VHDL Quick Start

Peter J. Ashenden

The University of Adelaide

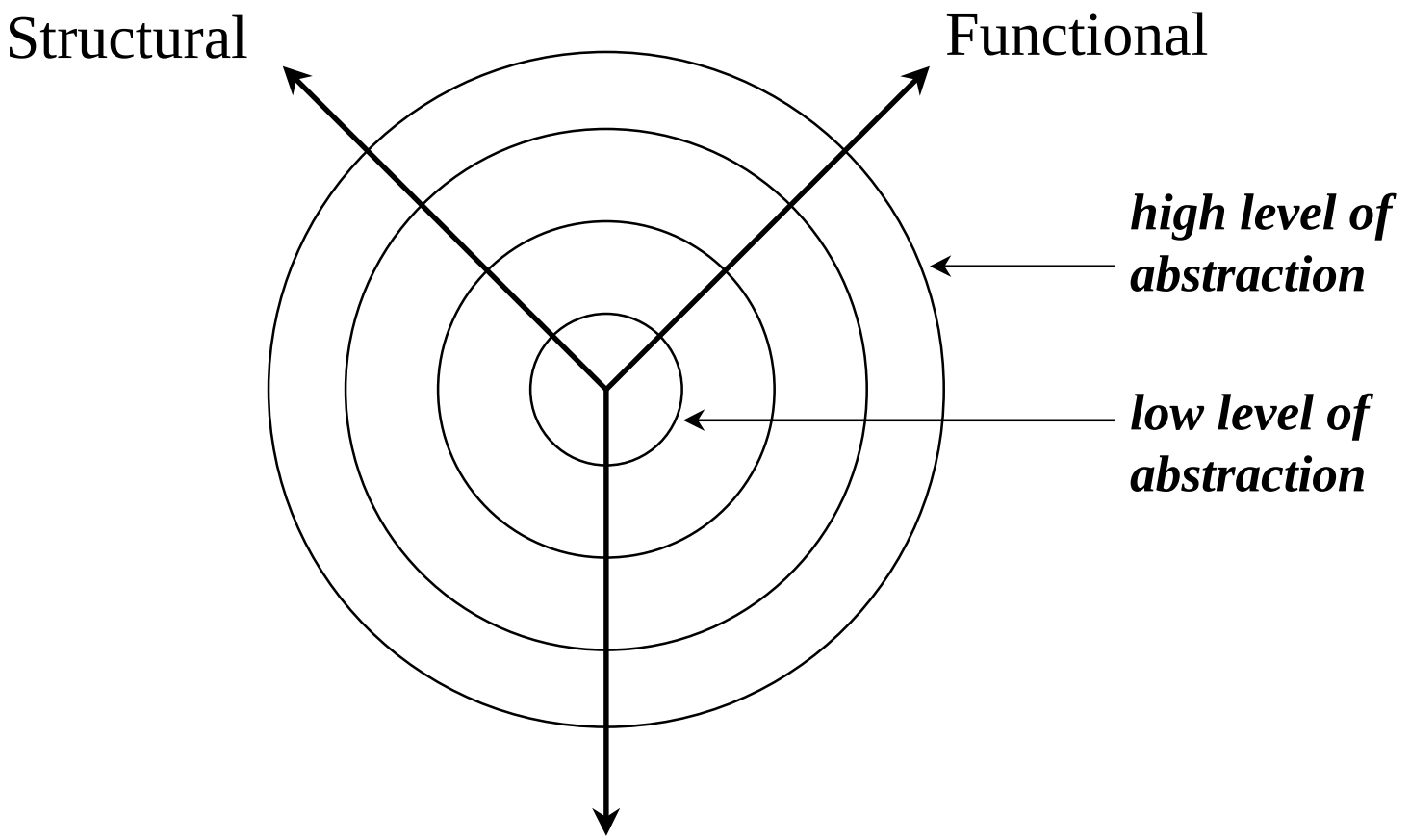
Objective

- Quick introduction to VHDL
  - basic language concepts
  - basic design methodology
- Use *The Student's Guide to VHDL* or *The Designer's Guide to VHDL*
  - self-learning for more depth
  - reference for project work

