发,LUSTRE 使用';'表示不同节点的并行执行,SIGNAL 中不同进程间的组合操作'|'也隐含着并行执行[32].同时,在分布式系统上,由于进程间通信受到物理载体的影响而产生不可预测的延迟[13],导致不同进程间的时钟可能不满足同步关系,被称为全局异步局部同步 GALS 系统.

文献[14]提出一种非侵入式的多线程代码生成方法,即在不改变已有 SINGAL 编译器 Polychrony 结构的基础上,利用其串行编译功能生成多个独立的进程,并基于主控函数完成调度,实现进程之间的并行执行.文献[15]提出从实际工业需求中导出功能行为等价的 Heptagon 程序[33](一种类 Lustre 同步语言)并进一步生成多线程代码.与文献[14]所提方案类似,首先基于 Heptagon 编译器生成多个串行节点,然后在集成规约中描述任务间通信和同步.通过构建节点间的依赖图,利用串行调度来保证确定性并发语义.除功能需求外,该研究还考虑了非功能性需求(如周期等实时性质)的代码生成.

此外,文献[34]提出从高层建模语言(AADL,SysML,SystemC 等)转换为 SIGNAL 同步程序并进一步生成多线程代码的方法.其中,同步程序基于 Weekly Endochronous 理论[35]:即程序中允许多个根时钟.在多线程生成过程中,采用符号化原子表达式和合并分支条件的策略约减线程数目.不过,其同步程序中仅考虑有限的数据类型(事件、布尔和枚举类型)和布尔操作(相等和取反操作).

多个进程之间并行划分的"粒度"相对较大[14],但在同步语言模型中,单个进程内部也往往存在潜在的并行执行信息.尤其是可以自然表达分布式系统的 SIGNAL 语言,其语法的基本结构中包含多时钟操作,可以比较自然地支持在进程内部描述并行执行操作.

(2) 基于单进程的多线程代码生成

文献[26]中介绍了同步语言 SIGNAL 编译器 Polychrony 中的代码生成策略,在多线程生成方面分为静态调度和动态调度.静态调度下的分簇(多线程)代码生成:将生成的代码划分到簇中来模拟调度图的结构.分簇的原则是:将调度图划分为对那些依赖于完全相同的输入集合进行计算.只要一个簇的所有输入都有确定值,那么这个簇就可以被自动执行,一个簇的时钟取决于簇中所有信号在时钟树上的公共父节点.动态调度下的分簇(多线程)代码生成:按任务实现分簇,并生成任务之间的同步信息.任务由簇来生成,主要思想是使用信号量的方式并行执行各个任务.然而,这种方式下生成的多线程数目太多,并且使用特定的线程库来生成仿真代码.

文献[29,30]是对纯同步语言 Quartz 编译过程的多线程代码的研究.文献[29]提出同步卫式动作(clocked guarded action,简称 CGA)到基于 OpenMP 的多线程 C 程序的编译方案.作者提出"垂直划分"的概念,即从数据依赖图中抽取出独立的线程,并生成对应的 C 代码.文献[30]引入软件流水线的概念,提到的软件流水线方法包含:分析卫式动作之间的依赖关系、创建流水线变量存储各阶段之间的中间结果以及卫式动作到流水线之间的转换.这种方法被称为"水平划分".

文献[31]对 SIGNAL 多线程代码生成进行了研究,提出基于方程依赖图(equation dependency graph,简称 EDG)的 OpenMP 并行代码生成方法.其中的并行代码生成算法主要是对数据依赖图进行拓扑排序,生成多个链表:多个链表之间顺序执行、单个链表内部节点并行执行.不过,文献[31]使用拓扑排序作为任务划分算法,没有考虑优化工作,而且需要在划分之后对划分结果单独加入时钟信息.而本文使用的 CDDG 中包含时钟依赖关系,即在划分过程中已经考虑时钟信息.

