原创性声明

声明：本文的核心思想及方案设计实现均为本文作者原创，除了文中特别加以标注的地方外，论文中不包含其他人已经发表和撰写过的研究成果。

电子设计题目：改进的一维 DCT 方案设计与实现

电子设计作者签名： 陈艳玲

日期： 二OO六年五月十八日

作者简介：

陈艳玲，女 ，北京大学，集成电路设计专业， 硕士研究生

E-mail：shuimutiantian@sohu.com

改进的一维 DCT 方案设计与实现

陈艳玲　北京大学硕士研究生

Abstract

This paper presented 1-D DCT architecture base on the DA, which is greatly reduce the hardware utilization. By exploiting the timing property of the DCT transform coefficients, the 1-D DCT scheme's realization use less additions. Performance of the proposed architecture is also analyzed in this paper.

The two proposed architecture is used only 13 or 11 additions, although some control component required, but the required hardware resource is less then previous 1-D DCT architecture base on the distributed arithmetic.

INDEX TERMS: Distributed Arithmetic(DA), Discrete Cosine Transform(DCT), Pipeline.

1. 引言

Ahmed，Natarajan 和 Rao 在 1974 年首先提出了 DCT 算法[1]。从那时开始，它成为了图像和视频编码最流行的算法，并被广泛应用，这主要有两个原因：首先，它把图像数据转变成容易压缩的形式；第二，它能有效地用软件和硬件实现[2]。至今，基于行列变换的 DCT 被应用最广泛，因些，有必要对一维 DCT 的硬件设计的改进并使其硬件资源达到最简。

分布式算法自提出 20 年来一直被广泛应用于 VLSI 与 DSP 中[3-8]。在它的运用过程中，大多数的运算量集成在加法器或乘法。因此，减少加法器或乘法器成了一大批学者们研究对象。

经研究发现，目前所用的分布式算法大都是基于组合电路设计的，即三级加法在同一个时钟周期完成，由于电路的最小时钟周期取决于电路中所需处理时间最多的模块，因此，基于组合电路的时钟处理频率必大大受影响，故，我们考虑用时序电路来完成基于分布式算法的一维 DCT，并且，为了加快处理速度，我们对一维 DCT 的变换数据进行了分析，发现如果改变对 DCT 各数据的处理顺序，更是可大大加快一维 DCT 的处理速度。

下面是我们对基于分布式算法的一维 DCT 的研究:

2. 分布式算法的相关知识 [3]

首先,考虑下式

$$Y=\sum_{k=1}^{L}A_k X_k \tag{2-1}$$

这里，A_k 是常数，X_k 是输入的数据，写成矩阵形式为：

$$Y=\left[A_1 A_2 \cdots A_K\right]\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \tag{2-2}$$

如果 A_k 是一个 N 比特的二进制数，则 A_k 可表示为：

$$A_k=-A_k^{N-1}2^{N-1}+\sum_{i=0}^{N-2}A_k^i 2^i \tag{2-3}$$

其中 $A_k^i=0$ or 1 而 i = 0 ,1 , … , N-1；A_k^{N-1} 表示 A_k 的最高比特位(符号位)，A_k^0 表示它的最低比特位。于是：

$$\begin{aligned}
Y&=\left[A_1A_2\cdots A_K\right]\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix}=\left[2^0 2^1\cdots 2^{N-1}\right]\begin{bmatrix} A_1^0 & A_2^0 & \cdots & A_L^0 \\ A_1^1 & A_2^1 & \cdots & A_L^1 \\ \vdots & \vdots & \cdots & \vdots \\ -A_1^{N-1} & -A_2^{N-1} & \cdots & -A_L^{N-1} \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \\
&=\left[2^0 2^1\cdots 2^{N-1}\right]\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & INV(.) \end{bmatrix}\begin{bmatrix} A_1^0 & A_2^0 & \cdots & A_L^0 \\ A_1^1 & A_2^1 & \cdots & A_L^1 \\ \vdots & \vdots & \cdots & \vdots \\ A_1^{N-1} & A_2^{N-1} & \cdots & A_L^{N-1} \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \\
&=\left[2^0 2^1\cdots 2^{N-1}\right]\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & INV(.) \end{bmatrix}\begin{bmatrix} Y^0 \\ Y^1 \\ \vdots \\ Y^{N-1} \end{bmatrix}
\end{aligned} \tag{2-4}$$

