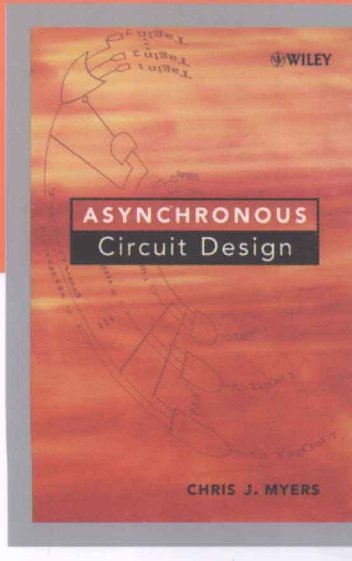
信息技术和电气工程学科国际知名教材中译本系列

WILEY

Asynchronous Circuit Design

异步电路设计



Chris J. Myers 著
李鹏 译

清华大学出版社

信息技术和电气工程学科国际知名教材中译本系列

WILEY

Asynchronous Circuit Design

异步电路设计

Chris J. Myers 著

李鹏 译

清华大学出版社

北京

图书在版编目(CIP)数据

异步电路设计/(美)迈尔斯(Myers, C. J.)著；李鹏译. --北京：清华大学出版社，2013
书名原文：Asynchronous Circuit Design
信息技术和电气工程学科国际知名教材中译本系列
ISBN 978-7-302-31915-3

Ⅰ. ①异…　Ⅱ. ①迈…　②李…　Ⅲ. ①电路设计－教材　Ⅳ. ①TM02

中国版本图书馆 CIP 数据核字(2013)第 052968 号

责任编辑：梁　颖　薛　阳
封面设计：傅瑞学
责任校对：李建庄
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 装 者：三河市李旗庄少明印装厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.75　　**字　　数**：513 千字
版　　次：2013 年 9 月第 1 版　　**印　　次**：2013 年 9 月第 1 次印刷
印　　数：1～3000
定　　价：49.00 元

产品编号：049798-01

译 者 序

众所周知，目前的同步集成电路随着集成度的提高，在不远的将来，将遭遇两个难以逾越的极限：极高的发热量和极大的设计复杂性。顶尖集成电路的发热量已接近百瓦。而其中最大的发热来源就是芯片上庞大的时钟树。时钟树的功耗已占总功耗的40%之多。另一方面，线宽越来越细，而芯片尺寸却越来越大，电信号从芯片的一端到达另一端要用好几个时钟周期。整个芯片保持同步已绝无可能。与其扬汤止沸，不如釜底抽薪。突破这两个极限的唯一办法就是取消用于全局同步的庞大时钟树。可见异步集成电路是发展的必然趋势。换句话说，很有可能未来的顶尖集成电路毫无例外地或多或少都将采用一些异步集成电路技术！

由于原著作者在原著序中所列举的六大原因，目前异步集成电路在国外的发展自20世纪90年代初以来已呈现加速趋势，已成为集成电路技术前沿的热点。Intel、IBM、ARM、Philip都在进行商用研究，其中ARM和Philip公司已推出面向低功耗和低电磁辐射的嵌入式异步芯片。与国外如火如荼的发展相反，异步集成电路技术在国内却未引起足够的重视，只在研究生层次开展了极少数的研究，对本科生的教育更是空白。市场上介绍异步集成电路设计的中文书迄今仅只有译著和编著各一本。国内数字电路课本中关于异步电路方面，有的完全没有，有的只有一章，内容仅限于异步计数器和异步单输入变化自动机等初级内容，这仅仅相当于国外20世纪五六十年代的水平。亡羊补牢，犹未为晚。本书推出的目的就是更快更好地填补国内这方面的空白，将异步集成电路这一具有战略意义的新事物介绍给广大的国内读者。

