第5章

FPGA 开发与主要工具使用

本章介绍 FPGA 开发流程，讲解仿真与综合过程，并列举了市面上各种仿真工具与综合工具。此外，以常用的仿真工具 ModelSim 和综合工具 Vivado 为例，对功能仿真与综合过程进行详细介绍。

5.1 开发流程

如图 5-1 所示，FPGA 开发一般流程首先要进行需求分析，明确项目的功能、性能和接口要求；接着进行 RTL 代码设计，结合状态机、流水线等设计思想，根据需求设计硬件电路，并在此阶段进行功能仿真，编写 Testbench 文件以验证 RTL 功能正确性；其次，通过综合工具将 RTL 设计转换为门级电路，再进行布局布线，此阶段可进行时序仿真；最后，将生成的比特流文件烧录进 FPGA 芯片进行板级验证，并使用集成逻辑分析仪（Integrated Logic Analyzer，ILA）获取 FPGA 芯片内部实时运行数据，以便于进行数据保存和离线分析。

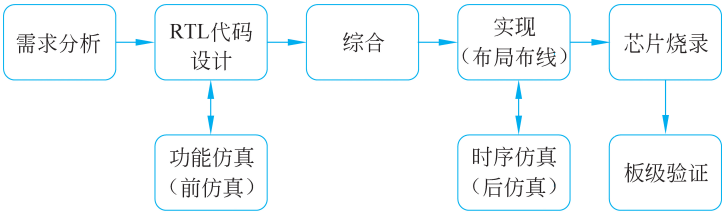

图 5-1 FPGA 开发流程图

5.1.1 需求分析

在 RTL 代码编写之前，需对项目需求进行详细分析，确定项目的功能要求、性能指标和接口定义等，用框图形式确定项目的整体功能框架，包括模块间交互和数据处理流程等。然后，RTL 代码实现应尽可能遵循已确定的设计方案，只在细节处灵活调整。若项目含有算法部分，在该阶段应准备好对应的 MATLAB Model 或 C 语言 Model。

1. 如何进行需求分析

（1）定义项目目标：对项目所要实现的功能和效果进行定义。

（2）确定输入输出：确定项目所需的输入数据和期望的输出结果。

（3）明确性能指标：明确项目对时序、时钟频率和吞吐量等性能指标的要求。

2. 初学者应该注意的问题

（1）确保对项目需求的理解清晰明了，项目组成员均需以设计方案为准进行设计，不可未经过讨论私自修改设计方案。

（2）将性能指标尽可能具体化，有助于后续的设计和开发工作。

5.1.2　RTL 代码设计

在 FPGA 开发中，代码设计涉及如何将项目需求转化为可实现的硬件描述语言（例如 Verilog、VHDL）代码，这一环节主要包括算法设计、系统分析、数据通路分析、控制通路分析和 RTL 代码编写等。关于 RTL 代码撰写的详细内容可参考第 3、6、7 章。代码设计阶段的一般流程如下。

（1）算法设计：在数学层面验证实现目标的可行性。使用软件（如 MATLAB、C 语言或 Python）逐个实现各项功能，并将其封装为独立函数。每个函数对应一种特定功能，通过调用这些函数实现整体设计目标。

（2）系统分析：将软件中的函数映射为功能模块，并分析这些模块间的输入输出信号，若某个模块过于复杂，可将其分为数据通路和控制通路。

（3）数据通路分析：根据软件实现的功能，设计相应的逻辑电路模块，以便将输入信号有效转化为输出信号，并分析数据流从输入至输出所需的控制信号。

（4）控制通路分析：根据输出的控制信号行为，设置电路的状态（如状态机、计数器），并将控制信号连接到数据通路。

（5）RTL 代码编写：分别编写数据通路和控制通路的 RTL 代码，将这两个模块重新封装成一个功能模块。然后实例化各功能模块，最终封装为一个顶层模块。在 RTL 代码设计阶段，良好的结构设计和清晰的代码编写是至关重要的，这将直接影响后续的功能验证和调试效率。