Modeling Digital Systems

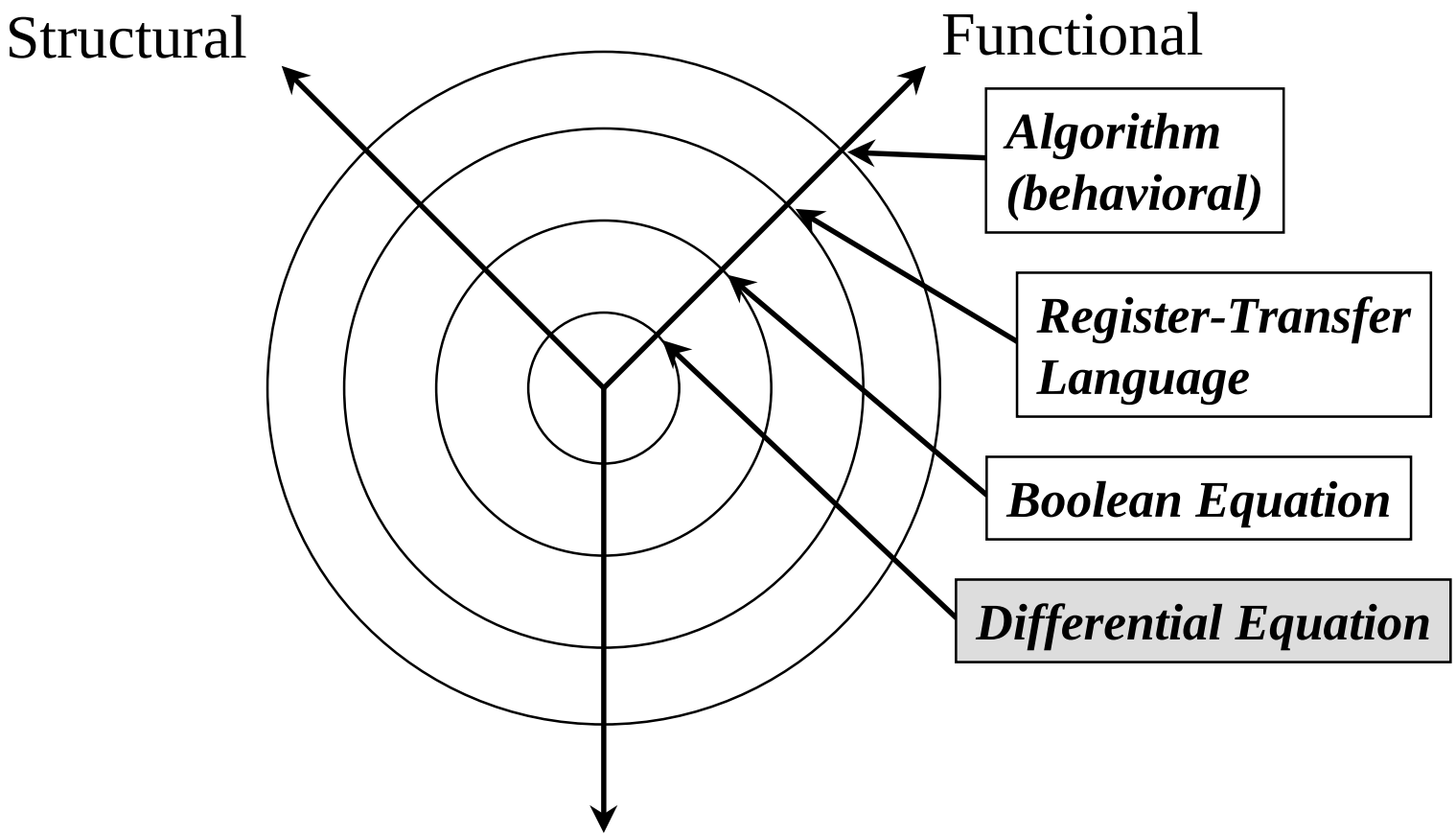
- VHDL is for writing models of a system
- Reasons for modeling
  - requirements specification
  - documentation
  - testing using simulation
  - formal verification
  - synthesis
- Goal
  - most reliable design process, with minimum cost and time
  - avoid design errors!

