Low-Power Design Using NoC Technology

By Linley Gwennap
Principal Analyst

May 2015

The Linley Group

www.linleygroup.com

Low-Power Design Using NoC Technology

By Linley Gwennap, Principal Analyst, The Linley Group

Network-on-a-chip (NoC) technology is not just for high-performance SoC designs. The size and power of the NoC can scale down to accommodate even very small and low-power processors. Furthermore, the NoC helps automate the chip's power management. The NoC can also simplify designing a single die that produces multiple end products. This white paper describes how a NoC can achieve these advantages, using TI's CC26xx microcontroller as a case study. The Linley Group prepared this paper, which Arteris sponsored, but the opinions and analysis are those of the author.

Network-on-a-chip (NoC) technology is usually associated with large, high-performance system-on-a-chip (SoC) designs that combine CPU, graphics, audio, and video cores with DRAM, flash, and a variety of I/O controllers. In these designs, the NoC provides a high-bandwidth pathway to move data among the various cores, ensuring high performance and avoiding congestion. To simplify these tasks, the NoC converts data to packets that can be prioritized and transmitted through the network. NoC vendors typically provide tools to help configure the interconnect to provide the necessary bandwidth while reducing the die area and power of the NoC, which can be configured as a ring, a tree, a multilayer design, or some combination of the above. Popular mobile and embedded processors from Cavium, Freescale, Qualcomm, Samsung, and others use licensed NoC technology.

Designers of microcontrollers and similar low-end processors generally don't consider using a NoC. They may think their chip is too simple or too low in performance to need a NoC. Or they may think that a NoC will add too much cost and power to their design. These designers should think again, because some NoC designs can help reduce chip-level cost and power, even for low-end devices. They provide flexibility for quickly producing multiple products from the same design.

As a case study, this white paper discusses the new SimpleLink CC26xx processor family from Texas Instruments. This chip was designed using FlexNoC IP from Arteris. Targeting the Internet of Things (IoT), the CC26xx combines an ARM microcontroller (MCU) and a wireless interface on the same chip. TI offers a family of products that offers different MCU configurations and different wireless protocols (e.g., Bluetooth, 6LoWPAN, Zigbee). The FlexNoC IP helped TI achieve best-in-class power in a tiny processor that sells for only a few dollars.

SimpleLink CC26xx Overview

Although nominally a microcontroller, the CC26xx is far from a simple chip. As Figure 1 shows, it includes several major blocks, starting with a Cortex-M3 CPU. The main CPU subsystem connects to SRAM, Flash, and ROM. The processor also includes a radio frequency (RF) core that includes a second CPU (Cortex-M0 plus RAM and ROM) to implement the wireless protocol as well as signal-processing blocks. A sensor control engine aggregates incoming analog and digital sensor data before passing it to the main

CPU. The chip also has a crypto engine and a DMA engine as well as a variety of serial ports, timers, and general-purpose I/O. It fits into a 4mm QFN32 package.

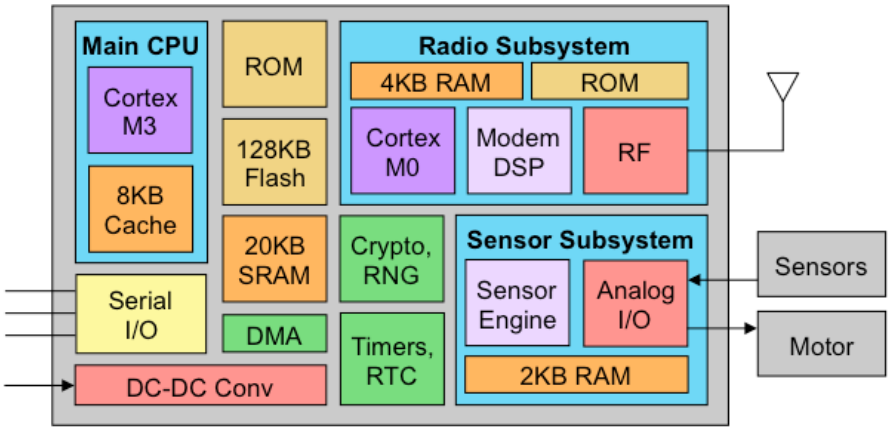

Figure 1. Block diagram of TI SimpleLink CC26xx processor. This IoT processor combines a Cortex-M3 microcontroller with a complete radio subsystem.

