

## 摘 要

离散余弦变换(DCT)及其反变换(IDCT)在图像编解码方面应用十分广泛,至今已被JPEG、MPEG1、MPEG2、MPEG4和H26x等国际标准所采用。由于其计算量较大,软件实现往往难以满足实时处理的要求,因而在很多实际应用中需要采用硬件设计的DCT/IDCT处理电路来满足我们对处理速度的要求。本文所研究的内容就是针对图像处理应用的 $8 \times 8$ 二维DCT/IDCT处理核的硬件实现。

本文首先介绍了DCT和IDCT在图像处理中的作用和原理,详细说明了DCT变换实现图像压缩的过程,并与其它变换比较说明了用DCT变换实现图像压缩的优势。接着,分析研究了DCT的各种快速算法,总结了前人对DCT快速算法及其VLSI实现所做的研究。根据图像处理的特点和基于IP复用的设计思想,结合DCT快速算法和硬件实现的特点,以提高速度、减少设计面积和功耗为目标,本文给出了一种DCT/IDCT的硬件设计方案。该方案利用DCT的行列分离特性,采用流水线设计技术,将二维DCT/IDCT实现转化为两个一维DCT/IDCT实现。在一维DCT/IDCT设计中,利用了DCT余弦因子预先知道和其具有的旋转特性,采用移位和加法逻辑来实现乘法运算,从而避免了采用乘法器设计所造成的资源和面积的浪费,同时提高了运算速度。最后,对所设计的DCT/IDCT处理核进行了综合和仿真验证,结果表明所设计的DCT/IDCT处理核能够在100M时钟频率下正确完成 $8 \times 8$ DCT或IDCT的逻辑运算。采用0.35  $\mu\text{m}$ 工艺库综合后除所用 $64 \times 15$ RAM外,其余部分逻辑的面积为 $55477.90625 \mu\text{m}^2$ 。

关键词: 离散余弦变换    离散余弦反变换    IP    VLSI    流水线

## Abstract

Discrete Cosine Transform and Inverse Discrete Cosine Transform are widely used in coding and decoding of image processing. So far, it has been adopted by many international standards like JPEG, MPEG1, MPEG2, MPEG4 and H26x etc. It's difficult to satisfy the requirement for real-time by software owing to its heavy quantity of computing. Therefore, we used to adopt hardware circuits to satisfy our requirement for speed in many practical applications. The content of the thesis researching is that hardware implementation of  $8 \times 8$  2-D DCT/IDCT IP core for application of image processing.

The thesis introduced principle and effect of DCT/IDCT in image processing at first, and further elaborated the advantages of DCT transform. And analyzed and studied various fast algorithms of DCT, summarized the working for DCT fast algorithms and its VLSI implementation which has been done by previous researchers. Based on characteristics of image process and IP design idea, combining DCT fast algorithms and hardware implementation, on the purpose of improving speed and decreasing area and power, the thesis presented a way of DCT/IDCT hardware implementation. The way adopted pipeline architecture and changed 2-D DCT/IDCT to two 1-D DCT/IDCT based on characteristic of row-column decomposition. In the design of 1-D DCT/IDCT, since we knew the value of DCT cosine coefficients, by making use of its rotation characteristic, multiplication function can be designed by shift and addition logic instead of direct multiplication unit by which can save design resources while improve the speed. At last, synthesis and verification for design has been done. The result shown that the design could complete the function of  $8 \times 8$  DCT or IDCT under 100M clock frequency. Except  $64 \times 15$  RAM which design needed, other logic area is about  $55477.90625 \mu m^2$  after synthesis by 0.35 technology library.

**Key words:** Discrete Cosine Transform    Inverse Discrete Cosine Transform    IP  
VLSI    Pipeline

---

# 华中科技大学硕士学位论文

---

## 独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律效果由本人承担。

学位论文作者签名：

薛峰

日期：2004年5月9日

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 ☐

在\_\_\_\_\_年解密后适用本授权书。

本论文属于

不保密 ☒

(请在以上方框内打“√”)

学位论文作者签名：

薛峰

日期：2004年5月9日

指导教师签名：

沈伟

日期：2004年5月9日

## 1 绪 论

### 1.1 研究背景及意义

娱乐、电信、因特网——这些都是我们日常生活的一部分，在生活中我们享受它们，使用它们，在报纸和杂志上读到它们，在电视上看到它们，并在这些东西上花钱，但是我们已经拥有收音机、电视机、音响、电话几十年了。现在有什么新东西呢？那就是“数字化”。为什么要数字化呢？因为数字信号易于存储和远距离传输，而且没有累积失真，数字化的存储信息可高品质地被还原。随着科技的飞速发展，我们正在进入数字化时代。数字化后的信息，尤其是数字化后的视频和音频信息具有数据海量性，如声音、音乐、电视、电影这些重要的信号数字化后，每秒需要更多的比特数去存储或传输，这样就造成了高成本。它给信息的存储和传输造成较大的困难，成为阻碍人类有效地获取和使用信息的瓶颈问题之一。因此，作为数字化技术核心之一的数字压缩备受人们关注<sup>[1]</sup>。

对于图像数据而言，一个显著特点就是信息量大，根据计算，在不进行编码压缩处理的情况下，一张 600M 的光盘仅能存放 20s 左右的  $640 \times 480$  像素的图像信息，这就使得在对图像的存储和处理中所遇到的困难和所需成本是非常高的<sup>[2]</sup>。同时，在现代通信中，图像传输已经成为重要内容，传输信息量的大小是影响传输速度的重要原因之一。要想提高通信速度，一个必要的手段就是采用图像的编码压缩技术，减少传输的数据量。组成图像的各像素之间，无论是在行方向还是在列方向上，都存在着一定的相关性。应用某种编码方法提取或者减少这种相关性，就可以达到压缩数据的目的。从信息论的观点看，描述图像信源的数据是由有效信息量和冗余量两部分组成，去除冗余量能够节省图像存储和传输中的开销，同时又要不损害图像信源的有效信息量，保证重构图像的质量。所谓图像压缩编码技术就是对要处理的图像源数据按一定的规则进行变换和组合，从而达到以尽可能少的代码(符号)来表示尽可能多的数据信息<sup>[3][4]</sup>。

离散余弦变换 (DCT) 能有效的消除图像的空域冗余，因而被广泛应用于图像

编码压缩中, 至今已被JPEG、MPEG1、MPEG2、MPEG4和H26x等国际标准所采用<sup>[5][6]</sup>。但由于其运算量较大, 为了满足实时要求, 常常采用硬件来实现其功能。而DCT所具有的快速算法也为DCT的硬件实现提供了便利。尽管DCT/IDCT快速算法很早就被提出, 并已被研究和应用了几十年, 但二维 $8 \times 8$ DCT快速算法, 仍然还未十分成熟, 现在的算法所要求的乘法次数还远高于理论值。目前, 还在不断的有新的快速算法被提出和应用, 对DCT算法的研究仍然是当今图像压缩编码研究的重点之一。在考虑DCT/IDCT的VLSI实现时, 根据不同的应用目的, 结合不同的算法, 有不同的实现方式。集成电路设计技术发展到今天, 逻辑功能的实现已经不是很大的挑战了, 而怎样在最短的时间设计出面积小, 速度快, 功耗低的电路才是最大的挑战。所以, 即便采用同一算法, 但在设计方法、工艺技术上不同, 设计出的电路的性能也会大相径庭。因此, 对DCT/IDCT的硬件实现的研究仍是十分有意义的。

## 1.2 国内外研究情况

DCT首先由N.Ahmed等人于1974年提出<sup>[7]</sup>。之后, 许多快速算法被提出。1977年, W.H.Chen等人根据变换矩阵具有对称性, 第一次用稀疏矩阵分解法得到DCT-II快速算法。1987年, 几乎同时有三篇论文论述了DCT的递归算法, 其中H.S.Hou的快速算法较具有代表性。森川良孝采用切比雪夫多项式同余形式提出了2-D DCT的直接算法。N.L.Cho的快速算法采用三角函数法, 将长度为2的n次幂的2-D DCT表示为两个新的二维变换之和, 再利用换序移位和附加实数加法运算。

九十年代以来, 随着JPEG、MPEG、HDTV的发展, DCT/IDCT的VLSI实现也有了很大进展, 各种快速算法的提出及工艺的改进也为DCT/IDCT结构的发展创造了条件。现在, DCT的VLSI实现主要还是基于行-列分解法, 因为此方法有较简单的数据传递结构。目前DCT中的乘法器多采用DA算法实现, 因其精度高, 结构规则, 占用面积相对较少。但随着工艺的改进, 特别是在深亚微米级, 同一硅片将集成较以往更多的晶体管, 因此晶体管数这一芯片性能参数的重要性有所降低, 脉动阵列结构有了一定的发展。总之, 目前DCT新算法正在被不断的提出, 相应的新的VLSI实现方法也在不断产生。以下列出了当今已实现的几种典型 $8 \times 8$ 结构DCT芯片的特性<sup>[8]</sup>。



表1-1 几种典型8×8结构二维DCT芯片特性列表

设计者	工 艺	芯片面积	晶体管数	时钟频率	备注
A.Madisetti	0.8 $\mu$ m CMOS triple metal	10mm <sup>2</sup>	67,000	100MHz	基于乘法器, 只用 一个1-D DCT处 理单元
K.Kchau	0.8 $\mu$ m BiCMOS gatearray	6.4×3.37mm <sup>2</sup>	78, 666	100MHz	基于DA算法
J.C.Carlach	CMOS 1.2 $\mu$ m	26mm <sup>2</sup>	50, 000	27MHz	基于DA算法, 改 进转换电路
S.Uramoto	0.8 $\mu$ m double metal CMOS	6.4×3.3mm <sup>2</sup>	102, 000	100MHz	基于DA算法, 双 层/列交叠ROM 技术
M.Matsui	0.8 $\mu$ m base-rule doublemetal CMOS	3.85×3.45mm <sup>2</sup>	120, 000	200MHz	基于DA算法, SA-F/F结构
T.Masaki	0.6 $\mu$ m CMOS triplemetals	17mm <sup>2</sup>	210, 345	102MHz	改进的DA算法, 只用一个1-D DCT处理单元
C.-L.Wang	0.8 $\mu$ m CMOS	6.95×6.9mm <sup>2</sup>	108, 363	33MHz	基于DCT定义的 脉动结构
C.-M.Wu	0.8 $\mu$ m single-poly double metal CMOS	4.75×5.269mm <sup>2</sup>	65, 920	15.6MHz	基于Lee算法脉动 整列, 只用一个1 -D DCT处理单 元
V.Srinivasan	0.8 $\mu$ m BiCMOS gatearray	14.7×16.2mm <sup>2</sup>	320, 000	50MHz	网格DCT结构算 法, 帧递归

## 1.3 本论文的研究内容

本文的内容安排共分为六个章节, 各章节内容如下:

第一章: 绪论, 也就是本章。主要介绍了本课题研究的目的是与意义、国内外研究状况以及本文的研究内容和安排等;

第二章: DCT在图像处理中的作用及IP设计技术。本章主要介绍图像压缩的相关知识、DCT的数学意义、DCT在图像处理中的作用和工作原理。另外, 还对ASIC设计方法和IP设计技术做了简要介绍。

# 华中科技大学硕士学位论文

---

第三章：DCT/IDCT 的算法及结构设计。本章主要研究了 DCT/IDCT 的各种一维和二维快速算法及其 VLSI 实现结构。首先对几种流行的算法结构做了介绍，分析了各种算法结构的优缺点，介绍了流水线的结构设计相关知识，然后给出了自己的设计结构，并做了详细介绍和说明。

第四章：DCT/IDCT 的逻辑设计。本章主要研究 DCT/IDCT IP 核的 VLSI 逻辑实现，对设计的各部分逻辑做了详细的设计说明，并对设计中应该注意的问题做了详细介绍和说明。

第五章：综合和验证。本章对所设计的 DCT/IDCT IP 核做了功能仿真，综合和门级仿真验证，对仿真和综合过程做了详细说明，给出了仿真和综合结果，并对结果进行了分析。

第六章：总结。本章对所做的研究工作做了概括和总结。

## 2 DCT在图像处理中的作用及IP设计技术

### 2.1 DCT 介绍

离散余弦变换（DCT）实际上是离散傅立叶变换（DFT）的一种特殊形式。傅立叶级数展开式中，如果被展开的函数是实偶函数，其傅立叶级数中只包含余弦项，由此可导出DCT变换。假如已知函数 $f(x)$  ( $x \geq 0$ )，并非实偶函数，人为的把它对称扩展到 $x < 0$ ，构成实偶函数 $f'(x)$ ，那么 $f'(x)$ 的傅立叶变换的正弦项被抵消，余弦项是 $f(x)$ 傅立叶变换中余弦项的两倍<sup>[9]</sup>。

### 2.2 DCT/IDCT 在图像处理中的作用

DCT变换的目的是消除图像的空域冗余，实现图像的压缩。离散余弦变换DCT可被看作为一个谐波分析仪，把离散余弦反变换IDCT看作一个谐波合成器。每个 $8 \times 8$ 二维源图像采样数据块，实际上是64点离散信号，该信号是空间二维参数 $x$ 和 $y$ 的函数。IDCT把这些信号作为输入，然后把它分解成64个正交基信号，每个正交基信号对应于64个独立二维（2D）空间频率中的一个，这些空间频率是由输入信号的频谱组成。IDCT的输出是64个基信号的幅值，或称DCT系数，每个系数值由64点输入信号唯一地确定，即离散余弦变换的变换系数。在频域平面上变换系数是二维频域变量 $u$ 和 $v$ 的函数。对应于 $u=0, v=0$ 的系数，称为直流分量，即DC系数，其余63个系数称作AC系数，即交流分量。因为在一幅图像中像素之间的灰度或色差信号变化缓慢，在 $8 \times 8$ 子块中像素之间的相关性很强，DCT变换实现图像数据压缩正是利用图像相邻像素点之间相关性很强的特点（这表明相邻像素点之间的像素差很小），把图像从时域变换到频域，这样，变换后的矩阵表示的不再是图像的像素值，而是图像的频率分布。通过离散余弦变换处理后，DC系数代表了图像的平均能量，即变换前64个图像像素的平均值；而AC系数，代表图像的各个频率成分，离DC系数越远的AC系数代表越高的频率成分。如图2-1所示，变换后的矩阵数值主要集中在矩阵的左上角，而越靠近右下角的数据数值越小。它所代表的物理意义是图像的能量主



要集中在图像的低频部分，而高频分量则很小。根据视觉特性，人的眼睛对图像的高频成分不敏感，这就大大的方便了我们进行数据压缩。对变换后的各个数据根据其在矩阵中的不同位置，除以不同的数值，越靠近矩阵右下角的代表高频成分的数据，所除的值越大，这个过程叫做量化，正是量化这个过程会造成图像信息的丢失（即失真），因为图像的高频成分被忽略了。经过量化后的数据，就会在矩阵右下角出现很多连续的0，这就可以方便的进行行程编码和霍夫曼编码，实现对图像数据的压缩。

183	160	94	153	194	163	132	165	313	56	-27	18	78	-60	27	-27
183	153	116	176	187	166	130	169	-38	-27	13	44	32	-1	-24	-10
179	168	171	182	179	170	131	167	-20	-17	10	33	21	-6	-16	-9
177	177	179	177	179	165	131	167	-10	-8	9	17	9	-10	-13	1
178	178	179	176	182	164	130	171	-6	1	6	4	-3	-7	-5	5
169	180	180	179	183	169	132	169	2	3	0	-3	-7	-4	0	3
179	179	180	182	183	170	129	173	4	4	-1	-2	-9	0	2	4
180	179	181	179	181	170	130	169	3	1	0	-4	-2	-1	3	1

a. 变换前的图像像素矩阵      图2-1 DCT变换      b. 变换后的图像频率矩阵

解码过程中离散余弦变换IDCT是DCT的逆过程。假如对变换系数不进行量化处理，那么64个DCT变换系数经过逆变换，重建64点的图像数值，与源图像输入值完全一致。也就是说，如果DCT/IDCT变换的计算精度足够高，并且DCT变换系数没被量化，那么原始的64点信号就能精确地恢复。理论上DCT变换不会使源图像信息丢失，它仅仅是把图像从时域变到频域，以便能更有效地编码。

由于DCT和IDCT计算公式中，都包含了超越函数，因此，没有一种物理工具或计算手段能以很高的精度计算它们。甚至在DCT输入和输出精度一定的条件下，以独立的设计来实现相同的DCT和IDCT算法时，在精度上也会有细微的差别，这些差别反映在余弦项或中间结果的表示，求和方法以及截尾或四舍五入数值上，这些因素都会使完全相同的输入，最终产生稍微不同的输出。

计算DCT/IDCT的方法很多，JPEG、MPEG等标准没有规定唯一的DCT或IDCT算法。但是提出把精度测试作为基于DCT/IDCT编码器和解码器一致性测试的要求，这样来保证克服使图像质量降低的余弦变换误差。

当今流行的失真图像压缩算法有三种: 离散余弦变换(DCT, Discrete Cosine Transformation) 压缩、分形(Fractal)压缩和小波变换(Wavelet Transformation) 压缩。在所有的正交变换编码中,DCT变换是仅次于KLT变换(无失真变换)的次最优变换。自从离散余弦变换(DCT)被提出以来,DCT已被广泛应用在数字信号处理中,特别是语音及图像数据压缩,自适应滤波以及通信系统等领域,目前基于 $8 \times 8$ 的二维DCT/IDCT的编/解码系统已被图像压缩标准JPEG、MPEG和H261等标准所采纳,并已用在HDTV中。JPEG是ISO和CCITT 联合制定的适用于连续色调、多级灰度、彩色和单色静止图像压缩的国际标准。JPEG的核心是DCT编码技术。对静止图像的压缩推荐使用的方法是将图像先分割成 $8 \times 8$  大小的方块,然后再对其变换、量化和编码,并提供了符合人的视觉变化的量化表。编码方法主要利用Huffman编码及算术编码。MPEG标准是运动图像压缩国际标准,到目前为止,已有MPEG1、MPEG2和MPEG4三个标准。MPEG是对运动图像的压缩,主要采用帧内编码与帧间编码相结合的方法,帧内编码技术与JPEG相同,帧间编码采用运动补偿技术,即预测和内插两种方法。在选取均方差准则下,Karhunen-Loeve变换(KLT)是信号处理中的最佳变换,但KLT没有快速算法,且计算十分困难,所以不具有实用价值,但可以用它作为衡量其他变换的标准。在有快速算法的次最佳变换中,DCT基向量最接近KLT,其编码技术已趋成熟,因而发展很快,应用广泛<sup>[10]</sup>。另外,因为DCT具有快算算法,可分离特性和中等的算法复杂度三个特点,十分适合于VLSI实现<sup>[11]</sup>。

## 2.3 IP 设计技术

随着IC设计方法学的进步,微电子工艺技术的不断发展,ASIC设计所要求的时间日益缩短。集成电路设计技术已逐渐从以通用芯片设计为主转向以专用芯片为主,以普通规模专用芯片为主转向超大规模的系统芯片为主,从基于时序驱动的数字IC设计方法转向基于IP的数字IC设计方法。

### 2.3.1 ASIC 设计流程

专用集成电路产品开发,最初总是从市场上得到需求信息或产品概念,根据这些概念需求,逐步完成IC规格定义和总体方案设计。总体方案定义了芯片的功能和

模块划分, 定义了模块功能和模块之间的时序关系等内容。

在总体方案经过充分讨论或论证后开始 ASIC 产品的开发。IC 的开发阶段包含了 RTL 代码设计、功能仿真、综合、DFT、形式验证、静态时序分析 (STA)、自动布局布线 (APR)、版图设计和版图后时序验证等过程。如图 2-2 所示, 其中的时序验证包含动态时序验证 (Simulation) 和静态时序验证 (STA)。

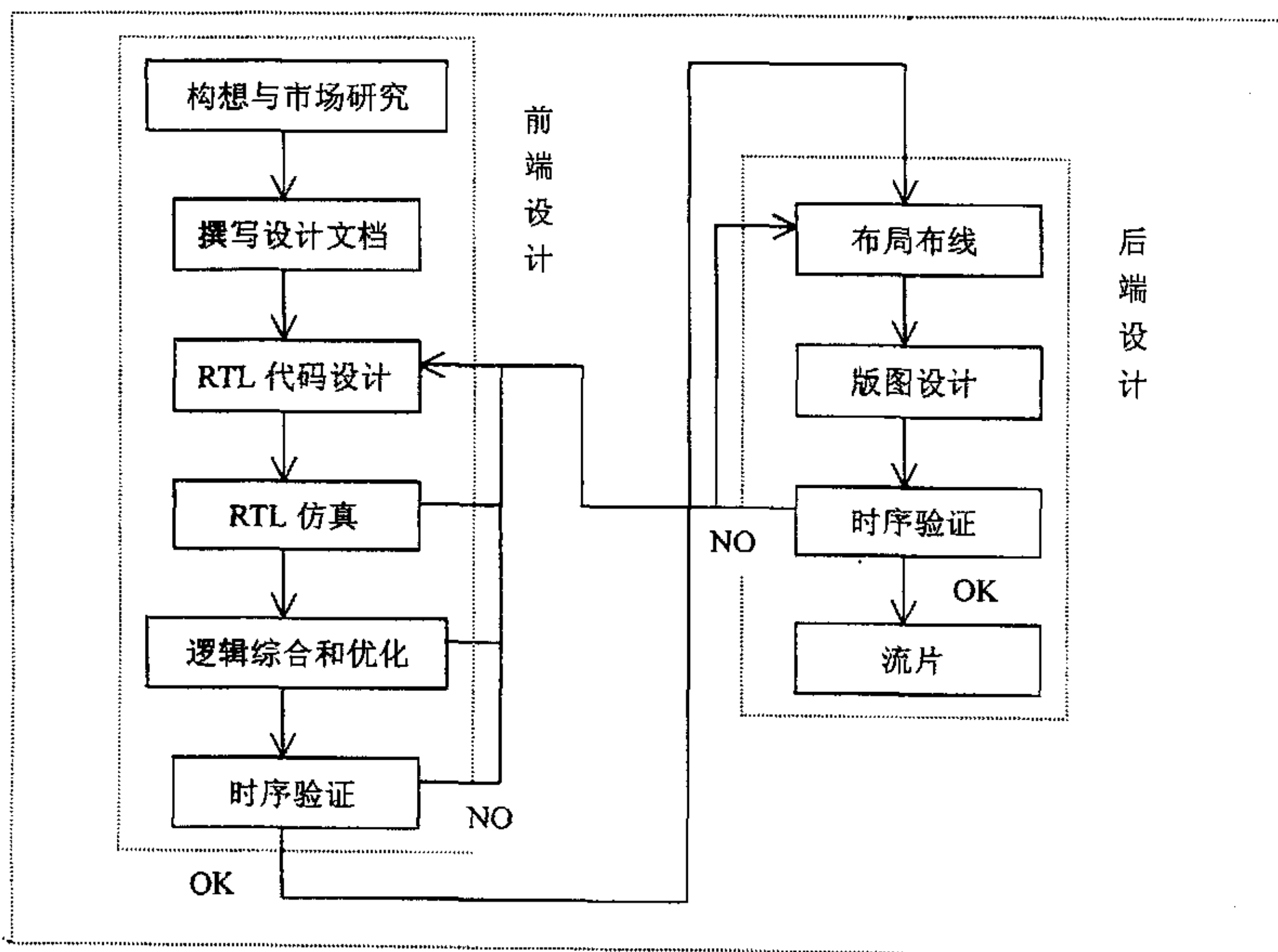


图 2-2 ASIC 设计流程简图

IC 设计开发过程是一个迭代的过程, 在设计的每一个环节完成后都要进行验证, 如果不满足要求就需要返回到上一个步骤, 甚至更前面的步骤。例如, 在综合后的门级仿真或静态时序分析中, 发现时序不能满足要求, 我们就需要从新改变综合约束条件重新综合, 如果不行, 还要返回到 RTL 代码设计阶段来修改 RTL 代码, 然后再继续 RTL 仿真、综合、然后再验证, 如此反复迭代, 直到验证通过, 才能再进行下一环节的设计。

## 2.3.2 IP 软核设计方法与技术

IP技术是针对可复用的设计而言的,其本质特征是功能模块的可复用性。IP通常必然满足以下三个基本特征:一是通用性好,二是正确性有100%的保证,三是可移植性好。通用性好是指IP的功能在某一应用领域广泛通用。正确性有100%保证是指IP的实现严格遵守一系列的可复用设计开发规范,IP的验证用例具有完备性,功能覆盖率、测试覆盖率都能达到100%。可移植性好是指IP的实现如行为描述、网表、版图文件具有可移植性,有规范通用的接口设计。

