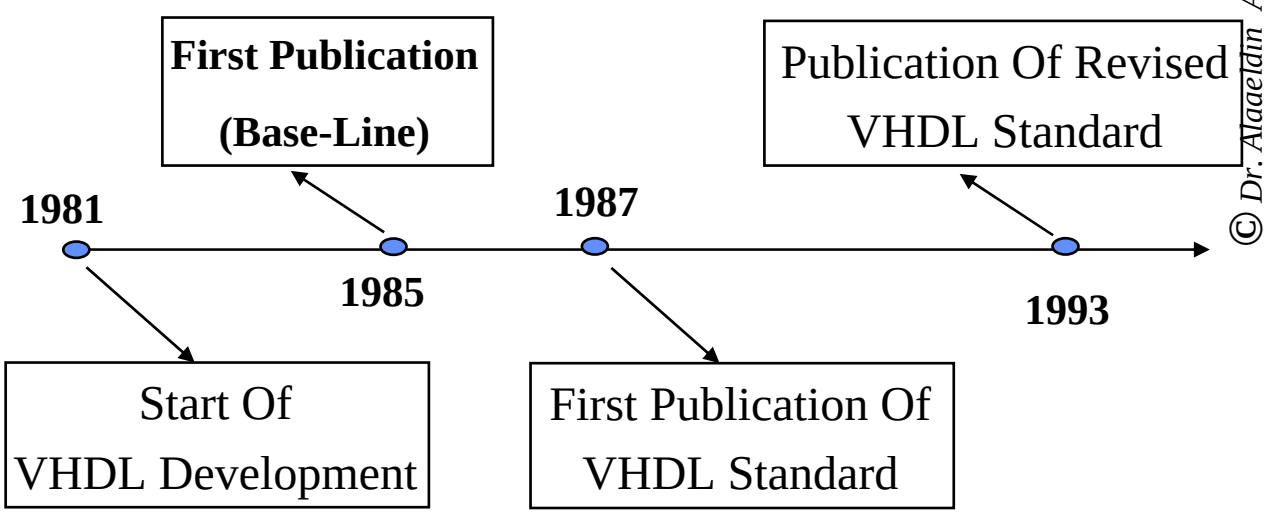
Hardware Modeling & Synthesis Using VHDL

- **VHDL** = **V**HSIC **H**ardware **D**escription **L**anguage

Very **H**igh **S**peed **I**ntegrated **C**ircuits



Background

- **<u>OBJECTIVES</u>: Provide a Standard Medium For**
  - Design Modeling
  - Design Documentation

- VHDL is an **IEEE Standard (Language Standard)**
  - IEEE VHDL-87
  - IEEE VHDL-93

- VHDL-93 is Upward Compatible with the VHDL-87 (Minor Differences May Require Code Modification)

