IEC 63504-2804

Edition 1.0 2023-10

INTERNATIONAL STANDARD

IEEE Std 2804™

colour inside

Software-Hardware Interface for Multi-Many-Core

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.01, 35.060

ISBN 978-2-8322-7514-6

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

SOFTWARE-HARDWARE INTERFACE FOR MULTI-MANY-CORE

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

   IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see* https://standards.ieee.org/ipr/disclaimers.html *for more information).*

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.

8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

IEC 63504-2804/IEEE Std 2804 was processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
2804 (2019)	91/1873A/FDIS	91/1889/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

IEEE Standard for Software-Hardware Interface for Multi-Many-Core

Developed by the

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 7 November 2019

IEEE SA Standards Board

Abstract: This standard is intended primarily for tool developers and hardware developers who would use Software Hardware Interface for Multi-Many-core (SHIM) to exchange hardware description for software tools. It also attempts to provide software developers with insights into what hardware information is described in SHIM to foster understanding of the intention and the extent of SHIM.

Keywords: IEEE 2804, many-core, multicore, SHIM, software hardware interface, software tools

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/ipr/disclaimers.html.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. A current IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Xplore at http://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IEEE Introduction

This introduction is not part of IEEE Std 2804-2019, IEEE Standard for Software-Hardware Interface for Multi-Many-Core.

This document is intended primarily for tool developers and hardware developers who would use SHIM to exchange hardware description for software tools. It also attempts to provide software developers with insights into what hardware information is described in SHIM to foster understanding of the intention and the extent of SHIM.

This document begins with the introduction to SHIM, providing the background, the overall concept, and model. It is followed by a clause detailing the concept of SHIM, such as the purpose, scope, design, interface, limitation, providing the basic idea why SHIM is as specified in this document and also trying to explain the basic principles for future extension of the specification. A clause describing the interface follows, which is a description of SHIM XML schema and APIs that are mostly derived directly from the schema via XML data binding technique. A clause providing some of the detailed use cases follows, allowing the reader to gain insights into how SHIM can be used in action. Finally, this document ends with various annexes that provide further detailed information.

IEEE Standard for Software-Hardware Interface for Multi-Many-Core

1. Overview

1.1 Scope

The scope of this standard includes performance estimation accuracy for complex processors like Very Long Instruction Word (VLIW) core and complex contention scenarios, description of caches to include uncached memory regions and caches for subsets of memories, properties for coarse power consumption estimation, and reusability by separating eXtensible Markup Language (XML) files for processor description and other memory/communication-related information.

1.2 Purpose

The Software-Hardware Interface for Multi-Many-core (SHIM) defines an architecture description standard from the software design perspective—this provides a common interface that abstracts the hardware properties that are critical to enable multicore tools.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals is required to).[1, 2]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (should equals is recommended that).

The word *may* is used to indicate a course of action permissible within the limits of the standard (may equals is permitted to).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to).

[1]The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.
[2]The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

1.4 General introduction

Multicore processors have become the norm, and processors with tens and even more than a hundred cores are emerging. These multicore and many-core processors vary not only in the number of cores, but also in inter-connects, cluster organization, and memory systems (including hierarchy and cache coherency), among others. While the trend for an increasing number of cores is both natural and unavoidable from a processor design perspective, this poses tremendous challenges to the software developers to cope with the significant hardware variance, while bearing a burden to re-use the existing and newly created software for different hardware. Moreover, all this must occur while achieving the performance expected from the multicore processors, which requires a deep understanding of the specific multicore architecture. Various tools, such as auto-parallelizing compilers, parallelization tools, OS/middleware configurators, and performance analysis tools, aid developers to design, implement, and analyze the software. However, these tools must comprehend the complex multicore processor, transferring the burden to the tool developers. Therefore, lowering the cost of supporting new multicore hardware by various tools is critical, but there has been a lack of effort in academia or industry to solve these issues, thereby hindering the development of the multicore tool eco-system.

The SHIM, Software-Hardware Interface for Multi-many-core, is a joint industrial and academic effort to standardize the interface between the multicore hardware and the software tools. As a result, the aim is to lower the cost of supporting new multicore hardware using the standard interface. This will encourage the development of new innovative multicore tools, resulting in a richer eco-system of multicore technologies, which in turn should benefit system developers, semiconductor vendors, and tool vendors.

1.4.1 Interface

The SHIM is defined as an XML schema. A multicore hardware implementation is expressed as a set of SHIM XML files that can be used by various tools (see Figure 1).

