电子科技大学示范性微电子学院

实验课程及实验室安全预习报告

（实验）课程名称 IC 综合实验 2

学号：2022340101012

姓名：李卓霖

实验地点：清水河校区国际创新中心 B523

实验日期：2024.11-2024.12

一、实验目的与要求

背景：

学院在我们两年的学习中，为这门专业最核心的课程—— IC 综合实验 II 打下了许多基础。我们依次学习了许多课程，在**数字逻辑**这门基础课上，我们学习了基本的数制知识、CMOS 逻辑门、基本逻辑代数、常用的逻辑功能块（MUX、DEMUX、译码器、比较器、存储器）、常用的时序功能块（DFF、计数器、分频器、状态机），最终能够利用多种方法构建序列发生器。在 **EDA** 这门课上，我们把数字逻辑中学到的重要元件用硬件描述语言来重新学习，核心是理解**代码的本质是电路的描述**。在**微处理器系统结构与嵌入式系统设计**这门课上。我们从处理器的视角看待电路，站在系统的高度审视电路，从 MCU 出发，学习典型的架构（冯诺伊曼结构、哈佛结构）等，学习指令集（ARM 指令集），学习嵌入式设计流程与方法，理解软硬件协同设计方法学，最终利用内核为 CortexM0 的开发板完成了许多实验。在**数字集成电路原理**这门课中，我们这一次站在集成电路的视角，定量的评估许多结构的 PPA(Power、 Performance、 Area)。这些都为我们最终的 IC 实验打下了坚实的基础。

实验目的：

1. 巩固并应用前期的基础课程
2. 熟悉数字 IC 标准设计全流程
3. 完成一个具体项目的设计

实验要求：

本次 IC2 实验课程（数字方向）的设计内容为一个已知结构的 AI 加速器的 ASIC 全流程设计并通过评估和流片。已知结构的 AI 加速器具有 8bit 输入和 8bit 输出，可以工作在至少 50Mhz 的时钟下，能够完成给定卷积神经网络的运算。总体的面积要求为在不带 I/O 的情况下通过 DC 综合得到的网表面积为 24×10^4 um^2，版图的面积最大限制为 900um * 900um，可以使用的最大的引脚数量为 28 个（含 4 到 6 个电源引脚）。

二、实验相关原理预习

2.1 前端设计

前端设计主要任务是完成架构设计、RTL 代码与前仿。

2.1.1 架构设计

针对“**如何算的快**”这个问题，我们提出：关键模块非流水线设计，内部 2 分频或 3 分频，片外高速输入时保证片内有足够的运算时间。注意到此时每次卷积（Converlution）运算时 27 个乘法器同时工作，并行计算。

依次思路，架构图见图 1.

