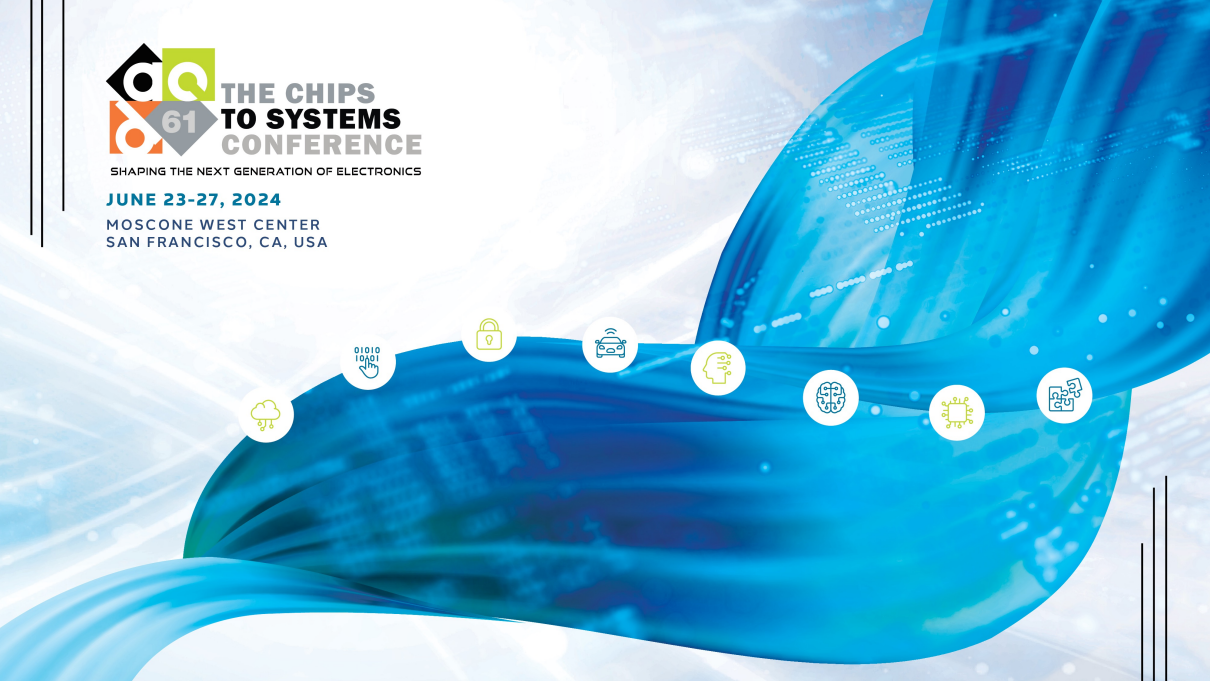

61
THE CHIPS
TO SYSTEMS
CONFERENCE
SHAPING THE NEXT GENERATION OF ELECTRONICS
JUNE 23-27, 2024
MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

61 THE CHIPS TO SYSTEMS CONFERENCE
SHAPING THE NEXT GENERATION OF ELECTRONICS
JUNE 23-27, 2024
MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

WinoGen: A Highly Configurable Winograd Convolution IP Generator for Efficient CNN Acceleration on FPGA

Mingjun Li[*1], Pengjia Li[*2], Shuo Yin[1], Shixin Chen[1], Beichen Li[2], Chong Tong[2], Jianlei Yang[3], Tinghuan Chen[2], Bei Yu[1]

[1]Chinese University of Hong Kong
[2]Chinese University of Hong Kong, Shenzhen
[3]Beihang University

Outline

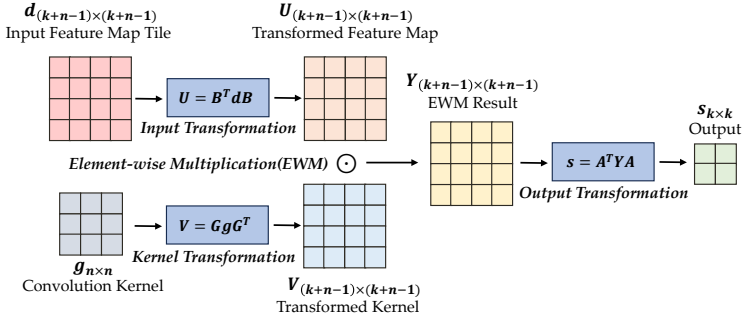
1. Introduction
2. Structured Direct Winograd Convolution
3. WinoGen IP Architecture
4. Experiments

