静态超标量 MCU-DSP 内核的 Load 先行访存调度*

刘 博[1]，张盛兵[1]，黄嵩人[2]

（1. 西北工业大学 计算机学院，西安 710072；2. 中国电子科技集团公司第五十八研究所，江苏 无锡 214035）

摘 要：针对嵌入式控制与数字信号处理混合应用领域，建立了一种基于 MCU-DSP 融合架构处理器的 Load 先行机制。该内核使用静态超标量技术，拥有整数、存取、循环三条流水线，并采用特殊的四级流水。在存取流水线中，Load 先行机制通过动态调度指令的访存顺序，实现了 Load 指令对 Store 指令的先行，提前了整数流水线中运算操作数的准备，加快了流水线的处理速度。

关键词：微控制器（MCU）；数字信号处理器（DSP）；Load 先行；静态超标量；动态调度

中图分类号：TP332　　**文献标志码**：A　　**文章编号**：1001-3695（2013）02-0450-04

doi：10.3969/j.issn.1001-3695.2013.02.036

Load bypassing memory access scheduling for static superscalar MCU-DSP core

LIU Bo[1], ZHANG Sheng-bing[1], HUANG Song-ren[2]

(1. *School of Computer Science & Engineering, Northwestern Polytechnical University, Xi'an* 710072, *China*; 2. *China Electronics Technology Group Corporation No.* 58 *Research Institute, Wuxi Jiangsu* 214035, *China*)

Abstract: For embedded controlling and digital signal processing mixed application field, this paper proposed a novel Load bypassing mechanism based on MCU-DSP hybrid architecture processor. The MCU-DSP core used static superscalar micro-architecture and includes integer, load-store and loop three 4-stage pipelines. By dynamically scheduling the memory accessing sequence of instructions in load-store pipeline, the Load bypassing mechanism implements the execution of Load instruction aheead of Store instruction, which eventually advances the preparation of the computing operands in integer pipeline and speeds up the entire pipelines.

Key words: MCU; digital signal processor(DSP); Load bypassing; static superscalar; dynamically scheduling

随着需求的不断演变，越来越多的嵌入式应用既需要复杂的控制功能，也需要强大的数字信号处理能力[1,2]。例如，移动电话既需要执行用户界面与系统控制程序，也需要处理网络与音/视频数据的编/解码任务。传统上，这种兼具控制与信号处理双重需求的任务主要通过以下两种方式来解决：a）在微控制器的基础上扩展有关数字信号处理的指令集与执行单元；b）采用微控制器（MCU）与数字信号处理器（DSP）分别执行控制与信号处理任务的双处理器方案。这两种方式虽然能够在某种程度上满足上述需求，但也存在各自的缺点，即扩展方式效率不高，而双处理器方式能耗大、编程与调试困难。因此，越来越多的半导体厂商开始研究与开发全新的 MCU-DSP 融合架构处理器。例如，由 ADI 与 Intel 联合开发的 Blackfin 处理器[3]、Infineon 公司的 TriCore 处理器[4]、ATMEL 公司的 AVR32 处理器[5]等。相比于传统方案，融合架构具有简化设计过程、降低测试与封装成本、降低面积与功耗、提高系统整体性价比等诸多优势。通常在超标量处理器中，采用减少处理器所执行的指令条数[6]和提高指令间的并行度来增加处理器 CPI[7]的方式来降低功耗、提高性能。使用编译器对指令序列进行优化和调度，或通过硬件对指令顺序进行重组等。

基于上述趋势，西北工业大学计算机学院与某研究所合作开发了某型号 MCU-DSP 融合架构处理器内核。该内核基于 RISC Load/Store 指令集架构，支持 16/32 位两种指令格式。微架构上，该内核采用哈佛存储架构以及顺序发射、乱序完成的静态超标量结构。为了获得更高的数字信号处理速度，该内核在流水线处理中使用了 Load 先行的动态调度技术，通过尽可能早地执行 Load 访存指令来加快流水线的整体处理速度。

