A Deep Convolutional Neural Network Based on Nested Residue Number System

Hiroki Nakahara[1] Tsutomu Sasao[2]

[1]Ehime University, Japan

[2]Meiji University, Japan

Outline

- Background
- Deep convolutional neural network （DCNN）
- Residue number system （RNS）
- DCNN using nested RNS （NRNS）
- Experimental results
- Conclusion

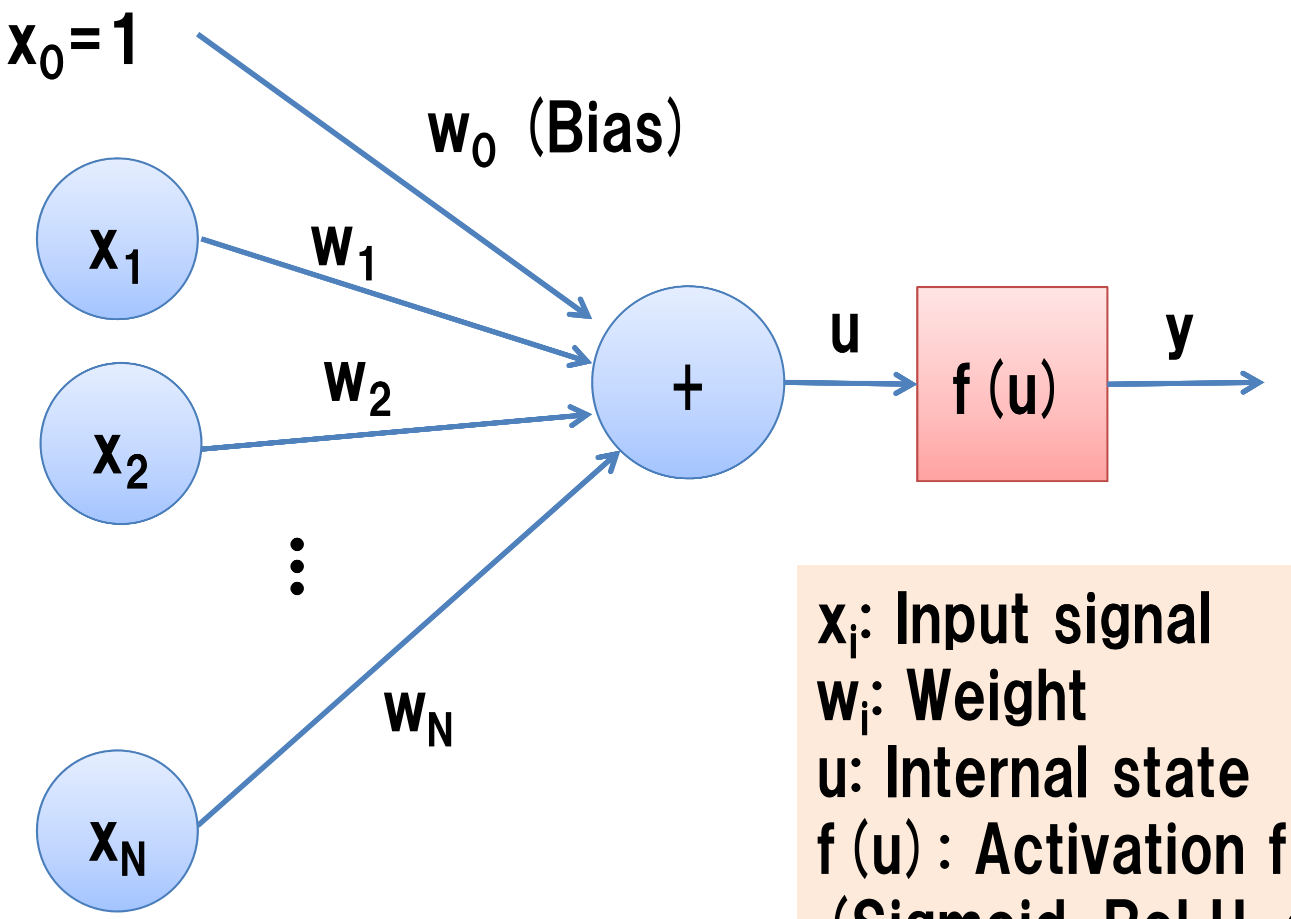
Background

- Deep Neural Network
  - Multi-layer neuron model
  - Used for embedded vision system
- FPGA realization is suitable for real-time systems
  - faster than the CPU
  - Lower power consumption than the GPU
  - Fixed point representation is sufficient