Introduction

Winograd Convolution

Winograd Convolution Algorithm

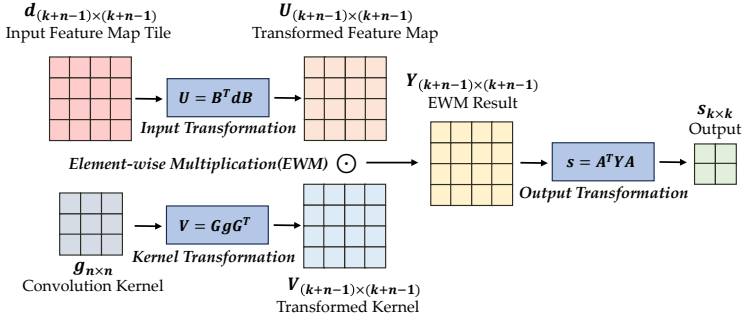
- Compute convolution by element wise multiplication in the Winograd domain.
- 2D Winograd convolution defined by $F(k, n)$: $s = d * g \Rightarrow s = A^\top \left[B^\top dB \odot GgG^\top \right] A$
  - d: $\omega \times \omega$ input feature map tile ($\omega = k + n - 1$);
  - g: $n \times n$ kernel;
  - s: $k \times k$ output.



Winograd Convolution

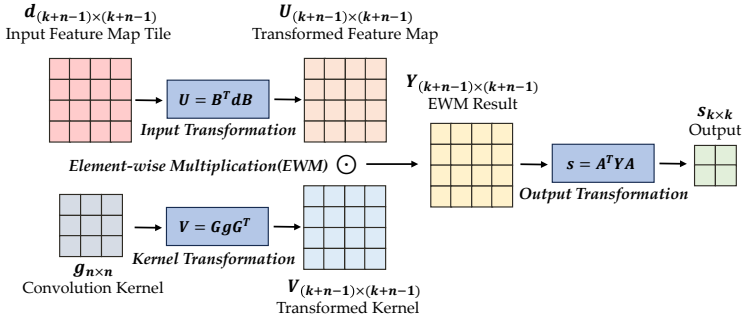
Winograd Convolution Stages

- Input transformation: $\boldsymbol{U} = \boldsymbol{B}^\top \boldsymbol{dB}$, B is $\omega \times \omega$ constant matrix decided by ω.
- Kernel transformation: $\boldsymbol{V} = \boldsymbol{GgG}^\top$, G is $\omega \times n$ constant matrix decided by $\{k, n\}$.
- Element-wise multiplication: $\boldsymbol{Y} = \boldsymbol{U} \odot \boldsymbol{V}$.
- Output transformation: $\boldsymbol{s} = \boldsymbol{A}^\top \boldsymbol{YA}$, A is $\omega \times k$ constant matrix decided by $\{k, n\}$.



Pros&Cons of Winograd Convolution

- **Advantage**: Reduce multiplication operations (from k^2n^2 to ω^2), thus saving DSP utilization on FPGA accelerators. But this will only be achieved when transformation stages are efficient enough.
- **Disadvantage**: When tile or kernel size (k or n) changes, the dimension and values in transformation matrices ($\boldsymbol{B}$, $\boldsymbol{G}$ and $\boldsymbol{A}$) will also change, making it challenging for efficient and homogeneous hardware implementation.



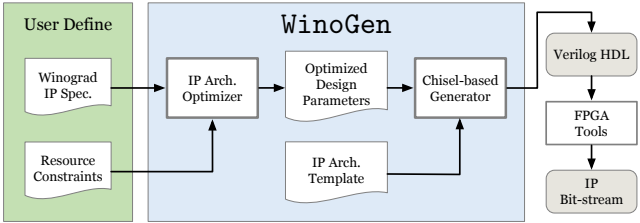
Challenges & Motivation

Challenges Exists in Efficient Winograd Hardware Implementation

- Building efficient and highly compatible IP for arbitrary Winograd convolution on FPGA needs a lot of **manual efforts**.
- **Dynamically** supporting multiple kernel and tile sizes using a single IP remain under-explored.
- Using Winograd convolution under relatively large kernel or tile sizes while maintaining high **resource efficiency** is difficult.
- The Decomposition-based Winograd method needs complex tiling architecture and dedicated memory hierarchy, thus being **unsuitable for IP generation**.