这里 INV(.)表示求补。

于是，在一维 DCT 变换公式式

$$F(u)=\frac{c(u)}{2}\sum_{x=0}^{7}f(x)\cos\left(\frac{(2x+1)u\pi}{16}\right) \tag{2-5}$$

中我们令

$$A_u(x)=\frac{c(u)}{2}\cos\frac{(2x+1)u\pi}{16} \tag{2-6}$$

则：

$$F(u)=\sum_{x=0}^{7}A_u(x)f(x)=\left[A_u(0)A_u(1)\cdots A_u(7)\right]\begin{bmatrix}f(0)\\f(1)\\\vdots\\f(7)\end{bmatrix} \tag{2-8}$$

当 u=1 时

$$A_1(0)=\frac{1}{2}\cos\frac{\pi}{16},A_1(1)=\frac{1}{2}\cos\frac{3\pi}{16},\cdots,A_1(7)=\frac{1}{2}\cos\frac{15\pi}{16}$$

如果运算精度取为 13 比特，则用二进制表示为：

$$\left[A_1(0)A_1(1)\cdots A_1(7)\right]=\begin{bmatrix}0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\\0&1&0&1&0&1&0&1\\1&0&0&1&0&1&1&0\\1&0&1&0&1&0&1&0\\0&1&1&0&1&0&0&1\\1&0&1&0&1&0&1&0\\1&1&0&1&0&1&0&0\\1&0&0&1&0&1&1&0\\1&1&0&0&1&1&0&0\\1&1&1&0&1&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&1&1&1\end{bmatrix} \tag{2-9}$$

$$F(1)=\begin{bmatrix}0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\\0&1&0&1&0&1&0&1\\1&0&0&1&0&1&1&0\\1&0&1&0&1&0&1&0\\0&1&1&0&1&0&0&1\\1&0&1&0&1&0&1&0\\1&1&0&1&0&1&0&0\\1&0&0&1&0&1&1&0\\1&1&0&0&1&1&0&0\\1&1&1&0&1&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&1&1&1\end{bmatrix}\begin{bmatrix}f(0)\\f(1)\\f(2)\\f(3)\\f(4)\\f(5)\\f(6)\\f(7)\end{bmatrix}=\begin{bmatrix}F_1(12)\\F_1(11)\\F_1(10)\\F_1(9)\\F_1(8)\\F_1(7)\\F_1(6)\\F_1(5)\\F_1(4)\\F_1(3)\\F_1(2)\\F_1(1)\\F_1(0)\end{bmatrix} \tag{2-10}$$

从上述分析很容易发现一维 DCT 可以完全用加法器来实现。

3. 一维 DCT 的算式分析

首先，让我们看看一维 DCT 各项算术式子：

第一个一维 DCT 变换系数的各项：

F1(0)=0; F1(1)=0;F1(2)=(A0-A7)+(A1-A6)+(A2-A5);
F1(3)=(A0-A7)+(A1-A6);F1(4)=(A0-A7)+ (A3-A4);
F1(5)=(A0-A7)+(A1-A6)+(A3-A4);F1(6)= (A0-A7)+ (A2-A5);
F1(7)=(A1-A6)+(A2-A5);F1(8)=(A0-A7)+(A2-A5);
F1(9)=(A0-A7)+(A3-A4);F1(10)=(A1-A6)+(A3-A4);
F1(11)=(A1-A6)+(A3-A4);F1(12)= (A0-A7)+(A1-A6)+(A3-A4);

