2018嵌入式系统的人工智能研究与思考学术研讨会

智能系统与嵌入技术

陈仪香

2018年5月25日

華東師範大學
EAST CHINA NORMAL UNIVERSITY
软硬件协同设计技术与应用教育部工程研究中心
Hardware/software Co-Design Technology and Application Engineering Research Center, MOE

大 纲

- 智能系统
- 规范建模
- 协同优化设计
- 交通标志识别系统
- 自动停车系统

大 纲

智能系统

规范建模

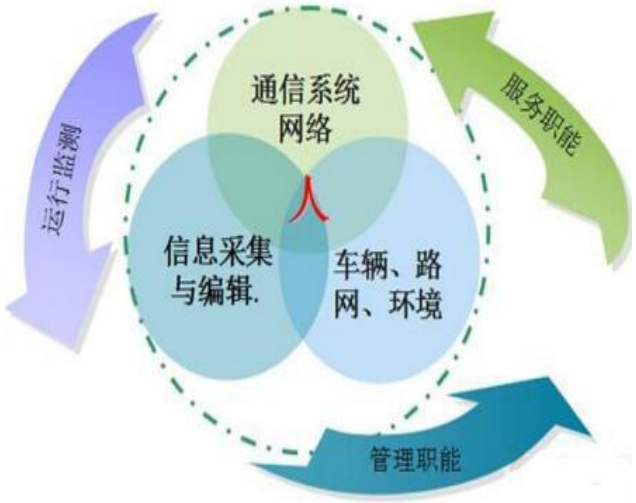
协同优化设计

交通标志识别系统

自动停车系统

智能系统

- 智能系统：在硬件基础上融入人工智能，让机器通过一定的方式进行判断、决策和控制，以便最有效地发挥作用。
- 它是一个具有传感、控制、人机交互、网络接入的实体系统。
- 信息物理融合系统CPS。
- 它也是嵌入式技术、软件工程、人工智能等知识的深入融合系统。



智能系统

■智能系统：无处不在。

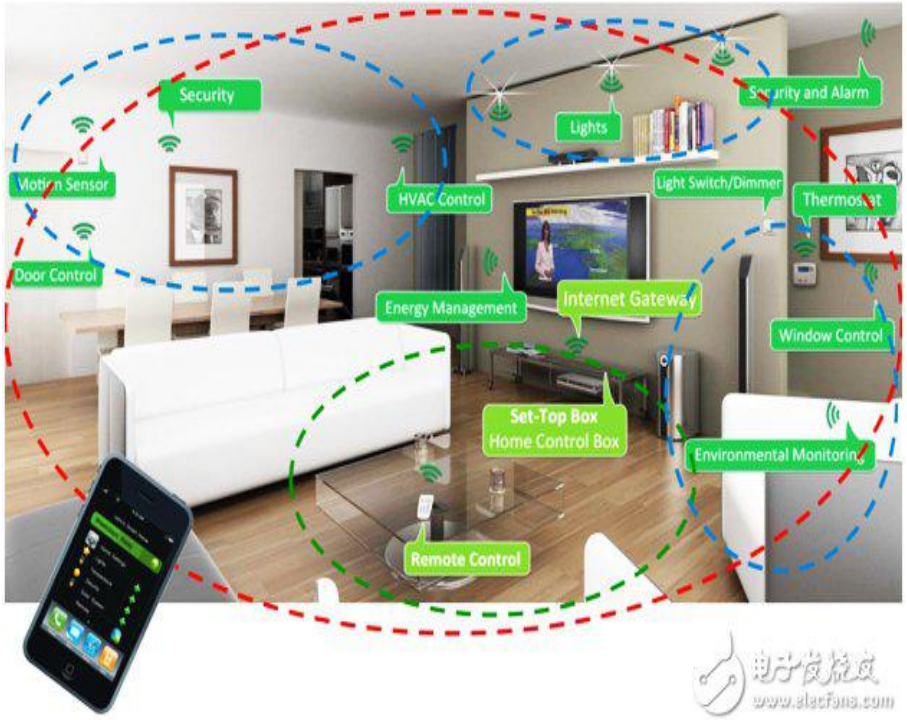
日常生活用品

可穿戴产品

智能系统

■智能系统：无处不在。

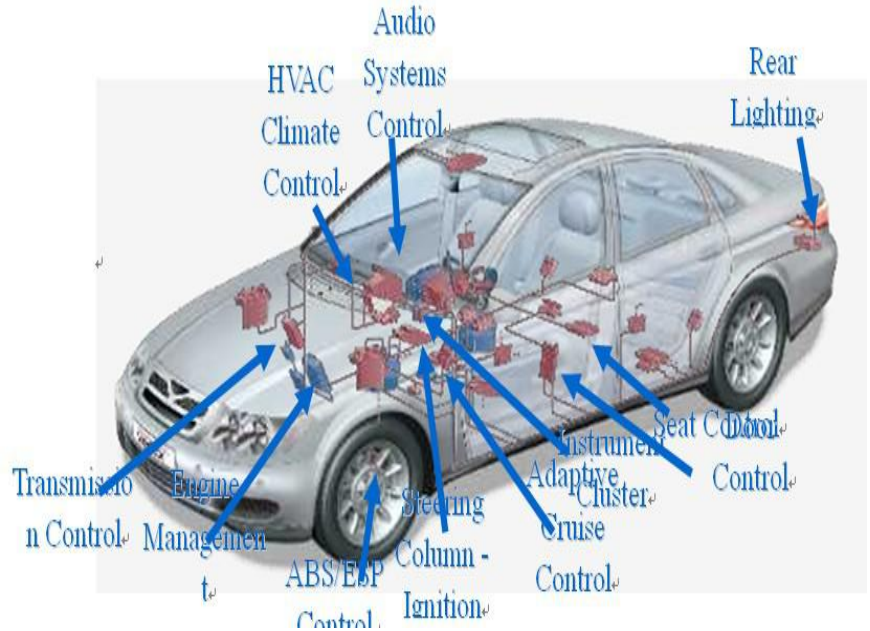
医疗器材



智能家居

智能系统

■智能系统：无处不在。



无人驾驶汽车

高铁工程

智能系统

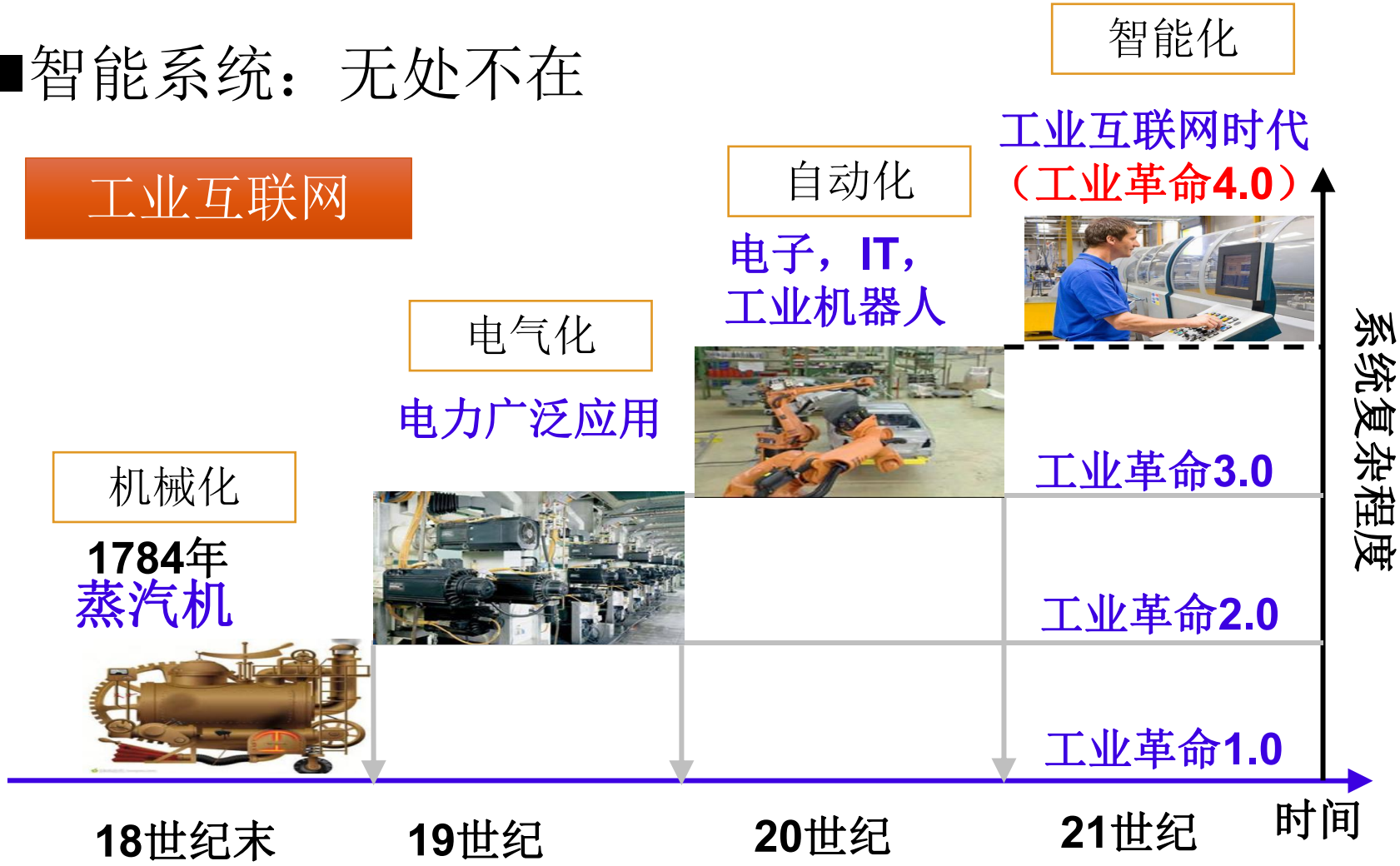
■智能系统：无处不在。

探月工程

飞机系列

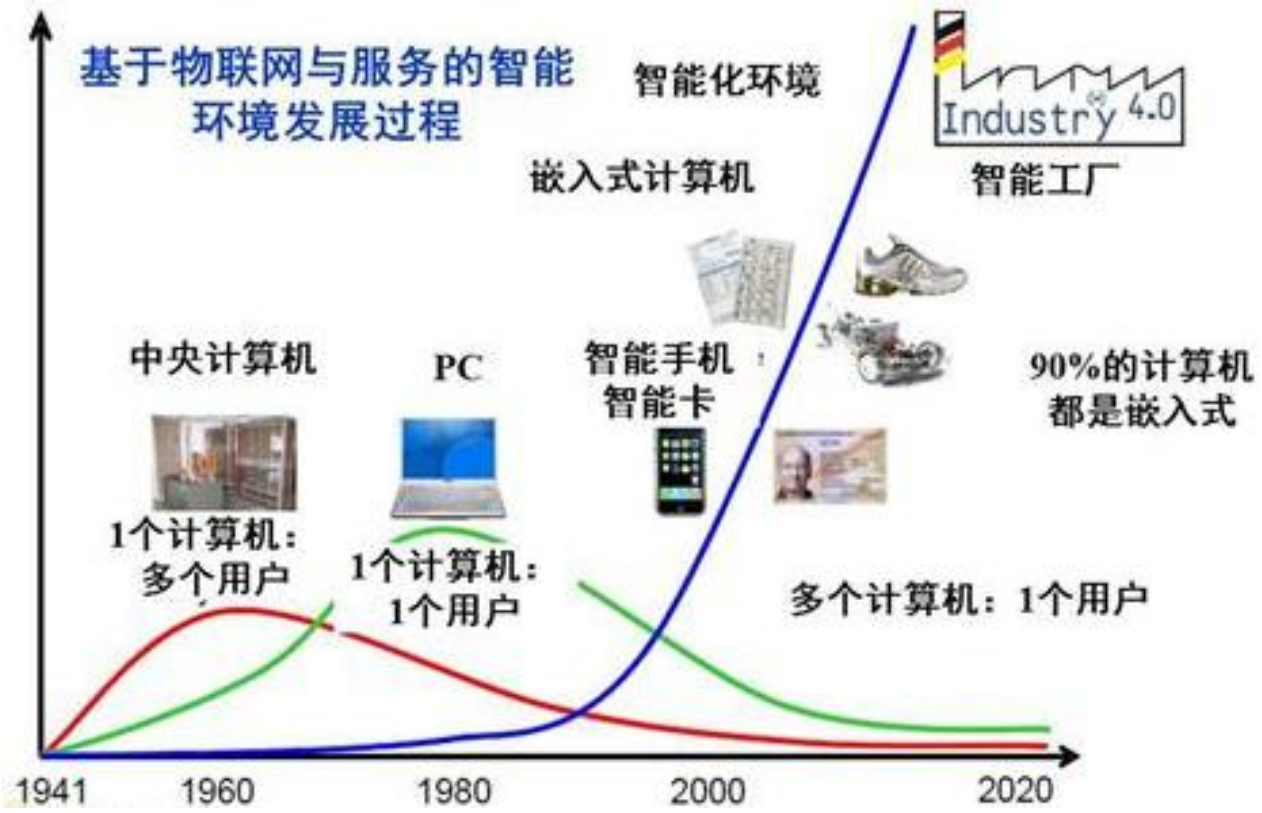
智能系统

■智能系统：无处不在



智能系统

■智能系统：无处不在



中国制造2025

智能系统

■智能系统：无处不在

- 经济转型期两化融合的需求
  - 经济转型，创新驱动
  - 新能源汽车、智能电网、智慧城市等行业发展
- 自主可控安全系统制造的需求
  - 制造业大国向制造业强国转变
  - 大飞机、核电、航天、轨道交通、……

