Topics to be covered

- Hardware description languages
- VHDL terms
- Design Entity
- Design Architecture
- VHDL model of full adder circuit

Topics to be covered

- Test Bench example
- VHDL objects
- Variables vs. Signals
- Signal assignment & Signal attributes
- Subprograms, Packages, and Libraries
- Data types in VHDL

Hardware Description Languages

- HDLs are used to describe the hardware for the purpose of modeling, simulation, testing, design, and documentation.
  - Modeling: behavior, flow of data, structure
  - Simulation: verification and test
  - Design: synthesis
- Two widely-used HDLs today
  - **VHDL:** VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
  - **Verilog** (from Cadence, now IEEE standard)

Applications of HDL

- Model and document digital systems
  - Different levels of abstraction
    - Behavioral, structural, etc.
- Verify design
- Synthesize circuits
  - Convert from higher abstraction levels to lower abstraction levels

Modeling

- Behavioral
  - High level, algorithmic, sequential execution
  - Easy to write and understand (like high-level language code)
- Dataflow
  - Medium level, register-to-register transfers,
  - Harder to write and understand (like assembly code)
- Structural
  - Low level, net list, component instantiations and wiring
  - Hardest to write and understand (very detailed and low level)

VHDL Terms …

- Entity:
  - All designs are expressed in terms of entities
  - Basic building block in a design
- Ports:
  - Provide the mechanism for a device to communication with its environment
  - Define the names, types, directions, and possible default values for the signals in a component's interface
- Architecture:
  - All entities have an architectural description
  - Describes the behavior of the entity
  - A single entity can have multiple architectures (behavioral, structural, …etc)
- Configuration:
  - A configuration statement is used to bind a component instance to an entity-architecture pair.
  - Describes which behavior to use for each entity

