学校代号 10532　　学　　号 S151000856

分 类 号 TP301　　密　　级 普通

湖南大學
HUNAN UNIVERSITY

硕士学位论文

能量感知的 ACPS 调度研究

学位申请人姓名 宋金林

培 养 单 位 信息科学与工程学院

导师姓名及职称 李仁发 教授

学 科 专 业 计算机科学与技术

研 究 方 向 嵌入式系统与 CPS

论文提交日 期 2018 年 5 月 8 日

学校代号：10532

学　　号：S151000856

密　　级：普通

湖南大学硕士学位论文

能量感知的 ACPS 调度研究

学位申请人姓名：　宋金林

导师姓名及职称：　李仁发 教授

培　养　单　位：　信息科学与工程学院

专　业　名　称：　计算机科学与技术

论 文 提 交日 期：　2018 年 5 月 8 日

论 文 答 辩日 期：　2018 年 5 月 20 日

答辩委员会主席：　赵欢 教授

Research on Energy-aware Scheduling in Automotive Cyber-Physical Systems

by

SONG Jinlin

B.E. (Hunan University) 2015

A thesis submitted in partial satisfaction of the

Requirements for the degree of

Master of engineering

in

Computer science and technology

in the

Graduate School

Of

Hunan University

Supervisor

Professor Li Renfa

May, 2018

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在_____年解密后适用本授权书。

2、不保密□。

（请在以上相应方框内打“√”）

作者签名：　　日期：　　年　月　日

导师签名：　　日期：　　年　月　日

摘　　要

汽车信息物理系统(Automotive Cyber Physical Systems, ACPS)是 CPS (Cyber Physical Systems)在汽车领域的典型应用，其本质是一个能与物理环境实时交互，计算与网络深度融合的异构分布嵌入式系统。新能源汽车如纯电动汽车是未来汽车工业发展的方向，能量资源将成为 ACPS 设计阶段尤为重要的一项因素，如何实现低能耗并且保证 ACPS 中功能应用的实时性与可靠性是研究的新热点。本文从能量感知的角度出发，在系统能量资源有限的情况下，针对 ACPS 中以 DAG 为模型的分布式功能应用的实时性与可靠性展开研究，采用了动态电压频率调节技术(DVFS)实现节能目的。主要的工作及贡献如下：

（1）本文首先针对 ACPS 中能量约束下的调度长度优化展开研究，所研究问题可分解为能量分配以及调度长度优化两个子问题，提出了能量约束下的高效调度算法(ESECC)。算法采用预分配的方式实现功能应用的能量约束与任务能量约束之间的转换，引入可用能量概念进行未调度任务的能量预分配，改进了现有算法不合理的能量分配所导致的悲观结果。

（2）本文接着针对 ACPS 中能量约束下的可靠性优化展开研究，所研究问题可分解为能量分配、满足截止时间以及可靠性优化三个子问题，提出了能量约束下的功能应用可靠性增强算法(REREC)。REREC 算法在本文提出的 ESECC 算法基础上进行能量的重分配，同时将功能应用的截止时间需求转换成任务的执行时间约束，在能量与时间双重约束下为任务选择高可靠性的处理器。弥补了现有研究单一考虑能量约束或者截止时间需求的不足。

（3）针对本文所提出的算法分别采用大量的对比实验进行验证。通过真实的并行应用验证了 ESECC 算法在满足能量约束下能有效地降低调度长度；通过真实的汽车功能应用以及随机生成的功能应用验证了 REREC 算法在满足能量约束以及不超过应用截止时间的情况下仍然能达到非常高的可靠性。因此本文所提出的算法可以为能量感知的 ACPS 设计提供一定的参考价值。

关键词： ACPS；DVFS；能量感知；异构系统；调度长度；可靠性

Abstract

Automotive Cyber Physical Systems (ACPS) is a typical application of Cyber Physical Systems in the automotive field. In essence, ACPS is a heterogeneous distributed embedded system, which interacts with the physical world in real-time, and integrate computing and network deeply. The new energy automobile such as blade electric vehicles significantly oriented the development of future automotive industry, so energy resourses will become a particularly important factor in the design phase of ACPS. Therefore, how to achieve low power consumption and ensure the real-time and reliability of applications in ACPS is a new research hotspot. For the perspective of energy aware scheduling, this paper studies the real-time and reliability of distributed function modeled by DAG in ACPS under the condition of limited system energy, and adopts Dynamic Voltage and Frequency Scaling (DVFS) technology to achieve the purpose of energy conservation. The main work and contributions are as follows:

(1) This paper first focuses on the research of schedule length optimization under energy constraint in ACPS. The problem is decomposed into two sub-problems, namely, energy allocation and scheduling length optimization. We propose an algorithm named Efficient Scheduling with Energy Consumption Constraint (ESECC) to solve the problem of minimizing the scheduling length of parallel application under energy constraint. The proposed algorithm realize the transfer of energy constraint of application to that of each task by a pre-allocation approach, and introduce a concept of available energy for energy pre-allocation of unscheduled tasks, and improves the pessimistic results caused by the unreasonable energy allocation of existing algorithms.

(2) This paper then focuses on the research of reliability optimization under energy constraint in ACPS. The problem is decomposed into three sub-problems, namely, energy allocation meet the deadline and reliability optimization. We propose an algorithm named Reliability Enhancement with Response time and Energy Constraints (REREC) to solve the problem of reliability maximization of the parallel applications under energy constraints.

REREC algorithm reallocates the energy based on the ESECC, and transfers the deadline of application to the time constraint of each task, then selects the processor with high reliability under both energy and time constraint. This study covers the shortage of current research which merely considering the constraint of energy or deadline.

(3) Extensive comparative experiments were do to verify the algorithms proposed in this paper. Experiments on the real parallel applications validate the proposed ESECC algorithm can effectively reduce the schedule length of applications while satisfying the energy constraints. Experiments on the real-life automotive function and synthetic randomly generated automotive functions show that the proposed REREC algorithm can still achieve very high reliability while satisfying energy constraints and not exceeding application deadlines. Therefore, our research could facilitates a part of energy-aware scheduling during the design phase of ACPS.

Key Words: ACPS; DVFS; Energy-aware; Heterogeneous system; Reliability; Schedule length

目 录

插图索引

附表索引

第1章 绪论

1.1 研究背景与意义

1.1.1 研究背景

21 世纪是信息技术飞速发展的时代，信息技术被广泛地应用于航天、医疗、工业生产以及人类生活等各个领域。生活中最常见的汽车，也在信息技术的推动下朝着智能化、网络化的方向发展，从最开始只单纯地用来代步演变成一个集代步、娱乐、休息功能于一身的载体。

汽车工业的发展历史已有几百年，最早可追溯到十七世纪末期的蒸汽汽车。早期的汽车都是机械控制，而嵌入式系统被应用于汽车领域则是在十九世纪中后期。1968 年，大众汽车公司第一次在生产的 Volkswagen 1600 车型中使用了微处理器来进行燃油喷射系统的控制，实现了物理控制与计算系统的结合[1]，自此开启了汽车电子系统的发展阶段。最初的汽车电子系统比较简单，微处理器只处理单一的任务，不与其它处理单元进行通信或交互。为了满足人们对汽车安全性、舒适性以及娱乐性的需要，汽车电子系统的功能应用日益增多，并且由最初单一的功能逐渐发展成分布式的功能应用，因此其复杂度也骤增，图 1.1 所示为汽车内部的汽车电子系统分布图。现代的一些高端轿车，其内部的电子系统由近百个 ECU(Electronic Control Units)以及多种传感器、执行器等设备组成，这些设备通过多种网络总线如 LIN、CAN、FlexRay 等进行通信[2,3]，运行的系统软件有着上千万行的代码并且还在持续增加[4]。

图 1.1 汽车电子系统

现代汽车电子系统是一个集合了计算、网络以及物理环境的复杂分布式系统。其内部的各个处理单元通过总线以及网关进行通信，并且利用多种传感器实时与外界的物理环境如汽车、路边基础设施单元等进行交互，共同协作完成一系列功能应用，实现汽车的安全、舒适驾驶[5]。所以现代汽车电子系统可以看做是一个典型的信息-物理融合系统(Cyber-Physical System，CPS)[6]，也称为汽车信息-物理融合系统(Automotive Cyber-Physical System，ACPS)[7]。如图 1.2 所示，ACPS 中的 Cyber 部分包括了汽车电子系统中的计算和通信系统，通常由多个 ECU 通过 CAN、FlexRay 总线乃至最新引入的以太网进行通信，从而实现对分布式任务的调度，管理硬件资源和实现与物理世界交互；而 Physical 部分则强调对物理世界的反馈控制，主要接口为传感器和执行器，传感器从物理环境如道路、天气、温度等采取数据，通过 Cyber 部分的高性能计算对数据进行整合、计算和加工，然后以控制命令的方式传输给执行器，通过执行器与物理世界进行反馈控制。

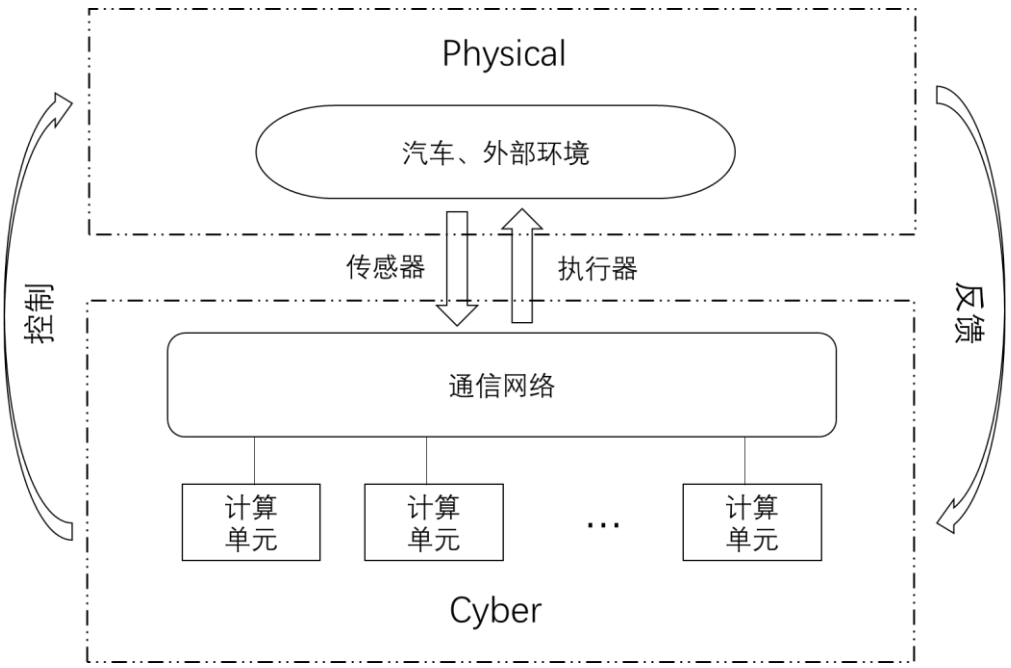


图 1.2 汽车电子系统与 CPS

CPS 是美国科学基金会在 2006 年提出的新技术概念，并被列为美国未来八大关键信息技术的首位，同时，德国的“工业 4.0”战略也将 Cyber-Physical Production System (信息-物理生产系统)作为核心技术[8]。CPS 的本质是实现计算、通信与物理系统的一体化设计，使系统更加可靠、高效、实时协同，具有重要而广泛的应用前景[9]。以 CPS 的观点进行系统设计越来越被航空、汽车以及能源等行业所采用，是工业发展的一种趋势[10]。汽车电子系统的分布式和异构化以及可以与物理环境交互的特性使得其成为以 CPS 观点进行设计最合适的候选者之一[11]，因此关于 ACPS 的设计与研究也已成为国内外研究人员共同关注

的热点问题[12,13]。

作为人类生活中的必不可少的一部分，ACPS 的设计必须要考虑低成本、低能耗。节能环保是汽车工业领域永恒的主题[14]，第四次工业革命的到来赋予了汽车工业新能源、绿色环保发展的新动力。近年来，新能源汽车得到大力发展，市场上也逐渐浮现一些油电混合动力汽车以及纯电动汽车，如 Tesla Model S 以及国产的比亚迪电动车[15]，图 1.3 显示了近几年我国新能源汽车的销售情况[16]，从图中可以看出新能源汽车的销售量呈现较高的增强趋势，而其中中纯电动汽车占了主体部分，纯电动汽车的占比率也逐年增长。传统汽车中汽车电子应用的成本能占到汽车总成本的 25%，而在新能源汽车中这一比例将达到 50%左右[17]，因此在设计初期需要严格控制相关的资源成本。电能也是资源的一种，在分布式计算领域，绿色计算是人们倡导的新型计算模式，一些大型的数据中心每年都会消耗巨大的电能，为了节约成本，公司往往在数据中心附近修建自己的发电厂来保障供电，同时能耗管理技术被广泛研究与应用[18]。ACPS 对电能的需求虽然远远没有数据中心那么高，但由于它是一个嵌入式系统，其所需的电能主要来源于电池，因此低能耗的研究同样非常重要。如今计算性能的发展速度远超电池技术，即使近年来有许多新型材料电池被研发出来，但要真正量产进入实用还需要走很长的一段路。ACPS 是一个对能量感知的系统[12]，因此 ACPS 的设计除了要考虑传统的计算资源和硬件成本外，必须考虑能量资源的限制因素，汽车设计厂商在初期需要对 ACPS 中有限的能量进行合理的设计分配与使用。

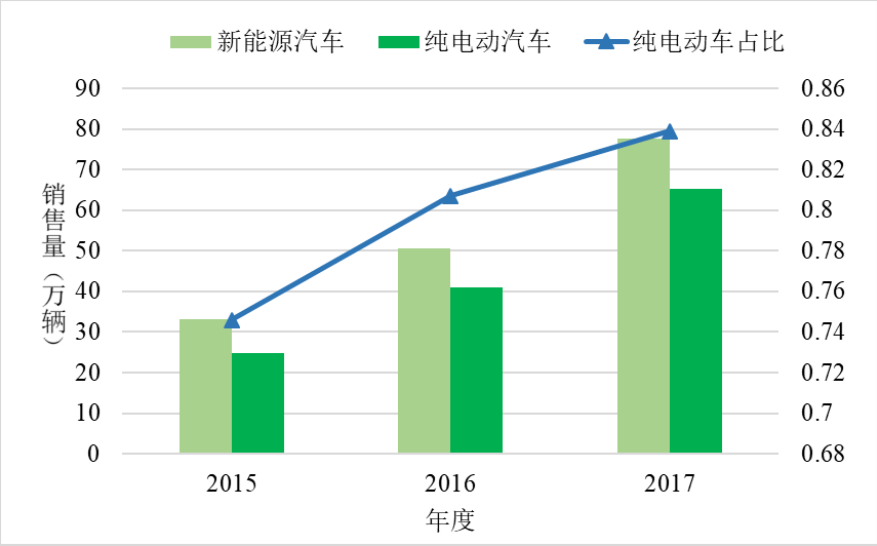


图 1.3 近几年我国新能源汽车销售情况

另一方面，伴随着近几年人工智能技术的突破发展，人们已经开始了自动驾驶的研究和实施[19,20]。自动驾驶的出现意味着在汽车驾驶的过程中人脑的工作将逐步被计算机所取代，自动驾驶所应用的机器视觉技术以及深度学习带来的是数据量与计算量的进一步增加，相应地能耗也因此增加。与此同时，汽车功能应

用遭受着软硬件瞬时故障、温度升高、外来网络攻击等安全风险问题，而 ACPS 高度的复杂度以及集成度使得这些风险发生的概率进一步增大，系统的可靠性问题日益严重。系统功能应用的可靠性需要得到进一步加强，否则一旦出现系统故障，将造成难以挽回的安全事故。例如安全气囊打开失效或者提前被打开都会威胁到驾驶者的生命安全。近年来，关于自动驾驶出现交通事故的例子时有发生，也有由于自动驾驶过程出现系统故障造成人员伤亡的案例。因此，ACPS 的设计必须需要满足安全可靠的需求，2011 年正式颁布了道路车辆-功能安全 ISO 26262 是一个适用于汽车电子/电气系统的国际标准[21]，给汽车研发设计的整个生命周期提供了指导并提出一系列安全标准要求，设计人员需提前评估功能应用的可靠性，以保障汽车电子系统功能的安全。

综上所述，我们可以总结得到 ACPS 的一些特点：

（1）ACPS 是一个计算系统、网络系统以及物理环境深度融合的智能系统。汽车通过传感器采集车身以及环境数据并在车内网络进行传输，由计算系统进行数据分析与处理，同时将处理结果交由执行器去执行，以实现汽车与物理环境的交互。

（2）ACPS 可以看做是一个异构分布式系统。异构性体现在计算单元异构以及网络的异构等，针对不同的计算性能需求以及分布位置的不同，车内的 ECU 的性能可能不一样，另外不同生产厂商供应的 ECU 也可能会不同；多种类型的数据传输对网络带宽以及实时性的需求也不一样，例如控制信号需要较小的带宽即可，但要求较高的实时性，而多媒体相关的应用则对带宽有较高的需求。另一方面，功能的实现离不开分布于车内各个地方的 ECU 的交互与协作，且这些 ECU 没有主从关系[5]，因此它也是一个分布式系统。

（3）ACPS 是一个对实时性、可靠性要求非常高的系统。汽车电子系统是一个典型的嵌入式实时系统，功能的实现需要 ECU 进行复杂的计算以及相互之间的交互协作完成。汽车电子功能一般需要在极端的时间内完成，例如与刹车相关的功能需要在毫秒级内完成，同时还要保证功能的安全可靠，不能出现故障，否则可能会带来严重的后果。

（4）ACPS 是一个对资源极其敏感的系统。这里的资源不仅包括计算资源、网络资源，同时也包括电能这样的能量资源。ACPS 中电子功能数量以及系统复杂度的增加给 ACPS 中的计算与网络资源带来了更高的需求；同时汽车的发展将走向纯电池驱动，因此对能量的使用将变得更加苛刻，在设计阶段就要对能量的分配与使用进行规划。

总之，ACPS 是一个复杂的异构分布嵌入式计算系统，并且对实时性以及安全、可靠性都有着非常高的要求，由于汽车与人们的日常生活息息相关并且已成为生活中必不可少的一部分，所以相应地 ACPS 设计的资源以及成本的使用也

必须严格的控制。因此在汽车的设计阶段需要从整体上考虑计算、网络、物理环境以及能量资源，实现最优的配置，这也是未来汽车工业设计与开发阶段所面临的难题与挑战。

在异构分布式系统中，任务调度是一项重要的研究课题，也是实现资源配置优化的方式之一[22]。通常在共享计算和网络等资源的前提下，需要对任务进行合理的分配执行以提升整个系统的吞吐量和计算效率，不合理的调度策略往往会浪费大量的资源，比如可能会出现部分处理器长期处于空闲的状态，这不仅浪费了计算资源，还浪费了能量资源。对于以 ACPS 为典型的异构分布嵌入式系统，实现能量与性能之间的优化更为迫切[31]。ACPS 中的功能应用往往由多个子任务组成，在有限的资源环境下，需要为任务进行处理器的分配，保证功能应用在截止时限内完成，并且保障功能的安全可靠。本文针对 ACPS 中的能量资源有限的情况，从调度的角度出发，对相关的任务调度理论与方法进行分析，根据系统的能量约束来确定任务执行所需要的能量，以实现 ACPS 中功能应用能量感知的调度。

1.1.2 研究意义

随着社会经济的迅猛发展，汽车逐渐走进每一个普通人的家庭，成为日常生活中常见的消费品，给人们的生活带来了便利。如今汽车产业已成为国家经济体系中非常重要的一部分，同时汽车产业的技术水平也能从侧面反映出一个国家的工业发展程度。ACPS 旨在使汽车更加智能化、舒适化，提升人们的驾驶体验与安全性，同时人们日益增长的应用需求给 ACPS 的设计带来了不少难题，ACPS 成为国内外学术界与产业界研究与开发的重要方向。考虑到未来新能源汽车发展，汽车的能量将越来越多地以电池供给。因此 ACPS 的设计需要充分提高能量的利用率，在有限的能量资源下保障汽车功能的高实时与高可靠要求，实现能量感知的 ACPS 设计。这不仅能促进汽车工业节能环保的发展，同时相应的设计思想也可为其它 CPS 系统的设计提供一定的参考价值。

1.2 研究现状

汽车电子系统的发展有着上百年的历史，但关于 ACPS 的研究是近些年才有的，属于比较新的领域。但由于汽车工业在国民经济中的重要性，因此受到学术界以及工业界的广泛关注，相关的研究主要包括建模分析、控制理论与方法的设计、功能验证以及资源优化等等。

1. 国外研究现状

国外学术界对于 ACPS 的研究侧重于系统模型以及控制理论的设计。ACPS

中人的地位与作用不可忽视，Osswald 等人将人看做 ACPS 中重要的一部分并从整体上进行分析与设计，举例说明了包含人在内参与反馈控制的 ACPS 结构，这和传统的嵌入式系统设计有着本质的区别，同时作者表明未来包含人参与反馈控制的 ACPS 设计更关注软件开销、安全性以及用户需求等[1]。ACPS 中网络通信与计算、控制同等重要， ACPS 的网络系统将越来越复杂，Zeng 等人阐述了五种广泛使用的车内网，从系统开销、数据传输能力以及容错能力三个方面进行比较，分析了车内网络拓扑结构的发展趋势[3]。控制理论方面，Chakraborty 等人对 ACPS 设计面临的系统控制问题、资源问题以及可靠性问题进行了分析，ACPS 的分布式以及异构性使得相应的问题变得更加复杂，需要新的控制理论与方法[7]。该研究团队在文献[12]中对 ACPS 中的资源问题又进行了非常详细的总结，针对资源感知的汽车控制系统，从存储资源、计算资源以及电能等方面对相关的理论与方法进行了综述。Zhang 等人提出一种协同仿真框架辅助时间触发的 ACPS 设计，该框架模型包括物理层、网络/平台层以及软件层三个部分，并针对具体应用进行了原型设计、分析与仿真[13]。针对越来越成熟的电动汽车，Tehrani 等人给出了一种信息能量系统的设计方法(Cyber Physical Energy System Design, CPESD)，通过控制理论对电动汽车中的电能、燃油、太阳能资源进行合理规划与使用[14]。

由于 ACPS 是一个异构分布嵌入式系统，因此异构分布式系统以及嵌入式系统中的调度理论与方法也可以拿来借鉴使用，以进行资源配置的优化。文献[23~28]对能量与可靠性优化的问题进行了大量研究，这两者之间的优化研究在 ACPS 中同样非常重要，虽然这些研究面向的不是异构分布嵌入式系统，但采用的基本的能量模型以及可靠性模型可以为 ACPS 中的能量与可靠性调度问题作参考。我们将在第二章对这些文献所提出的具体理论进行详细分析，由于不是最新研究，这里不单独列举描述。

此外，国外产业界以及部分研究机构已经有着相对比较成熟的辅助驾驶系统以及无人驾驶技术，典型应用如谷歌的无人车技术以及已经进入大众市场的特斯拉自动驾驶技术，这也是 ACPS 中极为复杂的系统功能，需要精确的机器视觉技术以及深度学习的理论。一些大型的互联网公司，如苹果公司也在开发适用于未来汽车的驾驶操作系统。此外汽车制造商、汽车零部件供应商以及通讯设备制造商也联合协作，研究车与车之间的通信交互系统[29]。这些都是已经从实验室走向实际应用的工程研究成果，在这方面，欧美的一些发达国家以及日本走在了前列。

2. 国内研究现状

单从汽车产业来说，国内与欧美发达国家还有一定的差距，相关的 ACPS 研究不多。关于 ACPS 的研究，首先要解决的就是建模问题。不同于传统的离

散建模，ACPS 中的物理状态是连续的，在文献[30]中，作者对基于数据的 ACPS 建模方法以及验证技术进行了仔细的研究，将离散的计算网络与连续的物理环境融合起来建模，从数据出发实现从离散到连续的建模过程，并通过反馈调节修复误差。Xie 等人从调度的角度对 ACPS 中功能应用的时间、资源以及安全展开了充分的研究[39~43]。其中文献[39] 针对集成体系结构的 WCRT 分析，提出了网关互连的 CAN 网络端到端 WCRT 分析，是目前该问题最新的研究成果。针对功能安全验证，文献[40] 提出了基于 ISO 26262 标准的功能安全验证方法，包括实时性验证和可靠性验证。此外针对资源优化，文献[41]提出了低成本的优化方法，文献[42] 提出了低能耗优化方法。针对 2017 年 5 月颁布的 AUTOSAR 新平台标准，文献[43]提出 ACPS 的自适应调度框架以及混合关键级系统自适应的动态调度方法。以上研究成果包括了从基础理论到功能安全验证与资源优化再到自适应动态调度，展现了一套完整的 ACPS 分析、验证、优化以及调度的理论与方法，这些研究都是目前 ACPS 调度领域最新的研究成果。此外，针对能量约束问题，Xiao 等人提出了能量约束下的调度长度最小化算法[32]，其因新颖的解决方法获得了 ISPA 2016 会议的最佳论文奖，该作者后续在文献[33] 中提出了能量约束下的可靠性最大化算法，这两项研究非常适用于能量有限的 ACPS 环境。