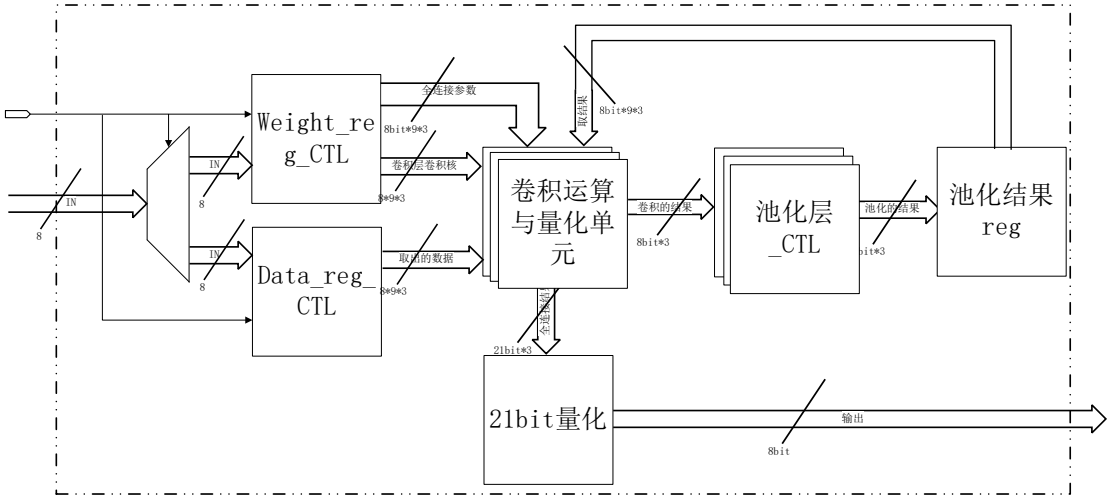

图 1 并行架构设计

这个设计看似内部运算时间充足，但运算时关键路径为单次乘加（Multiply-Accumulate）运算，而外部的 IO 有频率限制（50 M），经 DC 验证，在本项目的 SMIC .18 工艺下，单次 Mac 运算时间最快约 3 ns, 这并非是我们设计的瓶颈。

针对“**如何面积小**”这个问题，我们小组发现：乘法器是面积占用的一大主要来源，提出全设计乘法器数量仅 9 给这一思路。具体细节是：3 周期完成一个包含 3 次乘加（Multiply-Accumulate）运算的卷积（Converlution）运算。复用该模块使得每周期完成一次 Mac 运算，卷积层与全连接层再复用卷积模块。仅需要 9 给乘法器，由于可以流水输入，完全不停止，在 100 周期的数据处理下乘法器的利用率达到了 91.5%。乘法器的极高利用率是面积小的必要条件。

仍然针对**“如何面积小”**这个问题，我们小组在寄存器上有一个极其创新的优化。我们利用 python 模拟得到实际只需要 11 * 4 的存储空间，这是一个极为重大的发现。

我们的设计是在逻辑空间上进行的，我们的数据是在实际空间上存储的，它们之间见图 2.

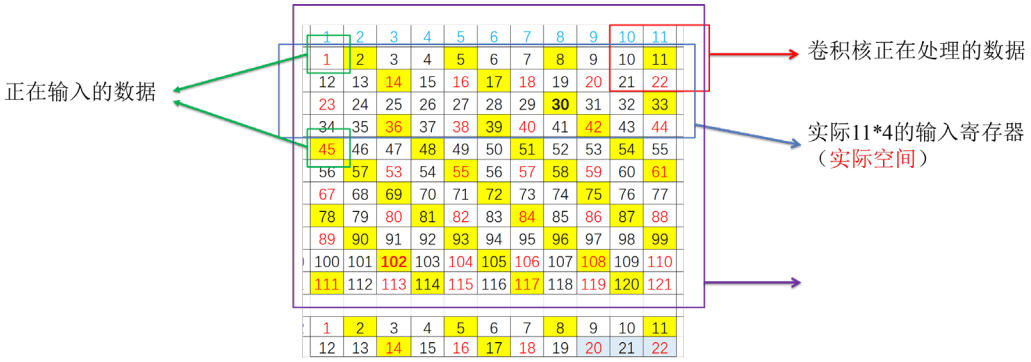

图 2 实际空间与逻辑空间

针对“**怎么分频、分频是否可行**”这个问题，我们小组利用 python 模拟得到细致的结果，详细的内容在“设计与仿真报告”中说明。模拟结果如图 3 所示。

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	
2	12	13	14	15	16	17	18	19	20	21	22	
3	23	24	25	26	27	28	29	*30*	31	32	33	
4	34	35	36	37	38	39	40	41	42	43	44	
5	45	46	47	48	49	50	51	52	53	54	55	
6	56	57	53	54	55	56	57	58	59	60	61	
7	67	68	69	70	71	72	73	74	75	76	77	
8	78	79	80	81	82	83	84	85	86	87	88	
9	89	90	91	92	93	94	95	96	97	98	99	
10	100	101	102	103	104	105	106	107	108	109	110	
11	111	112	113	114	115	116	117	118	119	120	121	
12												

1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	**30**	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	53	54	55	56	57	58	59	60	61
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	**102**	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22

