International Conference on
Mechanical, Electrical and Medical Intelligent System

Nov. 29, 2017 (Wed)

Study on Digital Multiplier Architecture Using Square Law and Divide-Conquer Method

Yifei Sun, Shu Sasaki, Dan Yao,
Nobukazu Tsukiji and Haruo Kobayashi

Gunma University
Division of Electronics and Informatics

t172d004@gunma-u.ac.jp

群馬大学
GUNMA UNIVERSITY

Kobayashi Laboratory

OUTLINE

- Research Background
- Multiplication Algorithm using Square Law
- Divide & Conquer Method
- RTL Design and Simulation
- Conclusion

OUTLINE

- Research Background
- Multiplication Algorithm using Square Law
- Divide & Conquer Method
- RTL Design and Simulation
- Conclusion

Research Background

Digital arithmetic devices
- Adder
- Subtractor
- Multiplier
- Divider

DSPs, μ Processors use several digital multipliers on a chip.

Requirements

Small scale
Low power
High speed

- Digital multiplier hardware implementation algorithm has been a research topic for 50 years.

- Decrease of the multiplier scale is still a research topic .

How Digital Multiplier Works

Decimal

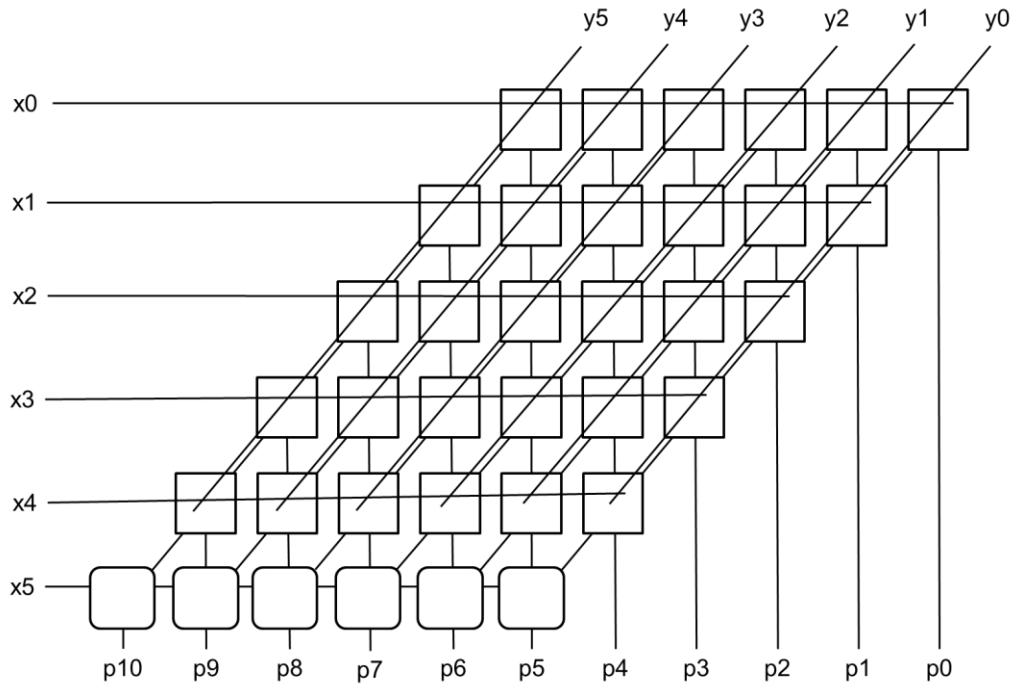
```
    2 5    multiplicand
  x 3 9    multiplicator
  -------
    4 5
   1 8      Partial products
  1 5
+ 6
  -------
  9 7 5     product
```

Binary

```
      0 1 1 0 0 1 : 25(10)   multiplicand
    x 1 0 0 1 1 1 : 39(10)   multiplicator
    -----------------------
      0 1 1 0 0 1
    0 1 1 0 0 1
  0 1 1 0 0 1                Partial products
0 0 0 0 0 0
0 0 0 0 0 0                  AND gate
+ 0 1 1 0 0 1
-----------------------------------------
0 0 1 1 1 1 0 0 1 1 1 1 : 975(10)   product
```

Calculation of the sum of partial products increases

Purpose of Study



Composition of array digital multiplier

The multiplier can be implementation in two dimensions by adder

Multiplier (Using square array of full adders)

- Circuit size
- Power
- Computation time

→ Big

Ex: In 6bit ✕ 6bit situation

6 ✕ 6 = 64 full adders are needed

↓

Reduce

circuit size ・ power ・ computation time

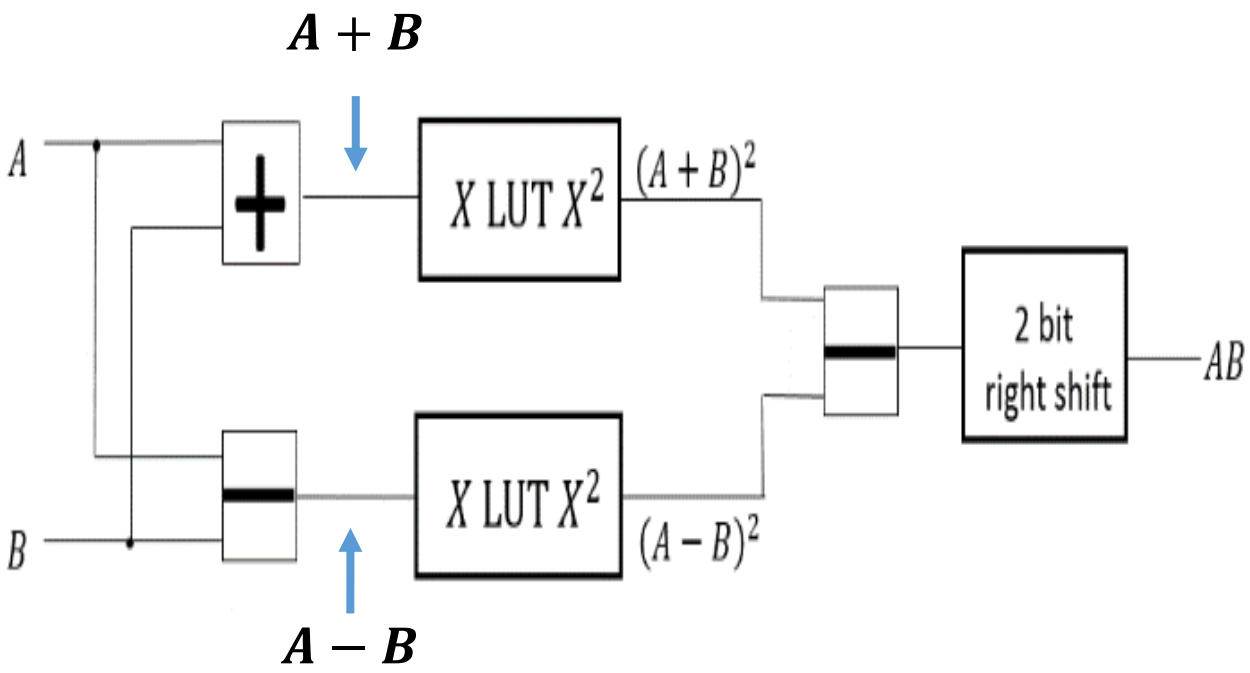
OUTLINE

- Research Background
- Multiplication Algorithm using Square Law
- Divide & Conquer Method
- RTL Design and Simulation
- Conclusion

Investigated Multiplier Algorithm①

Based on square law

$$AB = \frac{1}{4}[(A+B)^2 - (A-B)^2] \quad ①$$



Realization circuit

- Squaring 2 times
- Addition once
- Subtraction twice
- $\frac{1}{4}$ operation can be realized
  with 2-bit left shift
  or just interconnection change

