第3章 EDA与FPGA′SoC′DSP设计

3.1 EDA 技术

电子设计自动化(Electronic Design Automation,EDA)是一类软件工具的统称,利用这些工具,工程师可以完成电子系统的设计、仿真、验证和制造等工作。EDA 技术涵盖了数字电路、模拟电路、射频电路、数字信号处理等多个领域,涉及电子设计的方方面面。在当今的电子设计领域,EDA 技术正发挥着举足轻重的作用。随着电子产品的不断复杂化和多样化,传统的手工设计方法已经难以满足现代电子设计的需求。而 EDA 技术的出现,为电子设计带来了革命性的变革。它不仅提高了设计效率,还降低了设计成本,缩短了产品研发周期。

3.1.1 EDA 技术的主要工具

1. 电路设计工具

1) 原理图编辑器

原理图编辑器是进行电路设计的第一个步骤,用原理图编辑器可以将电阻、电容、晶体管、集成电路等各种元器件以图形的方式表达出来,做成一个电路原理图,如在 Cadence Allegro 原理图编辑器中,可以选择元件库中自己需要的元件然后将其放到绘图区,再用导线把各个元件的引脚连接起来,组合形成完整的电路原理图。原理图编辑器一般都会有诸如检查电路连接是否正确、元件属性是否正确等一系列智能化功能。

2) 原理图到网表转换工具

将原理图转换为网表文件是在完成了原理图设计之后需要执行的操作。网表文件是一种以文本形式存在的描述电路连接关系的文件,其中包含电路中的各个元件、各个元件引脚之间的相互连接关系以及电路的拓扑结构等信息。原理图转换为网表的工具可以将画好的原理图经过自动解析后得到对应的网表文件。这个网表文件是后期做仿真或做版图的重要输入文件之一,因此通过原理图生成网表也是原理图的基础工作之一。如在 OrCAD Capture 中,可以用自身的网表生成器将原理图转换为 SPICE 网表或 EDIF 网表等多种类型的网表文件。

2. 仿真工具

1) SPICE 仿真工具

SPICE 是一种用于经典电路仿真的软件,其主要作用就是根据求解电路的节点电压方程来得到电路在不同激励下的响应,如直流工作点、交流频率响应或者瞬态响应等。如 LTspice 即是一款 SPICE 仿真软件,其中包含大量元件模型库,具备非常强大的仿真能力。利用 LTspice 对电路进行精确仿真计算分析,可以检测电路是否达到设计指标。SPICE 仿真工具有时也要求提供电路的网表文件,并根据需要设置输入激励信号的形式、频率范围、仿真时间等参数。

2) 数字电路仿真工具

数字电路仿真用于实现数字电路系统的功能仿真与时序仿真。功能仿真主要检查电路的逻辑功能是否正确,并没有考虑到电路的延迟情况;时序仿真是在完成功能仿真

之后加上实际的延迟信息，在完成功能仿真的基础上对电路进行检验，查看是否有时序错误等问题，包括时序竞争等。如常见的 ModelSim 就是一种针对数字电路系统的仿真软件，可以通过用 Verilog HDL 或 VHDL 编写的设计代码实现编译、仿真及调试等功能，并且可以通过 ModelSim 的图形界面查看电路的输入/输出波形、信号状态、各寄存器的值等。

3. 版图设计工具

1）版图编辑器

版图编辑器是专为集成电路版图而开发的一种用来绘制布局的软件。设计人员会将版图上的每个器件（如晶体管、电阻、电容等）的图形符号放在硅片上，通过对应的金属线完成器件间的互连。版图编辑器往往有十分精确的几何画图功能，可画各种图形，如矩形、圆和多边形等。例如，Cadence Virtuoso 的版图编辑器便能通过选择不同的图层，用一种方式来表示各种不同的图形，如有源区、多晶硅、金属层等。此外，版图编辑器还可实现自动布线、区域填充、设计规则检查（DRC）、减小形迹长度等功能。

2）版图验证工具

版图验证旨在确定版图的设计是否符合工艺设计规则和电路设计的要求。工艺设计规则由半导体制造工厂规定，用来保证器件的制作流程能按设计要求进行，不发生短路、断路或工艺上的问题，根据规则进行几何图形的参数控制，如最小线宽、最小间距、过孔密度等，版图验证工具可通过自动化的方式检测版图中的几何图形是否符合这些规则要求。同样，它也可以做版图和原理图之间的对比验证（Layout versus Schematic，LVS），以保证两者之间的完整一致，如 Calibre 就是一种版图验证工具，它与 EDAS 中的各类 EDA 工具以及半导体工艺厂的各类工艺文件均有接口，用来进行 DRC 和 LVS 的检查。

4. 可编程逻辑器件开发工具

1）FPGA（Field-Programmable Gate Array，现场可编程门阵列）开发工具

FPGA 是一种可编程逻辑器件，在数字电路设计领域应用广泛。FPGA 开发工具有助于设计者从设计输入、综合、布局布线、下载调试等方面一次性进行全部操作，以 Xilinx 公司的 Vivado 为例，Vivado 作为一款 FPGA 设计全流程开发工具，设计者可以通过图形化编辑输入方法或者使用硬件描述语言输入（Verilog HDL/VHDL）的方法来进行操作，综合后将设计代码转化为 FPGA 门级网表，并通过布局布线的方法将门级网表对应到实际的 FPGA 芯片物理资源上，最后将生成的配置文件烧录到 FPGA 芯片中进行硬件调试。除了硬件调试，FPGA 开发工具还有众多其他的高级调试功能，如在线逻辑分析仪、信号探针等。

2）CPLD（Complex Programmable Logic Device，复杂可编程逻辑器件）开发工具

CPLD 也是一种可编程的逻辑器件，与 FPGA 相比，CPLD 的结构相对简单，适合小型的逻辑电路设计。CPLD 开发工具的功能与 FPGA 开发工具类似，如 Altera（现已被 Intel 收购）的 Quartus Ⅱ开发工具。Quartus Ⅱ支持对 CPLD 设计的编辑、编译、仿真和编程等操作。设计人员可以使用原理图或硬件描述语言进行设计输入，然后通过 Quartus Ⅱ的编译器进行综合和布局布线，生成 CPLD 的编程文件。将编程文件通过编程器下载到 CPLD 芯片中后，即可实现设计的逻辑功能。

3.1.2 EDA 技术的应用领域

1. 集成电路设计

作为集成电路设计的一项重要技术，EDA 技术贯穿了从集成电路设计原理的形成阶段到后续的电路设计、版图设计以及制作的整个流程；特别是在设计高性能微处理器 IC 芯片时，需要应用到指令集架构、微架构、逻辑综合、时序分析和功耗分析等工作，应用 EDA 工具可以帮助工程师使用计算机进行设计并不断迭代改进，在较短的时间内提升芯片设计的质量及性能。

2. 印制电路板设计

在印制电路板(PCB)设计中，EDA 技术同样重要，PCB 是组成电子设备的各种电子元器件安装、连接的基础，工程师可以运用 EDA 工具完成 PCB 的设计工作，如可以在 Altium Designer PCB 设计软件中按照原理图以及设计要求放置好各个电子元器件，并借助自动布线或手动布线工具将元器件之间的线路连接起来，另外，EDA 工具可以实现 PCB 信号完整性的检查，还可以开展 PCB 的电磁兼容性(EMC)检查工作，保证 PCB 设计的可靠性与稳定性。还有一些比较高端的 EDA 工具可以完成多层 PCB 设计、高速电路设计、3D PCB 设计和仿真等工作。

3. 系统级设计

随着电子系统的日益庞大复杂，系统级设计的重要性逐渐凸显。在此背景下，EDA 技术提供了虚拟化原型开发生态环境。在此环境中，设计人员可以在真实硬件系统制造前对其所构建的电子系统进行建模、仿真及验证，使用像 System C 这类系统级设计语言及对应的 EDA 工具对嵌入式系统的硬件-软件进行协同设计。设计人员可在一个平台上搭建起该电子系统中的各种硬件(如处理器、存储器、外设等)模型及软件(如操作系统、应用程序等)模型，之后使用仿真工具查看系统的运行状态、性能情况、硬件与软件的交互情况等，便于提早发现问题并解决问题，改善系统的结构、资源分配等问题，尽可能降低系统的设计风险。

3.1.3 EDA 技术的优势

1. 提高设计效率

以前电子设计使用的是人工手绘原理图、人工计算参数、实际试装电路板，耗时较长，出错概率大；而利用 EDA 技术提供的各种自动化设计工具与仿真平台，实现了加快设计进度的目的，如采用原理图编辑器就能快速画出多个复杂电路的原理图；用仿真工具可以反复地、无数次地在计算机上模拟试验，不需要做很多实物电路板，使得设计人员能在很短时间内完成多个方案的尝试和选择，大大提高了设计速度。

2. 降低设计成本

在传统的电子设计过程中，设计人员要投入大量的时间和精力用于频繁的实验电路板制作和硬件调试，其设计成本很高。但在使用了 EDA 技术后，大部分工作在计算机上就可以完成了，不再需要那么多的实验电路板和反复的硬件调试，如，在使用仿真工具

前，便可以提早发现一些硬件电路板上的问题并加以解决，不必等到实验电路板制作完成后再去发现错误和缺陷。使用 EDA 技术还可以提高一次设计的成功率，减少设计错误导致的产品返工、报废等，也就可以降低这部分的生产费用。

3. 增强设计质量与可靠性

EDA 工具可以做全方位的仿真和分析工作，通过对整个电子电路或系统的性能进行检测和诊断，检查电路运行是否符合设计要求；同时还可以通过对各种参数（电压、电流、频率、功耗）进行精细计算和仿真分析，保证电路满足设计的需求，还可以通过使用可靠性分析的方法来评价电路的温度特性、老化特性、电磁干扰特性等，提高电路的可靠性，如采用信号完整性分析方法，在高速电路中可能会产生信号反射或串扰等问题，应提前进行分析，并针对问题对布线进行设计或终端匹配，从而使电路的工作更加稳定可靠。

3.1.4 案例分析：基于 EDA 技术的数字电路设计

为了能更加直接地看到 EDA 技术在数字电路设计上的应用，下面以设计一个简单数字电路的过程来进行说明。

1. 设计目标

设计一个 4 位二进制加法器电路，能够实现两个 4 位二进制数的相加，并输出 5 位的和（包括进位）。

2. 设计步骤

1）设计输入

使用硬件描述语言 Verilog HDL 设计输入加法器电路，第一行定义模块输入/输出端口，包括两个 4 位输入端口 a、b，5 位输出端口 sum 以及 1 位进位输出端口 cout；接下来用 Verilog HDL 语句编写加法器的逻辑功能代码，用位运算符 &、|、^和赋值语句实现两个 4 位二进制数相加以及加法过程中的进位处理。

Verilog 代码如下：

```
module adder_4bit(
input [3:0] a,
input [3:0] b,
output reg [4:0] sum,
output reg cout
);
always @( * ) begin
sum = a + b;
cout = sum[4];
end
endmodule
```

2）仿真验证

在 ModelSim 仿真软件下完成对设计的加法器电路的功能仿真，编写好测试平台（Testbench）代码，给加法器电路输入激励信号，并得到输出结果是否正确的结论。

Verilog 代码如下：

```
module tb_adder_4bit;
  reg [3:0] a, b;
  wire [4:0] sum;
  wire cout;
  // 实例化被测模块
  adder_4bit uut(
    .a(a),
    .b(b),
    .sum(sum),
    .cout(cout)
  );
  initial begin
    // 初始化输入信号
    a = 4'b0000;
    b = 4'b0000;
    // 测试不同的输入组合
    #10 a = 4'b0101; b = 4'b0011; // 5 + 3 = 8
    #10 a = 4'b1010; b = 4'b0101; // 10 + 5 = 15
    #10 a = 4'b1111; b = 4'b0001; // 15 + 1 = 16(进位为 1)
  end
  // 监视输出信号
  initial begin
    $monitor("Time = %0t: a = %b, b = %b, sum = %b, cout = %b", $time, a, b, sum, cout);
  end
endmodule
```

运行仿真后，仿真工具会输出如下结果：

运行结果：

```
Time = 0: a = 0000, b = 0000, sum = 00000, cout = 0
Time = 10: a = 0101, b = 0011, sum = 01000, cout = 0
Time = 20: a = 1010, b = 0101, sum = 01111, cout = 0
Time = 30: a = 1111, b = 0001, sum = 10000, cout = 1
```

从仿真结果可以看出，加法器电路的输出结果与预期相符，说明设计的加法器电路功能正确。

3）综合与布局布线

将设计好的加法器电路代码输入 FPGA 综合工具（Xilinx Vivado 等）中，利用综合工具将 Verilog HDL 代码转换成 FPGA 的门级网表，并进行逻辑优化、映射到 FPGA 的硬件资源上，然后用布局布线工具将门级网表中的逻辑单元放置到 FPGA 芯片对应的物理位置，连线并连接到对应位置上；最终生成 FPGA 的配置文件（.bit），将其下载到 FPGA 开发板中即可实现加法器电路的功能。

3. 案例总结

从这个简单的数字电路设计实例可以看到，采用 EDA 技术可以极大地提高数字电

路设计效率、灵活性及可靠性。用 EDA 工具可以在计算机上很快地将所需的功能电路设计出来，并且不用再做实际的硬件电路板了。EDA 工具的庞大数据库和强大的功能使得整个电路设计工作过程十分清楚，方便实用。由此可见，EDA 技术在当前电子设计方面的作用巨大。

3.1.5 小结

EDA 技术是现代电子设计的重要技术之一，在集成电路设计、印制电路板设计、系统级设计等电子设计的不同领域均有应用。它一方面提高了电子设计的质量和效率、降低了设计成本；另一方面也为电子设计领域的创新和发展提供了强有力的支撑。伴随着电子技术的不断发展及应用的需求不断增加，EDA 技术日趋成熟，也面临着很多新的挑战与机会。

3.2 常用 EDA 软件

EDA 使整个电子设计的流程发生翻天覆地的变化。早些时候的电子电路图都是人工手绘，需要先安排元器件布局，再手绘布线，非常麻烦、费时费力，还有可能出错，若要修改错误，则需要撕毁手稿再重新开始。采用 EDA 技术后，人们可以通过计算机软硬件资源，将传统的电路设计、仿真、分析，直至生产出一个个具体的产品都通过软件自动实现出来，大大提高了电子设计的效率和质量，缩短了产品的研发周期，降低了生产成本。

