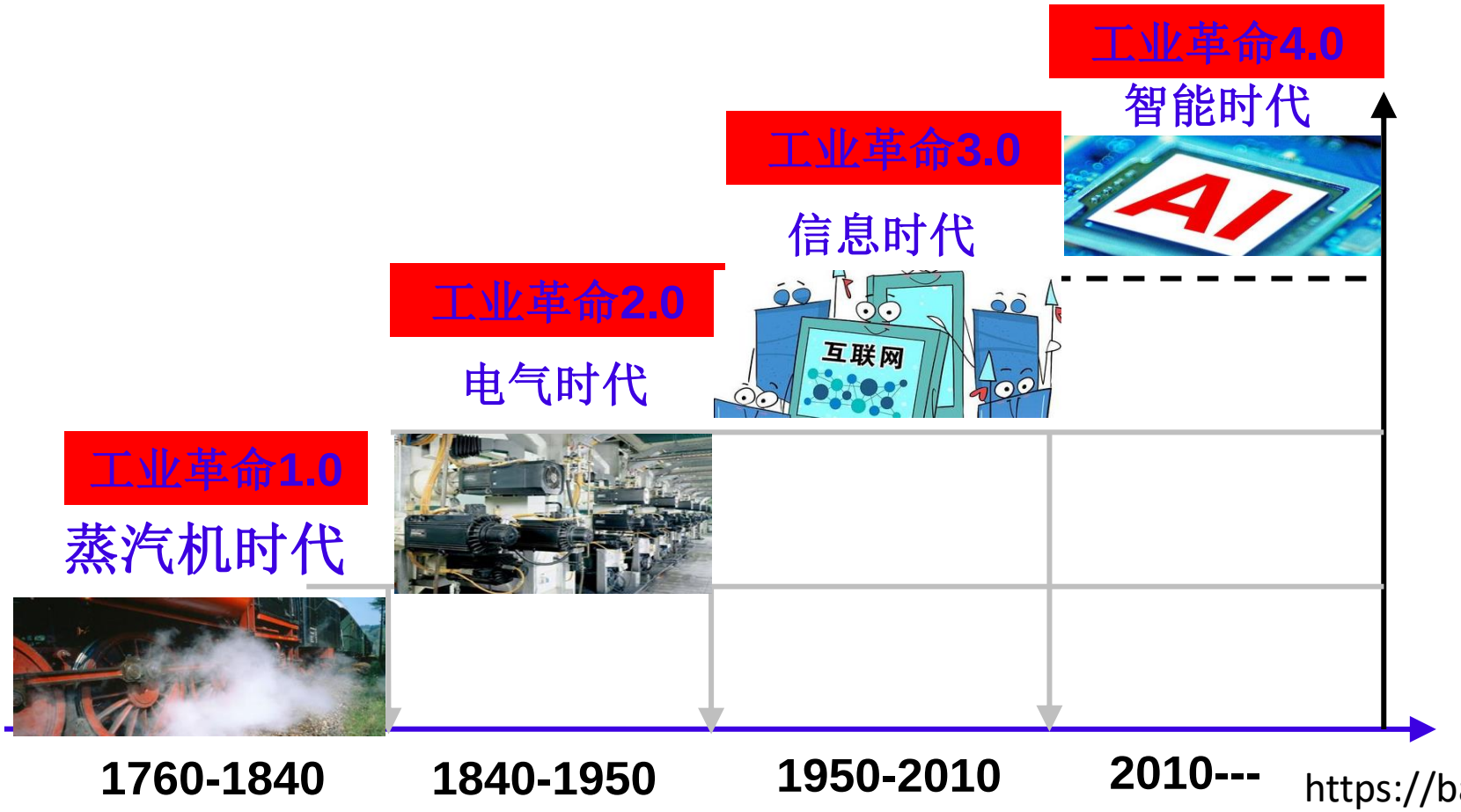
智能嵌入式系统软硬件优化配置方法

陈仪香
华东师范大学软件工程学院可信智能团队TrIG
2023年8月12日
第五届国产嵌入式操作系统技术与产业发展论坛

TrIG 华东师范大学软件学院可信智能团队
Trustworthy Intelligence Group

智能嵌入式系统: 智能制造



https://baijiahao.baidu.com/

智能嵌入式系统

- 智能嵌入式系统是现代社会的关键器件之一。
  - 智能嵌入式系统是先进的计算机技术、半导体技术、新一代人工智能技术等与各个行业的具体应用相结合的产物,正在成为各种智能体系的基础,具有技术密集、资金密集、高度分散、不断创新的特点。
  - 智能手机是典型的智能嵌入式系统，而围棋机器人Alpha Go虽然也呈现了嵌入式技术，但更体现了并行计算和高性能计算特征。

王泉, 吴中海, 陈仪香, 苗启广. 智能嵌入式系统专题前言. 软件学报, 2020, 31(9): 2625−2626

大　　纲

无处不在的智能嵌入式系统

智能嵌入式系统软硬件优化配置

基于CNN的交通标识识别系统

大　　纲

无处不在的智能嵌入式系统

智能嵌入式系统软硬件优化配置

基于CNN的交通标识识别系统

无处不在的智能嵌入式系统

日常生活用品
可穿戴产品
智能家居
智能汽车
智慧医疗器材

新一代飞机
航天飞行器
太空探测
智能制造
（工业革命4.0）

无处不在的智能嵌入式系统：日常生活用品

无处不在的智能嵌入式系统： 可穿戴产品

无处不在的智能嵌入式系统： 高铁

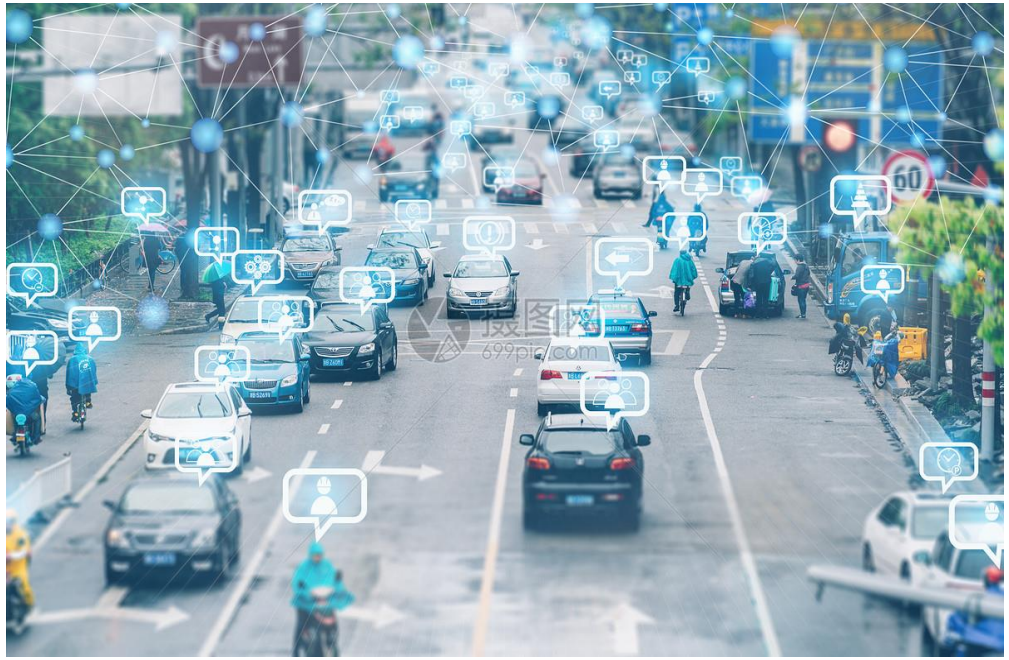
https://cn.bing.com/images

无处不在的智能嵌入式系统： 飞机

中国商用飞机

中国军用飞机

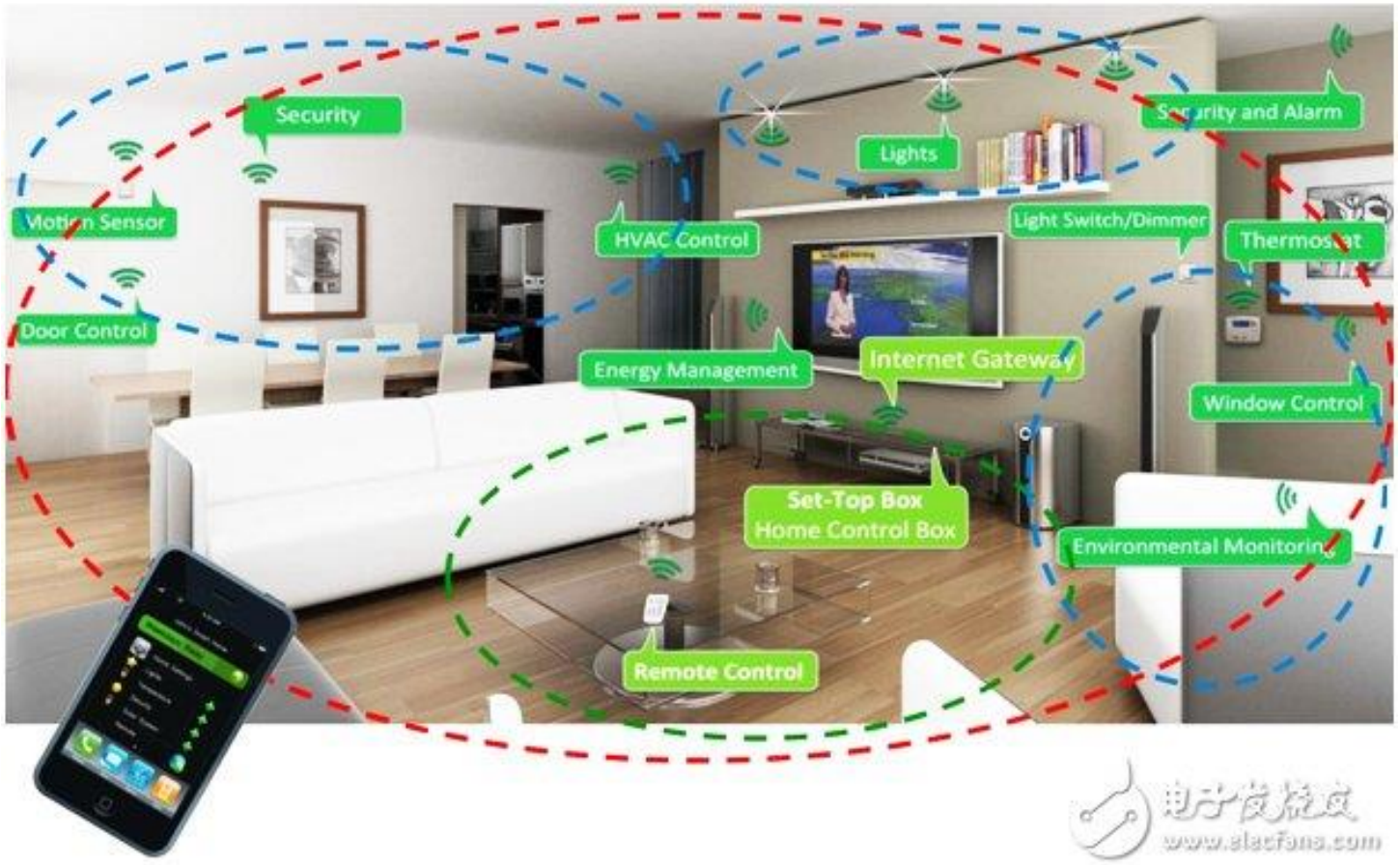
智能嵌入式系统: 智能交通



http://seopic.699pic.com/photo/40099/1953.jpg_wh1200.jpg

http://www.qyev.com/repository/image/WKrdHMgRTu-AyANQUjrilA.jpg_%7Bi%7Dxaf.jpg

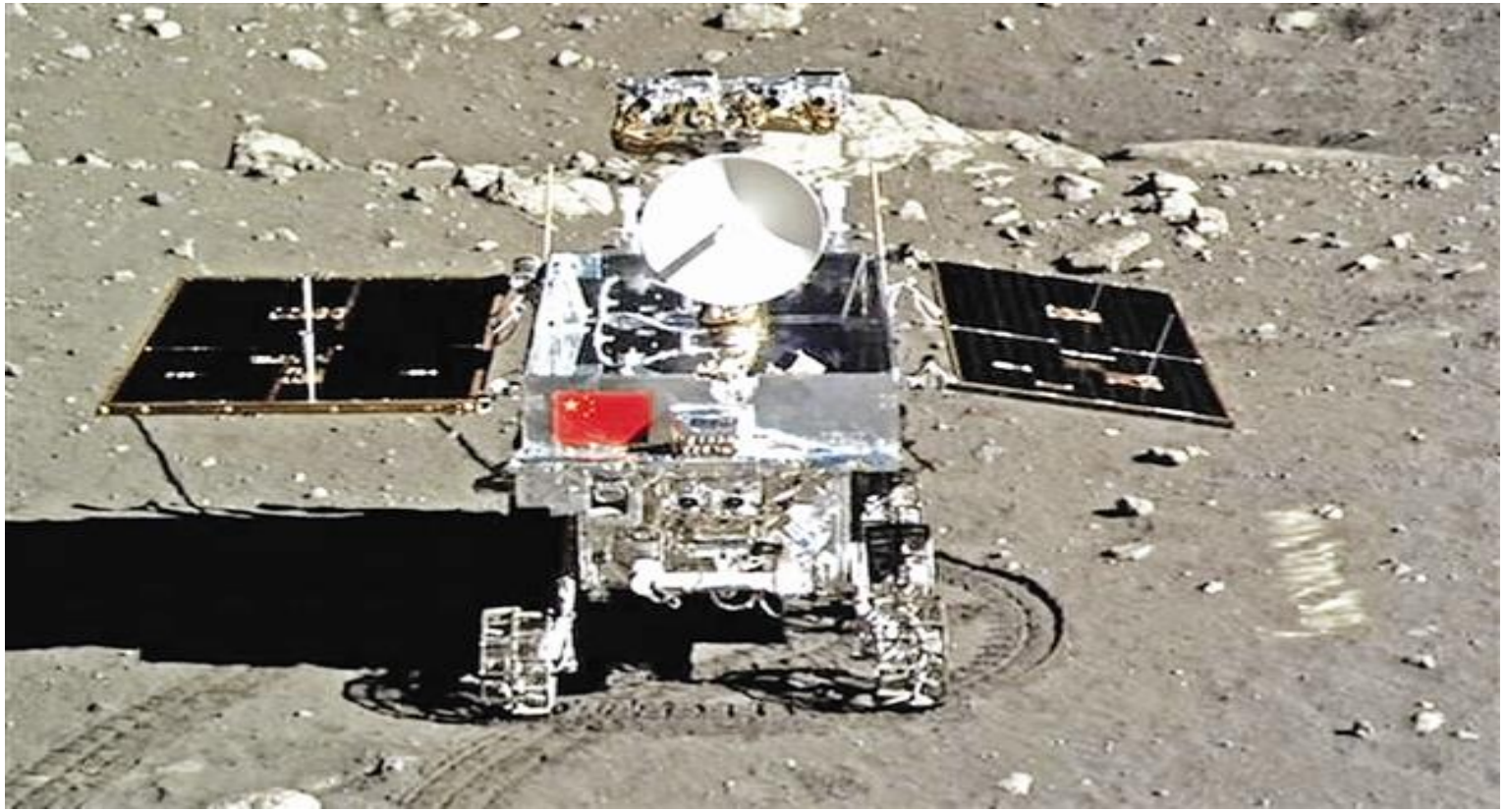
智能嵌入式系统: 智能家居



智能嵌入式系统：　　智能探测

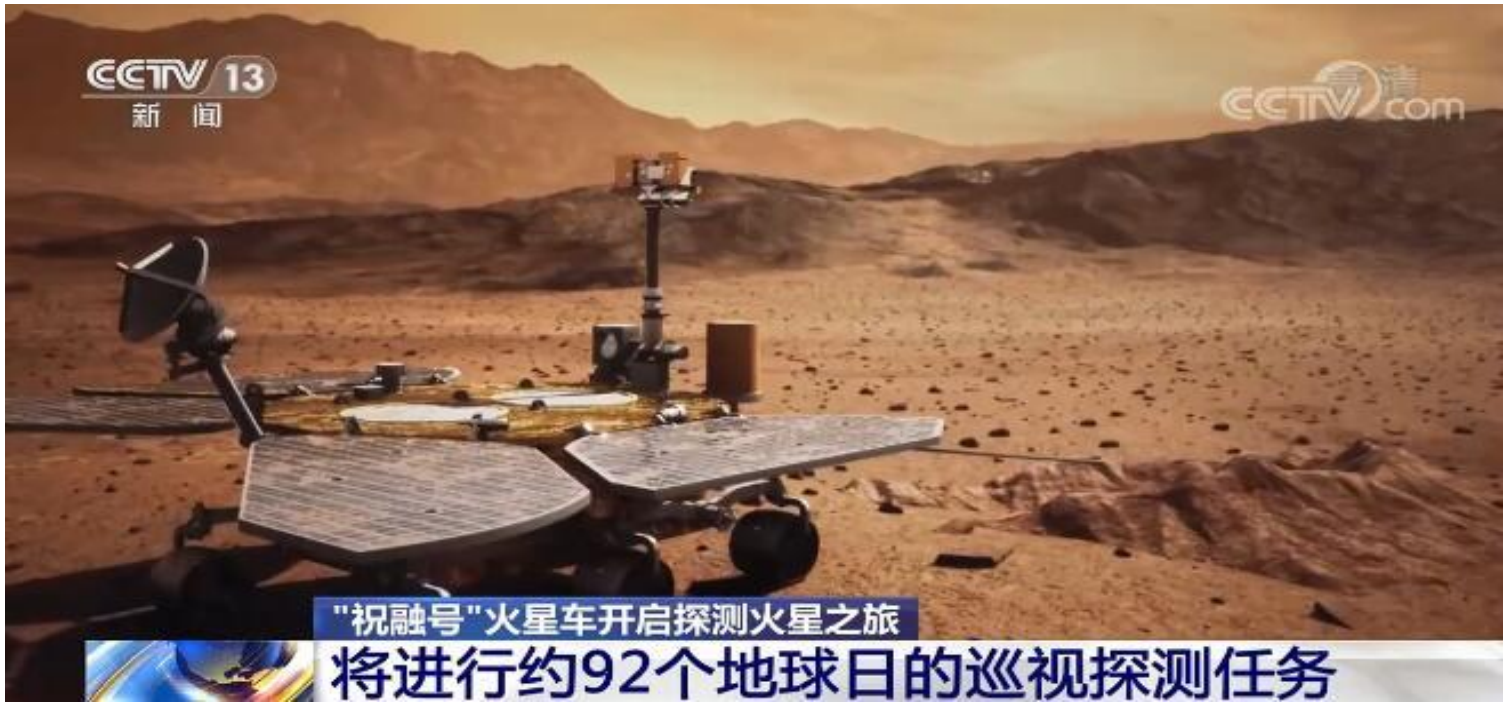
https://baike.baidu.com/item/中国探月工程/8492213

