زبان توصیف سخت‌افزار

Hardware Description Language

زبان‌ها

- VHDL
  - یادگیری حداقل یک زبان به طور کامل
    - یادگیری ناقص و سعی و خطا ← سخت شدن توصیف و اشکال‌زدایی
  - اصول یکسان – ساختارهای نحوی متفاوت
  - VHSIC HDL (اوایل دهۀ 1980)
  - مبتنی بر زبان ADA

VHDL

- **کاربردها:**
  - مستندسازی:
    - به جای توصیف زبان طبیعی:
      - نادقیق (برداشت‌های مختلف)
      - غیر قابل پردازش

VHDL

- **کاربردها:**
  - مدل‌سازی:
    - مراحل اولیهٔ طراحی:
      - عملکرد و رفتار کلی مشخص است
      - توصیف الگوریتم با دستورهای رفتاری
      - کمک به فهم صورت مسأله
      - جلوگیری از سرایت اشکالات به سطوح پایین طراحی

VHDL

- **کاربردها:**
  - مدل‌سازی:
    - مثال: کنترل‌کنندهٔ آسانسور

```
...

while (true)
   begin

      ...

      if (REQUEST = true)then
         if (REQUEST_FLOOR > CURRENT_FLOOR) then
            MOVE_UP (CURRENT_FLOOR, ...);
         else
            MOVE_DOWN (CURRENT_FLOOR, ...);
         end if;
      end if;

      ...

   end;

...
```

VHDL

- **کاربردها:**
  - سنتز
    - تولید خودکار مدار از توصیف
    - زیرمجموعهٔ قابل سنتز
      - لزوم آشنایی با این زیرمجموعه
      - تفاوت ابزارها ← portable code
        - کد غیر قابل سنتز برای دیگری
        - مدار متفاوت
      - کلیات یکسان
    - این درس:
      - موارد کلی مشترک
      - تحوهٔ کدنویسی مناسب

VHDL

- **کاربردها:**
  - درستی‌سنجی
    - محیط درستی‌سنجی: testbench
    - عناصر اصلی:
      - طرح مورد آزمون
      - تولید و اِعمال بردارهای ورودی
      - مشاهده و تحلیل خروجی‌ها

VHDL

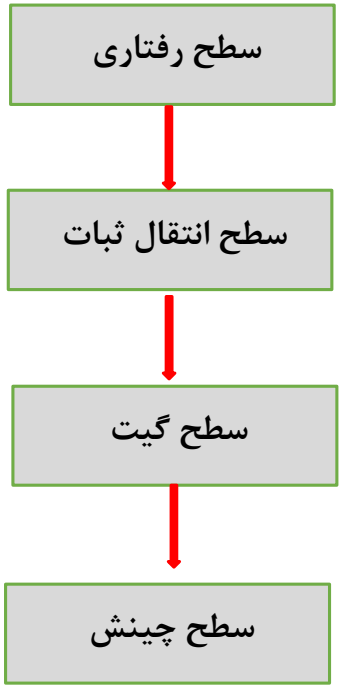
- **کاربردها:**
  - درستی‌سنجی
    - تولید بردارهای ورودی:
      1. تصادفی
      2. انتخاب هوشمندانه
      3. الگوریتم‌های تولید خودکار بردارهای آزمون (ATPG)
    - نیاز به توصیف با کد VHDL
      - برنامه‌نویسی روالی
      - خواندن از فایل

