บทที่ 2
องค์ประกอบของภาษา VHDL

- ความรู้เบื้องต้นเกี่ยวกับภาษา VHDL
- โครงสร้างของภาษา VHDL
- ไลบรารี (Libraries)
- คอมโปเนนต์ (Component)

ความรู้เบื้องต้นเกี่ยวกับภาษา VHDL

- การทำงานแบบลำดับและแบบขนาน
- ออบเจ็กต์ (OBJECTS)

การทำงานแบบลำดับและแบบขนาน

- การทำงานแบบลำดับ (Sequential)
- การทำงานแบบขนาน (Concurrent)

```
P1 = seg[i];
P2 = dig[i];
delay_ms(100)
P2 = 0x0F;
```

X <= A and B; Y <= C xor D; Z <= X or Y;	Z <= X or Y; X <= A and B; Y <= C xor D;

ออบเจ็กต์ (OBJECTS)

ออบเจ็กต์ในภาษา VHDL หมายถึงองค์ประกอบหนึ่งๆในระบบ ใช้สำหรับเก็บค่าต่างๆ มีอยู่ด้วยกัน 3 แบบ(Class) ได้แก่

- ค่าคงที่ (Constant)
  - เมื่อกำหนดค่าให้แล้วจะคงค่านั้นตลอดไป ไม่สามารถดัดแปลงหรือแก้ไขได้
- ตัวแปร(Variable)
  - สามารถกำหนดค่าและเปลี่ยนแปลงค่าได้ตลอดการทำงาน แต่ณ เวลาหนึ่งๆจะเก็บค่าได้เพียงค่าเดียวเท่านั้น
- สัญญาณ(Signal)
  - ใช้สำหรับการเชื่อมต่อโมดูลต่างๆเข้าด้วยกัน ถ้าเปรียบเทียบอย่างง่ายๆ Signal แทนขาของอุปกรณ์ หรือสายไฟที่ใช้เชื่อมต่ออุปกรณ์ต่างๆในวงจรเข้าด้วยกัน

กฎเกณฑ์ การตั้งชื่อ Object

- เป็นพยัญชนะ ตัวเลข หรือเครื่องหมายขีดเส้นใต้ในภาษาอังกฤษก็ได้แต่ต้องขึ้นต้นด้วยพยัญชนะเสมอ โดยมีความยาวไม่จำกัด
- พยัญชนะตัวเล็กหรือตัวใหญ่มีค่าเท่ากัน
- ห้ามใช้คำสงวนของ VHDL เช่น PORT ENTITY SIGNAL และ PROCESS เป็นต้น

การประกาศใช้ออบเจ็กต์

```
OBJECT_CLASS object_name : TYPE [:= initial_value];
```

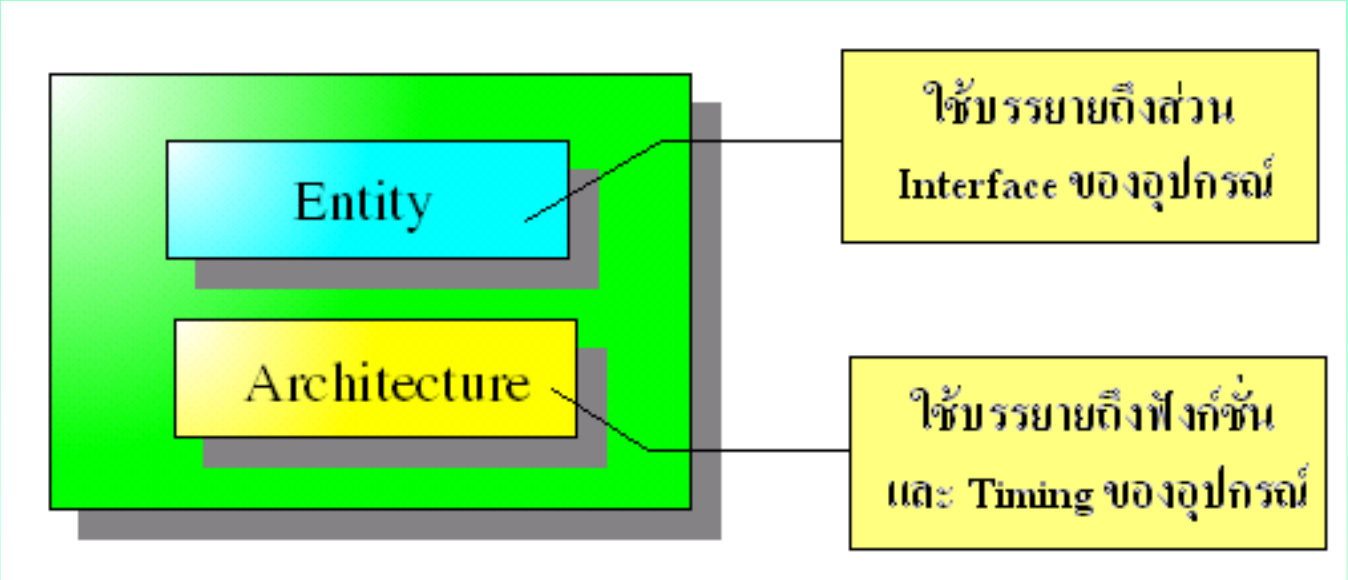
```
SIGNAL A_signal : BIT;
SIGNAL B_signal : BIT := '0';
```

```
CONSTANT pi IS: REAL := 3.14159;
```

