北方工业大学
NORTH CHINA UNIVERSITY OF TECHNOLOGY

5月28日, 2024 | 北京

HDL Coder / Simulink Coder 在电机控制中的应用

董哲, 教授，北方工业大学

MATLAB EXPO

运动控制系统设计与实现存在的问题：

- 1. 如何在实物控制中实现复杂控制策略？
- 2. 如何整合与协同多种平台多种接口？
- 3. 如何保证复杂控制算法运行实时性？
- 4. 如何通过总线将更多设备接入系统？

如何实现复杂控制策略？

基于 HDL Coder 和 Simulink Coder 的双电机协同控制

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-项目需求

高精度 01

红外设备需要对目标进行精确观测，这就要求电机控制精度非常高，以确保图像的清晰和稳定。

快速响应 02

红外系统主要应用特殊场景，需要快速对目标进行锁定和跟踪，这就要求双电机系统具有快速的响应能力。

稳定性 03

红外设备运行时会受到各种外力的影响，因此需要电机控制系统具有良好的稳定性和抗干扰能力。

协同工作 04

两个电机需要协同工作，以实现复杂的运动控制，如变倍、调焦。

能耗与效率 05

红外设备对能耗有严格的限制，因此电机控制系统需要在保证性能的同时，优化能耗。

降成本 06

原方案由 FPGA+DSP 实现联动控制，硬件成本较高，现将DSP控制器优化掉，达到节省成本需求。

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-研究意义

01

提升可靠性：

HDL Coder能够从MATLAB和Simulink模型自动生成可读、可追溯、可综合的Verilog代码，减少了人工编写代码的错误，同时可以反复迭代，优化算法。

02

提高开发效率：

HDL Coder允许开发者使用高级的Simulink模型来代替传统的硬件描述语言（HDL）编程，这样可以减少编程时间，提高开发效率。

03

优化硬件资源：

HDL Coder提供了多种优化选项，允许开发者在生成代码前探索不同的硬件架构，优化资源使用，实现更高效的硬件设计。

04

确保设计质量：

通过在Simulink环境中进行早期验证和仿真，可以在硬件实现之前发现并修正设计中的问题，从而提高最终产品的质量。

05

促进多学科协作：

Simulink作为一个多学科通用平台，可以促进不同背景的工程师之间的沟通与协作，共同开发复杂的系统。

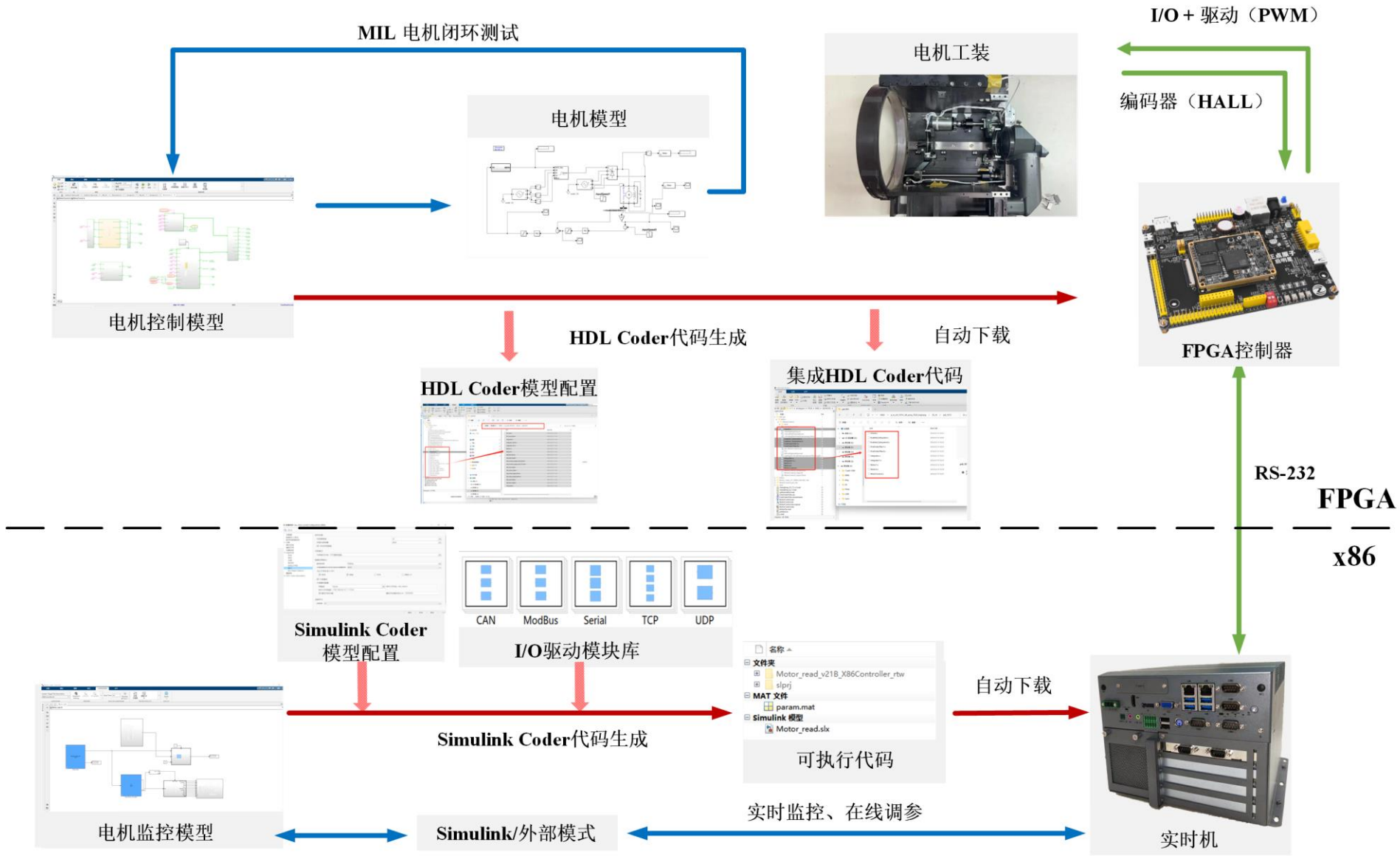
06

降低技术门槛：

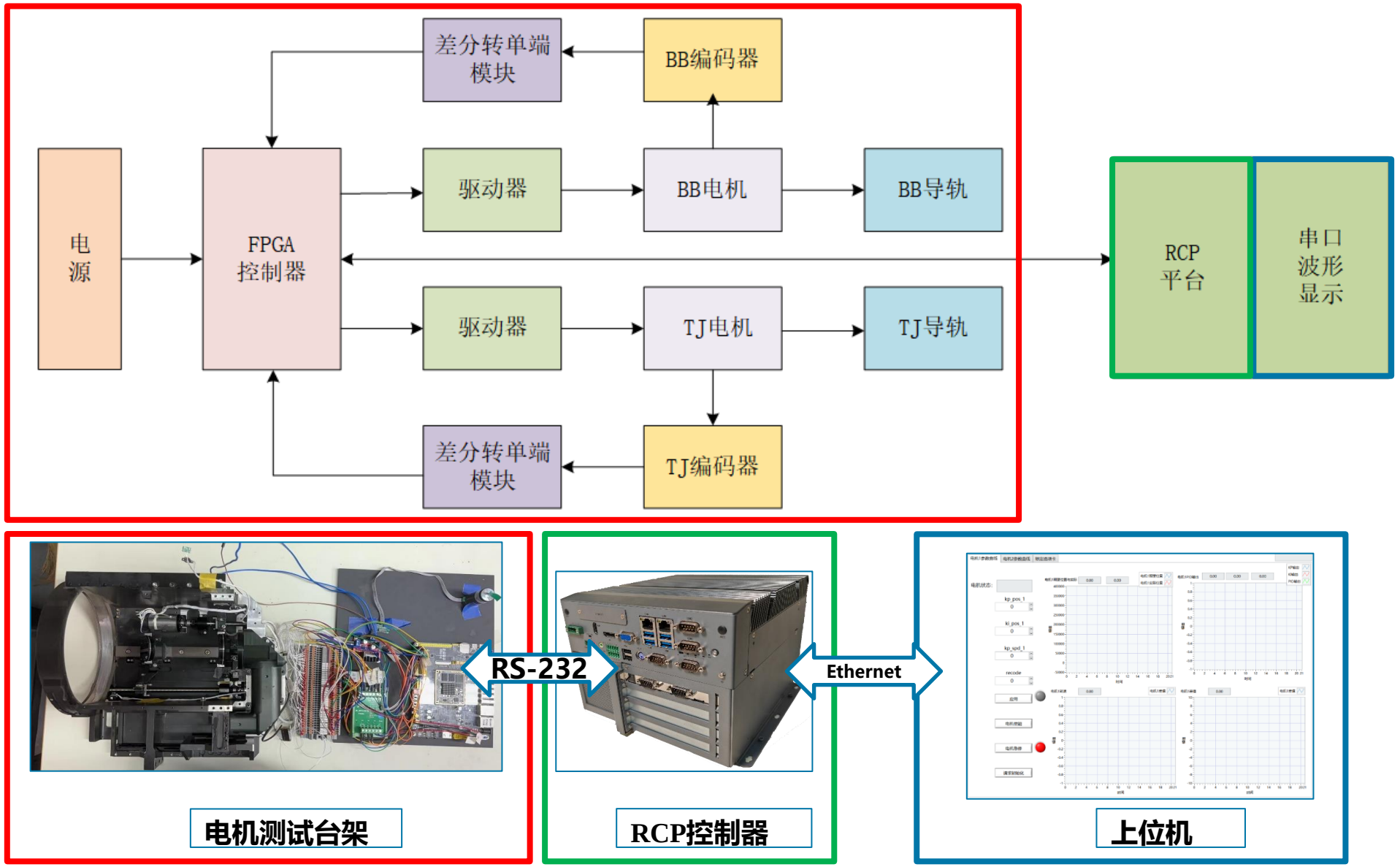
对于那些不熟悉硬件描述语言的工程师和研究人员，HDL Coder提供了一种更直观的设计方法，降低了进入FPGA开发领域的技术门槛。

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-解决方案

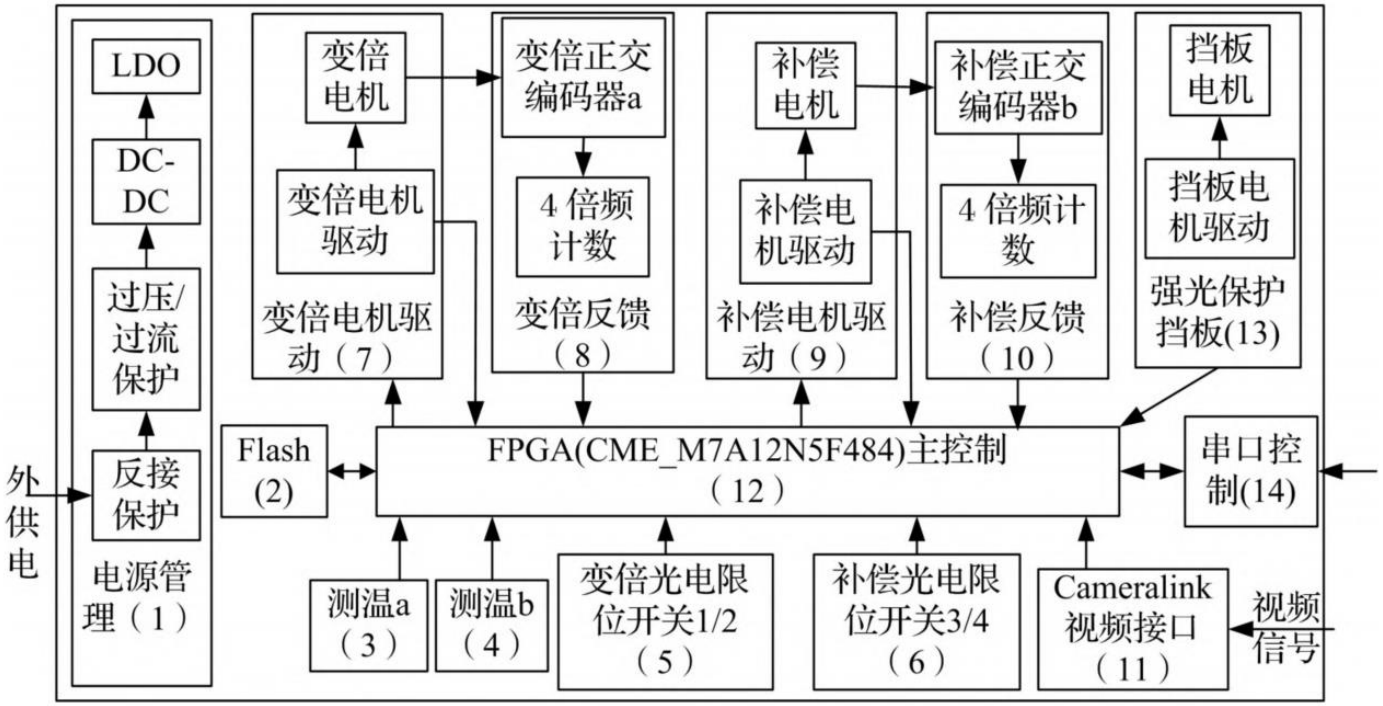
双电机协同控制架构



基于 HDL Coder 和 Simulink Coder 的双电机协同控制-硬件架构



基于 HDL Coder 和 Simulink Coder 的双电机协同控制-硬件原理



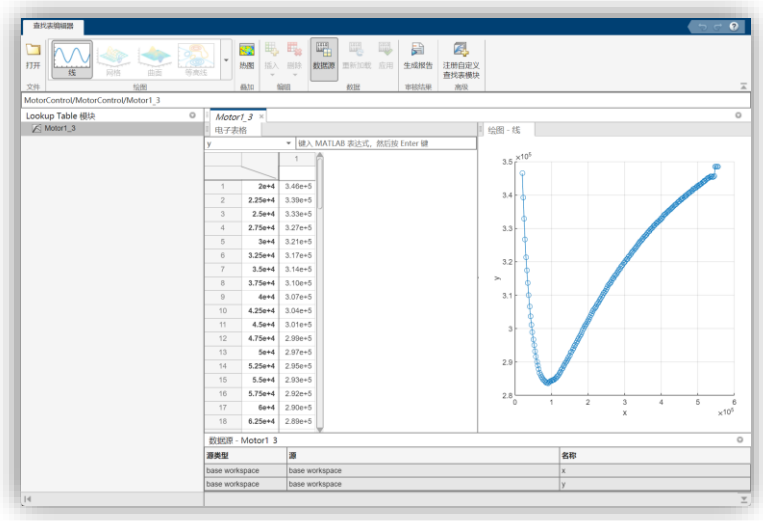
连续变焦红外镜头主从控制硬件原理框图

连续变焦红外镜头整体电路主要包括以下几个部分：

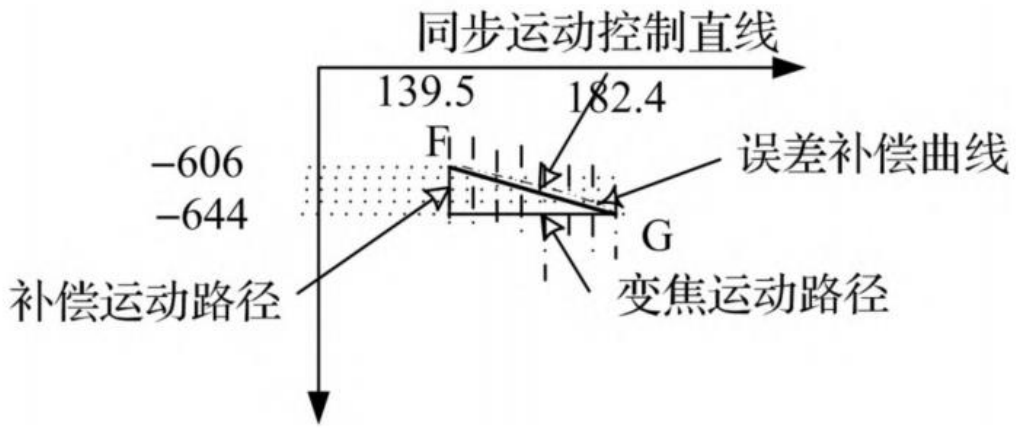
- 电源管理部件
- 主控制部件
- 驱动与反馈部件
- 限位保护部件
- 视频与通信接口部件
- 外壳与结构件等

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-算法设计

在进行连续变焦红外镜头主从控制时，考虑双电机预测同步补偿方法，通过FPGA 并行计算方式，控制变焦与调焦的同步运动，合成主从电机轨迹，同时将高低温下测试得出的补偿点作为拟合线的端点，结合电机轨迹简化成右图运动路径规划曲线，保证在变焦的过程中，实时成像清晰，且不受温度变化的影响。

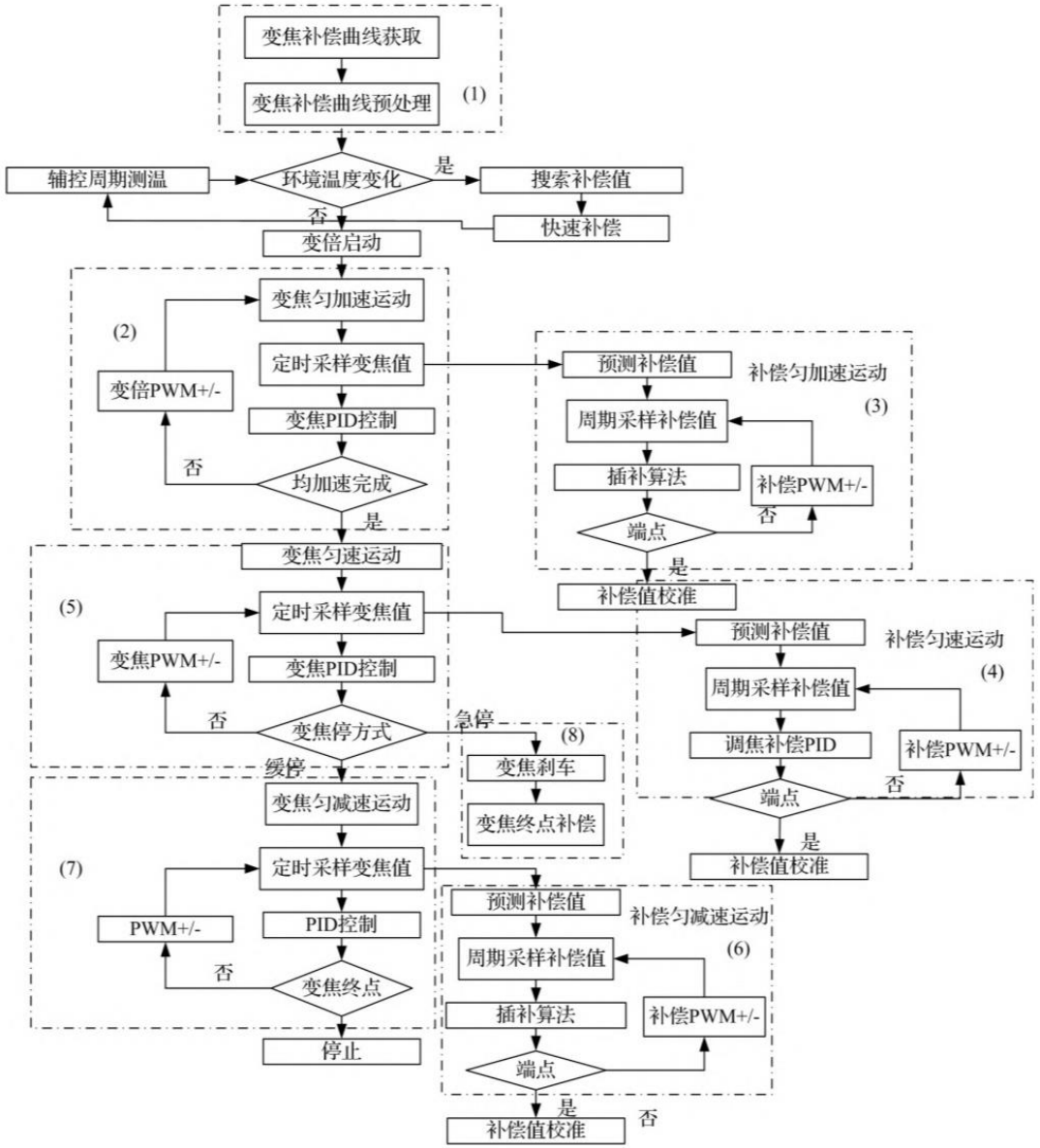


主从电机轨迹



拟合路径规划曲线图

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-算法设计



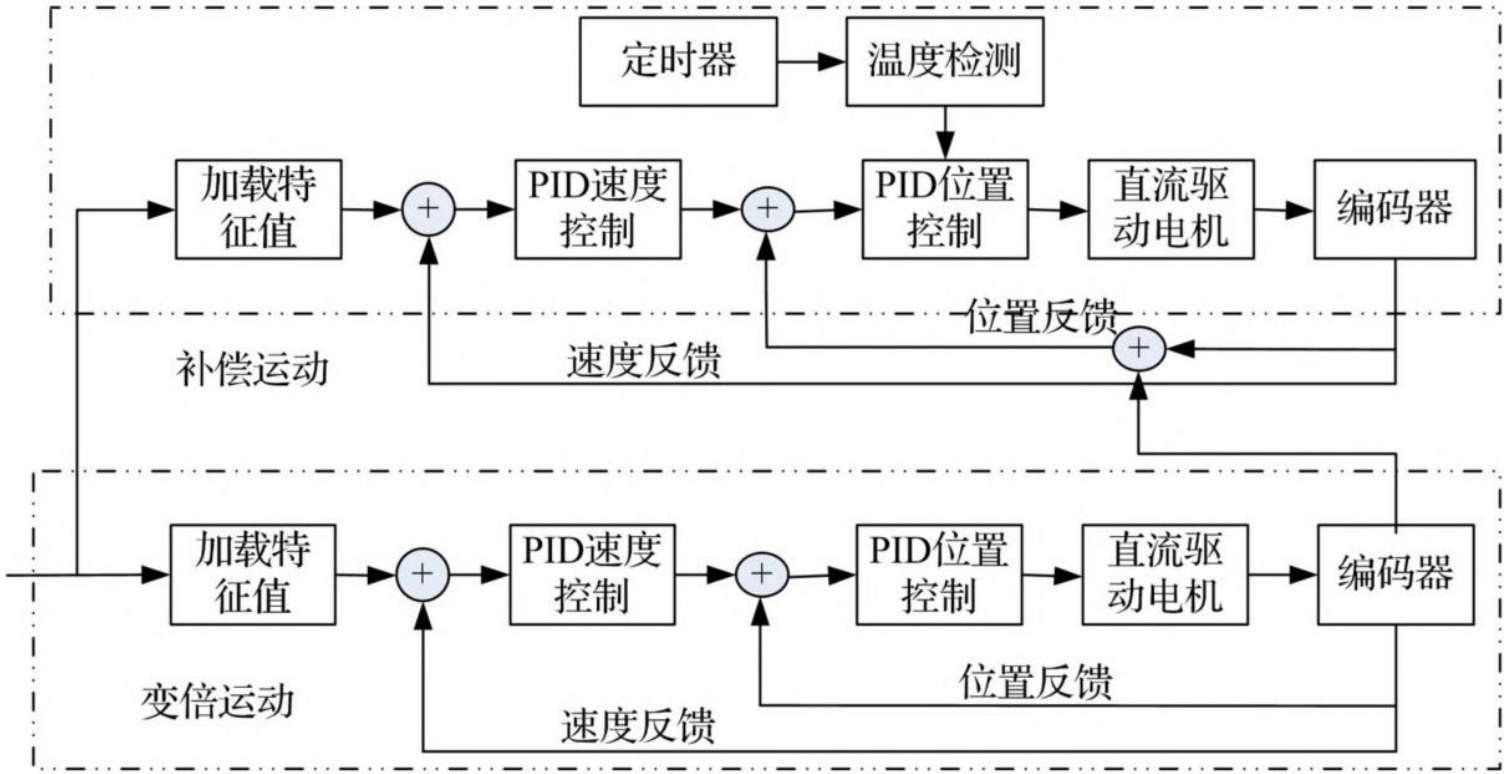
连续变焦红外镜头同步补偿控制执行流程图如左图所示。

变焦运动采用 PID 速度环控制算法，按系统设定的变焦速度匀速运动，在变焦停止时，变焦采用刹车方式，快速停止。补偿运动采用速度环与位置环串行 PID 控制，速度环用于匹配变焦运动，而位置环主要用于两段拟合直线接合处，为减小因速度变化产生的冲击和振动，在变焦停止点处，调整接合点处的相应位置。

