南京大學
NANJING UNIVERSITY

# Design of DNN Accelerator Based on Winograd Algorithm using Chisel

## --Bachelor Degree Defense

Respondent： **Shixin Chen**

Department： School of ESE,
Nanjing University

Email： shixinchen@smail.nju.edu.cn

Major： **Microelectronic Science and Engineering (VLSI Design and System Integration)**

# CONTENTS

南京大學
NANJING UNIVERSITY

1

## Preliminary

南京大學
NANJING UNIVERSITY

## The Development of Artificial Intelligence

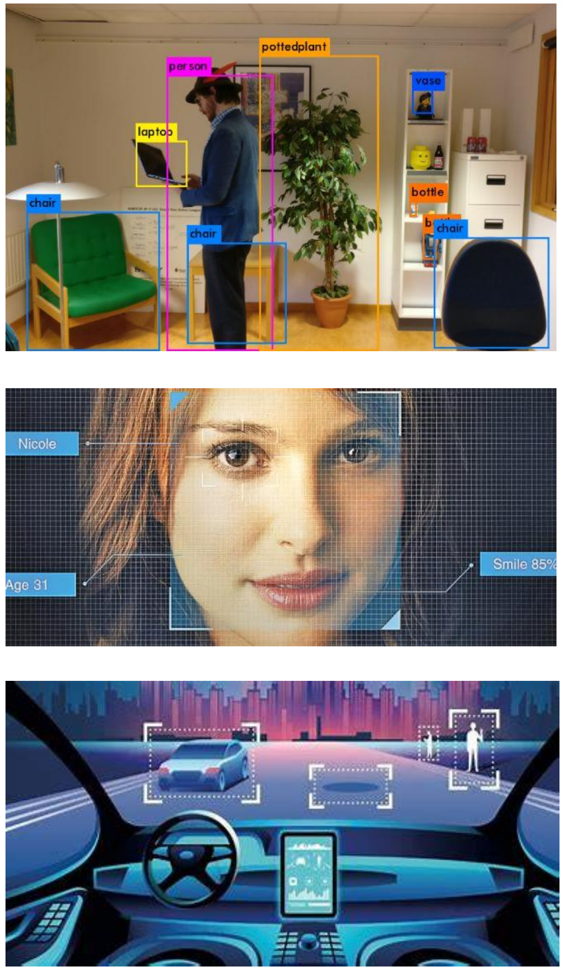


Fig 1. Wide usage of AI

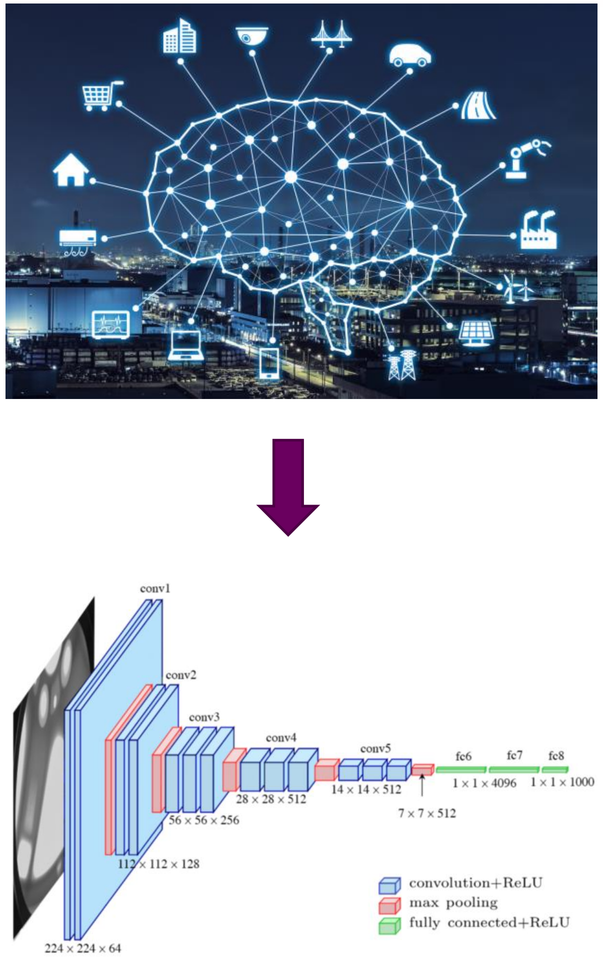
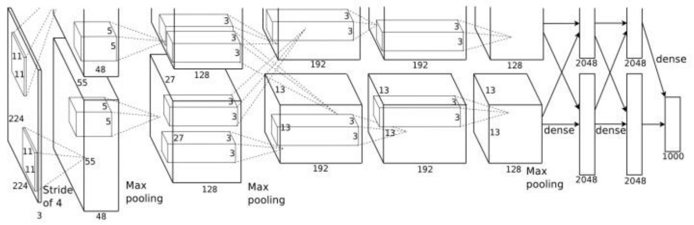


Fig 2. Deep Convolutional Neural Network

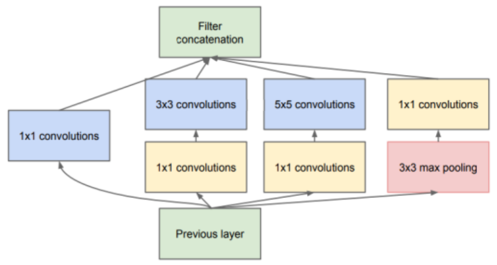
- **Artificial intelligence (AI) Applications are in popularity**
  - Object Detection
  - Face Recognition
  - Autonomous Vehicles
  - Super-resolution Videos
  - ……

- **Deep Convolutional Neural Networks (CNNs) stimulated AI research**
  - Mimicking the mechanisms of neurons
  - Extracting features effectively
  - Outperforming many traditional methods

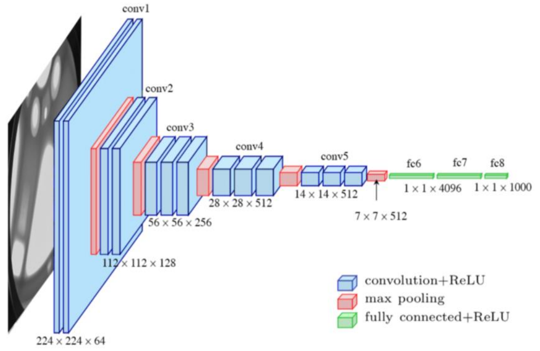
## The Development of Deep CNNs



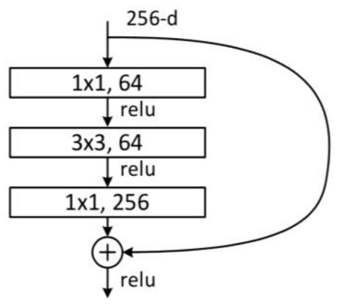
(a) AlexNet (Krizhevsky et al., 2012 )