IP 核的开发与设计与 ASIC 设计开发的流程基本一致,但其也具有自己的特点。IP 软核的开发可以满足芯片规模要求越来越大,设计周期要求越来越短的需求,其特征是 IC 设计中的 IP 功能模块的复用和组合。可复用 IP 模块首先必须是一个良好的模块设计,必须严格遵守一系列的设计规则。概括起来是一个基本准则和三个基本规则。

一个基本准则是:局部问题局部解决。因为当设计规模越来越大时,局部模块设计的缺陷如留到系统验证阶段,就会变得非常棘手,难以发现和修正。

三条基本规则:一是尽可能采用全同步的设计策略,对模块的输入和输出信号用寄存器锁存以达到时序优化问题局部化的目的;二是严格遵循自底向上的验证策略,保证每个功能模块在被集成到更高层次之前得到最充分的验证,以达到功能验证问题局部化的目的;三是在着手进行详细设计之前必须做好总体方案的设计和论证。

IP 软核的开发流程包括两条主线:IP 设计和 IP 验证。IP 设计流程一般可划分为确定规格和 IP 宏模块的总体设计,子模块的定义和设计,IP 宏模块的集成化设计等三个阶段;IP 验证流程包括了建立参照模型,建立测试平台(testbench)和准备验证用例(testcase),回归测试,形式验证等。图 2-3 是 IP 模块的设计开发流程。

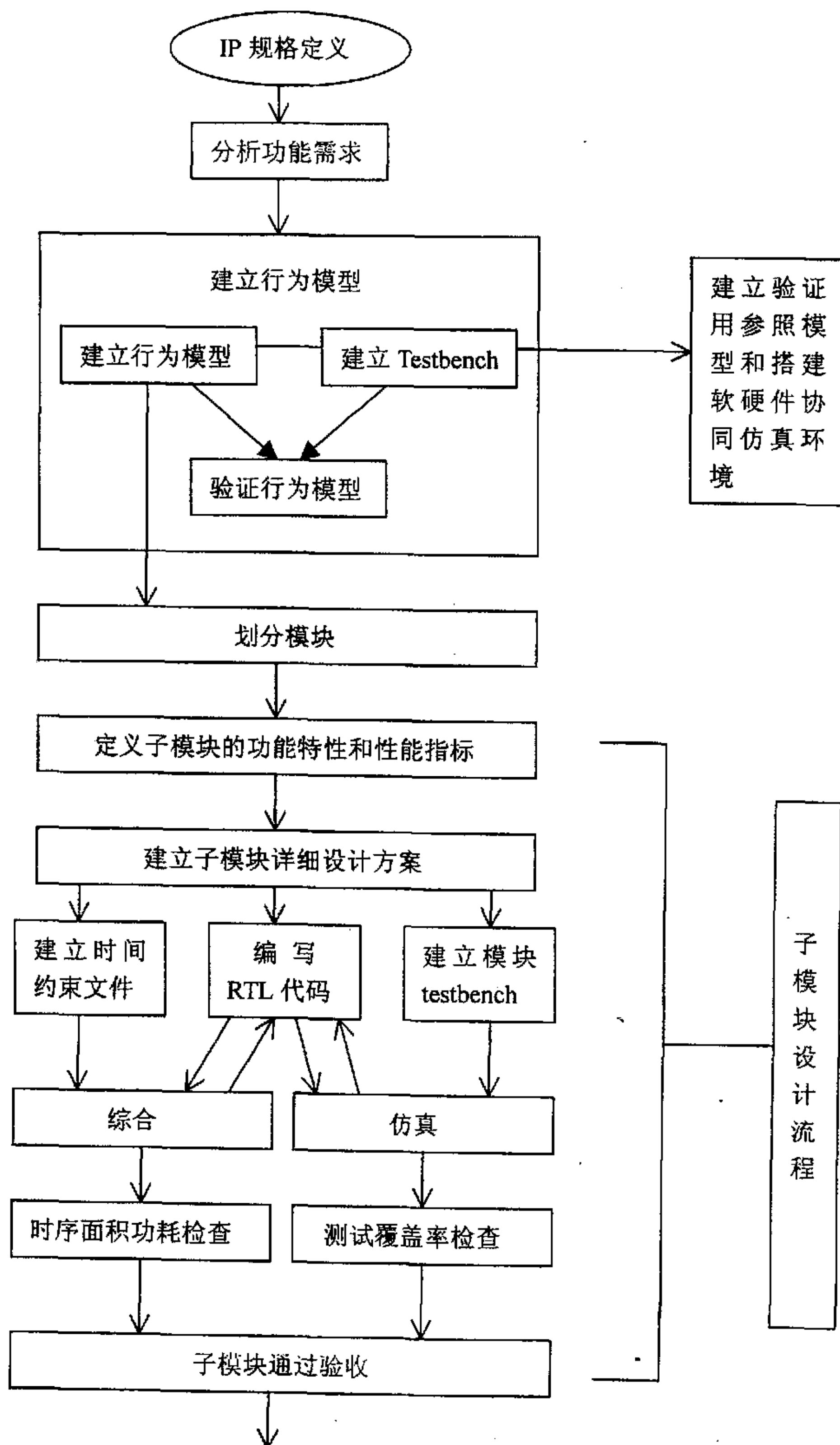


图 2-3 IP 模块化设计流程



### 3 DCT/IDCT IP核的算法及结构设计

#### 3.1 DCT 快速算法介绍

##### 3.1.1 一维离散余弦变换(1-D DCT)快速算法

DCT 首先由 N.Ahmed 等人于 1974 年提出, 设数据序列  $X(n)$ ,  $n=0,1,\dots,N-1$ , 则  $X(n)$  的 DCT 定义为:

$$Y(k) = \sqrt{\frac{2}{N}} \varepsilon(k) \sum_{n=0}^{N-1} X(n) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \quad k=0,1,\dots,N-1 \quad (3-1)$$

当  $k=0$ ,  $\varepsilon(k) = \frac{\sqrt{2}}{2}$ ; 当  $1 \leq k \leq N-1$  时,  $\varepsilon(k) = 0$ ;  $Y(k)$  是第  $k$  个 DCT

系数。若写成矩阵形式, 则有:  $X = [C_n]x$ , 其中  $[C_n]_{n,k} = \sqrt{\frac{2}{N}} \varepsilon_k [\cos\frac{(2n+1)k\pi}{2N}]$  表示 DCT 变换矩阵。

离散余弦逆变换 (IDCT) 定义为:

$$X(n) = \sqrt{\frac{2}{N}} \varepsilon(k) \sum_{k=0}^{N-1} Y(k) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \quad n=0,1,\dots,N-1 \quad (3-2)$$

1984 年, 王中得根据不同的变换矩阵将 DCT 分成了四类<sup>[12]</sup>。为了有效地计算 DCT, 许多快速算法被提出。一般 DCT 快速算法分为两种: 间接算法与直接算法, 两种算法都是集中在蝶形结构上, 且目的都是为了减少乘法及加法运算量。

间接算法是利用 DCT 和 DFT、DHT 等正交变换之间的关系, 用 DFT 或 DHT 快速算法来计算 DCT。间接算法过程简单, 主要工作是处理算法间的转换, 因此往往需要加一些额外的操作步骤。由于将其它变换的快速算法应用在 DCT 中总有其自身的局限性, 所以目前在算法上已很少有人用间接算法来计算 DCT 了。

直接算法包括 DCT 变换矩阵分解, 递归算法两种技术, 不同之处在于矩阵分解是利用稀疏矩阵分解法将变换矩阵分解, 而递归算法是由较低阶 DCT 矩阵递归产生较高阶 DCT 矩阵, 可以说递归算法是分解算法的逆算法, 但递归算法较矩阵分解算

法有较好的数值稳定性。1977 年, W.H.Chen 等人根据变换矩阵具有对称性, 第一次用稀疏矩阵分解法得到 DCT-II 快速算法<sup>[13]</sup>。这种算法不仅结构简单, 且使用余弦系数做乘子, 因而比前述各算法均优越, 但其实现结构复杂。1987 年, 几乎同时有三篇论文论述了 DCT 的递归算法<sup>[14][15][16]</sup>, 其中 H.S.Hou 的快速算法较具有代表性。近几年出现的新算法大多是这几种算法的改进。

Feig-Winograd 通过把 DCT 看作是一种循环旋转运算, 证明了在有理数域上 8 点 1-D DCT 变换所需的最小实数乘法次数为 11 次<sup>[17]</sup>。Loeffler 的 DCT 快速算法达到了这一极限, 它是将 DCT 运算转为旋转运算<sup>[18]</sup>。

### 3.1.2 二维 DCT 快速算法

二维 DCT 定义为:

$$Y(k, l) = \frac{2}{N} \varepsilon_k \varepsilon_l \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} X(n, m) \cos\left[\frac{(2n+1)k\pi}{2N}\right] \cos\left[\frac{(2m+1)l\pi}{2N}\right]$$

$$k, l = 0, 1, \dots, N-1 \quad (3-3)$$

二维 IDCT 定义为:

$$X(n, m) = \frac{2}{N} \varepsilon_k \varepsilon_l \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} Y(k, l) \cos\left[\frac{(2n+1)k\pi}{2N}\right] \cos\left[\frac{(2m+1)l\pi}{2N}\right]$$

$$n, m = 0, 1, \dots, N-1 \quad (3-4)$$

其中, 当  $k=1$  时,  $\varepsilon_k = \frac{\sqrt{2}}{2}$ , 当  $l=1$  时,  $\varepsilon_l = \frac{\sqrt{2}}{2}$ , 其它情况,  $\varepsilon_k = \varepsilon_l = 1$

2-D DCT 快速算法主要有两种: 行-列分解法<sup>[19]</sup> (RCM) 及非行列分解法 (NRCM), RCM 方法是将  $N \times N$  数据按行 (或列) 方向进行  $N$  个 1-D DCT 计算, 产生中间矩阵, 然后对中间矩阵按列 (或行) 方向再进行  $N$  个 1-D DCT 计算, 最后得到 2-D DCT 结果。

NRCM 方法即直接分解法, 典型的做法是 2-D 矢量基 DCT 算法。2-D 矢量基算法主要有基于 B.G.Lee 的算法和基于 Hou 的算法。对于较长长度的 DCT, 基于 Hou 的算法较基于 Lee 的算法性能优越。目前最有效的 2-D DCT 直接分解算法主

要有森川良孝等人的与 N.L.Cho 等人的算法。这两种算法都将算法运算量减至传统行列分解法的 50%。森川良孝算法采用切比雪夫多项式同余形式。把 2-D DCT 转换为  $N$  个  $N$  点 1-D DCT 和一个长度为  $N$  的切比雪夫多项式变换。N.L.Cho 算法<sup>[20]</sup>采用三角函数法,将长度为 2 的  $n$  次幂的 2-D DCT 表示为两个新的二维变换之和,再利用换序移位和附加实数加法运算。Fei-Winograd 证明了在有理域上计算长度为 2 的  $n$  次幂的 2-D DCT 所需最小实乘次数为  $2^{2n+1} - n2^n - 2^{n+1}$ , 现在还无法达此极限。

## 3.2 流水线设计技术

### 3.2.1 流水线设计的概念

所谓流水线设计实际上就是把规模较大、层次较多的组合逻辑电路分为几级,在每一级插入寄存器暂存中间数据。K级流水线就是从组合逻辑的输入到输出恰好有 K级寄存器组(分为K级,每一级有一个寄存器组),上一级的输出是下一级的输入,结果从流水线的最后一级送出,无反馈电路。流水线执行完成一种固定的功能<sup>[21]</sup>。

### 3.2.2 流水线技术的特点和优势

流水线的设计方法已经在高性能的、需要经常进行大规模运算的系统中得到广泛的应用,如 CPU等。目前流行的 CPU,如 Intel的奔腾处理器在指令的读取和执行周期中充分地运用了流水线技术以提高它们的性能。采用流水线设计的最大优势在于它能提高吞吐量。在ASIC设计中,流水线结构是提高组合逻辑电路吞吐量从而增强计算性能的一个重要办法<sup>[22]</sup>。当用 Verilog HDL描述流水线结构的部件时,要使用结构描述,才能真正综合成设计者想要的流水线结构。简单的运算符表达式只有在综合库中存有相应的流水线结构的宏库部件时,才能综合成流水线结构从而显著的提高运算速度。

## 3.3 总体结构设计

本文的 DCT/IDCT IP 核设计研究是针对图像数据压缩应用的。在 MPEG, JPEG 等标准中,数据以码流的形式被处理。为了提高编解码的处理效率,节约硬件资源,

流水线的设计是非常重要的。对于 JPEG 来说, 图像的每一帧都是帧内编码图像(I 帧), 所以图像的 DCT 变换的输入是  $0 \sim 255$  (象素点灰度级), 而输出是  $-1024 \sim 1023$ , 相应的需要 8 位的输入和 11 位的输出, 对于 IDCT, 输入和输出相反; 对于 MPEG 来说, 要处理的不单有帧内编码的 I 帧图像, 还有帧间编码的 P 帧图像和 B 帧图像, P 帧和 B 帧图像表示的是两帧图像象素点的差值, 所以其 DCT 变换的输入值表示范围为  $-255 \sim 255$ , DCT 变换输出值范围为  $-2000 \sim 2000$ , 相应的需要 9 位和 12 位来表示, IDCT 则相反<sup>[23]</sup>。本文的设计研究是按 MPEG 的标准进行的, 同时也兼容 JPEG 标准要求。

对于数码相机一类的应用, 既需要进行图像编码, 又需要进行图像解码, 也就是说需要在编码的时候用到 DCT 变换, 在解码的时候用到 IDCT 变换。DCT 变换和 IDCT 变换, 在算法和方法上都是一样的, 是完全可逆的, 因此可以将 DCT 变换和 IDCT 变换合在一起设计成一个 IP 核, 以节约硬件资源。

对于 2-D DCT 的结构, 也分为行列分解法 (RCM) 及非行列分解法 (NRCM) 两类。对于 2-D DCT NRCM, 采用的算法是各种非行列分解 2-D DCT 快速算法, 它往往要求整个  $N \times N$  输入数据同时参与计算, 因此 I/O 处理及数据传递电路复杂, 使得它们的 VLSI 实现在性能上不如 RCM 系统。另外, 因为图像数据的编解码过程往往是以数据流的形式进行, 为了实现流水线的设计和操作, 我们并不需要所有数据同时输入参与 DCT/IDCT 运算。那样做, 既需要设计串并转换电路, 而且还增加延迟, 不利于流水线的设计。因此, 现在行-列分解法 (RCM) 仍被广泛应用在 2-D DCT 芯片设计中。本文也是采用 RCM 的设计结构, 其结构如图 3-1 所示, 此结构采用的算法为各种一维 DCT 快速算法, 行和列的 1-D DCT 结构相同, 有时为了减少面积只用一个 1-D DCT 处理单元完成两次 1-D DCT 计算, 当然这将影响操作时间, 增加处理延迟, 为了获得较高的吞吐量, 本文的设计采用了两套 1-D DCT/IDCT 电路处理单元, 使其并行计算。对于所设计的 RCM 系统, 必须在两套的 1-D DCT/IDCT 处理电路之间对完成一次 1-D DCT/IDCT 运算的输出数据进行行-列转换, 这就需要设计中间数据转换电路 (TRAM), 所需要设计的转置 RAM 大小为  $64 \times 15\text{bits}$ 。

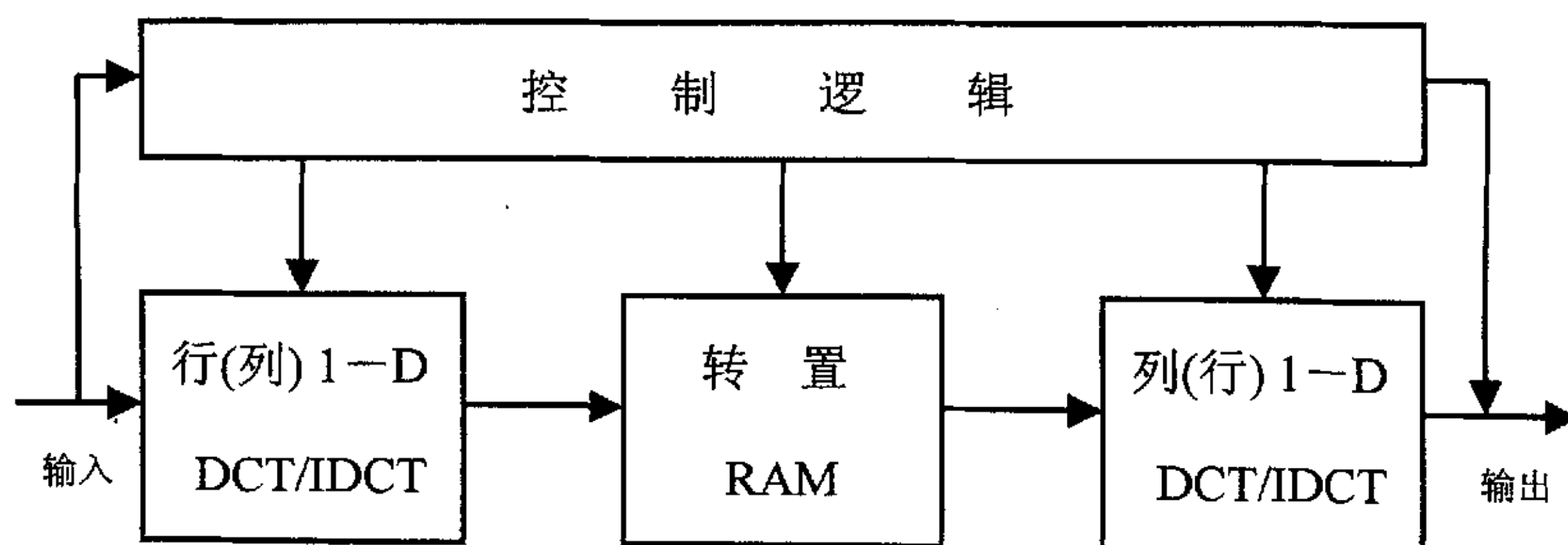


图 3-1 DCT/IDCT IP 核总体结构

### 3.4 一维 DCT/IDCT 结构设计

#### 3.4.1 算法结构介绍

一般一个算法结构的优劣主要决定于三个性能参数：面积、速度、精度，在此三个参数基础上，还要求结构规则，具有模块化，这样有利于设计及版图布局布线。现在已有多维 DCT 算法结构被提出，这些结构都有各自的缺点，因此在各种应用中这些结构并存。另外，因为每种 DCT 算法有自己的特殊性和应用领域，不是所有的算法都适合于 VLSI 实现。一个算法 VLSI 实现的有效性主要基于算法中算术单元之间的数据传递的复杂性，而不是算法的运算量。正如许多研究人员指出的那样，像很多 FFT 算法，它们有较低数量的乘法运算，但这些算法并不适合于 VLSI 实现。DCT 算法也遇到了同样的问题。当前的一维 DCT 算法结构主要有以下几类：

##### (1) 基于乘法器的 DCT/IDCT 结构

此结构多采用 W.H.Chen 的算法直接用乘法器来实现 DCT/IDCT 运算，改善此类 DCT/IDCT 结构的关键在于乘法器结构的改进。例如，采用 BOOTH 乘法算法的乘法器，与普通的乘法器相比，减少了乘法过程中的加法操作，提高了乘法运算速度，改善了 DCT/IDCT 的性能<sup>[24]</sup>。但众所周知，乘法器的引入也将导致面积的增加。另外，此类结构虽具有模块化，但结构不规则，不利于布局、布线。

##### (2) 基于分配算法的 DCT/IDCT 结构



基于分配算法的 DCT 结构由于其结构紧凑, 规则, 精度高, 速度快被广泛应用于各领域中。其基本思想是通过 ROM 查找表法, 利用 ROM 和累加器代替乘法器。它是利用 DCT/IDCT 中的余弦系数固定的特点, 将相应的输入数据的乘法结果余弦算好, 并存于 ROM 中, 根据输入数据对 ROM 取址即可<sup>[25][26]</sup>。由于分配算法具有高度的规则结构, 且因计算以分配方式进行, 截断被限制, 因而能获得较其它结构更高的精度, 很适合于 VLSI 实现。但大量 ROM 的使用同样大大增加了设计的面积, 而且对 ROM 的频繁访问和此算法结构采用的位串行实现, 同样极大的限制了速度, 难以获得很高的时钟频率。另外, 此结构要求的工艺复杂, 要同时用到三种工艺, 因此成本比较高。由于很多 FPGA 芯片本身有 ROM 模块, 而且 FPGA 时钟频率也不可能很高, 所以, 这种方法, 多用于 FPGA 的实现上。

### (3) 基于脉动阵列的 DCT/IDCT 结构

脉动阵列概念最早由 H.T.Kung 于 1982 年提出<sup>[27]</sup>。脉动阵列由一组简单重复的处理单元(PE)组成, 每个 PE 执行固定的、简单的操作, 每个 PE 只与相邻的 PE 有规则地联结, 操作时, 数据经过连成流水线的 PE, 沿途得到连续的有效处理。此高度并行处理结构, 大大提高了 VLSI 速度, 且结构规则。DCT 的脉动阵列种类多, 采用的算法也各异, 结构实现的主要工作是选取合适的 PE 单元<sup>[28][29]</sup>。

脉动阵列通过并行及流水操作可获得较高的处理速度, 且具有模块化、结构规则、局部内联等特点, 适合于 VLSI 实现。但脉动阵列往往要求多个数据同时参与计算, 因此 I/O 处理通常较为复杂。另外, 因每个 PE 内有多种算术运算、截断与舍入误差会影响精度。而且, 由于一个 PE 内多种运算并存, 有时会有多个乘法运算, 因而面积往往很大, 且脉动阵列结构的时钟控制电路复杂。这些原因使得现在各种应用中采用脉动阵列 DCT 结构的并不多见。

### (4) 基于 CORDIC 运算技术的 DCT/IDCT 结构

CORDIC 是坐标旋转数字计算机的同义词。这种计算是 Volder 于 1959 年提出来的<sup>[30]</sup>。这个概念虽然由来已久, 但它的实现与应用还在继续发展, 因为此技术简化了体系结构, 提高了速度, 降低了算术模块的功耗。

CORDIC 算法是一个通过以角度  $\alpha_j = \arctan 2^{-j}$  进行矢量旋转而得到的迭代过

程。通过采用 CORDIC 算法, 旋转运算中的乘法器可仅由加/减法和移位寄存器来实现, 减少了电路, 并通过流水线结构, 加快了数据传输速度。1990 年, Duh 与 Wu 首次将 CORDIC 引入 DCT 计算中<sup>[31]</sup>。E.P.Mariators 等人的旋转结构利用 DCT 中每个 CORDIC 旋转角度是预先确定的事实, 找到角度  $\Theta$  的表达式, 减少了运算量, 减少了硬件<sup>[32]</sup>。而 F.Zou 等人是将 DCT 结构改进<sup>[33]</sup>, 使得完成一个 DCT 需要较少的 CORDIC 计算, 获得了较高吞吐量。

## 3.4.2 结构设计考虑

以上对当前流行的几种算法结构做了介绍, 它们都有各自的优缺点。在选择或者设计一个算法结构时, 首先要考虑自己的设计的应用对象, 设计要求。本文所研究的设计是一个 IP 核的 ASIC 设计, 所以设计要独立化, 模块化, 以适于 SOC 设计技术。另外, 结构设计还应该考虑到设计对性能的影响, 比如速度、面积以及结构的规则性等。这些因素之间往往是一对一的矛盾, 有时为了更高的速度要牺牲面积, 有时为了节约硬件资源减小面积也会以牺牲速度为代价。所以, 设计者需要根据实现的要求和条件在这些性能指标之间做出折中, 以最大程度的利用硬件资源, 满足设计的要求<sup>[34]</sup>。

考虑图像码流的特点, 数据是以数据流的形式传递, 所以如果采用并行输入计算的结构可能会浪费资源, 还要额外设计串并/并串转换电路。因此, 基于脉动阵列的结构设计方法并不适合本文的设计。设计还要考虑到 ASIC 设计技术的特点, 因为对于应用于 FPGA 的 VLSI 设计与 ASIC 设计是有许多不同的。在 FPGA 设计中, 许多 FPGA 芯片本身包含了一些逻辑单元和存储单元, 我们可以尽量去利用, 上述的基于分配算法的结构就多用于 FPGA 实现, 因为可以利用 FPGA 中固有的 ROM 存储单元; 而 ASIC 设计如果使用 ROM 查找表则必然增加面积, 还会降低速度, 所以本文的设计也没有考虑采用分配算法。

## 3.4.3 设计实现

考虑一维 DCT 变换公式 (3-1):

$$Y(k) = \sqrt{\frac{2}{N}} \varepsilon(k) \sum_{n=0}^{N-1} X(n) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \quad k=0,1,\dots,N-1;$$

$$\text{有 } Y(k) = X(0) \cdot \cos\frac{k\pi}{16} + X(1) \cdot \cos\frac{3k\pi}{16} + \dots + X(7) \cdot \cos\frac{15k\pi}{16} \quad (3-5)$$