General Multiplier Algorithm

$$AB = A + A + \cdots + A$$

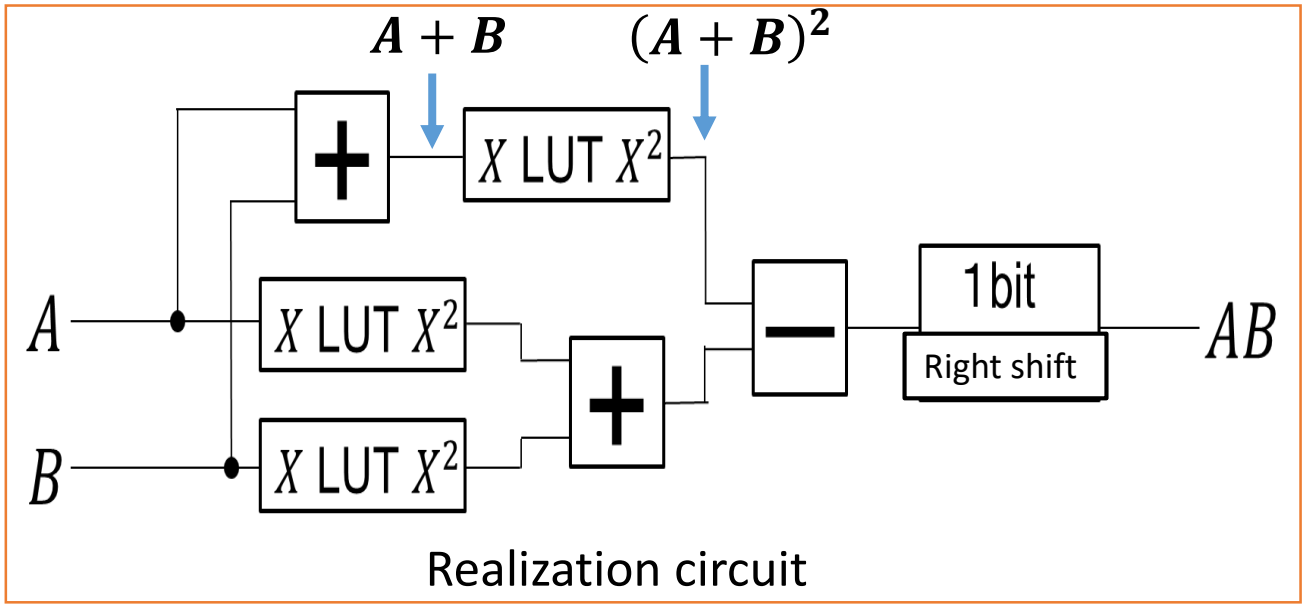
Multiplier A×B

Number of additions : B times

Investigated Multiplier Algorithm②

Based on square law

$$AB = \frac{1}{2}[(A+B)^2 - (A^2 + B^2)] \quad ②$$

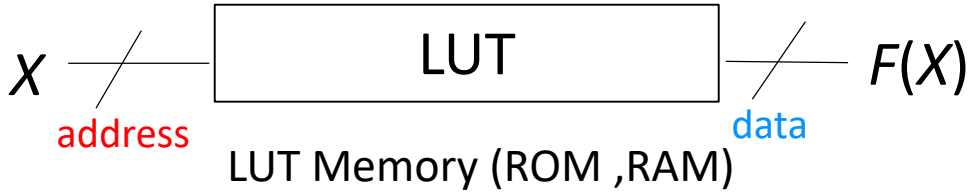


Realization circuit

- Squaring 3 times
- Addition twice
- Subtraction once
- $\frac{1}{2}$ operation can be realized
  with 1-bit left shift
  or just interconnection change

What is Look Up Table (LUT)