由于我国人口众多，汽车市场十分庞大，因此我国一直格外重视这一领域的发展。“十二五”期间，国家 863 计划对车联网的关键技术进行了研发，已经取得了初步的成果，基于车路交互的系统在实际道路上进行了应用实验[34]。国内的一汽、上汽、普华软件等公司和浙江大学、湖南大学等高校组成的中国汽车电子基础软件自主研发与产业化联盟也于 2011 年成立，以发展我国的汽车电子系统[35]。互联网巨头公司阿里巴巴、腾讯以及百度都在无人驾驶研究领域投入了大量的资金。这些都为 ACPS 的研究与发展奠定了坚实的基础，而伴随着“中国制造 2025”计划的落实，智能制造将成为主攻方向，同时也将会大大带动国内汽车电子产业的发展，逐步缩小与发达国家的差距。

1.3 研究问题

新能源汽车将是汽车电子产业发展的“新蓝海”，由于我国庞大的人口基数，新能源汽车的销量近年来也处于全球第一的位置，而其中又主要以纯电动汽车为主[16]，因此能量也将成为 ACPS 中的关键资源。ACPS 是对能量感知的系统，任务执行能量的分配需要根据有限的能量而来，即不能超过整个系统的能量约束。随着 ACPS 中功能应用的不断增多，如各种主动/被动安全系统、车联网系统以及越来越丰富的多媒体服务，车内的 ECU 数量不断增加，对 ACPS 的计算

性能以及可靠性提出更高的要求，能耗管理成为设计阶段不可忽视的一项内容。

1.3.1 能耗与实时性问题

ACPS 的低能耗与实时性是两个相互制约的目标。为了实现低能耗，除了设计出更低功耗的芯片，更多地是合理地利用能量资源，从调度的角度来看就是通过任务调度实现能量的合理使用，在性能与能量使用二者中达到有效平衡。汽车电子系统对功能应用的调度时间是十分苛刻的，功能应用必须在其截止时限内完成才有效，因此需要尽可能地降低调度长度以保证功能的正常运作。通常调度长度越短，说明系统的性能越好，但是一般来说高性能又意味着更高的能耗。因此，需要在保障性能需求的同时降低能耗，或者说在有限的能源下尽可能实现更高的性能。

目前能耗与时间优化相关的调度研究主要有两条研究路线，一是在保证整个应用的调度长度不超过其截止时限的前提下，尽可能地降低系统完成应用所消耗的能量，达到节能的效果；另一种是在系统能源有限的情况下，尽可能地降低整个应用的调度长度，达到最大化的性能，这也是实现能源的充分、有效的利用[32]。关于前者的研究相对较多，相关的调度策略也比较成熟，而后者是从一个与前者对立的角度去思考，由于 ACPS 中的能量将主要以有限容量的电池供应，因此从该角度考虑问题更为贴切。

1.3.2 能耗与可靠性问题

ACPS 是一个安全关键的异构分布嵌入式系统[5]，伴随着 ECU 的大量使用、性能的不断提升，其面临的可靠性问题也日益严重。当汽车在运行时，ECU 的故障或失效将可能带来灾难性的后果。道路车辆-功能安全标准 ISO 26262 就汽车行驶过程中潜在的风险划分了不同的等级，实际上是对不同的应用有着不同的可靠性等级要求。例如跟汽车动力控制相关的功能如线控刹车、防抱死刹车等，这些功能应用不仅需要在极短的时间内完成（毫秒级别）， 而且还要保证极高的可靠性，否则可能危及驾驶员生命安全。而像车窗、空调的控制系统以及一些娱乐多媒体服务则对实时和可靠性要求相对较低。

由于系统中的功能模块越来越多、越来越复杂，在进行能量管理的同时也会造成一定的系统瞬时故障，影响到系统的可靠性。因此在调度的过程中，需要对功能的能耗、时间以及可靠性进行分析，解决低能耗与实时可靠这三重目标优化的问题。而这三者之间是互相牵制、互相影响的，比如优化实时性目标的同时就可能导致能耗的增加以及可靠性的降低，因此这三者的优化很难同时满足。本文将考虑在 ACPS 中功能应用能量受限的情况下，同时满足实时性需求，通过相应的调度策略增强系统的可靠性，为资源优化提供一种新思路。

1.4 本文内容及贡献

本文主要研究 ACPS 中的调度问题，结合未来汽车朝着纯电动化、绿色环保的发展趋势，针对系统中的能耗、实时以及可靠性等因素提出更有效的调度理论和方法。主要包括以下内容：

（1）将 ACPS 抽象成异构分布嵌入式系统，根据 ACPS 中功能应用的特征进行 DAG 抽象描述，针对 ACPS 中存在的能耗、实时性以及可靠性问题，从任务调度的角度出发，对有能耗优化相关的调度研究进行详细的总结和归纳，重点分析了以 DAG 抽象建模的并行应用能耗优化调度问题的研究进展，包括 DAG 应用的能耗与调度长度优化以及能耗与可靠性优化。此外针对 ACPS 中能量有限的情况，总结了现有能量约束下调度研究的不足之处，对需要进一步研究以及改进的地方进行了分析总结。

（2）进行能量约束下的调度长度最小化研究。现有的研究大多从满足时间约束去降低能耗的角度出发，很少考虑能量约束条件，而仅有的考虑能量约束条件的相关研究，其不合理的能量分配策略导致调度结果很悲观。针对现有的低能耗调度策略研究的不足，本文提出更合理的能量分配策略，采用了更均匀的分配方式，任务在其分配的能量下选择最优的处理器以降低调度长度。所提出的能量约束下的高效调度算法相比现有算法复杂度并没有增加，同时本文通过理论分析以及实际的并行应用模型进行丰富的实验验证，并与现有的同类型最先进的算法进行对比，大量的实验结果展示了本文算法的高效性，在有限的能量约束下降低调度长度方面相比现有算法有着显著的提升。

（3）进行能量约束下的可靠性最大化研究。当前相关的研究只考虑了能量约束，并且最终的调度结果比较悲观，应用的截止时间需求也很难得到保证。因此本文将应用的截止时间需求进行了考虑，在保证能量不超过系统给定的能量约束下，重新进行能量分配，同时对应用截止时间进行分析，分析应用的截止时间与任务执行时间之间的转换关系，并根据可靠性模型，重新进行任务的处理器分配，以达到最大化的可靠性。提出一种能量约束下同时保障应用截止时间需求并增强功能应用可靠性的调度算法，通过真实的汽车电子功能应用以及仿真的功能应用进行实验验证，并和现有研究中比较类似的一些算法进行对比，实验结果显示所提出的方法在满足能量约束限制下，能有效地提高功能应用的可靠性，同时还考虑了截止时间需求。

1.5 本文组织结构

本文的组织结构如下：

第一章为绪论。首先介绍了汽车电子系统的发展现状以及ACPS相关概念，

对ACPS的特点进行分析总结，阐述相关的研究背景与意义；针对当前汽车产业朝着纯电动汽车发展的趋势，从计算的角度对ACPS系统中能量约束以及可靠性需求的调度问题进行描述，最后简述本文的研究内容与贡献，以及文章的组织结构。

第二章是相关研究基础及进展。先介绍ACPS的系统结构，并对相应的功能应用进行抽象描述，接着简要介绍与能耗相关的优化技术，最后针对能耗优化的调度研究展开全面且详细的综述，总结分析现有能耗相关的调度理论与方法的研究进展以及不足。

第三章研究了ACPS中能量约束下的调度长度优化问题。提出了一种预分配平均能量的方法来保证应用的能耗不超过给定的系统总能量，实现应用的能量约束与应用中每个任务能量约束之间转换，提出了相应的调度算法以优化调度长度，最后通过实验对算法进行验证。

第四章研究了ACPS中能量约束下的可靠性优化问题。基于上一章所提出的能量分配方法，本章进一步考虑了截止时间的约束，对能量进行重分配以及对截止时间进行松弛。提出了能量约束下的可靠性优化算法，并在最后用真实的汽车功能应用进行验证。

最后是结论和展望，针对本文的工作以及相应的创新点进行最后的总结，并对未来的研究工作进行展望。

第2章 相关研究基础及进展

2.1 引言

任务调度是实现资源优化配置并提高系统性能的主要途径之一，而相应的调度理论与算法也是一直以来热门的研究课题。ACPS 的异构化、网络化以及高度复杂化特性使得传统的嵌入式系统调度模型无法对其进行精确地描述，因此需要根据其系统结构以及应用的特点重新选取合适的模型并对相关的问题进行描述、分析与研究。

本章先介绍 ACPS 中的调度模型，根据汽车电子系统中实际的功能应用运行特点，分析 ACPS 结构以及功能应用的模型，并将 ACPS 抽象成异构分布嵌入式系统进行相关问题的研究；然后针对能耗问题，简述了常见的能耗管理技术，最后对异构分布式系统中能耗管理相关的调度理论与方法及其进展进行详细的综述与总结。

2.2 ACPS 调度模型

2.2.1 ACPS 结构

随着信息通信技术(Information and Communication Technology，ICT)的飞速发展以及在汽车电子系统中的大量应用，ACPS 逐渐异构化、网络化和复杂化[35]。如今高端轿车一般有上百个 ECU，自动驾驶技术的发展使得汽车 ECU 处理的数据量成几何级增长，因此需要有高性能计算能力以满足相应的计算需求，如物体识别需要强大的计算机视觉以及深度学习方面的实时计算来实现。传感器作为汽车电子系统各类数据信号的输入源，为车内各类 ECU 提供汽车的工作状况以及汽车周边的物理环境数据，如车内外的温湿度、踏板压力、汽车位置以及速度等等。汽车电子系统中典型的系统，如高级辅助驾驶系统(Advanced Driver Assistance Systems，ADAS)，又被称为主动安全系统，主要包括车身电子稳定系统、车道检测系统、停车辅助系统、自适应巡航系统以及夜视系统等等，其中每个子系统的运行都包含着数据的采集与处理过程。一个子系统的运作往往也离不开多种传感器，如停车辅助系统不仅利用摄像头采集汽车后方影像画面，并且当车身靠近障碍物时还会发出雷达警报声；自适应巡航系统利用雷达传感器获取车间距离信息，同时安装在速度输出轴上的车速传感器实时获取车速信

息，制动踏板传感器感知来自驾驶员脚底施加的压力等等，这些传感器协同工作以保障系统的正常安全运行。

ACPS 中功能应用的实现离不开多个 ECU 的交互协作，以线控刹车功能为例，它包含了一个传感器和两个执行器，功能的实现依赖于 5 个 ECU 的协作执行以及在两条 CAN 总线上进行数据传输。具体过程如图 2.1 所示，传感器接收外部数据并将数据以消息的形式打包通过 CAN 网络进行传输，任务的执行被分配到 5 个 ECU 上，最后通过两个执行器执行刹车命令[5]。汽车电子系统里的 ECU 根据其在汽车不同功能系统和配备位置的区别而呈现不同的形态，即汽车电子系统里的 ECU 具有异构性。在本文中，ECU 也被称为处理器，我们用集合 $U=\{u_1,u_2,\ldots,u_{|u|}\}$来描述这些异构处理器集合，|U|表示这些处理器的个数。

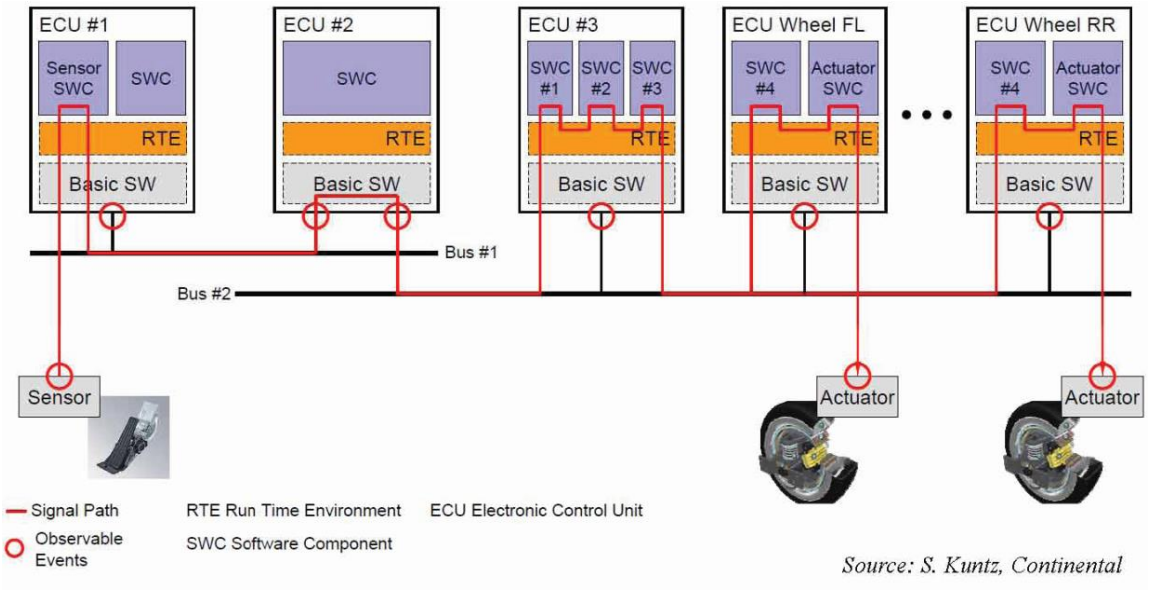


图 2.1 线控刹车功能应用的具体实现过程

通过分析可以得到 ACPS 的一般结构模型，图 2.2 展示了一个简单的 ACPS 系统结构[35]，该结构里包括四条 CAN 总线，并且这四条 CAN 总线由网关互联。系统中多个 ECU 连接在这四条 CAN 总线网络上，并且部分 ECU 与传感器(Sensor)相连，部分与执行器(Actuator)相连。接下来通过举一个简单例子来描述应用的执行过程，ECU_1 接收来自传感器的数据并触发任务 n_1，当 n_1 执行完之后，其会发送一个消息 $m_{1,2}$ 给下一个在 ECU_6 中的任务 n_2。在这一过程中，$m_{1,2}$ 需要先后在两条 CAN 总线 CAN_1 和 CAN_3 上传输。经过一系列的任务触发执行，最终通过将信号发送给执行器来完成相应的动作。由此可见 ACPS 的计算架构是一个典型的分布式计算模型。

2.2.2 功能应用模型

为了满足人们对汽车安全性、舒适性以及娱乐性方面的需求，汽车电子功能的数目也日益增多。传统的嵌入式系统领域中简单的周期任务模型已经不能精确

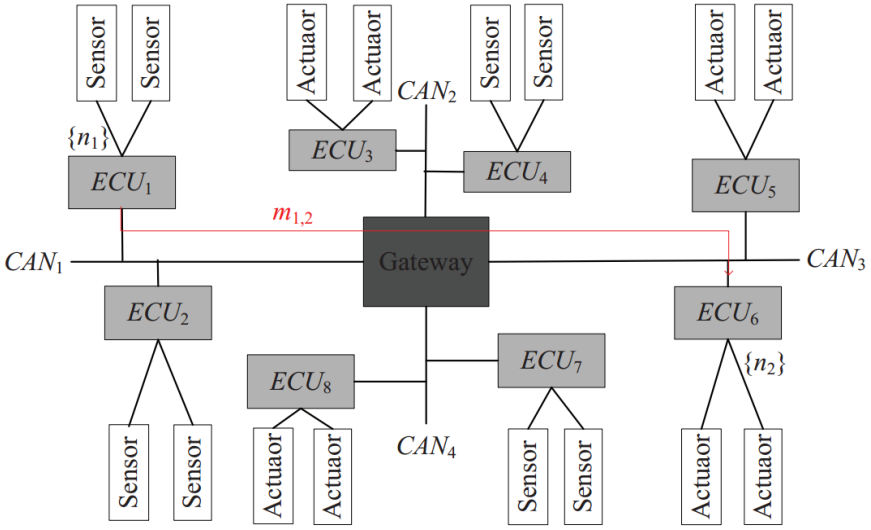


图 2.2 一个简单的 ACPS 结构

描述当前功能应用，当前的汽车电子系统需要从功能级的角度去考虑一系列问题，如最坏响应时间(WCRT)优化以及资源优化[35]。例如前面提到的线控刹车应用，当驾驶员踩下踏板时，传感器对汽车当前的运行状态以及环境数据进行采集，接着将数据以消息形式通过 CAN 总线传输至 ECU，ECU 收到信号并处理后再传送给其它 ECU 单元执行，最终将刹车信号传送给执行器完成刹车动作。整个过程以降低 WCRT 为核心，以保证车上人员的生命安全。 由此可见，一个功能应用往往包括多个子任务，而这些子任务一般也具有先后执行顺序，即具有优先级约束。针对任务间具有优先级约束的功能应用，常用的建模方法是用有向无环图(Directed Acyclic Graph，DAG)来对其进行描述，如图 2.3 为一个简单的 DAG 示例。

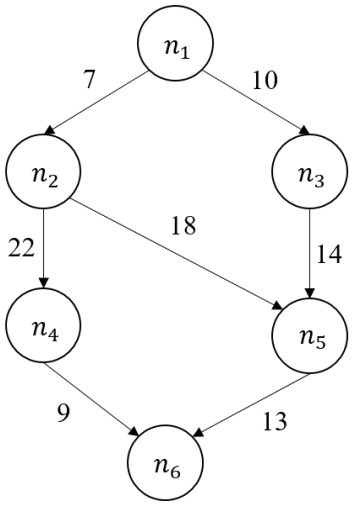


图 2.3 简单的 DAG 示例

DAG 可以用一个四元集合进行数学描述：$G=\{N,W,M,C\}$，下面对相应的物理元素以及 DAG 中常见的物理名词作简要介绍：

(1) N 表示 DAG 中的结点集合：$N=\{n_i\},1\le i\le|N|$ 。每一个结点表示一个

任务， 在 DAG 中， $pred(n_i)$表示任务 n_i 的直接前驱任务集合， $succ(n_i)$表示任务 n_i 的直接后继任务集合。例如图 2.3 所示的 DAG 中，任务 n_5 的前驱任务集合表示为 $pred(n_5)=\{n_2, n_3\}$,其后继任务集合为 $succ(n_5)=\{n_6\}$。如果一个任务没有前驱任务，则称为入口任务，记为 n_{entry}(例如图 2.3 中的 n_1) ；同样，如果一个任务没有后继任务，则称为出口任务，记为 n_{exit}（例如图 2.3 中的 n_6）。

(2) W 表示任务的最坏执行时间(WCET)矩阵，假设系统中的异构处理器集合为 $U=\{u_1,u_2,\ldots,u_{|u|}\}$，则它是一个大小为 $|N|\times|U|$ 的矩阵，同一任务在不同处理器上执行所需要的执行时间是不同的，这是由于处理器的异构性导致不同处理器的处理能力不同。本文用 $w_{i,k}$ 表示任务 n_i 在处理器 u_k 上以最大频率执行所产生的 WCET，一个任务在某个处理器上执行的 WCET 是该任务在该处理器上以最大频率执行可能需要的实际最长时间。实际上很难测量得出某个任务准确的 WCET，因为任务往往共享内存或其它硬件组件，执行时相互之间会产生影响，在本文中我们不作 WCET 的详细理论分析，并且假设所有的 WCET 是已知的，可由文献[37]的理论分析获得。

(3) M 表示 DAG 中边的集合。 $m_{i,j}\in M$ 表示从结点 n_i 到 n_j 的通信消息。相应的， $c_{i,j}\in C$ 表示消息 $m_{i,j}$ 的通信时间，这里的通信时间指的是消息 $m_{i,j}$ 的最坏响应时间(WCRT)。消息的 WCRT 表示消息所有传输情况下可能的最大实际响应时间，并且是个理论上的上限值，因为准确的 WCRT 计算是一个指数级的组合难题，因此通常都是从理论上去分析得到一个尽可能紧凑的 WCRT 上限。关于 WCRT 分析的理论和方法可参阅文献[38]和文献[39]，同样本文不作详细描述，并假设所有的 WCRT 是已知的。需要注意的是，当 n_i 和 n_j 处于同一处理器中执行时，定义它们的通信时间为 $c_{i,j}=0$。

(4) $SL(G)$表示 DAG 的调度长度(Schedule length)，也称应用的完成时间(Makespan)或响应时间(Response time)，一般为应用中最后一个任务的执行结束时间，同时用 $D(G)$表示应用的截止时间。

由于 ACPS 中的功能应用越来越走向复杂化，任务之间具有优先关系，因此引入 DAG 对功能应用进行抽象分析是研究趋势[5]。

2.3 能耗优化管理技术

能耗管理是近年来绿色计算领域研究的重要且紧迫的热门课题。通常人们熟知的降低能耗的技术主要包括两种：动态电源管理(Dynamic Power Management， DPM)技术和动态电压/频率调节(Dynamic Voltage and Frequency Scaling, DVFS)技术，一般针对不同的应用场景选择不同的节能技术或者同时采用多种节能技术。

2.3.1 动态电源管理

动态电源管理技术指的是在系统组件不工作的时候将其关闭或者切换为休眠模式，从而降低系统在空闲时段的能耗。在嵌入式系统中，一般通过实时分析功耗可管理组件 (Power Manageable Component，PMC)的状态和负载，动态调整其工作状态[44]。一般来说，PMC 具有多种功耗模式，如运行模式和休眠模式。当组件长时间处于空闲状态没有任务执行时，可以将其切换到休眠模式，而在有任务到来时再给其上电唤醒进入运行模式，如图 2.4 所示。为了实现更合理、更高效的电源管理，几类比较流行的 DPM 控制策略相继被提出，主要包括超时策略、预测策略和随机策略[45]。

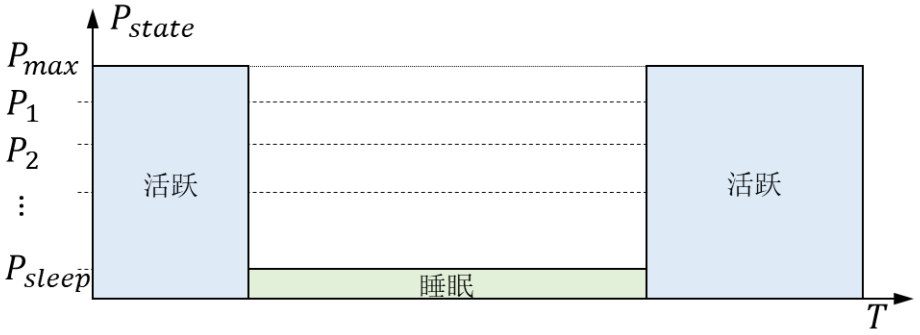


图 2.4 DPM 节能调度原理

(1) 超时策略指的是给系统设置时间阈值 T，当系统的空闲状态持续时间达到 T 时，则切换到休眠模式，一旦有任务需要执行则重新恢复运行模式。时间阈值 T 可以是固定的，也可以根据系统的负载状态进行自适应地动态调整。这种策略实现起来最为简单，也是应用最为广泛的一种。

(2) 预测策略是通过对历史数据进行“学习”，利用统计分析对即将到来的空闲时段的长度进行预测。因此这类算法的预测准确度是衡量算法性能的一项重要指标。一旦预测的准确度较低，则不但不能实现节能的目的，可能还会造成系统的性能损失。

(3) 随机策略是一个具有不确定性的策略，它是将 DPM 作为一个随机优化问题，基于不同的随机分析模型，用相应的随机优化方法进行设计求解。

通过分析可以得知，DPM 技术实现能耗优化依赖于系统空闲状态时间的长度，并且需要一个良好的 DPM 控制策略。因此，它比较适用于系统空闲时间比较长的情景，如果系统的空闲时间较短，频繁地进行系统模式的切换反而无法降低能耗，还会影响系统性能。