第二个一维 DCT 变换系数的各项：

F2(0)=0; F2(1)=0;F2(2)=(A0+A7)-(A3+A4);
F2(3)=(A0+A7)-(A3+A4)+(A1+A6)-(A2+A5);
F2(4)=(A0+A7)-(A3+A4)+(A1+A6)-(A2+A5);
F2(5)=0;F2(6)=(A0+A7)-(A3+A4);F2(7)=(A0+A7)-(A3+A4);
F2(8)=0;F2(9)=(A1+A6)-(A2+A5);
F2(10)=(A0+A7)-(A3+A4)+(A1+A6)-(A2+A5);
F2(11)=(A1+A6)-(A2+A5);F2(12)=(A1+A6)-(A2+A5);

第四个一维 DCT 变换系数的各项：

F4(0)=0; F4(1)=0;F4(2)={(A0+A7)-(A1+A6)}-{(A2+A5)- (A3+A4)};
F4(3)=0;F1(4)={(A0+A7)-(A1+A6)}-{(A2+A5)- (A3+A4)};
F4(5)={(A0+A7)-(A1+A6)}-{(A2+A5)- (A3+A4)};
F4(6)= 0;F4(7)={(A0+A7)-(A1+A6)}-{(A2+A5)- (A3+A4)};
F4(8)=0;F4(9)={(A0+A7)-(A1+A6)}-{(A2+A5)- (A3+A4)};
F4(10)=0;F4(11)=0;F4(12)= 0;

第三个一维 DCT 变换系数的各项：
F3(0)=(A1-A6)+(A2-A5)+(A3-A4); F3(1)=(A1-A6)+(A2-A5)+(A3-A4);
F3(2)=(A0-A7)+(A1-A6); F3(3)=(A0-A7)+(A1-A6) +(A3-A4);
F3(4)=(A3-A4);F3(5)=(A0-A7) +(A3-A4);F3(6)=(A1-A6);
F3(7)=(A0-A7)+(A1-A6)+(A2-A5);F3(8)=(A1-A6);
F3(9)=(A2-A5)+(A3-A4);F3(10)=(A0-A7)+(A3-A4);
F3(11)=(A0-A7)+(A3-A4);F3(12)=(A1-A6)+(A3-A4);

第五个一维 DCT 变换系数的各项：
F5(0)=(A1-A6); F5(1)=(A1-A6);F5(2)=(A0-A7)+(A3-A4);
F5(3)=(A3-A4);F5(4)=(A2-A5);F5(5)=(A2-A5)+(A3-A4);
F5(6)=(A0-A7);F5(7)=(A0-A7)+(A1-A6)+(A3-A4);
F5(8)=(A0-A7);F5(9)=(A1-A6)+(A2-A5);
F5(10)=(A2-A5)+(A3-A4);F5(11)=(A2-A5)+(A3-A4);F5(12)=(A0-A7)+(A2-A5);

第六个一维 DCT 变换系数的各项：
F6(0)=(A1+A6)-(A2+A5); F6(1)=(A1+A6)-(A2+A5);
F6(2)=0; F6(3)=(A0+A7)-(A3+A4);
F6(4)=(A0+A7)-(A3+A4);F6(5)=(A1+A6)-(A2+A5);
F6(6)=0;F6(7)=0;F6(8)=(A1+A6)-(A2+A5);
F6(9)=(A0+A7)-(A3+A4)+(A1+A6)-(A2+A5);
F6(10)=(A0+A7)-(A3+A4)+(A1+A6)-(A2+A5);
F6(11)=(A0+A7)-(A3+A4);F6(12)=(A0+A7)-(A3+A4);

第七个一维 DCT 变换系数的各项：
F7(0)=(A1-A6)+(A3-A4); F7(1)=(A1-A6)+(A3-A4);
F7(2)=(A2-A5); F7(3)=(A1-A6)+(A2-A5);
F7(4)=(A0-A7)+(A1-A6);F7(5)=(A0-A7)+(A1-A6) +(A2-A5);
F7(6)=0;F7(7)=(A2-A5)+(A3-A4);F7(8)=0;
F7(9)=(A0-A7)+(A1-A6)+(A3-A4);F7(10)=(A0-A7)+(A1-A6)+(A2-A5);
F7(11)=(A0-A7)+(A1-A6)+(A2-A5);F7(12)=(A0-A7)+(A1-A6);