近几年，我国集成电路设计业有了一定的发展。若干中国芯陆续涌现，但都是同步集成电路。这种在同步设计技术方面的追赶，无疑极有必要。但由于差距较大，在可以预见的将来，达到国际顶尖水平困难较大。但在异步设计技术方面国外也是摸着石头过河，我们虽有落后，但落后的程度远小于同步技术方面。我们若能抓住机遇，迎头赶上，缩小差距，在异步电路方面达到较高水平是完全有可能的。

本书是尤他大学Myers教授所写的异步集成电路教材。他领导着全球唯一的专门研究定时电路的实验室，是此方面的权威。本书覆盖了异步电路的各个方面，涉及信道模型、通信协议、异步自动机、延迟不敏感电路、定时电路，以及异步验证方面的内容。其中定时电路是所有异步电路中速度最快，最节省器件的一类，也是最难设计的。定时电路较少有人系统研究，一些信息仅零散出现于论文中。本书中第7章专门讲解了定时电路，所以此部分内容极具价值。Myers领导的定时电路实验室应用定时电路设计技术开发的RAPPID电路达到了很高的性能。另外书中异步集成电路的验证方面的内容也是少有的。读者可以下载此研究组开发的设计工具自行尝试异步设计。

愿本书对我国集成电路工业在新世纪的发展能有所帮助。

译者

2013年7月

序　言

一个重大的科学创新很少是通过逐步赢得反对者的支持和将反对者转化而成功的：索罗变成保罗是很少发生的。所发生的是其反对者们逐渐过世而成长起来的一代从一开始就熟悉这一思想。

——马克思·普朗克

我必须支配时钟，而不是时钟支配我。

——Golda Meir

All pain disappears, it's the nature of my circuitry.

——nine inch nails

1969年，Stephen Unger出版了他的关于异步电路设计的经典教科书。该书对定时的异步设计方法给出了全面的看法。从那以后的30年，出现了众多的技术出版物，甚至一些书[37,57,120,203,224,267,363,393]，但却未出现另一本教科书。本书试图填补这一空白，以一种易懂的方式为仅有少许数字逻辑设计背景知识的学生提供对异步电路设计的最新看法。

异步电路是不使用全局时钟完成同步的电路。异步电路有几个优于同步电路的地方。

(1) 时钟歪斜问题的消除。当系统变得更大时，越来越多的设计努力是必须的，以确保时钟信号到达芯片的不同部分时有最小歪斜。在异步电路中，同步信号的歪斜能被耐受。

(2) 平均情况的性能。同步系统中，性能由最坏情况摆布。时钟周期必须被设置得足够长以适应最慢的操作，即使操作的平均延迟通常短得多。在异步电路中，电路速度被允许动态地改变，所以性能由平均情况延迟支配。

(3) 对制程和环境变化的适应性。VLSI电路的延迟会因不同制程、电源电压和工作温度而变化显著。同步设计在一些允许的变化下设置其时钟频率以使其工作正常。由于其自适应特性，异步电路在所有变化下工作正常，只需加快或放慢下来。

(4) 部件的模块化和重用。在异步系统中，连接各部件时不会遇到在同步系统中那些与同步时钟相关的困难。

(5) 更低的系统功耗需求。异步电路因无须额外的时钟驱动器和为限制时钟歪斜的缓冲器而降低了同步所需的功耗。异步电路还自动对未使用部件断电。最后，异步电路不因无效翻转而浪费功耗。

(6) 降低噪声。在同步设计中，全部活动被锁定于非常精确的频率。结果是几乎全部能量集中于时钟频率及其谐波处的极窄频带内。因此，在这些频率处有大量的电噪声。异步电路中的活动是无关联的，导致更分布的噪声谱和更低的噪声峰值。

尽管有这些全部潜在的优势，迄今为止仅见到异步设计被有限地利用。虽然这种情形的原因很多，或许最严重的是缺乏有异步设计经验的设计者。本教科书是通过提供以下手段试图解决这一问题的直接尝试，即创建面向研究生甚至本科生教授现代异步设计方法的

