

数字系统设计

郑海永

上课时间地点：周 2/9 10 节/3308

电子信息科学与技术 2010 级和 2011 级

中国海洋大学 电子工程系

2013 年 4 月



VHDL

- ① 硬件描述语言
- ② **VHDL 基础**
- ③ VHDL 实例
- ④ VHDL 入门
- ⑤ VHDL 基本语句
- ⑥ VHDL 深入

目录

- 1 VHDL 基础
 - 开发流程
 - 快速入门
- 2 Altera Design Software
 - Logic Design
 - Embedded Design
 - DSP Design
- 3 Xilinx All Programmable
 - Introduction
 - Vivado Design Suite
 - ISE Design Suite

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

设计流程

- ① 创建项目
- ② 设计输入
- ③ 器件设置
- ④ 编译
- ⑤ 仿真
- ⑥ 引脚锁定
- ⑦ 下载编程

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

如何学习一种语言？

- 日常交流语言
- 计算机语言
- 字母、词汇、词义、句法、语义 \implies 语句
- 词法、语法、句法 \implies 语句
- 听说读写
- 反复训练

如何学习一种语言？

- 日常交流语言
- 计算机语言

- 字母、词汇、词义、句法、语义 \Rightarrow 语句
- 词法、语法、句法 \Rightarrow 语句

- 听说读写
- 反复训练

如何学习一种语言？

- 日常交流语言
- 计算机语言

- 字母、词汇、词义、句法、语义 \Rightarrow 语句
- 词法、语法、句法 \Rightarrow 语句

- 听说读写
- 反复训练

语言介绍

- sequential language
- concurrent language
- net-list language
- timing specifications
- waveform generation language

强类型、冗余的

- 大小写不敏感
- 注释 “--”

语言介绍

- sequential language
- concurrent language
- net-list language
- timing specifications
- waveform generation language

强类型、冗余的

- 大小写不敏感
- 注释 “--”

语言介绍

- sequential language
- concurrent language
- net-list language
- timing specifications
- waveform generation language

强类型、冗余的

- 大小写不敏感
- 注释 “--”

