上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

高性能 16 点 FFT 芯片设计报告

2020 年 5 月

目录

1. 课程设计规范简介

1.1 功能描述

本次我们组选择的课程设计题目是高能效的快速傅里叶变换(以下简称 FFT)电路。该电路实现 16 点基 4 FFT 算法，具体计算的蝶形运算原理如图 1 所示[1]。

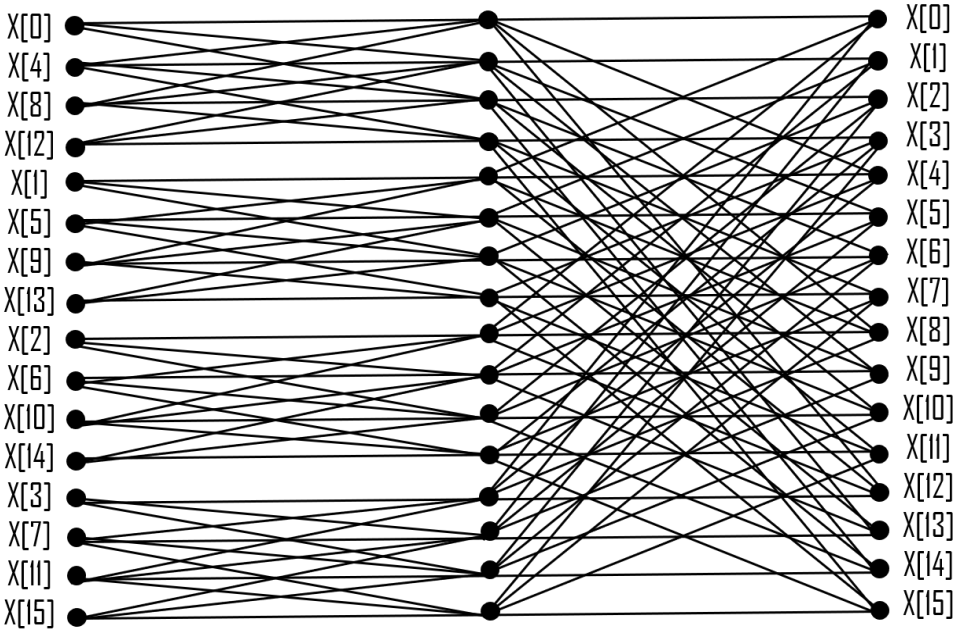

图 1 16 点基 4FFT 原理

其中每个输入值的实部与虚部均为 17 bit，分为 1 bit 符号位，8 bit 整数位和 8 bit 小数位；而采用的旋转因子的实部与虚部均为 8 bit，包含 1 bit 符号位和 7 bit 数据位。输入数据和旋转因子的数据表示方式如图 2 所示。

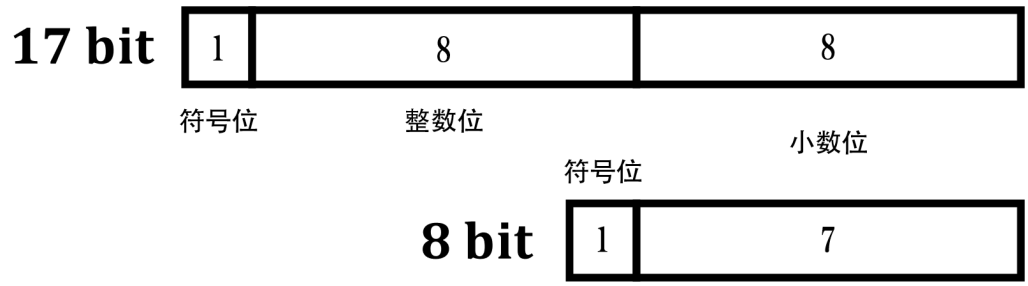

图 2 数据的表示方式

最终的功能要求是该 FFT 电路能正确完成 FFT 算法。

1.2 性能描述

本次课程设计采取的芯片设计工艺为 0.18 μm 工艺。芯片的性能评价指标包括：

表 3 性能评价指标综述

评价指标	具体数值
完成一次 16 点 FFT 运算的能耗	5.25118208×10^{-6} mJ
单位面积功耗时间 FFT 操作次数	4.1987688×10^{4} 次/(mm^2·mW·s)
工作频率	135.135135 MHz
面积开销	4.695300 mm^2

1.2.1 单次 FFT 能耗计算

由 2.1 节和 5.2 节可知，完成一次 FFT 运算的能耗

$$\frac{\text{每秒能耗}}{\text{每秒 16 点 FFT 运算次数}}=\frac{44.3512\ mW/s}{1\ s/16\times7.4\ ns}=5.251182\ \times10^{-6}\ mJ$$

1.2.2 单位面积功耗时间操作次数计算

由 2.1 节和 5.2 节可知，单位面积功耗时间 FFT 操作次数

$$\frac{\text{每秒 16 点 FFT 运算次数}}{\text{面积}\times\text{功耗}\times\text{时间}}=\frac{1\ s/16\times7.4\ ns}{4.535456\ mm^2\times44.3512\ mW\times1\ s}$$

