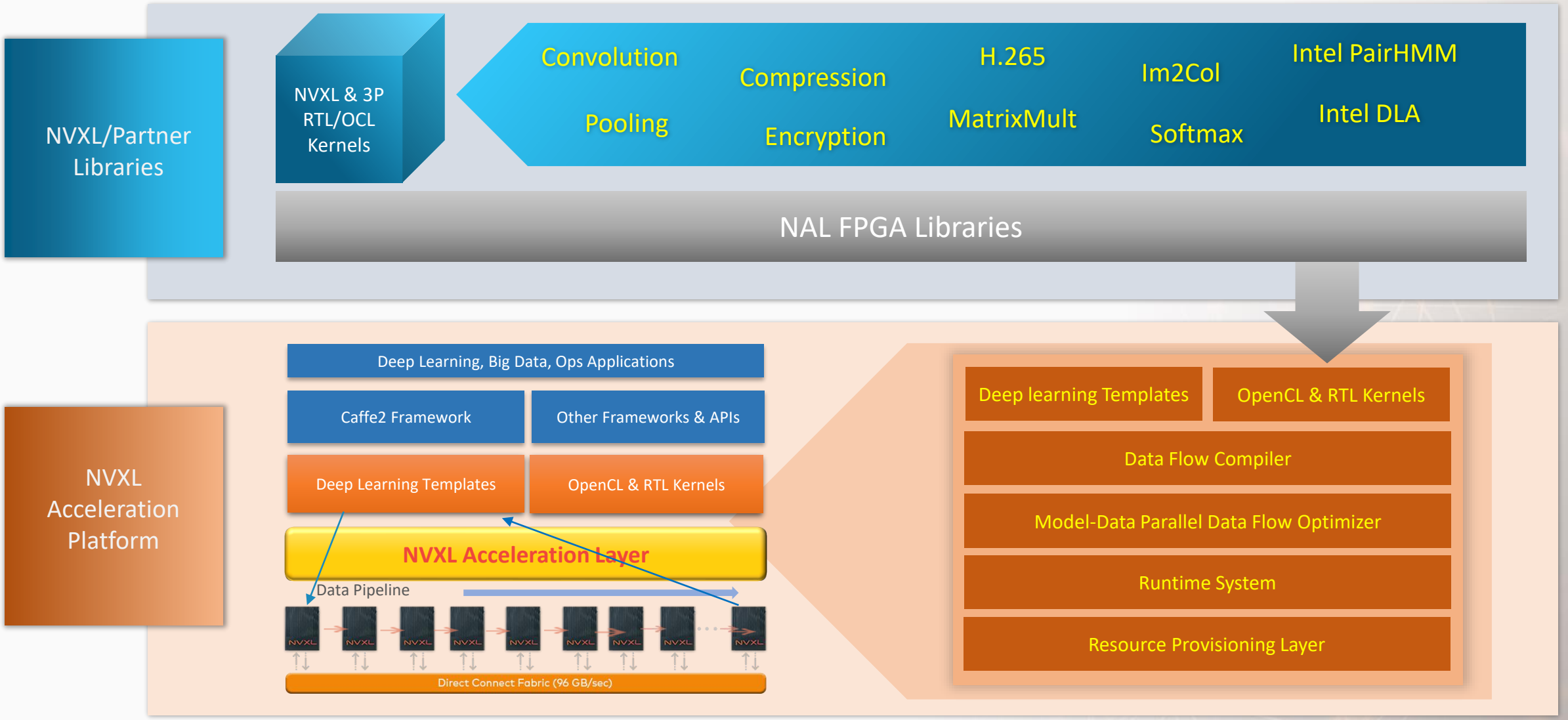
Design System for Machine Learning Accelerator

