# Об’єкти, типи і вирази мови VHDL

# Основи VHDL

Розробка **V**ery high **speed integrated circuit** **H**ardware **D**escription **L**anguage почата в 1980г.

**За основу** прийняті мовні концепції мови паралельного програмування встроюваних систем Ада.

Основні особливості VHDL:

1. **Типи даних,** що підтримують опис апаратури

2. **Оператори**, що підтримують моделювання поведінки системи в часі

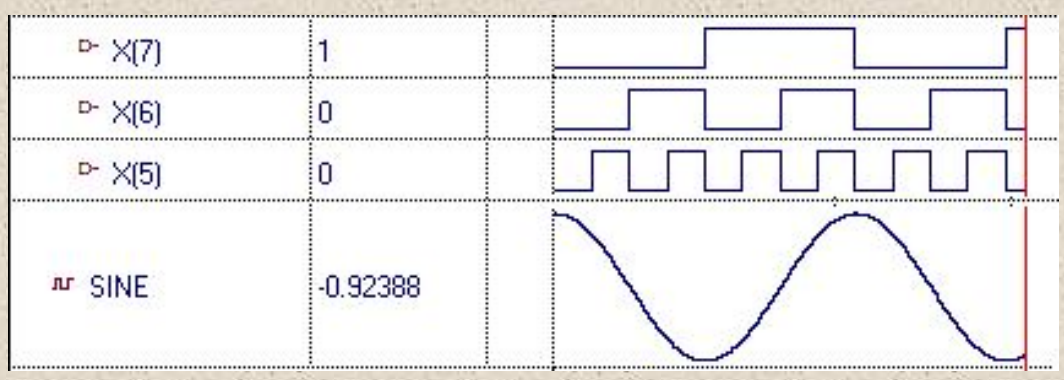
3. **Керування** як потоком команд, так і потоком даних

# Типи даних, що підтримують опис апаратури

**SIGNAL** - синхронізуюче **повідомлення** від процеса до процесу;

- дане, що передається між процесами;
- аналог сигнала, що моделюється.

**SIGNAL** можна запам'ятати і відтворити в його часовому розвитку в симуляторі VHDL



# Типи даних, що підтримують опис апаратури

**Типи даних**, аналогічні даним в реальних пристроях:

| | |
|---|---|
| **Boolean** | true \| false |
| **BIT** | ‘0’ \| ‘1’ |
| **BIT_VECTOR** | “0010101011” |
| **Integer** | 123456 |
| **Real** | 2.0001 |

**Типи даних** для моделювання ситуацій в лініях зв’язку і логічних схемах (STD_LOGIC) :

‘**U’** - не ініційований

‘**X’** - сильний невідомий

‘**W’**- слабкий невідомий

**‘Z’** - високий імпеданс

‘**H’** - слаба 1

**‘L’** - слабий 0

**‘-‘** - байдуже

# Типи даних, що підтримують опис аппаратури

**Фізичні типи даних**:

TIME в fs, ps, ns, us, ms

VOLTAGE в uv, mv, v, kv

NATURE: (voltage, current)

# Entity і Architecture

Опис проекта мовою VHDL складається з двох частин:

- декларація **ENTITY**
- опис **ARCHITECTURE**

- ENTITY описує входи-виходи проекту
- ARCHITECTURE описує функціонування проекту
- Для кожної архітектури повинно бути **entity**, тому до проекту звертаються як до пари **ENTITY(ARCHITECTURE)**

# Entity - опис інтерфейса

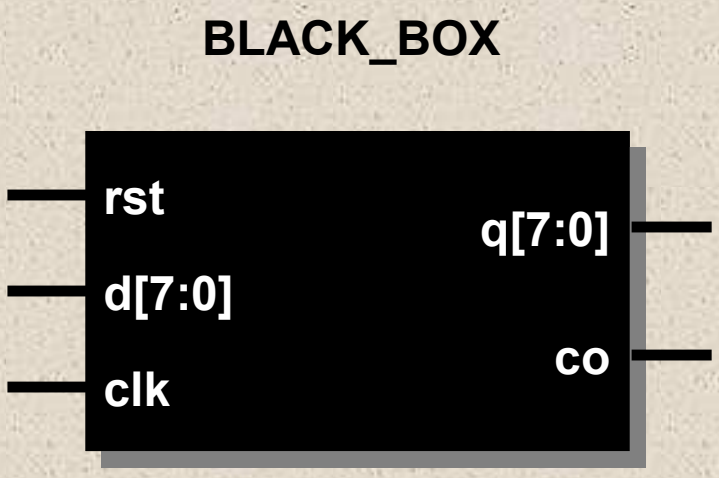
```
entity ENTITY_NAME is
   port (
      <port_declarations>
         );
end ENTITY_NAME;
```

```
architecture ARCH_NAME of ENTITY_NAME is
      <тут об'являють типи і сигнали,
      процедури, функції та компоненти>
begin
   <statements>
end ARCH_NAME;
```

