جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL 6

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

VHDL> Concurrent Code

- **COMPONENT**:
- A COMPONENT is simply a piece of conventional code (that is, LIBRARY declarations ENTITY ARCHITECTURE). However, by declaring such code as being a COMPONENT, it can then be used within another circuit, thus allowing the construction of hierarchical designs.
- A COMPONENT is also another way of partitioning a code and providing code sharing and code reuse.

VHDL> Concurrent Code

- **COMPONENT**:

COMPONENT declaration:

```
COMPONENT component_name IS
    PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END COMPONENT;
```

COMPONENT instantiation:

```
label: component_name PORT MAP (port_list);
```

VHDL> Concurrent Code

- **COMPONENT**:
  - Example: invertor as component

```
----- COMPONENT declaration: -----------
COMPONENT inverter IS
      PORT (a: IN STD_LOGIC; b: OUT STD_LOGIC);
END COMPONENT;

----- COMPONENT instantiation: -----------
U1: inverter PORT MAP (x, y);
```

VHDL> Concurrent Code

- **COMPONENT**:
- PORT MAP
- There are two ways to map the PORTS of a COMPONENT during its instantiation: **positional** mapping and **nominal** mapping. Let us consider the following example:

```
U1: inverter PORT MAP (x, y);

U1: inverter PORT MAP (x=>a, y=>b);

U2: my_circuit PORT MAP (x=>a, y=>b, w=>OPEN, z=>d);
```

VHDL> Concurrent Code

- The **GENERATE** statement:
  - allows a section of code to be repeated a number of times (loop).
  - GENERATE must be labeled.
  - limits of the range must be **static**.

```
label: FOR identifier IN range GENERATE
  (concurrent assignments)
END GENERATE;
```

VHDL> Concurrent Code

- IF/GENERATE: (ELSE is not allowed).
  - IF/GENERATE can be nested inside FOR/GENERATE, the opposite can also be done.

```
label1: FOR identifier IN range GENERATE
    ...
    label2: IF condition GENERATE
       (concurrent assignments)
    END GENERATE;
    ...
END GENERATE;
```