Joonyoung Kim
NVXL Technology
Senior Director of Machine Learning HW Development

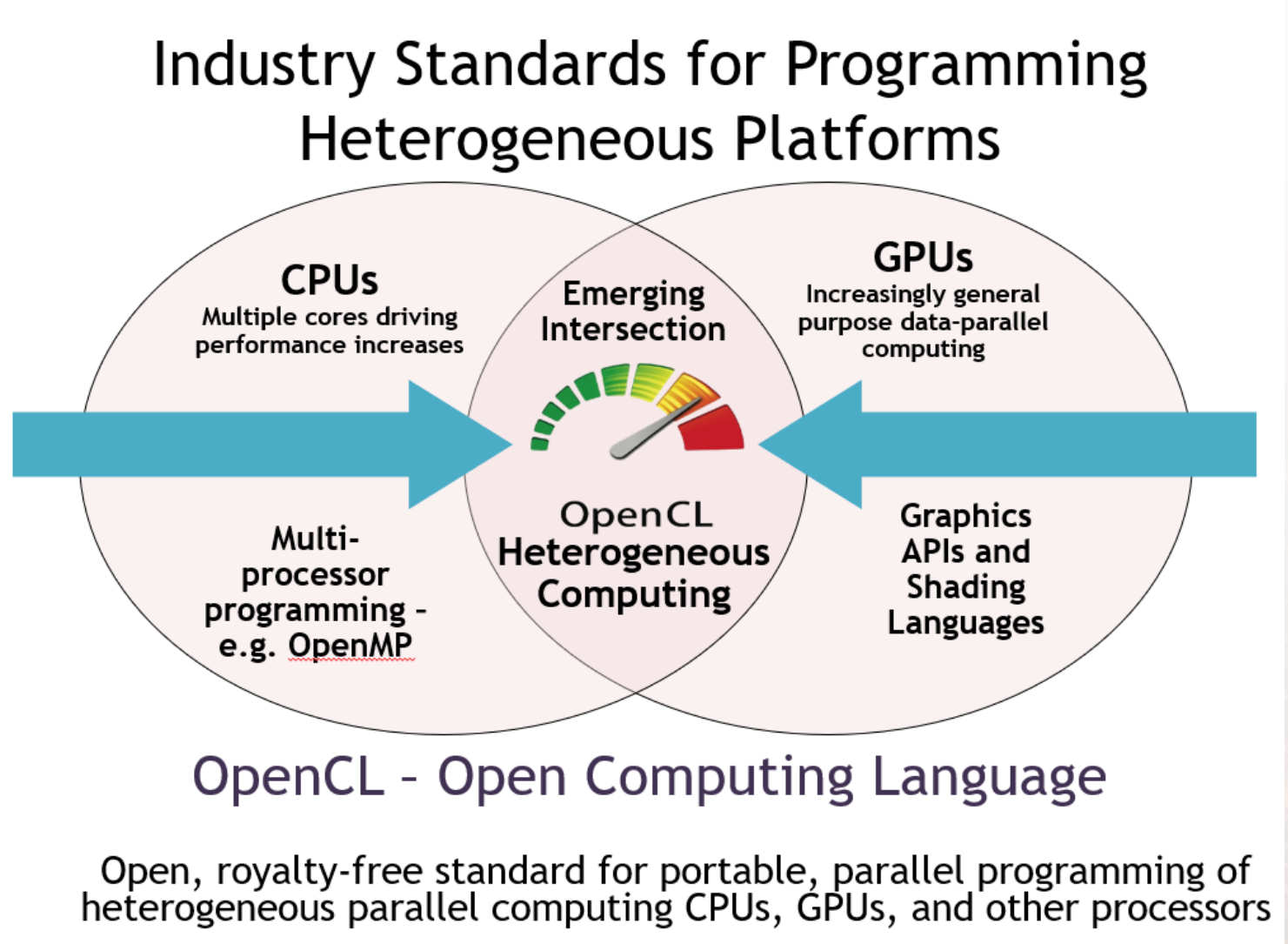
09/13/2018

NVXL
never stop accelerating.