本文主要考虑单进程的多线程代码生成方法,同时给出优化策略和流水线方式的任务划分方法,能够较好地约减线程数目以及提高目标多线程代码执行效率.

(3) 多核架构执行及 WCET 分析

ITEA 3 计划中的(affordable safe & secure mobility evolution,简称 ASSUME)项目[15,36]提出一种用于在多核/众核架构上提供可信的嵌入式软件工程方法,包括从 LUSTRE 同步模型到自动生成并行代码并在多核/众核平台上执行.ASSUME 项目扩展了 SCADE 工具的 KCG 编译器,以支持生成面向 MPPA 众核架构[37]的并行代码,但是当前的并行信息是通过用户来驱动的,即用户指定程序中并行执行区域.此外,在代码生成过程中提供每个任务在实时调度期间所需的属性(如 WCET、内存访问的最坏次数等).

法国国家航天航空研究中心 ONERA 的 SchedMCore 环境[38,39]为形式化多处理器调度分析和实验性多核

运行时基础架构提供了一套验证和执行工具.SchedMCore将同步语言Prelude[40]编译生成一组相互通信的周期任务(包含信息有:任务的周期、WCET、首次到达时间和 Deadline),并基于工具自带的不同多核调度策略实现任务在多核架构上的调度.文献[41]中给出一个基于 SchedMCore 设计并运行在多核/众核架构上的航空经度控制器案例.

德国 Kaiserslautern 工业大学嵌入式系统小组(http://es.cs.uni-kl.de/)基于同步语言 QUARTZ 开发了一个用于并行嵌入式系统的规约、验证和实现的框架 AVEREST(http://www.averest.org/).该框架可同时生成硬件电路逻辑代码和多线程代码,并且支持软硬件分析[42].此外,他们还研究了基于指令集并行(instruction level parallelism,简称 ILP)的同步语言 SCAD 最优化代码生成,包括使用回答集编程(answer set programming,简称 ASP)处理 SCAD 代码生成过程中的最优资源/时间约束调度问题[43],以及利用 SMT 求解器生成实现最大化指令级并行(给定处理器单元数目)的最优化代码[44]等.

综上所述,同步语言多线程代码生成涉及到多核硬件执行平台以及考虑 WCET 分析.我们在已有工作[18]中给出在同步语言代码生成过程中的时间可预测多核体系结构模型及软硬件映射方法.而本文主要侧重同步语言多线程代码生成方法及其实现,提出任务划分优化策略以及基于流水线方式的划分以提高目标代码的执行速度.

6 总结与展望

同步语言广泛用于安全关键嵌入式系统建模与验证,近年来,同步语言的多线程代码生成成为学术界的一个研究热点.本文提出一种基于 OCAML 的同步语言 SIGNAL 多线程代码生成方法和工具.首先将 SIGNAL 程序转换为经过时钟演算的 S-CGA 中间程序;之后将 S-CGA 中间程序转换为时钟数据依赖图以分析依赖关系;然后对时钟数据依赖图进行拓扑排序划分,并针对划分结果提出优化算法和基于流水线方式的任务划分方法;最后将划分结果转换为虚拟多线程结构,然后进一步生成可执行多线程 C/Java 代码.通过在多核处理器上的实验验证了本文提出方法的有效性.

编译器的形式化验证是一项重要工作,例如:CompCert 编译器(C 编译器)[45]、清华大学 L2C 项目(LUSTRE 编译器)[27,28]以及 Vélus 编译器(LUSTRE 编译器)[46].我们已经给出 SIGNAL 多线程代码生成器前端的语义保持证明,下一步工作将基于 Coq 完成代码生成器后端的语义保持证明.其次,扩展同步语言以支持描述实时性质,以及考虑在时间可预测多核处理器(如 Patmos[47])上执行多线程代码并进行 WCET 分析也是未来一项重要工作;最后,针对复杂嵌入式系统,我们正在开展嵌入式实时系统体系结构分析与设计语言 AADL(描述系统架构)[48]和同步语言(描述 AADL 单构件内的功能)的混合建模方法研究.

