- VHDL简介
  - 硬件与软件实现
  - 用 VHDL 描述硬件
  - 与非门的实现
  - 模拟与非门
  - Vivado 中与非门的仿真
  - 概括

VHDL简介

本节的学习目标是：

- 深入了解从通用处理器到 DSP、FPGA 到 ASIC 的嵌入式系统的原型制作选项；
- 熟悉分别由接口和实现体组成的VHDL组件的基本结构和相关语法；
- 通过一个简单的例子来理解如何使用并发分配来模拟简单的数字门功能；
- 理解软件测试平台的概念，通过模拟测试功能性数字组件；
- 通过运行第一次仿真熟悉用于 FPGA 编程的 Vivado IDE。

硬件与软件实现

软件和硬件实现是指某些所需功能的解决方案，这些功能可以通过对处理器（例如您在本实验室中使用的计算机中的处理器）进行编程来实现，也可以使用自定义硬件来实现。从一个简单的例子中可以最好地理解这一点：对作为文件存储在磁盘上的数字序列进行排序。假设输入引脚上的电压变高，这是将数据从磁盘上特定位置的块读取到系统中、对其进行排序并将其写回的信号。

如果实现是基于软件的，无论您选择哪种编程语言，它都会有一个例程将数字序列从磁盘读入主内存，这是一种数据抽象，允许您操作数据项并根据您选择的重新排序算法，以及将排序后的数据写回磁盘的例程。这里的重点是您计算机中的处理器执行这些步骤，这些步骤已通过您的纯文本程序的编译过程翻译成它可以理解的二进制指令。处理器不仅能够运行这个程序；只要可以用编程语言描述，它就可以执行任何功能。正是因为这种多功能性，计算机中的通用处理器在执行任何一种功能或功能类型时并不是最有效的；它为通用性牺牲了效率。

另一方面，硬件解决方案具有针对给定功能的定制硬件。对于上面的示例，自定义逻辑将监控输入引脚电压，从磁盘读取数据，对其进行排序，然后将其写回。这个逻辑不能做任何其他事情，因为它是专门为这个功能设计的。因此，它可以优化到通常比通用处理器高几个数量级的效率水平。这种特定于应用程序的解决方案导致称为 ASIC 的芯片，它代表特定于应用程序的集成电路。

这是两个极端。有某些处理器，例如数字信号处理器 (DSP)，它们在执行某类功能时非常有效，但具有比通用处理器更窄的指令集。它们在执行专用算法时的效率高于通用处理器，但低于 ASIC。另一方面，它们可以执行比 ASIC 更多的功能，但比通用处理器的功能少。

现场可编程门阵列 (FPGA) 是第四类实现选择，就实现给定功能的效率而言，它介于 DSP 和 ASIC 之间。FPGA 本质上是逻辑门和更复杂的块（例如未配置的存储器、加法器和乘法器）的集合。它们可以配置为通过可编程互连结构实现给定功能。布尔逻辑函数是通过查找表实现的，查找表可以用给定函数的真值表进行编程。FPGA 通常提供非常高效的解决方案，尽管不如 ASIC。另一方面，与 ASIC 不同，它们可以配置为实现多种功能。现代 FPGA 平台有大量可以重复使用的预设计块，称为 IP（知识产权）块。

与给定应用程序的实施选择相关的两个非常重要的非技术问题是开发时间和成本。高级编程语言中的例程是最省时、最经济的实现方式。设计 ASIC 在开发时间和成本方面都非常昂贵，因为必须设计和制造定制芯片。FPGA 是一个很好的折衷方案，因为它们可以现成购买，并通过编程来实现给定的功能，并且被用于越来越多的应用程序中。但是，请务必注意以下几点：

- FPGA 解决方案需要硬件设计；本质上，您必须设计能够执行您的功能的电路。该电路以硬件描述语言 (HDL) 进行描述，可以是 VHDL（在欧洲广泛使用）或 Verilog（在北美使用）。

