*Chapter*

# Tools and Workloads for Many-Core Computing

*Amit Kumar Singh*[1]
*Piotr Dziurzanski*[2]
*Geoff V. Merrett* [3] *and*
*Bashir M. Al-Hashimi* [4]

Proper tools and workloads are required to evaluate any computing systems. This enables designers to fulfill the desired properties expected by the end-users. It can be observed that multi/many-core chips are omnipresent from small scale to large scale systems, such as mobile phones and data centers. The reliance on multi/many-core chips is increasing as they provide high processing capability to meet the increasing performance requirements of complex applications in various application domains. The high processing capability is achieved by employing parallel processing on the cores where the application needs to be partitioned into a number of tasks or threads and they need to be efficiently allocated onto different cores. The applications considered for evaluations represent *workloads* and toolchains required to facilitate the whole evaluation are referred to as *tools*. Figure 1.1 provides three-layer view of a typical computing system, where the top layer contains applications and thus represents workloads. The tools facilitate realization of different actions (e.g., thread-to-core mapping and voltage/frequency control, which are governed by OS scheduler and power governor, respectively) and their effect on different performance monitoring counters leading to a change in the performance metrics (e.g., energy consumption and execution time) concerned by the end-users.

The design of multi/many-core chips has been the focus of several chip manufactures. The examples of some industrial chips include: Samsung Exynos 5422 System-on-Chip [1] that contains 4 ARM Cortex-A15 cores, 4 ARM Cortex-A7 cores and a six-core ARM Mali T628 MP6 GPU, Intel's Teraflop 80-core processor [2] and Xeon Phi 64-core processor [3], 16 and 64 core Epiphany processors [4], Tilera's TILE-Gx family 100-core processor [5], AMD's Opteron 16-core processor [6], Kalray's MPPA 256-core processor [7] and recently developed KiloCore 1000-core chip [8] by IBM and UCDavis. Even world's fastest supercomputers such

[1]School of Computer Science and Electronic Engineering, University of Essex, UK
[2]Department of Computer Science, University of York, UK
[3]School of Electronics and Computer Science, University of Southampton, UK
[4]School of Electronics and Computer Science, University of Southampton, UK

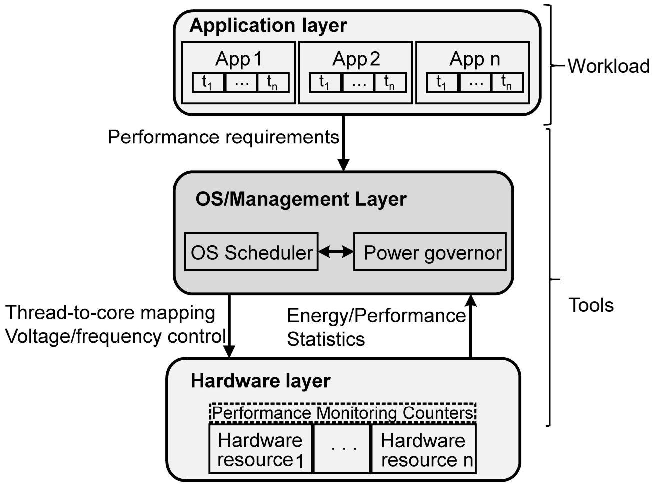

*Figure 1.1 Three-layer view of a computing system [11]*

as Tianhe-2 (MilkyWay-2) and Titan use many-cores and the total number of cores in Tianhe-2 is around 3 millions. The large number of cores within a chip is usually connected by an on-chip interconnection network [9, 10], whereas bus-based or point-to-point interconnections are used when the number of cores is small. The hardware in bottom-layer of Figure 1.1 can represent any of these chips.

These chips power systems of different scales to meet the respective user requirements. For small scale systems such as mobile phones and desktops, usually a single chip is used, whereas multiple chips are used for large scale systems such as data centers. Figure 1.2 classifies these systems into single-chip and multi-chip systems, where the hardware layer contains one and multiple chips, respectively. Examples of single and multiple chip multi-core systems are embedded systems (including mobile phones) and data centers, respectively. In embedded systems, typically a single chip containing small number of cores is used, e.g., Samsung Exynos 5422 System-on-Chip [1], which powers popular Samsung Galaxy series of mobile phones [11]. In a desktop computer, a chip having higher numbers of cores, e.g., Intel's Xeon Phi 64-core processor [3] and AMD's Opteron 16-core processor [6], are used [12]. An HPC data center connects a set of nodes (servers) [13], where each node contains a set of cores within a chip and the cores communicate via an interconnection network and the nodes communicate via a high-speed network, e.g., InfiniBand. When the number of cores within a chip is relatively smaller, it is referred to as a multi-core chip and the cores are usually interconnected by a shared bus or point-to-point links. However, the chip is referred to as a many-core chip when the number of cores is relatively higher and they are usually connected by a network-on-chip. Further, some of these systems might incorporate cores of different types to achieve efficiency over only one types of cores [14].

As shown earlier, since many-core systems can employ a single or multiple chips, it is important to identify appropriate tools and workloads to evaluate them.

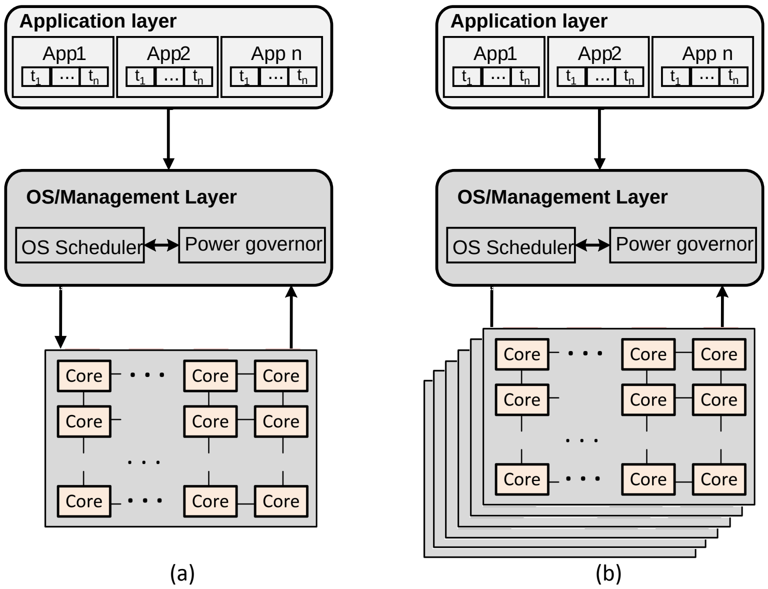

*Figure 1.2 Many-core systems with: (a) single chip and (b) multiple chips*

In this chapter, the tools for these systems are reviewed from three categories: *i)* Toolchains or scripts generated by designers to map and schedule application codes, e.g. C/C++ codes, on real hardware platforms, e.g. Samsung Exynos 5422 System-on-Chip [1], *ii)* Simulation tools, to evaluate systems by simulating the descriptions of applications and architectures at a high level, e.g. task graphs [15] and synchronous data flow graphs [16], and *iii)* Commercial tools or software development environments to program the real hardwares to run application(s), e.g., Xilinx's Software Development Kit (SDK) to program many-core systems available or created in a field-programmable gate array (FPGA) chip [17].

This chapter is organized around the descriptions/discussions of tools and workloads for these systems as follows. Section 1.1 provides overview of identified tools and workloads for systems using single chips. The same has been covered in Section 1.2 for systems using multiple chips. Section 1.3 provides a discussion about the tools and benchmarks covered in Sections 1.1 and 1.2. Section 1.4 concludes the chapter.

