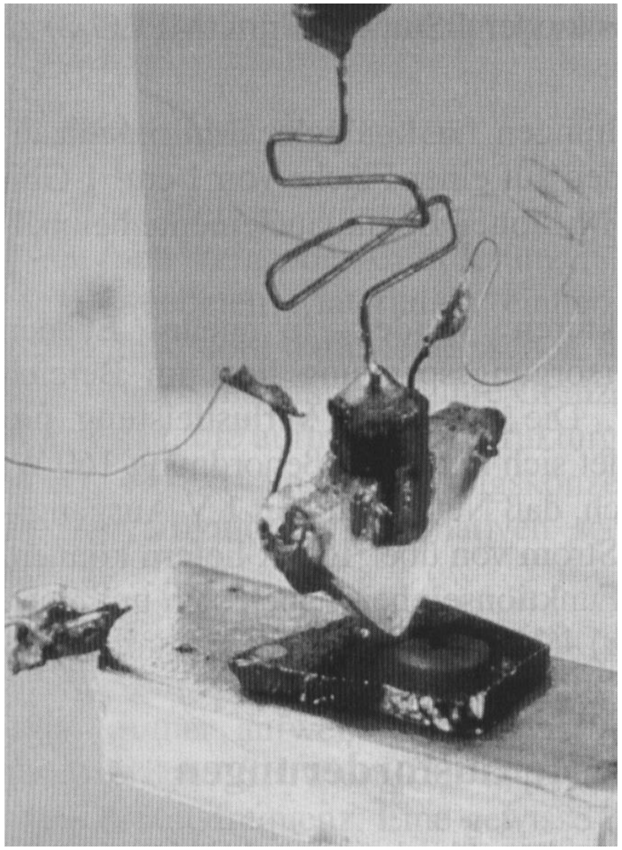
Introduction to VHDL I

Motivation – Entity – Architecture – Components - Instantiation - Examples

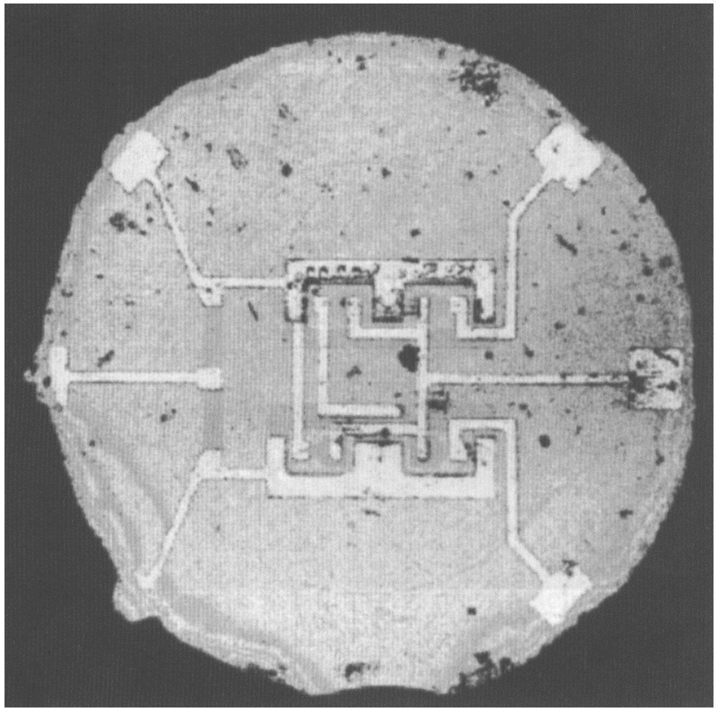
Roland Höller
Email: hoeller@technikum-wien.at

Motivation – First Transistor


[Sourcee: e&i 7/8/1998 S. 345, Springer Verlag, Wien]