$$=4.1987.688\ \times10^{4}\ \text{次}/(mm^2\cdot mW\cdot s)$$

2. 电路性能分析与电路结构设计

2.1 性能估算

2.1.1 面积开销

我们的芯片共有 90 个管脚，其中长为

$$76\ \mu m \times 23 + 210\ \mu m \times 2 = 2168\ \mu m = 2.168\ mm$$

宽为

$$76\ \mu m \times 22 + 210\ \mu m \times 2 = 2092\ \mu m = 2.092\ mm$$

总面积为

$$2.168\ mm \times 2.092\ mm = 4.535456\ mm^2$$

2.1.3 运算性能

芯片每秒可完成

$$1\ s / 16 \times 7.4\ ns = 8.445946\ \times 10^6 \text{ (次 FFT 运算)}$$

2.1.3 吞吐率

由于我们的芯片采用串行输入输出的方式，因此每 16 个周期完成一次信号的输入，同时从输入第 13 个信号时开始进行蝶形运算，经过 8 个周期完成蝶形运算后，再经过 16 个周期输出，因此单次 FFT 运算需要的周期为

$$13 + 8 + 16 = 37 \text{ (个周期)}$$

所以单次运算的吞吐率为

$$\frac{1\ s}{37 \times 7.4\ ns} \times 16 \times (17 + 17) = 1.986851717 \times 10^9\ bps$$

而我们的芯片也采用了三级流水的方式，所以在连续运算次数足够多的情况下，单次 FFT 运算需要的周期为

$$\lim_{n\to\infty}\frac{16n+21}{n}=16\ (\text{个周期})$$

吞吐率为

$$\frac{1\ s}{16\times 7.4\ ns}\times 16\times(17+17)=4.594594595\times 10^{9}\ bps$$

2.1.4 带宽

由于我们的芯片存储器时钟频率和数据输入速度保持恒定，因此峰值带宽与平均带宽相同，为

$$135\ MHz\times\frac{34\ bit}{9}=510.51\ MB/s$$

2.1.5 工作频率

电路使用的时钟周期为 7.4 ns，因此工作频率为

$$f=\frac{1}{7.4\ ns}=135.135135\ MHz$$

2.2 硬件结构图及说明

2.2.1 电路硬件结构图

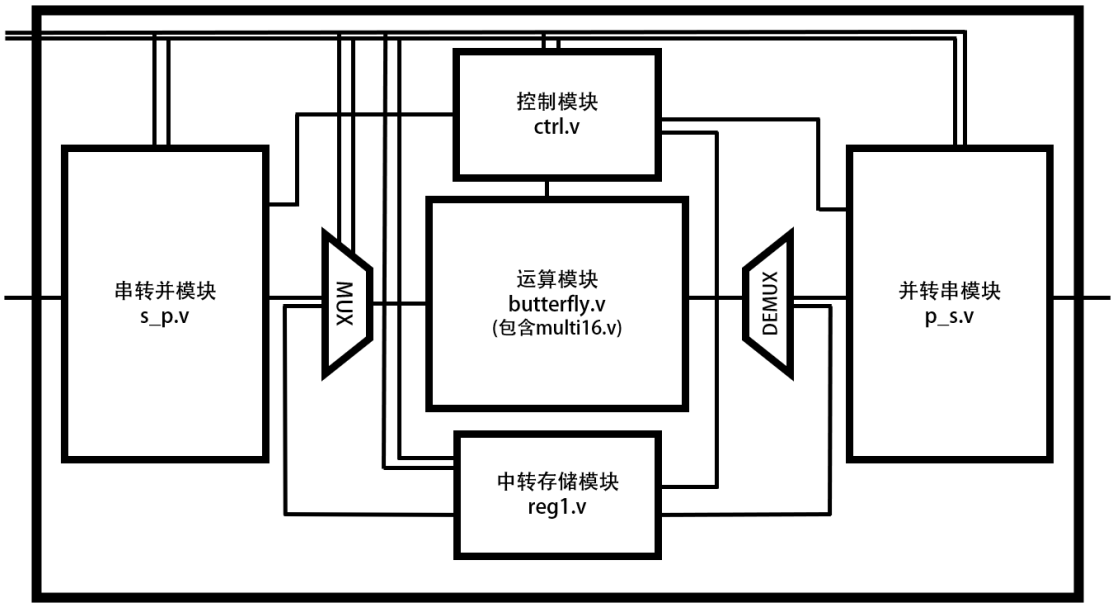

图 3 电路硬件结构图

2.2.2 说明

电路的硬件结构如图 4 所示，主要分为输入输出的两个串并转换模块，一个控制模块，一个中转存储模块，一个四输入蝶形运算模块(包括乘法器)和复选器模块。各模块的具体功能如下表所示：

表 2 各模块类型及功能描述

模块名称	文件名	模块类型	功能描述
控制模块	ctrl	时序逻辑	控制电路中各个模块的运行
串转并模块	s_p	时序逻辑	将串行输入的数据整合为并行并改变顺序
复选器	mux	时序逻辑	按照 FFT 级数选择进入运算模块的数据
中转存储模块	reg1	时序逻辑	存储并转发两级 FFT 运算间的运算数据
运算模块	butterfly	组合逻辑	进行四输入蝶形运算
乘法器	multi16	组合逻辑	计算数据与旋转因子间的乘法
并转串模块	p_s	时序逻辑	将并行输出的数据分解为串行并倒位序

3. 电路 RTL 模型与仿真验证

3.1 RTL 模型

3.1.1 电路验证方法

此 FFT 芯片的验证包括输出结果验证验证与流水线验证。输出结果验证是通过给 testbench 一些输入数据，然后通过模拟仿真电路将相应输入数据的数据流动过程一步一步显示并得到最终的输出结果，将其与理论上的正确结果对比以验证电路是否正确工作。而流水线验证只需一次输入多组数据，观察电路是否能够流水式地执行即可。