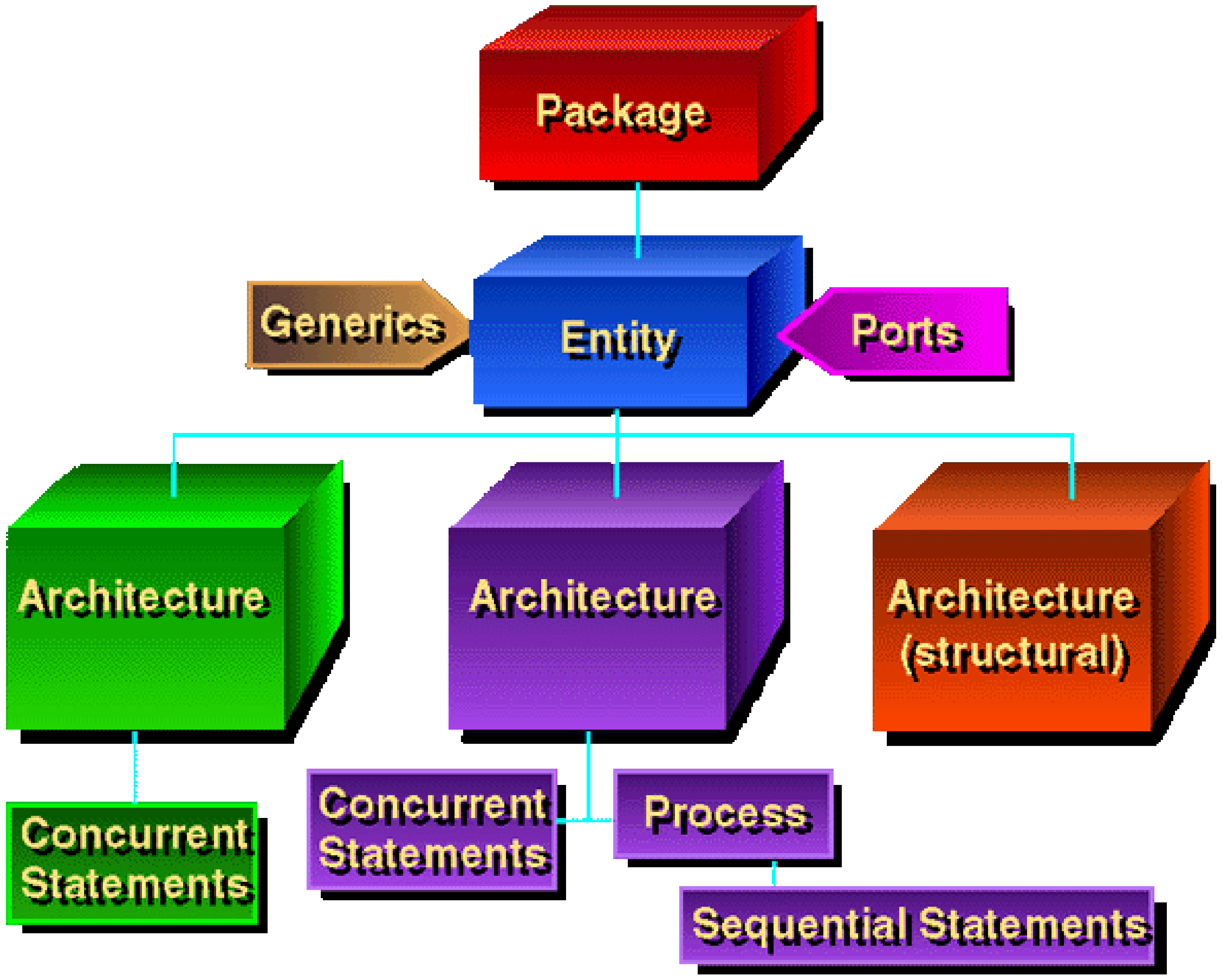
… VHDL Terms …

- Generic:
  - A parameter that passes information to an entity
  - Example: for a gate-level model with rise and fall delay, values for the rise and fall delays passed as generics
- Process:
  - Basic unit of execution in VHDL
  - All operations in a VHDL description are broken into single or multiple processes
  - Statements inside a process are processed sequentially
- Package:
  - A collection of common declarations, constants, and/or subprograms to entities and architectures.

VHDL Terms …

- Attribute:
  - Data attached to VHDL objects or predefined data about VHDL objects
  - Examples:
    - maximum operation temperature of a device
    - Current drive capability of a buffer
- VHDL is NOT Case-Sensitive
  - Begin = begin = beGiN
- Semicolon " ; " terminates declarations or statements.
- After a double minus sign (--) the rest of the line is treated as a comment

VHDL Models …


Package
Generics
Entity
Ports
Architecture
Architecture
Architecture (structural)
Concurrent Statements
Concurrent Statements
Process
Sequential Statements

Existing Packages

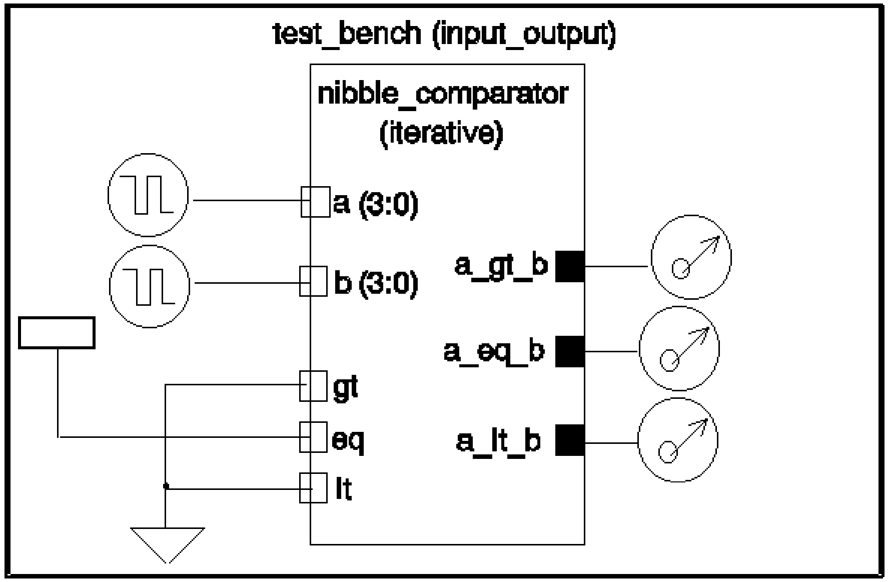
- Standard Package
  - Defines primitive types, subtypes, and functions.
  - e.g. Type Boolean IS (false, true);
  - e.g. Type Bit is (‘0’, ‘1’);
- TEXTIO Package
  - Defines types, procedures