2.3.2 动态电压/频率调节

动态电压/频率调节技术是针对 CPU 的电压和频率可以进行调节的能耗优化

技术[46]，也是嵌入式实时系统中普遍采用的一种技术，其基本思想是在不影响系统执行效率的前提下动态地调节系统组件的工作电压或频率，从而实现能耗的优化，如图 2.5 所示。目前主流的商业处理器系列如 Intel Atom、AMD Fusion 以及 ARM Cortex A9 都支持 DVFS 技术[47]。DVFS 技术能够降低能耗的基础在于当前的处理器主要采用的是 CMOS 电路技术，CMOS 电路的功耗和其供电电压有关，供电电压越高，功耗越大。另一方面，电压和时钟频率成正比的特性，降低电压会同时导致时钟频率的降低，对于同一任务其执行时间会随之增加，从而导致性能的下降。因此在实际应用中，需要在任务执行时间与能耗之间寻找一个最优的平衡点，既能实现节能，又能保证任务在截止时限内完成。

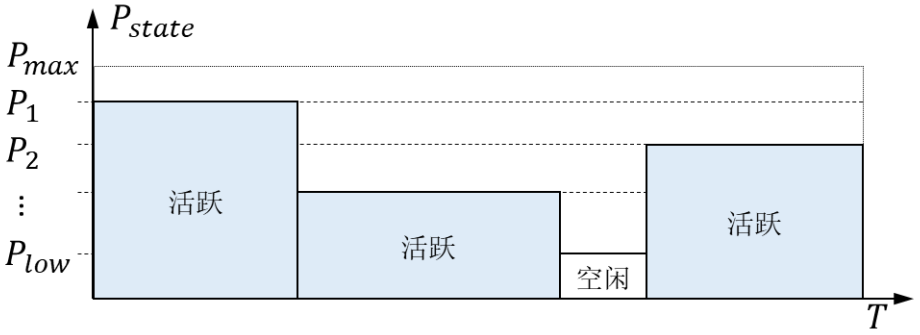


图 2.5 DVFS 节能调度原理

DVFS 主要用于降低处理单元的能耗，而 DPM 同时了降低处理器之外的硬件组件如内存、I/O 等模块的能耗，这一部分组件一般也可以被切换到低能耗状态[48]。同 DPM 技术一样，DVFS 技术在调节电压/频率的过程中也会产生额外的能耗和时间开销，但和 DPM 相比这些开销相对较小。DPM 技术只有在系统组件的空闲时间比较长的情况下才比较实用，并且采用 DPM 技术对系统组件不停地进行状态切换所带来的开销不容忽视，因此不是很适用于 ACPS 中对实时性以及可靠性要求较高的功能应用[45]。本文采用 DVFS 技术进行能耗优化，由于 DVFS 技术改变电压和频率的时间、能耗开销较小，出于简化考虑，本文将忽略这部分开销的计算。

2.4 能耗优化相关调度研究进展

在并行分布式计算领域，任务调度是提升系统性能的重要手段之一。调度问题是研究如何将一系列任务按照一定的调度目标，最优地分配到处理单元上进行执行。ACPS 作为典型的异构分布嵌入式系统，其应用模型为 DAG，因此可以从并行分布式计算的角度去研究 ACPS 中的调度问题，设计相应的低能耗调度算法去满足实时、可靠的设计目标。下面将介绍基本调度策略以及与能耗优化相关的调度研究进展，并进行详细的分析与总结。

2.4.1 基本调度策略

任务调度根据任务之间的关系特性，可分为独立任务和非独立任务，非独立任务即有着优先级约束关系的一组任务。对于满足任务优先级约束关系的任务调度问题，从本质上来看属于组合优化问题，此类问题的最优求解属于 NP 完全问题[49]。也就是说无法在多项式时间内找到最优的调度方案，因此在实际应用中，通常的策略是采用启发式算法求取调度问题的次优解，此类算法一般有着较低的复杂度以及较优的性能。启发式算法包含表调度算法、聚簇算法以及启发式的任务复制算法[54]，而其中表调度算法因能够得到较好的性能而得到广泛研究和应用。

表调度算法一般包含两个步骤：第 1 步是确定任务的优先级并根据优先级进行排序，第 2 步是为任务选择合适的处理器去执行。关于表调度算法的研究非常多，典型的表调度算法有：修改的关键路径算法(Modified Critical Path, MCP)[50]、动态关键路径算法 (Dynamic Critical Path, DCP)[51]、动态级调度算法 (Dynamic Level Scheduling, DLS)[52]、启发式映射算法(Mapping Heuristic, MH)[53]、异构最早完成时间优先算法(Heterogeneous Earliest Finish Time, HEFT) [54]、关键路径在同一处理器上算法(Critical Path On a Processor, CPOP) [54]等等。而这些算法中， HEFT 算法是 DAG 任务调度中最经典的算法之一。其基本思想是先根据任务的向上排序值的递减顺序确定任务的优先级，接着用插入的方法将任务分配给能获得任务最早完成时间的处理器，具体过程如算法 2.1 所示。

算法 2.1 HEFT 算法

输入：应用的 DAG、处理器集合 U

输出：DAG 调度结果

1. 初始化 DAG 中任务结点的计算开销与通信开销
2. 从出口任务向上计算每个任务结点的向上排序值 $rank_u$，并降序排序，得到降序排序列表 *dlist*
3. 当 *dlist* 不为空
4. 选择 dlist 中第一个任务结点 n_i
5. 对所有处理器，获取任务在其执行的最早开始时间
6. 计算 n_i 在所有处理器上的完成时间
7. 将最早完成时间对应的处理器分配给任务 n_i
8. 如果 *dlist* 为空，结束
9. 得到所有任务与处理器映射关系，以及最终调度长度。

由于 HEFT 和同类算法相比具有较好的性能，并且时间复杂度低，因此也

被广泛引用和比较，本文也将借鉴 HEFT 算法中的任务优先级排序以及处理器分配方法的思想，对相应的调度问题进行研究。

2.4.2 能耗与调度长度优化

能耗与调度长度优化是一个研究的比较多的问题，能耗的优化与调度长度优化二者之间是互相冲突的。通常来说，能耗的降低必然会带来调度长度的增加，即性能的损失。但是一个好的调度算法往往可以在保证性能的前提下实现能耗优化的目标。低能耗调度算法设计旨在满足一定的需求情况下，例如满足截止时间，尽可能地使任务执行所需要的能量最少，或者说是实现能量的高效利用。自上世纪九十年代开始，DVFS 技术被广泛应用于处理器能耗优化的相关研究。Weiser 等人首先提出通过对操作系统调度器中 CPU 的速率进行细粒度的控制来实现能耗优化[46]，其主要思想是监视 CPU 的空闲时间，通过降低时钟频率，延长运行时间以来降低能耗。从此激发出了大量关于空闲时间再利用以实现降低能耗的研究，从单个处理器平台到同构/异构多处理器平台，从独立任务集到非独立任务集。

DVFS 技术起初应用在单处理器上。Yao 等人根据任务的到达时间和截止时间提出了在线以及离线的算法来降低能耗[55]；Barnett 等人采用动态的电压调整技术，研究了在给定能量预算的情况下降低任务的执行时间以及给定任务执行的截止时间情况下降低预期的能耗两个问题[56]，但这些研究都只是面向单个处理器平台；文献[57~60]研究了多处理器平台下同样的问题，考虑了时间以及能量约束，但针对的主要是独立任务集。这些研究虽然提供了一些 DVFS 技术应用在任务调度上的思路，但不适用于本文研究的背景，本文更关心的是具有任务优先关系的调度问题研究。

1. 不关闭处理器的低能耗调度

Zong 等人以 DAG 为任务集模型，研究了同构系统中考虑任务复制的能量感知调度，并提出能量感知的复制以及性能-能耗平衡的复制两种算法，实现系统中性能与能量的综合考虑[61]。Lee 等人将调度长度和能耗作为优化目标，提出了异构分布式计算系统中能量感知的调度算法(Energy conscious scheduling，ECS) [23]。ECS 算法实现了双目标优化，将调度长度比以及能量利用率作为性能指标，实现了能耗以及调度长度之间的平衡优化，该研究也被后续多位学者所参考与对比。后来大多数研究重点考虑尽可能地优化其中一项目标，比如在满足应用不超过截止时间的条件下去实现能耗的最小化。

在文献[62]中，Huang 等人研究了异构分布式系统中并行应用调度长度约束下的能耗优化，提出了 EES(Energy-efficient Scheduling)算法。其基本思想是先用异构最早完成时间算法 HEFT 的思想对任务进行优先级排序以及处理器的

分配，获取应用的调度长度 SL_{HEFT}，然后根据应用的实际截止时限 T_{deadline} 与 SL_{HEFT} 之间的差值，基于 DVFS 技术，在保证应用在截止时限内完成的情况下适当地延长应用的执行时间，计算任务的最早开始时间和最晚允许结束时间之间的处理器时间空隙，即任务的松弛时间。任务松弛时间计算的顺序是根据任务的完成时间降序排序。由于任务的执行时间被适当拉长（称之为拉伸操作），处理器可以以较低的频率执行，从而实现降低能耗的目的。然而 EES 算法仅仅考虑了“向上优化”的方式[63]，即从出口任务到入口任务的方向进行拉伸操作，因此 Xie 等人结合了“向上优化”与“向下优化”（从入口任务到出口任务）的方式进行能耗优化[63]。在“向下优化”阶段，将应用的截止时限转换成每个任务的截止时限，这样任务的执行时间可以尽可能放宽到其各自的截止时限，通过适当延长任务执行时间降低能耗；在“向上优化”阶段，考虑同一处理器上相邻任务之间仍然存在松弛时间，可继续进行拉伸操作，从而进一步降低能耗。然而上述研究中对松弛时间的操作都是基于局部处理器，任务的拉伸仅限于当前处理器上进行。因此在文献[42]中，作者又提出了一种全局的能耗优化策略，即对任务的松弛时间计算时，考虑了将任务转移到其它处理器上的节能情况。在不影响任务优先级约束的前提下，通过将部分任务转移到其余处理器上执行并重新计算任务松弛时间以进行拉伸操作，相比只在当前处理器上拉伸更加有效。

2. 关闭处理器的低能耗调度

除了对任务进行拉伸操作降低能耗之外，Tang 等人引入了处理器关闭的策略，提出了相应的 DEWTS (DVFS-enabled Efficient energy Workflow Task Scheduling)算法[64]。算法过程分为三个阶段，首先进行任务与处理器的初步映射，和大多数研究一样，该阶段仍然采取 HEFT 算法求取应用的初始调度长度 MS_{HEFT}，根据给定的松弛系数 α 计算截止时间；然后根据前一阶段的调度结果，计算每个处理器上的任务调度情况，按照任务的分配数量进行降序排序，在不超过截止时间的前提下，关闭尽可能多的任务分配数比较少的处理器，保留任务数比较多处理器；最后计算保留下来的处理器上的任务松弛时间，利用 DVFS 技术对该部分任务拉伸，实现能耗优化。但是并不是关闭越多的处理器，能耗就会降的越多。被关闭的处理器上分配的任务需要转移到保留的处理器上执行，这会增加保留处理器的能耗，并且由于异构处理器能效的不同，增加的能耗可能比该任务在原处理器上的能耗要多。因此，应该以能否降低系统总能耗为目标来选择要被关闭的处理器。在文献[65]中，作者对上述工作做了深入分析与优化，提出了能量感知的处理器合并算法(Energy-aware Processor Merging, EPM)，该算法总能找到最能节省能量的处理器并进行关闭，实验结果相比 DEWTS 算法更优，但是引入了较高的时间复杂度。也就是说，当应用规模比较大时，EPM 算法并不高效。在此基础上，文献[65]又进一步优化了 EPM 算法，提出一种适

用于大规模并行应用的快速的处理器合并算法(Quick Energy-aware Processor Merging, QEPM)，实现了能量节约与算法时间复杂度之间的平衡。

3. 能量约束与时间优化

以上研究都是将能耗作为优化目标，以实现能耗最小化为目的，而能量约束下以调度长度为优化目标的的研究相对要少一些。虽然文献[56]和[57]以及[59]都考虑了将能量作为条件去满足并减小任务集的执行时间，但这些研究都是针对独立的任务集。在文献[66]中，Li 研究了具有优先关系的任务集并且有时间以及能量约束的调度问题，将问题分解为三个子问题：优先约束、任务调度、能量供给，并给出三种调度策略以及相应的算法：优先确定能量供给然后调度任务、优先进行任务调度然后考虑能量供给以及同时综合考虑任务调度以及能量供给。但是该研究考虑的是同构处理器平台。

在文献[32]中， Xiao 等人研究了异构分布式系统中能耗约束下的调度长度最小化问题，并提出了一种能量预分配的策略来保证系统应用的总能耗不超过给定的能量约束。能量约束下的调度长度最小化问题又被划分为两个子问题，一是先满足能量约束，二是降低调度长度。第一个子问题通过给应用中每个任务预分配一个能量最小值来解决，根据任务向上排序值的降序顺序，将系统应用的能量约束转换为每个任务各自的能量约束，进而完成能量的分配。第二个子问题借鉴了 HEFT 算法的思想，在任务完成能量分配后，按照任务最早结束时间为任务分配处理器，获得最小的调度长度。然而给任务预分配最低能量值具有一定的悲观性，从最终的分配结果来看，优先级高的任务最终获得的能量较多，而优先级较低的任务只能分配到较少的能量，这样会导致能量的分配不均，低优先级任务会显著增加最终的调度长度。

综上所述，关于能耗以及调度长度的优化研究主要有两条路线，满足时间约束前提下降低能耗以及满足能量约束前提下降低调度长度。关于前者的研究已经有很多，这一类研究属于能量高效的调度研究，而后者考虑到了总能量限制，并根据能量多少进行任务调度，属于能量感知的调度研究，也是本文要研究的问题，关于这部分的研究目前要么只考虑了单处理器或者独立任务的情景，要么考虑的是同构平台，真正与本文 ACPS 背景最相关的是文献[32]，但是其结果比较悲观，还有进一步提升的空间，这也是本文将要进行的工作之一。

2.4.3 能耗与可靠性优化

在 ACPS 中，可靠性是一项非常重要的性能指标，系统可靠性高意味着故障率低。系统的故障通常分为两类：永久故障和瞬时故障，其中瞬时故障是比较常见的，也是研究的热点问题。一般来说，系统的瞬时故障可用平均到达率为 λ 的

泊松分布来描述[67]。而在利用 DVFS 技术进行能耗优化时，瞬时故障的平均到达率跟系统的频率有着一定的关系。

1. 能耗与可靠性优化

Zhu 等人针对能耗优化管理过程中处理器频率的降低对系统可靠性的影响进行了研究，建立了一个嵌入式系统中能量管理和可靠性之间的关系模型[24]，分析表明瞬时故障发生的频率随着能耗优化时频率的降低而增加，也就是说能耗的优化会降低系统的可靠性；接着作者又研究了抢占式最早截止时间优先(EDF)调度系统中的可靠性感知的能量管理(Reliability-aware Power Management, RA-PM)方法[25]，针对单处理器上的周期性实时任务，运用 DVFS 技术并充分利用任务执行的松弛时间，在保障任务可靠性不低于目标值的情况下实现最大化的节能。该团队在文献[26]进一步研究了 RA-PM 方法，并提出了针对不同任务的任意可靠性目标的保障算法，在保障任务级的可靠性目标下再降低能耗；此外作者在文献[27]研究了时间以及能量约束下最大化可靠性问题，其核心思想是进行合理的能量分配，并将其转换成对应的频率选择与分配问题。

然而上述这些研究都是针对简单的周期任务模型，而 ACPS 中的应用是分布式的功能应用，任务之间具有优先关系约束。在文献[28]中，Zhao 等人研究了满足分布式功能实时以及能量约束的可靠性最大化问题，但仍然针对的是单处理器系统，不适用于复杂的 ACPS 异构环境，但是一些基本的思想可以作为借鉴。Aupy 等人研究了满足分布式应用可靠性以及执行时间约束的前提下降低应用执行所消耗的能量[68]，但针对的是同构处理器。文献[69]和[70]研究了异构分布嵌入式系统中的实时任务调度，应用 DVFS 技术在满足时间约束的前提下降低能耗，采用容错方式提高可靠性，但都没有将能量作为约束并进行充分地利用以最大化可靠性。

2. 能量约束与可靠性优化

在文献[71]中，Zhang 等人研究了能耗优化和可靠性感知的调度算法，并引入了三个策略来解决可靠性最大化问题。一是可靠性感知的异构最早完成时间算法(RHEFT)，基本思想是在为每个任务选择合适的频率后，根据任务的最早完成时间来选择处理器并计算其可靠性。二是可靠性感知的关键路径在关键处理器上算法(RCPOP)，该算法优先将关键路径上的任务分配到关键处理器上（关键路径上的任务在该处理器上执行时的计算开销最小），然后计算可靠性值。第三个是能量感知的可靠性最大化算法(RMEC)，该算法旨在选取最佳的任务处理器组合以实现能耗优化以及可靠性最大化。在此基础上，作者在文献[72]中进一步考虑了能耗以及可靠性的优化问题，并采用了任务恢复技术；在文献[73]中针对异构系统中不同服务质量需求（时间、节能、可靠性），采用遗传算法解决异构分布式系统中能量节约以及可靠性增强的双目标优化问题。但是这些研究实际上

并没有真正考虑将系统能量作为限制条件，本质上还是实现低能耗以及可靠性优化。Xiao 等人在文献[33]中研究了异构分布式系统中给定能量限制并实现并行应用可靠性最大化，其基本思想是采用预分配的方法，给任务预分配最低能量值，从而在调度过程中实现将应用的能量约束转换成任务的能量约束，任务在能量的约束范围内选择能最大化当前任务可靠性的处理器去执行。但是该研究中给任务预分配最小能量具有一定的悲观性，会显著增加应用的调度长度，而对于 ACPS 中的功能应用来说，不能在应用截止时间内完成任务会带来严重的安全隐患。

3. 时间约束与可靠性优化

节能技术采用 DVFS 技术会对可靠性产生影响，同时也会影响应用的执行时间，因此应用的截止时间也是潜在的约束条件。响应时间最小化以及可靠性最大化是两个对立的优化目标[74]。文献[75]研究了异构多处理器系统中满足响应时间约束以及可靠性目标保障的调度问题，实现时间以及可靠性目标双重约束下的资源最小化研究，虽然考虑的也是 DAG 模型，但是没有考虑任务的通信时间。Xie 等人针对汽车分布式应用中的功能安全验证，解决了在满足响应时间约束下实现功能应用的可靠性最大化(Maximizing Reliability under Response time Requirement, MRRR)，基本思想是先在满足可靠性基本的前提下尽可能降低响应时间，接着在响应时间约束范围内计算任务的松弛时间，对任务进行处理器重分配以提高可靠性[40]。但是该文献没有考虑能耗因素。

综上所述，目前关于能耗约束下的可靠性优化研究大多面向单处理器或者简单的独立任务模型，针对异构分布式系统中具有优先关系约束的 DAG 应用模型的研究较少。与本文研究 ACPS 环境最相近的研究如文献[33]解决了能耗约束下的可靠性最大化问题，文献[40]解决了响应时间约束下的可靠性最大化问题，但还未发现同时考虑能耗约束以及响应时间约束的相关研究。

2.4.4 相关调度研究小结

通过前面两小节的综述分析可知，目前低能耗优化研究大多只考虑满足系统应用的截止时间等约束前提下进行，而实际应用中往往也要考虑在能量预算有限的情况下去做系统优化。尤其是在一些以电池驱动的嵌入式系统中，在系统设计阶段要对能量（能量也即资源）进行合理分配。考虑截止时间约束的低能耗调度可称为高能效调度，而将能量作为资源约束的调度可称为能量感知的调度，因为任务的调度会考虑到总能量资源的限制而采用不同的调度方式，以满足相应的约束条件。

当前关于能量约束下的调度算法研究不多，并且大部分还是面向传统的嵌入式系统实时任务调度。针对像 ACPS 这样的异构分布嵌入式系统，能找到的相

关研究很有限。（1）对于能量约束下的调度长度优化，文献[32]虽然提供了一种解决方案，但是其结果具有悲观性；（2）其次在能量约束下系统可靠性优化方面，文献[33]中的算法仍然带有悲观性，其优化了可靠性，但未考虑时间约束，而 ACPS 中功能应用的实时性也是一项非常重要的性能指标；文献[40]虽然考虑了时间约束提高可靠性，但未考虑能耗因素。

因此本文旨在解决当前能量约束下的调度算法存在的问题，综合考虑能量使用、实时性以及可靠性三个方面，提出更加有效的能量分配策略，在实时性以及可靠性方面都能得到良好的优化，满足应用的需求。

2.5 小结

本章首先介绍了 ACPS 的调度模型，分别举例描述了 ACPS 的系统结构、功能应用模型；接着针对能耗管理介绍了典型的能耗优化技术 DPM、DVFS；随后从分布式计算角度介绍了相关的调度算法并重点综述了与能耗优化相关的调度研究，包括能耗与调度长度优化研究以及能耗与可靠性优化研究；最后进行总结，分析现有研究的不足以及待解决或进一步优化的问题。

第3章 能量约束下调度长度优化

3.1 引言

通过前面几章的介绍可知，ACPS 可以看作是一个异构分布嵌入式系统，在异构分布嵌入式系统中，能耗与时间是两个相互依赖的优化目标。低能耗主要包括两条研究路线：一是在保证应用在截止时间内完成的前提下采用低能耗技术降低系统能耗；二是在有限的能量供应下，最小化应用的执行时间（DAG 中应用的调度长度）。本章节采用第二种研究思路，设计一种在有限能量约束下的调度长度最小化算法。

3.2 相关模型

在进行问题的描述以及算法的设计之前，本小节先介绍相关的一些模型，包括应用模型和能量模型。

3.2.1 应用模型

在第 2 章的相关研究中，已经对应用模型进行了比较详细的描述，并用 DAG 来进行抽象表示。这里以一个简单并且广泛使用的 DAG 示例来进行补充说明[54]。

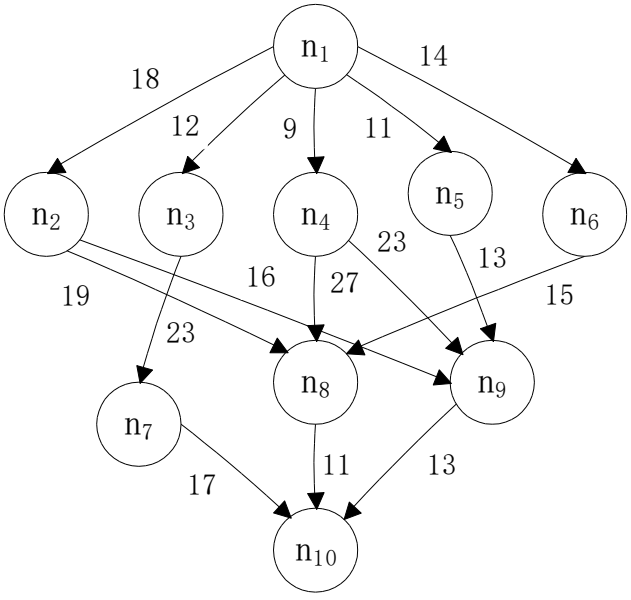


图 3.1 一个简单的 DAG 示例

以图 3.1 为例，该应用包含 10 个任务，执行在 3 个处理器上，任务在每个

处理器上执行所需要的时间如表 3.1 所示，例如从表 3.1 中可以看出 n_1 在 u_1 上执行时间为 14。由于处理器的异构性，同一个任务在不同处理器上执行所需要的时间是可以不同的。图 3.1 中的边表示任务之间的通信，例如 n_1 和 n_2 之间的边权值 18 表示对应的通信时间为 18，记为 $c_{1,2}$。

表 3.1 图 3.1 示例 DAG 中任务的执行时间矩阵

任务	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

3.2.2 能量模型

CMOS 电路中的供电电压和频率是近似线性关系，因此可以通过调节系统的频率来实现能耗的优化。本文采用一种广泛使用的系统级能量模型[25]。对于一个支持 DVFS 的系统，系统的功率与频率有着如下关系：

$$P(f) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + P_{\mathrm{d}}) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + C_{\mathrm{ef}} f^{m}). \tag{3.1}$$

其中，P_{s} 表示系统的静态功率，用来保持时钟运行并维持系统的基本组件。而系统的动态功率由两个部分组成：P_{ind} 和 P_{d}。P_{ind} 是跟频率无关的动态功率部分，只要系统未进入睡眠状态就一直存在，且是个常量；P_{d} 是和频率相关的动态频率部分，即会随着频率的变化而改变。h 表示系统的状态，$h=1$ 表示系统处于活跃状态，反之 $h=0$ 表示系统处于关闭或节能状态（没有任务执行）。C_{ef} 表示有效电容，m 表示功率指数（一般为 2~3 之间）。

