基于 CSP 的构件化嵌入式软件能耗分析与评估方法研究

张滕滕 吴 晓 李长德 董云卫
(西北工业大学计算机学院 西安 710072)

摘 要 随着嵌入式系统的发展,构件化软件开发技术已成为嵌入式软件开发的发展趋势.嵌入式系统通常是能源有限系统,如何在构件化嵌入式系统开发前期对其能耗进行分析与评估,发现系统能耗特性设计缺陷,从而提高开发效率,降低开发成本,已成为嵌入式系统设计的一个挑战.文中从构件化嵌入式软件体系结构出发,采用基于路径的系统能耗分析评估方法,在嵌入式系统架构设计阶段对其能耗特性进行分析与评估.在此评估体系中,软件体系结构应用进程代数语言 CSP 进行形式化描述,能耗特性在构件接口级别定义,最终建立了以 CSP 迹模型为基础的基于路径的系统能耗分析评估模型.文章最后通过案例分析验证了该模型分析方法的正确性和有效性.

关键词 构件;接口;体系结构;能耗;迹模型
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2009.01876

On Energy-Consumption Analysis and Evaluation for Component-Based Embedded System with CSP

ZHANG Teng-Teng WU Xiao LI Chang-De DONG Yun-Wei
(*College of Computer Science, Northwestern Polytechnical University, Xi'an* 710072)

Abstract With the rapid developments of embedded systems technology, the use of Component-Based Software Development (CBSD) in embedded systems is on the rise. Embedded systems are usually energy constrained, so it has become a great challenging issue to analyze the system energy-consumption at early phase of component-based embedded software development, which can help to improve the efficiency and reduce the costs. This paper consequently contributes to this problem. According to the component-based embedded software architecture, the authors analyze the energy-consumption of embedded system using path-based method in architecture design phase. In our evaluation model, the formal specification of software architecture is given based on CSP and the energy-consumption description is established on the interface-level. Finally, the authors establish a path-based energy-consumption evaluation model through CSP traces model. A case study is presented to validate the correctness and the effectiveness of our method in the end of this paper.

Keywords component; interface; architecture; energy consumption; trace model

1 引 言

随着嵌入式系统的发展,嵌入式系统的规模与复杂度越来越高,对系统的开发成本、开发周期及非功能特性的要求也越来越严格.为了提高嵌入式系统的开发效率、降低开发成本、提高系统开发质量以及系统的可靠性等非功能特性,构件化软件开发方

收稿日期:2009-04-14;最终修改稿收到日期:2009-07-27.本课题得到国家自然科学基金重点项目(60736017)、国家"八六三"高技术研究发展计划重大专项课题(2007AA010304)资助.**张滕滕**,男,1984 年生,硕士研究生,主要研究方向为嵌入式软件设计与验证.E-mail:ztt7588@163.com.**吴 晓**,女,1958 年生,硕士,副教授,研究方向为网络化嵌入式系统.**李长德**,男,1982 年生,博士研究生,研究方向为嵌入式软件设计与验证.**董云卫**,男,1968 年生,博士,教授,研究领域为嵌入式软件设计与验证.

法被引入到嵌入式领域.构件化软件开发方法利用现有的、成熟可靠的构件资源开发系统,可以提高软件资源的复用度和系统的可靠性,已成为未来嵌入式软件的开发趋势.

嵌入式系统大多使用有限容量的电源提供能量供应,传统的能耗分析评估方法是在系统实现后通过仿真或测试获得系统运行时的能耗性质,进而对能耗进行分析评估.为了提高系统开发效率,节约系统开发成本,避免在开发后期因能耗问题而重新设计开发,人们期望能够在系统设计阶段通过系统抽象模型就可以比较准确地了解系统在运行时的最大、最小、平均能耗等相关的能耗性质,从而尽早发现问题并对系统结构进行调整.基于体系结构的嵌入式软件能耗分析与评估技术成为了满足这一需求的技术手段.当前在软件体系结构级别上对嵌入式软件能耗进行分析评估主要采用基于路径的方式,将系统的执行路径与路径中各节点的能耗属性相结合进行能耗的分析.系统的执行路径主要通过两种方式获取:一种是通过对系统控制流图的分析而获得系统执行路径[1];另一种是根据系统交互行为的形式化规格描述,按照一定的规则推导而获得系统执行路径[2].控制流图比较直观,但通常无法对系统交互行为进行精确描述,而且随着状态的增加存在状态空间爆炸问题;而形式化方法通常能够精确地描述系统,但由于嵌入式系统的复杂性,还要求形式化方法具有较强的表达能力,并可以便捷地给出系统的执行路径.