NVXL ACCELERATION PLATFORM



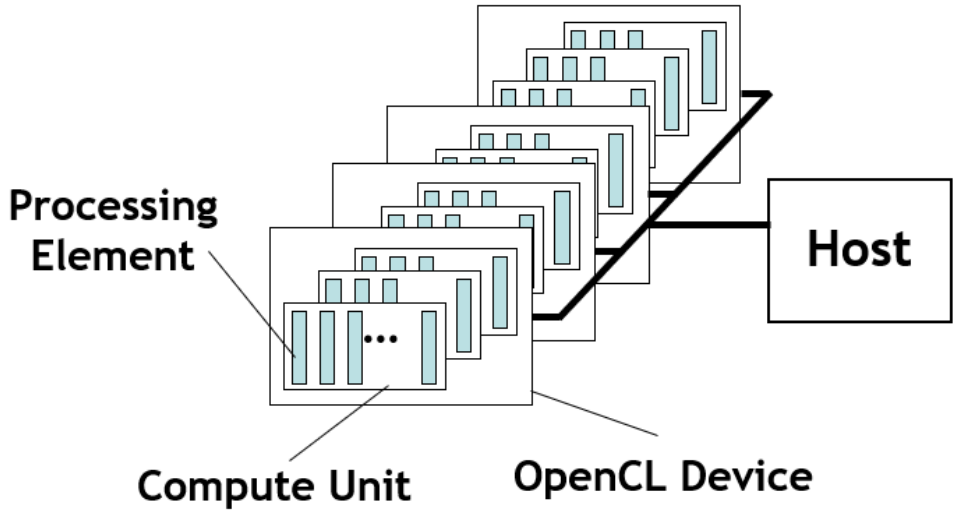
OpenCL Overview



Source: Khronos Initiative for Training and Education

OpenCL Overview

OpenCL Platform Model



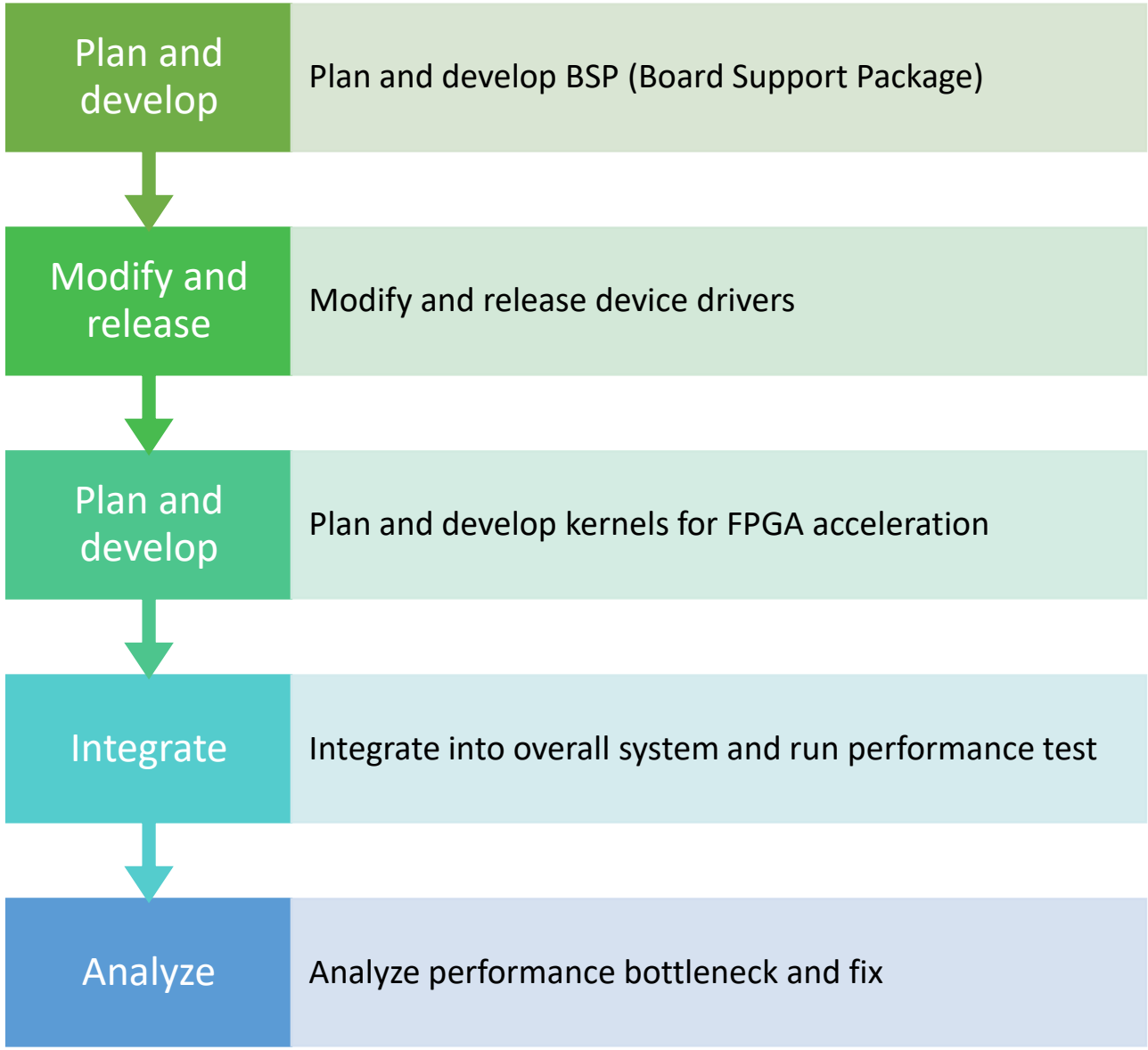
- One ***Host*** and one or more ***OpenCL Devices***
  - Each OpenCL Device is composed of one or more ***Compute Units***
    - Each Compute Unit is divided into one or more ***Processing Elements***
- Memory divided into ***host memory*** and ***device memory***