由于静态功率只跟系统组件有关且不可操控，出于简化考虑，本文忽略该部分而只考虑动态部分。从公式(3.1)直观来看，只要降低频率 f，系统中频率相关的动态功率就会降低，进而可以达到降低能耗的目的。但是，频率的降低会导致任务的执行时间变长，这样就会消耗更多的静态能耗以及频率无关部分的动态能耗。因此需要计算出一个最低的有效频率 f_{ee}，这个频率可以由公式(3.2)得出。

$$f_{ee}=\sqrt[m]{\frac{P_{ind}}{(m-1)C_{ef}}}. \tag{3.2}$$

假定给定的处理器可调频率范围为$[f_{min}, f_{max}]$，则最低的有效频率为：

$$f_{low}=\max(f_{min}, f_{ee}). \tag{3.3}$$

这样在 $f_{low}\sim f_{max}$ 范围内，频率越低，能耗越低，我们称此范围为有效频率范围。由于处理器的异构性，每个处理器的有效频率范围是不一样的。对于大小为$|U|$的处理器集合，可以得到有效频率集合：

$$\left\{\begin{array}{l}\{f_{1,low}, f_{1,\alpha}, f_{1,\beta},..., f_{1,max}\},\\ \{f_{2,low}, f_{2,\alpha}, f_{2,\beta},..., f_{2,max}\},\\ ...,\\ \{f_{|U|,low}, f_{|U|,\alpha}, f_{|U|,\beta},... f_{|U|,max}\}\end{array}\right\}.$$

类似地，可以得到频率无关的动态功率集合：$\{P_{1,ind}, P_{2,ind},...P_{|U|,ind}\}$；频率相关的动态功率集合：$\{P_{1,d}, P_{2,d},...P_{|U|,d}\}$；有效电容集合：$\{C_{1,ef}, C_{2,ef},...C_{|U|,ef}\}$；动态功率指数集合：$\{m_1, m_2,...m_{|U|}\}$。这样，当任务 n_i 在处理器 u_k 上以频率 $f_{k,h}$ 执行时，其消耗的能量计算公式为：

$$E(n_i, u_k, f_{k,h})=(P_{k,ind}+C_{k,ef}\times f_{k,h}^{m_k})\times w_{i,k,h}. \tag{3.4}$$

其中 $w_{i,k,h}$ 表示任务 n_i 在处理器 u_k 上以频率 $f_{k,h}$ 执行所需要的时间，假设任务以最大频率执行所需时间为 $w_{i,k}$，则 $w_{i,k,h}$ 计算公式为：

$$w_{i,k,h}=\frac{w_{i,k}\times f_{k,max}}{f_{k,h}}. \tag{3.5}$$

3.3 问题描述

介绍完相关模型后，接下来对问题进行描述。给定一个功能应用 G 以及一系列支持 DVFS 的异构处理器集合 U，本章要解决的问题是在系统能量约束为 $E_{given}(G)$ 时，如何进行任务的能量以及处理器分配以获取最小的调度长度。应用的调度长度表示为：

$$SL(G)=AFT(n_{exit}). \tag{3.6}$$

$AFT(n_{exit})$ 表示出口任务 n_{exit} 的实际结束时间(Actual Finish Time, AFT)。本文做简化处理，忽略任务与任务之间通信能耗，也就是说本文目前只考虑任务执行过程中处理器的动态能耗。因此对于整个应用 G，其消耗的能量用数学公式表示为：

$$E(G)=\sum_{i=1}^{|N|}E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}). \tag{3.7}$$

这里的 $u_{pr(i)}$ 和 $f_{pr(i),hz(i)}$ 分别表示任务 n_i 所分配的处理器以及相应的执行频率，并且有 $f_{pr(i),\text{low}} \le f_{pr(i),hz(i)} \le f_{pr(i),\max}, \forall i: 1 \le i \le |N|, u_{pr(i)} \in U$ 。

综上，本章所研究问题形式化表述为：

$$\text{Minimize：}\ SL(G) = AFT(n_{\text{exit}}), \tag{3.8}$$

$$\text{Subject to：}\ E(G) = \sum_{i=1}^{|N|} E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \le E_{\text{given}}(G). \tag{3.9}$$

在处理器参数已知的情况下，可以计算出任务 n_i 在所有处理器上执行时所消耗的能量最小值 $E_{\min}(n_i)$ 与最大值 $E_{\max}(n_i)$ ，即：

$$E_{\min}(n_i) = \min_{u_k \in U} E(n_i, u_k, f_{k,\text{low}}), \tag{3.10}$$

$$E_{\max}(n_i) = \max_{u_k \in U} E(n_i, u_k, f_{k,\max}). \tag{3.11}$$

以及整个应用 G 所消耗的能量最小值与最大值，计算如下：

$$E_{\min}(G) = \sum_{i=1}^{|N|} E_{\min}(n_i), \tag{3.12}$$

$$E_{\max}(G) = \sum_{i=1}^{|N|} E_{\max}(n_i). \tag{3.13}$$

在这里，给定的能量应满足 $E_{\text{given}}(G) \in [E_{\min}(G), E_{\max}(G)]$。如果系统给定的能量比应用 G 所需的最低能量还低，则应用不可调度，因为能量无法满足整个应用调度的需求；如果给定的能量比应用 G 所需要的最大能量还高，则能量始终能满足整个应用调度的最大能量需求，换句话说，也就不存在所谓的能量约束问题。

3.4 问题分析与算法设计

经典的 HEFT 算法分为两步：先进行任务排序，然后根据任务最早结束时间为任务选择处理器。这里同样先对任务进行优先级排序，然后进行任务的能量分配，最后选择最优处理器。当然选择处理器时不仅要考虑任务最早结束时间，同时还要保证能耗不能超过给定的能量限制。接下来按步骤顺序进行分析。

3.4.1 任务优先级排序

本章采用 HEFT 算法中任务优先级排序方法，每个任务的优先级顺序由其向上排序值($rank_{\text{u}}$)决定， $rank_{\text{u}}$ 的计算方法如下：

$$rank_{\text{u}}(n_i) = \bar{w}_i + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_{\text{u}}(n_j)\}. \tag{3.14}$$

$\bar{w}_i$ 表示任务 n_i 在所有处理器上的平均执行时间，其计算公式为：

$$\bar{w}_i = \left(\sum_{k=1}^{|U|} w_{i,k} \right) / |U|. \tag{3.15}$$

表 3.2 中列出了图 3.1 DAG 示例中各个任务的向上排序值，从而可以得到该例子中任务的调度顺序为 $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$。

表 3.2 图 3.1 示例 DAG 中任务向上排序值

任务	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$rank_u$	108	70	80	80	69	63.3	42.7	35.7	44.3	14.7

3.4.2 能量分配

1、预分配最低能量

对于一个包含 N 个任务的 DAG 应用，假设任务经过向上排序过后的顺序为 $\{n_{s(1)}, n_{s(2)}, ..., n_{s(|N|)}\}$，初始状态下所有任务都处于未分配能量的状态，接着依次为这些任务分配能量。这里可以将任务分为三类：已经分配过能量的任务集合 $\{n_{s(1)}, n_{s(2)}, ..., n_{s(j-1)}\}$、当前将要分配能量的任务 $n_{s(j)}$ 以及未分配能量的任务集合 $\{n_{s(j+1)}, n_{s(j+2)}, ..., n_{s(|N|)}\}$。

现有的 MSLECC 算法是给未分配能量的任务集合中的每一个任务预分配最低的能量值 $E_{\min}(n_i)$ 以保证任务是可调度的（分配能量小于 $E_{\min}(n_i)$ 则任务无法在任何处理器上执行）[32]。因此，当给 $n_{s(j)}$ 分配能量时，应用总能耗为：

$$E_{s(j)}(G) = \sum_{x=1}^{j-1} E(n_{s(x)}, u_{pr(s(x))}, f_{pr(s(x)),hz(s(x))}) + E(n_{s(j)}, u_k, f_{k,h}) + \sum_{y=j+1}^{|N|} E_{\min}(n_{s(y)}). \tag{3.16}$$

并且由于要满足能量约束，有：

$$E_{s(j)}(G) \le E_{\text{given}}(G). \tag{3.17}$$

由公式(3.16)和(3.17)可得：

$$E(n_{s(j)}, u_k, f_{k,h}) \le E_{\text{given}}(G) - \sum_{x=1}^{j-1} E(n_{s(x)}, u_{pr(s(x))}, f_{pr(s(x)),hz(s(x))}) - \sum_{y=j+1}^{|N|} E_{\min}(n_{s(y)}). \tag{3.18}$$

因此便可以得到任务 $n_{s(j)}$ 的能量约束为：

$$E_{\text{given}}(n_{s(j)}) = E_{\text{given}}(G) - \sum_{x=1}^{j-1} E(n_{s(x)}, u_{pr(s(x))}, f_{pr(s(x)),hz(s(x))}) - \sum_{y=j+1}^{|N|} E_{\min}(n_{s(y)}). \tag{3.19}$$

这样，当给任务分配处理器以及确定相应的频率时，只需要满足每个任务的能耗不超过 $E_{\text{given}}(n_{s(j)})$ 即可，这样总能耗一定会满足应用的能量约束条件。但是这样的能量分配方法具有悲观性。给未分配能量的低优先级任务分配最低能量会

带来一定的不公平性，因为高优先级任务会优先分配较多的能量，而低优先级任务在能量不够的情况下只能以最低能量执行任务，这样会显著增加总的调度长度。因此需要一种更客观且有效的能量分配方法。

2、预分配平均能量

在介绍本章能量预分配方法之前，先定义一个概念：可用能量(Available Energy)。

定义 3.1：可用能量(Available Energy)。本章定义可用能量为系统给定的能量约束值与应用所需最低的能量之间的差值，用数学公式表示如下：

$$\Delta E_{\text{ae}}(G) = E_{\text{given}}(G) - E_{\min}(G). \tag{3.20}$$

可用能量的平均分配为：

$$E_{\text{aa}}(n_i) = E_{\min}(n_i) + \frac{\Delta E_{\text{ae}}(G)}{|N|}. \tag{3.21}$$

考虑到任务消耗能量的上限，最终给任务的预分配能量为：

$$E_{\text{pre}}(n_i) = \min\{E_{\text{aa}}(n_i), E_{\max}(n_i)\}. \tag{3.22}$$

根据上述分配方法，当给任务 $n_{s(j)}$ 分配能量时，有

$$E_{s(j)}(G) = \sum_{x=1}^{j-1} E(n_{s(x)}, u_{pr(s(x))}, f_{pr(s(x)),hz(s(x))}) + E(n_{s(j)}, u_k, f_{k,h}) + \sum_{y=j+1}^{|N|} E_{\text{pre}}(n_{s(y)}). \tag{3.23}$$

并且该能量分配算法一定是合理的，即保证任务可调度且总能量不超过其约束限制。下面给出相应的证明过程。

定理 3.1：当采用上述方法给未分配能量的任务预分配 $E_{\text{pre}}(n_i)$ 时。应用 G 中的每一个任务 $n_{s(j)}$ 始终能找到一个处理器及相应频率满足公式(3.24)。

$$E_{s(j)}(G) = \sum_{x=1}^{j-1} E(n_{s(x)}) + E(n_{s(j)}) + \sum_{y=j+1}^{|N|} E_{\text{pre}}(n_{s(y)}) \le E_{\text{given}}(G). \tag{3.24}$$

$n_{s(x)}$ 表示已分配能量的任务，$n_{s(j)}$ 表示当前要分配能量的任务，$n_{s(y)}$ 表示未分配能量的任务。

证明：这里采用数学归纳法证明定理 3.1。当给第一个任务 $n_{s(1)}$ 分配能量时，公式(3.24)对应形式为：

$$E_{s(1)}(G) = E(n_{s(1)}) + \sum_{y=2}^{|N|} E_{\text{pre}}(n_{s(y)}) \le E_{\text{given}}(G). \tag{3.25}$$

根据公式(3.22)有 $E_{\text{pre}}(n_i) \le E_{\text{aa}}(n_i)$，因此可得：

$$E_{s(1)}(G) = E(n_{s(1)}) + \sum_{y=2}^{|N|} E_{\text{pre}}(n_{s(y)}) \le E(n_{s(1)}) + \sum_{y=2}^{|N|} E_{\text{aa}}(n_{s(y)}). \tag{3.26}$$

将公式(3.20)、(3.21)代入公式(3.26)有：

$$E(n_{s(1)})+\sum_{y=2}^{|N|}E_{\text{aa}}(n_{s(y)})=E(n_{s(1)})+E_{\text{given}}(G)-E_{\text{aa}}(n_{s(1)}). \tag{3.27}$$

因此当 $E(n_{s(1)})\le E_{\text{aa}}(n_{s(1)})$ 时，公式(3.28)总能成立。此时 $n_{s(1)}$ 可以分配一个 $[E_{\min}(n_{s(1)}),E_{\text{aa}}(n_{s(1)})]$ 范围内的能量值，保证可调度性。

$$E(n_{s(1)})+E_{\text{given}}(G)-E_{\text{aa}}(n_{s(1)})\le E_{\text{given}}(G). \tag{3.28}$$

结合公式(3.26)、(3.27)和(3.28)可得到公式(3.25)，即定理在 j=1 时满足。

接着假设第 j 个任务 $n_{s(j)}$ 能够找到一个处理器及对应频率满足公式(3.24)，即：

$$E_{s(j)}(G)=\sum_{x=1}^{j}E(n_{s(x)})+\sum_{y=j+1}^{|N|}E_{\text{pre}}(n_{s(y)})\le E_{\text{given}}(G). \tag{3.29}$$

变换可得：

$$\sum_{x=1}^{j}E(n_{s(x)})\le E_{\text{given}}(G)-\sum_{y=j+1}^{|N|}E_{\text{pre}}(n_{s(y)}). \tag{3.30}$$

当给第 j+1 个任务 $n_{s(j+1)}$ 分配能量时，总能耗为：

$$E_{s(j+1)}(G)=\sum_{x=1}^{j}E(n_{s(x)})+E(n_{s(j+1)})+\sum_{y=j+2}^{|N|}E_{\text{pre}}(n_{s(y)}). \tag{3.31}$$

将公式(3.30)代入(3.31)可得到：

$$E_{s(j+1)}(G)\le E_{\text{given}}(G)+E(n_{s(j+1)})-E_{\text{pre}}(n_{s(j+1)}). \tag{3.32}$$

因此当 $E(n_{s(j+1)})\le E_{\text{pre}}(n_{s(j+1)})$ 时，公式(3.33)总能满足。此时 $n_{s(j+1)}$ 可以分配一个 $[E_{\min}(n_{s(j+1)}),E_{\text{pre}}(n_{s(j+1)})]$ 范围内的能量值，保证可调度性。

$$E_{s(j+1)}(G)\le E_{\text{given}}(G). \tag{3.33}$$

也就是说 j+1 时，定理同样正确。因此综上可知，定理 3.1 得证，即所提出的的能量分配方法可行，证毕。

当给任务 $n_{s(j)}$ 分配能量时，需满足：

$$E(n_{s(j)})\le E_{\text{given}}(G)-\sum_{x=1}^{j-1}E(n_{s(x)})-\sum_{y=j+1}^{|N|}E_{\text{pre}}(n_{s(y)}). \tag{3.34}$$

即得到任务 $n_{s(j)}$ 的能量约束：

$$E_{\text{given}}(n_{s(j)})=E_{\text{given}}(G)-\sum_{x=1}^{j-1}E(n_{s(x)})-\sum_{y=j+1}^{|N|}E_{\text{pre}}(n_{s(y)}). \tag{3.35}$$

考虑到实际情况中任务 $n_{s(j)}$ 的的能量使用不可能超过 $E_{\max}(n_{s(j)})$，因此给任务 $n_{s(j)}$ 分配的能量上限为：

$$E_{\text{up}}(n_{s(j)})=\min\{E_{\text{given}}(n_{s(j)}),E_{\max}(n_{s(j)})\}. \tag{3.36}$$

3.4.3 调度长度优化

完成能量的分配也即意味着将应用的能量约束转换成了每个任务的能量约束，接下来要做的便是在保证任务不超过能量约束的情况下，为其找到最优的处理器及频率的组合。利用任务最早结束的优化思想，根据任务的最早结束时间来进行处理器的选择以降低应用的调度长度。

最早开始时间(Earliest start time，EST)：$EST(n_i,u_k,f_{k,h})$表示任务n_i在处理器u_k上以频率$f_{k,h}$执行的最早开始时间，计算方式如下：

$$\begin{cases} EST(n_{\text{entry}},u_k,f_{k,h})=0 \\ EST(n_i,u_k,f_{k,h})=\max\left\{avail[k],\ \max\limits_{n_x\in pred(n_i)}\{AFT(n_x)+c'_{x,i}\}\right\} \end{cases}. \tag{3.37}$$

对于入口任务n_{entry}，其最早开始时间为0,$avail[k]$表示处理器u_k上任务可执行的最早可用时间。$AFT(n_x)$表示任务n_x的实际结束时间，$c'_{x,i}$表示任务n_x和n_i之间的通信时间，且当n_x和n_i分配到不同处理器上时有$c'_{x,i}=c_{x,i}$，否则$c'_{x,i}=0$。

最早结束时间(Earliest finish time，EFT)：$EFT(n_i,u_k,f_{k,h})$表示任务在处理器u_k上以频率$f_{k,h}$执行的最早结束时间，计算方式如下：

$$EFT(n_i,u_k,f_{k,h})=EST(n_i,u_k,f_{k,h})+w_{i,k,h}. \tag{3.38}$$

任务的最早结束时间 EFT 是为任务选择处理器的标准，为任务选择能获得最早结束时间的处理器可以实现局部最优的调度长度优化。

3.4.4 算法过程

综合上面的描述可总结得出能量约束下的高效调度算法(Efficient Scheduling with Energy Consumption Constraint，ESECC)的整个过程，主要包括能量分配以及调度长度优化两个阶段。具体过程如算法 3.1 所示。

算法的关键是给未分配能量的任务选择一个合适的预分配能量值，从而尽可能地弱化高优先级与低优先级任务之间能量分配的不均问题，降低最小能量分配的悲观性。算法的详细解释如下：

（1）第 1 行，计算任务的向上排序值$rank_{\text{u}}$，并按照向上排序值降序排序，生成降序列表$dlist$。

（2）第 2-5 行，计算任务的最小/大能量值以及应用的最小/大能量值。

（3）第 6-8 行，按照向上排序值降序顺序，为每个任务计算预分配能量值。

（4）第 12-26 行，遍历所有处理器以及相应的频率，根据任务的最早结束时间来选择最优的组合。

（5）第 28-29 行，计算最终实际消耗的能量以及调度长度。

算法 3.1: The ESECC Algorithm

Input: $U=\{u_1, u_2, \ldots, u_{|U|}\}$, G, $E_{\text{given}}(G)$
Output: $SL(G)$, $E(G)$
1: Sort the tasks in a list *dlist* by descending order of $rank_u$ values;
2: **for all** ($n_i \in N$) **do**
3: Calculate $E_{\min}(n_i)$ and $E_{\max}(n_i)$ using Eqs. (3.10) and (3.11), respectively;
4: **end for**
5: Calculate $E_{\min}(G)$ and $E_{\max}(G)$ using Eqs. (3.12) and (3.13), respectively;
6: **for all** (n_i in *dlist*) **do**
7: Calculate $E_{\text{pre}}(n_i)$ using Eq (3.22);
8: **end for**
9: **while** (tasks in *dlist*) **do**:
10: $n_i = dlist.out()$;
11: Calculate $E_{\text{given}}(n_i)$ using Eq. (3.35);
12: **for all** ($u_k \in U$) **do**
13: **for all** ($f_{k,h} \in [f_{\text{low}}, f_{\max}]$ in ascending order) **do**
14: Calculate $E(n_i, u_k, f_{k,h})$ using Eq. (3.4);
15: **if** ($E(n_i, u_k, f_{k,h}) \le E_{\text{up}}(n_i)$) **then**
16: $f_{k,hz(i)} \leftarrow f_{k,h}$;
17: **end if**
18: **end for**
19: Calculate $EFT(n_i, u_k, f_{k,h})$ using Eq. (3.38);
20: **if** ($EFT(n_i, u_k, f_{k,h}) < AFT(n_i)$) **then**
21: $pr(i) \leftarrow k$;
22: $f_{pr(i),hz(i)} \leftarrow f_{k,hz(i)}$;
23: $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leftarrow E(n_i, u_k, f_{k,hz(i)})$;
24: $AFT(n_i) \leftarrow EFT(n_i, u_k, f_{k,hz(i)})$;
25: **end if**
26: **end for**
27: **end while**
28: Calculate the actual energy consumption $E(G)$ using Eq. (3.7);
29: Calculate the schedule length $SL(G)$ using Eq. (3.6);
30: **return** $E(G)$ and $SL(G)$.

算法中主要的时间复杂度集中在第 9-27 行。遍历 N 个任务的时间复杂度为 $O(|N|)$，第 12-26 行根据任务的最早结束时间选择最优处理器及频率组合的时间复杂度为 $O(|N|\times|U|\times|F|)$，$|F|$表示频率集合的大小。由于这两部分是嵌套关系，所以算法最终的时间复杂度为 $O(|N|^2\times|U|\times|F|)$，相比现有算法如 MSLECC[32]，其复杂度也是 $O(|N|^2\times|U|\times|F|)$，因此 ESECC 算法的复杂度并没有增加，它也是一个低复杂度的算法。

接下来用图 3.1 中的例子来验证说明所提出的算法。处理器的详细参数如表 3.3 所示。在这个例子中，我们设置处理器的最大频率为 1.0 且精度为 0.01，根据已知数据可以计算得到该应用所消耗的最大能量值为 161.99。这里能量约束的设置和文献[32]中一样，设置为 $E_{\text{given}}(G) = 0.5\times E_{\max}(G) = 80.995$，以便更客观地进行比较。

表 3.3 示例处理器参数值

u_k	$P_{k,\mathrm{ind}}$	$C_{k,\mathrm{ef}}$	m_k	$f_{k,\min}$	$f_{k,\max}$
u_1	0.03	0.8	2.9	0.22	1.0
u_2	0.04	0.8	2.5	0.21	1.0
u_3	0.07	1.0	2.5	0.19	1.0

运行算法 ESECC 后的调度结果如表 3.4 所示。根据表中数据可以得到最终实际的能耗值为 74.6252，低于能量约束值 80.995，同时调度长度为 84.033。之所以会出现能量剩余，是因为任务实际执行所需能量可能比其能量约束值少，即有可能出现最后几个任务的能量约束值大于其所需要的最大能量，因此就会产生能量结余。图 3.2 展示了相应的调度结果甘特图，图中的箭头表示具有通信关系的任务之间的通信。

表 3.4 ESECC 算法调度 DAG 示例的调度结果

n_i	$E_{\mathrm{given}}(n_i)$	$u(n_i)$	$f(n_i)$	$AST(n_i)$	$AFT(n_i)$	$E(n_i)$
n_1	8.5499	u_3	0.91	0	9.8901	8.5051
n_3	8.0628	u_1	0.93	21.8901	33.7181	8.0214
n_4	8.1888	u_2	1.00	18.8901	26.8901	6.7200
n_2	9.8414	u_3	0.56	9.8901	42.0330	9.7932
n_5	8.2436	u_2	0.81	26.8901	42.9395	8.2236
n_6	8.3925	u_1	0.86	33.7181	48.8344	8.2622
n_9	9.3174	u_2	0.94	58.0330	70.7990	9.2597
n_7	7.3667	u_1	1.00	48.8344	55.8344	5.8100
n_8	8.5112	u_1	1.00	61.0330	66.0330	4.1500
n_{10}	12.2487	u_2	1.00	77.0330	84.0330	5.8800

$$E(G)=\sum_{i=1}^{10}E(n_i)=74.6252,\ SL(G)=AFT(n_{10})=84.033$$

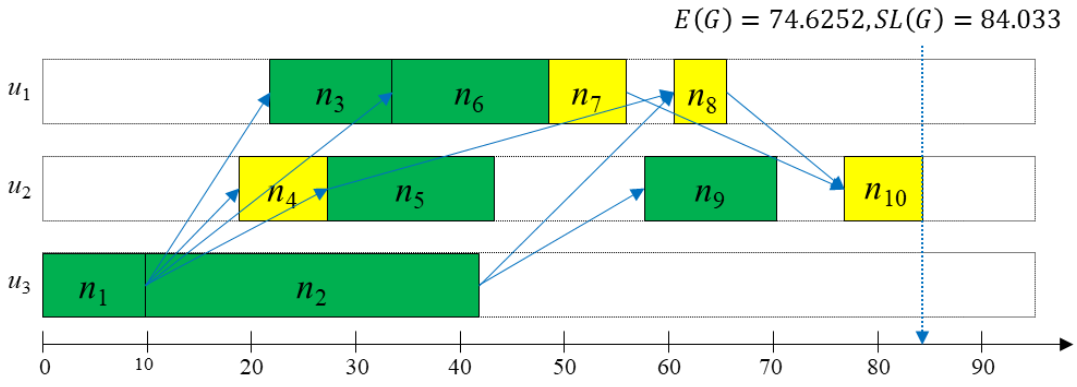


图 3.2 ESECC 算法调度 DAG 示例的调度甘特图

针对同一例子，我们拿最相近的 MSLECC 算法结果进行比较，表 3.5 列出了相关的能耗以及调度长度。从结果可以看出，ESECC 不仅消耗了较少的能量，并且还降低了应用的调度长度（降低了 35%左右）。

