FPGA implementation of a deep learning algorithm for real-time signal reconstruction in particle detectors under high pile-up conditions

J.L. Ortiz Arciniega\textsuperscript{a,1}, F. Carrió\textsuperscript{b} and A. Valero\textsuperscript{b}

\textsuperscript{a}Universitat de València, Avinguda de l’Universitat s/n, Country, Burjassot, Spain
\textsuperscript{b}Instituto de Física Corpuscular (Universitat de València-CSIC), Catedrático José Beltrán 2, Paterna, Spain

E-mail: orarjo@alumni.uv.es

Abstract: The analog signals generated in the read-out electronics of particle detectors are shaped prior to the digitization in order to improve the signal to noise ratio (SNR). The real amplitude of the analog signal is then obtained using digital filters, which provides information about the energy deposited in the detector. The classical digital filters have a good performance in ideal situations with Gaussian electronic noise and no pulse shape distortion. However, high-energy particle colliders, such as the Large Hadron Collider (LHC) at CERN, can produce multiple simultaneous events, which produce signal pileup. The performance of classical digital filters deteriorates in these conditions since the signal pulse shape gets distorted. In addition, this type of experiments produces a high rate of collisions, which requires high throughput data acquisition systems. In order to cope with these harsh requirements, new read-out electronics systems are based on high-performance FPGAs, which permit the utilization of more advanced real-time signal reconstruction algorithms. In this paper, a deep learning method is proposed for real-time signal reconstruction in high pileup particle detectors. The performance of the new method has been studied using simulated data and the results are compared with a classical FIR filter method. In particular, the signals and FIR filter used in the ATLAS Tile Calorimeter are used as benchmark. The implementation, resources usage and performance of the proposed Neural Network algorithm in FPGA are also presented.

Keywords: Data processing methods; Pattern recognition, cluster finding, calibration and fitting methods; Simulation methods and programs

\textsuperscript{1}Corresponding author.
1 Introduction

Particle detectors are used to study the shower of particles produced in high-energy particle colliders. The particles crossing the detectors produce signals that are gathered by the data acquisition electronics to measure their track, momentum, energy or time.

Calorimeters measure the energy of the particles by absorbing completely their energy [1]. The read-out electronics produce analog pulses proportional to the energy deposited, which are conditioned and shaped prior to the digitization in order to improve the signal to noise ratio (SNR).

The real amplitude of the analog signal is then obtained using digital filters, which provides information about the energy deposited in the detector. The classical digital filters have a good performance in ideal situations with Gaussian electronic noise and no pulse shape distortion. Due to its simplicity, they can be executed in real time in sequential Digital Signal Processors (DSPs) in relatively low rate experiments without introducing dead-time.

However, high-energy particle colliders, such as the Large Hadron Collider (LHC) [2] at CERN, can produce multiple simultaneous events, which produce high event rates and signal pileup. The performance of classical digital filters deteriorates in these conditions since the signal pulse shape gets distorted. In addition, this type of experiments produces a high rate of collisions requiring high throughput and deterministic and low latency data acquisitions systems.

In order to cope with these harsh requirements, new read-out electronics systems are based on high-performance FPGAs, which permit the utilization of more advanced real-time signal reconstruction algorithms to exploit the parallelization capabilities of FPGAs.
A deep learning method has been studied for real-time signal reconstruction in high pileup and high event rate particle detectors. The algorithm is trained offline using simulated data, which emulates the signal conditions in terms of noise and pileup of the experiment. Then, the trained network is loaded into the FPGA for real time operation.

2 Classical signal reconstruction algorithms

In classical Calorimeters the signals produced in the detector are conditioned, shaped and digitized in the front-end electronics synchronously with the bunch crossing. The digital samples are stored in memories and only part of the events, which are selected by a trigger system, are transmitted to the back-end for signal reconstruction. In the ATLAS Tile Hadronic Calorimeter case, the photomultipliers signals are digitized every 25 ns and upon the reception of the trigger acceptance signal, 7 digital samples per channel are transmitted to the back-end electronics at a maximum event rate of 100 kHz. Then, digital filters are used to obtain the real amplitude of the analog pulse, which is proportional to the energy deposited in the detector. Figure 1 shows a typical pulse shape and the digital samples generated in the ATLAS Tile Hadronic Calorimeter [3] in the LHC at CERN. A digital FIR filter called Optimal Filtering [4] is used to obtain the amplitude, phase and baseline (pedestal) of the pulse. The Optimal Filtering method is a FIR filter which exploits the knowledge of the pulse shape of the electronics and the amount of expected pileup to reduce the contribution of noise and determine the time of deposition. The amplitude of the pulse is obtained as a linear combination of the digital samples \( S_i \) and weights \( w_i \) computed from the known pulse shape and electronic and pileup noise:

\[
A = \sum_{i=0}^{n_{\text{samples}}} S_i w_i
\]

Thus, any pulse shape distortion strongly affects the performance of the Optimal Filtering method. In particular, signal pileup different than expected deforms the signal of interests and biases the results of the signal reconstruction.

