

高速数字系统中的时序分析与设计

卓沛, 严国萍

(华中科技大学 电子与信息工程系, 湖北 武汉 430074)

摘要:随着数字系统的工作频率的不断提高,时钟周期逐渐变小,而系统时序却越来越复杂。如何保证系统的工作时序正常,要涉及到比如保证信号完整性(Signal Integrity)、设计良好的电源分配系统(Power Distribute System)以及时序分析等诸多方面,因此成为了一项具有挑战性的工作。文中从实际设计出发,结合实际工作经验,讨论了在133MHz总线工作频率下,如何控制系统工作时序。

关键词:高速数字系统;最大/最小飞行时间;建立/保持时间;仿真

中图分类号: TN79; TP302

文献标识码: A

文章编号: 1673-629X(2007)07-0171-04

Timing Analysis and Design in High-Speed Digital System

ZHUO Pei, YAN Guo-ping

(Dept. of Electronics and Info. Eng., Huazhong University of Science & Technology, Wuhan 430074, China)

Abstract: Along with the increase of digital system working frequency, clock period gets shorter, timing of the system becomes more complex. How to ensure system timing correct involves signal integrity, power distribute system and timing analysis, that makes it a challenging job. In this paper, issues have been discussed about how to analyze and design system timing concerning 133MHz bus frequency based on practical design experience.

Key words: high-speed digital system; max/min flight time; setup/hold time; simulation

0 引言

当前数字电路的发展趋势是系统越来越复杂,工作频率越来越高,一方面使得从电气性能的角度看,封装和互连对于信号不再是畅通和透明的^[1],而是具有分布参数的传输线,这使得每一根互连线都得经过精心设计才能保证信号完整性(Signal Integrity, SI)。另一方面,随着时钟频率的提高,留给数据传输的有效读写窗口越来越小,SDRAM总线频率在百兆赫兹以上,DDR等总线速率更高,读写周期都在10ns以下,要想在这么短的时间限制里,让数据信号从驱动端完整地传送到接收端,就必须进行精确的时序计算和分析。

保证信号完整性是保证系统工作稳定的基础。由于工作频率的提高和上升沿的缩短,使得比如反射、过/欠冲、振铃、串扰等问题加剧,可能造成逻辑的误判甚至损坏逻辑门。解决这些问题的方法一是控制走线长度在电长度之内,从而使互连线对于信号传输是透明的;二是用控制走线阻抗和端接策略使得源端和负载

阻抗匹配,从而减小反射来保证信号完整性。如何保证信号完整性不是文中讨论的重点,在此不加叙述。

时序控制建立在保证信号完整性的基础之上,良好的信号质量是确保稳定的时序的关键,由于反射、串扰等造成的信号质量问题很可能带来时序的偏移和紊乱,另外如果电源分配系统(Power Distribute System)设计的不合理,也会使时钟源产生抖动(Clock Jitter)和倾斜(Clock Skew),从而影响系统时序。

笔者设计了一个网络视频监控系统,在文中主要讲述了针对该系统,在设计时利用CADENCE公司的板级设计工具ALLEGRO对关键网络进行仿真,在保证高速总线的信号完整性的基础上进行时序的分析计算和进行电气约束,从而实现对系统工作时序的控制。

1 系统简介

该网络视频监控系统主要完成的功能是实时地采集视频信号并进行压缩,再通过以太网发送出去,从而实现网络视频监控的目的。由于压缩算法复杂,本系统采用了ASM/DSP双核结构^[2],即采用两个CPU协同工作,一个是三星公司的S3C2410,另一个是德州仪器公司的TM320DM642。TM320DM642是一款高性

收稿日期:2006-09-21

作者简介:卓沛(1982-),男,四川人,研究方向为嵌入式系统设计;严国萍,教授,博士生导师,研究方向为视音频信号的传输与处理、信号检测与控制、电子电路与系统。

能的 DSP,工作频率在 600MHz,主要完成对视频信号的采集和实时的压缩,但 TM320DM642 的控制功能很弱,只能处理简单的网络请求,故再添加了一块 S3C2410 做主控器。S3C2410 是一款 ARM9 结构的处理器,工作频率 200MHz,控制功能很强,运算功能相对较弱,S3C2410 主要运行 LINUX,控制系统各部分资源,处理网络请求,并从 TM320DM642 处获得压缩好的视频包发送给请求端。整个系统框图如图 1 所示。

整个系统的关键网络在于两个 CPU 同各自的 SDRAM 之间的数据总线:S3C2410 的 SDRAM 数据总线工作频率 100MHz,读写周期 10ns,总线宽度 32 位。TM320DM642 和 SDRAM 数据总线工作频率 133MHz,读写周期 7.5ns,总线宽度 64 位。其他的网络比如同 FLASH 间的接口虽然也有时序要求,但由于读写周期较长(约 150ns),故要求不严格,分析较简单,此处就不再阐述。这里重点介绍如何控制 TM320DM642 和 SDRAM 数据总线的工作时序。

