WHITE PAPER

A Software-First Mindset for Driving Efficiency and Sustainability for Industrial IoT

arm | Capgemini | Schneider Electric | Witekio

Contents

“Schneider Electric is committed to developing next-generation, open edge infrastructure and ecosystems with key partners like Arm, Capgemini and Witekio, and embracing a wider ecosystem of partners who are also ready to be part of this journey. We want to drive maximum sustainability and efficiencies benefitting both the environment and customers.”
Fabrice Besancenot, VP Intelligent Edge, Schneider Electric

Introduction

A collaborative design effort involving Schneider Electric, Arm, and system integrators Witekio and Capgemini, has produced a software-defined platform for industrial automation and energy management.

Working within the Arm/Linux ecosystem, the team used cloud-native techniques, borrowed from the datacenter, to create a flexible, energy-efficient reference design that uses virtualization to enable real-time, mixed-criticality services at the embedded edge.

By flipping the traditional design process, and starting with software before committing to hardware, the team validated that such an approach worked more efficiently and addressed the challenges to combine real-time workloads alongside compute-intensive operations, including machine learning, in an embedded system for the industrial edge.

Over time, this new approach, which prioritizes the portability of software, can help Schneider Electric develop new business models. Schneider can then mix and match software elements to suit changing hardware environments and address different use cases across verticals, which drives efficiency and sustainability, and optimizes service delivery to its customers.

Moving From the Cloud to the Edge

Schneider Electric, a leader in the digital transformation of energy management and industrial automation, recognizes the value of edge computing. It is driving the development of smart and sustainable next-generation infrastructure and ecosystems by moving computation and data storage away from the cloud and central servers, decoupling hardware and software, and moving control away from dedicated hardware models to enable more efficient real-time data operations at the edge. This real-time data analysis and decision making at the point of data generation ultimately improves efficient resource allocation and utilization in various systems, such as energy grids and industrial processes. The consolidation of these workloads at the edge saves further power through reduced data transfer and bandwidth usage, and leverages the optimized energy efficiencies built into Arm embedded-edge compute systems.

Also, as edge-based computing has grown more powerful in terms of computing and memory capacity, embedded systems operating at the edge are doing much more than before. The integration of powerful computing capabilities, machine learning, artificial intelligence and advanced analytics lets edge devices work more independently, so they can do things like generating real-time insights, supporting predictive maintenance, and performing autonomous decision making, without needing to interact with the cloud.

What used to require a dedicated server in a datacenter or installed nearby, can now happen on a single multicore processor. Today's lightweight, processor-driven systems can collect data, process it, and analyze it in real time, for faster decision making, even in complex industrial environments. These lightweight systems consume less power than their server-based counterparts, and are easier to configure and install, so they also can scale more easily to support deployments that require hundreds, if not thousands, of devices.

Driven by a single multiprocessor core, these systems can also be equipped with robust security mechanisms, such as device and user authentication, data encryption, and over-the-air security updates, as part of end-to-end security for the entire installation. What's more, because more data remains local without being transmitted to the cloud, edge-based systems can reduce the risk of sensitive data being compromised.

Two Pain Points: Supply Chain and Design Reuse

Schneider Electric has been working at the edge and delivering embedded systems for use in Industrial IoT (IIOT) and energy management landscape for some time now. In recent years, however, ongoing supply-chain issues have hampered development across the market and Schneider Electric has faced the challenge of getting the ICs they needed when they needed them. They would start their development effort with a carefully selected multicore processor and associated platform that was readily available, but by the time they had finished creating the software to run on their chosen hardware platform, chip shortages and other logistical challenges could force them to switch platforms. Changing what they already had in place for the original, now hard-to-obtain multicore processor so that the design could be ported to a new one that was more readily available, involved a painful—and expensive—redevelopment effort.

At the same time as they started having more completed IIoT projects, they noticed that many of the solutions they'd developed targeting different use cases and different vertical markets, performed similar functions while running on different hardware platforms. An example of this is a safety mechanism that prevented over-temperature conditions which may be duplicated across a number of use cases. However, repurposing the company's already completed software function to run on a new processor tended to be time consuming and expensive.

