从电路设计的角度入门VerilogHDL

邸志雄，西南交通大学

www.dizhixiong.cn

参考资料

轻松成为设计高手:VerilogHDL实用精解. EDA先锋工作室. 北京航空航天大学出版社. 2012年.

中国大学MOOC平台：芯动力——硬件加速设计方法

https://www.icourse163.org/course/SWJTU-1207492806

CONTENTS

目录

VerilogHDL历史

Design ×

硬件**描述**语言： Hardware **Description** Language

VerilogHDL由Cadence发明，以全定制电路设计方法学的挑战者横空出世，在过去的几十年间，经历了逐步完善、标准制订等一系列成长、成熟的历程。

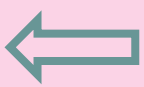
随着芯片的规模逐渐进入数亿晶体管的时代，近年来屡屡受到HLS、Chisel等语言的挑战。

但是，迄今为止，仍然是描述**数字电路和数字逻辑系统**的**主流**语言

VerilogHDL历史

使用If-else等较为高级的表达

方式来描述**电路**

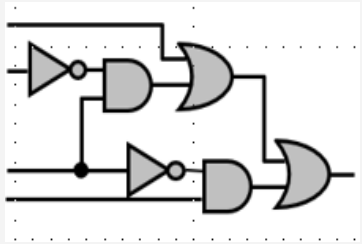


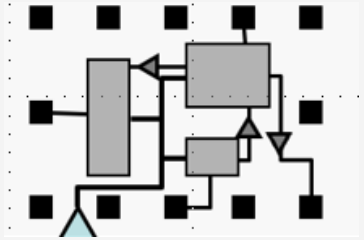
算法级、RTL级

可综合语句，Verilog很小的子集

构建测试环境：用于产生逻辑电路工作所需的信号，只要符合语法即可

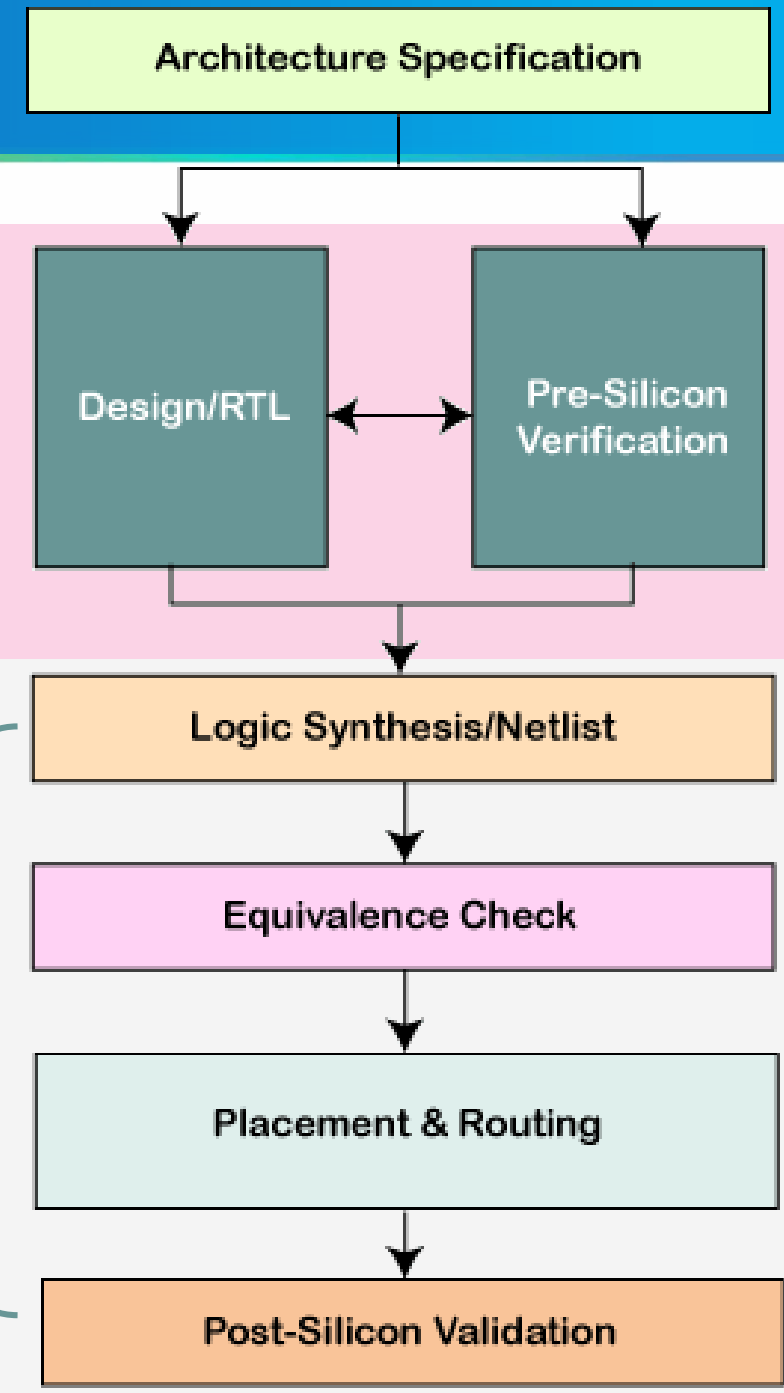
描述逻辑门（如与门、非门、或门、与非门、三态门等）以及逻辑门之间**连接**的模型