A → A^2



Address	Memory
1	1
2	4
3	9
100	10000

No calculation

Data
1
4
9
10000

Using LUT → Memory reference processing

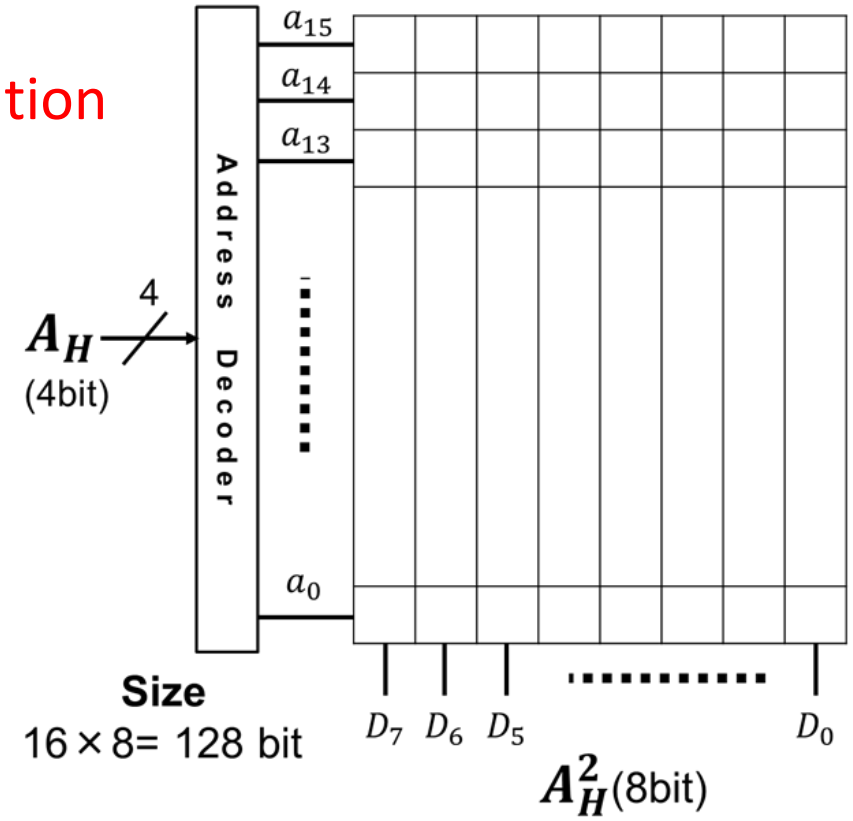
Efficient

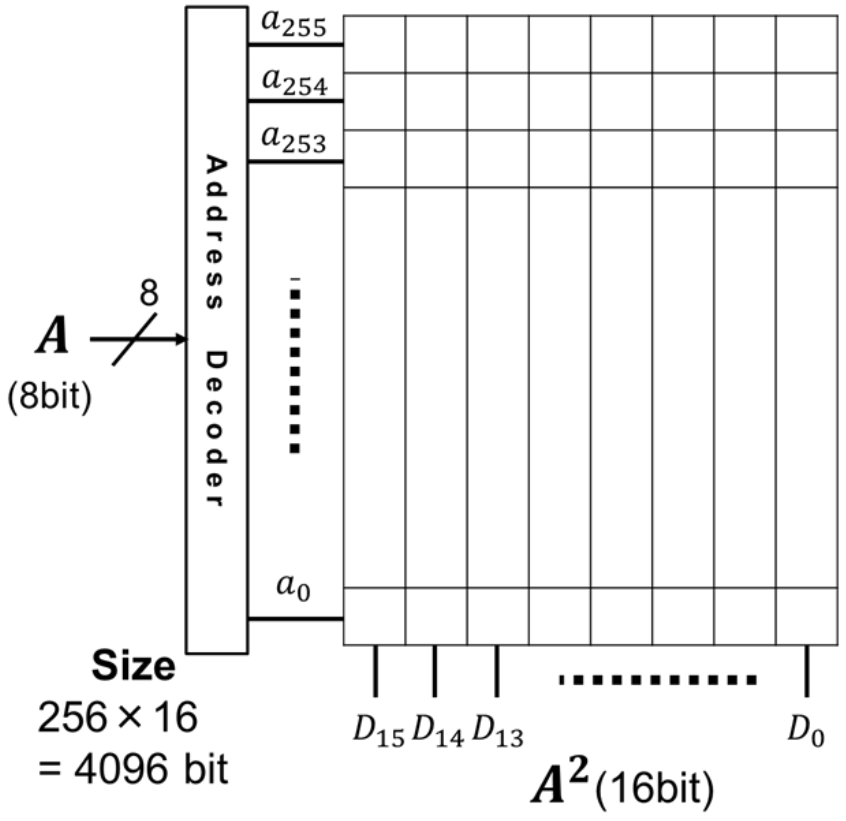
Disadvantage

LUT processing ⇒handled large number of bits ⇒ Large circuit size

Number of Bits Handled by LUT

4 bit situation

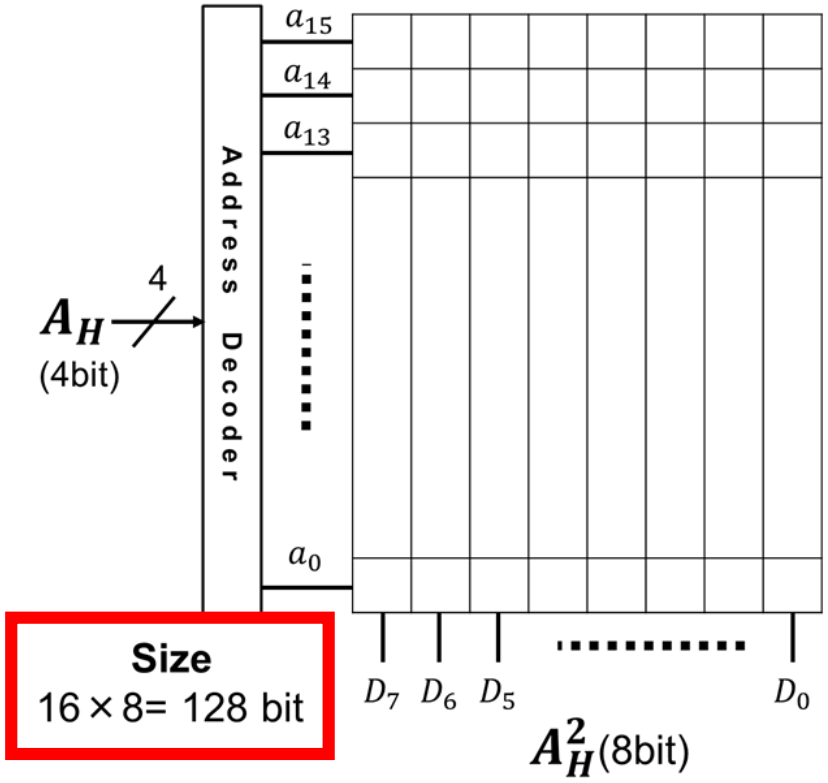


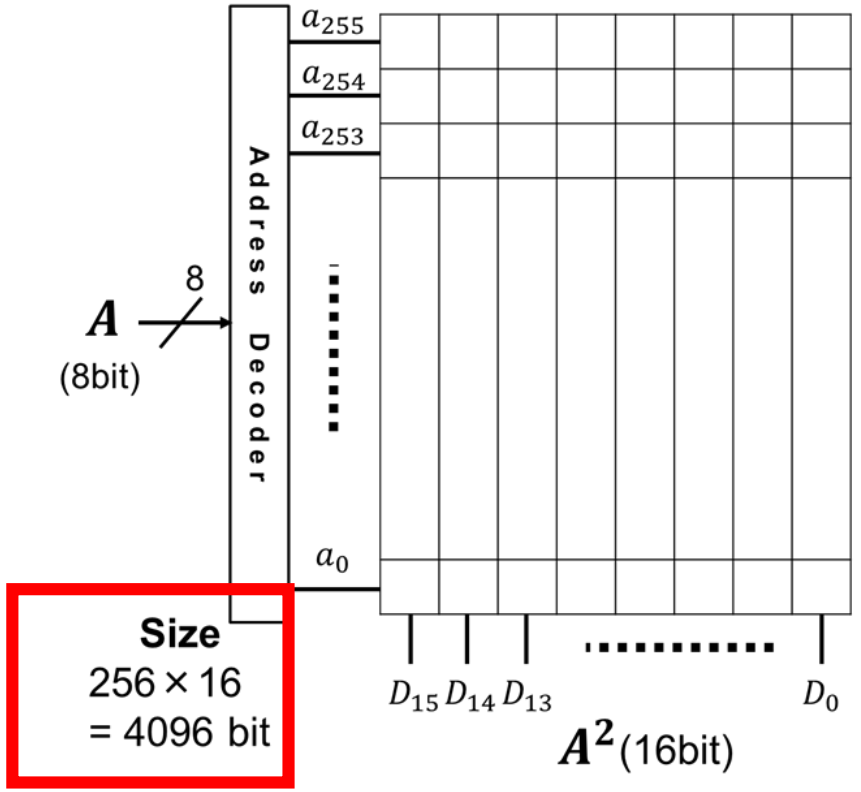


8 bit situation

Number of input bits is reduced by 1/2 ➡ LUT size is reduced by 1/32

Do Not use LUT to Implement Square Law





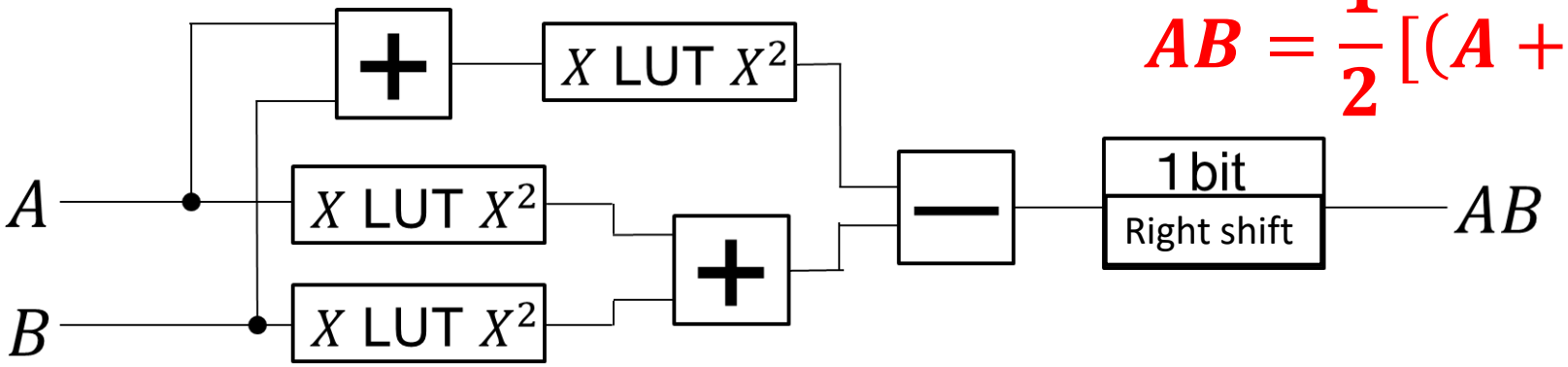
Use LUT to realized squaring calculation ➡ if lager number of bits was handled, memory size must be increased

⬇

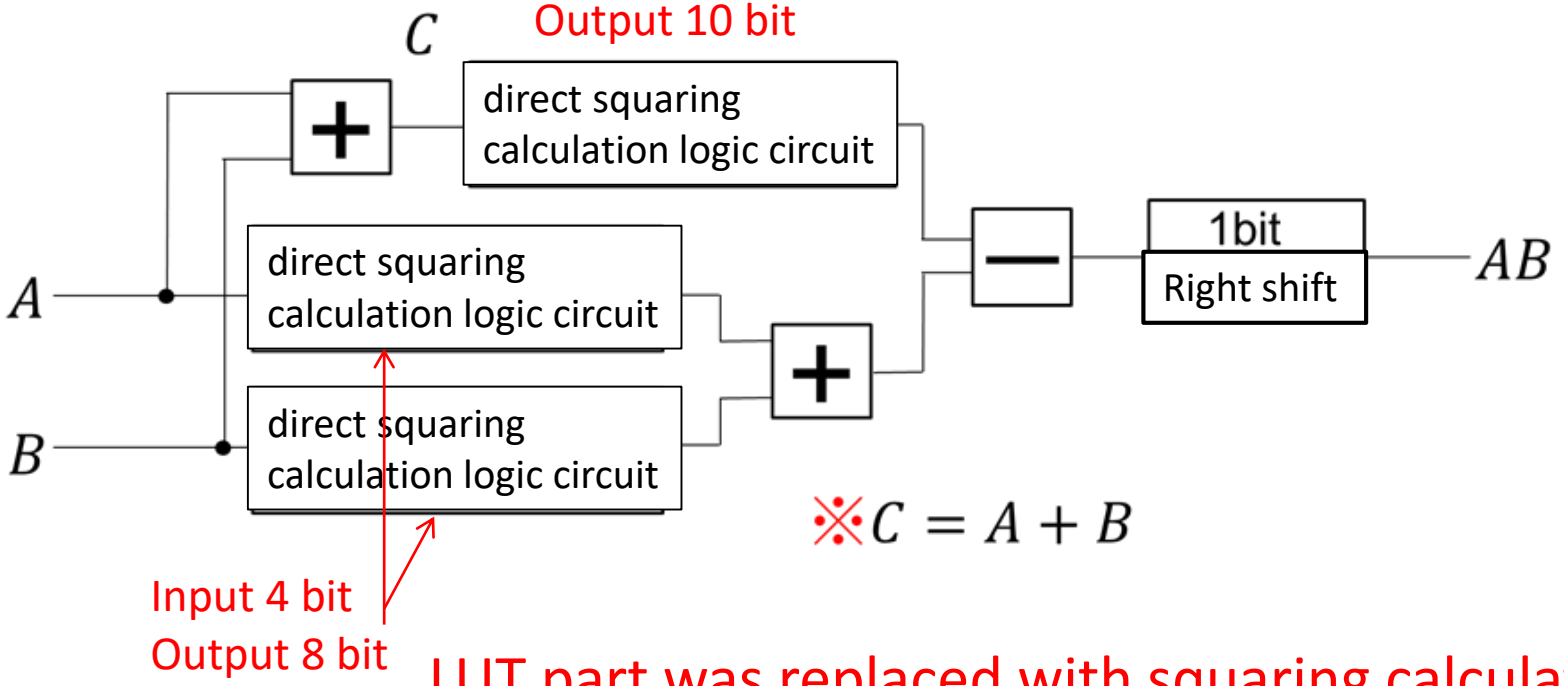
We have found that direct logic circuit Implementation of squaring can be simple.

