Basic Structures

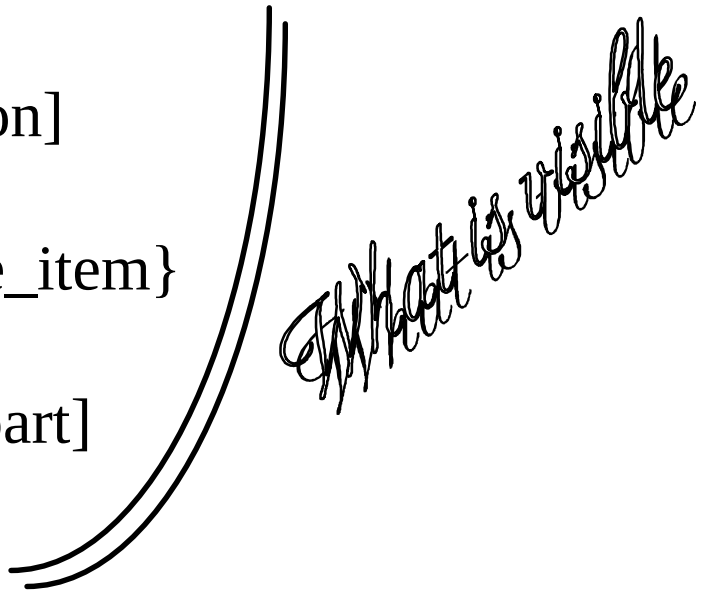
Basic Building Blocks

- **Entity**
- **Architecture**
- **Configuration**
- **Package**
- **Library**

Entity Declaration (1)

The External Aspect of a Design Unit

entity entity_name **is**
[generic_declaration]
[port_clause]
{entity_declarative_item}
[**begin**
entity_statement_part]
end [entity_name];



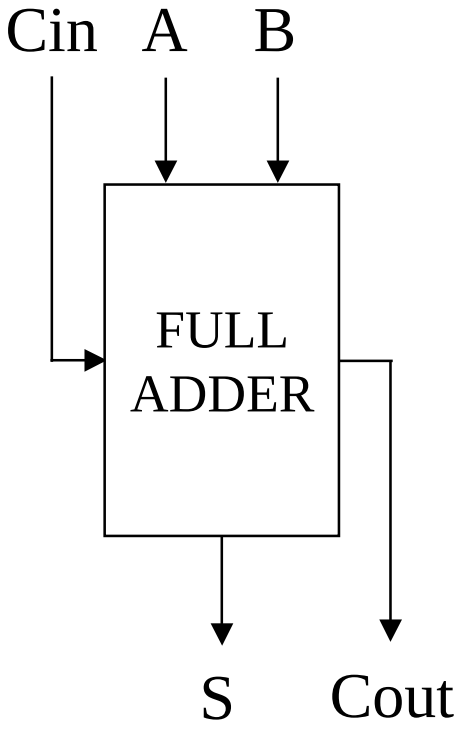
Entity Declaration (2)

Example

```
entity FULL_ADDER is
    port (A, B, Cin : in  BIT;
          S, Cout   : out BIT
    );
end FULL_ADDER;
```

MODE: **in**, **out**, **inout**...