由式 3-5 可以看出, 与  $X(0)$  相关的乘积项是  $X(0) \cdot \cos\frac{k\pi}{16}$ ; 与  $X(1)$  相关的乘积项是  $X(1) \cdot \cos\frac{3k\pi}{16}$ ; ……; 与  $X(7)$  相关的乘积项是  $X(7) \cdot \cos\frac{15k\pi}{16}$ 。根据  $k$  的取值, 每个  $X(n)$  对应的有 8 个乘积项, 可以把每个输入  $X(n)$  的所有乘积项都计算出来, 然后再把对应的项纵向相加来计算输出  $Y(k)$ :

$$\begin{aligned} X(0) \cdot \cos\frac{0\pi}{16} + X(1) \cdot \cos\frac{0\pi}{16} + \dots + X(7) \cdot \cos\frac{0\pi}{16} &= Y(0) \\ X(0) \cdot \cos\frac{1\pi}{16} + X(1) \cdot \cos\frac{3\pi}{16} + \dots + X(7) \cdot \cos\frac{15\pi}{16} &= Y(1) \\ \vdots & \\ X(0) \cdot \cos\frac{7\pi}{16} + X(1) \cdot \cos\frac{21\pi}{16} + \dots + X(7) \cdot \cos\frac{105\pi}{16} &= Y(7) \end{aligned} \quad (3-6)$$

这样, 对于每个输入值  $X(n)$ , 都需要得到相关的 8 个乘积项的结果。

基于此种思想, 本文的设计结构分为 3 个部分: 第一部分为乘法实现, 第二部分为交叉选择部分, 第三部分为结果累加部分, 如图 3-2 所示。

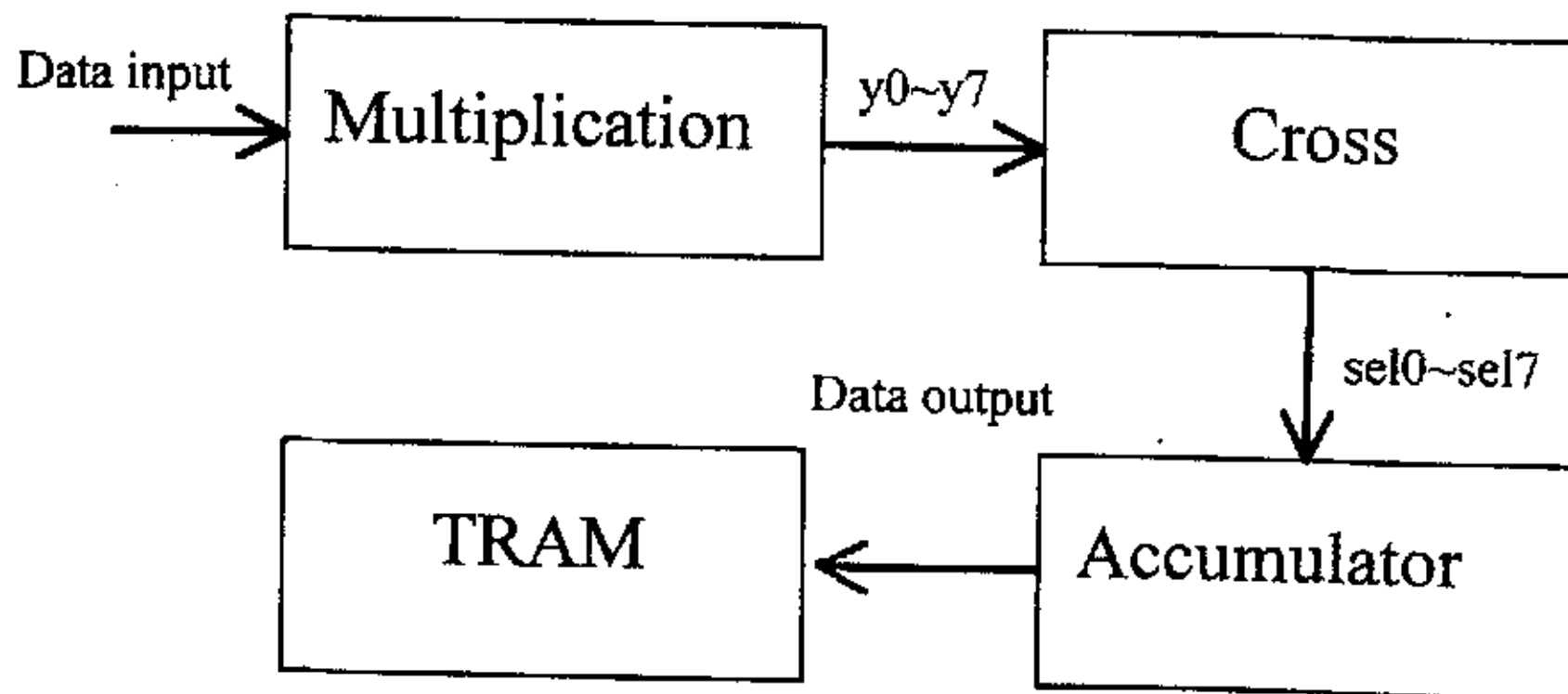


图 3-2 一维 DCT/IDCT 结构

在乘法模块的实现上, 本文的设计采用了基于 CORDIC 运算技术的思想, 利用 DCT/IDCT 余弦系数所具有的旋转特性, 既对于  $8 \times 8$  的二维 DCT/IDCT, 其 64 个余弦系数在坐标上重复旋转, 除去符号不同外, 数值上其实总共只有 8 个固定的值,

这样就可以把乘法看作是乘数固定的运算,从而可以通过移位和加法来实现乘法运算<sup>[35]</sup>。交叉选择部分是为了对乘法得到的结果进行交叉选择排序,这是因为乘法的结果只是输入数据与 8 个固定余弦值的机械相乘的结果。如式 3-6 分析,每个输入的  $X(n)$  的余弦乘积因子不同,但所有这些余弦因子都可等效为  $\cos\frac{0\pi}{16}$ ,  $\cos\frac{1\pi}{16}$ , ...,  $\cos\frac{7\pi}{16}$  共 8 个值中的一个。这样,乘法就可对每个输入数据分别与这 8 个余弦因子相乘,而交叉选择就是对这些结果重新排序组合,以便送入相应的累加器。累加部分就是累加求和,以输出一维 DCT/IDCT 的变换结果。整个一维 DCT/IDCT 结构采用 9 级流水线完成,在变换数据输入 9 个时钟周期后,将开始产生输出结果。

上述算法结构,一方面减少了采用通用乘法器所产生的面积和资源的浪费,另一方面也避免了采用查找表方法实现乘法而需要的存储器设计。9 级流水线的设计更体现了此结构具有的高速度和高吞吐量。同时,它不要求数据同时输入,符合图像编解码的特征,利于实现流水线的操作。因此,本文所采用的结构具有面积小,速度快,模块化的特点。

## 4 DCT/IDCT IP核的逻辑设计

本章将对 DCT/IDCT IP 核的逻辑设计做详细说明。首先需要说明的是，所有的逻辑设计采用的数制表示都是补码。这是因为，补码较之源码在处理减法时很有运算优势。所谓补码，就是指对一个源码数，如果为正数，则其补码与源码相同，如果为负数，则最高位（符号位）为 1，其它位与源码相反，然后再加上 1 所得到的数。例如，源码数 0101(十进制数 5)的补码为 1011。我们知道，在集成电路中的减法运算，实际上都是通过加上一个负数来完成的，这就要依靠补码来实现。例如，十进制运算  $7-3=4$ ，用二进制源码表示就是  $0111-0011=0100$ ，我们可以把此式看成  $0111+(-0011)$ ，再把其变做补码形式就是  $0111+1101$ ，所得结果为 0100，从而得到正确的结果。

### 4.1 乘法逻辑设计

此部分实现一维的 DCT/IDCT 乘法运算。由于 DCT/IDCT 的运算量很大，其中的乘法实现是最耗资源和时间的，乘法运算处理模块的设计直接关系 DCT/IDCT 处理核的性能。

#### 4.1.1 乘法器介绍

在数字信号处理中，乘法器是非常重要的基本运算。乘法器的种类很多，从逻辑设计观点来看，可以把乘法器分为时控乘法器和阵列乘法器两大类。后者具有足够的并行性，不必寄存中间结果，就可得到最后答案。此外，乘法器的结构也依所用的数码（原码，反码或补码）而异。通常，无符号数的乘法比补码乘法稍微简单，但补码最适合于高速乘法。

##### (1) 时控乘法器介绍

下图为移位相加乘法器简图。图中，被乘数（并行码） $X=\{x_4, x_3, x_2, x_1, x_0\}$ 与乘数（串行码） $Y=\{y_3, y_2, y_1, y_0\}$ 相乘。如果 Y 中的每一位都是 1，则此乘法器依次累加下面所示的各行：



x4	x3	x2	x1	x0	y0=1
x4	x3	x2	x1	x0	y0=1
x4	x3	x2	x1	x0	y0=1
x4	x3	x2	x1	x0	y0=1

如果  $y_i=0$ ，则送到加法器的第  $i$  行将是 0。

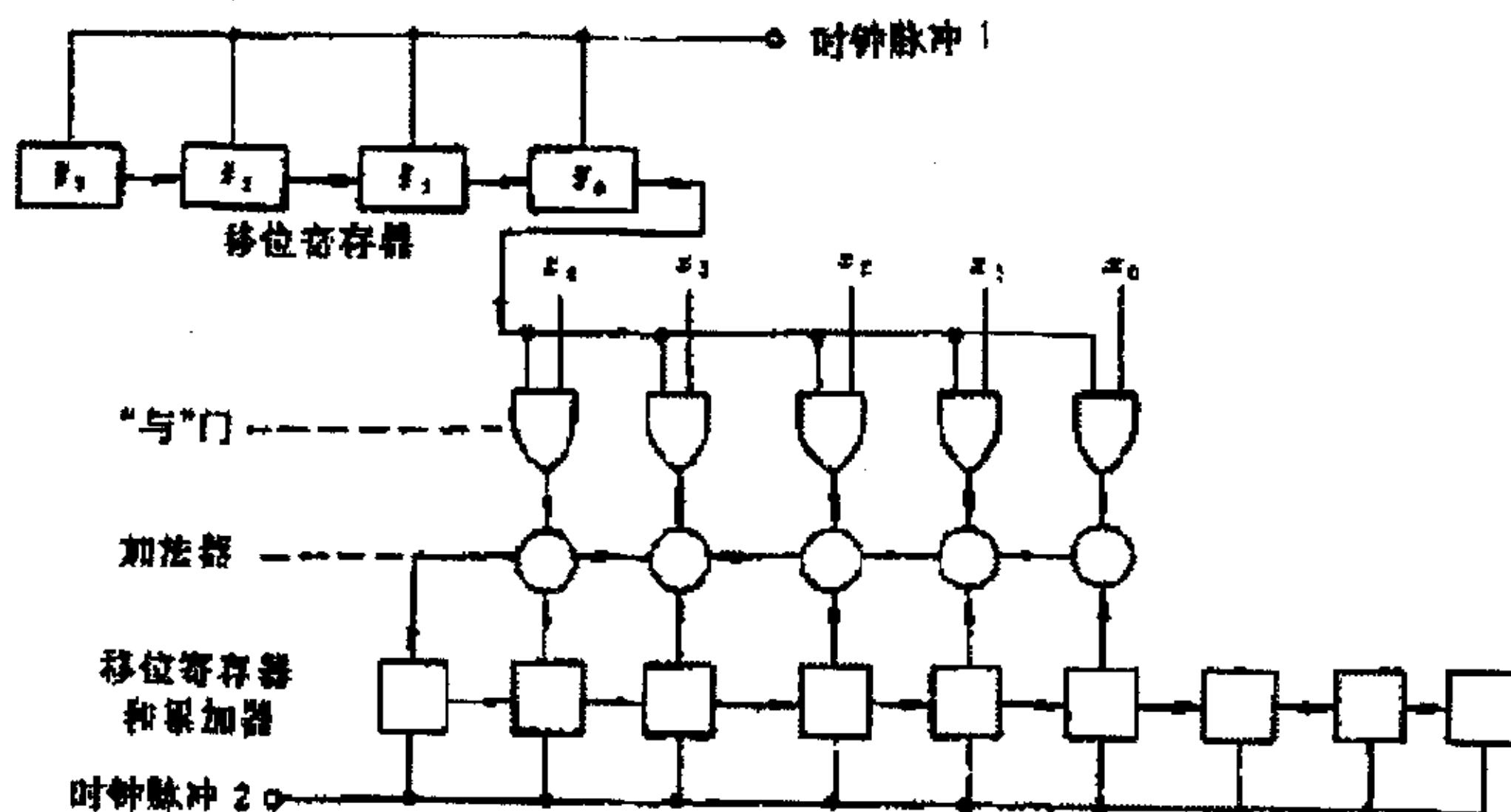


图 4-1 移位相加乘法器简图

根据乘法算法的规定，每送入一行被乘数  $X$ ，累加器就向右移位，以使相继各行逐一提供更高的值。图 4-1 的工作时序是这样的：当时钟脉冲 1 到来后，首先将  $y_i$  ( $i=0, 1, 2, 3$ ) 移入与门，于是  $X$  或 0 被送入加法器。待加法器电平稳定后，时钟脉冲 2 便将加法器的输出送入累加器，待累加器稳定后便右移移位。

显然，这种移位相加乘法器电路虽然简单，但是速度是很慢的。而且，在这种加法器内没有超前进位逻辑，所以每次累加只能让进位传过整个加法器。所以，又出现了采用 BOOTH 算法编码的乘法器，BOOTH 编码乘法器采用对乘数进行再编码的方法来减少加法次数，从而提高了运算速度。

## (2) 阵列乘法器介绍

阵列乘法器是用一位加法器排列成的二维阵列，它是最快的乘法器。与图 4-1 所示的时控乘法器不同，阵列乘法器是完全没有存储器逻辑网络的，它只需要规定一个输入信号作用之后，乘积产生之前的建立时间。阵列乘法器的结构，可以根据

加法器的连接方法或者依照处理负数的方法分成许多种。图 4-2 所示的是一个用于  $4 \times 4$  位无符号数相乘，但效率并不很高的阵列乘法器。图中每一个小圆圈表示一个一位加法器。可以看到，每行加法器是全行波加法器，且每行加法器将其部分和送给下面一行。最下面一行的  $p_7 \dots p_1, p_0$  为乘积。

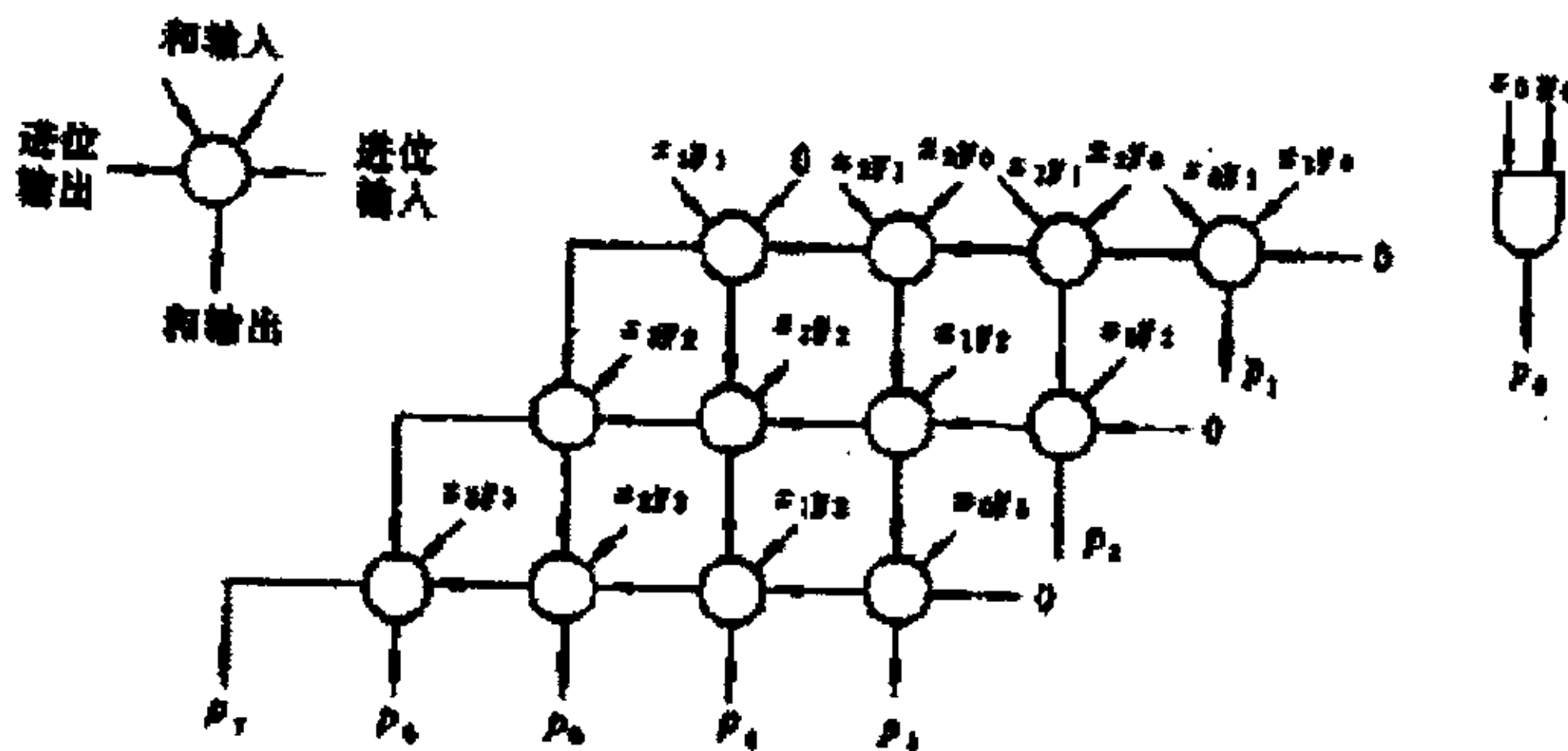


图 4-2 无符号  $4 \times 4$  阵列乘法器

## 4.1.2 利用移位求和实现乘法逻辑

根据 DCT/IDCT 计算公式，对于每一个输入变换数据  $X(n)$ ，为了计算  $Y(k)$ ，我们需要将每一个  $X(n)$  分别与不同的余弦因子相乘。对于 8 点的 DCT/IDCT 变换，余弦因子从  $\cos \frac{0\pi}{16}$  到  $\cos \frac{105\pi}{16}$  不等。由于余弦因子具有周期旋转特性 ( $\cos(x) = \cos(2\pi + x)$ ,  $\cos(x) = -\cos(\pi - x)$  以及  $\cos(x) = \cos(-x)$ )，实际上，除去符号不同，这些余弦因子只有 8 个固定的值，分别为  $\cos \frac{0\pi}{16}$ ,  $\cos \frac{1\pi}{16}$ ,  $\dots$ ,  $\cos \frac{7\pi}{16}$ 。由于这个特性，我们可以把余弦系数看作是固定的乘数，对每一个输入的被乘数  $X(n)$ ，分别乘以固定的几个余弦系数即可<sup>[36]</sup>。因此，此模块共需要 8 个乘法器来完成。

从以上介绍的乘法器，我们可以发现，用硬件实现乘法器会占用非常多的资源，而且多位的乘法器处理速度也非常慢，这样乘法器就会成为整个处理核的瓶颈。对于乘法器的处理，还有一种方法是使用基于 ROM 的查找表 LUT 方法。此方法是将一

255~255个灰度输入值与这8个固定的余弦系数相乘的可能结果预先算出，并按相应地址存放于ROM之中，然后根据地址逻辑对输入取址来获得乘法结果。这种方法不但会使乘法器的面积很大，而且只能微弱地改善均方差（MSE），从ROM中读出数据的速度也限制了处理核的速度。这种方法一般只适合于FPGA实现，因为很多FPGA芯片中包含了ROM模块，可以加以利用。而在ASIC设计中，这种方法一般不可取。因此本设计采用了一种新的解决方法。乘法器实质上是移位求和的过程，左移一位就是乘2，右移一位就是除2。经过以上分析，8个固定的余弦系数可作为乘数，因为余弦系数都是小于或等于1的小数。这里，首先需要将其转换为分式表达，如下所示：

$$\begin{aligned}\cos\left(\frac{\pi}{16}\right) &= (1024+512+256+128+64+16+8+1)/2048 = 0.9808; \\ \cos\left(\frac{2\pi}{16}\right) &= (256+128+64+16+8+1)/512 = 0.9238; \\ \cos\left(\frac{3\pi}{16}\right) &= (1024+512+128+32+4+2+1)/2048 = 0.8315; \\ \cos\left(\frac{4\pi}{16}\right) &= (128+32+16+4+1)/256 = 0.7071; \\ \cos\left(\frac{5\pi}{16}\right) &= (512+32+16+8+1)/1024 = 0.5556; \\ \cos\left(\frac{6\pi}{16}\right) &= (32+16+1)/128 = 0.3827; \\ \cos\left(\frac{7\pi}{16}\right) &= (16+8+1)/128 = 0.1951;\end{aligned}$$

每个余弦系数都可用2的 $n$ 次幂的和与2的 $m$ 次幂的商来近似表示。一个数乘以 $2^n$ ，相当于将此数左移 $n$ 位，除以 $2^m$ 相当于将其右移 $m$ 位。这样乘法操作就可以用左移求和，然后再对移位求和后的结果进行右移移位来实现。以上表达式中，分子和分母的选取关系到左移和右移的移位位数，这关系到计算的精度。分母越大，计算的精度就越高，分子也就越大（左移越多），同时也意味着加法的位宽越宽，也就增加了资源，和组合逻辑的延时；分母越小，就越意味着舍去的小数尾数越多，就可能造成精度无法满足要求。这里所取的分母和分子值可以使精度到小数点后第四位，这可以满足一般大多数的应用要求。从以上的余弦系数分式表达式中，我们可以看出，不同的系数所需的加法数不同，这十分不利于并行的加法计算，而且，最

多的需要七级串行的加法器和7个移位寄存器，最少的只需要两级的串行加法，这是十分浪费硬件资源的，而且七级的加法也不可能在一个时钟内完成。因此，还需要对以上的表达式进行改进。经过分析，可以先设计几个前级的移位加法器，对每一个输入数据x，可以通过左移移位和加法先得到x3, x5, x7, x17, 这几个中间值。

$$x3 = 2x + x;$$

$$x5 = 4x + x;$$

$$x7 = 8x - x;$$

$$x17 = 16x + x;$$

这样，就可以将先前余弦系数的分式表达式改写为以下形式：

$$\cos\left(\frac{\pi}{16}\right) = (17+8+64 \times 3+256 \times 7)/2048 = 0.9808;$$

$$\cos\left(\frac{2\pi}{16}\right) = (17+8+64 \times 7)/512 = 0.9238;$$

$$\cos\left(\frac{3\pi}{16}\right) = (7+32 \times 5+512 \times 3)/2048 = 0.8315;$$

$$\cos\left(\frac{4\pi}{16}\right) = (32 \times 5+4 \times 5+1)/256 = 0.7071;$$

$$\cos\left(\frac{5\pi}{16}\right) = (17+8+32 \times 7)/1024 = 0.5556;$$

$$\cos\left(\frac{6\pi}{16}\right) = (32+17)/128 = 0.3827;$$

$$\cos\left(\frac{7\pi}{16}\right) = (17+8)/128 = 0.1951;$$

在此种表达式中我们可以看出，相比上一种表达，这里最多只需要3级的加法器实现，加上前级实现x3, x5, x7和x17的加法器，最多需要4级加法器，最少只需要两级加法器可实现乘法操作。这样，平衡了各个乘法路径，最大程度的利用了硬件资源。在完成左移和加法操作后，我们得到数据是相应的乘积结果2<sup>n</sup>倍，因此，最后还需要按照以上余弦系数表达式的分母值的大小，对其进行右移操作，这样就完成了乘法器的逻辑功能。综合出的电路如图4—3所示，从图中可以看出，设计共用了14个加法器。比之采用通用的浮点乘法器而言，大大的节约了硬件资源。