针对以上问题,本文采用进程代数语言 CSP (Communicating Sequential Processes)[3-4] 对嵌入式软件体系结构进行描述.CSP 作为一种形式化建模语言,已经广泛地应用于软件的行为建模.它以进程的方式刻画系统的动态行为,具有强大的描述能力.此外,CSP 已经有自动化的验证工具 FDR 支持,并且使用 FDR 可以对系统中的死锁、活锁等安全问题进行自动化的验证.本文应用 CSP 对构件以及连接件的动态行为进行规格化描述,然后由构件与连接件的并发组装得到整个系统的动态行为,借助系统动态行为的 CSP 进程描述得出相应的 CSP 的迹模型,结合接口级事件的能耗特性,得出系统的能耗模型,并根据不同的系统模型给出相应的能耗分析评估方法.

2　相关研究

嵌入式软件的能耗问题已经引起相关研究领域研究者的广泛关注,但目前大多数研究还是停留在指令级、代码级的能耗分析与估算[5-7],而在更高的抽象级别的能耗研究相对较少.基于体系结构的系统能耗分析已成为一个研究热点.文献[1]提出了一种使用基于特性的宏模型来分析计算软件能耗的方法,该方法以目标处理器能耗模型来刻画函数级能耗,然后根据函数刻画整个软件的宏模型从而对软件的能耗进行评估.文献[2]描述了一种基于进程代数的形式化框架,对能源受限的实时系统进行建模与分析.文献[8]提出了一种体系结构转换的方法,从体系结构的角度对嵌入式软件的能耗进行优化.文献[9]针对具体的体系结构建模语言 AADL(the Architecture Analysis & Design Language)研究了基于 AADL 模型的嵌入式系统的能耗评估精化方法,但是 AADL 是一种半形式化的建模语言,其描述具有二义性.

3　理论基础

CSP 是一种进程代数描述语言,它将系统刻画为以事件为基本元素的进程,即进程是系统行为模式的描述,每个系统都可以看作一个进程.进程中所有事件的集合称为进程的字母表,如进程 P,它的字母表可以表示为 αP. CSP 的基本操作及其语义请参阅文献[3-4].由于一个系统的运行完全可以通过系统的执行路径来观察,为了从系统的执行路径上分析系统的能耗特性,我们采用 CSP 的 Trace Model 来描述系统的执行路径.下面首先介绍进程迹的概念.

进程的迹(traces). 进程 P 到某个时刻为止的行为的顺序记录称为进程 P 的迹,进程 P 迹的集合用函数 $traces(P)$ 表示[3]:

$$traces(P)=\{\langle\rangle,\langle a_1\rangle,\langle a_1,a_2\rangle,\cdots,\langle a_1,a_2,\cdots,a_n\rangle\}$$
$$\text{with } a_i\in(\alpha P\cup\{\surd\})\wedge a_n=\surd,$$

其中集合中的每一个元素都是进程 P 的迹. $traces(P)$ 可以由以下规则推导获得:

$traces(STOP)=\{\langle\rangle\}$;

$traces(SKIP)=\{\langle\rangle,\langle\surd\rangle\}$;

$traces(a\rightarrow P)=\{\langle\rangle\}\cup\{\langle a\rangle^\frown s\mid s\in traces(P)\}$;

$traces(x:A\rightarrow P)=$
$\{\langle\rangle\}\cup\{\langle a\rangle^\frown s\mid a\in A\wedge s\in traces(P[a/x])\}$;

$traces(c?x:A\rightarrow P)=$
$\{\langle\rangle\}\cup\{\langle c.a\rangle^\frown s\mid a\in A\wedge s\in traces(P[a/x])\}$.

对于 CSP 中所有二元操作 op 存在对应的迹操

作 op_{trace} 满足：

$$traces(P\ op\ Q)=traces(P)\ op_{trace}\ traces(Q).$$

关于 traces model 的具体细节请参阅文献[4].

4 系统体系结构模型及能耗描述

4.1 体系结构模型

软件体系结构是一个抽象的系统范畴，主要包括用其行为来表征的功能构件以及构件间的相互连接. 本文将构件间的连接定义为连接件，即用连接件实现构件间的连接，并且将连接件看作是与构件同级的一阶实体[10]，因此连接件也可以看作是连接构件. 一个构件可以有多个接口，每个接口对应 CSP 进程的一个事件，构件的动态行为通过以接口为事件的 CSP 进程来表示. 连接件通过描述构件间接口的交互行为实现构件间的交互. 下面我们将从接口、构件\连接件、系统 3 个层次对我们的系统体系结构进行描述.

