XenoJetBench: An Open Source Hard-Real-Time Multiprocessor Benchmark

Arsalan Shahid, HITEC University Taxila, Pakistan,
Muhammad Yasir Qadri*, Centres of Excellence in Science & Applied Technologies (CESAT), Pakistan,
Nadia Nawaz Qadri, COMSATS Institute of Information and Technology, Wah Cantt, Pakistan,
Jameel Ahmed, HITEC University Taxila, Pakistan

***Abstract*— Standard benchmark tools play an integral part in the design process for performance evaluation of a computer system. A previously proposed tool, JetBench, is an Open Source multiprocessor benchmark that can be used to analyze the performance of a specific target platform. JetBench uses reaction-propulsion engine parameters and thermodynamical equations used in the NASA's EngineSim program, and emulates reaction-propulsion engine performance calculator. This application is platform independent, i.e., target specific libraries, hardware counters and timers are not required. This paper presents an updated and enhanced version of JetBench named as 'XenoJetBench'. XenoJetBench is aimed to provide hard-real-time (HRT) performance evaluation of jet engine's thermodynamic parameters through the integration of a HRT framework on a real-time operating systems kernel; hence reducing the number of missed deadlines. In addition to that XenoJetBench is programmed to provide priority based thread scheduling of thermodynamic calculations. The results show that XenoJetBench gives no missed deadline for single and dual core processors and maximum number of missed deadlines are reduced to 9 for 16 cores.**

I. INTRODUCTION

Benchmarking has become an important part of design process for performance analysis of computer systems and is generally used as a tool for comparative analysis of various architectures [10]. Benchmarks can be classified into two types on the basis of level of performance they measure, i.e., 1) Synthetic benchmarks and 2) System level or Application benchmarks. Synthetic benchmarks are component level benchmarks as they evaluate particular capability of a system such as cache subsystem performance, I/O bandwidth, floating point processing capabilities, etc. Whereas, application benchmarks are system level benchmarks as they are designed to evaluate overall performance of a system for typical workload such as office automation, encryption, etc. Standard benchmarks have been commonly used to evaluate performance of a system and a few processor manufacturers have also developed their propriety benchmarks [13]. However, such benchmarks usually show better performance on the manufacturer's own platform, and may be biased in design to outperform a contender. Therefore, benchmarks developed by independent third parties are considered to be useful for performance comparison among various architectures transparently and impartially. Qadri et al. [16] proposed JetBench, that is a C language based multithreaded/multicore application level benchmark for shared memory architectures, for jet engines thermodynamic calculations. JetBench uses OpenMP API [2] to carry out parallel computations and could be ported to any platform that supports multithreading. The benchmark gives the user, the flexibility to customize workload which can be profile of a real flight including deadlines. The benchmark reports the missed deadlines and time consumed while calculating various data points. These deadlines can give better evaluation of processor performance in terms timing and processing capabilities to achieve a certain task. It must be noted that although JetBench can report timing and missed deadlines; it is not a real-time benchmark intrinsically. Being realistic models of real world applications, real-time (RT)-benchmarks are required to provide hard real-time results; delays being not accepted. This paper extends the state-of-the-art by following contributions:

1) Hard-Real-time (HRT) extension of the JetBench named 'XenoJetBench' with the integration of real-time framework for Linux (Xenomai) [5].
2) Evaluation of XenoJetBench on a Linux RT-kernel, patched with Xenomai, for different number of cores.
3) Improvement in algorithm with priority based thread scheduling of thermodynamic calculations.

XenoJetBench is programmed to provide hard realtime performance for thermodynamic calculations of Jet Engines. For this purpose Xenomai has been considered to provide real-time environment. The improvements were made in JetBench algorithm by incorporating priority based scheduling to reduce the number of missed deadlines. The rest of this paper is categorized in 6 sections. Following section discusses the related work. Section 3 overviews JetBench, illustrates its improved algorithm and describes the enhancements in the form of XenoJetBench. In section 4 results have been presented, including clear-cut analysis of number of missed deadlines. The results are discussed in section 5 and the section 6 concludes the paper.