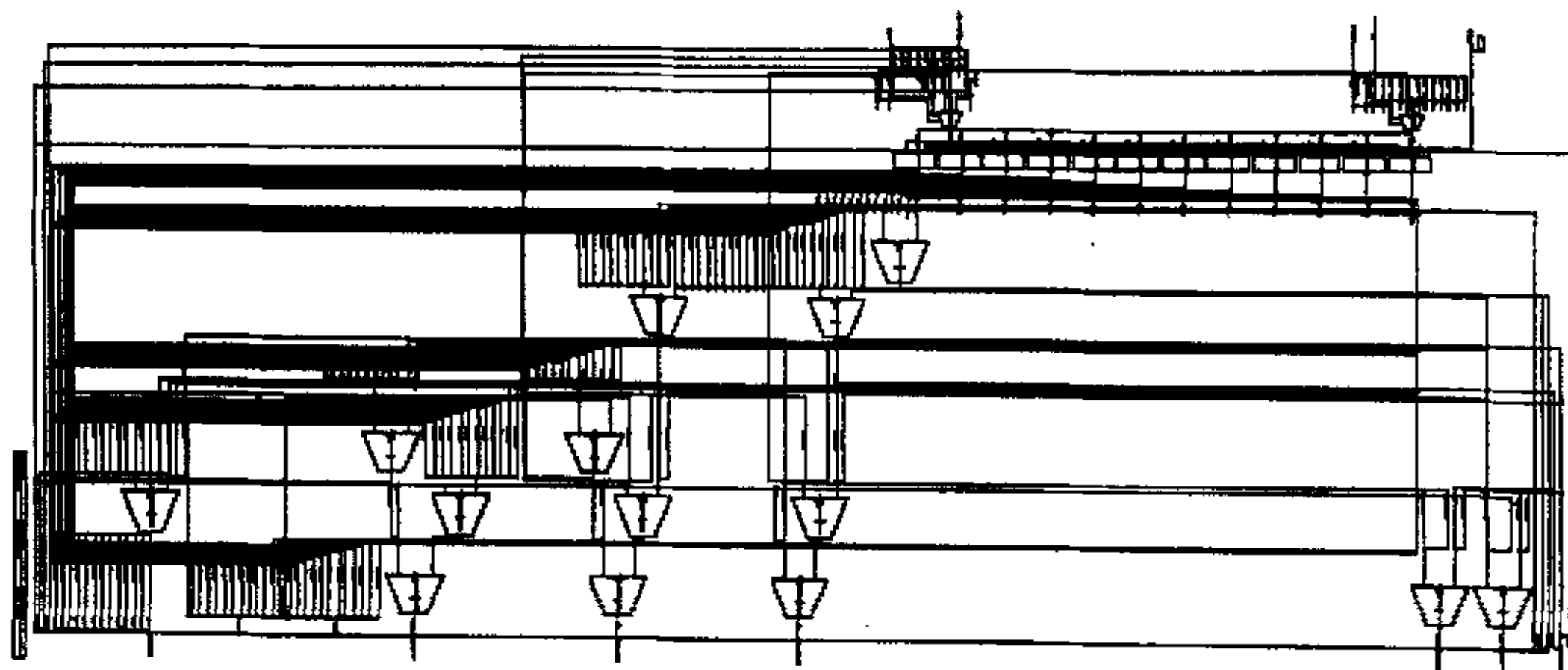


图4-3 乘法部分逻辑电路

### 4.1.3 乘法逻辑时序考虑

我们知道，移位操作是非常简单的运算，因此，在以上设计的乘法逻辑中，组合逻辑的延时主要在于加法器的延时。通过以上分析，我们可以找到有4级加法器的路径为关键路径。此路径中，包含一个17位宽的加法器和3个23位宽的加法器。在设计前经过实验发现，一个32位加法器的时延大约为2ns，所以在时钟内完成乘法运算的时延将一定小于8ns，这可以满足100M（周期为10ns）以上的时钟频率运行。图4-4是对所设计的乘法器逻辑电路的时序仿真图，图中的idct信号为低电平，表示所进行的是IDCT逻辑的仿真。因此，作为DCT逻辑的像素输入信号pixin为0，而IDCT逻辑的输入信号为xin，y0~y7为乘法器逻辑的输出。

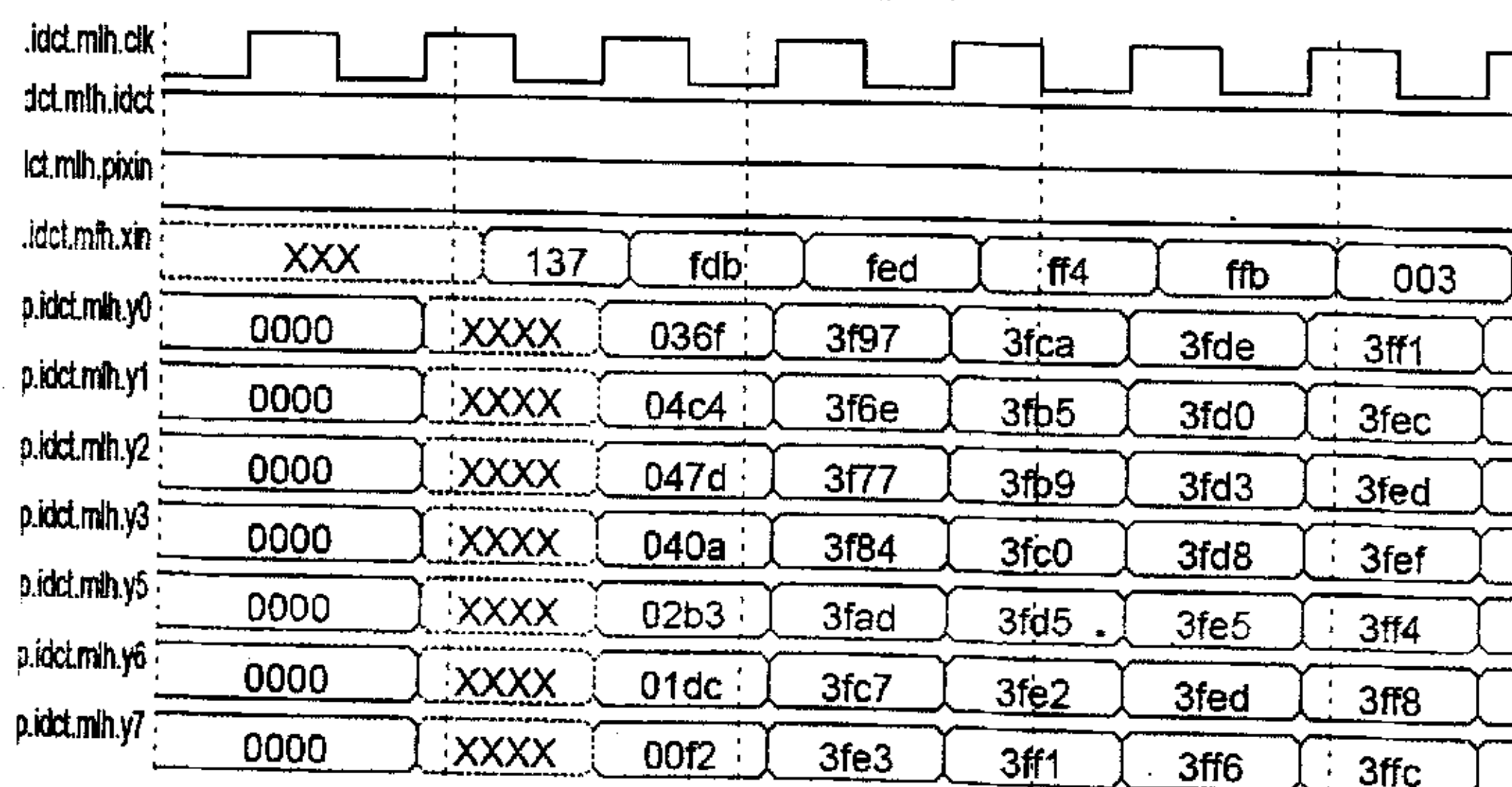


图 4-4 乘法器逻辑仿真时序图



## 4.2 交叉选择逻辑设计

乘法逻辑产生的输出仅仅是输入变换数据与以上几个固定的余弦系数的乘积结果。而根据离散余弦正反变换公式,对每一个 DCT/IDCT 输入数据,需要根据它在矩阵中的位置来乘以一特定的余弦因子。如前所述,所有余弦因子  $\cos(\frac{0\pi}{16})$ ,  $\cos(\frac{\pi}{16})$ ,  $\dots$ ,  $\cos(\frac{105\pi}{16})$  的数值实际上都只是  $\cos(\frac{0\pi}{16})$ ,  $\cos(\frac{\pi}{16})$ ,  $\dots$ ,  $\cos(\frac{7\pi}{16})$  这些固定的余弦值,但从乘法逻辑得到的结果只是对每一个输入数据分别与上述 8 个余弦因子机械相乘,而没有对这些结果排序。例如,根据公式

$$Y(k) = \sqrt{\frac{2}{N}} \varepsilon(k) \sum_{n=0}^{N-1} X(n) \cos\left[\frac{(2n+1)k\pi}{2N}\right], k = 0, 1, \dots, N-1 \quad \text{式 4-1}$$

当  $k=1$  时,即对应计算结果  $Y(1)$ ,对于输入的  $X(0)$ ,相应的余弦因子为  $\cos(\frac{\pi}{16})$ ; 对于输入的  $X(1)$ ,相应的余弦因子为  $\cos(\frac{3\pi}{16})$ ; 对于输入的  $X(2)$ ,相应的余弦因子为  $\cos(\frac{5\pi}{16})$ ; 对于输入的  $X(3)$ ,相应的余弦因子为  $\cos(\frac{7\pi}{16})$ ; 对于输入的  $X(4)$ ,相应的余弦因子为  $\cos(\frac{9\pi}{16})$ ; 对于输入的  $X(5)$ ,相应的余弦因子为  $\cos(\frac{11\pi}{16})$ ; 对于输入的  $X(6)$ ,相应的余弦因子为  $\cos(\frac{13\pi}{16})$ ; 对于输入的  $X(7)$ ,相应的余弦因子为  $\cos(\frac{15\pi}{16})$ 。也就是说,我们需要将  $X(n)$  与相应的余弦因子的乘积送入对应于某个  $Y(k)$  的累加器来计算  $Y(k)$ ,这就必须调整乘法逻辑机械相乘后的乘法结果顺序,调整是由控制逻辑来产生相应的调整信号来完成的。另外,还需要根据所要实现的是 DCT 变换还是 IDCT 变换来处理交叉选择逻辑,因为本设计是 DCT/IDCT 复用的 IP 设计。

图 4-5 是对交叉选择逻辑的仿真图。图中的  $y_0 \sim y_7$  为乘法器的输出结果,  $sel_0 \sim sel_7$  为交叉选择后的输出。从图中可以看出,  $sel_0 \sim sel_7$  的值都是对应的  $y_0 \sim y_7$  中的值,所不同的只是将它们重新做了排序。

ldct.crh.y0	0000	XXXX	036f	3f97	3fca	3fde	3ff1	0008	0005	000bc
ldct.crh.y1	0000	XXXX	04c4	3f6e	3fb5	3fd0	3fec	000b	0007	000fc
ldct.crh.y2	0000	XXXX	047d	3f77	3fb9	3fd3	3fed	000b	0007	000ec
ldct.crh.y3	0000	XXXX	040a	3f84	3fc0	3fd8	3fef	0009	0006	000dc
ldct.crh.y5	0000	XXXX	02b3	3fad	3fd5	3fe5	3ff4	0006	0004	0008c
ldct.crh.y6	0000	XXXX	01dc	3fc7	3fe2	3fed	3ff8	0004	0003	0006c
ldct.crh.y7	0000	XXXX	00f2	3fe3	3ff1	3ff6	3ffc	0002	0001	0003c
lct.crh.sel0	0000	XXXX	036f	3f6e	3fb9	3fd8	3ff1	0006	0003	
lct.crh.sel1	0000	XXXX	036f	3f84	3fe2	3ff6	3ff1	000b	0007	0008c
lct.crh.sel2	0000	XXXX	036f	3fad	3fe2	3fd0	3ff1	0002	0007	000dc
lct.crh.sel3	0000	XXXX	036f	3fe3	3fb9	3fe5	3ff1	0009	0003	000fc
lct.crh.sel4	0000	XXXX	036f	3fe3	3fb9	3fe5	3ff1	0009	0003	000fc
lct.crh.sel5	0000	XXXX	036f	3fad	3fe2	3fd0	3ff1	0002	0007	000dc
lct.crh.sel6	0000	XXXX	036f	3f84	3fe2	3ff6	3ff1	000b	0007	0008c
lct.crh.sel7	0000	XXXX	036f	3f6e	3fb9	3fd8	3ff1	0006	0003	

图 4-5 交叉选择逻辑仿真图

图 4-6 是对交叉选择逻辑综合的电路图。此部分逻辑是组合逻辑电路，没有时钟延迟，从图中可以看出，电路主要由多级的多路选择器构成。

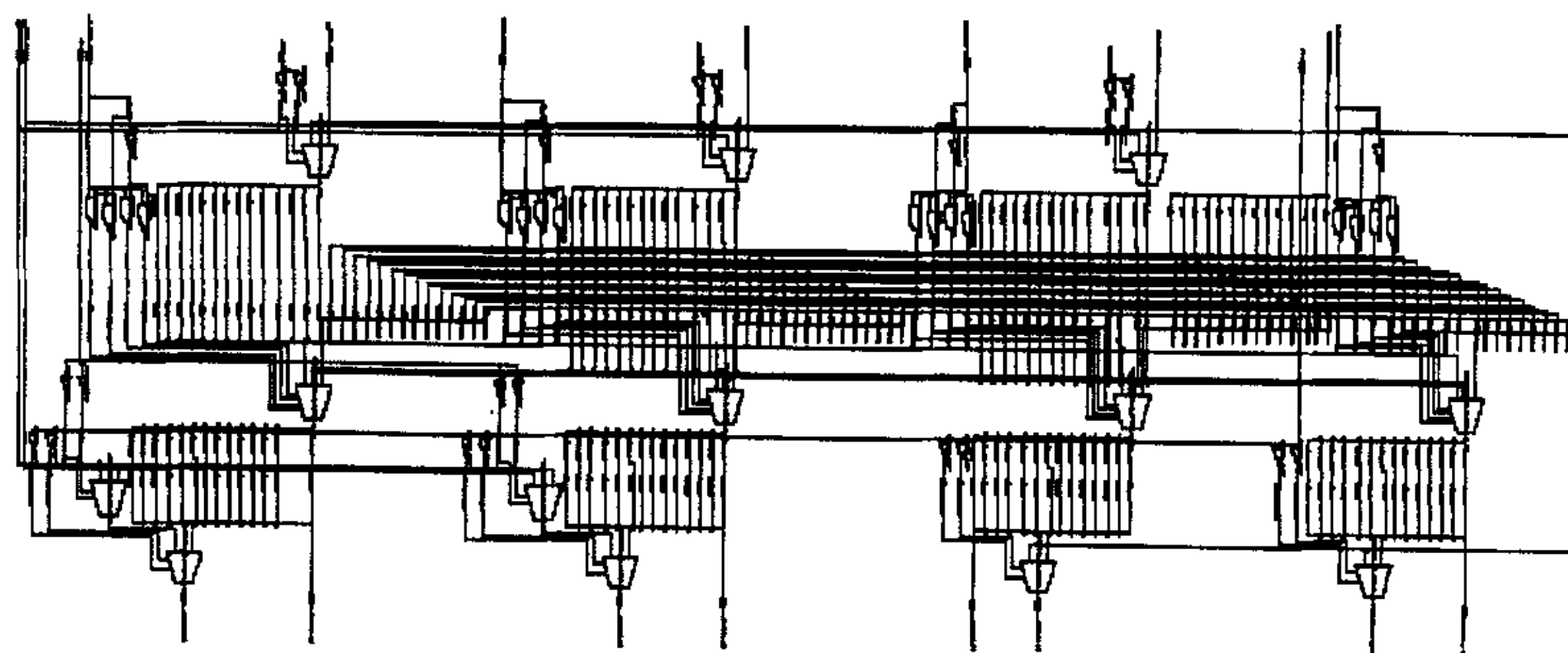


图 4-6 交叉选择逻辑电路图

### 4.3 累加器逻辑设计

此部分逻辑实现对交叉选择后的乘法结果的累加，产生一维 DCT/IDCT 的变换结果。对于 8 点的 DCT/IDCT 变换，每一个输入数据，要产生 8 个相乘的结果，对应需要 8 个累加器来分别累加产生 8 个变换结果。8 个累加器并行的工作，在 8 个时钟周期后，8 个累加器将产生一维 DCT/IDCT 的全部变换结果。为了实现流水的操作，这些变换结果将被放入移位寄存器中，在下 8 时钟周期被一个一个的输出，存入转置 RAM 中。由于要等待 8 个输入，才能产生累加结果，因此，累加器逻辑不可避免的造成了 8 个时钟周期的延时。图 4-7 是累加器逻辑的功能仿真图，从图中可以看出，输出 y 在输入 sel0~sel7 有效的 8 个时钟后开始产生有效数据。

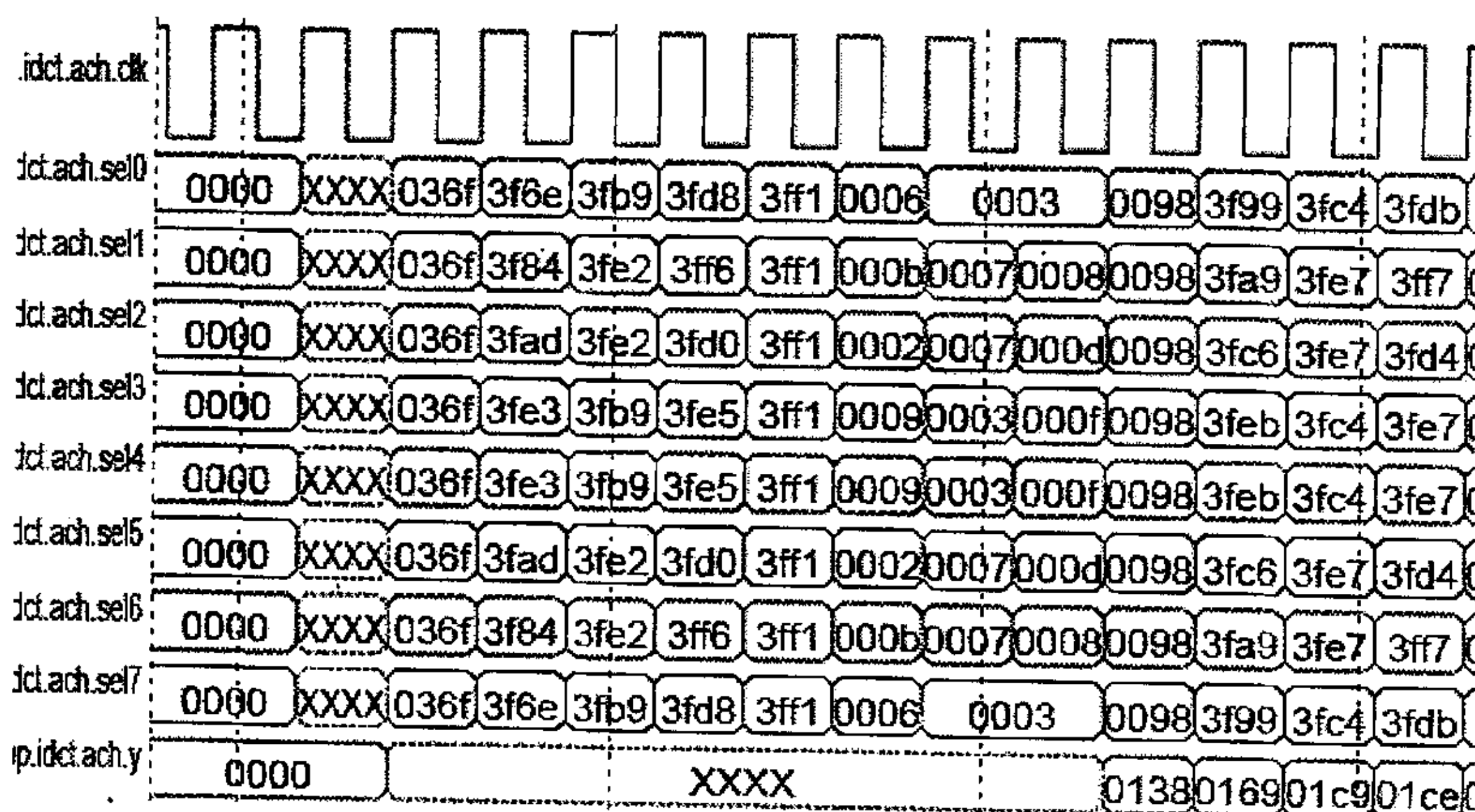


图 4-7 累加器逻辑仿真时序图

图 4-8 是累加器综合后的逻辑电路图。累加器主要由两部分寄存器组构成，第一个部分寄存器组是 8 个累加寄存器，第二部分是 8 个移位寄存器。累加寄存器寄存累加的中间结果，并由控制信号控制，在完成累加后，将最终的累加结果赋给移位寄存器。8 个移位寄存器从 8 个累加寄存器获取累加结果，然后在 8 个时钟周期内分别移位输出。图 4-8 中的左边部分是累加逻辑中的加法电路，而右边部分既是移位寄存器。

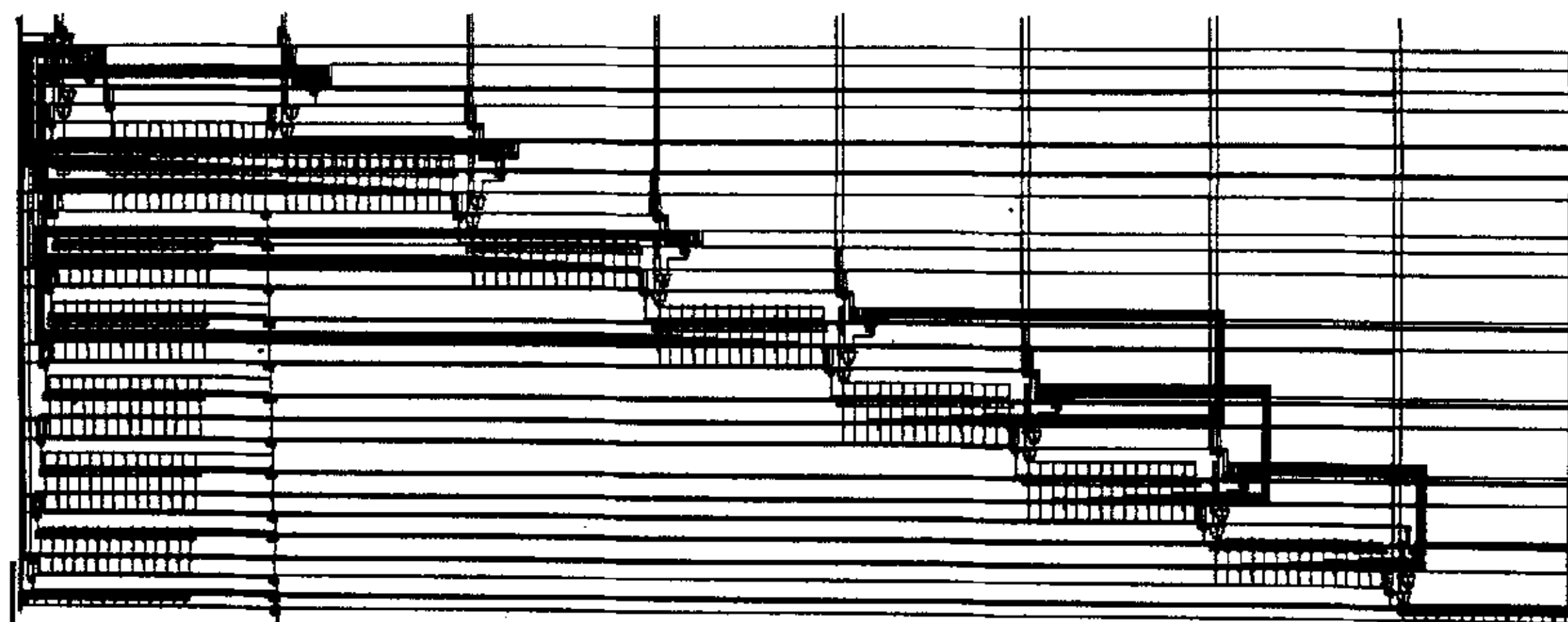


图 4-8 累加器逻辑电路

## 4.4 转置 RAM 逻辑

二维 DCT/IDCT 变换，在第一维变换按行方向（或列方向）结束后，要对其结果再按列方向（或行方向）做第二维的变换。转置 RAM 逻辑要实现的就是对一维 DCT/IDCT 变换结果的转置，以便进行第二维的变换。

RAM 模型可被看作是一个  $8 \times 8$  的矩阵。按行（按列）输入的变换数据的一维 DCT/IDCT 变换结果将被存入 RAM 模型的列(行)，而 RAM 中的行（列）将作为第二维 DCT/IDCT 变换的输入，这可根据二维 DCT/IDCT 变换公式 3-3 看出。为了保证流水线的操作，RAM 模型被设计为双端口的，以便能同时写入和读出数据，使两个一维 DCT/IDCT 变换核可以并行的工作。转置逻辑的实现主要依靠控制逻辑产生正确的 RAM 模型读写地址信号。转置的实现是造成 DCT/IDCT 变换延迟的最大原因，因为要等待第一维的计算完成，第二维计算的输入数据才能完全产生。图 4-9 是转置 RAM 的读写顺序：在数据按行写入的时候，将被按列读出；在数据按列写入时，将被按行读出，这样可以实现多个 DCT/IDCT 变换块连续流水的处理操作。

