UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Simulaciones de potencia de arquitecturas pipeline de aprendizaje automático para ASIC

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Fargueta Pelufo, Lidia

Tutor/a: Gadea Gironés, Rafael

Cotutor/a: Monzó Ferrer, José María

Cotutor/a externo: WISSING, JULIO

CURSO ACADÉMICO: 2021/2022

Resumen

La ejecución de algoritmos de machine learning en la nube conlleva altos requisitos de ancho de banda, un elevado consumo de energía y problemas en la seguridad de transmisión de los datos, entre otros. Diferentes sensores recogen datos y los envían a servidores remotos para que sean procesados. Una forma de solucionar los problemas mencionados es ejecutar estos algoritmos en el dispositivo local que no será tan potente como el remoto pero que serà més eficient. Este concepto es conocido como 'Edge Computing'. Este TFG se centra en esta parte de la red, donde los algoritmos de machine learning se integran en el sensor.

Uno de los principales problemas de la computación directamente en el sensor es el incremento del consumo de potencia. Por ello es importante encontrar la forma de reducirlo. El objetivo de este trabajo, realizado en Fraunhofer IIS en Erlangen (Alemania), es encontrar la configuración hardware eficiente en potencia adecuada para que, junto con los algoritmos más óptimos de machine learning se consiga encontrar una solución al problema del incremento del consumo de potencia. Para alcanzar este objetivo se llevaron a cabo simulaciones de potencia usando la herramienta Cadence Joules RTL y se ejecutaron diferentes modelos de machine learning con diferentes configuraciones tratando de encontrar el mejor balance entre precisión en la predicción y bajo consumo de potencia.

Resum

L'execució d'algorismes de machine learning en el núvol comporta alts requisits d'ample de banda, un elevat consum d'energia i problemes en la seguritat de transmissió de dades, entre altres. Diferents sensors arrepleguen data i l'envien a servidors remots per a que siguen processats. Una forma de solucionar els mencionats problemes es executar aquests algorismes en el dispositiu local que no siga tan potent com el remot, però que serà més eficient. Aquest concepte es conegut com a 'Edge Computing'. El TFG s'enfoca en aquesta part de la xarxa, on els algorismes de machine learning s'integren en el sensor.

Un del principals problemes de computació directament en el sensor es l'increment de consum de potència. Per això es important trobar la forma de reduir-ho. L'objectiu d'aquest projecte, que s'ha portat a terme a Fraunhofer IIS a Erlangen (Alemania), es trobar la configuració hardware eficient en potència adequada per aconseguir, junt amb els agorismes de machine learning, trobar la solució al problema d'increment de consum de potència. Per aconseguir aquest objectiu s'han portat a terme simulacions de potència usant la ferrament Cadence Joules RTL i s'han executat diferents models de machine learning amb diferents configuracions per trobar el millor balanç entre precisió en la predicció i baix consum de potència.

Abstract

Having to compute machine learning algorithms in the cloud leads to high bandwidth requisites, energy consumption and security problems among others. Sensors collect data and send this data to remote servers to be processed. A possible way of trying to solve the mentioned problems is to run these algorithms in a local device that won't be as powerful as the remote, but it will be more efficient. This concept is known as 'Edge computing'. This BSc. thesis is focused on that part of

the network, where machine learning algorithms are done integrated in the sensor ASIC.

One of the main problems of the computation directly in the sensor is the increased power consumption. That is why it is important to find a way of reducing it. The aim of this thesis, developed in the Fraunhofer IIS in Erlangen (Germany), is to find a proper power-efficient hardware configuration with optimal machine learning algorithms in order to find a solution to the energy-saving problem. To achieve this objective, power simulations have been performed by using Cadence Joules RTL software and different machine learning models with different configurations have been executed looking for the best accuracy and low-power consumption balance.

To my parents, for their support throughout my bachelor and, especially, during the entire final thesis, even though I had to be away from home.

To my dear friend Carla Morell, for being always there, thank you for your big empathy and your good heart.

To Amparo and Raquel, for having met you on the way here and for all the time we will spend together in the future.

To Filip, Legjenda, Nikolina, Veljko and Vera. Our friendship is the main reason why I decided to take this thesis.

To Jan, for the big support and ideas and for being the best office mate and friend. Keep being the person you are.

To Matthias, thank you for the amazing opportunity of coming to Fraunhofer IIS. I can definitely say that this experience changed the course of my life and has changed the person that I was the first day I arrived to Erlangen in July 2021.

To Sreenivas, for all the hours spent trying to find linking errors and for everything I have learnt working with you.

To Julio, for your patience and support throughout the thesis. Thank you for leading the way and for always being prepared to help me with my doubts and make me learn.

Contents

List of Figures

Table index

List of acronyms used

AI Artificial Intelligence.

ASIC Application-Specific Integrated Circuit.

CPU Central Processing Unit.

HDL Hardware Description Language.

HPC High Performance Computing.

IC Integrated Circuit.

IoT Internet of Things.

MCU Microcontroller Unit.

PULP Parallel Ultra-Low-Power.

rbf Radial Basis Function.

RTL Raster Transfer Level.

SDC Synopsys Design Constraints.

SoC System on Chip.

SPI Serial Peripheral Interface.

std Standard Deviation.

SVC Support Vector Classification.

SVM Support Vector Machine.

tcl Tool Command Language.

Chapter 1

Introduction

Energy-saving problem is one of the main problems faced in the field of Edge AI. Due to the high quantity of algorithms necessary to compute, the hardware has to be constantly working and this consumes power [1]. The main objective of this BSc. thesis is to find the most power efficient hardware/software configuration when executing machine learning pipelines in a specific microcontroller. In order to achieve it power simulations for different configurations are computed using Cadence Joules RTL.

Machine learning is defined as the ability to create algorithms that are learned by the machine itself from external data. By training on finding patterns in huge databases the machine learns how the distribution of data looks like and if new data matches this distribution. The concept of AI appeared for the first time in 1956 on a conference by John McCarthy in the Dartmouth Summer Research Project on Artificial Intelligence. As McCarthy said, the idea was to describe the intelligence on a so precise way 'that a machine can be made to simulate it' [2]. Different machine learning algorithms have been developed to try to achieve the fastest and more accurate prediction. Along the BSc. thesis some of them will be tested for the database used. These 'Machine Learning' pipelines will be used in 'Condition monitoring'[3]. The idea is to monitor the behaviour of industrial machines and to analyse the data collected by sensors, such as accelerators. The analyses will be performed to predict if the behaviour of these machines is correct or if something is failing. Failures in the bearing mechanism are most common when analysing what is not working correctly in these machines[3].

During the first part of the thesis different MCU hardware configurations (peripherals activation/deactivation, cache memory enabled/disabled, fpu working...) are studied while executing different features. For the second part, software is the main objective. First, a study of the behaviour of machine learning pipelines is done. After having clarified all the theoretical concepts, the first pipelines are run and the prediction accuracy with the power consumption is obtained, looking for a balance between both. The main goal is to find the best pipeline configuration, trying to search for proper features and the model with the most optimal hyperparameters.

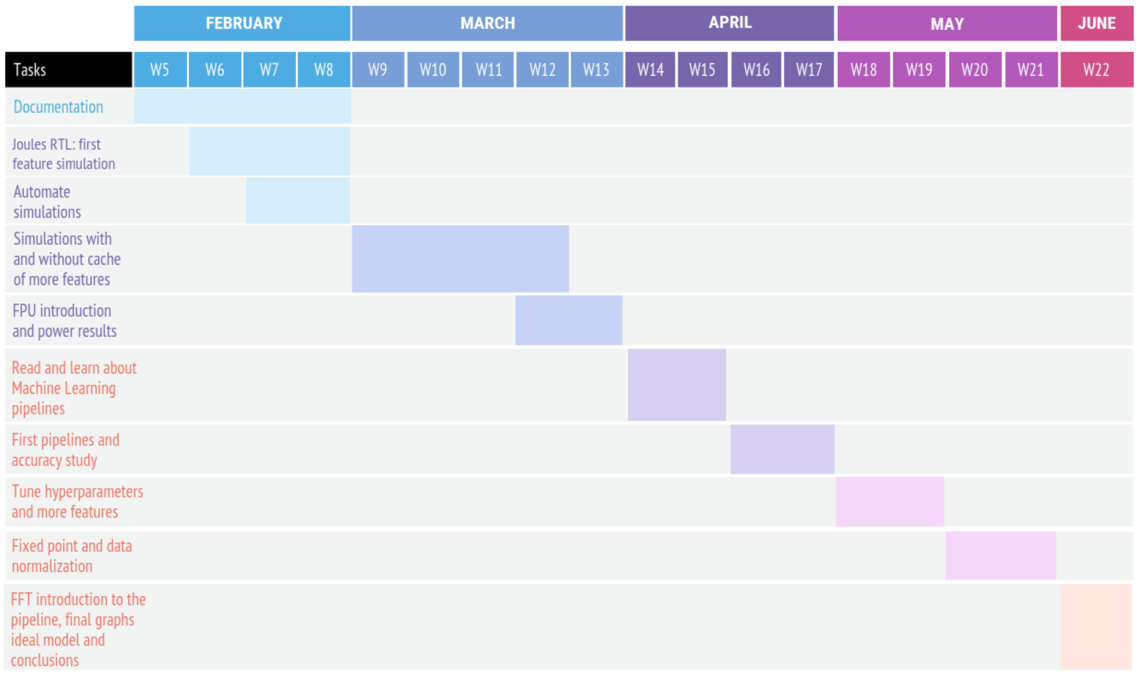
The thesis concludes summarizing the main difficulties reached during the working process and describing which is the best hardware/software configuration after all the research done. Different intermediate objectives are set in order to achieve the main goal of the BSc. thesis:

- Create a good theoretical background in order to understand the structure and working system of KI_Predict microcontroller, Joules RTL software, machine learning models and pipelines.

- Learn to perform simulations and get power and energy data with Joules RTL tool.
- Hardware optimization: energy consumption for different configurations (cache enabled/disabled, FPU connected/disconnected, peripherals on/off...).
- Automate the simulations to obtain fast results.
- Software optimization: Analyse the effect of fixed and floating point, data normalization, different machine learning models and different hyperparameters configuration in terms of power and energy consumption and accuracy of the prediction.
- Study the most common time and frequency domain features and apply them to the pipelines to search for the most optimal.

1.1 Working plan

To achieve the proposed objectives a planning is defined with the necessary tasks to be fulfilled. As can be seen in figure 1.1, the planned tasks are mainly divided in two big groups: one block of hardware goals and another of software. Time is more or less equally divided between these two main parts.



	FEBRUARY				MARCH					APRIL				MAY				JUNE
Tasks	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22
Documentation	■	■	■	■														
Joules RTL: first feature simulation		■	■	■														
Automate simulations			■	■														
Simulations with and without cache of more features					■	■	■	■										
FPU introduction and power results								■	■									
Read and learn about Machine Learning pipelines										■	■							
First pipelines and accuracy study												■	■					
Tune hyperparameters and more features														■	■			
Fixed point and data normalization																■	■	
FFT introduction to the pipeline, final graphs ideal model and conclusions																		■

Figure 1.1: Planning

Chapter 2

Background

The following chapter will introduce the theoretical background required to understand the results obtained during the research.

2.1 KI-Predict ASIC

KI-Predict [4] [5] is a project funded by Germany's Federal Ministry of Education and Research with objective of developing AI systems based on sensors to track industrial production plants. On the thesis the KI-Predict microcontroller energy consumption is tested, thus it is important to understand its architecture.

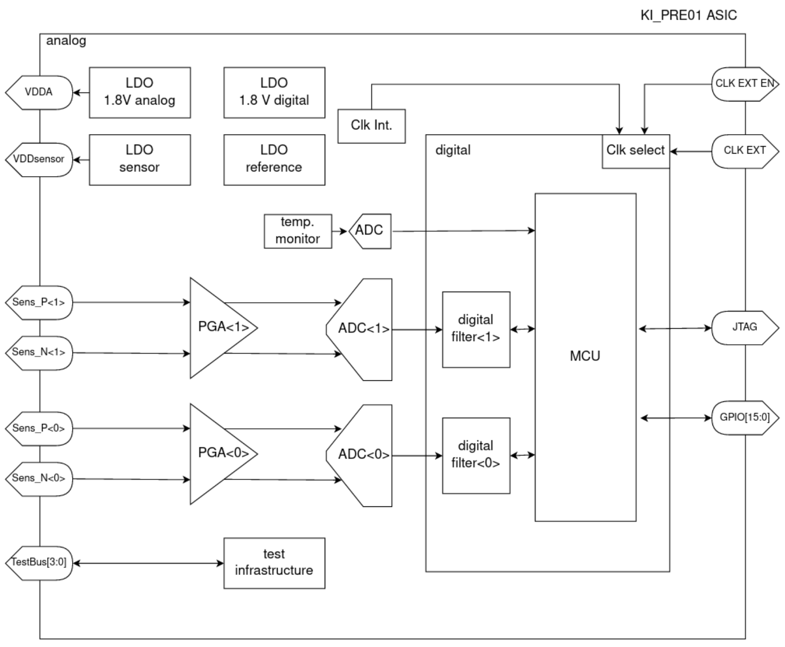


Figure 2.1: ASIC overall architecture divided in analog and digital section [6]

Figure 2.1 is a general overview of the KI-Predict chip. This simulations will be focused on the digital part (figure 2.2). The sensors in figure 2.1 link the real world with the analog section of the circuit and some converters transform the input analog signals into digital to be processed by the digital section. In this section, two main modules can be distinguished: the digital filter and the MCU subsystem. As seen in figure 2.2, the digital filters is the part of the architecture directly connected with the analog-digital converters, and it processes the information that comes directly from the sensors but in its digital form. For the MCU Subsystem, this block is formed by the CPU, memories, peripherals and buses interconnecting them. This subsystem processes the data coming from the filters. For the objective of the thesis the most interesting is the behaviour of the MCU block. MCU architecture is described in figure 2.3.

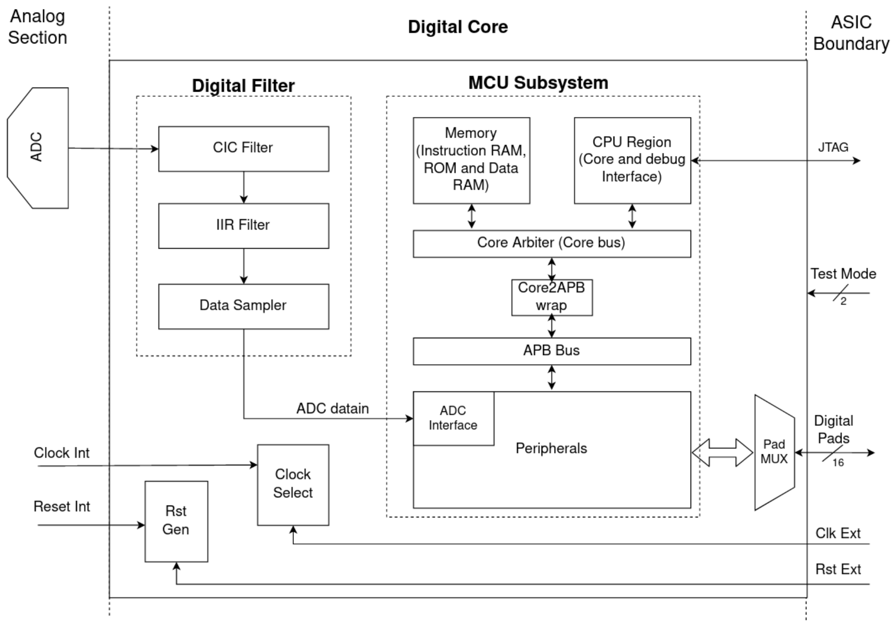


Figure 2.2: Digital core [6]. Its internal architecture is divided in two main blocks: the digital filters and MCU Subsystem block.

The digital domain works with a clock of 10MHz frequency that is obtained from the 30MHz clock that rules the rest of the chip. The MCU is developed mixing open-source hardware projects (as explained in next chapter when talking about the PULP open-hardware platform (figure 3.1)) with blocks specifically developed for this project. It uses Pulpino system structure (see chapter 3). When having a look at the architecture (figure 2.3), it is easy to see that, basically, all the important blocks are linked between them by the OBIXBAR module, which has the role of a bus interconnecting the CPU with the memories, peripherals and hardware acceleration unit. The peripherals are the block that are connected with the digital filters. While the main program is running in the MCU, the peripherals are deactivated.

The hardware acceleration unit (hwac) is currently under development, so it will be disabled for the simulations. This block can be optimised to perform specific operations, such as matrix and fft algorithms, to increase the computation speed. About the memories, the chip has a 48KB SRAM data memory, a 16KB ROM instructions memory, a 2KB VLPSRAM stack memory and a 2KB VLPSRAM cache memory. The stack is a low-power efficient memory and it is usually used for storing variables. When the MCU begins working, the first thing that is executed is the boot code. It is always executed in order to prepare the system when the MCU begins to work.

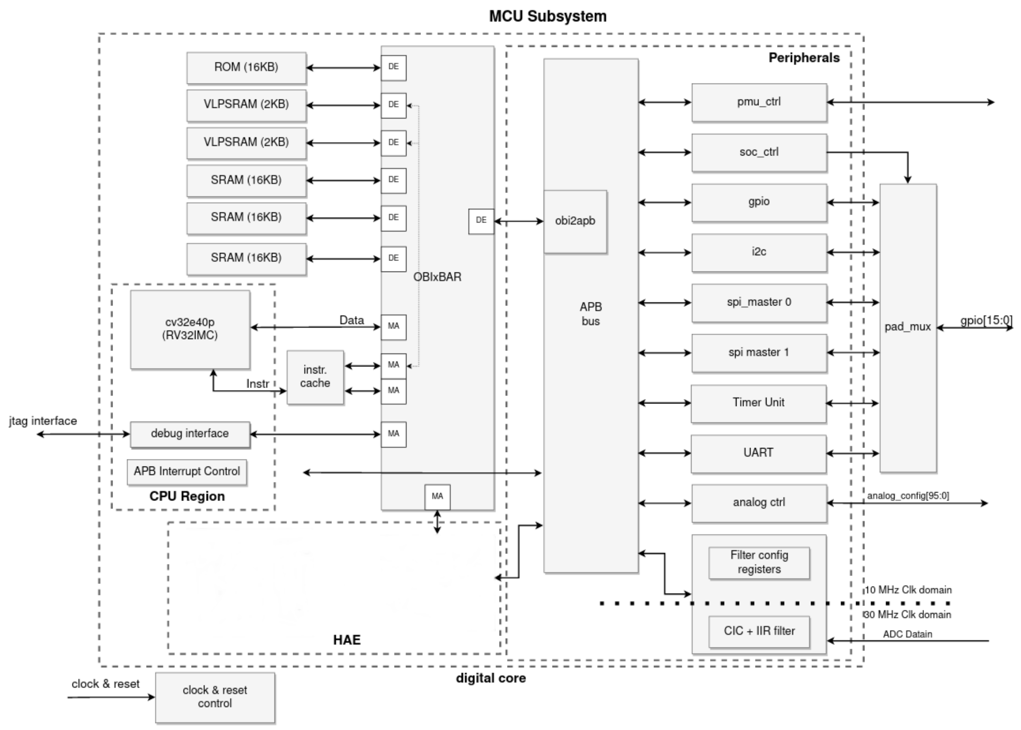


Figure 2.3: MCU Subsystem overall, main focus of the power consumption analyses[6]

This code is always stored in ROM but it is executed from ROM or from SRAM depending on the configuration of the system. It can be checked in the CMake configuration files. CMake is a build system that generates build files for a specific environment by using CMakeLists.txt scripts. In these scripts the different files of the project are specified following a hierarchy. It is in the CMakeLists.txt file of each specific software program where it is possible to check from which memory are going to be executed the main file and the boot code.

```
include_directories(.)
set(hwac.rms_SOURCES rms.cpp)
add_application(hwac.rms_powersim "${hwac.rms_SOURCES}")
```

Listing 2.1: CMakeLists.txt file for rms feature

Code 2.1 is an example of CMake file. The ”set” command sets the rms.cpp file as the main file of the rms software under the name hwac.rms_SOURCES and the ”add_application” command is the one that indicates from which memory is the software going to be executed. This command could have three formats:

- add_boot_code: everything, boot code and main program are executed from ROM.
- add_application: everything, boot code and main program are executed from SRAM.
- add_nvram_application: boot code is executed from ROM and main program from SRAM.

For the case of this thesis the add_application is used and all the code is going to be executed from the SRAM. Referring to the cache memory, 48KB of the SRAM are going to be cached.

The working system is the following: the CPU wants to access to a memory address but first the instr.cache block (Figure 2.4) checks in an internal address memory if this address has already been accessed and saved in the cache memory. This action is called detecting a HIT (it is saved in cache memory) or MISS (it is not saved yet) and it is performed in the cache_handler_s1.sv block. If it is a HIT then the instruction is read from the 2KB SRAM cache memory (obi_cache bus). If it is a MISS the instruction is read from the 48KB cached memory (obi_mem bus) and then saved in the cache memory. For the first power simulations computed the cache was disabled, so all this process was not computed and the address was bypassed directly from CPU to SRAM memory.