因此，了解 VHDL 的第一点是它不是一种软件编程语言。您在 VHDL“编程”中所做的是描述实现给定功能的硬件。该描述可以以如下所述的不同方式完成。无论您使用何种类型的描述，软件环境中相当于编译器的功能都会获取您的描述并将其转换为用于配置 FPGA 中资源的网表。电路配置的这些细节大部分在现阶段并不重要，重要的点将在稍后讨论，但中心信息是：

- VHDL 不是软件编程语言，而是描述硬件。
- 硬件基本上是并行工作的（例如，两个门将同时对它们的输入执行它们的功能），因此 VHDL 中内置了并发性。这是如何工作的是首先要了解的事情之一，并且与 VHDL 中的时序模型有关。我们将在接下来的练习中仔细研究这一点。

用 VHDL 描述硬件

VHDL 中的工作硬件模型至少包括两部分，描述硬件组件呈现给世界其他部分的接口的实体描述，以及架构描述组件功能的主体。使用软件类比，这类似于声明函数原型和函数体。架构主体可以在实例化组件时绑定到实体描述，即，使用组件的工作模型，例如在模拟中。单个实体可以有多个架构主体，每个架构主体代表不同的实现。这些通常在功能上是等效的，但可以以不同的样式或使用不同的组件来实现。拥有可以绑定到同一接口的多个实现在不同的场景中非常有用，例如调试不同的版本、使用不同的库组件或针对不同的性能和/或成本规范等。

在硬件建模中，VHDL 支持工程中用于处理复杂任务的一般“分而治之”方法。这实质上意味着一个复杂的任务被分成几个更小的任务，每个任务都更容易解决，并且将每个子任务的解决方案组合起来就可以解决整个问题。在微电子领域，我们使用称为分区和层次结构的两个术语来指代本质上是扁平的（即并行工作的组件）或层次结构（即，组件由子组件组成，而子组件又由它们自己的组件组成）的细分子组件，直到达到技术的基本构建块）。在实践中，一个真正的设计既有分区也有层次细分。

- 通过能够将硬件接口与其内部工作分离的抽象， VHDL 支持分区和层次结构[1]。接口由实体指定，实现由体系结构指定。

VHDL 中有几种不同的编码风格支持模块化设计。我们将推迟到稍后再对模块化进行更完整的讨论，因为我们不想在这个阶段偏离轨道，但现在将其视为让一个组件仅实现一些有限的功能，但做得很好。完整的设计由许多这样的模块组成。再次使用软件类比，模块化是编写可读和可重用程序的关键技术。

这些编码风格可以大致描述为：

1. 结构——根据子组件及其互连描述设计的结构；
2. 数据流或寄存器传输级别 (RTL) - 根据并发语句描述功能；
3. 行为 - 根据高级行为结构（例如 if-then-else 子句）以及并发语句描述功能。

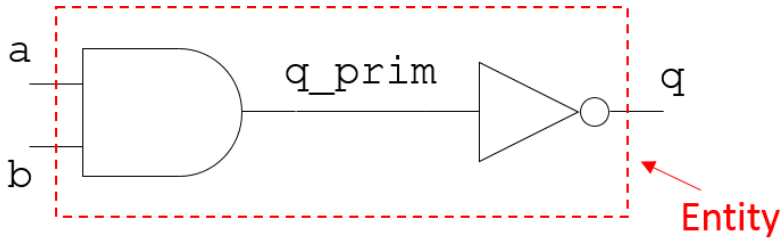
通常，要合成的 VHDL 代码（使用自动化过程转换为硬件）在数字设计社区中通常被称为 RTL，尽管在实践中它是所有三种风格的混合体。