## 1.1 Single-chip Multi/Many-core Systems

In this section, the typical tools used for design and analysis of single-chip many-core systems are investigated. Then, the characteristic workloads of these systems are discussed.

*Table 1.1 Tools for single-chip multi/many-core systems*

| Category | Reference | Comments/Remarks |
|---|---|---|
| Toolchains/scripts | Epiphany SDK [18] | For Epiphany |
| | COPRTHR SDK [19] | For Epiphany |
| | ARL OpenSHMEM [20] | For Epiphany |
| | ePython [21] | For Epiphany |
| | OMPi OpenMP compiler [22] | Ported for Epiphany |
| | Epiphany BSP [23] | For Epiphany |
| | SMYLE OpenCL [24] | Framework for OpenCL |
| | Adrenaline [25] | Framework for OpenVX |
| | GSNoC [26] | Frameworks for 3D NoC design perspectives |
| Simulation | HORNET [27] | Applicable to many-cores of various scales |
| | FOLCS [28] | Applicable to many-cores of various scales |
| | BookSim2 [29] | Applicable to many-cores of various scales |
| | Gem5+GPU [30] | Applicable to many-cores of various scales |
| | VIPPE [31] | Parallel native simulation |
| | SMVM-NoC [32] | NoC simulator based on OMNeT++ |
| | OVPSim [33] | Fast simulation of virtual platforms |
| Commercial tools | Multicore Development Environment$^{TM}$ [34] | For TILE-Gx72 and TILE-Gx36 |
| | MPPA® DEVELOPER [35] | For Kalray MPPA2®-256 |
| | AccessCore® SDK [35] | For Kalray MPPA2®-256 |
| | eMCOS IDE [36] | Profiler, trace analyzer for eMCOS |
| | Sourcery CodeBench [37],[38] | Complete development environment |
| | AbsInt aiT [39] | WCET static analyzer |
| | Open Virtual Platforms (OVP) [33] | For creating software virtual platforms |

### *1.1.1 Tools*

Table 1.1 lists various tools employed to evaluate single-chip multi/many-core systems. The first few entries list developed tools for the Epiphany microprocessor, probably the most widely supported many-core architecture. Despite the recent problems related to the public release of the 1024-core Epiphany-V version, its predecessor Epiphany-III is still publicly available as a co-processor in the Parallella Board [4]. The Epiphany SDK [18] and COPRTHR OpenCL SDK [19] are officially supported. The former is comprised of Eclipse IDE, GCC, GDB, an Epiphany driver, loader and runtime library. The toolchain includes a functional simulator for a single core. The CO-PRocessing THReads (COPRTHR) SDK provides libraries and tools facilitating programming low-power many-core RISC co-processors. It offers a portable API for targeting accelerators with an MPI programming model for parallel code development. An integrated many-core co-processor debugging tool is included. An open-source OpenCL implementation is available.

The US Army Research Laboratory (ARL) has developed the ARL OpenSHMEM for Epiphany, which is a standardized interface to enable portable applications for partitioned global address space (PGAS) architectures. Its high-performance execution while approaching hardware theoretical networking limits is demonstrated in [20]. ePython is a Python-based parallel programming environment for Epiphany and similar many-core co-processors [21]. It offers the capability of offloading specific Python functions (kernels) from an existing Python code to a many-core co-processor. Despite OpenMP was intended for shared memory multiprocessing programming and thus is more suitable for SMP architectures than for the cores with local memories, there exists its implementation for Epiphany [22]. A Bulk Synchronous Parallel (BSP) programming environment developed by Coduin is also available [23]. The programs following the BSP model are comprised of so-called supersteps performing local computations and non-blocking communication, finished with a barrier synchronization. Some popular Google technologies as MapReduce and Pregel are based on this model.

The processors developed by Tilera, including TILE64, TILEPro64, TILEPro36, TILE-Gx72, TILE-Gx36, TILE-Gx16 and TILE-Gx9 are also suitable for many-core embedded systems. After the acquisition of EZchip in February 2016, TILE-Gx72 and TILE-Gx36 are offered by Mellanox Technologies. This company offers the toolset named Multicore Development Environment$^{\text{TM}}$ (MDE) [34]. In this environment, cross-compilation is performed using a typical C/C++ GNU compiler. An Eclipse IDE facilitates many-core application debugging and profiling. A complete system simulator and hardware development platform is also available.

Kalray offers MPPA2®-256 (Bostan) many-core processors with 288 cores, optimized for networking and storage applications [35]. The EMB boards from the same company provide a complete environment to develop compute-intensive embedded systems. This processor is programmed using MPPA® DEVELOPER and AccessCore® Software Development Kit (SDK). Kalray's SDK is based on Eclipse and offers a set of simulation, profiling, debugging and system trace tools. Three programming styles are allowed: a low-level DSP style, POSIX-level CPU Style and GPU style based on OpenCL.

On Kalray MPPA, eMCOS [36] and ERIKA Enterprise [40] operating systems can be installed. The eMCOS IDE Plug-in development tools shipped with the former OS consist of eMCOS-specific system analysis tools and utility software for building, debugging and system analysis. They include Real-time Profiler for runtime analysis of each core, thread and function, Message Profiler for analysis of the message communication behaviors from the OS to the driver, middleware and application and Trace Analyzer for tracing the system events. Kalray cores are also targeted by Absint aiT static analysis tool for determining the worst-case execution time of a given taskset [39].

Despite the existence of the OpenCL SDK for Epiphany or Kalray mentioned above, OpenCL is rather rarely used in embedded many-core systems in general. The reasons for this fact, as explained in [24], are the large runtime overhead for creation and mapping of threads at runtime. Similarly, memory buffers and command queues required for an OpenCL program execution are created at runtime. In that paper, SMYLE OpenCL, a framework for OpenCL dedicated to embedded many-cores is proposed. This framework reduces the runtime overhead by creating the threads and objects statically, as demonstrated on a five-core SMYLEref architecture implemented on an FPGA prototype board.

OpenVX is an open standard for cross-platform acceleration of computer vision applications specified at a higher level of abstraction than OpenCL. In OpenVX, a computer vision application is specified as a connected graph of vision nodes executing a chain of operations. In [25], an open framework for OpenVX named Adrenaline is presented. It targets an embedded system on chip (SoC) platform with a general-purpose host processor coupled with a many-core accelerator, such as STM STHORM, KALRAY MMPA or Adapteva Epiphany.

Embedded Sourcery$^{TM}$ CodeBench from Mentor® is a commercial set of embedded C/C++ development tools [37]. It includes an Eclipse-based IDE with a performance-optimized compiler based on GCC and optimized runtime libraries for selected embedded cores. An advanced software insight allows the developers to identify and correct functional, timing, and performance bottlenecks. The attached multi-core debugger facilitates simultaneous debug of multiple operating systems or applications running on different cores. The applications can be simulated using the QEMU hypervisor. This toolset has been used for embedded many-core systems in, e.g., [38].