智能嵌入式系统: 智能探测



玉兔号登月车

智能嵌入式系统：　　智能探测



祝融号火星车

智能嵌入式系统：　　空间站

空间站天宫2号

智能嵌入式系统：　　智能医疗

http://p0.itc.cn/images01/20200707/6f71ae9dfc3b4e35b429a998be098d92.jpeg

智能嵌入式系统

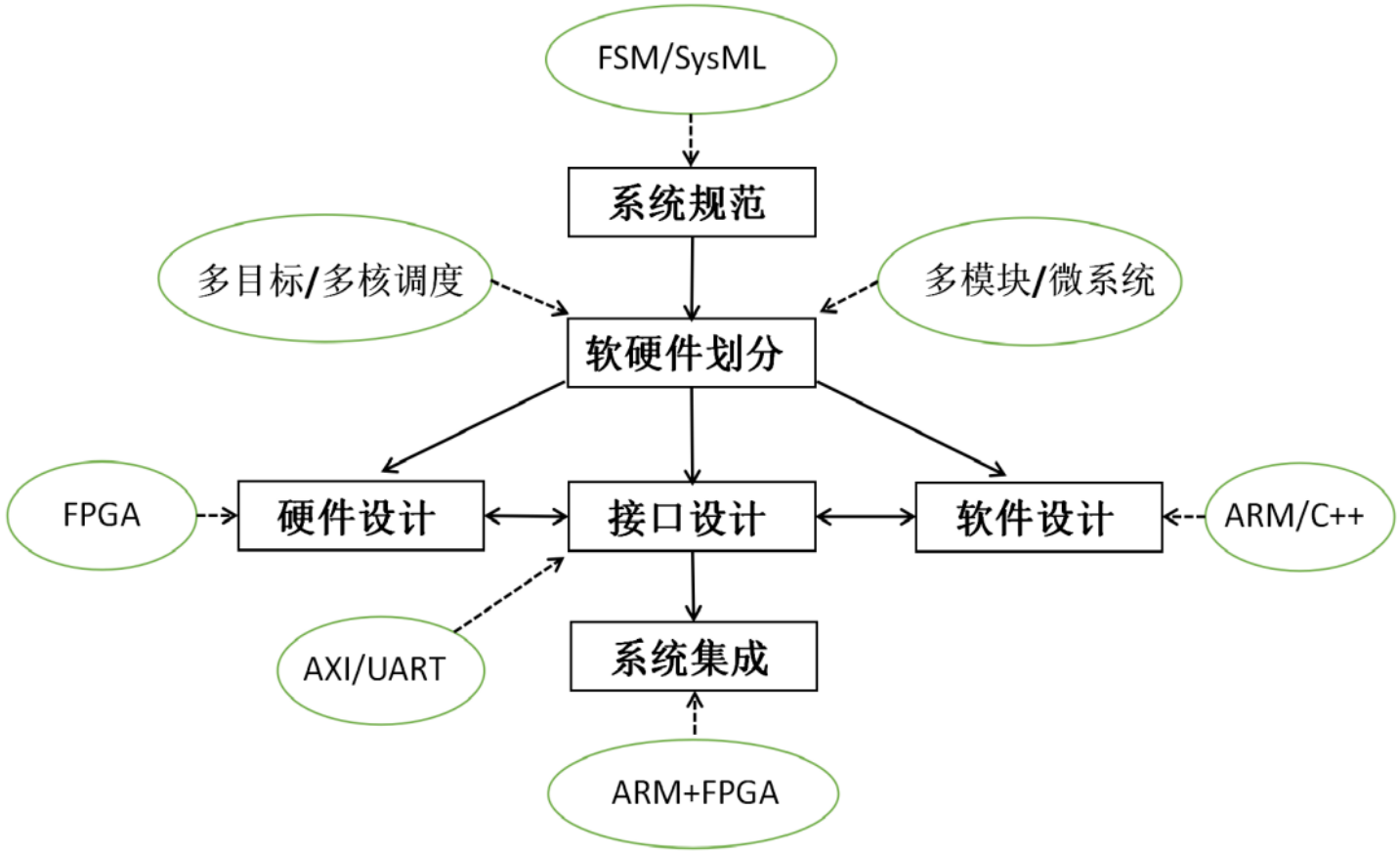
- 智能嵌入式系统特征
  - ✓ 智能化
  - ✓ 集成化
  - ✓ 性能最优化
  - ✓ 功能最大化
  - ✓ 微型化
  - ✓ 片型化
- 软硬件优化配置是实现智能嵌入式系统特征的有效方法之一。

智能嵌入式系统

- 优化设计是智能嵌入式系统设计与实现的最基本方法和技术之一，它使智能嵌入式系统产品的多项性能指标进行优化。

- 优化设计技术是依据嵌入式系统的需求进行软硬件划分配置与设计以及系统集成，实现嵌入式系统的整体性能最优化。

智能嵌入式系统



智能嵌入式系统软硬件优化设计架构

智能嵌入式系统设计

陈仪香、陈彦辉编著

机械工业出版社，2023年7月

由华东师范大学陈仪香与西安电子科技大学陈彦辉联合撰写的《智能嵌入式系统设计》由机械工业出版社出版了，网上书店有售。

这本书包含了基础篇（建模与仿真）、核心篇（软硬件划分）、实践篇（基于卷积神经网络的交通标识识别系统设计与实现）。

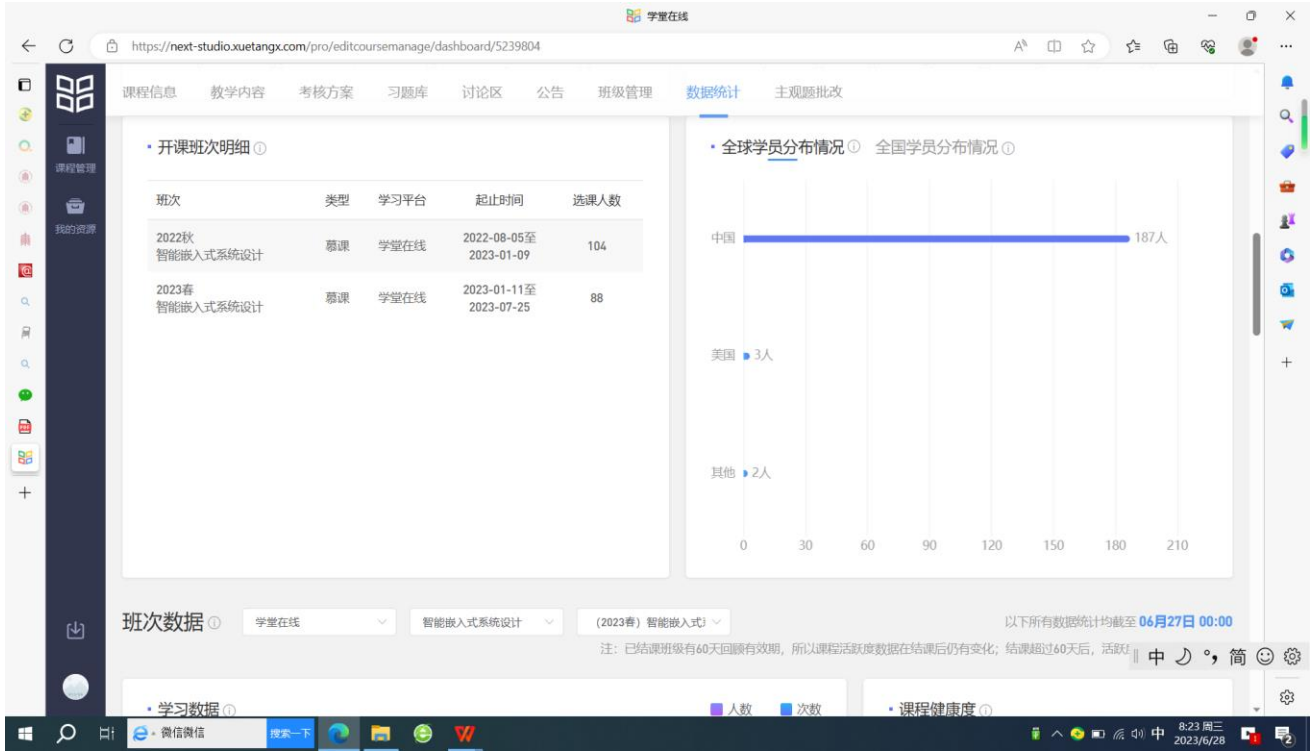
该书适合作为高等院校和科研院所电子信息类专业高年级本科生和研究生的教材，以及数学信息类和工程技术类专业学生的教学参考书，也适合从事计算机科学、软件开发技术、电子工程实践等信息领域的专业人员阅读参考。

与该书配套的慕课在“学堂在线平台”开课，课程名称仍是《智能嵌入式系统设计》。

慕课

- 慕课名称：智能嵌入式系统设计
- 课程上线平台：学堂在线
- 慕课老师：陈仪香， 华东师范大学

- https://next-studio.xuetangx.com/pro/editcoursemanage/courseinfo/5239804
- 课程上线时间：2022年8月份，第一次开课
- 2023年1月份，第二次开课



大　　纲

无处不在的智能嵌入式系统

智能嵌入式系统软硬件优化配置

基于CNN的交通标识识别系统

软硬件优化配置方法-核心篇

第五章 多目标划分（兼权熟计）

第六章 多核划分（万物并育）

第七章 多模块划分（絜矩之道）

第八章 微系统划分（鸢飞戾天）

软硬件优化配置方法-核心篇

第五章 多目标划分（兼权熟计）

第六章 多核划分（万物并育）

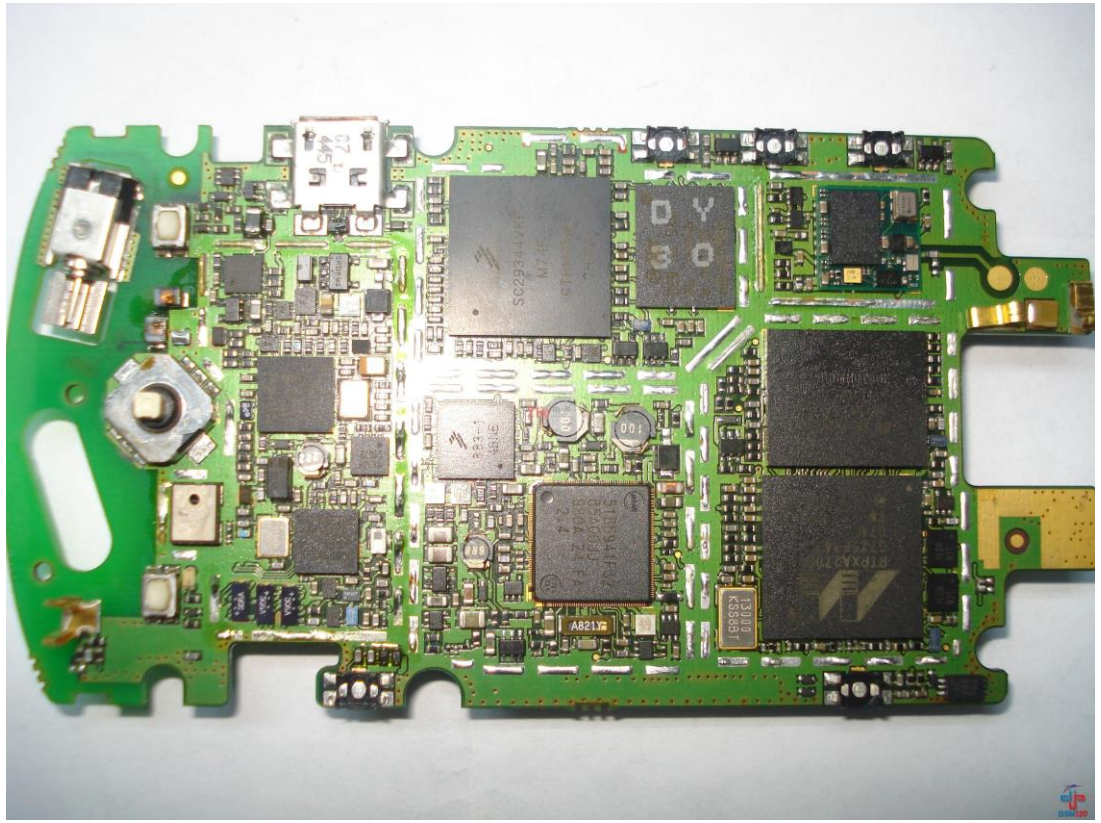
第七章 多模块划分（絜矩之道）

第八章 微系统划分（鸢飞戾天）

第8章 微系统划分

- 智能系统通常会含有众多任务，这些任务之间通过通信进行消息/数据/事件传输，进行协同完成整个系统需要完成的任务。
- 为了满足时间性能的要求，人们会把这些任务划分成多个模块，每个模块由一个或多个处理器或软件与硬件融合的异构平台进行处理。
- 模块内依据任务的性能指标（如时间、功耗、成本、硬件面积等）进行软硬件优化划分。
- 这样处理 一方面可以缩短设计周期，极大地提高设计效率， 另一方面可以根据系统各个部分的特点和设计约束选择软件或硬件实现方式，得到高性能、低成本的优化设计方案。
- 智能系统划分是将系统的规范按照软件部分和硬件部分进行划分，形成软件的规范和硬件的规范，为软件与硬件实现提供规范支撑。智能系统划分是智能系统优化设计的关键技术之一，它对整个系统的设计结果有着重要的影响。

第8章 微系统划分 微系统



- 微系统是由若干模块组成的一个板上系统
- 每个模块间通过通信完成数据传输
- 每个模块完成若干任务
- 每个任务可以软件实现也可以硬件实现

第8章 微系统划分 微系统

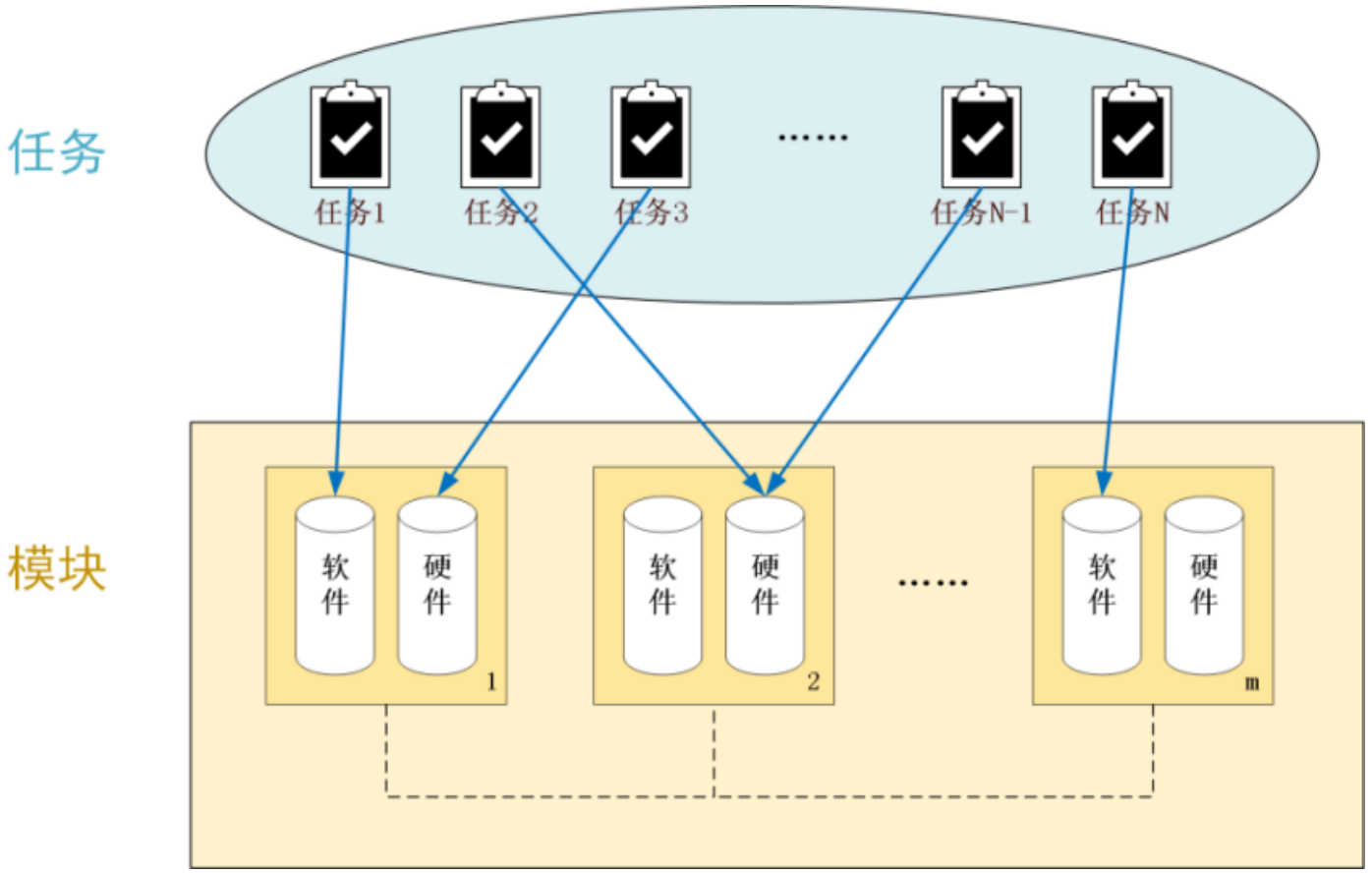
- 微系统是由若干模块组成的一个板上系统，每个模块间通过通信完成数据传输，每个模块完成若干任务，每个任务可以软件实现也可以硬件实现。
- 微系统划分是
  - 将整个系统的N个任务划分成m个模块， 使得模块间的通信代价最小；
  - 同时在每个模块内依据任务性能进行软硬件划分实现性能最优；
  - 最后确定的模块和软硬件划分结果称为这个系统的综合解决方案。

