WHITE PAPER

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

abaco SYSTEMS

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

Introduction

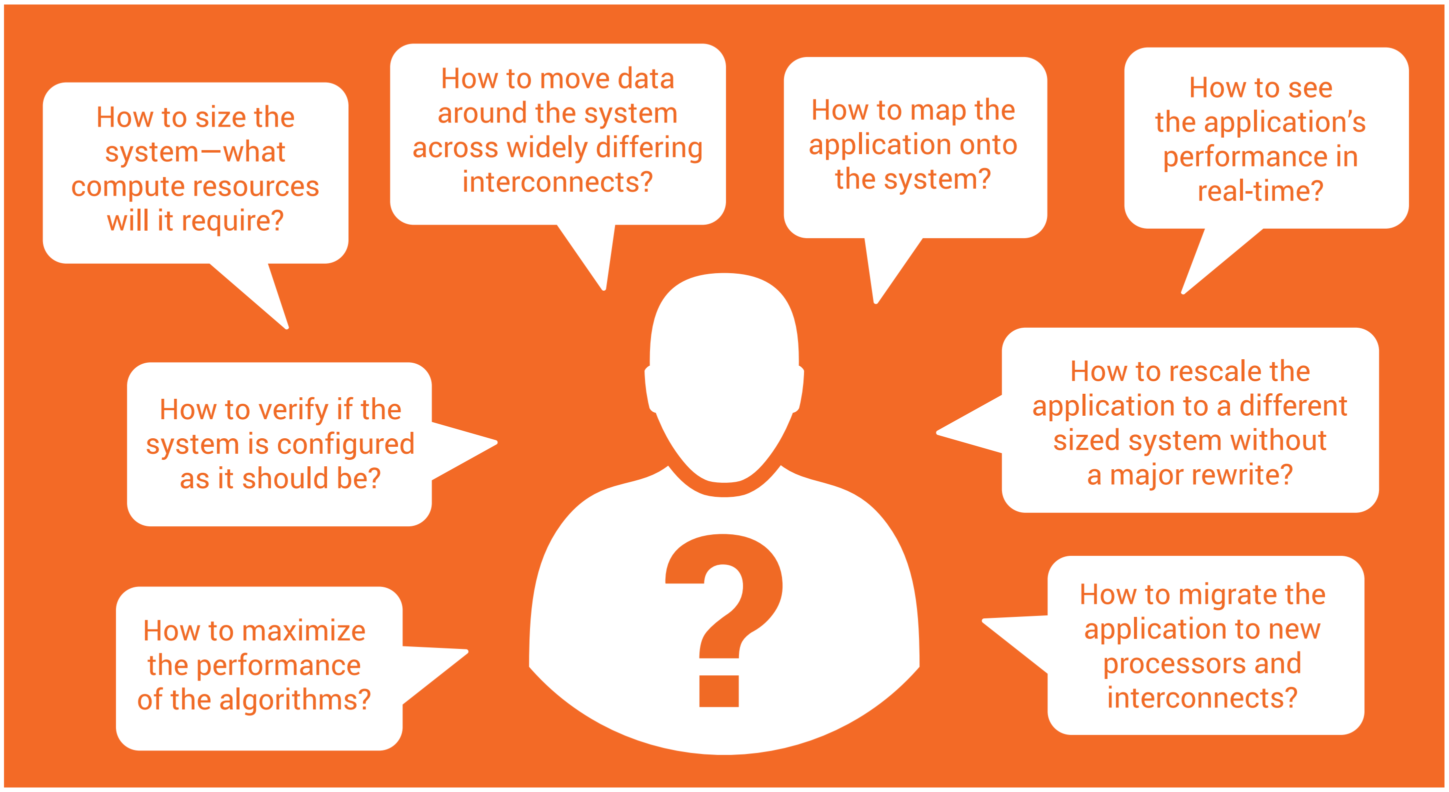
Complex multiprocessor systems require a broad array of software tools to speed system application software development.

Developing software for complex, multiprocessor embedded systems such as found in military, aerospace, and industrial applications is often a challenge due to the multitasking and multithreading employed to get the maximum performance from the CPUs. Those CPUs often integrate multiple processor cores, and the systems may even contain CPUs with different architectures, thus increasing the complexity of application programming. Generic software tools and algorithm libraries to program these multiprocessor embedded systems are available from multiple suppliers such as Wind River.

Additionally, system hardware vendors such as Abaco (formerly the embedded systems business of GE Intelligent Platforms) and others have developed their own tool suites and algorithm libraries that are optimized for their processor boards and other generic and application-speci ic boards used in the system.