(c) GoogLeNet (Szegedy et al., 2015)



(b) VGGNet (Simonyan et al., 2014 )



(d) ResNet (He et al., 2016)

Fig 3. The Development of Deep CNNs

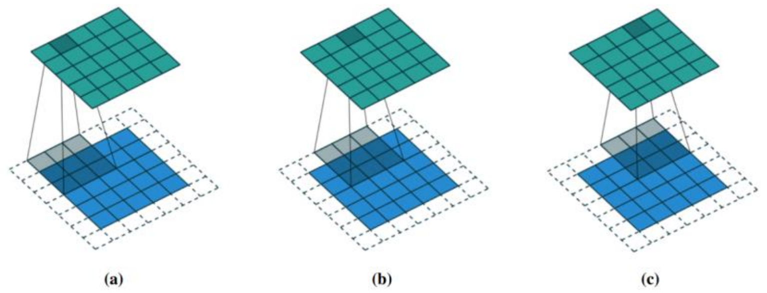


Fig 4. Convolutional Computation

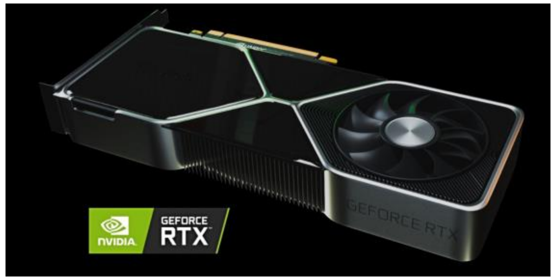
➢ **Deep CNNs developed rapidly**

- Deeper and deeper layers
- Huge parameters
- Convolution consists of 90+% computation
- **DNN Accelerator is necessary**

## Deployment Platforms of Deep CNNs

(a) CPU (Intel)

(b) FPGA (Xilinx)

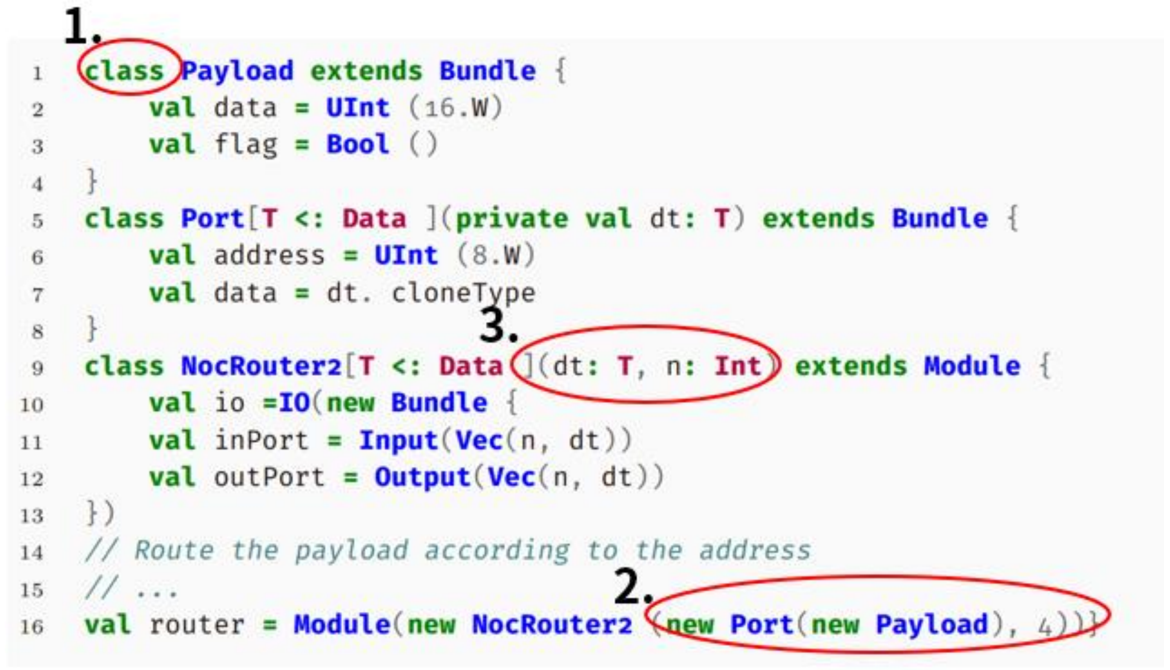


(c) GPU (NVIDIA)

Fig 5. The Deployment platforms of Deep CNNs

- **Deep CNNs deployment platforms**
  - Improving parallelism → GPU
  - Improving specialization → ASIC
  - Improving flexibility, parallelism and specialization → FPGA
- **Convolutional computation acceleration**
  - Accelerating multiplication-accumulation → Digital Signal Processor on FPGA
  - Fast convolution algorithm → FFT, Winograd
- **There are still many challenges**
  - Long development period of accelerator
  - Limited hardware resources on platforms
  - Balance between performance and resources

## Deployment Platformes of Deep CNNs



```
class Payload extends Bundle {
      val data = UInt (16.W)
      val flag = Bool ()
}
class Port[T <: Data ](private val dt: T) extends Bundle {
      val address = UInt (8.W)
      val data = dt. cloneType
}
class NocRouter2[T <: Data ](dt: T, n: Int) extends Module {
      val io =IO(new Bundle {
      val inPort = Input(Vec(n, dt))
      val outPort = Output(Vec(n, dt))
})
// Route the payload according to the address
// ...
val router = Module(new NocRouter2 (new Port(new Payload), 4)))
```

Fig 6. Chisel Code Samples: Parameterized NoC Routers

- **Drawbacks of Verilog**
  - Low level abstraction, consuming lots of time
  - Low efficiency in iterative development
  - Error-prone element types and data width

- **Characteristics of Chisel**
  - Object-oriented programming
  - Functional programming
  - High-level parameterized philosophy

  → Iteratively develop to the optimal design

  → Agile IC design

- **Chisel → FIRRTL → Verilog →Synthesis Tool**

## 1 Contribution

### Bottlenecks

- Designing accelerator is a long period using Verilog, consuming lots of manpower
- Accelerator consumes lots of on-chip resources, limiting implementation of big DNN model