A set of simulation tools, used for embedded many-cores, can be also applied to larger systems. The examples of such tools are HORNET [27], FOLCS [28], BookSim2 [29], GEM5 (with a GPU extension) [30]. However, there exist a couple of simulators that are dedicated solely to the small-scale many-core systems. One of them is VIPPE [31] that offers a parallel host-compiled simulation methodology that making an efficient use of multi-core host platforms. Some simulators are applicable only to embedded NoCs, such as SMVM-NoC, an OMNeT++ based Network-on-Chip simulator for embedded systems [32]. In this simulator, such parameters as network size, buffer size and clock frequency are customizable. To reduce the communication cost in NoCs, 3D chip technologies are emerging. Similarly, the necessary tools have been developed recently. One of them is Generic Scalable Networks-on-Chip (GSNoC) [26], which is a comprehensive design platform. It handles the 3D NoCs design at the application, architecture and circuit design levels. The platform is equipped with an application generator, design framework and a cycle accurate system simulator.

OVPsim [33] is one of the most mature embedded many-core simulators. This tool is a component of Open Virtual Platforms (OVP). OVP offers APIs allowing the users to model processors, peripherals and platforms to create software virtual platforms. Such platforms can be fast simulated with the OVPsim simulator, as the instruction accurate simulation can achieve up to 1,000 MIPS. Numerous example platform models including up to 24 processors are provided. These platforms benefit from the attached peripheral models, such as Ethernet or USB. Finally, several processor models can be used, including such families as OpenCores, ARM, Synopsys ARC, MIPS, PowerPC, Altera, Xilinx, Renesas. These OVP models are provided with interface wrappers for C, C++, SystemC and OSCI SystemC TLM2.0 environments. OVP is free for non-commercial usage and thus can boast with a huge community and numerous related research projects. For example, an accurate energy estimation for embedded many-core systems has been added to OVPSim in [38].

### *1.1.2 Workloads*

Table 1.2 lists various workloads/benchmarks used to evaluate single-chip multi/many-core systems. Among them are both industry-standard benchmark suites and the sets developed in academia.

*Table 1.2 Benchmarks for single-chip multi/many-core systems*

| Category | Reference | Comments/Remarks |
|---|---|---|
| Benchmark sets | Autobench 2.0 [41] | Commercially licensed |
| | MultiBench [42] | Commercially licensed |
| | SPLASH-2 [43] | HPC workloads mainly |
| | PARSEC [44] | Modern problems, not HPC |
| | E3S [45] | For high-level synthesis |
| | MiBench [46] | Wide range of embedded apps |
| | SD-VBS [47] | Vision domain apps |
| | Rodinia [48] | For heterogeneous platforms |
| | StreamIt [49] | Streaming apps |
| Popular workloads | Sobel [50],[25],[51],[52] | Edge detector filter |
| | MMUL [53],[54],[55], [56], [57] | Matrix multiplication |
| | QSORT [53] | Parallel versions from [58] |
| | NCC [50],[59],[55] | Normalized cross-correlation [60] |
| | FAST [55],[25],[61],[52] | Corner detection [62] |
| | Computer vision [55],[59] | Derived from OpenCV library [63] |
| | Canny [25],[52] | Edge detector |
| | Odd-even sorting [64],[65],[57] | Distributed sorting |
| | Papabench [66], Rosace [67] | Control apps of drone and plane |
| | Object tracking [65], [68],[69] | E.g. Vehicle localization |
| | 3D path planning [70], [71] | Avionic collision avoidance |
| | DemoCar [72] | Gasoline engine ECU |

Industry-standard benchmarks for embedded systems are licensed by the industry alliance named EEMBC. This organization offers benchmark sets for various mobile devices, networking, IoT, digital media, automotive, etc. Three of their suites are labelled as multi-core and can be also applicable to many-core architectures. The first of them, AutoBench 2.0, is dedicated to automotive processors. MultiBench, the second multi-core processor suite, includes more than 100 data processing and computationally intensive workloads for evaluating an impact of parallelization and scalability of multi- and many-core processors. Some workloads realize typical networking tasks (e.g., reassembling TCP/IP packets or compressing H.264 video streams), image processing (e.g., image rotations or color model conversions) or cryptographic functions (e.g., MD5 checksum calculation). These workloads are especially suitable for identifying memory bottlenecks and measure the efficiency of parallel task synchronization. Both AutoBench 2.0 and MultiBench are in a form of C/C++ codes intended to be run on a POSIX-compliant operating system. They may also be ported to a bare-metal platform with a custom scheduler, memory driver and thread synchronization mechanisms. Some examples of these benchmarks' licensing costs are provided in [42]. CoreMark-Pro is another set from EEMBC. It consists of 5 integer and 4 floating-point workloads including JPEG compression, XML parser, SHA256, Zip, FFT, linear algebra and a neural net. Some of these workloads (e.g., FFT or neural net) exhibit relatively low level of data dependencies and thus are more suitable for many-core systems than others (e.g., XML parser).

SPLASH-2 benchmark suite includes 11 workloads mainly from the high-performance computing domain (e.g., Cholesky factorization, FFT, LU decomposition) and graphic synthesis (e.g., Radiocity or Raytrace), which hardly cover the most typical modern usage patterns of parallel processing in many-cores. SPLASH-2 applications are written in C and are optimized to enhance scalability in the large-scale Cache Coherent Non-Uniform Memory Access (ccNUMA) architectures. In [73], some changes to make the suite compatible with modern programming practices and a number of bug fixes have been performed in order to port the original benchmark suite to a many-core architecture. The authors of [74] found that 7 original SPLASH-2 workloads contain data races due to the initial synchronization optimizations. They produced the SPLASH-3 suite, a sanitized version of the SPLASH-2 without data races and performance bugs, compliant with the contemporary C-standard memory model. Despite the year of its release, the SPLASH-2 benchmark suite still remains one of the most popular collections of multithreaded workloads. It has been used for an embedded many-core architecture evaluation in, e.g., [75]. Among the SPLASH-2 workloads, FFT and Cholesky are particularly often considered for many-cores, e.g. in [76], [71], [77].

PARSEC (Princeton Application Repository for Shared-Memory Computers) benchmark suite [44] contains fundamentally different types of programs than SPLASH-2. Among 13 workloads, there are representative applications from assorted areas such as enterprise servers, computer vision, data mining and animation. They reflect the contemporary computing problems and are not focused on the High-Performance Computing (HPC) domain. The applications are written in C and have been parallelized with pthreads and OpenMP. These workloads have been used for embedded many-cores in, e.g., [31],[75].

E3S is an embedded system synthesis benchmark set based on the EEMBC benchmarks suite. It includes task graphs of five applications from the automotive industry, consumer, networking, office automation and telecommunication areas without providing their codes (due to the EEMBC licensing restrictions). Consequently, their application is limited to the system-level allocation and scheduling, particularly when applied to the Network-on-Chip based architectures, as shown for example in [78].

MiBench [46] is a set of 35 applications covering the embedded system diversity at the time of its release (2001). The applications range from a sensor system on a simple microcontroller to a smart cellular phone. The whole set is divided into six categories: automotive & industrial control, consumer devices, office automation, networking, security and telecommunications and includes basic math calculations, quick sort, image recognition, Dijkstra's algorithm, Rijndael, SHA, JPEG encode/decode, MP3 encoder, spelling checker, FFT, CRC32 and many more. The programs are freely available as C source codes with (usually) two data sets: a light-weight but useful embedded application of the benchmark and a large real-world application. Despite their single-thread nature, the benchmarks can be successfully used to evaluate embedded many-core environments, as shown in [79].

The more recent SD-VBS [47] suite includes nine applications from the computer vision domain, namely disparity map, feature tracking, image segmentation, SIFT, SVM, robot localization, face detection, image stitch and texture synthesis. These applications are composed of over 28 computationally intensive kernels such as PCA, correlation, Gaussian filter, QR factorization, affine transforms, etc. The codes are provided in both MATLAB and C. For each benchmark, the data inputs of three different sizes are provided. The SD-VBS suite has been employed in, e.g., [79].