This algorithm is executed in the back-end electronics in Digital Signal Processors (DSPs). In TileCal, each DSP processes the data of one TileCal detector module instrumented with 48 channels. Then, for every triggered event the DSP receives 7 digital samples for 48 channels at the maximum trigger rate of 100 kHz. Then, the FIR filter is executed sequentially to reconstruct the amplitude and calibrate the energy. The result must be provided in less than 10 \( \mu s \) determined by the event rate not to introduce dead-time in the detector.

2.1 Signal reconstruction in high pileup and high rate environments

The performance of the digital filters explained in the previous section significantly deteriorates in the presence of pulse pileup since they rely in the knowledge of the pulse shape.

This means that a new signal reconstruction algorithm capable of supporting large signals pileup and with low latency is required to operate under these conditions. A new algorithm based on supervised learning, specifically on Artificial Neural Networks (ANN) has been studied. Previous studies showed that this type of algorithms could be used for event selection in particle
detectors [5]. We are proposing a new application for detector signal reconstruction before any event selection is applied.

The proposed method has been implemented to use 9 digital samples, which shows the most optimal utilization of FPGA resources in terms of performance.

In addition, the idea is to use the TileCal case as a benchmark. Thus, the new method has been implemented to reconstruct up to 48 channels in parallel in one single FPGA. This method provides reduced and deterministic latency, thanks to the characteristics of the ANN that allow great interconnection and parallelism.

Moreover, the idea is to exploit the parallelism and high performance of FPGAs to reconstruct the detector signals for every bunch crossing at 40 MHz before any trigger selection is applied. Then, the result of the reconstruction can be used as input for the trigger selection.

3 Neural Network algorithm for signal reconstruction

3.1 Neural Network model and architecture

For the design of the ANN, the neuronal model proposed by McCulloch and Pitts in 1943 [6] has been used, which despite being very old is still the most extended for the different architectures of ANNs. In this model, whose schema is shown in figure 2, the output of the neuron represented by $y$ is obtained by applying an activation function $g$ to the weighted sum among the multiple input signals $\{x_1, x_2, \ldots, x_n\}$ and their corresponding synaptic weights $\{w_1, w_2, \ldots, w_n\}$ ± a bias $\theta$ that allows us to specify an appropriate threshold. The following expressions synthesize the result produced by the artificial neuronal model proposed by McCulloch and Pitts:

\[
\begin{align*}
  u &= \sum_{i=1}^{n} w_i \cdot x_i \pm \theta \\
  y &= g(u)
\end{align*}
\]
The most optimal activation function in complex nonlinear systems is the sigmoidal, because it allows gradual processing responses where the output results are real numbers between 0 and 1. In this work, a variant of this function is used, due to the need for negative results given to the characteristics of the system. Its mathematical expression is as follows:

\[ g(u) = \frac{2}{1 + e^{-2u}} - 1 \]

A multilayer feedforward architecture is used, since it is optimized in the identification of systems or signals, where the neurons are organized forming 3 different types of layers: input layer, hidden layers and output layer. Another characteristic of this type of architecture is that the learning algorithm follows a supervised training, which means that synaptic weights and biases of the network are calculated from a set of inputs and desired outputs (targets).

The design of the ANN is done through the Neural Network Fitting application of MATLAB [7], which allows developing a multilayer feedforward network and generates the corresponding code for its subsequent optimization. The first step corresponds to selecting the inputs and targets of the network. The dataset used both for training and to evaluate the performance of the algorithms is generated using a pulse simulator. The simulator generates pseudorandom pulse amplitudes simulating the signals produced by the energy deposited by particles crossing the detector (true values and targets) and then produces a byte-stream emulating the digital samples produced by the electronics. Following this, the dataset is divided into training, validation and test subsets to proceed to select the number of hidden layers and train the network.

Three different algorithms are used to train the network and their performance is evaluated comparing the Mean Square Error (MSE), which must be close to 0, and the correlation between the target and the obtained result (R-square), where values are sought close to 1. This comparison, presented in table 1, provides an optimal view of which algorithm the network should be trained, the ANN is composed of a single hidden layer.

