

基于 FPGA 的高速流水定点乘法器的设计

吉伟, 黄士坦

(西安微电子技术研究所, 陕西 西安 710075)

摘要:目前,多数定点高速乘法器的速度都在百兆以下。在比较各种定点乘法器的基础上,提出了一种基于 Xilinx 的 Virtex FPGA 系列器件的快速流水定点乘法器的实现方法,可将乘法速度提高至 150MHz 以上,大大提高了运算速度。文中以 24×24 位乘法器为例,给出了 VHDL 代码与综合仿真布线结果。此乘法器已应用于工程实践中,并且收到了良好的效果。

关键词:高速流水定点乘法器; Virtex 器件; FPGA

中图分类号:TP332.2+2

文献标识码:A

文章编号:1673-629X(2007)09-0199-04

Design of High Speed Pipeline Fixed-point Multiplier Based on FPGA

Ji Wei, HUANG Shi-tan

(Xi'an Institute of Micro-electronics Technology, Xi'an 710075, China)

Abstract: At the present time the speed of most high speed multiplier is below 100MHz. Comparing with other multipliers, have put forward a method of realizing high speed multiplier based on Xilinx Virtex FPGA apparatus which could improve the speed to 150MHz. Have list the VHDL code for the example of 24×24 bit multiplier and the result of synthesis in the text. The multiplier designed has been used in project and the effect is quite nice.

Key words: high speed glide multiplier; Virtex; FPGA

0 引言

在高性能微处理器和 DSP 芯片设计中,乘法器都是一个关键部件,优化乘法器对提高整个处理器性能起着至关重要的作用。而定点乘法器更是乘法运算的基础和核心部件,因此,提高定点乘法器的性能就显得非常重要。

为了优化乘法器的速度和性能,人们提出了很多高效的算法和结构,比较常见的有利用 BOOTH 编码实现的高速乘法器^[1,2]。此乘法器的结构包括修正 BOOTH 编码部分,部分积产生,部分积压缩以及平方根求和电路。乘数首先进行 BOOTH 编码,同时被乘数进行移位操作,完成与 $\{-2, -1, 0, 1, 2\}$ 的相乘。然后在部分积产生模块中根据 BOOTH 编码值选择被乘数产生相应的部分积。部分积压缩模块将部分积压缩成两行,最后由高速求和电路进行求和相加得到最终的乘积。其中,运算单元采用 $4:2$ 压缩实现,部分积求和采用常用的华莱士压缩树^[1]。

收稿日期:2006-12-23

作者简介:吉伟(1982-),男,陕西韩城人,硕士研究生,主要从事计算机系统结构及硬件开发方面的研究;黄士坦,研究员,博士生导师,研究方向为计算机卸载并行技术及图像处理。

试验结果:FPGA 选用 XILINX 的 Virtex-II 的 2v1000fg256-4,完成一次 16 点的乘法需要 16.058ns,最高频率为 62.27MHz^[3]。

以上的乘法器结构虽然使乘法速度有所提高,但是由于没有充分考虑到所使用的 FPGA 器件的结构,因此并没有合理地利用器件本身的结构来进行设计,因而没有使乘法器达到应有的运算速度。

笔者认为硬件设计应当与所使用的可编程器件的结构紧密联系起来,充分利用期间本身提供的部件和单元,合理布局,尽量减小逻辑和布线延迟。在此基础上才能最大可能地提高乘法器的运算速度。因此,首先要研究所使用的 FPGA 的内部结构。

1 Virtex 器件的特点

Virtex 器件配置逻辑块(CLB)的基本组成是逻辑单元(LC, logic cell),它由一个四输入的逻辑函数产生器、一套进位逻辑和一个存储单元组成,逻辑函数产生器实现的逻辑函数可以直接输出,也可以经 D 触发器锁存后输出。

基于 Virtex 器件的结构,每个 slice 可以实现 2bit \times 2bit 乘法器的运算,结合 slice 外专门设计的进位通

路,每列 slice 可以方便地实现一路 $2 \times n$ 的高速乘法器设计,其中 n 的大小取决于所选择器件的规模。对于乘法运算中部分乘积的求和运算,因为 Virtex 器件提供了非常丰富的布线资源,一般都采用 Wallace 树形结构来求和。由于每个 slice 都固定带有独立的 D 触发器资源,因此可以尽量将乘法器设计为流水形式,而不用担心会浪费额外的系统资源。

