

数字集成电路课程 设计报告

面向 HEVC 的二维 IDCT 电路设计与实现

姓 名：孙永帅

学 号：515030910096

小组成员：刘真宏

提交日期：2018 年 4 月 28 日

目录

1.	课程设计规范简介.....	1
1.1	课程设计背景.....	1
1.2	课程设计要求.....	1
2.	电路性能分析与电路结构设计.....	2
2.1	电路性能分析.....	2
2.2	电路结构设计.....	2
2.2.1	HEVC 二维 IDCT 结构.....	2
2.2.2	乘法器的设计.....	3
2.2.3	Memory 缓存结构.....	4
2.2.4	流水线设计.....	5
2.2.4	电路整体架构.....	6
2.3	工作简介.....	6
3.	电路 RTL 模型与仿真验证.....	7
3.1	RTL 电路设计.....	7
3.1.1	IDCT 变换.....	7
3.1.2	Memory Change 设计.....	7
3.1.3	乘法 RTL 设计.....	8
3.2	仿真验证平台简介.....	8
3.3	验证方法设计.....	8
3.4	验证结果.....	9
3.5	工作简介.....	11
4.	电路逻辑综合策略与综合结果.....	12
4.1	逻辑综合简介与流程.....	12
4.2	逻辑综合关键参数.....	12
4.3	逻辑综合结果.....	13
4.4	工作简介.....	15
5.	电路物理实现与结果分析.....	16
5.1	物理实现简介.....	16
5.2	物理实现流程与结果.....	16
5.2.1	设计建立.....	16
5.2.2	布局规划.....	16
5.2.3	布局.....	19
5.2.4	时钟树综合.....	20
5.2.5	布线.....	20
5.3	工作简介.....	25
6.	任务分工与设计总结.....	26
6.1	任务分工.....	26
6.2	设计总结.....	26
6.2.1	遇到的问题与解决.....	26
6.2.2	实验总结与感想.....	27

1. 课程设计规范简介

1.1 课程设计背景

高性能视频编码(High efficiency video coding,简称 HEVC) 是 MPEG 和 ITU 组织面向未来超清视频应用(Ultra-High Definition TV,简称 UHD TV)而联合开发的新一代视频编码标准。

离散余弦变换(Discrete Cosine Transform, 简称 DCT)及其反变换 IDCT 是种被广泛用来压缩图像空间冗余的有效数学工具,也是 HEVC 编码器中最为重要的模块之一。它可以去除大量的图像冗余信息,将图像信息集中到少量的变换系数中,从而获得很高的视频压缩比。离散余弦变换不依赖于视频图像数据,它的变换系数矩阵拥有很好的对称性,有利于软、硬件快速有效的实现运算,因此被广泛应用于图像压缩、滤波等领域。在很多的视频编码标准(包括 HEVC 等)中, DCT 是消除图像空间冗余信息的重要手段,相应的在解码器端需要采用离散余弦反变换进行数据恢复。而由于 IDCT 需要的运算量很大,占了解码过程的大量时间,因此研究 IDCT 的模块设计和实现,在学习的同时具有很好的实用价值和意义。

在 HEVC 的 IDCT 模块的设计实现中,要实现面积成本的降低,就必须复用更多的模块,使得各模块尽量满负荷工作,然而这样的结果是速度的降低;要实现处理速度的进一步提高,就必须使用更多的模块来完成并行的处理能力,这样面积就会有所增加;于功耗成本亦是如此!因此面积、速度、功耗是在 IDCT 设计中的相互制约的要素,在满足相应的要求的情况下,可以进行一定的优化,探索一个减少硬件资源消耗,提高电路性能 of 电路结构。

1.2 课程设计要求

- 1) 面向 HDTV 应用,设计兼容 HEVC 标准的两维整数 IDCT 电路,完成电路的架构设计、Verilog HDL 代码设计、逻辑仿真、性能分析、逻辑综合、时序分析与验证和物理设计,进行结果分析比较等;
- 2) 支持兼容 HEVC 标准的 4 点和 8 点 IDCT 操作;
- 3) 满足实时处理要求,每秒 30 帧 HDTV 灰度图像(1920×1080@30fps)的实时处理能力;
- 4) 芯片的设计工艺: SMIC 0.18um 工艺;
- 5) IDCT 电路的评价指标:实时处理能力,芯片面积开销,输入/输出数据的带宽及其利用效率。

综上所述,本文设计的目的是完成兼容 4*4 点和 8*8 点的 IDCT 变换芯片设计,并满足相应的要求。重点在逻辑设计和相应优化。

2. 电路性能分析与电路结构设计

2.1 电路性能分析

首先从电路设计的工艺库分析，我们使用的是 SMIC 0.18um 的工艺库，在工作频率上，大约为 100M Hz，因此本设计基于 100M Hz 的工作频率下完成。

另一方面，由于需要满足实时处理要求，即为每秒 30 帧 HDTV 灰度图像 (1920×1080@30fps) 的实时处理能力，即每秒需要吞吐 30 帧 HDTV 灰度图像 (1920×1080@30fps)；考虑 HDTV 灰度图的每个像素为 8 Bits，但对于 IDCT 而言，输入为 16bits 有符号整数。因此一秒钟需要输入：

$$30 * 1,920 * 1,080 * 16 = 995,328,000 \text{ Bits}$$

对于 100MHz 的时钟频率而言，我们大约需要在每个时钟周期输入 10Bits 以满足实时处理的要求；但鉴于输入信号的关联性和完整性，这里我们可以考虑每个始终周期输入一个像素(16Bits)，这样既有利与后续的数据通路以及流水线的设计，也能给相应的模块留出足够的时钟裕度。

对于输出而言，我们同样可以在每个时钟周期输出 1 个像素(8Bits)，这样满足了每个时钟周期输入 1 个像素数据(16Bits)，同时，稳定工作后每秒钟也会输出一个像素的数据(8Bits)。这样就有：

$$\text{每秒输入: } 16 * 100M = 1600M \text{ Bits}$$

$$\text{每秒输出: } 8 * 100M = 800 \text{ M Bits}$$

2.2 电路结构设计

2.2.1 HEVC 二维 IDCT 结构

(1) 二维 DCT 的数学表示

$$f(x, y) = \frac{2}{n} \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} C(u) C(v) F(u, v) \cdot \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

其中， $f(x, y)$ 为输出变量， $F(u, v)$ 为输入变量，系数

$$C(m) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } m = 0 \\ 1 & \text{otherwise} \end{cases}$$

在视频的处理中，二维的 DCT 可以采用行列分解的方法分解成一维的行 DCT 和一维的列 DCT 变化，其中行列的 DCT 可以表述为：

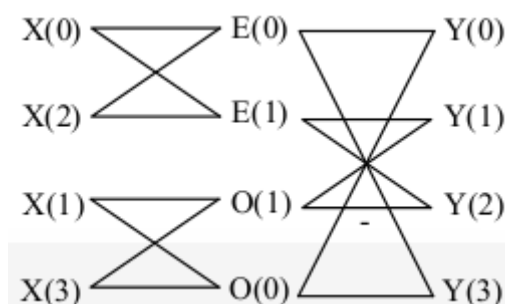
$$\frac{2}{\sqrt{2n}} \sum_{u=0}^{n-1} C(u)F(u) \cos \left[\frac{(2x+1)u\pi}{2n} \right]$$

(2) HEVC 一维的 IDCT 蝶形运算结构

根据上述对 DCT 的分析，我们可以得到一维的 IDCT 数学表达为：

$$X(k) = c(k) \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} Y(n) \cos \frac{(2n+1)k\pi}{2N}, k=0,1,2,\dots,N-1$$

因此，在设计的过程中，我们可以将 HEVC 中的二维的 IDCT 的变换分为两个一维的 IDCT 变换，分别为行变换和列变换，以便于在设计中的流水线的实现。对于 4*4 的一维 IDCT 蝶形运算结构如下图一：



图一 4 点一维蝶形运算结构

因此，在运算模块的设计，我们采用前后两个一维的 IDCT 蝶形变换结构来实现 HEVC 的二维 IDCT 变换。

2.2.2 乘法器的设计

对于 IDCT 运算，其数学表示如下图二所示：（以 4*4 点的 IDCT 列变换为例）

$$\begin{aligned} Y_0 &= (64X_0 + 83X_1 + 64X_2 + 36X_3 + add_{col}) \gg shift_{col} \\ Y_1 &= (64X_0 + 36X_1 - 64X_2 - 83X_3 + add_{col}) \gg shift_{col} \\ Y_2 &= (64X_0 - 36X_1 - 64X_2 + 83X_3 + add_{col}) \gg shift_{col} \\ Y_3 &= (64X_0 - 83X_1 + 64X_2 - 36X_3 + add_{col}) \gg shift_{col} \end{aligned}$$