图 3 双周期卷积周期功能与流水设计（上）

三周期卷积周期功能与流水设计（下）

根据诸多思考，我们最终得出的架构如图 4 所示。

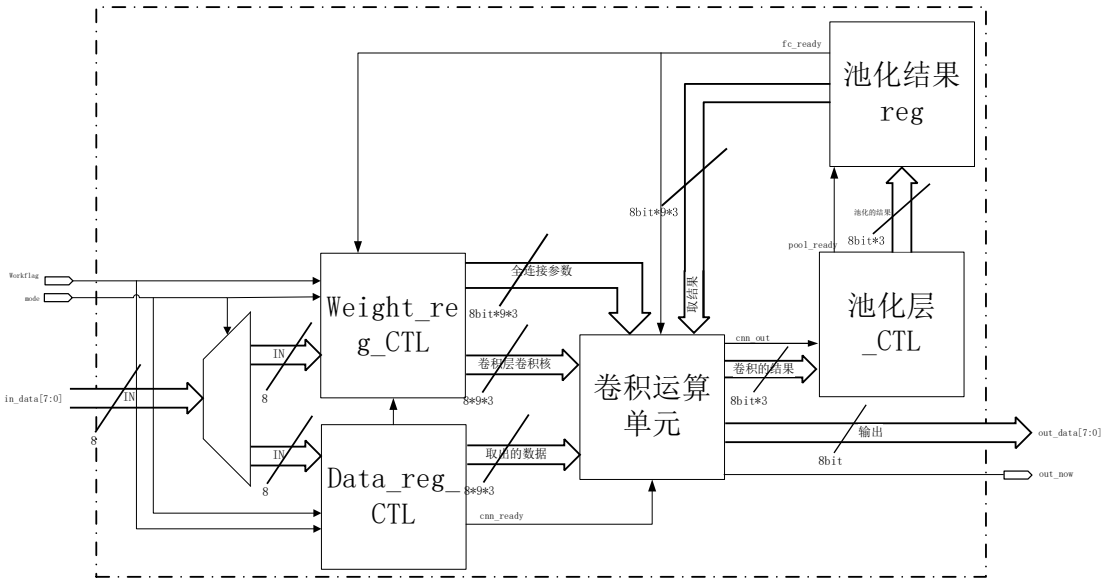

图 4 串行架构设计

2.1.2 RTL 代码设计

这部分严格按照图 4 架构进行，利用 Verilog 硬件描述语言完成设计，详细的信息请参考“设计与仿真报告”。

2.1.3 前仿

编写好代码后，我们需要对电路功能进行验证，这时候就用到 **Test bench** 以及 **VCS+DVE** 这两个软件了。下面是我对这两个软件预习后的简单认识：

VCS 是 Synopsys 提供的一款综合性硬件仿真工具，主要用于 RTL 代码的**功能验证**。它支持 Verilog 和 SystemVerilog 等主流硬件描述语言，我们本次接触的是 Verilog，能够完成从编译、仿真到调试的全过程。它具有比 Modelsim 明显突出的优势，如先进的编译优化技术，超快的仿真速度，支持**形式验证**和功耗分析等等，他还支持各种**验证方法**，如基于断言的验证和基于约束的随机验证。

DVE 是 VCS 的图形化用户界面工具，专门用于查看和分析仿真结果。设计者可以通过它直观地观察信号波形、设置断点、监控变量等，从而快速定位问题。它提供清晰的波形浏览界面，可对**仿真过程进行深入调试**；支持在仿真运行时进行实时**交互**，如动态修改信号值、观察模块行为等许多功能。

这两个工具配合使用，**VCS 负责高效仿真**，**DVE 负责便捷调试**，能够显著提升数字电路设计的验证效率和可靠性。

2.2 逻辑综合

逻辑综合是将设计的 RTL（寄存器传输级）代码通过特定算法和优化技术，转换为工艺相关的门级网表的过程，这是从功能设计到物理实现的重要桥梁。

我了解到，逻辑综合的核心流程包括读取设计、应用约束、综合优化和生成网表。首先，DC 通过 HDL 文件读取电路的**功能描述**，然后根据设计者提供的时序、面积和功耗等约束，对电路进行优化映射，将其映射到**标准单元库中的具体门电路**。最终生成的**门级网表**不仅能够反映设计功能，还可以用作后续物理实现的输入。

