基于 VHDL-AMS 的混合信号模拟器

肖立伊[1],叶以正[1],李 滨[1],黄国勇[2],郭进军[2],张 鹏[2]
(1. 哈尔滨工业大学微电子中心,黑龙江哈尔滨 150001;2. 华大集成电路设计中心,北京 100015)

摘 要: 文中概括地介绍了 VHDL-AMS 语言,研究了其混合信号机制,提出了一种新的同步算法及四种确定 A 到 D 转换时间的算法,实现了一个基于这种语言的混合信号模拟器,通过例子验证了该模拟器的有效性.

关键词: 行为级混合信号模拟; 同步算法; VHDL-AMS
中图分类号: TN319 **文献标识码**: A **文章编号**: 0372-2112 (2001) 08-1023-05

A Mixed-Signal Simulator Based on VHDL-AMS

XIAO Li-yi[1], YE Yi-zheng[1], LI Bin[1], HUANG Guo-yong[2], GUO Jin-jun[2], ZHANG Peng[2]
(1. *Microelectronic Center of Harbin Institute of Technology, Harbin, Heilongjiang* 150001, *China*; 2. *Huada IC Design Center, Beijing* 100015, *China*)

Abstract: The VHDL-AMS language is summarized in the paper. We have studied the mixed-signal mechanism of the language and offered a new synchronization algorithm and four algorithms for determining the convert time point of A to D. A mixed-signal simulator based on the language has been implemented. The efficiency of the simulator is tested by examples.

Key words: behavioral mixed-signal simulation; synchronization algorithm; VHDL-AMS

1 引言

随着电子系统复杂性的增加,需要在设计的初期就进行整体验证. 事实上所有电子系统都包含有模拟和数字电路,所以高层次混合信号模拟是当今 EDA 环境中不可缺少的工具. 国外从 80 年代中期就开始出现用于数/模混合信号系统高层次设计开发的模拟环境,如 ANACAD 公司的 VHDeLDO[1]、Analogy 公司的 Saber[2]、Cadence 公司的 Spectre[3]等模拟器. 由于这些模拟器都是以专利语言或 spice 作为输入的,通用性差. IEEE 标准化委员会已扩展 VHDL 到模拟及数/模混合信号领域,即 VHDL-AMS IEEE Std 1076.1-1999[4]. VHDL-AMS 是 VHDL '93 向模拟和混合信号领域的扩展. 这种语言区别于以往的混合信号描述方式是以单一的语言在一个统一的语言环境中描述模拟和数字部分. 为此 VHDL-AMS 中增加了新的语言结构,并定义了新的混合信号模拟周期.

对于模拟信号 VHDL-AMS 用保留字"quantity"定义,quantity 具有连续幅值,只能是浮点类型. VHDL-AMS 中同时引入了一些隐式 quantity,如 quantity 对时间的微分(Q'dot)、积分(Q'integ)等,Q 代表一个标量的 quantity.

VHDL-AMS 支持信号流和守恒系统建模方式. 对于守恒系统其端口及内部节点用保留字"terminal"定义,同时要说明端口及节点的性质.

联立语句是为描述连续系统的行为而新增的语句,可以在并发语句区出现,在确立时写成特征表达式的形式.

目前国际上用 VHDL-AMS 建模和模拟已成为 EDA 领域的研究热点[5,6].

2 VHDL-AMS 的混合信号机制

研究 VHDL-AMS 的混合信号机制,是实现混合信号模拟的前提. VHDL-AMS 中硬件行为的数字部分用并发语句描述,模拟部分的行为是用联立语句来描述的,这些并发语句和联立语句一起描述一个设计的行为或结构.

数字与模拟部分分别采用不同的算法执行,所以基于 VHDL-AMS 的混合信号模拟器集成了两类算法,一类是事件驱动算法,一类是微分/代数方程组的求解算法. 执行数字算法的引擎称为数字引擎,执行进程的集合;执行模拟算法的引擎称为模拟解算器,求解联立语句的集合.

2.1 D/A 转换

D/A 转换就是模拟能从执行进程到执行联立语句,并把数字部分的信息传给模拟部分,VHDL-AMS 中只有中断语句(break statement)可以实现 D/A 转换. 中断语句是 VHDL-AMS 中新增的语句,有并发和顺序两种形式. 执行时并发的中断语句转换成一个等价进程,该进程没有敏感表,说明部分是空的,语句部分包含一个顺序中断语句及紧跟其后的 wait 语句.