VHDL

- مثال:
  - سخت‌افزار محاسبهٔ $(x^2 + y^2)^{1/2}$
  - مقایسه با نتیجهٔ رفتاری

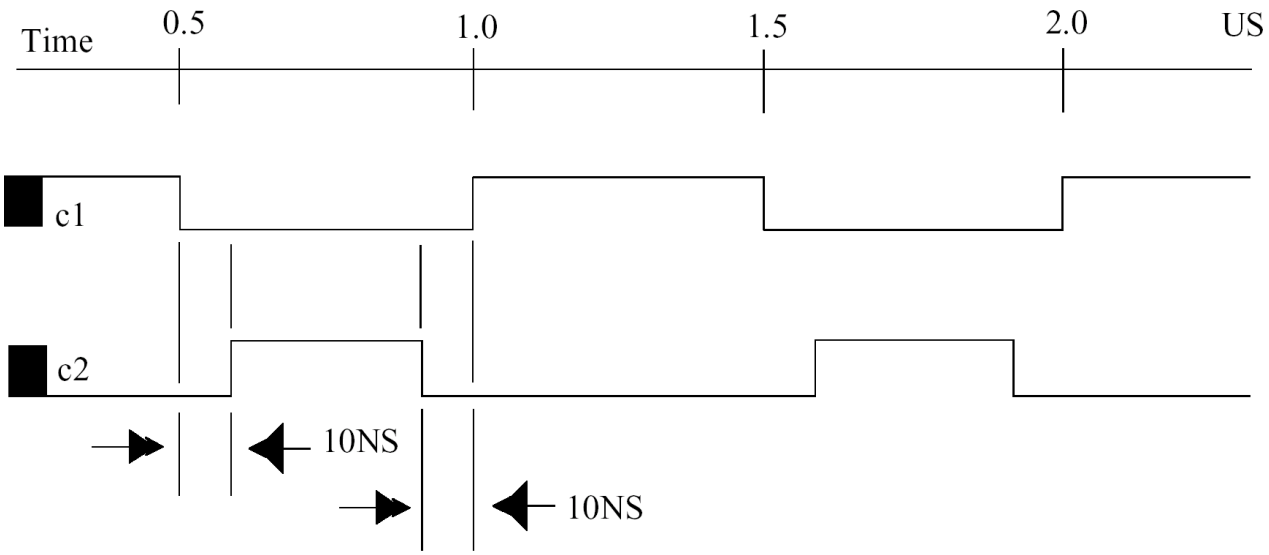
```
...

1          L1 : L_MODULE(X,Y,L);
           ⋮
11         LL : process (…)
12         begin
13         for X in 0 to 63 loop
14                   for Y in 0 to 63 loop
15                   begin
16                            L_BEHAVE := (X**2 + Y**2)**0.5;
17                            if(L != L_BEHAVE)
18                                     ERROR_SIG <= '1';
19                            end if;
20                   end for;
21         end for;
22         end process LL;

...
```

سطوح تجرید (Levels of Abstraction)



- **رفتاری:**
  - توصیف عملکرد مطلوب (الگوریتمی)
  - بدون پرداختن به مدار
  - همهٔ امکانات VHDL
  - قابل نمایش با روندنما (flow chart)

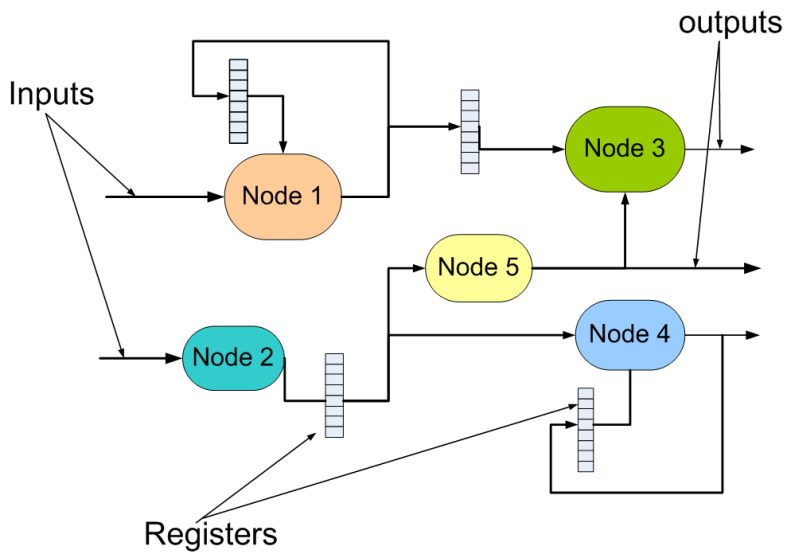
سطوح تجرید




```
...
phase2: PROCESS
BEGIN
  WAIT UNTIL c1 = '0';
  WAIT FOR 10 NS;
  c2 <= '1';
  WAIT FOR 480 NS;
  c2 <= '0';
END PROCESS phase2;
...
```

