تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة الثامنة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2023-2024

عناوين المحاضرة

- النمذجة البنيوية Structural modeling

1. الحزم Packages
2. المكون Component
3. عبارة Port Map
4. عبارة Generic Map

أمثلة Examples

النمذجة البنيوية

يقصد بالنمذجة البنيوية الطريقة التي يتم فيها وصف نظام رقمي معين باستخدام بنيته الداخلية اي من خلال العناصر المكونة للنظام وطريقة ربط هذه العناصر مع بعضها البعض

1. **الحزم Packages:**

- تستخدم الحزم لتجميع مجموعة من أنواع المعطيات والوظائف والإجرائيات بشكل مسبق يمكن استخدامها بشكل عام أثناء عملية التصميم والنمذجة.
- يوجد العديد من الحزم مسبقة التعريف موزعة بين مكتبات لغة VHDL ويمكن للمستخدم تعريف الحزم الخاصة وتضمينها بأنواع المعطيات والوظائف والإجرائيات الخاصة بالمستخدم وتخزينها ضمن مكتبة العمل work ومن ثم استثمارها ضمن برامج لغة VHDL.

تكتب الصيغة العامة على الشكل:

```
PACKAGE package _name IS
        (declarations)
END package_name;
```

```
Use work.my_package.all;
```

```
PACKAGE my_package IS
        TYPE state( st1,st2,st3,st4);
        TYPE color IS(red, green,blue);
CONSTANT vec: STD_LOGIC_VECTOR (7 DOWNTO 0):=''11111111'';
END my_package;
```

النمذجة البنيوية

2. عبارة المكون COMPONENT:

- تستخدم هذه العبارة لتوصيف دارة رقمية كالبوابة والقلاب ومن ثم إعادة استخدامها لبناء دارة أكثر تعقيداً (عداد، مسجل إزاحة، ALU) كما هو الحال بالنسبة إلى النمذجة البنيوية والنمذجة الهرمية.
- يمكن أن تستخدم عبارة مكون component ضمن البنيان والحزمة وضمن عبارة GENERATE.
- تكتب الصيغة العامة للتصريح بالشكل:

```
COMPONENT component_nameIS
        PORT (port_name:signal_model signal_type;
        port_name:signal_mode signal_type;
        …);
END COMPONENT;
```

```
COMPONENT nand_gate IS
        PORT (a,b:IN BIT; c:OUT BIT);
END COMPONENT;

COMPONENT half_adder IS
        PORT (xi,yi,cin:IN BIT;
                cout, si:OUT BIT);
END COMPONENT;
```

- تم التصريح عن مكون half-adder مع تحديد منافذ الدخل والخرج ونوع هذه المنافذ.

النمذجة البنيوية

3. عبارة PORT MAP:

- تستخدم هذه العبارة لربط مكونات النظام بعضها ببعض Instantiation كما هي ضمن الدارة أو النظام بعد التصريح عن هذه المكونات باستخدام عبارة مكون Component من خلال توافق منافذ الدارة الفعلية مع منافذ المكون.

- توجد طريقتان لعملية توافق المنافذ:

- طريقة الربط باستخدام الموقع: يتم فيها كتابة المنافذ كما هي بالترتيب المذكور في عبارة المكون.

- طريقة الربط باستخدام الاسم: تتم باستخدام المعامل => وفي هذه الحالة فإن الترتيب غير مهم.**ويجب كتابة اسم المكون قبل عبارة PORT MAP كما ورد في التصريح عن المكون.**

```
---------Component declaration----------------
COMPONENT nand_gate IS
        PORT(a,b:IN BIT;c:OUT BIT);
END COMPONENT;
---------Component instantiation:--------------
nand1: nand_gate PORT MAP (x,y,z);          --position mapping
nand2: nand_gate PORT MAP (a=>x, b=>y, c=>z);          --nomial mapping
```

مثال: ربط بوابة NAND باستخدام عبارة PORT MAP

النمذجة البنيوية

3. عبارة PORT MAP:

يمكن التصريح عن المكون Component بطريقتين مختلفتين: 1. **في جسم البرنامج الرئيسي Main code:** نحتاج في هذه الحالة إلى

a) مجموعة من ملفات VHDL عددها يساوي عدد العناصر المكونة للدارة الرقمية حيث يتضمن كل ملف كيان وبنيان العنصر

b) ملف البرنامج الرئيسي الذي يتم فيه التصريح عن هذه المكونات ضمن قسم التصريحات لبنيان الملف الرئيسي