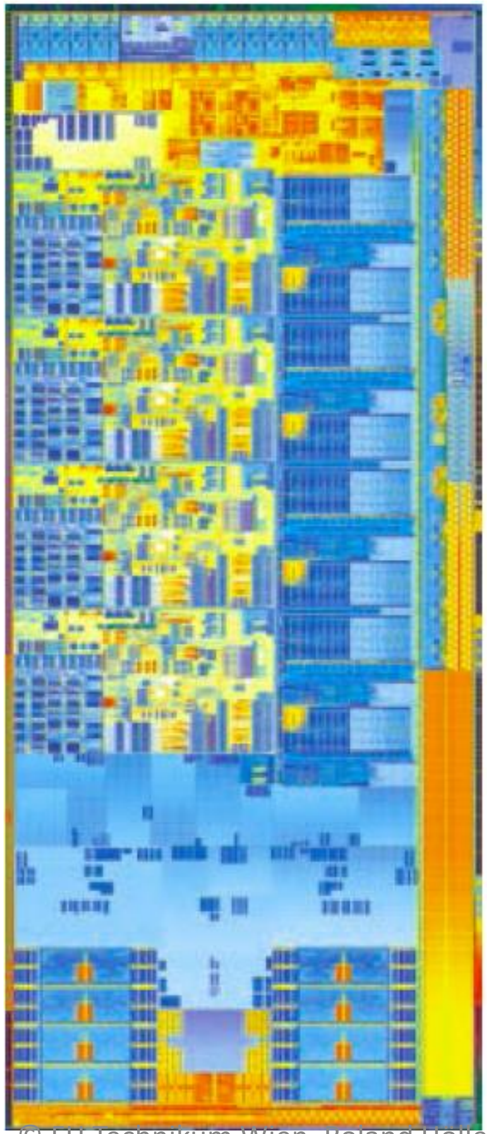
- 16th December 1947: first transistor
- (Bardeen, Brattain and
- Shockley - Bell Labs)

Motivation – First IC


[Source: e&i 7/8/1998 S. 345, Springer Verlag, Wien]

- One of the first ICs (Flip-Flop)