由于变焦运动是由系统给定的基准运动，因此该匀速运动不需要关联其他的运动。而补偿运动是根据变焦运动的速度及拟合补偿运动直线的斜率来重新计算不同补偿段的速度，因此，补偿运动的速度环是一个从属运动。

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-算法设计

PID控制器



变焦有两种控制方式：

1）随机变焦控制：

随机控制，由于没有变焦位置值，PID 采用速度单环控制；

2）固定位置变焦控制：

固定位置控制，PID 采用速度和位置双环控制，位置环只是在终点时作为主控制环使用。

变焦运动与补偿运动采用主从耦合控制方式：变焦运动为主运动，而补偿运动为从运动。

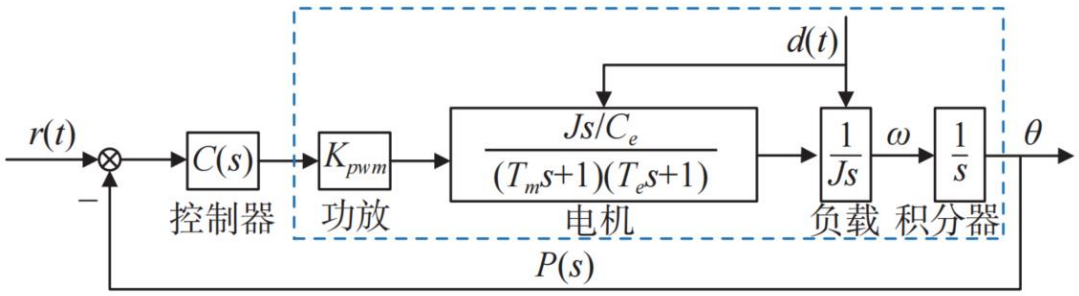
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-算法设计

分数阶PID控制器

$$C_{\mathrm{FOPID}}(s)=k_{\mathrm{p}}+k_{\mathrm{i}}\frac{1}{s^{\lambda}}+k_{\mathrm{d}}s^{\mu}$$

式中：k_{p}、k_{i}、k_{d} 分别为控制器的比例、积分与微分作用系数，λ和μ为积分微分项的阶次；取值范围（0,2）；分数阶 PID 控制器与传统 PID 控制器相比多了2个参数，这种扩展提高了控制 器的灵活性，但增加了控制器整定的难度。

$$C_{\mathrm{FOPID}}(s)=\frac{C_e(T_e+T_m)}{K_{pwm}\eta}+\left(\frac{C_e}{K_{pwm}\eta}\right)\frac{1}{s^{\lambda}}+\frac{C_eT_eT_m}{K_{pwm}\eta}s^{\mu}$$



变焦（补偿）伺服控制系统数学模型

在整个运动中，变焦开始和结束阶段速度变化很大，为了更好地分析变焦控制系统的性能，需建立控制系统数学模型。

变倍组与补偿组的控制电机均采用精度高、响应快的直流电机，电机输出转矩/输入电压的传递函数如下：

$$\frac{M_m(s)}{U_a(s)}=\frac{Js/C_e}{T_mT_es^2+T_ms+1}\xrightarrow{T_m\gg T_e}\frac{Js/C_e}{(T_ms+1)(T_es+1)}$$

为简化分数阶 PID 控制器的整定问题，采用一种基于内模控制策略的分数阶 PID 整定方法。

$$\begin{cases}\left|L(\mathrm{j}\omega_c)\right|=\left|C_{\mathrm{FOPID}}(\mathrm{j}\omega_c)\right|\times\left|\mathrm{P}(\mathrm{j}\omega_c)\right|=1\\ \mathrm{Arg}\left[L(\mathrm{j}\omega_c)\right]=\mathrm{Arg}\left[C_{\mathrm{FOPID}}(\mathrm{j}\omega_c)\right]+\mathrm{Arg}\left[P(\mathrm{j}\omega_c)\right]=-\pi+\phi_m\\ \left|\frac{\mathrm{d}(\mathrm{Arg}(\mathrm{L}(\mathrm{j}\omega)))}{\mathrm{d}\omega}\right|_{\omega=\omega_c}=\left\|\frac{\mathrm{d}(\mathrm{Arg}(C_{\mathrm{FOPID}}(\mathrm{j}\omega)))}{\mathrm{d}\omega}\right|+\left|\frac{\mathrm{d}(\mathrm{Arg}(\mathrm{P}(\mathrm{j}\omega)))}{\mathrm{d}\omega}\right\|_{\omega=\omega_c}=0\end{cases}$$

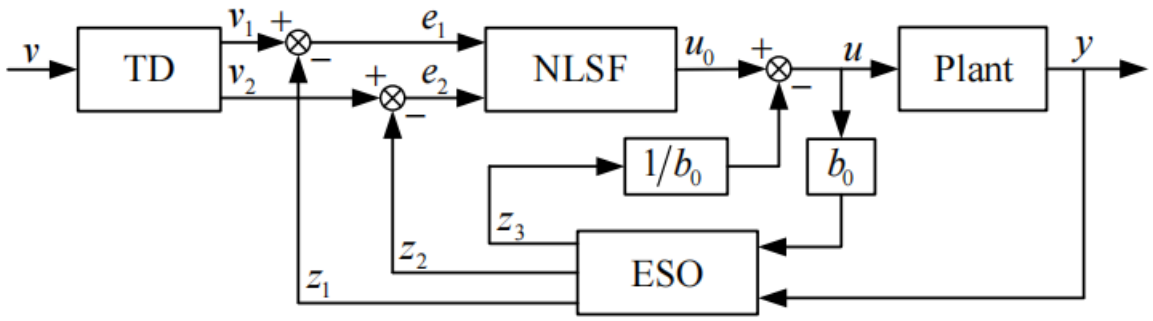
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-算法设计

ADRC控制器

在对连续变焦红外镜头电机控制时，传统控制结构存在一些问题：

（1）对独立电机控制而言，采用传统的PI控制时，PI的固有特性决定了单电机的转速动态响应曲线中会存在超调，此外，由于各台电机和控制器的差异，很难将各台控制器的动态响应特性调整到完全一致；

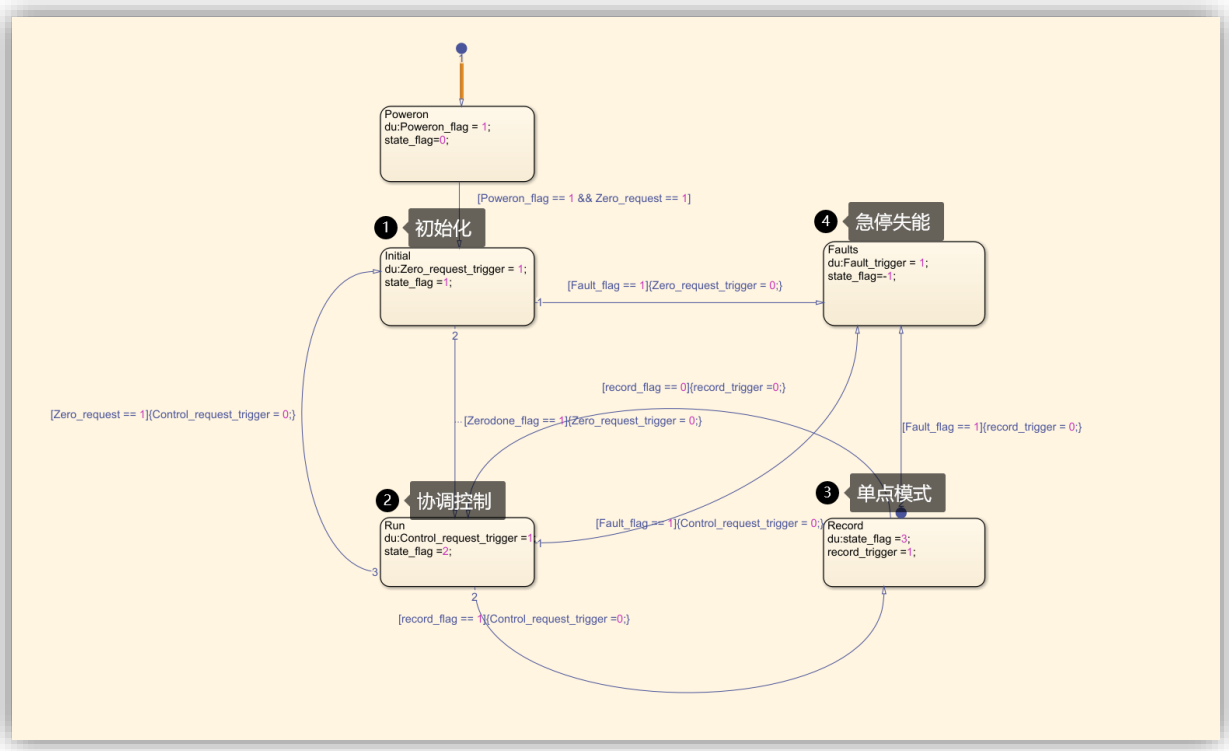
（2）对多电机系统而言，各电机之间负载，控制回路之间存在计算延时，这会导致在一定的滞后性，不利于系统的同步。



ADRC 控制结构地图

为了使多电机系统中的所有驱动单元都获得相同的起动过程，且起动过程中转速不能出现超调，针对连续变焦红外镜头的各驱动单元采用自抗扰控制（Active Disturbances Rejection Control，ADRC）方法。

基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程



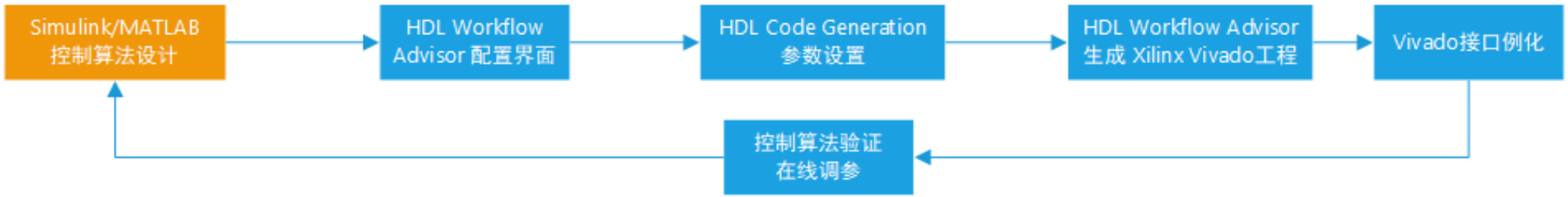
算法模型是基于Stateflow实现四状态切换：

① **初始化：**确保电机开始跟着远离初始点一段距离后再回到初始点。

② **协同控制：**双电机协同响应，从电机实时跟随主电机的位置状态。

③ **单点模式：**单点用于双电机单独控制，调试主从电机协同对应表。

④ **急停失能：**应用于电机运作突发状态，失能电机。



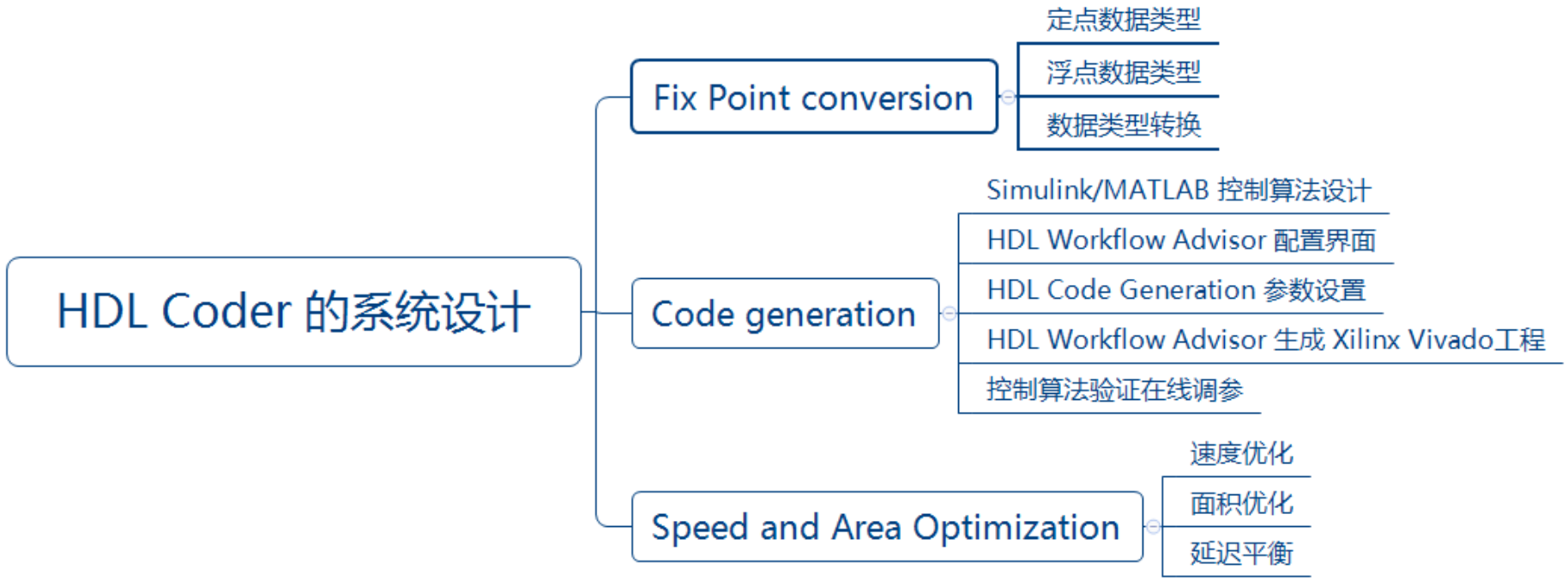
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程

滤波器近似实现

由于分数阶算子具有无限维，为了实现分数阶 PID 控制器，需使用近似算法对其微分算子进行逼近,改进 Oustaloup 滤波器传递函数为

$$s^{\alpha} \approx K\left(\frac{ds^2 + b\omega_h s}{d(1-\alpha)s^2 + b\omega_h s + d\alpha}\right)\prod_{k=-N}^{N}\frac{s+\omega_k'}{s+\omega_k}$$

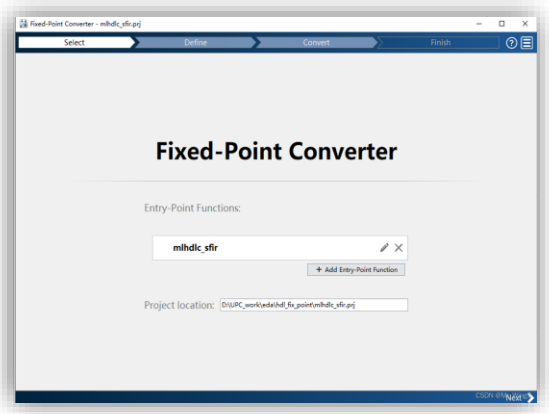
从功耗、内存使用、速度和成本的角度考量，需进行数据类型转换等工作

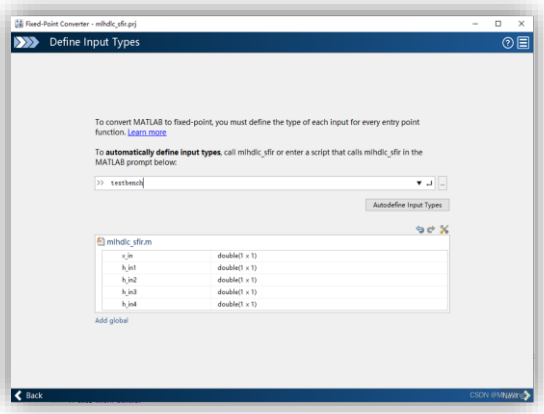


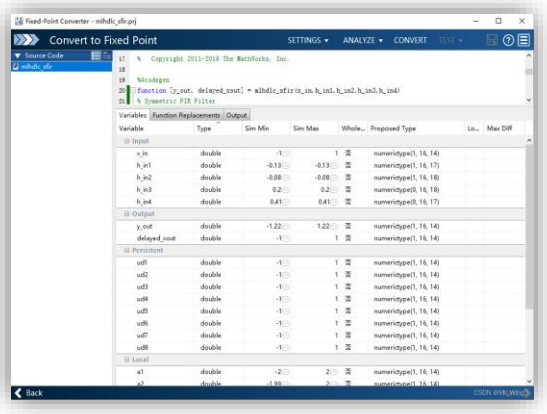
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程

数据类型转换

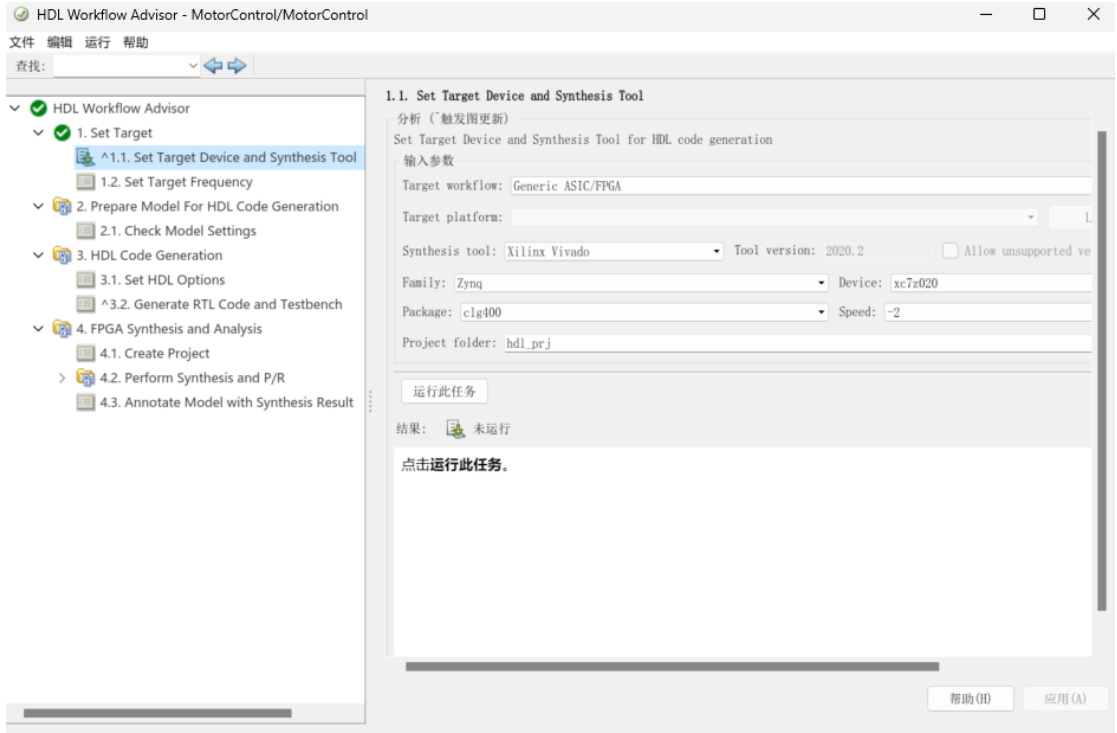
Fixed-Point Conversion 提供了一系列数据类型和工具，用于在嵌入式硬件上优化和实现定点和浮点算法。包括定点和浮点数据类型，以及特定于目标的数值设置。使用 Fixed-Point Conversion，可以执行目标感知的定点仿真。在硬件上实现设计之前，可以测试和调试量化效应，如溢出和精度损失。它还可以用于分析双精度算法，并将它们转换为精度较低的浮点或定点形式。优化工具帮助选择满足数值精度要求和目标硬件约束的数据类型。为了高效实现，可以用硬件最优模式来替代计算成本高昂的设计结构。



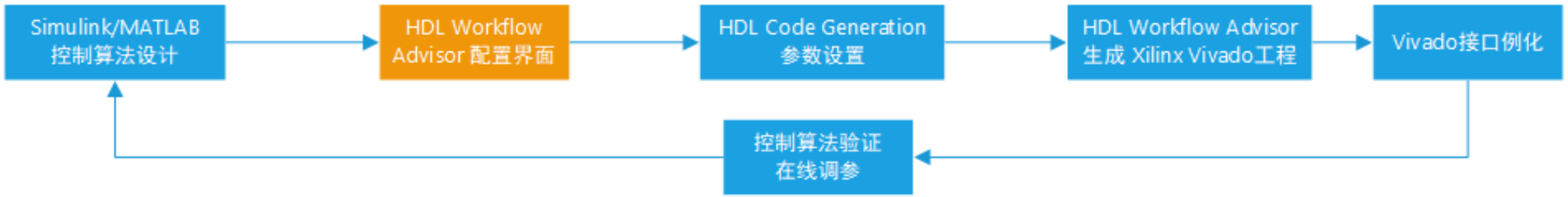




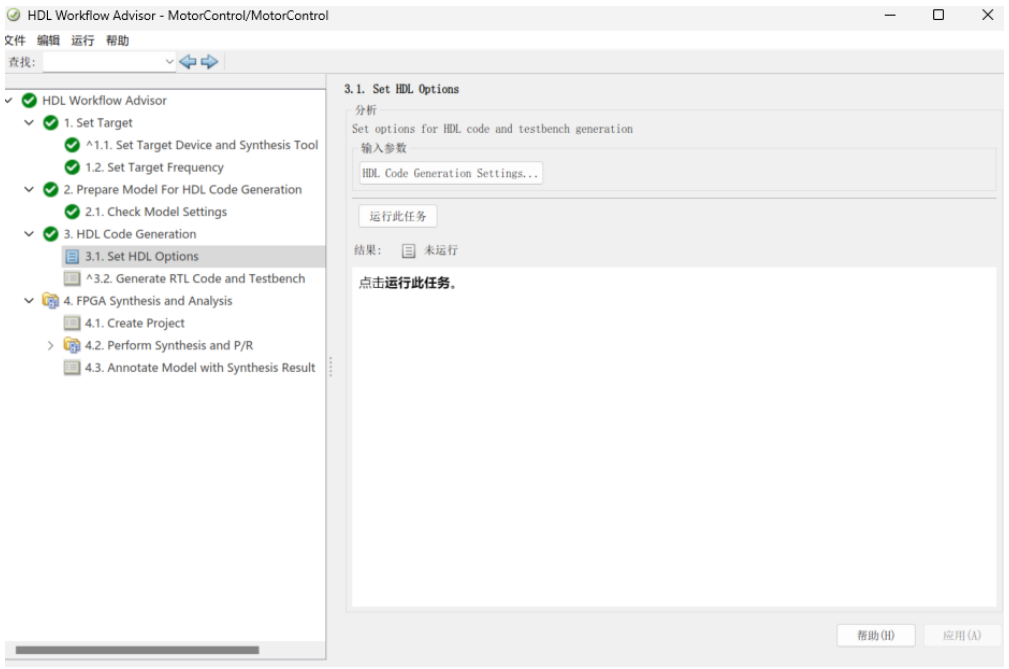
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程



