第35卷 第12期 计 算 机 工 程 2009年6月
Vol.35 No.12 Computer Engineering June 2009
·博士论文· 文章编号：1000—3428(2009)12—0001—04 文献标识码：A 中图分类号：TN911.72

数字信号处理器中高性能可重构加法器设计

马 鸿，李振伟，彭思龙
(中国科学院自动化研究所国家专用集成电路设计工程研究中心，北京 100080)

摘 要：设计一款适用于高性能数字信号处理器的16位加法器。该加法器结合条件进位选择和条件“和”选择加法器的特点，支持可重构，可以进行2个16位数据或者4个8位数据的加法运算，同时对其进位链进行优化。相对于传统的条件进位选择加法器，在典型工作条件下，采用0.18 μm工艺库标准单元，其延时降低46%，功耗降低5%。

关键词：条件进位选择加法器；条件“和”选择加法器；可重构加法器

High Performance Re-configurable Adder Design for Digital Signal Processor

MA Hong, LI Zhen-wei, PENG Si-long
(National ASIC Design Engineering Research Center, Institute of Automation, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 This paper presents the design of a high performance re-configurable 16-bit adder, which is well suitable for digital signal processor. The adder can add two 16-bit operands or four 8-bit operands. It is a hybrid of Conditional Carry Select adder(CCS) and Conditional Sum Select adder(CSS) with which the carry chain is also optimized. Simulation results show that the delay is reduced by 46% and the power is 5% lower compared with the general CCS under typical conditions with standard cell using 0.18 μm technology.

【Key words】 Conditional Carry Select adder(CCS); Conditional Sum Select adder(CSS); re-configurable adder

1 概述

加法器是数字信号处理器中非常重要的运算部件，它直接影响了数字信号处理器的速度和性能。特别是随着DSP向VLIW架构发展，其频率不断提高，支持可重构，即处理不同字长的数据，对加法器的性能提出了更高的要求。

加法器[1-4]有多种实现方式，如行波进位加法器、曼彻斯特进位加法器、超前进位加法器以及条件进位加法器等。在行波进位加法器[1,4]中，每一个全加器单元必须等待输入进位到达后才能产生一个输出进位，因此它的速度比较慢。曼彻斯特进位加法器[1]的进位链在产生组进位输出时可能出现多个MOS管串联放电的情况，这成为限制其速度提升的关键。超前进位加法器[1]每一位的进位输出以及“和”位输出都与前面的位无关，因此有效地消除了逐位进位效应。理论上这种加法器能以两级门电路实现，然而由于逻辑门的扇入扇出限制，这种方法是不可行的。因此一般用较简单的门，以多个逻辑层次来实现，但这样会使传播延时增大。条件进位选择加法器(CCS)[2-3]预先考虑进位输入的2种可能值(0或1)，并提前计算出针对这2种可能进位的输出。一旦输入进位的确切值已知，正确的进位输出信号就可以通过一个简单的多路开关选出。条件“和”选择加法器(CSS)则是利用条件选择原理，分别计算进位输入为0和1这2种情况下的“和”，并最终由多路选通器选择，从而进一步提高了加法器的速度。

2 传统CCS的算法和结构

条件进位选择加法器(CCS)中的关键电路是进位选择电路。该电路用于快速产生进位信号，从而快速得到加法运算结果。进位选择电路的算法如下所示：

$$C_i = G_i + C_{i-1} \bullet P_i \quad \text{for } i=0\sim n\text{-}1 \tag{1}$$

其中，G_i和P_i称作进位产生因子和进位传播因子，$G_i=X_i \bullet Y_i$，$P_i=X_i+Y_i$，X_i和Y_i是加法器第i位的2个数据输入；C_{i-1}和C_i分别是加法器第i位的进位输入和进位输出信号(C_{-1}是第0位的进位输入)。