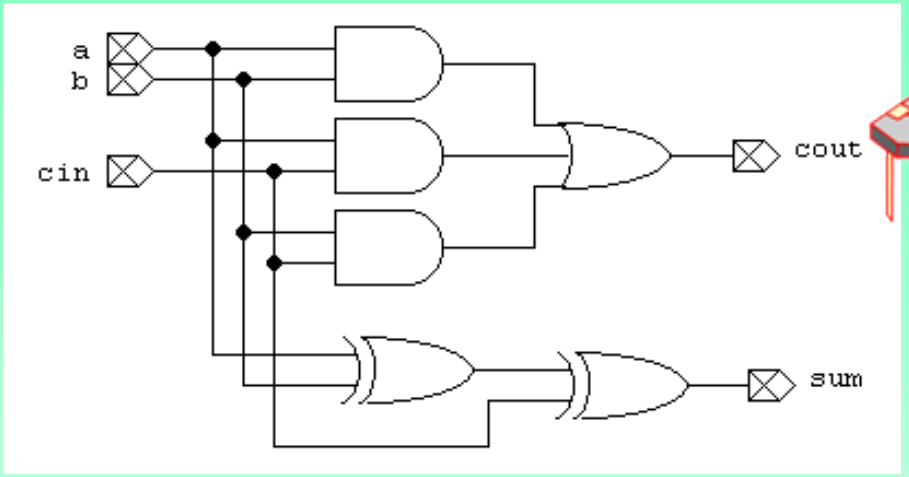
```
variable SUM, COUT : std_logic;
variable RESULT : std_logic_vector  (1 downto 0);
```

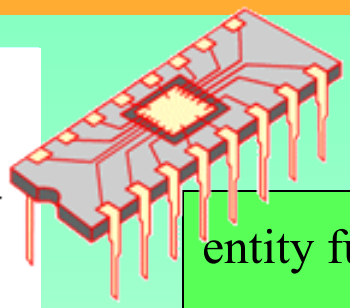
```
entity fulladd is
   port (a, b, cin : in std_logic;
            sum, cout : out  std_logic);
end fulladd;
```

โครงสร้างของภาษา VHDL



ตัวอย่าง





Entity

```
entity fulladd is
   port (a, b, cin : in bit;
         sum, cout : out bit);
end fulladd;
```

Architecture

```
architecture addflow of fulladd is
begin
   sum <= a xor b xor cin;
   cout <= (a and b) or (a and cin) or (b and cin);
end addflow;
```

ENTITY Declaration

```
ENTITY entity_name IS
        generic clause
        port clause
END ENTITY entity_name ;
```

รูปแบบ

GENERIC ใช้กำหนดค่าพารามิเตอร์ต่างๆ เช่นค่าพารามิเตอร์เกี่ยวกับเวลาหน่วง ของอุปกรณ์

PORT ใช้กำหนดขาหรือสัญญาณของโมดูลที่จะใช้ติดต่อกับโลกภายนอก

พอร์ท (Port)