- <u>High-performance per area is desired</u>

Deep Convolutional Neural Network （DCNN）

Artificial Neuron



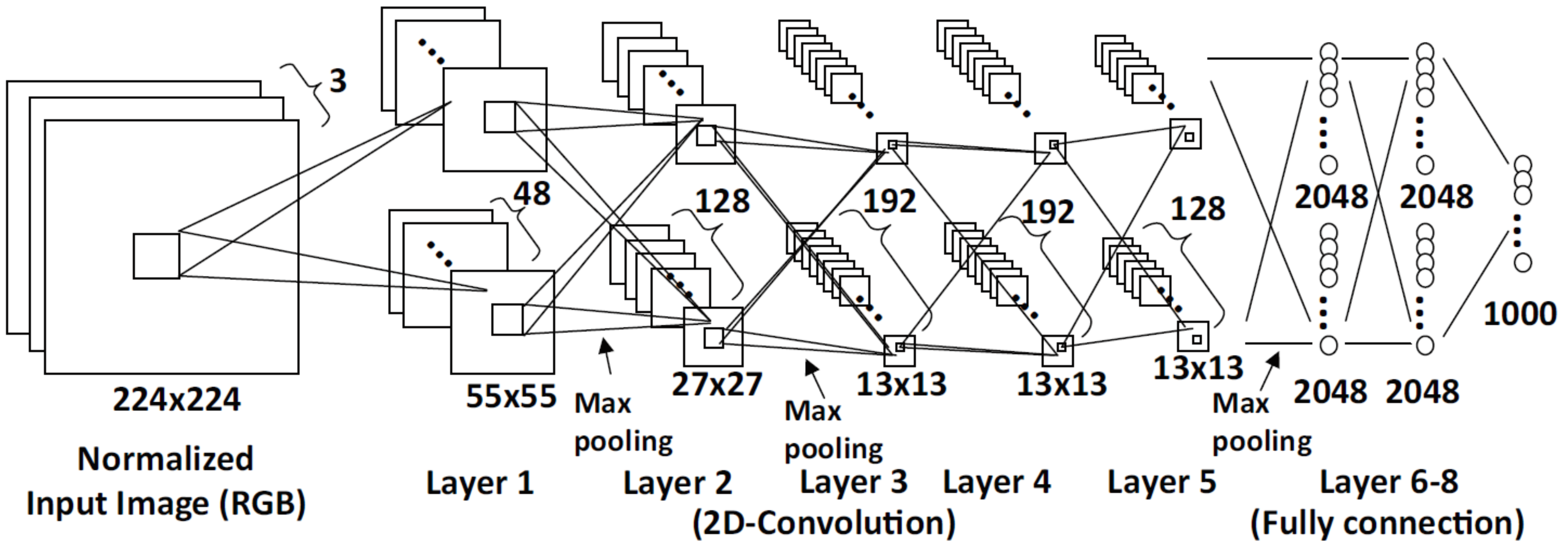
$$y = f(u)$$

$$u = \sum_{i=0}^{N} w_i x_i$$

x_i: Input signal
w_i: Weight
u: Internal state
f (u) : Activation function
(Sigmoid, ReLU, etc.)
y: Output signal

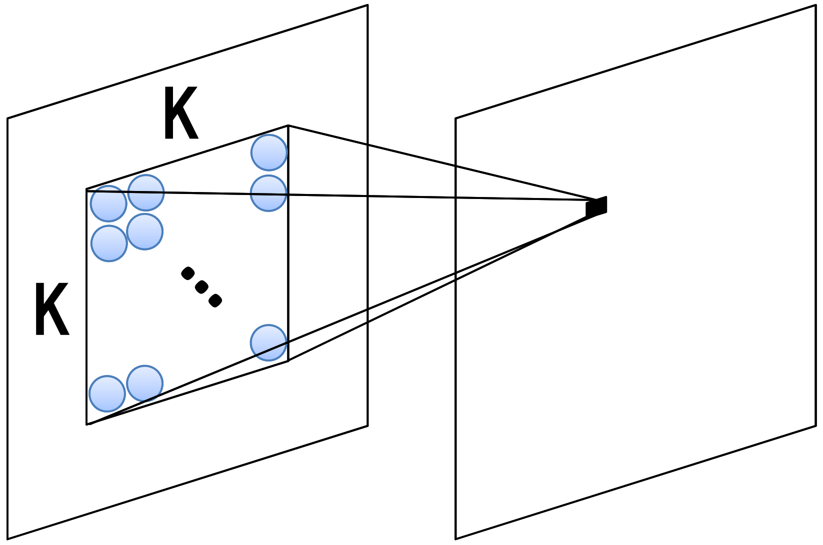
Deep Convolutional Neural Network (DCNN) for ImageNet

- 2D convolutional layer, pooling layer, and fully connection layer



2D Convolutional Layer

- Consumes more than 90% of the computation time
  - Multiply-accumulation (MAC) operation is performed