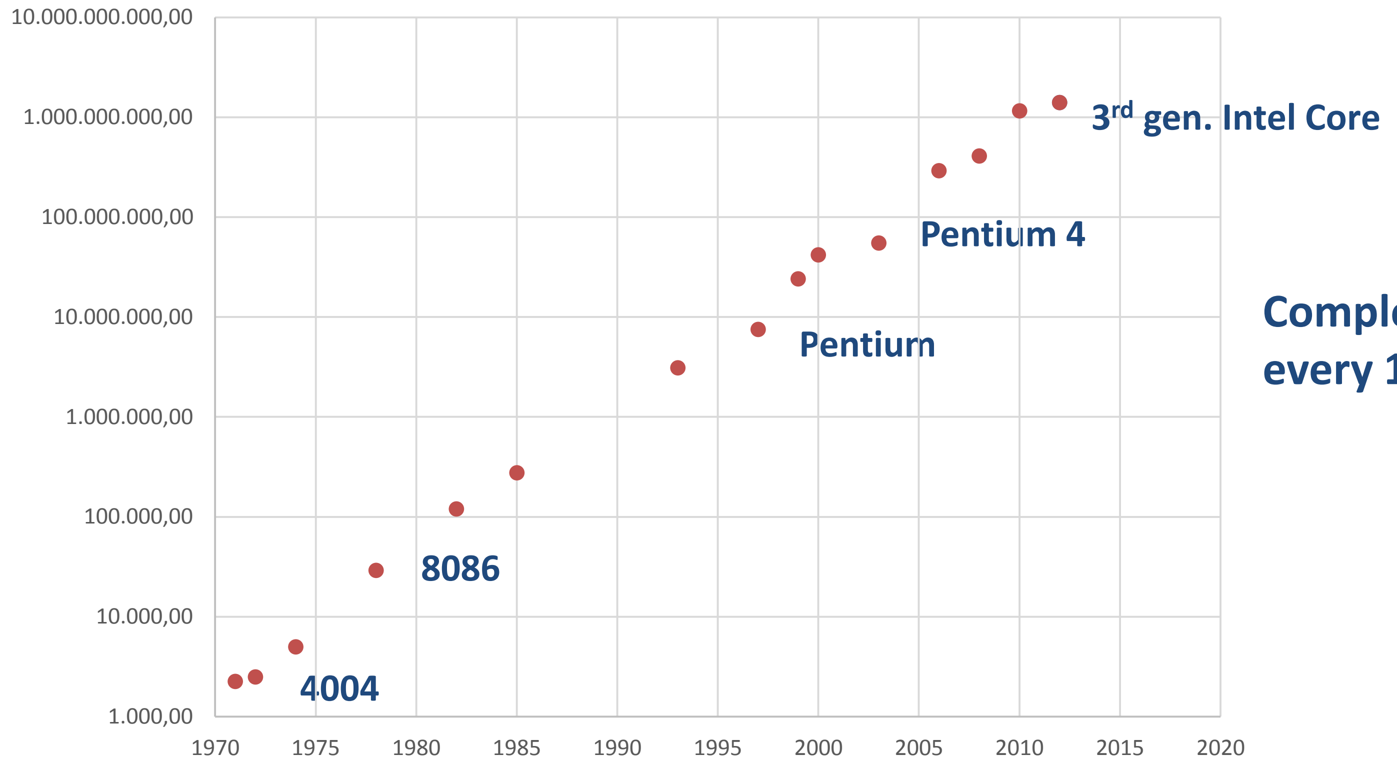
Motivation – State of the Art IC



- Modern Microprocessor (Intel Core):
  - approx. 1.4*10^7 transistors,
  - 22 nm CMOS technology,
  - 3-D Tri-Gate transistors

[Source: http://www.intel.com/content/www/us/en/history/history-intel-chips-timeline-poster.html]

Motivation - Moore's Law



[Source: http://www.intel.com/content/www/us/en/history/history-intel-chips-timeline-poster.html]

Complexity doubles every 18 months!

Motivation

Nowadays ASICs face:

- growing system complexity
- SoC
- ever shrinking time to market
- design reuse
- verification
- quality and reliability
- documentation