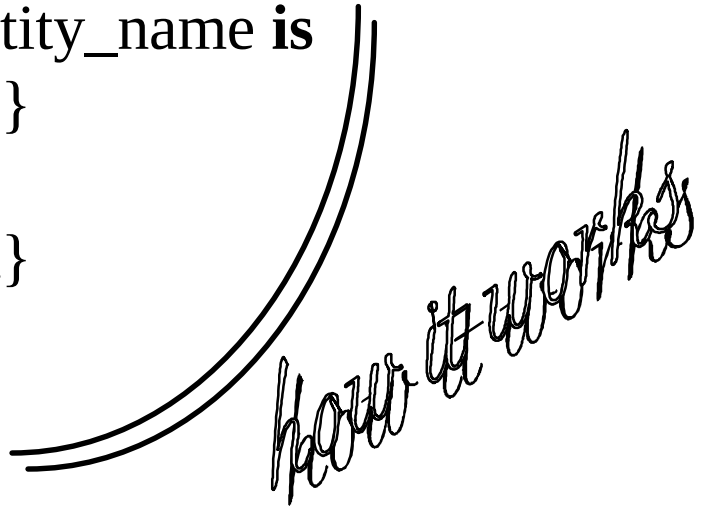
DATA TYPE



Architectures (1)

The Internal Aspect of a Design Unit

architecture architecture_name **of** entity_name **is**
{architecture_declarative_part}
begin
{architecture_descriptive_part}
end [architecture_name];



- **Collection of CONCURRENT Statements Executed in PARALLEL**
- **Concurrent Statements Communicate through SIGNALS**

Architectures (2)

A Behavioral Style

```
entity FULL_ADDER is
    port (A, B, Cin : in   BIT;
          S, Cout   : out BIT);
end FULL_ADDER;
architecture DATAFLOW of FULL_ADDER is
    signal X : BIT;
begin
    X    <= A xor B;
    S    <= X xor Cin after 10 ns;
    Cout <= (A and B) or (X and Cin) after 5 ns;
end DATAFLOW;
```

Architectures (3)

A Structural Style

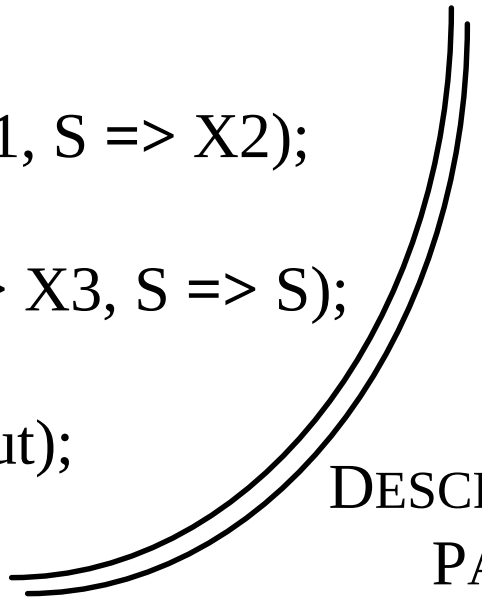
```
architecture STRUCTURE of FULL_ADDER is
    component HALF_ADDER
        port (I1, I2      : in  BIT;
              Carry, S    : out BIT);
    end component;

    component OR_GATE
        port (I1, I2      : in  BIT;
              O           : out BIT);
    end component;
    signal X1, X2, X3 : BIT;
```

DECLARATIVE PART

Architectures (4)