- Requires **Simulation** and **Synthesis** Tools

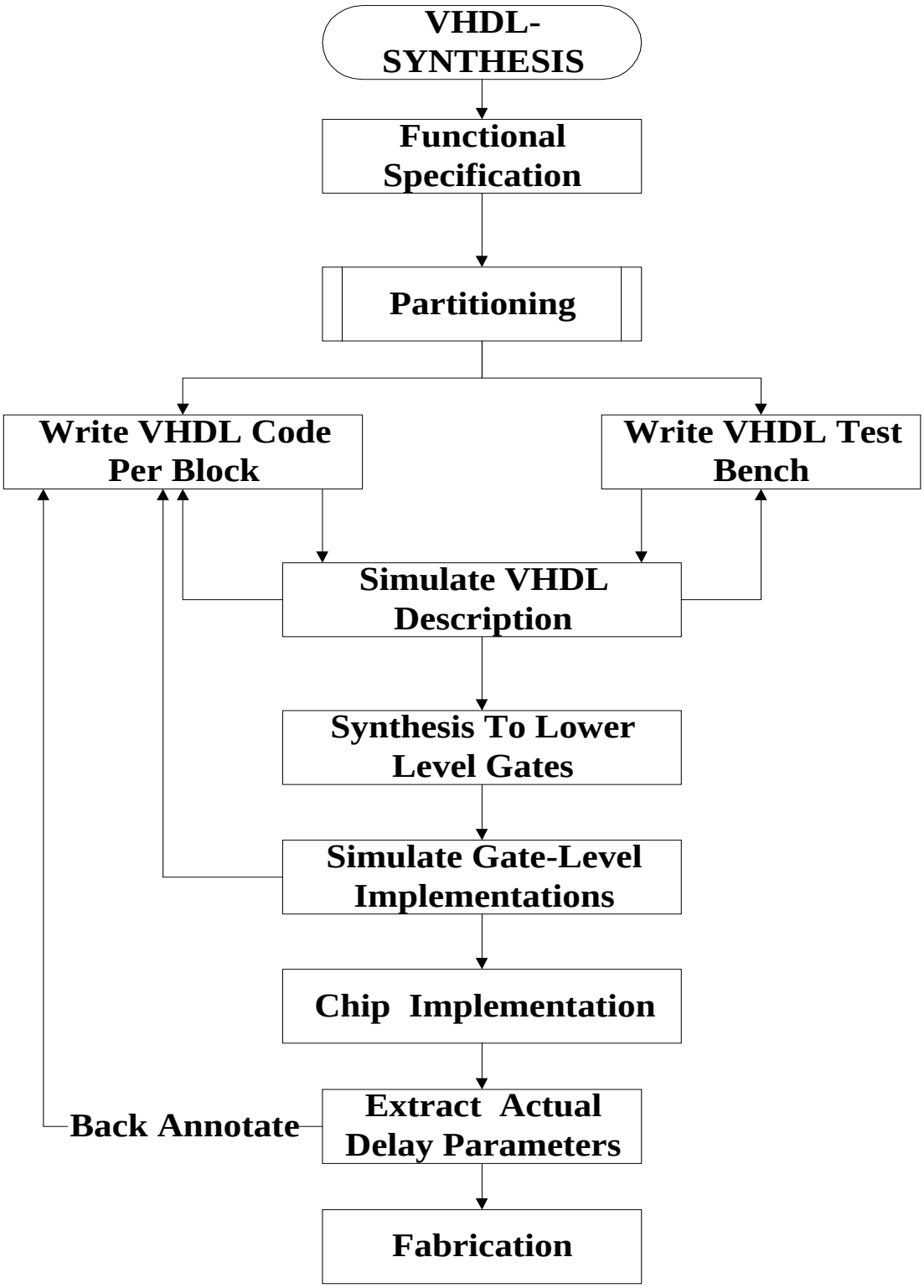
- By The End of 1995, Most Simulation Tools Have Incorporated Support for the VHDL-93

- VHDL Is Also Used For Design Synthesis

VHDL Advantages

- **Modular**
- **Hierarchical**, Allows Design Description :
  - TOP - DOWN,
  - BOTTOM - UP,
- **Portable**
- Can Describe the Same Design Entity using More than one View (Domain):
  - The Behavioral View (e.g. as an algorithm, Register-Transfer (Data Flow), Input-Output Relations, etc)
  - The Structural View.
- This Allows Investigation of *Design Alternatives* of the Same Entity
- It Also Allows *Delayed Detailed Implementations*.
- Can Model Systms at Various **Levels of Abstraction** (System, chip RTL, Logic (Gate))
- VHDL Can be Made to ***Simulate Timing*** At Reasonable Accuracy.