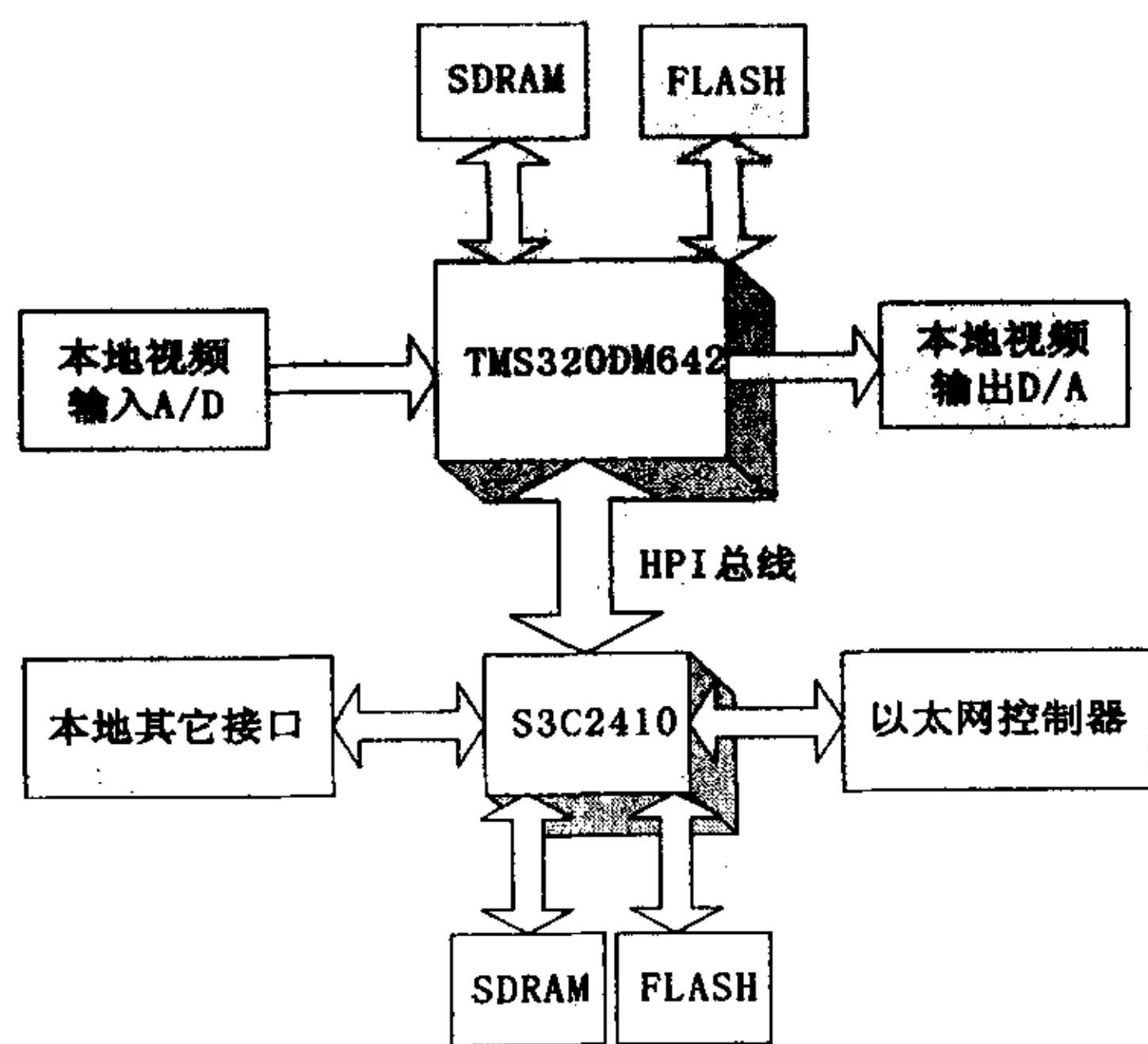


图 1 系统框图

2 时序分析

2.1 建立/保持时间

在时序分析中最重要的两个参数就是建立时间(Setup time)和保持时间(Hold time),它们是接收器本身的特性,表征了时钟边沿触发时数据需要在锁存器的输入端持续的时间(如图 2 所示)。时钟信号来的时候,要求数据必须已经存在一段时间,这就是器件需要

