CO102
Computer Hardware Experiment

Lecture 02: VHDL Programming

Liang Yanyan

澳門科技大學
Macau of University of Science and Technology

Basic Language Framework

```
library ieee;
use ieee.std_logic_1164.all;
-----------------------------------------
entity XYZ is
    port
    (
        A, B, C :   in  std_logic; -- Comments
        F       :   out std_logic
    );
end XYZ;
-----------------------------------------
architecture XYZ_arch of XYZ is
begin

    F <= (A and B) or (B and C) or (C and A);

end XYZ_arch;
```

Include…

Entity

Architecture

Entity

```
library ieee;
use ieee.std_logic_1164.all;
----------------------------------------
entity XYZ is
    port
    (
        A, B, C          :   in  std_logic;
        F                :   out std_logic
    );
end XYZ;
----------------------------------------
architecture XYZ_arch of XYZ is
begin
    F <= (A and B) or (B and C) or (C and A);
end XYZ_arch;
```

Include…

Entity

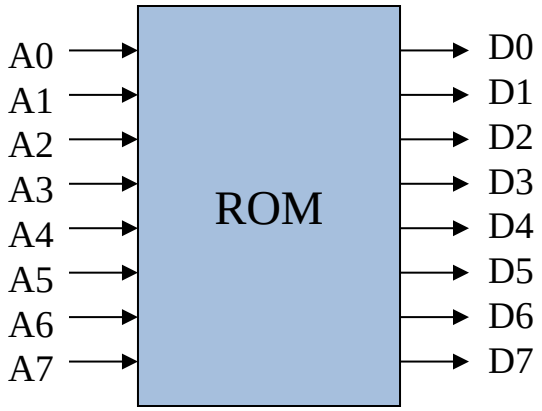
Architecture

Entity Definition

```
entity entity_name is
    [Generics;]
    [Ports;]
    [Other Declarative Parts;]
    [Statements;]
end [ entity ] [ entity_name ] ;
```

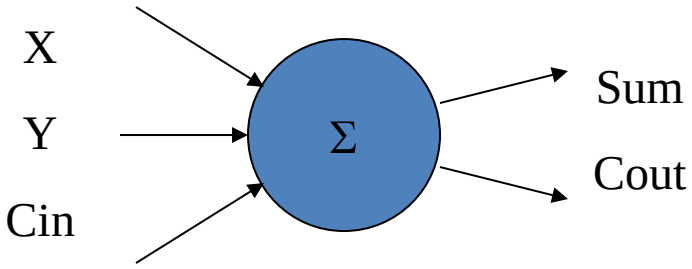
Entity Examples (ROM)

```
entity ROM is
    port ( D0                    : out bit;
           D1                    : out bit;
           D2                    : out bit;
           D3                    : out bit;
           D4, D5, D6, D7        : out bit;
           A   : in bit_vector(7 down to 0) );
end ROM;
```



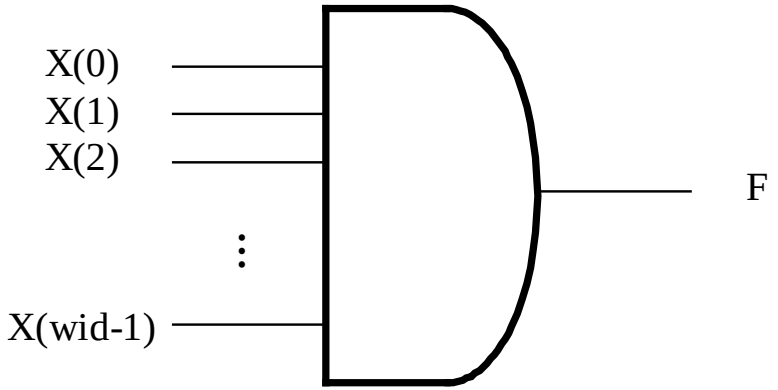
Entity Examples (Adder)

```
entity Full_Adder is
        port (X, Y, Cin: in Bit;
              Cout, Sum: out Bit) ;
end Full_Adder ;
```



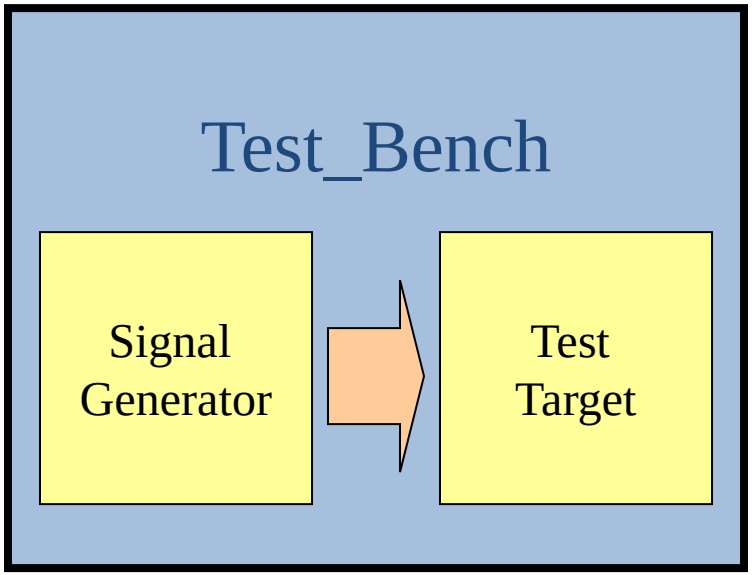
Entity Examples (n-input AND)

```
entity ANDN is
    generic (wid : integer := 2);
    port
    (
        X   :   in  bit_vector(wid-1 downto 0);
        F   :   out bit
    );
End ANDN;
```