سطوح تجرید

- انتقال ثبات (Register-Transfer Level) RTL:



- مسیر داده
  - ثبات‌ها
  - مدارهای ترکیبی
  - منابع ارتباط دهنده بین آنها
    - گذرگاه
    - مالتی‌پلکسر
    - سیم
    - ...
- کنترل‌کننده (مسیر کنترل)
  - ماشین حالت متناهی FSM

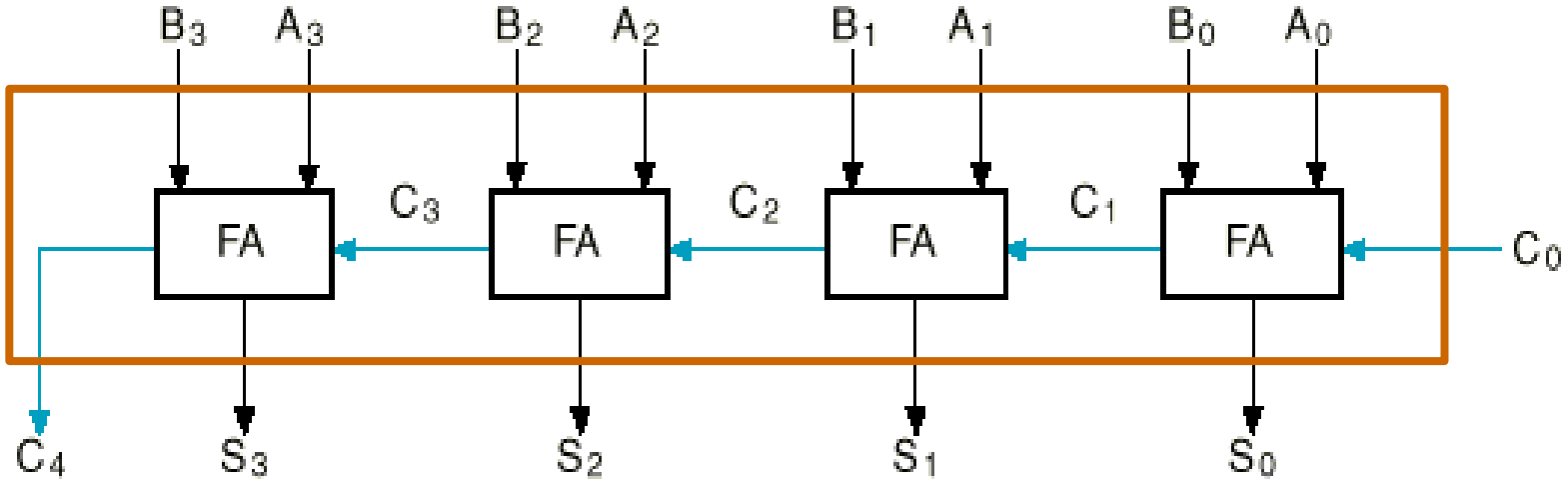
سطوح تجرید

- انتقال ثبات:

```
:
process(CLK)
begin
if (rising_edge(CLK)) then
        DOUT <= not(DOUT);
end if;
if (falling_edge(CLK)) then
        if (DIN == '0') then
                DOUT <= not(DOUT);
        end if;
end if;
end process;
:
```

سطوح تجرید

- ساختاری (structural):




```
begin
    FA1:FULL_ADDER port map (C0, A0, B0, S0, C1);
    FA2:FULL_ADDER port map (C1, A1, B1, S1, C2);
    FA3:FULL_ADDER port map (C2, A2, B2, S2, C3);
    FA4:FULL_ADDER port map (C3, A3, B3, S3, C4);
end;
```

- FULL_ADDER خود ممکن است از گیت‌ها تشکیل شده باشد.

