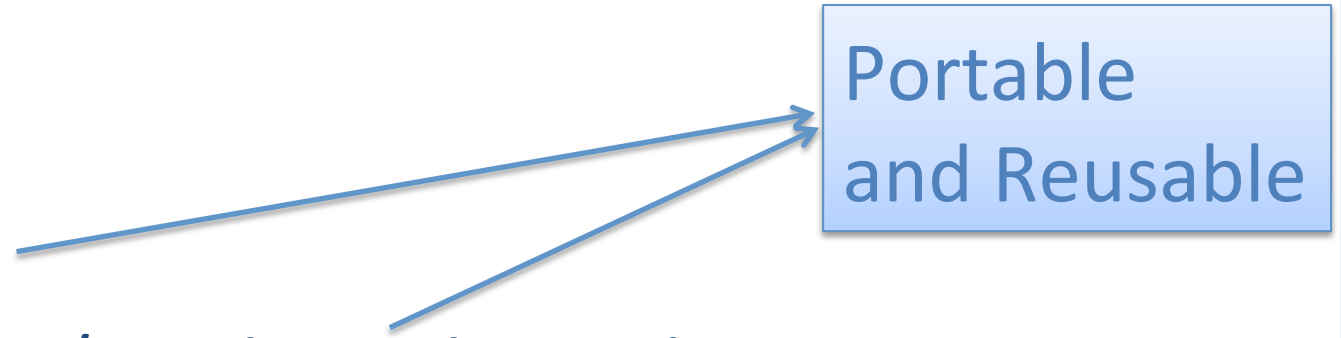
GONZAGA UNIVERSITY

VHDL: Code Structure

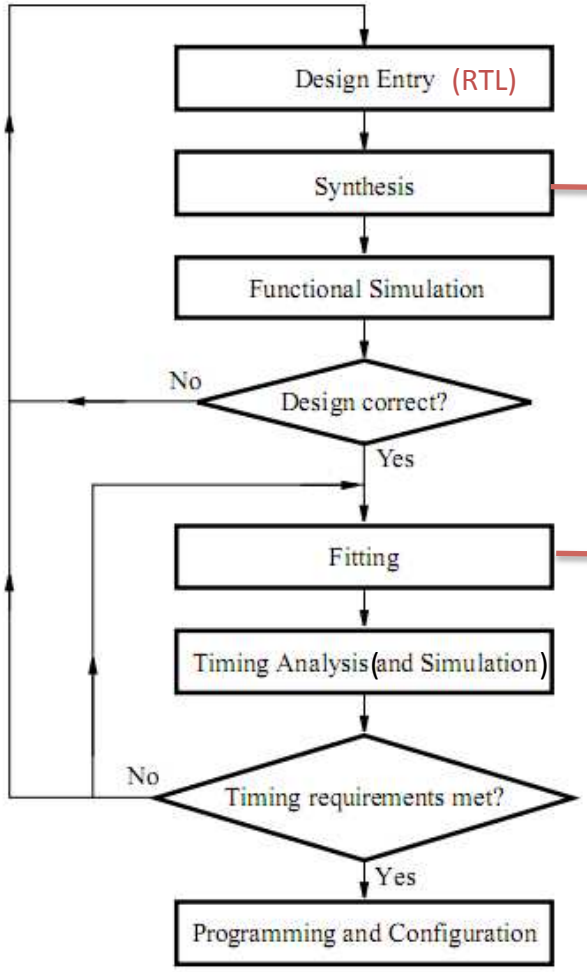
Motivation for HDL-based design

- Standard
- Technology/vendor independent



Altera's Design Flow

Typical Verification:
* Simulation
 - HDL Beh. Testbench
* STA
* Formal Verification