现在，我们来分析一下，这些项中的共用项。首先，我们按如下式子所示，把上面的式子所用到的项提取出来：

$$D07 = A0 + A7; D_07 = A0 - A7; D16 = A1 + A6; D_16 = A1 - A6;$$
$$D25 = A2 + A5; D_25 = A2 - A5; D34 = A3 + A4; D_34 = A3 - A4;$$
$$D0716 = D07 + D16; D2534 = D25 + D34;$$
$$D07_16 = D07 - D16; D25_34 = D25 - D34;$$
$$D07_34 = D07 - D34; D16_25 = D16 - D25;$$
$$D_2534 = D_25 + D_34; D_1625 = D_16 + D_25;$$
$$D_1634 = D_16 + D_34; D_0716 = D_07 + D_16;$$
$$D_0725 = D_07 + D_25; D_0734 = D_07 + D_34;$$

现在，我们来分析一下各二级加法器的结果都被哪些 DCT 变换系数所用：

第二级加法器生成项	用到对应项的 DCT 变换系数
$D0716$	F0
$D2534$	F0
$D07_16$	F4，F6
$D25_34$	F4
$D07_34$	F2，F3
$D16_25$	F2，F6
D_2534	F1，F3，F5，F7
D_1625	F1，F3，F5，F7
D_1634	F1，F7
D_0716	F1，F3，F5，F7
D_0725	F1，F5
D_0734	F1，F3，F5，F6

表 1 第二级加法器的研究

接下来，我们再分析一下第三级加法器都做了什么运算及各对应的 DCT 变换系数。

第三级加法器生成项	用到对应项的 DCT 变换系数
$D0716+D2534$	F0
$D0716-D2534$	F2，F4，F6
D_16+D_2534	F3
D_0716+D_34	F1，F3，F5，F7
D_0716+D_25	F1，F3，F7

表 2 对第三级加法器的研究

如此，我们就很容易发现，其实，我们并不需要每计算一个一维 DCT 变换系数都要用组合电路把所有的各项都重复的计算出来，我们只要在计算的过程中加入一些寄存器，把我们已经计算出来的项存起来，等到用到的时候从里面取出来就直接可以用了，这样，只要我们通过合理的布局布线，我们的一维 DCT 变换系数就可以很快的计算出来，这样，功耗肯定也是会降低的。下面，我们设计了两种基于时序电路的一维 DCT。事实证明，我们这样做，还可以大大减少加法器的数量。

4. DCT_13 的设计

仅用 13 个加法器的一维 DCT（为方便起见，我们暂且称之为 DCT_13）设计：
在这个设计中，我们只用了 13 个加法器，就完成了一维 DCT。我们的基本思路是这样的：首先，A0，A1，A2，A3，A4，A5，A6，A7 依次顺序输入，经过一个 8 数据的串并转换模块，这 8 个数据同时输入到第一级加法器，我们的第一级加法器共有 4 个加减可控制的简单 ALU，首先，我们当然是让这 4 个 ALU 做加法运算，因为，我们发现 F0，F2 并没有用到这一级加法器所计算出来的 D_07, D_16, D_25, D_34,而它们的输出当然也是需要一个时钟周期的,于是,我们完全可以先让这 4 个 ALU 做加法运算，再在下一个周期让它们做减法运算，当然，每次计算完的结果要保存在寄存器，于是，第一级的加法运算是不会引起任何延时的。

