

High Performance, Low Cost: 'Solution-on-Chip' Architectures for Today's Intelligent Devices

White Paper

Abstract

For OEMs of intelligent electronic devices, the demand for network connectivity and the increased functionality networking can deliver to these products is driving a new wave of innovation – one that represents the vanguard of an important and potentially lucrative new market. The challenges associated with OEMs adding network connectivity to their electronic devices, however, introduce many unique elements and choices to the design process.

In the Intelligent Networked Device (IND) market– where the embedded systems, networking and semiconductor industries converge – design teams must select hardware platforms that can handle several connectivity media, I/O and internal controllers, while still having plenty of bandwidth to run an application. Additionally, design engineers working on an embedded application must not only take performance into account when choosing a microprocessor, but also build a network-connected device with flexibility and scalability for the future.

Silicon Enhancement

Increasingly in an era of technological innovation and fierce competition between companies, it is almost unthinkable for most OEM design teams to develop a unique platform where the product's functionality will never be enhanced. Therefore, engineers are forced to design to platforms with an eye to future enhancements and modifications in an effort to stay ahead of the competition. Some design projects, light switches or sensors for example, demand very low Bill of Materials (BOM) costs, as little development time as possible, and may have very low-level connectivity needs requiring nothing but a basic TCP/IP stack and stripped-down Real-Time Operating System (RTOS) ported to the chip. In this scenario, an 8- or 16-bit microcontroller hardware platform is sufficient for a design team's needs.

But when an OEM faces a design challenge, such as a network-ready, synchronized PDA docking cradle that must live on a busy network, and where the enterprise environment may well differ from customer to customer, the solution is not as simple.

These more advanced applications, which require non-hardware features such as additional protocol stacks, a full-function RTOS and use of a complete development environment, mean the design engineer must not only focus on the successful implementation of low-cost, high-performance, flexible design architectures, but also note critical components that allow for growth without outpacing the platform. What's more, competitive pressure dictates the need to complete the design rapidly, without compromising performance integrity. In these instances, which describe the fast-growing intelligent device market, 32-bit processor are the intelligent choice.

What makes a product flexible and provides the opportunity for the product to be successful? Is it processor performance or support for controllers, such as Universal Serial Bus (USB), serial, Ethernet, memory or Bluetooth®? Or could success be related more to what the product or platform will offer in the future?

An OEM may select a low-powered processor in the short term to save on BOM costs, but the reality is that in the medium to long term hardware re-designs are costly – and drain design engineers' capacity – when upgrading a product's functionality. Intellectual property and future-proof design is where differentiation is found. It is critical to have a flexible platform that provides enough "headroom" for software enhancement – or even simple modifications – to the base hardware offering. If such a platform can also speed time-to-market by supplying pre-integrated hardware and software, so much the better. 32-bit processors are the foundation for this approach.

Beginning a network-enabled device project takes careful planning. Design engineers must address some concerns that can often be overlooked during the product development cycle, and more importantly the component selection process if not designing for future considerations.

The Hand that Docks the Cradle

In the case of Portsmouth, a Boise, Idaho-based developer of docking cradles for handheld computing systems, the challenge is in designing and maintaining high-performance product lines for the fast-moving PDA industry. Thus, for Portsmouth, platform flexibility is tantamount to shorter design cycles and faster time-to-market.

Portsmouth's enhanced docking cradles need to provide rapid access to corporate networks and the Internet to allow users to synchronize personal information management (PIM) data – such as calendars, contact management information, and e-mail – via dial-in service or through the Local Area Network (LAN). Additionally, the cradles required device support for laptop and handheld computer manufacturers such as Palm, Symbol, Handspring, IBM and others.

These network demands required standards-based technologies supporting serial bus, modem and 10/100 Ethernet, while also remaining customizable to meet specific network environments. Finally, Portsmouth's products needed dynamic IP addressing and had to be configurable to synchronize up to 32 terminals at a single station. Not a typical application for embedded networking, and not something that could be accomplished with an 8- or 16-bit SoC platform.

