**嵌入式系统**（Embedded System），是一种“嵌入机械或电气系统内部、具有专属功能的[计算机](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA" \o "计算机)系统”，由于控制功能单一却重要，通常要求及时应对的**[实时计算](https://zh.wikipedia.org/wiki/%E5%AE%9E%E6%97%B6%E8%AE%A1%E7%AE%97" \o "实时计算)性能**。被嵌入的系统通常是包含数字硬件和机械部件的完整设备，举例而言汽车的防打滑刹车就是这样的一个功能集成。现在常见的很多设备都采用简单的嵌入式系统来控制，以达到最高效。

**即时运算**（**Real-time computing**）也称为**实时计算**，是[电脑科学](https://zh.wikipedia.org/wiki/%E7%94%B5%E8%84%91%E7%A7%91%E5%AD%A6" \o "电脑科学)中对受到“即时约束”的[电脑硬件](https://zh.wikipedia.org/wiki/%E7%94%B5%E8%84%91%E7%A1%AC%E4%BB%B6" \o "电脑硬件)和[电脑软件](https://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6%E8%BB%9F%E9%AB%94" \o "电脑软件)系统的研究，即时约束像是从事件发生到系统回应之间的最长时间限制。即时程序必须保证在严格的时间限制内响应。通常即时回应时间会是以毫秒为单位，也有时是以微秒为单位。相比之下，非即时系统是一种无法保证在任何条件下，回应时间均符合实时约束限制的系统。有可能大多数的情形下，非即时系统都可以符合即时约束限制，甚至更快，只是无法保证在任何条件都可以符合约束限制。

在其他领域中也有用到“即时”这个词，但其含义不同：在仿真领域，即时是指“即时时钟同步”，此外在数据传输、多媒体处理和企业系统领域，实时意思是“感觉不到延迟”。

即时软件必须使用一种或多种[同步编程语言](https://zh.wikipedia.org/w/index.php?title=%E5%90%8C%E6%AD%A5%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80&action=edit&redlink=1)、[即时操作系统](https://zh.wikipedia.org/wiki/%E5%8D%B3%E6%99%82%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1" \o "即时操作系统)以及建立在一个实时软件应用程序上的即时网络提供的基本框架。

[防抱死系统](https://zh.wikipedia.org/wiki/%E9%98%B2%E6%8A%B1%E6%AD%BB%E7%B3%BB%E7%BB%9F" \o "防抱死系统)是个即时运算系统的简单例子，在这个系统中的即时即时约束是为了避免车轮锁死，刹车必须释放的时间。即时电脑系统的反应最后期限一般和事件有关，若没能在最后期限前完成，即为即时电脑系统的失效。不论[系统负载](https://zh.wikipedia.org/w/index.php?title=%E7%B3%BB%E7%BB%9F%E8%B4%9F%E8%BD%BD&action=edit&redlink=1)如何，即时电脑系统均需满足最后期限的限制条件。

**专用集成电路**（英语：**Application-specific integrated circuit**，[缩写](https://zh.wikipedia.org/wiki/%E7%B8%AE%E5%AF%AB" \o "缩写)：**ASIC**），是指依产品需求不同而[客制化](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%AE%9A%E5%88%B6" \o "全定制)的特殊规格[集成电路](https://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E9%9B%BB%E8%B7%AF" \o "集成电路)；相反地，非客制化的是[应用特定标准产品](https://zh.wikipedia.org/w/index.php?title=%E6%87%89%E7%94%A8%E7%89%B9%E5%AE%9A%E6%A8%99%E6%BA%96%E7%94%A2%E5%93%81&action=edit&redlink=1" \o "应用特定标准产品（页面不存在）)（Application-specific standard product）集成电路。