توصیف سخت‌افزار با VHDL

- **جزئیات:**
  - Language Reference Manual
    - به عنوان مرجع اصلی
    - 1076 LRM Ieee Standard Vhdl Language Reference Manual.pdf
  - زیرمجموعهٔ قابل سنتز: استاندارد 1076.6
  - VHDL-AMS 1076.1
  - فقط شبیه‌سازی
  - نحوهٔ شبیه‌سازی: معادلات دیفرانسیل

VHDL Structural Elements

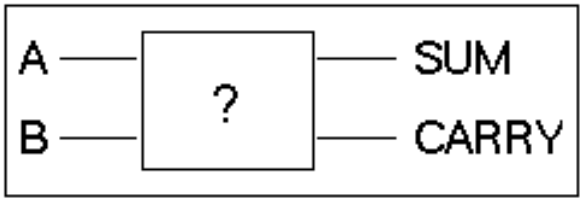
- **Entity:** **Interface**
- **Architecture:** **Implementation, behavior**
- **Configuration:** **Structure, hierarchy**
- **Process:** **Sequential Execution**
- **Package:** **Components (Modular design), Utilities (data types, constants, subprograms)**
- **Library:** **Group of compiled units, object code**

Entity

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

```
entity ADDER is          in   bit_vector (3 downto 0);
   port(                 out  bit_vector (3 downto 0);
    A, B:          in    integer range 15 to 0;
    SUM:           out   integer range 15 to 0;
    CARRY:         out   bit );
end ADDER;
```

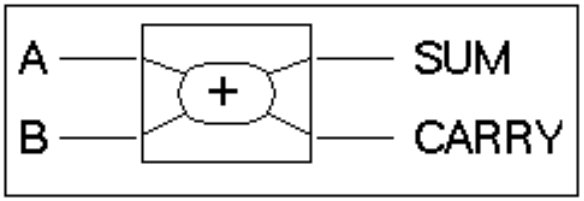
- **Interface description**
- **No behavior/implementation definition**



- **Linking via port signals**
  - data types
  - signal width
  - signal direction

