جامعة نينوى
كلية هندسة الإلكترونيات

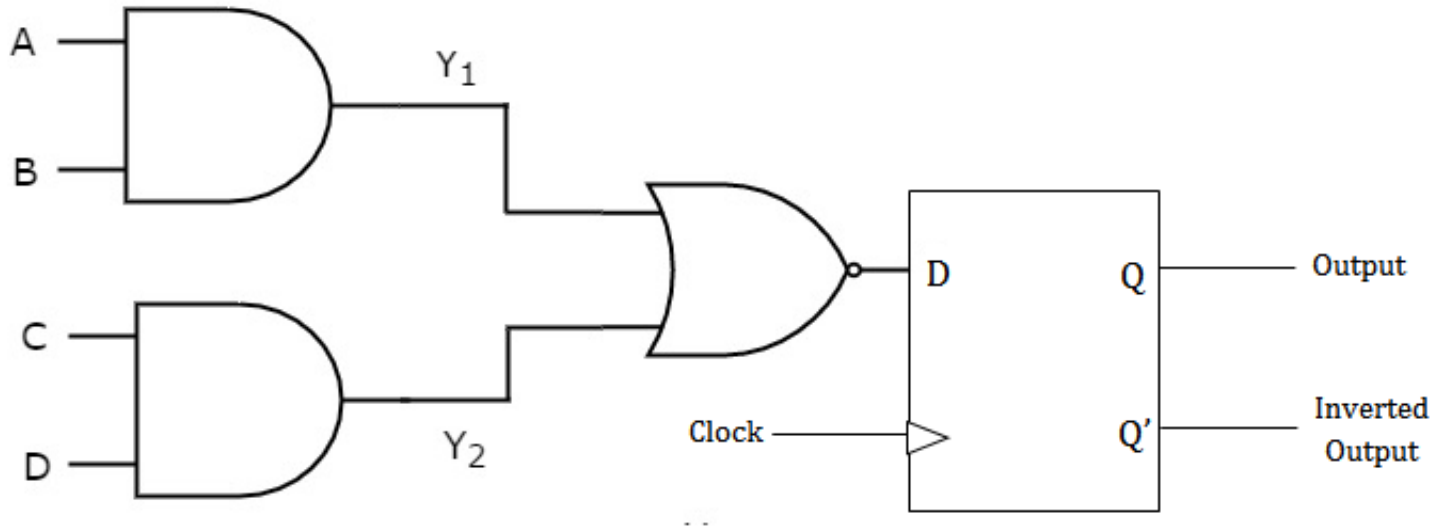
Circuit Design with VHDL 5

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

VHDL> Concurrent & sequential Code

- Example: Write VHDL code to implement the following circuit.



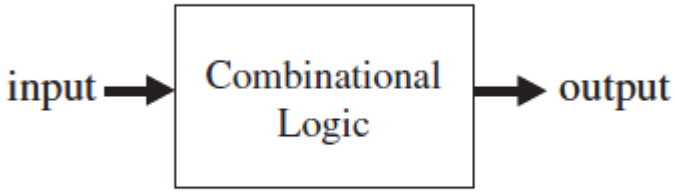
VHDL> Concurrent & sequential Code

- Concurrent Code:
  - WHEN,
  - GENERATE,
  - Assignments using only operators (AND, NOT, +, *, sll, etc.),
  - A special kind of assignment, called BLOCK.