VHDL synthesis

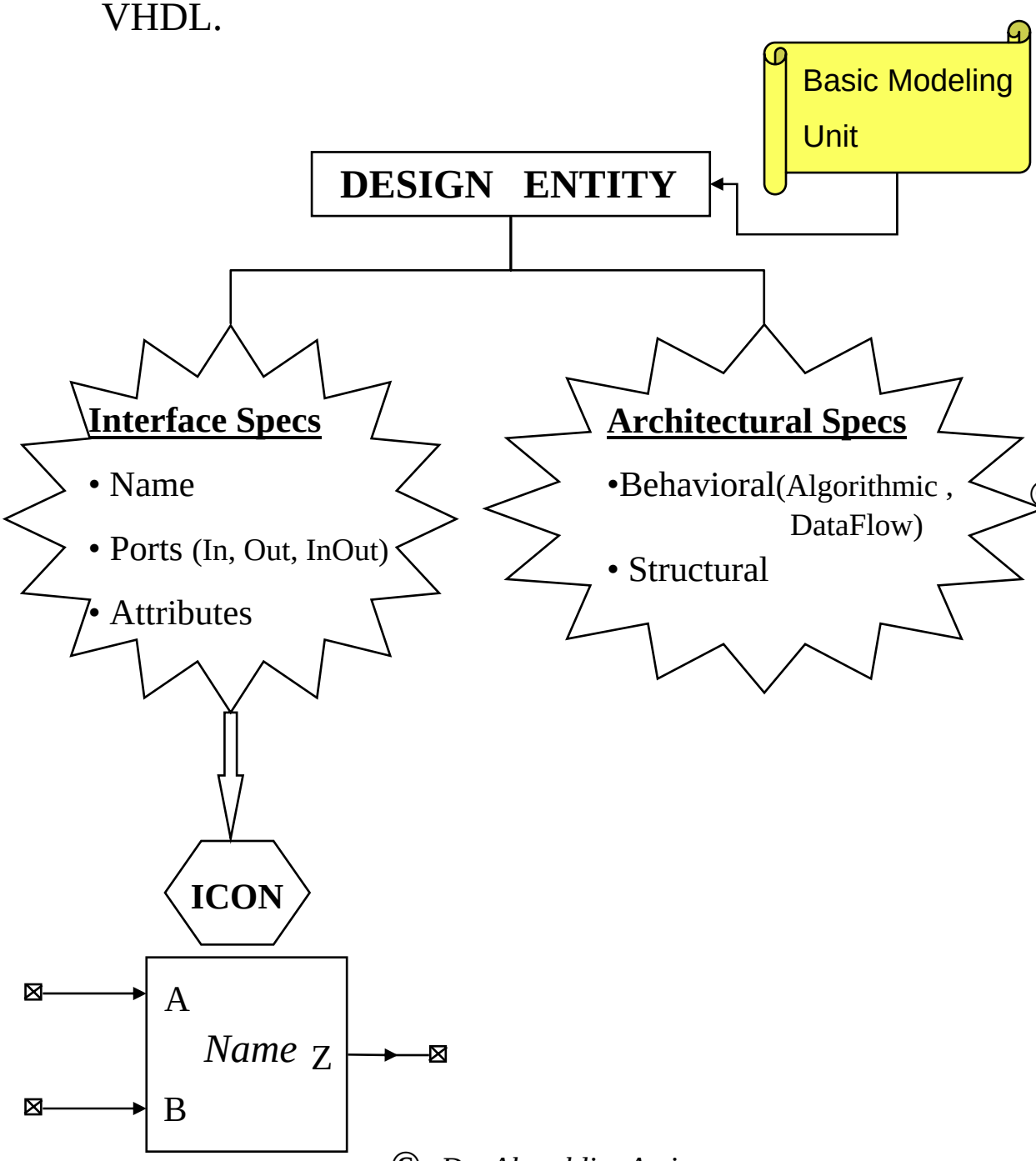


Hardware MOdeling using vhdl

- **VHDL** is **NOT** ***CaSe-SeNsItIvE*** **, Thus:**

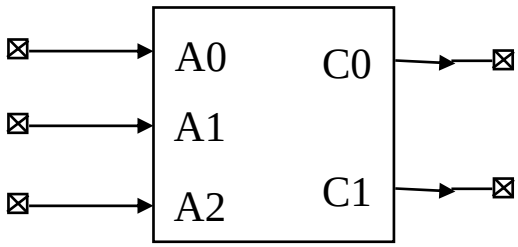
  Begin = begin = beGiN

- Semicolon “ **;** ” Terminates Declarations or Statements.
- Line Feeds and Carriage Returns are not Significant in VHDL.