门级（Gate Level）

VerilogHDL具备从“抽象表达”到“门级连接”的多层次表征的能力



代码与原理图输入

HDL 和原理图是两种最常用的数字硬件电路描述方法。

原理图设计输入法在早期应用得比较广泛，它会根据设计要求，选用器件，绘制原理图，完成输入过程。

这种方法的优点是直观，便于理解，元件库资源丰富;但是在大型设计中。

这种方法的可维护性较差，不利于模块建设与重用，更主要的缺点是，当所选用的芯片升级换代后，所有的原理图都要作相应的改动。

代码与原理图输入

因此，推荐初学者在描述和仿真数字电路时首选 HDL 语言方式，而在某些要求使用图形描述设计顶层的情况下才使用原理图，不要在设计顶层以外的其他层次使用原理图。

另外，不要依赖波形设计工具，因为简单的信号虽然用波形描述起来十分方便，但是复杂的测试激励几乎无法使用波形工具进行有效描述。

相关EDA工具

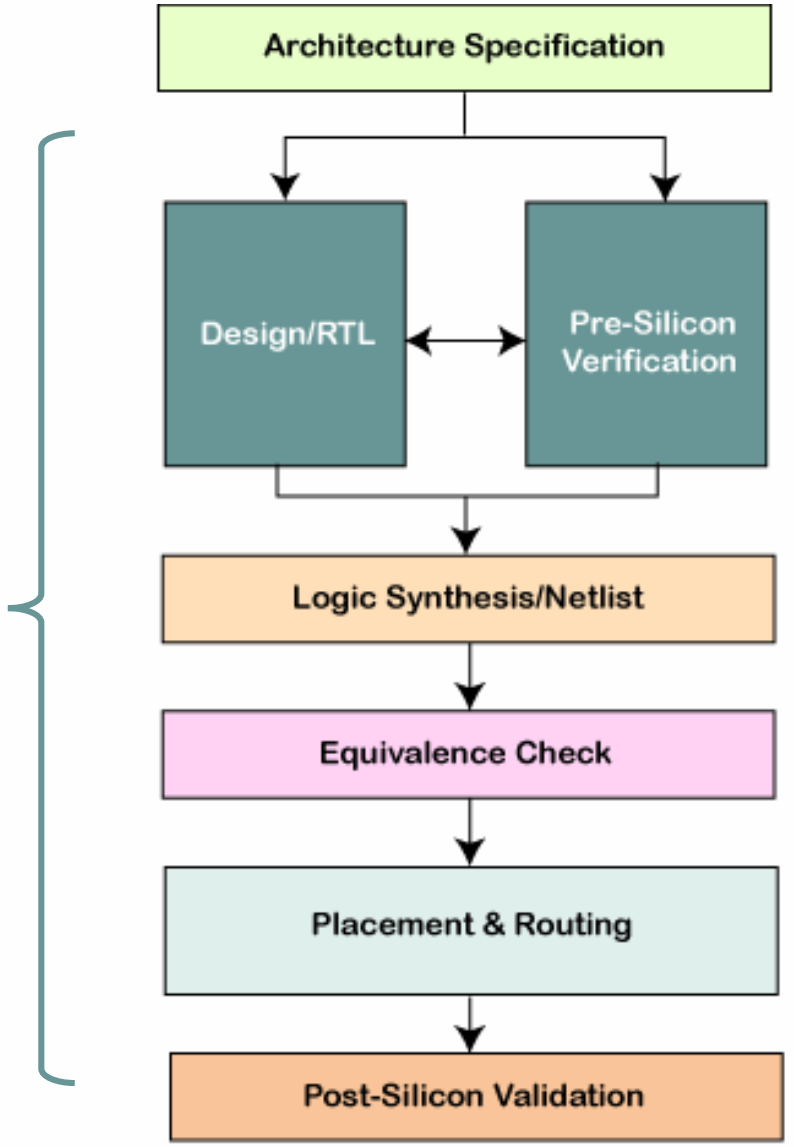
阶段	开源	国外	国产
编辑	Emacs、Vim、neovim、ultraedit、vscode 等		
仿真	i-verilog、verilator、GTK-Wave	VCS、Xcelium、Modelsim、Verdi；Vivado、Quartus等	芯华章：穹鼎GalaxSim、昭晓Fusion Debug 合见工软：UniVista Simulator、UniVista Debugger
逻辑综合	Yosys	DesignCompiler、Genus；Vivado、Quartus等	安路TangDynasty(TD)、高云、紫光同创