พอร์ทประกอบด้วย

- ชื่อพอร์ท (Name)
- โหมดหรือทิศทางของพอร์ท (Mode)

IN : เป็นสัญญาณอินพุท (Input)

OUT : เป็นสัญญาณเอาท์พุท (Output)

INOUT : เป็นสัญญาณสองทิศทาง (Bidirectional)

BUFFER : ถ้าพิจารณาภายนอกอุปกรณ์มีลักษณะเหมือน OUT และถ้าพิจารณาภายในอุปกรณ์มีลักษณะเป็น INOUT

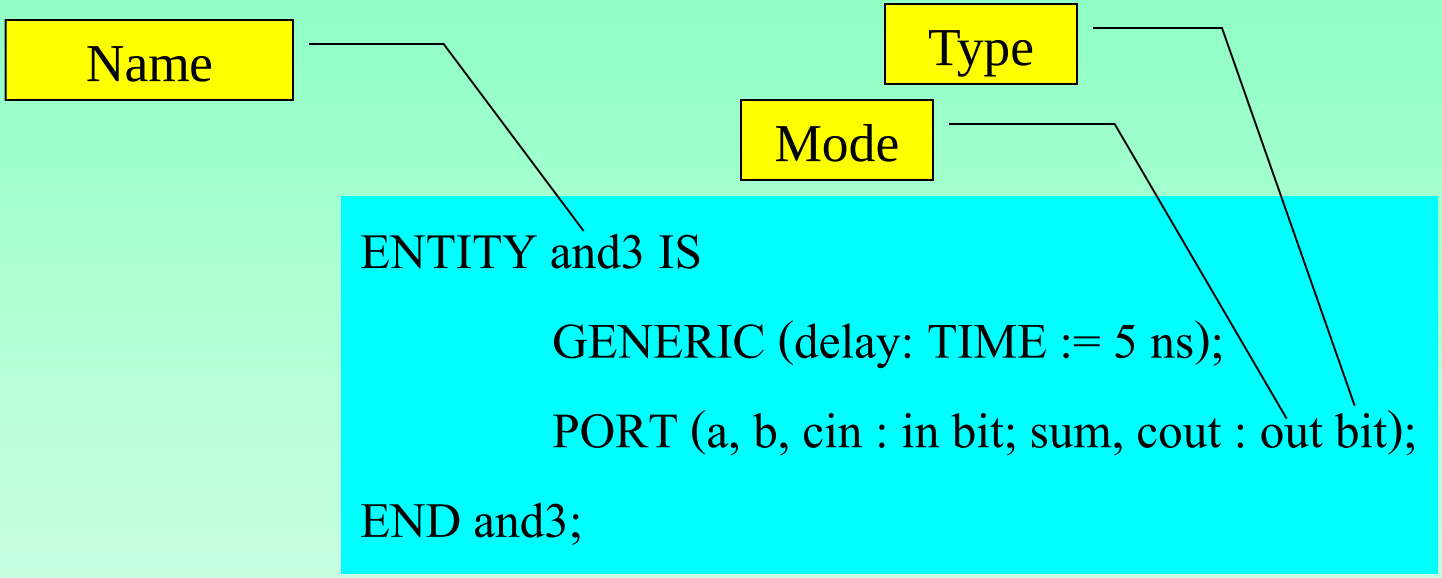
พอร์ท (Port) (ต่อ)

- Type เป็นแบบหรือชนิดของข้อมูลของพอร์ท ซึ่งภาษา VHDL จะมีแบบมาตราฐานอยู่ เรียกว่าStandard Types ได้แก่

BIT	มีค่าเป็น 0 หรือ 1
BIT_VECTOR	เป็นกลุ่มของ BIT
BOOLEAN	มีค่าเป็นจริง (1) หรือเท็จ (0)
INTEGER	มีค่าเป็นเลขจำนวนเต็ม
REAL	มีค่าเป็นเลขจำนวนจริง
TIME	มีค่าเป็นเวลา
CHARACTER	เป็นตัวอักขระ
STRING	เป็นกลุ่มของตัวอักขระ

นอกจากนี้ยังมี std_logic และ std_logic_vector (ใช้ไลบรารี่)