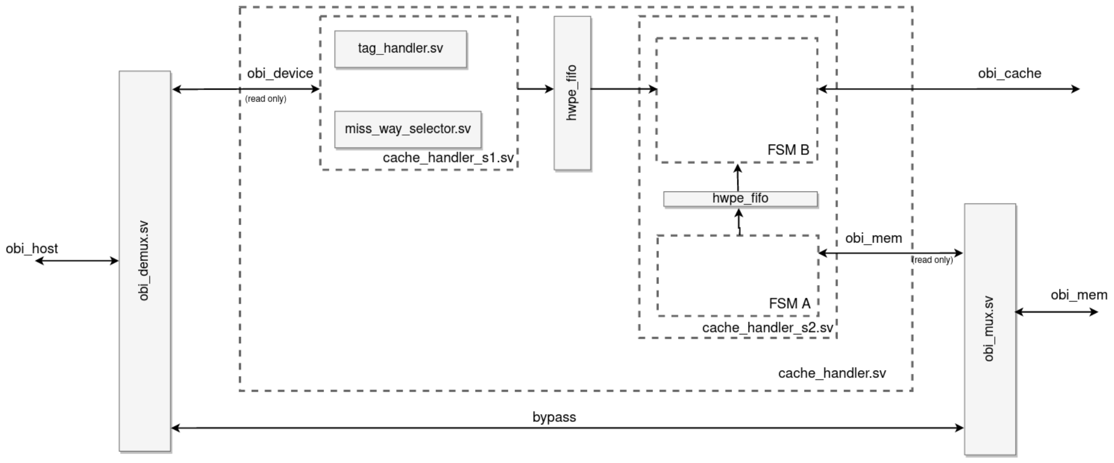


Figure 2.4: Detail architecture of instr.cache block [6]

About the cache memory, it is a 2-cache-ways of 2KB memory. This means that the lines of the memory are splitted into multiple sets. In our case, 2 sets. There are 8 words per line and each word is 4bytes, so just computing the maths, for a 2KB memory there are 64 lines. As it is working with 2 ways, there are 2 sets each of them with 32 lines each. Every address of the cached memory is mapped to a particular set.

The memory that saves the addresses that have already been accessed and whose instructions have been saved in cache memory is called ”Tag memory”. Each address of this memory indicates the following:

- The address where it is saved at cache (2KB cache memory).
- Which of the two sets it is caching.
- The line on the set.
- The word accessed.

2.2 Joules RTL Power Solution

There are many factors that have an impact in the total amount of power consumed but the main ones are the architecture of the IC, the implementation (libraries, constraints, clock-tree...) and the stimuli (input signals in the MCU - executed software). In order to be able to measure the total amount of power taking into account these three factors, Joules RTL is used. Joules RTL is a product from Cadence focused on RTL power analysis. RTL means Register-Transfer Level. It makes reference to how a HDL (Hardware Description Language) defines the way data is modified and transmited from register to register. In the case of the design in the thesis, the HDL used is Systemverilog. Joules RTL tool makes use of .tcl files to do the power analyses of the design. Figure 2.5 shows the structure of the whole power analysis process.

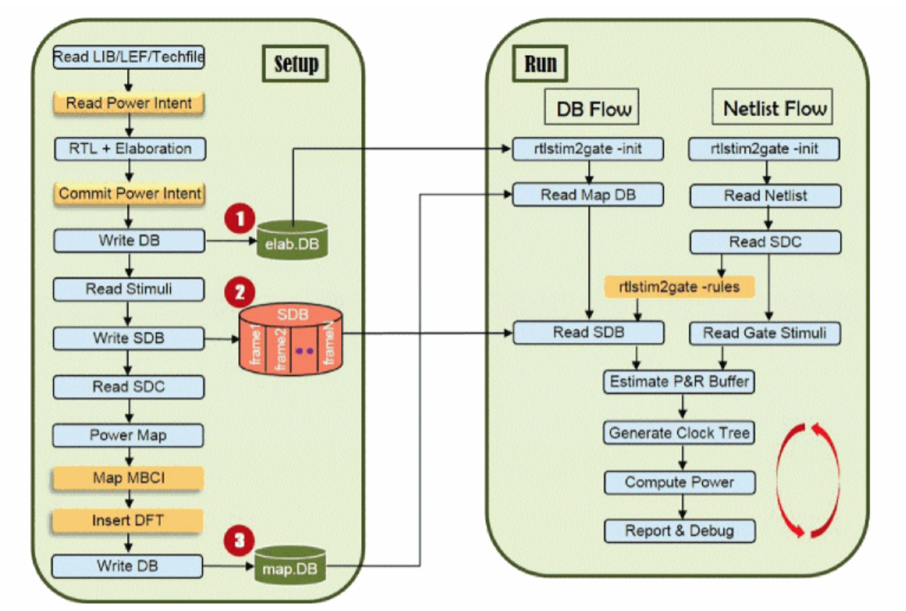


Figure 2.5: RTL Power Flow [7]

To summarize the power flow in figure 2.5: First, as seen in the setup section, the libraries, the power intent and the design are read and a design database is elaborated. The power intent is the set of different power domains in which the architecture is divided. A power domain is a set of gates powered by the same power supply. Secondly, the stimulus are read from the .vcd file and a stimulus database is created. After, the constraints (SDC file) are read (clock, external, power, environmental...) and synthesized and the power is mapped. Another database (map.db) with these data is created and with this, the set up is finished. For running the simulation the databases are read, the same as the netlist, constraints and stimuli. Then, the clock tree is generated and the power is computed. From here, power reports are obtained and all the necessary information can be read, for instance, activity reports by frame or hierarchy or time-based power profile by stimulus, among other examples.

2.3 Machine learning pipelines

In the thesis, the developed machine learning pipelines are used for ’Condition monitoring’. By processing the data, the pipeline will predict if the bearing mechanism of a motor is working correctly or not.

Figure 2.6 shows the structure of a pipeline. Different features are extracted from the data

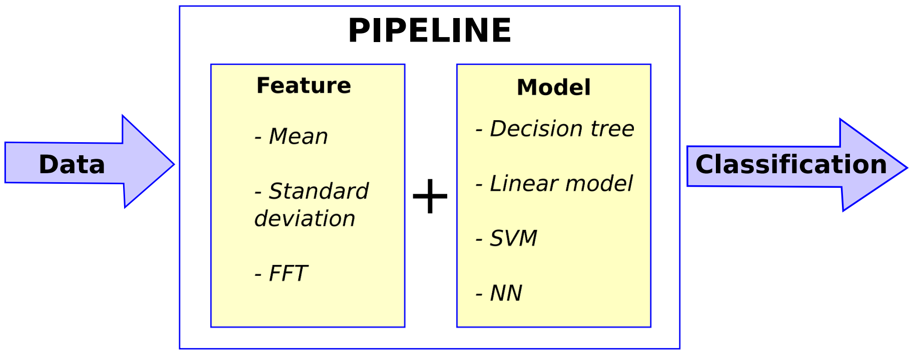


Figure 2.6: Machine learning pipeline

set samples. This information is used in a machine learning model in order to classify them. A common example are handwritten digits. The pipeline has to predict what digit corresponds to a specific image. In the proposed case, the data of the acceleration measures taken by the sensors are provided and the output is trained to predict if the system is working correctly or not.

There are two main techniques defined in machine learning[8] [9]:

- **Supervised learning**: Specific data sets are used to train the module. This way the network learns from this data set.
- **Unsupervised learning**: The data set output is not labeled. The net learns by discovering patterns in data.

Two types of problems can be faced for each of the two techniques:

- **Classification**: Data is separated into specific discrete categories, for example, the numeric digits. This is the technique for the data set in this thesis.
- **Regression**: Referring to continuous data, as for example, the height of a person.

After obtaining the features, a machine learning model is chosen [10]. A model is what is learned by a machine learning algorithm. By executing this algorithm a model is obtained comprised of, for instance, a vector of coefficients or a graph structure with matrices of weights. These structures are used to do the classification or regression. In this thesis, the proper combination feature plus model is found in order to achieve the most accurate and less power consuming configuration. Here are some examples for different models [11] used for classification.

- **Decision Tree Classifier**: A classifier based on if-else conditions. One of the main advantages is that it requires little data preparation. The bigger the complexity of the tree, the bigger the accuracy of the model. In order to understand, an example of tree can be seen in figure 2.7.

  Using the features extracted from the data, the tree classifies the input data and decides which label has to be assigned. For instance, if the bearing mechanism is not working properly, it decides according to the data the diameter of the damage between 4 possibilities: 0.007", 0.014", 0.021" and 0.028". In figure 2.7 each colour for each ball represents a different

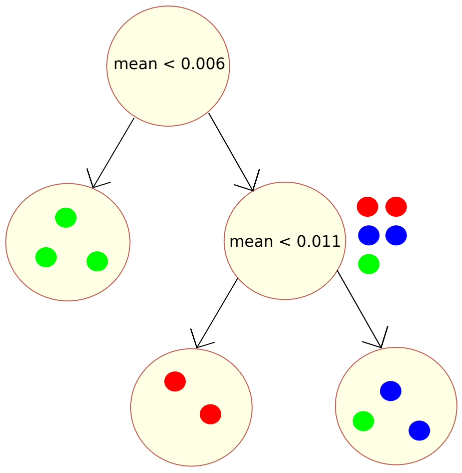


Figure 2.7: Decision tree

class. If a high accuracy on the prediction is achieved the entropy in every branch will be small. In the example in figure 2.7, although the model is pretty accurate still for those conditions a green ball is not correctly labeled. So the aim will be to improve the model.

- **Random Forest Classifier:** One of the main disadvantages of 'Decision tree' model is that is extremely sensitive to the training data. So, the slightest change in it can change the whole tree. The model Random Forest Classifier can be used to reduce the sensitivity.

  First, the training data is randomly grouped in 'x' number of sets, what is called 'Bootstrapping'. Also, 'x' number of trees are created for each different training set. The features used for each tree are randomly chosen. This reduces the correlation between trees. Finally, a prediction is done by each tree and the final prediction is the one obtained the most.

- **Support Vector Classification**: This algorithm classifies data in two or more subsets depending if one or more SVM (Support Vector Machines) are used. An example would be when having to decide if the motor is working properly or not depending on the value of the mean of the data (Figure 2.8). Support vectors will influence the position of the threshold,

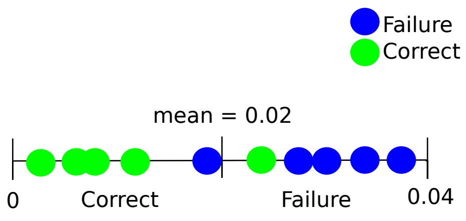


Figure 2.8: SVC example

also called hyperplane. Support vectors are the closest data values to the hyperplane. Soft margin is the distance between these vectors and the hyperplane. By cross validation, which means by trying different thresholds and analysing its prediction, the best hyperplane is set. This hyperplane is the one that minimises the amount of misclassifications by keeping the variance as low as possible. Variance is defined as the prediction accuracy between the training set and the testing one. In case of having 2 dimension data, like taking into account mean feature and max feature, the support vector classifier will be a line; in case of 3 dimensions, a plane; in case of 4, a 3D figure, and 5 or more on it is difficult to represent it. But sometimes is not so easy to divide data into two classes by a threshold. In figure 2.9 we can check an example of this problem.

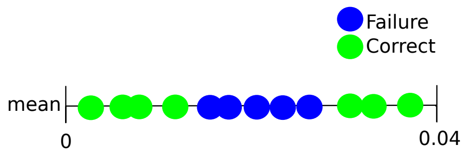


Figure 2.9: SVM example

Here is where support vector machines are needed. So another dimension is introduced, for example, if x axis is the value of mean, y axis will be x^2. By doing this a line can be set as a threshold to separate the features correctly. In order to find the optimal Support Vector Classifier, Support Vector Machines use Kernel Functions, which are algorithms that help on the classification of data that follows different patterns. Even though a SVM can not be well represented in higher dimensions visually, it can effectively separate very high dimensional spaces. Therefore, it has been one of the most popular ML-Algorithms before the rise of Neural Networks.

- **OneClassSVM**: This Support Vector Machine model is used for anomaly detection. The training data set contains just one data class. So by training the model, a boundary is created having in it all the training data. At test step, all data that fits inside the boundary is considered to be correct and all the data out of the boundary is the one that presents an anomaly.

Other important concept to have into account is the hyperparameters. These parameters specify some characteristics of the model the pipeline is using. So, playing with the value of these, the most accurate model will be found. Here, some of the most important for the models explained:

HYPERPARAMETERS:

- **SVC:**
  - **Kernel:** It indicates the type of hyperplane that will be used to separate the two categories in which data is classified. These hyperplanes can be linear, polynomial, radial basis function, sigmoid and precomputed. By default radial basis function is selected.
  - **Gamma:** When talking about ’Radial Kernel’, this one is defined as:

    $$k(u, v) = e^{-\gamma(a-b)^2}$$

    ’a’ and ’b’ are two vectors and γ configures the sensitivity towards the differences between ’a’ and ’b’. Having a big γ can lead to overfitting, because it is so well adjusted to the training data that when introducing new data for validation the model classifies it wrong.
  - **C:** It sets a balance between having a correct classification and setting a decision boundary not really strictly attached to the training data. As well as it happens with gamma, for a high value overfitting can be produced.
  - **Degree:** This parameter is just used if the kernel selected is polynomial. It will set the degree of the polynomial.

- **OneClass SVM:** For this model the hyperparameters are moreless the same as for SVC, with the only difference that instead of using C parameter, ν is the one being used with the same purpose. While C range goes from 0 to infinity, ν goes from 0 to 1, setting the quantity of misclassifications allowed while training the pipeline.

- **Decision Tree:**

  – **Criterion:** It measures the amount of misclassifications in every split of the tree. It can get the values of 'entropy' or 'gini', although by default 'gini' is selected. 'Gini' factor indicates the probability of picking up a random sample and being labeled incorrectly. It is defined as:

  $$gini = 1 - p_1^2 - p_2^2$$

  From the equation, p_1 and p_2 are the probabilities of data being classified in one branch or in the other respectively. The smaller the value, the better. That's why, if we have a look at Decision Tree in Appendix A in the final nodes the 'gini' gets value 0.

  Entropy index can have a value between 0 and 1, while gini from 0 to 0.5.

  – **min_impurity_decrease:** Every time that a node of the tree is going to be split in two branches, this parameter sets the amount of misclassifications allowed for it to be possible. By default is 0.

  – **max_depth:** It sets the maximum possible splits on the tree.

  – **max_features:** It sets the number of features that will be taken into account when doing the splitting.

  – **min_samples_split:** The minimum number of samples required to split a node. By default if it has value 2.

- **Random Forest:** The hyperparameters are almost the same as for decision tree, as this model is a composition of different decision trees. For this reason, one important hyperparameter is **n_estimators**, which sets the number of decision trees chosen to form the 'forest'. By default this parameter will have value 100.

2.4 The data set

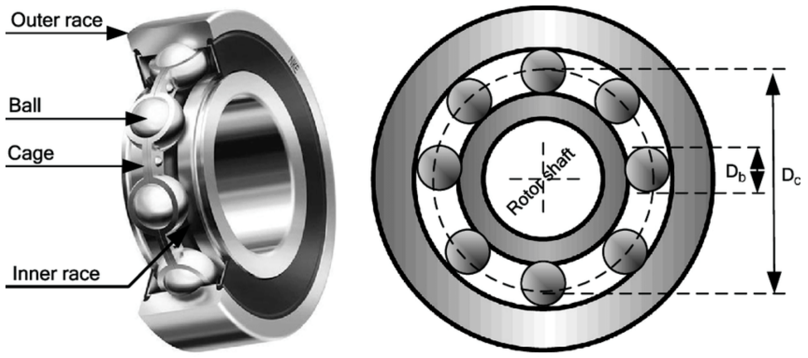


Figure 2.10: Bearing mechanism [12]

To train the pipelines the data set used is the CWRU data set [13] (Case Western Reserve University in Ohio). By using two sensors located on the 2 HP Reliance Electric motor in 2 axis positions, acceleration data is registered. So, the labels used for the classification indicate not only if it the bearing mechanism works or not, but the characteristics of it, such as the fault diameter, the motor load in horsepower, the inner race (figure 2.10), the ball or the outer race position relative to the load zone.

2.5 Features and pipeline model

For the features power consumption analyses, the C++ code run in the MCU is taken from a features library already developed by part of the team from Fraunhofer IIS working in the Nuremberg section. It is only necessary to include it in our main code. The most common statistical features in the time domain are described here [14]:

$$X_m = \frac{\sum_{n=1}^{N} x(n)}{N} \qquad X_{std} = \sqrt{\frac{\sum_{n=1}^{N}(x(n)-X_m)^2}{N-1}} \qquad X_{root} = \left(\frac{\sum_{n=1}^{N}\sqrt{|x(n)|}}{N}\right)^2$$

$$X_{rms} = \sqrt{\frac{\sum_{n=1}^{N} x(n)^2}{N}} \qquad X_{peak} = max|x(n)| \qquad X_{skewness} = \frac{\sum_{n=1}^{N}(x(n)-X_m)^3}{(N-1)X_{std}^3}$$

$$X_{kurtosis} = \frac{\sum_{n=1}^{N}(x(n)-X_m)^4}{(N-1)X_{std}^4} \qquad X_{crest} = \frac{X_{peak}}{X_{rms}} \qquad X_{clearance} = \frac{X_{peak}}{X_{root}}$$

$$X_{impulse} = \frac{X_{peak}}{\frac{1}{N}\sum_{n=1}^{N}|x(n)|}$$

Between these features only the ones providing the highest accuracy will be used in the final pipeline. In order to find which ones are the best, a research process will be taken. During this research an error in the definition of features 'impulse', 'crest' and 'clearance' was introduced. This error lead to a higher accuracy, so it was decided not to fix it. The error came from not working with the absolute value of the input data while it should be following the proper definition of the features. These are the 'wrong' definitions that are the formulas used in the research:

$$X_{crest} = \frac{X_{maximum}}{X_{rms}} \qquad X_{clearance} = \frac{X_{maximum}}{X_{root}} \qquad X_{impulse} = \frac{X_{maximum}}{X_m}$$

Apart from the temporal features there are also features in the frequency domain. To work with them it is necessary first to obtain the fft. These are some of the most common statistical features in frequency domain. They correspond to the mean frequency, variance, frequency center and root mean square frequency.:

$$X_{mf} = \frac{\sum_{k=1}^{K} s(k)}{K} \qquad \sigma^2 = \frac{\sum_{k=1}^{K}(s(k)-X_{mf})^2}{K-1} \qquad X_{fc} = \frac{\sum_{k=1}^{K} f_k s(k)}{\sum_{k=1}^{K} s(k)} \qquad X_{rmsf} = \sqrt{\frac{\sum_{k=1}^{K} f_k^2 s(k)}{\sum_{k=1}^{K} s(k)}}$$

When working on the creation of the pipeline, the features used for training the model were computed by using the commands from the numpy library from Python, except for the 'kurtosis' feature that the scipy library is used. And for the model selection another tool was needed, scikit-learn. Scikit-learn is a machine learning library for python language. It began as a project from the Google Summer of Code 2007 developed by David Cournapeau and has become the most used tool when working with machine learning in Python. Apart from the models it includes other tools such as the confusion matrix representation, which was really useful to see visually the accuracy of the pipeline.

Chapter 3

State of the Art

The main objective of the thesis is to achieve the best power efficient combination of machine learning pipeline and hardware configuration. This is, nowadays, one of the main problems faced in the field of Edge AI and the main goal of the project AutoML.

AutoML ASIC [15] is a research project developed at Fraunhofer Institute for Integrated Circuits in the field of Artificial Intelligence, specifically focused on the problem of computing Machine Learning algorithms energy-efficient for embedded systems. It tries to solve two main problems: developing software algorithms that take longer time periods to drain the battery, and having the appropriate hardware configuration in order for the algorithms to be performed better and faster. This problem is difficult to solve because the tools used to compute the evaluation of the hardware are really slow, taking around 1 hour per evaluation. So when training the network, if there is a dataset of 10000 samples, it will take 1.1 years to complete the power consumption evaluation (evaluation of not just one but different models), which is unthinkable [15]. The main objective is to reduce this time to get a fast prediction.

The other goal of the AutoML project is the development of machine learning pipelines for 'Condition monitoring' [3]. Sensors obtain data of industrial machines and these data are used in order to check if there is some anomaly in its state or behaviour. The amount of data is processed by the pipeline and a prediction is obtained. The ultimate goal will be to automatically choose the best pipeline steps.

On this thesis, one of the first tasks performed is the evaluation of the power consumption during the execution of some of the feature extraction programs. Instead of executing it in the SoC used at AutoML they are executed in the KI-Predict one. Why to use this MCU for the simulations has its reasons. The AutoML project is focused on finding the most efficient machine learning algorithms. These are computed in a commercial chip, but when computing the simulations it just allows to get the power and energy values of the total MCU, not being possible to have a look to the consumption of the different parts that compose it. That's why, as we have the KI-Predict project MCU, it is possible to use this to compute simulations and get the results for the different parts of the architecture.

KI-Predict uses open source hardware, the MCU Pulpino architecture, in order to be able to apply it to different applications on industry. To understand what is Pulpino and why is it use it is necessary to understand first what is PULP.