接口. 在本文的体系结构模型中，接口被定义为四元组，即 $I=(Type,Signature,Functionality,Properties)$，其中 $Type$ 用于描述接口的类型，表示接口是请求服务还是提供服务；$Signature$ 描述了接口的型构，描述接口元素的语法信息，即如何定义和使用此接口；$Functionality$ 是接口的功能属性规约，在语义级别上刻画接口的功能；$Properties$ 是接口的非功能属性规约，刻画接口的能耗、实时性、可靠性、等非功能约束，在 4.2 节我们将通过能耗的描述给出其描述形式.

构件. 一个构件定义为一个二元组：$C=(I,DB)$，其中 I 代表构件中接口的集合，包括服务接口的集合和请求接口的集合. DB 是构件接口交互的动态行为规约，刻画了构件与外界环境交互的行为规则，如上所述，我们用 CSP 进程描述构件的 DB，例如：

$$DB(C)=a\rightarrow b\rightarrow DB(C).$$

连接件. 一个连接件也对应一个二元组：$L=(Roles,Glue)$，其中 $Roles$ 表示参与交互的构件接口在连接件中所对应的角色(Role)的集合，Role 的描述形式同接口一样，每个 Role 对应一个构件接口. $Glue$ 是胶合进程，描述了各个 Role 如何协调工作来实现构件间的交互. 我们同样用 CSP 进程描述连接件的胶合进程，例如：

$$Glue=C_1.a\rightarrow C_2.b\rightarrow Glue.$$

系统. 一个系统是一个五元组：$S=(I,SubC,Mapping,IDB,ODB)$，其中 I 表示系统向外界提供的接口的集合，包括提供服务接口和请求服务的接口；$SubC$ 是组成系统的构件实例、连接件实例的集合，即子构件的集合；$Mapping$ 是子构件间接口的映射关系和系统接口与部分子构件接口的映射关系的集合，映射的语义参照文献[10]中的 Attachment；IDB 是系统的子构件经过并发组装后的系统动态行为，即系统的内部动态行为；ODB 为系统对外界表现出来的动态行为，即系统的外部动态行为，它的字母表为 I，设系统 S 的内部事件集为 A，则 $ODB(S)=IDB(S)\backslash A$. 在对系统的非功能特性进行分析评估时我们通常通过 $IDB(S)$ 来分析.

4.2 能耗描述

在基于构件的软件开发方法中，构件接口是构件与外界交互的唯一通道. 因此，构件间的交互可以描述为以构件接口为基本元素的事件序列. 传统的能耗分析方法，将能耗特性建立在构件级别的实体上，然而一个构件可能有多个接口，而且对于不同的应用场景，构件的每个接口的使用频率可能不同，这就导致不同场景中同一个构件的能耗不同，因此以构件为基本能耗单元(即将能耗特性建立在构件级别上)进行系统的能耗分析评估，可能造成较大的误差. 鉴于此，本文将能耗特性的描述建立在接口级别上. 在体系结构模型的构件接口的 $Properties$ 字段中，我们定义一个 $energy$ 特性以表明接口的能耗，其具体形式如下：

Properties：

$energy=\langle energy\text{-}consume\text{-}expression\rangle$

… //其它非功能特性

End Properties

能耗表达式($energy\text{-}consume\text{-}expression$)可以是与时间或问题复杂度等因素相关的函数，或者是一个固定的数值，例如：

$$energy=C\times t,$$

其中，C 是能耗率，t 是执行时间. 能耗表达式的具体形式可以根据具体的应用来确定. 连接件中的 Role 的能耗值描述同接口中的能耗描述方式相同. 由于在构件、连接件并发组装后，接口和对应的 Role 在组装后的 CSP 进程的迹中表现为一个事件，该事件的能耗等于相应的接口的能耗加上 Role 的能耗.

通常嵌入式软件系统的能耗是与具体的硬件平台相关的，不同硬件平台上的相同软件能耗往往不同，因此接口级事件的能耗必须与相应的硬件平台绑定. 为此本文针对特定的硬件平台，对嵌入式软件

系统的能耗进行描述分析,同时为了方便描述,省略了相应的硬件绑定信息.