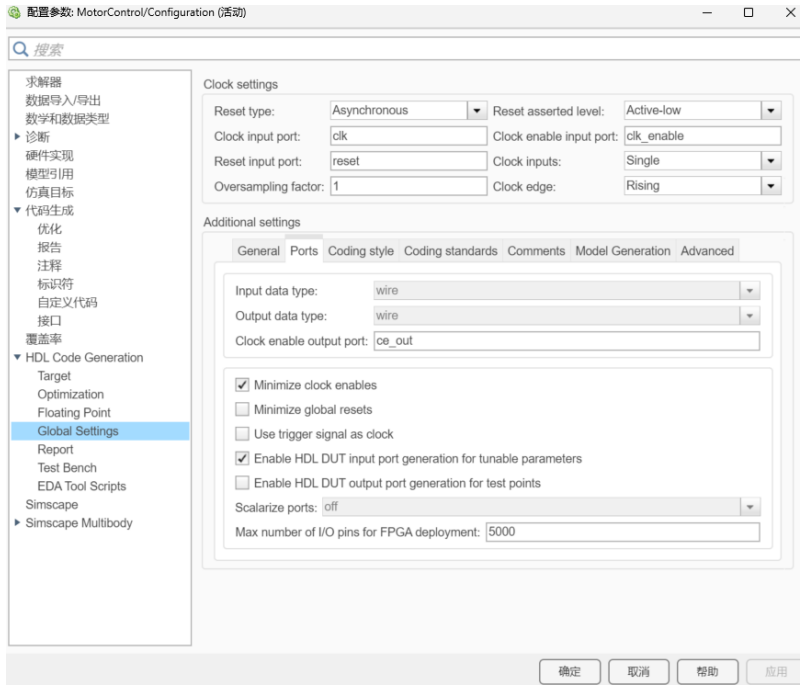
HDL Workflow Advisor初始界面图



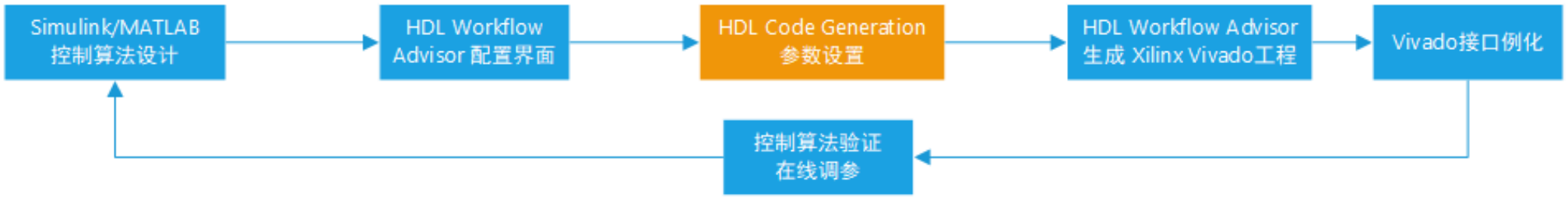
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程



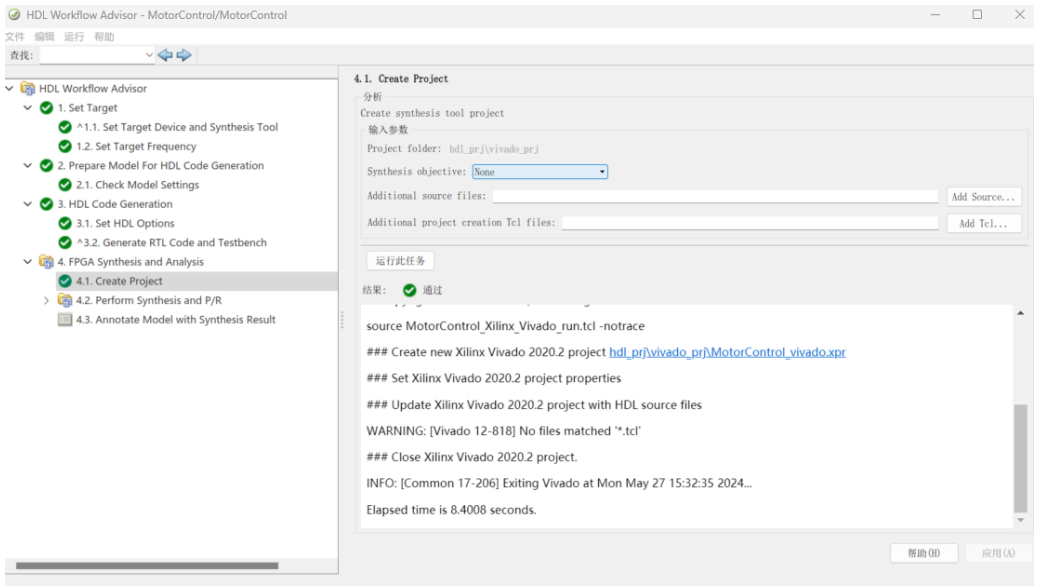
HDL Code Generation参数设置



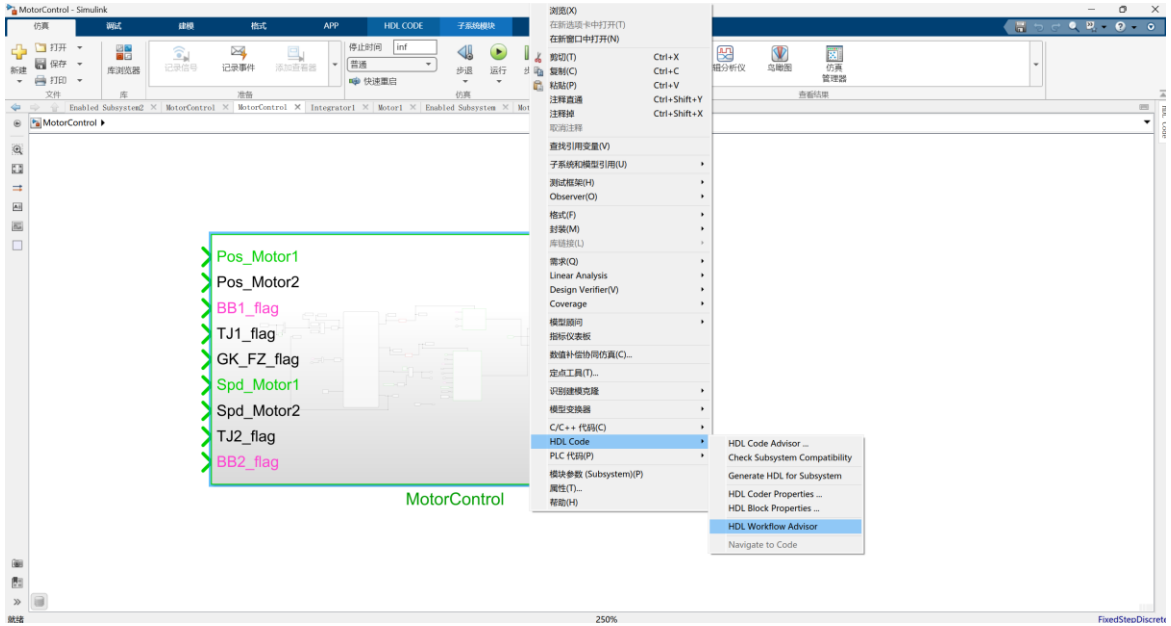
界面中包括定义输出、定点化、选择代码标签、代码生成



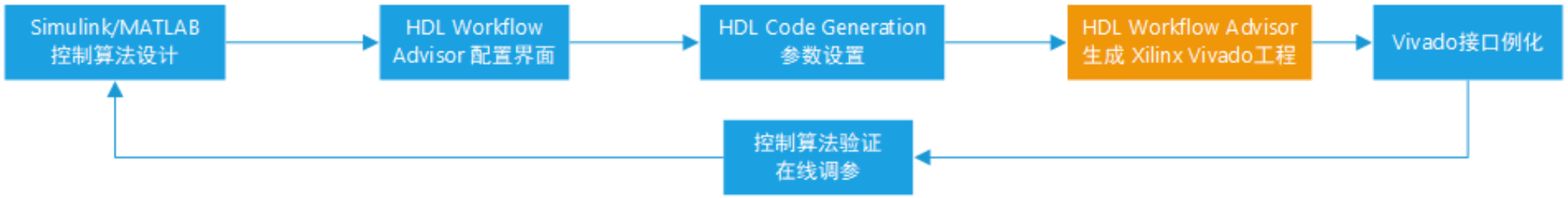
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程



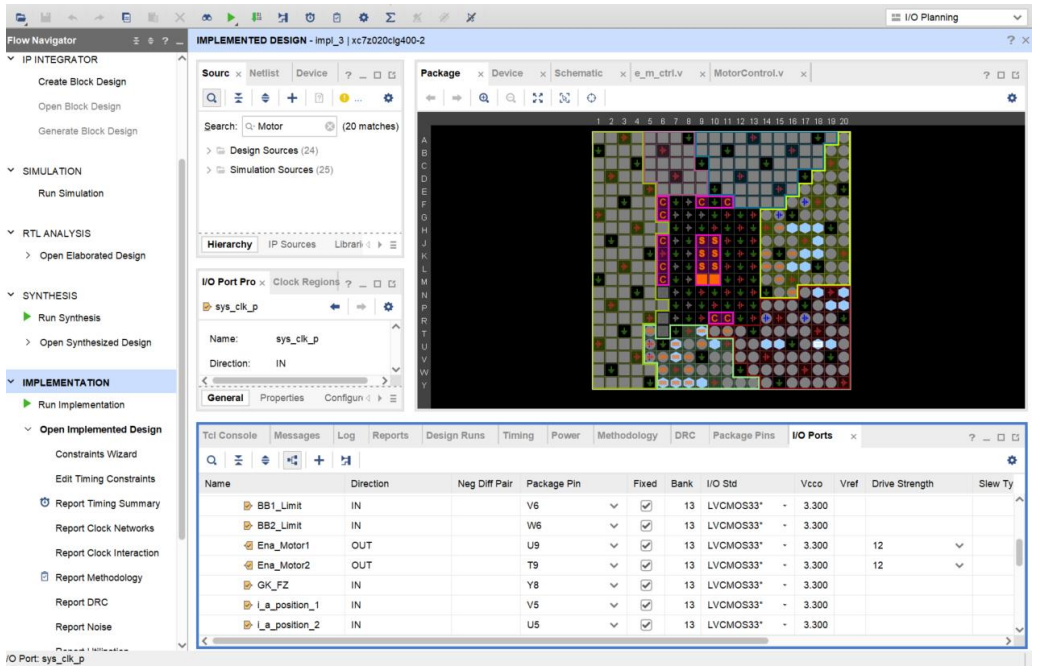
HDL Workflow Advisor 工程配置



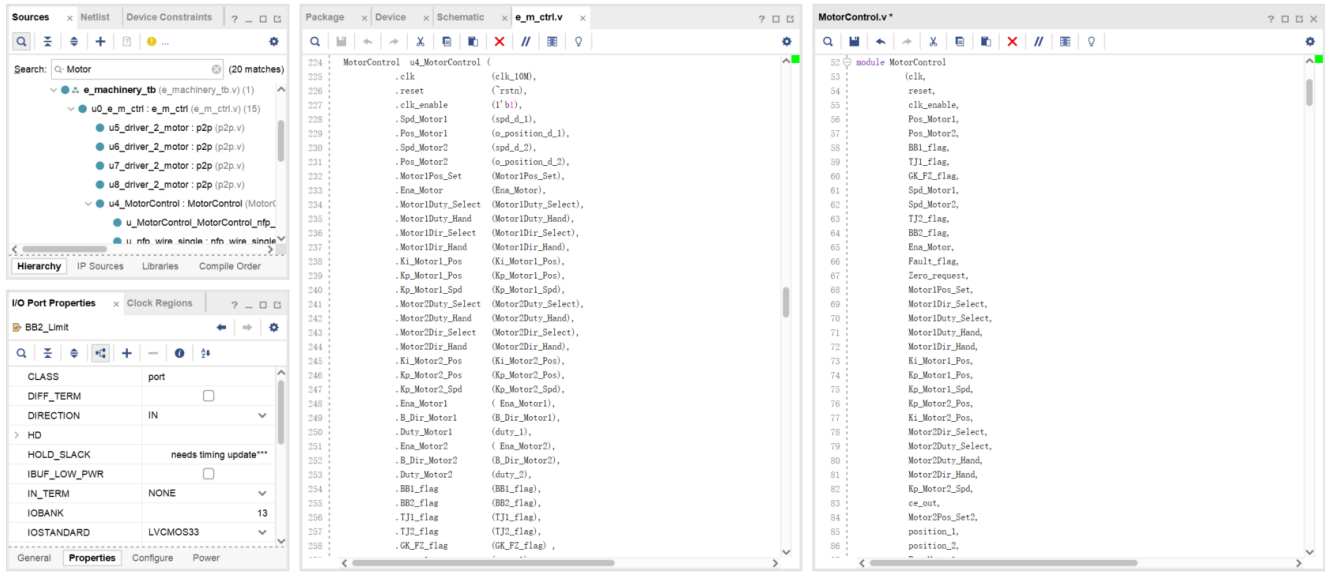
HDL Workflow Advisor 生成 Xilinx Vivado工程



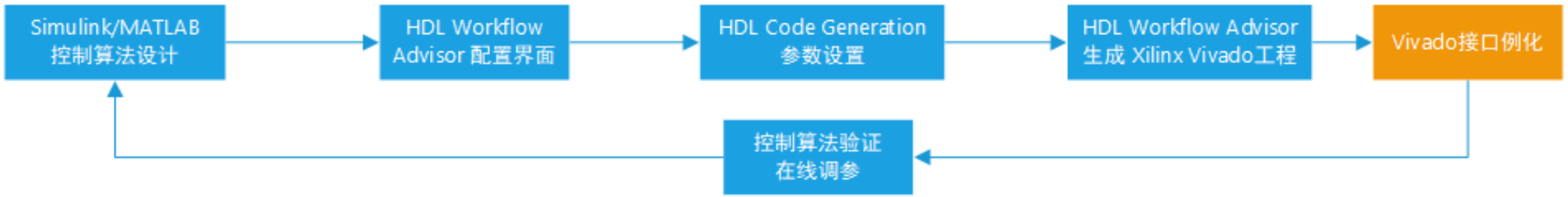
基于 HDL Coder 和 Simulink Coder 的双电机协同控制-开发流程



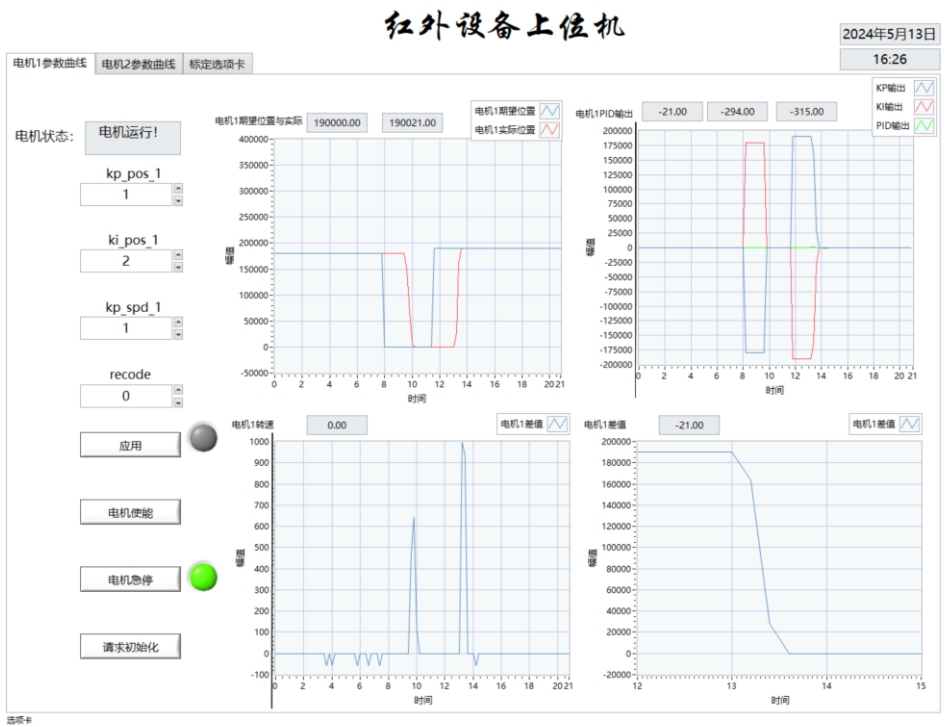
在工程门级网表，选择I/O Planning进行管脚配置



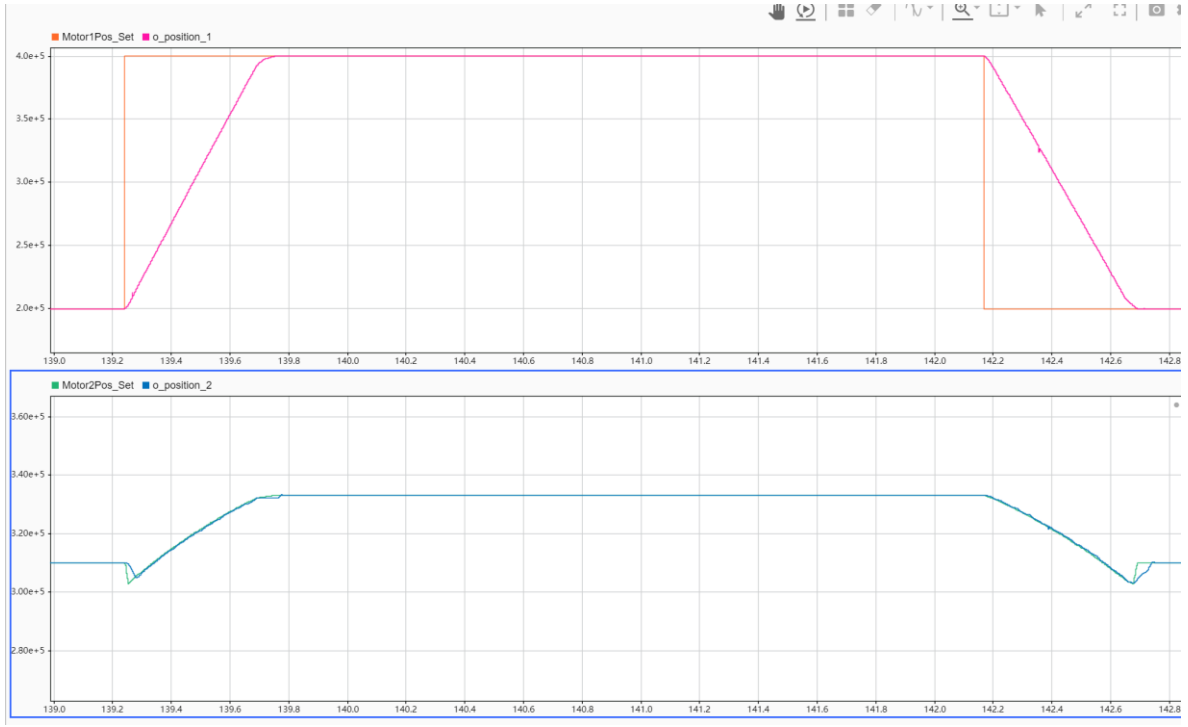
工程中添加驱动，以及顶层接口例化



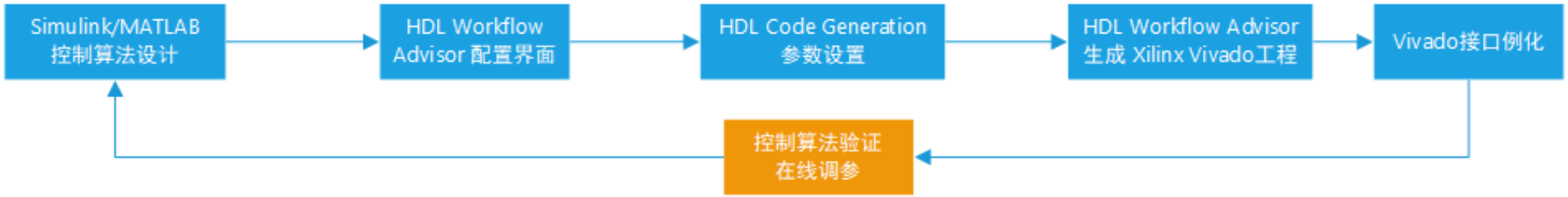
基于 HDL Coder 和Simulink Coder 的双电机协同控制-开发流程