PULP [16] is a project developed by the ETH Zurich and the University of Bologna in order to satisfy the demands of IoT applications. The acronym means Parallel Ultra Low Power. These kind of architectures allow to meet the requirements without exceeding the power of a few mW. There are several PULP configurations depending on the application. By analyziing the following schema it will be easier to understand why Pulpino was chosen for the KI-Predict project.

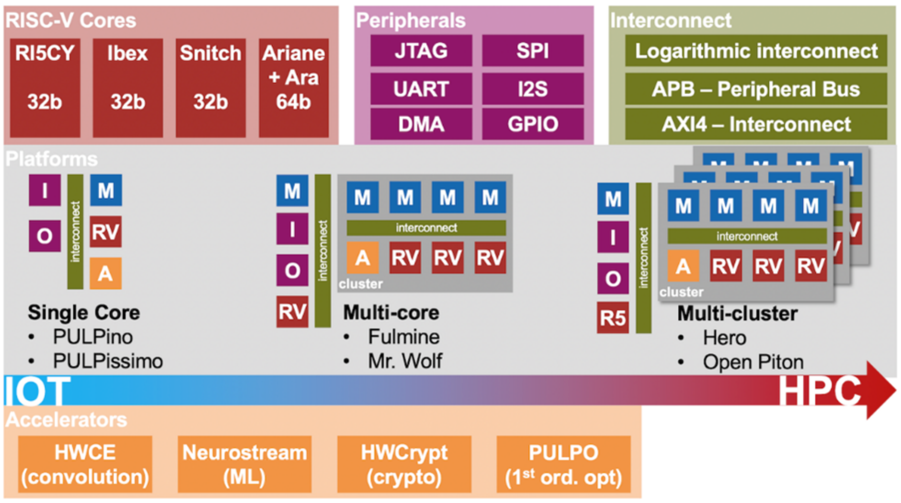


Figure 3.1: PULP platform [17]. In the center part of the image, different microcontrollers architectures for different fields of application are specified (from IoT application to HPC). At the top and bottom, different cores, peripherals, buses and accelerators used in these architectures.

As it can be see in figure 3.1, Pulpino is a single core MCU, what makes it more attractive for the project, as it consumes less power, the objective pursued. Its structure (figure 3.2): a single core, buses of class AXI4 interconnection and APB, RAM memory for instructions and data, ROM memory for running the boot code and several peripheral units. In figure 3.1 it is possible to see which cores are more used in the field of Internet of Things (normally single core) and which ones in High Performance Computing, usually with multiple clusters. When talking about the cores used in these microcontrollers, they are different depending on the application field. In the case of KI-Predict, it uses a RI5CY core. Here, a study into the RISC-V cores [18] used in the field of IoT to understand why it was chosen.

As explained in the article of Pasquale.D.Schiavone, et al.[18], the main RISC-V cores are Riscy, Zero-riscy and Micro-riscy. The first one is optimized to target DSP algorithms and to be integrated in a cluster of processors and it is the one used in KI-Predict, as for DSP applications it is the most energy efficient. Zero-riscy core is optimized to target arithmetic-control mixed applications and Micro-riscy is focused on pure-control applications. Apart from studying the different IoT cores, going up one level in the hierarchy, it is also important to have a look at the different MCUs in the field of Edge IoT.

Mr.Wolf [19] is an energy-precision scalable PULP SoC for IoT Edge Processing. One of the main characteristics is that this MCU is multicore, which improves the performance due to the high amount of computations to be done for always-on IoT end nodes. A good question would be why the KI- Predict is single core if it is also an IoT Edge MCU. The answer is that, in order to decide if it is better to work with single or multicore the main objective of the project must be

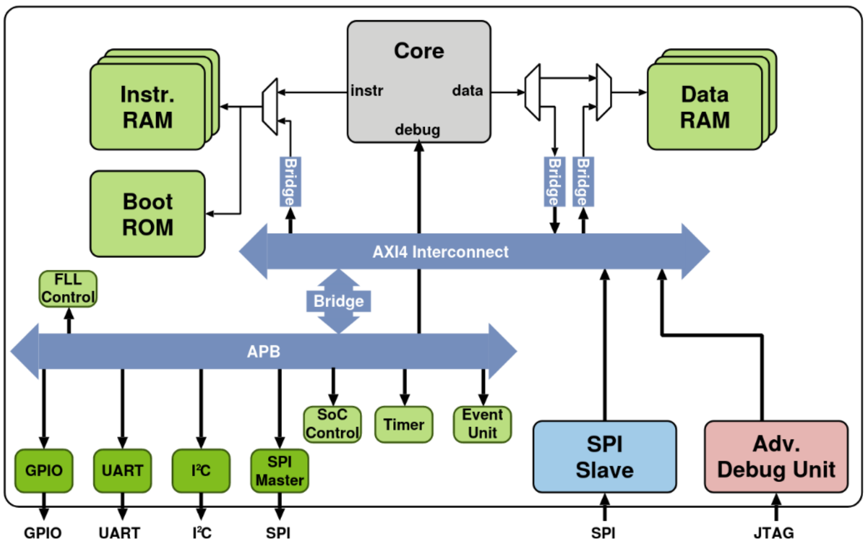


Figure 3.2: Pulpino single core MCU architecture[18]

considered. The principle objective of KI-Predict is the low energy consumption, and multicore systems consume more than single core. For Mr.Wolf this is not the main goal and multicore performs better and faster the algorithms. Moreover, Mr.Wolf uses the Zero-riscy core because power analyses for always-on applications reveal that this is the best power-efficient core for the case.

In summary, power consumption reduction will be the center around which the rest of tasks will revolve.

Chapter 4

Hardware optimization

The field of application of the Machine Learning pipelines of the AutoML project is condition monitoring. In order to classify the different input data under determined labels, some features need to be extracted and analyzed. Some of the studied features are already mentioned in Chapter 2. After training the model with a huge amount of data, the system is able to recognize if the machine that is being monitored is working properly or not.

As a first step in this thesis, power and energy analyses of some of these features are computed. The first results to get are from features mean, median, abs, grad and diff. Apart from Joules RTL tool, QuestaSim tool will be used to check the wave forms behaviour.

4.1 Automation of the process for power simulations

In order to face how to compute the simulation of the five features consecutively, the idea of automating the whole process is considered. Each feature is called from a different main file. A .csh file is created containing the necessary commands to add the paths of the different tools used in the terminal, set up the make commands and libraries, create the stimulus .vcd files for the different features or initializing Joules. Another problem is faced. The objective is to extract the power values of the time frame where the feature is running. In order to do this it is necessary to get the time values for the beginning and end of this frame.

The first option to face this is accessing to the generated .elf file and searching in the assembly code the part where the feature program begins to be executed. Then, after checking which address value corresponds to the first line of this code, the idea will be performing the simulation at QuestaSim and checking the wave forms in the moment when the CPU asks for this address to be read. The same for the ending address. Despite it could work it is really complicated trying to identify in the code the precise moment for every different feature, so this idea is discarded.

The second option is to check the SoC control unit. This module is in charge of indicating what other modules are running. The idea is to check the register STATUS. Bits 16 and 17 indicate respectively if the simulation is still running or not and if it has succeeded or not. Bits from 18 to 29 are just not used, so the idea is to use a bit from this range to activate it when the simulation begins running and to deactivate it when ended. Then, the testbench will be modified to save the time when this signal changes value. Instead of performing this idea, finally other option is developed.

Instead of having to give a value to a register bit it will be easier just checking the SPI clock, the peripherals clock. This clock is always disabled when executing the code of the program (figure 4.1). It is also enabled again when it is necessary to execute a ”printf” command. So, the only thing needed with this option is just detecting in the testbench when the clock stops running and saving the times.

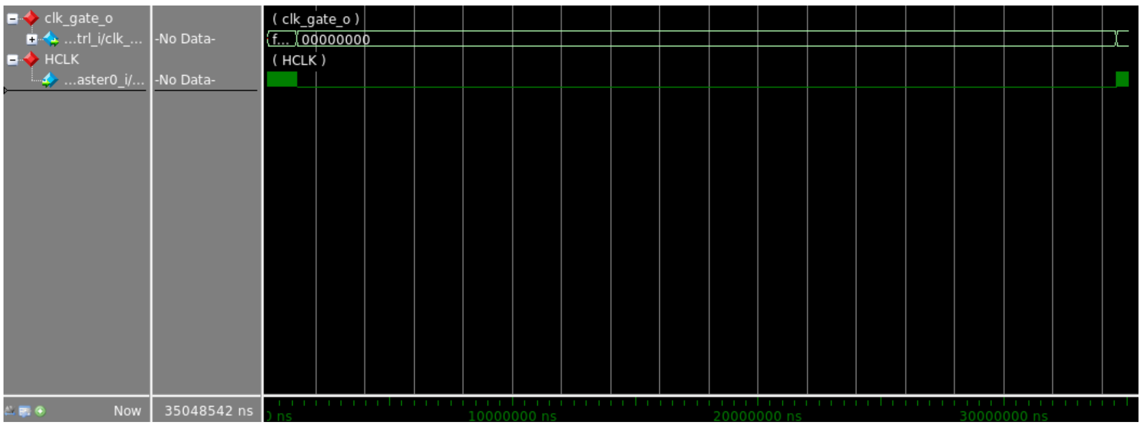


Figure 4.1: SPI clock signal

4.2 First simulations

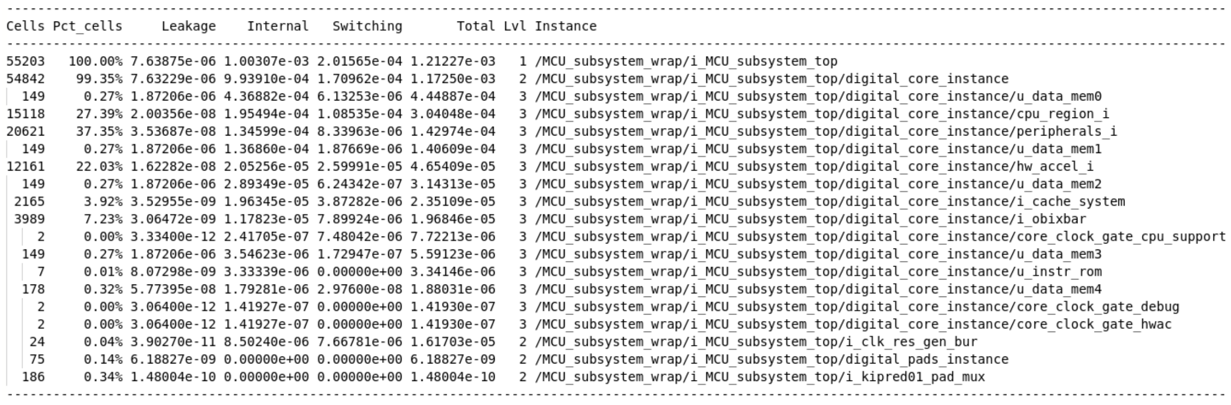
The first time that the features' files are going to be simulated a problem appeared. After compiling the design, when running the simulation a message is displayed indicating that an overflow has been produced because the memory is not big enough to run all the features. So, the SRAM memory has to be changed after this to a 254KB, keeping the 2KB stack memory and the 2KB cache. After fixing this problem it is possible to continue with the power analyses.

Joules allows to obtain power values but not energy reports. These last reports will have to be generated by a LibreOffice Calc table. The power simulation results during certain time will be analyzed by frames and by hardware design hierarchy. As an example, in figure 4.2 the results for feature median with cache disabled using a data array of 512 samples are displayed. That is the way power consumption results are displayed when analysing the hardware hierarchy consumption.

For a better understanding, the relation between the main instances and the design is explained:

- u_data_mem4 = Cache memory
- u_data_mem0-3 = SRAM memory (instructions and data).
- u_data_mem3 = stack memory
- i_cache_system = instr.cache block

Before comparing the results with the other features a problem with SRAM data interpretation is faced. The total power consumed by the different instance modules that form it are displayed, but it is not possible to know which modules contain the instructions and which the data. So first, some modifications in the hardware design code are made in order to match specific instances



```
------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------
55203   100.00% 7.63875e-06 1.00307e-03 2.01565e-04 1.21227e-03   1 /MCU_subsystem_wrap/i_MCU_subsystem_top
54842    99.35% 7.63229e-06 9.93910e-04 1.70962e-04 1.17250e-03   2 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance
  149     0.27% 1.87206e-06 4.36882e-04 6.13253e-06 4.44887e-04   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_data_mem0
15118    27.39% 2.00356e-08 1.95494e-04 1.08535e-04 3.04048e-04   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/cpu_region_i
20621    37.35% 3.53687e-08 1.34599e-04 8.33963e-06 1.42974e-04   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/peripherals_i
  149     0.27% 1.87206e-06 1.36860e-04 1.87669e-06 1.40609e-04   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_data_mem1
12161    22.03% 1.62282e-08 2.05256e-05 2.59991e-05 4.65409e-05   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/hw_accel_i
  149     0.27% 1.87206e-06 2.89349e-05 6.24342e-07 3.14313e-05   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_data_mem2
 2165     3.92% 3.52955e-09 1.96345e-05 3.87282e-06 2.35109e-05   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/i_cache_system
 3989     7.23% 3.06472e-09 1.17823e-05 7.89924e-06 1.96846e-05   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/i_obixbar
    2     0.00% 3.33400e-12 2.41705e-07 7.48042e-06 7.72213e-06   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/core_clock_gate_cpu_support
  149     0.27% 1.87206e-06 3.54623e-06 1.72947e-07 5.59123e-06   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_data_mem3
    7     0.01% 8.07298e-09 3.33339e-06 0.00000e+00 3.34146e-06   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_instr_rom
  178     0.32% 5.77395e-08 1.79281e-06 2.97600e-08 1.88031e-06   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/u_data_mem4
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/core_clock_gate_debug
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   3 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_core_instance/core_clock_gate_hwac
   24     0.04% 3.90270e-11 8.50240e-06 7.66781e-06 1.61703e-05   2 /MCU_subsystem_wrap/i_MCU_subsystem_top/i_clk_res_gen_bur
   75     0.14% 6.18827e-09 0.00000e+00 0.00000e+00 6.18827e-09   2 /MCU_subsystem_wrap/i_MCU_subsystem_top/digital_pads_instance
  186     0.34% 1.48004e-10 0.00000e+00 0.00000e+00 1.48004e-10   2 /MCU_subsystem_wrap/i_MCU_subsystem_top/i_kipred01_pad_mux
------------------------------------------------------------------------------------------------------------------------------------
```

Figure 4.2: Example of power consumption results for 'median' feature

with the instruction data parts: 'u_data_mem0' and 'u_data_mem1' will be instruction memory while 'u_data_mem2' and 'u_data_mem3' will be data memory. After this modification, the power consumption of some other features is obtained and compared.

4.3 First energy study

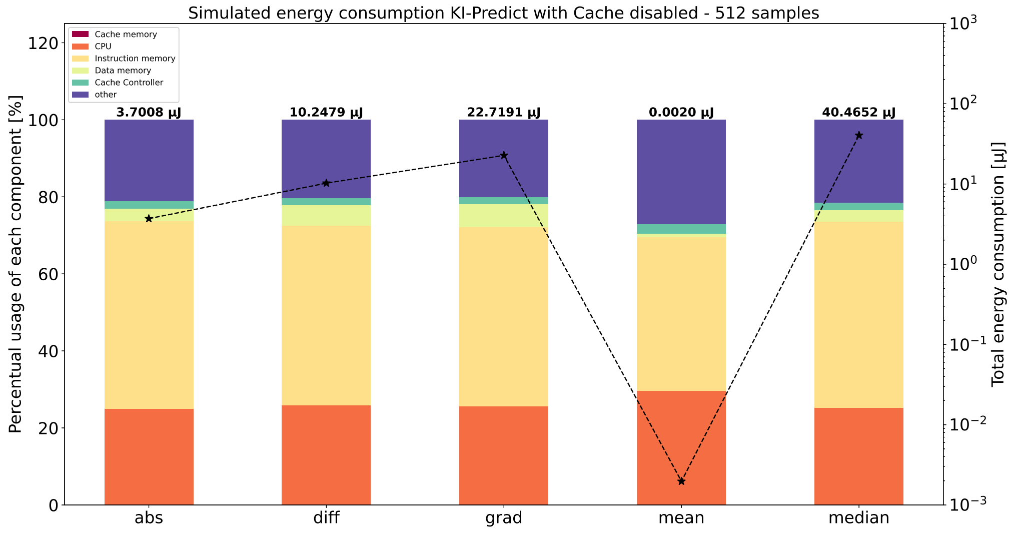


Figure 4.3: Energy consumption (axis y - right) of the features (axis x) and architecture modules consumption proportion (axis y - left)

Here a brief description of how to interpret the results in the graph (figure 4.3). On the left 'y' axis of the figure, the percentage of energy consumption per component respect to the total architecture is represented by using the bars. On the right 'y' axis, the real value of energy consumed in logarithm units is described in μJ by using the black dots line. The total value of energy consumed for each feature ('x' axis) is indicated at the top of the bars. By analyzing the results it is observed that the peripherals (included in the 'other' section in the graph) are consuming a high amount of power when supposedly they are not working. Just the SoC control module should be running because it is the one that defines the clock gates and decides which other modules are running. So the power should be mostly consumed by this part of the design. After analysing the hierarchy (for details of the hierarchy consumption check appendix C) it is checked that the block that consumes the most quantity of power in the peripherals is, surprisingly, the "apb_adc_int_i" block from which "i_dig_filt_wrap" consumes the most. This is the one that controls the digital filters of the system. The filters are running at 30 MHz and not at 10 MHz like other parts of the design. The 30 MHz clock is not clock gated, that's why it has a high power consumption. The peripherals hierarchy are modified in order to disable the digital filters and reduce the power consumed. By doing this, the power is considerably reduced (figure 4.4, 13% of reduction if we look at feature 'abs').

Other remarkable aspect from figure 4.3 is the fact that 'mean' feature consumes a really low quantity of energy in comparison with the other features (the figure is in logarithm units, the black line represents the consumption). At the beginning we give an explanation to this, although later on it will be explained why this conclusion is wrong: it has the simplest algorithm. Due to this fact, it also takes less time to finish it. As the time interval is shorter, although the power consumption values are quite similar to the other features, when obtaining the energy a big change is observed because energy is defined as

$$Energy(J) = Time(s) * Power(J/s).$$

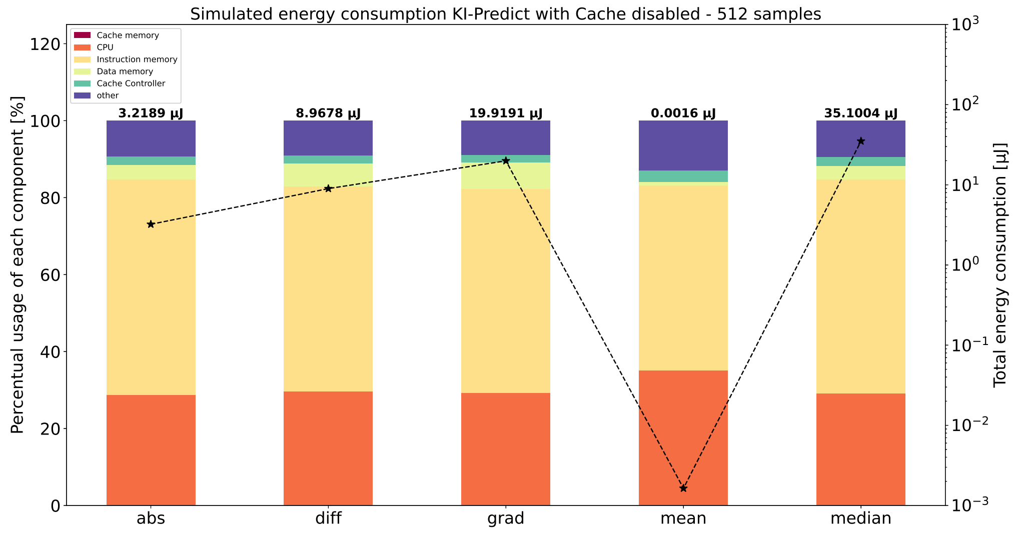


Figure 4.4: Energy consumption after disabling the digital filters - cache disabled

It is possible to check in figure 4.4 that the power consumed by the peripherals was reduced with this change. In this point the cache is be enabled and the simulations, computed. At first, as the instruction memory is formed by mem0 and mem1 instances, the mem1 is the one selected to be cached. But by analyzing the power results it is seen that depending on the feature mem0 or mem1 should be cached. Not seeing any pattern it is concluded that this decision is not predictable and depends on the compiler, so before enabling the cache it should be checked which instruction memory is being used the most. This one will be the cached. After changing this, the results are the ones seen in figure 4.5, being reduced the energy by a 33% value.

