高阶数字滤波器分布式算法结构比较

王法栋[1]，刘 宇[2]
(1. 中国人民解放军某部，辽宁葫芦岛 125000；2. 中国科学院声学研究所东海研究站，上海 200032)

摘要：基于 FPGA，对高阶 FIR 滤波器两种算法——“基于查找表(LUT)的分布式算法”和“改进的分布式算法”的功能及结构差异进行类比。“改进的分布式算法”是对“基于 LUT 分布式算法”在减少存储器技术上的改进。对于一个高阶滤波器来说，“改进的分布式算法”比“基于 LUT 的分布式算法”，需要更少的存储器和系统资源。减少存储器使用的递归反复技术极大地增加了在 FPGA 平台上可实现的滤波器阶数的最大值。“改进的分布式算法”不仅节省了使用的电子器件数量，而且还平衡了 FPGA 硬件资源中逻辑单元(LE)和存储器的使用。从 FPGA 的执行结果可以确定，“改进的分布式算法”可以实现一个 1024 抽头的 FIR 滤波器，且比“基于 LUT 的分布式算法”节省更多系统资源。

关键词：FIR 滤波器；分布式算法；FPGA
中图分类号：TN912 **文献标识码**：A **文章编号**：1000-3630(2009)-03-0307-05
DOI 编码：10.3969/j.issn1000-3630.2009.03.024

Comparison between distributed arithmetic architectures of high-order digital filters

WANG Fa-dong[1], LIU Yu[2]
(1. *Chinese People's Liberation Army Huludao*, 125000, *Liaoning*; 2. *Acoustics laboratory chinese academy of sciences*, *Shanghai* 200032, *China*)

Abstract: Based on FPGA a comparison between two distributed arithmetic algorithms (DA) of high-order FIR filters in function and architecture (LUT-based DA and Proposed DA) has been made. Proposed DA needs less memory and system resource than LUT-based DA due to a memory reduction technique. Recursive iteration of the memory reduction technique significantly increases the maximal number of filter order implementable on an FPGA platform by not only saving electronic devices, but also balancing hardware usage between logic element (LE) and memory. FPGA implementation results confirm that the proposed DA can implement a 1024-tap FIR filter with significantly smaller area usage than LUT-based DA.
Key words: FIR filter; distributed arithmetic; FPGA

1 简 介

声靶自导信号处理模块，主要进行自导信号的处理，它主要由一个 FPGA 自导信号部件，USB 数据接口和 PC104 CPU 等模块组成。由于 FPGA 在计算速度及结构上的一些优势，系统中的大部分计算是在 FPGA 自导部件中实现的，它是自导信号主要的硬件处理部件。针对 FPGA 的自身特点、运算资源等方面，对高阶 FIR 滤波器的两种算法进行对比分析，选择较优的算法来实现。

一个离散时间的线性有限脉冲响应（FIR）滤波器，其输出序列 $y[n]$，是系统延迟与输入信号 x 的采样序列 $x[n]$ 的加和，即：

$$y[n]=\sum_{i=0}^{K-1}\omega_i x[n-i] \tag{1}$$

一个超大规模集成电路的实现，需要 K 次的乘加运算，如果通过硬件来实现是很奢侈的，因为乘加运算的逻辑非常复杂而且需要很多的硬件资源。考虑到这些问题，就出现了“基于 LUT 的分布式算法”，由一个查找表来代替乘加运算。这样就使得复杂的乘加运算变得更加直观，更重要的是使运算大大的简化，节省了大量的硬件资源。分布式算法是一个位序列的运算，以固定的计算步骤，忽略要计算项的数量，来执行一系列定点的乘加运算。但这种基于 LUT 的分布式算法存在一个问题，就是其 LUT 的大小将随着滤波器阶数的增加，以 2^K-1 的指数速率增加。当查找表的体积增大到一定程度，就会给存储带来很大的麻烦。