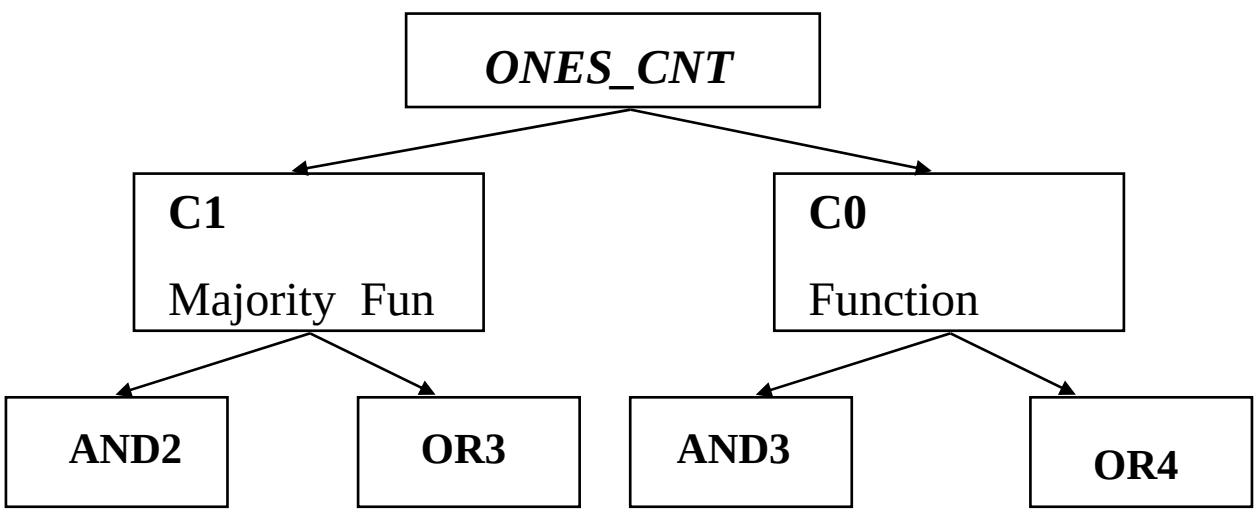
Example

"Ones Count CIRCUIT"

