Strategies for Adapting a Higher Education Traditional Discipline on High-Performance Computer Architecture to Remote Learning

E. C. Pedrino

Universidade Federal de São Carlos,
São Carlos, SP, Brazil
e-mail: emerson@ufscar.br

M. C. Nicoletti

Universidade Federal de São Carlos &
UNIFACCAMP, SP, Brazil
e-mail: carmo@ufscar.br

Abstract— This work reports the detailed process of adapting an undergraduate discipline about High-Performance Computer Architecture, traditionally face-to-face taught, to be taught in a remote mode. The discipline is part of the curricula of two Computer Science undergraduate courses. The adaptation was carried out to comply with precaution measures to avoid the dissemination of the covid-19 disease, caused by the new coronavirus. The article describes the reorganizational and methodological processes to adapt a face-to-face discipline into a remote mode, discusses ways to implement the guidelines established by the Department of Computing and presents the new structure adopted for theoretical and practical classes, as well as the results of the assessment of the new teaching approach adopted for dealing with the pandemic. The process of adapting a face-to-face discipline to a remote mode was quite successful, considering the short period of time the adaptation needed to be implemented, as well as the many decisions involved in the processes detailed in the paper. The teaching-learning process throughout the remote course was considered satisfactory by the students, with a 95% confidence interval.

***Index Terms* – Teaching high-performance computer architecture, Curriculum guidelines, Teaching methodology, Remote teaching.**

I. INTRODUCTION

The Department of Computing (DC) of the Federal University of S. Carlos (UFSCar), campus S. Carlos, offers undergraduate courses that cover knowledge in different areas of computing, namely: Computer Theory [1], Computer Mathematics [2,3], Methodology and Computer Techniques [4], and Computer Systems[5]. The subjects cover theoretical and practical aspects of Computer Science (CS) and are taught in classrooms as well as using several DC´s laboratories and, eventually, other spaces at the UFSCar.

The DC offers two undergraduate courses, namely the Bachelor in Computer Science (BCS) and the Bachelor in Computer Engineering (BCE). Students in both courses have some flexibility to tailor some modules of their course to their talents and aptitudes. Both degrees can be considered suitable for the many available employment opportunities in CS-related areas, and both are among the most sought courses offered by UFSCar.).

Most disciplines taught by the academic staff of the DC are for the BCS and BCE courses. However, the DC also offers CS-related courses for students pursuing degrees in other areas such as the many specialties of Engineering, Statistics and Mathematics.

For the Bachelor's Degree in Computer Science, the DC offers basic subjects as well as those requiring specific training. The four-year course is divided into seven nuclei of knowledge: (1) Fundamentals of Mathematics and Statistics, (2) Fundamentals of Computer Science, (3) Algorithms and Programming, (4) Methodology and Computer Techniques, (5) Computer Systems, (6) Multidisciplinary and Humanistic Training and (7) Orientations. The BCS Pedagogical Project presents, among other information, the course curriculum, its detailed contents and the course bibliography associated.

For the Bachelor´s Degree in Computer Engineering the DC also offers basic subjects and those with specific knowledge. The BCE offers an academic program that integrates the field of Computer Engineering and Computer Science. The course contents are distributed in five basic and technological cores: (1) Fundamentals of Mathematics and Statistics, (2) Algorithms and Programming, (3) Computer Architectures, (4) Methodology and Computer Techniques, and (5) Humanities. The BCE Pedagogical Project presents, among other information, the course curriculum, its detailed contents and the associated bibliography.

Since 2007 the DC has also been offering distance learning disciplines to students in the Bachelor of Information Systems (BIS) course, as part of the Open University of Brazil - UFSCar project [6]. The Bachelor of Information Systems has computing as a medium activity, as it brings together technologies in the areas of Computing, Information Systems and Administration.

Aiming at constant updating of UFSCar's computer-focused courses and the better preparation/formation of students to face future challenges, the DC is preparing Study Certificates, which will give students the opportunity to obtain qualification differentials throughout their undergraduate studies certifying both, specialization and the deepening of knowledge and experience, in subareas of CS.

The goal of this article is to present and describe in detail the adaptation process conducted by the lecturer and part of the academic staff for converting a face-to-face discipline on high-performance computer architecture, into a remote discipline, in a short period of time. The remainder of this paper is organized into five sections where Section II presents the main characteristics of the two CS-related undergraduate courses where the subject of Computer Architecture is part of the curriculum and approached with different focuses in three distinctive disciplines throughout the period of the courses. Section III presents an academic literature review of previous works involving the many adopted approaches for teaching undergraduate disciplines on Computer Organization and Architecture (COA), which usually is part of CS-related curriculum.

Section IV has its focus on the teaching of a particular branch of computer organization and architectures, which is related to the development and implementation of techniques that promote organization and architectures qualified as high performance. The section has its emphasis on a particular discipline, High-Performance Architectures (HPA), which due to the pandemic, had to be reorganized, restructured and adapted to be taught in remote mode.

Section V presents and analyses the results of a conducted survey that happened at the end of the HPA discipline, where students were asked to answer a group of questions about the contents of the course, the theoretical and practical aspects of it and, particularly, the way the discipline was taught i.e., in remote mode. Section VI concludes the presentation of the work done, by reviewing the major points of the several processes conducted aiming at adapting, at relatively short notice, a face-to-face taught discipline into a remote taught mode. The section also presents comments about the work done and about the future perspectives for the adapted discipline.