由上述等式可以推出，如果$C_{i-1}=0$，则$C_i=G_i$，否则，如果$C_{i-1}=1$，则$C_i=P_i$。所以，式(1)可重新表达为

$$C_i = \overline{C_{i-1}} \bullet G_i + C_{i-1} \bullet P_i \tag{2}$$

基于式(2)，可以得到一个基于mux的进位产生电路，图1是一个1位的进位产生电路。

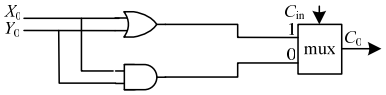

图1 1位的进位产生电路

常规的16位进位选择加法器如图2所示，加法器平均分为4个4位的进位块。每个进位块由一个进位信号C_j进行控制，根据式(3)，选出相应的进位输出信号C_i。定义C_i(if C_j=0)和C_i(if C_j=1)是2个变量，分别代表C_j=0和C_j=1是对应第i位产生的进位信号：

$$C_i = \overline{C_j} \bullet C_i(\text{if } C_i=0) + C_j \bullet C_i(\text{if } C_i=1) \tag{3}$$

其中，j= -1, 3, 7, 11; i=j+1 ~ j+4。

基金项目：国家科技支撑计划基金资助重点项目(2006BAK07B04)；中科院青年科技创新基金资助项目(DG07J01)

作者简介：马 鸿(1981－)，女，博士研究生，主研方向：VLSI设计自动化；李振伟，博士研究生；彭思龙，研究员、博士、博士生导师

收稿日期：2008-12-10 **E-mail：**hong.ma@ia.ac.cn

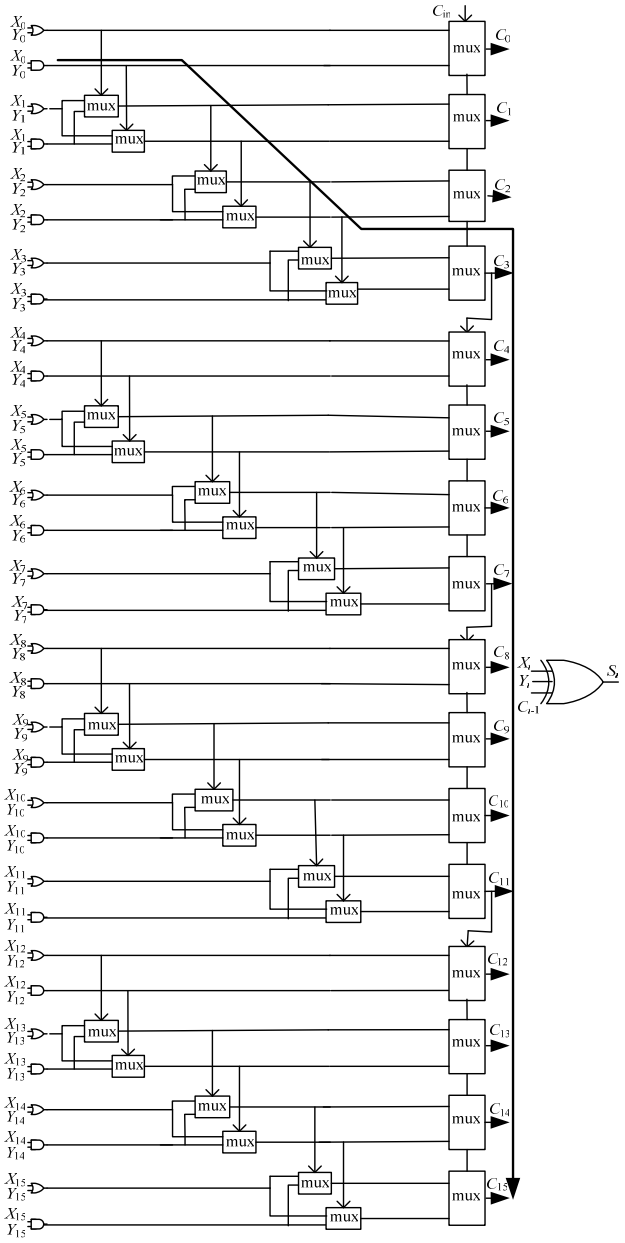

图 2　传统 16 位 CCS 的结构

下面以第一个进位块为例，描述 C_i(if C_j=0)和 C_i(if C_j=1)是如何产生的。第一个进位块的进位输出信号分别是 C_0, C_1, C_2 和 C_3，最低位进位信号是 C_{in}。