3.1.2 验证平台的搭建

在各个模块都写好的基础上，在 ModelSim SE-64 10.1c 中将所有模块的 verilog 代码通过一个 top 文件按照图 4 的关系连接，然后为其撰写 testbench 文件，从指定的文本文件中读入输入数据并按照仿真电路进行数据计算。

3.2 仿真验证结果

3.2.1 功能点的验证

本次项目中 FFT 芯片功能点的验证包括输出结果正确性验证与流水线验证。于是我们组取了三组输入作为测试向量对电路功能进行仿真。

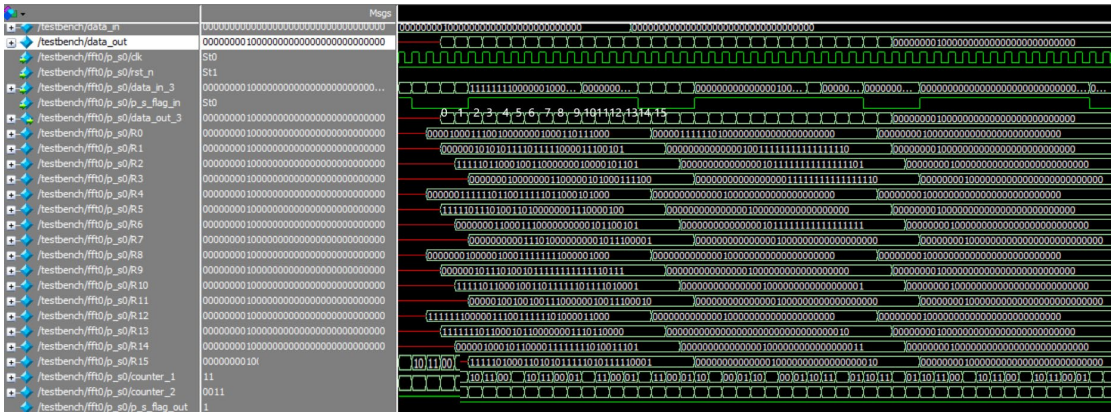

图 4 测试波形

从图中我们可以看到，p_s 模块中的数据的确按照我们预想的那样工作：每个周期进入 4 个数据，然后将数据逐个输出，并且上一组数据的输出刚好输出完全后下一个周期就开始输出下一组数的输出，衔接的很好，进行了流水式的执行。而功能验证则可通过比对得到。

表 3 第一组数据理论输出与实际输出对比图

X[k]	理论输出	实际输出
X[0]实部	0_00010010_00000000	0_00010001_11001000
X[1]实部	0_00000101_01101110	0_00000101_01011110
X[2]实部	1_11110110_00000000	1_11110110_00100110
X[3]实部	0_00000010_00001101	0_00000010_00000011
X[4]实部	0_00001000_00000000	0_00000111_11101100
X[5]实部	1_11110110_00110000	1_11110111_01001101
X[6]实部	0_00000011_00101011	0_00000011_00011100

(续前表)

X[7]实部	0_00000000_01101101	0_00000000_01110100
X[8]实部	0_00000010_00000000	0_00000010_00001000
X[9]实部	0_00000101_11100001	0_00000101_11010010
X[10]实部	1_11110110_00000000	1_11110110_00100110
X[11]实部	0_00001001_01000010	0_00001001_00100111
X[12]实部	1_11111100_00000000	1_11111100_00011100
X[13]实部	1_11111101_01111111	1_11111101_10001011
X[14]实部	0_00001000_11010100	0_00001000_10110000
X[15]实部	1_11110100_01000010	1_11110100_01101010
X[0]虚部	0_00010010_00000000	0_00010001_10111000
X[1]虚部	1_11110000_10111010	1_11110000_11100101
X[2]虚部	0_00000100_00111110	0_00000100_00101101
X[3]虚部	0_00001010_01011100	0_00001010_00111100
X[4]虚部	1_11110110_00000000	1_11110110_00101000
X[5]虚部	0_00000011_10010000	0_00000011_10000100
X[6]虚部	0_00000010_11100000	0_00000001_01100101
X[7]虚部	0_00000010_11100000	0_00000010_11100001
X[8]虚部	1_11111110_00000000	1_11111110_00001000
X[9]虚部	1_11111111_11110101	1_11111111_1111-111
X[10]虚部	1_11111011_11000001	1_11111011_11010001
X[11]虚部	0_00000100_11110011	0_00000100_11100010
X[12]虚部	1_11111010_00000000	1_11111010_00011000
X[13]虚部	0_00000011_10111111	0_00000011_10110000
X[14]虚部	1_11111110_10010101	1_11111110_10011101
X[15]虚部	1_11110101_11001111	1_11110101_11110001

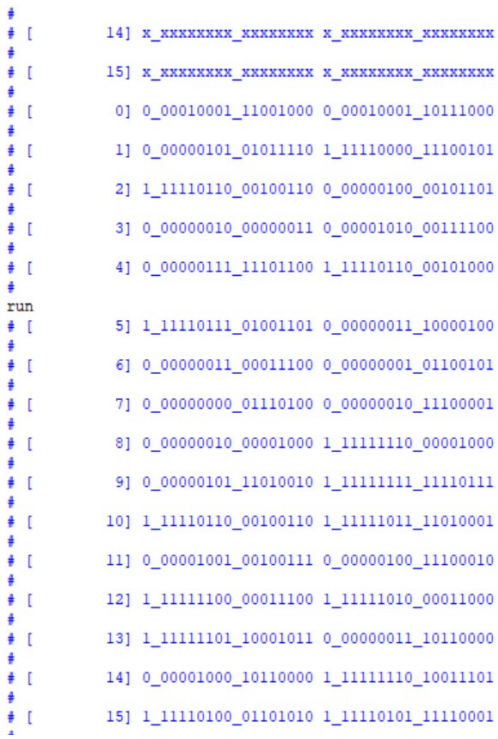

图 5 Transcript 输出

通过比较可知，第一组数据的实际输出与理论输出的实部与虚部都十分接近，在误差范围内可以认为是相等的（存在数据位位宽限制等不可避免的误差）。同样，后面的第二组与第三组数据也都经过一一验证可知实际输出与理论输出结果一致。私下我们再用其它数据测试结果也都符合要求，未出现错误。