- 描述硬件的VHDL（或 Verilog）代码[2-4]通常称为RTL。

我们将在以下部分中查看几个示例来说明这些样式，这些示例展示了良好的做法。

与非门的实现

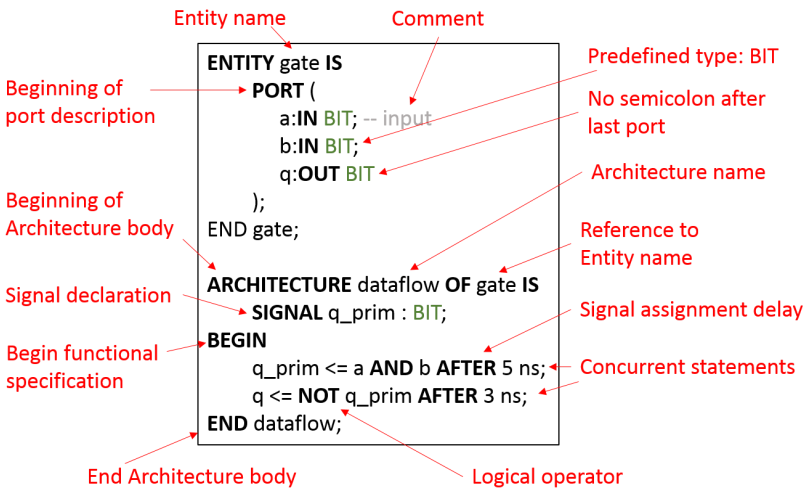
在这些作业中，您将使用 Vivado 软件套件，第一个示例包含一个与非门，该与非门由一个与门与反相器级联而成，如图 1 所示。



使用与门和反相器的 NAND 功能（即 NOT-AND 组合）

与外界的接口由两个输入引脚 a 和 b 以及一个输出引脚 q 定义。q_prim 信号本质上是一条内部连线，将与门的输出连接到反相器的输入。有许多可能的方法来实现此功能，下面给出了该电路的一种可能的 VHDL 模型。

```
ENTITY gate IS
  PORT (
    a:IN BIT; -- input
    b:IN BIT;
    q:OUT BIT
  );
END gate;

ARCHITECTURE dataflow OF gate IS
  SIGNAL q_prim : BIT;
BEGIN
  q_prim <= a AND b AFTER 5 ns;
  q <= NOT q_prim AFTER 3 ns;
END dataflow;
```



实体和体系结构定义的摘要。

描述以保留字entity开头，首先要注意的是 VHDL 不区分大小写；即，entity、ENTITY和EntiTy以及大小写组合的任何其他组合都是可接受的，不会产生语法错误。但是，您应该采用一些有助于提高可读性的明智约定，例如始终对保留字使用大写字母。在此示例中，所有保留字均以大写形式显

示。

- VHDL不区分大小写。

该接口指定了两个 in 类型的输入和一个out类型的输入，这限制了信号流的方向。允许双向信号流的类型inout也是可能的，但此时不推荐，因为它对硬件有影响。

- 观察原理图的输入输出与VHDL模型中PORT声明的对应关系。

关键字architecture开始架构主体，并且始终跟在实体引用之后，在本例中为gate。这意味着实体描述中定义的输入和输出可以在架构主体中读取和写入。

接下来是一个声明空间，其中声明了一个名为q_prim的信号。SIGNAL是 VHDL 中的值持有者，在这个例子中可以被认为是硬件中的电线。

- 注意内部连线和SIGNAL声明之间的对应关系。

类型位是 VHDL 中的预定义（本机）类型，可以取值“0”或“1”。我们在设计逻辑时实际上不会使用这种类型，因为在现实中，连线可以具有“0”或“1”以外的值，即使它们完全由数字驱动，例如高阻抗“Z”，未知“X”，当电线由多个驱动器驱动时，弱/强制高等。我们将使用的类型称为std_logic，它是在一个库中定义的，该库实现了一个名为IEEE.STD_LOGIC.1164的 IEEE 标准。但是我们现在使用它来进行说明，因为它是最简单的类型并且完全适合这个例子。

