A Programming Model
for
Migrating Multicore Applications

Simplifying multicore migration

White Paper
November 9, 2010

Ted Gribb
PolyCore Software Inc.
http://www.polycoresoftware.com/

Abstract

Several multicore architectures and programming models are available for developers to use when migrating applications to multicore platforms. Here, we will look at a multicore programming model as applied to migrating an application running across multiple single core DSPs to running on multiple multicore DSPs.

Moving to Multicore

Multi-chip, multi-processing, multi-execution path have been available technologies for decades and have been used to solve complex problems and improve performance. Since single core technology could address the vast majority of compute needs with constantly increasing performance, the tools and buzz was centered on single core processors.

The computer world is changing as should be the case. Three factors that have been driving the change continue to be the key drivers:

1. Power: increasing the MHz for a single core is approaching the physical limits. And, whether the product is battery operated or tethered by a power cord, developers are seeking to improve the performance/watt ratio.
2. Performance: Application features and functionality will continue to expand which fuels increasing compute performance.
3. Consolidation: The progression of more capabilities on a single chip. Applications currently running on multiple processors will move to multicore, and, applications currently running on multicore will move to denser multicore chips.

The Programming Model chosen for multicore applications must have the flexibility to address all three factors. While addressing power has been the focus of many single core applications moving to multicore, the discussion presented addresses the later two factors – consolidation and performance.

The broad range of platform technologies offers the system designer choices for the platform that would best fits the application's requirements. A sample of the hardware options include homogeneous and heterogeneous multicore processors, processors with accelerator integration as well as processor with optimized mechanisms for, inter-processor (IPC)/core communications (transports). The software paradigms executing on multicore processors also have variants like SMP[i] (Symmetric Multiprocessing), AMP[ii] (Asymmetric Multiprocessing), CPU affinity and virtualization, plus, the combinations and permutations of the models indicated above

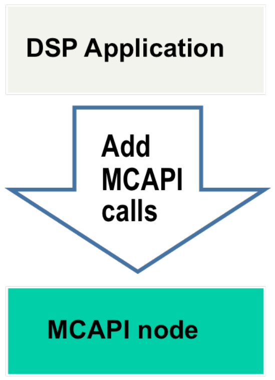
Programming Model

The Programming Model used in any multicore software paradigms needs to be consistent, have the flexibility to scale and enable optimizations across various hardware and software multicore and multi-processor platforms which are available today and are being planned.

The Multicore Association (MCA)[3iii] communications API – MCAPI, and TI DSPs are used to demonstrate a programming model for applications consolidating to denser processors and then looking for better ways to execute the application in a multicore environment.

MCAPI has a concept called node which is defined as a logical abstraction that can be mapped to many entities[iv] such as process, thread, hardware accelerator, or a processor core. For this example, a node will be the entire application that executes on a single DSP.

The application is encapsulated with MCAPI calls and we assume that the IPC transports are available.



{ MCAPI initialization }
{ MCAPI receive data }

[DSP Application Code]

{ MCAPI send data }

Now that our application is set up for MCAPI as encapsulated modules, the first pass on testing could be performed using any of the following environments:

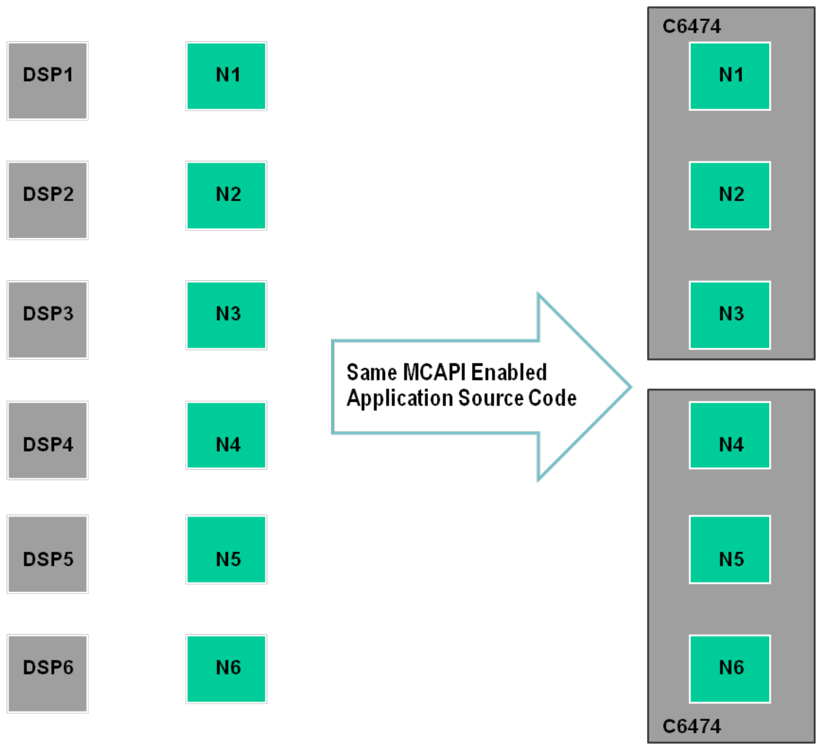
1. Windows provided that non-Windows calls are handled - such as an OS simulator on Windows
2. Hardware simulator where communications links are simulated
3. Existing single core hardware
4. Multicore hardware