3.2.2 测试向量

共有三组测试向量。

表 4 测试向量

第一组
00000000100000000000000000000000 00000000000000000000000000000000 00000000100000000000000000000000 00000000000000000000000000000000 00000001100000000000000000000000 00000001000000000000000010000000000 00000000100000000000000100000000000

(续前表)

00000001000000000000000001000000000
00000000100000000000000000000000000
00000001000000000000000010000000000
00000000100000000000000000000000000
00000000000000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
第二组
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
00000000100000000000000000000000000
第三组
00000000100000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000

3.2.3 仿真性能统计

从之前的功能点验证可知，我们成功实现了 16 点 FFT 算法并

得到了预期的结果与运行效果。我们成功实现每周期输入一个数据并且每周期输出一个数据，成功实现流水式运行并得到了正确的运行结果。从我们多次的测试可知输出结果始终没出现错误，性能很稳定。

4. 电路逻辑综合策略与综合结果

4.1 逻辑综合策略

4.1.1 逻辑综合流程

逻辑综合的主要目的是将 RTL 形式的硬件描述语言在约束下转换为 DC 内部统一用门级描述的电路，以 GTECH 或无映射的 ddc 形式展现。其主要流程包括指定库文件，读入设计，定义环境，设计和优化约束，编译策略、综合与优化及存储。

综合过程中，首先要指定库文件，包括工艺库和符号库等，搭建环境逻辑综合环境，随后将设计文件载入内存，并转换为 GTECH 格式。这一部分通过 read.tcl 中的指令实现。

读入设计后需要定义环境，包括工艺参数，I/O 端口属性等。详细的参数设置将在 4.1.2 节中进行描述。这一部分在 fft.con 等文件中设置。

在定义环境后，需要读入设计规则约束和优化约束，主要由 fft.con 设置。随后选择编译策略，进行编译，执行综合与优化过程，全部结果符合规范后存储网表、ddc 等设计文件。

4.1.2 参数设置

表 5 参数设置

参数	数值
工艺	SMIC 0.18um
时钟周期	7.4 ns
输出输出延迟	3.7 ns

(续前表)

参数	数值
工作电压	1.62 V
运行温度	125 ℃
面积约束	0
触发器类型	未设置
锁存器类型	未设置
互连模型	balanced_tree
引脚输入输出延迟	未设置
时钟延时	4.3 ns
不确定时钟	0.5 ns
过渡时钟	0.2 ns
端口和设计最大过渡	3 ns

4.2 逻辑综合结果

4.2.1 时序结果

表 6 时序结果

项目	时间 (ns)
clock clk (rise edge)	7.40
clock network delay (ideal)	4.30
clock uncertainty	-0.50
clk slack (MET)	0.00

时序结果如上表所示，可看到 7.4 ns 的时钟符合设计要求，换算为工作频率约为 135 MHz。

4.2.2 面积结果

表 7 面积结果

项目	数量	项目	面积 (um^2)
Ports	70	Combinational area	1483546.414358
Nets	140	Non-combinational area	153599.850292
Cells	71	Net Interconnect area	2089790.568665
References	3	Total Area	3726936.833315

4.2.3 功耗结果

表 8 功耗结果

项目	电压或功耗
Global Operating Voltage	1.62 V
Cell Internal Power	756.0848 mW
Net Switching Power	375.6310 mW
Total Dynamic Power	1.1317 W
Cell Leakage Power	28.5281 uW

4.2.4 资源使用

表 9 资源使用

项目	个数或使用率
Hierarchical Cell	85 个
Hierarchical Port	5373 个
Leaf Cell	16473 个
Buf/Inv Cell	2815 个
Total Number of Nets	18037 个

4.2.5 设计违例

电路逻辑综合结果中无设计违例。

5. 电路物理实现与结果分析

5.1 ICC 设计步骤

完成逻辑综合后，会得到电路的门级网表，位于 DC 的 output/文件夹中。将门级网表导入到 ICC 的 design/文件夹中便可以开始进行电路的物理实现。而物理实现主要分为布局规划(Floorplanning)，布局(Placement)，时钟树综合(CTS, Clock Tree Synthesis)和布线(Routing)四个阶段，整个物理实现过程可通过运行 icc.tcl 文件实现。以下对物理实现的这四个步骤进行简要描述，而最终的物理实现结果将在 5.1.2 节中进行展示。

在进入物理实现步骤前，首先需要进行数据设定(data setup)将逻辑和时序库文件(也即逻辑综合中的靶文件)、sdc 约束文件和逻辑综合得到的门级网表文件等综合数据读入 IC Compiler 中；除此之外，还需要读入 IO 等物理库文件，tf 工艺文件和描述电容和金属线单位面积电阻电容等属性的 TLU+文件等物理数据。相关的指令位于 run_data_setup.tcl 文件中。