基本术语

实体

实际器件的硬件抽象。元件

设计单元

描述实体的五种不同类型的结构：

实体声明 描述实体的外部视图。

结构体 （至少一个）实体的内部描述。

配置声明 用来生成实体配置。选出一个结构体与实体绑定。

程序包声明 集成了一系列相关的声明，包括类型声明、子类型声明和子程序声明。

程序包体 （内容）包括程序包声明中声明的子程序的定义。

系统

典型的 VHDL 系统包括分析器和仿真器。

分析器 读入单个文件中的一个或多个设计单元，经过语法验证和静态语义检查之后，编译到一个设计库中。

设计库 是主机环境（支持 VHDL 系统的环境）存储编译后的设计单元。

仿真器 从设计库中读入编译后的描述，对实体进行仿真。

基本术语

实体

实际器件的硬件抽象。元件

设计单元

描述实体的五种不同类型的结构：

实体声明 描述实体的外部视图。

结构体 （至少一个）实体的内部描述。

配置声明 用来生成实体配置。选出一个结构体与实体绑定。

程序包声明 集成了一系列相关的声明，包括类型声明、子类型声明和子程序声明。

程序包体 （内容）包括程序包声明中声明的子程序的定义。

系统

典型的 VHDL 系统包括分析器和仿真器。

分析器 读入单个文件中的一个或多个设计单元，经过语法验证和静态语义检查之后，编译到一个设计库中。

设计库 是主机环境（支持 VHDL 系统的环境）存储编译后的设计单元。

仿真器 从设计库中读入编译后的描述，对实体进行仿真。

基本术语

实体

实际器件的硬件抽象。元件

设计单元

描述实体的五种不同类型的结构：

实体声明 描述实体的外部视图。

结构体 （至少一个）实体的内部描述。

配置声明 用来生成实体配置。选出一个结构体与实体绑定。

程序包声明 集成了一系列相关的声明，包括类型声明、子类型声明和子程序声明。

程序包体 （内容）包括程序包声明中声明的子程序的定义。

系统

典型的 VHDL 系统包括分析器和仿真器。

分析器 读入单个文件中的一个或多个设计单元，经过语法验证和静态语义检查之后，编译到一个设计库中。

设计库是主机环境（支持 VHDL 系统的环境）存储编译后的设计单元。

仿真器 从设计库中读入编译后的描述，对实体进行仿真。

① 实体声明

实体声明 是一个设计实体的外部视图，包括**实体名称**、**类属声明**、**端口声明**和**实体语句**等信息。

```

1 ENTITY full_addr IS
2     PORT (a, b, c_in: IN Bit;
3           sum, c_out: OUT Bit);
4 END full_addr;
```

```

1 ENTITY name IS
2     [ GENERIC (list); ]
3     [ PORT (list); ]
4     [ BEGIN
5       { sentence } ]
6 END [ ENTITY ][ name ];
```

GENERIC({[CONSTANT]{ 名称 }:[IN] 类属类型 [:= 缺省值];});

PORT({[SIGNAL]{ 名称 }:[信号模式] 信号类型 [BUS] [:= 仿真初始值];});

① 实体声明

实体声明 是一个设计实体的外部视图，包括**实体名称**、**类属声明**、**端口声明**和**实体语句**等信息。

```

1 ENTITY full_addr IS
2     PORT (a, b, c_in: IN Bit;
3           sum, c_out: OUT Bit);
4 END full_addr;
```

```

1 ENTITY name IS
2     [ GENERIC (list); ]
3     [ PORT (list); ]
4     [ BEGIN
5       { sentence } ]
6 END [ ENTITY ][ name ];
```

GENERIC({[CONSTANT]{ 名称 }:[IN] 类属类型 [:= 缺省值];});

PORT({[SIGNAL]{ 名称 }:[信号模式] 信号类型 [BUS] [:= 仿真初始值];});

① 实体声明

实体声明 是一个设计实体的外部视图，包括**实体名称**、**类属声明**、**端口声明**和**实体语句**等信息。

```

1 ENTITY full_addr IS
2     PORT (a, b, c_in: IN Bit;
3           sum, c_out: OUT Bit);
4 END full_addr;
```

```

1 ENTITY name IS
2     [ GENERIC (list); ]
3     [ PORT (list); ]
4     [ BEGIN
5       { sentence } ]
6 END [ ENTITY ][ name ];
```

GENERIC({[CONSTANT]{ 名称 }: [IN] 类属类型 [:= 缺省值]});

PORT({[SIGNAL]{ 名称 }: [信号模式] 信号类型 [BUS] [:= 仿真初始值]});

类属

```
GENERIC({[ CONSTANT ]{ 名称 }: [ IN ] 类属类型 [ := 缺省值 ]});
```

名称 类属名称。

- 类属是静态数据，仿真时为常量，关键字 CONSTANT 可省略。
- 类属是设计者向设计实体传递的信息，所以类属流向为流入实体，关键字 IN 可省略。

类属类型 类属的数据类型。

端口

```
PORT({[ SIGNAL ]{ 名称 }:[ 信号模式] 信号类型 [ BUS ][:= 仿真初始值];});
```

名称 端口的信号名称（必须使用合法的标识符）。

信号模式 端口的数据流向，有五种模式：

IN （缺省）流入实体

OUT 流出实体

INOUT 时分复用双向端口

BUFFER 带有反馈的输出端口（限定该端口只能有一个驱动源）

LINKAGE 无特定方向

信号类型 端口的数据类型。

当某个输出端口与另外的实体输出端口直接相连时，应当声明关键字 **BUS**，同时将该端口设计为三态信号。

仿真初始值 **IN** 和 **INOUT** 的端口。

```
1 ENTITY latch IS
2   GENERIC (n: Positive := 8);
3   PORT (eg: IN Bit;
4         d: IN Bit_Vector(n-1 DOWNT0 0);
5         q: OUT Bit_Vector(n-1 DOWNT0 0);
6 END latch;
7 ARCHITECTURE latch_eg OF latch IS
8 BEGIN
9   PROCESS(eg)
10  BEGIN
11    IF eg='1' THEN
12      q <= d;
13    END IF;
14  END PROCESS;
15 END latch_eg;
```