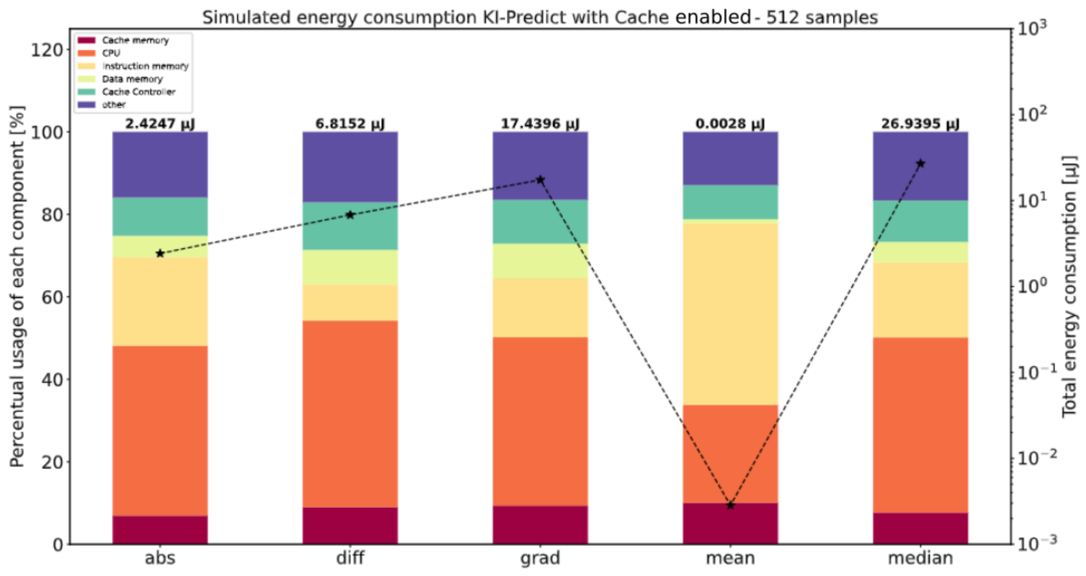


Figure 4.5: Energy consumption after disabling the digital filters - cache enabled

By enabling the cache some energy is saved. Although the module 'cache controller' consumes more, doing this saves a considerable amount of energy in the other modules, reducing the total. However, there is a strange behaviour on mean feature, because it is the only one that does not reduce the energy, but increases it. In the next analyses the truly explanation of this behaviour will be reached.

4.4 Features max, range and std

Three new features are executed by keeping the cache disabled, but when getting the results a strange pattern is observed (check figure 4.6): Features max and range are performed in a really short time interval, similar to what happened with mean, but the most remarkable fact is that both get the exact same values of power consumption. In order to better understand what is happening the .elf file was read. By analyzing the file it is discovered that, as these features are not being used in any other part of the code the compiler assumes that it is worthless to execute them, so they are optimized away. That is why the energy for feature 'mean' is greater when enabling the cache, as no code is being executed, so cache is not saving energy but just consuming more for the fact of it just being working. In order to fix this problem the value of the feature after the computation is going to be obtained and printed. The use of the print statement avoids the code being optimized away.

After performing this change in the main files, the energy consumption increases for these features, as observed in 4.7 (from $0.0016\mu J$ to $3.1219\mu J$). Now, cache is enabled and it is checked that energy is reduced for all the features (figure 4.8 respect to figure 4.7).

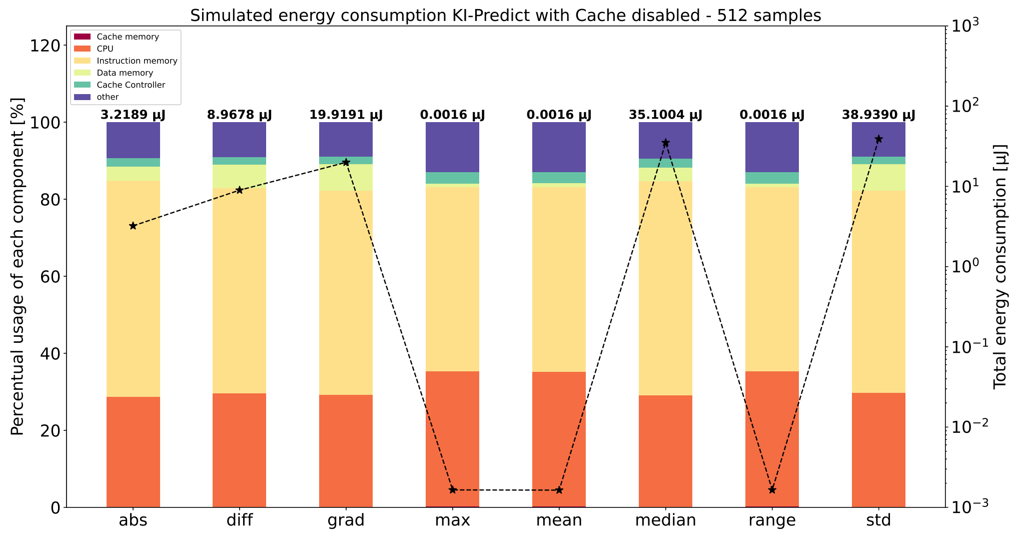


Figure 4.6: Energy consumption of all features with cache disabled. A strange pattern can be observed in features 'max', 'mean' and 'range': They consume the same amount of energy.

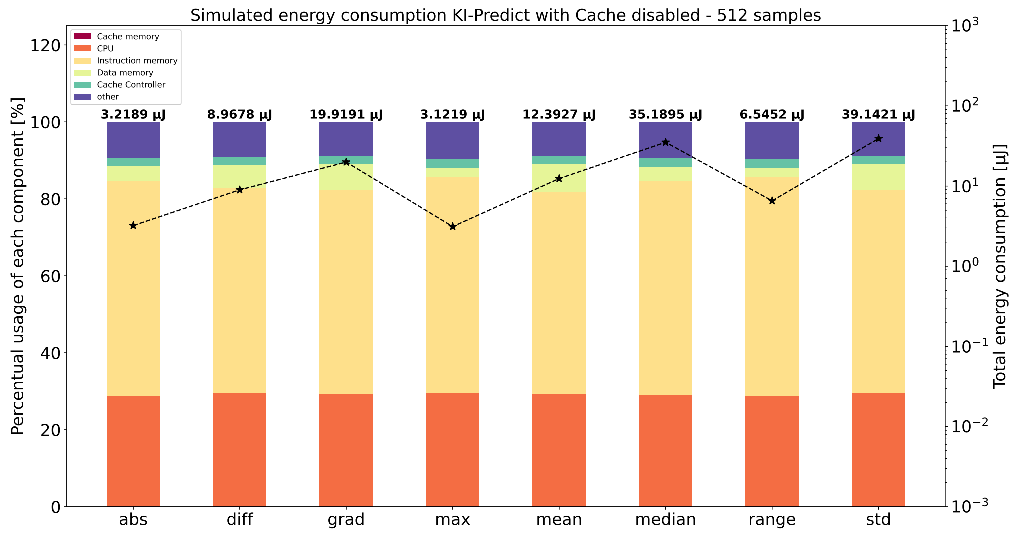


Figure 4.7: Energy consumption of all features with cache disabled. After avoiding the compiler optimizing away 'max', 'mean' and 'range' features, the energy consumption behaviour follows a similar pattern to the other features.

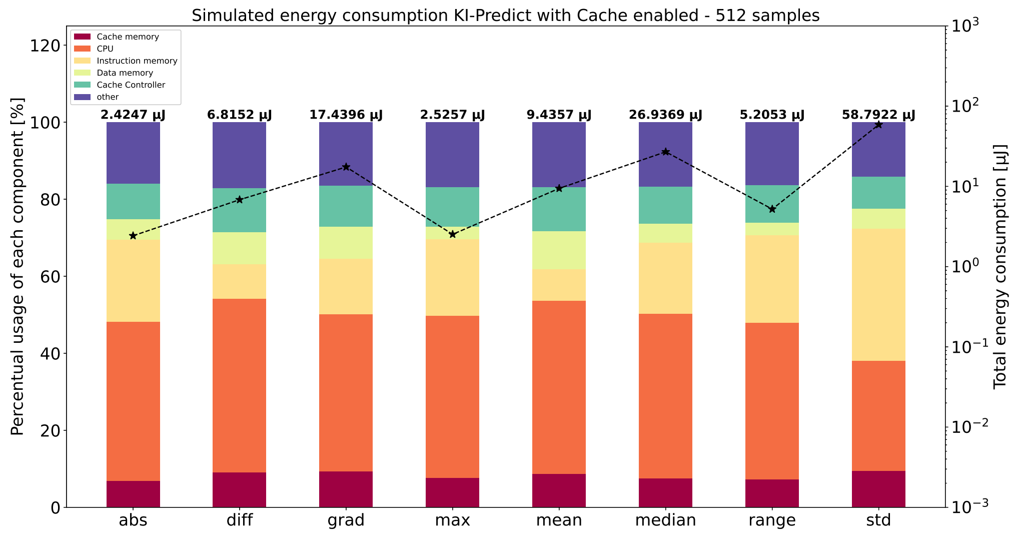


Figure 4.8: Energy consumption of all features with cache enabled

4.5 Activate FPU unit

With these results it is time to make a comparison between simulation and real measures. Measures are obtained in the lab for cache disabled by other workmates of the AutoML project. In the simulation, the execution of a feature in the microcontroller takes a time interval in the order of nanoseconds. When taking real measurements (they are taken in the commercial MCU used in AutoML) this is just not possible to check, so the feature is executed in a loop and the power computation obtained is proportionally divided in the amount of time and number of feature executions. In order to get the real power values the current is measured and, by using the formula

$$P(mW) = I(mA) * V(V)$$

the power is obtained. These data are compared with the simulation one and it is observed that measurement results for features diff, mean, grad and std do not fit the simulation results (compare figure 4.9 with figure 4.7). Features 'diff', 'grad', 'max' and 'std' present bigger values of consumption in the simulation. For example, 'std' consumes 39.1421 μJ in the simulation, but 4.3447 μJ is the value of the laboratory measurement, so it is more or less 10 times bigger.

These features are the ones that are working on computations with float numbers. So, it is decided to activate the FPU unit to check the differences between using it or not. The results after the activation of this unit can be seen in figure 4.10. Comparing them with results on figure 4.7, it is visible that the energy values are smaller with the FPU use. However, the desire behaviour appears, as the proportion of energy consumption between features follows the same pattern as the measurements in the real model (figure 4.9): Median is the one consuming the highest amount of energy and grad and std are consuming considerably way less and the proportion between mean and median in both graphs is approximately 13 times bigger median respect to mean.

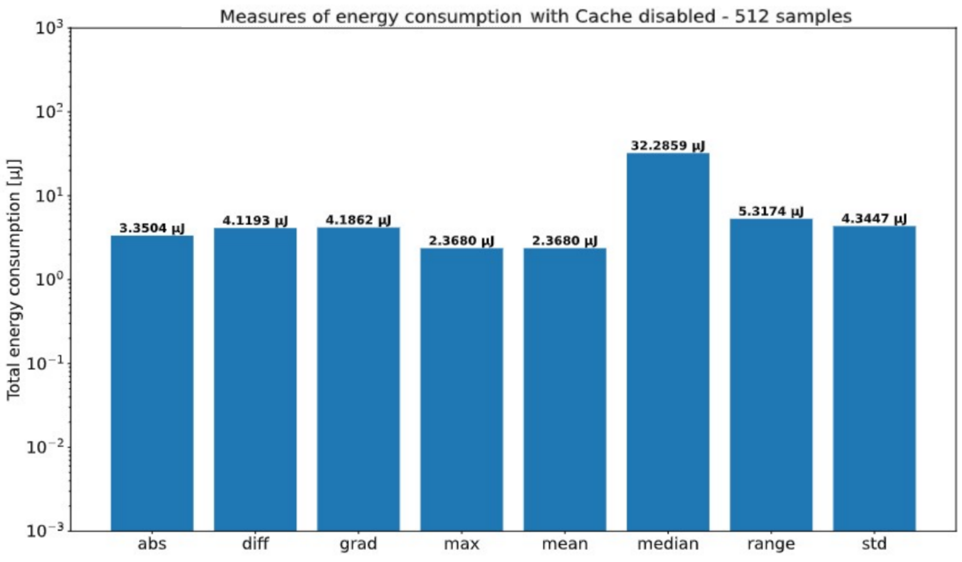


Figure 4.9: Measurement results. If this graph is compared with figure 4.7 it can be concluded that the simulation results don't follow the same behaviour pattern as the measures.

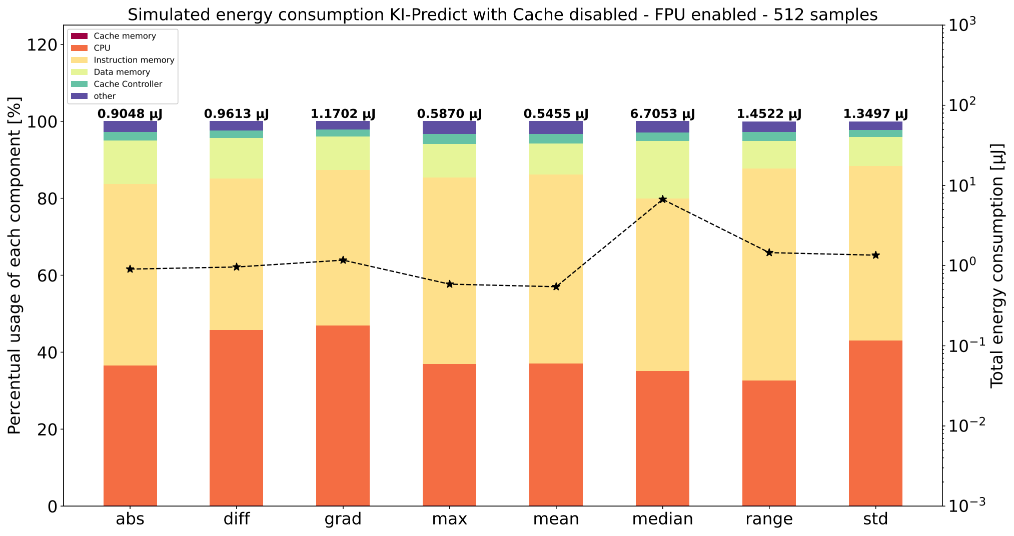


Figure 4.10: Results for FPU working, cache disabled

Chapter 5

Software optimization

After the study of the most optimal hardware configurations, the machine learning pipelines software plays also an important role. An study is taken to find the most accurate pipeline model. By using python, a machine learning pipeline is created. The structure of the file is seen in figure 5.1.

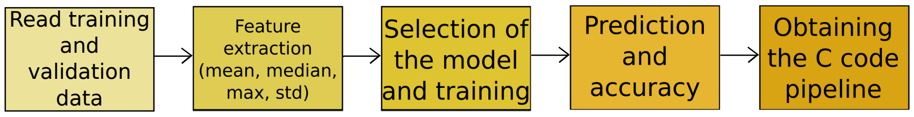

Read training and validation data
Feature extraction (mean, median, max, std)
Selection of the model and training
Prediction and accuracy
Obtaining the C code pipeline

Figure 5.1: Pipeline structure in python file

The last step of the python code, as seen on the figure, is the creation of the pipeline in C format, which is used to run it in the microcontroller and do the power simulations. To begin the analyses, just one simple feature is selected and all the different models are executed, obtaining the accuracy of the prediction. These models are: decision tree, random forest, support vector machines and oneclass svm. To begin, the default hyperparamenters configuration is used for decision tree and random forest. For svc, gamma = 0.001, and for OneclassSVM, gamma = 0.1, nu = 0.9 and kernel = 'rbf'.

5.1 Pipelines with one feature

For every single model a pipeline is created that works with just one feature. The analyses of the accuracy is performed by checking the results for features 'mean', 'median', 'max' and 'std'. A set of 1000 data samples are used to train the pipeline and 1000 to test it. Normally three different sets of data would be needed. The first one to train the pipeline, the second one to adjust the hyperparameters to get the best possible prediction accuracy and the third one to test it. For the purpose of this thesis it is possible to use just one set for training and adjusting hyperparameters and another for the test. These are the results of the accuracy for each possible pipeline (table 5.1).

By analyzing these results, several conclusions are extracted. First, it is important to say that OneClass can not work with multiple classification labels. The same will be applicable to SVC in case of working just with one SVM. In the research done SVC will work with just one and classify the data about the bearing mechanism behaviour in two: it is working properly or it presents a failure. Something similar occurs with OneClass: Everything that suits the conditions of the class

Accuracy of prediction for pipelines with one feature				
	SVC	OneClassSVM	Decision tree	Random Forest
Mean	0.769	0.783	0.291	0.346
Max	0.769	0.666	0.704	0.755
Median	0.769	0.768	0.294	0.342
Std	0.769	0.686	0.749	0.793

Table 5.1: Accuracy of the prediction for different pipelines

’works’ is predicted as ’1’, and everything that does not achieve the conditions to be classified as ’works’ is classified with a ’-1’ (fails). So the only two models that give detailed information about what are the characteristics of the motor when it fails are Decision Tree and Random Forest models. For these two models it is clear that the features ’max’ and ’std’ are the ones that give a more exact prediction. Model OneClassSVM presents the opposite behaviour: it is more accurate for features ’mean’ and ’median’. And when talking about SVC there’s the same prediction for every different feature pipeline. It is decided to study in detail the prediction of the SVC and it is discovered that for gamma=0.001 every single input is always classified as the same, as a failure. This behaviour can be checked visually on the graph in appendix B.

So for the current hyperparameters this model is not working properly. After the conclusions extracted from this first analyses, a second step is taken. For models Decision Tree and Random Forest, as both of them have a multi-label classification, the energy consumption of both is analysed to see which one is the most power/accuracy efficient. Moreover, as we have seen the accuracy for using just one feature, first an analyses is performed to check the accuracy after introducing a two feature pipeline with the two more accurate (max and std) and a pipeline that includes the four features.

5.2 Multi-features pipelines

Pipelines with two and four features are executed. Here it is possible to check the obtained accuracy results:

Accuracy of predictions for pipelines with multi-features				
	SVC	OneClassSVM*	Decision tree	Random Forest
Two features (max + std)	0.769	0.679	0.851	0.899
The four features	0.769	0.679	0.865	0.914

Table 5.2: Accuracy of the prediction for pipelines with multiple features

Several conclusions can be extracted from table 5.2. Firstly, there is a really small difference of accuracy for Decision Tree and Random Forest models between using the two features pipeline and the four features one, so the power consumption for both cases is studied. For the SVC case is the same problem that was faced before, the accuracy is the same but the hyperparameters are not well adjusted. Later on these will adjusted and the accuracy will be studied. For the OneClassSVM algorithm, in this case the kernel has been changed to a ’linear’ one because it gives a better accuracy when using multiple features than by using the ’rbf’. For the energy consumption, the following results are obtained.

Energy consumption (J)		
	Decision tree	Random Forest
Two features (max + std)	3.28E-08	1.37E-06
The four features	4.43E-08	1.84E-06

Table 5.3: Energy consumption of two and four feature pipelines

One important difference between the accuracy study and the energy consumption study: For the accuracy, 1000 data samples are used for training, while for the power consumption as the C++ code is run in the microcontroller, it has a memory limitation and it is not possible to work with so many data samples. So 100 samples are used for training except when running the four features pipeline for Random Forest, that 80 are used because this algorithm consumes more memory. Now, by analyzing the results a conclusion is reached: Decision Tree consumes considerably less power than Random Forest, and as the difference of prediction accuracy is not so big, Decision Tree model will be more optimal. Also, it is checked that using four features consumes more energy than two, and as the accuracy difference is not so big, the two features model are chosen.

5.3 Hyperparameters tuning

In order to improve the accuracy values, hyperparameters are tuned to get the best possible pipeline. As seen before, SVC is not working properly because of not having a good hyperparameters configuration, so the accuracy is obtained by changing the value of gamma, using different kernels and adjusting 'C' hyperparameter (check the results in table 5.4).

It is easy to see in table 5.4 that the best configuration is having a 'rbf' kernel, a gamma of value 0.7 and a C=1. This accuracy is better than the one for Decision Tree and Random Forest, but this model is just classifying in two categories, by the bearing mechanism working or non-working, not giving any more detail (more details are in appendix B). An energy analysis is performed by using 100 data samples for training, but the energy obtained (5.26E-05 J) confirms that this is the most energy consumption model, so not the ideal. For Decision Tree and Random Forest some hyperparameters are changed, looking for the most ideal.

For Decision Tree we reached the conclusion that by saying that the maximum tree depth must be 9, the accuracy is increased between 0.1-0.2. For Random Forest if it is specified that the number of decision trees used is 70 (by default is 100), the accuracy is just reduced by 0.1 but some memory consumption is saved.

So, as a conclusion of this study it could be said that the best power/accuracy efficient pipeline would have the following characteristics:

- Model: Decision Tree
- Hyperparameters: Everything default except maximum depth that is 9
- Two features: Maximum and Standard Deviation.

For more details, check the Decision Tree Model graph in appendix A.

Hyperparameters for SVC	Accuracy of the prediction
Default, gamma=0.001	0.769
Kernel 'rbf', gamma=0.7, C=1	0.97
Kernel 'rbf', gamma=0.5, C=1	0.9
Kernel 'rbf', gamma=0.9, C=1	0.96
Kernel 'linear', gamma=0.7, C=1	0.91
Kernel 'poly', gamma=0.7, degree=3, C=1	0.82
Hyperparameters for Decision Tree	**Accuracy of the prediction**
Default	0.85
Default, maximum depth of 9	0.86-0.87
Default, minimum sample split of 20	0.86
Default, minimum impurity decrease of 0.2	0.231
Hyperparameters for Random Forest	**Accuracy of the prediction**
Default	0.903
Default, number of estimators = 70	0.89

Table 5.4: Accuracy of Decision Tree and Random Forest models prediction when adjusting hyperparameters

To check the precision of the pipeline selected it is good to have a look at the confusion matrix 5.2. This matrix displays the predicted label ('x' axis) versus the real labels ('y' axis), so it is a way of visualizing the predictions of the pipeline. If the matrix is quite a diagonal matrix, the accuracy is good. The better the accuracy, the more diagonal the matrix is. Also, the colours, as well as the numbers, indicate the quantity of data from one category with a particular classification label.