5.1.1 布局规划

布局规划是整个物理实现的关键，后续三个阶段中都会有涉及。布局规划主要目标是形成好的布局，合理放置各模块，提高芯片面积的利用率，厘清输入输出的关系和数据传输路径，减少拥塞。

整个的布局规范分为四个步骤，首先是进行初始的布局规划，其次是尝试进行标准单元的布局，然后是在尝试布局的基础上分析拥塞，最后形成电源网络并分析时序，全部正确后布局规划结束。整个布局规划在 run_design_planning.tcl 中实现。

在初始的布局规划时，首先要定义所有用到的管脚，包括输入输出、时钟和初始信号，以及对管脚和标准单元进行供电的两组电源。并指定管脚位置等信息(位于 pad_cell_cons.tcl 和 insert_pad_filler.tcl 中国)，随后添加电源环，生成基本的布局规划。

而完成基本布局规划后，需要尝试对标准单元进行虚拟布局，主要通过命令 create_fp_placement -timing_driven -no_hierarchy_gravity 实现。由于主要是针对时序，所以选择-timing_driven 后缀。

接下来是分析拥塞，需要设计布局的策略并尝试进行布局。由于我们的设计并没有用到第三方 IP，因此这一步较为简单。我们的设计由于芯片面积较大，面积利用率设置得并不高，所以并没有遇到太多拥塞方面的问题。

而最后是形成电源网络并分析时序。我们用到的电源电压为 1.62V，在后续分析电压降后增加了电源管脚并调节了管脚位置，在将电压降调整到小于 10%后，布局规划阶段顺利完成。

5.1.2 布局

布局的主要目标是根据门级网表来决定各个标准单元的具体位置。相较于布局规划只决定各个宏单元，布局决定了各个标准单元在布局上的位置。布局主要减少预估的单元间互连线的长度(但并没有真正进行布线)，并解决拥塞的问题。

在 ICC 中，我们并不需要涉及 DFT 和 Power 的设定，因此布局部分较为简单，主要进行布局，布局优化和拥塞分析与优化三个步骤。整个布局的相关指令主要位于 run_placement.tcl 中。

布局部分主要遵循 NDR 规则(ndr.tcl)，主要考虑特殊信号线(如clock)的噪声控制等因素，将这些较为特殊的信号金属线做得更宽，间距更大，从而使其受到的影响更小。而布局主要通过命令 place_opt 来实现。

而在布局优化部分，可以通过添加后缀进行优化，常见的可优化项有面积(-area_recovery)，功率(-power)或拥塞(-congestion)。此处解决了之前布局规划中遇到的建立时间的违例。

而布局阶段的拥塞分析较布局规划部分更为准确。而此部分我们并没有遇到太大的问题，布局阶段顺利完成。

5.1.3 时钟树综合

时钟树综合的目标比较简单，主要目的是生成时钟网络。在前面两个阶段定义了时钟管脚，但从时钟到各个需要时钟的单元尚不存在金属线连接。因此，在时钟树综合阶段，主要生成时钟网络，并加入缓冲(buffer)以解决时钟信号负载和 slew time 等问题，并进行金属线的预布局，检查是否有绕线的情况。

在 ICC 中，时钟树综合主要通过命令 clock_opt 来实现。因为我们的设计中只用到了一个时钟，并且是同步设计，因此时钟树综合部分的工作较为简单，运行指令并检测报告后顺利完成。相关指令位于 run_cts.tcl 中。

5.1.4 布线

布线主要根据网表中的依赖关系将所有标准单元用金属线进行连接，并解决可能存在的所有保持时间、建立时间、最大电容、拥塞、

DRC 规则等问题，并决定金属线的层次。在完成布线后本次设计就宣告完成，但实际流片前一般还会有 DRC 的修复等工作。

在 ICC 中，布线主要通过命令 route_opt 来实现。在进行布线之前，我们的设计已经解决了所有拥塞和违例问题。在进行布线时，会将整个芯片分为很多网格，而各个格点用来布线。默认布线规则可以通过 report_preferred_routing_direction 指令查看。

而整个布线主要分为三个步骤。首先要进行全局的布线，其次是各个网络分配到具体的轨道上，然后修复前述步骤完成后产生的 DRC 违例。

全局布线首先对时钟进行布线，通过 route_group -all_clock_nets 指令完成。在布线过程中，我们在执行 verify_lvs 指令时出现了一些不明原因的报错，在执行增量优化时也出现过闪退的问题。对于前一个问题，我们查阅了相关资料，也请教了助教，认为相关报错对设计并没有实质性影响，因此忽略了相关报错；对于后者，我们发现是虚拟机存储空间和内存不够的原因，清理虚拟机后成功完成了布线工作。本次项目的计划也在布线后顺利完成。相关指令位于 run_routine.tcl 中。