### Innovative Points

- **Agile Development Perspective**
  - Design a parameterized DNN accelerator based on Winograd Algorithm using Chisel
  - Choose the optimal data quantization to balance resource and performance
- **Accelerator Design Perspective**
  - Optimize on-chip memory resource
  - Decrease 56% multiplication resources, 61% registers compared with commercial implementation
  - Achieve 3.6x speed compared with CPU

2

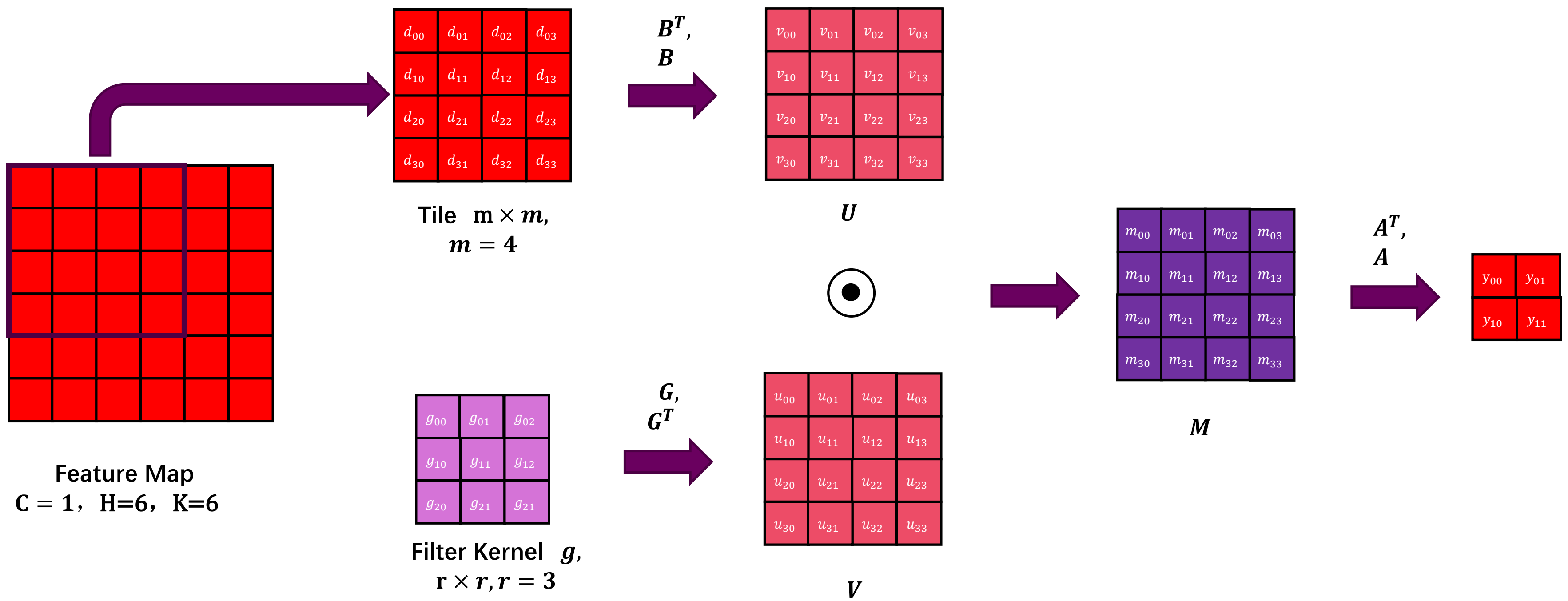
## Winograd Accelerator Architecture

南京大學
NANJING UNIVERSITY

## Global Architecture of Winograd Algorithm

- **Winograd algorithm can represent traditional convolutional algorithm, which uses addition operation to replace multiplication, saving 56% multiplication resources in theoretical analysis.**

$$\mathbf{Y} = \mathbf{A}^T \left[ (\mathbf{G}\mathbf{g}\mathbf{G}^T) \odot (\mathbf{B}^T \mathbf{d}\mathbf{B}) \right] \mathbf{A}.$$



$d_{00}$ $d_{01}$ $d_{02}$ $d_{03}$
$d_{10}$ $d_{11}$ $d_{12}$ $d_{13}$
$d_{20}$ $d_{21}$ $d_{22}$ $d_{23}$
$d_{30}$ $d_{31}$ $d_{32}$ $d_{33}$
Tile $m \times m$, $m = 4$
$B^T$, $B$
$v_{00}$ $v_{01}$ $v_{02}$ $v_{03}$
$v_{10}$ $v_{11}$ $v_{12}$ $v_{13}$
$v_{20}$ $v_{21}$ $v_{22}$ $v_{23}$
$v_{30}$ $v_{31}$ $v_{32}$ $v_{33}$
$U$
$\odot$
$m_{00}$ $m_{01}$ $m_{02}$ $m_{03}$
$m_{10}$ $m_{11}$ $m_{12}$ $m_{13}$
$m_{20}$ $m_{21}$ $m_{22}$ $m_{23}$
$m_{30}$ $m_{31}$ $m_{32}$ $m_{33}$
$M$
$A^T$, $A$
$y_{00}$ $y_{01}$
$y_{10}$ $y_{11}$
Feature Map
C = 1, H=6, K=6
$g_{00}$ $g_{01}$ $g_{02}$
$g_{10}$ $g_{11}$ $g_{12}$
$g_{20}$ $g_{21}$ $g_{21}$
Filter Kernel $g$, $r \times r, r = 3$
$G$, $G^T$
$u_{00}$ $u_{01}$ $u_{02}$ $u_{03}$
$u_{10}$ $u_{11}$ $u_{12}$ $u_{13}$
$u_{20}$ $u_{21}$ $u_{22}$ $u_{23}$
$u_{30}$ $u_{31}$ $u_{32}$ $u_{33}$
$V$