致谢 感谢匿名评审专家给予的宝贵意见.另外,感谢法国法国国家信息与自动化研究所(INRIA)Jean-Pierre Talpin 教授给予了很多重要的建议.

References:

[1] Leveson N. Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press, 2011.

[2] Parkinson P. Safety, security and multicore. In: Advances in Systems Safety. London: Springer-Verlag, 2011. 215–232.

[3] Quad-core LEON4 next generation microprocessor evaluation board. 2014. http://www.gaisler.com/index.php/products/boards/gr-cpci-leon4-n2x

[4] RTCA DO-178C. Software considerations in airbone systems and equipment certification. Report, RTCA, 2011. [doi: 10.1145/1869542.1869558]

[5] RTCA DO-331. Model-based development and verification supplement to DO-178C and DO-278A. Report, RTCA, 2011.

[6] RTCA DO-333. Formal methods supplement to DO-178C and DO-278A. Report, RTCA, 2011.

[7] Boussinot F, de Simone R. The Esterel language. Proc. of the IEEE, 1991,79(9):1293–1304.

[8] Halbwachs N, Caspi P, Raymond P, Pilaud D. The synchronous data-flow programming language Lustre. Proc. of the IEEE, 1991, 79(9):1305–1320.

[9] SCADE. http://www.esterel-technologies.com/products/scade-suite/

[10] Benveniste A, Le Guernic P, Jacquemot C. Synchronous programming with events and relations: The signal language and its semantics. Science of Computer Programming, 1991,16:103–149.

[11] Schneider K. The synchronous programming language QUARTZ. Internal Report, Department of Computer Science, University of Kaiserslautern, 2010.

[12] Sovani S. Simulation accelerates development of autonomous driving. ATZ Worldwide, 2017,119(9):24–29.

[13] Doucet F, Menarini M, Krüger IH, *et al*. A verification approach for GALS integration of synchronous components. Electronic Notes in Theoretical Computer Science, 2006,146(2):105–131.

[14] Jose BA, Patel HD, Shukla SK, Talpin JP. Generating multi-threaded code from polychronous specifications. Electronic Notes in Theoretical Computer Science, 2009,238(1):57–69.

[15] Souyris J, Didier K, Potop D, *et al*. Automatic parallelization from lustre models in avionics. In: Proc. of the ERTS2 2018, the 9th European Congress Embedded Real-Time Software and Systems. 2018. 1–4.

[16] Yang Z, Bodeveix JP, Filali M. Towards a simple and safe objective CAML compiling framework for the synchronous language SIGNAL. Frontiers of Computer Science, 2018, 1–20.

[17] Yang Z, Bodeveix JP, Filali M, *et al*. Towards a verified compiler prototype for the synchronous language SIGNAL. Frontiers of Computer Science, 2016,10(1):37–53.

[18] Yang ZB, Zhao YW, Huang ZQ, Hu K, Ma DF, Bodeveix JP, Filali M. Time-predictable multi-threaded code generation with synchronous languages. Ruan Jian Xue Bao/Journal of Software, 2016,27(3):611−632 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4984.htm [doi: 10.13328/j.cnki.jos.004984]

[19] Gamatié A. Designing Embedded Systems with the SIGNAL Programming Language. Springer-Verlag, 2010.

[20] Besnard L, Gautier T, Le Guernic P. SIGNAL V4 Reference Manual. 2010.

[21] Pnueli A, Siegel M, Singerman F. Tanslation validation. In: Proc. of the TACAS 98. 1998. 151–166.

[22] Yang Z, Bodeveix JP, Filali M. A comparative study of two formal semantics of the SIGNAL language. Frontiers of Computer Science, 2013,7(5):673–693.

