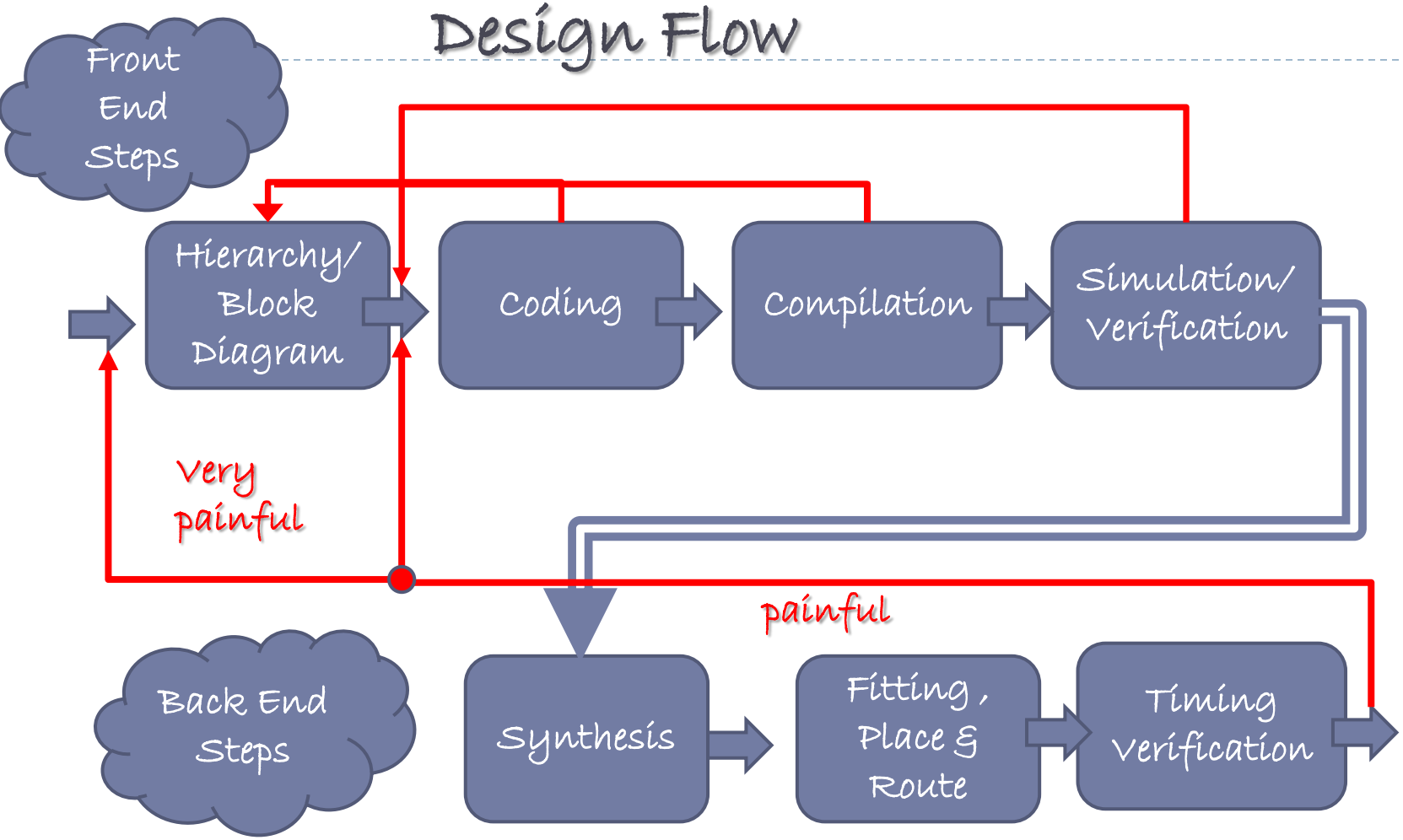
Digital design using HDL

Magdy Saeb

Design Flow


Front End Steps
Hierarchy/ Block Diagram
Coding
Compilation
Simulation/ Verification
very painful
painful
Back End Steps
Synthesis
Fitting, Place & Route
Timing Verification

Design Entry Techniques

- When using Schematic → Circuit
- When using VHDL → Algorithm
- When using ASM → Concept

Design Environment

- Specialized Text-editor:

with built-in templates for frequently used program structures, automatic indenting, syntax checking and compilation

- Simulation:

Test bench that automatically apply inputs

- Functional Verification:

Logical operation is checked independetly of timing consideration; gate delays and other parameters are also considered.