Getting Down to Business

To start the development cycle, there are several questions that designers or design teams should ask to get the project underway. Some of the points that should be addressed include:


How to size the system—what compute resources will it require?
How to move data around the system across widely differing interconnects?
How to map the application onto the system?
How to see the application's performance in real-time?
How to verify if the system is configured as it should be?
How to rescale the application to a different sized system without a major rewrite?
How to maximize the performance of the algorithms?
How to migrate the application to new processors and interconnects?
?

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

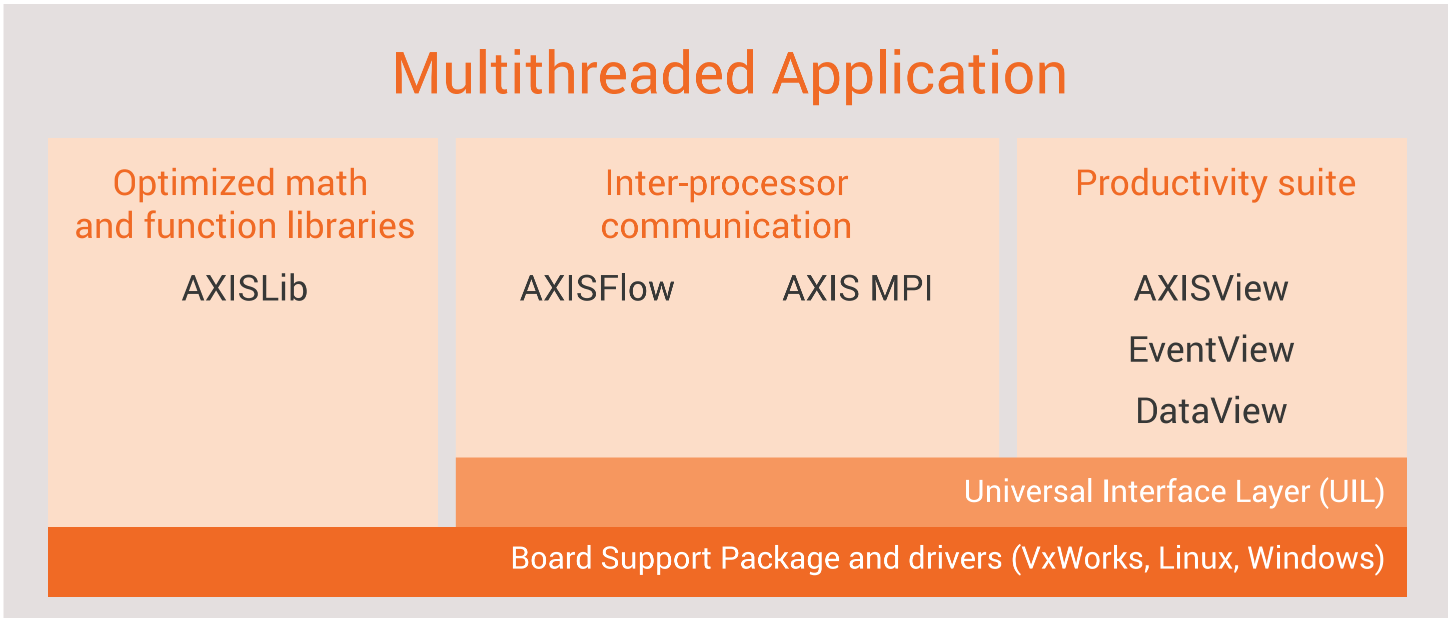

Multithreaded Application
Optimized math and function libraries
AXISLib
Inter-processor communication
AXISFlow
AXIS MPI
Productivity suite
AXISView
EventView
DataView
Universal Interface Layer (UIL)
Board Support Package and drivers (VxWorks, Linux, Windows)

Figure 1: The AXIS tool suite developed by Abaco provides a complete software and hardware development environment for complex multiprocessor, multithreaded applications. It contains an optimized math and function library (AXISLib), tools to support interprocessor communications (AXIS Flow), performance evaluation tools (Event View and Data View), and a suite of productivity tools to speed software development (AXIS View).

Addressing all these questions requires a modern software tool-chain with an easy-to-use and flexible graphical user interface (GUI) and a rich algorithm library to tackle the signal processing, control, and data handling applications. The combination of these tools and libraries will result in significant reduction in the time needed to develop the system software versus the use of generic software development tools.

Just such a suite of tools was developed by Abaco—the AXIS Advanced Multiprocessor Integrated Software environment, which consists of five main integrated software elements (Figure 1):

The suite consists of an integrated set of software development tools that are designed to simplify the process of creating DSP and other applications based on multiprocessor platforms.