当 C_{in}=1 时，根据式(2)，可以得到 C_0= $P_0 \equiv C_0$(if C_{in}=0)。

当 C_{in}=0 时，根据式(2)，可以得到 C_0=$G_0 \equiv C_0$(if C_{in}=1)。

同样，对于 C_1，当 C_{in}=1 时，可以得到

$$C_1 = \overline{C_0} \bullet G_1 + C_0 \bullet P_1 = \overline{P_0} \bullet G_1 + P_0 \bullet P_1 \equiv C_1(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_1 = \overline{C_0} \bullet G_1 + C_0 \bullet P_1 = \overline{G_0} \bullet G_1 + G_0 \bullet P_1 \equiv C_1(\text{if } C_{\text{in}} = 0)$$

同样，对于 C_2，当 C_{in}=1 时，可以得到

$$C_2 = \overline{C_1} \bullet G_2 + C_1 \bullet P_2 = \overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1} \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2 \equiv C_2(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_2 = \overline{C_1} \bullet G_2 + C_1 \bullet P_2 = \overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1} \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2 \equiv C_2(\text{if } C_{\text{in}} = 0)$$

同样，对于 C_3 ，当 C_{in}=1 时，可以得到

$$C_3 = \overline{C_2} \bullet G_3 + C_2 \bullet P_3 = \overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1 \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2 \bullet G3 + (\overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1} \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2) \bullet P3} \equiv C_3(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_3 = \overline{C_2} \bullet G_3 + C_2 \bullet P_3 = \overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1 \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2 \bullet G3 + (\overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1} \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2) \bullet P3} \equiv C_3(\text{if } C_{\text{in}} = 0)$$

总之，在上述 16 位进位选择加法器中，每个进位块产生 4 个进位输出信号。在第 1 个进位块中，进位控制信号是 C_{in}，输出信号是 C_0, C_1, C_2 和 C_3；第 2 个进位块中，进位控制信号是 C_3，输出信号是 C_4, C_5, C_6 和 C_7；第 3 个进位块中，进位控制信号是 C_7，输出信号是 C_8, C_9, C_{10} 和 C_{11}；第 4 个进位块中,进位控制信号是 C_{11},输出信号是 C_{12}, C_{13}, C_{14} 和 C_{15}。

3　改进型支持可重构CCS

在当前高性能处理器中，时钟频率不断升高，同时开始支持子字并行，加法器的性能很重要。本文描述的该款支持可重构 16 位进位选择加法器可轻松扩展为 32 位或者 64 位快速加法器。16 位加法器分成 2 个进位块，每个进位块采用对数级联的形式，下面以第 1 个进位块为例，描述 C_i (if C_j=0) 和 C_i (if C_j=1)是如何产生的。第 1 个进位块的进位输出信号分别是 C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7，最低位进位信号为 C_{in}。

根据式(2)，可以得到，当 C_{in}=1 时

$$C_0 = \overline{C_{in}} \bullet G_0 + C_{in} \bullet P_0 = P_0 \equiv C_0(\text{if } C_{in} = 1)$$

当 C_{in}=0 时

$$C_0 = \overline{C_{in}} \bullet G_0 + C_{in} \bullet P_0 = G_0 \equiv C_0(\text{if } C_{\text{in}} = 0)$$

同样，对于 C_1，当 C_{in}=1 时，可以得到

$$C_1 = \overline{C_0} \bullet G_1 + C_0 \bullet P_1 = \overline{P_0} \bullet G_1 + P_0 \bullet P_1 \equiv C_1(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_1 = \overline{C_0} \bullet G_1 + C_0 \bullet P_1 = \overline{G_0} \bullet G_1 + G_0 \bullet P_1 \equiv C_1(\text{if } C_{\text{in}} = 0)$$

同样，对于 C_2，当 C_{in}=1 时，可以得到

$$C_2 = \overline{C_1} \bullet G_2 + C_1 \bullet P_2 = \overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1} \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2 \equiv C_2(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_2 = \overline{C_1} \bullet G_2 + C_1 \bullet P_2 = \overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1} \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2 \equiv C_2(\text{if } C_{\text{in}} = 0)$$