如何把一个智能系统划分成若干个模块， 而每个模块中的每个任务进行软硬件配置，建立一个软硬件整体配置方案。

第8章 微系统划分 基于模块的微系统划分

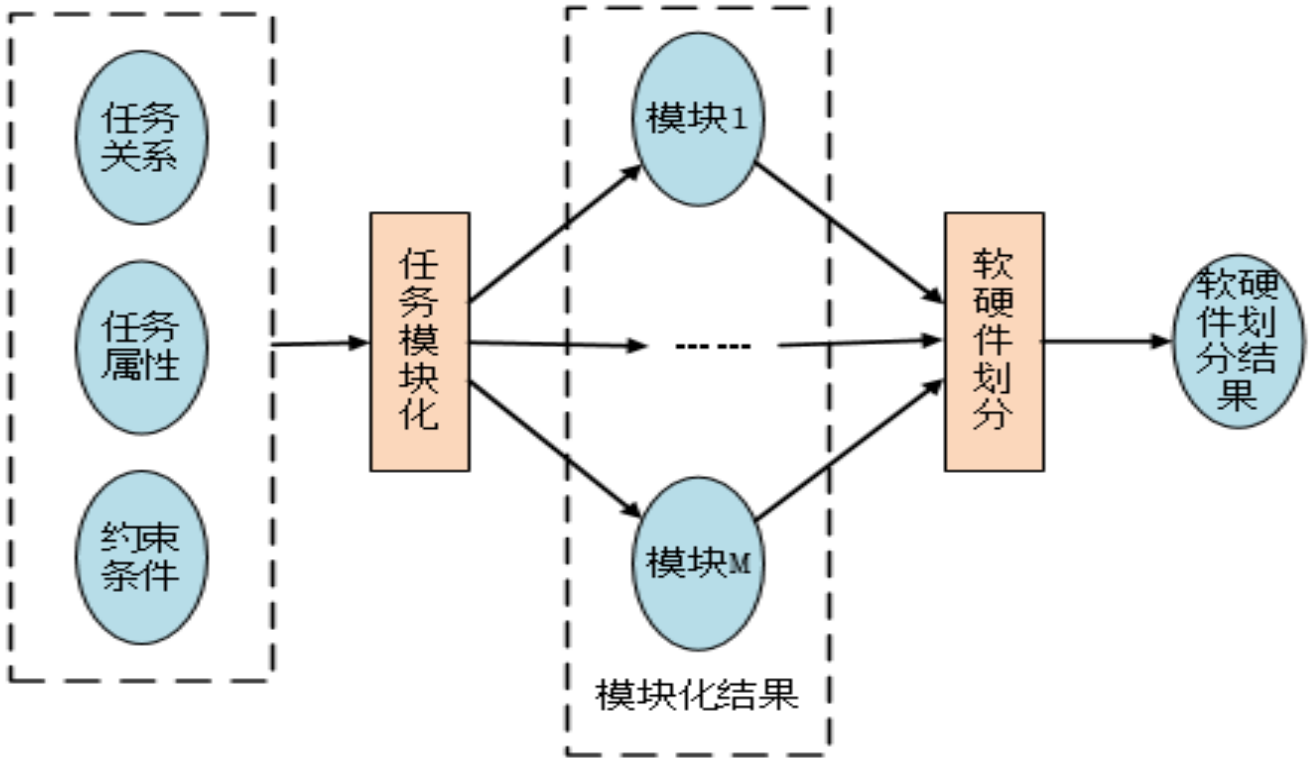
- 本节介绍基于模块的微系统划分方法，是将多模块划分方法和多目划分方法组合在一起：
  - 基于通信代价将整个系统进行模块化（多模块划分），
  - 对每个模块依据任务的多维性能属性进行软硬件划分（多目标划分）。
- 构建基于模块的微系统划分方法

第8章 微系统划分 基于模块的微系统划分



微系统划分示意图

第8章 微系统划分 基于模块的微系统划分



划分过程示意图

第8章 微系统划分 基于模块的微系统划分--例子

- 例8.1 有A1-A8八个任务，他们之间的通信代价和各任务的属性如表8-1和表8-2所示，

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	1	3	6	10	2	5	7
A2		0	8	2	10	12	13	15
A3			0	8	20	13	15	17
A4				0	19	18	17	5
A5					0	20	11	18
A6						0	12	13
A7							0	5
A8								0

表8-1 任务通信代价

第8.1节 基于模块的微系统划分--例子

- 例8.1：有A1-A8八个任务，他们之间的通信代价和各任务的属性如表8-1和表8-2所示，

任务	软时间	硬时间	硬面积	软可靠度	硬可靠度
A1	30	10	5	0.7	0.9
A2	40	12	6	0.65	0.8
A3	42	10	4	0.7	0.88
A4	35	9	5	0.76	0.91
A5	34	8	5	0.88	0.92
A6	22	4	2	0.89	0.92
A7	23	5	4	0.76	0.88
A8	20	6	5	0.75	0.86

表8-2 各任务的属性

第8章 微系统划分 基于模块的微系统划分--例子

- 例8.1 有A1-A8八个任务，他们之间的通信代价和各任务的属性如表8-1和表8-2所示，

表8-1 任务通信代价

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	1	3	6	10	2	5	7
A2		0	8	2	10	12	13	15
A3			0	8	20	13	15	17
A4				0	19	18	17	5
A5					0	20	11	18
A6						0	12	13
A7							0	5
A8								0

表8-2各任务的属性

任务	软时间	硬时间	硬面积	软可靠度	硬可靠度
A1	30	10	5	0.7	0.9
A2	40	12	6	0.65	0.8
A3	42	10	4	0.7	0.88
A4	35	9	5	0.76	0.91
A5	34	8	5	0.88	0.92
A6	22	4	2	0.89	0.92
A7	23	5	4	0.76	0.88
A8	20	6	5	0.75	0.86

现将这8个任务分成3个模块，使得每个模块不超过3个任务，且通信消耗最小；每个模块依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

第8章 微系统划分 基于模块的微系统划分--例子

第一步：划分模块。

- 依据（单链接）通信代价将8个任务划分成3个模块：
  - M1: {A3,A5,A6},
  - M2:{A1,A4,A8},
  - M3:{A2,A7}。
- 模块间通信代价：
  - C_{M1M2}=108,
  - C_{M1M3}=45,
  - C_{M2M3}=68,
- 整个通信代价为221.

第三步：计算整个系统的性能。

软时间	硬时间	硬面积	通信代价	可靠度
65	49	25	221	0.843

第二步：将每个模块进行软硬件划分,并计算其属性。

模块划分	软硬件划分	软时间	硬时间	硬面积	可靠度
M1：A3,A5,A6	软件：A6	22	18	9	0.897 (2.69/3)
	硬件：A3,A5,				
M2：A1,A4,A8	软件：A8	20	19	10	0.853(2.56/3)
	硬件：A1,A4				
M3：A2,A7	软件：A7	23	12	6	0.78(1.56/2)
	硬件：A2				

第8章 微系统划分 基于模块的微系统划分--例子

第一步：划分模块。

- 依据（单链接）通信代价将8个任务划分成3个模块：
  - M1: {A3,A5,A6},
  - M2:{A1,A4,A8},
  - M3:{A2,A7}。
- 模块间通信代价：
  - C_{M1M2}=108,
  - C_{M1M3}=45,
  - C_{M2M3}=68,
- 整个通信代价为221.

现将这8个任务分成3个模块，使得每个模块不超过3个任务，且通信消耗最小；每个模块依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

第三步：计算整个系统的性能。

软时间	硬时间	硬面积	通信代价	可靠度
65	49	25	221	0.843

第二步：将每个模块进行软硬件划分,并计算其属性。

模块划分	软硬件划分	软时间	硬时间	硬面积	可靠度
M1: A3,A5,A6	软件：A6	22	18	9	0.897 (2.69/3)
	硬件：A3,A5,				
M2: A1,A4,A8	软件：A8	20	19	10	0.853(2.56/3)
	硬件：A1,A4				
M3：A2,A7	软件：A7	23	12	6	0.78(1.56/2)
	硬件：A2				

性能指标约束是按照模块进行的。若把性能指标约束在整个系统层进行考虑，情况如何?

第8章 微系统划分 基于模块的微系统划分--例子

- **例8.1：** 有A1-A8 八个任务，他们之间的通信代价和各任务的属性如表8-1-1和表8-1-2所示，

表8-1 任务通信代价

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	1	3	6	10	2	5	7
A2		0	8	2	10	12	13	15
A3			0	8	20	13	15	17
A4				0	19	18	17	5
A5					0	20	11	18
A6						0	12	13
A7							0	5
A8								0

表8-2各任务的属性

任务	软时间	硬时间	硬面积	软可靠度	硬可靠度
A1	30	10	5	0.7	0.9
A2	40	12	6	0.65	0.8
A3	42	10	4	0.7	0.88
A4	35	9	5	0.76	0.91
A5	34	8	5	0.88	0.92
A6	22	4	2	0.89	0.92
A7	23	5	4	0.76	0.88
A8	20	6	5	0.75	0.86

现将这8个任务分成3个模块，使得每个模块不超过3个任务，在满足系统约束条件软时间<=275，硬面积<=29情况下，使得通信消耗最小，每个模块依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

第8章 微系统划分 基于模块的微系统划分--例子

第一步：划分模块。

- 依据（单链接）通信代价将8个任务划分成3个模块：
  - M1: {A3,A5,A6},
  - M2:{A1,A4,A8},
  - M3:{A2,A7}。
- 模块间通信代价：
  - C_{M1M2}=108,
  - C_{M1M3}=45,
  - C_{M2M3}=68,
- 整个通信代价为221.

任务	软时间	硬时间	硬面积	软可靠度	硬可靠度
A1	30	10	5	0.7	0.9
A2	40	12	6	0.65	0.8
A3	42	10	4	0.7	0.88
A4	35	9	5	0.76	0.91
A5	34	8	5	0.88	0.92
A6	22	4	2	0.89	0.92
A7	23	5	4	0.76	0.88
A8	20	6	5	0.75	0.86

第二步：将每个模块进行软硬件划分,并计算其属性。

模块划分	软硬件划分	软时间	硬时间	硬面积	可靠度
M1：A3,A5,A6	软件：A5, A6 硬件：A3	56	10	4	0.883 (2.65/3)
M2：A1,A4,A8	软件： 硬件：A1,A4, A8	0	25	15	0.89 (2.67/3)
M3：A2,A7	软件： 硬件：A2, A7	0	17	10	0.84 (1.68/2)

现将这8个任务分成3个模块，使得每个模块不超过3个任务，在满足系统约束条件软时间<=275，硬面积<=29情况下，使得通信消耗最小，每个模块依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

第8章 微系统划分 基于模块的微系统划分--例子

模块划分	软硬件划分	软时间	硬时间	硬面积	可靠度
M1：A3,A5,A6	软件：A6 硬件：A3,A5,	22	18	9	0.897 (2.69/3)
M2：A1,A4,A8	软件：A8 硬件：A1,A4	20	19	10	0.853 (2.56/3)
M3：A2,A7	软件：A7 硬件：A2	23	12	6	0.78 (1.56/2)

现将这8个任务分成3个模块，使得每个模块不超过3个任务，且通信消耗最小；每个模块都有各自的约束条件并依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

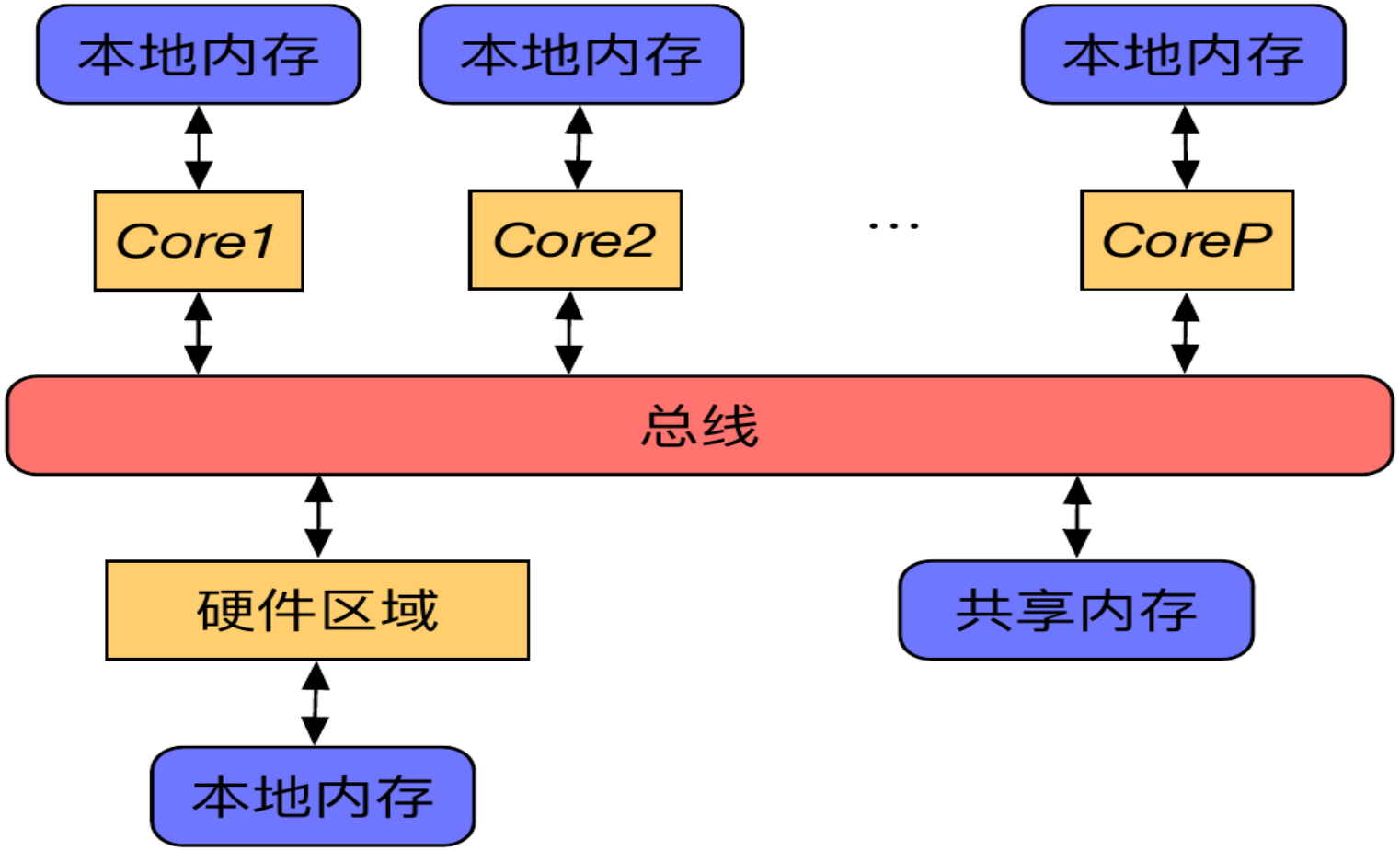
模块划分	软硬件划分	软时间	硬时间	硬面积	可靠度
M1：A3,A5,A6	软件：A5, A6 硬件：A3	56	10	4	0.883 (2.65/3)
M2：A1,A4,A8	软件： 硬件：A1,A4, A8	0	25	15	0.89 (2.67/3)
M3：A2,A7	软件： 硬件：A2, A7	0	17	10	0.84 (1.68/2)

现将这8个任务分成3个模块，使得每个模块不超过3个任务，在满足系统约束条件软时间<=275，硬面积<=29情况下，使得通信消耗最小，每个模块依据任务的性能要求进行软硬件划分，使得系统可靠度最高。

第8章 微系统划分 基于多核的微系统化

- 第6章介绍了多核调度算法，本节在第6章基础上，介绍基于多核的微系统化算法。
- 任务调度是给任务分配计算单元、安排任务执行时间的过程，调度结果对软硬件协同设计中的系统性能有着关键性影响。
- 任务调度与软硬件划分都是软硬件协同设计流程中的重要环节，前者重点考虑任务的执行顺序，而后者主要考虑任务的软硬件实现方式，二者间有必然的内在联系。