上位机界面显示



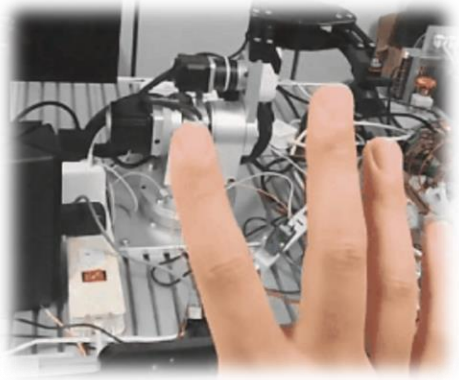
跟随曲线显示



如何整合与协同多种平台多种接口？

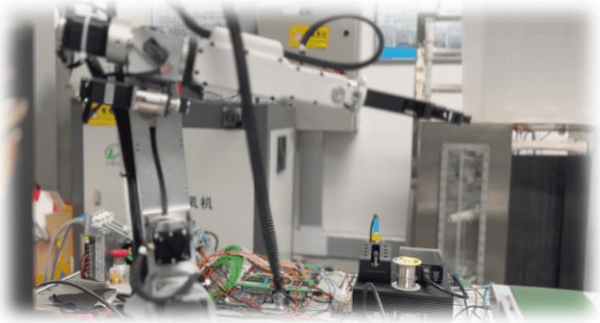
多平台与多种接口的打通与协同作用

多平台多接口整合：系统协同感知与控制框架



AI - Computer Vision
人工智能应用 视觉检测

Python
on CPU GPU



Motion Control
运动控制

C, C++, VHDL, Verilog
on MCU, CPU, FPGA, DSP

Digital Twin
数字孪生

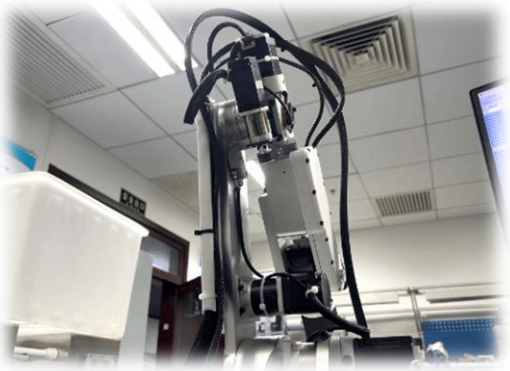
C, C#(Unity 3D), FMI
on CPU

控制需求 高算力，高实时性，高速通信…

平台接口 串口，管道通信，总线通信…

开发硬件 CPU、GPU、FPGA、DSP、MCU…

各有不同



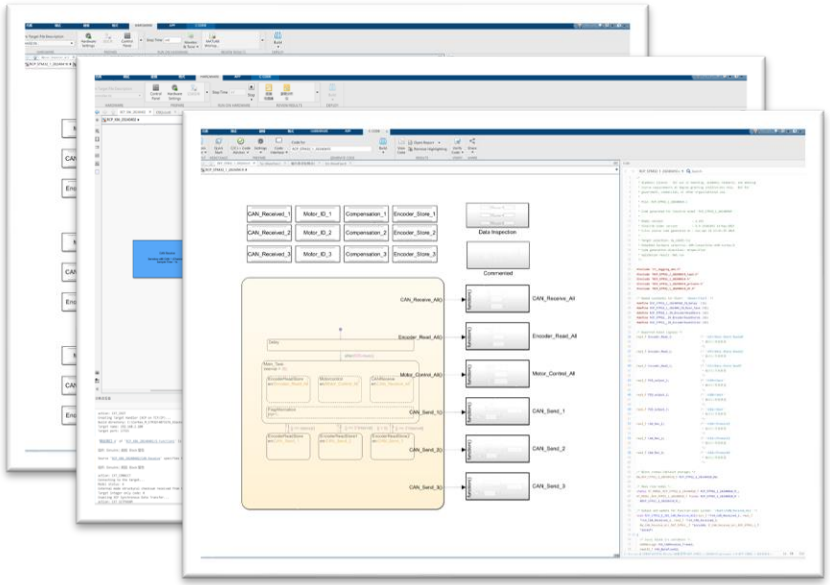
Force Sensing
力度等传感器信号采集

AIO, DIO, C, C++, m script
on CPU

多平台多接口整合：系统协同感知与控制框架

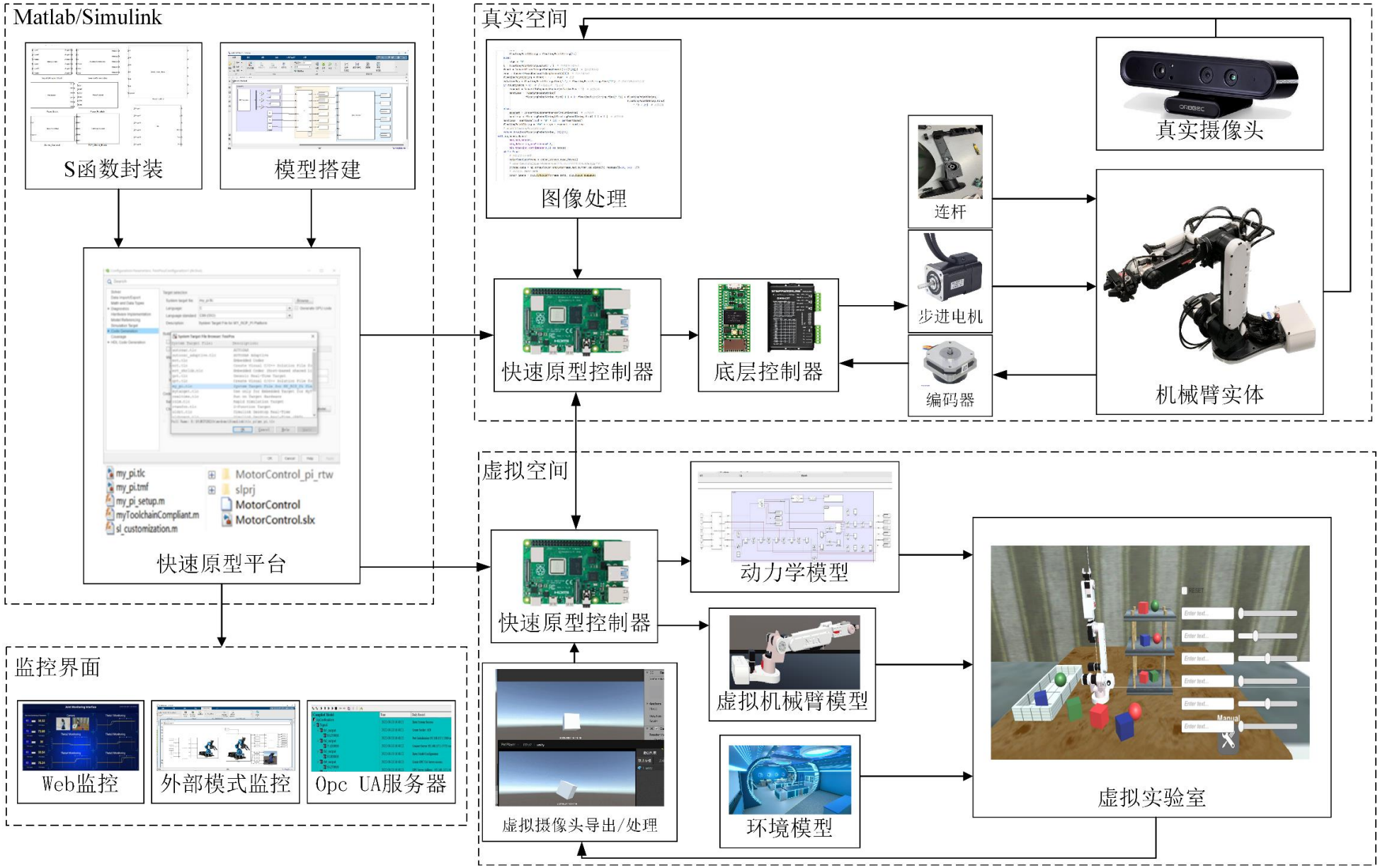
开发案例：结合 Computer Vision（CV）的机械臂控制

- 开发需求：
  - 整合CV Python应用
  - 多电机协同控制
  - 高速高效的硬件通信
- 开发助力
  - MATLAB Simulink S-Function 等
    助力RCP平台开发，打通顶层到底层
    LCM，Linux共享内存模块封装进程间通信
  - MATLAB Simulink Coder & Embedded Coder
    助力基于模型的开发（MBD），及自动代码生成
  - MATLAB Simscape
    助力被控对象建模，及MIL、RCP、HIL仿真



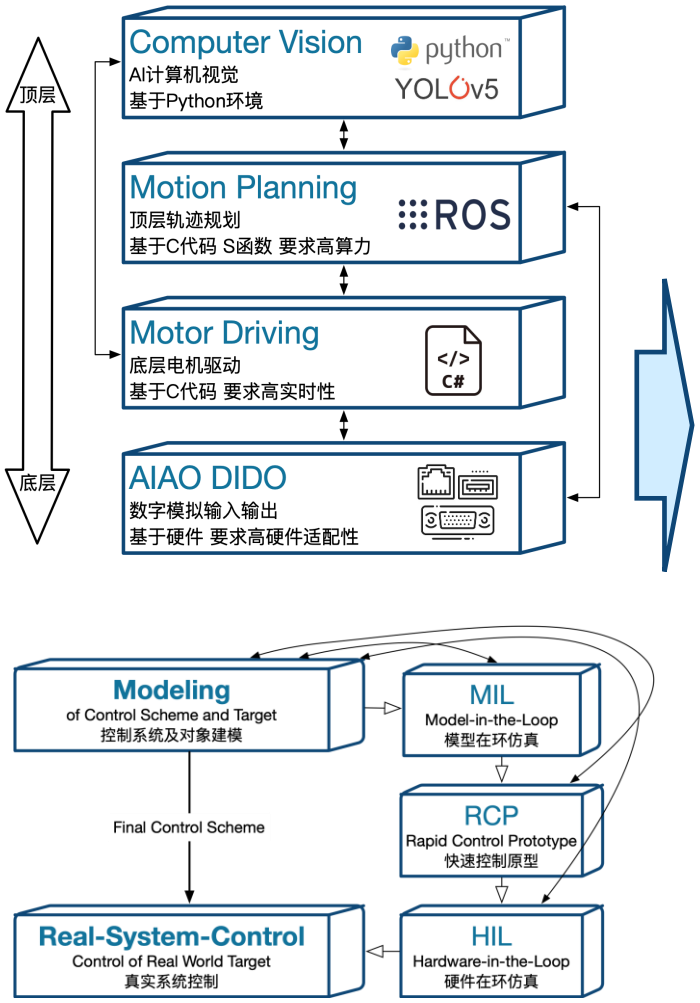
多平台多接口整合：系统协同感知与控制框架

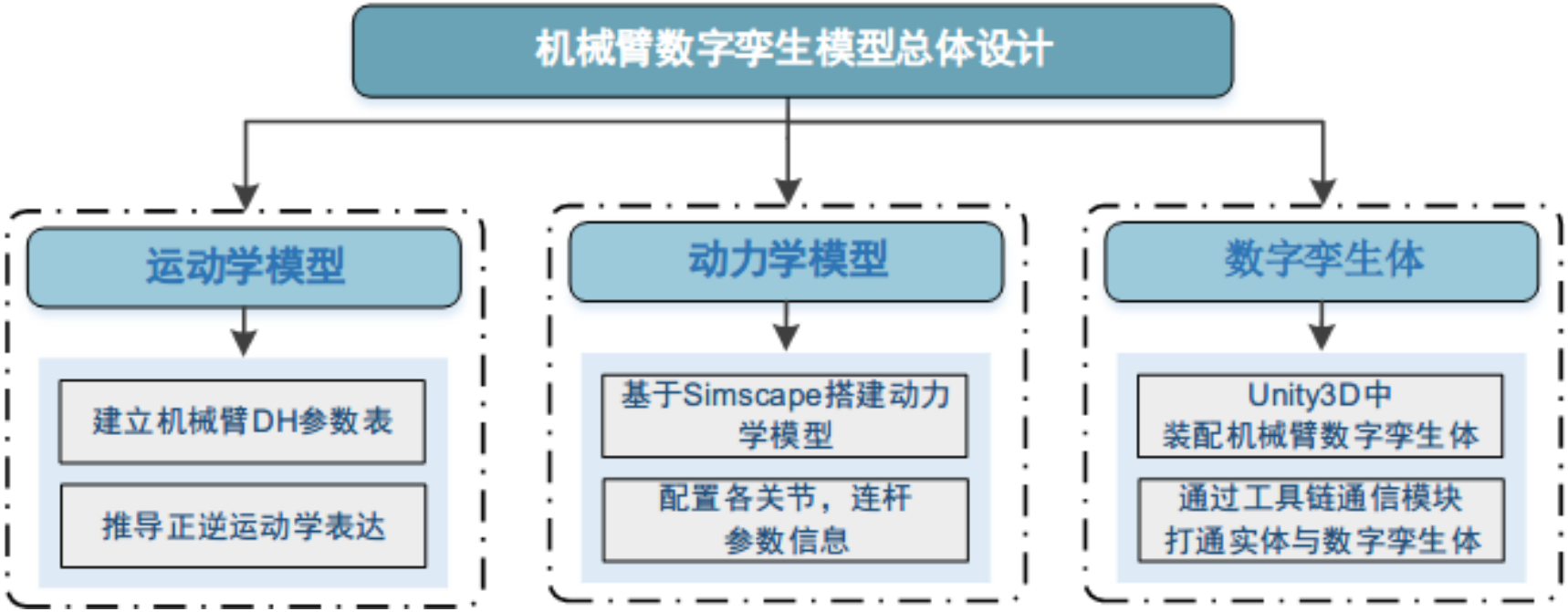
系统架构



多平台多接口整合：系统协同感知与控制框架

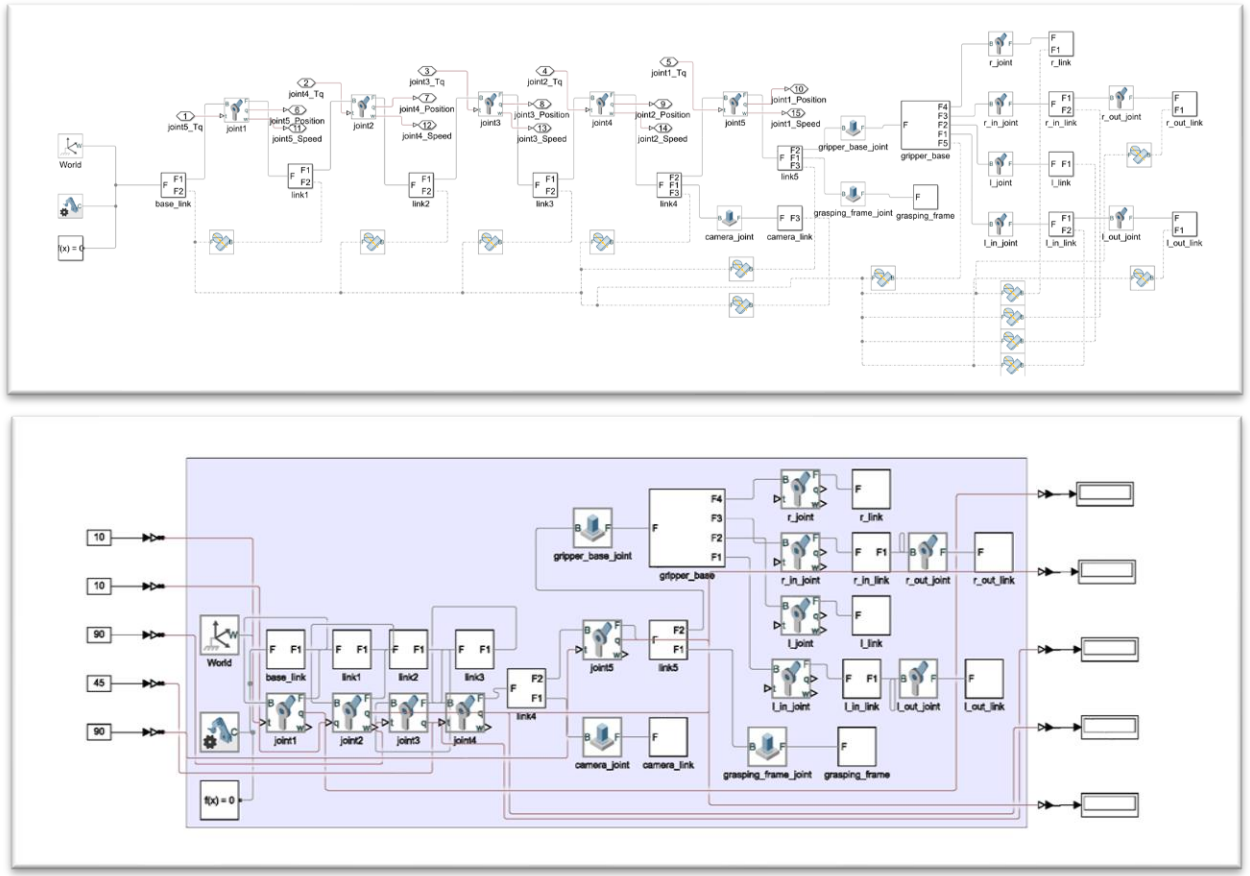
数字孪生建模





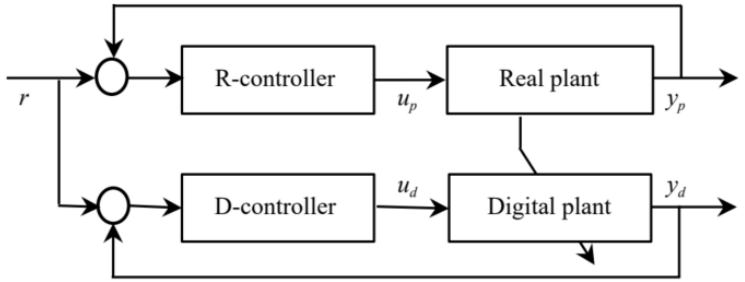
多平台多接口整合：系统协同感知与控制框架

动力学建模



Simscape机械臂动力学建模

基于系统辨识的自适应模型跟踪



系统非线性动力学模型：

$$x_p(t+1) = f\left(x_p^{[n-1]}(t), u_p^{[m]}(t), \theta_p\right)$$

$$y_p(t) = g\left(x_p^{[n-1]}(t), \varphi_p\right)$$

状态反馈及输出反馈控制器：

$$z_p(t+1) = H\left(z_p^{[q-1]}(t), y_p^{[l]}(t), r(t), h_p\right)$$

$$u_p(t) = K\left(u_p^{[m-1]}(t-1), z_p^{[q-1]}(t), y_p^{[l]}(t), r(t), k_p\right)$$

自适应模型跟踪策略

代价函数：$J_M(\theta_d, \varphi_d, t) = \sum_{k=1}^{t} J\left(\|\tilde{x}(k)\|, \|\tilde{y}(k)\|, k\right)$

模型参数：

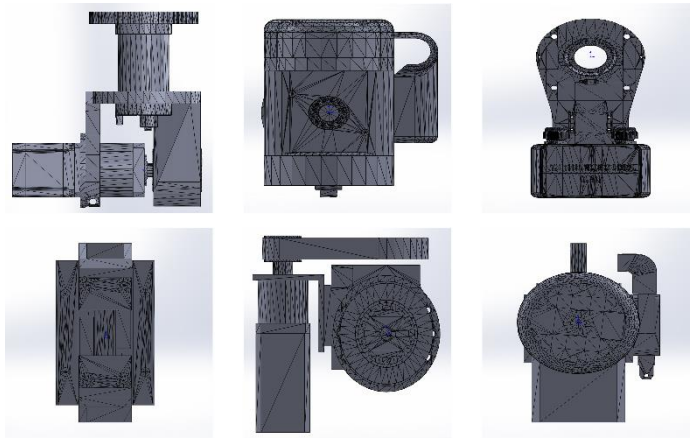
$$\hat{\theta}_d(t) = \hat{\theta}_d(t-1) + \left(\lambda_\theta + F^T(t-1)T_\theta(t-2)F(t-1)\right)^{-1}$$
$$\times T_\theta(t-2)F(t-1)\left(x_p(t) - \hat{\theta}_d^T(t-1)F(t-1)\right)$$

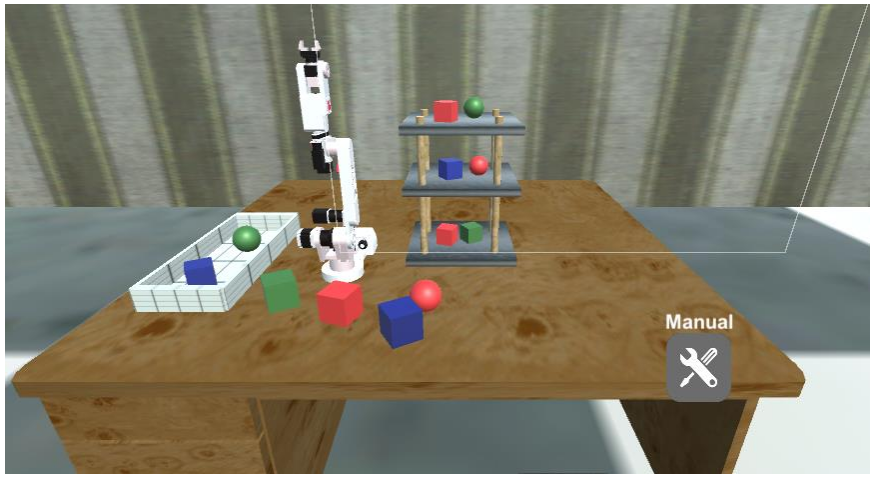
$$\hat{\varphi}_d(t) = \hat{\varphi}_d(t-1) + \left(\lambda_\varphi + G^T(t)T_\varphi(t-1)G(t)\right)^{-1}$$
$$\times T_\varphi(t-1)G(t)\left(y_p(t) - \hat{\varphi}_d^T(t-1)G(t)\right)$$

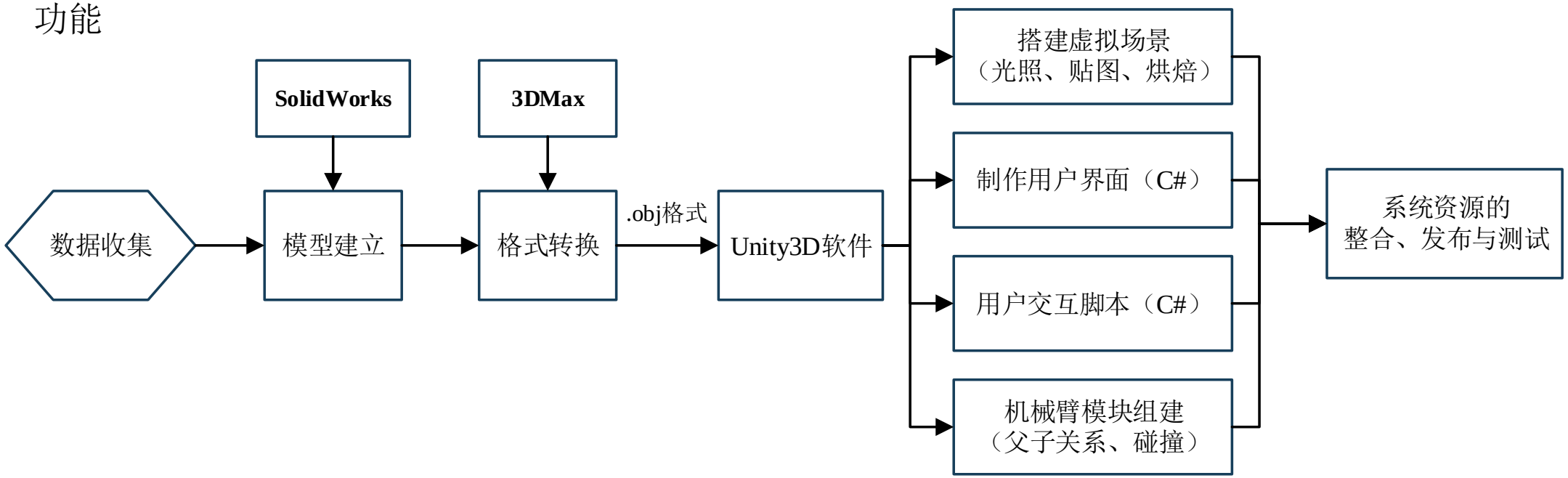
多平台多接口整合：系统协同感知与控制框架

3D空间建模

- 建立机械臂**数字孪生**模型
- 基于Solidworks建立机械臂的各关节模型，通过3DMax进行渲染，参数设置
- 再在Unity3D软件中进行装配，同时搭建场景，编写脚本等实现对应功能