表 3.5 MSLECC 与 ESECC 调度结果对比

算法	$E(G)$	$SL(G)$
MSLECC	80.9939	129.3660
ESECC	74.6252	84.0330

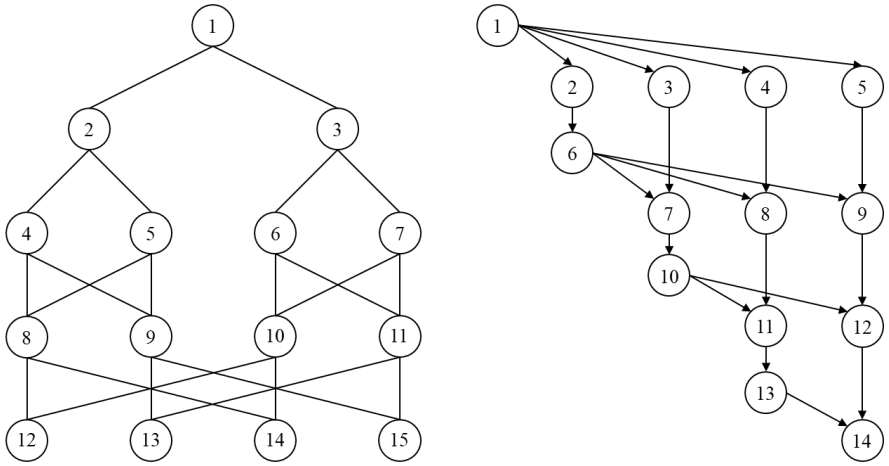
3.5 实验及分析

为了更好地说明算法的优越性，接下来做进一步的对比实验。对比的算法除了 MSLECC，本小节还将 HEFT 算法的结果作为参考，因为 HEFT 算法是一个经典的广泛使用的算法，具有较高的性能。换句话说，在不考虑能耗的情况下，HEFT 算法可以获得最小的应用调度长度，因此可以将它的调度长度作为参考标准。这里主要对比每个算法实际消耗的能量以及调度长度，并且采用实际的并行应用模型来做实验。

3.5.1 实验设置

本文采用仿真的方法进行实验。实验的硬件环境为一台 4 核 Intel i7-6770 处理器，8G 内存的个人电脑，系统为 Windows 10。仿真实验用的软件工具为 MyEclipse 2014，使用的编程语言为 Java。这里参考文献[63]中的实验设置本章所需的处理器以及应用的参数值范围：$10\mu s \le w_{i,k} \le 100\mu s$, $10\mu s \le C_{i,j} \le 100\mu s$, $0.03 \le P_{k,\text{ind}} \le 0.07$, $0.8 \le C_{k,\text{ef}} \le 1.2$, $2.5 \le m_k \le 3.0$, $f_{k,\max}$ =1GHz。所有的频率是离散的，精度为 0.01 GHz。异构处理器数量为 32 且参数值随机生成。

本章采用实际并行应用模型如快速傅里叶变换(Fast Fourier transformation, FFT)应用和高斯消元(Gaussian elimination, GE)应用进行实验[54]，这两种应用分别代表了高并行与低并行应用，相关的拓扑图如图 3.3 所示。图 3.3a 展示了一个快速傅里叶变换应用的例子，对应的 ρ 为 4；图 3.3b 展示了一个高斯消元应用的例子，对应的 ρ 为 5。ρ 在这里是一个用来描述应用规模的参数，ρ 越大表示应用规模越大，即对应 DAG 中结点数越多。对于 FFT 图而言，任务数量与 ρ 的关系为$|N|=(2\times\rho-1)+\rho\times\log_2\rho$，这里的 ρ 取 2 的指数级数字，即 2,4,8,16,...，对应 FFT 图中结点数为 15,31,63,127,...；在 GE 图中，任务数量与 ρ 的关系为$|N|=(\rho^2+\rho-2)/2$，这里的 ρ 取正整数。为了适应前文算法中的调度模型，对于有多个出口任务的 DAG，本文给这些出口任务额外添加一个虚拟的后继结点作为新的出口任务，并且该任务结点的计算时间和相应的通信时间为 0。



a b

图 3.3 快速傅里叶变换与高斯消元应用

3.5.2 实验过程及结果分析

1）变化的能量约束

首先考虑相同的并行应用在不同程度的能量约束下的实验情况，主要包括两个实验，详细描述如下。

实验 1：FFT 图（变化的能量约束）

该实验采用 FFT 应用图，对同一规模大小的 FFT 图做不同能量约束下的实验，比较不同调度算法所产生的能耗值（单位：Wμs）以及调度长度（单位：μs）。FFT 图的规模大小设置为 ρ=64，即总的任务数为 511。为了更方便地比较，这里以 HEFT 算法调度结果的能耗值为标准，记为 $E_{\mathrm{HEFT}}(G)$，能量的约束范围设置为 $E_{\mathrm{HEFT}}(G)$ ×0.5 ~ $E_{\mathrm{HEFT}}(G)$ ×0.9，即从严格的能量约束到宽松的能量约束。

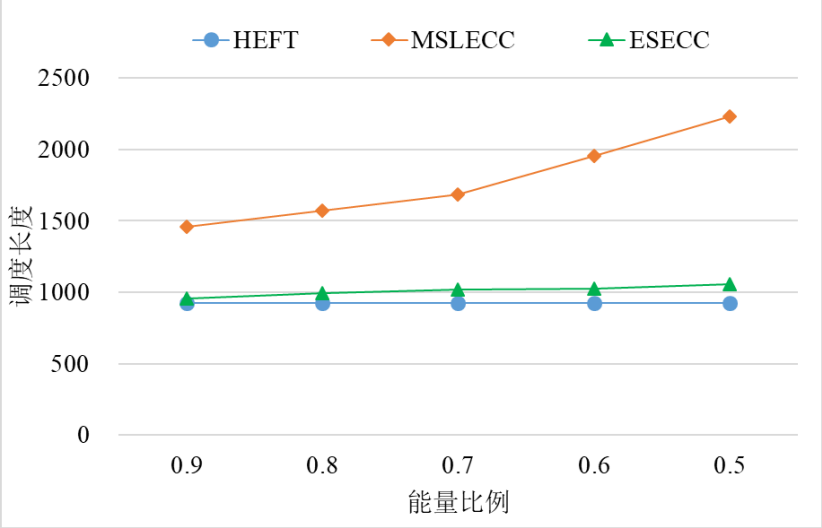


图 3.4 不同能量约束下 FFT 图调度长度对比

表 3.6 不同能量约束下 FFT 图调度结果

$E_{given}(G)$	HEFT		MSLECC		ESECC	
	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
10406.60	11562.89	924.00	10406.60	1458.83	10244.50	952.93
9250.31	11562.89	924.00	9250.31	1574.00	9183.69	996.71
8094.02	11562.89	924.00	8094.02	1688.13	8088.34	1016.17
6937.73	11562.89	924.00	6937.73	1957.15	6932.45	1027.00
5781.45	11562.89	924.00	5781.44	2231.28	5781.30	1053.47

图 3.4 显示的是使用三种算法调度 FFT 应用在不同能量约束下的调度长度变化曲线，三种算法调度结果的详细的数据值见表 3.6。从表中数据可以看到，ESECC 算法和 MSLECC 算法调度的能耗都能满足能量约束条件，并且 ESECC 算法在降低调度长度方面表现更好，和 HEFT 相比也比较接近。特别地，当约束条件比较苛刻时（即给定的能量比较少），ESECC 相比 MSLECC 提升更明显。例如表 3.6 所示，当能量约束设定为 $E_{HEFT}(G)$ ×0.5 时，ESECC 生成的调度长度为 1053.47μs，而 MSLECC 的调度长度为 2231.28μs，ESECC 相比 MSLECC 在调度长度上降低了有 52.8%。之所以能降低这么多，是因为当能量比较少时，采用 MSLECC 算法中的能量分配方法会导致低优先级任务只能分配到最低能量，这样会严重拖长整个应用的调度长度。很明显，MSLECC 总是优先给高优先级任务（接近入口任务）分配较多的能量，也就是说它更多地是降低靠近 DAG 入口的任务的执行时间。需要注意的是，HEFT 算法只是用来作为一个标准，其不适用于能量约束下的调度环境，通过和 HEFT 算法的调度长度对比能更好地展示 ESECC 在降低调度长度方面的优越性。

实验 2：GE 图（变化的能量约束）

该实验采用 GE 应用图，对同一规模大小的 GE 图做不同能量约束下的实验，比较不同调度算法所产生的能耗值以及调度长度。在这里，GE 图的规模大小被设置为 ρ=32，即相应的总任务数为 527，这是为了和实验 1 中 FFT 图的任务数更接近，这样可以近似地做一个横向的比较，比较两种应用图的并行效果。同样地，这里仍然以 HEFT 算法调度结果的能耗值为标准，能量的约束范围设置为 $E_{HEFT}(G)$ ×0.5 ~ $E_{HEFT}(G)$ ×0.9。

图 3.5 显示了 GE 应用在不同能量约束下的调度长度变化曲线，三种算法对比的详细数据值见表 3.7。通过和表 3.6 对比可以发现，相同算法在同一能量约束下，GE 应用的调度长度相比 FFT 应用要长，相差有 3 倍左右。这也验证了 FFT 应用的并行程度比 GE 应用要高的多。同时，ESECC 和 MSLECC 的对比也显示出 ESECC 具有更好的性能，相比 MSLECC 算法能产生更小的调度长度，

当能量约束为 $E_{HEFT}(G)$ ×0.9 时，其调度长度是 MSLECC 算法的 83.4%，而当能量约束为 $E_{HEFT}(G)$ ×0.5 时，为 MSLECC 算法的 70%。同样说明了能量越少，ESECC 的优势更明显。

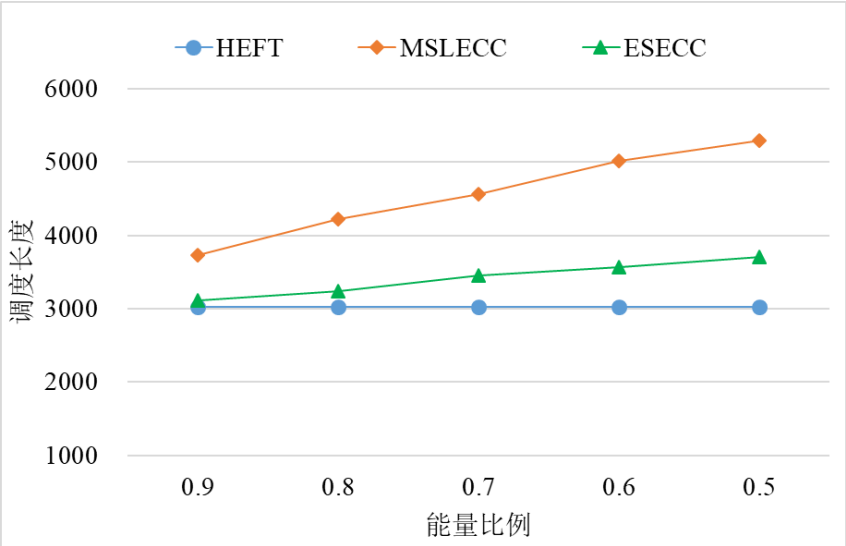


图 3.5 不同能量约束下 GE 图调度长度对比

表 3.7 不同能量约束下 GE 图调度结果

$E_{given}(G)$	HEFT		MSLECC		ESECC	
	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
11010.34	12233.71	3028.00	11010.34	3732.97	10982.10	3113.20
9786.97	12233.71	3028.00	9786.97	4221.89	9765.62	3237.30
8563.60	12233.71	3028.00	8563.60	4560.27	8556.34	3457.24
7340.23	12233.71	3028.00	7340.23	5009.08	7332.62	3560.89
6116.86	12233.71	3028.00	6116.85	5292.63	6114.13	3704.66

2）变化的任务数量

在这一部分，我们考虑不同规模的应用大小下算法的性能情况，和上一部分一样，依旧以 FFT 和 GE 应用为对象，从实际消耗的能量以及调度长度两个方面对比 ESECC、MSLECC 以及 HEFT 算法。

实验 3：FFT 图（变化的任务数量）

该实验采用 FFT 应用图，研究在不同应用规模下三种算法的性能。应用的规模大小参数 ρ 取 16、32、64、128、256，对应的任务数为 95、223、511、1151、2559，即包括了小规模的应用(ρ=16)到大规模的应用(ρ=256)。能量的约束设置为固定值，这里以 HEFT 算法调度结果实际需要的能量为标准，取其一半的能量为能量约束，即 $E_{HEFT}(G)$ ×0.5。

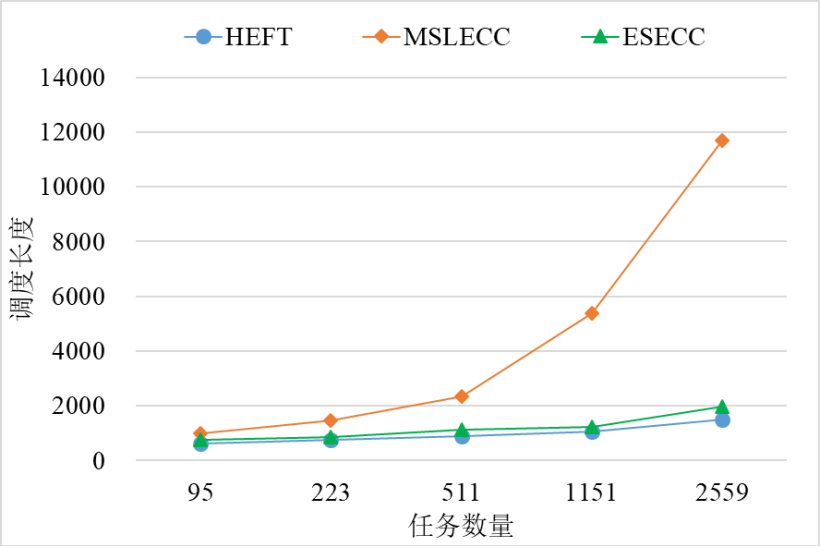


图 3.6 不同规模 FFT 图调度长度对比

图 3.6 显示了三种算法应用在不同规模大小的 FFT 图中的调度长度变化曲线，详细的数据值见表 3.8。可以看到 ESECC 和 MSLECC 算法在任何情况下都能满足能量约束条件，但是从曲线走势来看，当应用的规模比较大时，MSLECC 算法的结果悲观，调度长度非常大。如表 3.8 所示，当 ρ=256 时，任务数为 2559，此时 MSLECC 算法调度结果中调度长度为 11706.41μs，而 ESECC 算法的只有 1971.36μs，相比之下 ESECC 能在 MSLECC 基础上降低了 83.2%。即便和 HEFT 的调度长度相比也比较接近，而能耗却只有 HEFT 算法的 50%。这也说明了 ESECC 算法的表现性能非常好，尤其是在应用规模较大的情况下。

表 3.8 不同规模 FFT 图调度结果

$\|N\|$	$E_{\text{given}}(G)$	HEFT		MSLECC		ESECC	
		$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
95	1098.79	2197.58	624.00	1098.79	1006.78	1092.63	742.47
223	2569.80	5139.61	759.00	2569.80	1458.26	2569.71	865.10
511	5753.61	11507.21	902.00	5753.60	2350.80	5753.48	1121.31
1151	11390.77	22781.54	1078.00	11390.77	5396.74	11390.63	1236.92
2559	21452.59	42905.17	1502.00	21452.58	11706.41	21452.55	1971.36

实验 4：GE 图（变化的任务数量）

该实验采用了 GE 应用图，研究在不同应用规模下三种算法的性能。为了和 FFT 应用图做一个横向对比，在选取规模参数 ρ 时尽可能地做到 GE 应用的任务数和 FFT 应用任务数近似相同（由于两种应用图中任务数的计算公式不同，只能达到近似相等的任务数）。因此，GE 应用的规模大小参数 ρ 取 13、21、

31、47、71，与之对应的 GE 图中的任务数分别为 90、230、495、1127、2555，可以发现和实验 3 中 FFT 图中任务数比较接近。同时也包括了小规模的应用(ρ=13)到大规模的应用(ρ=71)。能量的约束同样被设置为 HEFT 算法调度结果实际需要能量的一半，即 $E_{\mathrm{HEFT}}(G)$ ×0.5。

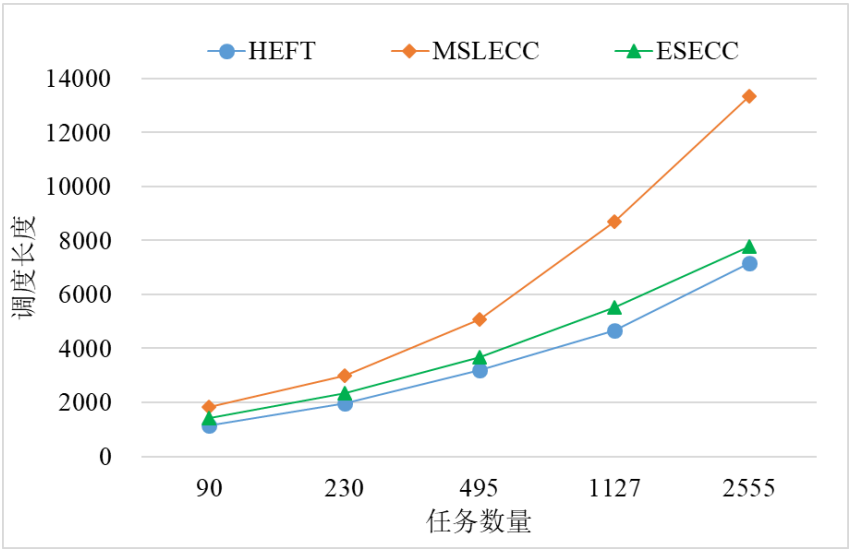


图 3.7 不同规模 GE 图调度长度对比

表 3.9 不同规模 GE 图调度结果

$\|N\|$	E_{given} (G)	HEFT		MSLECC		ESECC	
		E (G)	$SL(G)$	E (G)	$SL(G)$	E (G)	$SL(G)$
90	1016.02	2032.03	1150.00	1016.02	1819.37	1015.93	1435.30
230	2409.57	4819.14	1973.00	2409.57	3006.90	2409.46	2358.14
495	6240.72	12481.44	3210.00	6240.72	5084.34	6240.52	3662.51
1127	14756.24	29512.47	4661.00	14756.24	8696.26	14749.77	5510.80
2555	34575.24	69150.48	7174.00	34575.24	13332.84	34570.60	7791.77

图 3.7 显示了三种算法调度不同规模大小的 GE 图的调度长度变化曲线，相关详细数据值见表 3.9。从图中可以看到随着 GE 应用规模的增大，应用的调度长度显著增加，三条曲线的走向一致。但是在同一规模下比较 ESECC 和 MSLECC 算法，ESECC 还是要明显优于 MSLECC。当任务数为 2555 时，ESECC 算法调度长度为 7791.77μs，此时 MSLECC 为 13332.84μs，HEFT 为 7174μs，相比之下 ESECC 和 HEFT 结果比较接近，而 MSLECC 的结果接近 HEFT 的两倍。当任务数近似相同时，GE 应用的调度长度要比 FFT 应用调度长度大，也证明了其并行程度较低。

实验部分小结：

综合所有实验结果可以总结得出：ESECC 算法在不同能量约束以及不同规

模应用下的表现都要优于 MSLECC，并且部分实验情况中 ESECC 算法相较于 MSLECC 算法在降低调度长度方面有明显的优势；即便有能量约束的情况下，ESECC 算法的调度长度和 HEFT 也相当接近，这也验证了 ESECC 算法是一个高效的算法。

3.6 小结

如今绿色计算逐渐成为可持续发展的一种新型计算模式，而节能则是绿色计算的必要条件。ACPS 作为一个可与物理环境交互的异构分布嵌入式系统，其所需的能量主要通过电池供给，因此从计算的角度去设计低能耗的任务调度算法具有重要的意义。本章主要针对纯电动汽车中电池供给能量的局限性问题，设计一种异构分布式系统中能量约束下高效的调度算法，与传统低能耗调度以满足时间约束为前提并降低能耗不同，本章考虑的是在能量有限的情况下尽可能降低应用的调度长度，即提高性能，提出了相应的 ESECC 算法。通过数学推导以及对比实验分析验证了算法能在满足能量约束的情况下充分降低应用的调度长度，且比现有算法更加高效。相信 ESECC 算法的思想可以为 ACPS 中能量感知的系统应用设计提供一定的参考价值。

第4章 能量约束下可靠性优化

4.1 引言

ACPS 作为一个典型的异构分布嵌入式系统，除了低能耗以及实时性需求以外，系统的可靠性也非常重要。在第 3 章我们实现了系统中能量资源有限的情况下如何将 DAG 应用的调度长度尽可能降到最低，保证实时性需求，提高应用执行的性能，本章主要考虑 ACPS 中的可靠性优化问题。然而在降低能耗过程中所使用的 DVFS 技术会造成一定的瞬时故障，对系统的可靠性会产生相应的负面影响。为了同时解决能耗与可靠性问题，本章结合现有资料中可靠性与能耗优化中 DVFS 技术之间的关系模型，提出更加有效的有能量约束的可靠性增强算法。并以真实的汽车电子功能应用为对象，对所提出的算法进行实验验证分析。

4.2 相关模型

本小节主要介绍相关的一些模型如可靠性模型。和第 3 章一样，本章的应用模型仍然是用 DAG 来描述，并且继续采用第 3 章中的 DAG 示例作为例子来辅助说明，能量模型也和前面一章的相同，因此这里不再做相关描述。下面主要介绍可靠性模型，结合 ISO 26262 标准中相关的一些概念进行描述，并针对可靠性与能耗优化的关系进行分析。

4.2.1 可靠性模型

通常系统中的失效主要包括两种：一种是瞬时失效（如随机发生的硬件故障、软件 bug 等），另一种是永久性的失效。一般来说硬件失效以及软件 bug 是相对比较常见的，而本章主要考虑的就是这一类失效，也称瞬时故障。汽车电子功能安全 ISO 26262 标准中也明确了硬件随机失效在硬件的整个生命周期中是不可预见的，但存在一定的概率分布。ISO 26262 标准中定义了一个暴露率(Exposure)的概念，针对系统失效所造成危害场景，暴露率描述了人员暴露在相应危害场景中的概率，并且针对不同场景，ISO 26262 标准中的暴露率定义了不同的等级：E0，E1，E2，E3 以及 E4。E0 代表最低的级别（即几乎不可能发生）[21]，E4 代表最高级别（发生危害场景伤害的可能性很大）。虽然 ISO 26262 标准中没有定义可靠性的概念，但是可以根据给出的暴露率推断出相应的可靠性需求，用来描述系统无故障运行的概率。表 4.1 给出了相应的暴露率等级及对应的可靠性目标需求。

表 4.1 暴露率等级与对应的概率以及可靠性目标

暴露率等级	暴露率概率	可靠性目标
E1 极低概率	无明确规定	至少为 0.99
E2 低概率	<1%	≥0.99
E3 中等概率	[1%,10%]	>0.9
E4 高概率	>10%	≤0.9

对于一个系统来说，瞬时故障发生的概率服从泊松分布，它是概率统计中一种常见的离散概率分布[5]。对于一个长度为 t 的时间段内故障事件发生的概率，其表示公式为：

$$P_{\text{error}}(t)=\frac{(\lambda t)^{\gamma}}{\gamma!}e^{-\lambda t}. \tag{4.1}$$