Architecture

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

architecture RTL of HALFADDER is
begin
  SUM     <= A xor B;
  CARRY <= A and B;
end architecture RTL;
```



- **Implementation of the design**
- **Always connected with a specific entity**
  - one entity can have several architectures
  - entity ports are available as signals within the architecture
- **Contains concurrent statements**

Architecture Structure

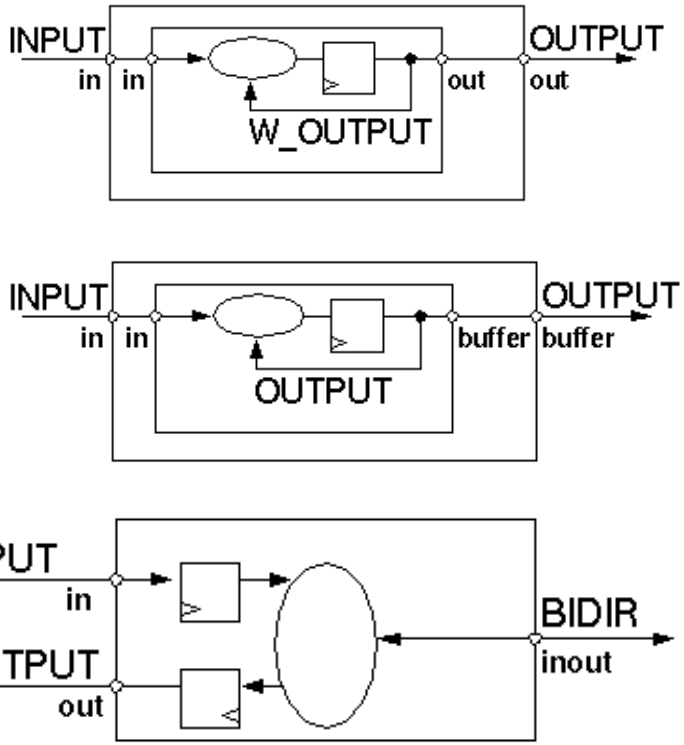
```
architecture RTL  of  STRUCTURE  is
  subtype DIGIT is integer range 0 to 9;
  constant BASE: integer := 10;
  signal DIGIT_A, DIGIT_B: DIGIT;
  signal CARRY:            DIGIT;
begin
  DIGIT_A <= 3;
  SUM <= DIGIT_A + DIGIT_B;
  DIGIT_B <= 7;
  CARRY <= 0 when SUM < BASE else
           1;
end EXAMPLE ;
```

- **Declarative part:**
  - data types
  - constants
  - intermediate signals
  - component declarations
  - ...
- **Statement part (after 'begin'):**
  - signal assignments
  - processes
  - component instantiations
  - concurrent statements: order not important

    → Simulator recalculates **sum** any time an operand changes

- **Port is a kind of signal.**

Entity Port Modes



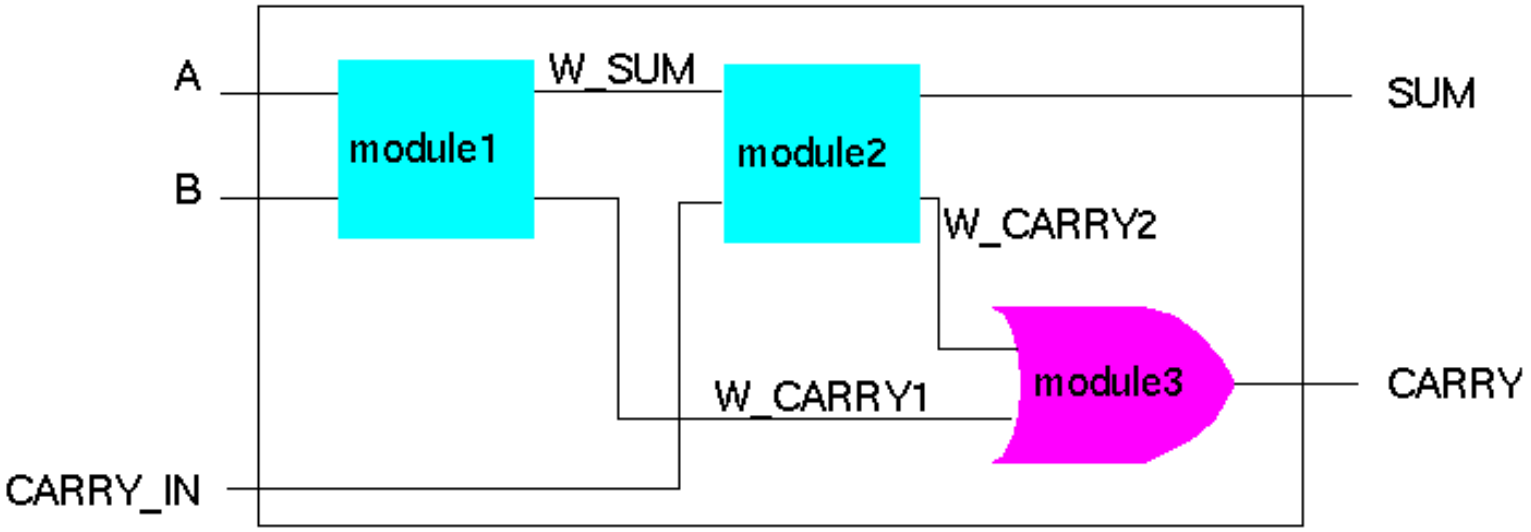
- **in:**
  - signal values are read-only
- **out:**
  - signal values are write-only
  - multiple drivers
- **buffer:**
  - similar to out
  - signal values may be read, as well
  - only 1 driver
- **inout:**
  - bidirectional port

- **Output port modes have to match.**

Use of VHDL Objects

Using Objects In VHDL		BODY					
		Concurrent			Sequential		
		Declare	Assign to	Use	Declare	Assign to	Use
OBJECT	Signal	YES	YES	YES	NO	YES	YES
	Variable	NO	NO	YES	YES	YES	YES
	Constant	YES	--	YES	YES	--	YES
	File	YES	--	YES	YES	--	YES

Structural Model



Full adder: 2 halfadders + 1 OR-gate

- توصیف ساختاری functionality را صریحاً توصیف نمی کند بلکه فقط لیستی از اجزا و نحوهٔ اتصال آنهاست.

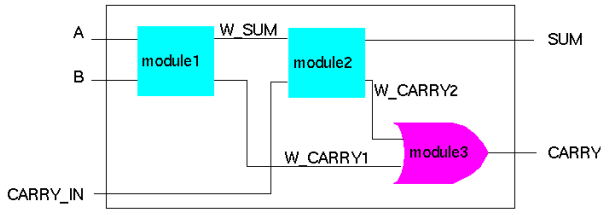
Component Declaration