Direct Squaring Calculation Logic Circuit

$$AB = \frac{1}{2}[(A+B)^2 - (A^2 + B^2)] \text{ ②}$$



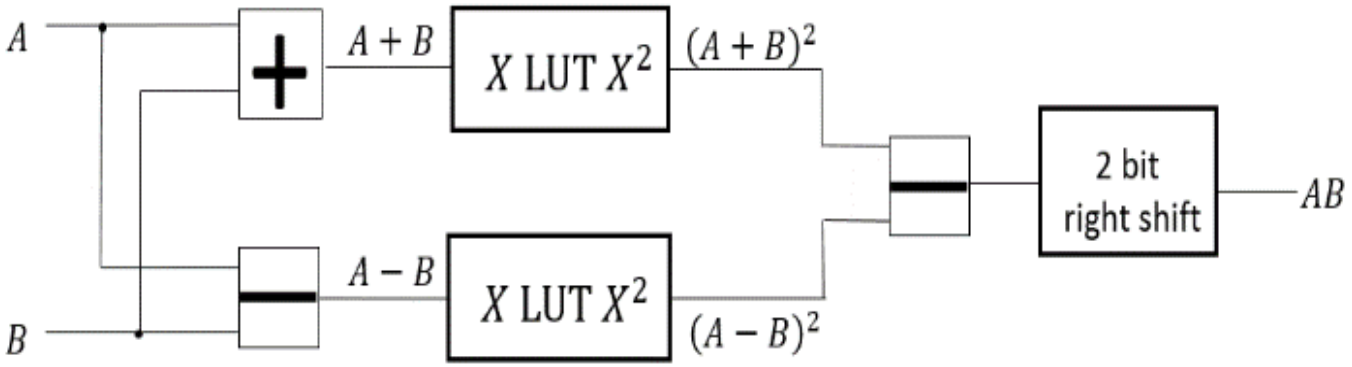
Input 5 bit
Output 10 bit



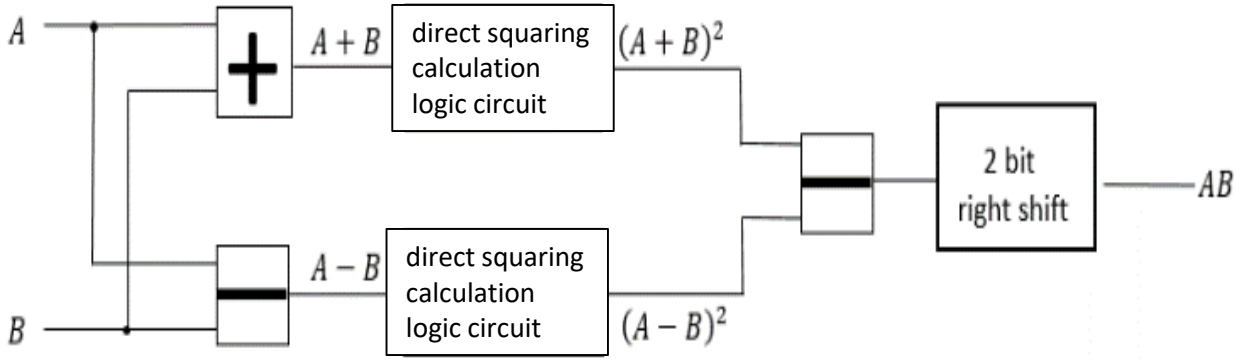
LUT part was replaced with squaring calculation circuit.

Direct Squaring Calculation Logic Circuit

$$AB = \frac{1}{4}[(A+B)^2 - (A-B)^2] \quad ①$$



Using LUT to realize square



Using direct squaring calculation logic circuit

Truth Table and Logic Expression

Input | Output

	I3	I2	I1	I0		O7	O6	O5	O4	O3	O2	O1	O0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0	0	0	1
2	0	0	1	0	4	0	0	0	0	0	1	0	0
3	0	0	1	1	9	0	0	0	0	1	0	0	1
4	0	1	0	0	16	0	0	0	1	0	0	0	0
5	0	1	0	1	25	0	0	0	1	1	0	0	1
6	0	1	1	0	36	0	0	1	0	0	1	0	0
7	0	1	1	1	49	0	0	1	1	0	0	0	1
8	1	0	0	0	64	0	1	0	0	0	0	0	0
9	1	0	0	1	81	0	1	0	1	0	0	0	1
10	1	0	1	0	100	0	1	1	0	0	1	0	0
11	1	0	1	1	121	0	1	1	1	1	0	0	1
12	1	1	0	0	144	1	0	0	1	0	0	0	0
13	1	1	0	1	169	1	0	1	0	1	0	0	1
14	1	1	1	0	196	1	1	0	0	0	1	0	0
15	1	1	1	1	225	1	1	1	0	0	0	0	1

O3 In the situation of output equal to 1 , the input are 0011 , 0101 , 1011 , 1101

$$O3 = \bar{I3}\bar{I2}I1I0$$
$$O3 = \bar{I3}I2\bar{I1}I0$$
$$O3 = I3\bar{I2}I1I0$$
$$O3 = I3I2\bar{I1}I0$$

Write in a theoretical way simplification

$$O3 = (I2 \oplus I1)I0$$

EXOR

$$O0 = I0$$
$$O1 = 0$$
$$O2 = I1\bar{I0}$$
$$O3 = (I2 \oplus I1)I0$$
$$O4 = \bar{I3}I2(\bar{I1} + I0) + I3\bar{I2}I0 + I3I2\bar{I1}\bar{I0}$$
$$O5 = (I3 \oplus I2)I1 + I3I2I0$$
$$O6 = I3\bar{I2} + I3I2I1$$
$$O7 = I3I2$$

Calculate logic expression O0~O7

Usage of Absolute Value for Squaring Calculation

Consider to handle negative numbers for the multiplier

$$AB = \frac{1}{4}[(A+B)^2 - (A-B)^2] \quad ①$$

Convert negative number to its absolute value

⬇

Two's complement

A or B are 3 bit situation

unsign	sign	binary
0	0	000
1	1	001
2	2	010
3	3	011
4	-4	100
5	-3	101
6	-2	110
7	-1	111

A+B or A-B are 4 bit situation

unsign	sign	binary	unsign	sign	binary
0	0	0000	8	-8	1000
1	1	0001	9	-7	1001
2	2	0010	10	-6	1010
3	3	0011	11	-5	1011
4	4	0100	12	-4	1100
5	5	0101	13	-3	1101
6	6	0110	14	-2	1110
7	7	0111	15	-1	1111