[23] The Coq Proof Assistant. https://coq.inria.fr/about-coq

[24] Brandt J, Gemunde M, Schneider K, Shukla SK, Talpin J-P. Embedding polychrony into synchrony. IEEE Trans. on on Software Engineering, 2013,39(7):917–929.

[25] Brandt J, Gemunde M, Schneider K, Shukla SK, Talpin J-P. Integrating system descriptions by clocked guarded actions. In: Proc. of the FDL. IEEE, 2011. 1–8.

[26] Besnard L, Gautier T, Talpin JP. Code generation strategies in the Polychrony environment [Ph.D. Thesis]. INRIA, 2009.

[27] Shi G, Wang SY, Dong Y, Ji ZY, Gan YK, Zhang LB, Zhang YC, Wang L, Yang F. Construction for the trustworthy compiler of a synchronous data-flow language. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):341−356 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4542.htm [doi: 10.13328/j.cnki.jos.004542]

[28] Liu Y, Gan YK, Wang SY, Dong Y, Yang F, Shi G, Yan X. Trustworthy translation for eliminating high-order operation of a synchronous dataflow language. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):332−347 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4785.htm [doi: 10.13328/j.cnki.jos.004785]

[29] Baudisch D, Brandt J, Schneider K. Multithreaded code from synchronous programs: Extracting independent threads for OpenMP. In: Proc. of the Design, Automation & Test in Europe Conf. & Exhibition (DATE). IEEE, 2010. 949–952.

[30] Baudisch D, Brandt J, Schneider K. Multithreaded code from synchronous programs: Generating software pipelines for OpenMP. In: Proc. of the MBMV. 2010. 11–20.

[31] Hu K, Zhang T, Shang LH, Yang ZB, Talpin JP. Parallel code generation from synchronous specification. Ruan Jian Xue Bao/Journal of Software, 2017,28(7):1698−1712 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5056.htm [doi: 10.13328/j.cnki.jos.005056]

[32] Le Guernic P, Talpin JP, Le Lann JC. Polychrony for system design. Journal of Circuits, Systems, and Computers, 2003,12(3): 261–303.

[33] The Heptagon language and compiler. 2018. http://heptagon.gforge.inria.fr

[34] Papailiopoulou V, Potop-Butucaru D, Sorel Y, De Simone R, Besnard L, Talpin JP. From design-time concurrency to effective implementation parallelism: The multi-clock reactive case. In: Proc. of the Electronic System Level Synthesis Conf. (ESLsyn). 2011. 1–6.

[35] Potop-Butucaru D, Caillaud B, Benveniste A. Concurrency in synchronous systems. Formal Methods in System Design, 2006,28(2): 111–130.

[36] ITEA 3 programme 14014 ASSUME project. 2018. https://itea3.org/project/assume.html

[37] De Dinechin BD, Van Amstel D, Poulhiès M, *et al*. Time-critical computing on a single-chip massively parallel processor. In: Proc. of the Conf. on Design, Automation & Test in Europe. European Design and Automation Association, 2014. 97.

[38] SchedMCore User's Manual. 2018. http://sites.onera.fr/schedmcore/sites/sites.onera.fr.schedmcore/files/schedmcore-user-manual.pdf

[39] Cordovilla M, Boniol F, Forget J, *et al*. Developing critical embedded systems on multicore architectures: The Prelude-SchedMCore toolset. In: Proc. of the 19th Int'l Conf. on Real-time and Network Systems. 2011.

[40] Forget J, Boniol F, Lesens D, *et al*. A multi-periodic synchronous data-flow language. In: Proc. of the 11th IEEE High Assurance Systems Engineering Symp., HASE 2008. IEEE, 2008. 251–260.

[41] Pagetti C, Saussié D, Gratia R, *et al*. The ROSACE case study: From simulink specification to multi/many-core execution. In: Proc. of the 20th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2014. 309–318.