5　体系结构级能耗分析

在系统体系结构级能耗分析中,首先根据 CSP 规则[3-4]从系统的动态行为中推导出系统的迹模型;然后结合接口能耗信息与系统迹模型计算出相应的系统能耗.本节将给出基于迹的能耗模型,并根据系统体系结构的具体特征给出相应的能耗分析方法.

5.1　基于迹的能耗模型

设系统某个执行过程的迹为 tr, tr 中共执行了 n 个事件 $a_1,a_2,\cdots,a_n$,每个事件的能耗为 e_i,每个事件的执行次数为 c_i,其中 $i=1,2,\cdots,n$,则系统在该执行过程的能耗为

$$E(tr)=\sum_{i=1}^{n}e_i\times c_i \tag{1}$$

对于仅与外界环境交互的接口(这类接口不参与构件间交互)所对应迹中事件的能耗定义为该接口的能耗;参与构件间交互的接口所对应迹中事件的能耗定义为该接口的能耗和与其相对应的连接件 Role 的能耗之和,即

$$e_i=e(Interface_i)+e(Role_i) \tag{2}$$

其中函数 $e()$用于获取接口或 Role 的能耗.

由于进程 $SKIP$ 在 CSP 中是一个辅助进程,用于标明一个成功运行完毕的进程,因此这里我们定义进程 $SKIP$ 的能耗为 0,即事件√的能耗为 0.此外,为了方便描述系统一次执行的迹,本文定义了一个特殊的事件 $Start$,用于表示系统每次运行时所执行的第一个事件,并规定对每个系统所对应的 CSP 进程都添加一个 $Start$ 事件,表示系统每次运行都是从 $Start$ 事件开始,即设 P 为系统 S 所对应的 CSP 进程,通过添加 $Start$ 事件后 S 所对应的进程为

$$SP=Start\rightarrow P.$$

由于 $Start$ 事件在系统体系结构描述中是不存在的,它所执行的操作是一个空操作,因此本文规定 $Start$ 事件的能耗为 0.

在现实世界中存在下面两类系统:一类是可以在有限的时间内成功终止的系统;一类是无限运行的不可终止的系统.在此将以上两类系统分别称作可终止系统和无限循环系统.在系统能耗分析时,对于第一类系统,通常关注系统一次运行的能耗及系统完成某项功能所消耗的能量;对于无限运行系统,除了关注系统提供某项服务的能耗外,还需要关注系统执行到某个时刻已消耗的能量以及能耗率等问题.下面分别对这两类系统的能耗问题进行讨论.

5.2　可终止系统能耗分析

这类系统可以在有限的时间内终止,即通常表现为不存在无限循环的系统,如图 1 所示.

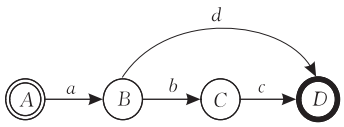

图 1　可终止系统

该类系统通常是由某个初始事件开始,运行到某个终止事件(通常用 $SKIP$ 或 $STOP$ 进程描述)结束.有些系统可能有多个初始事件和多个终止事件,多个终止事件通常表现为 $SKIP$ 或 $STOP$ 进程在系统的 CSP 进程中出现多次,由于我们通常关心系统正常终止时的系统特性,因此本文规定每个系统都以 $SKIP$ 进程终止,即以事件"√"终止.在具有多个初始事件的系统进程中,通过添加 $Start$ 事件将多个初始事件转化为一个唯一的初始事件 $Start$.此外,为了统一描述系统的执行路径,在只有一个初始事件的系统进程中也添加一个 $Start$ 事件表示系统开始运行.因此一个系统的一次成功的运行就是执行了一个以 $Start$ 事件开始以√事件终止的事件序列,即系统执行的一个迹,这种以 $Start$ 事件开始以√事件终止的迹称之为系统迹,其具体定义如下.

系统迹.系统 S 一次成功运行可能执行的事件序列称为系统 S 的系统迹,记为 str.

根据以上规定,str 必须满足以下条件:① str 的首元素必须为 $Start$ 事件,即 $str_0=Start$;② str 的最后一个元素必须为√事件,即 $\overline{str}_o=\surd$.因此根据式(1)和(2)可以得到系统迹 str 的能耗 $E(str)$.

由于系统内部可能存在选择和不确定的循环等情况,因此系统从初始事件到终止事件可能存在多条路径,即一个系统可能存在多个系统迹.这就引出了系统的最小、最大能耗.

