Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs

Johann Christoph Voigt[1,*] on behalf of the ATLAS Liquid Argon Calorimeter Group

[1]IKTP, TU Dresden, Germany

Abstract. With the High-Luminosity upgrade of the LHC, the number of simultaneous proton-proton collisions will be increased to up to 200. This requires an extensive overhaul of the detector systems. For the ATLAS Liquid Argon calorimeter electronics, 556 high performance FPGAs will be installed to reconstruct the energy for all 182 468 cells at the LHC bunch crossing frequency of 40 MHz. However, the current digital filter used for energy reconstruction (optimal filter) decreases in performance under these high pileup conditions.
We demonstrate, that small recurrent or convolutional neural networks can outperform the optimal filter. Prototype implementations of the respective inference code in VHDL show, that the use of these networks on FPGAs is feasible and the resulting firmware fits onto the planned Intel Agilex devices. The full design is capable of processing 384 detector cells per FPGA, by combining parallel instances of the firmware with time division multiplexing.

1 Introduction

The Liquid Argon (LAr) calorimeters of the ATLAS detector [1] are a set of sampling calorimeters covering a pseudorapidity range of $|\eta| < 4.9$, where the electromagnetic barrel and end-cap regions span $|\eta| < 3.2$. During Run 4 from 2029 onwards, the High-Luminosity upgrade of the Large Hadron Collider (LHC) [2] will increase the mean number of simultaneous proton-proton collisions $\langle\mu\rangle$ observed by ATLAS to 140-200. Since a single LAr signal pulse has a duration of about 25 bunch crossings (BC), the increased pileup is especially challenging for the LAr calorimeters due to the resulting overlap. Combined with the increased trigger rate and a new trigger scheme that allows the acceptance of events from consecutive bunch crossings, this requires a replacement of most of the LAr readout electronics.

The upgraded readout foresees an analog pulse shaping, amplification and analog-to-digital conversion on the detector for two gains in parallel. The front-end connects to the off-detector electronics through optical fibers, where 556 Intel Agilex-40 series FPGAs will be installed to allow the energy reconstruction at the full granularity of 182 468 cells at the ADC frequency of 40 MHz.

For the digital signal reconstruction, the current readout uses an optimal filter [3]. This is a linear filter that is optimized to reconstruct the energy deposited in one detector cell, accounting for noise and pileup. It is usually used with five coefficients multiplied with five pulse samples, which equals the field of view. However, the optimal filter struggles with overlapping signals. With the increased pileup and revised trigger scheme, this will

*e-mail: johann_christoph.voigt@tu-dresden.de

become more common and new algorithms for the digital energy reconstruction are under consideration.

2 Network training and architecture

It has been shown previously, that both recurrent neural networks (RNNs) and convolutional neural networks (CNNs) can outperform the optimal filter especially in the case of overlapping signals [4]. This paper further explores these two artificial neural network (ANN) architectures as potential replacements for the optimal filter.

The ANNs are trained using the Keras [5] API to Tensorflow on signal-enriched simulated detector pulses generated by the AREUS [6] software. This allows the use of the true energy deposit as a training target and during evaluation.

The first approach uses a vanilla RNN architecture with a sliding input window. As Figure 1 illustrates, the network consists of five RNN cells. Each cell uses the ADC value from one time step of the time series as input. The network always operates on input data from only one detector cell. The number of RNN cells is equal to the field of view. This means that the network can see a time window of five bunch crossings. The RNN cells are linked by an internal state vector of dimension 8. In total the network contains 304 multiplications. The limited field of view is an important property to make sure that the network has a finite impulse response. Otherwise, any erroneous input data could have long term effects on the output.