中国制造2025

大 纲

智能系统

规范建模

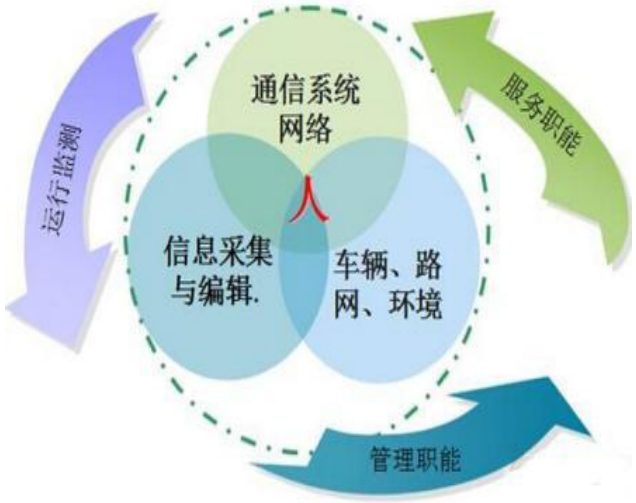
协同优化设计

交通标志识别系统

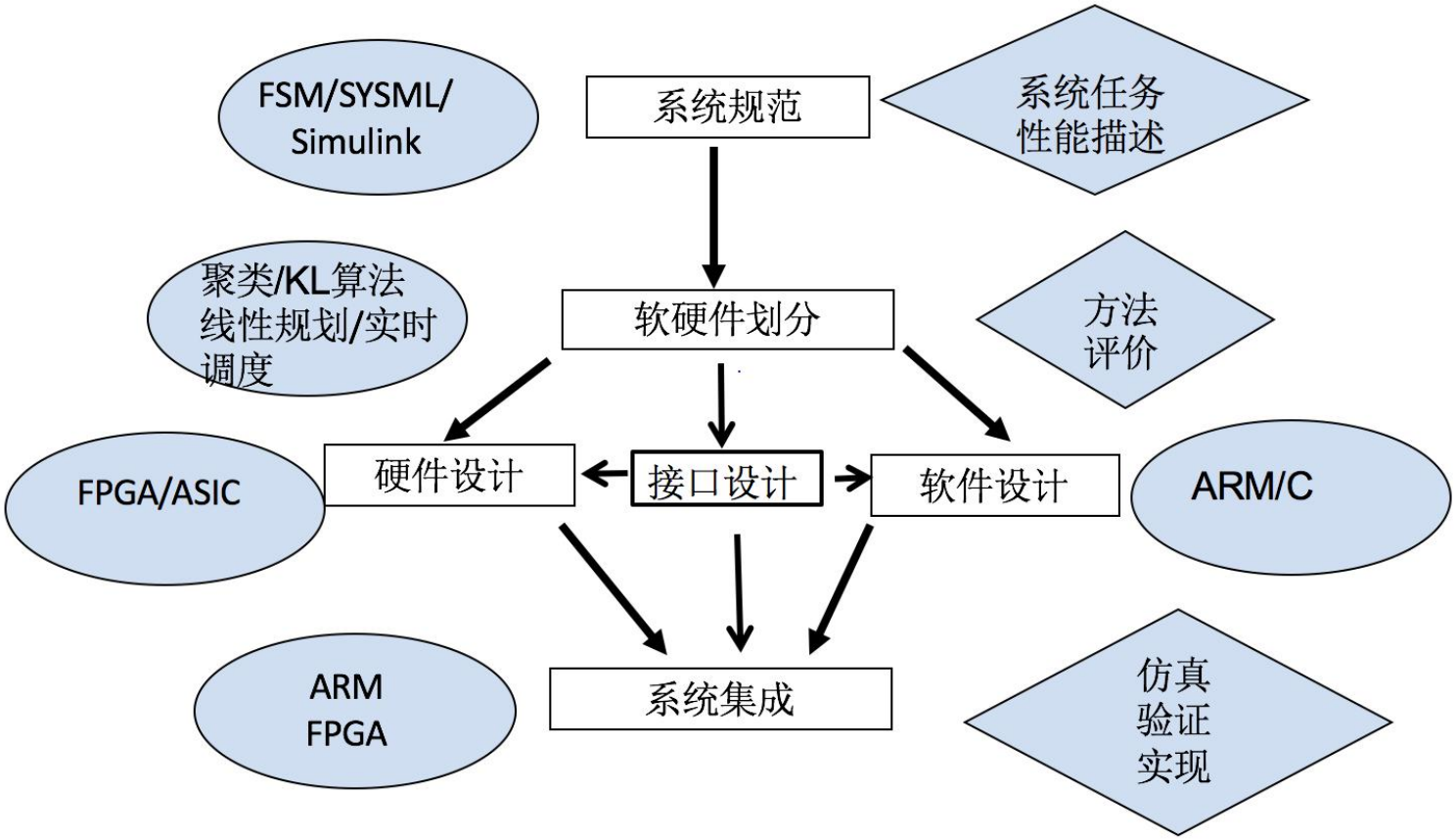
自动停车系统

智能系统

- 智能系统：在硬件基础上融入人工智能，让机器通过一定的方式进行判断、决策和控制，以便最有效地发挥作用。
- 它是一个具有传感、控制、人机交互、网络接入的实体系统。
- 它也是嵌入式技术、软件工程、人工智能等知识的深入融合系统。
- 使用软件工程方法来开发智能系统



智能系统优化协同设计体系



规范建模

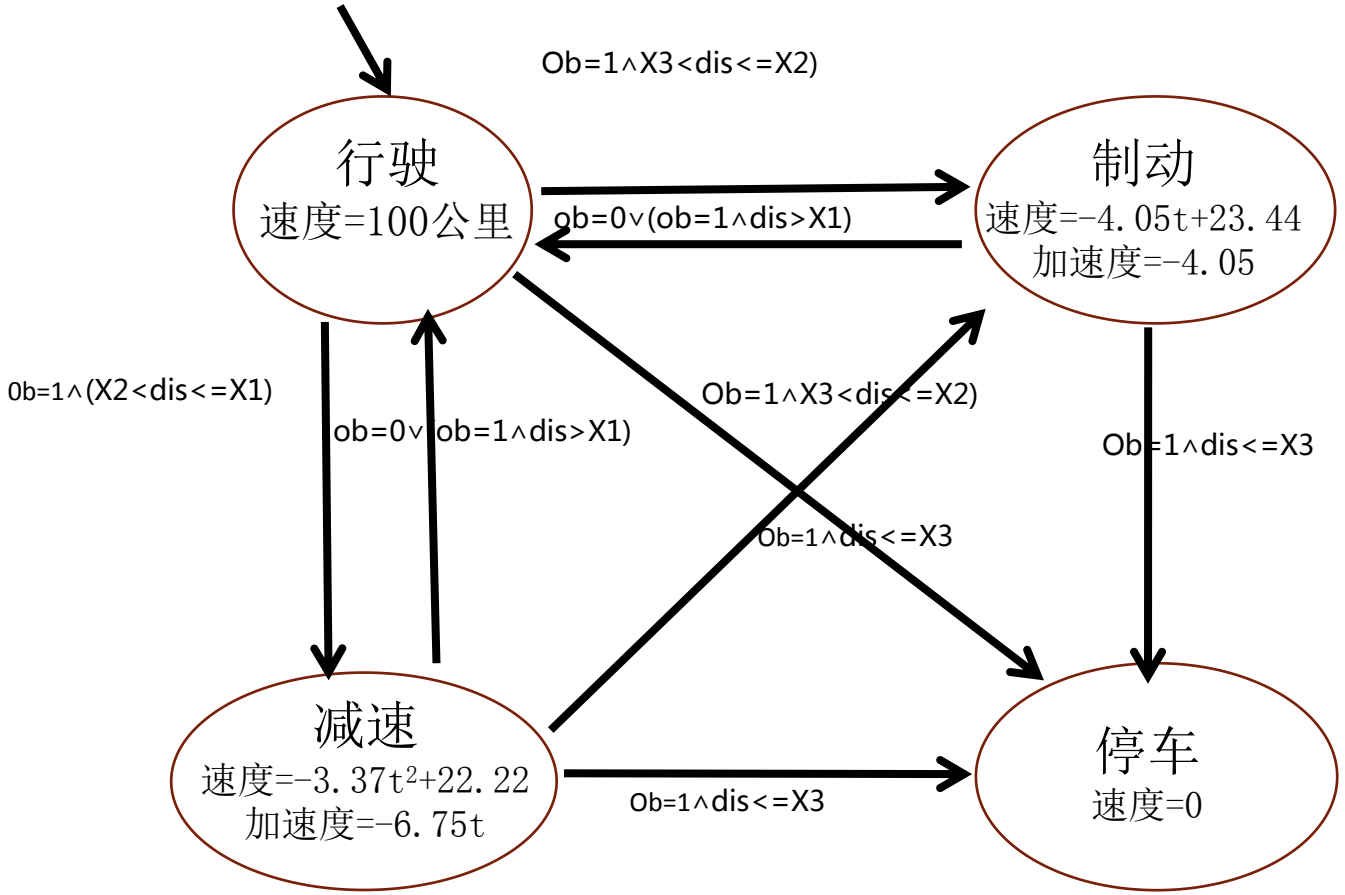
- 系统需求：规定系统需要完成的功能与任务，以及各种指标的确定数值。
  - ✓ 成本是多少？系统的硬件面积和软件字节大小，甚至FPGA的查找表个数；
  - ✓ 时间性能，整个系统完成整个任务需要多少时间？3秒还是3毫秒？
  - ✓ 环保：也会考虑功耗和能效这些指标，功耗和能效是多少？功耗是3瓦还是3.5瓦，能效等级是属于3级还是1级？

规范建模

- 系统规范建模：使用建模工具对系统需求进行建模，包括
  - 有限状态机对离散系统规范建模
  - Simulink对连续状态规范建模
  - 混成自动机对离散-连续系统规范建模
  - 时空一致性规范语言STeC
  - SysML对整个系统进行规范建模。

规范建模

✓ 混成自动机对离散-连续系统规范建模：自动驾驶



- Ob表示障碍物
- Xi表示汽车离障碍物的距离
- 四个离散状态：行驶、制动、减速、停车
- 在每个离散状态中都有速度和加速度组成的微分方程。

规范建模

✓时空一致性规范语言STeC

- 环境“土路（湿）：0.413”下，制动初速度80km/h，以制动距离模型制动

$$\begin{cases} \dot{s}(u)_p = -3.37u^2 + 22.22 \\ \dot{v}(u)_p = -6.75u \end{cases} \qquad \begin{cases} \dot{s}(u)_p = -4.05u + 23.44 \\ \dot{v}(u)_p = -4.05 \end{cases}$$