Following the Datacenter Model

These two situations—IC shortages and shared functionality—led Schneider Electric staff to thinking about ways to make what they developed more portable and less tied to a particular hardware implementation. If they could end their dependency on specific ICs, they would be less vulnerable to supply-chain issues, and if they could apply existing functions to new use cases, they would save time and effort on new projects.

The team decided to try a new approach. Instead of using hardware in the form of a particular multicore processor as the starting point for design and then developing software to run on that specific IC, they would do the reverse. They would start by developing the software and then modify it to run on a specific instance of hardware. This approach is common in hyperscale datacenters, where virtual representations of physical machines are used to run different applications for different tenants. With virtualization, the datacenter's physical pieces of equipment—their computing, storage, and networking resources—are represented by virtual machines (VMs) that mimic the functions of the physical equipment. Each VM behaves like an actual computer system—running its own operating system and supporting software, including libraries and applications—and is fully isolated from the other VMs sharing the same physical resources.

A separate software component, called a hypervisor, runs on the processor and takes control of the physical hardware resources, allocating them to each VM. These VMs allow multiple virtualized devices and their applications to run on a single physical device thus reducing the energy and resources needed to support a workload or service. The hypervisor enables dynamic allocation of compute resources on demand, reducing the overall hardware footprint required at the edge. This hypervisor is optimized to assign resources very efficiently, so each VM has enough to run as designed but doesn't slow down other VMs running at the same time.

This makes it possible to divide a single physical resource for use by multiple tenants. One tenant, for example, might use the datacenter to run an online booking service for hotel rooms, building on a website and mobile-app infrastructure to manage reservations and accept payments, while another tenant might run a content-streaming service that supports millions of concurrent users across various devices and uses AI to deliver personalized recommendations.

From the datacenter's point of view, virtualization and containerization save money and energy, since fewer resources are needed to support the same number of tenants and their applications. And from the tenant's point of view, virtualization makes it easier to deploy and operate an online business, because developers can focus on optimizing the workloads they need to run their operations, without spending time configuring for different pieces of hardware.

"The Arm-based software-defined edge allows Schneider Electric to meet growing customer demands, while minimizing its environmental impact at the same time."

Testing the Theory

Schneider Electric is targeting the idea of using virtualization and other cloud-native techniques to minimize its dependency on hardware and make its software more portable, but there were a lot of unknowns.

Would the software-defined approach even work for its embedded edge applications? Would a low-power, resource-constrained embedded system, built using a single multicore processor, be able to support more than one VM? Could a virtualized system deliver the reliability and real-time responsiveness required by so many industrial operations?

What about multitasking and mixed criticality? Could a virtualized system support multiple functions with different requirements for safety, reliability, and security? And what about compute-intensive operations? Would ML/AI algorithms, which require large amounts of processing power and memory for proper execution, cripple the rest of the system while they were running?

The logical next step was to explore the possibilities and put virtualization to the test in a simulated industrial/energy-management edge environment on an Arm platform.

Testing Within the Arm/Linux Universe

To create their test case, Schneider Electric staff chose to work in the Arm/Linux ecosystem. They have been using Arm platforms for many years and recognize the power efficiencies as well as the ability to scale from 32-bit architectures, for smaller, purpose-built systems, to 64-bit architectures when they needed extra processing power for a more complex embedded system. Arm's expanding ecosystem of third-party partners was also a benefit, since Schneider Electric has had good experience in finding the help it needs when developing specialized functions within the Arm/Linux ecosystem.