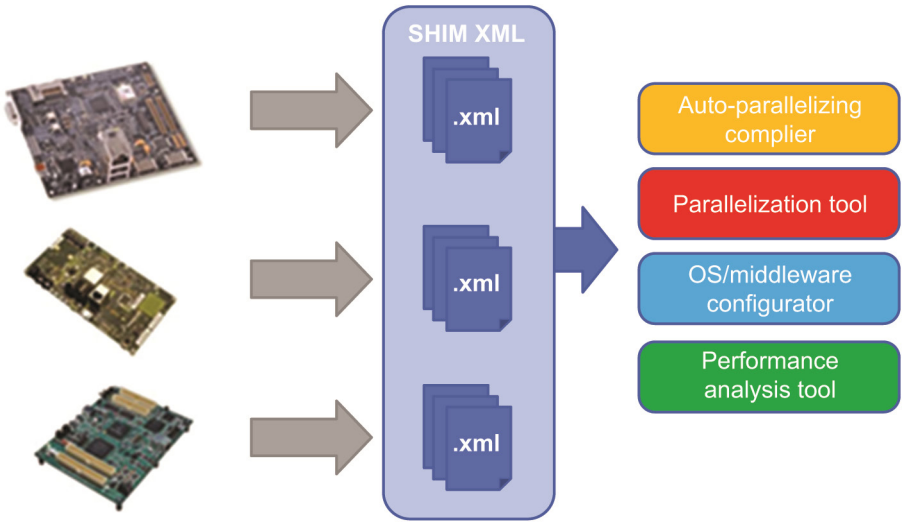

Figure 1—SHIM provides the interface between the hardware and the software tools

The SHIM XML file has a tree structure in which XML elements are used to describe the architectural and platform properties of a multi-many-core system. Figure 2 illustrates the main SHIM XML elements. A system description starts with a *SystemConfiguration* element with up to five children components, namely *ComponentSet*, *AddressSpaceSet*, *CommunicationSet*, *FrequencyVoltageSet,* and *ContentionGroupSet* each containing further child elements. The *ComponentSet* contains *MasterComponent (*representing a processor or accelerator) and *SlaveCodemponent (*representing a memory block or memory subsystem). The *AddressSpaceSet* contains one or more *AddressSpace*, which in turn contains *SubSpace*. The

CommunicationSet contains any number of *Communication* elements, describing the connection and communication between a pair of *MasterComponents*. The *FrequencyVoltageSet* contains definitions of frequency and voltage domains that defines sets of components sharing the same clocks or input voltage. It also contains definitions of component operating points, through *OperatingPointsSet*. Finally, the *ContentionGroupSet* allows for the defining of *ContentionGroup*s, which are communication resources used by component-to-component communications. Using those *ContentionGroup*s allows users to determine communication resources utilization and potential contentions.