课程。我已在一个本科生和研究生合班的课上使用过它。此课程和未来课程使用的讲义和其他材料将可从我们的网站获得：http://www.async.elen.utah.edu/book/。本书也可供想了解现代异步设计方法的工程师自学。每章包括许多供学生考察他或她的新技能的习题。

异步设计的历史相当久远。异步设计方法可追溯至20世纪50年代和尤为重要的两个人：霍夫曼(Huffman)和马勒(Muller)。每种异步设计方法学都可寻根至此二人之一。霍夫曼发展了今天称为基本模式(fundamental-mode)电路的设计方法学[170]。马勒发展了速度无关(speed-independent)电路的理论基础[279]。Unger是“霍夫曼学派”的成员，所以他的教科书主要集中于基本模式电路设计而只有关于马勒电路的简短论述。虽然我是“马勒学派”的一名学生，但在本书内我们将两种设计方法都作了介绍，希望两派成员增进彼此了解，或许甚至可以认识到两者差别并不是那么大。

从早期的日子起，异步电路被用于很多有趣的应用。20世纪50年代和20世纪60年代在伊利诺斯大学，马勒及其同事在ILLIAC和ILLIAC Ⅱ计算机的设计中使用了速度无关电路[46]。在早期的日子里，异步设计也被用于MU-5和Atlas大型计算机。20世纪70年代在圣路易斯的华盛顿大学，开发了异步宏模块[87]。这些模块能插在一起制成许多专用的计算引擎。还是20世纪70年代，异步技术被用于犹他大学的首台数据流计算机[102,103]和益世公司的首个商业图形系统中。

由于以上所列举的优点，对异步设计的兴趣已经高涨。最近有几个成功的设计项目。1989年，加州理工的研究人员设计了首枚全异步微处理器[251,257,258]。从那时起，许多其他研究人员已经制作了一些日渐复杂的异步微处理器[10,13,76,134,135,138,191,259,288,291,324,379,406]。在商业方面，异步电路已有一些新近的成功例子。Myranet在其路由器设计里用异步电路使流水线同步(pipeline synchronization)[348]。飞利浦已完成了许多面向低功耗的异步设计[38,136,192,193]。也许来自此小组的最著名成就是一个异步80C51微控制器，现用于一个由飞利浦销售的全异步寻呼机里。最后，Intel的RAPPID项目展示了一个x86指令集的全异步指令长度译码器与现存的同步设计比速度快3倍，能效上改进两倍[141~144,330,367]。

在Unger的课本时代，每年或许只有少量异步设计方面的出版物。如图0.1所示，此出版速率一直持续到大约1985年，这时出现了对异步电路设计兴趣的复苏[309]。从1985年起，出版速率已增长到每年超过100种技术出版物。因此，虽然Unger在此领域做了出色的综述性工作，但其工作还是远远不够的。每章末原始文献一节把感兴趣的读者引至一个巨大的书目(超过400项)供他们进一步探究。尽管已尝试介绍了那些正发展和使用的主要设计方法学，但不可能引用异步设计方面的每一篇已出版的论文，因为异步书目[309]内的文献数量现已超过1400篇。感兴趣的读者应查阅此书目和新近的异步电路及系统专题讨论会的会议论文集[14~20]。

本书组织如下：第1章通过一个小例子介绍异步设计问题，说明使用各种时序模型时的差别。第2章介绍异步通信的概念并且描述一个用VHDL说明异步设计的方法学。第3章讨论各种异步协议。第4章介绍了异步设计中使用的图形表示。第5章讨论霍夫曼电路。第6章描述马勒电路。第7章开发了时序分析和优化技术，它能导致电路质量的显著