同样，对于 C_3，当 C_{in}=1 时，可以得到

$$C_3 = \overline{C_2} \bullet G_3 + C_2 \bullet P_3 = \overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1 \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2 \bullet G3 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1 \bullet G_2 + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet P_2) \bullet P3} = \overline{\overline{P_0} \bullet G_1 + P_0 \bullet P_1} \bullet (\overline{G_2} \bullet G_3 + G_2 \bullet P_3) + (\overline{P_0} \bullet G_1 + P_0 \bullet P_1) \bullet (\overline{P_2} \bullet G_3 + P_2 \bullet P_3) \equiv C_3(\text{if } C_{\text{in}} = 1)$$

当 C_{in}=0 时，可以得到

$$C_3 = \overline{C_2} \bullet G_3 + C_2 \bullet P_3 = \overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1} \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2 \bullet G3 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1 \bullet G_2 + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet P_2) \bullet P3 = \overline{\overline{G_0} \bullet G_1 + G_0 \bullet P_1} \bullet (\overline{G_2} \bullet G_3 + G_2 \bullet P_3) + (\overline{G_0} \bullet G_1 + G_0 \bullet P_1) \bullet (\overline{P_2} \bullet G_3 + P_2 \bullet P_3) \equiv C_3(\text{if } C_{\text{in}} = 0)$$

对于 C_4，当 C_{in}=1 时，可以得到

$$C_4=\overline{C_3}\bullet G_4+C_3\bullet P_4=$$
$$\overline{C_3(\text{if } C_{\text{in}}=1)}\bullet G_4+C_3(\text{if } C_{\text{in}}=1)\bullet P_4=C_4(\text{if } C_{\text{in}}=1)$$

当 C_{in}=0 时，可以得到

$$C_4=\overline{C_3}\bullet G_4+C_3\bullet P_4=$$
$$\overline{C_3(\text{if } C_{\text{in}}=0)}\bullet G_4+C_3(\text{if } C_{\text{in}}=0)\bullet P_4\equiv C_4(\text{if } C_{\text{in}}=0)$$