- Timing verification:

Delays are considered and setup time and hold times in sequential (flip-flops) are considered.

- Synthesis:

Converting VHDL description statements into a set of primitives or components that can assembled in the target technology. For example with PLD or CPLD the synthesis tool may generate two-level sum-of-products equations.

- Netlist: with ASIC the synthesizer may generate a list of gates (netlist) that specifies how they are interconnected.

- Fitting:

The fitting tool (fitter) maps the synthesized primitives to available design resources. For example PLD or CPLD; it means assigning equations to avaialble AND-OR elements and for ASIC it means laying down individual gates and finding a way to connect them (Place & Route).

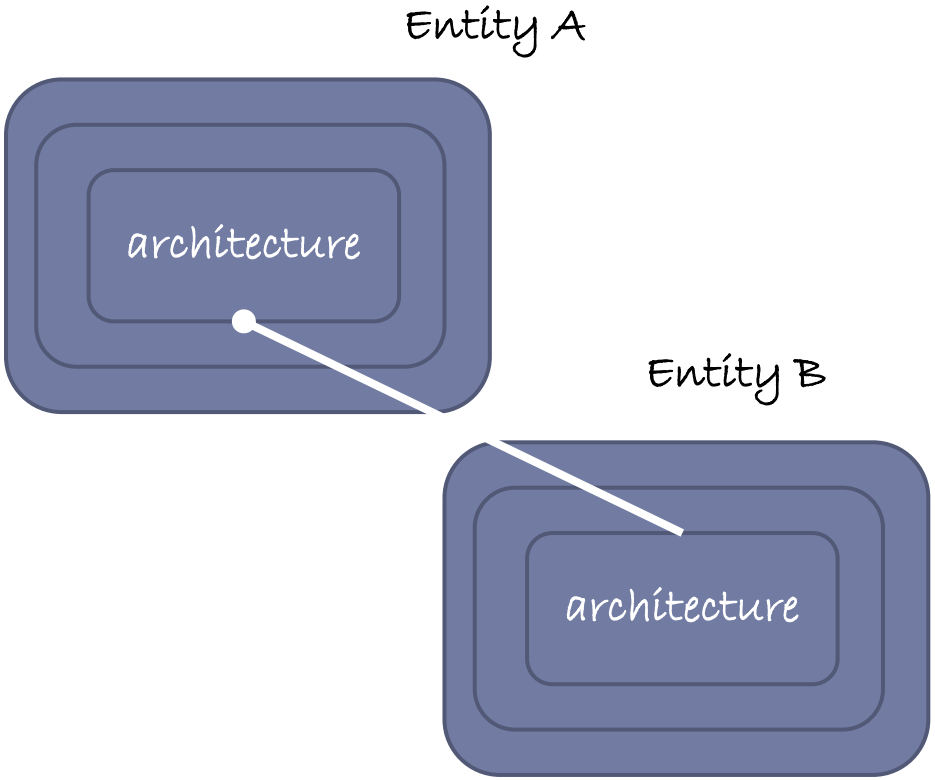
- Timing |Verification:

Actual delays and loading effects are considered.

Program Structure

Pascal + Ada = VHDL

Wrapper concept



VHDL Programming Styles

- Structural:

(AND/OR) Defines explicit component and their connections between them. It is a textual equivalent to drawing a schematic.

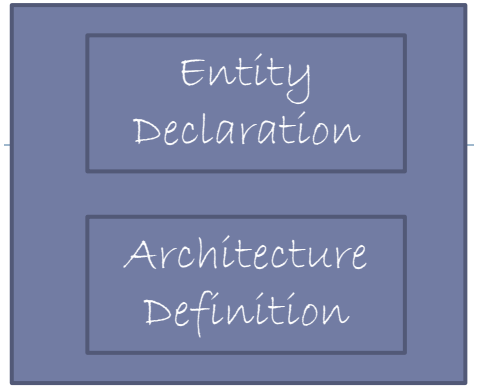
- DataFlow:

(Case, When, Select) Assigns expression to signals

- Behavorial:

(For Simulation) Writes an algorithm that describes a circuit output that may not be implementable.

VHDL Program Structure



Entity name

Signal names

mode

type

Architecture name

```
entity Inhibit is
  port ( X, Y : in BIT; Z:out BIT);
  end Inhibit;
  architecture Inhibit_arch of Inhibit is
  begin
    Z<= '1' when X ='1' and Y = '0' else '0';
  end Inhibit_arch;
```