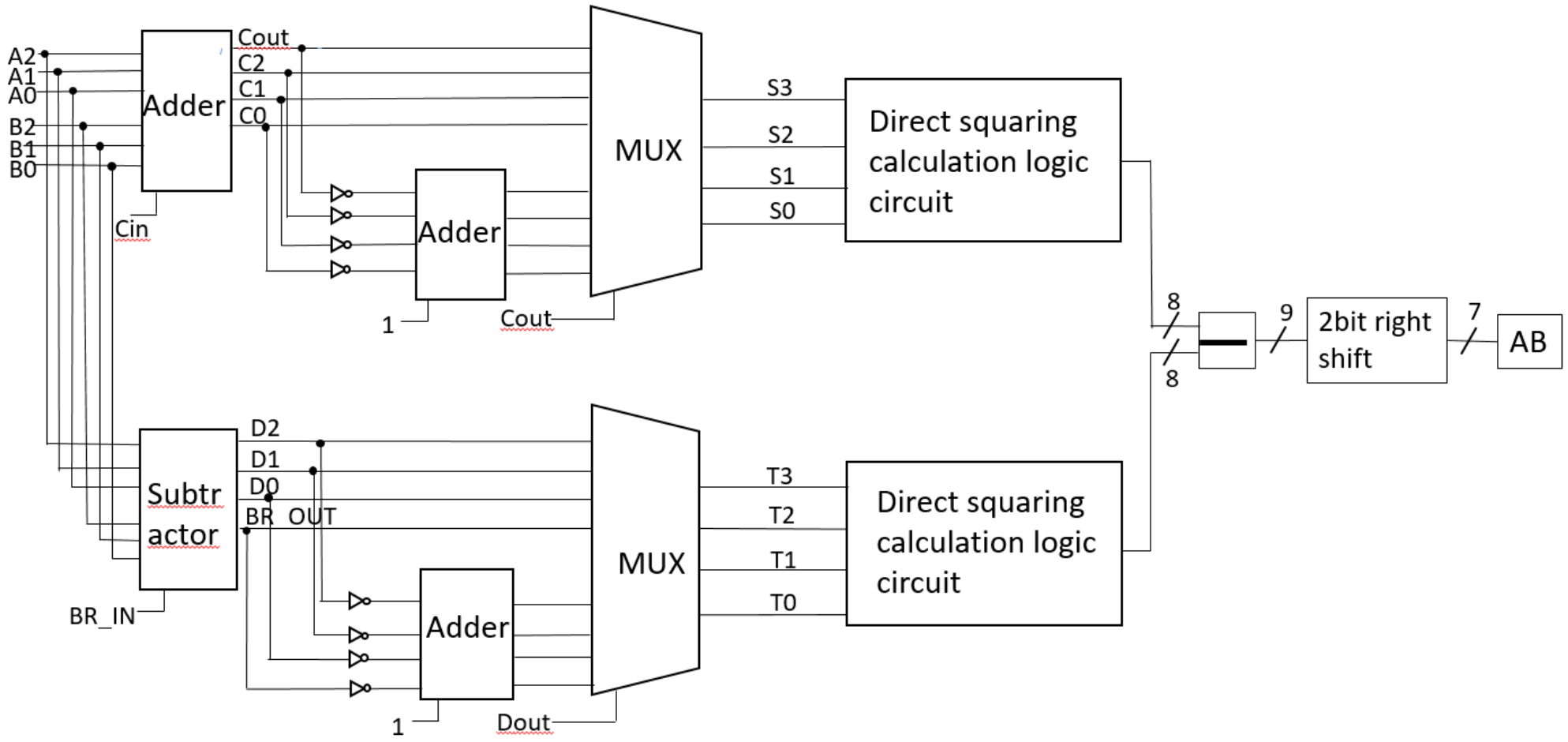
$A + B = C$
$A - B = D$

$-8 \le C \le 6$
$-7 \le D \le 7$

C or D

- $C \ge 0$ direct squaring calculation logic circuit
- $C \le -1$ reversal C in every bit ➡ plus 1 ➡ obtain $|C|$ than realizes direct squaring calculation logic circuit

Circuit Realization of Absolute Value for Squaring Calculation



This structure reduces the hardware whether it were implemented with LUTs or dedicated logic

OUTLINE

- Research Background
- Multiplication Algorithm using Square Law
- Divide & Conquer Method
- RTL Design and Simulation
- Conclusion

Improvement Plan of Implementation Circuit

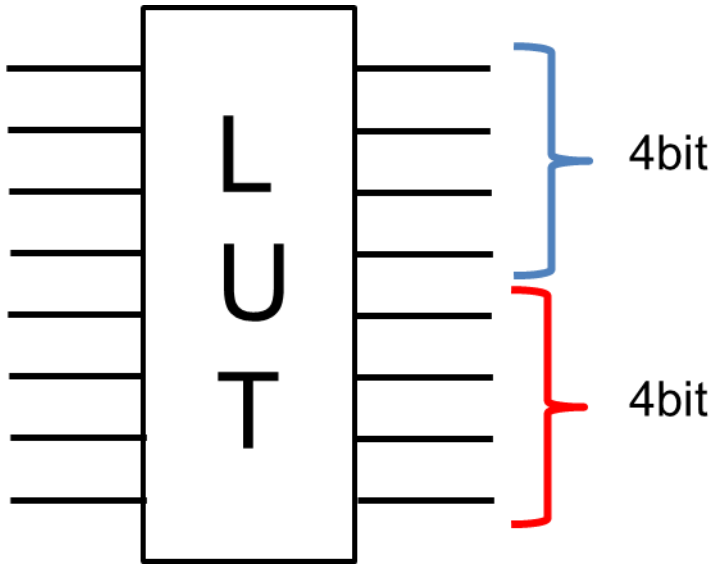
Come up with divide & conquer method

Cut LUT size

$$AB = \frac{1}{2}[(A+B)^2 - (A^2 + B^2)] \quad ②$$

A , *B* , *A+B* divide into upper bits and lower bits for calculation scale reduction

In 8 bit case : divide into
- upper 4 bit
- lower 4 bit



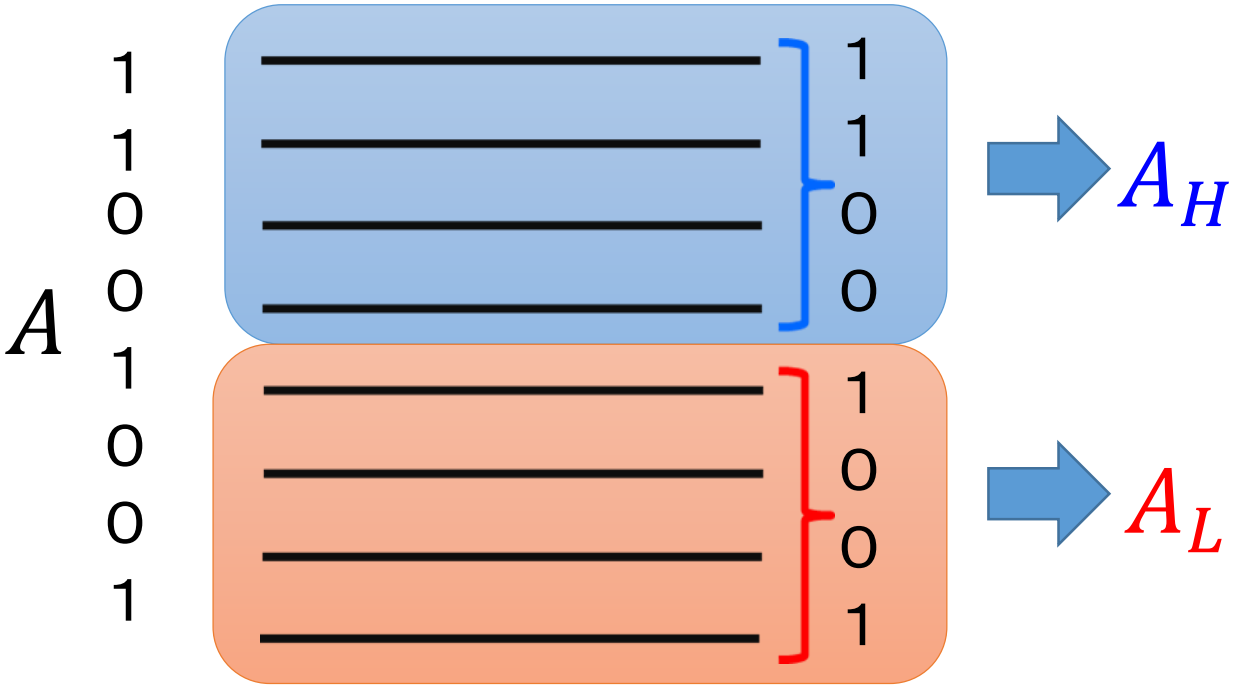
LUT size becoming smaller

Divide & Conquer Method Analysis

In 8 bit case $(A = 11001001 : 201_{10})$

8bit x 8bit divide 4bit $[A_H]$, $[A_L]$

Calculated by each $[A_H]$, $[A_L]$



$$A = 11001001$$

Divided input, output values up and down

$$A = 11001001$$

$$A_H = 1100 : 12_{10}$$

$$A_L = 1001 : 9_{10}$$

Conquer

$$A_H^2 = 10010000 : 144_{10}$$

$$A_L^2 = 1010001 : 81_{10}$$

$$A_H A_L = 1101100 : 108_{10}$$

Divide & Conquer Method Analysis

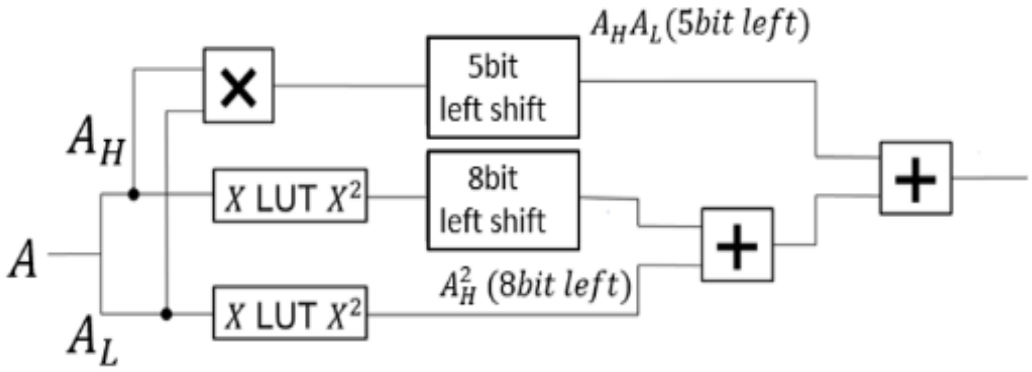
First method ➡

$$A^2 = A_H^2(8bit\ left\ shift) + 2A_HA_L(4bit\ left\ shift) + A_L^2$$

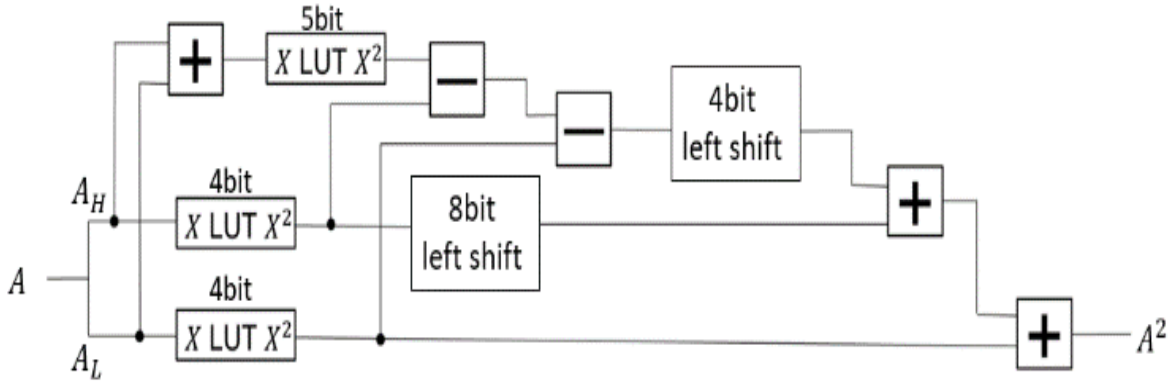
$$2A_HA_L = [(A_H + A_L)^2 - {A_H}^2 - {A_L}^2]$$

$$A^2 = A_H^2(8bit\ left\ shift) + \{(A_H + A_L)^2 - {A_H}^2 - {A_L}^2\}(4bit\ left\ shift) + A_L^2$$