[42] Schneider K, Brandt J. Quartz: A synchronous language for model-based design of reactive embedded systems. In: Handbook of Hardware/Software Codesign. 2017. 29–58.

[43] Dahlem M, Bhagyanath A, Schneider K. Optimal scheduling for exposed datapath architectures with buffered processing units by ASP. 2018. [doi: 10.1017/S1471068418000170]

[44] Bhagyanath A, Schneider K. Exploring the potential of instruction-level parallelism of exposed datapath architectures with buffered processing units. In: Proc. of the Int'l Conf. on Application of Concurrency to System Design. IEEE, 2017. 106–115.

[45] Leroy X. Formal verification of a realistic compiler. Communications of the ACM, 2009,52(7):107–115.

[46] Bourke T, Brun L, Dagand PÉ, *et al*. A formally verified compiler for Lustre. ACM SIGPLAN Notices, 2017,52(6):586–601.

[47] Schoeberl M, Silva C, Rocha A. T-CREST: A time-predictable multi-core platform for aerospace applications. In: Proc. of the Data Systems In Aerospace (DASIA 2014). 2014.

[48] Yang ZB, Pi L, Hu K, Gu ZH, Ma DF. AADL: An architecture design and analysis language for complex embedded real-time systems. Ruan Jian Xue Bao/Journal of Software, 2010,21(5):899−915 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3700.htm [doi: 10.3724/SP.J.1001.2010.03700]

附中文参考文献:

[18] 杨志斌,赵永望,黄志球,胡凯,马殿富,Bodeveix JP,Filali M.同步语言的时间可预测多线程代码生成方法.软件学报,2016,27(3): 611−632. http://www.jos.org.cn/1000-9825/4984.htm [doi: 10.13328/j.cnki.jos.004984]

[27] 石刚,王生原,董渊,嵇智源,甘元科,张玲波,张煜承,王蕾,杨斐.同步数据流语言可信编译器的构造.软件学报,2014,25(2):341−356. http://www.jos.org.cn/1000-9825/4542.htm [doi: 10.13328/j.cnki.jos.004542]

[28] 刘洋,甘元科,王生原,董渊,杨斐,石刚,闫鑫.同步数据流语言高阶运算消去的可信翻译.软件学报,2015(2):332−347. http://www.jos.org.cn/1000-9825/4785.htm [doi: 10.13328/j.cnki.jos.004785]

[31] 胡凯,张腾,杨志斌,Jean-Pierre Talpin.面向同步规范的并行代码自动生成方法研究.软件学报,2017,28(7):1698−1712. http://www.jos.org.cn/1000-9825/5056.htm [doi: 10.13328/j.cnki.jos.005056]

[48] 杨志斌,皮磊,胡凯,顾宗华,马殿富.复杂嵌入式实时系统体系结构设计与分析语言:AADL.软件学报,2010,21(5):899−915. http://www.jos.org.cn/1000-9825/3700.htm [doi: 10.3724/SP.J.1001.2010.03700]

杨志斌(1982－),男,江西吉安人,博士,副教授,CCF 专业会员,主要研究领域为安全关键嵌入式软件,形式化方法.

袁胜浩(1994－),男,学士,CCF 学生会员,主要研究领域为形式化方法,软件工程.

谢健(1988－),男,硕士生,CCF 专业会员,主要研究领域为形式化方法,系统安全性分析,服务计算.

周勇(1975－),男,博士,副教授,CCF 专业会员,主要研究领域为形式化方法,软件工程.

陈哲(1981－),男,博士,副教授,CCF 专业会员,主要研究领域为形式化方法,软件工程,软件验证.

薛垒(1982－),男,高级工程师,主要研究领域为嵌入式软件设计验证.

Bodeveix Jean-Paul(1963－),男,博士,教授,博士生导师,主要研究领域为实时系统,形式化方法.

Filali Mamoun(1957－),男,博士,高级研究员,博士生导师,主要研究领域为实时系统,形式化方法.