5.1.3　功能仿真

在 FPGA 开发中，功能仿真（前仿真）是验证设计功能正确性的重要步骤，通过编写 Testbench 文件对设计的 RTL 模块进行仿真，以确保 RTL 模块实现的功能满足预定需求。功能仿真的一般流程如下。

（1）准备仿真环境：选择合适的仿真工具（如 ModelSim、VCS 等），对于带 IP 核的仿真，使用综合工具自带仿真功能可简化仿真步骤。

（2）编写测试代码：考虑激励信号的覆盖率，编写仿真测试代码，可参考第 4 章。

（3）运行仿真：通过仿真工具加载设计文件和测试文件，运行仿真并观察仿真波形。

（4）波形分析：分析仿真波形，检查设计在仿真环境下的行为是否符合预期功能，排查可能存在的问题。

（5）调试和优化：根据仿真结果进行调试和优化，修改设计代码以解决功能性问题。

功能仿真是确保设计正确性的关键步骤，通过仿真可以及时发现并解决设计中的问题。在功能仿真阶段，认真分析仿真结果，及时调试和优化设计，将有助于缩短后续上板验证时间，提高开发效率。

5.1.4 综合

将 RTL 代码转换为逻辑门级网表的过程称为综合,对于 Xilinx 开发,该过程由综合工具 Vivado 完成,其输出结果是一个逻辑网表,包含逻辑门、寄存器和查找表等基本元件的信息,并作为下一阶段"实现"的必要输入。此外,在该阶段工程师可通过综合工具查看综合出的门级结构,即 RTL 图,此举有助于进一步确认设计的正确性。

1. 综合阶段的一般流程

(1) 读入设计文件:将设计代码输入综合工具中。

(2) 语法分析:综合工具对设计代码进行语法分析,确保代码符合语言规范。

(3) 逻辑综合:将高层次描述的 RTL 代码转换为底层的逻辑门级电路,包括 AND 门、OR 门、寄存器等。

(4) 优化:综合工具对逻辑电路进行优化,如冗余代码、无效逻辑等,以减小电路规模和提高性能。

(5) 生成 RTL 图:综合完成后会生成 RTL 图,用于表示逻辑门之间的连接关系和数据传输路径。

(6) 逻辑网表生成:基于 RTL 图生成逻辑网表,即表示逻辑电路中各个元件间的连接关系。

2. RTL 图的含义和重要性

(1) RTL 图详细展示了逻辑综合后的电路结构,包括寄存器、数据通路和控制逻辑等元素的布局和连接关系,有助于工程师理解设计的物理结构和时序特性。

(2) 通过分析 RTL 图,工程师可以验证综合结果是否符合预期,排查可能存在的问题,并在必要时对设计代码进行调整。因此,理解和分析 RTL 图对于判断综合后的电路结构是否符合设计要求至关重要。

3. 逻辑网表的内容和作用

网表(Netlist)是一种描述数字电路中各个逻辑元件之间连接关系的数据结构。在数字电路设计中,网表是逻辑综合的输出结果,它记录了设计中使用的逻辑门、时序元件以及它们之间的互连关系。网表通常以文本形式表示,每一行描述一个逻辑元件或连接。网表包含以下主要内容。

(1) 逻辑元件(Logic Cells):网表中会列出设计中使用的逻辑门、触发器、多路器等逻辑元件,每个元件都有一个唯一的标识符和相应的类型(AND 门、OR 门等)。

(2) 互连关系(Interconnections):网表描述了逻辑元件之间的连接关系,即信号从一个元件输出到另一个元件的传输路径,这些连接通常使用标识符或端口名称来表示。

