Neural Network-Based Equalizer Hardware-Accelerator

PRESENTED TO

DR.EMAD BADRY

Why NN Equalizer

Digital communication systems suffer from channel impairments such as:

- Inter-symbol interference (ISI)
- Nonlinear distortions
- Noise and fading

Traditional equalizers (LMS / MMSE) struggle with:
➢ Nonlinear channel distortion
➢ Time-varying multipath fading
➢ High-speed real-time processing

Why NN Equalizer

Recent NN-based equalizers have been shown to **outperform conventional methods** (higher Q-factor/BER) while using comparable numbers of real multipliers.

For example, a biLSTM+CNN equalizer matched DBP performance with +1.7 dB Q-factor gain over a CDC baseline in experiments

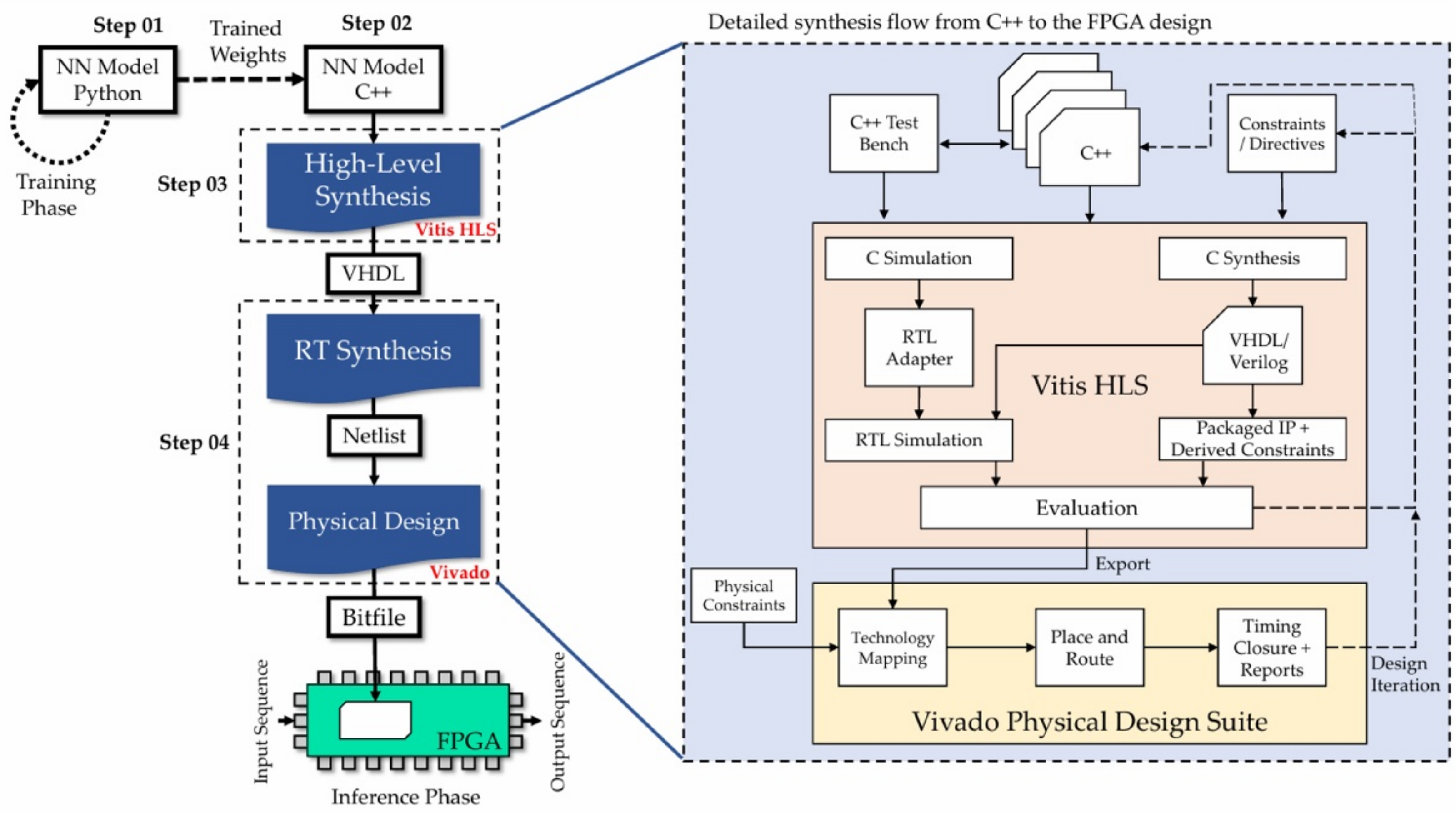
Why IC Hardware Instead of DSP?

