基于约束数据捆绑两相握手协议的8位异步Booth乘法器设计

何安平[1],刘晓庆[1],陈 虹[2]
(1. 兰州大学信息科学与工程学院,甘肃兰州 730000;2. 清华大学微电子学研究所,北京 100084)

摘 要: 以乘法器为代表的算术运算单元是现代数字系统的核心之一,其计算速度在很大程度上影响整个芯片的运算效率.本论文提出了一种改进的Booth乘法算法,其核心思想是先移位、再压缩,最后求和,减少了各模块间的耦合性,有利于控制电路的简化.本论文依据纯异步电路系统的设计方法,采用"约束数据捆绑"两相握手通讯协议的Click微流水线,根据控制和数据处理分离的策略,实现了这种改进算法的8位乘法器,并在FPGA上进行了验证.在45nm工艺制程的FPGA条件下,与相同体系结构的同步乘法器相比,这种异步乘法器在面积和功耗大体相同的情况下,运算速度大体提升超过12倍.

关键词: Booth算法; 异步设计; 两相约束数据捆绑握手协议; Click异步控制器; 微流水线
中图分类号: TN492 **文献标识码:** A **文章编号:** 0372-2112 (2018)04-0961-08
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.04.026

Study of 8-Bit Booth Asynchronous Multiplier Based on Two-Phase Handshake Protocol with Bounded Bundle Data

HE An-ping[1], LIU Xiao-qing[1], CHEN Hong[2]
(1. *School of Information Science & Engineering, Lanzhou University, Lanzhou, Gansu* 730000, *China*;
2. *Institute of Microelectronics, Tsinghua University, Beijing* 100084, *China*)

Abstract: The arithmetic cell, especially the multiplier, is one of the key components of the modern digital systems, which could affect the efficiency of the whole system dramatically. In this article, we propose an extended Booth algorithm, which is different from the traditional one with the strategy of keeping all the partial products during shifting, then compressing and finally summing. This new one could reduce the coupling of each function and simplify the control part of the circuit. Moreover, by following the asynchronous design methodology, the control circuit with the Click based asynchronous micro pipeline that obeys a two-phase handshake protocol with "bounded bundle data" is implemented. With the policy of isolating data processing from control, an 8-bit asynchronous multiplier is designed and then implemented with FPGA. With the FPGA of 45nm technology and the same architecture, this asynchronous Booth multiplier is over 12 times faster than the synchronous one while keeping the area and power almost same.

Key words: Booth algorithm; asynchronous design methodology; two-phase handshake protocol with bounded bundle data; Click based asynchronous controller; micro pipeline

1 引言

高性能数字乘法器是处理器和算法芯片的核心部件,是各类复杂计算的基础与核心,特别是完成高性能实时数字信号处理和图像处理的关键所在,乘法器的效率直接影响芯片的性能.数字乘法器的效率主要体现在两个方面,即面积和速度.选择不同的设计方法和实现算法,对乘法器的面积和速度的影响非常大.

数字乘法器是一种二进制的算术逻辑单元,因为数字电路系统架构在布尔逻辑之上,所以需要一种将算术转换成逻辑的机制,这种机制就是数字乘法器算法的本质.数字乘法器的算法已经比较成熟,最直观的阵列算法,从乘数的低位开始,依次计算每一位与被乘数的乘积(部分积),而后将部分积相加得到积,对于 n

收稿日期:2016-08-11;修回日期:2017-03-21;责任编辑:孙瑶
基金项目:国家自然科学基金(No.61402121,No.61073193,No.61300230);广西高校复杂系统与智能计算机重点实验室基金(No.2016CSCI05);甘肃省重点科技基金(No.1102FKDA010);甘肃省自然科学基金(No.1107RJZA188);甘肃省科技支撑计划(No.1104GKCA037)

位乘法器而言,需要 $n(n+1)$ 个全加器和 n^2 个“与”门,实现这种算法的乘法器计算速度慢,面积与功耗高.

实用的数字乘法器广泛采用 Booth 算法[1]. 经典 Booth 算法将乘数等分为若干乘数段,乘法问题规约为各被乘数与乘数段的部分积之和. 具体而言,在 Booth 算法中,可以根据乘数段的二进制数据特征,将各段同被乘数的乘法映射为等效的移位和减法运算来求得关于此乘数段的部分积,而后再进行多次相加求积,或者多次压缩后单次相加求积.

本篇论文首先对 Booth 算法做了改进,采纳了经典 Booth 算法移位、压缩和求和的基本框架,取消了移位后做减法计算此乘数段部分积的办法,而在移位过程中保留多个部分积,并对其多次压缩后加法求积. 这种改进增强了功能模块内部的内聚性,减弱了模块间的耦合关系,简化了乘法器控制电路的实现. 此外,本文采用异步设计方法来实现这种改进型的 Booth 算法,控制部分使用了易于时序分析[2]的 Click 异步控制器[3]组成的微流水线,功能电路使用组合逻辑实现,二者由触发器联结在一起,即异步微流水线通过管理触发器的导通时机,间接维护组合电路的计算次序,三者合作完成一次/多次乘法计算,构成了一种数据通路(Data-Path)式的计算结构. 本文将这种异步乘法器部署在 FPGA 上进行了仿真,在 45nm 工艺制程[4]的 FPGA 条件下,与相同体系结构的同步乘法器相比,这种异步乘法器在面积和功耗大体相同的情况下,运算速度大幅增加.