1 内核总体结构

MCU-DSP 融合架构处理器面向数字信号控制领域，兼具微控制器与数字信号处理器的典型特征。该内核采用 32 位 RISC 结构，物理的哈佛结构，分离数据存储器和指令存储器，具有写缓存的二路组相连数据 Cache，在存储结构上能够实现快速访问。该内核采用了两条主流水线——整数流水线（IP 流水线）和取存流水线（LS 流水线）并行执行运算与访存操作的设计方案，其中 IP 流水线包括算术与逻辑单元 ALU、位运算单元 BMU、两级乘累加单元 MAC 三种运算部件，负责执行算术与逻辑指令、数据端分支指令、位运算指令以及乘累加指令；LS 流水线处理地址计算、存储体存/取数据、系统调用，保证了数字信号处理器设计中取数、运算、存数指令序列的无阻塞执行。此外，该内核还包含了一条专为循环指令优化设计的循环流水线，以实现硬件零开销循环。该内核的基本结构如图 1 所示。

由于该内核基于 RISC 系统结构，所以运算使用的数据均来自寄存器，运算结果也都写入寄存器；能够访问存储器的操作只有两种指令，即从存储器中读取数据到寄存器的 Load 指令和从寄存器向存储器中写数据的 Store 指令[8]。在该内核存取流水线中执行的指令大致可以分为：a）访存类指令，包

收稿日期：2012-06-15；**修回日期**：2012-07-31　　**基金项目**：“核高基”重大专项基金资助项目（2009ZX01034-001-002-003）

作者简介：刘博（1988-），男，陕西宝鸡人，硕士研究生，主要研究方向为微处理器设计（liubo527@mail.nwpu.edu.cn）；张盛兵（1968-），男，教授，博导，主要研究方向为 VLSI 设计、计算机体系结构；黄嵩人（1972-），男，教授级高级工程师，主要研究方向为数字集成电路设计.

括 Load/Store 指令、原子指令和上下文指令;b)运算类指令,包括地址运算和系统指令;c)分支类指令,包括 Loop 指令和分支指令。其中,Load 指令为整数流水线中的运算准备操作数,Store 指令存储运算结果。而该内核为了减少整数流水线中对操作数的等待,在存取流水线中增加了 Load 先行机制,即通过提高 Load 指令的执行优先级,尽可能早地准备运算操作数,保证整数流水线能够满负荷工作,进而提高整个处理器的运算性能。

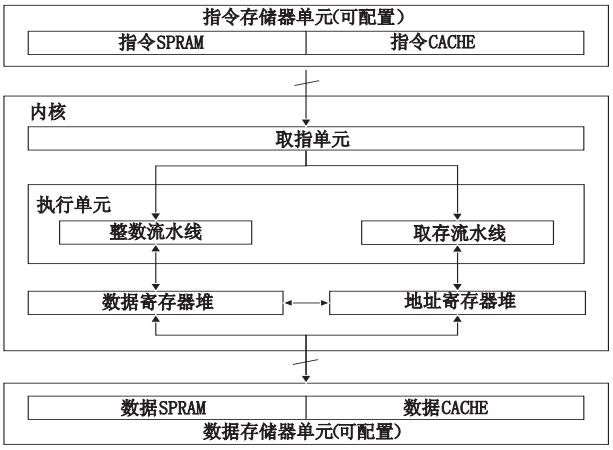

图1　内核的基本结构

2　流水段划分

基于降低成本与功耗并减轻中断响应时的流水线惩罚的微控制器设计考虑,该内核采用四级流水线设计:取指(FE)、译码(DEC)、执行(EXE)、写回(WB),相比于在单流水线内增加地址计算级与访存级的六级流水线设计,这种流水线设计方案能够以较少的开销获得更高的指令并行度与整体性能。其中取指由取指单元完成,写回由寄存器或存储器完成,存取流水线只负责存取指令的译码和执行,整数流水线只负责运算类指令的译码和执行。存取流水线的各流水段具体工作如表 1 所示,在整数流水线中,在译码阶段准备操作数,执行阶段进行运算操作。