Motivation: **efficient**, **automatic** and **user-friendly** Winograd hardware IP generation for **arbitrary** kernel & tile size, and being **dynamically reconfigurable**.

WinoGen Framework



User Specifications

- Winograd convolution config. (k and n in $\boldsymbol{F}(k, n)$), input/output datawidth.
- Hardware resource constraints.

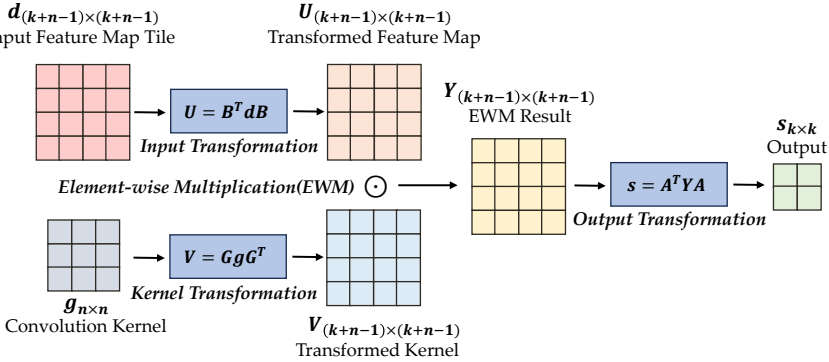
WinoGen Configurability

- Able to generate IP for arbitrary forms of Winograd convolution $\boldsymbol{F}(k, n)$ computation.
- Generated IPs dynamically support $\boldsymbol{F}(k', n')$, as long as $\omega' \leq \omega$ and $k' \leq k$.

Structured Direct Winograd Convolution

Structured Direct Winograd Convolution (SDW)

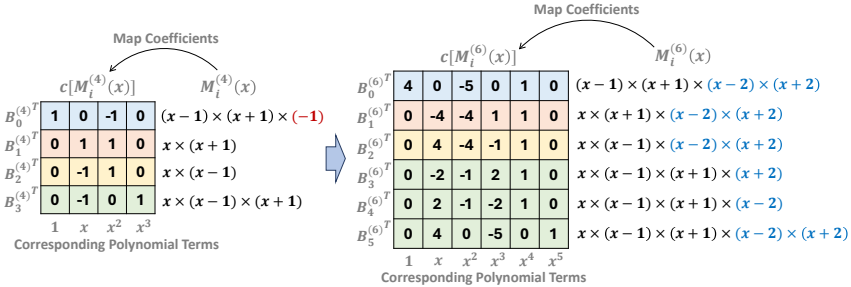
- We propose SDW with renewed input transformation and output transformation to realize arbitrary IP generation and dynamic support for multiple tile&kernel sizes.
- WinoGen FPGA IP conducts two stages above with element-wise multiplication online(on FPGA), and leave kernel transformation offline(off the chip).



Recursive Winograd Input Transformation (RIT)

Key Idea of RIT

- Compute the Winograd input transformation **recursively**.
- Build **nested** hardware to support multiple types of Winograd simultaneously.

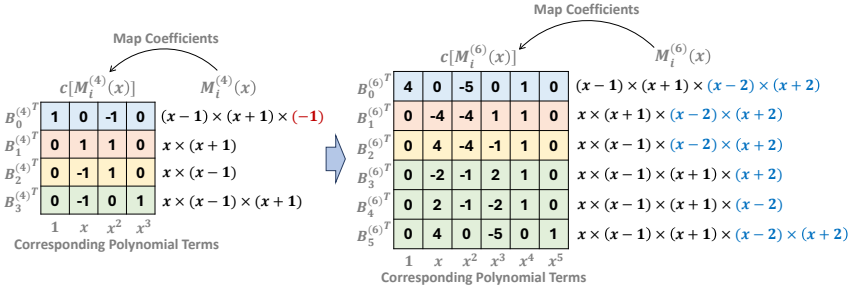


$\boldsymbol{B}^\top$ for $\boldsymbol{F}(\omega = 4)$ and $\boldsymbol{F}(\omega = 6)$. $\boldsymbol{F}(\omega)$ is used to denote the set of $\boldsymbol{F}(k, n)$ with a same ω.

