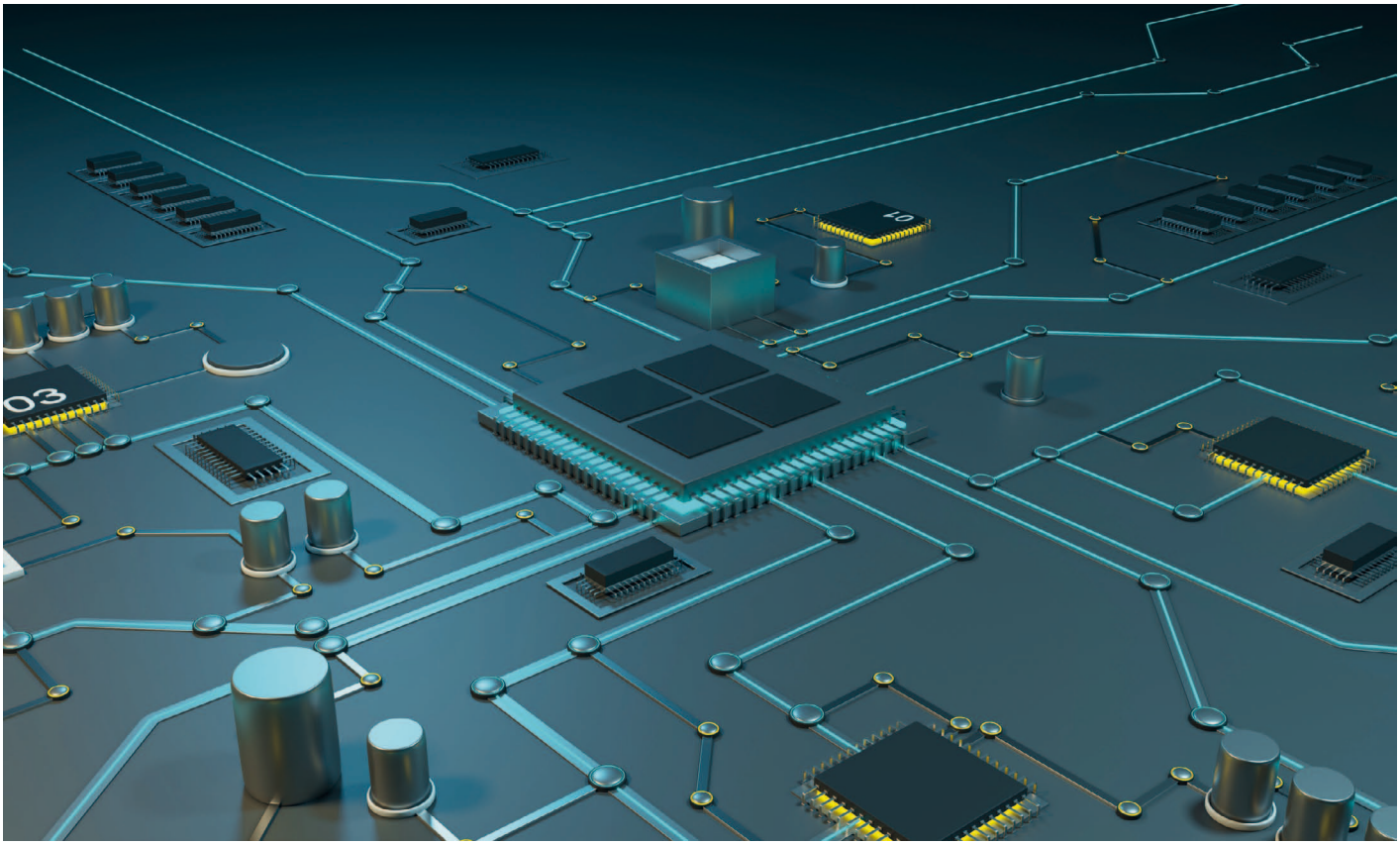
Verarbeiten oder steuern?

Mikroprozessoren, Mikrocontroller und ihre Peripherie

Für jedes neue Design braucht es einen Mikrocontroller oder Mikroprozessor. Bei der Auswahl gibt es einige Aspekte zu beachten. Ein Überblick über MCU, MPU und heterogene Architekturen.



Autor:
Hao Wang,
Product Manager Digital
Rutronik
www.rutronik.de

Betriebssystem

Für umfangreiche, rechenintensive Industrie- und Consumer-Applikationen, die auf einem Betriebssystem wie Linux oder Android basieren, mehrere High-Speed-Verbindungen benötigen oder Benutzeroberflächen mit großem Funktionsumfang bieten sollen, ist ein Mikroprozessor (MPU) die erste Wahl. Denn die meisten Mikrocontroller (MCU) haben statt eines Betriebssystems nur bare metal Programmierung, die Programme praktisch ohne menschliches Eingreifen durch sequentielle Verarbeitungsschleifen und Zustandsmaschinen laufen lassen. Viele hoch-performante MCUs können jedoch ein Echtzeitbetriebssystem (Real-Time-Operation-System, RTOS) wie FreeRTOS ausführen, um für Applikationen, die eine harte Echtzeit erfordern, Vorgänge zeitnah deterministisch zu beantworten.