(3) 时序信息(Timing Information):在一些高级综合工具生成的网表中,还可能包含时序信息,如时钟周期、延迟等,用于后续的时序分析和优化。

网表是数字电路设计的重要中间产物,它承载着设计工程师的设计意图和逻辑功能,在 FPGA 设计流程中起着桥梁的作用,连接了逻辑综合和物理实现两个重要环节,对于设计的正确性、性能和时序都具有重要意义。

5.1.5 实现

在 Xilinx 工具中,实现即 Implementation 阶段,主要任务包括将逻辑电路映射为

FPGA 的可编程逻辑器件、时钟资源和 I/O 资源，并生成相应的配置文件及布局布线信息。“实现”阶段通常由 FPGA 厂商提供的综合工具来完成，将逻辑网表映射至 FPGA 的物理资源上，在此之前需进行时序约束，并将 FPGA 内部电路与外界交互的信号映射至实际物理引脚，即引脚约束。实现阶段的一般流程如下。

（1）布局（Placement）：将逻辑元件映射至 FPGA 芯片内的具体位置，以满足设计的时序要求，并尽可能减小信号传输延迟。

（2）布线（Routing）：将逻辑元件之间的连接关系转换为 FPGA 芯片内部的可编程互连资源配置，确保信号能够准确、稳定地传输。

（3）比特流生成：布局布线后，根据具体的 FPGA 芯片配置生成比特流文件，以供板级下载调试。

5.1.6　时序仿真

时序仿真（后仿真）是指在 FPGA 设计中进行的一种仿真验证方法，将布局布线的延时信息反标注到设计网表中来检测有无时序违规。此时一般进行静态时序分析，对关键路径进行优化，得到满足时序要求的最高工作频率。在 FPGA 设计中，特别是对于时序敏感的设计，时序仿真是非常重要的一环。但是需要说明的是，从实际工程经验来看，除特殊情况外，如要求较高工作主频，大部分工程通过前仿真后基本可以满足设计要求。以下是时序仿真阶段的一般流程。

（1）设计时序要求：在 FPGA 设计中，通常会有一些时序要求，比如时钟频率、延迟要求等。这些时序要求决定了设计在运行时需要满足的时钟约束。

（2）时序约束设置：在进行时序仿真之前，需要设置适当的时序约束，以确保仿真能够按照设计的时序要求进行。时序约束通常包括时钟周期、时钟延迟等信息。

（3）验证时序正确性：通过时序仿真，可以验证设计在满足特定时序要求下的正确性。即可以检查设计是否在特定的时钟周期内完成所需操作，以及各个时序路径是否满足时序约束。

（4）调试和优化：时序仿真还可以帮助设计者发现设计中的时序问题。通过分析关键路径，可以进行调试和优化，使设计获得更高的工作频率。

通过时序仿真可以验证设计在满足特定时序要求下的正确性，以帮助设计者发现和解决时序相关问题，确保设计能够在硬件中正常运行。此外，利用得到的时间裕量还可以反推设计可能达到的最高工作频率。

5.1.7　比特流烧录

FPGA 芯片烧录是将已经生成的配置文件加载到目标 FPGA 芯片的过程。这是 FPGA 芯片编程的最后一步，通常使用 JTAG 接口来传输配置文件到 FPGA 芯片的存储器中。

1. 烧录的一般流程

（1）准备配置文件：首先需确保已经生成了适用于目标 FPGA 芯片的配置文件，通常称为比特流文件。

（2）连接下载器：将目标 FPGA 板子通过 JTAG 接口与下载器连接，下载器通常是一

种专门的硬件设备,用于将配置文件加载到 FPGA 芯片中。

① 选择目标设备:在下载工具(如 Vivado)中选择目标 FPGA 芯片型号,以确保配置文件与目标设备兼容。

② 烧录配置文件:通过下载工具将生成的配置文件加载到目标 FPGA 芯片中。在完成烧录后,通常会进行一些验证步骤来确保 FPGA 芯片正确地加载了配置文件,并且可以正常工作。