第8章 微系统划分 基于多核的微系统化



简化的多核异构片上系统结构

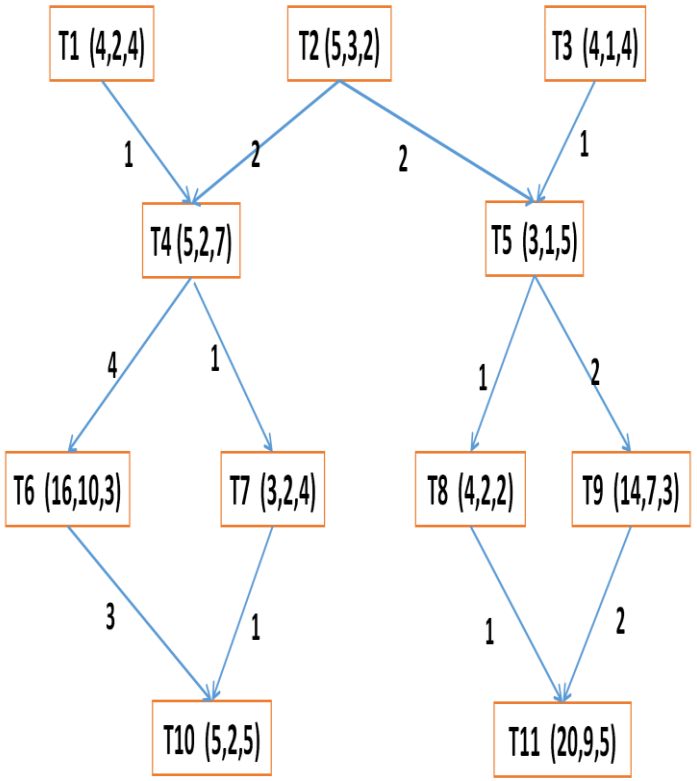
第8章 微系统划分 基于多核的微系统化

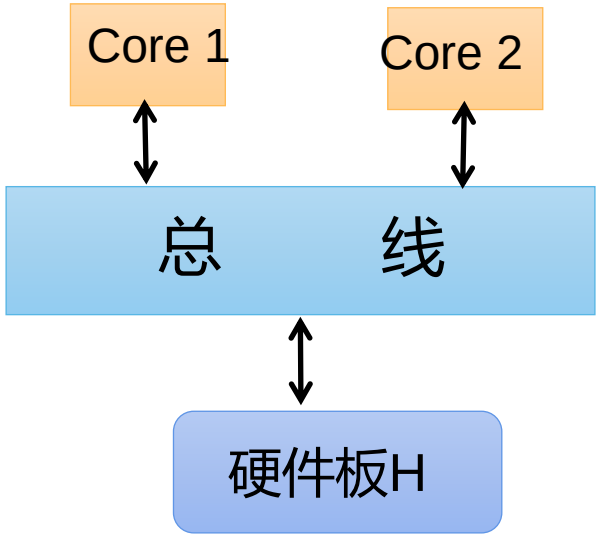
- 含有N个任务图G=(T,E),T是N个任务集,每个任务Ti有3个性能属性=(TSi,THi,AHi)，其中TSi为软件执行时间、THi为硬件执行时间、AHi为硬件执行面积；
- E为图G边的集合，(Ti,Tj)∈E是一条有向边，e(Ti,Tj)为边(Ti,Tj)的权值，表示任务Ti到Tj的通信时间。
- Ti调度后=(pi,tsi,tfi)，其中pi表示任务被分配到哪个处理单元；tsi表示任务的开始时间；tfi表示任务的结束时间；
- 处理器之间/处理器与硬件通信使用总线BUS完成，处理器内部通信以及硬件内部通信看成内部通信不在统计范围，B(Ti->Tj, tcsij, tcfij)。
- 算法目标：min L=$Max_{i\in\{1,2,\ldots,N\}}\{tfi\}$，
- 约束条件：$S=\sum_{i=1}^{N} a_i*AHi <= L$（当Ti为硬件任务是$a_i$=1,否则=0）。

第8章 微系统划分 基于多核的微系统化

例8.3

- 已知一个系统有11个任务：T1,…,T11,每个任务有3个属性表（软件执行时间，硬件执行时间，硬件执行面积），系统的硬件执行面积约束条件为S=18。
- 将系统在2个处理器Core1和Core2 和一块硬件板H上进行软硬件划分，在满足硬件约束条件前提下，系统执行时间最短，其中约定硬件板班H可以2并行执行，即2个任务可以并行执行，
- 系统通信通过Bus总线完成。





第8章 微系统划分 基于硬件实现增益的软硬件划分 $HSPA_{HG}$

- 嵌入式系统的硬件面积是非常重要的一个约束指标， 一方面硬件实现的任务执行时间比软件实现的任务执行时间要少得多，但另一方面硬件实现的任务要占用硬件的面积。
- 从执行时间角度来说，尽可能多得安排任务由硬件实现，但硬件面积的约束导致在安排硬件实现任务时要考虑到硬件实现任务带来的增益，
  - 若一个任务硬件实现的执行时间比软件实现的执行时间少得多，则硬件实现带来的增益要大。

第8章 微系统划分 基于硬件实现增益的软硬件划分 $HSPA_{HG}$

定义8.1：任务硬件实现增益

- 一个任务软件实现执行时间减去硬件实现执行时间的差称为这个任务硬件实现的增益。

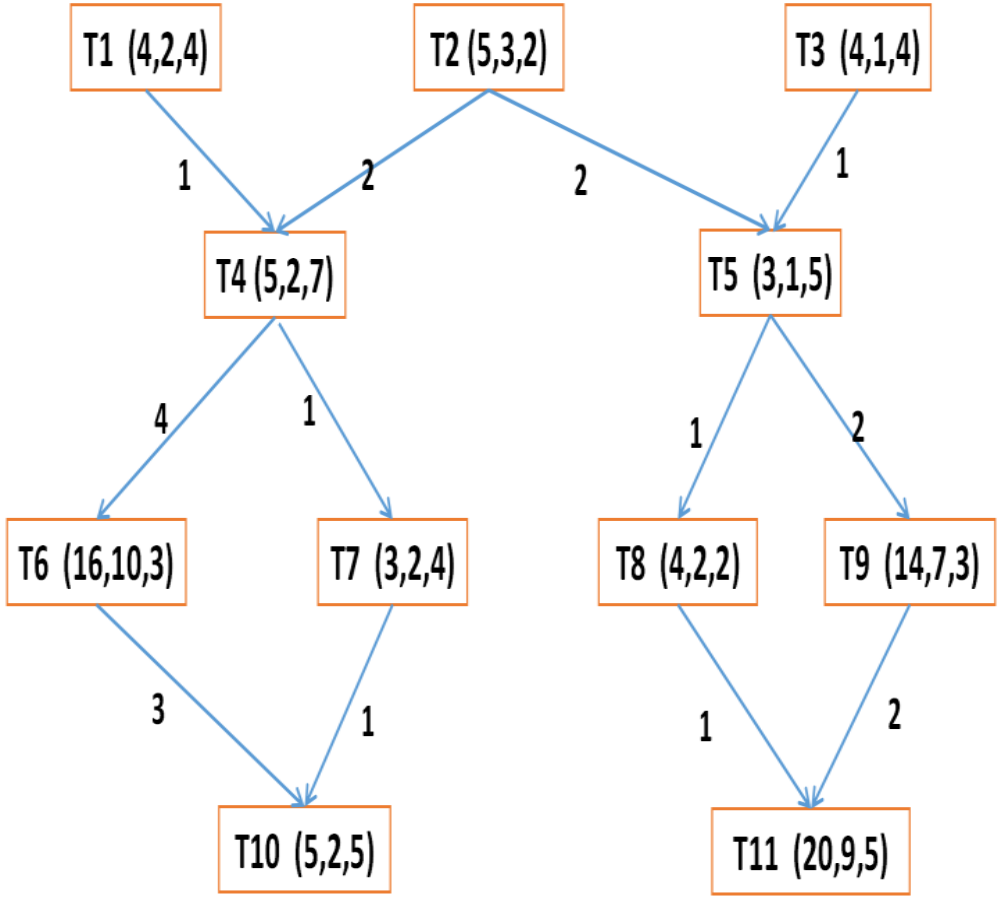
- 任务硬件实现增益越大硬件实现的可能性就越大。
- 可以依据任务的软时间和硬时间数据来构造任务的硬件实现增益表。

第8章 微系统划分 基于硬件实现增益的软硬件划分 $HSPA_{HG}$

定义8.1：任务硬件实现增益

- 一个任务软件实现执行时间减去硬件实现执行时间的差称为这个任务硬件实现的增益。

- 例8.3中11个任务硬件实现增益表为：（2,2,3,3,2,6,1,2,7,3,11）。
- 得到任务T11的硬件实现增益最大（=11），而任务T7的硬件实现增益最小（=1）。
- 任务T11最有可能硬件实现，而任务T7最没有可能硬件实现。

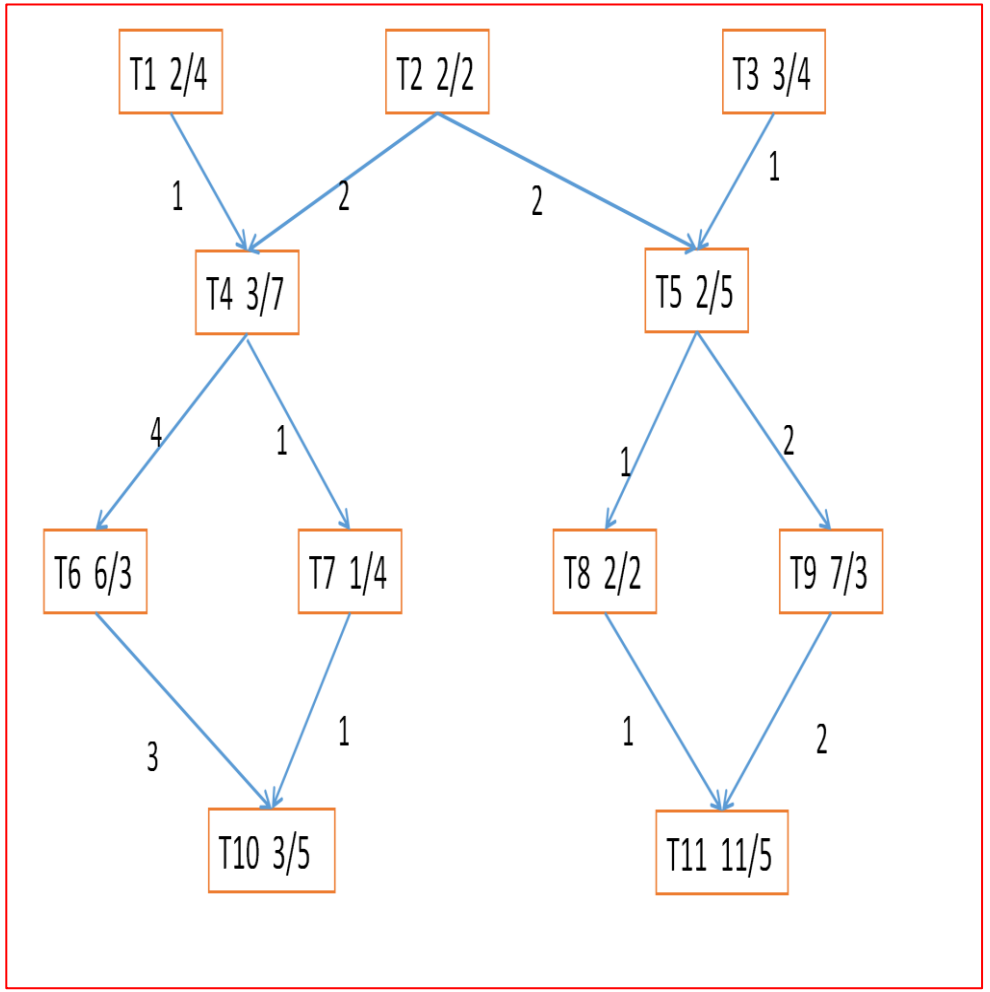


第8章 微系统划分 基于硬件实现增益的软硬件划分 $HSPA_{HG}$

例8.3中11个任务硬件实现增益表为：

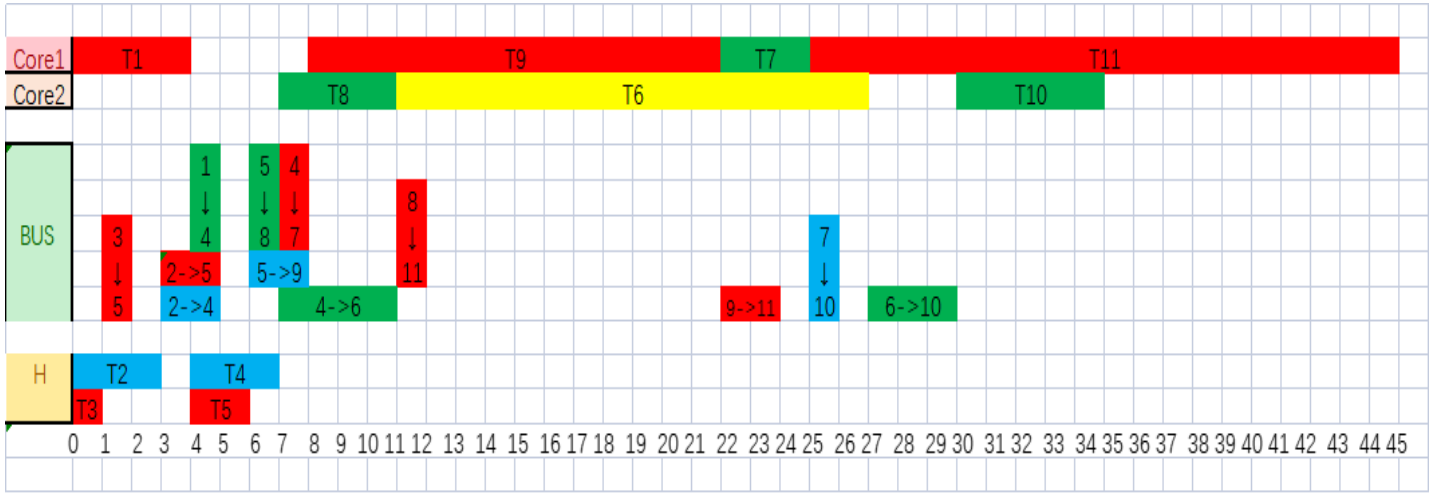
（2,2,3,3,2,6,1,2,7,3,11）。

- 得到任务T11的硬件实现增益最大（=11），而任务T7的硬件实现增益最小（=1）。
- 任务T11最有可能硬件实现，而任务T7最没有可能硬件实现。
- 将任务图中的时间性能属性换成硬件实现增益，保留硬件执行面积，这样就得到一个新的任务图，使用第6章介绍的实时调度算法MuPSA进行软硬件划分，
- 在计算任务优先级值时使用任务硬件实现增益值，在任务调度时累计硬件面积，若硬件面积超过了约束，则停止分配任务给硬件执行。