“改进的分布式算法”针对这一问题进行了很

收稿日期：2008-07-04；**修回日期**：2009-02-03
作者简介：王法栋(1971-)，男，山东临沂人，硕士，研究方向为信号处理。
通讯作者：刘宇，E-mail: ly-bi@163.com

好的改进，从而实现了一个能有效提高硬件效率的分布式算法结构。为了解决 LUT 的大小随着滤波器阶数的增加而快速增加，且占用系统存储资源过大的问题，将“基于 LUT 的分布式算法”的基本单元 LUT 的这部分结构进行改进，把一个 2^K-1 行的查表过程，转换为对 K 个 2 输入开关进行求和的过程，从而大大节省了系统的存储资源，极大地提高了系统的最大频率，使得在一个给定的 FPGA 平台上实现更高阶的 FIR 滤波器成为可能。

2 基于 LUT 的分布式算法及其结构

分布式算法(Distributed Arithmetic，DA)是一项重要的 FPGA 技术，广泛地应用于计算乘积和之中。

$$y=\sum_{n=0}^{N-1}\omega_n x[n] \tag{2}$$

其中：

$x[n]$——输入信号 x 的采样序列，

$y[n]$——输出序列，

ω_n——设计滤波器时可以计算出的常系数。

假设系统为无符号系统，则：

$$x[n]=\sum_{b=0}^{B-1}x_b[n]\times 2^b,\ x_b[n]\in[0,1] \tag{3}$$

其中，$x_b[n]$ 表示 $x[n]$ 的第 b 位，则：

$$y=\sum_{n=0}^{N-1}\omega_n\times\sum_{b=0}^{B-1}x_b[n]\times 2^b \tag{4}$$

重新分别求和，展开结果如下：

$$\begin{aligned}y=&\omega_0\left(x_{B-1}[0]2^{B-1}+x_{B-2}[0]2^{B-2}+\ldots+x_0[0]2^0\right)+\\&\omega_1\left(x_{B-1}[1]2^{B-1}+x_{B-2}[1]2^{B-2}+\ldots+x_0[1]2^0\right)+\cdots+\\&\omega_{N-1}\left(x_{B-1}[N-1]2^{B-1}+\ldots+x_0[N-1]2^0\right)=\\&(\omega_0x_{B-1}[0]+\omega_1x_{B-1}[1]+\ldots+\omega_{N-1}x_{B-1}[N-1])2^{B-1}+\\&(\omega_0x_{B-2}[0]+\omega_1x_{B-2}[1]+\ldots+\\&\omega_{N-1}x_{B-2}[N-1])2^{B-2}+\cdots+\\&(\omega_0x_0[0]+\omega_1x_0[1]+\ldots+\omega_{N-1}x_0[N-1])2^0\end{aligned} \tag{5}$$

可简写为如下形式：

$$y=\sum_{b=0}^{B-1}2^b\times\sum_{n=0}^{N-1}\omega_n x_b[n] \tag{6}$$

图 1 为基于 LUT 的分布式算法结构图。从图 1 可看出，正是通过 LUT 这部分算法实现了 $\sum_{n=0}^{N-1}\omega_n x_b[n]$ 的乘加运算，而 $\sum_{b=0}^{B-1}2^b$ 则通过后面的移位累加来实现，这样就大大地简化了复杂的乘加运算过程，节省了系统的硬件资源，使通过硬件能够实现复杂乘加运算。

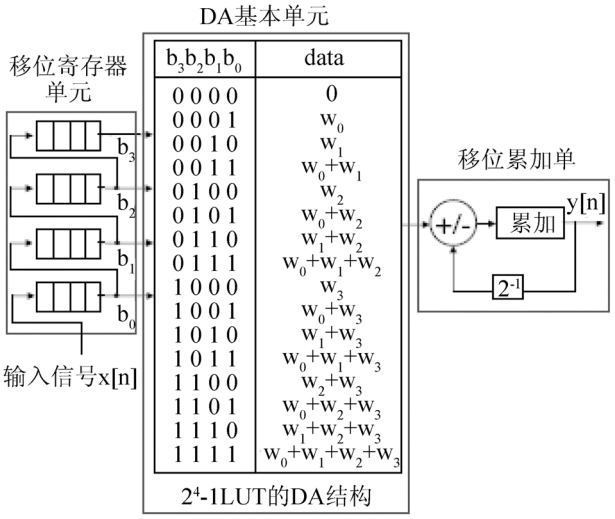


图 1 基于 LUT 的 DA 结构图(4 抽头 FIR 滤波器)