It can be concluded that the most optimal result is obtained by using the Bayesian Regularization training algorithm.
### Table 1. Comparison of training algorithms.

| Algorithms                | MSE  | R-square |
|---------------------------|------|----------|
| Levenberg-Marquardt       | -0.54| 0.9875   |
| Bayesian Regularization   | -0.4615| 0.9875  |
| Scaled Conjugate Gradient | 4.3820| 0.9874   |

#### 3.2 Implementation of the Neural Network algorithm in FPGA

The code corresponding to a feedforward network with 1 hidden layer, trained with the Bayesian Regularization algorithm and generated by the Neural Network Fitting application, describes the sequential steps carried out by the ANN. This includes three functions to normalize the inputs, activate the hidden layer by means of a variant of the sigmoidal function and denormalize of the outputs. In addition, the code includes some constants called “Neural Network Constants” that correspond to synaptic weights and network biases.

The implementation of the algorithm is done for the Xilinx Virtex-7 XC7VX485T FPGA, which is the model used for the upgrade of the TileCal readout electronics [8]. The first step is to convert the code generated by the Neural Network Fitting application to synthesizable VHDL and Xilinx Series 7 FPGA cores. Using the HDL Coder application, also from MATLAB, the functions and operations using floating point arithmetic are identified, and then optimized to fixed-point logic in order to obtain synthesizable code.

The normalization and denormalization functions are simplified to basic mathematical operations, since they only perform addition, subtraction and multiplication using the “Neural Network Constants”. On the other hand, the variant of the sigmoidal function (activation function) performs complex operations, as divisions and exponentials, and they are implemented as Look-Up Tables (LUT) containing the necessary values not to lose resolution.

The designed LUT is a vector that contains all the possible values of the activation function within a range of [-1, 1.2]. The resolution of the algorithm was studied for different LUT sizes in order to optimize the memory resources used in the FPGA to store the LUT values. Table 2 shows the regression and RMSE for different sizes of the LUT. The most optimal result was obtained for 5,000 samples, which present a correlation between the true and the reconstructed result of 98.79%, being only 0.03% lower than a LUT with 20,000 samples.

The fixed-point algorithm using the LUT for the activation function is implemented for the Virtex-7 FPGA including additional data flow and control blocks to obtain the final specifications:

- A digital pulse must be reconstructed every 25 ns (40 MHz).
- Each reconstructed pulse is composed by the last 9 samples received which are obtained with a 9-position shift register.
- Weights resolution in 18-bit values to fit in the DSP slices.
- The inputs are normalized and sent to the first layer of the network (hidden layer).
Table 2. Comparison of different sizes of LUT.

| Number of Samples | RMSE      | Regression (%) |
|-------------------|-----------|----------------|
| 20,000 (Original algorithm) | 45.9167   | 98.829         |
| 10,000            | 45.8876   | 98.813         |
| 5,000             | 46.0348   | 98.799         |
| 1,000             | 48.4637   | 98.721         |
| 500               | 51.4995   | 97.712         |
| 100               | 57.80     | 96.803         |

• In the hidden layer, the mathematical operations corresponding to the neuronal model of McCulloch and Pitts are performed and the output of the layer is activated using the LUT [7].

• The output of the hidden layer serves as input to the second layer of the network (output layer), where a process similar to the previous layer is performed. In this case, the output is denormalized to obtain the initial scale.

• An output result is obtained after one clock period, so that the latency is minimal and deterministic.

Figure 3 shows the process carried out by the Artificial Neural Network through its two layers:

The ANN algorithm is executed in pipeline mode with 5 steps running at 40 MHz (figure 3). Thus, the initialization time takes 5 clock cycles (125 ns). After this initialization time, it provides one output every 25 ns. This initialization time could be reduced using a faster frequency for the execution of the 5 steps. However, this initialization time meets the requirements of this application since it takes few clock cycles, fixed and deterministic.

The training of the ANN affects the values of the 10 weights, gain1 and gain2, which can be updated without loading a new firmware. The implementation in the FPGA uses a total of 13 DSP slices per channel: 2 in Step 1, 9 in Step 2 and 1 in Step 3.
4 Results

The performance of the proposed ANN algorithm has been studied using a data simulator, which permits the comparison between the obtained result and the true input. The TileCal Optimal Filtering algorithm is used as a figure of merit to evaluate the performance of the proposed method. The study has been done in MATLAB using the optimized fixed-point arithmetic version of the algorithm. Finally, the analysis has been replicated in the FPGA to validate the implementation and to qualify the migration of the algorithm to FPGA resources.