```
entity FULLADDER is
  port (A,B, CARRY_IN: in   bit;
        SUM, CARRY:    out bit);
end FULLADDER;

architecture STRUCT of FULLADDER is

  component  HALFADDER
    port (A, B :              in   bit;
          SUM, CARRY : out bit);
  end component;

  component  ORGATE
    port (A, B : in   bit;
          RES : out bit);
  end component;
begin
  signal W_SUM, W_CARRY1, W_CARRY2 : bit;
. . .
```

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

```
entity ORGATE is
  port(
    A, B:              in   bit;
    RES: out bit);
end entity ORGATE;
```

- **In declarative part of architecture**
- **Can use any name for the components**
  - **but better to use the same as entity name**

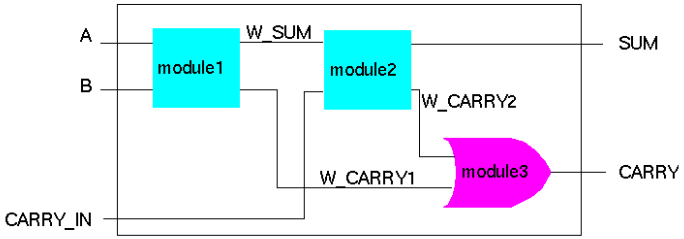


Full adder: 2 halfadders + 1 OR-gate

Component Instantiation

```
architecture STRUCT of FULLADDER is
  component HALFADDER
    port (A, B :            in   bit;
          SUM, CARRY : out bit);
  end component;
  component ORGATE
    port (A, B : in   bit;
          RES : out bit);
  end component;
  signal W_SUM, W_CARRY1, W_CARRY2: bit;
begin
    MODULE1: HALFADDER
    port map( A, B, W_SUM, W_CARRY1 );

   MODULE2: HALFADDER
    port map ( W_SUM, CARRY_IN,
               SUM, W_CARRY2 );

   MODULE3: ORGATE
    port map ( W_CARRY2, W_CARRY1, CARRY );
end STRUCT;
```

- **May need many of each**
- **Instantiation in statement part of architecture (after 'begin')**
- **Wires signals together:**
  - default: positional association

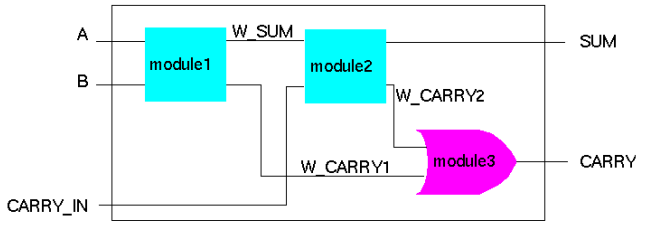


Full adder: 2 halfadders + 1 OR-gate

Named Signal Association

```
entity FULLADDER is
  port (A,B, CARRY_IN: in   bit;
        SUM, CARRY:     out bit);
end FULLADDER;

architecture STRUCT of FULLADDER is

  component HALFADDER
    port (A, B :              in bit;
          SUM, CARRY : out bit);
  end component;
. . .
  signal W_SUM, W_CARRY1, W_CARRY2 : bit;
begin
   MODULE1: HALFADDER
              port map (  A        => A,
                          SUM      => W_SUM,
                          B        => B,
                          CARRY    => W_CARRY1 );
  . . .
end STRUCT;
```

- **Named association:**
  - left side: "formals" (port names from component declaration)
  - right side: "actuals" (architecture signals)
- **Independent of order in component declaration**



Full adder: 2 halfadders + 1 OR-gate

Configuration