ตัวอย่าง Entity



เครื่องหมาย ";" ที่ปรากฏอยู่ส่วนท้ายของแต่ละบรรทัดเป็นการบอกว่าจบประโยคคำสั่ง ซึ่งเครื่องหมายนี้สามารถใช้ได้ทุกที่ในการเขียนจบประโยคคำสั่ง

เครื่องหมาย "- -" ใช้ระบุข้อความที่เป็นหมายเหตุ (Comment)

ตัวอย่าง Entity (ต่อ)

```
entity fulladd is
        port (a, b, cin : in bit;
        sum, cout : out  bit);
end fulladd;
```

```
entity fulladd is
        port (  a : in bit;
                b : in bit;
                cin : in bit;
                sum : out  bit;
                cout : out  bit);
end fulladd;
```

```
entity test_fulladd is
end test_fulladd;
```

Entity สำหรับ Testbench

ARCHITECTURE Declaration

เป็นส่วนที่ใช้สำหรับบรรยายถึงฟังก์ชั่นการทำงานของโมดูล โดยมีความสัมพันธ์กับ Entity ที่ได้กำหนดไว้

```
ARCHITECTURE arch_name OF entity_name IS
        architecture declarative part
        ……..
BEGIN
        statements
        ……..
        ……..
END arch_name ;
```

ARCHITECTURE Declaration (ต่อ)

arch_name เป็นชื่อของ Architecture การตั้งชื่อใช้กฎเกณฑ์เดียวกับการตั้งชื่อออบเจ็กต์

entity_name เป็นชื่อของ Entity ของ Architecture นี้ประกอบอยู่

architecture declarative part เป็นส่วนที่ใช้ประกาศออบเจ็กต์หรือสิ่งต่างๆที่จะนำไปใช้ในโมดูล ซึ่งมีอยู่หลายอย่างได้แก่

- Signal
- Constant
- Type
- Subtype
- Subprogram
- Component

ARCHITECTURE Declaration (ต่อ)

STATEMENTS เป็นคำสั่งของภาษา VHDL เป็นคำสั่งที่ใช้ทำงานหรือเป็นคำสั่งที่บ่งบอกการเชื่อมต่อของอุปกรณ์ต่างๆเข้าด้วยกัน คำสั่งเหล่านี้มีหลายแบบดังนี้

- **CONCURRENT Statements** คำสั่งที่มีการทำงานเป็นแบบขนาน (Concurrent) การทำงานภายใน Architure นี้ทุกคำสั่งถือว่าทำงานไปพร้อมๆกันคือเป็นแบบขนาน
- **PROCESS Statement** เป็นคำสั่งที่มีการทำงานเป็นแบบลำดับ แต่คำสั่งเหล่านี้ต้องอยู่ภายในคำสั่ง PROCESS
- **การกำหนดค่าสัญญาณ** เป็นการเชื่อมต่อสัญญาณจากสัญญาณหนึ่งส่งให้อีกสัญญาณหนึ่ง
- **การเชื่อมต่ออุปกรณ์ (Component)** เป็นการต่ออุปกรณ์ต่างๆเข้าด้วยกัน

คำสั่งภายใน Architecture เป็นอะไร ?

```
ARCHITECTURE arch_name OF entity_name IS
BEGIN
        concurrent statements;
        concurrent statements;
        ……….
        PROCESS
        BEGIN
           Sequential statements;
           Sequential statements;
           ……….
        END PROCESS;
END ARCHITECTURE arch_name ;
```

ถือว่า Process หนึ่งๆ เป็นหนึ่ง Concurrent statement

ตัวอย่าง ARCHITECTURE

```
ARCHITECTURE addflow OF fulladd IS
BEGIN
        sum <= a xor b xor cin;
        cout <= (a and b) or (a and cin) or (b and cin);
END addflow;
```

