COURSE SYLLABUS

1. **Program information**

Faculty	Faculty of Electrical Engineering and Computer Science
Department	Computer Science
Field of study	Computers and Information Technology
Cycle of studies	Master's Degree
Study Program	Artificial Intelligence & Social Robotics Applied to Global Health

2. **Course information**

Course name		**NUMERICAL SIGNAL PROCESSING SYSTEMS**				
Year of study	1	Semester	2	Type of assessment	Exam	
Course regime		Course category by educational content: DF – Fundamental, DS – Specialization, DC – Complementary				DS
		Course category by optionality: DOB – Compulsory, DOP – Optional, DFA – Elective				DOB

3. **Total estimated time (hours allocated to teaching activities)**

I a) Number of hours per week	3	Lecture	1	Seminar	-	Laboratory / practical work	1	Project	1
I b) Total number of hours per semester according to curriculum	42	Lecture	14	Seminar	-	Laboratory / practical work	14	Project	14

Distribution of time allocation per semester (hours)	No. hours
II. Individual study	105
III. Examinations	3
IV. Other activities (specify):	-

Total individual study hours (II+III)	108
Total hours per semester (I.b+II+III+IV)	150
Number of credits	6

4. **Specific competencies acquired**

Transversal competencies	CT1 – work in teams
Professional / general competencies	CP1 – perform scientific research CP2 – develop product design CP3 – design prototypes CP8 – establish data processes CP12 – perform project management

5. **Learning outcoms**

Knowledge	**Skills**	**Responsibility and autonomy**
The student/graduate identifies, critically evaluates, and applies as a foundation for original research specialized concepts and methods from software engineering and numerical signal processing.	The student/graduate applies numerical methods and advanced signal processing algorithms for the efficient acquisition, processing, analysis, and interpretation of data in intelligent systems using specialized platforms.	The student/graduate manages software development processes within a team across various contexts and platforms, taking responsibility for architectural and implementation decisions.

6. **Course objectives** (derived from the grid of specific competencies acquired)

General objective of the course	Development of advanced competencies in concepts and techniques of digital signal processing, the design and implementation of prototypes, and the extraction of information from data/signals; development of teamwork skills.

7. **Teaching and learning content**

Lectures	**No. of hours**	**Teaching methods**	**Remarks / Notes**
1. The ZYNQ Architecture (From ZYNQ to ACAP/Versal: CPU + FPGA + AI Engines for DSP and AI) 1.1. Overview: PS + PL 1.2. Processing System (PS) 1.3. Programmable Logic (PL) 1.4. The PS–PL interconnect: AXI 1.5. Clocking, I/O and boot 1.6. ZYNQ for DSP 1.7. ZYNQ for AI 1.8. Design flow and comparison 1.9. The next generation: ACAP (Versal)	5	Lecture, discussion, exemplification, demonstration	
2. The PYNQ-Z2 Board (Python + FPGA for hands-on DSP and AI) 2.1. PYNQ-Z2 kit (description) 2.2. The PYNQ framework 2.3. Python ↔ hardware 2.4. DSP & AI on PYNQ-Z2 2.5. Teaching & getting started	4		
3. The PYNQ Project Toolchain (From a C/C++ algorithm, through Vivado, to Jupyter) 3.1. The big picture 3.2. Stage 1: Vivado / Vitis HLS (High-Level Synthesis) 3.3. Stage 2: Vivado (schematic / block design) 3.4. Stage 3: Jupyter / PYNQ 3.5. The economics of iteration 3.6. A generic recipe	5		

Recommended minimum bibliography

1. Maurice Bellanger, Benjamin A. Engel. (2024). *Digital Signal Processing: Theory and Practice*, 10th Ed, John Wiley & Sons.
2. Steven W. Smith. (2002). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing San Diego.
3. David Smith. (2019). *Digital Signal Processing for Complete Idiots.* Independently Published.
4. Allen Downey. (2016). *Think DSP: Digital Signal Processing in Python*. O'Reilly Media.
5. Will C. Pirkle. (2021). *Designing Software Synthesizer Plugins in C++: With Audio DSP*. Focal Press.