Libraries

- Exiting libraries
  - STD Library
    - Contains the **STANDARD** and **TEXTIO** packages
    - Contains all the standard types & utilities
    - Visible to all designs
  - WORK library
    - Root library for the user
- IEEE library
  - Contains VHDL-related standards
  - Contains the std_logic_1164 (IEEE 1164.1) package
    - Defines a nine values logic system
    - De Facto Standard for all Synthesis Tools

Structural Test Bench

- A Testbench is an Entity without Ports that has a Structural Architecture
- The Testbench Architecture, in general, has 3 major components:
  - Instance of the Entity Under Test (EUT)
  - Test Pattern Generator (Generates Test Inputs for the Input Ports of the EUT)
  - Response Evaluator (Compares the EUT Output Signals to the Expected Correct Output)



VHDL Predefined Operators

- Logical Operators: NOT, AND, OR, NAND, NOR, XOR, XNOR
  - Operand Type: Bit, Boolean, Bit_vector
  - Result Type: Bit, Boolean, Bit_vector
- Relational Operators: =, /=, <, <=, >, >=
  - Operand Type: Any type
  - Result Type: Boolean
- Arithmetic Operators: +, -, *, /
  - Operand Type: Integer, Real
  - Result Type: Integer, Real
- Concatenation Operator: &
  - Operand Type: Arrays or elements of same type
  - Result Type: Arrays
- Shift Operators: SLL, SRL, SLA, SRA, ROL, ROR
  - Operand Type: Bit or Boolean vector
  - Result Type: same type

VHDL Reserved Words

abs	disconnect		label		package		
access	downto	library		Poll		units	
after	linkage				procedure		until
alias	else		loop		process	use	
all	elsif						variable
and	end		map		range		
architecture	entity		mod		record		wait
array	exit		nand		register	when	
assert	new		rem		while		
attribute file		next		report		with	
begin	for		nor		return		xor
block	function	not		select			
body	generate	null		severity			
buffer	generic		of		signal		
bus	guarded	on		subtype			
case	if		open		then		
component	in		or		to		
configuration	inout		others		transport		
constant is		out		type			

Variables & Signals

VARIABLES	SIGNALS
• Variables are only *Local* and May Only Appear within the Body of a Process or a SubProgram • Variable Declarations Are Not Allowed in Declarative Parts of Architecture Bodies or Blocks.	• Signals May be Local or Global. • Signals May not be Declared within Process or Subprogram Bodies. • All Port Declarations Are for Signals.
A Variable Has No HardWare Correspondence	A Signal Represents a Wire or a Group of Wires (BUS)
Variables Have No ***Time*** Dimension Associated With Them. (*Variable Assignment occurs instantaneously*)	Signals Have ***Time*** Dimension (*A Signal Assignment is Never Instantaneous* (Minimum Delay = δ Delay)
Variable Assignment Statement is always ***SEQUENTIAL***	Signal Assignment Statement is Mostly ***CONCURRENT*** (*Within Architectural* Body). It Can Be **SEQUENTIAL** (*Within Process Body*)
Variable Assignment Operator is :=	Signal Assignment Operator is <=

Variables Within Process Bodies are STATIC, i.e. a Variable Keeps its Value from One Process Call to Another. Variables Within Subprogram Bodies Are Dynamic, i.e. Variable Values are Not held from one Call to Another.

