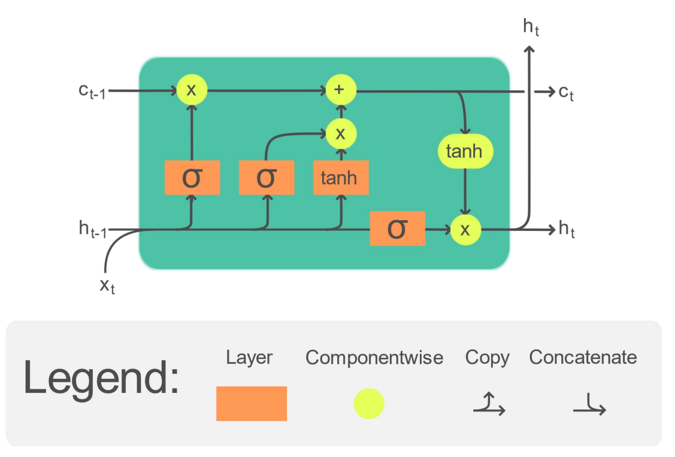
Towards enabling recurrent neural networks in FINN

Author : Shashwat Khandelwal – Ph.D. Student @ Trinity College Dublin

Motivation

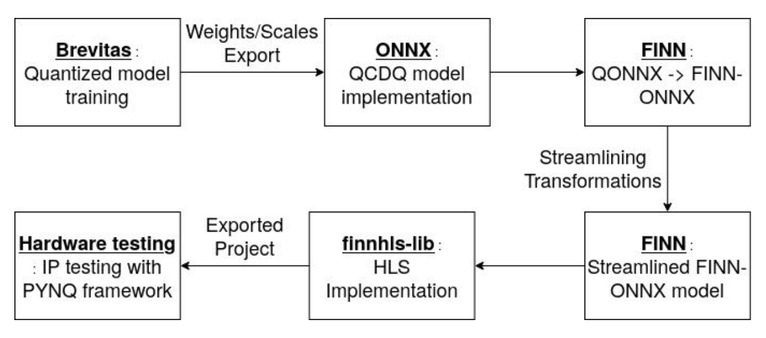
- Recurrent neural networks especially LSTMs have been known to be used for processing time-series data. They extract features from them more effectively than feed forward neural networks like convolutional neural networks (CNN's) or dense networks (MLP's).
- **IF** the success of quantized feed-forward classification models in terms of their accuracy, performance & energy consumption in previous years with frameworks like FINN can be extended to LSTM's, it would make them an attractive alternative for enabling resource-constrained real-time edge applications.

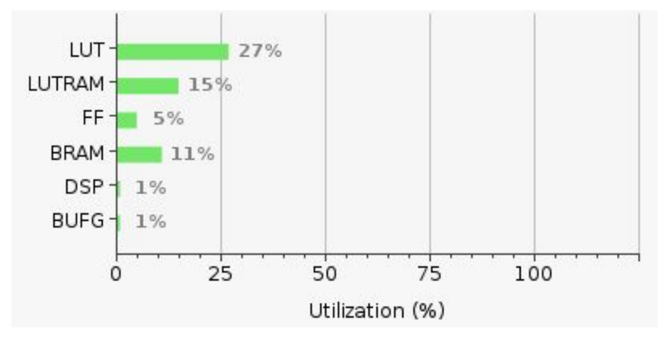


A young boy is playing basketball.
Two dogs play in the grass.
A dog swims in the water.
A little girl in a pink shirt is swinging.
A group of people walking down a street.
A group of women dressed in formal attire.
Two children play in the water.
A dog jumps over a hurdle.
Image Captioning

QLSTM-IDS for Automotive CAN

- Trained (*brevitas*) and deployed (*FINN*) a QLSTM based classification model on the ZCU104 FPGA.
  - The model comprises of an LSTM layer (inp_size = 10, hidden_units = 20) (**W8A6**)('A' refers to internal activations within the QLSTM layer) followed by 3 linear layers (20, 128, 64 dense units) with ReLU activations(**W8A6**). The model comprises of 13,122 trainable parameters.
  - The classification model enables per-message classification between normal and malicious messages on the Controller Area Network (CAN) traffic.
  - The model when tested on 465000 test messages comprising of **DoS, Fuzzing and RPM-Spoofing** attacks had only *4 misclassifications*.
  - The model achieved a per-message processing latency of **0.21ms** and energy utilization of **0.5mJ** per/inference. Design synthesized at **150 MHz**.