Domains and Levels of Modeling



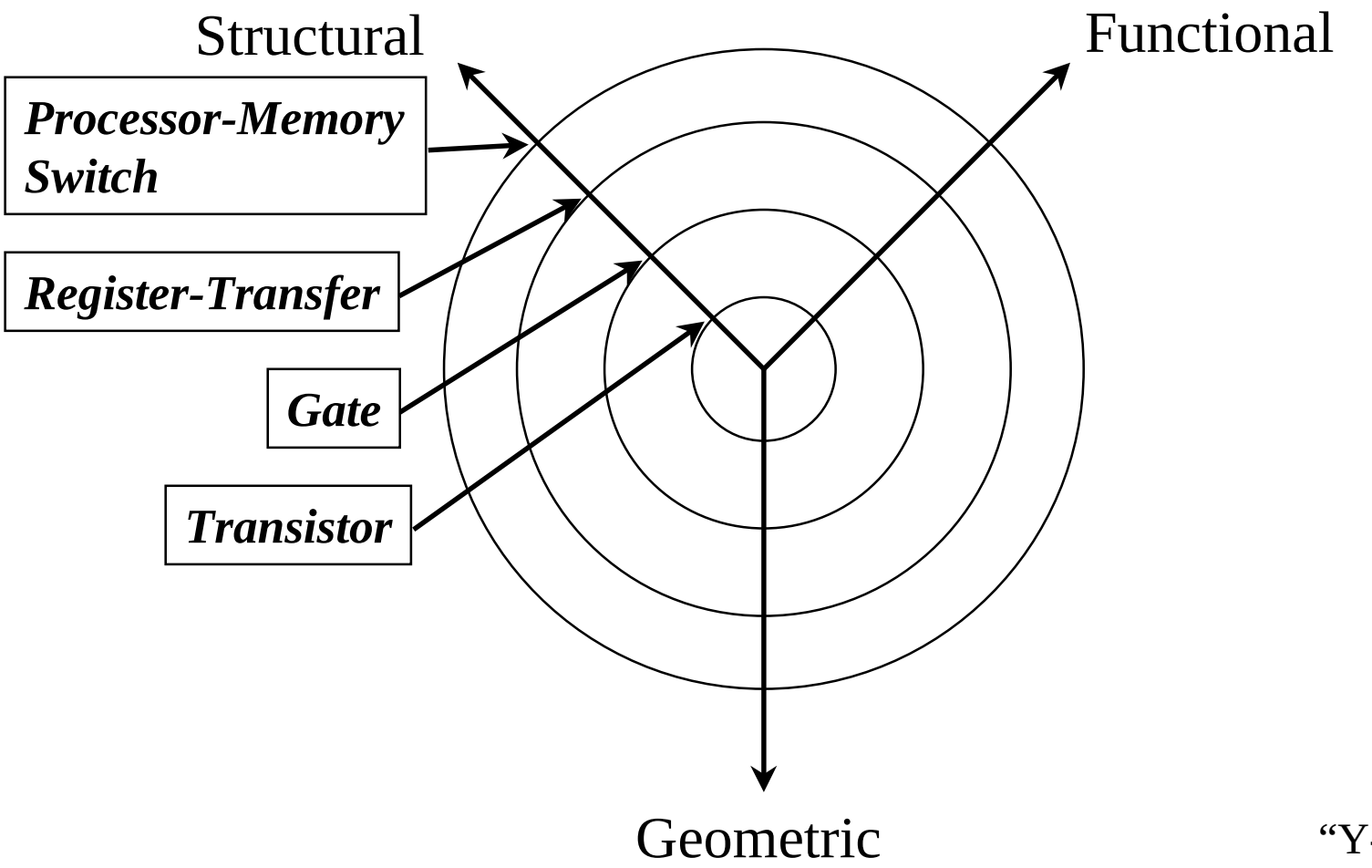
"Y-chart" due to Gajski & Kahn

Domains and Levels of Modeling



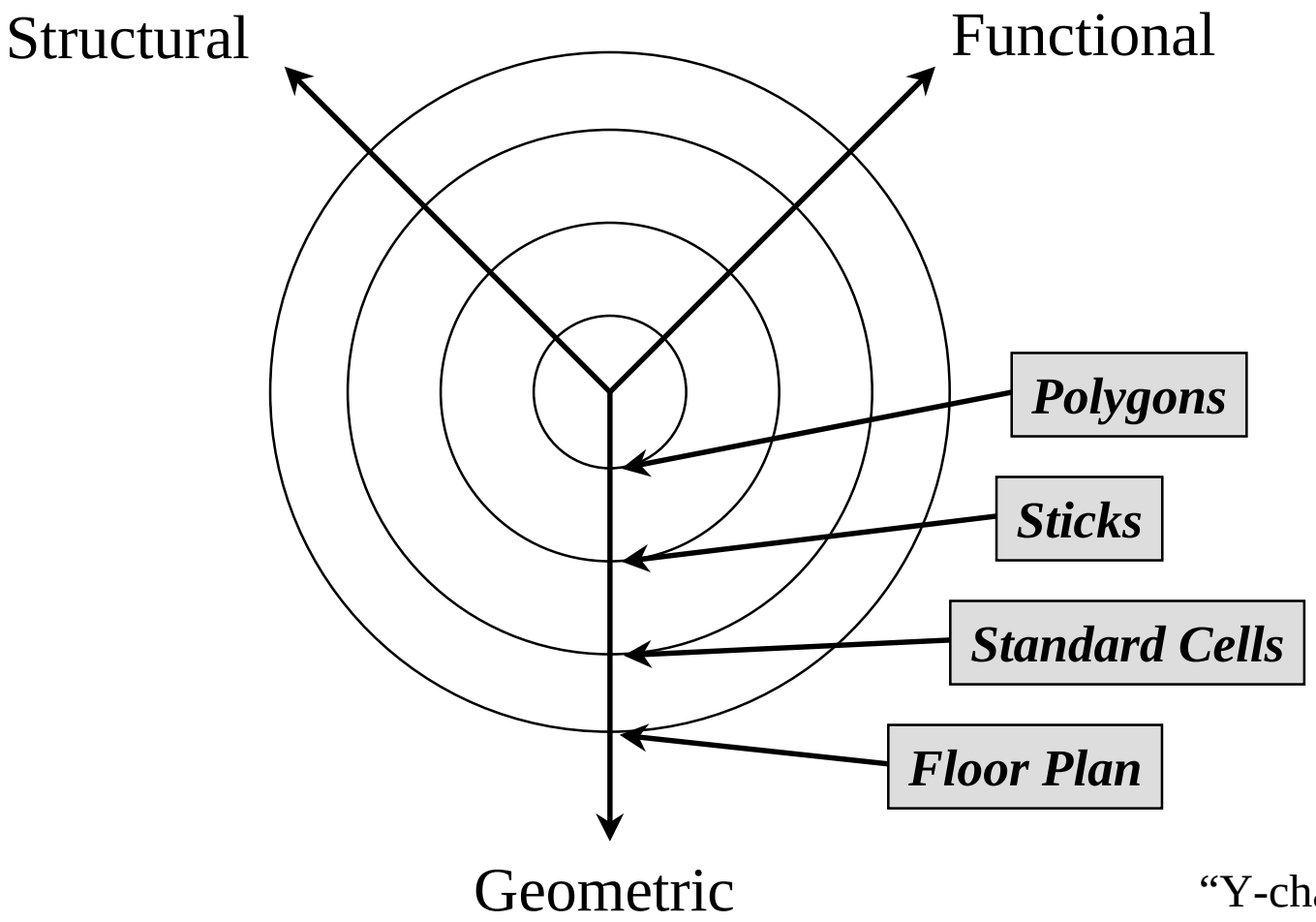
"Y-chart" due to Gajski & Kahn

Domains and Levels of Modeling



"Y-chart" due to Gajski & Kahn

Domains and Levels of Modeling



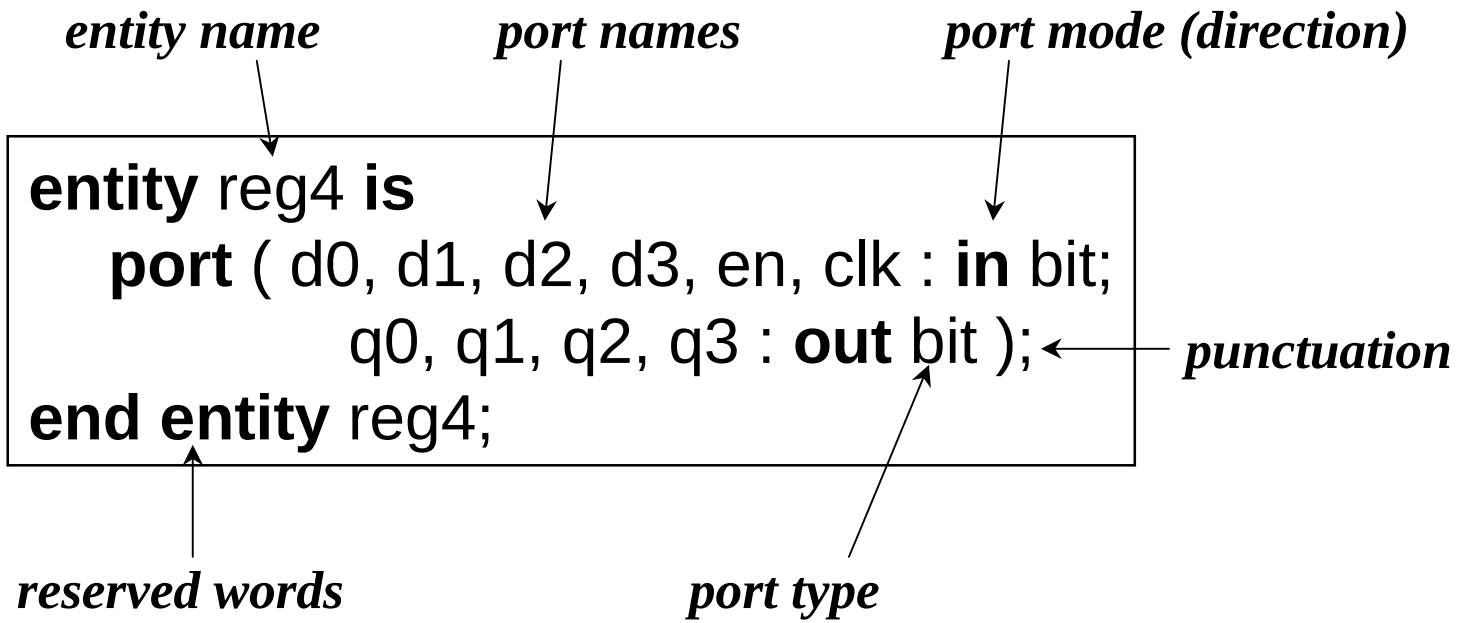
"Y-chart" due to Gajski & Kahn

Basic VHDL Concepts

- Interfaces
- Behavior
- Structure
- Test Benches
- Analysis, elaboration, simulation
- Synthesis

Modeling Interfaces

- *Entity* declaration
  - describes the input/output *ports* of a module



VHDL-87

- Omit **entity** at end of entity declaration

```