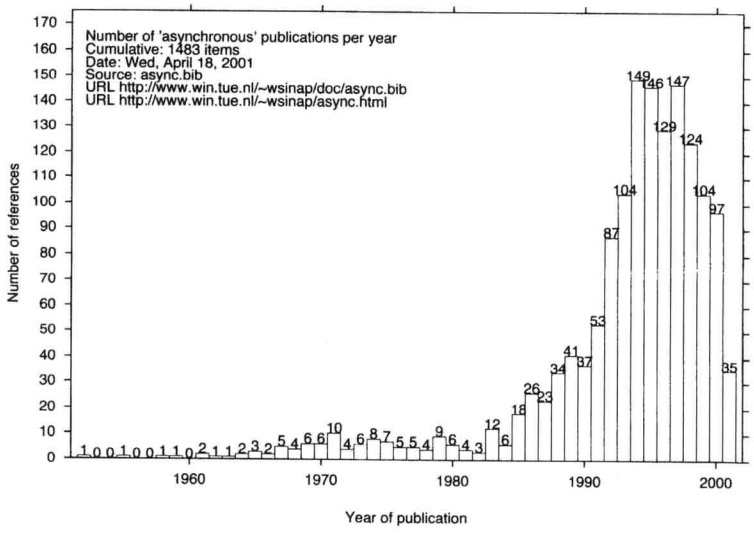


图 0.1　每年异步出版物的数量

改进。第 8 章介绍异步电路的分析和验证方法。最后，第 9 章对异步应用中的问题作了一个简短的讨论。

克里斯·J. 迈尔斯

盐湖城，犹他州

鸣谢

我感激加州理工的 Alain Martin 和 Chuck Seitz，将还是大学生的我引导到异步设计领域。我还要感谢斯坦福大学的 Teresa Meng 和 David Dill，我的研究生导师，是他们教了我审视异步设计的另类方法。我以前的办公室同事，Peter Beerel(USC)，通过多年来许多热烈的讨论，使我获益良多。

我要感谢 Erik Brunvand(Utah)，Steve Nowick(Columbia)，Peter Beerel(USC)，Wendy Belluomini(IBM)，Ganesh Gopalakrishnan(Utah)，Ken Stevens(Intel)，Charles Dike(Intel)，Jim Frenzel(U. of Idaho)，Steven Unger(Columbia)，Dong-Ik Lee(KJIST)和 Tomohiro Yoneda(Titech)，他们对早期版本手稿的意见和建议。我还要感谢来自我的研究生 Brandon Bachman，Jie Dai，Hans Jacobson，Kip Killpack，Chris Krieger，Scott Little，Eric Mercer，Curt Nelson，Eric Peskin，Robert Thacker 和 Hao Zheng 的意见和想法。我要感谢在 2000 年春季学期修我的异步电路设计课的学生，他们忍受了我的还不成熟的课本。我感谢画了书里许多图的 Sanjin Piragic。很多其他插图由 Jordi Cortadella(UPC)用 draw_astg，Eleftherios Koutsofios 和 StephenNorth(AT&T)用 dot 画出。

我特别想要感谢我的家庭，当我写此书时 Ching 和 John 表现出的耐心。如果没有他们爱和支持，此书是不可能完成的。

C. J. M

目　录

1 绪　论

酒是瓶装的诗。

——Robert Louis Stevenson

葡萄酒给人胆量并使男人更易于充满激情。

——Ovid

我用葡萄干造酒，所以我不必等它变陈。

——Steven Wright

本章用一个简单例子非正式地介绍书中涉及的很多概念和设计方法，本章中每个主题的更多细节将在随后的各章里更正式地论述。

1.1 问题说明

在犹他州南部的一个小镇，有一座小葡萄酒酿造厂，附近有一家售酒铺。想象一下作为仍实行禁酒令的县里的一个小镇，只有一位买酒顾客。酒铺有唯一的一个只能放一瓶酒的小架子。在每个小时的整点，店主从酒厂收到一瓶刚制成的酒并放在架子上。在每小时的半点，那位顾客来买走酒，为下瓶酒腾出地方。现在，顾客已认识到准时是非常重要的。若他早到，他会恼怒地发现只有一个空架子。若他迟到，他会发现店主为给新酒让地喝了那瓶酒。最令人沮丧的经验是当他到达的同时正赶上店主正把酒放在架子上。他因兴奋过度而与店主撞在一起，导致酒瓶掉到地板上，摔个粉碎，结果没人喝得到那瓶美酒。

