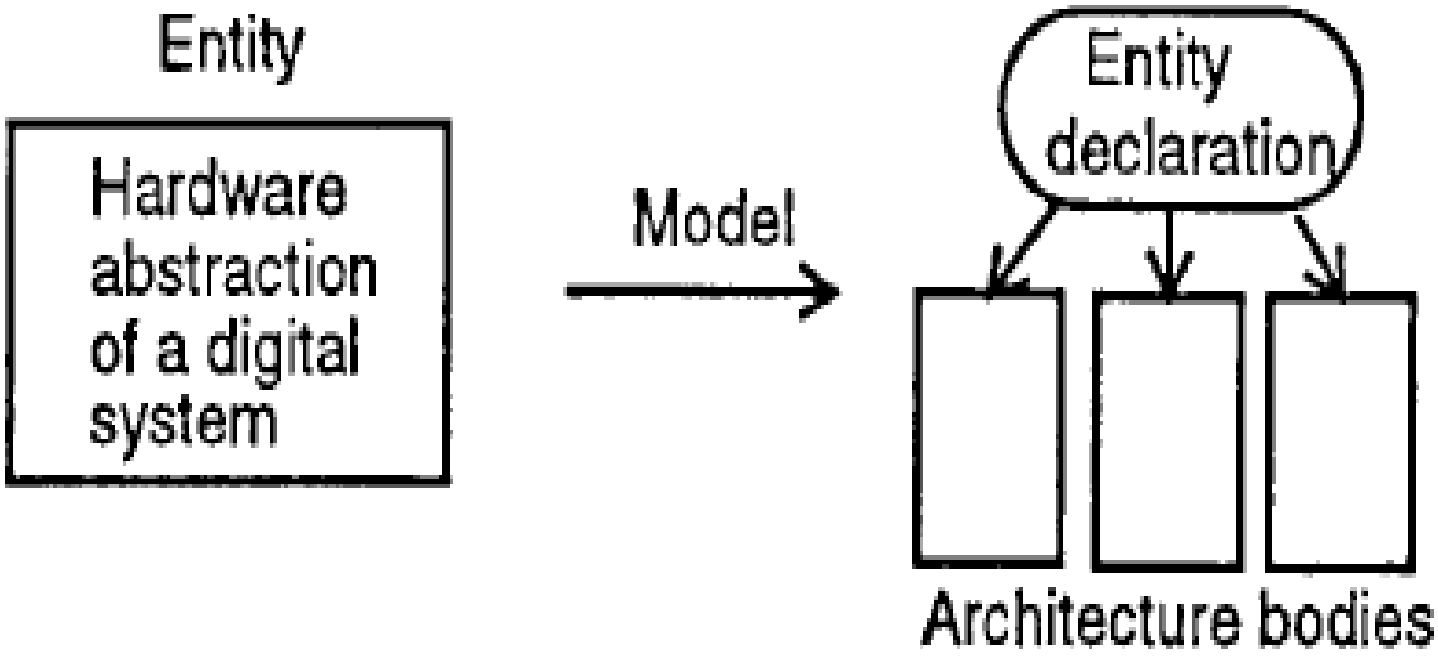
Basic Terminology

- To describe an entity, VHDL provides five different types of primary constructs, called" design *units.*

1. Entity declaration
2. Architecture body
3. Configuration declaration
4. Package declaration
5. Package body

Entity & its model



1. Entity

- An entity is modeled using an entity declaration and at least one architecture body. The entity declaration describes the external view of the entity, for example, the input and output signal names.

2. Architecture body

- The architecture body contains the internal description of the entity, for example, as a set of interconnected components that represents the structure of the entity, or as a set of concurrent or sequential statements that represents the behaviour of the entity.

3. Configuration Declaration

- A configuration declaration is used to create a configuration for an entity. It specifies the binding of one architecture body from the many architecture bodies that may be associated with the entity. It may also specify the bindings of components used in the selected architecture body to other entities. An entity may have any number of different configurations.