The main CPU operates at up to 48MHz, but it can run at lower speed to save power and is kept in sleep mode as much as possible. Using its own CPU and 4KB of SRAM, the RF core can autonomously transmit and receive wireless data, waking the main CPU only when necessary. The Cortex-M0 can be programmed for different wireless protocols, although the analog circuitry is tuned for operation in the unlicensed 2.4GHz band. The sensor controller has its own 2KB of SRAM for buffering data, which can come from a multichannel A/D converter or digital sensor interfaces such as SPI or I2C.

The CC26xx targets IoT applications such as smart lighting, connected appliances, and security systems, which require low cost and low power but need just enough CPU performance to monitor a few sensors or control a few actuators. These devices often connect wirelessly, to simplify installation, and may use various standard or proprietary protocols. The processor can also be used in wearable devices such as fitness trackers, which often connect to a smartphone via Bluetooth. Industrial, retail, and medical applications that require low-power wireless connections can also benefit from the CC26xx.

Implementing Power Management Using a NoC

These applications require maintaining the lowest possible power at all times. The CC26xx design accomplishes this using a variety of offload engines, including the RF core, sensor controller, and DMA engine. The challenge for the designer is to ensure that only the necessary circuitry is powered at any given time, using clock gating and voltage gating to power down the rest of the chip. In a traditional design, clock and power gating must be added to the design manually, with the control signals routed to a power manager that then needs to be programmed to handle a number of power states and safely transition from one state to another without losing data.

FlexNoC greatly simplifies this design process. Designers use the NoC configuration tool to partition the chip's functions into any number of clock and power domains. The NoC automatically handles all domain crossings, moving data from one domain to another, regardless of their clock speeds. This level of automation allows the designer to implement very fine grained partitioning, allowing each block to work at its optimum power efficiency and then be powered down when not needed. Furthermore, FlexNoC supports voltage domains, which simplifies dynamic voltage and frequency scaling (DVFS) in order to individually minimize the active power consumption of each IP.

As Figure 2 shows, the CC26xx has two voltage domains, each divided into several power domains. Designing and managing this many domains would have been very difficult without the automation delivered by FlexNoC. At best, using standard tools for such a complex design would have delayed the product's time to market.

