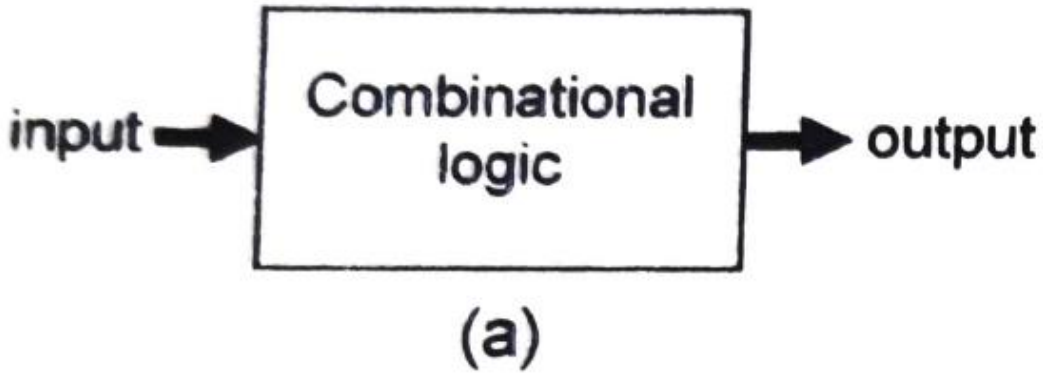
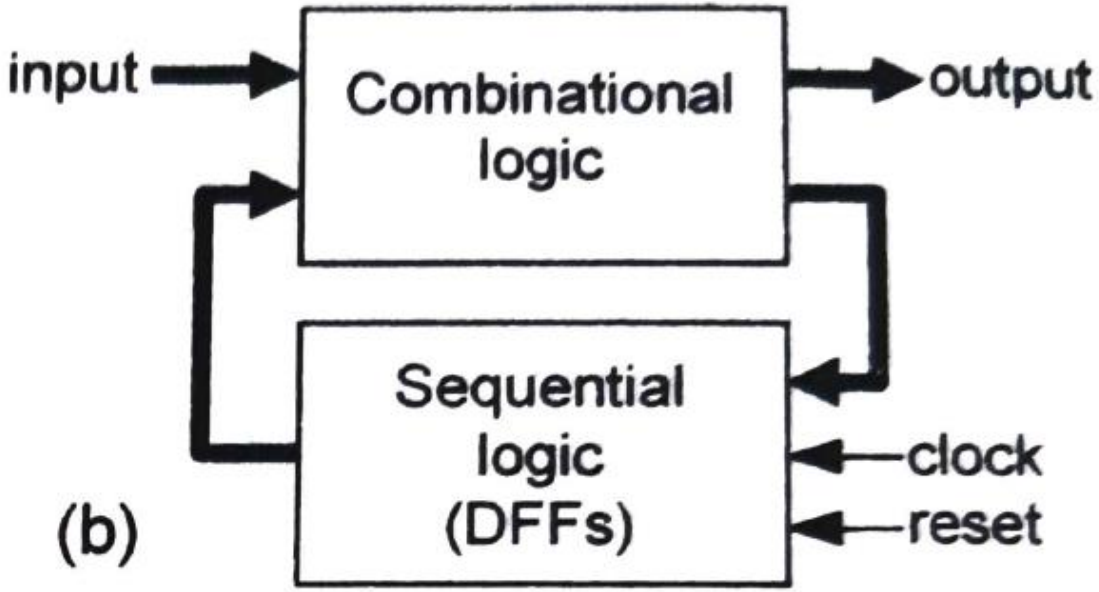
Concurrent and sequential statements
(designing combinational circuits)





While concurrent code is intended only for the design of combinational circuits, sequential code can be used indistinctly to design both sequential and combinational circuits.