我还发现，DC 具有强大的**优化能力**。比如，它能够根据目标工艺节点的特性，平衡性能和资源使用，生成符合设计要求的高效电路。这让我意识到逻辑综合在整个 IC 设计流程中的重要性，它不但决定了设计能否满足功能和性能需求，还直接影响后续的时序和功耗优化。

2.3 门仿

门级仿真是验证 RTL 设计综合后的门级网表在特定工艺库条件下功能、时序和功耗是否满足设计要求的关键步骤。与 RTL 仿真不同，门级仿真使用的是

包含具体逻辑门和工艺延迟信息的门级网表，因此更加接近电路实际运行的情况。

门级仿真的主要步骤：

1.准备环境：

获取综合工具（如 Design Compiler）生成的门级网表。

工艺库文件（Liberty 格式），其中包含标准单元的时序和功耗信息。

仿真激励文件（Testbench），通常可以复用 RTL 仿真中的激励。

2.功能仿真：

在不考虑时序的情况下运行门级网表，验证其功能是否与 RTL 仿真一致。

通过波形对比和输出检查，确认综合过程中没有引入功能错误。

3.时序仿真（包含 SDF）：

加载综合或布局布线后生成的 SDF（Standard Delay Format）文件，注入门级网表。运行仿真，检查时序是否满足设计约束，如 setup 和 hold 时间。

4.结果验证：

对比门级仿真和 RTL 仿真的输出，确保两者功能一致。

分析**时序波形**，排查可能的时序问题，如信号竞争、毛刺等。

通过预习，我了解到门级仿真比 RTL 仿真**更贴近实际电路行为**，但其运行速度较慢，因为需要考虑逻辑门延迟和信号传播时间等因素。

2.4 ICC 布局布线

布局布线是数字集成电路设计中将门级网表转化为物理电路版图的重要步骤。这个过程包括模块的布局、信号的布线以及各种设计约束的优化。

布局布线主要分为两个阶段：布局和布线。**布局阶段**主要确定标准单元和模块的放置位置，目标是优化面积利用率并确保时钟和信号延迟满足设计要求。**布线阶段**则是在确定的位置之间连接信号路径，同时考虑电阻、电容和其他寄生效应对时序的影响。在整个过程中，工具会不断调整布局和布线以平衡面积、功耗和性能。

ICC **具大的优势**在于能够处理非常复杂的设计约束。例如，它可以优化多电压域设计，平衡不同区域的功耗分布；还可以结合时序分析工具确保整个芯片的时序收敛。此外，布局布线工具还能对设计进行电气检查，发现可能的信号完

整性问题，如串扰和电迁移。

2.5 后仿

后仿是数字集成电路设计流程中一个重要的验证环节，主要用于验证版图生成后的设计是否与原始功能描述一致，并检查布局布线是否引入了时序或信号完整性方面的问题。后仿是在**版图完成后**进行的，所使用的设计文件包括门级网表（Gate-Level Netlist）、SDF（Standard Delay Format）文件、提取的寄生参数（RC parasitics）和工艺库文件。后仿的目标是通过加载完整的物理信息（如延迟、寄生效应等）进行仿真，确保设计在真实工艺条件下的功能和性能没有问题。

2.6 物理验证

物理验证是芯片设计的最后一环，旨在确保设计的版图符合工艺规则（DRC）、版图与网表一致（LVS）、并进行寄生参数提取（PEX），为后续流片提供可靠保障。

物理验证的主要内容

1.设计规则检查（DRC，Design Rule Check）

检查版图是否满足制造工艺的设计规则，如最小间距、宽度和过孔密度等。这是为了确保版图可以被实际制造工艺正确实现。

2.版图与网表一致性检查（LVS，Layout Versus Schematic）

比较版图中提取的网表与设计输入的门级网表，验证两者是否一致。主要检查功能是否在版图实现中被正确保留，确保没有连接错误或漏连。

3.寄生参数提取（PEX，Parasitic Extraction）

提取版图中的寄生电容、电阻等物理效应参数，用于后续的后仿和时序分析。这些参数会影响信号延迟和功耗，是物理设计中不可忽视的一部分。

通过查阅资料，我了解到物理验证工具（如 Calibre 或 Synopsys 的 IC Validator）能够自动化执行 DRC 和 LVS 检查，并生成详尽的错误报告。在复杂的设计中，物理验证是保障流片成功的重要环节，任何小问题都可能导致芯片无法正常工作。