这种同步购酒法一直持续了一段时间，当事人的合作很愉快。然而(在 20 世纪 80 年代中期)有一天，电话服务延伸到这个小镇。这一光辉的发明着实让小镇激动起来。顾客有了一个完美的主意。他知道只有他有了一个更快的买酒方法，酒厂才能更快地运转。因此，他建议店主，“哎，酒到时你何不给我个电话?”这样他就能免得太早出现，使他脆弱的性情感受挫折。不久之后的下一个小时，他接到一个电话让他取酒。他如此激动以致跑到店里。离开时他向店主建议，“哎，你何不给酒厂的家伙去个电话，告诉他们你有地儿放另一瓶酒了?”店主的确如此这般，您也许没想到，买酒顾客仅在 10min 后就又接到另一个电话说是新酒到了。在随后的一个小时里此过程一直持续。有时 10min 后接到电话，而有时需要长达 20min。有一次甚至离开店铺仅仅 5min 他就又接到一个电话(幸运的是，他住得很近)。在这个小时结束时，他意识到在仅仅一个小时内他已喝了 5 瓶酒!

此时，他感觉有点头昏眼花，于是决定小睡片刻。一小时后，他骤然惊醒。他意识到电话铃声大作。“哦，天哪，我把酒忘了!”他狂奔而至，料想会发现店主已经喝好几瓶他的酒了。但让他惊愕的是，他只看见一瓶酒在架子上而周围并无空瓶。他问店主，“他们为什么停止送酒了?”店主道：“哦，我没打电话，他们决定最好暂停送酒直到我的架子上有个空地儿。”

从那天以后，此异步购酒法成为各方接受的生意经。酒厂因售出了更多的酒（平均而言）而心满意足。店主妻子也高兴，因为店主不必饮酒过度了。顾客更是极其愉快，因为他现在能更快地得到酒，并且每当他感到有点饮酒过度时，他能从容地休息一下，因为他知道他将不会错过任何一瓶酒。

1.2 通信通道

有一天一位 VLSI 工程师停在小镇的这家酒铺旁，并与店主谈起他的小生意。生意不错，但他妻子总是吵着要他兑现多年的承诺，去夏威夷的毛伊岛度假。他真的不知该怎么做，因为当他不在时他不信任任何人打理他的店铺。此外，他有些担心若他不够小心谨慎的话，酒厂和顾客可能意识到他们实在不需要他就能彼此直接进行交易。对此他真的承受不起。

VLSI 工程师聚精会神地听着，当他说完后工程师宣布："我能解决你的所有问题。让我为你设计一个电路！"最初，当店主得知此电路将用电（当地人还没完全接受的新魔力）驱动时他十分怀疑。工程师说："实际上，它真的很简单。"工程师在他的餐巾上画了一个小草图（图 1.1）。此图显示两个必须保持同步的通信通道（channel）。一个在酒厂和店铺之间，另一个在店铺和顾客之间。当酒厂通过 WineyShop 信道（communication channel）从店铺收到一个请求时，酒厂送出一瓶酒。这能被说明如下：

```
Winery:process
begin
  send(WineryShop,bottle);
end process;
```

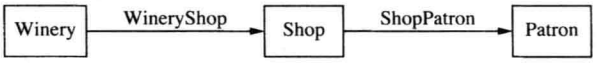


图 1.1 通信的通道

注意此 Process 语句暗示 Winery 永远循环地发送酒。买酒顾客，当接收到店铺经 ShopPatron 信道发来的请求时，他来到店里收下酒，这能被说明为：

```
Patron:process
begin
  receive(ShopPatron,bag);
end process;
```

现在，店主作为一个中间人所做的（除了标上价格以外）就是为酒提供一个缓冲区以使酒厂开始准备其下一瓶酒。这能被说明如下：

```
Shop:process
begin
  receive(WineryShop,shelf);
  send(ShopPatron,shelf);
end process;
```