VerilogHDL的缺陷

VerilogHDL具备从**“抽象表达”**到“门级连接”的多层次表征的能力

- 例化繁琐，需要手动连线
- 大量中间信号需要声明
- 模块参数能力弱（使用宏，parameter不支持函数）
- 错误检查能力弱（需要EDA工具的支持）
- 基础电路需要重复写
- 电路复用程度低
- …

错误检测能力弱
参数化能力弱
代码复用低



VerilogHDL的挑战者

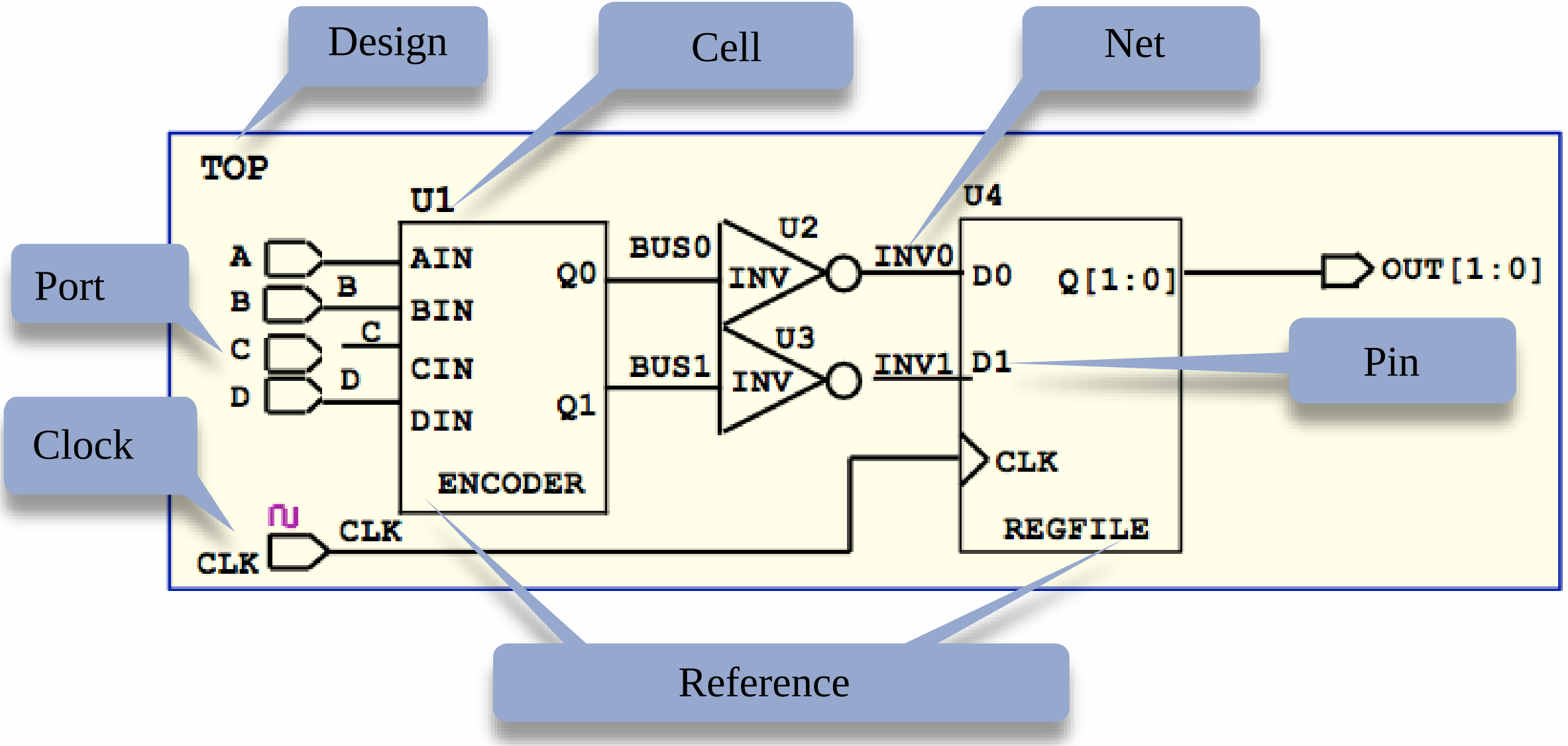
- C++
  - SystemC - an IEEE standard meta-HDL
  - VisualHDL - an integrated development environment (IDE) rapid design for FPGAs
  - **HLS**
- Haskell
  - concat - Haskell to hardware, 2016+
  - CλaSH - functional hardware description language
  - pipelineDSL - A Haskell DSL for describing hardware pipelines
  - Bluespec - Compiler, simulator, and tools for the Bluespec Hardware Description Language
- Java
  - jhdl 2006
  - PSHDL
- JavaScript
  - reqack - elastic circuit toolchain
  - hdl-js - Hardware description language (HDL) parser, andHardware simulator
  - shdl - Simple Hardware Description Language
- Julia
  - Julia-Verilog - a Verilog-generation DSL for Julia, 2017
- scala
  - **Chisel** – Meta HDL, 2014+
  - **SpinalHDL** – Meta HDL, 2014+
- Python
  - HWT - Meta HDL, verification env. IP-core generator, HDL glue
  - garnet - Coarse-Grained Reconfigurable Architecture generator based on magma, 2018+
  - magma - Meta HDL, 2017+
  - migen - Meta HDL, 2011+
  - MyHDL - Process based HDL, verification framework included, 2004+
  - nMigen - A refreshed Python toolbox for building complex digital hardware, 2018+
  - Pyrope - Python-like language supporting “fluid pipelines” and “live flow”,2017+
  - PyRTL - Meta HDL, simulator suitable for research
  - PyMTL - Process based HDL, verification framework included, 2014+
  - veriloggen - Python, meta HDL with HLS like features, 2015
- Ruby
  - RHDL
- Rust
  - hoodlum – Meta HDL, 2016+
  - kaze – Meta HDL, 2019+