$\Rightarrow$ Use of HDLs

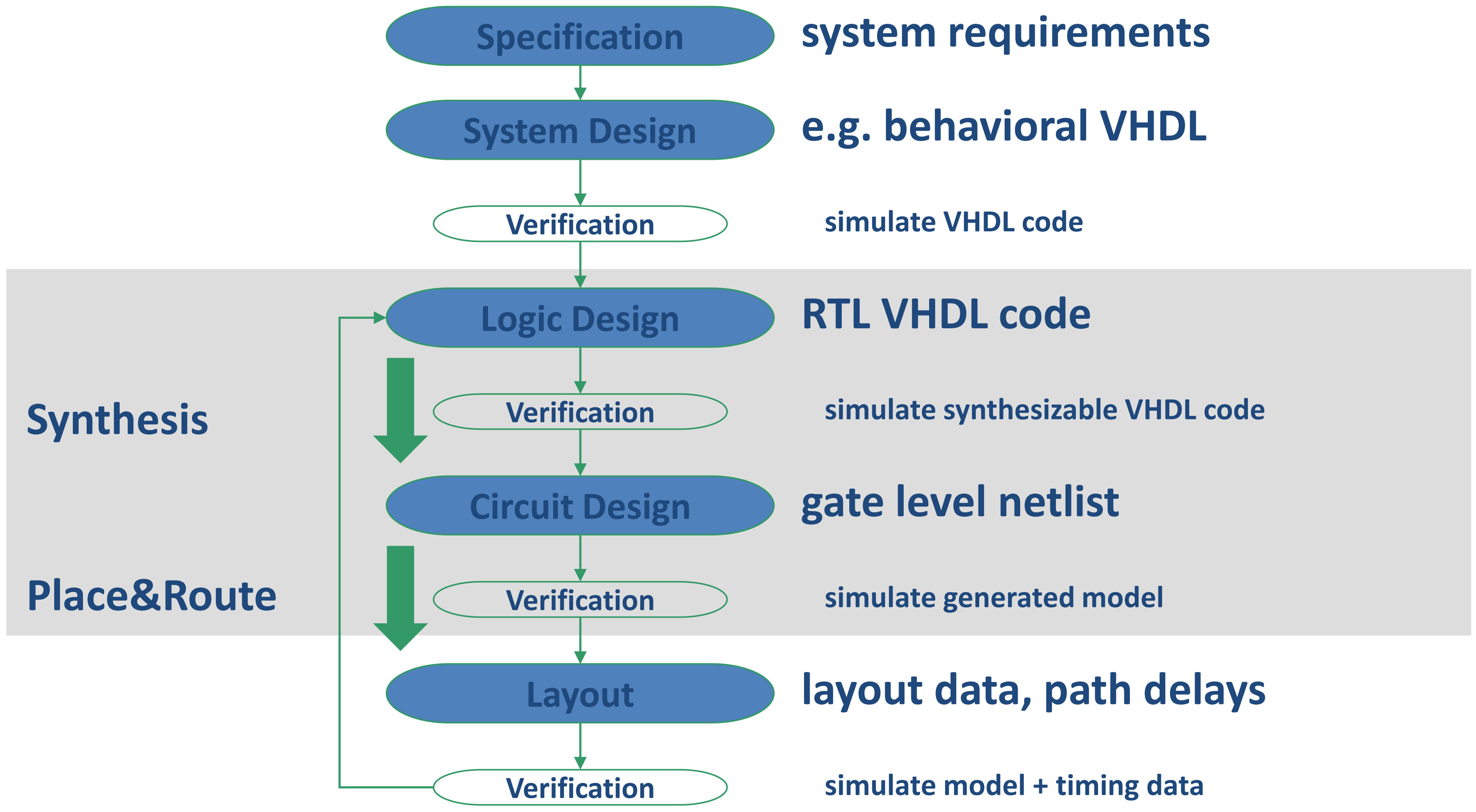
VHDL History

early ‘70s:	Initial discussion
late ‘70s:	Definition of requirements
mid - ’82:	Contract of development with IBM, Intermetrics and TI
mid - ‘86:	IEEE-Standard
1987:	DoD adopts the standard IEEE Std 1076-1987
mid - ‘88:	Increasing support by EDA tools
late ‘91:	Revision
1993:	New standard IEEE Std 1076-1993
1999:	VHDL-AMS IEEE Std 1076.1-1999
2000:	New standard IEEE Std 1076-2000

VHDL Application Field

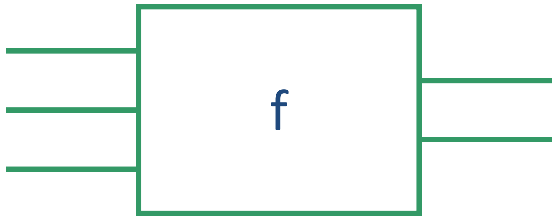
- IC Design (digital)
  - specification and documentation
  - modelling of digital systems
  - simulation with testbenches
  - ASIC-, FPGA-, CPLD-design
  - softcores
  - hardware/software co-design

HDL Design Flow

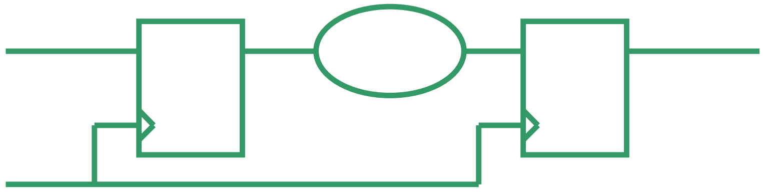


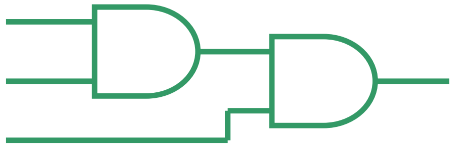
Levels of Abstraction

Behavioral Level

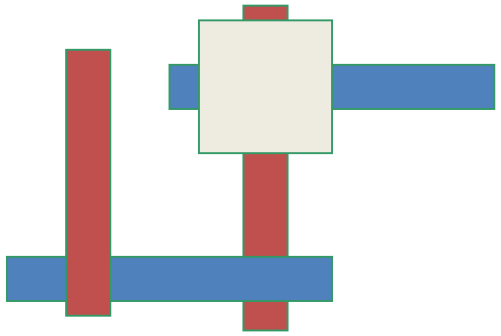


Register Transfer Level





Gate Level



Layout Level

Information in Abstraction Levels

fast simulation → slow simulation

		VHDL ?
Behavioral Level	functional and time behavior, NOT synthesizable	Yes
Register Transfer Level	clocks, function, events, NO times, synthesizable	Yes
Gate Level	gate delays	Yes
Layout Level	path delays, transistors, ...	No

VHDL Overview I

- VHDL properties
  - modelling of digital systems
  - man- and machine-readable specification
  - implies documentation
  - man- and machine-readable documentation
  - transportability
  - simulateable source code
  - strong type oriented, not case sensitive
  - concurrent and sequential statements

VHDL Overview II

- International Standards
  - IEEE Std 1076-1987
  - IEEE Std 1076-1993
  - IEEE Std 1076.1-1999 (AMS)
  - ....
  - IEEE Std 1076.6-1999 (RTL Synthesis)
  - IEEE Std 1076-2000 (LRM)
- Pure definition in the LRM
  - Language Reference Manual
  - No standards for application or methodology!