> AXISFlow or AXIS MPI, which handle interprocessor communications

> AXISView, which provides multiprocessor productivity tools, and

> AXIS EventView, which provides an event analysis tool.

> AXIS DataView, which provides a tool for GUI development.

> AXISLib, which contains optimized high-performance digital signal processing libraries.

Without such a tool, it can take a great deal of time to log into each node, extract the configuration and discovery data, and to correlate the data across the system. One solution to the software challenge is the AXIS suite of software development tools from Abaco. The suite consists of an integrated set of software development tools that are designed to simplify the process of creating DSP and other applications based on multiprocessor platforms.

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

AXISFlow and AXIS MPI middleware libraries focus on interprocessor communications, while AXISView offers multiprocessor productivity tools, and AXISLib contains a library of optimized digital signal processing algorithms. Although each component can be used independently, the tools are designed to work together as a single environment. Together, they can cut the time needed to map the application to the system, run it, test performance, and then remap until the desired result is achieved.

How to Size the System

When the time comes to develop the software, designers should have a pretty good idea about what the core of the application software needs to do. There may even be a C-code prototype of the application software. Using the prototype code or the application description, designers can figure out what compute resources will be needed to run the application in real-time. One approach to do that is to develop a tool that can model the compute performance of different processors at different clock speeds. By then feeding the code snippets into this tool, a reasonable estimate can be made for how the algorithm will perform on a CPU such as x86, a PowerPC, or a field-programmable gate array (FPGA), for instance. In the near future, the Abaco tools will also be able to handle development for ARM-based systems.

Such an exercise would let the designer decide which processor types best match which algorithms, and how many processors and/or FPGA resources will be needed to perform the required task in the timeframe allowed, which is typically dictated by the incoming data rate.

The more sophisticated the tool, the better the analysis. The tool should model the arithmetic-logic unit (ALU), the memory subsystem, multiple layers of caches and the I/O channels. Outputs from the tool can be defined as “costs,” which can then be manipulated—compute resources, power, etc. Such a utility has begun to find application in the real world, helping to size systems to create more accurate proposals.

How to Check if the System is Configured as It Should Be

Once a development system has been cobbled together, all the boards plugged in, and all the interconnect cables have been wired up, it is time to see if all the jumpers are set correctly, boards are in the right slots, interconnect cables are installed correctly, etc. This can prove to be a laborious task without some form of automation. To ease the task, a software tool suite that can probe all parts of the system and display the resultant configuration information graphically can be a boon.

The screenshot in Figure 2 shows a typical scenario of a system employing DSP engines and PowerPC compute engines. Generated by the HardwareView component of the AXISView software, the diagram shows multiple DSP cards and CPU cards being probed. The boards are linked by VME and StarFabric buses. By selecting functional blocks in the graphical representation, greater detail can be shown in the test pane to the left. For example, clicking on the baseboard will show board revision, revisions of any programmable devices, board serial number, etc. The ability to log this data to a file is a great way to document a system setup for configuration control.

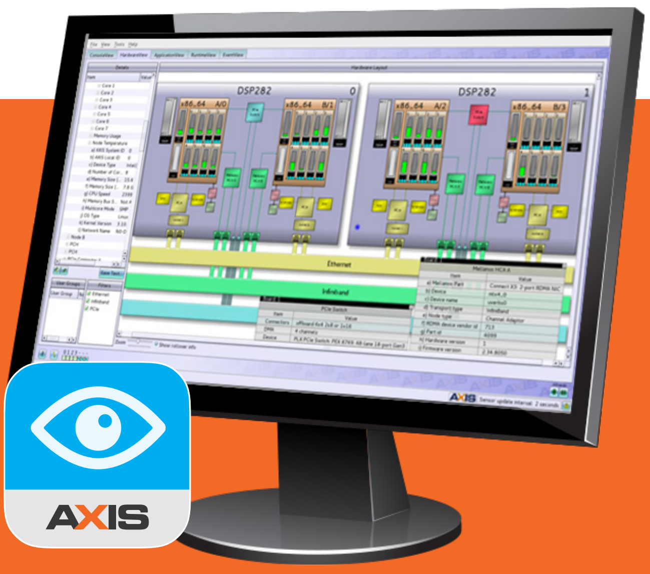


Figure 2: A typical system design modeled with AXISView can be probed using the HardwareView component of the software, allowing the designer to view all aspects of the design.