يتضمن الملف الرئيسي آلية ربط هذه المكونات باستخدام عبارة PORT MAP

2. **ضمن الحزمة Package :** نحتاج في هذه الحالة إلى

a) مجموعة من ملفات VHDL عددها يساوي عدد العناصر المكونة للدارة الرقمية حيث يتضمن كل ملف كيان وبنيان العنصر

b) ملف لإنشاء الحزمة التي يتم فيها التصريح عن المكونات

c) ملف البرنامج الرئيسي الذي تتم فيه عملية ربط المكونات

يصرح عن الحزمة المستخدمة والمخزنة ضمن مكتبة work في قسم التصريحات الخاصة بالمكتبات باستخدام u se

النمذجة البنيوية

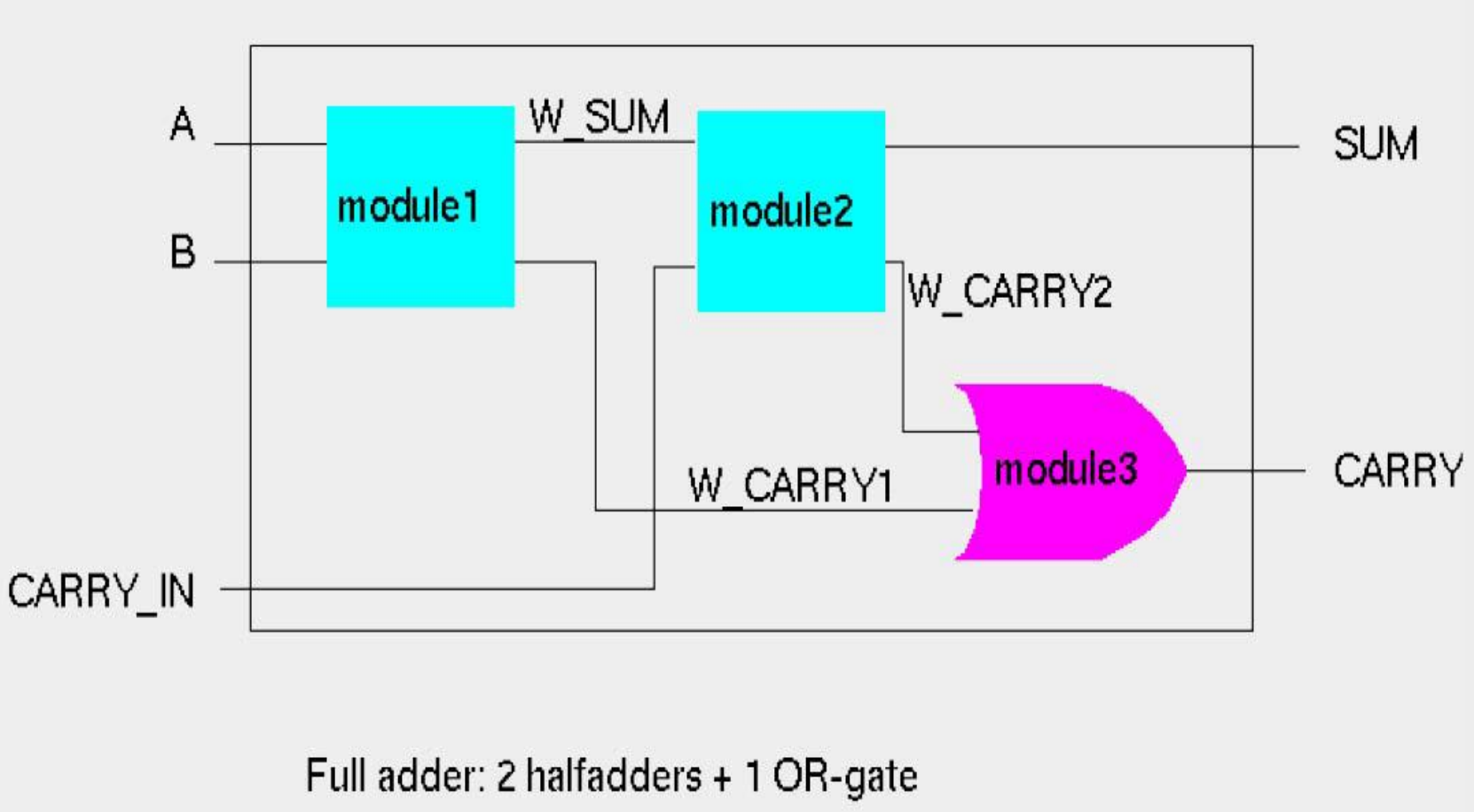
4. عبارة GENERIC MAP:

- إذا تضمنت عبارة المكون component في قسم التصريح عبارة GENERIC للتصريح عن المتحولات وجب عند عملية ربط المكونات مع بعضها البعض أخذ هذا المتحول بعين الاعتبار وذلك من خلال عبارة GENERIC MAP.

```
entity reg4 is
    port ( clk, clr : in bit;  d : in bit_vector(0 to 3);
           q : out bit_vector(0 to 3) );
end entity reg4;
-------------------------------------------------------
architecture struct of reg4 is
    component flipflop is
        generic ( Tprop, Tsetup, Thold : delay_length );
        port ( clk : in bit;  clr : in bit;  d : in bit;
               q : out bit );
    end component flipflop;
begin
    bit0 : component flipflop
        generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
        port map ( clk => clk, clr => clr, d => d(0), q => q(0) );
    bit1 : component flipflop
        generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
        port map ( clk => clk, clr => clr, d => d(1), q => q(1) );
    bit2 : component flipflop
        generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
        port map ( clk => clk, clr => clr, d => d(2), q => q(2) );
    bit3 : component flipflop
        generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
        port map ( clk => clk, clr => clr, d => d(3), q => q(3) );
end architecture struct;
```

أمثلة

جامع كامل باستخدام دراتي نصف جامع وبوابة OR



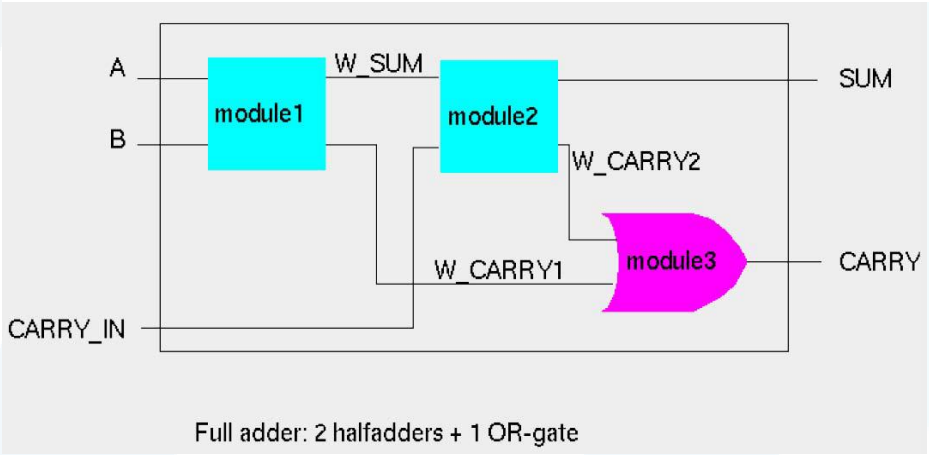
أمثلة

جامع كامل باستخدام دراتي نصف جامع وبوابة OR

```
entity FULLADDER is
    port (A,B, CARRY_IN: in bit;
          SUM, CARRY: out bit);
end FULLADDER;
architecture STRUCT of FULLADDER is
    signal W_SUM, W_CARRY1, W_CARRY2 : bit;
    component HALFADDER
        port (A, B : in bit;
              SUM, CARRY : out bit);
    end component;
    component ORGATE
        port (A, B : in bit;
              RES : out bit);
    end component;
begin
```

```
MODULE1: HALFADDER
        port map ( A     => A,
                   SUM   => W_SUM,
                   B     => B,
                   CARRY => W_CARRY1 );
```