With Ethernet connectivity mandatory in a majority of new product designs ranging from Portsmouth PDA cradles to Voice-over-IP (VOIP) handsets, to retail point-of-sale terminals and building access controllers, a new consideration needs to be placed on bandwidth: bandwidth is not only going to be measured for processor application requirement, but also for the high-speed Ethernet connection.

“System-on-Chip Off the Old Block”

The bandwidth example illustrates the careful consideration needed when determining what networking architecture to use. The days of discrete logic designs utilizing processor cores with external hardware ranging from memory and Direct Memory Access (DMA), to standard Universal Asynchronous Receiver Transmitter (UART) or Ethernet Media Access Controls (MAC) have passed for reasons such as board space, cost and development time. Most new designs are now based on System-on-Chip (SoC) technology. These specialized breeds of application-Specific Integrated Circuits (ASICs) provide the baseline for a complete product. The critical choice, however, is determining what SoC technology is right for the immediate task, as well as the software enhancements that will occur in the later versions of the product.

When choosing an SoC, one should start by asking:

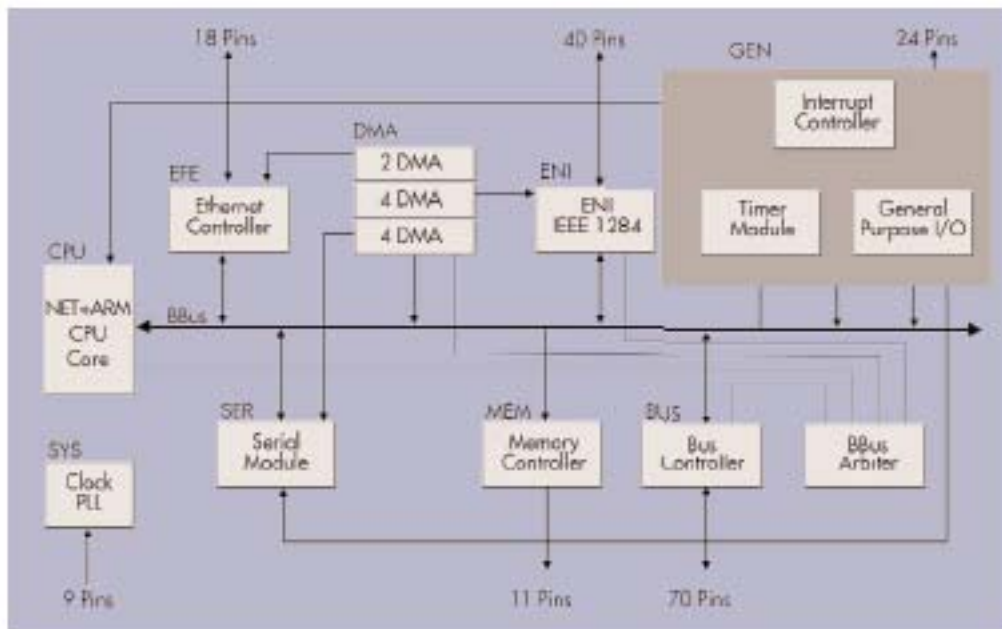
- What type of processor core is embedded in the ASIC?
- How does the SoC handle the high-speed interfaces it supports?
- How can one program the SoC?

With embedded Ethernet control, the critical question here is, how does the SoC handle the real data? Some ASICs may require that the processor core physically move all traffic from the MAC to memory. If this is the case on a 10BaseT network, there may be enough memory in the processor to handle the Ethernet connection and the primary application.

With 100BaseT network speeds, however, this is not the case. If a dedicated subsystem, such as a DMA controller, is not responsible for the repetitive data movement from the Ethernet front-end to the memory system, and vice versa, the application performance will most certainly suffer. The DMA controller allows data transfer from one memory area to another without having to go through the central processing unit. Higher performance processors, such as 32-bit processors with DMA channels, can transfer data to and from devices much more quickly than those in which the data path goes through the main processor.