Recursive Winograd Input Transformation (RIT)

Observations on Input Transformation Matrix $\boldsymbol{B}$

- Input transformation ($\boldsymbol{U} = \boldsymbol{B}^\top \boldsymbol{dB}$) is only determined by ω.
- $\boldsymbol{B}^\top$ is derived by coefficients of predefined polynomial sequences $M_i^{(\omega)}(x)$.
- When ω increases by 2, the $M_i^{(\omega)}(x)$ and $\boldsymbol{B}^\top$ will expand accordingly.
- Additional computations can be calculated incrementally by **correlation** operations.



$\boldsymbol{B}^\top$ for $\boldsymbol{F}(\omega = 4)$ and $\boldsymbol{F}(\omega = 6)$. $\boldsymbol{F}(\omega)$ is used to denote the set of $\boldsymbol{F}(k, n)$ with a same ω.

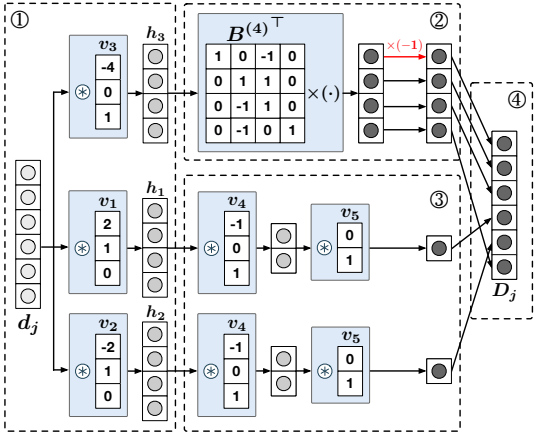
Recursive Winograd Input Transformation (RIT)

RIT procedure

1. Perform **correlation** between input feature vector and the coefficient vectors of additional polynomials.
2. Reuse matrix multiplication with $\boldsymbol{B}^\top$ of $(\omega - 2)$.
3. Perform correlation on remaining vectors.

Advantage of RIT

Preserve the hardware structure of $\boldsymbol{F}(\omega')$ while implementing the ITrans functionality of $\boldsymbol{F}(\omega)$, where $\omega' < \omega$.

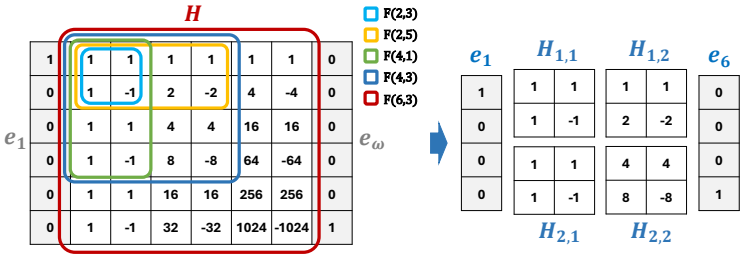


Part of `RIT` for $\boldsymbol{F}(\omega = 6)$. It generates same results as $\boldsymbol{D}_j = \boldsymbol{B}^\top \boldsymbol{d}_j$. And $\boldsymbol{U} = (\boldsymbol{B}^\top \boldsymbol{D}_j)^\top$.

Blockwise Winograd Output Transformation (BOT)

Key Idea of BOT

- Decompose $A^\top$ and Y in input transformation ($s = A^\top YA$) into $\mathbf{2} \times \mathbf{2}$ sub-matrices to carry out **blockwise** computation.
- Build hardware by stacking **minimum $\mathbf{2} \times \mathbf{2}$ units** to support multiple types of Winograd simultaneously.

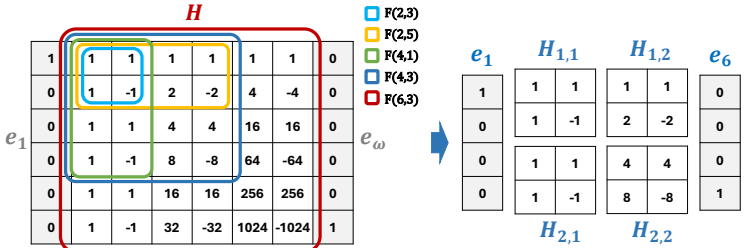


Sharing mechanism and block partition of $A^\top$ (using $F(6,3)$ and $F(4,3)$ as examples).