A Structural Style

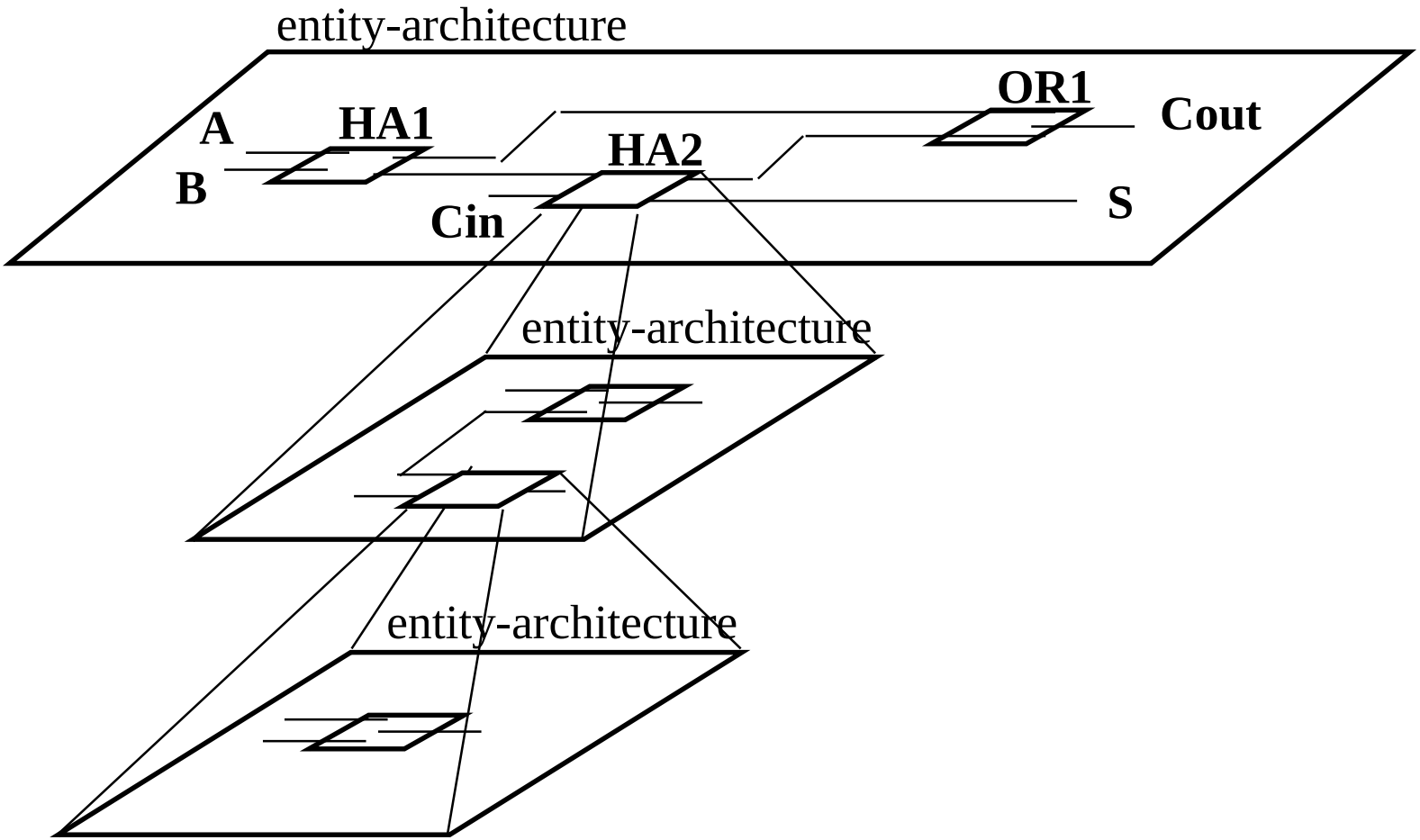
```
begin
    HA1  : HALF_ADDER port map (
             I1 => A, I2 => B, Carry => X1, S => X2);
    HA2  : HALF_ADDER port map (
             I1 => X2, I2 => Cin, Carry => X3, S => S);
    OR1  : OR_GATE    port map (
             I1 => X1, I2 => X3, O => Cout);
end STRUCTURE ;
```