Entity Example (Empty Entity)

```
entity Test_Bench is
end Test_Bench ;
```



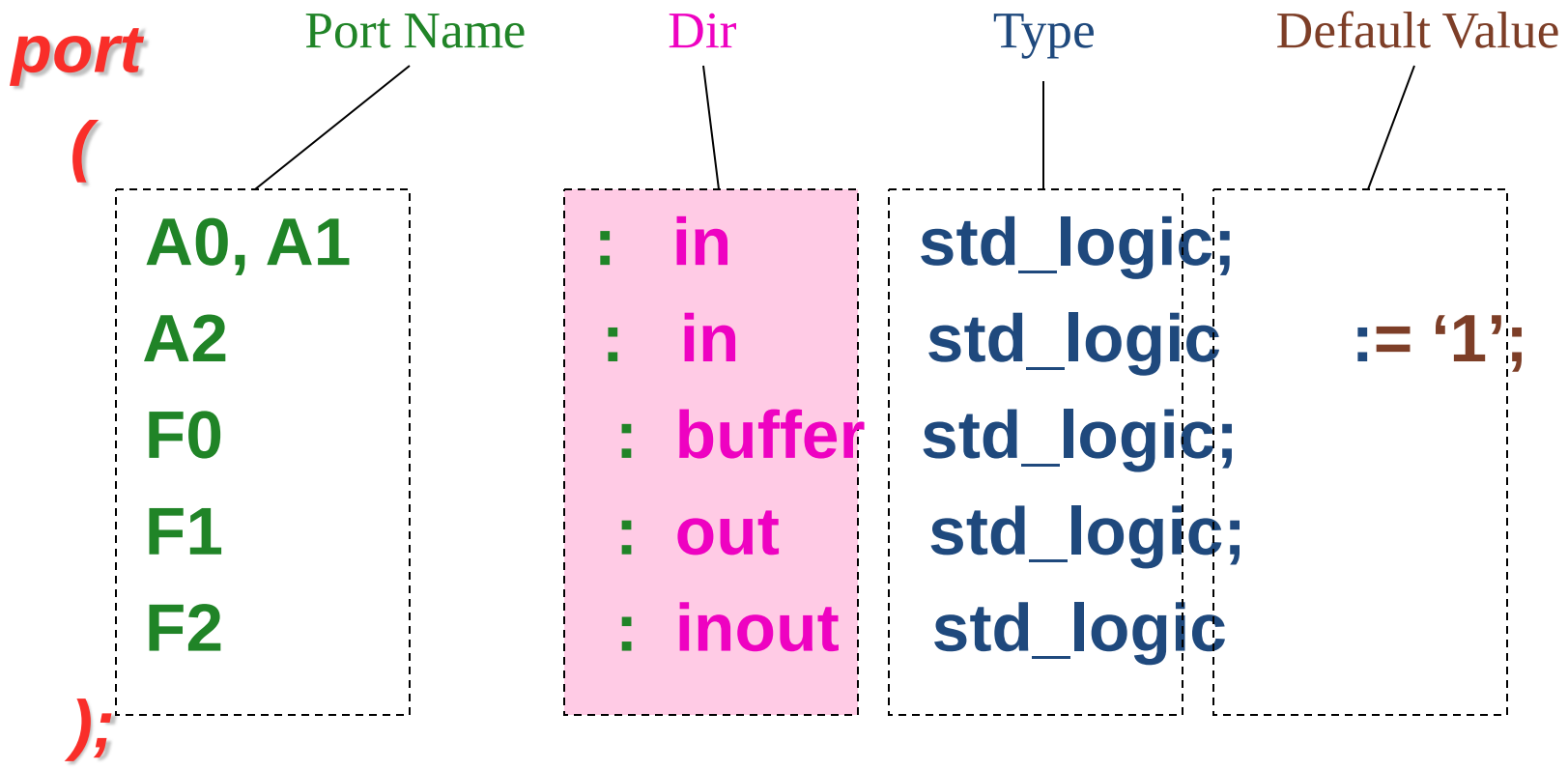
Entity Definition (Ports)

```
entity entity_name is
        Generics;
        Ports;
        Other Declarative Parts;
        Statements;
end [ entity ] [ entity_name ] ;
```