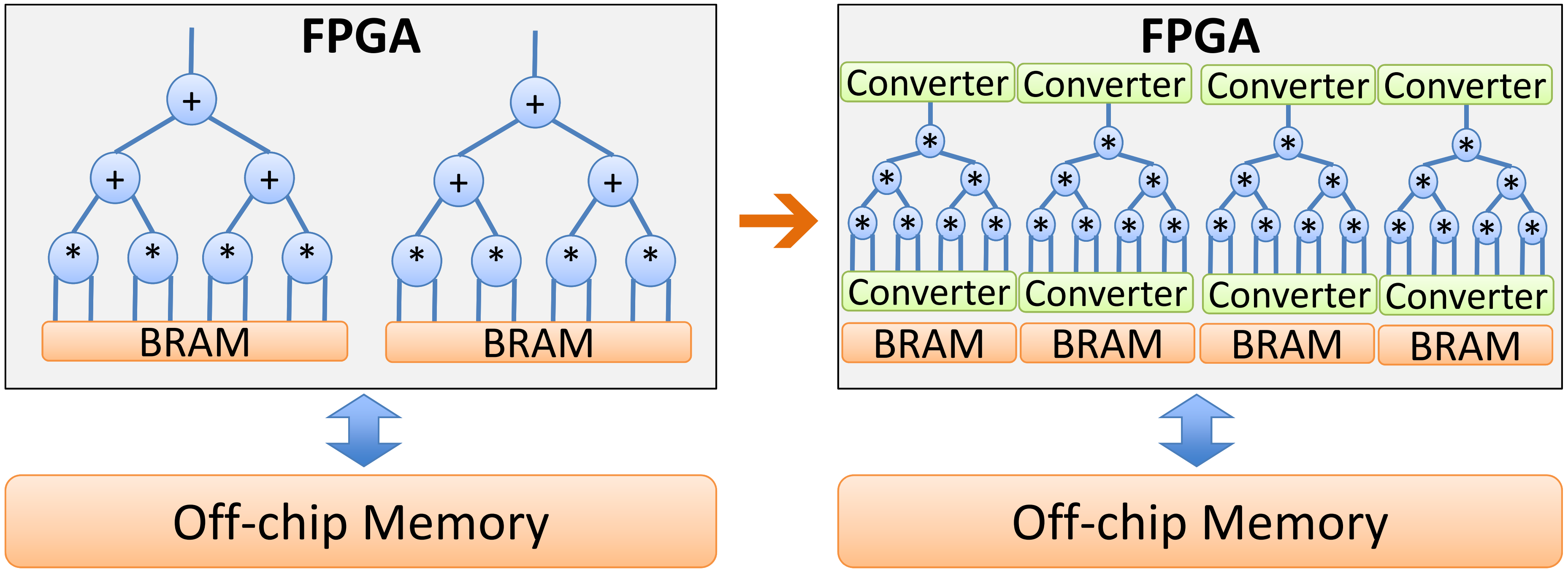


$$z_{ij} = y_{ij} + \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} x_{i+m,\,j+n} w_{mn}$$

x_{ij}: Input signal
y_{ij} : Bias
w_{mn}: Weight
K: Kernel size
z_{ij}: Output signal

Realization of 2D Convolutional Layer

- Requires more than billion MACs!
- Our realization
  - Time multiplexing
  - <u>Nested Residue Number System (NRNS)</u>



Residue Number System (RNS)

Residue Number System（RNS)

- Defined by a set of L mutually prime integer constants $\langle m_1, m_2, \ldots, m_L \rangle$
  - ✓ No pair modulus have a common factor with any other
  - ✓ Typically, prime number is used as moduli set
- An arbitrary integer X can be uniquely represented by a tuple of L integers $(X_1, X_2, \cdots, X_L)$, where $X_i \equiv X \pmod{m_i}$
- Dynamic range

$$M = \prod_{i=1}^{L} m_i$$

Multiplication on RNS

- Moduli set⟨3,4,5⟩, X=8, Y=2
- Z=X×Y=16= (1,0,1)
- X= (2,0,3) , Y= (2,2,2)

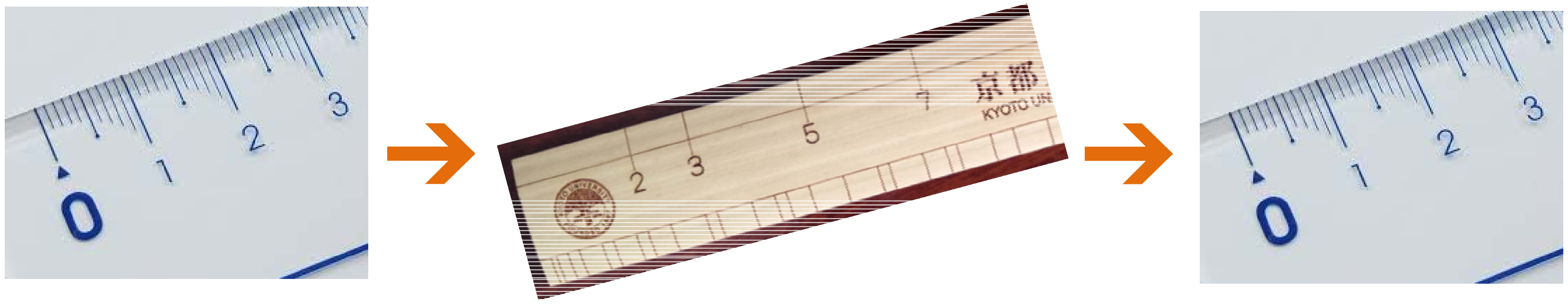
Binary2RNS Conversion

Parallel Multiplication

Z= (4 mod 3,0 mod 4,6 mod 5)

= (1,0,1) =16

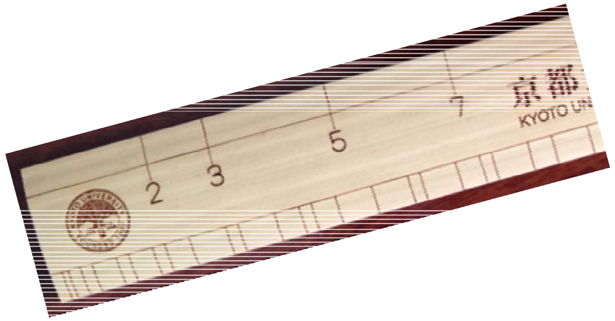
RNS2Binary Conversion



Binary2RNS Converter

X	mod 2	mod 3	mod4
0	0	0	0
1	1	1	1
2	0	2	2
3	1	0	3
4	0	1	0
5	1	2	1
⋮			

→



Functional Decomposition

Bound variables $X_1=(x_1, x_2)$

Free variables $X_2=(x_3, x_4)$

$X_2 \backslash X_1$	00	01	10	11
00	**0**	**1**	**0**	**1**
01	**1**	**1**	**1**	**1**
10	**1**	**0**	**1**	**0**
11	**1**	**0**	**1**	**0**
$h(X_1)$	0	1	0	1