② 结构体

描述风格 建模方式

- 描述设计实体的电路结构，即设计实体中电路元件之间的连接关系。
互连元件 \Leftrightarrow 结构描述
- 描述数据在设计实体中通过寄存器的传输和变换等。并行赋值语句 \Leftrightarrow 数据流 (RTL) 描述
- 描述设计实体中电路模块的行为功能。时序赋值语句 \Leftrightarrow 行为描述
- 上述三种组合而成的混合描述。

```
1 ARCHITECTURE name OF entity_name IS
2   { declaration sentence }
3 BEGIN
4   { concurrent sentence }
5 END [ ARCHITECTURE ][ name ];
```

② 结构体

描述风格 建模方式

- 描述设计实体的电路结构，即设计实体中电路元件之间的连接关系。互连元件 \Leftrightarrow **结构描述** 与传统的 **逻辑图** 描述相对应。
- 描述数据在设计实体中通过寄存器的传输和变换等。并行赋值语句 \Leftrightarrow **数据流 (RTL) 描述** 与传统的 **逻辑表达式 (或布尔方程)** 描述相对应。
- 描述设计实体中电路模块的行为功能。时序赋值语句 \Leftrightarrow **行为描述** 与传统的 **真值表 (或状态图)** 描述相对应。
- 上述三种组合而成的混合描述。

```

1 ARCHITECTURE name OF entity_name IS
2   { declaration sentence }
3 BEGIN
4   { concurrent sentence }
5 END [ ARCHITECTURE ][ name ];

```


多路选择器

```
1 ENTITY inverter IS
2   PORT(a: IN Bit; b: OUT Bit);
3 END inverter;
4 ARCHITECTURE inv_arch OF inverter IS
5 BEGIN
6   b <= NOT a;
7 END inv_arch;

9 ENTITY nand2 IS
10  PORT(a, b: IN Bit; c: OUT Bit);
11 END nand2;
12 ARCHITECTURE nand2_arch OF nand2 IS
13 BEGIN
14   c <= a NAND b;
15 END nand2_arch;

17 ENTITY mux IS
18  PORT(a, b, c: IN Bit; out1: OUT Bit);
19 END mux;
```

结构描述

```

1 ARCHITECTURE mux_arch1 OF mux IS
2   SIGNAL a_n, b_n, c_not: Bit;
3   COMPONENT inverter
4     PORT(a: IN Bit;
5          b: OUT Bit);
6   END COMPONENT
7   COMPONENT nand2
8     PORT(a, b: IN Bit;
9          c: OUT Bit);
10  END COMPONENT;

```

```

1 BEGIN
2   U0: inverter PORT MAP(a => c, b => c_not);
3   U1: nand2 PORT MAP(a => a, b => c_not, c => a_n);
4   U2: nand2 PORT MAP(a => b, b => c, c => b_n);
5   U3: nand2 PORT MAP(a => a_n, b => b_n, c => out1);
6 END mux_arch1;

```

```

1 BEGIN
2   U0: inverter PORT MAP(c, c_not);
3   U1: nand2 PORT MAP(a, c_not, a_n);
4   U2: nand2 PORT MAP(b, c, b_n);
5   U3: nand2 PORT MAP(a_n, b_n, out1);
6 END mux_arch1;

```

结构描述

```

1 ARCHITECTURE mux_arch1 OF mux IS
2   SIGNAL a_n, b_n, c_not: Bit;
3   COMPONENT inverter
4     PORT(a: IN Bit;
5          b: OUT Bit);
6   END COMPONENT
7   COMPONENT nand2
8     PORT(a, b: IN Bit;
9          c: OUT Bit);
10  END COMPONENT;