- 类型位是在 VHDL 中声明的本机类型，可以取值“0”或“1”。在设计要综合的逻辑时，我们实际上不会使用这种类型，因为实际上，连线可以具有“0”或“1”以外的值。我们将使用的类型称为std_logic，它在名为IEEE.STD_LOGIC.1164的库中定义。

在关键字 begin 之后，功能描述包含在两个并发的信号赋值语句中。这与标准的“编程”截然不同；这些语句并不像人们在典型的编程语言中所期望的那样按顺序执行。相反，它们同时执行，因为这正是事件在硬件中展开的方式。

这是如何运作的？

首先，让我们忽略AFTER子句，并假设示例中的两个并发语句是：

```
q_prim <= a AND b;
q <= NOT q_prim;
```

对任何单个并发语句的输入信号的任何更改都会触发该语句中的评估。a和/或 b中的更改将导致对第一个语句进行评估。如果这导致q_prim的值发生变化，则作为第二个语句的评估触发器，因为q_prim是第二个语句的输入。

- 架构体中的信号赋值语句是并发的。
- 因此，语句的顺序无关紧要。

需要注意的一个关键点是所有信号分配都会产生一些延迟；所有硬件组件都需要一些非零时间来更新。即使没有与上述分配相关的显式延迟（我们已删除AFTER子句），信号也具有无限小但量化的固有延迟，称为增量延迟。这可能会在运行模拟和单步执行代码时造成一些混乱。该主题将在后面的第 3.1.2 节中介绍。

现在，我们需要记住所有的信号分配都有一个相关的延迟。

- 信号分配总是有一个相关的延迟。

现在，让我们看一下该功能是如何实现的。NAND 功能是在名为q_prim的内部信号的帮助下完成的，分为两行；每行是一个信号分配。第一行指定信号q_prim应该用(a AND b)的值更新延迟 5 ns 后。该信号延迟旨在模拟真实逻辑门的传播延迟；没有物理门可以零延迟运行。理解这个 5 ns 的延迟是为了模拟具有真实延迟的门非常重要，这将有助于追踪危险。无法合成；即，根据该描述，无法实现物理延迟为 5 ns 的与门。但是，出于仿真目的，我们可以为物理延迟为 5 ns 的与门建模。

在延迟 3 ns 之后，第二行将q_prim的反转值分配给q，它是一个外部引脚。在用 VHDL 描述功能时，重要的是要考虑如何在硬件中实现此功能。在这个例子中，硬件含义非常清楚：第一个信号分配模拟了一个物理延迟为 5 ns 的与门。第二个作业模拟一个延迟为 3 ns 的反相器。

最后，您会说这是结构、数据流或行为风格描述吗？好吧，这显然不是结构描述，因为它没有使用任何子组件。在架构体内部，只有两个并发的赋值语句，因此正如架构体的名称所暗示的那样，这可以被认为是一种数据流风格的描述。您也可以将其视为门级描述，因为每个信号分配都清楚地对应于一个门。

- 为实体、端口、信号、体系结构主体和您可能使用的其他对象或构造（例如进程、变量和常量）命名的清晰直观的名称是生成可读和可重用代码的重要部分。

模拟与非门

为了模拟 NAND 门模型，我们使用测试台来实例化该组件并为其输入提供刺激并将信号连接到其输出，以便观察其瞬态行为。下面给出了我们将用来测试该模型的测试平台。

```
ENTITY test IS END test;
```