λ表示事件的平均到达率，γ表示故障事件的个数。而可靠性表示无故障事件发生的概率，即γ=0。因此可以推导出可靠性的公式为：

$$R(t)=e^{-\lambda t}. \tag{4.2}$$

$R(t)$表示 t 时间内无故障发生的概率。而对于一个 DAG 来说，整个应用的可靠性表示为应用中所有任务都无故障正常执行的概率。一个包含 N 个任务的应用可靠性为：

$$R(G)=\prod_{n_i\in N}R(n_i). \tag{4.3}$$

4.2.2 能耗优化与可靠性关系

对于一个支持 DVFS 的系统，瞬时故障的平均到达率λ跟系统当前的频率有关。假设$\lambda_{k,\max}$表示u_k在最大频率$f_{k,\max}$下运行任务时的瞬时故障的平均到达率，根据文献[28]中的描述，瞬时故障到达率跟处理频率成指数关系，当任务在u_k上以$f_{k,v}$频率运行时，计算公式为：

$$\lambda_{k,v}=\lambda_{k,\max}10^{\frac{d(f_{k,\max}-f_{k,v})}{f_{k,\max}-f_{k,\min}}}. \tag{4.4}$$

d 是一个常量，表示失效率对电压/频率变化的敏感度。从公式(4.4)可以分析出，当处理频率$f_{k,v}$越小时，$\lambda_{k,v}$越大，这也说明随着节约能耗时频率的降低，故障发生率会提高。结合前面的章节，对于任务n_i，其在u_k上以频率$f_{k,v}$执行时的可靠性计算公式为：

$$R(n_i,u_k,f_{k,v})=e^{-\lambda_{k,\max}10^{\frac{d(f_{k,\max}-f_{k,v})}{f_{k,\max}-f_{k,\min}}}\times\frac{w_{i,k}\times f_{k,\max}}{f_{k,v}}}. \tag{4.5}$$

相应地，整个应用的可靠性计算为：

$$R(G)=\prod_{n_i\in N}R(n_i,u_{pr(i)},f_{pr(i),hz(i)}). \tag{4.6}$$

其中 $u_{pr(i)}$ 和 $f_{pr(i),hz(i)}$ 分别表示给任务 n_i 实际分配的 ECU 以及对应的频率。

4.3 问题描述

介绍完模型，接下来对本章要解决的问题进行描述。给定一个功能应用 G 以及一系列支持 DVFS 的异构处理器集合 U，本章研究的问题是在满足系统能量约束的条件下，如何进行任务的能量分配以及处理器的选择，在保证所有任务在截止时间 $D(G)$内完成的同时，进一步提升系统的可靠性。对于一个有 N 个任务的 DAG 抽象的应用，其所有任务执行消耗的能量为：

$$E(G)=\sum_{i=1}^{|N|}E(n_i,u_{pr(i)},f_{pr(i),hz(i)}). \tag{4.7}$$

整个应用的调度长度为出口任务的实际结束时间（Actual Finish Time, AFT）：

$$SL(G)=AFT(n_{\text{exit}}). \tag{4.8}$$

由于要根据应用的截止时间从出口任务向入口任务的方向进行时间松弛（4.4 小节会详细说明），这样调度结果中入口任务的实际开始时间（Actual Start Time, AST）可能不为 0，因此整个应用实际的响应时间为：

$$RT(G)=AFT(n_{\text{exit}})-AST(n_{\text{entry}}). \tag{4.9}$$

因此可以得到所要研究问题的形式化描述：

$$\text{Maxmize:}\quad R(G)=\prod_{n_i\in N}R(n_i,u_{pr(i)},f_{pr(i),hz(i)}), \tag{4.10}$$

$$\text{Subject to:}\quad E(G)=\sum_{i=1}^{|N|}E(n_i,u_{pr(i)},f_{pr(i),hz(i)})\le E_{\text{given}}(G), \tag{4.11}$$

$$\text{and}\quad RT(G)\le D(G). \tag{4.12}$$

在处理器参数已知的情况下，可以计算出任务 n_i 在所有处理器上执行时所消耗的能量最小值 $E_{\min}(n_i)$ 与最大值 $E_{\max}(n_i)$，即：

$$E_{\min}(n_i)=\min_{u_k\in U}E(n_i,u_k,f_{k,\text{low}}), \tag{4.13}$$

$$E_{\max}(n_i)=\max_{u_k\in U}E(n_i,u_k,f_{k,\max}). \tag{4.14}$$

接着可得到整个应用 G 所消耗的能量最小值 $E_{\min}(G)$ 与最大值 $E_{\max}(G)$，即：

$$E_{\min}(G)=\sum_{i=1}^{|N|}E_{\min}(n_i), \tag{4.15}$$

$$E_{\max}(G)=\sum_{i=1}^{|N|}E_{\max}(n_i). \tag{4.16}$$

给定的能量应满足 $E_{\text{given}}(G)\in[E_{\min}(G),E_{\max}(G)]$ 。如果给定的能量比应用 G 所需的最低能量还低，则不可调度，因为能量无法满足整个应用调度的需要；如果给定的能量比应用 G 所需要的最大能量还高，则能量始终能满足整个应用调度的最大能量需求，换句话说，也就不存在所谓的能量约束。

4.4 问题分析与算法设计

由于要满足能量的约束，并且提升可靠性的同时还要满足截止时间的约束。在第 3 章中我们解决了能量约束下调度长度最小化问题，也即时间最小化，因此延续第 3 章中算法设计的思路，将本章的问题分成三个子问题：一是满足能量约束，二是满足截止时间需求，三是提高可靠性。在满足能量约束下所获得的调度长度（即响应时间，本章接下来主要以响应时间来表示）与实际的截止时间之间还存在一定的可松弛时间，通过计算松弛时间，对任务重新分配以选择能提高可靠性的处理器来实现整个应用最终的可靠性优化。

4.4.1 任务优先级排序

任务的向上排序值已经成为公认的任务优先级标准，已得到广泛的使用。在第 3 章已经对图 3.1 所示的 DAG 实例中各个任务的向上排序值进行了计算并得到相应的降序排序 $\{n_1,n_3,n_4,n_2,n_5,n_6,n_9,n_7,n_8,n_{10}\}$，而本章继续以向上排序值作为任务优先级的参考标准。

4.4.2 能量分配

通过前面的分析可知，第 3 章中给任务预分配方法相比预分配最低能量提升了性能，而本章需要同时考虑能量以及截止时间的约束。由于截止时间与出口任务最近，因此可以从出口任务向入口任务的方向进行松弛时间的计算，此外能量也基于上一章能量最终分配结果的基础上进行重新分配。对于上一章节 ESECC 算法中最终能量分配结果，本章定义 n_i 分配到的能量为 $E_{\text{ESECC}}(n_i)$，并且定义重新分配的能量为 $E_{\text{REREC}}(n_i)$，而 REREC 算法即接下来将要提出的能量约束下可靠性增强算法。和 ESECC 算法中的能量分配的顺序不同，本章根据任务向上排序值的升序顺序进行能量的重新分配，实现应用的能量约束与任务能量约束之间的转换。以图 3.1 中 DAG 为例，根据任务向上排序值计算结果得到的升序顺序为 $\{n_{10},n_8,n_7,n_9,n_6,n_5,n_2,n_4,n_3,n_1\}$。

对于包含 N 个任务的 DAG 应用来说，假设根据 $rank_u$ 升序排序后的顺序为

$\{n_{s(1)}, n_{s(2)}, ..., n_{s(|N|)}\}$，接下来将从集合中第一个任务开始进行能量的重新分配，一直到最后一个任务。当分配到第 j 个任务时，对应的集合$\{n_{s(1)}, n_{s(2)}, ..., n_{s(j-1)}\}$ 中的任务表示已经进行能量重新分配的任务，集合$\{n_{s(j+1)}, n_{s(j+2)}, ..., n_{s(|N|)}\}$ 中的任务为未进行能量重新分配的任务，但是需要注意的是，该集合中的任务已经预分配了上一阶段 ESECC 算法得出的任务实际能耗值。因此，当给第 j 个任务重新分配能量时，有：

$$E_{s(j)}(G) = \sum_{x=1}^{j-1} E_{\mathrm{REREC}}(n_{s(x)}) + E(n_{s(j)}) + \sum_{y=j+1}^{|N|} E_{\mathrm{ESECC}}(n_{s(y)}). \tag{4.17}$$

为了满足能量约束，必须满足：

$$E_{s(j)}(G) \le E_{\mathrm{given}}(G). \tag{4.18}$$

由公式(4.17)和(4.18)可以得到任务 $n_{s(j)}$ 的约束条件，即：

$$E(n_{s(j)}) \le E_{\mathrm{given}}(G) - \sum_{x=1}^{j-1} E_{\mathrm{REREC}}(n_{s(x)}) - \sum_{y=j+1}^{|N|} E_{\mathrm{ESECC}}(n_{s(y)}). \tag{4.19}$$

因此直接设置任务 $n_{s(j)}$ 的能量约束为：

$$E_{\mathrm{given}}(n_{s(j)}) = E_{\mathrm{given}}(G) - \sum_{x=1}^{j-1} E_{\mathrm{REREC}}(n_{s(x)}) - \sum_{y=j+1}^{|N|} E_{\mathrm{ESECC}}(n_{s(y)}). \tag{4.20}$$

考虑到实际情况中任务 $n_{s(j)}$ 的的能量使用不可能超过 $E_{\max}(n_{s(j)})$，因此给任务 $n_{s(j)}$ 分配的能量上限为：

$$E_{\mathrm{up}}(n_{s(j)}) = \min\{E_{\mathrm{given}}(n_{s(j)}), E_{\max}(n_{s(j)})\}. \tag{4.21}$$

至此完成了能量的重新分配，应用的能量约束被转换成任务各自的能量约束。在 ESECC 算法中预分配能量的方法已经通过数学方法以及实验证明了是满足能量约束条件的，而重新分配能量是在 ESECC 算法基础上进行的，只是换成了按任务向上排序值的升序顺序，因此从理论上分析也是能满足能量约束条件，这里不作详细证明。

4.4.3 截止时间松弛

和能量约束类似，对于截止时间需求也可以进行应用的截止时间与任务的截止时间之间的转换，同样也是根据任务的 $rank_{\mathrm{u}}$ 升序顺序进行，即从出口任务到入口任务的方向。主要思想是先计算任务的松弛时间，然后便可以根据松弛时间对任务进行处理器的重新分配以提高任务的可靠性。以第 3 章中图 3.1 所示 DAG 为例，就是先进行 n_{10} 的重新分配，在不超过能量约束以及截止时间的前提下重新选择一个能提高可靠性的处理器进行分配。在对 n_{10} 重新分配处理器时，其余任务仍然按照 ESECC 算法的分配方式不作修改，以免整个应用超过截止时间。为了详细描述该过程，这里先明确一些概念。

定义 4.1：最迟结束时间(Latest finish time, LFT)。最迟结束时间指的是任务在所有处理器上执行所允许的最晚结束时间。显然，对于出口任务而言，其最迟结束时间就是应用的截止时间，而其余任务的最迟结束时间则受到其后继结点任务的影响，即不能违背任务间的优先约束关系。任务的 LFT 计算如下：

$$\begin{cases} LFT(n_{\text{exit}},u_k)=D(G) \\ LFT(n_i,u_k)=\min\left\{\min\limits_{n_j\in succ(n_i)}\left\{AST(n_j)-c'_{i,j}\right\},AS_{\text{end}}(n_i,u_k)\right\} \end{cases}. \tag{4.22}$$

其中 $AST(n_j)$ 表示任务 n_j 的实际开始时间(Actual start time, AST)，$AS_{\text{end}}(n_i,u_k)$表示任务 n_i 在处理器 u_k 上可获得的松弛时间终点。在不同的处理器上，n_i 的最迟结束时间可能不同（对于出口任务都是应用的截止时间）。

同任务的最迟结束时间(LFT)类似，相应地也需要计算任务的早开始时间(Earliest start time, EST)。每个任务在不同处理器上的最早开始时间可能不同（对于入口任务而言都是 0），计算公式如下所示：

$$\begin{cases} EST(n_{\text{entry}},u_k)=0 \\ EST(n_i,u_k)=\max\left\{\max\limits_{n_h\in pred(n_i)}\left\{AFT(n_h)-c'_{i,h}\right\},AS_{\text{start}}(n_i,u_k)\right\} \end{cases}. \tag{4.23}$$

$AFT(n_h)$ 表示任务 n_h 的实际结束时间(Actual finish time, AFT), $AS_{\text{start}}(n_i,u_k)$ 表示任务 n_i 在处理器 u_k 上可获得的松弛时间起点。因此对于单独的任务来说，其时间约束便是最早开始时间与最迟结束时间这两个时间点，换句话说，任务的执行时间只允许在 EST 与 LFT 之间。以第 3 章图 3.1 所示 DAG 中任务 n_{10} 为例，当截止时间为 120 时，其在各处理器上的最早开始时间与最迟结束时间如下。

$$\begin{cases} EST(n_{10},u_1)=83.799 \\ EST(n_{10},u_2)=77.033 \\ EST(n_{10},u_3)=83.799 \end{cases} \qquad \begin{cases} LFT(n_{10},u_1)=120 \\ LFT(n_{10},u_2)=120 \\ LFT(n_{10},u_3)=120 \end{cases}$$

4.4.4 可靠性优化

前面对能量约束以及截止时间进行了分析，并且将应用的能量约束与截止时间需求转换成每个任务的能量约束与时间限制。接下来就是进行处理器的分配以提高任务的可靠性。用数学公式形式化表示为：

$$R(n_i)=R(n_i,u_{pr(i)},f_{pr(i),hz(i)})=\max_{\substack{u_k\in U,\,f_{k,\text{low}}\le f_{k,v}\le f_{k,\max} \\ E(n_i,u_k,f_{k,v})\le E_{\text{given}}(n_i) \\ w_{i,k,v}\le MET(n_i,u_k)}}\left\{R(n_i,u_k)\right\}. \tag{4.24}$$

$MET(n_i,u_k)$ 表示任务 n_i 在处理器 u_k 上的最大执行时间(Maximum execution time，MET)，计算方式为：

$$MET(n_i,u_k)=LFT(n_i,u_k)-EST(n_i,u_k). \tag{4.25}$$

4.4.5 算法过程

经过上述分析，可以整理得出相应的能量约束下的功能应用可靠性增强算法(Reliability Enhancement with Response time and Energy Constraints, REREC)的整个过程，主要包括能量分配、截止时间松弛以及可靠性优化三个阶段，具体过程如算法 4.1 所示。

算法的主要思想是先对 ESECC 算法的能量使用进行分析，根据任务向上排序值升序顺序给任务重新分配能量，实现应用能量约束到任务能量约束的转换，然后根据应用的截止时间来计算松弛时间，并将应用的截止时间转换成任务的时间约束，最后在任务能量以及时间的双重约束下为任务选择最大可靠性的处理器以实现可靠性的优化。

算法 4.1: The REREC Algorithm

Input: $U=\{u_1, u_2, \ldots, u_{|U|}\}$, G, $E_{\text{given}}(G)$, $D(G)$
Output: $E(G)$, $RT(G)$, $R(G)$
1: Run ESECC algorithm and get the task mapping result;
2: Sort the tasks in a list *uplist* by ascending order of $rank_u$ values;
3: **while** (tasks in *uplist*) **do**
4: $n_i = uplist.out()$;
5: Calculate $E_{\text{given}}(n_i)$ using Eq. (4.20);
6: **for all** ($u_k \in U$) **do**
7: Calculate $LFT(n_i,u_k)$ using Eq. (4.22);
8: Calculate $EST(n_i,u_k)$ using Eq. (4.23);
9: Calculate $MET(n_i,u_k)$ using Eq. (4.25);
10: **for** all($f_{k,v} \in [f_{\text{low}}, f_{\max}]$ in descending order) **do**
11: Calculate $E(n_i, u_k, f_{k,v})$;
12: **if** $((E(n_i, u_k, f_{k,v}) \le E_{\text{up}}(n_i)) \&\& (w_{i,k,v} \le MET(n_i,u_k)))$ **then**
13: Calculate $R(n_i, u_k, f_{k,v})$ using Eq. (4.5);
14: **if** $(R(n_i, u_k, f_{k,v}) > R(n_i, u_k))$
15: $R(n_i, u_k) \leftarrow R(n_i, u_k, f_{k,v})$;
16: $f_{k,hz(i)} \leftarrow f_{k,v}$;
17: **end if**
18: **end if**
19: **end for**
20: **if** $(R(n_i, u_k, f_{k,v}) > R(n_i))$ **then**
21: $pr(i) \leftarrow k$;
22: $f_{pr(i),hz(i)} \leftarrow f_{k,v}$;
23: $E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) \leftarrow E(n_i, u_k, f_{k,v})$;
24: $R(n_i) \leftarrow R(n_i, u_k, f_{k,v})$;
25: **end if**
26: **end for**
27: **end while**
28: Calculate the actual energy consumption $E(G)$ using Eq. (4.7);
29: Calculate the response time $RT(G)$ using Eq. (4.9);
30: Calculate the reliability $R(G)$ using Eq. (4.6);
31: **return** $E(G)$, $RT(G)$ and $R(G)$.

算法过程的详细解释如下：

（1）第 1 行，根据输入参数得到 ESECC 算法的调度结果以及任务分配情况，ESECC 算法调度结果中能量的分配结果是作为接下来能量重分配过程中任务的预分配值。

（2）第 2 行，计算任务的向上排序值 $rank_u$，并按 $rank_u$ 值升序排序得到任务分配顺序列表 $uplist$。

（3）第 5 行，计算任务的能量约束，将应用的能量约束转换成任务的约束。

（4）第 6-26 行，遍历所有处理器以及相应的频率，并计算任务的时间约束，在保证任务的能耗以及执行时间不超过约束条件的前提下，选择能使任务获得最大可靠性的处理器。

（5）第 28-30 行，计算最终实际消耗的能量以及应用的响应时间与可靠性。

上一章已经分析 ESECC 算法的时间复杂度为 $O(|N|^2\times|U|\times|F|)$，接下来分析 REREC 算法的时间复杂度。算法的第 1 行执行了 ESECC 算法，由上一章分析可知其时间复杂度为 $O(|N|^2\times|U|\times|F|)$，且这部分复杂度与下面的时间复杂度是独立关系；接下来算法中主要的时间开销在第 3-27 行。遍历 N 个任务的时间复杂度为 $O(|N|)$，第 6-19 行遍历所有处理器以及频率来为任务选择最优处理器和频率组合的时间复杂度为 $O(|N|\times|U|\times|F|)$，$|F|$表示频率集合的大小，由于这两部分是嵌套关系，所以最终第 3-27 行的时间复杂度为 $O(|N|^2\times|U|\times|F|)$。因此整个算法的时间复杂度仍然是 $O(|N|^2\times|U|\times|F|)$。因此可见 REREC 算法和 ESECC 算法的时间复杂度是一样的，所以 REREC 算法仍然是一个低复杂度的算法。

接下来用图 3.1 中的例子对所提算法进行说明验证。为了方便比较，参考文献[33]中的实验部分，处理器参数值如表 4.2 所示。处理器的最大频率设置为 1.0 且精度为 0.01。同时可以算出该应用能消耗的最小能量值为 19.9463，最大能量值为 157.74。为客观比较，和文献[33]一样，这里设置能量约束为 $E_{given}(G) = 3\times E_{min}(G) = 59.839$。由于能量较少，必然会导致调度长度的增大，在该能量约束下，假设应用的截止时间为 $D(G)=120$。

表 4.2 示例处理器参数值(含失效率)

u_k	$P_{k,ind}$	$C_{k,ef}$	m_k	$f_{k,min}$	$f_{k,max}$	$\lambda_{k,max}$
u_1	0.03	0.8	2.9	0.22	1.0	0.00015
u_2	0.04	0.7	2.5	0.21	1.0	0.0002
u_3	0.07	1.0	2.5	0.19	1.0	0.00025

执行算法 REREC 后的调度结果如表 4.3 所示。可以算出最终消耗的能量为 59.7094<59.839，因此满足能量约束条件，并且由于任务 n_1 的实际开始时间为 11.1594，所以应用实际响应时间为 120-11.1594=108.8406，因此在截止时间内。此外，应用最终的可靠性为 0.9575。图 4.1 表示相应的调度结果甘特图。

表 4.3 REREC 算法调度 DAG 示例的调度结果

n_i	$E_{given}(n_i)$	$u(n_i)$	$f(n_i)$	$AST(n_i)$	$AFT(n_i)$	$E(n_i)$	$R(n_i)$
n_{10}	6.5106	u_2	1.00	113	120	5.88	0.998601
n_8	5.4806	u_1	1.00	97	102	4.15	0.99925
n_7	6.5133	u_1	1.00	89	96	5.81	0.998951
n_9	7.7145	u_2	0.90	99.6667	113	7.7054	0.996351
n_6	6.2807	u_1	0.73	71.1918	89	6.2536	0.993831
n_5	6.1387	u_1	0.75	55.1918	71.1918	6.0376	0.994789
n_2	6.4973	u_1	0.74	37.6242	55.1918	6.3962	0.9941
n_4	6.0211	u_2	1.00	62	70	5.92	0.998401
n_3	6.0128	u_2	0.69	43.1594	62	5.9694	0.990032
n_1	6.4169	u_1	0.70	11.1594	31.1594	6.2873	0.992399

$$E(G)=\sum_{i=1}^{10}E(n_i)=59.7094,\ RT(G)=AFT(n_{10})-AST(n_1)=108.8406,\ R(G)=\prod_{i=1}^{10}R(n_i)=0.9575$$

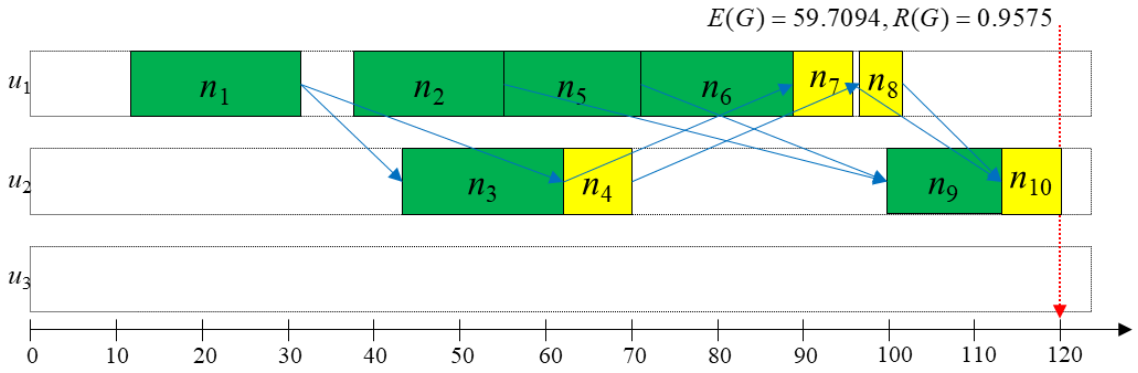


图 4.1 REREC 算法调度 DAG 示例的调度甘特图

针对简单示例，这里同样将本章算法与 MSLECC 算法、ESECC 算法以及 MREC[33]算法进行对比。MSLECC 算法针对的是能量约束下的调度长度最小化，并没有考虑可靠性；MREC 算法是研究的能量约束下的可靠性最大化，但是没有考虑截止时间因素。表 4.4 显示了相应的对比数据。从表中可以看出四种算法都能满足能量约束条件；此外无论是可靠性还是调度长度 MSLECC 算法都表现较差，MELECC 以及 MREC 算法无法满足截止时间需求。本章提出的 REREC 算法的实际响应时间为 108.8406，且其可靠性为几种算法中最高的。

表 4.4 几种算法调度结果对比

算法	$E(G)$	$RT(G)$	$R(G)$
MSLECC	59.8379	169.5083	0.7338
ESECC	58.5084	109.0068	0.9153
MREC	59.8384	147.9748	0.82
REREC	59.7094	108.8406	0.9575

4.5 实验及分析

为了更好地说明算法的优越性，本小节做进一步的对比实验。参与对比的算法有 HEFT、MRRR[40]、ESECC、MREC[33]、REREC。HEFT 算法作为最基本的算法，虽然没有考虑各种约束条件，但作为一个高性能算法，其调度结果能提供很好的参考价值； MRRR 算法是实现响应时间约束下最大化应用的可靠性，且为当前该问题研究中表现最好的算法之一，虽然没有考虑能量约束，但同样可以将其调度结果拿来以供参考。由于第 3 章已经验证了 ESECC 算法各方面性能都比 MSLECC 算法要优，因此这里只拿 ESECC 算法来对比，舍弃 MSLECC 算法；MREC 算法实现了有能量约束的可靠性最大化，和本章研究的问题比较接近，但没有考虑应用截止时间需求。我们主要对比每个算法实际消耗的能量和最终的响应时间以及整个应用的可靠性，并且采用真实的汽车功能应用以及随机生成的功能应用来验证相关算法。