II. CONTEXTUALIZING TWO UNDERGRADUATE COURSES

Taking into account the 8-semester curriculum (workload of 3,240 hours) of the BCS (Bachelor in Computer Science) course, three disciplines related to Computer Architecture are offered in the 2nd, 6th and 7th semester and named respectively Computer Organization and Architecture I (ArcI), Computer Organization and Architecture II (ArcII) and High-Performance Architecture (ArcIII), where ArcI is mandatory and both, ArcII and ArcIII, are elective disciplines.

In the first semester of BCS the following disciplines are mandatory: (1) Differential and Integral Calculus I (2) Introduction to Algorithmic Thinking, (3) Algorithm Development and Programming, (4) Digital Logic. Discipline (1) is offered by the Mathematics Department and the other three by the Department of Computing. So BCS students are introduced to the main concepts related to computer organization and architectures only after they have been taught and have learned concepts related to disciplines (1), (2), (3) and (4). Later in the course students have the opportunity to attend to ArcII and ArcIII, as elective disciplines. For BCS students to enroll in the two elective disciplines, the ArcI is a prerequisite.

Taking into account the 10-semester curriculum (workload of 3,660 hours) of a BCE (Bachelor in Computer Engineering), three disciplines related to Computer Architecture are offered in the 3rd, 4th and 7th semesters and named respectively Computer Organization and Architecture I (ArcI), Computer Organization and Architecture II (ArcII) and High-Performance Architecture (ArcIII), where ArcI, ArcII and ArcIII are mandatory.

In the first semester of the BCE all the following disciplines are mandatory: (1) Differential and Integral Calculus I (2) Analytic Geometry (3) Introduction to Algorithmic Thinking, (4) Algorithm Development and Programming, (5) Digital Logic. Disciplines (1) and (2) are offered by the Mathematics Department and the other three by the DC. In the second semester of BCE all the following seven disciplines are mandatory: (1) Calculus II (2) Linear Algebra I (3) Physics I (4) Experimental Physics A (5) Algorithms and Data Structures I (6) Object-Oriented Programming and (7) Digital Systems. Disciplines (1) and (2) are offered by the Mathematics Department, disciplines (3) and (4) are offered by the Physics Department and the remaining four are offered by the DC. Five among the seven disciplines require prerequisites from the first semester. Discipline (1) requires Differential and Integral Calculus I as prerequisite discipline (2) requires Analytic Geometry as prerequisite and discipline (5) and (6) require Algorithm Development and Programming.

So, BCE students are introduced to the main concepts related to computer architectures only after they have been taught and became familiar with concepts related to disciplines (1), (2), (3), (4) from the first semester and (1), (2), (3), (4), (5), (6) and (7) from the second semester. They still have ahead of them two computer architecture-related mandatory disciplines, the Computer Organization and Architecture II (ArcII) in the 4th semester, which have, as prerequisite, Computer Organization and Architecture I from the 3rd. semester and High-Performance Architectures (HPA), in the 7th semester, which also have Computer Organization and Architecture I as prerequisite.

III. LITERATURE REVIEW FOCUSING ON THE TEACHING OF COMPUTER ORGANIZATION AND ARCHITECTURE

Disciplines usually named as Computer Organization and Architecture (COA), or similar names, are an intrinsic part of Computer Science or Computer Engineering university courses. The teaching of these disciplines is usually planed in such a way that students can acquire the many theoretical Mathematics and CS-related concepts, as well as practical experiences related to those concepts and techniques.

This section presents a literature review on published academic works which have their main focus on the teaching of undergraduate disciplines related to computer organization and architectures and, also, the many variables involved in the process. The motivation for this section was to collect and briefly review different views on the subject of teaching academic material related to computer architecture, aiming at presenting a panorama of the diversified ways disciplines with focus on the subject have been implemented, and their different emphasis on the many concepts involved considering both, the theoretical and practical aspects of such disciplines as well as the use of software tools for supporting the learning process.

Taking into consideration all the works presented in the many articles that have been collected, read and considered for composing this section, it can be stated that a common view shared by most authors is that (1) concepts and formalisms related to Computer Organization and Architecture are not a trivial knowledge to be easily/quickly absorbed and mastered

(2) a solid background in Mathematics helps the mastering of most concepts used in these disciplines and (3) software tools can help promoting and ease the understanding of many theoretical concepts.

The work in [7] describes a 150-hour course on Advanced Computer Structure, part of the degree in Computer Engineering at the University of Murcia. The course was divided into theoretical and practical classes, where students were evaluated throughout the whole course period via individual work. The paper has its focus mainly on the analysis of the teaching methodology employed, the distribution of different subjects over the 150 hours and the synchronization between theoretical and practical classes.

The contents and planning of the course aimed at promoting and developing students´ competencies in: (1) analysis of the functioning of a simple computer; (2) ability to write simple programs using a low-level programming language (assembly language) (3) linking programs written in assembly language to high-level language programs and translating high level language code into assembly language; (4) acquire the concept of the memory hierarchy, different types of storage and basic principles related to the input/output system; (5) ability to evaluate and compare different versions of a small program, while running in different computer configurations.

Discipline units related to CS theoretical aspects emphasized: computer architecture performance analysis, pipeline processors, advanced pipelining and branch prediction, static allocation of instructions and high-performance memory system. The practical side of the discipline focused on: measuring and evaluating architecture performance, pipeline data path and control, allocation instructions in pipeline processors and evaluation of the cache memory. The contents of the discipline were assessed divided into two parts: theoretical and practical, graded independently, where the grade associated with the theoretical aspects was weighted by 0.7, and that of the practical aspects by 0.3, when assigning students' final grades. According to the author, the employed methodology and the scheduling and synchronization of the theoretical and practical units were the key aspect for achieving good learning results.