这 3 件事共同构成了一个规范（specification）。前两个过程描述了店主要与之打交道

的人(即他的环境),最后的过程描述了店铺的行为。

1.3 通信协议

在获得了信道级的规范之后,随后需要确定实现通信的通信协议(communication protocol)。例如,店主致电酒厂"请求"(request)一瓶新酒。过了一段时间之后,新酒到达,"确认"(acknowledging)了此请求。一旦酒瓶在架子上安放好,店主就能致电顾客"请求"他来买那瓶酒。过了一段时间之后,顾客来到店里并买走酒,"确认"了店主先前的请求。这能被描述如下。

```
Shop:process
begin
  req_wine;                -- call winery
  ack_wine;                -- wine arrives
  req_patron;              -- call patron
  ack_patron;              -- patron buys wine
end process;
```

为构建一个 VLSI 电路,必须把信号线一一分配给上述 4 个操作。其中两根线连接到一个能产生适当电话呼叫的装置。这些称为输出。另一根电线将来自一枚按钮,当送酒男孩送酒时,按下此按钮。最后那根线来自由顾客按压的一枚按钮。这两个信号是输入。因为此电路是数字电路,这些线只能处于两种状态之一:'0'(低电压状态)或'1'(高电压状态)。让我们假设上述动作用相应信号线上的信号变为'1'来表示。这能被描述如下。

```
Shop:process
begin
  assign(req_wine,'1');      -- call winery
  guard(ack_wine,'1');       -- wine arrives
  assign(req_patron,'1');    -- call patron
  guard(ack_patron,'1');     -- patron buys wine
end process;
```

上面使用的 assign 函数置某信号为某值。guard 函数等待直到某信号达到了某给定值。上面给出的规范里有一个问题,当第二瓶酒来时 req_wine 已经是'1'了。因此,需要在循环绕回来之前复位这些信号。

```
Shop_2Phase: process
begin
  assign(req_wine,'1');      -- call winery
  guard(ack_wine,'1');       -- wine arrives
  assign(req_patron,'1');    -- call patron
  guard(ack_patron,'1');     -- patron buys wine
  assign(req_wine,'0');      -- call winery
  guard(ack_wine,'0');       -- wine arrives
  assign(req_patron,'0');    -- call patron
  guard(ack_patron,'0');     -- patron buys wine
end process;
```

当 req_wine 从'0'变成'1'时，打出一个电话，而当它从'1'又变回'0'时，打出另一个电话。我们称为转换信号(transition signaling)。由于显然的原因，又称为两相(two-phase)或两下一循环(two-cycle)信号。显示两相店铺行为的波形如图 1.2(a)所示。另一种方案如下。

```
Shop_4Phase:process
begin
  assign(req_wine,'1');      -- call winery
  guard(ack_wine,'1');       -- wine arrives
  assign(req_wine,'0');      -- reset req_wine
  guard(ack_wine,'0');       -- ack_wine resets
  assign(req_patron,'1');    -- call patron
  guard(ack_patron,'1');     -- patron buys wine
  assign(req_patron,'0');    -- reset req_patron
  guard(ack_patron,'0');     -- ack_patron resets
end process;
```

这种协议被称为电平信号(level signaling)，因为当请求信号为'1'时发出一个电话呼叫。它也被称为四相(four-phase)或四下一循环(four-cycle)信号，因它需要 4 个信号转换才能完成。显示四相店铺行为的波形如图 1.2(b)所示。尽管这种协议也许看起来更复杂一点，因为它需要多一倍的信号线转换，但它经常导致更简单的电路。