Blockwise Winograd Output Transformation (BOT)

Observations on Output Transformation Matrix A

- Output transformation ($s = A^\top YA$) is determined by **k** and **n**.
- When k or n increases, the **center part** of $A^\top$ will **expand** in units of **2 × 2 cells**.
- Matrix multiplication with each cell can be transformed into **add, sub and shift**.



Sharing mechanism and block partition of $A^\top$ (using $F(6,3)$ and $F(4,3)$ as examples).

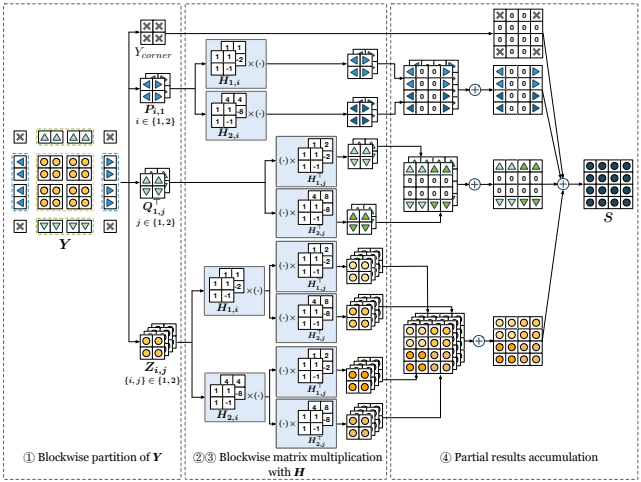
Blockwise Winograd Output Transformation (BOT)

BOT procedure

1. 2×2 sub-matrix partition of $\boldsymbol{Y}$ and $\boldsymbol{A}^\top$.
2. Blockwise sub-matrix multiplication.
3. Involves $k/2$ types of left- and right-multiplied sub-matrices.
4. Accumulate partial results.

Advantage of BOT

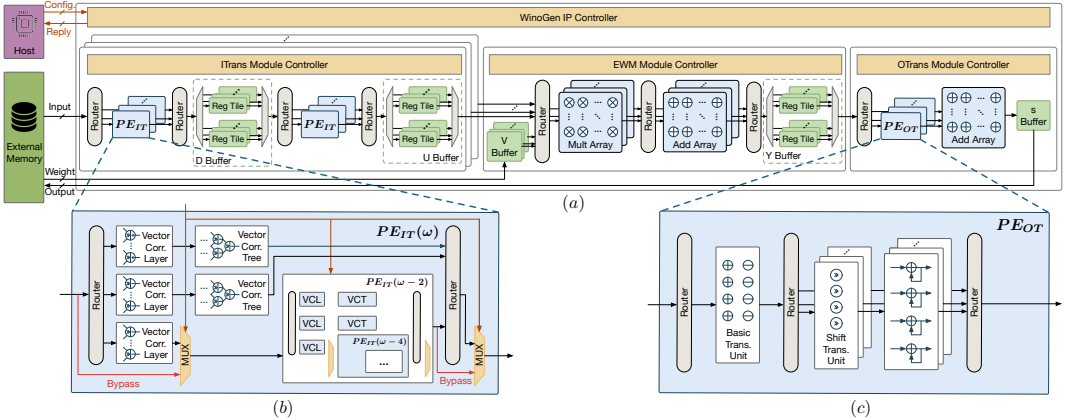
Unify hardware generation for arbitrary $\boldsymbol{F}(k, n)$ and dynamically support $\boldsymbol{F}(k', n')$, where $k' < k$.



`BOT` for $\boldsymbol{F}(4,3)$. It generates same results as $\boldsymbol{s} = \boldsymbol{A}^\top \boldsymbol{Y} \boldsymbol{A}$.