Column multiplicity=2

2^4x1=16 [bit]

x1	0	0	1	1
x2	0	1	0	1
h(X1)	0	1	0	1

↓

x3,x4 \ h(X1)	0	1
00	0	1
01	1	1
10	1	0
11	1	0

2^2x1+2^3x1=12 [bit]

Decomposition Chart for X mod 3

Bound variables
$X_2=(x_3, x_4, x_5)$

$X_1=(x_1, x_2)$

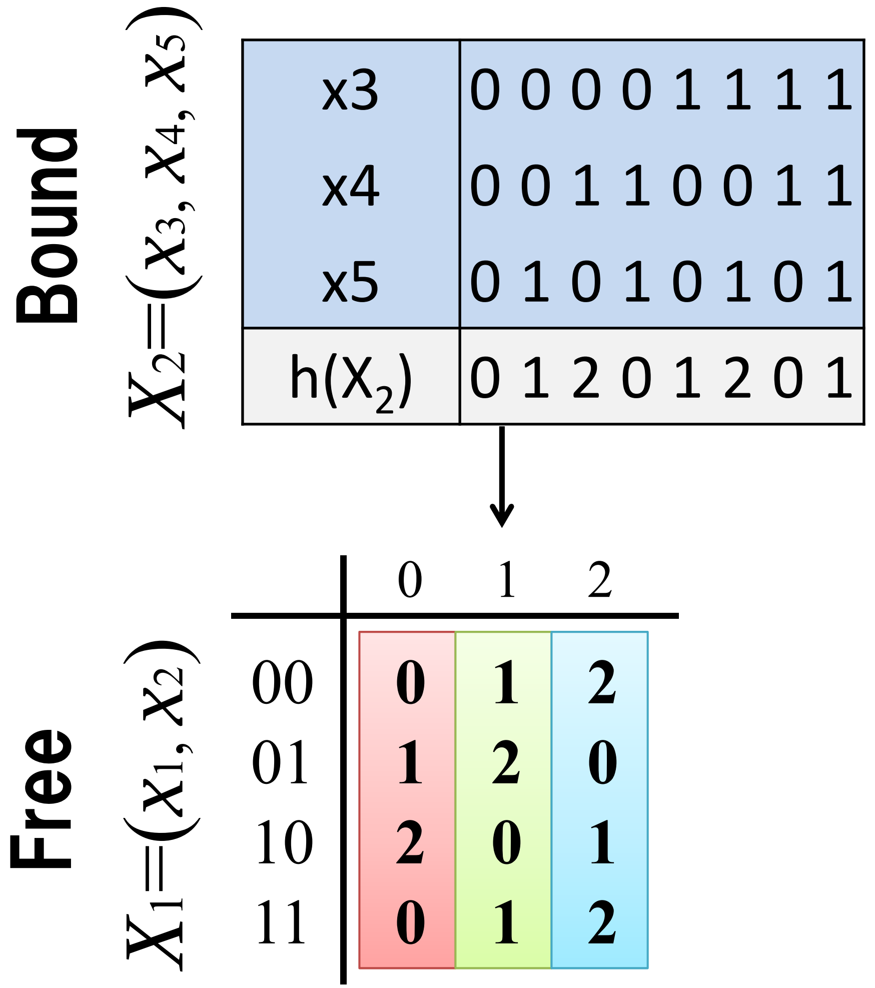
	000	001	010	011	100	101	110	111
00	0	1	2	0	1	2	0	1
01	1	2	0	1	2	0	1	2
10	2	0	1	2	0	1	2	0
11	0	1	2	0	1	2	0	1

Free variables

$$\begin{tikzcd}
0 \bmod 3 = 0 \\
1 \bmod 3 = 1 \arrow[ddd, bend left] \\
2 \bmod 3 = 2 \\
3 \bmod 3 = 0 \\
4 \bmod 3 = 1 \arrow[ddd, bend left] \\
5 \bmod 3 = 2 \\
6 \bmod 3 = 0 \\
7 \bmod 3 = 1 \arrow[ddd, bend left] \\
8 \bmod 3 = 2 \\
9 \bmod 3 = 0 \\
10 \bmod 3 = 1
\end{tikzcd}$$