4.1 Data simulator

In order to verify the correct functioning of the algorithm in MATLAB, it is necessary to develop a signal data simulator. The simulator generates pseudorandom pulse amplitudes following a normalized Gaussian distribution covering the entire amplitude range. For every input amplitude the simulator generates the corresponding digital samples emulating the pulse shaping of the electronics. For this study the TileCal pulse shape has been used in order to use the Optimal Filtering reconstruction result as reference. In the real detector, the possibility of having energy deposited in a channel in two consecutive events (pileup) is proportional to the accelerator instantaneous luminosity. The data simulator has been implemented to emulate the signals of the most exposed cell in the TileCal detector during the High Luminosity LHC where up to 200 collisions per bunch crossing are expected. In addition, it permits to simulate non-pileup conditions to study the performance of the algorithm at the beginning of a bunch train or at the end of accelerator fills where the luminosity decreases significantly. For the high pileup condition the generated data set has non-zero amplitude in approximately 54% of the events figure 4 shows the amplitude distribution for the high (left) and low (right) pileup cases. In addition, a Gaussian pseudo-random electronic noise is added to each generated sample to simulate the real electronics. The RMS of the noise distribution is 1.5 ADC-counts compatible with the TileCal electronics (figure 5). Finally, the different time arrival of particles to the detector is also simulated adding small phase variations to the signals generated (figure 5).

The simulator is used to train the Neural Network and then to evaluate the performance of the reconstruction with a different data set.

![Simulator amplitude distribution for high (left) and low (right) pileup conditions.](image)

**Figure 4.** Simulator amplitude distribution for high (left) and low (right) pileup conditions.
4.2 Signal reconstruction results

The study has been done using a generated input data of 200,000 events emulating the same number of consecutive collisions in the detector. Then, each event has been reconstructed using both the Optimal Filtering and the ANN methods and the result is compared with the generated target value.

The FIR filter weights and the training of the Neural Network can be optimized for the different pileup conditions. Figure 6 shows the correlation between the target and the reconstructed amplitude for high pileup data when the two methods are optimized for this high pileup condition. The R-square of the linear regression fit for the FIR filter is 0.78 and 0.97 for the ANN method.

The ANN algorithm is trained to detect pulses and suppress pileup and electronic noise whereas the FIR filter would require additional logic to detect peak pulses and perform zero suppression. This effect is observed for small target amplitudes and may distort the linear regression fit in particular in the case of the FIR method. In order to have a better comparison between the two methods, only events with a target above 10 ADC-counts were studied.
Figure 7 shows the results of the reconstruction for the two methods for target amplitudes greater than 10 ADC-counts. The R-square of the linear regression fit improves slightly in both cases with 0.81 for the FIR filter and 0.98 for the ANN method.

![Figure 7](image1)

**Figure 7.** Reconstructed versus target amplitudes for the FIR (left) and ANN (right) methods for high pileup data set when the two methods are optimized for high pileup conditions and the target amplitude is greater than 10 ADC-counts.

The performance of the two methods has been studied for a non-pileup data set in order to study the effect of variations in the pileup conditions during the fill or due to the beam train structure. Figure 8 shows the result of the reconstruction of the FIR (left) and ANN (right) methods for a non-pileup data set when the methods are optimized for high pileup conditions. In this case, both methods perform similarly with an R-square of the linear regression fit of 0.99.

![Figure 8](image2)

**Figure 8.** Reconstructed versus target amplitudes for the FIR (left) and ANN (right) methods for no pileup data set when the two methods are optimized for high pileup conditions and the target amplitude is greater than 10 ADC-counts.

During calibration data taking and in some physics special runs where no pileup effect is expected throughout the entire run, it may be useful to use a special optimization in the reconstruction algorithms in order to gain accuracy. Figure 9 shows the result of the reconstruction for a data set with no pileup when the two methods are optimized for these conditions. Both methods have an
optimal performance in this case with an R-square value of exactly 1.0 (figure 9). However, the linear fit and the number of outliers show a better performance of the ANN method with respect to the FIR filter.

Figure 9. Reconstructed versus target amplitudes for the FIR (left) and ANN (right) methods for no pileup data set when the two methods are optimized for no pileup conditions and the target amplitude is greater than 10 ADC-counts.