3.2.1 Altium Designer 介绍

1. 软件简介

Altium Designer 是一款功能非常强大的一体化电子设计软件，主要被应用在 PCB 的设计中，可以完成电路原理图的设计、PCB 布局布线和信号完整性的分析等工作，既可以用于小型电子设备，也可以用于大型复杂的电子系统设计。

2. 界面布局与基本操作

启动 Altium Designer 以后，看到的是 Altium Designer 界面，界面上面是菜单栏，下面是工具栏，左侧是设计管理器，右边是工作区和属性面板。菜单栏包含文件、编辑、视图、工具等操作命令；工具栏内摆放着一些常用的操作命令图标，如新建项目、保存、撤销、重做、导入或者导出文件等；左侧的设计管理器用来管理整个设计项目中的所有文件，如原理图文件、PCB 文件、库文件等，就像一个文件夹，把文件夹的结构都展现了出来；工作区作为设计的中心区域，可以进行电路原理图的绘制、PCB 的布局等工作；属性面板根据当前的选择项或操作方式的不同展示当前需要的属性选项。

例如，新建原理图时，会在设计管理器中得到对应的文件节点，在工作区可以看到一个新打开的空白原理图编辑窗口，可以使用工具栏中的“放置元件”命令将元件库中的元件放置到工作区，并且用“绘制导线”工具将刚才放置好的元件连接起来，做成一个完整的电路原理图，在这个过程中，可以在属性面板上看到每个元件的属性信息，如元件的名称、引脚号、封装形式等，可以根据设计的要求进行相应的设定与修改。

3. 原理图设计功能

Altium Designer 拥有很多的元件库，包含常见的电阻、电容、电感、二极管、三极管、IC 等元件库，还有一些特别的元件库，如连接器、继电器、传感器等元件库。元件库可以根据不同的需求随时调取，并根据需要添加或删除对应的元件库。绘制原理图时只需要简单地将鼠标指针放到所需元件库位置，然后拖到要放的位置就完成了相关的放置工作，再用线条将各个元器件之间的连接画出来即可得到完整的电路图，每一步都自动做了电气规则检查(ERC)，如果出现问题可当即改正。

例如，根据设计要求，在元件库中找出 10kΩ、20kΩ 两种类型的电阻，将它们放到电路设计画板上，然后将它们首尾相接，两端分别接入电源和地端，最后再用拖曳鼠标的方式将二者之间的连线直接通过，点击元器件的引脚即可，不需要理会每个节点之间到底是哪几个端子位置能正确实现彼此相连。连接元器件时，将鼠标指针移到需要连通的端子处，使之高亮显示后，单击并拖动出导线即可，软件可自动检测元器件引脚的电气属性并自动连接导线，不需要考虑引脚是否能连接在一起。在绘制完原理图后，运行 ERC 检查，一旦有错误，软件就会在界面上显示红叉或者报错窗口以提示设计人员怎样去改正。

4. PCB 设计功能

完成了原理图的设计并且通过 ERC 检查之后，就可以将设计方案从原理图转到 PCB 中做 PCB 的设计了。Altium Designer 具有与原理图设计无缝集成的 PCB 设计功能，可以直接将原理图中的元件、元件间的电气连接信息自动转化为 PCB 设计所需要元件的位置信息以及元件之间的布线约束等。

在 PCB 布局时，设计人员可以根据元器件大小、形状、安装方法及电路的电气性能等在 PCB 框中放置元器件，Altium Designer 具有元器件排列、对齐、间距检查等功能的布局工具和辅助布局功能，可以帮助设计人员完成 PCB 元件布局；也可以设定好元件布局的规则(如元件之间最小间距、元件与板框间距等)，保证 PCB 的制造工艺要求及电路的电气性能。

在布线时使用 Altium Designer 可以实现多样的布线方式，提供自动布线、手工布线两种模式，在自动布线方式下，用户可以通过设计布线策略、布线规则来实现快速生成高质量线路的目的；手工布线可按照自身的经验或对电路的理解对特殊线路或重要线路进行局部调整，以更好地满足要求。在布线过程中，软件能直接显示出布线的长度、宽度、间距、走线等具体情况，并且能够随时查看设计规则的检查情况，若出现布线间距偏小、过孔过多等问题，就会马上出现警告信息，并将出错位置提示出来。

例如，首先将元器件按功能模块分别摆放在一起，并且将工作相关的元器件放在一起，尽量使得这些元器件之间的连线距离越短越好，然后再来设置相应的布线规则，如信号线最小间距为 8mil，电源线和地线宽度为 20mil 等；启动自动布线，让软件根据规则自动为各条信号线分配布线路径；自动布线完成后还可以选择手工布线对一些重要信号线或走线密集区域的一些线路做手动优化处理。

5. 仿真功能

Altium Designer 仿真能力强，在设计阶段可以针对电路进行各种模拟、数字仿真分析，提前对电路的性能和功能进行评估，并发现实现过程中可能存在的问题，提前进行调试与改进，减少后期出错和材料消耗的成本和风险。

Altium Designer 拥有大量丰富的仿真模型，包含了许多常用的电子元器件及电路模块，可以满足大部分电路仿真的需求，只要将对应仿真模型连接到原理图中的相应元件上，再输入仿真的参数，如输入信号的类型、幅值和频率，仿真时间等，就可以启动仿真功能，根据仿真结果窗口中所呈现出的电路输出波形、电压电流的变化曲线等内容，判断电路是否符合设计要求，若存在相关问题，就可以对电路不断进行优化、修正、调整并再次启动仿真验证，直到最后获得的电路达到预期的标准为止。

例如，设计一个简单的 RC 滤波电路，在 Altium Designer 中搭好电路原理图，并将电阻、电容等都关联上对应的仿真模型。输入信号设置成 1kHz 的方波，幅度 5V，仿真时间为 0～10ms。开启仿真之后，在仿真结果中可以看到输出波形，并能判断出滤波电路对输入方波的作用是否满足设计要求，如果输出波形纹波较大，则通过调节电路中的电阻或电容的数值再次进行仿真，直到输出波形符合滤波效果为止。

3.2.2 Cadence 介绍

1. 软件简介

Cadence 是电子设计行业全球领先的 EDA 软件，有着非常强的专业性以及非常强大的功能，用于集成电路设计、高速 PCB 设计和系统级设计，在大型复杂电子系统设计方面表现十分突出，许多知名电子企业、知名电子设计公司都在使用这款软件。

2. 界面布局与基本操作

Cadence 软件的界面虽然比较复杂，但总体布局清晰，功能区分明确，从左到右分别为菜单栏、工具栏、设计浏览区、绘图区、属性编辑区。其中菜单栏包含所有操作命令，包括文件、编辑、设计、工具、视图等功能大类，通过菜单命令可以完成非常复杂的设定与设计。工具栏中包括以图标的样式出现的常用命令或工具，单击这些图标就可以完成对应操作。例如，新建文件、保存、复制、粘贴、撤销、重做、绘制导线、放置元件、添加文本等一般操作工具以及绘制原理图、PCB 布局、布图、光绘版图等设计相关工具。设计浏览区位于设计界面上方，用于浏览及展开设计项目中的文件、对象等。设计浏览区中的项目树显示了项目设计层次，可以查看文件间的引用情况及对象的各种属性设置。绘制区是整个工作区域的设计绘制区，打开文件后可以在这里看到文件的具体内容，并且可以在打开的窗口中设计原理图、PCB、版图等，一次可以打开多个窗口进行编辑操作。当前选中的对象或者正在进行的操作都会显示在属性编辑区中，然后根据对应的属性设置选项及参数输入框对所选对象进行相关调整或详细定制。

例如，打开已有的原理图文件，在设计浏览区可查看该原理图的节点、元件、网络等对象信息，在绘图区可看到此原理图的绘制窗口，查看该原理图的内容。用工具栏的“放置元件”从元件库中选择需要的元件放到绘图区，并用“绘制导线”工具连接元件，完成电

路。在此期间，属性编辑区会随之变化显示：当放置元件时，属性编辑区将显示元件的信息（元件名、引脚信息、封装形式）；当画导线时，属性编辑区将显示导线的信息（导线宽度、颜色、电气特性），并可通过这些信息更改对应的数据。

3. 原理图设计功能

Cadence 具有大量元件库，这些元件库都经过了精心的设计与验证，从而使所建元件模型更加可靠、精准，能够满足比较复杂的电路设计要求。对于原理图设计部分而言，Cadence 提供一套强有力的图形绘制工具以及编辑功能，可以满足用户对一些高精度、复杂电气连接关系的表达需求。

除了具有基础的元器件放置、导线连接等功能以外，Cadence 还有一些更为强大的绘图功能：总线绘制功能、网络标号功能、层次化设计功能等。利用这些功能，可以更有效地组织复杂电路以及进行复杂电路的管理，使整个电路更容易被理解。总线绘制功能可以将多个信号线组合在一起画出总线，以便更简单、简洁地表现电路图，并且让电路图更容易看懂；而网络标号功能能够给我们所绘制的电路图的导线加上网络标号，标注出它们之间的电气连接关系，这有助于在不同图纸中进行电气连接与信号传递；层次化设计功能可以将复杂的设计拆分成单独的模块来进行设计和测试，最后再通过层次化的设计方法设计模块之间的接口，并将它们进行结合和连接，对整体电路设计而言，这有助于提高设计的效率与可维护性。

例如，设计一个大型的数字信号处理系统时，可以将系统分成多个模块来考虑，如输入模块、处理模块和输出模块等。在某个模块内部就可以使用基本的元件放置和导线连接等手段，搭建出该模块内的具体电路逻辑；然后再用层次化设计的方法，将不同的模块连接在一起，并根据每个模块之间的输入/输出接口信号来完成整个系统的电路逻辑图设计。对于网线的编号来说，就是将所有模块之间的网线加上相同的编号，使每根网线都与信号进行正确的连接，并且通路也能够准确地将各种信号传输出去。使用总线画图工具可将多条相关联的信号线汇集成一条总线。例如，在传送数据总线时，将 8 条数据信号线合并成一条宽度为 8 位的数据总线。

4. PCB 设计功能

Cadence 在高速 PCB 设计方面具备非常强的优势，并且配有比较先进的布线算法、信号完整性分析工具以及电磁兼容分析等各项特性。

在 PCB 布局期间，利用 Cadence 强大的布局规划功能可以实现 PCB 形状、PCB 大小、PCB 层数、元件封装形式以及电气性能要求等方面的合理布局规划；Cadence 具有多边形区域划分及布局约束设置功能，工作人员能够将不同的功能模块划分到不同区域之中，并对元件间距离、布局方向等方面的条件予以设置，以满足 PCB 制造工艺及电气性能方面的需要；除拥有一定的智能元件布局优化功能外，Cadence 还可依据设计规则以及信号走向情况对元件位置做出自动化调整，保证电路板布局的合理性以及工作效率。

Cadence 布线支持多种布线方式：单层布线、多层布线、微带线布线、带状线布线等，不同的布线方式可满足不同信号类型的布线需求；并且 Cadence 的自动布线功能非常强大，在综合布线规则及约束条件的前提下能够基于不同的布线需求快速创建多样的布线

方案，并且可以针对现有的自动布线结果做出调整及优化；Cadence 还能在布线的过程中实时进行信号完整性和电磁兼容性分析，并给出相应的布线优化建议，如调整布线长度、加屏蔽、优化过孔等，以满足 PCB 信号完整性的需求以及电磁兼容性的要求。

例如，在实现高速信号处理 PCB 时对于高速信号线（时钟信号线、数据总线信号线）要采取专门的布线方式，并进行相关性分析，在使用 Cadence 软件时可以用 Cadence 带状线的布线方式将高速信号线放在 PCB 内部屏蔽层的两个面之间，这样可以避免外界对高速信号线产生的电磁干扰；另外还要对布线长度、走线阻抗进行严格的设置和控制，采用 Cadence 软件对布线的整个过程都可以实时检测信号反射、串扰、衰减等指标是否出现异常问题，一旦检测到存在异常信号就会立刻告警，并给出对应的解决方案提示（如：将布线的拓扑结构改变成树型或网状布线等，或在末端添加匹配电阻等），循环往复不断改进优化，直至实现高速信号的可靠传送。

5. 仿真功能

Cadence 的仿真功能处于行业顶尖水平，它的电路仿真模型十分精准，同时也有着比较不错的仿真分析手段，可以对电路系统进行详细且准确的仿真模拟分析。

Cadence 的仿真模型包中包含了从元器件到 IC 的不同层次的仿真模型，并且是经过了严格验证、调试以后可以真实反映元器件电特性的模型。设计工程师在做仿真分析时可以根据不同的需要选用相应的元器件仿真模型搭建整个电路仿真网络，并通过设置不同的仿真参数（如输入激励信号、仿真时间步长、仿真精度等）来测试仿真网络的具体性能指标。该软件还具有不同的仿真分析方式，如直流分析、交流分析、瞬态分析、噪声分析、失真分析等，可根据不同的应用来查看电路的性能好坏和稳定程度。

例如，可以在 Cadence 中建立电路图并给所有的元器件加上对应的仿真模型，在此基础之上通过交流分析来观察设定频率扫描范围的滤波器，得到滤波器在不同频率下的幅频特性及相频特性曲线，判断出该滤波器具有何种带通或带阻的滤波特性；接着进行瞬态分析，将脉冲信号送入电路，观察滤波器对脉冲信号的响应波形，得到滤波器对于瞬态信号的响应情况。之后根据仿真结果对滤波器进行优化调整，可以通过调整电阻、电容和电感的参数值实现对滤波效果的优化调整。

3.2.3 Altium Designer 与 Cadence 的比较

1. 功能侧重点

Altium Designer 可针对中小规模 PCB 设计提供一体化的解决方案，对于原理图设计、PCB 布局布线及一些基础仿真方面的功能都能够较好地平衡兼顾，并且十分容易上手，适合电子设计初学者以及中小规模的产品设计工程师使用。Cadence 在大型复杂的电子系统的电路板设计、高性能 IC 的设计、高速 PCB 设计等领域都有非常突出的优势，在整个电子设计的功能实现方面更加完善也更加专业，涵盖了信号完整性分析、电磁兼容性分析和先进的封装设计等更为专业的设计内容，因此非常适合拥有较多设计经验并且有着一定技术水平的专业设计团队与大型电子企业使用。

2. 学习曲线

对于新手而言，Altium Designer 的学习曲线不算陡峭，界面简单明了，操作较为简单，设计步骤也是由浅入深、循序渐进。一些最基本的操作如原理图绘制或 PCB 布局布线等都能很快掌握，有利于设计者开始接触真实项目。