Signals vs Variables

- Signals
  - Signals follow the notion of 'event scheduling'
  - An event is characterized by a (time,value) pair
  - Signal assignment example:

    `X <= Xtmp;` means

    Schedule the assignment of the value of signal `Xtmp` to signal `X` at (Current time + delta)

    where delta: infinitesimal time unit used by simulator for processing the signals

Signals vs Variables

- Variables
  - Variables do not have notion of ‘events’
  - Variables can be defined and used only inside the `process` block and some other special blocks.
  - Variable declaration and assignment example:

```
process (…)
variable K : bit;
  begin
    …
   -- Assign the value of signa
immediately
    K := L;
   …
end process;
```

Variables can only be defined and used inside the `process` construct and can be defined only in this place

Predefined Numeric Data Types

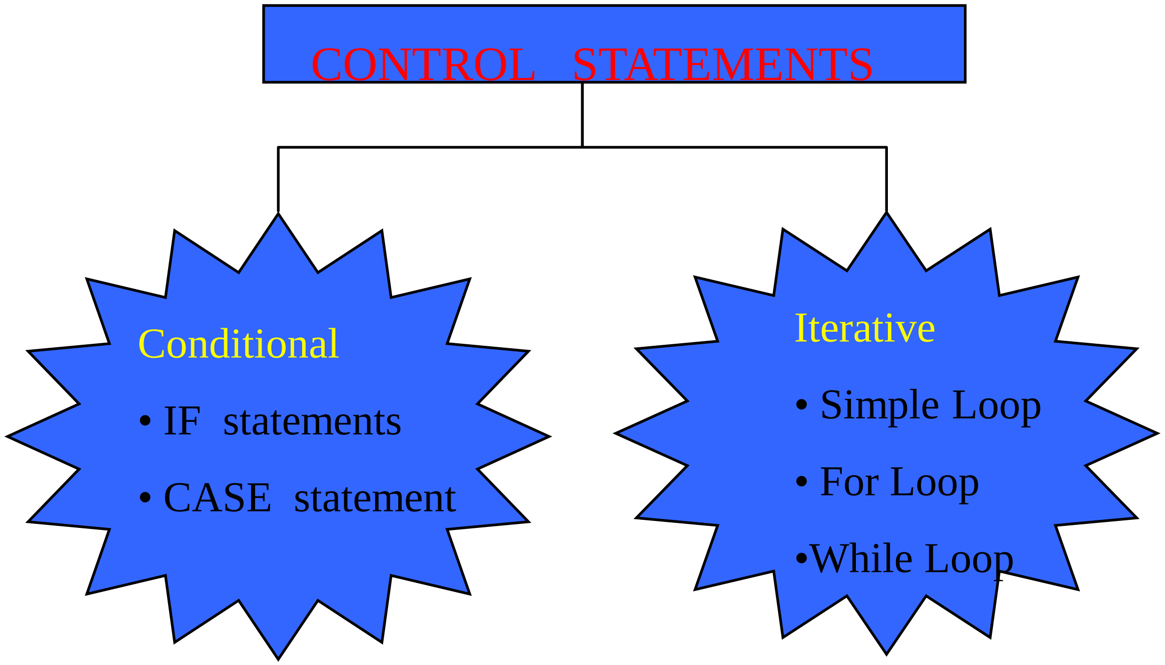
- INTEGER -- Range is Machine limited but At Least $-(2^{31} - 1)$ To $(2^{31} - 1)$
- POSITIVE -- INTEGERS > 0
- NATURAL -- INTEGERS $>= 0$
- REAL -- Range is Machine limited

Built-in Data types

- Scalar (single valued) signal types:
  - `bit`
  - `boolean`
  - `integer`
  - Examples:
    - `A: in bit;`
    - `G: out boolean;`
    - `K: out integer range -2**4 to 2**4-1;`
- Aggregate (collection) signal types:
  - `bit_vector`: array of bits representing binary numbers
  - `signed`: array of bits representing signed binary numbers
  - Examples:
    - `D: in bit_vector(0 to 7);`
    - `E: in bit_vector(7 downto 0);`
    - `M: in signed (4 downto 0);`
      `--signed 5 bit_vector binary number`