5.4 Floating to fixed point

In order to continue improving this model new ideas for reducing the energy consumption are put into practise. When working on the hardware improvement it is thought adding the FPU unit to compute more efficiently the algorithms and save power. But transforming everything, or at least, everything possible to fixed point so that the FPU used is reduced in order to save energy could work.

To develop this new idea, in the python pipeline design the input data is transformed to fixed

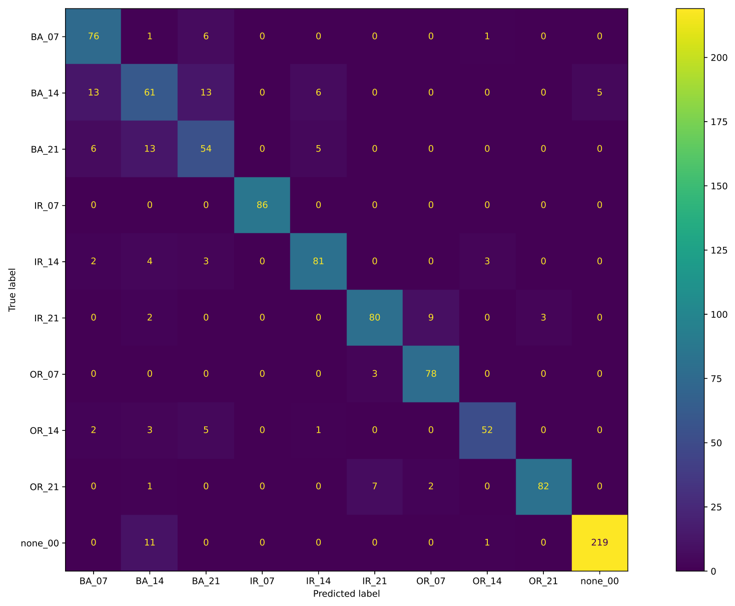


Figure 5.2: Confusion matrix for the selected pipeline

point, working with 7 binary digits to represent the fractional part. In order to understand how this transformation works a practical example will be explained by using a real number, for instance, 3.22.

The integer part, ’3’, would be ’11’ in binary, while the fractional part, ’0.22’, would be ’0011100001...’. So in binary the number would be

$$11.0011100001...$$

For the floating point representation, 32 bits are used. First, the binary number is adapted, leaving just one digit, which always will be ’1’, on the integer part. For our example it would be

$$1.10011100001... \cdot 2^1.$$

From the 32, the most significant bit will represent the sign of the number, being ’1’ negative sign and ’0’ positive. The following 8 bits represent the exponent number, and the rest, 23, the fractional part, as the integer part is always going to be a 1, so not necessary to save it.

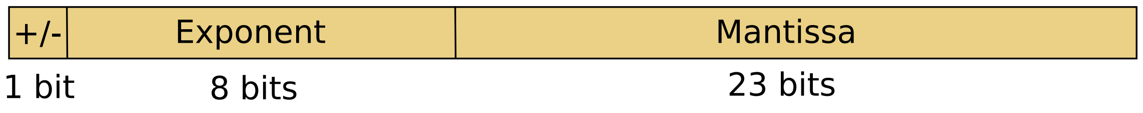


Figure 5.3: Floating point representation

To transform to fixed point the following algorithm is computed:

- First, 7 digits of representation for the fractional part want to be saved. In the case of the motor database, the input data goes from values between -1 and 1, so 8 bits in total are necessary to represent the total number. If this is not enough, 16 or 32 bits should be used. Keeping with the example of the 3.22 number, in order to save 7 digits the number is multiplied by 2^7, being the result

$$11.0011100001... \cdot 2^7 = 110011100.001... = 3.22 \cdot 2^7.$$

- Then, the number is rounded to keep just the integer part, which contains the 7 fractional digits to save. The result in the case of the example would be

$$110011100.$$

- Finally, the result is saved in an integer container. Working with it will reduce the use of the FPU.

Although this algorithm is made in the python pipeline, the 'transpile' function that obtains the .c file with the model works with floating arrays. So these arrays have to be changed by casting to integer. The accuracy after executing this change to the .c code is measured. This accuracy has to be compared with the accuracy of training with 1000 samples for floating point, so this value is also obtained (table 5.5).

Accuracy for floating point	Accuracy for fixed point
0.863	0.856

Table 5.5: Accuracy for floating and fixed point

The accuracy has been slightly reduced, but keeping almost the same value. When measuring the energy consumption values of the model the following results are obtained. As a reminder, for obtaining the energy the pipelines are always trained with 100 samples.

	Floating point energy consumed (J)	Fixed point energy consumed (J)
Total	3.08E-08	1.04E-08
FPU	3.24E-10	6.84E-11

Table 5.6: Energy consumption of the MCU for floating and fixed point

The total energy has been reduced to the half and FPU energy consumption has been reduced, from 3.24E-10J to 6.84E-11J. So, this improvement can be added to the search of the perfect pipeline. For details in the energy and power consumption check appendix C.

5.5 Features normalization

After the addition of the fixed point modification, the normalization of the features is performed. It is not always necessary in machine learning, it really depends on the training data and normally it is necessary when the different features have different ranges. It is the case for features 'max' and 'std'. So the idea is to convert these data in order for it to be in the range of [0-1]. In order to do this, for each feature, the minimum value is subtracted to every value of the set. Then everything is divided by the range. The importance of doing this is because, later on, when training the model, the feature with the bigger range will have a bigger influence in the prediction, and just because of having a bigger range interval would not mean that is more important. So, in case of having different ranges, it should be necessary to normalize.

However, after the normalization the accuracy (for fixed point from now on) was reduced to 0.75, the opposite behaviour to what was expected. After the analyses the conclusion was that, if the features are really dependent on the distribution and magnitude of the training data as could be in this case with 'maximum' and 'std', it could happen that the model can not set the boundaries correctly and the accuracy is reduced. Also Decision Tree is a model that relies on clear boundaries and data differences, so all these factors could justify the accuracy reduction. So, for this pipeline, normalization is ignored.

5.6 Common Statistical Features

After transforming the features to fixed point, the resulting confusion matrix for an accuracy of 0.856 is the following:

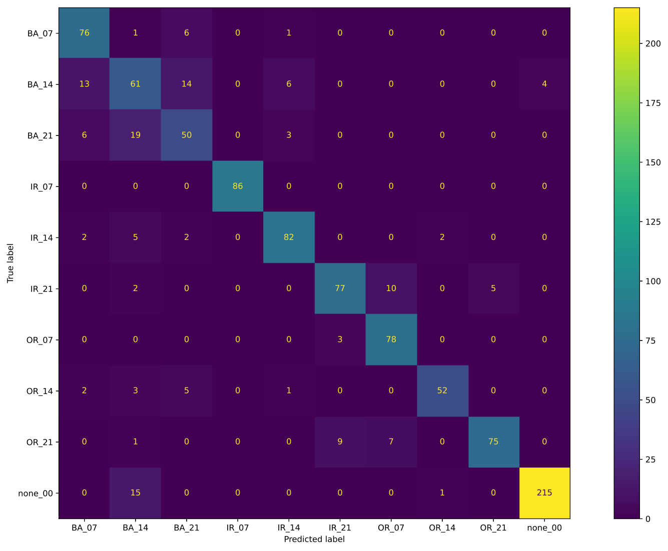


Figure 5.4: Fixed point confusion matrix

The objective now is to perfection it, trying to decrease the amount of prediction failures in those corners where the wrong classification is relatively big. For example, it is easy to see that 19 wrong classifications are produced when labelling as BA_14 data that should be BA_21. The acronyms BA, OR and IR indicate where the failure is produced (Ball, Outer Race or Inner Race, see figure 2.10) and the number (7, 14 or 21) specifies the fault diameter. Among the common statistical features seen in the Background, chapter 2.5, there are some that are more appropriate for detecting specific failures than others. For example, root amplitude, rms or peak will detect the changes in vibration and energy of the signal, as explained in chapter 2, section 2.3.2 of the book 'Intelligent fault diagnosis and remaining useful life prediction of rotating machinery' of Yaguo Lei [14]. To see the changes in time distribution, then, the appropriate features will be skewness, crest, clearance or kurtosis, for example[14]. To detect some defect in a bearing mating surface, features clearance and impulse will detect picks produced by this defect in the signal[14]. Also kurtosis and crest would be appropriate when having different operating conditions[14]. Kurtosis, due to the nature of its computing formula, will be good to detect early faults, but not severe ones, as it can take really small values when the mean is comparable bigger to the current data sample, which is translated in not good sensitivity when working with severe faults[14].

In order to improve the prediction, the values in the second column of the matrix 5.4 are studied. The biggest error is in the prediction of BA_21 data classified as BA_14. The error here is in the diameter where the failure is happening. On a rotating motor the pattern of the signal will be periodically repeated in different period values depending if the fault is in the Inner Race, Outer Race or in the Ball. Also, in the case of having a defect with a bearing mating surface, the spikes on the signal detected by the sensor will not be the same value for different failure diameters. Taking this into consideration it is decided to add features 'impulse' and 'clearance' to check if the failures related with the wrong classification due to the diameter are reduced (figure 5.5). The accuracy is improved, getting the value of 0.893, and in the confusion matrix the second column presents now less failures, as well as the seventh. However, the third one is worse. Now the objective is to improve that. It is decided to introduce other features which detect different failure properties to

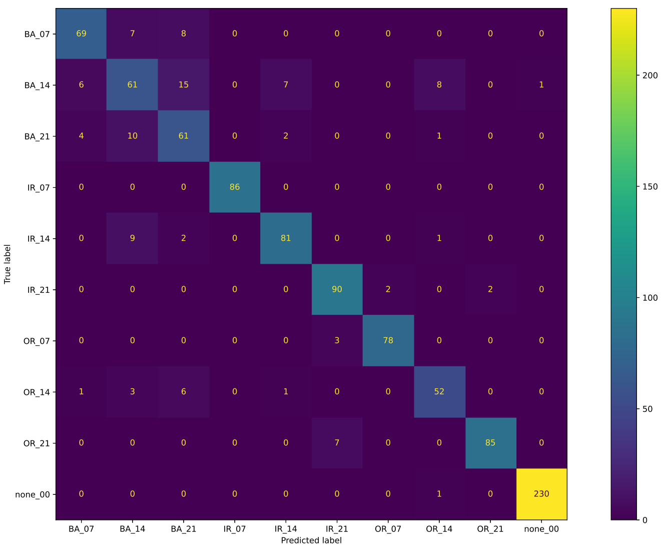


Figure 5.5: Confusion matrix for pipeline adding features impulse and clearance

try to improve more the accuracy. Kurtosis and crest are added to the pipeline. They improve the accuracy, obtaining 0.909 and the third column is improved, as it can be checked in the matrix in figure 5.6.

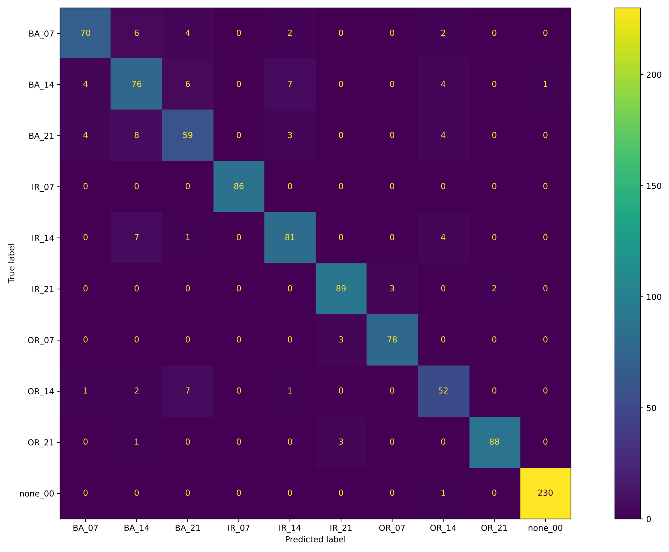


Figure 5.6: Confusion matrix for pipeline adding features kurtosis and crest features

Until now all the work has been done with time domain statistical features. These features are good detecting changes on the waveform of the signal but not in frequency. There are some changes in the frequency spectrum that can not be detected in time. So the first step to work with frequency domain features is to obtain the fast Fourier transform of the training and validation data. In order to train the model it is not possible to work with complex numbers, so instead of that all the work was done with the modulus. After several tests with different complex features it is checked that by adding the variance the accuracy is increased to 0.91, but by adding more extra features as X_{fc} or X_{rmsf} the accuracy is not increased, but a little bit reduced. So, just by adding the variance, it is increased to 0.91 obtaining the confusion matrix observed in figure 5.7.

5.7 Pipeline power consumption

After having found the most accurate pipeline, it is interesting to obtain the energy consumed. One important fact to remark is that, until now, as it can be seen in the structure in figure 5.1, all the features have been computed in python and what has been run in the MCU is just the trained model in C. So the power and energy consumption obtained was not of the whole pipeline but just of the model. So now the features are executed in the MCU obtaining the consumption of the whole pipeline. A problem appears and it is that, due to the lack of time, it is not possible to adapt the main file to compute the fft and obtain the frequency domain features in the MCU, so it is just possible to obtain the consumption for the temporal features pipeline. Nevertheless, as the accuracy difference between introducing the frequency variance or not is really small (from 0.909 to 0.91, almost the same), this analyses is performed with the features in the temporal domain. So, here the difference in energy consumption between executing the features in the MCU or not.

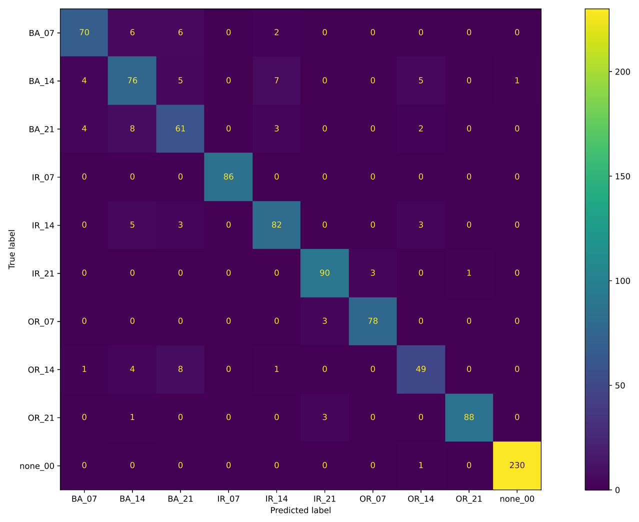


Figure 5.7: Confusion matrix for pipeline adding the frequency variance

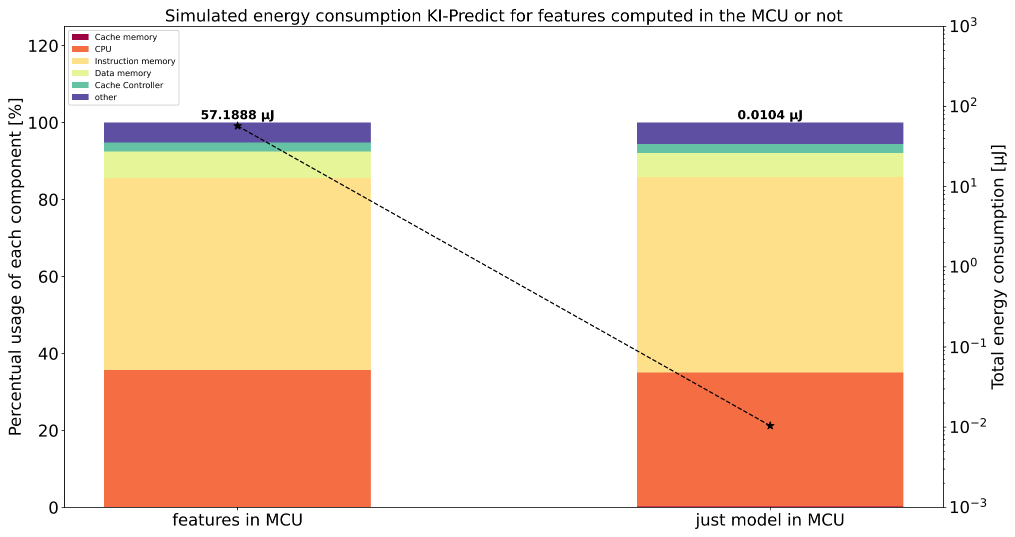


Figure 5.8: Energy consumption comparison for a pipeline with the features run in the MCU versus not

This big change of energy (an energy consumption 5499 times bigger, as seen in figure 5.8) it is expected, as running all the algorithms to get the features requires a lot of work for the CPU. But for the thesis the features should be run in the MCU, so these should be the values of energy consumption. A conclusion from this can be reached, and it is that depending on the quantity of features executed the energy consumption is greater or not. As a balance between accuracy and consumption has to be found it could be said that it will be not worthy to compute the fft due to the small accuracy difference, as it consumes more energy.

As it has been seen during the hardware power consumption research in chapter 4, enabling the cache memory reduces the power consumption. So it is going to be enabled and see the difference in the result (figure 5.9). The energy reduction is 89% compared to the previous value (from 57.1888uJ to 6.2503uJ). As a final step to conclude this BSc. thesis, a previous step is going to be undone: The FPU is going to be disconnected, as now it is working with fixed point. There is not a big noticeable difference in energy (from 6.2503E-6 J to 6.0759E-6 J), but at least some more energy is saved (check figure 5.10, 2.9% of energy saved).

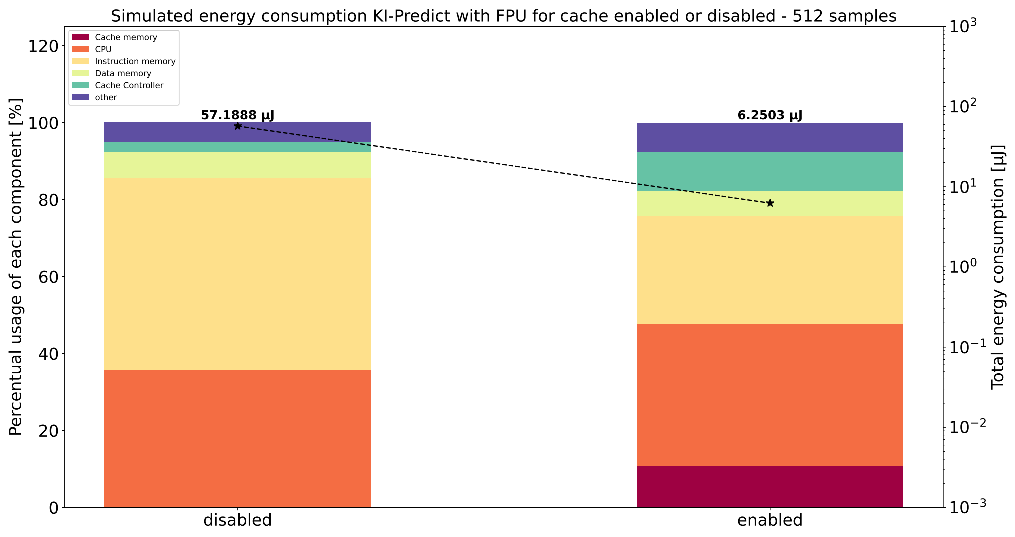


Figure 5.9: Energy consumption comparison between enabling the cache or not

For more details in the consumption values of the different modules check appendix C.

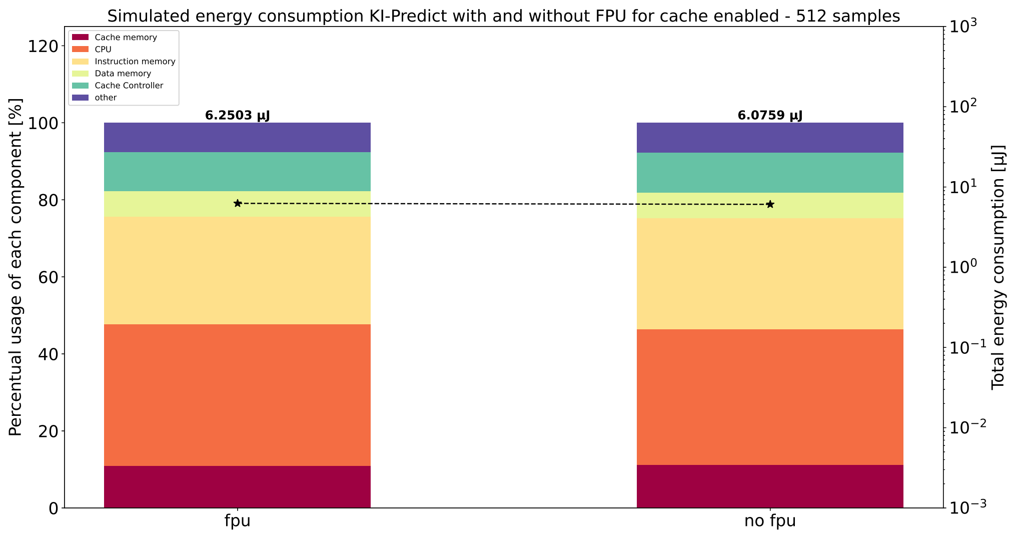


Figure 5.10: Energy consumption comparison between having the FPU connected or not

Chapter 6

Conclusions and future work

6.1 Conclusion

The objectives set at the beginning of the thesis have been fulfilled. It was necessary first to create a good theoretical background. The new concept of Edge Computing was comprehended in order to understand why it is so relevant to look for the minimum power consumption possible and the structure of the KI-Predict MCU has been studied to proper understand the results obtained later. Learning about machine learning, its different models and how pipelines work was an enriching task, as all these new concepts were understood by seeing them in practise in a condition monitoring application field. So, all the theory was acquired at the same time that new skills, such as working with Python, using the sklearn library, developing pipelines in C/C++ or learning how Cadence Joules RTL works, were achieved.