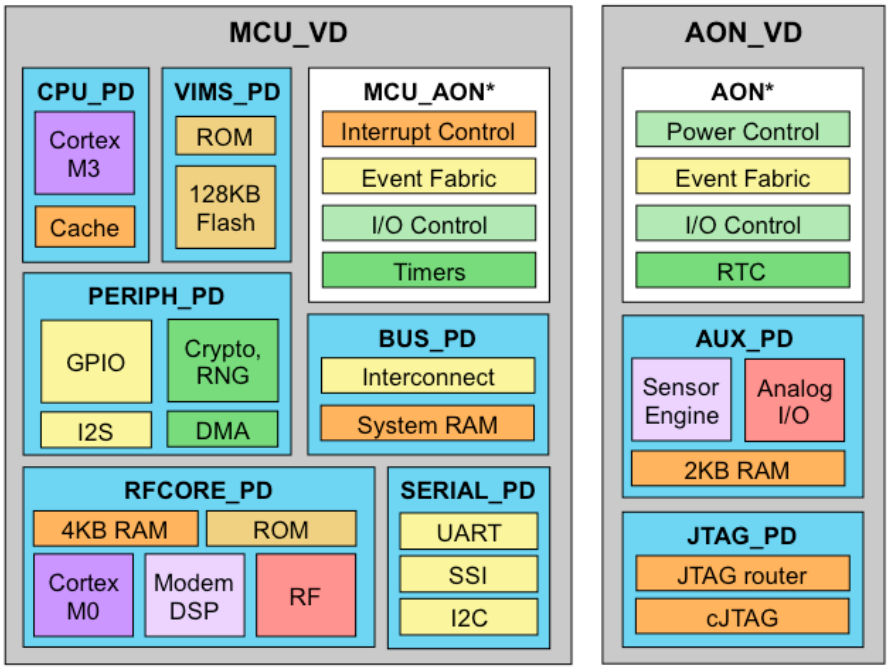

Figure 2. Power domains in TI CC26xx processor. The processor has two voltage domains in addition to the always-on logic (marked with *). Each voltage domain has several power domains. (Source: TI)

The NoC has power control logic that implements power state transitions. For example, if the SoC power manager wants to power down a particular IP core that is a slave device, it simply notifies the NoC. In this case, FlexNoC will disable internal access to that port to prevent any new requests from reaching the IP. Then it will wait for the device to finish responding to all pending requests. At that time, it signals the power manager that it is safe to power down the IP core. When the power manager requests to shut down a master device, it must first signal that device to stop sending new requests and prepare to shut down. Then it can follow the same procedure to wait for the device to finish transmitting and receiving data before it powers down the core. Thus, the power manager need send only simple requests to enable or disable a particular IP core; FlexNoC handles the rest of the process consistently and reliably.

The NoC itself can also be configured into any number of clock and power domains to minimize its own power. For example, the portion of the NoC that connects to the main CPU is on the same power domain as the CPU itself. When the NoC powers down the main CPU, it would also flush all traffic from that section of the interconnect, then safety shut it down while continuing to service traffic to the rest of the chip. Within a power domain, FlexNoC uses two levels of clock gating to shut off any unused elements on a cycle-by-cycle basis. For example, if a NoC port is not sending or receiving data for a particular cycle, it is not clocked on that cycle. This clock gating reduces clock-tree power.

By comparison, many SoC designers don't power down their internal buses unless the entire chip is asleep. Traditional buses are unable to shut down one portion at a time. Even if the bus can handle this, designers often don't know in what order the IP blocks will connect to the bus in the physical layout, so it is difficult to know when it is safe to shut down part of the bus. Thus, they simply leave the bus powered up.

A final power benefit is that the NoC allows designers to choose a topology that minimizes the number of elements that need to handle a particular traffic flow. Since the NoC clocks only the elements along the active connection path, other blocks are not clocked, saving power.

NoC Provides Design Flexibility

Many chip vendors today sell multiple products (SKUs) based on the same physical die. This approach is becoming more common with the increasing cost of physical design and multimillion-dollar tapeout fees. Using multiple SKUs, the vendor can maximize its revenue from the full-function product while creating reduced-function versions that meet the needs of a specific application at a lower price.

For example, TI sells three products—CC2630, CC2640, and CC2650—that are identical except for the wireless standards supported. The CC2630 also omits the digital audio (I2S) interface that appears on the other models. The multistandard CC2650 carries a list price of $6.30, whereas TI drops the price of the Bluetooth-only CC2640 to $2.70 to appeal to designers of wearable devices.

Designing a single die that can be sold in multiple configurations can be complicated. Ideally, a particular function is not just disabled but clock- or voltage-gated as well, eliminating its power. As discussed above, FlexNoC allows designers to easily create power domains, so individual blocks can be disabled at zero power while the rest of the system remains functional. The NoC will also turn off its port that connects to the disabled block as well as any portion of the interconnect that is only used by the disabled block(s).

The NoC also controls the memory map and security domains that allow software to access the peripheral blocks. It creates the memory map at boot time, allowing it to work around any disabled blocks. It also provides hardware firewalls to disable unauthorized access to secure peripherals. When an IP block is disabled, FlexNoC will either trap accesses to that block's memory-mapped registers or make the accesses silently

ineffectual. FlexNoC can even reconfigure the memory map to eliminate any holes left by disabling blocks, so the customer does not perceive that the product has reduced functionality.