Source: Khronos Initiative for Training and Education

Why OpenCL and FPGA

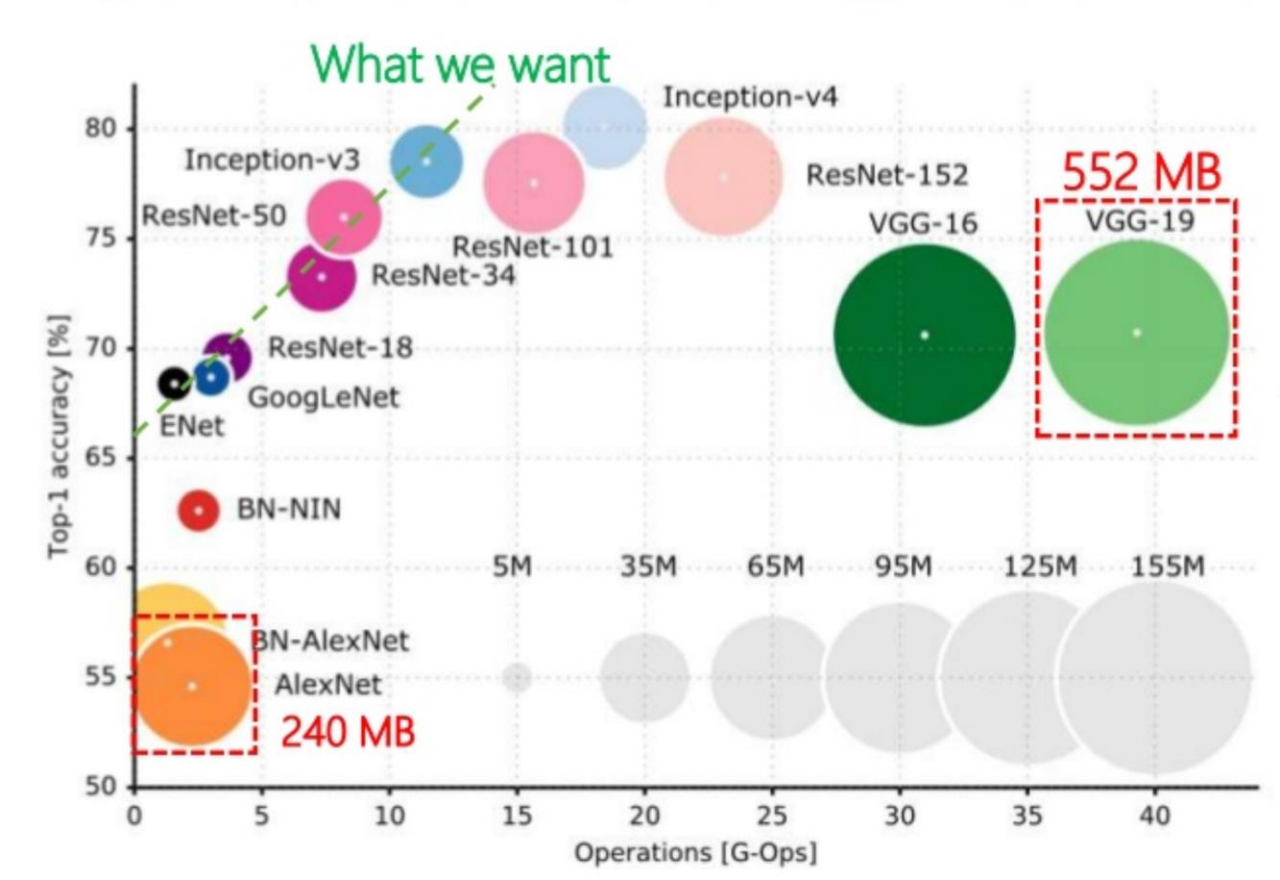
- Faster time-to-market (OpenCL vs RTL)
- Quick design exploration
- Easy design re-use
- Faster design completion
- Increased performance by offloading performance-intensive functions from the host processor to the FPGA
- Significantly lower power than a GPU or multicore CPU by using OpenCL, which generates only the logic needed