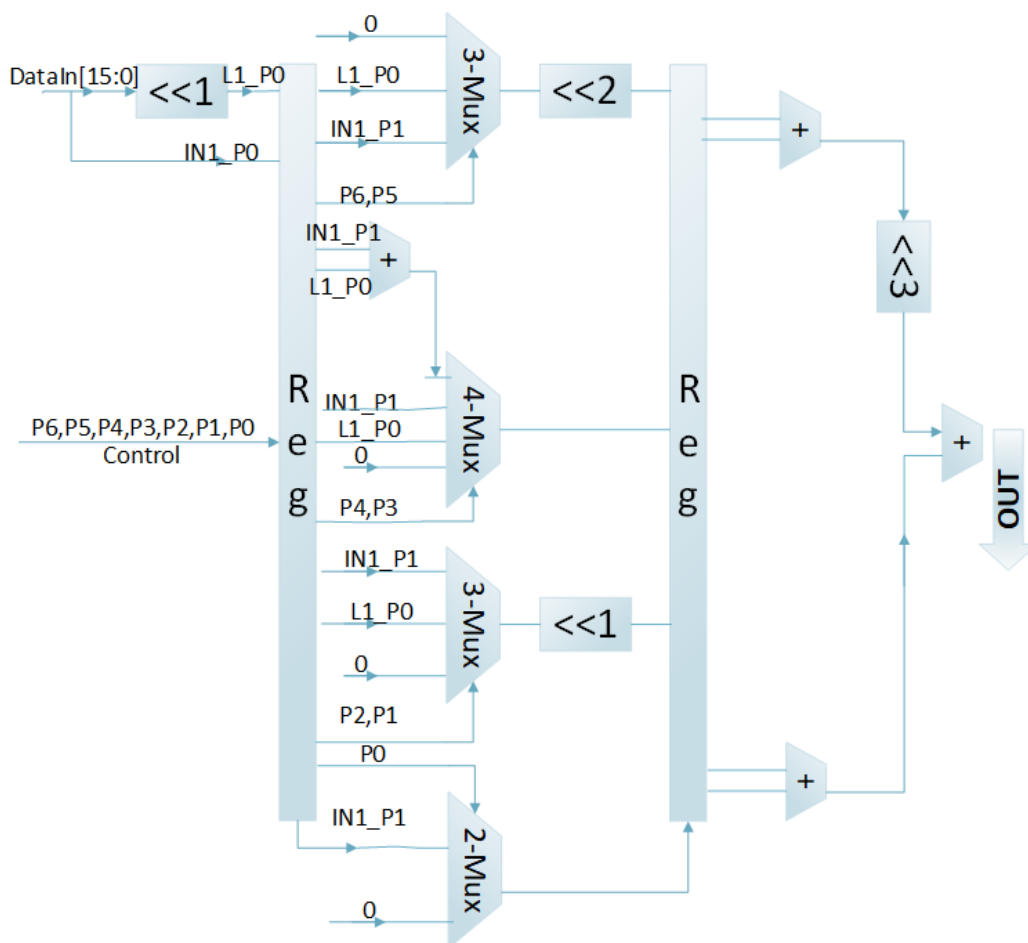
图二 IDCT 列变换

其中，x 为同一列输入像素数据，y 为变换后同一列输出像素数据。 add_{col} 和 $shift_{col}$ 共同实现了四舍五入和移位操作，标准中规定的移位位数在上式中给出。

注意到，在 IDCT 的运算中有很多的乘法的运算，大多是特定的数字乘以输入数据。这里我们规定输入数据为 16 位，同时发现需要乘的系数可以总结为七个（64 使用两次），如下：

64 89 83 75 64 50 36 18

因此，我们在设计乘法器是可以有一定的针对性。这里我们采用加法和移位操作来完成乘法操作。同时，为了缩短由于过多的移位和加法操作引起的最长路径，我们将乘法器（加法和移位）采用流水线的模式，将其在三个时钟周期内完成，其结构如下图三所示：



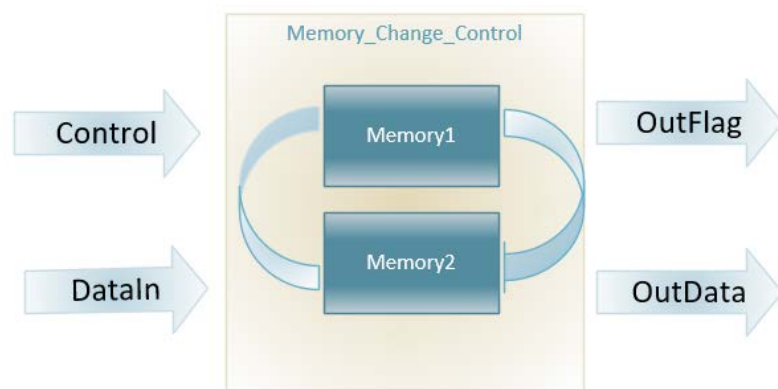
图三 乘法器设计

2.2.3 Memory 缓存结构

在上述的 IDCT 的结构的设计中，我们使用的是将二维的 HEVC 变换化为两个一维的 IDCT 变换，而这两个 IDCT 变换之间存在一定的数据联系。如果先进行行 IDCT 变换，那么在接下来的列 IDCT 变换中的输入数据将是前行 IDCT 变换的输出数据，但是由于输出数据的顺序和输入数据的顺序的不一致，我们不能直接将前一级的输出数据流水到后一级的输入数据。因此为了保证 IDCT 变换的高效有序进行，我们需要在一维的 IDCT 变换之间插入 Memory，来保证数据的有效传输。

由于本实验实现的是 8 点、4 点兼容的 IDCT 模块，因此在 Memory 的设计上应满足 8 点的 IDCT 变换所需的 Memory，因此这里设置为 $8*8=64$ 个 16 位位宽的 Memory。并通过 Memory Change 的模块进行地址和 Memory 编号的改变。这样就可以实现乒乓模式。而对于 $4*4$ 的模式下，将会 4 个 $4*4$ 的时候才会进行输出运算。即是 Memory 只有在填满后才会作为输出。

考虑到前面 IDCT 的行变换需要将数据写入到 Memory，而后一级的 IDCT 列变换需要将数据读出作为输入，但是由于写入和读出的数据的顺序的不同，如果使用单一的 Memory 将会出现写入和读出的冲突。因此，我们不能使用单一的 Memory 来完成相应的要求，因此这里采用两个 Memory 结果交替使用，来完成相应的数据交换，其结构图如下图四所示：

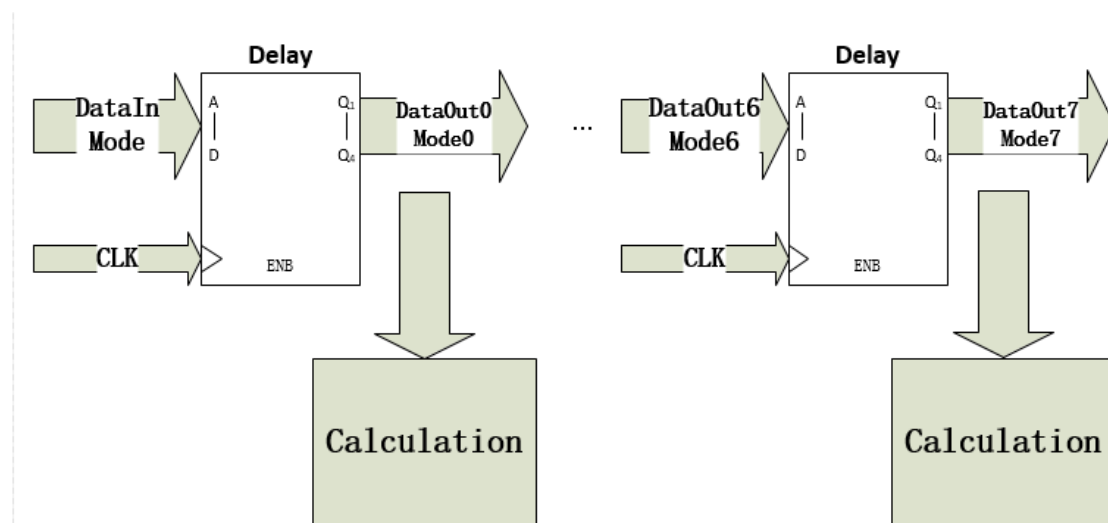


图四 Memory 乒乓结构示意图

2.2.4 流水线设计

在设计的过程中，为了提高模块的吞吐率，提高硬件资源的利用效率，往往采用流水线的模式。在这个设计过程中，流水线的思想更是贯穿始终。

由于要实现每个时钟周期并行输入一个像素(16Bits)，并且每个时钟周期输出一个像素(8Bits)，而由于中间的计算过程牵涉到每一块的所有的数据（行变换得到中间数据，中间数据列变换得到最终结果，因此结果中的每一个数据和块中的所有数据都有关系），因此我们需要使用流水线的模式，使得中间的计算过程有条不紊。在本实验中流水线主要体现在 IDCT 变换的数据上，其流水线的结构示意图如下图五所示：

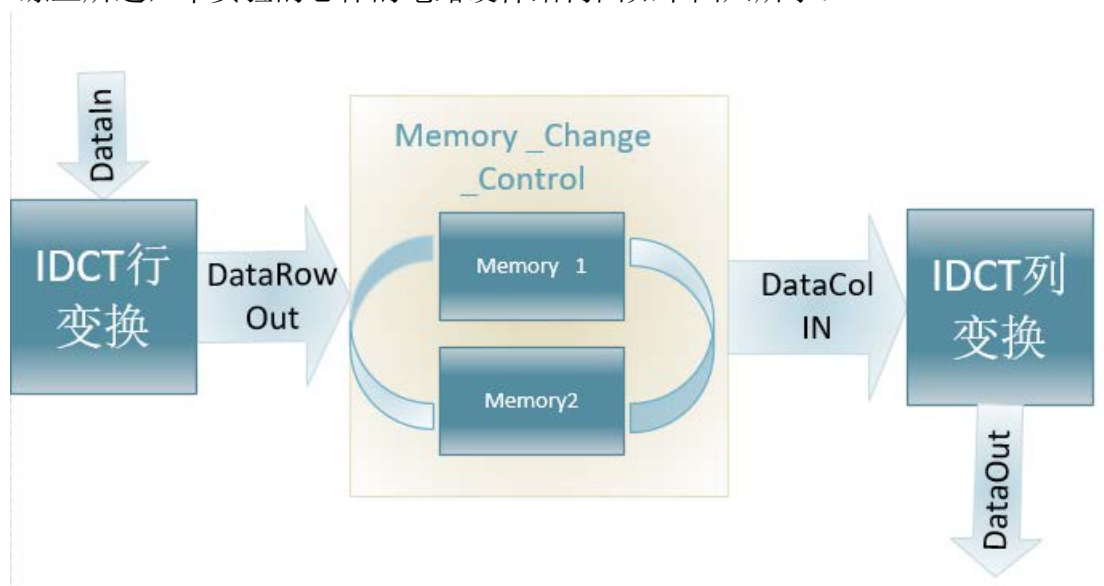


图五 IDCT 变换的流水线结构

上图展示了 8×8 点的 IDCT 输入数据的流水线数据通路；而对于 4×4 点的 IDCT 变换，采用复用 8×8 点的 IDCT 的模式。为了保证每个时钟周期输出一个数据，且中间没有输出数据的同时产生，这里使用 8×8 结构的后半部分计算 4×4 点 IDCT，即是 4×4 IDCT 流水到第四级的流水后开始计算，这样避免了 8×8 到 4×4 切换时的输出问题。