Rodinia (version 3.1) suite contains 23 applications (for example Gaussian elimination, K-means, back propagation, leukocyte tracking, BFS, path finder, stream cluster, similarity scores, LU decomposition) targeting heterogeneous architectures with CPUs and GPUs. The domains of these benchmarks range from data mining to fluid dynamics. The diversity of the benchmarks stems from applying various Berkeley Dwarves, such as Dense Linear Algebra, Dynamic Programming, MapReduce, Un/Structured Grid, etc. The CPU-targeted codes are written in C++ where parallelism and synchronization are defined with the OpenMP pragmas. The GPU implementations of the benchmarks are provided as CUDA codes. Additionally, the codes are available in OpenCL and OpenACC.

In [49], a relatively large set of streaming application benchmarks is available as dataflow programs written in the StreamIt language, described in [80]. These benchmarks include DCT, FFT, DES, FM radio, MP3 decoder, Serpent, JPEG decoder/encoder, MPEG2 decoder/encoder, etc. For several benchmarks, the corresponding C codes are also provided. This suite has been used with many-core architectures in [81] or [82].

Despite the abundance of available benchmark suites as presented above, a large number of research is still carried out using other workloads. Some of them implement classic computer science algorithms, such as matrix multiplication in [53], [54], [55], [56], [57], odd-even sorting [64], [65], [57], parallel quick sort in [53], normalized cross-correlation in [50], [59], [55], etc. The popularity of computer vision many-core applications grows rapidly which is also reflected in the workload selection. The traditional Sobel edge detector filter has been employed in [50], [25], [51], [52], the Canny edge detector in [25], [52] and FAST corner detection in [55], [25], [61], [52]. Various object tracking approaches (including data fusion from multiple sources) have been presented in [65], [68], [69]. Numerous computer vision algorithms derived from OpenCV library [63] have been studied in [55], [59]. Some researchers prefer to work with custom real-world applications. DemoCar, a minimal gasoline engine electronic contol unit (ECU) has been presented and studied in [72]. Control applications of a drone and plane has been used as workloads in [66] and [67], respectively. 3D path planning algorithms applied for avionic collision avoidance systems are analyzed in [70], [71].

Additionally, tasksets for multi/many-cores can be artificially created using various tools, for example Task Graph For Free [83], as it is done in [84], or Synchronous Dataflow 3 [85]. Various automotive ECU can be generated using the AMALTHEA tool platform [86].

## 1.2 Multi-chip Multi/Many-core Systems

The tools and workloads for multi-chip systems are as follows.

### *1.2.1 Tools*

Table 1.3 lists various tools employed to evaluate multi-chip multi/many-core systems, as shown in Figure 1.2.

The toolchains/scripts are limited for the evaluation of multi-chip multi/many-core systems. These have been developed to achieve some specific additional purposes.

SystemC based tools are used in [13], where resource allocation approaches are implemented in a C++ prototype and integrated with a SystemC functional simulator. To simulate real situations, it is considered that the number of jobs arriving during peak times is higher than that of off-peak times. All the jobs arriving over a whole day, i.e., 24-hour period, are considered to sufficiently stress the data center resources, where a job contains a set of dependent tasks.

Analytical models have been used to accelerate the evaluation process [87, 88]. In [87], analytical methods for estimating the total data center energy efficiency are proposed. This allows designers to evaluate energy efficiency of various power management approaches. For different approaches, polynomial efficiency models for cooling and power-conversion equipment are used to construct the system-level energy efficiency model. The models are evaluated for various example cases to show their benefits. In [88], an analytical model supports the design and evaluation of various resource allocation controller parameters.

There is a huge list of simulators to evaluate multi-chip many-core systems and the notable ones are listed in Table 1.3.

CloudSim [89] is a highly generalized and extensible Java based simulation tool for realizing data centers, virtual machines, applications, users, computational resources and policies for managing diverse parts of the system like scheduling and provisioning. The data center contains a set of nodes (server), where each node is comprised of a many-core chip. It also enables modeling and simulation of large scale cloud computing data centers by configuring the number of nodes to a high value. Further, it has support to incorporate user defined policies for allocating hosts to virtual machines (VMs) and include different network topologies.

CloudAnalyst [90] is a GUI based simulator derived from CloudSim. Therefore, it has some extended features and capabilities. It facilitates evaluation according to the geographical distribution of data centers and users. It is regarded as a powerful simulation framework for deploying real-time data centers and monitoring load balancing. The available extensions in this tool range from enabling GUI features, by saving configurations as XML files to exporting live results in the PDF format. The graphical outputs also include tables and charts, in addition to a large amount of statistical data. It also has a high degree of configuration ability as several entities such as data center size, memory, storage and bandwidth can be easily configured to perform a new set of experiments.

*Table 1.3 Tools for multi-chip multi/many-core systems*

| **Category** | **Reference** | **Comments/Remarks** |
|---|---|---|
| Toolchains/scripts | SystemC-based tool [13] | Configurable number of servers and cores |
| | Analytical model [87, 88] | For fast evaluation |
| Simulation | CloudSim [89] | Configurable number of nodes (servers) |
| | CloudAnalyst [90] | Configurable number of nodes (servers) |
| | GreenCloud [91] | Configurable number of nodes (servers) |
| | iCanCloud [92] | Configurable number of nodes (servers) |
| | EMUSIM [93] | Configurable number of nodes (servers) |
| | GroudSim [94] | Configurable number of nodes (servers) |
| | DCSim [95] | Configurable number of nodes (servers) |
| | CloudSched [96] | Configurable number of nodes (servers) |
| | CDOSim [97] | Configurable number of nodes (servers) |
| | TeachCloud [98] | Configurable number of nodes (servers) |
| | SPECI [99] | Configurable number of nodes (servers) |
| | MDCSim [100] | Configurable number of nodes (servers) |
| | Dist-Gem5 [101] | Gem5 extension to distributed systems |
| Commercial tools | CoolSim [102] | Tool for data center managers |
| | Apache Hadoop [103] | For computer clusters |
| | OpenMP [104] | For shared memory multiprocessing |
| | OpenMPI [105] | For multiprocessing |
| | OpenACC [106] | For heterogeneous CPU/GPU platforms |

GreenCloud [91] has been developed as an extension to the NS-2 packet-level network simulator. It provides an environment for simulating energy-aware cloud computing data centers. It offers a detailed fine-grained modeling of the energy consumed by the various equipments used in a data center. Examples of these equipments are servers, network switches and communication links. The data center servers, each containing a many-core chip, are created with a help of a script, where data center size can be configured. This simulator can be used to explore methods leading to minimized electricity consumption.

iCanCloud [92] is based on SIMCAN and developed over the OMNeT++ platform. It was developed with the aim of predicting the trade-offs between cost and performance of a given set of applications executed on specific hardware. Further, it supports flexibility, accuracy, performance and scalability, and thus has been widely used to design, test and analyze various existing and non-existing cloud architectures. It also provides a user-friendly GUI, which is useful for managing preconfigured experiments/systems and generating graphical reports.

EMUSIM [93] stands for Integrated Emulation and Simulation. It integrates emulation (AEF-Automated Emulation Framework) and Simulation (CloudSim) to enable fast and accurate simulations. It is particularly useful when there is limited information regarding the performance of the software under the varied levels of concurrency and parallelism as it accurately models application performance.