相比之下，Cadence 的学习曲线较为陡峭，因其内部涉及大量专业术语及较为复杂的功能设置、操作等。这就需要工作人员在有一定电子设计基本知识的基础上进行学习，并花费较多的时间和精力才能学会如何正确使用此款软件进行设计工作。特别是对于其中的一些高级功能以及比较复杂的项目设计等，还需要在工作中不断地学习与摸索，才能达到熟练的程度来完成设计。

3. 适用行业与企业规模

Altium Designer 适用于大中小型电子设计类企业的消费电子产品、工业控制产品、通信设备及计算机硬件等领域的小规模电子设计项目，尤其是较多地应用在那些比较重视成本，设计周期相对较短的中小企业、创业公司，其对设计软件的易用性、性价比要求很高，Altium Designer 的一体化设计平台以及较低的学习成本可以满足这类用户的使用要求。

Cadence 在航空航天、国防电子、大型通信基站、高性能计算机等领域被广泛应用于高性能电路系统以及大规模复杂电子系统的开发中，其用户主要是大中型的专业化设计单位或大型企业，由于涉及的都是性能和可靠性要求极高的产品，并且有着较高的技术壁垒，在产品投入市场的过程中会耗费大量的精力与金钱，但许多这样的企业为了保证其产品能够始终具有较大的竞争优势，仍旧会将大量资金投入 Cadence 等具备较高专业技术水平的设计软件上。

3.2.4 实际应用案例分析

1. Altium Designer 应用案例——智能手表 PCB 设计

某智能手表研发公司的 PCB 设计是通过 Altium Designer 完成的。设计人员使用 Altium Designer 中存在的大量元件库，在建立智能手表电路原理图前，通过 Altium Designer 完成了处理器模块、传感器模块、显示模块以及电源管理模块等诸多模块的电路设计，接下来只需将这些电路部分放到原理图文件中即可。并且，通过 ERC 检查出存在多个元器件引脚未焊接的问题以及电源网络短路的问题等。这些错误都属于设计错误和布线错误，发生错误后无法直接观察到线路的问题，很有可能造成 PCB 出现空焊等不良现象。

导入了原理图设计信息，按照智能手表外观设计小巧轻薄的要求，在 PCB 布局时使用 Altium Designer 的布局工具对元件进行了合理而紧凑的布局，使元件排列得更为紧密且合理，调整优化元件的排列顺序及元件间的间隔距离，并保持 PCB 的尺寸最小化；PCB 布线采用自动布线再结合手工布线完成，完成高密度的 PCB 布线任务；借助软件 DRC 功能，在整个布线过程中检测布线规则是否正确，以达到布线间距不大于 0.2mm，过孔不多于 2 个或 3 个的目标，提高 PCB 制造的可行性并保证电气性能良好。

Altium Designer 最后生成了所有的 PCB 生产文件（Gerber 文件、钻孔文件、装配图

等),送厂制作生产;之后,经过测试,智能手表的各项性能指标稳定可靠,由此可见,Altium Designer 能够很好地服务于中小规模电子产品 PCB 的设计任务。

2. Cadence 应用案例——高速服务器主板设计

一家中国知名的服务器制造商使用 Cadence 软件设计其全新高速服务器主板,涉及大量高速信号处理、复杂电源分配电路和高密度元件布线等问题,Cadence 在这些方面的能力得到了充分展现。

设计团队在原理图设计阶段,采用 Cadence 丰富的元件库及层次化设计功能,将服务器主板设计成 CPU 模块、内存模块、I/O 模块、高速背板接口模块等多种功能模块,用统一的电气接口和信号连接规范对不同的模块建立电气连接,利用 Cadence 平台提供的仿真功能对各模块电路的设计性能进行信号完整性仿真、电源完整性仿真、电磁兼容性仿真,针对存在的各种问题及时优化了电路设计参数,使主板具有更好的电气性能以及更可靠的工作性能。

在 PCB 的布局布线阶段,利用了 Cadence 的高速布线算法和信号完整性分析工具,在这样高密度的元件布局及复杂的高速信号布线要求下,通过使用 Cadence 布局规划工具和设置布局约束的功能来合理地设置每个功能模块的位置及元件的布置方向,让线路信号走最短的距离并减少延时;利用软件自动布线功能以及配合人工优化的方法完成了高速信号的高质量布线路径,并结合信号完整性分析结果对其中重要的信号线进行适当调整,如调整其布线长度,增加终端匹配等措施改善信号品质,降低了信号损失与干扰等问题,从而实现高速信号的优质传输。

经过几个月的反复设计与调试,成功完成了服务器主板的 PCB 设计,并经过了功能测试、性能测试、EMC 测试等一系列严格的测试与验证,高速服务器主板取得了批量上市的机会,彰显 Cadence 软件的强大功能优势,成为其高端复杂电子系统的成功案例。

3.2.5 小结

随着技术革新的不断出现,电子技术发展到现在的信息化高度。伴随着信息化建设的推进和加速,EDA 软件也在不断完善,从技术层面来讲将向着智能型、自动型、集成型、人机协同型方向发展;同时大量融入人工智能、机器学习、云计算、大数据等新技术,为广大电子设计人员提供更完善的解决方案与更有用的设计工具。

掌握 EDA 软件是从事电子相关工作需要掌握的基本技术之一,在学习 EDA 软件时,应根据自己的兴趣方向及岗位规划来选择适合自己的 EDA 软件,如初学者可以先从 Altium Designer 学起,逐步学习掌握原理图绘制、PCB 布局布线、仿真等基础功能。想做高端电子设计的学生,在掌握了基础知识以后可以学习 Cadence 等专业软件,完成较为复杂的项目;此外,还应该注意在学习过程中加强理论知识以及实践能力方面的学习;结合实例或参与各类电子设计竞赛,将自己所学的 EDA 知识运用于实际问题中,提升自身的设计能力和解决实际问题的能力;并且关注 EDA 技术的发展趋势及行业动向,多参加相关培训、研讨会或线上课程的学习,及时了解新的知识,使自己的知识得以更新,顺应发展的潮流。

3.3 FPGA 开发基础

3.3.1 FPGA 概述

FPGA(Field Programmable Gate Array)即现场可编程的门阵列,不妨将其比喻为一个可编程的“电子积木板”,在该板上集成了众多可编程的逻辑单元,具备一定的可配置的布线资源和大量的输入/输出(I/O)接口,许多小小的“工人”负责实施各种数字逻辑计算以及信号处理等工作,“道路”就是供“工人”来往活动的空间,“窗口”就是这个电路的接口。这块“积木板”通过“窗口”同外界相连,它既可以接收外界的输入信号,也可以向外界输出其处理完的信号。类似地,就像小时候玩乐高,把一块积木插进另一个地方就变成了不同形状的建筑或车船等;FPGA 就像一块随时随地都可以定义其功能的超级积木,可被设计人员用来实现简单的逻辑门电路的功能,也可用来完成比较复杂的数字信号处理的算法。

在传统电子电路的设计中,设计人员需要手工绘制电路图并放到面包板或其他实验台上做实物装配调试,在调试中出现错误就会产生很大的损失;在面对复杂的数字系统时,错误难以避免且难以纠正。然而当采用 EDA 技术后,只需使用 EDA 工具用 HDL(硬件描述语言 Verilog HDL、VHDL)来实现 FPGA 的逻辑功能,并通过软件工具进行仿真就可提前发现可能存在的问题,之后经过综合、优化及布局布线,最后下载到 FPGA 芯片中便可实现硬件电路的功能,从而使设计效率更高、设计质量更好。由此可见,EDA 工具在 FPGA 开发中的关键作用主要包括以下几点。

1. 设计输入

利用 EDA 工具提供的文本编辑器与图形化设计环境,可以将设计思路“输入”计算机中。常用的 Quartus Prime 软件,可直接编辑 Verilog HDL 的文本编辑器以及能够绘制由逻辑门、触发器等构成的电路图的图形化设计界面。Quartus Prime 软件中两者结合使用可以实现编写 Verilog HDL 设计代码与绘制电路原理图的操作。

2. 仿真验证

在设计过程中,在还没有真正将 FPGA 芯片变成硬件来实现功能之前,看不到 FPGA 真正的运行效果,利用 EDA 软件的仿真功能就可以模拟 FPGA 的不同输入激励,观察不同输入条件下产生的输出结果,检测是否存在逻辑错误或时序问题等。例如,使用 ModelSim 可以对设计代码进行功能仿真和时序仿真,功能仿真可以验证设计逻辑功能是否正确;而时序仿真可以查看信号的延时、建立时间和保持时间等是否达到要求,保证设计的正确性。

3. 综合优化

当完成了设计输入以及仿真的相关工作之后,接下来需要将设计代码转变为 FPGA 芯片能真正执行的电路结构,这个过程就是综合优化,EDA 工具的综合器会在一定的前提下将硬件描述语言代码转化为相应的网表,并能在这一过程中做一些优化工作,如消

去冗余的逻辑，或者调整电路结构去满足一定的时序约束等。而在 Synplify 综合工具中，可以采用设定时序约束条件或选择面积优化选项等方式来让综合器生成符合设计需要的网表。

4. 布局布线与下载编程

完成全面优化后，EDA 工具将继续完成布局布线的过程，根据网表文件，把电路逻辑单元放在物理上真正存在且能够支持该逻辑运算的 FPGA 芯片的相应位置，并规划好各个逻辑单元间信号的连通方式；决定整个设计在 FPGA 芯片上的物理实现方式。做完布局布线之后，会输出下载文件，通过编程器将这个下载文件灌入 FPGA 芯片，使得它开始按照我们的设计运行。例如，在 Vivado 软件下布局布线完成后，利用编程器将产生的比特流文件下载到 Xilinx 系列的 FPGA 开发板中，可以看到真实的硬件运行效果。

3.3.2 FPGA 开发基本流程

1. 需求分析

每个设计项目都需要以需求作为支撑点，我们在开展 FPGA 项目前需要与客户、项目组成员等相关人员沟通，确定好该项目的功能需求、性能指标和一些限制条件，其中功能需求是指该做什么事情，如一个简单的数码管显示控制器或一个复杂的图像处理系统；性能指标包括时钟频率有多高、数据传输有多快、计算结果精度有多高等；还有就是有大小，如选什么型号的 FPGA、成本多少、需要多少时间来完成等，弄清楚这几个问题后，就可以有的放矢地开展后续的设计。

例如，为了设计一种基于 FPGA 的智能交通灯控制系统，在该阶段应当完成对交通灯相位切换逻辑（如东西方向为绿灯时南北方向必须为红灯，并保持规定的时间）、是否需要根据实时的车流量情况进行绿灯时间长短的自动调节（包括怎样采集、处理传感器信号并最终控制相应的算法）、系统对于稳定性方面的要求（如是否在系统断电以后恢复供电时能自动工作等）等问题的分析与确定工作。这些问题直接决定了后面如何展开设计工作。

2. 设计输入

根据需求分析结果，有两种常见的设计输入方式：一种是用硬件描述语言（HDL）编写代码，如 Verilog HDL、VHDL 等，这两种都是 FPGA 设计常用的硬件描述语言，直接用代码去描述电路的逻辑功能、结构和行为；另一种是用 EDA 工具提供给用户的图形化设计界面画电路图，这类比较适合简单的电路设计，如果是比较复杂的电路设计，用这种方式可能会比较麻烦也容易出错。

以 Verilog HDL 为例，4 位二进制计数器的设计输入可以写成如下形式。

Verilog 代码：

```
module counter4bit( input clk, input rst, output reg [3:0] q );
always @(posedge clk or posedge rst)
begin
```

```
    if(rst) q <= 4'b0000;
    else q <= q + 1;
    end
    endmodule
```

这段代码定义了 counter4bit 模块，并给出了用于接收的时钟信号 clk、复位信号 rst 及输出信号 q。在 always 块内，判断时钟信号上升沿或复位信号上升沿，在复位信号 rst 为高电平时，将输出信号清 0(4 位二进制数 0000)；否则，每来一个时钟上升沿输出信号 q 就加 1，这样实现的是一个最简单的 4 位二进制计数器的功能。

3. 仿真验证

完成设计输入后需要接着进行仿真验证，在 EDA 工具的帮助下，可以完成功能仿真和时序仿真。

功能仿真的重点是验证电路或芯片的功能是否如设计图中的逻辑功能所定义，不需要关心具体的速度与实际的硬件时序关系等，以上面的 4 位二进制计数器为例，使用 EDA 工具编写好测试激励程序，在计数器中施加时钟信号、复位信号，看输出信号 q 的变化是否为 0000→0001→0010→…→1111→0000(循环计数)，如结果与理论相同则证明该电路的功能正确无误。

时序仿真就是在功能仿真中加入 FPGA 实际的时序参数(如信号传输延时、建立时间、保持时间等)，以检验所设计的电路在真实的硬件环境下能否稳定工作。对于高速数字信号处理系统的 FPGA 设计来说，可以通过时序仿真判断信号在传输过程中是否有因为时序原因导致的问题发生，如是否会发生数据采样错误或者信号上出现毛刺等情况，以此发现可能存在的时序隐患。

4. 综合

综合是在仿真的 HDL 代码的基础上，形成 FPGA 可以执行的电路结构。综合工具将代码的逻辑功能映射到器件的逻辑单元和连线等构成的网表文件中，并对代码设计进行优化，在满足性能指标和资源占用的前提下，使实现的设计更加接近预期的效果。

例如，综合工具可以自动排除一些代码中的冗余逻辑，并且会通过时序约束来修改电路结构，使得关键路径上的信号传递延时满足要求，在综合时可以给综合工具添加一些相关约束条件，如时钟频率、输入/输出延时等。

5. 布局布线与下载编程

综合完成后进入布局布线阶段，即 EDA 工具根据综合生成的网表文件将逻辑门布置到 FPGA 器件的正确物理位置并找出逻辑门之间连线信号通路的过程，是直接关系到 FPGA 器件中电路性能(如信号延时、功耗等)好坏的一个重要过程。

完成布局布线以后，便会生成可以下载的二进制文件，一般是 JIC 或 SOF 等文件类型，使用编程器将该文件下载到 FPGA 开发板的目标芯片中，就完成了硬件布局布线设计，实现了真实的电路效果。例如，用 Altera 公司生产的 cyclone 系列 FPGA 开发板，在做完布局布线之后，利用配套的编程器软件，选定好下载配置项，将生成的 SOF 文件下

载到开发板上的 FPGA 芯片中，之后开发板上的 LED 灯、数码管、按键等就会按照设计好的逻辑产生一定的动作，如 LED 灯按照规定的模式一闪一闪地亮着，数码管也可以显示出相应的数字等。