2.2.4 电路整体架构

综上所述，本实验的总体的电路硬件结构图如下图六所示：



图六 IDCT 电路硬件结构图

上述的电路硬件的结构中，数据通过输入总线进入模块，然后通过 IDCT 行变换的流水线，在每一级的流水线上依次进行计算，然后输出的数据通过 Memory 模块进行协调；另一方面 IDCT 列变换从 Memory 模块中读取数据并运算得到所得输出数据。

2.3 工作简介

在本部分的工作中，结构设计工作是我与队友共同分析完成的，很多的设计结合了我们共同的思想，区别主要在 RTL 电路的实现。我参与的工作包含：

- 1) IDCT 的流水线设计， 4×4 和 8×8 兼容模式；
- 2) 乘法器的结构设计，加法、移位、流水线设计；
- 3) Memory (RAM) 结构设计，乒乓模式的工作模式；
- 4) 电路结构整体规划，整合 IDCT、Memory 等。

上面的设计工作基于团队的合作完成，是我们共同努力的思路体现。

3. 电路 RTL 模型与仿真验证

3.1 RTL 电路设计

根据上一节中划定的模块,我们将整个工程分为行 IDCT 模块(IDCT_row.v)、列 IDCT 模块 (IDCT_col.v) 和 Memory Change 模块 (MemChange.v)。其中,行 IDCT 变换模块和列 IDCT 变换模块中核心乘法计算部分是用来乘法模块 (Multiply.v) 来完成,同时在 Memory Change 的模块中例化了两个 Memory 模块,来实现设计中的 memory 的乒乓模式。

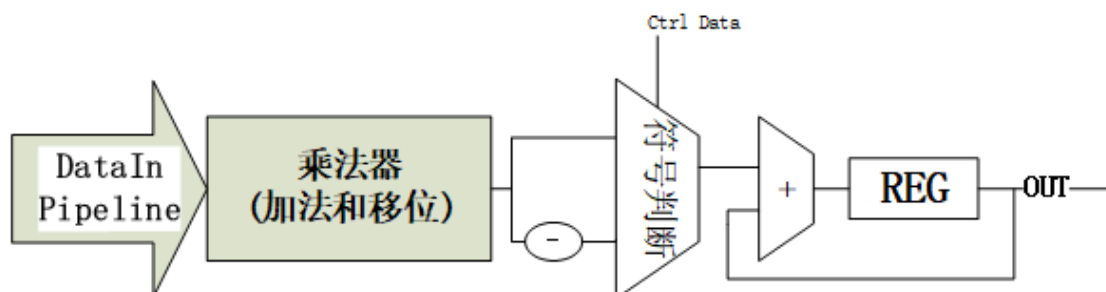
下面简要讲述下每个模块的设计。

3.1.1 IDCT 变换

在 IDCT 变换的设计中,主要采用流水线的设计模式。输入的数据、数据的模式 (4×4 or 8×8) 等数据通过流水线传送到每一级需要的地方,并进行相应的运算。其中,流水线的实现主要通过使用 Reg 来保存阶段数据来实现的。结合上述的总体结构,我们即可完成对 IDCT 的运算。

至于其中信号的输出,主要通过标志量来实现,比如何时取相应的输出输入,我们可以判断该条流水线所处的阶段,进而决定是否输出。同时设定输出的 Start 标志,确定何时接受输入数据。

其中单条的 IDCT 运算如下图七所示:



图七 IDCT 运算流程

3.1.2 Memory Change 设计

Memory Change 的设计主要是控制两个 Memory 模块的输入和输出,以及其读取和写地址,这样一来即可以实现对 Memory 的乒乓结构访存。其中,由于行列变换的数据顺序不同,关键的部分主要在于 Memory 地址的控制,本部分主要采用将取地址的情况,计算出其相应的地址表达式,如此一来,即可简化运算。此部分主要由队友设计完成。

3.1.3 乘法 RTL 设计

乘法的设计，主要依据上述的乘法结构的划分，将其做成流水线的模式。值得注意的是由于乘法器是流水线模式，因此同一时刻的输入输出并不对应，有一定的延时，因此在设计中要注意时序的控制。其中，乘法器模块的例化如下图八所示：

```

Multiply Multi_stage0(.clk(clk),
                      .rst_b(rst_b),
                      .data_multi(data_in_stage0),
                      .data_ctr(multiply_ctr_0),
                      .data_result(multi_res_stage0));

```

图八 乘法器的例化

3.2 仿真验证平台简介

本次的 RTL 级设计验证使用 Mentor 的 Modelsim10.02c Linux64 工具。将写好的 Verilog 模块加载到 Modelsim 的工程中，通过给与输入信号（Test Bench），即可得到相应的输出数据，来验证逻辑的正确性。

3.3 验证方法设计

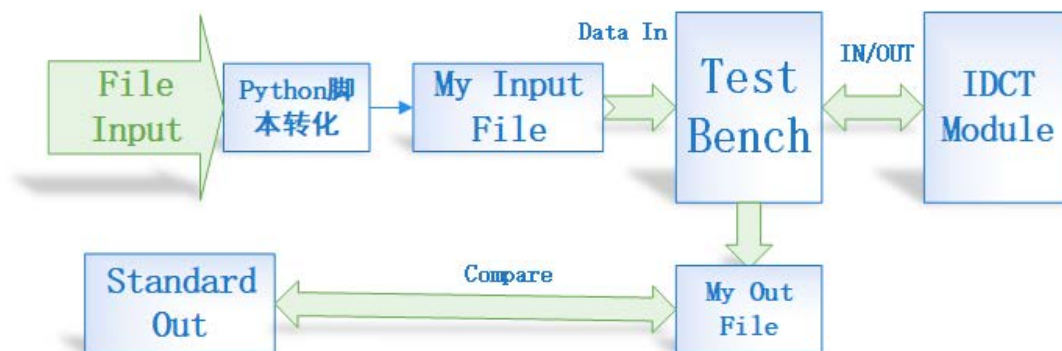
验证建立在 Modelsim 工具的使用，主要在于 Test Bench 的描述。在本次设计的验证中，测试给定的输入数据，来验证模块的功能。

首先，为了 Verilog 读出数据的方便，使用 Python 脚本，将输入的数据格式转化为我们方便读入的数据。这里我们在每一数据前添加单独的 Mode 标志。省去统一的 Mode 标志，方便 Verilog 读入。（j:mode=00; k:mode=01）

其次，在 Test Bench 中每个时钟周期读取一个数据，然后将其输入至例化的模块中，产生输入信号。

另外，通过例化的模块，我们也能够得到相应的输出，因此，我们可以将输出的信号和输入信号对比，从而验证电路的逻辑功能。

综上，其验证的流程大致如下图九：



图九 测试流程

3.4 验证结果

在完成相关的设计后，首先我们先将输入输出的标准数据转化为相应的格式，方便读入和对比，其格式大致如下图所示：（其中，j:mode=00；k: mode=01）

```

21503    j -11
21504    j -11
21505    k -67
21506    k -74
21507    k -83

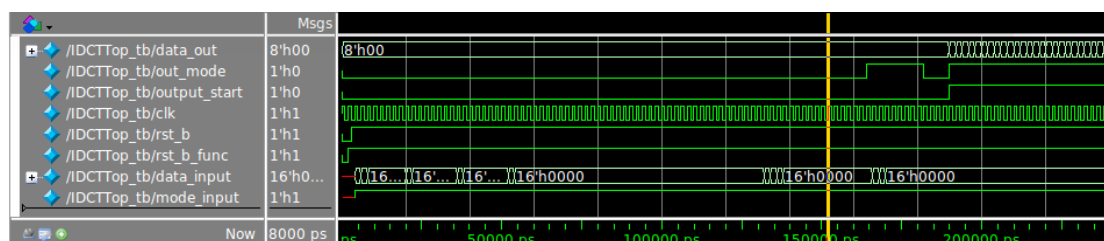
```

图十 转化的数据模式

在所给的源代码中给出了转化的 Python 脚本。仅供测试使用。

1)、 8*8 IDCT 变换

其波形图如下图所示：



图十一 8*8IDCT 变换波形图

其输出的结果对比如下图所示：

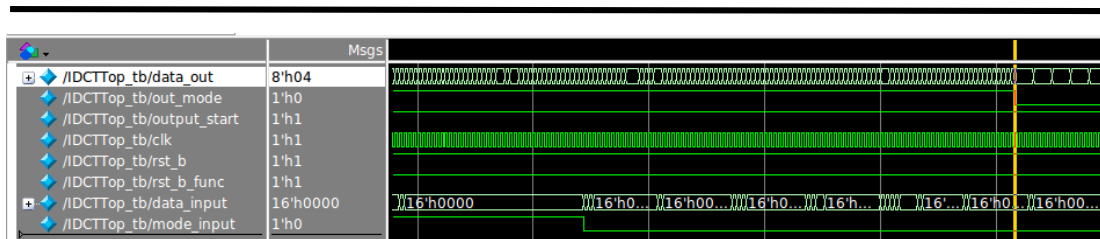
file_output_new_1.txt	file1_myOutput.txt
1 k -67	1 k -67
2 k -74	2 k -74
3 k -83	3 k -83
4 k -85	4 k -85
5 k -82	5 k -82
6 k -79	6 k -79
7 k -80	7 k -80
8 k -83	8 k -83
9 k -65	9 k -65
10 k -73	10 k -73