2 Booth 乘法器与异步电路设计基础

本节通过对比相关工作,介绍 Booth 算法与异步电路设计方法.

2.1 Booth 乘法器

Booth 算法是一种广泛采用的高效乘法器实现方法,这种方法首先计算被乘数与乘数各段的部分积,而后对其压缩求和得到最终的积. 其中部分积的产生和合并是关键,部分积的计算不仅影响计算速度,而且决定整个乘法器的规模.

Booth 算法实现为 Booth 乘法器时,需要通过一定的控制机制来协调各模块,使之互相配合完成乘法运算. 当前的主流技术使用时钟作为各模块的同步信号,已经研发了多种 Booth 乘法器. 在文献[5]中,设计了一种新型的基于 Radix-16 Booth 算法的 32 位乘法器结构,此结构不产生由进位组成的附加部分积,而且也减少了传统 Booth 算法生成复杂倍数时引入的比较大的电路延时,在保证速度的前提下,实现此结构所需的硬件资源得到了有效降低. 文献[6]中提出了一种改进的 Booth 算法,采用的是将乘法器的编码器、加法器树、快速加法器等分离出来,用指令表示各个部分的联系,通过多条指令描述基本单元的互联关系,以构成不同结构的乘法器. 在文献[7]中,采用并行的半脉冲阵列[8]结构,多米诺电路等型式,设计了一种面积较小的 8×8 乘法器.

乘法器的性能除了受算法影响之外,与选用的设计方法也直接相关. 基于异步设计[9]方法的乘法器采用微流水线[10]来控制各功能模块的时序. 在文献[11]中,并行阵列结构设计使计算时间更短并且具有更低功耗特点,仿真表明相比传统的 Booth 算法异步阵列结构计算时间减少 55%,而并行阵列结构的乘法器时间减少 40%,并具有低功耗特点. 文献[12]中介绍了一种采用 IEEE-754 双精度浮点数异步乘法器技术,目的是减少功耗,控制通路采用四相锁存控制器和非对称延迟部分. 实验表明异步乘法器功耗比同步低 16%,采用 180nm UMC 技术,面积占用 0.31%. 文献[13]采用 12×12 阵列乘法器,提出一种新的异步流水线,相对传统的异步阵列乘法器,新的阵列乘法器有 76.5% 吞吐率,43.8% 时间,38.7% 低静态功耗,该多重阈值阵列乘法器更适合于高速低功耗异步乘法器设计. 文献[14]中,实现低功耗的异步乘法器设计从三个方面实现低功耗,分别是异步控制,radix-2 算法和寄存器. 运用 Booth 算法功耗相对于同步乘法器分别减少 55%,23% 和 12%. 文献[15]中,介绍一种结合捆绑数据面积效率和数据计算时间的异步设计方法,该设计采用 16×16 乘法器进行估测,仿真结果表明计算时间相比同等的同步电路设计节约了 20%.

基于以上相关工作分析可知:与相应的同步设计相比,基于异步微流水线的 Booth 结构乘法器在功耗和时间方面具有较大优势. 为了简化异步微流水线设计,进一步提升异步 Booth 结构乘法器性能,本文从两个方面进行改进:一方面采用约束数据捆绑协议的 Click 异步控制器实现微流水线控制结构来简化控制机制;另一方面在借鉴经典 Booth 算法移位、压缩和求和的基本框架的基础上,保留被乘数与乘数段的部分积求解过程的中间结果,增加压缩功能,增强各功能模块的独立性以提高乘法器的计算效率.

2.2 基于 Click 的异步控制原理

自上世纪 70 年代晶体管技术出现以后,同步设计几乎成为数字系统的设计方法的代名词. 但当前的工艺已经趋向制造极限,12 纳米向 7 纳米的转变已经放缓,“极有可能首次背离摩尔定律”(John Gustafson, AMD 首席设计师). 制造工艺的巨大进步所导致的时钟歪斜、电源分布等问题,是同步设计方法的严峻挑战,同步设计方法本身无法提供这些严峻问题的解决方案,只能大量采用 GALS(全局异步和局部同步)设计方法,即采用了少量异步电路的多核技术,来缓解上述

挑战[16].

现代异步设计引入了基于微流水线设计方法,这种设计方法的核心是异步控制器电路,用于实现握手通讯协议和协调电路功能. 相比时钟方案,异步电路采用局部通信模式,以握手协议完成异步控制,不需要庞大的时钟分布网络,解决了时钟扭曲的问题. 异步电路空闲时几乎没有功耗,使整个系统的功耗得到有效控制. 这种异步设计方法在低功耗、低电磁辐射、低散热、模块化等多个方面优势明显[17].

异步设计方法的核心是异步控制器电路,目前主流的异步控制器单元具有三类,即 CElement[18]、GasP[19]和 Click[3]. CElement 由 Muller 于上世纪 50 年代提出,是应用最广的异步控制单元[20],实现了基于"数据捆绑"的握手协议,这种电路在握手通讯过程中,由于没有对数据进行任何约束,后期需要大量的时序验证工作才能保证电路的正确性. 而 GasP 和 Click 电路采用"约束数据捆绑"的握手协议[21],将通讯和数据管理分离为不同事件,这种事件分离的机制从原理上保证了电路的时序,与相对时序的分析[2]配合使用,可以显著简化异步设计方法. 论文中采用 Click 异步控制器作为微流水线的控制单元.