三、实验内容与步骤预习纲要

3.1 需求分析与规格定义

明确我们这次设计的需求，已知结构的 AI 加速器具有 8bit 输入和 8bit 输出，

可以工作在至少 50Mhz 的时钟下，能够完成给定卷积神经网络的运算。总体的面积要求为在不带 I/O 的情况下通过 DC 综合得到的网表面积为，版图的面积最大限制为 900um * 900um，可以使用的最大的引脚数量为 28 个（含 4 到 6 个电源引脚）。本次设计的**结构框图**如图 5 所示。

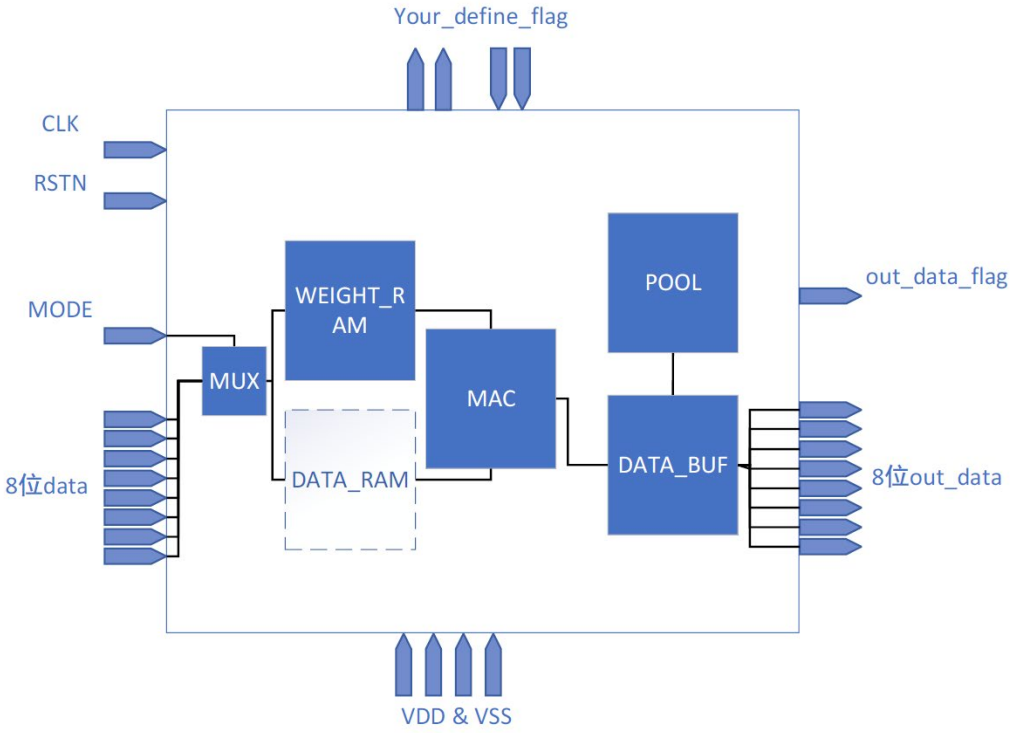

图 5 结构框图

给定的卷积核的结构包含卷积层、池化层、全连接层各一层，共计 3 层。各层的详细介绍如下。

卷积层：在 11×11 的矩阵数据外添加 Padding 后，对对应的矩阵进行卷积操作，其中，卷积核大小为 3x3，步长为 2，共有 3 个卷积核；

池化层：对输出的数据进行最大池化，其中池化核大小为 2x2，步长为 2；

全连接层：全连接层对应的权重矩阵大小为 3x3x3，与池化输出一一对应，需要完成一次乘加操作。

芯片的输入数据为 1x1 的矩阵，是位宽为 8bit 的有符号数补码，按照卷积核、全连接权重、100 组 11×11 的矩阵的数据依次输入。

芯片的输出数据为 1x1 的矩阵，位宽为 8bit，输出的为全连接的结果通过量化所得到的内容。

由于在数据处理过程中得到的乘积数据为 16bit，加和后需要用 20bit（全连

接 21bit）的数据记录才能保证数据不发生溢出，而后续进行池化、全连接和输出所需要的数据均为 8bit，因此需要一种量化规则来进行 20bit 或 21bit 的到 8bit 的量化处理。给定的卷积核的量化规则为：在保证符号正确的情况下拓展到 20 位来表示，然后将低 8 位抹除，剩余的 12 位若超过 int8 的表示范围，则取 int8 范围的最大值，反之，取这 12 位的低 8 位。由于数据均由补码表示，因此所得的结果只需要将最高位符号位向前扩展即可得到扩展数据。

3.2 系统级设计

在系统级设计阶段，我们最初讨论了采用并行架构来处理计算任务，计划通过 27 个乘法器在一个周期内并行计算所有卷积运算。然而，经过分析后，我们发现最大瓶颈并非计算能力，而是输入数据的 I/O 接口速度。尽管使用 27 个乘法器可以在一个周期内完成所有运算，但 I/O 接口的带宽限制使得这一方案的速度难以满足需求。