⬆ Second method



First method
Realization circuit



Second method
Realization circuit

Divide & Conquer Method Analysis

$$A^2 = A_H^2(8bit\ left\ shift) + A_H A_L(5bit\ left\ shift) + A_L^2$$

$$A = 11001001 = (201)_{10}$$

$A_H^2 = 10010000 = (144)_{10}$

$A_H A_L = 1101100 = (108)_{10}$

$A_L^2 = 1010001 = (81)_{10}$

$A^2 = 1001110111010001 : 40401_{10}$

$(A^2 = 201 \times 201 = 40401)$

$A_H^2(8bit\ left\ shift) = 1001000000000000 = (36864)_{10}$

$A_H A_L(5bit\ left\ shift) = 1101100000000 = (3456)_{10}$

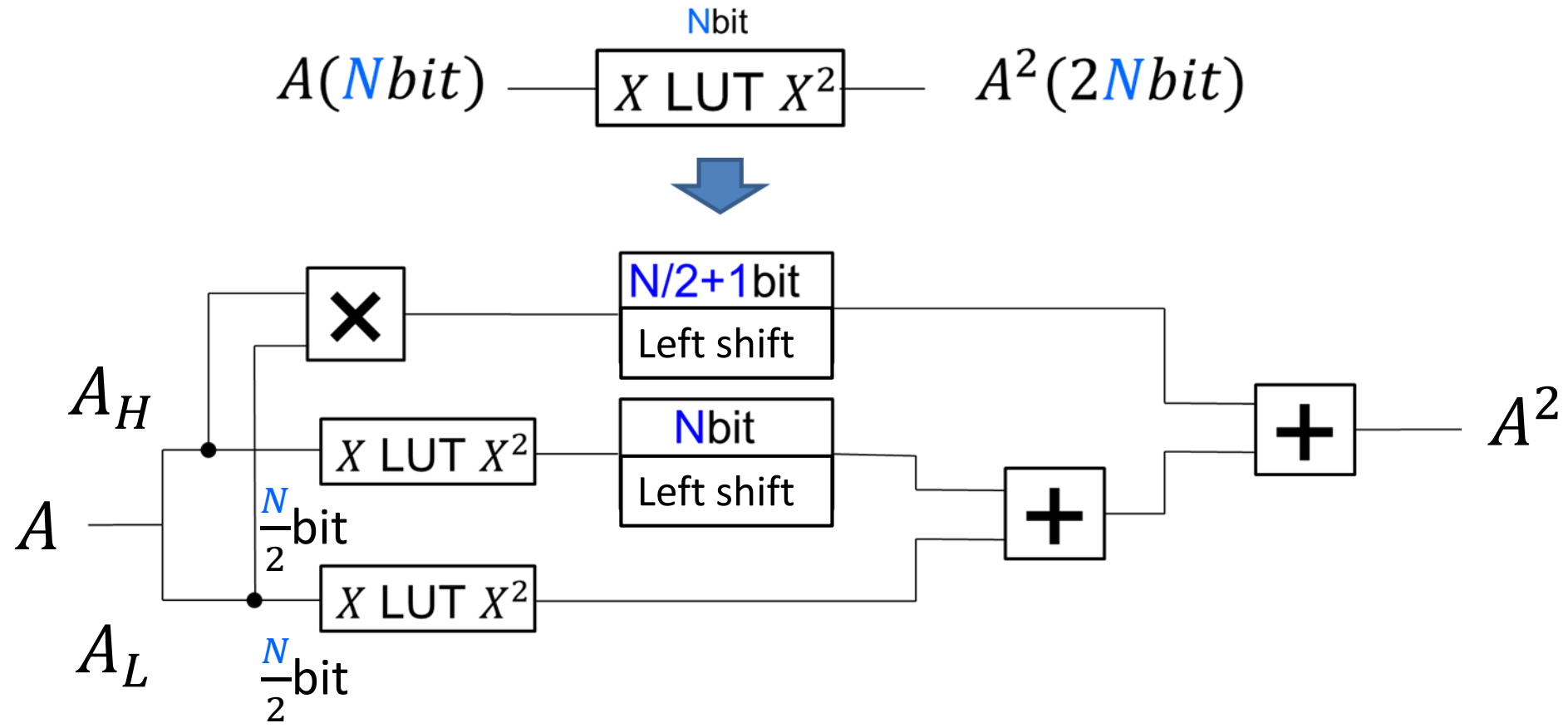
$A_L^2 = 1010001 = (81)_{10}$

First method using Divide & Conquer

$$A^2 = 36864 + 3456 + 81 = 40401$$

The value obtained by the Divide & Conquer method and the direct calculated value of square of A are the same

Divide & Conquer Method Circuit



$$A^2 = A_H^2(Nbit\ left\ shift) + A_H A_L\left((\frac{N}{2}+1)bit\ left\ shift\right) + A_L^2$$

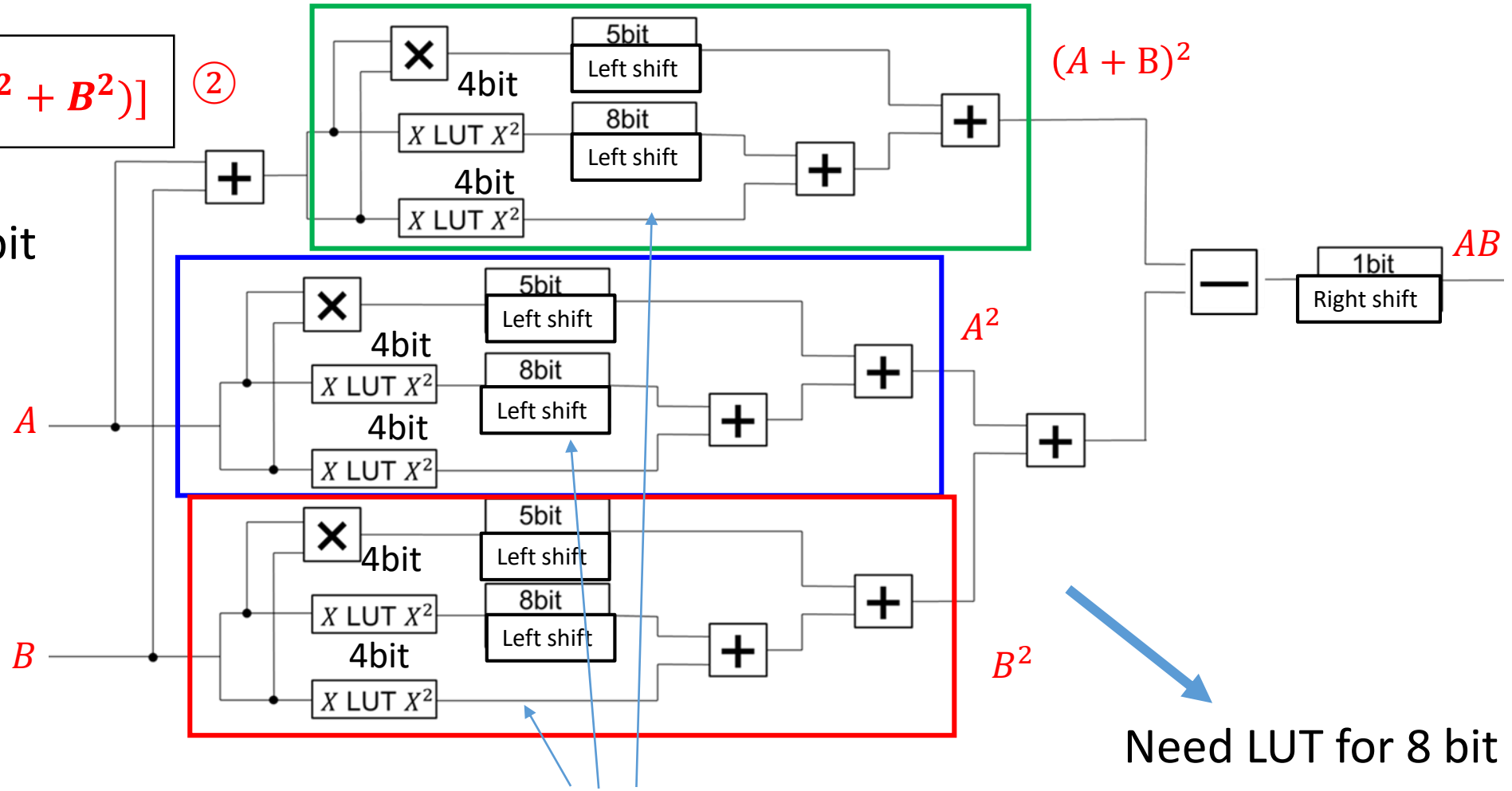
Using divide & conquer with X times , LUT size will decrease 2^X times