CONTENTS

目录

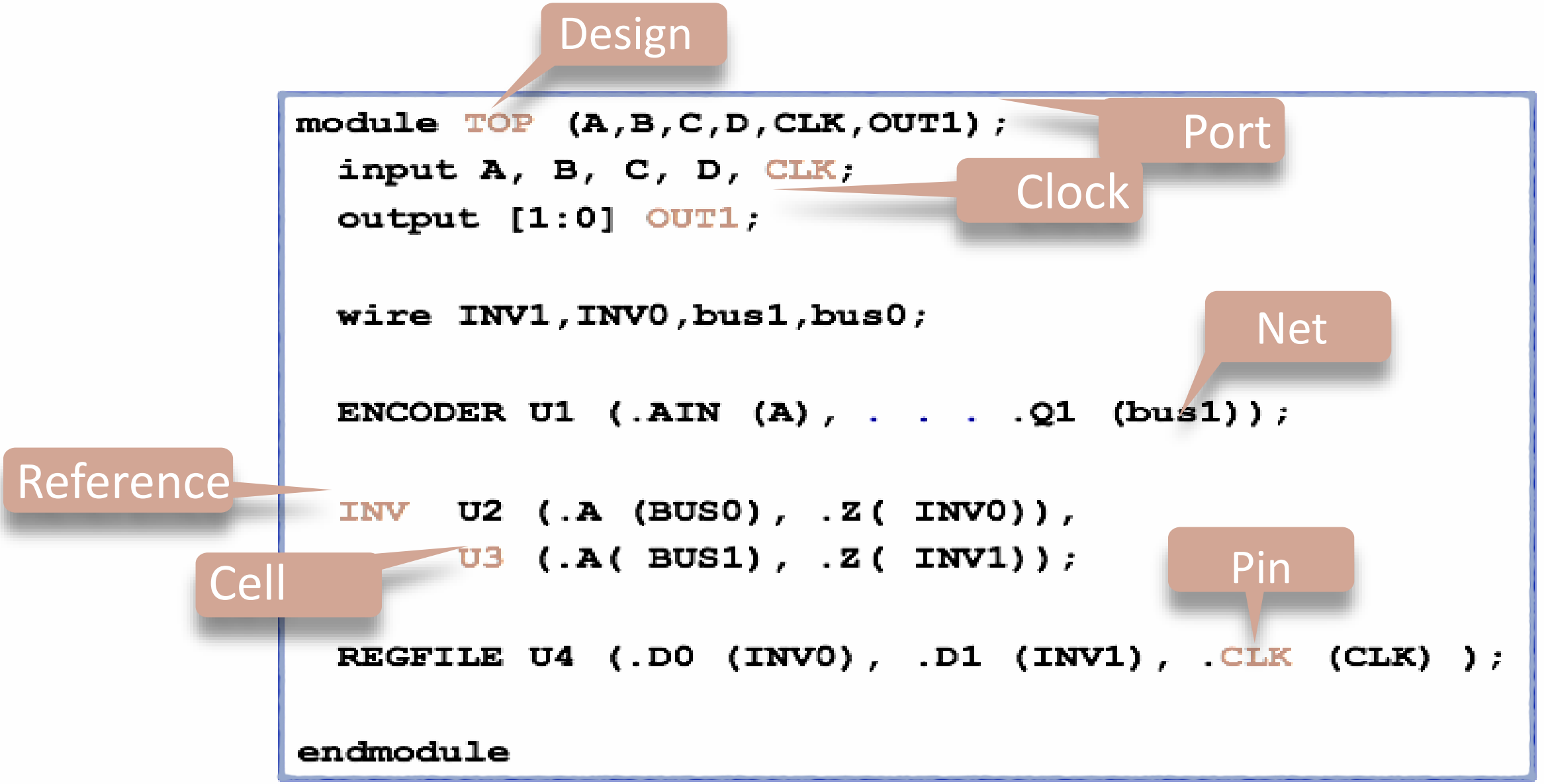
知识点列表

电路设计对象



Verilog不是编程语言；用来“描述”电路，先确定（想好）电路设计再用Verilog表达出来

代码描述对象


Design
Port
Clock
Net
Reference
Cell
Pin
module TOP (A,B,C,D,CLK,OUT1);
input A, B, C, D, CLK;
output [1:0] OUT1;
wire INV1,INV0,bus1,bus0;
ENCODER U1 (.AIN (A), ...Q1 (bus1));
INV U2 (.A (BUS0), .Z(INV0)),
U3 (.A(BUS1), .Z(INV1));
REGFILE U4 (.D0 (INV0), .D1 (INV1), .CLK (CLK));
endmodule

