Project code	Title of Project	Details	Limit
ESE1	**Verilog development of a simple embedded system (CPU + Memory + Peripherals)**	Goal is to develop a simple embedded system (CPU + Memory + Peripherals) in Verilog either by writing code of these modules from scratch or taking them from the internet. Simulate the complete embedded system.	
ESE2	**Computationally efficient instruction set architecture design for edge computing**	Follow from here: https://dl.acm.org/doi/10.1145/3007787.3001179 Aim is to develop a subset of any standard ISA in order to complete usual tasks in Deep learning such as Multiply-Accumulate (MAC), load, store etc. This may also be useful: https://ieeexplore.ieee.org/document/9718079	
ESE3	**Simulating a DL/AI accelerator in Verilog**	https://github.com/padhi499/Image-Classification-using-CNN-on-FPGA Verilog codes are available under "VerilogCodes" folder and block diagrams are also available in a separate folder. OR Verilog-based accelerator implementation:https://github.com/taoyilESE/clacc OR https://github.com/ChrisZonghaoLi/Rudi_CNN_Conv_Accelerator	
ESE4	**DL model optimization for Lightweight Gesture Recognition**	Automatic recognition of gestures, sign languages, and interaction of humans from the camera input of mobile phones or microcontrollers. Recommended Dataset: Jester dataset	
ESE5	**DL model optimization for Lightweight Face Mask Detection**	Automatic detention of whether a person is wearing a mask or not from either camera input or microphone input of mobile phones or microcontrollers. Recommended Dataset: FDDB, WiderFace, MaFA, and Kaggle Medical Mask	
ESE6	**Verilog implementation of Microcontroller-based system and running hex code of any useful application on it**	Goal is to model a microcontroller-based embedded system in Verilog and run the hex code (corresponding to any C application) during simulation. The simulation can be done in a simulator like Vivado etc.	
ESE7	**Smart Agriculture implementation with embedded system (e.g- Arduino+NodeMCU/Raspberry Pi etc.)**	Goal is to implement a sensor-based data acquisition system with embedded system (e.g- Raspberry Pi) and run some ML algorithms etc. on it for some prediction/forecast corresponding to particular task. Sensors need to be taken from the laboratory in EE Department.	3 Or 4
ESE8	**Getting familiarity with TinyML and one demonstration of model usage on hardware (Raspberry Pi kit etc.)**	Aim is to demonstrate the effectiveness of microcontrollers towards running ML applications. Any example TinyML application can be followed. For example- https://blog.tensorflow.org/2021/05/building-tinyml-application-with-tf-micro-and-sensiml.html	3 Or 4
ESE9	**Implementing AXI/AXI-Lite protocol for peripheral (like UART, timers) connections with CPU**	Aim is to implement from scratch/take help from the internet the interconnection protocol AXI/AXI-Lite in Verilog and connect some peripheral (Timer, UART etc.) with any CPU core and run simulations to check the functioning.	
ESE10	**Understanding USB/PCIE interconnect protocol for connecting CPU with peripherals and its simulation in Verilog**	Aim is to implement from scratch/take help from the internet the interconnection protocol like USB/PCIE in Verilog and connect some peripheral (Timer, UART etc.) with any CPU core and run simulations to check the functioning.	

ESE11	**Explore hardware security in embedded systems e.g. observe how a peripheral such as UART can be compromised or malicious code can be inserted during code compilation**	Take the verilog implementation of an embedded system (e.g- any CPU with different kinds of peripherals attached to it. You are allowed to take the codes from internet and then analyze in terms of hardware security like how an UART serial communication can be compromised to transmit wrong data because of attack OR how the CPU can start executing from wrong address because of an attack. You need to demonstrate such attack scenarios in simulation.
ESE12	**Hardware Implementation of ML algorithm (e.g. Principal Component Analysis) in reconfigurable manner**	https://jes-eurasipjournals.springeropen.com/articles/10.1186/s13639-017-0074-x Implement in Verilog The design can be implemented in C/C++ also
ESE13	**Neural Network for Microcontrollers (MCUNet) design analysis (from hardware resources usage viewpoint)**	https://mcunet.mit.edu/ Implement in Verilog if you wish The design can be implemented in Pytorch also, analyze the resources that this neural network is taking to run different tasks
ESE14	**Neural Network Quantization for mobile/resource-constrained devices**	Benchmark/analyze methods of improving post-training quantization (*e.g.*, weight equalization, bias correction), summarize the difficulties of post-training quantization on different types of neural networks. Explore algorithms for improving the accuracy of low-bit-width quantization (*e.g.*, 4-bit, ternary, binary quantization).Explore algorithms for quantized training, *e.g.*, training with 8-bit integers. Example: https://github.com/Tiiiger/QPyTorch
ESC1	**Running Neural Networks on Android devices**	Follow from here: https://arxiv.org/pdf/1810.01109.pdf Implementation: https://developer.android.com/ndk/guides/neuralnetworks Goal is to develop some API that can switch between AI models when required depending upon factors like the available resources in the device etc.
ESC2	**Understanding Linux Boot process on small embedded systems and a demonstration**	Take any small embedded system (C++/C) model and show how linux boot can be done. You can try modeling this for example: https://en.wikipedia.org/wiki/Das_U-Boot
ESC3	**Exploring Halide library/language and simple demonstration of a deep learning task like CNN processing**	Follow from here: https://github.com/binodkumar23/binodkumar23.github.io/blob/main/Halide_Report.pdf Implement new applications
ESC4	**Device driver development for small embedded systems**	Develop device driver code for small embedded systems
ESC5	**Neural Architecture Search for mobile/resource-constrained devices**	Benchmark/analyze different search algorithms (e.g., evolutionary search, reinforcement learning, bayesian optimization, differential evolution, genetic algorithm, and others). Can be followed from this paper: https://openreview.net/pdf?id=_0kaDkv3dVf
ESC6	**Neural Network Pruning for mobile/resource-constrained devices**	Explore/analyze different hardware-friendly pruning (e.g., channel pruning, 2:4 sparsity pruning) etc. Example Paper: https://arxiv.org/pdf/2205.10369.pdf