2. JTAG 接口

(1) JTAG 接口:JTAG 是一种标准化的调试和编程接口,广泛用于个人计算机与目标设备(如 FPGA 芯片)进行通信和配置。

(2) 在烧录过程中,个人计算机通过 JTAG 下载器与 FPGA 芯片连接,将个人计算机中存储的配置文件传输到 FPGA 芯片的存储器中。

3. SRAM 和 Flash/EEPROM 下载区别

FPGA 芯片烧录通常使用 JTAG 接口传输配置文件,可以选择将配置文件下载到 FPGA 芯片内部 SRAM 或外挂的 Flash/EEPROM 存储器中,不同配置方式对应不同的烧录文件格式,选择哪种下载方式取决于具体需求和应用场景。

1) SRAM 下载

(1) 配置文件通过 JTAG 接口加载到 FPGA 芯片的 SRAM 存储器中。

(2) SRAM 下载速度快,但每次上电需重新加载配置文件。

(3) 烧录文件一般为.bit 文件。

2) Flash 下载

(1) 配置文件通过 JTAG 加载到 FPGA 外挂的 Flash/EEPROM 存储器中。

(2) 配置文件可以永久保存在 Flash/EEPROM 中,即断电不丢失。

(3) Flash/EEPROM 下载适合需持久性配置的场景,但配置速度较慢。

(4) 烧录文件为.bin 文件,Vivado 可以在 settings→Bitstream→-bin_file 中勾选,即可生成该文件。

5.1.8 板级验证

FPGA 芯片烧录比特流后,为了抓取实际工作时的内部数据,一般使用集成逻辑分析仪。在 RTL 设计阶段,对想要抓取的信号进行标记,上板后通过集成逻辑分析仪获取对应的数据流以方便验证。

逻辑分析仪(Logic Analyzer)是一种常用于捕获和分析数字信号的工具,用于调试和验证电路设计。FPGA 中常见的在线逻辑分析仪工具包括 Xilinx ISE 中的 ChipScope、Xilinx Vivado 中的 ILA,以及 Altera Quartus Ⅱ中的 SignalTap。通过在 FPGA 设计中插入逻辑分析仪 IP 核来捕获和分析信号,支持实时观察信号波形、触发条件设置等功能,可以捕获复杂的信号波形和时序关系,并能将所捕获信号保存成文件导出,以方便后续分析。关于 ILA 的使用方法参见 8.2.3 节。

5.2　功能仿真与综合介绍

5.2.1　功能仿真概述

FPGA 功能仿真是一种模拟 RTL 模块在 FPGA 芯片中工作行为的手段。通过模拟 FPGA 内部的时序、接口和行为逻辑等，设计者可以在 FPGA 设计过程中进行上板前的功能验证和性能测试。

设计者在完成一个功能模块的代码设计后，需先验证其功能是否满足设计要求，这时就需要对其进行功能仿真测试（又称前仿真，简称前仿），即将 RTL 代码和测试激励（Testbench）导入仿真软件中，再通过运行仿真软件生成对应的波形图，最后通过观测波形图或进行自动化对比即可判断设计是否满足要求。FPGA 功能仿真可以通过硬件描述语言编写仿真测试台，并使用仿真工具（如 ModelSim、Xilinx Simulator）进行测试验证。

以下是 FPGA 功能仿真的一般步骤。

（1）编写仿真测试台：编写一个针对设计的 Testbench 文件，包括激励数据、复位信号、时钟信号生成以及其他必要的仿真控制逻辑。

（2）设置仿真环境：启动仿真工具（如 ModelSim、Xilinx Simulator）创建仿真工程，导入待测文件和 Testbench 文件，并设置仿真环境。