During the hardware research some important conclusions were achieved: Working with the cache enabled consumes less energy, and if the main program is working with floating point connecting the FPU will reduce the energy consumption. If it is working with fixed point it is better to disconnect it. For the peripherals, the clock will be disconnected while the main program is running in order not to consume unnecessary energy.

From the software research other conclusions were obtained: some models as OneClass can only classify data under two labels, or can do it also with more labels but consuming more energy, as SVC with more than one SVM. So the model would be chosen depending on objective. Random Forest model can be more accurate but requires more computations than Decision Tree. Running more computations is translated in consuming more energy, so as the accuracy difference is not so big and as the main purpose is reducing the energy consumed, Decision Tree will be the model used. After adjusting the hyperparameters and obtaining the accuracy for working with different features, this was the final hardware-software configuration:

- Model: Decision Tree.
- Hyperparameters: Default plus maximum depth of 9.
- Temporal features: maximum, std, kurtosis, clearance, crest and impulse.
- Cache enabled.

- FPU disconnected.
- Fixed point.
- Total energy consumption: 6.08E-06 J.
- Total power consumption: 5.74E-04 W.

Final pipeline: 6 temporal features computed in the MCU in fixed point		
Simulation time: 10584387 ns		
Module	**Energy (J)**	**Power (W)**
TOTAL	6.08E-06	5.74E-04
Cache (mem4)	6.81E-07	6.43E-05
CPU	2.14E-06	2.02E-04
OBIXBAR	1.11E-07	1.05E-05
ROM	3.54E-08	3.34E-06
Peripherals	3.76E-08	3.55E-06
Instr memory: mem0 + mem1	1.75E-06 = 1.69E-06 + 5.88E-08	1.65E-04 = 1.59E-04 + 5.55E-06
Data memory: mem2 + mem3	4.10E-07 = 4.94E-08 + 3.61E-07	3.87E-05 = 4.66E-06 + 3.41E-05
i_cache_system	6.29E-07	5.94E-05
i_cache_handler	5.62E-07	5.31E-05
i_tag_handler	2.13E-07	2.01E-05

Through the whole research it was possible to see that concepts as data normalization or FPU use can reduce or not the energy consumed depending on the pipeline and the database, so a universal statement can not be set to say that by using or not some of these a specific objective will be achieved. For the case of the database used, normalizing these data did not improve the accuracy, just the opposite behaviour.

Another of the objectives was the introduction of some frequency domain features. The variance could improve a bit the accuracy, but for our data set introducing more frequency domain features was not making any improvement. So, as computing the fft also requires some more energy it was decided that was better just to work with temporal features.

In the Fraunhofer IIS research institute I have learnt how a whole research process works and this will help me on the development of future projects. All these skills achieved could be used to improve this BSc. thesis. This possible future improvements are going to be explained in the next paragraphs. For now, as already said, all the proposed objectives have been completed and this thesis can be concluded.

6.2 Future work

Difficulties faced have led to new ideas to improve the work and results. Some of the possible future improvements are summarized in the next paragraphs.

6.2.1 MCU memory limitations

During the research big amounts of data were processed, for training the pipeline, for obtaining the features, for computing validation, classification... Several times there has come a problem when talking about the memory limitation of the microcontroller. At the beginning the memory had to be increased in order to be able to run the features simulation. Later on, when working with the pipelines, the whole database was a numpy array of 6556 data arrays of 512 values each. This was too much for the MCU when having to obtain the classification, so it had to be considerably reduced to 100 samples. Depending on the model this limit could be higher or lower, but it was considered to work using the same amount of input data for all the models in order to make proper comparisons.

Also, when talking about the classification labels, 40 initial labels was too much for the MCU, because a lot of computations had to be done. So, these labels were reduced to 10 in order to be able to consume less memory.

One possible future improvement [20] could be trying to find a method to select which are the, for example, 100 best data samples to use to train our model. When having to select them, we have always selected the first 100 samples of the array, but probably the density distribution of the samples is not equitable and it would be a good idea to pick up less samples from the areas where there is more density of them, and more from other parts where there is not so much data.

6.2.2 Training data sign

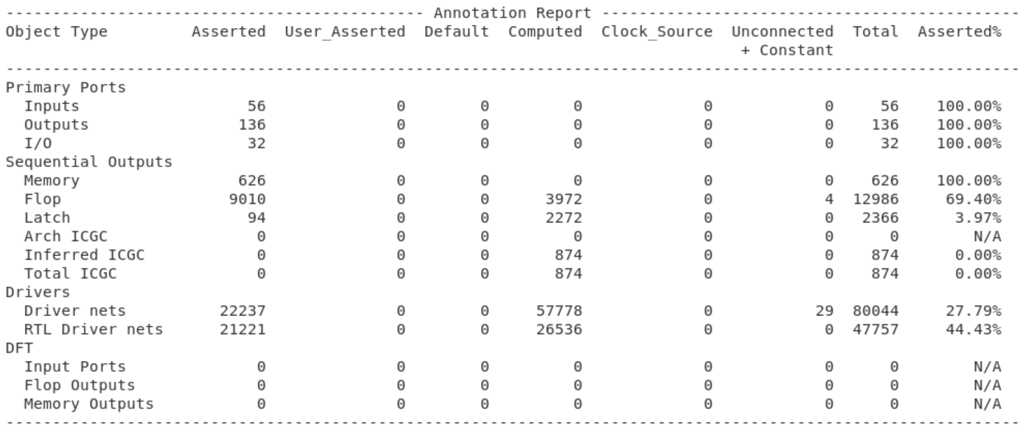
When working with the temporal features on the pipelines an unexpected conclusion due to an error was reached. As seen since the beginning of chapter 5.1, 'maximum' feature was being used. However, as defined in the background 2.5, 'peak' is not the same as 'maximum' because it is defined as the maximum of the absolute values of the input data. By mistake, 'crest', 'clearance' and 'impulse', that depend on peak value, were calculated by using the maximum value, not obtaining the absolute. When finding out the mistake and fixing it to follow the proper definition of the features the accuracy of the prediction was reduced (from 0.909 that was previously for the 6 temporal features used to 0.887). So, it could be concluded from this that the model does not depend just on the height of the mean, max, etc., but also on the sign. This could be studied in future. For the aim of this thesis, as the objective is to find the most optimal pipeline, the 'wrong' definition option will be kept.

6.2.3 Frequency domain features

As seen in section 5.6 the introduction of frequency domain features does not improve much the accuracy of the prediction. However, the expected behaviour is that for the introduction of these new features the accuracy could increase until 0.99. Why this is not happening could be a question to solve in future research.

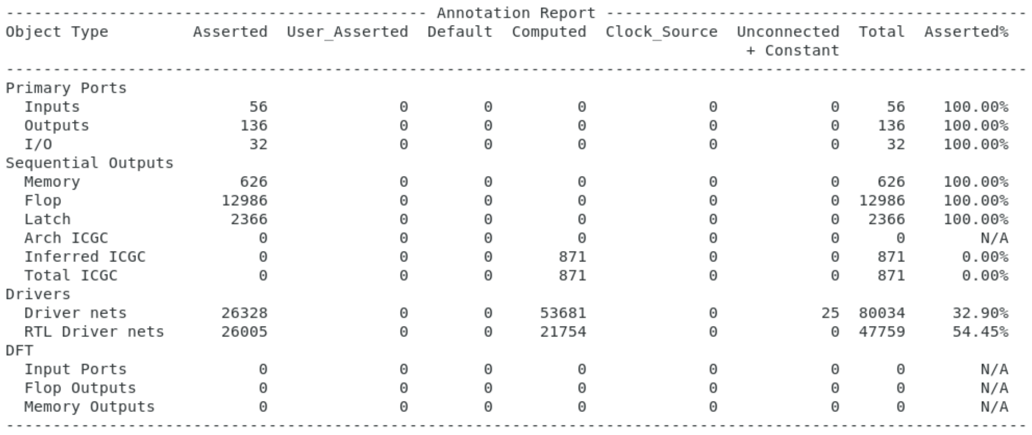
6.2.4 Questasim and Xcelium

Questasim has been used during the whole project. Using the designed testbench a verilog simulation was run and the corresponding file .vcd with the stimuli was created. This stimuli file was used when running a power simulation with Joules RTL. However, the percentage of asserted cells was quite small. An asserted object is the one where the tool is able to find a particular flip flop or net in the vcd file. When Joules is reading it not all the cells are identified, which can lead to some errors of accuracy in the results obtained from the power simulations if it is not properly finding all the nets. By changing the simulation tool to Xcelium this problem is solved. Xcelium, which is a simulation tool from Cadence, as well as Joules RTL, uses shm files, which are smaller and cleaner. Before, every activity was being stored on .vcd files, which is an old format that occupies a big memory space and has problems identifying nets/flip-flops which are defined as 2D arrays. It is not a big problem for the main purpose, but in order to obtain a bigger accuracy on the power consumption of the different parts of the MCU Xcelium would be the solution.



```
--------------------------------------------- Annotation Report ---------------------------------------------
Object Type          Asserted  User_Asserted  Default  Computed  Clock_Source  Unconnected  Total  Asserted%
                                                                               + Constant
-------------------------------------------------------------------------------------------------------------
Primary Ports
  Inputs                   56              0        0         0             0            0     56    100.00%
  Outputs                 136              0        0         0             0            0    136    100.00%
  I/O                      32              0        0         0             0            0     32    100.00%
Sequential Outputs
  Memory                  626              0        0         0             0            0    626    100.00%
  Flop                   9010              0        0      3972             0            4  12986     69.40%
  Latch                    94              0        0      2272             0            0   2366      3.97%
  Arch ICGC                 0              0        0         0             0            0      0        N/A
  Inferred ICGC             0              0        0       874             0            0    874      0.00%
  Total ICGC                0              0        0       874             0            0    874      0.00%
Drivers
  Driver nets           22237              0        0     57778             0           29  80044     27.79%
  RTL Driver nets       21221              0        0     26536             0            0  47757     44.43%
DFT
  Input Ports               0              0        0         0             0            0      0        N/A
  Flop Outputs              0              0        0         0             0            0      0        N/A
  Memory Outputs            0              0        0         0             0            0      0        N/A
-------------------------------------------------------------------------------------------------------------
```

Figure 6.1: Annotation report using .vcd file



```
--------------------------------------------- Annotation Report ---------------------------------------------
Object Type          Asserted  User_Asserted  Default  Computed  Clock_Source  Unconnected  Total  Asserted%
                                                                               + Constant
-------------------------------------------------------------------------------------------------------------
Primary Ports
  Inputs                   56              0        0         0             0            0     56    100.00%
  Outputs                 136              0        0         0             0            0    136    100.00%
  I/O                      32              0        0         0             0            0     32    100.00%
Sequential Outputs
  Memory                  626              0        0         0             0            0    626    100.00%
  Flop                  12986              0        0         0             0            0  12986    100.00%
  Latch                  2366              0        0         0             0            0   2366    100.00%
  Arch ICGC                 0              0        0         0             0            0      0        N/A
  Inferred ICGC             0              0        0       871             0            0    871      0.00%
  Total ICGC                0              0        0       871             0            0    871      0.00%
Drivers
  Driver nets           26328              0        0     53681             0           25  80034     32.90%
  RTL Driver nets       26005              0        0     21754             0            0  47759     54.45%
DFT
  Input Ports               0              0        0         0             0            0      0        N/A
  Flop Outputs              0              0        0         0             0            0      0        N/A
  Memory Outputs            0              0        0         0             0            0      0        N/A
-------------------------------------------------------------------------------------------------------------
```

Figure 6.2: Annotation report using .shm file

As seen, not all the flip-flops and latches are asserted while working with .vcd files (figure 6.1).

Literature

[1] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. "Edge machine learning for ai-enabled iot devices: A review". In: *Sensors* 20.9 (2020), p. 2533.

[2] S.L. Andresen. "John McCarthy: father of AI". In: *IEEE Intelligent Systems* 17.5 (2002), pp. 84–85. DOI: `10.1109/MIS.2002.1039837`.

[3] S. Nandi, H.A. Toliyat, and X. Li. "Condition Monitoring and Fault Diagnosis of Electrical Motors—A Review". In: *IEEE Transactions on Energy Conversion* 20.4 (2005), pp. 719–729. DOI: `10.1109/TEC.2005.847955`.

[4] "KI-PREDICT - Elektronik für verteilte Künstliche Intelligenz zur sensorba-sierten Prozess- und Zustandskontrolle". In: *Bundesministerium für Bildung und Forschung* (). URL: `https://www.elektronikforschung.de/projekte/ki-predict`.

[5] ""KI-PREDICT" – Intelligent process monitoring with on-sensor signal preprocessing". In: *Online magazine Fraunhofer Institute for Integrated Circuits IIS* (Apr. 2021). URL: `https://www.iis.fraunhofer.de/en/magazin/artificial-intelligence-ai-series/ki_predict.html`.

[6] Fraunhofer IIS. "Internal Technical Specification Manual of KI-Predict ASIC". In: (2021).

[7] "Joules User Guide 21.1". In: *support.cadence.com* (Jan. 2022). URL: `https://support.cadence.com/apex/techpubDocViewerPage?xmlName=joules_ug_cui.xml&title=Joules%20User%20Guide%20--%20Introduction%20-%20RTL%20Power%20Flow%20Using%20Joules&hash=pgfId-1054707&c_version=21.1&path=joules_ug_cui/joules_ug_cui21.1/chap1.html#pgfId-1054707`.

[8] "An introduction to machine learning with scikit-learn". In: *Scikit* (2017). URL: `https://scikit-learn.org/stable/tutorial/basic/tutorial.html`.

[9] Julianna Delua. "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications". In: *IBM* (Mar. 2021). URL: `https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning#:~:text=The%20main%20difference%20between%20supervised,unsupervised%20learning%20algorithm%20does%20not.`.

[10] "ageron/handson-ml: A series of Jupyter notebooks that walk you through the fundamentals of Machine Learning and Deep Learning in python using Scikit-Learn and TensorFlow." In: *GitHub* (). URL: `https://github.com/ageron/handson-ml6`.

[11] *Scikit-learn*. URL: `https://scikit-learn.org/stable/`.

[12] Lotfi Saidi, Jaouher Ben Ali, and Farhat Fnaiech. "Application of higher order spectral features and support vector machines for bearing faults classification". In: *ISA Transactions* 54 (Oct. 2014). DOI: 10.1016/j.isatra.2014.08.007.

[13] Case Western Reserve University - Case School of Engineering. *12k Drive End Bearing Fault Data: Case School of Engineering*. 2021. URL: https://engineering.case.edu/bearingdatacenter/12k-drive-end-bearing-fault-data.

[14] Yaguo Lei. *Intelligent fault diagnosis and remaining useful life prediction of rotating machinery*. Elsevier Butterworth-Heinemann, 2017.

[15] Phillipp Woller. "Fraunhofer IIS: AutoML ASIC – Energy-Aware AutoML for Embedded Systems ". In: *Eureka cluster AI - Project pitch* (Apr. 2021). URL: https://eureka-clusters-ai.eu/wp-content/uploads/2021/04/philipp-woller-g2_automl_asic_farunhofer_iis_pitch.pdf?x34473.

[16] PULP platform. "PULP FAQs". In: *Pulp-platform.org* (). URL: https://pulp-platform.org/faq.html.

[17] PULP platform. In: *Pulp-platform.org* (). URL: https://pulp-platform.org//.

[18] P. D. Schiavone et al. "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications". In: *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)* (2017). DOI: 10.1109/PATMOS.2017.8106976. URL: https://ieeexplore.ieee.org/document/8106976.

[19] A. Pullini et al. "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing". In: *IEEE Journal of Solid-State Circuits* (May 2019), pp. 1970–1981. ISSN: 0018-9200. DOI: 10.1109/JSSC.2019.2912307. URL: https://ieeexplore.ieee.org/document/8715500.

[20] Shenglong Zhou. "Sparse SVM for Sufficient Data Reduction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3075339.

Part I

Appendix

Appendix A

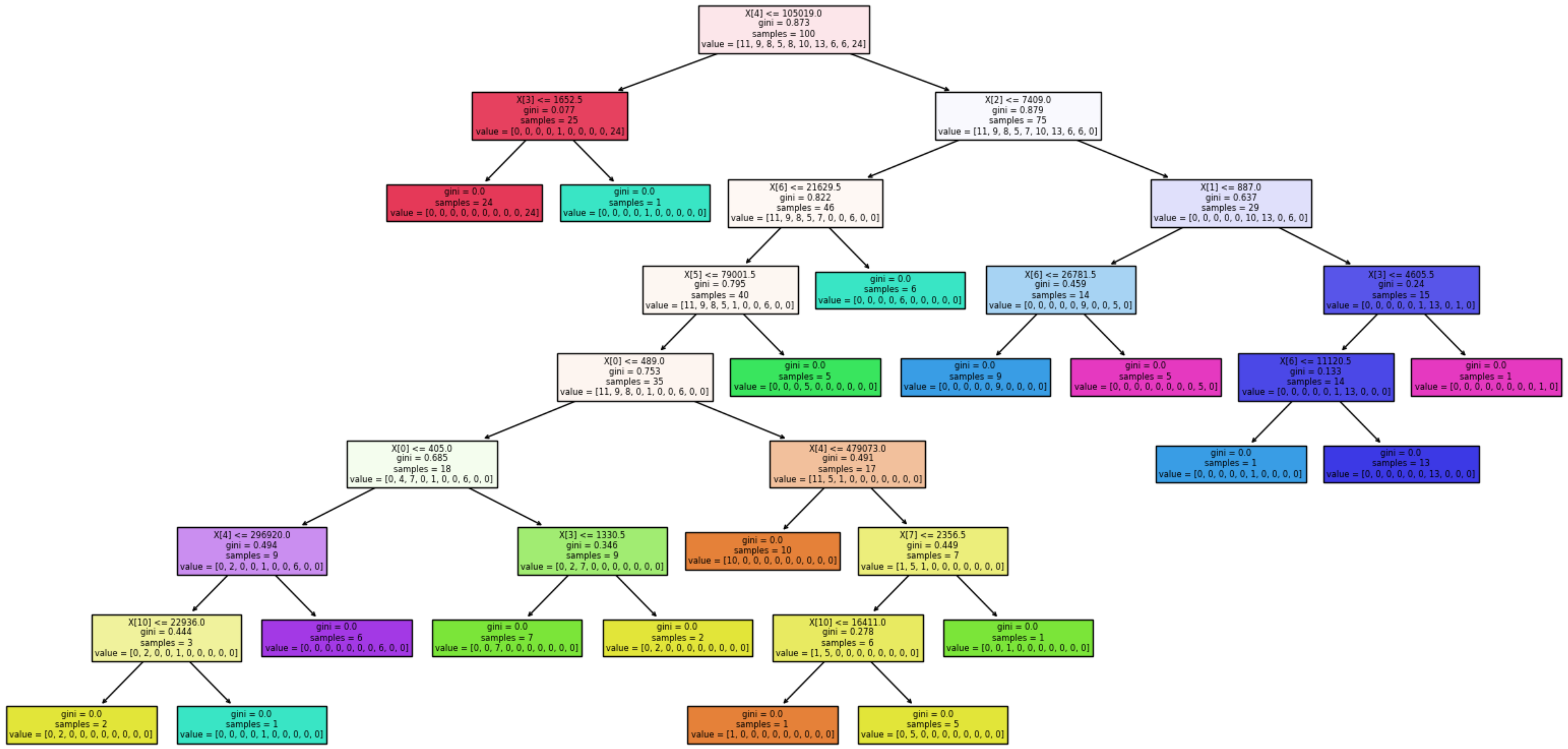
Decision Tree Model

When doing the first Machine Learning Pipelines analyses (see section 5.1) it was concluded that the best model to use in the pipeline was the Decision Tree one, when talking about power/accuracy efficiency, improving it slightly when selecting a maximum tree depth of 9 splits. Also, the best features configuration was using just the 2 most accurate ones, maximum and standard deviation. In this section of the appendix the tree is illustrated. Here, a brief description to understand it.

In each node there are four different labels. The first one indicates the condition that is going to be evaluated for splitting the training data samples in two groups. The data samples fulfilling the condition will go on one side, on the left. If some of these data don't achieve the specified condition, they will go to the right branch. In order to measure the probability of data being classified incorrectly there's the 'gini' factor.