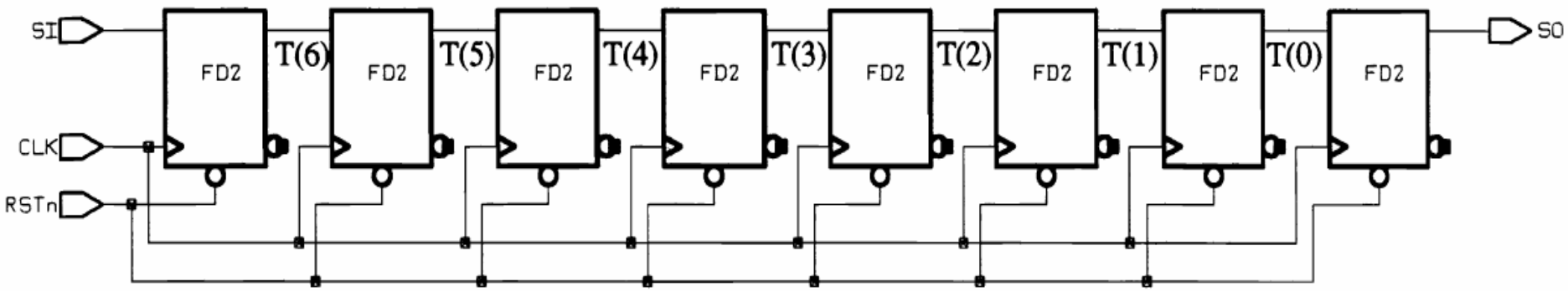
```
begin
  MODULE1: HALFADDER
  port map( A, B, W_SUM, W_CARRY1 );

  MODULE2: HALFADDER
  port map ( W_SUM, CARRY_IN,
             SUM, W_CARRY2 );

  MODULE3: ORGATE
  Port map (W_CARRY2, W_CARRY1, CARRY);
end STRUCT;
```



Full adder: 2 halfadders + 1 OR-gate

أمثلة

مسجل إزاحة 8 بت

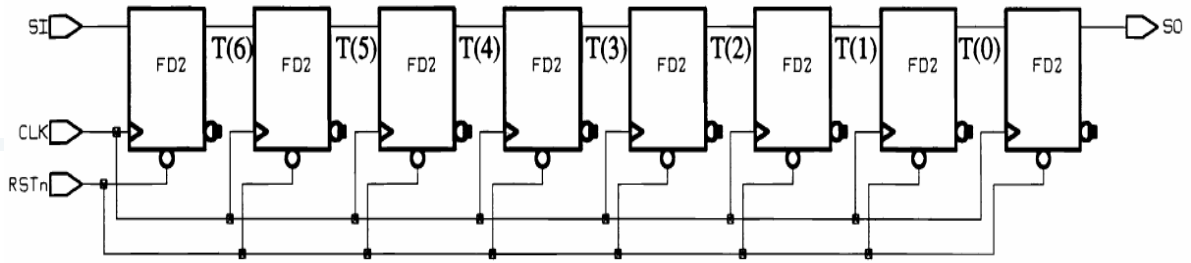



```
1     entity DFF is
2        port (
3           RSTn, CLK, D : in  bit;
4           Q            : out bit);
5     end DFF;
6     architecture RTL of DFF is
7     begin
8        process (RSTn, CLK)
9        begin
10          if (RSTn = '0') then
11             Q <= '0';
12          elsif (CLK'event and CLK = '1') then
13             Q <= D;
14          end if;
15       end process;
16    end RTL;
17    ----------------------------------------
18     entity SHIFT is
19        port (
20           RSTn, CLK, SI : in  bit;
21           SO            : out bit);
22     end SHIFT;
```

Structural modeling consists of three parts
1- behavioral model of the flip-flop using process statement
2- component declaration via component keyword
3- component instantiation via port map keyword