4.5.1 实验设置

本文研究的算法主要针对功能应用的设计阶段，考虑到实际硬件平台实验的困难与复杂，本文采用仿真平台进行实验。实验的硬件环境为一台 4 核 Intel i7-6770 处理器，8G 内存的个人电脑，操作系统为 Windows 10。仿真实验用的软件工具为 My Eclipse 2014，使用的编程语言为 Java。实验中处理器的失效率取值参考自文献[76]，详细的处理器和应用参数范围如下：$0.000001/\mu s \le \lambda_k \le 0.000009/\mu s$，$100\mu s \le w_{i,j} \le 400\mu s$, $100\mu s \le c_{i,j} \le 400\mu s$, $0.03 \le p_{k,\mathrm{ind}} \le 0.07$, $0.8 \le C_{k,\mathrm{ef}} \le 1.2$, $2.5 \le m_k \le 3.0$, $f_{k,\max}$=1GHz。所有的频率是离散的，精度为 0.01 GHz，生成的异构处理器数量为 16，并且具体数值是用随机程序产生。由于实验的参数范围固定，大量的实验结果都显示了相对稳定的规律，因此下面具体实验的结果数据只随机抽取一次来展示。

4.5.2 实验过程及结果分析

1）真实的汽车功能应用

本小节采用真实的汽车功能应用来进行算法的对比实验，该功能应用参考自文献[77]，应用中的任务拓扑图如图 4.2 所示。该功能应用包含了 6 个子功能模块：$n_1 \sim n_7$为发动机控制相关的任务，$n_8 \sim n_{11}$为自动变速箱相关的任务，$n_{12} \sim n_{17}$为 ABS 防抱死系统相关任务，$n_{18} \sim n_{19}$为转角传感器相关的任务，$n_{20} \sim n_{24}$为悬浮控制器相关的任务，$n_{25} \sim n_{31}$为车身稳定相关的任务。由图 4.2 可知该功能应用可以抽象成一个 DAG。

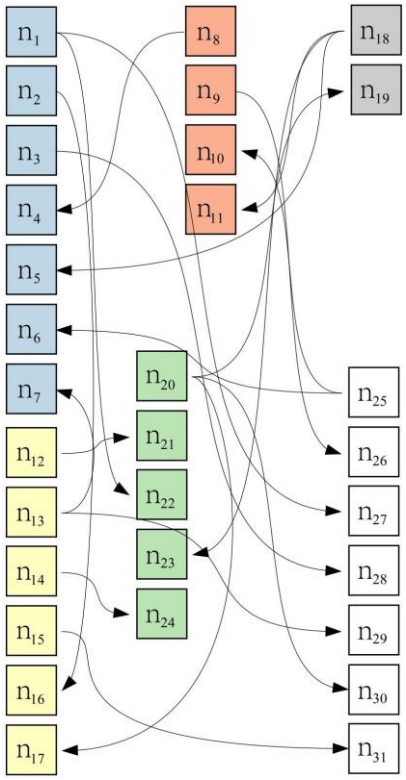


图 4.2 真实汽车电子功能应用

表 4.5 不同能量约束下真实汽车功能应用调度结果

算 法		0.5	0.6	0.7	0.8	0.9
HEFT	$E(G)$	5347.12	5347.12	5347.12	5347.12	5347.12
	$RT(G)$	578	578	578	578	578
	$R(G)$	0.9662	0.9662	0.9662	0.9662	0.9662
MRRR	$E(G)$	5286.75	5286.75	5286.75	5286.75	5286.75
	$RT(G)$	1455	1455	1455	1455	1455
	$R(G)$	0.9843	0.9843	0.9843	0.9843	0.9843
ESECC	$E(G)$	2671.78	3206.84	3739.64	4277.44	4787.05
	$RT(G)$	873.77	781.33	695.00	664.12	639.00
	$R(G)$	0.9158	0.9120	0.9452	0.9400	0.9534
MREC	$E(G)$	2673.55	3208.26	3742.97	4277.67	4812.37
	$RT(G)$	3052.38	2286.43	3014.40	2924.29	3237.64
	$R(G)$	0.5702	0.6111	0.6940	0.7603	0.8450
REREC	$E(G)$	2672.46	3207.86	3741.44	4269.93	4730.56
	$RT(G)$	1499.63	1343.99	1393.52	1456.90	1426.00
	$R(G)$	0.9513	0.9680	0.9770	0.9834	0.9839

实验结果见表 4.5（能量单位：Wμs，响应时间单位：μs），其中 HEFT 算法得到的能耗值为 $E_{\mathrm{HEFT}}(G)$=5347.12Wμs。由于 HEFT 算法是一个高性能的算

法，很多基于 DVFS 技术的 DAG 低能耗调度都参考了 HEFT 算法的核心思想，因此我们以 HEFT 算法的能耗为标准，将能量约束设置为 $E_{HEFT}(G)\times0.5$~$E_{HEFT}(G)\times0.9$，取系数 0.1 为步长，分别代表不同的能量约束，最后观察所对比的几种算法在不同情景下的调度性能。参考 HEFT 的调度长度，固定时间约束为 1500μs。我们用折线图以及柱状图做更直观的对比展示（图 4.3~图 4.5）。

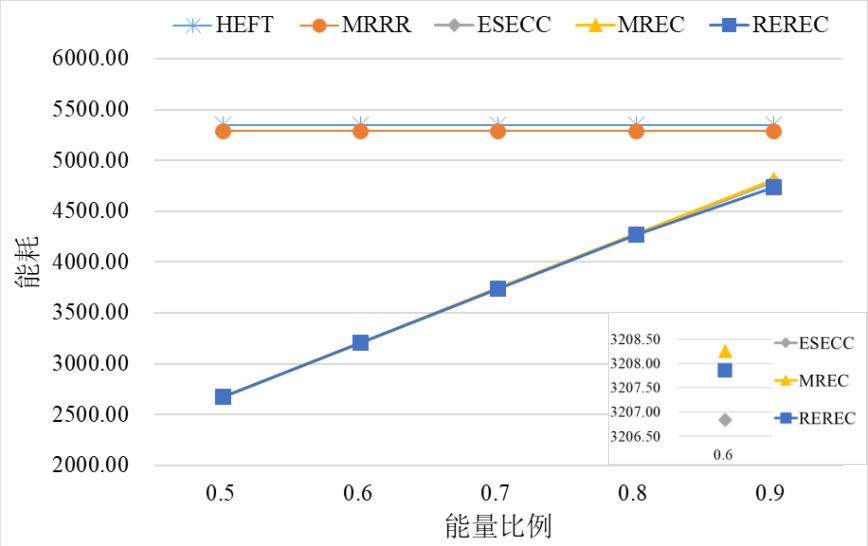


图 4.3 不同能量约束下的能耗对比

图 4.3 展示的是不同能量约束下，各个算法实际消耗的能量对比。可以看到 ESECC、MREC 以及 REREC 算法的曲线几乎是重叠的，很难区分（系数为 0.6 时，三者能耗值相差不超过 2），同时结合表 4.5 中的数据能算出它们的能耗在约束值范围内，这也应证了这三个算法是适用于有能量约束下的调度，因此消耗的能量都近似等于能量约束值。而 MRRR 算法产生的能耗值在各个情况下都是固定的，和 HEFT 算法很接近，且都超过相应的约束条件，这是因为 MRRR 算法只考虑了截止时间约束这唯一条件，并没有考虑能量的限制条件。

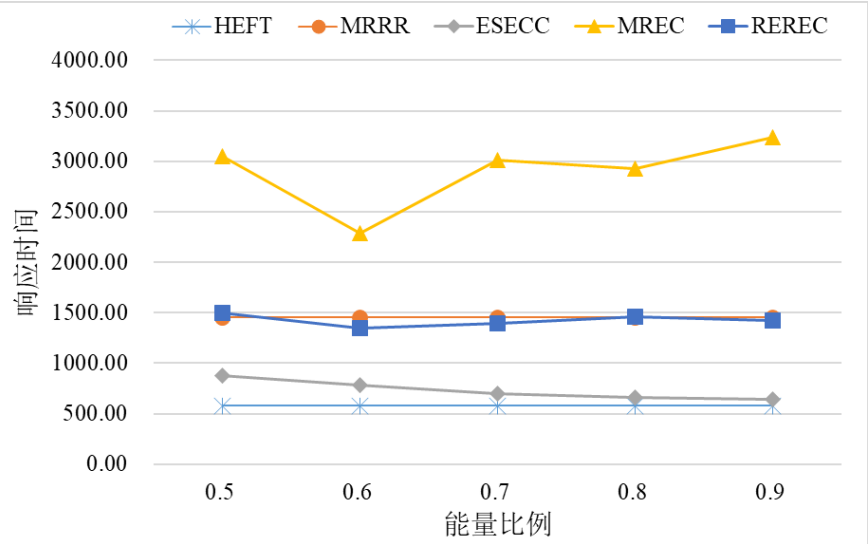


图 4.4 不同能量约束下的响应时间对比

图 4.4 展示的是不同能量约束下，各个算法的响应时间对比。考虑到响应时间约束为 1500μs，可以从图中看到 HEFT、MRRR、ESECC 以及 REREC 都能满足时间约束，而 MREC 算法的响应时间在几种情形下都无法满足需求。HEFT 算法的响应时间最短，这也验证了其是一个高性能的算法，但它不能满足能量约束。ESECC 算法即使在考虑能量约束的情况下，其响应时间也跟 HEFT 非常接近。MRRR 和 REREC 都考虑了时间约束，因此其响应时间都接近给定的约束值，这两种算法主要通过松弛时间来获取整个功能应用最大的可靠值。MREC 算法是实现能量约束下可靠性最大化，因此响应时间得不到保证，并且由于采用了贪心思想优化可靠性，跟 ESECC 算法随着能量增多响应时间降低的趋势不同，其响应时间不是随着能量的增多而降低，并且幅度变化也大，是一个不稳定的算法。

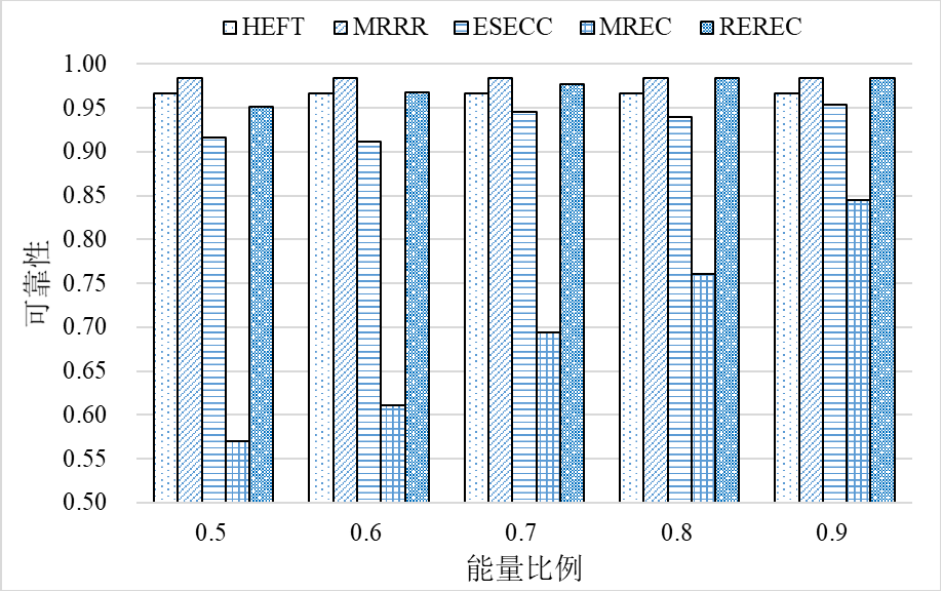


图 4.5 不同能量约束下的可靠性对比

图 4.5 展示的是不同能量约束下，各个算法的可靠性对比。由于数值比较接近，因此用柱状图来展示。HEFT 算法和 MRRR 算法都得到固定的可靠性值，其值分别为 0.9662 和 0.9843。MRRR 算法的可靠性在所有算法中是最高的，但是如图 4.3 所示，其能量约束条件得不到保证，和 HEFT 能耗近似相等。MREC 算法虽然考虑的是能量约束下提高可靠性，但实际上其可靠性值并不高，在能量为 $E_{\mathrm{HEFT}}(G)\times 0.5$ 时才只有 0.5702，而即便在 $E_{\mathrm{HEFT}}(G)\times 0.9$ 的约束下，可靠性也才只有 0.845。ESECC 和 REREC 算法在所有情况下可靠性都比较高，但相比之下，REREC 算法比 ESECC 有 3%~4%左右的提升，因为 REREC 在保证截止时间需求的情况下计算了松弛时间，进而为任务选择更合适的处理器来提高可靠性。此外 REREC 算法最大可靠性为 0.9839，和 MRRR,0.9843 非常接近，而能耗却只有后者的 89%，并且后者的能量使用超过了所给的限制。

2）随机生成的功能应用

考虑到未来 ACPS 将会越来越复杂，电子功能也只会越来越多，单个应用也可能达到近百个任务。为了进一步验证算法的性能，本小节采用人工随机生成的功能应用来进行实验。处理器参数和上一节真实的汽车电子功能应用一样，而随机生成的功能应用则采用任务图生成器来生成[78]，该程序可以根据不同参数生成不同特性的 DAG。其中通信与计算平均开销比值 CCR 为：CCR={0.1,0.5,1.0,2.0,5.0}，CCR 反映了应用中通信的重要程度，CCR 越小（小于 1）表示的是计算密集型的应用，越大（大于 1）表示通信密集型的应用；异构计算因子为：η={0.2,0.4,0.5,0.6,0.8}，其值越大表示异构性越大，主要反映在任务在不同处理器上处理任务的时间开销差异越大；形状因子 α 是个随机数，范围为$[0.35,\sqrt{|N|/3}]$，表示的是 DAG 的形状，反映了并行程度。

在该实验中，我们设置 CCR 为 1，异构计算因子为 0.5，形状因子为 1，研究任务数从 50 到 100 变化的调度情况，每次实验任务数增加 10 个。这里固定能量约束为一个适中的值，为 $E_{\mathrm{HEFT}}(G)\times 0.75$，同时考虑到任务数是一个变化的过程，相应的时间约束也是不同的，因此设置为 2500μs ~5000μs，并且随任务数增长的步长为 500μs。实验的详细结果见表 4.6（能量单位：Wμs，响应时间单位：μs），同时这里将数据用折线图以及柱状图做更直观的对比展示（图 4.6~图 4.8）。

表 4.6 不同任务数的随机应用调度结果

算 法		50	60	70	80	90	100
HEFT	$E(G)$	3110.42	3569.44	4343.49	5115.17	5538.80	6548.97
	$RT(G)$	1123	1392	1473	1608	1613	1652
	$R(G)$	0.9897	0.9855	0.9834	0.9819	0.9782	0.9760
MRRR	$E(G)$	4010.72	5005.03	5677.98	6793.51	7248.68	8967.33
	$RT(G)$	2439	2989	2884	3658	4023	4952
	$R(G)$	0.9953	0.9942	0.9933	0.9924	0.9916	0.9904
ESECC	$E(G)$	2332.39	2676.97	3256.92	3836.18	4153.88	4911.52
	$RT(G)$	1281.23	1590.88	1646.18	1831.98	1828.43	1829.97
	$R(G)$	0.9795	0.9725	0.9709	0.9635	0.9598	0.9583
MREC	$E(G)$	2332.81	2677.08	3257.62	3836.37	4154.10	4911.73
	$RT(G)$	4616.54	5179.54	5741.92	7330.41	6507.27	6763.17
	$R(G)$	0.8018	0.7206	0.7400	0.6944	0.6550	0.6036
REREC	$E(G)$	2332.39	2676.90	3256.92	3835.73	4153.94	4910.99
	$RT(G)$	1921.50	2508.46	2721.59	2929.29	3239.88	3417.27
	$R(G)$	0.9890	0.9849	0.9848	0.9812	0.9798	0.9766

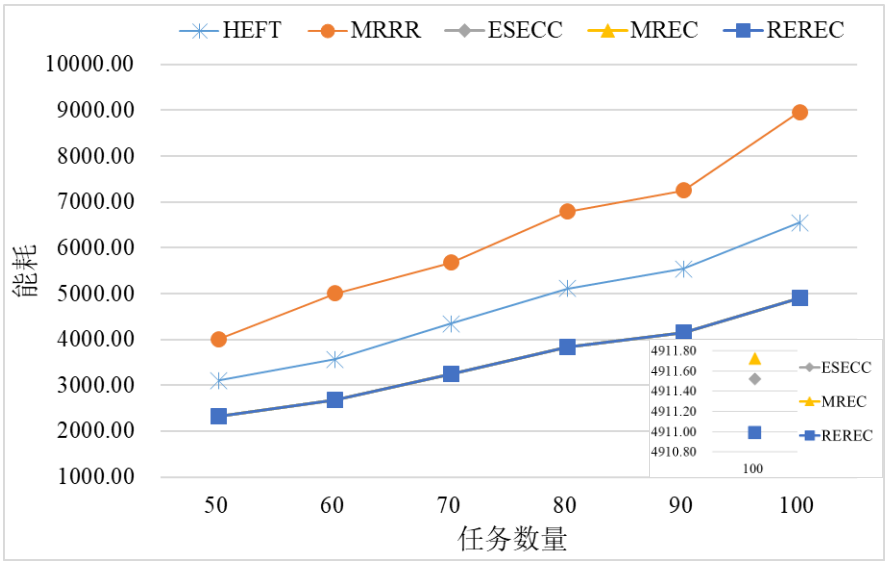


图 4.6 不同任务数下能耗对比

图 4.6 展示了不同任务数的随机功能应用，各调度算法的能耗值对比。同真实汽车功能应用实验结果一样，ESECC、MREC 以及 REREC 三个算法的能耗数值非常接近，如图 4.6 右下角小图所示，任务数为 100 时，三个算法的差值只有 1 左右，因此在主图中几乎重叠在一起，这也再次验证了这三个算法是为能量约束下的调度情况而设计的，其充分利用系统所给的有限的能量来调度任务，因此最终消耗的能量都是很接近约束值。而 MRRR 算法的能耗值非常大，甚至比 HEFT 算法的还要高，这也说明了该算法只考虑时间约束，在时间约束下以较高的能耗代价来提高可靠性。

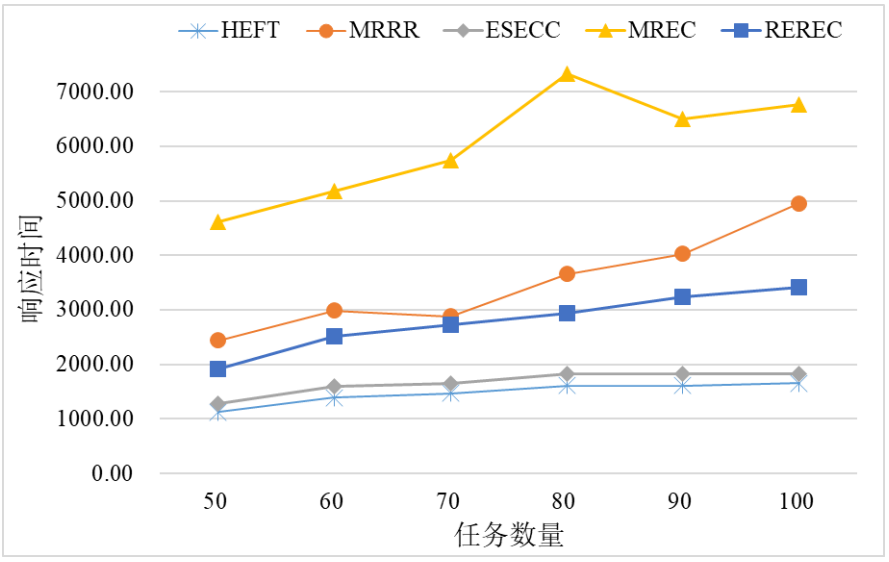


图 4.7 不同任务数下响应时间对比

图 4.7 展示了不同任务数的随机功能应用，各调度算法的响应时间对比。可以看到 HEFT 算法的响应时间总是最小，ESECC 算法次之，但非常接近。MRRR 算法和 REREC 都考虑了时间约束条件，实验结果也再次验证了这一点。

MREC 算法只考虑能量约束，没有考虑时间约束，因此其响应时间非常大；该算法牺牲了较多的时间来提升可靠性，并且在任务数为 80 时，响应时间有个较大的突变，多次实验都发现了类似的情况，这也说明该算法不太稳定，在响应时间上很难得到保证。

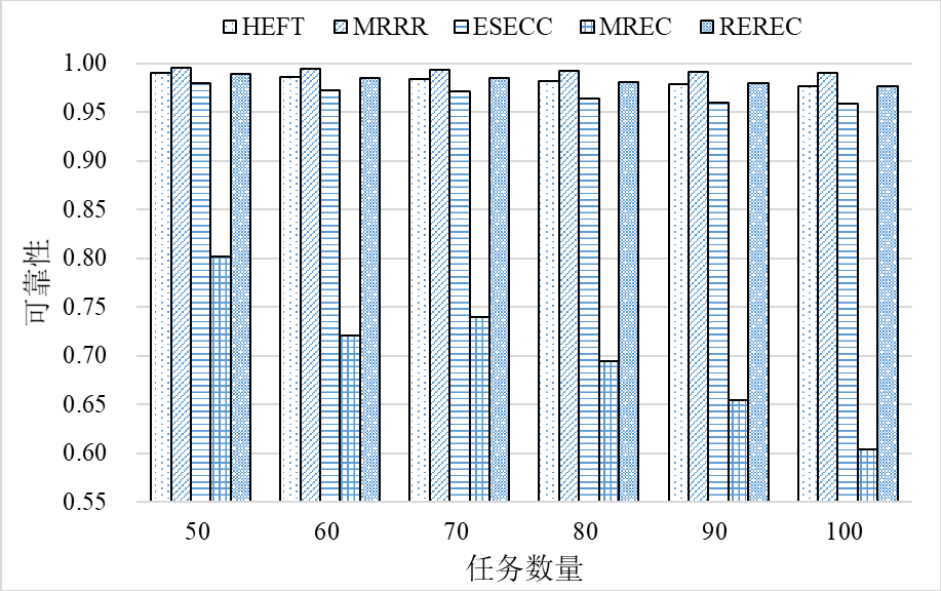


图 4.8 不同任务数下可靠性对比

图 4.8 展示了不同任务数的随机功能应用，各调度算法生成的可靠性对比。可以发现，MRRR 的可靠性在所有情况下是最高的，并且由表 4.6 可知都达到了 0.99 以上，即对应 ISO 26262 中的 E2 等级。HEFT 算法的可靠性也很高，在 0.98 左右，且随着任务数增加呈下降趋势。ESECC 算法的可靠性也是随着任务数增加而下降，在 0.95~0.98 之间，而 REREC 算法的可靠性一直在保持在 0.98 左右，相对于 ESECC 算法在所有情况下都有所提高。和 HEFT 算法相比，REREC 的能耗只有 HEFT 的 75%，但其可靠性和 HEFT 的几乎一致，甚至在任务数为 70~100 时，REREC 算法比 HEFT 的可靠性还略高一点。但是 MREC 算法的可靠性值比较低，并且当任务数为 100 时只有 0.6036，其主要原因是采用了贪心策略，为高优先级任务分配了较多能量以选择可靠性高的处理器，而低优先级任务往往只能在最低有效频率执行，这不仅严重增加了响应时间（如图 4.7 所示），也影响了可靠性。

实验部分小结：

综合上述实验结果可以总结得出：MRRR 算法能得到非常高的可靠性，但是其能耗也非常大，不满足本章所讨论的能量约束条件；MREC 算法虽然考虑的是有能量约束下的可靠性最大化，但其可靠性值在对比的算法中是最低的，表现较差，也是由于不合理的能量分配算法导致的；第 3 章提出的 ESECC 算法虽然没考虑增强可靠性，但能量的合理分配使得其不会存在部分任务严重降低可靠性的情况，因此整体表现中规中矩；而 REREC 算法考虑了可靠性优化，在较低

的能耗下仍能保持较高的可靠性，大部分情况下比 HEFT 还高，仅次于 MRRR 算法。和同样考虑能耗约束下可靠性优化的算法 MREC 相比，不仅响应时间比后者小，可靠性也比后者高很多。

4.6 小结

本章继续针对 ACPS 中的能耗问题，在上一章提出的能量约束下最小化调度长度的算法基础上，进一步考虑 ACPS 中的可靠性问题，并结合道路车辆功能安全标准 ISO 26262 中的相关概念与内容对功能应用的可靠性进行描述。针对低能耗设计中 DVFS 技术的应用对系统可靠性产生的影响，参阅现有文献对能耗与可靠性的优化建立关系，研究了在能量有限的情况下去提高功能应用的可靠性，并且还考虑了应用的截止时间约束，进而提出了 REREC 算法。通过真实的汽车电子功能应用以及利用模拟器生成的随机应用进行实验验证，并对比了现有研究中与本章最相关的几种算法。实验结果显示本章提出的算法在满足能量约束以及不超过截止时间的情况下能得到非常高的可靠性，相比现有算法更优，因此 REREC 算法在能量感知的 ACPS 中保障功能安全可靠性设计方面具有一定的可行性。