Applications (Laboratory)	**No. of hours**	**Teaching methods**	**Remarks / Notes**
Through these lab exercises, students go through a complete Hardware/Software Co-Design flow on heterogeneous System-on-Chip platforms, learning to transform C/C++ algorithms into dedicated hardware accelerators via High-Level Synthesis (Vivado HLS). They acquire the ability to assemble complex architectures in Vivado, integrating the ARM processor with DMA memory modules, AXI buses, and specialized IPs for signal processing (FIR filters and FFT spectral analysis) or image processing (Sobel edge detection). Furthermore, students master interfacing with the PYNQ-Z2 reconfigurable board, learning to control hardware accelerators directly from Python using the Jupyter Notebook environment. Last but not least, they understand how to overcome the FPGA's resource limitations through tiling techniques applied to high-resolution video frames, gaining a practical, end-to-end perspective on designing and optimizing modern digital systems.			
1. Part 1 — Vivado HLS (Designing and verifying the FIR kernel, step by step) ● What is a FIR filter? ● Where this fits ● Creating the project ● The kernel ● C simulation ● Synthesis and export as IP	1	Practical work, experiment, discussion, case study	

2. Part 2 — Vivado Block Design (From an HLS IP to a full hardware overlay) ● Where it fits into the overall project ● New project ● Assembling the IP blocks ● Wiring up the blocks ● Validate, generate, done	2		
3. Part 3 — Python on PYNQ-Z2 (Running the real overlay, step by step) ● Where it fits into the overall project ● Getting the overlay onto the board ● The notebook ● The real result	1		
4. Part 4 — An FFT block in the Vivado design (The signal spectrum, computed by the FPGA and shown in Python) ● The idea: FFT as a hardware IP ● Configuring the FFT IP ● The complete design: FIR + FFT ● The Python code ● The real result	1		
5. Part 5 — Different filters, seen through the FFT (A multi-frequency signal, filtered and spectrally analyzed on the FPGA) ● The signal and the filters ● The Python code ● The real result	1		
6. Sobel Filter on PYNQ-Z2 — Part 1 (Designing and verifying the edge-detection kernel, in Vivado HLS) ● Sobel algorithm basics ● Creating the project ● The kernel ● C simulation ● Synthesis and export as IP	2		
7. Sobel Filter on PYNQ-Z2 — Part 2 (Block design in Vivado: from an HLS IP to a full bitstream) ● New project ● The processor and automation ● Assembling the IP blocks ● Connecting the blocks ● Validation, generation, wrap-up	2		
8. Sobel Filter on PYNQ-Z2 — Part 3 (Running the real overlay on Python, step by step) ● Transferring the overlay to the board ● The notebook ● The real result	2		
9. Part 4 — Full-resolution Sobel, via tiling (A fixed 64×64 accelerator processes a 640×360 video, tile by tile, on real hardware) ● The idea ● The tiling geometry ● The Python code ● The real result ● Alternative: native hardware	2		
Applications (Project)	**No. of hours**	**Teaching methods**	**Remarks / Notes**
Development of a project that demonstrates the performance difference between software and hardware execution for a handwritten digit recognition application, using: ● ARM Cortex-A9 CPU (PS) ● Hardware accelerator on FPGA (PL) ● PYNQ-Z2 platform			