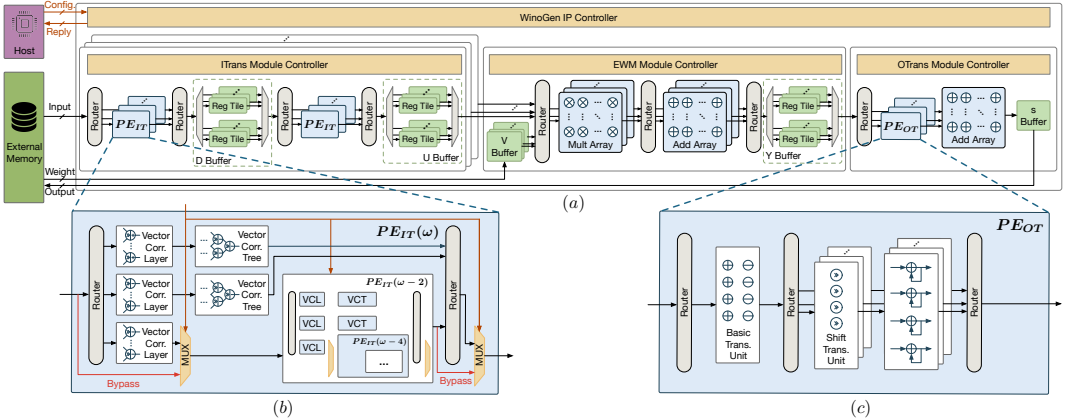
WinoGen IP Architecture

Overall Architecture of WinoGen IP



Architecture of the Winograd acceleration IP generated by WinoGen: (a) the overall architecture of WinoGen IP; (b) the detailed architecture of PE_{IT}; (c) the detailed architecture of PE_{OT}.

Overall Architecture of WinoGen IP



Configurability: can construct IP for arbitrary $\boldsymbol{F}(k, n)$, and dynamically support $\boldsymbol{F}(k', n')$, as long as $\omega' \leq \omega$ and $k' \leq k$.

Experiments

IP-level Evaluation

Table: Optimized WinoGen IP under different resource constraints on Xilinx ZCU104 FPGA (* is used to test the throughput of the IP)

IP Type	Config. Type	IP Config.				Resource Util. (Constraint)			Thro. /GOPS				Supported Winograd Type
		PN_{IT}	PN_{EWM}	PN_{OT}	PN_C	DSP	LUT	FF	1×1 Kernel	3×3 Kernel	5×5 Kernel	7×7 Kernel	
F(4,1)	PN_{min}	1	1	1	1	4	1689	2191	1.308	5.559	-	-	F(2,3)*, F(4,1)*
	Constrained	4	4	4	1	16(20)	2267(2400)	2901	5.232	22.236	-	-	
	PN_{max}	4	4	4	4	64	4858	7579	36.624	92.868	-	-	
F(4,3)	PN_{min}	1	1	1	1	6	2441	3587	1.308	9.883	7.121	-	F(2,5)*, F(4,3)*; F(2,3), F(4,1)*
	Constrained	2	2	3	4	48(64)	8267(8400)	11678	12.208	123.824	86.764	-	
	PN_{max}	6	6	9	4	144	17472	18476	36.624	371.472	260.292	-	
F(6,3)	PN_{min}	1	1	1	1	8	3757	5703	1.308	12.508	16.023	7.930	F(2,7)*, F(4,5)*, F(6,3)*; F(2,5), F(4,3), F(6,1)*; F(2,3), F(4,1)
	Constrained	8	8	16	2	128(128)	31899(40000)	25307	35.316	412.02	517.968	255.06	
	PN_{max}	8	8	16	4	256	41093	40238	82.404	**835.812**	**1041.168**	511.428	

Operator-level Comparison

- In cases where hardware resources are abundant, `WinoGen` can create IPs with ultra-high computational performance (over **1000** GOPS).
- For other scenarios, `WinoGen` can utilize its optimizer to generate optimized IPs.
- Generated IPs can support a wide range of Winograd convolutions.

Accelerator-level Evaluation

Table: WinoGen evaluation and comparison with state-of-the-art designs.