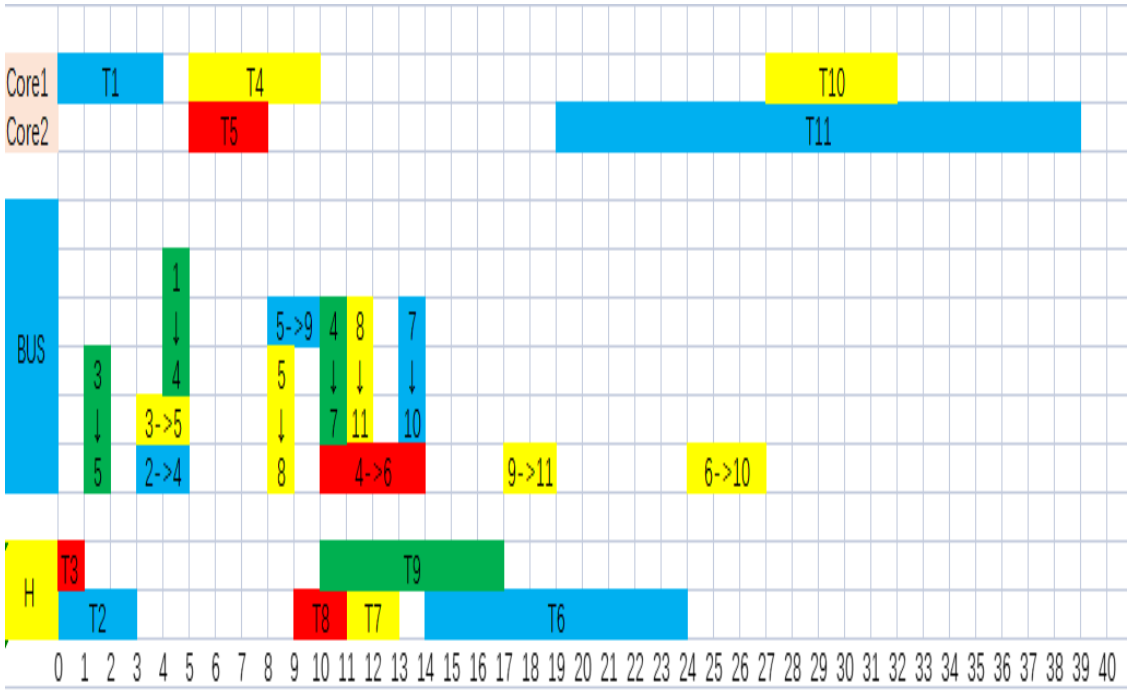


硬件实现增益/硬件面积

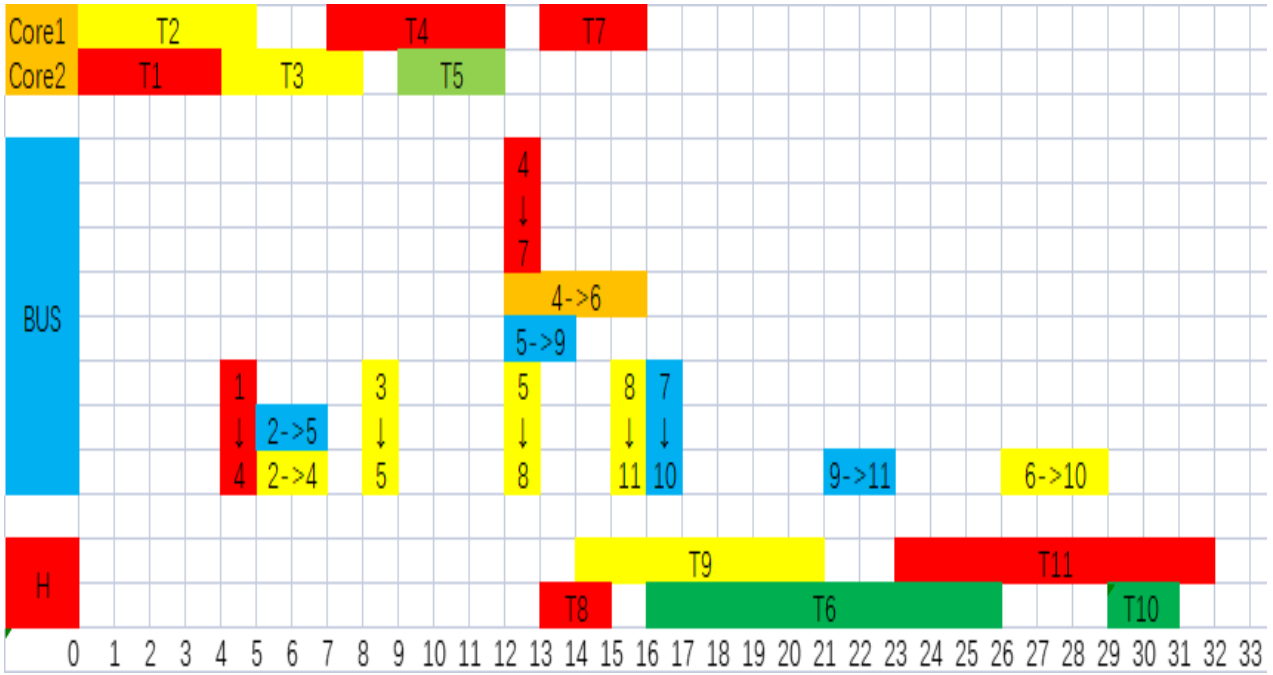
基于硬件增益的三种划分算法结果比较



HSPA$_{HG}$



HSPA$_{HGA}$



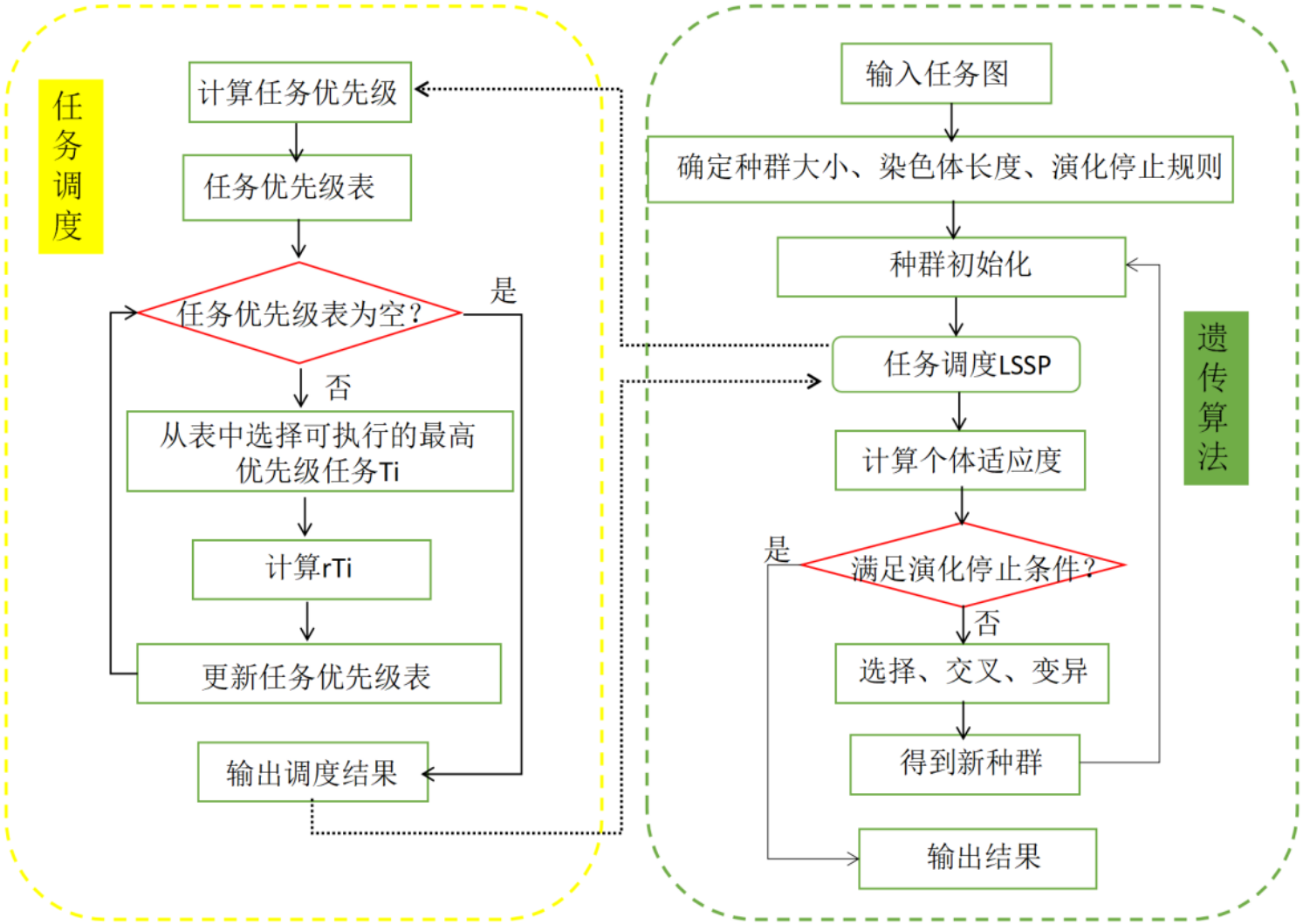
HSPA$_{AHG}$

第8.3节 基于遗传算法的微系统划分

- 本节介绍文献[40]的微系统划分算法$HSPA_{GL}$，该方法将遗传算法GA（Genetic Algorithm）与任务表调度算法LSSP（List Scheduling with Static Priorities）结合在一起，在确保满足多核片上系统硬件面积约束的前提下，能够同时给出软硬件划分结果与任务调度序列，并有效地缩短了系统总体运行时间。
- 本节介绍的方法与第8.2节还有不同之处是：BUS线上不能同时进行两次数据传输，即只能有一个任务在BUS上进行数据传输，同时同一个处理器数据传输不在BUS线上进行，因而安排任务是尽可能地安排在同一的处理器上，即同核上。

8.3.1 遗传算法与划分架构

- 基于遗传算法的软硬件划分$HSPA_{GL}$ 算法结合了遗传算法与任务调度算法，以遗传算法为主体，嵌入基于静态优先级的表调度算法，算法流程图如图8-11所示。



8.3.5 遗传微系统划分算法HSPA$_{GL}$ 示例

对8-6所示任务图进行划分与调度，设定处理器内核数P=2，硬件面积约束S_L=18。

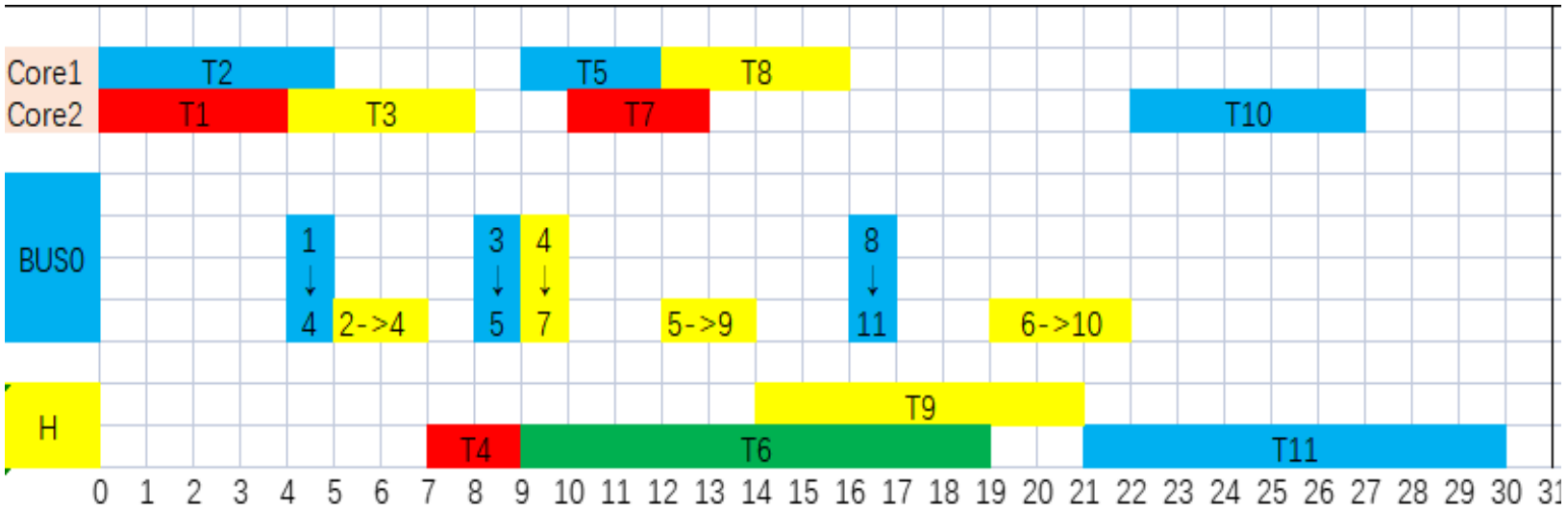
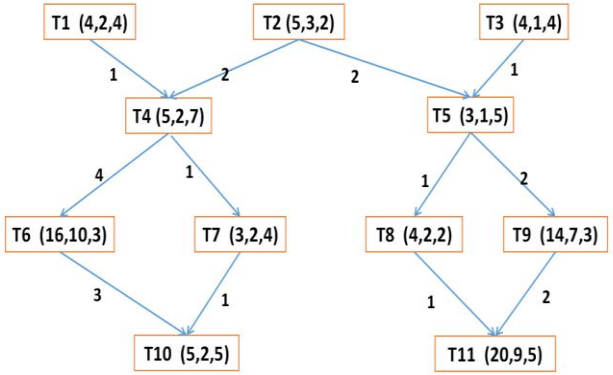


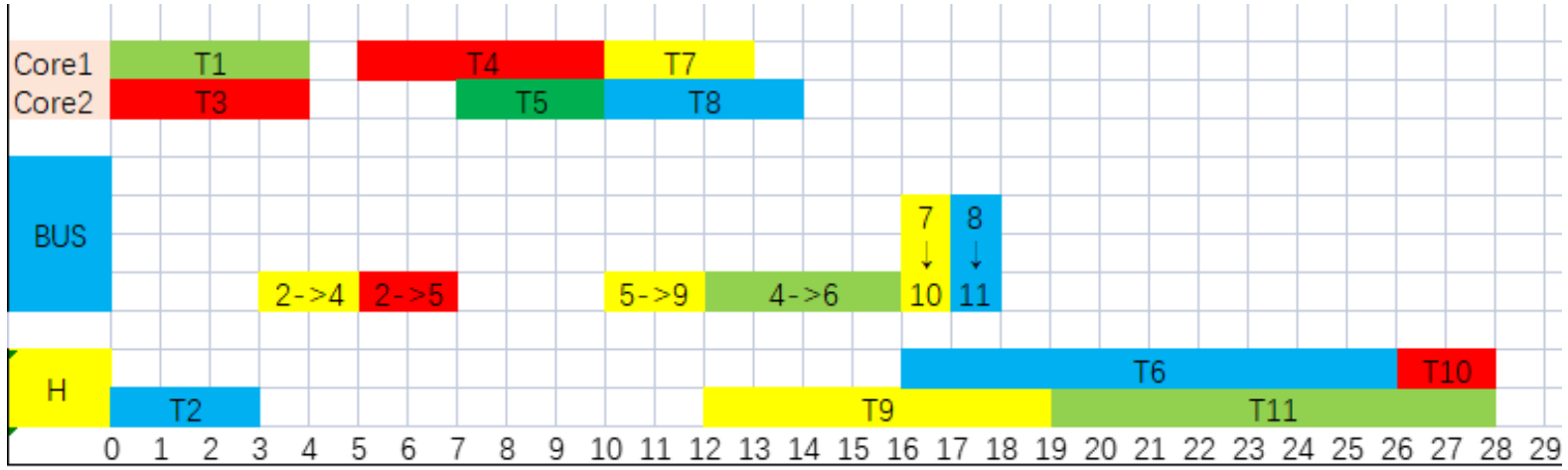
图8-14 个体α调度结果甘特图

初始划分
α=（00010100101）
T2>T1>T3>T̂4>T5>T̂6>T̂9>T8>T7>T̂11>T10。

$$Fitness = \frac{1}{0.7 \cdot e^{\frac{18-18}{18}} \cdot \frac{|18-18|}{18} + 0.3 \cdot \frac{30}{83}} = 9.222$$



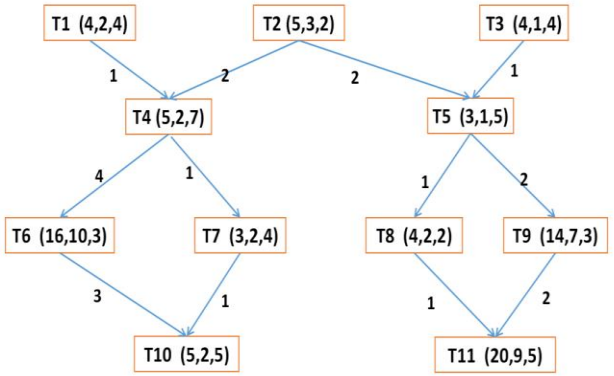
8.3.5 遗传微系统划分算法$HSPA_{GL}$ 示例



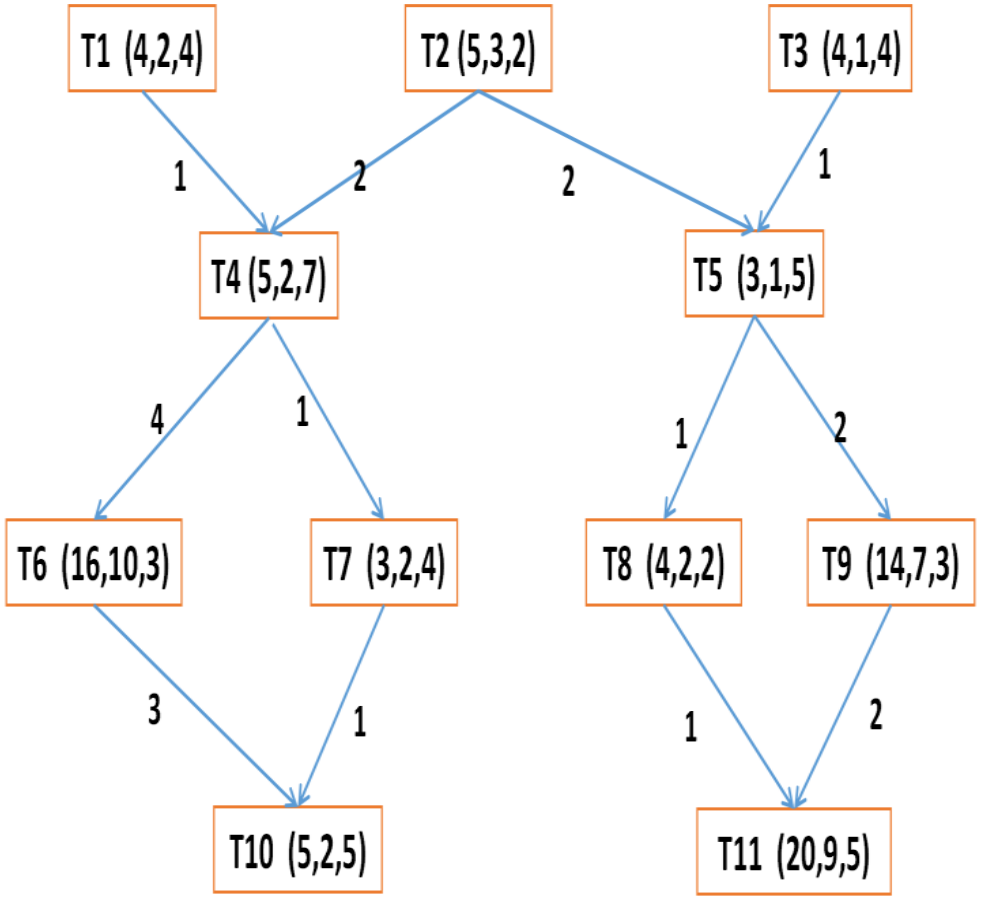
个体β调度结果甘特图

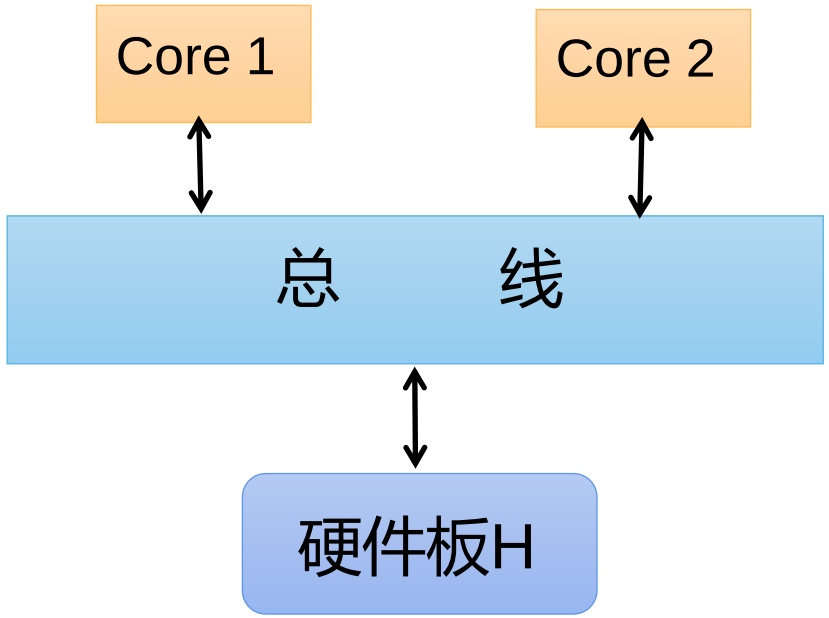
经过遗传算法求得新划分结果
β=01000100111
硬件任务计划
THW={T2,T6,T9,T10,T11},硬件面积之和等于18

$$Fitness = \frac{1}{0.7 \cdot e^{\frac{18-18}{18}} \cdot \frac{|18-18|}{18} + 0.3 \cdot \frac{28}{83}} = 9.8814$$