GroudSim [94] is designed for scientific applications on grid and cloud environments. It has a rich set of features, e.g, calculation of costs for job executions and background load on resources.

DCSim (Data Center Simulation) [95] is an extensible data center simulator. It facilitates high-end experiments on data center management for the evaluation of data center management policies and algorithms. It also contains a multi-tier application model that allows the simulation of dependencies and has support for feedback.

CloudSched [96] provides different metrics for load-balance, energy efficiency and utilization, etc. It uses the model suggested by Amazon, where physical machine and virtual machine specifications are predefined. It also supports migration algorithms.

CDOSim [97] is a cloud deployment option (CDO) that can simulate the response times, SLA violations and costs of a CDO. It has ability to represent the user's rather than the provider's perspective. It can be used to determine trade-off between costs and performance. It also has features to use workload profiles from production monitoring data.

TeachCloud [98] is made specially for education purposes. For students and scholars, it provides a simple graphical interface to modify a cloud's configuration and perform experiments. It uses CloudSim as the basic design platform and introduces many new enhancements on top of it, e.g., a GUI toolkit, a workload generator, new network models and a reconfiguration interface.

SPECI [99] stands for Simulation Program for Elastic Cloud Infrastructures and allows analysis and exploration of scaling properties of large data centers while taking the given design policies of the middleware into account. Due to its elastic nature, it allows exploration of performance properties of future data centers. Thus, the designer can have insights into the expected performance of data centers when they are designed, but not built.

MDCSim [100] is a scalable simulation platform for in-depth analysis of multi-tier data centers. It captures all the important design specifics of communication paradigm, kernel level scheduling algorithms and the application level interactions among the tiers of the data center.

Dist-Gem5 [101] is a flexible, detailed, and open-source full-system simulation infrastructure. It is an extension of Gem5 to model and simulate distributed computer system using multiple simulation hosts.

Commercial tools are also available to evaluate data centers. Such tools provide evaluation on physical real hardware. Some of these tools are listed in Table 1.3 and described as follows.

CoolSim [102] enables the analysis and design refinement of data centers. It offers several benefits such as an easy to use and quick to learn user environment. By using CoolSim, the best data center in terms of price/performance can be designed.

Apache Hadoop [103] is an open-source software framework usually employed for processing of big data applications/data using the MapReduce programming model. The framework contains a set of clusters built from hardware. The processing part in Apache Hadoop is accomplished by employing MapReduce programming model and there is a storage part, known as Hadoop Distributed File System (HDFS). Hadoop splits files into large blocks and then distributes them across nodes in a cluster to process the data in parallel. Thus, the data is processed fast. Apache Spark is another popular framework providing an interface for programming entire clusters. It allows the developers to efficiently execute the class of applications inappropriate to the Hadoop's MapRecuce model, such as iterative jobs, streaming jobs or interactive analysis [107]. Apache Spark can execute applications up to two order of magnitudes faster than Hadoop due to the reduced number of read/write operations. However, Spark usually requires more RAM memory than Hadoop and is perceived as slightly less secure because of limited authentication options.

OpenMP (Open Multi-Processing) [104] is an application programming interface (API) that supports multi-platform (multi-chip) shared memory multiprocessing. It is employed with programming languages C, C++, and Fortran and is compatible with most hardware platforms and operating systems such as Solaris, AIX, HP-UX, Linux, macOS, and Windows. This API has been implemented in a number of commercial compilers from various vendors (e.g., Intel, IBM), as well as the ones developed by open source communities. It offers a simple and flexible interface to develop parallel applications for platforms of various scales, e.g., desktop computers and supercomputers.

OpenMPI [105] is a Message Passing Interface (MPI) library project combining technologies and resources from several other projects. Similarly to OpenMP, it has been implemented in both commercial and open-source compilers. It has been used by several TOP500 supercomputers of the world. Some notable examples include Roadrunner, the world's fastest supercomputer from June 2008 to November 2009, and K computer, the fastest supercomputer from June 2011 to June 2012.

The fastest supercomputer in 2017, Sunway TaihuLight, has its own implementation of OpenACC [106], another directive-based parallel programming model. OpenACC is aimed at heterogeneous HPC hardware platforms with GPU accelerators. In contrast to OpenMP, where the possible parallelisms and data dependencies has to be expressed in the code explicitly, OpenACC can benefit from the user's guidance, but is capable of performing automatic parallelization of the user-selected regions (kernels) and offloading them to GPUs. Commercial compilers supporting OpenACC are available from CRAY and PGI, but this model is also supported by GCC7 and a number of academic compilers.

### *1.2.2 Workloads*

Table 1.4 lists various workloads/benchmarks used to evaluate multi-chip multi/many-core systems. Multithreaded applications are potential benchmarks to evaluate multi-chip systems. However, some of these benchmarks can be used to evaluate single-chip multi/many-core systems as well [11], e.g., PARSEC [44] and SPLASH-2 [43], and have been mentioned earlier in this chapter. Short descriptions of the additional benchmarks listed in Table 1.4 are as follows.

CloudSuite [108] consists of eight applications that have been selected based on their popularity in today's data centers. These benchmarks represent real-world setups and are based on real-world software stacks.

SPEC Cloud IaaS 2016 benchmark [109] is SPEC's the first benchmark suite to evaluate performance of cloud infrastructures. In addition to academic researchers, the benchmark is targeted for cloud providers, cloud consumers, hardware vendors, virtualization software vendors and application software vendors.

TPC Express Benchmark V (TPCx-V) [110] helps to measure the performance of servers running database workloads. It has features to simulate load variation in cloud data centers with the help of unique elastic workload characteristic. It stresses several resources such as CPU and memory hardware.

The High Performance LINPACK (HPL) benchmark [111] evaluates floating point computing power of a system. For a common task in engineering, they measure how fast a computer system solves a dense system of linear equations. Its latest version is used to evaluate and rank world's most powerful TOP500 supercomputers.

High Performance Conjugate Gradients (HPCG) Benchmark [112] has been proposed to create a new metric for ranking HPC systems. It is a complement to the LINPACK (HPL) benchmark. The benchmark has several basic operations such as sparse matrix-vector multiplication, vector updates and global dot products.

Additionally, the simulators listed in Table 1.3 also contain inbuilt functions to create data center workloads of varying natures. Therefore, they can also be used to stress the data center resources, mainly cores of the chips.

*Table 1.4 Benchmarks for multi-chip multi/many-core systems*

| Category | Reference | Comments/Remarks |
|---|---|---|
| Benchmark sets | PARSEC [44] | Modern problems, not HPC |
| | SPLASH-2 [43] | HPC workloads mainly |
| | CloudSuite [108] | Several software components |
| | SPEC Cloud_IaaS [109] | SPEC's first cloud benchmark |
| | TPCx-V [110] | TPC's data center benchmark |
| | LINPACK [111] | Currently used to rank the TOP500 computing systems |
| | HPCG [112] | Proposed to rank the TOP500 computing systems |

## 1.3 Discussion

The tools and workloads/benchmarks covered in the earlier sections can be used to evaluate systems of various scales, such as embedded, desktop and data centers. Typically, tools and benchmarks for single-chip systems are used to evaluate embedded systems and desktop computers equipped with many-core CPUs or accelerators. However, some of them, especially having multithreaded applications can be used to evaluate multi-chip systems as well.