DESCRIPTIVE PART

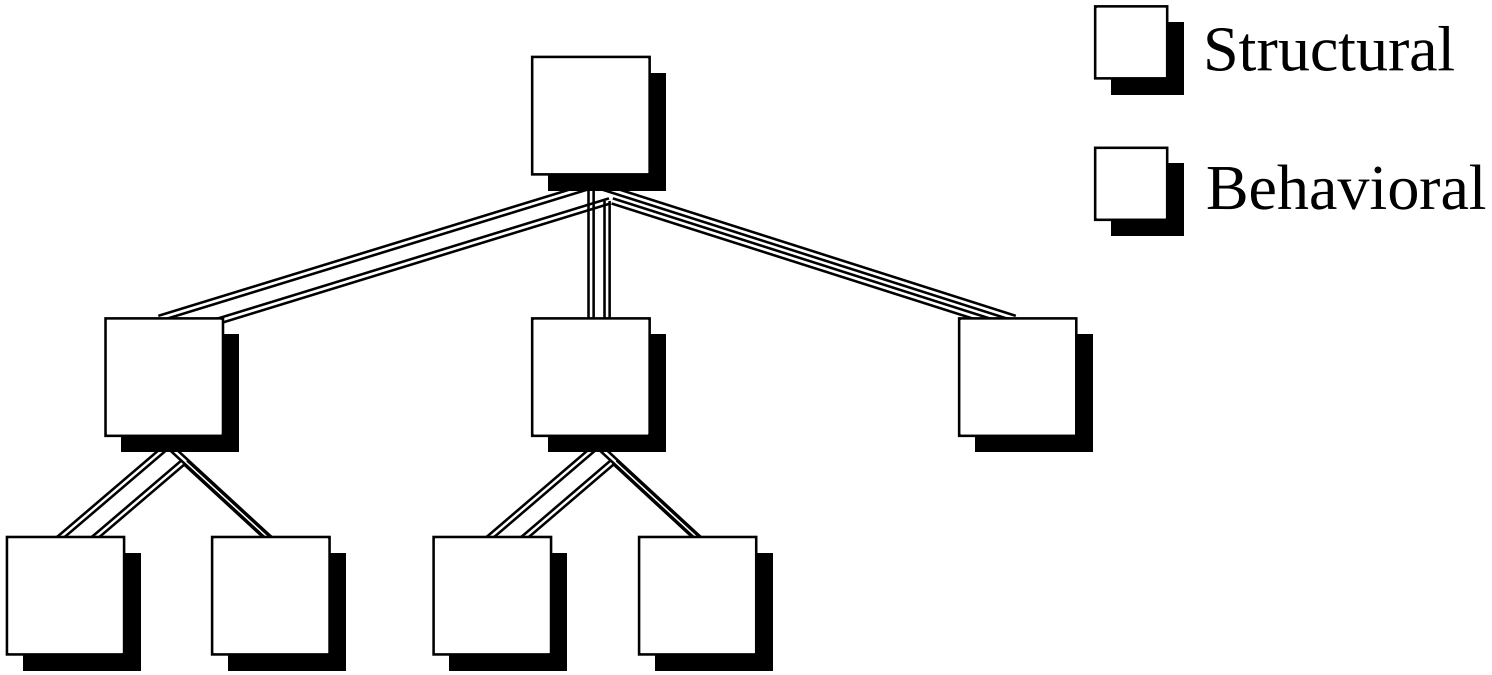
Architectures (5)