RTOS und Linux

Embedded Linux ist als universelles Betriebssystem mit viel freier Software, umfangreichem Hardware Support inklusive und einem wachsenden Ökosystem sehr erfolgreich. Weiterer Pluspunkt: Es sind weder Nutzungs- noch Lizenzgebühren fällig. Für eine Applikation, die mit Embedded Linux läuft, braucht es jedoch mindestens 300 bis 400 DMIPS (ARM-Dhrystone MIPS) – und damit eine MPU. Denn MCUs bringen hierfür weder genug Rechenleistung noch Speicherplatz mit.

RTOS

Mit einem RTOS kann auch eine hoch-performante MCU, die mindestens 50 DMIPS bietet, die Komplexität einer solchen Applikation bewältigen. Denn die klassischen RTOS sind so schlank konzipiert, damit sie auf einer MCU laufen. Das ist dann sinnvoll, wenn es um Hardware-Echtzeit-Computing geht, wie z. B. bei einem Antiblockiersystem für Fahrzeuge, wo eine zu große Latenz der Antwort tödliche Folgen haben kann. Auch wenn eine größere Anzahl an Funktionen, Interrupt-Quellen und Standard-Kommunikationsschnittstellen unterstützt werden müssen, empfiehlt sich eine MCU mit RTOS.

Speicher

Ein anderer Hauptunterschied zwischen MPUs und MCUs besteht im Regelfall darin, dass MPUs für das Speichern und Ausführen des Programms auf einen externen Speicher zurückgreifen und

nicht – wie MCUs – auf einen eingebetteten Flash-Speicher. Bei MPUs befindet sich das Programm in der Regel auf einem nichtvolatilen Speicher wie eMMC, Serial Flash o. ä.. Beim Start wird es in einen externen DRAM geladen und hier ausgeführt. Beide, sowohl der DRAM als auch der nichtflüchtige Speicher, können einige Hundert Megabyte oder sogar Gigabyte groß sein. Das heißt, MPUs sind hinsichtlich der Speichergröße kaum limitiert. Als ein Nachteil kann das Design des PCB-Layouts durch den externen Speicher aufwendiger werden.

Aktuelle hoch-performante MCUs bieten maximal 2 Mbyte Programmspeicher, was für manche Applikationen zu knapp sein kann. Da das Programm aber auf einem On-Chip-Speicher liegt, bieten sie wiederum den Vorteil, dass Start-up und RESET deutlich schneller ablaufen.

Rechenleistung

Die Rechenleistung ist ein klassisches Auswahlkriterium. Allerdings verschwimmen die Grenzen zwischen MPUs und MCUs in dieser Hinsicht immer mehr. Das wird deutlich, wenn man beispielsweise die ARM-Architektur als eine der am stärksten verbreiteten im MCU- und MPU-Markt betrachtet. ARM bietet verschiedene Prozessor-Architekturen für unterschiedliche Anforderungen:

Cortex-A bietet die höchste Leistung von allen und ist bereits optimiert für umfangreiche Betriebssysteme (Operating Systems, OS). Sie kommen vor allem in leistungsstarken Geräten zum Einsatz, z. B. Smartphones oder Server.

Cortex-M sind kleiner mit mehr on-Chip Peripherie bei allerdings geringerer Energieaufnahme - und damit optimiert für Embedded Anwendungen

Um die Leistung verschiedener Prozessoren zu vergleichen, ist als Benchmark Dhrystone üblich. Nach diesem liegt der Unterschied zwischen einer typischen kostengünstigen MCU mit 30 DMIPS und der MCU mit der höchsten aktuell verfügbaren Leistung mit eingebettetem Programm-Flashspeicher bei 1027 DMIPS. Im Gegensatz dazu fangen MPUs bei knapp 1000 DMIPS erst an.

Energieverbrauch

MCUs punkten mit einem Energieverbrauch, der um ein Vielfaches unter dem von MPUs liegt. Obwohl MPUs Energiespar-Modi bieten, ist ihr Verbrauch dennoch deutlich höher als der einer typischen MCU. Zudem erschwert es der externe Speicher, die MPU in einen Energiespar-Modus zu versetzen. Für Ultra-Low-Power-Applikationen, bei denen eine lange Batterielaufzeit im Vordergrund steht, und die keine oder eine nur wenig genutzte Benutzerschnittstelle haben, kommen deshalb vorrangig MCUs zum Einsatz, wie z. B. in Konsumerelektronik oder Smart Meter.

Konnektivität

Die Mehrzahl der MCUs und MPUs bringen alle gängigen Peripherie-Schnittstellen mit. Sind jedoch ultraschnelle Peripheriegeräte gefordert, sucht man die entsprechenden Schnittstellen, wie z. B. Gigabit Ethernet bei MCUs vergebens. Bei MPUs sind sie jedoch praktisch Standard. Das ist durchaus sinnvoll, da MCUs die hier anfallende Datenmenge kaum verarbeiten können. Denn eine Schlüsselfrage lautet: Stehen genug Bandbreite und Kanäle zur Verfügung, um den aufkommenden Datenverkehr zu bewältigen?