Divide & Conquer Method Circuit (8 bit case)

$$AB = \frac{1}{2}[(A+B)^2 - (A^2 + B^2)]$$

②

Need LUT for 16 bit



Need LUT for 8 bit

By dividing ,
number of LUT bits ➡ smaller

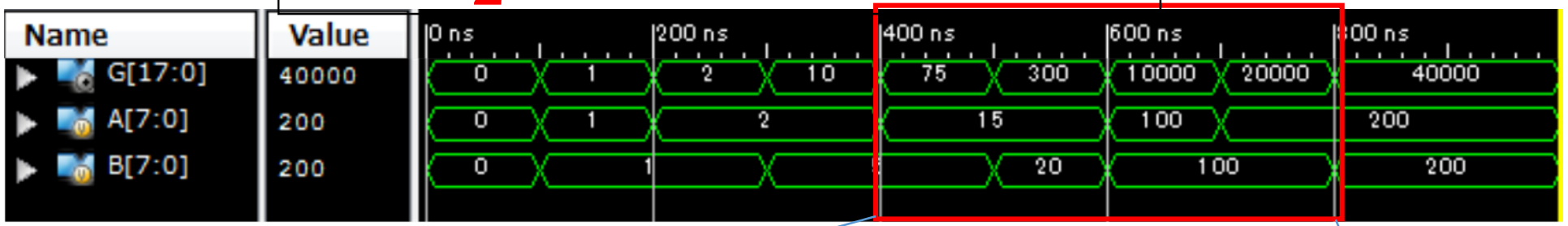
OUTLINE

- Research Background
- Multiplication Algorithm using Square Law
- Divide & Conquer Method
- RTL Design and Simulation
- Conclusion

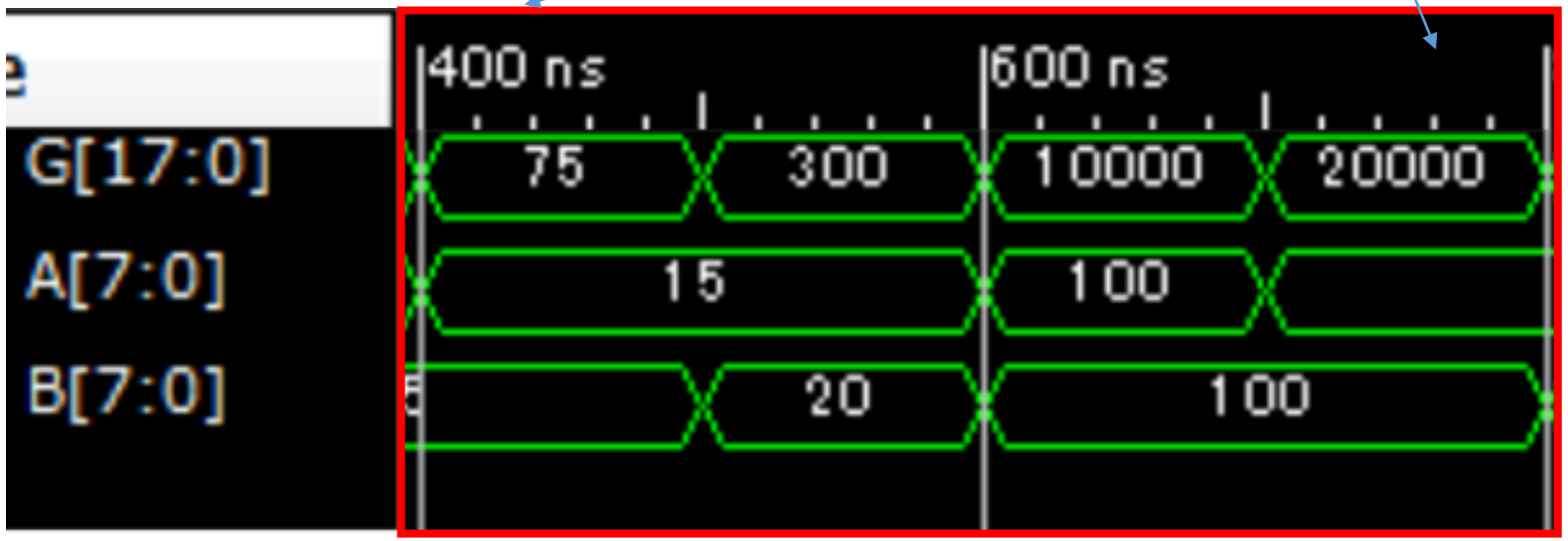
RTL: Register Transfer Level

1.RTL Simulation using Second Divide & Conquer Method

$$AB = \frac{1}{2}[(A+B)^2 - (A^2 + B^2)] \quad ②$$



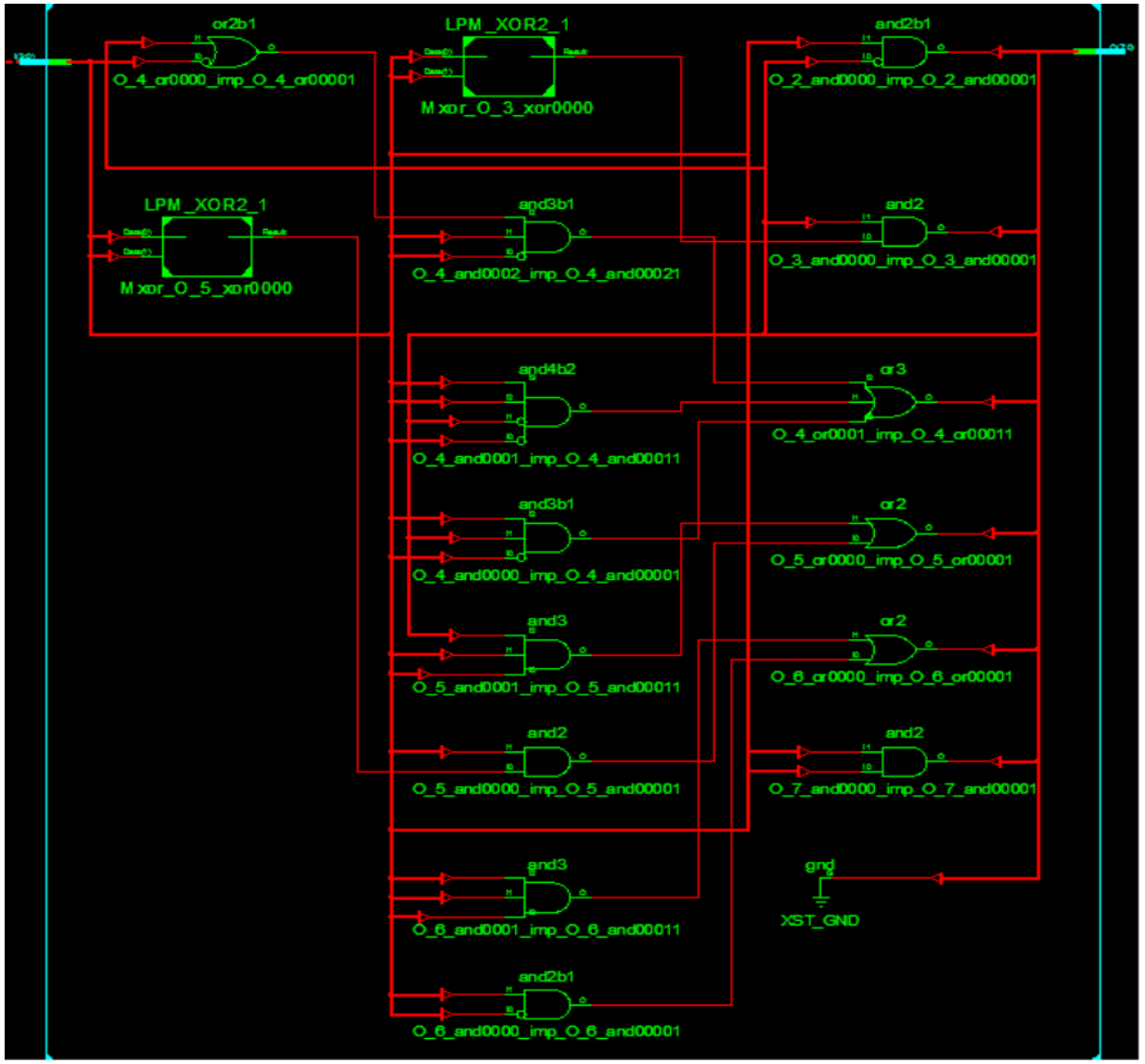
8bit ×8bit

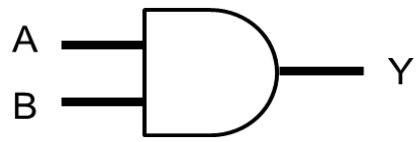


Input values A, B are changed every 100 ns and 200 ns.
A, B: input
G : output.