Structural Style to represent Hierarchy



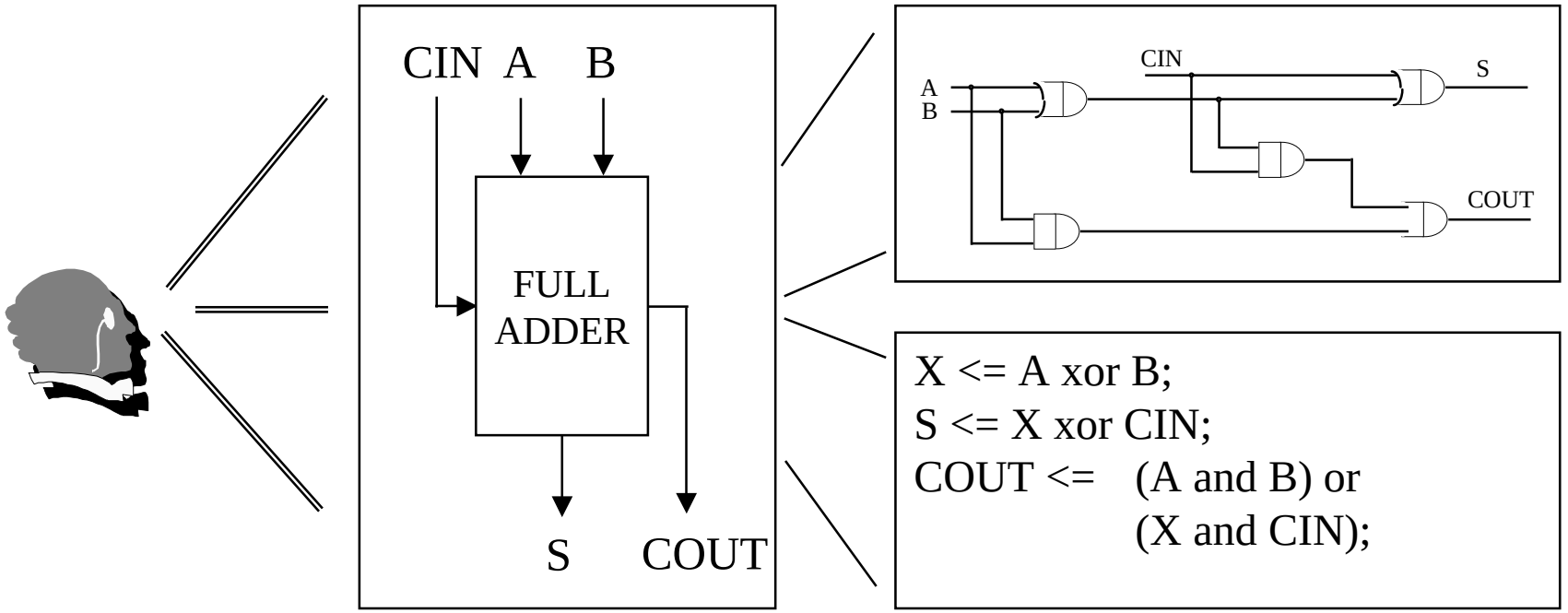
Architectures (6)

Structure & Behavior in a Design Tree



Architectures (7)

entity/architecture: a One to Many Relationship



Configurations (1)

Specification Inside the Architecture Body

for instantiation_list: component_name **use** binding_indication;

use library_name.entity_name [(architecture_name)];

- **Binding a couple "entity/architecture" to each instance**

Configurations (2)

Declaration as a Separate Design Unit

configuration configuration_name **of** entity_name **is**
    for { architecture | component } binding_indication;
end [configuration_name];

- **Can be compiled separately and stored in a library**
- **It defines a configuration for a particular entity**