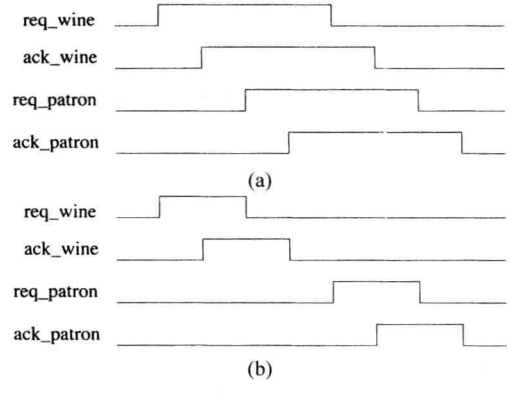


图 1.2 (a) 两相店铺的波形；(b) 四相店铺的波形

还有其他更多的选择。在原始协议里，店铺给酒厂和顾客打电话。换句话说，店铺在两个通信中是主动(active)方。酒厂和顾客是被动(passive)方。他们只是等待被告知何时行动。另一种选择也许是让酒厂作为主动方，当一瓶酒准备好时致电店铺，如下所示。

```
Shop_PA:process
begin
  guard(req_wine,'1');       -- winery calls
  assign(ack_wine,'1');      -- wine is received
  guard(req_wine,'0');       -- req_wine resets
  assign(ack_wine,'0');      -- reset ack_wine
  assign(req_patron,'1');    -- call patron
  guard(ack_patron,'1');     -- patron buys wine
```

```
  assign(req_patron,'0');     -- reset req_patron
  guard(ack_patron,'0');      -- ack_patron resets
end process;
```

类似地，顾客也可作为主动方，当他喝完其最后一瓶酒时再要一瓶。当然，在此情况下，店主需要安装第二根电话线。

```
Shop_PP:process
begin
  guard(req_wine,'1');        -- winery calls
  assign(ack_wine,'1');       -- wine is received
  guard(req_wine,'0');        -- req_wine resets
  assign(ack_wine,'0');       -- reset ack_wine
  guard(req_patron,'1');      -- patron calls
  assign(ack_patron,'1');     -- sells wine
  guard(req_patron,'0');      -- req_patron resets
  assign(ack_patron,'0');     -- reset ack_patron
end process;
```

不幸的是，照现在的样子所有这些规范都不能转换成电路。让我们回到最初的四相协议(即标有 Shop_4Phase 的协议)。开始，全部信号线都被置'0'并假设此电路致电酒厂要酒。在酒到达之后，信号 req_wine 和 ack_wine 被复位，信号线的状态又全为'0'。问题是在此情况下电路必须呼叫顾客。换句话说，当全部电线信号置'0'时，电路处在不确定状态。在此时到底应该呼叫酒厂还是顾客？我们需要确定某种方式来澄清这一点。再考虑一下最初的四相协议，可以通过调序(reshuffling)信号线的变化顺序来实现。尽管在致电顾客前酒到达是重要的，但握手信号线复位的精确时刻却不那么重要。如下所示重新安排的协议使电路总能知道该干什么。此外，急切的顾客可更快地接到电话。另外，它导致如图 1.3 所示非常简单的电路。

```
Shop_AA_reshuffled:process
begin
  assign(req_wine,'1');       -- call winery
  guard(ack_wine,'1');        -- wine arrives
  assign(req_patron,'1');     -- call patron
  guard(ack_patron,'1');      -- patron buys wine
  assign(req_wine,'0');       -- reset req_wine
  guard(ack_wine,'0');        -- ack_wine resets
  assign(req_patron,'0');     -- reset req_patron
  guard(ack_patron,'0');      -- ack_patron resets
end process;
```

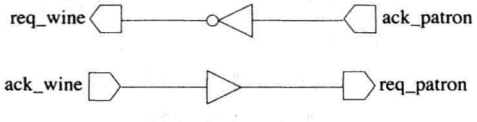


图 1.3 主动/主动店铺的电路

另外，可以将协议调序，让酒铺被动地等酒厂打来电话但仍主动地致电顾客，如下所示。得出的电路如图 1.4 所示。中间写有 C 的门叫马勒 C 单元(Muller C-element)。【译者注：