Click 电路最早由 Peeters 和 Willem 在 2010 年提出[3],实现了"约束数据捆绑"两相握手通讯协议. 异步控制器间以 req(请求)和 ack(应答)信号进行握手通讯,两个信号变化间实现数据传输,并且以 Fire(激发)信号管理数据传输,如图 1 所示.

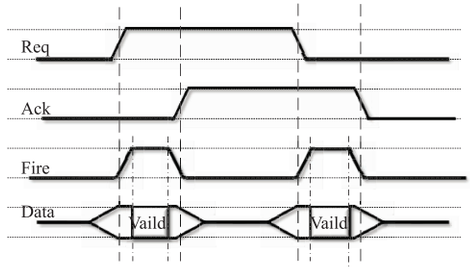


图1　"约束数据捆绑"两相握手通讯协议原理图

Click 异步控制电路的原理图[3]如图 2 所示,有两对输入/输出来实现"约束数据捆绑"两相握手通讯协议,左边一对(in_R/in_A)用于和左方的 Click 通讯,右方的另一对(out_R/out_A)用于和右方的 Click 通讯. 当输入握手请求信号 in_R 升高时,通过异或门使信号变高,然后与同或门输出的信号经过与门 and,使触发器 FF 工作,同时激活了局部 Fire,而后触发器输出的值将分别传到 in_A 和 out_R. 其中 in_A 表明应答上一个 Click 电路,而 out_R 表示进行下一次的握手请求. 之后左右两级的 Click 模块将会重置此 Click 的 in_A 和 out_R,同时取消了局部 Fire. 局部 Fire 在激活和取消的时间段内,进行数据处理.

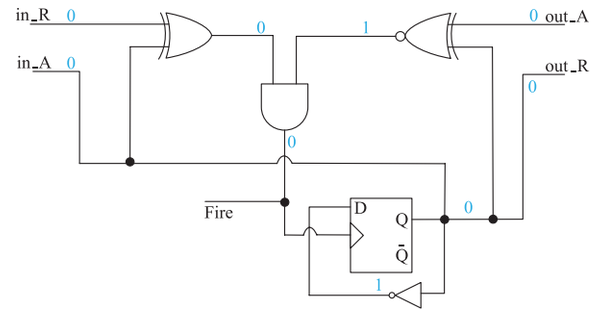


图2　Click电路原理图

3　Booth 乘法器改进与异步实现

在本文的异步乘法器基于算法 1 的 Booth 算法,这种算法的改进内容是,去掉了移位后可能进行的减法运算,增加部分积压缩次数,在乘法器的最后阶段进行一次加法操作. 对于位宽比较小的乘法器而言,算法 1 所示的 Booth 乘法器将具有比较高的性能.

算法 1　Extended 8-bit Booth Algorithm

```
01:procedure multiplier(A,B,Out)
02:Input:A,B;Output:Out;
03:for(state=0;state<2;state++)
04:for(count=0;count<3;count++)begin
05:      if(state=0 && count=0)begin
06:           Shifter←{A[00,01,10,11(01)],B};
07:                Output:4 partial products(first[3:0]);
08:      Compressor ←{first[3:0]};
09:                 Output:4 compressed values(one[1:0]);
10:        end
11:else if(state=1 && count=1)begin
12:           Shifter←{A[00,01,10,11(10)],B};
13:   Output:4 partial products(second[3:0]);
14:        Compressor ←{second[3:0]};
15:          Output:4 compressed values(two[1:0]);
16:end
17:else if(state=1 &&count=2)
18:           Compressor ←{one[1:0],two[1:0]};
19:             Output:4 compressed values(three[1:0]);
20:else;
21:end
22:   Carry Look-ahead Adder←{three[1:0]};
23:                              Output:CLA value(Out);
24:return
25:end procedure
```

3.1　改进型 Booth 算法

令 A 和 B 是 8 位二进制数,B 等分为 4 个两位二进制数段,令 G_i 为一个两位二进制数段,若 G_i 的两位的

值都为1,即等于十进制3的时候,经典算法将A左移两位,而后再减去A本身,得到部分积.本文提出的改进型Booth算法中,将分别产生两个部分积:A左移1位,和A本身,而后将所产生的8个部分积经过两级压缩和一级求和得到最终的A和B积.这种方法将传统Booth算法的多次求和(差)简化为单次,虽然增加了乘法器面积,但分隔了各模块的功能,可以显著提升乘法器的整体吞吐量.算法1是本文提出的改进型Booth算法的伪码.

3.2 多部分积生成方法

在8位异步乘法器的移位电路中,输入A,B两个值,将A分为(A_7A_6)(A_5A_4)(A_3A_2)(A_1A_0)4组,分别与B放入4个移位编码器中进行计算,最后得到的结果是一个15位的二进制数,其中($A_{i+1}A_i$)B分为(00)B,(01)B,(10)B以及(11)B共4种情况.