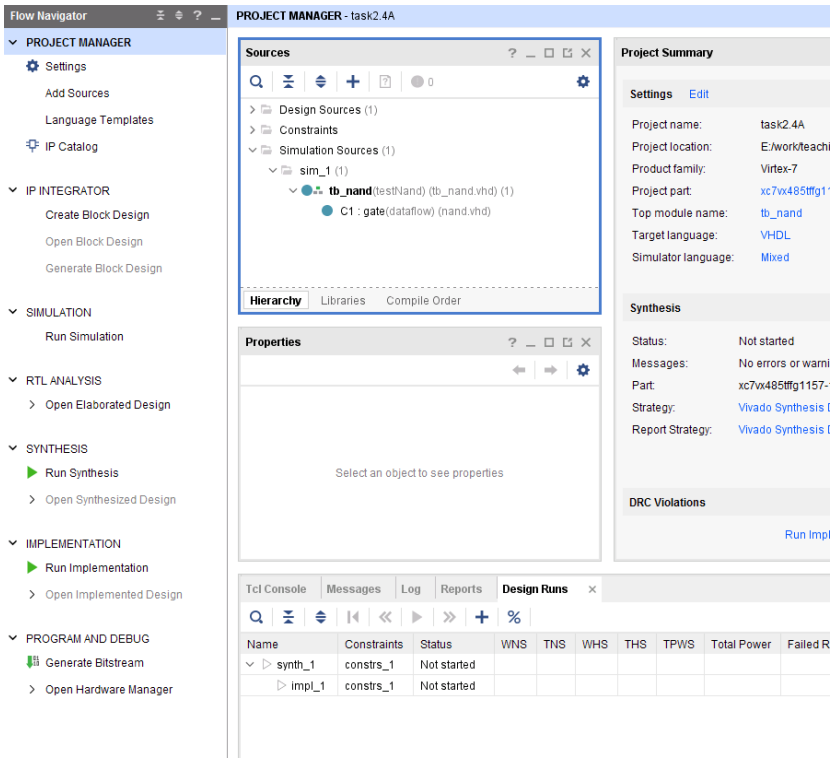
```
ARCHITECTURE testNand OF test IS

  COMPONENT gate
    PORT (
      a:IN BIT;
      b:IN BIT;
      q:OUT BIT
    );
  END COMPONENT;

  SIGNAL a_sig, b_sig, q_sig:BIT;
  SIGNAL c_sig:BIT_VECTOR(1 downto 0);

BEGIN
  C1:gate PORT MAP(a_sig, b_sig, q_sig);

  a_sig <= c_sig(1);
  b_sig <= c_sig(0);
  c_sig <="00",
          "01" AFTER 10 ns,
          "11" AFTER 20 ns,
          "10" AFTER 30 ns,
          "00" AFTER 40 ns;
END testNand;
```

请注意有关此软件测试平台的以下内容：

- 实体没有港口申报。这是因为测试台是独立的。
- 测试台中的信号 a_sig、b_sig 和 q_sig 对应于连接到图 1 电路原理图的输入和输出节点的导线。
- 测试台使用我们之前定义的 NAND 门，首先将其声明为一个组件，然后实例化它并在一行中映射端口信号：C1:gate PORT MAP(a_sig,b_sig,q_sig);
- 组件声明中的组件名称和端口名称、方向和数据类型与与非门的实体声明一致。括号内的信号顺序与组件的实体声明中的端口顺序相匹配。
- 组件的默认映射是将其绑定到具有相同名称的实体。通过给组件一个与现有实体相同的名称，使用默认映射，我们不需要明确指定绑定或配置。正如我们稍后将看到的，没有必要这样做，因为实体/组件和引脚具有相同的名称是非常有限制的。如果我们给出不同的名称，我们只需要明确地将组件映射到现有实体。
- c_sig信号被声明为向量，而所有其他信号都是标量。它被分配了随后通过a_sig和b_sig分配给门组件的a和b输入的位模式。
- 请注意信号分配的顺序并不重要，因为它们是并发的。因此，在将c_sig分配给a_sig和b_sig之前，没有必要将刺激模式分配给c_sig。这种分配顺序在顺序编程语言中没有意义，但在硬件建模语言（如 VHDL）中却非常有意义。