As pointed out in [8], the prestige enjoyed by computer architecture related topics as one of the main research areas in CS and a core topic in computing curricula, has been declining over the last years, perhaps due to a shift of interest to new CS areas. Considering that computer architecture is a very important topic to be taught in CS degrees, the author proposes a way of improving the interest in learning about computer architectures, by merging it with ubiquitous computing, an area that has been highly promoted by the CS research community. The author´s work-in-progress proposal combines parts of the existing computer architecture curriculum with the structure, organization and design of ubiquitous systems in an attempt to promote the visibility of computer architectures as a relevant issue in CS.

The work in [9] describes the use of the CTPracticals [10], a module of the Moodle (*Modular Object-Oriented Dynamic Learning Environment*) system [6,7] designed to support the teaching of practical contents related to basic computer architecture disciplines. The module provides an automatic verification engine capable to automatically process VHDL designs as they are submitted. Students can then follow the progress of their work by accessing the results of the automatic assessment, and lecturers can have a continuous global view on the developing status of their students. The paper reports on the authors´ experiences when using the CTPracticals module for teaching the contents of a Computer Technology laboratory. The CTPracticals was installed on a server having the Moodle 1.9.

The work described in [13] relates to a computer architecture course and has, as the main focus, the development, by students, of a pipelined CPU design project with a field-programmable gate array (FPGA) system. The intent of the project assignment was to teach students, via practical approaches, about several components involved in CPU design projects. Authors point out that the work required the implementation of a real CPU instead of using simulators or ready-made complete CPU models. The prerequisite disciplines required for taking the computer architecture course were C-programming and Computer Logic Design, the latter covering digital logic design and Verilog-HDL. Authors agree that students who have fulfilled the prerequisites were prepared to design any digital logic, including CPU, if properly guided.

Authors in [14] agree that the use of simulators can be a useful resource for helping the acquisition of concepts by students. Specifically, the article presents a teaching simulator, SICOME 2.0, which was used in practices related to the task of learning computer architecture. The simulator allows an interactive simulation on simple computer architecture. According to the authors the experiences with the simulator were very satisfactory and the results obtained showed that it helped to improve the students' comprehension of the subject.

The work described in [15] has its focus on the Computer Organization and Architecture discipline, part of the curricula of both undergraduate courses: Computer Engineering (CE) and Computer Science (CS) offered by the Faculty of CS of the Brawijaya University. According to the authors, for both courses it is important that students, besides learning the theoretical aspects of the course, also conduct practical experiments related to organization and architecture of computers and learn from them. Authors emphasize that the practical side of the teaching process can be approached using simulators, considering that the manufacture of CPUs, RAM memories and other devices is usually expensive and takes time. Currently there are many freely available simulators that can be downloaded from the Web and used straight away. However, the authors alert about choosing the proper simulator to use - the choice should be done taking into account the theoretical development of the discipline and the set of planned skills to be developed.

An important issue related to teaching Computer Organization and Architecture pointed out by the authors in [16] relates to reaching a balance between the acquisition of the necessary theoretical concepts and practical experiences. According to the authors the visualization of diverse computer architectures with the help of simulators can enhance students´ learning process. In their article the authors carried out an assessment by reviewing 12 simulators on how they fulfilled two different sets of requirements. The paper describes the use of simulation tools for teaching CS architecture to students and produce evidence that the use of such tools usually promotes the comprehension of the involved concepts.

The academic literature available related to teaching computer organization and architecture is very rich and has a large and increasing number of interesting proposals; unfortunately a more extensive and refined literature review goes beyond the goals of this article. Table I summarizes the main characteristics of works just presented.

TABLE I
SUMMARY OF THE LITERATURE REVIEW.

Ref.	Main Characteristics
[7]	Describes a 150-hour traditional discipline on Advanced Computer Structure, part of the degree in Computer Engineering at the University of Murcia. It focusses on the analysis of the teaching methodology employed, the distribution of different subjects over the 150 hours and the synchronization between theoretical and practical classes.
[8]	Has its focus on motivational issues to promote students´ interests in learning computer architecture related topics. Suggests teaching approaches that combine computer architectures with structure, organization and design of ubiquitous systems.
[9]	Describes the use of CTPracticals, a module of Moodle, for teaching practical contents related to basic computer architecture disciplines.
[13]	It is about a computer architecture course that emphasizes the practical side through the development, by the students, of a pipelined CPU design project with a field-programmable gate array (FPGA) system.
[14]	Presents a teaching simulator, SICOME 2.0, which was used in practices, related to learning computer architecture; the simulator allows an interactive simulation on simple computer architecture.
[15]	The article focuses on the importance of practical experiments when teaching organization and architecture of computers, and emphasizes the use of simulators for the experiments.
[16]	The paper describes the use of simulation tools for teaching CS architecture to students and produces evidence that the use of such tools promotes the comprehension of the involved concepts.

IV. ADAPTING A FACE-TO-FACE HIGH PERFORMANCE ARCHITECTURE DISCIPLINE TO REMOTE MODE

As previously mentioned, the High-Performance Architecture (HPA) discipline [16-21] is offered in the 7th. semester of the regular period of 5 years required for completing the BCE degree. It is a mandatory discipline for those majoring in BCE and an elective discipline for those majoring in BCS. It is the last of the three disciplines related to Computer Organization and Architecture taking into account the curricula of both courses.