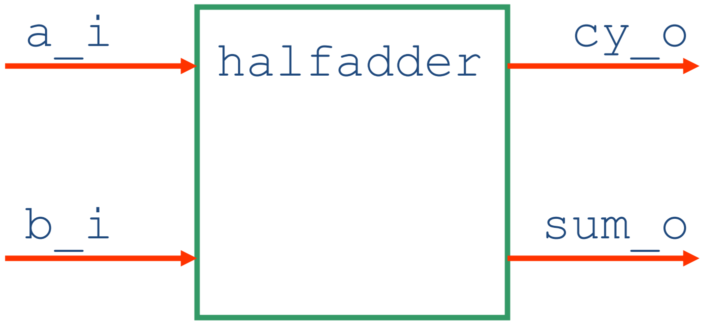
VHDL Constructs (VHDL 87/93/00)

Entity:	unit interface
Architecture:	unit behaviour, function
Configuration:	connects entity-architecture pairs
Library:	organize design units
Package:	data types, constants, components
Process:	concurrent, event controlled
Block:	concurrent, group assignments

The Entity

- Interface description
- No behavior or function
- Linking via ports (in, out, inout)
- Declared once within a design

```
entity halfadder is
  port (a_i :    in  std_logic;
        b_i :    in  std_logic;
        sum_o :  out std_logic;
        cy_o :   out std_logic);
end halfadder;
```

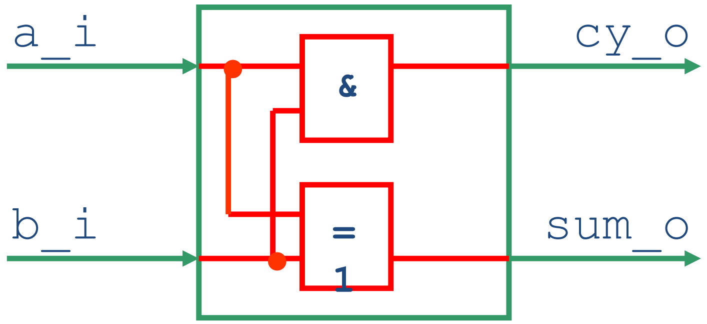


The Architecture

- Implementation of the design
- Describes circuitry or function
- Different architectures for one entity

```
architecture rtl of halfadder is
begin
   sum_o <= a_i xor b_i;
   cy_o  <= a_i and b_i;
end rtl;
```

Coding Styles:
- rtl
- behavioral
- structural
- sim



Circuit Representations

```
architecture rtl of halfadder is

begin
  sum_o <= a_i xor b_i;
  cy_o  <= a_i and b_i;
end rtl;
```