II. RELATED WORK

Real-time multiprocessing is widely adopted in all fields of computing i.e. from the lowest power consuming-battery operated mobile phones to the high end servers with hundreds

*This work was supported by the National ICT R&D Fund, Pakistan through grant numbered: ICTRDF/TR&D/2012/65
*Corresponding author. Email: yasirqadri@acm.org

of processors on board [17], [8]. Multiprocessor systems pose a challenge in real-time applications since the timing performance over multiple cores may vary due to threading overheads. On the other hand for hard-real-time systems the execution time for all the processors must meet certain deadlines and provide better timing predictability [17]. This section presents the related research in the development of embedded processor benchmarks and real-time frameworks.

Benchmarking has been addressed in several testing domains. Numerous benchmark suites are currently being used to measure the performance of microprocessors such as **S**tandard **P**erformance **E**valuation **C**orporation (SPEC) Benchmark suite [18], Whetstone [20], Dhrystone [19], SPLASH-2 and NAS parallel benchmarks [11]. But all of these benchmarks are non-embedded benchmarks. Moreover, embedded systems application domain is growing fast in microprocessor industry; increasing the need of embedded benchmarks.

One of the most commonly used benchmark suite that is specially designed for embedded systems is "Embedded Microprocessor Benchmark Consortium (EEMBC) benchmark suite". It comprises of applications and algorithms targeting telecom, networking, industrial products and automotives. AutoBench [6], an EEMBC benchmark allow users to predict the performance of 'single-core' microprocessor architecture and microcontrollers in automotive, industrial, and general purpose applications. This benchmark suite is composed up of real time applications including Tooth-Spark and Angle-Time Conversion [22]. A recent addition of Multibench [4] extends the scope to analyze multicore architectures, memory bottlenecks, and OS scheduling support.

PARSEC [1] is an application benchmark suite for System on Chip (SoC) architectures that focus on rising applications that include computer vision, animation, data mining, data storage and financial analysis. It is composed of multi-threaded applications. Another benchmark suite, SPLASH-2 [21] is especially aimed towards the High-Performance Computing (HPC) domain. It is composed of more than 12 benchmarks including various applications and kernels. However, none of the above discussed benchmarks provide real-time performance evaluation for a multiprocessor system.

Fadia et al. presented PapaBench [15], a real-time, single core, embedded processor benchmark for experimental works in Worst Case Execution Time (WCET) computation and for scheduling analysis. MiBench [9] is an embedded systems benchmark suite that is also a single core and non RT-implementation of various applications in different areas, i.e., automotive, workplace, network security, and telecommunication. But these benchmarks do not have multi-threaded implementation.

HieTao et al. presented MPbenchmark [12], an application benchmark for evaluating multicore processors. MPbenchmark is a revised version of JetBench [16], implemented in several programming languages. The languages have been chosen accordingly to support the shared memory computation model. But it was neither evaluated on real-time operating system (RTOS) nor implemented on a HRT domain. Moreover, MPbenchmark does not focus on percentage of real-time used during execution of benchmark which is an important factor to be examined.

POSIX [3], Message Passing Interface (MPI) [7], and OpenMP, are some of the standard APIs that facilitate in the development of multithreaded applications. Multicore platforms have also been used for real-time applications to achieve greater throughput with decreased power consumption. Gerum et al. presented Xenomai [5], A standard API that provides hard-real-time framework in Linux.

The state of the art in the area of embedded processor benchmarking still requires a specific multicore hard-real-time benchmark suite capable to test performance of multicore systems. Thus, an enhanced version of JetBench (XenoJetBench) is presented to evaluate efficiency of microprocessor embedded architecture for parallel processing in hard-real-time.

The following section presents an overview of the previously proposed JetBench application and updated JetBench algorithm; following its hard-real-time implementation, i.e., XenoJetBench.