中断语句中断的是模拟部分的连续性,模拟时用一个中断标志(BreakFlag)来表示,数字引擎执行到中断语句时,BreakFlag 赋为'1'. 在下一模拟周期,当模拟解算器判断出 BreakFlag 为'1'时,要处理这个非连续性,也预示该模拟周期是一个 δ 周期.

收稿日期:2000-09-01;修回日期:2001-01-15

数字部分通过两种方式向模拟部分传递信息:

(1) 联立语句中可以引用信号名. 由于进程的执行,信号值会发生变化,通过适当的中断语句模拟解算器能响应这一变化,就可以把数字部分的信息传递给模拟部分.

(2) 中断语句也可以为 quantity 赋新的初始条件,从而影响模拟部分的执行.

2.2 A/D 转换

由 A 到 D 的转换就是要求模拟能从模拟解算器到数字引擎,把模拟部分的信息传给数字部分. VHDL-AMS 中是由信号 Q'above(E) 来完成的. 前缀 Q 是一个标量 quantity, E 是一个与 Q 相同类型的表达式. Q'above(E) 的值是布尔型的,当 Q 的值大于阈值 E 时为'真',小于时为'假',这两种状态间的转换在这个信号上产生一个事件,这个事件可以激活进程. 由于 quantity 名可以用在任何类型允许的表达式中,所以进程可以取到 quantity 的值,即可以读到模拟部分的信息.

以上两种机制完成 D/A、A/D 转换.

3 混合信号模拟

3.1 模拟环境

VHDL-AMS 混合信号模拟环境如图 1 所示. 其中 VHDL-AMS 源码分析器负责检查模型的语法及语义,即编译,通过检查可以发现模型描述中的静态错误,如没有静态错误,则转换成中间数据格式,保存在设计库中. 确立是从中间数据格式确立设计层次. 一个 VHDL-AMS 模型完全确立以后,数字部分映射成进程的集合,连续部分映射成特征表达式,这些特征表达式分别属于 4 个集合,分别是:

(1) 显式集; (2) 结构集; (3) 增广集; (4) 中断集.

增广集又划分为 5 个,这些增广集在模拟时起不同的作用,在模拟的不同阶段可能用到某一增广集,此时该增广集称为当前增广集.

图 1 所示的混合信号模拟器包含两部分,一是数字引擎,另一个是模拟解算器. 对于混合信号模型,数字引擎与模拟解算器之间要通过同步算法交换信息.

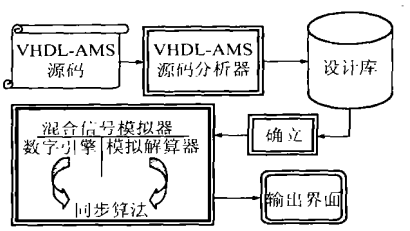

图 1 VHDL-AMS 混合信号模拟环境

3.2 同步算法

对于混合信号模拟,数字部分与模拟部分的通信是非常关键的,它影响模拟的正确性和模拟器的性能. 数字部分的执行是按照事件发生的先后顺序,依次执行. 每一事件执行结束后,模拟时间不前进,下一事件时间是已知的;对于模拟部分,每一个模拟解算点 *ASPs* (analog solution points) 都要经过反复的迭代求解,当计算出的 *ASPs* 收敛后,模拟解算器向前前进一个时间步长,然后利用已求出的 *ASPs* 估算出下一时间步长,计算下一个 *ASPs*,只有计算出的 *ASPs* 收敛后,才能说模拟解算器前进到某个时刻,计算非常消耗 CPU 时间. 所以要使这两部分间正确交换数据,并减少计算工作量,要作到以下两点:

(1) 数字与模拟部分应按自身的时间步长前进,不增加额外的步长计算.

(2) 精确判断 D/A 及 A/D 转换时刻,及时转换模拟执行方向.