表1中,这里将取最低2位A_1A_0说明移位编码器的移位策略.在移位算法中,移位编码器将进行2次计算,其中在$state_0$计算时,(01)b相乘结果为b,(10)b结果为b向左移动1位,而当计算(11)b时,分为(01)b和(10)b两种情况,此时只会进行(01)b的计算,即部分积PP_{00}为b.而在$state_1$计算时,当a为(00),(01)和(10)三种情况,得到的PP_{00}为0,只有当为(11)情况下将进行(10)b的计算,即PP_{00}等于b向左移动1位.

表1 移位策略

A_1A_0	B	PP_{00}	
		$State_0$	$State_1$
00	b	0	0
01	b	b	0
10	b	b <<1	0
11	b	b	b <<1

在整个移位编码器中,如图3所示,输入a(即$A_{i+1}A_i$),B,Sh以及标志位State,其中Sh代表a在A中分组位置.输入变量之后,便判断State值,同时分配3个两位的变量ax_2bit,ay_2bit,az_2bit.当State=0时,则进行第一次的计算,继而判断a[1] && a[0]的值.当值为0时,将a值赋给az_2bit,否则表明a的值为11,则把a分为01和10两种情况,分别赋予给ax_2bit,ay_2bit.而当在State=0的条件下,仅将ax_2bit=2'b01的值赋给az_2bit.

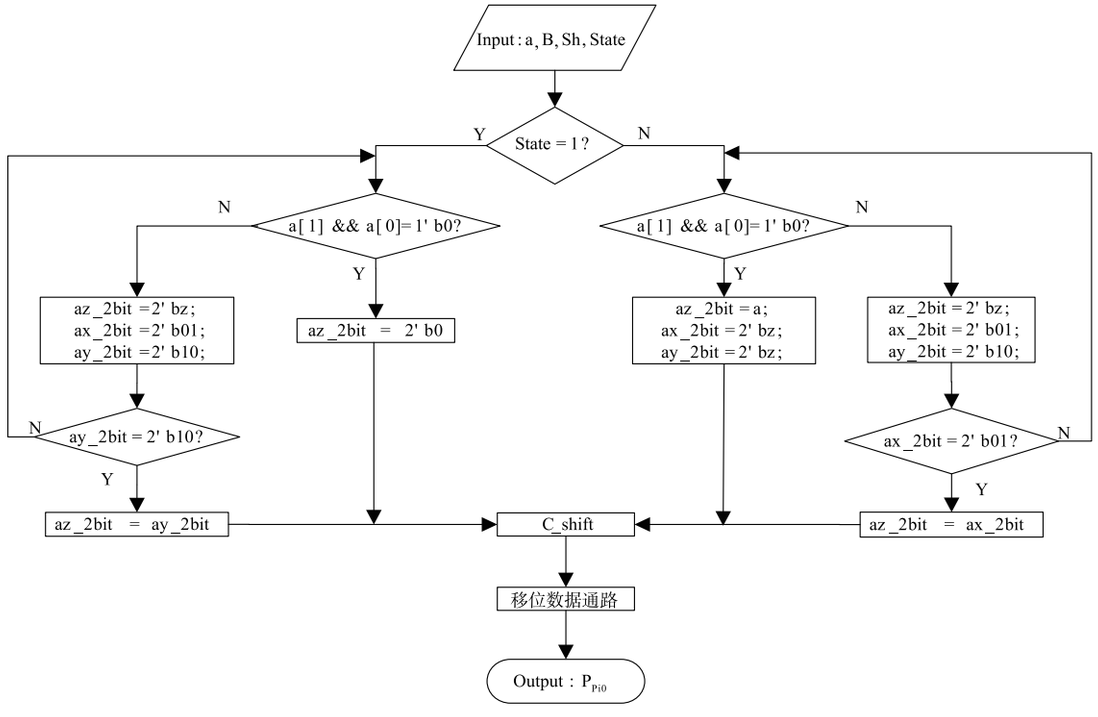


图3 移位编码器流程图

当State=1时,表明该移位编码器需要进行第二次的计算,同理判断a[1] && a[0]的值,当的值为0时,此时的az_2bit的值赋予0.当值为1时,表明a为11,仅把ay_2bit=2'b10赋给az_2bit.

通过az_2bit与Sh计算得到移位标志位C_shift,再经过数据通路得到最后部分积PP_{i0}.由于图3中的az_2bit为11时,需要分解成01和10进行分别计算.由此只要分析这两种情况,并分别根据他们处于分组

中的不同位置,计算需要移动的位数,最终得到标志位 C_shift.

根据图 4,可以计算出部分积 PP_{i0},其中 $B_0 - B_7$ 代表 8 位 B,标志位 C_shift[0] - C_shift[7]分别代表向左移动 0 - 7 位,C_shift 这 8 个标志位有且仅有一个为 1. 由此可以得到输出部分积的关系式,如:$P_0 = B_0$ & C_shift[0],$P_1 = (B_1$ & C_shift[0])^ (B_0 & C_shift[1]),依次类推,可以得到最后的 15 位部分积 PP_{i0}.