Fig 7. $F(2 \times 2,\ 3 \times 3)$ Winograd Algorithm Computing Flow

## 2 Winograd Accelerator Architecture

### Global Architecture of Winograd Accelerator

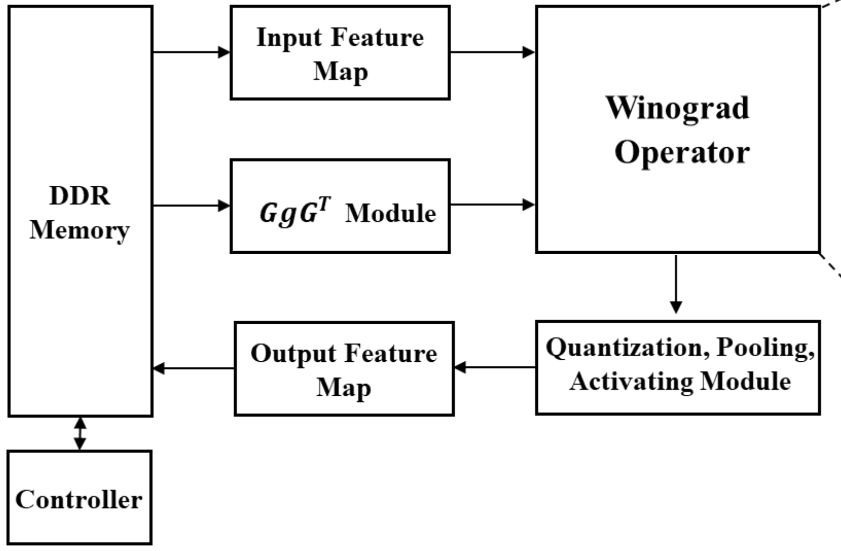


Fig 8. Global Architecture of Winograd Accelerator

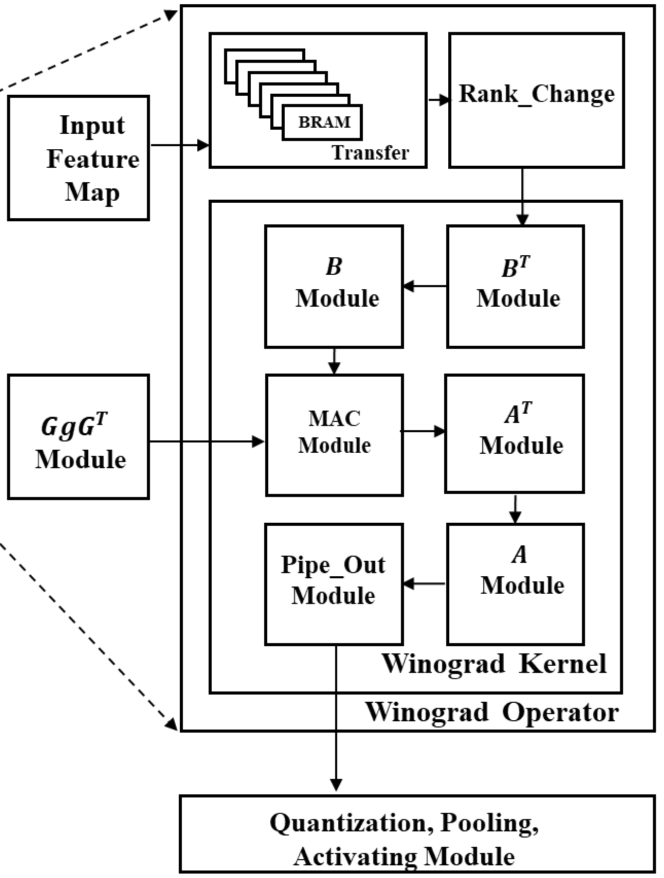


Fig 9. Winograd Operator

- Design a full-system accelerator base on Winograd using Chisel for the first time
- Save on-chip memory, only 6 BRAMs for rearrangement of data
- Change data type only by one line of code, highly parameterized
- Increase parallelism, resource utilization and throughout on 9-stage pipeline

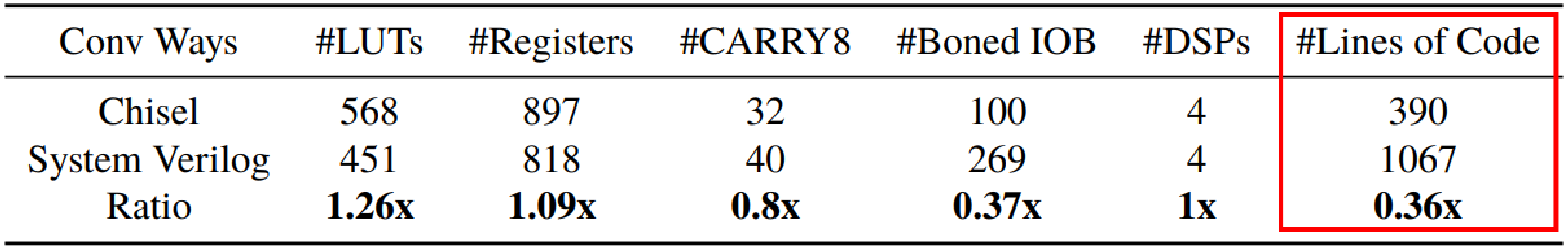
## 2 Winograd Accelerator Architecture

### Agile Development Analysis

- An engineer of a commercial company uses System Verilog to realize the same Winograd convolution operator

- Our work decreases 64% code.

Tab 1. Resources of Chisel and System Verilog on Single Winograd Channel



| Conv Ways | #LUTs | #Registers | #CARRY8 | #Boned IOB | #DSPs | #Lines of Code |
|---|---|---|---|---|---|---|
| Chisel | 568 | 897 | 32 | 100 | 4 | 390 |
| System Verilog | 451 | 818 | 40 | 269 | 4 | 1067 |
| Ratio | **1.26x** | **1.09x** | **0.8x** | **0.37x** | **1x** | **0.36x** |

# 2 Winograd Accelerator Architecture

## Design of on-Chip memory Buffer

➢ **Our design uses 6 BRAM to rearrange data and achieve data reuse in rows， meeting Winograd computing flow requirements and saving on-chip memory.**