表 1　存取流水线工作表

流水段	DEC	EXE	WB
Load 指令	指令译码;寄存器准备地址数据;访存地址产生	存储保护检测;访问存储器	
Store 指令	指令译码;寄存器准备地址数据;访存地址产生	存储保护检测;准备存储数据	访问存储器
原子指令	指令译码;寄存器准备地址数据;访存地址产生	存储保护检测;访问存储器; RF:"改"数据	访问存储器
地址运算	指令译码;寄存器准备运算数据	地址运算	
系统指令	指令译码;寄存器准备数据	数据移动	
分支指令	指令译码;IF:得到分支地址;IF:修改 PC	计算判断条件	

注:RF 表示寄存器堆的工作;IF 表示取指单元的工作。

从表 1 可知,Load 指令在执行阶段进行访存操作,而 Store 指令在写回阶段进行访存操作,这样做的好处如下:

a)避免了资源冲突。由于 Store 指令需要寄存器准备要存储的数据,如果 Store 访存在执行阶段 ,那么译码阶段就要访问寄存器获得存储的数据,但是译码阶段的地址产生过程也使用到了寄存器的读端口,除非扩展寄存器读端口,否则资源冲突。将 Store 访存放在写回阶段就会避免这个资源冲突。

b)便于使用前推技术解决 WAR 数据相关。当 Store 指令的访存在写回阶段执行时,Store 的数据源寄存器与其前一条指令产生的 WAR 数据相关就可以通过前推通路解决,并减少了寄存器相关的检测和处理。例如,同时发出的 IP 和 Store 指令,发生了读后写相关,可以采用前推通路解决;LS 流水线的前一条 Load 和后一条 Store 的读后写相关也可以用相同的办法解决。

c)原子指令可以流水执行。读修改写的原子指令 LDMST,在执行阶段对存储器进行 Load 操作,获得数据并进行修改,在写回阶段将修改后的数据 Store 回存储器的原地址。

d)便于提高 Load 指令的优先权。如表 1 所述,可以将 Load 和 Store 区分开,借此可以提高其一的优先权。Load 指令是将存储器的数据搬移到寄存器提供给 IP 进行运算,Store 指令多数是将寄存器中的运算结果保存至存储器。由于保证运算器件一直高效地工作比保存运算结果更重要,所以 Load 指令的优先级高于 Store 指令。当 Store 和 Load 同时访存(如 Store 指令后跟一条 Load 指令),如果之间不存在冲突,就可以优先执行 Load 指令,即使语法上 Load 在 Store 之后,这种机制就称为 Load 先行机制。

3　Load 先行控制

3.1　Load 先行约束规则

Load 先行是通过提高 Load 指令的优先级,在 Store 和 Load 同时访存且没有冲突时进行。并不是所有的 Store + Load 的操作组合就可以采取 Load 先行,Load 先行访存机制必须遵循以下几条约束:a)如果 Load 指令前是多周期 Store 指令、原子指令或上下文指令,那么 Load 指令必须被延迟;b)先行的 Load 指令或被延迟的 Store 指令必须是面向本地存储器的;c)Load 指令不能先行于相同访存地址的 Store 指令,即没有访存冲突;d)Store 指令之间的先后顺序不可改变,即一个后面的 Store 指令不能移动到前面的 Store 指令之前,如 Store 后的原子指令不能先行。

上述 a)和 d)中涉及的原子指令冲突,通过 hold 流水线来延迟执行;而 c)中的访存冲突在 Load 指令的 DEC 段进行检测,于 EXE 段信号有效,判定 Load 先行是否有效。

虽然有上述的四条约束,但是常用的 DSP 算法基本都能满足,所以 Load 先行机制可以在一定程度上提高整个处理器的运算性能。

3.2　Load 先行控制状态机

Load 先行机制是通过提高存取流水线中 Load 指令的优先级,尽可能早地执行 Load 指令,进行访存并获得整数流水线所需的操作数。在存取流水线中,Load 指令和 Store 指令的执行均由各自的状态机进行控制,通过各状态机之间的协调和配合实现 Load 先行控制。

存取流水线的控制通路有三个状态机,即 LOAD 状态机、STORE 状态机和 STORE 缓冲状态机。其中,LOAD 状态机用来控制存储器读操作,STORE 状态机和 STORE 缓冲状态机联合起来控制存储器写操作,通过在 STORE 缓存状态机中保存被延迟的存储器写操作来实现 LOAD 先行。