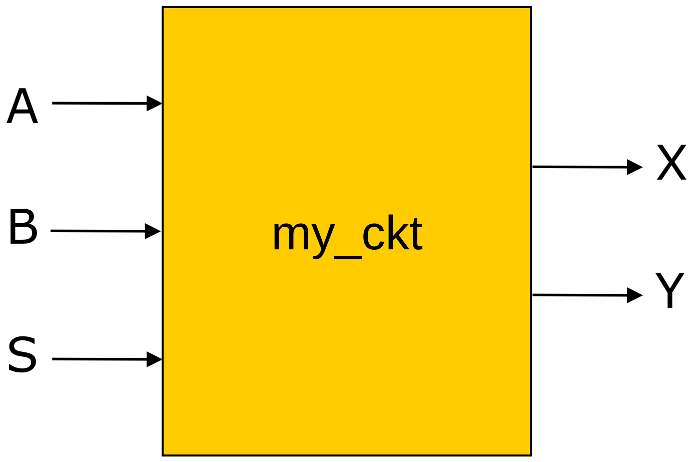
User-defined data type

- Construct datatypes arbitrarily or using built-in datatypes
- Examples:
  - `type temperature is (high, medium, low);`
  - `type byte is array(0 to 7) of bit;`


CONTROL STATEMENTS
Conditional
• IF statements
• CASE statement
Iterative
• Simple Loop
• For Loop
•While Loop

Functional specification

- Example:
  - Behavior for output X:
    - When S = 0
      X <= A
    - When S = 1
      X <= B
  - Behavior for output Y:
    - When X = 0 and S =0
      Y <= '1'
    - Else
      Y <= '0'



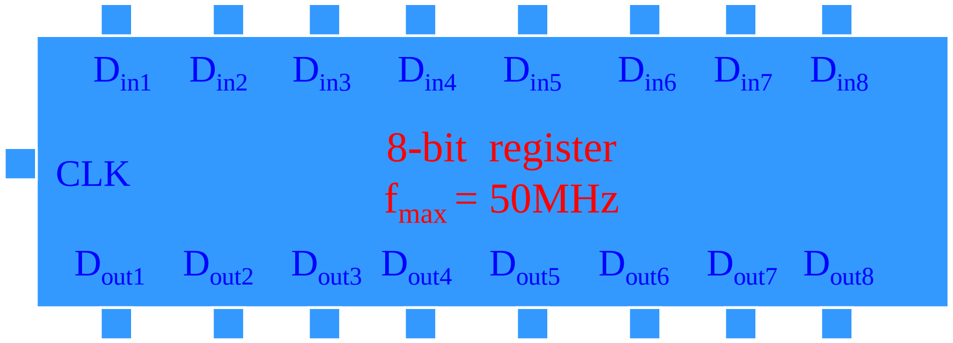
WHILE Loop

- Syntax:

```
[Loop_Label]: WHILE condition  LOOP
                                    statements;
                  End LOOP Loop_Label;
```

Design Entity

- In VHDL, the name of the system is the same as the name of its entity.
- Entity comprises two parts:
  - ***parameters*** of the system as seen from outside such as bus-width of a processor or max clock frequency
  - ***connections*** which are transferring information to and from the system (system's inputs and outputs)
- All parameters are declared as generics and are passed on to the body of the system
- Connections, which carry data to and from the system, are called *ports*. They form the second part of the entity.