- pSTeC操作语义

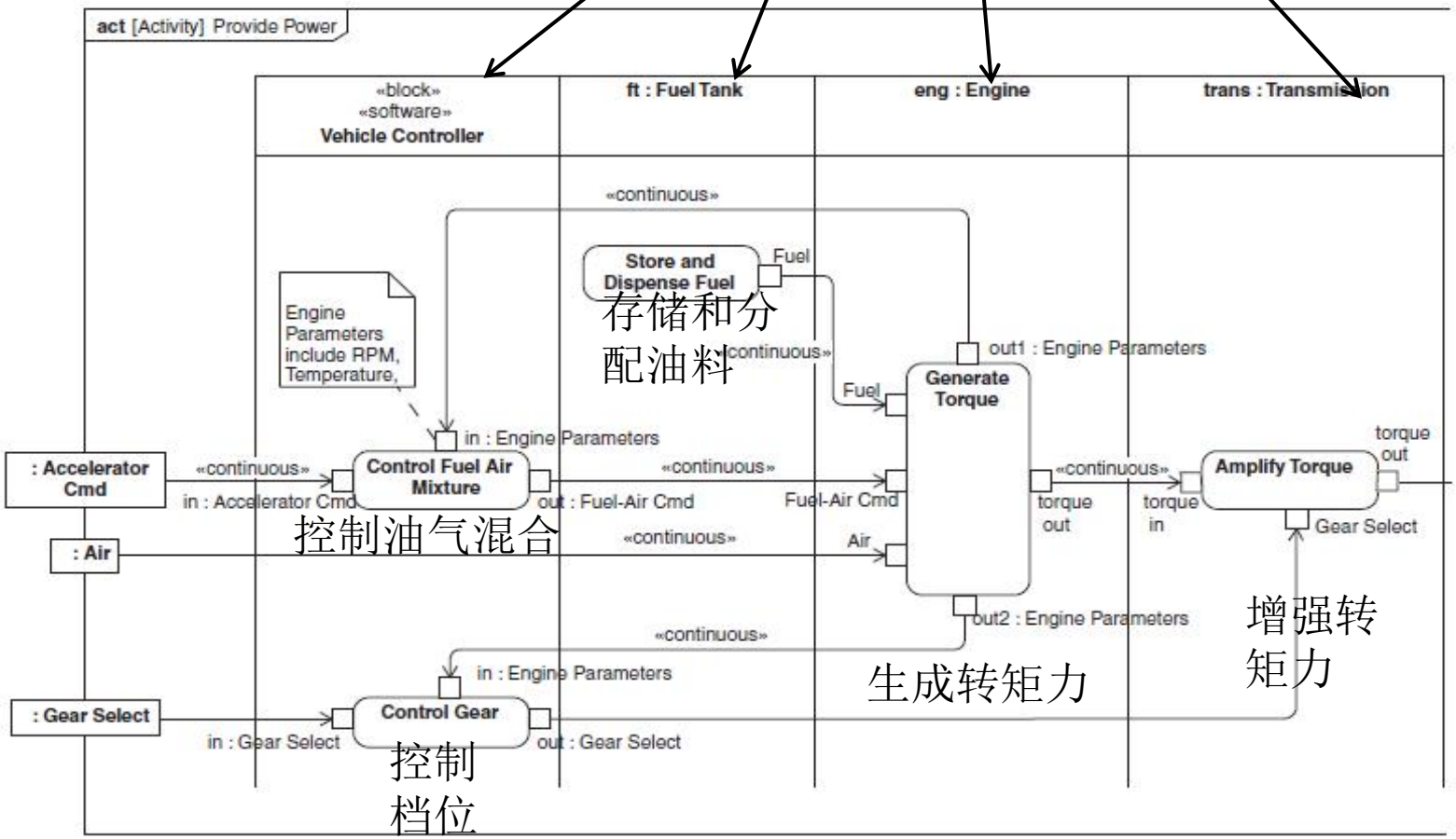
$$\langle \mathrm{RunDown}_1{}^{G}_{(0,0)}(13.09, 0.6)(\{0.413\}), (0, 0, \sigma, 22.22)\rangle$$

$$\xrightarrow[\dot{v}(u)_p=-6.75u]{\dot{s}(u)_p=-3.37u^2+22.22} \langle \mathrm{RunDown}_2{}^{G}_{(13.09,0.6)}(67.61, 5.19)(\{0.413\}),(13.09,0.6,\sigma,21.01)\rangle$$

$$\xrightarrow[\dot{v}(u)_p=-4.05]{\dot{s}(u)_p=-4.05u+23.44} \langle \mathrm{Parking}_{(67.61,5.79)}(\infty)(\{0.413\}), (67.61, 5.79, \sigma, 0)\rangle$$

规范建模 ✓ SysML对整个系统进行规范建模

泳道：描写了系统的构件（功能）



大 纲

智能系统

规范建模

协同优化设计

交通标志识别系统

自动停车系统

协同优化设计

- 依据系统需求进行模块、软硬件划分与设计以及系统集成的过程和方法。
- 为了实现系统的需求，将系统要完成的任务以及指标分解成若干个子任务并附上指标。
- 依据系统的整体需求把这些子任务进行划分成硬件实现和软件实现以及软硬件间的接口协议，形成软件的规范和硬件的规范以及软硬件之间通信协议规范。
- 协同优化设计目标是在满足一定系统一定约束前提下，使系统运行性能达到最优。

协同优化设计

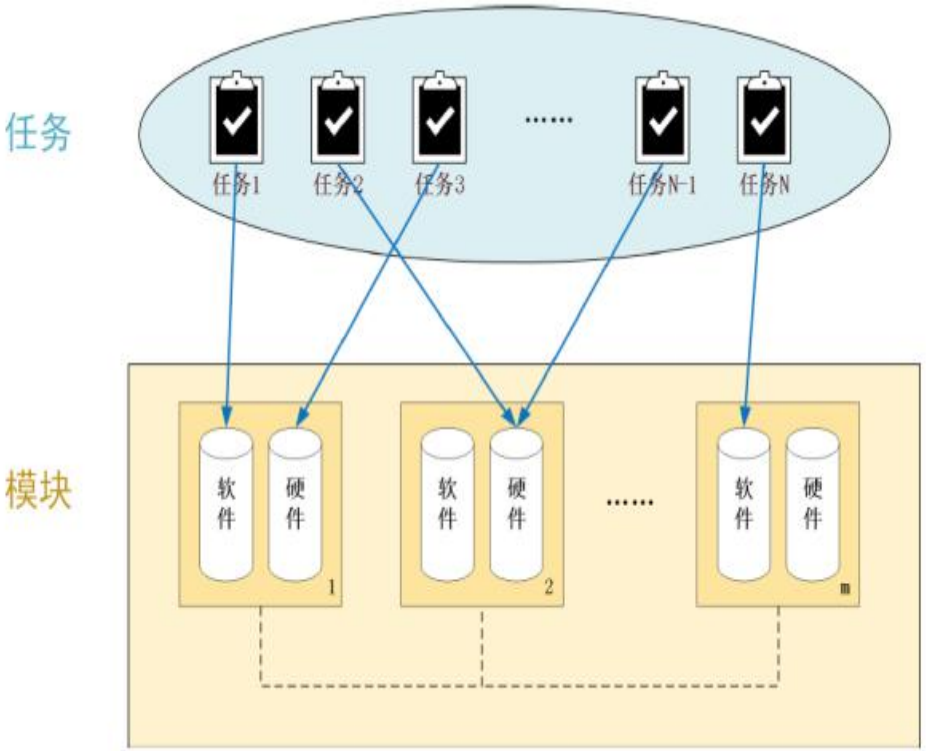
- 智能系统通常会含有众多任务，这些任务之间通过通信进行消息/数据/事件传输，进行协同完成整个系统需要完成的任务。
- 为了满足时间性能的要求，人们会把这些任务分解成多个模块，每个模块由一个或多个处理器或软件与硬件融合的异构平台进行处理。
- 模块内依据任务的性能指标（如时间、功耗、成本、硬件面积等）进行软硬件优化划分。
- 这样处理 可以根据系统各个部分的特点和设计约束选择软件或硬件实现方式, 得到高性能、低成本的优化设计方案。

协同优化设计

- 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，人们会把这些任务分解成多个模块，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。（**多模块划分、多核划分**）
- 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。（**多目标划分**）
- 这样处理 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。
- 高性能、低成本是矛盾的目标约束。

协同优化设计-多模块划分

- 依据任务间的通信代价进行聚类，两个任务通信代价越高或越频繁，越应该把这两个任务划分到一个模块；
- 依据划分结果，计算块间的通信代价，而模块内的通信代价量认为为零；
- 这个过程可以多次进行，最终选一个模块间通信代价之和最小者，算法终止。
- 每个模块内依据任务的性能再进行多目标划分实现模块内的软硬件划分。
- 实现了智能系统的多模块优化划分。



协同优化设计-多模块划分

- 已知智能系统含有N个任务{T1,...,Tn}，任务Ti有软件执行时间TSi，软件执行代价CSi，硬件执行时间THi，硬件执行代价CHi，硬件面积AHi，任务Ti与Tj之间的通信代价C(i,j)。
- 把这N个任务进行模块划分成m个模块，使得模块间通信代价极小化。
- 每个模块内依据任务属性进行多目标划分，实现软硬件优化设计。
- 我们的任务是设计完成上述目标的算法。

- 下图是一个含有8个任务的系统各属性值，划分成3个模块。

任务 T_i	软件实现		硬件实现		
	TS_i	CS_i	TH_i	CH_i	AH_i
T_1	30	80	10	210	5
T_2	40	100	12	263	6
T_3	42	95	10	175	11
T_4	35	76	9	182	7
T_5	34	75	8	156	5
T_6	22	63	7	163	2
T_7	23	51	5	144	4
T_8	20	49	6	99	8

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
T_1	0	1	3	6	10	2	5	7
T_2	1	0	8	2	10	12	13	15
T_3	3	8	0	8	20	13	15	17
T_4	6	2	8	0	19	18	17	5
T_5	10	10	20	19	0	20	11	18
T_6	2	12	13	18	20	0	12	13
T_7	5	13	15	17	11	12	0	5
T_8	7	15	17	5	18	13	5	0

协同优化设计-多模块划分

◆我们的任务是设计完成上述目标的算法。

算法 1 $HSPA_{Mod}$

Input:

Task set T;
Matrix of communication cost A;
The number of modulars m;
The limit of number of tasks in one modular max;
The number of iterations I;
Constraints;

Output:

Result of Modularization and Hardware/Software Partitioning;

```
Modularize;
Partition for the modularization result;
count = 1;
while count < I do
    ReModularize;
    Partition for the modularization result;
    count++;
end while
Select the best modularization and Hardware/Software Partitioning solution;
return Result of Modularization and Hardware/Software Partitioning;
```

算法 2 Modularize

Input:

Matrix of communication cost A;
The number of modulars m;
Limit of the number of tasks in one modular max;

Output:

Task sets $M_1, M_2, \cdots, M_m$;

```
List < int[] > edges = null;
Add all [a_{i,j}, i, j] into edges (a_{i,j}! = 0. i < j);
edges.sort();
for every [a_{i,j}, i, j] in edges do
    if both T_i and T_j haven't been allocated then
        Select the set with fewest tasks (mark as M_0);
        if |M_0| + 2 <= max then
            M_0 = M_0 ∪ T_i ∪ T_j;
            Delete T_i and T_j from task T;
        end if
    else
        if one of T_i and T_j has been allocated then
            if |the modular that T_i or T_j allocated in|+1 <= max then
                Add the other task into this modular;
                Delete the from task set T;
            end if
        end if
    end if
    if T.empty() then
        break;
    end if
end for
while !T.empty() do
    Add T.get(0) into the set with fewest tasks;
    T.remove(0);
end while
return Task sets M_1, M_2, ..., M_m;
```