⋮

Decomposition Chart for X mod 3

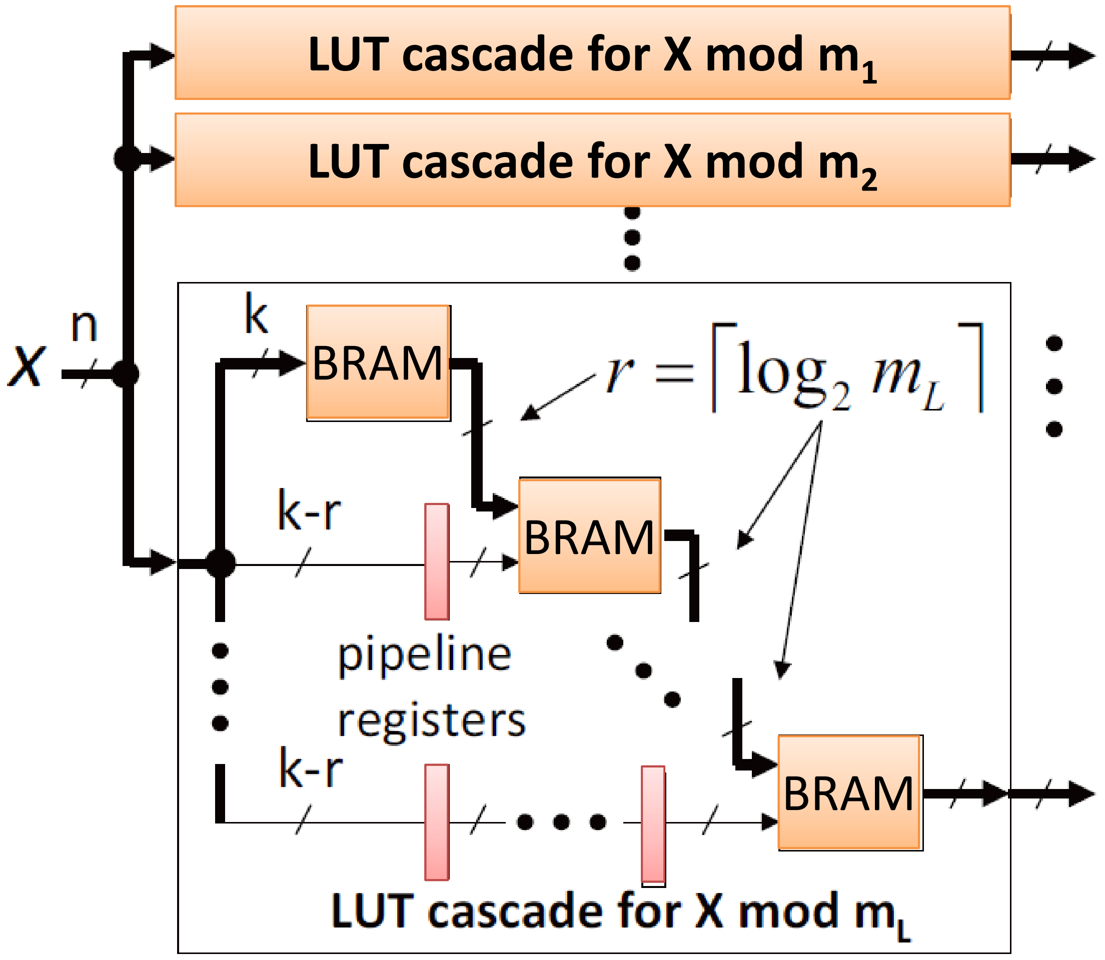


x3	0	0	0	0	1	1	1	1
x4	0	0	1	1	0	0	1	1
x5	0	1	0	1	0	1	0	1
h(X$_2$)	0	1	2	0	1	2	0	1

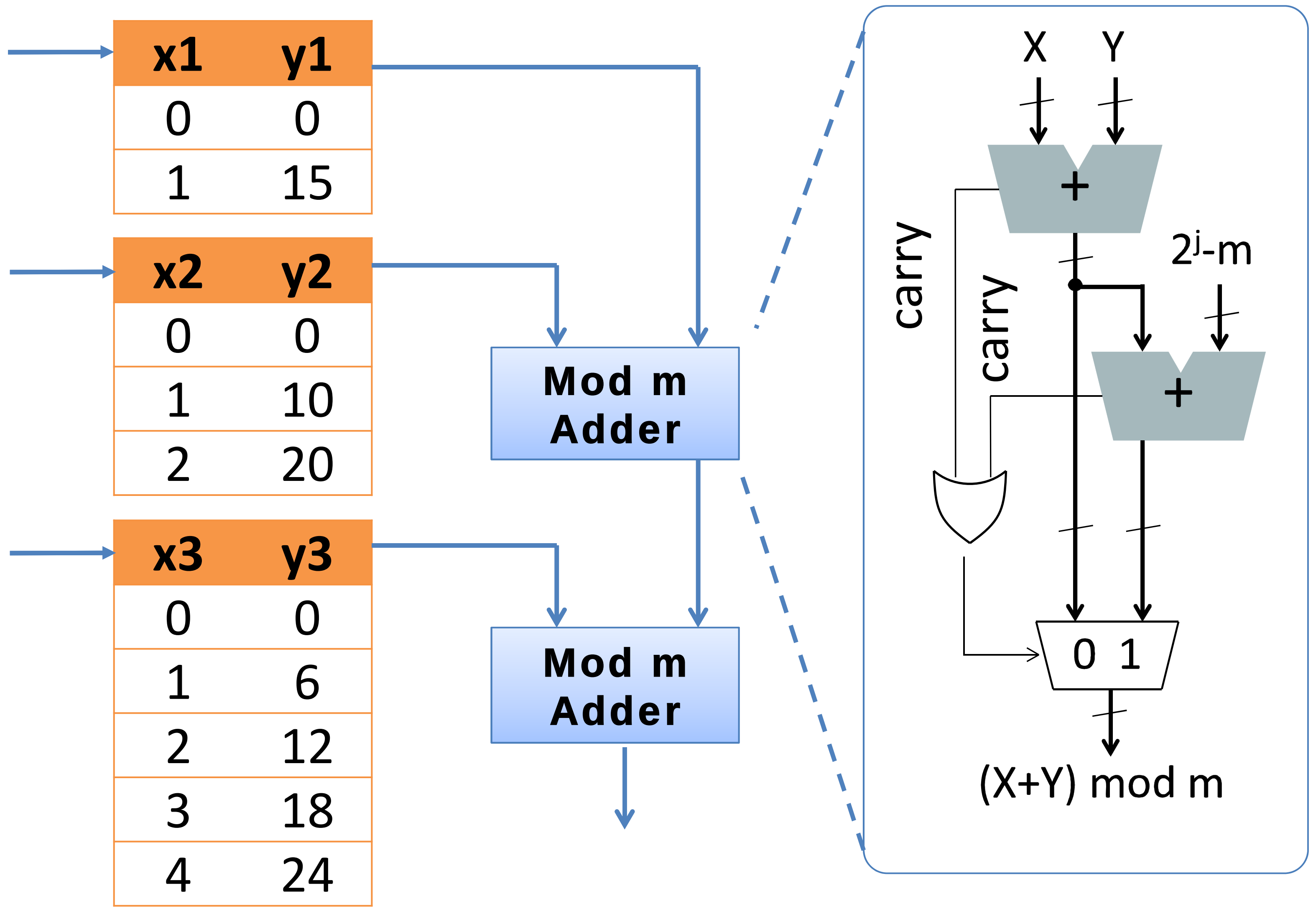
$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 00 & \mathbf{0} & \mathbf{1} & \mathbf{2} \\ 01 & \mathbf{1} & \mathbf{2} & \mathbf{0} \\ 10 & \mathbf{2} & \mathbf{0} & \mathbf{1} \\ 11 & \mathbf{0} & \mathbf{1} & \mathbf{2} \end{array}$$