Port Definition

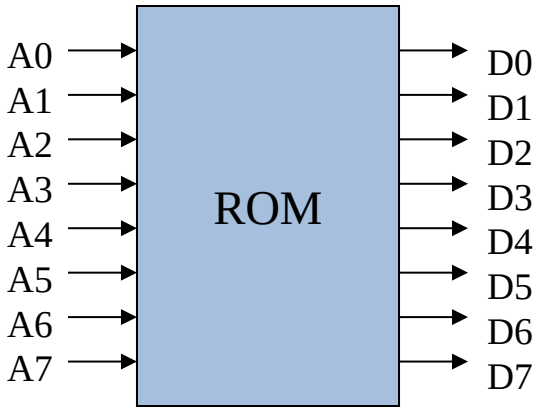
```
Port (
Port_Name[, Port_Name] : Dir   Type[:=Default_Val];
Port_Name[, Port_Name] : Dir   Type[:=Default_Val];
    .
    .
    .
Port_Name[, Port_Name] : Dir   Type[:=Default_Val]
);
```

Each Parts of Port



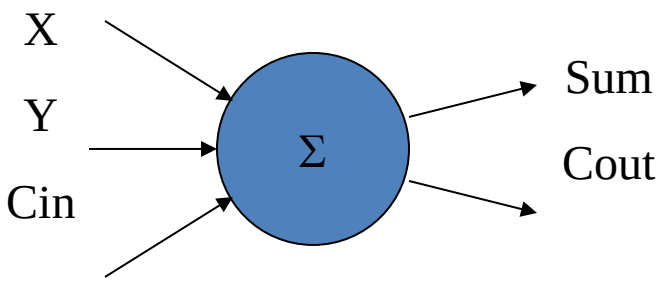
Port Examples (ROM)

```
entity ROM is
    port ( D0                  : out bit;
           D1                  : out bit;
           D2                  : out bit;
           D3                  : out bit;
           D4, D5, D6, D7      : out bit;
           A   : in bit_vector(7 down
    to 0) );
end ROM;
```



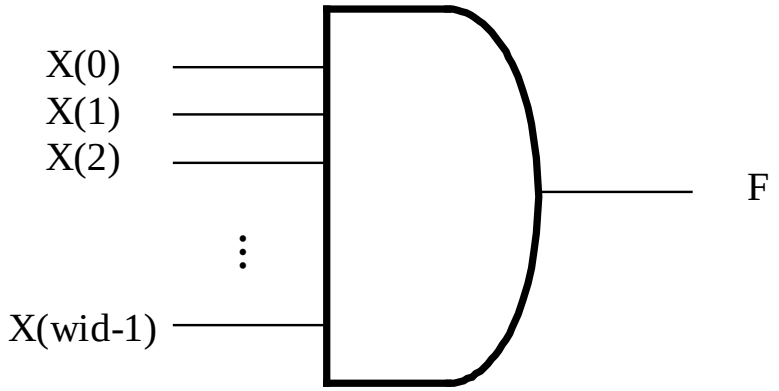
Port Examples (Adder)

```
entity Full_Adder is
   port (X, Y, Cin: in Bit;
         Cout, Sum: out Bit) ;
end  Full_Adder ;
```



Port Examples (n-input AND)

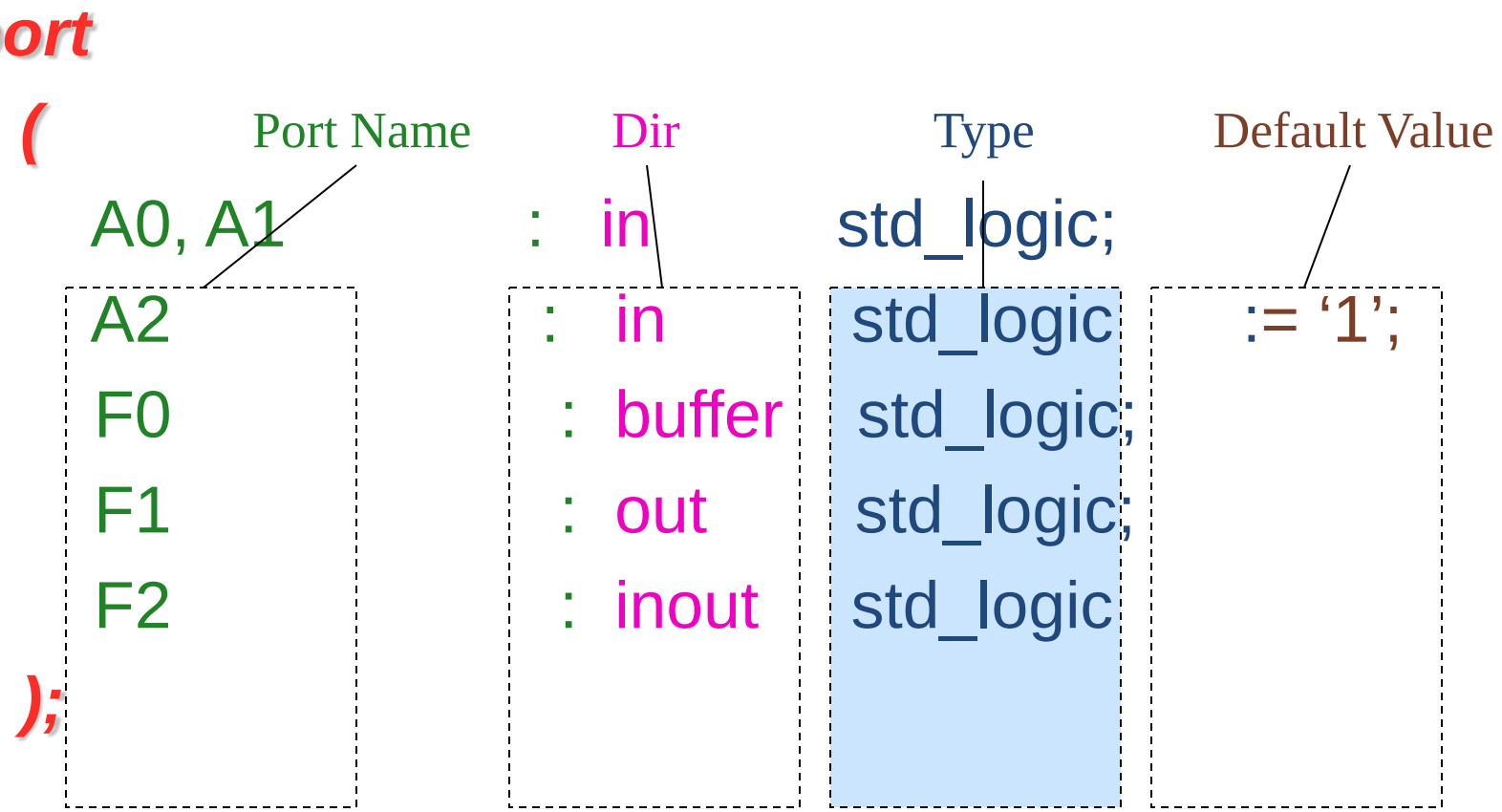
```
entity ANDN is
    generic (wid : integer := 2);
    port
    (
        X   :   in  bit_vector(wid-1 downto 0);
        F   :   out bit
    );
End ANDN;
```