图十二 8*8IDCT 输出结果对比

从上面的数据可以看出，实现的 IDCT 模块在完成 8*8 点 IDCT 变换时，逻辑功能正确。

2) 8*8IDCT 变化为 4*4 点 IDCT

其波形图如下图所示：



图十三 8*8 点 IDCT 变化为 4*4 点 IDCT

其数据结果如下图十四所示：

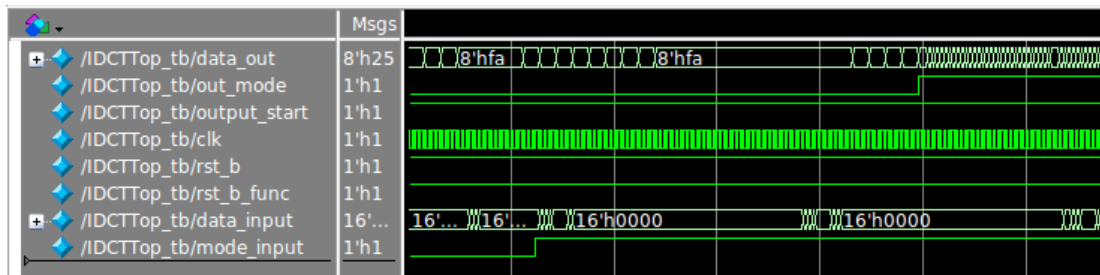
file_output_new_1.txt		file1_myOutput.txt
16382	k24	16382 k 24
16383	k35	16383 k 35
16384	k43	16384 k 43
16385	j4	16385 j 4
16386	j4	16386 j 4
16387	j4	16387 j 4
16388	j4	16388 j 4
16389	j8	16389 j 8

图十四 8*8 点 IDCT 变化为 4*4 点 IDCT

从上述的结果中我们可以看出，在 IDCT 变化中，可以兼容 8*8 点 IDCT 和 4*4 点 IDCT，中间可以完成切换，而没有错误。

3) 4*4 点 IDCT 切换 8*8 点 IDCT

波形输出结果如下图十五所示：



图十五 4*4 点 IDCT 变化为 8*8 点 IDCT

数据输出结果如下图十六所示：

file_output_new_1.txt		file1_myOutput.txt
21502	j -11	21502 j -11
21503	j -11	21503 j -11
21504	j -11	21504 j -11
21505	k -67	21505 k -67
21506	k -74	21506 k -74
21507	k -83	21507 k -83

图十六 4*4 点 IDCT 变化为 8*8 点 IDCT 输出数据对比

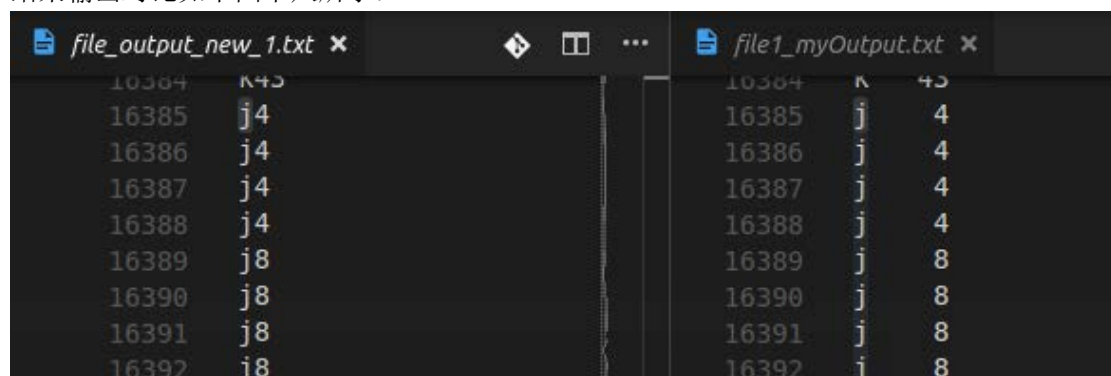
4) 4*4 点 IDCT 测试

波形输出结果如下图十七所示：



图十七 4*4 点 IDCT 变换波形

结果输出对比如下图十八所示：



图十八 4*4 点 IDCT 输出结果对比

从上述所得到的仿真结果来看，该设计的 IDCT 模块可以实现 4*4 点 IDCT 和 8*8 点 IDCT 的功能，并且在处理相应的运算时，可以兼容 4*4 点 IDCT 和 8*8 点 IDCT，而且中间没有时钟周期的浪费。

至于仿真的性能，在电路达到稳定状态时，每个时钟周期处理一个像素点的变换（输入一个，输出一个）；在刚开始工作时，需要经过约 90 个时钟周期开始产生输出数据（主要包含行变换的 64 个时钟周期，列变换的 8 个时钟周期，以及中间的延时流水线部分）。

经过所给定的完整数据的测试向量(file input 1.txt)测试结果可得，该 IDCT 模块功能正确完整，性能较好。

3.5 工作简介

本部分的工作主要是功能模块的 Verilog 实现和 Test Bench 的实现。在这一部分我的工作主要包含：

- 1) IDCT 流水线的设计实现，8*8 和 4*4 兼容实现；
- 2) 乘法器结构实现，移位、加法、流水线实现；
- 3) IDCT 顶层设计模块实现，IDCT_row, IDCT_col, Memory 的整合；
- 4) 整体电路的测试，整体电路测试工作，得到输出数据。

这部分工作中，在整合电路的时候得到了队友的很大的帮助，基于团队合作完成了该部分的工作，完备地完成了这些工作，得到了正确的结果。

4. 电路逻辑综合策略与综合结果

4.1 逻辑综合简介与流程

逻辑综合是指通过 EDA 工具将寄存器晶体管级的硬件描述语言自动编译产生符合所设定的约束条件的特定工艺下的门级网表(Netlist)的一个过程。其中, 束条件一般包括工艺条件、时钟、功耗和面积等, 而决定芯片性能的主要约束是时序约束信息。门级网表则是一种描述电路逻辑单元以及它们相互之间的关系的文件。

对电路的逻辑综合在其 RTL 代码通过功能仿真验证后进行, 综合工具是 Synopsys 公司的设计编译器(DC, Design Compiler), 采用 SMIC 0.18um 工艺。其流程大致包含如下:

- 1、Design Compiler 环境设置 (Libraries and Path)
- 2、RTL 文件读入(PAD 定义文件、IDCT 模块)
- 3、建立约束文件(时序约束、面积约束等)
- 4、查看报告(Reports)
- 5、优化(Optimize)

其中 PAD 的定义使用 PIW 作为输入 PAD, PO8W 作为输出 PAD。

4.2 逻辑综合关键参数

1) 时序约束

时序约束的定义如下图十九所示:

```
create_clock -period 10 [get_ports clk]
set_clock_uncertainty 0.3 [get_clocks clk]
set_clock_transition 0.1 [get_clocks clk]
set_dont_touch_network [get_clocks clk]
```

图十九 时间约束的定义

其中主要包含对时钟周期的约束, 这里定义为 10ns(100MHz); 时钟 uncertainty 的定义等

2) 线模型设定

这里使用的是 SMIC 0.18um 的工艺库, 线模型设定如下图二十所示:

```
set_wire_load_mode segmented
set_wire_load_model -name smic18_wl10
```

图二十 线模型设定

其他的参数主要包含最大面积设定、输出负载 (电容)、don't Touch 等。

4.3 逻辑综合结果

根据上述的设定，在 dc_shell-t 中编译的结果和报告如下：

1) Report Timing (Report : timing -path full -delay max -max_paths 1)

Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	4.00	4.00
IDCTTop1/rowIDCT/Multi_stage5/data_result_reg[1]/CK (DFFTRX2)	0.00 #	4.00 r
IDCTTop1/rowIDCT/Multi_stage5/data_result_reg[1]/Q (DFFTRX2)	0.60	4.60 r
IDCTTop1/rowIDCT/Multi_stage5/data_result[1] (Multiply_11)	0.00	4.60 r
IDCTTop1/rowIDCT/sub_631/B[1] (IDCT_row_DW01_sub_2)	0.00	4.60 r
IDCTTop1/rowIDCT/sub_631/U2/Y (INVX1)	0.17	4.76 f
IDCTTop1/rowIDCT/sub_631/U2_1/CO (ADDFX2)	0.57	5.33 f
IDCTTop1/rowIDCT/sub_631/U2_2/CO (ADDFX2)	0.39	5.73 f
IDCTTop1/rowIDCT/sub_631/U2_3/CO (ADDFX2)	0.39	6.12 f
IDCTTop1/rowIDCT/sub_631/U2_4/CO (ADDFX2)	0.39	6.51 f
IDCTTop1/rowIDCT/sub_631/U2_5/CO (ADDFX2)	0.39	6.91 f
IDCTTop1/rowIDCT/sub_631/U2_6/CO (ADDFX2)	0.39	7.30 f
IDCTTop1/rowIDCT/sub_631/U2_7/CO (ADDFX2)	0.39	7.69 f
IDCTTop1/rowIDCT/sub_631/U2_8/CO (ADDFX2)	0.39	8.09 f
IDCTTop1/rowIDCT/sub_631/U2_9/CO (ADDFX2)	0.39	8.48 f
IDCTTop1/rowIDCT/sub_631/U2_10/CO (ADDFX2)	0.39	8.87 f
IDCTTop1/rowIDCT/sub_631/U2_11/CO (ADDFX2)	0.39	9.26 f
IDCTTop1/rowIDCT/sub_631/U2_12/CO (ADDFX2)	0.39	9.66 f
IDCTTop1/rowIDCT/sub_631/U2_13/CO (ADDFX2)	0.39	10.05 f
IDCTTop1/rowIDCT/sub_631/U2_14/CO (ADDFX2)	0.39	10.44 f
IDCTTop1/rowIDCT/sub_631/U2_15/CO (ADDFX2)	0.39	10.84 f
IDCTTop1/rowIDCT/sub_631/U2_16/CO (ADDFX2)	0.39	11.23 f
IDCTTop1/rowIDCT/sub_631/U2_17/CO (ADDFX2)	0.39	11.62 f
IDCTTop1/rowIDCT/sub_631/U2_18/CO (ADDFX2)	0.39	12.02 f
IDCTTop1/rowIDCT/sub_631/U2_19/CO (ADDFX2)	0.40	12.42 f
IDCTTop1/rowIDCT/sub_631/U2_20/CO (ADDFHX2)	0.27	12.69 f
IDCTTop1/rowIDCT/sub_631/U2_21/S (ADDFHX4)	0.34	13.03 f
IDCTTop1/rowIDCT/sub_631/DIFF[21] (IDCT_row_DW01_sub_2)	0.00	13.03 f
IDCTTop1/rowIDCT/U559/Y (AOI22X1)	0.27	13.31 r
IDCTTop1/rowIDCT/U558/Y (OAI2BB1X1)	0.16	13.46 f
IDCTTop1/rowIDCT/data_out_5_reg[21]/D (DFFHQXL)	0.00	13.46 f
data arrival time		13.46
clock clk (rise edge)	10.00	10.00
clock network delay (ideal)	4.00	14.00
clock uncertainty	-0.30	13.70
IDCTTop1/rowIDCT/data_out_5_reg[21]/CK (DFFHQXL)	0.00	13.70 r
library setup time	-0.24	13.46
data required time		13.46
data required time		13.46
data arrival time		-13.46
slack (MET)		0.00