ESC7	**DL model optimization for Lightweight Keyword Spotting**	Automatic identification of keywords in utterances from the microphone input of mobile phones or microcontrollers. Recommended Dataset: Google Speech Commands
ESC8	**DL model optimization for Lightweight Visual Wake Words Task**	Automatic identification of whether a person is present or not from the camera input of mobile phones or microcontrollers. Recommended Dataset: Visual Wake Words Dataset
ESC9	**Software implementation of AI library and model efficiency evaluation**	Implement features from here: https://developer.qualcomm.com/software/ai-model-efficiency-toolkit OR, Follow from here: https://pytorch.org/mobile/home/
ESC10	**DL model optimization for Lightweight Pose Estimation**	Automatic detection of the position and orientation of a person by predicting the location of specific body key points, including hands, heads, elbows, and knees. From the camera input of mobile phones or microcontrollers. Recommended Dataset: MPII Human Pose, COCO
ESC11	**Real-time OS (RTOS) implementation**	Implement a RTOS for any practical application such as smartwatch, GPS tracker etc.
ESC12	**Understanding Apache TVM Compiler and ML model simulation on given Hardware design**	Attempt understanding the TVM compilation process and understand how it generates a hardware mapping corresponding to DL/ML model
ESC13	**Efficient Neural Network Kernels for Arm Cortex-M CPUs**	Follow from here: https://arxiv.org/abs/1801.06601
ESC14	**Efficient ML/DL inference for microcontrollers via operator execution optimization**	Follow from here: https://arxiv.org/abs/1910.05110
ESC15	**Neural Network for Microcontrollers (MCUNet) analysis and optimization**	https://mcunet.mit.edu/ design can be implemented in Pytorch also, analyze the resources that this neural network is taking to run different tasks
ESC16	**Hardware security analysis of embedded systems (application-level or micro-architectural level)**	Show/simulate attack scenarios on embedded systems, analyze in terms of hardware security like how an UART serial communication can be compromised to transmit wrong data because of attack OR how the CPU can start executing from wrong address because of an attack. You need to demonstrate such attack scenarios in simulation.

ALLOTMENT SCHEME:

1. Projects would be allotted on a first come first serve basis.
2. Every group/student is asked to give at least 7-8 choices.
3. Every project can be allotted to a maximum of 6-7/8 groups.
4. 2 projects have a hard limit of 3/4.
5. Once allotment is done by TA, it cannot be changed.

Choices must be made as per below Table of project codes (B19 can choose any topic):

B20EE001 to B20EE049	**ESE1 to ESE7, ESC8 to ESC14**
B20EE050 to last, All B20ES	**ESE8 to ESE14, ESC1 to ESC7**
B20CS001 to B20CS047	**ESC1 to ESC9, ESE1 to ESE6**
B20CS048 to last	**ESC8 to ESC16, ESE9 to ESE14**

EVALUATION SCHEME:

Have you understood the topic? You have got a plan to implement?	**03 Marks**
Have you implemented on your own OR with the help of some reference from internet? Are your results as per your expectations?	**05 Marks**
Have you done any innovation/improvement in your implementation? If yes, mention it in your report?	**05 Marks**
Have you written a README to run your codes/implementation?	**03 Marks**
TOTAL	**16 Marks**

PENALTY SCHEME:

Copied from internet	**-12**
Copied from other group and changed the language slightly	**-10**
Taken some help from internet but reference is not provided in the report	**-8**
No README is provided to run the codes	**-6**

SUBMISSION INSTRUCTIONS:

1. Only 2 members are allowed in a group/team.
2. Both team members need to submit on GC.
3. Submit all codes and a small Report file in a zip folder
4. Name your zip folder as ROLLNUMBER_PROJECTCODE.zip
5. No LATE SUBMISSION is allowed. A penalty of -1 per hour would be imposed.
6. Name your report as ROLLNUMBER_PROJECTCODE.pdf