D_{in1} D_{in2} D_{in3} D_{in4} D_{in5} D_{in6} D_{in7} D_{in8}
CLK
8-bit register
f_{max} = 50MHz
D_{out1} D_{out2} D_{out3} D_{out4} D_{out5} D_{out6} D_{out7} D_{out8}

entity Eight_bit_register is

parameters

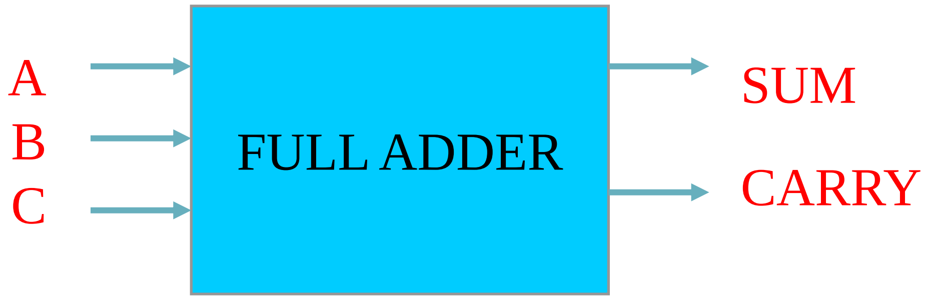
CLK one-bit input

connections

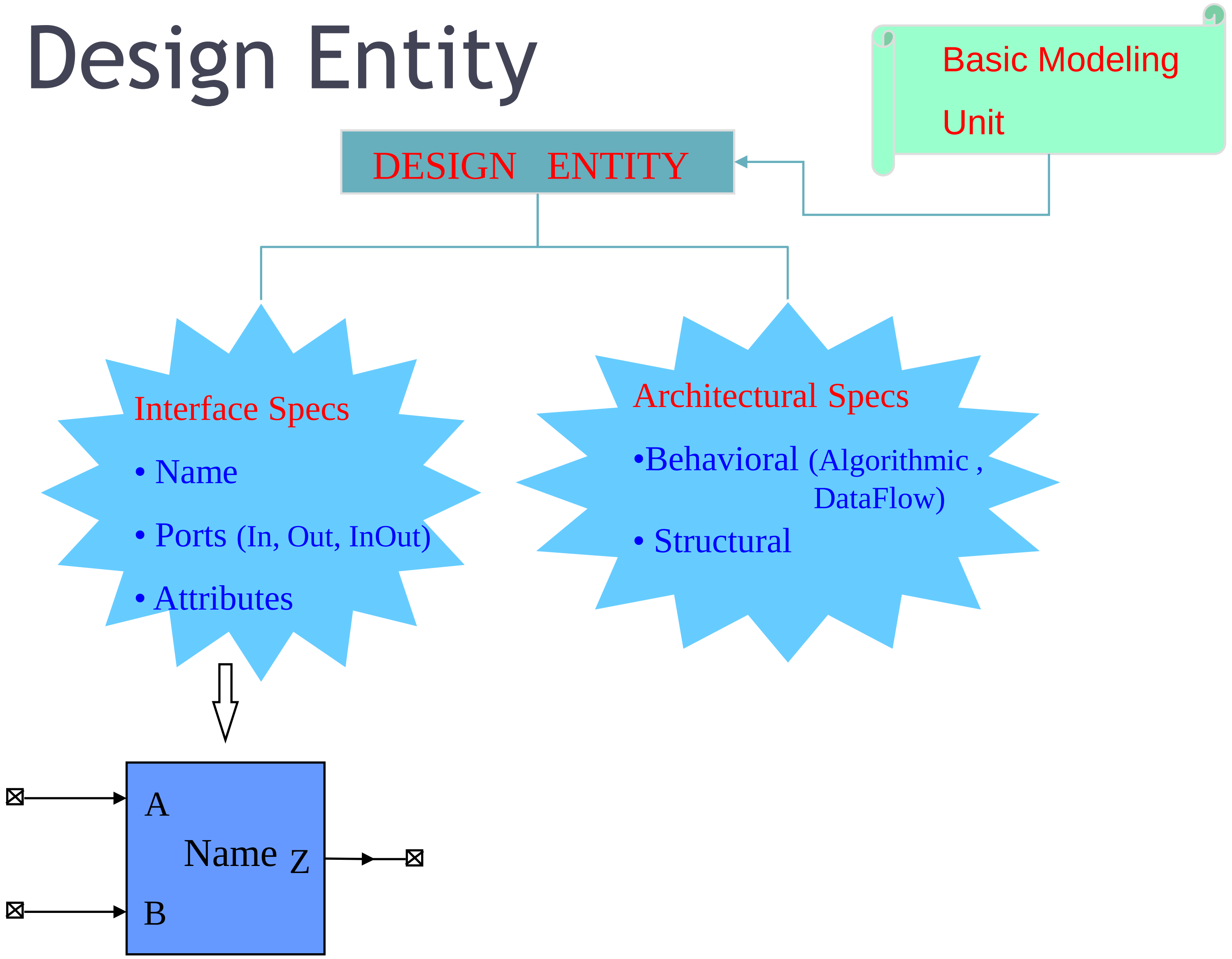
end [entity] [Eight_bit_register]