Echtzeit-Leistung

Geht es um Echtzeit-Leistung, ist eine MCU definitiv die Komponente der Wahl. Durch ihren Prozessorkern, den eingebetteten Flash und die Software (RTOS oder bare metal OS) erfüllt die MCU Echtzeitaufgaben ausgezeichnet. Da die Cortex A-MPUs „Pipelining“ beherrschen, ist hier mit zunehmender Tiefe der Pipeline mit höheren Latenzzeiten bei Sprüngen und Interrupts zu rechnen. Harte Echtzeit ist aufgrund des Multitasking des OS mit MPUs nur schwer realisierbar.

System-Basis-ICs

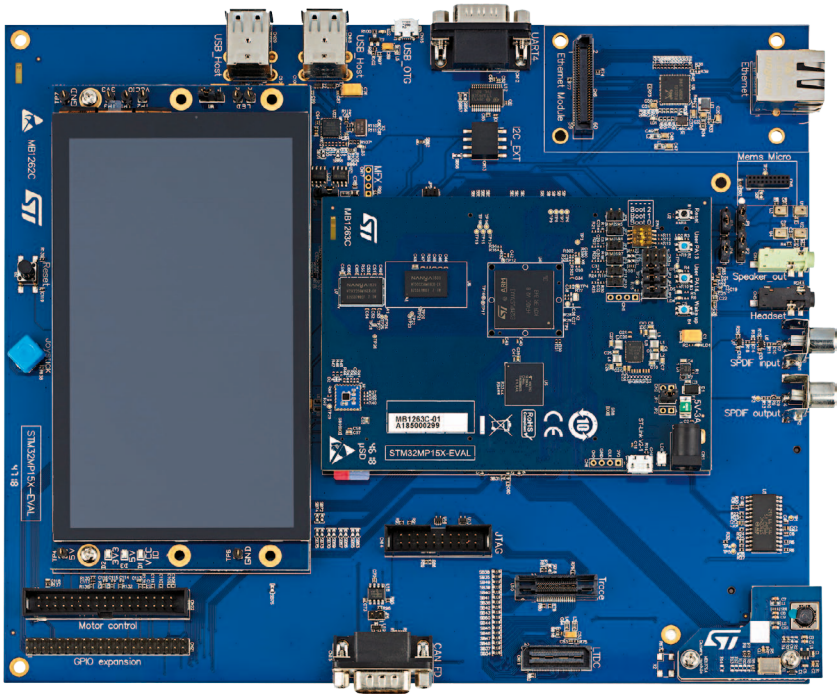
Eine Stromversorgung ist in MCUs bereits integriert, sie benötigen nur eine einzelne Spannungsschiene. MPUs hingegen benötigen mehrere, um den Core und andere Komponenten zu versorgen. Diese wird zum Beispiel durch spezielle ICs, sogenannte SystemBasisChips, die nur konfiguriert werden müssen, zur Verfügung gestellt.

Vorteile Heterogener Architekturen

Allerdings verschwimmt die Grenze zwischen MPU und MCU zusehends. Das eröffnet die Möglichkeit, die Vorteile beider Welten zu kombinieren. So kann auf MPU-Kernen zum Beispiel ein OpenST-Linux ausgeführt werden, während der M4-Kern ein FreeRTOS ausführt, um die Echtzeit- und Sicherheitsanforderungen wie etwa eine Prozessüberwachung zu erfüllen. Durch diese Trennung lassen sich alle Features des Betriebssystems wie z. B. HMI/GUI mit TouchControl und Netzwerk-Kommunikation nutzen, ohne auf die Sicherheitsanforderungen der Gesamtapplikation einzugehen. Heterogene Architekturen haben ebenfalls einen Vorteil, wenn es um die Latenzzeit beim Startup oder den Energieverbrauch geht, da jederzeit der „geeignetere“ Prozessor verwendet werden kann. So kann zum Beispiel der M4 durch seinen Retention-RAM innerhalb von Millisekunden aufwachen, während der A7 bis zu einer Sekunde WakeUp-Latenz hat. Das ermöglicht es, beispielsweise die MPU im Stop-Mode zu halten während der MCU-Teil eine AD-Wandlung durchführt, was im Vergleich zur RUN/RUN-Konfiguration etwa dreiviertel der Energie einspart.

Fazit

Wann eine MCU und wann eine MPU die bessere Wahl ist, lässt sich nicht pauschal sagen, hier gilt es immer, die verschiedenen Aspekte auch gegeneinander abzuwägen. Als ganz grobe Richtlinie gilt:

- MCUs sind optimal für eher kostengünstige und mobile Anwendungen, bei denen der Energieverbrauch im Vordergrund steht, sowie für Applikationen mit Echtzeit-Anforderungen.

- MPUs eignen sich perfekt für rechenintensive Applikationen, die mit einem Betriebssystem laufen, und die Hochgeschwindigkeits-Schnittstellen benötigen. Spiele und sonstige Grafik-intensive Anwendungen werden über spezielle MPUs abgebildet. ◄