III. JETBENCH

JetBench is an open source multiprocessor benchmark. The thermodynamic calculations in this benchmark are inspired by a sequential application named 'NASA EngineSim' [14]. There are a few unique forms of EngineSim which require distinctive levels of involvement with the package, knowledge of jet engines, and computing technologies. The JetBench benchmarking tool involves calculation of thermodynamic parameters for three different jet engines designs, i.e., 1) "TurboJet", 2) "Turbojet with afterburner", and 3) "Turbofan engine"; shown in Figure 1 (a, b, c).

These applications are programmed to be platform independent, i.e., target specific libraries and hardware counters and timers are not used at all. There are four input parameters that are defined in JetBench, i.e., altitude, throttle, speed, and deadline time that can be a part of some real flight profile, and reports back the time consumed for various thermodynamic calculations (see Figure 2).

In Figure 2, the Mach number 'M' is the ratio of speed 'V_o' and speed of sound 'a_o', '$M = V_o/a_o$' and pressure and the temperature depends on the altitude. In order to obtain the theoretical thrust for a turbojet engine, the general equation is given in the form of specific thrust as '$F_s = F/m = (1 + m_f) * V_e - V_o$', where '$F_s$' is the specific thrust, 'F' is the net thrust, 'm' is the air flow rate through the engine, 'm_f' is the fuel-air ratio, and 'V' is the exit or the free stream velocity. The fuel mass flow 'm_f' rate is related to the total engine air flow rate 'ma' by the fuel to air ratio 'f'. Engineers use this efficiency factor '$TSFC$' to characterize an engine's fuel efficiency. It is very important to compare the amount of thrust an engine generates and the amount of fuel used to generate that thrust. '$TSFC$' is calculated by using the equation '$TSFC = mf/F$', where 'F' is the net thrust.

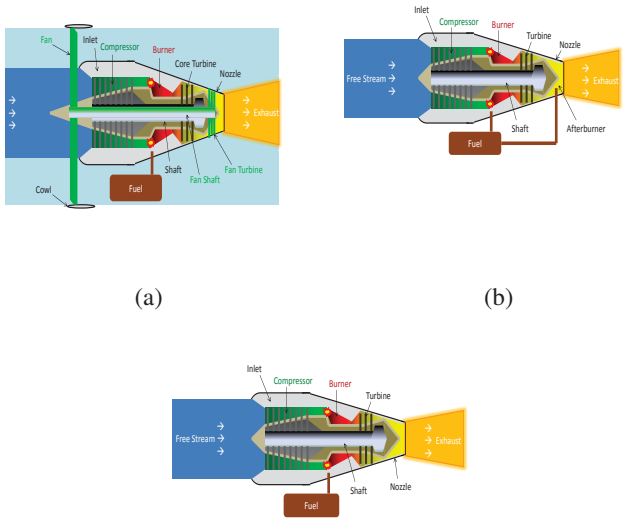

Fig. 1. Jet Models used in JetBench (a) Turbojet (b) Jet with Afterburner (c) Turbofan

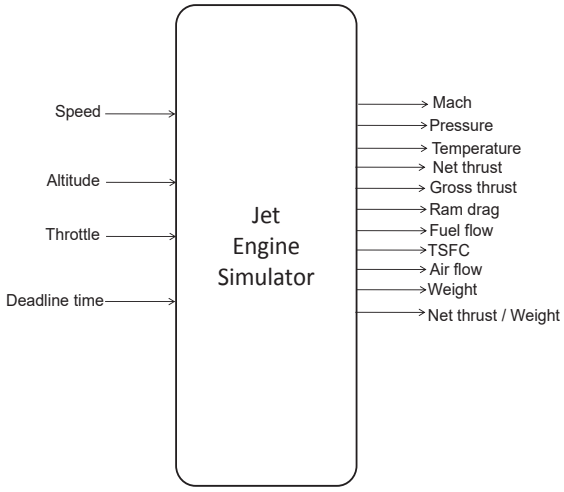

Fig. 2. JetBench Application I/O Parameters

Being an application benchmark, JetBench is an actual representation of the real workload unlike synthetic benchmarks. However, there exist some deviations in order to provide flexibility across various platforms. One such feature is flexibility to the user to define timeline for execution. The application needs to get executed in a given time period; unrealistic deadlines can bring about the benchmark to perform inadequately on majority of low end systems. Also, Jetbench is programmed to cover a limited number of typical thermodynamic calculations used in jet engines. As a result of the limited workload of the calculations, it might appear to be sufficiently small for high-end multicore systems that their real performance may not be accounted-for correctly, in contrast to a low-end multicore platform.