DSP Equalizer	Neural Equalizer IC
Programmable but slower	Ultra low-latency inference
Higher power consumption	Very low dynamic power
General-purpose design	Optimized for equalization only
Better for software flexibility	Better for IoT & high-speed hardware

Objectives

- **Implement** offline FPGA realization of NN equalizer for nonlinear impairment compensation.
- **Pipeline development:** Convert trained Python models to fixed-point C++ and then to FPGA IP via high-level synthesis (HLS).
- **Benchmarking:** Compare NN equalizer performance (pre-FEC BER/Q-factor) against 1-StpS DBP and standard chromatic dispersion compensation CDC (using simulated and experimental data).
- **Hardware analysis:** Evaluate activation-function approximation methods, FPGA resource utilization, and achievable latency/throughput for target bit-rates.

Workflow

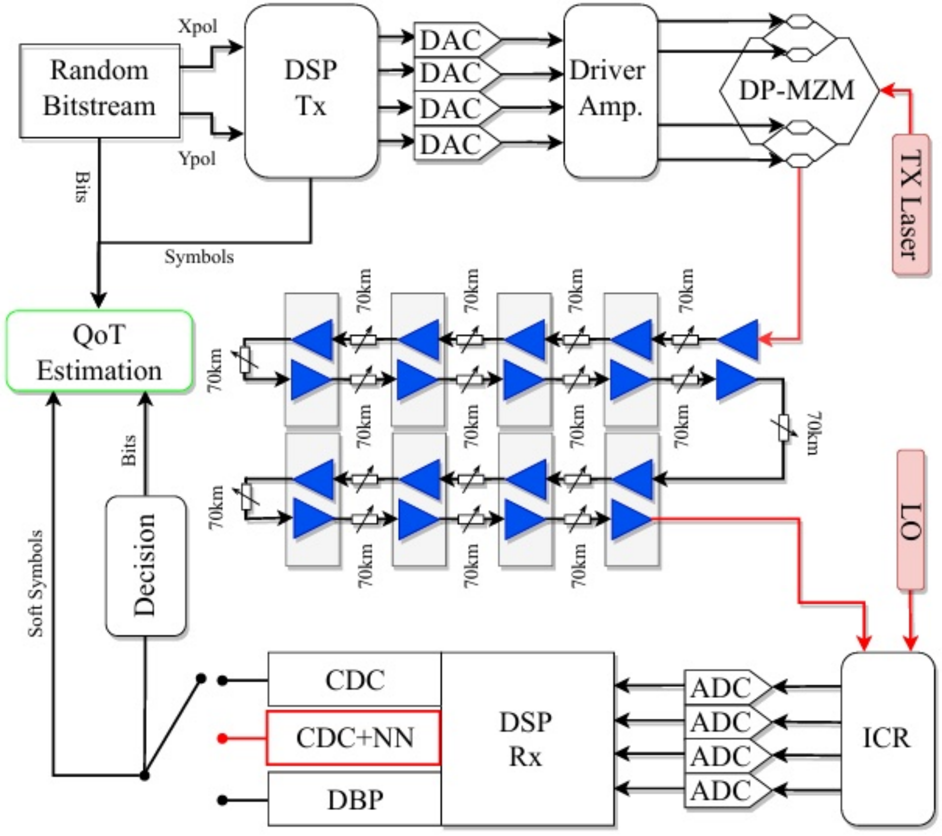


Step 01
NN Model Python
Trained Weights
Step 02
NN Model C++
Training Phase
Step 03
High-Level Synthesis
Vitis HLS
VHDL
RT Synthesis
Step 04
Netlist
Physical Design
Vivado
Bitfile
Input Sequence
FPGA
Output Sequence
Inference Phase
Detailed synthesis flow from C++ to the FPGA design
C++ Test Bench
C++
Constraints / Directives
C Simulation
C Synthesis
RTL Adapter
VHDL/ Verilog
Vitis HLS
RTL Simulation
Packaged IP + Derived Constraints
Evaluation
Export
Physical Constraints
Technology Mapping
Place and Route
Timing Closure + Reports
Design Iteration
Vivado Physical Design Suite