让我们再来看第二级的加法器设计。通过对表 2 的研究，你发现什么了？是的，如同第一级加法器的设计一样，我们同样可以改变输出的顺序，来计算第二级加法器所需的各项。结合第一级加法器的设计，我们让 DCT 变换系数的输出变为这个顺序吧：F0，F2，F6，F4，F5，F7，F1，F3。于是，我们可以先用两个加减可控制的简单的 ALU 计算出 F0 所需要的 $D0716$，$D2534$，同时，我们在设计中也让 $D07_34$，D_2534 也各用了一个加法器计算出来了，$D07_16$，$D25_34$ 当然也在接在来的一个时钟周期被前面所提到的 ALU 计算出来，接下来，我们再设计两个加法可控制的简单 ALU 来分别计算 D_1625，D_1634 和 D_0716，D_0725，D_0734，发现我们为什么这么做了吗？这是因为如此我们就

可以让 ALU 的一个输入固定下来，而不用再设计缓存器来缓存输入。当然，你也可以试试不把一个输入端固定下来会是什么样子。

第三级加法器的设计。首先，我们要输出的是 F0，所以，我们用一个加减可控制的简单 ALU 计算出了 $D0716+D2534$，再在下一个时钟周期计算出 $D0716-D2534$，同时，另一个加法器将 D_0716+D_34，D_0716+D_25，D_16+D_2534 依次计算出来，我们的输出顺序为：F0，F2，F6，F4，F5，F7，F1，F3。我们这种输出顺序是方便读者理解整个流程，其实，我们很容易发现第三个数据输出后，所有的项其实都已经被计算出来了，所以，我们后面的顺序是什么样的都是不重要的 (如果我们能曾加两个时钟的延时，也可以按顺序输出数据，即：F0，F1，F2，F3，F4，F5，F6，F7) 。

图 1 是我们的 DCT_13 的结构图。其中，series_to_parallel 完成八数据串并转换，Addmat1 完成第一级加法，Addmat2 完成一维 DCT 变换的其余工作。

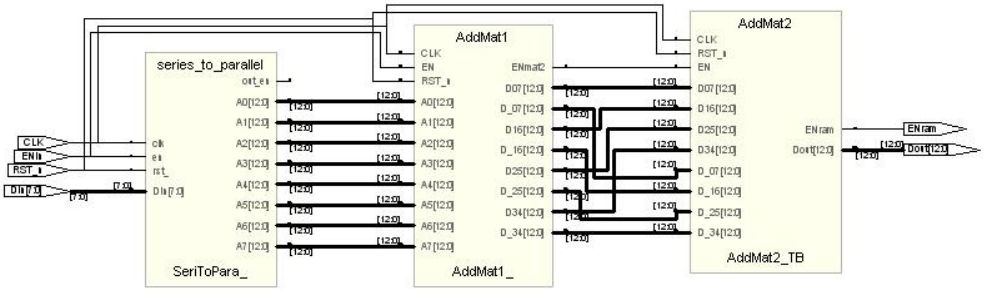

图 1　　DCT_13 的结构图

图 2 是我们的一维 DCT（13 个加法器）的仿真时序图。可以发现，我们的 8 个数据输入完成同时就可以有一维 DCT 变换系数持续输出。可以看到我们的第一组输入数据 A0—A7 依次为：170，153，153，153，170，153，153，153，输出的一维 DCT 变换系数 F0—F7 为：440，6，0，11，11，-4，0，9 与理论值：444，6，0，11，12，-2，0，9 基本相符，出现的差异是由于加法器的位数的有限性造成的，但它是可以满足实际要求的。