```
architecture …
begin
   component HALFADDER
    port(A, B:              in   bit;
         SUM, CARRY: out bit);
   end HALFADDER;

   signal W_SUM, W_CARRY1, W_CARRY2: bit;
for MODULE1: HALFADDER use entity work.HALFADDER (RTL);
for MODULE2: HALFADDER use entity work.HALFADDER (RTL);

   MODULE1 : HALFADDER
    port map(A, B, W_SUM, W_CARRY1);

   MODULE2: HALFADDER
    port map ( W_SUM, CARRY_IN,
               SUM, W_CARRY2 );
```

Library (work)

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

```
architecture RTL of HALFADDER is
begin
…
end architecture RTL;
```

```
entity ORGATE is
  port(
    A, B:              in   bit;
    RES: out bit);
end entity ORGATE;
```

```
architecture RTL of ORGATE is
begin
…
end architecture RTL;
```

Entities may be instantiated directly without preceding component declaration **(Not recommended)**

Configuration

- **Selects entity/architecture pairs for instantiated components**
- **Generates the hierarchy**
- **Default binding rules:**
  - selects entity with the same name as component
  - last compiled architecture is used

Simple Configuration Specification

```
entity FULLADDER is
  port(A, B, CARRY_IN: in   bit;
       SUM, CARRY:     out bit);
end FULLADDER;

architecture STRUCT of FULLADDER is
  component HA
    port(A, B:           in   bit;
         SUM, CARRY: out bit);
  end component;
  for MODULE1:HA
      use entity work.HALFADDER (RTL);
  for MODULE2:HA
      use entity work.HALFADDER (GATE);
signal W_SUM, W_CARRY1, W_CARRY2: bit;
  begin
  MODULE1:HA  port map (…, …);

  MODULE2:HA  port map(…, …);
end STRUCT;
```

Library (work)

```
entity HALFADDER is
   port(A,B:    in   bit;
        S, CY: out bit);
end A;
```