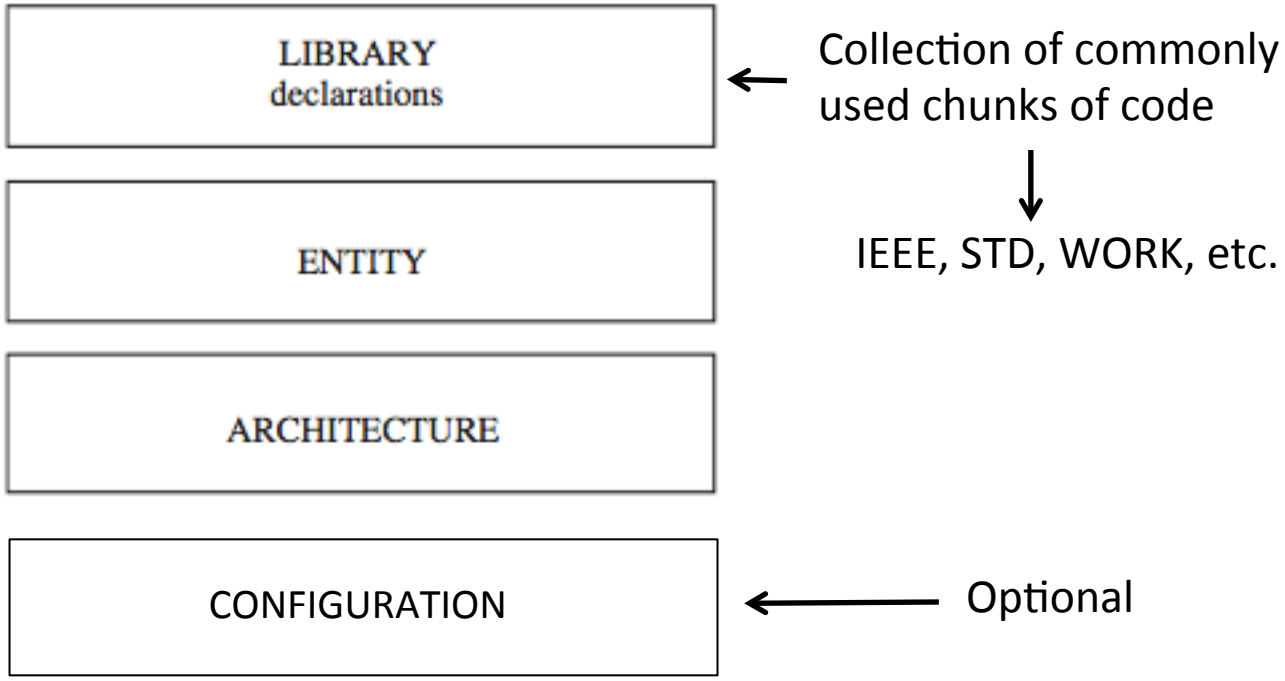
RTL
↓
Generic Boolean Eqns.
↓
Optimization to target technology

Place & Route

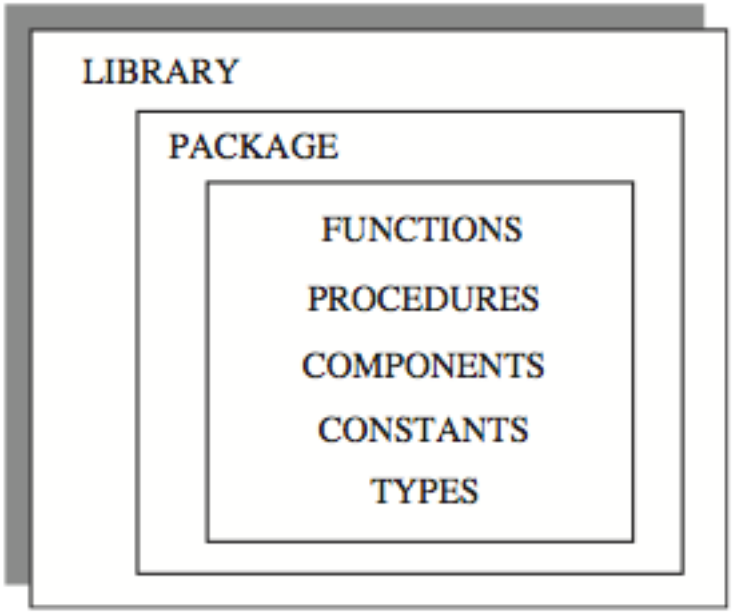
Static Timing Analysis = STA

GONZAGA UNIVERSITY

Fundamental VHDL Units



Fundamental parts of a Library



- **std**:
  resource library (basic data types, operators, etc.). It contains the packages **standard** and **textio**.
- **work**:
  denotes the current working library (all "compiled" code gets saved here)

