then  
        y <= b;  
      elsif (sel = "10") then  
        y <= c;  
      else  
        y <= d;  
      end if;  
    end process;  
  end architecture rtl;
```



SEQUENTIAL SIGNAL ASSIGNMENT

➤ Multiplexer

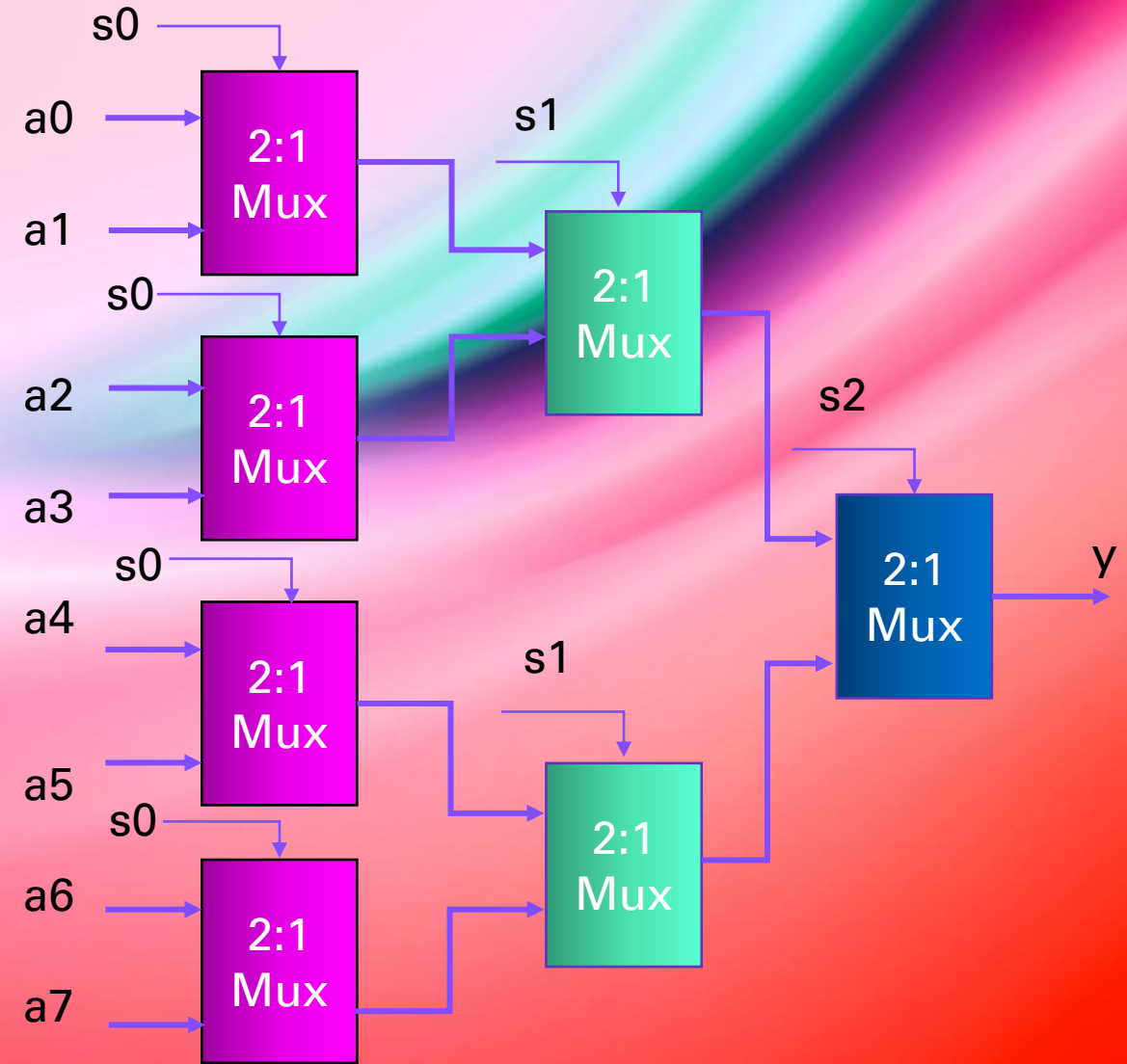
```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_arith.all;  
entity mux4_1 is  
  port (sel: in std_logic_vector (1 downto 0);  
        a, b, c, d: in std_logic;  
        y: out std_logic);  
end entity mux4_1;  
architecture rtl of mux4_1 is  
  begin  
    process (sel, a, b, c, d)  
    begin  