0	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	58	59	60	61	62	63

0	8	16	24	32	40	48	56
1	9	17	15	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

a.读出(写入)

b.写入(读出)

图 4-9 转置 RAM 读写顺序

图 4-10 和图 4-11 是转置 RAM 逻辑的仿真图,从图中可以看出,输入数据 din 按行的方式被写入 RAM,在数据写到地址 36(十进制为 54),即矩阵的第 6 行 6 个位置时,数据开始被按列读出。如果第一次数据被按行写入的,在写完一轮后,数据将被按列写入;数据第一次被按列读出的,在完成一轮后,又被按行读出,彼此交替运行,完成转置功能。转置逻辑的实现主要依靠读写地址逻辑,这是由 DCT/IDCT 的控制逻辑来产生的。从功能仿真图中可以看出,在本设计中,转置 RAM 的输入输出延时为 54 个时钟周期。

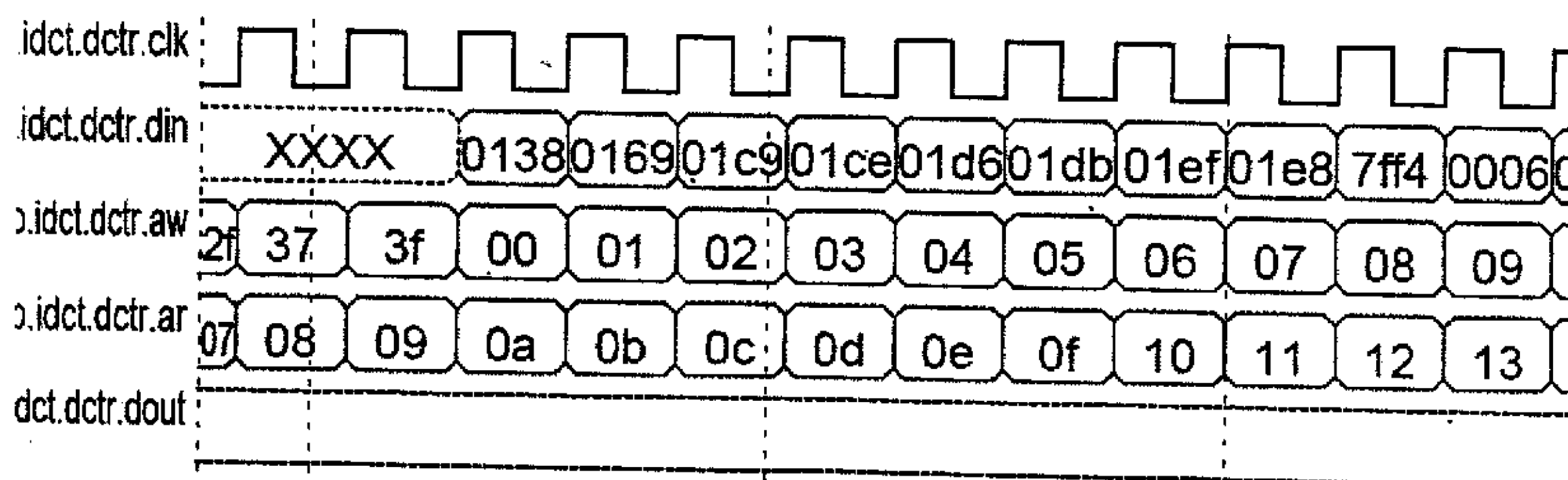


图 4-10 转置 RAM 数据写入仿真波形



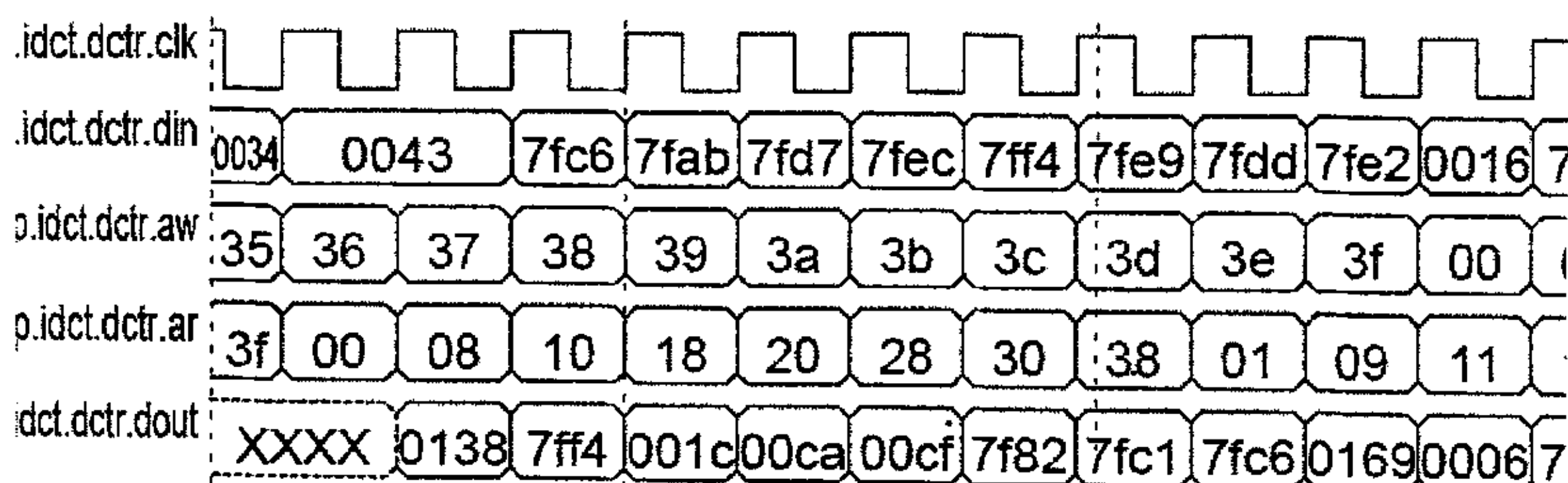


图 4-11 转置 RAM 数据读出仿真波形

## 4.5 控制逻辑

为了协调 DCT/IDCT 处理核各模块的工作，保证模块的流水处理，必须设计一个控制模块，由它来产生逻辑控制信号，协调整个变换过程中各模块的工作。控制逻辑产生 DCT/IDCT 变换核的逻辑控制信号，包括控制转置 RAM 的读写时序，乘法逻辑，交叉选择的逻辑，累加控制逻辑和数据输出等。在 ASIC 设计中，逻辑控制是一个设计的核心部分，也是设计最复杂的一部分。对于复杂的逻辑控制设计，我们往往采用状态机的设计方法来实现逻辑控制。但对于流水线的设计，因为数据路径在每个模块中的延迟已经被我们固化，不需要用状态机来实现逻辑控制。我们只需要根据一个 start 的启动信号，再跟据设计固化的模块延迟，为各个模块产生相应的逻辑控制信号，以使它们与输入数据协调，产生正确地计算结果。

逻辑控制的设计思想是要实现数据的流水线处理，即  $8 \times 8$  的变换块可以连续的输入 DCT/IDCT 变换核。但同时，这也对输入提出了要求，当 start 信号有效时，逻辑控制就认为数据开始以  $8 \times 8$  的块连续的输入，如果在输入完一个或几个  $8 \times 8$  的数据块后，有停顿，又接着输入，就有可能出问题。这是因为，逻辑控制都是以 start 信号为标志来产生控制信号，每个路径的延迟都是固定的，如果输入出现停顿，就必然会与控制信号不匹配，从而得出错误的结果。因此，一旦  $8 \times 8$  的输入数据块之间出现停顿，再次输入计算数据时，需要重新产生新的 start 信号。而且，停顿的间隔时间必须大于一次 DCT/IDCT 变换的延迟（73 个时钟周期）。因为，在此之前，上一次输入的数据仍在被处理，还未完全输出，新的 start 信号会产生新的转置 RAM 读写地址逻

辑而覆盖正在运行的，这样就会造成上一次被处理的数据产生错误的输出。

为了适应输入数据流的变化，匹配输入数据的速率，在输入数据产生比较慢时，使 DCT/IDCT 处理核可以暂停工作，等待输入数据，是十分必要的。这个功能的实现是靠一个外部的使能信号 `en` 来控制的。当外部把使能信号 `en` 拉低时，DCT/IDCT 处理核的所有逻辑就会保持各个寄存器的当前值，输出也同时被保持，在使能信号重新有效时，又继续计算。这样，DCT/IDCT 处理核就可以保证对输入数据处理的连续性，保证正确的逻辑输出。

## 4.6 DCT/IDCT 变换核输入输出逻辑

DCT/IDCT 变换核由第一维和第二维的 DCT/IDCT 处理逻辑，转置 RAM 逻辑和控制逻辑组成。以上介绍了一维处理部分的乘法逻辑，交叉选择逻辑和累加逻辑的设计，第二维的变换逻辑同样包含这几个部分，实现的逻辑与第一维完全一致。

DCT/IDCT 变换核的输入输出信号如下所示：

输入信号：

`clk`: 时钟信号；

`reset`: 异步复位信号；

`sw_res`: 同步复位信号；

`en`: 使能信号；

`start`: 变换开始信号；

`idct`: DCT/IDCT 选择信号；

`pixin` (位宽为 9 位): DCT 变换像素输入；

`xh` (位宽为 12 位): IDCT 变换数据输入；

输出信号：

`de_out` (位宽为 9 位): IDCT 变换后的输出；

`pixout` (位宽为 8 位): IDCT 变换后的像素输出；

`yv` (位宽为 12 位): DCT 变换后数据输出；

`ready`: 数据输出准备；

在变换数据开始输入时,应使 start 信号拉高保持到时钟的下一个上升延到来再变低, idct 信号为 DCT 变换或 IDCT 变换选择信号。为“1”,表示进行 IDCT 变换,相应的输入信号 xh 有效,为“0”,表示进行 DCT 变换,  $-255 \sim 255$  的输入信号 pixin 做 DCT 变换,在变换数据开始输出前一个时钟周期, ready 信号将会保持一个时钟的高电平,以提示数据输出。IDCT 变换后的输出结果是相应的  $-255 \sim 255$  的像素值,相应的输出为 de\_out,而对于 JPEG 而言, IDCT 输出只是  $0 \sim 255$  的图像像素值,相应的用 pixout 信号表示。DCT 变换后的输出是  $-2000 \sim 2000$  的值,用 yv 信号表示。数据的输入和输出都是以补码数表示。

模块的设计是 DCT/IDCT 复用的,但不能同时进行 DCT 和 IDCT 变换。DCT 变换与 IDCT 变换有着相同的时序。因为采用流水线的设计技术,从变换数据开始输入到变换后数据输出的延时固定为 73 个时钟周期。

## 4.7 逻辑设计中应注意的问题

### 4.7.1 组合逻辑设计中的竞争和冒险

在组合逻辑设计中,由于从输入到输出的过程中,不同通路上门的级数不同,或者门电路平均延迟时间的差异,使信号从输入经不同通路传输到输出级的时间不同。由于这个原因,可能会使逻辑电路产生错误输出。通常把这种现象称为竞争冒险<sup>[37][38]</sup>。如图4-12(a)所示的电路中,输出  $Y = A + \bar{A}$ ,在不考虑门和线网延迟的理想情况下,其输出波形如图4-12(b)所示。当考虑电路的门延迟时,假设门G1的传播延时为  $t_{pd}$ ,则其工作波形如图4-12(c)所示。由图可见,G2门的两个输入信号  $A$  和  $\bar{A}$ ,由于传输的路径不同,到达G2门的时间不同, $\bar{A}$  信号比  $A$  信号到的晚,于是在输出  $Y$  出现了一个不期望出现的负尖峰脉冲信号。在图4-13(a)所示的电路中,输出  $Y = A \cdot \bar{A}$ ,如考虑G1门的平均传输延时时间  $t_{pd}$ ,则在输出  $Y$  出现了一个应有的正尖峰脉冲信号,如图4-13(b)所示。竞争冒险现象的出现可能会导致负载电路产生错误的动作。

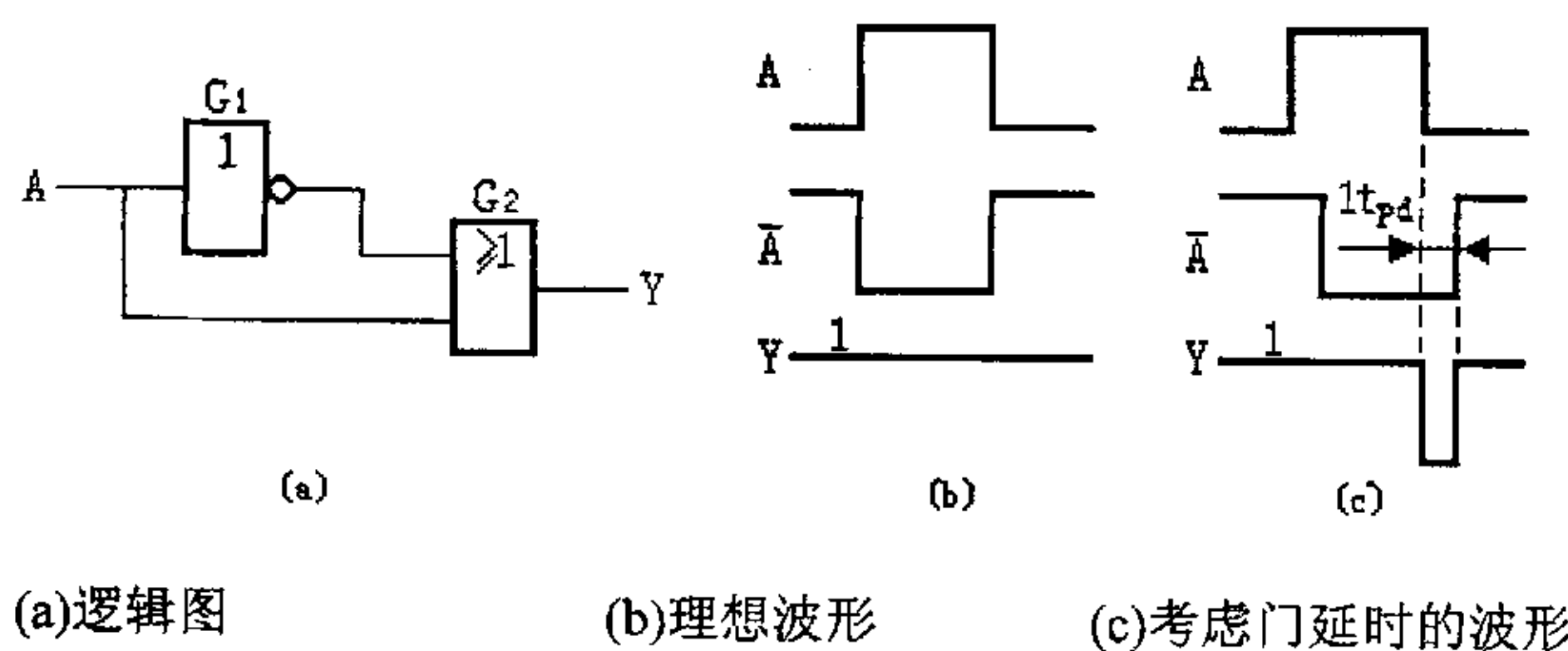


图4-12 产生负尖峰脉冲冒险

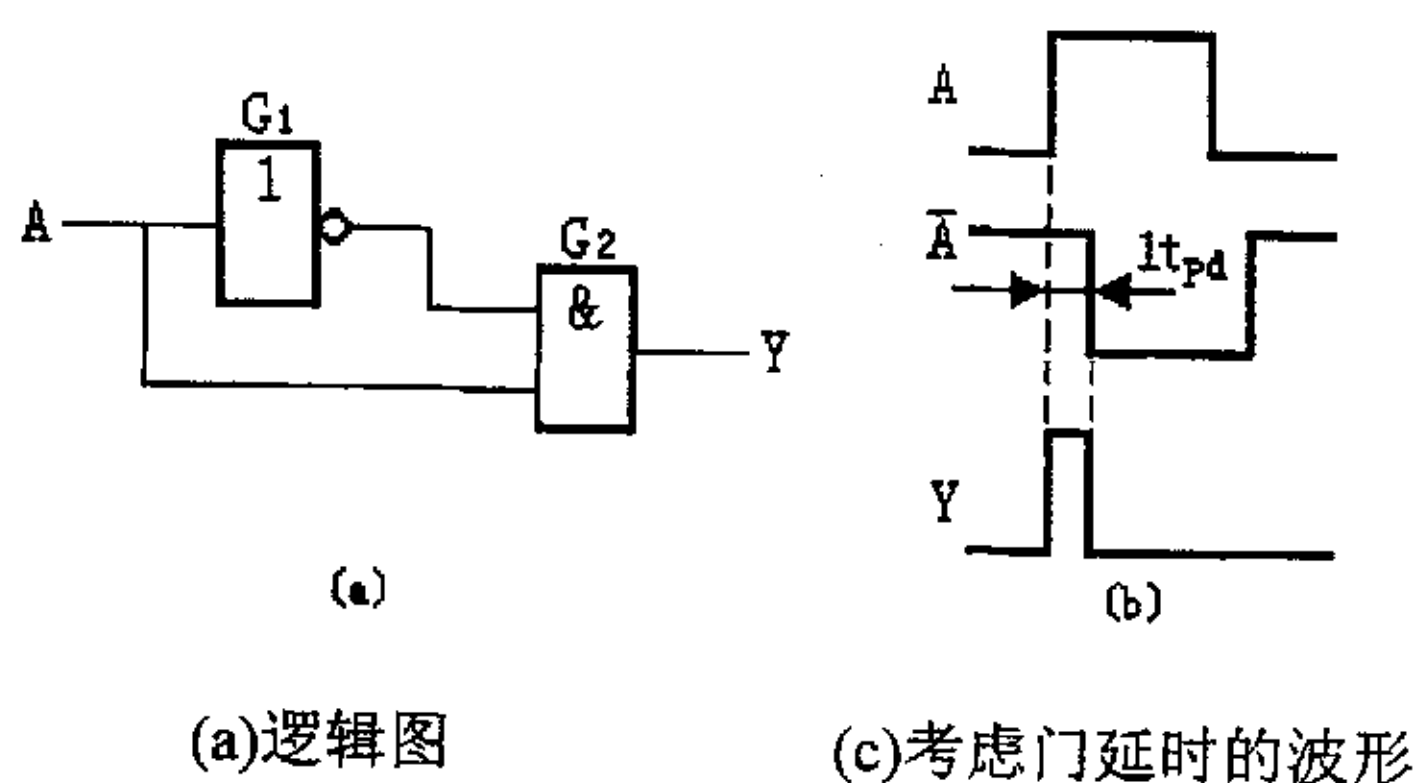


图4-13 产生正尖峰脉冲冒险

组合逻辑电路设计中，竞争冒险一般可以采用三种方法消除，即在逻辑表达式中增加冗余项、增加惯性延迟环节和加入选通脉冲，前两种方法是将具有较少门延时的信号人为地插入延时，使其在时间上与经过较多门延时的信号保持一致。第三种方法是利用选通脉冲在信号作用时间上加以控制，避开产生冒险的尖峰。但VLSI逻辑电路的设计有别于分立元件的电路设计。在分立元件的设计中，可以精确计算出每个信号在相互作用前所经过的门延迟时间，从而可以估计冒险是否可以发生及如何消除。在VLSI设计中，往往无法正确估算出某个信号的延时，而且增加冗余项也会加大逻辑块的开销。因此，在VLSI设计中，消除静态冒险最有效的方法是加入选通脉冲，采用同步时序逻辑设计。

所谓同步时序逻辑是指表示状态的寄存器组的值只能在唯一确定的触发条件发生时才改变。只能由时钟的上升延或下降延触发的状态机就是一列。与同步时序逻辑

辑相对应的是异步时序逻辑，异步时序逻辑是指触发条件由多个控制因素组成，任何一个因素的改变都可以触发。同步时序逻辑比异步时序逻辑具有更简单、更可靠的逻辑关系。电路设计中，需要尽可能避免使用异步时序逻辑，因为异步时序逻辑很容易在电路中产生竞争和冒险<sup>[39][40]</sup>。

## 4.7.2 时钟设计问题

无论是用离散逻辑、可编程逻辑，还是用全定制硅器件实现的任何数字设计，为了成功地操作，可靠的时钟是非常关键的。设计不良的时钟在极限的温度、电压或制造工艺的偏差情况下将导致错误的行为，并且调试困难、花销很大。时钟一般可分为如下四种类型：全局时钟、门控时钟、多级逻辑时钟和波动式时钟。多时钟系统能够包括上述四种时钟类型的任意组合。

### (1) 全局时钟

对于一个设计项目来说，全局时钟(或同步时钟)是最简单和最可预测的时钟。在ASIC设计中最好的时钟方案是：由专用的全局时钟输入引脚驱动的主时钟去时钟设计项目中的每一个触发器。只要可能就应尽量在设计项目中采用全局时钟。

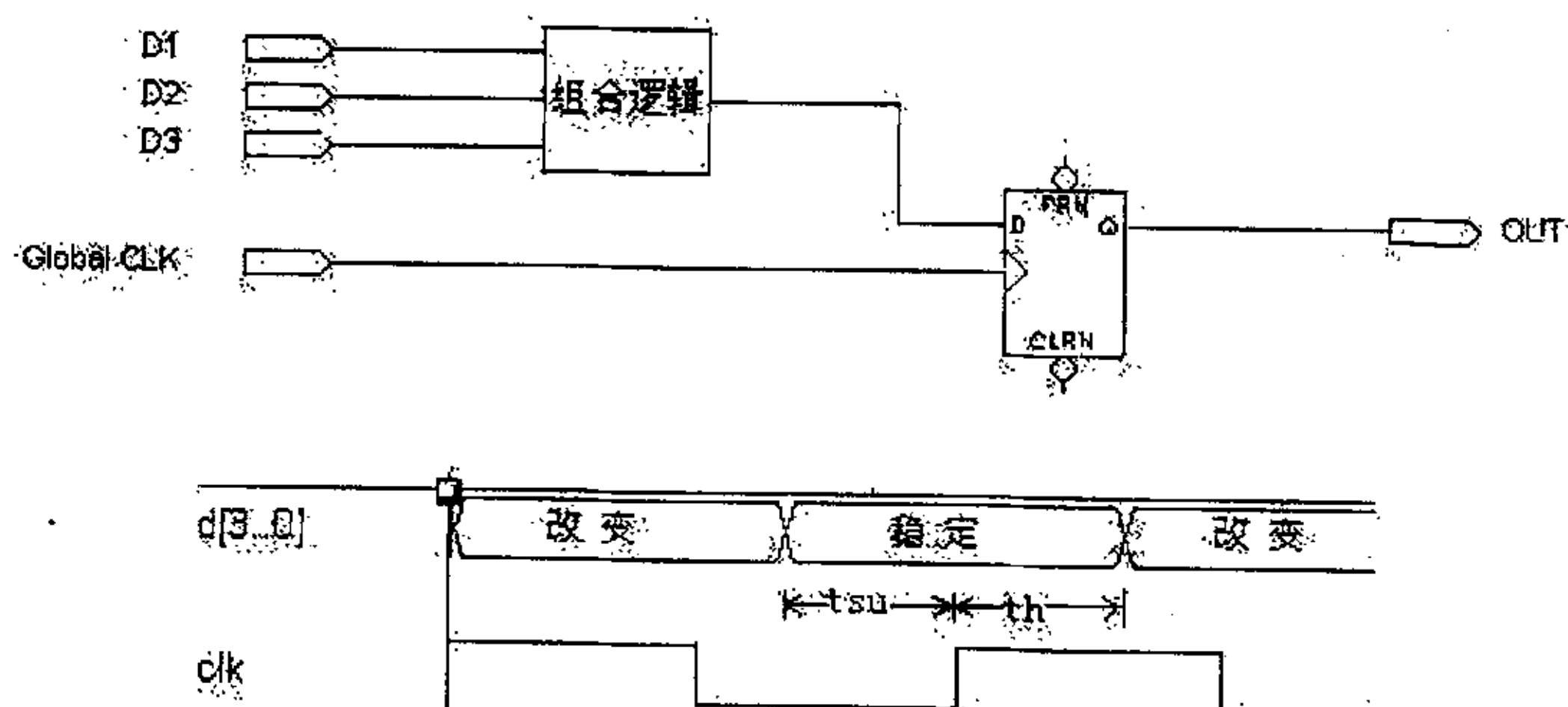


图 4-14 全局时钟

图 4-14 示出全局时钟的实例。图 4-14 定时波形示出触发器的数据输入  $D[1..3]$  应遵守建立时间和保持时间的约束条件。建立和保持时间的数值工艺库中给出。如果在应用中不能满足建立和保持时间的要求，则必须用时钟同步输入信号。