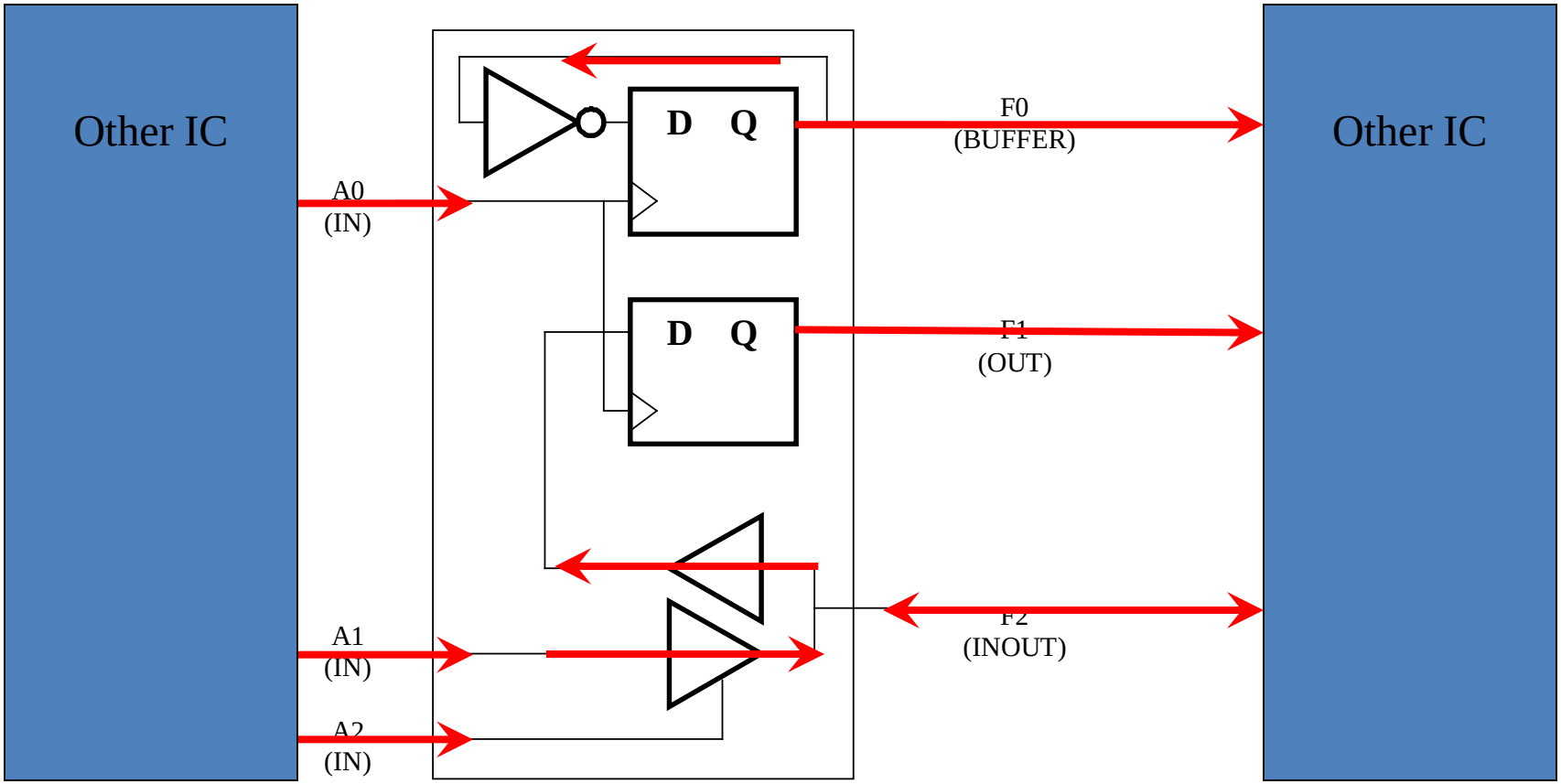
Type of “Dir”