```

```

1 BEGIN
2   U0: inverter PORT MAP(a => c, b => c_not);
3   U1: nand2 PORT MAP(a => a, b => c_not, c => a_n);
4   U2: nand2 PORT MAP(a => b, b => c, c => b_n);
5   U3: nand2 PORT MAP(a => a_n, b => b_n, c => out1);
6 END mux_arch1;

```

```

1 BEGIN
2   U0: inverter PORT MAP(c, c_not);
3   U1: nand2 PORT MAP(a, c_not, a_n);
4   U2: nand2 PORT MAP(b, c, b_n);
5   U3: nand2 PORT MAP(a_n, b_n, out1);
6 END mux_arch1;

```

数据流描述

```
1 ARCHITECTURE mux_arch2 OF mux IS
2 BEGIN
3   out1 <= (NOT c AND a) OR (c AND b);
4 END mux_arch2;
```

- 通过实体的数据流主要用并行信号赋值语句来表达。
- 实体结构在这种模型风格中没有显式定义，但可以隐含推出。
- 延迟信息（用户定义或默认 delta 延迟）。
- 信号赋值 “<=” 与变量赋值 “:=”

数据流描述

```
1 ARCHITECTURE mux_arch2 OF mux IS  
2 BEGIN  
3   out1 <= (NOT c AND a) OR (c AND b);  
4 END mux_arch2;
```

- 通过实体的数据流主要用并行信号赋值语句来表达。
- 实体结构在这种模型风格中没有显式定义，但可以隐含推出。
- 延迟信息（用户定义或默认 delta 延迟）。
- 信号赋值 “<=” 与变量赋值 “:=”

行为描述

```
1 ARCHITECTURE mux_arch3 OF mux IS
2 BEGIN
3   PROCESS(a, b, c)
4     BEGIN
5       IF c='0' THEN
6         out1 <= a;
7       ELSE
8         out1 <= b;
9       END IF;
10    END PROCESS;
11 END mux_arch3;
```

比较

- 结构体 `mux_arch1` (结构描述) 描述了设计实体 `mux` 的硬件是如何构成的, 即构成设计实体的各个元件之间的连接关系。
- 结构体 `mux_arch2` (数据流描述) 描述了设计实体 `mux` 的数据流向, 即描述了数据如何从输入端口通过何种变换传输到输出端口, 同时也隐含了设计实体的硬件结构。
- 结构体 `mux_arch3` (行为描述) 描述了设计实体 `mux` 的输出端口与输入端口之间的行为关系, 但不包含任何结构信息。

比较

- 结构体 `mux_arch1` (结构描述) 描述了设计实体 `mux` 的硬件是如何构成的, 即构成设计实体的各个元件之间的连接关系。
- 结构体 `mux_arch2` (数据流描述) 描述了设计实体 `mux` 的数据流向, 即描述了数据如何从输入端口通过何种变换传输到输出端口, 同时也隐含了设计实体的硬件结构。
- 结构体 `mux_arch3` (行为描述) 描述了设计实体 `mux` 的输出端口与输入端口之间的行为关系, 但不包含任何结构信息。

比较

- 结构体 `mux_arch1` (结构描述) 描述了设计实体 `mux` 的硬件是如何构成的, 即构成设计实体的各个元件之间的连接关系。
- 结构体 `mux_arch2` (数据流描述) 描述了设计实体 `mux` 的数据流向, 即描述了数据如何从输入端口通过何种变换传输到输出端口, 同时也隐含了设计实体的硬件结构。
- 结构体 `mux_arch3` (行为描述) 描述了设计实体 `mux` 的输出端口与输入端口之间的行为关系, 但不包含任何结构信息。