Minimizing Die Area and Manufacturing Cost

A NoC is more complex than a bus. Whereas a bus (or crossbar) negotiates a link between two nodes and then transfers data directly, the NoC converts all data into packets before moving them across the network. The NoC also requires some control logic to manage and prioritize the transfers and to enable and disable sections of the network. These extra capabilities require some amount of logic, so many designers assume a NoC will consume more die area than a traditional bus.

But a NoC has advantages that can overcome this extra logic. Transferring packet data requires fewer wires than a traditional bus. For example, control information is encoded in the packets rather than on separate wires. In addition, data paths can be narrower because they can be sized for sustainable bandwidth rather than the maximum required on a given cycle, as in a crossbar. Narrower data paths require narrower FIFOs, which consume less logic and die area.

This advantage is greater in a network with different clock rates. Many SoCs have multiple levels of buses, each operating at different clock rates and often different protocols, and these levels require complex bridge logic to connect them. In contrast, the NoC implements a single protocol, and packet data can move from a high-speed segment to a slower segment through a simple adapter.

The NoC also provides greater flexibility that helps designers to reduce its area. Arteris provides tools that model the amount of bandwidth needed at each segment of the NoC. These tools allow designers to easily configure and reconfigure the various segments, adding or removing connections to find the optimal topology that still meets the bandwidth requirements of each IP core. For example, one segment may be provisioned at 32 bits wide to handle high-bandwidth cores, whereas segments with only low-speed cores can be reduced to 16 bits or less. Without these tools, designers often overprovision their interconnects to avoid bottlenecks, but such designs use more area (and power).

This configurability includes not just the size but the functions of the interconnect. For example, FlexNoC interfaces normally support burst mode to improve throughput, but for IP blocks that don't require or don't implement burst mode, support logic for this feature can be removed from the interface of that port. This approach allows designers to implement only the logic that they need. Once the designer selects the desired features, the tools automatically configure the NoC logic.

This automation simplifies the layout process as well. In fact, the FlexNoC tools make it so easy to move blocks around, designers can try several different layouts to find the optimal configuration. An optimal layout (for example, grouping low-speed blocks) can greatly reduce the area of the NoC. This flexibility also allows designers to try different

physical layouts to optimize area and timing, which ultimately improves either performance or power for the end application.

NoCs Address Wide Range of Designs

NoCs are not just for high-performance SoC designs. The size and power of the NoC can scale down to accommodate even very small and low-power processors. Although the NoC requires some extra logic compared with a traditional bus structure, its efficiency, flexibility, and configurability can ultimately reduce total die area and power for the interconnect. The automation of the FlexNoC tools can further improve performance and utilization of the chip by enabling the designers to try different physical layouts.

Furthermore, the NoC provides advantages in automating the chip's power management. It can control the power state of all IP blocks and ensure that they are safely shut down and cleanly restarted. This simple and unified control mechanism simplifies the power-management firmware. Finally, the NoC can simplify designing a single die that produces multiple end products, disabling specified capabilities at boot time and even revising the memory map for each product.

TI's CC26xx design proves these capabilities. Using FlexNoC, the product operates at just 2.9mA when the CPU is running at full speed. Even more impressive, the chip draws just 0.55mA in idle mode, which disables the main CPU but still enables the NoC to transfer data among most other IP blocks. TI created three different products from the same base design and is able to sell the chip for less than $3. Whether your design targets a list price of $3 or $300, a network-on-a-chip could help achieve your performance and power goals while reducing design time.

Linley Gwennap is principal analyst at The Linley Group and editor-in-chief of Microprocessor Report. *The Linley Group offers the most comprehensive analysis of microprocessor and SoC design. We analyze not only the business strategy but also the internal technology. Our in-depth reports also cover topics including embedded processors, mobile processors, IoT processors, and processor IP cores. For more information, see our web site at www.linleygroup.com.*