### (2) 门控时钟



在许多应用中，整个设计项目都采用外部的全局时钟是不可能或不实际的。门控时钟是由逻辑产生的，每当用组合逻辑钟控触发器时，通常都存在着门控时钟。如果符合下述条件，门控时钟可以象全局时钟一样可靠地工作：一是驱动时钟的逻辑必须只包含一个“与”门或一个“或”门。如果采用任何附加逻辑，在某些工作状态下，会出现竞争产生的毛刺。二是逻辑门的一个输入作为实际的时钟，而该逻辑门的所有其它输入必须当成地址或控制线，它们遵守相对于时钟的建立和保持时间的约束。图 4-15 和图 4-16 是可靠的门控时钟的实例。在图 4-15 中，用一个“与”门产生门控时钟，在图 4-16 中，用一个“或”门产生门控时钟。在这两个实例中，引脚  $nWR$  和  $nWE$  考虑为时钟引脚，引脚  $ADD[0..3]$  是地址引脚，两个触发器的数据是信号  $D[1..n]$  经随机逻辑产生的。

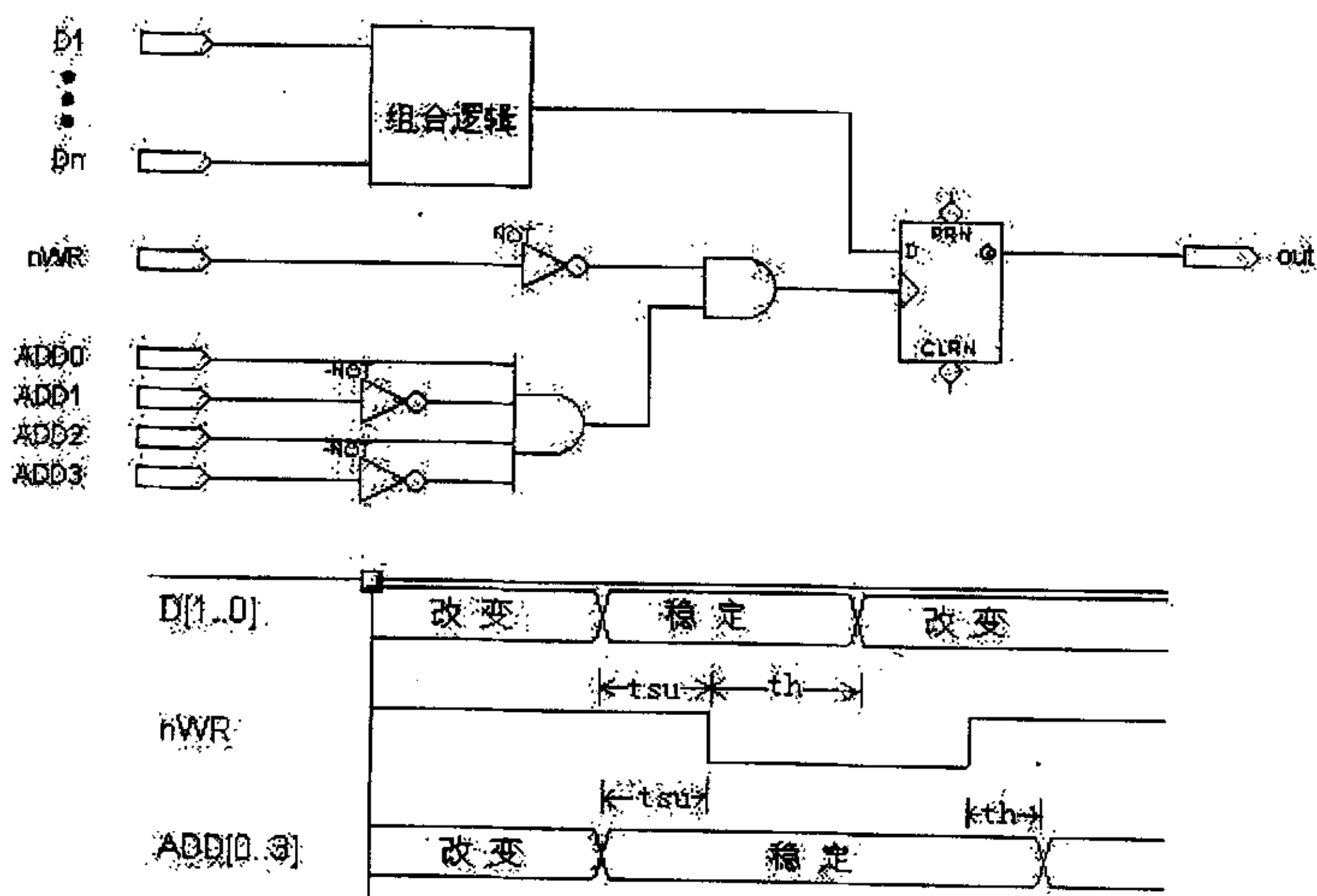


图 4-15 “与”门门控时钟

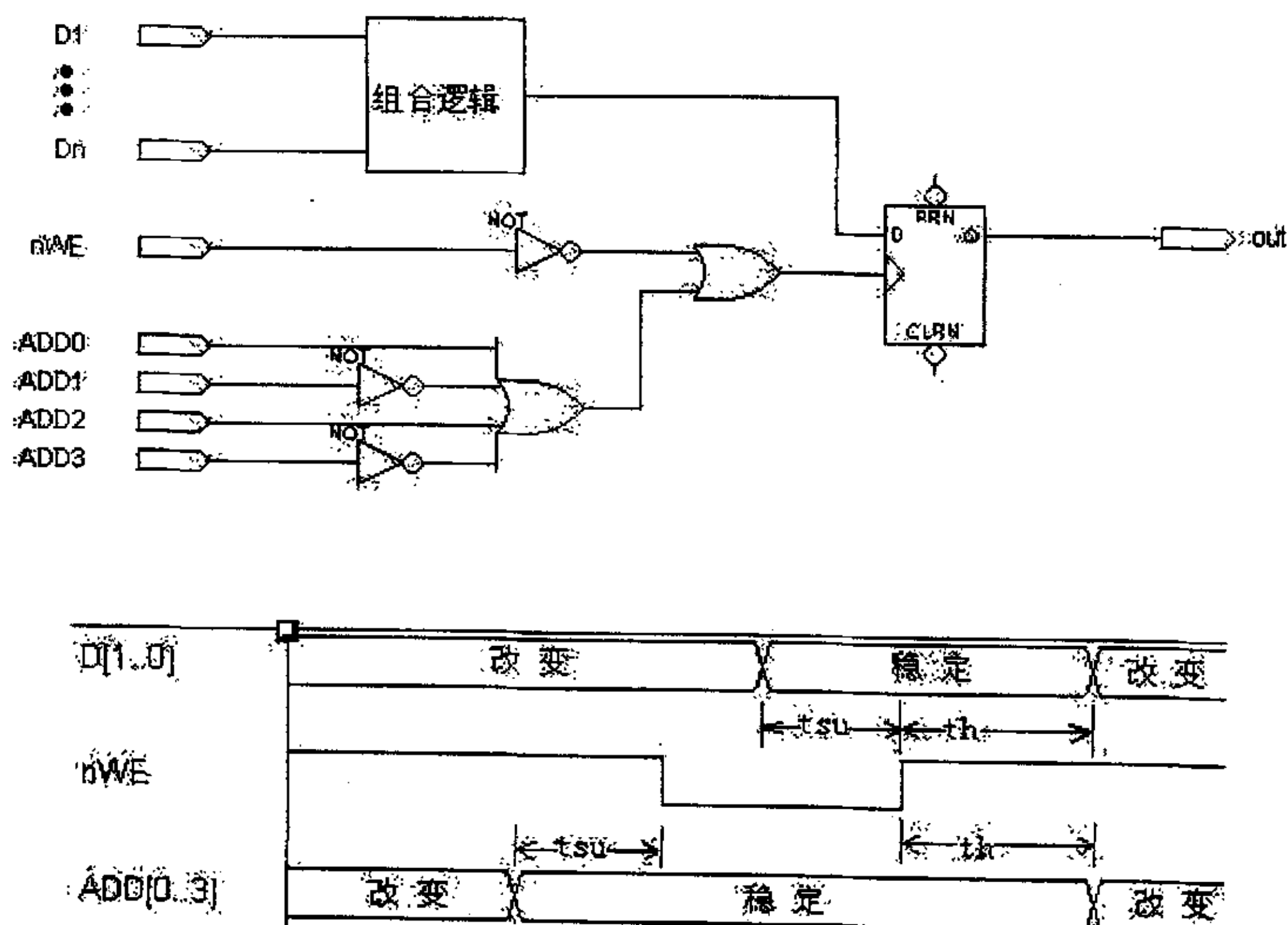


图 4-16 “或”门门控时钟

图 4-15 和图 4-16 的波形图显示出有关的建立时间和保持时间的要求。这两个设计项目的地址线必须在时钟保持有效的整个期间内保持稳定( $nWR$  和  $nWE$  是低电平有效)。如果地址线在规定的时间内未保持稳定,则在时钟上会出现毛刺,造成触发器发生错误的状态变化。另一方面,数据引脚  $D[1..n]$  只要求在  $nWR$  和  $nWE$  的有效边沿处满足标准的建立和保持时间的规定。我们往往可以将门控时钟转换成全局时钟以改善设计项目的可靠性。图 4-17 示出如何用全局时钟重新设计图 4-15 的电路。当  $ENA$  为高电平时,  $D$  输入端的值被钟控到触发器中;当  $ENA$  为低电平时,维持现在的状态。

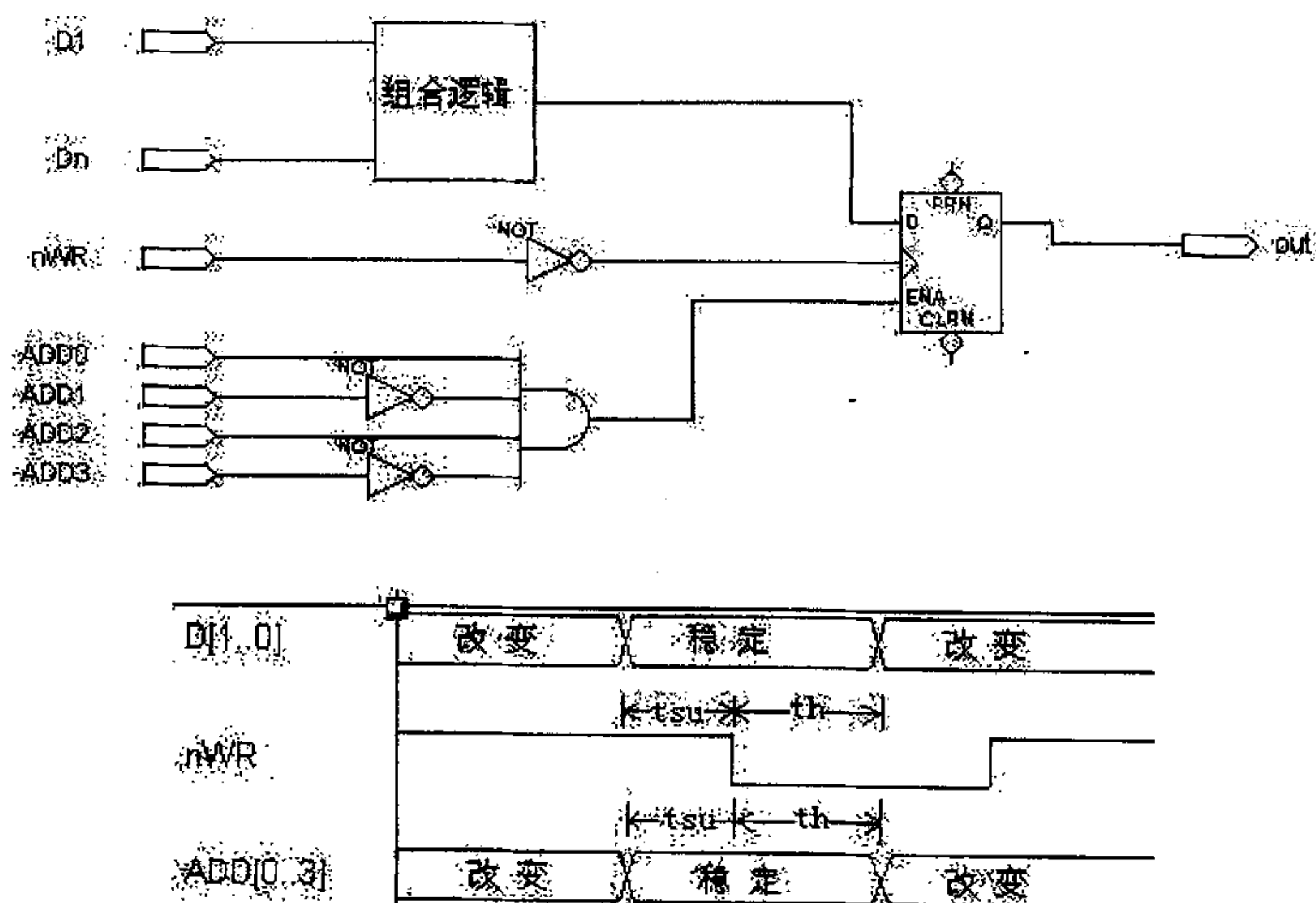


图 4-17 “与”门门控时钟转化成全局时钟

图 4-17 中重新设计的电路的定时波形表明地址线不需要在  $nWR$  有效的整个期间内保持稳定；而只要求它们和数据引脚一样符合同样的建立和保持时间，这样对地址线的要求就少很多。

图 4-18 给出一个不可靠的门控时钟的例子。3 位同步加法计数器的 RCO 输出用来钟控触发器。然而，计数器给出的多个输入起到时钟的作用，这违反了可靠门控时钟所需的条件之一。在产生 RCO 信号的触发器中，没有一个能考虑为实际的时钟线，这是因为所有触发器在几乎相同的时刻发生翻转。而我们并不能保证 QA, QB, QC 到 D 触发器的布线长短一致，因此，如图 4-18 的时间波形所示，在器从 3 计到 4 时，RCO 线上会出现毛刺（假设 QC 到 D 触发器的路径较短，即 QC 的输出先翻转）。

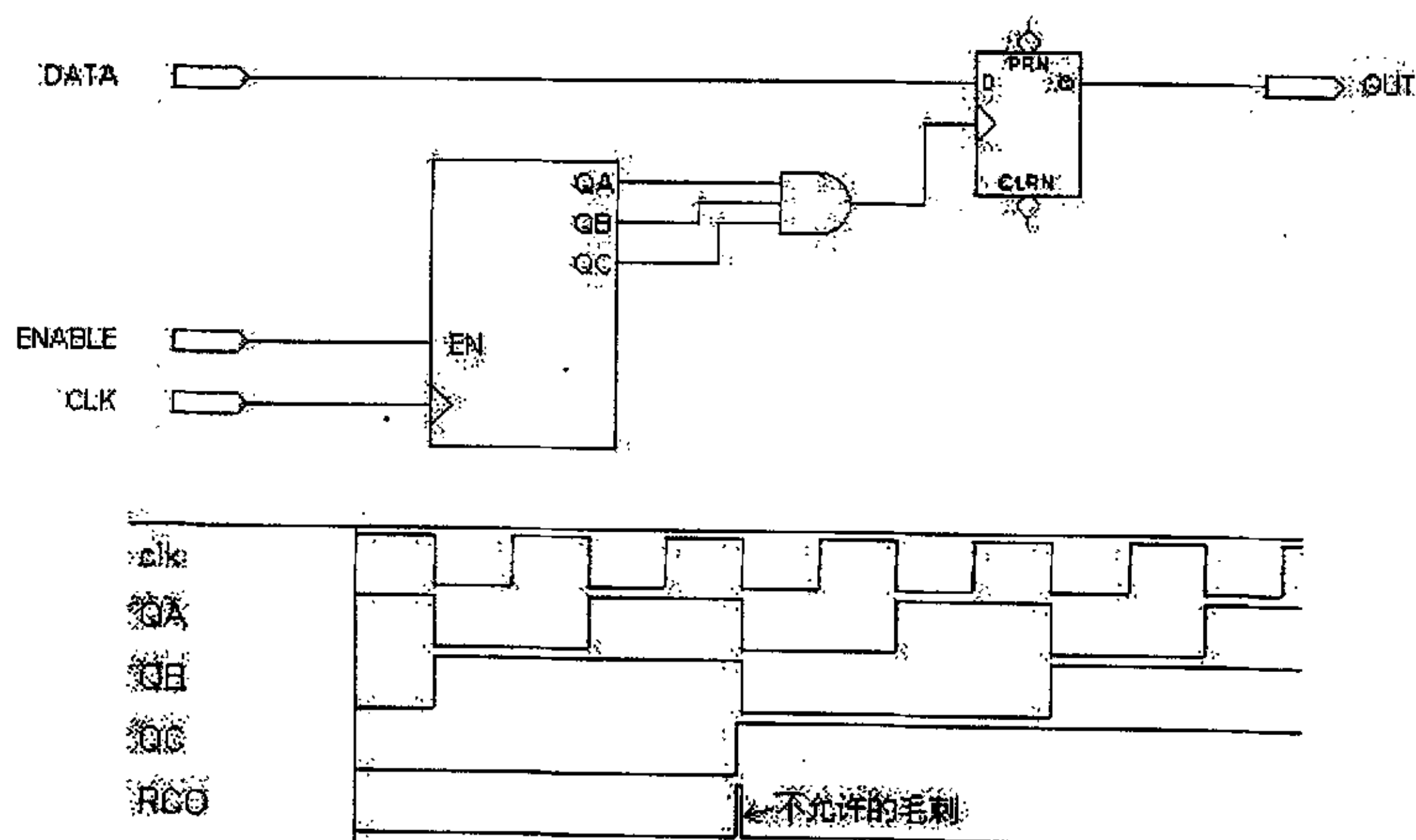


图 4-18 不可靠的门控时钟

图 4-19 给出一种可靠的全局钟控的电路，它是图 4-18 不可靠计数器电路的改进，RCO 控制 D 触发器的使能输入，这个改进不需要增加 PLD 的逻辑单元。这个电路等效于图 4-18 电路，但却可靠的多。

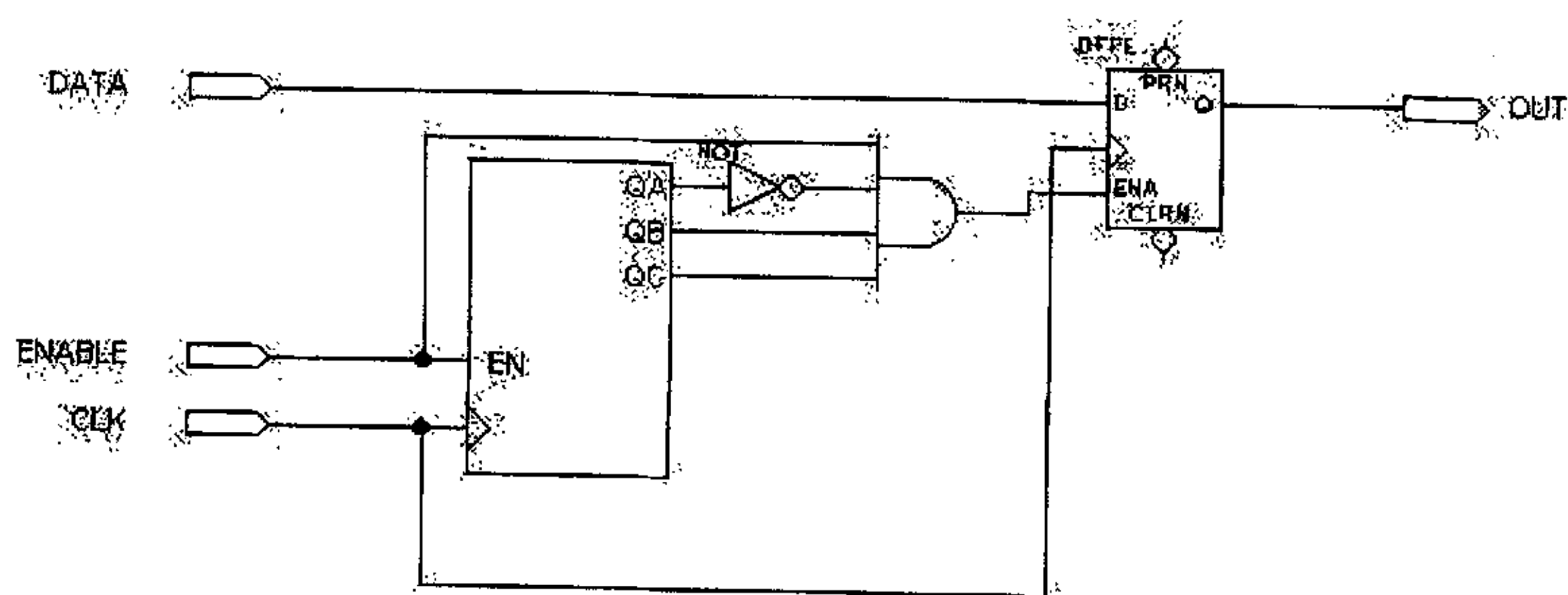


图 4-19 不可靠的门控时钟转换为全局时钟

## 4.7.3 存储器设计问题

在本设计中，如前所述，将要用到一个  $64 \times 15\text{bits}$  的双端口随机存取存储器 (RAM)。而我们知道，RAM 是由一系列特殊的门阵列电路所构成，在工艺上也与

一般的 ASIC 电路不同,是由存储器设计工程师来设计的<sup>[41]</sup>。实际的 RAM 器件是直接由 IC 产家所提供的,在做数字电路设计时,可以编写一个 RTL 级的 RAM 模型用于功能仿真,设计的仿真模型在接口信号,位宽和大小上必须完全与产家提供的相吻合。而对于综合,存储器是不需要进行的,可将 RTL 级的 RAM 屏蔽掉,屏蔽的方法是在要屏蔽的 RTL 代码之前和之后分别加上 `//synopsys translate_off` 和 `//synopsys translate_on` 语句,综合器遇到这个语句,将不会对中间的内容进行综合。综合得出的面积结果是不包含 RAM 面积的,所以总的面积应是综合得出的面积加上产家提供的 RAM 的面积参数。在进行门级仿真验证时, RAM 模型需要用产家提供的带有标准延时信息的仿真模型<sup>[42]</sup>。这样,才能正确的模拟设计的时序。



## 5 综合和验证

### 5.1 RTL 级功能仿真

功能仿真是设计验证的主要形式，其目的是为了验证设计逻辑的正确性和完备性。搭建的测试环境质量和测试激励的充分性决定了功能仿真的质量和效率。功能仿真包括行为级仿真，RTL 代码级仿真，综合后门级仿真和版图后门级仿真等几个层次，贯穿于整个设计流程始终，是 IC 设计中非常重要的一环<sup>[43]</sup>。