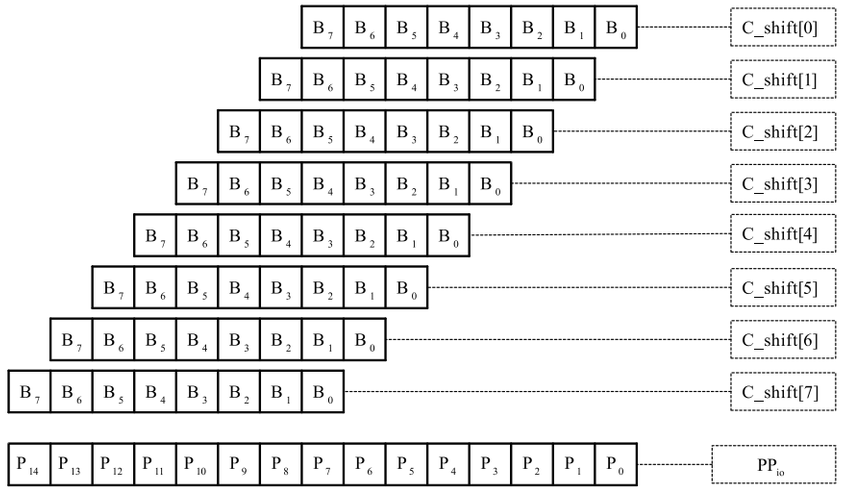


图4　组合移位算法原理图

3.3　异步 Booth 乘法器设计

图 5 的异步乘法器的的控制部分除了使用基于 Click 的微流水线之外,还需要选择器 Mux,以及两个计数器 $Count_0$ 和 $Count_1$. 功能模块包括:移位编码器 Shifter、4 - 2 压缩器 Compressor 和超前进位加法器 CLA (Carry Look-ahead Adder). $Count_0$ 控制移位编码器反复进行 2 次计算,当 a 的值为 00/01/10 第一次计算会得到部分积 ppi,而第二次计算结果 ppi 的值直接为 0,但当 a 为 11 时,移位编码器会进行 2 次不同的计算. 通过 $Count_1$ 控制 Mux,4 - 2 压缩器会反复进行 3 次计算. 前 2 次压缩是将 Shifter 电路 2 次计算的 8 个部分积 ppi,并分别存入 FF_7,FF_8 触发器中,第三次压缩将 FF_7,FF_8 中 4 个值重新进行压缩并放入 FF_9 中. 本文中改变传统的华莱士树压缩方法,此方法见文献[22],而是采用循环调用压缩模块完成数据压缩. 最后 FF_9 中的值传到 CLA 电路中进行计算,得到最后的乘积.

这种异步 Booth 乘法器的实现原理可以通过异步控制器的握手通讯来分析:

(1) $Click_0$:在 FF_1,FF_2 触发器中存有 a,b 两个值,当 $Click_0$ 信号到来,并产生一个 Fire0 信号. Fire0 触发 FF_1,FF_2 这两个值传给 Shifter 电路进行第一次计算,此时 $Count_0$ 为 0,计算得到的 4 个部分积 ppi 存储到 FF_3 - FF_6 这 4 个触发器中. 同时 $Click_0$ 将握手信号传到 $Click_1$ 中,并产生 Fire1 信号.

(2) $Click_1$:Fire1 信号触发 FF_3 - FF_6,将第一次计算得到的 4 个部分积 ppi 往下传输,同时握手信号传到 $Click_2$.

(3) $Click_2$:Fire2 信号触发 $Count_0$ 加 1,此时 Shifter 进行第二次计算,并将产生 4 个部分积 ppi 存储到 FF_3 - FF_6 中. 同时,Fire2 信号会触发 $Count_1$,使 Mux 把第一次计算得到的值传到 Compressor 中进行第一次压缩,将压缩得到的值存储到 FF_7 中.

(4) $Click_3$:Fire3 信号触发 Shifter 电路下面的 4 个触发器,将第二次计算结果继续往下传输. 同时,Fire3 触发 FF_7,将第一次的压缩值往下传输.

(5) $Click_4$:Fire4 信号触发 $Count_1$ 加 1,使 Mux 控制 Shifter 电路计算的第二次值传到 Compressor 中进行第二次压缩,并将压缩得到的值存储到 FF_8 中.

(6) $Click_5$:Fire5 信号触发 FF_8,使第二次压缩的值往下继续传输.

(7) $Click_6$:Fire6 信号触发 $Count_1$ 加 1,使 Mux 控制 FF_7 和 FF_8 的值传到 Compressor 进行第三压缩,并将压缩的值存储到 FF_9 中.

(8) $Click_7$:Fire7 的上升沿到来,将触发 FF_9,将第三次压缩的值传到 CLA 中进行计算,得到的输出值存到 FF_{10}触发器中. 当 Fire7 下降沿到来,将触发 FF_{10},输出最后的乘积.

4　FPGA 实现、仿真与验证

本文中采用 PlanAhead 平台进行 8 位异步乘法器的设计与仿真,硬件描述语言使用 Verilog-1995. PlanAhead 是 Xilinx 公司从 RTL 到比特流完整设计流程的工具,运用的 FPGA(Field-Programmable Gate Array)型号是 Xilinx 公司的 Spartan-6(xc6slx45tfgg484-3).