      if (sel = "00") then  
        y <= a;  
      elsif (sel = "01") then  
        y <= b;  
      elsif (sel = "10") then  
        y <= c;  
      else  
        y <= d;  
      end if;  
    end process;  
  end architecture rtl;
```



SOLVED PROBLEMS

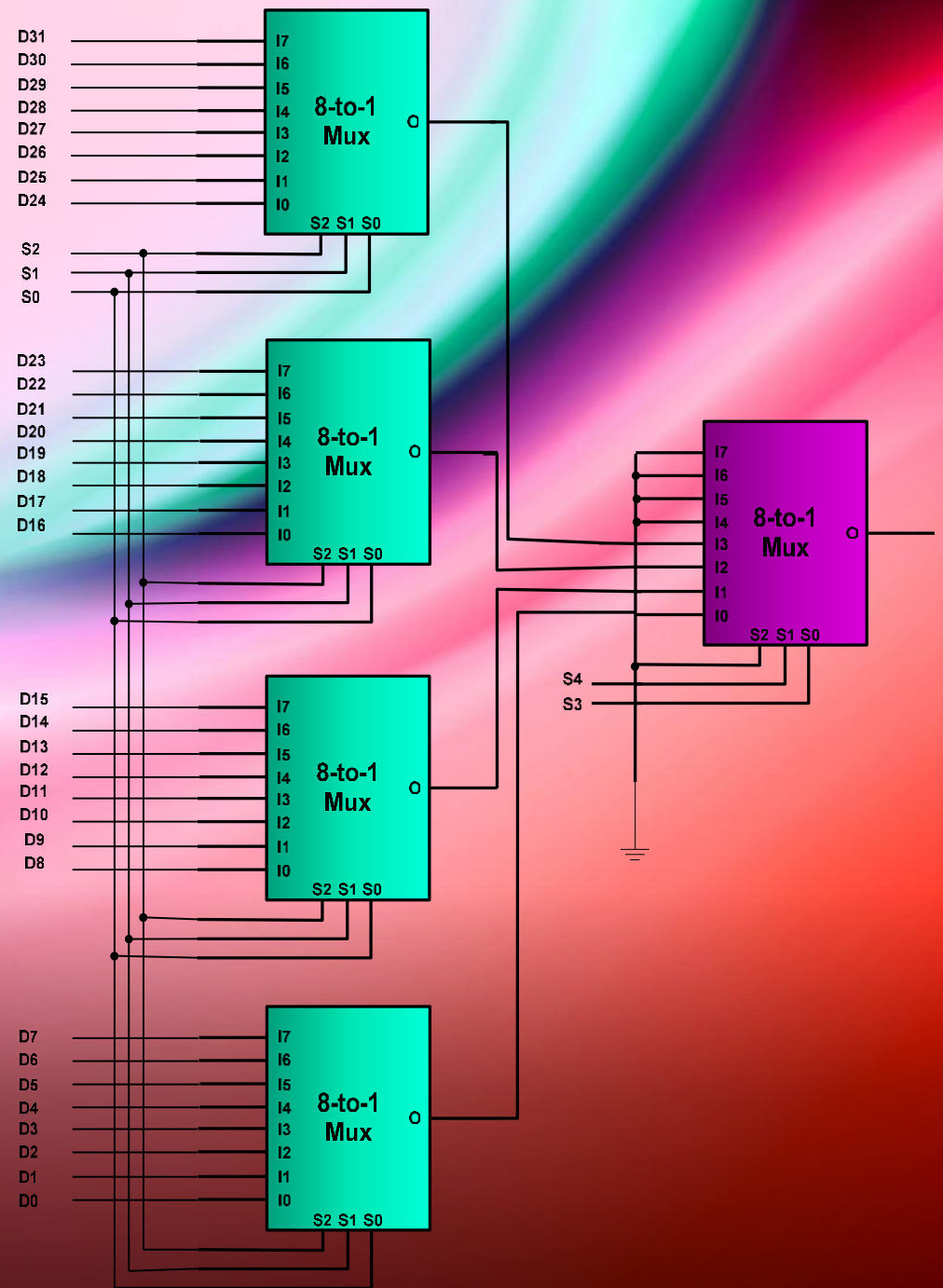
Design 8-to-1 multiplexer using only 2-to-1 multiplexer

s			y
0	0	0	a0
0	0	1	a1
0	1	0	a2
0	1	1	a3
1	0	0	a4
1	0	1	a5
1	1	0	a6
1	1	1	a7



SOLVED PROBLEMS

Design a 32-to-1 multiplexer using only 8-to-1 multiplexer



SOLVED PROBLEMS

Design a 8-to-1 multiplexer using
only 2-to-1 multiplexer