G = A × B

2.Layout of Direct Squaring Calculation Logic Circuit





AND gate

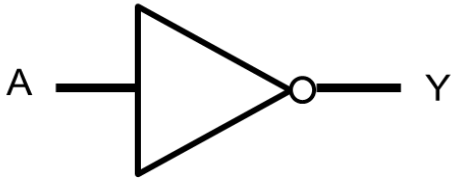
Inout		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

Inout		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

EXOR gate

Inout		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



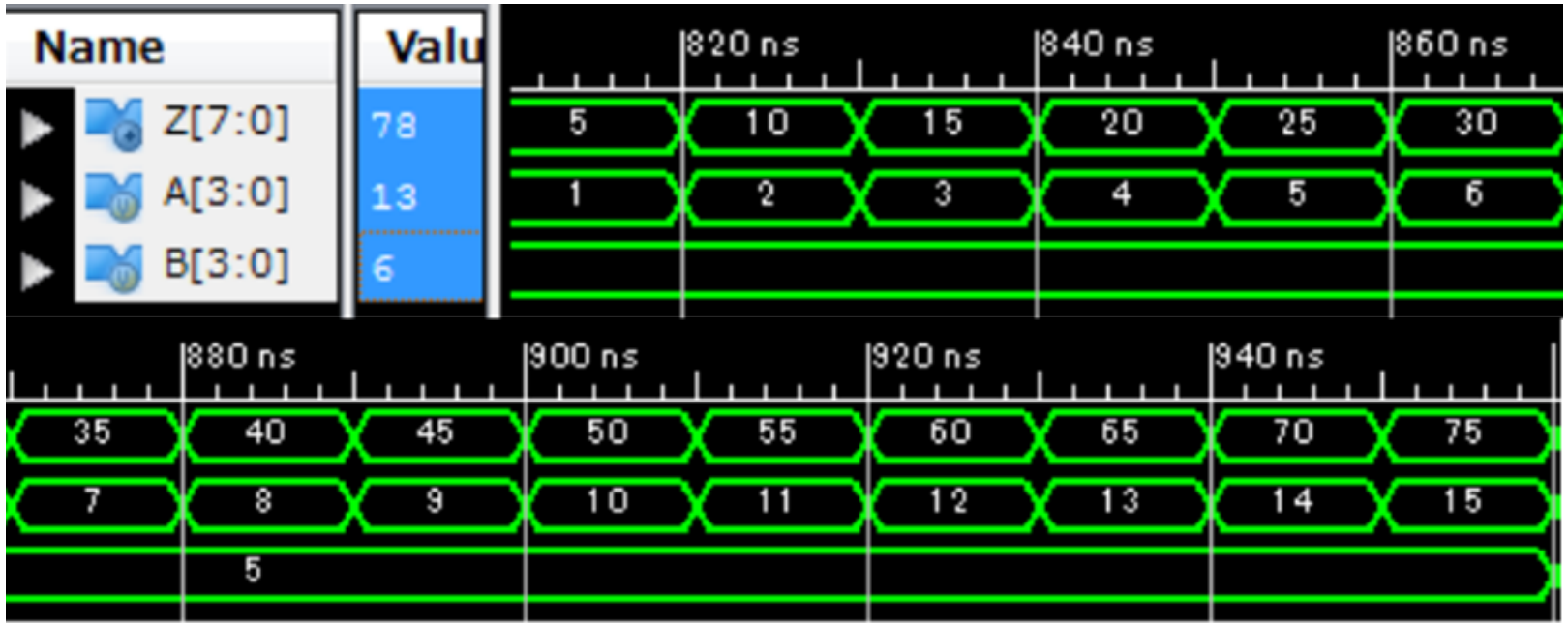
NOT gate

Inout	Output
A	Y
0	1
1	0

This Circuit creates individual logic expressions by the number of bits of input

2.RTL Simulation using Direct Method

4bit × 4bit

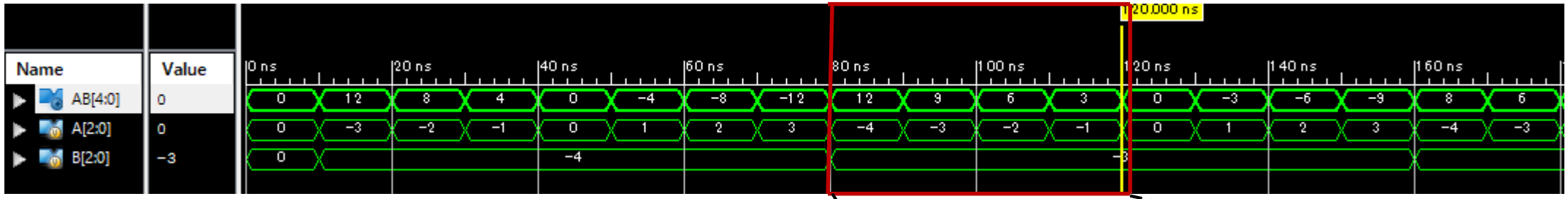


Input 4 bit × 4bit circuit

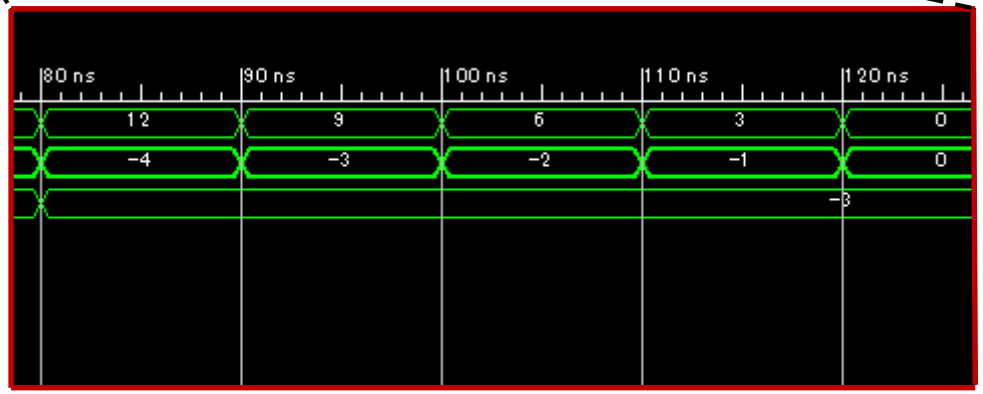
Input values A, B are changed every 10 ns and 160 ns.
A, B: input
Z : output.

Using direct squaring calculation logic circuit was validated.

3.RTL Simulation using Absolute Value



3bit × 3bit



AB=A × B

Input values A, B are changed every 10 ns and 70 ns.
A B: input
AB: output

OUTLINE

- Research Background
- Digital Multiplier Algorithm
- Multiplication Algorithm using Square Law
- RTL Design and Simulation
- Conclusion

Conclusion

- Discussed multiplication algorithms based on square law
- Proposed divide & conquer method to reduce LUT size in RTL level validation by simulation
  - ➡ reduce computation & circuit size
- Considered reduction of multiplication using squaring calculation logic in RTL level validation by simulation
  - ➡ create dedicated circuit to calculate square simple
- Consider to handle negative numbers for the multiplier in RTL level validation by simulation

Thanks for your listening

Q and A

1. You have investigated the multiplication algorithm, or multiplier algorithm. Can you extend this algorithm to divide or division algorithm?
Answer: I have not consider use Divide & Conquer method to using division algorithm yet. Using Divide & Conquer method may be also can reduce the LUT size in division algorithm. I will consider it in the future.

2. You have improve the speed of the circuit square calculation, could you tell me some limitation of your method?
Answer: For a large number of N, the LUT size is large and its speed may be slow. For a small number of N, its size is reduced significantly and also its access speed may be much faster.