Initial testing on target hardware could be challenging because of the number of variables introduced and this subject better handled by the debug experts. A simulated environment provides more control. However, eventually, the application will be moved to the target platform.

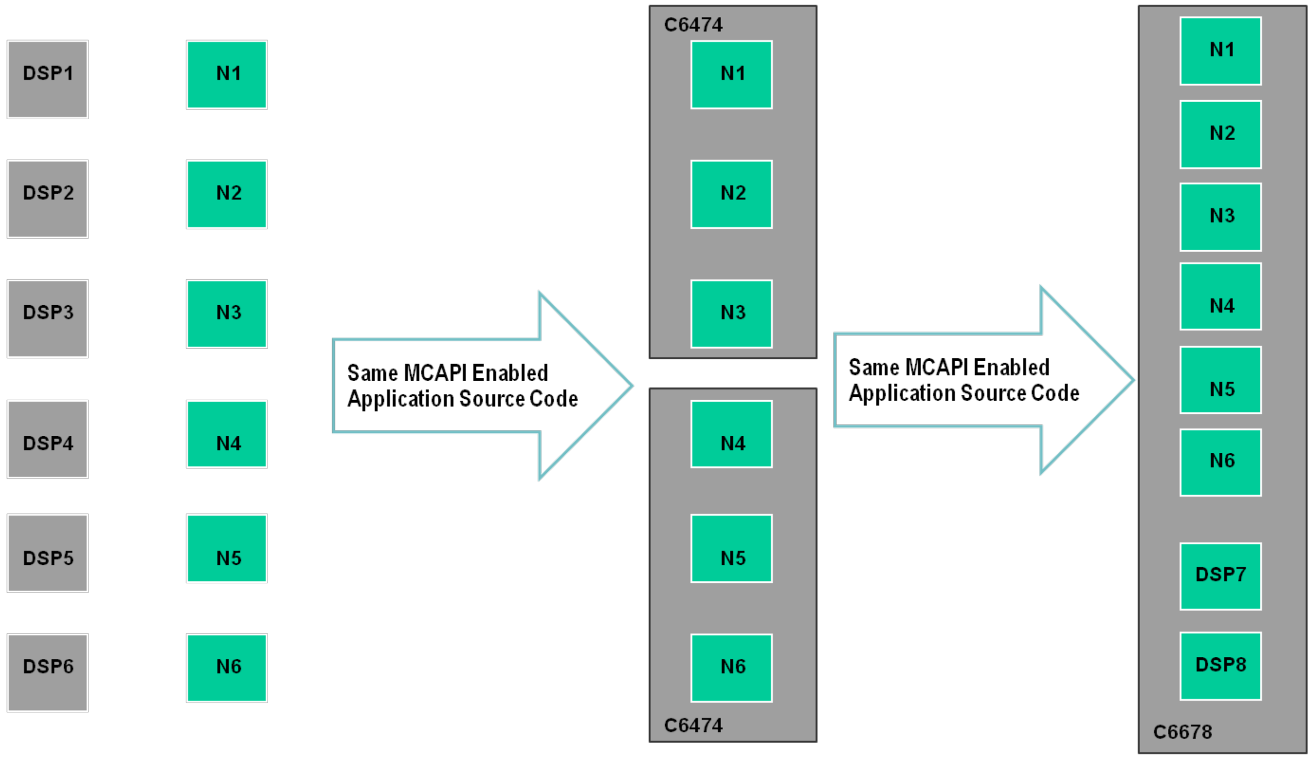
Applying the Programming Model

Let's look at the application moving the single core application to a multicore processor using Texas Instruments' (TI) DSPs such as the TMS320C6474 and then the TMS320C6678. The MCAPI enabled application communicates with other DSPs in the architecture which may be a single core DSP and/or a multicore C6474. That is, the same application source code as node(s) may be run as:

1. A single core DSP communicating to other single core DSPs
2. A multicore DSP communicating to DSPs on the same chip
3. A single core DSP communication to nodes on a multi-core DSP processor
4. A multicore DSP communicating to local DSPs or DSPs on a different multi/single core processor



By extension, moving to denser processors, that is, more cores on a chip, the MCAPI enabled application source code can move without source code change.