Since these benchmarks stress the systems in different ways, i.e. some impose high computation load and some high memory load, they need to be appropriately selected to properly evaluate the considered system. It might also be worth trying a mixture of some of the benchmark applications to cover a broad spectrum of workloads across different resources of a computing system.

In addition to the tools and benchmarks covered in this chapter, there are several developments for graphic processing unit (GPU) based multi/many-cores systems, e.g. CUDA [113], OpenCL [114], OpenHMPP [115], etc. However, in this chapter, our focus is on CPU-based multi/many-core systems, so we are not detailing GPU-based systems.

## 1.4 Conclusion

This chapter presents tools and workloads/benchmarks to evaluate systems containing a set of cores. These cores can be present in a single chip or multiple chips, forming single-chip and multi-chip systems, respectively. Depending upon the requirements such as programming model and evaluating cores stressing, appropriate tools and benchmarks can be chosen to evaluate a system under consideration.

## References

[1] Samsung. Samsung Exynos 5422; 2014. Www.samsung.com/exynos/.
[2] Vangal S, Howard J, Ruhl G, et al. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In: Proceedings of IEEE International Solid-State Circuits Conference (ISSCC); 2007. p. 98–589.
[3] Intel. Xeon Phi; 2016. Https://ark.intel.com/products/95828/Intel-Xeon-Phi-Processor-7230F-16GB-1_30-GHz-64-core.
[4] Olofsson A, Trogan R, Raikhman O, et al. A 1024-core 70 GFLOP/W floating point manycore microprocessor. In: Poster on 15th Workshop on High Performance Embedded Computing HPEC2011; 2011. .
[5] TILE-Gx. First 100-core Processor with the New TILE-Gx Family; 2009. Http://www.tilera.com/ (Last visited: 12 February, 2016).
[6] AMD. AMD Opteron 6000 series processors; 2011. Http://www.amd.com/en-us/products/server/opteron/6000 (Last visited: 12 February, 2016).
[7] De Dinechin BD, Van Amstel D, Poulhiès M, et al. Time-critical computing on a single-chip massively parallel processor. In: Proceedings of IEEE Conference on Design, Automation and Test in Europe (DATE); 2014. p. 1–6.
[8] Bohnenstiehl B, Stillmaker A, Pimentel J, et al. A 5.8 pJ/Op 115 Billion Ops/sec, to 1.78 Trillion Ops/sec 32nm 1000 Processor Array. In: IEEE Symposia on VLSI Technology and Circuits; 2016. .
[9] Benini L, De Micheli G. Networks on chips: a new SoC paradigm. Computer. 2002;(1):70–78.
[10] Worm F, Ienne P, Thiran P, et al. An adaptive low-power transmission scheme for on-chip networks. In: Proceedings of IEEE/ACM/IFIP

Conference on Hardware/Software Codesign and System Synthesis (ISSS+CODES); 2002. p. 92–100.

[11] Reddy BK, Singh AK, Biswas D, et al. Inter-cluster Thread-to-core Mapping and DVFS on Heterogeneous Multi-cores. IEEE Transactions on Multi-Scale Computing Systems. 2017;.

[12] Wang X, Singh AK, Li B, et al. Bubble budgeting: throughput optimization for dynamic workloads by exploiting dark cores in many core systems. IEEE Transactions on Computers. 2017;.

[13] Singh AK, Dziurzanski P, Indrusiak LS. Value and Energy Optimizing Dynamic Resource Allocation in Many-core HPC Systems. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom); 2015. p. 180–185.

[14] Smit LT, Smit GJM, Hurink JL, et al. Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture. In: Proceedings of IEEE International Conference on Field-Programmable Technology (FPT); 2004. p. 421–424.

[15] Dick RP, Rhodes DL, Wolf W. TGFF: task graphs for free. In: CODES+ISSS; 1998. p. 97–101.

[16] Stuijk S, Geilen MCW, Basten T. SDF$^3$: SDF For Free. In: Proceedings of IEEE Conference on Application of Concurrency to System Design (ACSD); 2006. p. 276–278.

[17] Xilinx SDK;. Https://www.xilinx.com/products/design-tools/embedded-software/sdk.html (Last visited: 04 November, 2017).

[18] Inc A. Epiphany SDK Reference; 2013.

[19] Brown Deer Technology L. COPRTHR-2 SDK; 2017. Available from: http://www.browndeertechnology.com/coprthr2.htm.

[20] Ross J, Richie D. An OpenSHMEM implementation for the adapteva epiphany coprocessor. In: Workshop on OpenSHMEM and Related Technologies. Springer; 2016. p. 146–159.

[21] Brown N. ePython: An Implementation of Python for the Many-core Epiphany Coprocessor. In: Proceedings of the 6th Workshop on Python for High-Performance and Scientific Computing. PyHPC '16. Piscataway, NJ, USA: IEEE Press; 2016. p. 59–66. Available from: https://doi.org/10.1109/PyHPC.2016.8.

[22] Agathos SN, Papadogiannakis A, Dimakopoulos VV. Targeting the Parallella. In: European Conference on Parallel Processing. Springer; 2015. p. 662–674.

[23] Coduin. Epiphany BSPs documentation; 2017. Available from: http://www.codu.in/ebsp/docs/.

[24] Tomiyama H, Hieda T, Nishiyama N, et al. SMYLE OpenCL: A programming framework for embedded many-core SoCs. In: 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC); 2013. p. 565–567.

[25] Tagliavini G, Haugou G, Marongiu A, et al. A Framework for Optimizing OpenVX Applications Performance on Embedded Manycore Accelerators.

In: Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems. SCOPES '15. New York, NY, USA: ACM; 2015. p. 125–128. Available from: http://doi.acm.org/10.1145/2764967.2776858.

[26] Ying H, Hollstein T, Hofmann K. GSNoC - The comprehensive design platform for 3-dimensional Networks-on-Chip based many core embedded systems. In: 2013 International Conference on High Performance Computing Simulation (HPCS); 2013. p. 217–223.

[27] Lis M, Ren P, Cho MH, et al. Scalable, accurate multicore simulation in the 1000-core era. In: (IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software; 2011. p. 175–185.

[28] Naruko T, Hiraki K. FOLCS: A Lightweight Implementation of a Cycle-accurate NoC Simulator on FPGAs. In: Proceedings of the 3rd International Workshop on Many-core Embedded Systems. MES '15. New York, NY, USA: ACM; 2015. p. 25–32. Available from: http://doi.acm.org/10.1145/2768177.2768182.

[29] Jiang N, Balfour J, Becker DU, et al. A detailed and flexible cycle-accurate network-on-chip simulator. In: Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on. IEEE; 2013. p. 86–96.

[30] Power J, Hestness J, Orr MS, et al. gem5-gpu: A heterogeneous cpu-gpu simulator. IEEE Computer Architecture Letters. 2015;14(1):34–36.

[31] Nicolas A, Sanchez P. Parallel Native-Simulation for Multi-processing Embedded Systems. In: 2015 Euromicro Conference on Digital System Design; 2015. p. 543–546.

[32] Mansour A, Gtze J. An OMNeT++ based Network-on-Chip simulator for embedded systems. In: 2012 IEEE Asia Pacific Conference on Circuits and Systems; 2012. .

[33] Software I. OVP - Open Virtual Platforms; 2017. Available from: http://www.ovpworld.org.

[34] Technologies M. Multicore Development Environmen (MDE); 2017. Available from: http://www.mellanox.com/page/products_dyn?product_family=250.

[35] Kalray. Kalray Software; 2017. Available from: http://www.kalrayinc.com.