Fig.1 LUT-based DA architecture (4-tap)

对于有符号的系统，则只需要增加一位符号位，用补码来表示，即最高位为符号位，则 $x[n]$ 表示为：

$$x[n]=-2^b\times x_B[n]+\sum_{b=0}^{B-1}x_b[n]\times 2^b \tag{7}$$

代入式(2)，有：

$$y=-2^B\times\sum_{n=0}^{N-1}\omega_n x_B[n]+\sum_{n=0}^{N-1}\omega_n\times\sum_{b=0}^{B-1}x_b[n] \tag{8}$$

可以看出，当系统为有符号数时，只需要将最后一位的累加，变为减法即可。

基于 LUT 的分布式算法的 VHDL 实现如下：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity lut is
port
(
clk : in std_logic;
      lut_in : in std_logic_vector(3 downto 0);
      lut_out : out std_logic_vector(7 downto 0)
);
end lut;
architecture behavior of lut is
constant w0 :
   std_logic_vector(7 downto 0) :=X"40";
constant w1 :
   std_logic_vector(7 downto 0) :=X"08";
constant w2 :
   std_logic_vector(7 downto 0) :=X"20";
constant w3 :
   std_logic_vector(7 downto 0) :=X"10";
signal lut_out1 :
   std_logic_vector(7 downto 0);
begin
process(clk)
```