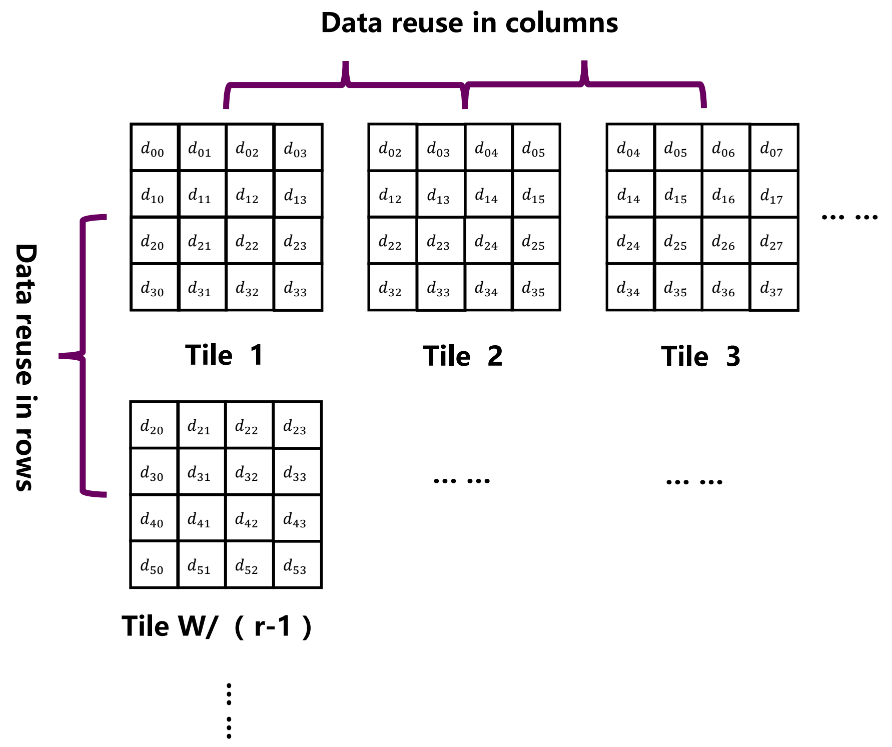


Fig 10. $F(2\times 2,\ 3\times 3)$ Data Tile Flow

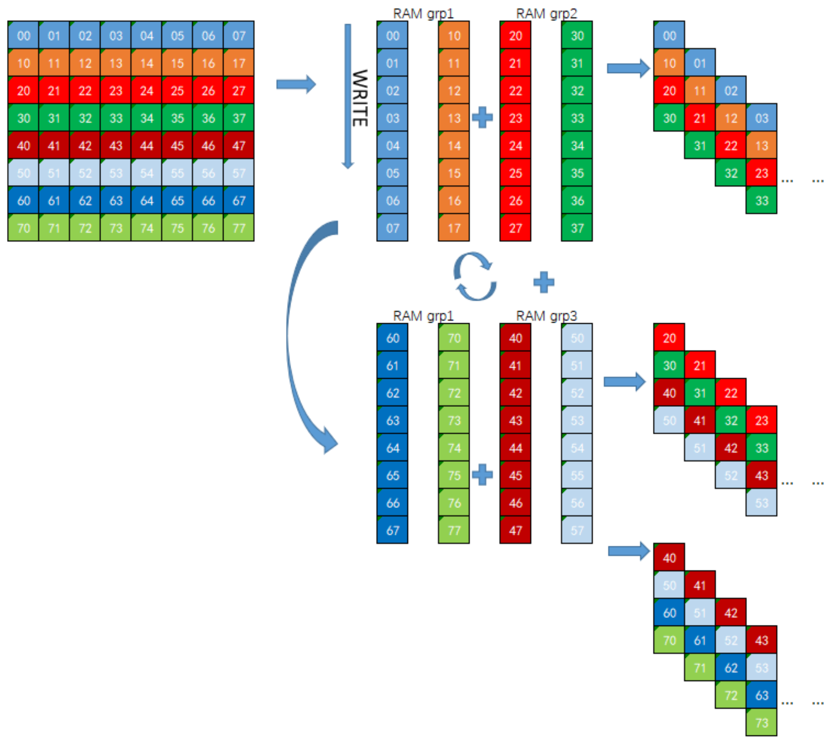
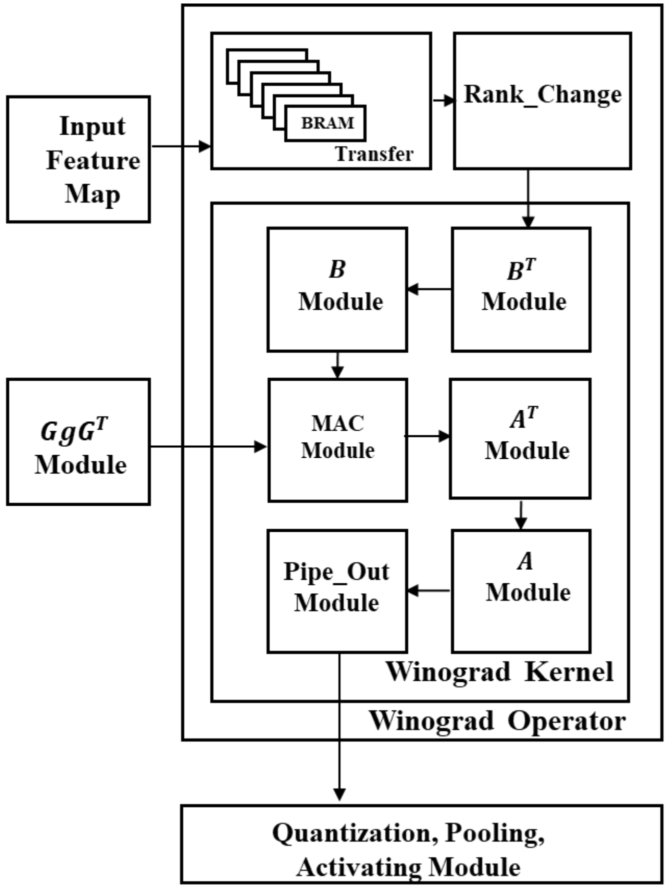
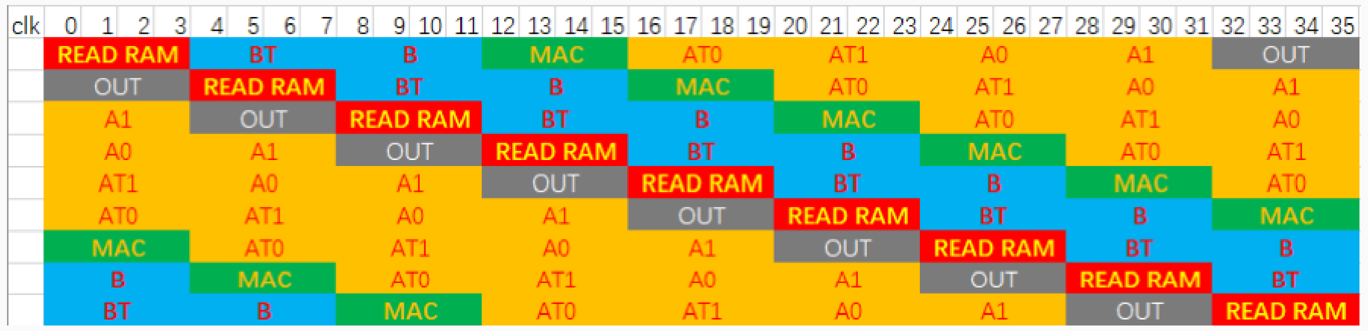