# Entity - опис інтерфейса

Концепція “чорного ящика”

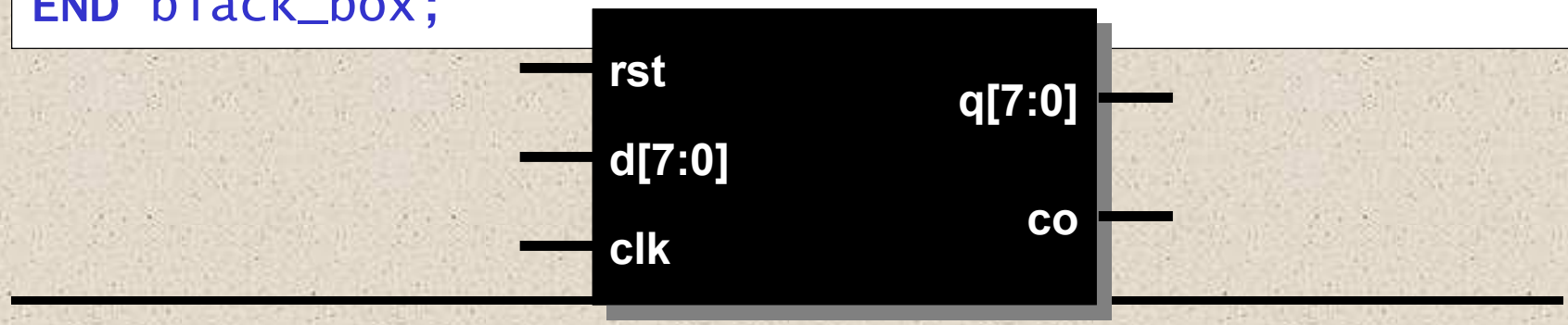
- ENTITY описує інтерфейс чорного ящика (тобто входи-виходи проекту)



# Entity - опис інтерфейса

Приклад опису ENTITY “BLACK BOX”

```
ENTITY black_box IS
      PORT (
            clk:  IN  std_logic;
            rst:  IN  std_logic;
            d  :  IN  std_logic_vector(7 DOWNTO 0);
            q  :  OUT std_logic_vector(7 DOWNTO 0);
            co :  OUT std_logic
      );
END black_box;
```



# Entity - опис інтерфейса

Структура Entity :

- **entity_name** – будь-яке ім’я
- **generics** - константи для настройки проекту
- **name** - ідентифікатор сигналу-порта
- **mode** - режим (напрямок) вводу-виводу порта
- **type** - вказує тип сигналу - BIT, STD_LOGIC

```
ENTITY entity_name IS
      -- optional generics
      PORT (
            name : -- mode type ;
            ...
      ) ;
END entity_name;
```

# Entity - опис інтерфейса

Розділ **entity** вміщує опис портів об’єкта проекта

- через ПОРТИ виконується обмін даними
- під ПОРТАМИ маються на увазі виводи пристрою

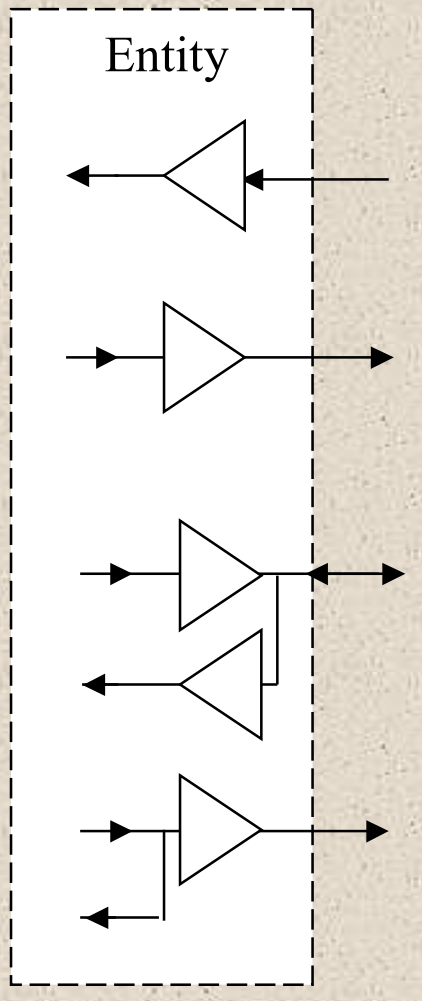
Порти завжди описуються рядом ознак

- Ім’я порту,
- режим порту,
- тип порту

# Entity - опис інтерфейса

Режим порту вказує напрямок передачі даних:

- IN дані тільки входять в entity
- OUT дані тільки виходять назовні (і не використовуються всередині)
- INOUT дані передаються в двох напрямках (входять в і виходять з entity)
- BUFFER дані видаються назовні і використовуються всередині



# Architecture - опис поведінки моделі

## Об’ява архітектури

**arch_name** – будь-яке ім’я

- **entity_name** – ім’я об’єкта - entity
- допоміжні об’яви **сигналів**
- тіло архітектури - поведінка моделі
- **оператори** описують функціонування архітектури

```
ARCHITECTURE arch_name OF entity_name IS
-- допоміжні об”яви сигналів
BEGIN
      --VHDL -оператори
END arch_name;
```

# Architecture - опис поведінки моделі

Архітектура описує поведінку і структуру **entity** - чорного ящика.

Структурний опис

- включення екземплярів (Instantiation – розміщення і з'єднання як в схемі) блоків, які називаються **компонентами**

```
U1:FDE port map(C=>clk,CE=>ce,D=>d1,Q=>q1);
```

Поведінковий опис і опис потоків даних

- Алгоритмічний опис:

```
IF a = b THEN state <= state5;
```

- Булевські рівняння (так званий, стиль потоків даних):

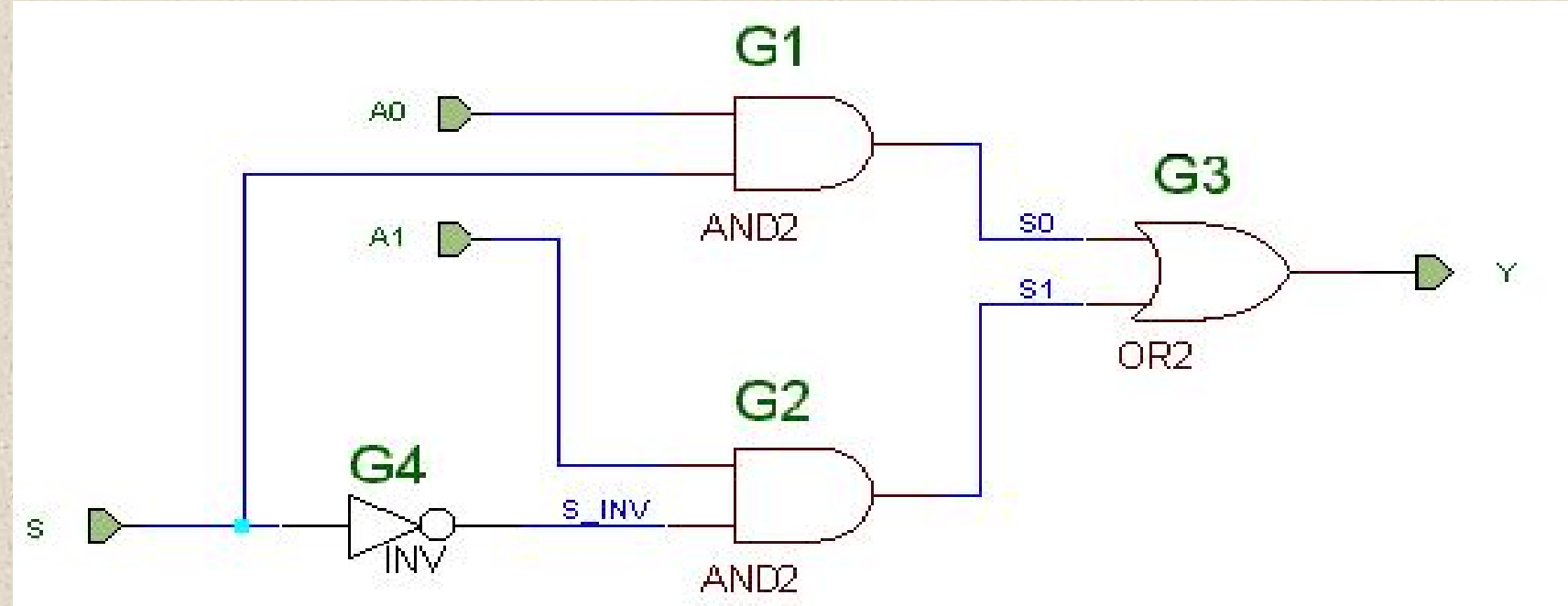
```