The main goal of HPA is to capacitate students to have a good understanding of the main types of nonconventional high-performance computer architectures, that prioritize low energy consumption rates and also, being able to design them. The course description has, among its main topics: Heterogeneous System Architectures (HAS), Accelerators, Application-Specific Instruction Set Processors (ASIP), Graphics Processing Units (GPS), DSP (Digital Signal Processors) and SoCs (System on a Chip).

A detailed description of the contents of theoretical classes is presented in Table II and of the practical classes in Table III, both in Section B. It is important to mention that much of the knowledge/problems presented in these classes were based on the contents of [19-21].

A. Strategies Adopted for Maintaining the Educational System Active

Due to the pandemic and the subsequent quarantine scheme installed to prevent contamination, by March 2020 the whole Brazilian educational system needed to be completely customized to the restrained social contact, and it went under a complete re-structuring process so to be adapted to be functional under the many imposed restrictions.

Thus, the first-time offered HPA discipline, which was theoretically and practically structured to be traditionally taught during a sixteen-week period, had to be completely redesigned and reorganized for the new scenario of online remote classes lasting eight weeks, that was adopted by UFSCar. Therefore, all the previous carefully planning of the discipline encompassing teaching strategies, programmed distribution of its theoretical and practical contents throughout the sixteen weeks period, the chosen simulation tools, the equipment to be used for the practices, and many more practical issues, had to be readjusted in relation to both, the remote approach adopted by the university and the short period of time available to implement the changes.

To implement the transition it was also necessary, in a short period of time, to train the tutor and technical personnel, as well as to motivate and guide the students' attitudes in face of the new challenge. The whole process required a full and intense engagement by the lecturer and the technical personnel involved, mostly due to the short time notice for implementing the change from traditional the remote mode of teaching.

B. Customizing the HPA Discipline for Remote Mode

The HPA discipline was planned and ready to be offered in a face-to-face mode in the first semester of 2020, starting at the beginning of Mach.

As already pointed out in the previous section, due to the pandemic and the mandatory precautions adopted by the university for dealing with it, the teaching of the HPA discipline had to undergo several changes and adaptations so that the program contents of the discipline, usually taught over a period of sixteen weeks in a face-to-face mode, could be remotely taught over a period of eight weeks.

Regarding the theoretical contents of the discipline, it was decided to record a sequence of short videos specifically focusing on a sequence of the many concepts and theoretical results relevant to the discipline and upload them on the online video platform YouTube [22], so that students could access the theoretical contents, considering that the interaction among teacher, tutor and students were required to happen asynchronously.

As part of the university planning for dealing with the pandemic, the Moodle system [11,12] was adopted, which helped to turn the adaptation process quicker, considering the several facilities provided by the Moodle platform, when organizing and storing all the necessary files and links related to the discipline; on top of that, allowing the exchange of messages among all the participants, via forums. Also, through the Moodle platform students were able to access results and grades of tests and tasks related to evaluation processes. In addition to posting explanatory videos for each class, after one week, students were asked to post via Moodle, solutions related to issues regarding the topics covered in the previous online class. Once such posts were uploaded to the Moodle system, the tutor and teacher had a week to mark them and give feedback to the students.

For implementing a computational environment for running the practical tasks related to the discipline, computers were installed in the CS Dept. laboratory, coupled with educational kits, containing ARMs (Advanced RISC Machines, where RISC is an acronym for reduced instruction set computing) processors [23] and FPGAs (Field Programmable Gate Array [24], as well as software for the development of parallel codes (see Table II), prepared for remote access, using the AnyDesk tool [25]. Figure 1 presents a diagram representing the computational environment used for the remote teaching of the HPA discipline.

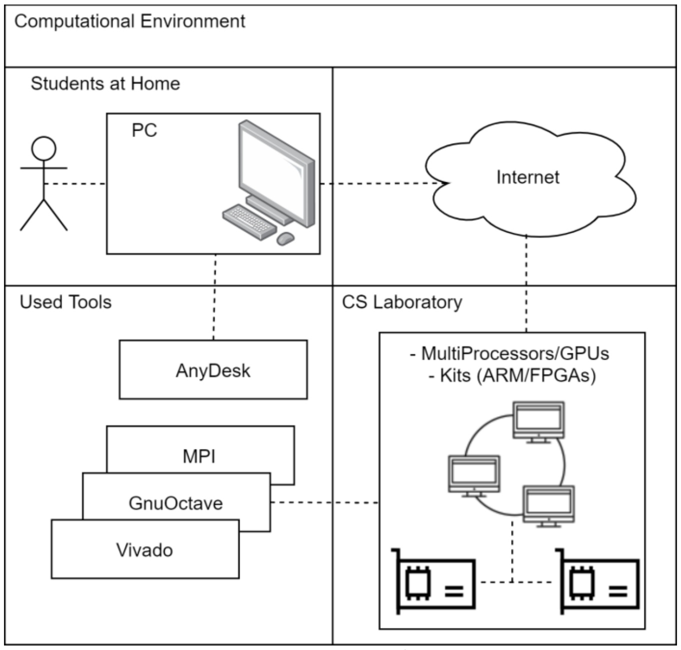

Fig. 1. Computational environment used for remotely teaching the HPA discipline. The four main tools used were: AnyDesk, installed in the students' PC and the other three MPI, GnuOctave and Vivado, in the PCs at the CS Dept. laboratory (diagram on the bottom right side of the figure, where both rectangles at the very bottom represent FPGAs).

