FTDL: An FPGA-tailored Architecture for Deep Learning Systems

Runbin Shi, Yuhao Ding, Xuechao Wei, Hang Liu, Hayden So and Caiwen Ding

The University of Hong Kong, Peking University, Stevens Institute of Technology, University of Connecticut

{rbshi, yhding, hso}@eee.hku.hk, xuechao@pku.edu.cn, hliu77@stevens.edu, caiwen.ding@uconn.edu

Motivation

Among the existing FPGA deep learning (DL) accelerators, most of them deployed application-specific integrated circuit (ASIC)-oriented architectures to FPGA without considering the FPGA underlying layout, which leads to the ***architecture-layout mismatch***. Such a mismatch results to a low f_{max} in implementation ($\approx$ 200MHz), while the computational unit (DSP) in FPGA can achieve an $f_{max} \approx$ 750MHz. The contributions of this work are summarized as,

- ***Good timing and scalability***: FTDL proposes a novel overlay architecture for convolutional and fully-connected layers that is tailored for the tiled structure of modern FPGAs, allowing post-place-and-route operating frequencies to reach over 88% of the theoretical DSP operating frequency across different devices and design scales.
- ***High hardware-efficiency***: FTDL provides a compilation tool that maps most DL-layers to the overlay with over 80% hardware-efficiency on average.

Hardware Overlay Design

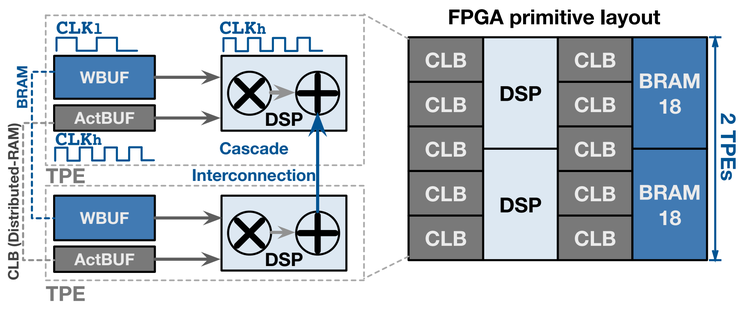

Figure 1: The tiled processing element (TPE) in FTDL, an FPGA-layout friendly design. The intrinsic DSP-interconnections accumulate the results of TPEs.

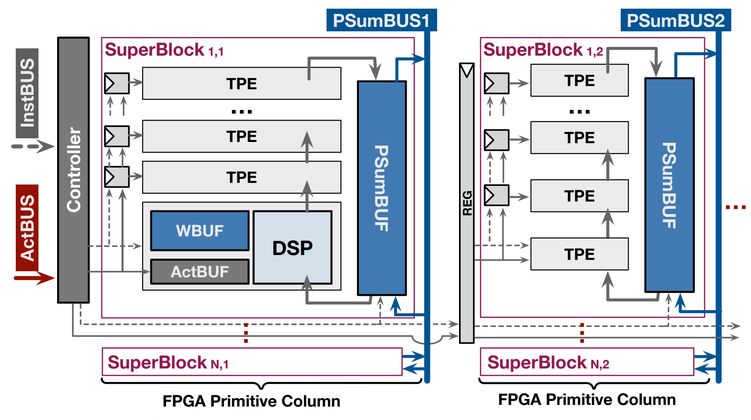

Figure 2: The SuperBlock organizes multiple TPEs in one column. Each SuperBlock contains a partial-sum buffer (PSumBUF) for accumulation.