系统最小能耗和最大能耗.设系统 S 共有 n 个系统迹 $str_1,str_2,\cdots,str_n$,则系统 S 的最小能耗为

$$minE(S)=\min(E(str_1),E(str_2),\cdots,E(str_n)) \tag{3}$$

系统 S 的最大能耗为

$$maxE(S)=\max(E(str_1),E(str_2),\cdots,E(str_n)) \tag{4}$$

其中函数 min()、max()分别为取参数中的最小、最大值函数.若用户给出每个系统迹的执行概率,即系统的使用剖面,我们就可以根据系统的使用剖面获得系统的平均能耗.

系统平均能耗.设系统 S 共有 n 个系统迹 str_1, str_2,…,str_n,每个系统迹的执行概率为 $f_i(i=1,2,\cdots,n)$,则系统的平均功耗为

$$avgE(S)=\sum_{i=1}^{n}E(str_i)\times f_i \tag{5}$$

或通过下面式(6)得到

$$avgE(S)=(e_1,\cdots,e_m)\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{pmatrix}\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \tag{6}$$

其中 e_i 为第 i 个事件的能耗,m 为系统中事件的个数,c_{ij} 为第 j 个迹中第 i 个事件发生的次数.

除了对系统的最小、最大、平均能耗验证外,有时还需要验证系统完成某项服务所消耗的能耗,这时可以根据完成该项服务所执行的迹,然后根据式(1)和(2)得到该服务所消耗的能耗,从而与用户需求相比较.

5.3 无限运行系统能耗分析

该类系统通常表现为以下两种:一种经过一系列操作后进入一个无限循环的状态,如图 2(a)所示,这一系列操作本文称之为初始化部分;另一种是整个系统就是一个无限循环系统,如图 2(b)所示.对于第 2 种系统,我们通过添加 *Start* 事件可以转化为第一种系统.通常在没有人为或故障引起终止的情况下这类系统会持续的运行下去,因此当这类系统无限运行下去时它的能耗也趋向于无穷大.对于这类系统,人们通常关心系统某个部分或系统运行到某个时刻的能耗以及系统的能耗率等能耗问题.这里只讨论系统部分的能耗,并对系统执行多次循环的能耗进行讨论.

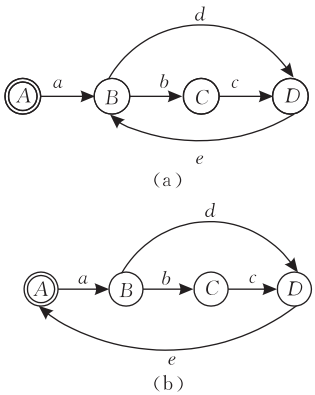

图 2 无限运行系统

这类系统通常表现为具有无限循环的系统,用 CSP 描述就是具有无限递归的进程.对于这类系统,我们关注两个部分的能耗:第一是初始化部分的能耗,即系统经过一系列初始化操作的能耗;第二是循环体的能耗.本文将系统初始化部分所执行的事件序列称作初始化迹,将循环体的一次循环所执行的事件序列称作循环迹,分别记为 itr 和 ltr.因此一个初始化迹或循环迹的能耗可以根据式(1)和(2)得到.对系统部分的能耗验证就是检验某个初始化部分或实现某个功能的循环迹的能耗是否小于等于用户所期望的值.

一般嵌入式系统的初始化迹只有一个,但有些系统可能不止一个,而且不同的初始化迹可能影响后继循环体的执行.下面对这类系统执行多次循环的能耗问题进行讨论,其它情况可视为这类系统的特例.

设系统 S 有 k 个初始化迹 itr_1, itr_2,…,itr_k,r 个循环迹,ltr_1, ltr_2,…,ltr_r, itr_i 执行的概率为 f_i,在执行 itr_i 时执行 ltr_j 的概率为 p_{ij},$i=1,2,\cdots,k$,$j=1,2,\cdots,r$,则系统执行 n 次循环时的平均能耗为

$$avgE(S^n)=\sum_{i=1}^{k}\left(E(itr_i)+n\times\left(\sum_{j=1}^{r}E(ltr_j)\times p_{ij}\right)\right)\times f_i \tag{7}$$