因此，我们对架构进行了**调整**，转而采用串行处理方式。通过在三个周期内完成一次卷积计算，我们用 9 个乘法器复用多次计算，既能够减少硬件资源的消耗，又能在每个周期内完成计算任务，从而在满足计算需求的同时，充分利用时间来换取空间的优势。这一修改优化了硬件设计，并有效提高了系统的整体性能。

3.3 RTL 设计

在 RTL 代码设计阶段，我们采用了多种设计方法，以便实现高效的多乘法器架构。为了满足系统性能要求，我们探索了多个设计思路，并对比了不同方案的优缺点。在这一过程中，我们不仅注重硬件资源的最优化，还确保了设计的可扩展性和可维护性。

具体来说，我们使用了标准化的 RTL 编写风格，确保了代码的规范性和可读性。这种编写风格帮助我们提高了代码的重用性，使得在不同模块之间的接口设计更加清晰。此外，我们还通过合理的模块化设计，将每个乘法器的功能划分成独立的模块，从而简化了调试和后期的修改。

在设计过程中，我们特别关注时序和资源的优化，确保多乘法器的并行计算能够高效运行，同时减少了功耗和面积的浪费。通过充分利用复用机制和流水线技术，我们确保了乘法器在不同周期内能够高效地工作，从而达到了设计预期的性能目标。

通过这些精心设计和优化，我们成功完成了 RTL 代码的编写，并且保证了设计的功能完整性和时序可靠性。该设计不仅满足了系统的性能需求，而且为后续的验证和实现奠定了坚实的基础。

3.4 综合

在综合（Synthesis）阶段，我们使用了 Synopsys Design Compiler（DC）工具，将编写的 RTL 代码转化为门级网表。在这个过程中，综合工具根据设计规格和约束，自动生成了一个优化过的门级实现，以满足目标频率、面积和功耗等设计目标。

首先是通过设置时序约束和面积约束，确保设计在时钟频率和功耗方面能够符合要求。时序约束是综合过程中非常关键的一部分，它定义了各个信号之间的时序关系，并确保设计在时钟周期内能够正确地完成数据传输。为了确保设计的正确性，我们仔细检查了时钟路径、组合路径以及数据路径中的潜在时序问题。

在综合过程中，还要对设计中的标准单元（如逻辑门、触发器等）进行了优化，以最大化资源利用率，并尽量减少功耗。通过 DC 工具的优化算法，我们对面积和功耗进行了平衡，力求在有限的资源下实现最佳的性能。