```
begin
  if clk' event and clk = '0' then
    case lut_in is    --查找表
      when "0000" =>
           lut_out1 <= "00000000";
      when "0001" =>
           lut_out1 <= w0;
      when "0010" =>
           lut_out1 <= w1;
      when "0011" =>
           lut_out1 <= w0+w1;
      when "0100" =>
           lut_out1 <= w2;
      when "0101" =>
           lut_out1 <= w0+w2;
      when "0110" =>
           lut_out1 <= w1+w2;
      when "0111" =>
           lut_out1 <= w0+w1+w2;
      when "1000" =>
           lut_out1 <= w3;
      when "1001" =>
           lut_out1 <= w0+w3;
      when "1010" =>
           lut_out1 <= w1+w3;
      when "1011" =>
           lut_out1 <= w0+w1+w3;
      when "1100" =>
           lut_out1 <= w2+w3;
      when "1101" =>
           lut_out1 <= w0+w2+w3;
      when "1110" =>
           lut_out1 <= w1+w2+w3;
      when "1111" =>
           lut_out1 <= w0+w1+w2+w3;
      when others =>
           lut_out1 <= "00000000";
    end case;
  end if;
  lut_out <= lut_out1;
end process;
end behavior;
```

3　改进的分布式算法及其结构

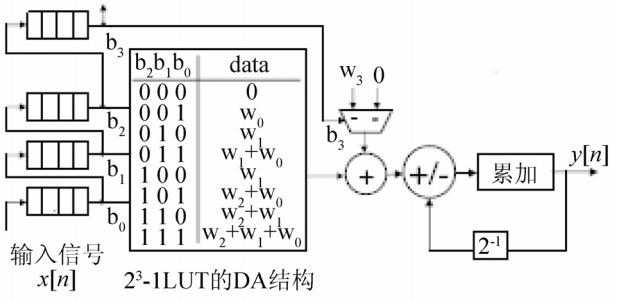
图 2 为改进过的分布式算法结构图。

“改进的分布式算法”与“基于 LUT 的分布式算法”有如下的对应关系：

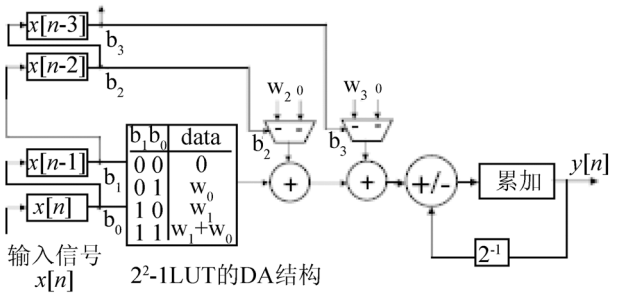
$$LUT(1,b_{k-2},\cdots,b_1,b_0)= LUT(0,b_{k-2},\cdots,b_1,b_0)+\omega_{k-1} \quad (9)$$

其中 k 是与 LUT 相连接的地址线的数量。

通过图 1 与图 2(a)对比可以看出，将移位寄存器输出的最高位从 LUT 中分离出来，由一个 2 输入开关和一个全加器来替代这部分 LUT，从而达到减小 LUT 大小的目的，如图 2(a)中所示。同理，LUT 的大小可以被进一步的缩小，如图 2(b)。通过与前面相同的方法，因为图 2(a)中的 LUT 仍然满足(9)所给出的对应关系。在对 LUT 进行反复的简化之后，最终“基于 LUT 的分布式算法”结构的 4 抽头 FIR 滤波器，被转化为如图 3 所示的形式——即“改进的分布式算法结构”。



(a) 2^3-1 的 LUT 实现 DA 结构



(b) 2^2-1 的 LUT 实现 DA 结构

图 2　改进的 DA 结构图(4 抽头的 FIR 滤波器)

Fig.2　Proposed DA architecture of a 4-tap FIR filter

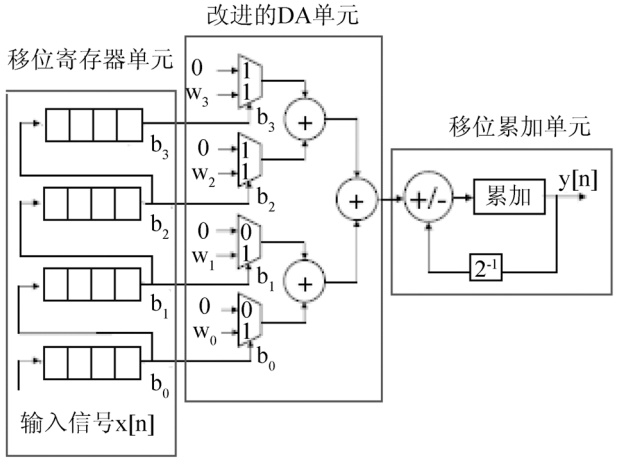


图 3　改进的 DA 结构图(4 抽头的 FIR 滤波器)

Fig.3　Proposed DA architecture (4-tap FIR filter)

改进的分布式算法的 VHDL 实现如下：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity add is
port
(
```

```
        clk : in std_logic;
        add_in : in std_logic_vector(3 downto 0);
        add_out : out std_logic_vector(7 downto 0)
);
end add;
architecture behavior of add is
constant w0 :
   std_logic_vector(7 downto 0) :=X"40";
constant w1 :
   std_logic_vector(7 downto 0) :=X"08";
constant w2 :
   std_logic_vector(7 downto 0) :=X"20";
constant w3 :
   std_logic_vector(7 downto 0) :=X"10";
signal ad_out1 :
   std_logic_vector(7 downto 0);
signal ad_out2 :
   std_logic_vector(7 downto 0);
signal ad_out3 :
   std_logic_vector(7 downto 0);
signal ad_out0 :
   std_logic_vector(7 downto 0);
signal add0_1 :
   std_logic_vector(7 downto 0);
signal add2_3 :
   std_logic_vector(7 downto 0);
begin
process(clk)
   begin
      if(clk' event and clk = '0')then
         if(add_in(3) = '1')then    --2 输入开关
            ad_out3 <= w3;
         else
            ad_out3 <= "00000000";
         end if;
         if(add_in(2) = '1')then
            ad_out2 <= w2;
         else
            ad_out2 <= "00000000";
         end if;
         if(add_in(1) = '1')then
            ad_out1 <= w1;
         else
            ad_out1 <= "00000000";
         end if;
         if(add_in(0) = '1')then
            ad_out0 <= w0;
         else
            ad_out0 <= "00000000";
      end if;
   end if;
end process;
  add0_1 <= ad_out0 + ad_out1;    --加法器
  add2_3 <= ad_out2 + ad_out3;
  add_out <= add0_1 + add2_3;
end behavior;
```