Each data path in the system can have a different mechanism and programming interface. This means the programmer must understand the low-level hardware details. In addition, rescaling and remapping of the application to the hardware may involve extensive recoding. A robust Inter-Processor Communications (IPC) library should hide all this detail, but not at the expense of performance. AXISFlow and AXIS MPI are such libraries. They present a simple data movement paradigm at the application level, but underneath efficiently select and use the most efficient transport with a minimum of abstraction inefficiency.

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

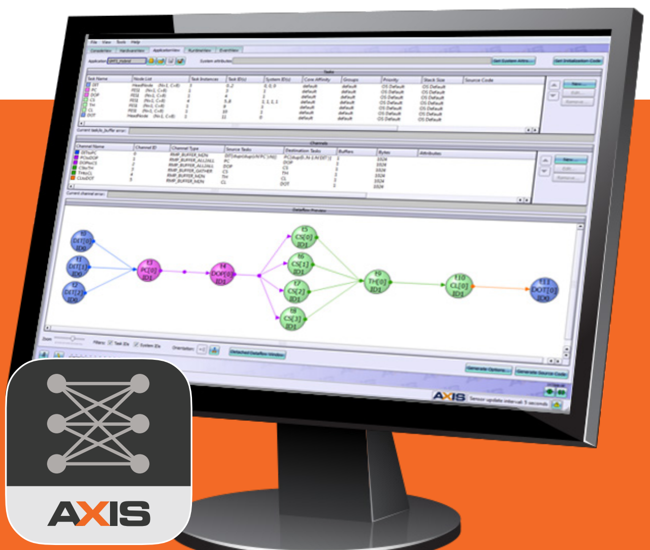


Figure 3: In this screenshot of the ApplicationView tool, system tasks are represented as circular objects and the interconnecting lines show the data flows between the tasks. The tool allows a single click to generate the source-code modules to configure the AXISFlow communications library elements, saving hours of coding.

The AXISFlow interprocessor communication software provides high throughput, low latency, reconfigurable interconnects that facilitate data transport between tasks, processes, processor cores, boards and systems. Processing elements can be integrated for seamless scalability to meet the requirements of the most demanding of applications. Recently added features enable control over task affinity in SMP architectures and partitioning of applications over multiple processes.

A user API, which is independent of processor, operating system and fabric, insures that the tool has a lot of flexibility, thus providing designers with the ability to handle a wide array of applications, both now and in the future. The AXISFlow tool can operate standalone or as an integral element within the AXIS software environment. Its integrated modular architecture has the flexibility to allow the engineer to select the specific functionality required, with the ability to reconfigure or scale the system to meet future application demands.

How to Map an Application onto the Processing Resources Available

The high-level system design will typically partition the application into discrete tasks. Now comes the chore of allocating the tasks to physical compute resources. Traditionally with a real-time operating system, this can require much messing around with console windows, scripts and typing. Worse, the placement of these tasks can change the interconnect between nodes, and thus require code changes (unless the tool suite allows for positionless communication strategies). Far better would be a tool that allows the user to place tasks based on processor type, board type, or automatically.

In the AXIS ApplicationView screenshot shown in Figure 3, tasks are illustrated as circular objects. The windows above the graphic are used to set the number of instantiations of each task and their allocation to compute resources. The interconnecting lines show data flow paths between the tasks. These paths include simple point-to-point transfers and some more complex data manipulations such as scatter, gather, and all-to-all. Once a configuration has been set, a single click generates the source code modules to configure the AXISFlow communications library and automatically generate the code to initialize the system and instantiate the required tasks or processes automatically, saving hours of coding.

To allocate the resources a graphical tool such as ApplicationView in combination with the AXISFlow libraries can handle task replication by formula. For example, based on the number of processors per board and the number of boards in the system, a configuration can be generated with N*C instantiations of a certain task, where N is the number the CPUs in the system and C is the number of cores per CPU; the system could then be made to rescale based on a revised core count.

For instance, with two dual-processor Intel i7 boards in the system, N=4 and C=4 so we generate 16 instances of the task. If we later find that this configuration does not meet our performance requirements, we can add another board; now N=6, and we have 48 instantiations. If the tools include the self-discovery feature, this rescaling can be fully automatic. Tied to a positionless communications scheme, this can lead to a highly flexible, scalable configuration tool.

Processing elements can be integrated for seamless scalability to meet the requirements of the most demanding of applications.

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

The tool also provides the ability for the user to control the allocation of tasks to specific CPU cores (core affinity) and also partitioning of application sub-sets in separate processes if desired.

Evaluating Application Behavior and Performance

One of the toughest problems facing real-time embedded application developers comes after the successful creation of a functional application. At this point, the developer usually finds performance is below expectations and significant tuning and optimization is required.