entity reg4 is
    port ( d0, d1, d2, d3, en, clk : in bit;
                q0, q1, q2, q3 : out bit );
end reg4;
```

Modeling Behavior

- *Architecture body*
  - describes an implementation of an entity
  - may be several per entity
- *Behavioral* architecture
  - describes the algorithm performed by the module
  - contains
    - *process statements*, each containing
    - *sequential statements*, including
    - *signal assignment statements* and
    - *wait statements*

Behavior Example

```
architecture behav of reg4 is
begin
    storage : process is
        variable stored_d0, stored_d1, stored_d2, stored_d3 : bit;
    begin
        if en = '1' and clk = '1' then
            stored_d0 := d0;
            stored_d1 := d1;
            stored_d2 := d2;
            stored_d3 := d3;
        end if;
        q0 <= stored_d0 after 5 ns;
        q1 <= stored_d1 after 5 ns;
        q2 <= stored_d2 after 5 ns;
        q3 <= stored_d3 after 5 ns;
        wait on d0, d1, d2, d3, en, clk;
    end process storage;
end architecture behav;
```

VHDL-87

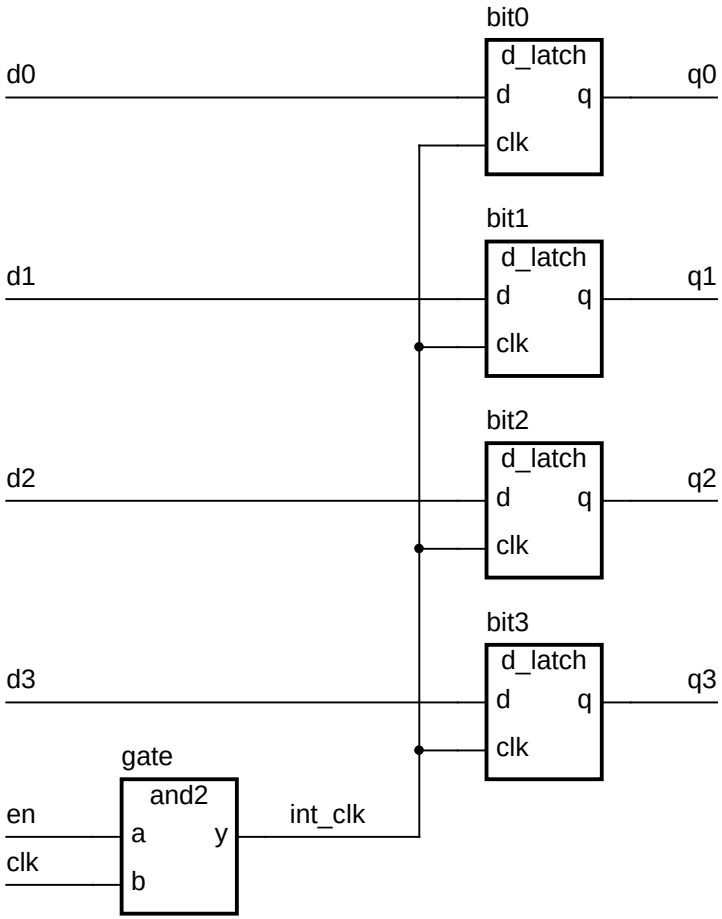
- Omit **architecture** at end of architecture body
- Omit **is** in process statement header

```
architecture behav of reg4 is
begin
    storage : process
        ...
    begin
        ...
    end process storage;
end behav;
```