Brevitas : Quantized model training
Weights/Scales Export
ONNX : QCDQ model implementation
FINN : QONNX -> FINN-ONNX
Streamlining Transformations
FINN : Streamlined FINN-ONNX model
finnhls-lib : HLS Implementation
Exported Project
Hardware testing : IP testing with PYNQ framework
Model-Deployment Pipeline

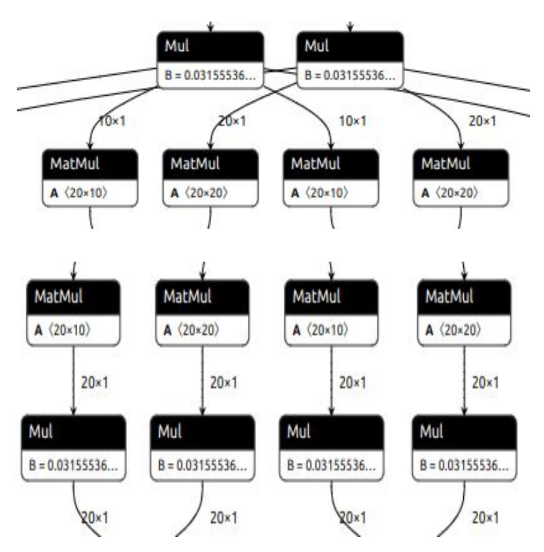


Model-Resource Utilisation (XCZU7EV)

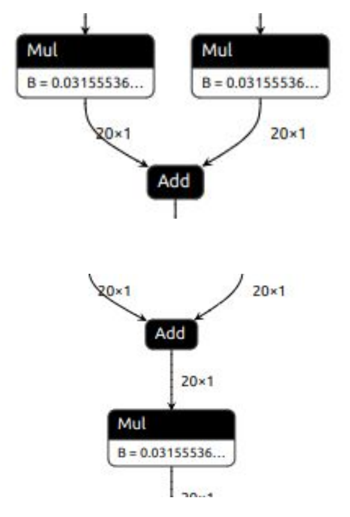
Appendix

Example transformation updates

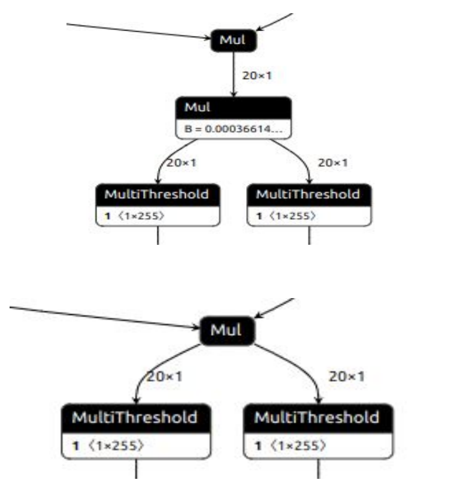
MoveScalarMulPastMul()