与 C 语言的区别

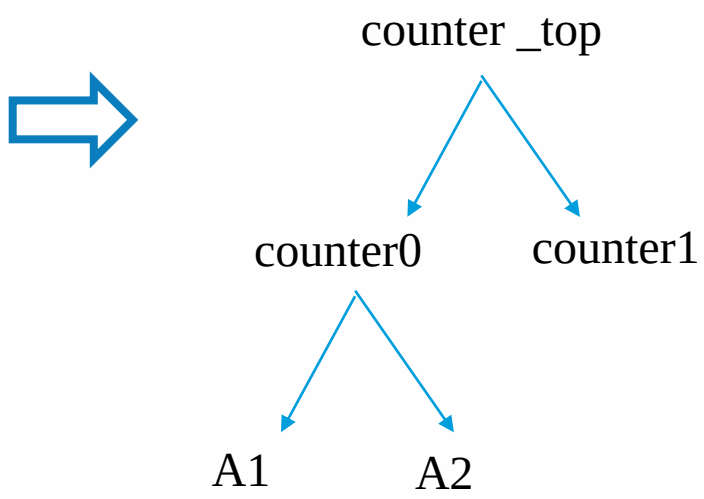
相比C语言，最显著的区别在与HDL语言具备以下**硬件设计**的基本概念：

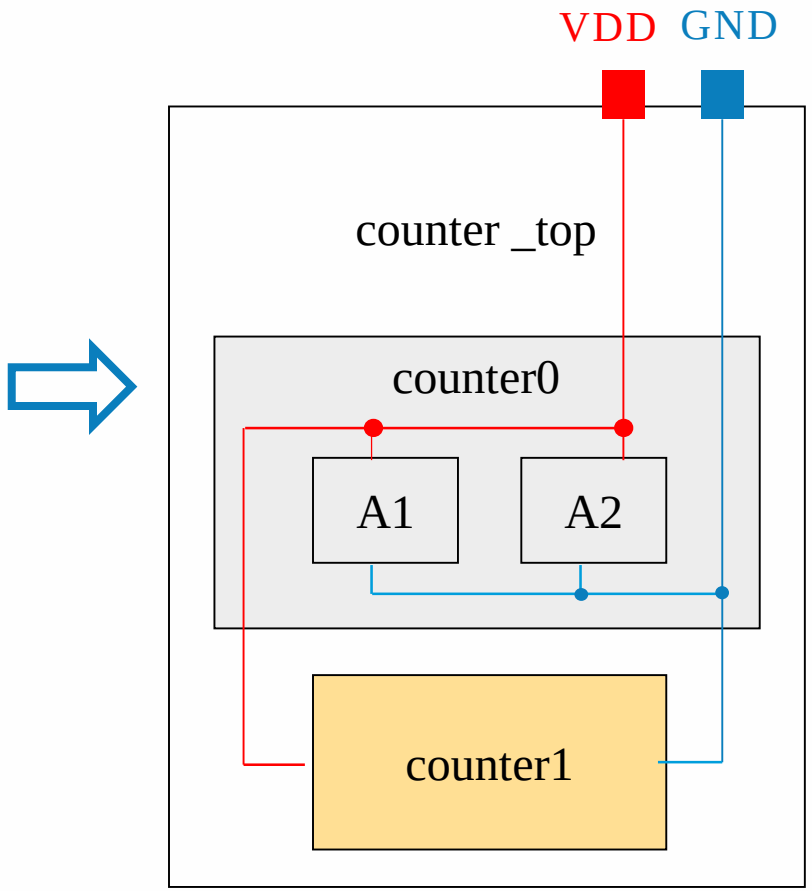
- 互连(connectivity)： wire型变量描述各个模块之间的端口与网线连接关系
- 并发(concurrency): 可以有效地描述并行的硬件系统
- 时间(time): 定义了绝对和相对的时间度量，可综合操作符具有物理延迟