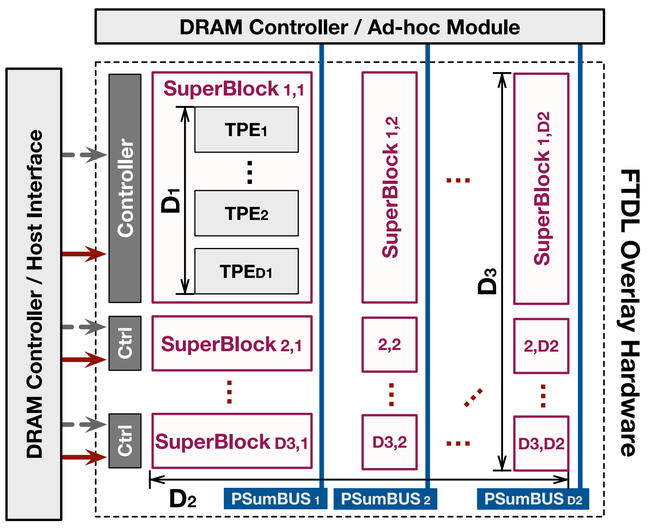

Figure 3: The general system diagram with parameterized FTDL hardware.

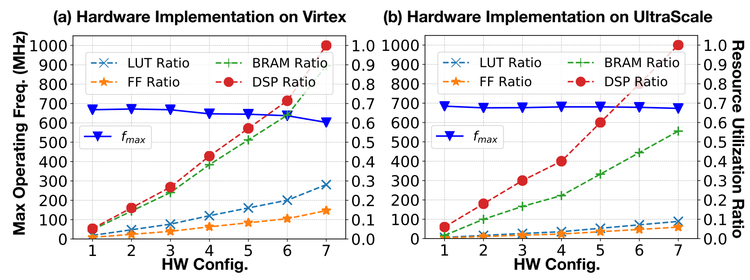

Figure 4: Hardware evaluation on Xilinx-Virtex device (a) and UltraScale device (b) after place and route. 7 hardware configurations are evaluated in a scale-up fashion. The FTDL hardware design shows a good timing and scalability that f_{max} stabilizes above 620 MHz on Virtex and 650 MHz on UltraScale.

Workload Scheduling

Objectives:

- Optimal performance
- Balance between performance and WBUF efficiency
- Optimal hardware configuration

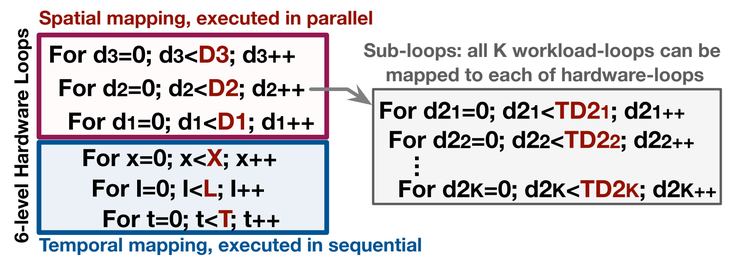

Figure 5: The tiled loops represent the workload scheduling in spatial and temporal; The trips counts of sub-loops compose the mapping-vector. Note that both CONV and MM are analyzed as a K-level nested loop.

Problem formulation:

$$\mathrm{TPE}[d_3][d_2][d_1][x][l][t] \Leftarrow workload[i_1][i_2]...[i_K]$$

$$(i_1, ..., i_K)^T = [\mathbf{T} \cdot \mathbf{H}] \cdot (d3, d2, d1, x, l, t)^T$$

$$\mathbf{T} = (\overrightarrow{TD1}, \overrightarrow{TD2}, \overrightarrow{TD3}, \overrightarrow{TX}, \overrightarrow{TL}, \overrightarrow{TT})^T = \begin{bmatrix} TD1_1 & TD2_1 & TD3_1 & TX_1 & TL_1 & TT_1 \\ TD1_2 & TD2_2 & TD3_2 & TX_2 & TL_2 & TT_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ TD1_K & TD2_K & TD3_K & TX_K & TL_K & TT_K \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ D_2 & 1 & 0 & 0 & 0 & 0 \\ D_1 & D_1 & 1 & 0 & 0 & 0 \\ X & X & X & 1 & 0 & 0 \\ L & L & L & L & 1 & 0 \\ T & T & T & T & T & 1 \end{bmatrix}$$

Top-200 optimal solutions:

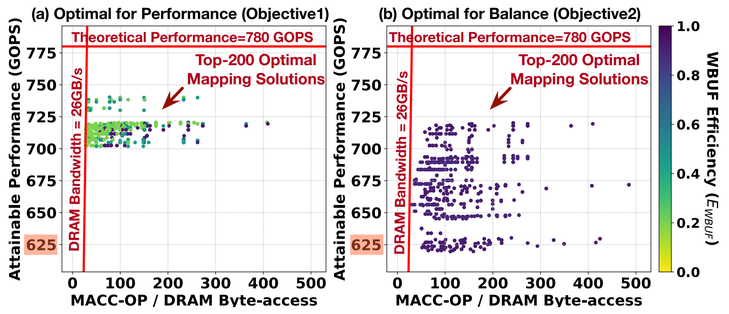

Figure 6: Roofline-based visualization tool for performance analysis. (a) and (b) plot top-200 optimal solutions by FTDL compiler for *performance* and *balance* objectives respectively. The solution in (b) is preferable as they saves WBUF 5$\times$ to (a) with only slight performance loss. Note that the y-axis has been scaled to the area of interest.

Comparison with Related Work

Work	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	**FTDL (this work)**
Weight Quantization	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit	**16-bit**
DSP Frequency (MHz)	150	100	125	167	200	200	150	150	170	240	**650**
Hardware Efficiency	45.4%	73.0%	72.0%	67.5%	48.3%	48.2%	71.9%	70.8%	76.5%	89.1%	**81.1%/74.8%**
GoogLeNet Perf. (FPS)	52.0 (1.0x)	55.7 (1.1x)	68.7 (1.3x)	86.1 (1.7x)	73.8 (1.4x)	73.5 (1.4x)	82.3 (1.6x)	81.1 (1.6x)	99.3 (1.9x)	163.3 (3.1x)	**402.6 (7.7x)**
ResNet50 Perf. (FPS)	21.2 (1.0x)	22.7 (1.1x)	28.0 (1.3x)	35.0 (1.7x)	30.1 (1.4x)	29.9 (1.4x)	33.5 (1.6x)	33.0 (1.6x)	40.4 (1.9x)	66.5 (3.1x)	**151.2 (7.1x)**
Power Efficiency (GOPS/W)	N/A	16.8 (1.2x)	N/A	21.4 (1.5x)	N/A	N/A	14.5 (1.0x)	30.4 (2.1x)	N/A	N/A	**27.6 (1.9x)**

References

[1] Y. Ma, M. Kim, Y. Cao, S. Vrudhula, and J.-s. Seo, ''End-to-end scalable fpga accelerator for deep residual networks,'' in *ISCAS*, 2017, pp. 1–4.

[2] Z. Liu *et al.*, ''Throughput-optimized FPGA accelerator for deep convolutional neural networks,'' *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 10, no. 3, p. 17, 2017.

[3] S. I. Venieris and C.-S. Bouganis, ''Latency-driven design for FPGA-based convolutional neural networks,'' in *FPL*, 2017, pp. 1–8.

[4] L. Lu, Y. Liang, Q. Xiao, and S. Yan, ''Evaluating fast algorithms for convolutional neural networks on FPGAs,'' in *FCCM*, 2017, pp. 101–108.

[5] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, ''An automatic RTL compiler for high-throughput fpga implementation of diverse deep convolutional neural networks,'' in *FPL*, 2017, pp. 1–8.

[6] Y. Ma *et al.*, ''Optimizing the convolution operation to accelerate deep neural networks on FPGA,'' *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1354–1367, 2018.

[7] Y. Guan *et al.*, ''FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates,'' in *FCCM*, 2017, pp. 152–159.

[8] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, ''Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks,'' in *FPGA*, 2017, pp. 45–54.

[9] Y. Shen, M. Ferdman, and P. Milder, ''Maximizing CNN accelerator efficiency through resource partitioning,'' in *ISCA*, 2017, pp. 535–547.

[10] X. Wei, *et al.*, ''Automated systolic array architecture synthesis for high throughput cnn inference on FPGAs,'' in *DAC*, 2017, p. 29.