图二十一 DC Report Timing

从上述的 Report 结果中，我们可以看到，设计的结果满足时序的约束，符合设计的要求。从中，我们也可以看出最长的延时路径为 IDCT 运算中的加法操作，这主要是由于 IDCT 加法是 23 位的加法器，延时链较长。

2) Report Area (Report : area)

```

*****
Report : area
Design : idct_chip
Version: L-2016.03-SP1
Date   : Thu Apr 26 22:49:50 2018
*****

Library(s) Used:

    slow (File: /home/sun/Files/ModelsimProjects/SMIC18/db/slow.db)
    SP018W_V1p5_max (File: /home/sun/Files/ModelsimProjects/SMIC18/db/
    SP018W_V1p5_max.db)

Number of ports:          7096
Number of nets:           23464
Number of cells:          14174
Number of combinational cells: 8383
Number of sequential cells:  5675
Number of macros/black boxes: 0
Number of buf/inv:        1129
Number of references:      3

Combinational area:       691247.258108
Buf/Inv area:             471744.772913
Noncombinational area:    349860.781063
Macro/Black Box area:     0.000000
Net Interconnect area:    2168657.843353

Total cell area:          1041108.039171
Total area:               3209765.882524
1

```

图二十二 Report Area

上面是 Design Compiler 的面积报告，由于存在较大的误差，这里仅作参考。

3) Report Constraint Violators (Report: Constraint -all_violators)

```

Information: Updating design information... (UID-85)
Warning: Design 'idct_chip' contains 1 high-fanout nets. A fanout number of 1000
will be used for delay calculations involving these nets. (TIM-134)

*****
Report : constraint
        -all_violators
Design : idct_chip
Version: L-2016.03-SP1
Date   : Thu Apr 26 22:24:18 2018
*****

This design has no violated constraints.

1

```

图二十三 Report Constraint Violators

上图是设计违例的报告，从上面的报告中，我们可以看出其设计不存在违例，设计良好。

4) Report Power (Report : power-analysis_effort low)

Operating Conditions: slow Library: slow
Wire Load Model Mode: segmented

Design	Wire Load Model	Library
idct_chip	smic18_wl10	slow

Global Operating Voltage = 1.62

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 1.000000pf

Time Units = 1ns

Dynamic Power Units = 1mW (derived from V,C,T units)

Leakage Power Units = 1pW

Cell Internal Power = 32.5632 mW (92%)

Net Switching Power = 2.7313 mW (8%)

Total Dynamic Power = 35.2945 mW (100%)

Cell Leakage Power = 20.9808 uW

Power Group (%) Attrs	Internal Power	Switching Power	Leakage Power	Total Power
io_pad (2.21%)	0.7717	8.8565e-03	2.8752e+05	0.7808
memory (0.00%)	0.0000	0.0000	0.0000	0.0000
black_box (0.00%)	0.0000	0.0000	0.0000	0.0000
clock_network (0.00%)	0.0000	0.0000	0.0000	0.0000
register (89.60%)	30.7401	0.8921	8.7613e+06	31.6410
sequential (0.00%)	0.0000	0.0000	0.0000	0.0000
combinational (8.19%)	1.0514	1.8303	1.1932e+07	2.8936
Total 1	32.5632 mW	2.7313 mW	2.0981e+07 pW	35.3154 mW

图二十四 Report Power

上图是 IDCT 模块的功耗报告，功耗约为 35mW，其中 Internal Power 占据了绝大部分的功耗，其他还包含开关功耗和漏电流功耗。

综上所述，Design Compiler 完成了 IDCT 模块的逻辑综合，生成了相关的门级网表。从结果来看，各个指标满足要求，符合预定目标。

4.4 工作简介

这部分的工作，在团队合作的基础上我主要完成了工作包含：

1) Design Compiler 环境的搭建；

2) 顶层相关 PAD 文件的定义。

这部分内容主要基于老师和助教提供的参考文档和参考 TCL 代码完成，改成我们所需要的约束条件，在自己的设计和创造上较少。

5. 电路物理实现与结果分析

5.1 物理实现简介

IC Compiler 是 Synopsys 公司的新一代的布局布线工具。本设计的物理实现部分采用 IC Compiler 完成布局布线等物理实现。本文设计基于 IC Compiler -vL - 2016.03 版本设计, 相关优化约束和旧版本有所不同。

布局布线是将门级网表转换成电路版图的过程,其目的是使芯片拥有较短的互连线和较小的布局布线面积,主要工作时对组成芯片的各个标准单元、子模块的位置进行合理的规划。这样,可以增加一个芯片上所能集成的标准单元个数,提高芯片的集成度和资源利用率,降低芯片设计成本。

本设计的实现主要基于参考的 ICC 脚本实现。其主要的 ICC 设计电路设计流程如下:

- 1、设计建立(Design Data Setup)
- 2、布局规划(Design Planning)
- 3、布局(Placement)
- 4、时钟树综合(Clock Tree Synthesis, CTS)
- 5、布线设计(Routing)

5.2 物理实现流程与结果

5.2.1 设计建立

集成电路后端设计的开始是设计建立环节,包括设计库和电路单元的建立。一个设计库应该包括门级网表文件、参考库的指向地址、参考库所包括的设计模块、时序约束文件以及所采用的工艺库文件。因此设计库建立的基本过程是:读入工艺库、指向参考库、读入网表、扩展网表等。本文的设计采用 SMIC 0.18um 工艺库,因此需要将工艺库文件导入。同时需要读入 DC 逻辑综合步骤中生成的门级网表文件等。至此,我们的 IDCT 模块已完成基本数据的建立。

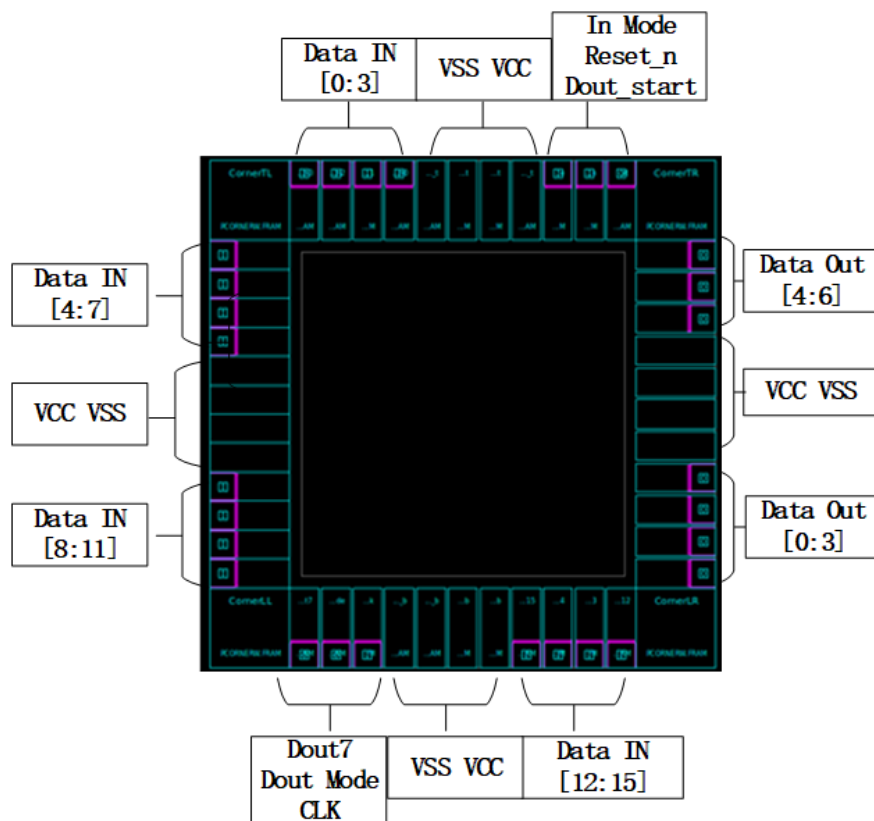
5.2.2 布局规划

随着集成电路设计规模越来越大,布局规划在后端设计中也越来越重要,它决定着芯片的布图质量。布局规划的工作是对芯片的整体结构及参数作出规划,主要包括:芯片 I/O 引脚的位置、调用宏单元的放置位置、芯片的面积大小和形状、组成芯片的各个标准单元的摆放位置和形式的设计。这一步骤主要包括一下几个方面:

- 1)、Create Starting Floor Plan
- 2)、Virtual Flat Placement
- 3)、Analyze & Optimize(Timing\Power)