3.2.1　LOAD 状态机

LOAD 状态机控制 Load 的执行,共有五个状态,即 idle、dacc、sacc、dmiss 和 hold,分别表示空闲、访存、多周期第二次访存、访存不命中和保持状态,如图 2 所示。当有新的 Load 访存

时,由 idle 或者 sacc 进入 dacc 状态。当为 dacc、sacc 状态时,对存储器进行访存,如果没有收到访存响应信号(存储器没有准备就绪),则进入 dmiss 状态等待,一旦存储器准备就绪,返回访存状态。Hold 状态用来延迟先行 Load,当先行 Load 和前面的 Store 发生冲突时。

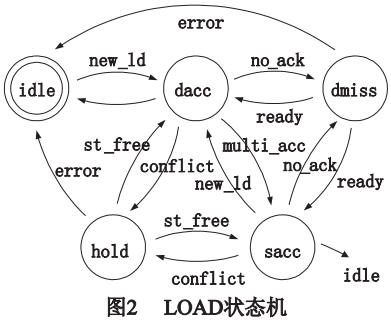

图2 LOAD状态机

3.2.2 STORE 状态机和 STORE 缓存状态机

整个 Store 指令的执行需要 STORE 状态机和 STORE 缓存状态机协作完成。STORE 状态机如图 3 所示,只有 idle、exe、stma 三个状态,分别表示空闲、执行、多周期执行状态,负责 Store 指令的译码和执行,负责发起单周期或多周期的 ST 访存请求,并交给 STORE 缓存状态机处理,在写回阶段访存。当有新的 Store 访存时,由 idle 进入 exe 状态,若为多周期访存则进入 stma 状态。STORE 缓存状态机负责 Store 指令的访存、未命中和缓存,用来辅助 STORE 状态机完成 Store 指令,实现 Load 先行机制。STORE 缓存状态机如图 4 所示,共有四个状态,即 idle、dacc、dmiss 和 pst,分别表示空闲、访存、访存不命中和缓存状态。dacc 状态时,进行存储器写操作;dmiss 状态时,等待存储器访存就绪的应答信号,以便完成指令的执行;pst 状态可以把 Load 先行时被延迟的 Store 指令的信息保存下来,等待 Load 释放存储器访问权后继续访问。当有新的 Store 访存时,一般情况由 idle 状态进入 dacc 状态;但当发生 Load 先行时,由 idle 状态进入 pst 状态,等待 Load 访存结束进入 dacc 状态。

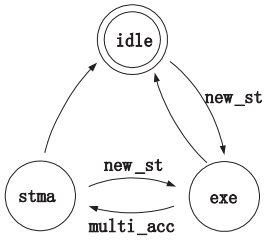

图3 STORE状态机

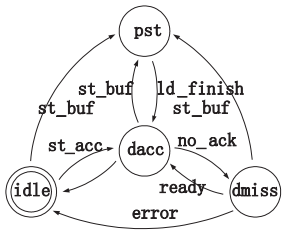

图4 STORE缓存状态机

3.2.3 状态机变化示例

状态机的状态随着指令的执行而不断变化,表 2 给出了多种 Load 访存情况、多种 Store 访存情况以及 Store + Load 访存情况时,LOAD 状态机(lmc)、STORE 状态机(smc)和 STORE 缓存状态机(smb)的变化。其中 dmiss 状态时,存储器可能需要多个时钟周期才能准备就绪,那么 dmiss 状态就要延续多个周期。

通过对比单条 Load 指令和 Store + Load 指令的状态机变化,当 Load 先行发生冲突后,Load 的访存执行就需要多两个时钟周期。执行一条 Store 指令时,当发生 Load 先行时,访存时刻要比没有发生 Load 先行时晚一个时钟周期,与有没有发生 Load 先行冲突无关。Load 先行是通过牺牲 Store 指令的执行来提高 Load 指令的优先级。