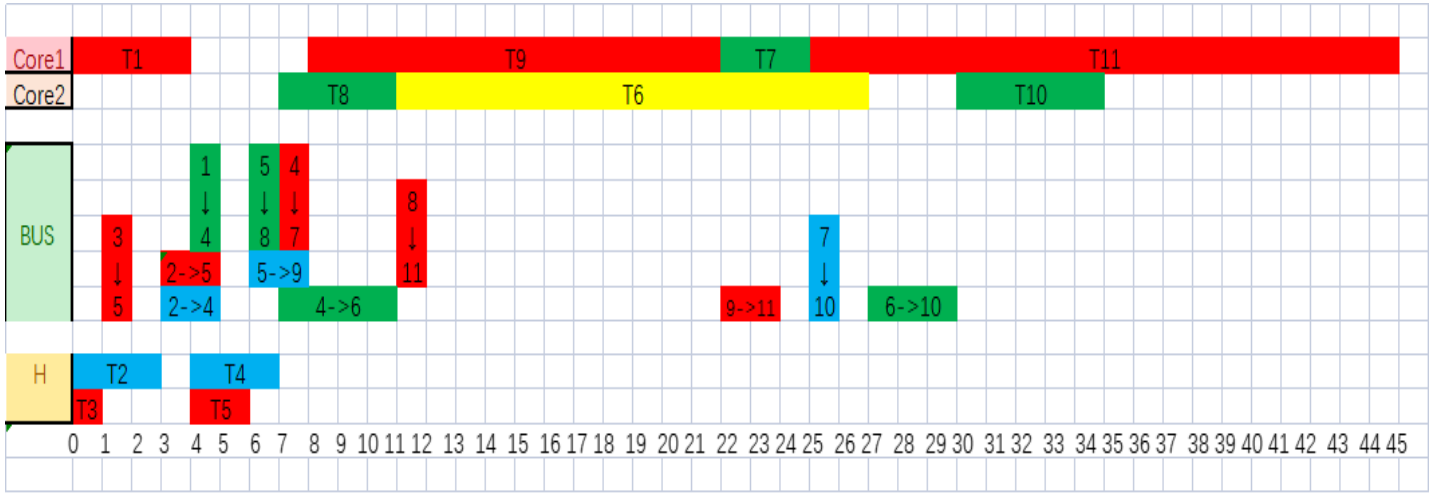


第8.1.1节 基于多核的微系统化

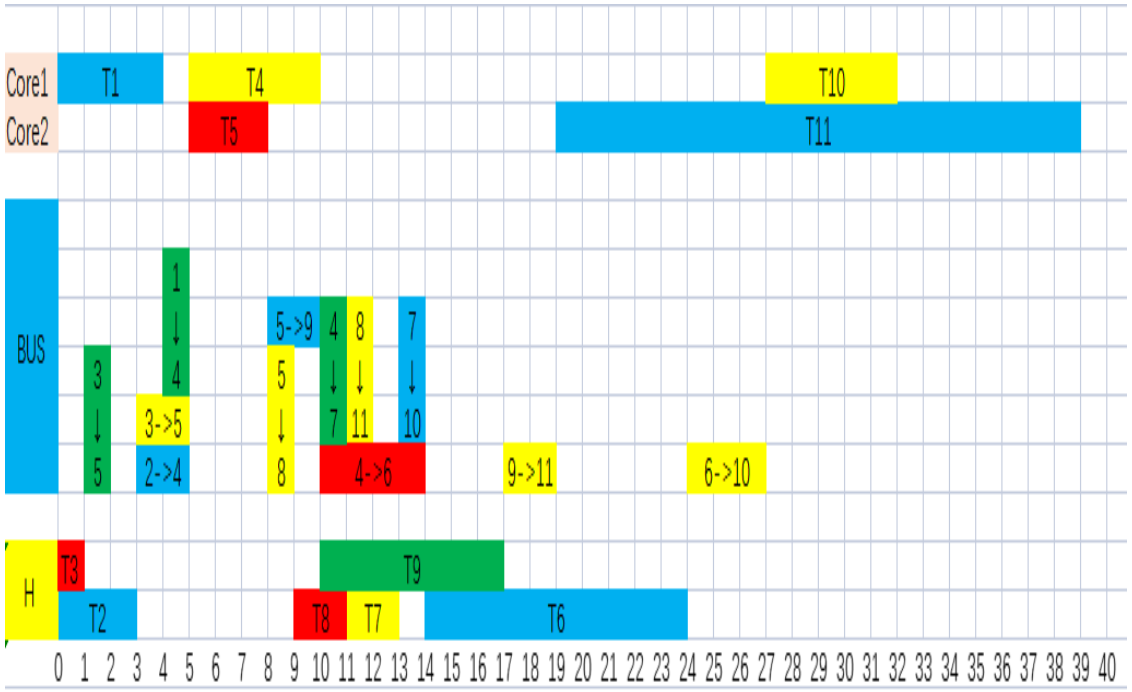




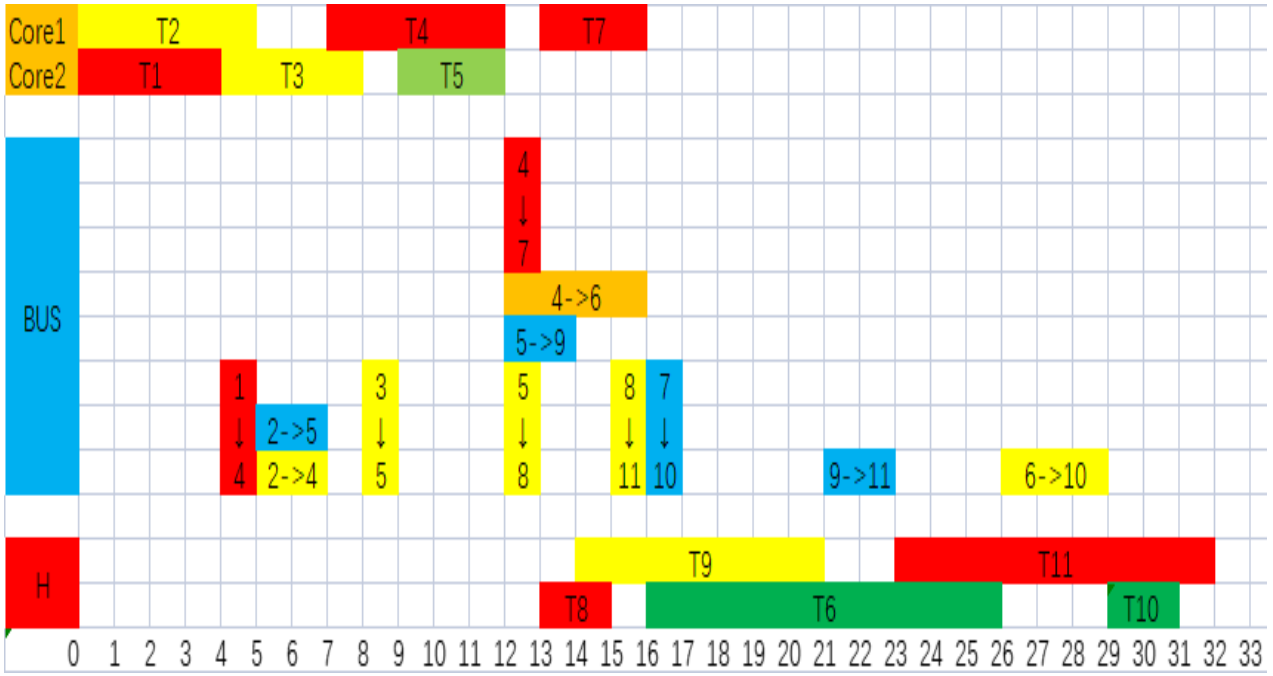
基于硬件实现增益的非遗传算法：三种划分算法结果比较



$HSPA_{HG}$



$HSPA_{HGA}$



$HSPA_{AHG}$

基于硬件实现增益的遗传算法：α和β调度结果甘特图

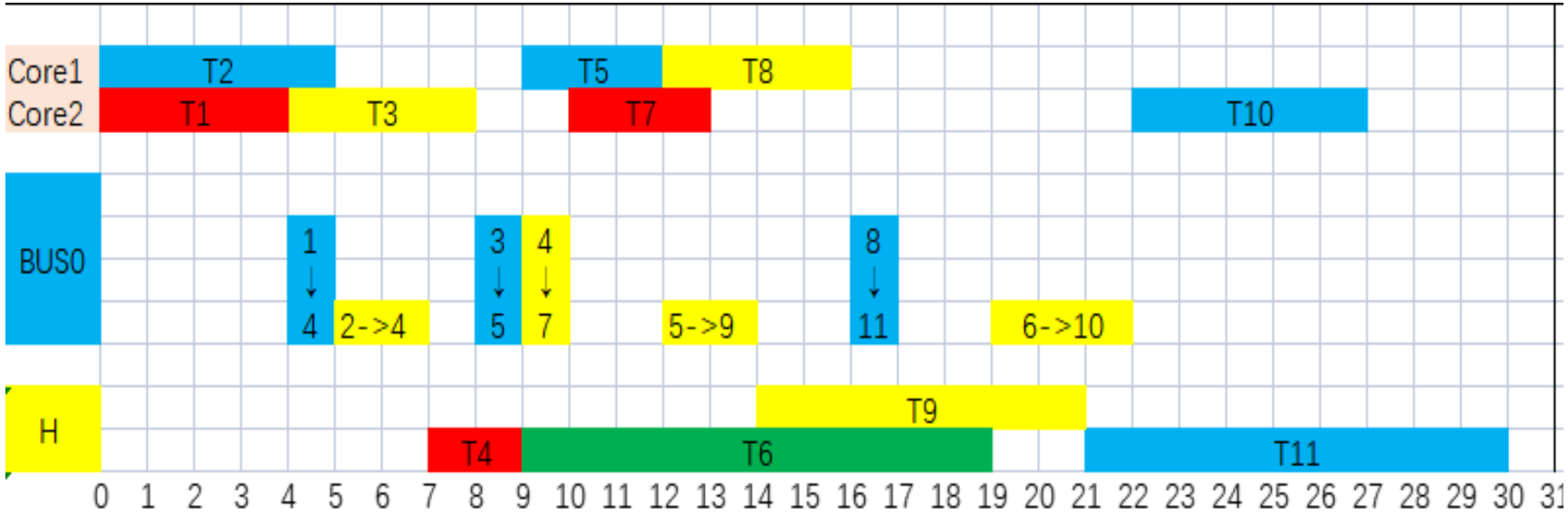


图8-14 个体α=00010100101调度结果甘特图

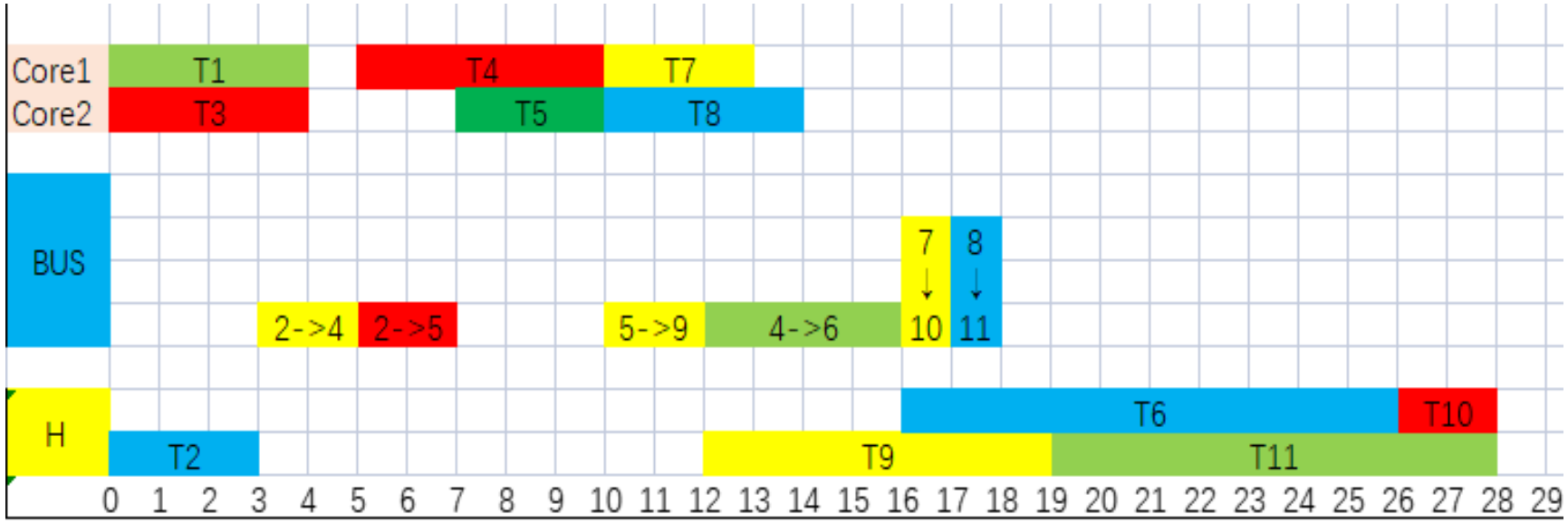


图8-15 个体β=01000100111调度结果甘特图

第8章 微系统划分 总结

RTSS industry session: 43rd Real-time Systems Syposium 2022,December 5 – 8, 2022,Houston, USA

- It is our great pleasure to invite you to participate in the RTSS 2022 Industry challenge. This is an open challenge to researchers in both industry as well as academia, to present novel ideas to tackle a specific problem/domain.
- The challenge, this year, is to understand how high-performance/heterogeneous/GPU-style architectures can be better used in real-time embedded systems. A detailed write up is available here: http://2022.rtss.org/wp-content/uploads/2022/06/rtss_2022_industry_challenge_cfp.pdf

Reviewings:

- The paper addresses an interesting problem
- The paper presents an allocation and a scheduling algorithm for FPGA-accelerated platforms, where a task can be implemented through hardware or software means, respectively on an FPGA or a multi-core complex. The allocation relies in particular on maximising the overall hardware gain, the timing improvements from deploying tasks in hardware of their software counterparts.The paper is a pleasant read as it manages to clearly present its system model and problem,
- This paper presents a hardware implementation gain based approach to scheduling. The paper presents a heuristic strategy based on linear programming that maximizes hardware gain 'makespan', i.e., the worst case task finish time. The paper presents some pseudocode in Algorithm 1 for task allocation phase and scheduling phase in Algorithm 2. Some empirical results are subsequently presented. The paper is readable to a reader from a technical background

大　　纲

无处不在的智能嵌入式系统

智能嵌入式系统软硬件优化配置

基于CNN的交通标识识别系统

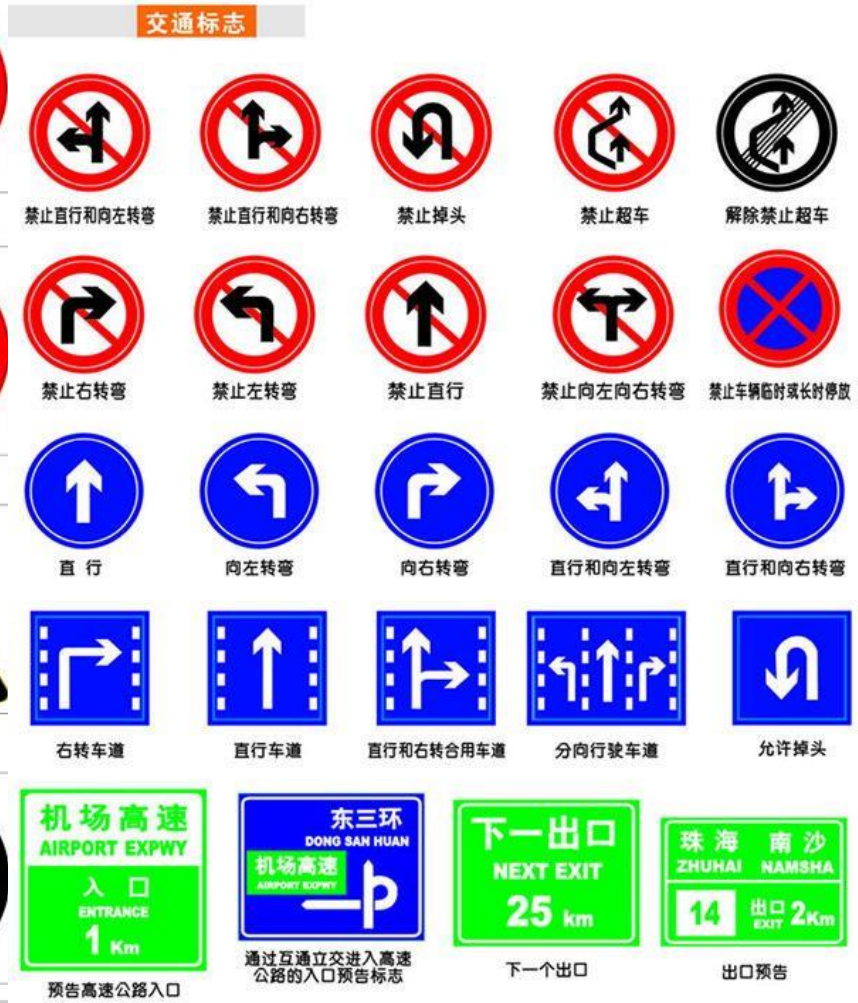
第十章：系统集成

- 止于至善 《礼记·大学》
- 本章重点介绍基于卷积神经网络的交通标识识别系统如何在ARM+FPGA平台的设计实践，实现加速目的。

第10.1节 道路交通标志识别系统

- 道路交通标志是现在交通重要设施，用文字或符号传递引导、限制、警告或指示信息的道路设施，其中道路交通标志分为主标志和辅助标志两大类。
- 主标志又分为警告标志、禁令标志、指示标志、指路标志、旅游区标志和道路施工安全标志六类，共有300多种。警告标志起警告作用，警告车辆、行人注意危险地点的标志。禁令标志起到禁止某种行为的作用，禁止或限制车辆、行人交通行为的标志。指令标志起指示作用，指示车辆、行人行进的标志。指路标志起指路作用，传递道路方向、地点、距离信息的标志。
- 旅游区标志提供旅游景点方向、距离的标志。道路施工安全标志是通告道路施工区通行的标志，用以提醒车辆驾驶人和行人注意。辅助标志是在主标志无法完整表达或指示其内容时，为维护行车安全与交通畅通而设置的标志，附设在主标志下，起辅助说明作用。

