A Flexible Design Automation Tool for Accelerating Quantized Spectral CNNs

Rachit Rajat, Hanqing Zeng, Viktor Prasanna

University of Southern California

fpga.usc.edu

FPL 2019, Barcelona

USC Viterbi School of Engineering

Outline

- Introduction
- Background
- Tool overview
- Architecture template
- Optimizations
- Experiments
- Conclusion

Introduction

- **Challenges** in CNN inferencing on FPGAs:
  - Computation complexity: sliding window operations
  - Design effort: design space search & manual hardware implementation
  - Design optimization: resource utilization & clock rate for large scale designs
  - Design flexibility: various CNN models and FPGAs and performance requirements
- **Need** fast generation of:
  - Performance meta-data to tune CNN models
  - Hardware code to deploy inference pipeline

Background & Motivation: Spectral CNN on FPGAs

- Convolutional Neural Networks (CNN)

- Spectral convolution [1]
  - Sliding window operation → Hadamard product
  - $I^{\text{output}} = \mathcal{F}^{-1}\left(\mathcal{F}\left(I^{\text{input}}\right) \circ K^{\text{spec}}\right)$
  - Partitioning on I and padding on K, Overlap-and-Add

- $\mathcal{F}$: Fourier transform
- $\mathcal{F}^{-1}$: Inverse Fourier transform
- I^{*}: image
- K^{spec}: conv. kernels after FFT

- Why spectral CNNs?
  - Computation reduction: $3\sim4\times$ for AlexNet, VGG16,….

[1]: Zeng, Chen, Zhang, Prasanna, A framework for generating high throughput CNN implementations on FPGAs, Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays

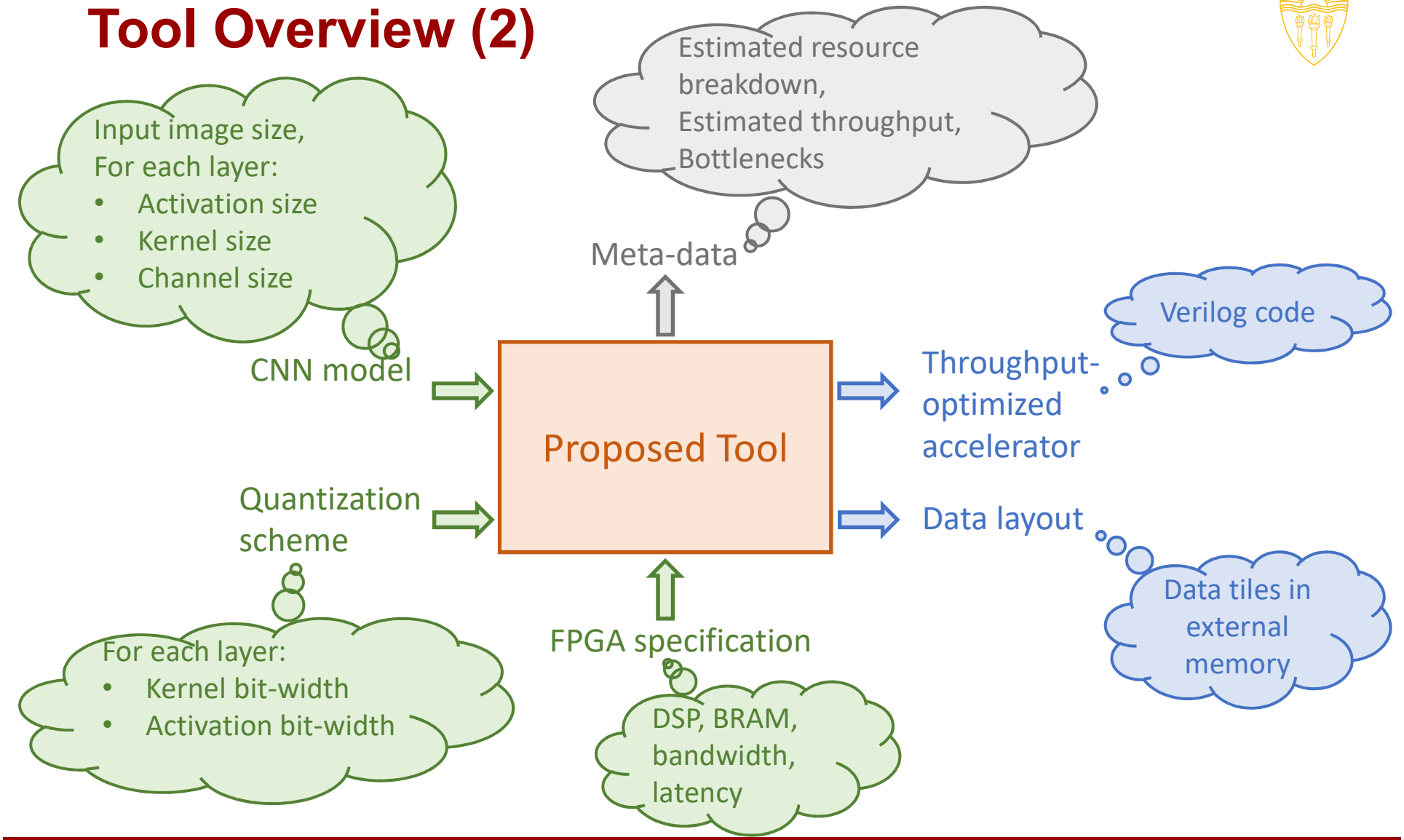
Problem is Non-trivial

- **Goal**: Fast and flexible design space exploration and generation of Verilog for *high throughput* inference
- **Constraints**: Limited BRAM and DSP resources
- Need to explore a huge design space quickly
- Optimization needed in spectral convolution engine to support large FPGA devices

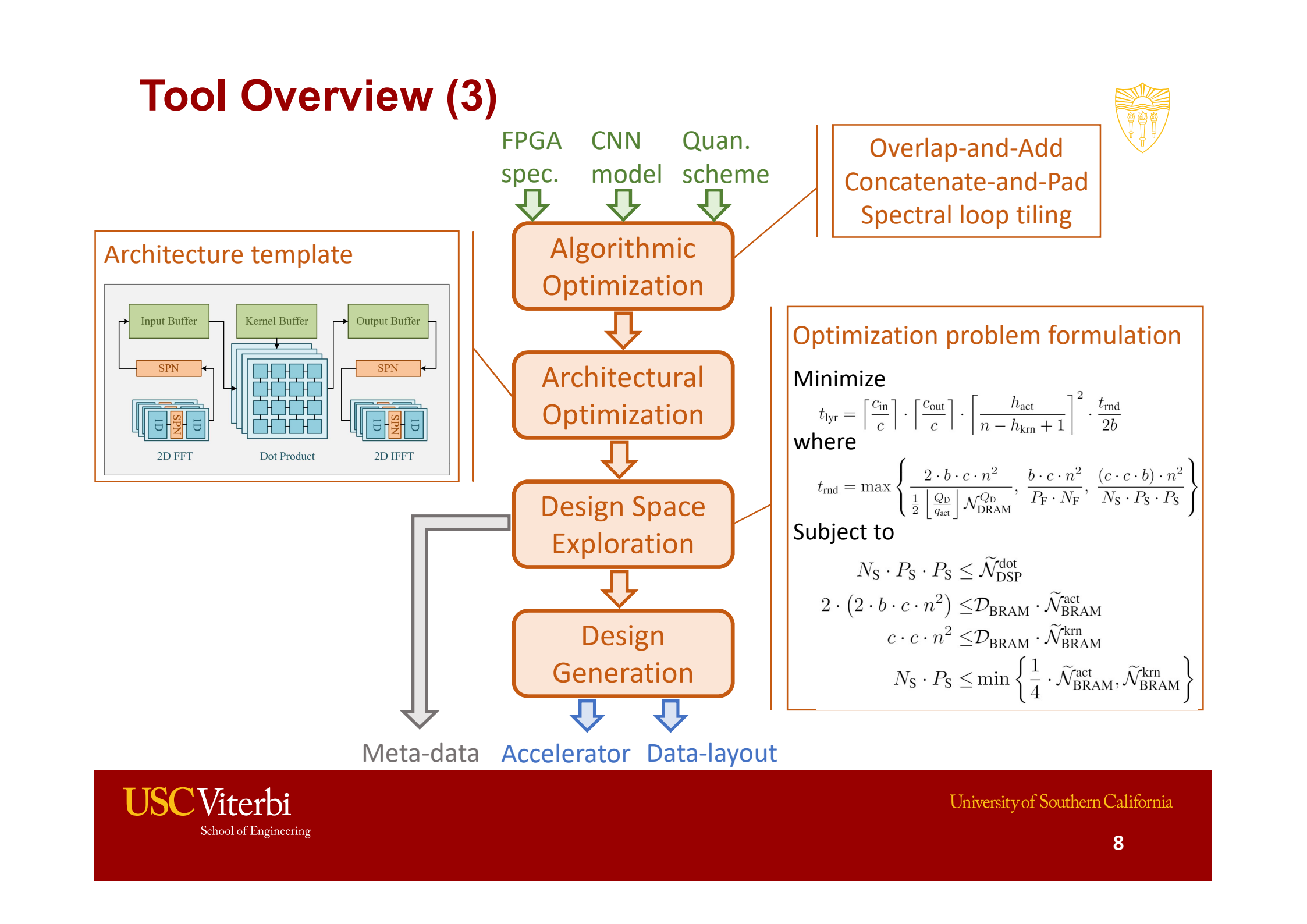
Tool Overview (1)