Modularize算法复杂度为$O(mN^2)$

协同优化设计-多模块划分

◆ 下图是一个含有8个任务的系统各属性值，划分成3个模块。

任务 T_i	软件实现		硬件实现		
	TS_i	CS_i	TH_i	CH_i	AH_i
T_1	30	80	10	210	5
T_2	40	100	12	263	6
T_3	42	95	10	175	11
T_4	35	76	9	182	7
T_5	34	75	8	156	5
T_6	22	63	7	163	2
T_7	23	51	5	144	4
T_8	20	49	6	99	8

$$
\begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \end{array} \\
\left[\begin{array}{cccccccc}
0 & 1 & 3 & 6 & 10 & 2 & 5 & 7 \\
1 & 0 & 8 & 2 & 10 & 12 & 13 & 15 \\
3 & 8 & 0 & 8 & 20 & 13 & 15 & 17 \\
6 & 2 & 8 & 0 & 19 & 18 & 17 & 5 \\
10 & 10 & 20 & 19 & 0 & 20 & 11 & 18 \\
2 & 12 & 13 & 18 & 20 & 0 & 12 & 13 \\
5 & 13 & 15 & 17 & 11 & 12 & 0 & 5 \\
7 & 15 & 17 & 5 & 18 & 13 & 5 & 0
\end{array}\right]
\end{array}
$$

模块	任务
M_1	T_3, T_5, T_6
M_2	T_4, T_7
M_3	T_1, T_2, T_8

模块	软件任务	硬件任务	时间	开销	面积
M_1	T_5, T_6	T_3	66	313	11
M_2	T_4, T_7	-	58	127	0
M_3	T_1, T_2	T_8	76	279	8

划分结果

协同优化设计-多模块划分

◆ 下图是一个含有8个任务的系统各属性值，划分成3个模块。

任务 T_i	软件实现		硬件实现		
	TS_i	CS_i	TH_i	CH_i	AH_i
T_1	30	80	10	210	5
T_2	40	100	12	263	6
T_3	42	95	10	175	11
T_4	35	76	9	182	7
T_5	34	75	8	156	5
T_6	22	63	7	163	2
T_7	23	51	5	144	4
T_8	20	49	6	99	8

$$
\begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \end{array} \\
\left[\begin{array}{cccccccc}
0 & 1 & 3 & 6 & 10 & 2 & 5 & 7 \\
1 & 0 & 8 & 2 & 10 & 12 & 13 & 15 \\
3 & 8 & 0 & 8 & 20 & 13 & 15 & 17 \\
6 & 2 & 8 & 0 & 19 & 18 & 17 & 5 \\
10 & 10 & 20 & 19 & 0 & 20 & 11 & 18 \\
2 & 12 & 13 & 18 & 20 & 0 & 12 & 13 \\
5 & 13 & 15 & 17 & 11 & 12 & 0 & 5 \\
7 & 15 & 17 & 5 & 18 & 13 & 5 & 0
\end{array}\right]
\end{array}
$$

模块	任务
M_1	T_3, T_5, T_6
M_2	T_4, T_7
M_3	T_1, T_2, T_8

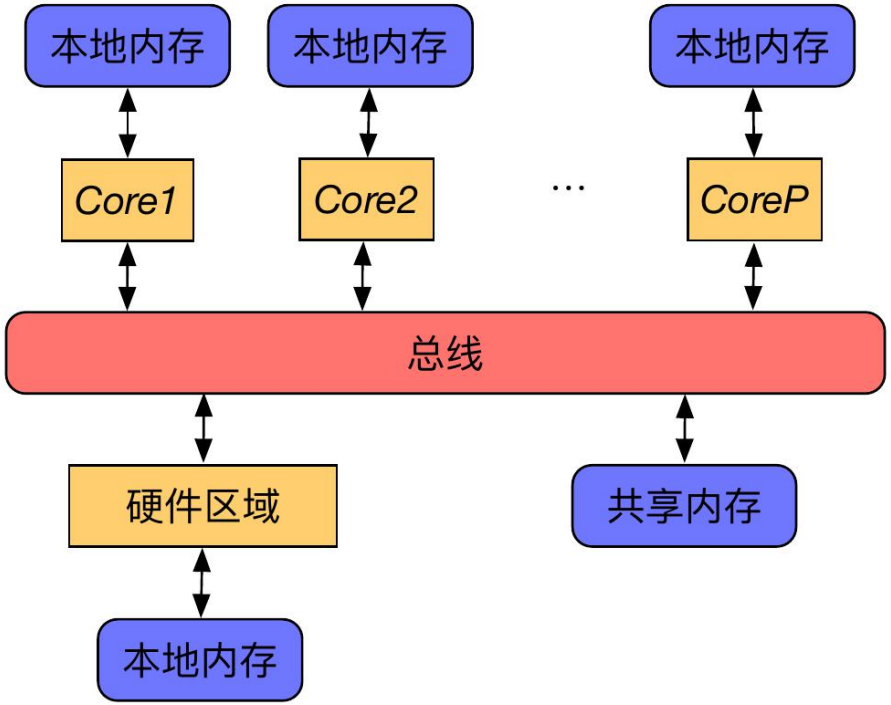
约束	模块	软件任务	硬件任务
开销约束	M_1	T_1, T_6	T_4
	M_2	T_2, T_8	T_5
	M_3	T_3, T_7	-
时间约束	M_1	T_5, T_6	T_3
	M_2	T_4, T_7	-
	M_3	T_1, T_2	T_8

协同优化设计-多核划分

- 将系统的N个任务划分到P个处理器上执行，这n个任务有执行依赖关系，以及性能属性。
  - ✓ 这个划分问题是强NP难的。
- 针对异构多核片上系统设计了一种基于任务调度的软硬件划分算法$HSPA_{TS}$，
- 该算法将遗传算法与基于基于静态优先级的表调度方法相结合，
- 能够给出有效缩短系统总体运行时间的软硬件优化设计方案，
- 并且满足任务依赖约束关系以及系统硬件面积的约束。

协同优化设计-多核划分

- 目标架构



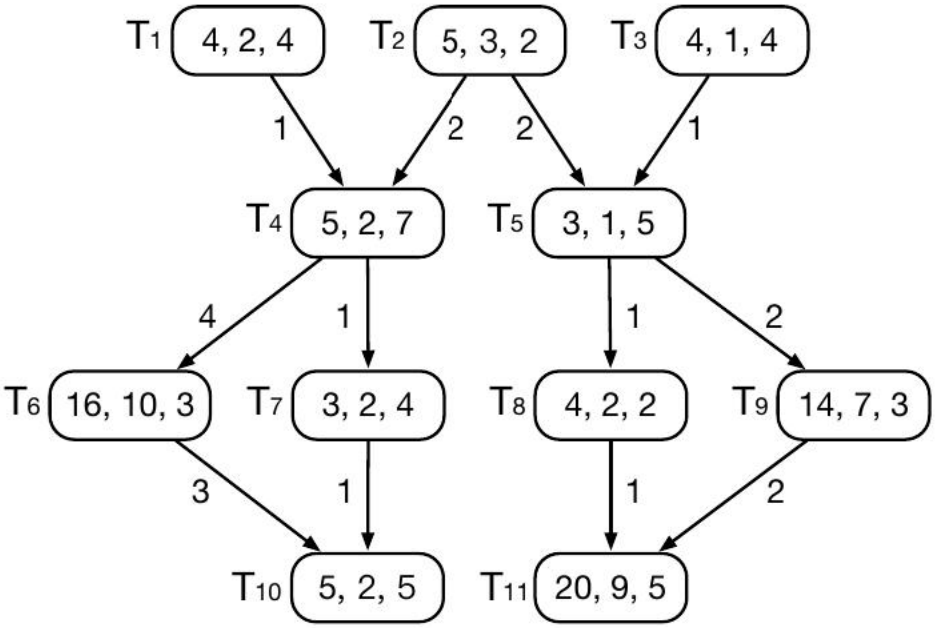
简化的多核异构片上系统结构

协同优化设计-多核划分

➢任务图

- 任务图 $G = \{T, E\}$
- T中任务$T_i = (TS_i, TH_i, AH_i)$
  TS_i为软件执行时间；
  TH_i为硬件执行时间；
  AH_i为硬件执行面积。
- E为边的集合,$(i,j) \in E$表示一条边,
  $e(i,j)$为任务T_iT_j间通信代价。

- 调度后$\bar{T}_i = (p_i, ts_i, tf_i)$
  p_i表示任务分配被到哪个处理单元；
  ts_i为任务开始时间；
  tf_i为任务结束时间。



任务图

协同优化设计-多核划分

- 算法描述

➢ 算法目标

$$Minimize \quad L = \max_{i \in \{1,2,\ldots,N\}} \{tf_i\}$$

➢ 约束条件

$$s.t. \quad S = \sum_{i=1}^{N} a_i AH_i \leq S_l \qquad (\text{仅当} T_i \text{为硬件任务时} a_i = 1)$$

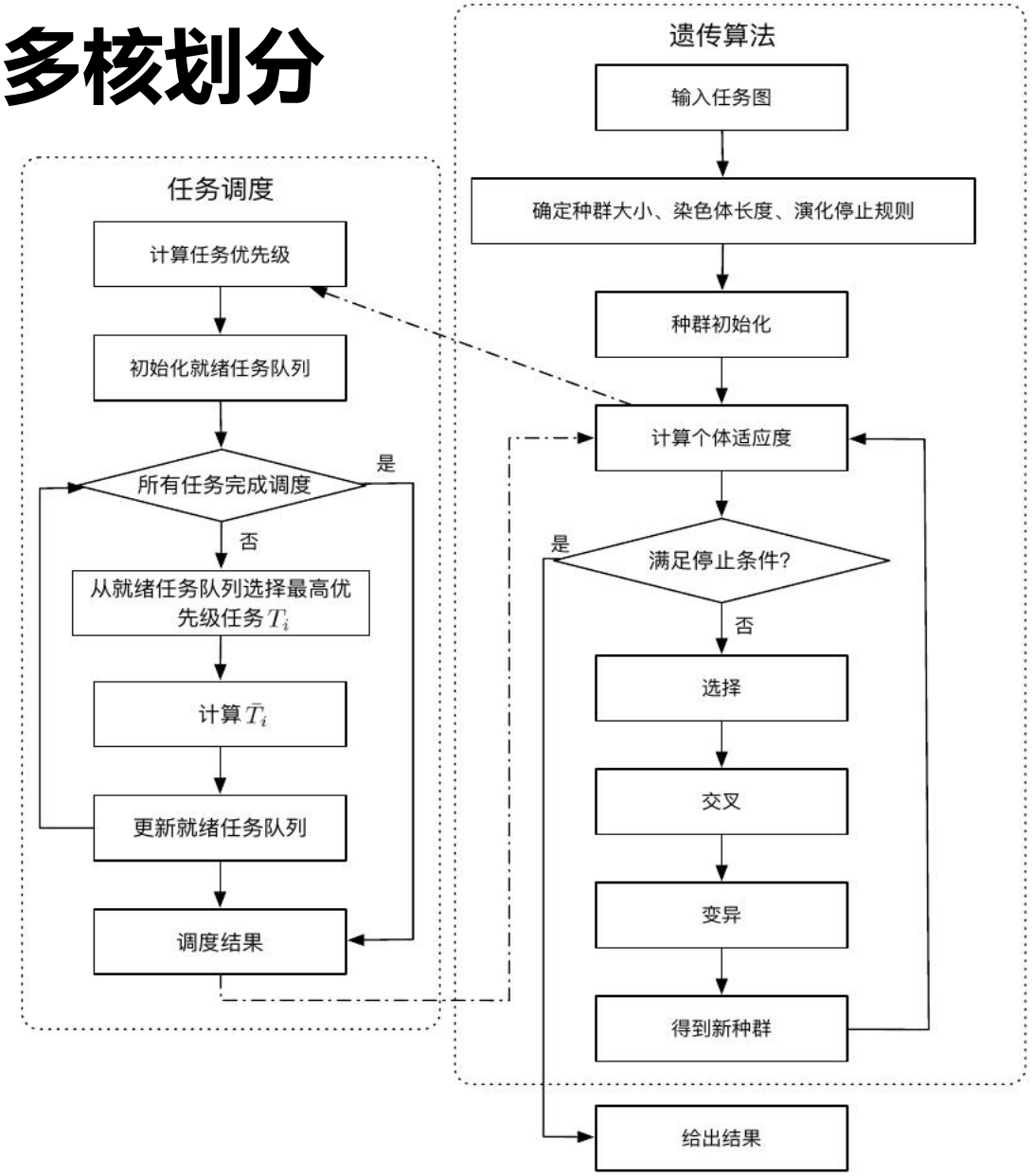
协同优化设计-多核划分

■ 算法描述

➢ 算法流程

- 流程图中右边的部分为遗传算法部分。

- 流程图左边的部分为任务调度部分，在遗传算法为每个个体计算适应度时都被调用一次，并将得到的调度结果传回给遗传算法。



协同优化设计-多核划分

- 算法描述

遗传算法部分

- 种群大小：$2N$
- 个体染色体长度：N，0表示用软件实现，1表示用硬件实现。
- 演化停止规则：演化到$3N$代
- 初始化种群：为保证本算法能找到满足硬件面积约束的解，使用贪心算法生成一个个体α加入到初始种群。其他个体则随机产生。
- 个体α的生成：