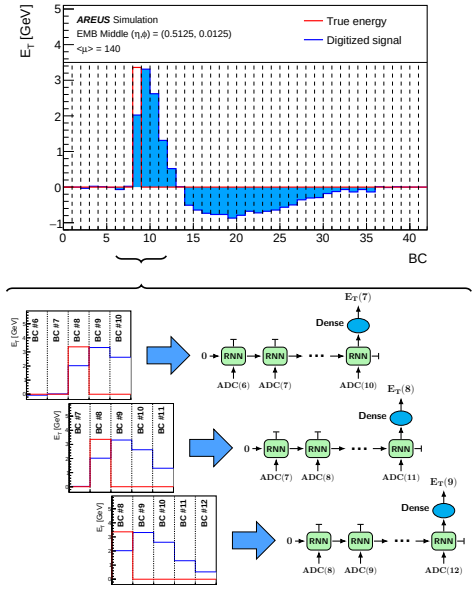

Figure 1. Recurrent neural network (RNN) using a sliding window approach. The network consists of five vanilla RNN cells in a chain, that passes an internal vector of dimension 8 along. The energy extraction happens through a dense neuron at the last cell. Each RNN cell uses one out of the five consecutive input samples. The network is computed at every bunch crossing on the current input window. [7]

The other approach uses CNNs, which operate on the continuous time data coming from the ADC of one detector cell as well. The simpler internal structure compared to the RNNs allows for a larger field of view of approximately 20 bunch crossings. The first CNN architecture under consideration contains two convolutional layers (2-Conv) for the energy reconstruction, as can be seen in Figure 2. A rectified linear unit (ReLU) is used as the activation function. The second layer utilizes dilation in order to increase the field of view without increasing the number of parameters.

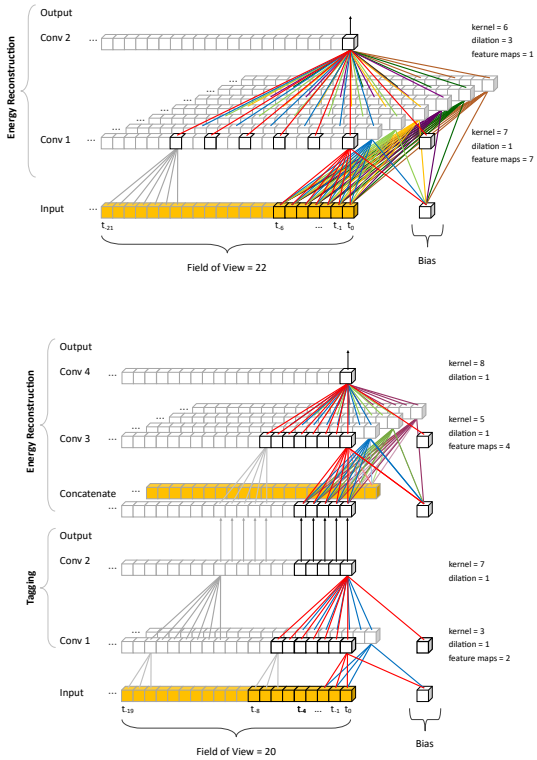

Figure 2. Top: Architecture of a convolutional neural network (CNN) with two layers. The data flow goes from bottom to top. The input sequence is first processed by a convolutional layer with dilation 1, which densely screens its input samples. The second convolutional layer with dilation 3 sparsely screens the output of the previous layer. This results in a larger total field of view of 22 bunch crossings without increasing the number of parameters.
Bottom: Architecture of a CNN with four layers and tagging part. The input sequence is first processed by the tagging part of the network in the bottom half of the figure. After a concatenation layer, the tag output and the input sequence are processed by the energy reconstruction part of the ANN. The total field of view of this network incorporates 20 bunch crossings. [7]

The second CNN architecture (4-Conv) uses two additional layers with the sigmoid activation function at the bottom to tag the undershoot of a previous pulse. The top two layers are again responsible for the energy reconstruction, based on the output from the tagging layers concatenated with the ADC sequence as input. Both networks are constrained to 100 total parameters, therefore the energy reconstruction layers are smaller when using tagging layers.

3 Energy reconstruction performance

Figure 3 shows the energy reconstruction of the optimal filter and a CNN for a challenging example sequence. The optimal filter cannot distinguish between the two very close pulses around bunch crossing 5. At bunch crossing 65, an energy deposit happens within the undershoot of a previous pulse. The optimal filter, with its limited knowledge of the past, cannot correct for this temporary shift in baseline and underestimates the deposited energy. The ANN can distinguish the pulses and properly recovers the full peak within the undershoot.

The performance of the ANNs compared to the optimal filter has been previously demonstrated [4]. For the CNNs, a new study based on updated input samples is presented here. This new sample set includes six characteristic cells from the barrel and electromagnetic end-cap region of the detector. Furthermore, it uses higher energies up to 80 % of the high-gain preamplifier saturation for the artificial signals that are injected on top of the pileup. This leads to a higher required dynamic range of the networks.