- Automated tool for generating quantized spectral CNN accelerators in synthesizable Verilog
- Performance metrics
  - Time to generate design
  - Throughput of generated design
- Flexibility
  - Quantization schemes
    - Various bit widths for kernels and activations
  - FPGA architecture
    - Various resources (DSPs, BRAMs, bandwidth, etc.)
  - CNN models
    - Various model parameters (channels, kernel sizes, image sizes, etc.)

Tool Overview (2)

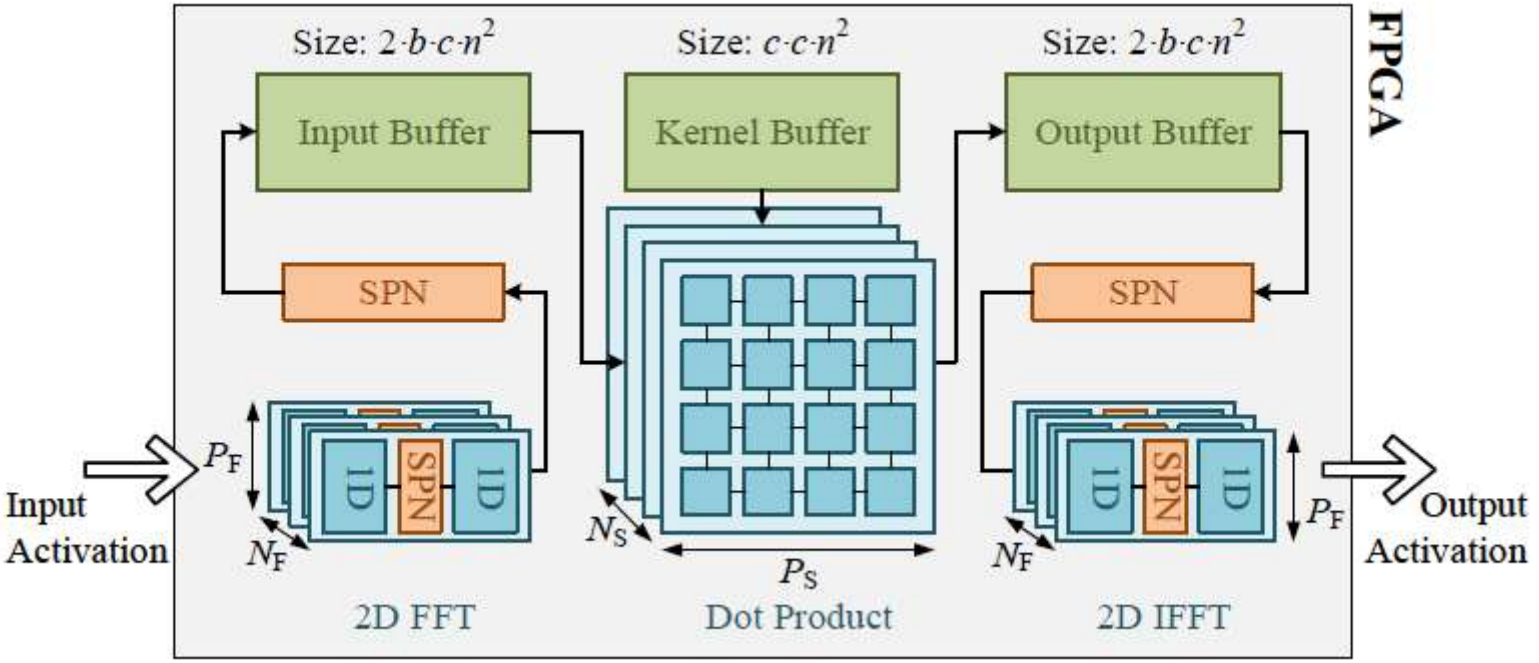


Tool Overview (3)