最后，通过后仿真验证，我们确保综合后的门级网表与 RTL 设计在功能和时序上保持一致。

3.5 布局与布线

布局与布线阶段的目标是根据设计规格（如时序、面积、功耗等）将逻辑单元（例如门、寄存器、乘法器等）放置到芯片的物理区域，并确定它们之间的连接。整个过程可以分为几个主要步骤，主要包括布局规划（Floorplan）、布局（Placement）、时钟树综合（CTS）和布线（Routing）等。

布局是指将电路的各个模块（例如逻辑门、寄存器、运算单元等）放置到芯片的特定位置上。这个过程的主要目标是确保电路的时序性能，同时优化面积和功耗。**布线**是将已放置好的模块通过导线连接起来的过程。布线的目标是保证所有信号的正确传输，同时优化信号路径的延迟、信号完整性和功耗。

布局与布线阶段，**时钟树的设计**是一个至关重要的步骤，它直接影响整个设计的性能和可靠性。时钟树的任务是确保时钟信号能够高效、稳定地传输到设计中的每个模块，避免时钟延迟和时序问题，从而保证设计在高频下能够正常工作。

就时钟树而言：

验证在进行布局与布线时，时钟树的设计通常包括以下几个关键步骤：

时钟源的选择与分配：设计者首先要选择时钟源（通常是外部输入时钟或内部时钟生成模块）并将其分配到设计的不同部分。时钟源的选择和分配必须满足时钟传输的稳定性和一致性。

时钟树的构建：使用布局与布线工具自动生成时钟树结构。时钟树的构建要确保时钟信号能够以最短的路径传播到所有模块，通常采用平衡的时钟树结构，使得时钟信号能够在所有时钟域内同步传递。

时钟树优化：在时钟树构建后，工具会进行时钟树优化，平衡各条时钟路径的长度和延迟，避免出现时钟信号的偏移和时序问题。时钟树优化还包括插入适当的缓冲器和反向器，以减少时钟信号的延迟和避免信号衰减。

时钟树的验证：时钟树设计完成后，使用**静态时序分析工具**（如 PrimeTime）对时钟树进行验证，不过我们这次实验是没有采用这一步验证的，确保时钟信号能够在每个时钟周期内正确到达设计的各个部分，同时避免出现时钟偏差和延迟过大的问题。

在完成初步布局和布线后，进行**优化和验证**是必要的步骤。优化包括以下内容：**时序优化**：通过进一步调整布局和布线，确保满足时钟频率和时序约束。**功耗优化**：减少由于布局和布线引起的功耗，降低电源和接地网络的功耗。**DRC/LVS 检查**：进行设计规则检查和布局与原理图的一致性检查，确保设计不会产生制造过程中的违例。

3.6 后仿

后仿是指在完成布局与布线（P&R）之后，对设计进行仿真验证的过程。后仿的目的是验证布局与布线后的设计是否仍然符合功能和时序要求，尤其是在物理层面引入了额外的时序延迟、信号反射和其他可能影响设计正确性的因素时。

3.7 物理验证

物理验证是指在布局与布线完成后，检查设计是否符合制造规则和工艺限制的过程。物理验证的主要目的是确保设计不会在制造过程中产生无法实现的结构或违背工艺规范的设计，避免因物理设计问题而导致芯片无法制造或工作不稳定。

3.8 制造与生产

将设计交给半导体制造厂进行生产。根据设计数据（如 GDSII 文件），生成制造所需的光掩膜文件，并提交给代工厂进行晶圆制造。制造过程涉及光刻、蚀刻、沉积等一系列工艺，最终得到完成的芯片。

四、使用的实验仪器（设备、元器件）

1. 计算机工作站；
2. EDA 工具软件：

Synopsys VCS 和 DVE：用于功能仿真和波形分析。

Design Compiler (DC)：进行逻辑综合，将 RTL 转换为门级网表。

Formality : 进行一致性验证。

IC Compiler (ICC)：用于布局布线，生成设计版图。

Calibre

3. 实验室服务器；

五、实验室安全操作事项

1. 使用计算机和其他电气设备时，确保电源连接稳固，避免插拔时触电。
2. 定期保存设计文件，防止因断电或设备故障丢失实验数据。
3. 避免私自拷贝或传播实验室专用工具或资源，以免违反实验室管理规定。
4. 使用实验室计算机时，不访问不安全网站或随意安装第三方软件，避免病毒入侵。
5. 实验结束后清理桌面，关机并关闭设备电源，确保教室整洁和设备安全。
6. 遇到设备故障或异常时，立即报告，不擅自操作或维修。及时询问老师。