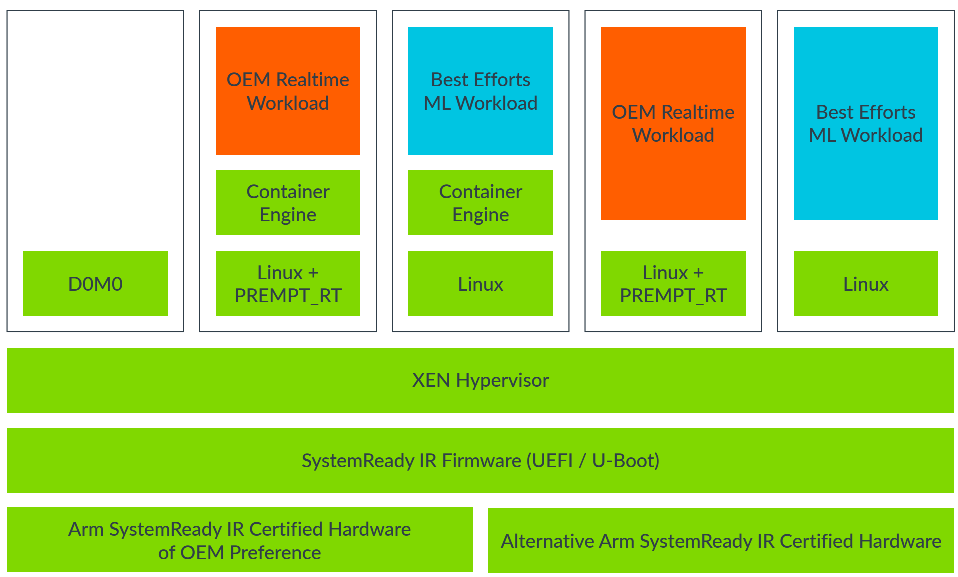
There were other factors that made the Arm/Linux ecosystem a particularly good choice for Schneider's experiment in the software-defined edge. To begin with, Arm architectures are a prominent feature in datacenters, where virtualization is a proven foundation for advanced computing. All the major hyperscale cloud providers, including Amazon, Google, and Microsoft, are either using Arm architectures at scale or have deployed their own in-house Arm-based CPUs and offer a wide selection of Linux-based tools forcloud-native development.

As a company, Arm is a heavy proponent of standards-based cloud-native development and has already tailored cloud-native techniques for use in various IIoT applications. Also, the combination of compute performance, energy efficiency, and low-cost operation has made Arm a leading choice for AI inference-optimized processors, with a growing presence in embedded applications that use ML/AI algorithms. Workload optimization is another area where the Arm architecture has a proven track record, given its widespread adoption in networking, security, and storage applications.

"Arm CPUs and GPUs have doubled their AI edge processing capabilities every two years during the past decade."

The Software-Defined Industrial System (SDIS)

As a first step in creating the test case, Schneider Electric established a collaborative partnership. The team included Arm for guidance and expertise on development within the Arm ecosystem and Linux, the system integrator Witekio, for knowledge of embedded software services and IoT operation, and the system integrator Capgemini, for experience with smart edge and IoT services, including ML/AIgorithms. Together, the team created a block diagram that defined the system and showed what they were aiming to create. They called it the software-defined industrial system (SDIS).



A Layered Approach to Operation

A review of the SDIS block diagram shows the basic structure of one system used by the Arm ecosystem to enable virtualization and leveraging the operational versatility enabled by the hypervisor and VMs.

A Flexible Hardware and Firmware Foundation

The SDIS block diagram doesn't identify any vendor-specific multicore processor by name. Instead, it specifies an Arm SystemReady IoT Ready (IR) certified hardware, either of the OEM's choosing or an alternative. This supports Schneider Electric's goal to end its dependence on a single hardware option.

"The future smart industrial edge devices will co-host critical control functions and AI workloads within the resource and energy constraints of the device. This will empower industrial deployments to be flexible and cost-effective. Our collaboration with Arm, Schneider Electric and Witekio is a step towards achieving this goal of building solutions for sustainable, intelligent, and connected devices for industries,"

Shamik Mishra, Vice President and CTO Connectivity at Capgemini

Arm SystemReady IR

Arm SystemReady IR is a subset of the Arm SystemReady program, a compliance certification program that verifies that a given platform implements a minimum set of hardware and firmware features that an operating system can depend on to deploy the OS image. Before SystemReady, porting an application from one hardware platform to another, for instance to improve power consumption or add programmable logic, could mean weeks of re-engineering and fiddling with firmware. With SystemReady IR, the development team creates the embedded app just once, and then can easily and quickly port it to a different SystemReady IR-certified hardware platform. In other words, working with only SystemReady IR-certified multiprocessors means developers can be confident that the subsequent layers of software will work, even if the underlying hardware is replaced with something different.