The JetBench benchmarking application overviews the real-time performance of the system and also discovers an optimum number of threads for obtaining desirable performance. The tool is mainly composed up of operations that are arithmetic logic unit (ALU) centric e.g. addition, integer/double multiplication, and division for the computation of square roots, exponents, and calculations such as degree-to-radian conversion and value of pi. All these operations are comprised of actual thermodynamic equations and operations used by the engine control unit (ECU) of a jet engine.

A. Hard-Real-time implementation of Jetbench (XenoJetBench)

This sections presents a modified, and enhanced hard-real-time version of JetBench hereby referred to as XenoJetBench. It uses two advanced application programming interfaces (APIs) i.e., OpenMP along with Xenomai (Real-time framework for Linux implementation). OpenMP supports multi-platform shared memory multiprocessing programming in C. Xenomai provides hard real-time support to user space applications and can be integrated in to Linux Kernel. Xenomai is based on an abstract RTOS core, capable of building any real-time interface which supports a set of generic RTOS services. Common features that are available in many traditional RTOS, particularly related to thread synchronization and scheduling, are also supported in Xenomai.

The XenoJetBench is programmed to provide real-time, priority based thread scheduling of thermodynamic tasks. The implementation of XenoJetBench is described by pseudo-code (see Figure 3). First of all Xenomai is initialized along with default parameters; avoiding memory swap by variables and initializing real time printing 'rt-print' functions. After that user is provided the option to select engine out of turbojet, afterburner and turbofan for which thermodynamic calculations are to be made. Then the number of threads for which application is needed to run and analyze are given i.e., during this step, the thermodynamic tasks are assigned their priority of execution. Subsequently the parallel section starts and benchmark calculates thermodynamic parameters. The parallel computations include the calculations of pie and various other calculations. The benchmark's input parameters i.e., speed, altitude, throttle, and deadline are read line by line and prioritized tasks for thermodynamic calculations start. The results of calculations are printed in the parallel section or we call those results as local results. Finally the parallel section ends and global results including the benchmark execution time and percentage of real-time used are displayed.

Figure 4 illustrates the structure of XenoJetBench using flowchart. After initialization of default parameters; input parameters are given to the benchmark from a separate text file. XenoJetBench reads the input parameters line by line

1: **Input:**
2: Data file
- (Speed, Altitude, Throttle, Deadline)

4: **Initialize:**
5: Init Xenomai
- Avoid memory swapping
- Auto init of rt_print buffers

8: Default Parameters
9: Engine Selection
- Turbojet
- Afterburner
- Turbofan

13: Set Number of Threads
14: **Parallel section:**
15: Create Tasks
16: Pi Calculation
17: Read input data points
18: Start tasks
19: Priority based scheduling of tasks
20: **Calculate:**
- Environment variables
- Thermo parameters
- Engine geometry
- Engine Performance

25: Print local results
26: **Results:**
27: Print global results
28: Delete tasks
29: **End**