步骤1：

计算每个任务T_i的硬件收益时间：

$$B_i = TS_i - TH_i$$

步骤2：

迭代查找最大硬件收益任务，将其放到硬件执行集合中，直到不满足面积约束为止。将硬件集合中的任务对应的基因设为1，其余设为0，则得到了一个初始个体α。

协同优化设计-多核划分

- 算法描述

遗传算法部分

- 选择机制：精英机制与轮盘赌方法相结合，保留每一代中个体适应度最大的前1/4的个体进入下一代，下一代中其他个体由使用轮盘赌方法选出的父母进行交叉产生。轮盘赌方法中各个个体被选中的概率与其适应度函数值大小成正比。
- 交叉操作：采用两点交叉法，随机选取染色体中两点间的基因进行交换。
- 变异操作：按照5%的变异概率进行变异操作，每次变异可随机改变1-3个基因。
- 适应度函数：对于每个个体，其染色体序列表示了一种对任务的软硬件划分，该划分所需硬件面积　，对划分后的任务采用基于静态优先级的调度算法得到调度总时长　，则个体适应度函数为：

$$Fitness = \frac{1}{\omega_1 \cdot e^{\frac{S-S_l}{\sigma_1}} \cdot \frac{|S-S_l|}{\sigma_1} + \omega_2 \cdot \frac{L}{\sigma_2}}$$

协同优化设计-多核划分

- 算法描述

任务调度部分：基于静态优先级的表调度算法

步骤1：
计算任务静态优先级，任务优先级定义为：

$$pri(T_i) = \begin{cases} TS_i + \max\limits_{T_j \in SUC(i)} \{e(i,j) + pri(T_j)\}, & \text{若 } T_i \text{ 为软件任务} \\ TH_i + \max\limits_{T_j \in SUC(i)} \{e(i,j) + pri(T_j)\}, & \text{若 } T_i \text{ 为硬件任务} \end{cases}$$

步骤2：
初始化就绪任务队列，任务就绪是指该任务的所有前驱任务已经执行完成。初始任务序列中包含前驱任务集为空的所有任务。

步骤3：
当就绪任务队列非空时，从就绪任务队列中选出当前优先级最高的任务，若为硬件任务则分配至硬件执行，为软件任务则为其选择最佳处理器核。

协同优化设计-多核划分

- 算法复杂度分析

  - 基于静态优先级的表调度算法的时间复杂度为 $O(PN^2)$
  - $HSPA_{TS}$ 算法中有3N代种群，每代种群中有2N个个体需进行调度，总的时间复杂度为$O(PN^4)$

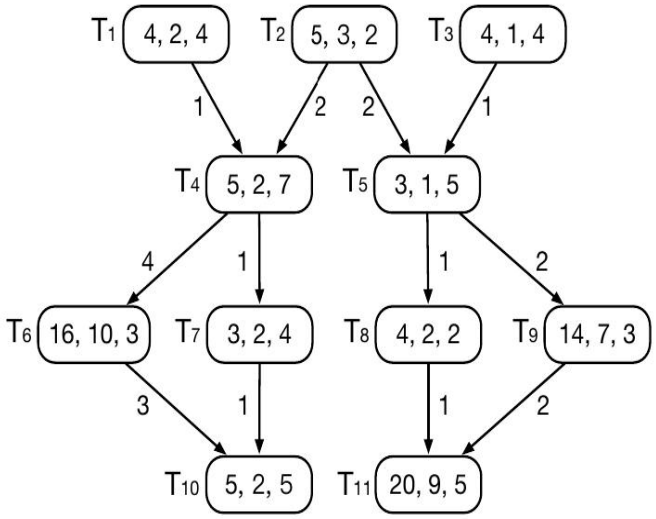
算法 5 $HSPA_{TS}$

Input:
Task Graph G; Number of processors P; Probability of mutation P_m;
Output:
Result of Hardware/Software Partitioning and Scheduling;

```
1: Size = 2N; Gen = 3N; gen = 0;
2: Generate individual α by using GeneAlpha;
3: Generate other individuals of the first generation pop randomly;
4: for every individual in pop do
5:    Compute fitness by using LSSP;
6: end for
7: while gen < Gen do
8:    newpop = ∅;
9:    Pick up the individuals with highest fitnesses and add them to newpop;
10:   while |newpop|< Size do
11:      Pick up two individuals indi_1, indi_2;
12:      (indi_3, indi_4) = cross(indi_1, indi_2);
13:      Add indi_3 and indi_4 to newpop;
14:   end while
15:   for every individual in newpop do
16:      if random (0, 1) < P_m then
17:         Mutation;
18:      end if
19:   end for
20:   pop = newpop; gen + +;
21:   for every individual in pop do
22:      Compute fitness by using LSSP;
23:   end for
24: end while
25: Pick up the individual with highest fitness;
26: return Result for Hardware/Software Partitioning and Scheduling;
```

协同优化设计-多核划分

- 算法示例



- 设定处理器内核数$p = 2$，硬件面积约束$S_l = 18$。
- 首先根据任务数11设置种群大小为22，染色体长度为11，演化代数为33。
- 然后初始化种群，根据贪心算法计算得到个体α。

步骤1：计算得到每个任务硬件收益时间分别为2, 2, 3, 3, 2, 6, 1, 2, 7, 3, 11。

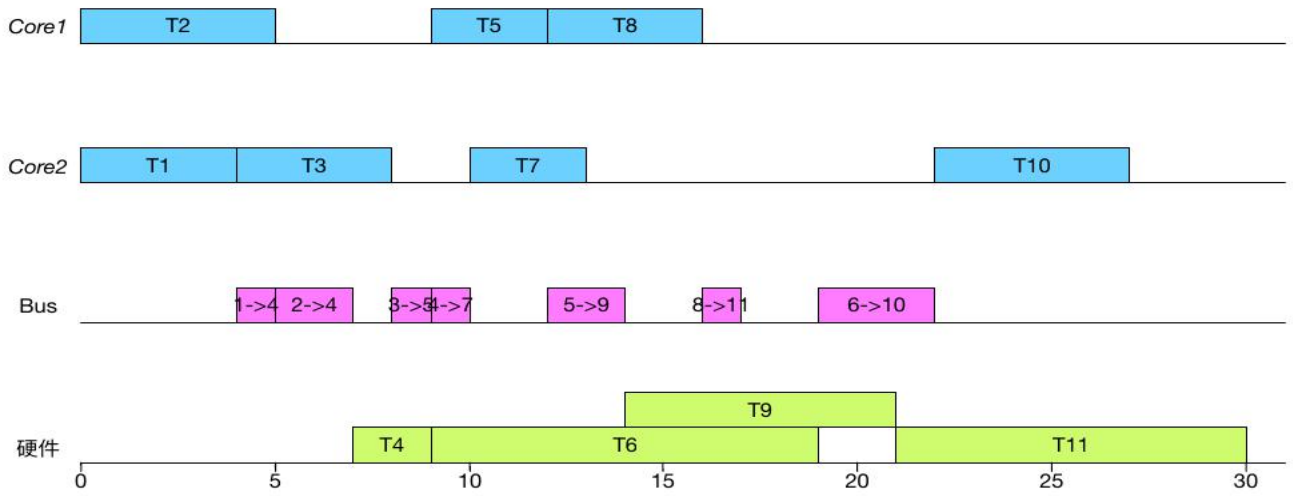
步骤2：依次选出硬件收益最大的任务T_{11}, T_9, T_6, T_4，硬件面积之和5+3+3+7=18。则一个初始个体α为00010100101。其余初始个体随机产生。

协同优化设计-多核划分

▪ 算法示例

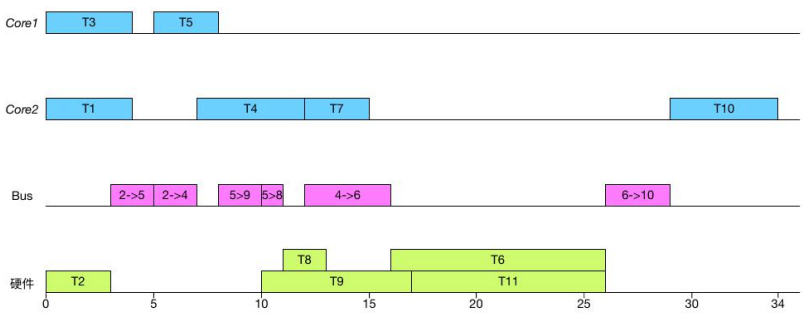
- 随后对于种群中的每个个体计算其适应度。以个体α为例，使用基于静态优先级的调度算法对其所代表的划分后的任务进行调度。

步骤1：计算得到每个任务的静态优先级分别为29, 31, 28, 24, 23, 18, 9, 14, 18, 5, 9。
步骤2：初始就绪任务队列为$\{T_1, T_2, T_3\}$。
步骤3：不断从非空的就绪任务队列中挑出优先级最高的任务，选出最佳处理器核（硬件），得到的调度结果如图所示，调度长度为30，计算得到个体适应度为$F = 9.2222$。

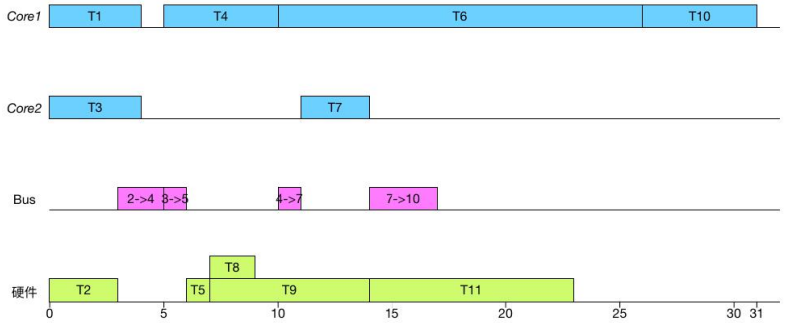


协同优化设计-多核划分

- 算法对比



CPCS 算法调度结果（双核）



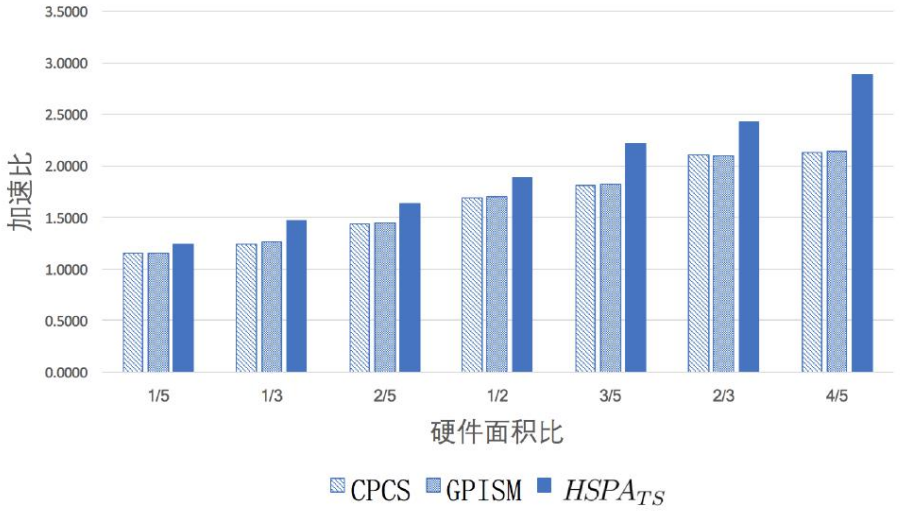
GPISM 算法调度结果（双核）

算法	处理器核数	调度长度	硬件任务集合	硬件面积	硬件利用率
$HSPA_{TS}$	2	28	$\{T_2, T_6, T_9, T_{10}, T_{11}\}$	18	100%
	1	32	$\{T_1, T_2, T_6, T_9, T_{11}\}$	17	94.4%
CPCS	2	34	$\{T_2, T_6, T_8, T_9, T_{11}\}$	15	83.3%
	1	39	$\{T_2, T_6, T_8, T_9, T_{11}\}$	15	83.3%
GPISM	2	31	$\{T_2, T_5, T_8, T_9, T_{11}\}$	17	94.4%
	1	34	$\{T_2, T_5, T_8, T_9, T_{11}\}$	17	94.4%