模型混合风格

```

1 ENTITY full_addr IS
2   PORT(a, b, c_in: IN Bit; sum, c_out: OUT Bit);
3 END full_addr;

5 ARCHITECTURE fa_mixed OF full_addr IS
6   COMPONENT xor2
7     PORT(p1, p2: IN Bit; pz: OUT Bit);
8   END COMPONENT;
9   SIGNAL s1: Bit;
10 BEGIN
11   x1: xor2 PORT MAP(a, b, s1); --Structural description
12   PROCESS(a, b, c_in)        --Behavioral description
13     VARIABLE t1, t2, t3: Bit;
14   BEGIN
15     t1 := a AND b;
16     t2 := b AND c_in;
17     t3 := a AND c_in;
18     c_out <= t1 OR t2 OR t3;
19   END PROCESS;
20   sum <= s1 XOR c_in;        --RTL description
21 END fa_mixed;

```

③ 配置声明

- 配置声明用于从实体的若干个结构体中选择一种结构体，
- 并将代表该结构体的元件绑定到设计库中实体-结构对或配置所代表的实体上。
- VHDL 中没有定义与配置声明相关的行为或仿真语义。
- 仅仅定义用来为实体建立配置的绑定从而可以进行仿真。

④ 程序包声明

- 用来存储一系列共同的声明，例如元件、类型、过程和函数。
- 这些声明用 `use` 语句读入其他设计单元中。

```
1 PACKAGE example_pack IS
2   TYPE summer IS (May,Jun,Jul,Aug,Sep);
3   COMPONENT d_flip_flop
4     PORT(d, ck: IN Bit;
5          q, qbar: OUT Bit);
6   END COMPONENT;
7   CONSTANT pin2pin_delay:time := 125 ns;
8   FUNCTION int2bit_vec(int_value: integer)
9     RETURN bit_vector;
10  END example_pack;
```

```
1 LIBRARY design_lib;
2 USE design_lib.example_pack.all;
3 ENTITY rx IS
4   ...
```

④ 程序包声明

- 用来存储一系列共同的声明，例如元件、类型、过程和函数。
- 这些声明用 use 语句读入其他设计单元中。

```
1 PACKAGE example_pack IS
2   TYPE summer IS (May,Jun,Jul,Aug,Sep);
3   COMPONENT d_flip_flop
4     PORT(d, ck: IN Bit;
5          q, qbar: OUT Bit);
6   END COMPONENT;
7   CONSTANT pin2pin_delay:time := 125 ns;
8   FUNCTION int2bit_vec(int_value: integer)
9     RETURN bit_vector;
10  END example_pack;
```

```
1 LIBRARY design_lib;
2 USE design_lib.example_pack.all;
3 ENTITY rx IS
4   ...
```

④ 程序包声明

- 用来存储一系列共同的声明，例如元件、类型、过程和函数。
- 这些声明用 use 语句读入其他设计单元中。

```
1 PACKAGE example_pack IS
2   TYPE summer IS (May,Jun,Jul,Aug,Sep);
3   COMPONENT d_flip_flop
4     PORT(d, ck: IN Bit;
5          q, qbar: OUT Bit);
6   END COMPONENT;
7   CONSTANT pin2pin_delay:time := 125 ns;
8   FUNCTION int2bit_vec(int_value: integer)
9     RETURN bit_vector;
10  END example_pack;
```

```
1 LIBRARY design_lib;
2 USE design_lib.example_pack.all;
3 ENTITY rx IS
4   ...
```

⑤ 程序包体

- 程序包体用来存储函数和过程定义,
- 这些函数和过程在相应的程序包声明中已经声明过,
- 也可以用来存储出现在程序包声明中的完整的常量声明。
- 程序包声明至多可以有一个程序包体。
- 而实体声明可以有多个结构体。

模型分析

分析器

- ① 读入包括一个或多个设计单元（一个实体声明、结构体、配置声明、程序包声明或程序包体）的文件，
 - ② 并把它们编译成中间态的形式。
 - ③ 编译后的中间态形式在 VHDL 语言中没有定义。
 - ④ 编译过程中，分析器要验证语法，并进行静态语义检查。
 - ⑤ 生成的中间态形式数据存储于特定的设计库（工作库）中。
- VHDL 语言环境中有一个逻辑名称为 STD 的设计库，这个库包括两个程序包：STANDARD 和 TEXTIO。
 - VHDL 语言中有一个 IEEE 标准程序包 STD_LOGIC_1164。