The model used is an MLP (Multi-Layer Perceptron) neural network, and the goal is to highlight the significant acceleration achieved through hardware parallelization. The implemented network has the following structure: Input: 1024 neurons (32×32 image); Hidden layer: 15 neurons; Output: 10 neurons (digits 0–9).			
1. The accelerator in Vivado HLS The accelerator is written in C++ and synthesized using Vivado HLS. ● m_axi interface - Direct access to DDR memory - Fast data transfer via DMA (Burst Mode) ● s_axilite interface - Control of the accelerator (start, addresses, result) ● HLS directives - PIPELINE for computation loops - Parallel execution of MAC operations Result: Very low latency per inference.	4	Practical work, experiment, discussion, case study	
2. Integration in Vivado (Block Design) Main components: ● Zynq Processing System ● AXI SmartConnect - for the data path (m_axi → DDR) ● AXI Interconnect - for the control path (s_axilite) ● Processor System Reset - clock and reset synchronization Separating the data path from the control path is essential for performance.	4		
3. Software Implementation (Python + Jupyter) The software application runs on ARM + Linux, using Jupyter Notebook. ● Loading and configuring the FPGA hardware ● Reading and organizing input data - Network weights (W and B) are read from text files and concatenated into a single continuous vector. - Input images (digits or phone numbers) are read from .coe or .txt files and converted into vectors of appropriate size. ● Dual implementation of the inference engine - A software (CPU) version of the neural network is implemented for reference and validation. - A hardware (FPGA) version is invoked through the CNN IP, using buffers allocated in shared memory and DMA access. ● CPU–FPGA communication mechanism ● Benchmark and SW vs HW comparison - Execution times for software and hardware are measured, results are compared, and acceleration (speedup) is calculated. ● Test automation and result presentation - Tests are run on sets of digits and phone numbers, and the final results are summarized in a table highlighting the correctness and performance of the FPGA solution.	4		
Project presentation and evaluation.	2	Demonstrati on	
Attendance at compulsory activities is regulated by USV's "Framework Regulation on Student Assessment". Make-up for mandatory practical activities scheduled during the semester is carried out in accordance with the provisions of this regulation. Practical activities scheduled throughout the semester may be replaced by the completion and presentation of high-complexity projects from the course syllabus, with the approval of the course instructor.			

Recommended minimum bibliography

1. Maurice Bellanger, Benjamin A. Engel. (2024). *Digital Signal Processing: Theory and Practice*, 10th Ed, John Wiley & Sons.
2. Steven W. Smith. (2002). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing San Diego.
3. David Smith. (2019). *Digital Signal Processing for Complete Idiots.* Independently Published.
4. Allen Downey. (2016). *Think DSP: Digital Signal Processing in Python*. O'Reilly Media.

5. Will C. Pirkle. (2021). *Designing Software Synthesizer Plugins in C++: With Audio DSP*. Focal Press.

8. **Evaluation**

Type of activity	Evaluation criteria	Evaluation methods	Weight in final grade
Final evaluation	Clear presentation of theoretical concepts and correctness in handling notions and methods specific to scientific research in the field of numerical signal processing. Proper understanding of the methods used in scientific research for the design of DSP algorithms.	Written test, followed by oral verification of the extent to which the requirements in the written work are met.	20%
Evaluation of ongoing practical activities	Correct design and functional implementation of a hardware acceleration application performing vectorized mathematical convolution. Participation in various practical activities related to the specific objectives of the course.	Project evaluation, systematic observation	30%
	Correct design and functional implementation of an MLP (Multi-Layer Perceptron) neural network, with the goal of highlighting the significant acceleration achieved through hardware parallelization.	Project evaluation, systematic observation	50%

Physical accessibility: activities are conducted in classrooms and laboratories accessible to wheelchair users via ramps/lifts (USV buildings C/D); **Accessibility of teaching materials:** documents are available in digital PDF format with selectable text and slides with large fonts; **Evaluation accessibility:** ongoing evaluation through projects; possibility of extended time for the written tests and oral verification.

Date	Academic title, Name, Surname, Signature of Course Instructor	Academic title, Name, Surname, Signature of Practical Work Instructor
23.09.2026	Conf. univ. dr. ing. Cristian-Andy TANASE	Conf. univ. dr. ing. Cristian-Andy TANASE

Date of approval	Academic title, Name, Surname, Signature of Program Coordinator
24.09.2026	Prof. univ. dr. ing. Radu-Daniel VATAVU

Date of departmental approval	Academic title, Name, Surname, Signature of Head of Department
24.09.2026	Prof. univ. dr. ing. Ovidiu-Andrei SCHIPOR

Date of Faculty Council Approval	Academic title, Name, Surname, Signature of Dean
25.09.2026	Prof. univ. dr. ing. Laurentiu-Dan MILICI