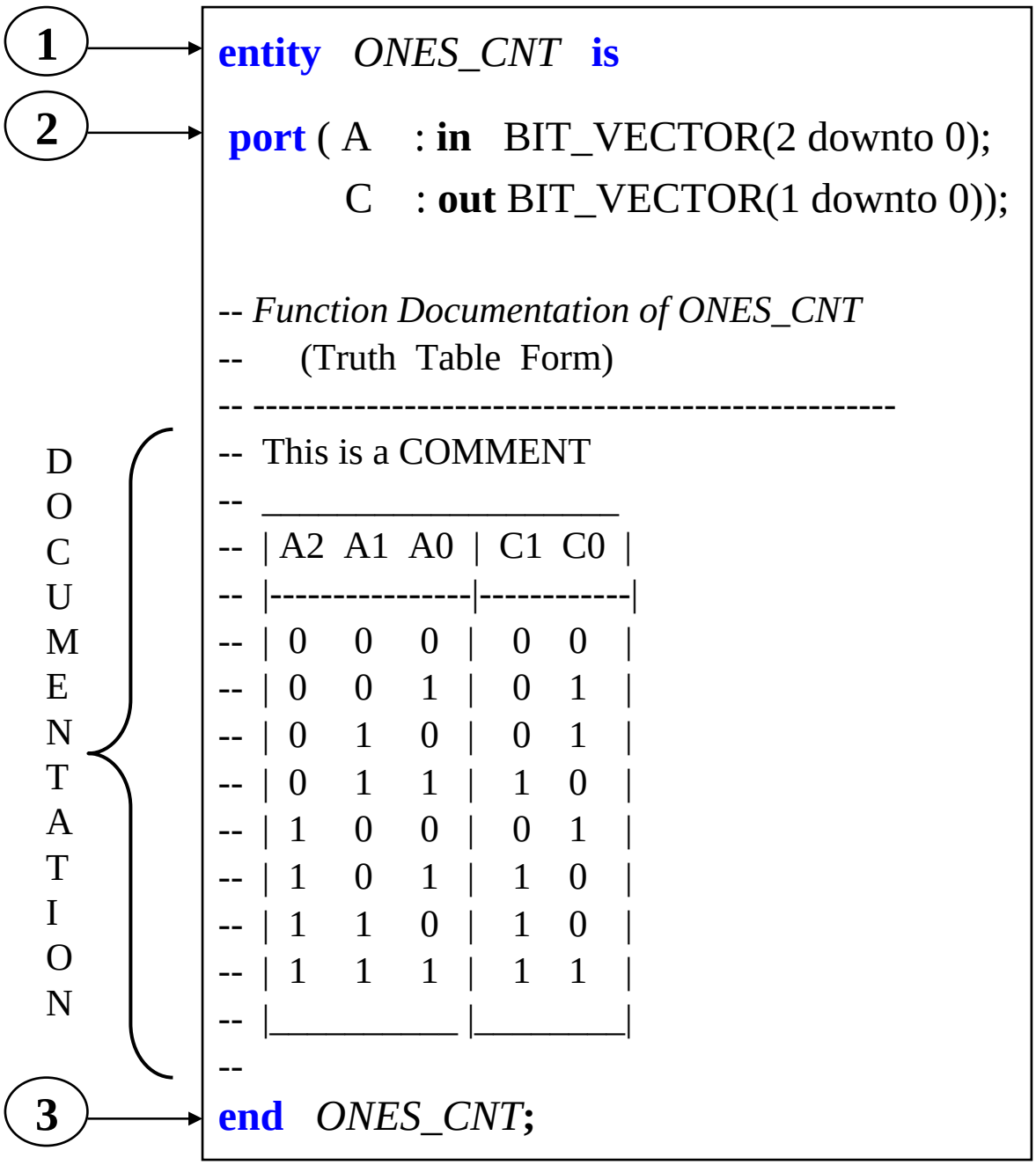


- Value of **C1 C0** = No. of Ones in the Inputs **A2, A1**, and **A0**
- **C1** is the Majority Function (=1 *IFF* Two or More Inputs =1)
- **C0** is a 3-Bit Odd-Parity Function (OPAR3))

- C1 = A1 A0 + A2 A0 + A2 A1
- C0 = A2 A1' A0' + A2' A1' A0 + A2 A1 A0 + A2' A1 A0'



Example "Ones Count CIRCUIT INTERFACE SPecs



example "Ones Count CIRCUIT
Architectural Body
(((((Behavioral view-1)))))

```
architecture Algorithmic of ONES_CNT is
begin

Process(A) -- Sensitivity List Contains only Vector A
    Variable num: INTEGER range 0 to 3;

    begin
        num :=0;
        For i in 0 to 2
        Loop
            IF A(i) = '1' then
                num := num+1;
            end if;
        end Loop;
--
--      Transfer "num" Variable Value to a SIGNAL
--
        CASE num is
            WHEN 0 => C <= "00";
            WHEN 1 => C <= "01";
            WHEN 2 => C <= "10";
            WHEN 3 => C <= "11";
        end CASE;
--
    end process;

end Algorithmic;
```

example “Ones Count CIRCUIT

Architectural Body
(Behavioral (Data Flow) view - 2)

- C1 = A1 A0 + A2 A0 + A2 A1
- C0 = A2 A1’ A0’ + A2’ A1’ A0 + A2 A1 A0 + A2’ A1 A0’

architecture *Two_Level* **of** *ONES_CNT* **is**

begin

```
--  C(1) <=(A(1) and A(0)) or (A(2) and A(0))
        or (A(2) and A(1));

    C(0) <= (A(2) and not A(1) and not A(0))
        or (not A(2) and not A(1) and A(0))
        or (A(2) and A(1) and A(0))
        or (not A(2) and A(1) and not A(0));
```

end *Two_Level*;