模型分析

分析器

- ① 读入包括一个或多个设计单元（一个实体声明、结构体、配置声明、程序包声明或程序包体）的文件，
 - ② 并把它们编译成中间态的形式。
 - ③ 编译后的中间态形式在 VHDL 语言中没有定义。
 - ④ 编译过程中，分析器要验证语法，并进行静态语义检查。
 - ⑤ 生成的中间态形式数据存储于特定的设计库（工作库）中。
- VHDL 语言环境中有一个逻辑名称为 STD 的设计库，这个库包括两个程序包：STANDARD 和 TEXTIO。
 - VHDL 语言中有一个 IEEE 标准程序包 STD_LOGIC_1164。

仿真

仿真器

- ① 细化过程 (Elaboration phase)。
 - 实体的层次结构被扩展、链接，
 - 库中的元件被绑定到实体，
 - 顶层实体被细化成由许多行为级模型构成的网络。
 - 为信号、变量和常量声明分配存储空间，为变量和常数赋初值等。
- ② 初始化过程 (Initialization phase)。ul>- 计算所有显式声明信号的驱动器和有效值，
- 给隐含声明信号赋值，
- 此时进程也开始运行直到被挂起，仿真时间设为 *0ns*。
- ③ 仿真。
 - 事件驱动仿真。当事件发生，原本要赋给信号的值赋给这个信号。
 - 若信号取值发生变化，且该信号出现在进程敏感信号列表中，进程将被运行直到挂起。
 - 当 VHDL 系统实现或 VHDL 语言定义的最大仿真时间到达时，异常声明发生，仿真停止。

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

Logic Design



- Quartus II Subscription Edition Software
- Quartus II Web Edition Software
- **ModelSim**.ModelSim-Altera Software

Embedded Design

- **Nios II** Nios II Embedded Design Suite (EDS)

DSP Design

- **DSP Builder** DSP Builder

Logic Design



- Quartus II Subscription Edition Software
- Quartus II Web Edition Software
- **ModelSim**.ModelSim-Altera Software

Embedded Design

- **Nios II** Nios II Embedded Design Suite (EDS)

DSP Design

- **DSP Builder** DSP Builder

Logic Design



- Quartus II Subscription Edition Software
- Quartus II Web Edition Software
- **ModelSim**. ModelSim-Altera Software

Embedded Design

- **Nios II** Nios II Embedded Design Suite (EDS)

DSP Design

- **DSP Builder** DSP Builder

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- **Logic Design**
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

Quartus II

Quartus II Subscription Edition Software

- The industry's #1 design software in performance and productivity.
- Microsoft Windows XP/Vista (32 and 64 bits), Red Hat Enterprise Linux 4.0 and 5.0 (32 and 64 bits), SUSE Linux Enterprise 9 and 10 (32 and 64 bits), CentOS 4.0 and 5.0 (32 and 64 bits).

Quartus II Web Edition Software

- A FREE, no license required version of Quartus®II software for your CPLD or medium-density FPGA.

Quartus II

Quartus II Subscription Edition Software

- The industry's #1 design software in performance and productivity.
- Microsoft Windows XP/Vista (32 and 64 bits), Red Hat Enterprise Linux 4.0 and 5.0 (32 and 64 bits), SUSE Linux Enterprise 9 and 10 (32 and 64 bits), CentOS 4.0 and 5.0 (32 and 64 bits).

Quartus II Web Edition Software

- A FREE, no license required version of Quartus®II software for your CPLD or medium-density FPGA.

ModelSim

ModelSim-Altera software supports behavioral and gate-level simulation, including VHDL or Verilog testbenches, for all Altera devices.

ModelSim-Altera Edition

- Recommended for simulating all FPGA designs (Cyclone®, Arria®, and Stratix® series FPGA designs).
- 33 percent faster simulation performance than ModelSim®-Altera® Starter edition.
- No line limitations but need \$945.

ModelSim-Altera Starter Edition

- Support for simulating small FPGA designs.
- 10,000 executable line limitation.