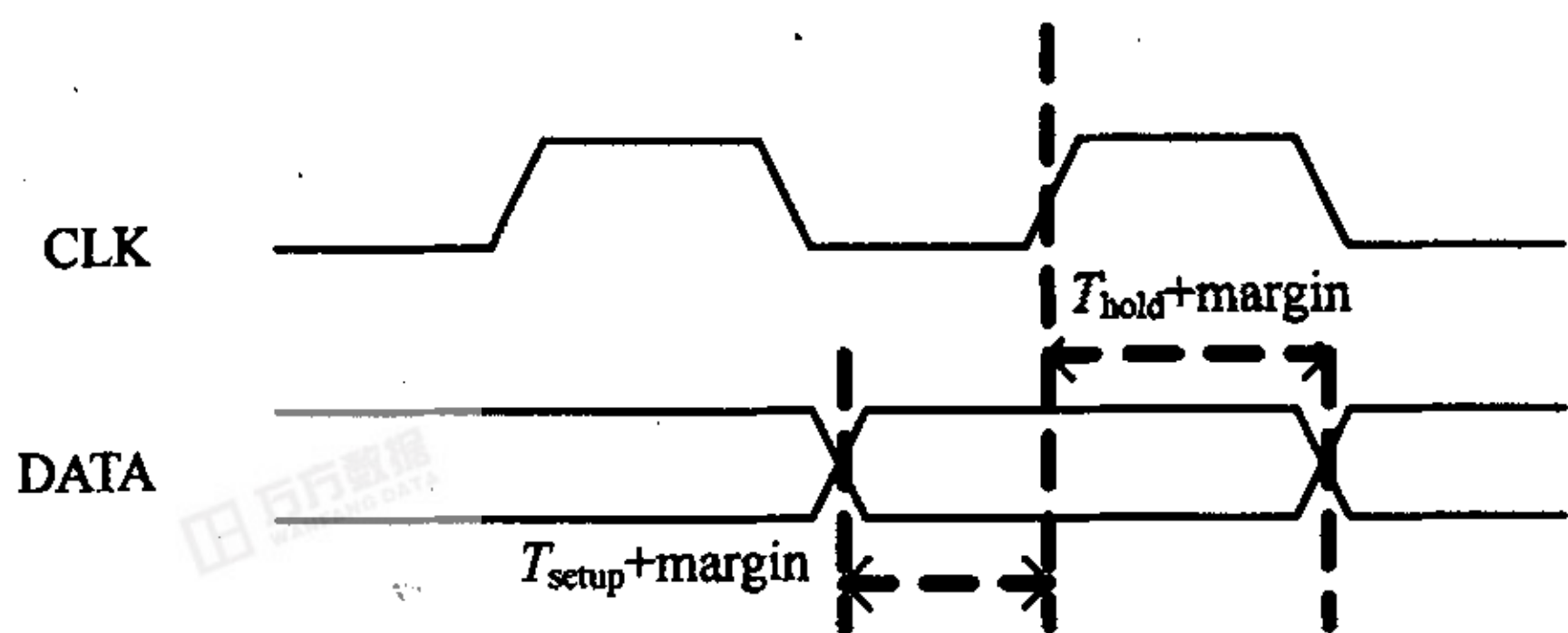


图 2 建立/保持时间

的建立时间;而时钟边沿触发之后,数据还必须要继续保持一段时间,以便能稳定的读取,这就是器件需要的保持时间^[3]。如果数据信号在时钟沿触发前后持续的时间均超过建立和保持时间,那么超过量被称为裕量(margin)。时序分析的目的就是要保证 $\text{margin} > 0$ 。

2.2 最小/最大飞行时间

信号从缓冲器出来后,要经过传输线到接收终端,信号在传输线上的传输延时称为传播延迟(Propagation delay),它只和信号的传播速度和线长有关。然而在大多数时序设计里面,最关键的却不是传播延迟这个参数,而是飞行时间(Flight time)参数,由于负载不完全匹配和反射的存在,使得信号经驱动端到接收端时,上升沿会变化,另外,由于反射和振铃的存在,使得接收端电平稳定的时间变长,这两者都将影响接收端进行有效阈值判断的时间。飞行时间包含了传播延迟和信号上升沿变化这两部分因素。一般来说,飞行时间大于传播延迟。另外,飞行时间包括最小飞行时间(Min Flight Time)和最大飞行时间(Max Flight Time)两个参数。最小飞行时间是从驱动端信号电平达到阈值电平(V_{means})时到接收端信号电平第一次穿过最小判决电平门限(V_{IL})的时间差;最大飞行时间是指从驱动端信号电平到达阈值电平时到接收端信号最后一次穿过最大判决电平门限(V_{IH})的时间差。由于信号的判决电平可能是最小判决电平和最大判决电平之间的任何电平值,所以用最小飞行时间和最大飞行时间来表征了接收端可以进行有效电平判断的一个时间范围。从图 3 中可以很容易看出两者的区别。