VHDL-AMS 的混合信号模拟周期明确规定了数字与模拟执行间的同步方式. 时域混合信号模拟周期包含以下步骤:

(1) 执行模拟解算器;

(2) $T_c = T_n$, T_c 是当前模拟时间, T_n 是下一模拟周期时间,如果 $T_n = TIME'HIGH$ 或没有激活的驱动器或在 T_n 时刻没有进程重新开始,则模拟结束;

(3) 更新激活的显式信号,更新隐式信号,可能产生事件;

(4) 执行所有激活的非延缓进程,直到挂起. 如执行了中断语句,则中断标志 *BreakFlag* 为'1', $T_n = T_c$,下一模拟周期为一 δ 周期;否则确定下一模拟时刻 T_n;

(5) 如果 $T_n = T_c$,则下一模拟周期是一 δ 周期;

(6) 执行延缓进程,直到挂起;确定下一模拟时刻 T_n

按照混合信号模拟周期数字与模拟部分执行的同步算法如图 2 所示. 数字和模拟两条时间线分别代表数字引擎和模拟解算器,实心圆点"●"和方块"■"分别代表已执行完的数字事件和计算出的 *ASPs*,空心"○"代表尚未执行的数字事件. 假设数字与模拟部分在时刻 T_c 同步, T_n 为下一事件时间,模拟解算器从 T_c 向 T_n 前进,可能产生一系列的 *ASPs*,如图 2(a) 所示,最后一点 *ASPs* 要在 T_n 时刻产生,然后模拟解算器挂起,模拟转向数字部分. 如果在 T_c 到 T_n 的过程中,有事件产生(某一 Q'above(E) 信号上有事件),这时模拟解算器要挂起,并把这一事件插入事件队列. 接下来数字引擎要处理这一额外事件,此时又获得同步,如图 2(b) 所示. 图 2(c) 所示的情况是,在 T_n 时刻由于执行了中断语句,模拟解算器要处理这一非连续性,与 T_n 时刻相对应重新计算出一个 *ASPs*,使数字与模拟部分保持同步.

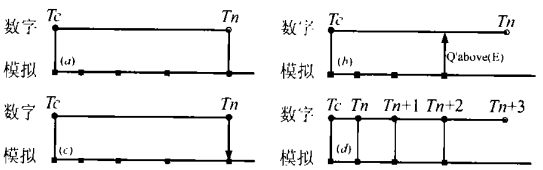

图 2 (a) T_c 到 T_n 过程中没有 Q'above(E) 事件产生; (b) T_c 到 T_n 过程中有 Q'above(E) 事件产生; (c) 在 T_n 点出现非连续性,模拟解算器重新计算 T_n 点的 *ASPs*; (d) 模拟步长大于数字事件的情况

根据这种算法. 模拟过程中数字引擎是按照自身的步长执行的,模拟解算器在 $[T_c, T_n]$ 间隔内,最后一点 *ASPs* 必须落在 T_n 时刻,这就意味着数字事件是主宰,模拟解算器必须与每一数字事件时间点保持同步. 对于数字事件密集或模拟部分步长较大的情况,如图 2(d) 所示,模拟解算器就要增加很多的额外步长,而在混合信号模型的执行时间中模拟部分的执行时间占主导地位,从而延长了模型的整体执行时间.

针对这种情况,我们对上述算法进行了改进,提出一种新的算法,如图 3 所示. 新算法与原算法的不同之处在于对 T_n 时刻模拟步长的处理上. 新算法允许模拟解算器超过 T_n 计算出一点 *ASP*s,然后挂起,转由数字引擎执行 T_n 时刻的事件. 如果数字部分用到某一 quantity 的值,由 T_n 时刻两侧的 quantity 值作线性插值求出. 如图 3(*a*)所示,"×"代表由插值求出的 *ASP*s,而不必调用模拟解算器去求解系统方程组,从而减少模拟执行时间. 如果在 T_n 时刻中断标志为'1',出现非连续性,模拟解算器必须返回跟踪,对应 T_n 时刻计算出一点 *ASP*s,超过 T_n 时刻的 *ASP*s 要丢掉,如图 3(*b*)所示. 这种方法的优点是数字引擎与模拟解算器按各自的时间步长前进,只有当非连续性发生时,模拟解算器返回跟踪一次,丢掉最后计算出的 *ASP*s.