FPGA Accelerator Design Process



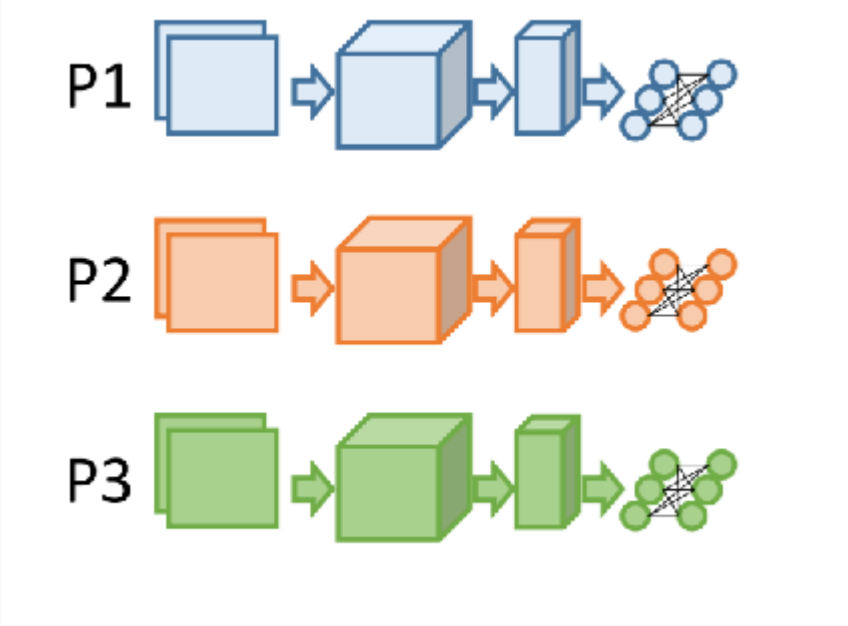
Machine Learning Accelerator on FPGA

- Design consideration
  - High throughput for popular CNNs
  - Low latency
  - DRAM bandwidth
  - DSP and BRAM utilization
  - Number of MACs
  - Amount of parameters
  - Accuracy