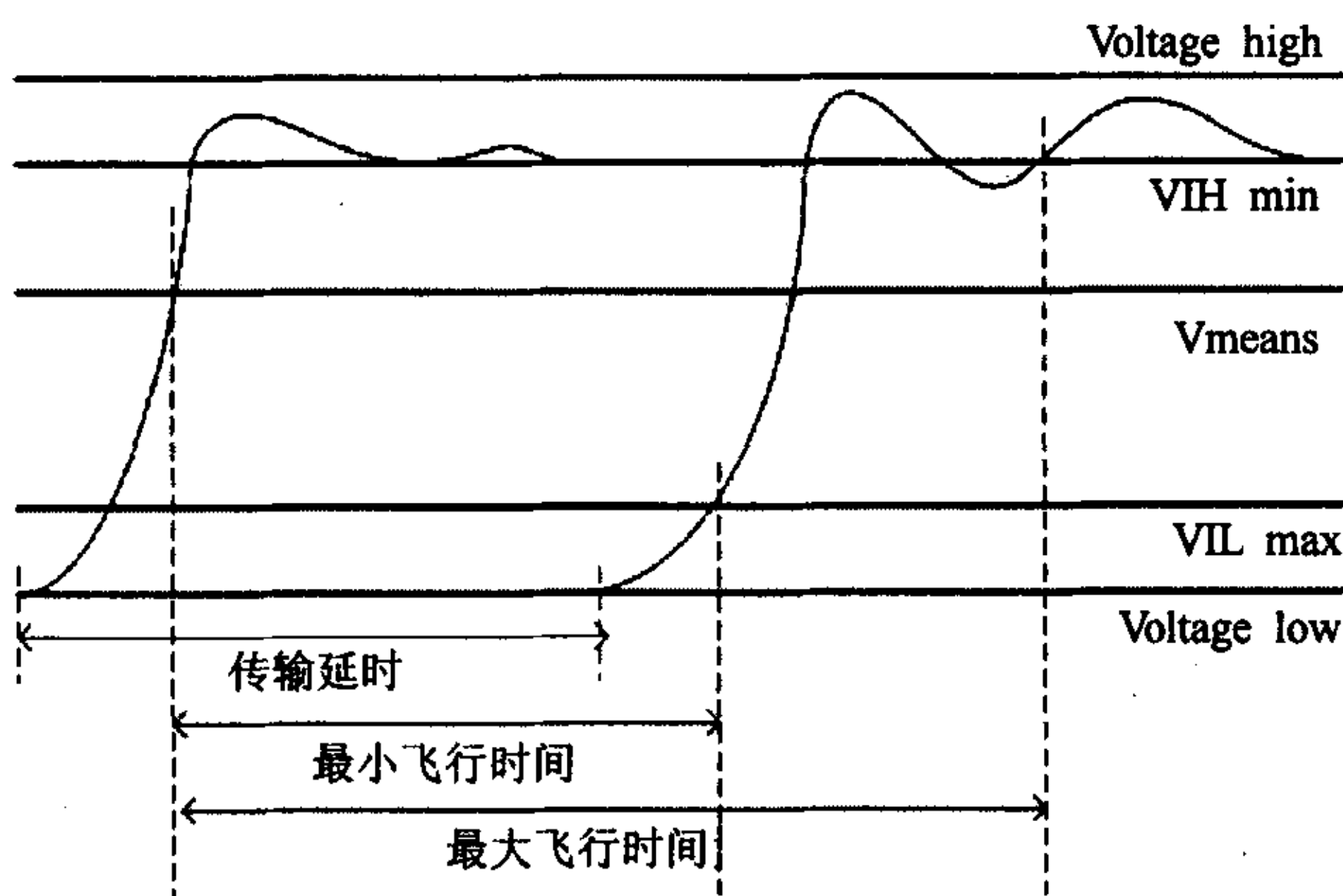


图 3 最小/最大飞行时间

2.3 时序分析和建立约束不等式

在本系统中需要分析的是 CPU 同 SDRAM 之间的数据总线的时序。由于数据总线是双向的,因此需要分别分析 CPU 读和写 SDRAM 时的总线时序。

1) CPU 写 SDRAM 时的数据总线时序。

CPU 写 SDRAM 时的数据总线时序关系如图 4 所示。其中, t_{pd} 指数据在 CPU 内部的延时,该参数是一

个范围,对应最小/最大延时两个值,下文中用 $t_{pd-\min}$ 和 $t_{pd-\max}$ 来表示。 $t_{clk-flight}$ 表示时钟信号的飞行时间,该参数也对应最小/最大时钟信号飞行时间两个值。 $t_{data-flight}$ 表示数据信号的飞行时间,同样对应最小/最大数据信号飞行时间两个值。