六、思考题（至少 3 道）

6.1 逻辑综合中，如何平衡电路的面积、时序和功耗？这些目标之间是否存在冲突？

面积优化：减少使用的逻辑单元数量和布线资源，可以降低制造成本，但可能会增加时延或功耗。

时序优化：通过插入缓冲器、增加驱动能力或优化路径延迟来确保时序收敛，

但可能导致面积和功耗增加。

功耗优化：通过减小切换频率、降低电压或减少信号切换活动可以降低功耗，但可能影响电路性能或增加面积。

这些目标之间确实存在冲突。例如，增加驱动能力以优化时序通常会增加功耗；减小面积可能导致布线拥塞，从而影响时序。为平衡这些目标，设计者通常设置综合约束（如面积上限、时序限制）并使用工具（如 Design Compiler）的多目标优化功能。权衡的关键在于根据设计需求选择优先级，并结合 PPA 综合优化技术。

6.2 门级仿真与 RTL 仿真相比，增加了哪些真实物理效应？

增加的物理效应：

1.逻辑门延迟：门级仿真考虑了实际工艺库中逻辑门的传播延迟。

2.布线延迟：门级仿真加入了布线寄生参数（RC 延迟），更接近实际电路行为。

3.信号完整性问题：如竞争、毛刺和信号反射，在 RTL 仿真中无法体现。

6.3 门级仿真的运行速度为什么会比 RTL 仿真慢？

更复杂的模型：RTL 仿真直接执行逻辑操作，而门级仿真需要模拟门延迟和信号传播，模型更复杂。

精确的时序计算：门级仿真需要加载 SDF 文件并对路径延迟进行精确计算，占用额外的计算资源。

寄生参数的影响：门级仿真加入寄生参数，导致信号传播的计算量增加。

总之，门级仿真为了更接近真实硬件，牺牲了仿真速度，但提供了更高的验证准确性

6.4 在布局布线中，时钟树综合（CTS）的作用是什么？如何确保时钟信号的延迟和偏移（Skew）满足设计需求？

时钟树综合（CTS）的作用：

CTS 是布局布线中的一个重要步骤，用于构建芯片的时钟分配网络，确保时钟信号能够在芯片的所有触发器间高效分布。主要目标包括：

①最小化时钟偏移（Skew）：减少不同触发器之间的时钟到达时间差异，确

保时钟同步。

②控制时钟延迟：减少全局时钟的传播延迟，提升时序性能。

③优化功耗：通过合理的树形结构和缓冲器分布，降低时钟网络的功耗。

如何确保时钟延迟和偏移满足设计需求：

①合理选择 CTS 算法：使用 EDA 工具内置的 H-tree 或 X-tree 算法，根据设计规模和时钟负载分布优化时钟树。

②设置时序约束：在工具中定义时钟延迟和偏移的目标值（clock uncertainty 和 skew constraints）。

③插入缓冲器和平衡负载：在时钟树中插入缓冲器，调整不同分支的负载，减少时钟偏移和延迟。

④仿真验证：在时钟树综合完成后，通过时序仿真和物理验证（如 RC 提取）检查实际的时钟偏移和延迟是否满足要求。

七、预习遇到的问题：

1.逻辑综合与物理设计的衔接：虽然了解了逻辑综合的**基本流程**，但对如何在综合后生成的网表上添加物理设计约束（如面积限制、功耗优化）还不够清楚。

2.**寄生效应的具体影响**：在后仿和物理验证过程中，寄生参数（如电容、电阻）对时序的影响如何量化？如何通过调整设计来减小寄生效应的影响？

3.EDA 工具的操作细节：对工具（如 ICC 和 Calibre）的具体使用流程还不熟悉，如**如何设置约束文件**和**分析验证结果**。