（3）运行仿真：运行仿真，观察对比仿真波形和输出结果，可先肉眼观察波形进行正确性的初步判断，再通过自动化对比仿真进一步对比验证。

（4）调试和分析：根据仿真结果进行调试和分析，检查设计是否符合预期，优化设计以提高性能。

（5）修改设计：如在仿真过程中发现问题，可对设计代码进行修改，然后重新仿真验证。

5.2.2　综合概述

FPGA 厂商的集成设计套件（如 Vivado、Quartus 等）的主要功能是经过一系列的操作步骤（综合、映射、布局布线），最终将设计代码，如 VHDL 和 Verilog 等硬件描述语言，转换成可以配置 FPGA 芯片的比特流文件。关于上述步骤的具体解释可参考 5.1 节的 FPGA 开发流程。

5.3　ModelSim 用法

ModelSim 是一款常用的硬件描述语言功能仿真软件。ModelSim 在 Windows 系统下运行，对于 FPGA 初学者具有下载方便、上手简单和仿真速度快等优点，本节以 ModelSim 2020.4 为例介绍软件的具体使用方法。

5.3.1　常规用法

1. 建立工程

在 ModelSim 工具栏中选择 File→New→Project 命令，如图 5-2 所示。

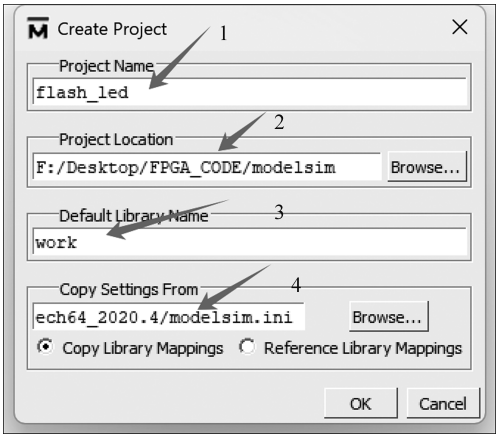

图 5-2 建立工程界面(1)

在图 5-2 中,箭头 1 所指的为建立工程的名称,箭头 2 为工程所保存的地址,工程名和工程地址均不建议包含中文字符和其他特殊符号,箭头 3 为默认仿真库名称,箭头 4 为仿真库地址路径,这里使用默认即可。单击 OK 按钮后,弹出的窗口中会出现 4 种操作方式:Create New File(创建新文件)、Add Existing File(添加已有文件)、Create Simulation(创建仿真)和 Create New Folder(创建新文件夹),如图 5-3 所示。

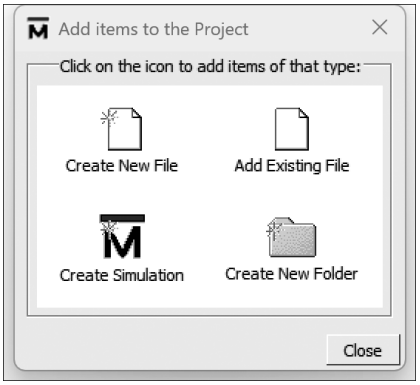

图 5-3 建立工程界面(2)

选择 Create New File(创建新文件),如图 5-4 所示,箭头 1 所指为文件名,根据项目实际情况命名,箭头 2 处默认为 VHDL,应将此处修改为 Verilog,箭头 3 处选择默认选项即可。

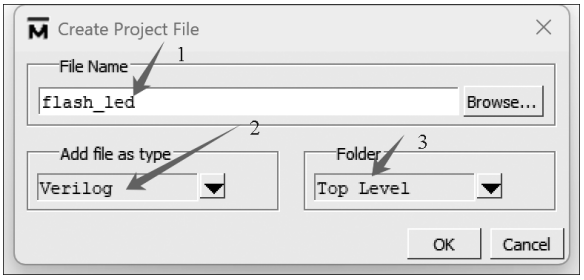

图 5-4 建立工程界面(3)