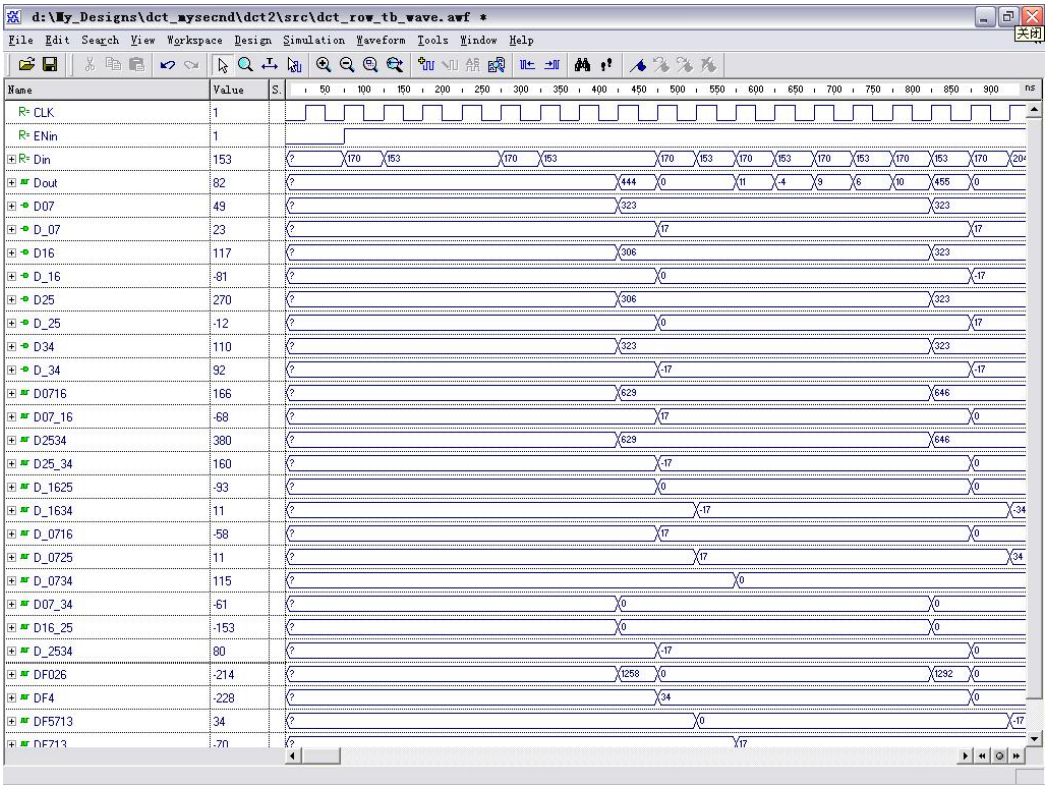

图 2　　DCT_13 的仿真时序图

图 3 是我们用 Synplify 综合工具综合后得到的一些数据，所选用的器件为 APEX20KE160etc144-2x，可以看到在这个 DCT_13 中，我们仅用了 1257 个 LUT。

```
I/O ATOMs:      24

Total LUTs:  1257 of 6400 (19%)
Logic resources:  1257 ATOMs of 6400 (19%)
ATOM count by mode:
  normal:       1248
  arithmetic:   0
  counter:      6
  qfbk_counter: 3


ESBs:          0  (0% of 40)
LPM latches:    346

ATOMs using regout pin: 75
  also using enable pin: 10
  also using combout pin: 1
ATOMs using combout pin: 1179
ATOMs using cascin pin: 16

Number of Inputs on ATOMs: 4612
Number of Nets:   3551

##### END OF AREA REPORT #####
```

图 3 DCT_13 的一些综合相关数据

5. DCT_11 的设计

仅用 11 个加法器的一维 DCT（为方便起见，我们暂且称之为 DCT_11）设计。在这个设计中，我们改变了前面提到的 13 个加法器的一维 DCT 设计的串并转换和第一级加法器。而第二级和第三级加法器与前面提到的是一样的。

在这里，我们让输入的顺序改变一下，为 A0，A7，A1，A6，A2，A5，A3，A4。发现什么没有？是的，如此，我们可以让串并转换变为二数据的串并转换，这样对资源的节约是巨大的，让两个并行出来的数据输入到第一级加法器。在这里，我们的第一级加法器就可以只用两个加减可控制的简单 ALU。其中一个计算出 D07，D_07， D25， D_25， 另一个则计算出 D16，D_16， D34 ，D_34,并且，我们当它们都被计算出来的时候，我们让它们同时保存到另一组寄存器，再做接下来的运算。