The third label in any node indicates the total number of samples in that point that are going to be splitted in two. That's why, as we are working with 100 training samples, at the beginning, in the first node, the value of samples is 100. Finally, the last label is 'value'. There are 10 different labels into which the samples can be classified. In an array, every position is a different label and the number in every position is the number of samples under that label. It is possible to check that, at the end of the tree, the final nodes where gini has value 0 and all the samples are under one label, the nodes classifying data under the same label have the same colour to visually be able to identify them quicker: purple, yellow, light green, dark green, light blue, normal blue, dark blue, orange, pink and red.

Decision tree with maximum depth '9', 100 training data samples and 6 features


X[4] <= 105019.0 gini = 0.873 samples = 100 value = [11, 9, 8, 5, 8, 10, 13, 6, 6, 24]
X[3] <= 1652.5 gini = 0.077 samples = 25 value = [0, 0, 0, 0, 1, 0, 0, 0, 0, 24]
X[2] <= 7409.0 gini = 0.879 samples = 75 value = [11, 9, 8, 5, 7, 10, 13, 6, 6, 0]
gini = 0.0 samples = 24 value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 24]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
X[6] <= 21629.5 gini = 0.822 samples = 46 value = [11, 9, 8, 5, 7, 0, 0, 6, 0, 0]
X[1] <= 887.0 gini = 0.637 samples = 29 value = [0, 0, 0, 0, 0, 10, 13, 0, 6, 0]
X[5] <= 79001.5 gini = 0.795 samples = 40 value = [11, 9, 8, 5, 1, 0, 0, 6, 0, 0]
gini = 0.0 samples = 6 value = [0, 0, 0, 0, 6, 0, 0, 0, 0, 0]
X[6] <= 26781.5 gini = 0.459 samples = 14 value = [0, 0, 0, 0, 0, 9, 0, 0, 5, 0]
X[3] <= 4605.5 gini = 0.24 samples = 15 value = [0, 0, 0, 0, 0, 1, 13, 0, 1, 0]
X[0] <= 489.0 gini = 0.753 samples = 35 value = [11, 9, 8, 0, 1, 0, 0, 6, 0, 0]
gini = 0.0 samples = 5 value = [0, 0, 0, 5, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 9 value = [0, 0, 0, 0, 0, 9, 0, 0, 0, 0]
gini = 0.0 samples = 5 value = [0, 0, 0, 0, 0, 0, 0, 0, 5, 0]
X[6] <= 11120.5 gini = 0.133 samples = 14 value = [0, 0, 0, 0, 0, 1, 13, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
X[0] <= 405.0 gini = 0.685 samples = 18 value = [0, 4, 7, 0, 1, 0, 0, 6, 0, 0]
X[4] <= 479073.0 gini = 0.491 samples = 17 value = [11, 5, 1, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
gini = 0.0 samples = 13 value = [0, 0, 0, 0, 0, 0, 13, 0, 0, 0]
X[4] <= 296920.0 gini = 0.494 samples = 9 value = [0, 2, 0, 0, 1, 0, 0, 6, 0, 0]
X[3] <= 1330.5 gini = 0.346 samples = 9 value = [0, 2, 7, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 10 value = [10, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[7] <= 2356.5 gini = 0.449 samples = 7 value = [1, 5, 1, 0, 0, 0, 0, 0, 0, 0]
X[10] <= 22936.0 gini = 0.444 samples = 3 value = [0, 2, 0, 0, 1, 0, 0, 0, 0, 0]
gini = 0.0 samples = 6 value = [0, 0, 0, 0, 0, 0, 0, 6, 0, 0]
gini = 0.0 samples = 7 value = [0, 0, 7, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 2 value = [0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
X[10] <= 16411.0 gini = 0.278 samples = 6 value = [1, 5, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 2 value = [0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
gini = 0.0 samples = 1 value = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 5 value = [0, 5, 0, 0, 0, 0, 0, 0, 0, 0]

Appendix B

SVC graphical representation

When comparing different pipeline models in Section 5.3 one of the models with high accuracy was the SVC one. However, it was decided not to use it as it was only classifying data in two groups, 'fails' or 'works', instead of giving more details of what was failing.

Nevertheless, for a better understanding of how the models do the predictions we will see graphically how SVC works, as it is the simplest model to represent.

In figure B.1 it is seen the prediction for a pipeline with two features, standard deviation and maximum. All the dots are the validation data, so black would be the data that is a case of failure in the bearing motor and white is the data that represents a case of correct behaviour, just taking into account one of the sensors (let's remember that there are 2 sensors located on the bearing motor). These data are represented on the graph depending on its value of standard deviation and maximum. Each of the four different graphs represents a different configuration of the hyperparameters. Depending on its value, the model fits the shape of two areas, the considered 'failure' area in blue colour and the 'working' area in red colour. All the validation data that is enclosed by the red area is classified as working correctly, independently if this labeling is correct or not. For example, it is easy to identify in the gamma = 0.001 graph that there are some black dots in the red area. These would be wrongly classified.

So, by observing the graph, it could be concluded that the prediction that fits the best to the validation data is the one for RBF kernel and gamma=0.7. But it is also important to be sure that we don't get into overfitting, in other words, if the prediction area is too tight to the training data it can be that some validation data easily fall outside of this area and are classified wrongly.

Other point to consider is that, in this case there are only 2 features, so the prediction area is easily represented in a graph in 2D. In case of having 3, 4 or more features this should be represented in 3, 4 or more dimensions, which is really difficult to visualise. Also, in case of having more than 2 labels, as it is the case of Decision Tree model, more than 2 classification areas would be represented in the graph.

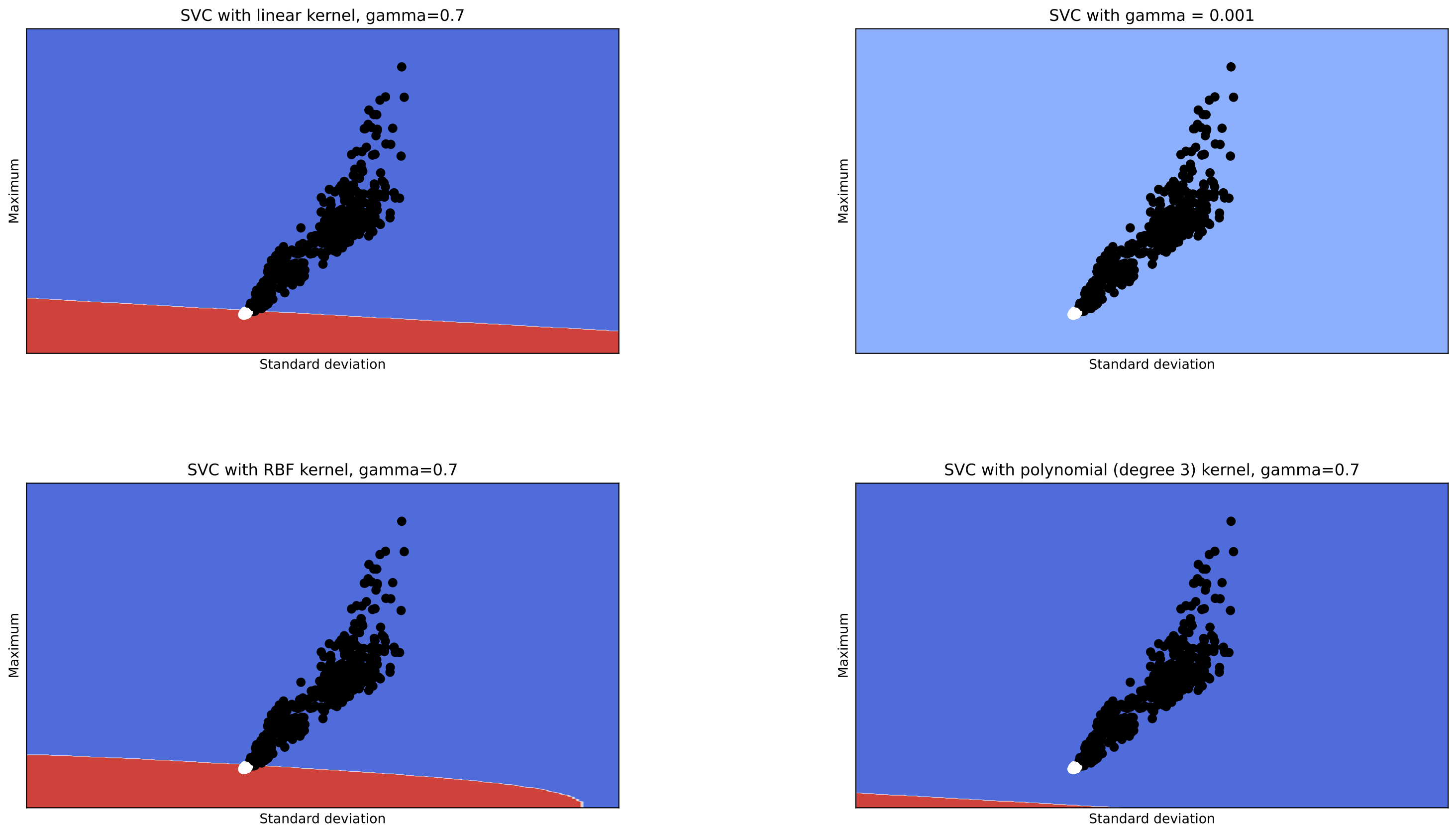


Figure B.1: SVC hyperparameters tuning for features maximum and standard deviation

Appendix C

Energy and power main results

In this appendix the detail energy and power consumption data of the most important results are collected. To begin, the first table shows the energy consumption of the final ideal pipeline model. After, more important data of previous results are exposed.

Final pipeline plus hardware configuration:

- Decision Tree model
- Hyperparamenters: everything default except maximum depth = 9
- 6 features: standard deviation, maximum, impulse, kurtosis, crest and clearance
- Fixed point
- FPU disconnected
- Cache enabled

HIERARCHY DETAILS (POWER IN W)

```
------------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------------
59168   100.00% 7.64187e-06 4.60781e-04 1.05623e-04 5.74047e-04   1 */i_MCU_subsystem_top
58800    99.38% 7.63540e-06 4.51616e-04 9.61101e-05 5.55362e-04   2 */i_MCU_subsystem_top/digital_core_instance
16337    27.61% 1.95692e-08 1.40545e-04 6.17749e-05 2.02340e-04   3 */i_MCU_subsystem_top/digital_core_instance/cpu_region_i
  149     0.25% 1.87206e-06 1.55089e-04 2.35258e-06 1.59314e-04   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem0
  178     0.30% 5.77270e-08 6.13254e-05 2.95193e-06 6.43351e-05   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem4
 2165     3.66% 3.45101e-09 4.39124e-05 1.55128e-05 5.94286e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_cache_system
  153     0.26% 1.87206e-06 3.15763e-05 6.34568e-07 3.40829e-05   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem3
 3963     6.70% 3.03836e-09 5.66713e-06 4.85135e-06 1.05215e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_obixbar
    2     0.00% 3.33400e-12 2.41705e-07 7.48042e-06 7.72212e-06   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_cpu_support
  149     0.25% 1.87206e-06 3.50270e-06 1.77860e-07 5.55262e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem1
  149     0.25% 1.87206e-06 2.62552e-06 1.66596e-07 4.66418e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem2
20562    34.75% 3.52263e-08 3.33200e-06 1.84299e-07 3.55152e-06   3 */i_MCU_subsystem_top/digital_core_instance/peripherals_i
    7     0.01% 8.07298e-09 3.33339e-06 0.00000e+00 3.34146e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_instr_rom
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_debug
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_hwac
14975    25.31% 2.00450e-08 0.00000e+00 0.00000e+00 2.00450e-08   3 */i_MCU_subsystem_top/digital_core_instance/hw_accel_i
   24     0.04% 3.90270e-11 8.50240e-06 7.66781e-06 1.61702e-05   2 */i_MCU_subsystem_top/i_clk_res_gen_bur
   75     0.13% 6.18827e-09 0.00000e+00 0.00000e+00 6.18827e-09   2 */i_MCU_subsystem_top/digital_pads_instance
  190     0.32% 1.52509e-10 0.00000e+00 0.00000e+00 1.52509e-10   2 */i_MCU_subsystem_top/i_kipred01_pad_mux
------------------------------------------------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------------------
16337    27.61% 1.95692e-08 1.40545e-04 6.17749e-05 2.02340e-04   3 */digital_core_instance/cpu_region_i
12003    20.29% 1.44135e-08 1.40121e-04 6.17749e-05 2.01910e-04   4 */digital_core_instance/cpu_region_i/cv32e40p_core_instance
12003    20.29% 1.44135e-08 1.40121e-04 6.17749e-05 2.01910e-04   5 */digital_core_instance/cpu_region_i/cv32e40p_core_instance/core_i
  560     0.95% 6.27955e-10 4.24570e-07 0.00000e+00 4.25198e-07   4 */digital_core_instance/cpu_region_i/apb_int_cntrl_i
 3734     6.31% 4.51179e-09 0.00000e+00 0.00000e+00 4.51179e-09   4 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance
 2215     3.74% 2.66633e-09 0.00000e+00 0.00000e+00 2.66633e-09   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_csrs
  601     1.02% 9.78321e-10 0.00000e+00 0.00000e+00 9.78321e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/dap
  809     1.37% 7.53183e-10 0.00000e+00 0.00000e+00 7.53183e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_mem
  103     0.17% 9.59750e-11 0.00000e+00 0.00000e+00 9.59750e-11   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_sba
------------------------------------------------------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------------------
 1621     2.74% 2.86225e-09 4.07372e-05 1.23584e-05 5.30984e-05   4 */digital_core_instance/i_cache_system/i_cache_handler
 1011     1.71% 1.92168e-09 2.35986e-05 7.10745e-06 3.07080e-05   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm
  878     1.48% 1.74600e-09 1.54795e-05 4.64591e-06 2.01271e-05   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_tag_handler
  114     0.19% 1.60085e-10 7.84956e-06 2.12583e-06 9.97555e-06   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_miss_way_selector
  147     0.25% 2.22203e-10 7.29386e-06 2.29024e-06 9.58432e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_pipeline_1to2_fifo
  291     0.49% 5.18767e-10 4.59929e-06 1.27079e-06 5.87060e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_data_buffer_fifo
   84     0.14% 9.53970e-11 2.91832e-06 1.09631e-06 4.01473e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_cache_if_fsm
   54     0.09% 7.02960e-11 1.70395e-06 3.82586e-07 2.08661e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_load_fsm
------------------------------------------------------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------------------
20562    34.75% 3.52263e-08 3.33200e-06 1.84299e-07 3.55152e-06   3 */digital_core_instance/peripherals_i
   74     0.13% 1.80830e-10 1.23763e-06 9.63438e-08 1.33415e-06   4 */digital_core_instance/peripherals_i/u_obi2apb
    2     0.00% 2.90800e-12 4.25781e-07 0.00000e+00 4.25784e-07   4 */digital_core_instance/peripherals_i/core_clock_gate_30M
  792     1.34% 6.91735e-10 2.49719e-07 8.79552e-08 3.38366e-07   4 */digital_core_instance/peripherals_i/periph_bus_i
  792     1.34% 6.91735e-10 2.49719e-07 8.79552e-08 3.38366e-07   5 */digital_core_instance/peripherals_i/periph_bus_i/apb_node_wrap_i
  564     0.95% 1.09566e-09 1.41523e-07 0.00000e+00 1.42619e-07   4 */digital_core_instance/peripherals_i/apb_soc_ctrl_i
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[0].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[1].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[2].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[3].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[5].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[6].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[7].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[8].core_clock_gate
    2     0.00% 3.06400e-12 1.41927e-07 0.00000e+00 1.41930e-07   4 */digital_core_instance/peripherals_i/genblk1[9].core_clock_gate
 9968    16.85% 1.85872e-08 0.00000e+00 0.00000e+00 1.85872e-08   4 */digital_core_instance/peripherals_i/apb_adc_int_i
 8542    14.44% 1.60736e-08 0.00000e+00 0.00000e+00 1.60736e-08   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_dig_filt_wrap
   66     0.11% 8.92940e-11 0.00000e+00 0.00000e+00 8.92940e-11   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_tempadc_ctrl
 2360     3.99% 3.85117e-09 0.00000e+00 0.00000e+00 3.85117e-09   4 */digital_core_instance/peripherals_i/apb_spi_master0_i
  845     1.43% 1.17166e-09 0.00000e+00 0.00000e+00 1.17166e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_spictrl
  510     0.86% 9.58632e-10 0.00000e+00 0.00000e+00 9.58632e-10   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_txfifo
  509     0.86% 9.37833e-10 0.00000e+00 0.00000e+00 9.37833e-10   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_rxfifo
  399     0.67% 6.46261e-10 0.00000e+00 0.00000e+00 6.46261e-10   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_axiregs
 2355     3.98% 3.84180e-09 0.00000e+00 0.00000e+00 3.84180e-09   4 */digital_core_instance/peripherals_i/apb_spi_master1_i
  857     1.45% 1.17981e-09 0.00000e+00 0.00000e+00 1.17981e-09   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_spictrl
  503     0.85% 9.36033e-10 0.00000e+00 0.00000e+00 9.36033e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_txfifo
  502     0.85% 9.35495e-10 0.00000e+00 0.00000e+00 9.35495e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_rxfifo
  396     0.67% 6.53682e-10 0.00000e+00 0.00000e+00 6.53682e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_axiregs
 2005     3.39% 3.02508e-09 0.00000e+00 0.00000e+00 3.02508e-09   4 */digital_core_instance/peripherals_i/apb_timer_i
  476     0.80% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[0].timer_i
  476     0.80% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[1].timer_i
  476     0.80% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[2].timer_i
  476     0.80% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[3].timer_i
 1169     1.98% 1.90789e-09 0.00000e+00 0.00000e+00 1.90789e-09   4 */digital_core_instance/peripherals_i/apb_uart_i
  340     0.57% 5.99802e-10 0.00000e+00 0.00000e+00 5.99802e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_fifo_i
  345     0.58% 5.96753e-10 0.00000e+00 0.00000e+00 5.96753e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_fifo_i
  185     0.31% 2.69233e-10 0.00000e+00 0.00000e+00 2.69233e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_i
  157     0.27% 2.17725e-10 0.00000e+00 0.00000e+00 2.17725e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_i
   22     0.04% 3.33940e-11 0.00000e+00 0.00000e+00 3.33940e-11   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_interrupt_i
  426     0.72% 7.56852e-10 0.00000e+00 0.00000e+00 7.56852e-10   4 */digital_core_instance/peripherals_i/apb_i2c_i
  310     0.52% 5.56458e-10 0.00000e+00 0.00000e+00 5.56458e-10   5 */digital_core_instance/peripherals_i/apb_i2c_i/byte_controller
  444     0.75% 6.62886e-10 1.45600e-12 0.00000e+00 6.64342e-10   4 */digital_core_instance/peripherals_i/apb_gpio_i
  237     0.40% 3.88994e-10 0.00000e+00 0.00000e+00 3.88994e-10   4 */digital_core_instance/peripherals_i/apb_analog_ctrl_i
  148     0.25% 2.05732e-10 0.00000e+00 0.00000e+00 2.05732e-10   4 */digital_core_instance/peripherals_i/apb_pmu_ctrl_i
------------------------------------------------------------------------------------------------------------------------------------------------
```

MODEL DECISION TREE CONSUMPTION - FIXED POINT*

*these data are the ones commented in section 5.4, so in this case FPU is conected and cache memory disabled.