Controlling the Means of Production

Another consideration along the lines of SoC performance is the concept of shared resources with the SoC itself. In the case of a DMA controller handling the interface communication and the processor core focusing on the application, there still needs to be some method of getting their respective data and operands to and from a common memory architecture. This is accomplished with a system bus. In the case of an internal memory and bus controller, the system bus can be viewed as two different, but functionally similar, buses. In the case of memory mapped peripherals, when access is made to a device via the application code, the memory controller will decode what external device the memory controller set-up is actually accessing. The data or instructions will then be either read or written to the device.

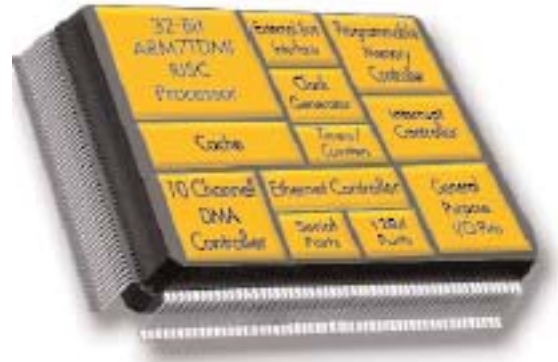


The system bus moves data to and from a common memory architecture.

Now, suppose the DMA controller, at the same instant, is trying to move data coming in the Ethernet receive channel to memory. While not initially obvious, there is bandwidth resource contention. Both the memory controller and the DMA controller need to utilize the bus to move their respective data. How the architecture deals with this contention will directly impact the performance of the entire product. Eight- and 16-bit processors simply cannot handle these demands efficiently enough to function effectively as the core platform.

The silicon vendor can sometimes alleviate potential issues such as the bandwidth contention. SoC designs are much like generic hardware and software designs in that a team of engineers, using a process involving code review and lengthy simulation cycles, develops these designs. So, potential issues have been analyzed and workarounds implemented.

In the case of the NET+ARM® 32-bit RISC processor from Digi International – a SoC utilizing the ARM7TDMI core with internal memory, DMA, Ethernet, serial, and bus controllers – a limitation is implemented in the SoC to ensure no current bus master can hold the system bus for more than four cycles. This will allow the ARM core to transfer up to four long words (during a burst cycle) while allowing the DMA controller (another bus master) to transfer the same amount of data on the next cycle.



The NetSilicon NS7520 from Digi International is a 32-bit RISC processor that features an ARM core with internal memory, DMA, Ethernet, serial and bus controllers.

In the NET+ARM example, one will also find a cache system capable of storing instructions, thereby allowing the ARM to continue to fetch instructions while the DMA is transferring data. In the case where the ARM does not need to utilize the bus because of its cache, both bus masters are running at 100 percent. The bus controller will still monitor the potential bus masters to make sure a different bus master besides the current one does not need the bus. Until another bus master need the bus, back-to-back cycles can occur with the current bus master.

Riding the Bus Downtown

Another potential issue with performance relates directly to bus width. With 10BaseT embedded applications, 8- or 16-bit devices may be suitable for the network speed, combined with the application at hand. Most of these devices are geared for providing network connectivity to low-level embedded devices. The intensity of the applications running on the device, as well as the real-time requirements, may not be as rigid. What makes them unique is their cost. Being able to provide Ethernet connectivity with some level of processing is appealing in some applications.

Peeling back the onion, however, usually leads to severe product roadmap limitations. It must also be noted that even though a particular product doesn't need the data rate offered by 100BaseT, it may still need to support 100BaseT as most installed networks are 100BaseT today. Moreover, a 10BaseT device requires a switched port to connect to a 100BaseT network. The cost of a switched port could be almost as much as the cost of the device being attached.