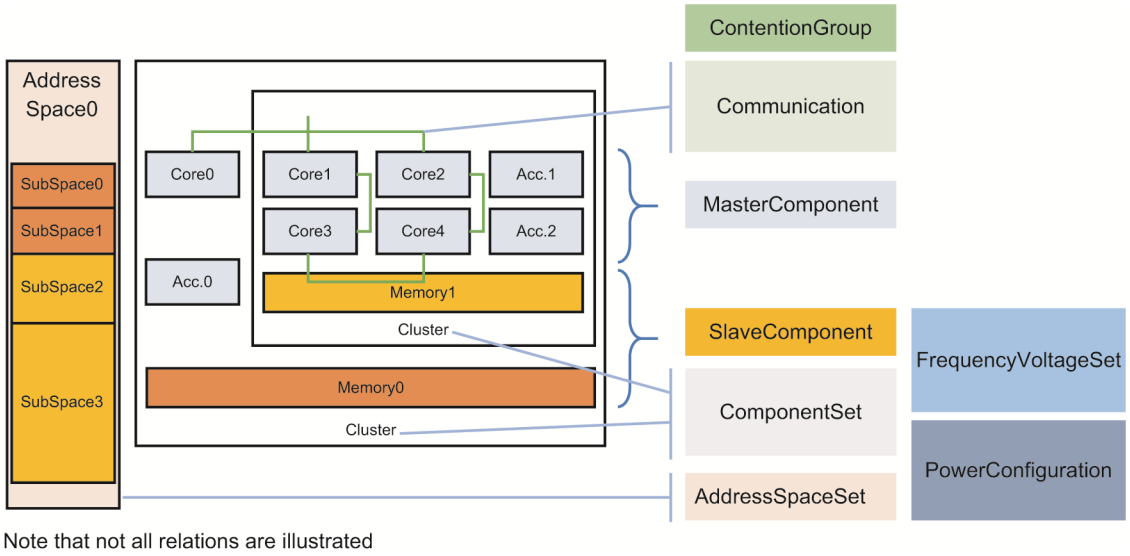

Figure 2—The SHIM elements mapped to a pseudo-multicore hardware

A *ComponentSet* can nest itself. For example, it can be used to express a chip that contains multiple hardware clusters, each cluster containing multiple cores with a cluster local memory. It can also be used to describe a board, which in turn may contain one or more multicore chips. A *ComponentSet* can even be used to describe a system with multiple boards, each board connected via PCI Express, for example. As such, the *ComponentSet* tree describes the multicore hardware system topology. This topological architectural information is important for software tools to be able to identify the number of cores, the location of the memory devices, and how cores are organized into different clusters.

Since SHIM is for software tools, it is essential to understand from a software perspective, the connection and communication mechanism between the cores (including accelerators), as well as how these cores can access the different memories. The former is described as *CommunicationSet* containing different communication classes. A simple example of defined classes is *InterruptCommunication*, which contains one or more "connection" classes, which binds a pair of *MasterComponents*. For memory access, the *SubSpace* contained in the *AddressSpace* includes its start address and size and one or more *MasterSlaveBinding*, containing references to a *MasterComponent* and *SlaveComponent*, describing which core/accelerator can access which memory through the address range.

The hardware architectural information described so far allows tools to understand the hardware topology, and how the cores and memory devices are connected. However, this alone is often insufficient for many tools, since the application software supported by these tools must not just 'run', but run with performance qualifiers. To achieve this, the tools must 'estimate' the rough performance so that the system designers and software developers know the expected performance from the given application and multicore hardware. Therefore, SHIM, in addition to the hardware topological information, describes the performance properties associated with the processor cycles consumed to perform the various core-to-core communication (*CommunicationSet*) and also the memory access cycles by different cores and accelerators. The performance is described as *Performance* element, which contains *Latency* and *Pitch*, expressed in processor cycles. The *Performance* element exists for each *CommunicationSet*, for each specific pair of two

MasterComponents. For memory access performance, for each *MasterSlaveBinding* of each *SubSpace*, and for each *AccessType*, which are defined for each *MasterComponent*, a specific *Performance* element is included. So, for each different access type (e.g., read or write, word access, or double word access), a different *Performance* element is provided. The cycles can be described in a form of the triplet, which is 'best', 'typical', and 'worst', to accommodate the possible performance variance. The tool must be intelligent enough to benefit from these figures, such as analyzing the application code if it is issuing a sequential memory access, which generally falls into the use of the 'best' cycles. Note that the cycles mentioned here are processor-cycles, whose related frequency is defined using *OperatingPoint*s.

1.4.2 Software view—what is in and what is not

Tools should primarily use SHIM to aid developing software that runs on multi-many-core hardware. Therefore, the key strategy in defining the SHIM specification is to describe the hardware but only for the information that is relevant to such tools. This is referred to as a 'software view' of hardware, as opposed to 'hardware view', where the focus would be the physical/electrical means of inter-connects between processing cores, the Network on Chip (NoC) protocol used to route the memory read request by a particular core, the number of processor pipeline stages, the cache coherency protocol, etc., unless these features matter greatly to some class of tools that aid software development.

It is tempting and relatively easy to include additional hardware properties in SHIM; however, this will result in a more complex SHIM XML, requiring more effort to grasp the schema and complicating the effort for tools to use this information. Furthermore, the most critical issue is the challenge to create a SHIM XML in the first place—leading to limited adoption of the SHIM standard.

The basic principle is to capture the properties that affect the software at the architectural design level. This is to say, if a design-aid tool uses SHIM to produce an appropriate software design for a particular hardware described by a SHIM XML, then the design should not require modification at the software architectural level at the later stages of system development.

Although the "software architectural design level" is the baseline, it is sometimes difficult to agree on whether a particular hardware property is important. The rule of thumb is that if an actual (even imaginable) use case cannot be derived, the SHIM specification excludes it.

For various reasons, a number of potential hardware properties have not been included in the current specification. One of the primary reasons is that the excluded types of hardware properties are peripheral to existing properties in the specification. Such hardware properties may be included in a future version of the specification, but it was decided to take an evolutionary approach and stabilize the more basic properties first.