3.3 实例分析

在 3.2.3 节中,给出了单条 Load、单条 Store 以及 Store + Load 的多种情况下的状态机变化。当先行的 Load 没有冲突时,可以提前完成取数操作;当发生冲突时,Load 指令需要额外的两个时钟周期,浪费了 Store 暂存的一个时钟周期,增加了整个开销。下面以数字信号处理算法中常用的向量运算为例,对 Load 先行的访存进行分析。向量 $A(a_0,a_1,\cdots,a_N)$ 和向量 $B(b_0,b_1,\cdots,b_N)$ 进行相加操作,结果为向量 $C(c_0,c_1,\cdots,c_N)$,其中 $N\geqslant 1$,计算表达式如下:

$$A+B=(a_0+b_0,a_1+b_1,\cdots,a_N+b_N)=(c_0,c_1,\cdots,c_N)=C$$

表 2 状态机变化示例

Store/Load	访存	DEC	EXE	WB	…	…
单周期 Load	lmc	idle	dacc	idle		
多周期 Load	lmc	idle	dacc	dacc	idle	
单周期 Load miss	lmc	idle	dacc	dmiss	dacc	
多周期 Load miss	lmc	idle	dacc	sacc	dmiss	idle
单周期	smc	idle	exe	idle		
Store	sbm	idle	idle	dacc		
多周期	smc	idle	exe	stma	idle	
Store	sbm	idle	idle	dacc	dacc	idle
单周期	smc	idle	exe	idle	idle	
Store miss	sbm	idle	idle	dacc	dmiss	dacc
多周期	smc	idle	exe	stma	idle	idle
Store miss	sbm	idle	idle	dacc	dacc	dmiss
Store + 单	lmc		idle	dacc	idle	
周期 Load	smc	idle	exe	idle	idle	idle
	sbm	idle	idle	pst	dacc	idle
Store + 多	lmc		idle	dacc	dacc	idle
周期 Load	smc	idle	exe	stma	idle	idle
	sbm	idle	idle	pst	pst	dacc
Store + 单	lmc		idle	dacc	hold	dacc
Load 冲突	smc	idle	exe	idle	idle	idle
	sbm	idle	idle	pst	dacc	idle

注:“ + ”表示存取流水线中一指令后紧跟着另一指令。

上述运算的指令序列采用了 $N+1$ 组 ld、ld、add、st,即准备两个操作数、运算、存储结果,表 3 给出了 $N=3$ 时的指令序列。

表 3 向量操作指令序列

NUM	指令			NUM	指令	
0	ld	d0,[a2 +]4;	L1	8	ld	d6,[a2 +]4
1	ld	d1,[a2 +]4;	L2	9	ld	d7,[a2 +]4
2	add	d2,d0,d1;	A1	10	add	d8,d6,d7
3	st	[a4 +]4,d2;	S1	11	st	[a4 +]4,d8
4	ld	d3,[a2 +]4;	L3	12	ld	d9,[a2 +]4
5	ld	d4,[a2 +]4;	L4	13	ld	d10,[a2 +]4
6	add	d5,d3,d4;	A2	14	add	d11,d9,d10
7	st	[a4 +]4,d5;	S2	15	st	[a4 +]4,d11

注:ld d0,[a2 +]4 表示从 a2 指示的地址取数据,并且 a2 自增 4;add d2,d0,d1 表示 d2 = d0 + d1;st [a4 +]4,d2 表示把 d2 中的数据存储到 a4 所指的地址,同时 a4 自增 4;L1、A1、S1 等为指令标志符。

本文内核采用静态超标量多流水结构,所以上述指令在内核中的执行是 IP 流水线和 LS 流水线同时进行的,即算数运算的 2、6、10、14 指令在 IP 流水线中,其他的指令在 LS 流水线中执行;IP + LS 指令之间可以并发执行,如 2 与 3、6 与 7、10 与 11、14 与 15,指令顺序在编译器中优化。在指令序列中,add 指令总是相关于存取指令,即需等待前面两条 Load 指令准备数据,后面的 Store 指令需要等待其运算结束,就有可能导致 IP 流水线的停顿。假定所有的访存均在一个时钟周期完成,分两种情况进行分析:

a)在没有 Load 先行访存机制时,假设 Load 的访存依然在执行阶段,Store 的访存依然在写回阶段;add 指令的运算在执行阶段进行。指令序列在流水线中的状态如图 5(a)所示,S1 和 L3 之间访存周期冲突,L3 延迟访存;通过分析计算,内核执行完 $4\times N$ 条指令需要 $1+4\times(N+1)+1=4\times N+6$ 个时钟周期,$CPI=(4\times N+6)/(4\times N)=1+6/(4\times N)$。

b)在使用 Load 先行访存时,指令序列中 Store + Load、Store + Load + Load组合下,Load 指令在和 Store 指令没有访存冲突时就可以先行。在表 4 中的指令序列中,Store + Load +