设$\{ltr_{i1},ltr_{i2},\cdots,ltr_{im}\}$为执行 itr_i 后可能执行的循环迹的集合,其中 $m\leq r$,则当系统初始化执行 itr_i 时的 n 次循环的最小、最大能耗分别为

$$minE(S_i^n)=E(itr_i)+\min(E(ltr_{i1}),E(ltr_{i2}),\cdots,E(ltr_{im}))\times n \tag{8}$$

$$maxE(S_i^n)=E(itr_i)+\max(E(ltr_{i1}),E(ltr_{i2}),\cdots,E(ltr_{im}))\times n \tag{9}$$

则系统执行 n 次循环时的最小、最大能耗分别为

$$minE(S^n)=\min(minE(S_1^n),minE(S_2^n),\cdots,minE(S_k^n)) \tag{10}$$

$$maxE(S^n)=\max(maxE(S_1{}^n),maxE(S_2{}^n),\cdots,maxE(S_k{}^n)) \tag{11}$$

6 案例分析

本节以文献[11]中咖啡机控制系统(COFFEE)为例,阐述基于上述能耗模型的能耗分析与预测过程及其相对于以构件为基本能耗单元分析方法的正确性和有效性.在此我们将该系统中每个控件作为构件,并定义以下两个连接件:ButtonContr 连接

件、ContrBox 连接件,如图 3 所示.

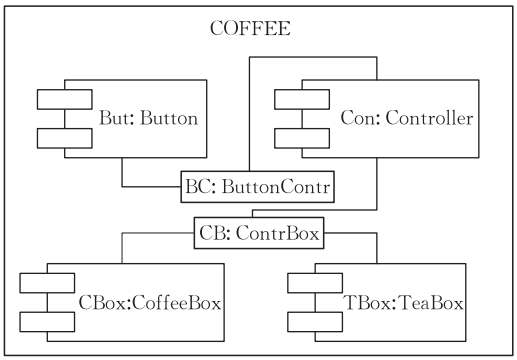

图 3　COFFEE 系统体系结构

图 3 中,ButtonContr 连接件用于实现 Button 构件和 Controller 构件间的连接;ContrBox 连接件用于实现 CoffeeBox 构件、TeaBox 构件与 Controller 构件的连接. 由于该系统中连接件 Role 的能耗很小,本文忽略了各连接件 Role 的能耗. 各构件接口的能耗如表 1 所示.

表 1　COFFEE 系统接口能耗

构件	接口	能耗/mJ
Button	coffee	2.0
	tea	2.1
	next	1.0
Controller	coffee	1.5
	tea	1.5
	cofford	1.5
	teaord	1.2
	coffin	1.6
	teafin	1.5
	next	1.5
CoffeeBox	cofford	2.0
	cupdeliver	2.5
	coffin	4.5
TeaBox	teaord	2.1
	cupdeliver	2.5
	teafin	5.0

各构件的动态行为和各连接件的胶合进程如下所示:

$$DB(\text{Button}) = coffee \rightarrow next \rightarrow DB(\text{Button}) \square tea \rightarrow next \rightarrow DB(\text{Button});$$

$$DB(\text{Controller}) = coffee \rightarrow cofford \rightarrow coffin \rightarrow next \rightarrow DB(\text{Controller}) \square tea \rightarrow teaord \rightarrow teafin \rightarrow next \rightarrow DB(\text{Controller});$$

$$DB(\text{CoffeeBox}) = cofford \rightarrow cupdeliver \rightarrow coffin \rightarrow DB(\text{CoffeeBox});$$

$$DB(\text{TeaBox}) = teaord \rightarrow cupdeliver \rightarrow teafin \rightarrow DB(\text{TeaBox});$$

$$Glue(\text{ButContr}) = Button.cofee \rightarrow Controller.coffee \rightarrow Glue(\text{ButContr}) \square Button.tea \rightarrow Controller.tea \rightarrow Glue(\text{ButContr}) \square Controller.next \rightarrow Button.next \rightarrow Glue(\text{ButContr});$$

$$Glue(\text{ContrBox}) = Controller.cofford \rightarrow CoffeeBox.cofford \rightarrow Glue(\text{ContrBox}) \square Controller.teaord \rightarrow TeaBox.teaord \rightarrow Glue(\text{ContrBox}) \square CoffeeBox.coffin \rightarrow Controller.coffin \rightarrow Glue(\text{ControBox}) \square TeaBox.teafin \rightarrow Controller.teafin \rightarrow Glue(\text{ControBox}).$$

经过实例化及映射后,系统的内部动态行为如下:

$$IDB(\text{COFFEE}) = But: DB(\text{Button}) \parallel Con: DB(\text{Controller}) \parallel CBox: DB(\text{CoffeeBox}) \parallel TBox: DB(\text{TeaBox}) \parallel BC: Glue(\text{ButContr}) \parallel CB: Glue(\text{ContrBox}).$$