~~Concurrent~~

Today's agenda

- Statements for concurrent Code
  - ~~When Select Statement~~
  - When-else * Statement
  - With-Select-When Statement
- Statements for Sequential Code
  - If-else Statement
  - Case Statement

When-else statement

```
assignment_expression WHEN conditions ELSE
     assignment_value WHEN conditions ELSE
     ...;
```

When-else statement

- Syntax

```
output_signal <= expa when cond1 else
                 expb when cond2 else
                 ........
                 expx when condx else
                 expy;
```

output_signal: output(s),
condi: condition in terms of inputs
expi: expression in terms of inputs

With-select-when statement

```
WITH identifier SELECT
   assignment_expression WHEN values,
        assignment_value WHEN values,
        ...;
```

Syntax → With-select-when statement

```
with sel select
output_signal <= expra when chioces,
                 exprb when choices,
                 ........
                 exprx when others;
```

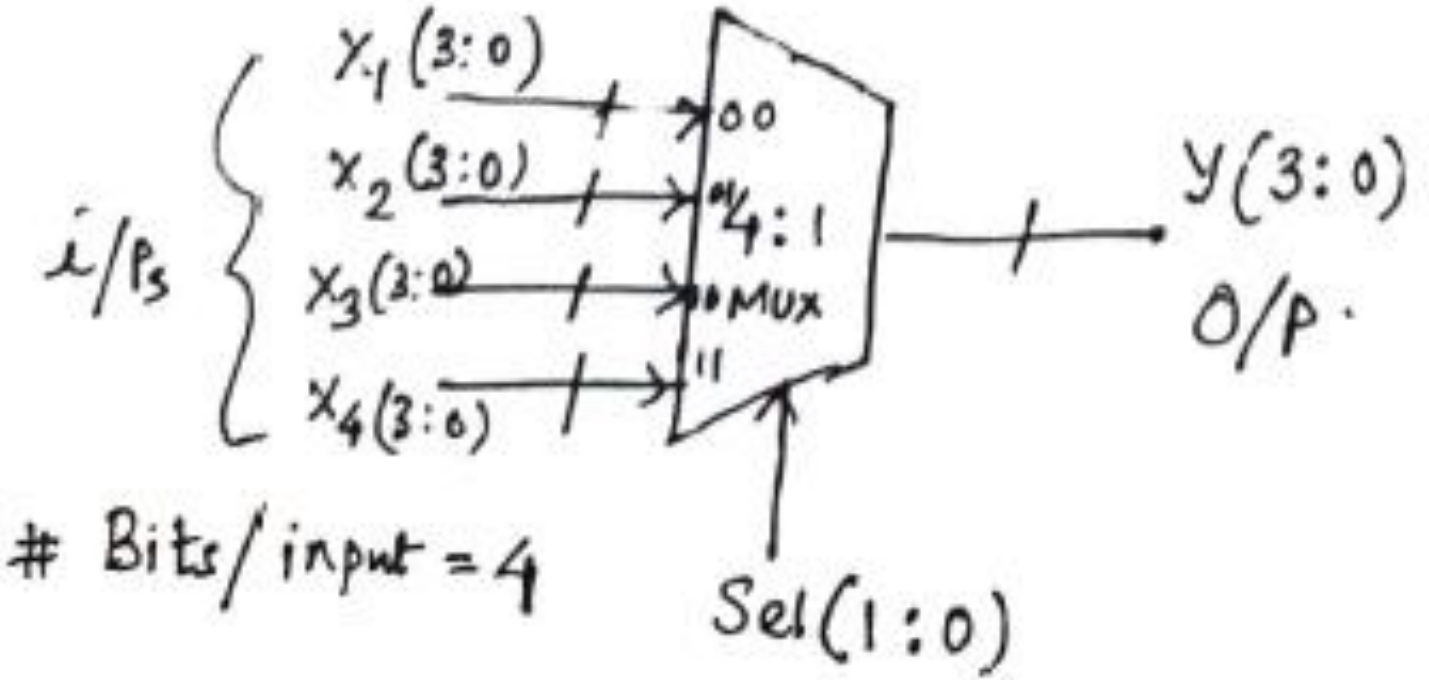
Sequential statement: If-elsif-else

```
[label:] IF conditions THEN
   assignments;
ELSIF conditions THEN
   assignments;
...
ELSE
   assignments;
END IF [label];
```