Thus, specifying SystemReady IR hardware for the SDIS project meant that Schneider Electric could avoid supply-chain issues by deploying workloads across different Arm SystemReady IR-certified platforms. In this case, Schneider Electric used a Xilinx ZCU102, a 64-bit Arm multiprocessor Arm Cortex-A53, Arm Cortex-R5F, and Arm Mali-400 GPU cores and a Raspberry Pi4, which uses a Broadcom BCM2711 multicore processor with Arm Cortex-A72 cores.

The Arm software enablement, delivered through SystemReady IR, allows the team to deploy the same software across two different targets.

"We have over 21 years of experience in embedded and IoT software services, and have seamlessly integrated embedded software solutions, ensuring rapid and reliable implementation across diverse hardware platforms and industries. Our collaboration with Schneider Electric and Arm on the SystemReady project is a great example of our commitment to accelerating development cycles and enhancing platform flexibility, enabling businesses to adapt swiftly to changing market demands."

Nicolas Duvernay, VP Sales & Marketing, Witekio

Hypervisor for Virtualization

The next layer up in the block diagram, after the SystemReady IR hardware and firmware, is a XEN hypervisor, which acts as the intermediary between the underlying physical hardware and the VMs. XEN is a bare-metal hypervisor, which means it sits directly on top of the hardware and firmware, managing and allocating resources for the VMs without going through a host operating system. The XEN hypervisor has its own device drivers and interacts with each component directly for any I/O processing or OS-specific tasks. Using a bare-metal hypervisor enables a secure, stable solution that requires very little overhead.

The XEN hypervisor works with Domain Zero ("Dom0"), which manages the interaction with the other layers above the hypervisor and is the first domain started by XEN on boot. Dom0 runs the XEN management stack and has special privileges, including being able to access the hardware directly. With Dom0 helping to offload tasks, XEN is a thin layer that works well in resource-constrained embedded environments.

Two Carefully Chosen Workloads

The top layer of the block diagram, where the VMs reside, is in many ways, the real focus of the SDIS experiment, because that's where real-time, mixed-criticality performance is put to the test. The team defined two workloads that might typically be used in an industrial setting and configured them so they would push the system to perform.

The team wasn't trying to create a real-world application or satisfy the requirements of a particular use case. Instead, they were trying to create a hypothetical system that would test whether virtualization in a resource-constrained embedded environment could work at levels that were fast enough and efficient enough to meet requirements for industrial automation. The initial configuration of the SDIS was an attempt to prove the concept and determine whether further development would be worthwhile. The team specified one real-time workload and one best-effort workload in a separate VM. The goal was to run both workloads individually and simultaneously to gauge performance. To test and evaluate the effects of different approaches to virtualization, the SDIS was configured to run each workload two ways, with a container engine and without. Here's a closer look at the two workloads and the challenges they presented to the SDIS:

– OEM Real-Time Workload: Virtual Device for Intelligent Electronic Control

Designed to control a circuit breaker in an electrical substation, this workload gathers and processes data from a circuit breaker so it can react if there's a problem. The workload is not an actual working device, but a tool that Schneider Electric developed in house to test how a real workload would behave. The workload runs independently of external services and provides a good way to measure task latency, test response times, and monitor data traffic without having to build a fully functional controller. On the SDIS, the real-time workload is supported by PREEMPT-RT, a Linux patch that provides real-time mechanisms that help reduce latency.

To be considered a success, the workload's maximum task wake-up jitter should be under 50 microseconds, while a maximum of under 100 microseconds is still acceptable for some applications. This makes the reference deployment of this workload a good starting place when evaluating the SDIS's ability to deliver real-time performance, while running independently or in combination with other workloads.