Example "Ones Count CIRCUIT

Architectural Body
(Behavioral (Data Flow) view - 3)
Using Functions

```
architecture Macro of ONES_CNT is

begin

    C(1) <= MAJ3(A);
--
    C(0) <= OPAR3(A);

end Macro ;
```


- Functions OPAR3 and MAJ3 Must Have Been Declared and Defined Previously

example “Ones Count CIRCUIT

Architectural Body
(((((Behavioral view -4)))))

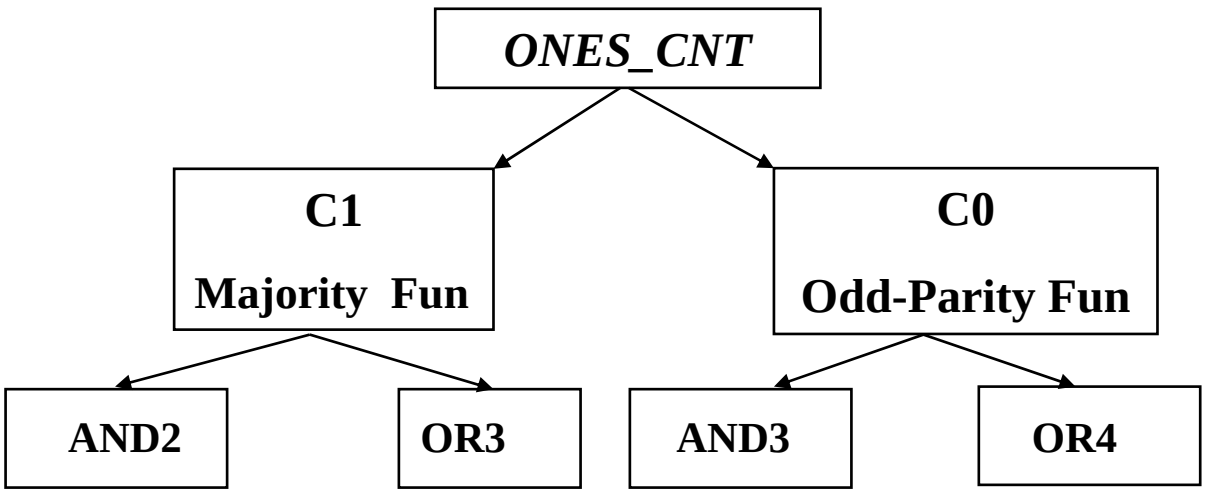
```
architecture Truth_Table of ONES_CNT is
begin
--
Process(A) -- Sensitivity List Contains only Vector A
    Variable num: BIT_VECTOR(2 downto 0);
    begin
        num :=A;
        CASE num is
            WHEN "000" => C <= "00";
            WHEN "001" => C <= "01";
            WHEN "010" => C <= "01";
            WHEN "011" => C <= "10";
            WHEN "100" => C <= "01";
            WHEN "101" => C <= "10";
            WHEN "110" => C <= "10";
            WHEN "111" => C <= "11";
        end CASE;
--
    end process;

end Truth_Table;
```

VHDL STRUCTURAL DESCRIPTION

"Ones Count CIRCUIT example "

- C1 = A1 A0 + A2 A0 + A2 A1 =**MAJ3**(A)
- C0 = A2 A1' A0' + A2' A1' A0 + A2 A1 A0 + A2' A1 A0' = **OPAR3**(A)