Example of Library Declaration:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

The libraries std and work are made visible by default, so there is no need to declare them

```
LIBRARY std;
USE std.standard.all;

LIBRARY work;
USE work.all;
```

GONZAGA UNIVERSITY

Common packages of the IEEE library

- std_logic_1164
  - multi-level logic system [std_logic (8 levels) and std_ulogic (9 levels)]

UNRESOLVED

```
'X' Strong Unknown
'0' Strong Low
'1' Strong High
'Z' High Impedance
'W' Weak Unknown
'L' Weak Low
'H' Weak High
'-' Don't Care
```

STD_LOGIC and STD_LOGIC_VECTOR

Table 3.1
Resolved logic system (STD_LOGIC).

	X	0	1	Z	W	L	H	-
X	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
-	X	X	X	X	X	X	X	X

Multiple Drivers Resolution System

```
'U' unresolved
'X' Strong Unknown
'0' Strong Low
'1' Strong High
'Z' High Impedance
'W' Weak Unknown
'L' Weak Low
'H' Weak High
'-' Don't Care
```

STD_ULOGIC and STD_ULOGIC_VECTOR

GONZAGA UNIVERSITY

Common packages of the IEEE library

- std_logic_arith (it calls std_logic_1164)
  - Defines signed and unsigned data types, operations and conversion functions
    - conv_integer(p)

it returns an integer

integer, unsigned, signed, std_ulogic

    - conv_unsigned(p,b)

it returns an unsigned of size b bits

integer, unsigned, signed, std_ulogic

    - conv_signed(p,b)

it returns a signed of size b bits

integer, unsigned, signed, std_ulogic

    - conv_std_logic_vector(p,b)

it returns a std_logic_vector of size b bits

integer, unsigned, signed, std_ulogic

GONZAGA UNIVERSITY

Common packages of the IEEE library

- std_logic_signed (it calls std_logic_arith)
  - Contains functions that allow operations with std_logic_vector data to be performed as if the data were of type signed.
  - it overloads conv_integer(p)

it returns an integer

std_logic_vector

GONZAGA UNIVERSITY

Common packages of the IEEE library

- std_logic_unsigned (it calls std_logic_arith)
  - Contains functions that allow operations with std_logic_vector data to be performed as if the data were of type unsigned.
  - it overloads conv_integer(p)

it returns an integer

std_logic_vector

NOTE: cannot use std_logic_unsigned and std_logic_signed packages simultaneously

GONZAGA UNIVERSITY

Common packages of the IEEE library

- numeric_std (it calls std_logic_1164)
  - alternative package to std_logic_arith and std_logic_unsigned (or std_logic_arith and std_logic_signed)
  - Defines signed and unsigned data types, operations and conversion functions
  - numeric_std does not attempt to imply a numerical interpretation on std_logic_vector (SLV)

Example:
SLV input port and SLV output port

```
output <= std_logic_vector(unsigned(input) + 1)
```

Coding Example #1: DFF with Asynchronous RESET

GONZAGA UNIVERSITY

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

entity DFF_asy is
  port (
    D   : in  std_logic;
    RST : in  std_logic;
    CLK : in  std_logic;
    Q   : out std_logic);
end DFF_asy;