– Best-Effort Workload: Machine Vision (AI-Driven Video Monitoring)

Supplied by Capgemini, this is a workload that uses an ML inference engine to take and process images. It might, for example, be used to replace hardware-based sensors typically used to monitor the temperature of a system. With this machine-vision workload, instead of mounting a series of temperature sensors on each piece of equipment—a resource-intensive project that requires the design, installation, and maintenance of extra hardware devices—a single video camera can be used to monitor equipment for signs of overtemperature conditions.

Deploying machine vision at the edge is a demanding task on its own, given the constraints on compute, storage, and power budgets in most embedded systems. Running machine vision in a mixed-criticality environment, alongside a real-time workload, meant optimizing the SDIS to gauge effects on the workload from noisy neighbors. Some of the techniques that isolate workloads from each other, including virtualization and the use of containers, can place additional stress on already limited resources. The goal was to see what techniques were helpful in optimizing performance, and how well the SDIS could execute concurrent workload execution.

Once Capgemini had the first configuration for the machine-vision workload in place, staff began experimenting with how much the Arm architecture could handle. They started with one camera, then increased it to two.

In the end, they were able to support four cameras and still maintain their target level of performance needed for this specific application. Capgemini credits the Arm NN (neural network) inference engine and its supporting accelerator libraries for Arm/NEON-based CPU acceleration, with the ability to optimize performance and maximize efficiency with Arm SystemReady IR hardware.

Encouraging Initial Results

At the time of this writing, the team continues to fine tune the system, but the initial results are already extremely promising. Even though the system is designed to be generic with the ability to perform many different functions, the chosen metrics indicate that the approach—using virtualization to support real-time, mixed-criticality operation at the embedded edge—is a viable approach to apply to different scenarios.

In particular, the team has been impressed by how dynamic, flexible, and sustainable the SDIS is proving to be.

1. Dynamic

The real-time and best-effort workloads work effectively in any combination. The relationship between the two is not static, but still constrained, that is a one real-time workload plus zero to four best-effort workloads, and they can be changed on the fly to accommodate different operating requirements. For example, one or more cameras can be turned on and off, and the real-time workload can be active or inactive. It doesn't matter — the SDIS still meets specified performance targets.

2. Flexible

Using Arm SystemReady IR-certified hardware reduces complexity when mixing and matching hardware and software, which means the baseline functionality of mixed criticality can be quickly ported to other applications and other industries. Outside of industrial automation, the real-time control of machine vision has widespread application in a number of areas, including robotics, telecommunication, healthcare, and consumer electronics.

3. Sustainable

The SDIS is a low-power solution, so it offers opportunities to save energy and lower operating costs. Working at the embedded edge away from the cloud, there's less need to expend energy by transmitting data or involve the energy-intensive servers that typically reside in a datacenter. Also, the SDIS uses processing capabilities very efficiently, so the system can cost-effectively scale across a facility. More cameras can be deployed on the shop floor, for example, while keeping the energy consumption and operating overhead low.

Valuable Real-world Potential

As a proof of concept, the SDIS is helping Schneider Electric and its partners evaluate the potential of using cloud-native principles when developing for the embedded edge. As part of real-world development, the SDIS promises to deliver several important benefits, especially in terms of system performance and development time:

1. Reduced latency in critical applications

On its own, virtualization tends to add latency, because the extra layers of software can slow things down. The SDIS includes several optimizations that reduce the impact of virtualization and help reduce latency. Techniques which limit involvement of the Linux kernel help reduce execution time and make the system more responsive.

2. Enhanced cyber security

Since the hypervisor can run different workloads in isolation, virtualization makes it easier to protect data. The data associated with each workload remains separate from other workloads, making it harder to gain unauthorized access to workload-specific data or modify the results.

3. Reduced reliance on cloud resources

The high-performance, SystemReady IR-certified multiprocessors that form the basis of the SDIS are more than capable of running the kinds of compute-intensive applications, including AI/ML functions, that up to now have only been able to run in the cloud. Reduced reliance on cloud resources saves on operating costs and enables increased innovation at the embedded edge.