Model consumption with temporal domain features - fixed point		
Simulation time: 10550 ns		
Module	**Energy (J)**	**Power (W)**
TOTAL	1.04E-08	9.88E-04
Cache (mem4)	3.08E-11	2.92E-06
CPU	3.62E-09	3.43E-04
OBIXBAR	1.96E-10	1.86E-05
ROM	3.53E-11	3.35E-06
Peripherals	4.56E-11	4.32E-06
Instr memory: mem0 + mem1	5.29E-09 = 5.25E-09 + 4.29E-11	5.02E-04 = 4.97E-04 + 4.07E-06
Data memory: mem2 + mem3	6.48E-10 = 4.29E-11 + 6.05E-10	6.15E-05 = 4.07E-06 + 5.74E-05
i_cache_system	2.55E-10	2.42E-05
i_cache_handler	1.64E-10	1.55E-05
i_tag_handler	9.58E-11	9.08E-06
FPU	6.84E-11	6.48E-06

HIERARCHY DETAILS (POWER IN W)

```
------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------
74294   100.00% 7.65734e-06 8.18241e-04 1.61656e-04 9.87554e-04   1 */i_MCU_subsystem_top
73920    99.50% 7.65087e-06 8.09072e-04 1.51062e-04 9.67785e-04   2 */i_MCU_subsystem_top/digital_core_instance
  153     0.21% 1.87206e-06 4.87835e-04 7.78960e-06 4.97497e-04   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem0
31464    42.35% 3.47324e-08 2.23653e-04 1.19562e-04 3.43251e-04   3 */i_MCU_subsystem_top/digital_core_instance/cpu_region_i
  153     0.21% 1.87207e-06 5.42221e-05 1.29517e-06 5.73893e-05   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem3
 2170     2.92% 3.53745e-09 1.98713e-05 4.33444e-06 2.42093e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_cache_system           |
 3953     5.32% 3.08336e-09 8.73389e-06 9.82653e-06 1.85635e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_obixbar
    2     0.00% 3.33500e-12 2.42278e-07 7.49814e-06 7.74042e-06   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_cpu_support
20548    27.66% 3.52026e-08 3.92676e-06 3.56565e-07 4.31852e-06   3 */i_MCU_subsystem_top/digital_core_instance/peripherals_i
  153     0.21% 1.87206e-06 2.03635e-06 1.58872e-07 4.06729e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem1
  153     0.21% 1.87206e-06 2.03635e-06 1.58872e-07 4.06729e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem2
    7     0.01% 8.07298e-09 3.34129e-06 0.00000e+00 3.34936e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_instr_rom
  178     0.24% 5.77387e-08 2.77899e-06 8.12589e-08 2.91799e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem4
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_debug
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_hwac
14975    20.16% 2.02171e-08 0.00000e+00 0.00000e+00 2.02171e-08   3 */i_MCU_subsystem_top/digital_core_instance/hw_accel_i
   24     0.03% 3.90300e-11 8.50564e-06 8.74843e-06 1.72541e-05   2 */i_MCU_subsystem_top/i_clk_res_gen_bur
   75     0.10% 6.18827e-09 0.00000e+00 0.00000e+00 6.18827e-09   2 */i_MCU_subsystem_top/digital_pads_instance
  196     0.26% 1.56422e-10 0.00000e+00 0.00000e+00 1.56422e-10   2 */i_MCU_subsystem_top/i_kipred01_pad_mux
------------------------------------------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------
31464    42.35% 3.47324e-08 2.23653e-04 1.19562e-04 3.43251e-04   3 */digital_core_instance/cpu_region_i
17926    24.13% 1.99977e-08 2.17279e-04 1.19041e-04 3.36340e-04   4 */digital_core_instance/cpu_region_i/cv32e40p_core_instance
17926    24.13% 1.99977e-08 2.17279e-04 1.19041e-04 3.36340e-04   5 */digital_core_instance/cpu_region_i/cv32e40p_core_instance/core_i
 9172    12.35% 9.58141e-09 5.94889e-06 5.21293e-07 6.47977e-06   4 */digital_core_instance/cpu_region_i/genblk1.fp_wrapper_i
 9172    12.35% 9.58141e-09 5.94889e-06 5.21293e-07 6.47977e-06   5 */digital_core_instance/cpu_region_i/genblk1.fp_wrapper_i/i_fpnew_bulk
  560     0.75% 6.27955e-10 4.25577e-07 0.00000e+00 4.26205e-07   4 */digital_core_instance/cpu_region_i/apb_int_cntrl_i
 3766     5.07% 4.50928e-09 0.00000e+00 0.00000e+00 4.50928e-09   4 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance
 2215     2.98% 2.66633e-09 0.00000e+00 0.00000e+00 2.66633e-09   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_csrs
  602     0.81% 9.78522e-10 0.00000e+00 0.00000e+00 9.78522e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/dap
  841     1.13% 7.54112e-10 0.00000e+00 0.00000e+00 7.54112e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_mem
  103     0.14% 9.59750e-11 0.00000e+00 0.00000e+00 9.59750e-11   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_sba
------------------------------------------------------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
------------------------------------------------------------------------------------------------------------------------------------
 1623     2.18% 2.96207e-09 1.49751e-05 5.65567e-07 1.55437e-05   4 */digital_core_instance/i_cache_system/i_cache_handler
 1011     1.36% 2.05423e-09 9.22083e-06 0.00000e+00 9.22288e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm
  878     1.18% 1.87410e-09 9.07897e-06 0.00000e+00 9.08084e-06   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_tag_handler
  114     0.15% 1.63736e-10 1.41859e-07 0.00000e+00 1.42023e-07   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_miss_way_selector
  291     0.39% 4.73007e-10 1.55805e-06 2.43744e-07 1.80227e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_data_buffer_fifo
   54     0.07% 7.28370e-11 1.55955e-06 1.99470e-07 1.75910e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_load_fsm
  149     0.20% 2.27548e-10 1.40569e-06 1.22353e-07 1.52827e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_pipeline_1to2_fifo
   84     0.11% 9.87430e-11 1.23102e-06 0.00000e+00 1.23112e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_cache_if_fsm
------------------------------------------------------------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells     Leakage    Internal   Switching       Total Lvl Instance
--------------------------------------------------------------------------------------------------------------------------------------------
20548    27.66% 3.52026e-08 3.92676e-06 3.56565e-07 4.31852e-06   3 */digital_core_instance/peripherals_i
   74     0.10% 1.80841e-10 1.56069e-06 2.25077e-07 1.78595e-06   4 */digital_core_instance/peripherals_i/u_obi2apb
  791     1.06% 6.90370e-10 4.37346e-07 1.02265e-07 5.40302e-07   4 */digital_core_instance/peripherals_i/periph_bus_i
  791     1.06% 6.90370e-10 4.37346e-07 1.02265e-07 5.40302e-07   5 */digital_core_instance/peripherals_i/periph_bus_i/apb_node_wrap_i
    2     0.00% 2.90800e-12 4.25781e-07 0.00000e+00 4.25784e-07   4 */digital_core_instance/peripherals_i/core_clock_gate_30M
  564     0.76% 1.09580e-09 1.77506e-07 1.21064e-08 1.90708e-07   4 */digital_core_instance/peripherals_i/apb_soc_ctrl_i
    2     0.00% 3.06400e-12 1.44207e-07 1.71157e-08 1.61326e-07   4 */digital_core_instance/peripherals_i/genblk1[2].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[0].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[1].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[3].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[5].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[6].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[7].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[8].core_clock_gate
    2     0.00% 3.06200e-12 1.42263e-07 0.00000e+00 1.42266e-07   4 */digital_core_instance/peripherals_i/genblk1[9].core_clock_gate
 2351     3.16% 3.83420e-09 4.16789e-08 0.00000e+00 4.55131e-08   4 */digital_core_instance/peripherals_i/apb_spi_master0_i
  844     1.14% 1.17613e-09 1.93304e-08 0.00000e+00 2.05065e-08   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_spictrl
  398     0.54% 6.48008e-10 9.26942e-09 0.00000e+00 9.91743e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_axiregs
  508     0.68% 9.37516e-10 5.03042e-09 0.00000e+00 5.96793e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_rxfifo
  504     0.68% 9.35765e-10 5.03042e-09 0.00000e+00 5.96618e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_txfifo
 9963    13.41% 1.85831e-08 0.00000e+00 0.00000e+00 1.85831e-08   4 */digital_core_instance/peripherals_i/apb_adc_int_i
 8536    11.49% 1.60698e-08 0.00000e+00 0.00000e+00 1.60698e-08   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_dig_filt_wrap
   66     0.09% 8.92940e-11 0.00000e+00 0.00000e+00 8.92940e-11   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_tempadc_ctrl
 2356     3.17% 3.84150e-09 0.00000e+00 0.00000e+00 3.84150e-09   4 */digital_core_instance/peripherals_i/apb_spi_master1_i
  857     1.15% 1.17981e-09 0.00000e+00 0.00000e+00 1.17981e-09   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_spictrl
  504     0.68% 9.38833e-10 0.00000e+00 0.00000e+00 9.38833e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_rxfifo
  502     0.68% 9.32389e-10 0.00000e+00 0.00000e+00 9.32389e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_txfifo
  396     0.53% 6.53682e-10 0.00000e+00 0.00000e+00 6.53682e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_axiregs
 2005     2.70% 3.02508e-09 0.00000e+00 0.00000e+00 3.02508e-09   4 */digital_core_instance/peripherals_i/apb_timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[0].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[1].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[2].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[3].timer_i
  444     0.60% 6.59978e-10 1.44100e-09 0.00000e+00 2.10098e-09   4 */digital_core_instance/peripherals_i/apb_gpio_i
 1168     1.57% 1.90822e-09 0.00000e+00 0.00000e+00 1.90822e-09   4 */digital_core_instance/peripherals_i/apb_uart_i
  338     0.45% 6.00414e-10 0.00000e+00 0.00000e+00 6.00414e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_fifo_i
  346     0.47% 5.96447e-10 0.00000e+00 0.00000e+00 5.96447e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_fifo_i
  185     0.25% 2.69233e-10 0.00000e+00 0.00000e+00 2.69233e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_i
  157     0.21% 2.17725e-10 0.00000e+00 0.00000e+00 2.17725e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_i
   22     0.03% 3.33940e-11 0.00000e+00 0.00000e+00 3.33940e-11   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_interrupt_i
  426     0.57% 7.56761e-10 0.00000e+00 0.00000e+00 7.56761e-10   4 */digital_core_instance/peripherals_i/apb_i2c_i
  310     0.42% 5.56458e-10 0.00000e+00 0.00000e+00 5.56458e-10   5 */digital_core_instance/peripherals_i/apb_i2c_i/byte_controller
  237     0.32% 3.88994e-10 0.00000e+00 0.00000e+00 3.88994e-10   4 */digital_core_instance/peripherals_i/apb_analog_ctrl_i
  149     0.20% 2.07311e-10 0.00000e+00 0.00000e+00 2.07311e-10   4 */digital_core_instance/peripherals_i/apb_pmu_ctrl_i
--------------------------------------------------------------------------------------------------------------------------------------------
```

MODEL DECISION TREE CONSUMPTION - FLOATING POINT*

Model consumption with temporal domain features - floating point		
Simulation time: 30200 ns		
Module	**Energy (J)**	**Power (W)**
TOTAL	3.08E-08	1.02E-03
Cache (mem4)	5.68E-11	1.88E-06
CPU	1.10E-08	3.64E-04
OBIXBAR	5.25E-10	1.74E-05
ROM	1.01E-10	3.34E-06
Peripherals	1.32E-10	4.36E-06
Instr memory: mem0 + mem1	1.52E-08 = 1.51E-08 + 1.23E-10	5.05E-04 = 5.01E-04 + 4.06E-06
Data memory: mem2 + mem3	2.20E-09 = 1.23E-10 + 2.08E-09	7.29E-05 = 4.06E-06 + 6.88E-05
i_cache_system	7.42E-10	2.46E-05
i_cache_handler	4.70E-10	1.56E-05
i_tag_handler	2.74E-10	9.07E-06
FPU	3.24E-10	1.07E-05

HIERARCHY DETAILS (POWER IN W)

```
-------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells      Leakage     Internal    Switching        Total Lvl Instance
-------------------------------------------------------------------------------------------------------------------------------
74294   100.00% 7.65777e-06 8.40165e-04 1.73204e-04 1.02103e-03   1 */i_MCU_subsystem_top
73920    99.50% 7.65129e-06 8.30999e-04 1.62622e-04 1.00127e-03   2 */i_MCU_subsystem_top/digital_core_instance
  153     0.21% 1.87206e-06 4.90555e-04 8.16430e-06 5.00591e-04   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem0
31464    42.35% 3.51735e-08 2.32222e-04 1.31781e-04 3.64038e-04   3 */i_MCU_subsystem_top/digital_core_instance/cpu_region_i
  153     0.21% 1.87206e-06 6.56745e-05 1.29122e-06 6.88378e-05   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem3
 2170     2.92% 3.53702e-09 2.00217e-05 4.54336e-06 2.45686e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_cache_system
 3953     5.32% 3.07510e-09 8.71353e-06 8.68151e-06 1.73981e-05   3 */i_MCU_subsystem_top/digital_core_instance/i_obixbar
    2     0.00% 3.33400e-12 2.41905e-07 7.48661e-06 7.72852e-06   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_cpu_support
20548    27.66% 3.52031e-08 3.99206e-06 3.27750e-07 4.35502e-06   3 */i_MCU_subsystem_top/digital_core_instance/peripherals_i
  153     0.21% 1.87206e-06 2.03322e-06 1.58627e-07 4.06391e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem1
  153     0.21% 1.87206e-06 2.03322e-06 1.58627e-07 4.06391e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem2
    7     0.01% 8.07298e-09 3.33615e-06 0.00000e+00 3.34422e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_instr_rom
  178     0.24% 5.77395e-08 1.79429e-06 2.97846e-08 1.88181e-06   3 */i_MCU_subsystem_top/digital_core_instance/u_data_mem4
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_debug
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   3 */i_MCU_subsystem_top/digital_core_instance/core_clock_gate_hwac
14975    20.16% 2.02171e-08 0.00000e+00 0.00000e+00 2.02171e-08   3 */i_MCU_subsystem_top/digital_core_instance/hw_accel_i
   24     0.03% 3.90320e-11 8.50312e-06 8.73607e-06 1.72392e-05   2 */i_MCU_subsystem_top/i_clk_res_gen_bur
   75     0.10% 6.18827e-09 0.00000e+00 0.00000e+00 6.18827e-09   2 */i_MCU_subsystem_top/digital_pads_instance
  196     0.26% 1.56423e-10 0.00000e+00 0.00000e+00 1.56423e-10   2 */i_MCU_subsystem_top/i_kipred01_pad_mux
-------------------------------------------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells      Leakage     Internal    Switching        Total Lvl Instance
-------------------------------------------------------------------------------------------------------------------------------
31464    42.35% 3.51735e-08 2.32222e-04 1.31781e-04 3.64038e-04   3 */digital_core_instance/cpu_region_i
17926    24.13% 2.08359e-08 2.23877e-04 1.28980e-04 3.52878e-04   4 */digital_core_instance/cpu_region_i/cv32e40p_core_instance
17926    24.13% 2.08359e-08 2.23877e-04 1.28980e-04 3.52878e-04   5 */digital_core_instance/cpu_region_i/cv32e40p_core_instance/core_i
 9172    12.35% 9.18442e-09 7.92031e-06 2.80027e-06 1.07298e-05   4 */digital_core_instance/cpu_region_i/genblk1.fp_wrapper_i
 9172    12.35% 9.18442e-09 7.92031e-06 2.80027e-06 1.07298e-05   5 */digital_core_instance/cpu_region_i/genblk1.fp_wrapper_i/i_fpnew_bulk
  560     0.75% 6.27955e-10 4.24922e-07 0.00000e+00 4.25550e-07   4 */digital_core_instance/cpu_region_i/apb_int_cntrl_i
 3766     5.07% 4.50928e-09 0.00000e+00 0.00000e+00 4.50928e-09   4 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance
 2215     2.98% 2.66633e-09 0.00000e+00 0.00000e+00 2.66633e-09   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_csrs
  602     0.81% 9.78522e-10 0.00000e+00 0.00000e+00 9.78522e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/dap
  841     1.13% 7.54112e-10 0.00000e+00 0.00000e+00 7.54112e-10   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_mem
  103     0.14% 9.59750e-11 0.00000e+00 0.00000e+00 9.59750e-11   5 */digital_core_instance/cpu_region_i/jtag_debug_interface_instance/i_dm_sba
-------------------------------------------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells      Leakage     Internal    Switching        Total Lvl Instance
-------------------------------------------------------------------------------------------------------------------------------
 1623     2.18% 2.96207e-09 1.49521e-05 5.95378e-07 1.55504e-05   4 */digital_core_instance/i_cache_system/i_cache_handler
 1011     1.36% 2.05423e-09 9.20664e-06 0.00000e+00 9.20870e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm
  878     1.18% 1.87410e-09 9.06500e-06 0.00000e+00 9.06688e-06   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_tag_handler
  114     0.15% 1.63736e-10 1.41641e-07 0.00000e+00 1.41804e-07   6 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage1_fsm/i_miss_way_selector
  291     0.39% 4.73007e-10 1.55565e-06 2.43369e-07 1.79950e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_data_buffer_fifo
   54     0.07% 7.28360e-11 1.55715e-06 2.29844e-07 1.78707e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_load_fsm
  149     0.20% 2.27548e-10 1.40353e-06 1.22165e-07 1.52592e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_pipeline_1to2_fifo
   84     0.11% 9.87430e-11 1.22913e-06 0.00000e+00 1.22923e-06   5 */digital_core_instance/i_cache_system/i_cache_handler/i_cache_stage2_cache_if_fsm
-------------------------------------------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------------------
Cells Pct_cells      Leakage     Internal    Switching        Total Lvl Instance
-------------------------------------------------------------------------------------------------------------------------------
20548    27.66% 3.52031e-08 3.99206e-06 3.27750e-07 4.35502e-06   3 */digital_core_instance/peripherals_i
   74     0.10% 1.80845e-10 1.71013e-06 2.25265e-07 1.93557e-06   4 */digital_core_instance/peripherals_i/u_obi2apb
  791     1.06% 6.90333e-10 4.07387e-07 9.29803e-08 5.01058e-07   4 */digital_core_instance/peripherals_i/periph_bus_i
  791     1.06% 6.90333e-10 4.07387e-07 9.29803e-08 5.01058e-07   5 */digital_core_instance/peripherals_i/periph_bus_i/apb_node_wrap_i
    2     0.00% 2.90800e-12 4.25781e-07 0.00000e+00 4.25784e-07   4 */digital_core_instance/peripherals_i/core_clock_gate_30M
  564     0.76% 1.09569e-09 1.54125e-07 3.52498e-07 1.58746e-07   4 */digital_core_instance/peripherals_i/apb_soc_ctrl_i
    2     0.00% 3.06400e-12 1.42723e-07 5.97909e-09 1.48705e-07   4 */digital_core_instance/peripherals_i/genblk1[2].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[0].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[1].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[3].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[5].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[6].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[7].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[8].core_clock_gate
    2     0.00% 3.06400e-12 1.42044e-07 0.00000e+00 1.42047e-07   4 */digital_core_instance/peripherals_i/genblk1[9].core_clock_gate
 9963    13.41% 1.85831e-08 0.00000e+00 0.00000e+00 1.85831e-08   4 */digital_core_instance/peripherals_i/apb_adc_int_i
 8536    11.49% 1.60698e-08 0.00000e+00 0.00000e+00 1.60698e-08   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_dig_filt_wrap
   66     0.09% 8.92940e-11 0.00000e+00 0.00000e+00 8.92940e-11   5 */digital_core_instance/peripherals_i/apb_adc_int_i/i_tempadc_ctrl
 2351     3.16% 3.83420e-09 1.45599e-08 0.00000e+00 1.83941e-08   4 */digital_core_instance/peripherals_i/apb_spi_master0_i
  844     1.14% 1.17613e-09 6.75276e-09 0.00000e+00 7.92890e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_spictrl
  398     0.54% 6.48008e-10 3.23813e-09 0.00000e+00 3.88614e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_axiregs
  508     0.68% 9.37516e-10 1.75730e-09 0.00000e+00 2.69481e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_rxfifo
  504     0.68% 9.35765e-10 1.75730e-09 0.00000e+00 2.69306e-09   5 */digital_core_instance/peripherals_i/apb_spi_master0_i/u_txfifo
 2356     3.17% 3.84150e-09 0.00000e+00 0.00000e+00 3.84150e-09   4 */digital_core_instance/peripherals_i/apb_spi_master1_i
  857     1.15% 1.17981e-09 0.00000e+00 0.00000e+00 1.17981e-09   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_spictrl
  504     0.68% 9.38833e-10 0.00000e+00 0.00000e+00 9.38833e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_rxfifo
  502     0.68% 9.32389e-10 0.00000e+00 0.00000e+00 9.32389e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_txfifo
  396     0.53% 6.53682e-10 0.00000e+00 0.00000e+00 6.53682e-10   5 */digital_core_instance/peripherals_i/apb_spi_master1_i/u_axiregs
 2005     2.70% 3.02508e-09 0.00000e+00 0.00000e+00 3.02508e-09   4 */digital_core_instance/peripherals_i/apb_timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[0].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[1].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[2].timer_i
  476     0.64% 7.44967e-10 0.00000e+00 0.00000e+00 7.44967e-10   5 */digital_core_instance/peripherals_i/apb_timer_i/TIMER_GEN[3].timer_i
 1168     1.57% 1.90822e-09 0.00000e+00 0.00000e+00 1.90822e-09   4 */digital_core_instance/peripherals_i/apb_uart_i
  338     0.45% 6.00414e-10 0.00000e+00 0.00000e+00 6.00414e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_fifo_i
  346     0.47% 5.96447e-10 0.00000e+00 0.00000e+00 5.96447e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_fifo_i
  185     0.25% 2.69233e-10 0.00000e+00 0.00000e+00 2.69233e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_rx_i
  157     0.21% 2.17725e-10 0.00000e+00 0.00000e+00 2.17725e-10   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_tx_i
   22     0.03% 3.33940e-11 0.00000e+00 0.00000e+00 3.33940e-11   5 */digital_core_instance/peripherals_i/apb_uart_i/uart_interrupt_i
  444     0.60% 6.60363e-10 1.00680e-09 0.00000e+00 1.66716e-09   4 */digital_core_instance/peripherals_i/apb_gpio_i
  426     0.57% 7.56761e-10 0.00000e+00 0.00000e+00 7.56761e-10   4 */digital_core_instance/peripherals_i/apb_i2c_i
  310     0.42% 5.56458e-10 0.00000e+00 0.00000e+00 5.56458e-10   5 */digital_core_instance/peripherals_i/apb_i2c_i/byte_controller
  237     0.32% 3.88994e-10 0.00000e+00 0.00000e+00 3.88994e-10   4 */digital_core_instance/peripherals_i/apb_analog_ctrl_i
  149     0.20% 2.07526e-10 0.00000e+00 0.00000e+00 2.07526e-10   4 */digital_core_instance/peripherals_i/apb_pmu_ctrl_i
-------------------------------------------------------------------------------------------------------------------------------
```