专用集成电路是由特定使用者要求和特定电子系统的需要而设计、制造。由于单个专用集成电路芯片的生产成本很高，如果出货量较小，则采用专用集成电路在经济上不太实惠。这种情况可以使用[可编程逻辑器件](https://zh.wikipedia.org/wiki/%E5%8F%AF%E7%BC%96%E7%A8%8B%E9%80%BB%E8%BE%91%E5%99%A8%E4%BB%B6" \o "可编程逻辑器件)（如FPGA）来作为目标硬件实现[集成电路设计](https://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF%E8%AE%BE%E8%AE%A1" \o "集成电路设计)。此外，可编程逻辑器件具有用户可编程特性，因此适合于大规模芯片量产之前的原型机，来进行调试等工作。但是可编程逻辑器件在面积、速度方面的优化程度不如全定制的集成电路。

专用集成电路的特点是面向特定用户的需求，品种多、批量少，要求设计和[生产周期](https://zh.wikipedia.org/w/index.php?title=%E7%94%9F%E4%BA%A7%E5%91%A8%E6%9C%9F&action=edit&redlink=1" \o "生产周期（页面不存在）)短，它作为集成电路技术与特定用户的整机或系统技术紧密结合的产物，与通用集成电路相比具有体积更小、重量更轻、功耗更低、可靠性提高、性能提高、保密性增强、成本降低等优点。

**现场可编程逻辑闸阵列**（英语：Field Programmable Gate Array，缩写为FPGA），它是在[PAL](https://zh.wikipedia.org/wiki/%E5%8F%AF%E7%A8%8B%E5%BC%8F%E9%99%A3%E5%88%97%E9%82%8F%E8%BC%AF" \o "可编程阵列逻辑)、[GAL](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E9%98%B5%E5%88%97%E9%80%BB%E8%BE%91" \o "通用阵列逻辑)、[CPLD](https://zh.wikipedia.org/wiki/%E8%A4%87%E9%9B%9C%E5%8F%AF%E7%A8%8B%E5%BC%8F%E9%82%8F%E8%BC%AF%E8%A3%9D%E7%BD%AE" \o "复杂可编程逻辑器件)等[可编程逻辑器件](https://zh.wikipedia.org/wiki/%E5%8F%AF%E7%A8%8B%E5%BC%8F%E9%82%8F%E8%BC%AF%E8%A3%9D%E7%BD%AE" \o "可编程逻辑器件)的基础上进一步发展的产物。它是作为[专用集成电路](https://zh.wikipedia.org/wiki/%E7%89%B9%E6%AE%8A%E6%87%89%E7%94%A8%E7%A9%8D%E9%AB%94%E9%9B%BB%E8%B7%AF" \o "专用集成电路)领域中的一种半定制电路而出现的，既解决了[全定制](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%AE%9A%E5%88%B6" \o "全定制)电路的不足，又克服了原有可编程逻辑器件门电路数有限的缺点。

目前以[硬件描述语言](https://zh.wikipedia.org/wiki/%E7%A1%AC%E4%BB%B6%E6%8F%8F%E8%BF%B0%E8%AF%AD%E8%A8%80" \o "硬件描述语言)（[Verilog](https://zh.wikipedia.org/wiki/Verilog" \o "Verilog)或[VHDL](https://zh.wikipedia.org/wiki/VHDL" \o "VHDL)）描述的逻辑电路，可以利用[逻辑综合](https://zh.wikipedia.org/wiki/%E9%80%BB%E8%BE%91%E7%BB%BC%E5%90%88" \o "逻辑综合)和[布局](https://zh.wikipedia.org/wiki/%E5%B8%83%E5%B1%80_(%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF)" \o "布局 (集成电路))、[布线](https://zh.wikipedia.org/wiki/%E5%B8%83%E7%BA%BF_(%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF)" \o "布线 (集成电路))工具软件，快速地烧录至FPGA上进行测试，这一过程是现代[集成电路设计验证](https://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF%E8%AE%BE%E8%AE%A1" \o "集成电路设计)的技术主流。这些可编程逻辑元件可以被用来实现一些基本的[逻辑门](https://zh.wikipedia.org/wiki/%E9%82%8F%E8%BC%AF%E9%96%98" \o "逻辑门)[数字电路](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%97%E7%94%B5%E8%B7%AF" \o "数字电路)或者更复杂一些的组合逻辑功能，比如译码器等。在大多数的FPGA里面，这些可编辑的元件里也包含记忆元件，例如[触发器](https://zh.wikipedia.org/wiki/%E8%A7%B8%E7%99%BC%E5%99%A8" \o "触发器)（Flip－flop）或者其他更加完整的记忆块，从而构成[时序逻辑电路](https://zh.wikipedia.org/wiki/%E6%97%B6%E5%BA%8F%E9%80%BB%E8%BE%91%E7%94%B5%E8%B7%AF" \o "时序逻辑电路)。