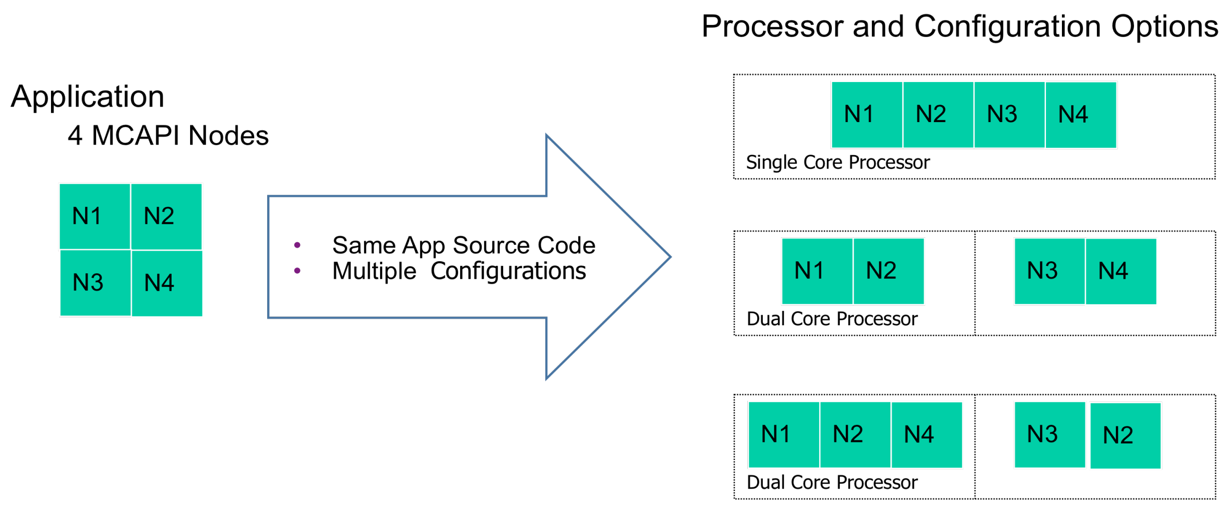

DSP1
DSP2
DSP3
DSP4
DSP5
DSP6
N1
N2
N3
N4
N5
N6
Same MCAPI Enabled Application Source Code
C6474
N1
N2
N3
N4
N5
N6
C6474
Same MCAPI Enabled Application Source Code
N1
N2
N3
N4
N5
N6
DSP7
DSP8
C6678

In the previous example, note that application had six nodes that mapped one-to-one in a dual C6474 configuration. When moving to the C6678 which has eight cores, the designer/architect has two extra cores. The additional two cores could be used as backup or to expand functionality or to improve the overall system performance. For this system, the two cores will be used to improve performance.

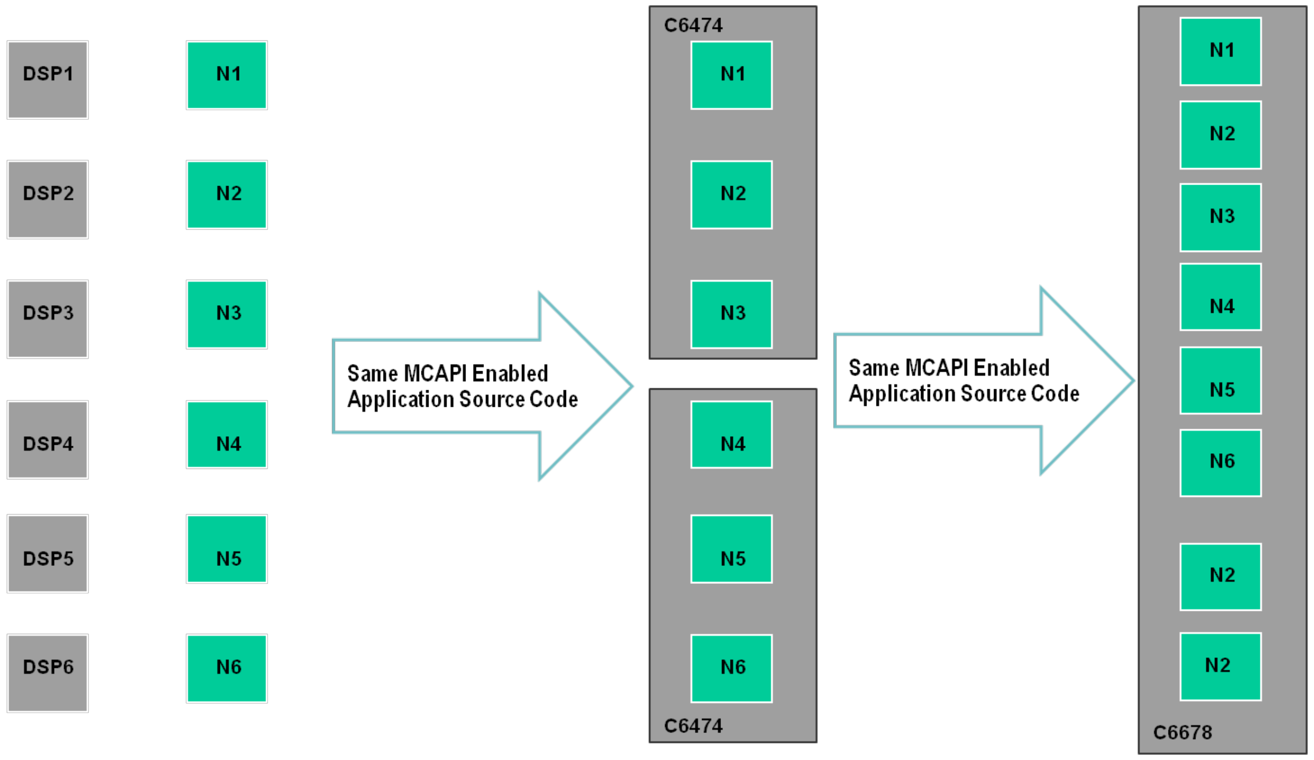
Optimizing the Application for Multicore