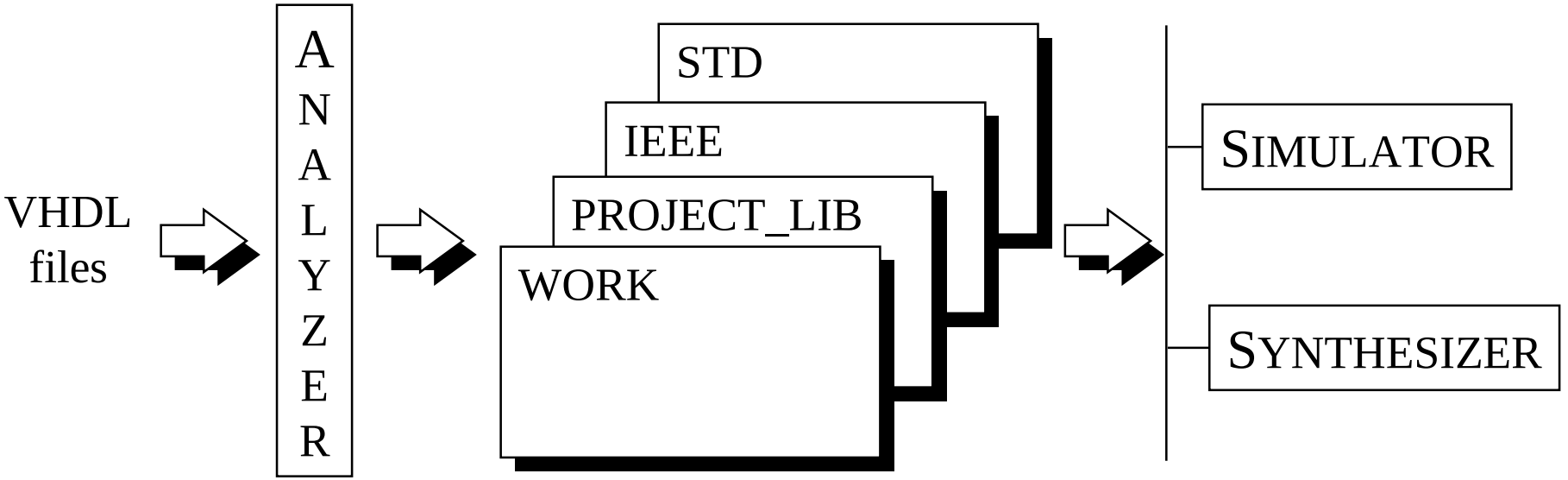
Packages

Global Design Unit

package package_name **is**
{package_declarative_item}
end [package_name];
package body package_name **is**
{package_body_declarative_item}
end [package_name];

- **Same declarations visible by a number of design entities**
- **May contain subprograms, components, signals, ...**

Design Libraries




```
library library_name ;
use library_name.package_name.all;
```

- **May contain: packages, entities, architectures, configurations**

Data Objects (1)

Three Classes

- **Constants**
  - **Initialized to a specific value and <u>never</u> modified**

    constant MSB : INTEGER := 5;

- **Variables**
  - **Used to hold temporary data**
  - **<u>Only</u> used within processes & subprograms**

    variable DELAY : INTEGER range 0 to 15 := 0;

Data Objects (2)

Three Classes

- **Signals**
  - **Used to communicate between processes**
  - **When declared in a package : Global Signals**
  - **Also declared within entities, blocks, architectures**
  - **Can be used but not defined in processes and subprograms**

```
signal CLK : BIT;
```

Data Types (0)

Concept

- Each signal must have a data type associated with it when the signal is declared
- A type defines a set of values
- An assignment must always be of a value defined by that set
- Types in either sides of the assignment must match

Data Types (1)

Enumeration Types

- **The first identifier is the default value**

```
type COLOR is (RED, ORANGE, YELLOW);
type TERNARY is ( '1', '0', 'X' );
variable X : COLOR;
signal Y : TERNARY;
```

Data Types (2)

Integer Types

- **The range must be specified**
- **No logical operations on integer**

type MEMORY_SIZE is range 1 to 2048;

Data Types (3)

Predefined VHDL Data Types
IEEE 1076-1987 Standard Package

- **BOOLEAN : (false , true)**
- **BIT : ('0', '1')**
- **CHARACTER**
- **INTEGER : range -2 147 483 647 to +2 147 483 647**
- **NATURAL : Subtype of INTEGER (Non Negative)**
- **POSITIVE : Subtype of INTEGER (positive)**
- **BIT_VECTOR : array of BIT values**
- **STRING : array of CHARACTERS**
- **REAL : range -1.0E+38 to +1.0E+38**
- **TIME : Physical type used for simulation**

Data Types (4)

Array Types

- **Constrained Array**

```
type VEC_64 is array (0 to 63) of INTEGER;
variable S : VEC_64;
variable S1 : INTEGER;
S1 := S (1);
```

- **Unconstrained Array**

```
type BIT_VECTOR is array (POSITIVE range <>) of BIT;
signal S : BIT_VECTOR (4 downto 0);
```

- **Multiple Dimentional Arrays**

```
type TWO_D is array (0 to 7, 0 to 3) of INTEGER;
```