Because the energy reconstruction of overlapping pulses has been identified as a critical case, this is further evaluated in Figure 4. The histograms show the deviation of the reconstructed energy over the distance between two consecutive high energy deposits. For the

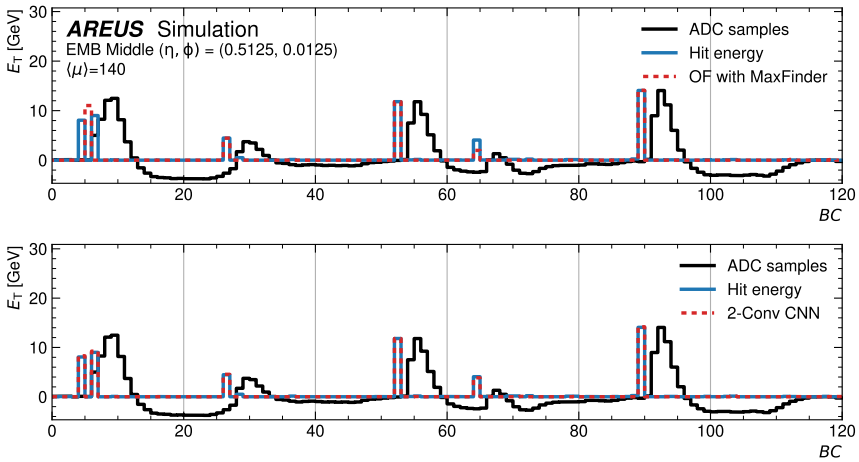

Figure 3. Top: Display of a sample sequence (black) of an EMB Middle cell at (η, ϕ) = (0.5125, 0.0125), together with the true transverse momentum magnitude E_T (blue) and the optimal filtering (OF) with maximum finder output (red) as a function of the BC counter.
Bottom: Similar display but with the output of the 2-Conv CNN (red). The sequence was simulated with AREUS, using $\langle\mu\rangle$ = 140. True energies include in-time pile-up. The CNN shows an improvement in detecting hits coming in close succession. The two hits with a gap of 2 around bunch crossing 5 are well reconstructed by the CNN. That is not the case for the OF, which detects only a single peak. Around BC 65 a hit follows within the undershoot of a previous hit. While the OF underestimates its energy, the CNN reconstructs it much better. [7]

optimal filter, the energy reconstruction mostly breaks down for pulses closer than 20 bunch crossings. In the case of the CNN, the performance also suffers in this region, but the shape of the histogram is still more symmetric and centered around 0 GeV. The higher density of deposits shows that more pulses are recognized overall compared to the optimal filter. The structure seen in both plots below −0.5 GeV is an artifact of the energy cut applied to suppress low energies and contains all the pulses that were reconstructed as 0 GeV.

Eventually, the goal is to use the networks for the energy reconstruction of all 182 000 detector cells. In order to evaluate whether the same architecture is suitable for the entire detector, six characteristic cells of the barrel and electromagnetic end-cap region have been chosen and the CNN training repeated.

Figure 5 shows the difference between reconstructed and true energy over the distance to the preceding pulse in the input sequence in units of bunch crossings for the six representative cells. The general shape of the histogram is similar for all subplots. The principle approach and network architecture can therefore be used for the barrel and end-cap regions. A strategy on how to handle the training for the full detector still needs to be developed, however.

4 FPGA implementation

4.1 Recurrent neural networks (RNNs)

Initially, hls4ml [8] was used to prototype the FPGA implementation for the RNNs. However, time division multiplexing is required to handle the planned 384 detector cells on one FPGA. The ADC samples the data at a frequency of only 40 MHz, while the FPGA supports much