4. Real-time decision making for mission-critical applications

In applications that require extremely fast reaction times, the SDIS has the processing power needed to support real-time operation. The PREEMPT-RT option in Linux reduces latency further by making the scheduler better able to react to real-time events. To improve core density the Linux scheduler must be challenged to run multiple containers on a single CPU, each container hosting a real-time critical application with multiple tasks with varying periodicity and priority.

5. Decreased time-to-market

By using SystemReady IR-certified hardware, the SDIS simplifies the design of complex applications because development teams can focus on the details of software operation without having to worry about the underlying hardware. New systems that run application versions on different hardware platforms to achieve better power ratings or to add extra functionality can be brought to market much faster than before.

6. Easier deployment and maintenance

The SDIS shows that it's possible to create small, lightweight systems that can replace the IPCs, servers and other complex hardware systems currently used to automate industrial processes. Running virtualized workloads makes it simpler to tailor operations for a particular use case and simplifies deployment because functionality is easier to install and scale. What's more, existing workloads can be updated and new workloads can be introduced without having to redesign already deployed hardware. Long-term maintenance becomes easier and systems can stay in the field longer without needing hardware upgrades, even if standards and regulatory requirements continue to evolve.

"The SDIS simplifies development while delivering the right levels of performance."

The First Step Toward a New Era

The SDIS represents a new approach and a significant evolution for an industry that is seeking ever faster evolutions. Automation in the industrial and energy management sectors is often a highly complex undertaking, involving intricate process management and operation on a very large scale. Many installations remain in place for years at a time, if not a decade or more. Any problems associated with the installation upon deployment or down the road have the potential to disrupt essential business processes and even put lives at stake.

Schneider Electric is driving the cutting edge in the exploration of cloud-native technologies to deliver the next-generation portfolio of services and sustainability. It's important to note that this is just one proof of concept within that journey, demonstrating that this vision is achievable on an Arm platform.

Over the longer term, however, the SDIS is a first step toward what Schneider Electric hopes will be an enhanced way of doing business. Enabling its development teams to deliver further high-quality solutions and services tailored to meet customers' evolving needs and unlocking new business opportunities.

Using cloud-native technologies to build a portfolio of portable, flexible software offerings will make Schneider Electric a better partner, too, because its modular solutions will make it easier for customers to scale up or down quickly, as demand dictates.

Ultimately, the cloud-native approach can become a foundation for new business models, where Schneider Electric offers software that can run on many different implementations of hardware. Industrial automation services, deployed as software-defined resources, can be used to service many different types of customers, whether they're operating a factory, running a hospital, or managing a supply chain. Schneider Electric's development activities become device/functionality-centric solutions and more platform-oriented, with the ability to apply expertise gained from completed projects to new challenges in different areas. Each round of custom development yields an IP that can be re-used in the form of a virtual workload that can be applied wherever Schneider Electric's services are needed.

"Thinking software first frees developers to be more effective —and more creative."

Conclusion

As more and more IoT devices are connected, the industrial edge plays an increasingly crucial role in supporting diverse systems with a range of power and performance requirements. A key element in the transition to the industrial edge is the move away from traditional approaches to product development, where individual solutions are developed for each given use case and developers begin by choosing a hardware platform. The adoption of cloud-native techniques that let developers focus on value-add software, instead of the details of a given hardware implementation, frees development teams to be more creative and more responsive. To deliver ease of digital transformation, overcome OT/IT integration challenges, and meet requirements for system updates and testing, the industrial edge must embrace cloud-native software principles.

The collaboratively developed SDIS is a proof of concept that builds on the processing performance, energy efficiency, and standards-based operations that are the hallmarks of Arm platforms. The SDIS successfully demonstrates that using virtualization is a viable option for creating modular, reusable software components that enable real-time, mixed-criticality systems at the embedded edge.

"Cloud native has proved itself in the IT world. It's time to bring it to industrial."

How To Get Started With This Example

Use the example referenced in this white paper for your own software build. Visit gitlab.arm.com/bluegreen for setup and to build software for any Arm SystemReady platform.

To learn more about the Arm ecosystem and technologies for software-defined systems in industrial, visit arm.com/markets/industrial.