Basically, students logged on their PCs through the AnyDesk tool and, by doing that, were able to access the PCs at the CS Dept. laboratory and have access to all the development tools and simulators to implement the practices described in Table II. For example, the first practice demanded only the use of software tools to run the experiments on the multicore processors installed in the laboratory PCs. For that, each student had an area on the hard disk of the laboratory PCs to run and test their codes, in addition to being able to store the results of their experiments for that given practice.

Some practices required the use of the processors in the development kits coupled to the PCs. Thus, access was allowed through the CAD tools described in Table II which, together with the Anydesk tool, allowed students to check and view the results of their experiments by emulating terminals via the use of such tools, without the need to have cameras capturing the behavior of LEDs and hardware kit keys, for example, besides other peripherals that could be used, otherwise, it would be necessary to have the presence of a technician in the laboratory, which was not viable due to the restriction rules.

The totally automatic new methodology devised contributed for emphasizing the social isolation required throughout the duration of the discipline. Finally, practices involving implementations of hardware architectures used the Xilinx's Vivado software [26] for the synthesis and analysis of HDL designs, while those involving only the development of parallel applications used the MPI (Message Passing Interface), a standard for data communication processes in parallel computing [27] and GnuOctave, a scientific programming language [28].

TABLE II
CONTENTS OF THE EIGHT THEORETICAL MODULES OF HPA DISCIPLINE.

1	•Performance •Moore´s law, •Architectural innovations •Energy consumption •ISA extensions for performance •Parallelism at instruction level •Classification of computing systems •Amdhal´s law.
2	•Vector and array processor • Interleaved memory •SIMD •MSIMD •MIMD.
3	•Multiprocessing with shared memory •Interconnection networks: Processor-Memory •Butterfly and the related Beneš network as examples of processor-to-memory interconnection network in a multiprocessor •Shared memory programming & broadcasting •Multiple caches and cache´s coherence •Snoopy cache protocol of coherence • Structure of a distributed shared-memory multiprocessor •Distributed shared-memory multiprocessor with a cache directory and memory module associated with each processor • Structure of a ring-based distributed-memory multiprocessor.
4	•Structure of distributed multicomputer and of a generic router •Send & receive message-passing primitives to synchronize two processes •Direct and indirect interconnection networks •Messages and their parts for message passing •Wormhole switching •Routing algorithms •Arbitration algorithms •Growing clusters via modular nodes •Network of workstations.
5	•Moore´s law •Factors affecting performance •Problems with CPUs •Accelerators •Graphic Processing Unit (GPU) × Central Processing Unit (CPU) •Examples of commercial GPUs •SMs (Streaming Multiprocessors) •Programming GPUs •Supercomputers •Kernels, Threads, Warps •Heterogeneous memory.
6	•SoC (System on a Chip) •Hardware architecture of a SoC •The *Zynq*®-7000 SoC family • Relation between software and hardware in Zynq • Architecture of the hardware system of an embedded SoC • Project flowchart for Zynq SoC • Zynq processing system •APU (Application Processing Unit) •Neon

	Engine (SIMD) • I/O Peripheral Interfaces • PL (Programmable Logic) • DSP & Blocks of RAM • GPIO •Communication interfaces.
7	•Embedded systems •Applications • Generic architecture • Processors, Co-processors and Cache •DRAM × SRAM •Caches L1, L2 and L3 •Execution cycles •Fetching, decoding and executing instructions •Interruptions •Bus, master, slaves • DMA (Direct Access to Memory).
8	•ARM architecture •ARM cortex A9 •PS components • •Interconnections among masters and slaves •PS interconnections •Memory map •Memory resources •Boot • Processor´s peripherals •MIO (Multiplexed IO) •Interfaces PS-PL •Zynq reset •Interfaces AXI (Advanced Extensible Interface).

TABLE III
CONTENTS OF THE EIGHT PRACTICAL MODULES OF HPA DISCIPLINE.

1	• Calculation of Pi (π) by Monte Carlo. -Step 1: State the method. To do that use a circle of radius r inscribed in a square of sides 2r. Do some research about the method to understand its basic idea. Deduce all equations involved. -Step 2: Use the Octave Parallel Processing Toolbox [28] to implement your code with a very large number (n) of points, in addition to using 4 processors of its multicore architecture. You must first get familiarized with the toolbox and the facilities it offers. -Step 3: Compare the previous result with the result of a serial implementation. Inform the speedup found and comment about it. -Step 4: Describe all the steps in a document containing all figures, equations and codes used. Comment about the type of parallel architecture (as seen in theory) most suitable to be used for the resolution of the problem. -Step 5: Upload the tutorial and script.m to the corresponding task (Practice 1) in the Moodle, with the title: Practice_1_Pi_Octave.
2	• Calculation of Pi (π) by Monte Carlo. -Step 1 & 3: The same as in Practice 1. -Step 2: Use the MPI library for Python to implement your code with a very large number of points (n), plus four processors in its multicore architecture. At this point, you must first understand the MPI library and its associated functions [27]. -Step 4: Produce a tutorial with a description of all steps as well as figures, equations and codes (functions) used. Comment about the type of parallel architecture (as seen in theory) most suitable to be used for the resolution of the problem. -Step 5: Upload the tutorial and the script.py to the corresponding task in the Moodle, with the title: Practice_2_Pi_MPI.
3-4	•Calculation of PI (π) by Leibniz. -Step 1: State the Method ([29]) and deduce all equations involved. Tip: use the Taylor series expansion to simplify. -Step 2: Use the Zybo kits from Xilinx. The idea is to build a multicore shared memory architecture to implement Step 1. So you should specify, in the Vivado software (2017.4), the Arm Processor (665 MHz dual-core Cortex - A9) contained in the Zynq chip (PS side) of the board, containing two cores and a BRAM memory (PL side - FPGA Artix-7) connected to a controller that should connect to the AXI bus of the present project. Note: tips on how to use this card can be found in [30] and also in Internet tutorials. -Step 3: In this step, you must use the SDK environment integrated with Vivado to develop the codes for each processor. Obs: do not superimpose the memories of both cores; try to figure out how to run both codes simultaneously ([31]); -Step 4: Produce a tutorial describing all the steps, with the figures, equations and codes (functions) used; -Step 5: Upload the tutorial and project for this practice task on Moodle with title: Practice_3_Arq_Mem_Compart. -Note: It will also be possible to access a shared card on the Internet to conduct the tests. For that, you should contact the tutor.
5	•Develop two modules in FPGA, one in Verilog and the other in HLS, for multiplication of two unsigned numbers of 16 bits; After the development of both modules, they should be interfaced with the Zynq/Zybo ARM processor for testing [33].
6	•Develop an FPGA module at Vivado HLS 2017.4, for detecting edges of images in gray levels. -Step 1: you should describe the theoretical edge detection process, which can be found in [34]; -Step 2: detail all the steps taken in [35] regarding the implementation of the process at Vivado HLS; You must understand, simulate and implement the project for verification.