	DNNBuilder[1]		LCE[2]	HybridDNN[3]	Vitis-AI[4]		WinoCNN[5]		WinoGen IP-F(4,3)×12			WinoGen IP-F(6,3)×5		
Platform	KU115		VCU118	VU9P	ZCU102		ZCU102		ZCU104			ZCU104		
CNN Model	VGG16	AlexNet	VGG16	VGG16	VGG16	YOLOv2	VGG16	YOLOv2	VGG16	AlexNet	YOLOv2	VGG16	AlexNet	YOLOv2
Freq. (MHz)	235	220	200	167	281		214		317			317		
Supported Precision	8 bit / 16 bit	8 bit / 16 bit	16 bit	12 bit	8 bit		8-16 bit		8-16 bit			8-16 bit		
DSP (% of total)	4318 (78%)	4318 (88%)	3224 (47%)	5163 (75%)	1926 (76%)		2345 (93%)		1728 (100%)			1280 (74.1%)		
LUT (% of total)	257862 (39%)	257862 (40%)	-	-	-		-		209664 (91%)			205465 (89.2%)		
Power (W)	22.3	22.9	-	45.9	-		-		8.25			7.45		
Thro. (GOPS)	4022	3265	3772	3775.7	1225.2	1008	3120.3	1717.7	4453.2	4190.8	2408.6	4170.7	4863.1	2640.8
DSP Eff. (GOPS/DSP)	0.991	0.764	1.17	0.65	0.636	0.523	**1.33**	0.73	2.58	2.43	1.39	**3.26**	**3.80**	2.06
Energy Eff. (GOPS/W)	**180.4**	98.3	-	73.5	-	-	-	-	539.78	507.98	291.95	**559.83**	**652.77**	354.47

[1] X. Zhang *et al.*, "DNNBuilder: An automated tool for building high-performance DNN hardware accelerators for FPGAs", in *Proc. ICCAD*, IEEE, 2018, pp. 1–8.

[2] J. Shen *et al.*, "Toward an efficient deep pipelined template-based architecture for accelerating the entire 2-D and 3-D CNNs on FPGA", *IEEE TCAD*, vol. 39, no. 7, pp. 1442–1455, 2019.

[3] H. Ye *et al.*, "HybridDNN: A framework for high-performance hybrid DNN accelerator design and implementation", in *Proc. DAC*, IEEE, 2020, pp. 1–6.

[4] Xilinx, *Vitis-ai model zoo*, https://github.com/Xilinx/Vitis-AI/tree/master/model_zoo, 2023.

[5] X. Liu *et al.*, "WinoCNN: Kernel sharing winograd systolic array for efficient convolutional neural network acceleration on FPGAs", in *Proc. ASAP*, IEEE, 2021, pp. 258–265.

Accelerator-level Evaluation

Table: WinoGen evaluation and comparison with state-of-the-art designs.

	DNNBuilder[Zha+18]		LCE[She+19]	HybridDNN[Ye+20]	Vitis-AI[Xil23]		WinoCNN[Liu+21]		WinoGen IP-F(4,3)×12			WinoGen IP-F(6,3)×5		
Platform	KU115		VCU118	VU9P	ZCU102		ZCU102		ZCU104			ZCU104		
CNN Model	VGG16	AlexNet	VGG16	VGG16	VGG16	YOLOv2	VGG16	YOLOv2	VGG16	AlexNet	YOLOv2	VGG16	AlexNet	YOLOv2
Freq. (MHz)	235	220	200	167	281		214		317			317		
Supported Precision	8 bit / 16 bit	8 bit / 16 bit	16 bit	12 bit	8 bit		8-16 bit		8-16 bit			8-16 bit		
DSP (% of total)	4318 (78%)	4318 (88%)	3224 (47%)	5163 (75%)	1926 (76%)		2345 (93%)		1728 (100%)			1280 (74.1%)		
LUT (% of total)	257862 (39%)	257862 (40%)	-	-	-		-		209664 (91%)			205465 (89.2%)		
Power (W)	22.3	22.9	-	45.9	-		-		8.25			7.45		
Thro. (GOPS)	4022	3265	3772	3775.7	1225.2	1008	3120.3	1717.7	4453.2	4190.8	2408.6	4170.7	4863.1	2640.8
DSP Eff. (GOPS/DSP)	0.991	0.764	1.17	0.65	0.636	0.523	**1.33**	0.73	2.58	2.43	1.39	**3.26**	**3.80**	2.06
Energy Eff. (GOPS/W)	**180.4**	98.3	-	73.5	-	-	-	-	539.78	507.98	291.95	**559.83**	**652.77**	354.47

Network-level Comparison

- DSP efficiency reaches **3.80** GOPS/DSP and energy efficiency reaches **652.77** GOPS/W.
- When computing VGG16, our design achieves DSP efficiency up to **3.26** GOPS/DSP and energy efficiency up to **559.83** GOPS/W, thus showing **2.45**× and **3.10**× improvements compared with state-of-the-arts.

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

THANK YOU!