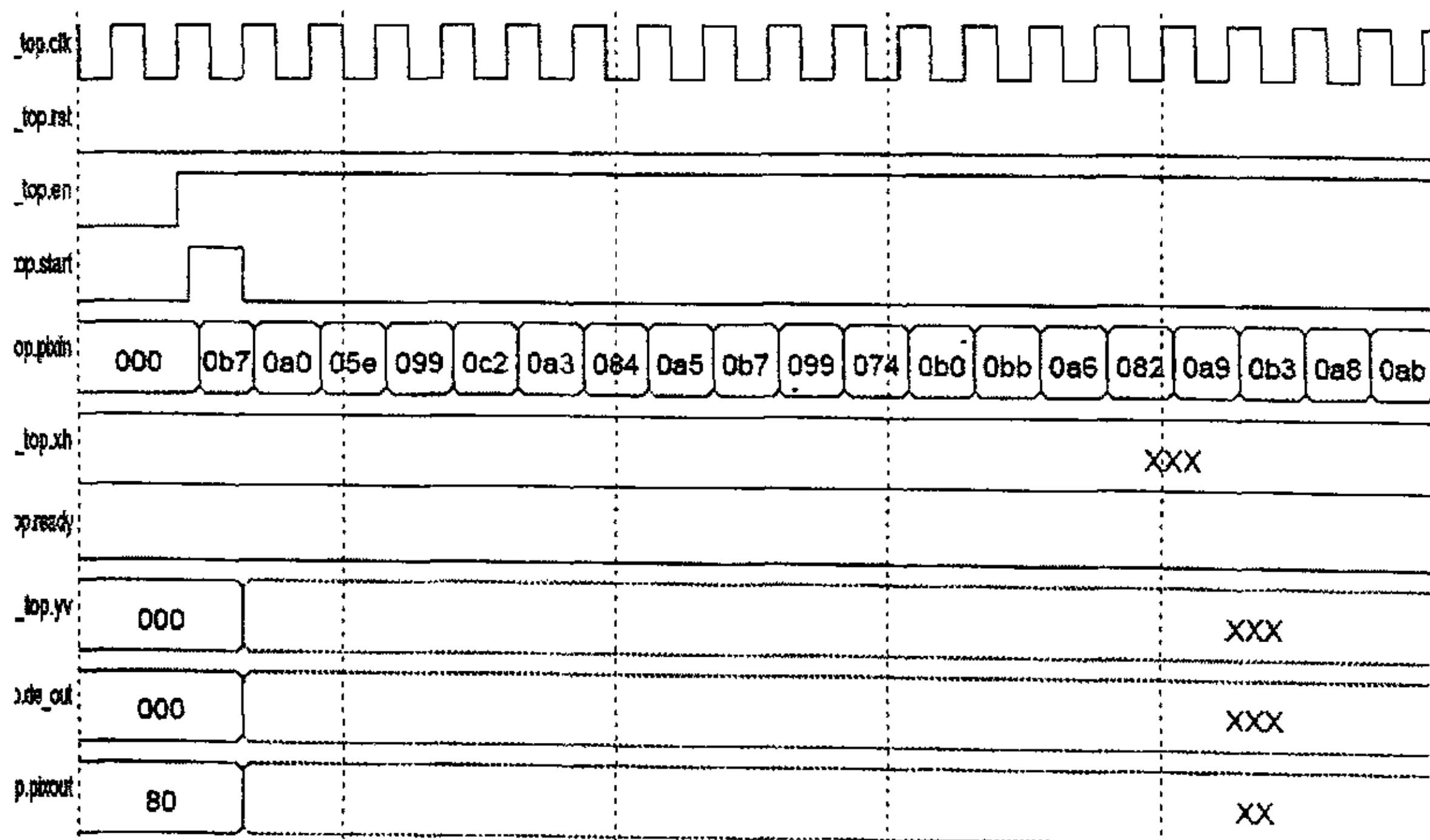
在完成 Verilog 代码设计后，首先需要对代码进行功能仿真，以验证代码设计是否实现了想要实现的逻辑功能。仿真是在 SUN Solaris/Unix 平台上，用 Cadence 设计公司的 Verilog-xl 仿真工具进行的。

对 DCT 变换的逻辑功能仿真

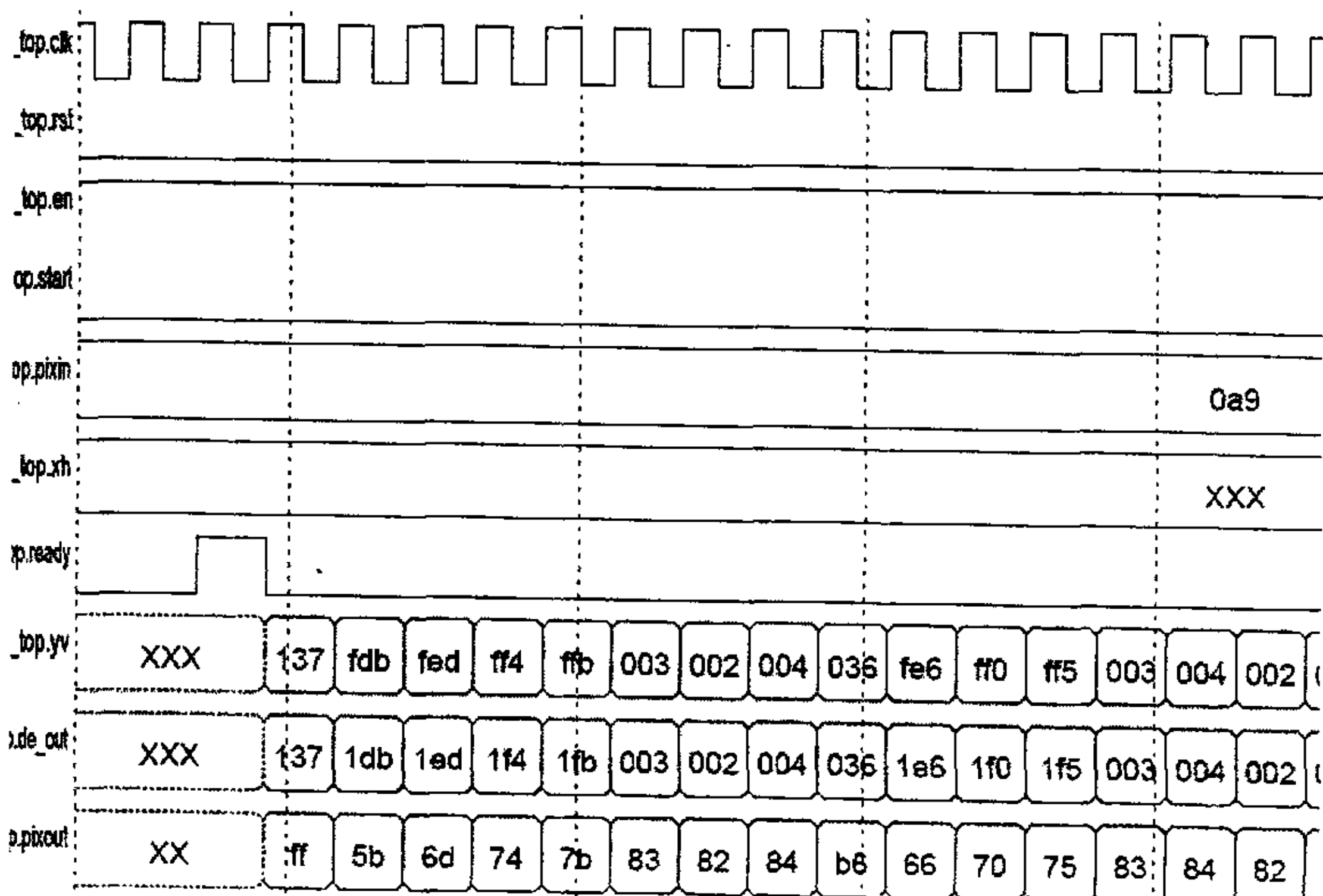
#### 5.1.1 对 DCT 变换的逻辑功能仿真

DCT 逻辑功能仿真的输入数据来源于普通的影像文件，由 C 语言编码器对其编码，在其中的 DCT 变换之前截取我们仿真需要的输入数据进行仿真。然后，将仿真的输出结果与软件代码 DCT 变换后的结果做比较，以验证 DCT 逻辑变换功能是否正确。经过对大量数据的仿真实验，我们获得了期望的结果。实验结果与 C 语言软件代码的结果基本一致，但存在一定的误差，这种误差是由于计算中，对乘法结果尾数及余弦系数尾数舍去的不同产生的，误差的大小是非常小的，是完全可以满足视频应用要求的。

图 5-1 是对 DCT 变换逻辑功能仿真波形的截取。在此次仿真中，输入数据为图 2-1(a)中的数据，以十六进制形式输入，如图 5-1(a)中的 pixin 信号所示。变换结果的输出如图 5-1(b)的 yv 信号所示，以十六进制补码形式表示，其十进制数据形式在图 5-2(a)中示出。图 5-2 是仿真的结果比较，5-2(a)是软件计算产生的 DCT 输出，5-2(b)是本设计的仿真结果，从中可以看出，设计是符合 DCT 变换逻辑的。



(a)数据输入部分波形



(b)数据输出部分波形

图 5-1 DCT 逻辑功能仿真波形

313	56	-27	18	78	-60	27	-27
-38	-27	13	44	32	-1	-24	-10
-20	-17	10	33	21	-6	-16	-9
-10	-8	9	17	9	-10	-13	1
-6	1	6	4	-3	-7	-5	5
2	3	0	-3	-7	-4	0	3
4	4	-1	-2	-9	0	2	4
3	1	0	-4	-2	-1	3	1

311	54	-29	16	76	-61	26	-27
-37	-26	14	46	33	0	-23	-10
-19	-16	11	34	22	-5	-16	-9
-12	-11	7	14	7	-11	-14	1
-5	3	8	6	-2	-6	-5	6
3	4	1	-2	-6	-3	1	3
2	2	-3	-4	-11	-2	2	3
4	3	1	-2	0	0	4	2

(a)软件计算结果

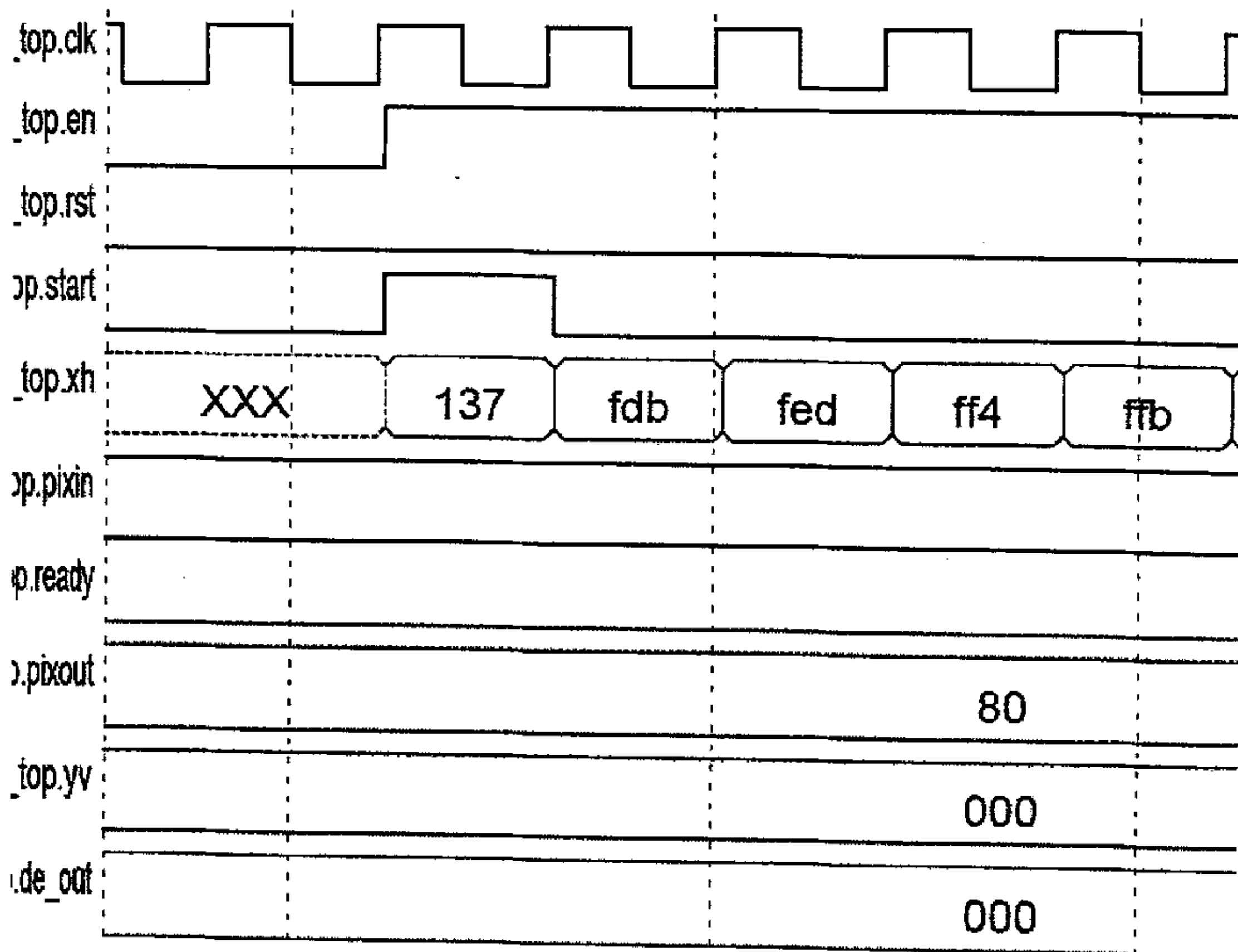
(b)仿真输出结果

图 5-2 DCT 逻辑功能仿真输出结果比较

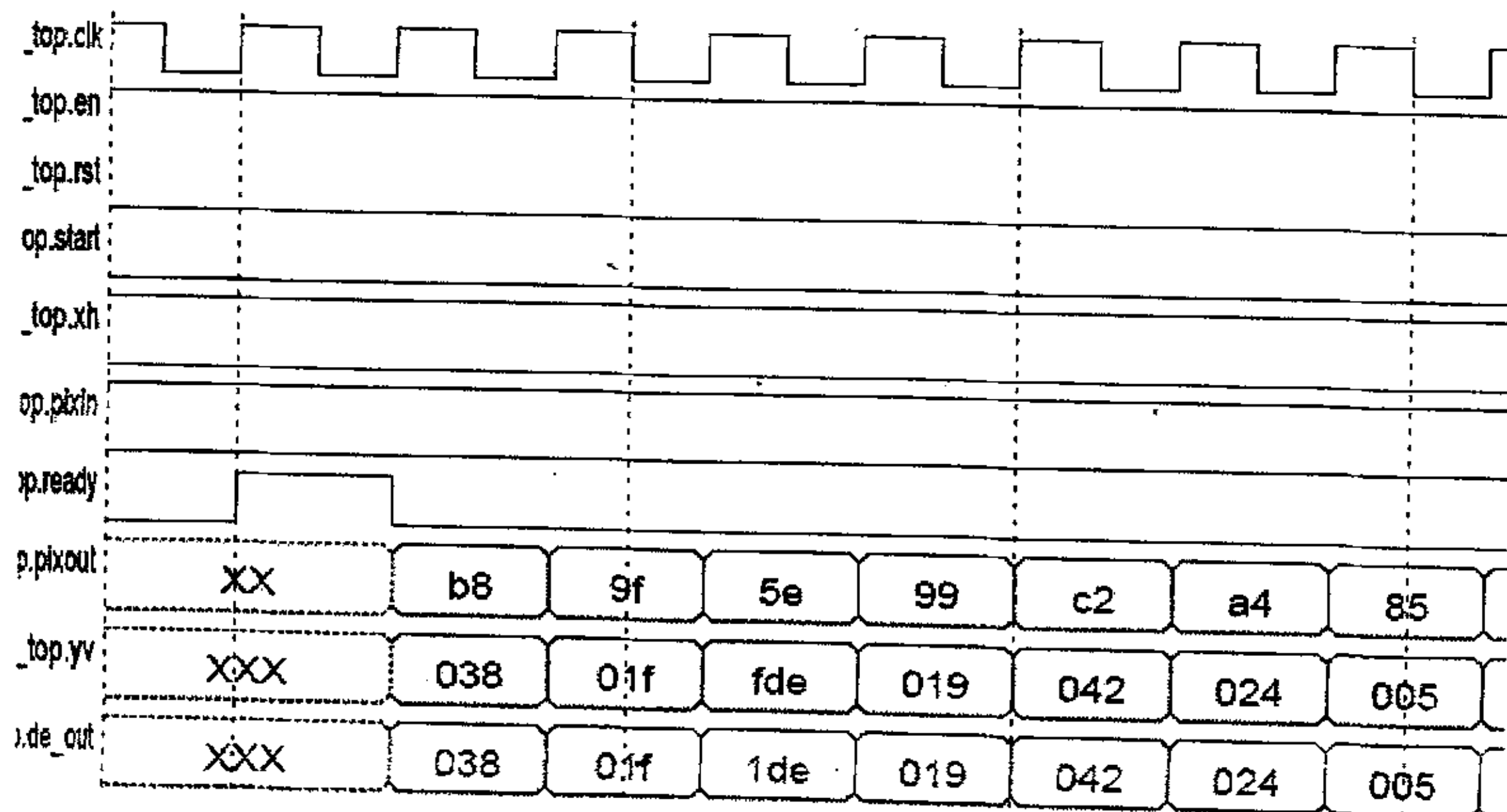
## 5.1.2 对 IDCT 变换的逻辑功能仿真

DCT 逻辑功能仿真的输入数据来源于 MPEG-1 压缩影像文件，由 C 语言解码器对其解码，和 DCT 逻辑功能仿真一样，在其中的 IDCT 变换之前截取我们仿真需要的输入数据进行仿真。然后，将仿真的输出结果与软件代码的 IDCT 变换后的结果做比较，以验证 IDCT 逻辑变换功能是否正确。经过对大量数据的仿真实验，我们同样获得了与软件解码一致的结果，从而验证了代码的逻辑功能。

图 5-3 是对 IDCT 变换逻辑功能仿真波形的截取。在此次仿真中，输入数据为图 2-1(b)中的数据，以十六进制补码形式输入，如图 5-3(a)中的 xh 信号所示。变换结果的输出如图 5-3(b)的 pixout 信号所示，以十六进制形式表示，在图 5-4 中示出。图 5-4 是仿真结果比较，(a)为软件仿真结果，(b)为硬件仿真结果。从图中可以看出，结果基本一致，从而验证了设计的 IDCT 变换逻辑。



(a)数据输入部分波形



(b)数据输出部分波形

图 5-3 IDCT 逻辑功能仿真波形

183	160	94	153	194	163	132	165
183	153	116	176	187	166	130	169
179	168	171	182	179	170	131	167
177	177	179	177	179	165	131	167
178	178	179	176	182	164	130	171
169	180	180	179	183	169	132	169
179	179	180	182	183	170	129	173
180	179	181	179	181	170	130	169

(a)软件计算结果

184	159	94	153	194	164	133	165
183	153	115	176	187	167	130	169
179	168	171	182	180	171	130	167
177	177	179	177	179	166	131	167
177	178	179	176	182	164	131	171
169	180	181	178	183	169	132	170
179	179	180	182	183	170	129	173
180	180	181	179	181	170	130	169

(b)仿真输出结果

图 5-4 IDCT 逻辑功能仿真输出结果比较

### 5.1.3 对控制信号的逻辑功能仿真

设计中的控制信号主要包括异步复位信号 `reset`，同步复位信号 `sw_res`，使能信号 `en`，DCT/IDCT 选择信号 `idct`，输出准备信号 `ready`。对于异步和同步复位信号，所要实现的都是使 DCT/IDCT 功能模块复位。所谓异步复位，也叫硬件复位，即复位信号 `reset` 与硬件开关直接相连，直接受开关控制产生复位，而与系统时钟无关；所谓同步复位，也叫软件复位，即复位信号 `sw_res` 由处理器的寄存器产生，由处理器软件代码触发，其复位有效只在时钟上升边沿。使能信号 `en` 是控制 DCT/IDCT 模块是否工作的信号，当 `en` 为“0”时，DCT/IDCT 处理核的各个寄存器保存当前值，输出也保持当前值，而当 `en` 为“1”时，模块开始根据时钟信号工作，根据输入产生相应的逻辑；`idct` 信号是模块实现 DCT 或 IDCT 逻辑的选择信号，如图 5-1 和 5-3 所示，在进行 IDCT 变换时，`idct` 信号为高电平，在进行 DCT 变换时为低电平；`ready` 信号的设计是为了让外部逻辑知道模块何时开始输出数据，如图 5-1 和 5-3 所示，它在数据输出前一个时钟将产生一个周期的高电平，之后数据开始输出。在对 DCT 和 IDCT 逻辑功能仿真的同时，我们也对这些控制逻辑做了仿真，验证了它们的逻辑功能，完全合乎设计思想。



## 5.2 综合

逻辑综合是把硬件描述语言转化为门级网表。综合的过程其实就是翻译加优化。翻译就是把设计的 HDL 描述转化为工艺库元件组成的逻辑电路，此时的电路网表包含了相关的工艺参数；优化就是根据设计者设定的延迟、面积、线负载模型等综合约束条件对电路网表进一步优化的过程。图 5-5 是对综合过程的说明。从图中可以看到，综合工具根据设定的约束条件，完成可综合的 RTL 描述到综合库单元之间的映射，得到一个门级网表文件。综合工具可内嵌静态时序分析工具，可以根据综合约束来完成门级网表的时序优化和面积优化。

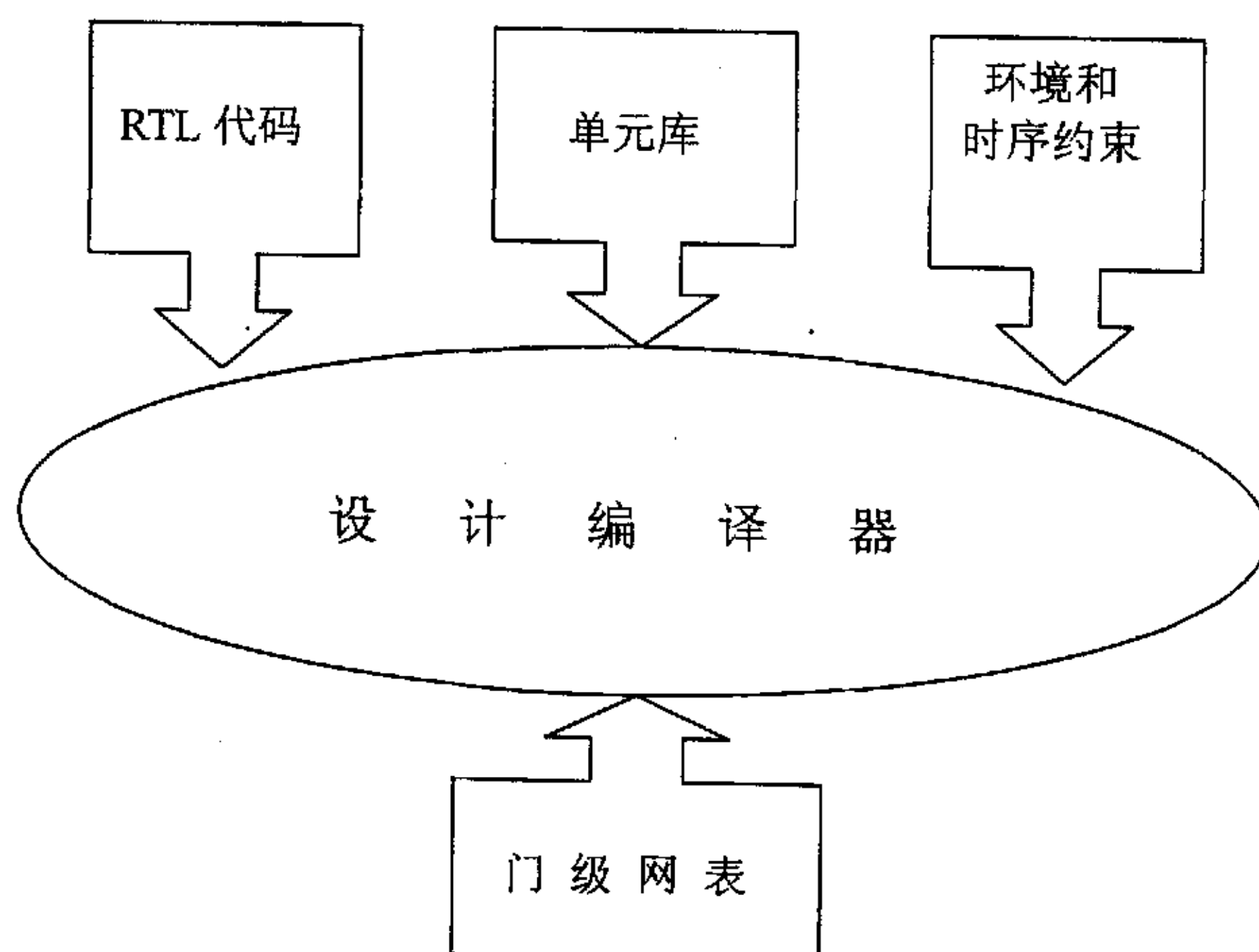


图 5-5 综合过程示意图

综合约束是综合器执行优化的依据。约束和面积的基本关系是：约束越宽松，面积越小。过分苛刻的约束，将导致面积显著增加；约束和时间延时的基本关系是：约束越明确，时间延时就越小，这是因为约束条件一般都是为了减小最大延时。约束宽松到一定阶段，延时也不会显著增加。

在 RTL 代码通过功能仿真后，开始进入综合流程。综合使用的工具是 Synopsys

公司的设计编译器 (Design Compiler), 工艺库采用旺宏电子 0.35um 工艺库。综合采用自底向上 (bottom\_up) 的编译方法进行编译。综合约束以取得最高时钟频率为目的, 经过多次编译, 最大时钟频率可达到 100M。在 100M 时钟频率下, 除去所用到的 RAM 外, 其余部分面积为  $554770.90625\mu\text{m}^2$ 。其中, 每个乘法模块的面积约为  $89522\mu\text{m}^2$ , 一个累加器模块的面积约为  $142938\mu\text{m}^2$ 。两个乘法模块在设计中总共约占 33% 的面积, 而两个累加器模块约占设计 52% 的面积。

综合后, 对设计进行了时序分析, 在采用 slow.db 库进行编译后, 100M 时钟下所有路径的延迟都能满足要求。采用 slow.db 分析时序验证了时序建立时间 (setup time) 满足要求。在采用 fast.db 库进行编译后, 100M 时钟频率下, 保持时间 (hold time) 也满足要求。图 5-6 是 DCT/IDCT 综合后的总体电路结构。