The most basic properties selected for inclusion in SHIM are the following: topology, address space, inter-core communication, and performance and configuration.

With SHIM 2.0, several optional additions have been included in the interface, namely *FunctionalUnitSet*, to define in which functional unit an instruction is executed and improve estimation accuracy, *ContentionGroup* to handle communication resource utilization and contention, which may impact software execution performance, *OperatingPoints,* and *PowerConfiguration*, which may influence software implementation strategy or automatic optimization.

Clause 5 of this document illustrates how the information provided by a SHIM XML file can be exploited by software development tools. Three examples are provided: performance estimation, tool configuration, and hardware modeling.

1.4.3 XML

The SHIM interface uses extensible markup language (XML); specifically, the SHIM XML uses the XML Schema (XSD) to define its XML structure. XSD is essentially the same as a UML class diagram. Each unique hardware is described by one or more SHIM XML files. There must be at least one file for the system description. Optionally there could be separate files for core instruction set descriptions. The standard also allows for the defining of additional files for power configuration and for vendor extensions. All files must conform to the SHIM XML schema. The UML class diagram representation of SHIM XML schema is shown in Figure 3.

The XML schema allows the definition of the SHIM XML structure, but with the help of a validating parser that reads XML files, the schema also allows validation of SHIM XML. Validating parsers are readily available, both openly and commercially, often bundled with various XML related libraries in many different programming languages.

Therefore, technically speaking, the SHIM XML schema, or the shim20.xsd, is the core interface definition of SHIM.

1.4.3.1 Data binding

A common technique to read XML files is via SAX or DOM libraries. Using XSD, it is possible to generate class libraries in many choices of programming languages by running a schema compiler against the shim.xsd. The generated library includes all the SHIM XML classes of the chosen programming language, with automatically added methods or functions to get and set the data. This allows tools to access hardware properties expressed in a SHIM XML similar to accessing normal objects in their programming languages.

1.4.3.2 Who creates SHIM XML

The hardware provider is expected to create and provide the SHIM XML, which will then be used by the software tools. On the other hand, a hardware provider may not provide the SHIM XML. If SHIM XML can only be authored by the hardware provider, it can be a significant roadblock in the hardware's adoption. Therefore, reference authoring tools (see 1.4.5) are made freely available along with the specification. If a user has access to the basic technical reference manuals, and either simulator or actual hardware (e.g., evaluation board), the reference authoring tools allow for the creation of the SHIM XML for most multi-many-core hardware in fewer than 1−2 days.

1.4.4 SHIM Editor

Although a SHIM XML schema is relatively simple, as can be seen on the UML Class diagram representation of the SHIM XML schema (see Figure 3), the resulting SHIM XML file can be quite large, mostly due to all the *Performance* element descriptions for all types of memory accesses. Writing it manually can be tedious and error-prone, so an editor tool called the SHIM Editor has been developed to foster authoring a SHIM XML file. The generated SHIM XML file is shown in Figure 4, and the SHIM Editor prototype's main window is shown in Figure 5.

1.4.5 Reference authoring tools

In addition to the specification itself, SHIM also provides a free set of reference authoring tools. As a reference, anyone can provide their own version of the SHIM authoring tools. The OpenSHIM[3] provides the reference authoring tool, SHIM Editor, for the following reasons:

a) Easy authoring of SHIM XML to enable better adoption

b) Serves as a sample SHIM application with source code

[3] See https://github.com/openshim/shim.

1.4.6 Changes introduced in SHIM 2.0

The following changes have been introduced in SHIM 2.0:

- SHIM 2.0 introduces processor functional units modeling with the new *FunctionalUnitSet* element, which allows for the defining of the functional units of a processor and their supported instructions. See 6.6.7 and 6.6.8 for more information.
- SHIM 2.0 improves the SHIM XML file componentization and reuse by allowing to define a processor model in a separate file, through the use of the *FunctionalUnitSetFile.* See 6.6.8.
- The *Instruction* element has been extended with new parameters allowing, for instance, to define the instruction bit width, the number of input and output, the supported signedness, the instruction SIMD width, etc. The complete parameter list and more information can be found in 6.6.10.
- Instruction modeling has been further extended by allowing to define custom instructions using the *CustomInstruction* element. This element provides support for complex instructions. More information can be found in 6.6.11.
- SHIM 2.0 replaces the previous *ClockFrequency* element, which was limited to clocks definition, with the new *OperatingPointSet* element that allows for the defining of one or more operating points (consisting in a frequency value and an optional voltage value) for system components. This enables modeling of dynamic frequency scaling and/or dynamic voltage scaling for systems that support it. See 4.3 for more information.
- SHIM 2.0 introduces frequency and voltage domains, which allow defining sets of components sharing resp. the same frequencies or the same voltages. See 4.3 for more information.
- SHIM 2.0 allows improves cache hierarchy definition by allowing the definition of more complex cache hierarchies, like for instance memory-side caches. *MasterComponent*s now refer to their related *Cache*s using the *CacheRef* element. See 6.6.3 for more information.
- SHIM 2.0 further improves cache modeling by extending the *Cache* element with new parameters such as replacement policy, write back/write allocation policy, prefetch support, etc. The complete parameter list and more information can be found in 6.6.3.
- In SHIM 2.0, cached regions can be explicitly defined by referencing the cache element caching the memory accesses in a *PerformanceSet* element. See 6.8.7.
- SHIM 2.0 introduces support for communication resource utilization and contention modeling with the new element *ContentionGroup* and *ContentionGroupSet*. See 4.4 for more information.
- SHIM 2.0 introduces the new *PowerConfiguration* element, which enables power consumption modeling by defining the system components' power consumption. To further improve componentization and flexibility, *PowerConfiguration* is an optional element defined in a separate file. See 4.6 for more information.
- SHIM 2.0 introduces the new *VendorExtension* element, which provides a solution for vendor desiring to extend a SHIM model with their own functionalities. To further improve componentization and flexibility, *VendorExtension* is an optional element defined in a separate file. See 4.7 for more information.
- SHIM 2.0 enforces a stricter XML schema to improve interoperability. A SHIM 2.0 XML file only allows the new *shim* element as the root element. This root element only accepts a *SystemConfiguration*, a *FunctionalUnitSet*, a *PowerConfiguration* or a *VendorExtension* child element. See Clause 6 for more details.