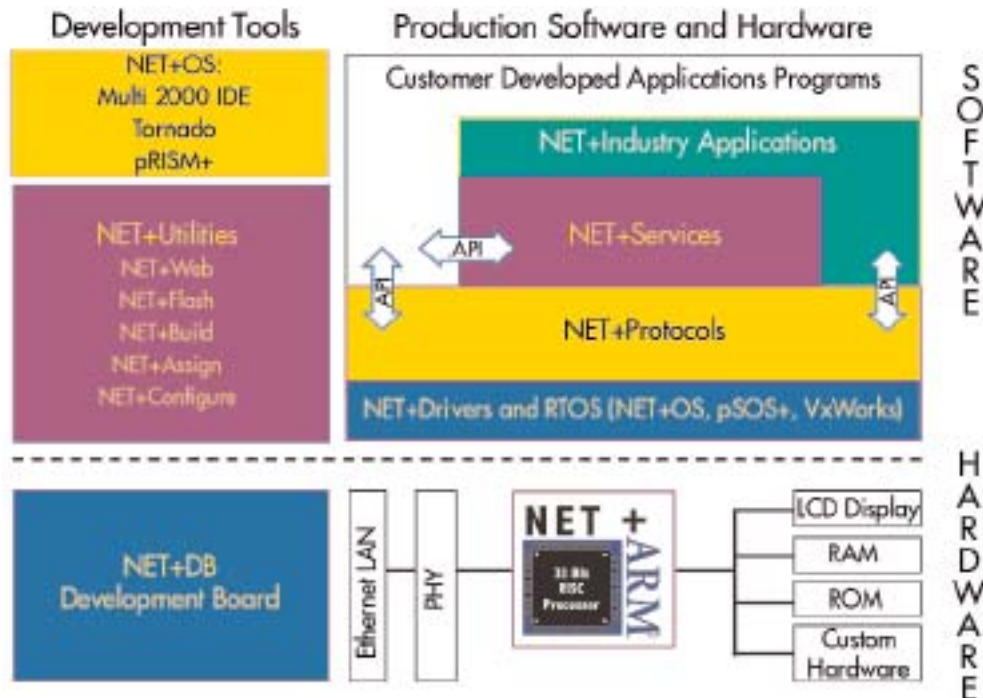
These devices, despite having had their place in the grand scheme, usually do not fit in well with the common goal of faster, smaller, cheaper—and let's not forget—at minimal rework cost and effort. The limited bus bandwidth, horsepower, and in many of cases, lack of an integrated Development Environment (IDE), are causing system architects to take a closer look at the more powerful 32-bit SoC architectures. These 32-bit devices have some understandable advantages versus the 8- or 16-bit devices: they have enough bandwidth to support high-speed serial for Bluetooth, and they are generally built around an industry-standard processing core, such as ARM or PowerPC.

This presents one of many challenges to an OEM design team. While project managers will have an eye on BOM cost, a key decision to be made with 8- and 16-bit hardware solutions is, for example, whether these will be able to support USB long term. Eight and 16-bit microcontrollers may be able to attach USBs, but they will have trouble with the data rates. Why? At data rates as high as 12 megabytes per second, this extra functionality will overtax microcontrollers executing any other tasks. Remember, most 8- or 16-bit microprocessors, no matter at which frequency they run, still have very low-powered instruction sets.

Thirty-Two Paths of Wisdom

Additionally, an engineer building intelligence and networking capabilities into an electronic product must consider that these devices will be required to live on busy networks. Looking back at the example of a Portsmouth cradle, 10/100 Ethernet is standard in most corporate LANs, and some microcontrollers are able to deliver a minimum connection to Ethernet 10BaseT with a basic TCP protocol stack. Portsmouth opted for a 32-bit solution, however, because of what their products need now, and more critically, in the future. And because many Ethernet networks are now 100BaseT, building on a 32-bit platform avoids the inherent hardware limitations when dealing with 8- and 16-bit platforms. On a busy network the more packets the platform has to receive, try to move up the TCP stack, and operate on, the more difficulty the device will have surviving. It can only deal with information 8 or 16 bits at a time, versus the higher horsepower 32.

NET+Works® Architecture



Digi's NET+Works device networking platform features the NET+ARM 32-bit hardware foundation, fully integrated with the necessary networking software including, protocols, services, RTOS and APIs.

Furthermore, with development risks and time-to-market huge factors in the design process, some vendors are promising OEMs hardware and software integration. But engineers must be careful to examine what "integration" really means in most cases. What might an OEM look for when discussing hardware and software integration? Is the TCP/IP stack a full, robust protocol stack from a well-known, established vendor, or is it a silicon vendor's own basic, "homegrown" stack ported to the chip?