3.3.3 常用 FPGA 开发工具

1. Vivado

Vivado 是 Xilinx 公司开发的功能强大的 FPGA 开发软件，面向 Xilinx 公司的所有 FPGA 产品线。该软件具有以下特点。

1）强大的设计输入功能

Vivado 支持 Verilog HDL、VHDL 等硬件描述语言代码的编写和输入，提供图形化设计界面绘制电路原理图，为各种不同的设计风格需求提供方便。

2）高效的仿真与调试能力

Vivado 包含了用于仿真的模型，能快速准确地定位逻辑错误和时序问题。同时，在仿真时可通过设置断点、查看波形图、查看变量值等方式帮助设计人员查找到问题所在。

3）最先进的综合与布局布线算法

通过使用创新的综合器、布局布线器优化算法，Vivado 基于目标 FPGA 的性能及资源约束，对设计实现了深度优化，使其运行得更快且占用资源更少。在进行大规模设计时，可以通过 Vivado 降低关键路径上的时序延迟以满足高速信号处理的要求。

4）丰富的 IP 核资源

Xilinx 提供了大量已经验证过的 IP 核，包括通信类的 IP 核、信号处理类的 IP 核和存储类的 IP 核等。而在 Vivado 中我们可以方便地使用这些已经存在的 IP 核，快速搭建需要实现的子功能模块，大大地缩短了设计周期。例如，在进行以太网通信系统设计时，直接调用 Xilinx 提供的以太网 MAC 控制器 IP 核就可以了，并不一定要从头开始做。

2. Quartus Prime

Quartus Prime 是 Intel(原 Altera)公司的 FPGA 设计软件，可以应用于 Intel 公司的 FPGA 和 CPLD 器件的开发，主要有以下优点。

1）界面及操作上的友好体验

Quartus Prime 具有操作简单、易于入门的特点，无论是新手还是有一定经验的老工程师都可以轻松学习使用；Quartus Prime 的设计流程合理有序，各模块之间的联系紧密，在 Quartus Prime 中可以很方便地完成输入、仿真实现、综合实现、布局布线实现及下载编程等工作。

2）出色的时序分析和优化功能

这一工具提供精确的时序分析方法，可细致分析设计中的所有时序路径，可以准确得出每一条路径的时序裕度并根据实际分析情况对设计进行自动优化或提出优化建议。例如，在高速数字信号处理系统的高速设计过程中，可能造成时序异常问题，利用该工

具，可以快速定位问题，以优化布局布线方式或者修改设计代码的方式达到解决问题的目的。

3）高效率的综合和布局布线技术

Quartus Prime 的综合和布局布线器针对 Intel FPGA 产品特有的工艺特性做了充分的优化，可充分发掘芯片的潜能；在保持设计正确性的同时可尽量节省资源和提高系统的工作频率，如在做低功耗的设计时，可以对电源的消耗做出调整，以降低 FPGA 芯片的功耗，延长电池供电时间。

4）与第三方工具良好兼容

Quartus Prime 可以和其他很多第三方 EDA 工具进行无缝对接（如 Mentor Graphics 公司的 ModelSim 仿真工具、Synplicity 公司的 Synplify 综合工具等），能给设计人员更多的选择，如：使用 ModelSim 来实现更深一层的功能仿真和验证，然后再将仿真结果代入 Quartus Prime 进行下一步的综合、布局布线工作。

3. ModelSim

作为将硬件描述语言（HDL）用于 FPGA 设计领域的一款非常著名的仿真软件，ModelSim 具有以下特点。

1）强大的仿真能力和兼容性

ModelSim 可以对多种硬件描述语言（Verilog HDL、VHDL、SystemVerilog 等）所编写的代码进行混杂仿真实验，在整个设计阶段可以正确反映 FPGA 设计各层次（从 RTL 到门级）实现的特性及功能，不论是简单的芯片级的小型设计还是超大规模系统级的设计均能满足其实验验证的要求；同时它也支持对主流 FPGA 开发环境（Vivado、Quartus Prime 等）编译生成的设计文件进行仿真实验，并且能够被无缝集成进整个 FPGA 设计过程中，成为最基础的验证环节。

2）丰富的调试手段

ModelSim 提供了很多调试手段以及对应的视图，包括波形窗口、对象窗口、列表窗口等。这些都为设计人员实时观测信号的变化情况、得知每个时刻的变量值、了解模块的层次结构以及各个模块的连接关系等提供了极大的方便，也使迅速准确定位错误源头并解决设计问题成为可能。例如，在调试某一个较为复杂的数字电路时，在波形窗口可以看到该电路信号与时钟的关系，是否存在信号冲突或者毛刺的问题等；在对象窗口可以查看该电路模块的一些端口以及参数等内容，以判断设计是否准确。

3）自动化仿真测试功能十分高效

ModelSim 除了支持常用的 EVM 验证环境外，对 TCL（Tool Command Language）、Python 等脚本语言均有支持，设计人员可以通过编写自动化测试脚本，使用 ModelSim 完成大量设计的批量化仿真测试。对大规模 FPGA 设计来说可以大大提升测试速度，节省人力成本，降低出错概率。如需验证一个包含很多功能模块的大规模 FPGA 系统时，可编写一个测试脚本，实现系统中各个功能模块按顺序分别被测试的功能，利用该脚本可通过 ModelSim 依次对系统各功能模块实施功能测试和时序测试，并且可将相应的测试信息生成测试报告，供设计人员评估。

3.3.4 FPGA开发中常用的IP核

1. IP核的概念

IP核(Intellectual Property core)是一个已经设计好的能够执行某种特定功能的、可以重复使用的、可调用的数字电路模块。它可以被应用到各种FPGA设计中,由于经过严格的验证测试,故能保障可靠性与性能,极大地节省设计者的开发时间与精力。

例如,一款成熟的以太网MAC控制器IP核包含了以太网通信协议中的物理层及数据链路层控制逻辑,如帧的收发控制、地址过滤、差错检测等,若采用FPGA技术实现以太网通信系统,则在实际工程应用中可直接调用这一IP核实现基本的以太网通信功能,而不必深入到每一条以太网协议的细节中去进行研究和实现,能够充分地节约开发时间,工作重心可以更偏重系统的整体结构设计以及应用层的功能研发。

2. FPGA开发中常用IP核类型

1) 通信类IP核

以太网MAC控制器IP核用于实现FPGA和以太网PHY设备之间的通信控制,支持多种以太网协议标准(如10/100/1000BASE-T等),应用范围很广,可以用于各种网络设备以及工业通信或智能安防设备等。例如,在以FPGA为基础的视频监控系统中使用以太网MAC控制器IP核,可以利用以太网将视频摄像头获取到的视频数据传输给远方的服务器端进行存储或者处理。

PCIe(Peripheral Component Interconnect Express)控制器IP核可用来实现FPGA和基于PCIe总线的外设间的数据通信(如高速显卡、高速采集卡等),其可以使得FPGA系统能够获取高速外部信息资源,并向高速外部输出计算结果;而在数据中心的服务器加速、高端图像处理等领域,则能充分发挥FPGA的并行处理能力以及高速的特性,提升整体系统性能,甚至在此类应用中,利用PCIe控制器IP核还可以完成FPGA和GPU之间的异构计算协调。例如,在GPU-FPGA异构计算体系结构中,采用PCIe控制器IP核来协调FPGA与GPU之间的数据交换,这样就可以实现FPGA与GPU之间的优势互补。

2) 信号处理类IP核

在通信、雷达、音频处理等领域需要用到信号频域分析,即需要将信号从时域转成频域,然后对得到的信号频谱进行后续的信号处理算法(如滤波、谱分析等),而FFT(Fast Fourier Transform)IP核就是实现上述需求的一种重要工具,如在5G通信系统的信号调制解调中需要使用FFT核通过频域分析的方法,对无线信号进行频谱分析,从而获取其中有用的信息。

有限冲激响应(finite impulse response)滤波器IP核可用来对数字信号进行滤波、消除噪声或是抽取到特定频段范围内的信号。根据需要可以自由配置滤波器系数、阶数等。如在音频信号处理中,使用音频FIR滤波器IP核可对音频信号实现低通、高通和带通等滤波,从而得到更好的音质,在图像处理中,用FIR滤波器能实现图像边缘锐化、去噪等功能。

3）存储类 IP 核

对于 FPGA 内部的高速数据的存取需要使用 RAM IP 核，利用 FPGA 资源为算法暂时缓存一部分数据或者算法中产生的临时数据等。例如，在用 FPGA 实现某一套高清视频编码系统时，可以通过设置 RAM IP 核来将视频帧缓存起来，并将编码用到的一些数据暂存起来。矩阵运算、图像处理等各种各样的数字信号处理算法在运行时，都离不开这一部分用来高速存储数据及中间计算结果的存储单元，一般称为 RAM IP 核。这些用来做计算存储且能高速读写的内嵌存储单元都是为了配合相关的运算逻辑，存储大量供算法读写的大型数据集。

FIFO(First-In-First-Out)IP 核的作用是在数据传输速率不同或者不同模块的数据处理速度不同的情况下进行缓冲功能，如将高速 ADC(模数转换器)采集到的数据暂存于 FIFO 中，并由 FPGA 依次从 FIFO 中按顺序取出数据进行处理；再如，当一种环境的工作电压变化范围很宽时，则可以在该电路的电压检测电路与采集处理电路之间插入一个 FIFO，保证该电路在不同工作环境下采集数据的波形基本相同。

3.3.5 FPGA 开发实例——简单的交通灯控制系统设计

1. 设计目标

设计一个基于 FPGA 的十字路口交通灯控制系统，实现东西方向和南北方向交通灯的自动切换控制，模拟现实生活中交通灯的基本运行逻辑。

2. 设计思路

1）状态定义

把整个交通灯控制的过程分解成几个不同的状态，如：初始状态(所有灯为红色)，东西绿灯南北红灯状态、东西黄灯南北红灯状态、东西红灯南北绿灯状态、东西红灯南北黄灯状态等。

2）状态转换逻辑

参考交通灯工作过程，确定各个状态转换所满足的条件和转换顺序，在东西绿灯南北红灯状态下持续时长(如 30s)后再转换到东西黄灯南北红灯状态上保持时间(如 5s)，之后再转换到东西红灯南北绿灯状态下并保持时间(如 30s)，然后再转到东西红灯南北黄灯状态下并保持时间(如 5s)，再转回至东西绿灯南北红灯状态，然后一直重复此工作过程直到程序结束。

3）输出控制

根据现实情况控制该处交通信号灯(红灯、绿灯、黄灯)、人行横道信号灯亮灭变化，仿真真实路况下交通信号灯的工作模式。

3. 设计实现

1）代码编写(使用 Verilog HDL)

Verilog 代码：

```
// 定义模块的输入输出端口
module traffic_light_ctrl( input clk,        // 时钟信号
```

```
input rst,                                          // 复位信号
output reg [2:0] east_west_red,                     // 东西方向红灯控制信号
output reg [2:0] east_west_green,                   // 东西方向绿灯控制信号
output reg [2:0] east_west_yellow,                  // 东西方向黄灯控制信号
output reg [2:0] south_north_red,                   // 南北方向红灯控制信号
output reg [2:0] south_north_green,                 // 南北方向绿灯控制信号
output reg [2:0] south_north_yellow                 // 南北方向黄灯控制信号);
// 定义状态寄存器和下一个状态寄存器
reg [2:0] current_state, next_state;                // 定义状态编码
localparam S_INIT = 3'b000,                         // 初始状态
S_EW_G_SN_R = 3'b001,                               // 东西绿灯南北红灯状态
S_EW_Y_SN_R = 3'b010,                               // 东西黄灯南北红灯状态
S_EW_R_SN_G = 3'b011,                               // 东西红灯南北绿灯状态
S_EW_R_SN_Y = 3'b100;                               // 东西红灯南北黄灯状态
// 时钟周期计数器,用于控制状态持续时间
reg [5:0] cnt; reg [5:0] cnt_max;                   // 状态寄存器更新
always @(posedge clk or posedge rst)
begin
if(rst) current_state <= S_INIT;
else current_state <= next_state; end               // 计数器更新
always @(posedge clk or posedge rst)
begin if(rst) cnt <= 6'b0;
else if(cnt == cnt_max) cnt <= 6'b0;
else cnt <= cnt + 1; end                            // 状态转换逻辑
always @()
begin
next_state = current_state;
cnt_max = 6'b0;
case(current_state) S_INIT:
begin
next_state = S_EW_G_SN_R;
cnt_max = 6'b0;
end
S_EW_G_SN_R:
begin cnt_max = 6'b11110;     // 设置状态持续时间为 30 个时钟周期(假设时钟频率合适)
if(cnt == cnt_max)
next_state = S_EW_Y_SN_R;
end
S_EW_Y_SN_R:
begin cnt_max = 6'b101;                             // 设置状态持续时间为 5 个时钟周期
if(cnt == cnt_max)
next_state = S_EW_R_SN_G;
end
S_EW_R_SN_G:
begin cnt_max = 6'b11110;
if(cnt == cnt_max)
next_state = S_EW_R_SN_Y;
end
S_EW_R_SN_Y:
begin cnt_max = 6'b101;
```

```
if(cnt == cnt_max)
next_state = S_EW_G_SN_R;
end default: next_state = S_INIT;
endcase
end                                          // 输出控制逻辑
always @() begin                             // 默认所有灯熄灭
east_west_red = 3'b000;
east_west_green = 3'b000;
east_west_yellow = 3'b000;
south_north_red = 3'b000;
south_north_green = 3'b000;
south_north_yellow = 3'b000;
case(current_state) S_INIT:
begin east_west_red = 3'b111;                // 初始状态东西方向红灯亮
south_north_red = 3'b111;                    // 初始状态南北方向红灯亮
east_west_green = 3'b000;
east_west_yellow = 3'b000;
south_north_green = 3'b000;
south_north_yellow = 3'b000;
end
S_EW_G_SN_R:
begin east_west_green = 3'b111;              // 东西绿灯亮
south_north_red = 3'b111;                    // 南北红灯亮
east_west_red = 3'b000;
east_west_yellow = 3'b000;
south_north_green = 3'b000;
south_north_yellow = 3'b000;
end
S_EW_Y_SN_R:
begin east_west_yellow = 3'b111;             // 东西黄灯亮
south_north_red = 3'b111;                    // 南北红灯亮
east_west_red = 3'b000;
east_west_green = 3'b000;
south_north_green = 3'b000;
south_north_yellow = 3'b000;
end
S_EW_R_SN_G:
begin east_west_red = 3'b111;                // 东西红灯亮
south_north_green = 3'b111;                  // 南北绿灯亮
east_west_green = 3'b000;
east_west_yellow = 3'b000;
south_north_red = 3'b000;
south_north_yellow = 3'b000;
end
S_EW_R_SN_Y:
begin east_west_red = 3'b111;                // 东西红灯亮
south_north_yellow = 3'b111;                 // 南北黄灯亮
east_west_green = 3'b000;
east_west_yellow = 3'b000;
south_north_red = 3'b000;
```

```
    south_north_green = 3'b000; end
    endcase
    end
    endmodule
```