The challenge is to identify the areas in which to focus the effort. There may be sections of code that can be made an order of magnitude faster by some focused optimization, but if that piece of code only represents 2% of overall application execution time, then it is probably not worth the effort.

The developer needs to identify the 'long poles in the tent.' These may be processing algorithms that are inefficient or bottlenecks in the data movement between threads and processors. A tool is needed to identify these performance issues. Traditionally, at this point, two types of tools can be used.

Profiling tools

Profilers are a traditional tool used to determine the efficiency of an application. Offering a useful overview of the application's effective utilization of the underlying processing resources, profilers can show where the majority of time is being taken in the code.

There are a number of commercial profilers available that do a good job. Allinea MAP is one worthy of special mention as this also has hooks into MPI, an open standard communication middleware layer, to profile of inter-process communications.

However, profilers generally do not tell the whole story, as they tend to report averages. If a bottleneck resides in a specific thread, it can be difficult to use a profiler to identify this explicitly.

AXIS EventView's interface enables the developer to visualize time-aligned event traces across many application threads.

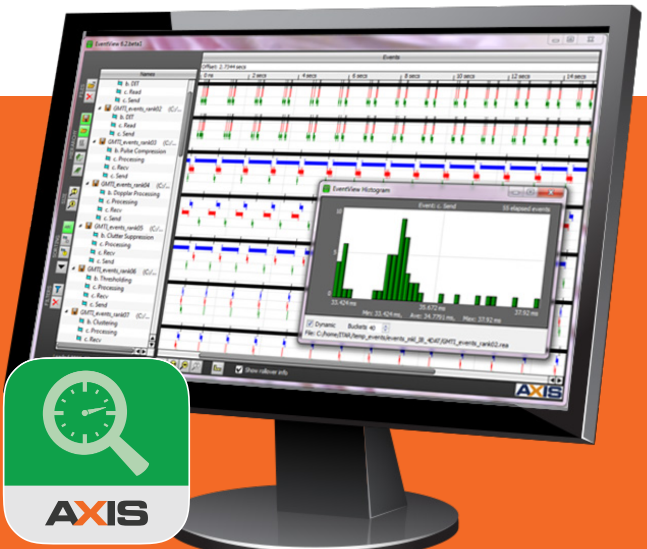


Figure 4: Able to analyze real-time embedded applications that consist of multiple threads distributed over multiple CPU cores, the EventView tool enables the developer to visualize time-aligned event traces across many application threads.

Event analyzers

This is where an event analyzer can be invaluable. Event analyzers present timing measurements with more granularity and give a timeline of events in the application. This can enable the developer to home in on problem areas much more quickly than the results from a profiler will facilitate.

The AXIS EventView tool developed by Abaco is an example of an event analyzer focused on analyzing real-time embedded applications that consist of multiple threads distributed over multiple CPU cores. AXIS EventView's interface enables the developer to visualize time-aligned event traces across many application threads. Additional statistical information, including histograms, can be displayed to immediately indicate application jitter and therefore determinism.

Profilers and event analyzers can be viewed as complementary performance analysis tools. In general, however, event analyzers often provide more valuable insight. Some tools are designed for the high-performance computing market, to handle thousands of threads across thousands of cores. Others perform very in-depth analysis of operating systems events and their interaction with the hardware. The key question to ask is: does this suit the application being tuned?

Software Challenges When Developing Applications for Multiprocessor Embedded Systems

Conclusion

The challenges and difficulties associated with the development of complex, high performance applications are well recognized—and the focus on this has only increased with the growing pressure on developers to move applications to deployment in a shorter time, at lower cost and with minimal risk. That's why a number of tools to support the process have become available, notably from software vendors as well as CPU and DSP hardware suppliers. However, the majority of these tools address only a few of the issues that can be encountered—and mixing and matching a variety of tools from a range of vendors can be daunting and potentially time-consuming.

The ideal tool would be an integrated solution with a full suite of functionality from visualization through debugging, optimization, integration and verification. It should be portable across hardware architectures and operating system environments and support scalability. Ideally it should also intuitive and easy to use. Such a tool can make a substantial contribution to the on-time delivery of the most challenging applications.

WE INNOVATE. WE DELIVER. YOU SUCCEED.

Americas: 866-OK-ABACO or +1-866-652-2226 **Asia & Oceania:** +81-3-5544-3973
Europe, Africa, & Middle East: +44 (0) 1327-359444
Locate an Abaco Systems Sales Representative visit: abaco.com/products/sales

abaco.com | @AbacoSys

abaco SYSTEMS

©2017 Abaco Systems. All Rights Reserved. All other brands, names or trademarks are property of their respective owners. Specifications are subject to change without notice.