Architecture Template

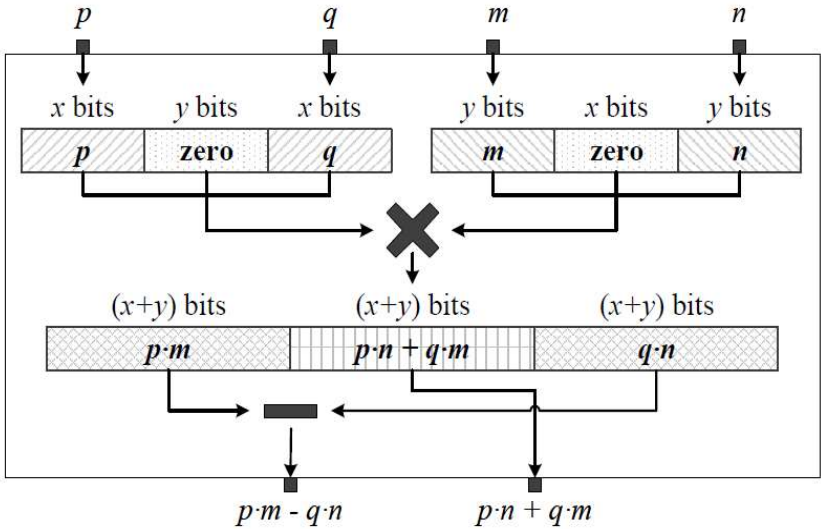
- **Design parameters**: FFT size, FFT parallelism, batch size, systolic array size, systolic array parallelism and number of channels
- **Architecture template** for Verilog generation:



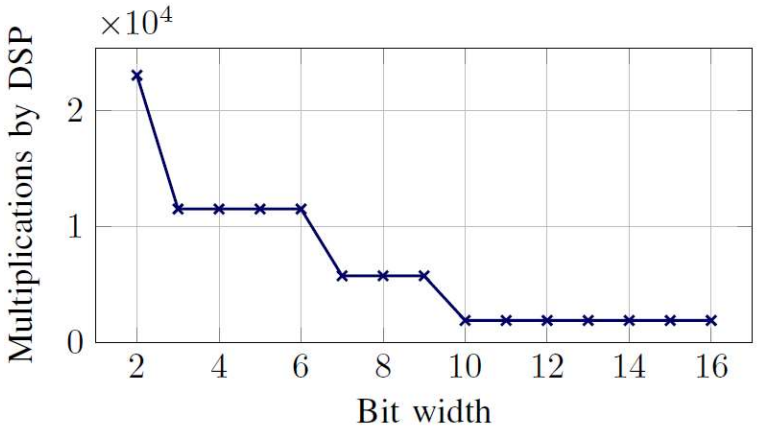
Optimization 1: Variable Bit-width Multiplier

- **Requirement** Unique to spectral CNN: low bit-width complex multiplication
- **Challenge**: DSPs accept fixed, high bit-width inputs
- **Idea**: Pad the data of low bit width to match the DSP input width

Example: $(p + jq) \cdot (m + jn)$



Performance estimation on Stratix 10

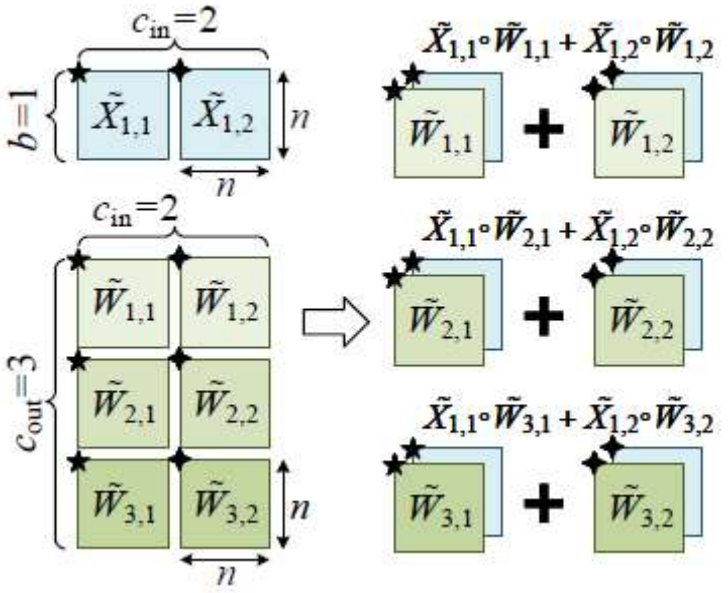


Number of complex multiplications *vs.* bit width

Optimization 2:
Switching Parallelization Dimensions (1)