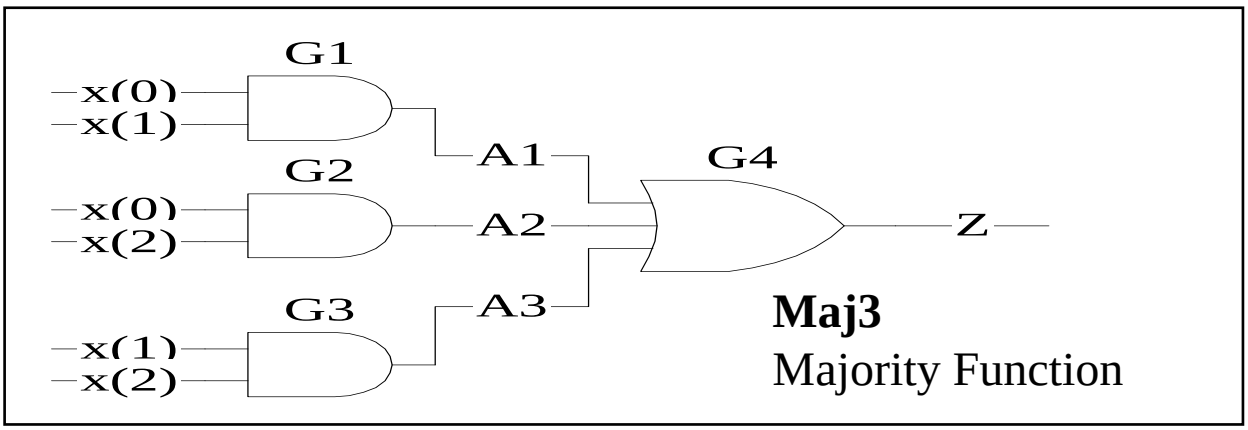
Structural Design Hierarchy




```
entity  MAJ3  is
    PORT( X: in BIT_Vector(2 downto 0);
          Z: out BIT);
end  MAJ3;
```

```
entity  OPAR3  is
    PORT( X: in BIT_Vector(2 downto 0);
          Z: out BIT);
end OPAR3;
```

VHDL STRUCTURAL DESCRIPTION




```
architecture  Structural  of  MAJ3  is
COMPONENT AND2
     PORT( I1, I2: in BIT;
           O: out BIT);
END COMPONENT;
COMPONENT OR3
     PORT( I1, I2, I3: in BIT;
           O: out BIT);
END COMPONENT;
--
SIGNAL  A1, A2, A3:   BIT;
begin
-- Instantiate Gates
     g1: AND2 PORT MAP (X(0), X(1), A1);
     g2: AND2 PORT MAP (X(0), X(2), A2);
     g3: AND2 PORT MAP (X(1), X(2), A3);
     g4: OR3   PORT MAP (A1, A2, A3, Z);
end Structural;
```