物理实现最终的物理实现结果在 5.1.2 节中进行展示。

5.2 ICC 设计结果

5.2.1 芯片概貌

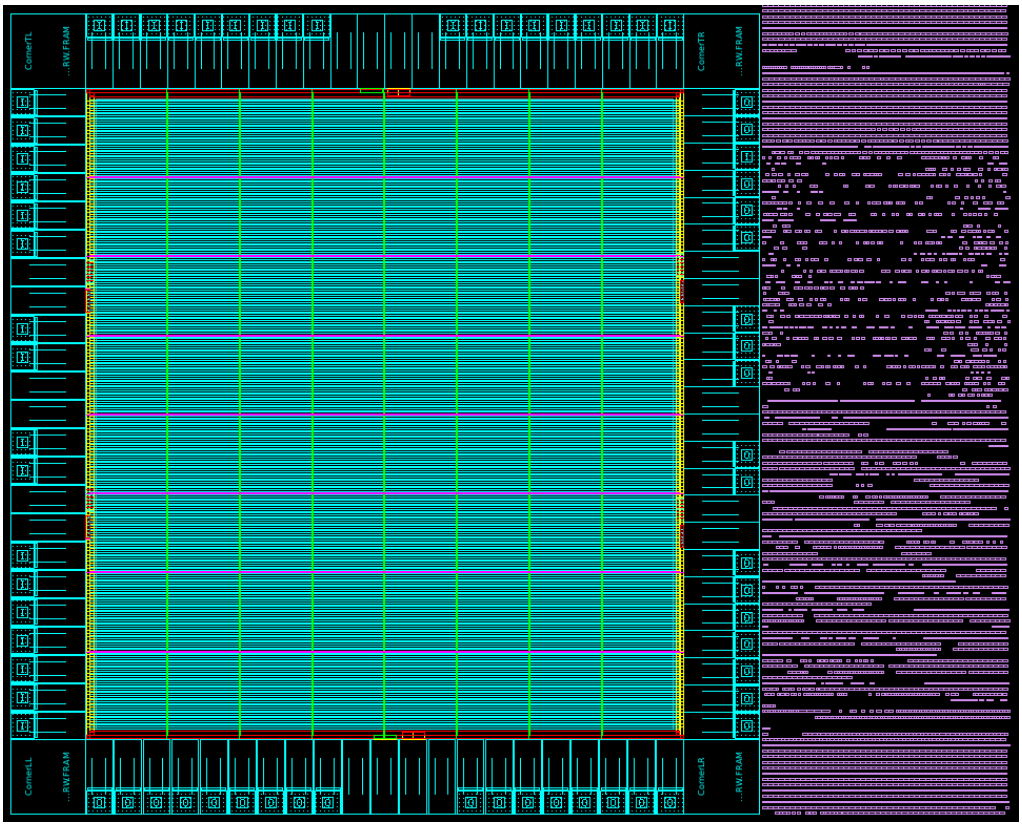

图 6 芯片概貌

芯片的概貌如图所示。可以看到，顶部和底部各有 22 个管脚，而左边和右边各有 23 个管脚。共有 10 组电源管脚，其中 4 组为管脚供电，分列于四周；6 组为内核供电，其中上下各一组，左右各两组。具体的管脚布局约束位于 pad_cell_cons.tcl 文件中。

5.2.2 时序结果

表 10 时序结果

阶段	Slack (ns)
data_setup	0.48(clk)
data_setup_zic	1.43(input),0.48(output),1.19(clk)
floorplan	0.95(input),-1.02(outputs),-1.39(clk)
placement	1.01(input),0.59(output),0.19(clk)

(续前表)

阶段	Slack (ns)
cts_only_psyn	1.26(input),0.82(output),0.62(clk)
cts_only_cts	1.26(input),0.83(output),0.36(clk)
route_initial	1.35(input),0.88(outputs),0.65(clk)
route_final	1.35(input),0.85(output),0.01(clk)

5.2.3 面积结果

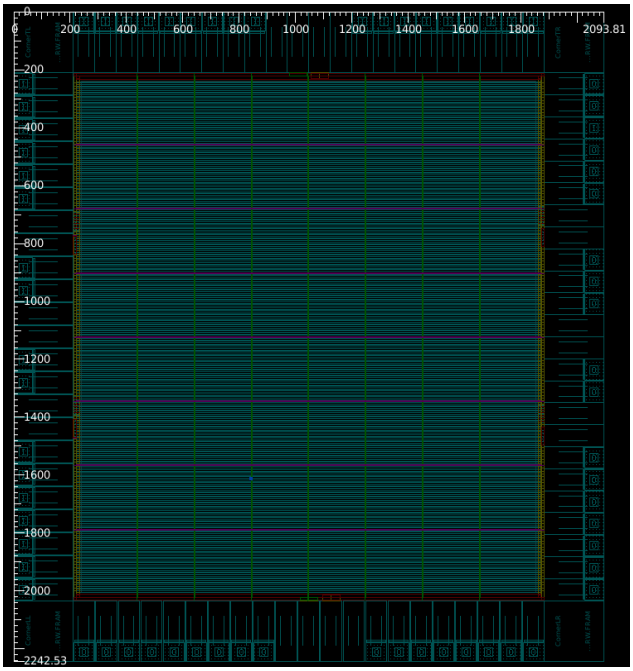

图 7 芯片面积测量

表 11 面积结果

项目	面积或长度
Combinational Area	1477888.208778 um^2
Non-combinational Area	154248.498180 um^2
Net Area	0.000000 um^2
Net X Length	620954.62 nm
Net Y Length	666869.44 nm
Cell Area	1632136.706958 um^2
Design Area	1632136.706958 um^2
Net Length	1287824.00 nm
Total Area	**4695300.72 um^2**

5.2.4 功耗结果

表 12 功耗结果

项目	电压或功耗
Global Operating Voltage	1.62 V
Cell Internal Power	33.6445 mW
Net Switching Power	12.7067 mW
Total Dynamic Power	44.3512 mW
Cell Leakage Power	26.0446 uW