- **Challenge**: Concurrent memory accesses for Hadamard product

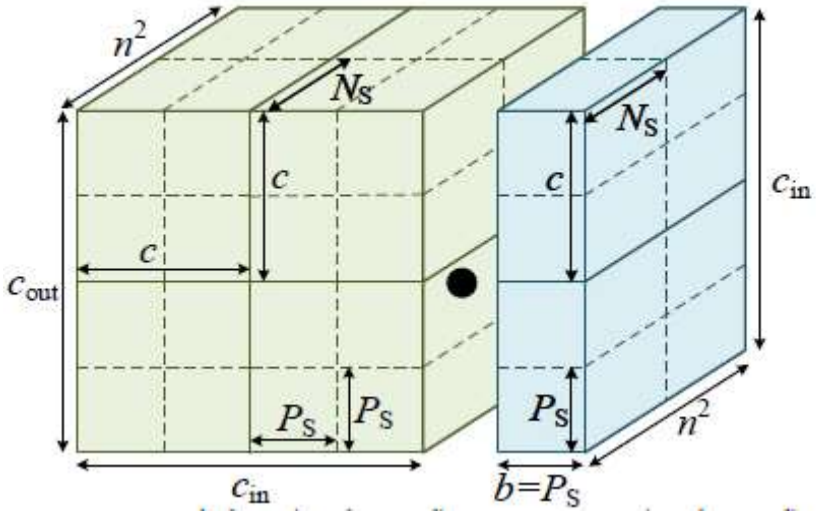
- Example:
  - $(1 * 2 * 3) \cdot n^2$ operations (n = FFT size)
  - $(1 * 2 + 2 * 3) \cdot n^2$ distinct BRAM accesses
  - Thousands of BRAM accesses per cycle to support parallelism of thousands of DSPs



- Severe **clock rate degradation** due to the pressure on BRAMs

Optimization 2:
Switching Parallelization Dimensions (2)

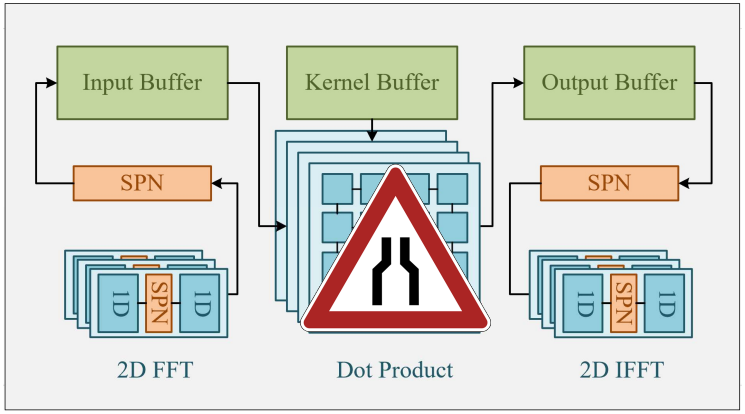
- Parallelize along ~~width & height~~ dimensions → ~~Hadamard~~ products
- Parallelize along batch & channel dimensions → Matrix dot products



- Systolic array: blocked matrix multiplication
- Analysis
  - $2N$ BRAM accesses/cycle for $2N^2$ DSP operations
- Efficient for FPGAs with large number of DSPs

Optimization 3: Design Space Exploration

- **Challenge:**
  - Large Design space:
    - 4 HW parameters: Parallelism of modules
    - 3 SW parameters: Data layout & tiling
- **Optimization goal**:
  - Inference throughput (batch processing)
  - ➢ Identify bottleneck stage in the pipeline



- **Optimization Problem/Constraints**: (see paper)
  1. SW-HW coordination — Tiling matches (device) parallelism
  2. Limited resources — Share DSP: FFT / Sys-array / IFFT
     Share BRAM: input / kernel / output buffers
     Share bandwidth: input / output activation
  3. Load-balance — Keep the pipeline always busy
- **Optimization Technique:** Hierarchical priority parameter sweep

Experimental Setup

- **Target FPGA devices** Stratix-10 GX, Stratix-V GX
- **Bit widths** 2- to 16-bit
- **CNNs** AlexNet, VGG16
- **Tool execution** Intel Core-i5 CPU

  Design space exploration + generation $<$ 2 sec

Comparison with State-of-the-art Designs (1)

- Comparison with state-of-the-art spectral CNN tool (FPGA '18)