Data Parallelism

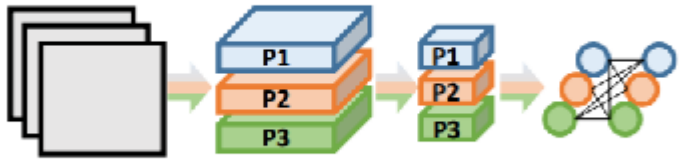
- Same model mapped to multiple FPGAs
- Data is split by orchestration SW into different mini batches and processed in parallel
- Overall performance scales linearly with number of modules



Source: https://arxiv.org/pdf/1802.09941.pdf

Model Parallelism

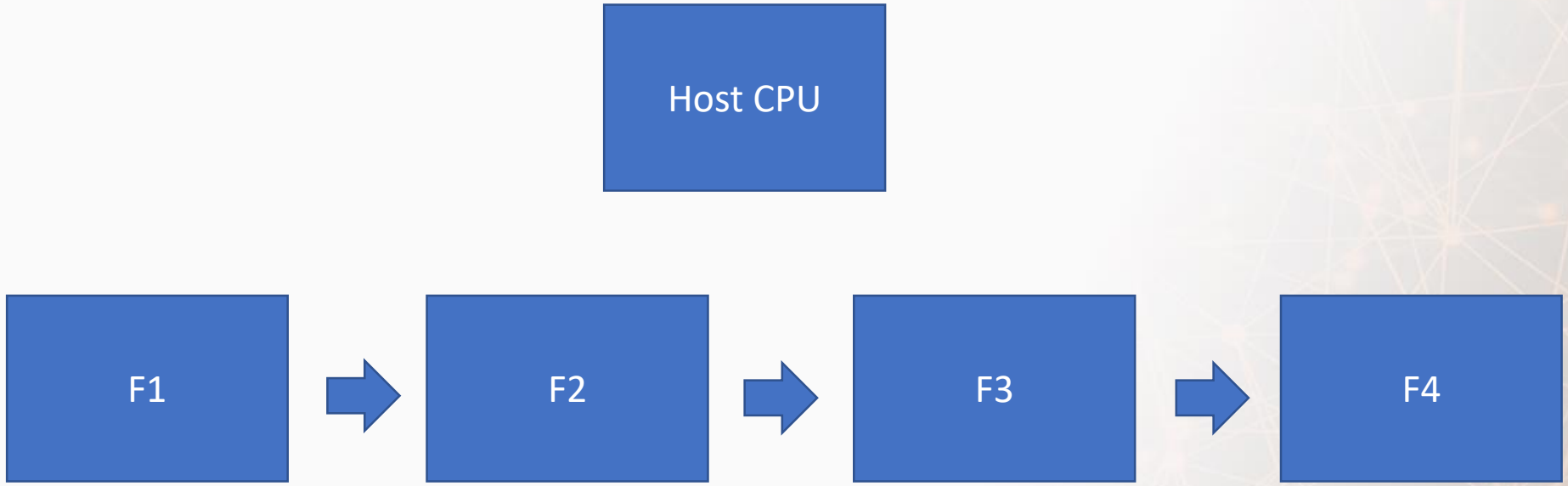
- Take a complex stage (layer) and split into multiple modules
  - E.g. 128x128 matrix is split into sixteen 32x32 matrices and processed in parallel
  - Initially done to balance out overall run time
- Requires explicit synchronization



Source: https://arxiv.org/pdf/1802.09941.pdf

Pipeline Parallelism (Layer Pipelining)

- Multiple FPGAs are used to implement certain network that is too large to fit in one FPGA



BSP Development

- Start with a vendor reference design
- Identify feature set
- Implement required features and verify
- Board test
- Deployment

Kernel Development

Identify vertical areas

Computation modeling in C/C++
(often open source code)