architecture RTL of DFF_asy is
begin
  DFF_p : process(RST, CLK)
  begin
    if RST = '1' then
      Q <= '0';
    elsif CLK'event and CLK = '1' then
      Q <= D;
    end if;
  end process DFF_p;
end architecture RTL;
```

Port modes:
IN, OUT, ~~INOUT~~, ~~BUFFER~~

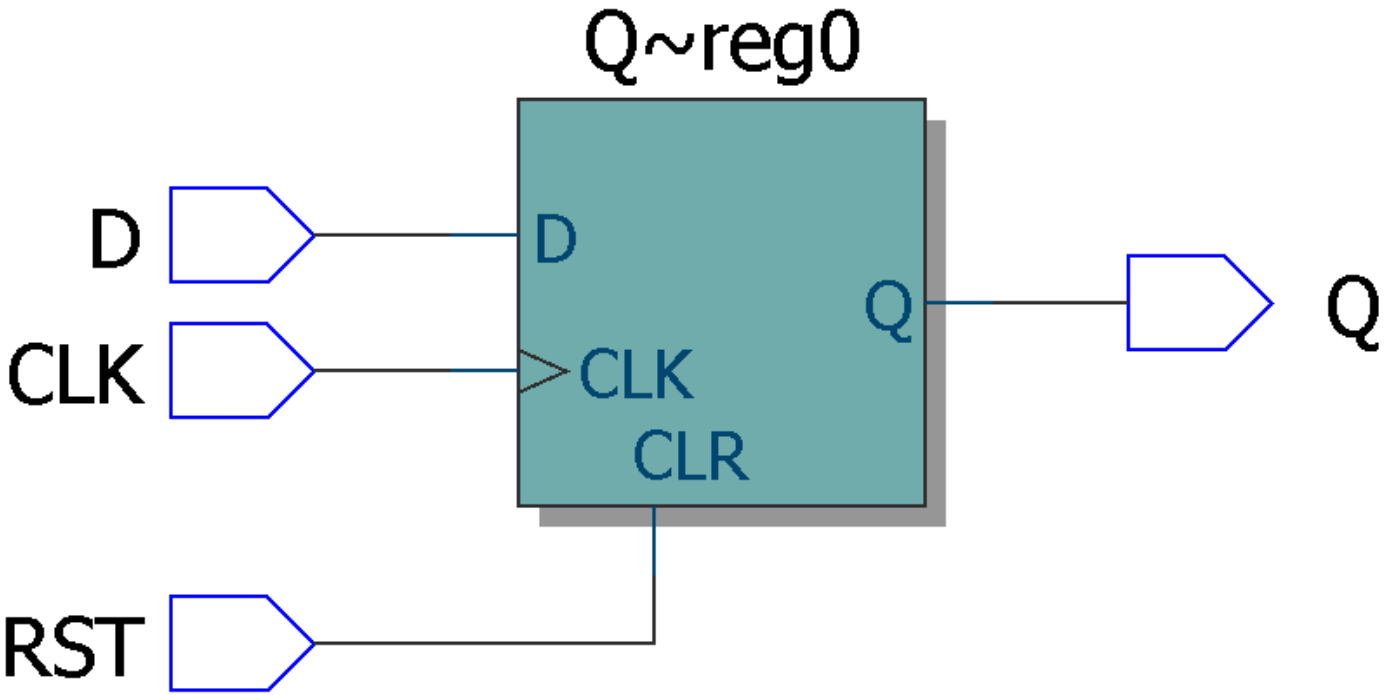
Data type associated to the signal at the port:
std_logic, std_logic_vector
(DO NOT USE bit and bit_vector !)

abstraction level of coding style

- The code inside **process** is executed sequentially (NOTE: this has nothing to do with sequential or combinational nature of the logic implemented)
- The process is executed every time a signal in its sensitivity list changes

Assignment types:
<= signals
vs.
:= variables

DFF with Asynchronous RESET



Coding Example #2: DFF with Synchronous RESET

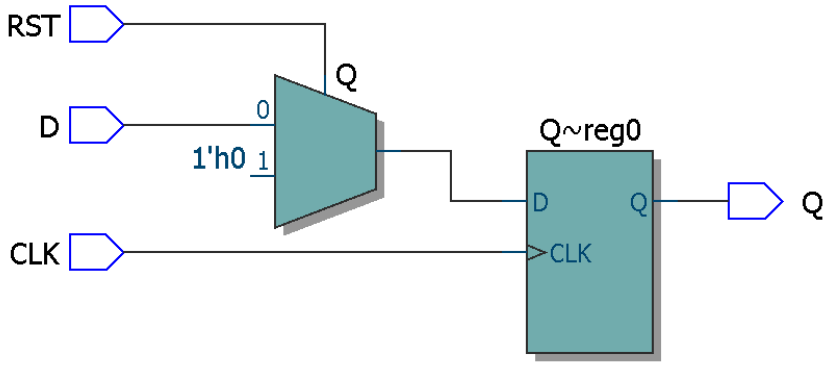
GONZAGA UNIVERSITY

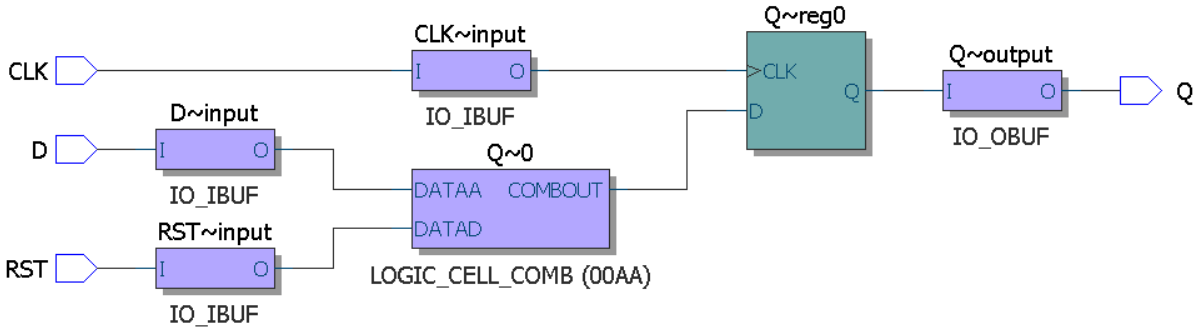
```
library IEEE;
use IEEE.std_logic_1164.all;