4 性能对比

从使用电子器件的数量和FPGA的综合结果这两方面，对“改进的分布式算法”的硬件复杂程度与“基于 LUT 的分布式算法”做比较。

比较电子器件的数量是用来评价一个超大规模集成电路芯片中硅面积的常用技术。然而，数字逻辑功能根据设计优化目的的不同，可以通过很多方法来实现，如硅的面积、功率消耗和最大频率等。例如，异或逻辑可以通过 4 个与非门来实现，相当于使用了 16 个晶体管；也可以通过一个全静态 CMOS 逻辑来实现，相当于使用了 8 个晶体管。表 1 给出了两种 DA 结构中基本单元晶体管数量使用的对比。其中 B 和 k 分别代表原始 LUT 的字长和基本单元的大小。

表 1 两种结构电子器件数量评估的对比

Table 1 Electronic device amount estimates for the two architectures

逻辑功能	基于 LUT 的 DA 结构	改进的 DA 结构
ROM 解码器	$C(1，k)$	0
ROM 数据	D（1，k，B）	0
2×1MUX	0	$6k\times B$
加法器	0	（k-1）×30B

两个函数定义如下：

$$C(a,b)=4\left(2\sum_{i=a}^{b-1}(b-i)+2^{b-a+1}\right) \tag{10}$$

$$D(a,b,c)=2^{b-a+1}\times c \tag{11}$$

移位寄存器和移位累加单元在表中没有涉及，因为在两种 DA 结构中，这两部分是相同的，故不做比较。

图 4 中给出了滤波器的阶数从 4 增加到 16 时，两种结构使用晶体管数量的变化曲线。

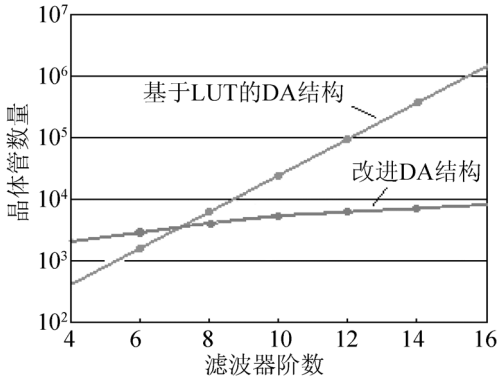


图 4 两种结构随滤波器阶数的增加而使用的电子器件数量对比

Fig.4 Comparison of electronic device amount with various of filter order

从图 4 可以明显地看出，当滤波器阶数较小(小于 8 阶)的时候，“基于 LUT 的分布式算法”略优于

“改进的分布式算法”，而当滤波器的阶数继续增大时，“改进的分布式算法”所使用晶体管数量的变化比较平缓，增加得很慢；而“基于 LUT 的分布式算法”几乎呈指数增加。这样在使用高阶滤波器时，选用“改进的分布式算法”将会带来较明显的优势。

下面用一个具体实例，进一步说明“改进的分布式算法”的优越性。这里选用了 Altera 公司的 Stratic 系列 FPGA 中的 EP1S80F1508C6 芯片来进行下面的测试。在 LUT 的字长 B=18，基本单元的大小 k=4 的条件下，测试滤波器的阶数从 4 到 1024 时，两种算法结构各自消耗系统资源的状况及测试结果如表 2 所示。

表 2　不同滤波器阶数时, FPGA 实现结果的对比(k=4, B=18)