Fig 11. **Transfer Design (Data reuse in row)**

## Pipeline Design of Architecture



- In the architecture, we design a 9-stage pipeline to continuously process the input data and improve throughput.



| clk | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | READ RAM | | | | BT | | | | B | | | | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | | OUT | | | |
| | OUT | | | | READ RAM | | | | BT | | | | B | | | | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | |
| | A1 | | | | OUT | | | | READ RAM | | | | BT | | | | B | | | | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | |
| | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | | BT | | | | B | | | | MAC | | | | AT0 | | | | AT1 | | | |
| | AT1 | | | | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | | BT | | | | B | | | | MAC | | | | AT0 | | | |
| | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | | BT | | | | B | | | | MAC | | | |
| | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | | BT | | | | B | | | |
| | B | | | | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | | BT | | | |
| | BT | | | | B | | | | MAC | | | | AT0 | | | | AT1 | | | | A0 | | | | A1 | | | | OUT | | | | READ RAM | | | |

Fig 12. C=1, $F(2 \times 2,\ 3 \times 3)$Winograd 9-stage pipeline

3

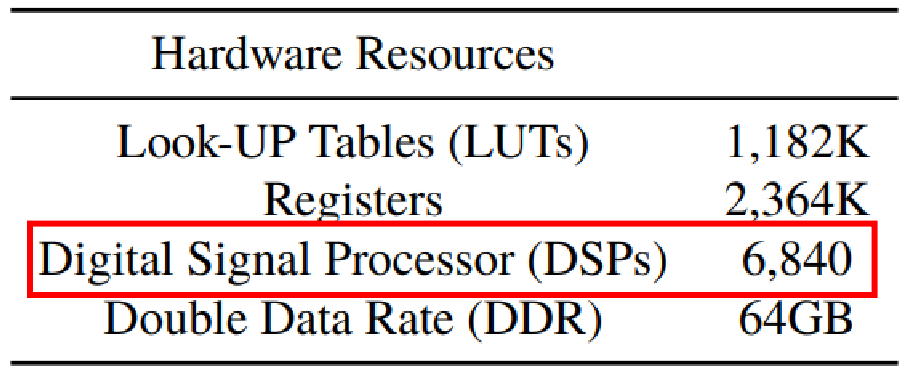
## Analysis of Performance

南京大學
NANJING UNIVERSITY

## 3 Analysis of Performance

### Deployment Platform and Resource

**Tab 2. XILINX Alevo U200 on-Chip Resources**



| Hardware Resources | |
|---|---|
| Look-UP Tables (LUTs) | 1,182K |
| Registers | 2,364K |
| Digital Signal Processor (DSPs) | 6,840 |
| Double Data Rate (DDR) | 64GB |

- **Deployment Platform: XILINX Alevo U200 FPGA**
- **Instantiate 32 Winograd operator channel in our design**
- **DSPs are expensive and usually become the bottleneck of DNN deployment**
- **Therefore, decreasing the number of DSPs in single unit will promote DNNs to deploy on cheaper FPGAs**

## 3 Analysis of Performance

### Hardware Resource Analysis

- **Baseline : MAC-Tree Acceleration with the same throughout**
- **Baseline is used in commercial company, implemented by experienced FPGA engineer**
- **Our design decreases 52% DSPs and 61% registers**

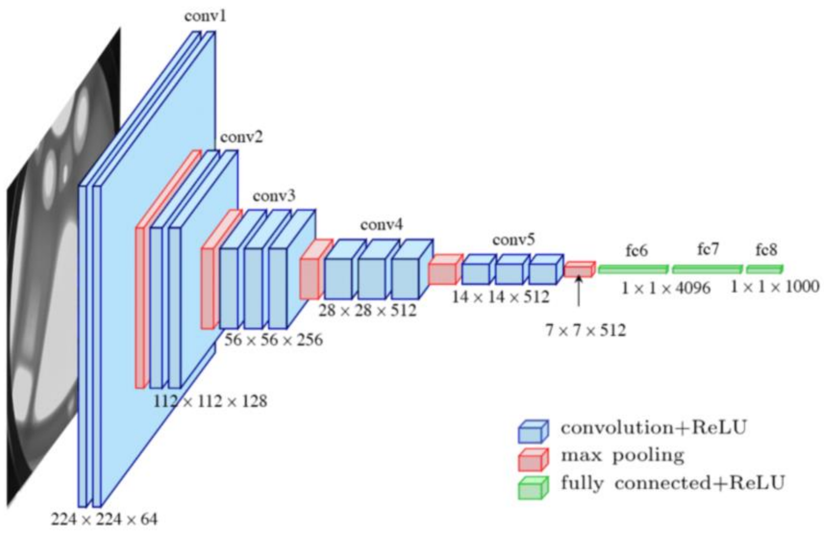
**Tab 3. Resources of Winograd and MAC-Tree in Single Channel on FPGA**

| Convolution Ways | #LUTs | #Registers | #CLB | #CARRY8 | #DSPs |
|---|---|---|---|---|---|
| Winograd | 337172 | 456302 | 66597 | 18772 | 2382 |
| MAC-Tree | 316887 | 1173517 | 138527 | 7661 | 4942 |
| Ratio | **1.06x** | **0.39x** | **0.48x** | **2.45x** | **0.48x** |