2 设计原理

以八位无符号数相乘为例,设: $P = A \times B$

$$A = \sum_{i=0}^7 a_i 2^i \quad B = \sum_{i=0}^7 b_i 2^i$$

将积的计算式展开可以得到:

$$P = A \times ([b_7 b_6] \times 2^6 + [b_5 b_4] \times 2^4 + [b_3 b_2] \times 2^2 + [b_1 b_0])$$

记: $P_I = A \times [b_1 b_0]$, $P_{II} = A \times [b_3 b_2]$,

$P_{III} = A \times [b_5 b_4]$, $P_{IV} = A \times [b_7 b_6]$,

$P_A = P_{II} \times 2^2 + P_I$ $P_B = P_{IV} \times 2^2 + P_{III}$

则 $P = P_{IV} \times 2^6 + P_{III} \times 2^4 + P_{II} \times 2^2 + P_I =$

$$P_B \times 2^4 + P_A$$

P_I, P_{II}, P_{III} 和 P_{IV} 为计算过程中得到的部分积,

在 Virtex 系列器件中,可以用四列各五个 slice 实现, P_A, P_B 的计算各需一个 10bit 的加法器,最后用一个 12bit 的加法器得到最后的计算结果。

对于有符号数的乘法,也可以与无符号数的乘法统一成一种形式。

此外,由于 Virtex-II 系列器件内部带有 18×18 位的高速乘法器,则可以考虑将乘数和被乘数拆分,然后调用此乘法器来实现,下文将简称其为 18×18 结构的乘法器。

3 VHDL 代码及性能比较

3.1 $2 \times n$ 位结构的乘法器

基于这种思想,可以实现多位数的定点乘法。以 24×24 位无符号乘法器为例,其 $2 \times n$ 结构的乘法器的 VHDL 代码如下:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiply is
port (clk:in std_logic;
      a:in std_logic_vector(23 downto 0);
      b:in std_logic_vector(23 downto 0);
      cout:out std_logic_vector(47 downto 0));
```

end multiply;

architecture multiply of multiply is

```
signal p1:std_logic_vector(47 downto 0);
signal p2:std_logic_vector(45 downto 0);
signal p3:std_logic_vector(43 downto 0);
signal p4:std_logic_vector(41 downto 0);
signal p5:std_logic_vector(39 downto 0);
signal p6:std_logic_vector(37 downto 0);
signal p7:std_logic_vector(35 downto 0);
signal p8:std_logic_vector(33 downto 0);
signal p9:std_logic_vector(31 downto 0);
signal p10:std_logic_vector(29 downto 0);
signal p11:std_logic_vector(27 downto 0);
signal p12:std_logic_vector(25 downto 0);
signal pa:std_logic_vector(47 downto 0);
signal pb:std_logic_vector(43 downto 0);
signal pc:std_logic_vector(39 downto 0);
signal pd:std_logic_vector(35 downto 0);
signal pe:std_logic_vector(31 downto 0);
signal pf:std_logic_vector(27 downto 0);
signal pg:std_logic_vector(23 downto 0);
signal ph:std_logic_vector(19 downto 0);
signal pi:std_logic_vector(15 downto 0);
signal pj:std_logic_vector(11 downto 0);
signal pk:std_logic_vector(7 downto 0);
begin
```