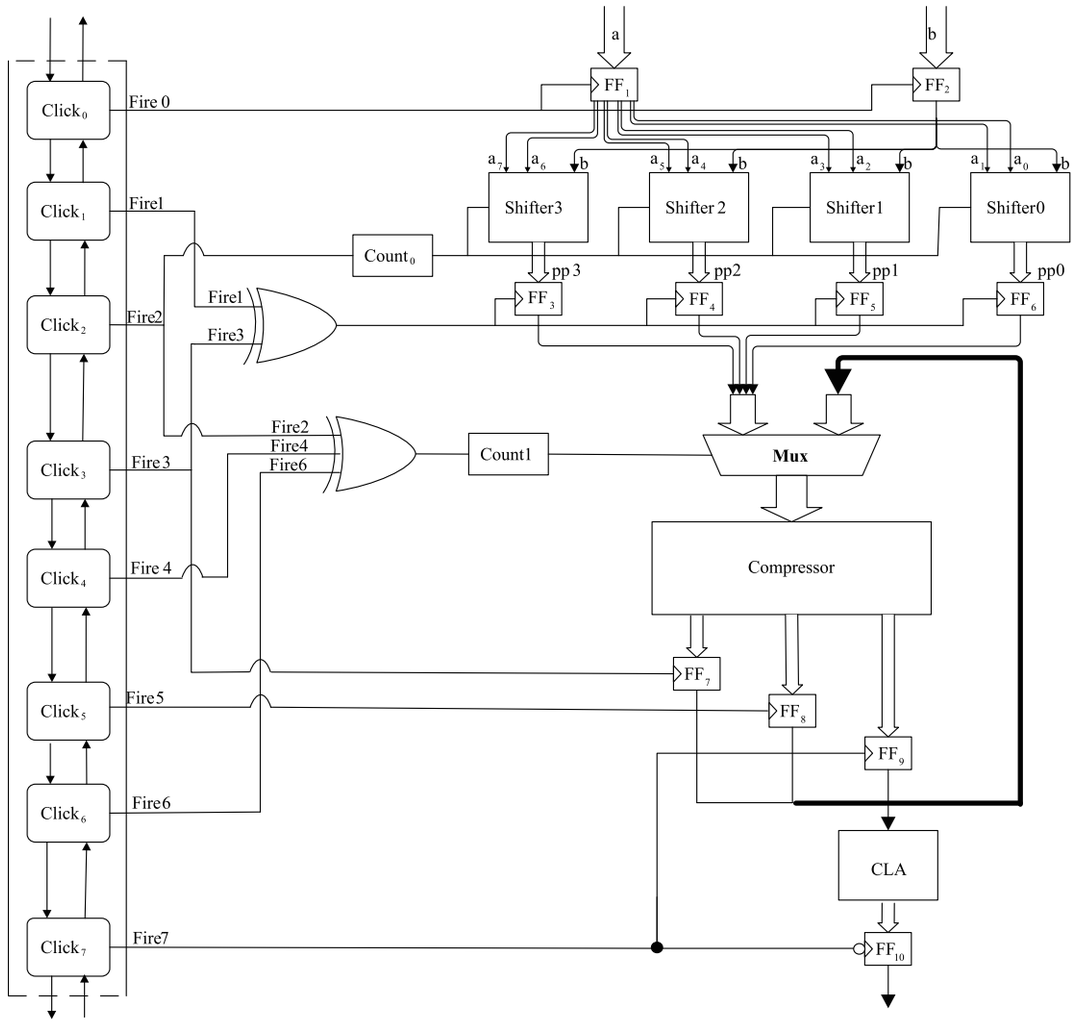


图5 8位异步Booth乘法器设计

图6是其中的wi_a_8bit为3,wi_b_8bit为45的一种仿真结果,当in_req变为高电平,异步控制器握手通讯开始,乘法器开始读取wi_a_8bit和wi_b_8bit两个乘数,使触发器工作并产生Fire0信号. 触发器输出信号传到$Click_1$,即$Click_1$/in_R_delayed信号升高,期间会使$Click_1$中的触发器工作并产生Fire1信号,触发器输出值会应答上级$Click_0$电路,即$Click_0$/out_A_delayed信号升高. 最后out_req变为高电平,一次乘法计算结束,得到乘积为135.

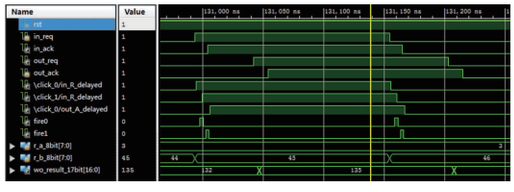


图6 仿真图

在下载到Spartan-6进行测试之前,采用了软硬件联合仿真的办法来验证此设计. 这种方法是通过C++遍历所有可能的乘数并生成输入文件,此文件数据在isim仿真时读入到异步乘法器中,isim仿真得到的乘积生成输出文件,此文件数据再次经由C++程序验证其正确性,流程如图7所示.

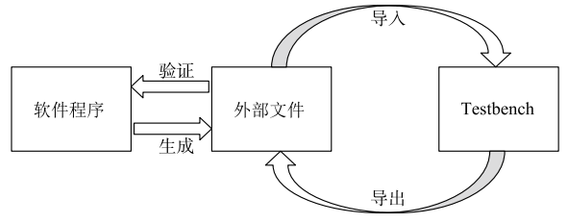


图7 验证流程

8位异步乘法器通过仿真与验证,保证了其功能正确性. 将此乘法器与数码管相连,将PlanAhead产生的比特流文件下载到Spartan-6(xc6slx45tfgg484-3)FPGA

开发板上,乘法器工作正常,见图 8.