Fig. 3. Pseudo-code of XenoJetBench

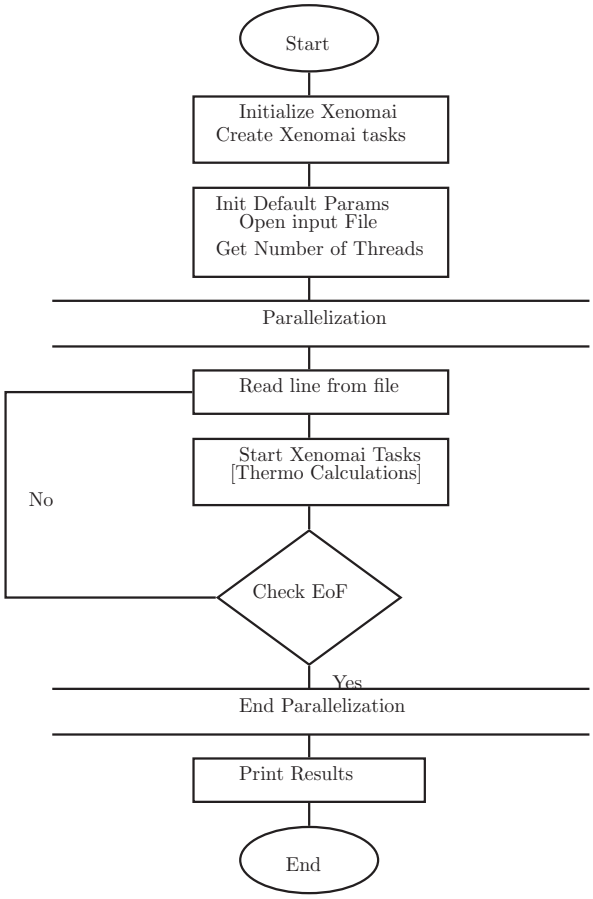

Fig. 4. Flowchart of XenoJetBench

and performs the thermodynamic calculations in the parallel section and displays the results (output parameters). At the end of the input file, benchmark comes out of the parallel section and global results are displayed which are total execution time, number of missed deadlines and percentage of real-time used.

IV. RESULTS

To analyze the performance of XenoJetbench, several experiments have been performed. This section shows the results of those experiments for different number of cores. By decreasing the deadline time, given as an input to the benchmark, number of missed deadlines were observed for various deadline definitions.

Figure 5 shows the missed deadlines with respect to number of cores using a turbojet engine, it can be observed that as the number of cores increase the missed deadlines also increase for both benchmarks however number of missed deadlines for XenoJetBench are lower than JetBench i.e. missed deadlines for XenoJetBench are almost same for 8, 12 and 16 cores that is at around 8 missed deadlines whereas for JetBench more than 35 deadlines were missed. This is because of hard real-time implementation, priority scheduling and evaluation of XenoJetBench on RTOS. Similarly, Figure 6 and 7 shows the number of missed deadlines by using Afterburner and Turbofan engines. Maximum number of missed deadlines were observed for turbofan, i.e., 45. With the increase in number of cores, the execution time of XenoJetBench was also observed as shown in Figure 8. It can be seen that the execution time decreases with the increase in number of cores and benchmark's execution time is minimum for 16 cores. Moreover, percentage of real-time used for execution is also observed as an effect of increase or decrease of deadline time and the number of cores (see Figure 9). It is observed that as the number of cores increases the XenoJetBench executes within close range of deadline time. In other words the percentage of real time being used is increasing with the increase in number of cores.

V. DISCUSSIONS

XenoJetbench has been tested on Linux kernel 3.5.7, on an Intel Core I7, with 16GB RAM, running at 3.2 GHz. Xenomai was patched with the updated kernel for real-time performance evaluation of benchmark. Table 1 shows the statistics collected from Linux *perf* utility. The statistics include i.e., number of cycles, instructions, cache memory