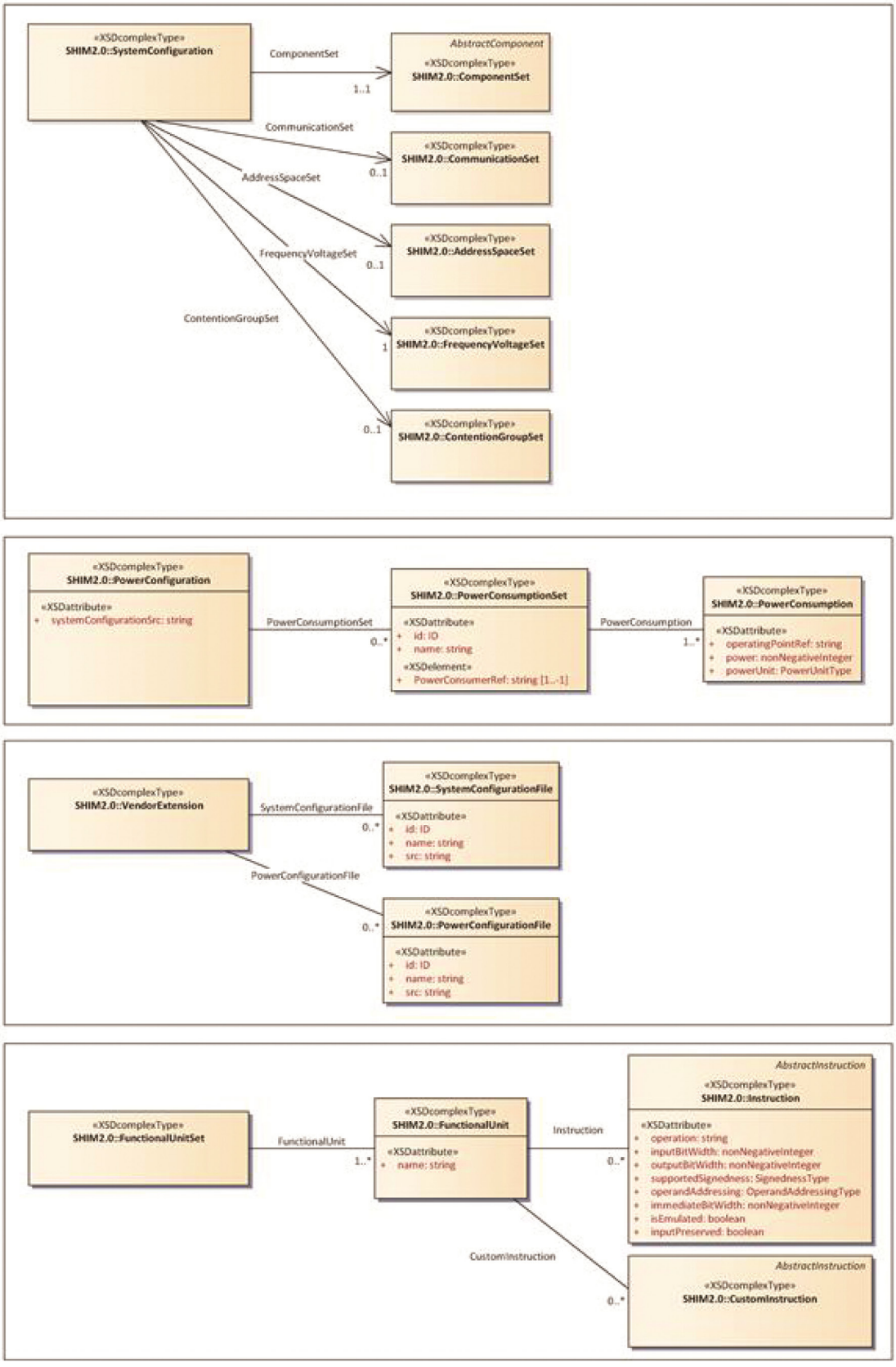

Figure 3—Class diagram representation of the SHIM XML schema (top-level elements)

Figure 4—SHIM XML file example

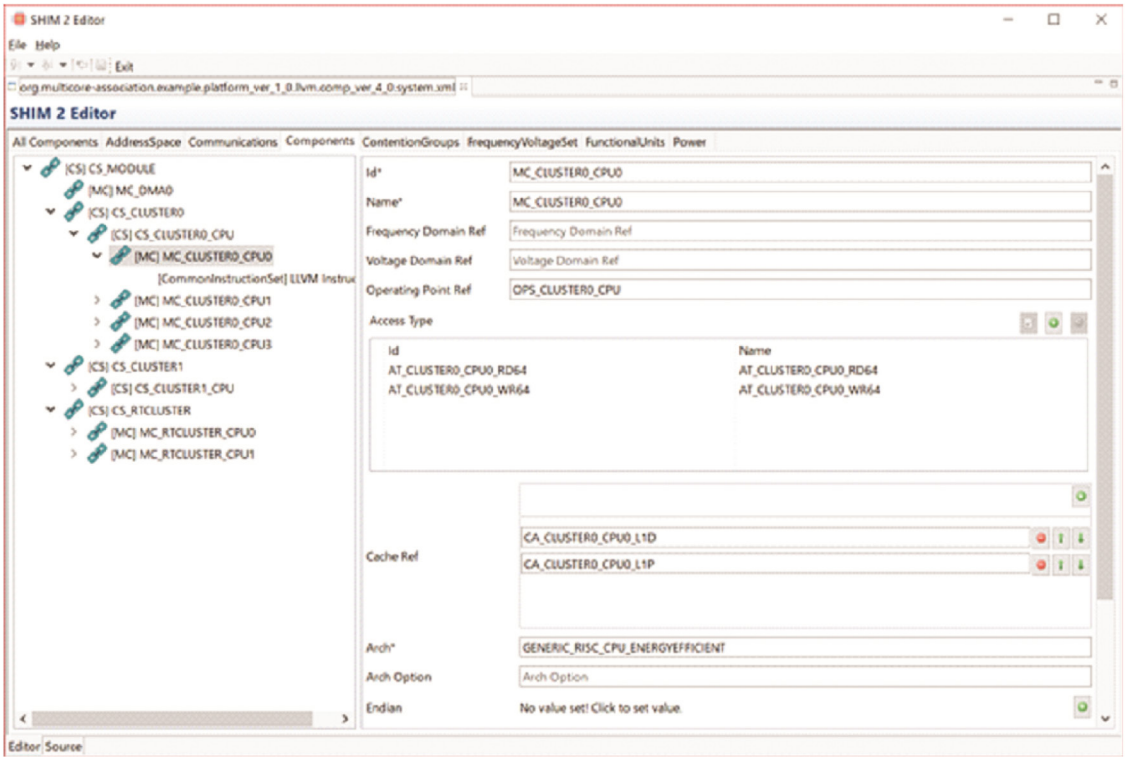

Figure 5—SHIM editor main window

2. Normative references

Not applicable