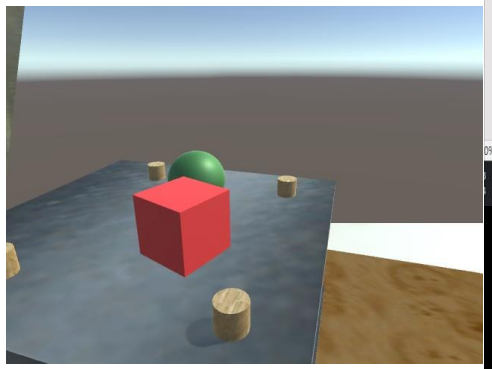


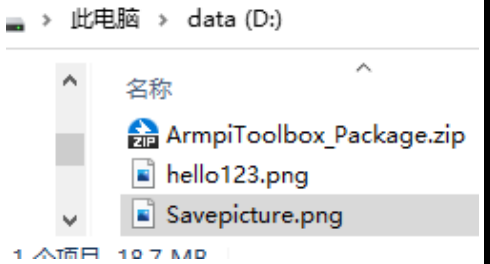


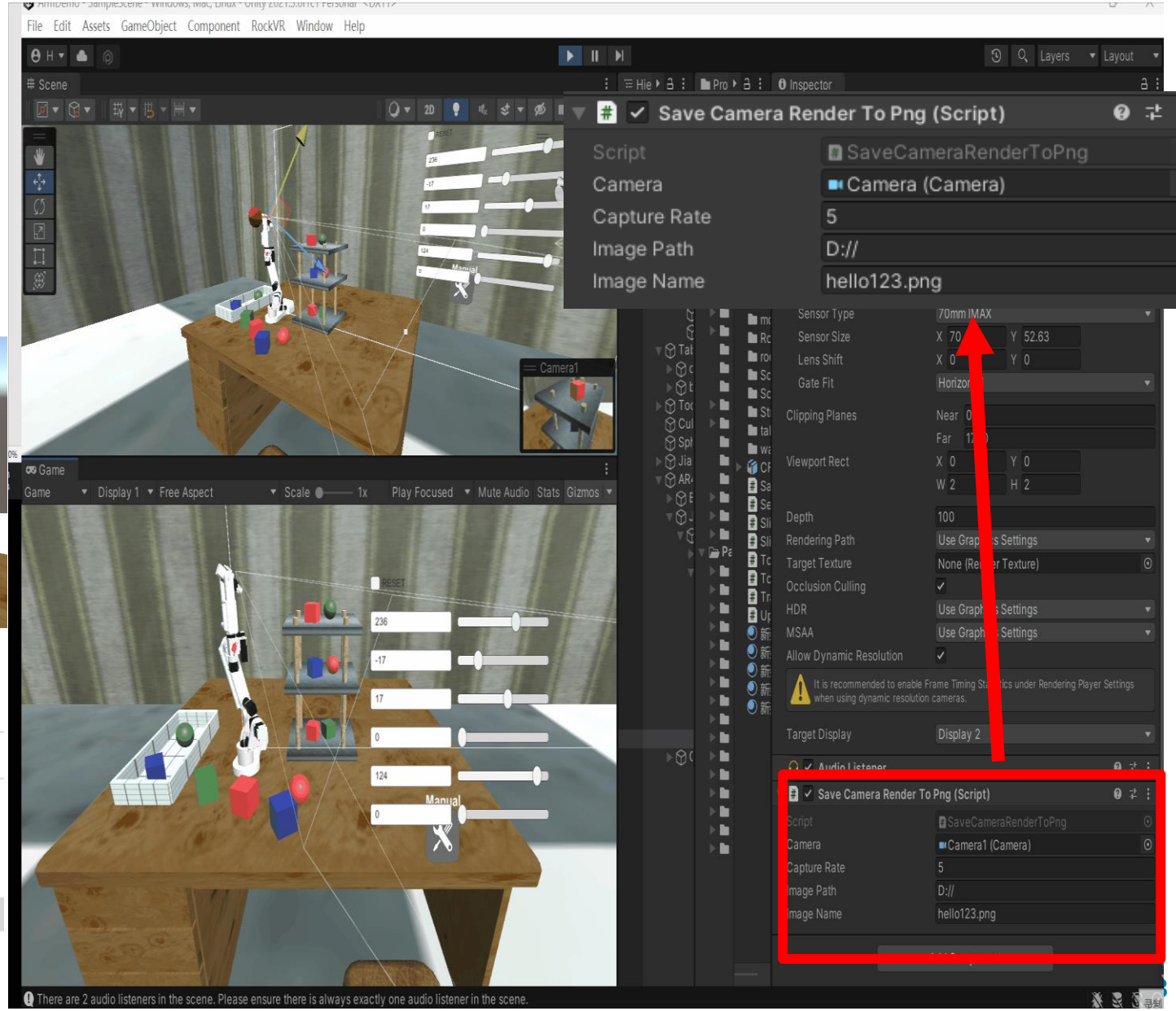
多平台多接口整合：系统协同感知与控制框架

- 虚拟摄像头**视觉导出**
- 编写脚本实现摄像头**图像导出**，可配置导出周期，路径，分辨率等信息
- 编写脚本实现摄像头**视频流导出**，采用推流的方式实现摄像头实时直播

```
public class SaveCameraRenderToPng : MonoBehaviour
{
    [Header("图片的横宽")]
    public int m_nWidth = 8192;
    [Header("图片的竖高")]
    public int m_nHeight = 8192;
    [Header("图片保存的位置")]
    public string m_strImagePath = "D://hello123.png";
    [ContextMenu("截摄像机信息")]
    //public void FixedUpdate()
    public void 截摄像机信息()
    {
        Camera camera = GetComponent<Camera>();
        if (camera != null)
        {
            StartCoroutine(SavePng(camera));
        }
        //Thread.Sleep(10000);
    }
    private IEnumerator SavePng(Camera camera)
    {
        RenderTexture rt = RenderTexture.GetTemporary(m_nWidth, m_nHeight, 24);
        camera.targetTexture = rt;
        camera.Render();
        RenderTexture currentActiveRT = RenderTexture.active;
        RenderTexture.active = rt;
        Rect rect = new Rect(0, 0, m_nWidth, m_nHeight);
        Texture2D screenShot = new Texture2D(m_nWidth, m_nHeight, Text
        screenShot.ReadPixels(rect, 0, 0);
        screenShot.Apply();
        //等待帧结束
        yield return new WaitForEndOfFrame();
        camera.targetTexture = null;
        RenderTexture.active = currentActiveRT;
        byte[] pngBytes = screenShot.EncodeToPNG();
        File.WriteAllBytes(m_strImagePath, pngBytes);
        RenderTexture.ReleaseTemporary(rt);
        //释放 screenShot
        if (Application.isPlaying)
        {
            Destroy(screenShot);
        }
        else
        {
            DestroyImmediate(screenShot);
        }
```

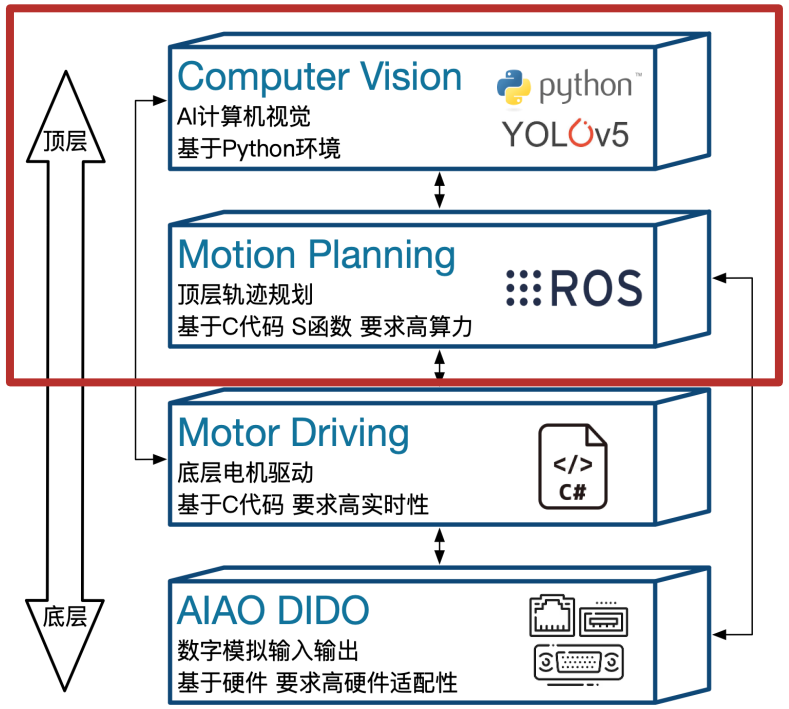


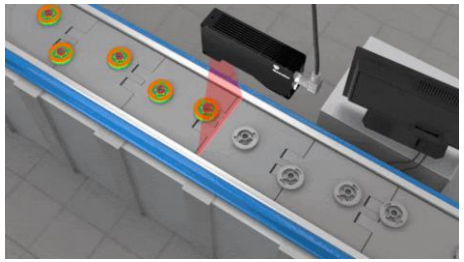




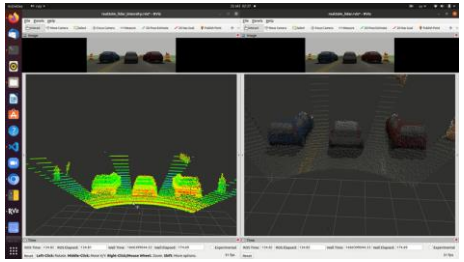
多平台多接口整合：系统协同感知与控制框架

结合Computer Vision（CV）的机械臂控制

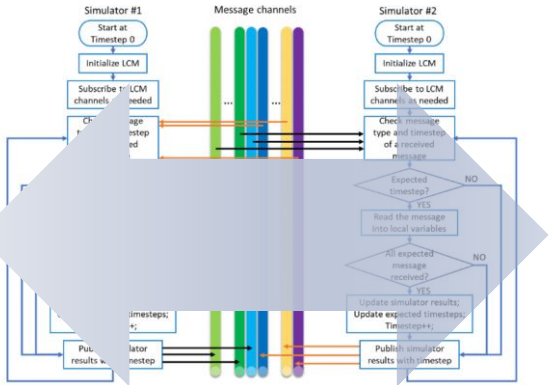


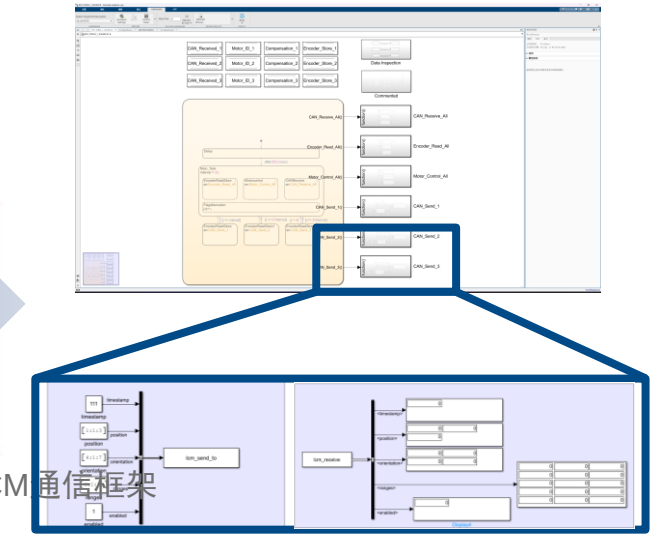


python YOLOv5



ROS
ROS2





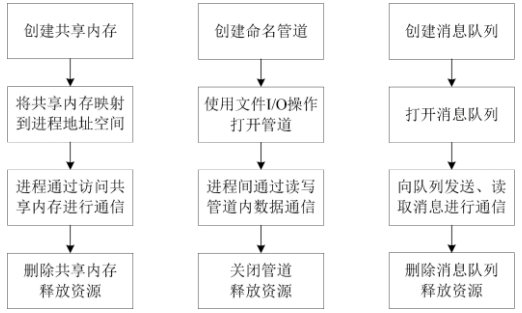
LCM通信框架

- **通过**

Shared Memory 内存共享

Message Queue 消息队列

Lightweight Communication and Marshalling (LCM)