أمثلة

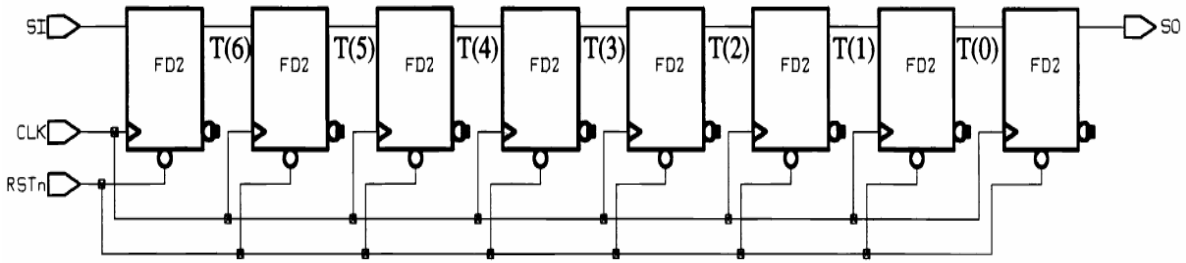
مسجل إزاحة 8 بت




```
23   architecture RTL1 of SHIFT is
24      component DFF
25      port (
26         RSTn, CLK, D : in  bit;
27         Q            : out bit);
28      end component;
29      signal T : bit_vector(6 downto 0);
30   begin
31      bit7 : DFF
32         port map (RSTn => RSTn, CLK => CLK, D => SI, Q => T(6));
33      bit6 : DFF
34         port map (RSTn, CLK, T(6), T(5));
35      bit5 : DFF
36         port map (RSTn, CLK, T(5), T(4));
37      bit4 : DFF
38         port map (CLK => CLK, RSTn => RSTn, D => T(4), Q => T(3));
39      bit3 : DFF
40         port map (RSTn, CLK, T(3), T(2));
41      bit2 : DFF
42         port map (RSTn, CLK, T(2), T(1));
43      bit1 : DFF
44         port map (RSTn, CLK, T(1), T(0));
45      bit0 : DFF
46         port map (RSTn, CLK, T(0), SO);
47   end RTL1;
48   ----------------------------------------
```

أمثلة

مسجل إزاحة 8 بت باستخدام عبارة GENERATE




```
1   architecture RTL2 of SHIFT is
2      component DFF
3      port (
4         RSTn, CLK, D : in  bit;
5         Q            : out bit);
6      end component;
7      signal T : bit_vector(6 downto 0);
8   begin
9      g0 : for i in 7 downto 0 generate
10        g1 : if (i = 7) generate
11           bit7 : DFF
12              port map (RSTn => RSTn, CLK => CLK, D => SI, Q => T(6));
13        end generate;
14        g2 : if (i > 0) and (i < 7) generate
15           bitm : DFF
16              port map (RSTn, CLK, T(i), T(i-1));
17        end generate;
18        g3 : if (i = 0) generate
19           bit0 : DFF
20              port map (RSTn, CLK, T(0), SO);
21        end generate;
22     end generate;
23  end RTL2;
```

In RTL2, generate statement with if and for statement is used:

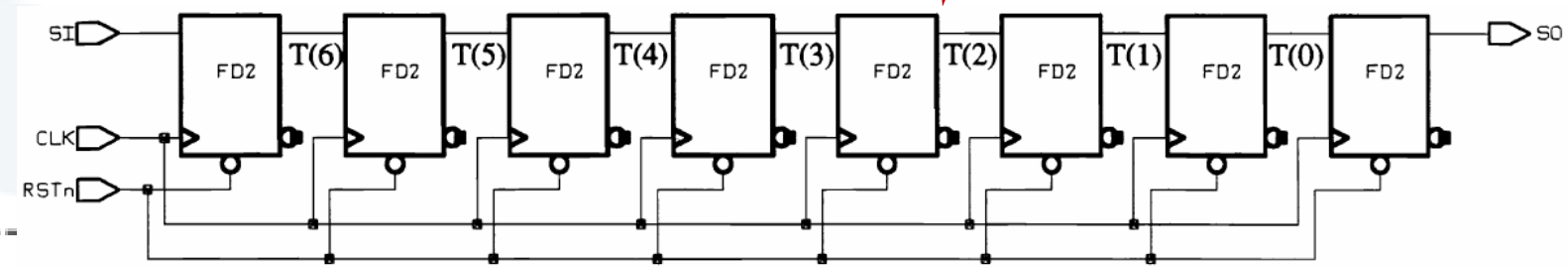
To generate the bit number 7.

To generate the bits from 1 to 6.

To generate the bit number 0

أمثلة

مسجل إزاحة 8 بت باستخدام عبارة GENERATE




```
24    ----------------------------------
25    architecture RTL3 of SHIFT is
26       component DFF
27       port (
28          RSTn, CLK, D : in  bit;
29          Q            : out bit);
30       end component;
31       signal T : bit_vector(8 downto 0);
32    begin
33       T(8) <= SI;
34       SO   <= T(0);
35       g0 : for i in 7 downto 0 generate
36          allbit : DFF
37             port map (RSTn => RSTn, CLK => CLK, D => T(i+1), Q =>
                T(i));
38       end generate;
39    end RTL3;
```

In RLT3, generate statement with for statement is used to generate the all 8-bits