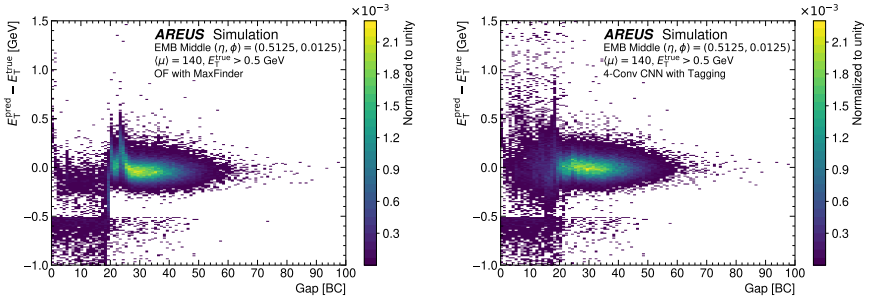

Figure 4. Difference of the reconstructed and true transverse momentum magnitude (E_{T}) as a function of the gap, i.e., the distance in units of bunch crossings (BC), between two consecutive energy deposits for the optimal filtering (OF) algorithm and a subsequent maximum finder (left) and the 4-Conv convolutional neural network (CNN) with tagging (right). The inputs are sampled pulses obtained from the simulation of an EMB Middle LAr cell ($\eta = 0.5125$ and $\phi = 0.0125$) with AREUS using $\langle\mu\rangle = 140$. The structure at $E_{\mathrm{T}}^{\mathrm{pred}} - E_{\mathrm{T}}^{\mathrm{true}} = -0.5\,\mathrm{GeV}$ arises, since only deposits with $E_{\mathrm{T}}^{\mathrm{true}}$ (including in-time pile-up) above 0.5 GeV are considered. [7]

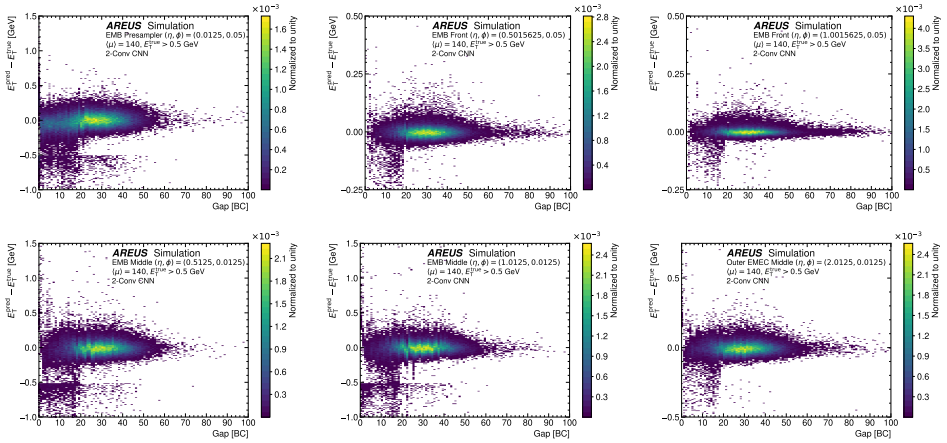

Figure 5. Difference of the reconstructed and true transverse momentum magnitude (E_{T}) as a function of the gap, i.e., the distance in units of bunch crossings (BC), between two consecutive energy deposits for the 2-Conv CNN algorithm for a set of representative detector cells. Inputs to the CNN are sampled pulses obtained from the AREUS simulation using $\langle\mu\rangle = 140$. The structure at $E_{\mathrm{T}}^{\mathrm{pred}} - E_{\mathrm{T}}^{\mathrm{true}} = -0.5\,\mathrm{GeV}$ arises, since only deposits with $E_{\mathrm{T}}^{\mathrm{true}}$ (including in-time pile-up) above 0.5 GeV are considered. [7]

faster processing. The multiplexing concept allows one instance of the RNN firmware to run at a multiple of the ADC frequency and continuously cycle through the input from a fixed number of detector cells.

To better customize the firmware and support multiplexing, the RNN inference was implemented using the Intel high level synthesis (HLS) [10] language directly. This implementation has been used to study the effects of rounding in different parts of the network. The FPGA works most efficiently with 18 bit fixed point numbers, while the training happens in 32 bit floating point numbers. Different rounding or truncation strategies have been evaluated.