- **实现**

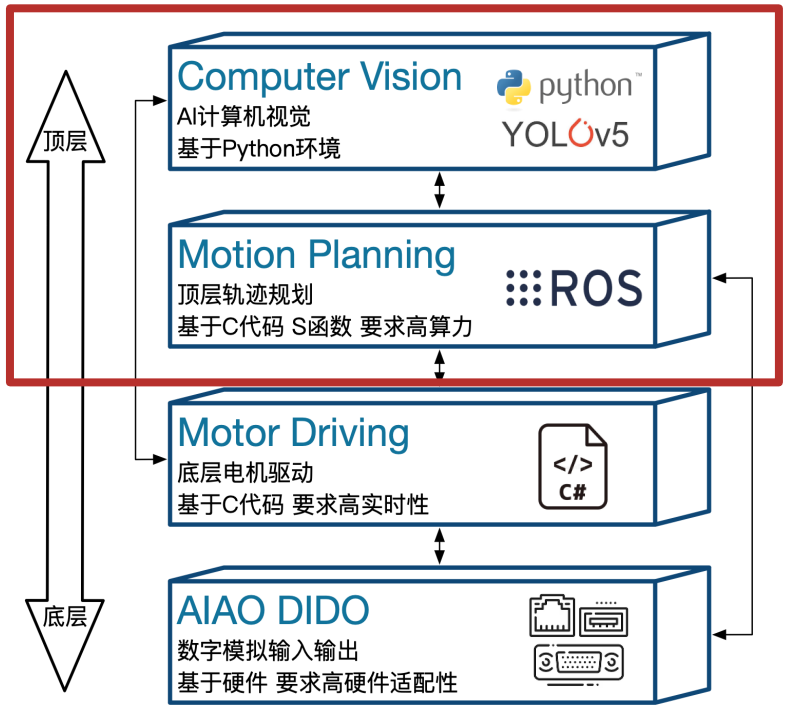
将AI应用整合进系统

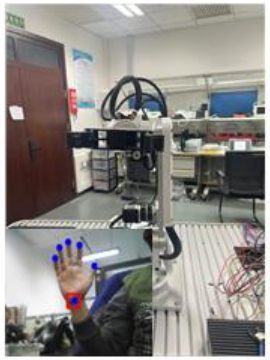
对 ROS 的支持和应用

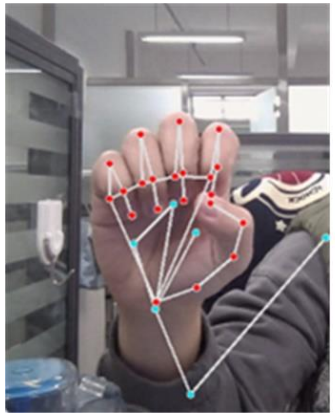
对 python 的应用融合视觉信息

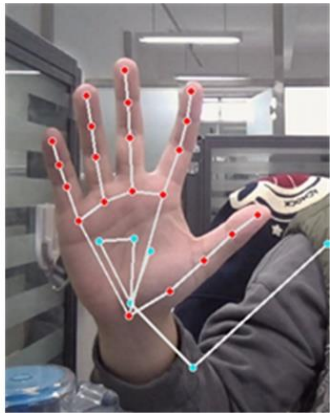
多平台多接口整合：系统协同感知与控制框架

基于手势引导的机械臂遥操作实例









手势引导机械臂控制

- **机械臂手势引导控制**

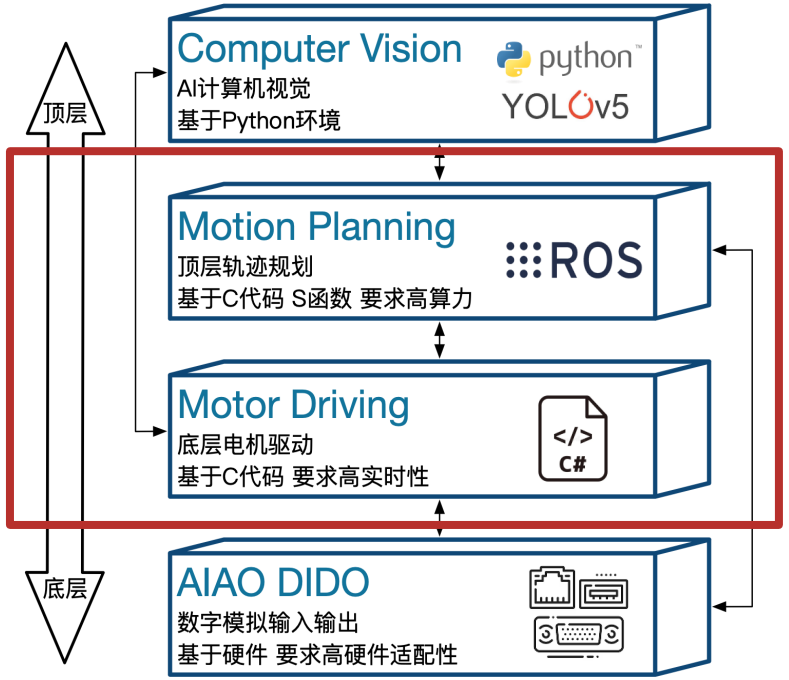
  深度摄像头识别手部运动趋势

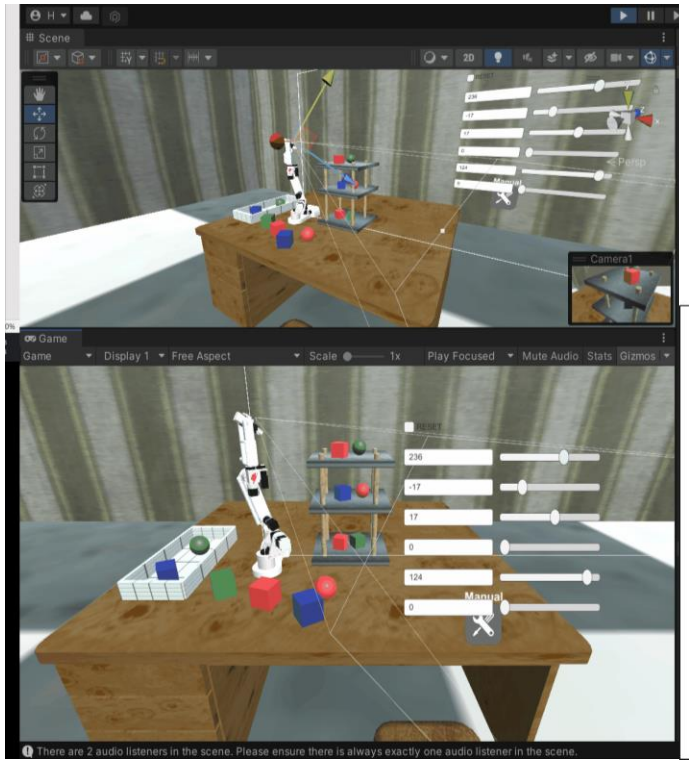
  控制器根据运动趋势控制机械臂运动

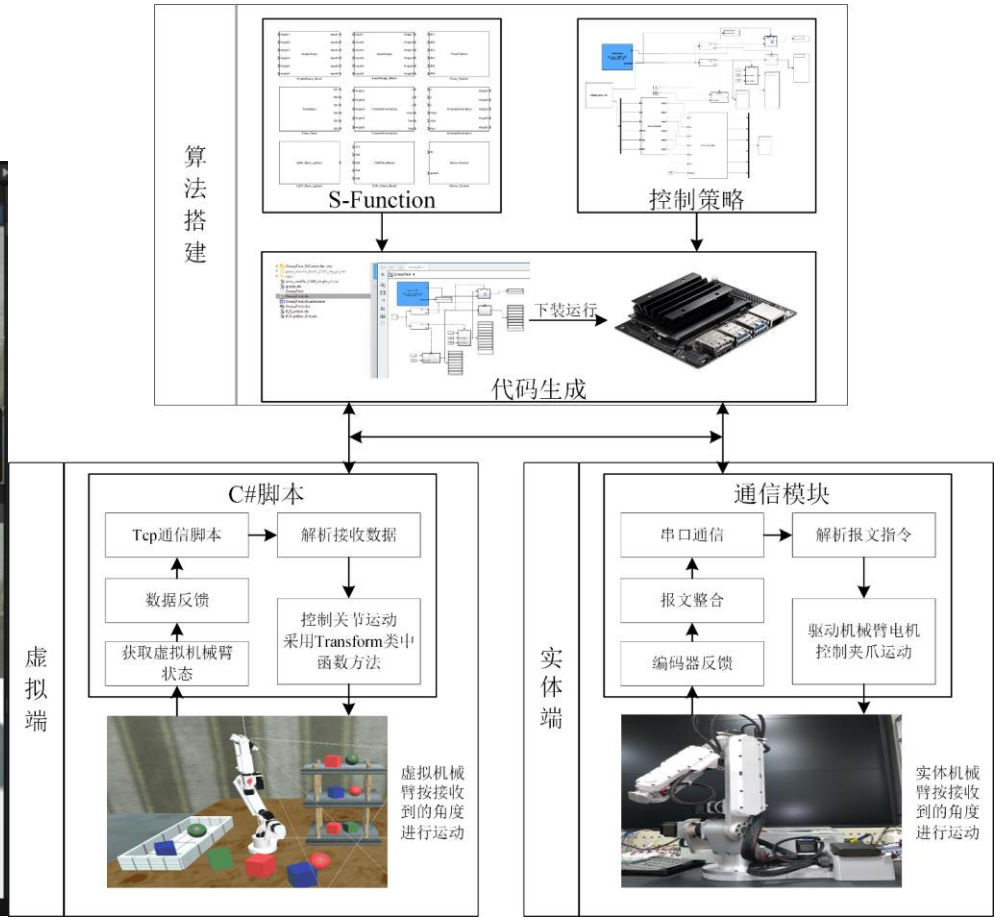
  三维空间各方向运动演示

多平台多接口整合：系统协同感知与控制框架

控制系统系统开发







机械臂的数字孪生控制系统

- **机械臂数字孪生平台实现**

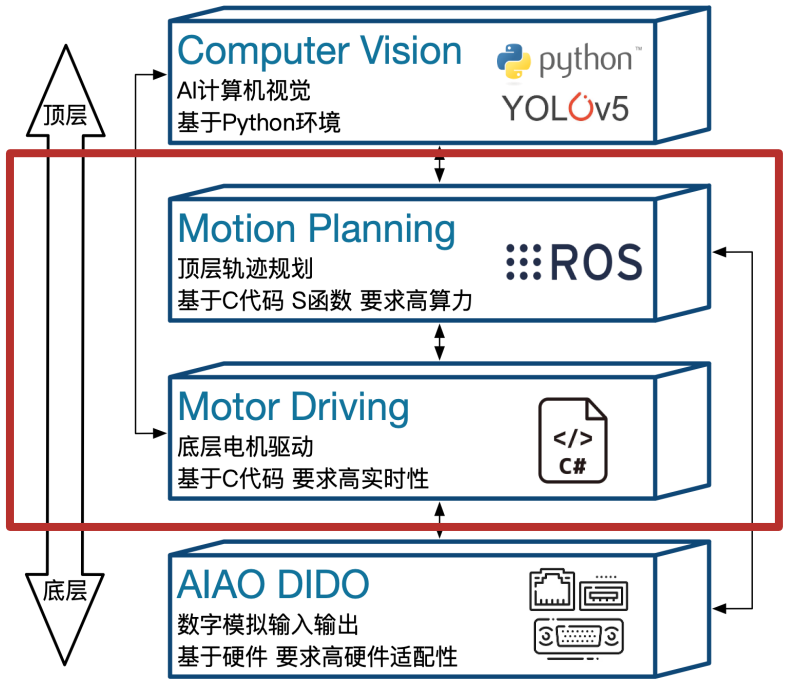
控制器与Unity3D通信

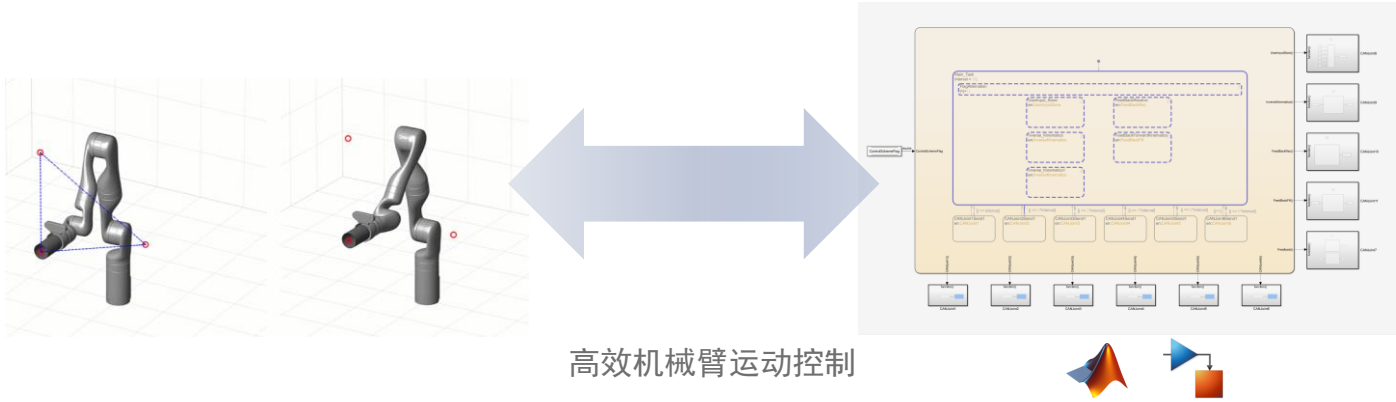
工具链通过TCP通信模块打通本体与孪生体接口

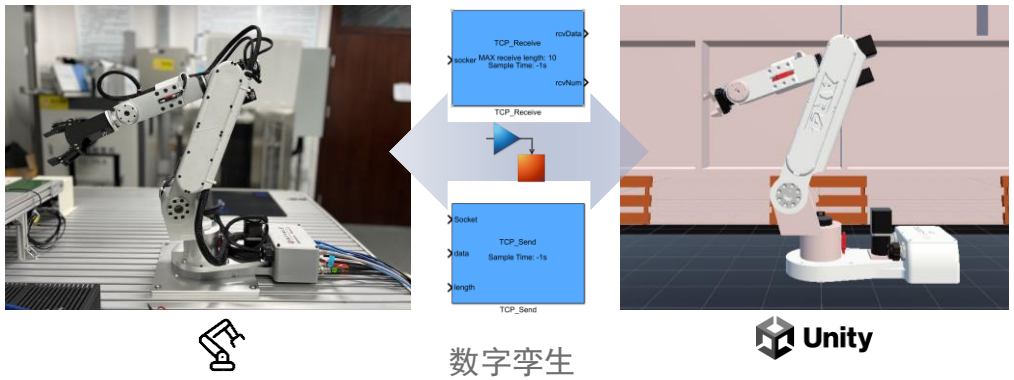
机械臂在虚拟场景下的孪生体控制演示

多平台多接口整合：系统协同感知与控制框架

多平台协同







- **通过**

  CAN总线通信, Serial串口通信

  TCP/IP通信等

  高算力平台进行轨迹规划运算及电机控制调度

- **实现**

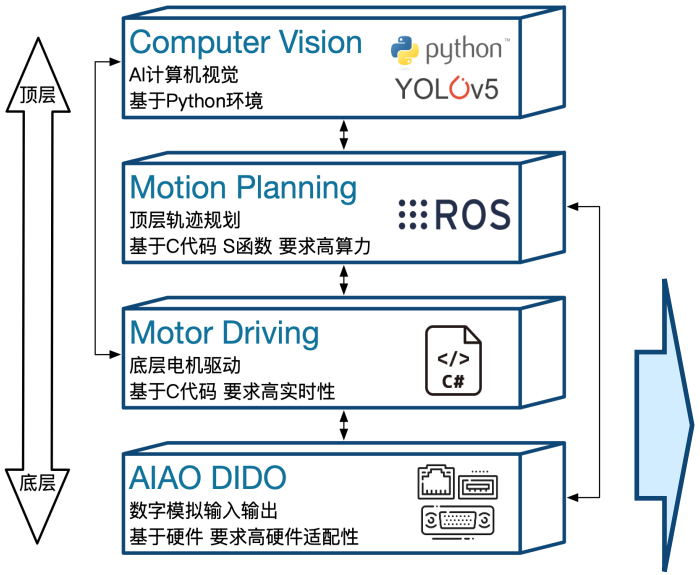
  高效机械臂轨迹规划

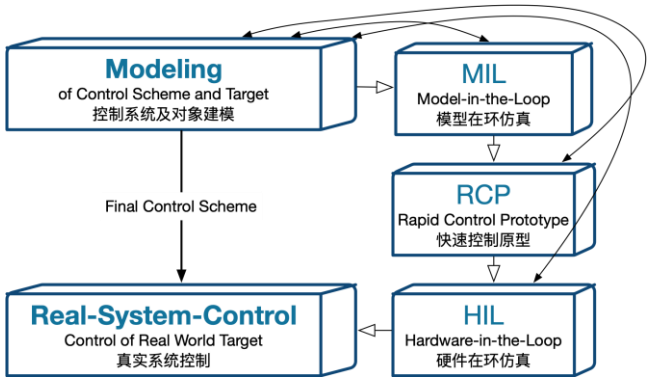
  高实时性高精度电机控制

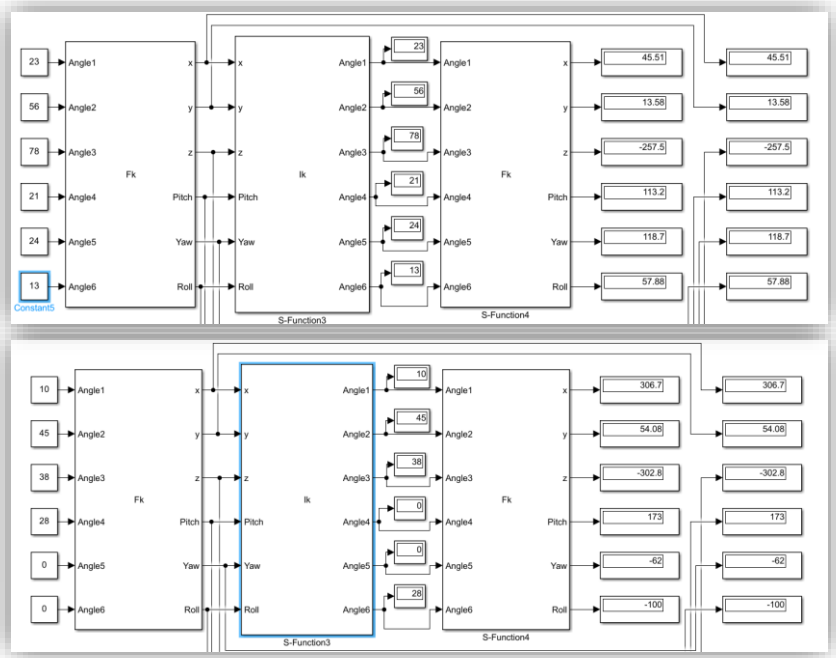
  数字孪生，包括Unity 3D平台等

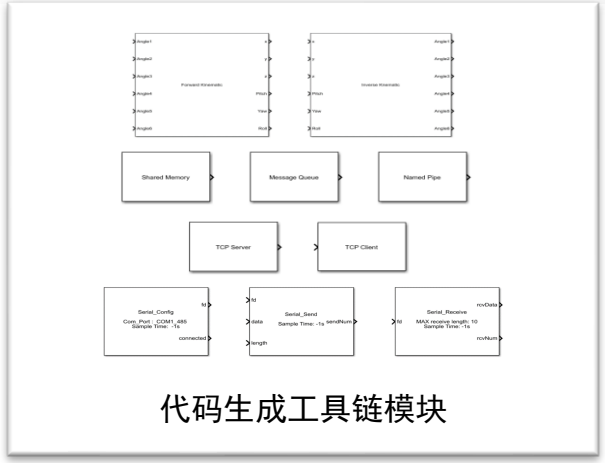
多平台多接口整合：控制系统开发新方案

控制系统模块库

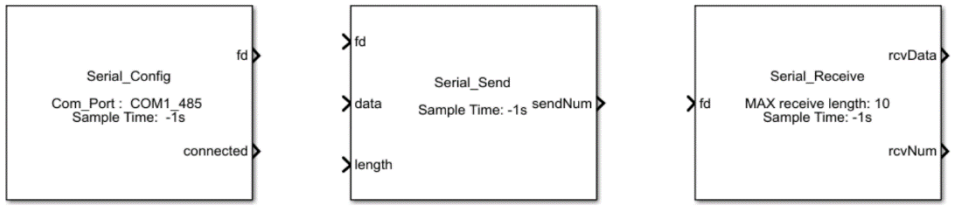








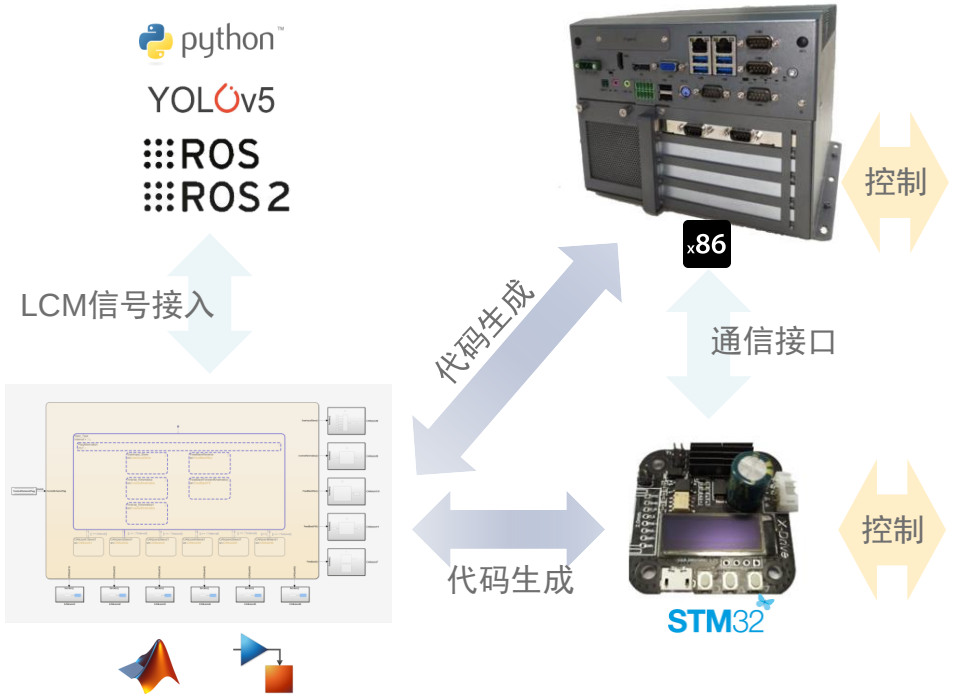
代码生成工具链模块

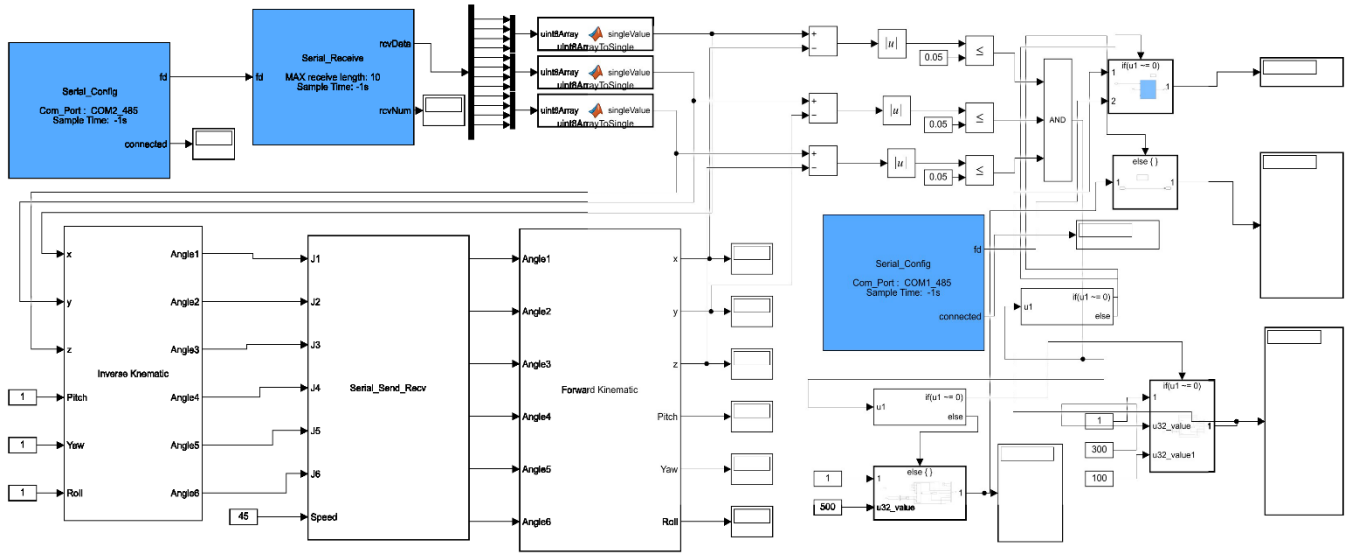


基于S-Function的机械臂控制系统模块库

多平台多接口整合：控制系统开发新方案

控制策略设计与实现





基于RTW的快速控制系统原型

- **通过**

  多平台的应用

  面向X86/ARM/STM32的Embedded Coder代码生成

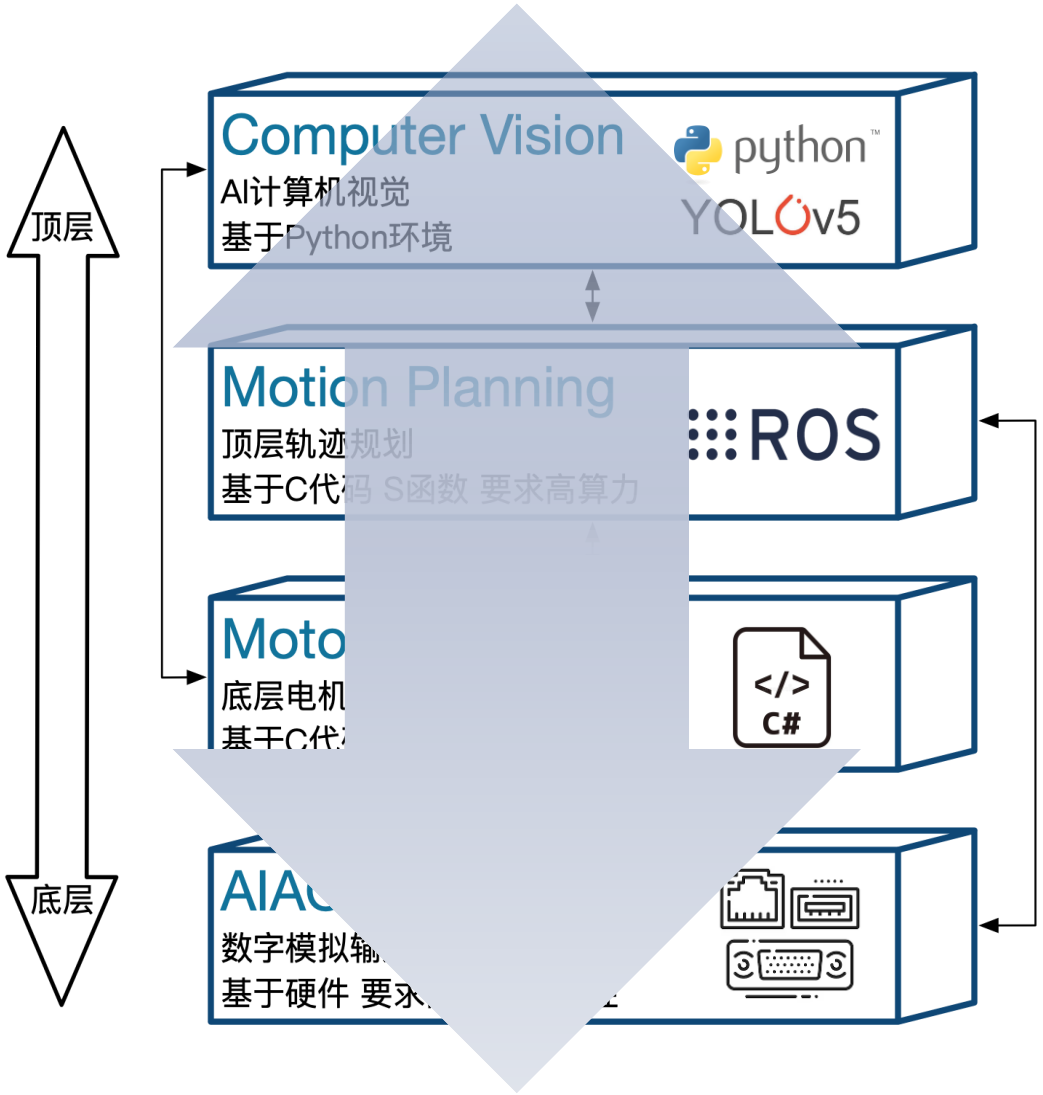
- **实现**

  分层控制策略的快速实现与验证

  在线调参与测试

多平台多接口整合：控制系统开发新方案

开发案例：结合Computer Vision（CV）的机械臂控制

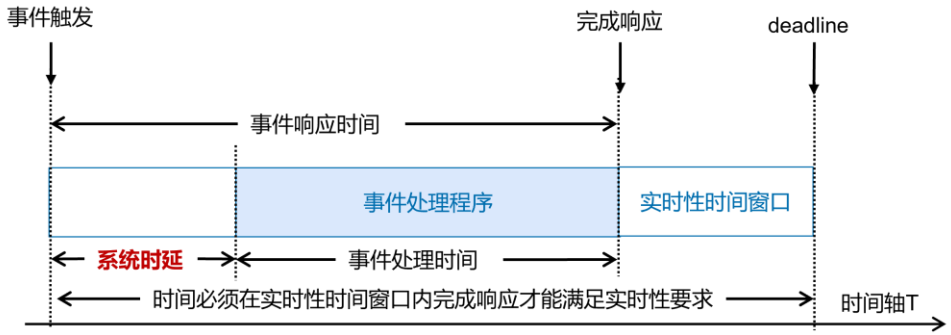


- **实现从顶层到底层的打通**
- 基于RCP的开发也使得系统可以**灵活重组**
- 其他应用：
  - ROS智能小车控制：　SLAM建图及自主导航
  - 外骨骼控制系统开发：　快速由反馈构建**复杂**外骨骼系统
  - 数字孪生系统开发：　灵活开发对象数字孪生
  - 电力系统**复杂**模型：　**复杂**负载调配算法部署
  - ……
- **复杂系统如何保证实时性？**

如何保证复杂控制算法运行实时性？

多核编程+实时内核能极大提升模型运行实时性

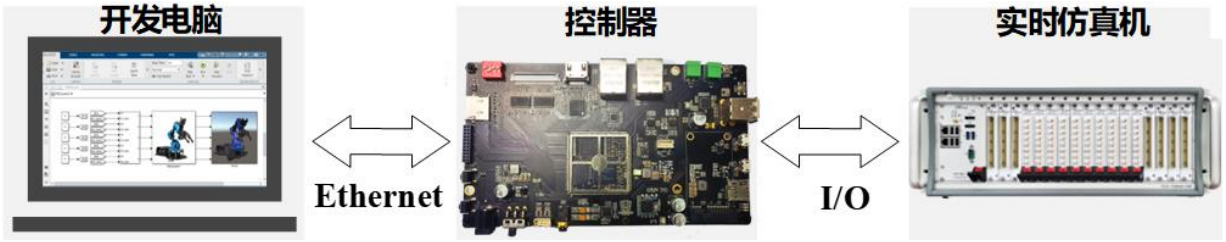
如何提升系统硬实时任务的实时性？



实时性概念

快速控制原型：硬实时任务，模型复杂度小

E.g. 电机控制，运动控制系统



硬件在环测试：硬实时任务，模型复杂度大

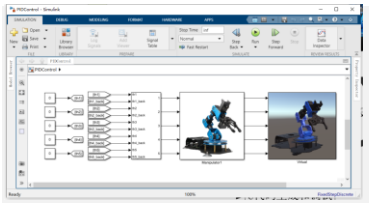
E.g. 氢能硬件在环测试，微电网能源管理算法硬件仿真，智能电网建模仿真优化

操作系统中任务分类	定义
硬实时任务	在截止时间之后产生结果，可能对受控系统造成**灾难性后果**
强实时任务	在截止日期之后产生结果对系统无用，但不会造成任何损害
软实时任务	实时任务在截止日期之后产生结果仍然对系统有用，尽管会导致性能下降

如何提升系统硬实时任务的实时性？


自动配置编译环境


I/O驱动模块库


用户模型

一键编译

Robot_X86Controller_rtw
slprj
Robot
Robot.slx
可执行代码

自动下装

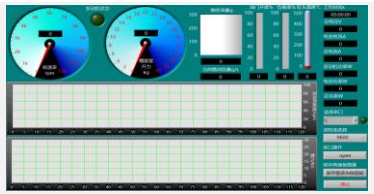

RCP控制器

I/O

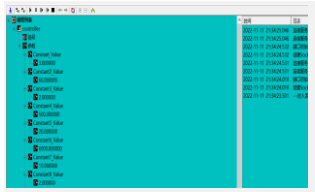

被控对象

Simulink/外部模式

或


第三方上位机软件

OPC UA/DA


上位机监控界面

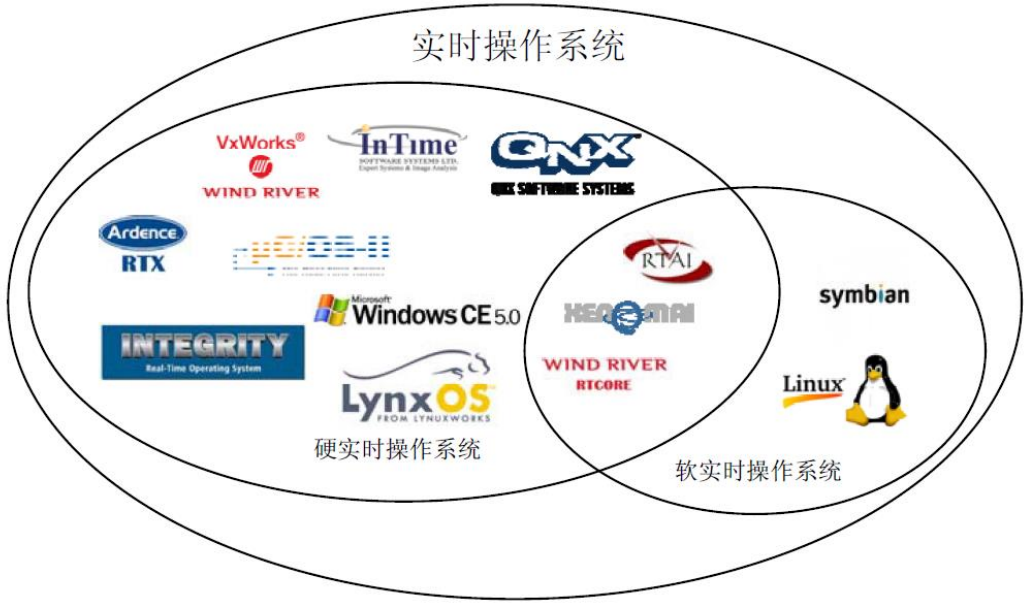
实时监控、在线调参

问题一：操作系统**实时性差**，控制算法模型由**非实时内核调度**

问题二：Simulink外部模式**数据监控实时性差**，通讯**带宽有限**

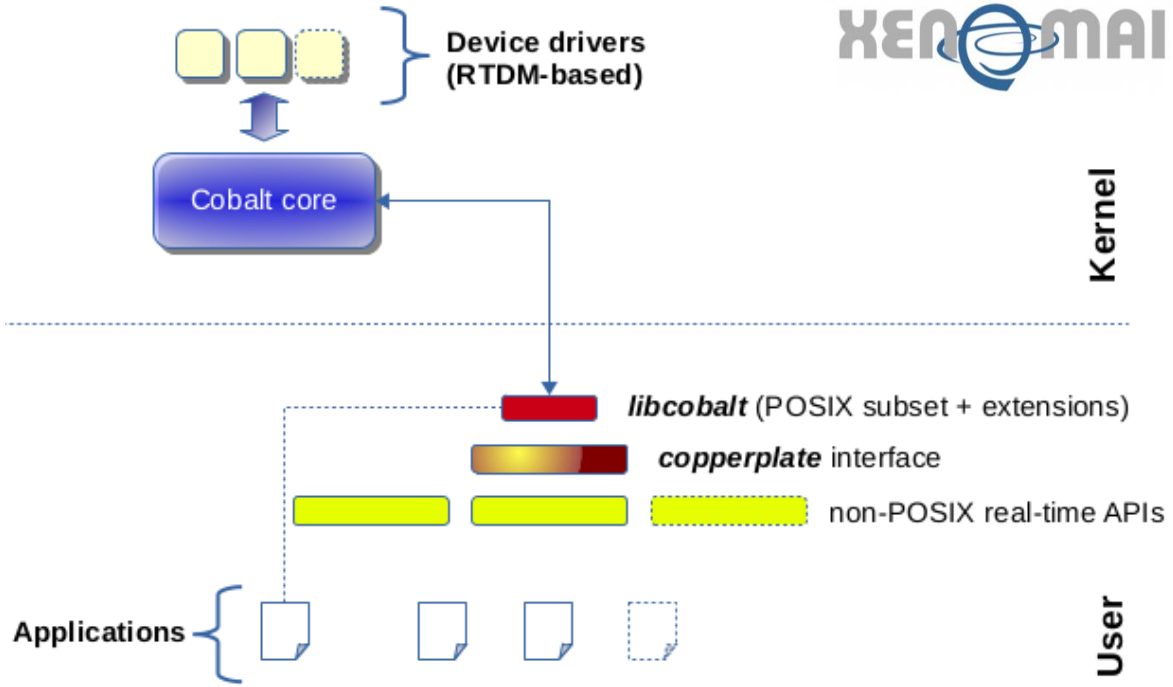
问题三：复杂模型需**多任务并行**，不能充分利用**多核**、**多核异构**处理器性能