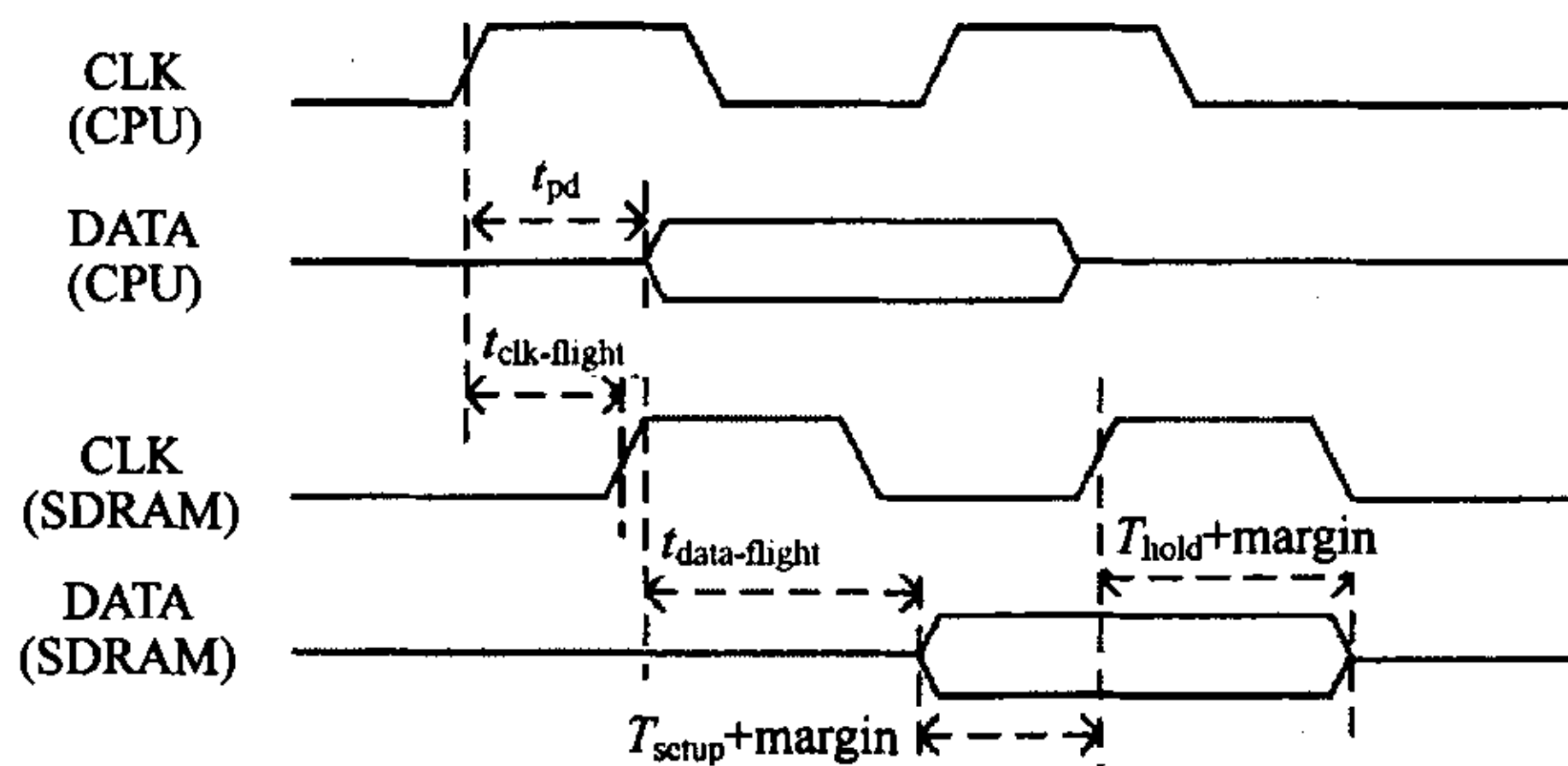


图4 CPU写SDRAM时数据总线时序

以上几个参数, t_{pd} 可以从器件的数据手册中查到。 $t_{clk-flight}$ 和 $t_{data-flight}$ 受系统的布局和布线以及阻抗匹配等关系影响。 T_{setup} 和 T_{hold} 为SDRAM端的数据建立/保持时间,可以由SDRAM的数据手册查到。由图4可以得出如下等式,其中 T_{clk} 为总线时钟周期:

$$T_{clk} + t_{clk-flight} - t_{pd} - t_{data-flight} = T_{setup} + \text{margin} \quad (1)$$

$$t_{data-flight} + t_{pd} - t_{clk-flight} = T_{hold} + \text{margin} \quad (2)$$

一般来说要求 $\text{margin} > 0$, 再引入时钟抖动 ($t_{clk-jitter}$) 和时钟倾斜 ($t_{clk-skew}$) 的影响,可以得出CPU写SDRAM时的时序约束不等式如下:

$$T_{clk} + t_{clk-flight-\min} - t_{pd-\max} - t_{data-flight-\max} - t_{clk-jitter} - t_{clk-skew} > T_{setup} \quad (3)$$

$$t_{data-flight-\min} + t_{pd-\min} - t_{clk-flight-\max} - t_{clk-jitter} - t_{clk-skew} > T_{hold} \quad (4)$$

2) CPU读SDRAM时的数据总线时序。

CPU读SDRAM时的数据总线时序如图5所示。其中, t_{ACC} 指读取命令到达SDRAM之后,SDRAM内部要花费一些时间将数据取出再送到数据总线上,这个时间称为 t_{ACC} 。 $T_{oh}(\text{SDRAM})$ 指SDRAM数据输出时的保持时间,这两个值都可以在器件数据手册中找到。