V. SURVEY AND EVALUATION OF HPA

Once the HPA discipline was finished, the students were asked to fill out an on-line survey contained 28 statements about different aspects of the teaching/learning processes experienced by the students, particularly those related to: (a) its objectives, organization, methodology and contents, (b) synchronization between the development of the course in relation to theory and practices, (c) the necessary background knowledge from previous disciplines, (d) the relevance of the practical classes for helping to grasp theoretical concepts of the discipline, (e) the remote teaching/learning processes carried on, (f) evaluation of the use of software tools throughout the discipline (g) interactions with the lecturer and the tutor, (h) self-evaluation with respect to motivation and acquired knowledge. The survey was conducted using the Google Forms [32], software suitable for creating and managing surveys.

Students were instructed to evaluate each statement according to the category-value shown in Table IV. The 28 statements evaluated are described in Table V and Table VI present the numbers related to the evaluation.

TABLE IV
CATEGORIES USED IN THE STUDENTS' SURVEY.

Category	Integer value associated
Strongly disagree	1
Disagree	2
Neutral	3
Agree	4
Strongly agree	5

In general, the marks assigned by the students to the 28 statements in the survey were satisfactory, as can confirmed by the error values within the 95% confidence interval associated with each of them. In particular, statements 13 and

18 which are bold faced in Table V, had higher values for errors when compared with those by the others.

TABLE V
STATEMENTS IN STUDENTS' SURVEY.

ID	Statements
1	The objectives of the course were clearly described.
2	There was coherence between objectives and the content taught.
3	The evaluation criteria adopted was clear and fair.
4	The bibliography provided was easy to find and suitable for the content of the course.
5	You were able to fully understand the mathematical concepts involved in the course.
6	You were able to fully understand the mathematical concepts involved in the course.
7	The practical experiments contributed to a better understanding of the theoretical foundations presented by the lecturer.
8	**The use of the Vivado-HLS tool for describing accelerator circuits in some practical classes was more intuitive, practical and motivating than the use of the Verilog-HDL for the same practical classes.**
9	**The development of architectures (containing ARM cores) in some practical classes, using Vivado-HLS was more intuitive, practical and motivating than the use of the Verilog-HDL for the same practical classes.**
10	**To be able to use high-level tools for practical classes related to GPU and PC multicore processors helped you to enhance your grasp of theoretical concepts taught.**
11	Both, the use of tools and the practical classes were very satisfactory for a better understanding of the discipline´s contents.
12	The adopted methodology was suitable for the 8-weeks emergency remote discipline.
13	**This discipline was the first you attended in remote mode.**
14	**The use of computers connected to FPGA integrated circuits was an efficient solution for the distance learning you have experienced.**
15	The practical classes using Octave, MPI and GPU were effective for a better understanding of the basic theoretical concepts presented and discussed in theoretical classes.
16	You would recommend the use of the same methodology for a new offering of the discipline.
17	There were substantial differences between both approaches of teaching: the face-to-face and remote.
18	**Taking into consideration your performance in the discipline, you would have learnt much more if the discipline was taught in face-to-face mode.**
19	Your interaction with your colleagues was satisfactory.
20	The lecturer mastered the contents of the discipline and was available for clarifying doubts related to the discipline´s contents.
21	The interaction with the tutor was satisfactory and always helped to clarify doubts.
22	The programmed set of students' activities was pertinent and promoted/refined the acquired knowledge from the theoretical classes.
23	Theoretical and practical classes were synchronized.
24	Your commitment to the discipline was satisfactory.
25	You have done all the proposed activities, watched all video classes, invested in complementary reading, exercise solving, requested implementations, etc.
26	Grade your interaction with the lecturer and the tutor.
27	You were motivated by the experiments and theoretical verifications; that helped to sustain your motivation in relation to the discipline as a whole.
28	You would recommend this discipline to a colleague of yours.

In Table VI the answers related to the statement 13 revealed that to some students attending the HPA discipline taught in remote mode was their first experience with such mode of teaching/learning. Based on results related to statement 18 it can be inferred that the discipline, remotely or face-to-face taught, would possibly reach the same results as far as learning is concerned.