2）仿真验证

采用 ModelSim 等 EDA 工具编写测试激励程序，在 EDA 开发系统中仿真实现交通灯控制系统，检验交通灯信号灯在各自规定时间内的亮灭情况是否符合设计的工作状态逻辑及时间要求，从而检验系统的设计是否正确。

3）下载编程与硬件测试

设计代码经过验证之后通过 EDA 开发工具（如 Quartus Prime、Vivado）对其综合、布局布线后得到网表文件，将之下载到实际 FPGA 开发板上，利用开发板上的 LED 灯等外设来模拟交通灯工作状态，实现硬件测试，确定此设计是否可以在实际的硬件环境下运行。

通过本节实例可以了解到 FPGA 开发的基本过程，以及如何用硬件描述语言实现设计输入、怎样进行仿真验证与怎样将设计下载到硬件平台上进行测试。

3.3.6 小结

随着 FPGA 芯片的规模不断变大，所涉及的应用场景也越发复杂，要想设计出一个庞大的系统并非易事，其中不但要面对大量的多模块间的相互作用问题、跨时钟域数据传递问题、复杂的时序问题等，还要验证整个系统。

使用 EDA 技术及工具开发 FPGA，可以极大地提高电子系统设计的边界和跨度，为不同行业创造新的发展空间。希望能让大家对此有更全面、更深入的理解，在学习中多思考、多钻研，能够在 FPGA 的开发上有所突破和创新。

3.4 SoC 设计基础

SoC 也就是人们所说的片上系统（System on Chip）。随着现代电子产品越来越朝向小型化、高性能以及多功能发展，在 SoC 的支持下大大提升了电子设备的功能水平和整体性能，并且极大地缩短了电子产品从设计到成品的周期，可以说，正是在 SoC 设计的推动下才如今电子系统的开发方式出现了划时代的变化；同时 EDA 技术及工具也是 SoC 设计发展的有力促进者。本节内容让大家更深刻地了解 SoC 设计的魅力，多方面熟悉 EDA 技术与工具，为更好地探索电子设计领域做好准备。

3.4.1 SoC 概述

SoC 是指将电子系统的所有核心部分都放在一块芯片上，包括处理器核、存储器、各种外围的功能模块（如定时器、UART、I^2C、SPI 等通信接口模块，ADC、DAC 等模数、数模转换模块）以及相关的控制逻辑等。打个比方，SoC 就像是一座很发达的微型城市，城

里的老百姓(即处理器核)负责接收和处理消息并做出决定；城外仓库(即存储单元)存放着物资(即数据)；城中建有水电站(即电源管理模块),给该市供电；城市有交通网络(通信接口)连接其他城市；城内还建立有工厂(功能模块),生产出所需的产品(实现某种特定功能)。整个城市的各个部分相互联系又相互协调。例如,我们在手机上所看到的,目前主流手机的主芯片就是一个 SoC,它内部集成了高性能应用处理器、存储单元(用来存放中间的数据和指令)、图形处理单元(负责一些关于图像的复杂运算工作,让我们可以看到非常清晰的、高分辨率的画面以及运行一些大型 3D 游戏)、USB 接口模块(用来连接 PC 去传输数据)、摄像头接口模块(用来对接手机摄像头进行拍照录像等功能)等一些外围的功能模块,也正是因为所有的这些模块都汇聚到了一块芯片上,所以手机才拥有了丰富且强大的功能。SoC 的发展史也是随着半导体技术、集成电路设计技术和电子系统设计思想的演变而展开的。

对 SoC 设计来说,EDA 是涵盖系统设计、逻辑设计、电路设计、验证、综合、布局布线及物理验证的软件工具和方法学的一套完整体系,利用 EDA 技术,可以在计算机上实现复杂芯片的设计,大大提高了设计效率,降低了设计成本,同时也缩短了产品的研发周期。在 SoC 设计时,可以借助 EDA 技术,让设计人员从 SoC 系统的顶层结构出发,使用更高一级的抽象模型与描述语言(如用 System C 语言来描述),实现 SoC 的各种功能、性能,以及各模块间的交互描述工作,再经过 EDA 工具进行逻辑综合,将高层次描述转换成网表,并进行布局布线、生成版图,最后形成可用于生产的掩膜；此外,在设计过程中会有各种验证过程,应用 EDA 工具对 SoC 的各个部分做功能验证、时序验证和功耗验证等,保证 SoC 能够实现所期望的功能及性能指标。EDA 技术在 SoC 设计中的重要性主要体现在如下几方面。

1. 系统设计与架构探索

在设计之初使用 EDA 工具对 SoC 进行系统级建模和仿真的平台,如 Synopsys 公司的 System Studio 等。利用这样的平台工具来创建 SoC 的虚拟模型,能够快速搭建不同的系统结构供设计人员选择和评价,如确定采用什么样的处理器内核(是单核还是多核,是同构多核还是异构多核)、各功能模块间采用何种通信方式(总线架构还是新的片上网络)、分配给系统何种类型的存储资源等重要架构决策等都能够在早期得到模拟分析,并能够预测其结构在不同架构下的运行情况以及功耗大小和面积成本高低,从而可以选择出最适合该 SoC 的最佳 SoC 架构方案。

2. 硬件描述与逻辑综合

一般来讲,SoC 的逻辑设计是通过 Verilog HDL 或者 VHDL 等硬件描述语言来实现的,最后使用 EDA 中的逻辑综合工具(如 Cadence 的 Genus 综合工具)将编写的代码综合成门级网表。这个过程会根据目标工艺库、设计时序约束、面积约束、功耗约束等条件对逻辑电路进行一定的优化,如进行逻辑表达式化简以减少门数量、按照关键路径的时序来调整电路结构、用低功耗的设计方法来降低功耗(时钟门控、逻辑重构等)等,使 SoC 逻辑设计既可以满足性能要求,又能兼顾高效率、低功耗等优点。

3. 验证与调试

验证是最费时间的一个阶段，占到整个 SoC 设计周期的 70％以上。验证手段有多种，如 EDA 的功能仿真器（如 Mentor Graphics 的 Modelsim）、形式验证器（如 Synopsys 的 VC Formal）、硬件在环（HiL）验证平台等。功能仿真的主要目的是仿照功能上的逻辑来检验 SoC 在不同的层次（从 RTL 级别、到门级）是否实现了正确的设计逻辑；形式验证则用数学的方式验证 SoC 本身设计的正确性，重点是验证大模块的时序逻辑以及大小模块之间的等价性；最后，硬件在环（HiL）验证平台可以把真实的硬件板子上拥有的各种传感器、各种通信接口设备与 SoC 设计模型连起来，检查 SoC 在真实世界的运行情况。使用这些验证手段能够尽早地发现设计过程中的问题，并通过修改设计来消除潜在的问题点，保证 SoC 最终的正确性和可靠性。

4. 布局布线与物理实现

完成了 SoC 的逻辑综合和验证之后，就进入 SoC 的设计流程中的布版阶段，在该阶段用 EDA 软件中的布版工具（Cadence 公司的 Innovus）通过读入门级网表和工艺设计规则，将 SoC 中各种不同类型的逻辑单元（如标准单元、宏单元等）放置到芯片的硅片区域，并且设置各种不同功能的信号线的连线方式。同时还要考虑到如何布置电源网络，如何避免信号间的完整性问题，以及怎样保证 SoC 的设计达到时序收敛，也就是从时序的要求出发在版图实现中考虑用怎样的布局和布线方法等。在这一步骤，可以通过布版工具对 SoC 版图进行时序、信号完整性和功耗的优化。例如，改变关键路径中的逻辑元件的库表位置或拓扑结构等来减小它的时间延迟；也可以将电源线和地线布置得更加规则，这不但有助于降低功耗，也能使电源电压更加稳定。

5. 物理验证与签核

SoC 版图设计完成之后要使版图与设计意图完全一致，且满足制造工艺的要求。对于这些内容可以用以下方式来进行物理验证，EDA 工具提供了设计规则检查（DRC）、版图与网表对比（LVS）、提取寄生参数后的时序验证（Synopsys 公司的 PrimeTime 等）。具体来说，DRC 用来检查版图上有没有违反制造工艺规则的地方，如金属线间距太近或过孔大小不对等；LVS 用来比对版图和门级网表，确认版图上的逻辑与设计是否吻合；而提取寄生参数后的时序验证则需要考虑版图上寄生参数（如金属线的电阻、电容等）带来的影响，然后再次计算时序，使 SoC 在真实的制造过程中也能满足设计的时序要求。只有完成这些严苛的物理验证及签核过程之后，SoC 的设计才可以送到晶圆代工厂制造。

3.4.2 SoC 设计基本流程

1. 需求分析与系统规格定义

任何一款优秀的 SoC 设计都是在确定了需求以后开始进入到系统规格的定义阶段，在这个阶段会涉及设计团队和客户方、市场部、应用工程师等多个部门的人共同来商讨和研究讨论，具体内容包括该款 SoC 设计的产品是应用在什么产品上，该产品需要具备哪些功能特性，性能指标（即处理速度、存储容量、通信带宽等）有哪些要求，对于功耗、面积有什么样的限制要求等。

例如，对于设计物联网智能传感器节点 SoC 而言，需求分析中规定了其必须具有低功耗的特点，即传感器节点中使用的 SoC 大多使用电池作为电源，工作时间会非常长；并需要对外挂各种传感器的接口模块进行集成（包括温度、湿度、加速度等多种传感器接口）来获取不同的环境信息；除了能给外设传感器接口进行供电之外，还应采用低功耗且简单易用的无线通信方案来进行无线通信（如可以支持蓝牙低功耗协议或 ZigBee 协议），将采集的数据直接送给网关设备；由于硬件级别的无线通信仅是对采集的传感器数据进行一些简单的预处理操作（如对采集的数据进行滤波和特征提取等），因此只需要用 30～100MHz 主频的处理器就可以达到正常要求；并且将 SoC 的面积做到最小也可以契合传感器节点体积较小的特点。基于以上详细描述的需求，设计人员就可以汇总成完整的系统规格书，并将之作为整个 SoC 设计的工作指南。

2. 系统架构设计

根据系统规格说明书，设计师进入系统架构设计阶段，在本阶段要完成 SoC 的整体架构设计，具体包括确定 SoC 选用什么样的处理器核（也可以采用定制化处理器核），完成 SoC 存储层级结构的划分，以及 SoC 各功能块之间相互连接的总线或 NoC 通信架构的搭建，最后还要确定 SoC 各功能块之间的信号传输方式，定义好 SoC 内部、SoC 与外部相关子模块之间传输数据、命令、控制信号的标准与协议。

如果继续讨论上面所述的物联网智能传感器节点 SoC 架构的话，那么在注重超低功耗的前提下，可以选择一些比较适合低功耗应用的处理器内核，如 ARM Cortex-M 系列的处理器，来实现更为简单的设计。由于需要保证传感器数据的正常存储，因此可以建立缓存架构，并配合一定的片上 SRAM，预留外部 Flash，作为设备的固件存储和部分历史数据的存储器。在通信方面，可使用比较简单的总线架构将各种传感器接口模块、无线通信模块、处理器核心、存储器等连接起来，保证数据可以在各模块之间有序而高效地流通。考虑到总线通信会带来性能瓶颈及功耗方面的问题，在总线宽度、时钟频率、总线仲裁等方面应做好相关优化。此外，不同类型的接口模块需根据其使用的传感器通信协议（如 I^2C 通信协议用于连接温度、湿度传感器，SPI 通信协议用于连接加速度传感器等）或无线通信协议（蓝牙低功耗协议栈中的物理层及链路层等）对接口规则进行规约，使 SoC 能够与外部设备进行正确的交互。

3. RTL 级设计

在确定 SoC 的系统架构后，就可开始 SoC 的设计工作，即进入 RTL（寄存器传输级）设计阶段，在这个阶段利用硬件描述语言来描述 SoC 各个模块的行为以及结构，并且明确定义各模块间的数据传送关系、控制逻辑流程、寄存器的功能等。

在 RTL 级对该部分的设计进行详细描述，就是针对无线通信模块中的 SoC，将其中的功能组合按照功能和应用场景划分成一个个有限状态机的形式，再将这些功能组合的软件描述代码编译成硬件描述代码。其中，在描述蓝牙低功耗协议物理层部分时，需要定义调制解调器的电路逻辑，讲明如何将基带数据转换为射频信号并发送出去，又或者如何将接收到的射频信号通过调制解调得到基带数据；而链路层则需要设计实现蓝牙设备的连接建立、数据包的发送接收、链路的维护断开等功能的状态机，再用硬件描述语言

精准地表述各个状态机之间的状态转换以及各个状态下所要完成的操作，如连接建立状态包括哪些具体的动作；最后的数据链路层对于缓冲区的管理部分，需要使用 RTL 代码对 FIFO(先进先出)的数据结构进行编写，并且能够对数据包进行暂时性的存储、读取操作，使发送或接收的数据能够有序地通过无线通信模块。同时，每一个 RTL 的设计均基于前面所述系统架构设计中的接口规范与通信协议，使得设计的模块能够在 SoC 芯片中与其他模块自由地配合工作。

3.4.3 常用 SoC 设计工具

1. Cadence Encounter 数字实现系统

Encounter 数字实现系统是 Cadence 公司基于自家 ECAD/ICAD 核心库打造的针对 SoC 的全流程(RTL 综合、布局布线、时序收敛及物理验证)数字实现 EDA 工具集，包括从开始构建原型的早期设计探索工具到进一步细化电性性能的后端互连模拟器等在内的全部系列，堪称业界空前强大的集成化解决方案之一。

1) 强大的综合与优化能力

对于一些复杂的大规模 SoC 设计，可以对其进行高效的逻辑综合，同时根据不同设计的目标(如高带宽、低功耗、小面积等)实现针对性的深层次优化。例如，使用一些智能的逻辑重组、冗余消除技术去减小逻辑门数、降低功耗；用一些先进的时钟网络综合技术来优化时钟树，如减小时间差、提升时序性能等。