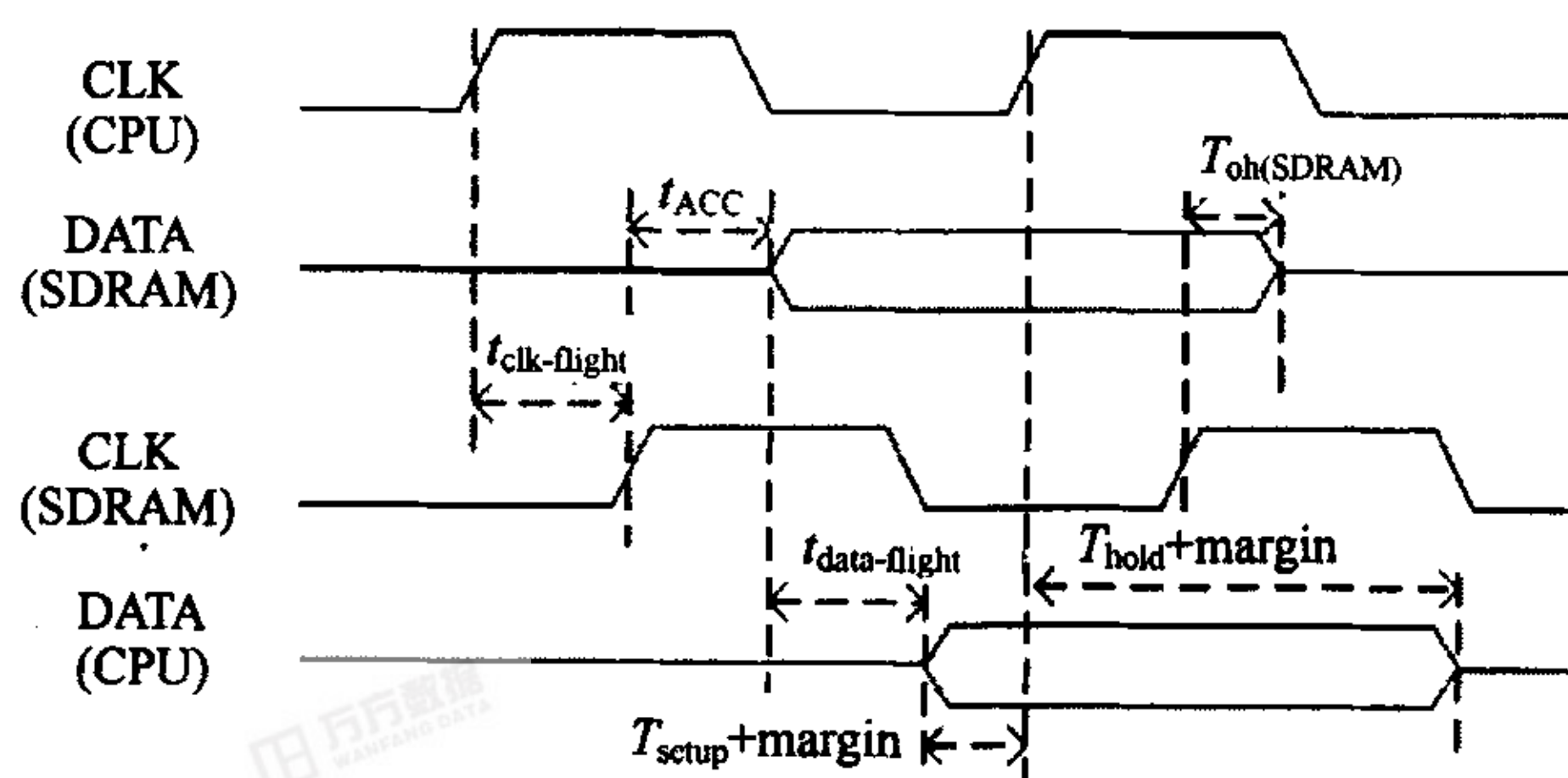


图5 CPU读SDRAM时数据总线时序

由图5可以得出以下等式:

$$T_{clk} - t_{clk-flight} - t_{ACC} - t_{data-flight} = T_{setup} + \text{margin} \quad (5)$$

$$t_{data-flight} + t_{clk-flight} + t_{oh}(\text{SDRAM}) = T_{hold} + \text{margin} \quad (6)$$

再引入时钟抖动 ($t_{clk-jitter}$) 和时钟倾斜 ($t_{clk-skew}$) 的影响,可以得出CPU读SDRAM时的时序约束不等式如下:

$$T_{clk} - t_{clk-flight-\max} - t_{ACC} - t_{data-flight-\max} - t_{clk-jitter} - t_{clk-skew} > T_{setup} \quad (7)$$

$$t_{data-flight-\min} + t_{clk-flight-\min} + t_{oh}(\text{SDRAM}) - t_{clk-jitter} - t_{clk-skew} > T_{hold} \quad (8)$$

3 用时序约束不等式指导系统设计

由以上分析看出,要使CPU和SDRAM之间的数据总线正常工作,必须满足式(3)、(4)、(7)、(8)所规定的条件,式中, T_{clk} , t_{ACC} , $T_{oh}(\text{SDRAM})$, T_{setup} 和 T_{hold} 为常量,其他的都是和系统设计相关的变量,在这些变量中间, $t_{clk-jitter}$ 和 $t_{clk-skew}$ 对结果影响较小(约为时钟周期的0.02倍),在本系统中可以忽略,前提是电源分配系统设计良好,能将由于芯片高速工作时的电流涌动 (dI/dt) 产生的“地弹”噪声减小到最小^[4]。剩下的变量中,对时序起到至关重要是 $t_{clk-flight}$ 和 $t_{data-flight}$,这两个值与系统布局布线实际情况和阻抗匹配情况有关,可以通过仿真来进行设计前规划和设计后验证。