Configuration Declaration cont..

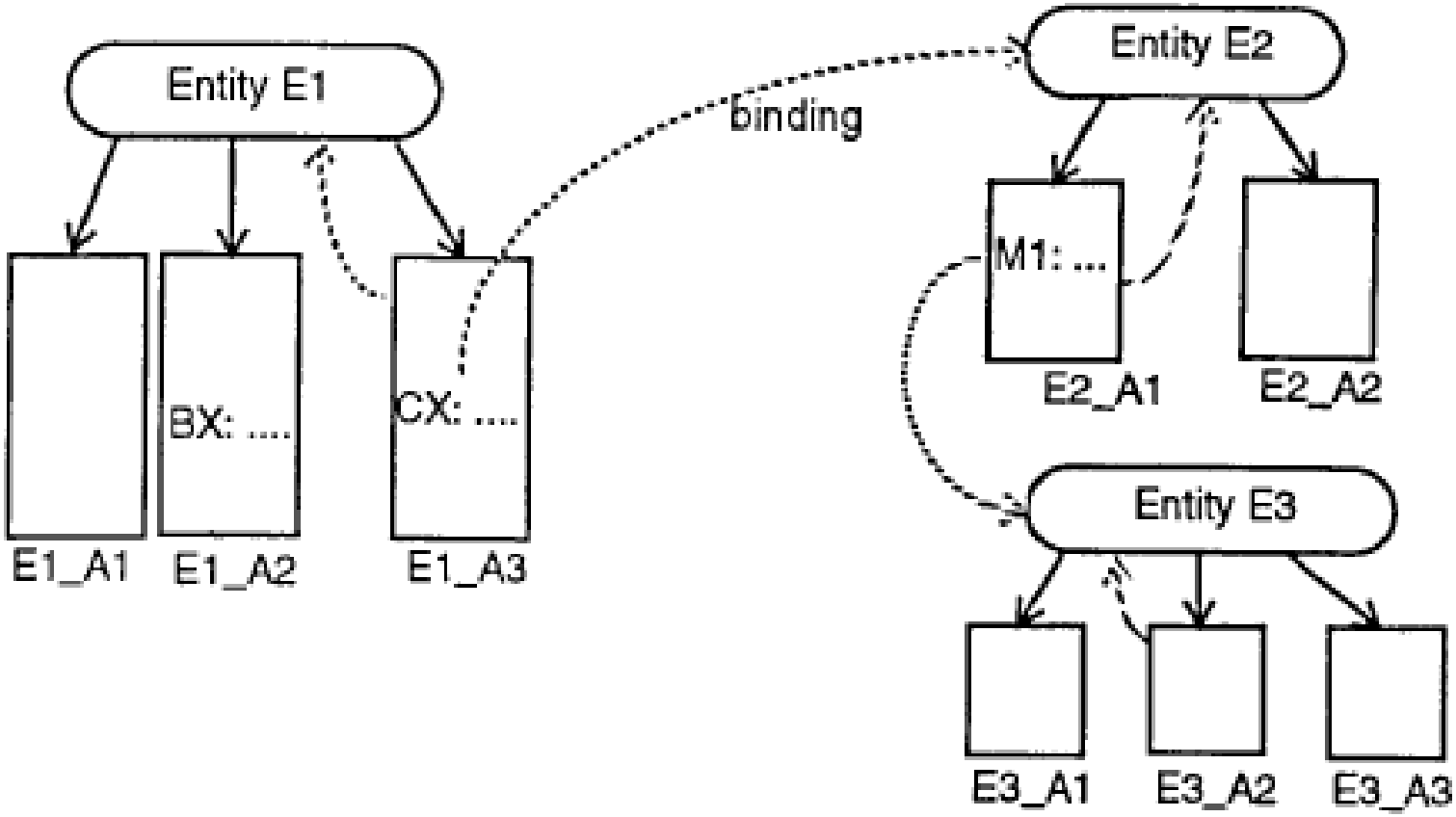


Figure 2.2 A configuration for entity El.