2) 高效的布局布线算法

在布局布线阶段，利用 Encounter 自带的布局布线引擎可以实现 SoC 逻辑单元快速布局，并为其制定最优的布线路径，也可以自动解决一些十分复杂的时序约束及信号完整性问题，对于信号的关键路径进行时序收敛，尽量降低信号之间的串扰及电磁干扰的风险。例如，对于高速信号总线的布线问题，可以通过优化布线的拓扑结构、线宽、间距等参数，来降低信号传输延迟以及信号衰减程度，从而保持信号的完整性。

3) 良好的多模式设计支持

对 SoC 的设计来说，大多是在不同模式下同时工作，如高性能的工作模式或低功耗的待机模式等，这就要求对于不同的模式要实现不同的时序分析和优化。Encounter 可以为这些模式建立独立的时序约束模型，并且全流程考虑这些模式下的时序要求。举个例子，在进行手机 SoC 的设计时，既要保证工作模式下具有很高的处理性能，也要满足在低功耗待机模式下需要控制功耗的目标，此时就需要用 Encounter 来达到这样的效果，尽量满足各种模式下的需求。

2. Synopsys Design Compiler

Design Compiler 是 Synopsys 公司研制的旗舰级 RTL 综合工具，在 SoC 设计中有着广泛的应用，具有灵活方便的特点。

1) 业界领先的综合技术

基于成熟的算法及丰富的工艺库支持，可以将 RTL 代码转化为高质量的门级网表；支持先进综合技术，包括低功耗综合(时钟门控综合、多阈值电压综合)、性能提高综合

(布局感知综合、时序驱动综合)、可布线性提高综合(层次化分解技术)等。例如,使用时钟门控综合方法在某些功能模块未被使用时会将该功能模块对应的时钟信号关闭以达到节省芯片动态功耗的目的;又如,布局感知综合是在综合阶段就考虑到了之后的布局布线信息,进而进行前期优化,使得后面的设计布局布线得到优化,设计效率也有所提升。

2) 强大的综合优化功能

针对用户所给出的设计约束(时序约束、面积约束、功耗约束等)来自动优化门级网表,包括自动调整逻辑门大小、自动插入缓冲器和反相器进行时序优化、自动合并冗余的逻辑表达式等,从而使设计得到最优的性能与资源利用率。例如,在遇到比较紧张的时序路径的情况下,可以在该路径上增加一些缓冲器以提高驱动能力,降低时序延迟,使关键路径满足时序要求;但又要考虑到整个综合过程不能过多牺牲面积或者功耗。

3) 良好的可扩展性及可集成性

集成功能良好,拥有很好的可扩展性,可以与 Synopsys 的其他 EDA 工具(如 Star-RCXT 寄生参数提取工具、PrimeTime 时序签核工具等)一起集成实现完整的 SoC 设计流程;还可实现多操作系统平台/多硬件加速环境下的部署应用,如大型 SoC 设计项目在大规模版图铺布完成后与 Star-RCXT 相结合得到版图寄生参数,将这些参数反馈到 Design Compiler 做后端综合优化以获得更准确的时序和更好的设计性能。

3. Mentor Graphics ModelSim

ModelSim 是 Mentor Graphics 公司开发的功能强大的仿真工具,被大量应用于 SoC 设计验证阶段,其功能特点如下:

1) 支持全面仿真语言且兼容性强

ModelSim 支持 Verilog HDL、VHDL、System Verilog 等主流硬件描述语言的混合仿真,并且仿真精度足够高,能真实地反映 SoC 设计在不同抽象层(从 RTL 到门级)的行为和功能。从简单的数字模块,到复杂的混合信号 SoC 设计模型(利用协同仿真的手段与其他工具配合实现),ModelSim 的仿真效果都是非常准确可靠的。同时,ModelSim 与当前大多数主流 EDA 工具链具有较好的兼容性,能很好地融合于整个 SoC 的设计流程中,并作为验证的重要工具使用。

2) 调试高效且可视化

ModelSim 具备众多功能强大的调试工具以及图形化的界面,如波形窗口、对象窗口、列表窗口等,在仿真过程中,可以随时查看信号的状态、查看变量的当前值、分析模块之间的层次结构及连接关系等,找到问题并解决问题。例如,调试 Soc 中的总线仲裁模块,能在波形窗口中看到各个主设备发出的仲裁请求信号以及总线使用权的应答信号之间的时序关系,由此可判断其是否存在问题(如有没有信号间的冲突或者时序上出了错误);从对象窗口中可以获得该模块端口的相关信息与其中的参数配置情况等。还可采用 Tcl 脚本编写自动化的测试脚本以实现设计的大批量仿真验证,从而提高验证效率。

3) 验证能力强且支持覆盖范围检查

ModelSim 不仅能够进行基本的功能仿真,还具备强大的验证能力。它支持形式化

验证技术，能够对设计进行形式化的属性检查，发现隐藏的错误和不一致的地方。此外，它还提供详细的覆盖率分析功能，包括代码覆盖率、分支覆盖率、条件覆盖率等，可帮助设计人员评估验证的充分性，确保设计经过充分的测试。例如，在验证一款处理器 SoC 的指令执行模块时，通过覆盖率分析，可以了解各个指令的执行路径是否都被测试到，对于未覆盖到的路径，可以有针对性地补充测试用例，提高验证的完整性，从而提高 SoC 设计的质量和可靠性。

3.4.4 SoC 设计中的关键技术

1. 低功耗设计技术

1）电源管理策略

SoC 的低功耗设计需要从整体电源管理策略开始考虑，如可以使用多电源域的方式来划分 SoC 的不同功能模块区域，每个区域都可以根据自己所处的某个状态（是处于高性能的工作状态，还是低功耗的待机状态或睡眠状态）来进行电源控制。如移动设备中的摄像头功能一般不经常使用，则可关闭摄像头接口所在电源域，断开供电，彻底消除其中的无用功耗；对于需要始终开启工作的核心模块（如一些处理器核心的部分），给予稳定的供电，但也可以使用动态电压频率调节（DVFS）技术根据工作负载的大小来动态地调节电源电压或者工作频率来达到降低功耗的效果。像处理器在工作量较小时就将工作频率下调，相应的电压也下调以降低功耗；如果工作量较大，则将电压和频率升高，这同样可以在保证整个系统性能的情况下降低功耗。

2）电路层级的低功耗设计技术

在电路层级，使用了诸如时钟门控、逻辑门关闭、寄存器时钟门控等技术。时钟门控是最为常见的低功耗技术，如果某个模块当前阶段不需要被时钟信号驱动，那么可以让该模块的时钟信号通过门控电路进行阻断；这样，在不需要时钟翻转时就达到了节能的效果。逻辑门关闭是根据模块的状态去决定使能部分逻辑门来达到降低功耗的目的，即降低静态功耗以及动态功耗；寄存器时钟门控则是通过控制寄存器的时钟输入端，只有寄存器要更新数据时才开启寄存器的工作，只有这样才会发生时钟翻转，避免无效时钟翻转所造成的能耗。例如，在音频处理 SoC 中，如果没有音频输入数据，则可关闭音频输入缓冲区逻辑门的时钟信号，并关闭部分寄存器的时钟使能；在有音频输入数据时，恢复其时钟信号及寄存器操作，以启动音频处理功能。

2. 高性能设计技术

1）并行处理架构

使用并行处理架构可以提升 SoC 的性能，如多核处理器架构可以让多个处理器核心在同一时间同时对不同的任务或数据流做并行处理；对图像处理 SoC 来说，可以采用几个处理器核心协作的方式，一个核心负责完成图像预处理（滤波、灰度化等），另一核心负责完成图像特征提取工作，再有一个核心负责完成图像识别与分类等工作；各核心之间通过共享内存或者消息传递机制来相互传送数据及同步工作。除了以上所说的多核处理器架构之外，还有硬件加速器与处理器相结合的异构并行处理架构，即采用更多的专

用 CPU 内核和 GPU 或 FPGA 来为数据中心服务器 SoC 提供可编程、高性能和低成本的可重配置的硬件加速模块，使其能够完成一些传统上需要服务器群和巨大开支才能做到的大规模并行处理数据量的计算工作。例如，在大数据中心服务器 SoC 上面会存在常见的各种 CPU 通用核心，并且会额外集成 GPU 以及 FPGA，其主要用来完成一些大规模并行数据运算（如深度学习中的大规模矩阵计算）以及用 FPGA 实现基于某算法的可配置硬件加速器。

2）高速通信接口与总线架构

高性能 SoC 的设计离不开高速通信接口和高效的总线架构，高速通信接口让 SoC 可以快速和外界的设备进行通信，如 PCIe（Peripheral Component Interconnect Express）有较大的带宽和较小的延迟可完成与对外接口（如用于连接高性能显卡或者高速的数据采集卡等）的通信，实现数据的大容量传输；对于移动 SoC 来说，则选用 USB 3.0 甚至更高版本的接口，在文件复制的过程中达到更快的速度，为用户带来更好的数据交换体验。高效的总线架构可实现 SoC 内部各模块间数据传输和通信，如基于 AMBA-AXI 总线协议高效完成多主设备和多从设备之间的通信，具有分层的总线架构，并且该总线架构可灵活设置总线宽度、时钟频率等相关参数，以便满足不同模块间的通信带宽与时序要求，有效解决数据传输瓶颈，提升整个 SoC 的处理能力，例如，某款高性能网络处理器 SoC 通过对 AXI 总线的改进配置，使得网络处理器的各个网络接口可以同时向处理器的核心以及缓存和主内存传输数据，最终实现一路或多路网络数据的同时高效处理和转发。

3. 可测性设计技术

1）边界扫描测试技术

边界扫描测试技术即 JTAG 测试技术。边界扫描测试是常用的 SoC 可测性设计的一种测试技术，采用 IEEE 1149.1 JTAG（联合测试行动组）标准，在 SoC 芯片的每一个引脚旁添加边界扫描单元（BSU），将所有引脚边上的 BSU 串联成边界扫描链。通过给边界扫描链输入测试激励信号，并捕捉到输出响应信号的方式实现 SoC 芯片引脚互连、输入/输出缓冲区等的测试。例如，测试 SoC 与外部存储器之间连接是否正常，可以通过 JTAG 接口向边界扫描链发送测试数据，然后通过边界扫描单元将这些数据送到外部存储器接口引脚上，让存储器将相关响应信号回传给边界扫描单元，再经过边界扫描单元将响应信号捕获回传，最后通过判断这些响应信号是否符合期望要求来判断开路、短路等情况。此测试无需复杂的测试设备及大量的测试工具，大大地简化了 SoC 的板级测试及系统级测试的过程，提高了测试效率及覆盖率。

2）内部自测试（Built-In Self-Test ，BIST）技术

自测试（self-test）技术是将自测试电路集成在 SoC 芯片内部，使其能够自我检测相关的重要功能块（如存储器、组合逻辑电路等）的功能是否出现异常情况。如果被检测功能块为存储器，那么可以通过内置的内存 BIST（Memory BIST）技术来实现，即在 SoC 制作完成后，使用芯片内置的测试控制器自动生成所需的地址、数据和控制信号对存储器执行读写操作，检查其是否存在单元故障、线短路、线开路等问题；对处理器核心中的算术逻辑单元（ALU）和浮点运算单元（FPU）等复杂的逻辑电路，可以采用逻辑 BIST

(Logical BIST)技术,设计相应的逻辑 BIST 电路,将伪随机测试向量产生器产生的测试激励信号输入待测逻辑电路的输入端上,经由响应压缩电路对测试逻辑电路输出的测试响应信号进行压缩后输出,再根据经过压缩后的测试响应信号来判断逻辑电路是否正常工作。例如,采用 BIST 技术对性能优越的 DSP SoC 的多组滤波器模块进行自测试,并保证每一组滤波器均能将信号处理好,并且能够及早发现相关芯片制造过程中的缺陷,提高其可靠度,降低其生产成本。

3.4.5 SoC 设计实例——一个简单的物联网 SoC

1. 设计目标

设计一款应用于智能农业环境监测的物联网 SoC,实现对农田环境参数(如温度、湿度、光照强度、土壤湿度等)的实时采集、处理、无线传输以及简单的本地控制功能(如根据土壤湿度自动控制灌溉系统启停)。

2. 设计思路

1) 系统架构规划

采用 ARM Cortex-M3 处理器作为 SoC 的核心控制器,其功耗低、性能高、实时性强的特点适合用作物联网终端设备。集成传感器接口模块:I^2C 接口用于连接温度传感器与湿度传感器,SPI 用于连接光照强度传感器和土壤湿度传感器;集成蓝牙低功耗通信模块,用于将所获取的环境数据传送到农场主的手机和平板等监控终端;设计灌溉控制模块包括控制灌溉水泵电机启停的电机驱动接口、相应控制逻辑电路等,通过土壤湿度传感器采集数据并结合预先设置好的灌溉阈值实现自动灌溉控制;配置了一定容量的 SRAM 来存放采集的环境数据以及中间处理的数据结果,并且有一小部分容量的 Flash 来存放 SoC 的固件程序以及一些基本配置参数等。

2) 功能模块设计

(1) 传感器接口模块。

每一个传感器接口子模块(如 I^2C 接口模块,SPI 接口模块)均要完成对应传感器相关的通信协议配置、数据采集启动和停止控制、数据接收暂存等功能。以 I^2C 温度传感器接口为例,采用硬件描述语言(HDL)编写寄存器配置逻辑,配置 I^2C 总线的通信速率、设备地址等,设计状态机完成 I^2C 总线的启动、发送读取命令、读取温度数据、停止通信等工作,并将读取到的数据暂存到 FIFO,等待处理器核心取出进行进一步的计算处理。

(2) 蓝牙通信模块。

该模块主要包括蓝牙基带控制子模块、射频收发子模块、协议栈处理子模块。基带控制子模块:根据蓝牙协议规范将数据分成数据块,按照一定的规则对数据进行编码,并进行差错检测及纠正等操作;射频收发子模块:实现射频信号的发送和接收,完成对射频信号的调制解调、频率合成、功率放大等功能;协议栈处理子模块:运行蓝牙协议栈,进行蓝牙设备的连接建立、数据传输、连接断开等工作。利用硬件描述语言和相关的 IP 核(如蓝牙 PHY 层 IP 核)将以上 3 个子模块组合集成在一起,最终实现蓝牙模块与外部蓝牙设备通信的功能。