5.2.5 芯片压降

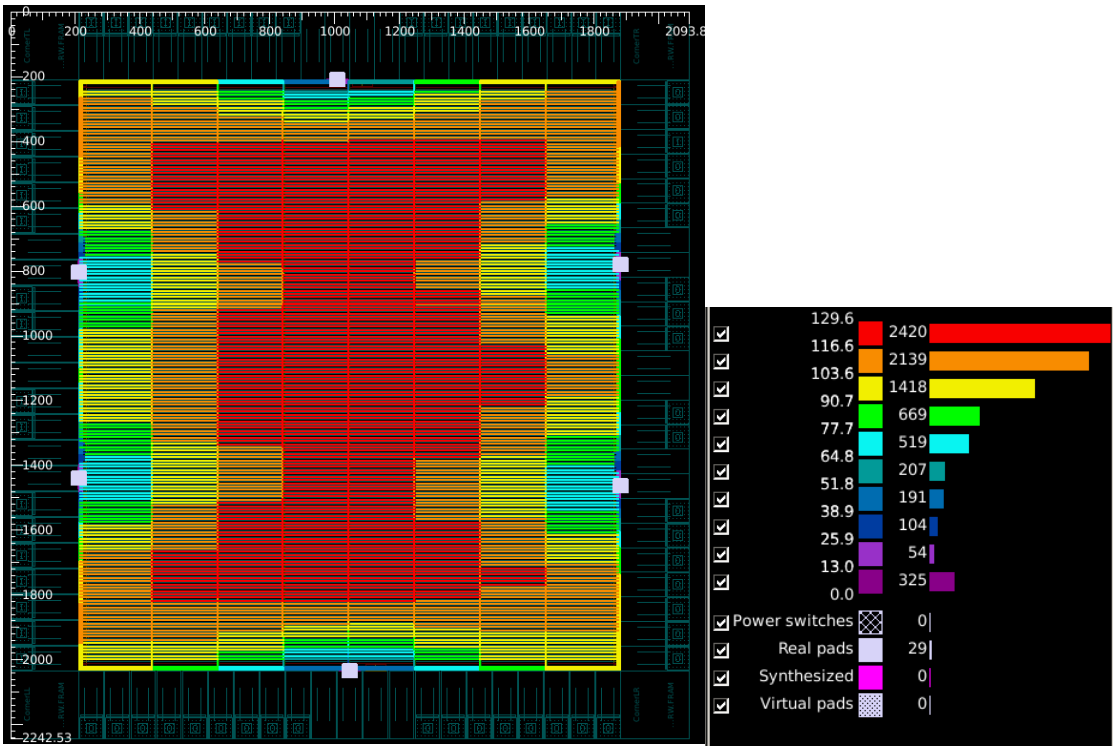

图 8 芯片压降

芯片的压降如图所示。由于芯片面积较大，所以共有五组电压源和接地管脚为内核供电。而由于芯片采用的工作电压为 1.62 V，而最大压降为 129.6 mV，小于工作电压的 10%，符合设计要求。

5.2.6 拥塞分析

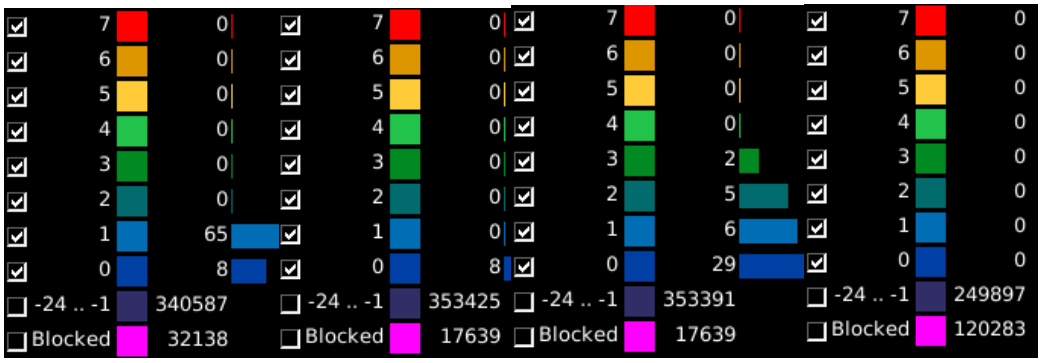

图 9 拥塞分析

如图所示，从左至右四幅图依次为 Placement、Detail Route、Track Assignment 和 Global Route 报告的拥塞。可以看到，由于我们的芯片面积较大，面积利用率设置得并不高，所以并没有遇到太多拥塞方面的问题，很容易地解决了拥塞问题。

5.2.7 资源使用

表 13 资源使用

项目	数量或利用率
Module Cells	16240 个
Pins	91614 个
IO Pad Cells	90 个
IO Pins	70 个
Nets	18151 个
Average Pins Per Net	3.0535 个
Total Std Cell Area	514936.70 um^2
Total Pad Cell Area	1612800.00 um^2
Core Size	1612.38×1764.00 = 2844238.32 um^2
Pad Core Size	1672.38×1824.00 = 3050421.12 um^2
Chip Size	2092.38×2244.00 = 4695300.72 um^2
Std cells utilization	18.10%

(续前表)