Declare Components To Be Instantiated

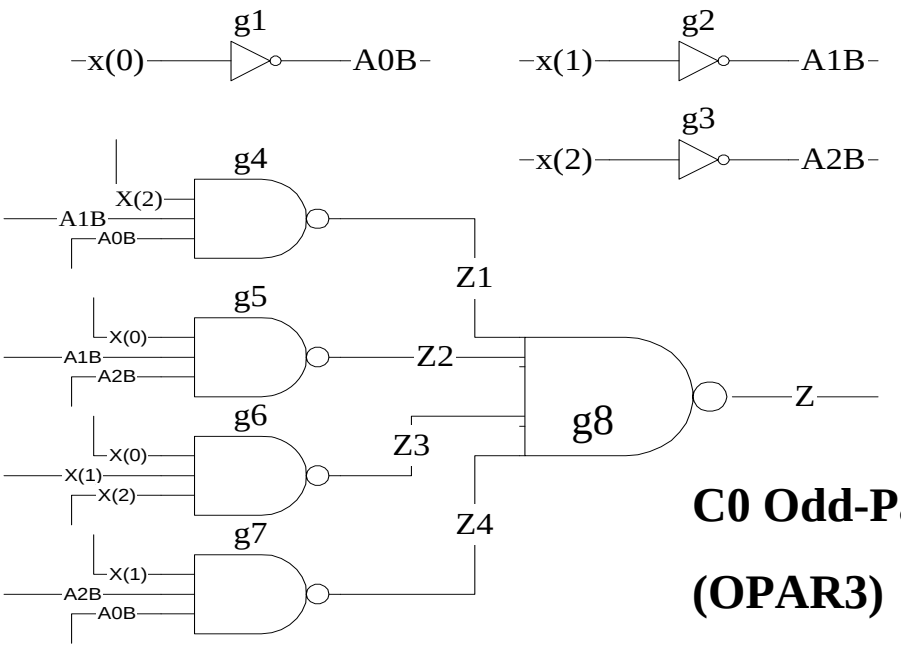
Declare Maj3 Local Signals

Wiring of Maj3 Compts.

```
entity OPAR3 is
    PORT( X: in BIT_Vector(2 downto 0);
          Z: out BIT);
end OPAR3;
--
architecture Structural of OPAR3 is
COMPONENT INV
    PORT(Ipt: in  BIT;
         Opt: out BIT);
END COMPONENT;
COMPONENT NAND3
    PORT( I1, I2, I3: in BIT;
          O: out BIT);
END COMPONENT;
COMPONENT NAND4
    PORT( I1, I2, I3, I4: in BIT;
          O: out BIT);
END COMPONENT;
--
SIGNAL A1B, A2B, A0B, Z1, Z2, Z3, Z4: BIT;
begin
-- Instantiate Gates
    g1: INV PORT MAP (X(0), A0B);
    g2: INV PORT MAP (X(1), A1B);
    g3: INV PORT MAP (X(2), A2B);
    g4: NAND3 PORT MAP (X(2), A1B, A0B, Z1);
    g5: NAND3 PORT MAP (X(0), A1B, A2B, Z2);
    g6: NAND3 PORT MAP (X(0), X(1), X(2), Z3);
    g7: NAND3 PORT MAP (X(1), A2B, A0B, Z4);
    g8: NAND4 PORT MAP (Z1, Z2, Z3, Z4, Z);
end Structural;
```

Interface Specs

Architecture Body




```
architecture  Structural  of  OPAR3  is
Component INV
    PORT( Ipt: in BIT; Opt: out BIT);
end Component ;
Component NAND3
    PORT( I1, I2, I3: in BIT;  O: out BIT);
end Component ;
Component NAND4
    PORT( I1, I2, I3, I4: in BIT; O: out BIT);
end Component ;
--
SIGNAL A1B, A2B, A0B, Z1, Z2, Z3, Z4: BIT;
begin
            g1: INV PORT MAP (X(0), A0B);
            g2: INV PORT MAP (X(1), A1B);
            g3: INV PORT MAP (X(2), A2B);
            g4: NAND3 PORT MAP (X(2), A1B, A0B, Z1);
            g5: NAND3 PORT MAP (X(0), A1B, A2B, Z2);
            g6: NAND3 PORT MAP (X(0), X(1), X(2), Z3);
            g7: NAND3 PORT MAP (X(1), A2B, A0B, Z4);
            g8: NAND4 PORT MAP (Z1, Z2, Z3, Z4, Z);
end Structural;
```

Top Structural level of ones_cnt

```
architecture Structural of ONES_CNT is
COMPONENT MAJ3
    PORT( X: in BIT_Vector(0 to 2);
          Z: out BIT);
END COMPONENT;
COMPONENT OPAR3
    PORT( X: in BIT_Vector(0 to 2);
          Z: out BIT);
END COMPONENT;
--
begin
-- Instantiate Components
--
    c1: MAJ3 PORT MAP (A, C(1));
    c2: OPAR3  PORT MAP (A, C(0));
end Structural;
```

Behavioral definition of lower level components

```
entity INV is
    PORT( Ipt: in BIT;  Opt: out BIT);
end INV;
--
architecture behavior of INV is
begin
    Opt <= not Ipt;
end behavior;
```

```
entity NAND2 is
    PORT( I1, I2: in BIT; O: out BIT);
end NAND2;
--
architecture behavior of NAND2 is
begin
    O <= not (I1 and I2);
end behavior;
```

Similarly Other Lower Level Gates Are Defined