- In
- Out
- Inout
- Buffer
- Linkage

Type

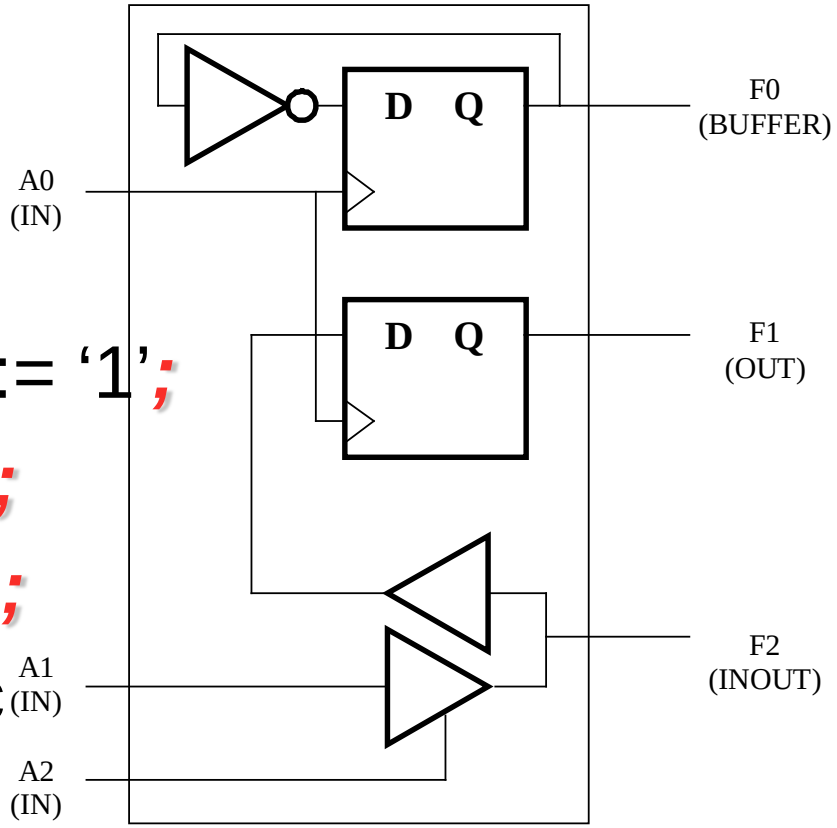

port
(
Port Name
Dir
Type
Default Value
A0, A1 : in std_logic;
A2 : in std_logic := '1';
F0 : buffer std_logic;
F1 : out std_logic;
F2 : inout std_logic
);

Signal Direction



Dir Example

```
port
(
  A0, A1 :   in      std_logic;
  A2     :   in      std_logic := '1';
  F0     :   buffer  std_logic;
  F1     :   out     std_logic;
  F2     :   inout   std_logic
);
```



Use of Dir

```
library ieee;
use ieee.std_logic_1164.all;
------------------------------
entity ABC is
  port
  (
    A0, A1, A2 : in     std_logic;
    F0         : buffer std_logic;
    F1         : out    std_logic;
    F2         : inout  std_logic
  );
end ABC;
```

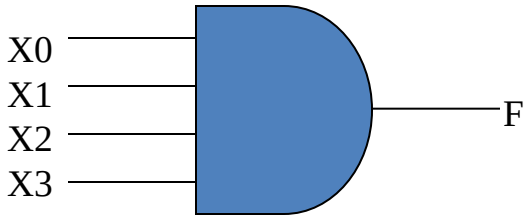
```
------------------------------
architecture ABC_arch of
ABC is
begin
  process(A0)
  begin
    if rising_edge(A0) then
      F0 <= not F0;
      F1 <= F2;
    end if;
  end process;
  F2 <= A1 when A2 = '1'
ELSE 'Z';
end ABC_arch;
```

Typical Port Type

- Bit
- Bit_vector
- Std_logic
- Std_logic_vector

- Bit
  - ‘1’
  - ‘0’

Bit_vector



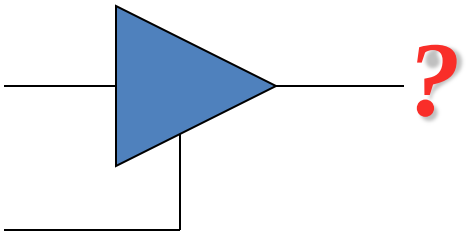

```
Port ( X0 : in bit;
       X1 : in bit;
       X2 : in bit;
       X3 : in bit;
       F  : out bit );
```

```
Port (
   X0, X1, X2, X3 : in bit;
   F  : out bit );
```