Concurrent statements

Concurrent statements

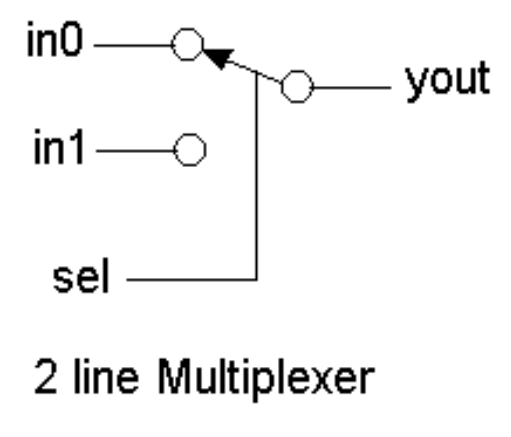
Architecture Styles

1. แบบอธิบายพฤติกรรม (Behavioral Description) เป็นรูปแบบการเขียนระดับสูง คล้ายกับการโปรแกรมในภาษาคอมพิวเตอร์ทั่วไป
2. แบบอธิบายการไหลของข้อมูล (Dataflow Description) คล้ายกับสมการโลจิก
3. แบบอธิบายโครงสร้าง (Structural Description) คล้ายกับการต่ออุปกรณ์
4. แบบผสม (Mixed Model Description)

อนึ่ง Entity หนึ่งๆสามารถมีได้หลาย Architecture

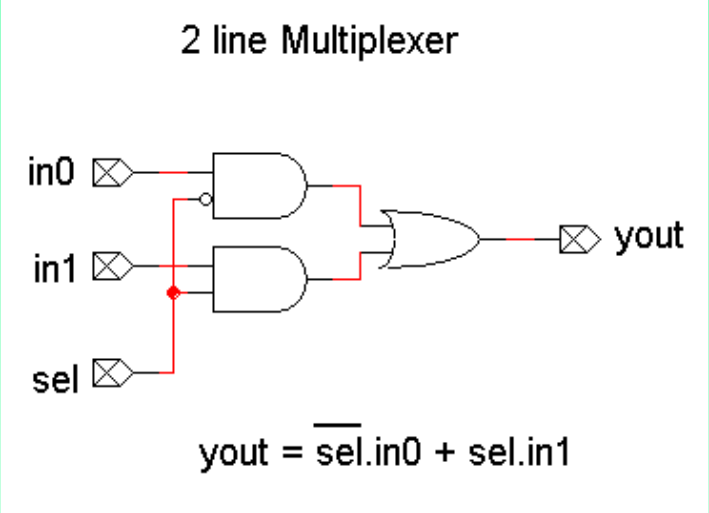
แบบอธิบายพฤติกรรม (Behavioral Description)

ตัวอย่างวงจร Multiplexer




```
ARCHITECTURE behavioral OF
   Mux2x1 IS
BEGIN
   PROCESS (in0, in1, sel)
   BEGIN
        IF sel = 0 THEN
              yout <= in0;
        ELSE
              yout <= in1;
        END IF;
   END PROCESS;
END behavioral ;
```

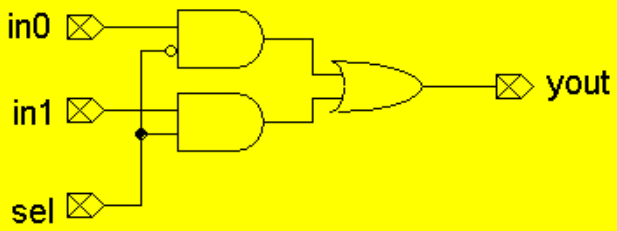
แบบอธิบายการไหลของข้อมูล (Dataflow Description)