0 mod 3 = 0
1 mod 3 = 1
2 mod 3 = 2
3 mod 3 = 0
4 mod 3 = 1
5 mod 3 = 2
6 mod 3 = 0
7 mod 3 = 1
8 mod 3 = 2
9 mod 3 = 0
10 mod 3 = 1
⋮

Binary2RNS Converter


LUT cascade for X mod m_1
LUT cascade for X mod m_2
X
n
k
BRAM
$r = \lceil \log_2 m_L \rceil$
k-r
BRAM
pipeline
registers
k-r
BRAM
LUT cascade for X mod m_L

RNS2Binary Converter（m=30）



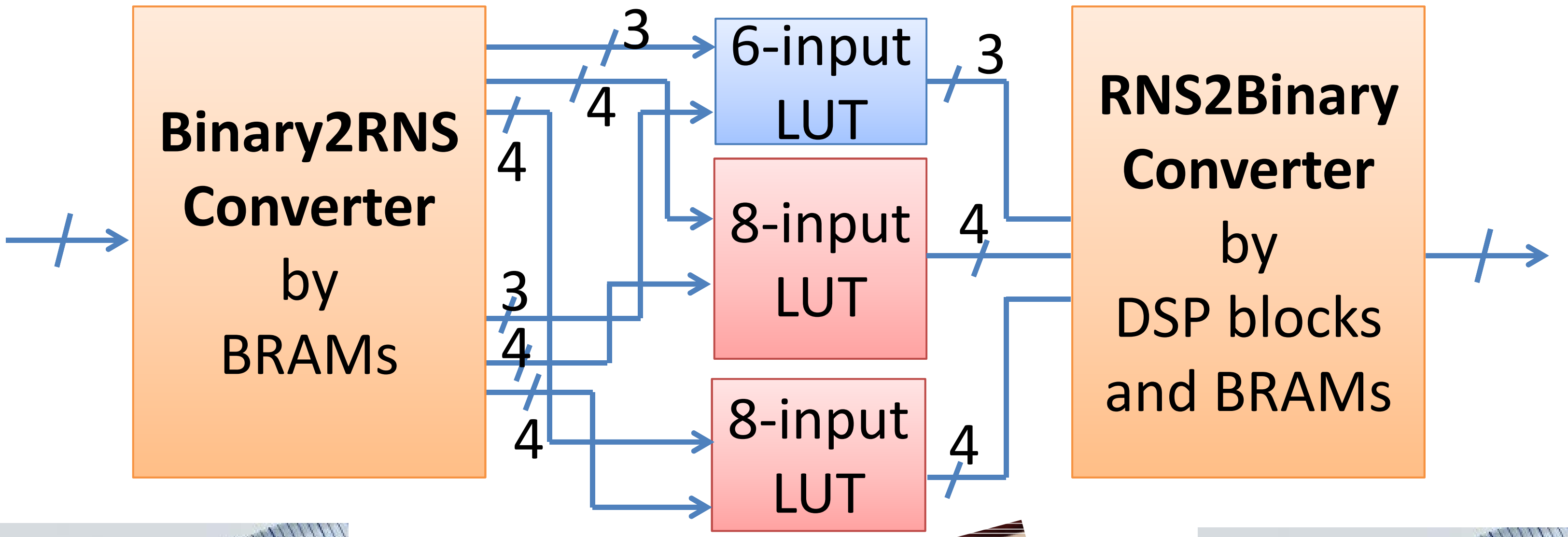
x1	y1
0	0
1	15

x2	y2
0	0
1	10
2	20

x3	y3
0	0
1	6
2	12
3	18
4	24

Problem

- Moduli set of RNS consists of mutually prime numbers
  - sizes of circuits are all different
- Example: <7,11,13>



Binary2RNS Converter by BRAMs
3
4
4
3
4
4
6-input LUT
8-input LUT
8-input LUT
3
4
4
RNS2Binary Converter by DSP blocks and BRAMs

DCNN using Nested RNS

Nested RNS

- $(Z_1, Z_2, \cdots, Z_i, \cdots, Z_L) \rightarrow (Z_1, Z_2, \cdots, (Z_{i1}, Z_{i2}, \cdots, Z_{ij}), \cdots, Z_L)$
- Ex: $\langle 7, \underline{11}, \underline{13} \rangle \times \langle 7, 11, 13 \rangle$