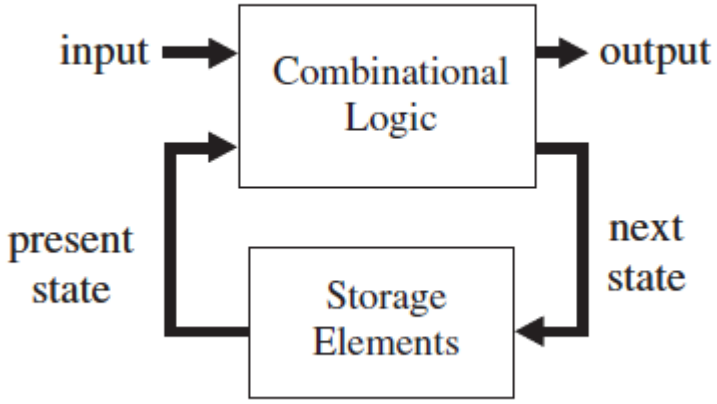
- Sequential Code:
  - PROCESSES, FUNCTIONS, PROCEDURES.
  - IF, WAIT, CASE, and LOOP.
  - VARIABLES.

VHDL>Combinational vs Sequential Logic

- **Combinational Logic:** output depends solely on the current inputs.



- **sequential logic:** output depend on previous inputs.



VHDL> Concurrent versus Sequential

- **VHDL** code is inherently concurrent (parallel).
- Only statements placed inside a PROCESS, FUNCTION, or PROCEDURE are sequential.
  - the block, as a whole, is concurrent with any other (external) statements.
- Concurrent code is also called **dataflow** code.
- Concurrent: The order does not matter.

VHDL> Concurrent Code

- In summary, in concurrent code the following can be used:
  - Operators;
  - The WHEN statement (WHEN/ELSE or WITH/SELECT/WHEN);
  - The GENERATE statement;
  - The BLOCK statement.

VHDL> Concurrent Code

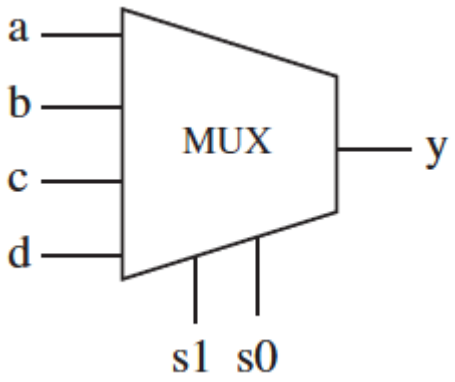
- Operators:

Table 5.1
Operators.

Operator type	Operators	Data types
Logical	NOT, AND, NAND, OR, NOR, XOR, XNOR	BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR, STD_ULOGIC, STD_ULOGIC_VECTOR
Arithmetic	+, −, *, /, ** (mod, rem, abs)	INTEGER, SIGNED, UNSIGNED
Comparison	=, /=, <, >, <=, >=	All above
Shift	sll, srl, sla, sra, rol, ror	BIT_VECTOR
Concatenation	&, (,,,)	Same as for logical operators, plus SIGNED and UNSIGNED

VHDL> Concurrent Code

- Multiplexer #1




```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY mux IS
6  PORT ( a, b, c, d, s0, s1: IN STD_LOGIC;
7  y: OUT STD_LOGIC);
8  END mux;
9  ----------------------------------------
10 ARCHITECTURE pure_logic OF mux IS
11 BEGIN
12 y <= (a AND NOT s1 AND NOT s0) OR
13 (b AND NOT s1 AND s0) OR
14 (c AND s1 AND NOT s0) OR
15 (d AND s1 AND s0);
16 END pure_logic;
17 ----------------------------------------
```

VHDL> Concurrent Code

- WHEN (Simple and Selected)

WHEN / ELSE:

```
assignment WHEN condition ELSE
assignment WHEN condition ELSE
...;
```

simple

WITH / SELECT / WHEN:

```
WITH identifier SELECT
assignment WHEN value,
assignment WHEN value,
...;
```

selected

VHDL> Concurrent Code

- Whenever WITH / SELECT / WHEN is used:
- all permutations must be tested,
  - keyword **OTHERS** is often useful.
- keyword **UNAFFECTED**,
  - which should be used when no action is to take place.