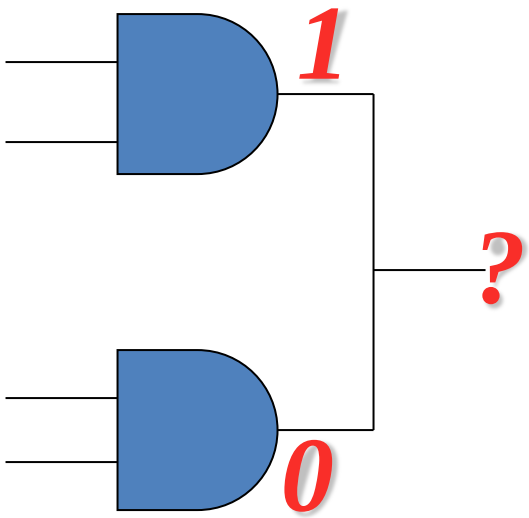
```
port
( X  :  in  bit_vector(3 downto 0);
  F  :  out bit );
```

Std_logic

- 'U', -- Uninitialized
- 'X', -- Forcing Unknown
- '0', -- Forcing 0
- '1', -- Forcing 1
- 'Z', -- High Impedance
- 'W', -- Weak   Unknown
- 'L', -- Weak   0
- 'H', -- Weak   1
- '-'   -- Don't care

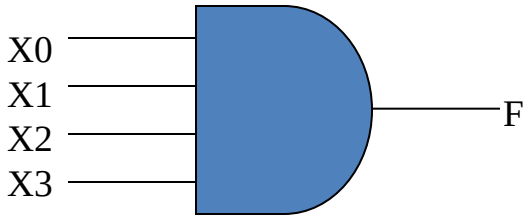


Resolution Function of Std_logic



	U	X	0	1	Z	W	L	H	-
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
-	U	X	X	X	X	X	X	X	X

Std_logic_vector




```
port
(  X   :   in  std_logic_vector(3 downto 0);
   F   :   out std_logic );
```

STD_LOGIC conversion functions

- VHDL Data Types
  - Boolean, integer, real
- STD_LOGIC conversion functions
  - TO_STDLOGICVECTOR(bit_vector)
  - CONV_STD_LOGIC_VECTOR(integer, bits)
  - CONV_INTEGER(std_logic_vector)

Function	Example:
TO_STDLOGICVECTOR(*bit_vector*)	**TO_STDLOGICVECTOR**(X"FFFF")
Converts a bit vector to a standard logic vector.	Generates a 16-bit standard logic vector of ones. "X" indicates hexadecimal and "B" is binary.
CONV_STD_LOGIC_VECTOR(*integer, bits*)	**CONV_STD_LOGIC_VECTOR**(7, 4)
Converts an integer to a standard logic vector.	Produces a standard logic vector of "0111".
CONV_INTEGER(*std_logic_vector*)	**CONV_INTEGER**("0111")
Converts a standard logic vector to an integer.	Produces an integer value of 7.

Signals, Time, and Simulation

- Variables vs. signals
  - VHDL variables change value without delay
  - VHDL signals have an associated delay
- A signal is given a value at a specific point in time, and retains that value until it is given a new value
  - A waveform is a sequence of values over time
  - Example: in1 <= ‘0’, ‘1’ after 5 ns, ‘0’ after 15 ns;
  - " A variable has a single value, whereas a signal has multiple value / time pairs
- A discrete event simulator executes VHDL code by advancing time to the next event, updating signal values, then possibly scheduling new events

VHDL Logic Operators

miscellaneous operators	** | abs | not
multiplying operators	* | / | mod | rem
sign operators	+ | -
adding operators	+ | - | &
shift operators	sll | srl | sla | sra | rol | ror
relational operators	= | /= | < | <= | > | >=
logical operators	and | or | nand | nor | xor | xnor

Example: entity of 4 bits adder

```
ENTITY add4 IS
    PORT(a, b: IN STD_LOGIC_VECTOR(3 downto 0);
         Ci:   IN STD_LOGIC;
         Sum: OUT STD_LOGIC_VECTOR(3 downto 0);
         Co: OUT STD_LOGIC);
END add4;
```

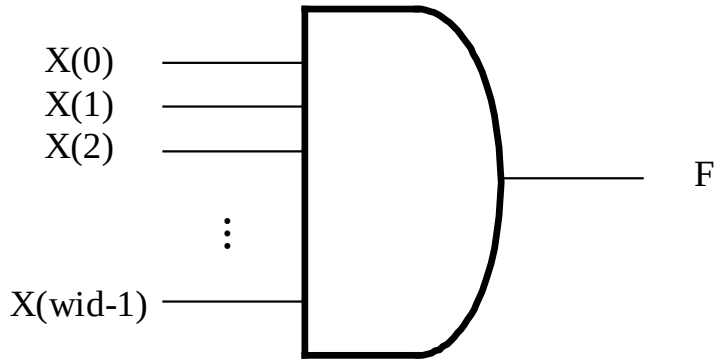
The diagram of add4 is as following:

Entity Definition (Generics)

```
entity entity_name is
    Generics;
    Ports;
    Other Declarative Parts;
    Statements;
end [ entity ] [ entity_name ] ;
```

An AND Gate With Unknown Inputs

```
entity ANDN is
    generic (wid : integer := 2);
    port
    (
        X   :   in  bit_vector(wid-1 downto 0);
        F   :   out bit
    );
end ANDN;
```