Another element is the depth and completeness of a chip vendor's API set. For instance, most vendors can say they have an HTTP server. But is it HTTP 1.0 or is it HTTP 1.1? How many simultaneous connections does it support? How easy is it to add dynamic data? What type of brackets can you embed in it? How complete is the server? Does the hardware platform support HTML? Can you send and receive e-mail messages, and have they been tested against the variety of mail services? Do they support Domain Name System (DNS)? Do they support Dynamic Host Configuration Protocol (DHCP)? Do they support multi-casting? More differentiation between 8-, 16- and 32-bit hardware platforms is demonstrated by multi-casting. Multi-casting occurs when a device can automatically send a message to multiple addresses at the same time. Multi-casting is very important if the end-user requires some kind of peer-to-peer communications, or if peer-to-peer computing is the foundation of publisher-subscriber software.

XML: X-Panded Flexibility and Functionality

When designing with future software modifications in mind, one must take into account newer data formats like Extensible Markup Language (XML), which is able to define data structure, content, and type, and Simple Object Access Protocol (SOAP), which allows devices to advertise their services. Arguably, XML and SOAP will be the basis of communication and distributed intelligence for the next generation of intelligent, networked devices. Thus, it is critical for OEMs to select a hardware platform that not only has the horsepower to transfer data, but also allows design teams to expand their product's functionality quickly and easily.

Importantly, another future need for low-cost devices is to be able to have untrained people install them on a network. As networked devices become more and more popular, it is unreasonable to expect system administrators to install them.

Devices will need to be installed by non-technical employees, for example, maintenance people and even vice presidents! To do this will require technology known as Universal Plug and Play (UPnP). One of the key aspects of UPnP is that it requires a lot of code space, much more than the address space of some micros, but it will be essential to future OEM product sales.

One major benefit of SoC technology is the ability to limit development risks. If a vendor has taken the time to integrate and test the peripherals, more time can be spent on actual application development. Being able to design, test and produce a product takes resources. By eliminating the amount of time one spends redesigning an entire product, a baseline built on an industry-standard SoC can lead to more rapid product release.

A design engineer must also keep in mind that product scalability is as much dependent on software and firmware as it is on hardware. While the SoC technology, whether 8-, 16- or 32-bit based, makes the hardware design, layout and integration much easier, the same technology can lead to some design nightmares during the test and debug phase if one carelessly architects the entire system.

SoC technology, in a majority of cases, is limited only by one's ability to program correctly and efficiently. The key here for designing high-performance, low-cost devices is minimal rework cost and effort. To be competitive in today's demanding market, when selecting an 8-, 16- or 32-bit hardware platform an engineer must choose a processor that allows for software enhancements or modifications with a modern set of software tools. Total cost must be calculated, and more often than not, a penny-wise engineer may find that he is pound-foolish if the processor is not up to the planned task.

Digi International

11001 Bren Road E.
Minnetonka, MN 55343
U.S.A.
PH: 877-912-3444
952-912-3444
FX: 952-912-4952
www.digi.com

Digi International France

31 rue des Poissonniers
92 200 Neuilly sur Seine
PH +33-1-55-61-98-98
FX +33-1-55-61-98-99

Digi International KK

NES Building South 8F
22-14 Sakuragaoka-cho,
Shibuya-ku
Tokyo 150-0031, Japan
PH +81-3-5428-0261
FX +81-3-5428-0262

Digi International (HK) Limited

Suite 1703-05, 17/F.,
K Wah Centre
191 Java Road
North Point, Hong Kong
PH: +852-2833-1008
FX: +852-2572-9989
www.digi.cn

www.digi.com

**email:
info@digi.com**

**91001377
B1/306**



© 2002-2006 Digi International Inc.

Digi, Digi International, the Digi logo, NetSilicon and NET+Works are trademarks or registered trademarks in the United States and other countries worldwide. ARM and NET+ARM are trademarks or registered trademarks of ARM Limited. All other trademarks are the property of their respective owners.