第10.1节 道路交通标志识别系统



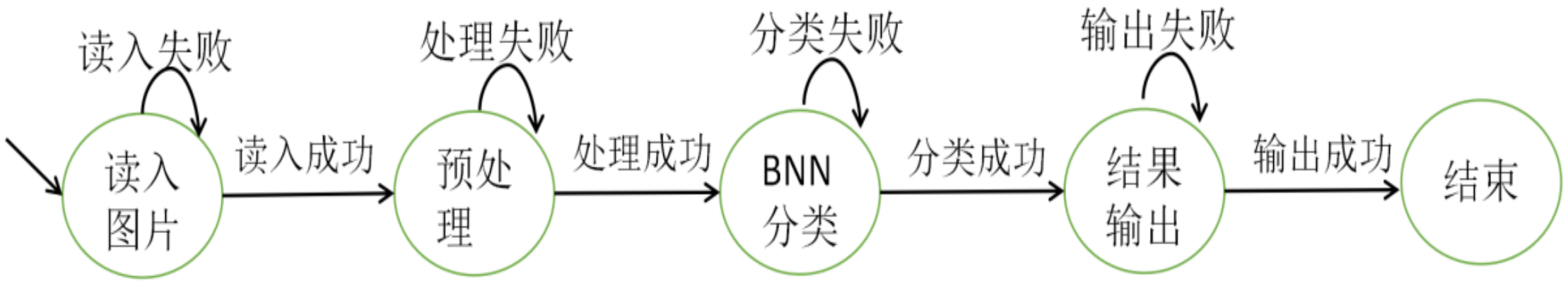
第10.1节 道路交通标志识别系统TSR

- 在无人驾驶或辅助驾驶中， 车辆能自动地及时正确识别道路交通标志是非常重要的。
- 李金洋在PYNQ-Z1开发板上使用ARM和FPGA协同设计实现了一个基于卷积神经网络的道路交通标志自动识别系统TSR，实验表明该自动识别系统识别一张交通标志用时49463微秒，每秒可识别图片20.22张；而完全使用软件实现识别用时812910微秒，每秒可识别图片1.23张。前者识别速度是后者的16倍。
- 本章主要介绍李金洋的工作。

第10.2节 系统建模

10.2.1. 系统概述

- 交通标志识别系统的输入为交通标志图片，经过分类器识别输出该图片所表示的标志含义。



系统采用Cifar-10数据库中的图片格式）。

- CNN分类决策是系统重点部分，负责标志的识别。识别结果输出则将结果返回给用户。

第10.2节 系统建模

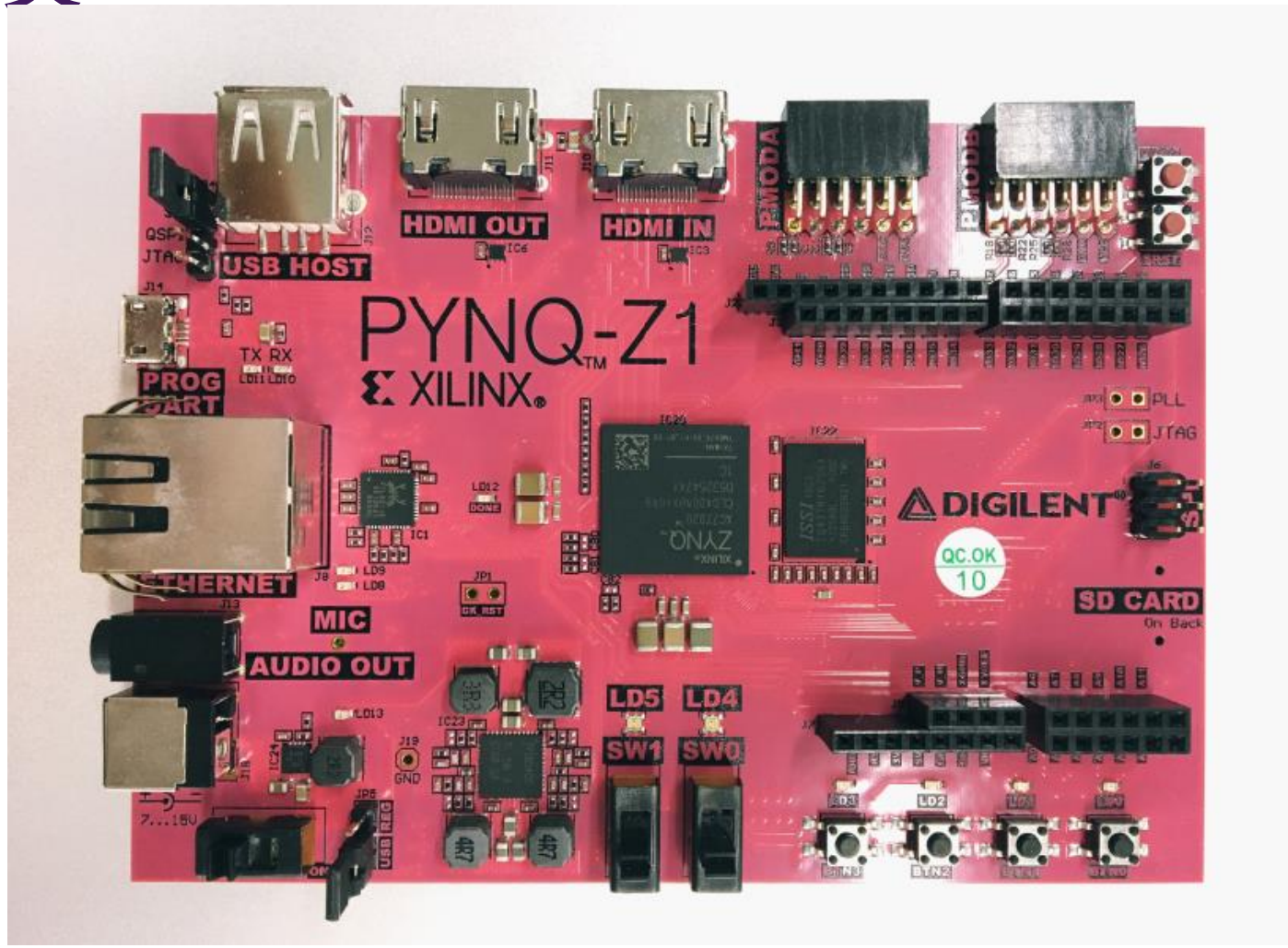
10.2.3. 系统约束

- 当交通标志识别系统被用于辅助驾驶或自动驾驶时，由于车辆在运动中速度较快，需要及时识别标志进行相应操作，对交通标志识别系统的时间性能要求很高。
- 因此，使用FPGA对交通标志识别系统进行硬件加速。

第10.2节 系统建模

10.2.3. 系统约束

- 系统采用PYNQ-Z1开发板进行开发，该开发板上集成了ZYNQ-7020 SoC器件。



第10.2节 系统建模

10.2.3. 系统约束

- 系统采用PYNQ-Z1开发板进行开发，该开发板上集成了ZYNQ-7020 SoC器件，还给用户提供了丰富的硬件外设接口，具体参数如表所示。
- 从表中可以看出该设备中LUT数量为53200个，因此系统要求硬件部分所使用的LUT数量不超过53200个。

参数	配置
尺寸	87mm x 122mm
处理器	双核ARM Cortex A9
FPGA	53200个LUT 130万个可重配置门电路
内存	512M DDR3/FLASH
存储	支持Micro SD卡
接口	HDMI输入输出接口、音频输入输出接口、 千兆以太网接口、USBOTG接口、 Arduino和Pmod等
其他	LEDx6、按键x4、开关x2

第10.3节 任务属性-10.3.1. 任务提取

系统中图片输入、预处理和识别结果输出这三个过程操作十分简单，使用软件就可以很容易实现，

因此只需考虑将较为复杂、耗时较长的BNN分类决策部分采用硬件加速。

首先将BNN分类器整体使用硬件实现，对C++代码使用Vivado HLS软件进行综合。BNN分类器整体的综合结果如图所示。

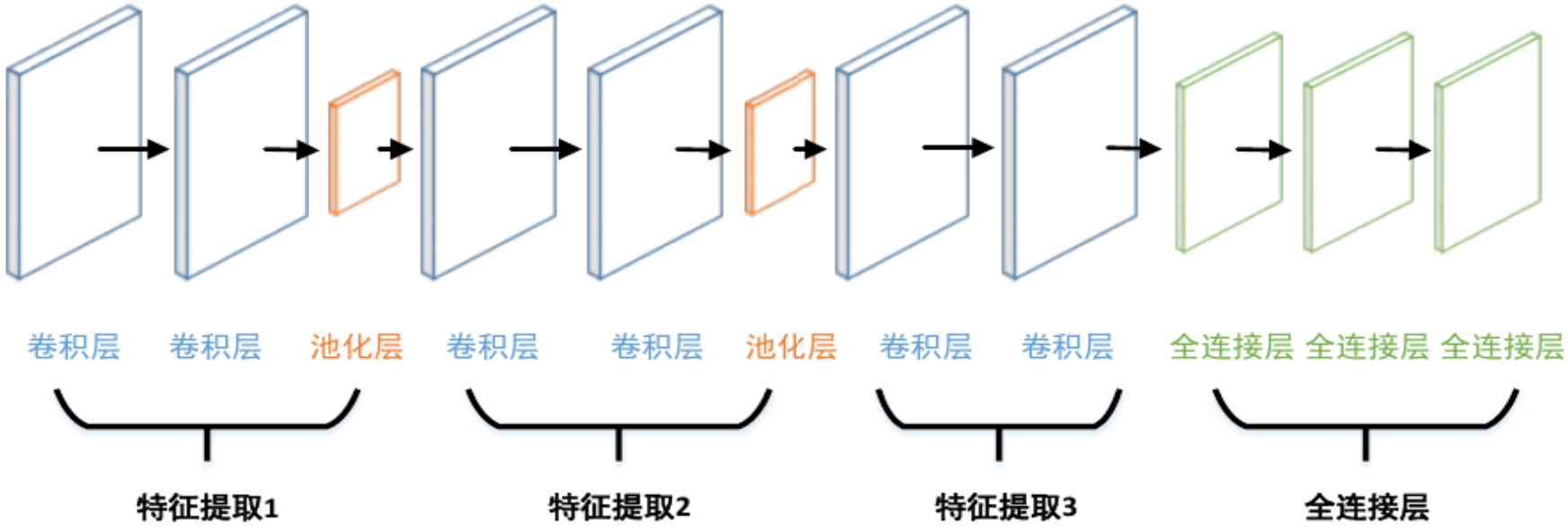
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	2
FIFO	-	-	-	-
Instance	43	52	69569	65232
Memory	189	-	3104	416
Multiplexer	-	-	-	6939
Register	-	-	319	-
Total	232	52	72992	72589
Available	280	220	106400	53200
Utilization (%)	82	23	68	136

第10.3节 任务属性-10.3.1. 任务提取

- 系统所用BNN层次结构如图所示，包括6个卷积层、2个池化层和3个全连接层。
- 其中第1、2、4、5、7、8层为卷积层，用来进行特征提取；
- 第3、6层池化层对输入的特征图进行压缩，一方面使特征图变小，简化网络计算复杂度，另一方面进行特征压缩，提取主要特征；
- 因此卷积层和池化层相连即可完成一次主要特征提取。



第10.3节 任务属性-10.3.2. 任务性能获取

任务软件执行时间属性获取方式为：

◆将任务使用C++实现，并在任务执行前后加时间戳，运行代码，将相应时间戳相减得到该任务的一次软件执行时间，多次运行结果取平均值得到最终任务的软件执行时间。

◆运行环境对软件的执行时间有着很大影响，考虑到最终系统集成后软件部分在ARM核中运行，为了保证数据准确有效并具有参考价值，此处的运行环境与系统集成后环境一致，即任务的软件执行时间为该任务在ARM核中运行所需时间。

◆经执行计算得到四个任务所需软件执行时间分别为：T1：534,910 μs，T2：240,385 μs，T3：32,629 μs，T4：9,467 μs。

第10.3节 任务属性-10.3.2. 任务性能获取

任务硬件执行时间属性获取方式为：

- 将任务在Vivado HLS中对C++代码进行综合，从综合结果报告中可得到每一部分执行所用时钟周期数和每个时钟周期所用时间，将其相乘即可估算出任务的硬件执行时间。
- 通过估算得到四个任务所需硬件执行时间分别为：
  - ✓T1：7,293μs，T2：4,018μs，
  - ✓T3：514μs，T4：377μs。

第10.3节 任务属性-10.3.2. 任务性能获取

基于BNN的交通标志识别系统任务属性表如表所示

任务	软件执行时间	硬件执行时间	硬件面积	备注
T_1	534,910 μs	7,293 μs	24,956个LUT	特征提取1
T_2	240,385 μs	4,018 μs	21,566个LUT	特征提取2
T_3	32,629 μs	514 μs	9,045个LUT	特征提取3
T_4	9,467 μs	377 μs	4,639个LUT	全连接层

第10.3节 任务属性-10.3.3. 软硬件划分

对BNN分类器中的任务进行软硬件划分。

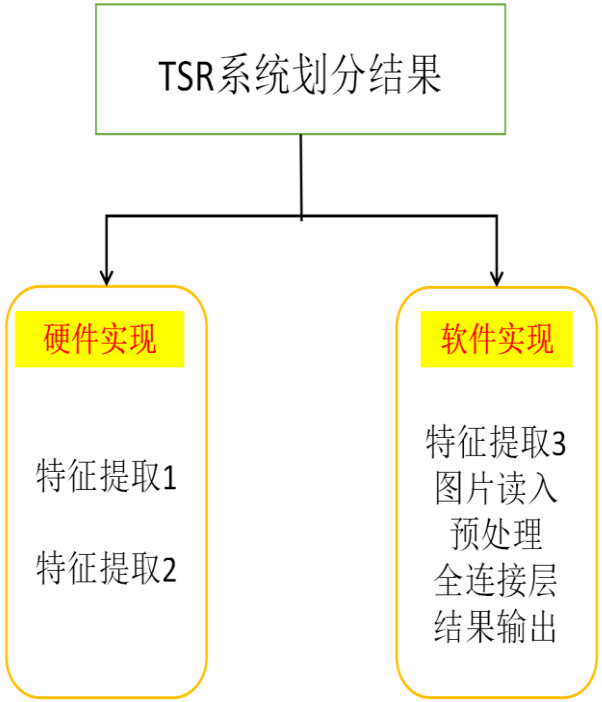
◆预留6000个LUT用于存储、复用器及总线控制，则剩余的47200个LUT为系统硬件面积约束。

◆使用第5章多目标划分方法对BNN分类器的四个任务进行软硬件划分：

- ➢将时间作为总体极小化目标，约束条件是LUT数量不超过47200个。
- ➢划分结果为硬件实现T1和T2，软件实现为T3和T4，LUT数量使用了46522个，时间为53407μs。

◆BNN分类器中特征提取1和特征提取2需要使用硬件进行加速，特征提取3和全连接层则由软件实现。

◆再加上图片读入和预处理等任务，最终完整的系统软硬件划分结果如图所示。

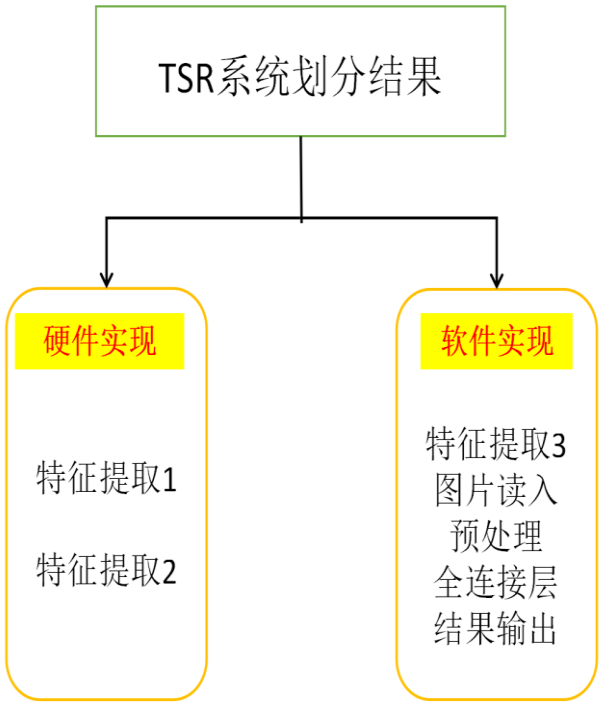


第10.4节 软硬件综合-10.4.1. 软件综合

软件综合需选取合适的语言对软件任务进行实现。

◆对于特征提取3和全连接层这两个任务，由于在之前获取任务属性时该部分已经使用C++代码实现并进行过测试，为了提高代码复用性，减少工作量，此部分的软件实现沿用C++实现。

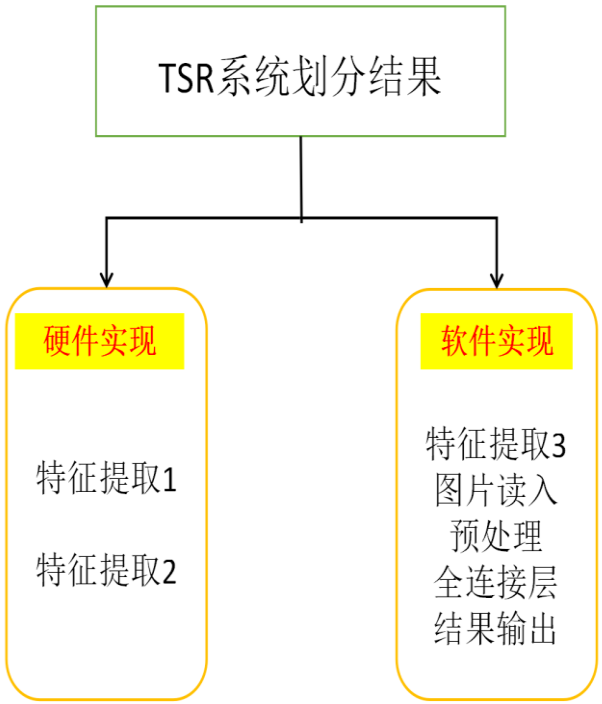
◆考虑其他三个软件任务即图片读入、预处理、识别结果输出的实现。Python中的Python Image Library有着强大的图片处理能力，可以满足图片读入和预处理的需求，并且Python可以直接调用C++语言，再考虑到系统实现所选取的PYNQ-Z1开发板顶层支持Python语言，因此Python是实现这三个任务的最佳语言。



第10.4节 软硬件综合-10.4.2. 硬件综合

用硬件实现特征提取1和特征提取2这两个任务：

- 需要创建相应IP核，这一过程仍然在Vivado HLS软件中进行。
- 为了创建我们所需要的IP核，需要在Vivado HLS中创建一个工程，导入实现特征提取1和特征提取2功能的C++代码并指定综合时的顶层函数，此处仍然复用之前综合时使用的代码，然后导入为该部分重新设计的测试文件。
- 选择开发板型号为“xc7z020clg400-1”（与目标开发板对应）。