Profiling and identify offloading opportunity

Design in OpenCL

Performance tuning

Deployment

Matrix Factorization

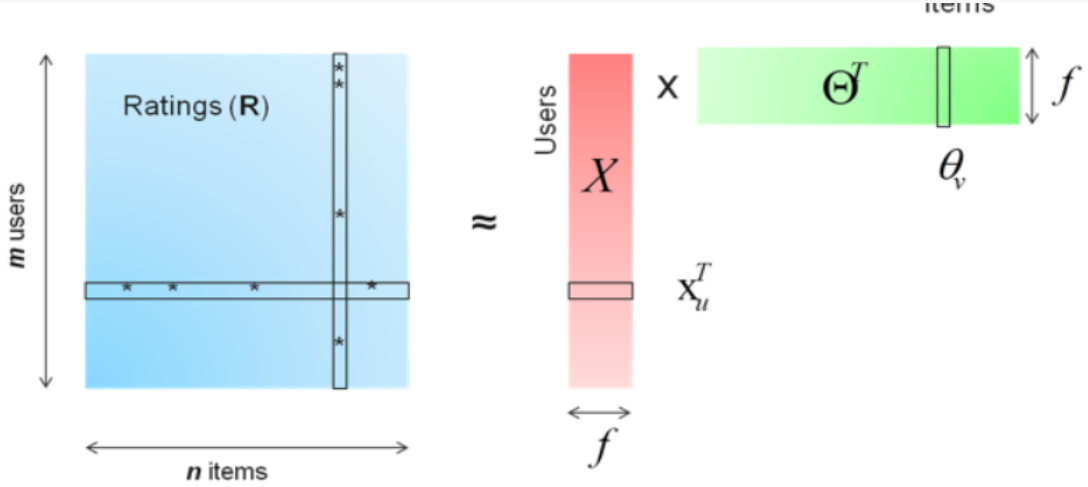


Figure 1. Matrix factorization factors a sparse ratings matrix R (m-by-n, with N_z non-zero ratings) into a m-by-f matrix (R) and a f-by-n matrix (Θ^T).

Suppose we obtained m users' ratings on items (say, movies). If user u rated item v, we use r_{uv} as the non-zero element of R at position (u, v). We want to minimize the following cost function J. To avoid overfitting, we use weighted-λ-regularization proposed in [1], where n_{x_u} and $r_u v$ denote the number of total ratings on user u and item v, respectively.

$$J = \sum_{u,v}(r_{uv} - x_u^T\theta_v)^2 + \lambda(\sum_u n_{x_u}\|x_u\|^2 + \sum_v n_{theta_v}\|\theta_v\|^2)$$

Many optimization methods, including Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD) [1] have been applied to minimize J. We adopt the ALS approach, which first optimizes X while fixing θ, and then optimizes $theta$ while fixing X. That is, in one iteration we need to solve these two equations alternatively:

$$\sum_{r_{uv}\neq 0}(\theta_v\theta_v^T + \lambda I)\cdot x_u = \theta^T \cdot R_{u*}^T \ (1)$$

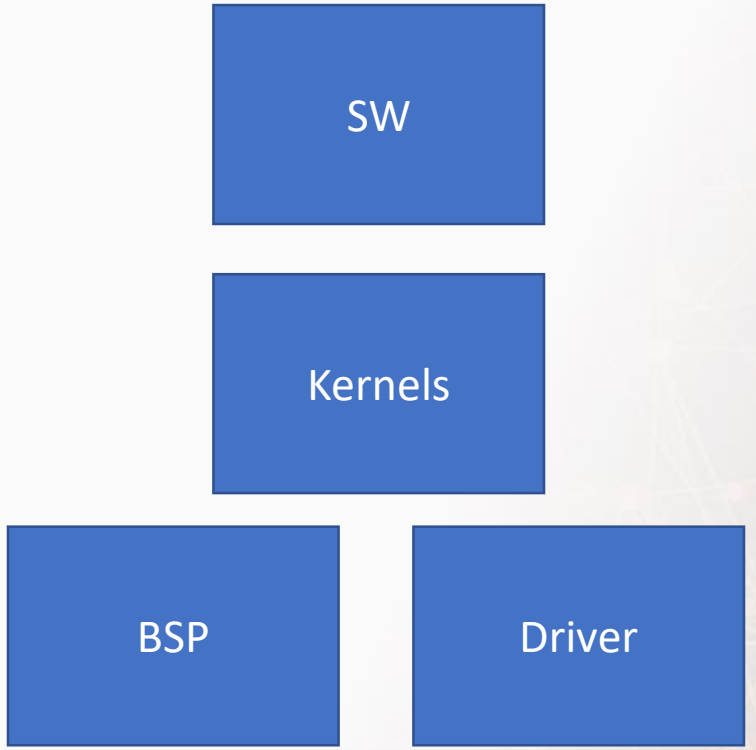
$$\sum_{r_{uv}\neq 0}(x_u x_u^T + \lambda I)\cdot \theta_v = X^T \cdot R_{*v} \ (2)$$