Data Types (5)

Array Assignements : by POSITION

signal z_bus : bit_vector (3 downto 0);
signal c_bus : bit_vector (1 to 4);
signal A, B, C, D : BIT;

- **z_bus <= c_bus;**
- **z_bus (3 downto 2) <= “00”;**
- **c_bus (2 to 4) <= z_bus (3 downto 1);**
- **z_bus (0 to 1) <= “00”;**
- **z_bus <= A & B & C & D;** **- - concatenate**
- **z_bus <= (A , B , C , D);** **- - aggregates**
- **z_bus <= (3=>’1’, 1 downto 0 => ’1’, 2 => B); - - by name**
- **z_bus <= (3=>’1’, 1 => ’0’, others => B); - - synthesis**

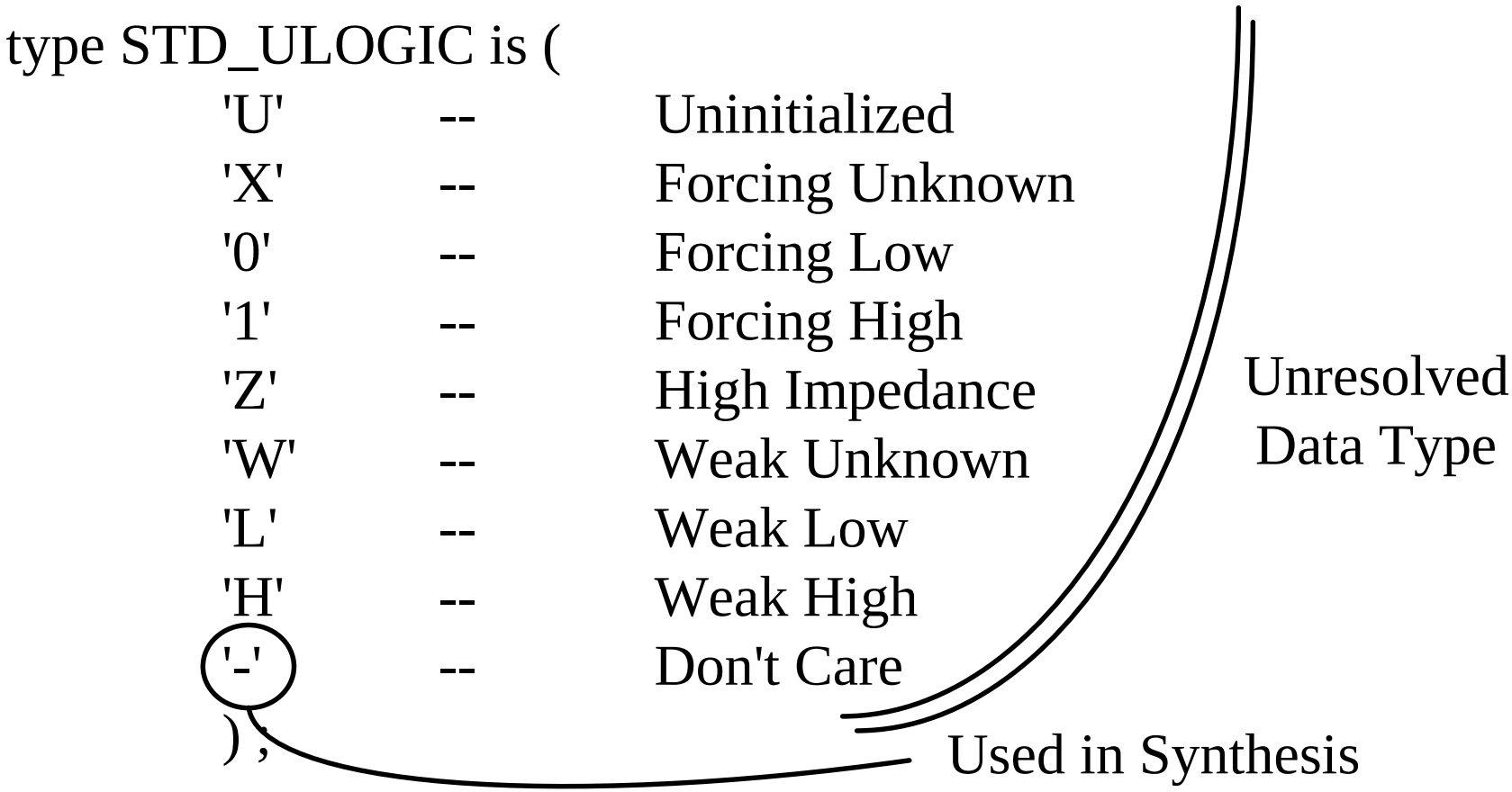
Data Types (6)