4. Package Declaration

- A package declaration encapsulates a set of related declarations such as type declarations, subtype declarations, and subprogram declarations that can be shared across two or more design units.

5. Package Body

- A package body contains the definitions of subprograms declared in a package declaration.

Entity Declaration

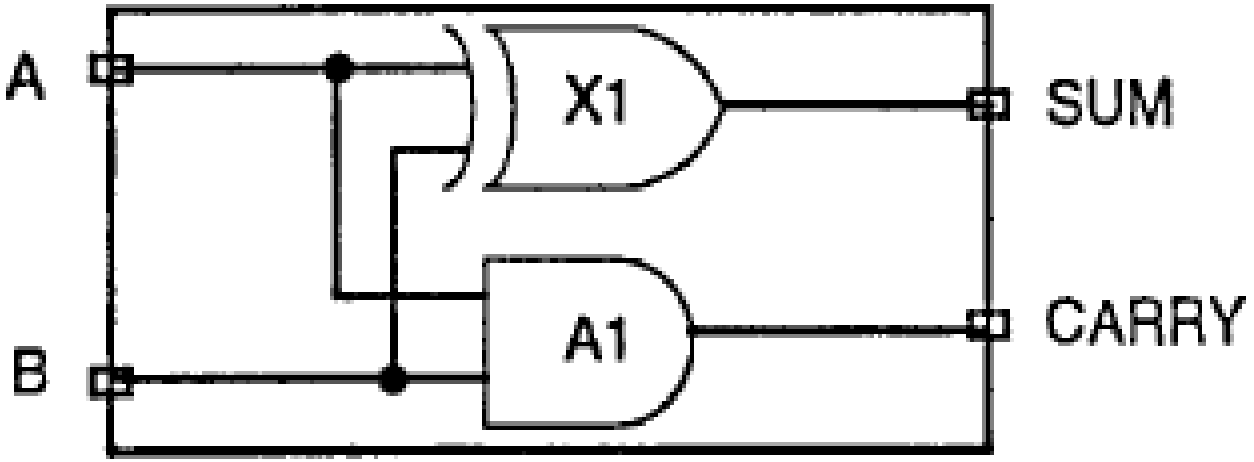


Figure 2.3 A half-adder circuit.

```
entity HALF_ADDER is
port (A, B: in BIT; SUM, CARRY: out BIT);
end HALF_ADDER;
```

Architecture Body

- The internal details of an entity are specified by an architecture body using any of the following modeling styles:

1. As a set of interconnected components (to represent structure),

2. As a set of concurrent assignment statements (to represent dataflow),

3. As a set of sequential assignment statements (to represent behavior),

4. Any combination of the above three.

Structural Style of Modeling

- **Example: Half Adder**

```
architecture HA_STRUCTURE of HALF_ADDER is
component XOR2
    port (X, Y: in BIT; Z: out BIT);
end component;
component AND2
    port (L, M: in BIT; N: out BIT);
end component;

begin
    X1: XOR2 port map (A, B, SUM);
    A1: AND2 port map (A, B, CARRY);
end HA_STRUCTURE;
```

Dataflow Style of Modeling

- **Example: Half Adder**

```
architecture HA_CONCURRENT of
  HALF_ADDER is
begin
SUM <= A xor B after 8 ns;
CARRY <= A and B after 4 ns;
end HA_CONCURRENT;
```

Behavioral Style of Modeling

- **architecture HA_beh of HALF_ADDER is**

begin

process (A, B)

begin

SUM <= A **xor B ;**

CARRY <= A **and B ;**

end process;

End HA_beh;

Mixed Style of Modeling

- It is possible to mix the three modeling styles that we have seen so far in a single architecture body.