4)、Write out

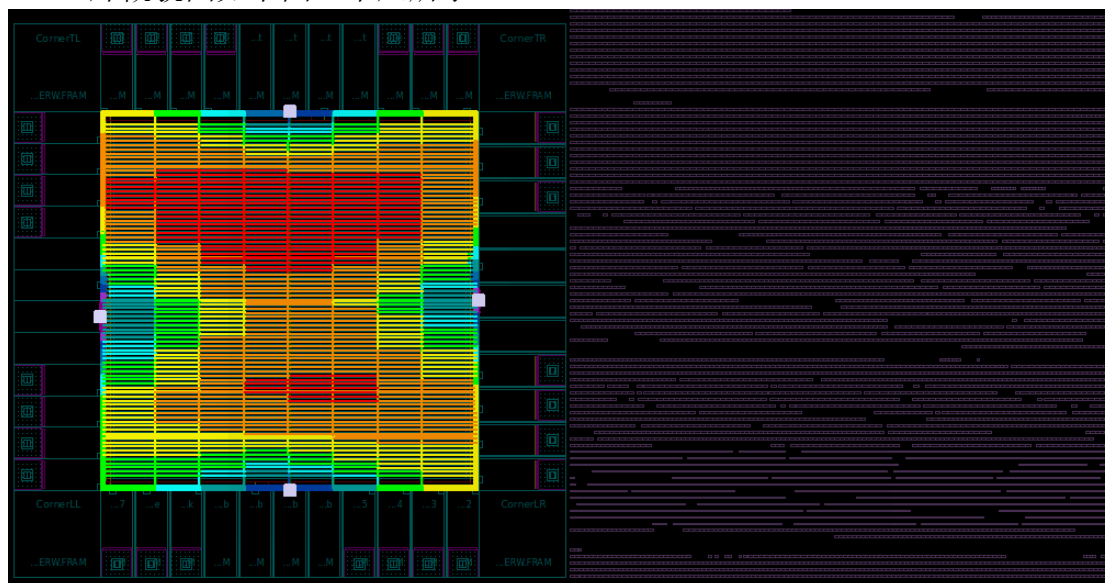
建立起始的 Floor plan，我们需要定义芯片的 I/O 引脚的位置，这里我们在 pad_cell_cons.tcl 中将我们所有的 I/O 引脚进行定义，并添加 VDD 和 VSS 的 PAD，为了更好的压降，我们将 VDD 和 VSS 引脚摆放在四周，均匀分布。至此我们得到了芯片的大致引脚分配图，如下图二十五所示：



图二十五 引脚分配图

然后依次进行虚拟布局，和时序的分析优化，我们可以得到如下结果：

1) 芯片概貌图如下图二十六所示：



图二十六 布局规划后的芯片概貌

2) 布局规划后其时序分析结果如下图二十七所示:

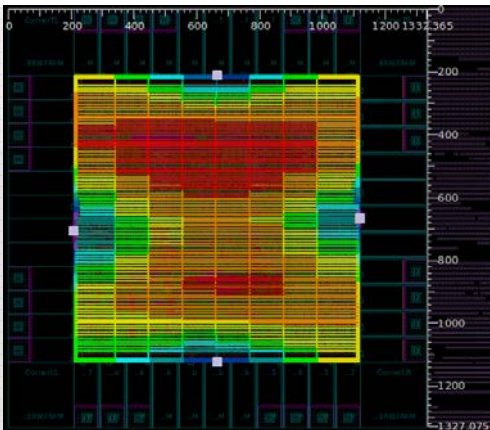
IDCTop1/colIDCT/Multi_stage4/add_90_3/SUM[22] (Multiply_4_DW01_add_1)	0.55	12.04	f
IDCTop1/colIDCT/Multi_stage4/data_result_reg_22/D (DFFTRXL)	0.00	12.64	f
data arrival time	0.00	12.64	f
clock clk (rise edge)	10.00	10.00	
clock network delay (ideal)	4.00	14.00	
clock uncertainty	-0.30	13.70	
IDCTop1/colIDCT/Multi_stage4/data_result_reg_22/CK (DFFTRXL)	0.00	13.70	r
library setup time	-0.25	13.45	
data required time		13.45	
data required time		13.45	
data arrival time		-12.64	
slack (MET)		0.81	

图二十七 Floor plan 后的时序结果

从图中我们可以看出, 在 Floor Plan 后仍满足要求。

3) 其面积结果如下图二十八所示:

Number of ports:	29
Number of nets:	60
Number of cells:	30
Number of combinational cells:	29
Number of sequential cells:	0
Number of macros/black boxes:	0
Number of buf/inv:	0
Number of references:	3
Combinational area:	691247.258108
Buf/Inv area:	471744.772913
Noncombinational area:	349860.781063
Macro/Black Box area:	0.000000
Net Interconnect area:	2179484.577545
Total cell area:	1041108.039171
Total area:	3220592.616716
1	



(a) Report Area

(b) 测量尺寸

图二十八 布局规划后芯片面积报告

从上面的图中我们可以看出通过 Report Area 命令得到的 Total Area 较实际测量的面积大, 这里我们以实际测量的为主。其面积大小约为 $1327\mu\text{m} \times 1327\mu\text{m}$ 。

4) 功耗结果如下图二十九所示:

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io pad	0.7973	2.8910e-02	2.8752e+05	0.8265	(2.42%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	30.6220	0.4857	8.6413e+06	31.1164	(91.06%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	1.1993	1.0182	1.1495e+07	2.2289	(6.52%)	
Total	32.6186 mW	1.5328 mW	2.0424e+07 pW	34.1718 mW		

图二十九 功耗报告

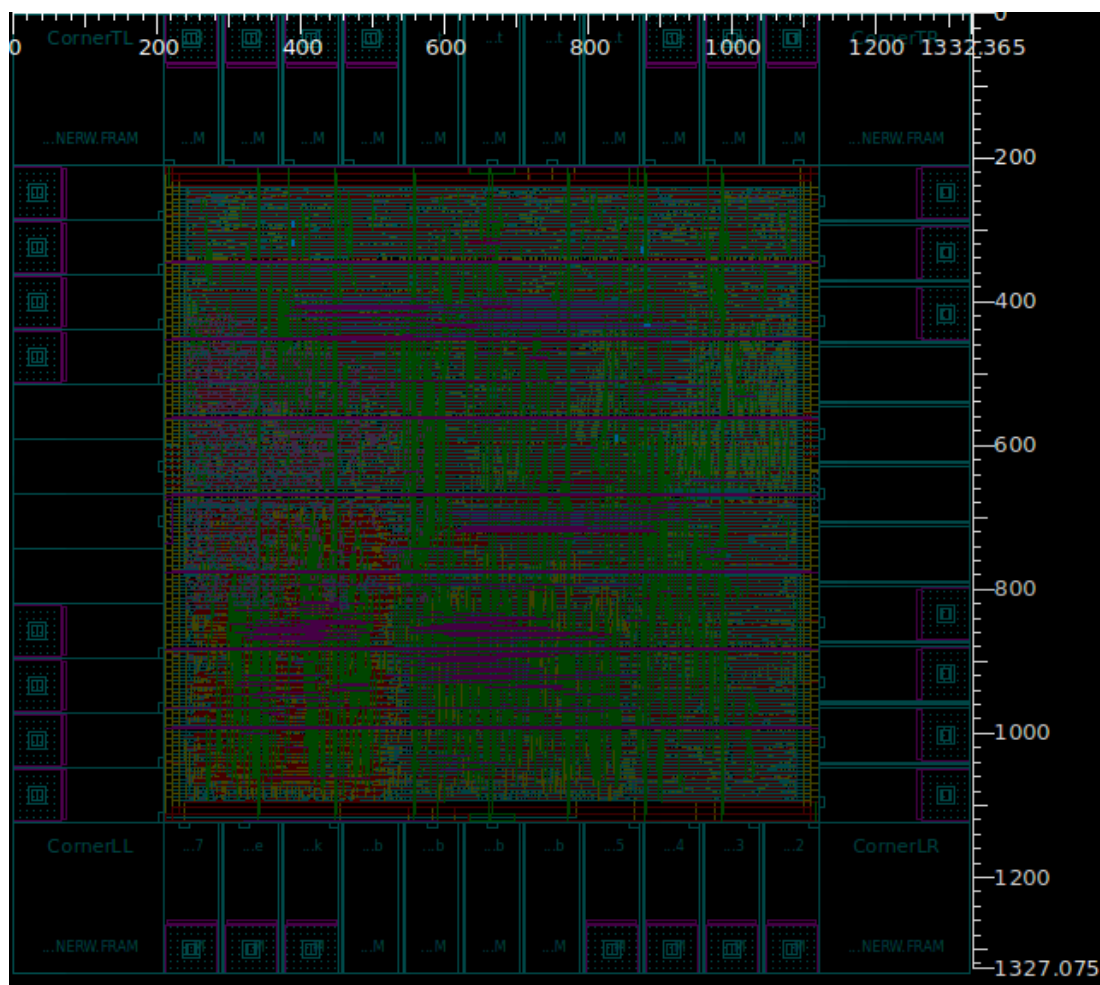
从上面的报告中我们可以看出,功耗报告大约为 34mW。其中芯片的 Internal Power, 约为 32.6mW, 占据了功耗的大部分。另外还包含开关功耗和漏电流功耗。

5.2.3 布局

布局设计是一个将每个电路标准单元在芯片中的位置进行确定的过程。芯片的性能、面积消耗、布线布通率以及后端设计需要的时间都会受到布局好坏的影响,一个合理布局的基本要求是标准电路单元之间不会重叠覆盖,并且都放在芯片相应的有效位置上。一般在布局规划后进行布局设计,它需要的输入信息应该包括:电路网表信息、布局规划信息、工艺库中标准单元的物理信息等。布局设计结束后的输出满足设计需求的电路版图。

布局设计是一个多个目标折中的过程,需要在多个目标中进行权衡,一个布局设计往往需要进行多次迭代,多次调解来得到最佳解决方案。这里设定的芯片逻辑所占比为 0.6。这里我们通过在 run_placement.tcl 中添加相关约束后, Source 来执行来完成布局。