按照上述推导步骤，同理可以推导出 C_5~C_7 的表达式。该加法器的结构如图 3 所示。

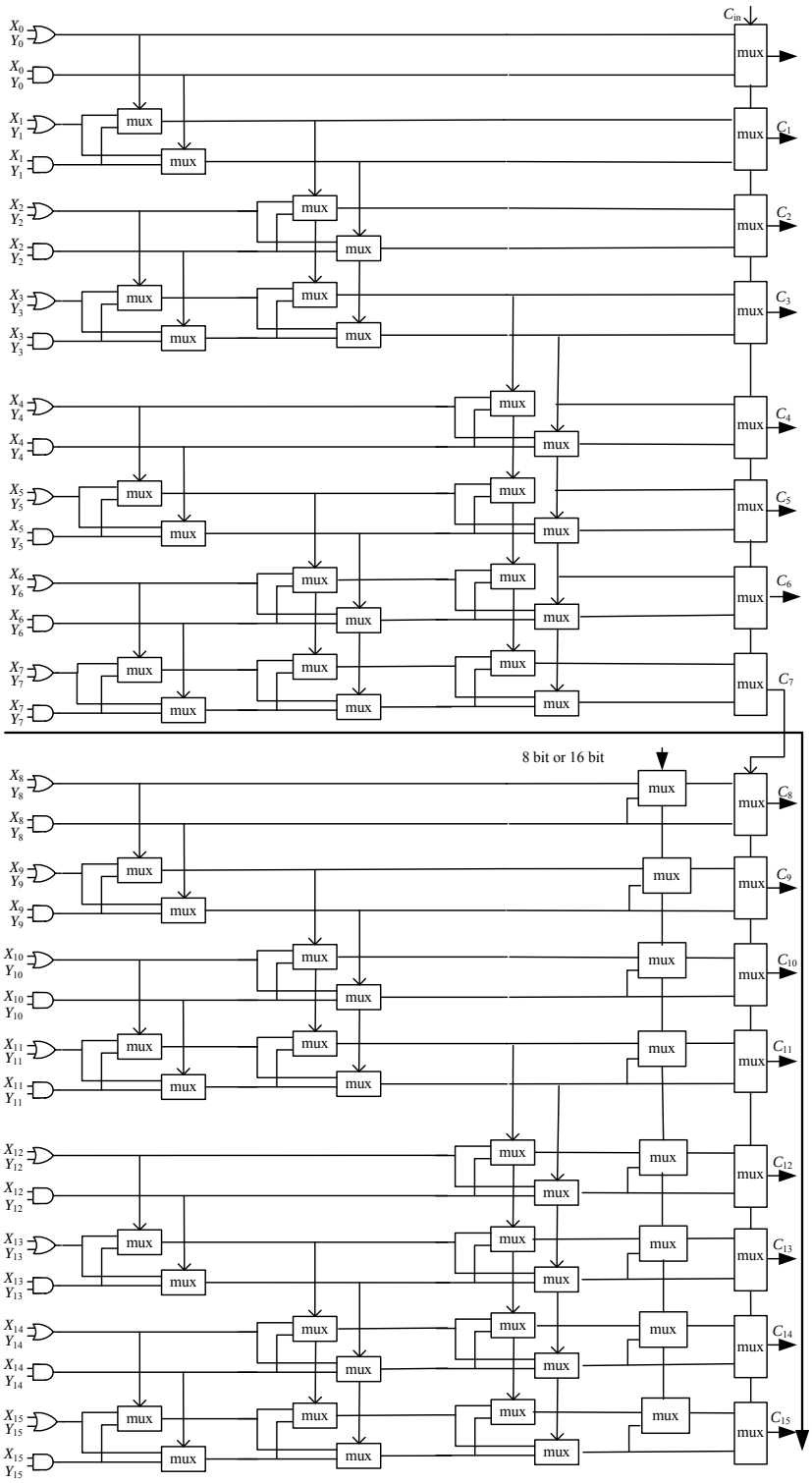

图 3　改进的支持可重构的 CCS

若使该加法器支持 8 位的运算，最简单的办法就是控制 C_7，即当该加法器工作在 8 位模式时，使 C_7 等于 0；工作在 16 位模式时，C_7 正常导通。但是这样做的坏处就是给关键路径增加了延时，因此是不可取的。

通过分析图 3 可以发现，因为 16 位加法器的高 8 位结果最终要由 C_7 选出，所以高 8 位需要额外等待一个传输门的延时。这样就可以将字长工作模式的控制加在高 8 位的进位链上，这样增加的多路选通器正好利用了高 8 位额外等待的那个延时时间，不会给关键路径额外增加延时。当加法器工作在 16 位字长模式时，模式控制 mux 选择上面的输入，这与没有增加模式控制时的电路是一致的；当加法器工作在 8 位字长模式时，模式控制 mux 选择下面的输入，这样，信号 C_7 控制的 mux 的 2 个输入就都为进位为 0 时的进位输出信号，从而实现了进位链的断开，支持了 2 个独立的 8 位数据的运算。

由上图可见，其关键路径包括 5 级 2 选 1 选通器和 1 级与或门，共经过 6 级门电路。而传统的进位选择加法器要经过 8 级逻辑门，该加法器在进位产生上节省了 2 级逻辑门延时。由于该加法器电路结构规则，采用模块化堆叠设计，布线比较简单，有利于版图的规则化。

4　改进型加法器的原理和结构

如图 2 所示，传统的加法器在进位信号产生后再进行求和异或运算，这无疑增加了关键路径的延时。本文利用 CSS 的原理，与提出的改进型 CCS 相结合，并行地计算加法器的“和”，从而进一步缩短关键路径的延时。

对于一位的全加器，其“和”的计算可以由下面的公式得到：

$$S_i=X_i\oplus Y_i\oplus C_{i-1}=$$
$$\overline{C_{i-1}}\bullet(X_i\oplus Y_i)+C_{i\text{-}1}\bullet(X_i\square Y_i)\quad(4)$$