Sequential : Case statements

```
[label:] CASE expression IS
    WHEN value => assignments;
    WHEN value => assignments;
    ...
END CASE;
```

Case study: 4:1 MUX



Code examples

Concurrent code → when-else

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity mux1 is
24      Port ( x1 : in  STD_LOGIC_VECTOR (3 downto 0);
25             x2 : in  STD_LOGIC_VECTOR (3 downto 0);
26             x3 : in  STD_LOGIC_VECTOR (3 downto 0);
27             x4 : in  STD_LOGIC_VECTOR (3 downto 0);
28             sel : in  STD_LOGIC_VECTOR (1 downto 0);
29             y : out  STD_LOGIC_VECTOR (3 downto 0));
30  end mux1;
31
32
33  architecture concurrent1 of mux1 is
34
35  begin
36  y <= x1 when sel="00" else
37       x2 when sel="01" else
38       x3 when sel="10" else
39       x4 when sel="11" else "XXXX";
40  end concurrent1;
```

Concurrent code → with-select-when

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  entity mux2 is
24      Port ( x1 : in  STD_LOGIC_VECTOR (3 downto 0);
25             x2 : in  STD_LOGIC_VECTOR (3 downto 0);
26             x3 : in  STD_LOGIC_VECTOR (3 downto 0);
27             x4 : in  STD_LOGIC_VECTOR (3 downto 0);
28             sel : in  STD_LOGIC_VECTOR (1 downto 0);
29             y : out  STD_LOGIC_VECTOR (3 downto 0));
30  end mux2;
31
32  architecture concurrent2 of mux2 is
33  begin
34  with sel select
35  y <= x1 when "00",
36       x2 when "01",
37       x3 when "10",
38       x4 when "11",
39       "XXXX" when others;
40  end concurrent2;
```

Sequential code → if-else statements

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  entity mux3 is
23      Port ( x1 : in  STD_LOGIC_VECTOR (3 downto 0);
24             x2 : in  STD_LOGIC_VECTOR (3 downto 0);
25             x3 : in  STD_LOGIC_VECTOR (3 downto 0);
26             x4 : in  STD_LOGIC_VECTOR (3 downto 0);
27             sel : in  STD_LOGIC_VECTOR (1 downto 0);
28             y : out  STD_LOGIC_VECTOR (3 downto 0));
29  end mux3;
30
31  architecture sequential1 of mux3 is
32  begin
33  process(x1,x2,x3,x4,sel)
34  begin
35  if sel="00" then
36     y<=x1;
37  elsif sel="01" then
38     y<=x2;
39  elsif sel="10" then
40     y<=x3;
41  elsif sel="11" then
42     y<=x4;
43  else
44     y<="XXXX";
45  end if;
46  end process;
47  end sequential1;
```

Sequential code → Case statement

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  entity mux4 is
23      Port ( x1 : in  STD_LOGIC_VECTOR (3 downto 0);
24             x2 : in  STD_LOGIC_VECTOR (3 downto 0);
25             x3 : in  STD_LOGIC_VECTOR (3 downto 0);
26             x4 : in  STD_LOGIC_VECTOR (3 downto 0);
27             sel : in  STD_LOGIC_VECTOR (1 downto 0);
28             y : out  STD_LOGIC_VECTOR (3 downto 0));
29  end mux4;
30
31  architecture sequential2 of mux4 is
32  begin
33  process(x1,x2,x3,x4,sel)
34  begin
35  case sel is
36     when "00" => y<=x1;
37     when "01" => y<=x2;
38     when "10" => y<=x3;
39     when "11" => y<=x4;
40     when others => y<="XXXX";
41  end case;
42  end process;
43  end sequential2;
```