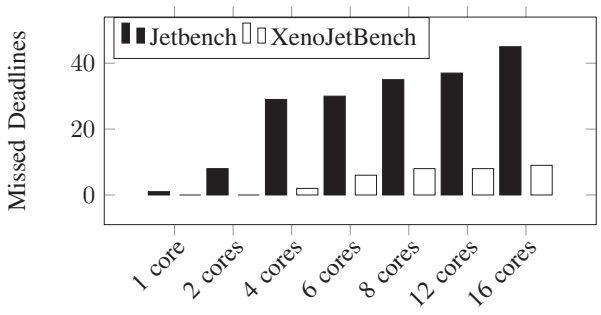

Fig. 5. Number of missed deadlines for TurboJet

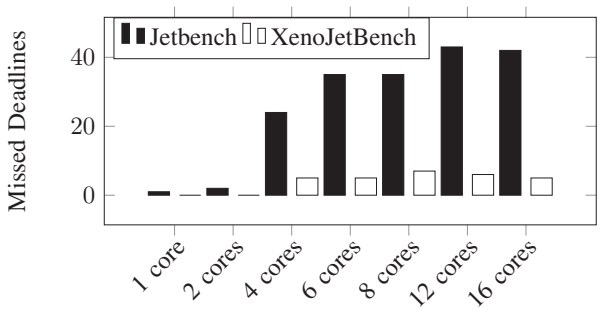

Fig. 6. Number of missed deadlines for Afterburner

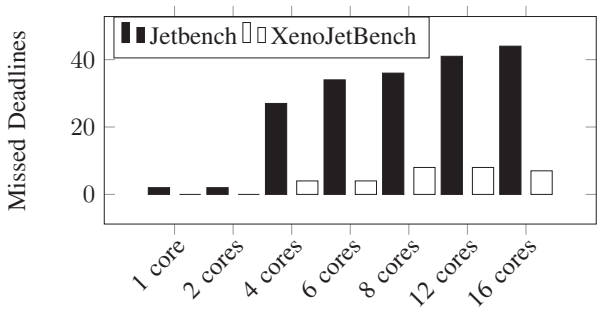

Fig. 7. Number of missed deadlines for Turbofan

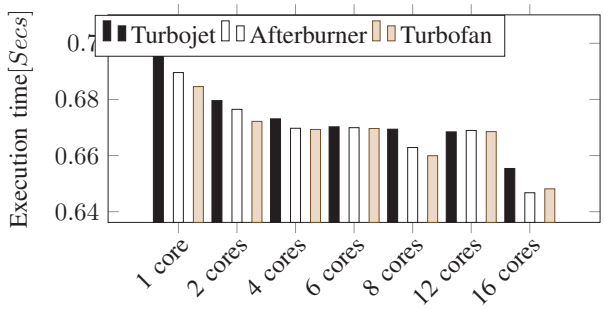

Fig. 8. Benchmark Execution time

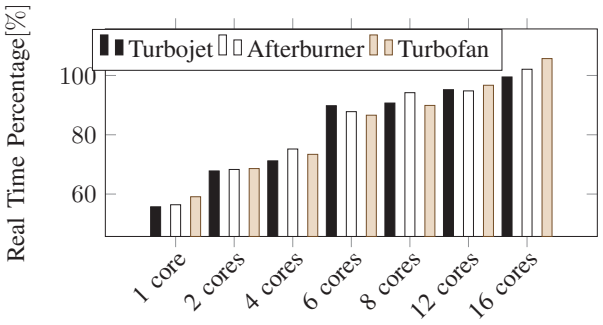

Fig. 9. Percentage of Real Time used

TABLE I
PERFORMANCE MONITORING OF TARGETED SYSTEM

Parameters	Value
Time [Secs]	5
Cycles	5,968,534,86
Instructions	6,851,750,31
L1 Dcache Loads	19,254,901
L1 Dcache Load Misses	1,403,68
L1-Dcache Store Misses	1,392,827
L1 Icache Loads	17,134,510
L1 Icache Load Misses	1,512,70
Page-Faults	17,213,142
LLC-Loads	5,671,691
LLC-Stores	3,344,764
dTLB Load Misses	1,342,836
dTLB Store Misses	2,067,418
iTLB-loads	8,427,937
iTLB-load-misses	1,652,904

miss during total benchmark's execution time. Following are the observations made after experiments with XenoJetBench.

1) We examined that the number of missed deadlines for XenoJetBench were significantly reduced. For single and dual cores, XenoJetBench resulted in no missed deadline. The maximum number of missed deadlines found for 16 cores were 45 in case of JetBench and for XenoJetBench these deadlines were reduced to 9 (see Figure 5, 6 & 7).
2) With the increase in number of cores the benchmark's execution time was reduced. Execution time was maximum for single core in case of Turbojet whereas minimum for 16 cores in case of Afterburner (see Figure 8).
3) The percentage of real-time used was increased with the increase in number of cores (see Figure 9).