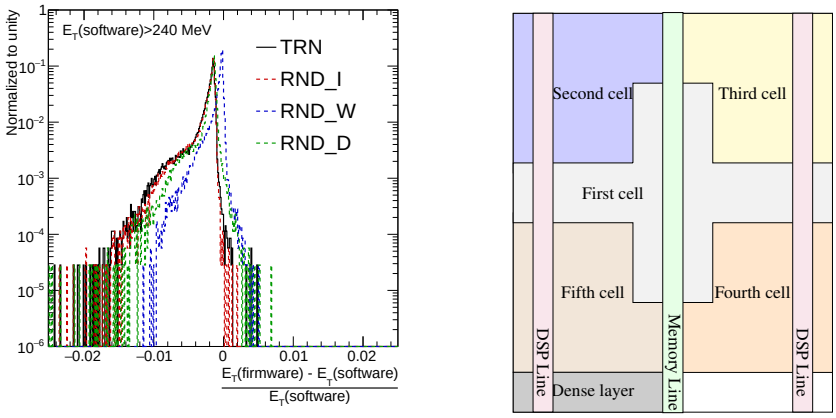

Figure 6. Left: Distribution of the relative deviation between the transverse momentum magnitude E_{T} obtained from the fixed point firmware and the floating point software model for rounding at different stages of the network. The default is truncation, the dashed graphs signify the distribution resulting from rounding of intermediary results (I), weights (W) and input/output data (D).
Right: Placement constraints for one RNN firmware instance. The five RNN cells and the dense layer for extraction are placed to have access to the DSPs and on-device memory and create short signal paths between connected cells. [9]

The most impactful position to apply rounding instead of truncation are the network weights as can be seen in Figure 6 (left) [9]. This can be done in software during pre-processing and therefore has no influence on the FPGA resource requirements. Rounding the input and output data, as well as intermediate results between layers, to 18 bit has a reduced influence compared to the weights.

Finally, a low level implementation directly in VHDL has been created based on the experience gained from the HLS version to further reduce the resource utilization. This allows more optimizations, like the reuse of common results between RNN cells or custom placement constraints as can be seen in Figure 6 (right). The latter allows to define a particular area on the FPGA lattice for every instance of the RNN firmware. Inside this area, the placement of the different RNN cells can be fixed. This makes sure that the signal paths between the cells are short and consistent between multiple parallel instances of the firmware block.

This also allows incremental compilation, where the network is compiled multiple times to improve the signal timings. At each iteration, the parts that meet the timing constraints can be retained, while the problematic regions are recompiled.

4.2 Convolutional neural networks (CNN)

The CNN inference code has been implemented in VHDL directly. This supports multiplexing targeting a factor of 12. To process the full number of 384 cells, a total of 33 parallel instances is required. While 32 instances would be sufficient, the input data arrives via three optical receivers. It is therefore beneficial to follow this symmetry, even if parts of the CNN firmware are not fully utilized this way.

The firmware has a modular structure, where the number of layers and the parameters for each layer can be configured at compile time. The basic building block of the CNN is one filter. The CNN structure can be constructed out of parallel filters per layer and then stacking layers on top of each other.

Table 1. FPGA resource estimates for the Intel Stratix 10 (reference 1SG280HU2F50E2VG) and Agilex (reference AGIC040R39A2E2VR0) devices.

FPGA	Network	Multiplex.	Detector cells	$f_{\max}$	ALMs	DSPs
Stratix-10	RNN (HLS)	10	370	393 MHz	90 %	100 %
	RNN (VHDL)	14	392	561 MHz	18 %	66 %
	2-Conv CNN	12	396	415 MHz	8 %	28 %
	4-Conv CNN	12	396	481 MHz	18 %	27 %
Agilex	2-Conv CNN	12	396	539 MHz	4 %	13 %
	4-Conv CNN	12	396	549 MHz	9 %	12 %

The weights are stored in memory blocks inside the FPGA for each filter individually. This reduces the required signal paths and helps to meet the clock frequency requirement. Significant performance increases, especially for the multiplexed case, have been achieved by storing the network weights in the order required by the multiplier (DSP) chain and not the logical order as it is produced in the network training. This reduces the required logic for routing the weights to the correct DSP significantly without any loss in functionality. It is only required to reorder the weights in software, before loading them onto the FPGA.