系统设计师可以根据需要，通过可编辑的连接，把FPGA内部的逻辑块连接起来。这就好像一个电路试验板被放在了一个芯片里。一个出厂后的成品FPGA的逻辑块和连接可以按照设计者的需要而改变，所以FPGA可以完成所需要的逻辑功能。

FPGA一般来说比[专用集成电路](https://zh.wikipedia.org/wiki/%E7%89%B9%E6%AE%8A%E6%87%89%E7%94%A8%E7%A9%8D%E9%AB%94%E9%9B%BB%E8%B7%AF" \o "专用集成电路)（ASIC）的速度要慢，无法完成更复杂的设计，并且会消耗更多的电能。但是，FPGA具有很多优点，比如可以快速成品，而且其内部逻辑可以被设计者反复修改，从而改正程序中的错误，此外，使用FPGA进行除错的成本较低。

**VHDL**，全称**超高速集成电路硬件描述语言**（英语：**[VHSIC](https://zh.wikipedia.org/wiki/VHSIC" \o "VHSIC) hardware description language**），在基于[复杂可编程逻辑器件](https://zh.wikipedia.org/wiki/%E8%A4%87%E9%9B%9C%E5%8F%AF%E7%A8%8B%E5%BC%8F%E9%82%8F%E8%BC%AF%E8%A3%9D%E7%BD%AE" \o "复杂可编程逻辑器件)、[现场可编程逻辑门阵列](https://zh.wikipedia.org/wiki/%E7%8E%B0%E5%9C%BA%E5%8F%AF%E7%BC%96%E7%A8%8B%E9%80%BB%E8%BE%91%E9%97%A8%E9%98%B5%E5%88%97" \o "现场可编程逻辑门阵列)和[专用集成电路](https://zh.wikipedia.org/wiki/%E7%89%B9%E6%AE%8A%E6%87%89%E7%94%A8%E7%A9%8D%E9%AB%94%E9%9B%BB%E8%B7%AF" \o "专用集成电路)的[数字系统](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%97%E7%94%B5%E8%B7%AF" \o "数字电路)设计中有着广泛的应用。VHDL语言诞生于1983年，1987年被[美国国防部](https://zh.wikipedia.org/wiki/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E9%98%B2%E9%83%A8" \o "美国国防部)和[IEEE](https://zh.wikipedia.org/wiki/IEEE" \o "IEEE)确定为标准的硬件描述语言。