	AlexNet		VGG16	
	FPGA '18 *	Proposed	FPGA '18 *	Proposed
FPGA	Stratix-10 GX2800	Stratix-10 GX2800	Stratix-10 GX2800	Stratix-10 GX2800
Clock (MHz)	120	200	120	200
Quantization	16-bit	16-bit	16-bit	16-bit
DSP	3264 (56%)	3264 (56%)	3264 (56%)	3264 (56%)
Logic	413K (45%)	140K (15%)	419K (47%)	140K (15%)
BRAM	6129 (52%)	1616 (22%)	6133 (32%)	2616 (22%)
Throughput (img/sec)	1704	**2841**	77	**129**

Switching parallelization dimensions improves **clock rate**

Optimized architectural template reduces **logic**

*: Original design on Strativ-V; Re-implemented on Stratix-10

Comparison with State-of-the-art Designs (3)

- Comparison with state-of-the-art spatial CNN tool (ICCAD '18)

16-bit	AlexNet		VGG16	
	ICCAD '18	Proposed	ICCAD '18	Proposed
FPGA	UltraScale KU115	Stratix-10 GX2800	UltraScale KU115	Stratix-10 GX2800
Clock (MHz)	220	200	235	200
Quantization	16-bit	16-bit	16-bit	16-bit
DSP	4854 (88%)	3264 (56%)	4318 (78%)	3264 (56%)
Logic	262K (40%)	140K (15%)	258K (39%)	140K (15%)
BRAM	986 (46%)	1616 (22%)	1578 (81%)	2616 (22%)
Throughput (img/sec)	1126	**2841**	65	**129**

Comparison with State-of-the-art Designs (3)

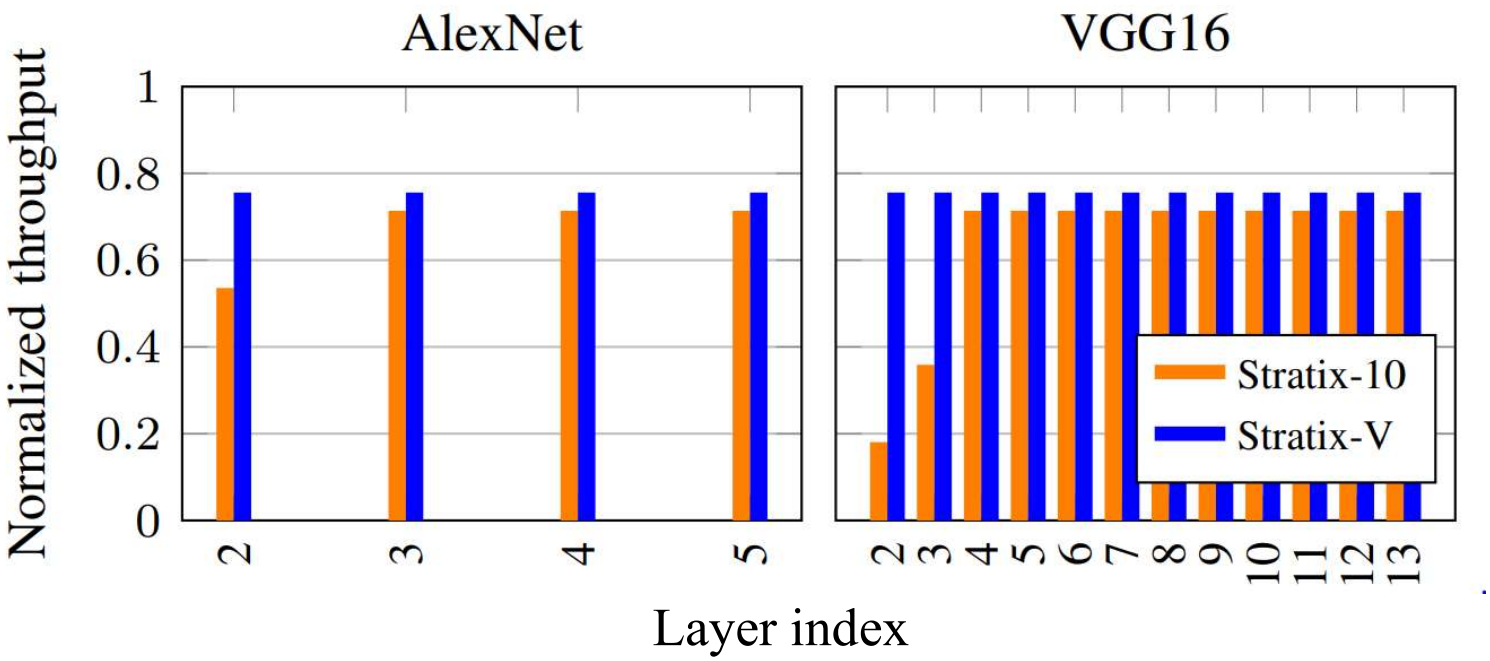
- Comparison with state-of-the-art spatial CNN tool (ICCAD '18)