(3) 灌溉控制模块。

作为电机驱动接口电路,该模块主要是使用 PWM 信号来实现水泵电机的启停及转动控制;此外还实现了土壤湿度数据处理与决策子模块。土壤湿度数据处理与决策子模块将通过传感器接口模块采集来的土壤湿度数据进行滤波、校准等预处理操作后与设置好的灌溉阈值进行比较,在土壤湿度低于设定下限时给电机驱动接口电路发出开启灌溉指令,在土壤湿度达到或者大于设定上限后给出停止灌溉指令,可以做到智能灌溉,从而更加节水并且提高农业灌溉效率。

3. 设计实现与验证

1) RTL 级代码编写

可使用 Verilog HDL 完成 SoC 各功能模块的 RTL 级编程实现,编写 ARM Cortex-M3 处理器的配置代码,针对智能农业监测应用,对处理器缓存大小、中断优先级等进行配置;编写传感器接口模块、蓝牙通信模块、灌溉控制模块、存储控制模块等 RTL 代码,保证各个模块的功能逻辑正确性,并且满足系统架构设计中定义的接口规范及通信协议。编写代码时要注意保持代码的可读性、可维护性和可移植性,应遵循良好的编程规范和风格,如:模块化设计方式,将不同的功能代码放在单独的模块内,用参数化方式提高代码的通用性和灵活性,以方便后期验证和修改。

2) 仿真验证

利用 ModelSim 对设计的 SoC 进行功能仿真,编写测试激励程序模拟农田各种环境参数变化(包括温度从 0℃变到 40℃,湿度从 0%变到 100%,光照强度从 0Lux 变到 100 000Lux,土壤湿度从干燥变到湿润等),查看这些变化的数据是否能够被传感器接口模块采集,进而传送至处理器核心;检验蓝牙通信模块在不同的通信距离和不同的干扰环境下能否正确地将采集的数据传递给监控终端,使监控终端判断其能否与监控终端匹配成功,并保持稳定地向监控终端传递采集的环境数据;检验灌溉控制模块是否能在不同的土壤湿度条件下做出正确的控制逻辑决策,灌溉系统能否根据需求正确地停止或开启,最后使得 SoC 能够实现智能农业灌溉的目的。

3) 综合与布局布线

先将 RTL 代码在 Synopsys Design Compiler 中通过功能验证后生成门级网表,并在综合时设定相关的时序约束、面积约束、功耗约束,选用适合目标工艺的单元库及 IP 核对设计进行优化,以实现智能农业物联网 SoC 的性能、功耗要求;再将门级网表导入 Cadence Encounter 进行布局布线,制定 SoC 芯片的物理布局方案,对布线拓扑结构进行优化,保证信号完整、时序收敛、功耗控制等。同时对关键路径的时序进行优化,如传感器采集数据关键路径、蓝牙实时传送数据路径以及灌溉快控关键路径,在此期间应注意调整单元布局布线等,从而减小时间延迟,提升 SoC 的运行速度和稳定性。

4) 物理验证与签核

进行布局布线后,采用 Cadence 公司的物理验证工具,执行设计规则检查(DRC)、版图与网表对比(LVS)等操作,校验版图是否正确,确保版图与门级网表相同,并且没有违反制造工艺规则的错误。另外,用 Synopsys PrimeTime 对提取完寄生参数后所得的时

序进行验证，并用 Cadence 的 Star-RCXT 来提取版图寄生参数（如金属线的电阻、电容等），再计算 SoC 的时序，保证 SoC 上的各个模块在设计中的时序要求能够在实际制造出来的芯片中得以实现，使 SoC 可以正常稳定地运转。经严格的物理验证和签核之后，就可以将智能农业物联网 SoC 设计交给晶圆代工厂去制作生产了。

3.4.6 实时操作系统应用

实时操作系统（RTOS）的应用在嵌入式系统领域至关重要，它就像隐藏在众多智能设备背后的“大脑管家”，确保各项任务按时、高效地完成。

1. RTOS 概述

1）定义

实时操作系统能够实现规定的实时要求和时间约束，各种具体设备的运行就像一场宏大的交响乐演出一般，如果一个节奏没有掌握好，就可能造成整场演出紊乱失谐；实时操作系统所要面对的就是种种类似的实时任务。例如，汽车的防抱死制动装置（ABS）要求在一定的时间内就根据速度传感器的数据进行制动压力调整，假如停留了 0.2s，则可能造成车辆制动的效果变差，进而可能会造成严重的安全事故。

2）特点

（1）实时性。

RTOS 是一套适用于实时处理的操作系统，包括硬实时和软实时两种。硬实时系统的任务需要在规定的截止时间前全部完成，如航天航空器的控制，如果有任务没有及时完成就会带来灾难性的结果；而软实时则可以在一定范围内允许偶尔出现的少量超时，如平时使用的视频播放软件，如果在网络出现时有时无的情况时，可以让其略微滞留之后再继续播放下去，并不会对用户的体验造成严重的影响。

（2）多任务处理。

RTOS 能同时管理多个任务，就像一个优秀的杂要演员，能同时抛接多个球。例如，在智能手表中，RTOS 可同时处理心率监测、显示时间、接收手机通知等任务，让手表功能丰富且高效运行。

（3）高效性。

RTOS 通常体积小巧、运行高效，资源占用少。以 FreeRTOS 为例，它内核小巧，能在资源受限的微控制器（MCU）中流畅运行，适合各种小型嵌入式设备，如智能家居的传感器节点等。

2. RTOS 主要功能模块

1）任务管理

任务是 RTOS 中的基本执行单元，就像一个个有特定职责的员工。RTOS 能创建、删除任务，还能对任务进行调度。

（1）先来先服务（FCFS）。

先来先服务就像排队买票，先到的任务先执行。但若前面任务执行时间过长，后面紧急任务就只能干等。

(2) 优先级调度。

优先级调度是指为任务设定优先级,并按照从高到低依次完成,如在工业自动化控制系统中,紧急停机属于优先级最高的任务,当出现该类任务时,其他任务都会立刻停止运行。

2) 时间管理

RTOS 支持系统时钟、时延与定时器。在物联网设备的数据采集系统中,可以通过定时器设置固定的定时间隔周期采集传感器数据,如 1s 采集一次传感器的温度数据,从而保证数据采集的规则性和实时性。

3) 内存管理

合理利用内存对 RTOS 来说是一大要务,这里的内存管理方式采取动态分配和静态分配相结合的办法,在一些要求严格的航空航天系统中常使用静态分配的方式给每一个任务分配固定的内存大小,这样可以规避一些动态分配所带来的内存碎片以及分配延迟的问题。

4) 通信与同步

任务之间可以相互沟通和同步工作。常见的通信方法有如下几种。

(1) 消息队列。

任务可以将消息发送到消息队列,另外的任务可以从队列中获取消息。对于多传感器数据融合系统来说,多任务分别使用对应的传感器进行数据采集,将采集的结果通过消息队列发送给数据处理任务,完成数据的汇集及处理。

(2) 信号量。

信号量是用于任务间同步的一个组件,它像一把“钥匙”,获取这把“钥匙”的任务才可以进入特定区域执行,完成之后再将这把“钥匙”还回去。这样可保证某个共享资源(如打印机)一次只能被一个任务访问,避免数据混乱的情况发生。

3. RTOS 在不同领域的应用案例

1) 工业自动化

在自动化生产线上,RTOS 应用广泛。下面以汽车制造车间为例具体说明。

(1) 机器人控制。

若干工业机器人协同作业,完成对汽车零部件的焊接和组装等工作,每一个机器人都有一套实时操作系统(RTOS)控制其自身的动作任务,RTOS 精确分配各个机器人的任务,保证焊接、组装的任务在规定时间内完成,而且可以实现多个机器人协同工作,提高生产效率、产品合格率。

(2) 生产数据监控与采集。

生产线大量使用传感器对设备的运行状态和生产环境的相关参数(如温度、湿度)及产品相关质量数据进行采集,并上传至监控系统中。当出现异常数据时(如设备温度过高),可以及时触发警报,在第一时间进行操作,以减少因设备故障导致的停机时间,保证生产的顺利进行。

2）消费电子

（1）智能手机。

RTOS在手机中的大部分功能模块都起着至关重要的作用，对音频播放来说，保证实时性是非常重要的，对音频的实时解码、播放应避免出现卡顿或者中断的情况，必须能够实现快速反应；当有来电时可以及时切换任务，先停止正在执行的任务，优先处理接听电话的任务。

（2）智能家电。

例如，用RTOS控制智能空调的制冷/制热、风扇转速、显示等功能，根据室内温度传感器及用户设置的温度来动态控制压缩机的启停及运转频率和风扇转速，并且实时更新显示信息，包括温度、模式等，达到使用良好和节能舒适的目的。

3）汽车电子

（1）动力系统控制。

由于在汽车的动力系统中，RTOS将根据曲轴位置传感器、空气流量传感器等大量传感器的信息对发动机进行实时计算，控制燃油喷射量以及点火时刻等重要参数，因此也可以使发动机在各种不同的工况下达到最优的运行状态，并且达到节约燃油和减少尾气排放的目的。

（2）车载信息娱乐系统。

RTOS可以实现多任务并行，对实时性要求较高的任务可以放在同一时刻并行运行。在行车时，RTOS会随时对地图进行刷新，同时也会计算导航的最佳路径，并会不断地获取最新的路况数据更新当前正在使用的路线，并且不会妨碍声音的播放与蓝牙电话的正常使用。

4. RTOS开发流程

1）需求分析

与嵌入式系统的开发一样，首先应该确定应用领域、需要的功能点、对性能指标的要求等。如在开发一个基于RTOS的智能医疗监控设备时，应清楚需监控的各项生理指标（如心率、血压、血氧等），还需要明确对于心率数据采样显示的周期、功耗等方面的限定，对于此类相关的参数，还应明确医疗终端与该设备通信等的实现方式。

2）RTOS选型

根据应用选择合适的RTOS，常用的RTOS有FreeRTOS、VxWorks、ThreadX等。FreeRTOS开源免费，社区支持强大，适用于中小企业开发小型化、中型化嵌入式设备；VxWorks是一款稳定性极高且性能非常出色的RTOS产品，通常应用于航空领域、航天领域以及通信领域等；ThreadX则是因为其代码量小、开销低、软件开发及移植工作简单等特点被广泛应用到物联网设备中。

3）硬件设计

应根据RTOS特性和功能需要进行硬件的选择和电路的设计。要注意硬件是否支持RTOS的运行，如芯片是否拥有足够的内存、运算能力以及可以适配RTOS和该应用任务的外设接口等。例如，对采用FreeRTOS设计的无线传感网络节点来说，应选用一

种适合低功耗并且内存和存储容量大的、带无线通信接口的微控制单元(MCU),并根据实际应用需求设计相应的电源管理电路以及传感器接口电路等。

4) 软件开发

(1) 移植 RTOS。

将选择的 RTOS 移植到目标硬件平台上,在这个过程中要对移植选型的 RTOS 配置启动代码、时钟、中断控制器等。针对硬件结构化特点对 RTOS 内核进行调整优化,在此基础上使得 RTOS 可以在硬件上完成启动和运行。

(2) 应用软件开发。

基于 RTOS 的 API 编写应用程序,完成创建/销毁任务、配置时间管理、任务间的通信及同步等功能,在开发一个智能安防监控摄像头系统时,可以创建视频采集任务、运动检测任务以及网络传输任务等,并利用消息队列实现视频数据从采集任务到网络传输任务的传递,利用定时器做定期拍摄并上传监控数据等操作。

5) 调试与优化

(1) 调试。

使用调试工具(如 J-Link、GDB 等)对系统进行调试,查看任务运行状态、分析中断处理的情况、排查任务之间通信的问题。例如,当发现系统出现任务死锁的问题时,通过调试工具发现系统的任务由于信号量互斥使系统陷入死锁,则应通过对任务信号量互斥执行方式进行更改来解决死锁问题。

(2) 优化。

通过调整任务优先级、优化代码结构、减小程序内存占用等方式可对系统在实时性、性能、功耗等方面进行优化。在功耗敏感型物联网应用中,针对 RTOS 进行优化调整(如控制 RTOS 睡眠模式的配置),如正确地选择需要进行的任务调度的时间点等,可以降低设备功耗,提升电池使用时间。

6) 测试与验证

(1) 功能测试。

功能测试主要是确认系统应该具备的功能是否正常。例如,在人体健康监视智能设备的研发中,测试心跳测量功能是否稳定、准确,数据结果的显示是否正常,与上位机软件、手机应用程序等的通信是否正常等。

(2) 性能测试。

性能测试主要是检验所开发系统的响应时间、信号处理实时性、数据通信的容量等性能指标。例如,在自动化工业控制系统中,主要测试主控系统对各个分传感器反馈信号的处理时间是否满足工业生产要求,系统通信数据处理的带宽是否能满足大规模的数据传输等。

(3) 可靠性测试。

可靠性测试包括通过温度环境测试、稳定运行测试、抗震动测试等方式验证所设计系统的多个方面的可靠性。例如,针对某环境智能监测设备,需要对设备在户外条件下的长时间运行进行测试,还需要模拟设备在高温、低温、各种湿度等环境下的测试,进一

步保证系统正常运转、不出错，从而避免出现系统崩溃和数据丢失的情况。

5. RTOS 发展趋势

1）物联网融合加深

由于 RTOS 与物联网发展进程趋同，两者将会逐步发展到有机融合的状态；其中，在 RTOS 技术的参与下，可以实现在各种类型的智能设备（智能灯泡、智能门锁、智能窗帘等）之间进行互联互通，并且 RTOS 会对智能终端设备的本地控制以及同云端的通信进行处理，最终实现终端设备之间进行协同工作和远程操控智能终端设备的效果。

2）安全性强化

金融、医疗、工业控制等行业对安全性的要求非常高，这就促使 RTOS 采用更加先进的安全手段，如添加更多的安全机制，包括加密算法、安全启动、访问控制等。例如，在医疗设备上使用 RTOS 后，利用加密技术保护了患者的数据，数据不能被他人截获、读取或篡改。

3）人工智能结合

RTOS 和人工智能结合是未来的一个发展趋势。在智能安防系统中，RTOS 控制的摄像头系统会融合图像识别、行为分析等人工智能算法，实现实时调度人工智能任务。通过监控视频进行智能分析，能及时检测出异常行为并发出报警，极大地提高了安防监控的智能化程度。

RTOS 是嵌入式系统的一项关键技术，其应用非常广泛，在许多领域都起着很重要的作用。希望读者在学习了解 RTOS 之后，可以掌握 RTOS 的全方位的应用知识，在以后的工作学习中学以致用，用 RTOS 开发出更多优质的产品。