[36] eSOL Co. Software Development Kit for Many-core Processors; 2017. Available from: https://www.esol.com/embedded/emcos_sdk.html.

[37] Oliver K. How-To Guide: Creating and Debugging Linux Applications Using Sourcery Codebench for ARM GNU/Linux; 2012. Available from: http://s3.mentor.com/public_documents/whitepaper/resources/mentorpaper_72367.pdf.

[38] Rosa F, Ost L, Raupp T, et al. Fast energy evaluation of embedded applications for many-core systems. In: 2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS); 2014. p. 1–6.

[39] Heckmann R, Ferdinand C. Worst-case execution time prediction by static program analysis. In: In 18th International Parallel and Distributed Processing Symposium (IPDPS 2004). IEEE Computer Society; 2004. p. 26–30.

[40] Technology EE. Erika Enterprise RTOS v3; 2017. Available from: http://www.erika-enterprise.com/.

[41] Poovey JA, Conte TM, Levy M, et al. A Benchmark Characterization of the EEMBC Benchmark Suite. IEEE Micro. 2009 Sept;29(5):18–29.

[42] Halfhill TR. EEMBCS Multibench Arrives; 2008. Available from: http://www.eembc.org/benchmark/pdf/080812_MPRarticle_MultiBench.pdf.

[43] Woo SC, Ohara M, Torrie E, et al. The SPLASH-2 programs: characterization and methodological considerations. In: Proceedings 22nd Annual International Symposium on Computer Architecture; 1995. p. 24–36.

[44] Bienia C, Li K. PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors. In: Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation; 2009. .

[45] Dick R. Embedded system synthesis benchmarks suite (E3S); 2010. Available from: http://ziyang.eecs.umich.edu/dickrp/e3s/.

[46] Guthaus MR, Ringenberg JS, Ernst D, et al. MiBench: A free, commercially representative embedded benchmark suite. In: Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538); 2001. p. 3–14.

[47] Venkata SK, Ahn I, Jeon D, et al. SD-VBS: The San Diego Vision Benchmark Suite. In: 2009 IEEE International Symposium on Workload Characterization (IISWC); 2009. p. 55–64.

[48] Che S, Boyer M, Meng J, et al. Rodinia: A benchmark suite for heterogeneous computing. In: 2009 IEEE International Symposium on Workload Characterization (IISWC); 2009. p. 44–54.

[49] Amarasinghe S, Gordon M, Soule R, et al.. StreamIt benchmarks; 2009. Available from: http://groups.csail.mit.edu/cag/streamit/shtml/benchmarks.shtml.

[50] Tagliavini G, Haugou G, Benini L. Optimizing memory bandwidth in OpenVX graph execution on embedded many-core accelerators. In: Proceedings of the 2014 Conference on Design and Architectures for Signal and Image Processing; 2014. p. 1–8.

[51] Hollis SJ, Ma E, Marculescu R. nOS: A nano-sized distributed operating system for many-core embedded systems. In: 2016 IEEE 34th International Conference on Computer Design (ICCD); 2016. p. 177–184.

[52] Lepley T, Paulin P, Flamand E. A novel compilation approach for image processing graphs on a many-core platform with explicitly managed memory. In: 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES); 2013. p. 1–10.

[53] Gunes V, Givargis T. XGRID: A Scalable Many-Core Embedded Processor. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on

Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems; 2015. p. 1143–1146.

[54] Burgio P, Marongiu A, Valente P, et al. A memory-centric approach to enable timing-predictability within embedded many-core accelerators. In: 2015 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST); 2015. p. 1–8.

[55] Capotondi A, Marongiu A, Benini L. Enabling Scalable and Fine-Grained Nested Parallelism on Embedded Many-cores. In: 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip; 2015. p. 297–304.

[56] Jose W, Neto H, Vestias M. A Many-Core Co-Processor for Embedded Parallel Computing on FPGA. In: 2015 Euromicro Conference on Digital System Design; 2015. p. 539–542.

[57] Lai JY, Huang CT, Hsu TS, et al. Methodology of exploring ESL/RTL many-core platforms for developing embedded parallel applications. In: 2014 27th IEEE International System-on-Chip Conference (SOCC); 2014. p. 286–291.

[58] Quinn MJ. Parallel programming in C with MPI and OpenMP. McGraw-Hill Higher Education; 2004.

[59] Vogel P, Marongiu A, Benini L. Lightweight Virtual Memory Support for Many-core Accelerators in Heterogeneous Embedded SoCs. In: Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis. CODES '15. Piscataway, NJ, USA: IEEE Press; 2015. p. 45–54. Available from: http://dl.acm.org/citation.cfm?id=2830840.2830846.

[60] Magno M, Tombari F, Brunelli D, et al. Multimodal Abandoned/Removed Object Detection for Low Power Video Surveillance Systems. In: Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance. AVSS '09. Washington, DC, USA: IEEE Computer Society; 2009. p. 188–193. Available from: https://doi.org/10.1109/AVSS.2009.72.

[61] Koutras I, Anagnostopoulos I, Bartzas A, et al. Improving Dynamic Memory Allocation on Many-Core Embedded Systems With Distributed Shared Memory. IEEE Embedded Systems Letters. 2016 Sept;8(3):57–60.

[62] Rosten E, Porter R, Drummond T. Faster and Better: A Machine Learning Approach to Corner Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010 Jan;32(1):105–119.

[63] team OpenCV. OpenCV: Open source computer vision. software library; 2017. Available from: http://opencv.org/.

[64] Huang CT, Tasi KC, Lin JS, et al. Application-level embedded communication tracer for many-core systems. In: The 20th Asia and South Pacific Design Automation Conference; 2015. p. 803–808.

[65] Chien HW, Lai JL, Wu CC, et al. Design of a scalable many-core processor for embedded applications. In: The 20th Asia and South Pacific Design Automation Conference; 2015. p. 24–25.

[66] Nemer F, Casse H, Sainrat P, et al. PapaBench: a Free Real-Time Benchmark. In: Mueller F, editor. 6th International Workshop on Worst-Case Execution Time Analysis (WCET'06). vol. 4 of OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; 2006. .

[67] Pagetti C, Saussi D, Gratia R, et al. The ROSACE case study: From Simulink specification to multi/many-core execution. In: 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS); 2014. p. 309–318.

[68] Louise S, Dubrulle P, Goubier T. A Model of Computation for Real-Time Applications on Embedded Manycores. In: 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs; 2014. p. 333–340.

[69] Stan O, Sirdey R, Carlier J, et al. A GRASP for Placement and Routing of Dataflow Process Networks on Many-Core Architectures. In: 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing; 2013. p. 219–226.

[70] Panic M, Quiones E, Zavkov PG, et al. Parallel many-core avionics systems. In: 2014 International Conference on Embedded Software (EMSOFT); 2014. p. 1–10.

[71] Vargas RE, Royuela S, Serrano MA, et al. A lightweight OpenMP4 runtime for embedded systems. In: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC); 2016. p. 43–49.

[72] Dziurzanski P, Singh AK, Indrusiak LS, et al. Benchmarking, System Design and Case-studies for Multi-core based Embedded Automotive Systems. In: 2nd International Workshop on Dynamic Resource Allocation and Management in Embedded, High Performance and Cloud Computing DREAMCloud; 2016. Available from: https://arxiv.org/abs/1601.03708.