Original modulus

$$\langle 7, \langle 5,6,7 \rangle_{11}, \langle 5,6,7 \rangle_{13} \rangle \times \langle 7, \langle 5,6,7 \rangle_{\bigcirc\!\!\!\!\!\;11}, \langle 5,6,7 \rangle_{13} \rangle$$

1. **Reuse** the same moduli set
2. **Decompose** a large modulo into smaller ones

Example of Nested RNS

- 19x22 (=418) on $<7,<5,6,7>_{11},<5,6,7>_{13}>$

19×22

$=<5,8,6>\times<1,0,9>$

Binary2NRNS Conversion

$=<5,<3,2,1>_{11},<1,0,6>_{13}>\times<1,<0,0,0>_{11},<4,3,2>_{13}>$

Modulo Multiplication

$=<5,<0,0,0>_{11},<4,0,5>_{13}>$

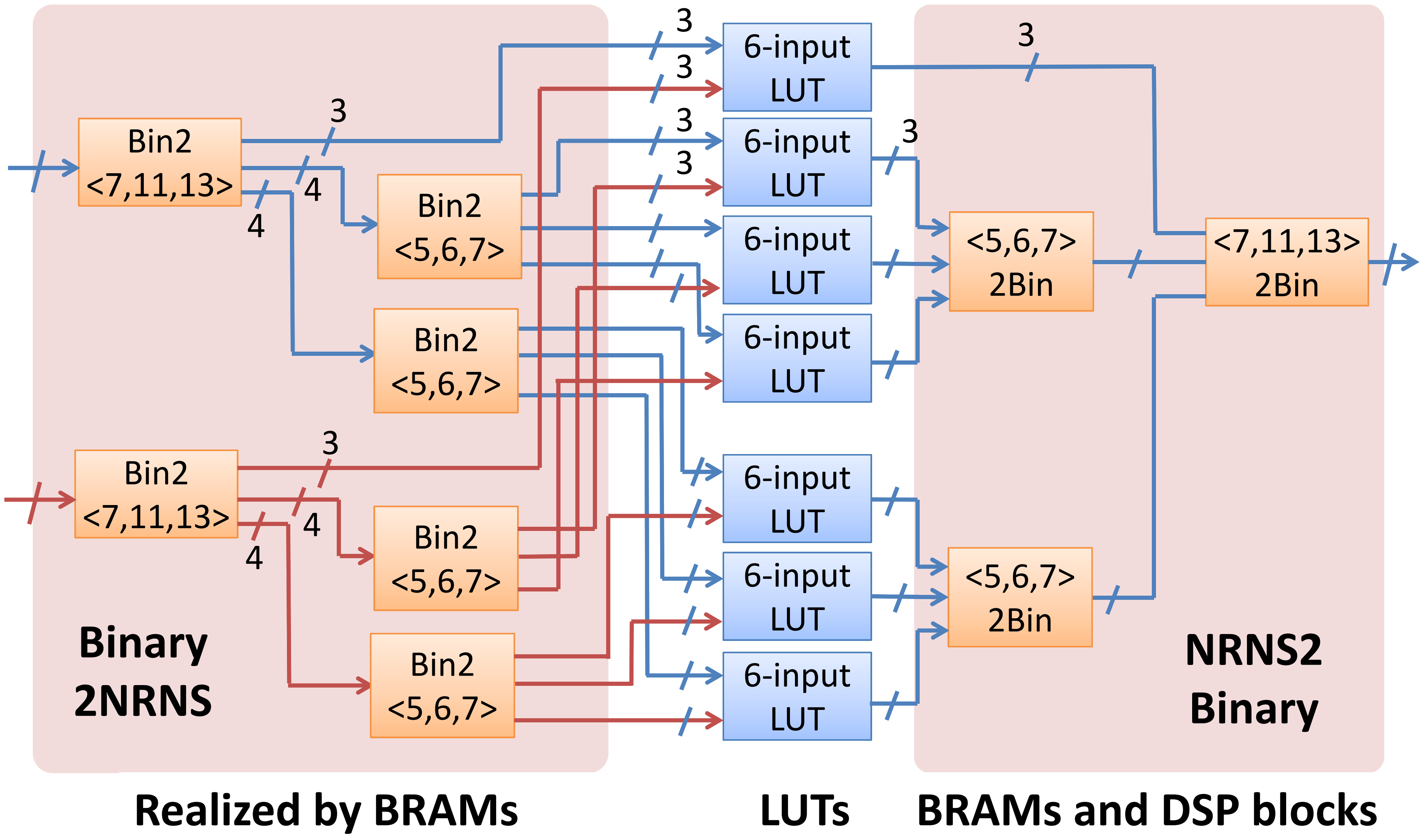
Bin2RNS on NRNS

$=<5,0,2>$

RNS2Bin

=418

Realization of Nested RNS



Realized by BRAMs LUTs BRAMs and DSP blocks

Moduli Set for NRNS

- Conventional RNS (uses 23 moduli)