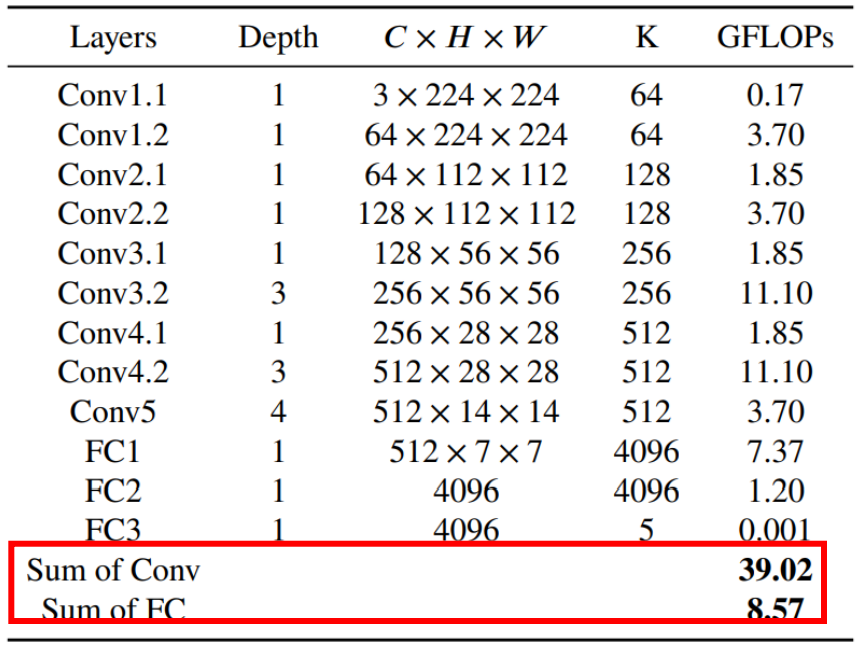
## 3 Analysis of Performance

### Acceleration Analysis

- **Tasks :** Inference acceleration of VGG-E neural network for classifying flowers
- **Neural Network Architecture:** 16 Convolutional layers and 3 full-connected layers
- **Convolutional Computation:** 39 GFLOPs



**Tab 4. VGG-E Parameter**



| Layers | Depth | $C \times H \times W$ | K | GFLOPs |
|---|---|---|---|---|
| Conv1.1 | 1 | $3 \times 224 \times 224$ | 64 | 0.17 |
| Conv1.2 | 1 | $64 \times 224 \times 224$ | 64 | 3.70 |
| Conv2.1 | 1 | $64 \times 112 \times 112$ | 128 | 1.85 |
| Conv2.2 | 1 | $128 \times 112 \times 112$ | 128 | 3.70 |
| Conv3.1 | 1 | $128 \times 56 \times 56$ | 256 | 1.85 |
| Conv3.2 | 3 | $256 \times 56 \times 56$ | 256 | 11.10 |
| Conv4.1 | 1 | $256 \times 28 \times 28$ | 512 | 1.85 |
| Conv4.2 | 3 | $512 \times 28 \times 28$ | 512 | 11.10 |
| Conv5 | 4 | $512 \times 14 \times 14$ | 512 | 3.70 |
| FC1 | 1 | $512 \times 7 \times 7$ | 4096 | 7.37 |
| FC2 | 1 | 4096 | 4096 | 1.20 |
| FC3 | 1 | 4096 | 5 | 0.001 |
| Sum of Conv | | | | **39.02** |
| Sum of FC | | | | **8.57** |

## 3 Analysis of Performance

### Acceleration Analysis

- **Baseline : VGG-E inference on Intel CORE i5 CPU**
- **Achieve about 3.6x inference speed**

**Tab 5. Inference time**

| Deployment Platforms | CPU | FPGA Winograd | Ratio |
|---|---|---|---|
| Infrence time | 1398ms | 385.42ms | **3.6x** |

- **Setting: FPGA@200M Hz**

$$T_{adder\_tree} = \log_2 C, \tag{3.1}$$

$$cycles = \frac{(T_{pipeline} + T_{adder\_tree} + H \times W) \times K \times C}{K_{wino}}, \tag{3.2}$$

$$t = cycles/F. \tag{3.3}$$

$$VGG\_conv\_time = \frac{\sum_{l=1}^{16}((T_{pipeline} + T_{adder\_tree} + H_l \times W_l) \times K_l \times C_l/K_{wino}}{F}. \tag{3.4}$$

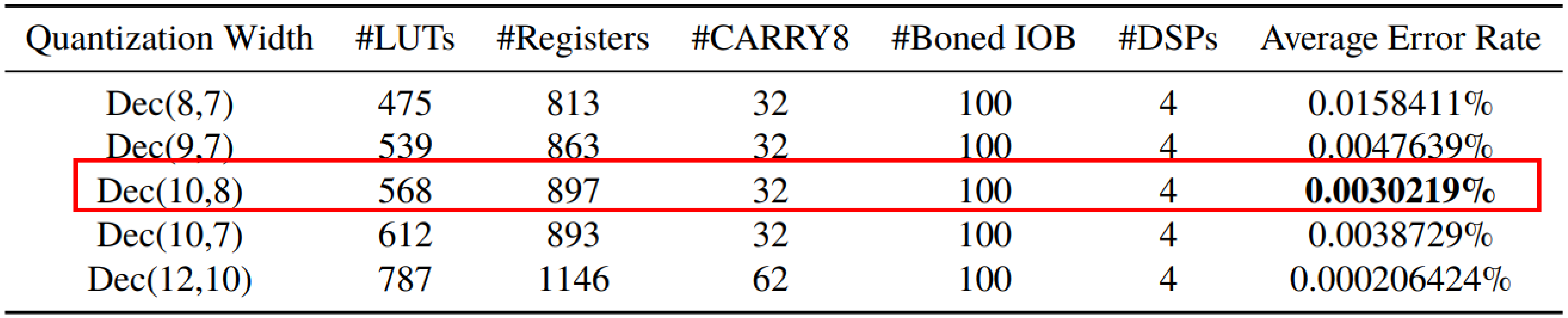
$$VGG\_time = VGG\_conv\_time + T_{pooling} + T_{fc} + T_{mem}. \tag{3.5}$$

## 3 Analysis of Performance

### Agile Development Analysis

- **The optimal quantization width is decided by our highly parameterized architecture**
- **The optimal config, with data error only 0.003%, decreases 30% utilization of LUTs compared with full-precision quantization**
- **With rapid iterative development enabled by Chisel , we can balance the performance and resources**

**Tab 6. Resources of Winograd quantization width of Winograd Operator**



| Quantization Width | #LUTs | #Registers | #CARRY8 | #Boned IOB | #DSPs | Average Error Rate |
|---|---|---|---|---|---|---|
| Dec(8,7) | 475 | 813 | 32 | 100 | 4 | 0.0158411% |
| Dec(9,7) | 539 | 863 | 32 | 100 | 4 | 0.0047639% |
| Dec(10,8) | 568 | 897 | 32 | 100 | 4 | **0.0030219%** |
| Dec(10,7) | 612 | 893 | 32 | 100 | 4 | 0.0038729% |
| Dec(12,10) | 787 | 1146 | 62 | 100 | 4 | 0.000206424% |

4

## Conclusion and Outlook

南京大學

NANJING UNIVERSITY

## 4 Conclusion and Outlook

### Conclusion

➢ **Agile Development Perspective**

- Design a parameterized DNN accelerator based on Winograd Algorithm using Chisel
- Choose the optimal data quantization to balance resource and performance