协同优化设计-多核划分

算法对比

- 定义硬件面积比为硬件面积约束与所有任务所需硬件面积之和的比；
- 加速比为所有任务用软件实现所需时间与使用了硬件加速由算法调度得出的实际时间的比。
- $HSPA_{TS}$ 算法在不同硬件面积比的情况下都能比其他两种算法得到更高的加速比。



算法平均结果对比

大 纲

智能系统

规范建模

协同优化设计

交通标志识别系统

自动停车系统

交通标志识别系统

- 自动驾驶技术在近年来得到了越来越多的关注。
- 若自动驾驶汽车具备识别交通标志的能力，则可以相应地进行控制车速、自动规划路线、避让障碍物等操作。
- 对路面已有交通标志的识别可以使得自动驾驶汽车更快地应用到实际驾驶中。

交通标志识别系统

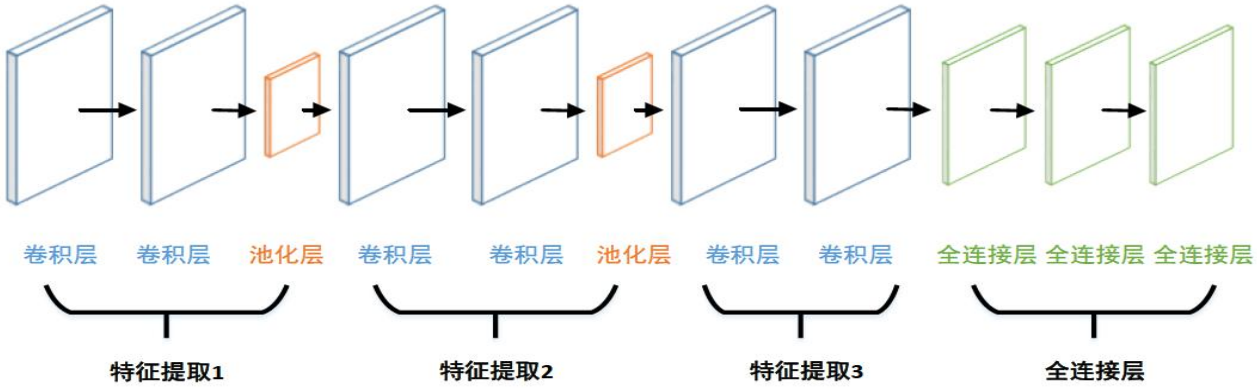
➢ 系统概述

- 系统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，系统处理时间在毫秒级。
- 系统可识别限速、停止、禁止左转、注意行人等43种常用交通标志。
- 使用提前训练好的BNN分类器进行标志的识别，以降低传统CNN中庞大的计算量。
- 本交通标志识别系统共包括四部分：图片输入、图片预处理、BNN分类决策和识别结果输出。

交通标志识别系统

- BNN层次结构如下图所示，包括6个卷积层，2个池化层和3个全连接层。其中卷积层用来进行特征提取；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。
- 最后三层全连接层则连接所有的特征，计算每类别对应的得分。我们根据层次功能，把每次特征提取看做一个任务，分别称为特征提取1，特征提取2和特征提取3，再将最后3个全连接层看做一个整体，整个BNN分类器被分成了四个任务。我们将对这四个任务进行软硬件划分。