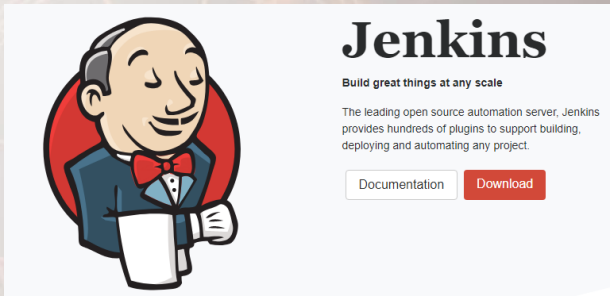
Kernels for Matrix Factorization

- **Matrix Multiplication:** For Matrix-Matrix and Matrix-Vector Multiplication)
- **Conjugate Gradient Solver**: To solve system of linear equations). This solver will solve the system of equations in maximum k iterations. k is the number of features.

Continuous Integration and Deployment (CICD)

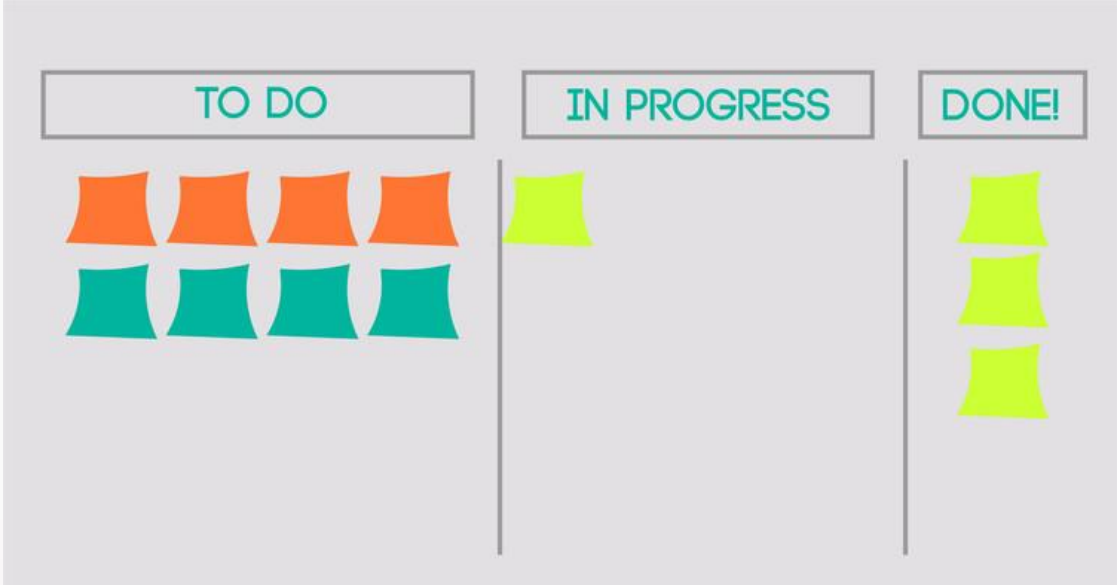
- Changes in different system component can trigger various actions including regression and release
- Need a dedicated CI and release server because of board test component
- Similar to DevOps model in SW





Agile Development Process

- Fully embracing agile development methodology
  - Fully open and transparent issue tracking
  - Frequent stand-ups
- Monthly sprint cadence
  - Planning
  - Execution
  - Retrospective




NVXL
never stop accelerating.