Also, the obtained results regarding the strategies adopted mentioned in statements 8, 9, 10 and 14 (i.e., HLS for describing accelerator circuits, development of architectures containing ARM cores, use of high level tools related to GPUs and multicore processors and, finally, the use of computers connected to the FPGAs) proved to be promising for the teaching/learning process of the HPA discipline, considering that the proposed computational environment allowed the real implementation of the architectures instead of just using simulators for that purpose.

Out of the total of 11 students enrolled in the discipline, 8 students answered the survey. The two students who failed to answer the survey were the two students who failed the discipline. The small number of students enrolled in the HPA discipline in 2020 was unusual. It seems that one of the reasons for such a low number was the fact that the discipline changed mode, from face-to-face to remote. Due to the reorganization of all the disciplines from face-to-face to remote mode, there was a tendency among students of enrolling in as many as allowed. Many of them, however, were not able to cope with the load of work required and several dropped out of some disciplines at their very beginning.

TABLE VI
RESULTS ASSOCIATED WITH THE 28 STATEMENTS IN THE STUDENTS' SURVEY, WHERE AVG STANDS FOR AVERAGE, STD FOR STANDARD DEVIATION AND E-CONF_INT FOR ERROR WITHIN CONFIDENCE INTERVAL.

ID	Avg	Std	E-Conf_Int (95%)
1	3.88	0.93	0.64
2	4.13	0.60	0.42
3	4.25	0.83	0.57
4	3.50	1.22	0.85
5	3.25	1.20	0.83
6	3.50	0.71	0.49
7	4.00	0.87	0.60
8	**4.25**	**0.83**	**0.57**
9	**4.00**	**0.71**	**0.49**
10	**4.38**	**0.70**	**0.48**
11	3.88	0.60	0.42
12	3.63	0.86	0.59
13	**3.63**	**1.80**	**1.25**
14	**3.00**	**1.32**	**0.92**
15	4.38	0.48	0.34
16	3.50	1.12	0.77
17	4.50	0.71	0.49
18	**3.00**	**1.50**	**1.04**
19	4.00	1.12	0.77
20	5.00	0.00	0.00
21	4.75	0.43	0.30
22	4.25	0.43	0.30
23	4.25	0.43	0.30
24	4.38	0.48	0.34
25	4.00	0.87	0.60
26	4.50	0.50	0.35
27	3.50	0.87	0.60
28	4.00	1.00	0.69

VI. Conclusions and Future Perspectives

This article has its focus on the work and the many tasks involved in adapting, at short notice, a face-to-face CS-related discipline into a discipline to be taught remotely, as a consequence of the adopted sanitary precautions to avoid the pandemic provoked by the covid-19. Initially, aiming at contextualizing the academic environment the work took place, the article gives general information about the Federal University of S. Carlos, located in S. Carlos-SP, Brazil, and also, about the CS Dept. and the two undergraduate CS-related courses the department offers.

Next the two undergraduate courses are detailed in relation to the disciplines that compose their curricula. Aiming at contextualizing even deeper the HPA discipline focus of the work, the article presents a short literature review of works related to methodologies and teaching techniques employed for teaching concepts, results, hardware, software, etc. related to Computer Organization and Architecture, that have been used in other academic institutions commonly taught in a face-to-face mode.

Next, the High-Performance Architecture (HPA) discipline, which can be considered the last stage of academically dealing with computer organization and architecture, is approached in detail, considering it is the main focus of the work done.

As previously mentioned, in a face-to-face mode where theoretical classes are given in classrooms and practice classes in the laboratory provided by the CS Dept., the teaching of HPA spans over a period of 16 weeks. Taking into account the emergency restrictions imposed by the university, to adapt and condense the contents of HPA to eight weeks in remote mode required a considerable effort from the team formed by the lecturer, tutor and technicians involved in the process, to fully re-adapt the discipline, both theoretically and practically, to the new scenario prescribed by the university. In addition, it was a consensus among the members of the team that by only using simulators would not be very effective, taking into account the advanced computer architectures that are presented and discussed in the HPA discipline.

Thus, regarding the adaptations required in relation to the topics *versus* time of remotely teaching HPA, the team decided to innovate, by designing an online and automated structure, where theoretical classes would be recorded offline and their corresponding videos, together with the other related files would be made available on UFSCar's Moodle system, which was also used as a communication and evaluation tool among the lecturer, tutor and students.

In relation to practical classes, laboratory kits containing FPGAs and ARM processors, and their development tools, were installed on the computers in the laboratory and, by using a remote access tool, each student was able to carry out the practices through the terminals emulated on their PCs. Thus, all this adaptation allowed students to be evaluated in relation to the theoretical and practical contents required by the discipline. At the end of the discipline an evaluative questionnaire was filled out by the students and marks were given in relation to several methodological aspects, particularly those involving theoretical and practical teaching/learning processes. The results can be considered satisfactory, with a 95% confidence interval, which supports the viability of the adapted teaching/learning strategy for a period of eight weeks during the pandemic.

This article presents and discusses several issues related to implementing the adaptation in a short period of time and, particularly, how laboratorial experiments were adapted for being conducted remotely. The students' feedback at the end of the course, combined with their comments and their grades in tasks done throughout the course emphasize that the discipline met its purposes i.e., that of promoting the learning of several issues related to high-performance computer architecture.

Taking into account the literature review conducted for contextualizing the work described in this article, although its contribution was good in some aspects (such as the emphasis in synchronizing theory and practices), the articles did not contribute much (except [9]) for helping to plan the adaptation of a face-to-face discipline to a remote mode, particularly taking into consideration the short period of time for accomplishing the adaptation task. It is a fact though that the shortage of time due to the pandemic (or any other event of such impact), has not been contemplated in any of the selected literature works. So, under the perspective of 'adaptation under the pressure of time', the work done was original and successful.