提升系统硬实时任务的实时性：使用实时内核调度模型



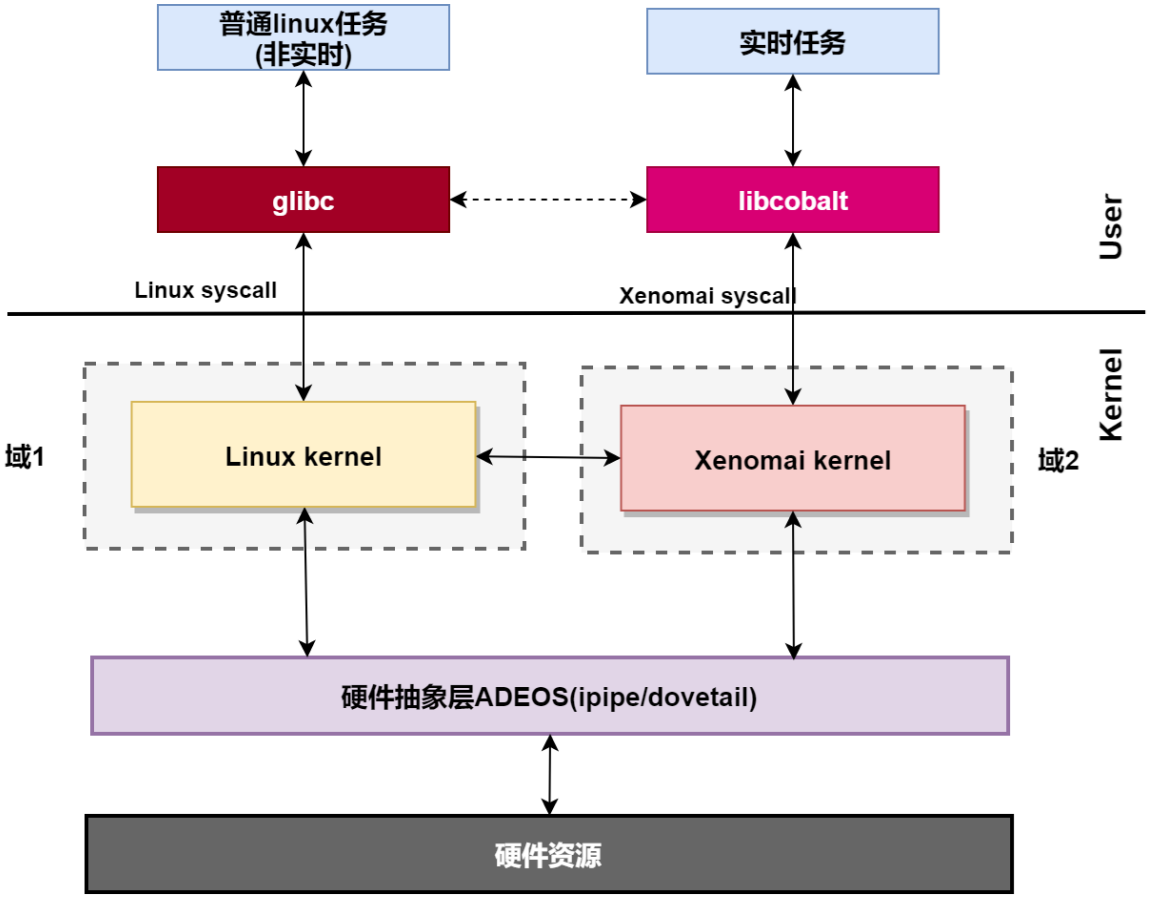
- 操作系统：Linux + **Xenomai**
- 使用Cobalt**实时内核**调度Simulink业务模型硬实时任务和数据监控任务
- 使用RTDM开发外设硬件驱动赋予**硬件驱动接口实时性**

方案	Xenomai	RTAI	RT-Linux
用户态实时应用	支持	支持	否
内核态实时应用	支持	支持	支持
X86支持	支持	支持	支持
POSIX编程接口	支持	否	否
是否开源	是	是	已被风河收购

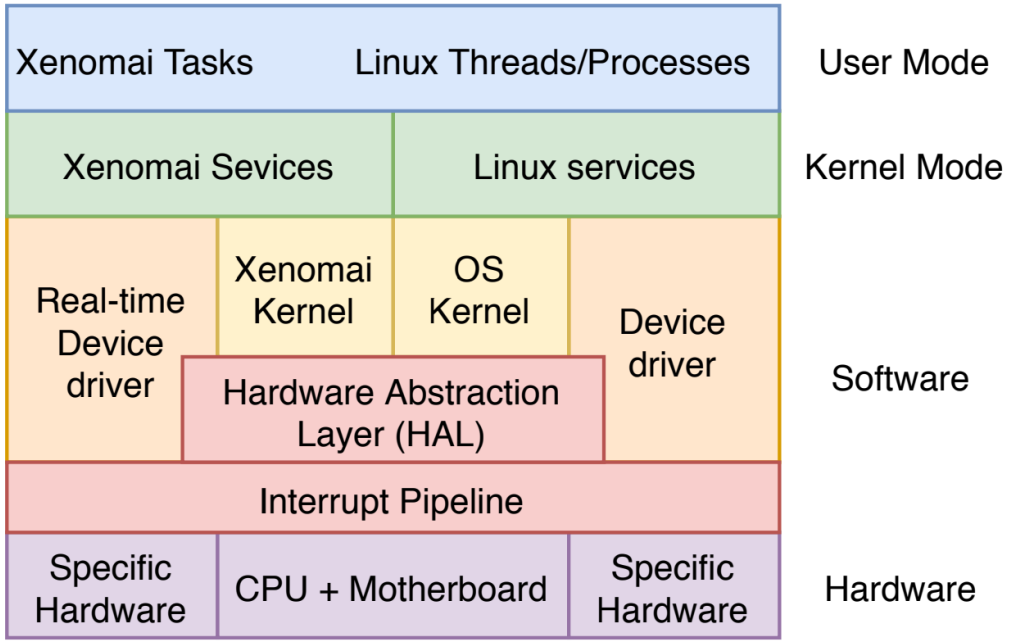


Xenomai任务调度原理

提升系统硬实时任务的实时性：使用实时内核调度模型



Xenomai整体结构



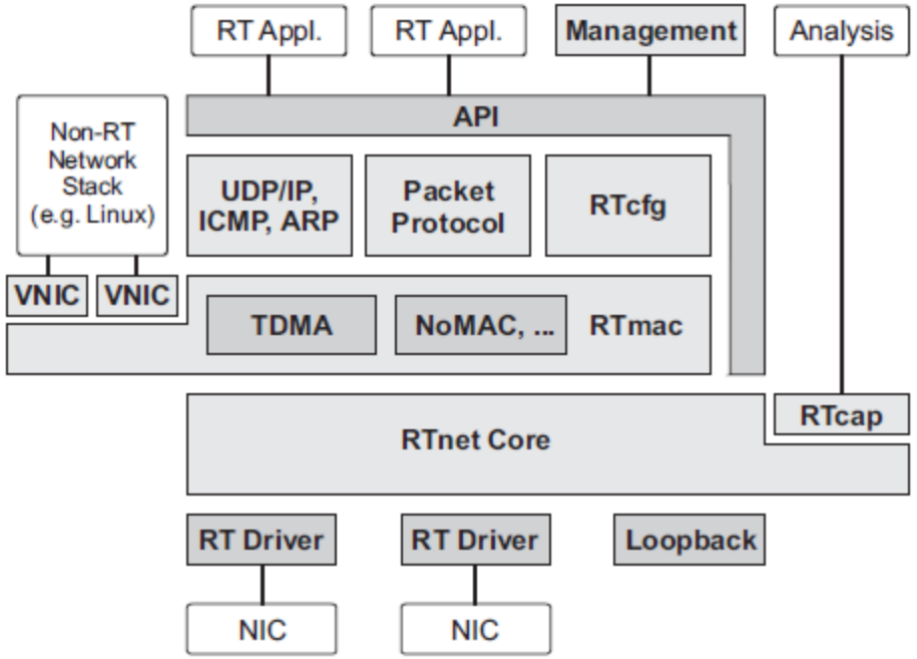
在标准Linux基础上添加一个实时内核Cobalt，得益于基于ADEOS（Adaptive Domain Environment for Operating System），使Cobalt在内核空间与Linux内核并存，并把标准的Linux内核作为实时内核中的一个Idle进程在实时内核上调度。

实时内核Cobalt与**非实时内核Linux**相结合，既能提供工业级RTOS的**硬实时性能**，又能利用Linux操作系统非常出色的**生态、网络和图形界面服务**，在产品的开发周期和成本控制方面都有巨大优势 。

提升系统硬实时任务的实时性：硬实时以太网通信外部模式实现

影响外部模式数据监控的实时性的因素：

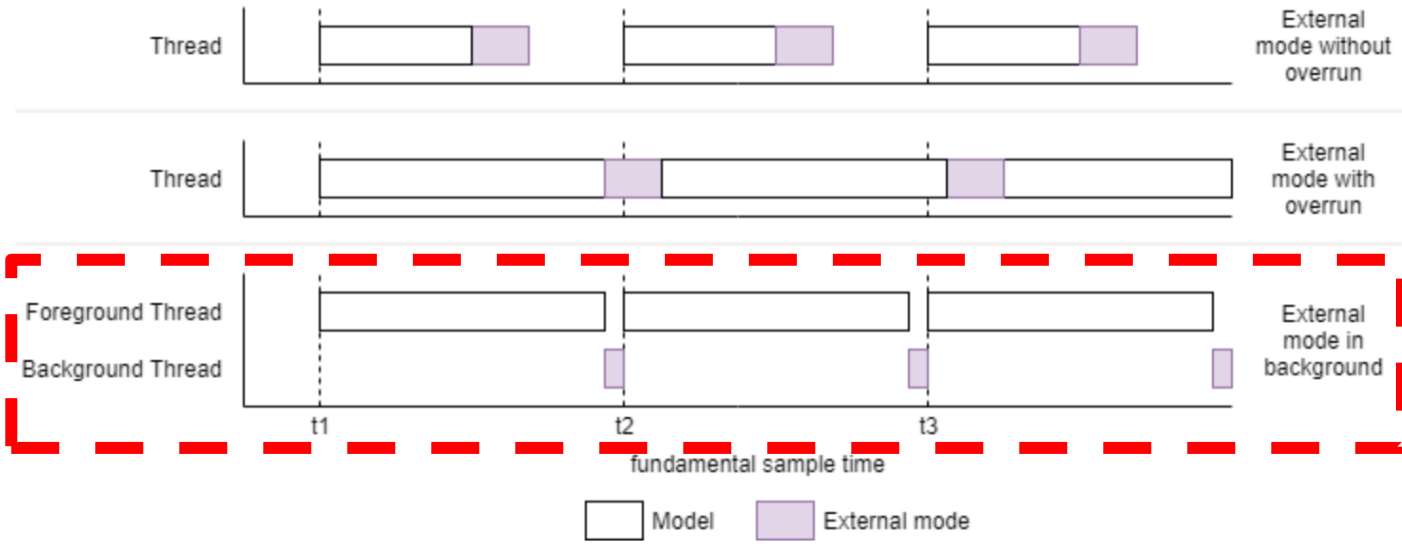
- 单位通信周期传输**数据量过大**；
- 外部模式基于TCP/IP协议**数据传输慢**，且没有**通用性**；
- 数据传输与实际业务代码为**同一线程**，**顺序执行效率低**，且影响主模型运行。



Rtnet硬实时网络架构

影响外部模式数据监控的实时性的因素：

- 移植**RTnet硬实时网络框架**，提供通信**硬实时性**；
- 外部模式改用基于TCP/IP的XCP协议，传输层使用TCP/IP，协议层采用XCP，**提升通用性**；
- 数据传输与实际业务代码分**不同线程**，**并行效率高**，不影响主模型运行。



分多线程的外部模式

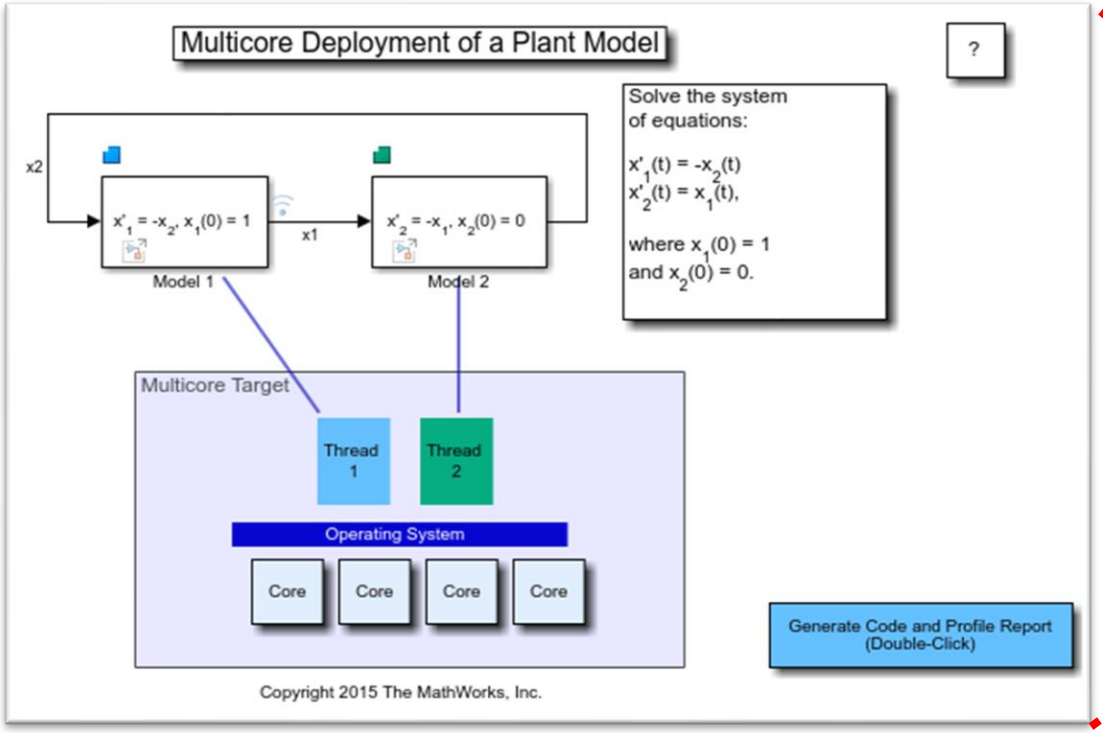
提升复杂模型的实时性：多核CPU + Simulink 多核编程 + 实时内核

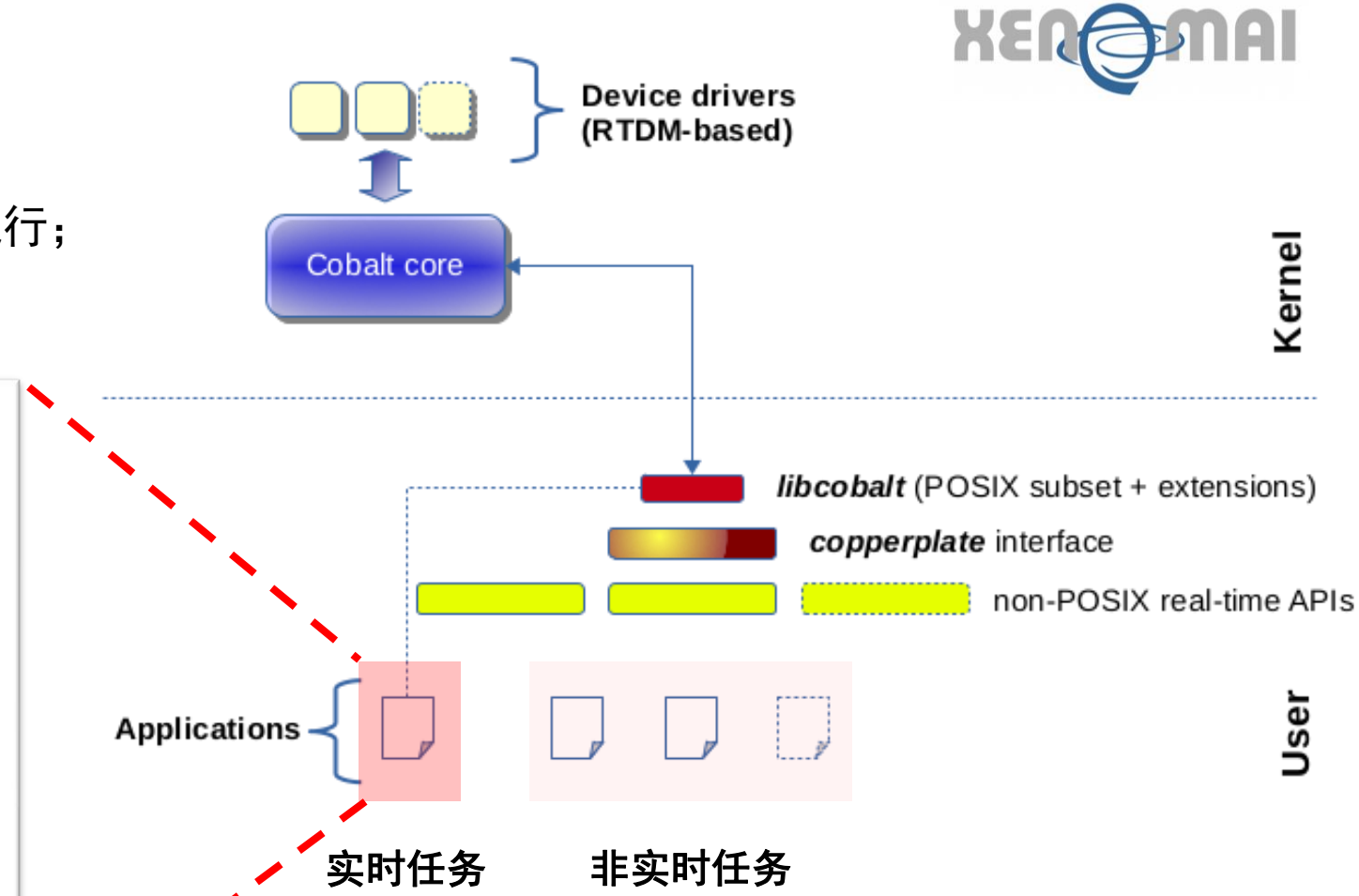
Simulink多核编程流程：

Step1 将子系统转化为模型引用/原子子系统；

Step2 模型配置-求解器-允许任务在目标上并发执行；

Step3 配置任务和映射，CPU核心分配。





Xenomai多线程任务调度原理

提升复杂模型的实时性：Simulink 多核编程 + 实时内核效果

- 以一个小步长（150微秒 μs）运动控制系统仿真模型为例测试：

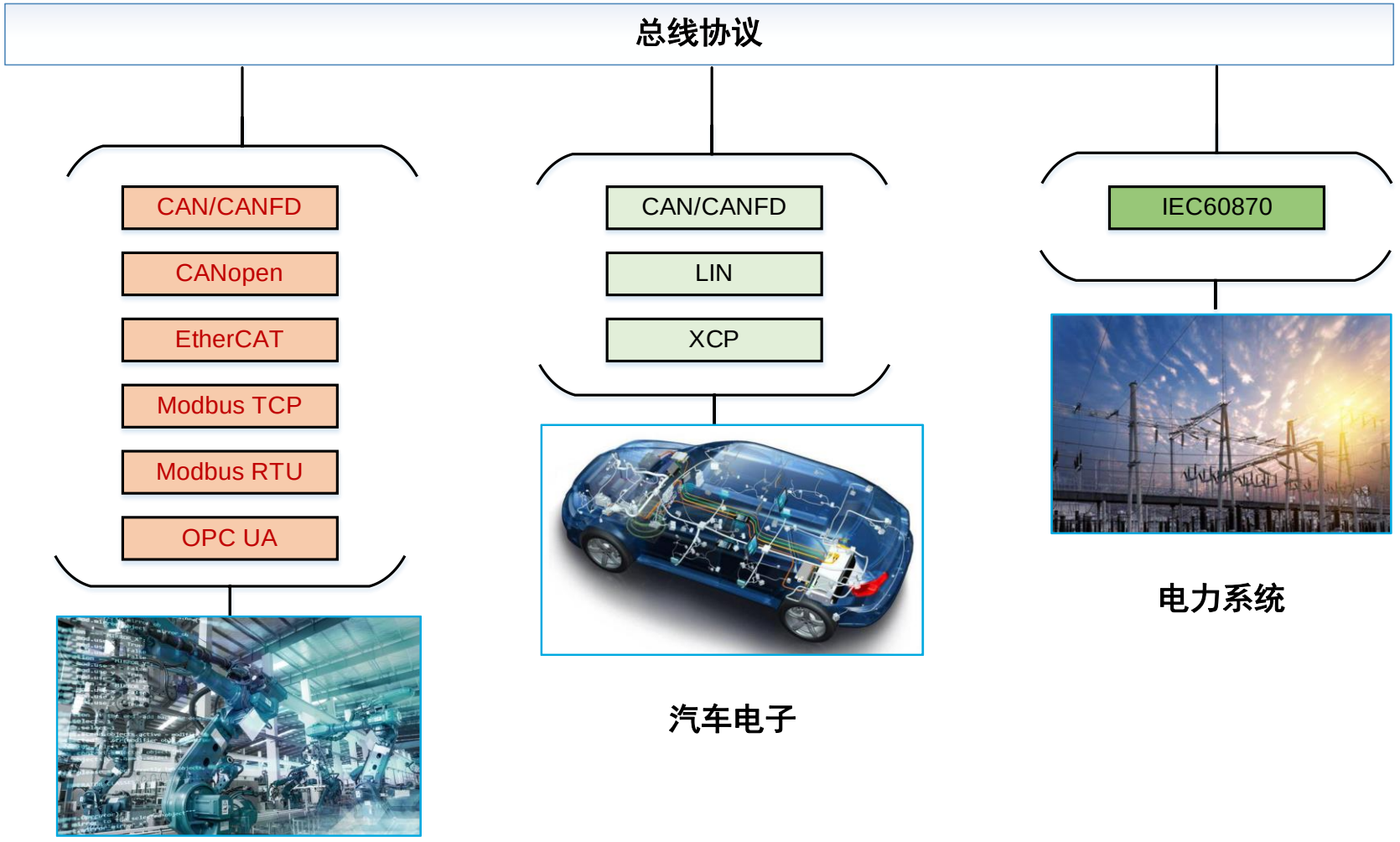
Tasks名称	单个步长（Output+Update）平均耗时（微秒 μs） X86 i5 6500			
	单核CPU	单核CPU+ 实时内核	多核CPU + Simulink多核编程	**多核CPU + Simulink多核编程 + 实时内核**
Task1	50	37	42	**38**
Task2	151	125	252	**129**
Task3	108	82	140	**83**
Task4	56	46	51	**48**
Total Tasks	**345**	**290**	**252**	**129**
满足采样时间要求	×	×	×	√