同理，再创建一个 flash_led_tb.v 文件，由图 5-5 可知，此处创建了两个文件，分别为设计文件 flash_led.v 和 Testbench 仿真文件 flash_led_tb.v，后缀_tb 代表此文件为 Testbench。

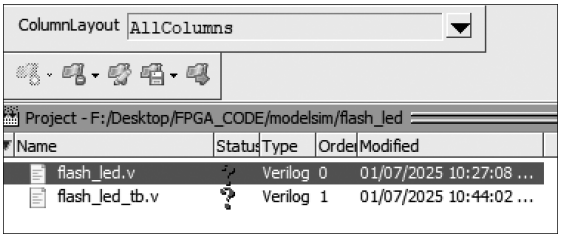

图 5-5　建立工程界面(4)

2. 编写代码

双击文件后，会提示选择何种软件打开此文件，此处选择 VScode 打开，即可在 VScode 界面对代码进行编写和修改，如图 5-6 所示。

flash_led.v ×

F: > Desktop > FPGA_CODE > modelsim > flash_led.v

```
1   // ----------------------------------------------------------------------
2   // ----------------------------------------------------------------------
3   module flash_led #(
4       parameter CNT_MAX=50_000_000    //计数的最大值，让其1s闪烁一次，时钟频率为50Mhz，时钟周期为20ns，计数50M为1s
5   )
6   (
7       input            clk   ,
8       input            rst_n ,
9       output reg [1:0]  led
10      );
11  reg [31:0] cnt;
12  //***************cnt********************
13  always@(posedge clk or negedge rst_n)begin
14    if(!rst_n)
15          cnt<='d0;
16    else if(cnt==CNT_MAX-1'b1)
17          cnt<='d0;
18    else
19          cnt<=cnt+1'b1;
20  end
21  //***************led***********************
22  always@(posedge clk or negedge rst_n)begin
23    if(!rst_n)
24          led<=2'b11;  //led灯低电平点亮，高电平熄灭
25    else if(cnt==CNT_MAX-1'b1)
26          led<=~led;
27    else
28          led<=led;
29  end
30  endmodule
```

图 5-6　编写代码界面(1)

保存代码后，回到 ModelSim 界面，选择 Compile All 命令，如图 5-7 所示。

文件编译后，在 Status 列可能会有三种不同状态："√""△""×"。如图 5-8 所示，Status 栏显示"√"表示编译通过，另外还有两个可能出现的状态："△"代表警告，"×"代表编译不通过。出现编译错误可能有多种原因，可以参考软件下方 Transcript 中的提示信息进行错误排查。

3. 运行仿真

在 View 中打开 Library，找到刚刚创建的工作库，单击 tb_flash_led 命令并选择 Simulate 命令，如图 5-9 所示。

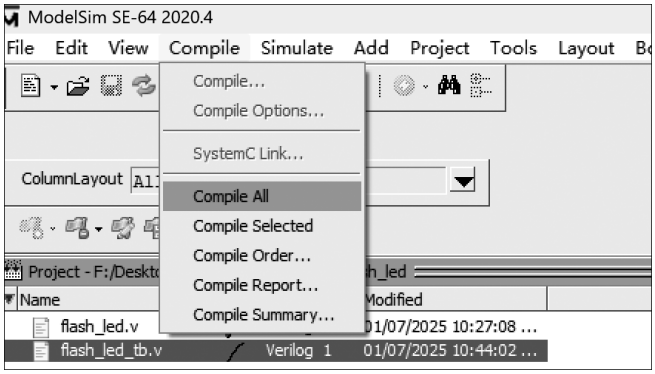

图 5-7 编写代码界面(2)