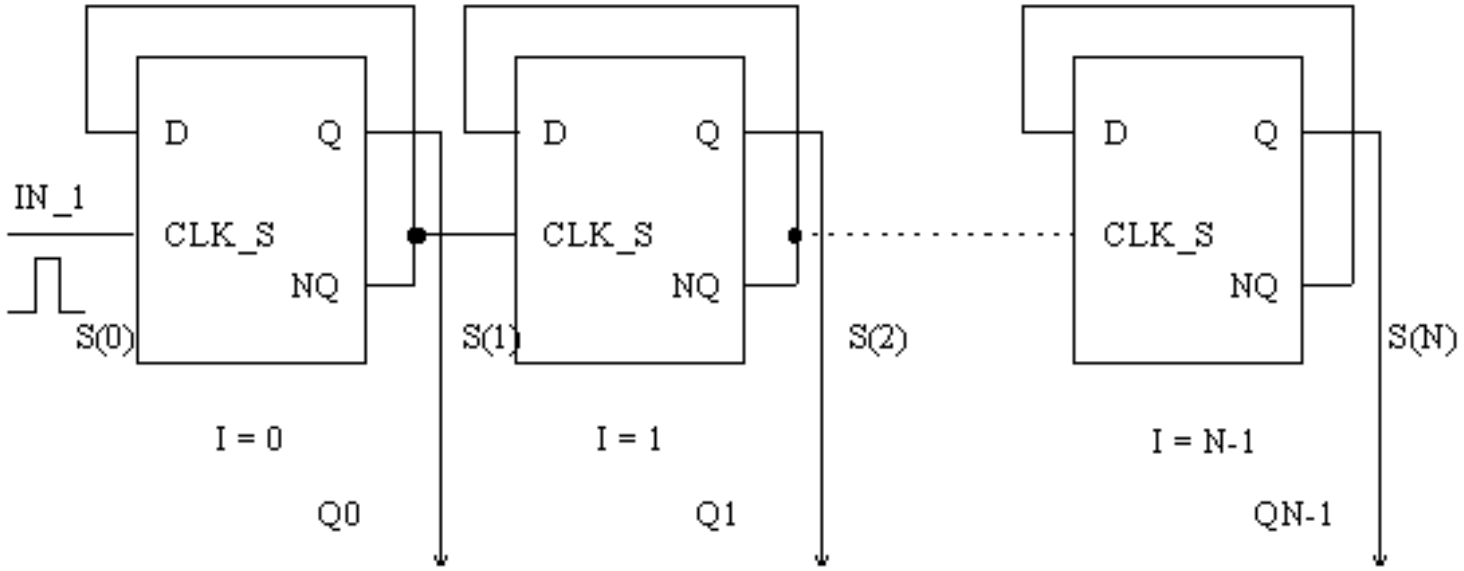
VHDL> Concurrent Code

- Example:

```
SIGNAL x: BIT_VECTOR (7 DOWNTO 0);
SIGNAL y: BIT_VECTOR (15 DOWNTO 0);
SIGNAL z: BIT_VECTOR (7 DOWNTO 0);
...
G1: FOR i IN x'RANGE GENERATE
    z(i) <= x(i) AND y(i+8);
END GENERATE;
```

VHDL> Concurrent Code

- Example: N-bit counter



VHDL> Concurrent Code

- Example:

```
entity COUNTER_BIN_N is
        generic (N : Integer := 4);
         port (Q : out Bit_Vector (0 to N-1);
                IN_1 : in Bit );
end entity COUNTER_BIN_N;


architecture BEH of COUNTER_BIN_N is
      component D_FF
              port(D, CLK_S : in BIT; Q, NQ : out BIT);
      end component D_FF;
signal S : BIT_VECTOR(0 to N);


begin
    S(0) <= IN_1;
    G_1 : for I in 0 to N-1 generate
              D_Flip_Flop :
                  D_FF port map
                    (S(I+1), S(I), Q(I), S(I+1));
          end generate;
end architecture BEH;
```

VHDL> Concurrent Code

- BLOCK:
- Simple BLOCK
  - locally partitioning the code.
  - turning the overall code more readable (long codes).
- can be nested inside another BLOCK.

```
label: BLOCK
   [declarative part]
BEGIN
   (concurrent statements)
END BLOCK label;
```

VHDL> Concurrent Code

- Simple BLOCK:

```
ARCHITECTURE example ...
 BEGIN
   ...
   block1: BLOCK
      BEGIN
      ...
   END BLOCK block1
   ...
   block2: BLOCK
      BEGIN
      ...
   END BLOCK block2;
   ...
END example;
```

VHDL> Concurrent Code

- Simple BLOCK:

nested

```
label1: BLOCK
      [declarative part of top block]
BEGIN
      [concurrent statements of top block]
      label2: BLOCK
            [declarative part nested block]
      BEGIN
            (concurrent statements of nested block)
      END BLOCK label2;
      [more concurrent statements of top block]
END BLOCK label1;
```

VHDL> Concurrent Code

- Guarded BLOCK:
  - includes an additional expression, called guard expression.
- A guarded statement executed only when the guard expression is TRUE.
- sequential circuits can be constructed.

```
label: BLOCK (guard expression)
  [declarative part]
BEGIN
  (concurrent guarded and unguarded statements)
END BLOCK label;
```

VHDL> Concurrent Code

- DFF with Guarded BLOCK:

```
-------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------
ENTITY dffwGBlock IS
PORT ( d, clk, rst: IN STD_LOGIC;
q: OUT STD_LOGIC);
END dffwGBlock;
-------------------------------
ARCHITECTURE dff OF dffwGBlock IS
BEGIN
b1: BLOCK (clk'EVENT AND clk='1')
BEGIN
q <= GUARDED '0' WHEN rst='1' ELSE d;
END BLOCK b1;
END dff;
```