3.4.7 小结

由于物联网（IoT）、人工智能（AI）以及 5G 通信等新技术的发展，SoC 要实现的功能越来越多，其集成的功能模块也会越来越多，各功能模块之间也要进行很多交互，因此 SoC 设计的复杂度将呈指数级增长。SoC 是现代电子设计中的一项核心技术和方法，集成了很多最新设计理念、技术以及方法学，电子器件将向着越来越智能化、性能更高、体积更小的方向发展。不可否认的是，随着先进的制造工艺使得实现其功能的半导体晶圆尺寸不断缩小，SoC 也面临很多问题和挑战，但是对于未来的发展来说，其应用前景十分光明。

3.5 DSP 设计基础

DSP（Digital Signal Processing）即数字信号处理。在现代电子设备中，每时每刻都有成千上万的数字信号穿梭于不同的部件之间，这项技术可以对信号进行各种复杂的运算和变化，筛选并留下想要的部分，剔除不想要的部分，使信号更加清晰有用。

3.5.1 DSP 概述

从自然界中获取的声音、温度、光线强度等均属于连续信号，即模拟信号；将其转换为数字化后的信号即是离散信号（以二进制的形式存储和表示），并以此对各种模拟信号

进行处理。DSP 就是对上述这些数字信号进行加、减、乘、除等各种运算及相关操作，通过滤波、傅里叶变换、相关性分析等方式完成对信号的放大、压缩、提取等功能。例如，当下各款音乐 App 上的各种音频特效，如混响、均衡器等，这些功能实现的特性都依赖于 DSP 技术，例如，在给歌曲加混响时，通过 DSP 算法将音频信号按照事先设置好的参数进行运算处理，模拟出某种空间音响效果产生的回声效果，使音效更丰富，更具层次感。DSP 具备以下特点。

1. 高速性

高速性是指数据处理速度非常快，能在很短的时间内做大量数字信号数据的处理工作。这是因为 DSP 的硬件架构与众不同（如采用了哈佛架构），数据总线和指令总线分道扬镳、各司其职，可以做到取数与读数同时进行，大大提升了运行速率；再就是在内部还嵌入了一些用于数字信号处理的专门的运算法则（如乘法累加操作），这种运算法则就是在一个指令周期内就可以做到既能做乘法又能做加法，对于复杂的 DSP 算法（如 FFT 算法）可以很快计算出结果。

2. 灵活性

DSP 设计主要通过编写软件程序来实现各种功能，这就使得它具有很强的灵活性。一旦硬件平台确定，如选定了一个 DSP 芯片，通过修改软件代码，就可以轻松地改变信号处理的算法和功能，以适应不同的应用场景和需求。例如，一个无线通信基站的 DSP 系统，当需要更新通信协议或者优化信号处理算法时，只需要对软件进行更新，而不需要更换硬件设备，大大节省了成本和时间。

3. 可编程性

这与灵活性密切相关，DSP 处理器提供了丰富的编程指令集，这些指令集专门为数字信号处理任务量身定制，涵盖了从简单的算术运算到复杂的信号变换等各种操作。工程师们可以使用高级语言（如 C、C++）或者汇编语言，按照自己的设计思路编写程序，对 DSP 处理器进行编程，实现个性化的信号处理方案，就像搭建积木一样，将不同的功能模块组合在一起，构建出复杂的功能系统。

3.5.2 DSP 设计流程与工具

1. 需求分析与系统规划阶段

这时应当首先了解 DSP 系统需要完成什么样的功能，需要满足哪些性能指标，例如，为实现一个语音识别 DSP 系统，需求分析时就要确定语音识别的准确率、能识别的语音命令种类、系统的实时性指标（如在多少毫秒内必须完成识别，然后做出回应等），还需要考虑其输入/输出接口，包括麦克风输入口，与外界相连的 USB、蓝牙等接口。

以上述要求为基础开展整体系统规划，在此基础上确定选用何种类型的 DSP 芯片或 FPGA，确定整个系统架构。如果选用 DSP 芯片，则需要考虑其运算性能是否符合要求、内存大小是否够用、外设接口是否可以满足系统的要求；如果选用 FPGA，则需要考虑其逻辑资源是否满足应用要求、硬件描述语言的支持程度、开发工具是否简单易用等。在这个过程中，EDA 工具中的系统设计软件可以帮助设计师做系统级的设计与仿真工作，

如采用 MATLAB/Simulink 搭建系统的高层级模型以实现对系统功能与性能的粗略验证，并自动生成对应 HDL 代码，为后面的设计提供硬件层面的支持。

2. 算法设计与验证阶段

算法是 DSP 系统的核心，算法的好坏决定着 DSP 系统能否充分发挥其优良特性。设计阶段一般会利用 MATLAB 等数学软件进行算法的设计与验证，在此平台上，可以调用 MATLAB 的各种数字信号处理工具箱（包括各种滤波器设计函数、变换算法函数、统计分析函数），快速地实现算法的设计及调试。例如，利用 MATLAB 平台进行图像处理算法的设计，在图像处理工具箱内使用其内部函数变换算法（如二维 FFT 等）完成图像的频域分析，并使用适当的滤波器对图像进行滤波去噪操作，再以图像的可视化工具来判断滤波前后的效果，以此判断所设计算法的正确性。

由于 MATLAB 算法在验证以后，还要将其转换成 HDL 代码才能放到硬件上运行，这里就需要 EDA 的算法综合工具，它可以将 MATLAB 代码或算法的数学模型转换成 HDL 代码，并且还可以优化代码的结构和代码质量，使之更适合用硬件实现。如 MATLAB 中的 HDL Coder 就可以根据算法自动生成 Verilog 或 VHDL 代码，同时还提供各种优化选项，如流水线、资源共享等，以此来达到更好的硬件实现。

3. 硬件设计与实现阶段

到了这个步骤，将基于 EDA 工具完成硬件电路设计实现。在这里要说明一下原理图设计，首先使用原理图输入工具（Cadence 的 OrCAD 或 Mentor Graphics 的 PADS 等），绘制出 DSP 系统硬件原理图，包括 DSP 芯片、存储器、输入/输出接口电路、滤波电路等各模块之间的相互连接；然后完成 PCB（印制电路板）设计，使用 PCB 设计软件（Altium Designer）完成布局布线，按照信号完整性、EMC 的要求把电路板上的各器件放置好，并连接上对应的焊盘，最后生成各种文件并下发到制造部门完成相关硬件的制作。

同时，对于使用 FPGA 实现 DSP 设计的过程，需要先用 HDL 编辑器编出相应的代码，然后借助 EDA 工具完成代码的综合、映射以及布局布线等工作，并将得到的代码下载到 FPGA 开发板上进行硬件测试。举例说明，运用 Xilinx 公司开发工具 Vivado 实现 DSP 的设计过程：通过 Verilog HDL 语言编程实现数字信号处理算法；利用综合工具将 Verilog HDL 代码转化为网表；再进行布局布线；将网表中的逻辑电路映射到 FPGA 上；将配置文件用下载电缆下载至 FPGA 开发板，让 FPGA 开发板按照设计好的算法来运行程序即可。

4. 测试与验证阶段

通过测试和验证来证明 DSP 设计是否正确，是必不可少的一个步骤。一方面，需要利用示波器、逻辑分析仪等测试仪器，对 DSP 设计的硬件电路的电源、时钟、输入/输出信号等进行检测和测试，检验电路是否按照设计的要求运行；另一方面，则需要做系统功能和性能测试，将整个 DSP 系统放到实际应用环境中，向系统提供相应的信号，并观察系统输出是否符合预期功能及性能要求。

例如，在测试数字通信系统中的 DSP 发送和接收模块时，利用信号发生器输出模拟

通信信号，经发送模块处理后，在通道模拟器中模拟实际通信环境下的噪声、干扰信号，再将信号送入接收模块，然后用频谱分析仪、误码率测试仪等测量仪器来测量、测试接收模块解调出来的信号的误码率及频谱特性等是否达到通信标准的要求。如测试结果不满足通信标准的要求，则返回前面所述的设计环节，对算法、硬件电路等进行改进或修改，直至系统性能达到要求为止。

3.5.3 典型 EDA 工具在 DSP 设计中的应用案例

1. MATLAB/Simulink 在 DSP 算法开发中的应用

MATLAB 是应用于科学研究及工程领域的常用数学软件，在 DSP 中功能很强。Simulink 是 MATLAB 的一个图形化仿真工具，使用 Simulink 能够采用模块化的方式实现 DSP 算法的仿真。

例如，在使用 DSP 系统设计一个用于噪声抵消的自适应滤波器时，在 Simulink 中可以通过输入信号源模块（产生带噪信号）→自适应滤波器模块（选择 LMS 算法或 RLS 算法）→误差信号计算模块→自适应控制器模块这一链路完成整个系统的建模。设置相关模块参数（滤波器阶数、步长因子等），仿真得到随时间变化的误差信号，由此判断系统是否具有噪声抵消的能力，即能否让系统输出接近原始无噪声信号；仿真的同时也能看到对各种参数的调整，如不同的滤波阶数、不同的步长、不同截断的方式、不同权值更新规则等在系统中的运用，有利于我们比较不同的结果，迅速确定出最优的算法参数。同时 Simulink 可以方便地对接各种硬件平台实现代码生成与部署，经过验证的算法模型可以直接被转换成 HDL 代码或者其他格式的硬件实现代码，然后将代码下载到 DSP 或者 FPGA 上进行实际的硬件运算。

2. FPGA 开发工具（如 Xilinx 的 Vivado）在 DSP 硬件实现中的应用

Xilinx 的 Vivado 是一款功能非常强大的 FPGA 开发工具，在 DSP 硬件实现上运用广泛。我们设计的基于 FPGA 的实时图像处理系统是采用 Vivado 的 HDL 编辑器编写的，用 Verilog HDL 语言实现了图像边缘检测算法，采用 Sobel 算法来实现边缘检测。

在编写代码过程中，通过定义模块、信号及进程，描述整个图像数据的输入、处理、输出的过程；编写完毕后，在 Vivado 中进行代码综合，将综合后的文件转换成网表文件，并且分析代码的时序与面积信息；然后通过查看综合报告来确定代码需要怎样修改，如增加流水线，加快算法速度；或者优化存储结构以降低 FPGA 占用资源。

接下来就是布局布线了，将网表中的逻辑电路映射到具体的 FPGA 器件上，如 LUT、FF、BRAM 等。然后由 Vivado 自动将 FPGA 物理布局和布线资源映射到逻辑电路，给对应的 FPGA 分配资源并优化布线，以使电路可以在 FPGA 上跑通，达到既定的时序要求。

最后，将生成的比特流文件通过 JTAG 下载线下载到 FPGA 开发板上，接入摄像头作为图像输入端、接入显示器作为输出端，就可以对捕获的图像边缘进行处理了，且在显示器上可以看到处理的结果。在真正将硬件搭建出来后，就可以看到算法是否实现了硬件加速的效果，还可根据实际运行结果对硬件进行优化改进。

3.5.4 DSP设计优化技术和EDA工具支持

1. 算法优化技术

1）快速算法

DSP在设计上大多使用快速算法提高运算速度，如快速傅里叶变换（FFT）就是一种将DFT的运算量级从$O(N^2)$降低到$O(N\log N)$的算法。利用DFT的计算是将大尺度运算转换成许多小尺度运算后再进行处理，并利用旋转因子共轭对称、周期等性质大大减少计算次数，有利于大规模信号频谱的快速获取。

EDA工具中的MATLAB内置有FFT函数，设计者可以直接调用MATLAB中的FFT函数来完成变换操作，还可选用用户自定义设置FFT点数和FFT算法实现方式。此外，一些HLD代码生成工具能将FFT算法转变为满足需求的高效硬件实现代码，在DSP芯片或FPGA中实现频谱分析功能。

2）数据量化与压缩

量化和压缩是一种常用的算法优化方法，DSP系统往往存在资源上的限制。数字信号数据过多时无法给每个数据都分配足够的存储空间，并且如果给定每个数据占据相同的资源，则总的运算时间和计算量会过大，因此要对数据进行量化，如用浮点数代替定点数，量化以后的数据既节省了数据存储的空间又降低了运算的难度；此外，还应该运用各种各样的压缩算法来达到节省传输与存储资源的目的，如将音频信息用MP3压缩，将图像信息用JPEG压缩等。

EDA工具可以提供一些量化、压缩等库函数，在算法设计时可以让设计师选择使用，然后再针对硬件的特点对量化、压缩后的数据处理流程进行优化。

2. 硬件优化技术

1）流水线优化

采用流水线技术是提高DSP硬件系统性能的一种重要方式。它是将上述复杂的过程分成几个简单的小过程，每个小过程都在不同的时钟周期同时进行，那么一个时钟周期就可以完成几个小的操作。例如，在一颗DSP芯片里，最常用的乘法累加运算就可以拆分为取数（从存储器获取运算因子）、乘法（完成两运算因子相乘）、累加（将乘法得到的结果累加到累加器里面）3个阶段，采用流水线的方式运行，在每个时钟周期都可以启动一个新的乘法累加操作。

采用EDA工具时，在设计者的硬件描述代码中可以通过增加流水线寄存器等方法进行流水线优化。利用工具依据综合约束条件以及目标硬件的特点对流水线结构本身进行优化，可保证流水线的各个阶段能够较好地配合工作，尽量避免流水线堵塞或者相关问题的发生。

2）资源复用

受硬件资源有限的影响，采用资源复用技术可以提升资源使用率、节约硬件投入，如在DSP系统中，有多种算法的实现都会用到乘法操作，那么就可建立一个通用乘法器，通过输出相应的控制信号，并在不同时钟周期给各个算法模块提供乘法运算服务，以此达

到资源共享的目的。

EDA 工具利用 HDL 中的条件语句、时序控制语句实现了对资源复用设计思想的支持，在综合、布局布线过程中，EDA 工具会针对资源复用结构进行相应的优化，在保障信号传输时序、可靠性的前提下使硬件资源最大限度地发挥效用。

3.5.5 小结

本节讲述了 DSP 的基本知识，介绍了它的特点、常用的设计方法，并且结合了 EDA 典型工具来简要介绍了一个完整的 DSP 设计过程；在此基础上进行了相关的优化技术和未来发展趋势的介绍；希望学生可以充分认识到 DSP 技术在现代电子设计中的重要作用，掌握相关的知识，为以后相关方面的学习和工作打下良好的基础。