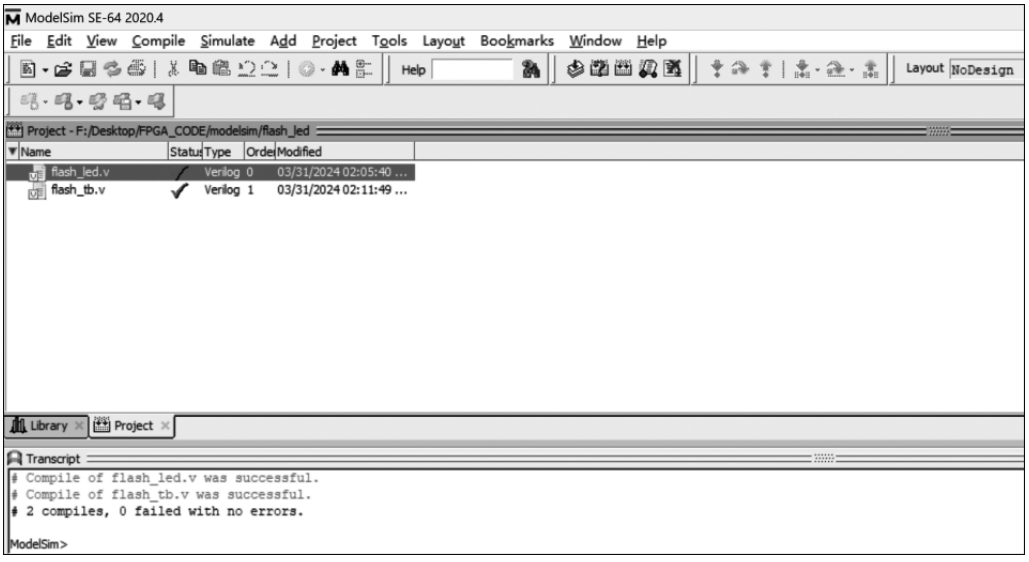

图 5-8 编写代码界面(3)

如图 5-10 所示,在仿真界面中可以设置仿真时长。

如图 5-11 所示,将波形加入仿真图中,单击“运行”按钮可出现仿真波形。

5.3.2 IP 核仿真

在 FPGA 设计中,有时为缩短设计周期需调用 FPGA 提供的 IP 来加速设计,因此在仿真验证阶段需对设计的代码与 IP 核进行联合仿真。此处以 Vivado 为例,介绍一种在 ModelSim 中仿真 Vivado 生成的 IP 核的方法。

1. 生成仿真库:在 Vivado 生成 ModelSim 所需的仿真库

(1) 打开 Vivado 软件,选择 Tools→Settings 命令,在 3rd Party Simulations 界面中,单击右侧的“...”按钮,然后选择已经安装好的 ModelSim 路径。接下来,要设置编译库文件的存放目录,单击右侧的“...”按钮,然后选择一个目录来存放编译库文件。可以在 ModelSim 的安装目录下创建一个新的文件夹,例如 Vivado_lib,并将其指定为编译库文件的存放目录,如图 5-12 所示。需要注意的是,目前还没有生成仿真所需的编译库文件,因此