Table 2　Comparison of FPGA implementation results for various filter size K(k=4, B=18)

滤波器阶数 K	基于 LUT 的 DA 结构		改进的 DA 结构	
	逻辑单元 LE	存储器 Memory	逻辑单元 LE	存储器 Memory
4	272	344	210	56
16	551	1376	367	224
64	1639	5504	887	896
128	3056	11008	1569	1792
256	5890	22016	2946	3584
512	11547	44032	5659	7168
1024	22862(100%)	88064(100%)	11086(48%）	14336(16%)

从表 2 可以看出，“改进的分布式算法”比“基于 LUT 的分布式算法”需要更少的逻辑单元(LE)和存储器。在“基于 LUT 的分布式算法”中，由于 LUT 的存在，对存储器的要求，无论滤波器的阶数高低，都要比“改进的分布式算法”高很多。同时，对 LEs 的使用数量也相当大。这对一个系统来说是一个很大的负担，而“改进后的分布式算法”在这两方面都有很大的改善与提高，不但节省了大量的系统资源，还提高了系统的整体效率。

5　结 论

本文针对声靶自导信号处理的特点，选择通过 FPGA 实现 FIR 滤波器的前提下，对在高阶滤波器下“改进的分布式算法”和“基于 LUT 的分布式算法”做了类比。“改进的分布式算法”是通过将“基于 LUT 的分布式算法”做对应关系的变化而得到的。在保持结果不变的条件下，增加控制电路，通过多个 2 输入开关和加法器来有效的减少存储器的使用。

从文中可以看出，“改进的分布式算法”在超大规模集成电路和 FPGA 两方面的实践中，都能有效地提高硬件的效率。在 FPGA 的实验结果中可以看出，对于一个 1024 抽头的 FIR 滤波器，在 k=4，B=18 的条件下，“改进的分布式算法”比“基于 LUT 的分布式算法”，节省了 52%的 LEs 和 84%的存储器。从而在声靶自导信号处理中，选择在 FPGA 上通过“改进的分布式算法”实现 FIR 滤波器。

参　考　文　献

[1] Uwe Meyer-Baese 著, 刘凌译. 数字信号处理的 FPGA 实现[M]. 清华大学出版社. 2006, 6.

[2] 丁玉美, 高西全编著.数字信号处理[M]. 西安电子科技大学出版社. 2001, 1.

[3] James R.Armstrong, F. Gail Gray 著. VHDL 设计、表示和综合(英文版)[M]. 机械工业出版社. 2004, 10.

[4] 刘韬, 楼兴华. FPGA 数字电子系统设计与开发实例导航[M]. 人民邮电出版社. 2005, 6.

[5] EDA 先锋工作室, 吴继华, 王成编著. Altera FPGA/CPLD 设计(高级篇)[M]. 人民邮电出版社. 2005, 7.

[6] 曾繁泰, 陈美金著. VHDL 程序设计[M]. 清华大学出版社. 2000, 8.

[7] 阎石, 主编. 清华大学电子教研组编.数字电子技术基础[M]. 高等教育出版社. 1998, 11.

[8] Yu S, Swartzlander E E. DCT implementation with distributed arithmetic[J]. IEEE Transactions on Computers, 2001, **50**(9): 985-991.

[9] Chang T.-S., Chen C, Jen C.-W. New distributed arithmetic algorithm and its application to IDCT[J]. IEEE. Proceedings Circuits, Devices and Systems, 1999, **146**(4): 159-163.

[10] Chang T.-S, Jen C.-W. Hardware-efficient implementations for discrete function transforms using LUT-based FPGAs[J]. IEEE Proceedings Circuits, Devices and Systems, 1999, **146**(6): 309-315.

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

[12] Acock, S. J. B, Dimond, K. R. Automatic mapping of algorithms onto multiple FPGA SRAM modules, Field-programmable Logic and Applications[A]. 7th International Work shop, FPL'97 Proceedings[C]. 1997.