Generic Definition

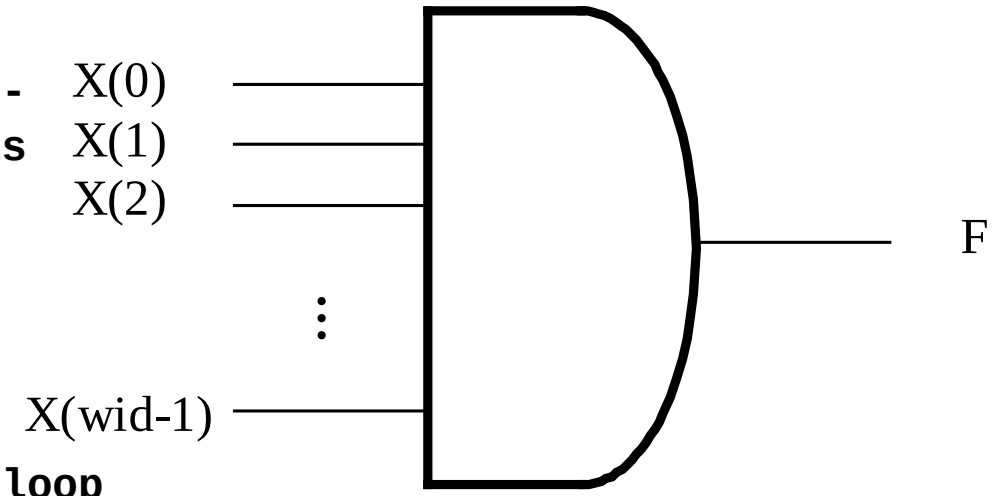
```
generic (
  Name [, Name] : DataType [:= DefaultValue];
  Name [, Name] : DataType [:= DefaultValue];
  ...
  Name [, Name] : DataType [:= DefaultValue]
);
```

Generic Example (1)

```
entity abcd is
    generic (
        p_a : integer : = 2;
        p_b : integer : = 7
    );
    port (
        A   :  out bit_vector(0 to p_a - 1);
        F    :  in bit
    );
End abcd;
```

Use of the Generic (ANDN.vhd)

```
library ieee;
use ieee.std_logic_1164.all;
---------------------------------
entity ANDN is
    generic (wid : integer := 2);
    port
    (
        X   :   in  bit_vector(wid-1 downto 0);
        F   :   out bit
    );
end ANDN;
---------------------------------
architecture ANDN_arch of ANDN is
begin
    process(X)
        variable tmp : bit;
    begin
        tmp := '1';
        for i in wid-1 downto 0 loop
            tmp := tmp and X(i);
        end loop;
        F <= tmp;
    end process;
end ANDN_arch;
```



Architecture

```
library ieee;
use ieee.std_logic_1164.all;
-----------------------------------------
entity XYZ is
    port
    (
        A, B, C          :   in  std_logic;
        F                :   out std_logic
    );
end XYZ;
-----------------------------------------
architecture XYZ_arch of XYZ is
begin
    F <= (A and B) or (B and C) or (C and A);
end XYZ_arch;
```

Include…

Entity

Architecture

Architecture Definition

architecture **arch_name** ***of*** **entity_name** ***is***

architecture_declarative_part

begin

architecture_statement_part

end [architecture] [**arch_name**] ;

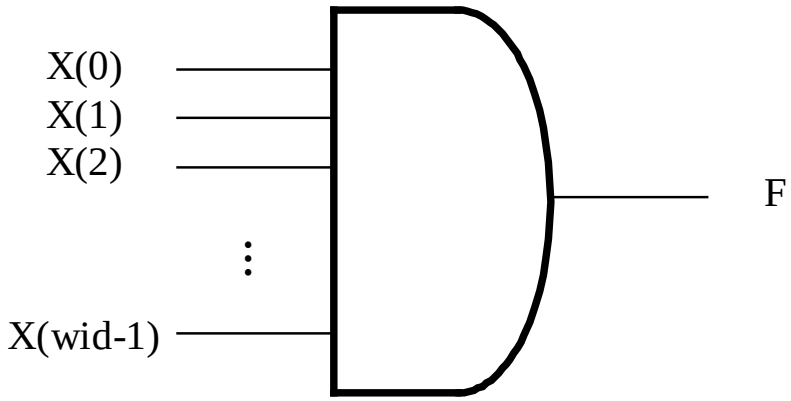
Architecture Example (ABC.vhd)

```
library ieee;
use ieee.std_logic_1164.all;
--------------------------------
entity ABC is
  port(
    A0,A1,A2 : in std_logic;
    F0   : buffer std_logic;
    F1   : out    std_logic;
    F2   : inout  std_logic );
end ABC;
```

```
--------------------------------
architecture ABC_arch of
ABC is
begin
  process(A0)
  begin
    if rising_edge(A0) then
      F0 <= not F0;
      F1 <= F2;
    end if;
  end process;
  F2 <= A1 when A2 = '1' ELSE 'Z';
end ABC_arch;
```