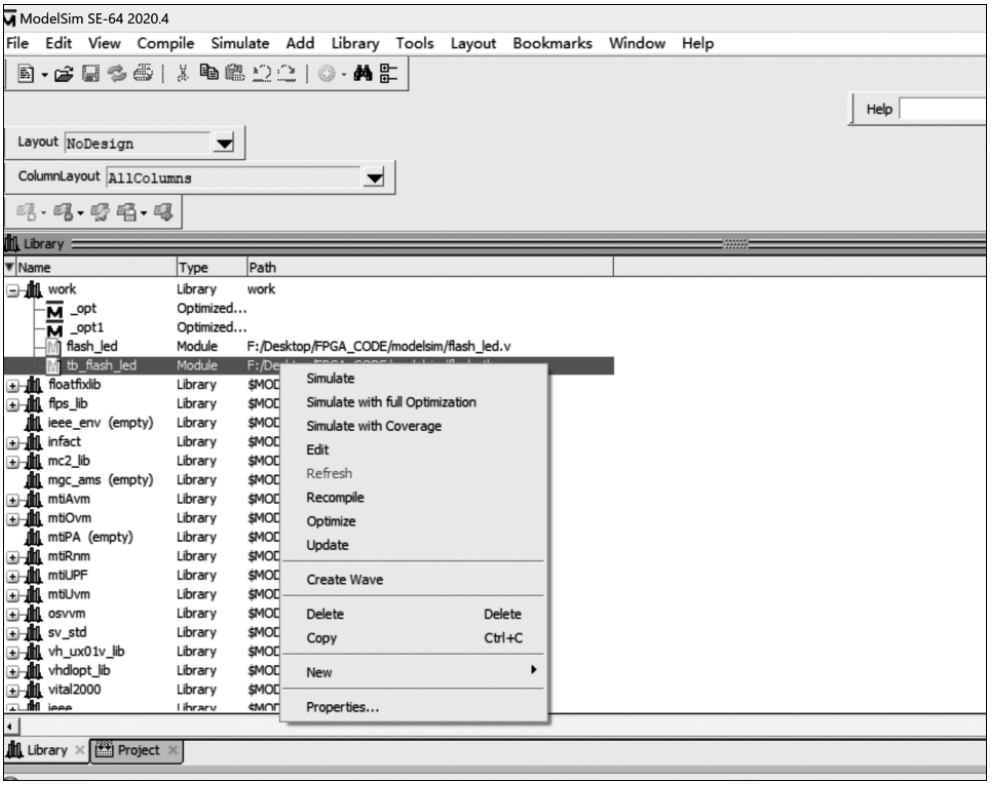

图 5-9　仿真界面(1)

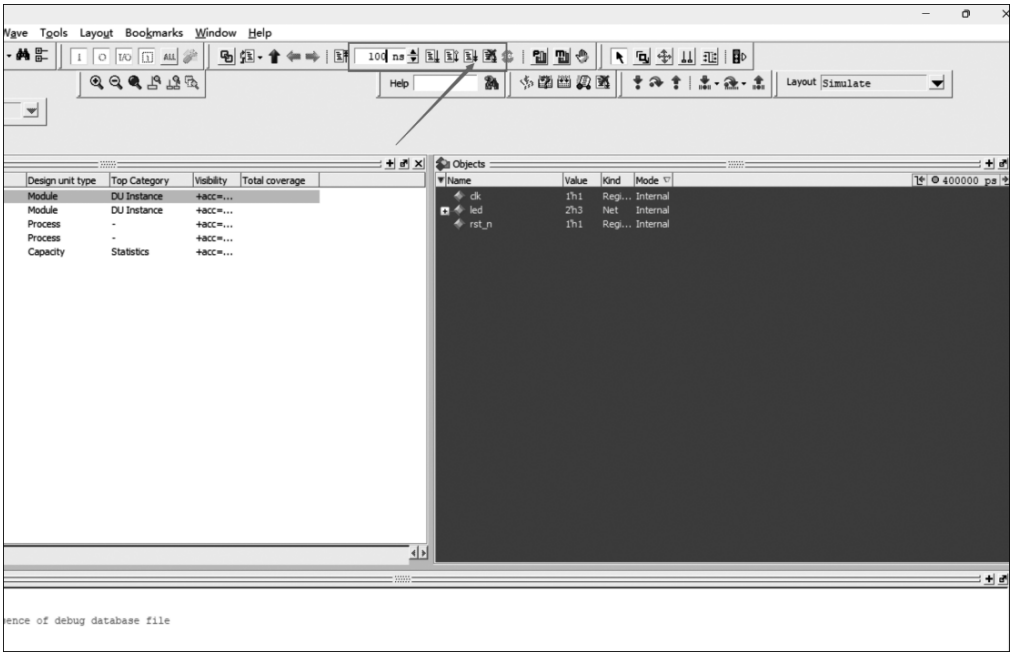

图 5-10　仿真界面(2)