ModelSim-Altera Web Edition Discontinued

ModelSim

ModelSim-Altera software supports behavioral and gate-level simulation, including VHDL or Verilog testbenches, for all Altera devices.

ModelSim-Altera Edition

- Recommended for simulating all FPGA designs (Cyclone®, Arria®, and Stratix® series FPGA designs).
- 33 percent faster simulation performance than ModelSim®-Altera® Starter edition.
- No line limitations but need \$945.

ModelSim-Altera Starter Edition

- Support for simulating small FPGA designs.
- 10,000 executable line limitation.

ModelSim-Altera Web Edition Discontinued

ModelSim

ModelSim-Altera software supports behavioral and gate-level simulation, including VHDL or Verilog testbenches, for all Altera devices.

ModelSim-Altera Edition

- Recommended for simulating all FPGA designs (Cyclone®, Arria®, and Stratix® series FPGA designs).
- 33 percent faster simulation performance than ModelSim®-Altera® Starter edition.
- No line limitations but need \$945.

ModelSim-Altera Starter Edition

- Support for simulating small FPGA designs.
- 10,000 executable line limitation.

ModelSim-Altera Web Edition Discontinued

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

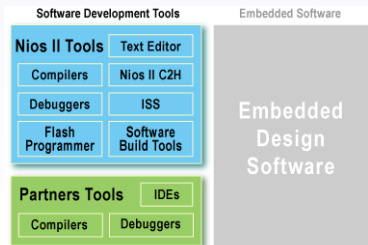
- Logic Design
- **Embedded Design**
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

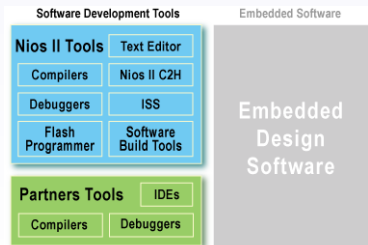
Nios II

- When you design with the Nios®II embedded processor, you have access to a portfolio of robust software development tools and software components available from Altera and our partner ecosystem.
- The Nios II Embedded Design Suite (EDS) is a collection of cutting-edge software tools, utilities, libraries, and drivers to help you bring your design to market in record time.



Nios II

- When you design with the Nios®II embedded processor, you have access to a portfolio of robust software development tools and software components available from Altera and our partner ecosystem.
- **The Nios II Embedded Design Suite (EDS) is a collection of cutting-edge software tools, utilities, libraries, and drivers to help you bring your design to market in record time.**



内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- **DSP Design**

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

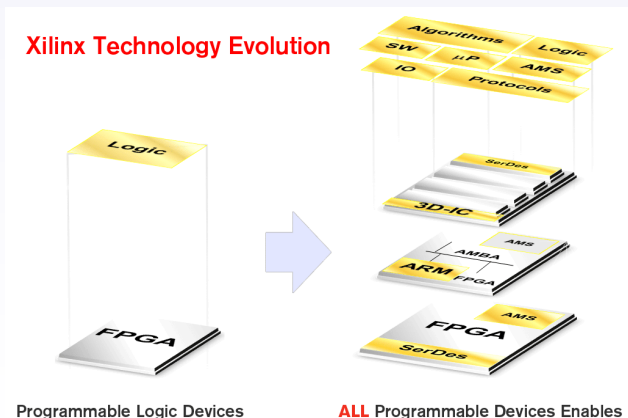
- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- **Introduction**
- Vivado Design Suite
- ISE Design Suite

All Programmable

- Xilinx 是 All Programmable FPGA、SoC 和 3D IC 的全球领先提供商。
- 这些行业领先的器件与新一代设计环境以及 IP 完美地整合在一起，可满足客户对可编程逻辑乃至可编程系统集成的广泛需求。



内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- **Vivado Design Suite**
- ISE Design Suite