entity DFF_syn is
  port (
    D   : in   std_logic;
    RST : in   std_logic;
    CLK : in   std_logic;
    Q   : out  std_logic);
end DFF_syn;

architecture RTL of DFF_syn is
begin
   DFF_p : process(CLK)
   begin
     if CLK'event and CLK = '1' then
        if RST = '1' then
          Q <= '0';
        else
          Q <= D;
        end if;
     end if;
   end process DFF_p;
end architecture RTL;
```

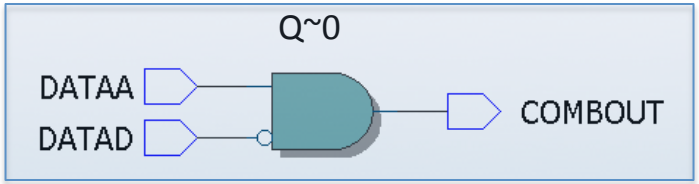
- The code inside **process** is executed sequentially (NOTE: this has nothing to do with sequential or combinational nature of the logic implemented)
- The process is executed every time a signal in its sensitivity list changes

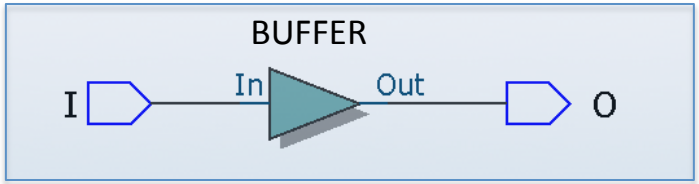
Assignment types:
<= signals
vs.
:= variables

DFF with Synchronous RESET









GONZAGA UNIVERSITY

Coding Example #3: counter

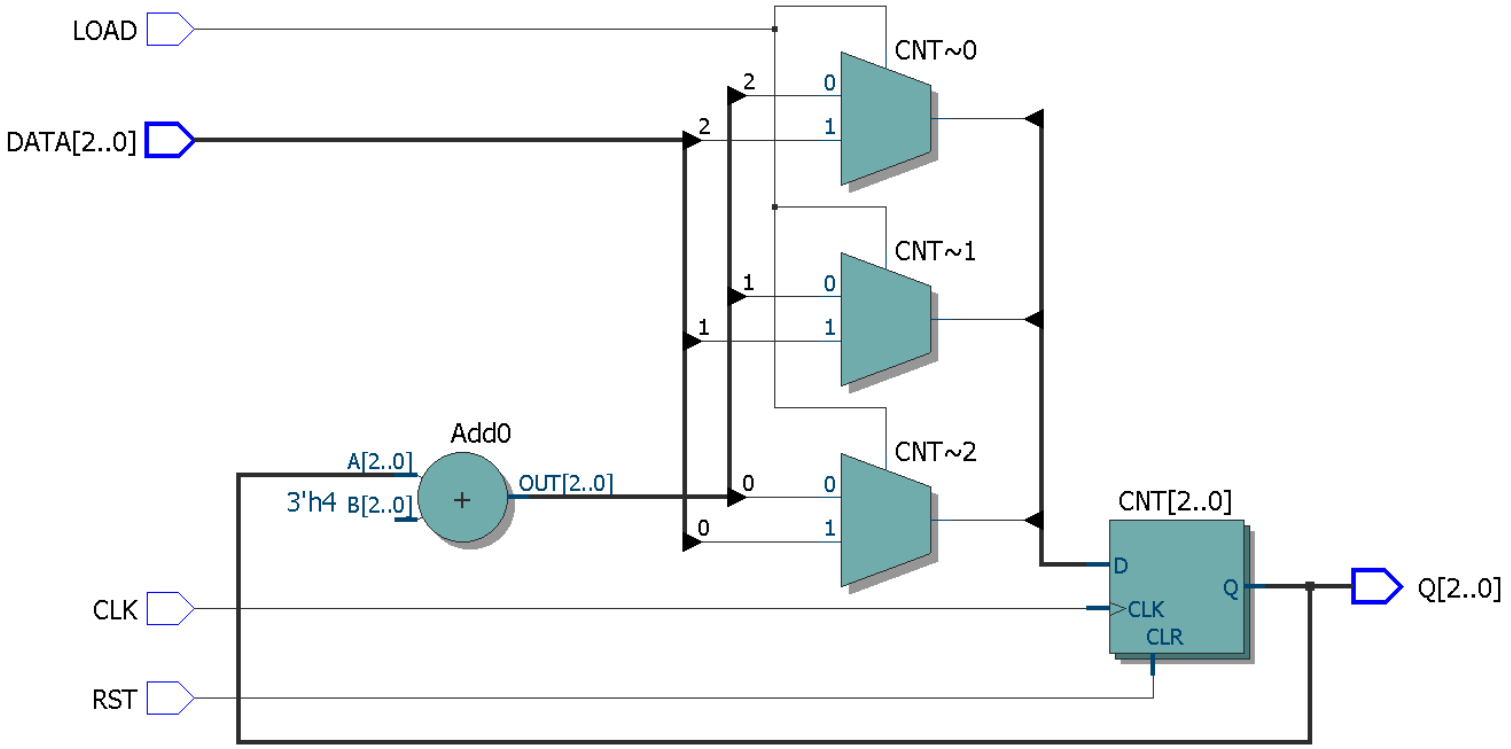
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;    -- for the unsigned type

entity COUNTER is
  generic (
    WIDTH : in natural := 3);
  port (
    RST   : in std_logic;
    CLK   : in std_logic;
    LOAD  : in std_logic;
    DATA  : in std_logic_vector(WIDTH-1 downto 0);
    Q     : out std_logic_vector(WIDTH-1 downto 0));
end entity COUNTER;
```

```
architecture RTL of COUNTER is
  signal CNT : unsigned(WIDTH-1 downto 0);
begin
  process(RST, CLK) is
  begin
    if RST = '1' then
      CNT <= (others => '0');
    elsif rising_edge(CLK) then
      if LOAD = '1' then
        CNT <= unsigned(DATA); -- converted to unsigned
      else
        CNT <= CNT + 1;
      end if;
    end if;
  end process;

  -- type is converted back to std_logic_vector
  Q <= std_logic_vector(CNT);
end architecture RTL;
```

sequential code

concurrent code

Counter


LOAD
DATA[2..0]
CLK
RST
CNT~0
CNT~1
CNT~2
Add0
A[2..0]
3'h4 B[2..0]
OUT[2..0]
CNT[2..0]
D
CLK
CLR
Q
Q[2..0]

GONZAGA UNIVERSITY

Counter