```
architecture RTL of
HALFADDER is
. . .
end RTL;
```

```
architecture GATE of
HALFADDER is
. . .
end GATE;
```

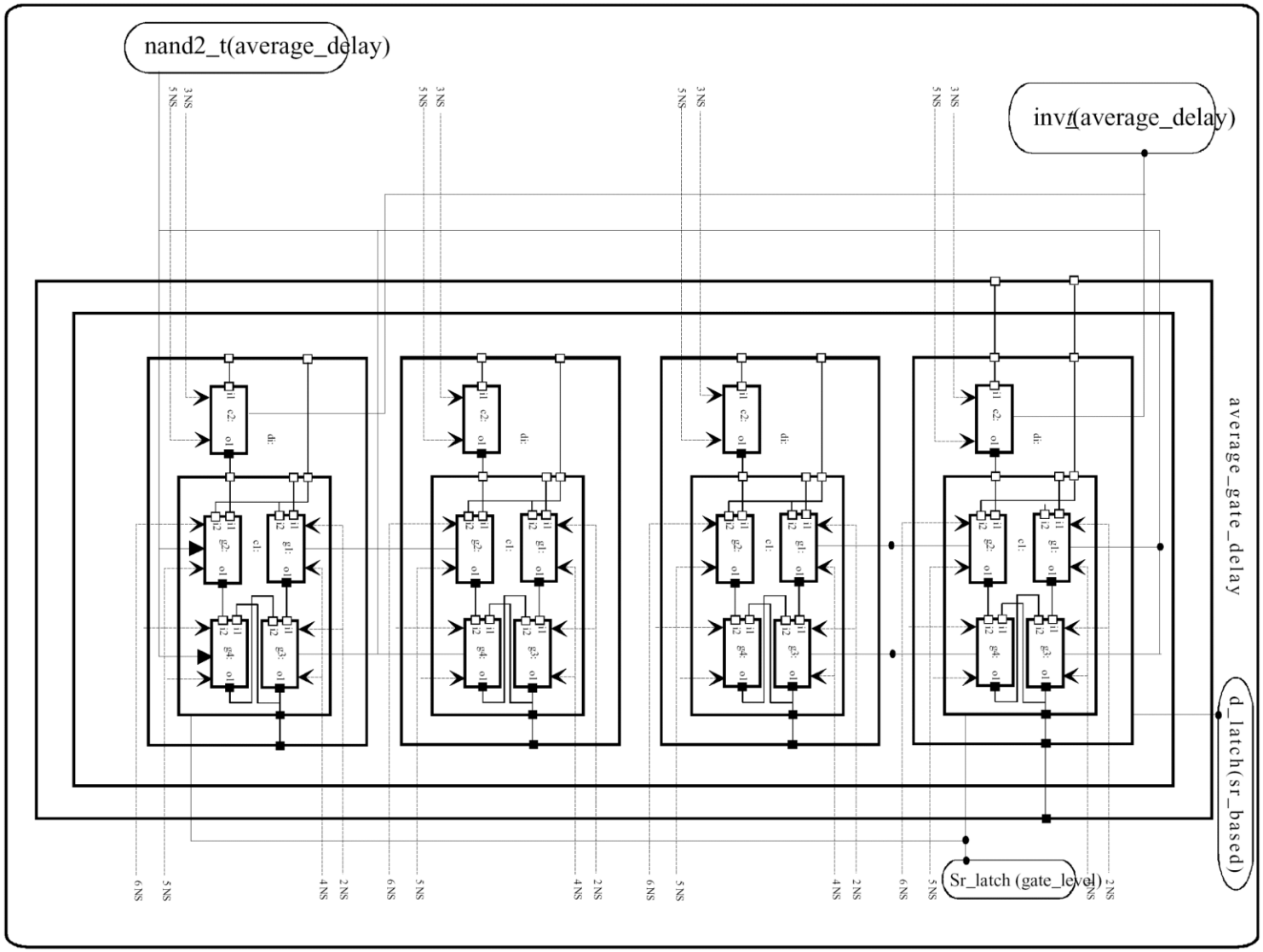
Configuration Example

```
configuration CONFNAME of ENTITYNAME is
    for ARCHNAME
        for INSTANCENAME  use entity LIBNAME.ENTNAME(ARNAME);
```

```
configuration CFG_FULLADDER of FULLADDER is
  for STRUCT
    for MODULE2:HA
      use entity work.HALFADDER(GATE);
    end for;

    for others:HA use entity work.HALFADDER(RTL);
    end for;
  end for;
end CFG_FULLADDER;
```

- **Entity/architecture pairs may be selected by use of**
  - instance names
  - 'all': all instances of the specified component
  - 'others': all instances not explicitly mentioned
- **Possible to reference an existing configuration of a submodule**

```
for all:FA use configuration work.CFG_FULLADDER;
```


nand2_t(average_delay)
invt(average_delay)
average_gate_delay
d_latch(sr_based)
Sr_latch (gate_level)

```
CONFIGURATION average_gate_delay OF d_register IS
  FOR latch_based
    FOR dr
      FOR di : dl
        USE ENTITY WORK.d_latch(sr_based);
        FOR sr_based
          FOR c1 : sr
            USE ENTITY WORK.sr_latch(gate_level);
          FOR gate_level
            FOR g2, g4 : n2
                USE ENTITY WORK.nand2_t(average_delay)
            END FOR;
            FOR g1, g3 : n2
                USE ENTITY WORK.nand2_t(less_delay)
            END FOR;
          END FOR;
        END FOR;
        FOR c2 : n1
           USE ENTITY WORK.inv_t(average_delay)
         END FOR;
      END FOR;
    END FOR;
  END FOR;
END FOR;
END average_gate_delay;
```

Configuration

- **سنتز:**
  - ابزارهای امروزی پشتیبانی می‌کنند
  - بعضی از ابزارهای سنتز جملات configuration را در نظر نمی گیرند و ممکن است قوانین پیش فرض را اعمال کنند.
    - در این ابزارها در declarationها:
      - نام componentها عین نام entity باشد
      - نام، مود و نوع portها عین پورت entity باشد