Vivado

- 2008 年开始到 2012 年发布的 SoC 增强型设计套件，以 IP 及系统为中心的新一代设计环境。
- Vivado 仅为 Verilog 的时序仿真提供支持。但是 Vivado 可为 Verilog 和 VHDL 以及混合语言提供功能仿真支持。
VHDL 时序仿真是基于 VITAL 的仿真，该标准速度很慢，限制性较大，且已长期未进行更新。

Vivado

- 2008 年开始到 2012 年发布的 SoC 增强型设计套件，以 IP 及系统为中心的新一代设计环境。
- Vivado 仅为 Verilog 的时序仿真提供支持。但是 Vivado 可为 Verilog 和 VHDL 以及混合语言提供功能仿真支持。
VHDL 时序仿真是基于 VITAL 的仿真，该标准速度很慢，限制性较大，且已长期未进行更新。

专注于集成的组件 集成瓶颈

- 集成 C 语言算法和 RTL 级 IP
- 混合 DSP、嵌入式、连接功能、逻辑领域
- 模块和“系统”验证
- 设计和 IP 重用

Vivado

- 2008 年开始到 2012 年发布的 SoC 增强型设计套件，以 IP 及系统为中心的新一代设计环境。
- Vivado 仅为 Verilog 的时序仿真提供支持。但是 Vivado 可为 Verilog 和 VHDL 以及混合语言提供功能仿真支持。
VHDL 时序仿真是基于 VITAL 的仿真，该标准速度很慢，限制性较大，且已长期未进行更新。

专注于实现的组件

实现瓶颈

- 层次化芯片布局规划与分区
- 多领域和多晶片物理优化
- 多变量“设计”和“时序”收敛的冲突
- 设计后期发生的 ECO 及变更引起的连锁反应

内容提要

1 VHDL 基础

- 开发流程
- 快速入门

2 Altera Design Software

- Logic Design
- Embedded Design
- DSP Design

3 Xilinx All Programmable

- Introduction
- Vivado Design Suite
- ISE Design Suite

ISE

- 为各代 Xilinx All Programmable 器件提供业经验证的解决方案。
- ISE Design Suite: Embedded Edition, System Edition, WebPACK Edition.
- ISE Design Suite 嵌入式版本和系统版本现已作为 Vivado Design Suite 的组成部分推出。

ISE

- 为各代 Xilinx All Programmable 器件提供业经验证的解决方案。
- ISE Design Suite: Embedded Edition, System Edition, WebPACK Edition.
- ISE Design Suite 嵌入式版本和系统版本现已作为 Vivado Design Suite 的组成部分推出。

ISE Design Suite：嵌入式版本包括 Xilinx Platform Studio (XPS)、软件开发套件 (SDK)、包括 MicroBlaze™ 软处理器和外设的大型即插即用 IP 库以及完整的 RTL 到比特流设计流程。嵌入式版本可提供实现最佳设计结果所需的基本工具、技术和熟悉的设计流程。具体包括动态降低功耗所需的智能时钟门控、面向多站点设计团队的团队设计支持、面向时序重复性的设计保存，以及用于提高系统灵活性和降低系统尺寸、功耗和成本的部分配置选项。

ISE

- 为各代 Xilinx All Programmable 器件提供业经验证的解决方案。
- ISE Design Suite: Embedded Edition, System Edition, WebPACK Edition.
- ISE Design Suite 嵌入式版本和系统版本现已作为 Vivado Design Suite 的组成部分推出。

ISE Design Suite：系统版本基于嵌入式版本而构建，并添加了 System Generator for DSP™。System Generator for DSP 是业界领先的高级工具，用于设计采用 Xilinx All Programmable 器件的高性能 DSP 系统，可提供 Simulink® 和 MATLAB®（MathWorks 公司）的系统建模和自动代码生成功能。

ISE

- 为各代 Xilinx All Programmable 器件提供业经验证的解决方案。
- ISE Design Suite: Embedded Edition, System Edition, WebPACK Edition.
- ISE Design Suite 嵌入式版本和系统版本现已作为 Vivado Design Suite 的组成部分推出。

ISE WebPACK 提供了全面的、front-to-back 设计流程，让您能够立即免费获取 ISE 特性和功能。