4.3 Implementation of the Neural Network method in FPGA

The same data set used for the performance study in MATLAB has been used to evaluate the behavior of the algorithm in the FPGA. The data set has been introduced in an internal FPGA memory and then the reception of the digital samples at 40 MHz has been simulated. The code has been implemented for one channel and then scaled for 48 channels, which represents one TileCal detector module.

The results obtained in the FPGA for the 200,000 events in the data set match exactly the results obtained in MATLAB with the fixed-point arithmetic version of the algorithm. Therefore, the implementation of the proposed algorithm in FPGA does not present any limitation. In addition, the number of FPGA resources used is much reduced except for the number of DSP slices, where the bulk of the processing is done (table 3). Thus, the parallelization could be increased even further to process more than 48 channels in parallel.

The FIR filter algorithm running in DSP ICs in the current system has been implemented in the same FPGA model in order to compare the resource utilization and performance of the two methods. The FIR filter is a much simpler algorithm and uses less FPGA resources than the ANN method. It also shows smaller latency for the same operation frequency. However, both the ANN and FIR filter meet the requirements of the HL-LHC to operate at 40 MHz with reduced and deterministic latency.

5 Conclusions

An algorithm based on neural networks has been proposed for real-time signal reconstruction in high pileup and high rate particle detectors. The performance of the new method has been studied using
Table 3. Comparison of the resources used by the two methods in the FPGA.

| Resource summary | Available | ANN | OF-FPGA | OF-DSP |
|------------------|-----------|-----|---------|--------|
| Slice registers  | 607,200   | 695 | 98      | -      |
| Slice LUTs       | 303,600   | 1297| 87      | -      |
| RAM blocks       | 1,030     | 4   | 0       | -      |
| DSPs             | 2,800     | 13  | 3       | -      |

Performance summary

| Operation mode | Parallel (pipelined) | Parallel (pipelined) | Sequential |
|----------------|----------------------|----------------------|------------|
| Latency        | -                    | 125 ns               | 25 ns      | 10 μs     |
| Operation frequency | -                | 40 MHz               | 40 MHz     | 100 KHz   |

Simulated data that emulates the harshest possible pileup conditions on a detector. The Optimal Filtering digital filter used in the ATLAS Tile Calorimeter has been used as benchmark to evaluate the performance and behavior of the Neural Network method.

The obtained results show better performance of the new Neural Network compared with the Optimal Filtering method in all the cases studied. In particular, the regression obtained for the Neural Network method is 0.98 while it is 0.81 for the Optimal Filtering for a high pileup conditions and when the two methods are optimized for these conditions. Moreover, the ANN method is trained to zero suppress while the Optimal Filtering would require additional logic to detect pulse peaks.

The utilization of FPGA resources for the ANN method is higher than for Optimal Filtering case but in both cases, it represents a small percentage of the total resources available in the FPGA used.

The latency measured from the reception of the last used sample and the output of the result is 5 clock cycles for the ANN method and one cycle for the Optimal Filtering. Running the code synchronously with the LHC clock (40 MHz) it represents 125 ns for the ANN method and 25 ns for Optimal Filtering. Nevertheless, the two methods are pipelined data-parallel algorithms, thus they do not introduce dead-time during the data processing at 40 MHz.

This latency could be reduced further using a faster operation frequency. The result of the timing estimation of the FPGA compiler concludes that it could be executed four times faster (160 MHz) meeting the timing constraints.

Acknowledgments

We acknowledge the support of the Ministerio de Economía y Competitividad (Spain) and the European Regional Development Found (ERDF), project FPA2015-65652-C4-2R.

References

[1] R. Wigmans, *Calorimetry: Energy measurement in particle physics*, *Int. Ser. Monogr. Phys.* 107 (2000) 1.

[2] CERN, *The Large Hadron Collider*, available at: https://home.cern/topics/large-hadron-collider (2014).
[3] ATLAS collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, 2008 *JINST* **3** S08003.

[4] A. Valero et al., *ATLAS TileCal Read Out Driver production*, 2007 *JINST* **2** P05003.

[5] J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, 2018 *JINST* **13** P07027.

[6] W. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, *Bull. Math. Biol.* **5** (1943) 115.

[7] MathWorks, *Neural Net Fitting*, available at: https://es.mathworks.com/help/nnet/ref/neuralnetfitting-app.html#bvdhano-1 (2018).

[8] F. Carrió, P. Moreno and A. Valero, *Performance of the Tile PreProcessor Demonstrator for the ATLAS Tile Calorimeter Phase II Upgrade*, 2016 *JINST* **11** C03047.