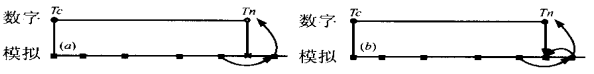

图 3　(*a*) T_n 点的 *ASP*s 由插值求出;(*b*)在 T_n 点出现非连续性,模拟解算器返回计算 T_n 点的 *ASP*s

3.3　模拟解算器

每一模拟周期开始都调用模拟解算器,根据图 3 所示改进的同步算法,模拟解算器的求解过程可以描述为:

每一模拟周期{
如果 *BreakFlag* = 1,则
　返回 T_n 时刻;
　当前增广集 ⇐非连续性增广集 ⇐中断集;
　求解集合 = 显式集 + 结构集 + 当前增广集;
　计算 *ASP*s 直到收敛;
　BreakFlag = 0,中断集 ⇐Ø;
如果 *BreakFlag* = 0,则
　重复{
　当前增广集 ⇐时域增广集
　预估 T_i, $T_i \in [T_c, T_n]$, $i \geq 1$;
　如果 $T_i > T_n$,则超过 T_n 确定一点;
　求解集合 = 显式集 + 结构集 + 当前增广集;
　在 T_i 点计算 *ASP*s;
　如果收敛,则
　　如果没有 Q'above(*E*)事件产生同时 $T_i < T_n$,则
　　重复以上步骤;
　　如果有 Q'above(*E*)事件产生,则
　　Q'above(*E*)事件插入事件队列;
　　模拟解算器挂起;
　　如果 $T_i \geq T_n$,则
　　模拟解算器挂起;
　否则
　　减少时间步长,重复以上步骤}
}

求解过程中如果某一 Q'above(*E*)信号上产生事件,则模拟解算器要挂起,由数字引擎处理这一额外事件. 如果模拟解算器重新开始执行的时间是 $T_n = T_c$,则是一个 δ 周期,模拟解算器需产生一个 *ASP*s.

模拟解算器实质上是一个算法的集合,其中包括的基本算法有,求解微分方程组的梯形法和吉尔法,求解非线性代数方程组的牛顿法,求解代数方程组的 LU 分解法等.

在每一模拟周期中,如果设当前时刻为 T_i,相应的 quantity 值为 Q_i,前一时刻为 T_{i-1},对应 Q_{i-1},如果 $(Q_i - E) * (Q_{i-1} - E) < 0.0$,则认为 Q'above(*E*)上有事件产生,否则无事件产生. Q'above(*E*)上产生事件的时间事先并不知道,只能在求解过程中确定. 当判断出 Q'above(*E*)信号上有事件产生后,模拟解算器必须返回,确定出事件产生的精确时间 T_e. 这也意味着 Q'above(*E*)信号产生事件时间点的确定比一般的 *ASP*s 要花费更多的 CPU 时间. 本文就这一问题进行了研究,给出了四种算法.

(1)线性插值法　阈值交叉时间 T_e 是利用一阶插值方法确定的,前后时间点构成一个直线方程,如图 4(*a*)所示. 由 *E* 与直线方程的交点计算出产生事件的时间 T_e:

$$T_e = T_{i-1} + \frac{E - Q_{i-1}}{Q_i - Q_{i-1}}(T_i - T_{i-1})$$

(2)二次插值法　二次插值法如图 4(*b*)所示,先由线性插值计算出一点 T_e',再由模拟解算器计算出 T_e' 时刻对应的 Q_e',由 (T_{i-1}, Q_{i-1}), (T_e', Q_e') 和 (T_i, Q_i) 三点作二次插值得插值函数 $L_2(T)$,由阈值 *E* 与 $L_2(T)$ 的交点计算出 T_e.

(3)割线法　割线法如图 4(*c*)所示,先由线性插值法计算出一点 T_e',再由模拟解算器计算出 T_e' 时刻对应的 Q_e',如果 $Q_e' < E$,则由 (T_e', Q_e') 和 (T_i, Q_i) 点作割线;否则由 (T_e', Q_e') 和 (T_{i-1}, Q_{i-1}) 点作割线,如此循环,直到 *Q* 与 *E* 的差满足精度要求为止.

(4)二分法　Q'above(*E*)信号产生事件的时间用二分查找法确定. 如图 4(*d*)所示,当在 T_i 与 T_{i-1} 之间判断出有事件产生时,模拟解算器取当前步长的 1/2 返回计算,如果计算出的值在事件产生时间的右侧,则重复上述步骤,如果在左侧,则从该点开始,重复上述步骤,直到 *Q* 与 *E* 的差满足精度要求为止.