8-bit	AlexNet		VGG16	
	ICCAD '18	Proposed	ICCAD '18	Proposed
FPGA	UltraScale KU115	Stratix-10 GX2800	UltraScale KU115	Stratix-10 GX2800
Clock (MHz)	220	200	235	200
Quantization	8-bit	8-bit	8-bit	8-bit
DSP				
Logic				
BRAM				
Throughput (img/sec)	2252	**9114**	130	**308**

Throughput improvement due to
- Spectral convolution algorithm
- Optimized design generation process

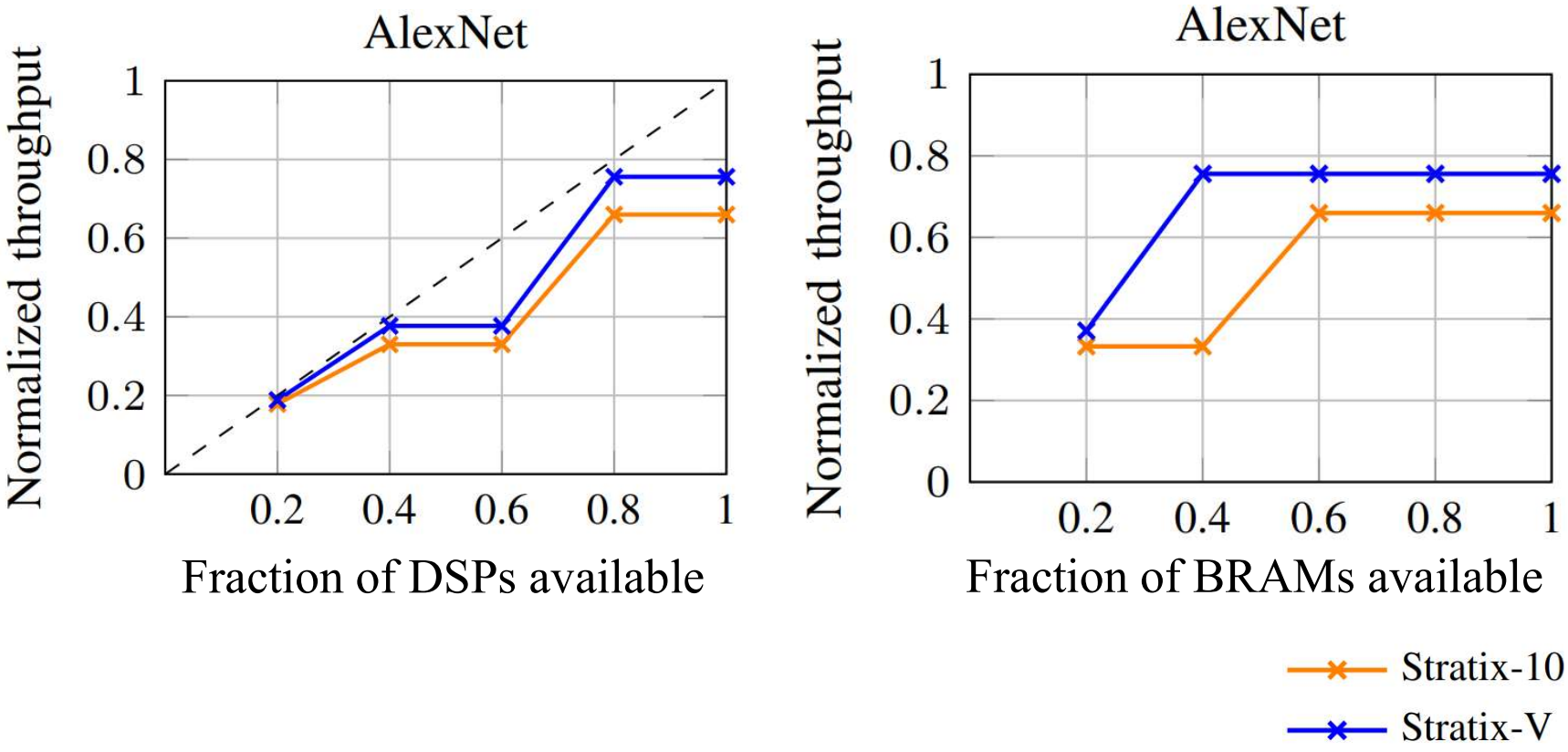
Evaluation on Flexibility (1)