图 4 是我们的 DCT_11 的结构图。其中，SeriToPara 完成二数据串并转换，Addmat1 完成第一级加法，Addmat2 完成一维 DCT 变换的其余工作。

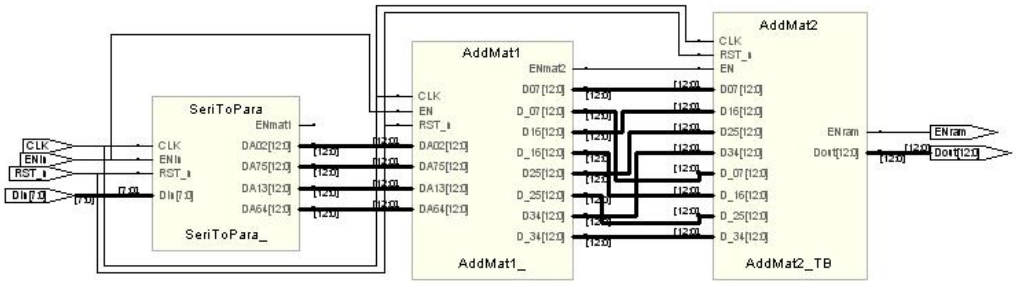

图 4　　DCT_11 的结构图

图 5 是我们用 Synplify 综合工具综合后得到的一些数据，所选用的器件为 APEX20KE160etc144-2x，可以看到在这个 DCT_13 中，我们仅用了 1151 个 LUT。

```
I/O ATOMs:      24

Total LUTs:  1151 of 6400 (17%)
Logic resources:  1151 ATOMs of 6400 (17%)
ATOM count by mode:
  normal:      1142
  arithmetic:  0
  counter:     6
  qfbk_counter: 3


ESBs:          0  (0% of 40)
LPM latches:    380

ATOMs using regout pin: 27
  also using enable pin: 9
  also using combout pin: 0
ATOMs using combout pin: 1121
ATOMs using cascin pin: 20

Number of Inputs on ATOMs: 4118
Number of Nets:   3192

##### END OF AREA REPORT #####|
```

图 5 DCT_11 的一些综合相关数据

6. 方案设计比较

下面，我们将我们的设计与目前已有的一维 DCT 设计进行了比较。通过比较可以看到我们的设计大大减少了一维 DCT 设计中所需要的加法器数。

一维 DCT 实现结构	所用加法器数	资源节约率(%)
直接实现	308	-
Initial Compression	97	68．5
NEDA 结构	35	88．64
DCT_13	13	95．78
DCT_11	11	97．38

表 3　　一维 DCT 设计比较

7. 结语

我们举了两个例子，来说明基于时序电路的一维 DCT。也许我们的设计还不是最简，我们也只希望能起到一个抛砖引玉的作用，让研究人员能够设计出更节约硬件资源的一维 DCT。

参考文献

[1]N.Ahmed, T.Natrajan and K.R.Rao, “Discrete cosine transform”,IEEE trans. Computers, January 1974.

[2]欧阳合，韩军，“视频编解码器设计_开发图像与视频压缩系统”，国防科技大学出版社，2005.

[3] high performance distributed DCT architecture, " in Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, Pennsylvania, April 2002.

[4] A.Peled, B.Liu, "A new hardware realization of digital filters," IEEE Trans. ASSP, vol.ASSP-22,no.6, pp.456-462, Dec.1974.

[5] D.F.Elliott (ed.), Handbook of Digital Signal Processing, Academic Press, pp.964-972, 1987.

[6] S.A.White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Magazine, pp.4-19, July 1989.

[7] G-K.Ma and F.J.Taylor, "Multiplier policies for digital signal processing," IEEE ASSP Magazine, pp.6-19, Jan.1990.

[8] Iain E. G. Richardson, "Video Codec Design _ Developing Image and Video Compression Systems " The Robert Gordon University Aberdeen. UK