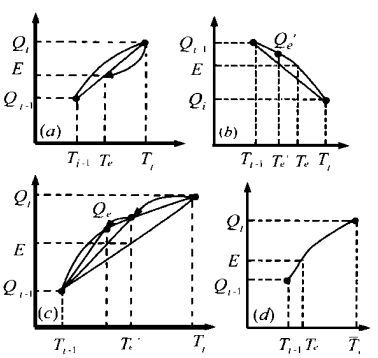

图 4　(*a*)线性插值法;(*b*)二次插值法;(*c*)割线性;(*d*)二分法

4　验证及分析

对上节中确定 Q'above(*E*)事件时间点的算法我们用例子进行了比较,模拟结果示于表 1. 其中返回次数是指在模拟时间内,模拟解算器确定 Q'above(*E*)时间点需返回跟踪次数的总和,可以用来分析每种算法的计算量,精度 ε 用于二分法和割线法. 从计算量看,线性插值方法计算量最小,其次是二次插值法或割线法,计算量最大的是二分法. 前三个模型采用四

种方法的运行时间差别不大,因为模型规模很小,只有一到两个联立语句,所以从运行时间上看不出哪种方法占优势. 模型 4 是一个混合信号的锁相环电路,其结构如图 5(*a*) 所示. 其中相位探测器采用纯数字行为级模型,电荷泵是混合信号模型,环路滤波器是结构级模型,压控振荡器是模拟行为级模型. 该模型相对复杂,从表 1 可以看出运行时间上的差别.

每一模型用四种方法模拟的结果非常接近. 通过比较可以看出,线性插值和二次插值法的计算量小,二分法和割线法精度可调,在相同的精度下割线法的计算量小,模型中所包含的方程越多,运行时间差别越显著,从我们目前所模拟的例子看线性插值法就可以满足要求,所以在我们的模拟器中首选线性插值法.

对于两种同步算法,用四个模型进行了比较,结果示于表 2. 表中加速率是指采用新同步算法比 VHDL-AMS 同步算法模拟速度提高的速率. 其中模型 ADC_B 和 ADC_S 是一个用于语音系统的数/模转换器的行为级和结构级描述,用两种同步算法得到的输出波形相同,但新算法速度更快一些,ADC_S 与 ADC_B 相比包含较多的方程,所以速度提高得更多. 这里给出的数/模转换器,输入是一个线性变化的模拟信号,经过转换输出一个 8 位的数字信号,模拟结果如图 6 所示.

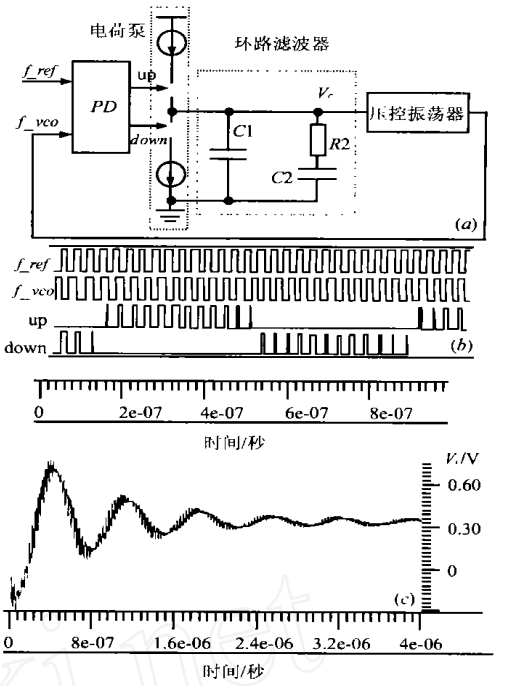

图 5 (*a*) 锁相环电路; (*b*) 锁相环电路中的数字波形; (*c*) VCO 的控制电压

表 1 模型 1 到 4 用四种方法的计算结果 精度 $\varepsilon = 1.0e-3$

模型	模拟时间(秒)	返回次数(次)				运行时间(秒)			
		线性插值	二次插值	割线法	二分法	线性插值	二次插值	割线法	二分法
1	60	123	246	216	1130	6.28	6.41	6.38	7.59
2	35	166	243	413	1041	3.12	3.17	3.37	4.15
3	40	160	320	320	1075	9.65	9.87	9.91	11.18
4	4.0e-6	372	550	969	1492	1014.16	1023.21	1050.63	1082.88