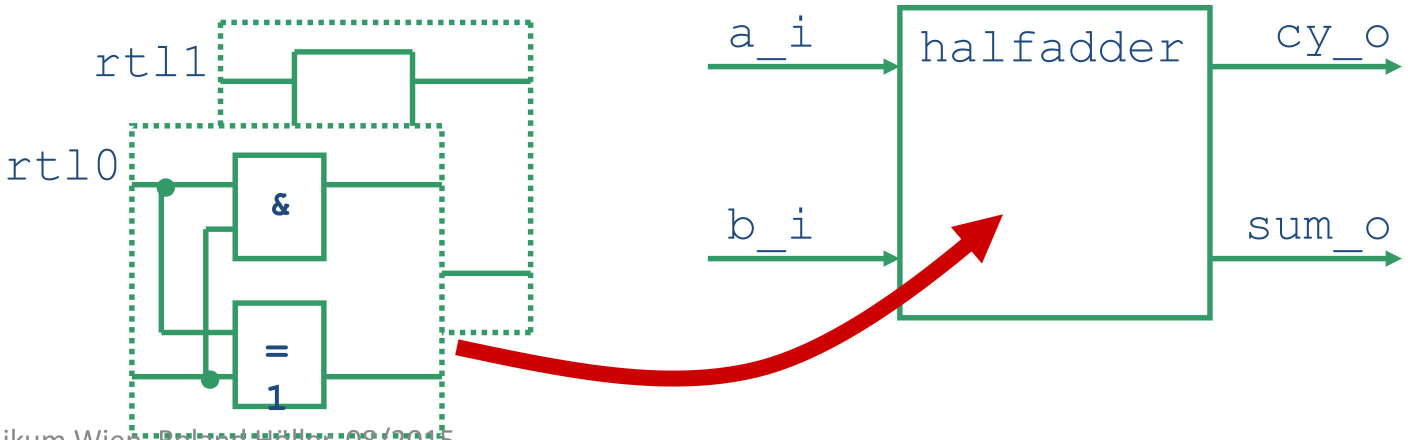
```
architecture behavioral of halfadder is

begin
  sum_o <= a_i xor b_i after 5 ns;
  cy_o  <= a_i and b_i after 4 ns;
end behavioral;
```

```
architecture structural of halfadder is

begin
  i_xor_gate : xor_gate
    port map (opa_i => a_i,
              opb_i => b_i,
              res_o => sum_o);

end structural;
```

The Configuration

- Declaration of entity-architecture pairs for simulation
- Default configuration is most recently analyzed
- Non-default configuration sometimes necessary
- Entity and component names are identical (default)

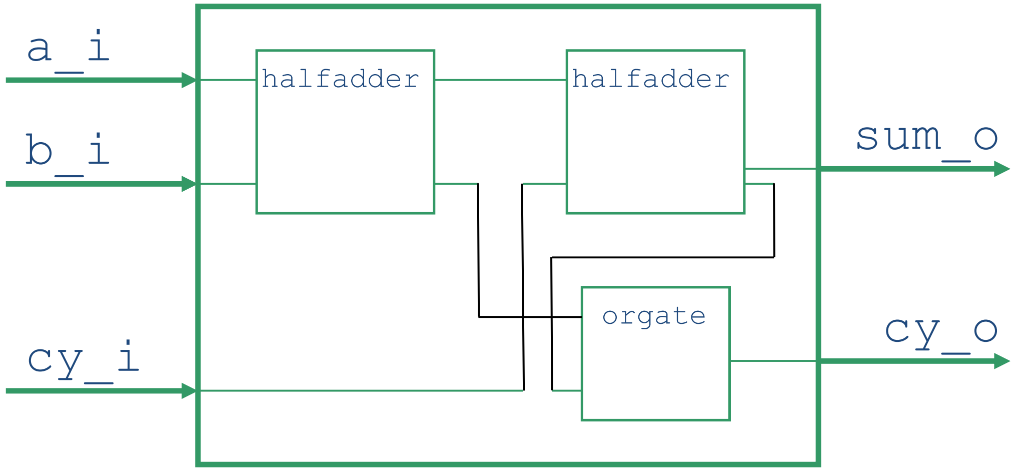
```
configuration halfadder_rtl_cfg of
halfadder is
  -- architecture rtl is used
  -- for entity halfadder
  for rtl
  end for;
end halfadder_rtl_cfg;
```