第10.4节 软硬件综合-10.4.2. 硬件综合

创建工程之后利用测试文件进行C++仿真，测得仿真结果与预期相符。

- 之后进行C综合，同样可以得到资源利用情况，此部分综合结果如图所示，此时LUT使用情况已满足数量限制，且BRAM、DSP、FF等资源的使用数量均有所减少。
- 综合过程中还会自动生成RTL设计即相应的VHDL和Verilog代码；综合完成后可进行C/RTL协同仿真，验证设计的功能正确性；
- 验证完成之后，即可将RTL封装成可重用IP核导出，随后可导入到Vivado IP Catalog中用于之后的实现。

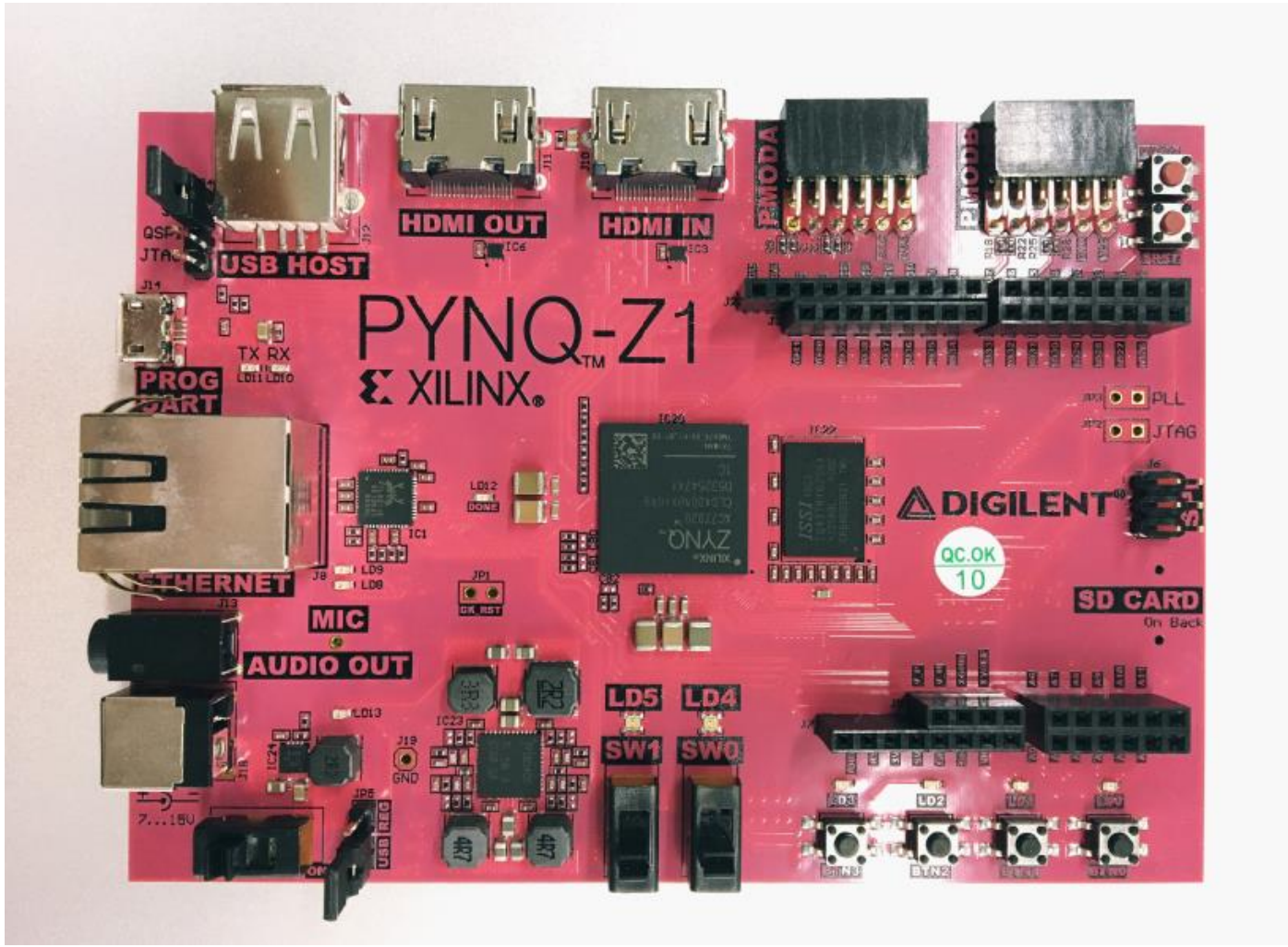
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	8
FIFO	-	-	-	-
Instance	30	24	24881	46151
Memory	32	-	1888	96
Multiplexer	-	-	-	3813
Register	-	-	319	-
Total	62	24	27088	50068
Available	280	220	106400	53200
Utilization (%)	22	10	25	94

第10.5节 系统实现-10.5.1. 实现环境

- 系统使用PYNQ-Z1开发板进行开发，图为其实物图。
- PYNQ-Z1由Digilent公司推出，基于Xilinx ZYNQ-7020，并在其基础上添加了对Python的支持。
- ZYNQ-7020是Xilinx公司推出的一款可扩展处理芯片，集成了双核ARM处理器和FPGA可编程逻辑器件，
- 旨在为视频监视、汽车驾驶员辅助以及工厂自动化等高端嵌入式应用提供所需的处理与计算性能水平。

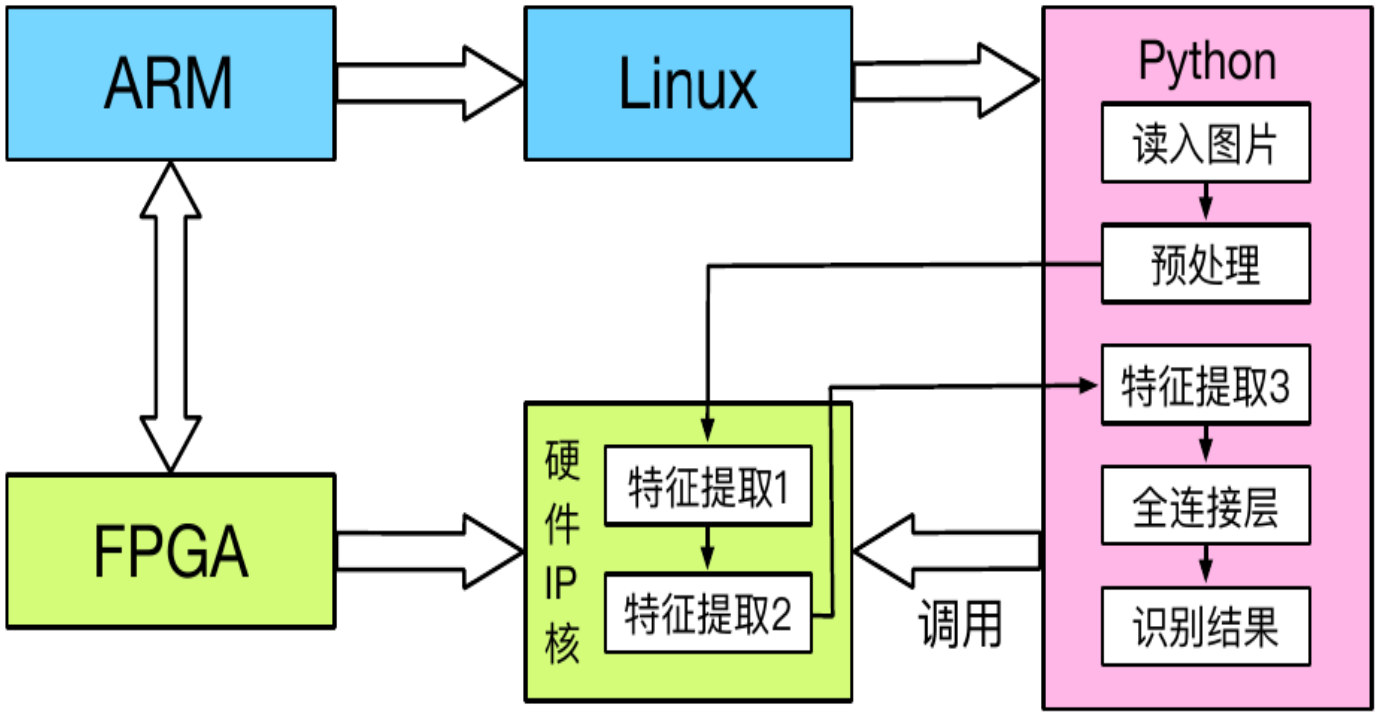


第10.5节 系统实现-10.5.2. 实现过程

在软、硬件综合过程中已经实现了每一个任务的功能，

但各个任务没有联系起来，系统功能还不能完整实现，

在最后实现阶段要将每个任务模块集成起来，整个系统实现架构图如图所示。



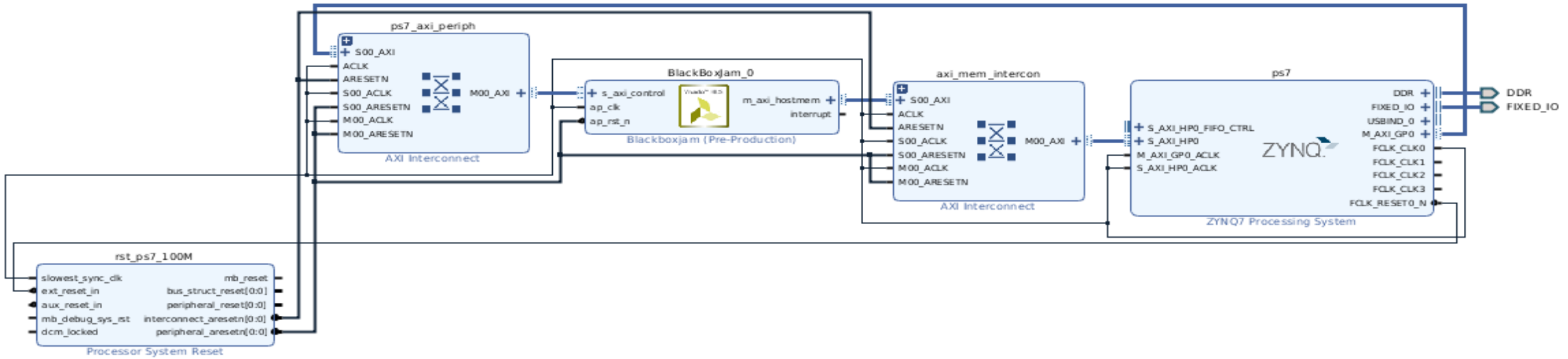
第10.5节 系统实现-10.5.2. 实现过程

软件实现部分：

对于已经用C++实现的代码，需先将其编译生成共享对象库（.so文件），Python便可以使用CFFI（Foreign Function Interface for Python）直接进行调用。

硬件实现部分：

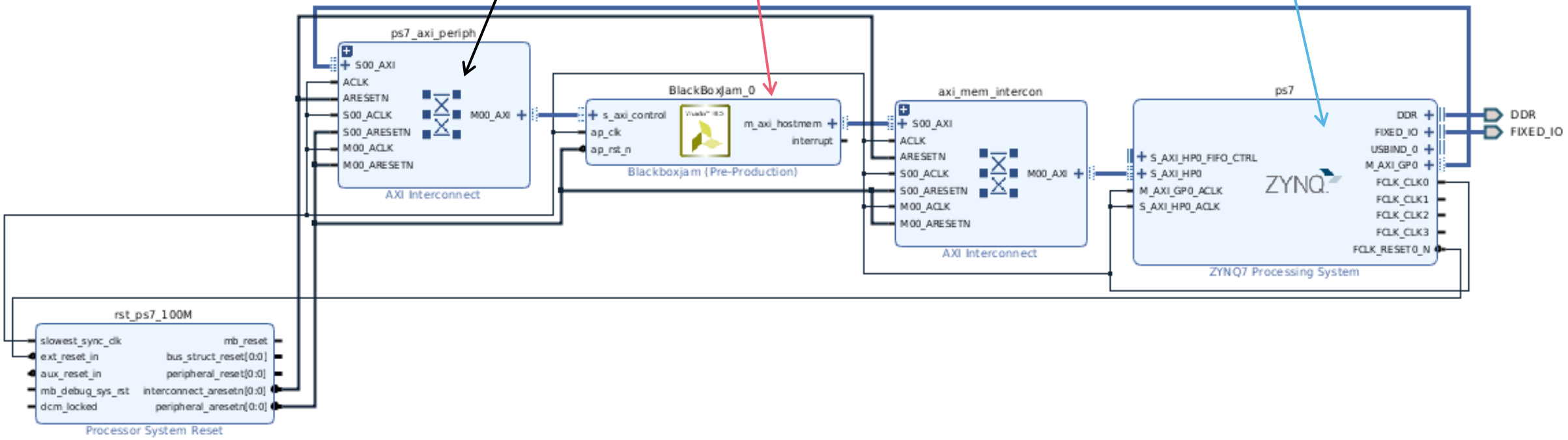
首先要在Vivado集成设计环境中选择开发板创建新的工程，加入硬件部分所需IP核，在此过程中将之前在Vivado HLS中生成的IP核导入，IP核的连接情况如图所示。



第10.5节 系统实现-10.5.2. 实现过程

图中中间的IP核是硬件综合后生成的IP核，最右边的为ZYNQ7 Processing System IP核，其余IP则用于AXI总线控制、内存访问和复位，这些IP核均由Xilinx提供。

IP核连接好之后可进行综合和布局布线，完成后生成二进制比特流文件。将其拷贝至开发板即可使用Python用PYNQ提供的Overlay类进行加载。



第10.5节 系统实现-10.5.3. 实现结果

本小节对系统实现结果进行验证。

- 首先正确连接计算机和PYNQ-Z1开发板，随后在浏览器中通过开发板访问Jupyter Notebook。
- Jupyter Notebook是一个在线交互式计算环境，对Python有着很好的支持。在Jupyter中编辑ipynb文件调用我们写好的系统实现并运行就可以即时看到结果输出。

- 以一张含有停止标志的图片为例，将其输入到交通标志识别系统中，对系统功能进行验证并计算执行时间。

第10.5节 系统实现-10.5.3. 实现结果

验证时的Jupyter Notebook界面如 所示，对于输入的图片，所得到的识别结果为“Stop”，说明系统实现正确。
整个识别过程用时49,463微秒，每秒可识别图片20.22张。

jupyter Traffic Sign Recognition (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

Test TSR with hardware

In [5]:
```
results = classifier_hw.classify(images)
time = results[0]
for i in range(1, len(results))
    print("Identified sign: {0}".format((classifier_hw.class_name(results[i]))))
print("Identification took %.2f microseconds, %.2f usec per image" % (time,time/len(images)))
print("TSR can identify %.2f images per second with hardware" % (1000000.0/(time/len(images))))
```

```
Identified sign: Stop
Identification took 49463.00 microseconds, 49463.00 usec per image
TSR can identify 20.22 images per second with hardware
```

第10.5节 系统实现-10.5.3. 实现结果

为了说明软硬件协同设计对系统的加速情况，我们进行了对比实验.
所有任务都由软件实现，在ARM核上运行。
实验结果如图所示。可以看到系统完全使用软件实现识别用时812,910微秒，
每秒可识别图片1.23张。

Test TSR with software

```
results = classifier_sw.classify(images)
time = results[0]
for i in range(1, len(results))
    print("Identified sign: {0}".format((classifier_sw.class_name(results[i]))))
print("Identification took %.2f microseconds, %.2f usec per image" % (time,time/len(images)))
print("TSR can identify %.2f images per second with software" % (1000000.0/(time/len(images))))
```

```
Identified sign: Stop
Identification took 812910.00 microseconds, 812910.00 usec per image
TSR can identify 1.23 images per second with software
```

第10.5节 系统实现-10.5.3. 实现结果

再以“注意前方儿童”、“野生动物出没”、“50KM/h”几个标志作为输入进行了对比实验，结果如图所示。

Test TSR with hardware

```
results = classifier_hw.classify(images)
time = results[0]
for i in range(1, len(results))
    print("Identified sign: {0}".format((classifier_hw.class_name(results[i]))))
print("Identification took %.2f microseconds, %.2f usec per image" % (time,time/len(images)))
print("TSR can identify %.2f images per second with hardware" % (1000000.0/(time/len(images))))
```

```
Identified sign: Children crossing ahead
Identified sign: Wild animals
Identified sign: 50 Km/h
Identification took 142478.00 microseconds, 47492.67 usec per image
TSR can identify 21.06 images per second with hardware
```

Test TSR with software

```
results = classifier_sw.classify(images)
time = results[0]
for i in range(1, len(results))
    print("Identified sign: {0}".format((classifier_sw.class_name(results[i]))))
print("Identification took %.2f microseconds, %.2f usec per image" % (time,time/len(images)))
print("TSR can identify %.2f images per second with software" % (1000000.0/(time/len(images))))
```

```
Identified sign: Children crossing ahead
Identified sign: Wild animals
Identified sign: 50 Km/h
Identification took 2459037.00 microseconds, 819679.00 usec per image
TSR can identify 1.22 images per second with software
```

第10.5节 系统实现-10.5.3. 实现结果

实验结果如表所示，采取软硬件协同设计方法对基于BNN的交通标志识别系统进行开发、ARM和FPAG协同工作时，相比于系统完全由软件实现，识别速度提升了约17倍。

实现方式	识别单张图片用时	识别3张图片用时	平均每秒识别图片
ARM	812910μs	245037μs	1.22张
ARM+FPGA	49463μs	142478μs	20.84张

大　　纲

无处不在的智能嵌入式系统

智能嵌入式系统软硬件优化配置

基于CNN的交通标识识别系统

谢 谢

智能嵌入式系统设计

陈仪香、陈彦辉编著

机械工业出版社，2023年7月

由华东师范大学陈仪香与西安电子科技大学陈彦辉联合编写的《智能嵌入式系统设计》由机械工业出版社出版了，网上书店有售。

这本书包含了基础篇（建模与仿真）、核心篇（软硬件划分）、实践篇（基于卷积神经网络的交通标识识别系统设计与实现）。

该书撰写虽然经历近10年，也多次给研究生和本科生讲授，但错误在所难免，请各位专家批评指正。