- "WHEN value" can indeed take up three forms:

```
WHEN value                    -- single value
WHEN value1 to value2         -- range, for enumerated data types
                              -- only
WHEN value1 | value2 |...     -- value1 or value2 or ...
```

VHDL> Concurrent Code

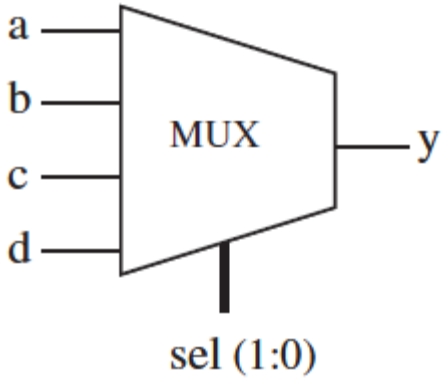
- Examples:

```
------ With WHEN/ELSE ------------------------
outp <= "000" WHEN (inp='0' OR reset='1') ELSE
        "001" WHEN ctl='1' ELSE
        "010";


---- With WITH/SELECT/WHEN --------------------
WITH control SELECT
   output <= "000" WHEN reset,
             "111" WHEN set,
             UNAFFECTED WHEN OTHERS;
```

VHDL> Concurrent Code

- Multiplexer #2: when/else

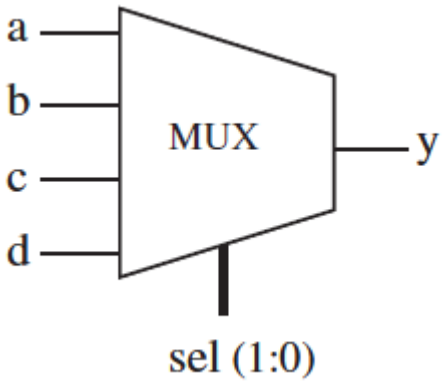



```
1  ------- Solution 1: with WHEN/ELSE --------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY mux IS
6     PORT ( a, b, c, d: IN STD_LOGIC;
7            sel: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
8            y: OUT STD_LOGIC);
9  END mux;
10 ----------------------------------------
11 ARCHITECTURE mux1 OF mux IS
12 BEGIN
13    y <=  a WHEN sel="00" ELSE
14          b WHEN sel="01" ELSE
15          c WHEN sel="10" ELSE
16          d;
17 END mux1;
```

Simple when

VHDL> Concurrent Code

- Multiplexer #2: with/select/when



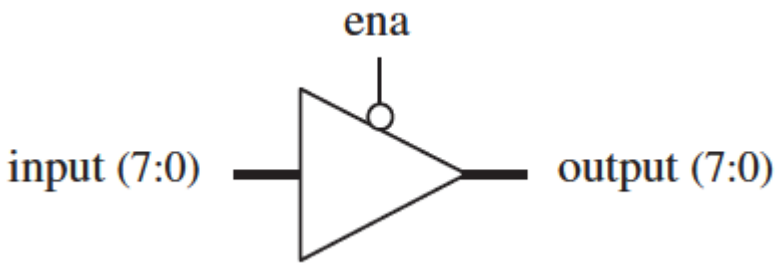

```
1  --- Solution 2: with WITH/SELECT/WHEN -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY mux IS
6     PORT ( a, b, c, d: IN STD_LOGIC;
7            sel: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
8            y: OUT STD_LOGIC);
9  END mux;
10 ----------------------------------------
11 ARCHITECTURE mux2 OF mux IS
12 BEGIN
13    WITH sel SELECT
14       y <=  a WHEN "00",      -- notice "," instead of ";"
15             b WHEN "01",
16             c WHEN "10",
17             d WHEN OTHERS;    -- cannot be "d WHEN "11" "
18 END mux2;
```