The Component Declaration

- In advance of usage in an architecture
- Comparable to a „socket“

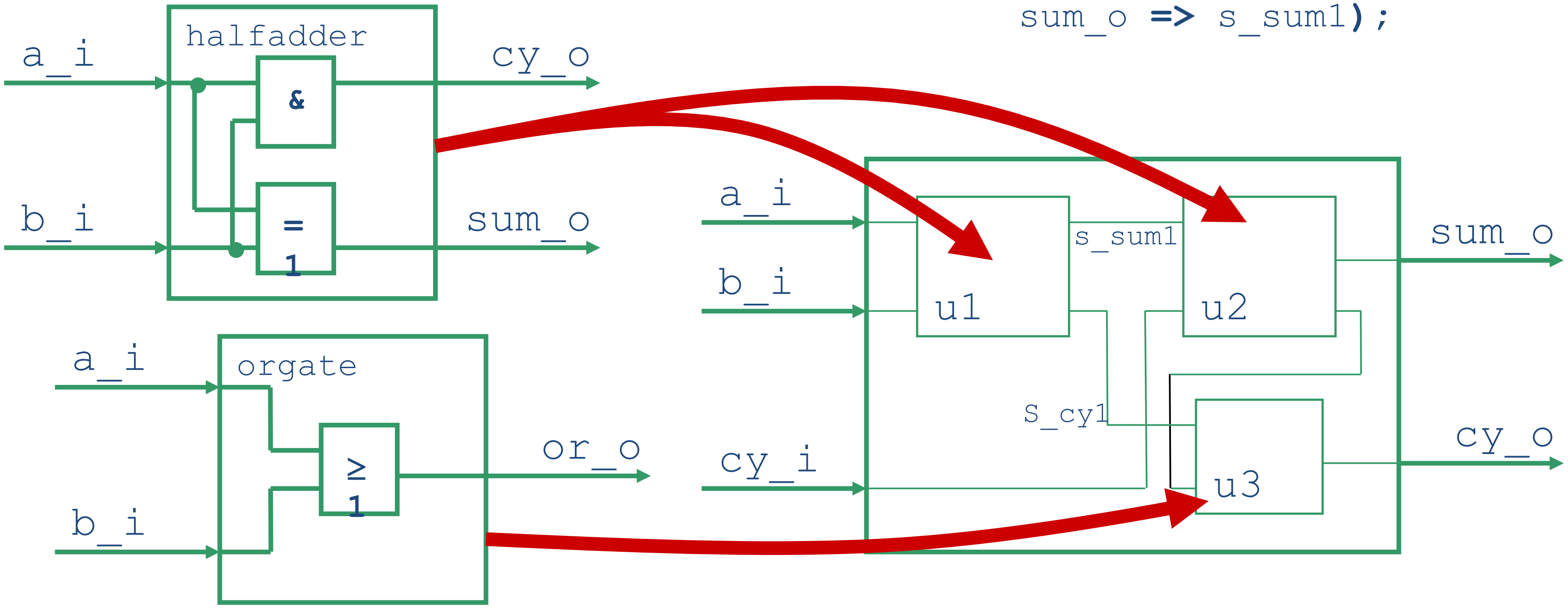
```
architecture struct of fulladder is
  component halfadder
    port (a_i    : in  std_logic;
          b_i    : in  std_logic;
          cy_o   : out std_logic;
          sum_o  : out std_logic);
  end component;
begin
```



The Component Instantiation

- Wiring of the modules
- After architecture begin

```
begin
u1 : halfadder
   port map
      (a_i   => a_i,
       b_i   => b_i,
       cy_o  => s_cy1,
       sum_o => s_sum1);
```



The Half Adder

```
entity halfadder is
  port (a_i :   in  std_logic;  -- operand a
        b_i :   in  std_logic;  -- operand b
        sum_o : out std_logic;  -- sum of a and b
        cy_o :  out std_logic); -- carry
end halfadder;

architecture rtl of halfadder is
begin
  -- generate the output sum_o
  sum_o <= a_i xor b_i;
  -- generate the output cy_o
  cy_o  <= a_i and b_i;
end rtl;

configuration halfadder_rtl_cfg of halfadder is
  -- architecture rtl is used
  -- for entity halfadder
  for rtl
  end for;
end halfadder_rtl_cfg;
```

The Or Gate

```
entity orgate is
  port (a_i :   in  std_logic;   -- operand a
        b_i :   in  std_logic;   -- operand b
        or_o :  out std_logic);  -- output
end orgate;

architecture rtl of orgate is
begin
  -- generate the output or_o
  or_o <= a_i or b_i;
end rtl;

configuration orgate_rtl_cfg of orgate is
  -- architecture rtl is used
  -- for entity orgate
  for rtl
  end for;
end orgate_rtl_cfg;
```

The Full Adder I

```
entity fulladder is
  port (a_i :   in  std_logic;   -- operand a
        b_i :   in  std_logic;   -- operand b
        cy_i :  in  std_logic;   -- carry input
        cy_o :  out std_logic;   -- carry output
        sum_o : out std_logic);  -- output
end fulladder;

architecture structural of fulladder is

  component halfadder
    port (a_i    : in  std_logic;
          b_i    : in  std_logic;
          cy_o   : out std_logic;
          sum_o  : out std_logic);
  end component;
```