多核CPU+Simulink多核编程+实时内核能极大提升复杂模型在硬件上运行的实时性！

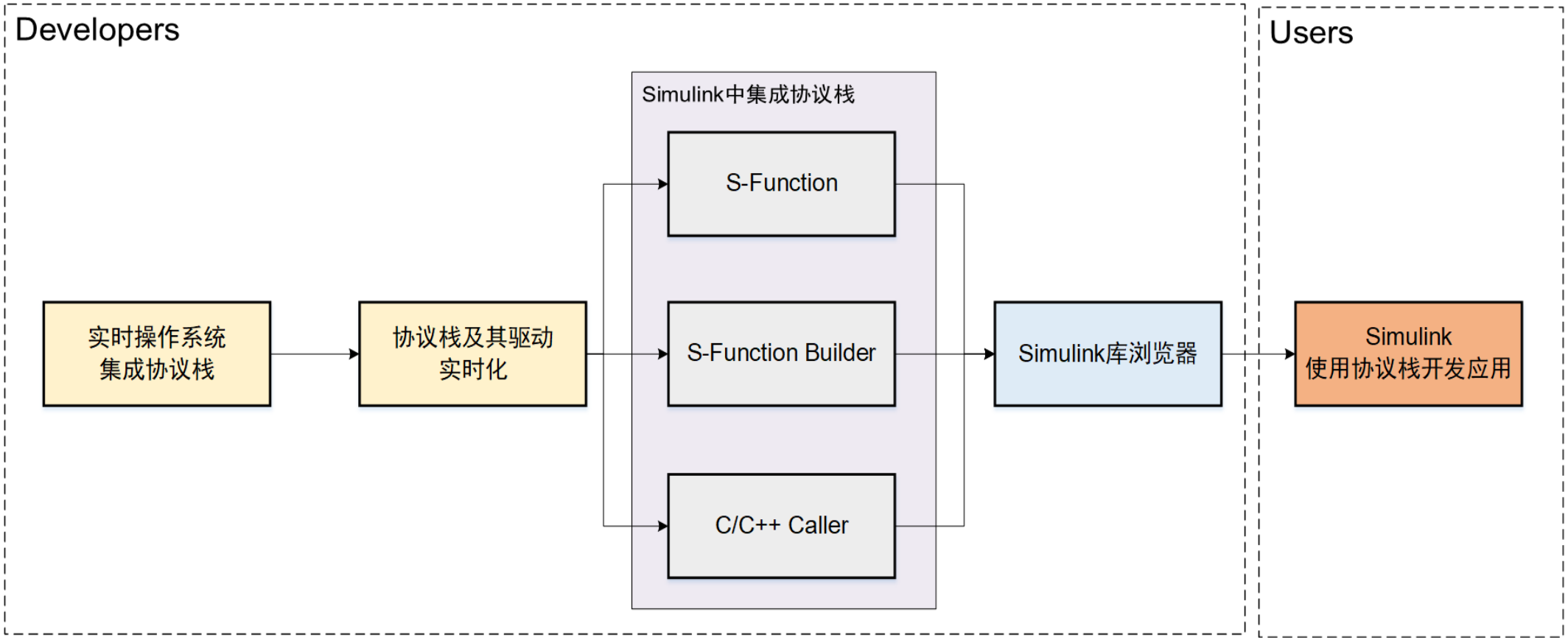
如何通过总线将更多设备接入系统？

在工业自动化，汽车电子，电力系统等领域中的应用

如何在多领域应用：总线协议

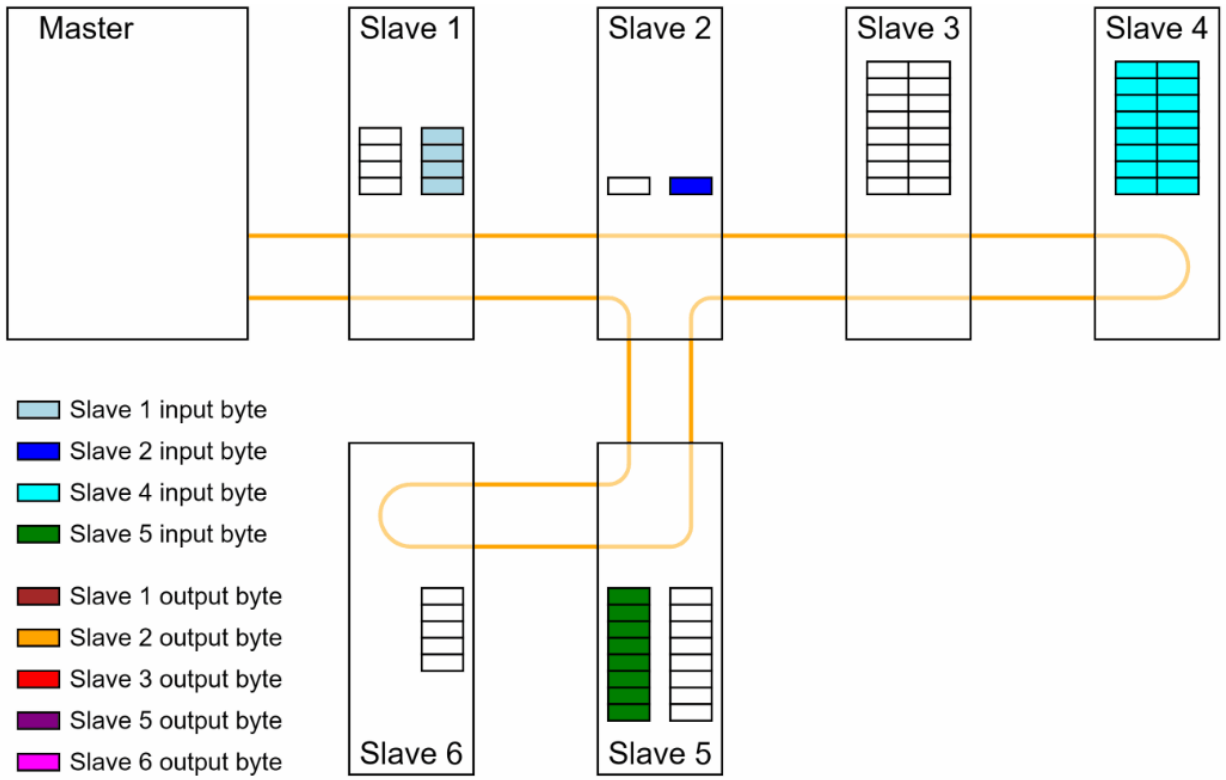


如何在多领域应用：Simulink协议栈开发与应用

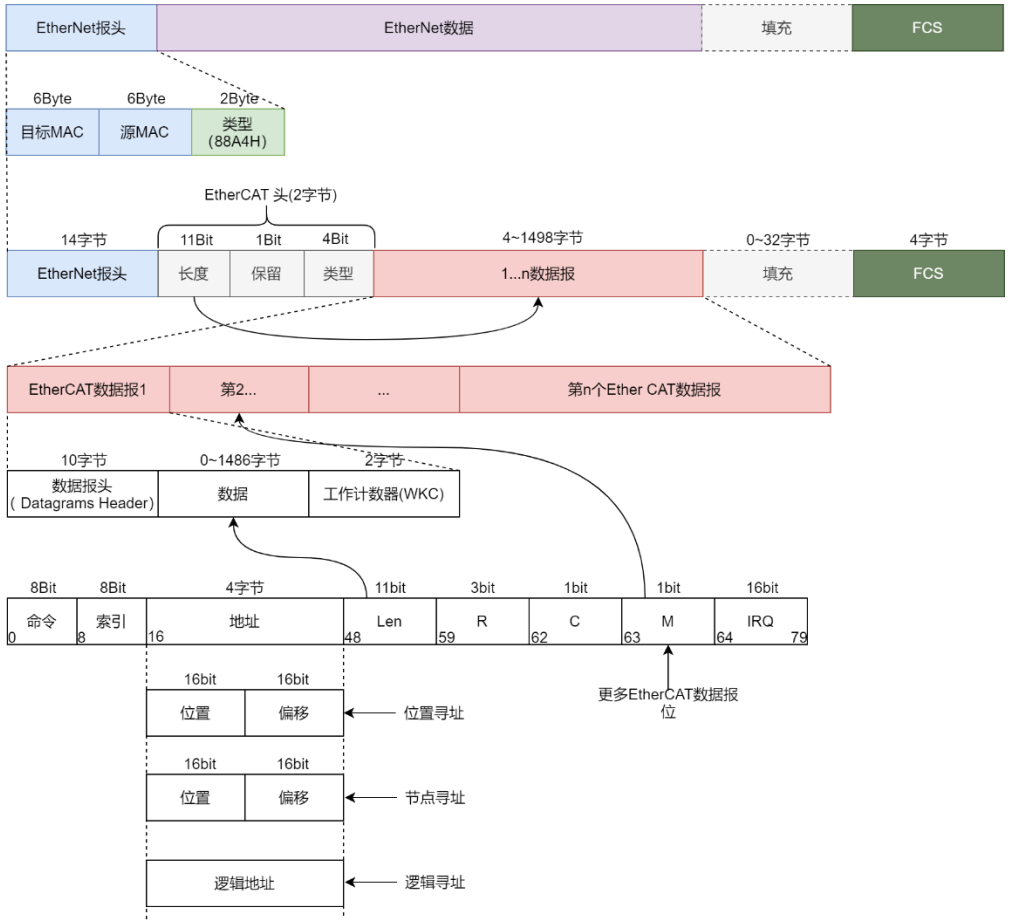


Simulink协议栈开发与应用

Simulink开发集成协议栈：以EtherCAT主站为例



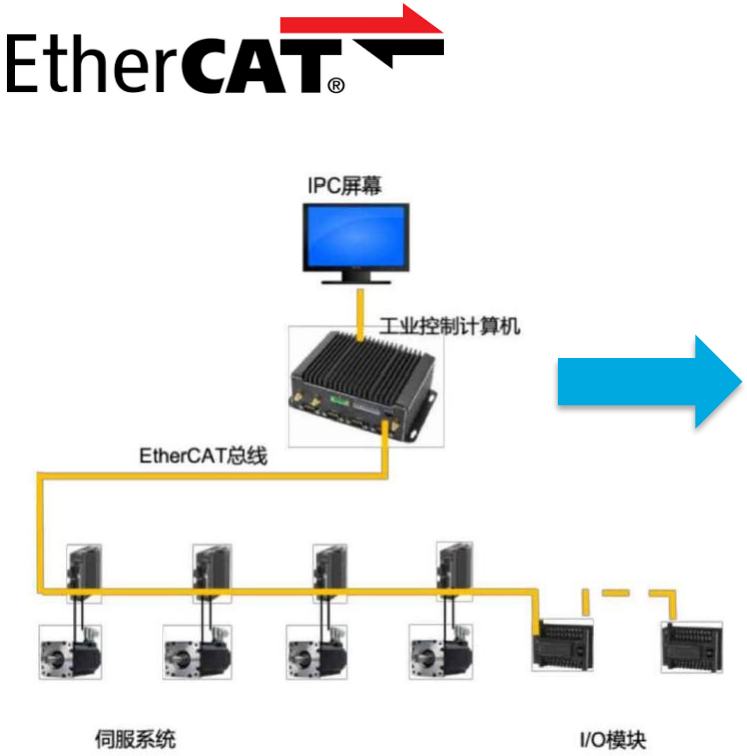
EtherCAT通信原理

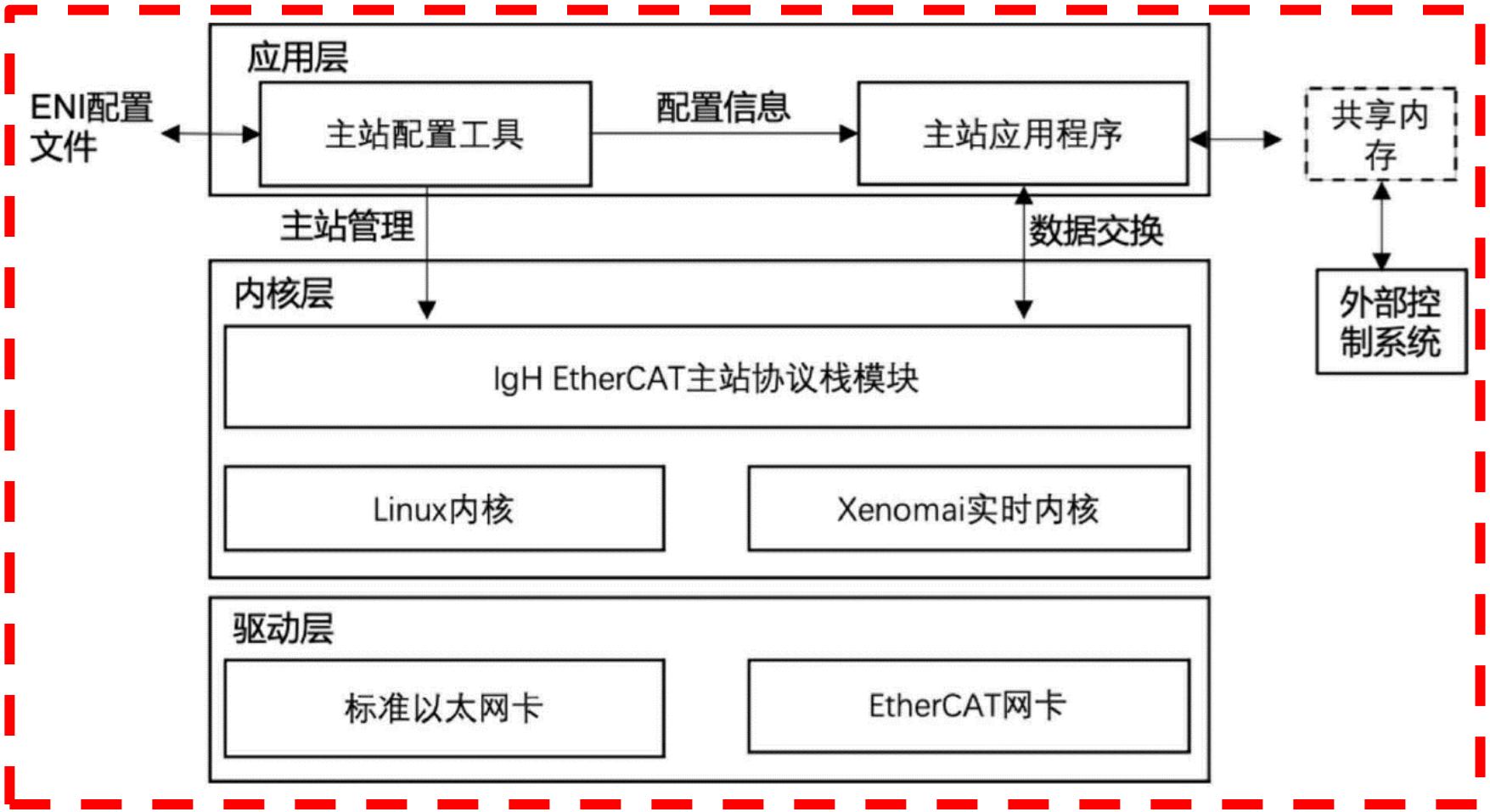


EtherCAT数据帧结构

EtherCAT是以太网为基础的现场总线系统，其使用标准的IEEE802.3以太网帧，在主站一侧使用标准的以太网控制器，**不需要额外的硬件**。

Simulink开发集成协议栈：以EtherCAT主站为例



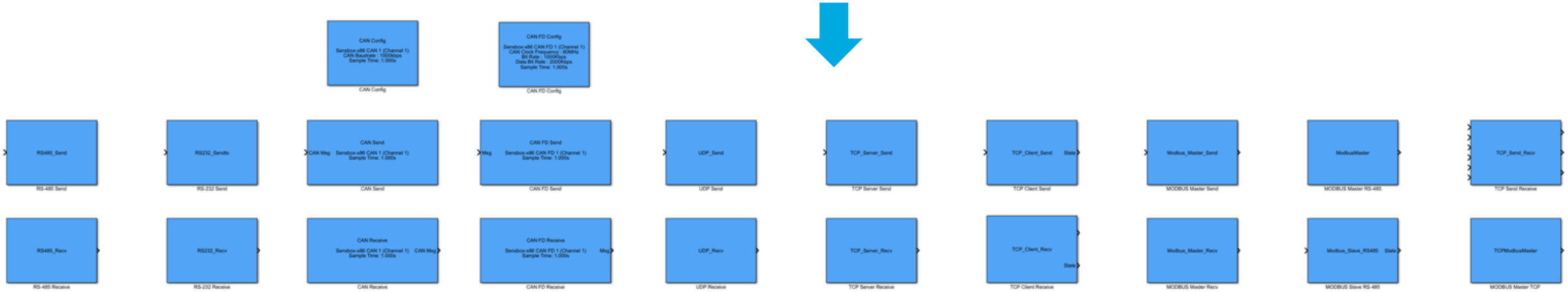


EtherCAT主站软件系统架构

基于模块化、分层化的原则，整个EtherCAT主站软件可分为三层，应用层、内核层以及驱动层。

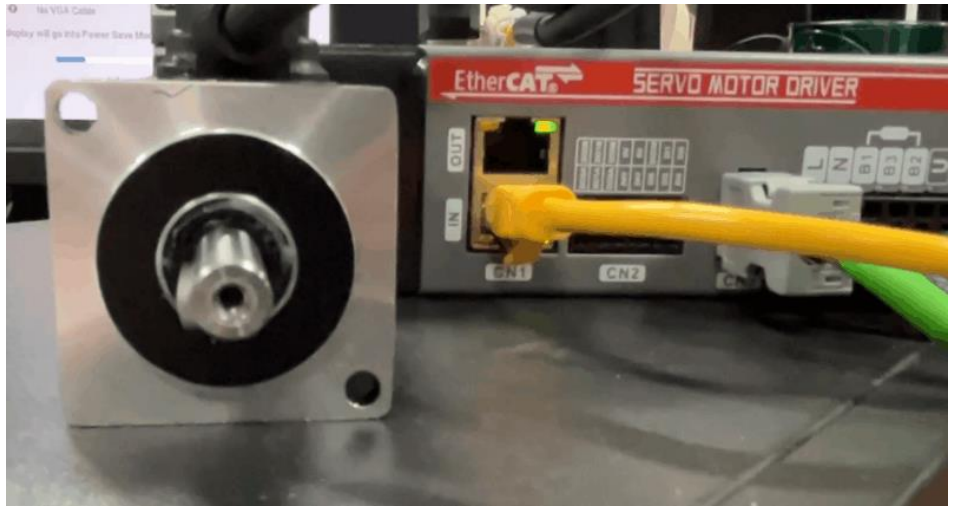
基于S-Function封装EtherCAT主站模块

EtherCAT模块	S-function函数名	功能
初始化	Init	主站初始化
State	Get State	读取从站配置
	Set State	下载从站配置
SDO	SDO Download	服务数据对象下载
	SDO Upload	服务数据对象上传
PDO	PDO Download	过程数据对象下载
	PDO Upload	过程数据对象上传



通过移植EtherCAT协议栈后，将EtherCAT对应**功能API**封装为对应的**S-function**，并加载到Simulink库浏览器中。

EtherCAT主站模块测试结果



EtherCAT主站测试	具体测试	备注
功能测试	网络信息ENI导入	\
	PDO、SDO通讯方式测试	
	系统稳定性测试	
性能测试	500us	带负载/不带负载
	1ms	
	2ms	

模拟工业机器人控制器运行过程中的负载：**CPU**、**内存**、**I/O负载**。拟使用Stress工具模拟负载

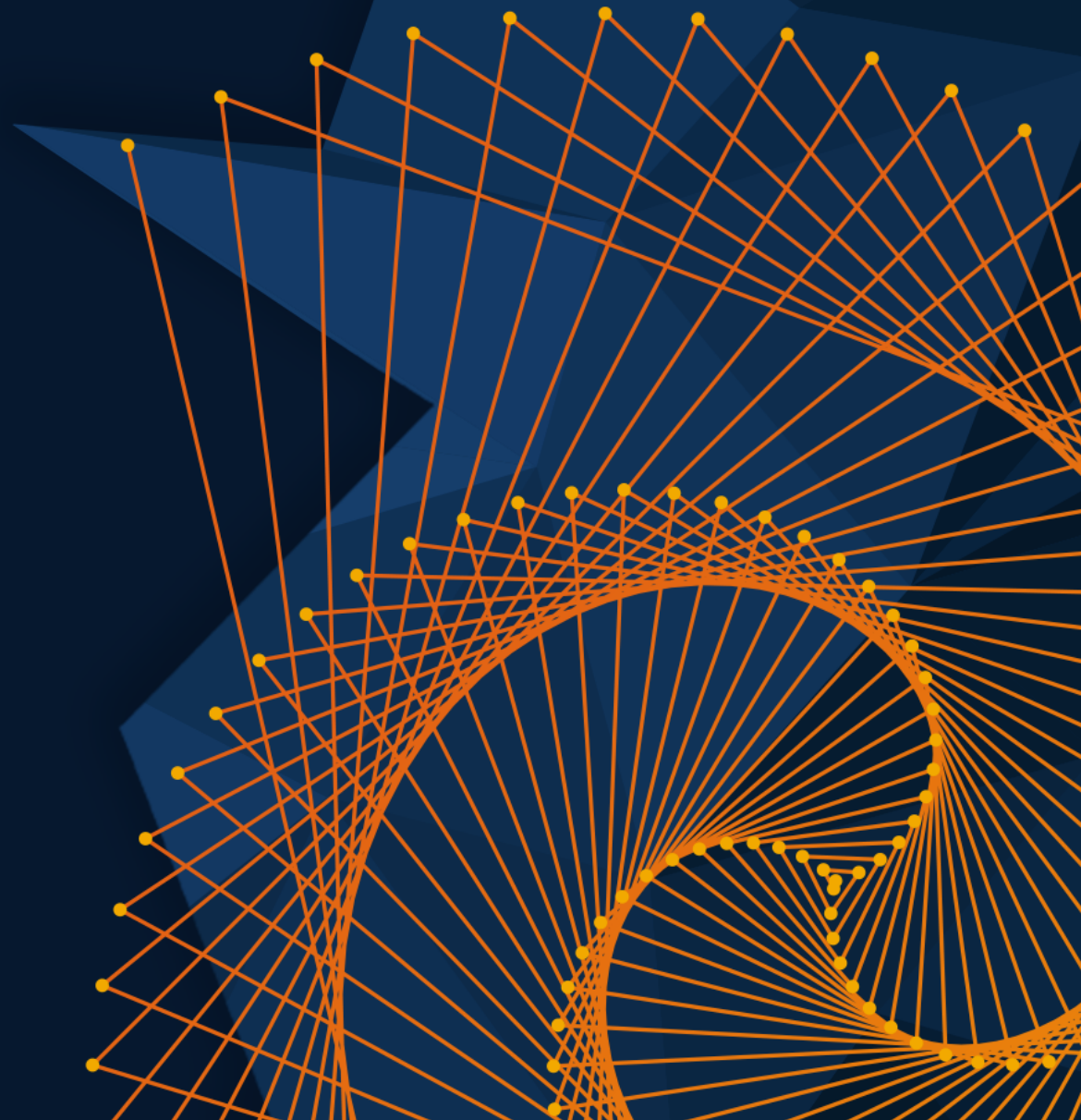
基于以上封装将EtherCAT主站与若干伺服驱动器从站连接，拟使用Stress工具模拟负载，收集**主站通讯最小周期**，**主站抖动**等指标，最终实现快速控制原型平台的实际EtherCAT主站的实时性验证。

总结

- 基于HDL Coder和Simulink Coder快速实现FPGA电机控制
- 开发I/O接口、通信接口，利用LCM、FMI、管道通信等机制打通多平台接口，整合不同层级模型资源
- Simulink多核编程、实时内核保证模型运行实时性
- 基于Simulink的总线协议应用层内嵌，连接更多对象和系统，实现多领域应用

MATLAB EXPO

Thank you



MathWorks®