根据 CSP 规则可以得出系统的迹模型为

$$traces(\text{COFFEE}) = \{s \mid \exists i,j,k,l,m,n.\ s \leq^{\wedge} / \langle \langle Start \rangle, \langle But.coffee, Con.coffee, Con.cofford, CBox.cofford, CBox.cupdeliver, CBox.coffin, Con.coffin, Con.next, But.next \rangle^{i}, \langle But.tea, Con.tea, Con.teaord, TBox.teaord, TBox.cupdeliver, TBox.teafin, Con.teafin, Con.next, But.next \rangle^{j} \rangle^{k} \vee s \leq^{\wedge} / \langle \langle Start \rangle, \langle But.tea, Con.tea, Con.teaord, TBox.teaord, TBox.cupdeliver, TBox.teafin, Con.teafin, Con.next, But.next \rangle^{l}, \langle But.coffee, Con.coffee, Con.cofford, CBox.cofford, CBox.cupdeliver, CBox.coffin, Con.coffin, Con.next, But.next \rangle^{m} \rangle^{n} \}.$$

通过系统的内部动态行为可以得出,系统 COFFEE 是一个不存在初始化部分的循环系统,根据 CSP 规则得出系统存在以下两个循环迹:

$$ltr_1 = \langle Start, But.coffee, Con.coffee, Con.cofford, CBox.cofford, CBox.cupdeliver, CBox.coffin, Con.coffin, Con.next, But.next \rangle.$$

$$ltr_2 = \langle Start, But.tea, Con.tea, Con.teaord, TBox.teaord, TBox.cupdeliver, TBox.teafin, Con.teafin, Con.next, But.next \rangle.$$

根据式(1)得 ltr_1,ltr_2 的能耗分别为

$$E(ltr_1) = 18.1\text{mJ},\ E(ltr_2) = 18.4\text{mJ}.$$

在本系统中,执行 ltr_1,ltr_2 的概率分别为 58%,42%,则根据式(7)~(11)得出系统执行一次循环的

平均能耗、最小能耗、最大能耗，具体如表 2 所示.

表 2 系统 COFFEE 的平均、最小、最大能耗

	平均能耗/mJ	最小能耗/mJ	最大能耗/mJ
COFFEE[1]	18.226	18.1	18.4

由表 2 的结果可以看出，系统的最小、最大能耗分别等于迹 ltr_1，ltr_2 的能耗. 这是因为该系统没有初始化部分，所以相对于式(7)～(9)中的初始化的迹的能耗为 0mJ，系统在最小、最大能耗时所执行的迹就分别为 ltr_1，ltr_2，即当用户选择咖啡时系统能耗最小，选择茶时系统能耗最大.

根据 CSP 规则和相应接口事件的使用概率，可以得到各个构件的最小、最大、平均能耗，如表 3 所示.

表 3 各个构件的能耗性质

构件名	平均能耗/mJ	最小能耗/mJ	最大能耗/mJ
Button	3.042	3.0	3.1
Controller	5.932	5.7	6.1
CoffeeBox	9.0	9.0	9.0
TeaBox	9.6	9.6	9.6

根据表 1、表 3 和系统的 CSP 进程可知：当选择咖啡时，Button 达到最小能耗值，Controller 达到最大能耗值；CoffeeBox 和 TeaBox 在选择咖啡或茶的情况下所执行的迹是固定且唯一的，所以它们的最小、最大、平均能耗相等.

对系统能耗进行分析时，采用以构件为基本能耗单位，并结合构件迁移方式进行分析，所得的平均能耗为 18.226mJ，这与表 2 中采用接口为基本能耗单位的分析结果相等，而且两者的等价性与表 1 的数据是无关的. 但是当分析最小、最大能耗时，以构件为基本能耗单位的分析将会比较复杂，因为当一个构件处于最小能耗时，它的后继构件并不一定也处于最小能耗，比如在本案例中，当用户选择咖啡时构件 Button 取最小能耗 3.0mJ，而 Controller 取最大能耗值 6.1mJ，显然直接将每个构件的最小值相加以得到系统最小能耗是错误的. 因此要通过构件能耗正确地分析系统的能耗，必须给每个构件划分工作模态，而对于本文的所提的方法却不存在这个问题.