则 1 位加法器的实现如图 4 所示。

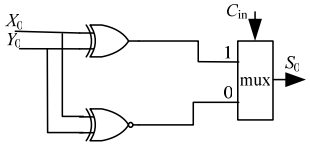

图 4　1 位加法器的进位产生电路

将该电路与图 3 所示的快速进位产生电路相结合，便可以生成 16 位快速加法器的完整电路。为了描述方便，仅以加法器的低 4 位结构描述完整的快速进位选择加法器，如图 5 所示。

传统的 CCS 加法器的“和”的计算是在进位信号计算出之后，与输入数据经过异或逻辑产生的，如图 2 所示。由于异或逻辑一般相当于 2 级逻辑门延时，对于 3 输入的异或门，其延时更大，因此基于传统的先计算进位输出再计算“和”的结构会浪费更多的延时。然而采用同步选择、同步计算的方法，使加法器省去了异或门逻辑的延时，从而又将关键路径缩减了至少 2 个传输门的

延时，结合图 3 中进位链中省掉的 2 个逻辑门的延时，该 16 位加法器比传统的 16 位条件进位选择加法器节省了 4 个逻辑门的延时。

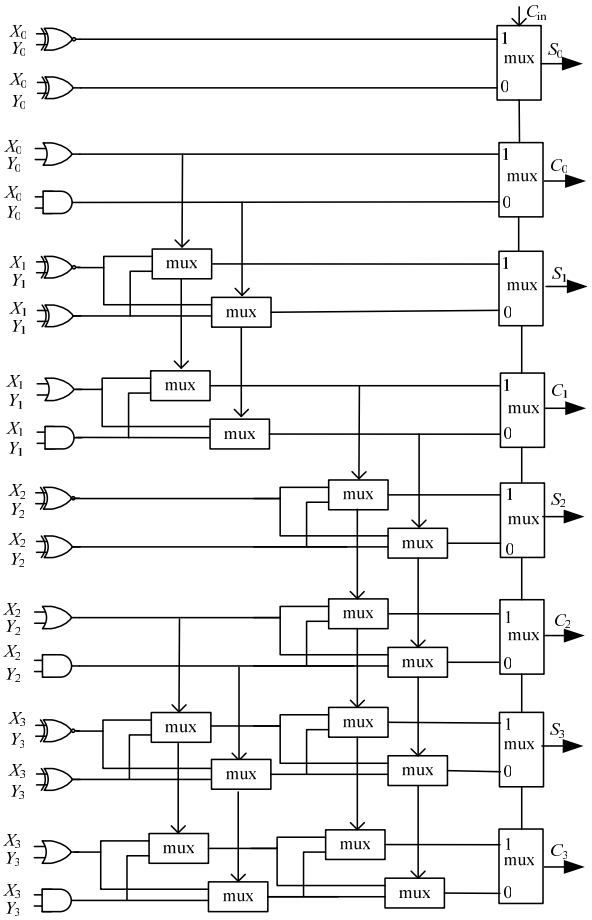

图 5　4 位基于 CCS 和 CSS 的加法器

5　仿真结果

本文采用 Verilog 语言描述了该款 16 位快速可重构加法器，并用 Synopsys 工具进行了综合，采用 0.18 μm 工艺下的标准库，用 HSPICE 进行了时序分析。如图 6 所示，在 0.00°C 和 1.80 V 电压下，该加法器的关键路径的延时为 0.71 ns。相对于传统的条件进位选择加法器，其延时降低了 46%，功耗降低了 5%。如果采用定制设计或者采用更快的单元[5]，该款加法器将会具有更高的性能。