Record Types

```
type DATE is
        record
                YEAR : INTEGER range 1900 to 1999 ;
                MONTH : INTEGER range 1 to 12 ;
                DAY : INTEGER range 1 to 31 ;
        end record ;
signal S : DATE;
variable Y : INTEGER range 1900 to 1999 ;
Y := S.YEAR ;
```

Data Types (7)

STD_LOGIC Data Types
IEEE 1164-1993 Standard Logic Package



Data Types (8)

STD_LOGIC Data Types
IEEE 1164-1993 Standard Logic Package

- **STD_LOGIC : Resolved (Resolution Function provided)**
- **STD_LOGIC_VECTOR**
- **STD_ULOGIC_VECTOR**

library ieee;
use ieee.std_logic_1164.**all**;

Data Types (9)

Also,

- **FILE : Useful for RAM Values or Stimuli Files**
- **ACCESS : Like "pointers" in High Level Languages**
- **TEXT : FILE of STRING (TEXTIO package)**
- **LINE : access STRING (TEXTIO package)**

Subtypes

Subsets of Other Types

- **To Insure Valid Assignments**
- **Inherit All Operators and Subprograms from the Parent Type**

```
subtype DIGIT is INTEGER range 0 to 9;
```

Operators

Six Classes

LOGIC OPERATOR	and , or , nand , nor , xor
RELATIONAL OPERATOR	= , /= , < , <= , > , >=
ADDING OPERATOR	+ , - , &
SIGN	+ , -
MULTIPLYING OPERATOR	* , / , mod , rem
MISCELLANEOUS OPERATOR	** , abs , not

PRECEDENCE ORDER

Operands (1)

- **Literals : 'x' , "1100" , 752 , B"11001" , O"277" , X"4C"**
  - **numeric, character, enumeration, or string**
- **Identifiers :**
  - **starts with (a-z) followed by letters, '_', or digits**
  - **Not case-sensitive**
  - **Some are reserved words**
- **Indexed Names : S (3) , DATA (ADDR)**

Operands (2)

- **Slice Names : variable ORG : BIT_VECTOR (7 downto 0)**
  - ✧ **Sequence of elements of an array object**
- **Aliases : alias MSB : BIT is ORG (7)**
  - ✧ **New name for a part of a range of an array**
- **Aggregates**
- **Qualified Expressions**
- **Function Calls**
- **Type Conversions**

Operands (3)

Attributes Names
A Data Attached to VHDL Objects

- **S'LEFT : Index of the leftmost element of the data type**
- **S'RIGHT : Index of the rightmost element of the data type**
- **S'HIGH : Index of the highest element of the data type**
- **S'LOW : Index of the lowest element of the data type**
- **S'RANGE : Index range of the data type**
- **S'REVERSE_RANGE : Reverse index range**
- **S'LENGTH : Number of elements of an array**

Operands (4)

Attributes Names
A Data Attached to VHDL Signals

- **S'EVENT : A change value at the current simulation time**
- **S'STABLE : No change value at the current simulation time**

  if (CK = 0 and not CK'STABLE)

- **.....**