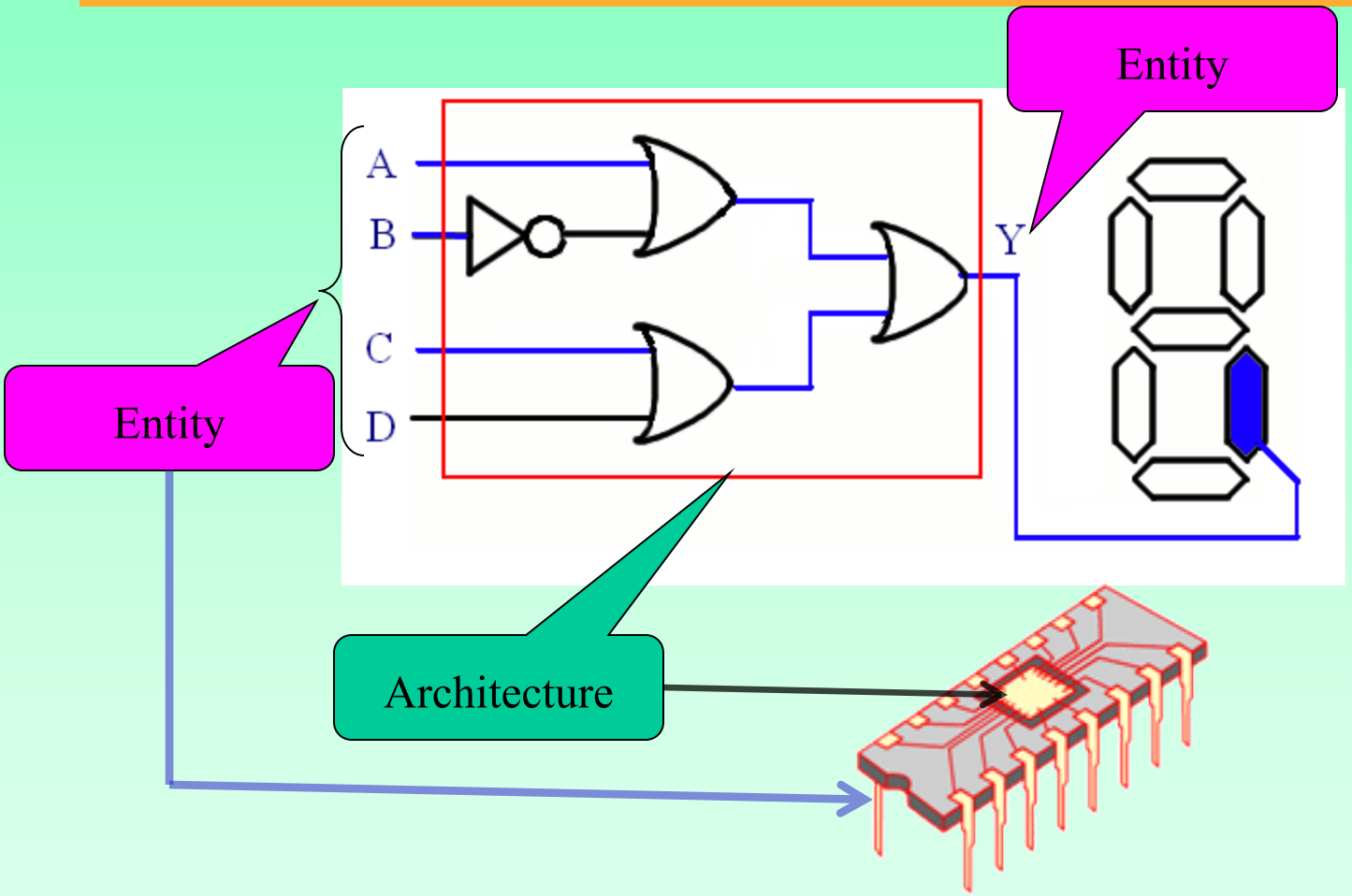

```
ARCHITECTURE dataflow OF Mux2x1 IS
BEGIN
   yout <= ((NOT sel) AND in0) OR (sel AND in1);
 END dataflow;
```

แบบอธิบายโครงสร้าง (Structural Description)

```
ARCHITECTURE structural OF Mux2x1 IS
Component INV
    PORT (i1: in bit; y1 : out bit);
End component;
Component AND2
    PORT (iand1, iand2 : in bit; yand : out bit);
End component;
Component OR2
    PORT (ior1, ior2 : in bit; yor : out bit);
End component;
SIGNAL S0, S1, S2 : bit;
BEGIN
    IC1: INV PORT MAP (i1 => sel, y1 => s0);
    IC2: AND2 PORT MAP (iand1 => in0, iand2 => s0, yand => s1);
    IC3: AND2 PORT MAP (iand1 => in1, iand2 => sel, yand => s2);
    IC4: OR2 PORT MAP (ior1 => s1, ior2 => s2, yor => yout);
END structural;
```



สรุป ครั้งที่ 1 ลักษณะโดยรวมของ VHDL



สรุป ครั้งที่ 1 ลักษณะโดยรวมของ VHDL