Challenges using FPGA Clusters for Distributed CNN Training

Philipp Kreowsky*†, Justin Knapheide*†, Benno Stabernack*†

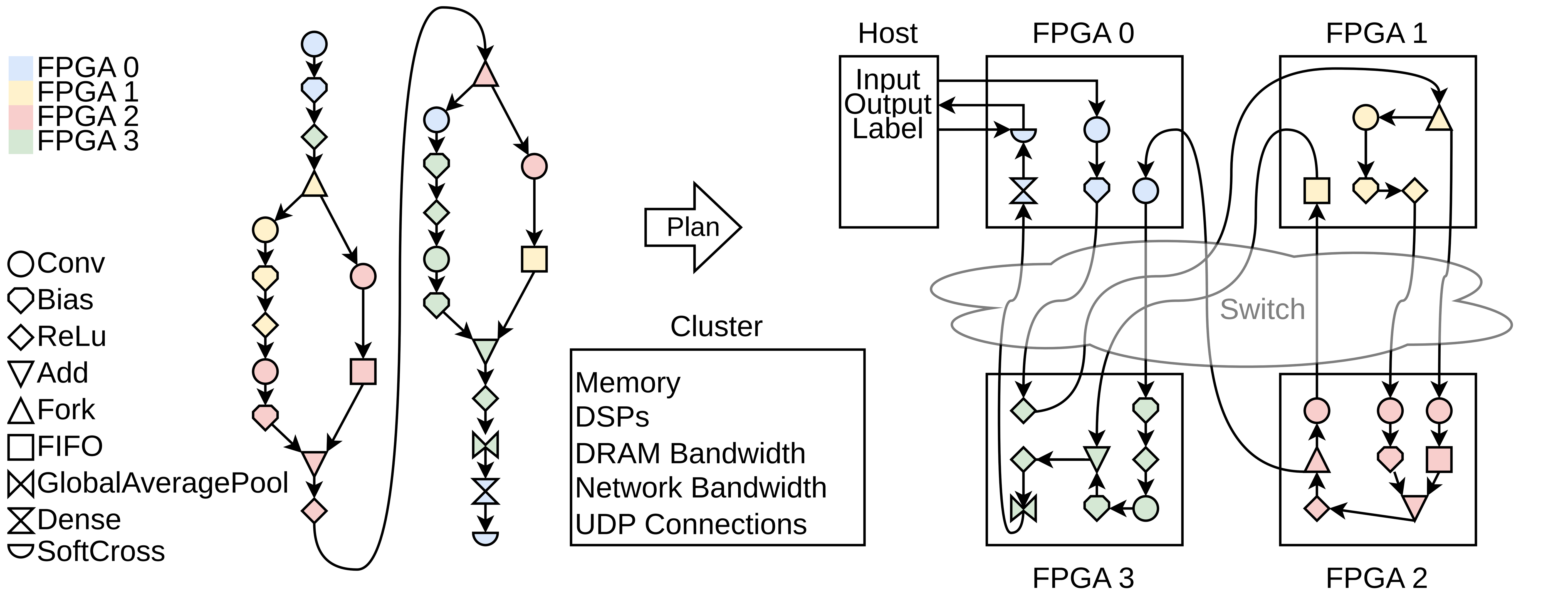
*Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Berlin, Germany
{philipp.kreowsky, justin.knapheide, benno.stabernack}@hhi.fraunhofer.de
†University of Potsdam, Embedded Systems Architectures for Signal Processing, Potsdam, Germany

Abstract

An easy-to-use framework for CNN training on network-attached FPGA clusters:

- **Training** of deep neural networks on FPGA clusters
- Arbitrary **DAG** structure
- Focus on **layer parallelism**
- Dedicated implementation for each layer
- Implemented in **SpinalHDL**
- Resource estimates derived from hardware description
- **Automatic** placement strategy across FPGAs in the cluster

Mapping a Network of Layers onto an FPGA Cluster



Problem

We need to **find mappings of L layers to N FPGAs**, represented as $d_{l,n,c} \in \{0,1\}$ with $0 \le l < L$, $0 \le n < N$, $c \in C_l$, where $d_{l,n,c} = 1$ means layer l is placed on FPGA n with configuration c. C_l is the set of all possible configurations for layer l.
Each layer must be placed exactly once with one configuration.