The multiplexing logic itself has been redesigned compared to the version previously introduced in [4]. For this particular CNN use case, the multiplexing is functionally similar to a dilation. Using this similarity, the processing can be done on one unified input data stream, where every n-th input sample belongs to the same detector cell with a multiplexing factor of n. This reduces the overhead from any switching logic, as only the weights need to be cycled to make sure that the input data from one cell is always multiplied by the matching weights. Only a slight change in the network architecture is required, by effectively multiplying the dilation of every layer by the multiplexing factor.

It is planned to compile the firmware of the network once and use it for all 556 FPGAs. This means that the network weights will be loaded only afterwards and are not available at compile time. Therefore, it is not possible to use pruning as an optimization technique.

4.3 Compilation results

Table 1 shows the FPGA resource estimates as a percentage of total available resources for the Intel Stratix-10 and Agilex 40-series devices. The resource estimates for the different network architectures are obtained using the Intel Quartus Prime software [10]. The number of detector cells shown is the maximum number of parallel input streams the respective firmware has been compiled for, where 384 are required. This is achieved by using the shown multiplexing factor and parallel instances of the base firmware block. The maximum clock frequency reported by Intel Quartus can be compared to the product of the multiplexing factor and 40 MHz to determine whether the timing constraints are met. Adaptive logic modules (ALMs) are the general purpose logic elements of the FPGA and therefore required for most tasks performed on the FPGA, while the DSPs are specialized multiplication units.

It can be observed that the RNNs do not fit the Stratix-10 even in the optimized HLS version, while the VHDL implementation has sufficiently low resource requirements. The maximum clock frequency is high enough to allow an increased multiplexing factor of 14.

The CNNs still do not consistently reach the required 480 MHz on the Stratix-10. The ALM and DSP consumption is even lower than for the RNNs. The difference between the results for the 2-Conv CNN compared to the 4-Conv CNN with tagging layers illustrates, that the sigmoid activation function used by the tagging layers requires a high number of ALMs.

First compilation results for the Intel Agilex show a significant improvement in the maximum clock frequency such that this is no longer problematic. The resource utilization is reduced to approximately half compared to the Stratix-10. This leaves plenty of space for the other firmware blocks of the project to be placed as well with sufficient margins.

5 Conclusion

Both RNNs and CNNs are shown to be capable of outperforming the optimal filter especially in case of overlapping signals. The results for the CNNs show that the dynamic range of the networks can be extended to cover larger energies compared to previous studies and the approach works for different detector regions. The next step is to demonstrate the effect on higher level object reconstruction, like electrons, photons and jets.

The firmware implementation for both architectures has evolved considerably and now supports placing the full firmware required for processing 384 detector cells on one FPGA. This has been achieved by implementing the RNNs directly in VHDL and redesigning the multiplexing logic of the CNNs. First tests on FPGA hardware are ongoing to verify the firmware outside of simulations.

6 Acknowledgments

This work was in part supported by the German Federal Ministry of Education and Research within the project 05H19ODCA9. The project leading to this publication has received funding from Excellence Initiative of Aix-Marseille Universitè - A*MIDEX, a French *Investissements d'Avenir* programme, AMX-18-INT-006 and from the French *Agence National de la Recherche*, ANR-20-CE31-0013.

References

[1] The ATLAS Collaboration, Journal of Instrumentation **3**, S08003 (2008)

[2] L. Evans, P. Bryant, Journal of Instrumentation **3**, S08001 (2008)

[3] W. Cleland, E. Stern, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **338**, 467 (1994)

[4] G. Aad, A.S. Berthold, T. Calvet, N. Chiedde, E.M. Fortin, N. Fritzsche, R. Hentges, L.A.O. Laatu, E. Monnier, A. Straessner et al., Computing and Software for Big Science **5** (2021)

[5] F. Chollet et al., *Keras*, `https://keras.io`

[6] N. Madysa, EPJ Web of Conferences **214**, 02006 (2019)

[7] ATLAS LAr Calorimeter Group, *Public liquid argon calorimeter plots on upgrade*, `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LArCaloPublicResultsUpgrade`

[8] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran et al., Journal of Instrumentation **13**, P07027 (2018)

[9] G. Aad, T. Calvet, N. Chiedde, R. Faure, E. Fortin, L. Laatu, E. Monnier, N. Sur, Journal of Instrumentation **18**, P05017 (2023)

[10] Intel, *Intel quartus prime software and hls compiler*, `https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html`