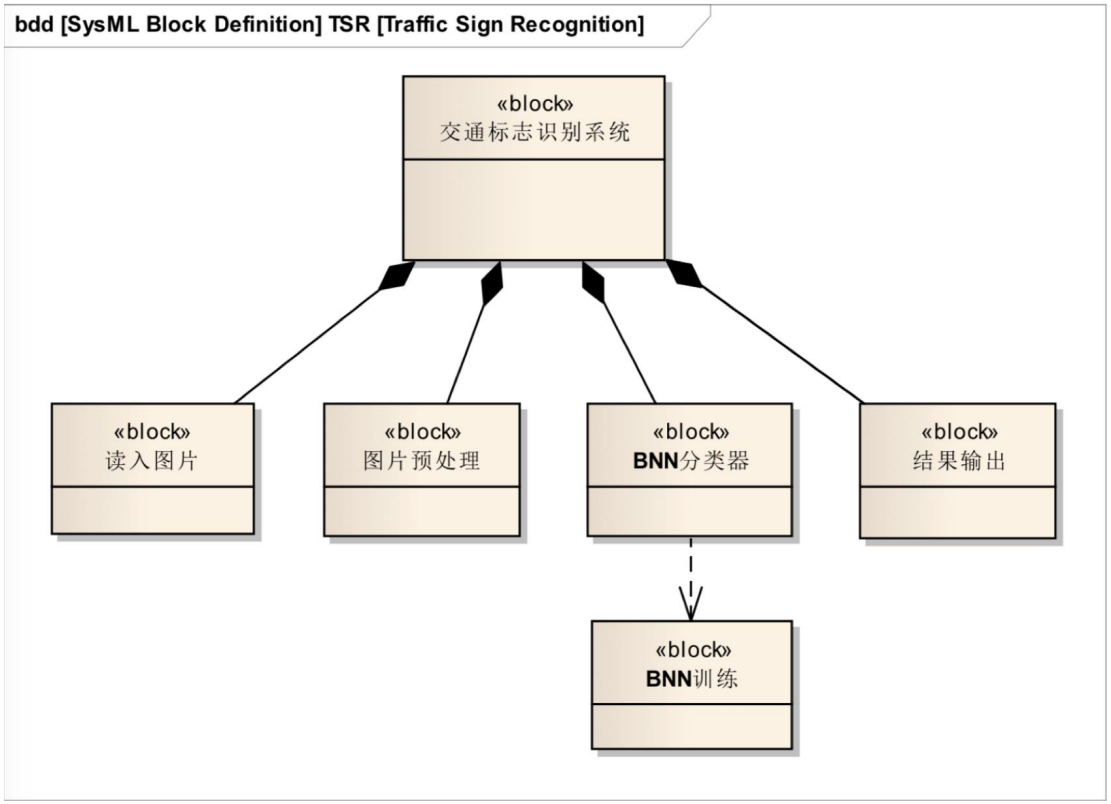
BNN 分类器



交通标志识别系统

- 系统建模

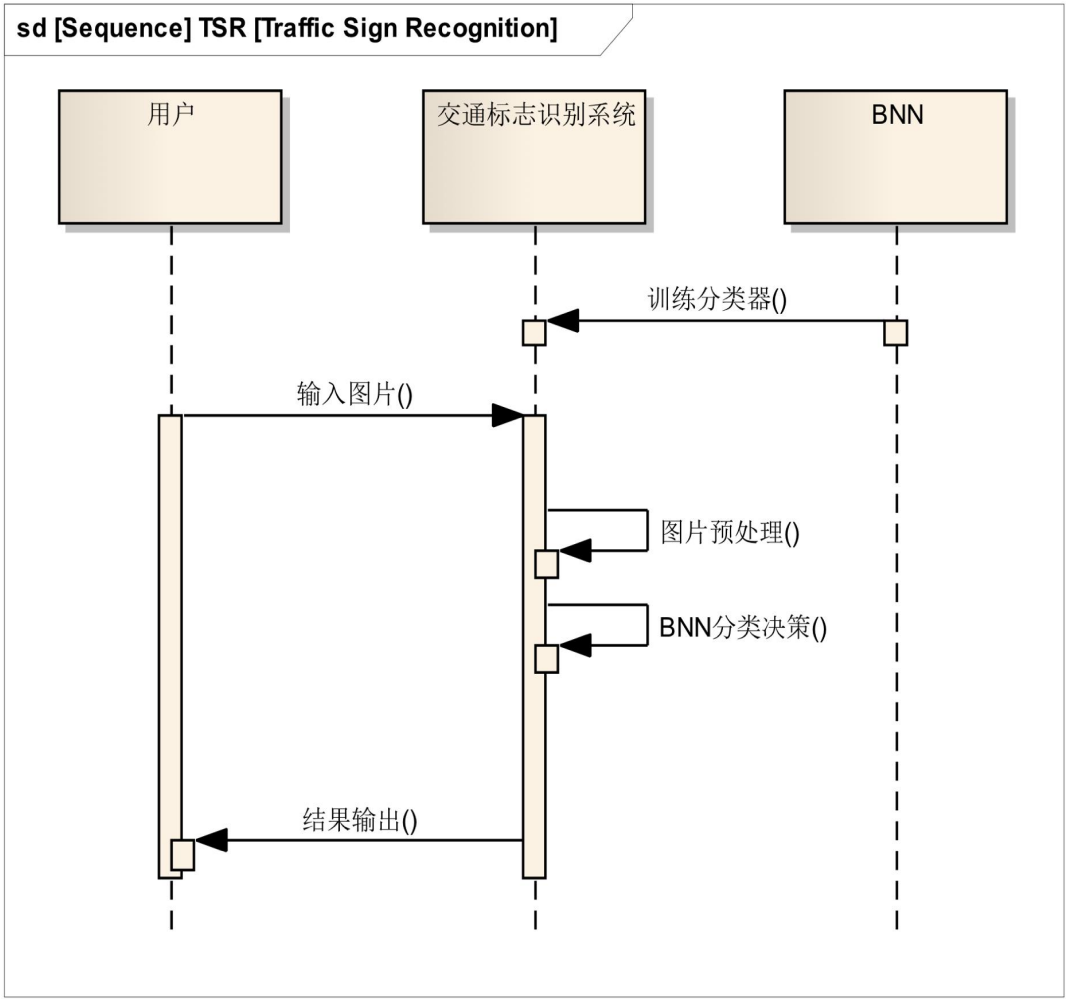
SysML建模



基于BNN的交通标志识别系统

- 系统建模

SysML建模



交通标志识别系统

系统约束

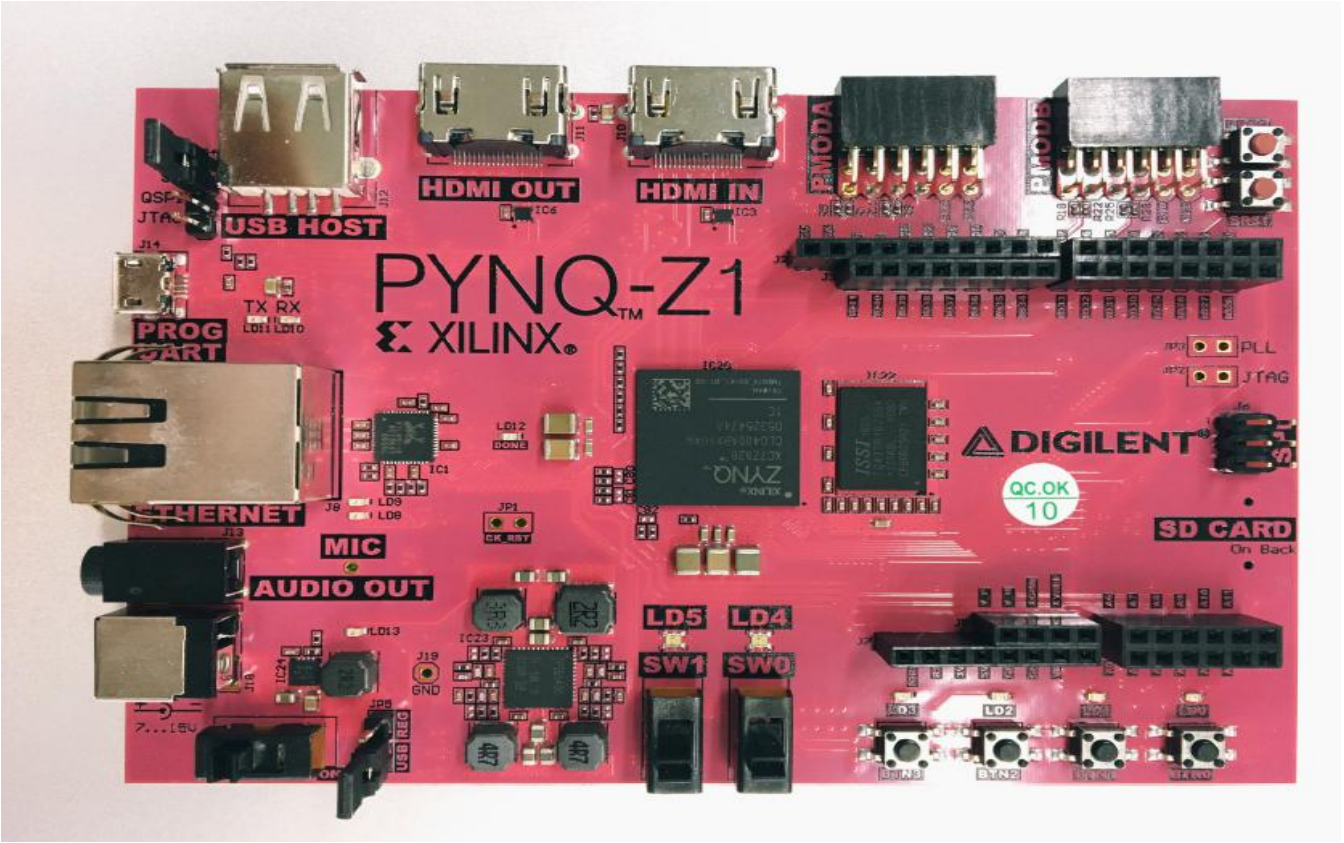
- 使用FPGA对TSR系统进行硬件加速。LUT是FPGA的基本单元，LUT的数量对FPGA的性能有着重要影响，不同的FPGA中所包含的LUT资源数也不同，因此开发所使用的FPGA中LUT资源数量是系统实现的重要约束。

- 本系统采用PYNQ-Z1开发板进行开发，该设备中LUT数量为53200个，因此本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 音频输入输入接口 千兆以太网接口 USB OTG 接口 Arduino 和 Pmod 等
其他	LEDx6、按键 x4、开关 x2

交通标志识别系统

系统约束



使用PYNQ-Z1开发板完成系统开发

交通标志识别系统

BNN分类器

- 本系统中图片输入、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的实现，我们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。

- 若整个BNN使用FPGA加速，所需LUT数量为72589个，超过了资源数53200。 如果换成LUT 数量更多的开发板，其价格较高（市场价4000元左右），会大大增加实现成本。

- 因此将BNN进行划分，将部分任务用硬件实现。

Utilization Estimates

Summary

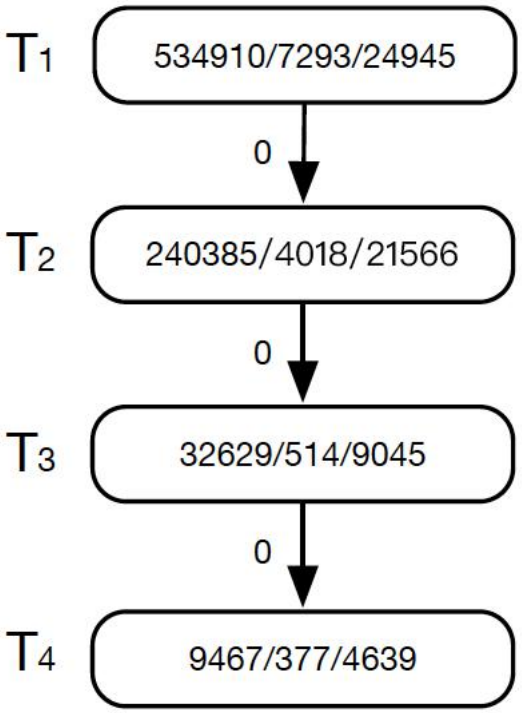
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	2
FIFO	-	-	-	-
Instance	43	52	69569	65232
Memory	189	-	3104	416
Multiplexer	-	-	-	6939
Register	-	-	319	-
Total	232	52	72992	72589
Available	280	220	106400	53200
Utilization (%)	82	23	68	136

交通标志识别系统

系统性能指标

- 任务软件执行时间由执行C++代码得到。
- 硬件执行时间及硬件面积（LUT数量）通过Vivado HLS综合结果估算得出。

任务	软件执行时间	硬件执行时间	硬件面积	备注
T_1	534910 μs	7293 μs	24945 ↑ LUT	特征提取 1
T_2	240385 μs	4018 μs	21566 ↑ LUT	特征提取 2
T_3	32629 μs	514 μs	9045 ↑ LUT	特征提取 3
T_4	9467 μs	377 μs	4639 ↑ LUT	全连接层



交通标志识别系统

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。
- 使用开发的软硬件划分工具进行划分。

交通标志识别系统

- 软硬件划分

硬件实现	软件实现
特征提取 1 特征提取 2	图片读入 预处理 特征提取 3 全连接层 识别结果输出

软硬件划分结果

交通标志识别系统

- 软硬件设计

➢ 软件设计

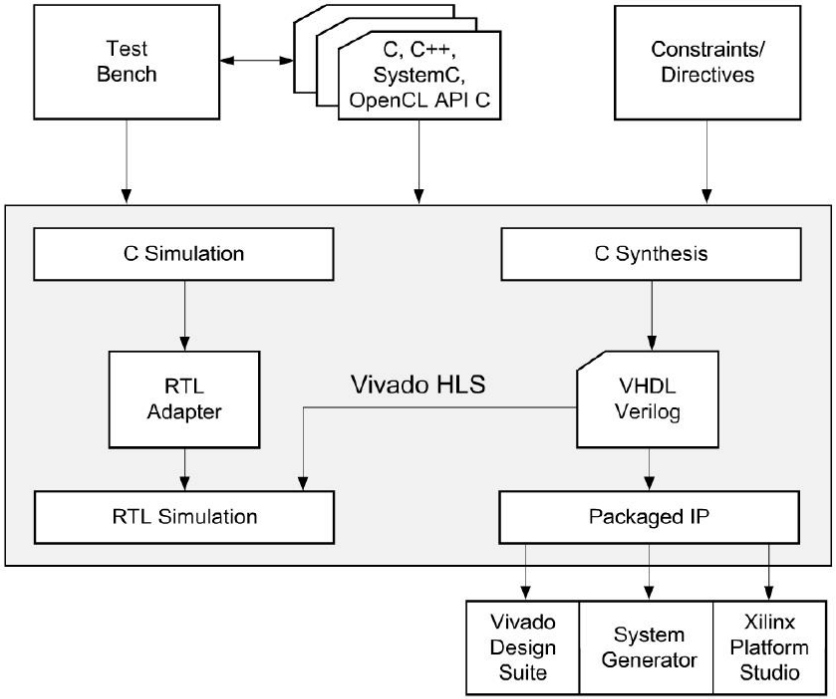
- 特征提取3，全连接层：
  - ➢ C++

- 图片读入、预处理、结果输出：
  - ➢ Python

交通标志识别系统

- 软硬件设计

➢ 硬件设计



Vivado HLS 开发流程

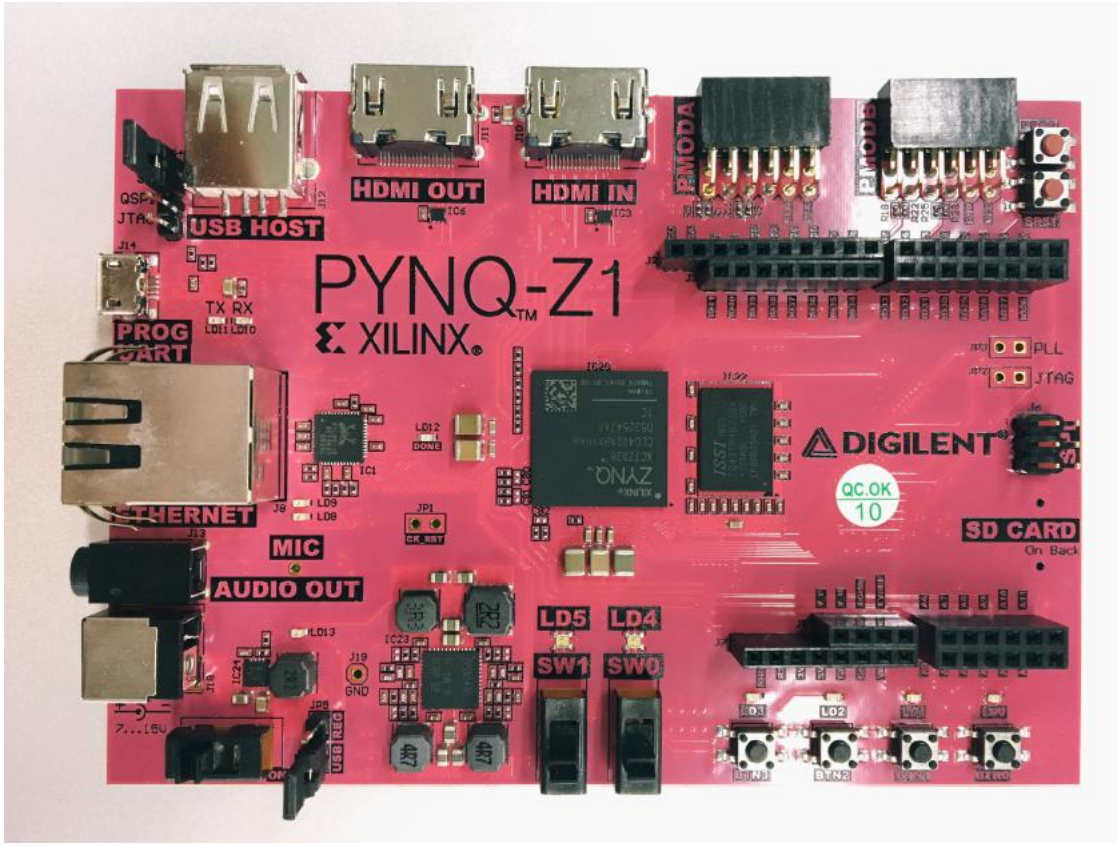
Utilization Estimates

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

交通标志识别系统

- 系统集成
  - 实现环境

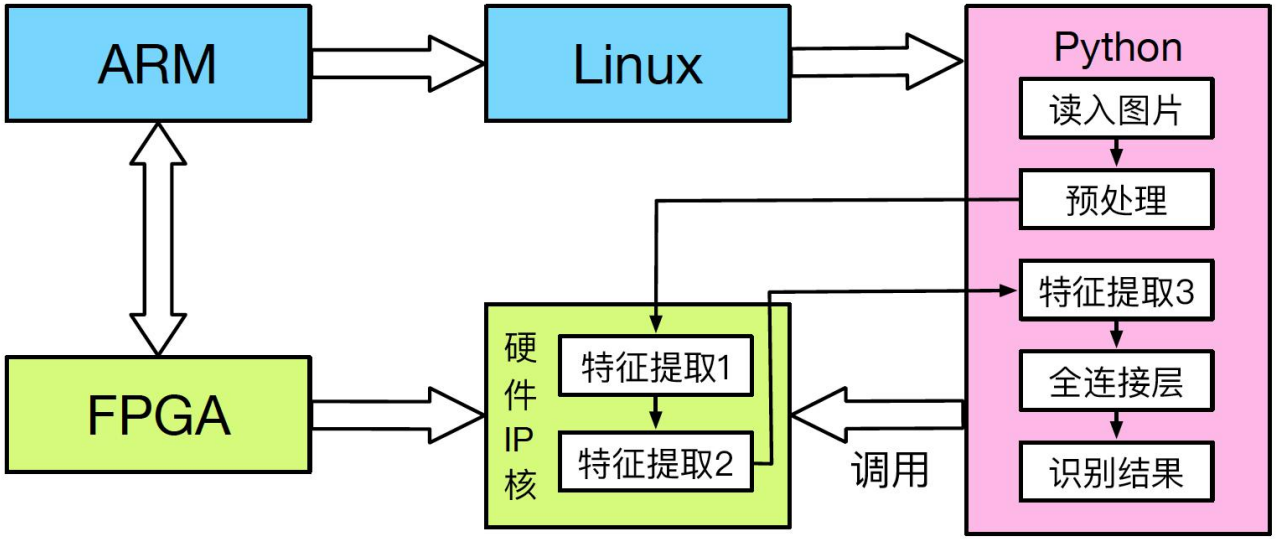


使用PYNQ-Z1开发板完成系统开发

交通标志识别系统

■ 系统集成

- 系统通过ARM+FPGA协同运行。
- PYNQ将ARM处理器与FPGA器件的底层交互逻辑完全封装起来，顶层封装使用Python，使用时只需导入对应的模块名称即可导入对应的硬件模块，进行底层到上层数据的交互或者为系统提供硬件加速。