$$\tilde{d}_{l,n} = \sum_{c\in C_l} d_{l,n,c}, \quad \sum_{n=0}^{N-1} \tilde{d}_{l,n} = 1 \quad \forall l < L, c \in C_l \tag{1}$$

These mappings need to **respect the available resources** $R^{\text{dev}}_{n,t} \in \mathbb{R}$ for each device n and resource type $t \in T$. For now, we consider the amount of **on-chip memory**, number of **DSP cores** and **DRAM bandwidth**. For each layer- and device type, we require estimates of the implementation's achievable throughput $S_{l,n}(c) \in \mathbb{R}$ and resource consumption $R^{\text{layer}}_{l,n,t}(c) \in \mathbb{R}$ depending on layer configuration c.

$$\sum_{l=0}^{L-1}\sum_{c\in C_l} d_{l,n,c} R^{\text{layer}}_{l,n,t}(c) \le R^{\text{dev}}_{n,t} \quad \forall n < N, t \in T \tag{2}$$

Additionally, we need to **consider the available network bandwidth** $C^{\text{dev}}_n \in \mathbb{R}$. The required bandwidth $C^{\text{layer}}_{l,k}(t)$ between layers l and k can be calculated as $\hat{S}F_{l,k}$, where $\hat{S}$ is the overall throughput and F is the sum over all connections between l and k of the connection's feature map size.

$$\sum_{l=0}^{L-1}\sum_{k=0}^{L-1} \tilde{d}_{l,n}\left(1-\tilde{d}_{k,n}\right) C^{\text{layer}}_{l,k}(t) \le C^{\text{dev}}_n \quad \forall n < N \tag{3}$$

Each network hop introduces some amount of latency, which can lead to stalls when two branches that have taken different paths across the cluster join. In order to avoid having to calculate and compensate for these delays, we **require joining branches to have taken an equal number of network hops** $H_l \in \mathbb{N}$ from the input.

$$H_l = H_k + \sum_{n=0}^{N-1} \tilde{d}_{l,n}\left(1-\tilde{d}_{k,n}\right) \quad \forall l,k < L, F_{l,k} > 0 \tag{4}$$

Finally, we **force input- and output layers onto FPGA** 0.

$$\tilde{d}_{l,0} = 1 \quad \forall l \in P, \tag{5}$$

where $P \subset \{0..L-1\}$ is the set of input- and output layers.
Given these constraints, we want to maximize the overall throughput $\hat{S}$, which is equal to that of the slowest layer.

$$\text{maximize } \hat{S} \text{ subject to Equations (1) to (5),}$$
$$\hat{S} \le \sum_{n=0}^{N-1}\sum_{c\in C_l} d_{l,n,c} S_{l,n}(c) \quad \forall l < L \tag{6}$$

Strategy

Until now, we use binary search until the throughput is known to be within 5 % of the achievable throughput.

Simplified approach

- Select target throughput
- The optimization problem turns into a **Constraint Satisfaction Problem**
- Solvability check allows **binary search for max throughput**
- Pareto-optimal configurations considered for target throughput

Current further simplification

- Select one config per layer and device that meets target throughput
- Use achievable throughput as cost function
- For convolutions, we use DSPs core usage
- For smaller clusters and models, solved using CP-SAT Solver (Google OR-Tools)
- Runtime issues with larger models and clusters
- Future considerations: Explore greedy, annealing, or deep learning methods

Planning Times

Model	Cluster	Plan	Pinned Layers	# Layers	FPS	Utilized DSPs
VGG mini	2 × 10AX115	7 min	No	34	600	38.5 %
Demo	4 × 10AX115	3 min	No	127	558	72.2 %
MobileNetV2	4 × AGFB014	21 min	Yes	225	500	37.3 %
		94 min	No	225	516	37.8 %
	6 × 10AX115	13 min	Yes	240	696	81.2 %
		169 min	No	240	727	82.2 %
ResNet18	All Devices	51 min	Yes	332	727	78.8 %
		69 min	No	332	800	88.1 %

All Devices: 1 × Agilex (AGFB027) + 4 × Agilex(AGFB014) + 6 × Arria(10AX115)

A typical CNN Layer Combination for Training