连接特性

```
module counter_top(cin,clk,cout,q);
……
counter    counter0( .cin(cin),  .clock(clk),  .cout(cout0),  .q(q[3:0]));
counter    counter1( .cin(cout0), .clock(clk), .cout(cout),  .q(q[7:4]) );
……
endmodule
```

```
module counter(xxx, xxx, xxx);
……
and A1 (a,b,c);
and A2 (a,b,c);
……
endmodule
```

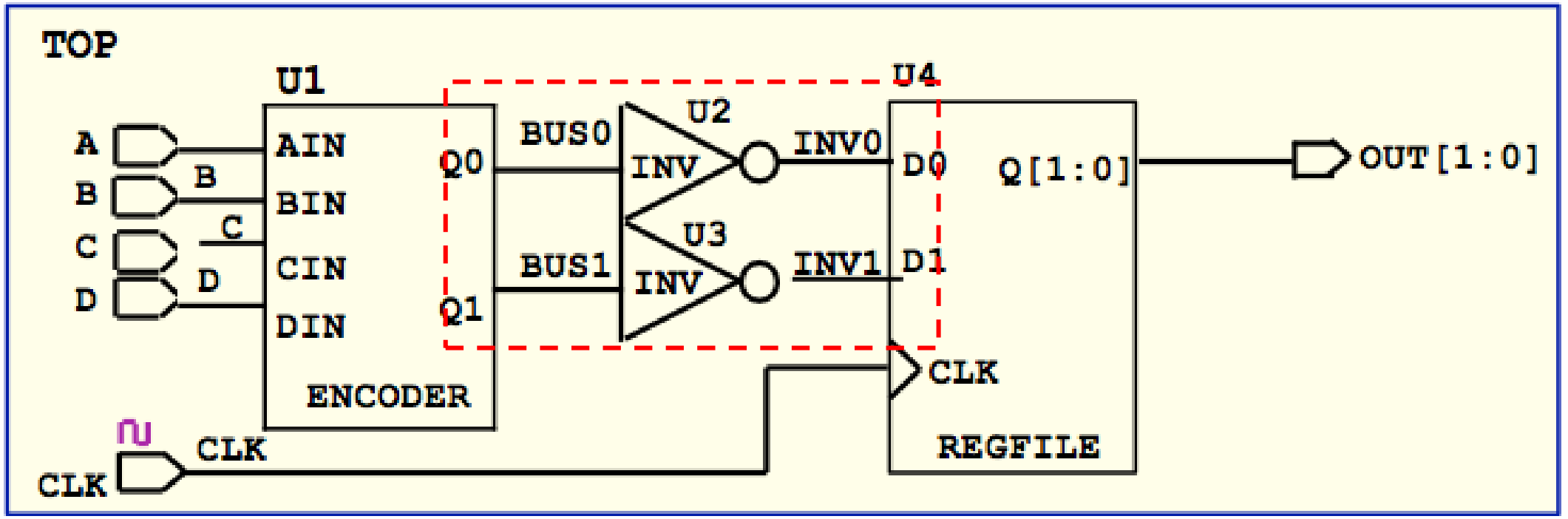




并发执行与顺序执行

与在处理器上运行的软件（如C语言）不同，并行是Verilog 硬件中一个非常重要的概念。

在 Verilog 语言的 module 中，所有描述语句(包括连续赋值语句、行为语句块 always initial 以及模块实例化等)都是并行发生的。而 begin...end 中存在的语句应该是顺序执行的。



并发执行与顺序执行

与在处理器上运行的软件（如C语言）不同，并行是Verilog 硬件中一个非常重要的概念。

在 Verilog 语言的 module 中，所有描述语句(包括连续赋值语句、行为语句块 always initial 以及模块实例化等)都是并行发生的。而 begin...end 中存在的语句应该是顺序执行的。

```
module  TOP ;

always  @  (*)
begin
   if   ()
   if   ()
end

assign  a = b ;

endmodule
```

- always, assign 之间并行
- begin…end 内部串行

参考资料

轻松成为设计高手:VerilogHDL实用精解. EDA先锋工作室. 北京航空航天大学出版社. 2012年.

中国大学MOOC平台：芯动力——硬件加速设计方法

https://www.icourse163.org/course/SWJTU-1207492806

谢谢！

邸志雄，西南交通大学

www.dizhixiong.cn