3.1 简化时序约束不等式

在本系统中,以DSP与SDRAM之间的数据总线为例,工作频率在133MHz,所以 T_{clk} 为7.5ns;从数据手册中可以查到 $t_{pd-\min}$ 和 $t_{pd-\max}$ 分别为1.1ns和4.9ns,DSP的输入建立/保持时间 T_{setup} 和 T_{hold} 分别为1.2ns和0.8ns,SDRAM输入建立/保持时间 T_{setup} 和 T_{hold} 分别为1.5ns和1ns, t_{ACC} 为4.5ns, $T_{oh}(\text{SDRAM})$ 为2ns。代入这些数据,可将时序约束不等式简化为:

1) 简化后CPU写SDRAM时的数据总线时序。

$$t_{data-flight-\max} < 1.1\text{ns} + t_{clk-flight-\min} \quad (9)$$

$$t_{data-flight-\min} > -0.1\text{ns} + t_{clk-flight-\max} \quad (10)$$

2) 简化后CPU读SDRAM时的数据总线时序。

$$t_{data-flight-\max} < 1.8\text{ns} - t_{clk-flight-\max} \quad (11)$$

$$t_{data-flight-\min} > -1.2\text{ns} - t_{clk-flight-\min} \quad (12)$$

3.2 利用仿真完成时序控制

经过以上分析,最后未定的参数还剩下四个:

$t_{data-flight-\min}$, $t_{data-flight-\max}$, $t_{clk-flight-\min}$ 和 $t_{clk-flight-\max}$, 其中先由系统的布局决定 $t_{clk-flight-\min}$ 和 $t_{clk-flight-\max}$, 然后再以时钟信号的最小/最大飞行时间为基准,由式(9)至(12)决定数据总线上信号的飞行时间 $t_{data-flight-\min}$

和 $t_{data-flight-max}$ 。

本系统中使用 CADENCE 公司的板级设计工具 ALLEGRO 进行设计和仿真,布局之后首先布好时钟线,用 33 欧姆的电阻源端端节以保证源-负载阻抗匹配,然后进行仿真,时钟信号波形如图 6 所示。