x <= a OR (b AND c);
```

# Поведінкова модель

Напишемо VHDL - модель для наступної схеми

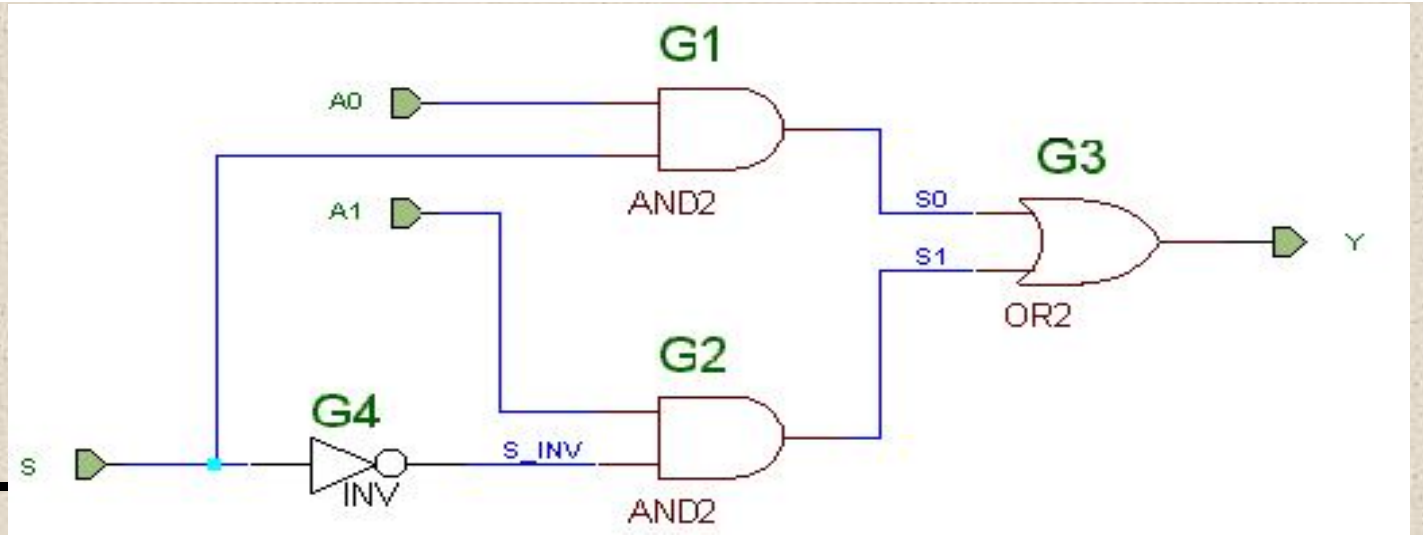
MY_MUX – ім’я об’єкта проекта **Entity**

MY_MUX_ARCH – ім’я архітектури



# Поведінкова модель

```
entity MY_MUX is
  port(S,A0,A1: in BIT;
       Y: out BIT);
end MY_MUX;
architecture MY_MUX_ARCH of MY_MUX is
  begin
     Y<=(A0 and S) or (A1 and not S);
end MY_MUX_ARCH ;
```



# Типи даних в VHDL

Всі об’єкти у VHDL , тобто порти, сигнали, змінні, константи необхідно об’являти як об’єкти визначеного типу.

## Типи стандарту IEEE 1076

- BIT - тип, який приймає тільки значення '0' або '1'
- BIT_VECTOR - група бітів - вектор

```
SIGNAL      a: BIT_VECTOR(0 TO 3);
SIGNAL      b: BIT_VECTOR(3 DOWNTO 0);
      a <=  "0111";
      b <=  "0101";
--що означає:
      a(0) = '0'  b(0) = '1'
      a(1) = '1'  b(1) = '0'
      a(2) = '1'  b(2) = '1'
      a(3) = '1'  b(3) = '0'
```

# Типи даних в VHDL

## INTEGER

використовується як індекси в циклах, константи, настроювальні константи (generics) або для високорівневого моделювання

## BOOLEAN

приймають значення ‘TRUE’ або ‘FALSE’