通过以上比较，可以看出采用接口为基本能耗单位的能耗模型分析时比以构件为基本能耗单位的能耗模型分析更有效.

7 总 结

本文在构件化嵌入式软件体系结构的层次上，使用 CSP 进程描述的系统动态行为，避免了非形式化体系结构描述语言所带来的二义性，保证了系统描述的精确性和完备性；建立了以接口为基本能耗单元的能耗模型，解决了以构件为基本能耗单元存在的误差问题；此外，本文还对体系结构中的循环结构以及对相应的能耗分析策略进行了初步讨论. 我们下一步的工作包括：进一步细化完善能耗的分析与评估的方法体系；开发嵌入式软件建模与验证的原型工具.

参 考 文 献

[1] Tan T K, Raghunathan A K, Lakishminarayana G, Jha N K. High-level software energy macro-modeling//Proceedings of the 38th ACM Conference on Design Automation. Las Ueges, USA, 2001: 605-610

[2] Lee I, Philippou A, Sokolsky O. Process-algebraic modeling and analysis of power-aware real-time system. Journal of Computing and Control Engineering, 2002, 13(4): 180-188

[3] Hoare C A R. Communicating Sequential Processes. Englewood Cliffs: Prentice-Hall, 1985

[4] Roscoe A W. The Theory and Practice of Concurrency. Prentice Hall International Series in Computer Science. Prentice Hall, 1997

[5] Tiwari V, Malik S, Wolf A et al. Instruction level power analysis and optimization of software. Journal of VLSI Signal Processing, 1996, 13(2-3): 223-238

[6] Kostas Zotos, Andreas Litke, Alexander Chatzigeorgiou, Spyros Nikolaidis, George Stephanides. Energy complexity of software in embedded systems//Proceeding of IASTED International Conference on Automation, Control and Applications (ACIT-ACA 2005). Novosibirsk, Russia, 2005

[7] Zhao Xia, Guo Yao, Lei Zhi-Yong, Chen Xiang-qun. Estimation and analysis of embedded operating system energe consumption. Acta Electronica Sinica, 2008, 36(2): 209-215 (in Chinese)
(赵霞，郭耀，雷志勇，陈向群. 基于模拟器的嵌入式操作系统能耗估算与分析. 电子学报，2008，36(2)：209-215)

[8] Tan T K, Raghunathan A, Jha N K. Software architectural transformations: A new approach to low energy embedded software//Proceedings of the Design, Automation Test in Europe Conference and Exhibition. Munich Germary, 2003: 1046-1051

[9] Senn Eric, Laurent Johann, Juin Emmanuel, Diguet Jean-Philippe. Refining power consumption estimations in the component based AADL design flow//Proceedings of the IEEE Conference on Specification, Verification and Design Languages, 2008. FDL 2008. Forum on 2008: 173-178

[10] Robert J, Allen R. A formal approach to softwa re architecture [Ph. D. dissertation]. Carnegie Mellon University, Pittsburgh, 1997

[11] Zhao Li-Fang. Formal analysis and application of real time systems based on UPPAL and UML[M. S. dissertation]. Suzhou University, Suzhou, 2008(in Chinese)
(赵丽芳. 基于 UPPAAL 和 UML 的实时系统形式化分析与应用[硕士学位论文]. 苏州大学, 苏州, 2008)

ZHANG Teng-Teng, born in 1984, M. S. candidate. His research interests focus on design and verification of embedded software.

WU Xiao, born in 1958, M. S., associate professor. Her research interests focus on networked embedded system.

LI Chang-De, born in 1982, Ph. D. candidate. His research interests focus on design and verification of embedded software.

DONG Yun-Wei, born in 1968, Ph. D., professor. His research interests include design and verification of embedded software.

Background

The work is supported by National Natural Science Foundation of China under grant No. 60736017 and the National High Technology Research and Development Program (863 Program) of China under grant No. 2007AA010304.

Because most of embedded systems are energy-constrained systems, the energy problem has become a hot topic in the field of embedded systems. At present, researchers have carried out a lot of works on energy consumption of embedded software, however, most of which are based on instruction-level or code-level, and few researches focus on software architecture. There is a team in NPU to carry out energy consumption analysis according to the component-based embedded software architecture with using path-based method which supported by NFC key project and other national projects. In the evaluation model of energy consumption, the energy- consumption formal description is established on the interface-level, and then an energy-consumption evaluation model is establish through CSP traces model to get the of software energy performance in the design phase, and to improve the design according to the elevation results.