Entity Examples …

```
Entity FULLADDER is
 -- Interface description of FULLADDER
port (          A, B, C: in bit;
           SUM, CARRY: out bit);
end FULLADDER;
```



… Design Entity



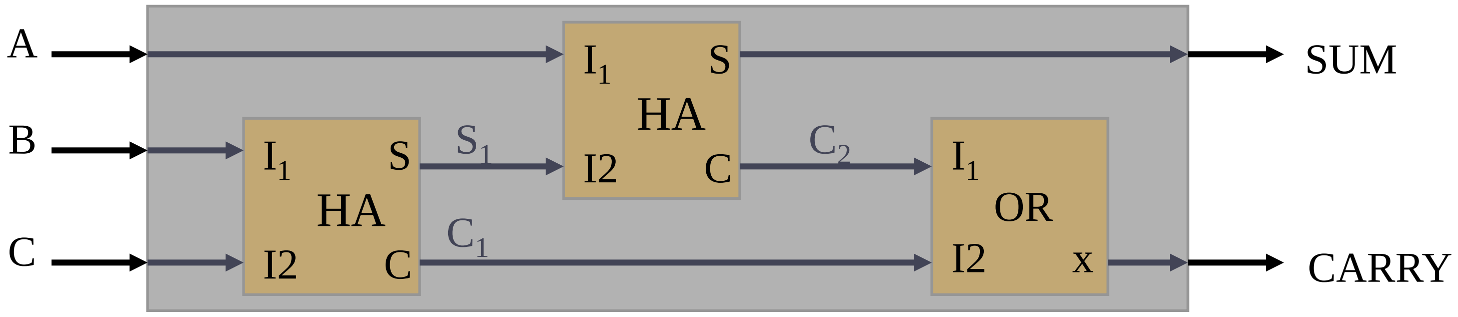
Architecture Examples: Behavioral Description

-
  ```
  Entity FULLADDER is
    port (            A, B, C: in bit;
                 SUM, CARRY: out bit);
  end FULLADDER;
  ```

-
  ```
  Architecture CONCURRENT of FULLADDER is
  begin
    SUM <= A xor B xor C after 5 ns;
    CARRY <= (A and B) or (B and C) or (A and C) after 3 ns;
  end CONCURRENT;
  ```

Architecture Examples: Structural Description …

```
architecture STRUCTURAL of FULLADDER is
  signal S1, C1, C2 : bit;
  component HA
    port (I1, I2 : in bit; S, C : out bit);
  end component;
  component OR
    port (I1, I2 : in bit; X : out bit);
  end component;
begin
  INST_HA1 : HA     port map (I1 => B, I2 => C, S => S1, C => C1);
  INST_HA2 : HA     port map (I1 => A, I2 => S1, S => SUM, C => C2);
  INST_OR : OR  port map (I1 => C2, I2 => C1, X => CARRY);
end STRUCTURAL;
```


A
B
C
SUM
CARRY
I_1
S
HA
I2
C
S_1
C_1
C_2
OR
x