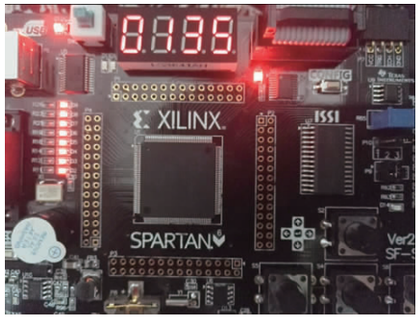


图8　FPGA验证

5　性能对比

为了对比异步设计的性能,将图 5 左部的握手电路替换为时钟控制的状态机[23],并与图 5 右部相连,得到了一种与上面所介绍的异步乘法器体系结构相同的同步实现. 使用了 PlanAhead 自带的综合及布局布线工具来实现了这两种 8 位异步/同步乘法器. 这里用 SyncBooth 表示同步设计的乘法器,AsyncBooth 表示论文设计的异步乘法器.

首先从两种乘法器占用的 FPGA 面积角度来比较,两种电路所占用的系统资源如表 2 所示,FPGA 的系统器件资源主要包括 LUT 和触发器,异步设计比同步设计的触发器个数少大约 10%,而异步设计的 LUT 数比同步设计的多 20%,所以对于 8 位乘法器而言,异步设计与同步设计所消耗的面积基本相同. 在能耗方面,通过 PlanAhead 中的 XPower Analyzer 进行比较两种设计的功耗,得到的能耗是相同的. 再仔细分析这两种设计,会发现,异步设计的控制部分为 CLICK 微流水线机制,而同步设计的控制为状态机. 在更大规模的控制电路中,微流水线机制的面积将线性增长,但状态机电路面积会指数增长. 所以,从面积消耗而言,异步设计在小规模系统中并不占优势,但在较大规模集成电路设计中优势明显.

再来分析时间因素方面二者的异同,采用相同体系结构的 SyncBooth 的状态机共有 10 个状态,所以完成一次 8 位乘法的计算需要 10 个时钟周期,Spartan-6 型号 FPGA 的系统时钟为 66ns,那么整个乘法计算需要 660ns 时间. 与之相对的异步设计 AsyncBooth 乘法器不由系统时钟驱动,仿真结果表明,AsynBooth 在 53.4ns 时间内即可完成一次乘法计算. 结果表明,采用本文体系结构的异步设计方法的效果远远优于同步设计方法. 通过仔细分析两种设计的计算时间,可以看出异步设计的计算时间小于 Spartan-6 的系统时钟周期,所以即使不采用本文提出的体系结构,而用组合逻辑电路实现 8 位乘法器的计算过程,其速度也会差于提出的异步设计方法.

表 2　系统消耗资源

Category	SyncBooth	AsyncBooth
Number of Slice Registers	169 out of 54,576	153 out of 54,576
Number of Slice LUTs	196 out of 27,288	246 out of 27,288
Time per Multiplication	660ns	53.4ns
Power(w)	0.38	0.38

6　结论

本论文提出了一种增加部分积压缩次数并将加(减)法后置的低耦合 Booth 算法,这种算法通过分隔模块的功能来提高计算效率. 这种算法非常适合异步控制,本文运用异步微流水线机制和组合功能模块完成移位,压缩和加法功能,设计模块化程度高,流程简单清晰. 与相同体系结构下的同步乘法器相比,本文提出的 8 位异步乘法器在能耗和面积大体不变的情况下,计算速度快了超过 12 倍. 采用本文中基于 Click 微流水线的异步设计方法在算数电路设计中是有效的.

相对于目前实用的乘法器而言,8 位异步乘法器处理能力有限,本文后期的工作已经开展基于此 Booth 算法思想的64 位异步乘法器研发,通过清晰划分模块功能以及模块局部复用的办法来提高计算效能. 此外对于大位数的乘法器而言,模块复用的设计方法必然构造在优化策略之上,异步控制下的优化方法也是亟待解决的一项工作.

致谢

感谢美国波特兰州立大学异步研究中心(ARC)的 Ivan Sutherland 教授、MarlyRoncken 教授以及和我们长期合作的博士后 Hoon Park 博士在异步控制器和时序分析领域的指导和合作,感谢兰州大学先进计算实验室对本项研究的支持和协助.

参考文献

[1] BOOTH A D. A signed binary multiplication technique [J]. Quarterly Journal of Mechanics & Applied Mathematics,1951,4(2):236 – 240.

[2] HoonPARK, AnpingHE, et al. Modular timing constraints for delay-insensitive systems[J]. Journal of Computer Science and Technology,2016,31(1):77 – 106.

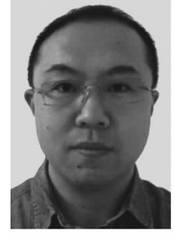
[3] PEETERS A,BEEST F T,et al. Click elements:an implementation style for data-driven compilation[A]. Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits & Systems[C]. USA: IEEE, 2010.3 – 14.

[4] Xilinx. The FPGA Specification[DB/OL]. http://www.xilinx.com,2015 – 05 – 30.