  $<3,4,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83>$

- Applied the NRNS to moduli that are greater than 15

  $<3,4,5,7,11,13,$

  $<3,4,5,7,11,13>_{17},$

  $<3,4,5,7,11,13>_{19},$

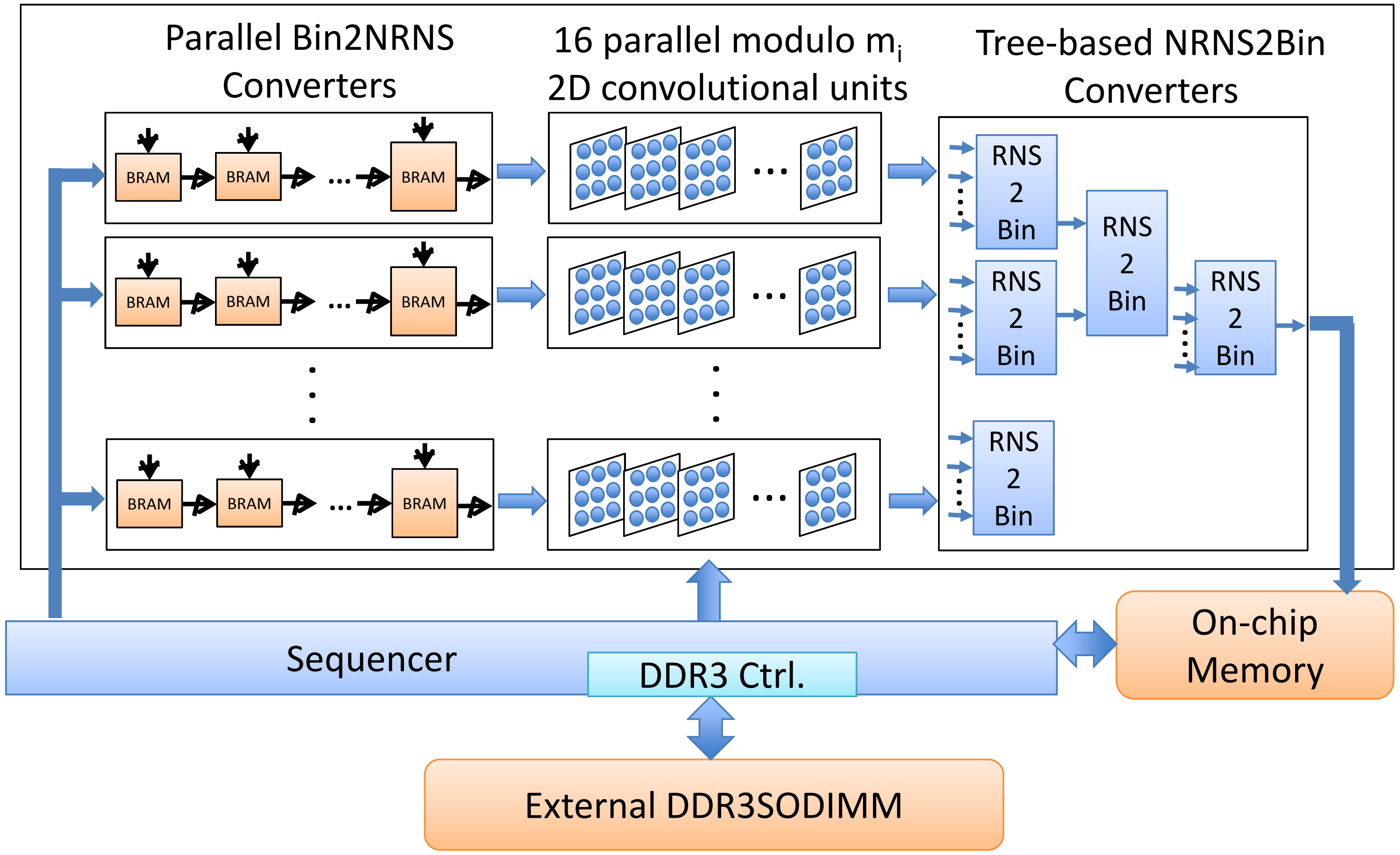
  $<3,4,5,7,11,13,<3,4,5,7,11,13>_{17}>_{23},$

  $<3,4,5,7,11,13,<3,4,5,7,11,13>_{17}>_{29},$

  $\cdots, <3,4,5,7,11,13,<3,4,5,7,11,13>_{17}>_{83}>$

All the 48-bit MAC operations are decomposed into 4-bit ones

DCNN Architecture using the NRNS



Experimental Results

Implementation Setup

- FPGA board: Xilinx VC707
  - FPGA: Virtex7 VC485T
  - 1GB DDR3SODIMM
    (Bus@800MHz, 64 bit width)
- Realized the pre-trained ImageNet by Convnet2
  - 48-bit fixed precision
- Synthesis tool: Xilinx Vivado2014.1
  - Timing constrain: 400MHz

Comparison with Other Implementations

	Precision	Max. Freq. [MHz]	FPGA	Performance [GOPS]	Performance per area [GOPS/ Slice x 10^{-4}]
ASAP2009	16bit fixed	115	Viretex5 LX330T	6.7	1.3
PACT2010	--- fixed	125	Viretex5 SX240T	7.0	1.9
FPL2009	48bit fixed	125	Spartax3A DSP3400	5.3	2.2
ISCA2010	48bit fixed	200	Virtex5 SX240T	16.0	4.3
ICCD2013	--- fixed	150	Virtex6 LVX240T	17.0	4.5
FPGA2015	32bit float	100	Virtex7 VX485T	61.6	8.1
Proposed	48bit fixed	400	Virtex7 VX485T	**132.2**	**25.2**

Conclusion

- Realized the DCNN on the FPGA
  - Time multiplexing
  - Nested RNS
    - MAC operation is realized by small LUTs
    - Functional decomposition are used as follows:
      - Bin2NRNS converter is realized by BRAMs
      - NRNS2Bin converter is realized by DSP blocks and BRAMs
- Performance per area (GOPS/Slice)
  - 5.86 times higher than ISCA10's