selected

when

VHDL> Concurrent Code

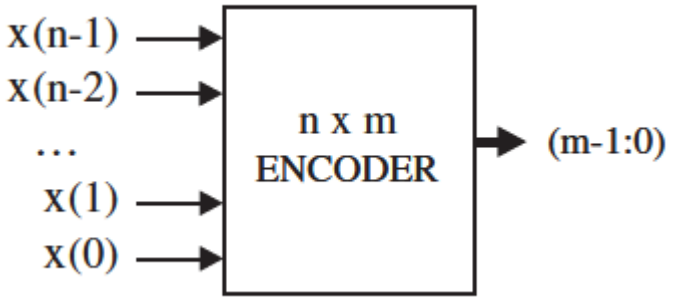
- Tri-state Buffer:




```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3   ---------------------------------------------
4   ENTITY tri_state IS
5      PORT ( ena: IN STD_LOGIC;
6             input: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7             output: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
8   END tri_state;
9   ---------------------------------------------
10  ARCHITECTURE tri_state OF tri_state IS
11  BEGIN
12     output <= input WHEN (ena='0') ELSE
13               (OTHERS => 'Z');
14  END tri_state;
```

VHDL> Concurrent Code

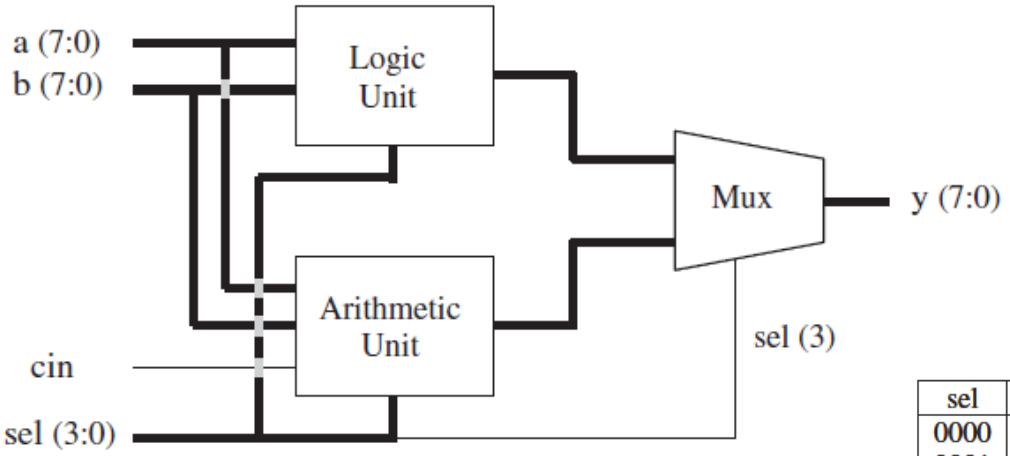
- Home Works: Encoder: page 73:




```
ENTITY encoder IS
   PORT ( x: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
          y: OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
END encoder;
-----------------------------------------
ARCHITECTURE encoder1 OF encoder IS
BEGIN
   y <=   "000" WHEN x="00000001" ELSE
          "001" WHEN x="00000010" ELSE
          "010" WHEN x="00000100" ELSE
          "011" WHEN x="00001000" ELSE
          "100" WHEN x="00010000" ELSE
          "101" WHEN x="00100000" ELSE
          "110" WHEN x="01000000" ELSE
          "111" WHEN x="10000000" ELSE
          "ZZZ";
```

VHDL> Concurrent Code

- Home Works: ALU: page 75



sel	Operation	Function	Unit
0000	y <= a	Transfer a	Arithmetic
0001	y <= a+1	Increment a	
0010	y <= a-1	Decrement a	
0011	y <= b	Transfer b	
0100	y <= b+1	Increment b	
0101	y <= b-1	Decrement b	
0110	y <= a+b	Add a and b	
0111	y <= a+b+cin	Add a and b with carry	
1000	y <= NOT a	Complement a	Logic
1001	y <= NOT b	Complement b	
1010	y <= a AND b	AND	
1011	y <= a OR b	OR	
1100	y <= a NAND b	NAND	
1101	y <= a NOR b	NOR	
1110	y <= a XOR b	XOR	
1111	y <= a XNOR b	XNOR	