```
process(a,b,clk)
```

```
begin
```

```
if clk'event and clk='1' then
```

```
p1<="000000000000000000000000"&a*(b(1)&b(0));
```

```
p2<="000000000000000000000000"&a*(b(3)&b(2));
```

```
p3<="000000000000000000000000"&a*(b(5)&b(4));
```

```
p4<="000000000000000000000000"&a*(b(7)&b(6));
```

```
p5<="000000000000000000000000"&a*(b(9)&b(8));
```

```
p6<="000000000000000000000000"&a*(b(11)&b(10));
```

```
p7<="000000000000000000000000"&a*(b(13)&b(12));
```

```
p8<="000000000000000000000000"&a*(b(15)&b(14));
```

```
p9<="000000000000000000000000"&a*(b(17)&b(16));
```

```
p10<="000000000000000000000000"&a*(b(19)&b(18));
```

```
p11<="000000000000000000000000"&a*(b(21)&b(20));
```

```
p12<="000000000000000000000000"&a*(b(23)&b(22));
```

```
end if;
```

```
end process
```

```
process(clk)
```

```
begin
```

```
if clk'event and clk='1' then
```

```
pa<=p1+(p2&"00");
```

```
pb<=p3+(p4&"00");
```

```
pc<=p5+(p6&"00");
```

```
pd<=p7+(p8&"00");
```

```

pe<= p9 + (p10 & "00");
pf<= p11 + (p12 & "00");
end if;
end process;
process(clk)
begin
    if clk'event and clk = '1' then
        pg<= pa + (pb & "0000");
        ph<= pc + (pd & "0000");
        pi<= pe + (pf & "0000");
    end if;
end process;
process(clk)
begin
    if clk'event and clk = '1' then
        pj<= pg + (ph & "00000000");
        pk<= pi & "0000000000000000";
    end if;
end process;
process(clk)
begin
    if clk'event and clk = '1' then
        cout<= pj + pk;
    end if;
end process;
end multiio;

```

首先将乘数 b 从最低位开始每两位进行合并, 得到 $p_1 \sim p_{12}$ 。然后按照上文介绍的方法进行 $2 \times n$ 位的乘法, 得到部分积 $pa \sim pf$, 之后用华莱士树的方法将部分积分两步进行合并。整个乘法过程按流水方式实现, 提高了并行度^[4]。其仿真波形如图 1 所示。

用 Virtex-II 1000-4fg256 进行综合布线, 占用了 12 个 24×2 位的乘法器, 11 个加法器, 得到的全局最小时钟为 6.111ns, 最高频率可达到 163.639MHz。

若用 Virtex-II 1000-6fg256 得到的全局最小时钟为 5.194ns, 最高频率可达到 192.530MHz。大大提高了乘法速度。

3.2 18×18 结构的乘法器

还可以考虑将乘数和被乘数拆分,直接调用 Virtex-II 系列器件内部的 18×18 位高速乘法器。

仍以 24×24 位乘法为例来说明, 其 VHDL 代码如下^[5]:

24×24 位无符号乘法(采用 18×18 的结构)

代码:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
```

```

entity multiply is
port
(clk:in std_logic;
a:in std_logic_vector(23 downto 0); --被乘数
b:in std_logic_vector(23 downto 0); --乘数
cout:out std_logic_vector(47 downto 0) --乘积结果
);
end multiply;
architecture multiply of multiply is
signal a1,b1:std_logic_vector(17 downto 0); --被乘数和乘数的低18位
signal a2,b2:std_logic_vector(23 downto 18); --被乘数和乘数的高6位
signal cout1:std_logic_vector(47 downto 0); --部分积之和
signal cout2:std_logic_vector(47 downto 0); --部分积之和
signal a1b1,a2b1,a1b2,a2b2:std_logic_vector(47 downto 0); --18x18,6x18,18x6,6x6 位的部分积
begin
process(a,b,clk) --同时计算4个部分积
begin
if clk'event and clk='1' then
a1b1<="00000000000000"&(a(17 downto 0) *
b(17 downto 0));
a2b1<="000000"&(a(23 downto 18) *
b(17 downto 0))&"00000000000000000000";
a1b2<="000000"&(a(17 downto 0) *
b(23 downto 18))&"000000000000000000000000";
a2b2<=(a(23downto18) * b(23downto18))&
"00000000000000000000000000000000000000000000";
end if;
end process;
process(clk) --两两相加求部分积之和
begin
if clk'event and clk='1' then
cout1<=a1b1+a2b1;
cout2<=a1b2+a2b2;
end if;
end process;
process(clk) --部分积求和得到最终结果
begin
if clk'event and clk='1' then
cout<=cout1+cout2;
end if;
end process;
end multiply;

```

波形如图 2 所示。

此程序采用流水实现定点无符号乘法, 仅用 3 拍即可实现。器件采用 Virtex-II xcv1000 fg256-6, 综合布线后得到最小时钟周期为 5.891ns, 171.51MHz。