完成布局后的芯片概貌如下图三十所示:



图三十 布局后芯片概貌

5.2.4 时钟树综合

当集成电路设计进入深亚微米阶段,决定电路时钟频率的主要因素有两个:一是同步电路之间的时钟偏差,二是组合逻辑电路中的关键路径延时。随着集成电路设计技术的不断发展,路的开关频率不断提高,因此时钟偏差是制约电路处理性能的关键因素之一。减小电路系统的时钟偏差是时钟树综合的主要目的,为了满足电路时序收敛的要求,保证每个模块和每个寄存器的时钟输入相位差最小,必须在时钟源到寄存器最短的通路上插放延时单元,使得时钟到元器件的所有路径在延时上都与最长路径相同。时钟树综合可以完成上面对时钟的优化,通过在run_cts.tcl中添加约束,其中包含:

设置时钟树: set_clock_tree_options -target_skew 0.3

设置时钟不确定性: set_clock_uncertainty 0.2 [all_clocks]

时钟优化: clock_opt -only_cts -no_clock_route -update_clock_latency

时钟优化: clock_opt -only_psyn -no_clock_route

其中最后结果如下图三十一所示:

IDCTTop1/colIDCT/U46/Y (A0I22X1)	0.00	14.21	f
IDCTTop1/colIDCT/U3/Y (OAI2BB1X1)	0.19	14.40	r
IDCTTop1/colIDCT/U3/Y (OAI2BB1X1)	0.10	14.50	f
IDCTTop1/colIDCT/data_out_2_reg_22_/D (DFFHQXL)	0.00	14.50	f
data arrival time		14.50	
clock clk (rise edge)	10.00	10.00	
clock network delay (propagated)	5.19	15.19	
clock uncertainty	-0.20	14.99	
IDCTTop1/colIDCT/data_out_2_reg_22_/CK (DFFHQXL)	0.00	14.99	r
library setup time	-0.20	14.79	
data required time		14.79	
data arrival time		-14.50	
slack (MET)		0.29	

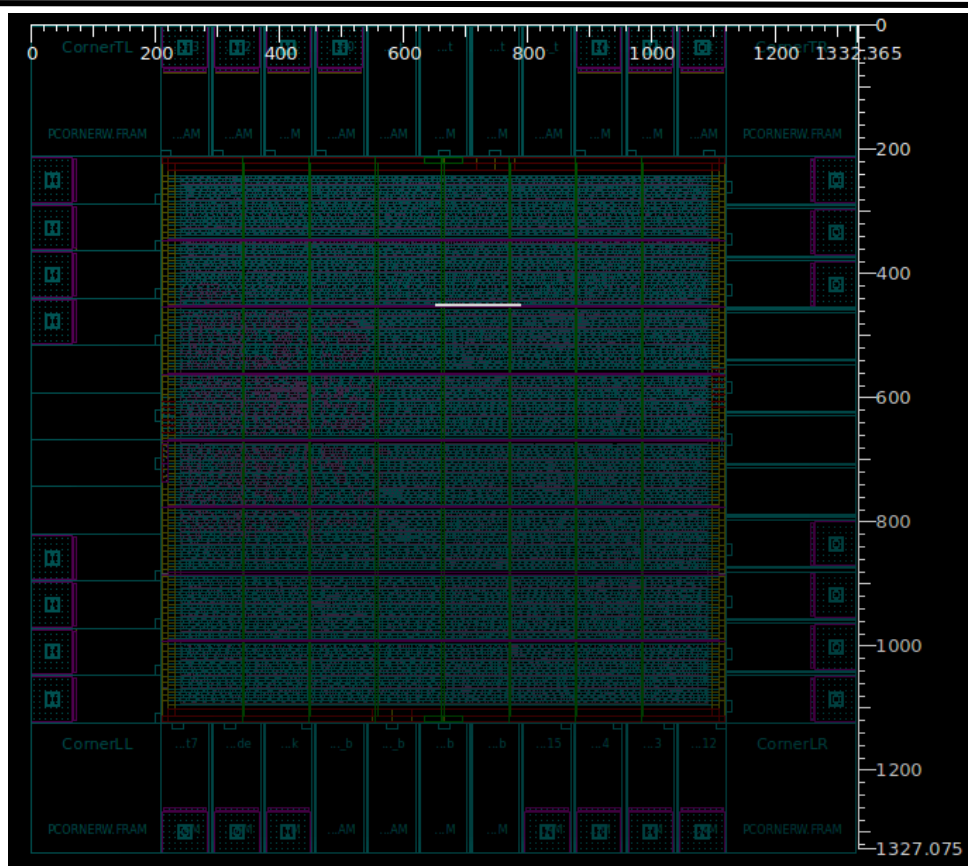
图三十一 CTS 中后时钟结果

从结果中,我们可以看出时钟树综合后仍满足时序要求(100MHz)。

5.2.5 布线

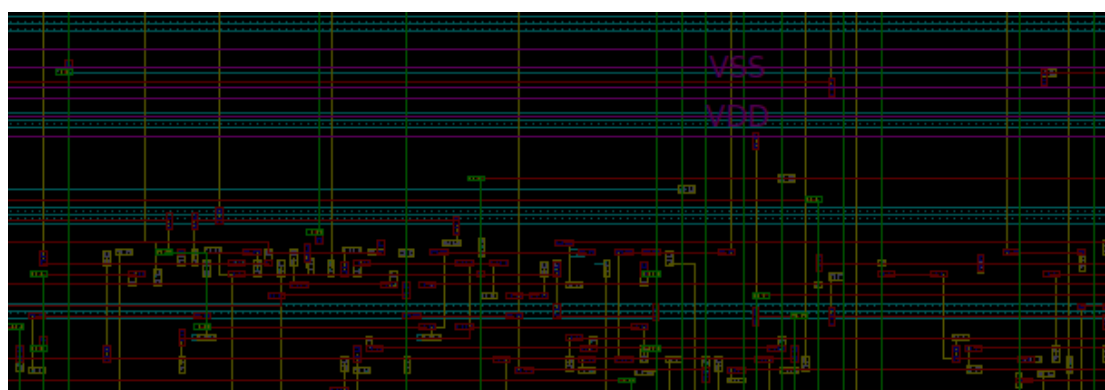
当布局设计和时钟树综合完成后,芯片中各个单元的位置都已放置完毕,然后将完成芯片的布线工作,将各个电路单元进行连接。布线是完成版图中各个器件单元之间的物理连接。

1) 布线后芯片概貌图如下三十二所示:



图三十二 布线后电路图概貌

其中的细节部分如下图三十三所示：



图三十三 布线细节体现

2) 布线后时序结果如下图三十四所示：

IDCTop1/colIDCT/Multi_stage0/add_90_3/SUM[22] (Multiply_8_DW01_add_1)	0.24 @	14.65 f
IDCTop1/colIDCT/Multi_stage0/data_result_reg_22_/D (DFFTRXL)	0.00	14.65 f
data arrival time	0.00 @	14.65 f
clock clk (rise edge)	10.00	10.00
clock network delay (propagated)	5.06	15.06
clock reconvergence pessimism	0.09	15.16
clock uncertainty	-0.20	14.96
IDCTop1/colIDCT/Multi_stage0/data_result_reg_22_/CK (DFFTRXL)	0.00	14.96 r
library setup time	-0.21	14.75
data required time		14.75
data required time		14.75
data arrival time		-14.65
slack (MET)		0.10

图三十四 routing 时序报告后

3) 功耗结果如下图三十五所示:

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	8.7992e-02	3.5839e-02	2.8752e+05	0.1241	(0.29%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.7458	7.9985	2.8906e+05	8.7446	(20.35%)	
register	31.5074	0.4538	8.6394e+06	31.9698	(74.39%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	1.1645	0.9639	1.0929e+07	2.1393	(4.98%)	
Total	33.5057 mW	9.4520 mW	2.0145e+07 pW	42.9778 mW		

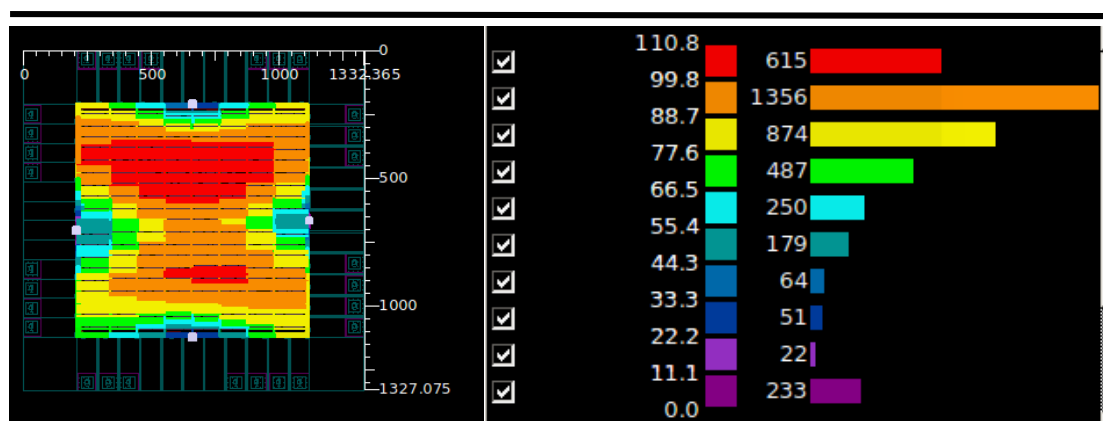
图三十五 routing 后功耗 Report

4) 面积结果报告如下图三十六:

Number of ports:	29
Number of nets:	60
Number of cells:	30
Number of combinational cells:	29
Number of sequential cells:	0
Number of macros/black boxes:	0
Number of buf/inv:	0
Number of references:	3
Combinational area:	690409.005685
Buf/Inv area:	472576.372983
Noncombinational area:	348363.901051
Macro/Black Box area:	0.000000
Net Interconnect area:	2186297.418549
Total cell area:	1038772.906736
Total area:	3225070.325284