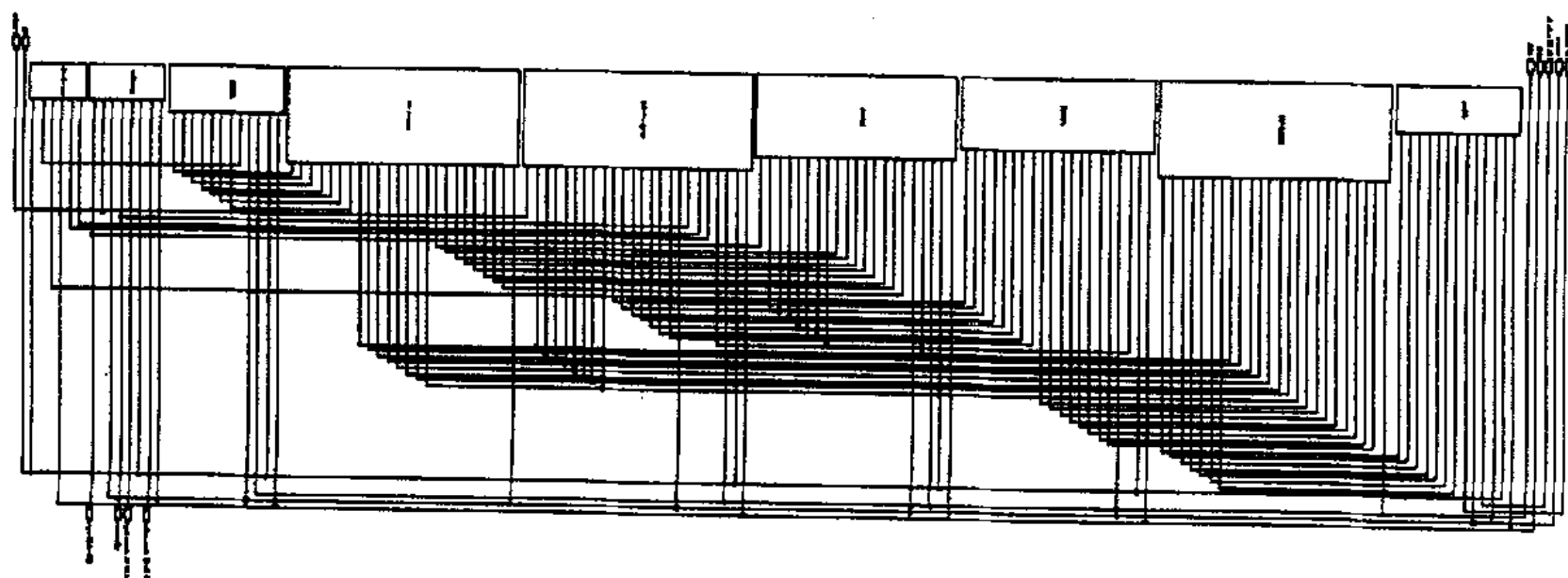


图 5-6 DCT/IDCT 综合电路图

## 5.3 门级仿真验证

在综合编译后, 对建立时间和保持时间进行时序分析, 如果时序满足, 就可进入门级功能仿真验证。如上所述, 在综合阶段, 已对时序进行了分析, 能够满足 100M 时钟频率要求。此时, 将综合后产生的标准延时 (sdf) 文件和门级网表文件, 库 verilog 模型文件及测试文件一起进行仿真。测试文件内容与功能仿真时相同, 但时钟频率设定为 100M。

图 5-7 和图 5-8 是在 100M 时钟频率下分别对 DCT 逻辑和 IDCT 逻辑的门级仿真波形图的截取。图中示出了仿真的输出数据, 从图中可以看出, 仿真的结果与

图 5-1(b)和图 5-3(b)所示的功能仿真的结果完全一致,从而验证了综合出的门级网表逻辑功能的正确性。

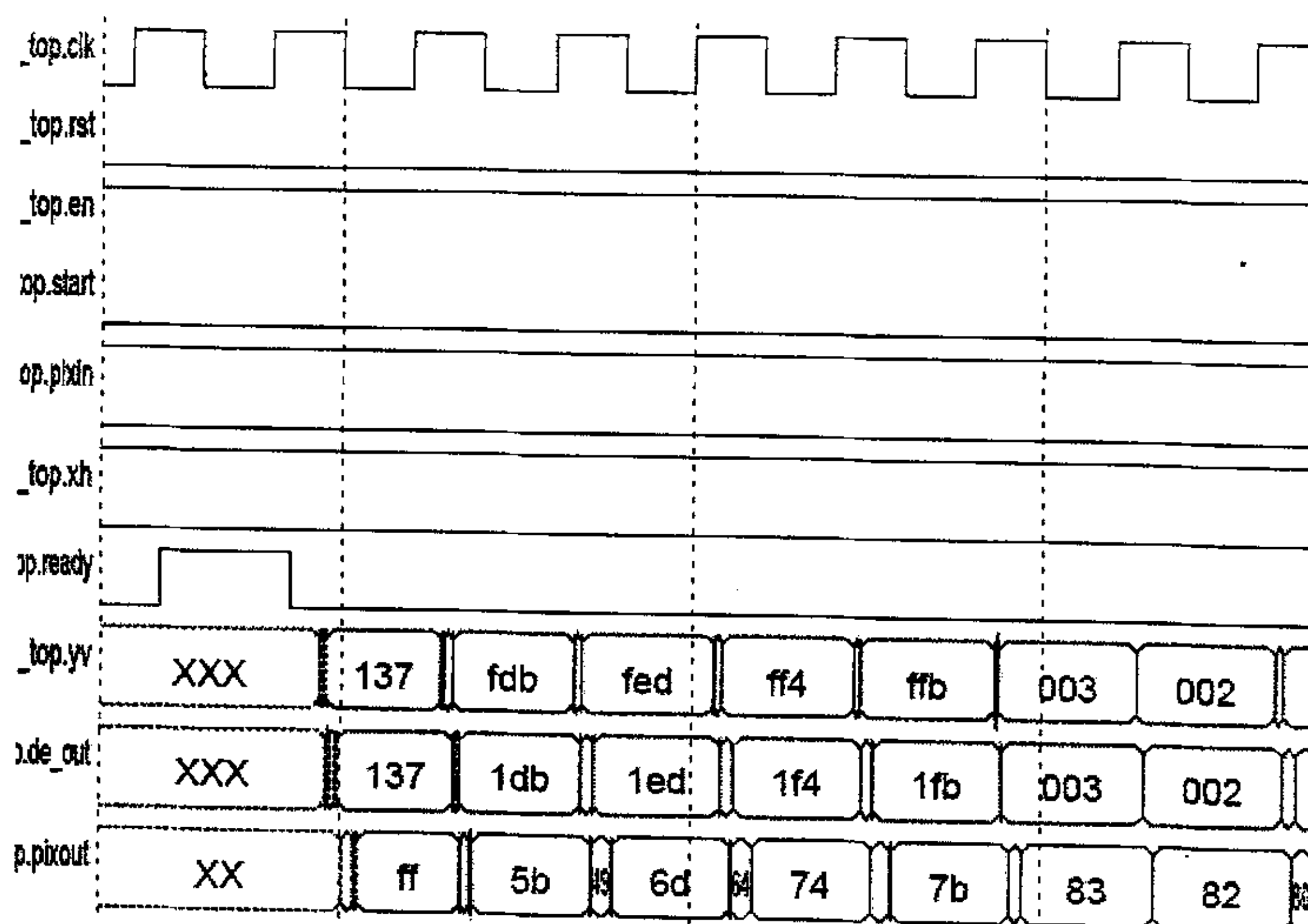


图 5-7 DCT 逻辑的门级仿真波形

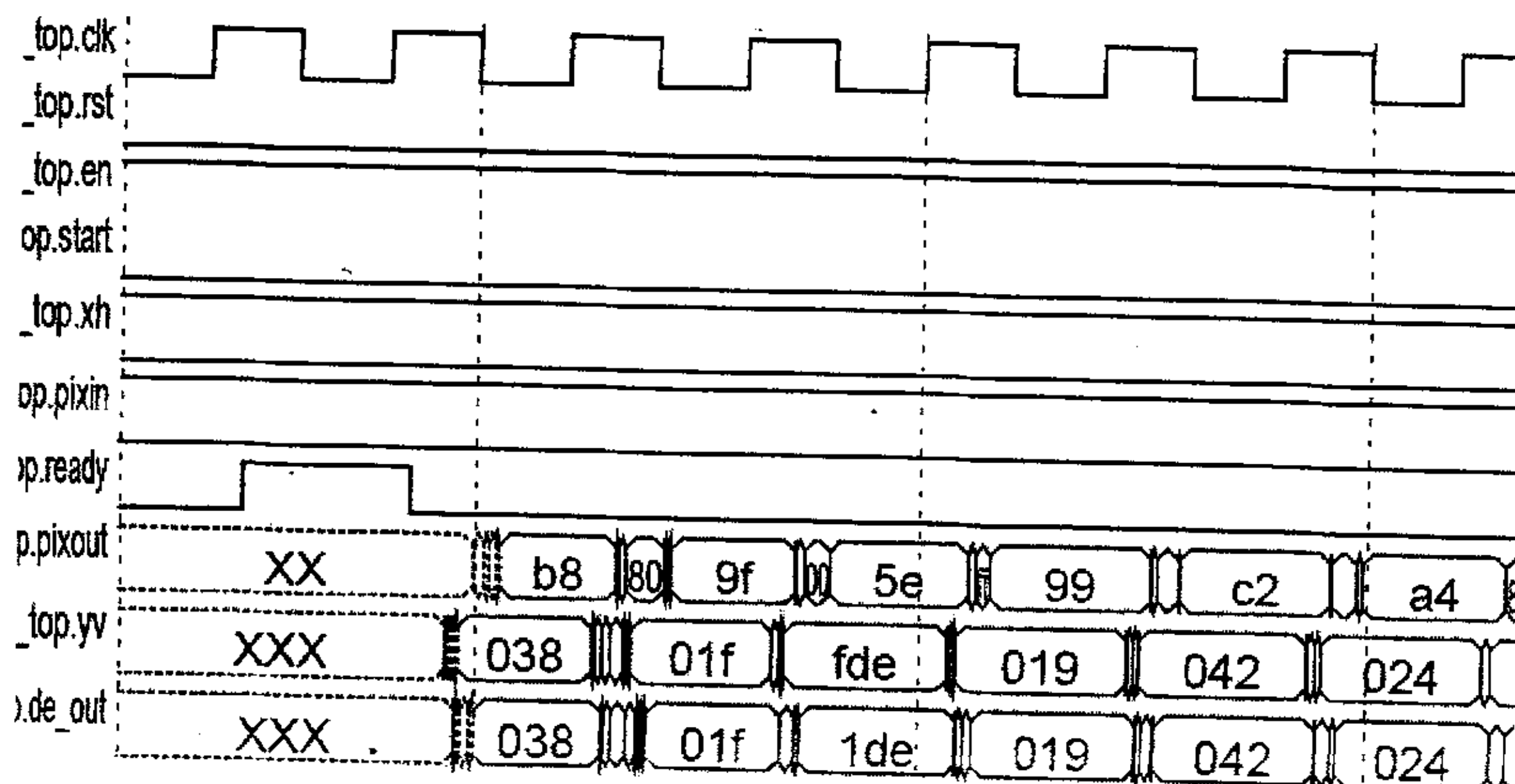


图 5-8 IDCT 逻辑的门级仿真波形

## 5.4 实时性分析

由以上综合和验证结果得知,设计的DCT/IDCT处理核的工作频率可达到100M。因为本文的流水线设计结构,每一个时钟就可以处理一个数据,所以,这意味着每秒可以处理100M个数据。对于MPEG1 PAL制式的 $352 \times 288 \times 25$ 的图像,其数据量约为2.5M/每秒,本文的设计可以容易的满足处理速度的要求。即便对于CCIR格式、PAL制、4:4:4采样的数字电视图像,其每帧数据量为 $720 \times 576 \times 3 = 1.24\text{M}$ ,每秒数据量为 $1.24 \times 25 = 31.3\text{M}$ ,本文的设计也能轻松满足实时要求。

## 5.5 计算精度分析

在进行DCT运算时,假设采用8位的采样精度,第一维运算的四舍五入截位处理会引入0.5个灰度等级的误差(8位样本值有256个灰度等级)。在进行第二维DCT运算时,误差将会被放大,含有0.5个灰度等级误差的中间值将乘以一个余弦因子和系数1/2后,8个中间值再进行相加并四舍五入,8次相加的结果的误差为:系数 $\times$ 单项误差值 $\times$ 余弦因子值(最大为 $\cos\frac{\pi}{4}$ ) $\times$ 项数 $+$ 四舍五入误差,代入具体值后为: $0.5 \times 0.5 \times 0.7 \times 8 + 0.5 = 1.9$ 个灰度等级。计算精度带来的误差,将表现为图像的加性噪声,从而造成图像压缩质量的下降。降低计算精度误差的有效方法是采用二次截位法,即在运算过程中保持高精度,在计算完成后再舍去多余尾数。在本文的设计中,对输入数据取12位精度采样,并在中间运算中保持了较高精度,对第二维变换输出结果再进行二次截位(对DCT运算,截位到8位;对IDCT运算,截位到12位)。这样,可以有效的把运算精度控制在系统误差以内。

## 6 总结

离散余弦正反变换(DCT/IDCT), 由于其运算量很大, 因而在一些高速或实时场合, 能否快速实现DCT或IDCT变换就成为一个关键因素, 硬件实现显然是满足实时和高速应用的良好解决办法。在考虑硬件实现时, 主要考虑计算的复杂度, 速度, 芯片面积, 精度等因素, 同时在设计中还要考虑结构的规则化和模块IP化, 以方便后端设计和移植。硬件实现的方法多种多样, 总是在速度和面积, 功耗与速度等几个因素间折中, 具体的要根据实际的应用要求来选择合适的设计方案, 在这些矛盾的因素中进行合适的取舍, 才能设计出一个高性价比的电路。对当今的ASIC设计而言, 逻辑功能的实现已经不是最大的挑战, 真正的挑战是怎样在最短的时间内设计出性能好的电路(面积小, 功耗低, 速度快)。由于应用的目的不同, 采用的算法不同, 现在有很多DCT/IDCT的ASIC实现方法, 所以对算法的VLSI实现很难找到一个确切的衡量标准。

本文主要是研究针对以MPEG和JPEG等图像编解码应用为目的的DCT/IDCT IP设计方法和实现。通过对算法的比较、选择和研究, 并与硬件实现的特点相结合, 本文提出了新的设计思想, 运用移位和加法的逻辑运算来代替乘法运算。因此, 本文设计的DCT/IDCT处理核与传统的用乘法器或是查找表来实现乘法的处理核相比, 大大节约了芯片面积, 而且本文设计的乘法实现是单周期完成运算的, 比之采用多时钟的乘法器, 还提高了速度, 减少了延迟。

本设计的另外一个特点就是IP化, 具有比较强的通用性, 既可以进行DCT变换又可以进行IDCT变换。设计提供了为实现暂停所需要的外部控制信号输入端口和复位输入端口, 以及输出控制端口, 能够容易的与其它设计接口, 适合SOC设计应用。

经过功能仿真、逻辑综合和门级仿真验证, 结果表明, 本文的设计合乎设计思想, 逻辑功能正确, 最高时钟频率可达 100M。在 100M 时钟频率下, 除所需  $64 \times 15$ SRAM 外的面积为  $55477.90625\mu\text{m}^2$ 。

## 致 谢

三年前，沈绪榜导师和陈朝阳导师将我从一个集成电路设计的门外汉引入 IC 设计之门，并在学习和研究上进行指导，在生活上给予关心。集成电路设计是一门非常精妙的艺术，在三年的学习和研究工作中，我有过迷惑，有过彷徨，有过挫折，是陈朝阳老师及时的给予帮助，加以引导，使我克服了一个个困难，积累了大量的知识和许多宝贵的经验，最重要的是沈绪榜，陈朝阳和邹雪城等老师严谨的治学态度和不畏困难的研究精神更潜移默化的影响着我，使我受益匪浅。为此，我将最诚挚的向以上几位导师表示衷心的感谢！

另外，在我的学习和研究工作中，我还得到了中船重工集团第 722 研究所马晓明高级工程师的悉心指导和大力帮助。在毕业课题的设计和研究中，世宏科技有限公司的王建群主管对我的研究给予了极大的指导和帮助。在此，也要对这二位工程师表示诚挚感谢！

我还要感谢郑兆青博士在我学习和研究中给予的关心和无私的帮助，感谢实验室秦付莘，郭胜江，胡晓波，孙建伟等同学的帮助和合作。感谢宿舍舍友杜长江同学在生活和学习上的关心与合作。

深深感谢我的父母对我一如既往的无私支持和深深的爱。感谢我的女友夏柏琳对我的鼓励和一如既往的关心支持。

论文的完成，离不开上述老师，同学，亲人的帮助和关心。再次向他们及所有关心支持过我的人致以最衷心的感谢！



## 参考文献

- [1] Jerry D.Gibson, Toby Berger, Tom Lookabaugh et al. 多媒体数字压缩原理与标准. 第一版. 李煜晖, 朱山凤, 段上为等译. 北京: 电子工业出版社, 2000
- [2] 王云平, “DCT 算法在图像处理中的作用,” 辽宁工学院学报, 2003: 23(1), 31~33
- [3] Tekalp.A.M. 著. 崔之祜, 江春, 陈丽鑫译. 数字视频处理. 北京: 电子工业出版社, 1998
- [4] 沈兰荪, 图像编码与异步传输. 北京: 人民邮电出版社, 1998
- [5] 钟玉琢, 乔秉新, 祁卫译. 运动图像及其伴音通用编码国际标准 MPEG-2. 北京: 清华大学出版社, 1997
- [6] 蔡士杰, 岳华, 刘小燕. 连续色调静止图像的压缩与编码——JPEG. 南京: 南京大学出版社, 1995
- [7] N.Ahmed, T.Natarajan, and K.R.Rao, “Discrete cosine transform,” IEEE Trans.Comput, vol.C-23, Jan.1974, pp90~93
- [8] 陈禾, 毛志刚, 叶以正, “DCT 快速算法及其 VLSI 实现,” 信号处理, 1998; 14 (增刊), 62~68
- [9] 钟玉琢, 王琪, 贺玉文. 基于对象的多媒体数据压缩编码国际标准 MPEG-4. 北京: 科学出版社, 2000
- [10] V.Srinivasan, Ray Liu, K.J., “Full custom VLSI implementation of high-speed 2-D DCT/IDCT chip,” Image Processing, ICIP-94.IEEE International Conference, 1994, vol.3, 13-16 Nov.1994, pp606~610
- [11] A.Rosenfeld, A.C.Kark 著. 余英林等译. 数字图像处理. 北京: 人民邮电出版社, 1982
- [12] Zhongde Wang, “Fast Algorithms for The Discrete W Transform and for The Discrete fourier Transform,” IEEE Trans.Signal Processing, vol.32, no.4, Aug.1984, pp803~816

- [13] W.H.Chen, C.H.Smith and S.C.Fralick, "A fast computational algorithm for the discrete cosine transform, "IEEE Trans.Commun, vol.4, Sep.1977, pp1004~1009
- [14] H.S.Hou , "A fast recursive algorithm for computing the discrete cosine transform, "IEEE Trans.Signal Processing, vol.35, no.10, Oct.1987, pp1455~1461
- [15] 马维祯, 殷瑞祥, "DFT( $2^m$ )和 DCT( $2^m$ )的快速递归新算法, "中国电子学会电路与系统年会论文集, 深圳 1987, 26.20~26.23
- [16] 王中德, "计算离散 cosine 变换的一种算法, " 中国电子学会电路与系统年会论文集, 深圳 1987, 26.9~26.12
- [17] E.Feig and S.Winograd, "On the Multiplicative Complexity of Discrete Cosine Transforms, "IEEE Trans.Inform., vol.35, no.4, Apr.1992, pp1387~1391
- [18] C.Loeffler, A.Ligtenberg, G.S.Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications,"Proc.IEEE.ICASSP 89, 1989, pp988~991
- [19] 杨睿, 郑学仁,  $8 \times 8$  矩阵高速 DCT 的硬件实现, 半导体技术, 1999, 12(6), pp47-51;
- [20] N.L.Cho and S.U.Lee, "Fast algorithm and implementation of 2-D discrete cosine transform, "IEEE Trans.Circuits Syst., vol.38, no.3, Mar.1991, pp297~305
- [21] C.Jen-Shiun, H.Hsiang-chou, "New Architecture for High Throughput-rate Real-time 2-D DCT and the VLSI Design, "ASIC Conference and Exhibit, 1996, Proceedings., Ninth Annual IEEE International, 23-27, Sep.1996, pp219~222
- [22] W.Chin-Liang, C.Yu-Tai, "Highly parallel VLSI architectures for the 2-D DCT and IDCT computations, "TENCON'94.IEEE Region 10's Ninth Annual International Conference. Theme:'Frontiers of Computer Technology'.Proceedings of 1994, vol.1, Aug.1994, pp22~26
- [23] 山洪刚, 郑南宁, 杨国安等, "一种应用于  $8 \times 8$  二维 DCT/IDCT 的高效结构, " 半导体技术, 2002: 27(6), 13~17
- [24] A.Madisetti and A.N.Willson, "A 100MHZ 2-D  $8 \times 8$  DCT/IDCT processor for HDTV applications, "IEEE Trans.Circuits Syst, vol.2, no.2, Apr.1995, pp158~164
- [25] A.Peled and B.Liu, "A new hardware realization of digital filters,"IEEE Trans.Signal

- Processing, vol.22, no.6, Dec.1974, pp456~462
- [26] M.T.Sun, T.C.Chen and A.M.Gottlieb, "VLSI implementation of a  $16 \times 16$  discrete cosine transform, "IEEE Trans.Circuits Syst, vol.36, no.4, Apr.1989, pp610~617
- [27] H.T.Kung, "Why systolic architecture, "Computer, vol.15, no.1, 1982, pp37~46
- [28] N.L.Cho and S.U.Lee, "DCT algorithms for VLSI parallel implementations,"IEEE Trans.Signal Processing, vol.38, no.1, Jan.1990, pp121~127
- [29] N.R.Murthy and M.N.S.Swamy, "On the real-time computation of DFT and DCT through systolic architectures, "IEEE Trans.Signal Processing, vol.42, no.4, Apr.1994, pp988~991
- [30] J.E.Volder, "The CORDIC trigonometric computing technique, "IRE Trans on electronic computers., vol.8, no.3, Sep.1959, pp330~334
- [31] W.J.Duh and J.L.Wu, "Constant-rotation DCT architecture based on CORDIC techniques, "INT.J.ELECTRONIC, vol.69, no.5, 1990, pp583~593
- [32] E.P.Mariatos, D.E.Metafas, J.A.Hallas et al, "A fast DCT processor, based on special purpose CORDIC rotators, "1994 IEEE International Symposium on Circuits and Systems., vol.4, Jun.1994, pp271~274
- [33] F.Zhou and P.Kornerup, "High speed DCT/IDCT using a pipeline CORDIC algorithm, "in Computer Arithmetic.1995.Proceedings of the 12<sup>th</sup> Symposium on, Jul.1995, pp180~187
- [34] V.Srinivasan, K.J.R.Liu, "VLSI Design of High-speed Time-recursive 2-D DCT/IDCT processor for video applications, "IEEE Trans.Circuits Syst, vol.6, no.1, Feb.1996, pp87~96
- [35] Docef.A, Kossentini.F., Khanh Nquuyen-Phi et al, "The quantized DCT and its application to DCT-based video coding"IEEE Trans.Image Processing, vol.11, no.3, Mar.2002, pp177~187
- [36] S.Cucchi, M.Fratti, "A Novel architecture for VLSI Implementation of The 2-D DCT/IDCT,"Signal processing, 1992 IEEE International Conference, San
-

# 华中科技大学硕士学位论文

---

Francisco,CA,USA, vol.5, pp693~696

- [37] 康华光. 电子技术基础数字部分. 第三版. 北京: 高等教育出版社, 1979;
- [38] Jan M.Rabaey, Digital Integrated Circuits:A Design Perspective, Prentice Hall, 1998
- [39] Keshab K.Parhi. VLSI 数字信号处理系统设计与实现. 北京: 机械工业出版社
- [40] 张亮. 数字电路设计与 Verilog HDL. 北京: 人民邮电出版社, 2000
- [41] 张明. Verilog HDL 实用教程. 成都: 电子科技大学出版社, 1999
- [42] 刘志红, “设计用可综合 HDL,”电子产品世界, 1998 (4), pp58-59
- [43] 牛风举, 刘元成, 朱明程. 基于 IP 复用的数字 IC 设计技术. 北京: 电子工业出版社, 2003