[73] Venetis JE, Gao GR. The Modified SPLASH-2 Benchmarks Suite Home Page; 2007. Available from: http://www.capsl.udel.edu/splash/index.html.

[74] Sakalis C, Leonardsson C, Kaxiras S, et al. Splash-3: A properly synchronized benchmark suite for contemporary research. In: 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS); 2016. p. 101–111.

[75] Biswas D, Balagopal V, Shafik R, et al. Machine learning for run-time energy optimisation in many-core systems. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2017; 2017. p. 1588–1592.

[76] Ross JA, Richie DA, Park SJ, et al. Parallel Programming Model for the Epiphany Many-Core Coprocessor Using Threaded MPI. In: Proceedings of the 3rd International Workshop on Many-core Embedded Systems. MES '15. New York, NY, USA: ACM; 2015. p. 41–47. Available from: http://doi.acm.org/10.1145/2768177.2768183.

[77] Nikolakopoulos Y, Papatriantafilou M, Brauer P, et al. Highly Concurrent Stream Synchronization in Many-core Embedded Systems. In: Proceedings of the Third ACM International Workshop on Many-core Embedded Sys-

tems. MES ’16. New York, NY, USA: ACM; 2016. p. 2–9. Available from: http://doi.acm.org/10.1145/2934495.2934496.

[78] Wildermann S, Teich J. Self-Integration for Virtualization of Embedded Many-Core Systems. In: 2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops; 2014. p. 170–177.

[79] Li Z, He S, Wang L. Prediction Based Run-Time Reconfiguration on Many-Core Embedded Systems. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). vol. 2; 2017. p. 140–146.

[80] Thies W, Karczmarek M, Amarasinghe SP. StreamIt: A Language for Streaming Applications. In: Proceedings of the 11th International Conference on Compiler Construction. CC ’02. London, UK, UK: Springer-Verlag; 2002. p. 179–196. Available from: http://dl.acm.org/citation.cfm?id=647478.727935.

[81] Rouxel B, Puaut I. STR2RTS: Refactored StreamIT Benchmarks into Statically Analyzable Parallel Benchmarks for WCET Estimation & Real-Time Scheduling. In: Reineke J, editor. 17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017). vol. 57 of OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; 2017. p. 1:1–1:12. Available from: http://drops.dagstuhl.de/opus/volltexte/2017/7304.

[82] Selva M, Morel L, Marquet K, et al. Extending Dataflow Programs with Throughput Properties. In: Proceedings of the First International Workshop on Many-core Embedded Systems. MES ’13. New York, NY, USA: ACM; 2013. p. 54–57. Available from: http://doi.acm.org/10.1145/2489068.2489077.

[83] Dick RP, Rhodes DL, Wolf W. TGFF: task graphs for free. In: Hardware/Software Codesign, 1998. (CODES/CASHE ’98) Proceedings of the Sixth International Workshop on; 1998. p. 97–101.

[84] de Lima OA, Fresse V, Rousseau F. Evaluation of SNMP-like protocol to manage a NoC emulation platform. In: 2014 International Conference on Field-Programmable Technology (FPT); 2014. p. 199–206.

[85] Stuijk S, Geilen M, Basten T. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs. In: DAC; 2006. p. 899–904.

[86] Wolff C, Krawczyk L, Httger R, et al. AMALTHEA - Tailoring tools to projects in automotive software development. In: 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). vol. 2; 2015. p. 515–520.

[87] Malkamäki T, Ovaska SJ. Analytical model of data center infrastructure efficiency for system level simulations. In: Proceedings of the 8th International Conference on Simulation Tools and Techniques. ICST (Institute for Com-

puter Sciences, Social-Informatics and Telecommunications Engineering); 2015. p. 319–326.

[88] Faraci G, Schembra G. An analytical model for electricity-price-aware resource allocation in virtualized data centers. In: Communications (ICC), 2015 IEEE International Conference on. IEEE; 2015. p. 5839–5845.

[89] Calheiros RN, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience. 2011;41(1):23–50.

[90] Wickremasinghe B, Calheiros RN, Buyya R. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE; 2010. p. 446–452.

[91] Kliazovich D, Bouvry P, Khan SU. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 2012;62(3):1263–1283.

[92] Núñez A, Vázquez Poletti JL, Caminero C, et al. iCanCloud: A flexible and scalable cloud infrastructure simulator. Journal of Grid Computing. 2012;.

[93] Calheiros RN, Netto MA, De Rose CA, et al. EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. Software: Practice and Experience. 2013;43(5):595–612.

[94] Ostermann S, Plankensteiner K, Prodan R, et al. GroudSim: an event-based simulation framework for computational grids and clouds. In: European Conference on Parallel Processing. Springer; 2010. p. 305–313.

[95] Keller G, Tighe M, Lutfiyya H, et al. DCSim: A data centre simulation tool. In: Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE; 2013. p. 1090–1091.

[96] Tian W, Zhao Y, Xu M, et al. A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. IEEE Transactions on Automation Science and Engineering. 2015;12(1):153–161.

[97] Fittkau F, Frey S, Hasselbring W. Cloud user-centric enhancements of the simulator cloudsim to improve cloud deployment option analysis. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2012. p. 200–207.

[98] Jararweh Y, Alshara Z, Jarrah M, et al. Teachcloud: a cloud computing educational toolkit. International Journal of Cloud Computing 1. 2013;2(2-3):237–257.

[99] Sriram I. SPECI, a simulation tool exploring cloud-scale data centres. Cloud Computing. 2009;p. 381–392.

[100] Lim SH, Sharma B, Nam G, et al. MDCSim: A multi-tier data center simulation, platform. In: Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on. IEEE; 2009. p. 1–9.

[101] Mohammad A, Darbaz U, Dozsa G, et al. dist-gem5: Distributed simulation of computer clusters. In: Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on. IEEE; 2017. p. 153–162.
[102] CoolSim. CoolSim for Data Center Managers; 2017. Available from: http://www.coolsimsoftware.com/.
[103] Foundation AS. Apache Hadoop; 2017. Available from: https://www.apache.org/.
[104] OpenMP. Open Multi-Processing; 2017. Available from: http://www.openmp.org.
[105] OpenMPI. Open Source Message Passing Interface; 2017. Available from: https://www.open-mpi.org/.
[106] org OS. The OpenACC Application Programming Interface; 2017. Available from: https://www.openacc.org/sites/default/files/inline-files/OpenACC.2.6.final.pdf.
[107] Zaharia M, Chowdhury M, Franklin MJ, et al. Spark: Cluster computing with working sets. HotCloud. 2010;10(10-10):95.
[108] CloudSuite. Benchmark suite for cloud services; 2017. Available from: http://cloudsuite.ch/.
[109] SPEC. SPEC Cloud_IaaS 2016; 2016. Available from: https://www.spec.org/benchmarks.html.
[110] TPC. TPCx-V; 2017. Available from: http://www.tpc.org/information/benchmarks.asp.
[111] LINPACK. LINPACK Benchmark; 2017. Available from: https://www.top500.org/project/linpack/.
[112] HPCG. High Performance Conjugate Gradients (HPCG) Benchmark; 2017. Available from: http://www.hpcg-benchmark.org/.
[113] CUDA. Compute Unified Device Architecture; 2017. Available from: https://developer.nvidia.com/cuda-zone.
[114] OpenCL. Open Computing Language; 2017. Available from: https://www.khronos.org/opencl/.
[115] OpenHMPP. Open Source Hybrid Multicore Parallel Programming; 2017. Available from: http://www.ithistory.org/resource/openhmpp.