Vivado中与非门的仿真

1. 从 Blackboard 下载作业 1 的源文件包，将其保存在本地文件夹中并解压缩存档。
2. 单击开始 -> 所有程序 -> Xilinx 设计工具 -> Vivado 2022.2 启动 Vivado。
3. 单击快速启动下的创建项目，将打开一个新对话框。在适当的领域：
4. 指定一个文件夹（在您的 O: 驱动器中，以便您可以在任何机器上工作），您将在其中存储程序生成的源文件和其他文件以及项目名称。为避免以后混淆，建议您创建一个名为“Assignment1”的文件夹和包含每个任务名称的子文件夹。如果您遵循此约定，对于此示例，项目名称文件夹将为“Assignment1/task2.4”），项目名称将为 task2.4。
5. 创建一个 RTL 类型的项目。
6. 添加两个源文件。
7. 不要添加任何约束。
8. 通过筛选 Artix-7 系列、封装 cpg236 和速度等级 -1，选择组件 XC7A35TCPG236-1。（这对于这个例子来说不是必需的，因为你不会综合任何东西，但这是你将在整个课程中使用的板。）
9. 创建项目后，您应该会看到类似于图 2.3 的画面。



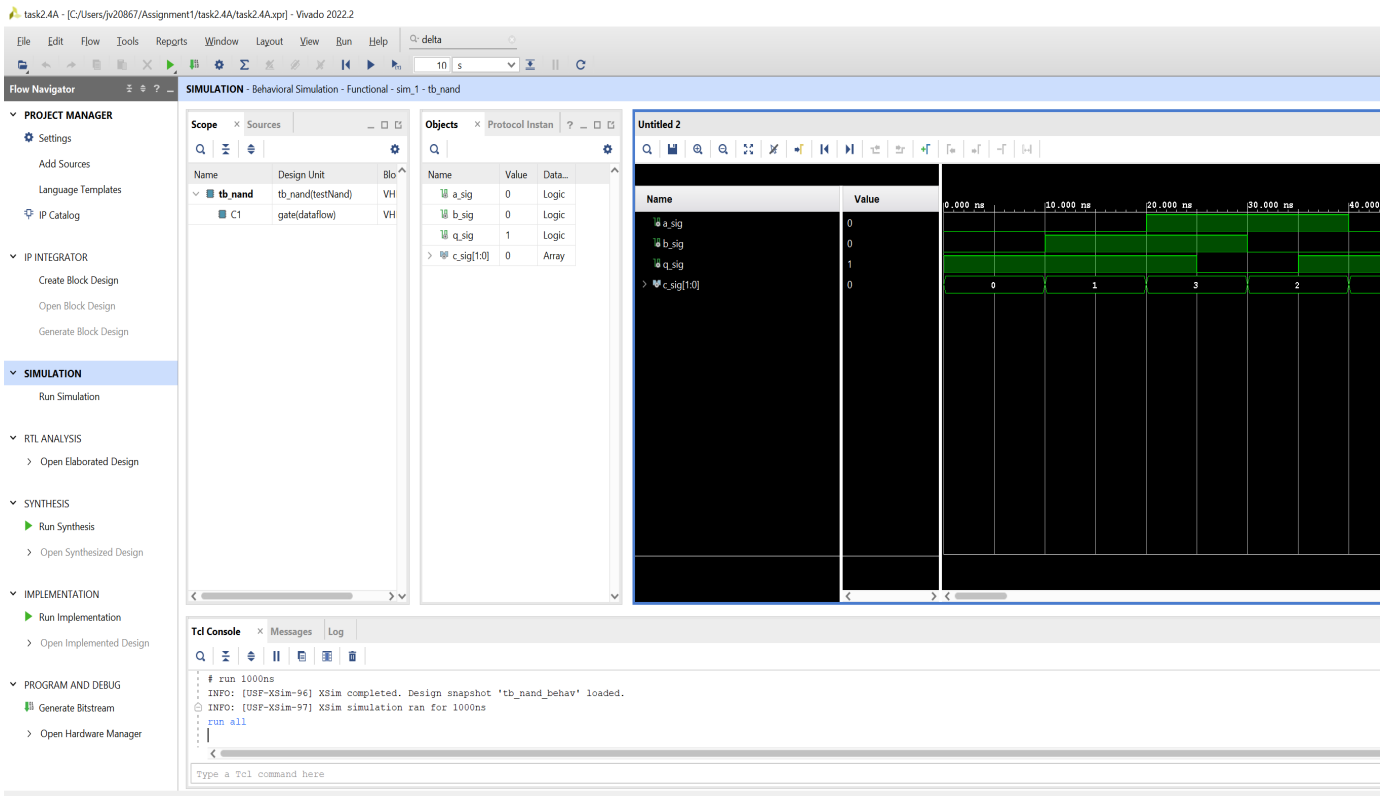
维瓦多集成开发环境。

左侧是 Flow Navigator，其中包含一系列任务，可控制设计和实施过程。它们是项目配置、IP 集成、仿真、RTL 分析、综合、实现和比特流生成。Flow Navigator 中可用的命令和选项取决于设计的状态。在完成所需的设计任务之前，不可用的步骤会显示为灰色。在作业 1 中，您只需要探索项目管理器和模拟。有关 Vivado 集成设计环境 (IDE) 和 Vivado 模拟器的更多详细信息，请参阅参考资料[5-6]。

简而言之，综合是将 RTL 中描述的硬件映射到逻辑的过程。综合后，RTL 分析可用于详细设计。实施是布局和路线的过程；综合网表映射到目标 FPGA 平台上的可用资源。比特流生成生成可用于对 FPGA 进行编程的比特流。

在流导航器右侧的源窗口中，选择层次结构选项卡，如图所示。您添加的两个源文件都应该显示在 sim 文件夹下，以及设计源下。双击源并查看代码。确保它们与上面列出的“nand”和“tb_nand”模块的来源相同。