Load 组合有 3 +4 +5、7 +8 +9、11 +12 +13,并且 Store 指令与 Load 指令之间没有访存冲突,可以进行 Load 先行,如图 5(b)中深色部分所示,L3、L4 先行与 S1 访存。通过仿真验证,内核执行完 $4\times N$ 条指令需要 $1+3+3\times N+2=3\times N+6$ 个时钟周期,$CPI=(3\times N+6)/(4\times N)=0.75+6/(4\times N)$,其中主要避免了 Store 指令和 Load 指令间的访存等待。

显然,使用了 Load 先行机制后快了 N 个时钟周期,CPI 少了 0.25。Load 先行可以加快运算数据的准备,进而使 IP 中的算数运算尽早完成,而 IP 流水线的工作饱和度由具体的指令序列来确定。

		1	2	3	4	5	6	7	8	9
整数流水线	取指	L1L2		A1S1	L3L4			A2S2		
	译码				A1				A2	
	执行					A1				A2
	写回						A1			
取存流水线	译码		L1	L2	S1	L3	L4	L4	S2	
	执行			L1	L2	S1	—	L3	L4	S2
	写回				L1	L2	S1	—	L3	

(a)

		1	2	3	4	5	6	7	8	9
整数流水线	取指	L1L2		A1S1	L3L4		A2S2			
	译码				A1			A2		
	执行					A1			A2	
	写回						A1			A2
取存流水线	译码		L1	L2	S1	L3	L4	S2		
	执行			L1	L2	S1	L3	L4	S2	
	写回				L1	L2	S1	L3	L4	S2
Store Buffer							S1	S1	S1	

(b)

图5　流水线状态

4　结束语

本文在静态超标量 MCU-DSP 融合内核的基础上,提出了在存取流水线中动态调度访存指令顺序的 Load 先行机制。在存取流水线的特殊设计下,Load 先行机制通过提高流水线中 Load 指令的优先级,尽可能早地访问数据存储器获得算数运算的操作数,从整体上加快流水线的处理速度。Load 先行是通过调整存取指令的访存顺序来获得一定的加速,要获得更高的加速比需要加大存取指令的存取带宽,提高存取的吞吐率。

参考文献:

[1] EYRE J,BIER J. The evolution of DSP processors[J]. IEEE Signal Processing Magazine,2000,17(2):43-51.

[2] GYGER A C,MASGONTY S,MORGAN J M, *et al*. Low-power 32-bit dual-MAC 120uW/MHZ 1.0V icyflex DSP/MCU core[C]//Proc of the 34th European Solid-State Circuits Conference. 2008: 190-193.

[3] KOLAGOTLA R K,FRIDMAN J,HOFFMAN M M, *et al*. A 333-MHz dual-MAC DSP architecture for next-generation wireless applications[C]//Proc of IEEE International Conference on Acoustics, Speech and Signal Processing. 2001: 1013-1016.

[4] MARTIN D,OWEN R E. A RISC architecture with uncompromised digital signal processing and microcontroller operation[C]//Proc of IEEE International Conference on Acoustics, Speech and Signal Processing. 1998: 3097-3100.

[5] UTHUS J,STRΦMΦ. MCU architectures for computer-intensive embedded applications[R]. San Jose: Atmel Corporation, 2005.

[6] MAHJUR A,TAGHIZADEH M,JAHANGIR A H. Lazy instruction scheduling: keeping performance, reducing power[C]//Proc of ACM/IEEE International Symposium on Low Power Electronics and Design. 2008: 375-380.

[7] TATE D,STEVEN G,STEVEN F. Static scheduling for out-of-order instruction issue processors[C]//Proc of the 5th Australasian Computer Architecture Conference. 2000:90-96.