- Flexibility w.r.t. CNN models

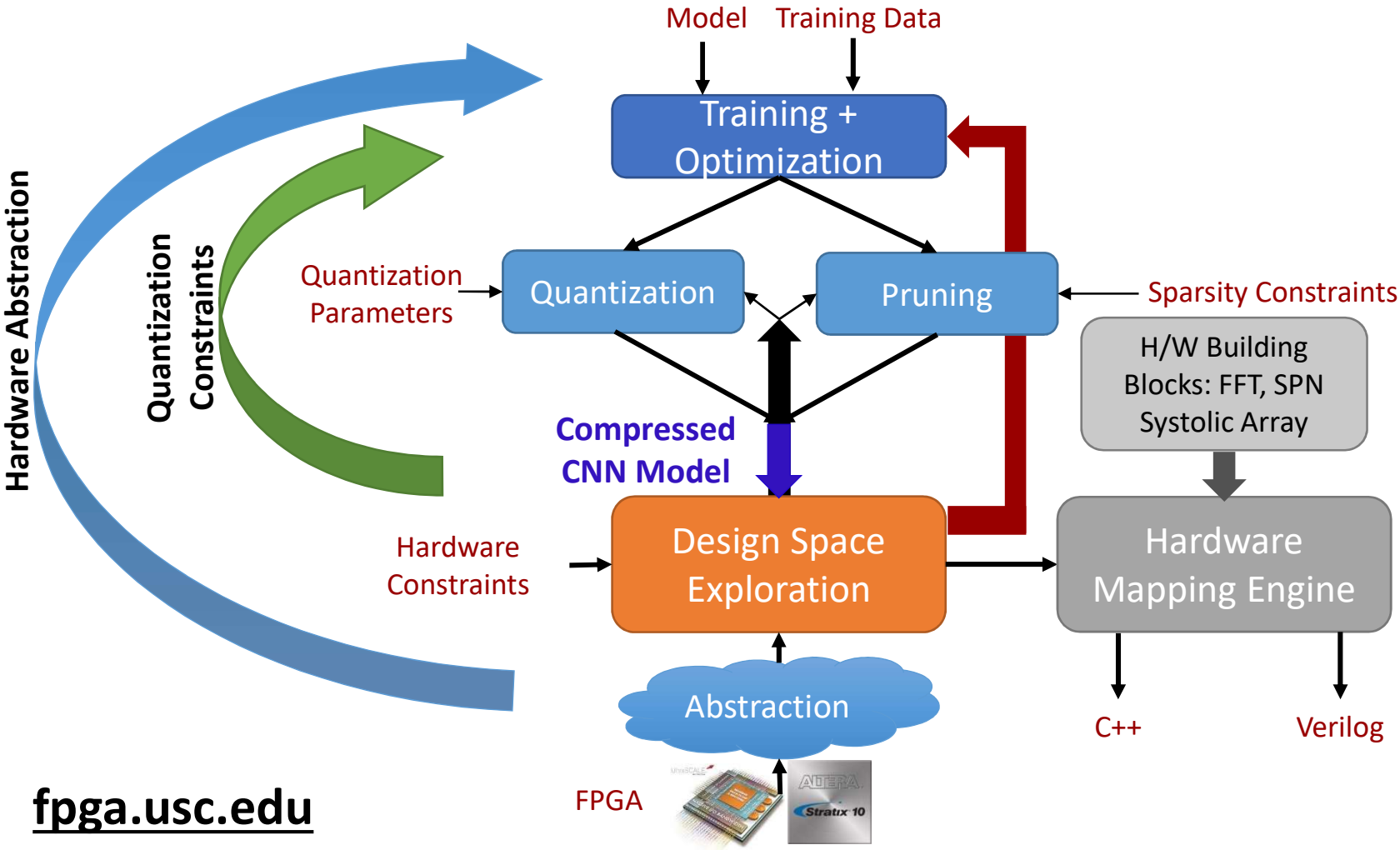


Evaluation on Flexibility (2)

- Flexibility w.r.t. FPGA resources



Flexible Tool for Automatic Generation of Pruned and Quantized Spectral CNNs: The Big Picture



fpga.usc.edu

Conclusion

- Design automation tool for generating high throughput spectral CNN accelerator
- Flexibility:
  - CNN models
  - Quantization schemes
  - FPGA devices
- Significantly higher throughput ($4\times$) than designed by state-of-the-art tools
- Spatial or Spectral??
- Implications: Multi-core, GPU platforms??

Thank you!

https://fpga.usc.edu/

prasanna@usc.edu