图三十六 Routing 后面积报告

5) 芯片压降如下图三十七所示:



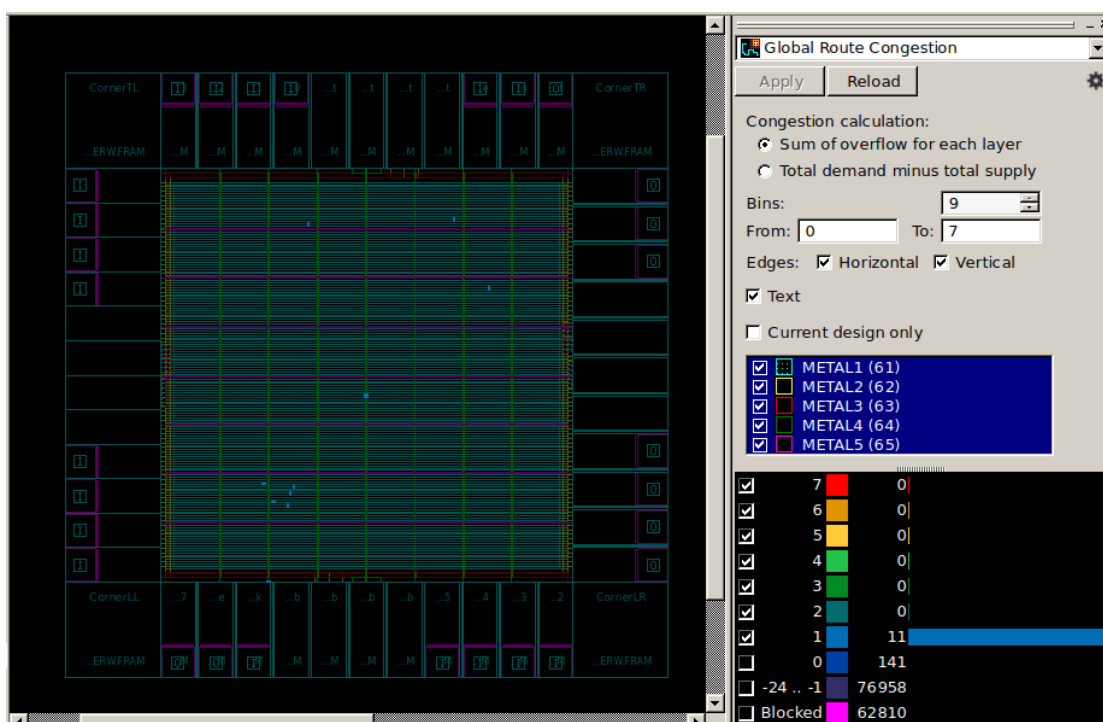
(a) 压降分析芯片图

(b) 压降分析图例

图三十七 压降分析图

从上面的压降分析来看，芯片中间的压降为最多，从图例中我们可以看出，其压降约为 110mV，满足最大的压降不超过 10%的要求(即： $1.8V * 10\% = 180mV$)，满足设计要求。

6) 芯片拥塞分析如下图三十八所示：



图三十八 芯片拥塞分析

从上面的结果中我们可以看出没有严重的拥塞情况，设计符合要求。

7) 资源使用情况

使用 Report_utilization 命令得到其芯片的 Summary，其中资源使用情况如下图三十九所示：

```

*****
Report : Chip Summary
Design : idct_chip
Version: L-2016.03-SP1
Date   : Sat Apr 28 14:05:29 2018
*****
Std cell utilization: 79.62% (173315/(217672-0))
(Non-fixed + Fixed)
Std cell utilization: 79.55% (172525/((217672-790)))
(Non-fixed only)
Chip area:      217672 sites, bbox (240.00 240.00 1090.08 1091.76) um
Std cell area:  173315 sites, (non-fixed:172525 fixed:790)
Macro cell area: 13752 cells, (non-fixed:13620 fixed:132)
Placement blockages: 0 sites
Routing blockages: 0 sites, (excluding fixed std cells)
Lib cell count:  790 sites, (include fixed std cells & chimney area)
Avg. std cell width: 5.44 um
Site array:      unit (width: 0.66 um, height: 5.04 um, rows: 169)
Physical DB scale: 1000 db_unit = 1 um

*****
Report : pnet options
Design : idct_chip
Version: L-2016.03-SP1
Date   : Sat Apr 28 14:05:29 2018
*****

```

Layer	Blockage	Min_width	Min_height	Via_additive	Density
METAL1	none	---	---	via additive	---
METAL2	partial	10.08	2.52	via additive	---
METAL3	partial	10000.00	10000.00	via additive	---
METAL4	partial	10000.00	10000.00	via additive	---
METAL5	none	---	---	via additive	---

1

图三十九 芯片中的面积具体情况

从中我们可以看出 Std Cell utilization 约为 80%，达到设计要求。同时从上面的报告中我们也可以得到面积、金属层、等相关的信息。

8) 设计违例如下图四十所示:

```

*****
Report : constraint
        -all_violators
Design : idct_chip
Version: L-2016.03-SP1
Date   : Fri Apr 27 02:33:37 2018
*****

```

max_area

Design	Required Area	Actual Area	Slack
idct chip	0.00	3225070.25	-3225070.25

(VIOLATED)

图四十 设计违例报告

从上图中，我们可以看出设计中只有面积的设计违例，这是由于在约束中设置的最大的面积为 0，因此为违例。所以，总体来讲，本文的设计基本无违例情况，符合设计要求。

5.3 工作简介

这部分的主要工作是基于老师提供的 ICC 设计文档和提供的 ICC 所需的 TCL 文件完成。在这部分我们本着学习的原则，和我的队友都有完成。这部分我参与的工作包含：

- 1) PAD 位置的定义设定，定义相关文件，确定 PAD 位置
- 2) 布局，参考教程与相关 TCL 文件，完成芯片布局
- 3) 布线，参考教程和相关 TCL 文件，完成芯片布线

这一部分，主要是有关的优化和设计报告的查看，相关的约束参考教程和给定的 TCL 文件较多，个人的设计较少。同时，这部分我和队友都将自己的 IDCT 项目整合，在相关平台上完成。

6.任务分工与设计总结

6.1 任务分工

本实验在团队合作的基础上，完成相关的设计。相关的思路和结构的设计基本上是我们讨论的结果，下表一简要记录下相关工作的参与情况，仅作参考！

表一 任务参与记录表

	孙永帅	刘真宏
逻辑架构设计	IDCT 流水线 Memory 乒乓结构 等	IDCT 流水线 4*4 和 8*8 兼容设计 Memory 寻址 等
RTL 实现与测试	IDCT 流水线实现 IDCT Top 结构实现 整体测试 等	Memory 结构实现 Test Bench 实现 等
逻辑综合	Design Compiler 完成	Design Compiler 完成
物理实现	IC Compiler 完成 (布局布线, 时钟树综合)	IC Compiler 完成 (布局布线, 时钟树综合)

6.2 设计总结

6.2.1 遇到的问题与解决

在这次的设计中，我们较为顺利地完成了 IDCT 设计相关的工作，符合要求地完成了相关的内容，达到了相应的目标。但其中也遇到了一些问题，这里作简要概述。

1) Test Bench 的设计

最开始完成 IDCT 功能时，我们测试的样本一般为自取的 4*4 或者 8*8 的单组数据，这样使得测试的数据较少。后面我们改为 Test bench 里面读取测试文件的方法，尽可能多地完成相关测试。这里面主要的问题在于 Verilog 读取文件并转化为相关的数值。

2) Design Compiler 最长路径优化

在 Design Compiler 后的时序报告中，我们了解到了我们设计的最长路径主要集中在乘法器的设计（移位和加法）。主要原因在于我们的相关乘法的设计为纯组合逻辑，导致其为最长路径，延时较大。这里我们采用将乘

法器的设计（移位和加法）中添加流水线，缩短最长路径的延时。

3) RTL 逻辑仿真时序问题

在最初的设计中，我们也遇到了相关的结果错误的问题。主要包括逻辑的中时序不对。另一方面，我们中间为了减小最长路径的延时，添加了很多的流水线，导致时序错误较多，最终在波形对比和输出的数据对比中发现问题予以改正。

同样的，在逻辑综合、物理实现上也遇到了些问题，主要集中在软件的使用上，同时由于对软件的使用不太熟悉，导致在 DC 逻辑综合、ICC 物理实现上以参考教程和例程为主。

6.2.2 实验总结与感想

本次设计中，采用 Verilog 完成了兼容 4*4 点和 8*8 点 IDCT 的 RTL 设计。实现电路 RTL 级设计后，在 SMIC 0.18um 的工艺库物理实现下，对所设计的 IDCT 电路做了功能仿真、时序分析、逻辑综合和自动布局布线设计。经过对功能仿真和逻辑综合结果的分析，本文的设计可以在 100MHz 频率下实现每秒 30 帧 HDTV 灰度图像(1920×1080@30fps)的实时处理能力，平稳运行时，每秒中输入 16*100M Bits，每秒钟输出 8*100M Bits。逻辑综合和布局布线后的结果表明，本文设计的电路芯片面积为 1332*1332 平方微米，IDCT 电路逻辑占芯片总面积的 60%，功耗约为 42mW。

在本次的数字集成电路课程设计中，学到了很多知识。从 Verilog 编写 RTL 可综合电路，到使用 DC、ICC 工具完成逻辑综合、物理实现等数字芯片设计的重要过程，掌握了数字芯片设计的基本流程，对以后的工作和学习也是有很大的帮助。学习到了 IDCT 在 HEVC 等数字视屏图像处理的用途，对相关的领域知识有所了解。在实验中也遇到了这样那样的问题，从对软件的基本使用，到设计优化的方案，何老师和主要都给予了很多的帮助，在这里对他们表示衷心的感谢！