➢ **Accelerator Design Perspective**

- Optimize on-chip memory resource
- Decreasing 56% multiplication resources, 61% register compared with commercial implementation
- 3.6x speed compared with CPU

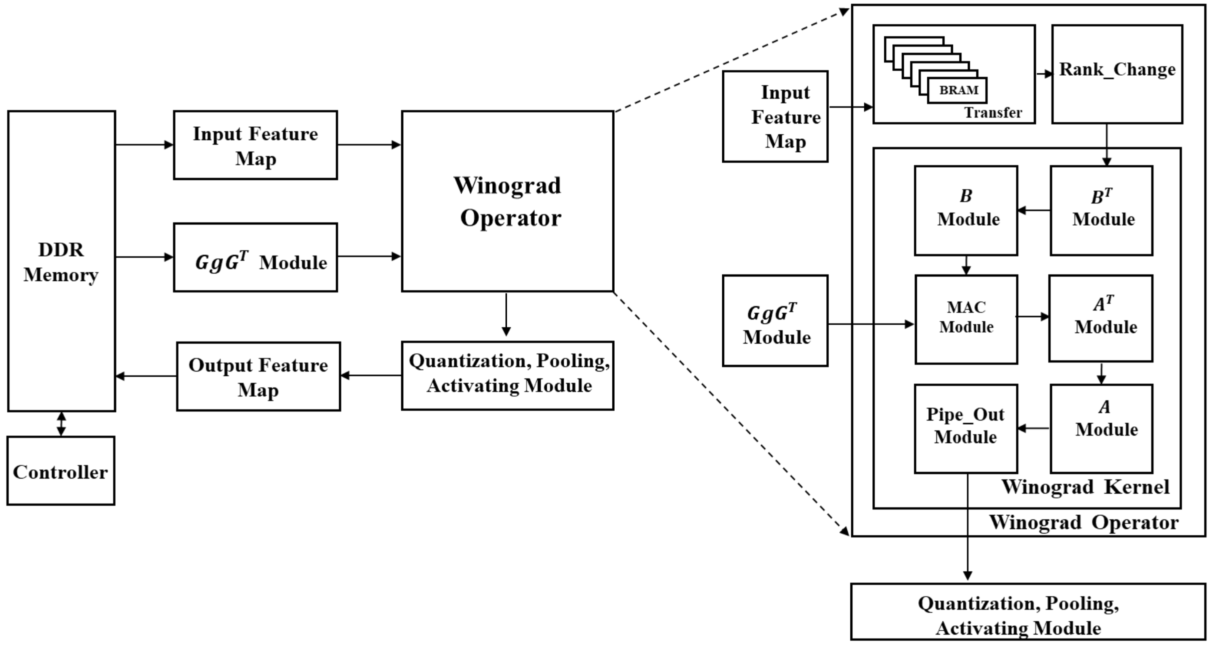


Fig 8. Global Architecture of Winograd Architecture

## 4 Conclusion and Outlook

### Agile Method

- Promote  the development of AI accelerator with agile development methods
- Build a more general DNN accelerator based on Chisel in future to make AI more accessible.
- Contribute to hardware and software co-design

南京大學
NANJING UNIVERSITY

# Thank you!

**Design of DNN Accelerator Based on Winograd Algorithm using Chisel**

Respondent： **Shixin Chen**

Department： School of ECE,
Nanjing University

Email： shixinchen@smail.nju.edu.cn

Major： **Microelectronic Science and Engineering (VLSI Deisgn and System Integration)**

## Reference

[1] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.

[2] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for largescale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[3] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[4] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 770-778. DOI: 10.1109/CVPR.2016.90.

[5] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[6] KRISHNAMOORTHI R. Quantizing deep convolutional networks for efficient inference: A whitepaper[J]. arXiv preprint arXiv:1806.08342, 2018.

[7] COURBARIAUX M, BENGIO Y, DAVID J P. Binaryconnect: Training deep neural networks with binary weights during propagations[J]. Advances in neural information processing systems, 2015, 28.

[8] BACHRACH J, VO H, RICHARDS B, et al. Chisel: constructing hardware in a scala embedded language[C]//DAC Design automation conference 2012. 2012: 1212-1221.

## Reference

[9] IZRAELEVITZ A, KOENIG J, LI P, et al. Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations[C] //2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 2017: 209-216. DOI: 10.1109/ICCAD.2017.8203780.

[10] ASANOVIC K, AVIZIENIS R, BACHRACH J, et al. The rocket chip generator[J]. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016, 4.

[11] GENC H, HAJ-ALI A, IYER V, et al. Gemmini: An agile systolic array generator enabling systematic evaluations of deep-learning architectures[J]. arXiv preprint arXiv:1911.09925, 2019, 3: 25.

[12] WINOGRAD S. Arithmetic complexity of computations: vol. 33[M]. Siam, 1980.

[13] LAVIN A, GRAY S. Fast algorithms for convolutional neural networks[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 4013-4021.

[14] LAN Q, WANG Z, WEN M, et al. High performance implementation of 3D convolutional neural networks on a GPU[J]. Computational intelligence and neuroscience, 2017, 2017.

[15] AHMAD A, PASHA M A. Towards design space exploration and optimization of fast algorithms for convolutional neural networks (CNNs) on FPGAs[C]// 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2019: 1106-1111.

## Reference

[16] WANG S, ZHU J, WANG Q, et al. Customized Instruction on RISC-V for Winograd-Based Convolution Acceleration[C]//2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP). 2021: 65-68.

[17] KUNG H T. Why systolic architectures?[J]. Computer, 1982, 15(01): 37-46.

[18] JOUPPI N P, YOUNG C, PATIL N, et al. In-datacenter performance analysis of a tensor processing unit[C]//Proceedings of the 44th annual international 35 symposium on computer architecture. 2017: 1-12.

[19] CHEN Y H, KRISHNA T, EMER J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. IEEE journal of solid-state circuits, 2016, 52(1): 127-138.

[20] CHEN Y H, YANG T J, EMER J, et al. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(2): 292-308.

[21] ASANOVIC K, PATTERSON D A, CELIO C. The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor[R]. University of California at Berkeley Berkeley United States, 2015.

[22] MATHIEU M, HENAFF M, LECUN Y. Fast training of convolutional networks through FFTS: International Conference on Learning Representations (ICLR2014), CBLS, April 2014[C]//. 2014.