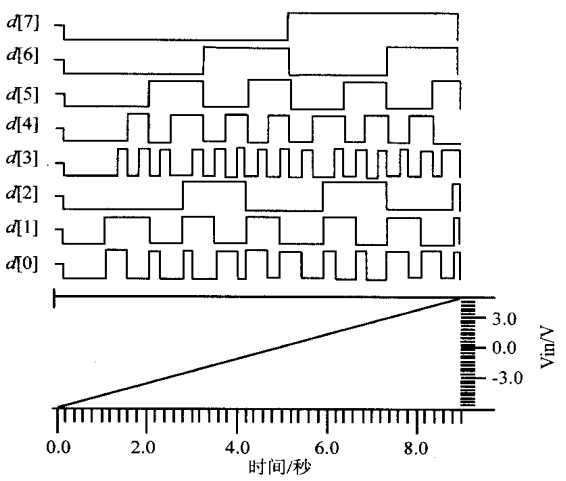

图 6 模拟到数字转换器的输入和输出

模型 PLL_3 是图 5(*a*) 所示的锁相环电路,参数取自文[7]. PLL_4 也是一个锁相环模型,结构与图 5(*a*) 所示的锁相环电路相近,只是环路滤波器由四个元件组成,该模型的参数取自文[8]. 从表 2 看出采用新算法模拟速度提高 30 %以上.

PLL_3 的模拟波形示于图 5(*b*) 和图 5(*c*). 图 5(*b*) 是模型中的数字波形,图 5(*c*) 是压控振荡器的控制电压 *Vc*,该锁相环在 2.53μs 以后稳定.

表 2 两种同步算法的比较结果

模型	模拟时间	运行时间		加速率
		VHDL-AMS 同步算法	新同步算法	
ADC_B	9ms	25.4s	20.84s	18 %
ADC_S	9ms	513.89s	382.07s	26 %
PLL_3	10us	3367.18s	2228.52s	34 %
PLL_4	10ms	1051.28s	712.68s	32 %

5 结论

本文实现的基于 VHDL-AMS 的混合信号模拟器经验证是有效的. 用线性插值法确定 A 到 D 转换时间及采用新的同步算法都可以提高模拟效率,特别是对数字事件密集的情况. 采用该混合信号模拟器,设计者可以在高层次设计模拟及混合信号电路与系统,是一种通用的高层次的模拟及混合电路仿真工具.

参考文献:

[1] H Ei Tahawy, D Rodriguez, S Garicia-Sabiro, J-J. Mayal. VHDeLDO: A new mixed mode simulator [A]. EURO DAC [C], 1993: 9/20 - 9/24.

[2] C K Chuang, C G Harrison. Analogue. Behavioral modeling and simulation using VHDL and SABER-MAST [A]. IEE Colloquium Computer And Control Division [C], 1994: 1/1 - 1/5.

(上接第 1022 页)

分析指出,DD 的杂散谱有三个来源:一是相位累加器的相位舍位误差;二是幅度量化误差;三是 DAC 非线性误差. 这些误差使 DDS 杂散电平增加,应通过仿真、优化设计来降低. 参考时钟的长稳和短稳(相噪)性能对 DDS 输出信号质量作用重大,优选高性能时钟十分必要.

2.2　采用 DDS 倍频扩展频带,改善频谱的方法.

由于 DDS 杂散电平和谐波电平高,而且目前输出频率还不高,限制了它在高性能雷达中的直接应用. 对 DDS 频谱分析和仿真研究表明:为了扩展频带,改善频谱,在产生宽带高纯频谱信号时,不能完全利用 DDS 相对带宽很宽的优点,只能选择 DDS 低杂散和低谐波电平一段有限的频带,而宽的频带和高的输出频率可通过多次倍频或变频达到. 这不仅降低了对滤波器的要求,而且可获得优良性能. 用 STEL - 1175 芯片制成 DDS,输出 0～25MHz 宽带信号,经倍频扩展频带和改善频谱处理,使输出 75～105MHz 宽带信号,杂散电平比倍频前降低 10～15 dB,谐波电平比倍频前降低 15～20dB.

2.3　采用 DDS/ DPLL 扩展频带的方法

数字式锁相频率合成(DPLL)技术具有很高的工作频率(> 2000MHz)、宽的频带、纯的频谱,但频率转换时间较长和频率分辨率较低是其缺陷;而 DDS 则具有极高的频率分辨率和极短的频率转换时间,但工作频率受限. 现将二者结合起来,取长补短,具有电路简单、使用硬件少、功耗低的优点,并获得高性能.