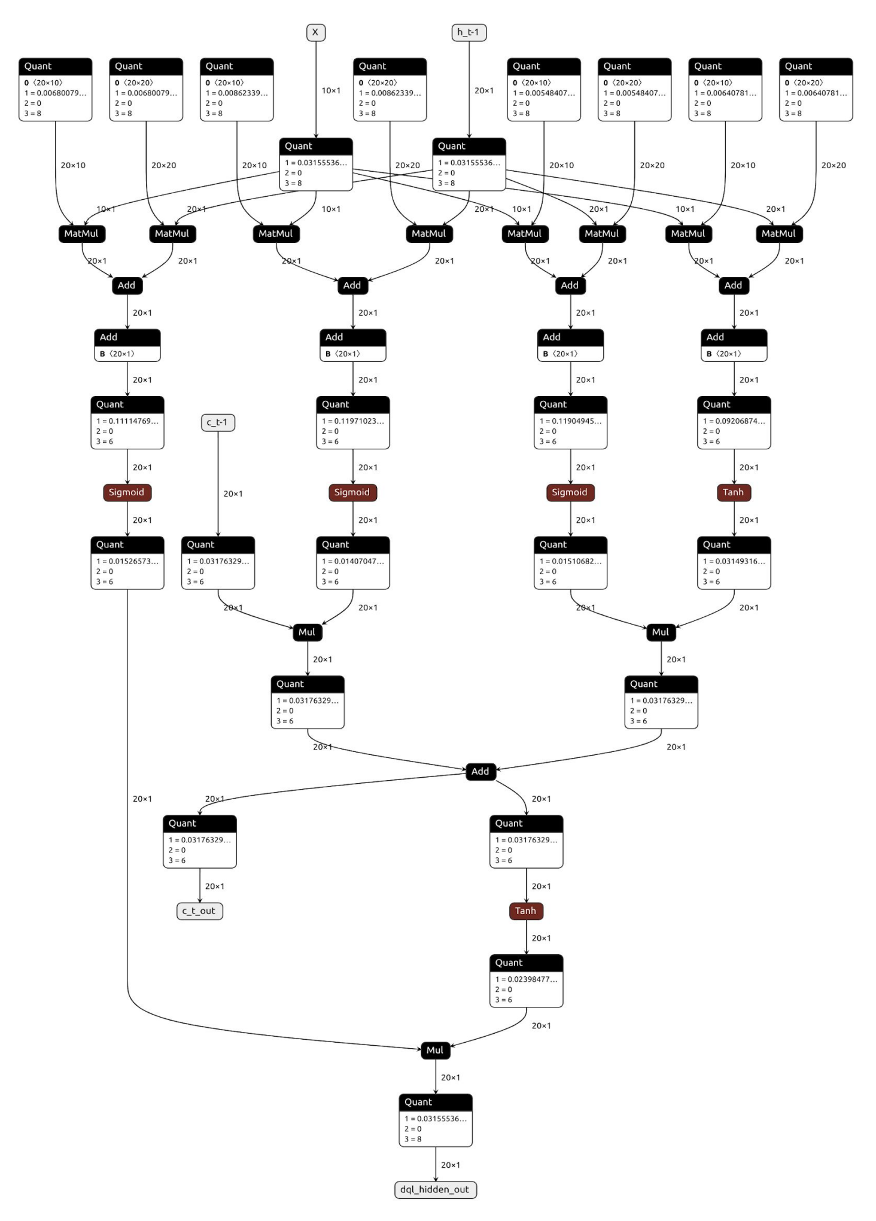
MoveLinearPastEltwiseAdd()



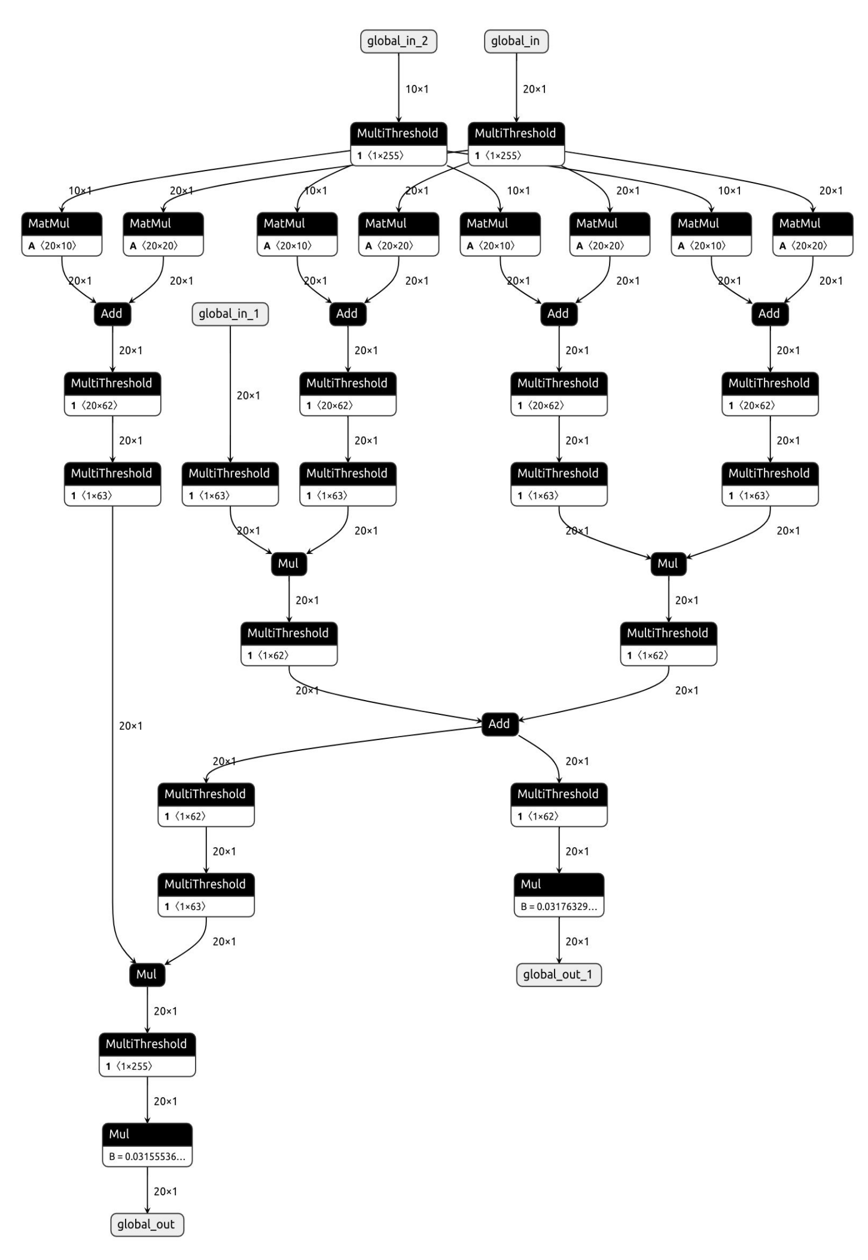
AbsorbMulIntoMultithreshold()