Architecture Example (ANDN.vhd)

```
library ieee;
use ieee.std_logic_1164.all;
---------------------------------
entity ANDN is
    generic (wid : integer := 2);
    port(
      X   :   in  bit_vector(wid-1 downto 0);
      F   :   out bit );
end ANDN;
---------------------------------
architecture ANDN_arch of ANDN is
begin
    process(X)
        variable tmp : bit;
     begin
        tmp := '1';
        for i in wid-1 downto 0 loop
            tmp := tmp and X(i);
        end loop;
        F <= tmp;
    end process;
end ANDN_arch;
```



Inside Architecture

- How to maintain large architecture?
- How to modulate the architecture code?
- Separate the architecture in to several parts
- How to separate the architecture?
- VHDL language that can separate an architecture
  - Process

Process Definition

[process_label:] ***process*** **[(sensitivity_list)]** [is]

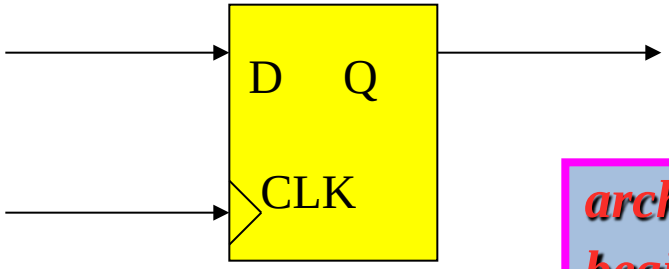
process_declarative_part

begin

process_statement_part

end process **[process_label];**

Process Example With Sensitive Table (MY_DFF.vhd)

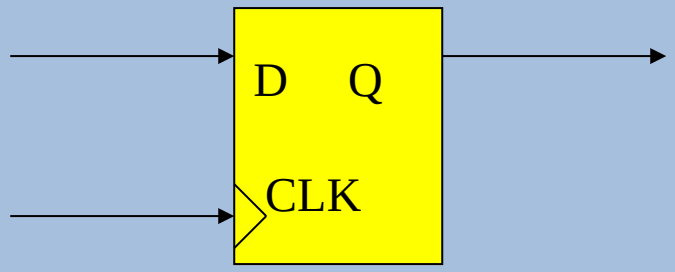



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use
    ieee.std_logic_unsigned.al
    l;
------------------------------
entity MY_DFF is
   port
   (  D, CP      :   in  std_logic;
      Q          :   out std_logic  );
end MY_DFF;
```

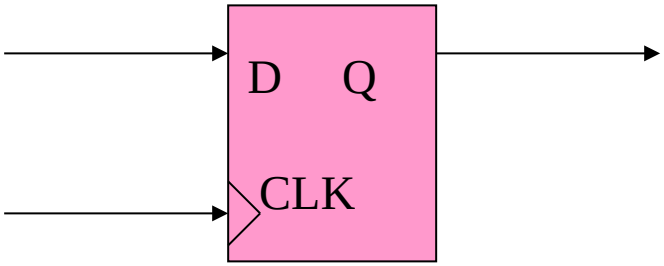
```
architecture MY_DFF_arch of MY_DFF is
begin
  process (CP)
  begin
    if  rising_edge(CP)  then
      Q <= D;
    end if;
  end process;
end MY_DFF_arch;
```

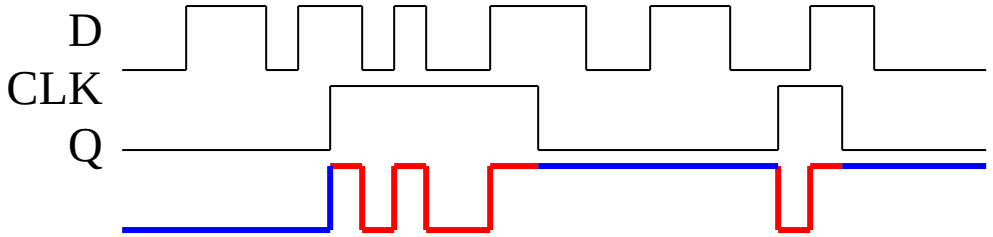
Process Example With ‘Wait’ (MY_DFF.vhd)

```
architecture MY_DFF_arch of MY_DFF is
begin
   process
  begin
      wait until CP'event and CP
   ='1';
       Q <= D;
   end process;
end MY_DFF_arch;
```



Process Example (Latch.vhd)






```
library ieee;
use
    ieee.std_logic_1164.all;
--------------------------------
entity LT is
   port
   (  D, CLK     :   in  bit;
      Q          :   out bit  );
end LT;
```

```
architecture LT_arch of LT is
begin
   process (CLK, D)
   begin
        if CLK = '1' then
             Q <= D;
        end if;
   end process;
end LT_arch;
```

Library & Use (Definition)

library LibraryName, LibraryName, …;

use LibraryName.PackageName.ItemName;

use LibraryName.PackageName.all;

use LibraryName.ItemName;

use LibraryName.all;

…

Library & Use

- Library
  - Name of library, contain pre-defined design items
  - Is treated as directory in most EDA tools
- Commonly used library
  - IEEE IEEE standard library
  - STD VHDL standard library
  - Work Used defined library

Library & Use (Example)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

library work;
use work.my_package.all;
```