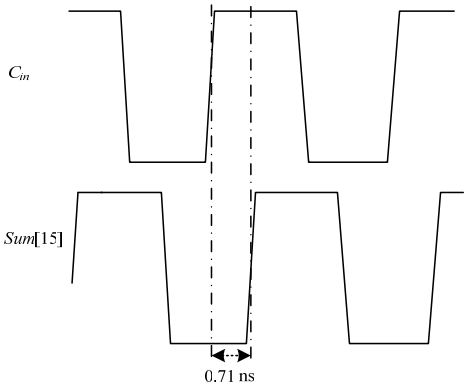

图 6　SPICE 仿真的 CCS 关键路径延时

作为比较，本文同时用 Verilog 语言描述了基于传统结构的 16 位 CCS 加法器，用 Synopsys 工具进行了综合，并用 HSPICE 进行了时序分析。这 2 种加法器在速度、面积、功耗、功耗延时积的比较如图 7 所示。

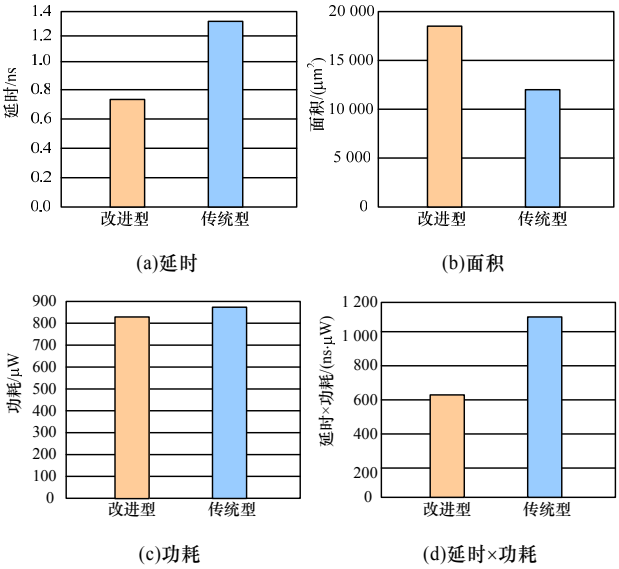

图 7　2 种加法器的性能比较

通过比较可以发现，本文描述的该款加法器的功耗和传统 CCS 加法器相比略有降低，面积有所增大，但是明显减小了延时。改进加法器的延时功耗积比传统的 CCS 加法器降低了将近 40%。因此，该加法器具有更好的特性，适用于 VLSI 数字信号处理器，能够满足高速度、低功耗的要求。

6　结束语

本文给出一款 16 位可重构快速加法器的设计，详细地描述了该款加法器的原理和结构。该款加法器为高性能 DSP 处理器所设计，以满足 DSP 运算的灵活性和高性能，并且具有较小的功耗。同时，该款加法器也适合多媒体、图形处理等专用处理器的应用。此外，该 16 位加法器可以通过级联的方式构成更大字长的加法器，如 32 位或者 64 位可重构加法器。本文用 Verilog 语言描述了此款 16 位可重构快速加法器以及 16 位传统 CCS 加法器，并用 Synopsys 的 DC 工具进行了综合，在 HSPICE 中进行了时序分析。采用 0.18 μm 工艺标准库，在 0.00°C 和 1.80 V 电压的工作条件下，HSPICE 的报告显示，该款 16 位加法器的关键路径延时为 0.71 ns，相对于传统的 CCS 加法器，其延时降低了 46%，功耗降低了 5%。如果采用动态逻辑和全定制，该加法器将具有更高的性能。

参考文献

[1] Kai Hwang. Computer Arithmetic——Principles, Architecture and Design[M]. New York, USA: John Wiley and Sons Inc., 1979.

[2] Chang T Y, Hsiao M J. Carry-select Adder Using Single Ripple-carry Adder[J]. Electronic Letters, 1998, 34(22): 2101-2103.

[3] Tyagi A. A Reduced-area Scheme for Carry-select Adders[J]. IEEE Trans. on Computers, 1993, 42(10): 1163-1170.

[4] Chen Waikai. The VLSI Handbook: Adders[M]. [S. l.]: CRC Press, 2000.

[5] Kuo J B, Chen S S, Chiang C S, et al. A 1.5 V BiCMOS Dynamic Logic Circuit Using a “BiPMOS Pull-down” Structure for VLSI Implementation of Full Adders[J]. IEEE Transactions on Circuit and Systems, 1994, 41(4): 329-332.

编辑　任吉慧