其中,占用资源有:一个 18×18 位,两个 18×6 位,一个 6×6 位的乘法器,3个48位加法器,7个48位的寄存器。

对于有符号数的相乘结构与此大致相同,只需将符号位考虑在内即可。综合布线后仿真,器件采用 Virtex-II xcv1000 fg456-6,最高频率可达到195.160MHz。

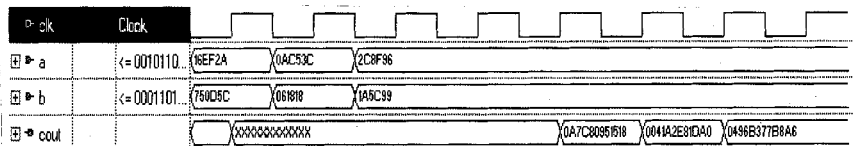


图1 2xn结构乘法器仿真波形



图2 18x18结构乘法器仿真波形

参考文献:

- [1] 梁峰,邵志标,梁晋. Radix-16 Booth流水线乘法器的设计[J]. 西安交通大学学报, 2006, 40(10): 1111-1114.
- [2] Booth A D. A signed binary multiplication technique[J]. Quarterly Journal of Mech & Appl Math, 1951, 4(2): 236-240.

(上接第195页)

幅,不再一一描述。希望文中描述的声卡模式自动切换系统中的进程间通讯机制应用能够给大家带来借鉴作用。

参考文献:

- [1] Richter J. Programming Applications for Microsoft Windows [M]. 4th Edition. [s.l.]: Microsoft Press, 1999.

(上接第198页)

参考文献:

- [1] Wright G R, Stevens W R. TCP/IP 详解 卷2:实现(英文版)[M]. 北京:机械工业出版社, 2002.
- [2] Stevens W R. TCP/IP Illustrated Volume I: The Protocols [M]. Beijing: China Machine Press, 2000.
- [3] 李勇,戴瑜兴. 基于UDP协议的实时监控系統[J]. 电子技术, 2003(11): 37-40.
- [4] 费仁元,王民,徐洪安. 产品生产服务过程中的网络监

4 总结

通过比较,可以看到采用相同的器件,该方法已经将运算速度提高了150MHz以上,大大提高了运算速度,而且采用流水技术,实现了并行计算。此外,这种方法还具有结构电路简单的明显优势。

在工程实践中,已经将这种方法应用于浮点乘法器的设计当中,收到了良好的效果。

- [3] 李小进. 定点符号高速乘法器的设计与FPGA实现[J]. 微电子学与计算机, 2005, 22(4): 119-125.
- [4] 常静波,郭立. 一种3级流水线Wallace树压缩器的硬件设计[J]. 微电子学与计算机, 2005, 22(1): 160-165.
- [5] 侯伯亨,顾新. VHDL 硬件描述语言与数字逻辑电路设计[M]. 西安:西安电子科技大学出版社, 2000.

- [2] 吴群飞. Hook技术在监控FSpice中的应用研究[J]. 计算机应用研究, 2005(4): 185-186.
- [3] 杨宁学. 内存影射文件及其在大数据量文件快速存取中的应用[J]. 计算机应用研究, 2004(8): 187-188.
- [4] 王文磊. 多线程编程技术实现经典进程同步问题[J]. 计算机技术与发展, 2006, 16(3): 110-112.
- [5] 谈蓉. Win32系统下进程间通讯[J]. 电脑开发与应用, 2003(10): 19-20.

控技术[J]. 机械工程学报, 2003(9): 35-37.

- [5] Yuan F, Huang C. Performance Analysis of Resilient Packet Ring with Single Transit Buffer [EB/OL]. 2002. [http://www.scc.carleton.ca/faculty/huang/ICT2002 final version. pdf](http://www.scc.carleton.ca/faculty/huang/ICT2002%20final%20version.pdf).
- [6] Francisco M, Yuan F, Huang C, et al. A Comparison of Two Buffer Insertion Ring Architectures with Fairness algorithms, Anchorage [EB/OL]. 2002. [http://www.scc.carleton.ca/faculty/huang/mark-icc03. pdf](http://www.scc.carleton.ca/faculty/huang/mark-icc03.pdf).