[5] 梁峰,邵志标,梁晋. Radix-16 Booth 流水线乘法器的计

[J]. 西安交通大学学报,2006,40(10):1111 – 1114.
LIANG Feng, SHAO Zhibiao, LIANG Jin. Design of Radix-16 booth pipeline multiplier[J]. Journal of Xi'an Jiaotong University,2006,40(10):1111 – 1114. (in Chinese)
[6] 焦继业,穆荣,郝跃. 快速设计高性能有符号乘法器电路的编程语言研究[J]. 电子学报,2013,41(11):2256 – 2261.
JIAO Jiye, MU Rong, HAO Yue. A programming language for rapid design of high performance signed multiplier circuits[J]. Acta Electronica Sinica,2013,41(11):2256 – 2261. (in Chinese)
[7] 陈弘毅,岳震五,顾群. 一种位级流水线乘法器的设计[J]. 电子学报,1992,20(5):39 – 46.
CHEN Hongyi, YUE Zhenwu, GU Qun. A design on the bit-level pipelined multiplier[J]. Acta Electronica Sinica, 1992,20(5):39 – 46. (in Chinese)
[8] ZHANG S, WANG N, ZHOU R. Power analysis and optimization methods of the pipelined array multiplier[A]. Proceedings of the International Conference on ASIC[C]. USA:ASIC,2003, Vol2. 1231 – 1234.
[9] Ivan E SUTHERLAND, Jo EBERGEN. Computers without clocks[J]. Scientific American,2002,287(2):62 – 69.
[10] Ivan E SUTHERLAND. Micropipelines[J]. Communications of the ACM,1989,32(6):720 – 738.
[11] PARK C H, CHOI B S, et al. Asynchronous array multiplier with an asymmetric parallel array structure[A]. Proceedings of Conference on Advanced Research in VLSI [C]. USA:IEEE Computer Society,2001. 202 – 212.
[12] SU Bo, WANG Zhiying, et al. Reducing power consumption of floating-point multiplier via asynchronous technique[A]. Proceedings of Fourth International Conference on Computational and Information Sciences[C]. USA: IEEE,2012. 1360 – 1363.
[13] YANG Y, YANG Y, et al. A high-speed asynchronous array multiplier based on multi-threshold semi-static NULL convention logic pipeline[A]. Proceedings of the IEEE 9th International Conference on ASIC[C]. USA: IEEE, 2011. 633 – 636.
[14] LIU Y, FURBER S. The design of a low power asynchronous multiplier[A]. Proceedings of the International Symposium on Low Power Electronics and Design[C]. USA:IEEE,2004. 301 – 306.
[15] KEARNEY D, BERGMANN N W. Bundled data asynchronous multipliers with data dependent computation times[A]. Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems[C]. USA:IEEE Computer Society,1997. 186.
[16] CHEN H, GAO J, SU S, et al. A visual-aided wireless monitoring system design for total hip replacement surgery [J]. IEEE Transactions on Biomedical Circuits & Systems,2015,9(2):227 – 236.
[17] SPARSØ J. Asynchronous circuit design-A tutorial[J]. Microlab. ti. bfh. ch,2006,623:1 – 49.
[18] MULLER D E, BARTKY W S. A theory of asynchronous circuits[J]. Radical Philosophy,2010,14(5):204 – 243.
[19] Ivan SUTHERLAND, Scott FAIRBANKS. GasP: A minimal FIFO control[A]. Proceedings of Seventh International Symposium on Asynchronous Circuits and Systems [C]. Salt Lake City, Utah,2001. 46 – 53.
[20] BARDSLEY A, EDWARDS D A. The balsa asynchronous circuit synthesis system[A]. Forum on Design Languages (FDL)[C]. USA:IEEE,2000. 1 – 8.
[21] RONCKEN M, GILLAS M, et al. Naturalized communication and testing[A]. Asynchronous Circuits and Systems [C]. USA:IEEE,2015. 77 – 84.
[22] 徐加全,候朝焕,张骥. 18 × 18 并行流水乘法器芯片设计[J]. 电子学报,1995,23(2):82 – 84.
XU Jiaquan, HOU Chaohuan, ZHANG Ji. 18 × 18 parallel and pipeline multiplier chip design[J]. Acta Electronica Sinica,1995,23(2):82 – 84. (in Chinese)
[23] OLIVEIRA D L, STRUM M, SATO S S. Burst-mode asynchronous controllers on FPGA[J]. International Journal of Reconfigurable Computing,2008(1):1687 – 7195.

作者简介



何安平 男,兰州大学信息科学与工程学院,获博士学位. 研究方向为集成电路设计与形式化分析验证方法,包括异步芯片设计、异步电路系统相对时序分析、集成电路系统形式化分析验证和工具研发.
E-mail:heap@ lzu. edu. cn

刘晓庆 男,1991 年生于山东青岛. 兰州大学信息科学与工程学院研究生,研究方向为异步电路设计等.
E-mail:liuxq2015@ lzu. edu. cn

陈 虹 女,清华大学电子工程系,获工学博士学位. 研究方向极低功耗电路与系统,包括压电陶瓷供电、亚阈值电路设计、异步电路设计及基于多传感器的数据融合技术及图像识别技术等.
E-mail:hongchen@ tsinghua. edu. cn