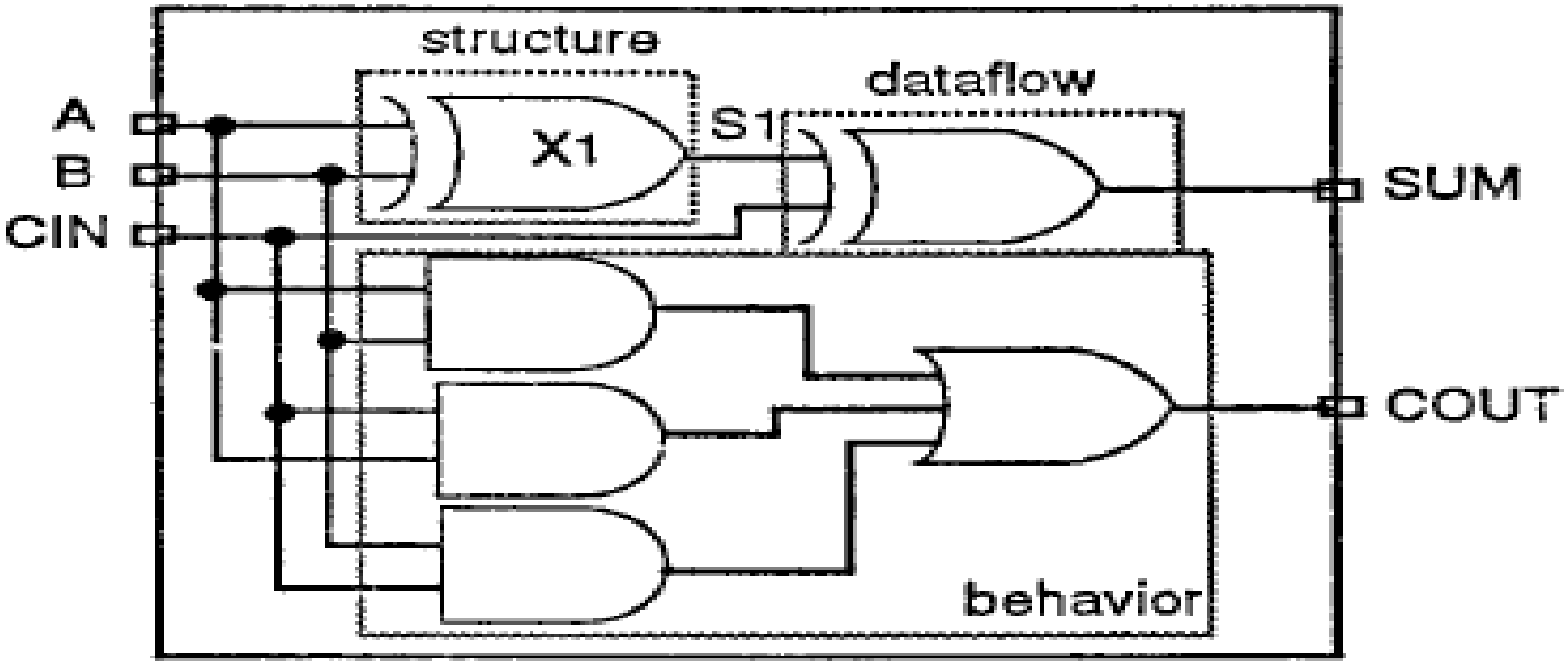


Figure 2.7 A 1-bit full-adder.

Mixed Style of Modeling cont..

```
entity FULL_ADDER is
port (A, B, CIN: in BIT; SUM, COUT: out
   BIT);
end FULL_ADDER;

architecture FA_MIXED of
   FULL_ADDER is
component XOR2
   port (X,Y: in BIT; Z: out BIT);
end component;
signal S1: BIT;
```

Mixed Style of Modeling cont..

```
begin

X1: XOR2 port map (A, B, S1 );                    - structure.
process (A, B, CIN)                               - behavior.
    variable T1, T2, T3: BIT;

begin
    T1 :=A and B;
    T2 := B and CIN;
    T3:=A and CIN;
    COUT <= T1 or T2 or T3;
end process;
SUM <= S1 xor CIN;                                - dataflow.
end FA_M!XED;
```