[8] HENNESSY J L,PATTERSON D A. 计算机系统结构:量化研究方法[M]. 4 版. 北京:电子工业出版社,2007:298-300.

(上接第 446 页)

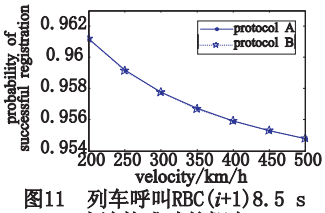

图11　列车呼叫RBC(i+1)8.5 s 内连接成功的概率

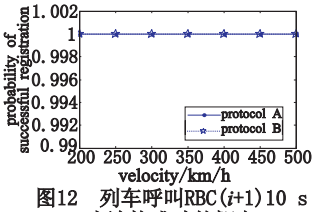

图12　列车呼叫RBC(i+1)10 s 内连接成功的概率

4　结束语

高速列车越区 RBC 切换质量是制约列车安全、高效运行的关键因素之一。通过对我国 CTCS-3 列控系统建模、分析与仿真研究,验证了不同速度条件下交接协议 A、B 的合理性和安全性。无论交接协议 A 还是交接协议 B,列车在不同速度下的 GSM-R 网络注册成功时延和连接建立时延基本相同。前者的优势在于两部电台可并发工作,具有“先换后切”的特点;后者则是“先切后换”,列车在其尾部越过切换应答器 RN 时,必须先切断与 RBC(i) 的通信连接,再与 RBC(i+1)通信,车地通信中断时间为 τ(0.2 s≤τ≤0.3 s,一般取 0.3 s)。与交接协议 A 相比,完成 RBC 切换在时间上滞后了 trainlength/$V+\tau+t_B$,如图 1、2 所示。为了列车在正常情况下不减速运行,CTCS-3 列控系统在列车间隔时间上预留了 RBC 注册时间[6] 和通信折算时间[8],约 52 s,其中也考虑了交接协议 A、B 切换时 τ=0.3 s 的车地通信中断时间。可见,交接协议 B 作为协议 A 的冗余措施,并不存在安全上的问题,但 RBC 重叠覆盖区域在协议 A 的基础上作了延伸,达到 80.3 s + trainlength/V($=t_A$ + trainlength/$V+\tau+t_B$,这是协议 A 可以正常进行且在无法正常进行时切换至协议 B 顺利完成车地通信必要的 GSM-R 覆盖区域(时间)),因而 RBC 切换效率有所降低,行车效率也受到一定影响。提速情况下,可考虑适当增加 RBC 重叠覆盖区域、场强和列车间隔时间裕量,进一步增强 RBC 交接的可靠性。

参考文献:

[1] 牛儒,曹源,唐涛. ETCS-2 级列控系统 RBC 交接协议的形式化分析[J]. 铁道学报,2009,31(4):52-58.

[2] 曹源,牛儒,唐涛,等. 基于 SPN 的越区切换模型分析[J]. 铁道学报,2009,31(4):104-107.

[3] 徐田华,唐涛. 基于有色 Petri 网的 ETCS 通信系统与列车间隔分析[J]. 系统仿真学报,2007,19(21):5038-5041.

[4] 陈永,王晓明,党建武,等. 基于 SPN 的 CTCS 无线通信形式化建模与分析[J]. 铁道学报,2011,33(8):63-68.

[5] 刘中田,孙伟亮. 基于安全计算机平台的无线闭塞中心仿真研究[J]. 计算机应用研究,2011,28(3):1017-1019.

[6] 铁道部. 科技运[2008]168 号,CTCS-3 级列控系统 GSM-R 网络需求规范(v 1.0)[S]. 北京:中国铁道出版社,2008.

[7] 铁道部. 科技运[2009]28 号,GSM-R 数字移动通信网设备技术规范 第二部分:机车综合无线通信设备(v 2.0)[S]. 北京:中国铁道出版社,2009.

[8] ZIMMERMANN A, HOMMEL G. A train control system case study in model-based real time system design[C]//Proc of International Parallel and Distributed Processing Symposium. Wshington DC: IEEE Computer Society,2003:118-130.