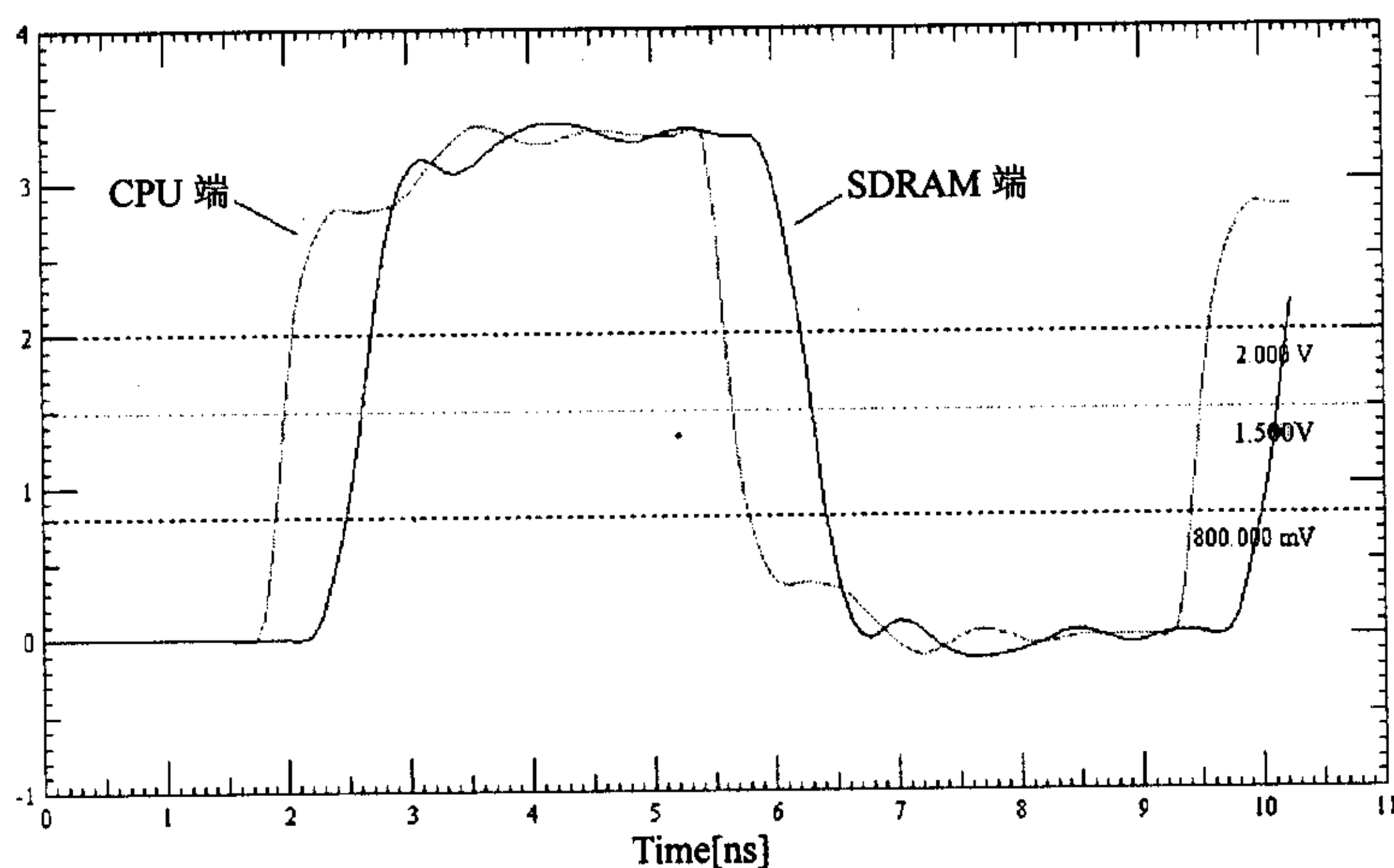


图 6 时钟信号仿真波形

从振铃、上下过冲、单调性等方面分析仿真波形,可以看出无论是源端还是负载端信号质量都较好,在此基础上再看仿真给出的 $t_{clk-flight-min}$ 和 $t_{clk-flight-max}$,这两个值分别为 0.463ns 和 0.757ns。

再以 $t_{clk-flight-min}$ 和 $t_{clk-flight-max}$ 两个值为基准,由式(9)至(12)算出数据总线上信号的飞行时间 $t_{data-flight-min}$ 和 $t_{data-flight-max}$ 满足:

$$t_{data-flight-min} > 0.657ns \text{ 且 } t_{data-flight-max} < 1.563ns \quad (13)$$

式(13)即为最终的对本系统的数据总线的时序要求,只有当数据总线上的信号满足这个式子,整个时序才能被满足。在 ALLEGRO 工具中给数据总线加上

这两个电气约束以指导布线,ALLEGRO 会对那些经过仿真不能满足上式要求的网络加上警告,便于检查和修改^[5],一般来说,最大飞行时间,即 $t_{data-flight-max}$ 容易满足条件,而 $t_{data-flight-min}$ 则可以添加延时线以符合要求。布线完毕之后,再对数据总线上的所有网络进行仿真,从波形中观察信号质量并查看最小/最大飞行时间是否满足式(13)的要求。若数据总线中每个网络的信号质量良好,且满足式(13),则表明该系统数据总线部分的时序得到了有效的控制。

4 总 结

在数字系统的运行速度越来越高,时序要求越来越严格的背景下,给出了如何保证系统关键网络时序正常的一种方法。该方法从满足器件建立/保持时间入手,并以可靠的电路仿真工具为重要手段,根据系统工作时序,推导出最小/最大飞行时间的约束,并以此来指导系统设计。

参考文献:

- [1] Bogatin E. 信号完整性分析[M]. 李玉山,李丽平译. 北京:电子工业出版社,2005.
- [2] 刘 斌,李仲阳. ARM/DSP 双核系统的通信接口设计[J]. 单片机与嵌入式设计,2005(5):22-24.
- [3] Brooks D. 信号完整性问题和印制电路板设计[M]. 刘雷波,赵 岩译. 北京:机械工业出版社,2005.
- [4] Johnson H, Graham M. 高速数字设计[M]. 沈 立,朱来文,陈宏伟译. 北京:电子工业出版社,2004.
- [5] Jon P, Mazur D. IBIS evolves: Keeping pace with signal integrity issues[M]. [s.l.]: Printed Circuit Fabrication, 1998.

(上接第 126 页)

称作集合性元素)进行处理。按照 UML 基本模型元素所建立的转换规则对集合性元素的子元素进行转换。当源模型集合中的所有元素都扫描完毕后,就自动生成了完整的 XML 模型。

4 结 论

对模型以及模型转换进行了论述,阐述了模型的概念以及模型转换的一般方法与原则。从模型的角度,把 XML 作为一种模型描述语言来描述 UML 模型,通过自动转换生成 XML 模型。最后,提出了一种从 UML 模型自动转换到 XML 模型的方法。该方法不完善的地方就是,XML 模型仅记录了 UML 模型的语义信息,没有记录 UML 图形模型的位置信息,所以

还不能从 XML 模型转换到 UML 模型。下一步的工作就是要解决这个问题。

参考文献:

- [1] 马于涛,陈建勋. 基于 UML 的软件体系结构建模技术的研究[J]. 武汉科技大学学报,2003,26(3):308-311.
- [2] Kleppe A, Warner J, Bast W. 解析 MDA[M]. 鲍志云译. 北京:人民邮电出版社,2003.
- [3] 颜玉兰,何克清,刘 进. 一种基于有限状态机的模型转换方法[J]. 计算机工程,2006,32(1):93-95.
- [4] 刘升平,林作铨,梅 婧,等. 一种 XML 模型论语义[J]. 软件学报,2006,17(5):1089-1097.
- [5] 韦银星,张申生,曹 键. UML 类图的形式化及分析[J]. 计算机工程与应用,2002(10):5-7.