在[计算机科学](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6" \o "计算机科学)中，[算法](https://zh.wikipedia.org/wiki/%E7%AE%97%E6%B3%95" \o "算法)的**时间复杂度**是一个[函数](https://zh.wikipedia.org/wiki/%E5%87%BD%E6%95%B0" \o "函数)，它定量描述了该算法的运行时间。这是一个代表算法输入值的[字符串](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2" \o "字符串)的长度的函数。时间复杂度常用[大O符号](https://zh.wikipedia.org/wiki/%E5%A4%A7O%E7%AC%A6%E5%8F%B7" \o "大O符号)表述，不包括这个函数的低阶项和首项系数。使用这种方式时，时间复杂度可被称为是[渐近](https://zh.wikipedia.org/wiki/%E6%B8%90%E8%BF%91%E5%88%86%E6%9E%90" \o "渐近分析)的，亦即考察输入值大小趋近无穷时的情况。例如，如果一个算法对于任何大小为 *n* （必须比 *n0* 大）的输入，它至多需要 5*n*3 + 3*n* 的时间运行完毕，那么它的渐近时间复杂度是 O(*n*3)。为了计算时间复杂度，我们通常会估计算法的操作单元数量，每个单元运行的时间都是相同的。因此，总运行时间和算法的操作单元数量最多相差一个常量系数。

**鲁棒控制**（英语：Robust control）：所谓“鲁棒性”，是指控制系统在一定（结构，大小）的参数摄动下，维持某些性能的特性。是控制理论中的一个分支，是专门用来处理控制器设计时逼近的不确定性。

鲁棒控制方法一般应用于只要在一些集合（特别是[紧集合](https://zh.wikipedia.org/wiki/%E7%B4%A7%E7%A9%BA%E9%97%B4" \o "紧空间)）中存在不确定参数或者扰动的情况。鲁棒控制意在是系统具有鲁棒性，并在存在有界建模误差的情况下使系统稳定。

与[自适应控制](https://zh.wikipedia.org/wiki/%E8%87%AA%E9%80%82%E5%BA%94%E6%8E%A7%E5%88%B6" \o "自适应控制)的对比：鲁棒控制专注于状态，而不是对变量的调整，控制器需要在基于某些变量未知但有界的假设下，才能够有效的工作。

**自适应控制** （Adaptive control）也称为**适应控制**，是一种对系统[参数](https://zh.wikipedia.org/wiki/%E5%8F%82%E6%95%B0" \o "参数)的变化具有适应能力的[控制方法](https://zh.wikipedia.org/wiki/%E6%8E%A7%E5%88%B6%E7%90%86%E8%AE%BA" \o "控制理论)。在一些系统中，系统的参数具有较大的不确定性，并可能在系统运行期间发生较大改变。比如说，客机在作越洋飞行时，随着时间的流逝，其重量和重心会由于[燃油](https://zh.wikipedia.org/wiki/%E7%87%83%E6%B2%B9" \o "燃油)的消耗而发生改变。虽然传统控制方法（即基于[时不变](https://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E6%97%B6%E4%B8%8D%E5%8F%98%E7%B3%BB%E7%BB%9F%E7%90%86%E8%AE%BA" \o "线性时不变系统理论)假设Non-Time-Variant Assumption的控制方法）具有一定的对抗系统参数变化的能力，但是当系统参数发生较大变化时，传统控制方法的性能就会出现显著的下降，甚至产生发散。

需要注意区别的是，虽然同样是为对抗系统参数的不确定性和时变性而设计的，自适应控制与[鲁棒控制](https://zh.wikipedia.org/wiki/%E9%B2%81%E6%A3%92%E6%8E%A7%E5%88%B6" \o "鲁棒控制)有着本质区别。[鲁棒控制](https://zh.wikipedia.org/wiki/%E9%B2%81%E6%A3%92%E6%8E%A7%E5%88%B6" \o "鲁棒控制)是采用过大的控制量来保证受控对象的状态向收敛方向移动。其优点是，只要参数的改变程度处在控制器的设计范围之内，系统就能保持稳定。而缺点在于，过大的控制量会导致系统发生“抖动”（Chattering），从而导致系统跟踪精度有限或驱动机构磨损加剧。而自适应控制则是通过逐步逼近系统特性来保证跟踪精度，其缺点是，在开始阶段不一定能保证稳定，而且往往需要运行一段时间才能实现精确跟踪输入量。其优点是在正常运行时系统可以比较平稳地实现精确跟踪。