VI. CONCLUSIONS

Realtime Benchmarks are an essential tool to analyze hard-real-time performance of a system where delays are not acceptable. Jet Engine is one of those applications in which we need hard real-time results. This paper presents 'XenoJet-Bench', a hard real-time implementation of previously proposed JetBench with the integration of Xenomai i.e. a real-time framework. Furthermore, XenoJetBench provides priority based scheduling of thermodynamic calculation. With

the evaluation of XenoJetBench on RTOS kernel significant reduction in number of missed deadlines is observed. For single and dual cores XenoJetBench resulted in no missed deadline. The maximum number of deadlines observed were 9 for 16 cores. In the Future work, the benchmark will be evaluated on various hardware platforms.

JetBench is available online at: https://sourceforge.net/projects/jetbench/

XenoJetBench is available at: https://sourceforge.net/projects/xenojetbench.jetbench.p/

Note: In order to run benchmark, users need to extract the package after installing Xenomai [Hard-Real-time programming framework for linux] on their linux kernel. The benchmark compilation can be done using *make* command by keeping default parameters, i.e., number of threads=2. The execution of benchmark is to be done using *run.sh* file.

REFERENCES

[1] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[2] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[3] Ulrich Drepper and Ingo Molnar. The native posix thread library for linux. *White Paper, Red Hat Inc*, 2003.

[4] Shay Gal-On and Markus Levy. Measuring multicore performance. *Computer*, 41(11):99–102, 2008.

[5] Philippe Gerum. Xenomai-implementing a rtos emulation framework on gnu/linux. *White Paper, Xenomai*, 2004.

[6] M Ginsberg. Autobench: A 21st century vision for an automotive computing benchmark suite. *High Performance Computing in Automotive Design, Engineering, and Manufacturing, Cray Research, Inc., Eagan, MN*, pages 67–76, 1997.

[7] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

[8] Halûk Gümüskaya and Bülent Orencik. A parallel pipelined computer architecture for digital signal processing. 1998.

[9] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst, Todd M Austin, Trevor Mudge, and Richard B Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14. IEEE, 2001.

[10] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012.

[11] Haoqiang Jin, Michael Frumkin, and Jerry Yan. The openmp implementation of nas parallel benchmarks and its performance. Technical report, Technical Report NAS-99-011, NASA Ames Research Center, 1999.

[12] HaiTao Mei and Andy Wellings. Using jetbench to evaluate the efficiency of multiprocessor support for parallel processing. In *Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems*, page 47. ACM, 2014.

[13] Greg Morton and K Venkat. Msp430 competitive benchmarking. *Texas Instruments*, 2005.

[14] NASA. Nasa enginesim. https://www.grc.nasa.gov/www/k-12/Enginesim/index.htm.

[15] Fadia Nemer, Hugues Cassé, Pascal Sainrat, Jean-Paul Bahsoun, and Marianne De Michiel. Papabench: a free real-time benchmark. In *OASIcs-OpenAccess Series in Informatics*, volume 4. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

[16] Muhammad Yasir Qadri, Dorian Matichard, and Klaus D McDonald Maier. Jetbench: an open source real-time multiprocessor benchmark. In *Architecture of Computing Systems-ARCS 2010*, pages 211–221. Springer, 2010.

[17] Brinkley Sprunt, Lui Sha, and John Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1(1):27–60, 1989.

[18] Joseph Uniejewski. Spec benchmark suite: designed for today's advanced systems. *SPEC Newsletter*, 1(1):1, 1989.

[19] Reinhold P Weicker. Dhrystone: a synthetic systems programming benchmark. *Communications of the ACM*, 27(10):1013–1030, 1984.

[20] Reinhold P Weicker. An overview of common benchmarks. *Computer*, 23(12):65–75, 1990.

[21] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: Characterization and methodological considerations. In *ACM SIGARCH Computer Architecture News*, volume 23, pages 24–36. ACM, 1995.

[22] Lotfi Asker Zadeh, Bo Yuan, and George J Klir. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh*. World Scientific Publishing Co., Inc., 1996.