## Перелічуваний тип

вміщує множину можливих значень, визначених користувачем

```
TYPE \колір\ IS (\зелений\,\жовтий\,\червоний\);
```

# Тип Standard Logic з пакету 1164

Тип STD_LOGIC, заданий в стандартному пакеті, розширює обмеження типу BIT

9 значень замість всього лише двох ('0' і '1')

- Збільшення гнучкості програмування VHDL , синтезу і моделювання
- типи **STD_LOGIC** і **STD_LOGIC_VECTOR** використовуються замість типів **BIT** і **BIT_VECTOR** у випадку задання систем з декількома логічними рівнями
- типи **STD_LOGIC** і **STD_LOGIC _VECTOR** повинні використовуватись якщо задається логіка з трьома станами (0,1,Z)
- Для використання цих типів необхідно об’явити наступний пакет:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

# Тип Standard Logic

Типи STD_LOGIC і STD_LOGIC_VECTOR є *стандартними логічними типами* для промислових розробок цифрових пристроїв:

**Значення для моделювання і синтезу**

- ‘0’ -- Сильний ‘0’
- ‘1’ -- Сильна ‘1’
- ‘Z’ -- Високий імпеданс
- ‘L’ -- Слабий ‘0’
- ‘H’ -- Слаба ‘1’
- ‘-’ -- Довільне

**Значення тільки для моделювання (металогіка):**

- ‘U’ -- Неініціалізоване
- ‘X’ -- Сильне невідоме
- ‘W’ -- Слабе невідоме

# Сигнал, змінна, константа

- Поняття **signal** відноситься до об’єктів, які об’явлені в розділах декларації сигналів і декларації портів.
- Сигнал може мати декілька джерел, кожне з яких може мати наявне значення і деяке майбутнє значення.
- Сигнал є основним об’єктом, який описує аппаратуру і представляє деякий еквівалент лінії зв’язку.

декларація_сигналу ::= **signal** /список_ідентифікаторів/ :
/вказівник_підтипу/ [/вид_сигналу/] [:= /вираз/];

/вид_сигналу/ ::= **register** | **bus**

# сигнал

**Сигнал може бути явно об’явлений в декларативних частинах:**

- **декларація пакета** – сигнал, що видимий у всіх об’єктах (**entity**) проекта, які використовують цей пакет;
- **архітектура** - сигнал видимий тільки в тілі архітектури;
- **блок** - сигнал видимий тільки в межах даного блока;
- **підпрограма** (функція або процедура) - сигнал видимий тільки у відповідному виклиці підпрограми;

```
SIGNAL a: BIT_VECTOR(0 TO 3);-- зростаючий діапазон
SIGNAL b: BIT_VECTOR(3 DOWNTO 0);--зменшуваний діапазон
      a <=  "0111";        -- парні лапки - для векторов
      b <=  "0101";
```

# Змінна

**Змінна –** об’єкт з єдиним наявним значенням.

Значення змінної міняється миттєво у операторі присвоювання змінної.

**Змінна** використовується в процесах і підпрограмах, в яких вона об’явлена. Область видимості змінної обмежена процессом або підпрограмою.

**Об’ява змінної** включає один або кілька ідентифікаторів, вказівку на підтип і необов”язково - вираз, який задає начальне значення.

```
variable /ім’я_змінної/: тип;
variable /ім’я_змінної/: тип := /початкове значення/;
```

# Змінна

```
variable X,Y : INTEGER;
variable A,B : BIT_VECTOR (0 to 7);
type \комбінація_біт\ is record
    \поле_біт\ : BIT;
    \поле_ціле\ : INTEGER;
end record;
variable C,D : \комбінація_біт\ ;

X:= 1000;
A:= B;
A:= “11111111”;

A (3 to 6) := (‘1’,’1’,’1’,’1’);
A (0 to 5) := b(2 to 7);

A (7) := ‘0’;
B (0) := A(6);

C.\поле_біт\  := ‘1’;
D.\поле_ціле\ := C.\поле_ціле\ ;
```

# Константа

**Константа** – об’єкт, значення якого задається один раз в проекті і не може бути змінене.

**Константа** может бути явно об’явлена або може бути піделементом явно заданної константи (вектора, агрегата, комбінованого значення).

```
Декларація_константи ::=
  constant /список_ідентифікаторів/:/вказівник_типу/
    [ := /вираз/];

type \день_т\ is (\пн\,\вт\,\ср\,\чт\,\пт\,\сб\,\нд\);
constant \ПершийДень\  : \день_т\ := \пн\;
constant GND : bit := ‘0‘;
```