Considering a non-pandemic scenario and in an attempt to foresee the future use of the material and strategies produced for the remote teaching of the high-performance computer architecture discipline, the authors believe that all the efforts and investments in producing the adaptation of the discipline can be used and refined in educational environments where the remote teaching is adopted i.e., on-line learning environments.

Acknowledgments

Both authors thank the DC-UFSCar for its constant support. The second author also thanks the CNPq.

References

[1] D Cohen, "Introduction to Computer Theory", John Wiley & Sons, 1986.
[2] T Stuart, "Understanding Computation: From Simple Machines to Impossible Programs", O'Reilly Media, 2013.
[3] R Graham, D Knuth, O Patashnik, "Concrete Mathematics: A Foundation for Computer Science", Addison-Wesley Professional, 2nd. ed., 1994.
[4] Z Sen, "Innovative Trend Metodologies in Science and Engineering", Spring-Verlag, 1st. ed., 2017.
[5] R E Bryant and D R O'Hallaron, "Computer Systems: A Programmer´s Perspective", 3rd. ed., Pearson India Education Services Pvt. Ltd., 2016.
[6] M C Nicoletti, A M S Reali, S Abib, and V Neris, "Developing a new course at an Open University," *Asian Journal of Distance Education*, vol. 10, no. 2, pp. 36-45, 2012 Retrieved from http://www.asianjde.org/ojs/index.php/AsianJDE/article/view/199
[7] G Bernabé, "Teaching experience in advanced computer structure,"*2016 International Symposium on Computers in Education (SIIE)*, Salamanca, 2016, pp. 1-6, doi: 10.1109/SIIE.2016.7751817.
[8] A Clements, "Work in progress - computer architecture meets ubiquitous computing," *2009 39th IEEE Frontiers in Education Conference*, San Antonio, TX, 2009, pp. 1-2, doi: 10.1109/FIE.2009.5350715.
[9] M A Trenas, J Ramos, E D Gutierrez, S Romero, and F Corbera, "Use of a New Moodle Module for Improving the Teaching of a Basic Course on Computer Architecture," in *IEEE Transactions on Education*, vol. 54, no. 2, pp. 222-228, May 2011, doi: 10.1109/TE.2010.2048570.
[10] E Gutiérrez, M A Trenas, J Ramos, F Corbera, and S Romero, "A new Moodle module supporting automatic verification of VHDL-based assignments", *Computational Education*, v. 54, no. 2, 2010, pp. 562-577.
[11] "Moodle developer documentation," [Online]. Available:

http://docs.moodle.org/en/Development

[12] Moodle site, [Online]. Available: http://moodle.org

[13] J H Lee, S E Lee, H C Yu and, T Suh, "Pipelined CPU Design With FPGA in Teaching Computer Architecture," in *IEEE Transactions on Education*, vol. 55, no. 3, pp. 341-348, Aug. 2012, doi: 10.1109/TE.2011.2175227.

[14] M Brox, A Gersnoviez, M A Montijano, E Herruzo and, C D Moreno, "SICOME 2.0: A teaching simulator for Computer Architecture," *2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE)*, La Laguna, 2018, pp. 1-7, doi: 10.1109/TAEE.2018.8476041.

[15] W Kurniawan and M H H Ichsan, "Teaching and learning support for computer architecture and organization courses design on computer engineering and computer science for undergraduate: A review," *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Yogyakarta, 2017, pp. 1-6, doi: 10.1109/EECSI.2017.8239076.

[16] P W C Prasad, A Beg, A Chan, "Using simulators for teaching computer organization and architecture," *Computer Applications in Engineering Education*, v. 24. no. 2, 2016, pp. 215-224, https://doi.org/10.1002/cae.21699

[17] R Robey and Y Zamora, "Parallel and High Performance Computing", Manning Publisher, 2021.

[18] G Hager and G Wellein, "Introduction to High Performance Computing for Scientists and Engineers", Chapman & Hall/CRC Computational Science, 2010.

[19] B Parhami, "Computer Architecture: From Microprocessors to Supercomputers," The Oxford Series in Electrical and Computer Engineering, Oxford University Press, 2005.

[20] A Lastovetsky and J Dongarra, "High-performance Heterogeneous Computing," Wiley-Interscience, 1st ed., 2009.

[21] W Stallings, "Computer Organization and Architecture: Designing for Performance," 10th ed., Pearson, 2015.

[22] https://www.youtube.com

[23] https://www.arm.com/

[24] S M Trimberger (ed.) "Field Programmable Gate Array Technology," Springer Science+Business Media, LLC, 2013

[25] https://anydesk.com/en

[26] https://www.xilinx.com/products/design-tools/vivado.html

[27] https://www.open-mpi.org/

[28] https://octave.sourceforge.io/parallel/

[29] https://crypto.stanford.edu/pbc/notes/pi/glseries.html

[30] https://reference.digilentinc.com/learn/programmable-logic /tutorials/zybo-getting-started-with-zynq/start

[31] https://www.youtube.com/watch?v=n0hbwp36hBs

[32]google.com/forms/about

[33]http://venividiwiki.ee.virginia.edu/mediawiki/index.php/ToolsXilinxLabs RTLHLSIP

[34] R C Gonzalez and R E Woods, "Digital Image Processing", 3rd ed., Pearson, 2007.

[35] https://www.hackster.io/adam-taylor/fpga-based-edge-detection-using-hls-192ad2