QONNX LSTM compute graph



Streamlined FINN-ONNX compute graph



Key Ideas

Present the implementation of a Q-(quantized)LSTM layer-based classification model on the ZCU104 FPGA for intrusion detection in automotive controller area network. The implementation presents a first-step into the generalized integration of **QLSTM** layer within the FINN framework. The steps involved in the implementation are presented below:

- **ONNX-Frontend** : Simulations & FINN-ONNX representation
  - Use the 'SCAN' operator from ONNX to enable simulation testing of LSTM layer based neural network models with ONNX. This operator enables mixed quantization with the quantizers used in the QLSTM layer in *Brevitas*.
  - The representation of LSTM computation using the SCAN operator is converted to it' s QONNX and FINN-ONNX representation.
  - ***Key Contributions***:
    - LSTM compute encapsulated in the **SCAN** operator in ONNX.
    - Support for threshold generation of '*Tanh*' and '*Sigmoid*' activations functions in FINN (*ConvertQONNXtoFINNONNX*).

- **FINN-Compiler** : FINN-ONNX streamlining
  - The aim of this step was to eliminate as many floating-point operations from the FINN-ONNX graph obtained in the above step.
  - There were some common compute patterns FINN-ONNX graph which were directly reduced (merged into the threshold nodes) with standard transformations in the FINN compiler.
  - For other unique compute patterns that could not be streamlined using existing transformations, new transformations were created or older versions were updated to obtain a compute-graph completely rid of floating-point operations.
  - ***Key Contributions***:
    - The following transformations were modified or introduced: *MoveScalarMulPastMul(), AbsorbAddIntoMultithreshold(), MoveLinearPastEltwiseAdd(), AbsorbMulIntoMultithreshold()*

- **FINN-hlslib :** HLS backend
  - The streamlined computation graph obtained from the above step was implemented in Vitis HLS v2022.2 using the pre-existing hardware blocks from finn-hlslib library.
  - ***Key Contributions***:
    - Built the HLS backend with the finn-hlslib hardware blocks. Demonstrating pre-existing hardware blocks in FINN backend can be used to emulate the computation in this compute graph.

- **Hardware Implementation**: PYNQ
  - This larger design was then exported from Vitis HLS with streaming interfaces and was incorporated in a larger design with data read and write from PS to PL using the DMA IP.
  - This IP was then tested as an overlay using the PYNQ framework to benchmark the latency and energy measurements on the ZCU104 FPGA.

Future-Work

- **Implementation generalization**:
  - Generalizing threshold generation of '*tanh*' and '*sigmoid*' activations for all kinds of inputs and activation bit-widths.
  - Some transformations introduced in the streamlining process were custom for this specific model implementation, making them general again is a goal.
  - Optimizing the HLS backend implementation to reduce resource utilisation of the QLSTM layer.
  - Automating transfer of parameters between each step in the above implementation pipeline to realize a generalized QLSTM implementation in FINN.

- **Quant-GRU:**
  - Given a very similar compute structure of GRUs, a similar implementation pipeline can be established for them in FINN.
  - This can be made possible when a *QuantGRU* layer is introduced in brevitas.

Acknowledgements

I would like to thank the entire team at AMD Dublin for their help in this project. Given the nature of the project, I feel I have contacted everyone at some point during this project duration. Special thanks to Thomas and Jakoba for guiding me through the entire project with immense patience. Also, thanks to Alessandro and Yaman for having doubt sessions with me during the initial phase of the project. I would also like to thank Michaela for giving me an opportunity to be a part of the team as an intern. I have thoroughly enjoyed the entire learning process. Finally , I would like to say a massive thank you to my advisor Dr. Shreejtih Shanker for being available for discussions during the entire 15 months of the project duration.

Contact: khandels@tcd.ie