Cell/Core Ratio	18.10%
Cell/Chip Ratio	45.32%
Number of Cell Rows	350

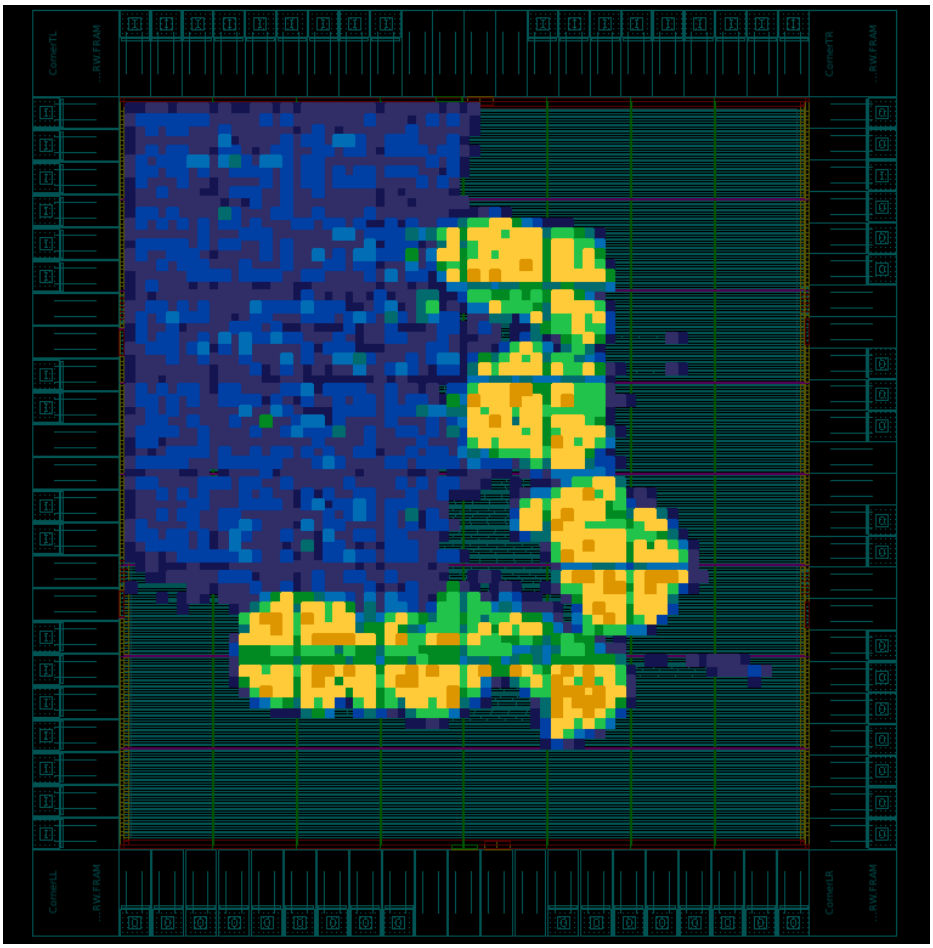

图 10 单元密度

5.2.8 设计违例

最终设计无设计违例，具体信息可参考各项报告。最终报告位于 final_design.rpt 中。

6 设计总结

6.1 遇到的主要问题

1）结构设计管脚数量过多，芯片面积过大；

2）电路的计算结果不正确；

3）时序模块数据传输不同步，拖慢运算速度；

4）芯片工作频率过低；

5）芯片电压降问题。

6.2 相应的解决思路

1）小组共同阅读了一些相关文献和博客，用串并转换的方法减少了管脚的使用；同时改用了基 4 FFT 算法，并复用了蝶形运算模块，减小了芯片面积；

2）计算结果不正确有多方面原因，包括测试模块编写的问题和各模块本身的问题。小组手动计算了许多关键的中间数据，共同比较和分析了各个模块的输入输出，几乎每个模块都有修复一些问题，最终正确实现了芯片功能；

3）时序模块数据传输不同步的解决方法是在时序电路传输标志信号的同时用组合逻辑电路传输数据，从而避免了数据传输不同步的问题；

4）芯片工作频率过低的解决方法是重写乘法器模块，将原来的乘法运算改为了用组合逻辑状态机判断旋转因子，然后采用移位的方法完成乘法运算，此项优化将芯片的工作频率从 82 MHz 提高到了 135 MHz；

5）电压降问题通过修改供电管脚数量和位置解决。这个问题的解决得到了何卫锋老师的帮助。

致谢

感谢何卫锋老师开设的课程给我们提供了学习完整的芯片设计流程的机会，也感谢老师课堂上的细致讲解。在设计阶段老师开设的每周两次的讨论极大地帮助我们优化了芯片的设计，也让我们学到了很多。

感谢张超助教在整个设计流程中提供的资料，这些资料极大地帮助我们优化了芯片的设计，而整个过程中助教也耐心解答了我们遇到的各种不明的报错信息，对我们有很多直接的帮助。

也要感谢同样做 FFT 的第 1 组的张佳玲和第 4 组的张云芳同学，虽然我们采取的 FFT 算法不同，但他们的思路对我们的局部优化也有一些参考价值。尤其是在我们前期设计工作频率过低的时候，他们两组的乘法器实现方案启发了我们对乘法器的优化。

同时也向帮助我们完成设计和帮助课程顺利进行的其他助教和同学表示感谢。除此之外，在前期我们还不清楚设计方向的时候，GitHub 和 CSDN 博客上其他人的工作也给我们了很多启发，在此一并致谢。我们的整个设计流程也通过 GitHub 实现版本管理，项目结束后也采取了 Mozilla 2.0 公共许可证进行开源，作为对开源社区的一点微小贡献。[项目链接]

参考资料

[1] Siva Kumar Palaniappan, et al. Design of 16-point Radix-4 Fast Fourier Transform in 0.18μm CMOS Technology [J]. American Journal of Applied Sciences 4(8): 570-575, 2007

[2] N. Weste, M. Bickerstaff, et al. A 50MHz 16-point FFT processor for WLAN application: IEEE 1997 Custom Integrated Circuits Conference: 457-460, 1997

[3] 丁晓磊等. 16 点基 4-FFT 芯片设计技术研究[J]. 信息技术. 64-71, 2007(1)