The Full Adder II

```
  component orgate
    port (a_i  : in  std_logic;
          b_i  : in  std_logic;
          or_o : out std_logic);
  end component;

  signal s_sum1 : std_logic;
  signal s_cy1  : std_logic;
  signal s_cy2  : std_logic;

begin

u1 : halfadder
  port map                 -- named association
    (a_i   => a_i,
     b_i   => b_i,
     cy_o  => s_cy1,
     sum_o => s_sum1);
```

The Full Adder III

```
u2 : halfadder
  port map
    (a_i   => s_sum1,
     cy_o  => s_cy2,
     b_i   => cy_i,
     sum_o => sum_o);

u3 : orgate
  port map
    (s_cy2, s_cy1, cy_o);

end structural;

configuration fulladder_struc_cfg of fulladder is
  -- architecture structural is used
  -- for entity fulladder
  for structural
  end for;
end fulladder_struc_cfg;
```

named association:
- independent of order
- more readable

positional association:
- order defines connection
- less readable

incomplete configuration
- just binds E/A for top level
- relies on MRA for subunits

Summary

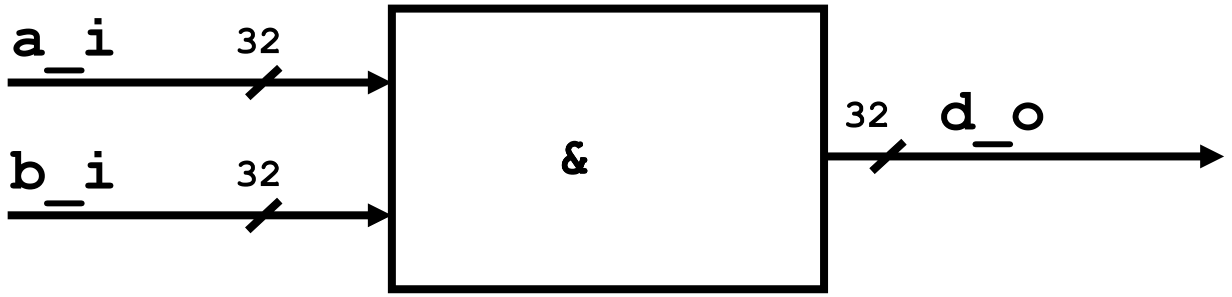
- Motivation
- Design Flow
- Abstraction Levels
- VHDL Overview
- Entity
- Architecture
- Configuration
- Component Declaration
- Example : Full Adder

Questions

- 1) VHDL is used in which phases of a digital circuit design flow?
  - A ASIC-, FPGA-, or PLD-Design only
  - B Description of a system before partitioning into hardware and software
  - C Description of hardware only
  - D All of the abovementioned items
- 2) Which statements are correct?
  - A VHDL is perfectly suited for the description of analog systems
  - B Different synthesis tools support different VHDL language subsets
  - C For the description of a testbench, VHDL coding on RTL is mandatory
  - D All VHDL code, that can be simulated, can also be synthesized
- 3) How many architectures can be associated with a VHDL entity?
  - A One or more
  - C Exactly one
  - B More than one
  - D None
- 4) In which part of a VHDL architecture do signals have to be declared, that shall only be visible locally in that architecture?
  - A In the port list of the corresponding entity
  - C In the architecture after the begin
  - B In the corresponding configuration
  - D In the architecture before the begin
- 5) What do the entity and the corresponding component declaration in most of the cases have in common?
  - A Nothing
  - C Entity and Component different names, but the same ports
  - B Entity and Component have the same name, but different ports
  - D Entity and Component the same name and the same ports

Class Example: AND 2 data busses

1) Create and simulate the design



```
library IEEE;
use IEEE.std_logic_1164.all;

entity vectorgates is
  port (a_i :   in  std_logic_vector(31 downto 0);      -- operand a
        b_i :   in  std_logic_vector(31 downto 0);      -- operand b
        d_o :   out std_logic_vector(31 downto 0));    -- output
end vectorgates;

architecture rtl of vectorgates is
begin
  -- generate the output d_o
  d_o <= a_i and b_i;
end rtl;
```

2) „AND“ high word and „OR“ low word

3) Add c_i, d_i of type std_logic and “XOR” them to the output e_o.