Modeling Structure

- *Structural* architecture
  - implements the module as a composition of subsystems
  - contains
    - *signal declarations*, for internal interconnections
      - the entity ports are also treated as signals
    - *component instances*
      - instances of previously declared entity/architecture pairs
    - *port maps* in component instances
      - connect signals to component ports
    - *wait statements*

Structure Example



Structure Example

- First declare D-latch and and-gate entities and architectures

```
entity d_latch is
    port ( d, clk : in bit;  q : out bit );
end entity d_latch;

architecture basic of d_latch is
begin
    latch_behavior : process is
    begin
        if clk = '1' then
            q <= d after 2 ns;
        end if;
        wait on clk, d;
    end process latch_behavior;
end architecture basic;
```

```
entity and2 is
    port ( a, b : in bit;  y : out bit );
end entity and2;

architecture basic of and2 is
begin
    and2_behavior : process is
    begin
        y <= a and b after 2 ns;
        wait on a, b;
    end process and2_behavior;
end architecture basic;
```

Structure Example

- Now use them to implement a register

```
architecture struct of reg4 is
    signal int_clk : bit;
begin
    bit0 : entity work.d_latch(basic)
        port map ( d0, int_clk, q0 );
    bit1 : entity work.d_latch(basic)
        port map ( d1, int_clk, q1 );
    bit2 : entity work.d_latch(basic)
        port map ( d2, int_clk, q2 );
    bit3 : entity work.d_latch(basic)
        port map ( d3, int_clk, q3 );
    gate : entity work.and2(basic)
        port map ( en, clk, int_clk );
end architecture struct;
```

VHDL-87

- Can’t directly instantiate entity/architecture pair
- Instead
  - include *component declarations* in structural architecture body
    - templates for entity declarations
  - instantiate components
  - write a *configuration declaration*
    - binds entity/architecture pair to each instantiated component

Structure Example in VHDL-87

- First declare D-latch and and-gate entities and architectures

```
entity d_latch is
    port ( d, clk : in bit;  q : out bit );
end d_latch;

architecture basic of d_latch is
begin
    latch_behavior : process
    begin
        if clk = '1' then
            q <= d after 2 ns;
        end if;
        wait on clk, d;
    end process latch_behavior;
end basic;
```

```
entity and2 is
    port ( a, b : in bit;  y : out bit );
end and2;

architecture basic of and2 is
begin
    and2_behavior : process
    begin
        y <= a and b after 2 ns;
        wait on a, b;
    end process and2_behavior;
end basic;
```

Structure Example in VHDL-87

- Declare corresponding components in register architecture body

```
architecture struct of reg4 is
    component d_latch
        port ( d, clk : in bit;  q : out bit );
    end component;
    component and2
        port ( a, b : in bit;  y : out bit );
    end component;
    signal int_clk : bit;
...
```

Structure Example in VHDL-87

- Now use them to implement the register

```
...
begin
    bit0 : d_latch
        port map ( d0, int_clk, q0 );
    bit1 : d_latch
        port map ( d1, int_clk, q1 );
    bit2 : d_latch
        port map ( d2, int_clk, q2 );
    bit3 : d_latch
        port map ( d3, int_clk, q3 );
    gate : and2
        port map ( en, clk, int_clk );
end struct;
```

Structure Example in VHDL-87

- Configure the register model

```
configuration basic_level of
```