Experimental Setup

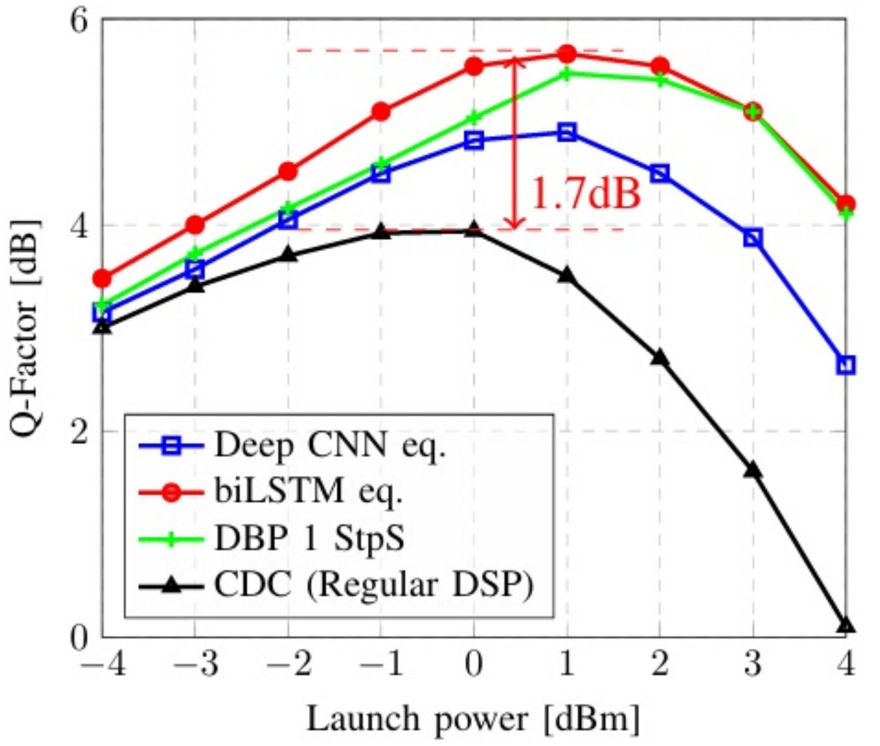
- **Transmission link:** Single-channel dual-polarization 16QAM at 34 GBd over 17×70 km spans of large-effective-area fiber (LEAF).
- **Receiver DSP:** Coherent optical RX (integrated coherent front-end).
- **NN Equalizer:** Processes the *soft* output of the DSP (before symbol decisions) to mitigate residual fiber nonlinearity.
- **three possible equalization paths (for comparison)**:

1. **CDC** (Chromatic Dispersion Compensation)
2. **CDC + NN** (standard dispersion compensation + neural-network nonlinear equalizer)
3. **DBP** (Digital Backpropagation)



Experimental Setup

- biLSTM outperforms DBP, especially in noise-dominated regions.
- Deep CNN Equalizer Performs Well but Worse Than biLSTM & DBP.
- In real experiments, NNs also learn and compensate:
  1. Hardware distortions (ADC/DAC, drivers)
  2. Polarization mismatch
  3. Connector loss
  4. Fiber parameter variations



Neural Network Design

- **Architectures:**

a) *biLSTM+CNN equalizer:* one bi-directional LSTM layer (35 hidden units) followed by a convolutional layer (total 70 filters).

b) *Deep CNN equalizer:* two convolutional layers (each 35 filters, 1×11 kernel, zero-padding) followed by an output conv layer.

- **Network details:** Hidden layers use hyperbolic-tangent (tanh) activation; the final output layer uses a linear (identity) activation.
- **Training:** Both models trained with MSE loss and Adam optimizer. Hyperparameters: mini-batch size 200, learning rate 5×10^–4. Training set of 2^20 symbols was used, with early stopping on validation BER. Models were trained for 30,000 epochs on a fixed dataset (sampling 2^18 new inputs each epoch).
- **Results (software):** After training, both NNs significantly reduced BER. The biLSTM+CNN model consistently outperformed the pure CNN equalizer in simulations.

Possible Adjustments

- Use a different channel type (e.g. wireless channel).
- Optimize the design to suit specific applications, like IOT.
- Synthesize for ASIC target library

THANK YOU