The MCAPI enabled application nodes provide flexibility in deploying the nodes throughout the system. Nodes can be grouped onto cores for systems with a memory protected operating environment and/or replicated to improve systems performance. By grouping the nodes, the system architect may be able to balance the operational load by having nodes with lesser resource needs share the resource. Should a node need substantially more resources, the node could be assigned to a resource and replicated which would allow more data sets to be processed in parallel.

The following diagram shows an application with four nodes that are mapped to three configurations. In the top configuration all nodes operate on a single core. In the dual core architecture, the nodes have different allocations. The middle mapping shows an optimized configuration where the resource load for the first two nodes is approximately the same as the later two nodes. Should the architect find that the nodes have very different resource loads, the lower configuration may be the best fit for the application. Important to note that the nodes are redeployed using the same application source code.



Returning to the application running on TI DSPs, the additional two DSPs could be used by node(s) that are more resource intensive. Suppose that N2 is substantially more resource intensive. Then, the architect may replicate N2 and deploy N2 on the seventh and eighth DSP which should yield an improved system performance. The new configuration is seen in the following diagram.


DSP1
DSP2
DSP3
DSP4
DSP5
DSP6
N1
N2
N3
N4
N5
N6
Same MCAPI Enabled Application Source Code
C6474
N1
N2
N3
N4
N5
N6
C6474
Same MCAPI Enabled Application Source Code
N1
N2
N3
N4
N5
N6
N2
N2
C6678

Alternately, the additional core could be used to improve a single node's performance. As the developer learns more about the application's behavior, a "resource intensive" segment of a node may be identified. The "resource intensive" segment may be separated from the original node, encapsulate as described earlier and made into a new node. This new node could be move to the additional core(s) where a improvement in performance could be realized.

Programming Model for today and for the Next Generation

Multicore platforms are being used and we know more applications will be moving to multicore platforms. Most applications have been written as serial tasks or processes which should be well defined execution blocks. Architects can employ the programming model with MCAPI described in this paper to move their application to multicore platforms with minimal effort.

Also, the MCAPI enabled application has the benefit to quickly move to the next generation multicore platform.

For this discussion, the nodes were deployed to homogeneous cores. The programming model readily extends to heterogeneous cores and processor platforms. Thus, a node could be allocated to a core that would better fit the node's workload.

Economic benefits for the manufacturer are that the same application source code could be used across a product line where the underlying platform differs by compute power. An entry level product could have a single or fewer cores platform. A mid-range product could have more compute power. The high end product could have a heterogeneous platform. Regardless of the products underlying platform, the same application source would be used.

By adopting the programming model based on the MCA standards, the developer is able to start using multicore platforms today. As more is learned about the applications operation, the application can be modified to operate as more nodes with a result of improved performance and improved scalability for more core platforms and next generation platforms. The programming model works for today's platform and prepares the application for tomorrow's platforms.

---

[i] SMP - http://en.wikipedia.org/wiki/Symmetric_Multi-Processing
[ii] AMP - http://en.wikipedia.org/wiki/Asymmetric_multiprocessing
[iii] MultiCore Assocation - http://www.multicore-association.org
[iv] Multicore Communications API Specification, MCAPI - http://www.multicore-association.org/workgroup/mcapi.php

Trademarks
All trademarks are the property of their respective owners.