- 在 Flow Navigator 的模拟下，选择 Run simulation -> Run Behavioral Simulation。默认情况下，将编译源代码并模拟运行 50 ns。菜单将根据上下文而变化，您将看到类似于图 2.4 的视图。



仿真上下文中的 Vivado IDE。

您可能需要放大仿真窗口以查看波形详细信息：右键单击窗口中的任意位置并选择全视图。如果q_prim不可见，在对象窗口中，右键单击q_prim 信号并选择添加到波形窗口。不会有与此信号关联的波形，因为它是在模拟运行后添加的。在模拟菜单中单击重新启动，这是一个看起来类似于倒带

按钮的蓝色图标。之后重新运行仿真，将生成 q_prim的波形。

- 将光标指向任何菜单按钮都会弹出一条解释性说明。
- 波形窗口有几个有用的实用程序，允许光标移动到过渡和特定时间点，以及标记的位置。

- 花一些时间了解波形如何与代码中的描述相关联。特别注意：

  - 为q_prim和q指定的布尔函数的正确性，以及
  - 与其更新相关的延迟。

例如。b_sig在 10 ns 时改变，但输出q_prim不需要改变，因为布尔函数(a_sig AND_b_sig)仍然是'0'。在a_sig在 20 ns 更改后，q_prim应该转到'1'，但只有在延迟 5 ns 之后，当它正式更改为'1'时。q_prim上的此事件导致q发生变化，3 ns 后。

概括

本节介绍了以下几点。

- 基于通用指令集处理器、特定领域处理器、FPGA 和专用集成电路 (ASIC) 的定制应用功能解决方案的能效大约存在一个数量级的差异。
- FPGA 具有可编程结构，包括未配置逻辑门阵列和可编程互连网络。现代 FPGA 拥有大量可重复使用的预设计 IP 块，包括处理器内核、算术、通信和 DSP 块，以及片上总线。
- VHDL 是一种硬件描述语言，用于为硬件建模以进行仿真和综合。VHDL 能够对硬件中固有的并发性进行建模。VHDL描述的基本结构是定义接口的实体和定义实现的架构体。
- 数字逻辑功能可以通过包含信号分配的并发语句来描述。信号分配总是有关联的延迟，无论是否明确指定。
- Xilinx Vivado 用于模拟和编程 Xilinx FPGA 设备。