结论

1．工作总结

随着计算机技术、通信技术以及智能控制技术的快速发展，CPS 概念越来越被学术界以及产业界所重视，从 CPS 的角度进行生产制造、交通运输以及城市建设已经成为一种发展趋势。伴随着以 CPS 为核心的新一代信息技术革命，汽车产业信息化程度也不断提升，传统的机械型汽车逐渐发展成高度协同、集成化和智能化的智能汽车，从简单的机械控制系统演变成融合计算过程与物理过程的 ACPS。汽车已不再是一个简单的代步工具，有着 ACPS 的汽车更像是一个集代步、娱乐、休息功能于一体的平台。汽车工业作为一个国家层面重要的工业领域，发展 ACPS 是提升国家汽车产业创新力和竞争力的重要手段，也是一个新的研究热点。

第四次工业革命给汽车工业带来了绿色环保的发展动力，新能源汽车已经成为未来汽车工业发展的方向，而其中又将以纯电动汽车为主，而电池技术的发展远远跟不上计算技术的发展，使得能量资源成为 ACPS 中重要的约束。此外，为了满足人们对汽车安全性、舒适性以及娱乐性的需要，ACPS 中的功能应用规模不断增长，并从过去的单一功能发展成分布式的功能。自动驾驶技术相关的机器视觉处理与深度学习需要大量的数据采集与处理，给 ACPS 的计算能力带来了压力，同时对能量的使用也进一步增加。道路车辆功能安全—ISO 26262 标准对功能安全问题进行了详细的阐述，由此引出功能安全可靠性问题。因此 ACPS 的设计必须重视能量资源的有限供应，对能量进行合理分配与使用，提升系统的实时、安全与可靠。

本文针对 ACPS 中的能量资源有限的情景，分析了 ACPS 系统结构和任务调度模型，并从调度的角度研究分布式功能应用中任务的能量资源分配，以提高实时性、可靠性为需求目标，提出一系列的调度方案，并通过实验进行验证。主要的工作及成果如下：

（1）将 ACPS 抽象成异构分布嵌入式系统，从任务调度出发，对有能耗优化相关的调度研究进行详细的总结和归纳，重点分析以 DAG 抽象建模的并行应用调度问题，包括针对 DAG 应用的能耗与调度长度优化以及能耗与可靠性优化，传统的低能耗优化属于高能效调度，而考虑有限能量并进行合理分配利用的可称为能量感知调度。针对 ACPS 中能量有限的情况，总结了现有能量约束下调度研究的不足之处，对需要进一步研究以及改进的地方进行总结。

（2）研究了 ACPS 中功能应用在有能量约束下的调度长度优化问题。现有

研究不合理的能量分配策略导致调度结果很悲观，因此本文提出了改进的启发式的 ESECC 调度算法，能量的分配将更加均匀，任务在其分配的能量下选择最优的处理器以降低调度长度。通过理论分析验证了所提出的 ESECC 算法的可行性，并就实际的并行应用 FFT 应用以及 GE 应用和现有的算法进行对比实验，大量的实验结果都显示了 ESECC 算法的高效性，在满足能量约束条件下能显著地降低应用的调度长度，和现有算法的结果相比具有明显的优势，且没有增加时间复杂度。

（3）研究了 ACPS 中功能应用在有能量约束下的可靠性优化问题。现有能量约束下的可靠性优化研究未考虑应用实际的截止时间，因此本文提出相应的启发式的 REREC 调度算法。算法包括三个步骤：能量分配、时间松弛以及处理器选择。在 ESECC 算法的基础上对能量进行重新分配，并考虑了应用的截止时间需求，对任务的执行时间进行松弛，任务在能量约束以及时间约束范围内选择最优的处理器提高系统可靠性。通过真实的汽车电子功能应用以及仿真的功能应用实验，并和现有相关的一些算法进行对比，结果显示在满足能量约束下 REREC 算法能有效提高可靠性，比现有类似算法更优，且未超过应用截止时间。

综上，本文围绕 ACPS 中能量资源约束条件所提出的任务调度算法实现了任务能根据有限能量进行合理分配利用，任务能量的分配结果取决于系统所给定的总能量。ACPS 是一个对能量资源敏感的系统，本文提出的算法能为能量感知的 ACPS 设计提供一定的参考价值。

2．研究展望

ACPS 是一个包含多个功能应用的混合关键级系统，除了本文研究的能量资源约束下的调度问题，ACPS 中还有许多问题需要进一步研究，后续可针对下面几个方面进行：

（1）ACPS 中考虑任务复制的能耗优化研究。为了提高系统功能应用的可靠性，通常的途径是采用容错技术，而任务复制是实现容错调度的主要方式之一。后续可以考虑在满足能量约束下采用任务复制技术来进一步增强应用的可靠性，同时将本文的研究与满足实时性以及可靠性目标需求下的低能耗研究相结合，从互补的角度进行优化设计。

（2）ACPS 中多功能调度分析。本文只针对了单个功能应用的调度进行研究，实际上还需要分析多个功能同时调度的情况。ACPS 中不同子系统包含了不同的功能，这些功能在实时性与可靠性需求上往往也不一样，因此会存在不同的时间或可靠性关键级别。就能量资源而言，与单个功能应用调度不同，多个功能应用同时调度需要考虑功能之间的能量分配以及每个功能的截止时间与可靠性需求，因此相关的研究将更加复杂。

（3）ACPS 中的 WCRT 分析。随着 ACPS 中应用数量越来越多，通信需求也越来越大，未来 ACPS 中将引入更高速的通信网络如以太网以满足需要。ACPS 的网络体系将从基于中央网关的集成体系结构逐渐发展成以骨干网为核心，多种网络子系统并存的复杂体系结构，因此需要新的消息响应时间分析理论与方法。本文对任务通信消息的 WCRT 作了假设处理而并没有深入分析，因此未来可以针对 ACPS 中更复杂的通信网络做端到端的 WCRT 分析，使 ACPS 的研究更具完整性。

（4）ACPS 中的信息安全研究。关于安全方面，本文主要研究的是系统瞬时故障造成的功能可靠性降低。ACPS 是一个能和外界网络互连的系统，车联网的发展使汽车在行驶过程中将时刻与云服务中心、路边单元以及周围其它车辆进行信息交互，因此也就面临和传统互联网类似的信息安全问题，外部的信息攻击（如木马病毒、信息篡改等）会影响功能的正常运行。例如攻击者可以利用系统软件的漏洞获取汽车安全关键应用的权限，进而可以实现车内空调、车窗甚至是方向盘与发动机的随意控制，这给行车安全带来极大的隐患。因此有必要采用更安全的信息加密方法，从信息安全的角度研究 ACPS 中功能应用的安全性。

参考文献

[1] Osswald, Sebastian, Matz, et al. Prototyping Automotive Cyber-Physical Systems. In: Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. Seattle, WA, USA: ACM, 2014, 1-6

[2] Navet N, Song Y, Simonot-Lion F, et al. Trends in Automotive Communication Systems. Proceedings of the IEEE, 2005, 93(6): 1204-1223

[3] Zeng W, Khalid M A S, Chowdhury S. In-Vehicle Networks Outlook: Achievements and Challenges. IEEE Communications Surveys & Tutorials, 2017, 18(3): 1552-1571

[4] Charette R N. This Car Runs on Code. IEEE Spectrum, 2009, 46(3):3

[5] 谢国琪. 面向汽车的异构网络化嵌入式系统多 DAG 调度研究：[湖南大学博士学位论文]. 长沙： 湖南大学，2014，1-27

[6] 李仁发, 谢勇, 李蕊,等. 信息-物理融合系统若干关键问题综述. 计算机研究与发展, 2012, 49(6): 1149-1161

[7] Chakraborty S, Faruque M A A, Chang W, et al. Automotive Cyber–Physical Systems: A Tutorial Introduction. IEEE Design & Test, 2016, 33(4): 92-108

[8] 王喜文. 工业 4.0:智能工业. 物联网技术, 2013(12): 3-4

[9] Rajkumar R, Lee I, Sha L, et al. Cyber-physical systems: the next computing revolution. In: Proc of the 47th design automation conference. Anaheim, CA, USA: IEEE, 2010, 731-736

[10] Lee E A. Cyber Physical Systems: Design Challenges. In: Proc Of 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC). Orlando, FL, USA: IEEE, 2008,363-369

[11] Lukasiewycz M, Steinhorst S, Sagstetter F, et al. Cyber-Physical Systems Design for Electric Vehicles. In: Proc of 2012 15th Euromicro Conference on Digital System Design. Izmir, Turkey: IEEE, 2012, 477-484

[12] Chang W, Samarjit C. Resource-aware Automotive Control Systems Design: A Cyber-Physical Systems Approach. Foundations & Trends® in Electronic Design Automation, 2016, 10(4):249-369

[13] Zhang Z, Eyisi E, Koutsoukos X, et al. A co-simulation framework for design of time-triggered automotive cyber physical systems. Simulation Modelling Practice & Theory, 2014, 43(4):16-33

[14] Tehrani K, Maurice O. A cyber physical energy system design (CPESD) for electric vehicle applications. In: Proc of 12th System of Systems Engineering Conference. Waikoloa, HI, USA: IEEE, 2017:1-6.

[15] 久陵. 从特斯拉电动车看中国新能源汽车. 汽车与配件, 2013(23):11-13

[16] 王紫，闫枫. 中国新能源汽车销量连续三年全球第一 市场份额最高. http://auto.people.com.cn/n1/2018/0418/c1005-29933208.html, 2018-04-18

[17] 周晓莺. 2018 年中国汽车电子行业白皮书,盖世汽车研究院. http://auto.gasgoo.com/News/2018/03/121059035937003 6304C601.shtml, 2018-03-12

[18] Dong Z, Zhuang W, Rojas-Cessa R. Energy-aware scheduling schemes for cloud data centers on Google trace data. In: Proc of 2014 IEEE Online Conference on Green Communications (OnlineGreencomm). Tucson, AZ, USA: IEEE, 2015,1-6

[19] BOSCH. Automation is on its way-with safety, and convenience. https://www.bosch-mobility-solutions.com/en/highlights/automated-mobility/automated-driving/, 2018

[20] Kerner B S. Failure of classical traffic flow theories: Stochastic highway capacity and automatic driving. Physica A Statistical Mechanics & Its Applications, 2016, 450:700-747

[21] Road Vehicles-Functional Safety, ISO Std. 26262, 2011

[22] 王小乐. 信息物理融合系统资源调度关键技术研究：[国防科学技术大学博士学位论文]. 长沙：国防科学技术大学, 2014,5-16

[23] Lee Y C, Zomaya A Y. Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(8):1374-1381

[24] Zhu D, Melhem R, Mossé D. The effects of energy management on reliability in real-time embedded systems. In: Proc of IEEE/ACM International Conference on Computer Aided Design. San Jose, CA, USA: IEEE, 2004, 35-40

[25] Zhu D, Aydin H. Reliability-Aware Energy Management for Periodic Real-Time Tasks. In: Proc of IEEE 13th Real Time and Embedded

Technology and Applications Symposium. Bellevue, WA, USA: IEEE, 2007,225-235

[26] Zhao B, Aydin H, Zhu D. Energy Management under General Task-Level Reliability Constraints. In: Proc of IEEE 18th Real Time & Embedded Technology & Applications Symposium. Beijing, China: IEEE, 2012,285-294

[27] Zhao B, Aydin H, Zhu D. Reliability-aware Dynamic Voltage Scaling for energy-constrained real-time embedded systems. In: Proc of IEEE International Conference on Computer Design. Lake Tahoe, CA, USA: IEEE, 2008, 633-639

[28] Zhao B, Aydin H, Zhu D. On Maximizing Reliability of Real-Time Embedded Applications Under Hard Energy Constraint. IEEE Transactions on Industrial Informatics, 2010, 6(3):316-328

[29] 吴忠泽. 智能汽车发展的现状与挑战. 时代汽车, 2015(7):42-45

[30] 杨帆. 基于数据驱动的 ACPS 建模及验证方法研究：[湖南大学博士学位论文]. 长沙：湖南大学, 2016，87-89

[31] Huang J, Li R, An J, et al. Energy-Efficient Resource Utilization for Heterogeneous Embedded Computing Systems. IEEE Transactions on Computers, 2017, PP(99):1-1

[32] Xiao X, Xie G, Li R, et al. Minimizing Schedule Length of Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems. In: Proc of Trustcom/BigDataSE/ISPA, 2016 IEEE. Tianjin, China: IEEE, 2017:1471-1476

[33] Xiao X, Xie G, Xu C, et al. Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems. Journal of Computational Science, 2017,1-10

[34] 吴勇毅. 智能交通成“十二五”规划重点 车联网概念迎来新机遇. 通信世界, 2011(4):27-27

[35] 谢勇. 新一代汽车电子系统的网络体系结构若干关键技术研究：[湖南大学博士学位论文]. 长沙：湖南大学，2013，1-10

[36] Xie G, Zeng G, Li Z, et al. Adaptive Dynamic Scheduling on Multifunctional Mixed-Criticality Automotive Cyber-Physical Systems. IEEE Transactions on Vehicular Technology, 2017, 66(8):6676-6692

[37] Bernat G, Colin A, Petters S M. WCET Analysis of Probabilistic Hard Real-Time Systems. In: Proc of 23rd Real-Time Systems Symposium. Austin, Texas, USA: IEEE,2002,279

[38] Xie Y. Worst Case Response Time Analysis for Messages in Controller Area Network with Gateway. Ieice Transactions on Information & Systems, 2013, E96.D(7):1467-1477

[39] Xie G, Zeng G, Kurachi R, et al. WCRT Analysis of CAN Messages in Gateway-Integrated In-Vehicle Networks. IEEE Transactions on Vehicular Technology, 2017, 66(11):9623-9637

[40] Xie G, Zeng G, Liu Y, et al. Fast Functional Safety Verification for Distributed Automotive Applications during Early Design Phase. IEEE Transactions on Industrial Electronics, 2017, 65(5):4378-4391

[41] Xie G, Chen Y, Li R, et al. Hardware Cost Design Optimization for Functional Safety-Critical Parallel Applications on Heterogeneous Distributed Embedded Systems. IEEE Transactions on Industrial Informatics, 2017, PP(99):1-1

[42] Xie G, Zeng G, Xiao X, et al. Energy-efficient Scheduling Algorithms for Real-time Parallel Applications on Heterogeneous Distributed Embedded Systems. IEEE Transactions on Parallel & Distributed Systems, 2017, PP(99):1-1

[43] Xie G, Zeng G, Li Z, et al. Adaptive Dynamic Scheduling on Multifunctional Mixed-Criticality Automotive Cyber-Physical Systems. IEEE Transactions on Vehicular Technology, 2017, 66(8):6676-6692

[44] 唐志芳, 时海涛, 鲁华祥,等. 系统级动态电源管理算法的研究. 计算机工程与应用, 2005, 41(6):190-193

[45] 程珍珍. 异构环境基于 DVFS 的工作流任务节能调度算法研究：[湖南大学硕士学位论文]. 长沙：湖南大学, 2014.1-24

[46] Weiser M, Welch B, Demers A, et al. Scheduling for Reduced CPU Energy. In: Proc of Symposium on Operating Systems Design and Implementation. New York, USA: ACM,1994,2

[47] Jarus M, Varrette S, Oleksiak A, et al. Performance Evaluation and Energy Efficiency of High-Density HPC Platforms Based on Intel, AMD and ARM Processors. In: Proc of Revised Selected Papers of the Cost Ic0804 European Conference on Energy Efficiency in Large Scale Distributed Systems. New York: Springer, 2013,182-200

[48] Zhao B, Aydin H. Minimizing expected energy consumption through optimal integration of DVS and DPM. In: Proc of IEEE International Conference on Computer-Aided Design-Digest of Technical Papers. San Jose, CA, USA: IEEE, 2009,449-456

[49] Ullman J D. NP -complete scheduling problems *. Journal of Computer & System Sciences, 1975, 10(3):384-393

[50] Wu M Y, Gajski D. A Programming Aid for Message-passing Systems. In: Proc of Siam Conference on Parallel Processing for Scientific Computing. Siam: Society for Industrial and Applied Mathematics, 1987,328-332

[51] Kwok Y K, Ahmad I. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. IEEE Trans Parallel & Distributed Systems, 1996, 7(5):506-521

[52] Sih G C, Lee E A. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. Parallel & Distributed Systems IEEE Transactions on, 1993, 4(2):175-187

[53] El-Rewini H, Lewis T G. Scheduling parallel program tasks onto arbitrary target machines. Journal of Parallel & Distributed Computing, 1998, 9(2):138-153

[54] Topcuouglu H, Hariri S, Wu M Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(3):260-274

[55] Yao F, Demers A, Shenker S. A scheduling model for reduced CPU energy. In: Proc of 36th Annual Symposium on Foundations of Computer Science. Milwaukee, WI, USA: IEEE, 2002,374-382

[56] Barnett J A. Dynamic task-level voltage scheduling optimizations. IEEE Transactions on Computers, 2005, 54(5):508-520

[57] Bunde D P. Power-aware scheduling for makespan and flow. Journal of Scheduling, 2009, 12(5):489-500

[58] Rusu C A, Melhem R, Mosse D. Maximizing the system value while satisfying time and energy constraints. Ibm Journal of Research & Development, 2002, 47(5.6):689-702

[59] Li K. Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed. IEEE Transactions on Parallel & Distributed Systems, 2008, 19(11):1484-1497

[60] Funaoka K, Takeda A, Kato S, et al. Dynamic voltage and frequency scaling for optimal real-time scheduling on multiprocessors. In: Proc of International Symposium on Industrial Embedded Systems. Le Grande Motte, France: IEEE, 2008, 27-33

[61] Zong Z, Manzanares A, Ruan X, et al. EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters. IEEE Transactions on Computers, 2011, 60(3):360-374

[62] Huang Q, Su S, Li J, et al. Enhanced Energy-Efficient Scheduling for Parallel Applications in Cloud. In: Proc of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Ottawa, ON, Canada: IEEE, 2012,781-786

[63] Xie G, Jiang J, Liu Y, et al. Minimization Energy Consumption of Real-Time Parallel Applications using Downward and Upward Approaches on Heterogeneous Systems. IEEE Transactions on Industrial Informatics, 2017, PP(99):1-1

[64] Tang Z, Qi L, Cheng Z, et al. An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment. Journal of Grid Computing, 2016, 14(1):55-74

[65] Xie G, Zeng G, Li R, et al. Energy-Aware Processor Merging Algorithms for Deadline Constrained Parallel Applications in Heterogeneous Cloud Computing. IEEE Transactions on Sustainable Computing, 2017, 2(2):62-75

[66] Li K. Scheduling Precedence Constrained Tasks with Reduced Processor Energy on Multiprocessor Computers. IEEE Transactions on Computers, 2012, 61(12):1668-1681

[67] Shatz S M, Wang J P. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. IEEE Transactions on Reliability, 2002, 38(1):16-27

[68] Aupy G, Benoit A, Robert Y. Energy-aware scheduling under reliability and makespan constraints. 2012, 15(3):1-10

[69] Pop P, Poulsen K H, Izosimov V, et al. Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems. In: Proc of 5th IEEE/ACM/IFIP International Conference on

Hardware/software Codesign and System Synthesis. Salzburg, Austria: IEEE, 2007,233-238

[70] Junhe, Gruian, Flavius, et al. Energy/reliability trade-offs in fault-tolerant event-triggered distributed embedded systems. In: Proc of 16th Asia and South Pacific Design Automation Conference. Yokohama, Japan: IEEE, 2011,731-736

[71] Zhang L, Li K, Xu Y, et al. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Information Sciences An International Journal, 2015, 319(C):113-131

[72] Zhang L, Li K, Li K, et al. Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems. International Journal of Electrical Power & Energy Systems, 2016, 78:499-512

[73] Zhang L, Li K, Li C, et al. Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Information Sciences, 2016, 379: 241–256

[74] Girault A, Kalla H. A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate. IEEE Transactions on Dependable & Secure Computing, 2009, 6(4):241-254

[75] Yi J, Zhuge Q, Hu J, et al. Reliability-Guaranteed Task Assignment and Scheduling for Heterogeneous Multiprocessors Considering Timing Constraint. Journal of Signal Processing Systems, 2015, 81(3):359-375

[76] Xie G, Chen Y, Liu Y, et al. Resource Consumption Cost Minimization of Reliable Parallel Applications on Heterogeneous Embedded Systems. IEEE Transactions on Industrial Informatics, 2016, 13(4):1629-1640

[77] Gan J. Tradeoff analysis for dependable real-time embedded systems during the early design phases: [Ph.D. dissertation of Technical University of Denmark]. Lyngby, Danmarks, Technical University of Denmark, 2014, 39-44

[78] Task graph generator. https://sourceforge.net/projects/taskgraphgen/, 2015-08-02

致　谢

从本科到研究生，不知不觉我已在湖大呆了七个年头。依稀记得 2011 年的秋季，懵懂的我带着一颗求知的心踏进了岳麓山下的这片“没门”的校园，从此开始了快乐的学习生涯。2015 年送别了本科的同学与朋友，而我选择了留下继续学习。但时间总是过得那么快，转眼之间又到了毕业的季节，怀揣千般不舍即将离开学校。在此，我要感谢这些年来帮助和鼓励过我的老师、同学和家人，是他们让我的学习生涯不再孤独，并且顺利地完成学业。

首先我要感谢我的导师李仁发教授。李老师渊博的知识、前言的视野以及务实的学术风范深深地影响着我。从论文的选题到实验，再到写作环节，李老师给我提供了非常专业且严谨的指导，小组汇报上的点评为我拨开迷雾，让我能把握住问题的关键所在。生活上，您对学生无微不至的关怀让我铭记于心，时刻为学生着想，让我们能一心一意地进行学术研究。在此表示对恩师深深地敬意。

特别感谢实验室的谢国琪老师，而平日里我和同学更喜欢“师兄”、“琪哥”这样的称呼。谢老师给我的学术论文中的实验以及撰写提供了非常专业的帮助，是我在研究方向上的指路人，和谢老师的交流让我能在正确道路上去学习和研究。谢老师一丝不苟的学术态度，不仅是我，更是全实验室学生学习的榜样。

感谢实验室的谢勇老师和吴迪老师。谢勇老师带我第一次走进汽车电子领域，由于地理原因未能和谢老师进行广泛、深入的学术交流，但谢老师严谨的学术研究思路一直影响着我；和吴迪老师做无人机项目让我学习到了怎样和团队做实际工程项目研究，对我如何解决研究中的难题提供了一些思路。感谢实验室付彬老师在我学习生涯中提供的帮助，感谢刘彦老师在我开题以及中期检查上提供的指导建议，感谢所有教过我课程的老师。

感谢实验室的吴武飞师兄、黄晶师兄、周佳师兄、李万里师兄、白洋师姐，博士师兄师姐们都平易近人，不仅在学习上给予我启迪，在生活中也十分关照，让我感受到实验室的温暖。在这里要特别感谢周佳师兄，佳哥既是我学习的榜样，也是我大学以及研究生生涯最重要伙伴。感谢实验室同门邓湘军、袁娜、屠晓涵，和你们一起学习让我倍感快乐。感谢研究生室友莫济谦、王泽朋、顾文武在生活中相互帮助与支持，感谢实验室一群带我打羽毛球的小伙伴，是你们让我的学习生涯丰富多彩。

特别感谢我的父母以及亲戚朋友，你们对我无私的爱与支持让我不断前进，让我顺利地完成学业，我将永远记得你们为我付出的心血。

最后衷心感谢评审专家以及答辩组老师们在百忙中对本文的审阅与指导。

附录 A 攻读硕士学位期间发表的学术论文

[1] Jinlin Song, Guoqi Xie, Renfa Li, Xiaoming Chen. An Efficient Scheduling Algorithm for Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems. In: Proc of 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2017:32-39. (CCF C).

[2] Guoqi Xie, Hao Peng, Zhetao Li, Jinlin Song, Yong Xie, Renfa Li, Keqin Li. Reliability Enhancement Towards Functional Safety Goal Assurance in Energy-Aware Automotive Cyber-Physical Systems. IEEE Transactions on Industrial Informatics. (major revision).

附录 B 攻读硕士学位期间所参与的项目

[1] 国家自然科学基金 [61672217]：新一代汽车嵌入式系统功能安全的建模与算法研究.

[2] 国家自然科学基金 [61702172]：基于AUTOSAR新平台标准的汽车CPS自适应安全调度.