2.4　DDS 控制电路与雷达波形库

为了产生各种信号形式,求雷达模糊函数的数值解,编制雷达波形软件库,输出雷达信号形式有:单载频矩形脉冲信号、普通连续波信号、高斯包络单载频脉冲信号、正弦调频信号、线性调频脉冲信号、非线性调频脉冲信号、Barker 码脉冲信号、L - 序列脉冲信号、M - 序列脉冲信号等. 由主控计算机选择雷达波形库参数后,通过 DSP(TMS32031)去控制 DDS 和 PLL 电路,产生所需的各种宽带信号.

3　DDS/ PLL/ 混频产生宽频带雷达信号的技术方案

该方案由 DDS 电路模块、锁相倍频电路、时钟电路和 ×N 相参本振、CPU 及 DSP 控制电路、雷达波形库、混频滤波六大部分组成. 主控计算机及 DSP 控制电路完成对 DDS 波形产生和实现 DPLL 的控制.

DDS 基准信号发生器在控制电路的作用下,可产生 0～25MHz 宽带复杂波形信号;或控制 DDS 基准频率激励 PLL 锁相倍频电路实现频率扩展和全程扫频与跳频. 在该方案中,一路输出微波段(1200～1650MHz)宽带雷达信号,可以获得优良的相位噪声指标和杂散性能. 另一路经过混频器进行频率变换,可获得所需的超倍频程带宽(200～650MHz)的雷达信号输出. 本技术方案的特色是解决了宽带、超宽带下复杂波形产生与改善频谱的难题.

4　性能结果及应用前景

①DDS 输出性能:

频率范围:0～25MHz;　频率分辨率:1Hz;　杂散电平:< - 45dBc;

相位噪声:< - 125dBc/ 1Hz/ 1KHz;转换时间:< 500ns;输出功率:10dBm 1dBm.

②DDS/ DPLL 输出性能

频率范围:1200～1650MHz;　频率分辨率:30Hz,可选择;杂散电平:< - 70dBc

相位噪声:< - 100dBc/ 1Hz/ 1KHz;　转换时间:22μs;输出功率:> 10dBm.

③DDS/ PLL/ 混频方案产生的宽带雷达信号

频率范围:200～650MHz;　频率分辨率:30Hz,可选择;杂散电平:< - 55dBc

相位噪声:< - 95～100dBc/ 1Hz/ 1KHz;输出功率:> 10dBm;　谐波电平:< - 35dBc.

该成果相关技术已应用于电子对抗系统、抗反辐射导弹设备上,还可应用于成像雷达、目标识别雷达、相控雷达、抗干扰通信等,是新一代雷达和通信系统的关键技术,具有广泛的军用和民用前景.

[3] R J Binns, P Hallam, B Mark, R Massara. High-level design of analogue circuitry using an analogue hardware description language [A]. Mixed-signal AHDL/ VHDL Modeling and Synthesis IEE Colloquium [C], 1997:3/ 1 - 3/ 8.

[4] DASC of IEEE Computer Society. IEEE Standard VHDL Analog and Mixed-Signal Extensions [S]. New York, USA, IEEE Inc, 1999.

[5] A J Perkins, M Zwolinski, C D Chalk, B R Wilkins. Fault modeling and simulation using VHDL-AMS [J]. Analog Integrated Circuit and Signal Processing, 1998, 16(2):141 - 155.

[6] Peter Frey. SEAMS: Simulation environment for VHDL-AMS [A]. Proceedings of the 1998 Winter Simulation Conference [C]. 1998:539 - 546.

[7] Henry Chaug. A Top_down Constraint_driven Design Methodology for Analog Integrated Circuits. KLUWER ACADEMIC PUBLISHERS, Boston/ London/ Dordrecht, 1997:97.

[8] M Zwoliński, C Garagate, Zmrcarica, T J Kazmierski, A D Brown. Anatomy of a simulation backplane [J]. IEE Proc.-Comput. Digit. Tech, 1995, 142(6):377 - 385.

作者简介:

肖立伊　女,1961 年生于湖南省,哈尔滨工业大学副教授,主要从事教学和科研,研究方向是高层次模拟和混合信号电路与系统的建模和仿真.

叶以正　教授,博士生导师,主要从事 ASIC 设计方法学和 EDA 技术的研究工作.