交通标志识别系统

- 实现结果

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
```

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
```

ARM+FPGA	ARM
49463微秒	812910微秒

交通标志识别系统

- 实现结果

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	812910 μs	2459037 μs	1.22 张
ARM+FPGA	49463 μs	142478 μs	20.84 张

- 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+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。
- 识别速度：50ms/张。基本上接近实际反应速度。

大 纲

- 智能系统
- 规范建模
- 协同优化设计
- 交通标志识别系统
- 自动停车系统

自动停车系统

有人驾驶车辆：

- 停车是一个复杂的事情
- 会受到车辆本身的重量和路面情况干扰
- 这些干扰导致停车时制动策略不同，特别行车速度的不同。

无人驾驶车辆：

- ✓ 自动停车系统：
- ✓ 依据车辆自身的重量和路面情况，实时选择合适的停车策略
- ✓ 保证行车舒适和安全。

路面	附着系数
沥青或混凝土（干）	0.675
沥青（湿）	0.525
沥青土（湿）	0.600
砾石	0.450
土路（干）	0.510
土路（湿）	0.413
雪（压实）	0.150
冰	0.075

汽车重量1500公斤

自动停车系统

✓时空一致性规范语言STeC

- 环境“土路（湿）：0.413”下，制动初速度80km/h，以制动距离模型制动

$$\begin{cases} \dot{s}(u)_p = -3.37u^2 + 22.22 \\ \dot{v}(u)_p = -6.75u \end{cases} \qquad \begin{cases} \dot{s}(u)_p = -4.05u + 23.44 \\ \dot{v}(u)_p = -4.05 \end{cases}$$

- 环境变成“沥青（湿）：0.525”，制动初速度80km/h，制动距离模型制动

$$\begin{cases} \dot{s}(u)_p = -4.29u^2 + 22.22 \\ \dot{v}(u)_p = -8.58u \end{cases} \qquad \begin{cases} \dot{s}(u)_p = -5.15u + 23.77 \\ \dot{v}(u)_p = -5.15 \end{cases}$$

自动停车系统

- 环境变成“雪（压实）：0.150”，制动初速度80km/h，制动距离模型制动

$$\begin{cases} \dot{s}(u)_p = -1.23u^2 + 22.22 \\ \dot{v}(u)_p = -2.45u \end{cases} \qquad \begin{cases} \dot{s}(u)_p = -1.47u + 22.66 \\ \dot{v}(u)_p = -1.47 \end{cases}$$

- 环境“雪（压实）：0.150”下，制动初速度变为49km/h，制动模型制动

$$\begin{cases} \dot{s}(u)_p = -1.23u^2 + 13.61 \\ \dot{v}(u)_p = -2.45u \end{cases} \qquad \begin{cases} \dot{s}(u)_p = -1.47u + 14.05 \\ \dot{v}(u)_p = -1.47 \end{cases}$$

自动停车系统

- 安全停车方案

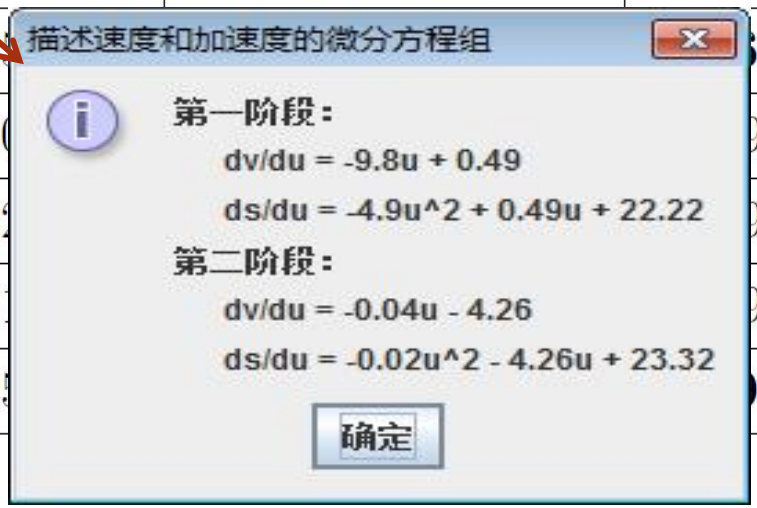
路面材质与路面情况	路面附着系数	制动初速度 (km/h)	行驶距离 (m)
沥青土（湿）	0.600	80	62.59
沥青（湿）	0.525	80	62.59
土路（干）	0.510	80	62.59
砾石	0.450	80	**62.59**
沥青土（湿）	0.600	100	95.75
沥青（湿）	0.525	100	95.75
土路（干）	0.510	100	95.75
砾石	0.450	100	**95.75**

不同环境下，相同制动初速度，调整运行曲线，确保安全停车

自动停车系统

- 安全停车方案

路面材质与路面情况	路面附着系数	制动初速度 (km/h)	行驶距离 (m)
沥青土（湿）	0.600	80	62.59
沥青（湿）	0.525	80	62.59
土路（干）	0.510	80	62.59
砾石	0.4		2.59
沥青土（湿）	0.6		95.75
沥青（湿）	0.5		95.75
土路（干）	0.5		95.75
砾石	0.4		95.75



不同环境下，相同制动初速度，调整运行曲线，确保安全停车

自动停车系统

- 安全停车方案

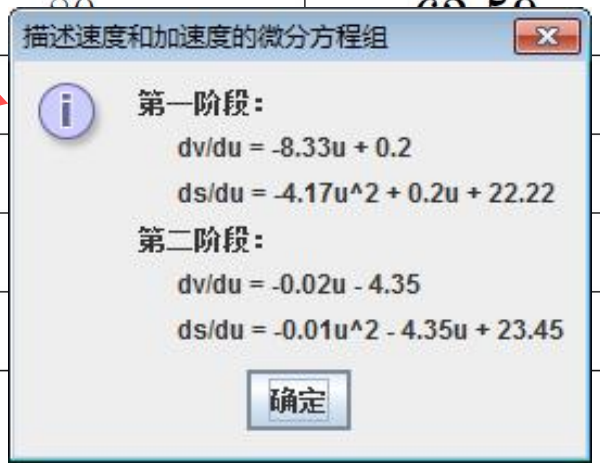
路面材质与路面情况	路面附着系数	制动初速度 (km/h)	行驶距离 (m)
沥青土（湿）	0.600	80	62.59
沥青（湿）	0.525	80	62.59
土路（干）	0.510		
砾石	0.450		
沥青土（湿）	0.600		
沥青（湿）	0.525		
土路（干）	0.510		
砾石	0.450		

不同环境下，相同制动初速度，调整运行曲线，确保安全停车

自动停车系统

- 安全停车方案

路面材质与路面情况	路面附着系数	制动初速度 (km/h)	行驶距离 (m)
沥青土（湿）	0.600	80	62.59
沥青（湿）	0.525	80	62.59
土路（干）	0.510	80	62.59
砾石	0.450		
沥青土（湿）	0.600		
沥青（湿）	0.525		
土路（干）	0.510		
砾石	0.450		



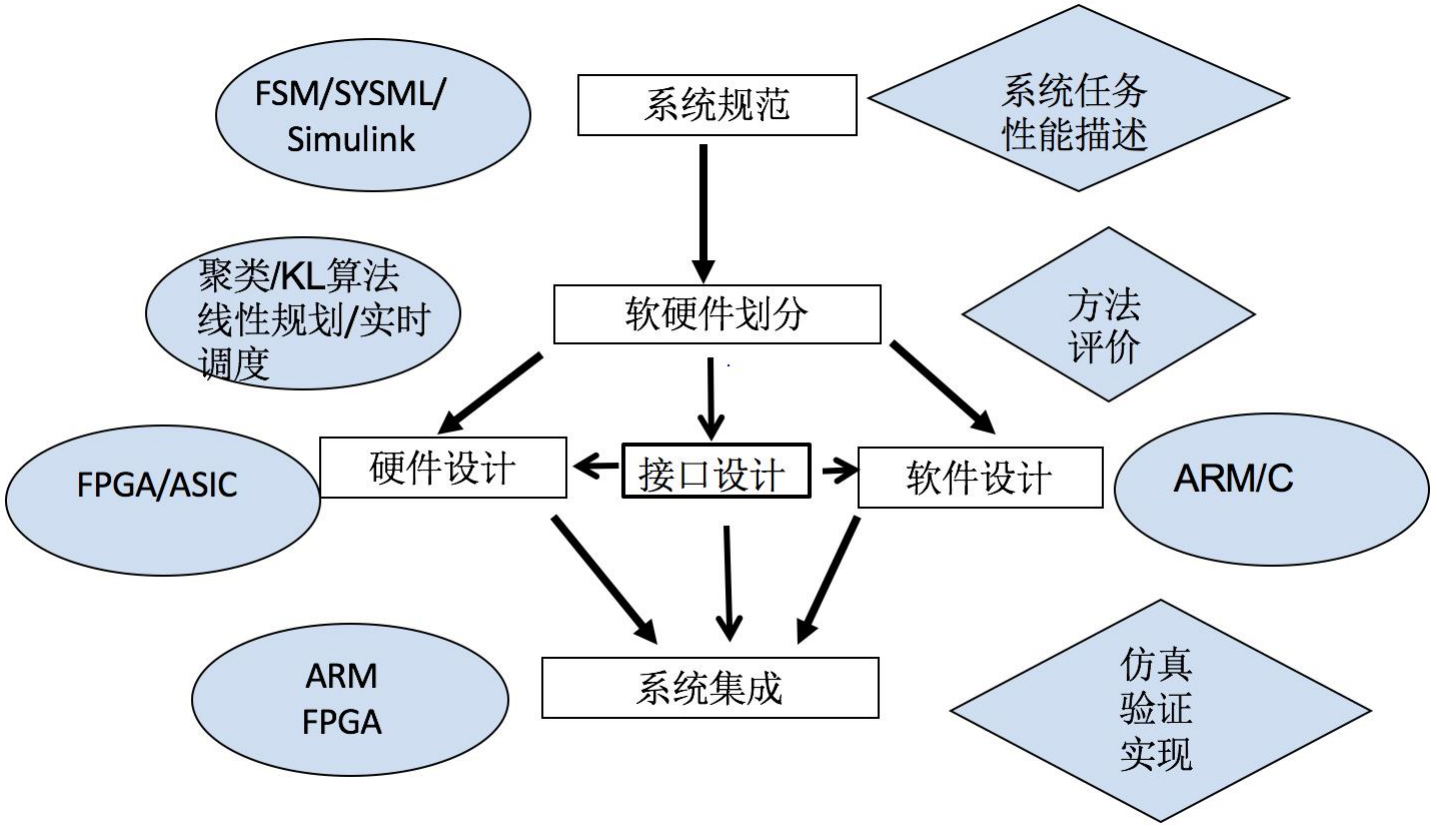
不同环境下，相同制动初速度，调整运行曲线，确保安全停车

总结

- 智能系统
- 规范建模
- 协同优化设计
- 基于BNN的交通标志识别系统
- 自动停车系统

总结

智能系统优化协同设计体系---一本教材



谢 谢